

Book Collection

Learning Path

Deployment of Microsoft Azure Cloud Solutions

A complete guide to cloud development
using Microsoft Azure

Florian Klaffenbach, Jan-Henrik Damaschke,
Oliver Michalski and Ritesh Modi

Packt>

www.packt.com



Deployment of Microsoft Azure Cloud Solutions

A complete guide to cloud development using
Microsoft Azure

Florian Klaffenbach
Jan-Henrik Damaschke
Oliver Michalski
Ritesh Modi



BIRMINGHAM - MUMBAI

Deployment of Microsoft Azure Cloud Solutions

Copyright © 2018 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: December 2018

Production reference: 1181218

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-78995-405-0

www.packtpub.com



mapt.io

Mapt is an online digital library that gives you full access to over 5,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why Subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Mapt is fully searchable
- Copy and paste, print, and bookmark content

Packt.com

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.Packt.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packt.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

Contributors

About the authors

Florian Klaffenbach started in 2004 with his IT carrier as 1st & 2nd Level IT Support Technician and IT Salesman Trainee for a B2B online shop. After that he changed to a small company working as IT Project Manager, planning, implementing & integration from industrial plants & laundries into enterprise IT. After spending some years, he changed his path to for Dell Germany. There he started from scratch as Enterprise technical Support Analyst and later worked on a project to start Dell technical Communities and support over social Media in Europe and outside of the U.S.

Currently he is working as Technology Solutions Professional for Microsoft specialized on Hybrid Microsoft Cloud Infrastructure. Additionally, to his job engagement he is active as Microsoft Blogger and Lecturer. He blogs for example on his own page Datacenter-Flo.de or Brocade Germany Community. Together with a very good friend, he founded the Windows Server User Group Berlin to create network of Microsoft ITPros in Berlin.

Florian is maintaining a very tight network to many vendors like Cisco, Dell or Microsoft and Communities. That helps him to grow his experience and to get the best out of a solution for his customers. Since 2016 he is also Co-Chairman of the Azure Community Germany. In April 2016 Microsoft awarded Florian as Microsoft Most Valuable Professional for Cloud and Datacenter Management. In 2017 after joining Microsoft, Florian became MVP reconnect member.

Jan-Henrik Damaschke is an IT Consultant for Security, Network, and Infrastructure from Germany. He was MVP awarded in the categories of Enterprise Security, PowerShell, and AzureStack. PKI implementation and management is one of his core competencies as well as cloud-related security. He writes articles on security-related topics and is involved in many community events as a speaker as well as an organizer. He is passionate about sharing knowledge with others. For this purpose, he is member of the Microsoft Student Partner program and is engaged on forums and on other platforms. Currently, he is writing a book on Azure infrastructure solutions.

Oliver Michalski started in 1999 with his IT career as a Web Developer. Now, he is a Senior Software Engineer for Microsoft .NET and an SOA Architect. He also works as an Independent Enterprise Consultant in the field Microsoft Azure. When he started in 2011 with Microsoft Azure, there was no Azure Community on the German market. Therefore, Oliver founded the Azure Community Germany (ACD). Oliver is Chairman of the Azure Community Germany, and since April 2016 he has been a Microsoft Most Valuable Professional for Microsoft Azure.

Ritesh Modi is an ex Microsoft senior technology evangelist and Microsoft regional lead. He has worked on Ethereum and Solidity, extensively helping and advising companies. Ritesh is a regular speaker on blockchain and Solidity at conferences and local meetups. He is an architect, evangelist, speaker, and a known leader for his contributions toward blockchain, data centers, Azure Bots, cognitive services, DevOps, Artificial Intelligence, and automation. He is the author of five books.

About the reviewers

Sebastian Durandeu is a senior software engineer with a primary focus on building applications and services for the Cloud using Microsoft technologies. He works at Southworks, a high-end software development company that helps businesses leverage the latest technologies. There he has worked closely with several Microsoft divisions helping the developer community adopt emerging technologies using recommended practices. Sebastian currently lives in Buenos Aires, Argentina where he is an active contributor in the developer community.

Rafael dos Santos Christian, Geysla's husband, Ted's father, Entrepreneur, Developer, Cloud specialist. 10 years experience developing and architecting web based solutions.

Paul Glavich has been an ASP.NET MVP for 13 years and currently works as a principal consultant for Readify. Previously, he was the Chief Technology Officer (CTO) for Saasu, a solution architect at Datacom, then the senior consultant for Readify, and prior to that was a technical architect for EDS Australia. He has over 20 years of industry experience ranging from PICK, C, C++, Delphi, and Visual Basic 3/4/5/6 to his current specialty in .NET with C#, ASP.NET, Azure, Cloud, and DevOps.

Paul has been developing in .NET technologies since .NET was first in beta, and was a technical architect for one of the world's first internet banking solutions using .NET technology. Paul can be seen on various .NET-related newsgroups, has presented at the Sydney .NET user group and TechEd, and is also a member of ASPInsiders. He has also written some technical articles, which can be seen on community sites such as ASP Alliance. Paul has authored a total of three books, Beginning AJAX in ASP.NET, Beginning Microsoft ASP.NET AJAX, and the latest book on .NET Performance Testing and Optimisation. He is currently focusing on overall architecture, solution design, and Microsoft Cloud solutions. On a more personal note, Paul is married with three children, three grandkids, holds a 5th degree black belt in Budo-Jitsu and also practices Wing Chun Kung fu.

Vikram Pendse is a Microsoft MVP for Azure and has been a distinguished speaker at various Microsoft events over the last 10 years. He is a very active member of various Microsoft communities in India. He is a cloud solutions architect and is currently working with one of the leading Microsoft Partners in Pune, where he is responsible for building the strategy for moving Amazon AWS workloads to Azure, providing cloud-centric solutions, architecture, supporting RFPs, and global deliveries.

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit authors.packtpub.com and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Table of Contents

Preface	1
Chapter 1: Getting Started with Azure Implementation	7
Service Models	8
Deployment Models	9
Cloud Characteristics	11
Microsoft Azure	12
Azure Services Overview	12
Azure Basics	15
Azure Resource Manager	15
Resources	15
Azure Regions	16
Azure Portal	17
Azure Automation	17
Azure Automation Tools	17
REST APIs	18
Summary	18
Chapter 2: Azure Resource Manager and Tools	20
Understanding the Azure Resource Manager	21
Functionalities Provided by the Azure Resource Manager	21
Working with the Azure Resource Manager	22
Creating an Azure Resource Group	23
Adding a Resource to an Azure Resource Group	26
First Approach: Adding a Storage Account to your Azure Resource Group	26
Second Approach: Adding a Storage Account to your Azure Resource Group	29
Third Approach: Adding a Storage Account to your Azure Resource Group	31
Tagging in the Azure Resource Manager	40
Locking Azure resources	46
Azure Resource Locks	46
Working with ARM Templates	49
Exporting a Deployment as an ARM Template (for IT pros)	50
Example 1: Exporting a Resource Group to an ARM Template	51
Example 2: Exporting a Resource (from the Deployment History) to an ARM Template	53
Example 3: Exporting a Resource (Classic) to an ARM Template	55
Modifying an ARM Template	57

Authoring an ARM Template	63
Creating your own ARM Template (for Developers)	65
Azure Resource Explorer	72
Summary	74
Chapter 3: Deploying and Synchronizing Azure Active Directory	75
Azure Active Directory	76
Azure Active Directory Options	77
Azure AD Free	78
Azure AD Basic	78
Azure AD Premium P1	78
Deploying a Custom Azure AD	79
Adding Accounts and Groups to Azure AD	88
Installing Azure AD Connect: Prerequisites	112
Installing Basic Azure AD Connect Environment	115
Azure AD Connect High Available Infrastructure	130
Summary	133
Chapter 4: Implementing Azure Networks	134
Azure Networking Limits	135
Azure Networking Components	136
Azure Virtual Networks (VNet)	136
VNet Peering	139
Azure VPN Gateways	139
Azure Local Gateway	146
Azure ExpressRoute	147
Azure Global Points of Presence	150
National Azure Points of Presence	152
Azure Connections	155
Azure Routes	155
Azure Third-Party Network Devices	157
Azure Load Balancer	159
Hash-Based Distribution	160
Port Forwarding	160
Automatic Reconfiguration	160
Service Monitoring	161
Azure Application Gateways	162
Azure Traffic Manager	163
Azure DNS	164
Setting up Azure Networks	164
Setting up Azure VNet	165
Setting up Azure Virtual Network Site-to-Site VPN	171
Configuring Local Network Gateway	171
Configuring Azure Virtual Network Gateway	173
Configuring Connection Between Local and Virtual Network Gateways	179
Setting up Azure Virtual Network with MPLS and ExpressRoute	183
Configuring Azure Virtual Network Gateway	184

Configuring Azure ExpressRoute Circuit	185
Configuring Connection Between ExpressRoute Circuit and Azure Virtual Gateway	190
Setting up Azure VNet Peering	192
Preparing the Deployment	193
Configuring VNet Peering	195
Configuring Custom Routes	199
Configuring Third-Party Network Appliances	204
Common Azure Network Architectures	211
Summary	214
Chapter 5: Implementing and Securing Azure Storage Accounts	215
Storage Accounts	216
General Purpose Storage Account	218
Blob Storage Account	218
Replication and Redundancy	220
Locally Redundant Storage	220
Zone-Redundant Storage	220
Geo-Redundant Storage	221
Read-Access Geo-Redundant Storage	221
Azure Storage Services	223
Blob Storage Services	223
Table Storage Services	229
Queue Storage Services	231
File Storage Services	234
Access Keys	243
Exploring Azure Storage with Azure Storage Explorer	245
Premium Storage Account	249
Premium Storage Requirements	249
Pricing	249
Summary	250
Chapter 6: Planning and Deploying Virtual Machines in Azure	251
Azure Virtual Machine Types	252
A-series Virtual Machines	253
D-series and DS-series Virtual Machines	255
F-series and FS-series Virtual Machines	257
G-series and GS-series Virtual Machines	258
H-series Virtual Machines	259
NV-series and NC-series Virtual Machines	259
NV Virtual Machines	260
NC Virtual Machines	260
Ls-series Virtual Machines	261
Virtual Machine Extensions	262
Managed Disks	263
Availability Sets	265

Deploying a Virtual Machine in Azure	267
Accessing a Virtual Machine in Azure	285
Changing IP and DNS Settings	288
Common Scenarios for Virtual Machines	289
Optimization of Azure Related Communication Traffic	289
On-demand Usage for Calculations	290
Disaster Recovery for On-Premises Servers	292
Summary	294
Chapter 7: Implementing Azure Cloud Services	295
What is an Azure Cloud Service?	295
Understanding the Cloud Service Architecture	296
Roles	297
The Service Endpoint	298
Going Deeper into the Cloud Services	299
Service Definition File	300
LoadBalancerProbes	301
WebRole	301
WorkerRole	305
NetworkTrafficRules	307
Service Configuration File	308
Role	309
NetworkConfiguration	310
Azure Cloud Services versus other Azure PaaS Offerings, such as Azure App Services	312
Selection of a Guest OS and an Update Level	312
Selection of an Azure Series	314
Series A	314
Series D	316
series F	317
series G	318
series N	319
Series H	320
In a nutshell	322
Creating your first Azure cloud service	322
Part 1	322
Part 2	332
Summary	374
Chapter 8: Implementing Azure Container Service	375
What is an Azure Container Service?	375
Understanding Containers	378
Cluster	378
Orchestrator	379
Mesosphere DC/OS	379
Docker Swarm	381
Kubernetes	382

Creating your First Azure Container Service	383
How to Work with Your Azure Container Service Cluster	399
Summary	414
Chapter 9: Implementing Azure Security	415
Azure Identity Security	416
Azure Active Directory	416
Azure Active Directory Editions	416
Privileged Identity Management	418
Identity Protection	423
Detection of Vulnerabilities and Potential Risky Accounts	423
Investigation of Potential Suspicious Events	423
Multi-factor Authentication	426
Conditional Access	432
Resource Security	434
Role-Based Access Control	435
Resource Tags and Policies	440
Azure Network Security	441
Network Security Groups	441
Priority	441
Source	441
Destination	441
Network Virtual Appliances	443
Azure Storage Security	444
Key Management	444
Shared Access Signatures	446
Service-Level SAS	446
Account-Level SAS	447
Storage Encryption	447
Storage Account Encryption	447
Azure Disk Encryption	449
Logging and Monitoring	450
Azure Logs	451
Azure Security Center	453
Summary	456
Chapter 10: Azure Design Patterns	457
Azure Zones and Regions	457
Availability of Resources	458
Data and Privacy Compliance	458
Performance of Applications	459
Cost of Running Applications	459
Virtual Network	459
Architectural Considerations for Virtual Networks	460
Benefits of Virtual Networks	463
Virtual Network Design	463
Connecting to Resources within the Same Region and Subscription	464

Connecting to Resources within the Same Region in Another Subscription	464
Connecting to Resources in Different Regions in Another Subscription	465
Connecting to On-Premise Data Centers	465
Storage	468
Storage Categories	468
Storage Types	468
Storage Features	469
Architectural Considerations for Storage Accounts	471
Design Patterns	472
Messaging Patterns	473
Competing Consumers	474
Priority Queue	475
Queue-Based Load Leveling Pattern	476
Performance and Scalability patterns	477
The Command and Query Responsibility Segregation (CQRS) Pattern	477
Throttling Pattern	479
Other Patterns	480
Retry Pattern	480
Circuit Breaker Pattern	482
Summary	483
Chapter 11: Designing High Availability	484
High Availability	485
SLA	485
Factors Affecting High Availability	486
Planned Maintenance	486
Unplanned Maintenance	486
Application Deployment Architecture	486
High Availability versus scalability	487
High Availability versus Disaster Recovery	487
Azure High Availability	487
Concepts	488
Availability Sets	488
Fault Domain	488
Update Domain	489
Availability Zones	489
Load Balancing	489
Virtual Machine High Availability	490
Computing High Availability	490
Storage High Availability	492
PaaS High Availability	493
Application High Availability	494
Load Balancing	494
Azure Load Balancers	495
Public Load Balancing	496
Internal Load Balancing	497
Port Forwarding	499

Azure Application Gateways	499
Azure Traffic Manager	500
Architectural Considerations for High Availability	501
High Availability within Azure regions	502
High Availability across Azure regions	503
Best Practices	504
Application High Availability	505
Deployment	505
Data Management	506
Monitoring	506
Summary	507
Chapter 12: Implementing Scalability	508
Scalability	509
Scalability versus Performance	510
Azure Scalability	510
Concepts	510
Scaling	511
Scaling Up	511
Scaling Down	511
Scaling Out	512
Scaling In	512
Auto Scaling	512
PaaS Scalability	513
PaaS Scaling Up and Down	514
PaaS Scaling Out and In	515
IaaS Scalability	516
Virtual Machine Scale Sets	517
VMSS Architecture	518
VMSS Scaling	518
Horizontal versus Vertical Scaling	519
Capacity	519
Auto Scaling	519
Upgrades	522
Application Updates	523
Guest Updates	524
Image Updates	524
Scaling Best Practices	524
Prefer Scaling Out	524
Bare Metal versus Dormant Instances	525
Configuring Maximum and Minimum Number of Instances	525
Appropriately	525
Concurrency	525
Stateless	525
Caching and CDN	526
N+1 Design	526

Summary	526
Chapter 13: Designing IoT Solutions	527
IoT	528
IoT Architecture	529
Connectivity	530
Identity	532
Capture	532
Ingestion	533
Storage	533
Transform	533
Analytics	534
Presentation	535
Azure IoT	535
Identity	536
Capture	536
Ingestion	537
Storage	537
Transform and Analytics	538
Presentation	539
IoT Hubs	539
Protocols	540
Device Registration	540
Message Management	542
Device to Cloud Messaging	542
Cloud to Device Messaging	543
Security	544
Security in IoT	544
Scalability	545
Sku Edition	545
Units	547
High Availability	548
Summary	548
Chapter 14: Designing and Implementing Data Solutions	549
Azure SQL	550
Azure SQL Availability	553
Azure SQL Security	554
Elastic Pools	555
Azure SQL Scaling Out	556
Stream Analytics	557
Data Sources	559
Data Integration	559
Data Transformation	559
Stream Analytics Engine	559
Storage and Presentation	560

Architecture	560
Azure Data Factory	561
Data Sources	563
Data Transformation	564
Publish and Presentation	564
Using Data Factory	564
Azure Data Lake	575
Azure Data Lake store	576
Data Lake Security	577
Data Lake Performance	578
Azure Data Lake Analytics	578
Azure SQL Data Warehouse	579
Table Storage	583
Summary	585
Chapter 15: Designing and Implementing Serverless Solutions	586
A Short History of Serverless	587
Serverless	589
Principles of Serverless Technology	590
Lower Cost	590
Event-Driven	590
Single Responsibility	590
Execute Quickly	590
Azure Functions or Functions-as-a-Service (FaaS)	591
Azure Functions Runtime, Bindings, and Triggers	591
Azure Function Runtime	591
Azure Function Binding and Triggers	592
Azure Function Proxies	594
Monitoring	595
Authentication and Authorization	596
Azure Function Configuration	597
Platform Configuration	598
App Services Function Settings	599
Azure Function Cost Plans	599
Azure Functions Advantages	600
Azure Functions Use Cases	604
Implementing Microservices	604
Integration Between Multiple Endpoints	605
Data Processing	605
Integrating Legacy Applications	605
Scheduled Jobs	605
Communication Gateways	606
Types of Azure Functions	606
Creating Your first Azure Function	606
Creating an Event-Driven Function	613

Creating a Connected Architecture with Functions	616
Summary	621
Chapter 16: Designing Policies, Locks, and Tags	622
Azure Tags	623
Tags with PowerShell	625
Tags with the ARM Template	625
Resource Groups versus Resources	626
Azure Policies	627
Built-in Policies	627
Policy Language	628
Allowed Fields	630
Azure Locks	630
Azure RBAC	632
Custom Roles	635
How is it Different from RBAC?	635
Examples of Implementing Azure Governance Features	636
Background	636
Role-Based Access Control	636
Summary	637
Chapter 17: DevOps on Azure	638
What is DevOps?	639
DevOps Practices	642
Configuration Management	643
Desired State Configuration	645
Chef, Puppet, and Ansible	645
Azure Resource Manager Templates	646
Continuous Integration	646
Build Automation	648
Test Automation	648
Packaging	648
Continuous Deployment	649
Test Environment Deployment	650
Test Automation	650
Staging Environment Deployment	651
Acceptance Tests	651
Deployment to Production	651
Continuous Delivery	651
Continuous Learning	652
Visual Studio Team Services	653
Team Foundation Version Control	655
GIT	655
Preparing for DevOps	656
Provision VSTS Account	658
Provision Azure Key Vault	658

Provisioning a Configuration Management Server	658
Provisioning Log Analytics	658
Azure Storage Account	659
Images	659
Monitoring Tools	659
Management tools	659
DevOps for PaaS Solutions	660
Azure App Services	662
Deployment Slots	662
Azure SQL	662
Build and Release Pipeline	663
DevOps for Virtual Machine (IaaS) Based Solutions	663
Azure Virtual Machine	664
Azure Public Load Balancer	664
Build Pipeline	665
Release Pipeline	666
DevOps for Container Based (IaaS) Solutions	667
Containers	667
Docker	668
DockerFile	668
Build Pipeline	668
Release Pipeline	669
Azure Automation	670
Provision Azure Automation Account	671
Author DSC Configuration	672
Importing DSC Configuration	673
Compiling DSC Configuration	674
Assigning Configuration to Nodes	675
Browse the Server	676
Azure for DevOps	676
Summary	679
Other Books You May Enjoy	680
Index	683

Preface

This learning path will guide you through all the important and tough decision-making aspects involved in architecting a Azure public cloud for your organization. The book starts with an extensive introduction to all the categories of designs available with Azure. These design patterns focus on different aspects of cloud such as high availability, data management, and so on.

You will move on to various aspects such as building your cloud structure and architecture. It will also include a brief description about different types of services provided by Azure, such as Azure functions and Azure Analytics, which can prove beneficial for an organization. This book will cover each and every aspect and function required to develop an Azure cloud based on your organizational requirements.

By the end, you will be able to develop a full-fledged Azure cloud as well as implement Azure solutions such as networking, storage, and cloud effectively.

Who This Book Is For

This book focuses on providing essential information about the implementation of Azure solutions to software architects, developers, and IT professionals. The technical information will be offered to you in an intermediate and expert level. Basic knowledge is not necessary, but it is useful for a better understanding of the text. Therefore, some knowledge of programming languages is recommended.

What This Book Covers

Chapter 1, Getting Started with Azure Implementation, provides an overview of cloud service models, cloud deployment models, cloud characteristics, and Azure services.

Chapter 2, Azure Resource Manager and Tools, explains all about the Azure Resource Manager and its concepts (Azure Resource Groups, Azure Resource Tags, and Locks). You will also get an introduction to working with the ARM Templates area.

Chapter 3, Deploying and Synchronizing Azure Active Directory, gives you an overview of the deployment, management, and functionalities of Azure Active Directory and its relation to a Microsoft Azure subscription.

Chapter 4, Implementing Azure Networks, explains how networking in Azure works, how to plan Azure network components, and how to deploy different network components within Azure.

Chapter 5, Implementing and Securing Azure Storage Accounts, explains all about Azure Storage Management and its concepts (BLOBs, tables, queues, and files). You will also get some basic storage configurations.

Chapter 6, Planning and Deploying Virtual Machines in Azure, demonstrates the difference between the Azure virtual machine types, the common use cases for the different types, and how to deploy virtual machines.

Chapter 7, Implementing Azure Cloud Services, explains all about Azure Cloud Services, the Cloud Service architecture, Azure Cloud Services versus Azure App Services, and how to create your first Cloud Service.

Chapter 8, Implementing Azure Container Service, provides basic knowledge about the Azure Container Service area and how to create your first container service. You will also learn the necessary steps to work with the service afterward.

Chapter 9, Implementing Azure Security, explains all about Azure security concepts (identity management with Azure AD, Role-based Access Control (RBAC), and Azure Storage security) and the Azure Security Center.

Chapter 10, Azure Design Patterns, talks about Azure cloud patterns related to virtual networks, storage accounts, regions, and availability sets. It also briefly discusses cloud patterns that help in implementing scalability and performance. Messaging patterns help with the building of reliable solutions. Messaging patterns will also be focused in this chapter.

Chapter 11, Designing High Availability, focuses on describing the high availability features available on Azure. Enterprises need high availability for their deployments. This chapter will build a solid foundation on high availability concepts and help the user make informed decisions related to IaaS and PaaS deployment strategies.

Chapter 12, Implementing Scalability, focuses on designing solutions that can automatically increase and decrease the available resources based on its current consumption to maintain its performance levels. Azure provides virtual machine scale sets (VMSS) for deploying highly scalable solutions. This chapter focuses on VMSS-based architecture and deployment. It will also describe PaaS-based scalability and its strategies.

Chapter 13, Designing IoT Solutions, provides in-depth information about implementing an IoT solution using the Azure cloud. The Azure cloud provides a complete IoT platform for developing device-based solutions. This chapter will show how to architect IoT-based solutions using the Azure cloud. It will also describe architectural concerns that every architect should keep in mind while creating a solution. This chapter will discuss topics related to IoT hubs, event hubs, registering devices, a device to platform conversation, and logging and routing them to appropriate destinations.

Chapter 14, Designing and Implementing Data Solutions, dedicates itself to data storage and services. Azure provides multiple features related to data services. This chapter will focus on providing insights about which features and resources to use for different types of solutions, and their pros, cons, and advantages. A complete architecture for ingesting data, cleaning and filtering data, and storing it in appropriate data stores, such as Data Lake and Cosmos DB, and then pushing data to Power BI for visualizing will be part of this chapter.

Chapter 15, Designing and Implementing Serverless Solutions, focuses on serverless computing. Azure functions are a versatile platform for hosting small business functionalities as functions and help in weaving solutions together. This chapter will focus on understanding the serverless paradigm, Azure functions, its capabilities, creating solutions by combining multiple functions, understanding triggers and parameters, and different sources of inputs and outputs.

Chapter 16, Designing Policies, Locks, and Tags, focuses on using the management features provided by Azure to implement better manage deployments. Tags help by adding additional metadata information to Azure resources. They also help by providing information architecture regarding Azure resources. This chapter provides design guidelines for defining tags for deployments. It also provides details about policies and locks to restrict and control Azure resources regarding their location, usage, size, accessibility, permissions, and so on. It is an important concept, providing management control over Azure resources.

Chapter 17, DevOps on Azure, dedicates itself to DevOps. The Azure cloud provides rich tools, utilities, and scripting support to enable automation for DevOps. Azure supports Azure Resource Manager templates, desired state configuration, PowerShell, Rest API, and open source technologies such as Chef, Python, and Linux to devise end-to-end automation of continuous integration, delivery, and deployment. Infrastructure as Code and configuration management is also supported inherently using Azure features such as Azure automation. This chapter will focus on building CI/CD pipelines and configuration management for Azure resources using VSTS.

To Get the Most out of This Book

This book is essentially intended for cloud architects, developers, consultants, and DevOps engineers who are using Azure to provide their services to end customers and employers. If you are also willing to architect complete solutions on Azure, then this book is ideal for you. If you already have some experience with a architecture on Azure, this book can help you to speed up with it in a fast-paced way.

What You Need for This Book

Learning from a book only works if you have the opportunity to implement what you have learned in practice. That's why you need an Azure subscription. To do this without any cost, you can use a free Azure trial from <https://azure.microsoft.com/en-us/free/>. To understand parts of the book, you also need an installation of Visual Studio. You can use any edition of Visual Studio. To avoid unnecessary costs, it is recommend using the free Visual Studio Community Edition from <https://visualstudio.microsoft.com/downloads/>. If you think you do not possess that knowledge, it is always possible to catch up on the basic requirements by quickly reading up on the major components from the Azure documentation at <https://docs.microsoft.com/en-us/azure/>.

This book assumes a basic level of knowledge on cloud computing and Azure. All you need is a valid Azure subscription and internet connectivity to use this book. A Windows 10 operating system having 4 GB of RAM is sufficient for using Powershell and executing ARM templates.

Conventions

In this book, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "The current project consists of three artifacts in two solution folders, `Templates` and `Scripts`."

A block of code is set as follows:

```
"parameters": {  
  "  storageAccountName": {  
    "    type": "string",  
    "    metadata": {  
      "    description": "Storage Account Name"  
    }  
  }  
}
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
  description: "Storage Account Name"
```

Bold: Indicates a new term, an important word, or words that you see onscreen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "Notice how the **white dress** price was used to pad the **missing values**."



Warnings or important notes appear like this.



Tips and tricks appear like this.

Get in Touch

Feedback from our readers is always welcome.

General feedback: Email feedback@packtpub.com and mention the book title in the subject of your message. If you have questions about any aspect of this book, please email us at questions@packtpub.com.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit www.packtpub.com/submit-errata, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the Internet, we would be grateful if you would provide us with the location address or website name. Please contact us at copyright@packtpub.com with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit authors.packtpub.com.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit packtpub.com.

1

Getting Started with Azure Implementation

Cloud services have come a long way in the last 5 to 10 years. Cloud was and still is one of the biggest trends in **Information Technology (IT)**, with new topics still to be discovered.

In the early 2000s, cloud computing wasn't a widely used phrase, but the concept as well as data centers with massive computing power were already existent and used. Later in that decade, the word *cloud* became a buzzword for nearly anything that was not tangible or online. But the real rise of cloud computing with all its different service models happened before, when the big IT companies started their cloud offers. That was Amazon, Google, and Microsoft in particular. As the cloud offers developed, they enabled companies from startups to Fortune 500s to use cloud services, from web services to virtual machines with billing exact to the minute.

In this chapter, we'll explore the following topics:

- Cloud service models
- Cloud deployment models
- Cloud characteristics
- Azure services overview

Service Models

Cloud computing the new trend model for enabling workloads, that use resources from a normally extreme huge resource pool, that is operated by a cloud service provider. These resources include servers, storage, network resources, applications, services or even functions. These can be rapidly deployed, operated and automated with a low effort and the prices are calculated on a minute base. This cloud model is composed of five essential characteristics, three service models, and four deployment models.

Cloud offers are mainly categorized into the following service models:

- **Infrastructure as a Service (IaaS):** Infrastructure as a Service describes a model in which the cloud provider gives the consumer the possibility to create and configure resources from the computing layer upwards. This includes virtual machines, networks, appliances, and many other infrastructure-related resources:
 - The most popular IaaS resources in Azure contain virtual machines, virtual networks (internal and external), container services and storage.
- **Platform as a Service (PaaS):** Platform as a Service gives the consumer an environment from the operating system upwards. So, the consumer is not responsible for the underlying IaaS infrastructure. Examples are operating systems, databases, or development frameworks:
 - Microsoft Azure contains many PaaS resources such as SQL databases, Azure app services, or cloud services.
- **Software as a Service (SaaS):** Software as a Service is the model with the lowest levels of control and required management. A SaaS application is reachable from multiple clients and consumers, and the owning consumer doesn't have any control over the backend, except for some application related management tasks.
 - Examples of SaaS applications are Office 365, Visual Studio Online, Outlook website, OneDrive, and even the Amazon website itself is a SaaS app with Amazon as its own consumer.

A comparison of service model responsibilities is as follows:

On Premises	Infrastructure (as a Service)	Platform (as a Service)	Software (as a Service)
Application	Application	Application	Application
Data	Data	Data	Data
Runtime	Runtime	Runtime	Runtime
Middleware	Middleware	Middleware	Middleware
O/S	O/S	O/S	O/S
Virtualization	Virtualization	Virtualization	Virtualization
Server	Server	Server	Server
Storage	Storage	Storage	Storage
Network	Network	Network	Network

Cloud service models

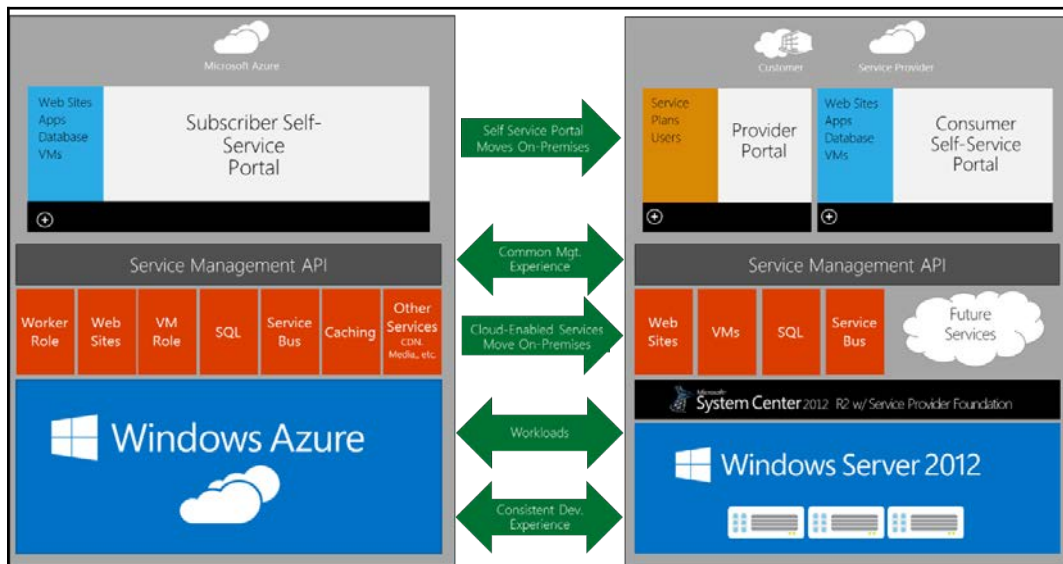
Deployment Models

Furthermore, there are a number of deployment models for cloud computing that need to be discussed. These deployment models cover nearly all common cloud computing provider scenarios. They describe the group of consumers that are able to use the services of the cloud service, rather than the institution or the underlying infrastructure:

- **Public Cloud:** A Public Cloud describes a cloud computing offer that can be accessed by the public. This includes individuals as well as companies.
 - Examples of a Public Cloud are Microsoft Azure and Amazon AWS.
- **Community Cloud:** A Community Cloud is only accessible by a specified group. These are, for example, connected by location, an organization membership, or by reasons of compliance.
 - Examples of a Community Cloud are Microsoft Azure Germany (location) or Microsoft Azure Government (organization and compliance) for US Government authorities.

- **Private Cloud:** Private Cloud describes an environment/infrastructure built and operated by a single organization for internal use. These offers are specifically designed for the different units in the organization.
 - Examples are Microsoft **Windows Azure Pack (WAP)** or Microsoft Azure Stack, as well as OpenStack, if they are used for internal deployments.
- **Hybrid Cloud:** The Hybrid Cloud combines the Private and Public Clouds. It is defined as a Private Cloud environment at the consumer's premises as well as Public Cloud infrastructure that the consumer uses. These structures are generally connected by site-to-site VPNs or **Multiprotocol Label Switching (MPLS)**. A Hybrid Cloud could also exist as combination of any other models such as Community and Public Clouds.
 - Examples are Azure VMs connected to an on-premises infrastructure via ExpressRoute or site-to-site VPN.

Below is a comparison between a related example, in this case between Azure (Public Cloud) and Azure Pack (Private Cloud):



Comparison of Microsoft Azure (Stack) to Azure Pack



With Summer 2017 Microsoft will release the new version of the private cloud adoption from Azure Resource Manager. The new version will be named Azure Stack and will sooner or later be equal to the Azure Resource Manager Framework.

Cloud Characteristics

Microsoft Azure is one of the biggest cloud service providers worldwide, offering a wide range of services from IaaS to PaaS to SaaS. It fulfills all the characteristics that the **National Institute of Standards and Technology (NIST)** describes for cloud computing. These are as follows:

- **On-demand self-service:** An automated deployment of resources that a consumer orders through an interface such as a consumer portal.
- **Broad network access:** Providing availability of cloud services through a standardized network interface that are, at best, accessible by several endpoint devices.
- **Resource pooling:** This means that the automated assignment and reassignment of diverse resources from various resource pools for individual customers is possible.
- **Rapid elasticity:** It is also known as rapid scaling, and describes the ability to scale resources in a massive way. The automatic and fast assignment and reassignment of resources, and rapid up- and down-scaling of single instances are keywords when talking about **rapid elasticity**. The adjustment of web server resources depending on the demand is an example of rapid elasticity.
- **Measured service:** All usage data for consumer resources is monitored and reported to be available for consumers and the cloud provider. This is one of the requirements for minute-based billing.

Microsoft Azure

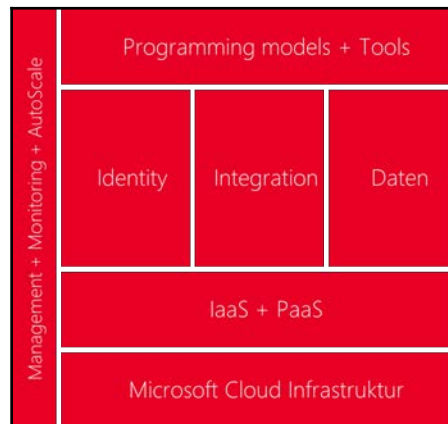
When Windows Azure came online to the general public in February 2010 there were solely database services, websites, and virtual machine hosting available. Over time, Microsoft constantly added features and new services to Azure, and, as there were more and more offers for Linux and other non-Windows services, Microsoft decided in April 2014 to rename Windows Azure as Microsoft Azure. This supported Microsoft's commitment to transform itself into a services company, which means that, in order to be successful, you have to offer as many services as possible to as many clients as possible. Since then, Microsoft has constantly improved and released new services. Additionally, it constantly builds and expand its data centers all over the world.



The service updates happen very frequently. That is the reason why you need to keep yourself informed. For example, the database offer you are using could have improved storage or performance capabilities. Information sources are the official Microsoft Azure Blog and the Azure Twitter channel. Furthermore, information can be found on the websites of several Azure MVPs.

Azure Services Overview

Azure offers many services in its cloud computing platform. These services include the following:



The service categories, differentiated between platform services and infrastructure services, are as follows:

- **Platform Services:**
 - **Management:** The management services include the management portal, the marketplace with the services gallery, and the components to automate things in Azure.
 - **Services compute:** Services compute are the Azure cloud services that are basically PaaS offers for developers to quickly build and deploy highly scalable applications. The service fabric and Azure RemoteApp are also in this category.
 - **Security:** Containing all the services that provide identity in Azure, such as the Azure Active Directory, multi-factor authentication, and the key vault that is a safe place for your certificates.
 - **Integration:** The integration services include interface services such as BizTalk and Azure Service Bus, but also message helpers such as storage queues.
 - **Media and CDN:** These are basically two services. One is the CDN that makes it possible to build your own content delivery network based on Azure. The other is the media services that make it very easy to use and process different media with the help of Azure.
 - **Web and mobile:** These include all the services that assist in creating apps or backend services for web and mobile; for example, web apps and API apps.
 - **Developer services:** These are cloud-based development tools for version control, collaboration, and other development-related tasks. The Azure SDK is a part of the developer services.
 - **Data:** The data services contain all the different database types that you can deploy in Azure (SQL, DocumentDB, MongoDB, Table storage and so on) and diverse tools to configure them.

- **Analytics and IoT:** As the name suggests, analytics services are tools to analyze and process data. This offers a broad range of possibilities from machine learning to stream analytics. These can, but don't have to, build on certain data services. **Internet of Things (IoT)** services include the fundamental tools needed to work with devices used for the IoT such as the Raspberry Pi 2.
- **Hybrid operations:** This category sums up all the remaining services that could not clearly be categorized. These include backup, monitoring, and disaster recovery as well as many others.
- **Infrastructure services:**
 - **Operating system and server compute:** This category consists of compute containers. It includes virtual machine containers and, additionally, the container services that are quite new to the product range.
 - **Storage:** Storage services are the two main storage types: **BLOB** and **file storage**. They have different pricing tiers depending on the speed and latency of the storage ordered. Storage is looked at in detail in *Chapter 6, Planning and Deploying Virtual Machines in Azure*.
 - **Networking:** This category consists of the basic networking resources. Examples are Load Balancer, ExpressRoute, and VPN Gateways.

The important thing is to remember that we are talking about a rapidly changing and very agile cloud computing platform. After this chapter, if you have not already done so, you should start using Azure by experimenting, exploring, and implementing your solutions, while reading the correlating chapters.

For testing purposes, you should use the **Azure Free Trial**

(<https://azure.microsoft.com/en-in/offers/ms-azr-0044p/>), **Visual Studio Dev Essentials** (<https://www.visualstudio.com/dev-essentials/>) or the included Azure amount from a MSDN subscription.

Azure Basics

In the following section, we will take a look at the basic Microsoft Azure key concepts. This should provide an overview and an idea of how to use Azure.

Azure Resource Manager

In the previous major version of Azure, a deployment backend model called **Azure Service Manager (ASM)** was used. With higher demand on scaling, being more flexible and more standardized a new model called the ARM was introduced and is now the standard way of using Azure.

This includes a new portal, a new way of looking at things as resources and a standardized API that every tool, including the Azure portal, that interacts with Azure uses.

With this API and architectural changes, it's possible to use such things as Azure Resource Manager templates for any size of deployment. ARM templates are written in **JavaScript Object Notation (JSON)** and are a convenient way to define one or more resources and their relationship to another programmatically. This structure is then deployed to a resource group. With this deployment model, it's possible to define dependencies between resources as well as being able to deploy the exact same architecture again and again. The next part will dive a little deeper into resources.

Resources

Azure resources are the key to every service offering in Azure. Resources are the smallest building blocks and represent a single technical entity like a VM, a Network Interface Card, a storage account, database or a website.

When deploying a web app, a resource called **App service** will be deployed along with a service plan for billing.

When deploying a Virtual Machine from Azure Marketplace template a VM resource will be created as well as a storage account resource holding the virtual hard disks, a Public IP Address resource for initial access to the VM, a Network Interface card and a Virtual Network resource.

Every resource has to be deployed to one specific resource group. A resource group can hold multiple resources, while a single resource can only exist in one resource group. Resource groups also can't contain another resource group, what leads to a single layer of containers regarding resources.

One resource group can contain all resources of a deployment or multiple resources of different deployments. There are no strong recommendations on structuring resource groups, but it's recommended to organize either resource of one project/enrollment/deployment in separate resource groups or distribute resources based on their purpose (networking, storage, and so on) to resource groups.

Azure Regions

Azure as a global cloud platform provides multiple regions to deploy resources at. One region consists of at least one highly available data center or data center complex.

The (at the time of writing) 34 regions are distributed all over the world and include community clouds like Azure Germany or government clouds like US Gov Virginia. Additionally, to the implicit high availability inside a region and a data center, it's possible to select a region and creation of most resources. This region represents a set of data centers that are connected through Microsoft owned high-speed network to replicate the data in near real-time between the specific data centers.

Regions can also have an impact on the performance and availability of some resources. Some services may be not or only partially available in a specific region.

The costs of offered services also vary by region. For reduced latency, it's recommended to choose a region next to the physical location of the consumer. It might also be important which legal requirements must be met. This could, for example, result in a deployment only in EU regions or even regions in specific countries.

- The available Azure regions are listed here: <https://azure.microsoft.com/en-us/regions/>
- This list lists all the services available in specific regions: <https://azure.microsoft.com/en-us/regions/services/>

Azure Portal

The Azure Portal is a web application and the most straightforward way to view and manage most Azure resources. The Azure portal can also be used for identity management, to view billing information and to create custom dashboards for often used resources to get a quick overview of some deployments.

Although it's easy to start with using and deploying services and resources, it's highly recommended to use some Azure Automation technologies for larger and production environments.

The Azure portal is located at <https://portal.azure.com>.

Azure Automation

Azure automation is a service and a resource as well as an Azure concept in the context of cloud computing.

It's very important to see automation as an essential concept when it comes to cloud computing. Automation is one of the key technologies to reduce operational costs and will also provide a consistent and replicable state. It also lays the foundation to any rapid deployment plans.

As Azure uses very much automation internally, Microsoft decided to make some of that technology available as a resource called **automation account**.

Azure Automation Tools

Azure provides several ways of interacting and automating things. The two main ways to interact with Azure besides the portal are the Azure PowerShell and the **Azure Command-Line Interface (CLI)**.

Both are basically just wrappers around the Azure API to enable everyone not familiar with REST APIs, but their specific scripting language to use and automate Azure.

The Azure PowerShell module provides cmdlets for managing Azure services and resources through the Azure API. Azure PowerShell cmdlets are used to handle Account management and Environment management like creating, updating and deleting resources. These cmdlets work completely the same on Azure, Azure Pack and Azure stack, Microsoft's private cloud offerings. Azure PowerShell is open source and maintained by Microsoft. The project is available on GitHub at the following link: <https://github.com/Azure/azure-powershell>.

The Azure CLI is a tool that you can use to create, manage, and remove Azure resources from the command-line. The Azure CLI was created for administrators and operators that are not that experienced with Microsoft technologies, but with other Server technologies like Unix or Linux. The Azure CLI is an open source project as well and is available for Linux, macOS, and Windows here: <https://github.com/Azure/azure-cli>.

REST APIs

All Azure Services, including the Azure Management Portal, provide their own REST APIs for their functionality. They can, therefore, be accessed by any application that RESTful Services can process.

In order for software developers to write applications in the programming language of their choice, Microsoft offers wrapper classes for the REST APIs.

These are available as a so-called Azure SDK for numerous programming languages (for example, .NET, Java, Node.js) here: <https://github.com/Azure>

Summary

In the last chapter, we learned that the classic model had changed a lot. In the classic world, you had complete control over the hardware and software that is deployed. This has led to hardware decisions that focus on massive scaling. An example is purchasing a server with more cores to satisfy peak performance needs. Unfortunately, this infrastructure might be underutilized outside the demand window. With Azure, you can deploy only the infrastructure that you need, and adjust this up or down at any time. This leads to a focus on scaling out through the deployment of additional compute nodes to satisfy a performance need. Although this has consequences for designing an appropriate software architecture. These days it's sure that scaling out cloud services is more cost-saving than scaling up through racks and servers.

Microsoft has built many Azure datacenters around the globe. There are even more planned, especially sovereign clouds in regions like China and Germany. Only the largest global enterprises can deploy datacenters in this manner, so using Azure makes it easy for enterprises of any size to deploy their services close to their customers.

For small businesses, Azure allows for a low-cost entry point, with the ability to scale on demand. This prevents a large capital investment in infrastructure and provides the flexibility to architect and re-architect systems as needed. Using cloud technologies supports the startup mentality of growing fast and failing fast.

In the next chapter, we will take a look at Azure Resource Manager and the Azure resource manager tools.

2

Azure Resource Manager and Tools

The Azure platform consists primarily of three parts: part 1, called **Azure Execution Model**, denotes the areas where you can provide your services and applications in the cloud; parts 2 and 3, called **Azure Application Building Blocks** and **Azure Data Services**, refer to services that extend the platform to common capabilities and functionalities.

We could actually quit the description of the platform, because most users get only these three parts to see, but there are still more. Many other services are working under the hood of the platform and ensure the ongoing operation. These services include, for example, the Azure traffic manager, the Azure load balancer, and the Azure resource manager. All of these services can be customized using various interfaces for your personal needs.

In this chapter, we'll introduce you to the Azure resource manager in detail and we'll explore the following topics:

- Azure resource manager and Azure resource groups
- Azure resource tags
- Azure resource locks
- Working with ARM templates and the Azure resource explorer
- Creating your own ARM template

Understanding the Azure Resource Manager

With the classic Azure system management, you could previously manage only one resource on the Azure platform at the same time. But what about more complex applications, as are common today? The infrastructure of today's applications typically consists of several components - a virtual machine, a storage account, a virtual network, a web app, a database, a database server, or a third-party service. To manage such complex applications, with the first preview of the Azure management portal 3.0, the concept of resource groups was introduced.

You now see your components no longer as separate entities, but as related and interdependent parts of a single entity. So you will be able to manage all the resources of your application simultaneously.

As an instrument for this type of management, the Azure resource manager (and the Azure resource manager tools) was introduced, and can be accessed via a variety of different technologies and interfaces. These access options include the following:

- The traditional way via the Azure management portal (portal version 3.0 and newer)
- The script-based way via Azure PowerShell (look for PowerShell modules with the prefix AzureRM) or via the cross-platform command-line interface Azure CLI
- For developers, there are also SDKs available (.NET and some other programming languages) and, as with all Azure services, an extensive REST API

Functionalities Provided by the Azure Resource Manager

In this section, we will give you a brief overview of the functionalities of the Azure resource manager. The list, however, is only a selection and is limited to the most frequently used features. You will find detailed information on the use of the features in the following sections of this chapter.

Let us have a look at the list:

- Access control with Azure **Role-Based Access Control (RBAC)**.
- Logical organization of all the resources of a subscription, with Azure resource tags (for example, for each project and tenant).
- Improved cost control. You can view the costs for the whole group or for a group of resources with the same tag.
- Use of Azure resource manager templates (ARM templates):
 - As a deployment template, in the provision of individual solutions on the Azure platform (the most popular example is deploying a SharePoint server farm).
 - As a resource provider template, for the implementation of measures (for example, configuration) within the resource groups.
- By using templates, you have the ability to define dependencies between resources, so that they are provided in the correct order.
- Through the use of templates, you have the possibility to repeatedly and securely provide your application and resources throughout the entire lifecycle, and this always in the same form.
- You can modify the templates (JSON data files) to your own needs and even create your own templates.

Working with the Azure Resource Manager

We now know that the Azure resource manager serves as the technical base for the provision of resources. How are we going to continue? First, we will deal with the basic workflows in Azure resource manager. Then, in the second part, we will look at working with templates.

Before we begin, let's introduce some very important facts that are important for all workflows:

- All of the resources in your resource group have the same life cycle. You will deploy, update, and delete them at the same time.
- Each resource can only exist in one resource group.
- You can add or remove a resource to a resource group at any time. You can also move a resource from one resource group to another.

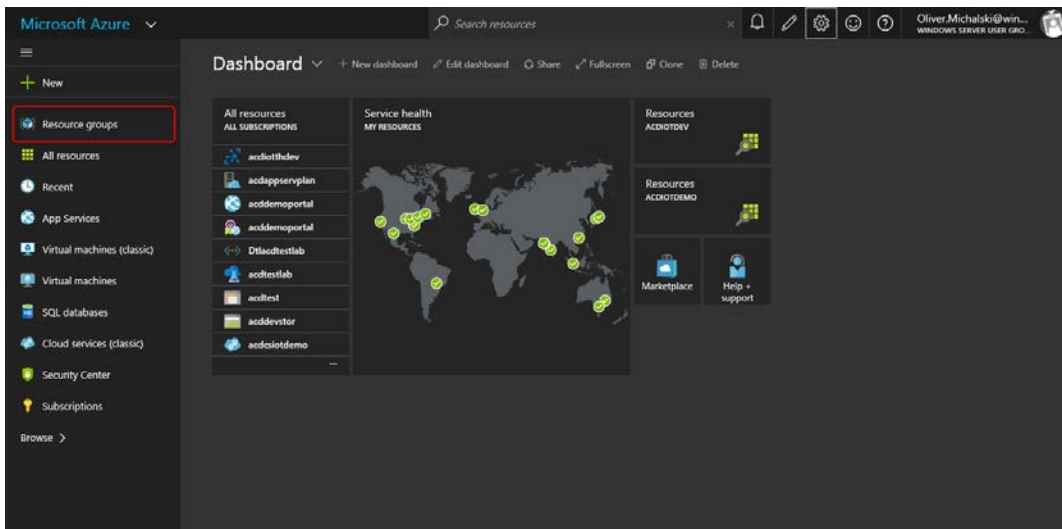
- A resource group can contain resources that exist in different locations.
- A resource can interact with a resource in another resource groups when the two resources are related but they do not share the same life cycle (for example, a web app connecting to a database).

OK, let's start with the first workflow.

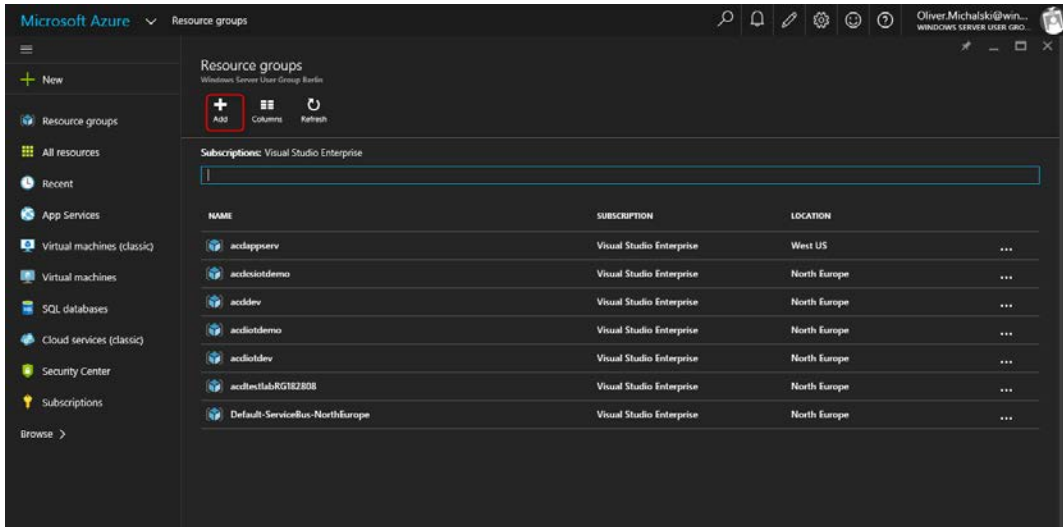
Creating an Azure Resource Group

The **Creating an Azure resource group** workflow is the first in a series of basic workflows, but also the most important. Why? Simple answer: an Azure resource group is the central element of the Azure resource manager concept. Without an existing resource group, nothing works, and this means not only individual services, but your complete Azure subscription. To create an Azure resource group, perform the following steps:

1. Open your Azure management portal at <https://portal.azure.com>.
2. In the portal, click on **Resource groups**:



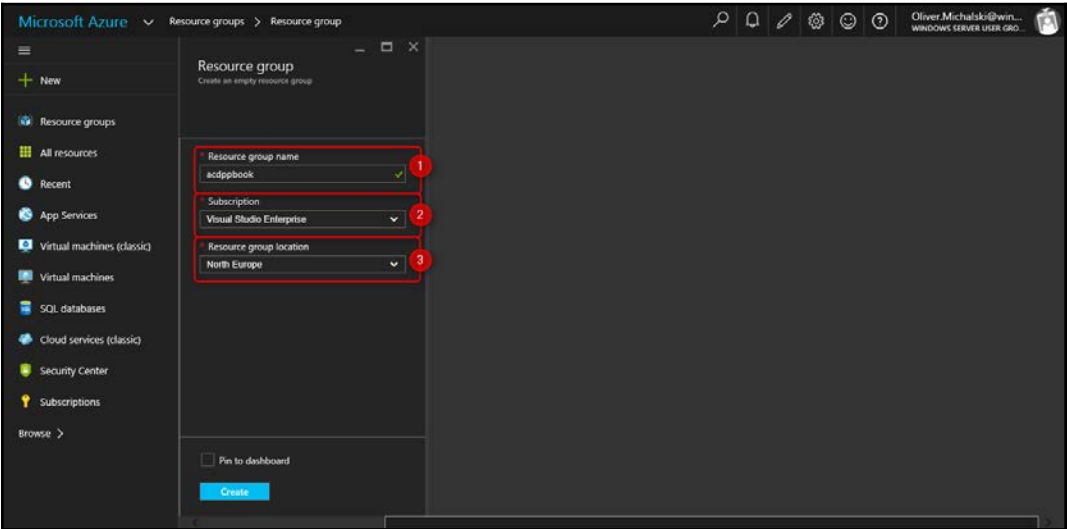
3. On the **Resource groups** blade, click on the **Add** option:



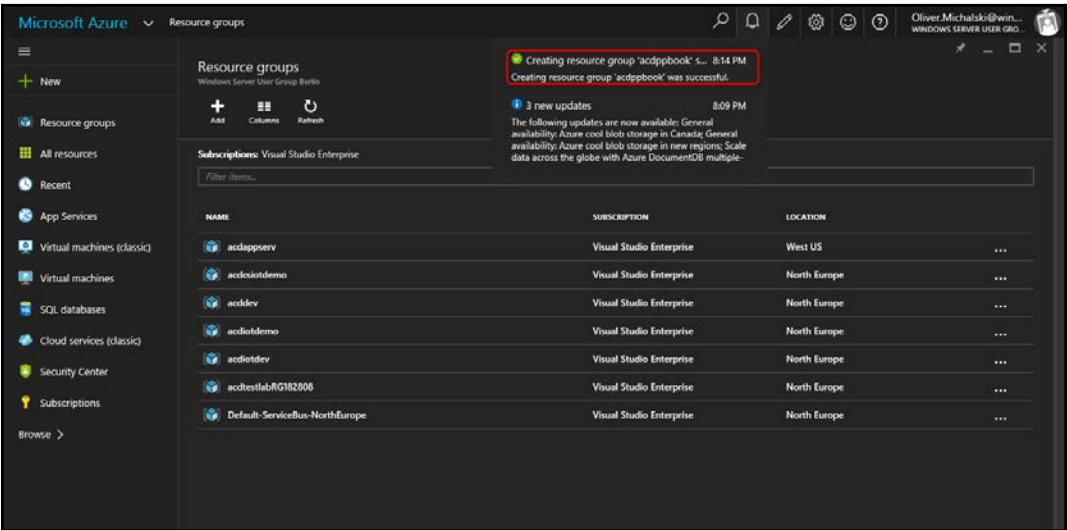
4. On the **Resource groups** blade, type the following values, and then click on the **Create** button:

- **Resource group name:** acdppbook (or the name of your choice)
- **Subscription:** Use the default subscription

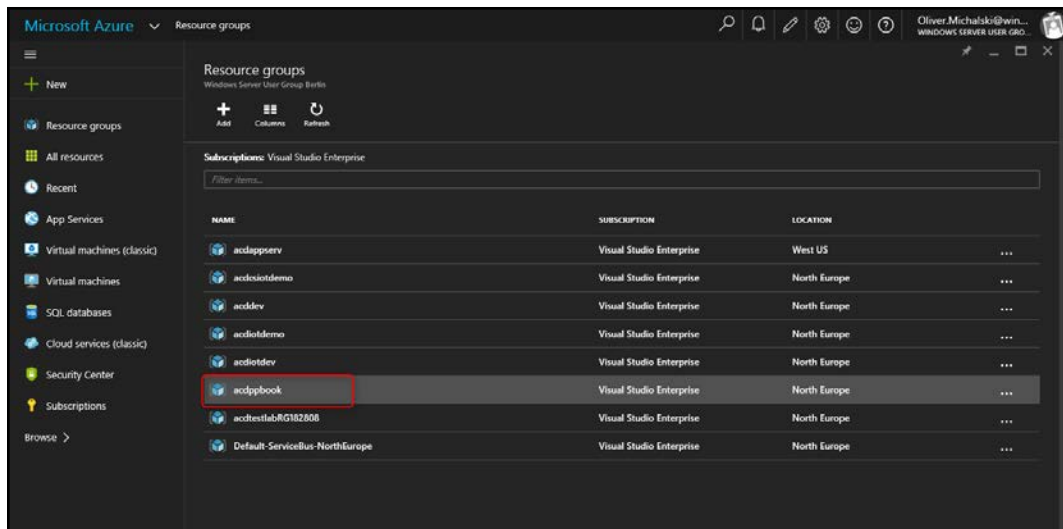
- **Resource group location:** Select your preferred location:



5. You can see the progress when creating the resource group, in the notification service of the portal:



6. As soon as the resource group has been created, you can find it in the list:



Adding a Resource to an Azure Resource Group

We have just learned how to create an Azure resource group. Now we'll fill the new resource group with life and add a resource. To complete this process, the Azure platform has a total of three possible approaches. We will now introduce you to them one by one.

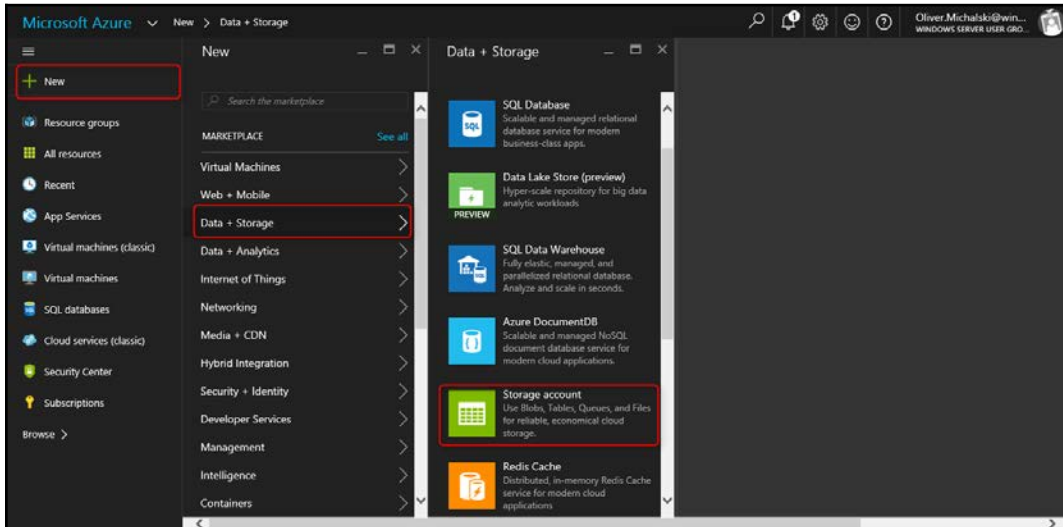
For all approaches, we will show the necessary work steps on the example of adding an Azure storage account. But note that the description of the procedure applies also to all other resource types in the same or slightly modified form.

First Approach: Adding a Storage Account to your Azure Resource Group

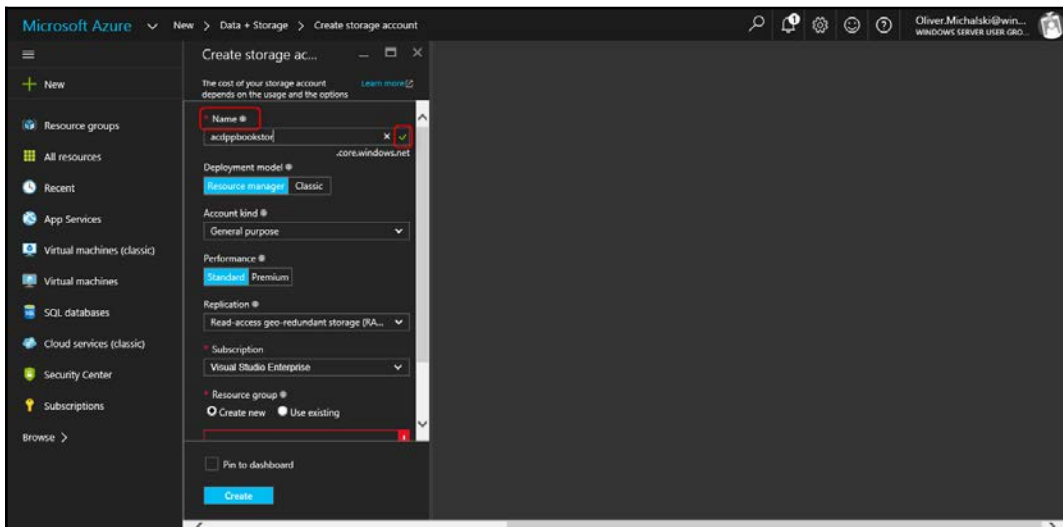
To add a storage account to your Azure resource group, perform the following steps:

1. Open your Azure management portal at <https://portal.azure.com>.

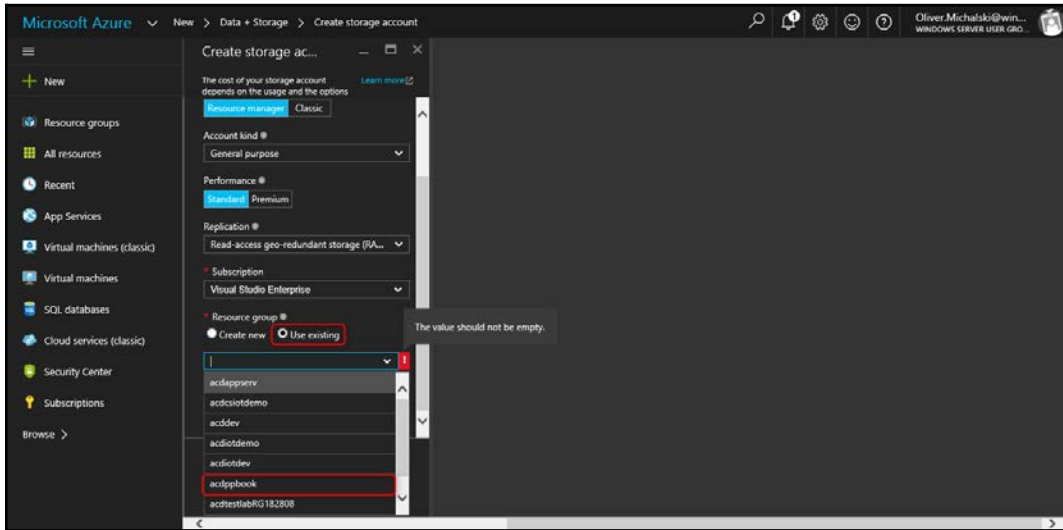
2. In the portal, click on **New**, then click on **Data + Storage**, and then click on **Storage account**:



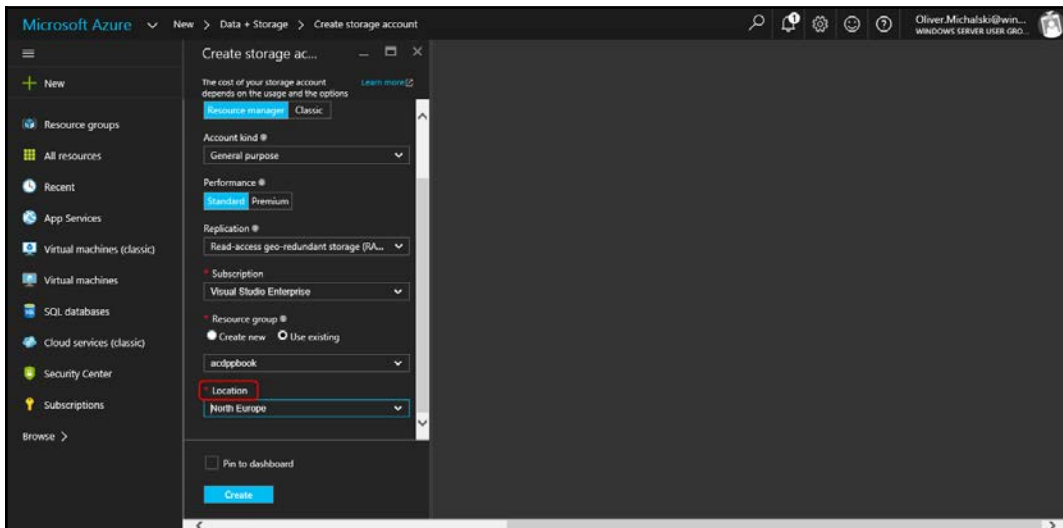
3. On the **Create storage account** blade, type a unique name for the storage account you are creating in the **Name** textbox. If the name is unique, you will see a green tick:



4. In the **Resource group** section, click the **Use existing** checkbox, then search and select **acdpbook** in the drop-down list:



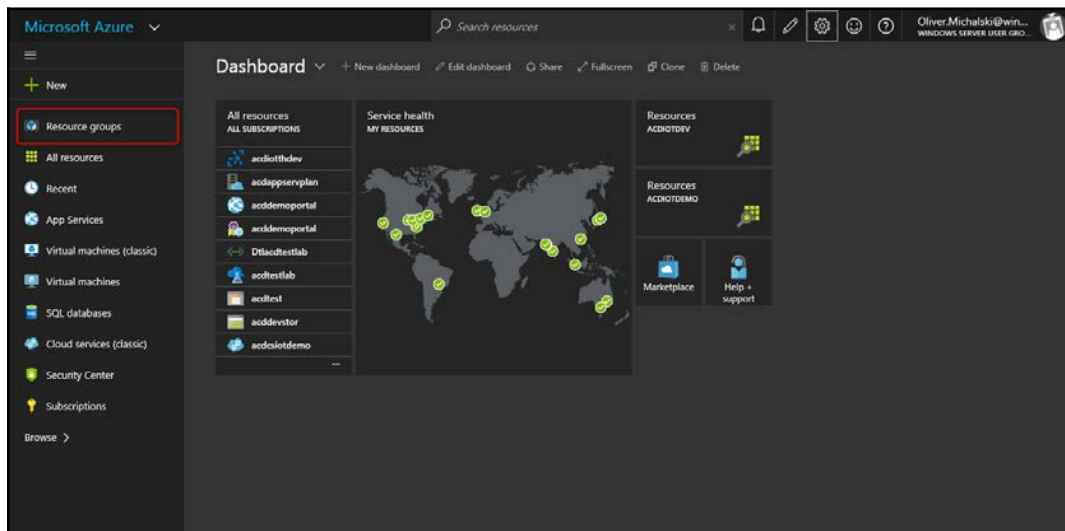
5. In the **Location** list, select the same location you have been using for the Azure resource group, and then click on the **Create** button:



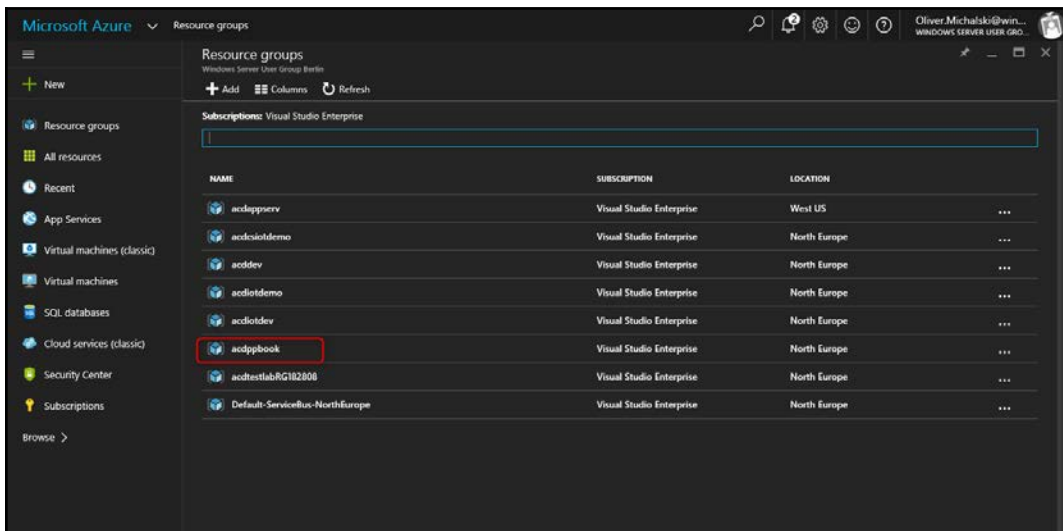
Second Approach: Adding a Storage Account to your Azure Resource Group

The first approach is the default path for adding Azure resources and usually suffices in most cases. There is a second possibility available, which is also applicable for all types of Azure resources but was originally intended for offers from third-party companies: the way over the Azure marketplace. Let's have a look:

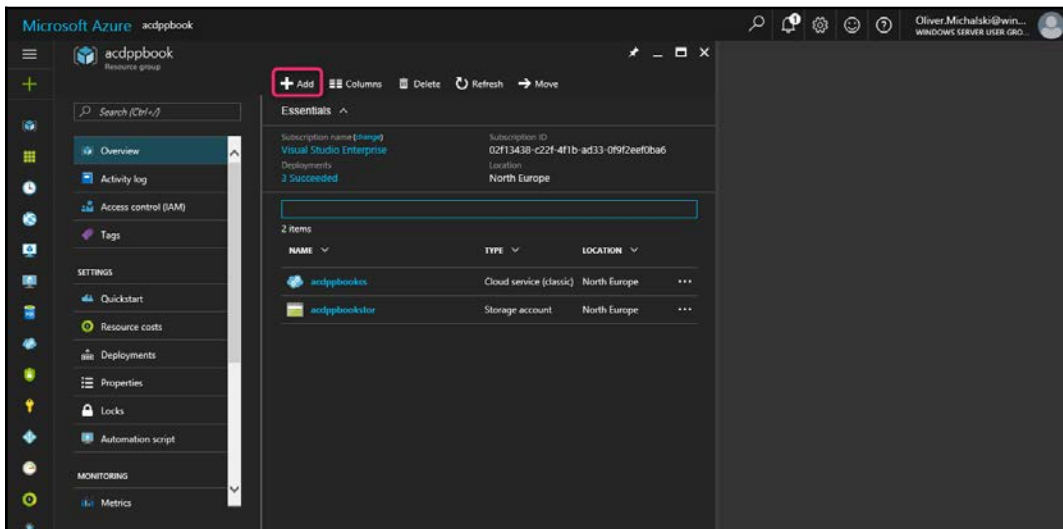
1. In the portal, click on **Resource groups**:



2. On the **Resource groups** blade, click on the **acdppbook** name field:



3. On the **Resource groups** dashboard, click on the **Add** option:



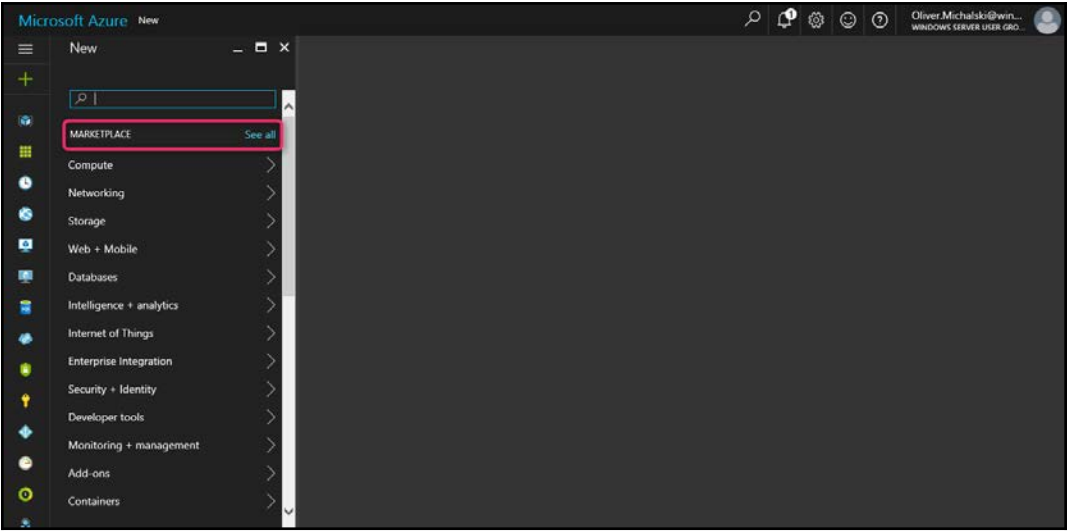
4. Now the Azure marketplace opens. Select a resource and create it as described in the previous section:



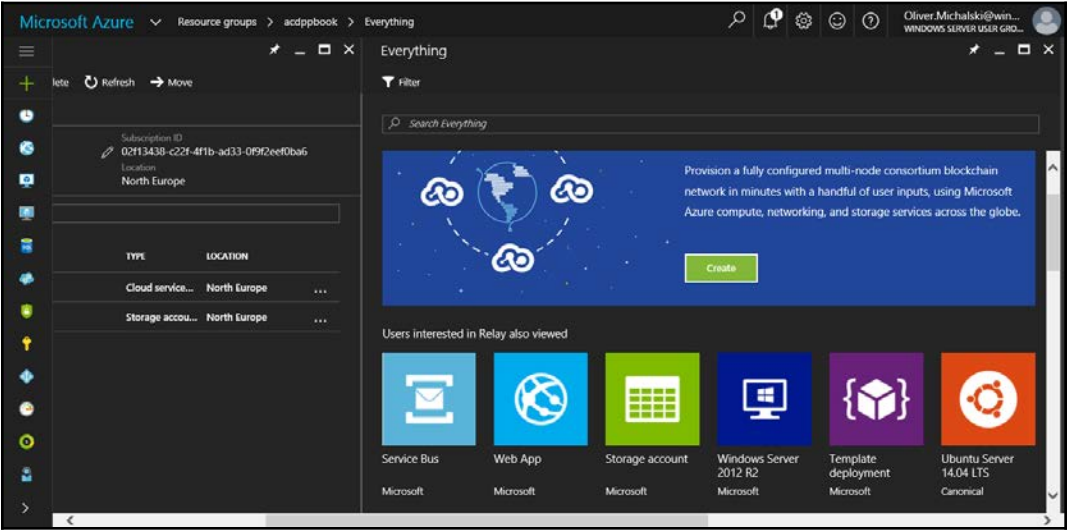
Third Approach: Adding a Storage Account to your Azure Resource Group

From the second approach (the way over the Azure space), a special case is still available: template deployment. This is, however, at the time of writing this section, still a preview. Changes in the procedure and functionality are possible at any time. Let's have a look at this:

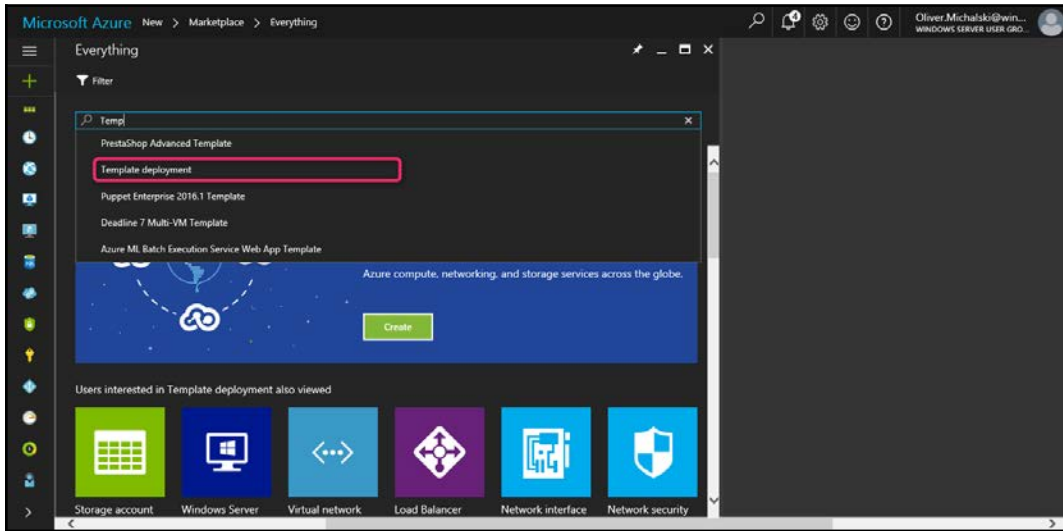
1. In the portal, click on **New** and then click on **See all in MARKETPLACE**:



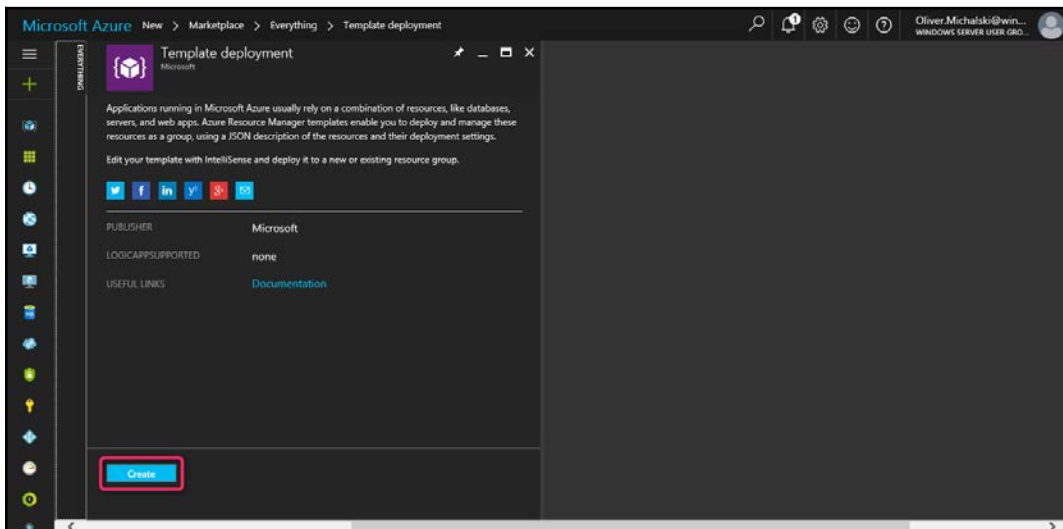
2. Now the Azure marketplace opens. Select the **Template deployment** offer:



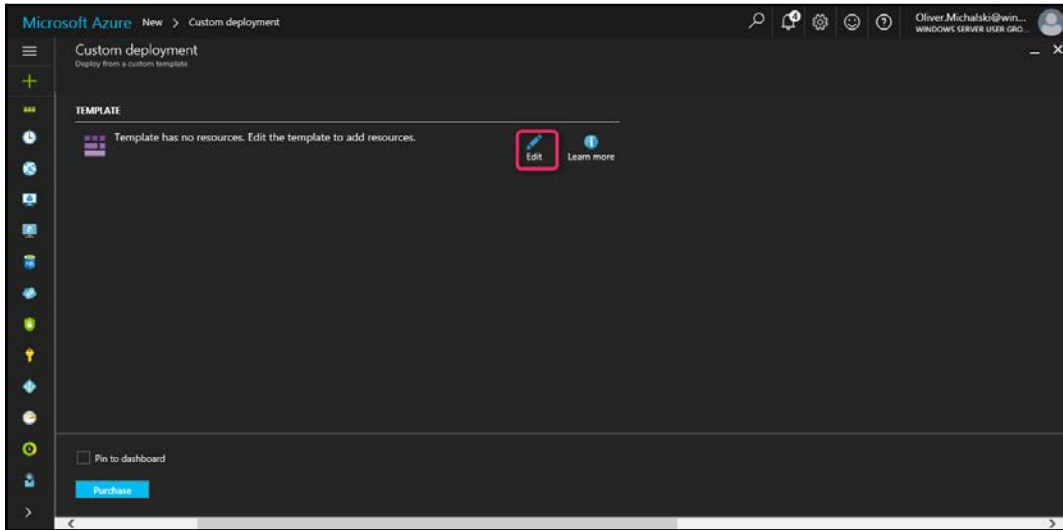
3. If you don't see the entry immediately, simply enter `Template deployment` into the search field:



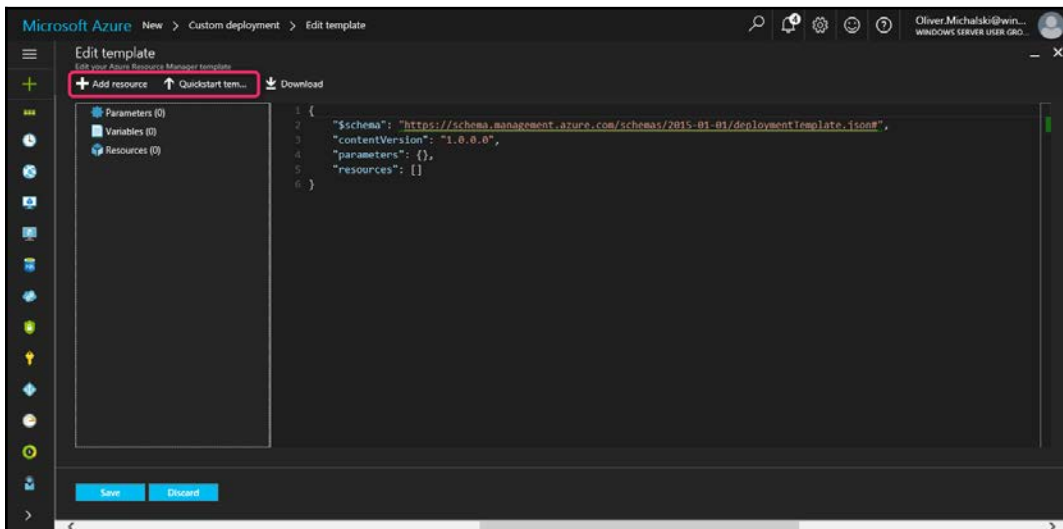
4. Now the info page of the offer opens; press the **Create** button here:



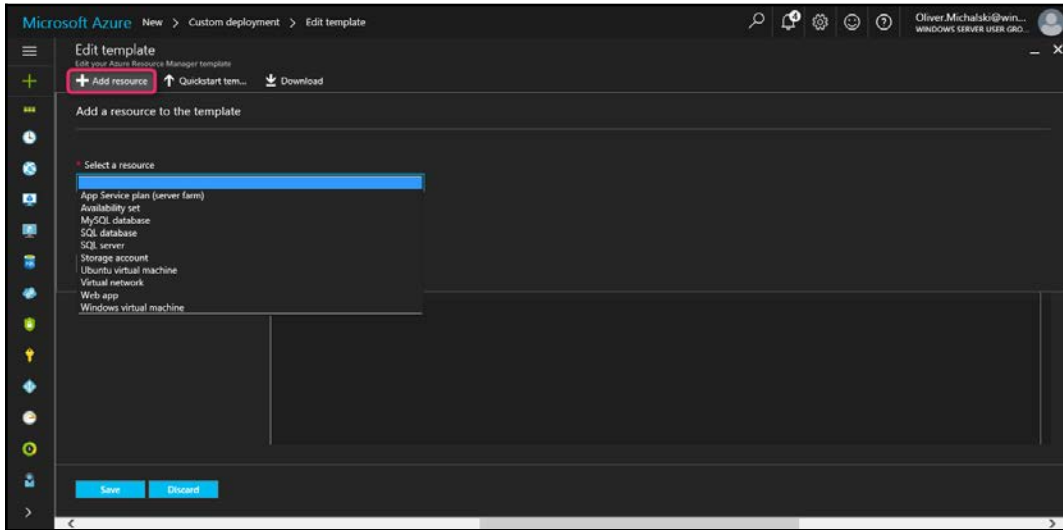
- Next, the **Custom deployment** blade opens. By default, however, there are no resources, so you must first press the **Edit** button:



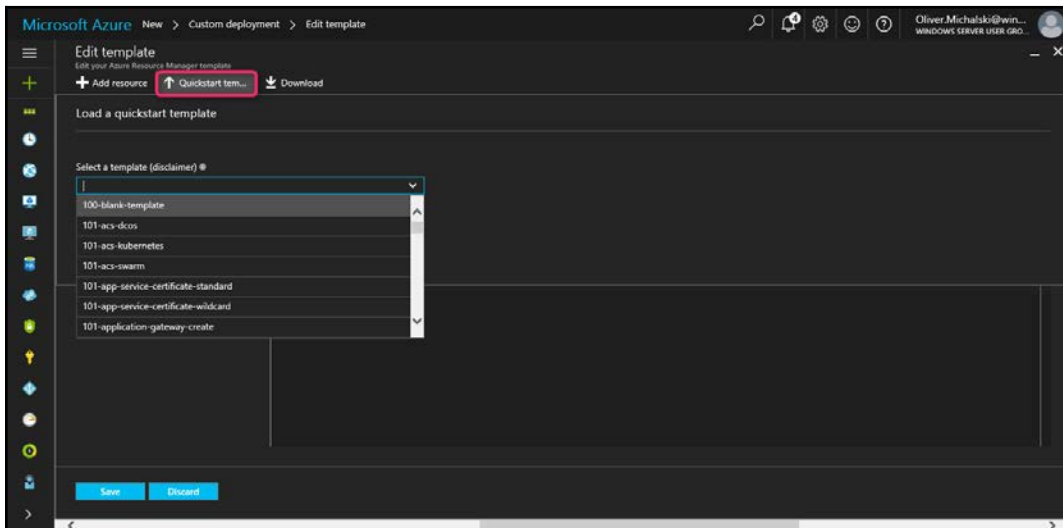
- Next, the **Edit template** window will open with a default template. For your work with this, you will find the two buttons **Add resource** and **Quickstart templates**:



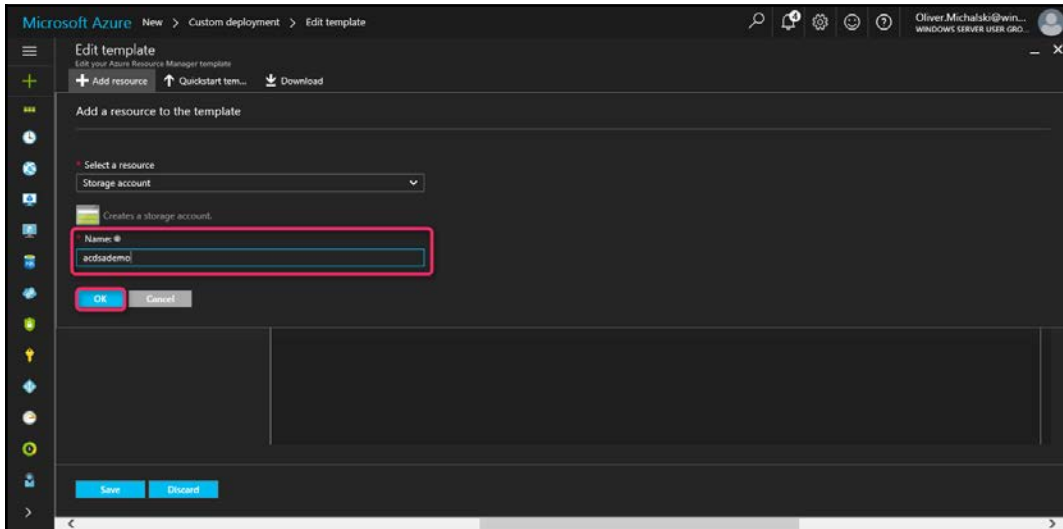
7. With the **Add resource** button, you can select an Azure resource from a list and add it to your template:



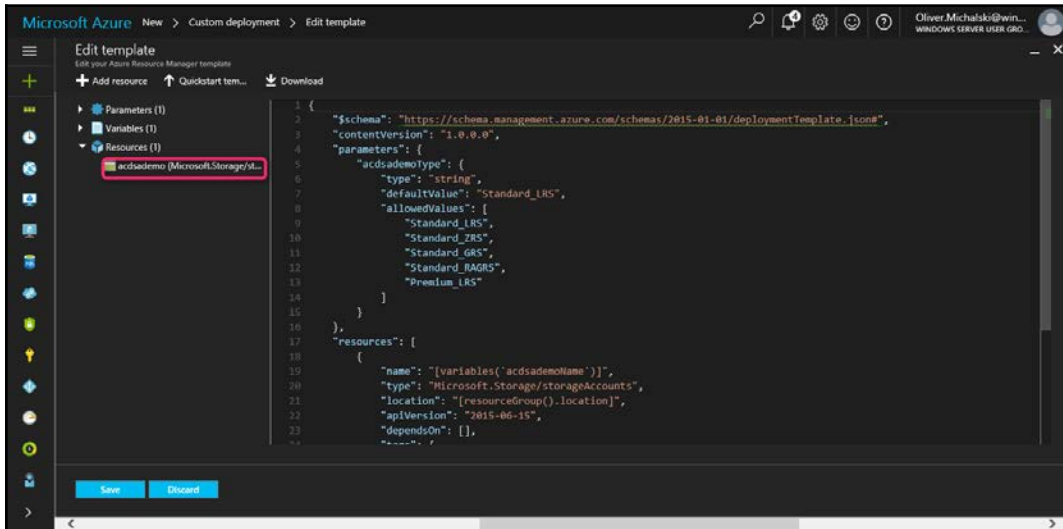
8. With the **Quickstart template** button, you can load a more complex template, from the constantly growing collection of community templates:



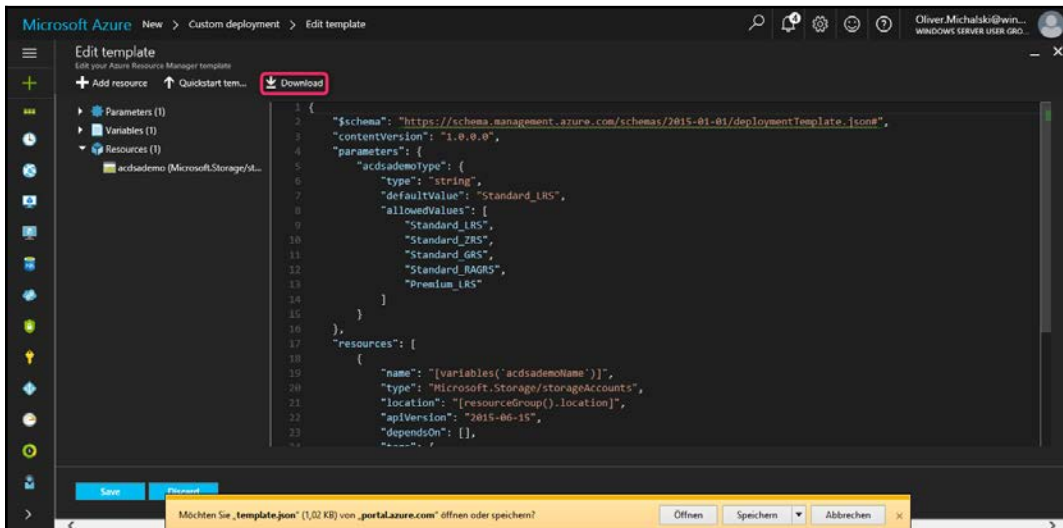
9. For our small demo, we selected **Storage** account in the **Select a resource** list. Depending on the selected resource, additional input fields will appear (usually at least one input field for the resource **Name**). When you have filled everything in, press the **OK** button:



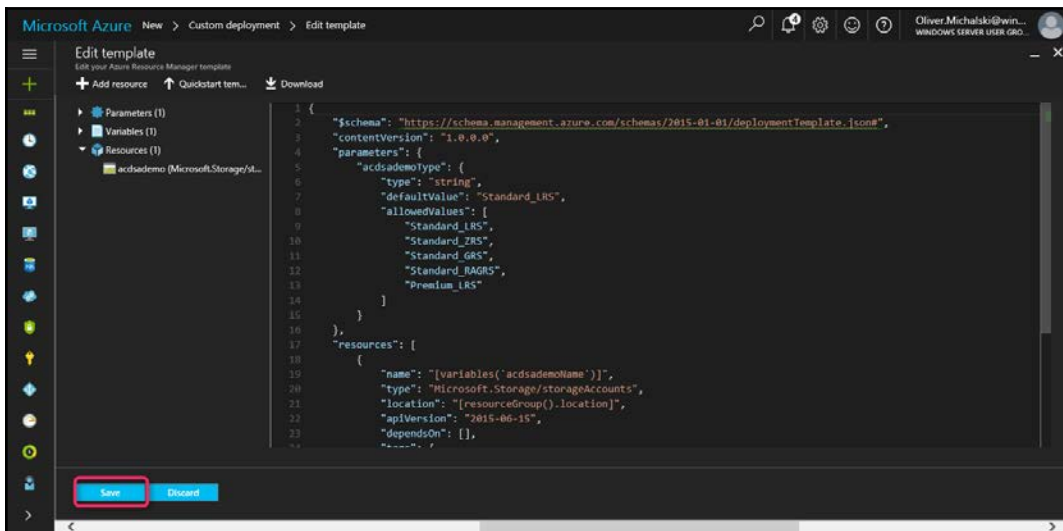
10. Now the **Edit template** window opens again, but this time with a template for your resource. If necessary, you can add more resources to your template by repeating the previous working step:



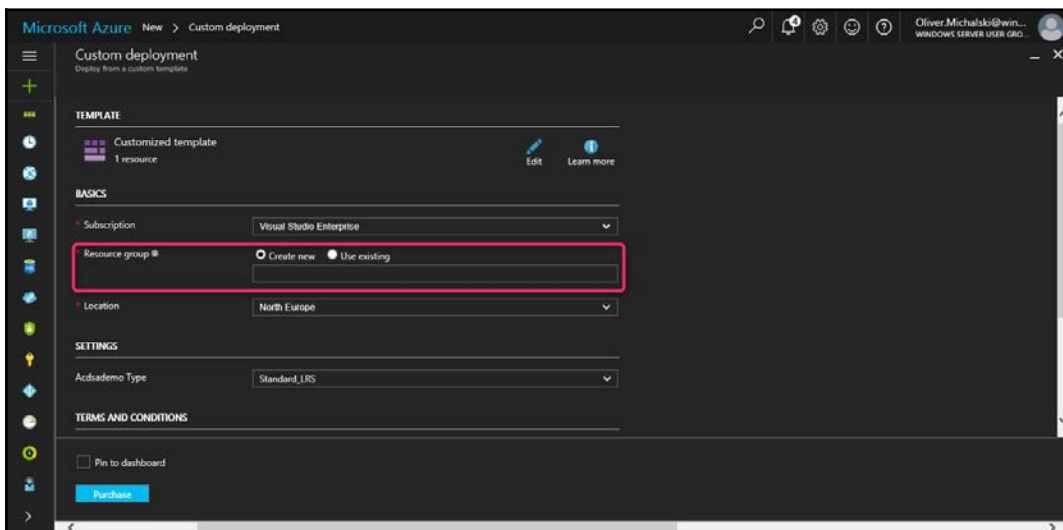
11. If you want, you can also download a copy of the template:



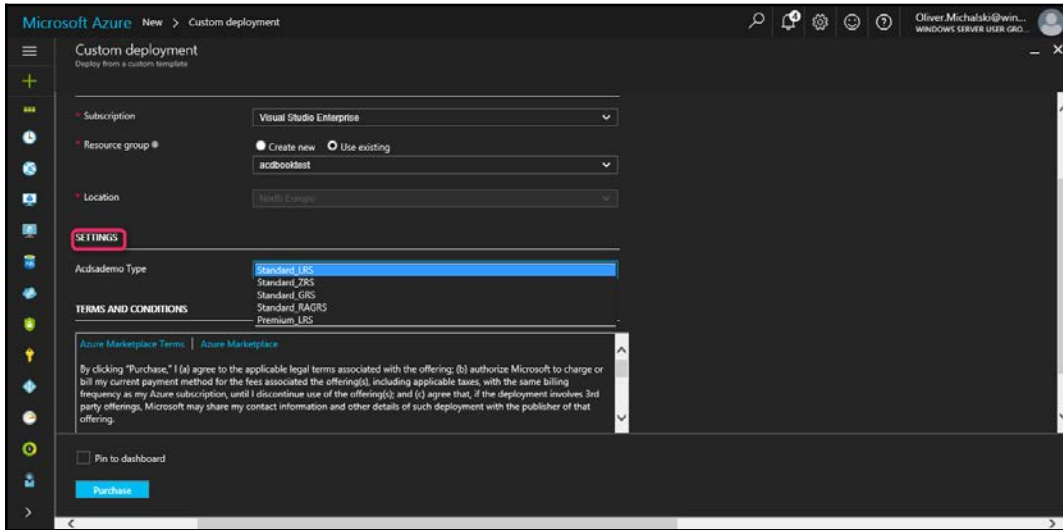
12. When everything is ok, press the **Save** button:



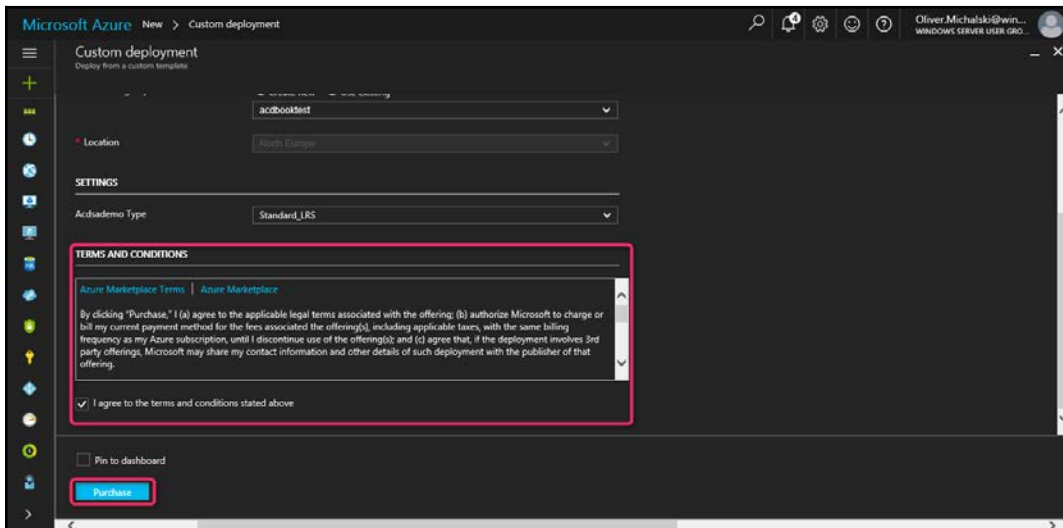
13. We are back in the **Custom deployment** blade. Here you have to make a few final entries. First, select a **Resource group**:



14. Then you can change, depending on the selected resource type, the **SETTINGS** of your resource (optional):



15. Finally, you must accept the Azure marketplace license, and then press the **Purchase** button:





This is not really a purchase!

Now the deployment of the template is running.

Tagging in the Azure Resource Manager

We have just learned how to create an Azure resource group, and how to add an Azure resource. What we are still missing? Short answer: we still need a way to organize our resources logically, for example, for the calculation of cost or for a targeted tracking.

The Azure resource manager offers a solution for this—Azure resource tags. Azure resource tags are any key/value pairs that appear useful to describe a resource.

Let's see an example:

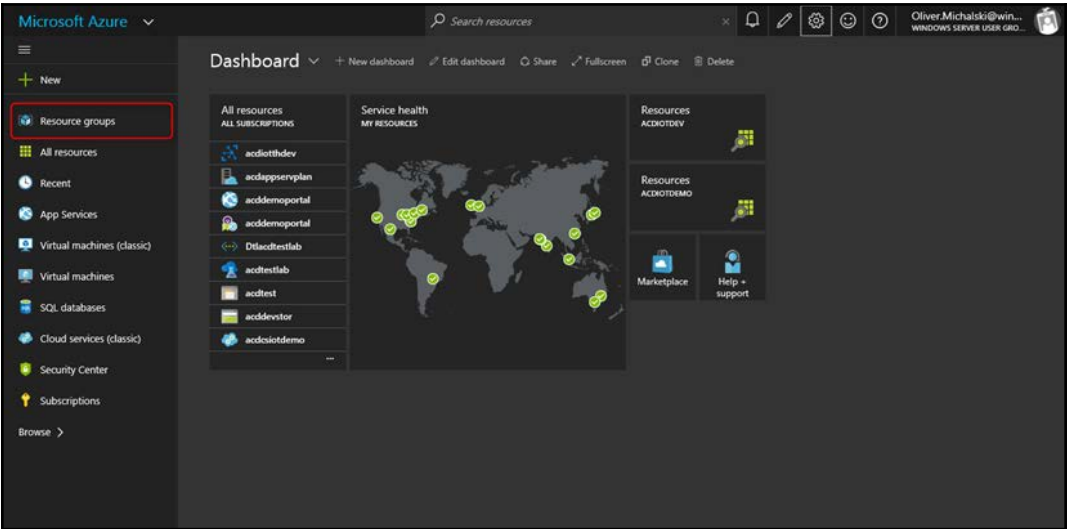
Key	Value
Department	Management
Project	ppbook
Tenant	ACD

Once you have defined a resource tag, you can use this as a filter in Azure PowerShell or in the Azure Billing APIs (Azure Usage API, Azure RateCard API). Up to 15 tags can be defined per resource.

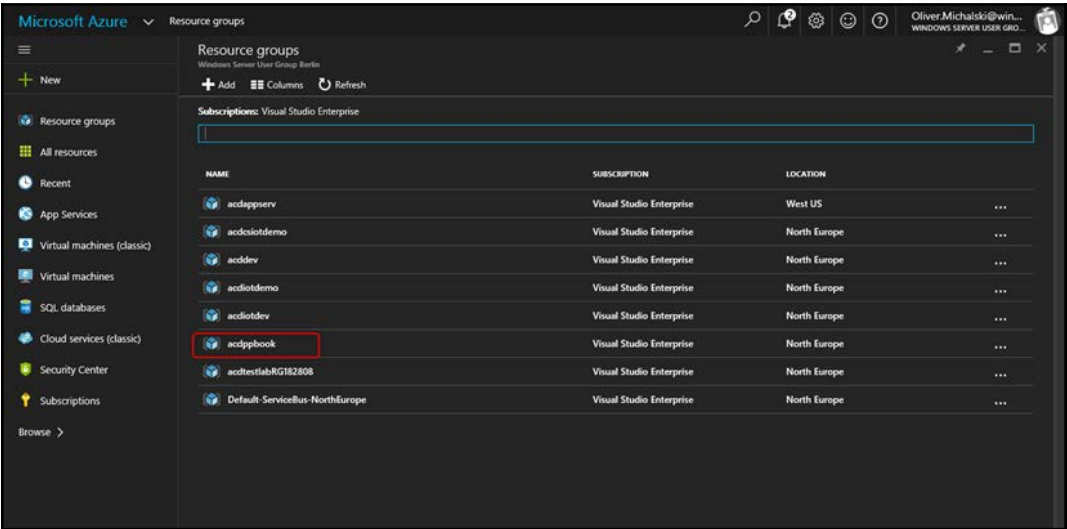
We will show you the necessary work steps on the example of tagging an Azure storage account, but note that the description of the procedure applies to all other resource types in the same form:

1. Open your Azure management portal at <https://portal.azure.com>.

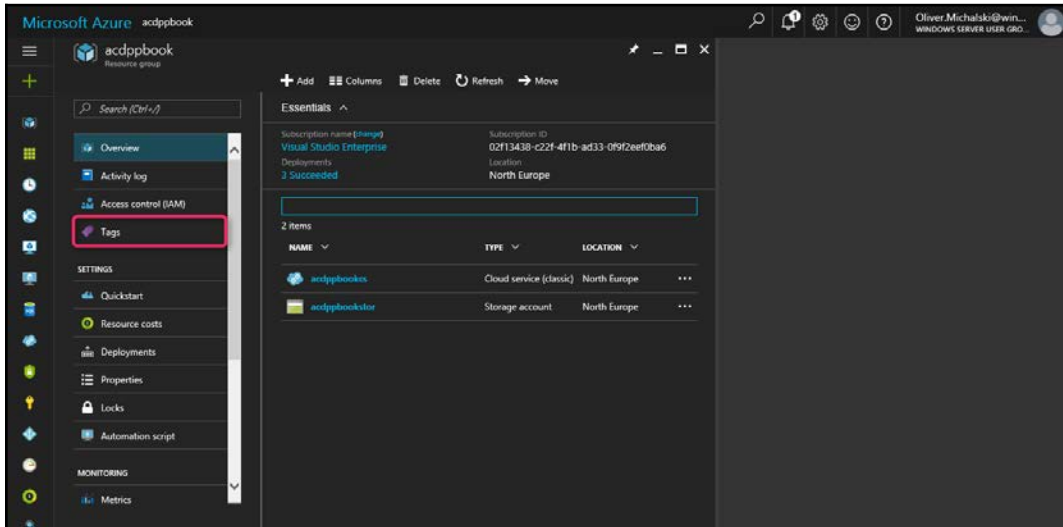
2. In the portal, click on **Resource groups**:



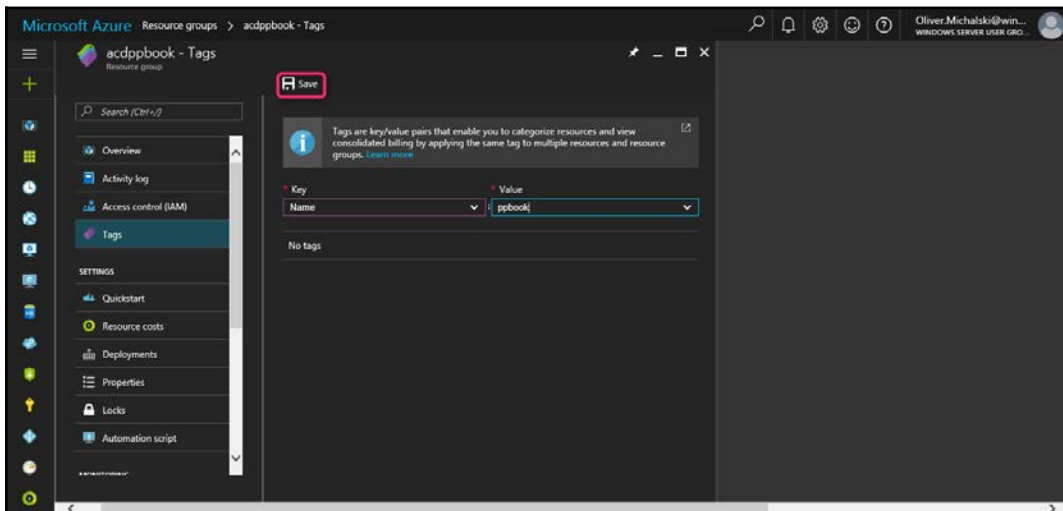
3. On the **Resource groups** blade, click the **acdppbook** name field:



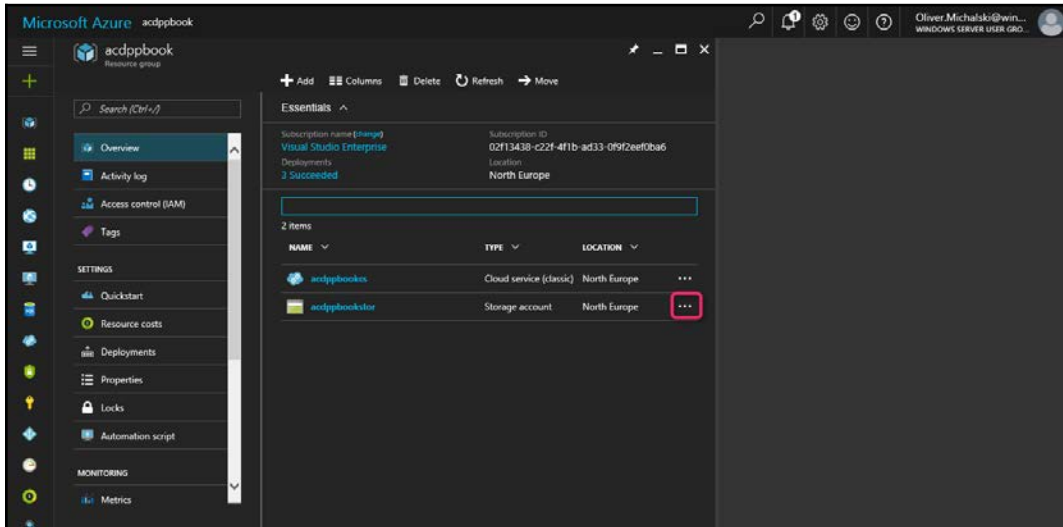
4. In the navigation section on the **Resource groups** dashboard, click on the **Tags** button:



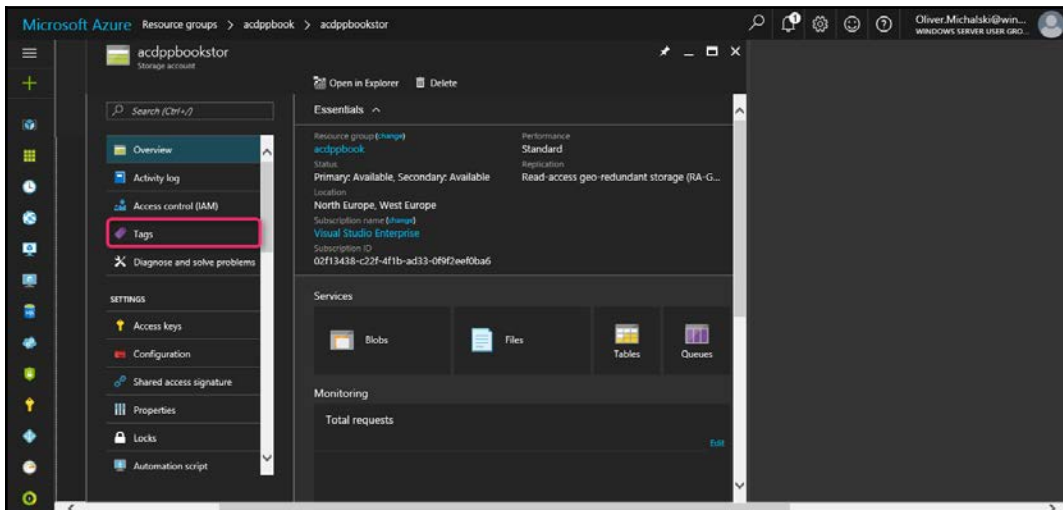
5. Now open the **Tags** blade. In the **Key** textbox, type Name, and then in the **Value** textbox, type ppbook. After this, click on **Save**:



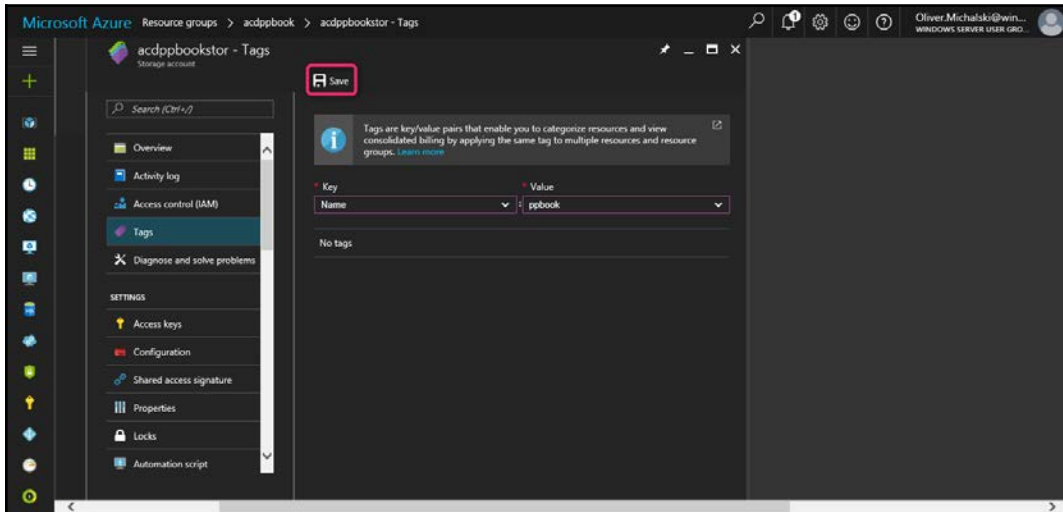
- Go back to the **Resource groups** dashboard. In the resource grid, select the **acdppbookstor** row, and then click the ... field:



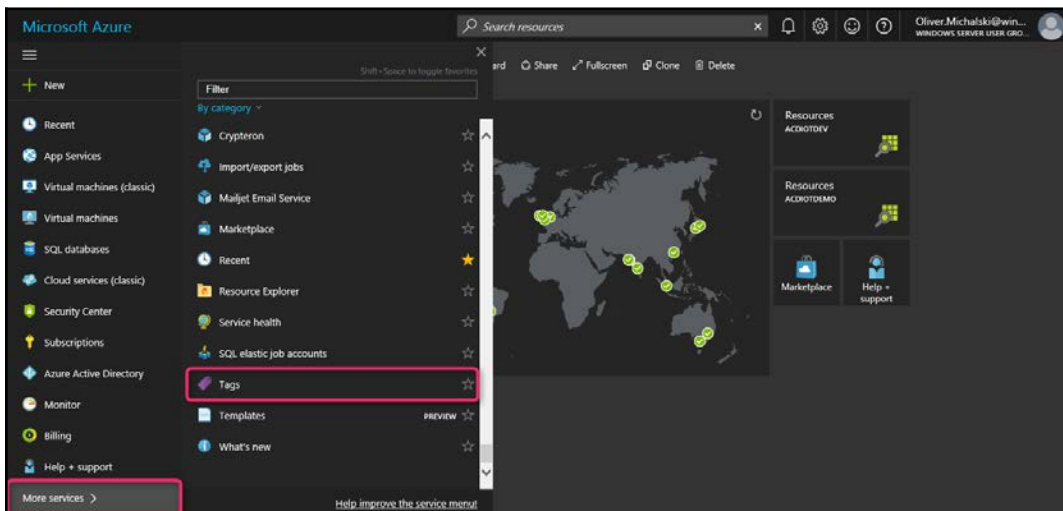
- In the navigation section of the **Resources** dashboard (**acdppbookstor**), click the **Tags** button:



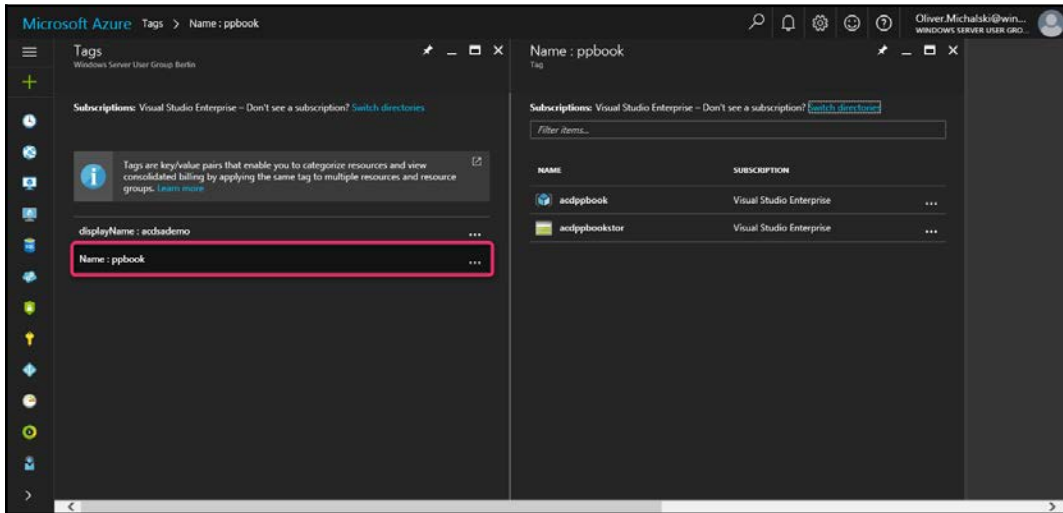
8. On the **Tags** blade, in the **Key** box, type again `Name`, and then in the **Value** box, type `ppbook`. Click **Save**:



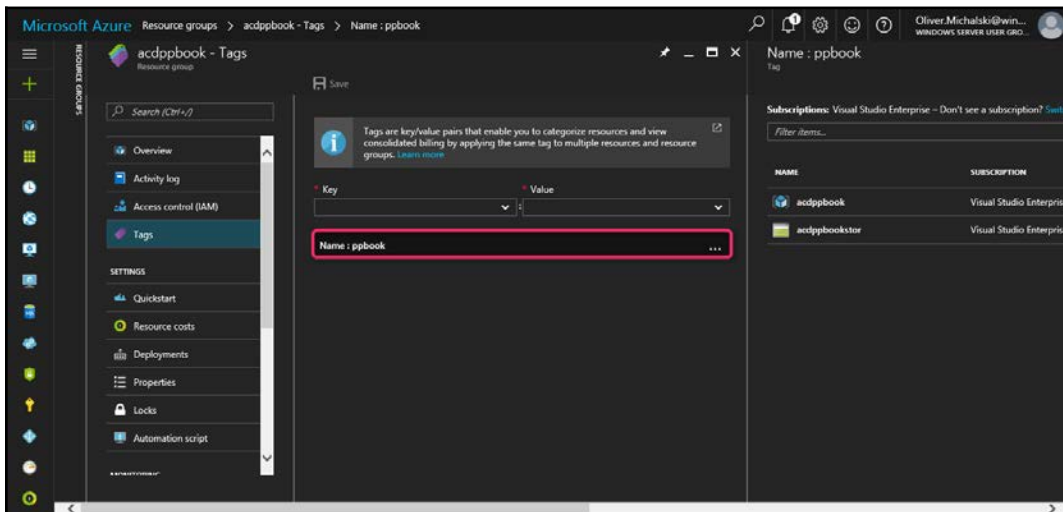
9. In the portal, click on **More services**, scroll down the list, and then click on **Tags**:



10. In the **Tags** blade, search the row for your tag **Name: ppbook**, and then click the ... field. Now you can see all the resources that are associated with the tag:



11. You can also see all the resources that are associated with the tag, by clicking the list field on the **Tags** dashboard of your resources:



Locking Azure resources

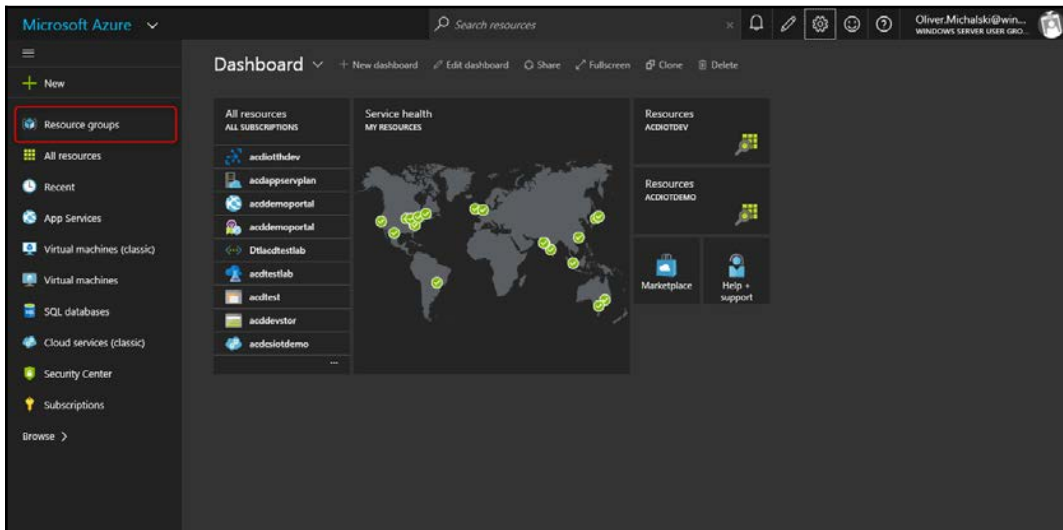
Now you know how to organize your resources, but for working with the resources, there is still another functionality that is important, which we will introduce now.

Azure Resource Locks

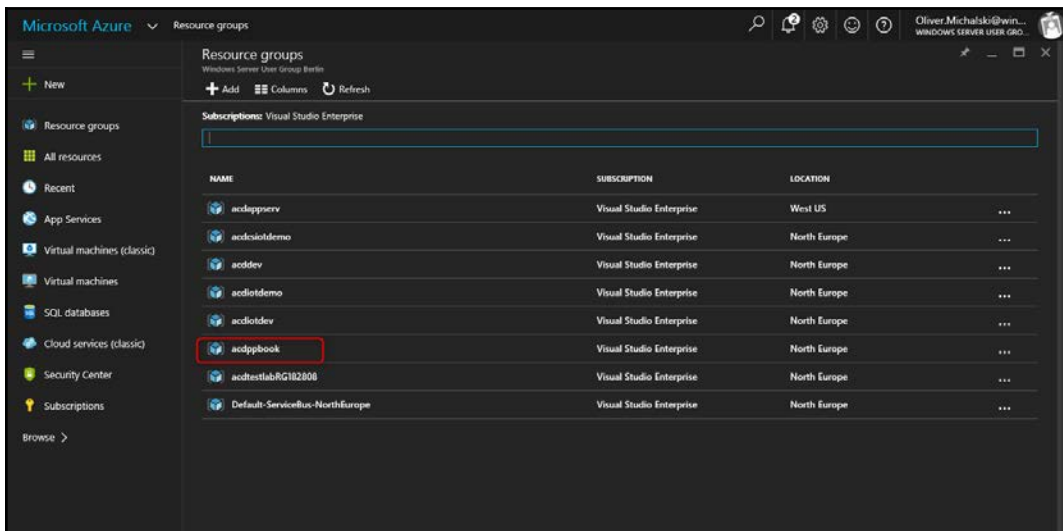
What does this mean? As an administrator, you may need to lock a resource group, or resource to prevent other users from accidentally deleting or modifying critical resources. The Azure resource manager offers a mechanism with two levels (**CanNotDelete** or **ReadOnly**) to be able to make appropriate settings.

Let's take a look at this:

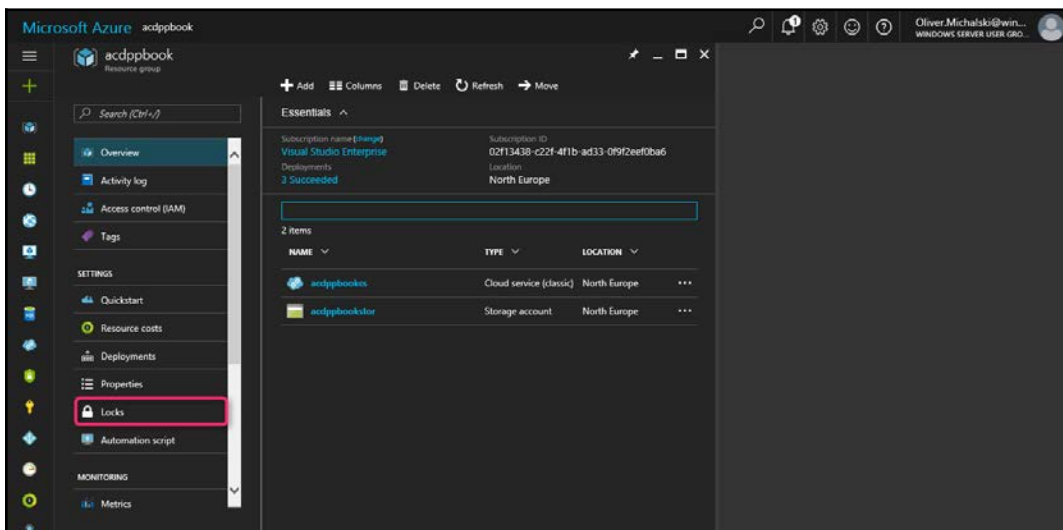
1. In the portal, click on **Resource groups**:



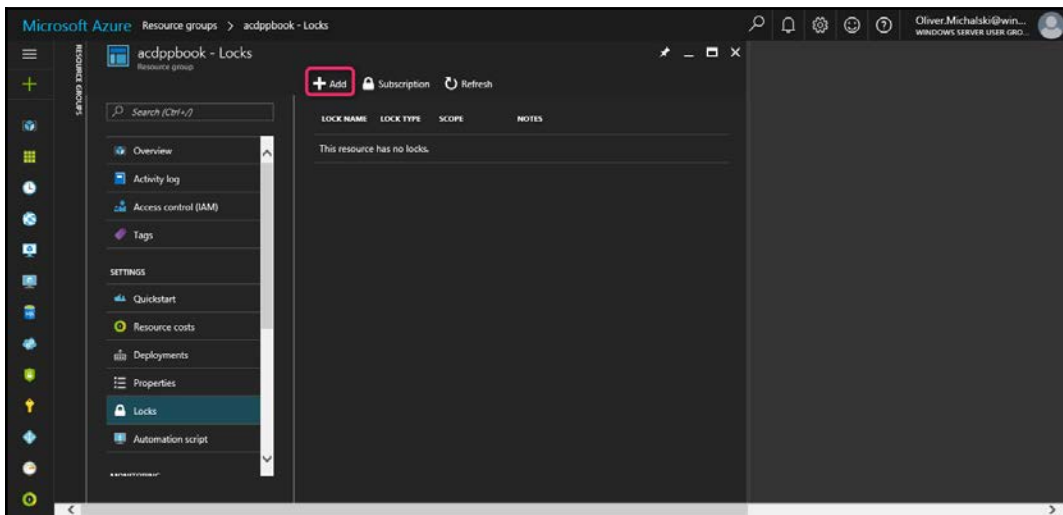
2. On the **Resource groups** blade, click the **acdppbook** name field:



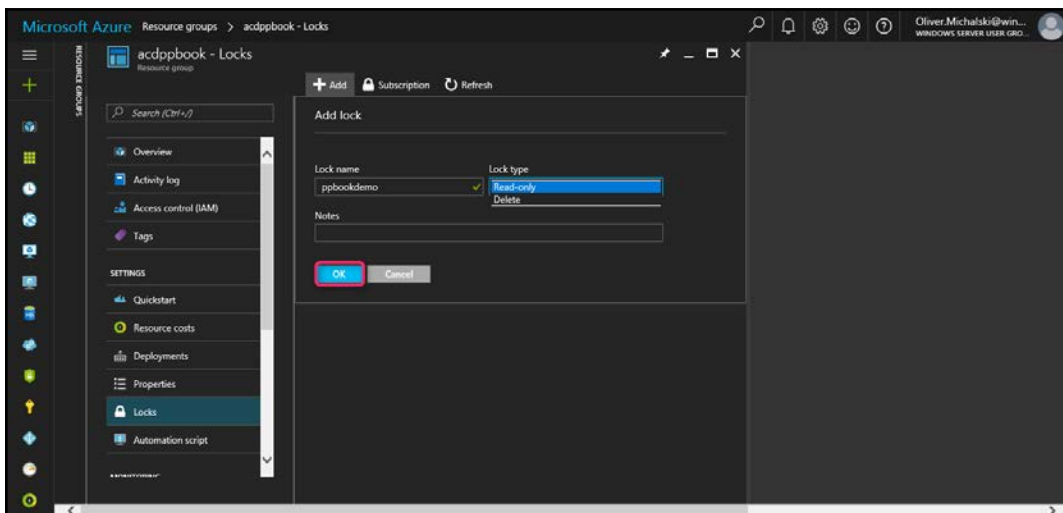
3. In the navigation section on the **Resource groups** dashboard, click on the **Locks** button:



4. On the **Locks** blade, click on the **Add** button:



5. Now type `ppbookdemo` in the **Lock name** textbox, select a **Lock type**, and click on **OK**:

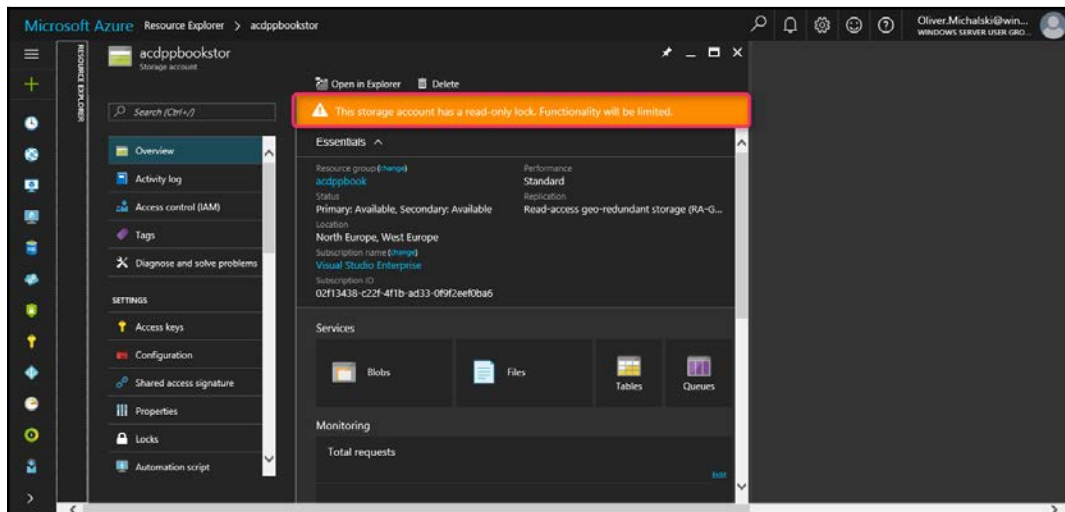


6. Your first lock is ready.

**Attention!**

We have put in our example **Read-only** as a **Lock type** on **Resource groups**. This lock is automatically inherited to all subordinate resources. This has the consequence, however, that the functionality of individual resource types is no longer guaranteed.

7. For example, no keys can be retrieved for a **Storage account** (this is a read and write operation). The operation is, however, mandatory for access:



Working with ARM Templates

Now we come to the more advanced section of the explanations on the subject of working with the Azure resource manager. The content of this section is also divided into two parts.

In part 1, we consider the issue from the view of an IT professional. IT pros are less interested in developing their own templates but want to reuse their existing deployments in template form.

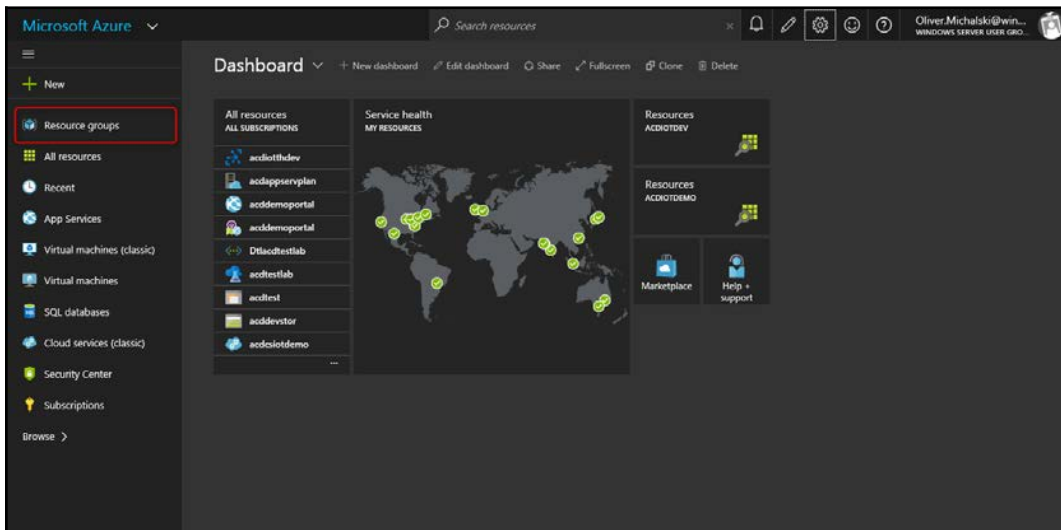
The corresponding feature in Azure resource manager is to export a deployment as an ARM template. This feature allows you to create templates of your deployment, download the templates to secure them in a source code repository, modifying your templates and of course the redeployment.

Part 2 (from the perspective of the developer) is easier to explain. This section deals with the topic of authoring an ARM template.

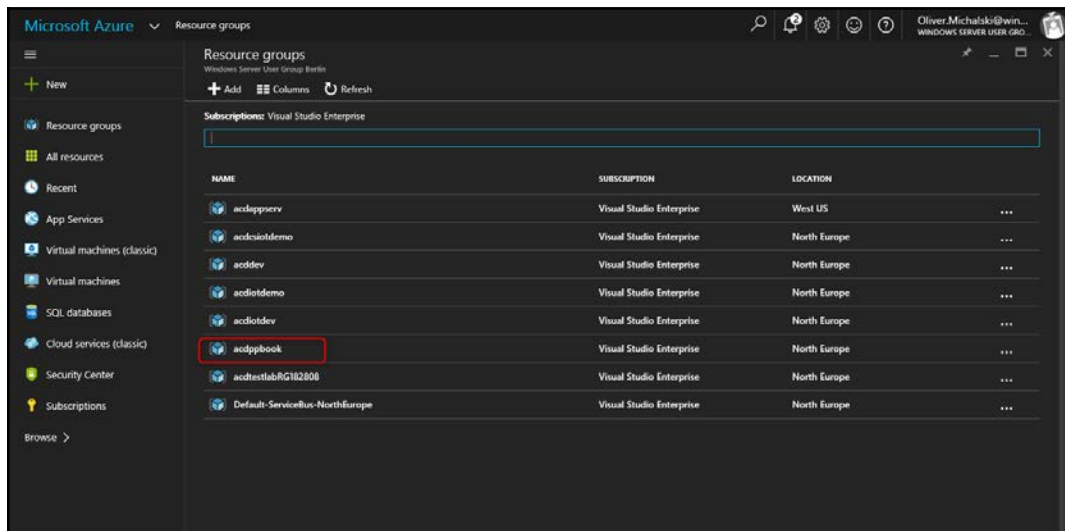
Exporting a Deployment as an ARM Template (for IT pros)

Do you always roll out the same basic configuration in Azure? Do you often realize demos or customer projects and always roll out the same basic configuration in Azure? Then the following procedure, in the future, can make the work easier: Open your Azure management portal at <https://portal.azure.com>.

1. In the portal, click on **Resource groups**:



2. On the **Resource groups** blade, click on the **acdppbook** name field:

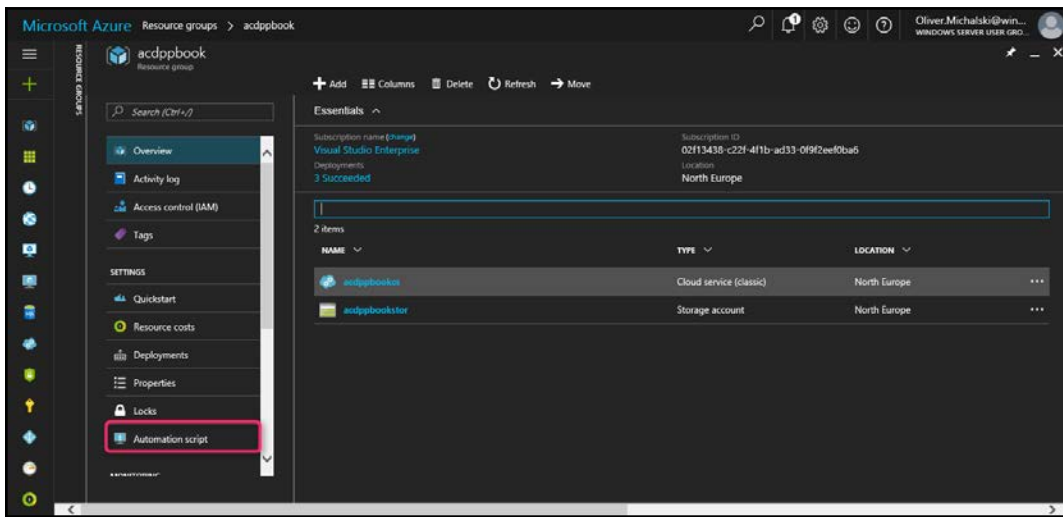


Now we will export our deployment to an ARM template.

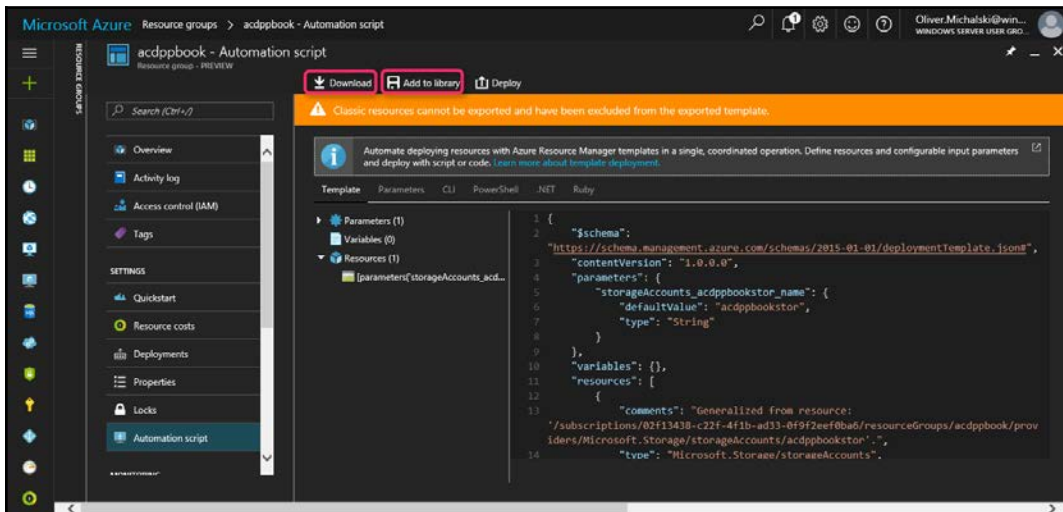
Example 1: Exporting a Resource Group to an ARM Template

To export a resource group, perform the following steps:

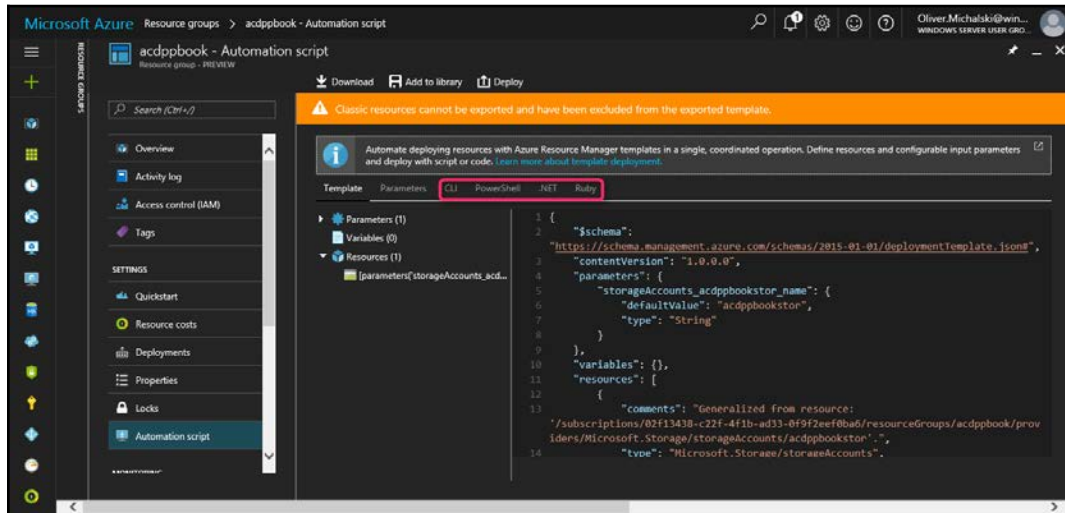
1. The first example is about the export of a complete resource group, that is a resource container and any number of other resources. To export a resource group, perform the following steps: In the navigation section of the Resource groups dashboard, click on Automation script, as shown:



2. Now you can see the resource group template. Click on **Download** to save the template to finalize your work:



3. In addition to the template in the JSON data format, special scripts (or classes) are also provided for Azure CLI, Azure PowerShell, NET, or Ruby. You will see this by pressing one of the links in the navigation area. These scripts help you to deploy the template:

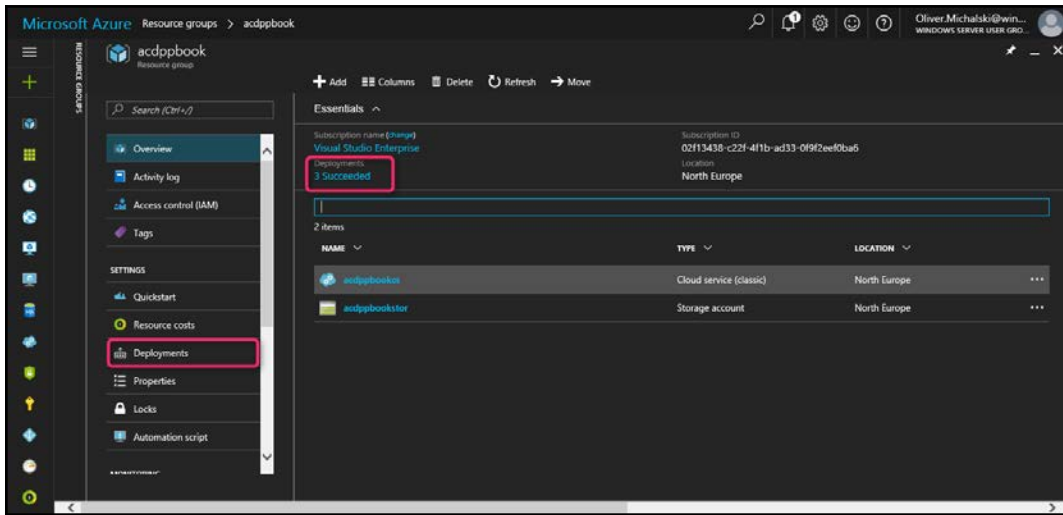


Not all resource types support the export template function. If your resource group only contains a storage account, a virtual machine, or a virtual network, you will not see an error. However, if you have created other resource types, you may see an error stating that there is a problem with the export.

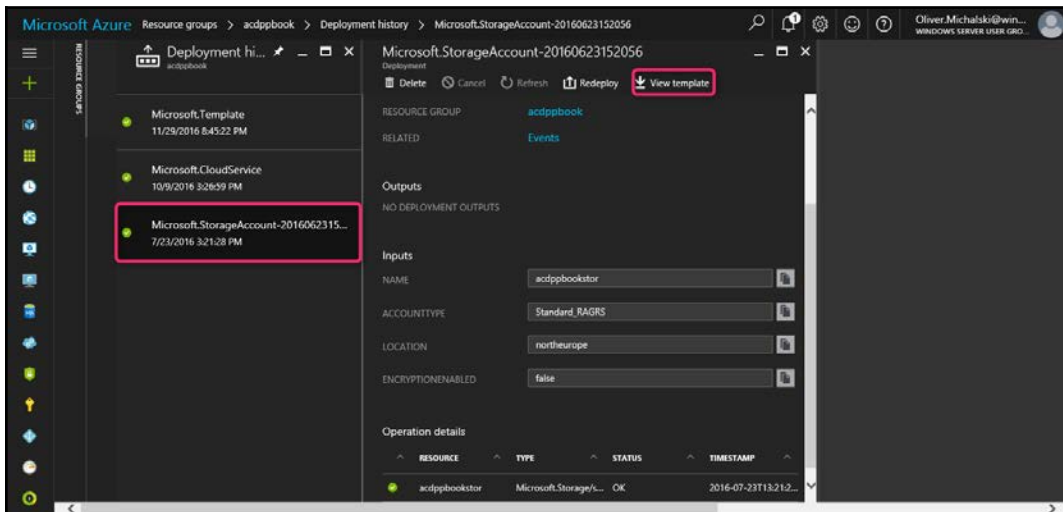
Example 2: Exporting a Resource (from the Deployment History) to an ARM Template

In the second example, we only want to export a single resource (app, database, etc.). For this, we usually use the deployment history list to select the version that works best for us. To export a resource from the deployment history, perform the following steps:

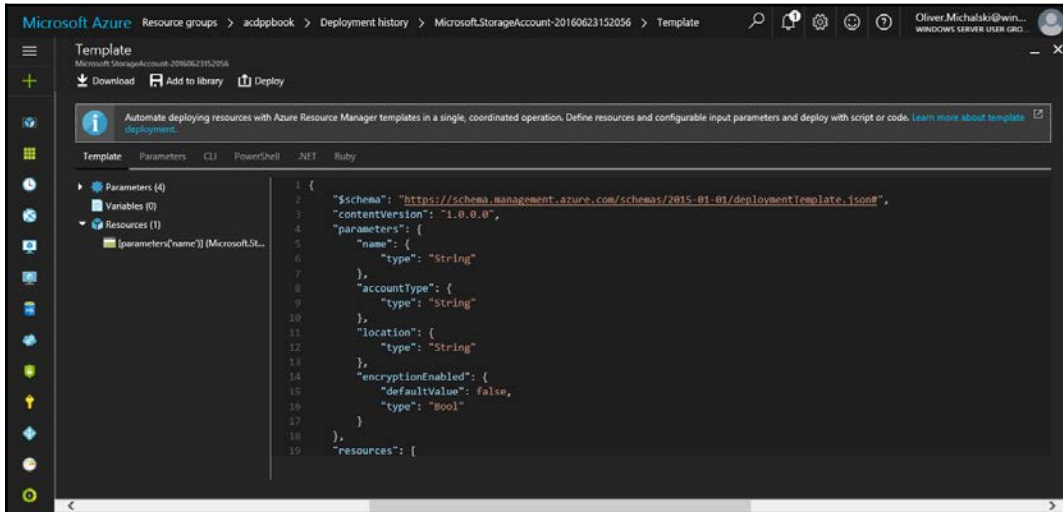
1. Go back to the **Resource group** dashboard for **acdpbook**. In the navigation section, click on **Deployments** or use the link **Deployments**, shown in the field **Essentials**:



2. Select the deployment that you want to export. Click on the field. Now the deployment properties site opens. Click on **View template**:



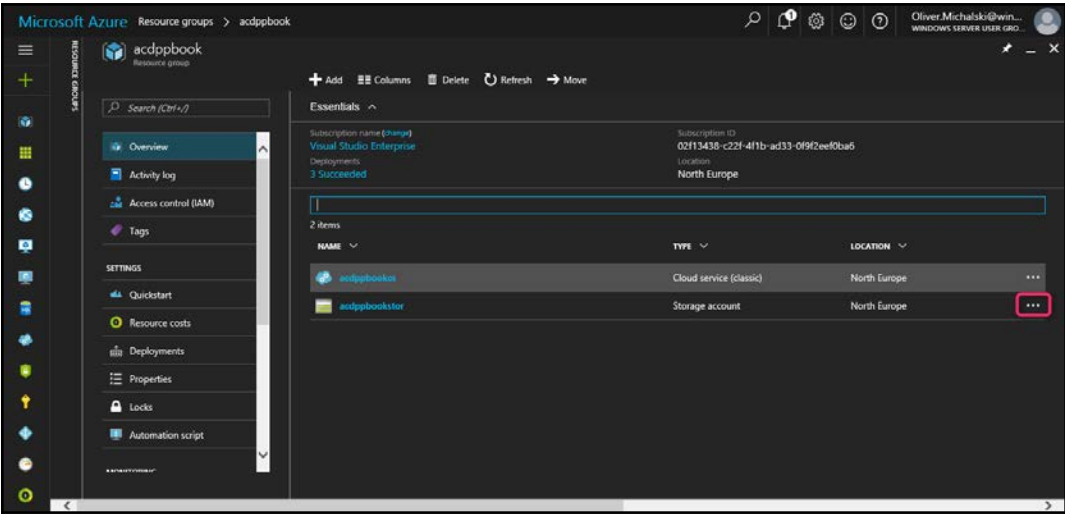
3. Now you can see the resource template. Click on **Download** to save the template to finalize your work:



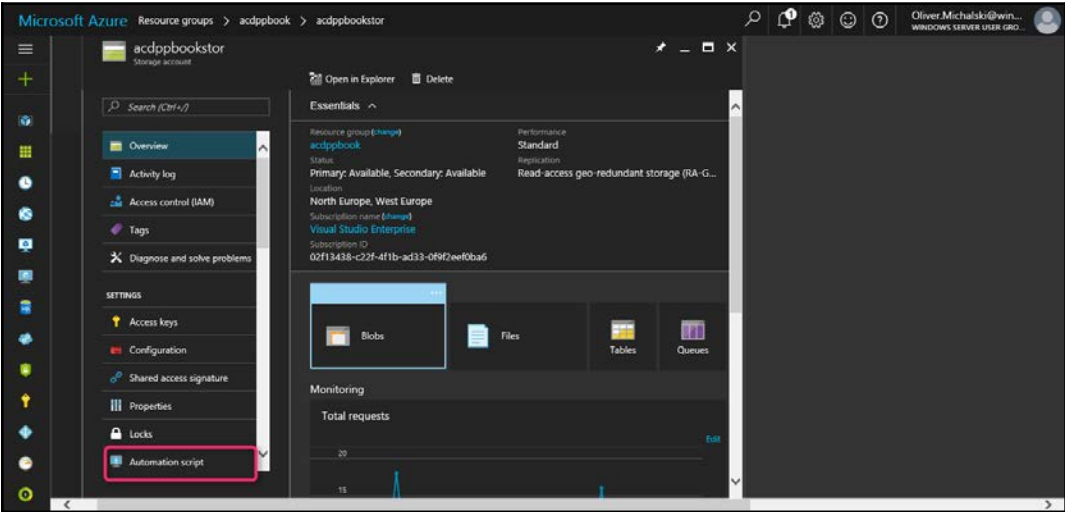
Example 3: Exporting a Resource (Classic) to an ARM Template

Also in the third example, we want to export a single resource (app, database, etc.). This time, however, in the so-called classical way. This is different from *Example 2* by the fact that only the most recent version of the resource is exportable. To export a resource on the classical way, perform the following steps:

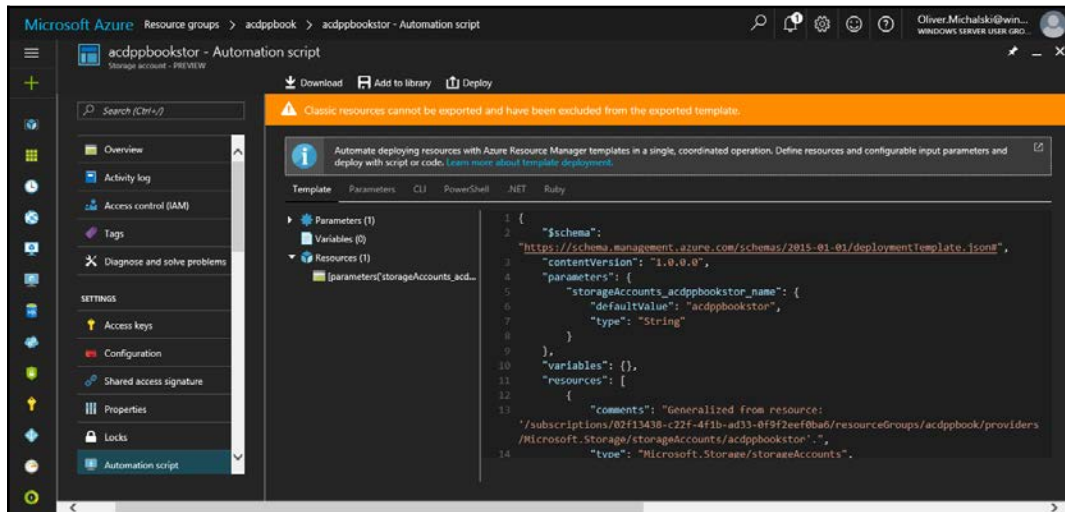
1. Go back to the **Resource groups** dashboard for **acdppbook**. In the resource grid, select the **acdppbookstor** row, and then click the ... field:



2. On the **Resource** dashboard, click on **Settings**, and then click on **Automation script**:



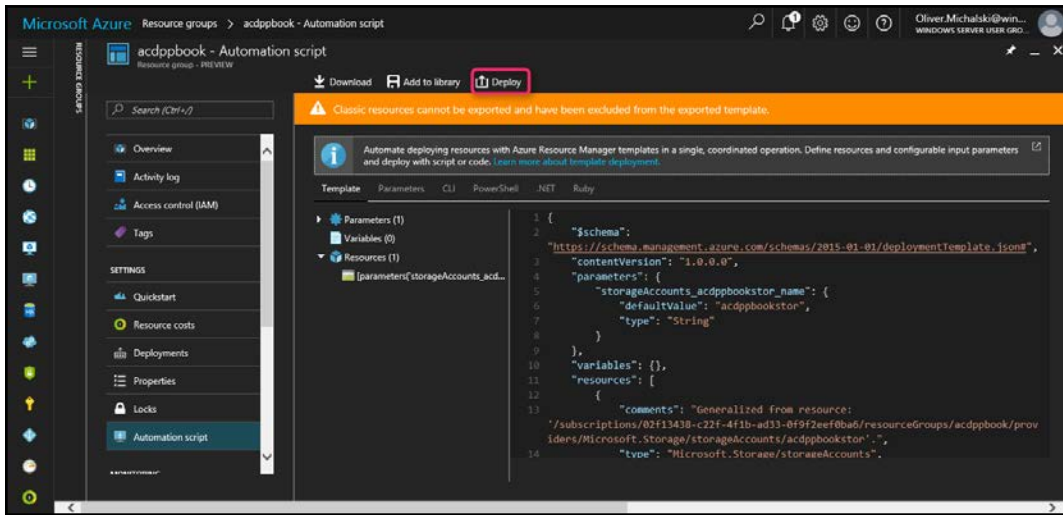
3. Now you can see the resource template. Click on **Download** to save the template to finalize your work:



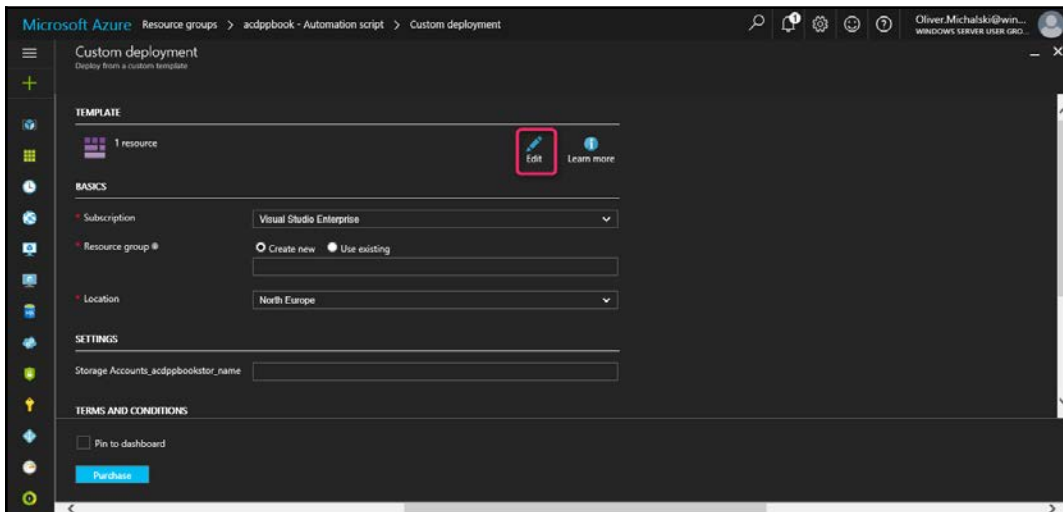
Modifying an ARM Template

We have finished the topic of exporting to an ARM template. Now the question arises: what is missing? As mentioned earlier, the feature supports the ability to redeploy and modify templates. We shall take a look now:

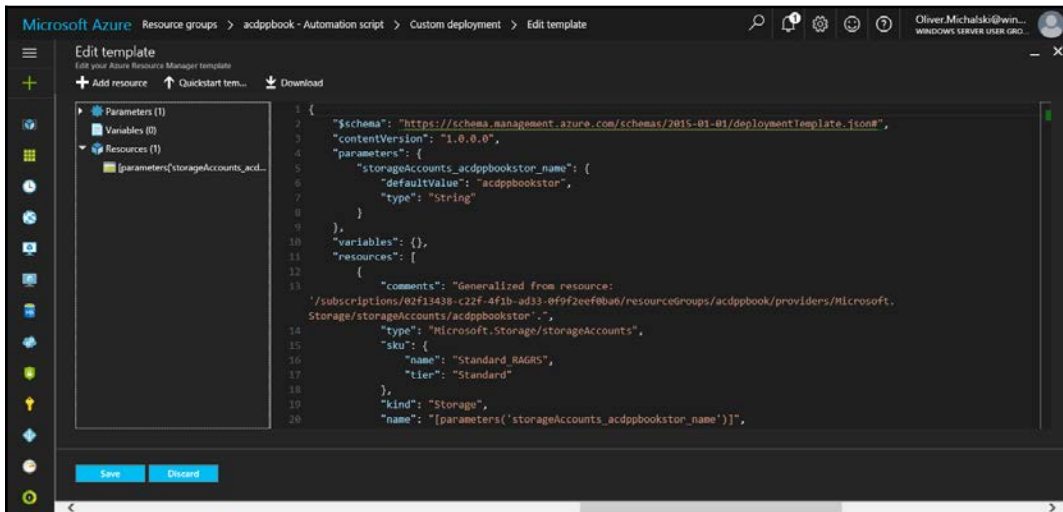
1. Go back to the **Export resource group template** view. Click on **Deploy**:



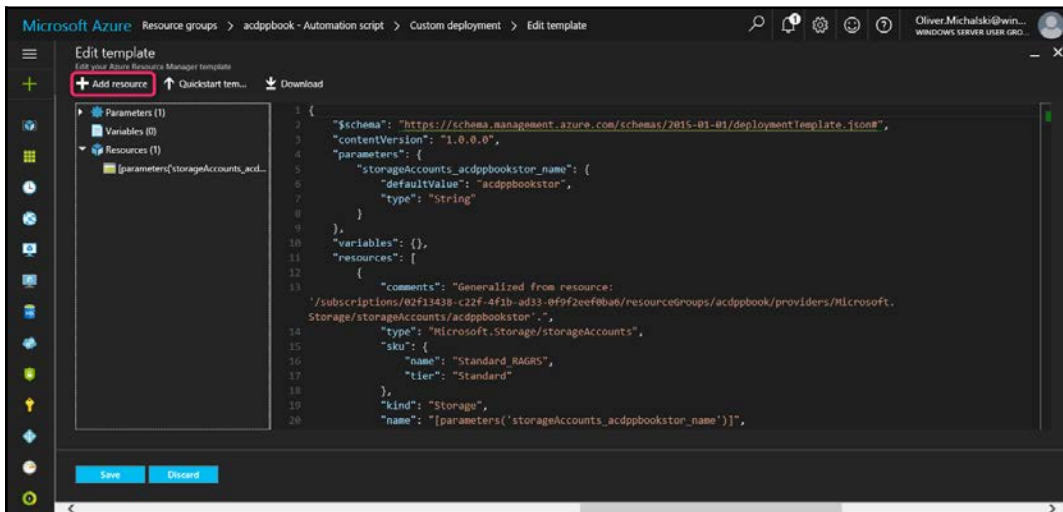
- Next, the **Custom deployment** blade opens. This time, there is one resource available. To start the working process, you must press the **Edit** button:



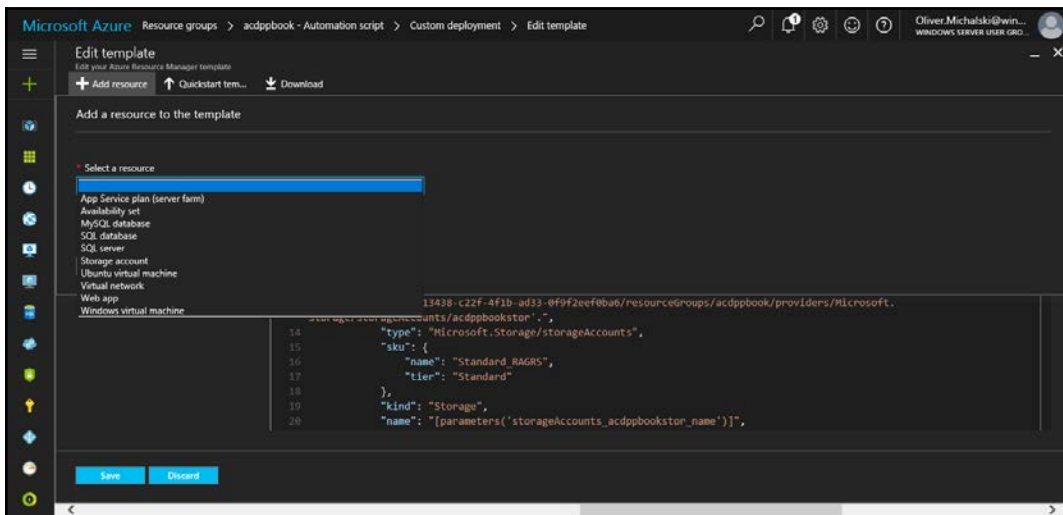
3. The next window is the **Edit template** view. Here you can edit your template directly in the text:



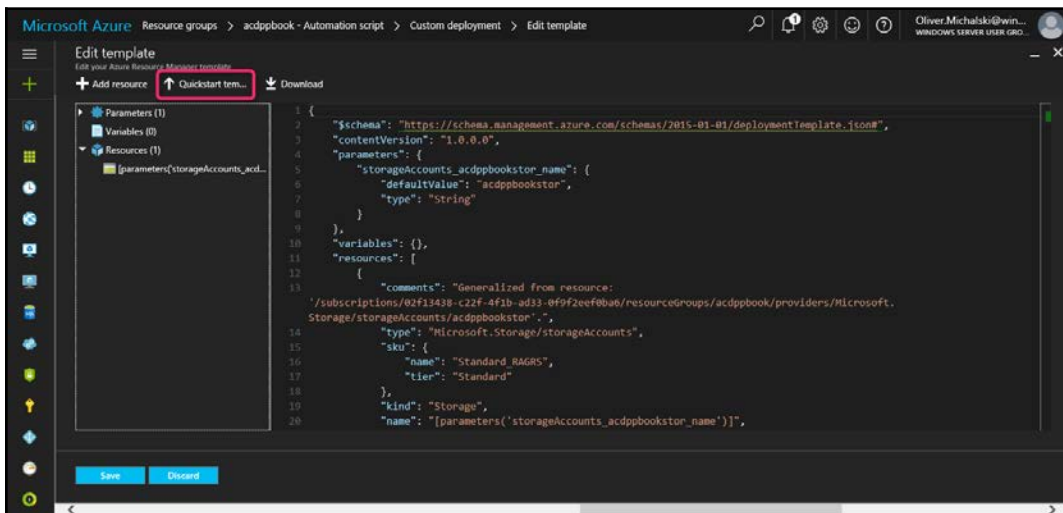
4. You can add another resource to your template (for example, a virtual network to a virtual machine). Click on **Add resource** and then select the desired resource from the list:



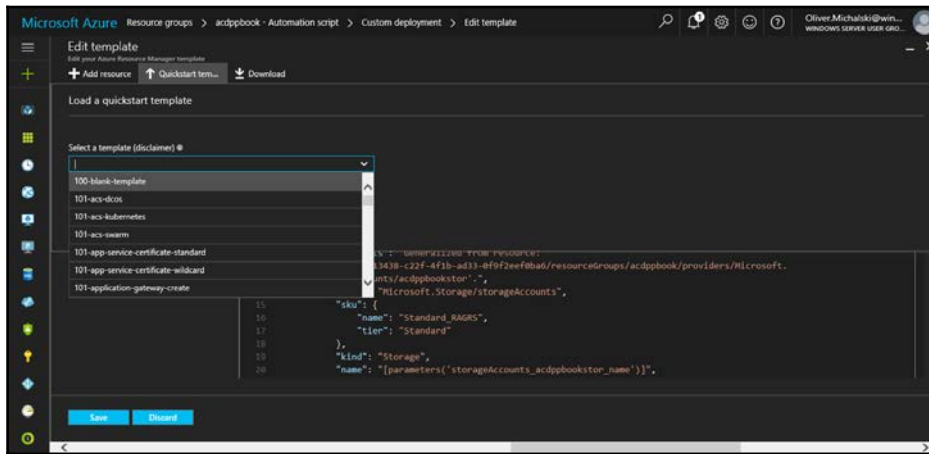
5. You can select any resource from the **Select a resource** drop-down list:



6. You can also take a **Quickstart template** as a reference for your work. Click on **Quickstart template** and then select the desired template from the list:



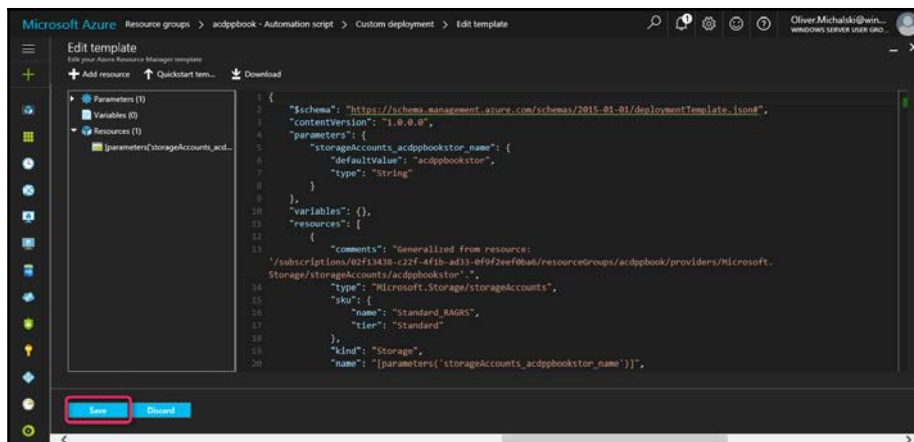
7. You can select any desired quick start template from the **Select a template** drop-down:



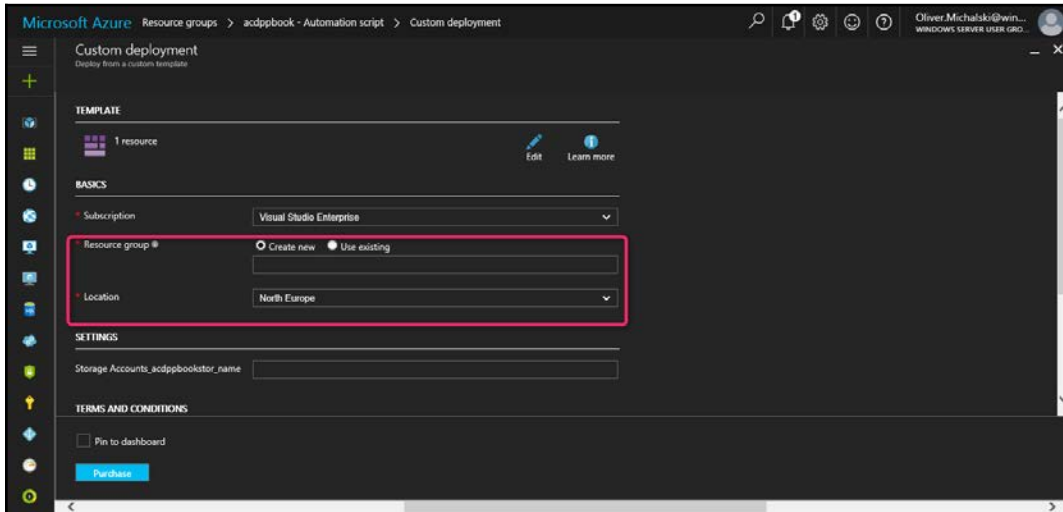
Azure **Quickstart template** are a collection of ARM community templates (with solutions for many workloads) and you can find them

here: <https://github.com/Azure/azure-quickstart-templates>.

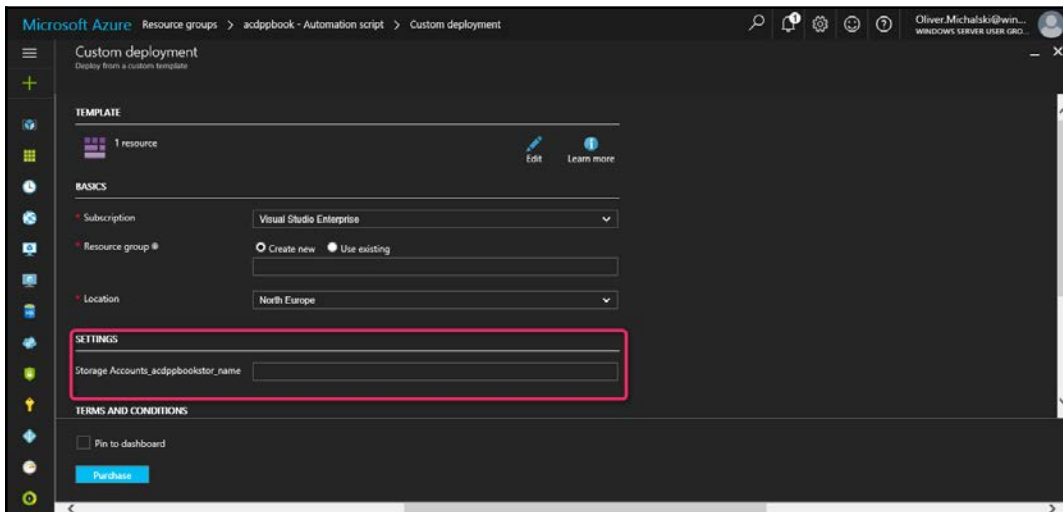
8. When everything is ok, press the **Save** button:



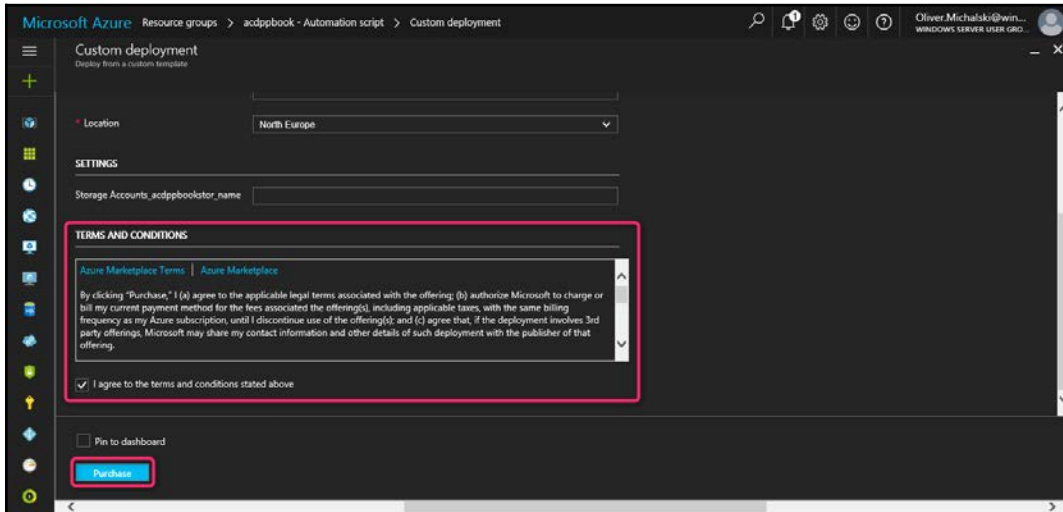
9. We are back in the **Custom deployment** blade. Here you have to make a few final entries. First, define a **Resource group** and a **Location**:



10. In the **SETTINGS** section, you must type a new name for your resource:



11. Finally, you must accept the **Azure Marketplace Terms**, and then press the **Purchase** button:



Authoring an ARM Template

Before you can create a template, some planning tasks are required:

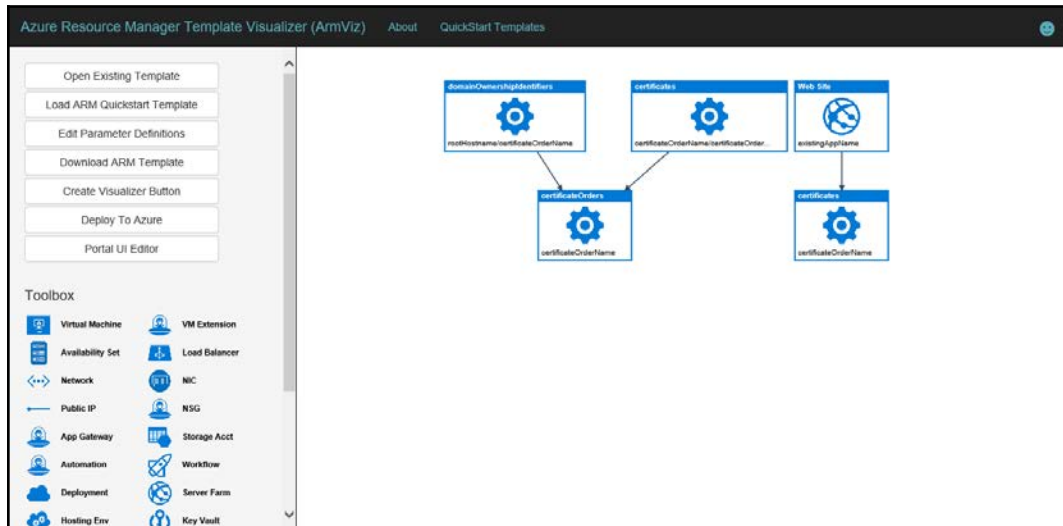
- What are the types of resources to be provided?
- What are the locations of resources?
- What is the version of the resource provider API used to deploy resources?
- Must some resources be provided for other resources?
- What values should be passed during the deployment, and what values would you define directly in the template?
- Do you need to return values from the deployment?

Once you have answered these questions, you can get started. Ok, at least theoretically, you can start. Before you can start, we should clarify the software requirements.

Because a JSON data file is an XML-based text file, you really need only a simple text editor (for example, Notepad). We recommend you still use Visual Studio (with an excellent JSON editor). To prevent unnecessary expense, the Visual Studio Community edition is sufficient for our purposes completely.

There is still an alternative available - **Azure Resource Manager Template Visualizer (ArmViz)**. With ArmViz, you can create templates with purely graphic means. ArmViz has one weakness, however - it is a preview and only available outside the platform.

Let's take a look at it:

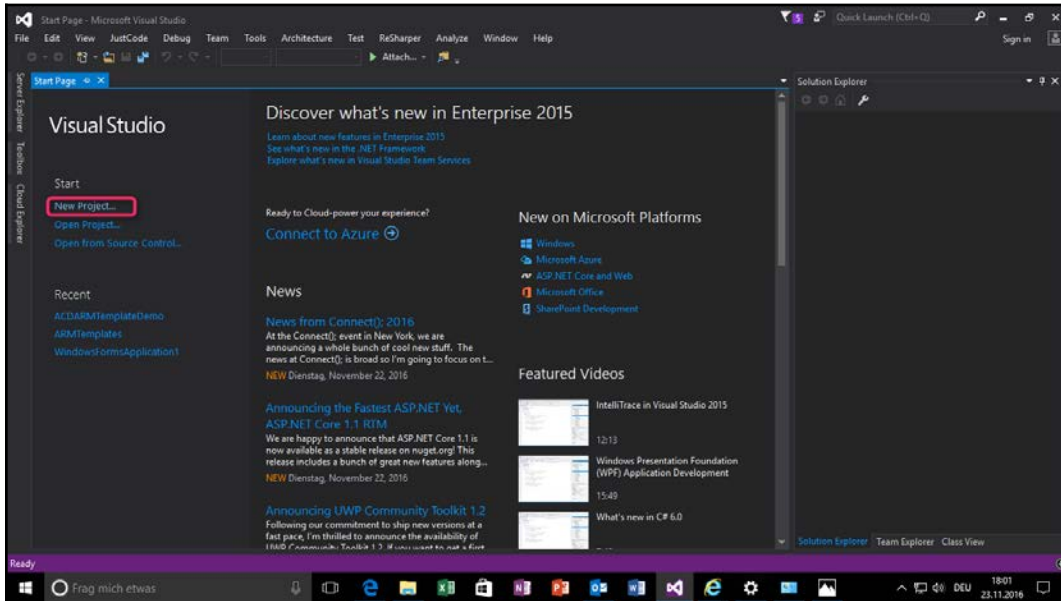


You can find ArmViz at <http://armviz.io/#/>.

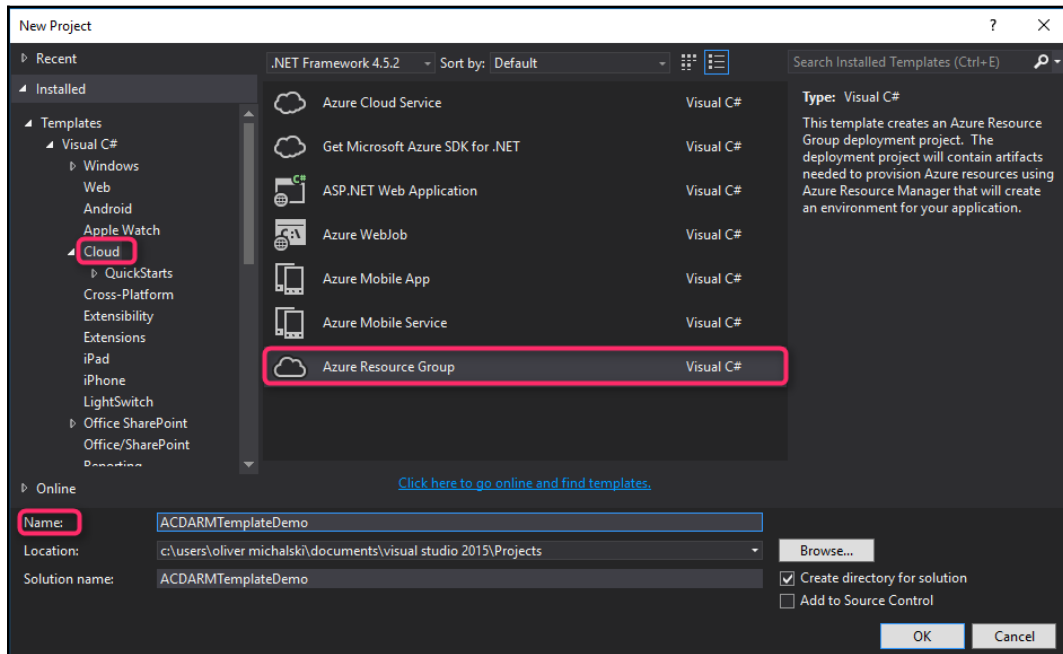
Creating your own ARM Template (for Developers)

To create an ARM template, perform the following steps:

1. Open your Visual Studio, and click on the **New Project...** link:



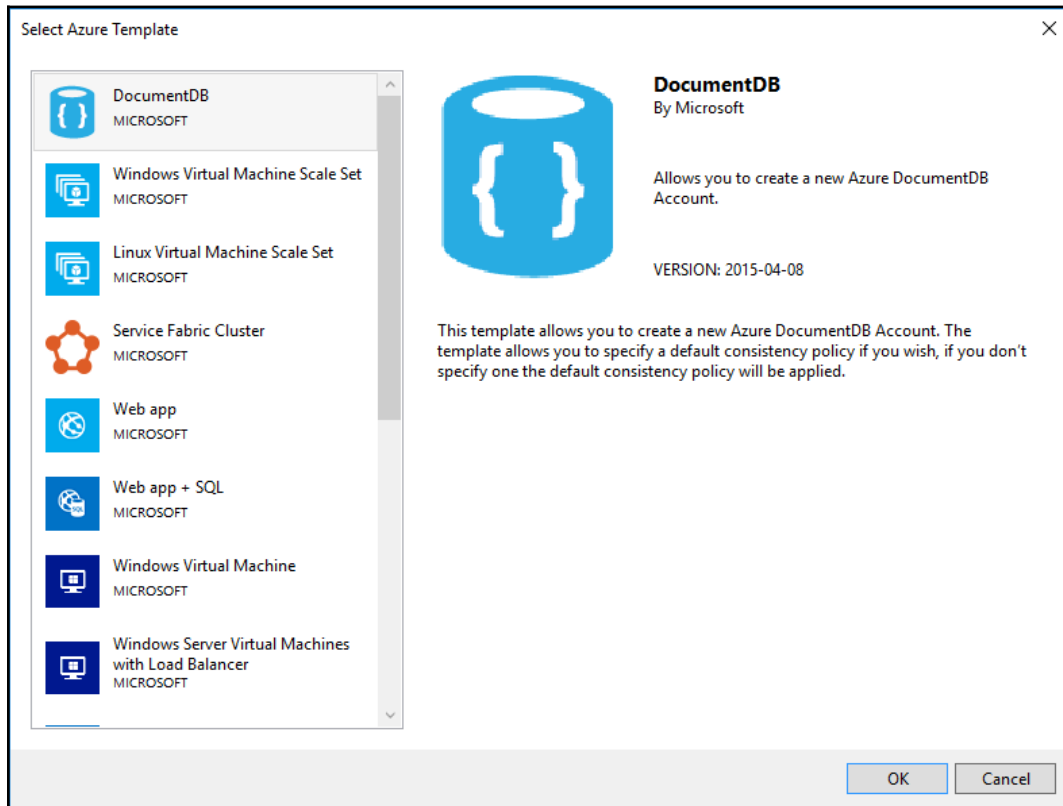
2. Now the selection dialog opens with the available project templates. The required template, **Azure Resource Group**, can be found in the **Cloud** area:



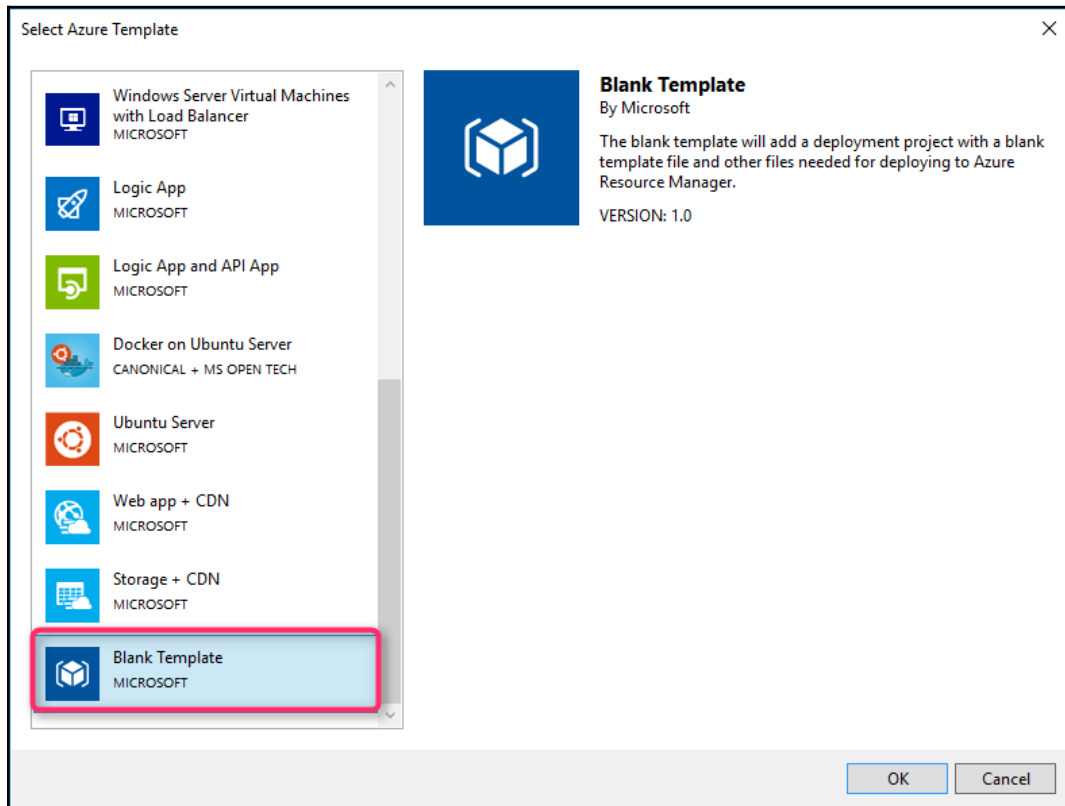
If you do not find the entry there, you have forgotten to install the Azure SDK and Azure VS tooling.

3. If everything is clear, specify a project **Name**, for example, `ARMTemplateDemo`, and press the **OK** button.

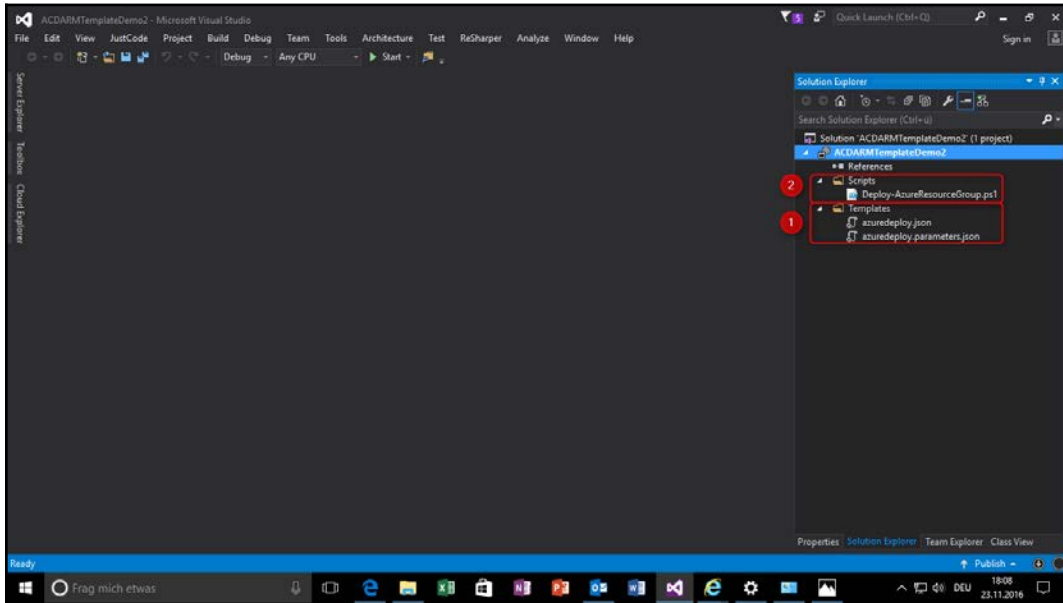
4. Now another selection dialog opens, this time with a list of available Azure templates:



5. The template **Blank Template** that we need for our demo can be found as the last entry in the list. Select the entry and press the **OK** button:



6. Wait briefly until the project has been loaded. You should now see the following screen:



The current project consists of three artifacts in two solution folders, Templates and Scripts.

In the Templates folder, you will find two JSON files:

- `azuredeploy.json`
- `azuredeploy.parameters.json`

`azuredeploy.json` is the template for the real template and `azuredeploy.parameters.json` is used as a store for the required parameter values. The name `azuredeploy` you can change in any way. but the endings `.json` and `.parameters.json` must be maintained. In the Scripts folder, you will find a PowerShell script that will help you to perform the final deployment..

Let's have a look at `azuredeploy.json`:

```
{
  "$schema":
    "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "variables": {},
  "resources": [],
  "outputs": {}
}
```

This is the simplest form of an ARM template but we should like to point out that in this form, the template is not valid and cannot be executed.

Here is the reason, some of these fields are required and others are optional. The following table shows whether a field should be filled in:

Field	Required
<code>\$schema</code>	Yes
<code>contentVersion</code>	Yes
<code>parameters</code>	No
<code>variables</code>	No
<code>resources</code>	Yes
<code>output</code>	No

As you can see, the fields `$schema`, `contentVersion`, and `resources` need to be filled in. For the fields `$schema` and `contentVersion`, you can continue to use the pre-enclosed values, so you must add at least one resource.

Now let's add a resource. Once again, we chose a storage account as the resource type. The relevant section in the `azuredeploy.json` file looks like this:

```
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
    "name": "[parameters('storageAccountName')]",
    "apiVersion": "2015-06-15",
    "location": "[resourceGroup().location]",
    "properties": {
      "accountType": "Standard_LRS"
    }
  }
]
```

There's one thing that needs to be pointed out: the use of parameters is not necessary but without parameters, your template would always deploy the same resources with the same names, locations, and properties.

In order to avoid this situation, in the presented code segment, a parameter for the resource name is used.

For the parameters, we have to provide the corresponding definition. The relevant section in the `azuredeploy.json` file looks like this:

```
"parameters": {
  "storageAccountName": {
    "type": "string",
    "metadata": {
      "description": "Storage Account Name"
    }
  }
}
```

The template is now ready, valid, and executable. The complete code for `azuredeploy.json` looks like this:

```
{
  "$schema":
    "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageAccountName": {
      "type": "string",
      "metadata": {
        "description": "Storage Account Name"
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "name": "[parameters('storageAccountName')]",
      "apiVersion": "2015-06-15",
      "location": "[resourceGroup().location]",
      "properties": {
        "accountType": "Standard_LRS"
      }
    }
  ]
}
```

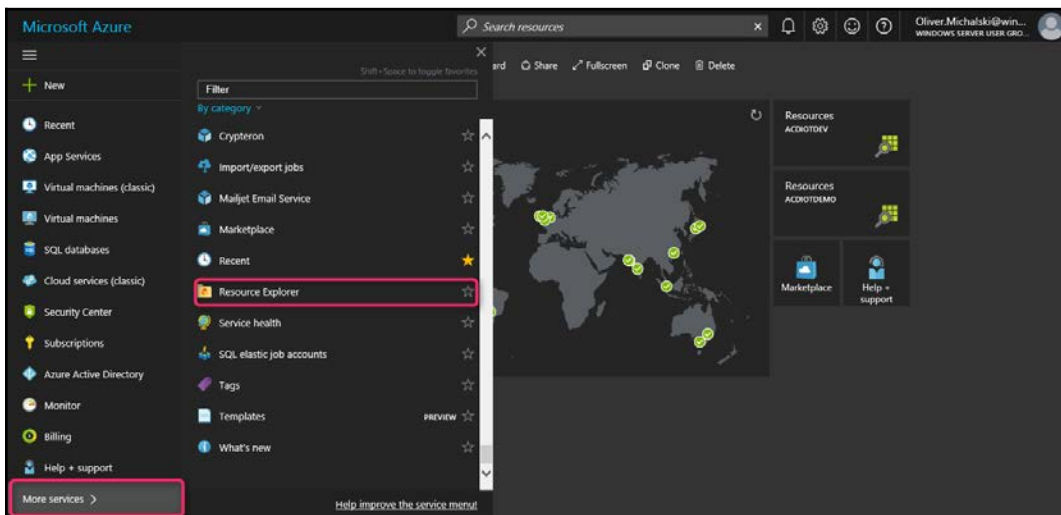
Azure Resource Explorer

The authoring of an ARM template is a difficult process. Especially when it comes to the individual components of the template and the use of valid values, many users have an information gap. But also for this problem, there is a solution on the Azure platform: the Azure resource explorer.

The Azure resource explorer is a tool for looking at your resources or at the resources of the Azure platform. By using this tool, you can see how the resources are structured, and it enables you to view the properties for resources.

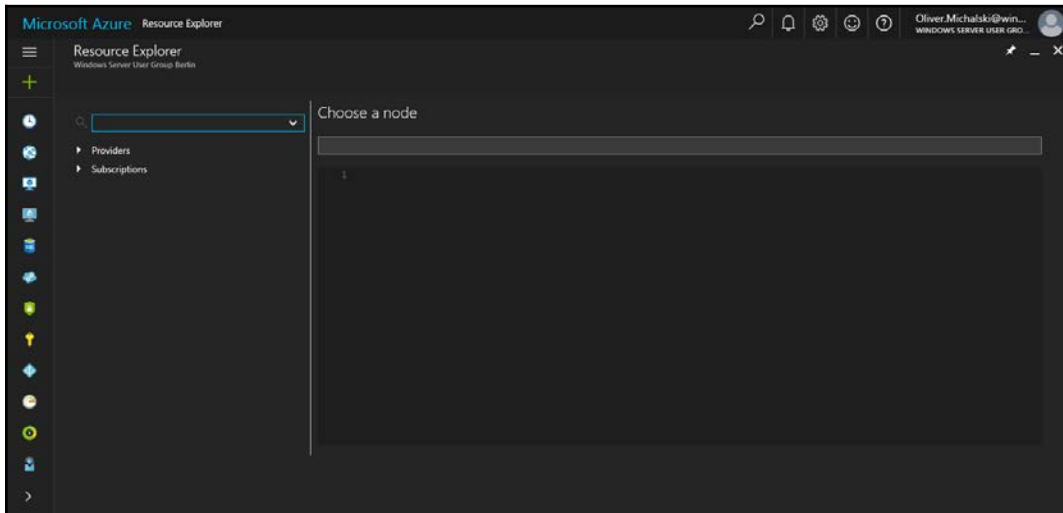
How can we find the Azure resource explorer?

1. Open your Azure management portal at <https://portal.azure.com>.
2. In the portal, click on **More services**, scroll down the list, and then click the **Resource Explorer** button:

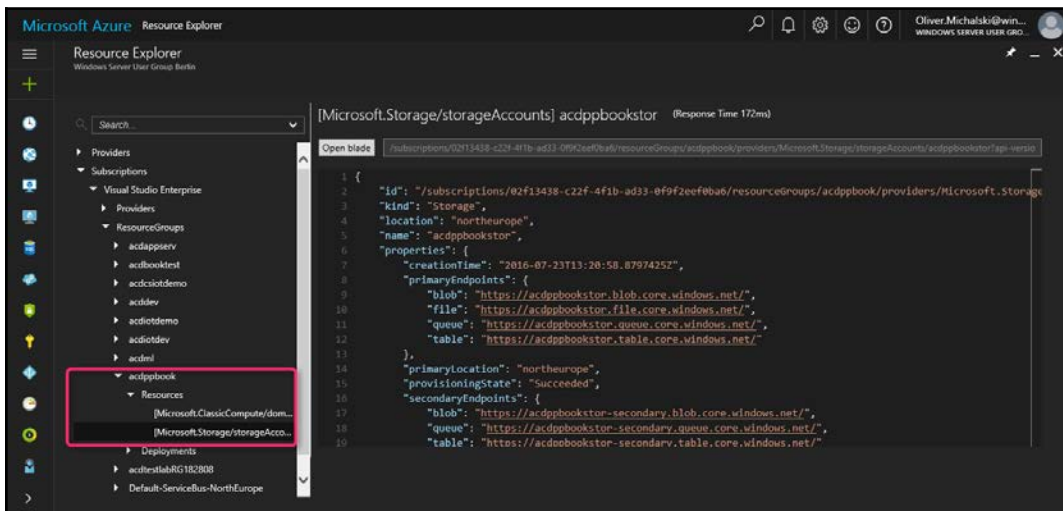


3. In **Resource Explorer**, you can find two nodes:

- **Providers** (the resources from the Azure platform itself)
- **Subscriptions** (your resources):



4. By clicking the nodes, you can search for the information you want:



Summary

In this chapter, we learned all about the Azure resource manager and therefore the basics for implementing Azure solutions.

In the first part of this chapter, we saw the creation of an Azure resource group, how to add resources to this group, and last but not least, how to organize the work with the resources in this group.

In the second part of this chapter, there followed detailed information for developers and IT professionals.

In the following chapters, we will use this knowledge and deal with the concrete implementation.

3

Deploying and Synchronizing Azure Active Directory

As the quantity of cloud services increase, identity management and security, as well as access policies within cloud environments and cloud services, is becoming even more essential and important.

The Microsoft central instance for this is Azure Active Directory. Every security policy or identity Microsoft provides for their cloud services is based on Azure Active Directory.

Within this chapter, you will learn the basics about Azure Active Directory, how you implement Azure AD and hybrid Azure Active Directory with connection to Active Directory Domain Services.

We are going to explore the following topics:

- Azure Active Directory overview
- Azure Active Directory subscription options
- Azure Active Directory deployment
- Azure Active Directory user and subscription management
- How to deploy Azure Active Directory hybrid identities with Active Directory Domain Services
- Azure Active Directory hybrid high availability and non-high availability deployments

Azure Active Directory

Azure Active Directory or Azure AD a multi-tenant cloud based directory and identity management service developed by Microsoft.

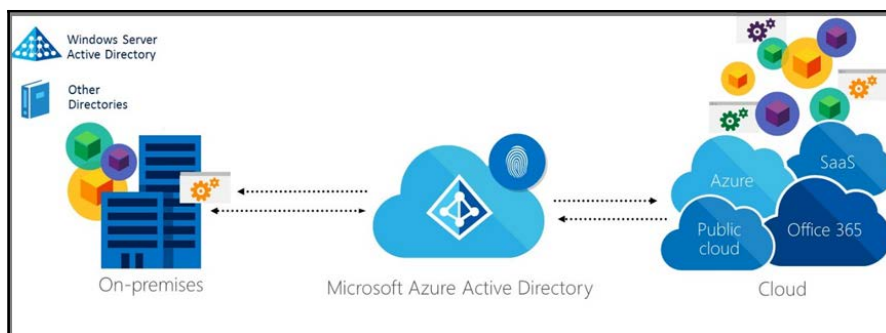
Azure AD also includes a full suite of identity management capabilities including:

- Multi-factor authentication
- Device registration
- Self-service password management
- Self-service group management
- Privileged account management
- Role based access control
- Application usage monitoring
- Rich auditing
- Security monitoring and alerting

Azure AD can be integrated with an existing Windows Server Active Directory, giving organizations the ability to leverage their existing on-premises identities to manage access to cloud based SaaS applications. An organization is also able to easily implement Single Sign-On on a multi factor authentication through Azure AD without adding third party software into their environment.

After this chapter you will know how to setup Azure AD and Azure Connect. You will also be able to design a high available infrastructure for identity replication.

The following figure describes the general structure of Azure AD in a hybrid deployment with Active Directory Domain Services.

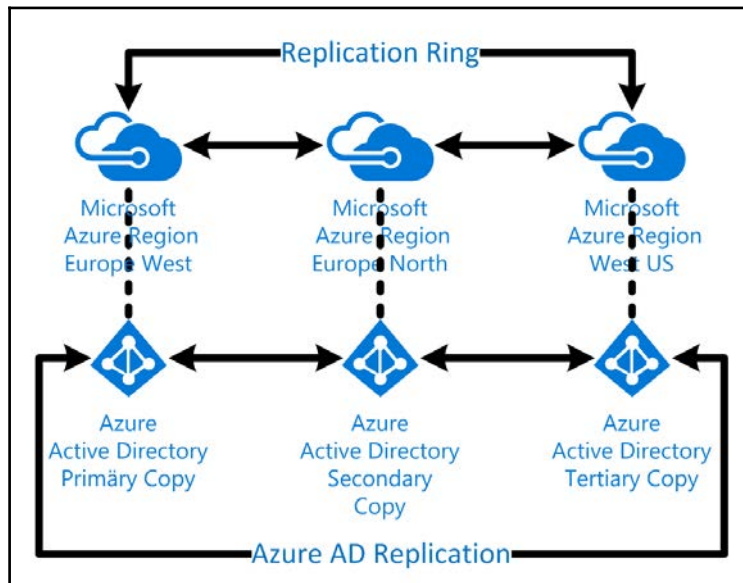


Source: <https://docs.microsoft.com/en-us/azure/active-directory/active-directory-hybrid-identity-design-considerations-overview>

Customers using different Microsoft services like Office 365, CRM online, or Intune are already using Azure AD for their service. You can easily identify if you use Azure AD if you have a username like `user@domain.onmicrosoft.com`. Other top-level-domains like `.de` or `.cn` are also possible if you are using Microsoft Cloud Germany or Azure China.

Azure AD is a multi-tenant, Geo-distributed, high availability service running in every Microsoft data center around the world. Microsoft has implemented automated failover with a minimum of two copies of your Azure Directory Service in other regional or global data centers.

Your directory is running in your primary data center, is regularly replicated into another two in your region. If you only have two Azure data centers in your region, as in Europe, the copy will distribute to another data center in another region:



Azure Active Directory Options

There are currently three selectable options for Azure Active Directory with different features to use. There will be a fourth option, **Azure Active Directory premium P2** available since late 2016.

Azure AD Free

Azure AD free supports common features such as:

- Up to 5,00,000 directory objects which could be user, devices, applications, or groups
- User/group management (add/update/delete), user-based provisioning and device registration
- **Single Sign-On (SSO)** for up to ten applications per user
- Self-service password change for the cloud users
- Connect and sync with on-premises Active Directory Domain Services
- Up to three basic security and usage reports

Azure AD Basic

This supports all the common features from free Azure AD and more, such as:

- Same features like free and additional Group-based access management/provisioning
- Self-service password reset for the cloud users
- Company branding (logon pages / access panel customization)
- Application proxy
- Service level agreement 99.9%

Azure AD Premium P1

Supports common features from free and basic Azure AD such as:

- Group-based access management/provisioning
- Self-service password reset for the cloud users
- Company branding (logon pages / access panel customization)
- Application proxy
- Service level agreement 99.9%

Premium features such as:

- Self-service group and app management / self-service application additions / dynamic groups
- Self-service password reset/change/unlock with on-premises write-back
- Multi-factor authentication (cloud and on-premises (MFA server))
- MIM CAL plus MIM server
- Cloud app discovery
- Connect health
- Automatic password rollover for group accounts

Since Q3/2016 Microsoft has also enabled customers to use the Azure Active Directory P2 plan, which includes all the capabilities in Azure AD premium P1 as well as the new **Identity Protection** and **Privileged Identity Management** capabilities. This was an important step for Microsoft to extend its offering for Windows 10 Device Management with Azure AD.

Currently Azure AD enable Windows 10 customers to join a device to Azure AD, implement SSO for desktops, Microsoft passport for Azure AD, and central administrator BitLocker recovery.

Depending on what you plan to do with your Azure environment, you should choose your Azure Active Directory option.

Deploying a Custom Azure AD

To understand how you deploy an Azure Active Directory you need to understand that Azure Active Directory is directly connected to your Azure subscription. So, the Azure account subscription administration is always the first service administrator for your Azure environment. There can only be one account administrator per Azure subscription. The account administrator is the only one who can manage Azure AD and subscription connections. If you lose your administrator credentials or lose access to the administrator account, you can no longer manage your subscription.

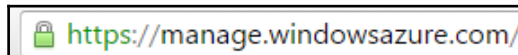
You should therefore plan who will create your subscription and which account is the account administrator. To create a subscription, the subscription administrator must have a Microsoft Accounts or former named Live ID or Microsoft Account, Azure Active Directory account. This can be created, for example, via Office 365 before adding Azure agreements and subscription payments or could be an account created and synchronized via the Active Directory Domain Service to Azure. What you shouldn't do is to use a personal account of an employee to function as an account administrator and Global Azure Administrator. If there is any change with that employee, you could lose Azure AD access. In any case, you should work with group accounts or service accounts, so that a minimum of two people are able to access the subscription.

Normally an Azure Active Directory is created when you create an Azure subscription or you subscribe to a Microsoft Cloud service like Office 365.

As an Azure account administrator you can create a new Azure AD and change your Azure subscription to the new Azure AD.

Let's begin by creating an Azure AD:

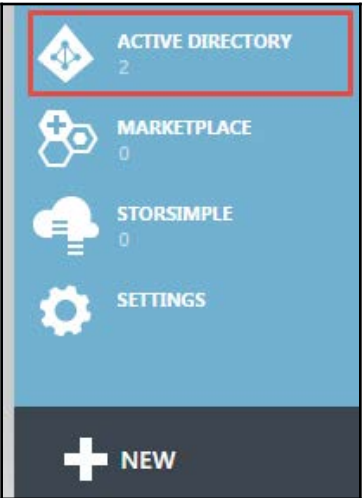
1. First you need to log in to <https://manage.windowsazure.com/>:



Currently, Microsoft offers two portals:

<https://manage.windowsazure.com> which represents the old deployment engine based on **Microsoft Service Manager**; and the modern portal <https://portal.microsoft.com>, which is based on the Microsoft resource manager engine and should be the preferred portal. As soon as all service are migrated to resource manager engine, the old portal will be disabled.

2. Navigate to Azure **ACTIVE DIRECTORY** on the lower left site of the navigation bar.



3. When you click on it, you see the Azure AD which is already deployed. With a fresh subscription, you normally see the **Default Directory**. What you also see is the **DATACENTER REGION** where your directory is deployed, and which region your subscription is in:

active directory

DIRECTORY ACCESS CONTROL NAMESPACES MULTI-FACTOR AUTH PROVIDERS RIGHTS MANAGEMENT

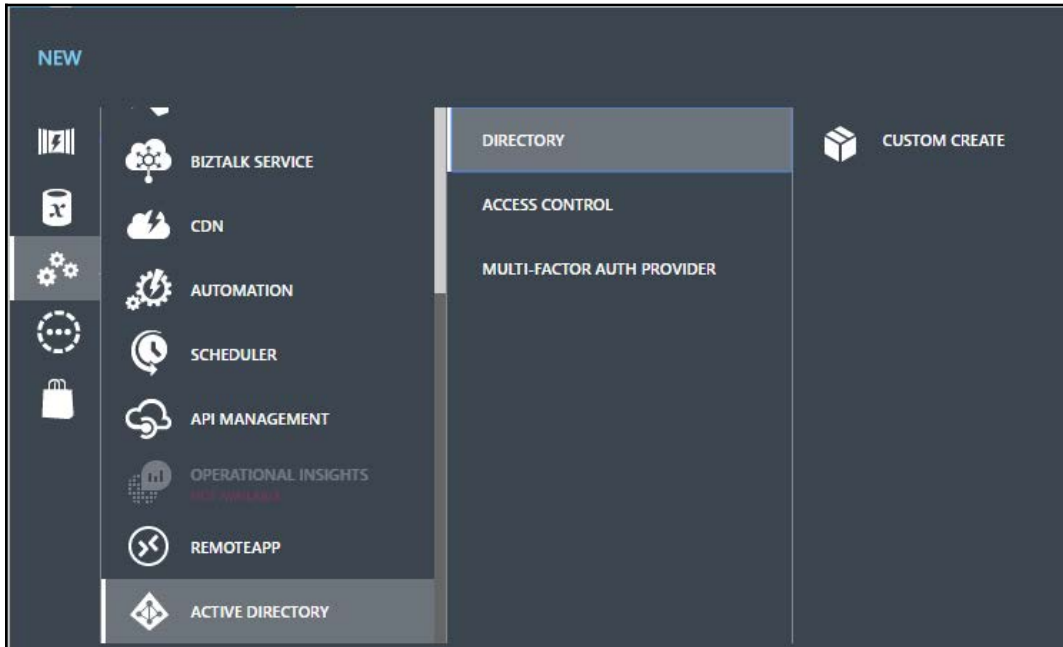
NAME	STATUS	ROLE	SUBSCRIPTION	DATACENTER REGION	COUNTRY OR RE...	
Standardverzeichnis →	✓ Active	Global Administrator	Shared by all Standardverz...	Europe	Germany	

active directory

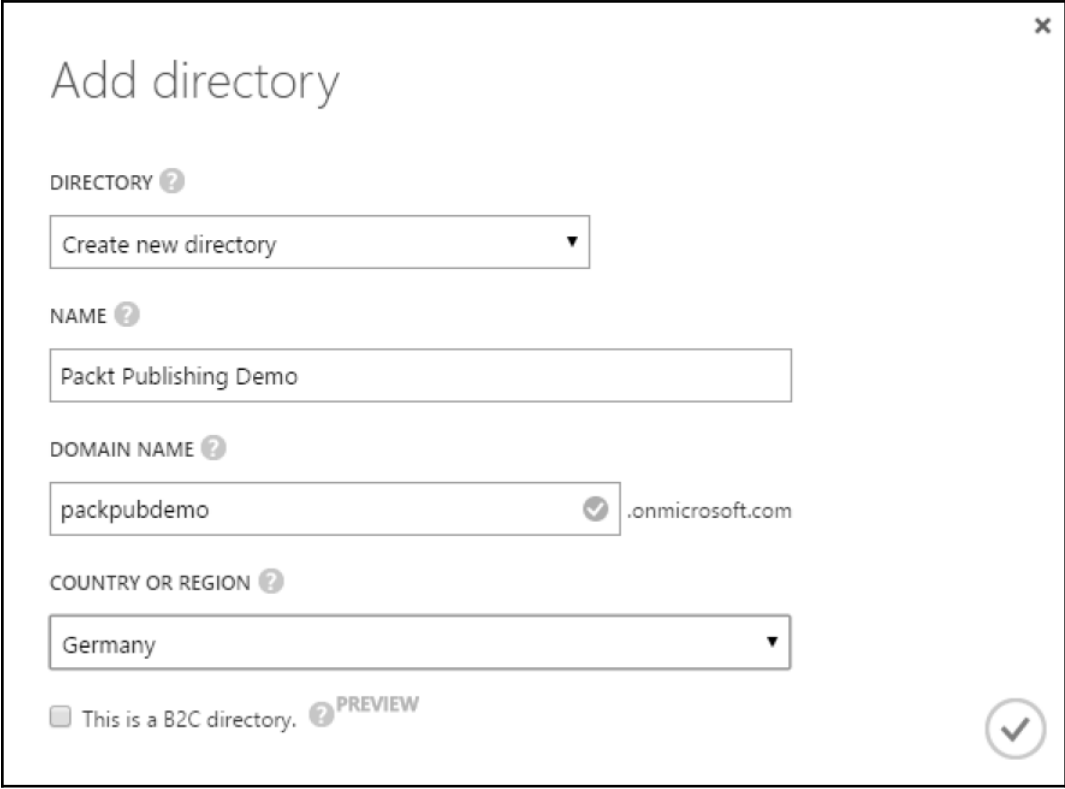
DIRECTORY ACCESS CONTROL NAMESPACES MULTI-FACTOR AUTH PROVIDERS RIGHTS MANAGEMENT

NAME	↑	STATUS	ROLE	SUBSCRIPTION	DATACENTER REGION	COUNTRY OR RE...	
	→	✓ Active	Global Administrator	Shared by all DataCenter-Flo sub...	Europe, United States	Germany	
Default Directory →		✓ Active	Global Administrator	Shared by all Default Directory su...	Europe, United States	Germany	

4. To create a new directory, you need to click on **NEW** in the lower left corner and navigate as shown in the following screenshot:



5. Click on **CUSTOM CREATE** to start the wizard that will show guide you through the installation. Within the wizard you will define the **DIRECTORY**, **NAME** and **COUNTRY OR REGION** for your Active Directory. Every Azure AD name is unique, so you cannot create one which is already in use:



Add directory

DIRECTORY ?

Create new directory ▼

NAME ?

Packt Publishing Demo

DOMAIN NAME ?

packpubdemo ✓ .onmicrosoft.com

COUNTRY OR REGION ?

Germany ▼

☐ This is a B2C directory. ? **PREVIEW**

✓

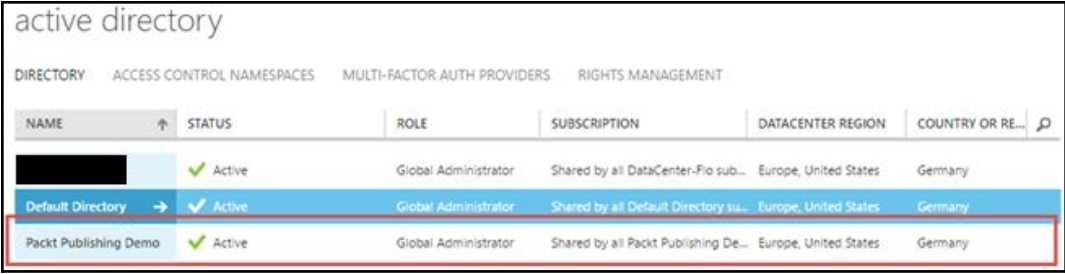


Microsoft also added the option to use a business to customer Azure Active Directory. Azure Active Directory B2C is a cloud identity management solution for your consumer-facing web and mobile applications.

You can find more information in the Azure Documentation:

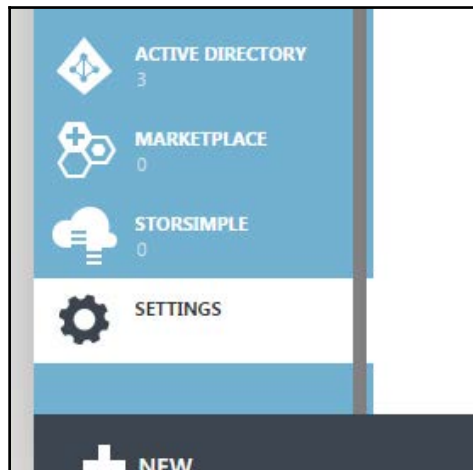
<https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-overview>.

6. After a few minutes, your new Azure AD will appear in the list:



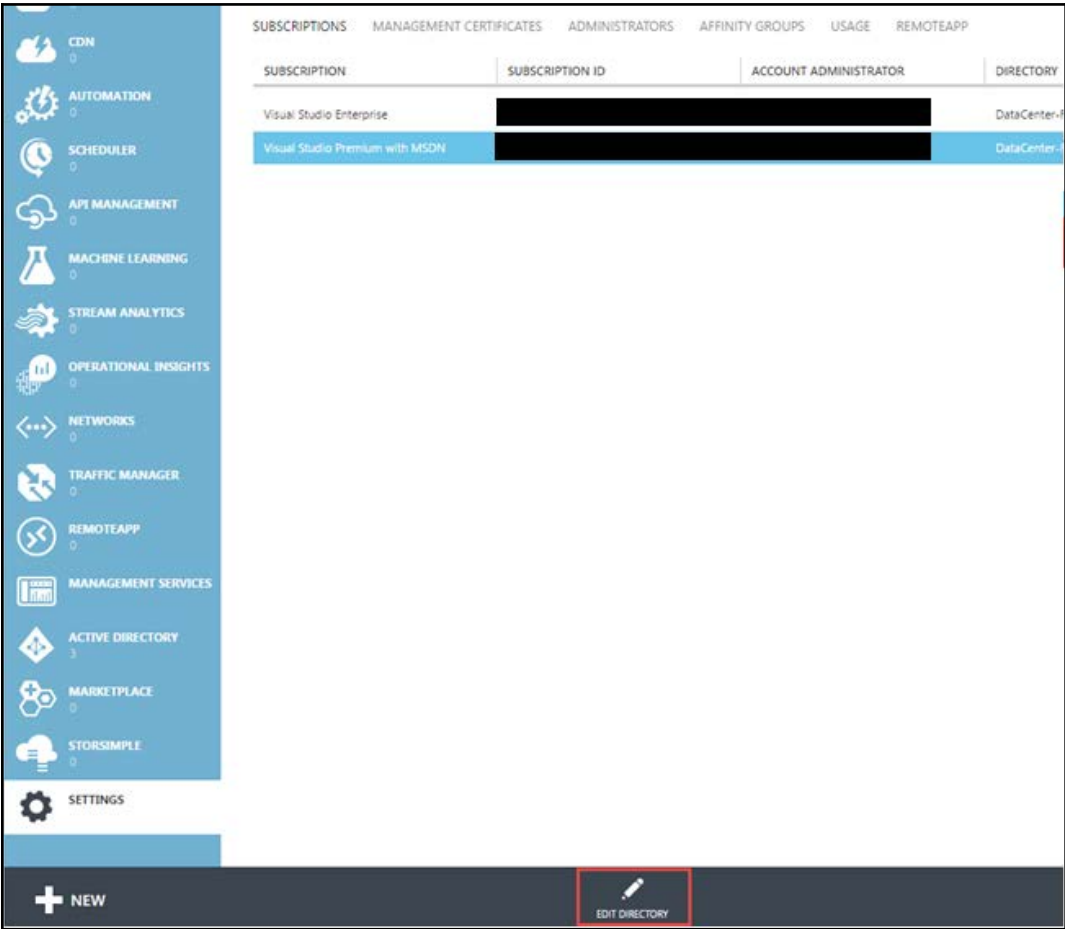
active directory						
DIRECTORY ACCESS CONTROL NAMESPACES MULTI-FACTOR AUTH PROVIDERS RIGHTS MANAGEMENT						
NAME		STATUS	ROLE	SUBSCRIPTION	DATACENTER REGION	COUNTRY OR RE...
		Active	Global Administrator	Shared by all DataCenter-Fio sub...	Europe, United States	Germany
Default Directory	→	Active	Global Administrator	Shared by all Default Directory su...	Europe, United States	Germany
Packt Publishing Demo		Active	Global Administrator	Shared by all Packt Publishing De...	Europe, United States	Germany

7. Now you can change your subscription to the new Azure Active Directory and you can add co-administrators, who can also connect to Azure and manage your resources. You need to navigate to **SETTINGS** on the left lower site in the menu:

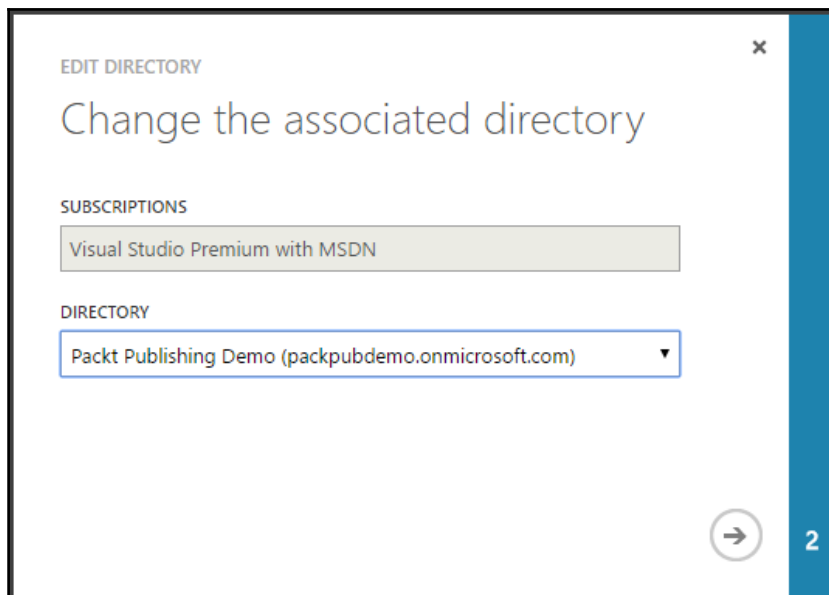


In this menu, you can select the **SUBSCRIPTIONS** which should be changed to the new Azure AD.

8. In the lower half of the page you see the **EDIT DIRECTORY** button:

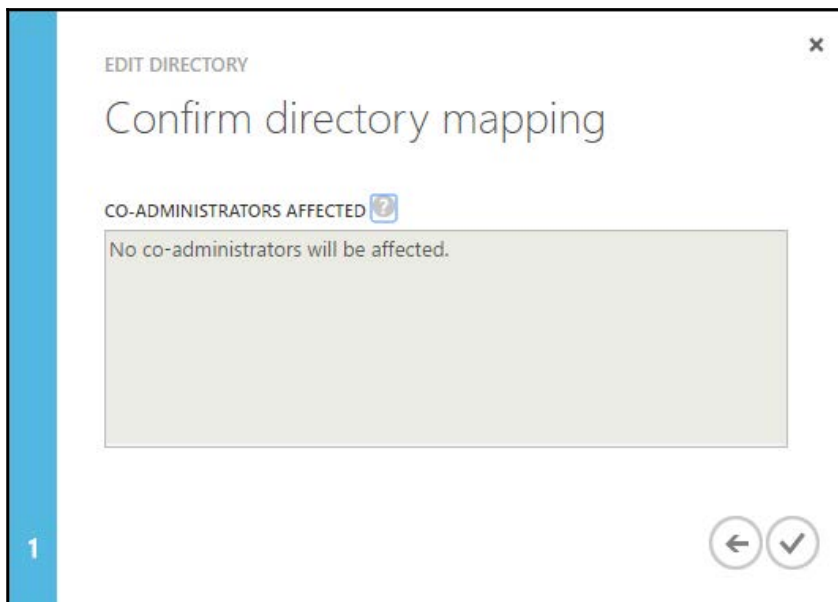


9. Within the upcoming wizard, you can change the **DIRECTORY**:



The screenshot shows a dialog box titled "EDIT DIRECTORY" with a close button (X) in the top right corner. The main heading is "Change the associated directory". Below this, there are two sections: "SUBSCRIPTIONS" and "DIRECTORY". The "SUBSCRIPTIONS" section contains a single entry: "Visual Studio Premium with MSDN". The "DIRECTORY" section contains a dropdown menu with the selected option: "Packt Publishing Demo (packpubdemo.onmicrosoft.com)". At the bottom right, there is a circular button with a right-pointing arrow and a blue vertical bar on the right side containing the number "2".

10. And confirm the directory mapping:

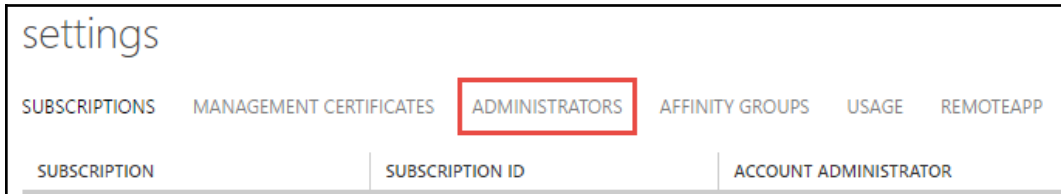


The screenshot shows a dialog box titled "EDIT DIRECTORY" with a close button (X) in the top right corner. The main heading is "Confirm directory mapping". Below this, there is a section titled "CO-ADMINISTRATORS AFFECTED" with a help icon (question mark). The text below this section reads: "No co-administrators will be affected." At the bottom right, there are two circular buttons: one with a left-pointing arrow and one with a checkmark. A blue vertical bar on the left side contains the number "1".

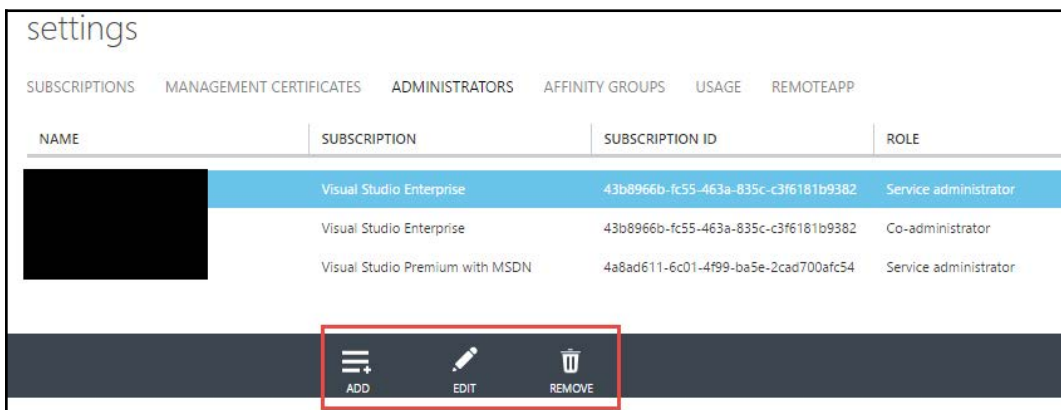


After you agreed that the directory is mapped to the new subscription all Co-Admins will be removed and you can use the subscription within the new Azure AD.

11. If you wish or need to add additional administrators on your subscription, you need to navigate to **ADMINISTRATORS**:



12. There you can **ADD**, **EDIT** or **REMOVE** additional administrators to within the subscription:



13. You can add the following account types as co-administrators:

- Microsoft Accounts or Live IDs
- Azure Active Directory accounts within that subscription

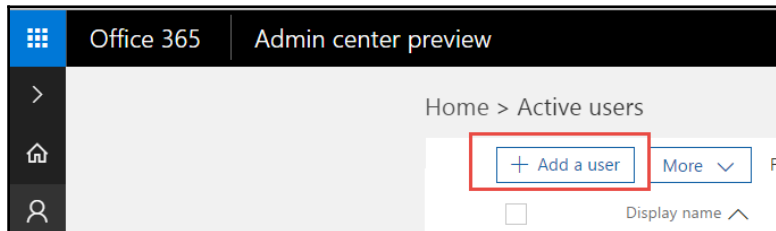
Adding Accounts and Groups to Azure AD

Firstly, you need to understand what accounts can be added to Azure AD. Basically there are two types of accounts:

- **Cloud accounts:** Accounts that are created via Azure Active Directory or other Microsoft Cloud services, like Office 365.
- **Hybrid accounts:** Accounts that are created and located in on-premises Microsoft Active Directory Domain Services. Those accounts are deployed via the Azure Active Directory connect and synchronization tool.

To create cloud accounts, you have several options. Most Azure AD users start with **Office 365** and do not natively add users via Azure. If you've used **Office 365** before, that would be the simplest for you.

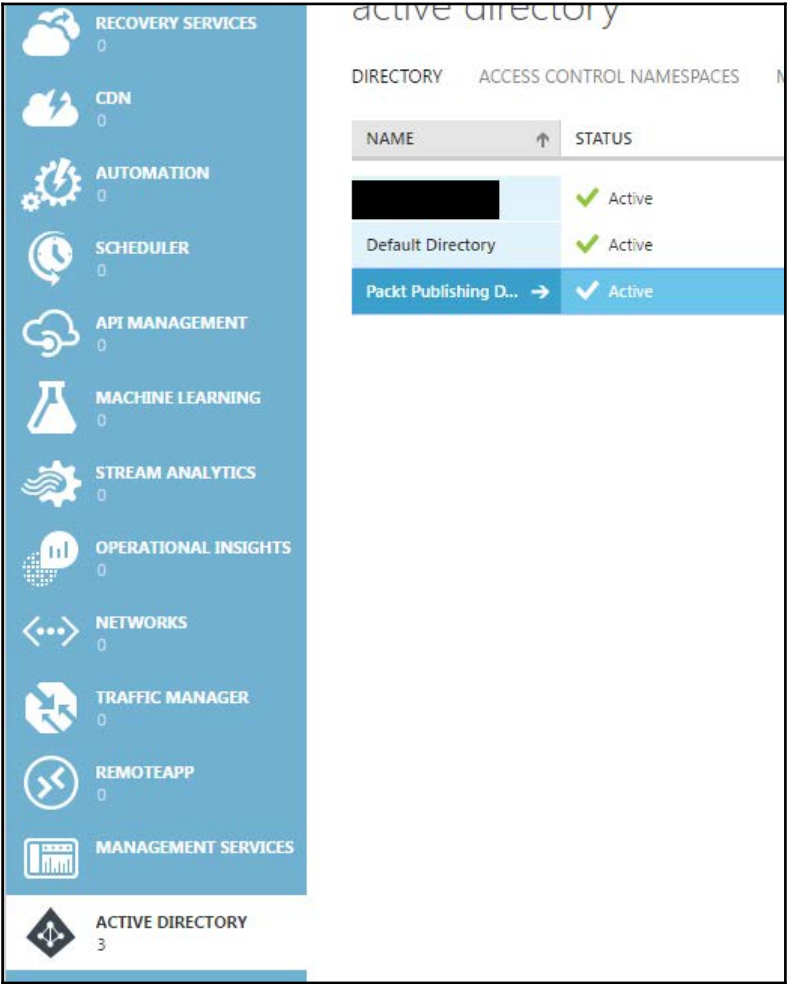
The example shown in the following screenshot guides you how to add a user from the **Office 365** preview portal via <https://portal.office.com>:



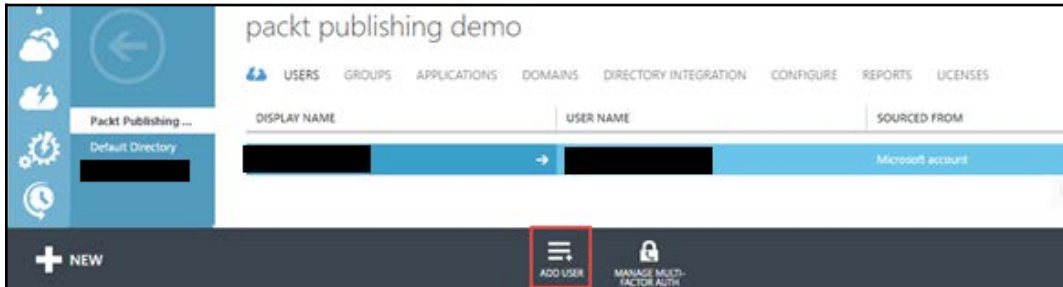
Alternatively, to create new users in Azure AD via the Azure Portal you need to follow these steps:

1. Visit the <https://manage.windowsazure.com/> Azure Portal.

2. Scroll down to Azure Active Directory and select the AD where you want to create the user:

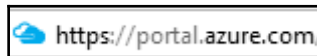


3. In the menus for **USERS** and **GROUPS** you can add new users or groups to your Azure AD:

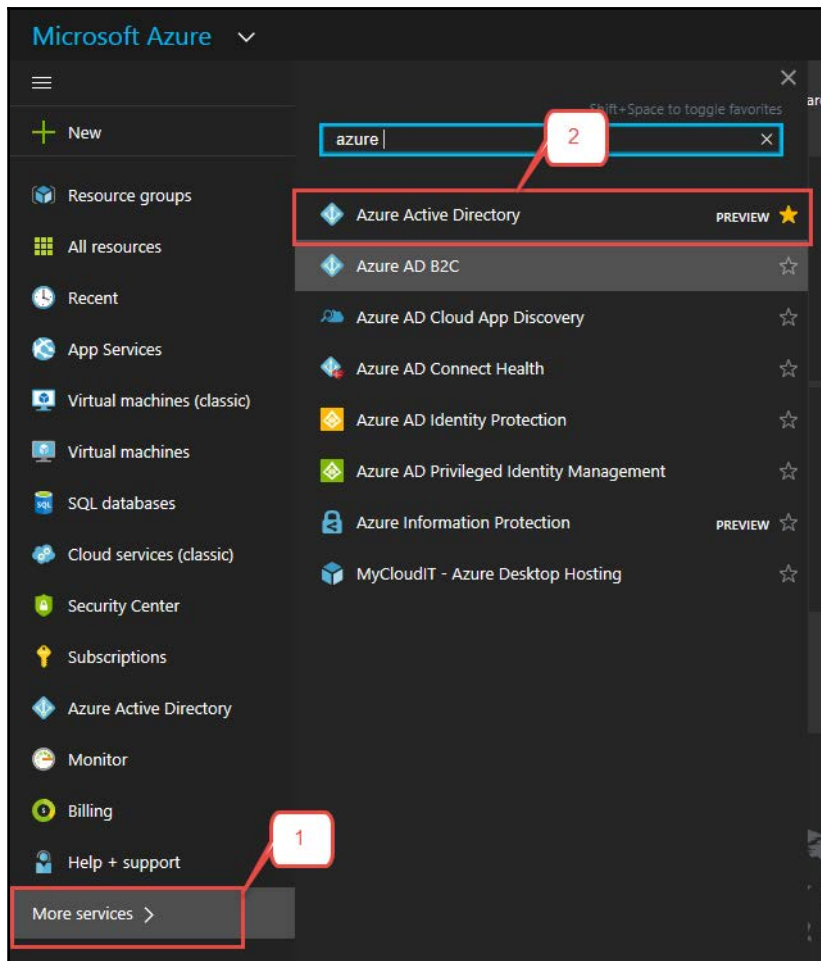


Users can have different roles and access to applications or parts of Azure via **Microsoft Azure Resource Manager Security** and **Access Roles**. You will get more detail on the security features within the following chapters.

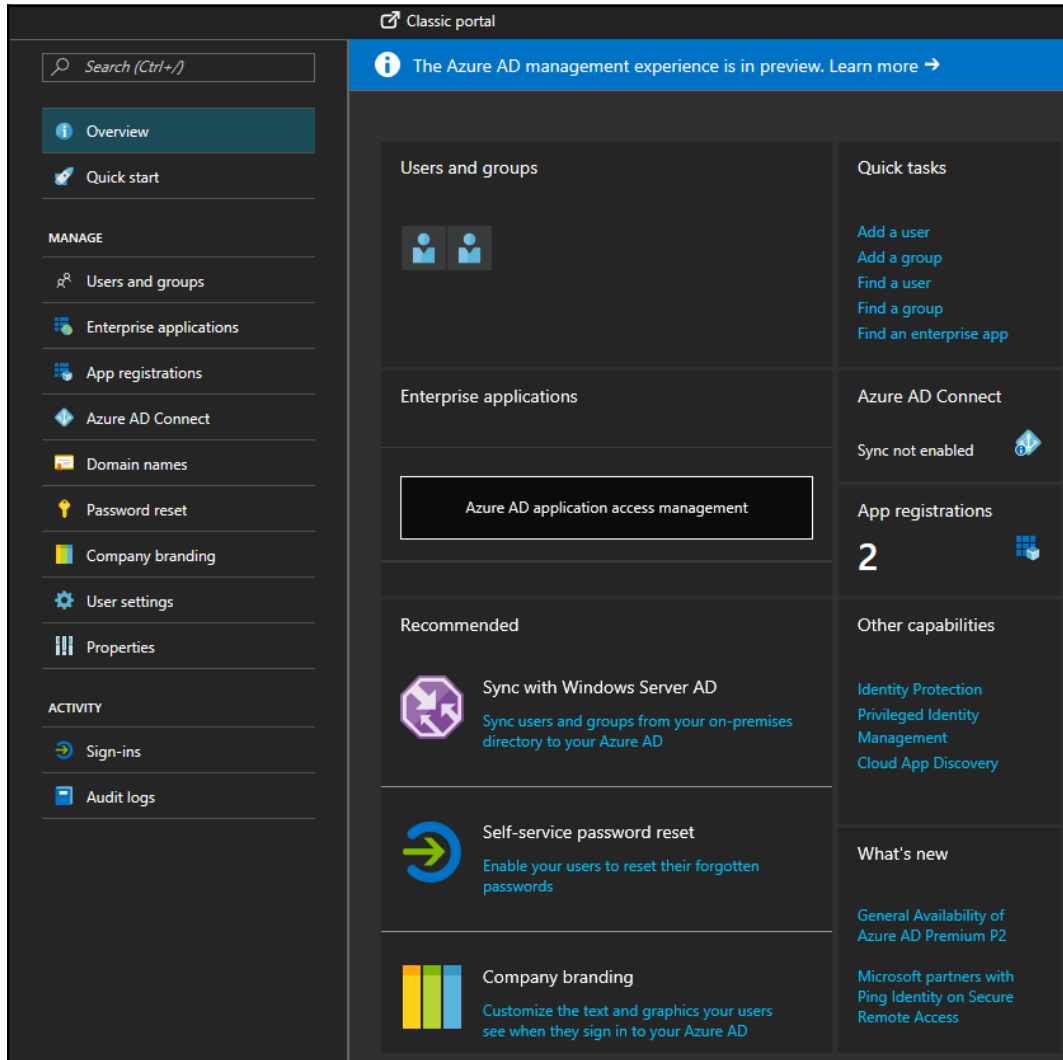
4. By August 2016 Microsoft started to migrate all Azure AD Service into Azure resource manager engine. So, you will need to become familiar with the new interface. Browse to <https://portal.azure.com/>:



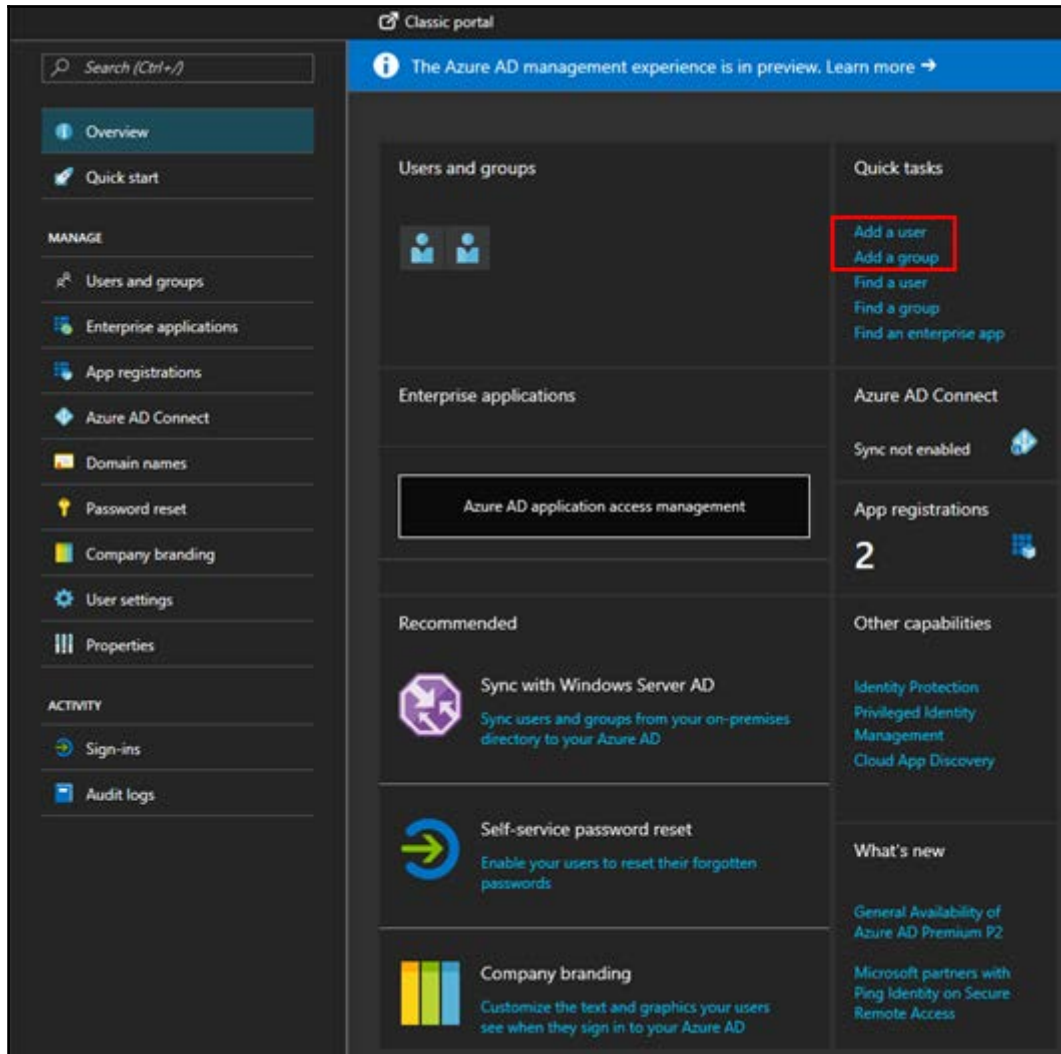
5. Click on **More services** on the sidebar and look



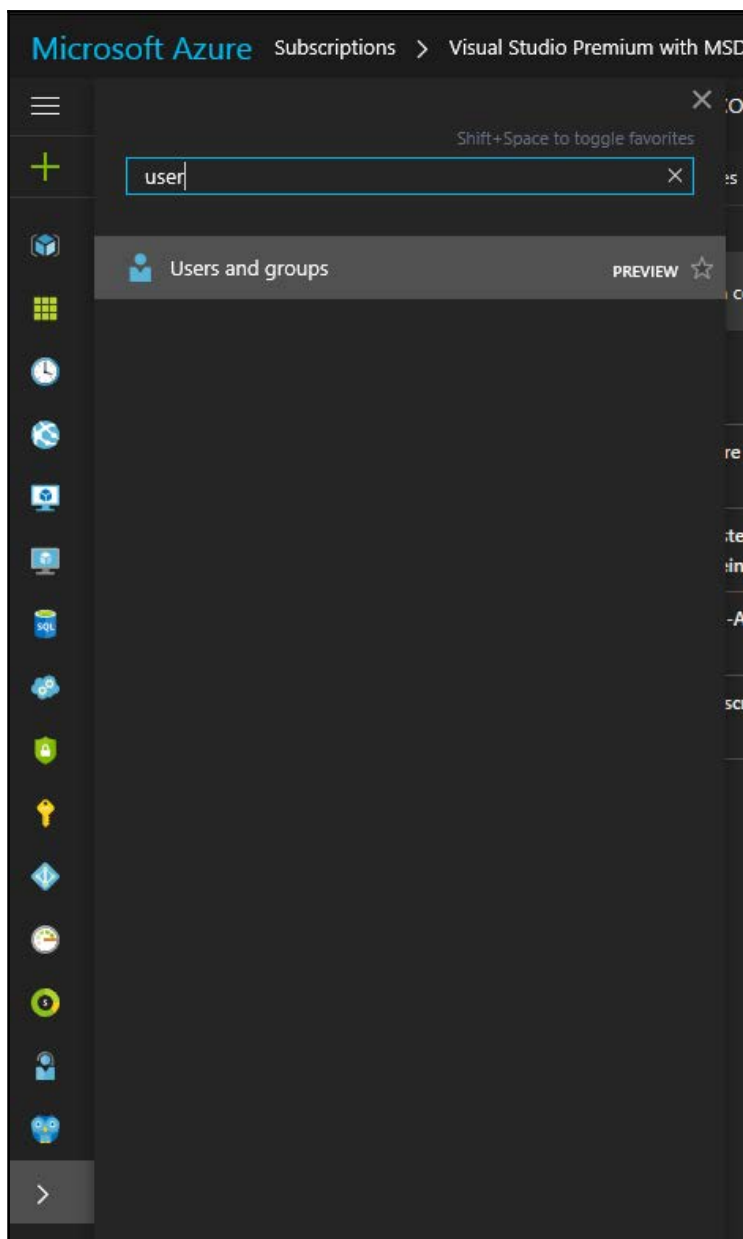
6. The new **Azure Active Directory** interface does not yet have all the features enabled. Currently it is not possible to create new Azure Active directories but you can perform most of the user and application operations:



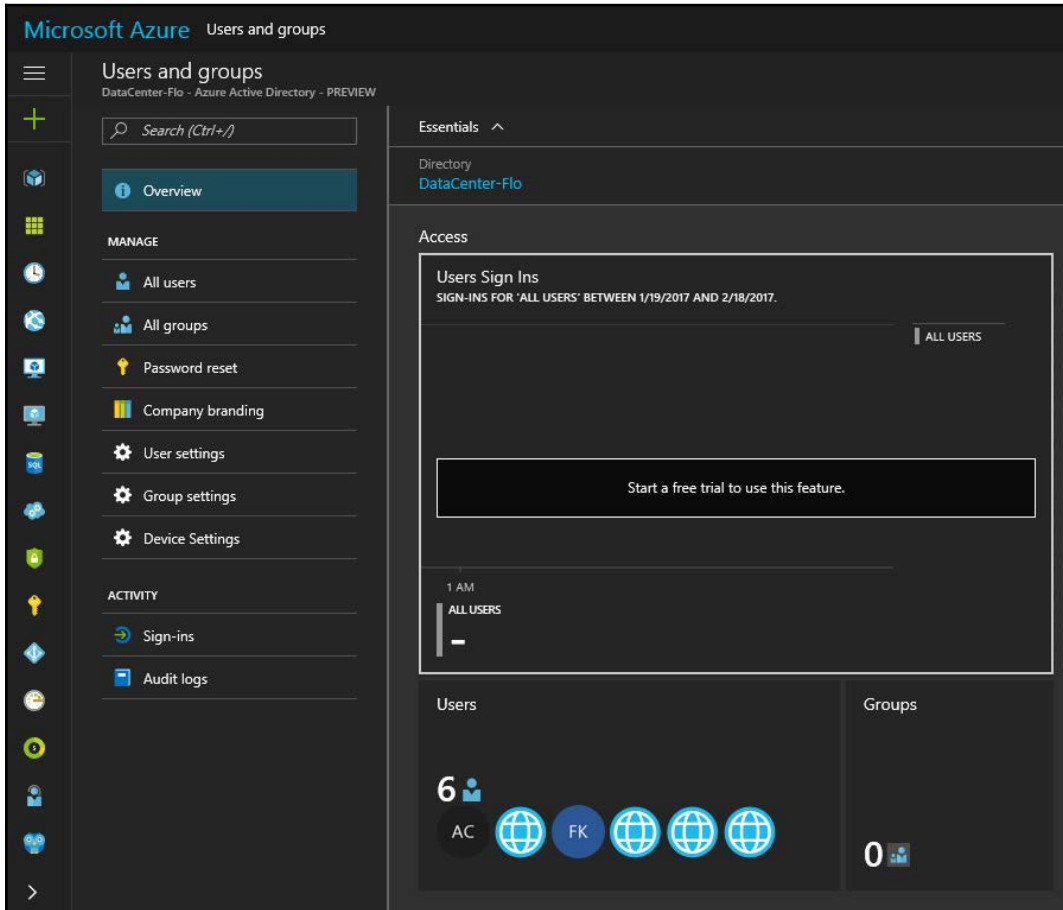
7. In addition to the new portal Microsoft also extended the user management to the **Azure AD management** portal. While writing the book the new user and group management is still in preview, so changes are still possible. To add or change user accounts, you now have different opportunities. The first one would be to open user and group management via Azure Active Directory interface:



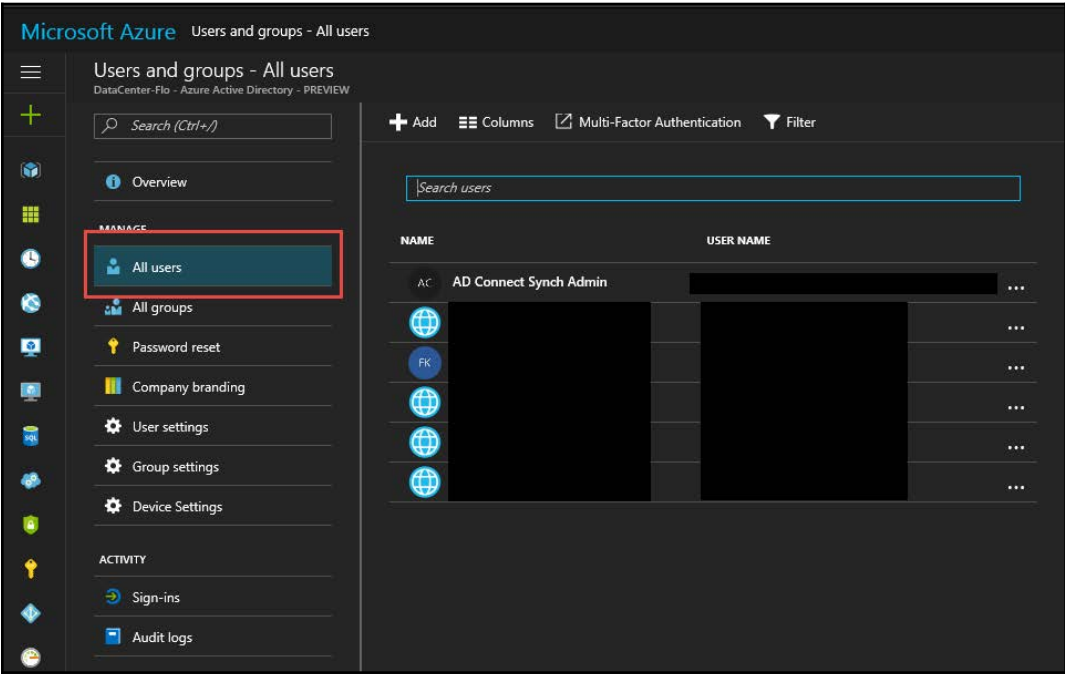
8. Or look for **Users and groups** with the Microsoft resources:



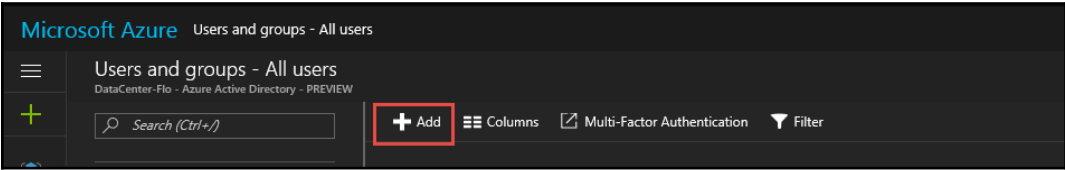
9. Both ways will bring you to the same blade with options to create **Users** and **groups**:



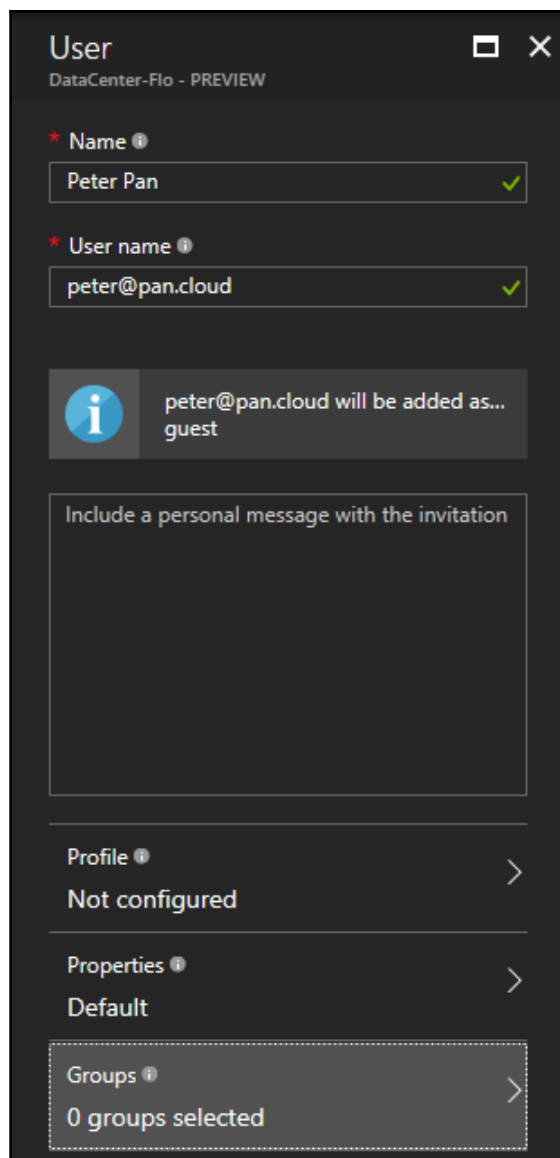
10. To add a user within the new UI, you click on **All users** as shown in the following screenshot:



11. There you click on **Add** and follow the instructions shown in the following screenshot:



12. The blade will ask you to provide a username, as shown here:



The screenshot shows the 'User' blade in the Azure portal. The title bar indicates 'DataCenter-Flo - PREVIEW'. The form contains the following fields and sections:

- Name:** A text box containing 'Peter Pan' with a green checkmark to its right.
- User name:** A text box containing 'peter@pan.cloud' with a green checkmark to its right.
- Information:** A section with a blue information icon and the text 'peter@pan.cloud will be added as... guest'.
- Message:** A large text area with the placeholder text 'Include a personal message with the invitation'.
- Profile:** A section with a right-pointing arrow and the text 'Not configured'.
- Properties:** A section with a right-pointing arrow and the text 'Default'.
- Groups:** A section with a right-pointing arrow and the text '0 groups selected'.



Please be aware that with Azure AD free without Office 365 the username must be an e-mail or a Microsoft Account (former Microsoft Live account) to be able to receive the invitation for Azure AD.

13. While creating the user, you have different options for pre-staging information about the user, including **First name**, **Last name**, **Job title** or **Department**:

Profile
User - PREVIEW

General

First name

Last name

Work info

Job title

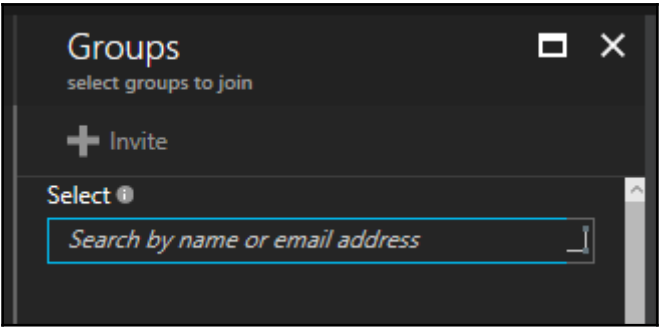
Department

14. With a synced Active Directory Domain Service and other joined services, you can change the **Source of Authority**:

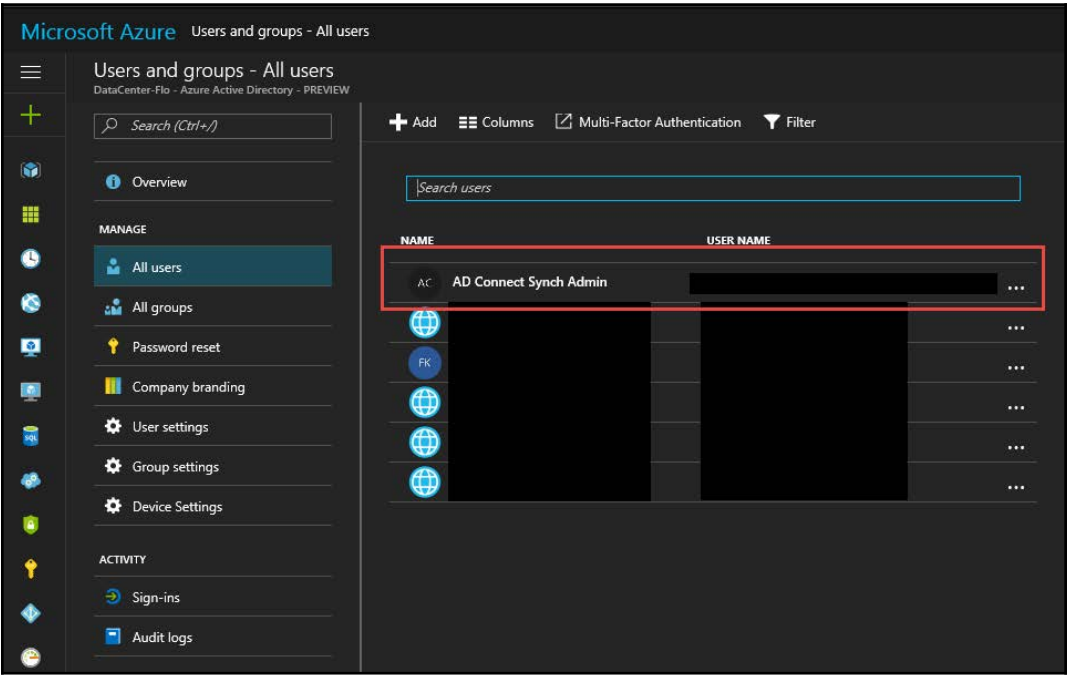
Properties
User - PREVIEW

Source of Authority

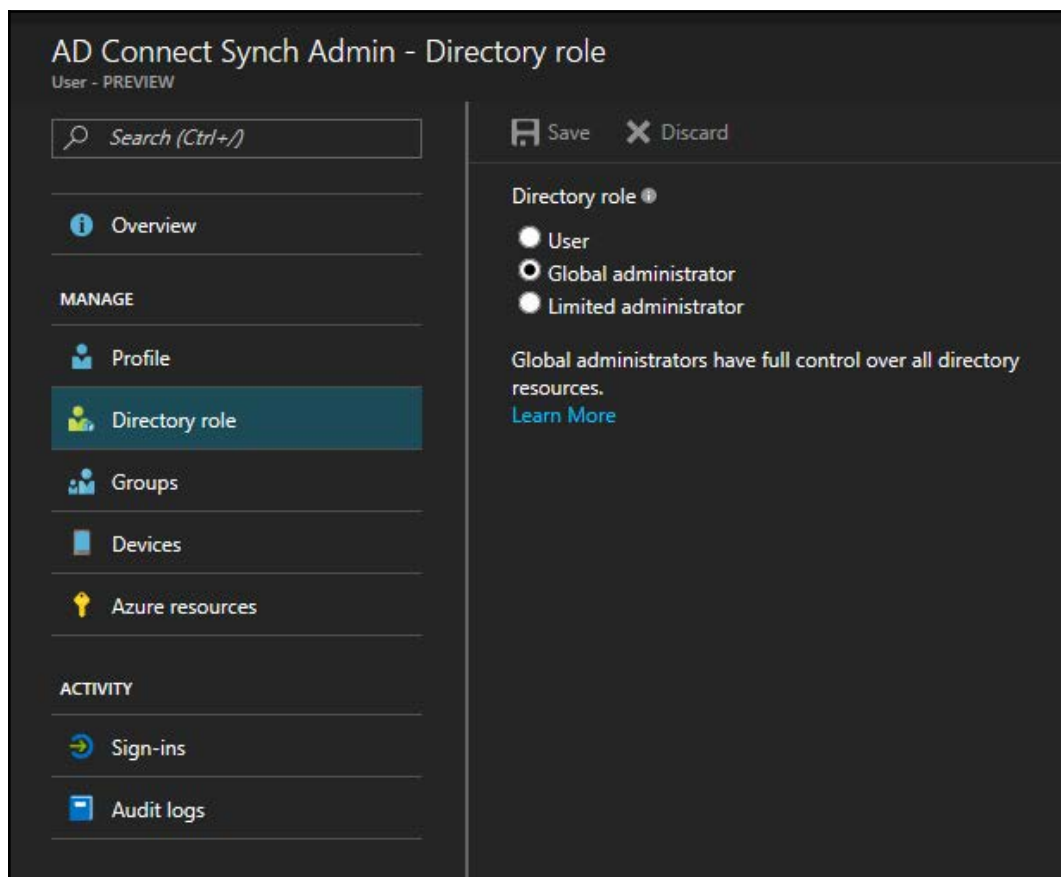
15. You can also join the account directly to Azure AD **Groups** during creation:



16. The new UI for Azure AD included a new option to join users as account admins (formerly know as co-administrators). To do this, change the admin rights of the user by clicking on the relevant user to open the user blade:



17. Then, click on **Directory role** to open the role options:



18. Not every user needs to be **Global administrator** to be able to fulfill his job. Mostly, one of or more of the options of a **Limited administrator** should be enough.

Save Discard

Directory role ⓘ

☐ User

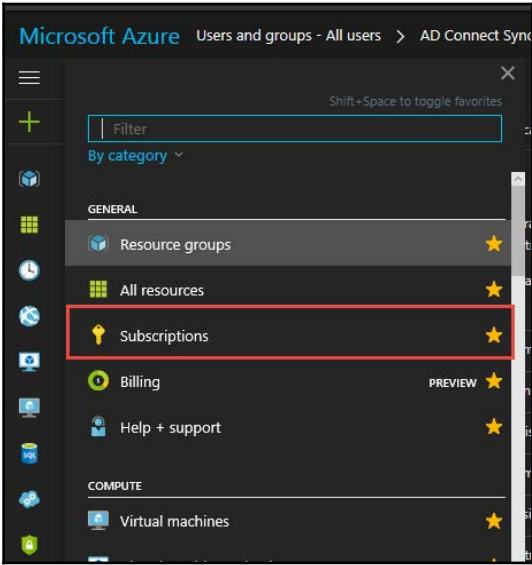
☐ Global administrator

☐ Limited administrator

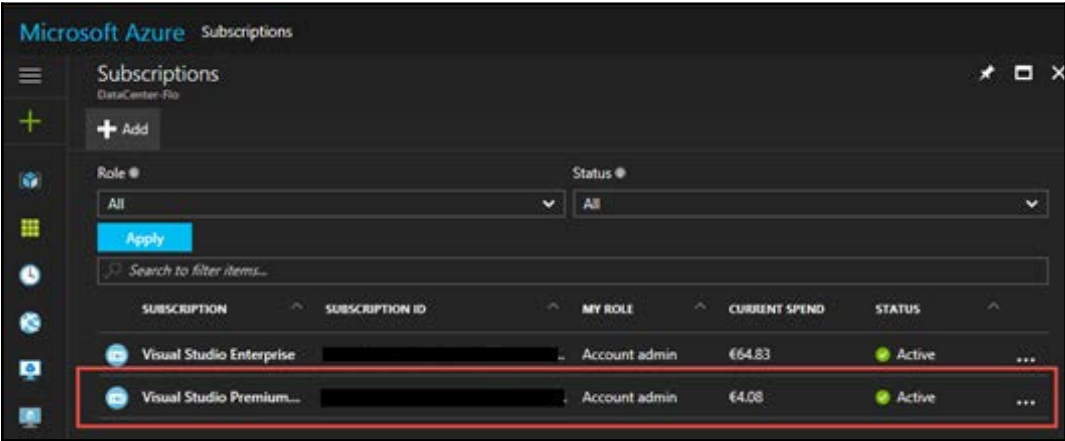
Select the administrative role or roles for this user.
[Learn More](#)

- Password administrator ⓘ
- Service administrator ⓘ
- Billing administrator ⓘ
- Exchange administrator ⓘ
- Skype for Business administrator ⓘ
- User administrator ⓘ
- SharePoint administrator ⓘ
- Compliance administrator ⓘ
- Security reader ⓘ
- Security administrator ⓘ
- Privileged role administrator ⓘ
- Intune Service administrator ⓘ
- Guest inviter ⓘ

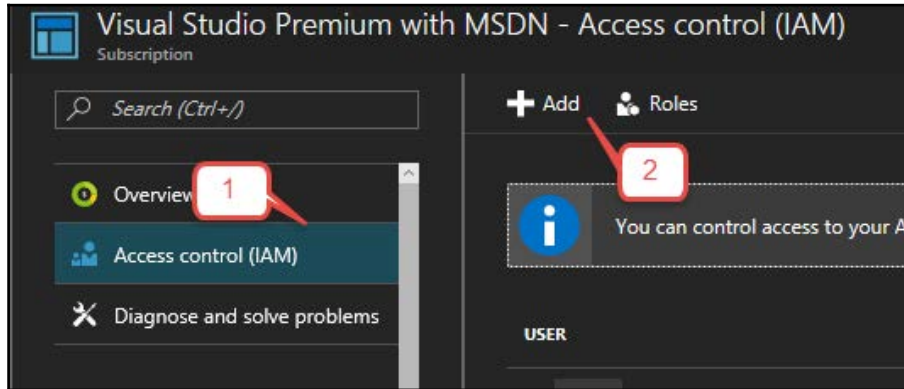
19. After you add the user to his admin role, you need to go to **Subscriptions** within the Azure resources:



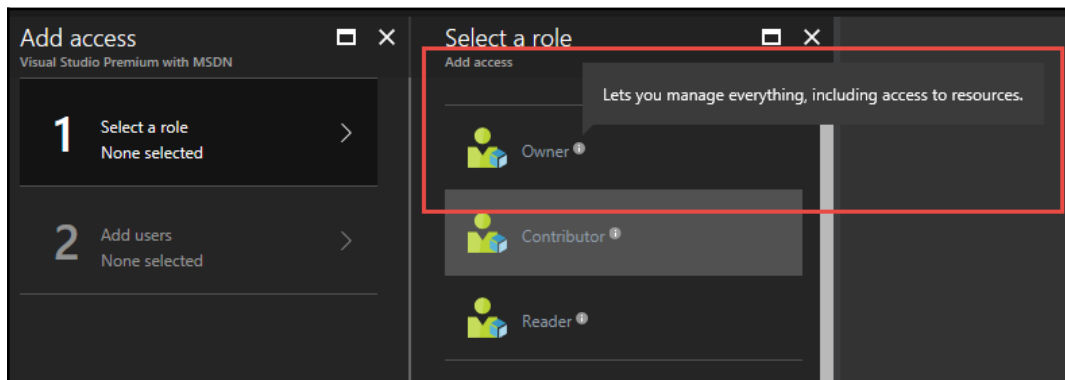
20. There you select the **Subscriptions** for which the user should be the co-administrator:



21. In the following blade, you click on **Access control (IAM)** and then on **Add**:



22. Now **Select a role** for the user. To make him a co-administrator, we need to give him **Owner** rights:



TIP

As an **Owner**, he has full access to the subscription resources, which would enable him to do all operations that need to be done on a subscription. As a **Reader**, he can see billing and resources of a subscription but not change things inside the subscription. That is the option most billing tools like **Cloud Cruiser** or **Azure Costs** need to create their statistics.

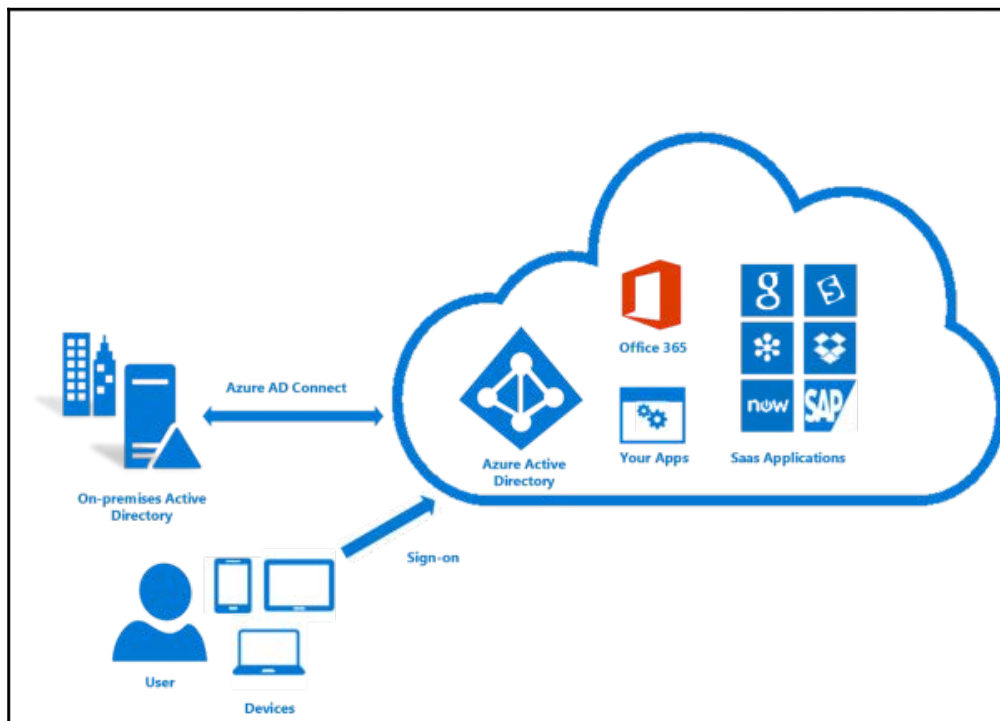
23. Next, select the user or group which should have the permissions. This user can now manage the subscription:



You can also invite Microsoft Accounts to your subscription without creating a user first. When it comes to best practice or how it would be done in the field, you wouldn't add any single accounts into the subscription. You should give permission on the subscription or resource to a group, and then add users to the group.

The other option to create new users and groups, is to sync them via Active Directory Domain Service from your connected on-premises Active Directory Domain Service. To do this, you'll need an additional tool named Azure Active Directory Connect, Azure AD Connect or short AAD Connect.

Azure AD Connect will integrate your on-premises directories with **Azure Active Directory**. This allows you to provide a common identity for your users for Office 365, Azure, and SaaS applications integrated with Azure AD:



Source: <https://azure.microsoft.com/en-us/documentation/articles/active-directory-aadconnect/>

Azure AD Connect is the central tool to implement hybrid identities in Azure which enables you to license your software or implement identity and access management like Single Sign-On, Multi-factor authentication or the AD Rights Management Service.

Azure AD Connect brings five programs with different purposes for Azure AD Connect:

- **Azure AD Connect:** Azure AD Connect is the configuration tool with an integrated and detailed wizard to configure Azure AD and on-premises Active Directory Domains Services synchronization.
- **Synchronization Rule Editor:** Is a basic tool to configure and customize synchronizations between Azure AD and on-premises Active Directory Domains Services:

Synchronization Rules Editor

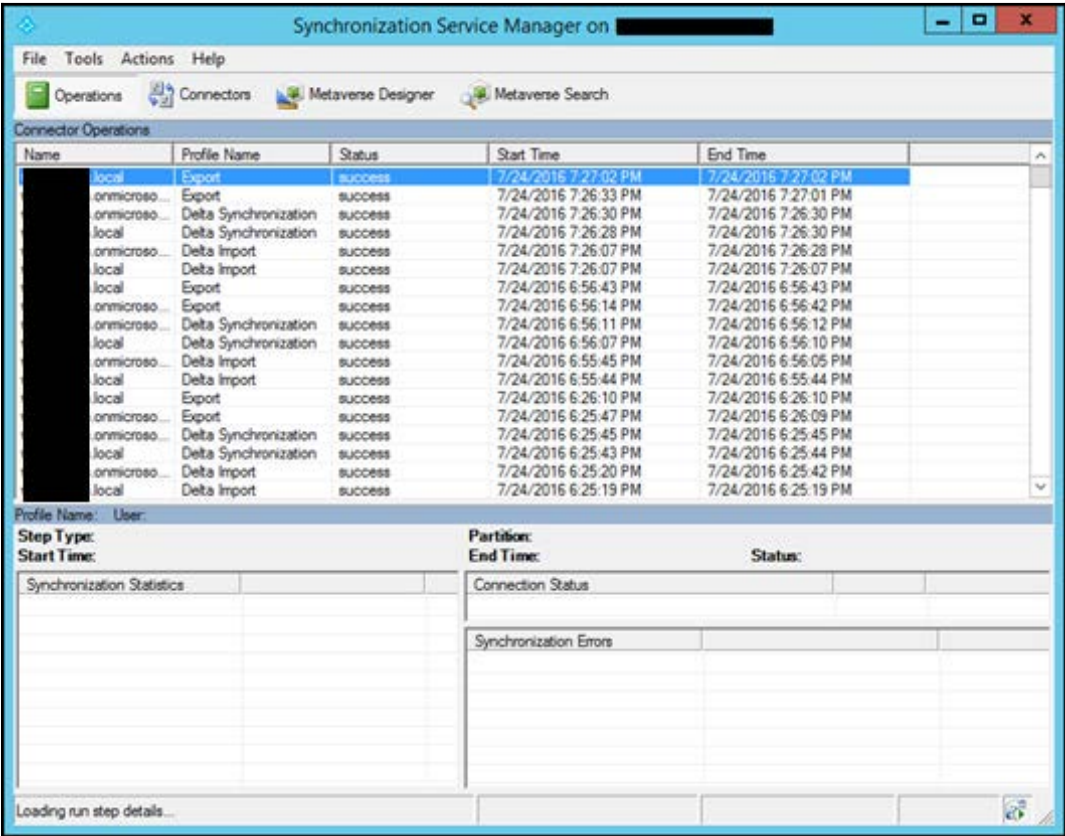
View and manage your synchronization rules

Direction: Inbound MV Object Type: Connector: Connector Object Type: Password Sync: MV attribute: Connector Attribute: Add new rule

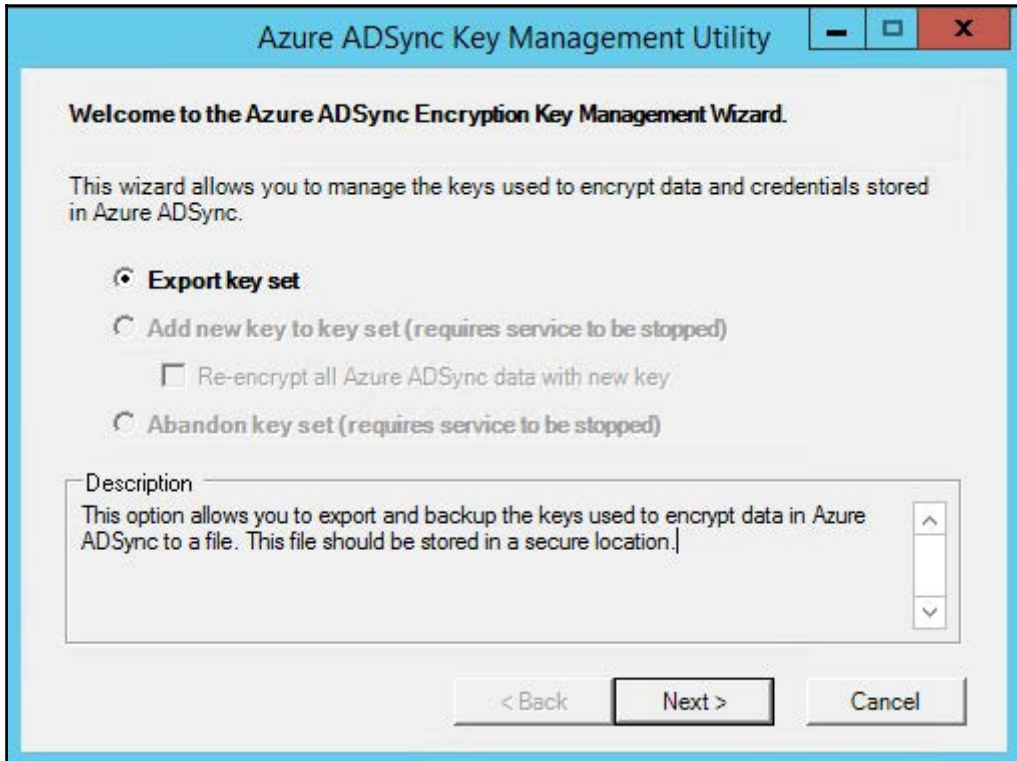
Name	Connector	Precedence	Connector Object Type	Metaverse Object Type

Type: Transformations: Disabled: Scoping filters: Join rules: Edit Export Delete

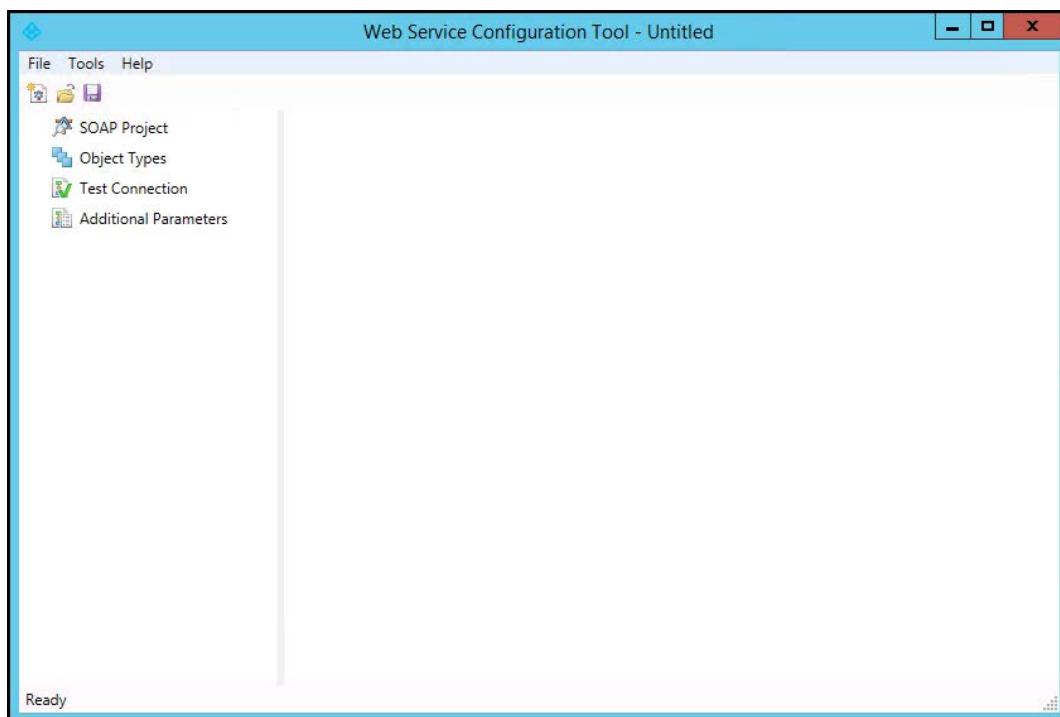
- **Synchronization Service:** The **Synchronization Service** is a tool to basically monitors and logs the synchronization between Azure AD and on-premises Active Directory Domains Services. You can supervise the synchronization process:



- **Synchronization Service Key Manager:** Helps to manage security keys to encrypt data transferred between Azure AD and on-premises Active Directory Domains Service:



- **Synchronization Service Web Service Configuration Manager:** This came with the version 1.1.189.0 from Azure Active Directory Connect in June 2016. It is used to configure **Microsoft Account Entity Management (MIM)** endpoints with Azure AD Connect:



Azure AD Connect and all components are freely available and can be downloaded from

Microsoft <https://www.microsoft.com/en-us/download/details.aspx?id=47594>.

You can deploy Azure AD Connect in three different ways for the users. Each method gives different kinds of integration levels and is more or less dependent to your Azure AD subscription level:



User sign-in

Select the Sign On method.

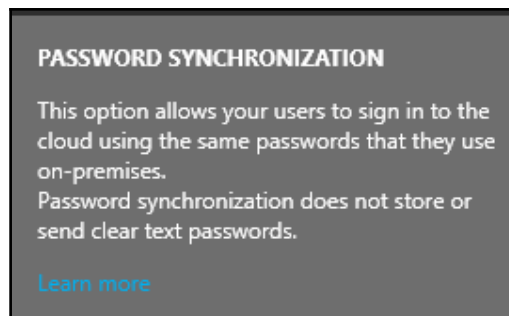
- ☒ Password Synchronization ?
- ☐ Pass-through authentication ? (Preview)
- ☐ Federation with AD FS ?
- ☐ Do not configure ?

Select this option to enable single sign on for your corporate desktop users:

- ☒ Enable single sign on ? (Preview)

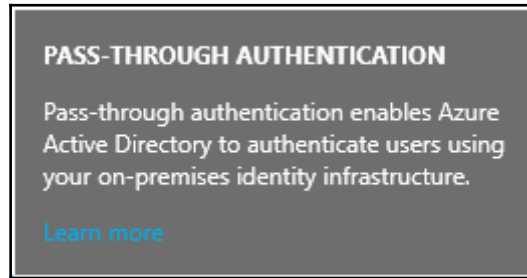
The first solution is to use **Password Synchronization**. This option transfers user passwords as hashed values to the cloud. The option is also that one which enables the basic Single Sign-On option for your users to Azure AD-based applications in your organization. The password synchronization is only based on user account and password replication. So changes within Azure AD will take some time or be manual. This option is also only practical for small environments with less than 300 users. For larger environments, you would have too much replication traffic and too many changes within Azure AD, which would take too long. As an example, a replication of around 4,000 users could take up to 12 hours before it is visible in the Azure AD:

The following screenshot shows you the tooltip you get when you hover over the question mark at **PASSWORD SYNCHRONIZATION**:



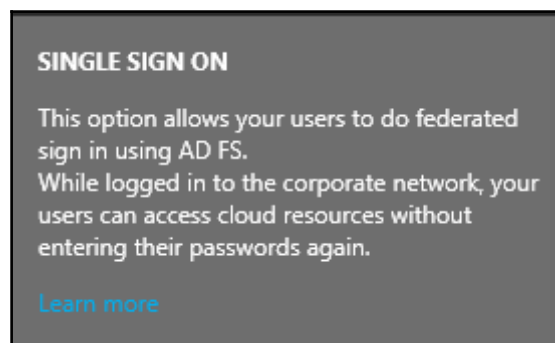
The second option is to select the new **PASS-THROUGH AUTHENTICATION**. With this option, your AD Connect and Azure AD will not store any identity data, and they will send all requests directly to your environment.

The following screenshot shows you the tooltip you get when you hover over the question mark at **PASS-THROUGH AUTHENTICATION**:



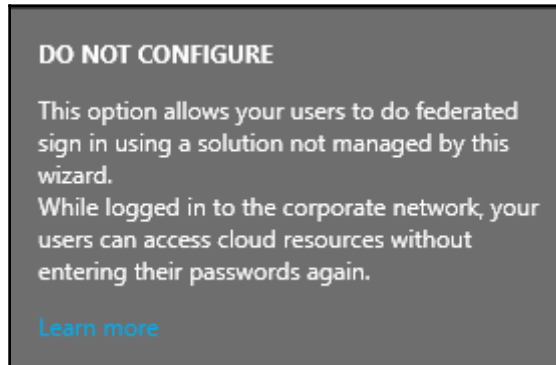
The third and most complex solution is to implement an **Active Directory Federation Services (ADFS or AD FS)** with Azure AD, in the interface named federation with AD FS. That will enable full Single Sign-On and add multifactor authentication. The organizations and implementations. If you want to implement ADFS you need also to have Public and Private Key infrastructure and certificates from a trusted agency in place. For ADFS you need good response times so you might need to upgrade your Internet access and/or wide area network connectivity.

The following screenshot shows you the tooltip you get when you hover over the question mark at **Federation with AD FS**:

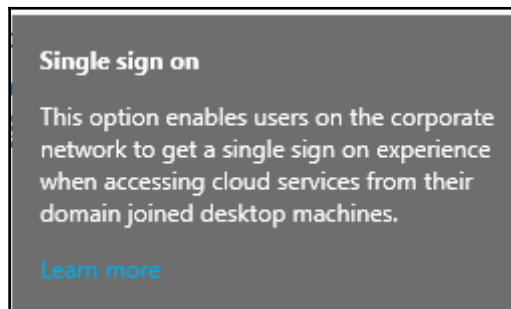


The fourth and easiest way is to not configure user sign-in. This option enables only license and user replication from local to cloud and vice versa. There is no option to replicate passwords, and your users will not be able to sign in or use Azure AD resources. Users would be able to use, for example, Office 365 or Azure remote app.

The following screenshot shows you the tooltip you get when you hover over the question mark at **DO NOT CONFIGURE**:



With the new enhancements in Windows 10 and with Azure AD, you can now also configure Single Sign-On directly from your on-premises Windows systems. To enable this feature, you need to mark **Enable single sign on**:



Installing Azure AD Connect: Prerequisites

For those who haven't worked with AD FS before, Federations offer a standard-based service that allows the secure sharing of identity information between trusted business partners (known as a **federation**) across an extranet. When a user needs to access a web application from one of its federation partners, the user's own organization is responsible for authenticating the user and providing identity information in the form of *claims* to the partner that hosts the web application. The hosting partner uses its trust policy to map the incoming claims to claims that are understood by its web application, which uses the claims to make authorization decisions. So, basically Azure AD and Microsoft become your business partner in AD FS.

From a planning perspective, the following prerequisites must be in place:

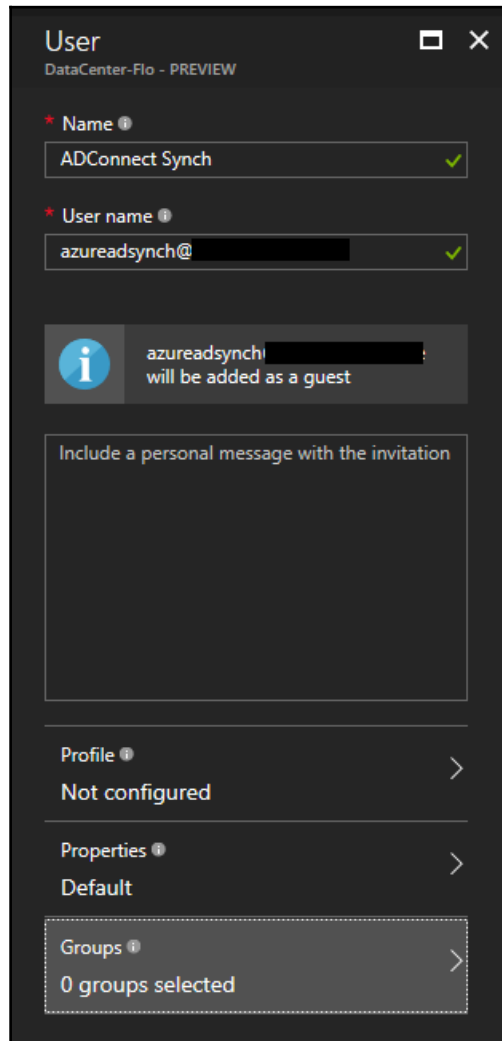
- An Azure subscription or an Azure trial subscription-this is only required for accessing the Azure Portal and not for using Azure AD Connect. If you are using Azure PowerShell or Office 365 you do not need an Azure subscription to use Azure AD Connect.
- An Azure AD global administrator account for the Azure AD tenant you wish to integrate with.
- An AD Domain Controller or member server with Windows Server 2008 or newer (see the following for the appropriate sizing for that machine)

The Domain Controller or member server you will be using as the Azure AD Connect machine in your environment must meet the following minimum specifications:

Number of objects in AD	CPU	Memory	Hard drive size
Fewer than 10,000	1.6 GHz	4 GB	70 GB
10,000 to 50,000	1.6 GHz	4 GB	70 GB
50,000 to 100,000	1.6 GHz	16 GB	100 GB
For 100,000 or more objects the full version of SQL Server is required otherwise Windows internal database or SQL Express can be used.			
100,000 to 300,000	1.6 GHz	32 GB	300 GB
300,000 to 600,000	1.6 GHz	32 GB	450 GB
More than 600,000	1.6 GHz	32 GB	500 GB

First before you start the installation of the AD Connect you need to configure two user accounts. The first one should be a service account with enterprise admin rights or for subdomains with domain admin rights within your on-premises Active Directory Domain Service. The other one must be a global administrator in Azure Active Directory.

The following screenshot shows you the **User** settings you need for the Azure AD synch administrator:



User
DataCenter-Flo - PREVIEW

* **Name** ⓘ
ADConnect Synch ✓

* **User name** ⓘ
azureadsynch@ ✓

azureadsynch@ will be added as a guest

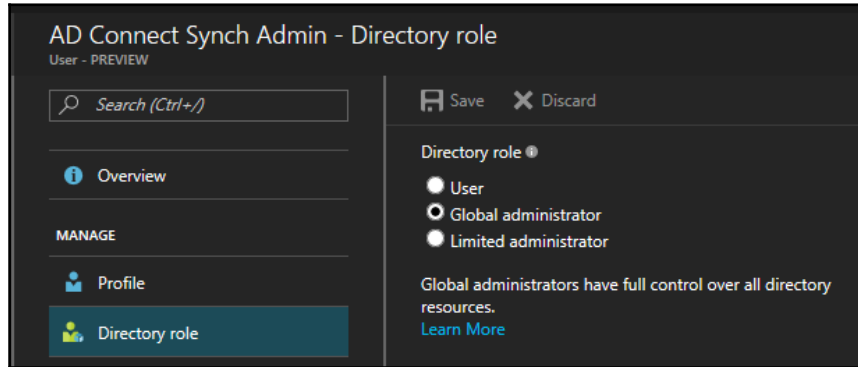
Include a personal message with the invitation

Profile ⓘ
Not configured >

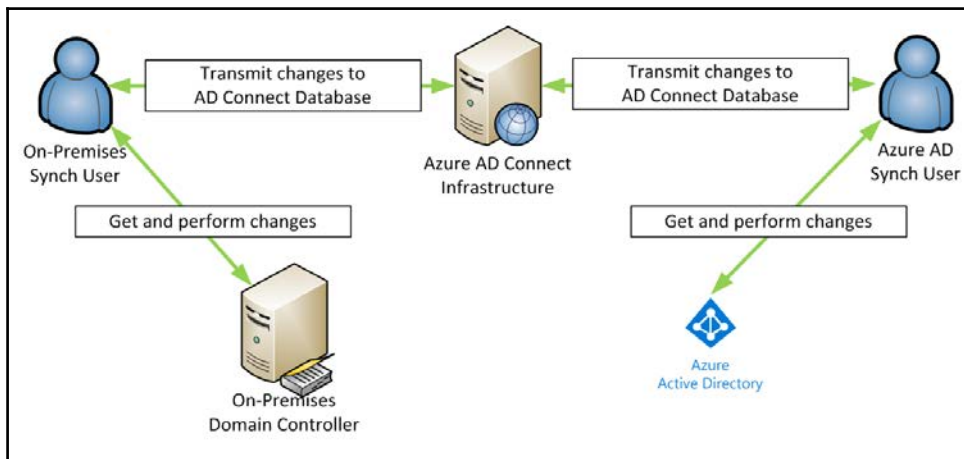
Properties ⓘ
Default >

Groups ⓘ
0 groups selected >

These two accounts will perform actions on both directories. So, the on-premises user won't have access to Azure AD, and the Azure AD admin won't have access to the on-premises AD:

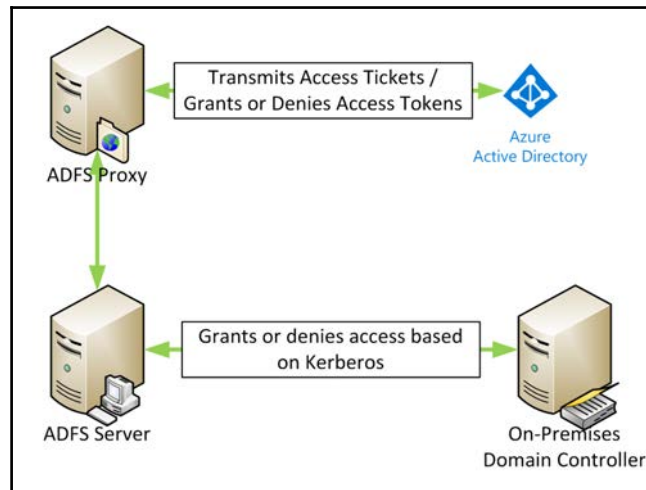


The following diagram shows you the basic workflow behind the communication from Azure AD, AD Connect and on-premises Active Directory Domain Service:



After you choose which type of user sign-on you want to use for your Azure AD users implementation, the wizard changes and adds or removes configuration options. For the ADFS implementation you need to perform some more steps to get the federation running. For the ADFS implementation, you will need at least one **ADFS Server** and one **ADFS Proxy** additionally as well as additional service accounts. The connection will work automatically based on **Windows Remote Management (WinRM)**. As explained earlier, the Azure AD will authenticate against your on-premises ADFS implementation.

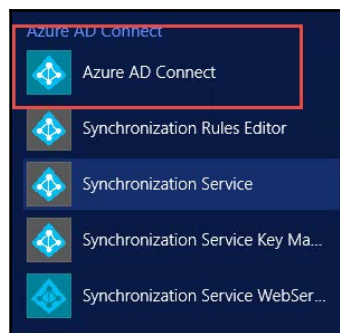
The following diagram reflects the more complex environment:



Installing Basic Azure AD Connect Environment

As an example, we will show you how to implement Azure AD Connect with password synchronization. The other options for the setup are similar:

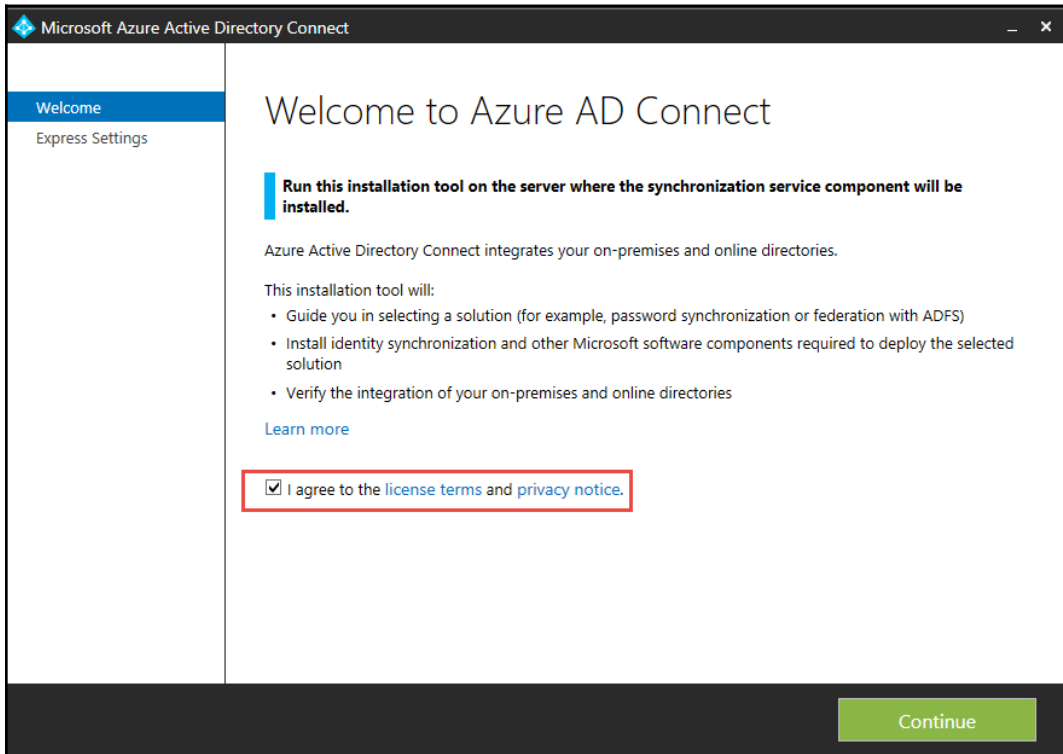
1. First you need to download **Azure AD Connect**. Within the setup you will have to do the configurations for the synchronization but you can redo your setup by looking for **Azure AD Connect** in the Windows Start menu:



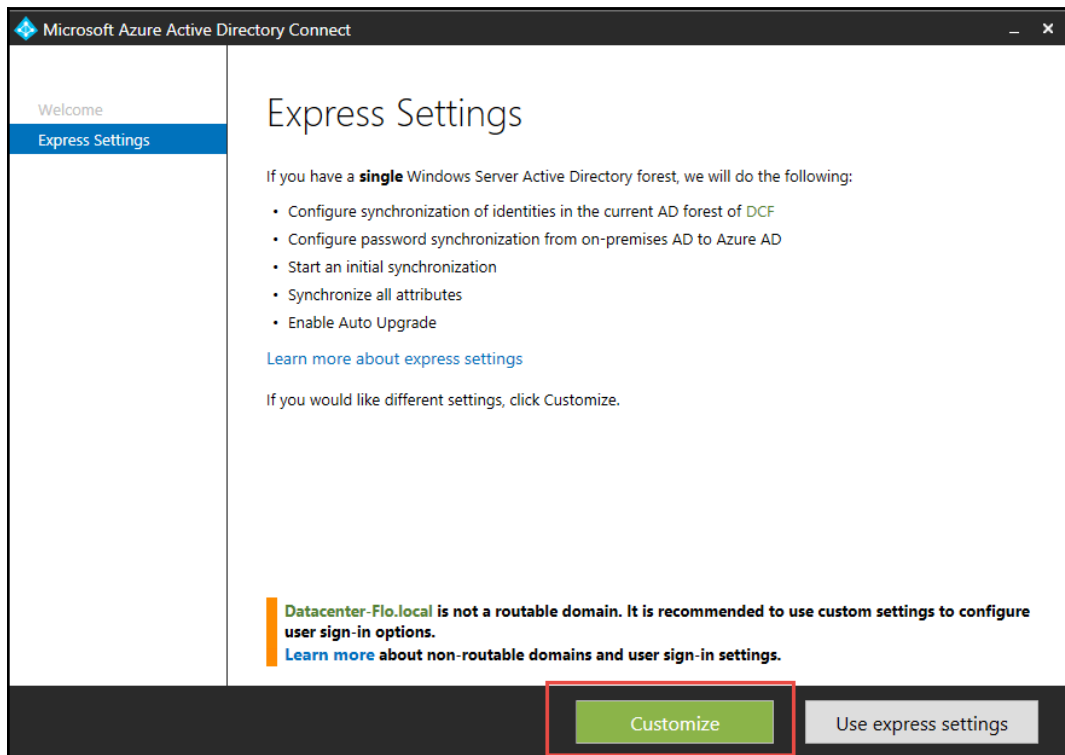


To be aware, your AD Connect Server must be joined to the on-premises Active Directory Domain.

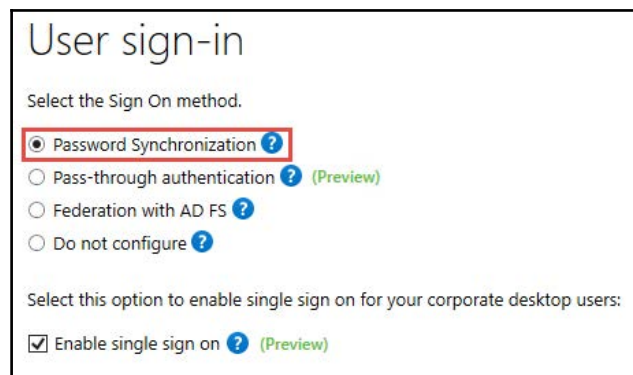
2. Accept the license terms and click on **Continue**:



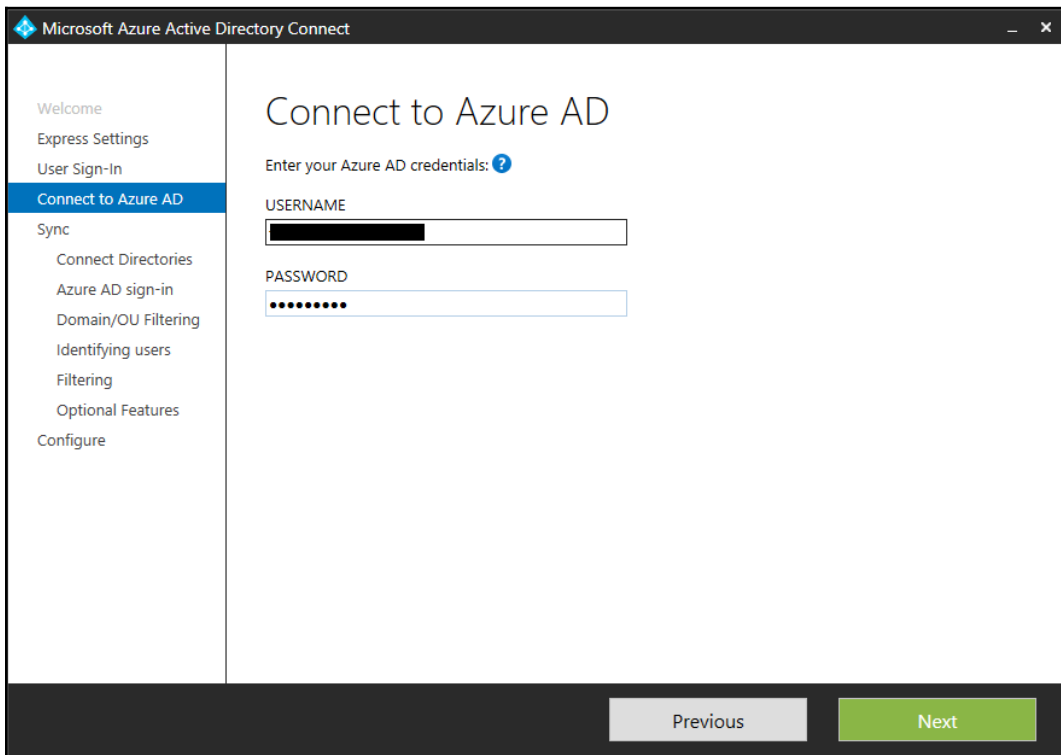
3. Choose **Customize** to continue:



4. Choose **Password Synchronization** and click on **Next**:



5. In the next window enter the credentials of your Azure AD `synch` account:



The screenshot shows the 'Microsoft Azure Active Directory Connect' application window. The title bar includes the Microsoft logo and the text 'Microsoft Azure Active Directory Connect'. On the left is a navigation pane with the following items: 'Welcome', 'Express Settings', 'User Sign-In', 'Connect to Azure AD' (highlighted in blue), 'Sync', 'Connect Directories', 'Azure AD sign-in', 'Domain/OU Filtering', 'Identifying users', 'Filtering', 'Optional Features', and 'Configure'. The main content area is titled 'Connect to Azure AD' and contains the instruction 'Enter your Azure AD credentials: ?'. Below this are two input fields: 'USERNAME' with a blacked-out value and 'PASSWORD' with masked characters (dots). At the bottom right of the window are two buttons: 'Previous' (disabled, grey) and 'Next' (active, green).

6. On many websites and also Microsoft documentations you will find terms like **DirSync** and **AD Sync**. Both are out of support since Azure AD Connect reached production state. Please visit this website for more information http://blog.azureandbeyond.com/2017/04/06/dirsync-azure-ad-sync-end-of-support/?fb_action_ids=1199177306847421&fb_action_types=news.publishes

7. Next, you need to add your on-premises administrator account:

Microsoft Azure Active Directory Connect

Welcome

Express Settings

User Sign-In

Connect to Azure AD

Sync

Connect Directories

Azure AD sign-in

Domain/OU Filtering

Identifying users

Filtering

Optional Features

Configure

Connect your directories

Enter connection information for your on-premises directories or forests: ?

DIRECTORY TYPE

Active Directory

FOREST ?

Datacenter-Flo.local

USERNAME

DCFadministrator

PASSWORD

.....

Add Directory

CONFIGURED DIRECTORIES

Datacenter-Flo.local (Active Directory) ✓

Remove

Previous

Next



TIP

For multi domain forest Active Directory infrastructures, your admin account on-premises need to be an *enterprise administrator*. If you only have a single domain forest, you can use a regular domain admin.

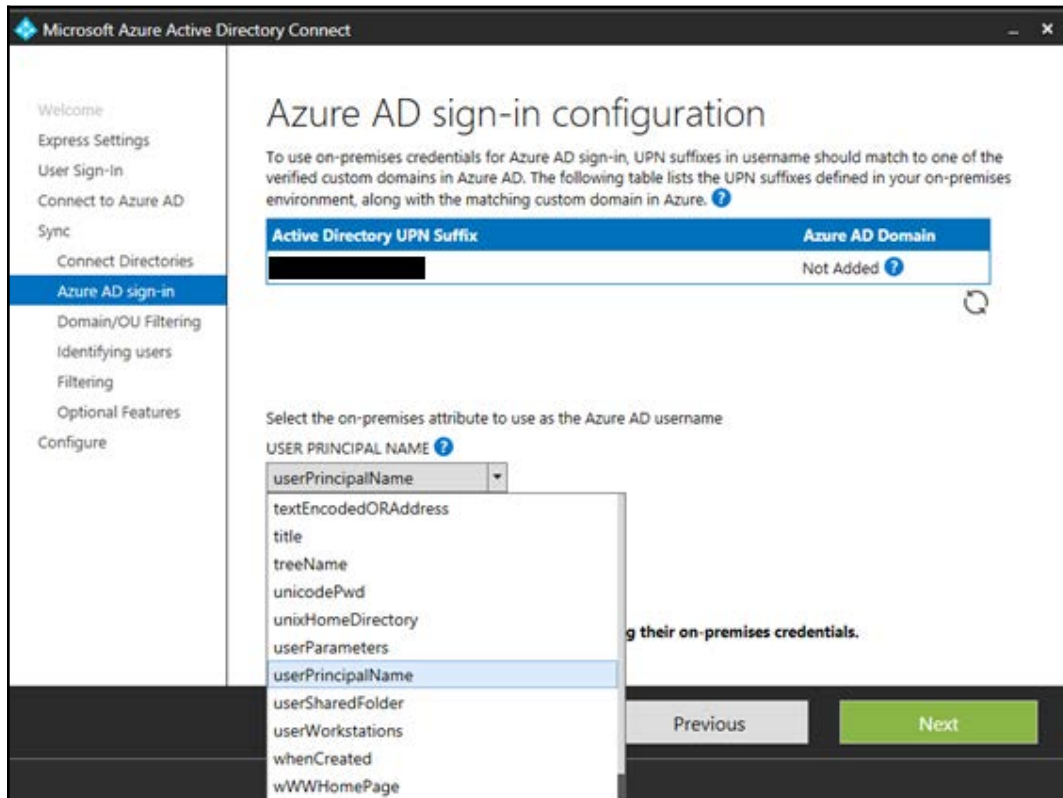
8. For the next step, you need to select the attribute in your on-premises domain which identifies the **User Principal Name (UPN)** in the Azure AD.



TIP

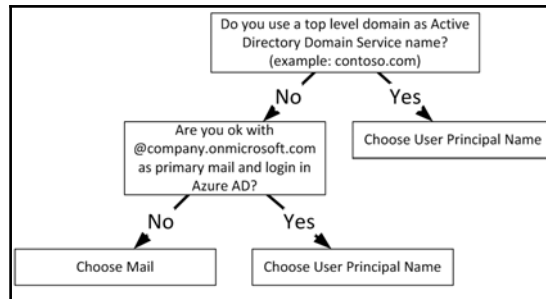
The UPN is an Internet-style login name for a user based on the Internet standard RFC 822.

9. Microsoft best practice is to select the user principal name of your on-premises domain. This isn't the best option in all scenarios:

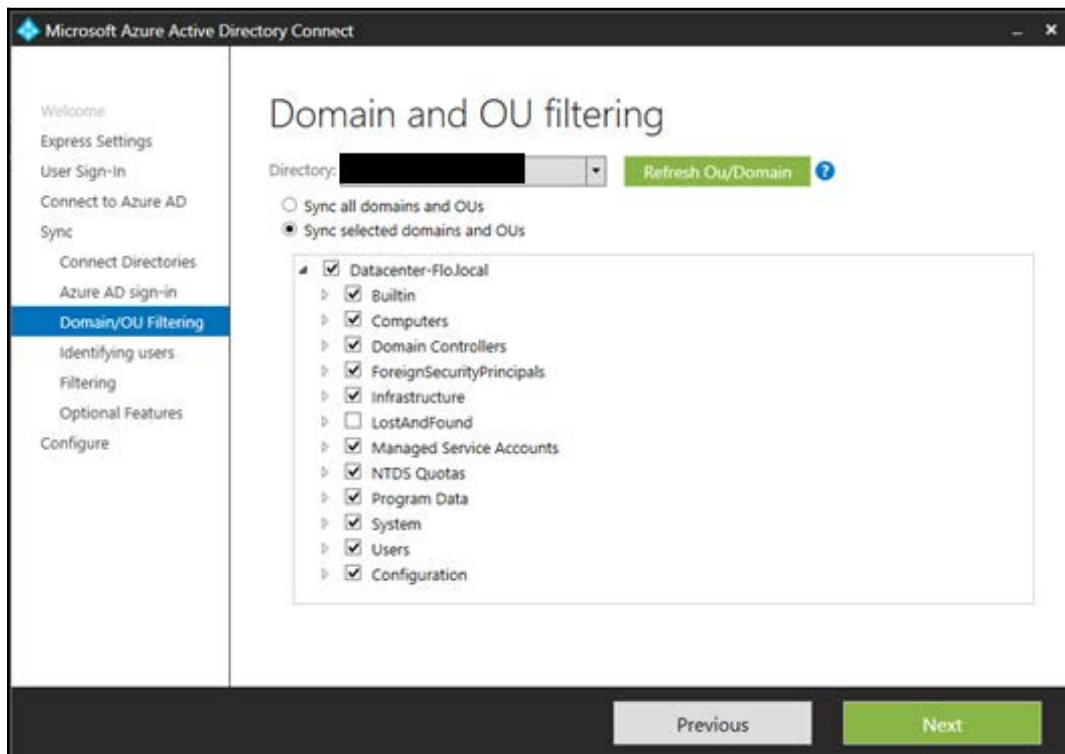


Just from the field, it is not always helpful to choose the user principal. Depending on the domain, sometimes it is better to choose the e-mail instead of the user principal name in Azure AD. That's because the primary e-mail in a synced scenario is always the login name. So, if your login name is `@company.onmicrosoft.com` the primary e-mail would become this domain too.

10. The following diagram shows you an example for a decision path, when to choose UPN and when an e-mail address. It's very basic but helps in most cases:



11. After you select which attribute that will be your identifier, you need to set the AD filter. It isn't recommended to parse the whole ADDS for synchronization, because that would take a lot of time, unless you have a large on-premises Active Directory. For faster synchronization, it is recommended to select only a subset of the on-premises Active Directory objects; for example, select only the **organizational units** or **OUs** that contains the requested objects:



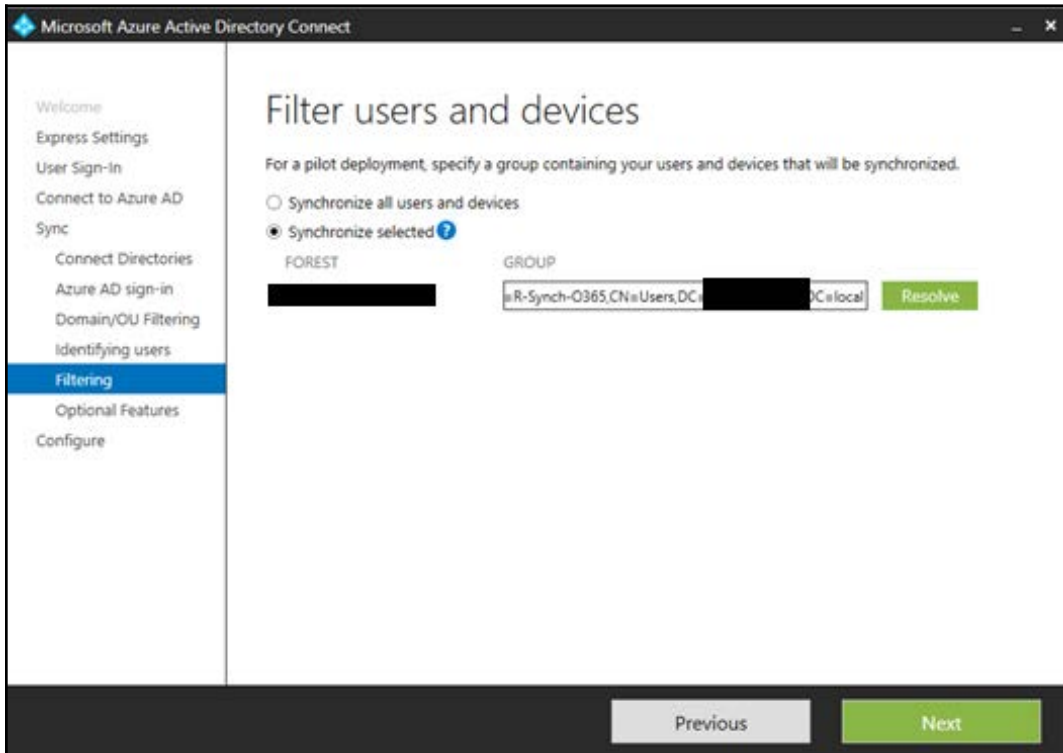
12. Now choose how your users should be identified over your domain.
Depending of your decision you now select user principal name or e-mail:

The screenshot shows the 'Microsoft Azure Active Directory Connect' window. The left sidebar contains a list of steps: Welcome, Express Settings, User Sign-in, Connect to Azure AD, Sync, Connect Directories, Azure AD sign-in, Domain/OU Filtering, **Identifying users** (highlighted), Filtering, Optional Features, and Configure. The main content area is titled 'Uniquely identifying your users'. It contains the following options:

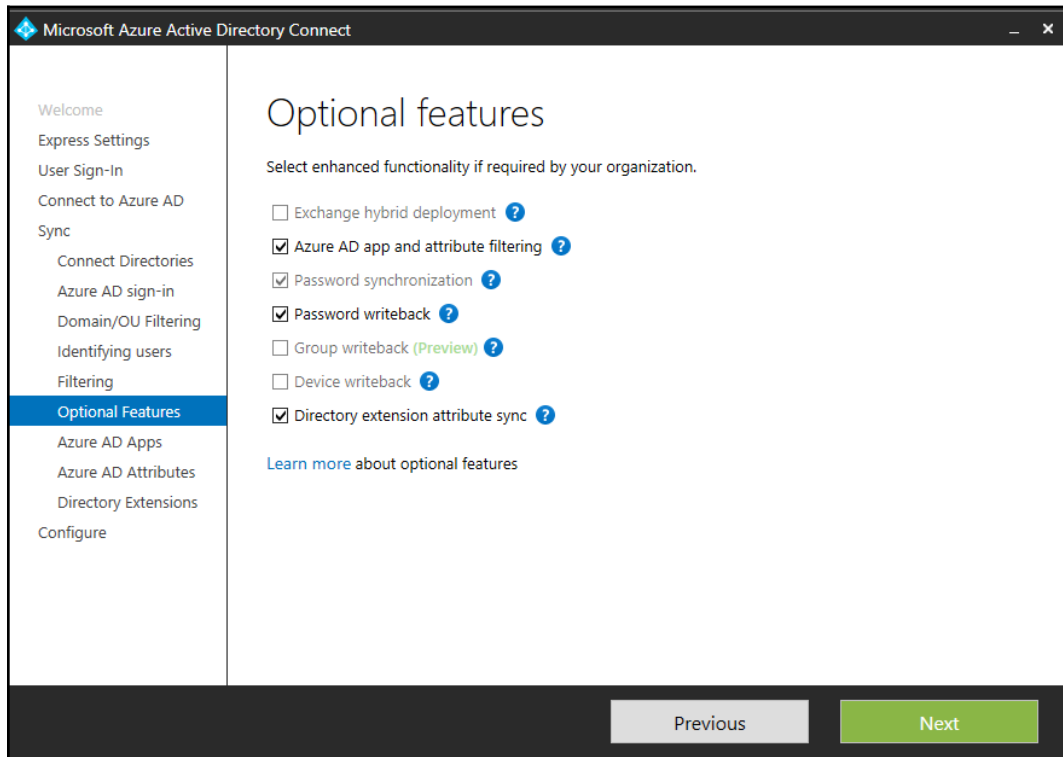
- Select how users should be identified in your on-premises directories. ?
 - ☒ Users are represented only once across all directories.
 - ☐ User identities exist across multiple directories. Match using:
 - ☒ Mail attribute
 - ☐ ObjectSID and msExchMasterAccountSID/msRTCSIP-OriginatorSID attributes
 - ☐ SAMAccountName and MailNickName attributes
 - ☐ A specific attribute
- CUSTOM ATTRIBUTE: [Text box with dropdown arrow]
- Select how users should be identified with Azure AD.
SOURCE ANCHOR ?
[objectGUID dropdown]

At the bottom right, there are two buttons: 'Previous' (disabled) and 'Next' (active).

13. Afterwards configure the filter for your accounts. You can either synchronize all users and device or use a filter group. Here the best way is to have specified group for your users. From a security and resource perspective you shouldn't select all users otherwise you would sync unnecessary or possibly restricted accounts to Azure AD:

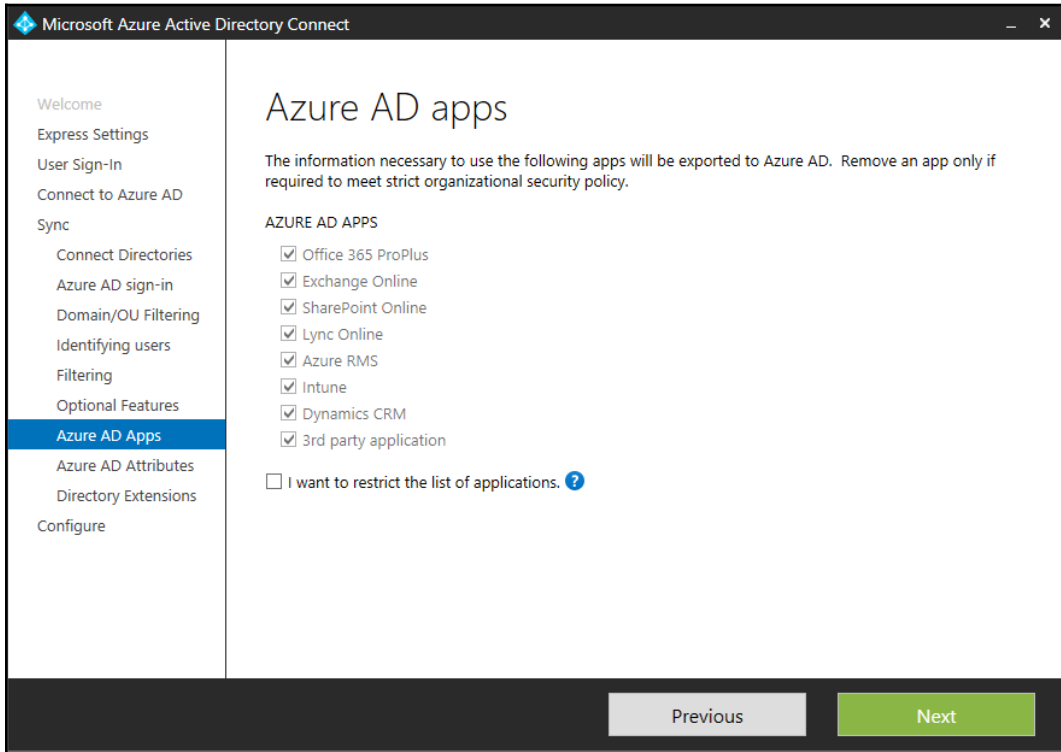


14. In the next steps, you choose which features you want to have synced in addition to the attributes:

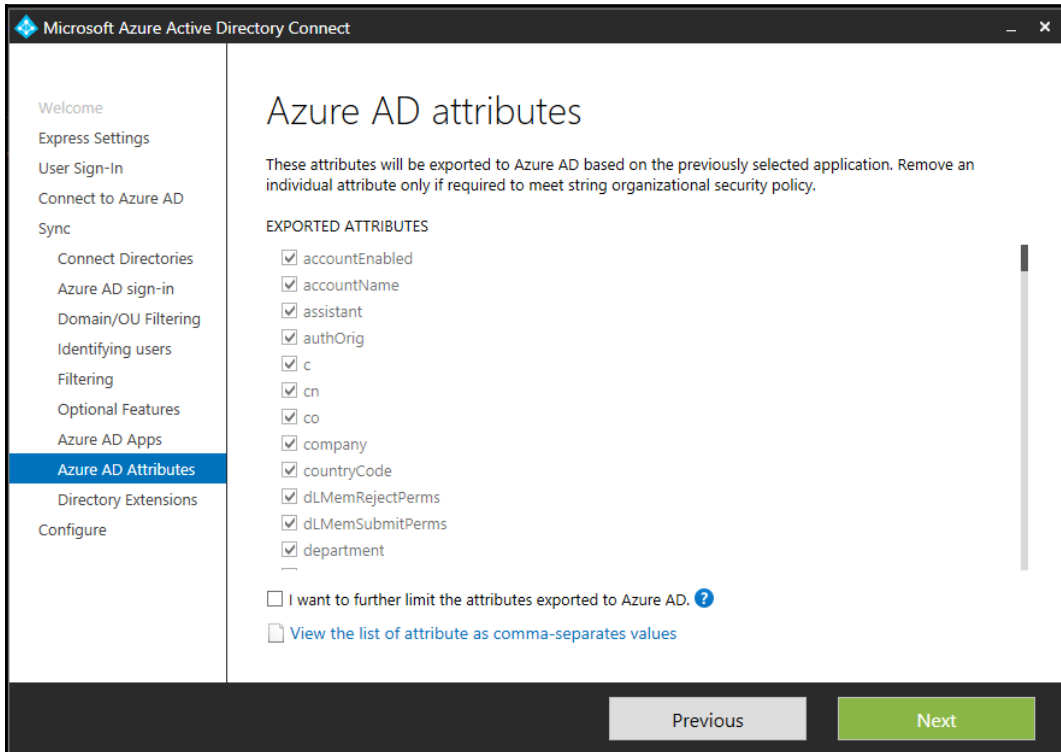


Some, like **Exchange hybrid deployment** and **Device writeback**, are only available with certain Active Directory extensions or with additional subscriptions like Azure AD premium or mobile device management like Intune. Best field practice is to at least sync the **Azure AD app and attribute filtering**. That will enable you to sync allowed apps and license attributes between Azure AD and Active Directory, which is, for example, necessary if you license your Microsoft Office in your Terminal server environment with Office 365 E3 or E5 plans. If you do not choose to enable the feature, it could be possible that some applications will not appear to be licensed. Office 365 installed on a Citrix Terminal Server is a good example for that. Without synchronization, Citrix does not know about the license details and the user receives an Office licensing login every time he starts his Office applications on the Terminal Server.

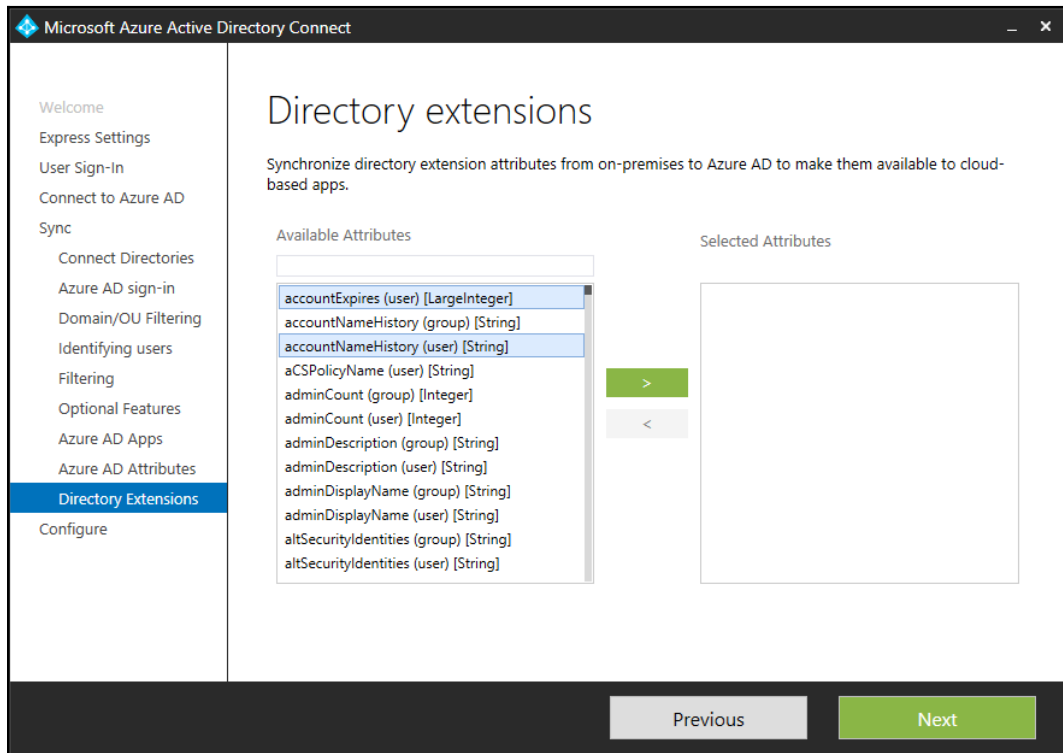
15. In the next few windows, you can decide which **Azure AD apps** will be synced:



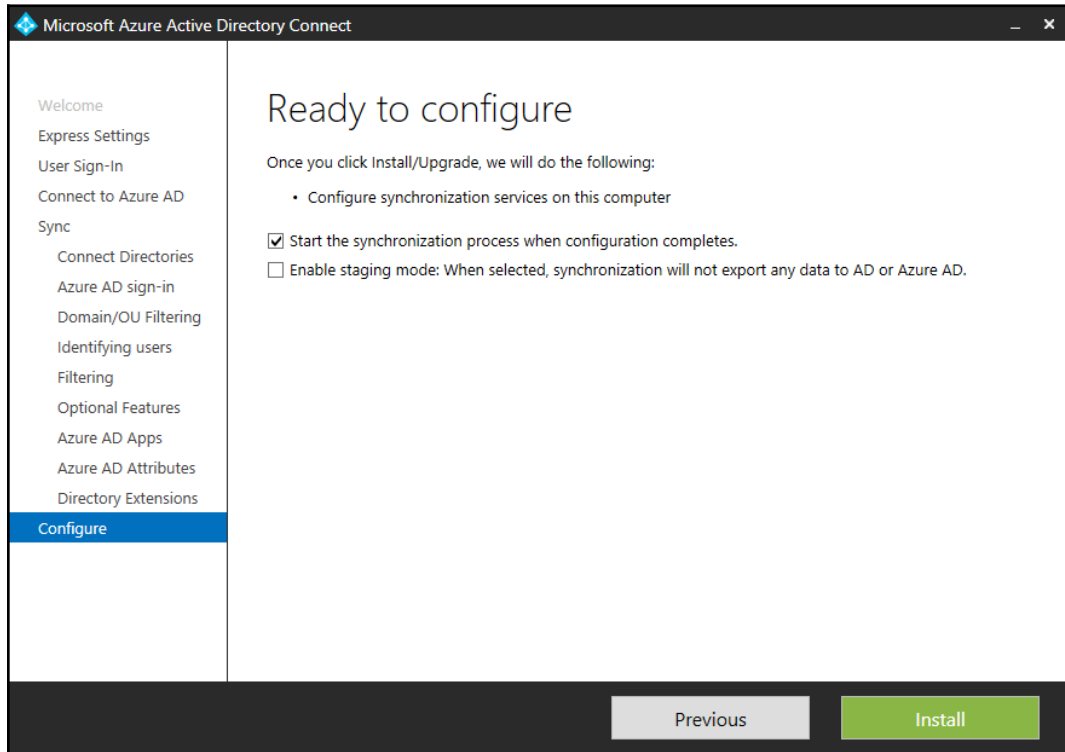
16. Afterwards you can limit the attributes from Active Directory that will be transferred to Azure AD. Microsoft does not recommend limiting the attributes:



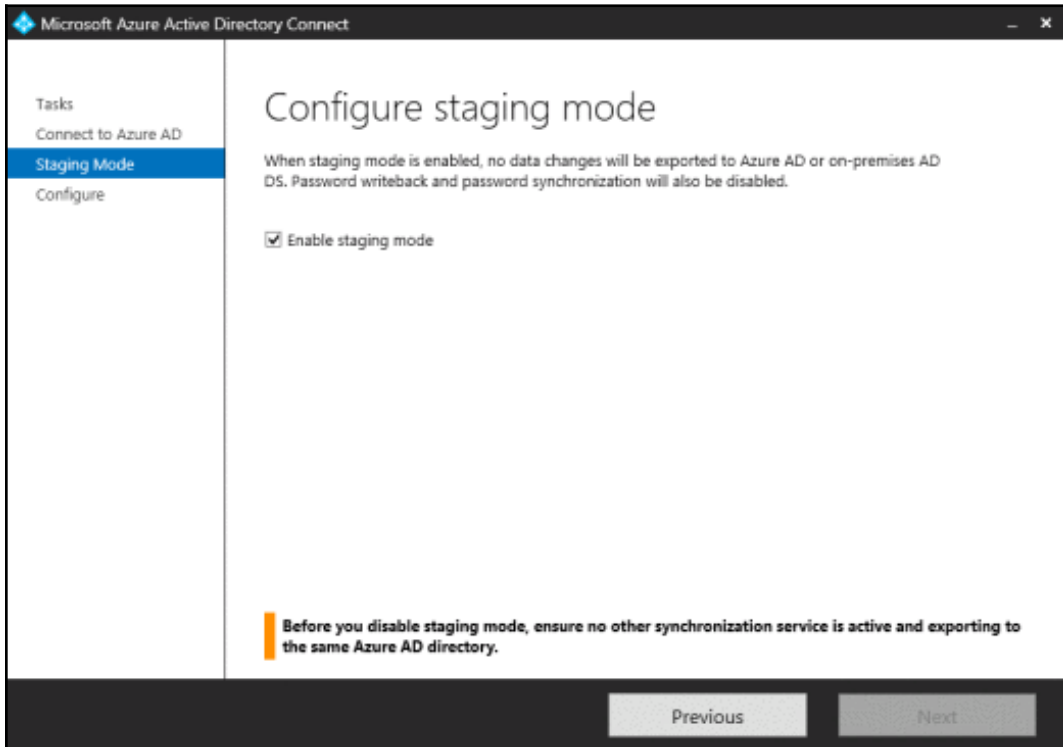
17. Now you can select which **Directory extension** attributes from your on-premises environment are transferred to Azure AD. This is important for certain applications running in the cloud. In this walkthrough, we will not select any attribute:



18. Once you've run through these steps, you can complete the installation. Outside of a migration scenario, or as long as you don't want to enable high availability for Azure AD Connect, you can start synchronization. Otherwise you should stage your installation first and start the first synchronization after:



19. To disable the staging mode, you need to start Azure AD Connect from the Start menu again. It will act like a regular Azure AD Connect, except that the second wizard option is to disable the staging mode:



You have now implemented a simple SSO-enabled Azure AD and Active Directory synchronization.

After you've started synchronization, Azure AD Connect will replicate changes every hour between Azure AD and Active Directory.

You can either trigger synchronization manually by using following PowerShell commands:

- With `Start-ADSyncSyncCycle -PolicyType Delta`, you sync all changes since the last sync.

- With `Start-ADSyncSyncCycle -PolicyType Initial`, you perform a full sync with all settings you have configured in Azure AD Connect. It's likely to be what you would do when you start *sync* after installation.



From a common practice a security standpoint, you should not sync any on-premises domain admin account in Azure AD or shouldn't give the personal Azure AD account of you IT people any admin rights in Azure AD. Anytime you should create cloud only accounts for your admins, which also only serve as admin accounts in Azure AD. That protects both directory from capturing if any of the admins is corrupted.

Azure AD Connect High Available Infrastructure

Now you know how to set up a basic AD synchronization without considering availability infrastructure, and now we'll look at how you can achieve Azure AD synchronization in a high availability environment.

The first thing you should know is that the Azure AD Connect tool cannot be clustered, so you need to use the *staged mode* to implement it in passive mode.

So, for placement in either high availability or non-high availability infrastructures, it is recommended that you place the systems which are involved in the synchronization in Azure virtual machines. This is so that you do not transfer as much data through the open Internet. More details about these concept will be explained in the next chapter about Azure networking.

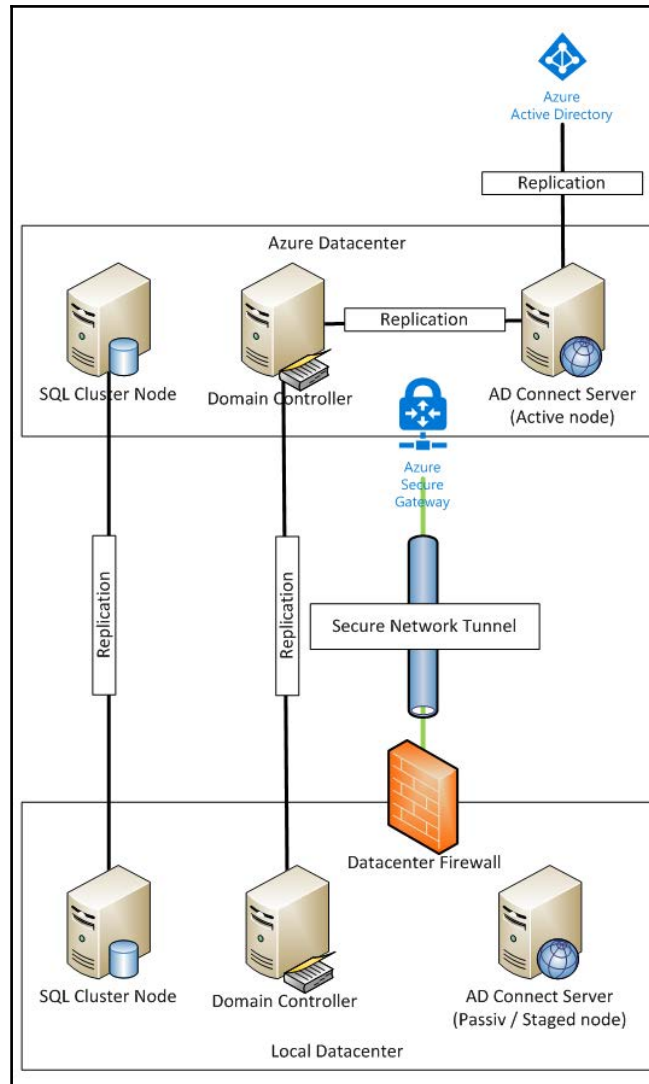
In our high availability scenario, every active and primary source of synchronization is placed into Azure. The backup and passive parts are based into the on-premises data center and connected via VPN or MPLS to Azure.

So, to have a user and password synchronization option in a high availability environment, we need the following system:

- Two Active Directory domain controllers with Global Catalogue and DNS
- Two SQL server database servers within an application cluster
- Two Azure AD Connect servers, one in active and one in staged mode

For the domain controllers and database servers, both will automatically fail-over if one system falls out. For the Azure AD Connect server you need to disable the staged mode and perform the fail-over manually.

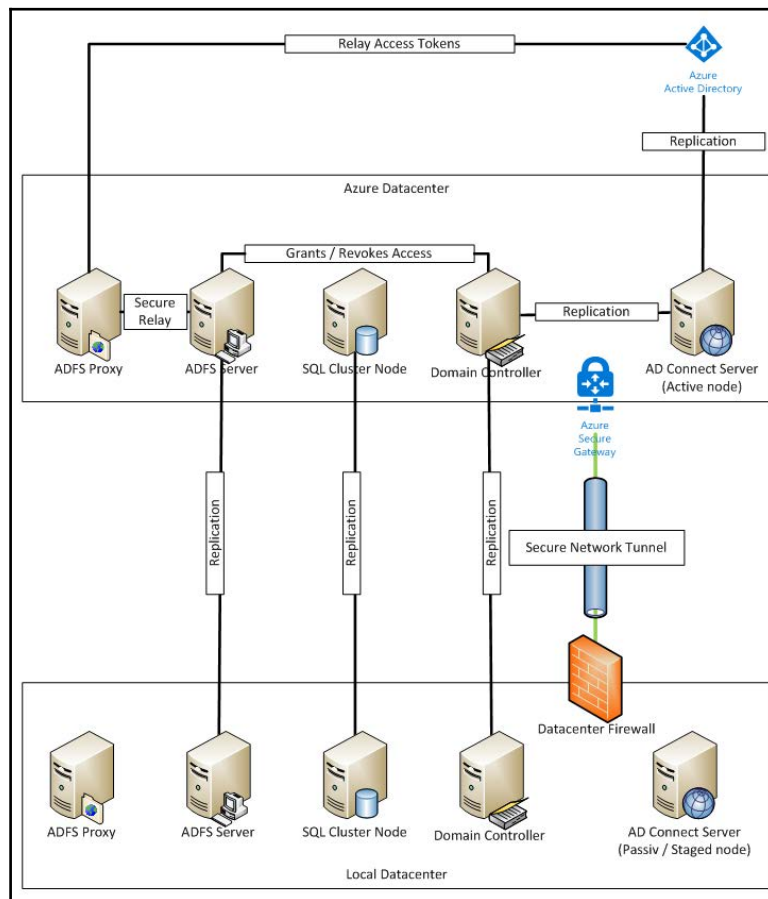
The following diagram shows you how such an infrastructure could look:



Looking at a more complex scenario, if you want to implement a high availability ADFS infrastructure some additional systems are needed:

- Two Active Directory domain controllers with Global Catalogue and DNS
- Two SQL Server database servers within an application cluster
- Two Azure Active Directory Connect servers, one in active and one in staged mode
- Two Active Directory federation servers
- Two Active Directory federation proxies

Such an infrastructure could look like the following architectural schema:



Summary

You should have learned how Azure AD works and what feature each subscription level of Azure AD offers to you. You should also be familiar with creating users in Azure AD and syncing them with your on-premises Active Directory. You saw also other options for user sign in with ADFS and how to make your Azure Hybrid environment highly available.

Within the next chapter, we will cover the internal, external and hybrid networking with and within Azure.

4

Implementing Azure Networks

Just as you plan the network in your data center or company, you need to do it in Azure also. Networking is essential in Azure and if you do not plan it right you could force outages or bottlenecks for deployed services.

Depending on what you plan with Azure, you should put some effort in planning your network and connections into Azure.

Within this chapter, you will learn the basics about Azure networking, how to implement Azure networking, and how to decide which WAN and connectivity solution you should use.

We are going to explore the following sections:

- Azure virtual networks
- Azure VPN gateways
- Azure local gateways
- Azure site-to-site and point-to-site VPN
- Azure ExpressRoute
- Azure connections and routes
- Azure DNS
- Azure application gateway

We will also set up a basic network configuration during this chapter.

Azure Networking Limits

Before we start deep diving into the Azure components, let us take a view on the networking limits in Azure:

Resource	Default limit	Maximum limit
Virtual networks per subscription	50	500
DNS servers per virtual network	9	25
Private IP addresses per virtual network	4,096	4,096
Concurrent TCP connections for a virtual machine or role instance	500k	500k
Network interfaces	300	10k
Network security groups	100	400
NSG rules per NSG	200	500
User defined route tables	100	400
User defined routes per route table	100	400
Public IP addresses (dynamic)	60	Limit can be increased by support
Public IP addresses (static)	20	Limit can be increased by support
Load balancers (internal and Internet facing)	100	Limit can be increased by support
Load balancer rules per load balancer	150	150
Public frontend IP per load balancer	5	Limit can be increased by support
Private frontend IP per load balancer	1	Limit can be increased by support
Application gateways	50	50

There are additional limits for Azure ExpressRoute. We will take a look at them when we are talking about ExpressRoute later on.

Azure Networking Components

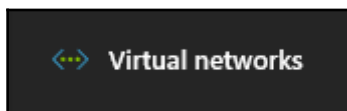
To start with Azure networking, you need to know and understand the components which are needed to set up an Azure solution.

Let us start from the easiest part to the more difficult ones.

Azure Virtual Networks (VNet)

An Azure VNet is a logical isolated network for your services connected to your subscription in Azure. You have full control about the IP address blocks, DNS settings, security policies, and route tables within this network. You can also split your VNet into subnet and launch Azure IaaS virtual machines and cloud services within these subnets. By using Azure virtual gateways and WAN solutions, you can also connect your virtual networks to the Internet or your on-premises environment.

When you look for Azure VNet in Azure, you basically search for the network and you should see the symbol shown in the following screenshot:

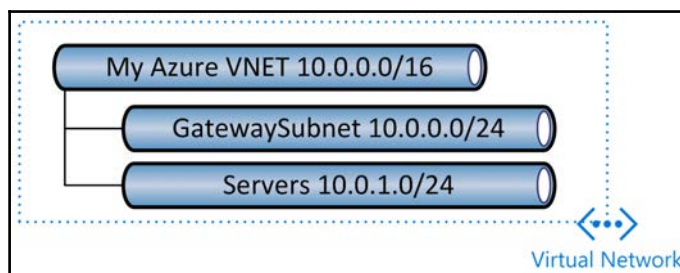


Normally you're setting up a network like you do in your on-premises network. You create a network with an IP range such as `10.0.0.0/16` and split it up into different subnetworks. Every Azure VNet has at least a minimum of two subnets. The first is the **gateway subnet**, which is basically a router network where every internal network router for the other Azure VNet subnetworks is in. We personally prefer to use the first subnet of the Azure VNet as the gateway subnet but you can choose any subnet you like. The only thing you need to know is that the gateway subnet needs a minimum of `/29` CIDR IPs. We normally recommend `/24` CIDR. You would never use it but it's logical and you can follow up with a `/24` CIDR subnet design. The second one is the network for your services or servers depending on your own design, normally it is `/24` CIDR.

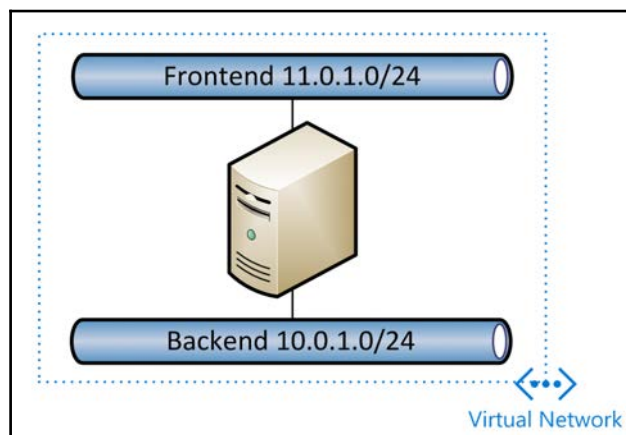


As of September 2016, Azure started to support IPv6 to be used in Azure. The deployment and support of IPv6 is still in progress while writing the book.

The following diagram shows you an example for a network configuration:



All subnetworks are fully routed to each other. That is not the best situation in most of the cases. One example is when you need a **Frontend** and a **Backend** network in Azure, as shown in the following diagram:

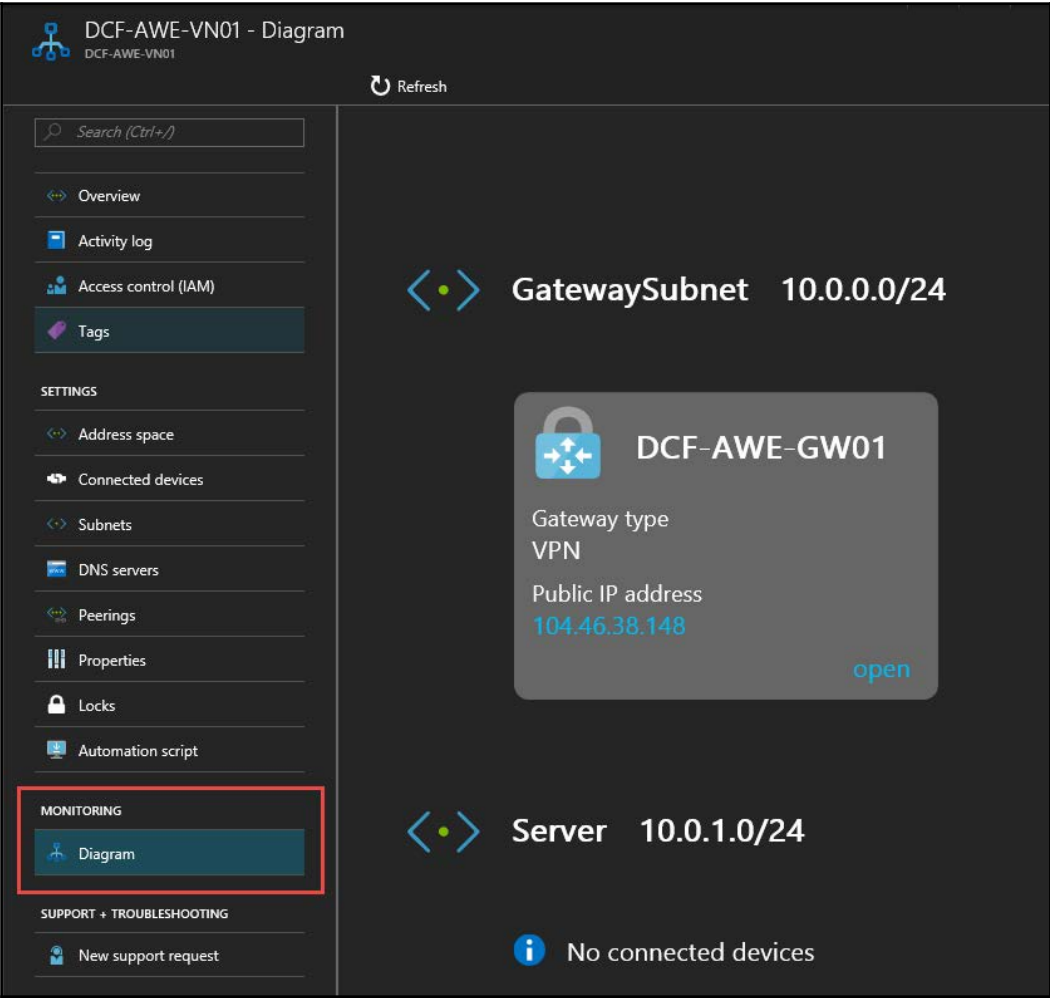


Currently there are only two ways to resolve that issue:

- The first one is to create two VNets and put a virtual machine with two network adapters in both networks and route within the virtual machine.
- The other way is to implement custom routes and send packages for the frontend network into the either of your other Azure or on premises networks.

There is a great *nice to know* within the VNet setting. Under **Monitoring**, you can see a detailed networking diagram of your Azure network.

The following screenshot shows an example of a **Diagram**:



VNet Peering

There is also an option to connect different Azure VNet works in the same Azure data center. This option is called **Azure VNet peering**. VNet peering for Azure virtual network lets you directly link two virtual networks in the same region via private IPs. VNet peering routes packets between virtual networks through the internal Azure backbone network. There is no Azure gateway between these networks. This allows a low-latency, high-bandwidth connection between virtual machines in the virtual networks.

VNet peering also allows transit through the peered virtual networks, so a network virtual appliance or a VPN gateway in one virtual network can be used by a virtual machine in another peered virtual network. Peering works across virtual networks in different subscriptions. So you are able to connect, for example, subscriptions paid by different departments or subscription owners. Later in this chapter, we will show you how to set up VNet peering. In the future, it may also be possible to connect Azure services such as Azure SQL or Azure Storage directly via VNet peering to Azure VNets. That would allow you to use Azure PaaS resources without using a public network or connection in Azure.



As of September 2016, Microsoft allows VNet peering between **Azure Resource Manager (ARM)** environments and **Azure Service Manager (ASM)** environments. That would enable customers to migrate from ASM to ARM more easily.

Azure VPN Gateways

Azure VPN gateways are basically your core routers and firewalls within your Azure environment.

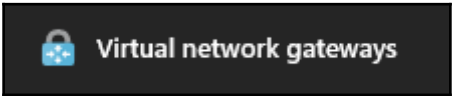
An Azure gateway can serve different purposes:

- Internet gateway
- Site-to-site VPN gateway
- Point-to-site VPN gateway
- ExpressRoute gateway
- VNet-to-VNet gateway



We won't be able to cover the deployments of point-to-site VPN gateways in this book but you can find a detailed guide in the Microsoft documentation at <https://azure.microsoft.com/en-us/documentation/articles/vpn-gateway-howto-point-to-site-rm-ps/>.

The following screenshot shows the Azure service you need to look for when you want to implement an Azure VPN gateway:

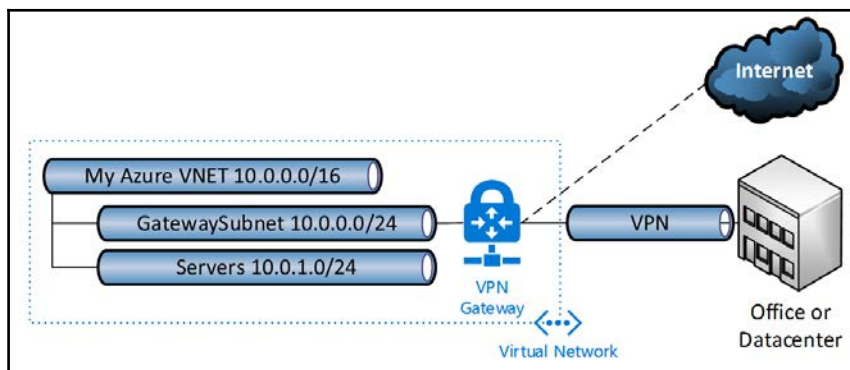


Every VNet can have at least one VPN gateway. VPN gateways are available in different service offerings with different features and available services.

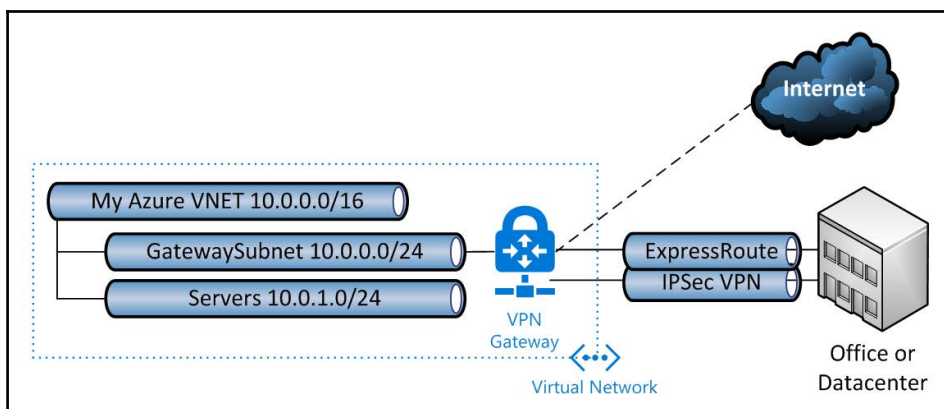
The following table shows a short summary:

	VPN gateway throughput	VPN gateway max IPSEC tunnels	Active - Active VPN	ExpressRoute gateway throughput	VPN gateway and ExpressRoute coexist
Basic	100 Mbps	10	No	No	No
Standard	100 Mbps	10	No	1000 Mbps	Yes
High Performance	200 Mbps	30	Yes	2000 Mbps up to 10000 Mbps	Yes

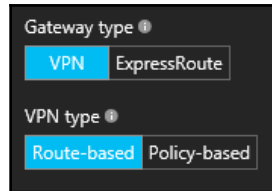
The following diagram shows how the basic **VPN Gateway** is connected to your Azure network:



With the standard or performance gateway it would look like the following diagram:



When you start the setup of a gateway, you need to decide what kind of gateway you want to deploy. The basic offering can be deployed via Azure GUI; for the other offerings, you need to do some PowerShell. The following screenshot shows the GUI version:



Depending on your WAN solution, you choose either **VPN** or **ExpressRoute**. For **ExpressRoute**, you need an MPLS solution in place. This will be explained later. For the VPN solution, you need to decide between a **Route-based** or **Policy-based** VPN, which means you need to decide if you want to enable dynamic routing with IPSEC IKEv2 or static IPSEC IKEv1.

The decision as to which VPN type you need must be done based on your on-premises VPN device. Not every device can speak **Route-based** VPN. Microsoft has published a list of supported devices. You can see them here at <https://azure.microsoft.com/en-us/documentation/articles/vpn-gateway-about-vpn-devices/>.

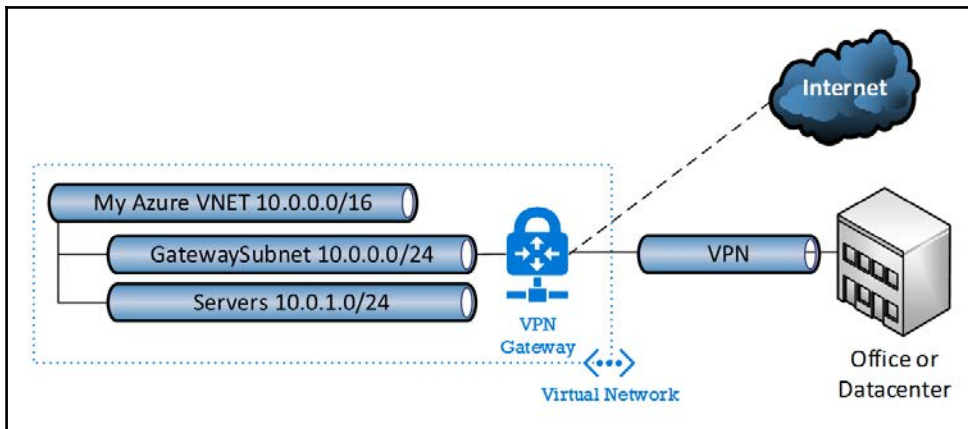
There are also some more additional requirements you need to think of when choosing your VPN gateway in Azure. The following table shows you those provided by Microsoft:

	Policy-based basic VPN gateway	Route-based basic VPN gateway	Route-based standard VPN gateway	Route-based high performance VPN gateway
Site-to-site connectivity (S2S)	Policy-based VPN configuration	Route-based VPN configuration	Route-based VPN configuration	Route-based VPN configuration
Point-to-site connectivity (P2S)	Not supported	Supported (can coexist with S2S)	Supported (can coexist with S2S)	Supported (can coexist with S2S)

Authentication method	Pre-shared key	Pre-shared key for S2S connectivity, certificates for P2S connectivity	Pre-shared key for S2S connectivity, certificates for P2S connectivity	Pre-shared key for S2S connectivity, certificates for P2S connectivity
Maximum number of S2S connections	1	10	10	30
Maximum number of P2S connections	Not supported	128	128	128
Active routing support	Not supported	Not supported	Supported	Supported

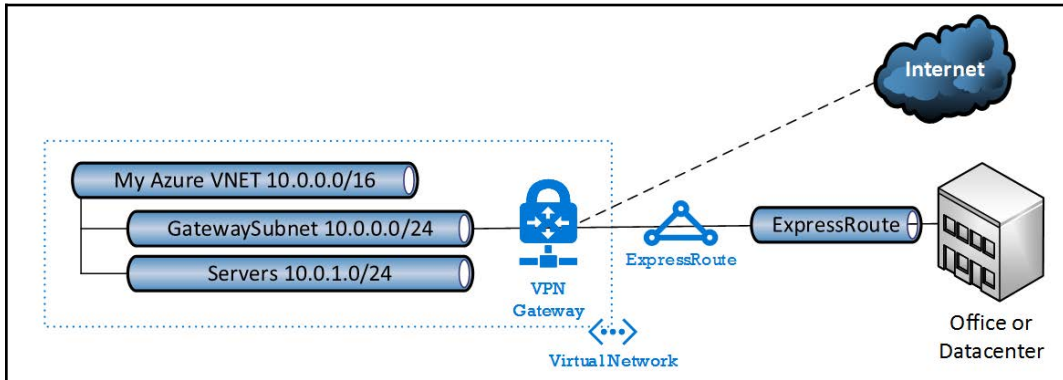
In summary, you can basically have the following gateway configurations:

- The policy-based basic **VPN Gateway** with site-to-site VPN is shown in the following diagram:

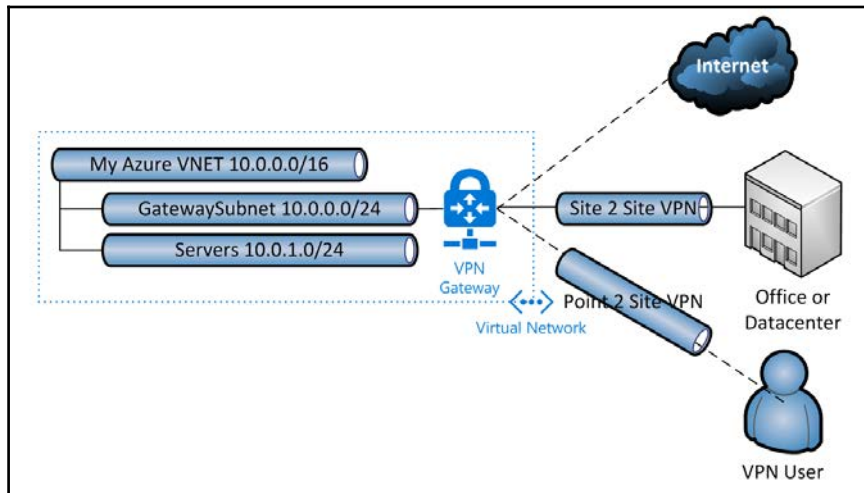


Looking on the current WAN developments and most of the customer infrastructures, a policy-based VPN Gateway should only be used if there is absolutely no other option. Most enterprise grade Firewalls are able to work with route-based VPN. Otherwise you can switch to a virtual network device in Azure.

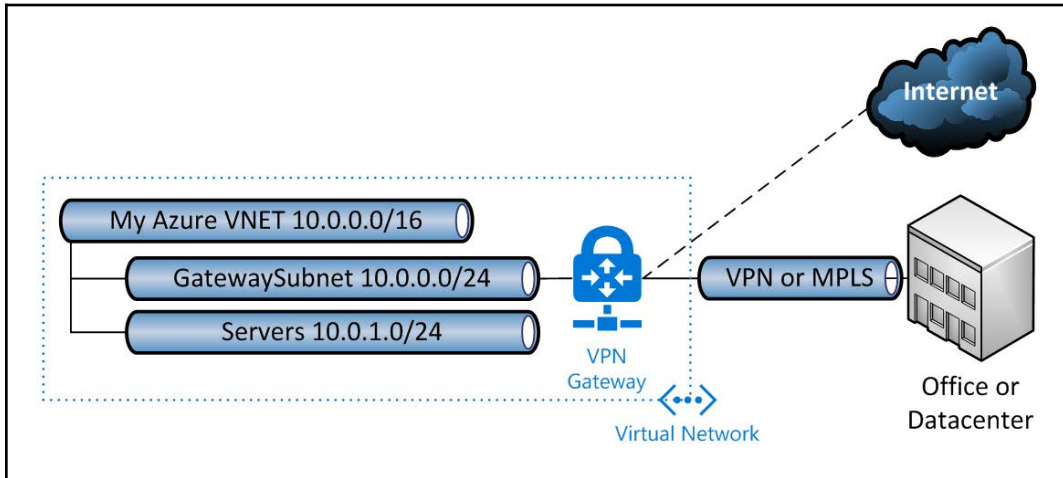
- Route-based standard **VPN Gateway** with **ExpressRoute** shown in the following diagram:



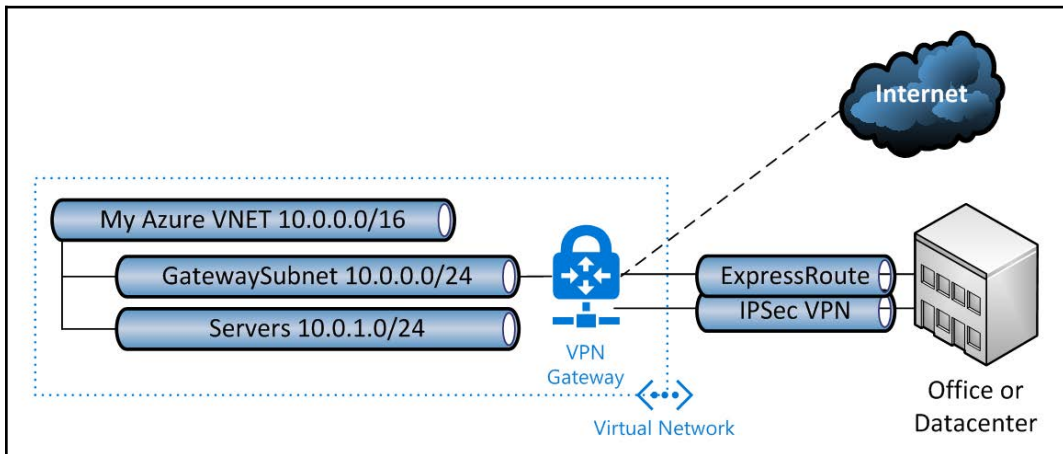
- Route-based basic **VPN Gateway** with a **Site 2 Site VPN** and **Point 2 Site VPN** or a **Route-based** standard or performance **VPN Gateway** with a **Site 2 Site VPN** and **Point 2 Site VPN**:



- Route-based standard or performance **VPN Gateway** with site-to-site or ExpressRoute:



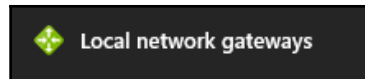
- Route-based standard or performance **VPN Gateway** with a site-to-site VPN and **ExpressRoute**:



Later in the chapter, you will learn how to configure a VPN Gateway with ExpressRoute and a basic VPN with a site-to-site VPN and how to upgrade that VPN to standard or performance. You will also learn what you need to do to implement a point-to-site VPN.

Azure Local Gateway

Local network gateways represent the configuration of your local firewall environment. Within a local network gateway, you configure the public IP of your firewall device as well as the IP spaces you manage within a local environment. The following screenshot shows the Azure service you need to look for when you want to implement an Azure local network gateway:



The following screenshot shows you how to configure a local network gateway:

A screenshot of the "Create local network gateway" form in the Azure portal. The form is dark-themed and contains the following fields and options:

- Name:** A text box containing "LocalNetworkGateway" with a green checkmark icon on the right.
- IP address:** An empty text box.
- Address space:** A section with a list of address ranges. The first two are "192.168.10.0/24" and "16.0.0.0/16", each with a three-dot menu icon to its right. Below the list is a button labeled "Add additional address range" with a three-dot menu icon.
- Subscription:** A dropdown menu showing "Visual Studio Premium with MSDN".
- Resource group:** A section with two radio buttons: "Create new" (selected) and "Use existing". Below the radio buttons is a dropdown menu showing "DCF-AWE-RG01".
- Location:** A dropdown menu showing "West Europe".

Currently, it is not possible to work with DNS entries or dynamic public IPs. Azure is also not supporting IPv6 within the environment as a local network gateway at the present time. So you definitely need a public IPv4 IP for your production environment. That may change in the near future when the IPv6 deployment is moving on in Azure.



There is an option to work with dynamic public IPs but we only recommend that for test environments or home labs. You can use a dynamic DNS provider such as DynDNS to collect your changing IP address. Afterwards, you can recreate your Azure local gateway with the newly obtained IP. MVP Florent Appointaire wrote a little script for the Azure Resource Manager to configure to help you with that; please refer to <https://gallery.technet.microsoft.com/Update-AzureRM-S2S-VPN-c46cc39e>.

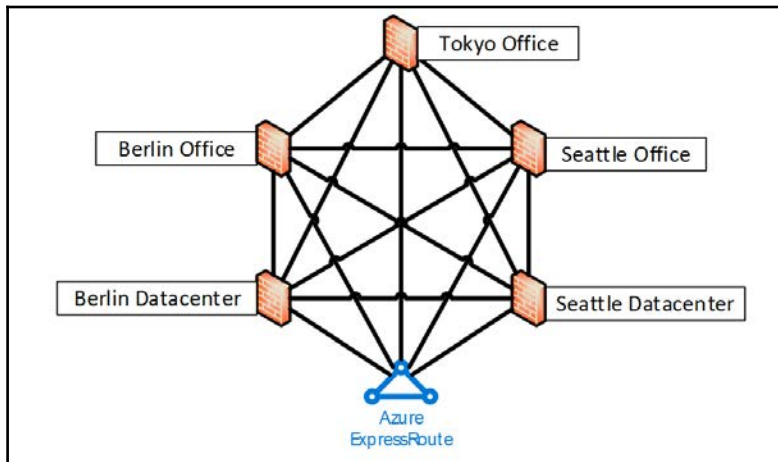
Azure ExpressRoute

When we are talking about ExpressRoute, we are talking about a common **Internet Service Provider (ISP)** technology called **Multi Protocol Label Switching (MPLS)** or **ISP IP VPN**.

MPLS is a type of data-carrying technique for telecommunications networks that directs data from one network to the next based on short path labels rather than long network addresses. This technology avoids long and complex routing tables. The labels identify virtual links between distant nodes. MPLS can encapsulate packets of various network protocols; that's why it is named *multiprotocol*. MPLS supports nearly all common access technologies, including T1/E1, ATM, frame relay, and dark fiber connects, into points of presence or DSL.

The routing within those networks is based on **Border Gateway Protocol (BGP)** routing. BGP is a standardized gateway protocol designed to exchange routing and reachability information among autonomous systems on the Internet. The protocol is often classified as a **path vector** protocol but is sometimes also classed as a **distance-vector routing** protocol. The BGP makes routing decisions based on paths, network policies, or rule sets configured by a network provider and makes core routing decisions.

Normally you see MPLS when connecting a range of offices or data centers with very complex routing or mashed networks between the network sites. MPLS also does not terminate Quality of Service settings at the gateway; all settings can be transported from network site to network site. The following diagram shows an example for such a mashed environment:



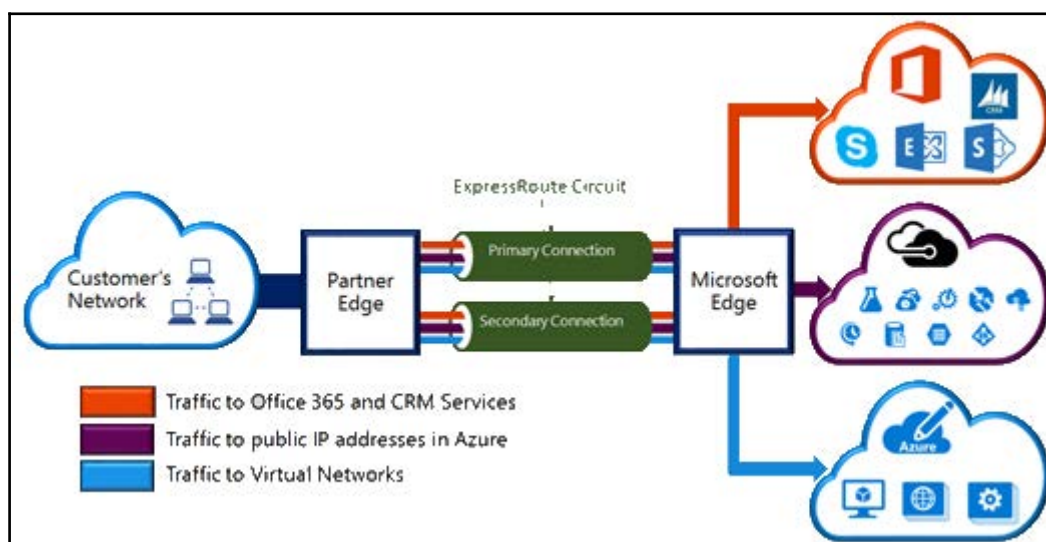
Microsoft offers with ExpressRoute the option to connect your Azure and Office 365 environment directly to your MPLS network.

When you configure, you will be able to configure the different peering's. The three peering types are:

- **Private peering:** Azure compute services, namely virtual machines and cloud services that are deployed within a virtual network can be connected through the private peering domain. The private peering domain is considered to be a trusted extension of your core network into Microsoft Azure. You can set up bidirectional connectivity between your network and Azure virtual networks.
- **Public peering:** Services such as Azure Storage, SQL databases, and websites are offered on public IP addresses. You can privately connect to services hosted on public IP addresses, including VIPs of your cloud services, through the public peering routing domain. You can connect the public peering domain to your DMZ and connect to all Azure services on their public IP addresses from your WAN without having to connect through the Internet.

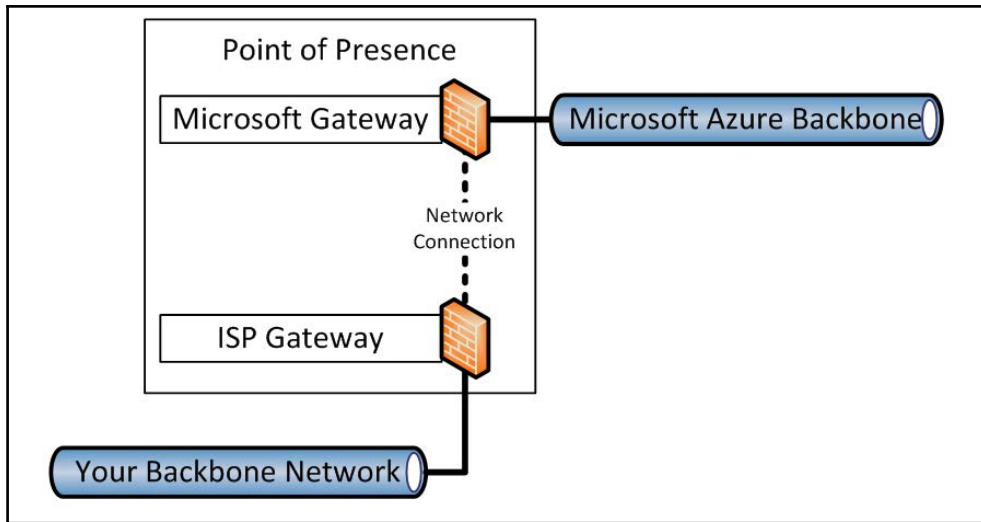
- **Microsoft peering:** Connectivity to all other Microsoft online services (such as Office 365 or Dynamics CRM) will be through the Microsoft peering. You can enable bi-directional connectivity between your WAN and Microsoft cloud services through the Microsoft peering routing domain.

The following diagram shows the basic schema on the Microsoft site:



Source: <https://azure.microsoft.com/en-us/documentation/articles/expressroute-introduction/>

What basically happens is that your ISP connects your network to the network of Microsoft. Those connections happen at most of the **Point of Presence(PoP)** or Internet exchange hubs all over the globe. The following diagram shows how this happens within the Azure data center:



Azure Global Points of Presence

The following table shows you the list of PoPs Microsoft and partner ISPs peer at the moment:

Location	Service providers
Amsterdam	Aryaka Networks, AT&T NetBond, British Telecom, Colt, Equinix, euNetworks, GÉANT+, InterCloud, Internet Solutions-Cloud Connect, Interxion, Level 3 Communications, Orange, Tata Communications, TeleCity Group, Telenor, Verizon
Atlanta	Equinix
Chennai	Tata Communications
Chicago	AT&T NetBond, Comcast, Equinix, Level 3 Communications, Zayo Group
Dallas	AT&T NetBond, Equinix, Level 3 Communications, Megaport
Dublin	Colt
Hong Kong	British Telecom, China Telecom Global, Equinix, Megaport, Orange, PCCW Global Limited, Tata Communications, Verizon
London	AT&T NetBond, British Telecom, Colt, Equinix, InterCloud, Internet Solutions-Cloud Connect, Interxion, Jisc+, Level 3 Communications, MTN, NTT Communications, Orange, Tata Communications, Telecity Group, Telenor, Verizon, Vodafone

Las Vegas	Level 3 Communications+, Megaport
Los Angeles	CoreSite, Equinix, Megaport, NTT, Zayo Group
Melbourne	Equinix, Megaport, NEXTDC, Telstra Corporation
New York	Equinix, Megaport, Zayo Group
Montreal	Cologix+
Mumbai	Tata Communications
Osaka	Equinix, Internet Initiative Japan Inc.-IIJ, NTT Communications, Softbank
Paris	Interxion
Sao Paulo	Equinix, Telefonica
Seattle	Equinix, Level 3 Communications, Megaport
Silicon Valley	Aryaka Networks, AT&T NetBond, British Telecom, CenturyLink+, Comcast, Equinix, Level 3 Communications, Orange, Tata Communications, Verizon, Zayo Group
Singapore	Aryaka Networks, AT&T NetBond, British Telecom, Equinix, InterCloud, Megaport, Orange, SingTel, Tata Communications, Verizon
Sydney	AT&T NetBond, British Telecom, Equinix, Megaport, NEXTDC, Orange, Telstra Corporation, Verizon
Tokyo	Aryaka Networks, British Telecom, Colt, Equinix, Internet Initiative Japan Inc.-IIJ, NTT Communications, Softbank, Verizon
Toronto	Cologix, Equinix, Zayo Group
Washington DC	Aryaka Networks, AT&T NetBond, British Telecom, Comcast, Equinix, InterCloud, Level 3 Communications, Megaport, Orange, Tata Communications, Verizon, Zayo Group

National Azure Points of Presence

For the National Azure environment or Azure environment disconnected from Azure global, Microsoft offers different PoPs to connect to. Those PoPs are completely disconnected from the Azure Global PoPs. The internet services providers to connect to are in the following table.

National Azure	Location	Service providers
US Government	Chicago	AT&T NetBond, Equinix, Level 3 Communications, Verizon
US Government	Dallas	Equinix+, Verizon+
US Government	New York	Equinix, Level 3 Communications+, Verizon
US Government	Washington DC	AT&T NetBond, Equinix, Level 3 Communications, Verizon
China	Beijing	China Telecom
China	Shanghai	China Telecom
Germany	Berlin	Colt, e-shelter, Megaport
Germany	Frankfurt	Colt, Equinix, Interxion

If your provider is not directly listed, you need to get in contact with your ISP to find the right location and partner for deployment.



To find information about the Azure PoPs and peering partners, you can visit the Azure documentation website at <https://azure.microsoft.com/en-us/documentation/articles/expressroute-locations-providers/>.

Microsoft also started to maintain a list of direct through ISPs, those ISP who leverage Equinix, Interxion, e-shelter etc. to connect to Azure ExpressRoute. The list can be found in the Azure Documentation visiting following website <https://docs.microsoft.com/en-us/azure/expressroute/expressroute-locations-providers#a-namec1partnersaconnectivity-through-service-providers-not-listed>

Another point Microsoft also started is to name certified and qualified Solution Integrator for ExpressRoute which support customers with planning, deploying and maintaining ExpressRoute in a customer environment. Microsoft maintains the list of those Partners on their Azure documentation website <https://docs.microsoft.com/en-us/azure/expressroute/expressroute-locations-providers#expressroute-system-integrators>.

Microsoft offers ExpressRoute in the following two service levels: **Standard SLA** and **Premium SLA**. As described next, the premium offering expands the standard offering in the following limits:

- Increased routing table limit from 4k routes to 10k routes for private peering.
- Increased number of VNets that can be connected to the ExpressRoute circuit (default is 10).
- Global connectivity over the Microsoft core network. You will now be able to link a VNet in one geopolitical region with an ExpressRoute circuit in another region. Example: You can link a VNet created in Europe West to an ExpressRoute circuit created in Silicon Valley.
- Connectivity to Office 365 services and CRM Online.

Depending on the bought ExpressRoute service level there are different limitations:

Resource	Default limit
ExpressRoute circuits per subscription	10
ExpressRoute circuits per region per subscription for ARM	10

Maximum number of routes for Azure private peering with ExpressRoute standard	4,000
Maximum number of routes for Azure private peering with ExpressRoute premium add-on	10,000
Maximum number of routes for Azure public peering with ExpressRoute standard	200
Maximum number of routes for Azure public peering with ExpressRoute premium add-on	200
Maximum number of routes for Azure Microsoft peering with ExpressRoute standard	200
Maximum number of routes for Azure Microsoft peering with ExpressRoute premium add-on	200

Depending on the ISP and network location, Microsoft offers the following bandwidths and connections:

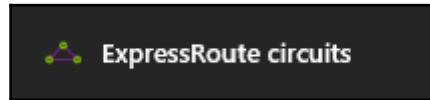
Circuit size	Number of VNet links for standard	Number of VNet links with premium add-on
50 Mbps	10	20
100 Mbps	10	25
200 Mbps	10	25
500 Mbps	10	40
1 Gbps	10	50
2 Gbps	10	60
5 Gbps	10	75
10 Gbps	10	100

ExpressRoute is highly recommended for enterprise environments which need a guarantee for latency and bandwidth for their Azure and Office 365 environment.



Microsoft will also enable a high-performance ExpressRoute circuit. The high-performance ExpressRoute will enable customers to throughput 10 Gbps from the WAN directly to their VM's.

An Azure ExpressRoute circuit is represented in the Azure portal with the following symbol:



Later on in the chapter, we will explain how to deploy an ExpressRoute circuit.

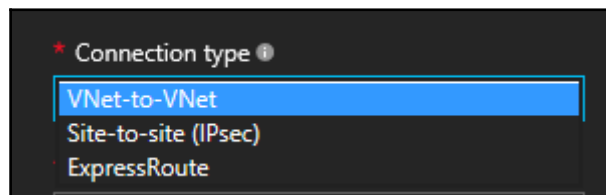
Azure Connections

Azure connections are the *wire* between the internal VNet gateway and your Azure VPN gateway or ExpressRoute circuit. With these connections, you can establish the tunnel through the Azure network and establish the connection to your on-premises environment or other VNets.

You can find the Azure connections by searching for **Connections**, as shown in the following screenshot :



An Azure connection offers the options shown in the following screenshot:

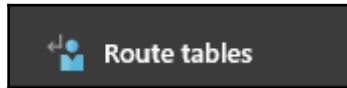


Later on in the chapter, we will use Azure connections to build up a connection between VPN and MPLS gateways.

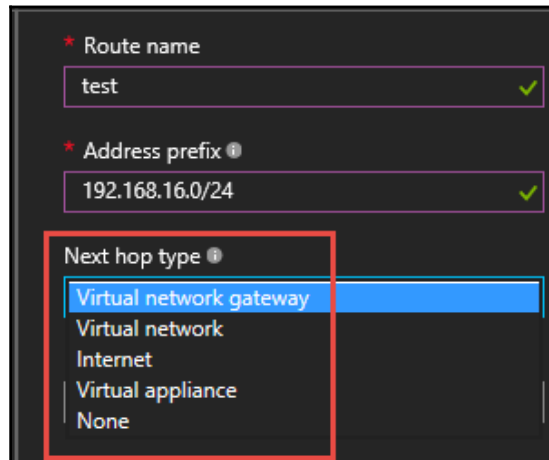
Azure Routes

With routes in Azure, you can change the default *Any to Any* routing within Azure to meet your needs.

The following screenshot shows the Azure service you need to look for when you want to implement an Azure route:



With routes, you can basically redirect traffic from one subnet to another location. The following screenshot shows the current offerings of that Azure service:



Within the setup part of this chapter, we will explain how to configure a custom route:

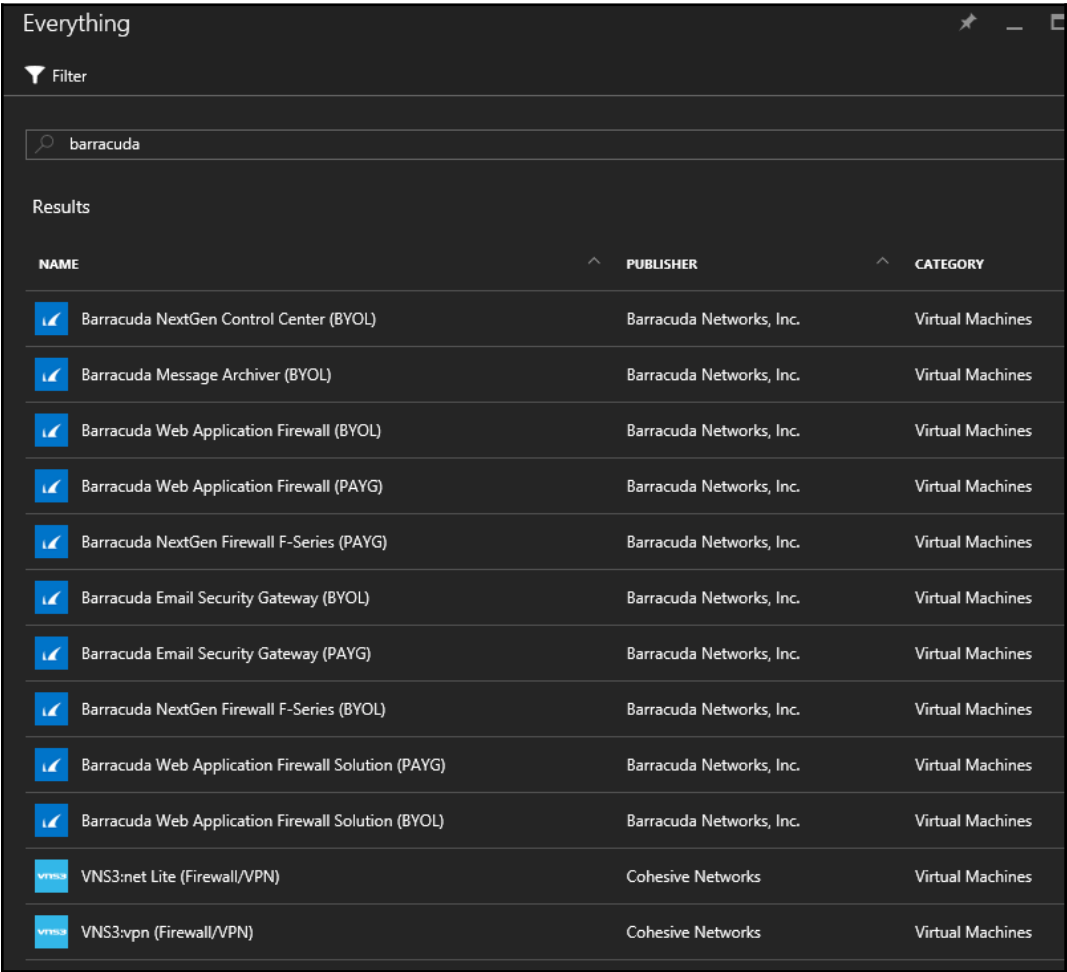
- **Virtual network gateway:** The traffic will be forwarded to another Azure gateway. This option can be used if you maybe want to send traffic via another gateway or route to its target. Or you have redirected all traffic to a Virtual Appliance and want specific traffic to bypass the appliance.
- **Virtual network:** Transfers traffic directly into another VNet. That could be used to transfer traffic from one VNet to another which can't be reached directly.
- **Internet:** Traffic will be send directly into the Internet.
- **Virtual appliance:** The traffic will be sent to a third-party virtual network device hosted in your Azure environment. That can be a Barracuda Next Generation Firewall or a Cisco Nexus device, for example.
- **None:** Traffic will be dropped and will not be routed.

Azure Third-Party Network Devices

Some vendors such as Cisco, Barracuda, or F5 offer VPN and network devices such as firewalls or load balancers as Azure virtual appliances via the Azure marketplace. Those devices can be directly integrated into your Azure infrastructure.

If you want to use one of these devices, you can look after them in the Azure marketplace and deploy them out like regular virtual machines.

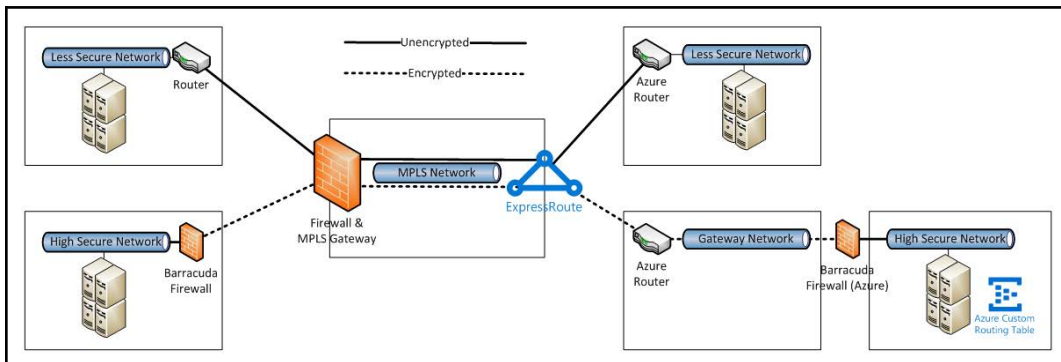
The following screenshot shows an example search for `barracuda`:



To integrate one of these devices into your environment, you need to implement Azure routes to pass traffic to the third-party devices, as shown in the next diagram.

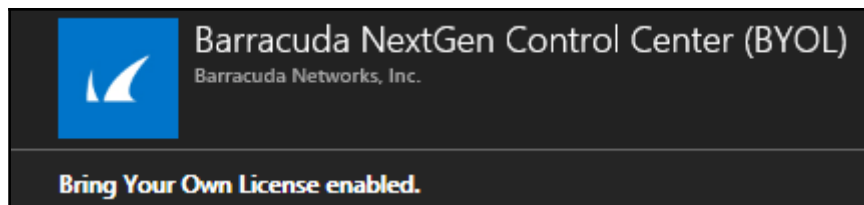
Normally there is no need to implement a third-party device because the Microsoft standard services offer approximately 80% of all services that are needed by most customers. Sometimes there are cases where you need to implement special systems such as a load balancer in Azure. Under certain circumstances, your target application has to use a load balancer feature that is not supported by the Azure load balancer.

The following diagram shows another case where you have the requirements for additional data encryption on a transfer level. In that case, you implement a VPN Tunnel or other encryption technology within an **ExpressRoute** and Azure VPN gateway. The following diagram is based on the Barracuda Next Generation Firewall design:



One thing you need to be aware of is that most of the third-party network solutions need an additional license and have some additional costs which might be not covered by you Microsoft Azure Subscription. So please read the introduction page of the product carefully.

Normally you should see information as shown in the following screenshot:



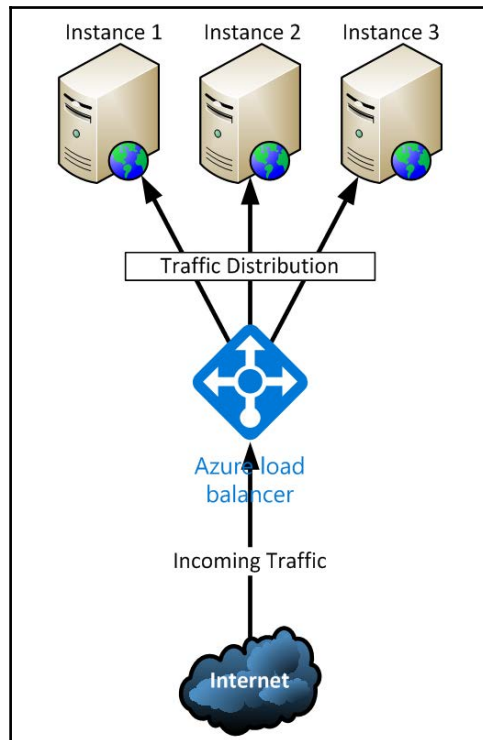
In the text, there should be additional information, as shown in the example screenshot :

- This Barracuda NextGen Control Center is licensed using the Bring-Your-Own-License (BYOL) model. Fill out the evaluation form to receive a 30-day evaluation license or purchase one of the licenses depending on your requirement.

Azure Load Balancer

The Azure load balancer is a layer 4 (TCP, UDP) load balancer which distributes incoming traffic among healthy instances of a service or among virtual machines in Azure. You can compare it with well-known load balancers such as Citrix Netscaler, Microsoft TMG or Windows server load balancer.

The Azure Resource Manager load balancer can distribute traffic that works with public IPs and Azure DNS entry. The following diagram shows the basic load balancing mechanism:



The feature list of an Azure load balancer is shown in the following sections.

Hash-Based Distribution

The load balancer uses a hash-based distribution algorithm. By default, it uses a 5-tuple hash to map traffic to available services and servers. It provides stickiness only within a transport session. Packets in the same TCP or UDP session will be directed to the same instance behind the load balancer. When the sender site closes, and reopens the session or starts a new session from the same source IP, the source port changes. This may result in a redirection of the traffic to a different data center and load balancer endpoint.

Port Forwarding

The load balancer gives you control over how inbound communication is managed. This communication can include traffic that's initiated from Internet hosts or virtual machines in other cloud services or virtual networks.

An input endpoint listens on a public port and forwards traffic to an internal port. You can map the same ports for an internal or external endpoint or use a different port for them.

You can use port forwarding to redirect traffic from an incoming port to another port your server listens to. For example, your endpoint listens to port 443 and you have a web application on the server listening to port 8443. You can configure the endpoint to redirect the traffic from port 80 to port 8443 on the server.

Automatic Reconfiguration

The load balancer instantly reconfigures itself when you scale instances up or down. That happens, for example, when you add or remove new servers or instances into the same load balancer set.

Service Monitoring

The load balancer can probe the health of the various server instances. When a probe fails to respond, the load balancer stops sending new connections to the unhealthy instances. Three types of probes are currently supported:

- **Guest agent probe (on PaaS VMs only):** The load balancer utilizes the Azure guest agent inside the virtual machine. The guest agent listens and responds with an HTTP 200 response only when the instance is ready and healthy. If the agent fails to respond with an HTTP 200 response, the load balancer marks the instance as unresponsive and stops sending traffic to that instance. The load balancer will continue to ping the instance until it responds again or the instance is removed from the load balancer set. Attention: If you are running a website, the code is typically running the process `w3wp.exe`. This processes are not monitored by the guest agent so the load balancer will never informed when the instance fails.
- **HTTP custom probe:** This probe overrides the default (guest agent) probe. You can use it to create your own custom logic for your application to determine the health of the role instance. As an example, the HTTP probe could login to a page and changes some values as a result of the logic. If the when fine without issues, the probe is good, otherwise it will generate an alert. The load balancer will regularly probe your endpoint (every 15 seconds, by default).
- **TCP custom probe:** This probe relies on successful TCP session establishment to a defined probe port. Source NAT

All outbound traffic to the Internet comes from your instances and services that go through **source NAT (SNAT)**. The VMs and Services use the same **virtual IP (VIP)** address as the incoming traffic. SNAT provides different benefits:

- It enables upgrade and disaster recovery of services, since the VIP can be dynamically mapped to another instance of the service.
- It reduces the access control list easier. ACLs expressed in terms of VIPs do not change as services scale up, down, or get redeployed.

The load balancer configuration supports full NAT for UDP. Full NAT is a type of NAT where the port allows inbound connections from any external host.



To learn more about the Azure network load balancer, you should visit the Microsoft MSDN source at <https://azure.microsoft.com/en-us/documentation/articles/load-balancer-overview/>.

Azure Application Gateways

The Azure application gateway is another form of load balancing in Azure. Application load balancing enables Azure Customers to create routing rules for network traffic based on HTTP protocols like for publishing Websites on the same IP Address.

Application gateways currently support layer-7 application delivery for the following application-based load balancing algorithms:

- HTTP load balancing
- Cookie-based session affinity
- **Secure Sockets Layer (SSL)** offload
- URL-based content routing
- Multi-site routing

Application gateways currently offer the sizes **small**, **medium**, and **large**.

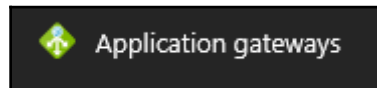
You can create up to 50 application gateways per subscription. Each application gateway can have up to 10 instances each, which makes up to 500 instances depending on the gateway site.

Please note that the small size is only for testing purpose and shouldn't be used in production.

Every gateway has a limited throughput performance. The following table shows the performance per gateway:

Back-end page response	Small	Medium	Large
6 K	7.5 Mbps	13 Mbps	50 Mbps
100 K	35 Mbps	100 Mbps	200 Mbps

The Azure application gateway is represented in the Azure portal with the symbol shown in the following screenshot:



We will not cover Azure application gateways further in this book, but to get more information, you can access the Azure application gateway documentation at <https://azure.microsoft.com/en-us/documentation/articles/application-gateway-introduction/>.

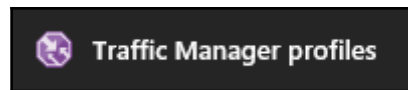
Azure Traffic Manager

Like the application gateway or the load balancer, the **Traffic Manager** is a mechanism to distribute incoming traffic among different Azure data centers. Unlike the load balancing of the other Azure balancers, the Traffic Manager works based on distribution via DNS entries, which means you deploy an DNS Name for the traffic manager. The clients connect directly to the endpoint for the application which has the best response time for his location. Traffic manager is mostly used as a frontend for Content Delivery Networks or Applications distributed over different Azure Regions. The following table summarizes the differences between all three load balancers:

Service	Azure load balancer	Application gateway	Traffic Manager
Technology	Transport level (OSI layer 4)	Application level (OSI layer 7)	DNS level

Application protocols supported	Any	HTTP and HTTPS	Any (An HTTP/S endpoint is required for endpoint monitoring)
Endpoints	Azure VMs and cloud services role instances	Any Azure internal IP address or public internet IP address	Azure VMs, cloud services, Azure web apps and external endpoints
VNet support	Can be used for both Internet facing and internal (VNet) applications	Can be used for both Internet facing and internal (VNet) applications	Only supports Internet-facing applications
Endpoint monitoring	Supported via probes	Supported via probes	Supported via HTTP/HTTPS GET request

The Azure Traffic Manager is symbolized with the following item in the Azure portal:



Azure DNS

With Azure DNS, you can host your DNS domains in Azure and manage your DNS records using the same credentials you use for other Azure services. So Azure DNS offers basically the same services as GoDaddy or UnitedDomains and other DNS providers.

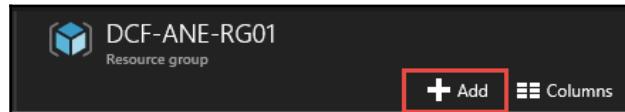
Setting up Azure Networks

Let us start deploying our Azure network infrastructure. We will start from the basics and then go up with different external and internal connections. All steps we do are also possible to do via PowerShell but we will stay with the Portal GUI within this guide.

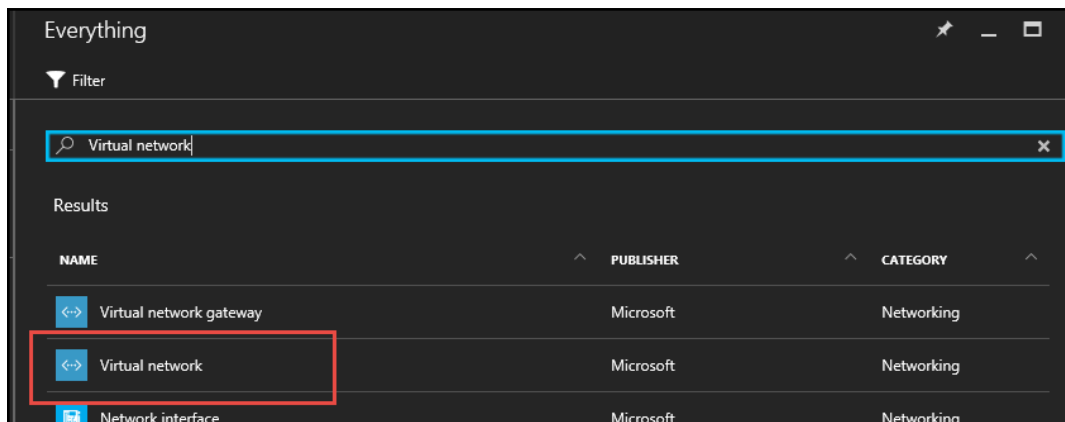
Setting up Azure VNet

The following are the steps to set up Azure VNet:

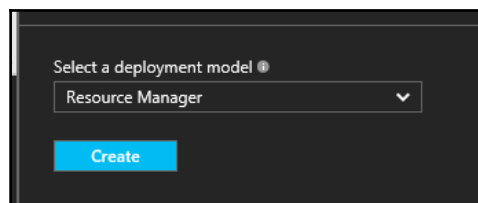
1. First of all, we navigate in our **Resource group** and use **Add** to open the Azure marketplace:



2. In the next step, we look for `Virtual network` within the Azure marketplace and click **Virtual network**:

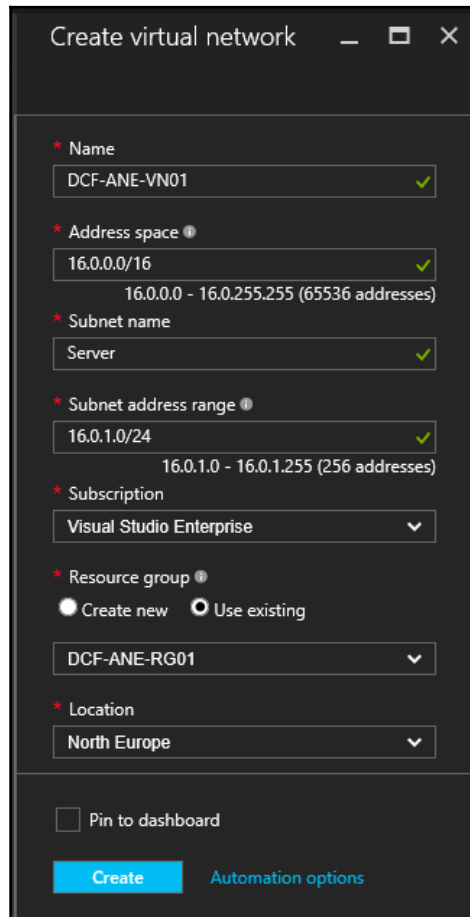


3. In the Azure blade afterwards, you need to decide between Resource Manager and Classic. You should only choose **Resource Manager**. The Classic is based on the old **Azure Service Manager (ASM)** environment and has certain limitation. Microsoft is currently migrating all services left in ASM. After the migration, Microsoft will somehow remove ASM and all resources deployed within it:



4. In the next interface, we need to configure the network details:
- **Name:** The name of your Azure network.
 - **Address space:** The IP address range you want to use within your Azure environment.
 - **Subnet name:** The name of your first subnet that could be either the gateway subnet or another one. We will stay with our server network for now and add the gateway network later.
 - **Subnet address range:** The IP range of the subnet you want to use.

The rest should be predefined by your resource group:



The screenshot shows the 'Create virtual network' form in the Azure portal. The form is titled 'Create virtual network' and has a dark theme. It contains several fields with red asterisks indicating required fields. The fields are: Name (DCF-ANE-VN01), Address space (16.0.0.0/16), Subnet name (Server), Subnet address range (16.0.1.0/24), Subscription (Visual Studio Enterprise), Resource group (DCF-ANE-RG01), and Location (North Europe). Each field has a green checkmark indicating it is valid. Below the Resource group field, there are radio buttons for 'Create new' (selected) and 'Use existing'. At the bottom, there is a checkbox for 'Pin to dashboard' and a blue 'Create' button. A link for 'Automation options' is also present.

Create virtual network

* Name
DCF-ANE-VN01 ✓

* Address space ⓘ
16.0.0.0/16 ✓
16.0.0.0 - 16.0.255.255 (65536 addresses)

* Subnet name
Server ✓

* Subnet address range ⓘ
16.0.1.0/24 ✓
16.0.1.0 - 16.0.1.255 (256 addresses)

* Subscription
Visual Studio Enterprise ▼

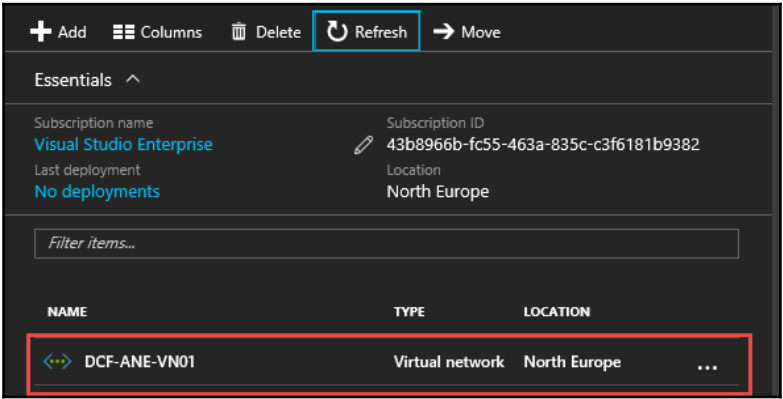
* Resource group ⓘ
☒ Create new ☐ Use existing
DCF-ANE-RG01 ▼

* Location
North Europe ▼

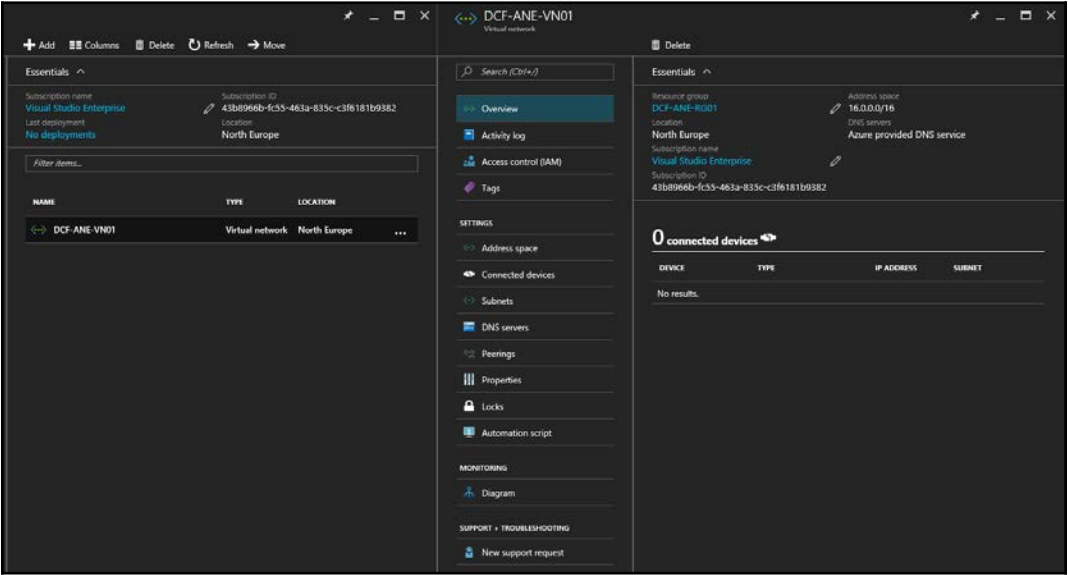
☐ Pin to dashboard

Create Automation options

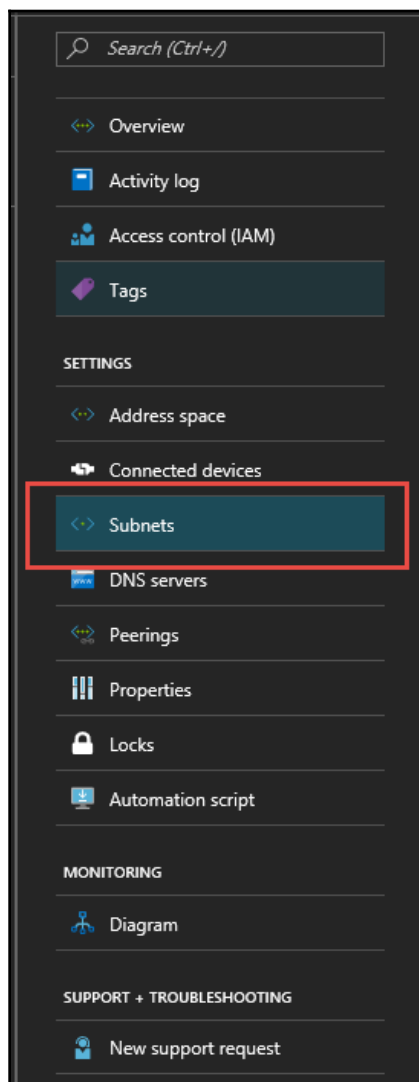
5. After the creation, the **Virtual Network (VNet)**, should be listed in your resource group. You maybe need to click on **Refresh**:



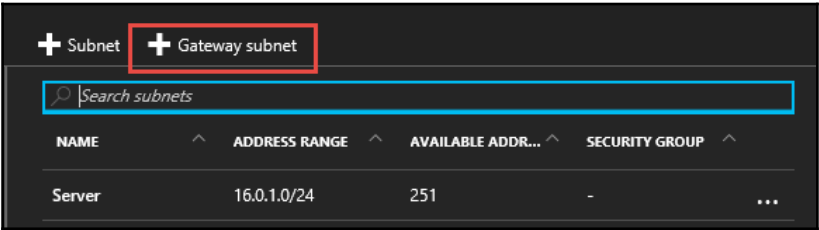
6. Now we will add our gateway subnet. Therefore, we select the VNet to open the settings:



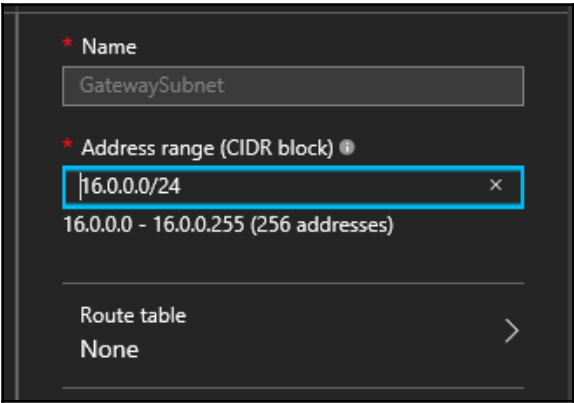
7. Yet we want to add the gateway subnet. Click on **Subnets** to open the subnet blade:



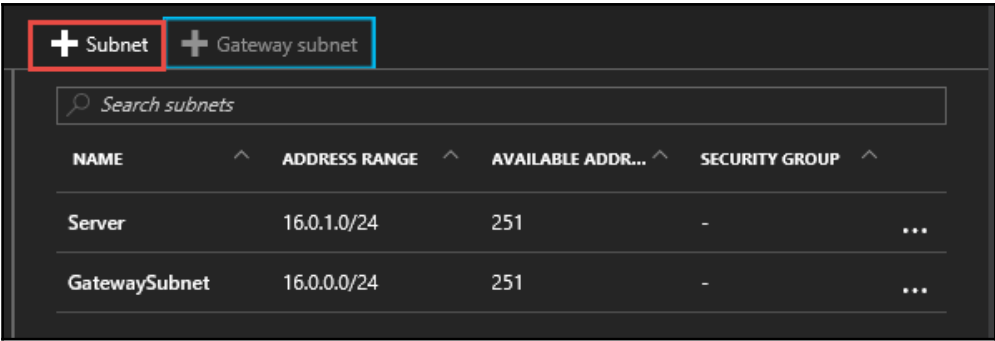
8. Afterwards, you select the + **Gateway** subnet button and another blade with the creation details will open:



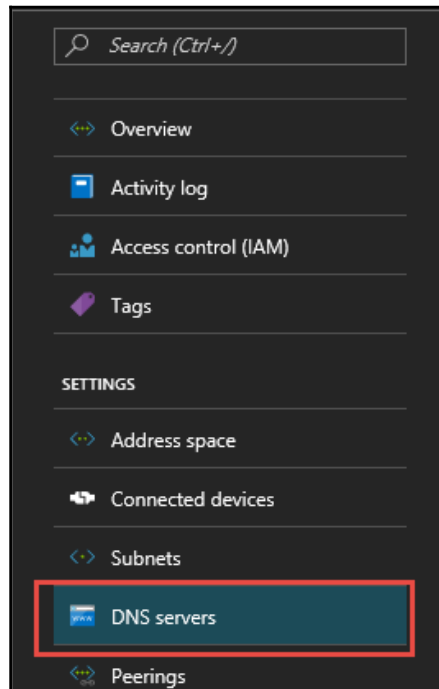
9. Within that blade, you need to define the subnet mask that the gateway uses. As explained earlier, we need a minimum of /29 CIDR addresses:



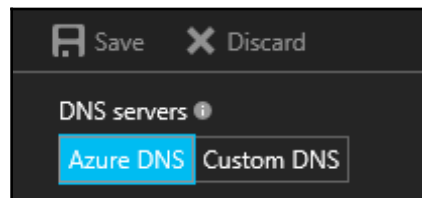
10. After clicking **OK**, the gateway subnet will be deployed in Azure. If you want to add more subnetworks, you can use the **Subnet** button:



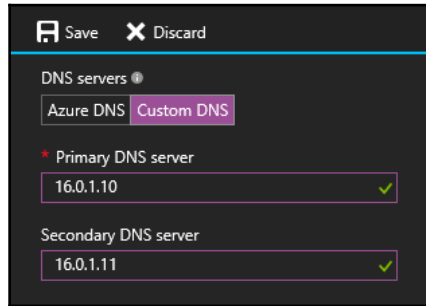
11. In the VNet settings, there is another option that could come in handy. Azure deploys every system with DHCP and by default configures Microsoft DNS servers as default DNS to the virtual machine. In most scenarios, you will need to change this setting to your own DNS server. Therefore, you can do this manually within the VM or change the default configuration within your Azure VNet. To do so, you need to select the **DNS servers** option in the **Settings**:



12. There you have a switch which changes between **Azure DNS** and **Custom DNS**:



13. We need to change it to **Custom DNS** and then we can add our DNS servers and **Save** the change:



14. Now every system within that VNet will use the **Custom DNS** server settings.

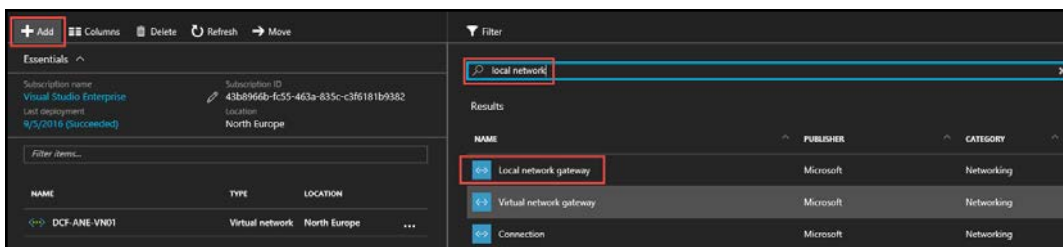
Setting up Azure Virtual Network Site-to-Site VPN

After we have deployed our network, we can now start to deploy our VPN gateway. At first, we deploy a site-to-site VPN to one of your sites. We will use the Azure portal to deploy the gateway and later update it via PowerShell.

Configuring Local Network Gateway

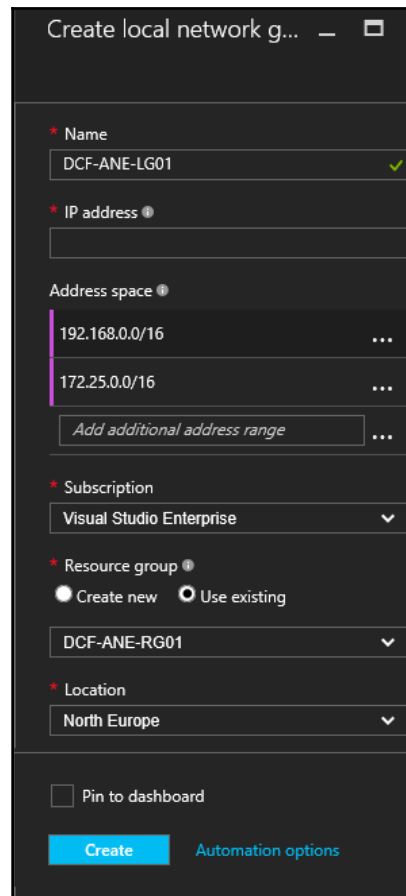
To set up a local network gateway, follow the given steps:

1. To configure the local network gateway follow the given steps: At first, we need to deploy the Local network gateway. Therefore, we click +Add in our resource group and search after local network in the Azure marketplace:



2. In the upcoming menu, you need to click on **Create**, and afterwards, the set up for the local gateway configuration will appear:

- **Name:** The name of your Azure local gateway
- **IP address:** The public IPv4 address of your local firewall or router device
- **Address space:** The IP ranges you use behind your local firewall and router device



The screenshot shows the 'Create local network gateway' form in the Azure portal. The form is titled 'Create local network g...' and has a dark theme. It contains the following fields and options:

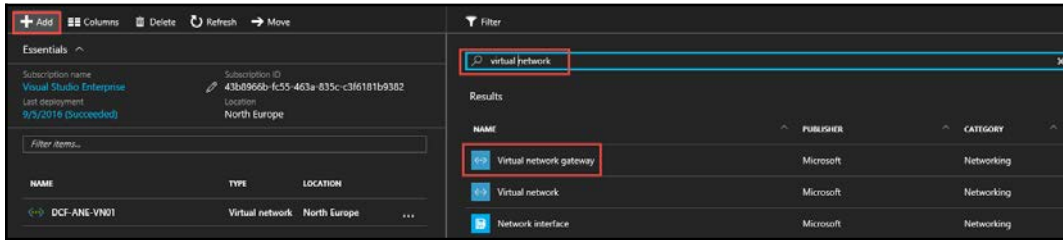
- Name:** A text input field with the value 'DCF-ANE-LG01' and a green checkmark icon.
- IP address:** A text input field with a help icon.
- Address space:** A section with a help icon containing a list of IP ranges: '192.168.0.0/16' and '172.25.0.0/16', each with a three-dot menu icon. Below the list is a text input field with the placeholder 'Add additional address range' and a three-dot menu icon.
- Subscription:** A dropdown menu with the value 'Visual Studio Enterprise'.
- Resource group:** A section with a help icon containing two radio buttons: 'Create new' (selected) and 'Use existing'. Below the radio buttons is a dropdown menu with the value 'DCF-ANE-RG01'.
- Location:** A dropdown menu with the value 'North Europe'.
- Pin to dashboard:** A checkbox that is currently unchecked.
- Create:** A blue button.
- Automation options:** A link.

3. That's all from for the local network gateway. Now we go on with the configuration of our VPN gateway in Azure.

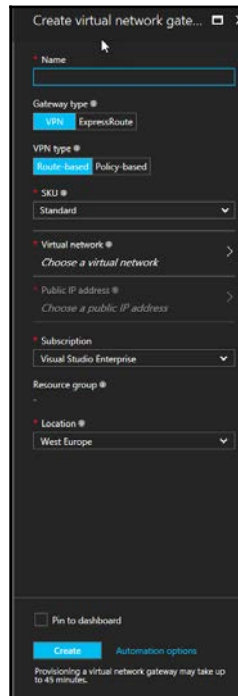
Configuring Azure Virtual Network Gateway

To configure the virtual network gateway, follow the given steps:

1. At first, we need to deploy the **Virtual network gateway**. Therefore, we click **Add** in our resource group and search after **virtual network** in the Azure marketplace:



2. After changing to the next blade, you need to configure your gateway. Therefore, you need to proceed as follows:

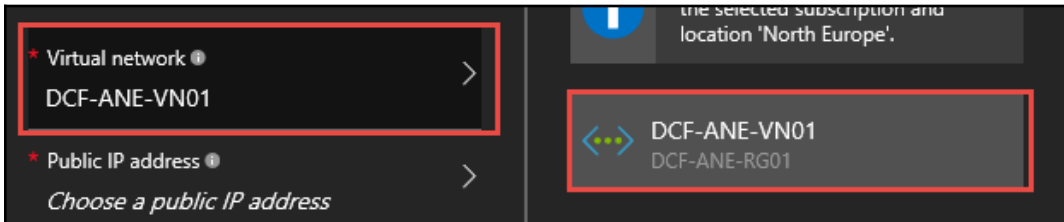


3. First, you need to give your gateway a **Name**:



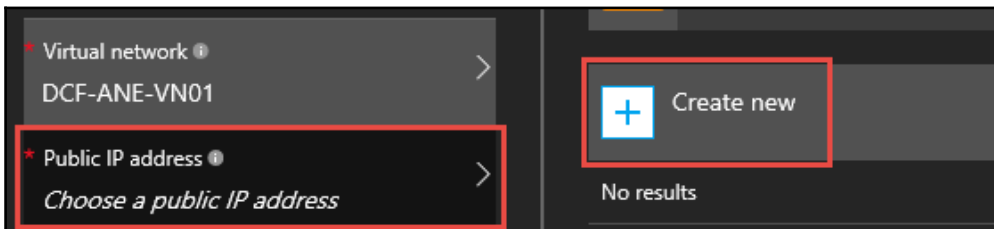
A screenshot of a form field labeled "Name" with a red asterisk. The text "DCF-ANE-GW01" is entered into the field, which is highlighted with a red rectangle.

4. Then extend the **Virtual network** blade and connect your gateway to the gateway network we previously created in the VNet:



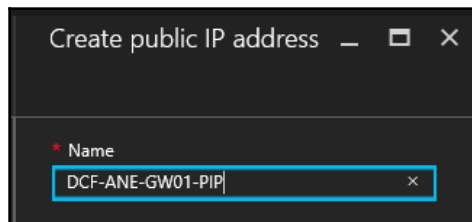
A screenshot of the Azure portal interface. On the left, the "Virtual network" blade is extended, showing "DCF-ANE-VN01" and "Public IP address" with a red rectangle. On the right, a card for "DCF-ANE-VN01" is shown, also with a red rectangle.

5. Now we need to create a public IP for our gateway. Extend the **Public IP address** blade and click on **Create new**:



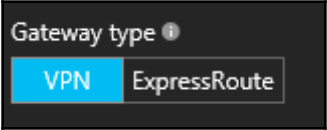
A screenshot of the Azure portal interface. On the left, the "Public IP address" blade is extended, showing "Choose a public IP address" with a red rectangle. On the right, a card with a blue plus icon and the text "Create new" is shown, also with a red rectangle.

6. Another blade opens and you need to give the new **Public IP address** a name for the resource and need to save with **OK**:

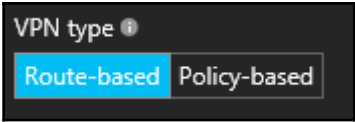


A screenshot of a dialog box titled "Create public IP address". It has a red asterisk next to the "Name" label. The text "DCF-ANE-GW01-PIP" is entered into the field, which is highlighted with a red rectangle.

7. After you created the public IP, you need to decide which gateway type you want to use. To deploy a site-to-site VPN based on IPSEC, you need to choose the gateway type **VPN**:



8. Now you need to decide if you want to use a **Route-based** (dynamic routing) or **Policy-based** (static routing) VPN:



9. Which type of VPN you can use is based on your on-premises firewall. The following table shows the configuration you need to do on your on-premises firewall. IKE phase 1 setup:

Property	Policy-based	Route-based and standard or high performance VPN gateway
IKE version	IKEv1	IKEv2
Diffie-Hellman group	Group 2 (1024 bit)	Group 2 (1024 bit)
Authentication method	Pre-shared Key	Pre-shared Key
Encryption algorithms	AES256 AES128 3DES	AES256 3DES
Hashing algorithm	SHA1(SHA128)	SHA1(SHA128), SHA2(SHA256)
Phase 1 Security Association (SA) lifetime (time)	28,800 seconds	10,800 seconds

10. IKE phase 2 setup:

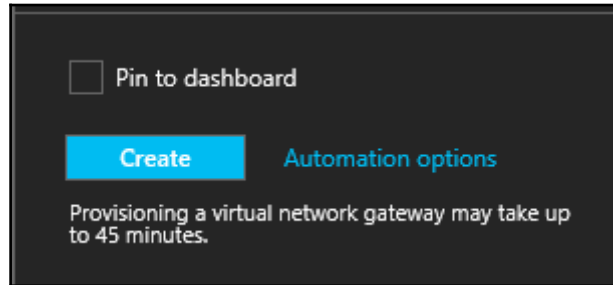
Property	Policy-based	Route-based and standard or high performance VPN gateway
IKE version	IKEv1	IKEv2
Hashing algorithm	SHA1(SHA128)	SHA1(SHA128)
Phase 2 SA lifetime (time)	3,600 seconds	3,600 seconds
Phase 2 SA lifetime (throughput)	102,400,000 KB	-
IPSEC SA encryption and authentication offers (in the order of preference)	1. ESP-AES256 2. ESP-AES128 3. ESP-3DES 4. N/A	See Route-based gateway IPSEC SA offers
Perfect forward secrecy (PFS)	No	No (*)
Dead Peer Detection	Not supported	Supported



Microsoft maintains a list of test and supported VPN devices which can be used by customers. You can find the list of devices and more information about the VPN setup at <https://azure.microsoft.com/en-us/documentation/articles/vpn-gateway-about-vpn-devices/>.

If you don't have any of these devices or you didn't want to use a Windows server as VPN gateway, there is also the option to use free firewall solutions such as pfSense. Bart Decker wrote a great blog about the topic. You can find the blog at <http://www.hybrid-cloudblog.com/pfsense-azure-hybrid-cloud/>.

11. To finish the setup, we click **Create**. Now it will take around 45 minutes until our gateway is deployed:



During the deployment process, Microsoft is deploying several virtual machines and VXLAN tunnels within their environment. Microsoft also integrates the new gateway and public interfaces into their security system, which includes honey potting and DDoS prevention for the gateways. This process takes some time.

12. After the deployment is finished, we have created an Azure virtual network gateway as with the SKU *basic*. If you want to upgrade the gateway to standard or performance, you only need to run following PowerShell script against your Azure environment:

```
Resize-AzureVNetGateway -GatewaySKU <gatewaysize>  
-VnetName <gatewayname>
```

13. PowerShell command example to resize to high performance gateway:

```
Resize-AzureVNetGateway -GatewaySKU HighPerformance  
-VnetName DCF-ANE-GW01
```

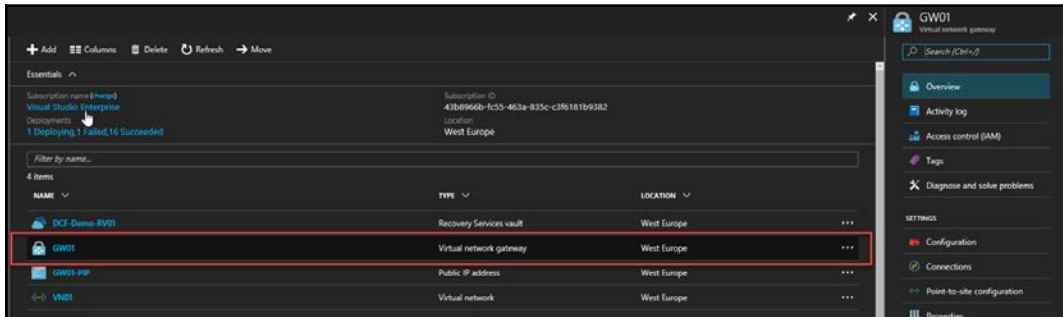
14. PowerShell command example to resize to standard gateway :

```
Resize-AzureVNetGateway -GatewaySKU Standard -VnetName  
DCF-ANE-GW01
```

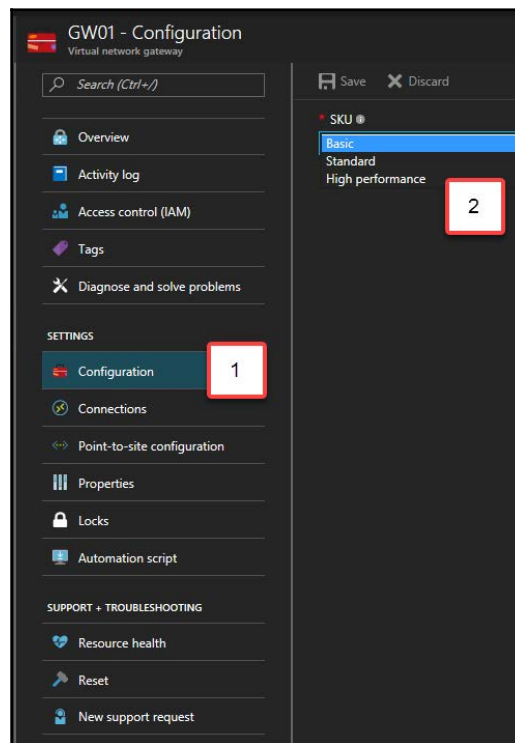
15. The same works also with downsizing a gateway:

```
Resize-AzureVNetGateway -GatewaySKU Basic -VnetName  
DCF-ANE-GW01
```


16. Besides the PowerShell way of resizing the gateway, Microsoft started to include the feature into the portal GUI. Therefore you need to navigate to the **Gateway** and open the detail blade:



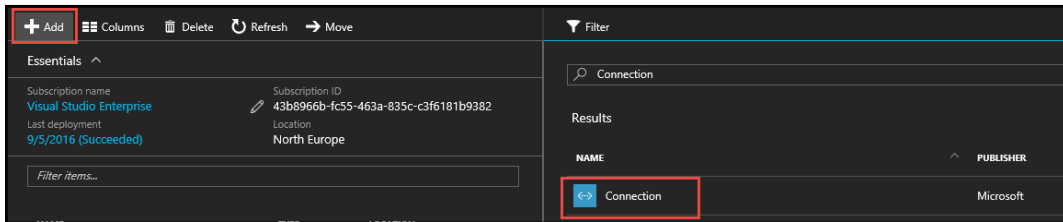
17. Within the detail blade you go to **Configuration** and change the **SKU**. Afterwards you need to save the new SKU. Please be aware that the change of the SKU will take again up to 45 minutes:



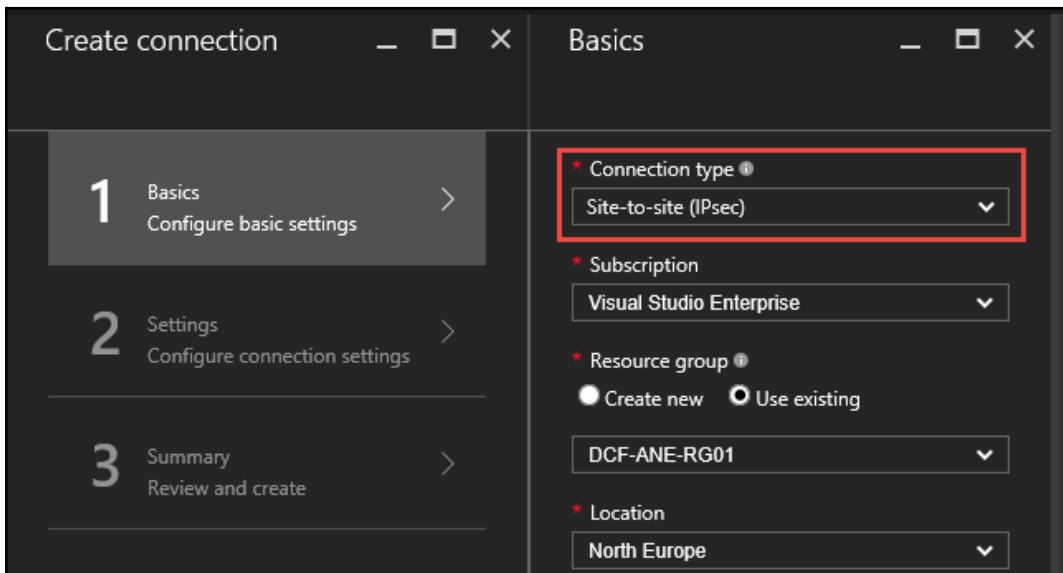
Configuring Connection Between Local and Virtual Network Gateways

Follow the steps below:

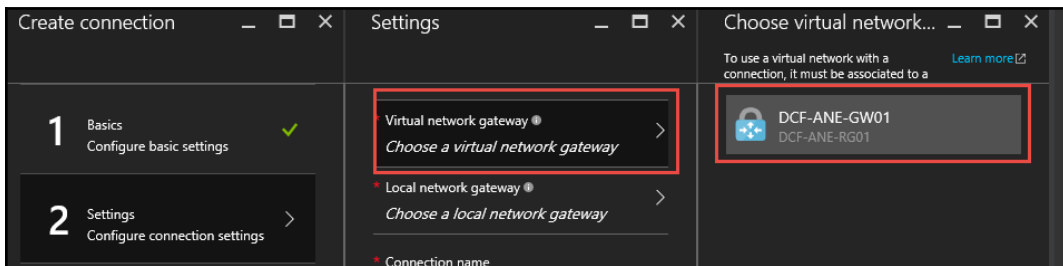
1. Now we need to establish a connection between gateways and enable the routing: Please go back to your Resource group and click Add again. Now we look for Connection in the marketplace. Then select Connection:



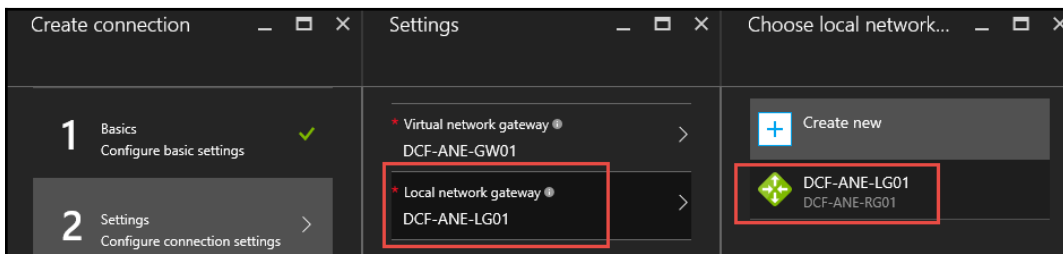
2. Now change the **Connection type** to **Site-to-site (IPsec)**:



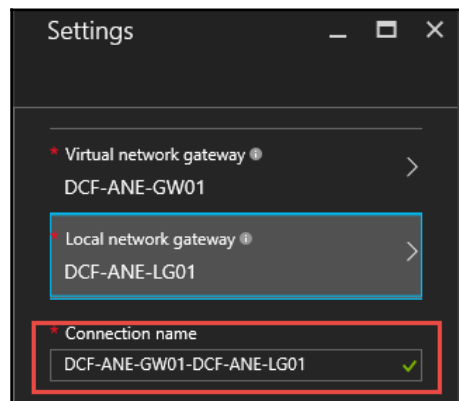
3. In the **Settings** phase, we first select the Azure **Virtual network gateway**:



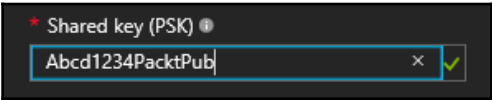
4. Afterwards, we select the **Local network gateway**:



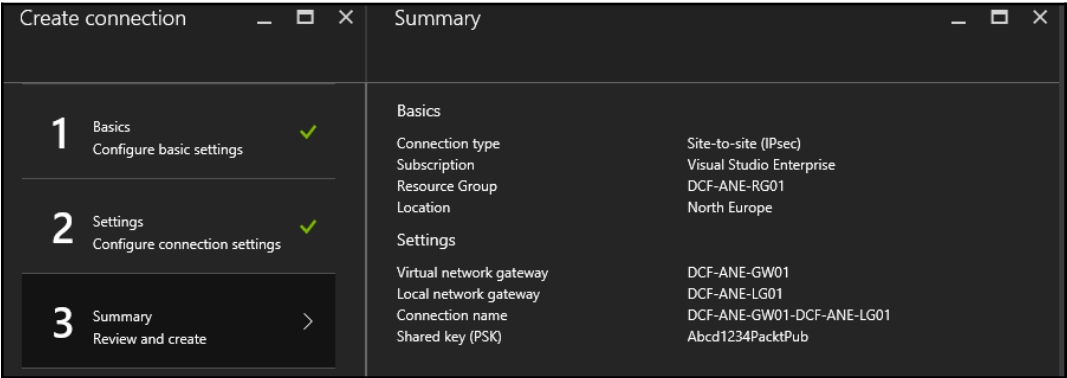
5. The **Connection name** will be created automatically but you can change it if necessary:



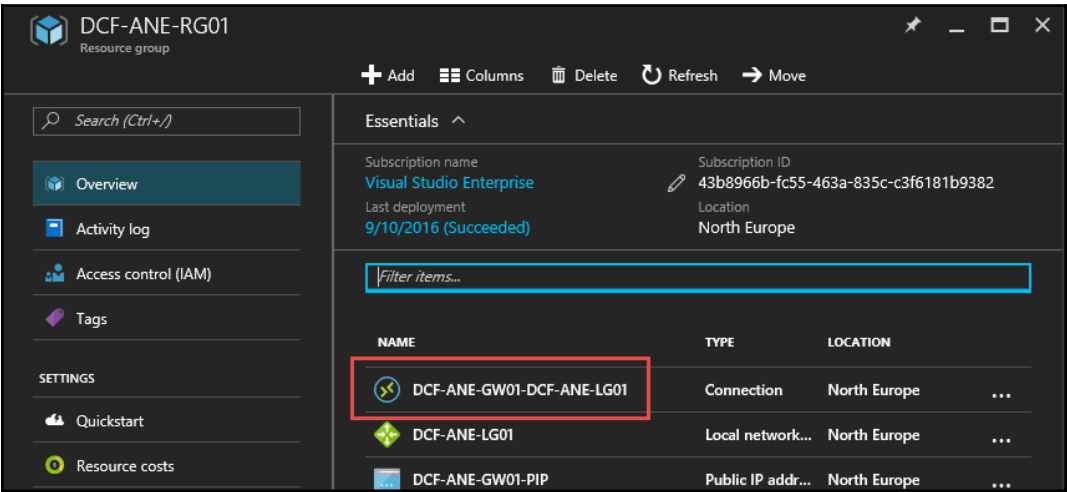
6. You then need to create a **pre-shared key (PSK)** secret for both gateways to share:



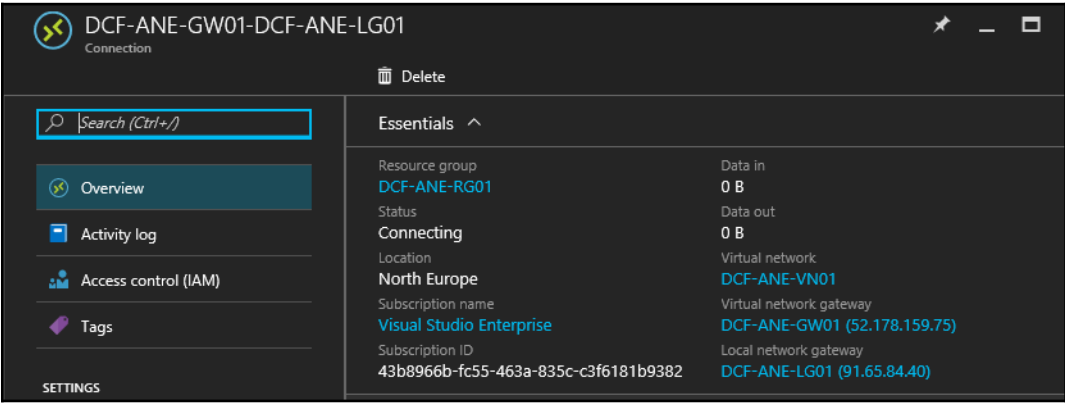
7. You can check the **Summary** and deploy the connection:



8. To check if the connection is deployed and working fine, you need to leverage the connection item in your **Resource group**:



9. There is an **Overview** within the detail blade of the connection. When the connection is successful the **Status** will change to **Successful**:



In most of the cases when the Status of the connection is not changing to connected, there are misconfiguration on the on premises Firewall or Network device. Mostly it's because there are different configurations of timeouts or encryption. For most of the VPN Devices you can find configuration guides on within the Azure documentation.
<https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-about-vpngateways>. If you still have issues, please contact the support of your VPN device manufacturer.

10. If you want to connect multiple sites to one Azure environment, you need to configure the Azure **Virtual network gateway** as a **Route-based VPN** and create a **Local network gateway** for each on-premises site. Then you create a connection for each site link. The following diagram illustrates the count of connections that need to be set up:

NAME		TYPE	LOCATION	
DCF-ANE-GW01-DCF-ANE-LG01	4	Connection	North Europe	...
DCF-ANE-GW01-DCF-ANE-LG02		Connection	North Europe	...
DCF-ANE-LG01	3	Local network...	North Europe	...
DCF-ANE-LG02		Local network...	North Europe	...
DCF-ANE-GW01-PIP	2	Public IP addr...	North Europe	...
DCF-ANE-GW01	1	Virtual network...	North Europe	...
DCF-ANE-VN01		Virtual network	North Europe	...

You need at least one route-based **Virtual network gateway** (1), one **Public IP address** for the gateway (2), two **Local network gateways** (3) and two **Connections** between **Virtual network** and **Local network gateway** (4).

11. Now you have deployed a site-to-site tunnel to your on-premises environment.



As of September 2016, Microsoft started to support active/active site-to-site VPN tunnels with high-performance virtual network gateway. Therefore, you need to configure two local network gateways: VNets and VPN gateways.

Setting up Azure Virtual Network with MPLS and ExpressRoute

For enterprise customers, a regular VPN connection may not be enough. Most of those customers will want to deploy an ExpressRoute connection. In the next part of the chapter, we will go through an ExpressRoute deployment.

First of all, to deploy ExpressRoute, you need some prerequisites. You need a contract with an ISP who connects your office to an MPLS network. That's a thing Microsoft cannot do for you at the moment.



The future goal of Microsoft and other Cloud Providers is, that you can deploy and order even ISP connections for your on premises location via Cloud Provider Portals and it's deployed on demand. In most countries that is some kind of science fiction because it requires a fully supported and over all available Software Defined WAN in-dependent from ISP infrastructures.

After you have signed the contract, you need to evaluate the peering location, the peering partner, and the bandwidth with your ISP. You will need this information during the ExpressRoute deployment.

As soon as you have this information, you can start with the deployment.

Configuring Azure Virtual Network Gateway

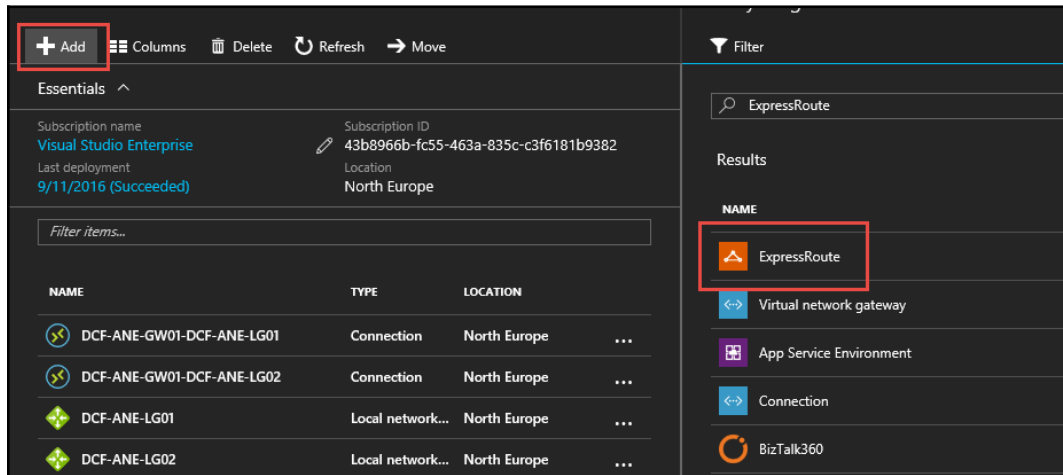
We need to configure an Azure virtual network gateway as an ExpressRoute gateway. The following screenshot shows an example:

A screenshot of the 'Create virtual network gateway' dialog box in the Azure Portal. The dialog has a dark theme and a title bar with a close button. It contains several fields and options for configuring a new gateway. The 'Name' field is empty with a cursor. The 'Gateway type' section has two buttons: 'VPN' and 'ExpressRoute', with 'ExpressRoute' selected. The 'SKU' dropdown is set to 'Standard'. The 'Virtual network' field shows 'Choose a virtual network' with a right arrow. The 'Public IP address' field shows 'Choose a public IP address' with a right arrow. The 'Subscription' dropdown is set to 'Visual Studio Enterprise'. The 'Resource group' field is empty. The 'Location' dropdown is set to 'West Europe'.

Configuring Azure ExpressRoute Circuit

To configure an ExpressRoute circuit, follow these steps:

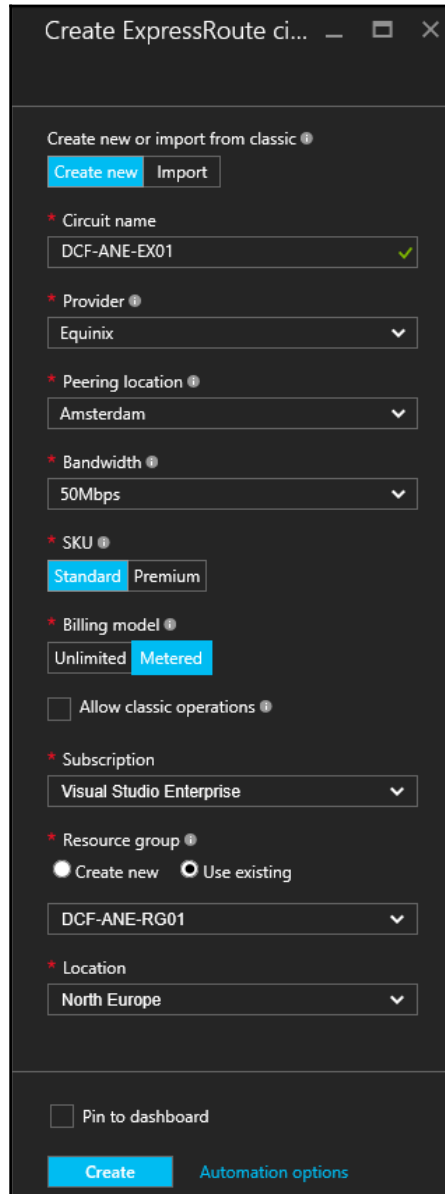
1. During the installation process of the gateway, you can proceed and install the ExpressRoute circuit. To do so, you need to go back to your Resource group and click **Add** again. From the marketplace, we select the ExpressRoute:



2. In the next blade, we set up our ExpressRoute circuit. Now we need the information your ISP gave you:

- **Circuit name:** The name of your ExpressRoute circuit.
- **Provider:** The provider you or your ISP uses for the peering with Microsoft Azure.
- **Peering location:** The edge gateway location your provider peers with Microsoft.
- **Bandwidth:** The bandwidth you ordered from your ISP.
- **SKU:** Select the service level for your ExpressRoute.
- **Billing model:** Select your billing. With **Metered** you will pay per download. With **Unlimited** you have a flat rate for your network traffic.

- **Allow classic operations:** Enables your Azure Service Manager deployment model environment to use the ExpressRoute too.



The screenshot shows the 'Create ExpressRoute circuit' form in the Azure portal. The form is titled 'Create ExpressRoute ci...' and has a close button. It contains the following fields and options:

- Create new or import from classic:** Two buttons, 'Create new' (highlighted in blue) and 'Import'.
- * Circuit name:** A text input field containing 'DCF-ANE-EX01' with a green checkmark.
- * Provider:** A dropdown menu with 'Equinix' selected.
- * Peering location:** A dropdown menu with 'Amsterdam' selected.
- * Bandwidth:** A dropdown menu with '50Mbps' selected.
- * SKU:** Two buttons, 'Standard' (highlighted in blue) and 'Premium'.
- * Billing model:** Two buttons, 'Unlimited' and 'Metered' (highlighted in blue).
- Allow classic operations:** A checkbox that is currently unchecked.
- * Subscription:** A dropdown menu with 'Visual Studio Enterprise' selected.
- * Resource group:** Radio buttons for 'Create new' (selected) and 'Use existing'. Below them is a dropdown menu with 'DCF-ANE-RG01' selected.
- * Location:** A dropdown menu with 'North Europe' selected.
- Pin to dashboard:** A checkbox that is currently unchecked.
- Create:** A blue button at the bottom left.
- Automation options:** A link at the bottom right.



Please be aware that the billing for your Azure ExpressRoute will start as soon as you click **Create**. To reduce unnecessary deployment costs, you should do that together with your service provider during a live activation session for both sites of the service.

- 3. After you have created the ExpressRoute circuit, you need to provide the **Service key** to your provider. The **Service key** will identify your Azure **Subscription** against its deployment and it can then create the connection to your environment.
- 4. You can find the **Service key** within the settings of the ExpressRoute circuit after it is completely deployed:

DCF-ANE-EX01
ExpressRoute circuit

Delete

Search (Ctrl+J)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

SETTINGS

Configuration

Connections

Peerings

Properties

Locks

Automation script

SUPPORT + TROUBLESHOOTING

New support request

Initiate the provisioning process with your service provider.

Essentials

Resource group

DCF-ANE-RG01

Circuit status

Enabled

Location

North Europe

Subscription name

Visual Studio Enterprise

Subscription ID

43b8966b-fc55-463a-835c-c3f6181b9382

Provider

Equinix

Provider status

Not provisioned

Peering location

Amsterdam

Bandwidth

50 Mbps

Service key

2c3b3488-2f70-48d6-a2c0-8a0d0e7db148

Peerings

TYPE	STATUS	PRIMARY SUBNET	SECONDARY SUBNET	
Azure private	Disabled	-	-	...
Azure public	Disabled	-	-	...
Microsoft	Disabled	-	-	...

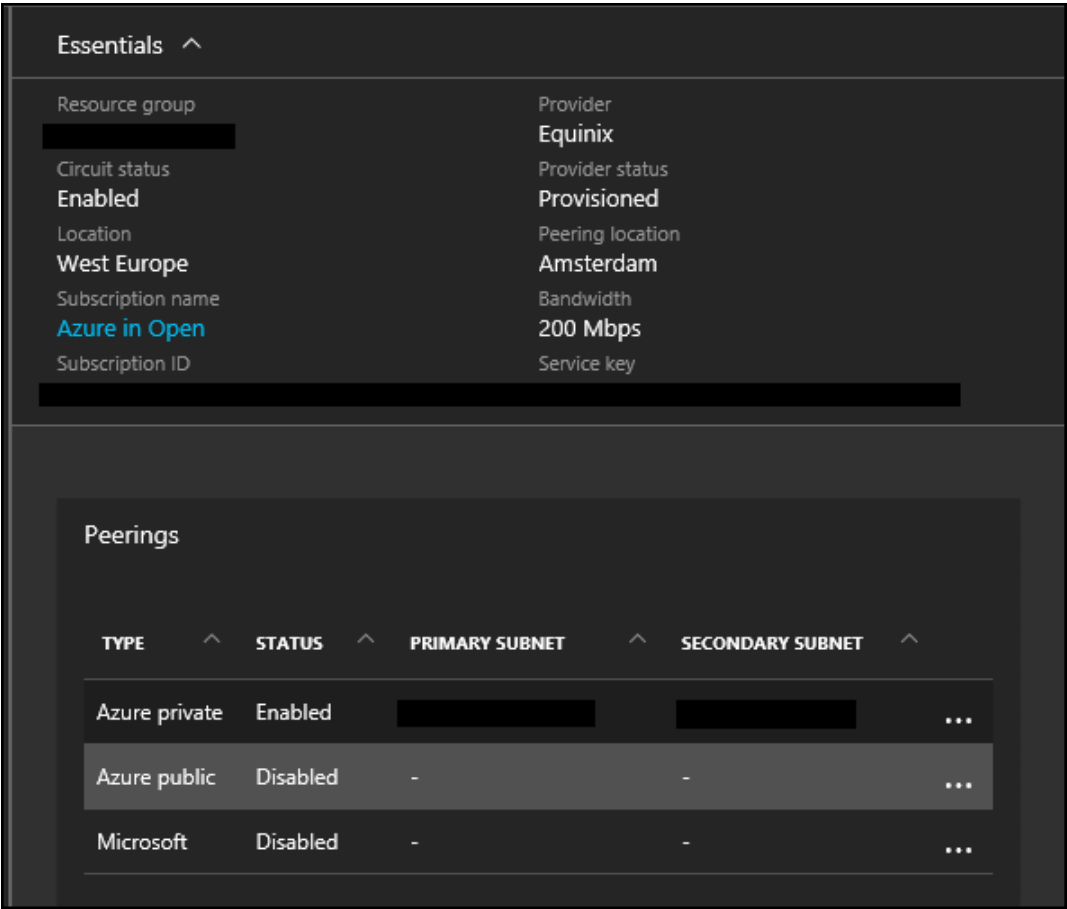
5. After the **Provider** status has changed to **Provisioned**, you can configure the Azure peering's:

Resource group	Provider
	Equinix
Circuit status	
Enabled	
Location	Peering location
West Europe	Amsterdam
Subscription name	Bandwidth
Azure in Open	200 Mbps
Subscription ID	Service key

6. To configure the peering's in Azure, you need additional information from your ISP. You need the **Peer ASN**, **Primary subnet**, **Secondary subnet**, **VLAN ID**, and for Microsoft peering, the **Advertised public prefixes**:

* Peer ASN ⓘ
0
* Primary subnet ⓘ
* Secondary subnet ⓘ
* VLAN ID ⓘ
0
* Advertised public prefixes ⓘ
Status: Not configured
Customer ASN ⓘ
0
Routing registry name ⓘ
None ▼
Shared key

7. To configure the peering, click on the peering type you want to configure:



8. Within the upcoming blade, you configure the information you've got from your ISP:

Private peering
WH-AWE-ER01

Save Discard Delete

* Peer ASN ⓘ
9999

* Primary subnet ⓘ
213.56.34.15/30

* Secondary subnet ⓘ
203.66.34.15/30

* VLAN ID ⓘ
99

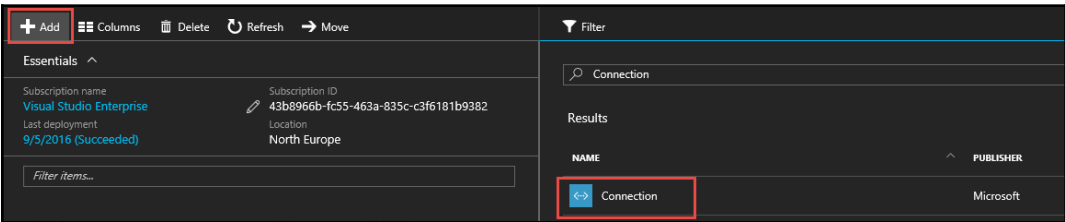
Shared key

9. In Azure you don't need to do additional routing. As soon as you establish the connection, Azure will directly configure the BGP settings.
10. After you have configured the peering's, you need to create a connection between the ExpressRoute circuit and the Azure virtual network gateway.

Configuring Connection Between ExpressRoute Circuit and Azure Virtual Gateway

Now we need to establish a connection between the gateway and the ExpressRoute circuit and enable the routing:

1. Please go back to your **Resource group** and click **Add** again. Now we look for **Connection** in the marketplace. Then select **Connection**:



2. In the upcoming blade, you need to configure the connection to be an ExpressRoute connection. Therefore, you need to select **ExpressRoute** in the **Configure basic settings** blade:

The screenshot shows the 'Create connection' blade with the 'Basics' tab selected. The left sidebar lists three steps: 1. Basics (Configure basic settings), 2. Settings (Configure connection settings), and 3. Summary (Review and create). The main area contains the following configuration fields:

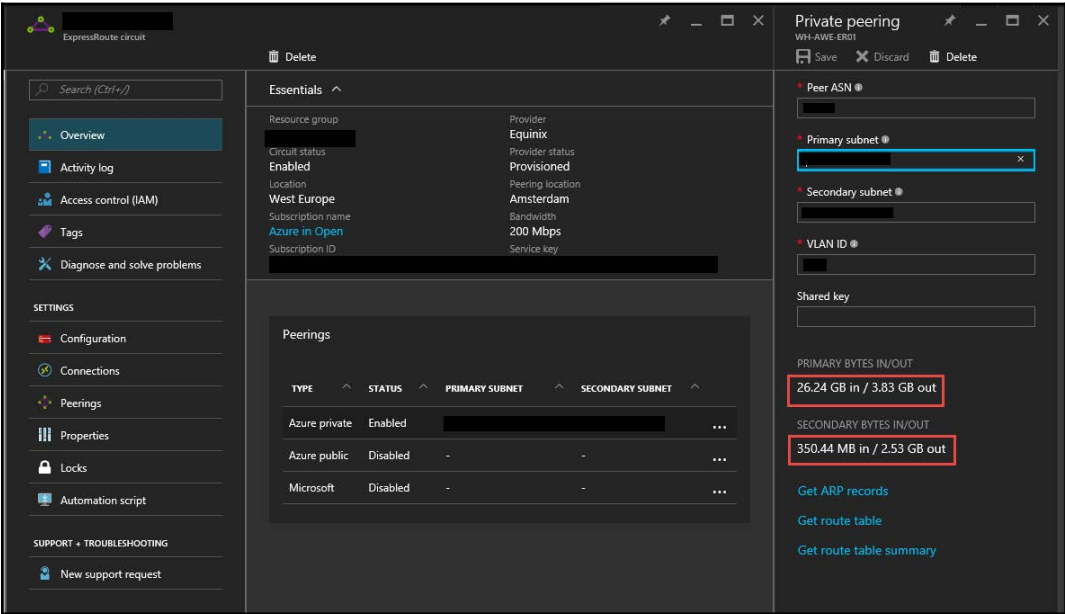
- Connection type**: ExpressRoute
- Subscription**: Visual Studio Enterprise
- Resource group**: DCF-ANE-RG01
- Location**: North Europe

3. Afterwards, you need to connect the **Virtual network gateway** and the **ExpressRoute circuit**. You can only select a circuit if private peering is enabled:

The screenshot shows the 'Create connection' blade with the 'Settings' tab selected. The left sidebar lists three steps: 1. Basics (Configure basic settings), 2. Settings (Configure connection settings), and 3. Summary (Review and create). The main area contains the following configuration fields:

- Virtual network gateway**: DCF-ANE-GW03
- ExpressRoute circuit**: Choose a circuit
- Connection name**: (empty text box)

4. After the review and creation of the gateway and all routes on the MPLS site are set, you should be able to reach your services. You can also check them when you open the peering configuration in the ExpressRoute **Settings** again. There you should see some bits of traffic listed in the **Summary**. The following screenshot shows an example:



Setting up Azure VNet Peering

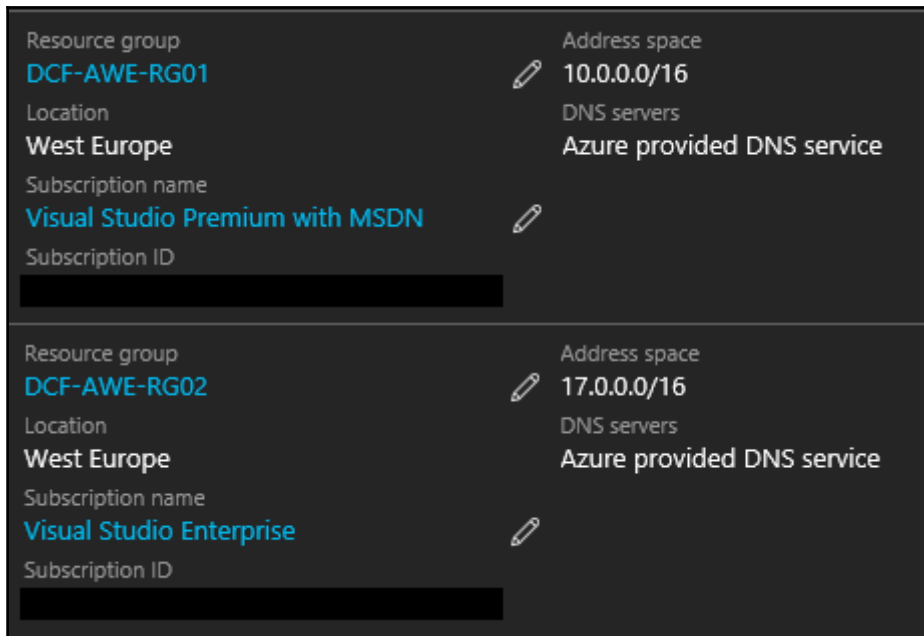
Now let's look at how you implement the new VNet peering. As already mentioned, you can use VNet peering to configure a connection between VNets in the same Azure region via the Azure backbone.

We will look at the configuration for VNet peering of VNets with different subscriptions. That's the most difficult scenario and you normally would use it to pair networks within different company subscriptions or with subscriptions from other companies.

Preparing the Deployment

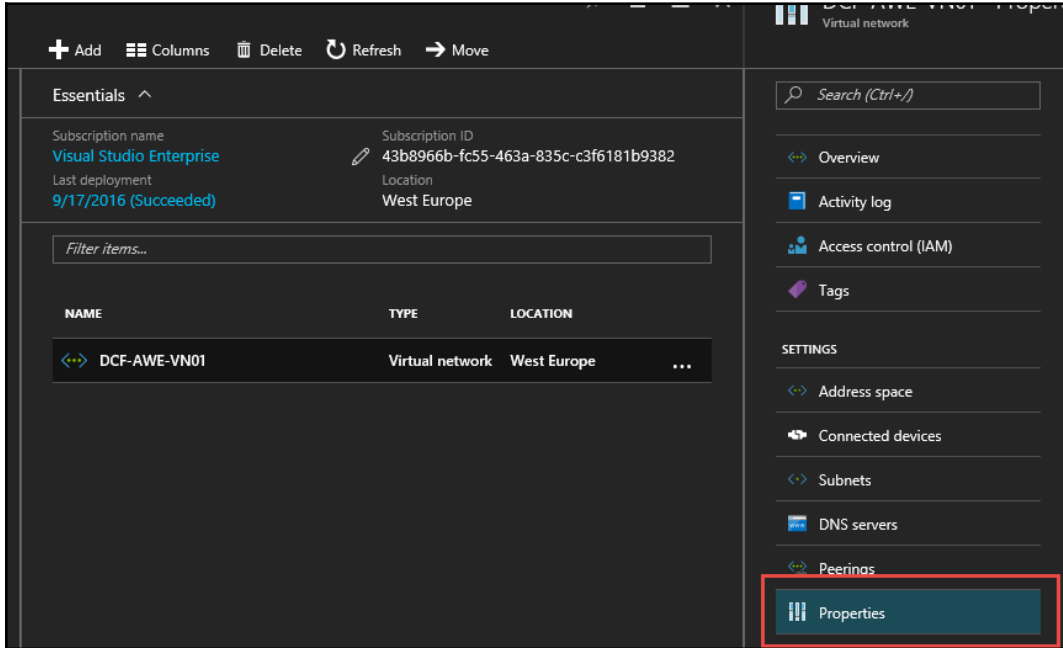
The following are the steps to prepare the deployment:

1. First, you need two subscriptions and both subscriptions need a VNet in the same Azure region. The following screenshot shows you an example:

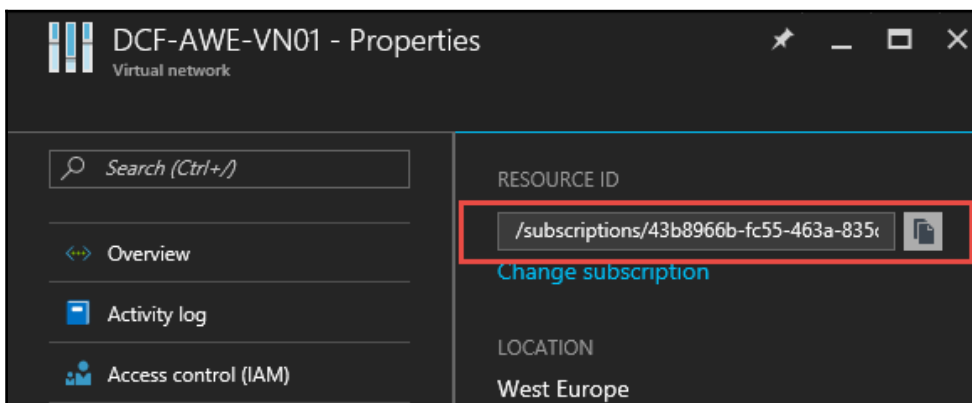


2. Both need to use different IP subnetwork address ranges. In my case, we use 10.0.0.0/16 for **subnet A** and 17.0.0.0/16 for **subnet B**.

- Before we start, we need to get the **Resource ID** of our partner VNet. You can find your resource IDs by navigating to the settings of your resource. You need to open the **Properties** to find the ID:



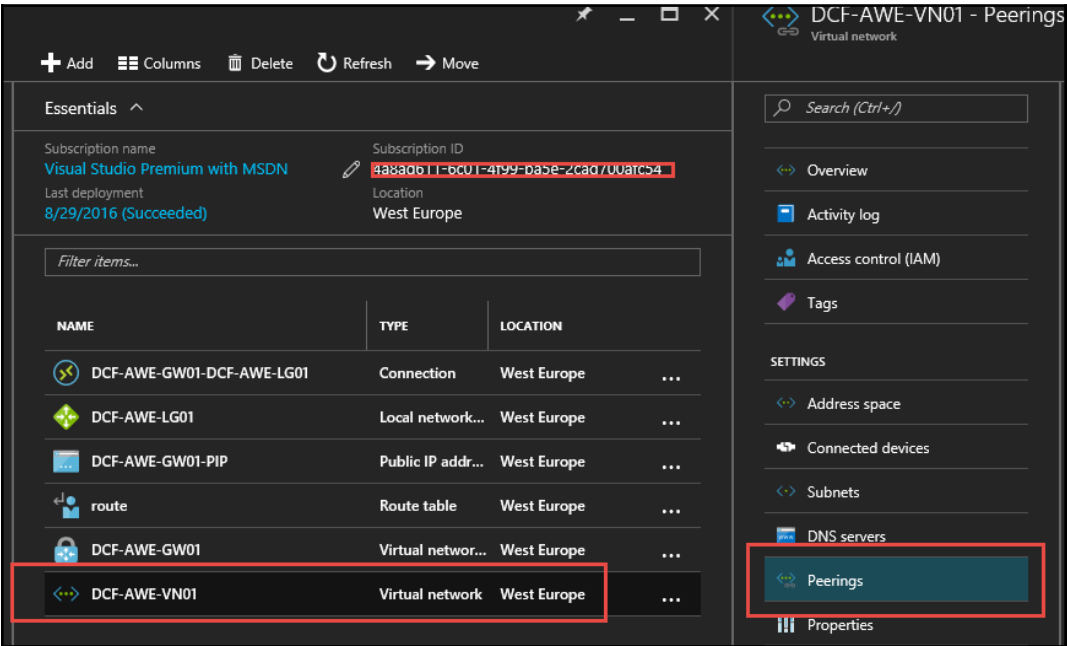
- In the **Properties** blade, you will see the ID in the upper-right corner:



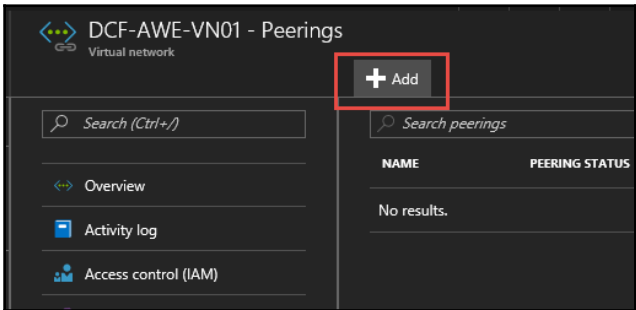
Configuring VNet Peering

For configuring the VNet peering, perform the following steps:

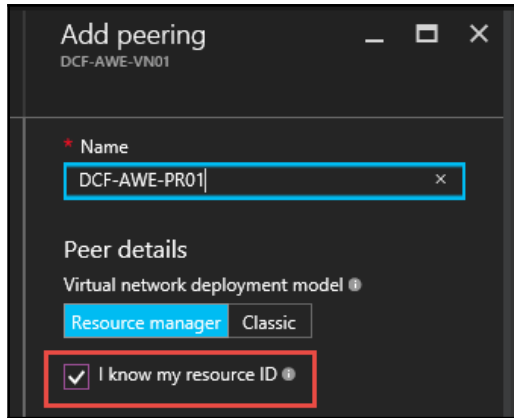
- 1. After the VNets in both subscriptions are prepared, navigate to one of the **Virtual network** and in the **SETTINGS** you need to click on **Peerings**:



- 2. In the **Peerings** blade, click on **Add**:



3. In the upcoming blade, you have the opportunity to create a new peering subnet in the existing **Subscription** and **Resource group**. In our current scenario, we need to select **I know my resource ID** to connect a VNet in another subscription:



Add peering
DCF-AWE-VN01

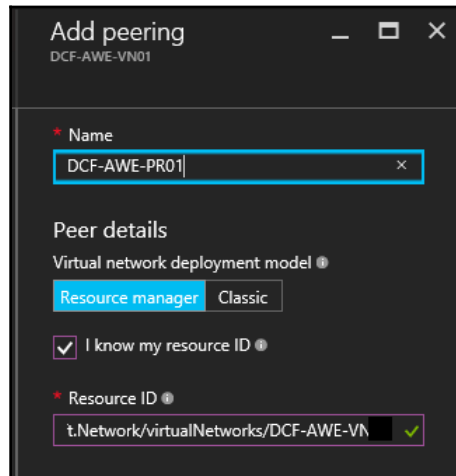
* Name
DCF-AWE-PR01

Peer details
Virtual network deployment model ⓘ

Resource manager Classic

☒ I know my resource ID ⓘ

4. Now you need the resource ID. You need to fill in the ID in the context field after the checkbox:



Add peering
DCF-AWE-VN01

* Name
DCF-AWE-PR01

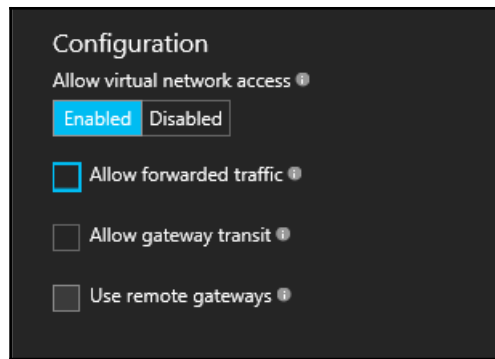
Peer details
Virtual network deployment model ⓘ

Resource manager Classic

☒ I know my resource ID ⓘ

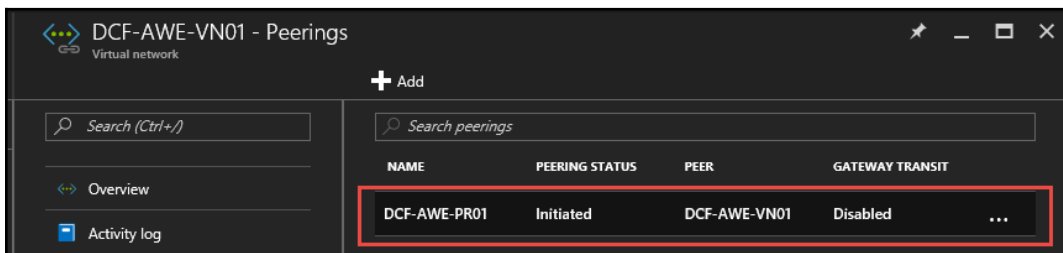
* Resource ID ⓘ
t.Network/virtualNetworks/DCF-AWE-VN ✓

5. Now you can do some more settings if you need them for your scenario:

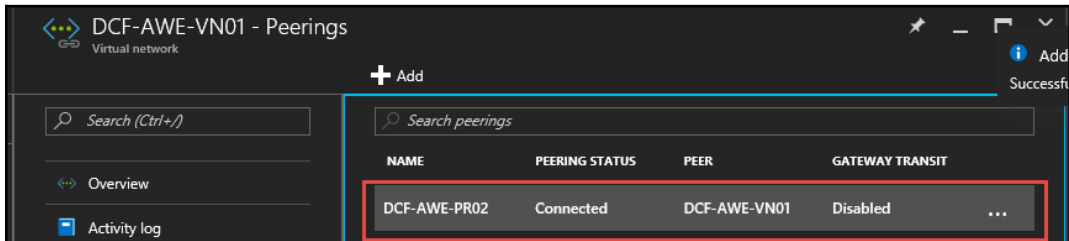


- **Allow virtual network access:** Allows the address space of the peer VNet to be included as part of the `Virtual_network`. In general the peered networks are linked to each other and become one big network.
- **Allow forwarded traffic:** Allows traffic not originated from the peered VNet to be accepted or dropped.
- **Allow gateway transit:** Allows the peer VNet to use your VNet gateway.
- **Use remote Gateways:** Uses your peer's VNet gateway. The peer VNet must have a gateway configured and a `AllowGatewayTransit` selected. You cannot use this option if you have a gateway configured.

6. After clicking **OK**, it will take a few minutes until the connection is established. After you have deployed the peering in the first subscription, you should see the status of the connection in the peering blade as **Initiated**:



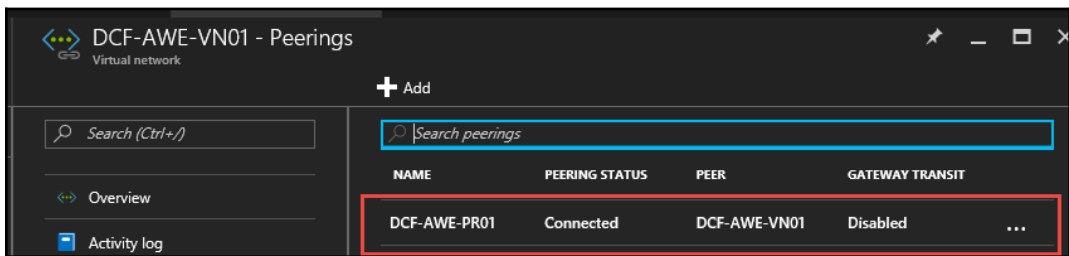
7. To set the **PEERING STATUS** of both networks as **Connected**, you need to repeat the steps mentioned previously in the other subscription too. Afterwards, you change the status in both subscriptions to **Connected** like shown in the following diagram for DCF-AWE-PR02:



The screenshot shows the Azure portal interface for 'DCF-AWE-VN01 - Peerings'. The left sidebar contains a search bar and links to 'Overview' and 'Activity log'. The main content area has a '+ Add' button and a search bar labeled 'Search peerings'. Below the search bar is a table with the following data:

NAME	PEERING STATUS	PEER	GATEWAY TRANSIT	
DCF-AWE-PR02	Connected	DCF-AWE-VN01	Disabled	...

8. The same should happen for DCF-AWE-PR01:



The screenshot shows the Azure portal interface for 'DCF-AWE-VN01 - Peerings'. The left sidebar contains a search bar and links to 'Overview' and 'Activity log'. The main content area has a '+ Add' button and a search bar labeled 'Search peerings'. Below the search bar is a table with the following data:

NAME	PEERING STATUS	PEER	GATEWAY TRANSIT	
DCF-AWE-PR01	Connected	DCF-AWE-VN01	Disabled	...

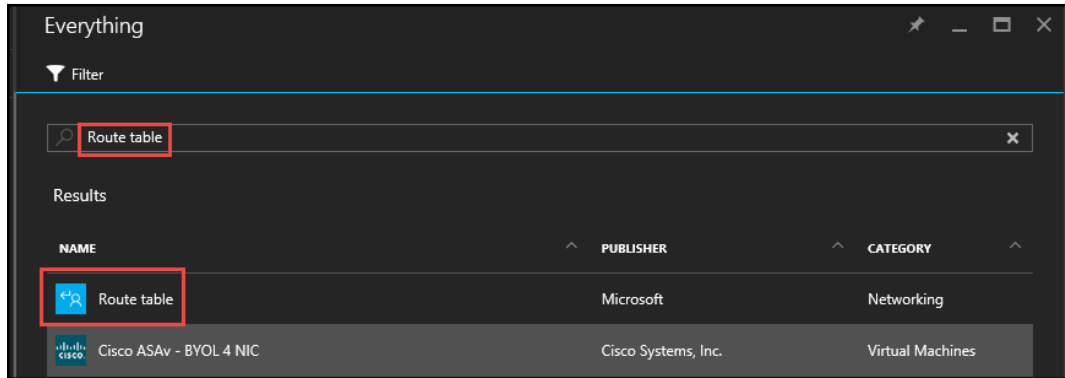


Currently, VNet peering is still in preview and under development. Microsoft will extend this service in the near future to support VNet peering with Microsoft Azure public services such as Storage or Azure SQL too.

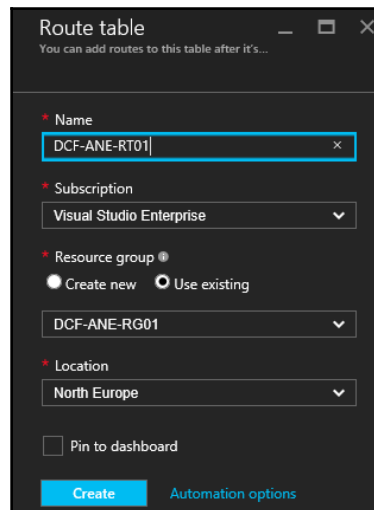
Configuring Custom Routes

As you already know, Azure by default routes every traffic to its virtual network gateways. Azure also routes any traffic in any direction. If you want to change those default behaviors, you need to create custom routes:

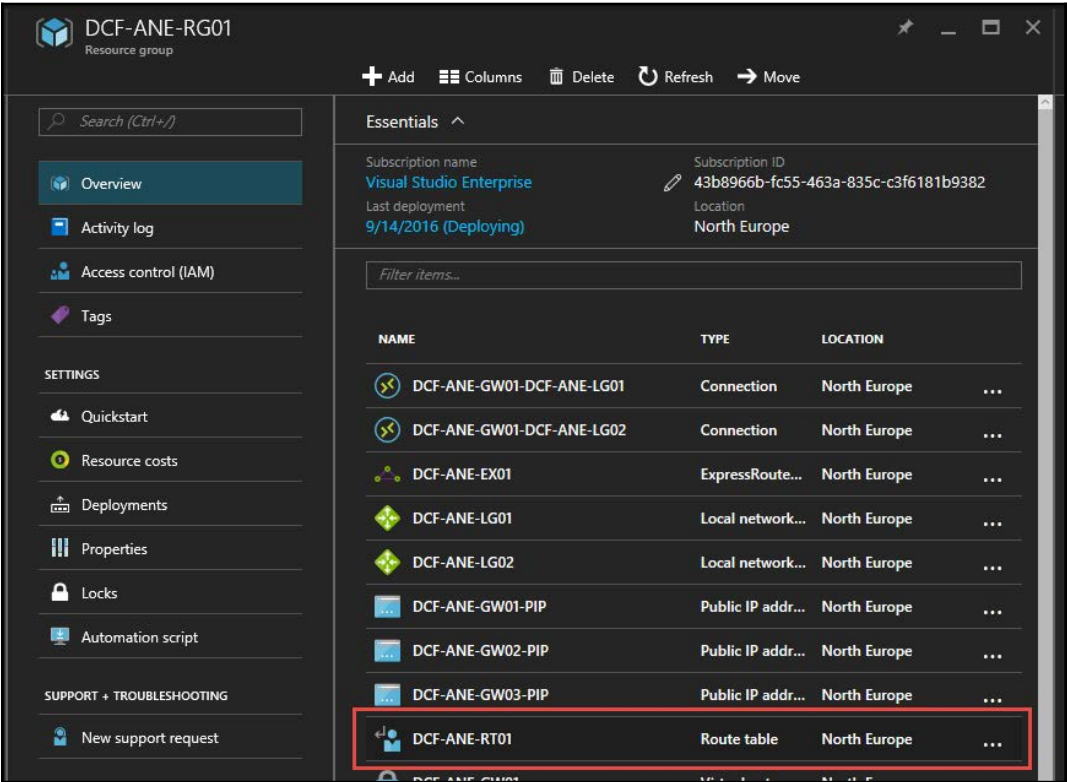
1. First you need to look for the `Route table` within the Azure marketplace:



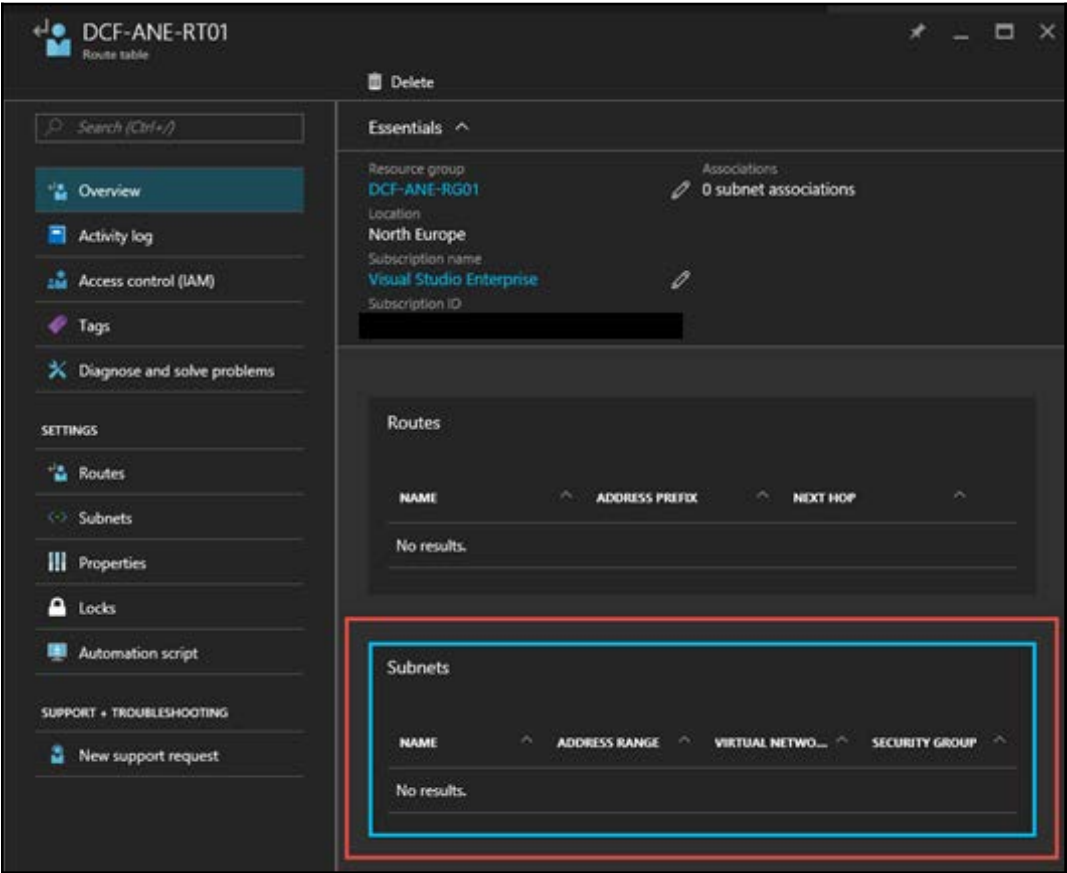
2. The only option in the enrolment process to give your route table a name. The rest will be done through **Route table** settings in the **Resource group**:



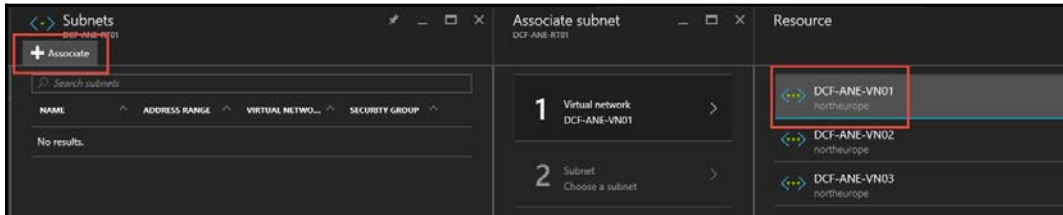
3. After you created the **Route table**, you need to go back to your **Resource** group and select the route table you created:



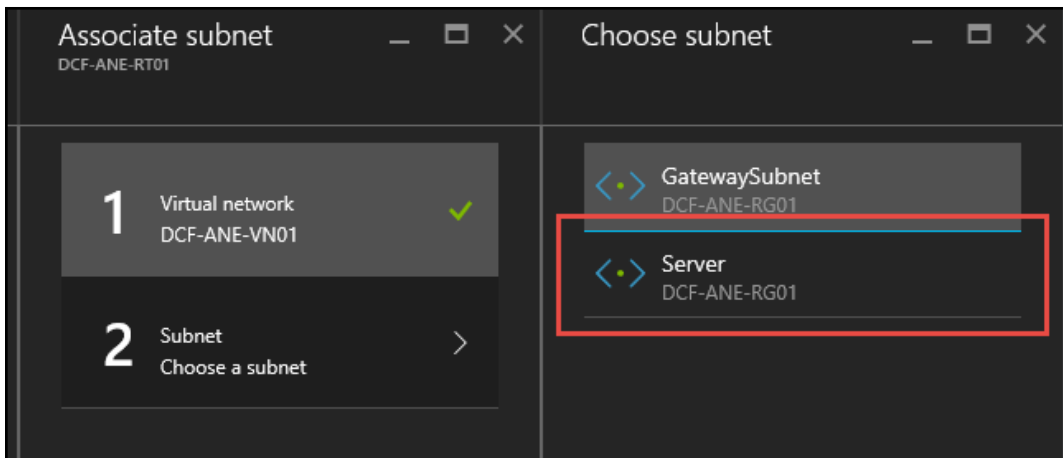
4. The **Settings** blade opens. Here you click first on **Subnets** to open the detail blade to associate subnets to that routing table:



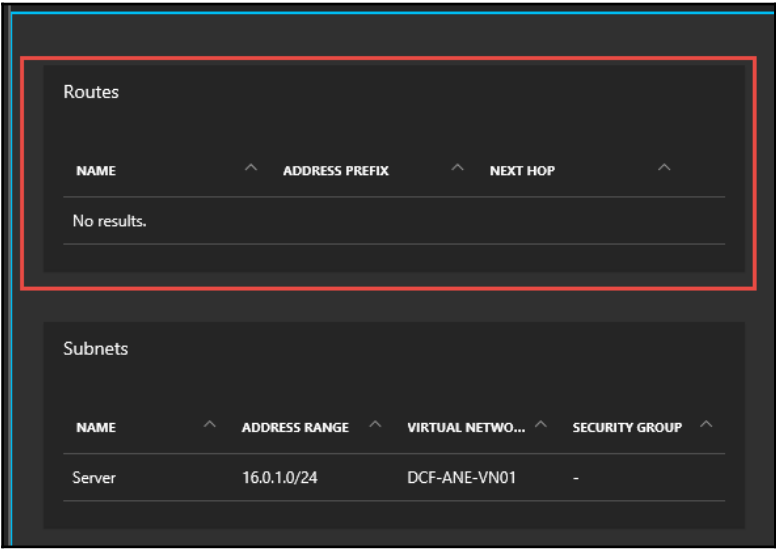
5. In the details blade, you click on **Associate** and add the Azure VNet where the route table should be applied to:



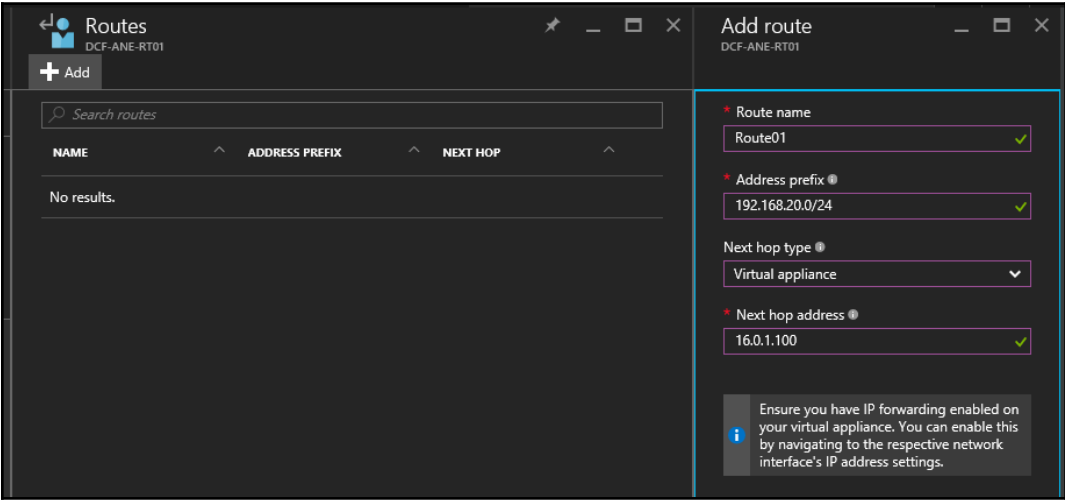
6. Then choose the subnet where you want to apply the table to and click **OK** to commit:



7. As soon as the subnet association is created, you need to configure the route. Click on **Routes** to open the detail blade:



8. In the detail blade, click **Add** to configure the new route. In the upcoming blade, you select a name for your router, the **Address prefix**, the **Next Hop type**, and the address of the next hop:

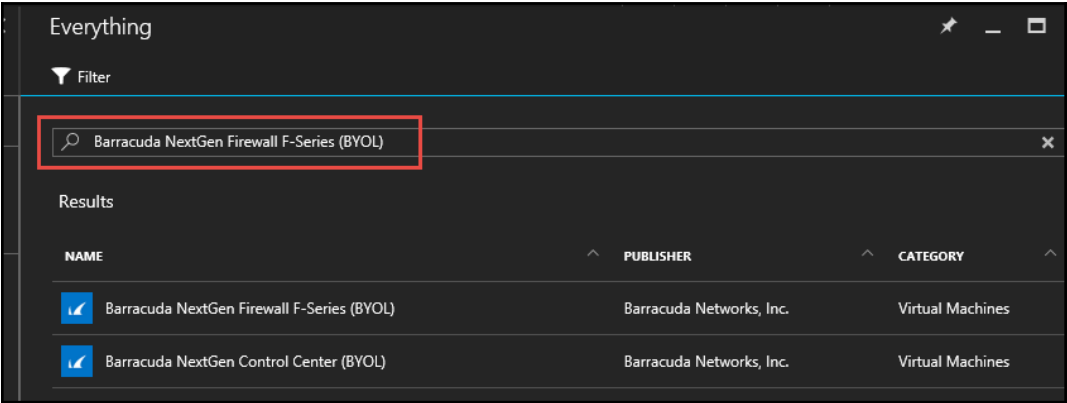


9. It will take a moment until the route will be applied to the subnet.

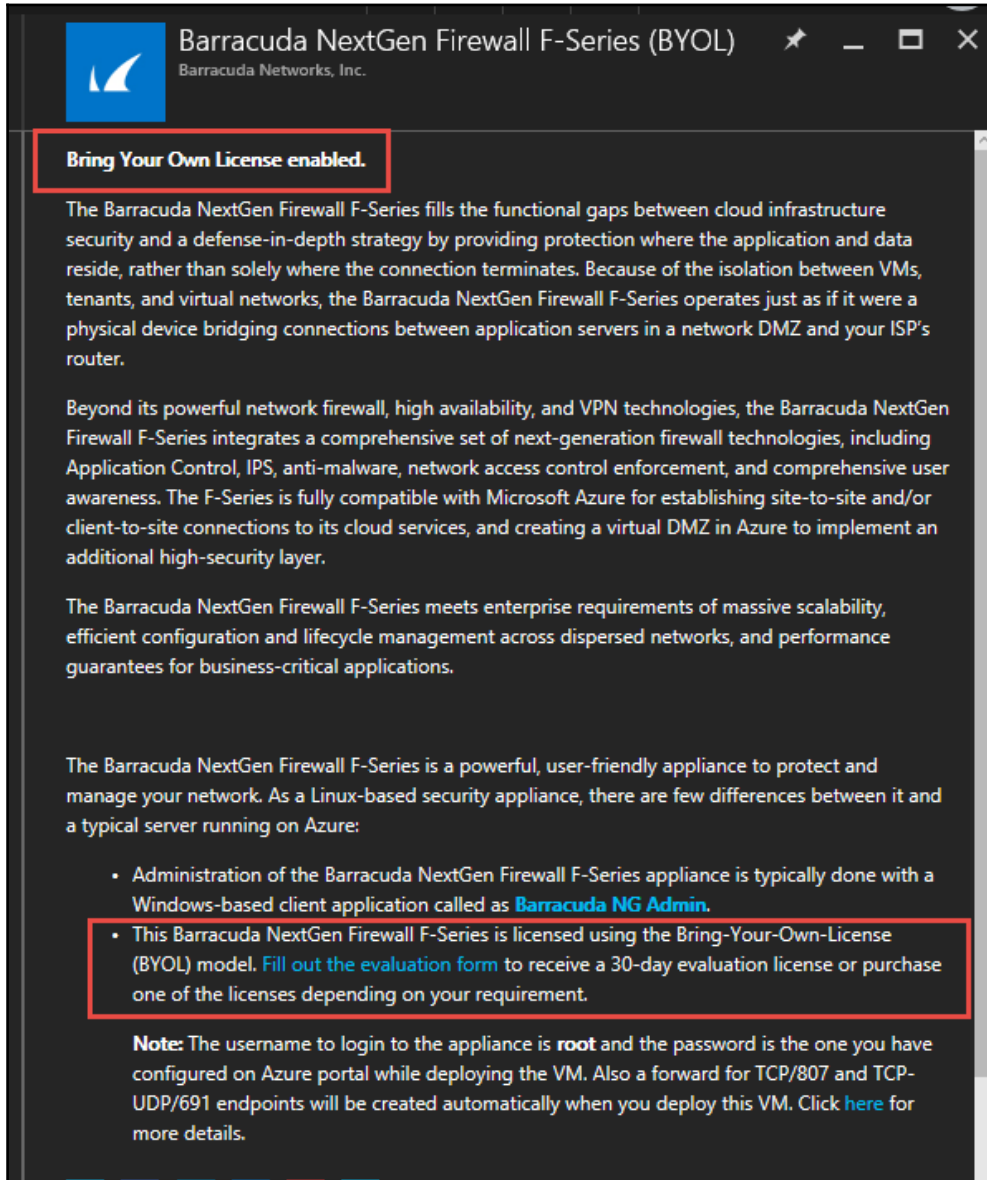
Configuring Third-Party Network Appliances

To deploy these machines, you need third-party networking appliances are Azure VMs running the software of the network vendor.

1. Then you need to look for the device you want to deploy:



2. Please be aware that for most of these devices, the license is not covered by Azure. So you need to buy an additional license:



Bring Your Own License enabled.

The Barracuda NextGen Firewall F-Series fills the functional gaps between cloud infrastructure security and a defense-in-depth strategy by providing protection where the application and data reside, rather than solely where the connection terminates. Because of the isolation between VMs, tenants, and virtual networks, the Barracuda NextGen Firewall F-Series operates just as if it were a physical device bridging connections between application servers in a network DMZ and your ISP's router.

Beyond its powerful network firewall, high availability, and VPN technologies, the Barracuda NextGen Firewall F-Series integrates a comprehensive set of next-generation firewall technologies, including Application Control, IPS, anti-malware, network access control enforcement, and comprehensive user awareness. The F-Series is fully compatible with Microsoft Azure for establishing site-to-site and/or client-to-site connections to its cloud services, and creating a virtual DMZ in Azure to implement an additional high-security layer.

The Barracuda NextGen Firewall F-Series meets enterprise requirements of massive scalability, efficient configuration and lifecycle management across dispersed networks, and performance guarantees for business-critical applications.

The Barracuda NextGen Firewall F-Series is a powerful, user-friendly appliance to protect and manage your network. As a Linux-based security appliance, there are few differences between it and a typical server running on Azure:

- Administration of the Barracuda NextGen Firewall F-Series appliance is typically done with a Windows-based client application called as **Barracuda NG Admin**.
- This Barracuda NextGen Firewall F-Series is licensed using the Bring-Your-Own-License (BYOL) model. [Fill out the evaluation form](#) to receive a 30-day evaluation license or purchase one of the licenses depending on your requirement.

Note: The username to login to the appliance is **root** and the password is the one you have configured on Azure portal while deploying the VM. Also a forward for TCP/807 and TCP-UDP/691 endpoints will be created automatically when you deploy this VM. Click [here](#) for more details.

3. In the deployment blade, you go through the different options for deploying a virtual machine.



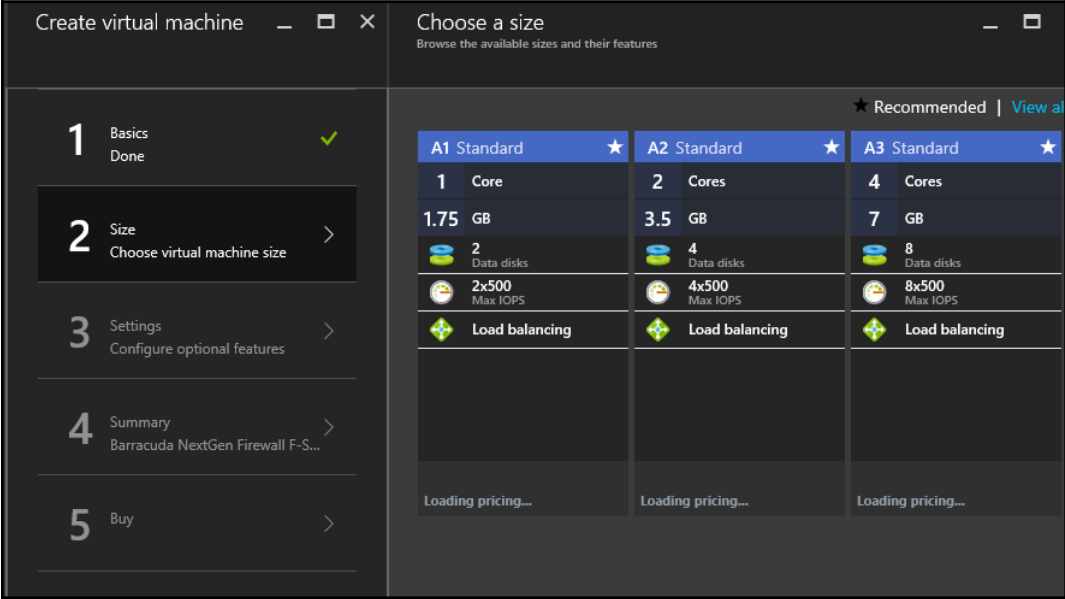
We will cover virtual machine deployments deeper in the upcoming chapters.

4. Depending on the operating system of the appliance, the interface may defer. For the used barracuda firewall example you have to give a name to the appliance, add the user account, and set a password or SSH key for connection later:

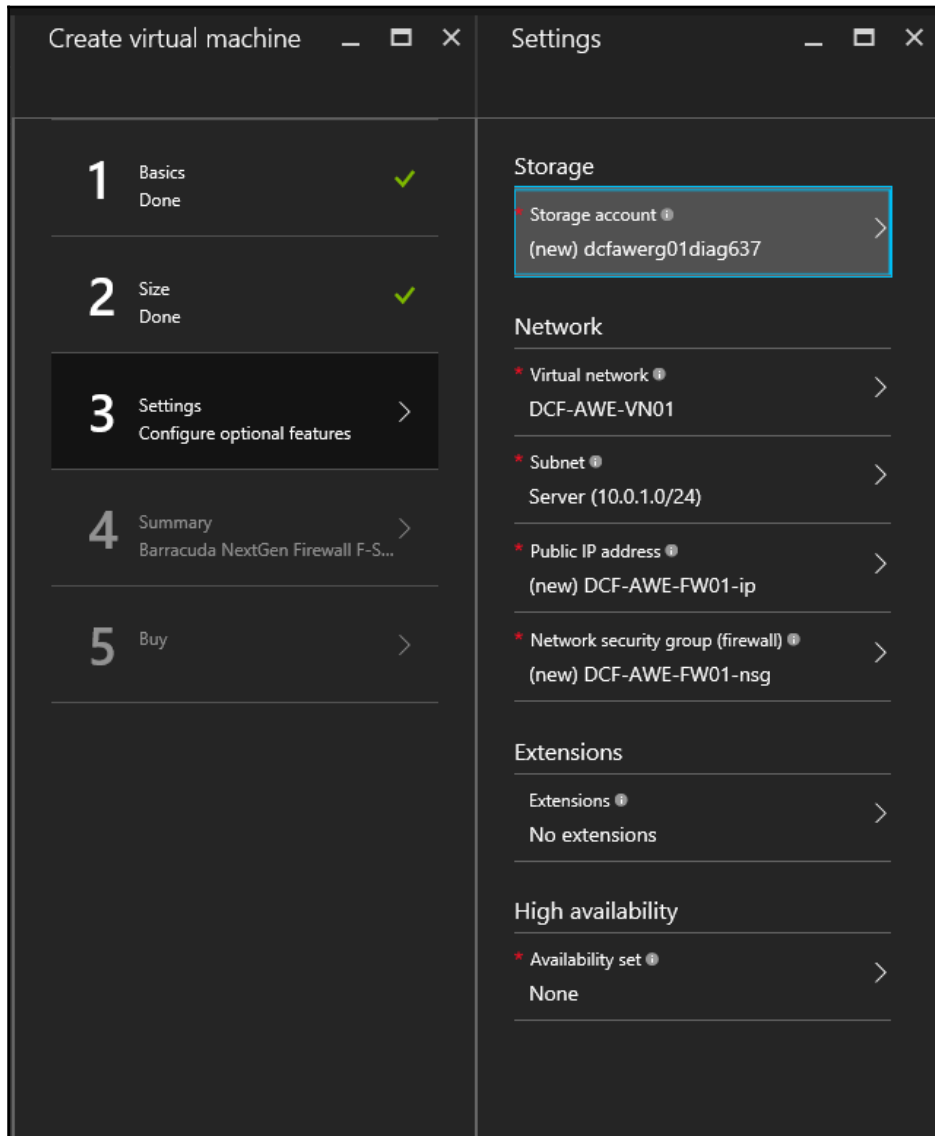
The screenshot displays the 'Create virtual machine' blade in the Azure portal, specifically the 'Basics' tab. The left sidebar shows a progress bar with five steps: 1. Basics (selected), 2. Size, 3. Settings, 4. Summary, and 5. Buy. The main area contains the following fields and options:

- Name:** DCF-AWE-FW01 (with a green checkmark)
- VM disk type:** HDD (with a dropdown arrow)
- User name:** [Redacted] (with a green checkmark)
- Authentication type:** SSH public key and Password (with a blue highlight on 'Password')
- Password:** [Redacted] (with a green checkmark)
- Confirm password:** [Redacted] (with a green checkmark)
- Subscription:** Visual Studio Premium with MSDN (with a dropdown arrow)
- Resource group:** DCF-AWE-RG01 (with a dropdown arrow)
- Location:** West Europe (with a dropdown arrow)

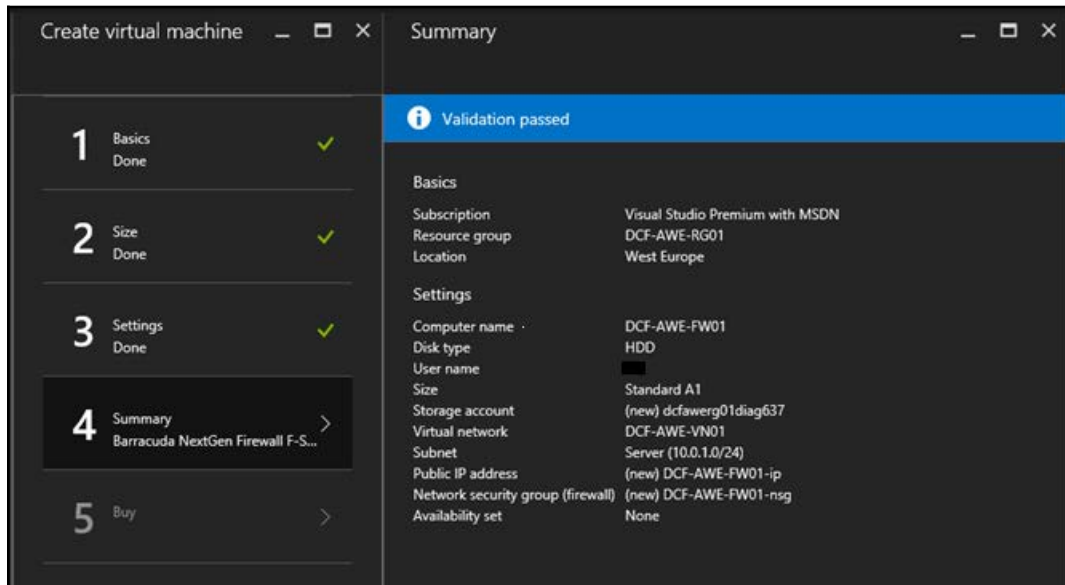
5. Now you need to select the VM size. The vendor already proposes some VM types but you should get in contact with the vendor to get the perfect sizing for your infrastructure:



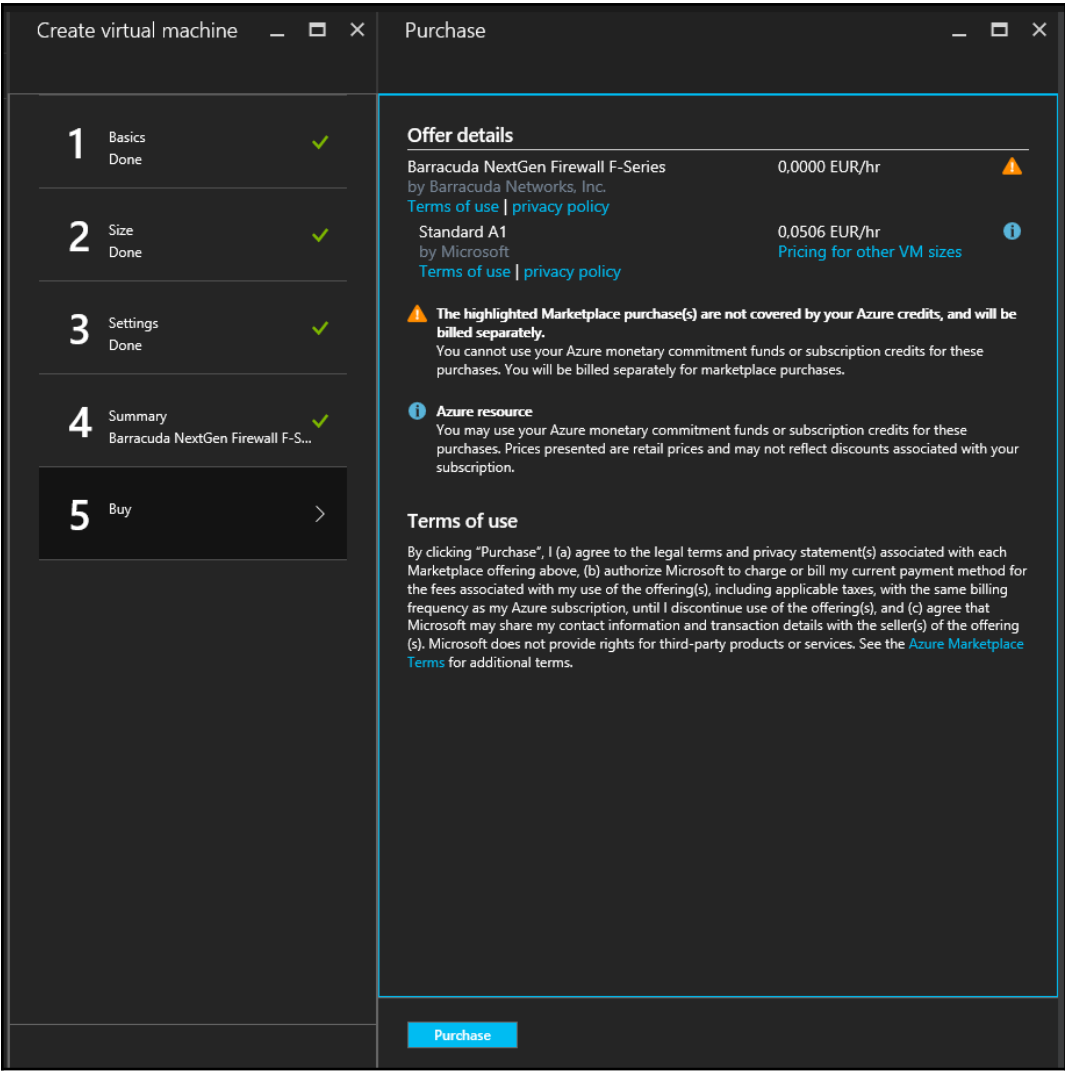
6. After you choose the size, you need to do some optional settings such as setting the **Storage account**, **Network**, IPs, the VM **Extensions**, or **High availability** partners you want to use. For now, go with the defaults; we will explain these deeper in Chapter 6, *Planning and Deploying Virtual Machines in Azure*:



7. Microsoft will show you a **Summary** and validate the settings against the support and configuration matrix:

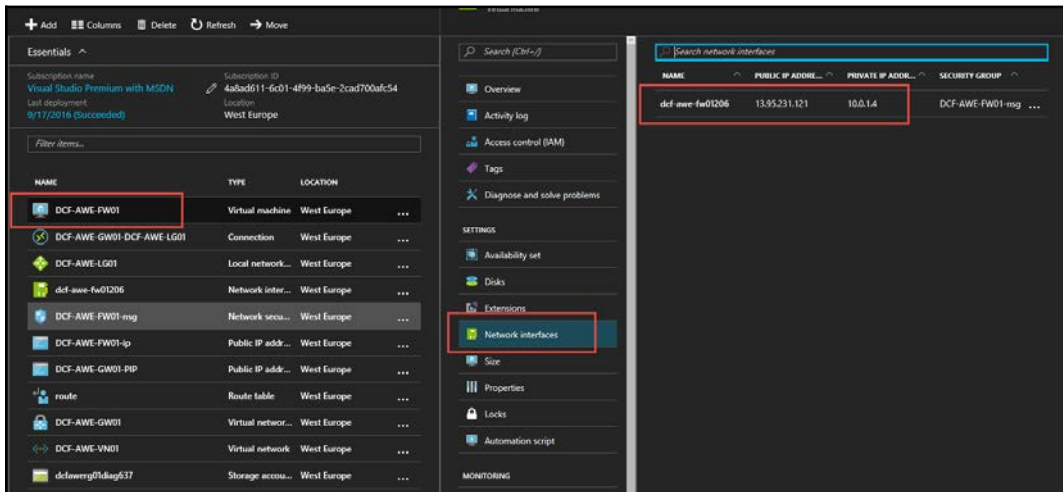


8. For a regular VM, the deployment would be finished at that point, for appliances running on Azure, you need to agree the third-party purchase before deploying:



9. After the deployment is finished, you should see the virtual machine in your resource group and also be able to access the device.

10. To connect to the device, you need to look for the IP first. You can find the IP given by Azure in the settings of the device:

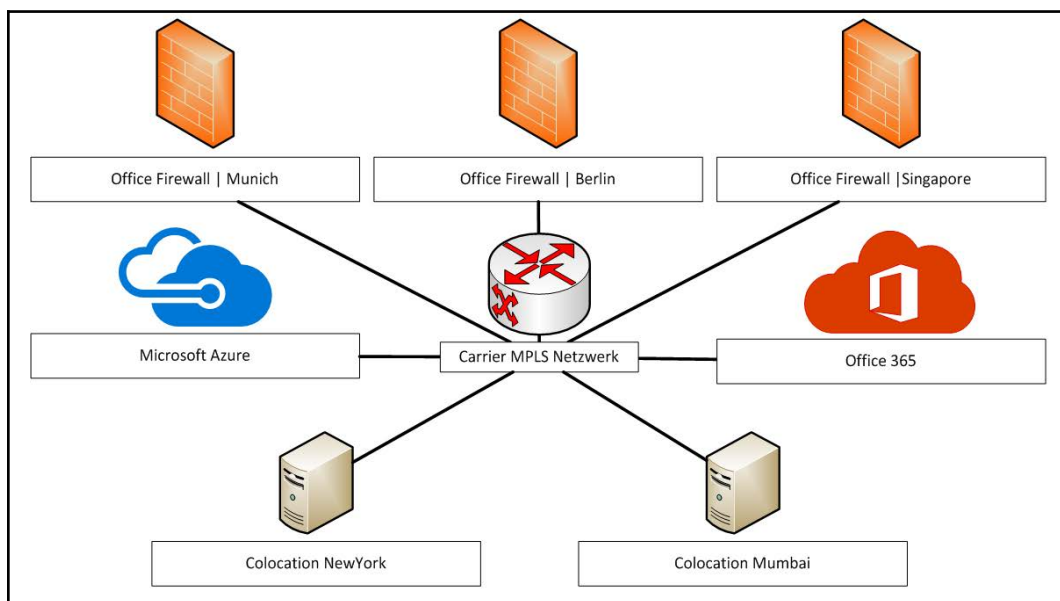


To connect to the device, the vendor normally provides a short How To on its websites. Most providers prefer the connection via SSH, HTTP, HTTPs, or remote desktop.

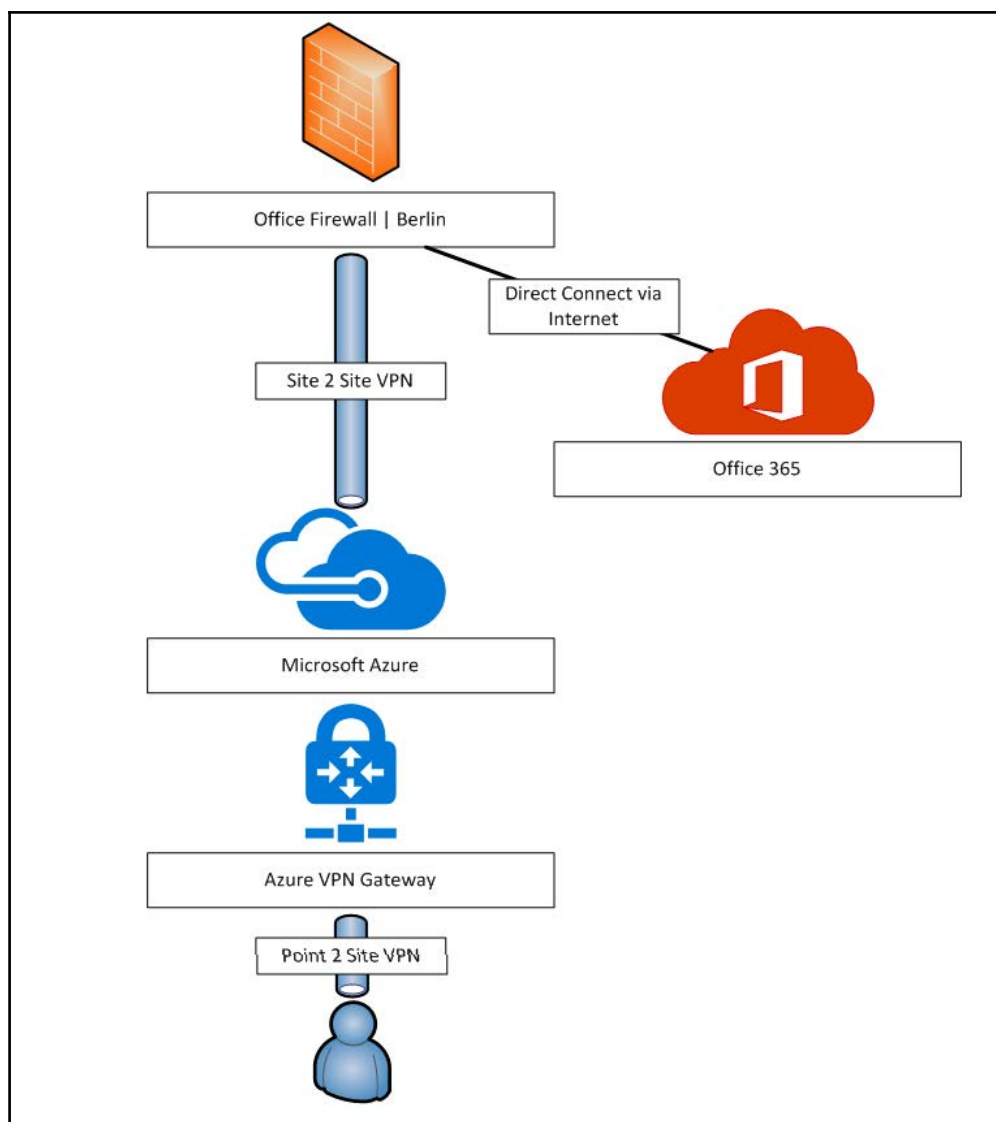
Common Azure Network Architectures

Looking at the networking scenarios, the most common one is to integrate Azure and Office 365 directly into your MPLS. Every connection from any location is transmitted via the MPLS network.

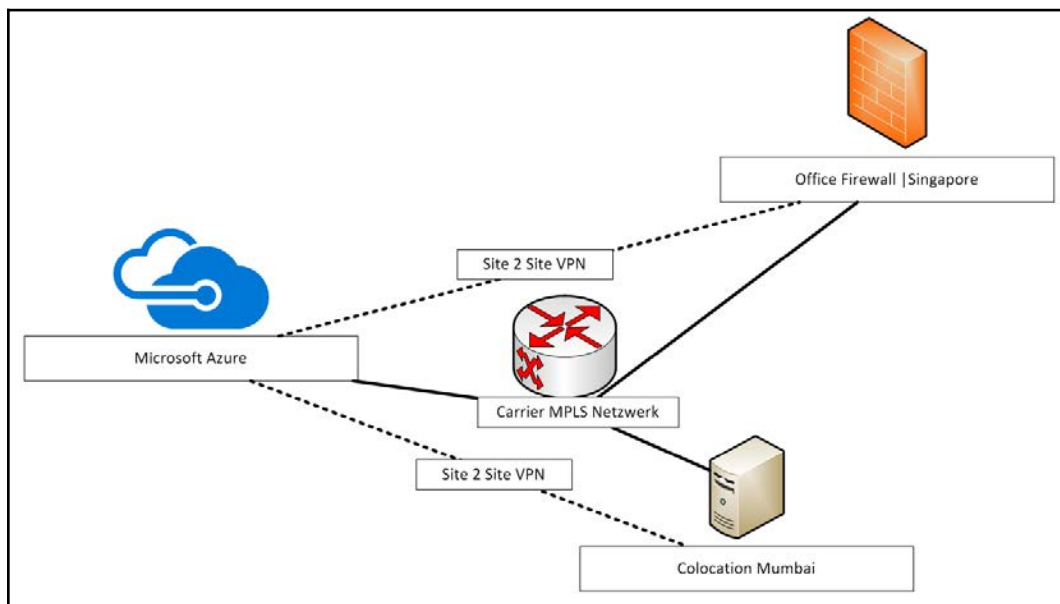
The following diagram shows a short abstract of such an environment:



There are also options to use Azure as colocation and connect offices via a VPN. This option is often used by small or medium business companies. There every VPN connection terminates in Azure. Office 365 is reached via Internet from the Office directly:



Another very common form of setting up WAN links to offices or other data centers is to have a primary link via ExpressRoute and a secondary link via a **Site 2 Site VPN** with BGP enabled. So your services stay available for your users even if your MPLS fails. You only have a performance impact but stay in production. The rerouting will happen automatically because of the enabled BGP:



There are also common scenarios where Azure is used only for online services without any on-premises connection or where resources are resold to other customers or end users:

Summary

So let us summarize this chapter. Now you should have learned how networking in and with Azure works. You now know what basics need to be deployed and how to deploy external connections with VPN or MPLS. You also learned about common scenarios and how to change implemented Azure behaviors such as routing.

In the next chapter, we will take a deeper look into Microsoft Azure Storage.

5 Implementing and Securing Azure Storage Accounts

Like you plan the network in your data center or company, you need to do it in Azure also. Many services in Azure are based on storage. Therefore, it has to be planned well.

You should consider scalability, durability, and high availability depending on the scenario and target you try to achieve.

Within this chapter, Azure Storage management is discussed. The key takeaways are how and when to implement and integrate the different Azure Storage types. The following sections and their various options are touched:

- Storage accounts
- Replication and redundancy
- Azure Storage services:
 - Blob
 - Table
 - Queue
 - File
- Exploring Azure Storage with Microsoft Azure Storage Explorer
- Premium Storage accounts
- Pricing

This chapter does not handle deep backup, Azure site recovery, or StorSimple sections. We will set up some basic storage configurations in this chapter.

Storage Accounts

Azure Storage implements four services: Blob storage, Table storage, Queue storage, and File storage.

Blob storage stores unstructured object data, which can be text or binary data.

Structured datasets are stored in Table storage. Table storage is a NoSQL key attribute data store, enabling rapid development and fast access to data.

In addition to providing reliable messaging for workflow processing, Queue storage makes communication between segments of cloud services available.

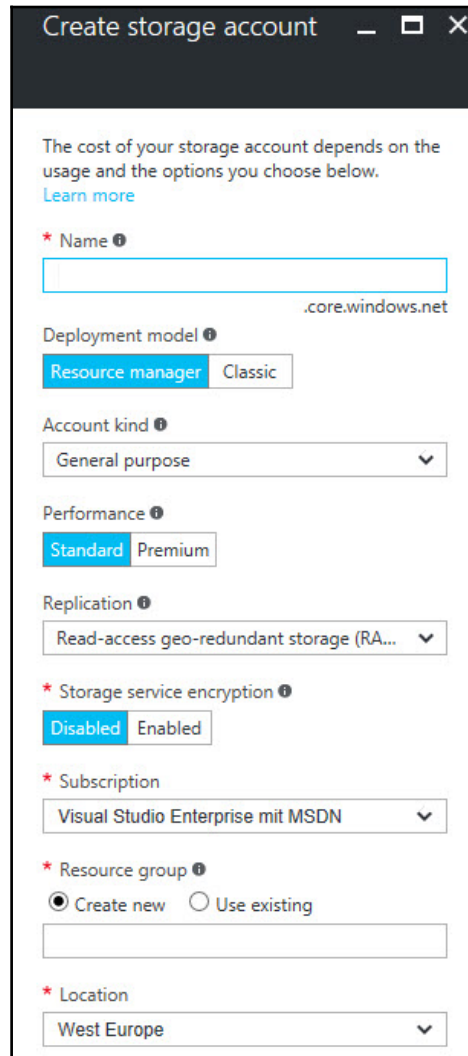
Using the standard SMB protocol, File storage offers shared storage for legacy applications. Azure virtual machines and cloud services can share file data over application components using mounted shares. Utilizing the file service REST API, on-premises applications can obtain file data:

1. For creating a Storage account, first search for `Storage` in the gallery and select **Storage account**:



Azure Storage account in the gallery

2. After clicking on **Create** button, there is a form with diverse options to fill in:



The screenshot shows the 'Create storage account' form in the Azure portal. The form is titled 'Create storage account' and includes a close button. It contains the following fields and options:

- Name:** A text input field with a hint icon. Below it, the text '.core.windows.net' is displayed.
- Deployment model:** Two buttons: 'Resource manager' (selected) and 'Classic'.
- Account kind:** A dropdown menu with 'General purpose' selected.
- Performance:** Two buttons: 'Standard' (selected) and 'Premium'.
- Replication:** A dropdown menu with 'Read-access geo-redundant storage (RA...)' selected.
- Storage service encryption:** Two buttons: 'Disabled' (selected) and 'Enabled'.
- Subscription:** A dropdown menu with 'Visual Studio Enterprise mit MSDN' selected.
- Resource group:** Two radio buttons: 'Create new' (selected) and 'Use existing'. Below them is a text input field.
- Location:** A dropdown menu with 'West Europe' selected.

Available options at creation

3. The first step is to choose a **Name**. The important thing about Storage account names is that they can only consist of lowercase letters and number. No spaces here! The example Storage account is named `implementingazuredemo`.

To understand the options in the above figure, it is important to understand the different possibilities. There are two separate kinds of Storage account to choose from:

- General purpose Storage account
- Blob Storage account

General Purpose Storage Account

The general purpose Storage account is very universal. It can contain storage services of any type available. That includes Blobs (of course also virtual machine disks based on Blobs), Files, Queues, and Tables in a Storage account. On creation, there are two available performance settings available. The first available performance option is the standard option. This type of Storage account holds Queues, Tables, Blobs, and Files.

The second option is the premium one. This is only capable of storing Azure virtual machine disks. For more information about this see the *Premium Storage* section later in this chapter.

Blob Storage Account

A general purpose Storage account provides entrance to Azure Storage services such as Tables, Queues, Files, Blobs, and Azure virtual machine disks, combined under a single account. The two performance tiers are:

- **Standard storage performance tier:** The standard storage performance tier permits the customer to file Tables, Queues, Files, Blobs, and Azure virtual machine disks
- **Premium storage performance tier:** This currently exclusively supports Azure virtual machine disks

To store unstructured data as Blobs (objects), a Blob Storage account is available in Azure Storage. Blob storage shares characteristics with existing general purpose Storage accounts. Similar are the durability, availability, scalability, and performance features. Microsoft recommends using Blob Storage accounts for applications requiring entirely block or append Blob storage.

Blob Storage accounts expose the access tier attribute which can be specified in the process of account creation. It is possible to modify later if needed. Two types of access tiers can be defined based on the data access pattern:

- **Hot access tier:** This tier designates that the objects in the Storage account will be obtained on a frequent basis. This allows data storage at a lower access cost.
- **Cool access tier:** This tier indicates that the objects in the Storage account will be less regularly accessed. This too allows data storage at a lower cost.

It is permitted to switch between these tiers if there is a change in the usage pattern of data. It must be noted that changing the access tier can result in additional costs.



MSDN subscribers, for example, can get free monthly credits which can be used with Azure services, including Azure Storage.

The requirement to create a Storage account is to have an Azure subscription. The subscription enables the customer access to numerous Azure services. It is possible to create a free Azure account to get started. Once the consumer decides to acquire a subscription plan, it is possible to choose from a variation of purchase alternatives. A single customer can create up to 100 Storage accounts with an individual subscription.



As there are several differences in pricing for the two account types, in a Blob Storage account, the access tier (hot, cold) also indicates the billing model. The SLAs for both are nonetheless the same.

In the example, we will choose a general purpose Storage account. The next configurable field, besides standard and premium storage, is the replication setting.

To be able to know which one we need, it's important to first understand the different replication redundancy settings.

Replication and Redundancy

In order to guarantee stability and high availability, the customer's data in the Azure Storage is replicated constantly. The customer may choose between two replication options: either a storage within the same data center or to a second data center. Replication guards the user's data; in the case of hardware failures, the application is preserved. The use of a second data center provides security for the case of a catastrophic failure in the location of the primary data center.

The process of replication warrants that the customer's Storage account meets the **Service Level Agreement (SLA)** for storage.

There are four replication options between which the user can choose when creating an Azure Storage account.

Locally Redundant Storage

Locally redundant storage (LRS) means that the data is held three times in a data center in a region. The LRS manages three copies of the customer's data to protect it from hardware failures. LRS does not protect the workloads from the failure of a whole data center.

Zone-Redundant Storage

Zone-redundant storage (ZRS) stores three copies of the customer's data as well as the LRS. The difference is that the data is guarded in two to three facilities. These facilities can be located in different regions. This concept provides more enhanced durability than LRS. The user profits from durability within a region.

Geo-Redundant Storage

An even higher durability can be achieved with **Geo-redundant storage (GRS)**. GRS manages six copies of the user's data. The first three copies are replicated in the primary region. Additionally, another three copies are maintained in a secondary region which is located remotely from the primary region. This concept provides an even higher level of durability. This means that Azure Storage will failover to the secondary region if a failure in the primary region should occur.

Read-Access Geo-Redundant Storage

The replication to a secondary geographic location is provided with **Read-access geo-redundant storage (RA-GRS)**. The customer holds read access to the data, maintained in the secondary location. Access from the primary and the secondary region is possible. The RA-GRS is the default option for your Storage account on creation.

This is an overview about the redundancy options in Azure Storage.

Replication strategy	LRS	ZRS	GRS	RA-GRS
Data is replicated across multiple data centers	No	Yes	Yes	Yes
Data can be read from the secondary location as well as from the primary location	No	No	No	Yes
Number of copies of data maintained on separate nodes	3	3	6	6

Source: <https://docs.microsoft.com/en-us/azure/storage/storage-redundancy>

For our example, we select **Locally-redundant storage (LRS)**, as high durability is not necessary in our example.

Let's look at the current settings:

Create storage account

The cost of your storage account depends on the usage and the options you choose below.
[Learn more](#)

* Name ⓘ

implementingazuredemo

✓

.core.windows.net

Deployment model ⓘ

Resource manager

Classic

Account kind ⓘ

General purpose

▼

Performance ⓘ

Standard

Premium

Replication ⓘ

Locally-redundant storage (LRS)

▼

* Storage service encryption ⓘ

Disabled

Enabled

* Subscription

Visual Studio Enterprise

▼

* Resource group ⓘ

☐ Create new

☒ Use existing

AzureExamples

▼

* Location

North Europe

▼

☐ Pin to dashboard

Create

[Automation options](#)

Storage account settings

In the example, an existing **resource group** is used to store the Storage account resource. **North Europe** is used as deployment **Location** as it's the nearest location and will probably have the least latency.

To achieve the same using PowerShell with the Azure PowerShell module, the following commands can be used:

```
New-AzureRmStorageAccount `
-ResourceGroupName 'AzureExamples' `
-Name 'implementingazuredemo' `
-Location 'northeurope' `
-SkuName 'Standard_LRS' `
-Kind 'Storage'
```

In the previous command, `ResourceGroupName` is the resource group, that the Storage account should be deployed to. `Name` is the planned name for the Storage account, `location` is the deployment Azure region, and `SkuName` is a mix of the performance and replication setting from the Azure portal. This parameter can take the following values:

- 'Standard_LRS': Locally redundant storage
- 'Standard_ZRS': Zone-redundant storage
- 'Standard_GRS': Geo-redundant storage
- 'Standard_RAGRS': Read-access geo-redundant storage
- 'Premium_LRS': Premium locally redundant storage



For more information visit

<https://docs.microsoft.com/en-us/powershell/resourcemanager/azurerm.storage/v2.3.0/new-azurermstorageaccount>.

Azure Storage Services

As previously mentioned, the difference between the Blob storage and the general purpose storage is to be found in the purpose of usage. While the Blob storage stores unstructured data, the general purpose account stores structured data. Azure differentiates between four types of storage: Blob, Queue, Table, and File storage. It's important to understand the scopes, in order to be able to decide for a certain type of storage.

Blob Storage Services

For customers needing to store large sets of unstructured data, Blob storage offers an attractive and scalable answer. The types of data may be retained in Blob storage are: documents, photos, music, videos, blogs, file backup, databases, images and text for web applications, big data, or configuration data for cloud applications.

Containers offer a useful way to assign security policies to sets of objects; each Blob is assigned a container. A Storage account can hold indefinite containers; a container may contain an indefinite number of Blobs. The only restriction is the 500 TB capacity limit of the Storage account.

There are three types of Blobs: block blobs, append blobs, and page blobs (disks).

Block blobs are utilized for streaming and storing cloud objects. They are a best used for storing documents, media files, backups, and so on.

Whereas block blobs are used for streaming and storing, append blobs fulfill the same task with the addition that they are optimized for append operations. Updating an append blob can only be done by adding a new block to the end. Append blob's field of application consists of logging, in which data has to be written to the end of the blob.

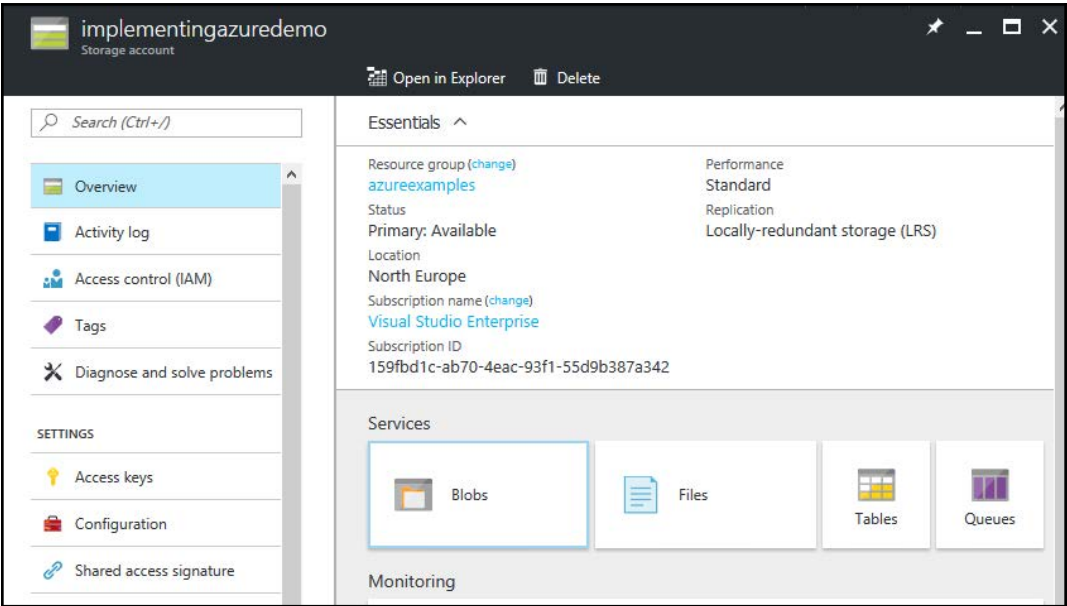
The third type of Blob is the page blob. In most cases, page blobs are used to store IaaS disks. They support random writes. This means that an Azure IaaS VM VHD is stored as a page blob.

In the cases where downloading data over the wire to Blob storage is unrealistic, for example for large datasets, the customer is able to send a hard drive directly to Microsoft, where the data gets directly imported or exported from the data center. In Azure, Blobs are stored in containers. These containers are the upper-most element that needs to be used, to store files as Blobs.

In Azure IaaS there will be a VHDs container created when you deploy an image from the gallery. These containers hold the VHD files for the VMs as page Blobs and also hold status Blobs as block blobs.

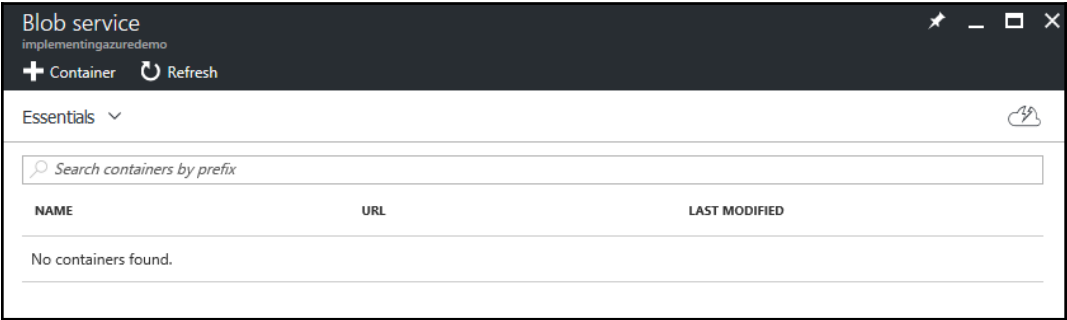
Let's look at Azure Blob storage:

1. Navigate to the earlier created Storage account, named **implementingazuredemo**:



Storage account overview

2. In the previous figure, are all the types of storage that can be used in the Storage account. For managing Blobs, the tile named **Blobs** needs to be clicked:



Blob service overview

3. There is an empty window, showing, that there are no containers. To create a container, click on **Container**. This will ask for a **Name** and **Access type**. The **Name** has a common Azure policy, so lowercase only.
4. The access type defines how your Blobs can be accessed publicly:

Type	Description
Private	If Private is selected, the Blobs can only be accessed by the account owner with the access key and no anonymous access is granted. In PowerShell, this option is referred to as off.
Blob	When Blob is selected, only the Blobs can be access from the outside with read permissions.
Container	If Container is selected, the whole container content will be publicly available with read rights.



Remember that these are access policies for the Blob container only and have no influence on the other containers or the Storage account.

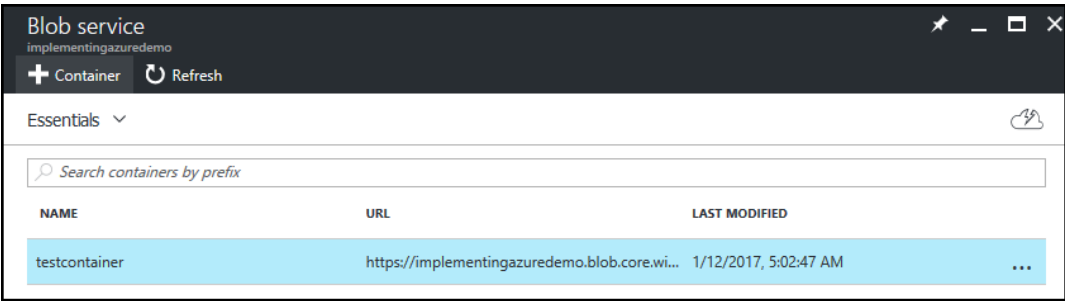
5. In the **New container** window, type `testcontainer` as **Name** and select **Private** as **Access type**:

New container dialogue

6. After checking all the provided details are correct, click on **Create** button to create new container:

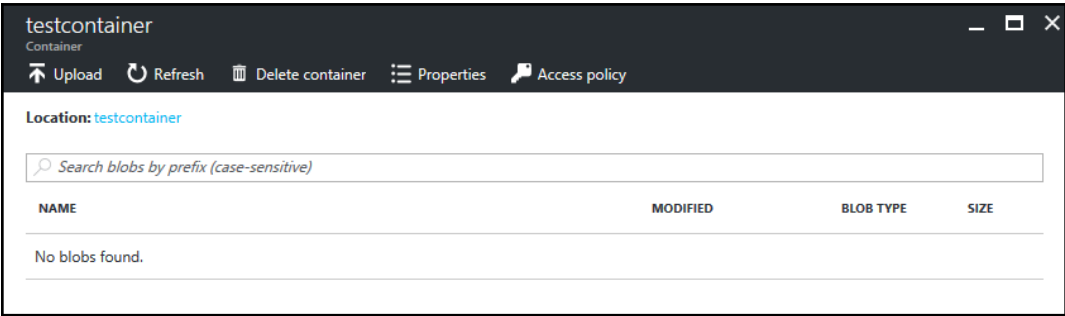


7. The new container is named `testcontainer`. After clicking on **Create** button the deployment process will start:



Blob services overview

8. After the creation, the newly created resource is listed, and when clicked a new window appears. In this window, it's possible to upload files, view the **Properties** of the container, and manage the **Access policy**:



Overview of testcontainer's Blobs

9. To achieve the same with PowerShell, the following cmdlets have to be executed. The cmdlets require that Azure PowerShell is already connected to an account and an active subscription is selected:

```
#Define required variables
$storageAccountName = 'implementingazuredemo'
$resourceGroupName = 'AzureExamples'

#Get the Storage account keys
$storageAccountKey = Get-AzureRmStorageAccount `
-name $storageAccountName `
-ResourceGroupName $resourceGroupName `
| Get-AzureRmStorageAccountKey

#Create a Storage context for working with the Storage account
$storageContext = New-AzureStorageContext -StorageAccountName
$storageAccountName -StorageAccountKey
$storageAccountKey[0].Value

#Create a blob container named testcontainer
New-AzureStorageContainer -Name 'testcontainer' -Context
$storageContext

#Verify the creation
Get-AzureStorageContainer -Context $storageContext
```

10. In the example, we first need to get a so-called Azure Storage context. In this context, we can execute common storage tasks such as creating containers, uploading and downloading files, and so on.
11. To create a remote context, one option is to use a Storage account key. This is the easiest one, as it can be done with the `Get-AzureRmStorageAccountKey` cmdlet.
12. After creating the context based on the Storage account name and the first of both account keys, a container called `testcontainer` is created.
13. For more information on Azure Storage account keys, see the section Access keys.



Blob storage can also be managed via Azure REST API.

Table Storage Services

Table storage in Azure can be described as a NoSQL database. This basically means that the database has no schema and each value in a table has a typed property name. This property name can be used for filtering, sorting, and as selection criteria. There are multiple entities in a table that each consist of a collection of values and their property names.

NoSQL thus also Azure Table storage has much higher performance, scalability, and flexibility at a much lower, complexity.

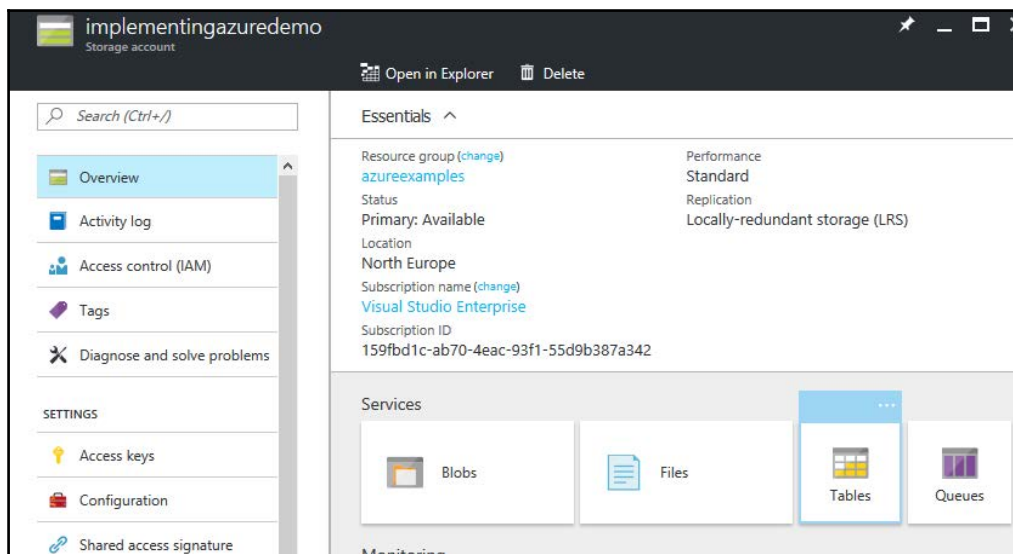
Common usage scenarios for Table storage are databases or datasets for web applications, collections of metadata, or bigger collections, for example, addresses.

As with the other Azure Storage services, the only limiting factor for Table storage is the size of your Storage account, which means, there is no limit for the number of tables or entities in tables.

Since Table storage is fast to set up and access, the next demo will show the creation of a simple table. Often Table (NoSQL) storage is much cheaper than traditional relational databases.

Let's create a Table services table:

1. First, click on **Tables** in the **Storage account** dashboard:



Storage account overview

2. After the **Table service** page has loaded, it's recognized that there is no way of creating or interacting with tables in the Azure portal:

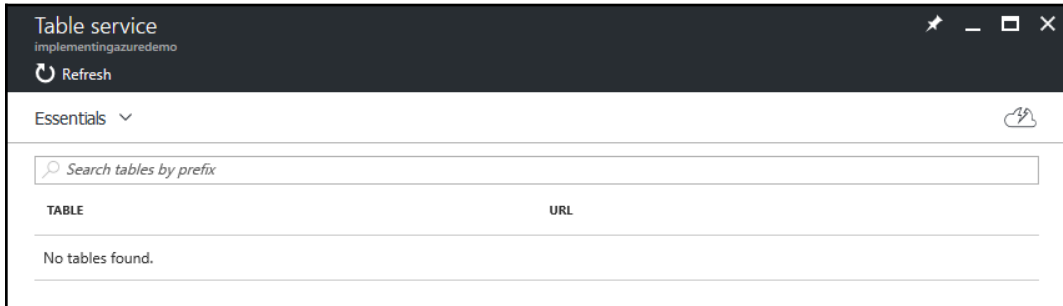


Table service overview

3. As there is nothing more to do in the Azure portal, the following PowerShell code is used to create a table:

```
#Define required variables
$storageAccountName = 'implementingazuredemo'
$resourceGroupName = 'AzureExamples'
$tableName = 'testtable'

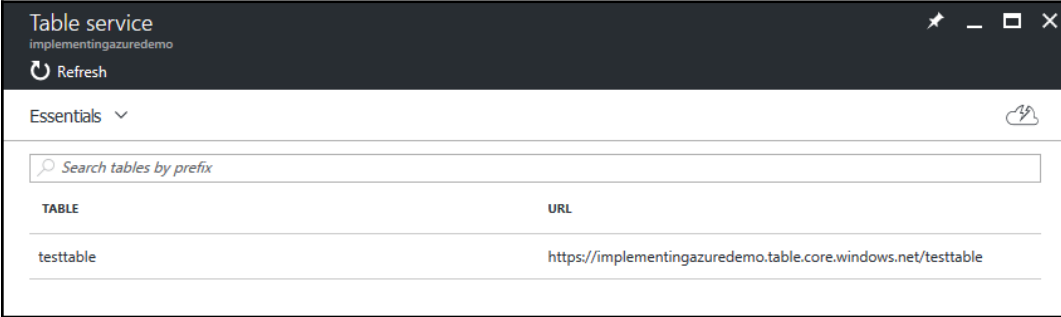
#Get the storage account keys
$storageAccountKey = Get-AzureRmStorageAccount `
    -name $storageAccountName `
    -ResourceGroupName $resourceGroupName `
    | Get-AzureRmStorageAccountKey

#Create a Storage Context for working with the storage
account
$storageContext = New-AzureStorageContext -
    StorageAccountName $storageAccountName -StorageAccountKey
    $storageAccountKey[0].Value

#Create a table named testtable
New-AzureStorageTable -Name $tableName -Context
    $storageContext

#Verify the creation
Get-AzureStorageTable -name $tablename -Context
    $storageContext
```

4. The successful deployment can also be seen in the Azure portal. When navigating back to the **Table service** overview of the Storage account, the newly created table appears under **TABLE**, including its **URL**, on Azure:



The screenshot shows the Azure portal interface for the 'Table service' of an account named 'implementingazuredemo'. At the top, there is a 'Refresh' button and a tab labeled 'Essentials'. Below this is a search bar with the placeholder text 'Search tables by prefix'. A table lists the available tables, with columns 'TABLE' and 'URL'. One table is listed: 'testtable' with the URL 'https://implementingazuredemo.table.core.windows.net/testtable'.

TABLE	URL
testtable	https://implementingazuredemo.table.core.windows.net/testtable

Newly created table

5. As there are currently no cmdlets available for managing table entries directly, this has to be done using the SDK in .NET or using the respective library in PowerShell. That is obviously out of scope for this book.



Table storage can also be managed via Azure REST API.

Queue Storage Services

Azure Queue enables messaging between different parts of applications. This is used in the development of highly scalable and flexible applications. Components of applications are often decoupled, to enable independent scalability for the single parts.

Queues are also used as an asynchronous method of communication between components that run on different locations (cloud, on-premises, desktop, mobile). It's also possible to build workflows and asynchronous tasks based on Azure Storage Queues.

One Storage account has no limit on the number of queues, as well as the number of messages these contain. A single message can be up to 64 KB in size.

There are two different kinds of queues in Microsoft Azure. The first one is Azure Queues, which is a part of Azure Storage and the one that we are working within this chapter.

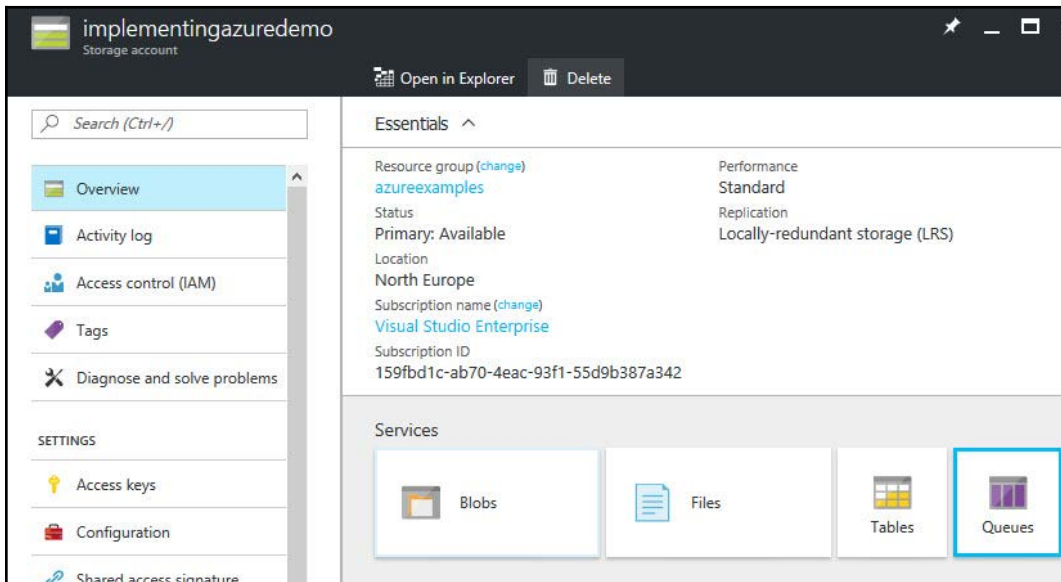
The second is Service Bus queues, which is a feature of Microsoft's Azure messaging infrastructure and has more advanced features for application development. According to Microsoft, Azure Storage queues should be used, when your application must store over 80 GB of messages in a queue, where the messages have a lifetime shorter than 7 days. And Service Bus queues should be used when you need more advanced features such as guaranteed **first in first out (FIFO)** ordered delivery, bigger handles, message receiving without polling, and so on.



For more information on how to choose the right queue solution, visit <https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-azure-and-service-bus-queues-compared-contrasted>.

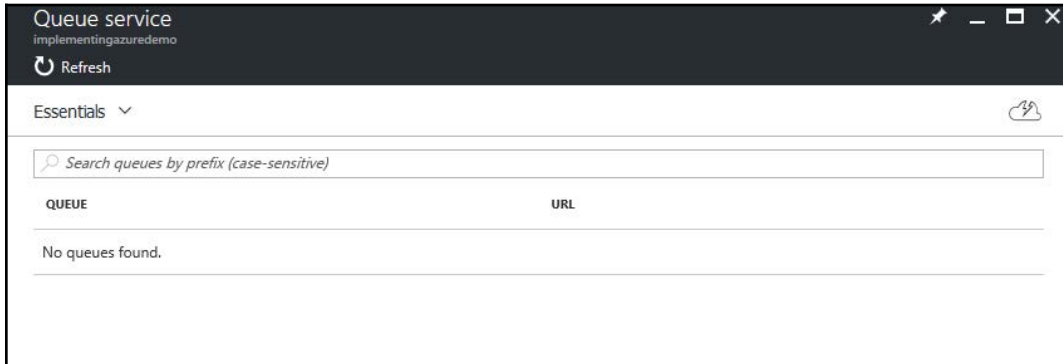
Let's see how queues look on the Azure portal:

1. First, navigate to the earlier created Storage account:



Storage account overview

2. Click on **Queues**, to open the **Queue service** overview. There should be no queues in there:



Queue service overview



These are all steps that are possible in the Azure portal. Any creation, polling, or management of queues must happen within PowerShell, Azure CLI or REST API.

3. As all the other things would go too deep into development, we use PowerShell to create a new queue called `testqueue`:

```
#Define required variables
$storageAccountName = 'implementingazuredemo'
$resourceGroupName = 'AzureExamples'
$queueName = 'testqueue'

#Get the storage account keys
$storageAccountKey = Get-AzureRmStorageAccount `
  -name $storageAccountName `
  -ResourceGroupName $resourceGroupName `
  | Get-AzureRmStorageAccountKey

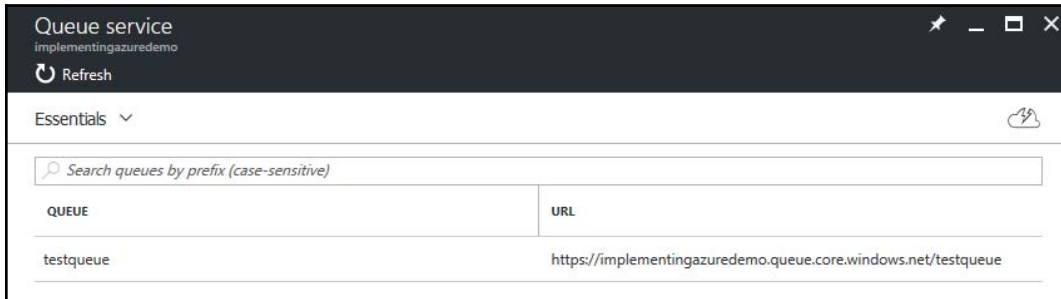
#Create a Storage Context for working with the storage account
$storageContext = New-AzureStorageContext -StorageAccountName
$storageAccountName -
  StorageAccountKey $storageAccountKey[0].Value

#Create a queue named testqueue
$Queue = New-AzureStorageQueue -Name $queueName -Context
$storageContext
```



```
#Verify the creation
Get-AzureStorageQueue -name $queueName -Context
$storageContext
```

4. As with Table storage, we first need to create a storage context to interact with the queue components. To verify that the deployment was successful, the Azure portal should be opened again. In the **Queue service** overview, there should be a new queue called `testqueue`:



Newly created queue service

5. Also, there is no cmdlet available for directly interacting with queues in PowerShell. For interacting with queues the .NET library should be used.



Queue storage can also be managed via Azure REST API.

File Storage Services

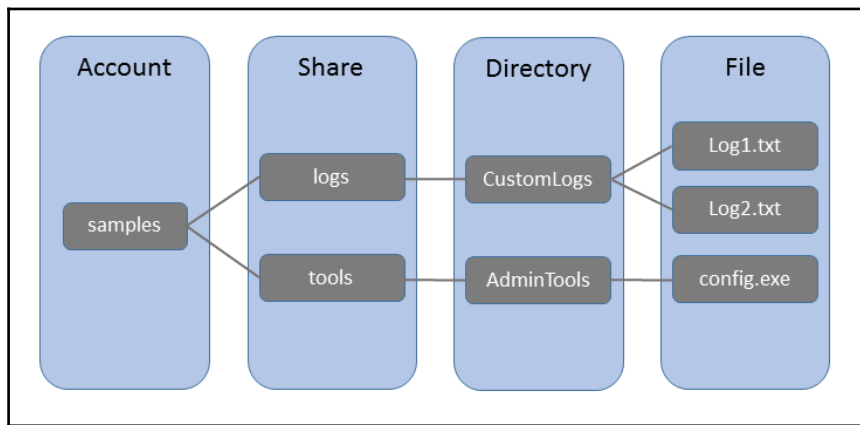
Currently, the most interesting type of storage for the IT Professional is the File storage.

File storage in Azure refers to cloud-based **Server Message Block (SMB)** or **Common Internet File System (CIFS)** such as that provided by traditional Windows or Samba file servers.

Like an SMB share, an Azure Storage share can be used from multiple computers and by multiple users simultaneously. The difference is that the users don't have to be connected to the company network anymore.

Azure File shares are commonly used for so-called **lift-and-shift** migrations, where the on-premises app is basically copied to the cloud as is. This is often fast and easy, but not always the most cost-efficient solution.

Other scenarios are shares for diagnostics or debugging data, shared application files, or simply temporary storage:

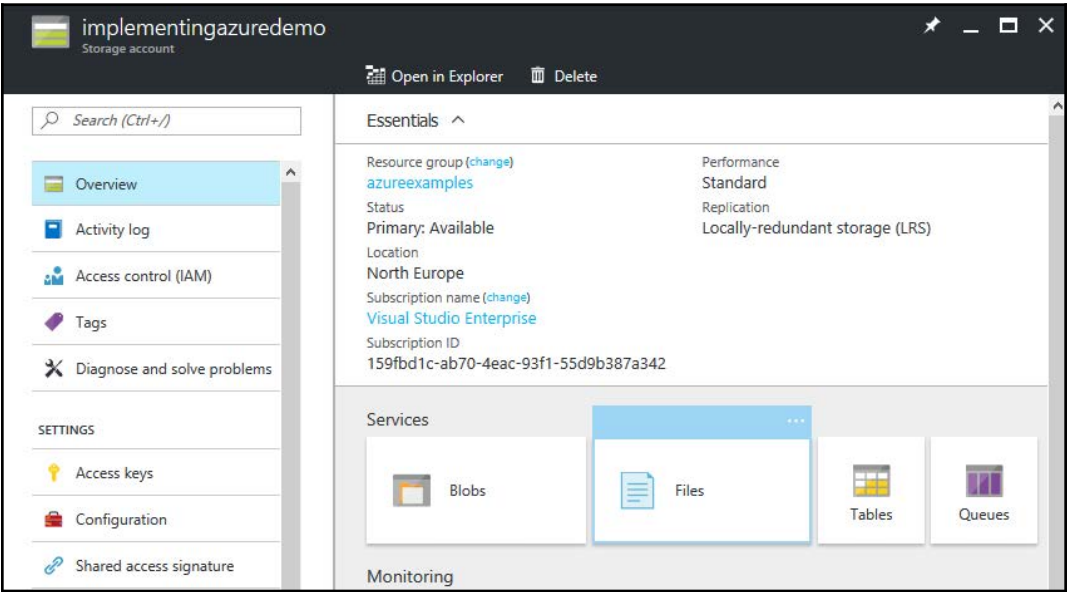


Levels of File storage (<https://docs.microsoft.com/en-us/azure/storage/storage-dotnet-how-to-use-files>)

In the above figure, the different logical levels of Azure File storage are shown. Directories and files are optional. Therefore it's enough to create a share and connect to it, to start working with File storage.

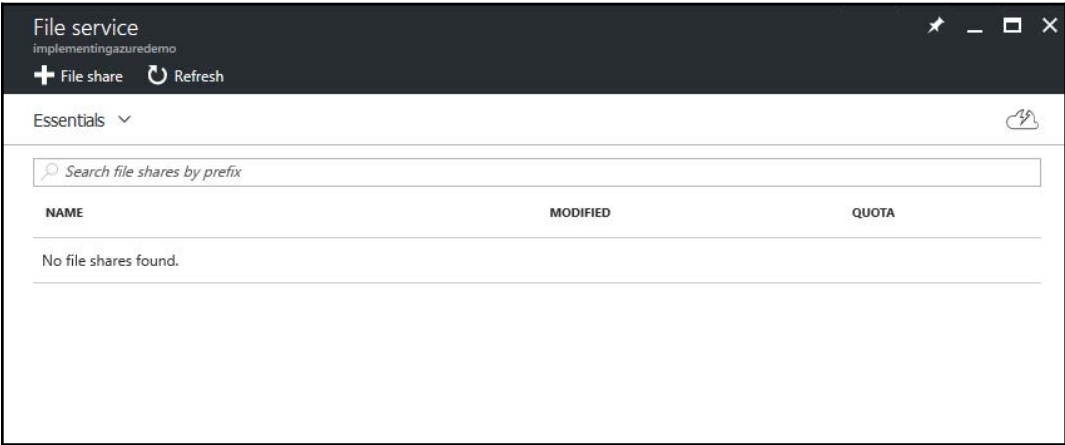
Let's create a File share and mount it locally:

1. The first thing is to open the overview of the earlier created Storage account **implementingazuredemo**:



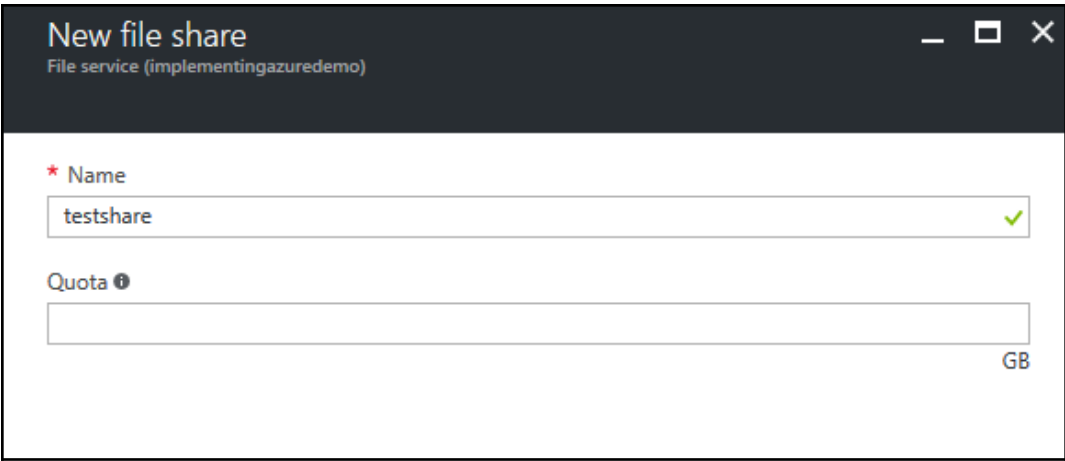
Storage account overview

2. After clicking on **Files**, the **File service** overview opens and shows that there are no File shares:



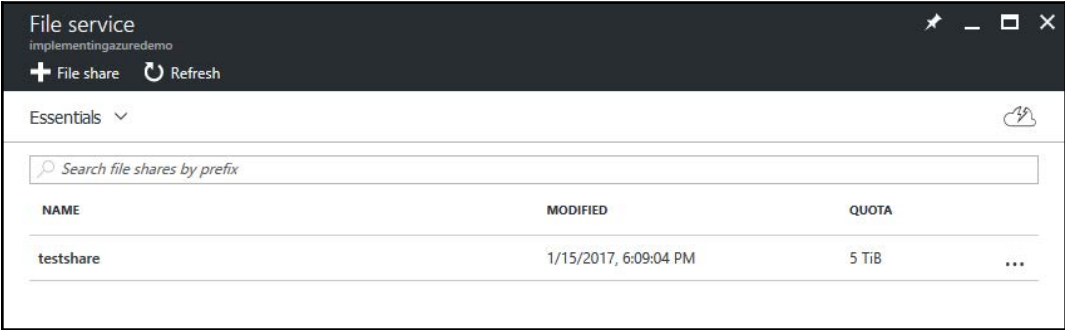
File service overview

3. To create a **New file share**, click the **File share** button and fill in a **Name** for the new File share. **Quota** is an optional parameter:



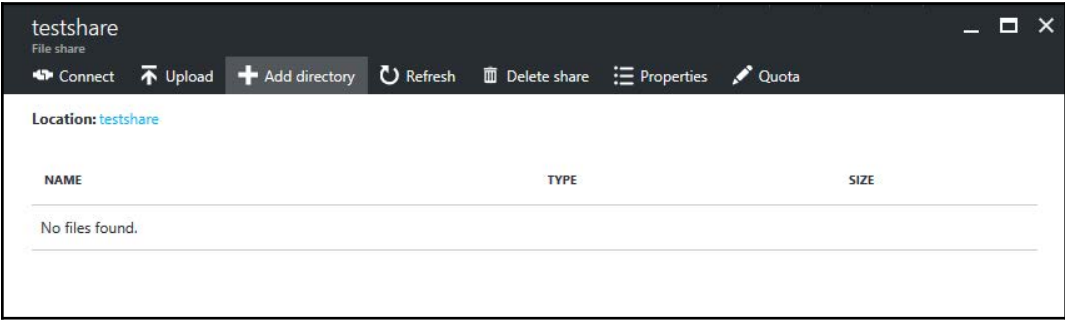
New file share dialogue

4. After clicking on the **Create** button, the portal takes us back to the **File service** overview. The new share named `testshare` appears in the **File service** overview window and can now be used to work with:



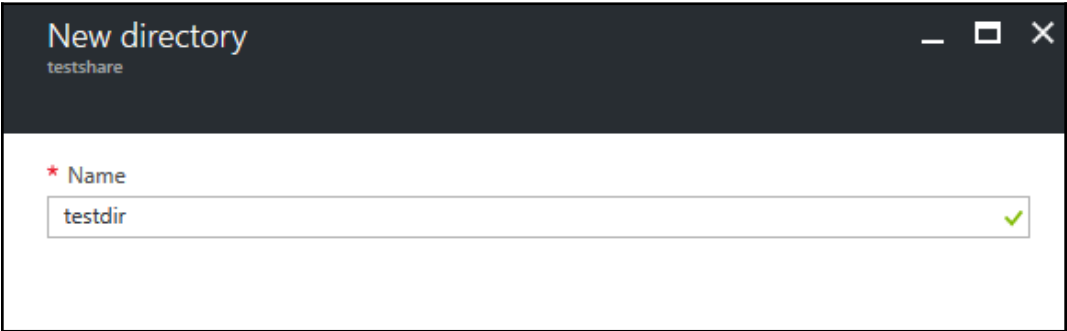
Newly created File share in overview

5. When selecting the share, the **testshare** properties open. In contrast to Queue and Table storage, it's possible to interact with the storage in the portal. Creation, directory management, quota setting, as well as uploading and downloading files is supported:



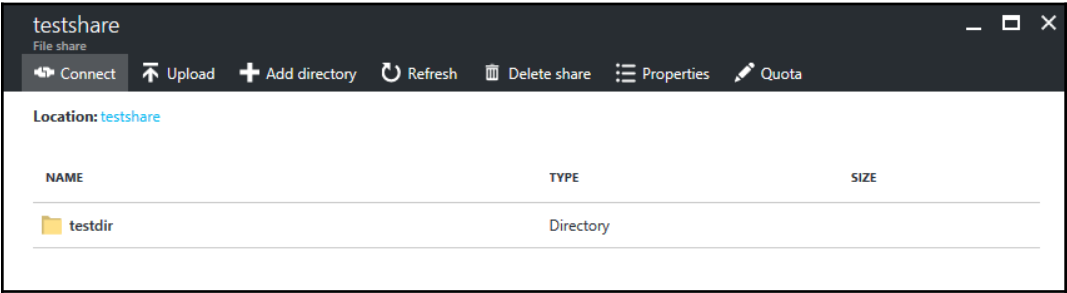
File share properties

6. To validate the connection to the test machine, a directory called `testdir` is created:



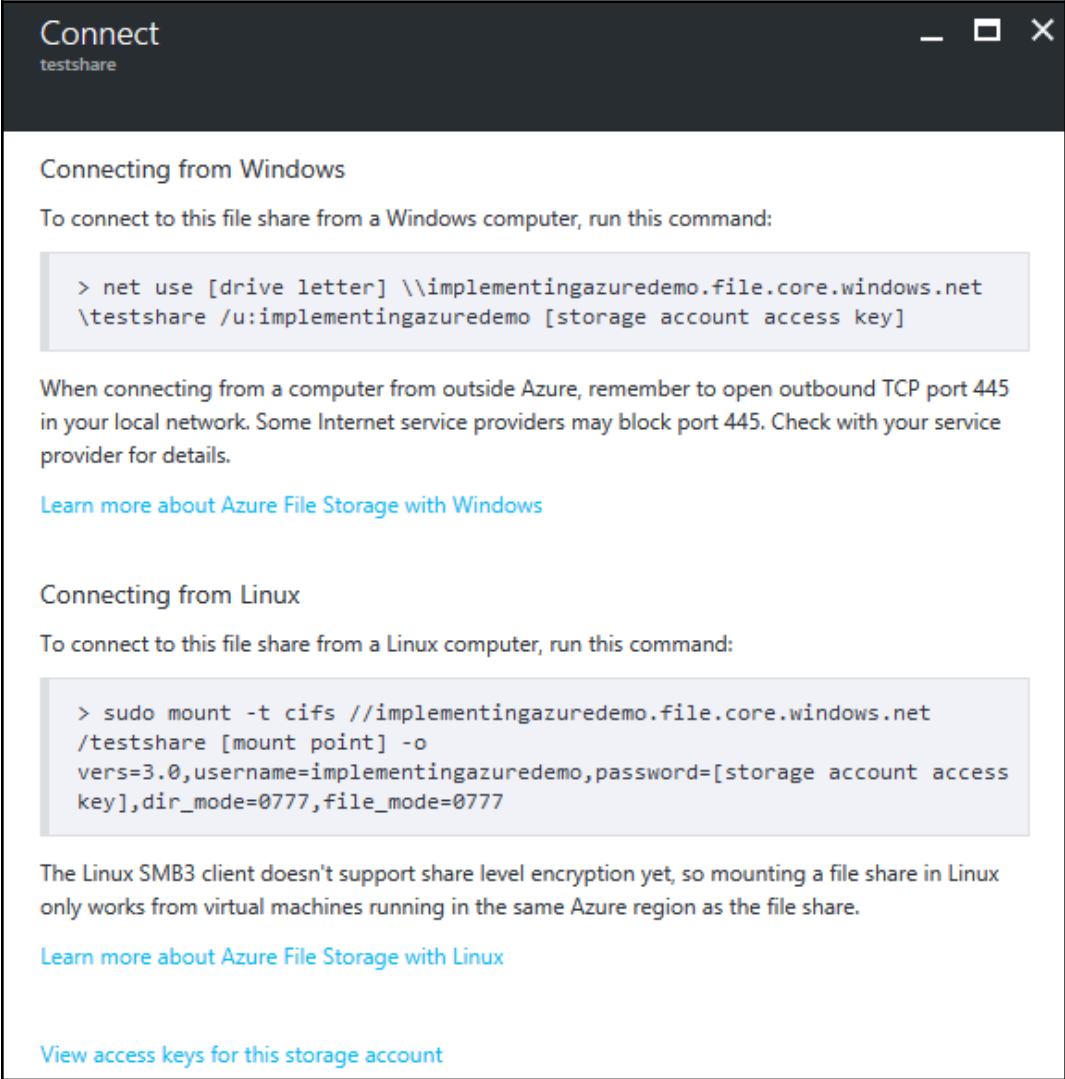
Directory creation dialogue

7. After clicking on the **Create** button, it shows up in the **testshare** contents in a new blade:



Directory browser of the testshare

8. Now it's time to connect the share to a desktop computer. For this purpose, the **Connect** button was created. After selecting **Connect**, the instruction to connect to Windows, as well as Linux, show up:



The screenshot shows a window titled "Connect" with a subtitle "testshare". It contains two sections: "Connecting from Windows" and "Connecting from Linux". Each section provides a command to run on the respective operating system to connect to the file share. There are also links for more information and a link to view access keys.

Connect
testshare

Connecting from Windows

To connect to this file share from a Windows computer, run this command:

```
> net use [drive letter] \\implementingazuredemo.file.core.windows.net
/testshare /u:implementingazuredemo [storage account access key]
```

When connecting from a computer from outside Azure, remember to open outbound TCP port 445 in your local network. Some Internet service providers may block port 445. Check with your service provider for details.

[Learn more about Azure File Storage with Windows](#)

Connecting from Linux

To connect to this file share from a Linux computer, run this command:

```
> sudo mount -t cifs //implementingazuredemo.file.core.windows.net
/testshare [mount point] -o
vers=3.0,username=implementingazuredemo,password=[storage account access
key],dir_mode=0777,file_mode=0777
```

The Linux SMB3 client doesn't support share level encryption yet, so mounting a file share in Linux only works from virtual machines running in the same Azure region as the file share.

[Learn more about Azure File Storage with Linux](#)

[View access keys for this storage account](#)

Azure File Storage connection commands

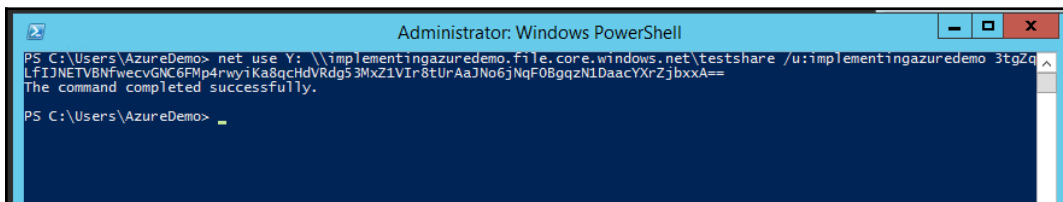
9. The Windows instructions are used on a Server 2012 R2 box to connect to the File share. The `net use` command is the traditional command-line tool to connect to network shares. It could also be done via PowerShell with the following line:

```
New-PSDrive -Name Y -PSProvider FileSystem -Root  
implementingazuredemo.file.core.windows.nettestshare -  
Credential (Get-Credential)
```

10. It's important to use the same credentials as with the `net use` command; the name of the storage account as username and the access key as password. The `net use` command looks like this:

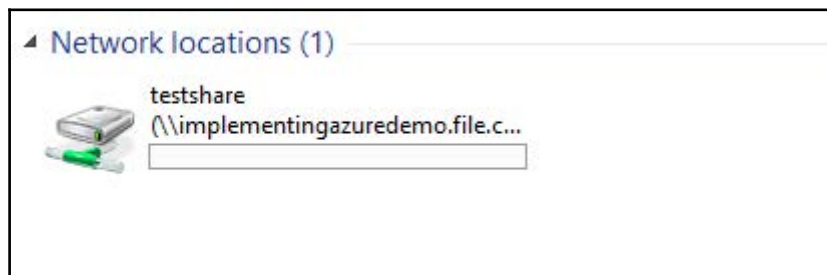
```
net use Y:  
implementingazuredemo.file.core.windows.nettestshare  
/u:implementingazuredemo [AccessKey]
```

11. [AccessKey] has to be replaced with one of the Access keys of the Storage account. The output should look similar to this:



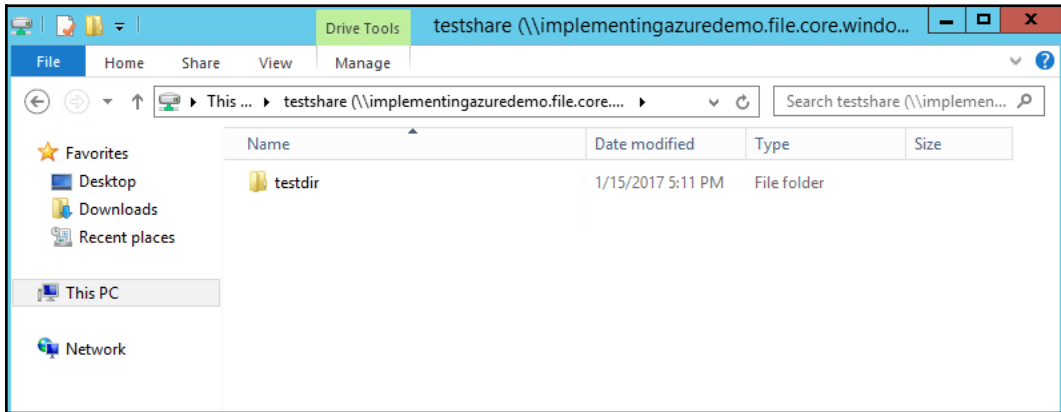
Using net use to connect to an SMB share

12. To verify the share connection, just open Windows Explorer and see if there is a new drive under network locations:



The connected SMB share in explorer

13. In the following screenshot the network shared drive `testshare` is illustrated:



The test directory in explorer

14. Now the File share can be used to store files of any kind.



To connect a share to a virtual machine, there are some requirements on SMB versions.

If a VM in the same Azure region as the Storage account is used, SMB 2.1 is supported.

If a VM in another Azure region as the Storage account is used, only SMB 3.0 or higher is supported.

If a local machine is used, only SMB 3.0 or higher is supported.

But there are also several downsides of the current File share implementation. Here are some important ones to consider:

- **Storage limit:** Additional to the 500 TB per Storage account and the limit of 200 Storage accounts per subscription, there are also limits on File shares. The maximum size of a File share is 5 TB and the maximum size per file in a File share is 1 TB. On the other hand, there is no limitation on the number of files in total, if you stick to the file and File share size.
- **File access:** Currently, there is nothing like an **Access Control List (ACL)** or rights management. It's only possible to allow or deny access to a File share.

The only ways to authenticate for File share access are Access keys and **shared access signatures (SAS)**. SAS are basically links that can be generated if someone needs limited access to a storage resource. They can be limited by time and storage service type (Queue, Table, Blob, File). Basic permissions such as read, write, delete, and so on can also be defined when generating an SAS. SAS should be used for untrusted external staff, temporary development test, or for customers for tests.

Access keys will be discussed in the next part.



Microsoft released a tool for working with Blob, File, and Table storage called **AzCopy**. Its main purpose is to transfer data from and to Azure Storage. It can be found at <http://aka.ms/azcopy>.

Access Keys

In Azure, storage Access keys are used to authenticate applications that use external or internal interfaces to interact with Azure Storage. Example interactions are a REST API call or a simple net use of an SMB share.

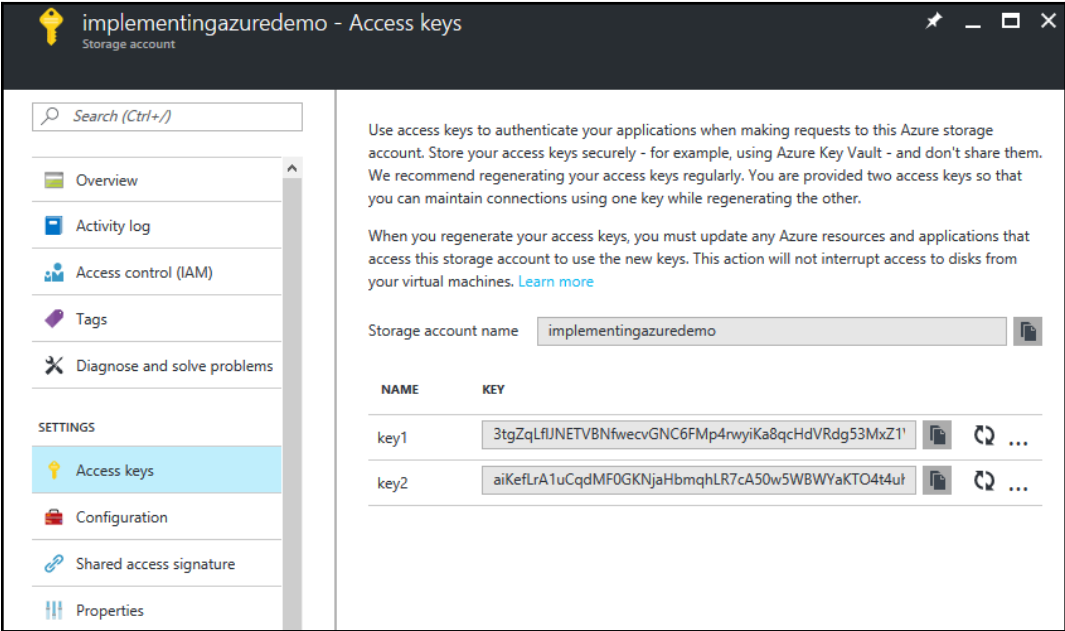
When a Storage account is created, Azure generates two 512 bit Access keys. These keys are very important to the security of the Storage account, for this reason, they must be kept safe all the time.

SAS is also created based on the Storage accounts Access keys. That means when the access key that a specific SAS is based on is regenerated, the SAS is invalid and has to be regenerated.

The reason that there are two Access keys in each Storage account is mainly high availability. As it's recommended, to regenerate the Access keys on a regular basis, keys should be rotated to avoid any downtime.

Key regeneration does not influence the access of your VMs to their VHDs.

The current Access keys of a Storage account can be found in the **Access keys** menu in a Storage account:

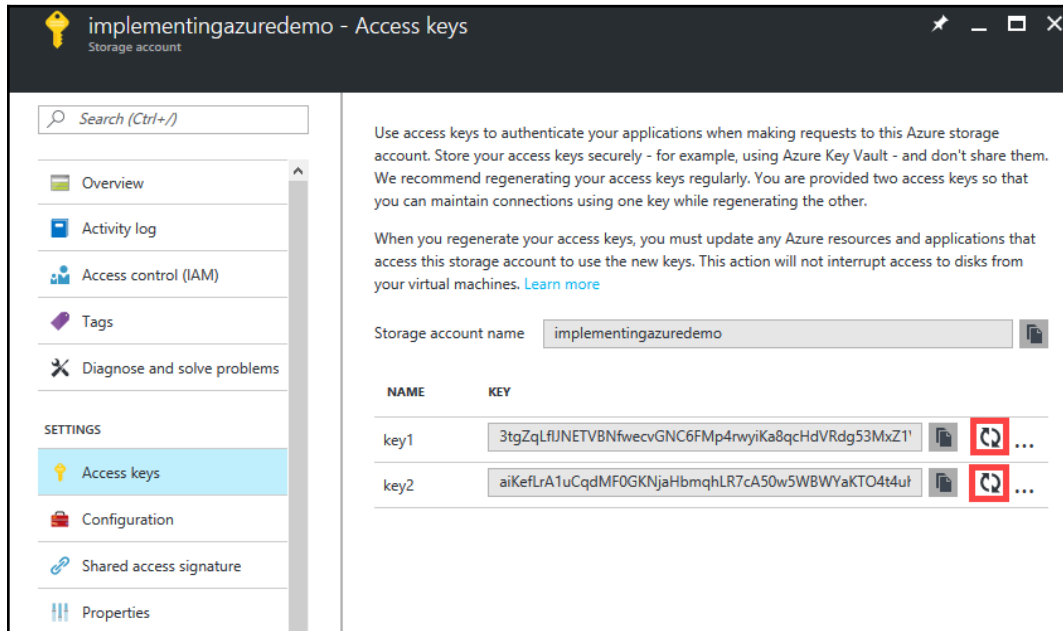


Storage account Access key overview

They can also be received with PowerShell:

```
Get-AzureRmStorageAccount `
  -name $storageAccountName `
  -ResourceGroupName $resourceGroupName `
  | Get-AzureRmStorageAccountKey
```

To regenerate a storage key, the regenerate button in the portal is used, as highlighted in the following diagram:



Regenerate keys with the marked buttons

To regenerate a key using PowerShell, the following cmdlet is used:

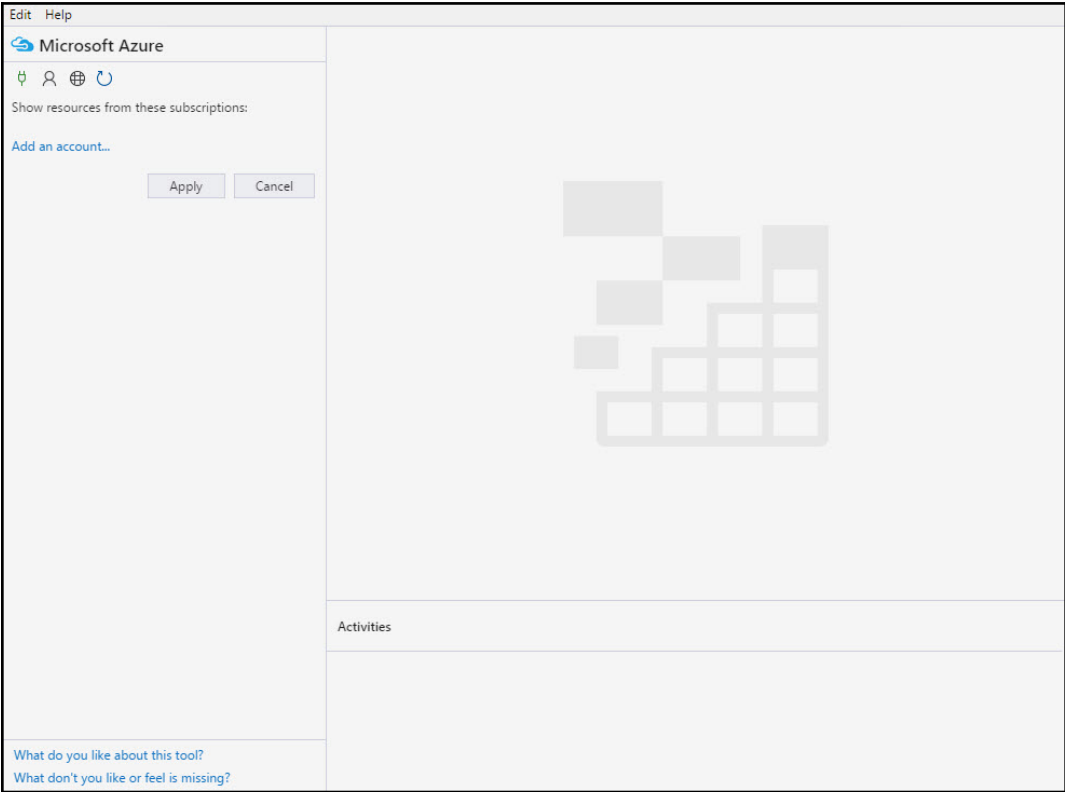
```
New-AzureRmStorageKey -ResourceGroupName "MyResourceGroup" -  
AccountName "MyStorageAccount" -KeyName "key1"
```

Exploring Azure Storage with Azure Storage Explorer

Microsoft Azure Storage Explorer is one of the free available graphical tools for managing Azure Storage without the portal or PowerShell.

Storage Explorer is still in preview and could have some bugs in some places. To start working with Azure Storage Explorer, it needs to be downloaded first. The current download link is <http://storageexplorer.com/>.

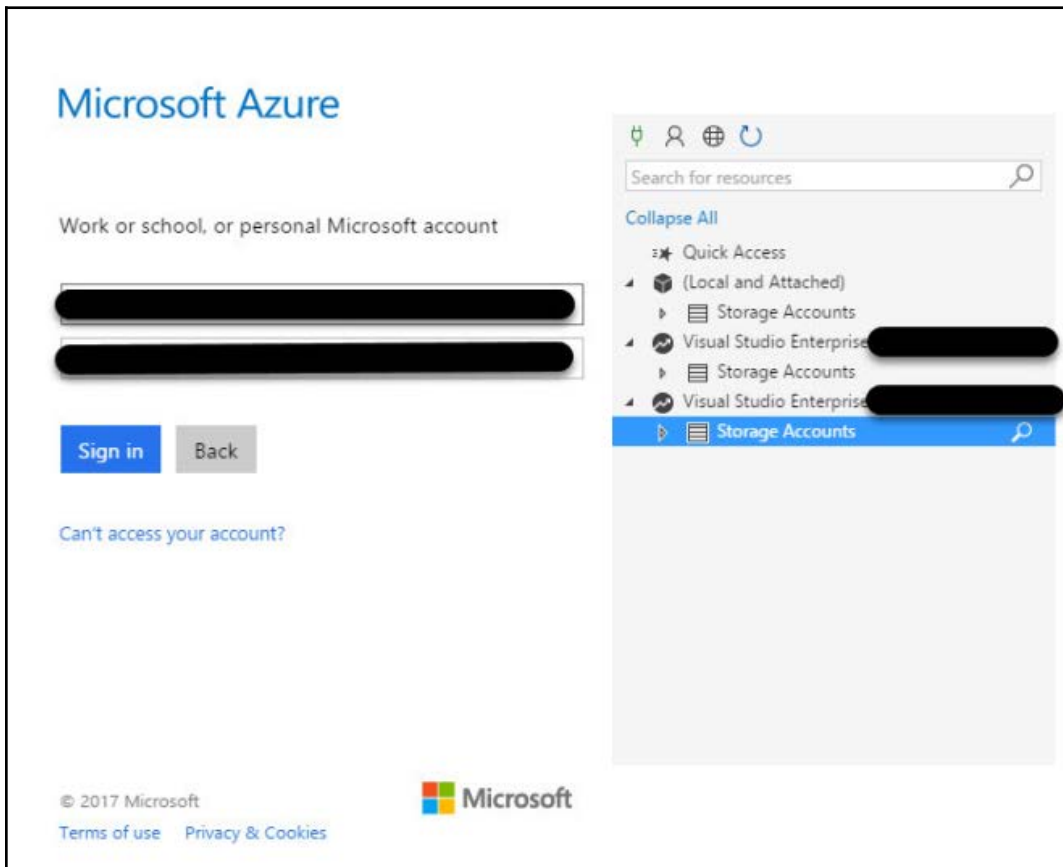
After downloading and installing, the following screen shows up:



Storage explorer dashboard

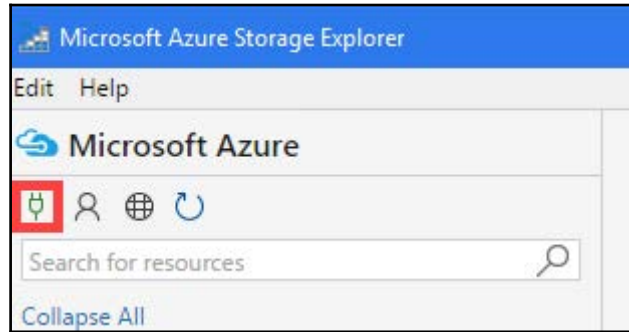
First, a Storage account needs to be connected. There are two possibilities how one can connect to a Storage account:

1. Add a Microsoft or company account. This is the easiest solution. Azure storage explorer asks for credentials of a Microsoft or company account that has an active subscription. After typing the username and password, Azure explorer has the possibility to select which subscriptions should be managed. By default, all are selected. After clicking on **Apply**, the setup is done, and Azure Storage nodes can be explored:



Left-Azure login dialogue, Right-Storage Explorer subscriptions overview

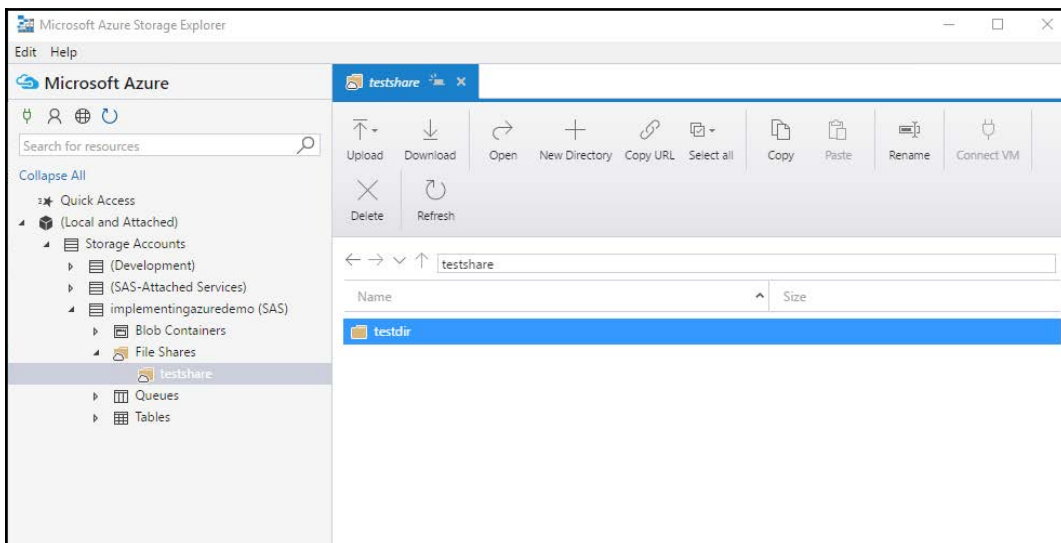
2. Add a connection string, SAS URI, or account. It's also possible to connect to your **Storage Accounts** by typing a Storage account key or SAS:



Connect with URI, SAS, or key button

The plug symbol opens the dialog, which is used to connect to Azure Storage by SAS, access key, or connection string.

3. After connecting to the Storage account, a node with the name of the added Storage account appears in Storage Explorer. Browsing through the **File Shares**, the earlier created File share can be found:



The connected Storage account node

Premium Storage Account

In the third figure at the beginning of this chapter, there is a performance property at Storage account creation. This property decides whether a standard or a premium Storage account is created. For most workloads, standard accounts are more than suitable but in some cases, more I/O intensive applications need very fast storage. In this case, the premium Storage account was introduced. Premium storage is fully backed by SSD tiers and provides high performance, low latency storage.

Premium storage can currently only be used for virtual disks used in VMs (page blobs). The performance property can't be changed after Storage account creation, but it's possible to migrate VMs from the standard to premium storage tier.

Depending on the machine size, it's possible to attach up to 64 disks to a VM (Standard_GS5). A Standard_GS5-sized machine supports up to 80,000 uncached IOPS and 2000 MB/s disk throughput.

Microsoft examples for enterprise applications that may need premium storage are Dynamics AX, Dynamics CRM, Exchange Server, SharePoint Farms, SAP Business Suite, SQL Server, Oracle, MongoDB, MySQL, and Redis.

Premium Storage Requirements

Premium storage supports DS-series, DSv2-series, GS-series, and Fs-series Azure virtual machines. Standard and premium VM disks can be attached to premium storage VMs. Premium storage disks cannot be used if the VM is not premium storage compatible.

Pricing

The billing for Azure Storage usage depends on the used Storage account. Also, storage costs are based on these determinants:

- **Location:** This describes the geographical region in which the account is based.
- **Storage capacity:** This refers to how much of the Storage account is used to store data.

- **Account type:** Based on the usage of either the general purpose Storage account or the Blob Storage account, the account type affects the billing. When using a Blob Storage account, the access tier also defines the billing model for the account.
- **Storage transactions:** Transactions are all read and write operations to Azure Storage.
- **Replication scheme:** This describes the number of copies of your data and where they are stored.
- **Data egress:** This is the data that is transported out of an Azure region. This usually happens when the data is obtained by an application that is located in another region or in an on-premises location. If data egresses Azure, charges apply.



For more information on pricing, please use the Azure pricing calculator, located at

<https://azure.microsoft.com/en-us/pricing/calculator/>.

Summary

Now you should have learned about the different types of storage that are available in Azure as well as which replication and availability options they have.

Also, we have created a Storage account and a few storage services in it. You should also know some basic tools and PowerShell cmdlets to interact with Azure Storage.

The next chapter will focus on Azure virtual machines.

6

Planning and Deploying Virtual Machines in Azure

If you want to run your services within Azure and Microsoft has no PaaS or SaaS offering for that specific service, it is necessary to implement virtual machines in your environment.

To identify the right virtual machine for your services, you need to know the virtual machines types that currently exist and how to implement a virtual machine within your Azure environment.

In this chapter, you will learn the basics of Azure virtual machines, how to implement virtual machines, and virtual machine license offerings from the Azure marketplace.

We are going to explore the following topics:

- Virtual machine types and their usage
- License offerings
- How to deploy a virtual machine
- How to change virtual machine settings
- Common virtual machine use cases

We will also setup a virtual machine during this chapter.

Azure Virtual Machine Types

As soon as you start working with virtual machines, you will notice that Microsoft offers lots of different virtual machines and is still expanding its offerings. The reason why Microsoft offers so many virtual machines types is easy to explain. Microsoft needs to support different kinds of workload. virtual machines are always the basement for every Microsoft service offered out of Azure or other Microsoft Cloud services. Virtual machines are also the core component for virtual appliances in Azure such as Hadoop, Microsoft Dynamics Navision, third-party network appliances, or open source appliances.

Microsoft currently offers the following types of virtual machines:

- **Basic A-series:** For testing and development.
- **Standard A-series:** All-round virtual machines for various workloads.
- **Compute intense A-series:** Designed for high performance computing.
- **D and DS-series:** For enterprise applications and applications for higher demand on compute power and temporary disk performance.
- **F and FS-series:** Optimized for network operation. They support workloads for virtualized network devices or applications with a high network demand such as web servers or real-time communication.
- **G and GS-series:** Built for enterprise applications with high compute demand such as databases, SharePoint, and NoSQL or big data calculations.
- **H-series:** Built for high performance computing.
- **N-series:** These virtual machines include an NVIDIA Tesla M60 and K80. They are built for VDI or GPU computing.
- *** LS-series:** Build for low latency storage demands above the offerings of GS-series.

As you can see, there are some virtual machine types which have an S within the name. Those types represent the storage. S-series virtual machines use solid state disks or SSDs and Azure premium storage as the primary storage type for data and operating systems. They should be chosen for regular storage performance over 500 IOPS.

Most of the virtual machine types are only made different by certain quality of service standards and priorities against other virtual machine types but there are also connections to their virtualization host. Microsoft mostly uses standard Intel Xeon processors. Microsoft standard is currently based on Intel Haswell architecture and will be upgraded to Skylake processor architecture.



When looking into this you will find an expression for the used CPUs in the Microsoft documentation at <https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-windows-sizes/>.

To get an idea which type of CPU is used by Microsoft to host your virtual machine, you should look up the virtual hardware. For example, the CPU which is used for your virtual machine. Microsoft updates the virtual machine series along with their hardware and CPU updates. To track these changes, it is recommended to refer to the Microsoft documentation.

A-series Virtual Machines

A-series virtual machines are divided into two categories. The first one is **basic** and the second is **standard**, which is also the class of all other virtual machine series.

A-series virtual machines are very common and Microsoft deploys them over a variety of hardware types and CPUs. The performance of an A-series virtual machine is throttled, based upon the hardware, to offer consistent processor performance for the running instance, regardless of the hardware.

The differences between Azure A-series basic and standard are:

- **Availability:** Basic tier VMs are only available in small A0-A4 instances for testing purpose, standard tier VMs are available on all size instances and regions.
- **Disk IOPS:** Data disk IOPS for basic tier VMs is up to 300, lower than standard tier VMs which have up to 500 IOPS for the data disk.
- **Price:** The price of a single tier VMs can be up to 27% less expensive than standard tier VMs.
- **Feature cut:** Basic tier VMs do not include load balancing or auto-scaling options. To achieve those features for a basic tier VMs, you need to add those to the availability set for high availability and implement your own load-balance mechanisms for example with Azure Load Balancer and on the application level.
- **CPU:** Standard tier VMs have better CPU performance than basic VMs.
- **Usage:** Basic tier VMs are used for testing and development while standard tier VMs are used for the production usage.

The standard tier is good for common usage such as Active Directory Domain Services, Federation Services, or other basic network and application services.

For basic tier VMs, the following performance and virtual machine types are available:

Size	CPU	Memory	HDD GB	Max disks	Max disk IOPS	Max NICs/Network bandwidth
A0	1	0.768 GB	20	1	1 x 300	1/low
A1	1	1.75 GB	40	2	2 x 300	1/moderate
A2	2	3.5 GB	60	4	4 x 300	1/moderate
A3	4	7 GB	120	8	8 x 300	2/high
A4	8	14 GB	240	16	16 x 500	4/high

For standard tier VMs, the following performance and virtual machine types are available:

Size	CPU	Memory	HDD GB	Max disks	Max disk IOPS	Max NICs/Network bandwidth
A0	1	0.768 GB	20	1	1 x 500	1/low
A1	1	1.75 GB	70	2	2 x 500	1/moderate
A2	2	3.5 GB	135	4	4 x 500	1/moderate
A3	4	7 GB	285	8	8 x 500	2/high
A4	8	14 GB	605	16	16 x 500	4/high
A5	2	14 GB	135	4	4 X 500	1/moderate
A6	4	28 GB	285	8	8 x 500	2/high
A7	8	56 GB	605	16	16 x 500	4/high



The A0 size is over-subscribed on the physical hardware. For this specific size, other customer deployments may impact the performance of your running workload. The relative performance is the expected baseline, subject to an approximate variability of 15%.

Within the A-series, there are some much bigger virtual machine types. The A8-A11 sizes are also known as compute-intensive instances. The hardware that runs these sizes is designed and optimized for compute-intensive and network-intensive applications such as databases, high-performance computer cluster software, or modeling, and simulations such as Autodesk or CATIA. The A8-A11 series uses Intel Xeon E5-2670 clocked at 2.6 GHZ and the H-series uses Intel Xeon E5-2667 v3 clocked at 3.2 GHz.

For standard compute-intensive instances tiers, the following performance and virtual machine types are available:

Size	CPU	Memory	HDD GB	Max disks	Max disk IOPS	Max NICs/Network bandwidth
A8	8	56	382	16	16 x 500	2/high
A9	16	112	382	16	16 x 500	4/very high
A10	8	56	382	16	16 x 500	2/high
A11	16	112	382	16	16 x 500	4/very high



All A-series from A8 to A11 **are remote direct memory access (RDMA) capable.**

D-series and DS-series Virtual Machines

D-series VMs are designed to run applications that demand higher compute power and temporary disk performance. To achieve this, a D-series VM provides a better processors performance with higher limits against A-series, a higher memory-to-core ratio, and a solid state drive for the temporary disk.

They are built for more modern operating systems such as Windows Server 2012 R2. These virtual machines can easily handle workloads from file servers, databases, applications, and web servers.

The following performance levels are available with D-series VMs:

Size	CPU	Memory	HDD GB	Max disks	Max disk IOPS	Max NICs/Network bandwidth
D1	1	3.5	50	2	2 x 500	1/moderate
D2	2	7	100	4	4 x 500	2/high
D3	4	14	200	8	8 x 500	4/high
D4	8	28	400	16	16 x 500	8/high
D11	2	14	100	4	4 x 500	2/high
D12	4	28	200	8	8 x 500	4/high
D13	8	56	400	16	16 x 500	8/high
D14	16	112	800	32	32 x 500	8/very high

The following performance levels are available with DS-series VMs:

Size	CPU	Memory	HDD GB	Max disks	Max disk IOPS	Max NICs/Network bandwidth
DS1	1	3.5	7	2	4000	1/moderate
DS2	2	7	14	4	8000	2/high
DS3	4	14	28	8	16000	4/high
DS4	8	28	56	16	32000	8/high
DS11	2	14	28	4	8000	2/high
DS12	4	28	56	8	16000	4/high
DS13	8	56	112	16	32000	8/high
DS14	16	112	224	32	64000	8/very high



Microsoft has already updated the D and DS-series to version 2 which are represented by Dv2 or DSv2. They feature more CPUs. The Dv2-series CPU is about 35% faster than the D-series CPU. It is based on the 2.4 GHz Intel Xeon® E5-2673 V3 (Haswell) processor, and with the Intel® Turbo Boost Technology 2.0, can go up to 3.1 GHz. The Dv2-series has the same memory and disk configurations as the D-series.

F-series and FS-series Virtual Machines

F-series is based on the 2.4 GHz Intel Xeon® E5-2673 v3 (Haswell) processor, which can achieve clock speeds as high as 3.1 GHz. This is the same CPU performance as the Dv2-series of VMs.

The F-series VMs are an excellent choice for workloads that demand faster CPUs but do not need as much memory or local SSD per CPU core. Workloads such as analytics, gaming servers, web servers, and batch processing will benefit from the value of the F-series.

Beginning with F2 or FS2 you will have two network cards within the virtual machine.

The following performance levels are available with F-series VMs:

Size	CPU	Memory	HDD GB	Max disks	Max disk IOPS	Max NICs/Network bandwidth
F1	1	2	16	2	2 x 500	1/moderate
F2	2	4	32	4	4 x 500	2/high
F4	4	8	64	8	8 x 500	4/high
F8	8	16	128	16	16 x 500	8/high
F16	16	32	256	32	32 x 500	8/extremely high

The following performance levels are available with FS-series VMs:

Size	CPU	Memory	HDD GB	Max disks	Max disk IOPS	Max NICs/Network bandwidth
F1s	1	2	16	2	4000	1/moderate
F2s	2	4	32	4	8000	2/high
F4s	4	8	64	8	16000	4/high
F8s	8	16	128	16	32000	8/high
F16s	16	32	256	32	64000	8/extremely high

G-series and GS-series Virtual Machines

G-series sizes are built to provide the most memory, the highest processing power, and the largest amount of local SSD of any virtual machine size offered by Azure. This extraordinary performance will allow the deploying of very large scale-up enterprise applications, such as large relational database servers (SQL Server, MySQL, and so on) and large NoSQL databases (MongoDB, Cloudera, Cassandra, and so on). G-series offers up to 32 vCPUs using the latest Intel® Xeon® processor E5 V3 family, 448 GB of memory, and 6.59 TB local SSD drives. Partly those VMs are running on dedicated physical hardware.

The following performance levels are available with G-series VMs:

Size	CPU	Memory	HDD GB	Max disks	Max disk IOPS	Max NICs/Network bandwidth
G1	2	28	384	4	4 x 500	1/high
G2	4	56	768	8	8 x 500	2/high
G3	8	112	1,536	16	16 x 500	4/very high
G4	16	224	3,072	32	32 x 500	8/extremely high
G5	32	448	6,144	64	64 x 500	8/extremely high

The following performance levels are available with GS-series VMs:

Size	CPU	Memory	HDD GB	Max disks	Max disk IOPS	Max NICs/Network bandwidth
GS1	2	28	384	4	10000	1/high
GS2	4	56	768	8	20000	2/high
GS3	8	112	1,536	16	40000	4/very high
GS4	16	224	3,072	32	80000	8/extremely high
GS5	32	448	6,144	64	160000	8/extremely high

H-series Virtual Machines

Azure H-series virtual machines are high-performance computing VMs used for high-end computational needs, such as molecular modeling, and computational fluid dynamics such as wave calculations. These 8 and 16 core VMs are built on the Intel Haswell E5-2667 V3 processor technology with DDR4 memory and local SSD based storage.

In addition to CPU power, the H-series offers diverse options for low latency RDMA networking using FDR InfiniBand and different memory configurations to support memory-intensive computational requirements.

The following performance levels are available with H-series VMs:

Size	CPU	Memory	HDD GB	Max disks	Max disk IOPS	Max NICs/Network bandwidth
H8	8	56	1000	16	16 x 500	2/high
H16	16	112	2000	32	32 x 500	4/very high
H8m	8	112	1000	16	16 x 500	2/high
H16m	16	224	2000	32	32 x 500	4/very high
H16r	16	112	2000	32	32 x 500	4/very high
H16mr	16	224	2000	32	32 x 500	4/very high



The virtual machines with size H16R and H16RM are RDMA capable.

NV-series and NC-series Virtual Machines

The NC and NV sizes are GPU-enabled instances. These are specialized virtual machines that include NVIDIA's GPU cards. The NV sizes are optimized and designed for remote visualization, streaming, gaming, encoding and VDI scenarios such as with Citrix utilizing frameworks (for example, OpenGL, and DirectX). The NC sizes are more optimized for compute-intensive and network-intensive applications and algorithms, including CUDA and OpenCL based applications and simulations.

NV Virtual Machines

The NV instances are built with NVIDIA Tesla M60 GPUs and NVIDIA GRID for desktop accelerated applications and virtual desktops where customers are able to visualize their data or simulations. Users will be able to visualize their graphics-intensive workflows on the NV instances to get superior graphics capability and additionally run single-precision workloads such as encoding and rendering. The Tesla M60 delivers 4,096 CUDA cores in a dual-GPU design with up to 36 streams of 1080p H.264.

The following performance levels are available with FS-series VMs:

Size	CPU	Memory	HDD GB	GPU
NV6	6	56	380	1 x NVIDIA M60
NV12	12	112	680	2 x NVIDIA M60
NV24	24	224	1440	4 x NVIDIA M60

NC Virtual Machines

The NC instances are built with NVIDIA Tesla K80. Users can crunch through data faster by leveraging CUDA for energy exploration applications, crash simulations, ray-traced rendering, deep learning, and more. The Tesla K80 delivers 4,992 CUDA cores with a dual-GPU design, up to 2.91 Teraflops of double-precision, and up to 8.93 Teraflops of single-precision performance.

The following performance levels are available with FS-series VMs:

Size	CPU	Memory	HDD GB	GPU
NC6	6	56	380	1 x NVIDIA K80
NC12	12	112	680	2 x NVIDIA K80
NC24	24	224	1440	4 x NVIDIA K80

Ls-series Virtual Machines

The Ls-series is built for workloads that require low latency local storage with high demand on IOPS, like NoSQL databases or virtualized Windows Server Storage Spaces Direct Servers. The Ls-series offers up to 32 CPU cores, using the Intel® Xeon® processor E5 v3 family. This is the same CPU performance as the G/GS-series and comes with 8 GB of memory per CPU core.

The following performance levels are available with Ls-series VMs:

Size	CPU	Memory	HDD GB	Max disks	Max disk IOPS	Max NICs/Network bandwidth
Standard_L4s	4	32	678	8	5000	2/high
Standard_L8s	8	64	1388	16	10000	4/very high
Standard_L16s	16	128	2807	32	20000	8/extremely high
Standard_L32s	32	256	5630	64	40000	8/extremely high

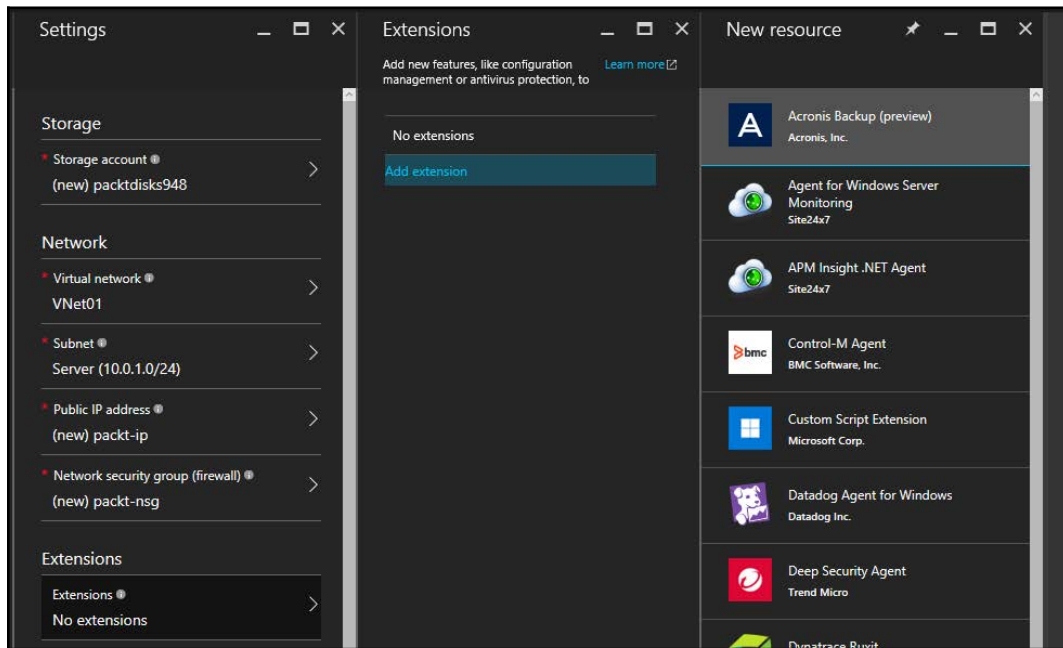


The virtual machines with size `Standard_L32s` run on isolated and dedicated hardware.

Virtual Machine Extensions

Together with partners and out of its own portfolio, Microsoft offers a wide range of extensions for virtual machines. Those extensions range from simple anti-malware solutions, backup, to deployment extension for desired state configuration to open source plugins such as for Chef server or Puppet.

The following screenshot shows some of the **Extensions**:



The number of **Extensions** for virtual machines is rapidly growing. To get a full and accurate list, you need to run the following PowerShell command against Microsoft Azure:

```
Get-AzureVMAvailableExtension | Select ExtensionName, Version
```

The virtual machine extensions are already delivered with a license for the product and the cost for that license will be calculated together with your virtual machine costs. Even with the license not purchased by yourself, you can include those extensions into your central management; for example, when it comes to anti-malware, and so on.



If you are using, for example, anti-malware software which is also offered in Azure, it could be more cost-effective to use the virtual machine extension license instead of buying a new one.

Managed Disks

With the progress, Microsoft is changing and evolving the Azure environment, Microsoft recently added a new option to virtual machines in Azure.

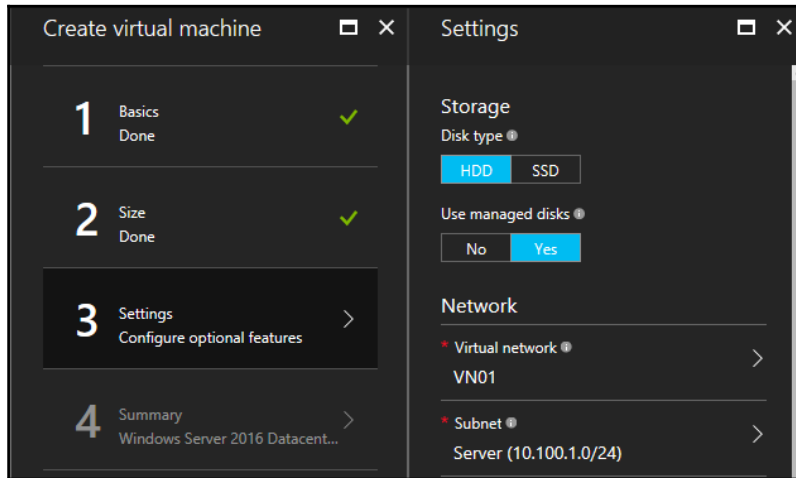
The option is called **Managed Disks**. These disks are abstracted from the Storage account and Storage account limitations.

You only have to specify the type which can be standard or premium storage and the size of the disk you need, and Azure creates and manages the disk.

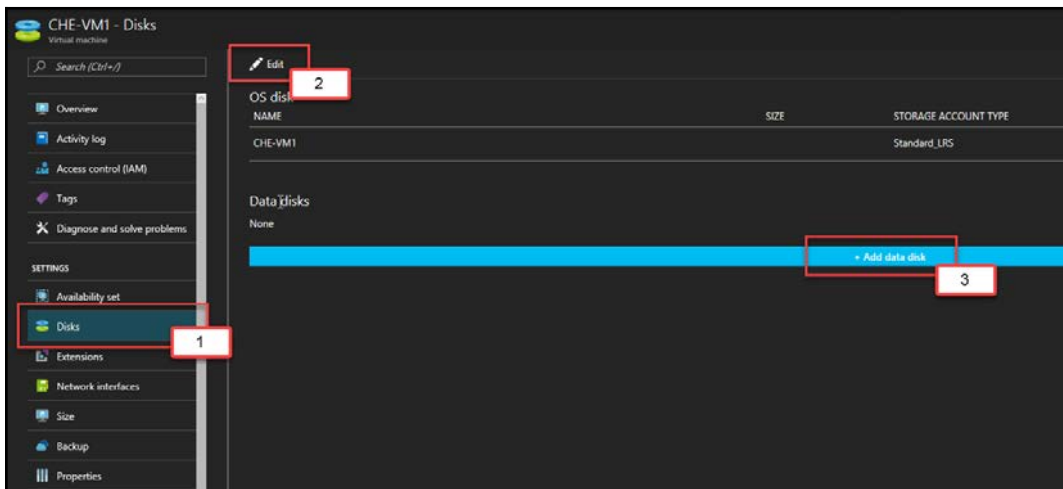
Currently, Microsoft offers you following types of Managed Disks:

Disk type	Disk name	Disk size
Premium	P10	128 GB
Premium	P20	512 GB
Premium	P30	1024 GB
Standard	S4	32 GB
Standard	S6	64 GB
Standard	S10	128 GB
Standard	S20	512 GB
Standard	S30	1024 GB

To create a virtual machine with Managed Disks, you can do so during the deployment process, we will go through later in the chapter.



If you want to manage the disks of a virtual machine, you now can find a configuration blade within the virtual machine settings where you can **Edit** or **Add data disk** the managed disks of the VM.





If you are currently working with VM Disks on Storage accounts and want to migrate to Managed Disk, Microsoft offers a detailed guide on the Azure Documentation Website <https://docs.microsoft.com/en-us/azure/virtual-machines/windows/migrate-to-managed-disks>.

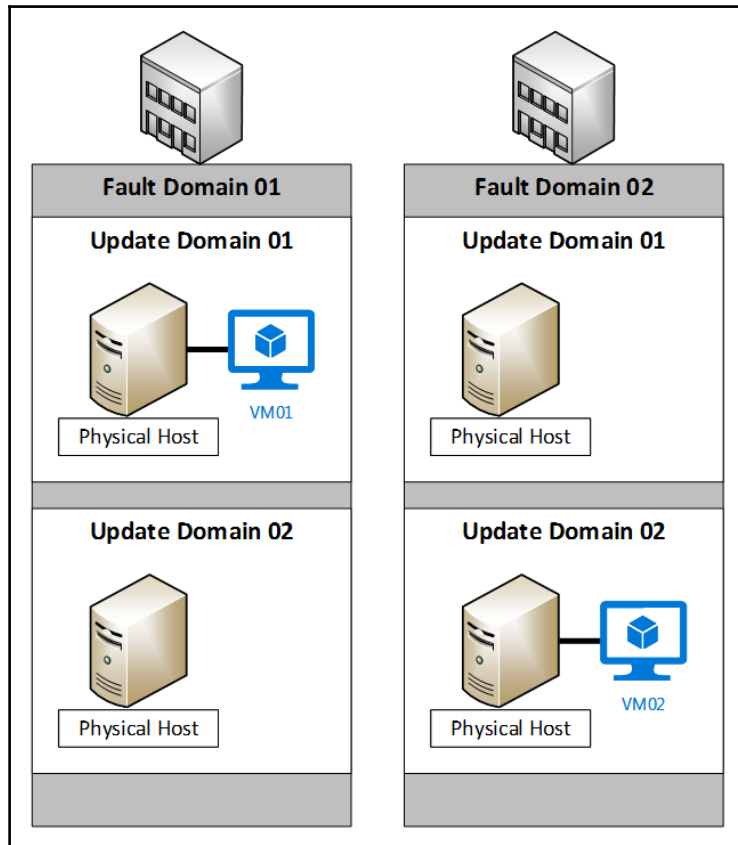
Availability Sets

Availability sets are a basic way to let Microsoft Azure know that these two virtual machines belong to a cluster or application group and are now allowed to go down together.

For virtual machines within a availability Azure manages that these machines run within different fault and update domains of an Azure region:

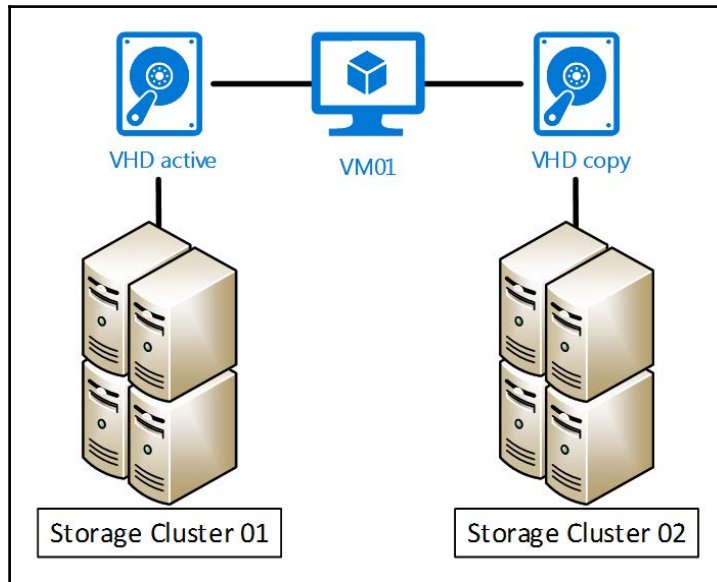
- **Update domain:** Update domain in Azure means, that all physical servers in on update domain will get host updates like firmware, drivers and OS updates at the same time.
- **Fault domain:** Fault domain in Azure means that all servers in these domains run in the same fire sections, with same air condition or electrical source. Which means all physical servers within in those domains, can have an outage at the same time.

The following diagram a schematic view on the distribution of a VM within a availability set:



With Azure virtual machine Managed Disks, there is an additional option which comes with availability sets. Azure takes also care that Managed Disks for those VMs within an availability set are placed in different Storage Cluster of the Azure Region.

The following diagram illustrates a simplified how those disks are distributed among the storage clusters:

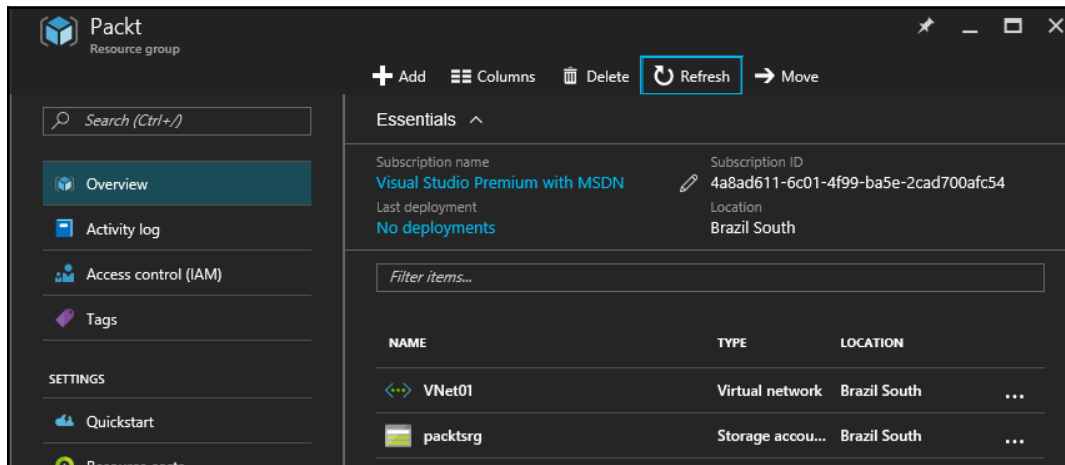


Deploying a Virtual Machine in Azure

Within this part of the chapter, we will deploy a virtual machine by using the Azure Resource Manager portal. When you followed the guidance in the chapters before, you should have the following things already configured:

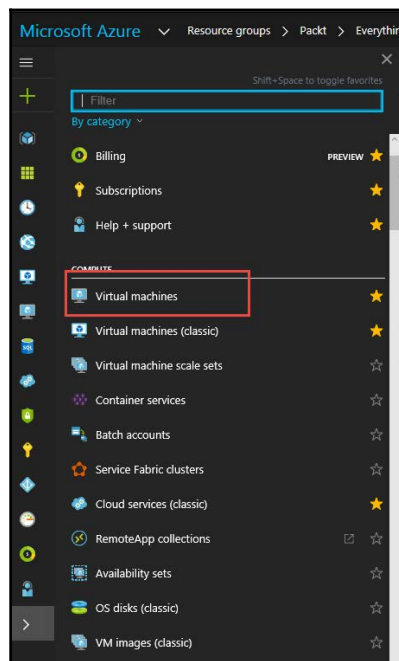
- Resource groups
- Virtual networks
- Azure storage

These parts are also the basics for deploying an Azure virtual machine:

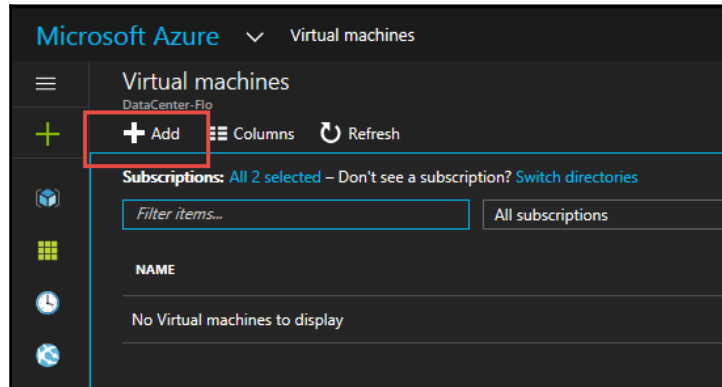


If you have a Resource groups, network and storage ready, you can start to deploy a virtual machine.

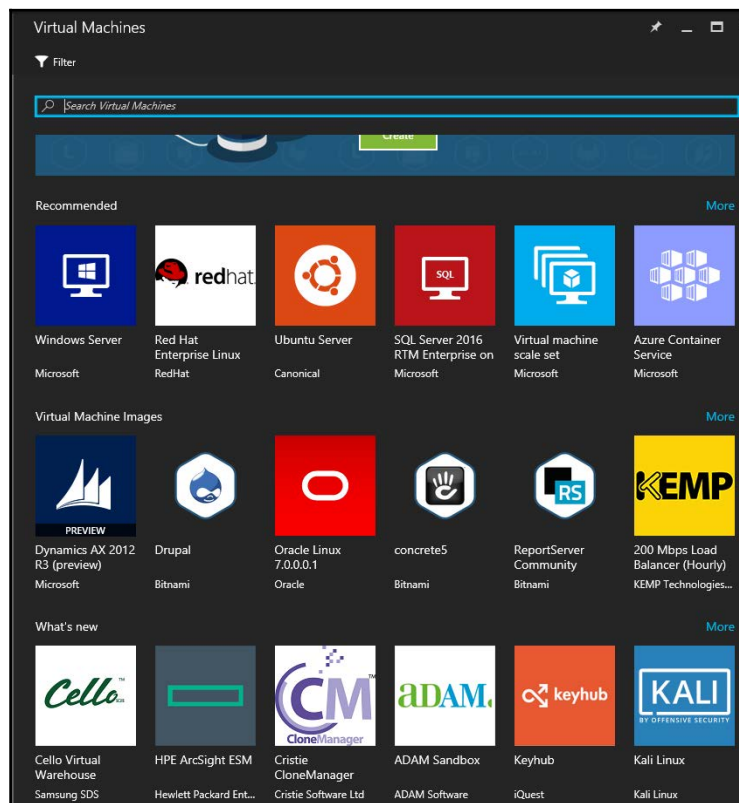
1. First you open the side bar and select **Virtual machines**:



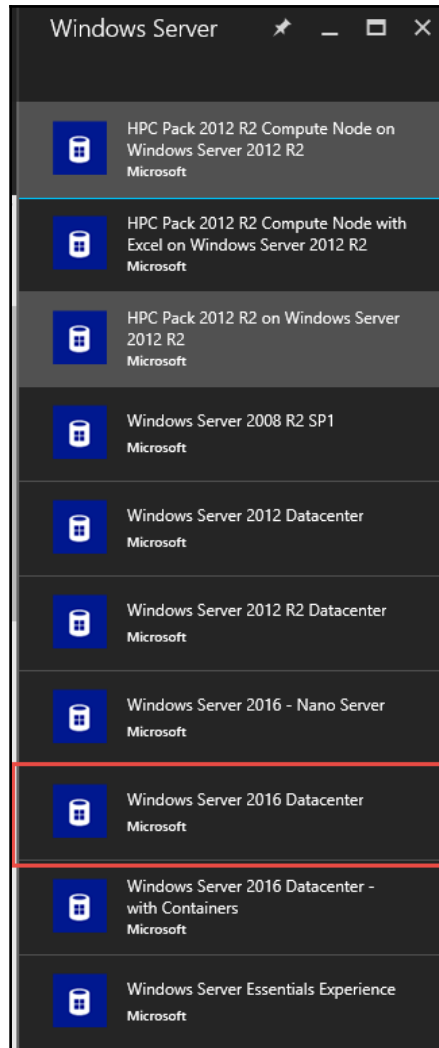
2. Within the new window, click on the **Add** button:



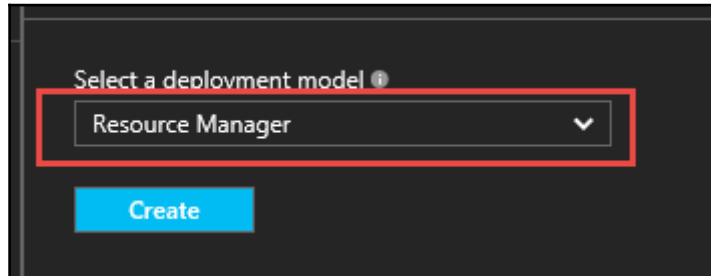
3. Afterwards, you need to select the image and operating system you want to deploy. In the current scenario, we will select **Windows Server**:



4. Select the version you want to deploy. In this scenario, we will select **Windows Server 2016 Datacenter**:



5. Select the **deployment model** as **Resource Manager** from the drop-down menu. It is not recommended to use **Classic** any more. Click on the **Create** button to start the VM configuration blade:



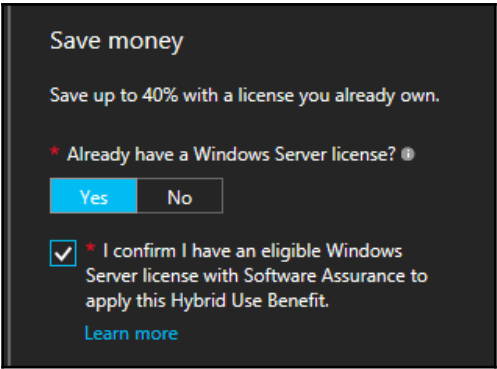
6. Within the basics configuration, you need to set the following things:
 - **Name:** The name of your virtual machine.
 - **VM disk type:** The disk type you want to use, which is standard storage or premium storage. Depending on this decision, the site and offered VM type will change within the next configuration.
 - **User name:** The name of your local administrator. The username can't be *admin* or *administrator*.
 - **Password** and **Confirm password:** The password you want to use for the local admin.
 - **Subscription:** The subscription you want to use for the VM.
 - **Resource group:** The resource group you want to deploy the VM to.

- **Location:** The region you want to deploy to.

The screenshot shows the 'Create virtual machine' wizard in the Azure portal, specifically the 'Basics' step. The left sidebar contains four steps: 1. Basics (selected), 2. Size, 3. Settings, and 4. Summary. The main panel displays the following configuration options:

- Name:** A text input field with a green checkmark.
- VM disk type:** A dropdown menu set to 'SSD'.
- User name:** A text input field with a green checkmark.
- Password:** A password input field with a green checkmark.
- Confirm password:** A password input field with a green checkmark.
- Subscription:** A dropdown menu set to 'Visual Studio Enterprise'.
- Resource group:** Radio buttons for 'Create new' (selected) and 'Use existing', followed by a dropdown menu set to 'RG-DCF-Demo'.
- Location:** A dropdown menu set to 'West Europe'.

7. Since Microsoft starts support **bring your own license (BYOL)** for Windows server VMs in Azure, there came new option available when deploying VM. You now can use your Windows server license with software assurance to active your Azure VM. Afterwards Windows VMs will be priced up to 40% less:



8. After you have configured the basic configuration of the virtual machine, you need to follow up with the sizing. Microsoft shows you recommendations in relation to the operating systems you chose before:

Create virtual machine

1 Basics
Done

2 Size
Choose virtual machine size

3 Settings
Configure optional features

4 Summary
Windows Server 2016 Datacent...

Choose a size

Browse the available sizes and their features

Prices presented are estimates in your local currency that include only Azure infrastructure costs and any discounts for the subscription and location. The prices don't include any applicable software costs. Recommended sizes are determined by the publisher of the selected image based on hardware and software requirements.

Supported disk type: HDD | Minimum memory (GiB): 0 | Minimum cores: 1

★ Recommended | [View all](#)

DS1_V2	Standard	★	DS2_V2	Standard	★	DS11_V2	Standard	★
1	Core		2	Cores		2	Cores	
3.5	GB		7	GB		14	GB	
2	Data disks		4	Data disks		4	Data disks	
3200	Max IOPS		6400	Max IOPS		6400	Max IOPS	
7 GB	Local SSD		14 GB	Local SSD		28 GB	Local SSD	
Load balancing			Load balancing			Load balancing		
Premium disk support			Premium disk support			Premium disk support		
42,66	EUR/MONTH (ESTIMATED)		85,33	EUR/MONTH (ESTIMATED)		119,21	EUR/MONTH (ESTIMATED)	

9. Sometimes it is necessary to change the VM type because the application you want to run needs more resources or you have some services running which need less resources. To select other VM types, you need to click **View all** to see all possible VM types:

Create virtual machine

1 Basics
Done

2 Size
Choose virtual machine size

3 Settings
Configure optional features

4 Summary
Windows Server 2016 Datacent...

Choose a size

Browse the available sizes and their features

Prices presented are estimates in your local currency that include only Azure infrastructure costs and any discounts for the subscription and location. The prices don't include any applicable software costs. Recommended sizes are determined by the publisher of the selected image based on hardware and software requirements.

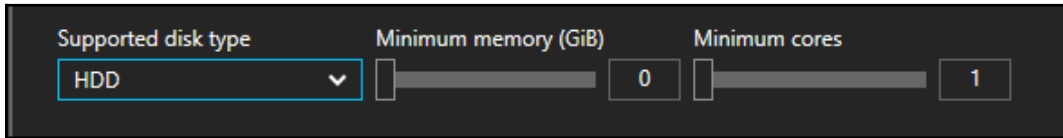
Supported disk type: SSD Minimum memory (GiB): 0 Minimum cores: 1

★ Recommended | View all

DS1_V2 Standard ★	DS2_V2 Standard ★	DS3_V2 Standard
1 Cores	2 Cores	4 Cores
3.5 GB	7 GB	14 GB
2 Data disks	4 Data disks	8 Data disks
3200 Max IOPS	6400 Max IOPS	12800 Max IOPS
7 GB Local SSD	14 GB Local SSD	28 GB Local SSD
Load balancing	Load balancing	Load balancing
Premium disk support	Premium disk support	Premium disk support
42,66 EUR/MONTH (ESTIMATED)	85,33 EUR/MONTH (ESTIMATED)	170,66 EUR/MONTH (ESTIMATED)
DS4_V2 Standard	DS5_V2 Standard	DS11_V2 Standard ★
8 Cores	16 Cores	2 Cores
28 GB	56 GB	14 GB
16 Data disks	32 Data disks	4 Data disks
25600 Max IOPS	51200 Max IOPS	6400 Max IOPS
56 GB Local SSD	112 GB Local SSD	28 GB Local SSD
Load balancing	Load balancing	Load balancing
Premium disk support	Premium disk support	Premium disk support

Select

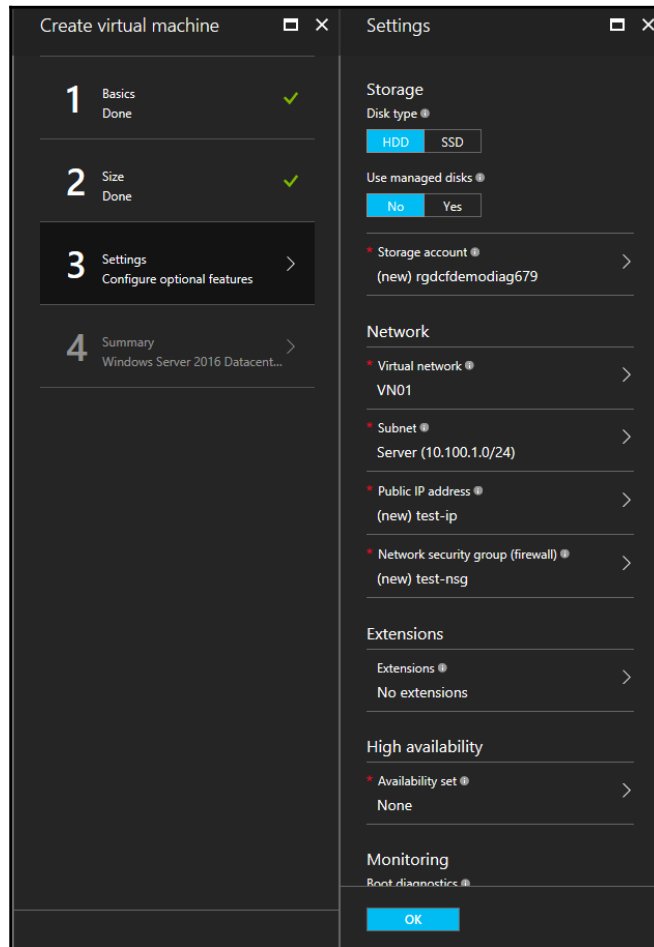
Since the amount of VMs and VM series increased tremendously, Microsoft starts to add filters to support customers to find their right VM series and size. One of the first implemented filter is shown below:



The image shows a dark-themed UI for configuring VM filters. It contains three main sections: 'Supported disk type' with a dropdown menu showing 'HDD', 'Minimum memory (GiB)' with a slider and a numeric input box showing '0', and 'Minimum cores' with a slider and a numeric input box showing '1'.

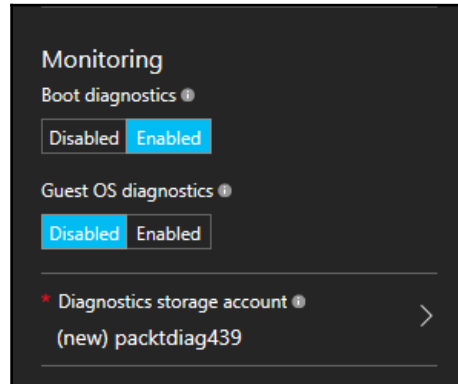
1. After you selected the VM size and type, you need to do the detailed setting. Therefore, we configure the following settings:
 - **Storage account:** Storage of your virtual machine. Here you have the chance to decide if you want to deploy the VM into a Storage account or with Managed Disk
 - **Network:** Private virtual network where you want to deploy your virtual machine
 - **Subnet:** The subnet where you want to deploy your virtual machine
 - **Public IP address:** Here you can configure a public IP for your virtual machine if needed
 - **Network security group (firewall):** Here you configure the firewall setting of your virtual machine
 - **Extensions:** Here you add the extensions such as antivirus agents or deployment agents to your virtual machines

- **Availability set:** Here you can add dependencies to other VMs if you need to

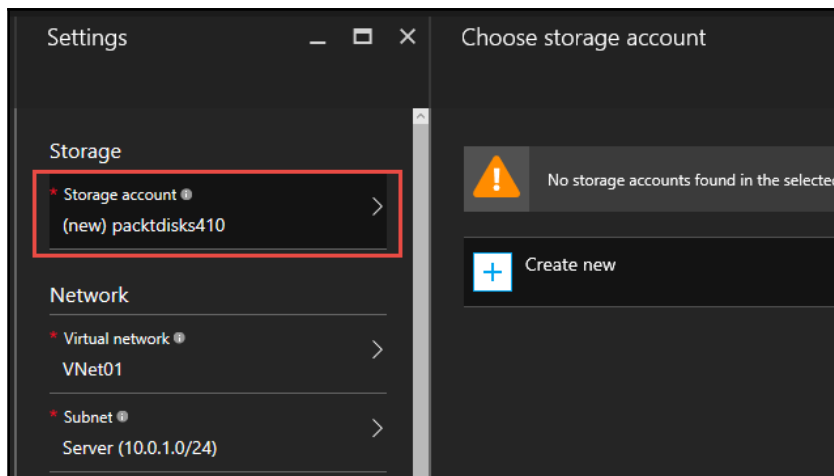


2. Additionally, you can configure monitoring for the virtual machine. The following options can be configured:
 - **Boot diagnostics:** Here you give Azure the option to monitor and save the boot of your virtual machine
 - **Guest OS diagnostics:** Here you give Azure the option to monitor and save the operating system diagnostic data of your virtual machine

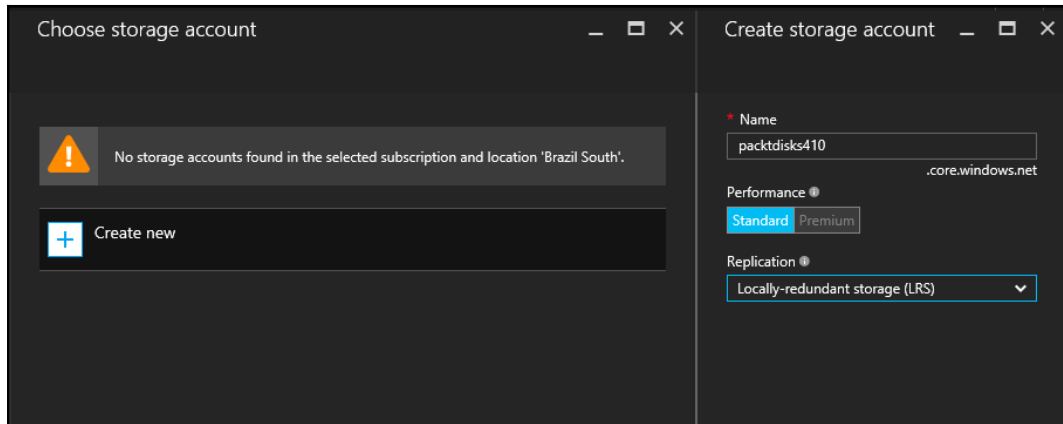
- **Diagnostics storage account:** The storage location of the monitoring log files



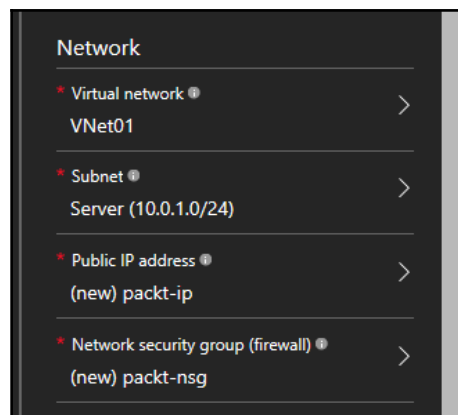
3. Now let us look a bit deeper into the single configuration options.
4. First, we start with the storage. As explained before, now you have the option to configure the storage of our virtual machine. You can decide between Storage account and Manage Disk. In our scenario, we will proceed with deploying a Storage account. Currently Storage account deployments are still the most common and currently the complicate type of VM storage.
5. To deploy the storage, click on the **Storage account** to open the configuration blade for storage:



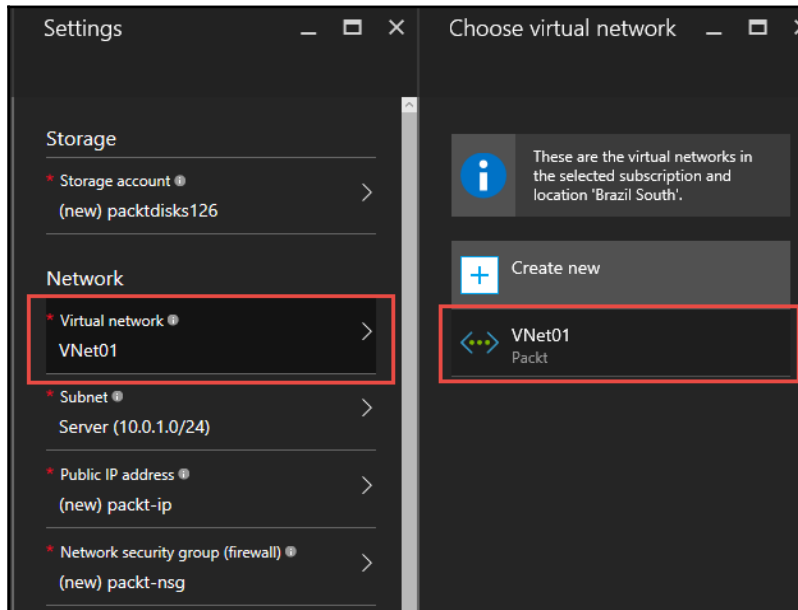
6. If you have already one which fits the type of your virtual machine, you can choose it. Otherwise you need to create a new one:



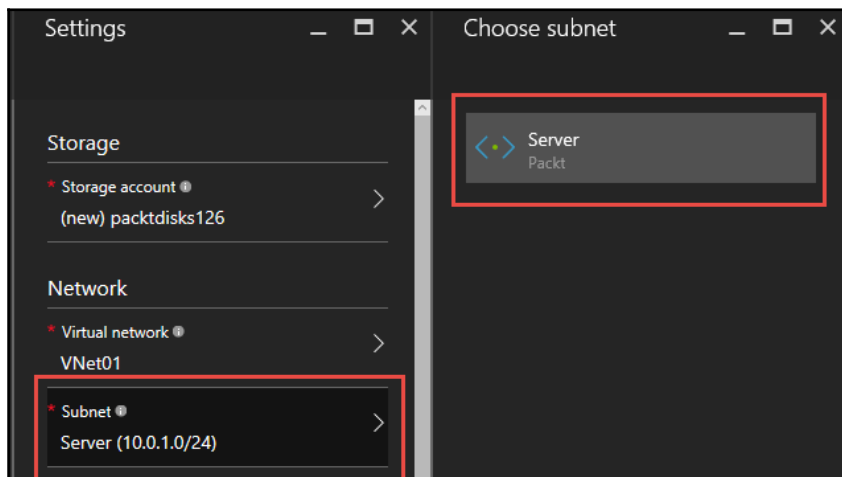
7. After you selected or create the storage disk, you move on and configure the **Network**. Therefore you need to do two options. The first option you configure is the private Azure network and subnet. The second option is only necessary if you do not have any access to your virtual machine via VPN or ExpressRoute. Here you configure the virtual machine with a public IP:



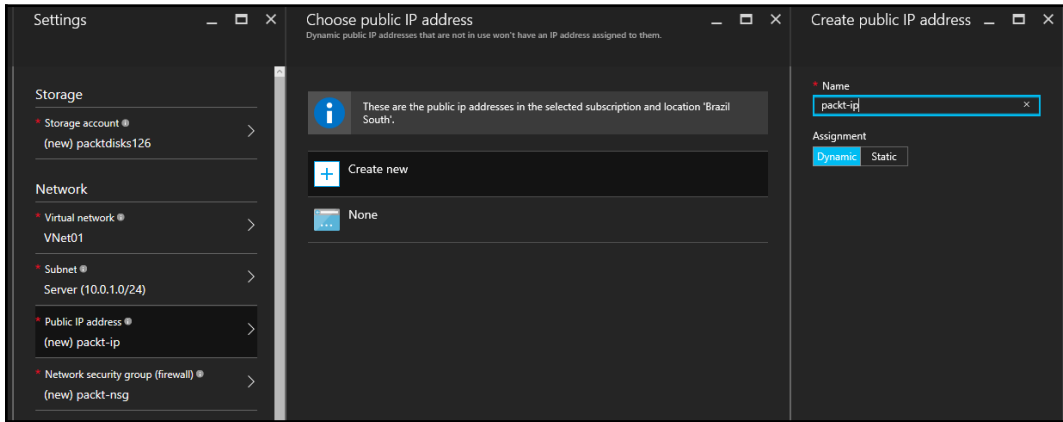
8. If you have already created a virtual network in Azure, it will be listed within the selection. If you have more than one network, you choose in which one the Azure VM will be deployed:



9. After you selected the **Virtual network**, you select the **Subnet**:

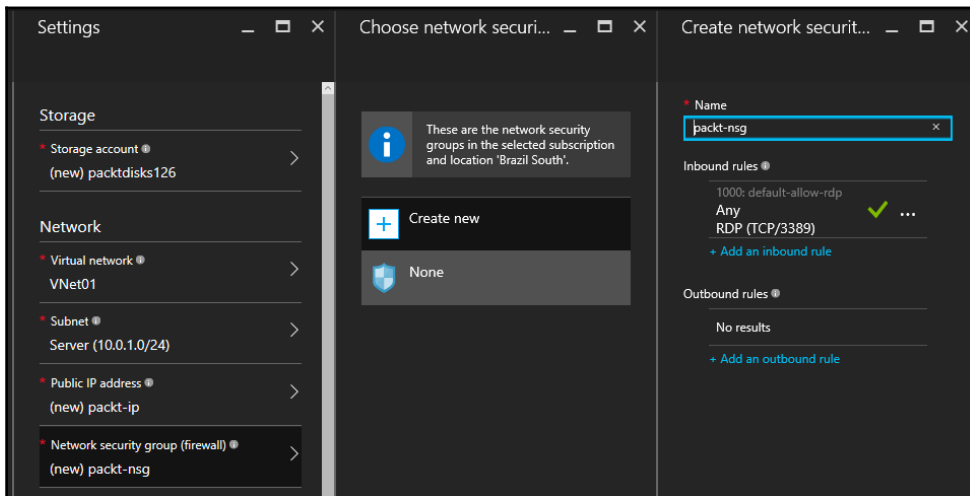


10. Afterwards you create or remove the public IP for the virtual machine:

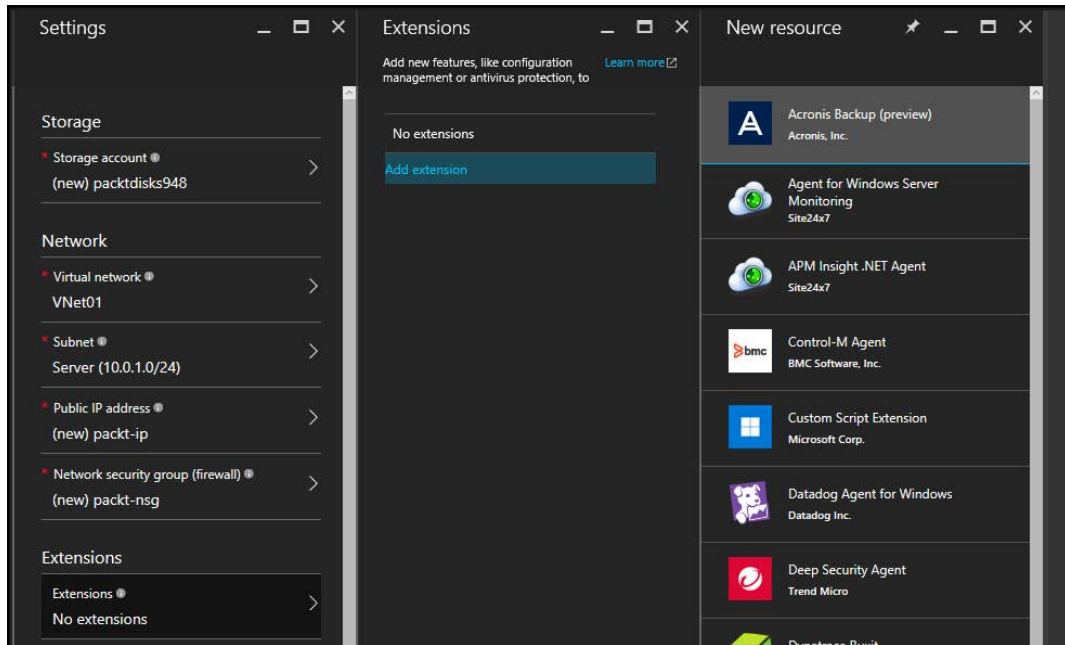


You should only create a public IP for your virtual machine if it is an Internet-facing server or virtual device or you have no other option to access the virtual machine. Otherwise, you should click on **None** to remove the public IP.

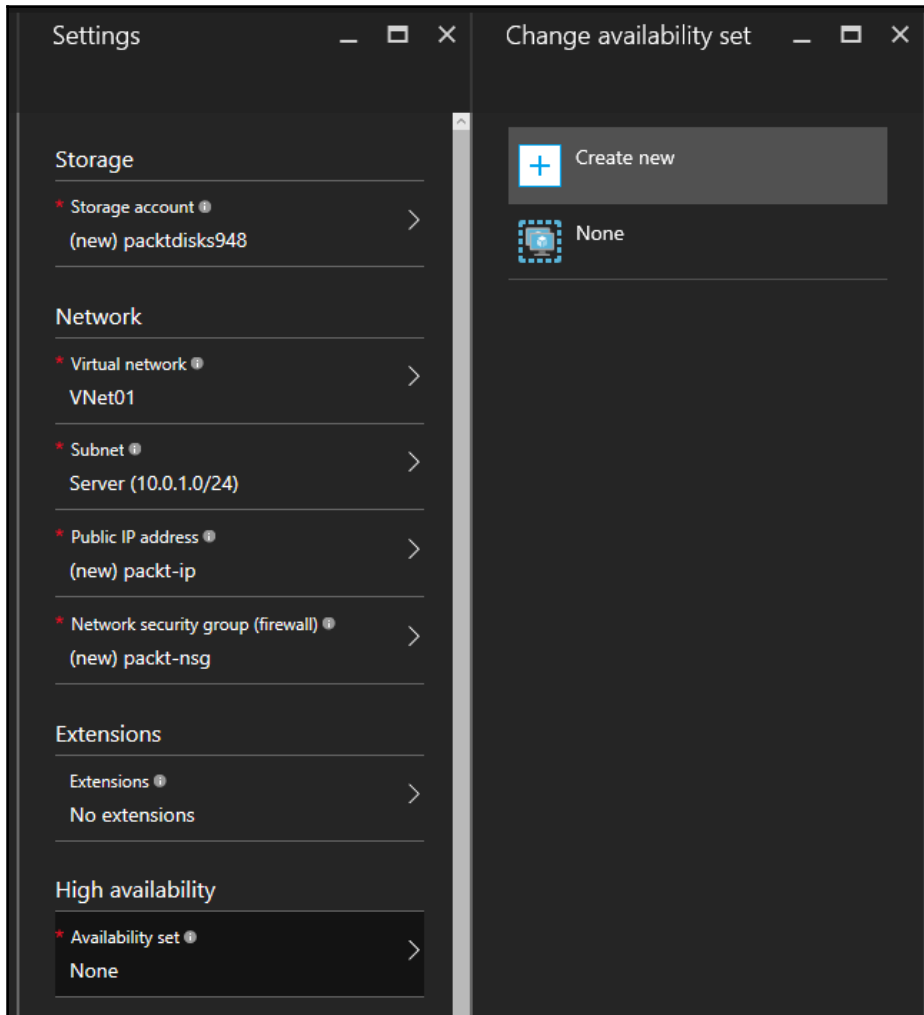
11. In the next step, you can configure the firewall of your virtual machine. Normally RDP (for Windows VMs) and SSH (for Linux) is already enabled. You shouldn't remove the default settings as soon as you have another way to connect to the virtual machine:



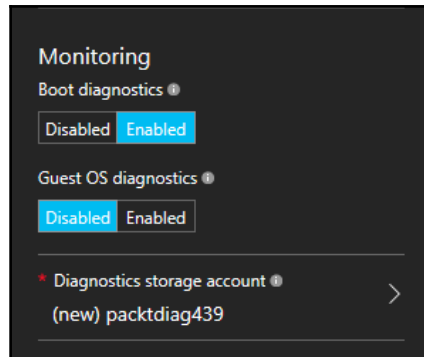
12. After you have configured the firewall, you can add virtual machine **Extensions** such as security software or deployment engines:



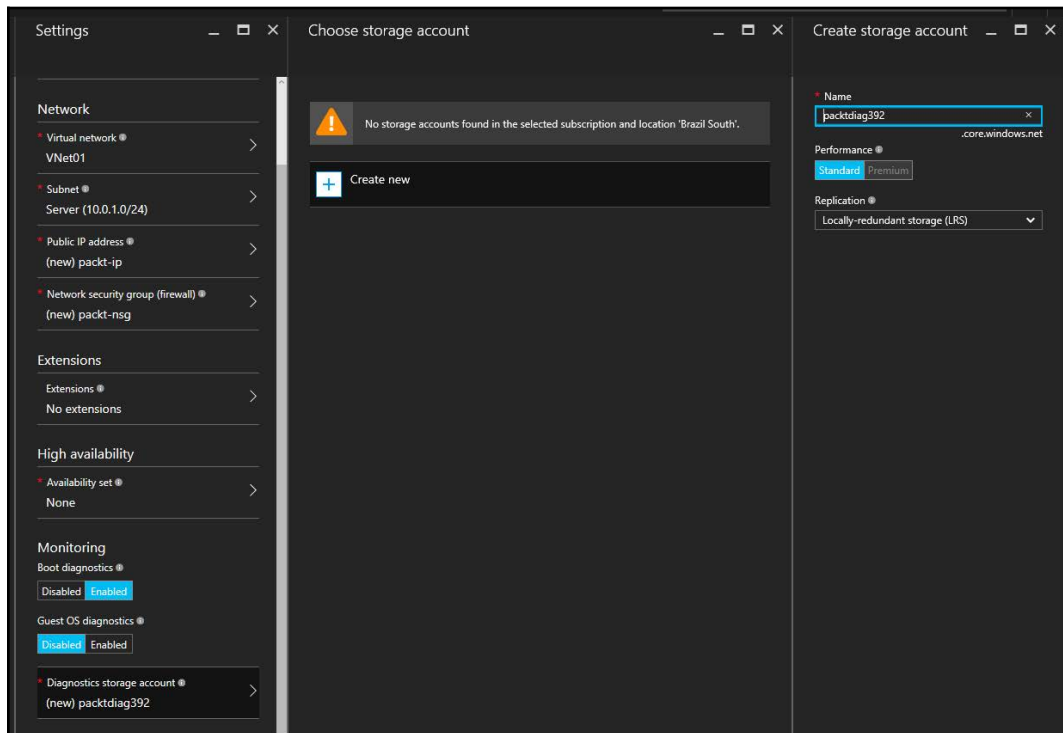
13. In the next step, you can configure availability sets and dependencies between other virtual machines. That supports **High availability** scenarios in Azure, for example when you run clusters or systems which depend on each other:



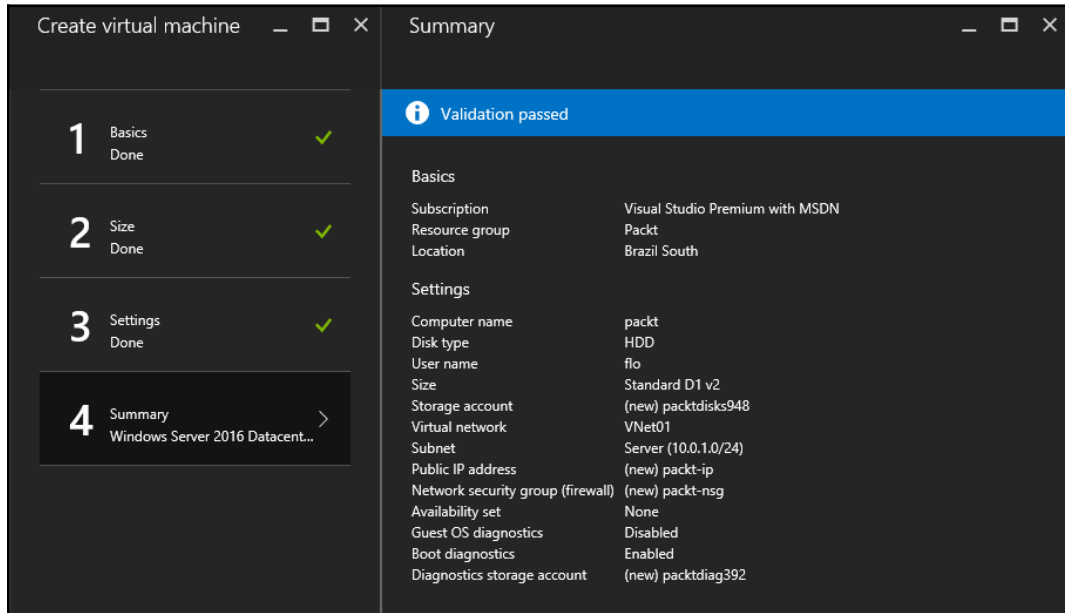
14. Finally, you configure the **Monitoring**. Therefore, you decide what to monitor:



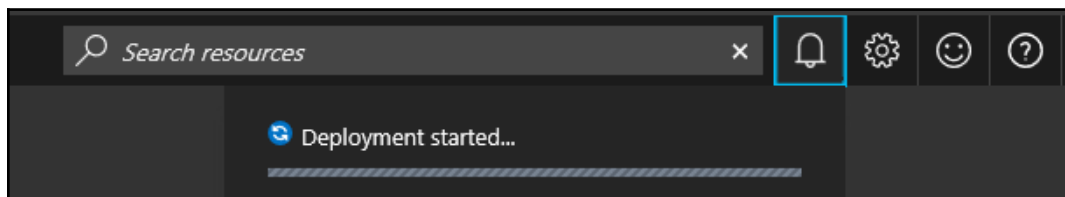
15. As soon as you have **Enabled** or **Disabled** the options you need, you create another Storage account for the diagnostic data. That works the same as the Storage account you already created in the previous chapter or for the virtual machine storage:



16. At the end, you get a **Summary** for your virtual machine. Click on **Create**:



17. Now the creation starts:

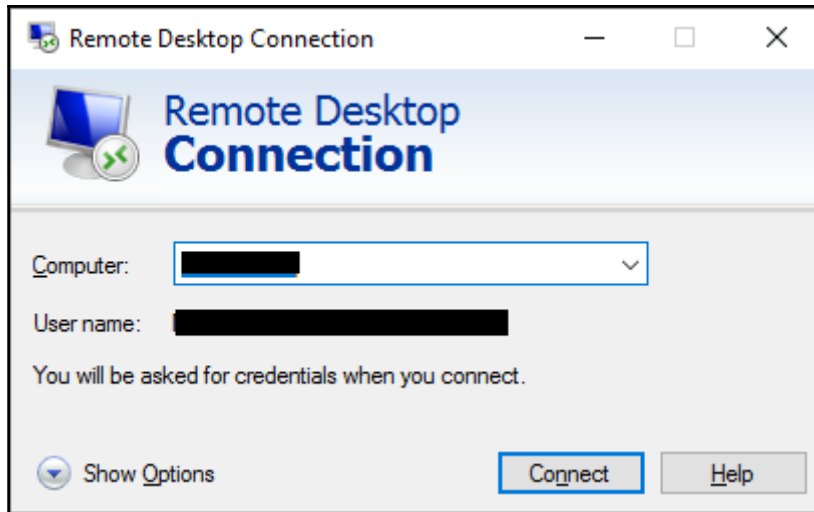


The time needed for deployment depends on your virtual machine size and type. Smaller virtual machines will take longer to be deployed. Every machine in Azure will be installed and configured from scratch. Azure performs no image based deployment such as System Center Virtual Machine Manager.

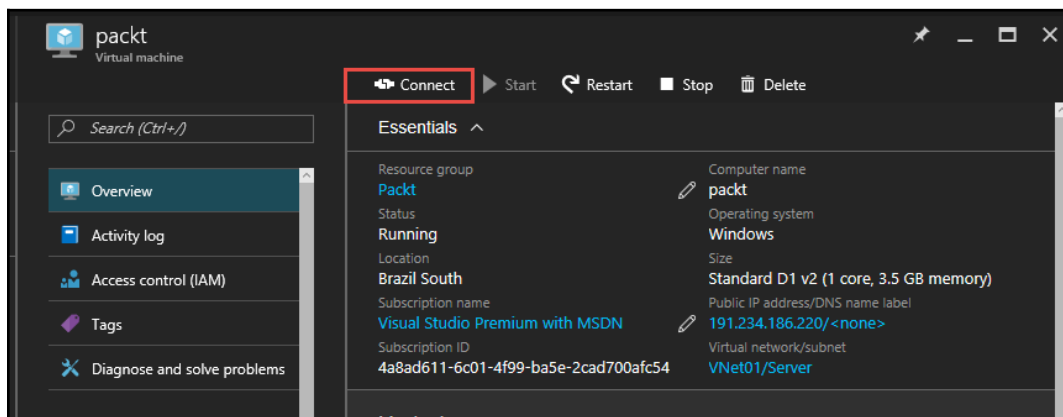
Accessing a Virtual Machine in Azure

As soon as your virtual machines are deployed you can access it.

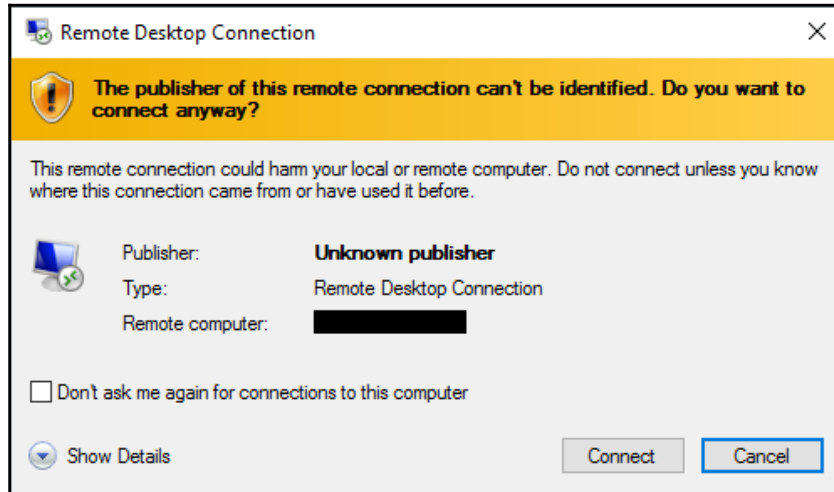
1. Therefore, you can leverage Microsoft **Remote Desktop Connection** for Windows or SSH tools such as PuTTY for Linux:



2. To connect to your virtual machine, you need to navigate to your virtual machine within your resource groups. On the **Overview** blade, you will see the **Connect** button:



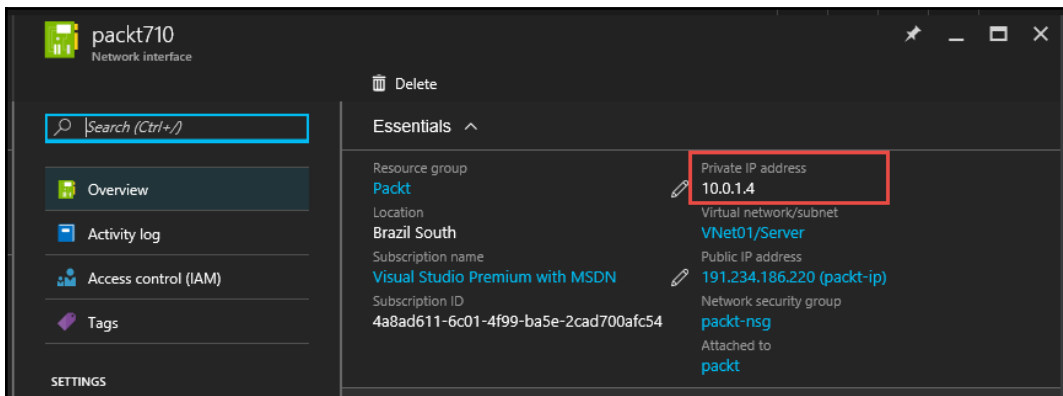
3. For a virtual machine with a public IP, you can download a **Remote Desktop Connection** file and connect directly to the virtual machine:



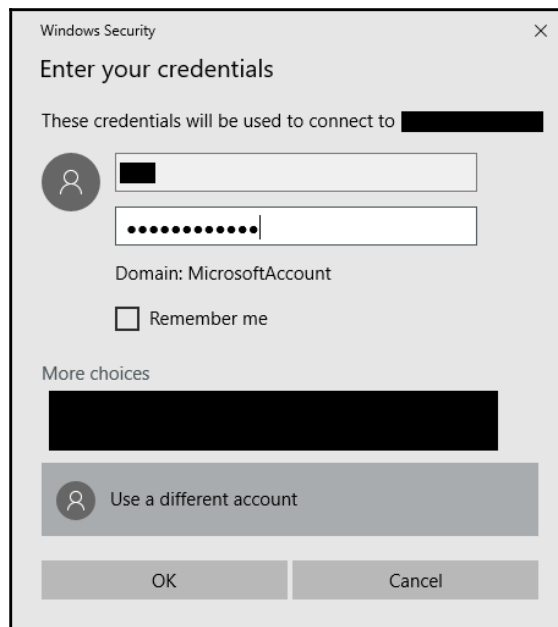
4. If you have a connection to Azure via VPN or ExpressRoute, you can use the private IP given from Microsoft to your virtual machine. This IP can be found on the virtual network adapter of the virtual machine in your resource groups:

NAME	TYPE	LOCATION	
 packt	Virtual machine	Brazil South	...
 packt710	Network inter...	Brazil South	...
 packt-nsg	Network secu...	Brazil South	...
 packt-ip	Public IP addr...	Brazil South	...
 VNet01	Virtual network	Brazil South	...
 packtdiag392	Storage accou...	Brazil South	...
 packtdisks948	Storage accou...	Brazil South	...
 packtsrg	Storage accou...	Brazil South	...

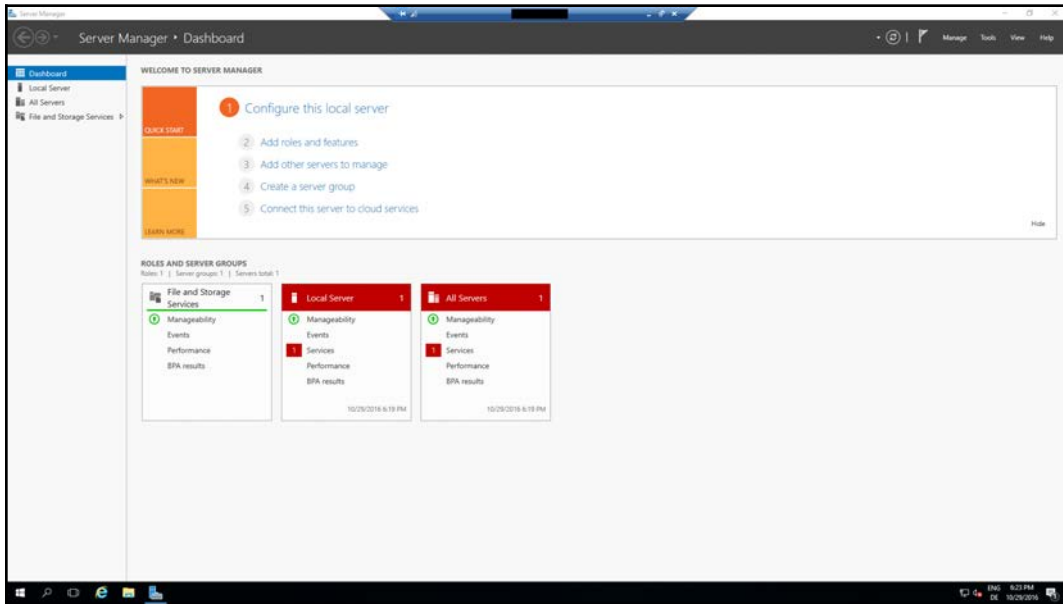
5. There you have also the **Overview** blade and you can see the IP:



6. After you have started the connection, you will enter the credentials you used for the local administrator:

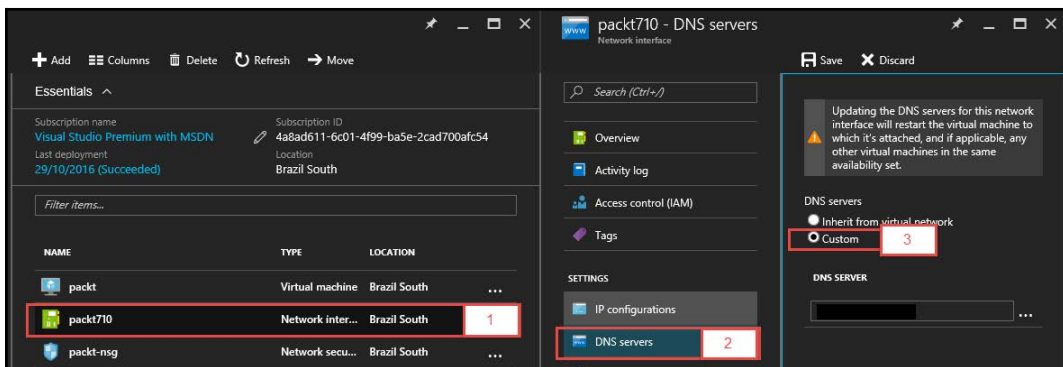


7. Afterwards, you should get the desktop of your virtual machine:

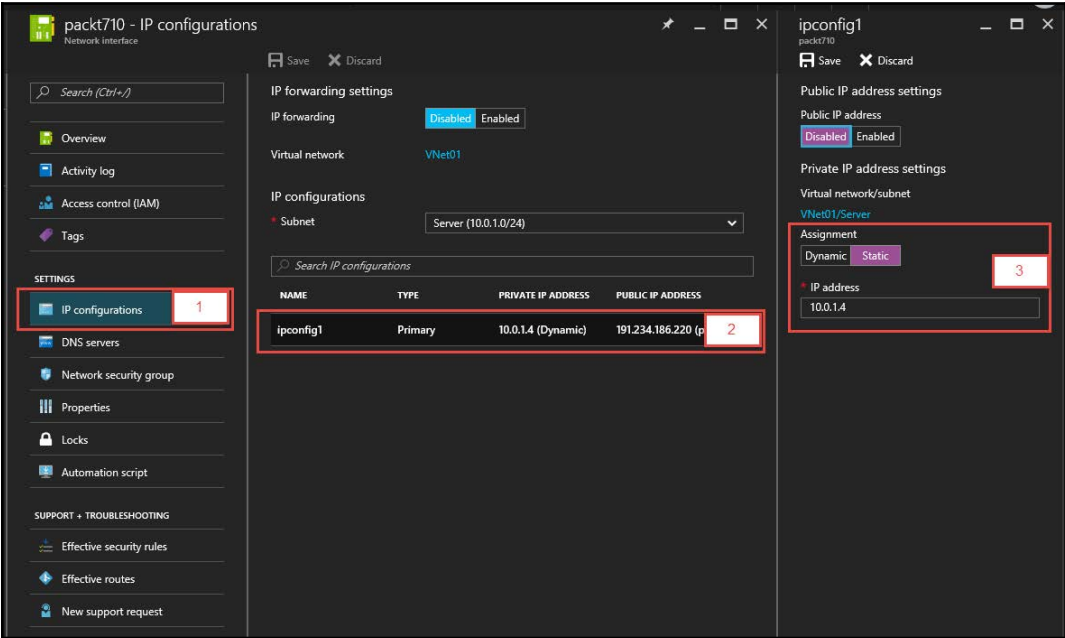


Changing IP and DNS Settings

As soon as you are using virtual machines in an enterprise environment, you may want to change the IPs or DNS servers of a virtual machine, for example, to connect to your domain controller. To change these, you need to navigate back to the network address of the virtual machine. There you click on **DNS servers** and afterwards, you change the option to **Custom** and enter the IP addresses of your **DNS SERVERS**:



To change the IP, you click on **IP configurations**. Then click on the IP configuration. After that you can change the **Assignment** from **Dynamic** to **Static** and change the **IP address** to an IP of your choice within the subnet:



Common Scenarios for Virtual Machines

In the following part of the chapter, you will learn a view common scenarios to start with Azure virtual machines.

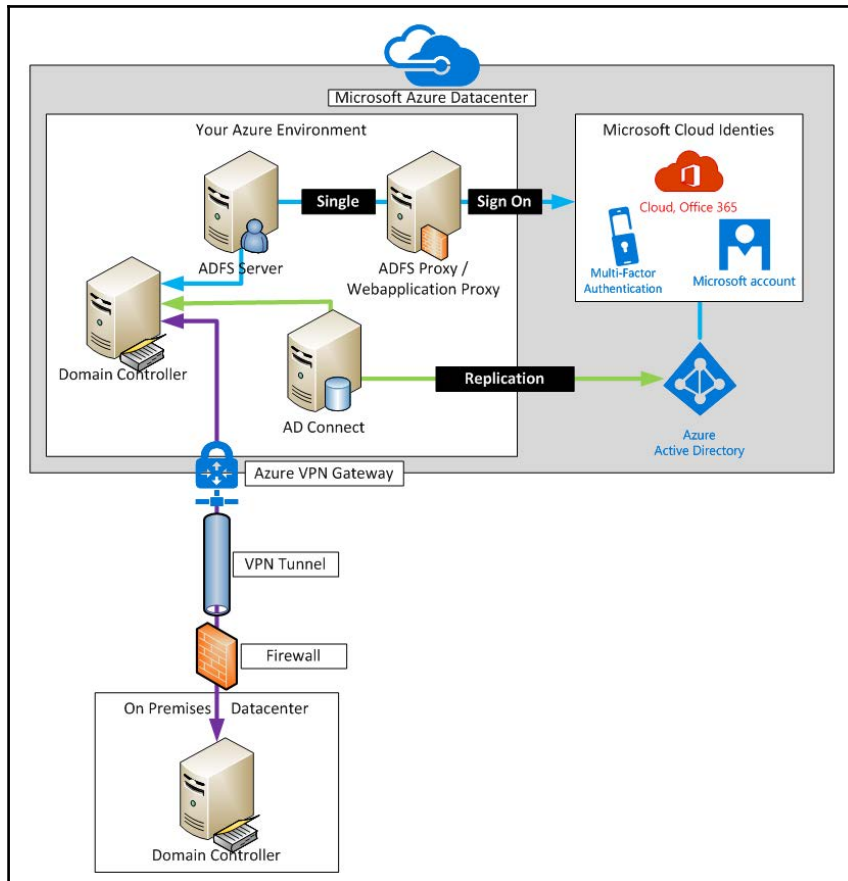
Optimization of Azure Related Communication Traffic

As you already learned in Chapter 3, *Deploying and Synchronizing Azure Active Directory*, replication traffic for your hybrid identities normally goes through the Internet. It's only encrypted by using SSL on port 443.

There is an option to optimize security for that traffic by placing the virtual machines in Azure. They will still communicate against the Azure public IP from Azure Active Directory but the traffic is handled on the internal switches and router from Microsoft and the traffic isn't leaving the Azure data center.

To get the Active Directory account from your on-premises, you build up a VPN tunnel or use ExpressRoute to build a secure connection. Afterwards, you place an Active Directory domain controller in Azure and replicate from a bridgehead domain controller in your on-premises data center.

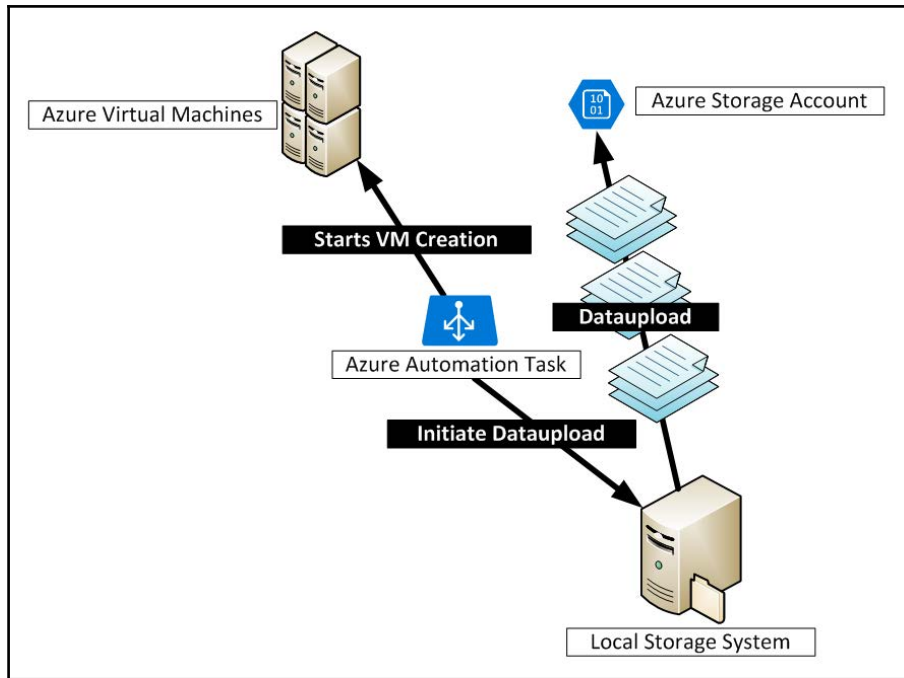
The following diagram shows the concept and virtual machine placing:



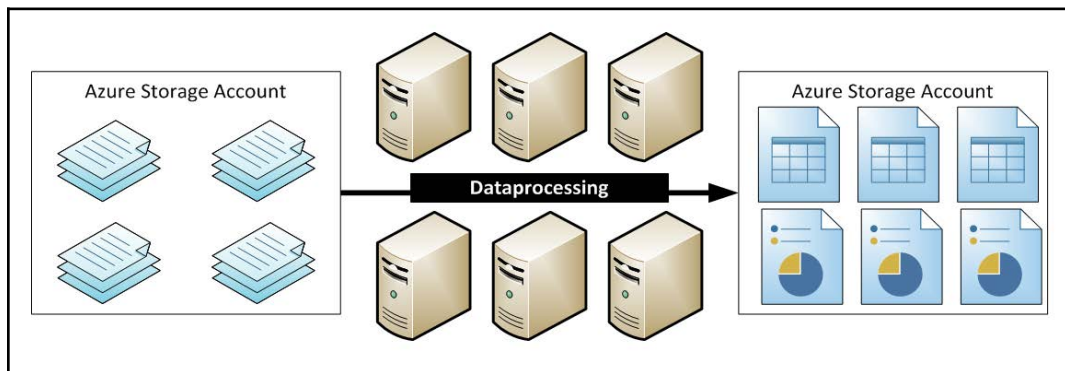
On-demand Usage for Calculations

Another scenario is to use Azure for workloads which are only temporarily needed; for example, science calculations which need a high amount of calculation capacity and where you have a high investment in the hardware.

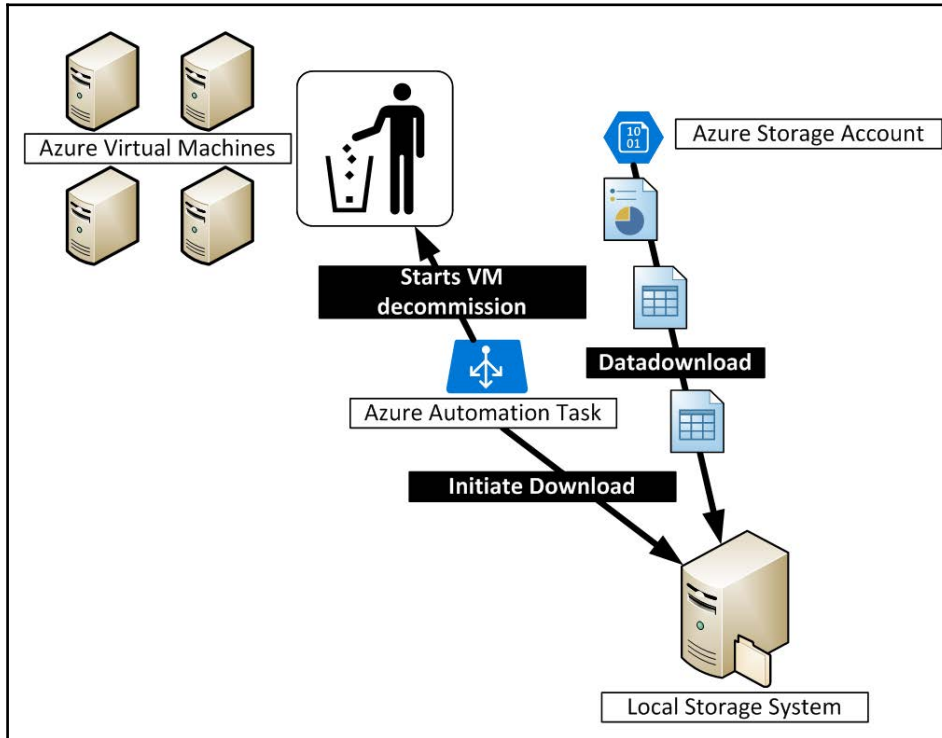
So, what do you do? Normally you create a virtual machine Azure Resource Manager template for those machines. Then you create a task which triggers your virtual machine deployment and the transfer of the raw data into Azure. The following diagram shows a draft of the workflow:



The deployed Azure virtual machines run the necessary calculations and deliver the result into an **Azure Storage Account**. The following diagram shows the process:



As soon as the calculations are finished, the automation task will shut down and delete the Azure virtual machines. Now the final data can be downloaded or you can use other Azure services such as Power BI Embedded or App Services to present or work further on them. The following diagram shows workflow:



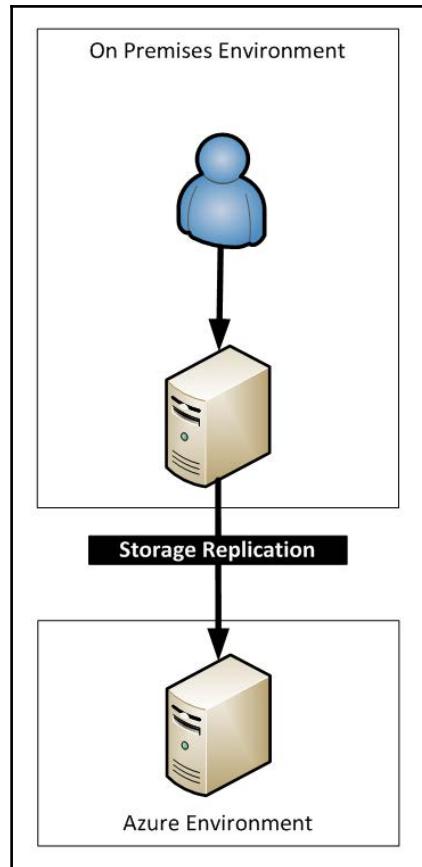
Disaster Recovery for On-Premises Servers

In some cases, it isn't the best choice to place all systems in Azure; for example, when using applications or systems which have a high demand on latency.

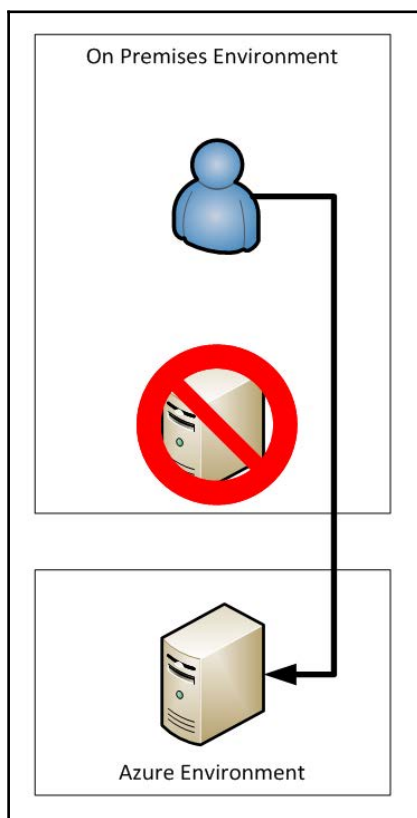
In those cases, it is necessary to keep the system on-premises but Azure could still offer support in those situations with disaster recovery or failover options.

For this example, we look at Windows Server 2016 and the new feature storage replication. With storage replication, you can perform an asynchrony storage replication from one Windows Server 2016 to another.

In our case, we place one server on-premises and one server in Azure. The on-premises server is the primary target and replicates to its partner in Azure. That only produces incoming traffic and your clients still connect to the on-premises server. The following diagram shows an abstract of the workflow:



As soon as the on-premises server fails, your users will be redirected to the system within Azure. For most applications, you will have a decreased user experience but your users are still able to work and you get time to get the on-premises server up and running:



Summary

After this chapter, you should be able to deploy virtual machines and able to decide which VM type and size is the right one for your application. You also know in which scenarios you can benefit from Microsoft virtual machine services and with which scenarios you can start.

In the next chapter, we will take a look into Microsoft Cloud services. Those services are based on Microsoft virtual machines. So, you will recognize some behaviors which are identical to your virtual machines.

7 Implementing Azure Cloud Services

The subject of this chapter is Azure Cloud Services. Azure Cloud Services is the oldest part of the Azure platform and it has been available since its first preview (announced at the Microsoft Professional Developers Conference 2008).

In this chapter, we will cover the following topics:

- Azure Cloud Service
- Cloud Service architecture
- Diving deeper into the Cloud Services
- Azure Cloud Services versus Azure App Services
- Creating your first Azure Cloud Service

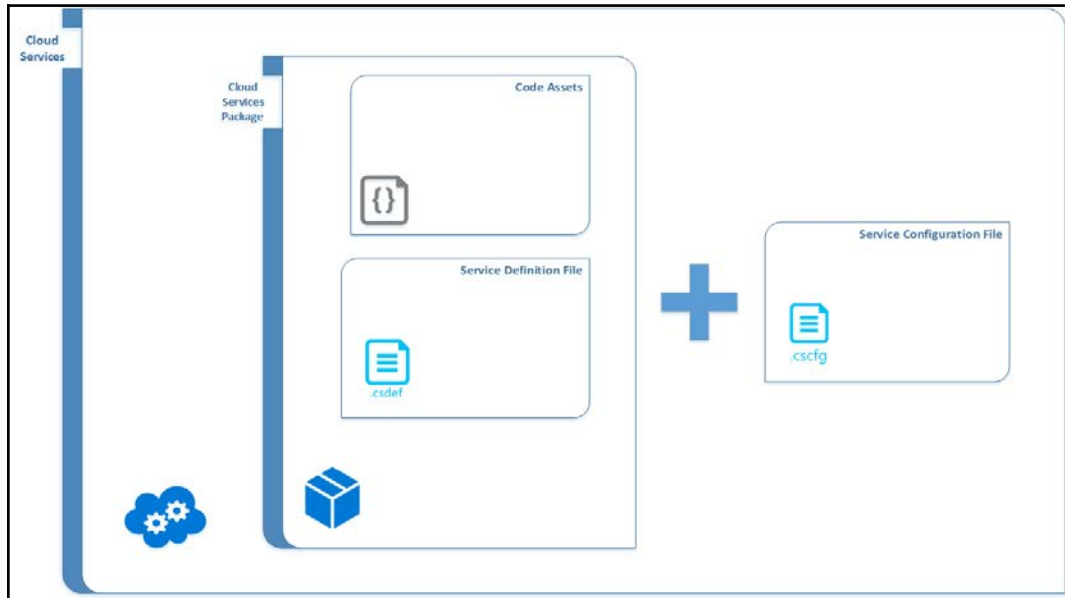
What is an Azure Cloud Service?

Good question! An Azure Cloud Service is a highly available, scalable, and multi-layered web app hosted on a Windows VM with an installed IIS.

Hosted on a VM? Now you're probably thinking that an Azure Cloud Service is an Azure IaaS solution. Though the idea is obvious, there is a major difference. Cloud Services must be designed to work properly when any parts of the service fail. For this reason, the applications must not store their state in the filesystem of their own virtual machines. Unlike virtual machines created with Azure virtual machines, writes to virtual Cloud Services computers are not persistent because they do not have any virtual machines data disks. OK, enough of the remarks.

Understanding the Cloud Service Architecture

We now come to the topic Cloud Service architecture. Look at the following diagram:



As you can see, an Azure Cloud Service consists of the following two elements:

- **Cloud Services Package:** The service package (`ServicePackage.cspkg`) is a ZIP file and it includes the **Service Definition File** (`ServiceDefinition.csdef`) and the **Code Assets** for the service and the required binary-based dependencies
- **Service Configuration File:** `ServiceConfig.cscfg`

As you also can see, the **Service Configuration File** is outside of the **Cloud Service Package**. This is because changes in the configuration can be made without interruption at runtime (by uploading a new Service Configuration File). However, changes to the service itself require a redeployment of the **Cloud Service Package**.

Roles

Let us continue with the next diagram and the other elements of the Cloud Services architecture. These elements are called **roles** and they are created by the Service Definition File and the code assets.



Each role is an instance of the Cloud Service itself (or at least a part of it). In your planning of the Cloud Service, you should therefore consider which task is associated with a role, how often this task is performed, and, per these findings, determine the number of role instances.

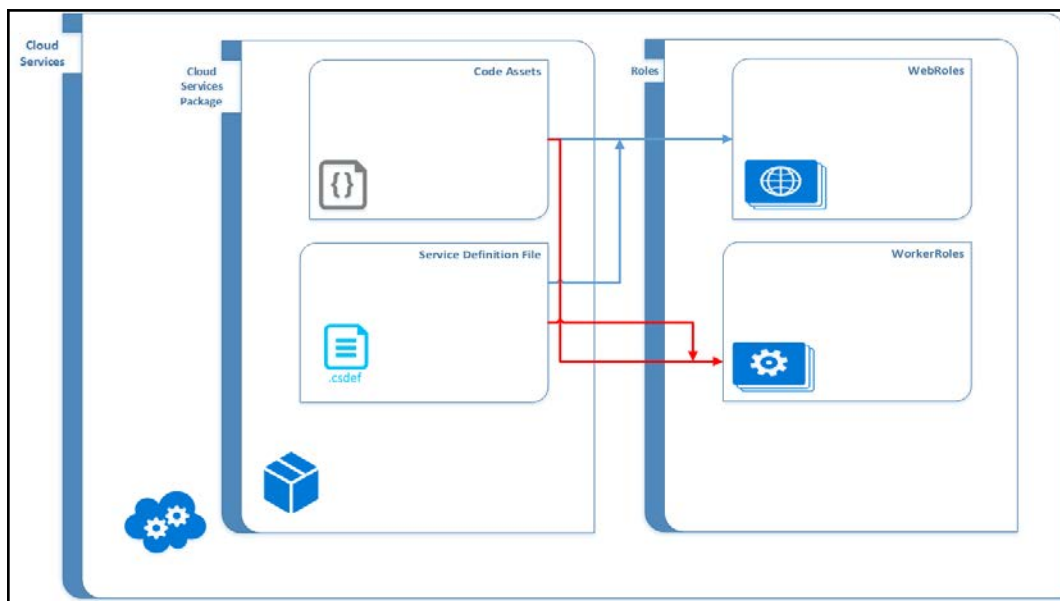
There are currently two options for the roles available:

- WebRoles
- WorkerRoles

The term **WebRoles** is referred to Cloud Service instances that are running on a Windows VM with installed IIS and the term **WorkerRoles** are referred to Cloud Service instances that are running on a Windows VM without installed IIS.

Now we know how the roles are different. The question remains: what are the roles doing?

While instances of the WebRoles serve the actual hosting of your web apps, WorkerRoles are constantly available for the internal processing of business logic and for communicating on the Azure platform (partly using the Azure Service Bus or the Azure Storage Queue Service):



Each Cloud Service has at least one **WebRole** or **WorkerRole**, otherwise the service is not reachable (visible) from outside.

There are limitations for the maximum count of different WebRoles or different WorkerRoles: 25 WebRoles, 25 WorkerRoles, or 25 roles in any combination of both, per provision.

The Service Endpoint

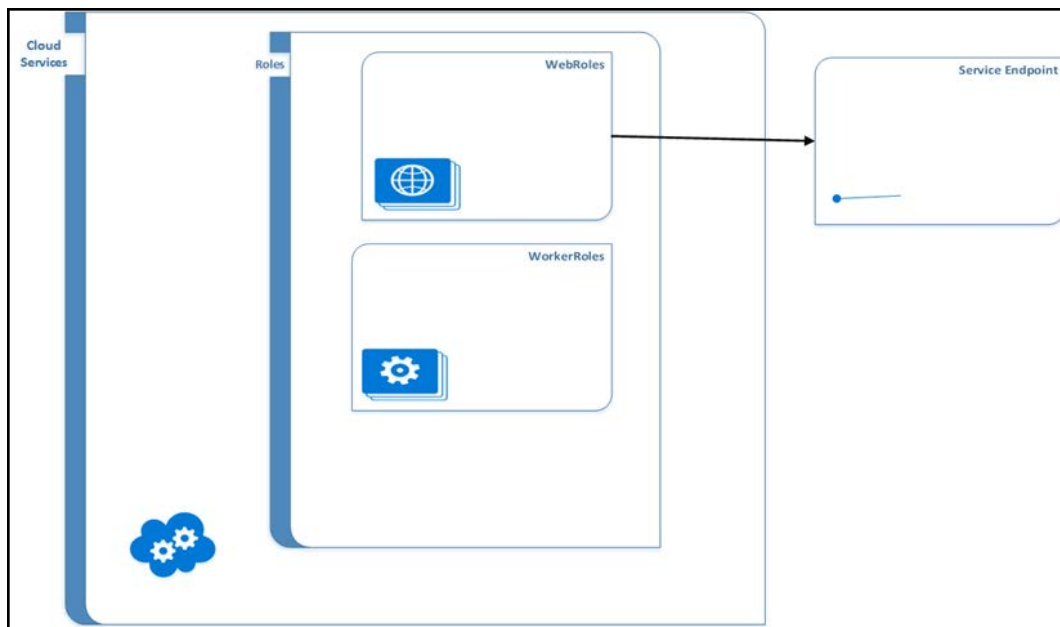
Let's go to the preceding diagram and thus to the last element of the Cloud Service architecture, the service endpoint.

The service endpoint (an IP address or URL) is provided by a WebRole and it is the public interface to the outside world.

**Attention!**

Each Cloud Service has only one service endpoint, but this does not mean that additional endpoints cannot be provided.

For example, a **Cloud Service** may have a normal **Service Endpoint** over the HTTP protocol, while a second endpoint may respond to internal calls (for example, from the intranet) over the TCP protocol:



In the first part of this chapter, we learned about the Cloud Services architecture in a rather simple overview and now it is time to discover the finer details of the Cloud Services.

Going Deeper into the Cloud Services

For this discovery process, we will explain the blueprints of Cloud Services in more detail. Before the question arises, under blueprints of Cloud Services we understand:

- The Service Definition File (`ServiceDefinition.csdef`)
- The Service Configuration File (`ServiceConfig.cscfg`)

Let's take a look.

Service Definition File

The Service Definition File is an XML file based on the **Azure Service Definition Schema** and it describes the components of the Cloud Service.

A basic template of a Service Definition File looks like this:

```
<ServiceDefinition name="<service-name>"
topologyChangeDiscovery="<change-type>"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefi
nition" upgradeDomainCount="<number-of-upgrade-domains>"
schemaVersion=" <version>">
  <LoadBalancerProbes>
  </LoadBalancerProbes>
  <WebRole ...>
  </WebRole>
  <WorkerRole ...>
  </WorkerRole>
  <NetworkTrafficRules>
  </NetworkTrafficRules>
</ServiceDefinition>
```

Overall, we find five elements in the Azure Services Definition Schema. The root element (that is, top level element) is `<ServiceDefinition>` with the mandatory attribute `name` and the three optional attributes `topologyChangeDiscovery`, `schemaVersion`, and `upgradeDomainCount`.

The following are additional elements that you will find:

- `<LoadBalancerProbes>`: Defined in the Azure load balancer probe schema
- `<WebRole>`: Defined in the Azure WebRole scheme
- `<WorkerRole>`: Defined in the Azure WorkerRole scheme
- `<NetworkTrafficRules>`: Defined in the Azure network traffic rules scheme

While the use of `LoadBalancerProbes` and `NetworkTrafficRules` is optional, the elements `WebRole` or `WorkerRole` must be filled to at least one of them.

Now we will look at the four elements in detail.

LoadBalancerProbes

Now for the first question: what is a load balancer probe?

The **Azure Load Balancer (ALB)** is responsible for routing incoming traffic to your role instances. In order for the traffic to be correctly routed, ALB must first send a query to the respective endpoints and check that the URI returns a HTTP 200 OK code. This process is called **load balancer probe**.

In other words, the load balancer probe is a customer defined health probe of endpoints in your role instances.

A `LoadBalancerProbe` is not a standalone element, but it exists only in combination with a `WebRole` or a `WorkerRole`. A `LoadBalancerProbe` may be provided for more than one role.

A template of a `LoadBalancerProbe` element looks like this:

```
<ServiceDefinition ...>
  <LoadBalancerProbes>
    <LoadBalancerProbe name="<load-balancer-probe-name>"
      protocol="[http | tcp]"
      path="<uri-for-checking-health-status-of-vm>"
      port="<port-number>" intervalInSeconds="<interval-in-seconds>"
      timeoutInSeconds="<timeout-in-seconds>"/>
  </LoadBalancerProbes>
</ServiceDefinition>
```

The attribute `name` and `protocol` are always required; the attribute `path` may only be set if you have selected as the attribute `protocol` value as `http`. All other attributes are optional.

WebRole

With the `WebRole` element you define your web application. The only requirement is the ability to run the application on IIS 7 or higher.

Typical examples for this are applications based on ASP.NET (ASP.NET, ASP.NET MVC, and so on), PHP, Windows Communication Foundation, or FastCGI.

A template of a WebRole element looks as follows:

```
<ServiceDefinition ...>
  <WebRole name="<web-role-name>" vmSize="<web-role-size>"
    enableNativeCodeExecution="<[true | false]>"
    <Certificates>
      <Certificate name="<certificate-name>" storeLocation="
        <certificate-store>" storeName="<store-name>" />
    </Certificates>
    <ConfigurationSettings>
      <Setting name="<setting-name>" />
    </ConfigurationSettings>
    <Imports>
      <Import moduleName="<import-module>" />
    </Imports>
    <Endpoints>
      <InputEndpoint certificate="<certificate-name>"
        ignoreRoleInstanceStatus="<[true | false]>"
        name="<input-endpoint-name>" protocol="<[http | https | tcp | udp]>"
        localPort="<port-number>" port="<port-number>"
        loadBalancerProbe="<load-balancer-probe-name>" />
      <InternalEndpoint name="<internal-endpoint-name>"
        protocol="<[http | tcp | udp | any]>" port="<port-number>">
        <FixedPort port="<port-number>" />
        <FixedPortRange min="<minium-port-number>"
          max="<maximum-port-number>" />
      </InternalEndpoint>
      <InstanceInputEndpoint name="<instance-input-endpoint-name>"
        localPort="<port-number>" protocol="<[udp | tcp]>">
        <AllocatePublicPortFrom>
          <FixedPortRange min="<minium-port-number>"
            max="<maximum-port-number>" />
        </AllocatePublicPortFrom>
      </InstanceInputEndpoint>
    </Endpoints>
    <LocalResources>
      <LocalStorage name="<local-store-name>"
        cleanOnRoleRecycle="<[true | false]>"
        sizeInMB="<size-in-megabytes>" />
    </LocalResources>
    <LocalStorage name="<local-store-name>"
      cleanOnRoleRecycle="<[true | false]>"
      sizeInMB="<size-in-megabytes>" />
    <Runtime executionContext="<[limited | elevated]>">
      <Environment>
        <Variable name="<variable-name>" value="<variable-value>">
          <RoleInstanceValue
            xpath="<xpath-to-role-environment-settings>" />

```

```

        </Variable>
    </Environment>
    <EntryPoint>
        <NetFxEntryPoint
            assemblyName="<name-of-assembly-containing-entrypoint>"
            targetFrameworkVersion="<.net-framework-version>" />
        </EntryPoint>
    </Runtime>
    <Sites>
        <Site name="<web-site-name>">
            <VirtualApplication name="<application-name>"
                physicalDirectory="<directory-path>" />
            <VirtualDirectory name="<directory-path>"
                physicalDirectory="<directory-path>" />
            <Bindings>
                <Binding name="<binding-name>"
                    endpointName="<endpoint-name-bound-to>"
                    hostHeader="<url-of-the-site>" />
            </Bindings>
        </Site>
    </Sites>
    <Startup priority="<for-internal-use-only>">
        <Task commandLine="<command-to-execute>"
            executionContext="<[limited | elevated]>"
            taskType="<[simple | foreground | background]>">
            <Environment>
                <Variable name="<variable-name>" value="<variable-value>">
                    <RoleInstanceValue
                        xpath="<xpath-to-role-environment-settings>" />
                </Variable>
            </Environment>
        </Task>
    </Startup>
    <Contents>
        <Content destination="<destination-folder-name>" >
            <SourceDirectory path="<local-source-directory>" />
        </Content>
    </Contents>
</WebRole>
</ServiceDefinition>

```

The number of different elements or attributes within the WebRole scheme is, unfortunately, too extensive to be described in detail here. That's why we want just to cover the key elements.

The key elements are:

Key element	Characteristics
Sites	Contains a collection of definitions for websites or web applications that are hosted in IIS. If the no <code>Sites</code> element is specified, you can only have one website or web application hosted.
Site (child of <code>Sites</code> element)	Contains a definition for a website or web application hosted in IIS.
Endpoints	Contains a collection of definitions for input (external), internal, and instance input endpoints for a role.
InputEndpoints (child of Endpoints element)	Contains the definitions for external endpoints that are used to contact the Cloud Service. You can define HTTP, HTTPS, UDP, or TCP endpoints.
InternalEndpoints (Child of Endpoints element)	Contains the definitions for endpoints that are used for the internal communication within the Cloud Service. You can define HTTP, UDP, or TCP endpoints. You can also use <code>Any</code> as a valid value for an endpoint definition.
InstanceInputEndpoint (Child of Endpoints element)	Contains the definitions for instance input endpoints. An instance input endpoint is associated with a specific role instance, and is used for port forwarding in Azure load balancer. You can define UDP or TCP endpoints.
FixedPort	Specifies the port for the internal endpoint, which enables connections to the Azure load balancer.
FixedPortRange	Specifies a range of ports for internal or instance input endpoints, which enables connections to the Azure load balancer.
ConfigurationSettings	Contains the setting definitions for features.
Certificates	Contains the definitions for certificates that are needed for the role.
Imports	Contains the definitions for imported modules.

VirtualApplication	Contains the definition of a virtual application in IIS. When you create a virtual application in IIS, the application's path becomes part of the site's URL.
VirtualDirectory	Contains the definition of a virtual directory in IIS. A virtual directory is a mapping to a physical directory on your IIS.
Startup	Contains tasks that you can use to perform operations before a role starts. Typical operations are: Installing a component (for example, a runtime environment), registering a COM components, setting up windows registry keys, or starting a long running process. The tasks are defined in a .ps1, a .cmd or executable file.

WorkerRole

With the `WorkerRole` element you define tasks for background processing inside from a `WebRoles` process. `WorkerRoles` run in instances without IIS installed.

A template of a `WorkerRole` element looks as follows:

```
<ServiceDefinition ...>
  <WorkerRole name="<worker-role-name>" vmSize="<worker-role-size>"
    enableNativeCodeExecution="[true | false]">
    <Certificates>
      <Certificate name="<certificate-name>"
        storeLocation="[CurrentUser | LocalMachine]
        storeName="[My|Root|CA|Trust|Disallow|TrustedPeople|
        TrustedPublisher|AuthRoot|AddressBook|<custom-store>]" />
    </Certificates>
    <ConfigurationSettings>
      <Setting name="<setting-name>" />
    </ConfigurationSettings>
    <Endpoints>
      <InputEndpoint name="<input-endpoint-name>"
        protocol="[http | https | tcp | udp]"
        localPort="<local-port-number>" port="<port-number>"
        certificate="<certificate-name>"
        loadBalancerProbe="<load-balancer-probe-name>" />
      <InternalEndpoint name="<internal-endpoint-name>"
```



```

    protocol="[http | tcp | udp | any]" port="<port-number>">
      <FixedPort port="<port-number>" />
      <FixedPortRange min="<minium-port-number>"
        max="<maximum-port-number>" />
    </InternalEndpoint>
    <InstanceInputEndpoint name="<instance-input-endpoint-name>"
      localPort="<port-number>" protocol="[udp | tcp]">
      <AllocatePublicPortFrom>
        <FixedPortRange min="<minium-port-number>"
          max="<maximum-port-number>" />
      </AllocatePublicPortFrom>
    </InstanceInputEndpoint>
  </Endpoints>
  <Imports>
    <Import moduleName=
      "[RemoteAccess | RemoteForwarder | Diagnostics]" />
  </Imports>
  <LocalResources>
    <LocalStorage name="<local-store-name>"
      cleanOnRoleRecycle="[true | false]"
      sizeInMB="<size-in-megabytes>" />
  </LocalResources>
  <LocalStorage name="<local-store-name>"
    cleanOnRoleRecycle="[true | false]"
    sizeInMB="<size-in-megabytes>" />
  <Runtime executionContext="[limited | elevated]">
    <Environment>
      <Variable name="<variable-name>" value="<variable-value>">
        <RoleInstanceValue
          xpath="<xpath-to-role-environment-settings>" />
        </Variable>
      </Environment>
    <EntryPoint>
      <NetFxEntryPoint
        assemblyName="<name-of-assembly-containing-entrypoint>"
        targetFrameworkVersion="<.net-framework-version>" />
      <ProgramEntryPoint
        commandLine="<application>"
        setReadyOnProcessStart="[true | false]" />
    </EntryPoint>
  </Runtime>
  <Startup priority="<for-internal-use-only>">
    <Task commandLine="<command>" executionContext="[limited | elevated]"
      taskType="[simple | foreground | background]">
      <Environment>
        <Variable name="<variable-name>" value="<variable-value>">
          <RoleInstanceValue
            xpath="<xpath-to-role-environment-settings>" />
          </Variable>
        </Environment>
      </Task>
    </Startup>
  </Service>

```

```

        </Variable>
    </Environment>
</Task>
</Startup>
<Contents>
    <Content destination="<destination-folder-name>" >
        <SourceDirectory path="<local-source-directory>" />
    </Content>
</Contents>
</WorkerRole>
</ServiceDefinition>

```

We will dispense with a description of the various elements or attributes within the `WorkerRole` schema because they are essentially identical to the elements and attributes of the `WebRole` schema.

NetworkTrafficRules

With the `NetworkTrafficRules` element you can specify how a role communicates with other roles. Somewhat more specifically, it can limit which roles can access the internal endpoints of the specific role.

A `NetworkTrafficRules` is not a standalone element, but it exists only in combination with a `WebRole` or a `WorkerRole`. A `NetworkTrafficRules` may be provided for more than one role.

A template of a `NetworkTrafficRules` element looks like this:

```

<ServiceDefinition ...>
  <NetworkTrafficRules>
    <OnlyAllowTrafficTo >
      <Destinations>
        <RoleEndpoint endpointName="<name-of-the-endpoint>"
          roleName="<name-of-the-role-containing-the-endpoint>" />
      </Destinations>
      <AllowAllTraffic/>
      <WhenSource matches=" [AnyRule] ">
        <FromRole
          roleName="<name-of-the-role-to-allow-traffic-from>" />
      </WhenSource>
    </OnlyAllowTrafficTo>
  </NetworkTrafficRules>
</ServiceDefinition>

```

The elements of the NetworkTrafficRules schema are:

Elements	Characteristics
OnlyAllowTrafficTo	Contains a collection of endpoints and the roles that can communicate with them. You can specify multiple nodes of this element.
Destinations	Contains a collection of RoleEndpoint.
RoleEndpoint	Contains a description of an endpoint on a role and allows the communications with this endpoint element. You can specify multiple nodes of this element.
AllowAllTraffic	Contains a rule that allows all roles to communicate with the endpoints defined in the Destinations node.
WhenSource	Contains a collection of roles than can communicate with the endpoints defined in the Destinations node.
FromRole	Specifies the roles that can communicate with the endpoints defined in the Destinations node. You can specify multiple nodes of this element.

Service Configuration File

This file is based on the Azure Service Configuration Scheme and it describes how the service is configured.

The Service Configuration File is an XML file based on the Azure Service Configuration Scheme and it describes how the service is configured. A basic template of a Service Configuration File looks as follows:

```
<ServiceConfiguration serviceName="<service-name>"
osFamily="<osfamily-number>" osVersion="<os-version>"
schemaVersion="<schema-version>">
  <Role>
  </Role>
  <NetworkConfiguration>
  </NetworkConfiguration>
</ServiceConfiguration>
```

Overall, we find three elements in the Azure Services Configuration Scheme. Root element (that is, top level element) is `<ServiceConfiguration>` with the mandatory attribute `serviceName` and the three optional attributes `osFamily`, `osVersion`, and `schemaVersion`.

The following are additional elements that you will find:

- `<Role>`: Defined in the Azure role scheme
- `<NetworkConfiguration>`: Defined in the Azure network configuration scheme

We will now look at the two additional elements in detail:

Role

The role element specifies the number of role instances, the values of configuration settings, and the thumbprints for certificates associated with a role.

A template of a role element looks as follows:

```
<ServiceConfiguration>
  <Role name="<role-name>" vmName="<vm-name>">
    <Instances count="<number-of-instances>" />
    <ConfigurationSettings>
      <Setting name="<setting-name>" value="<setting-value>" />
    </ConfigurationSettings>
    <Certificates>
      <Certificate name="<certificate-name>"
        thumbprint="<certificate-thumbprint>"
        thumbprintAlgorithm="<algorithm>" />
    </Certificates>
  </Role>
</ServiceConfiguration>
```

The components of the role schema are:

Components	Properties
name (attribute)	Specifies the name of the role.
vmName (attribute)	Specifies a DNS name for a VM. This component is optional.
Instances	Specifies the number of instances to deploy for the role. To guarantee a failure-free operation, you need at least two instances but there are no limits to increasing the number of instances according to your needs
Setting	Specifies a setting name and value in a collection of settings. This component is optional.
Certificate	Specifies the name, thumbprint, and algorithm of a service certificate. This component is optional.

NetworkConfiguration

The `NetworkConfiguration` element specifies virtual network and DNS values. The `NetworkConfiguration` element is optional for Cloud Services.

A template of a `NetworkConfiguration` element looks as follows:

```
<ServiceConfiguration>
  <NetworkConfiguration>
    <AccessControls>
      <AccessControl name="aclName1">
        <Rule order="<rule-order>"
          action="<rule-action>" remoteSubnet="<subnet-address>"
          description="rule-description"/>
      </AccessControl>
    </AccessControls>
    <EndpointAcls>
      <EndpointAcl role="<role-name>" endpoint="<endpoint-name>"
        accessControl="<acl-name>"/>
    </EndpointAcls>
    <Dns>
      <DnsServers>
        <DnsServer name="<server-name>" IPAddress="<server-address>"
      />
      </DnsServers>
    </Dns>
    <VirtualNetworkSite name="<site-name>"/>
    <AddressAssignments>
```

```
<InstanceAddress roleName="<role-name>">
  <Subnets>
    <Subnet name="<subnet-name>" />
  </Subnets>
</InstanceAddress>
<ReservedIPs>
  <ReservedIP name="<reserved-ip-name>" />
</ReservedIPs>
</AddressAssignments>
</NetworkConfiguration>
</ServiceConfiguration>
```

The components of the network configuration schema are:

Components	Characteristics
AccessControl	Specifies the rules for accessing to the endpoints.
Rule	Specifies the action that should be taken for a specified subnet range of IP addresses.
EndpointAcl	Specifies the relations of access control rules to an endpoint.
DnsServer	Specifies the settings for a DNS server.
VirtualNetworkSite	Specifies the name of a virtual network in which you want deploy your service.
InstanceAddress	Specifies the association of a role to a subnet or set of subnets in the virtual network.
Subnet	Specifies a subnet.
ReservedIP	Specifies the reserved IP address that should be associated with the deployment.



The combination of service definition and Service Configuration Files is denoted as Cloud Service model.

Azure Cloud Services versus other Azure PaaS Offerings, such as Azure App Services

Azure Cloud Services and Azure App Services are parts of the PaaS offer from the Azure platform. Both services support applications that are scalable, reliable, and easy to handle. Both services are hosted in a VM. Is everything the same?

Of course not, because Azure Cloud Services also offer more possibilities for influencing the VM. These possibilities of influence are:

- The selection of a Guest OS and an update level
- The selection of an Azure series (that is, VM size)

In Cloud Services it is also possible to build a remote desktop connection to the VM for diagnose and troubleshoot issues. This is not possible in App Services

Let's take a closer look at these selections.

Selection of a Guest OS and an Update Level

What does this point mean?

The answer is really very simple: you can, if necessary, influence the OS edition (that is, the OS family) and the version (that is, the update level) for the host VM of your Cloud Service.

All you have to do is:

1. In the first line of the Service Configuration File (ServiceConfiguration.cscfg), you must add the attributes `osFamily=""` and `osVersion=""` to the ServiceConfiguration element.
2. Then fill the attributes with values:

```
<ServiceConfiguration serviceName="<service-name>"
osFamily="<osfamily-number>" osVersion="<os-version>"
schemaVersion="<schema-version>">
</ServiceConfiguration>
```

3. Valid values for the `osFamily` attribute can be found in the following table:

OS family	Server OS	Comments
1	Windows Server 2008 SP2	No longer available
2	Windows Server 2008 R2 or Windows Server 2008 R2 SP1	Partially no longer available
3	Windows Server 2012	
4	Windows Server 2012 R2	
5	Windows Server 2016	

With the selection of an OS family, you also have an influence on which .NET framework and which Azure SDK version is supported. Details can be found in the following table:

OS family	.NET framework	Azure SDK
1	No information available	Version 1.0 and higher
2	3.5, 4.0, 4.5, 4.5.1, 4.5.2	Version 1.3 and higher
3	4.0, 4.5, 4.5.1, 4.5.2	Version 1.8 and higher
4	4.0, 4.5, 4.5.1, 4.5.2	Version 2.1 and higher
5	4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2	Version 2.9.5.1 and higher



I cannot publish a list of available values for the `osVersion` attribute, since it is updated every month. The actual valid list can be found at <https://docs.microsoft.com/en-us/azure/cloud-services/cloud-services-guestos-update-matrix>.

In the list of available values for the `osVersion` attribute, you will find three dates:

- **Release date:** From this date, the update level is available
- **Disable date:** Up to this date, the update level is available
- **Expired date:** All instances with this update level are switched off on this date

Between the disable date and expired date are usually 12 months. After this period you must install your instances to a new update level.

Selection of an Azure Series

The term Azure series identifies the available performance levels of an IaaS deployment (provision of a Cloud Service and/or a VM). The performance levels (that is, instance) generally differ in the number of CPU cores, the amount of memory, and the maximum size of the data disk.

Some performance levels or even entire Azure series are also defined by special hardware equipment.

Attention! The selection of the performance level also determines the amount of costs incurred for the service.

Let's take a look at the available Azure series.

Series A

Series A is the classic among the offers and also the only series that is at least partly (in the instances A0 and A4) exists in two versions:

- **Version 1** (basic): This is only intended for workloads in the development and test area
- **Version 2** (standard): This supports the full feature scope of Azure VMs

Instances A8 and A11 are high-end solutions based on an Intel Xeon E5 hardware architecture that is especially suited for data intensive workloads (for example, video encoding, cluster processing, and so on).

Instances A8 and A9 are also network optimized, this means you get a fast network with InfiniBand support.

In detail, each A8 or A9 instance has two network adapters as standard:

- A 10 Gbps Ethernet adapter (to connect to Azure services, such as Azure Storage)
- An InfiniBand 32 GB network adapter and remote direct memory access (RDMA) technology (as a way to communicate with low latency and high throughput between instances in a single Cloud Service or in a single availability group)

The InfiniBand network adapter is reserved for **Message Passing Interface (MPI)** traffic only. Typical applications are, for example, high performance clusters, modeling, and simulations.

Instance	Basic	Standard	Cores	RAM	Max. disc size
A0	X		1	0,75 GB	20 GB
A0		X	1	0,75 GB	20 GB
A1	X		1	1,75 GB	40 GB
A1		X	1	1,75 GB	70 GB
A2	X		2	3,5 GB	60 GB
A2		X	2	3,5 GB	135 GB
A3	X		4	7 GB	120 GB
A3		X	4	7 GB	285 GB
A4	X		8	14 GB	240 GB
A4		X	8	14 GB	605 GB
A5		X	2	14 GB	135 GB

A6		X	4	28 GB	285 GB
A7		X	8	56 GB	605 GB
A8		X	8	56 GB	382 GB
A9		X	16	112 GB	382 GB
A10		X	8	56 GB	382 GB
A11		X	16	112 GB	382 GB

Series D

Strictly speaking, series D not only one series, but it is divided into series D, Dv2 (version 2), DS, and DSv2.

series D is based on a latest generation CPU (unfortunately not exactly declared) and it reaches a speed of 60% higher than series A (at least for instances A0 to A7).

The Dv2 series is based on a latest generation CPU (Intel Xeon E5-2673 V3 processor (Haswell) with 2.4 GHz) and it achieves a 35% higher speed compared to the D series. In addition, the Dv2 CPU, with the Intel Turbo Boost Technology 2.0, can be overclocked to 3.1 GHz.

The higher CPU performance is only one criterion to describe the series D and its offshoots. Other criteria are as follows:

- The higher ratio of memory to core
- The use of an SSD as a temporary disk

When using the SSD as a temporary data carrier, you should know that this is from the series D local.

The DS and DSv2 series are identical to the D and Dv2 series, but they use Azure premium storage as memory:

Instance	Instance with PS	Cores	RAM	Max. disc size
D1	DS1	1	3.5 GB	50 GB
D2	DS2	2	7 GB	100 GB
D3	DS3	4	14 GB	200 GB
D4	DS4	8	28 GB	400 GB
D11	DS11	2	14 GB	100 GB
D12	DS12	4	28 GB	200 GB
D13	DS13	8	56 GB	400 GB
D14	DS14	16	112 GB	800 GB

The following table shows the available instances of the Dv2 series (version 2):

Instance	Instance with PS	Cores	RAM	Max. disc size
D1v2	DS1v2	1	3.5 GB	50 GB
D2v2	DS2v2	2	7 GB	100 GB
D3v2	DS3v2	4	14 GB	200 GB
D4v2	DS4v2	8	28 GB	400 GB
D5v2	DS5v2	16	56 GB	800 GB
D11v2	DS11v2	2	14 GB	100 GB
D12v2	DS12v2	4	28 GB	200 GB
D13v2	DS13v2	8	56 GB	400 GB
D14v2	DS14v2	16	112 GB	800 GB
D15v2	DS15v2	20	140 GB	1000 GB

series F

Series F is based on a state-of-the-art CPU (Intel Xeon E5-2673 V3 (Haswell) at 2.4 GHz), offering 2 GB of RAM per core and a 16 GB temporary disk based on an SSD.

The F series is especially suited for computer-intensive workloads and is used in scenarios such as batch processing, web server operation, analysis, and gaming.

The F CPU can additionally be overclocked with the Intel Turbo Boost Technology 2.0, still up to 3.1 GHz.

The FS series is identical to the F series, but it uses Azure premium storage as its memory.

Instance	Instance with PS	Cores	RAM	Max. disc size
F1	FS1	1	2 GB	16 GB
F2	FS2	2	4 GB	32 GB
F4	FS4	4	8 GB	64 GB
F8	FS8	8	16 GB	128 GB
F16	FS16	16	32 GB	256 GB

series G

series G is also based on a latest generation CPU (Intel Xeon E5-2673 V3) (Haswell with 2.4 GHz) and it is based on the older series D, but the G instances have a twice larger memory and four times larger temporary disks based on an SSD. With exceptional, high-performance VM sizes in the G range, you can easily handle business critical applications such as large relational database servers (SQL Server, MySQL, and so on) or large NoSQL databases (MongoDB, Cloudera Cassandra, and so on).

The G CPU can additionally be overclocked with the Intel Turbo Boost Technology 2.0, still up to 3.1 GHz.

The GS series is identical to the G series, but it uses Azure premium storage as a memory:

Instance	Instance with PS	Cores	RAM	Max. disc size
G1	GS1	2	28 GB	412 GB
G2	GS2	4	56 GB	824 GB
G3	GS3	8	112 GB	1649 GB

G4	GS4	16	224 GB	3298 GB
G5	GS5	32	448 GB	6596 GB



G5 instances are run in isolation on dedicated hardware that is deployed for only one customer.

series N

Now we could report on the type of CPU (it is again an Intel Xeon E5-2690 V3 processor), about memory, or about the size of the temporary disk. But it does not matter, because the series N is not defined by these factors, but by the possibility to use GPUs (the processors of the graphics card) as an additional power factor.

What is the basic idea?

In the form of the GPU, additional computing capacity is provided, whereby the GPU generally operates faster than the CPU in highly parallelizable program sequences (high data parallelism), and therefore, a simple way is sought to supplement the computing performance of CPUs by the computing performance of GPUs. The CUDA and/or OpenCL technologies are used for this purpose.

The use of the GPU performance is particularly suitable for computer- and graphics-intensive workloads and supports you in scenarios such as high-end visualization, deep learning, and predictive analytics.

What is CUDA?

Compute Unified Device Architecture (CUDA) is a technology developed by NVidia, which can be used to process program parts through the GPU. In other words, with the help of CUDA, the GPU becomes a coprocessor in the processing process.



What is OpenCL?

Open Computing Language (OpenCL) is an interface (or an open standard) for cross-platform computing on CPUs and GPUs. OpenCL was developed by Apple, in cooperation with the companies AMD, IBM, Intel, and NVidia and standardized by the Khronos Group (an industry consortium).

Let us return to series N-all offers includes GPUs from the NVIDIA Tesla accelerated computing platform (this is the NVIDIA high-end data center solution).

All offers with the identification NC (the C stands for Compute) use the GPU NVIDIA Tesla K80 and all offers with the identification NV (the V stands for **visualization**) use the GPU NVIDIA Tesla M60.

For all NV-rated offers the NVIDIA GRID technology (NVIDIA solution for graphics-accelerated virtual desktops) is used in addition.

NC24r is a special case and it offers you a network interface for high throughput (RDMA) and latencies of only a few seconds.

The following table shows the available instances of the N series:

Instance	Cores	GPUs	RAM	Max. disc size
NC6	6	1	56 GB	340 GB
NC12	12	2	112 GB	680 GB
NC24	24	4	224 GB	1440 GB
NC24r	24	4	224 GB	1440 GB
NV6	6	1	56 GB	340 GB
NV12	12	2	112 GB	680 GB
NV24	24	4	224 GB	1440 GB

Series H

Series H is based on a latest generation CPU (Intel Xeon E5-2667 V3) (Haswell) with 3.2 GHz) and it can be additionally overclocked with the Intel® Turbo Boost Technology 2.0, up to 3.6 GHz (this option is per default active). Furthermore, the H series is characterized by modern DDR4 RAM and an SSD-based data memory.

series H is specifically designed for processing **high performance computing (HPC)** workloads. These include, for example, financial risk models, simulations in the field of seismology and deposits, calculation of flow dynamics, and genome research.

Instance	Cores	RAM	Max. disc size
H8	8	56 GB	1000 GB
H16	16	112 GB	2000 GB
H8m	8	112 GB	1000 GB
H16m	16	224 GB	2000 GB
H16r	16	112 GB	2000 GB
H16mr	16	224 GB	2000 GB

You have already noticed that some instances have additional markings in the form of a *lowercase* letter.

Instances with the label *m*, have a double RAM memory, compared to the normal instance.

Instances with the label *r* also have a second low latency and high throughput network interface (RDMA) optimized for tightly coupled parallel compute workloads (for example, MPI applications). The second network interface is provided by an FDR InfiniBand network.



As mentioned previously, there are some Azure series (with the name suffix *S*) that use Azure Premium Storage instead of the normal Azure Storage. Of course, the question arises: Why should we opt for the Azure Premium Storage variant? With Azure Premium Storage, for example, significantly higher costs are involved.

I think we should just clarify what you get, and then you can decide if you need it:

Azure premium storage is a high-performance SSD-based storage that is designed for I/O-intensive workloads with very high throughput and very low latency. The local temporary disk also provides a local SSD disk cache.

In a nutshell

If you don't need the additional control options, it's typically easier to get a web application up and running in Azure app service web apps.

Creating your first Azure cloud service

This workflow is usually divided into two parts. In part 1, we do the necessary preliminary work in the Azure management portal, and then in part 2 we create the process.

Part 1

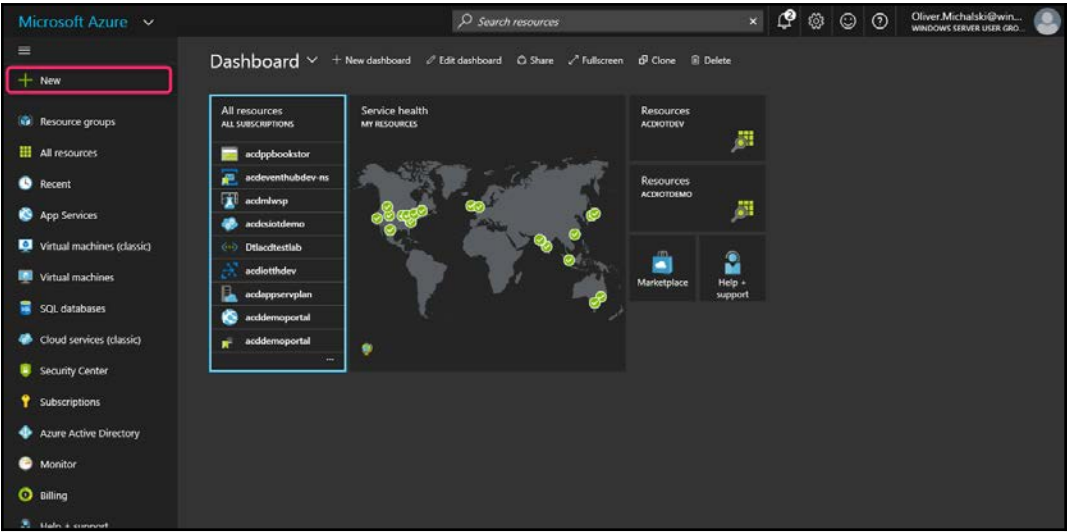
As mentioned, in the first part we are doing the necessary preliminary work in the Azure management portal. Of course, the question immediately arises: what preliminary work?

Short answer, before we can create an Azure Cloud Service and deploy it to the Azure platform, we still need a so-called hosting container for the service.

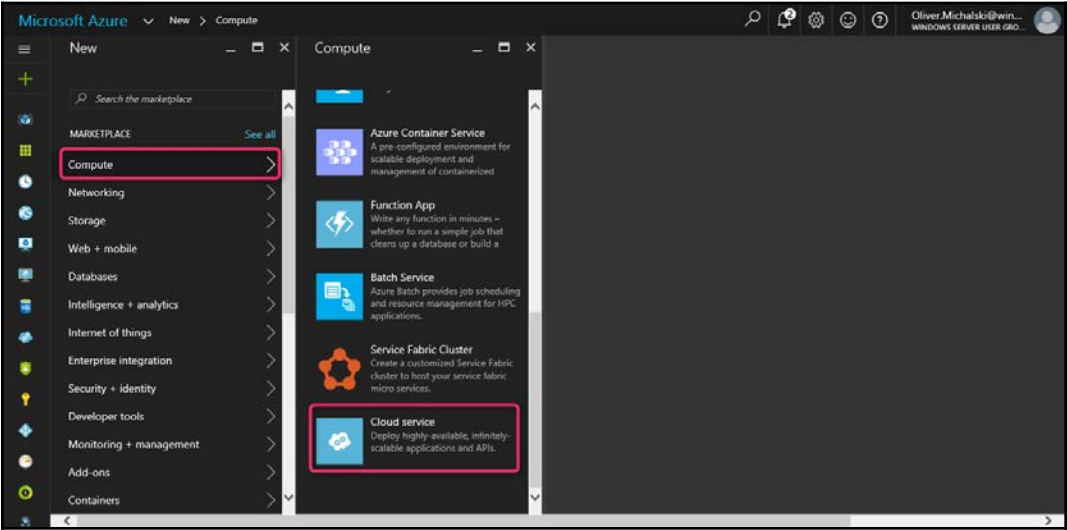
We are now going deeper into the hosting of the container. We will see how to create a hosting container and how you can work with it.

Let's start:

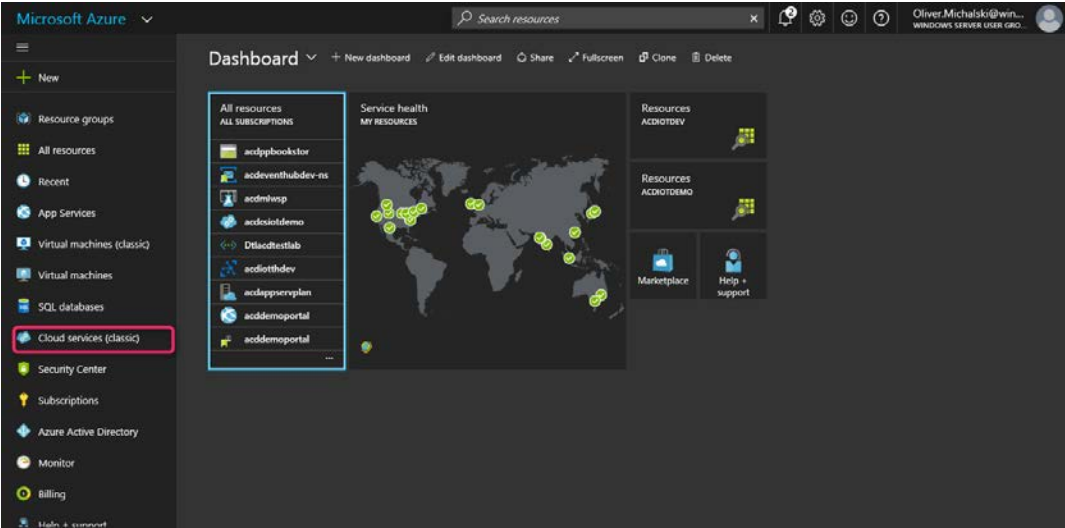
1. Open your Azure management portal at <https://portal.azure.com>.
2. In the portal, click on **New**:



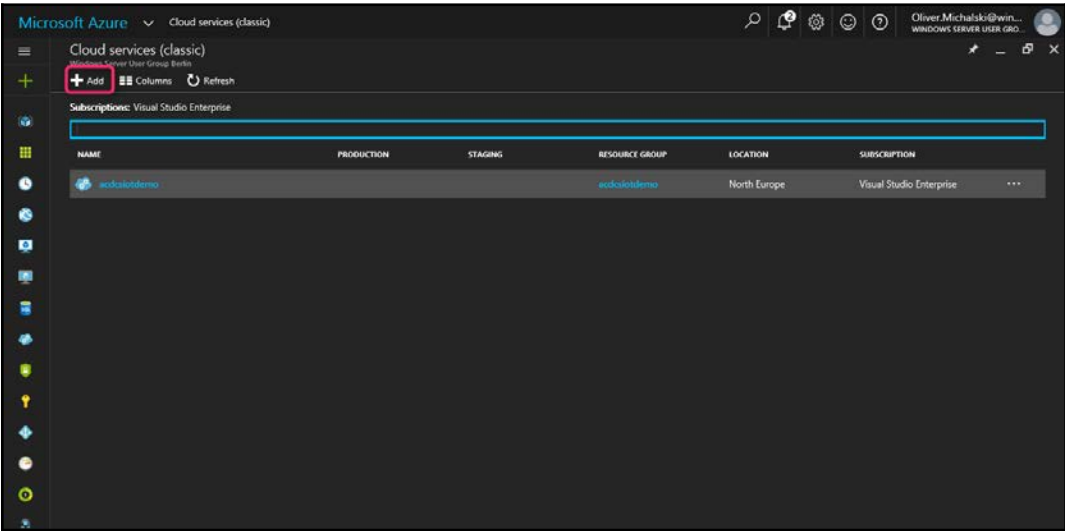
3. Now click on **Compute**, and then click **Cloud Service**:



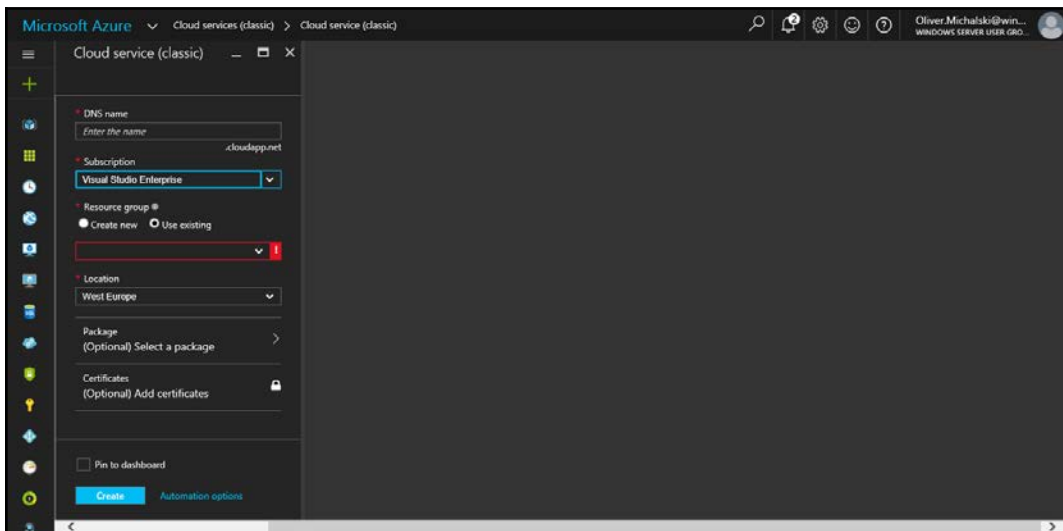
4. Alternatively, you can also click the navigation entry **Cloud Service (classic)**:



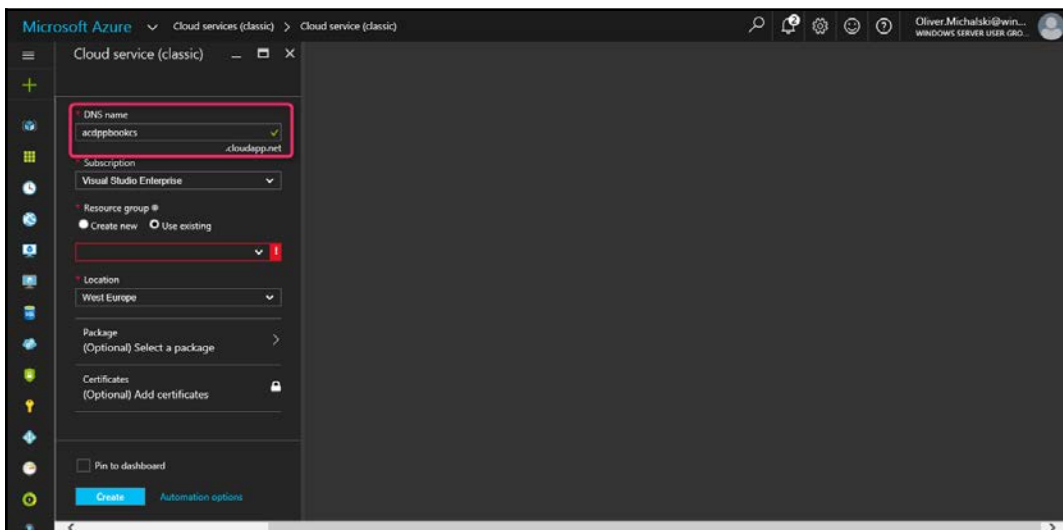
5. Now press the **Add** button in the **Cloud Services (classic)** blade:



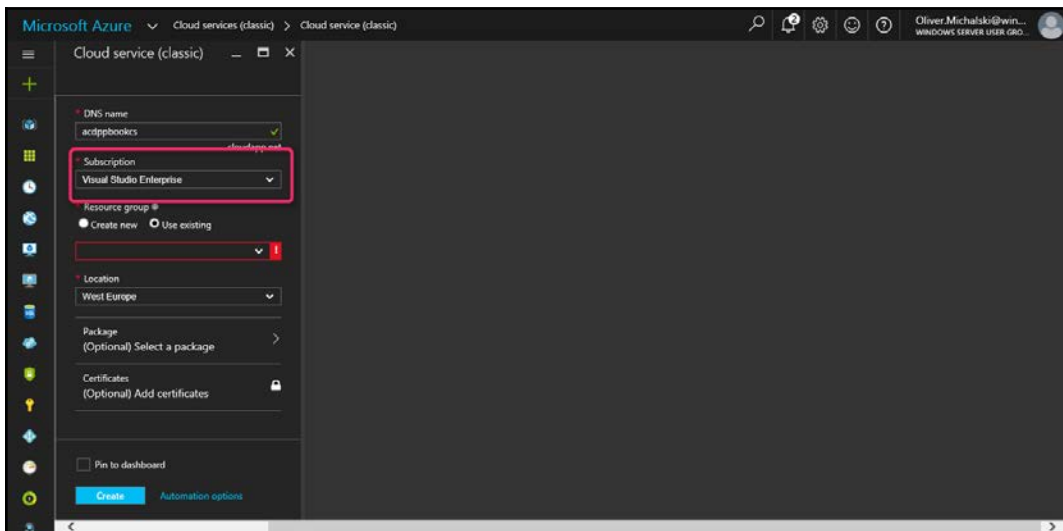
6. Now the **Cloud Services (classic)** window will appear:



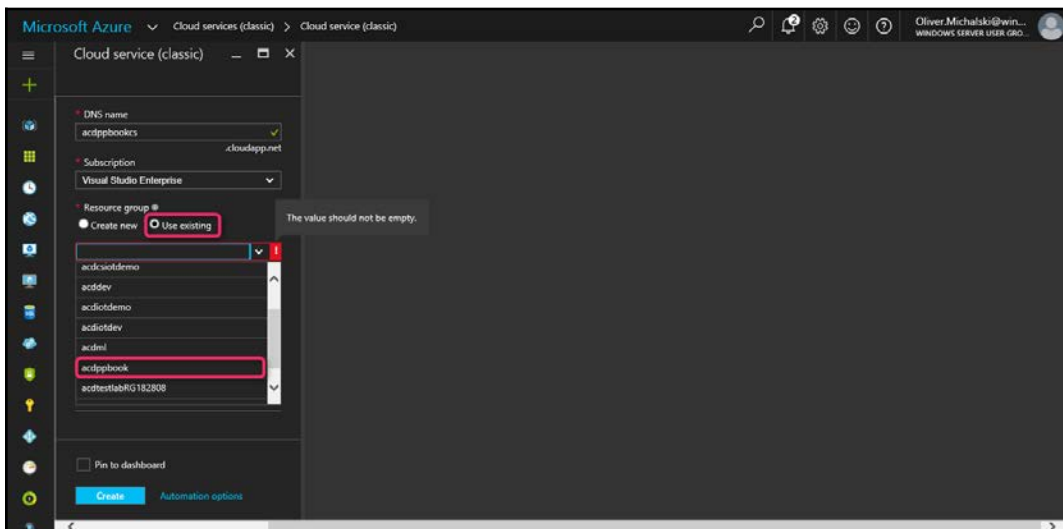
7. Type a unique name for the service you are creating in the **DNS name** box. If the name is unique, you will see a green tick:



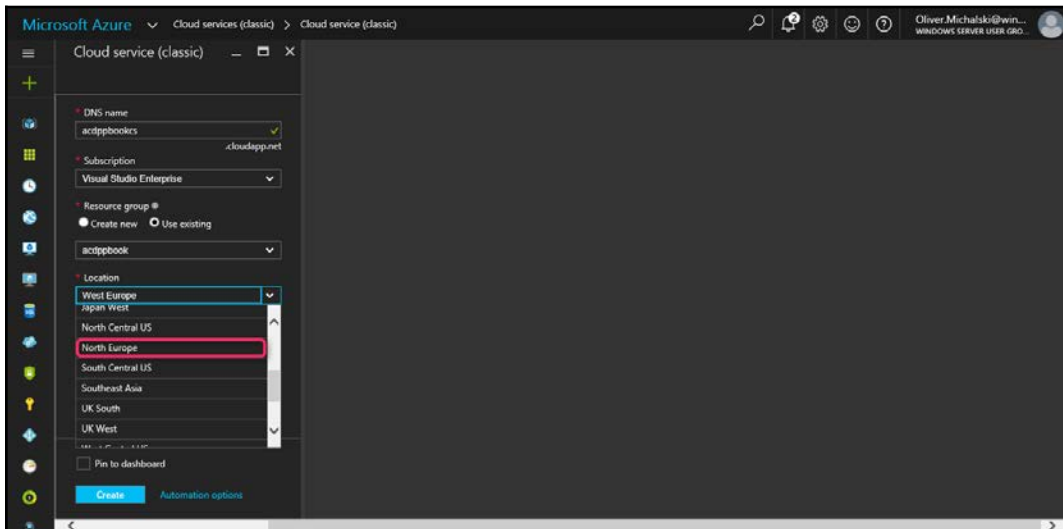
8. Next, choose a **Subscription** (use the default subscription):



9. In the **Resource group** section, click the **Use existing** checkbox, and then search and select **acdpbook** (you can find information about creating this resource group in Chapter 2, *Azure Resource Manager and Tools*), in the drop-down list:

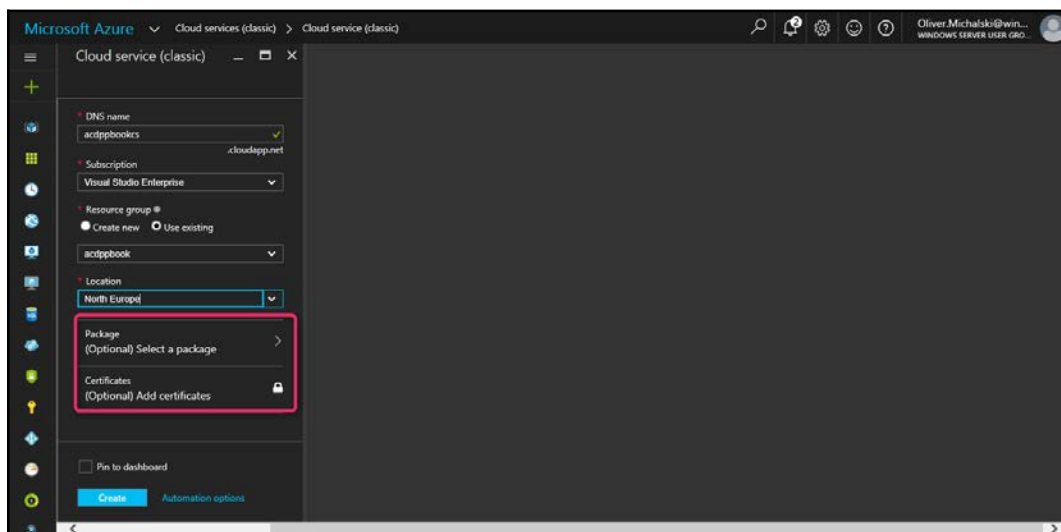


10. In the **Location** list, select the same location you have been using for the Azure resource group:

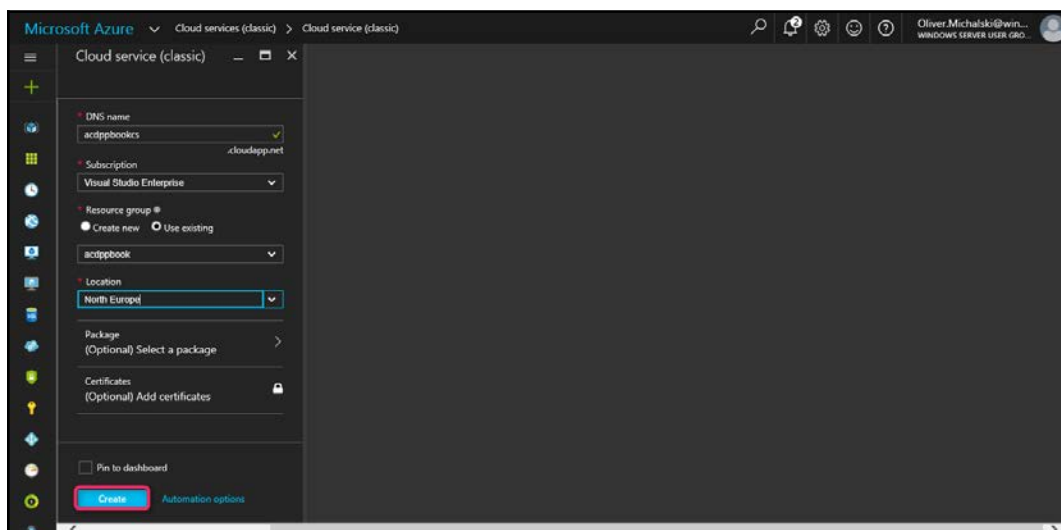


11. In the dialog box two optional settings are available:
- **Package:** For the deployment of an existing service package file
 - **Certificates:** For the deployment of a certificate

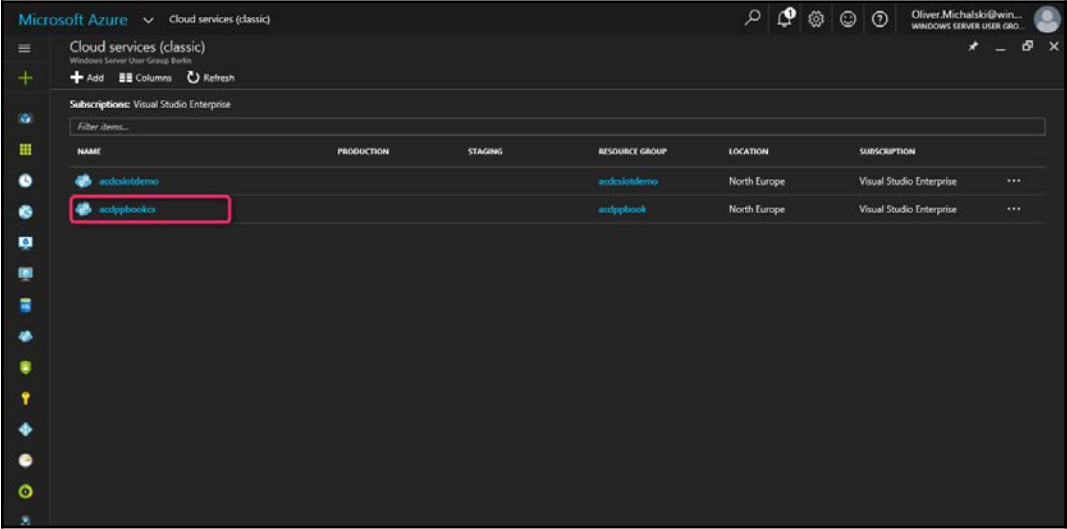
12. Since both settings are not relevant to our description, we will not be using these options for this tutorial:



13. Finally, press the **Create** button.



14. A few minutes later you will see the result in the **Cloud Service (classic)** blade. For moving forward, please press the **NAME** field of your new service:

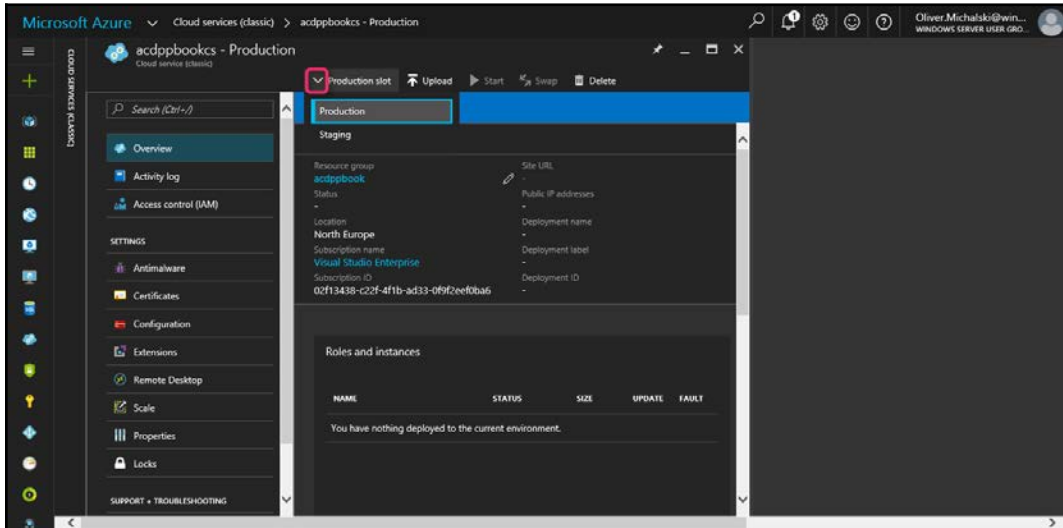


Now the dashboard of your Cloud Services will open. Here we will briefly review the navigation area at the top of the dashboard.

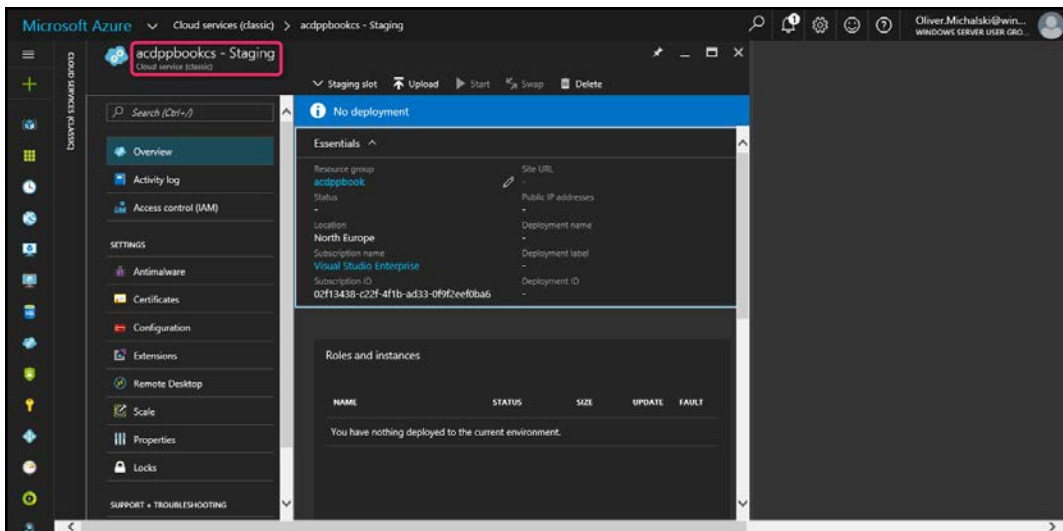
The first part of the considerations concerns the so-called **slots**. Microsoft Azure provides two slots for your later deployments:

- **Production**
- **Staging** (for testing or development)

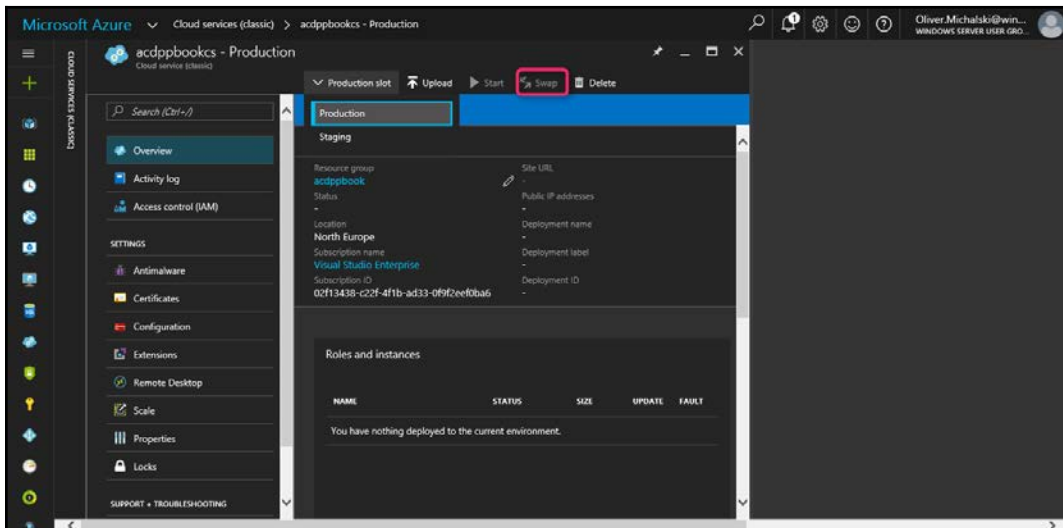
So you have access to these slots, you will find a fold-out list in the navigation bar:



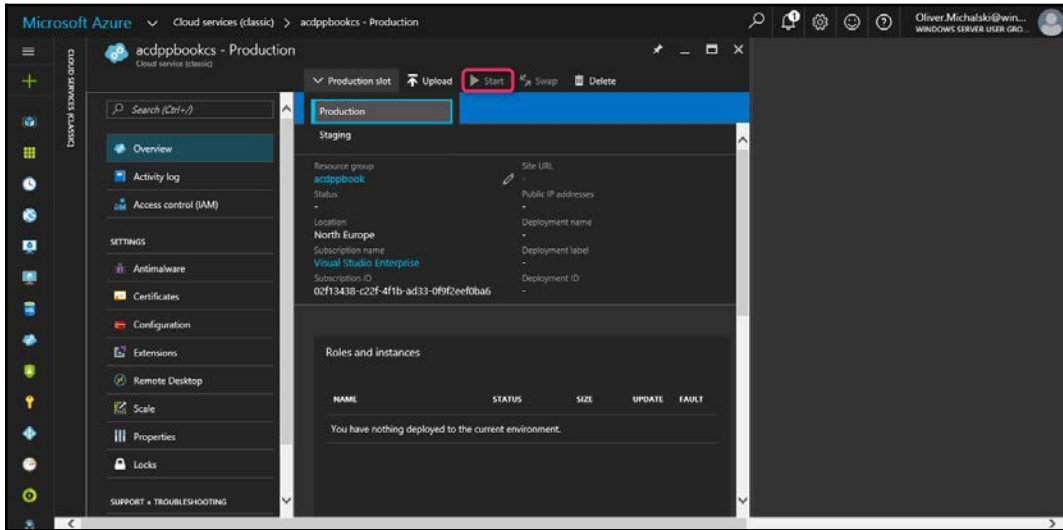
1. A simple click on the list immediately changes the view:



2. As soon as you have completed your tests, and have made the necessary adjustments in the settings, you can use the **Swap** buttons to move the deployment from the **Staging** slot into the **Production** slot:



3. Deployment of the service does not mean that the services are already running. You must first start the VM with the help of the **Start** button:



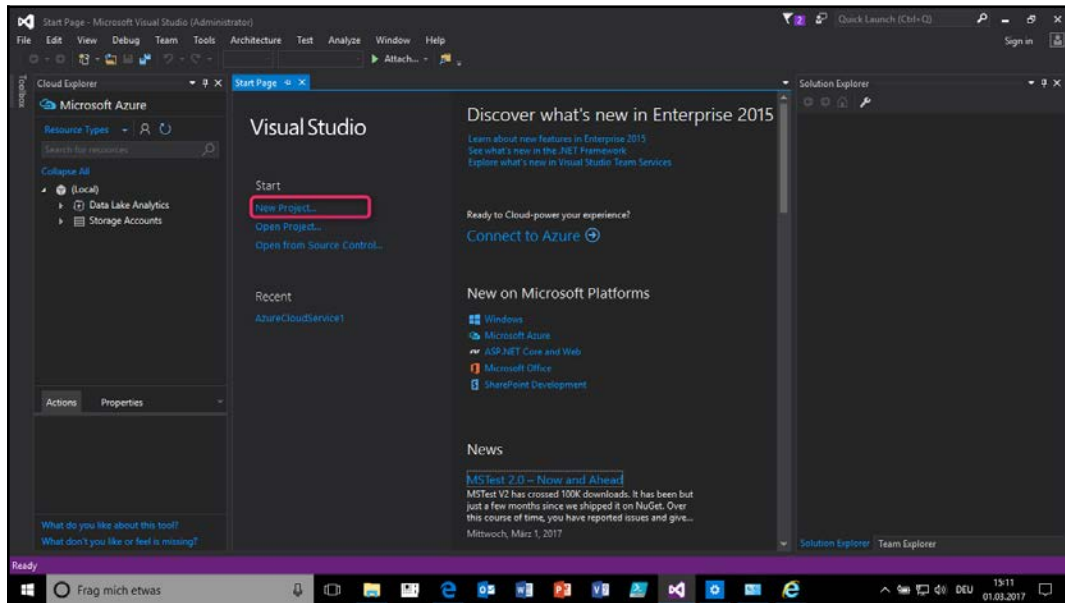
Part 2

Some important notes: We will now create the Cloud Service implementation as an example workflow in Visual Studio, but depending on your desired programming language, you can also use Eclipse or IntelliJ IDE.

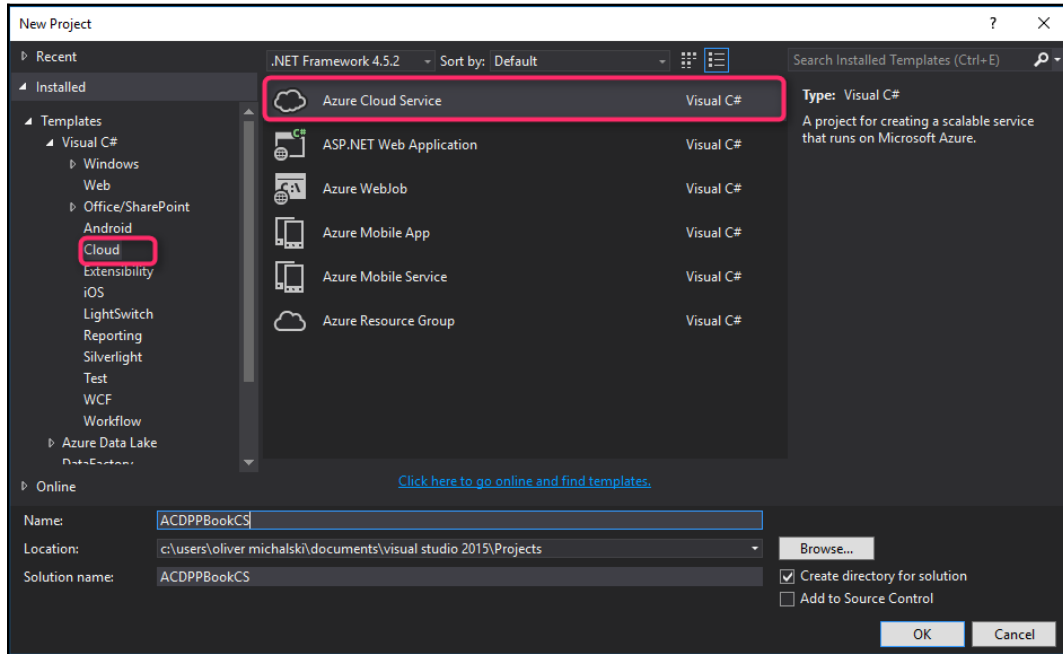
We are using Visual Studio 2015 Enterprise Edition for work, but you can also use any other edition of Visual Studio for this. To prevent unnecessary expense, the free Visual Studio Community Edition is sufficient for our purposes completely.

Because we want to work with Microsoft Azure, you must start Visual Studio in the **Run as Administrator** mode:

1. Open **Visual Studio**, and then click the **New Project...** link:

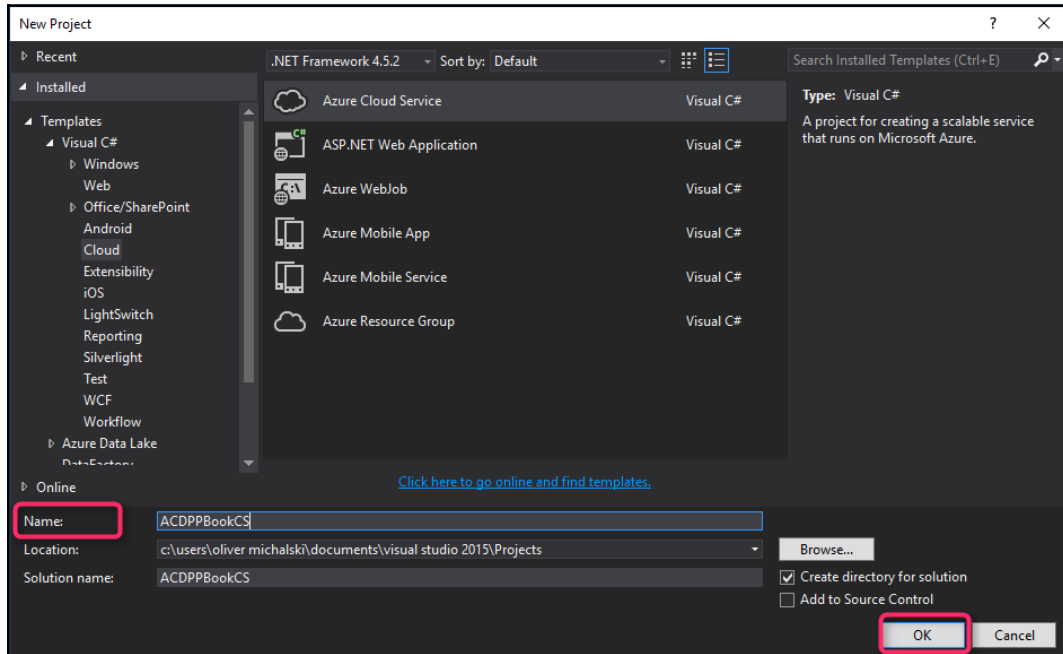


2. Now open the selection dialog with the available project templates. The required template **Azure Cloud Service**, can be found in the **Cloud** area:

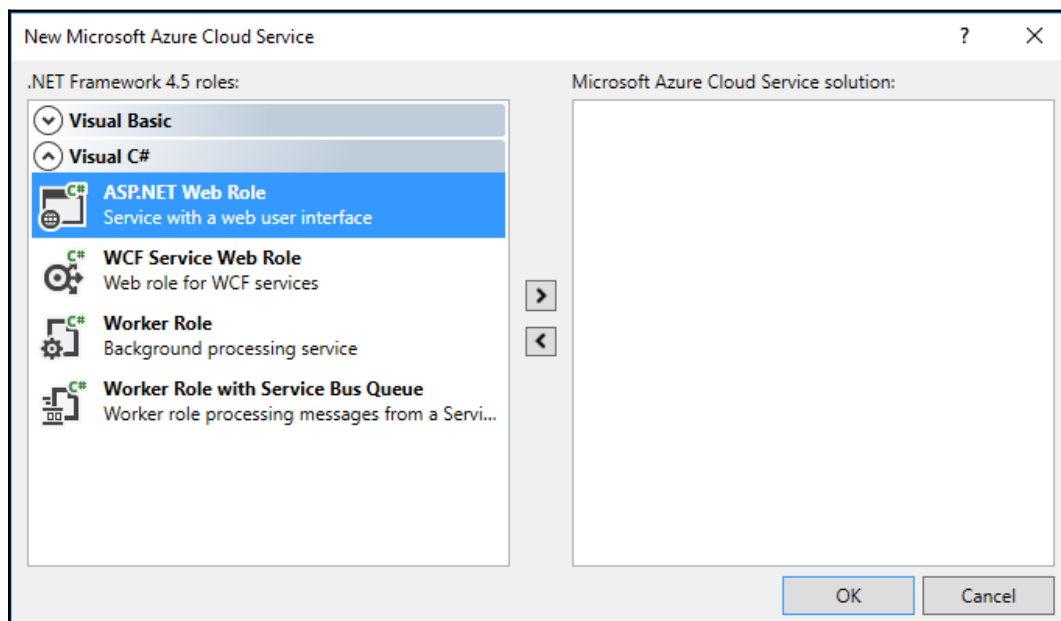
**Attention!**

If you do not find the entry there, you need to install Azure SDK and Azure VS tooling.

3. If everything is clear, specify a project name (for example, ACDPPBookCS) and press the OK button:

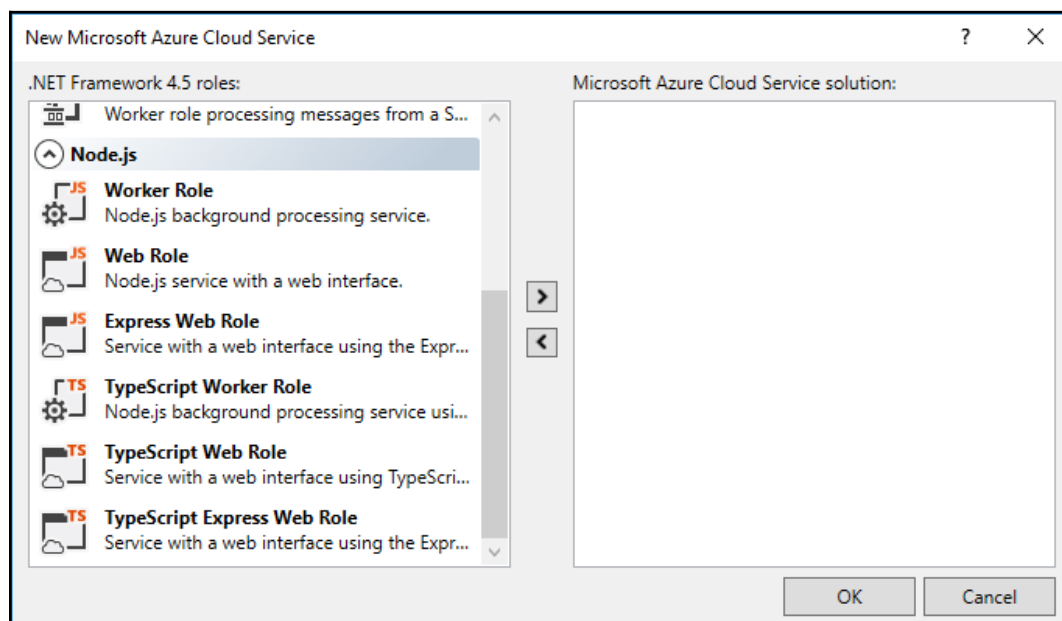


4. Now open another selection dialog, this time with a list of available role templates:

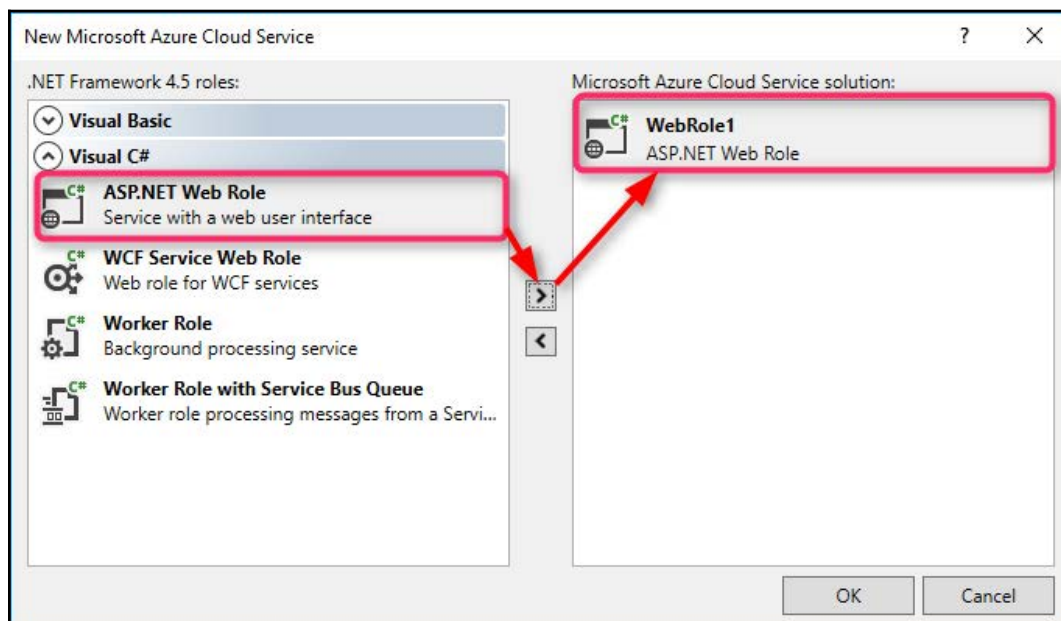


Templates are available for all supported programming languages. Depending on the edition or the chosen installation of Visual Studio, this can also be a more extensive selection.

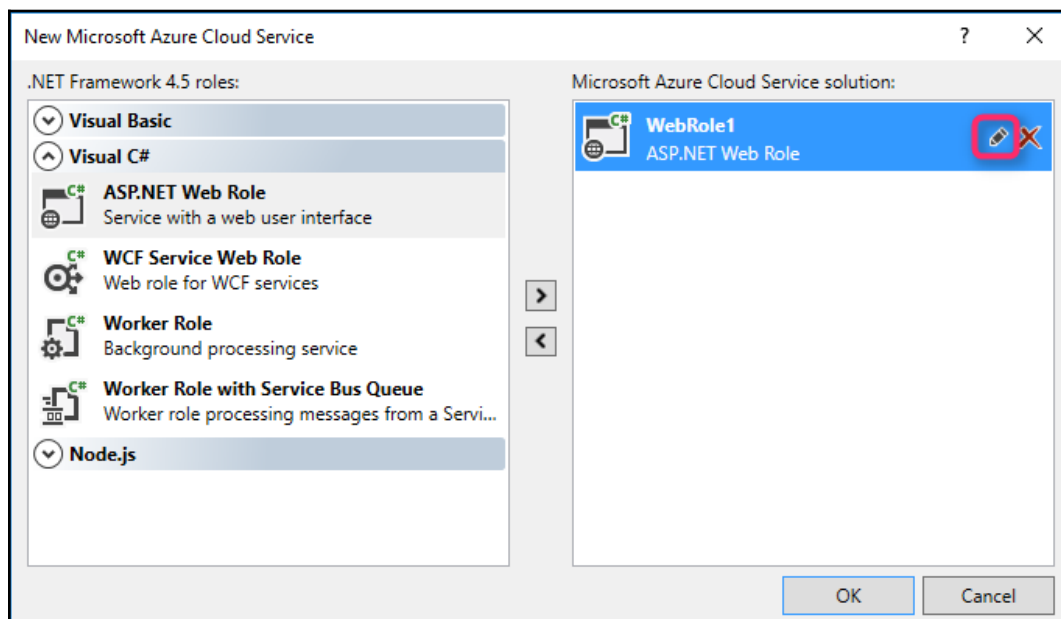
For example, in the following screenshot, you can also see templates for Node.js, Node.js (Express), and TypeScript:



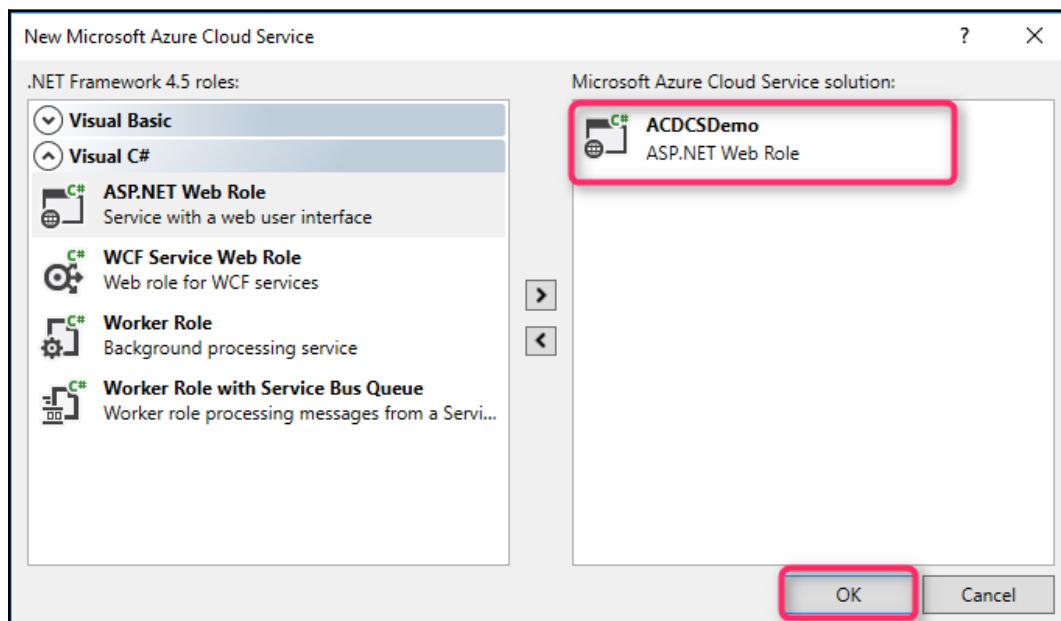
1. For our demo, we will use an **ASP.NET WebRole**. Please select the appropriate role and then press the arrow button:



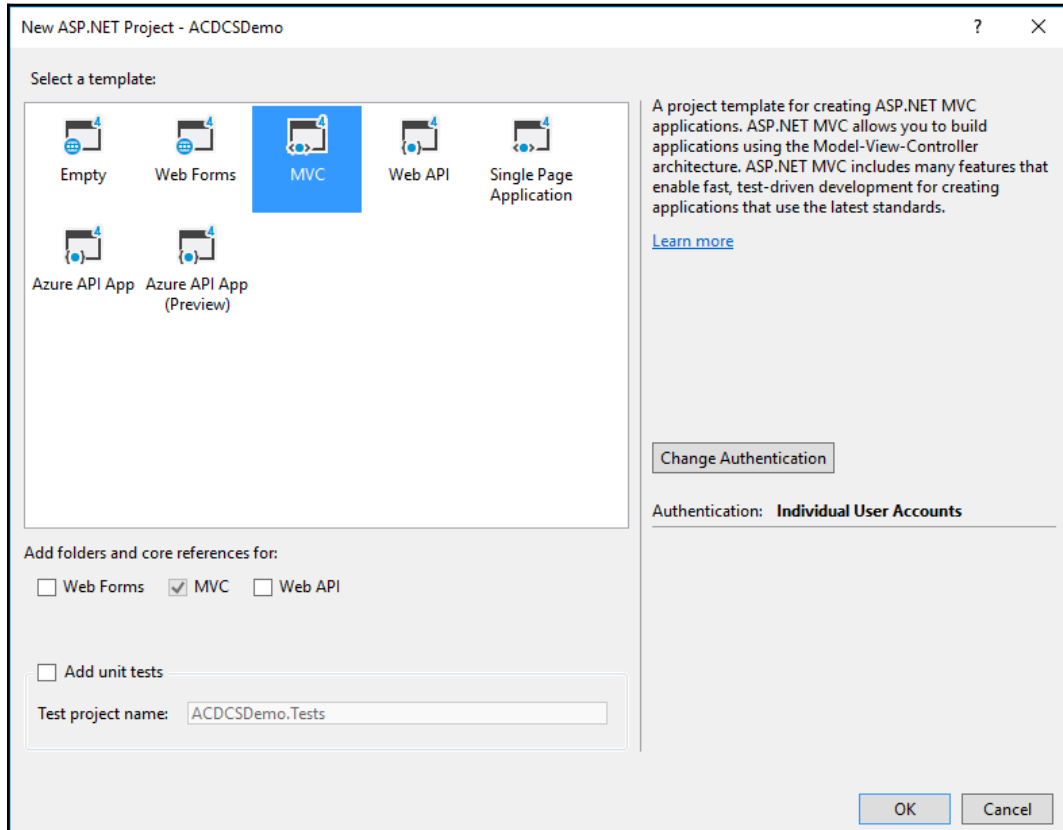
2. **WebRole1** is not a meaningful name. Of course, we want to change that, so please click on the Edit button (button with the pencil icon):



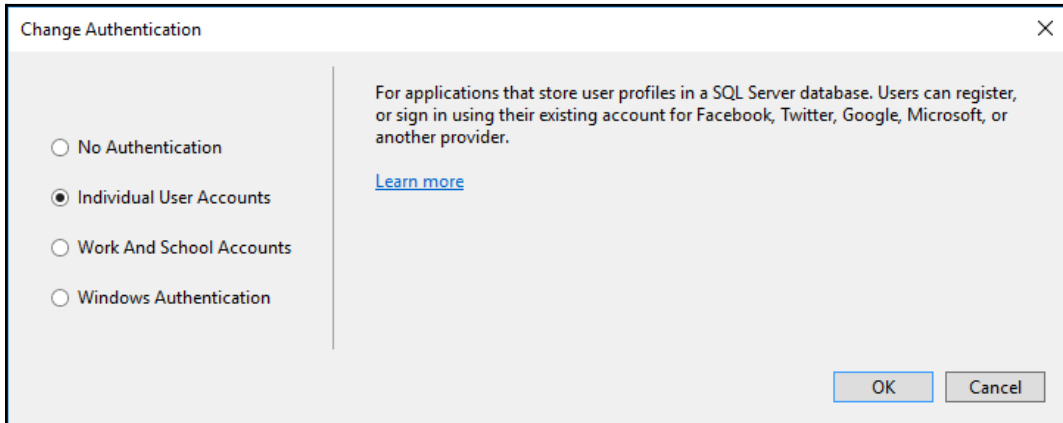
3. Now write the role name of your choice (for example, ACDCSDemo) and press the OK button:



4. We have now created our project and defined the role type. Now it's time to refine the type of our project. For our demo, we will use ASP.NET **MVC**:

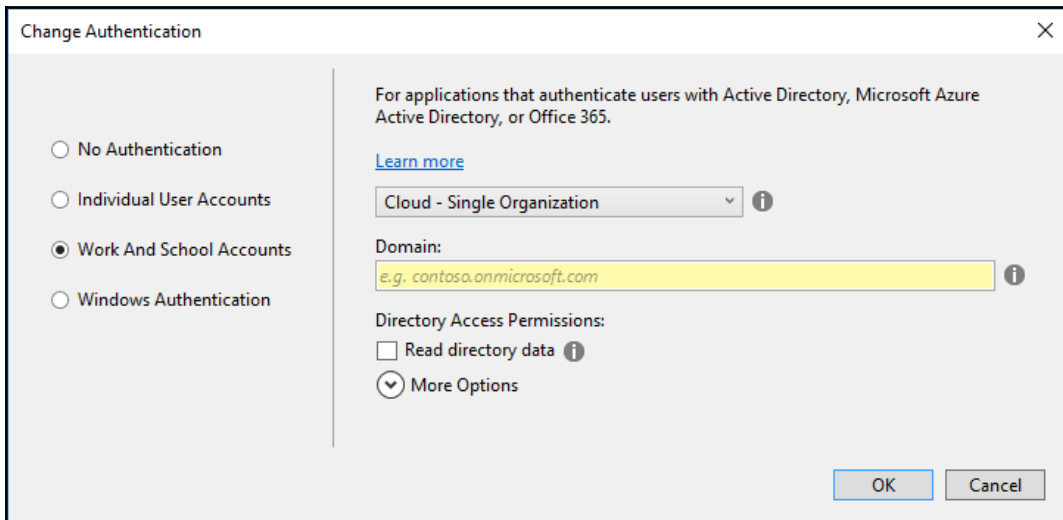


5. If necessary, you can now also change the authentication procedure (simply press the **Change Authentication** button). Since this is not relevant for our demo, we will not pursue it. Changes in the authentication procedure can make individual:



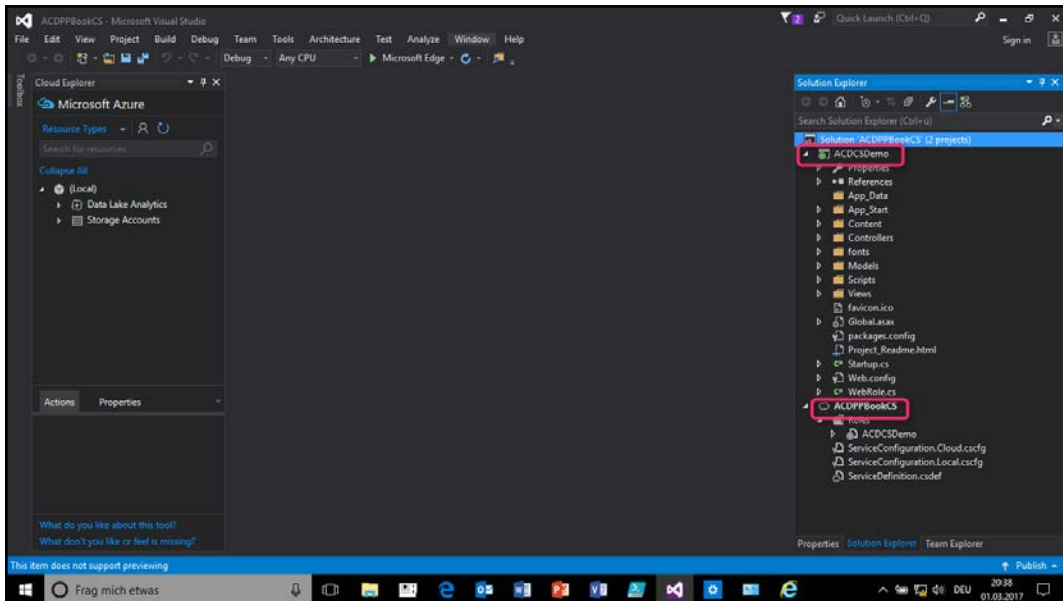
The 'Change Authentication' dialog box is shown. On the left, there are four radio buttons: 'No Authentication', 'Individual User Accounts' (which is selected), 'Work And School Accounts', and 'Windows Authentication'. On the right, there is explanatory text: 'For applications that store user profiles in a SQL Server database. Users can register, or sign in using their existing account for Facebook, Twitter, Google, Microsoft, or another provider.' Below this text is a blue link labeled 'Learn more'. At the bottom right, there are 'OK' and 'Cancel' buttons.

6. Based on Azure AD:



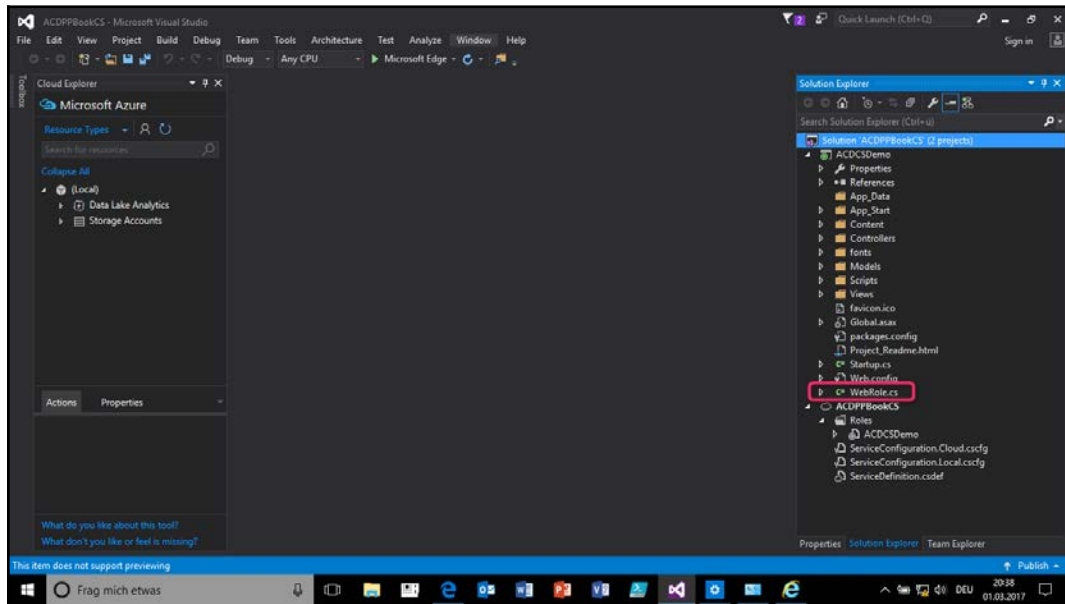
The 'Change Authentication' dialog box is shown. On the left, the 'Work And School Accounts' radio button is selected. On the right, the text reads: 'For applications that authenticate users with Active Directory, Microsoft Azure Active Directory, or Office 365.' Below this is a 'Learn more' link. A dropdown menu is set to 'Cloud - Single Organization'. Below that, the 'Domain:' label is followed by a text box containing 'e.g. contoso.onmicrosoft.com'. Under 'Directory Access Permissions:', there is an unchecked checkbox for 'Read directory data' and a selected dropdown for 'More Options'. 'OK' and 'Cancel' buttons are at the bottom right.

7. To conclude, please press the **OK** button in the project type selection dialog. Then your project solution is created, according to your requirements.
8. After a few minutes the project solution will become available. It consists of the following two projects:
 - The actual web project (website or web application)
 - The role project

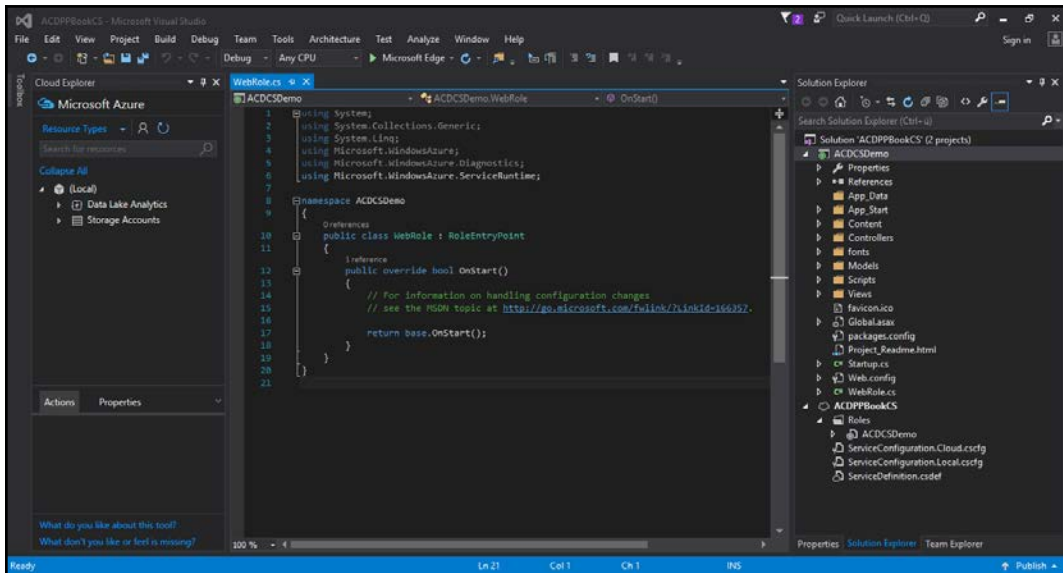


9. Next, we will look at the most important components of the solution.

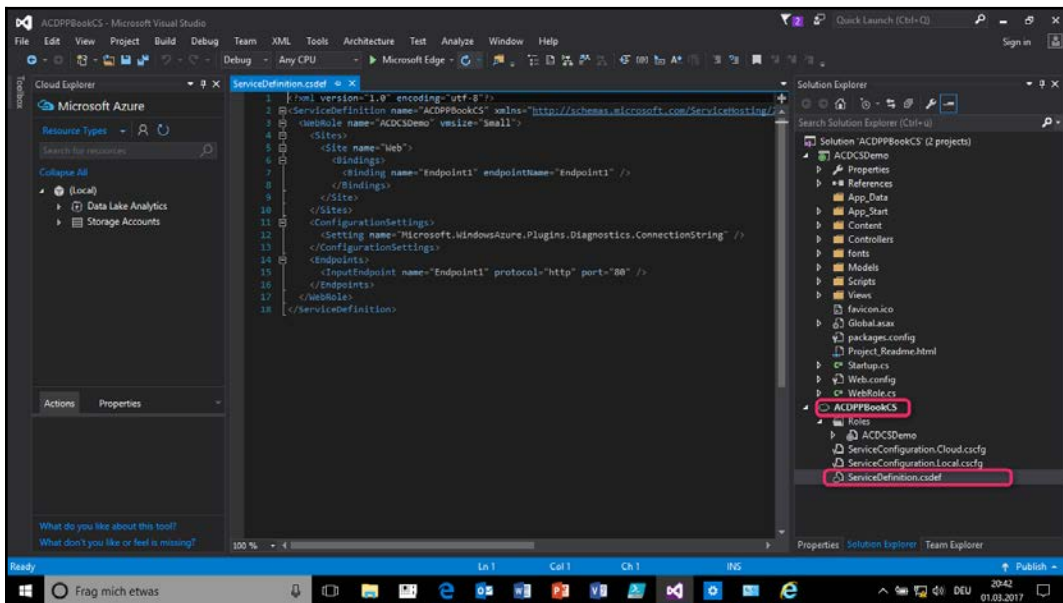
10. In the web project, you will find the file `WebRole.cs`. This is the position where you realize the program logic of your role:



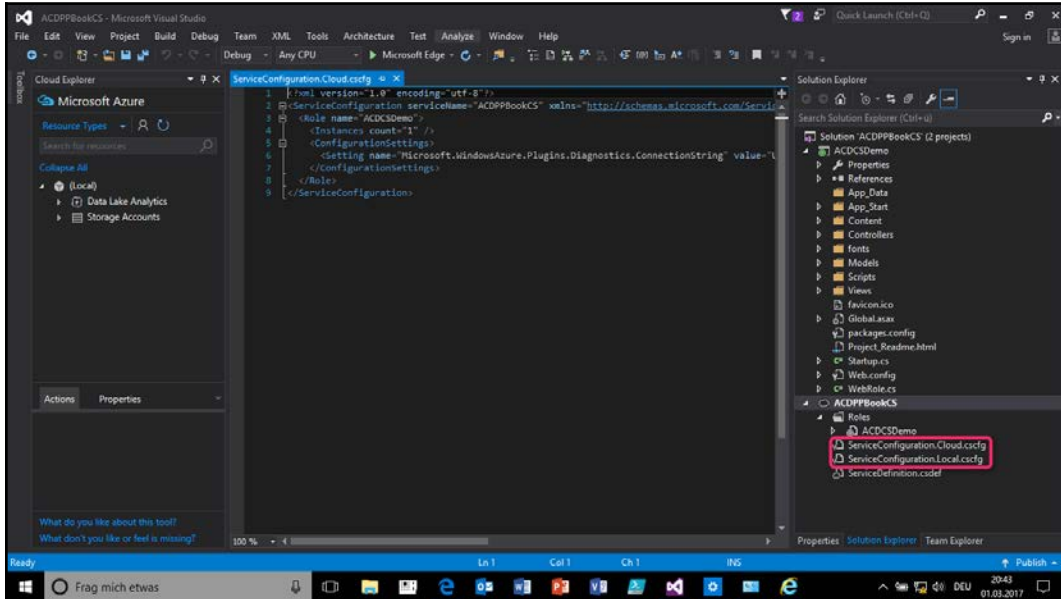
11. A double-click on the filename opens a template, which you can edit according to your ideas:



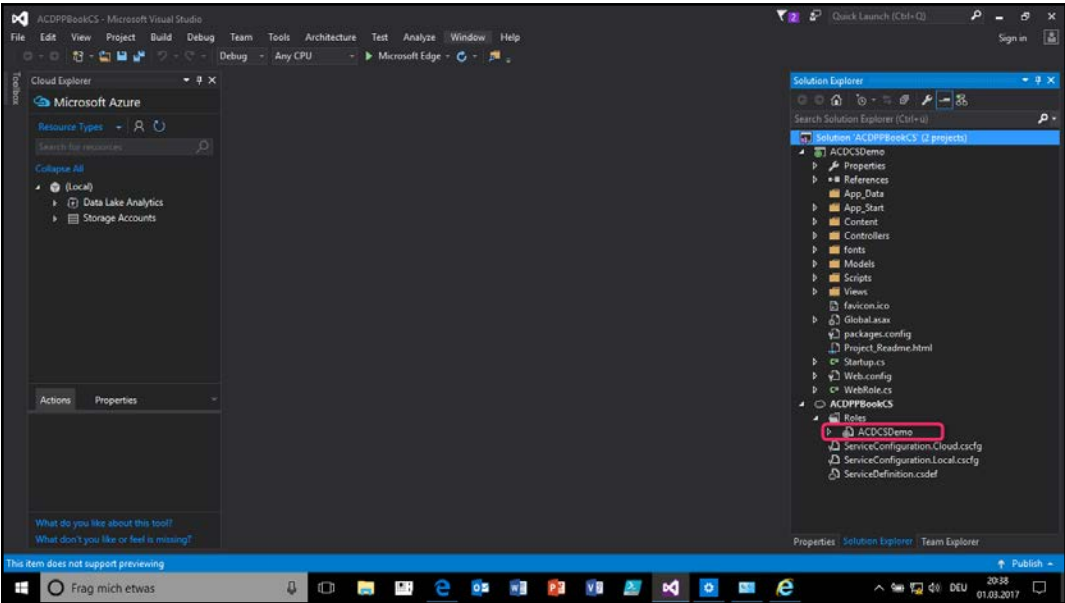
12. In the role project, you will find the templates for the Service Definition File, and the Service Configuration File:



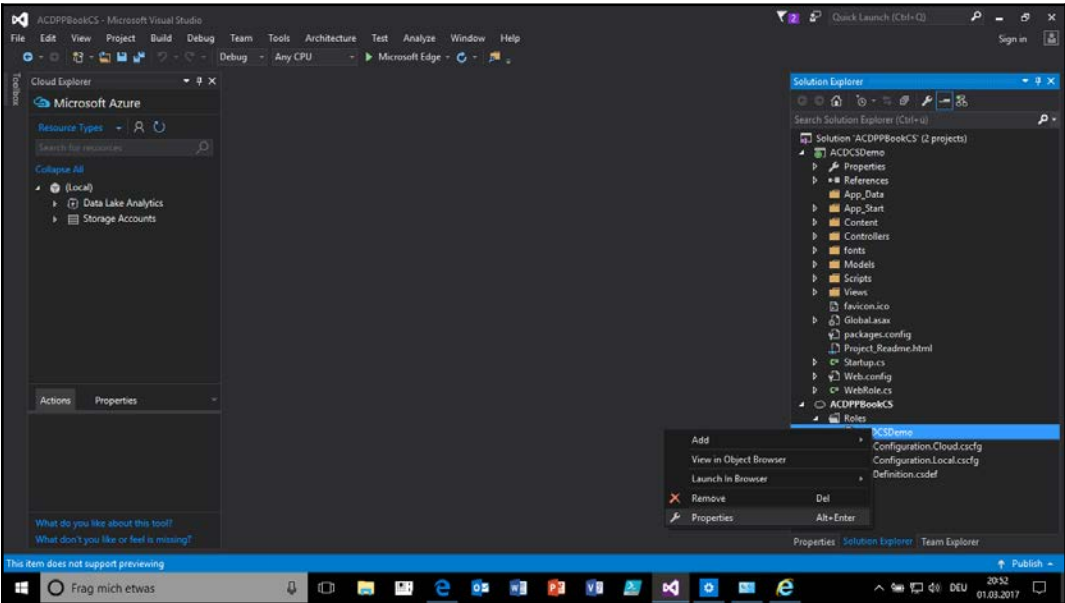
13. The Service Configuration File is available in two variations (*cloud* and *local*). This allows you to separate configuration settings in the local development cycle:



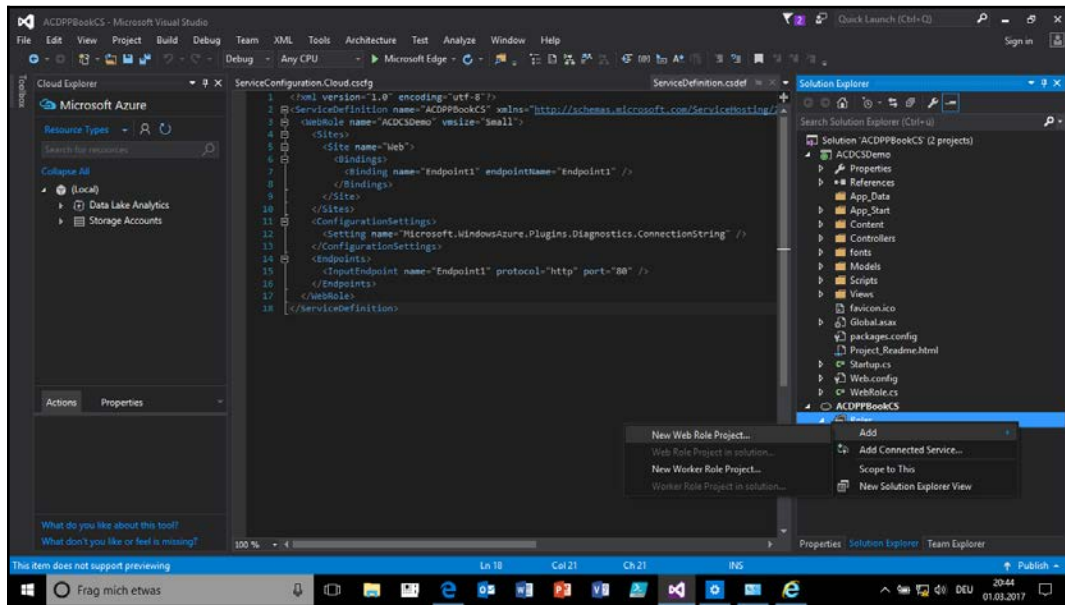
14. What's next? In our demo, we have only defined one role. However, a typical Cloud Service solution usually consists of several roles. How can we define other roles? Very easily! Please right-click on the field marked in the following screenshot:



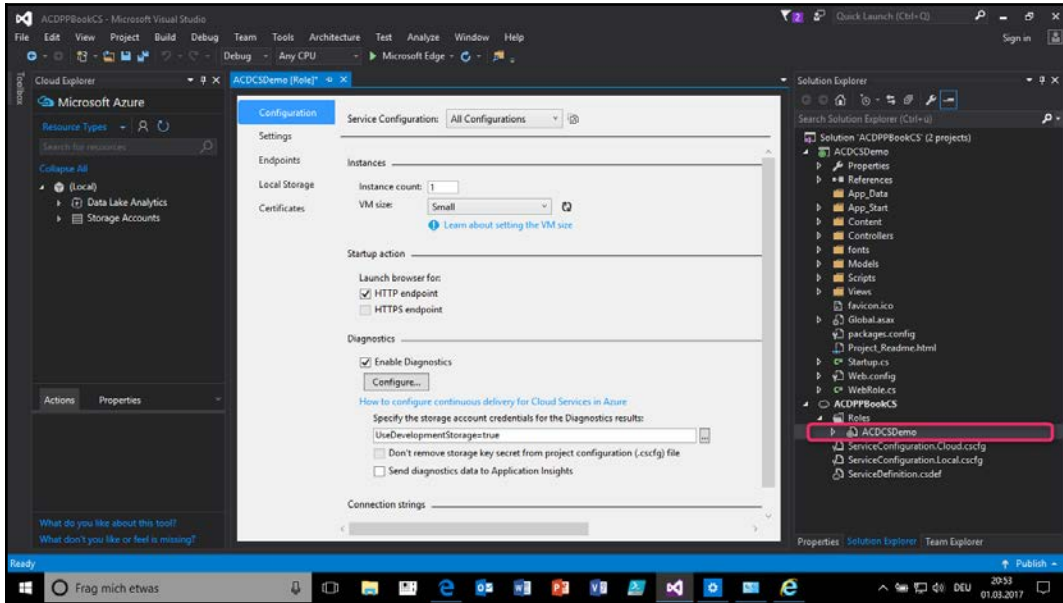
15. Select **Add**:



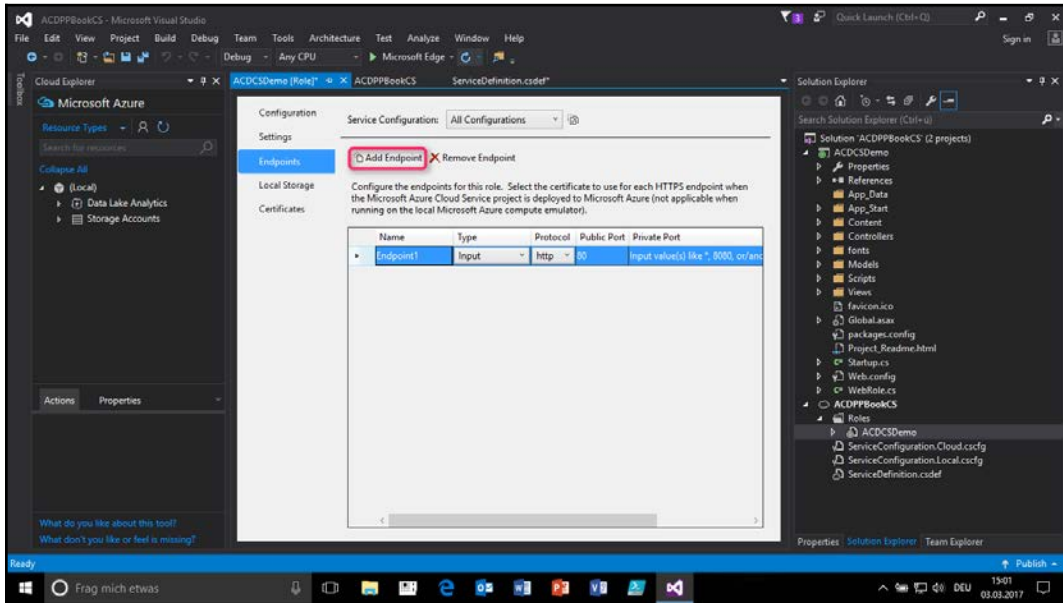
16. Now you can add another role to your solution with the commands **New Web Role Project...** or **New Worker Role Project...**:



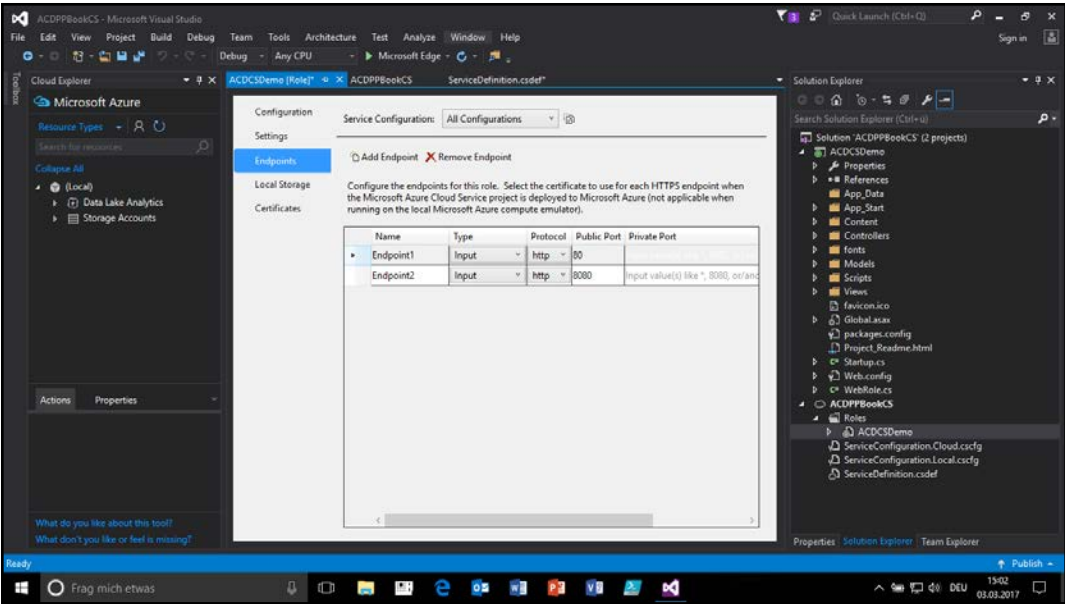
17. And now? The Visual Studio also offers opportunities to edit the Service Definition and the Service Configuration File and not only manually. Please right-click on the field marked in the following screenshot, and then click **Properties**. Now a dialog box with numerous possibilities for individual settings opens. All the settings you define here are immediately applied to the solution:



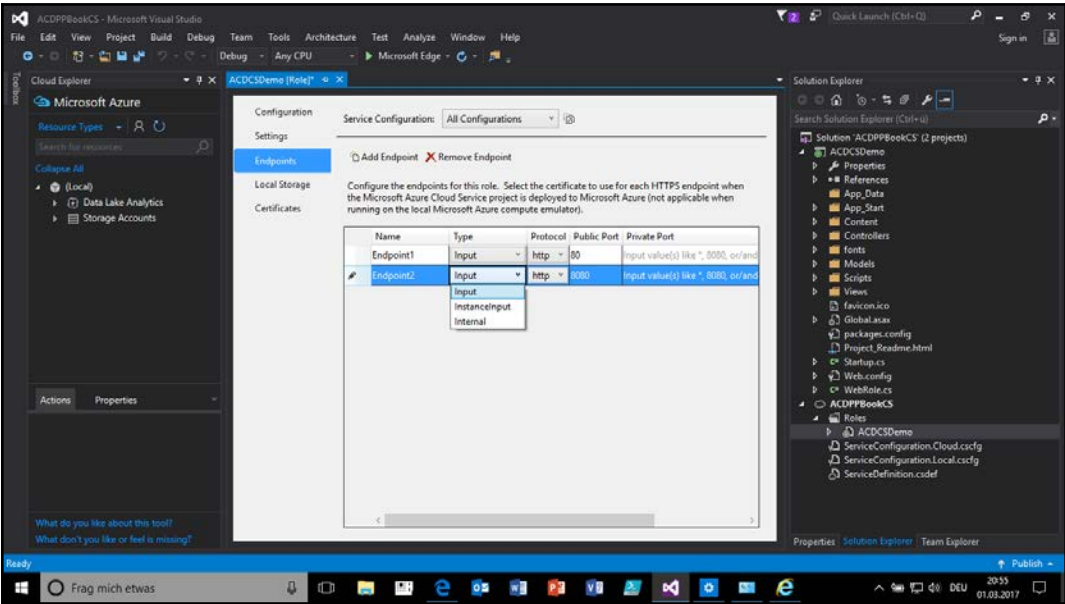
18. For example, please open the **Endpoints** page and then press the **Add Endpoint** button:



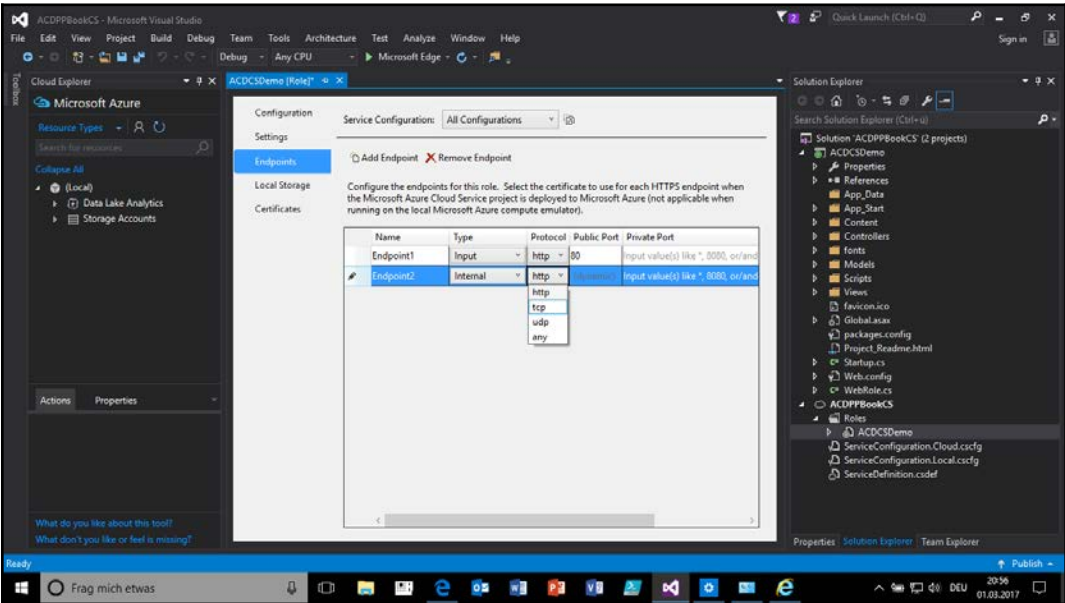
19. Immediately a new endpoint is available:



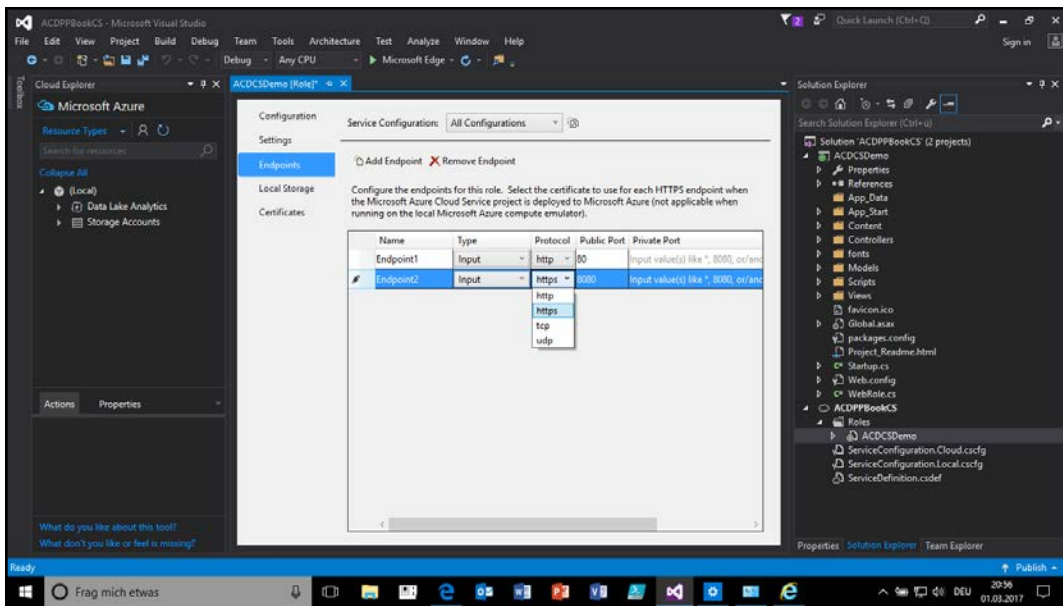
20. Next, select an endpoint **Type**:



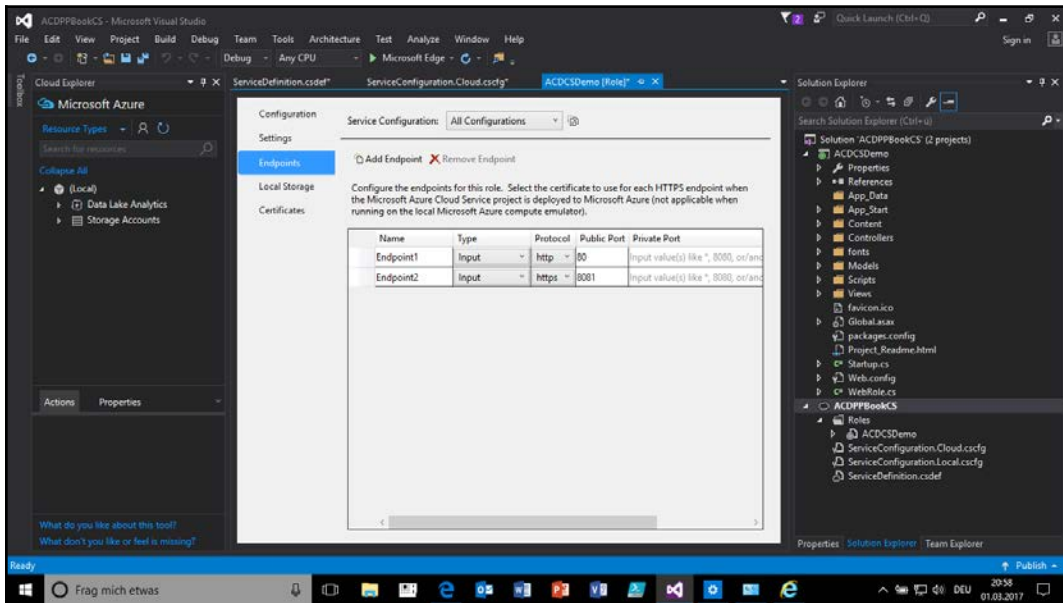
21. And then select a **Protocol**:



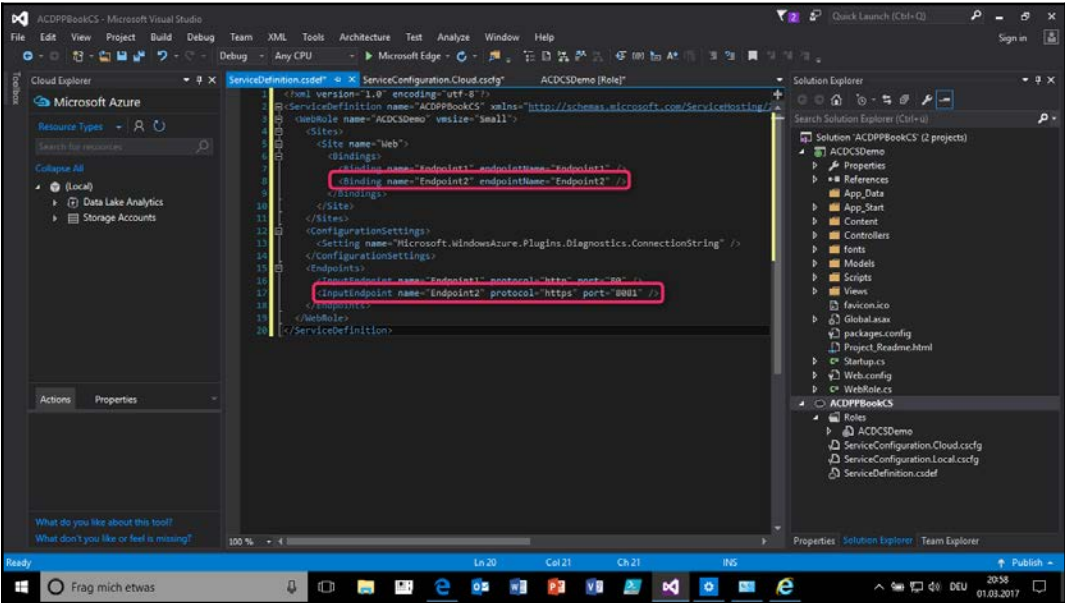
22. The range of protocols varies with the different endpoint types:



23. Ready, a new endpoint is defined:

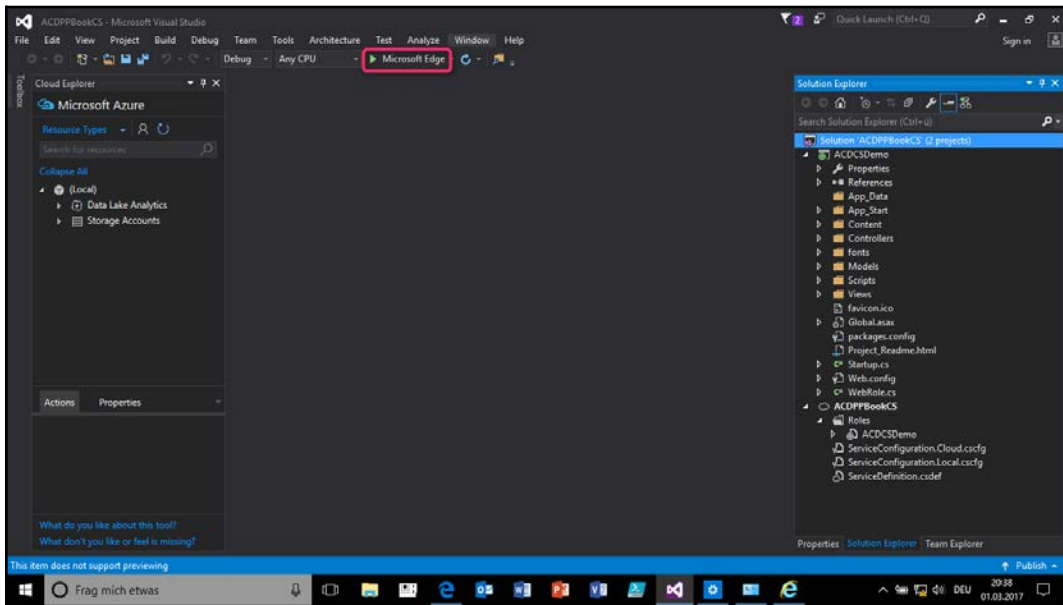


24. The supplements were automatically adopted. Here is the result as evidence:

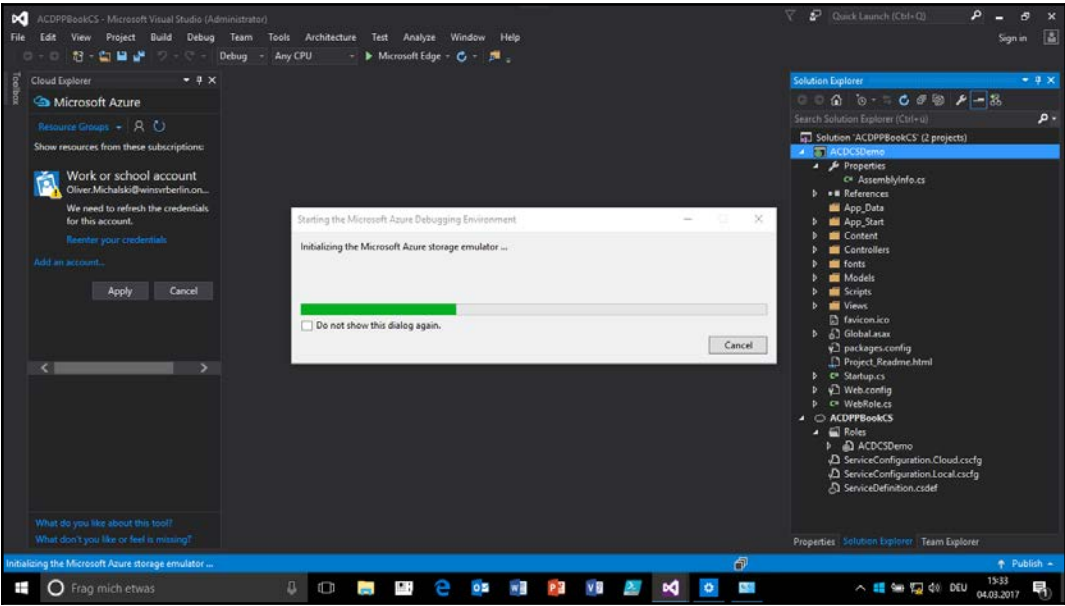


```
1 <?xml version="1.0" encoding="utf-8"?>
2 <ServiceDefinition name="ACDPPBookCS" xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">
3   <WebRole name="ACDCSDemo" vmSize="Small">
4     <Sites>
5       <Site name="Web">
6         <bindings>
7           <binding name="Endpoint1" endpointName="Endpoint1" />
8           <binding name="Endpoint2" endpointName="Endpoint2" />
9         </bindings>
10      </Site>
11    </Sites>
12    <ConfigurationSettings>
13      <Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString" />
14    </ConfigurationSettings>
15    <Endpoints>
16      <inputEndpoint name="Endpoint1" protocol="http" port="80" />
17      <inputEndpoint name="Endpoint2" protocol="https" port="8081" />
18    </Endpoints>
19  </WebRole>
20 </ServiceDefinition>
```

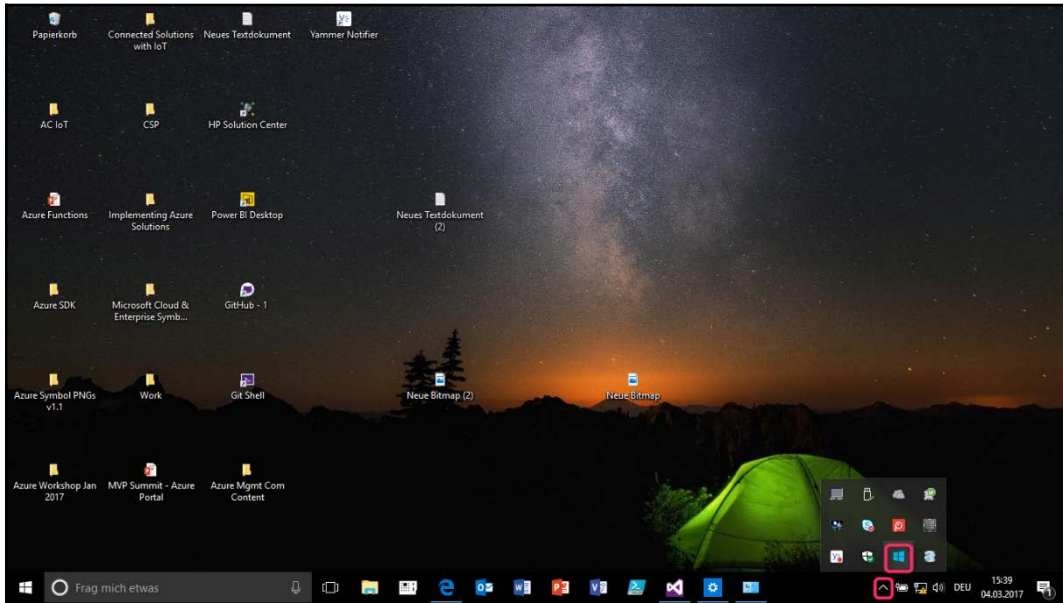
25. The next step is the local test run. Please click on the **Run** button marked in the following screenshot or press the *F5* key:



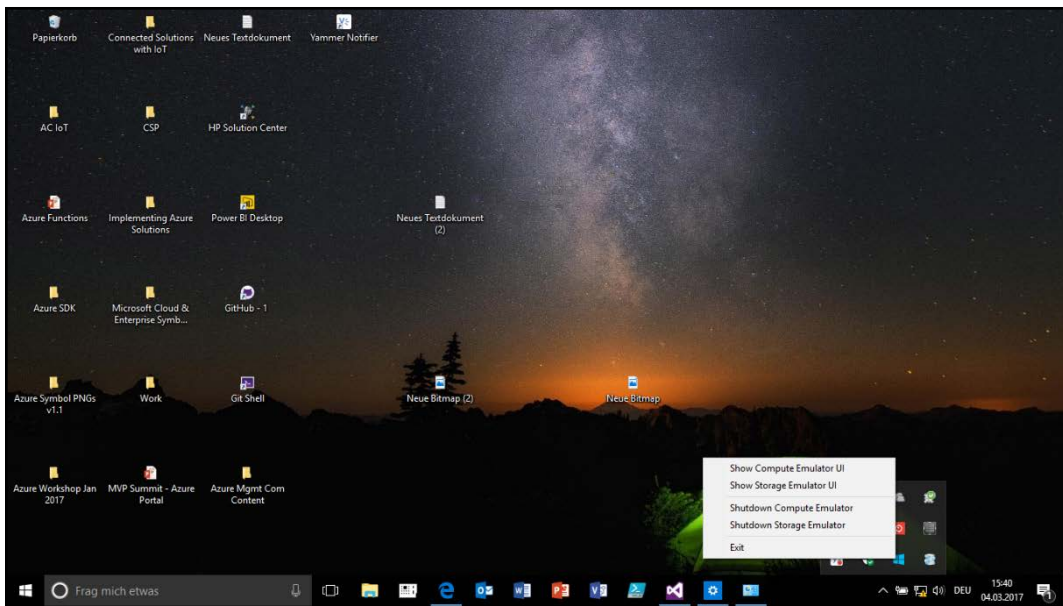
26. Now the **Microsoft Azure Debugging Environment** is starting up:



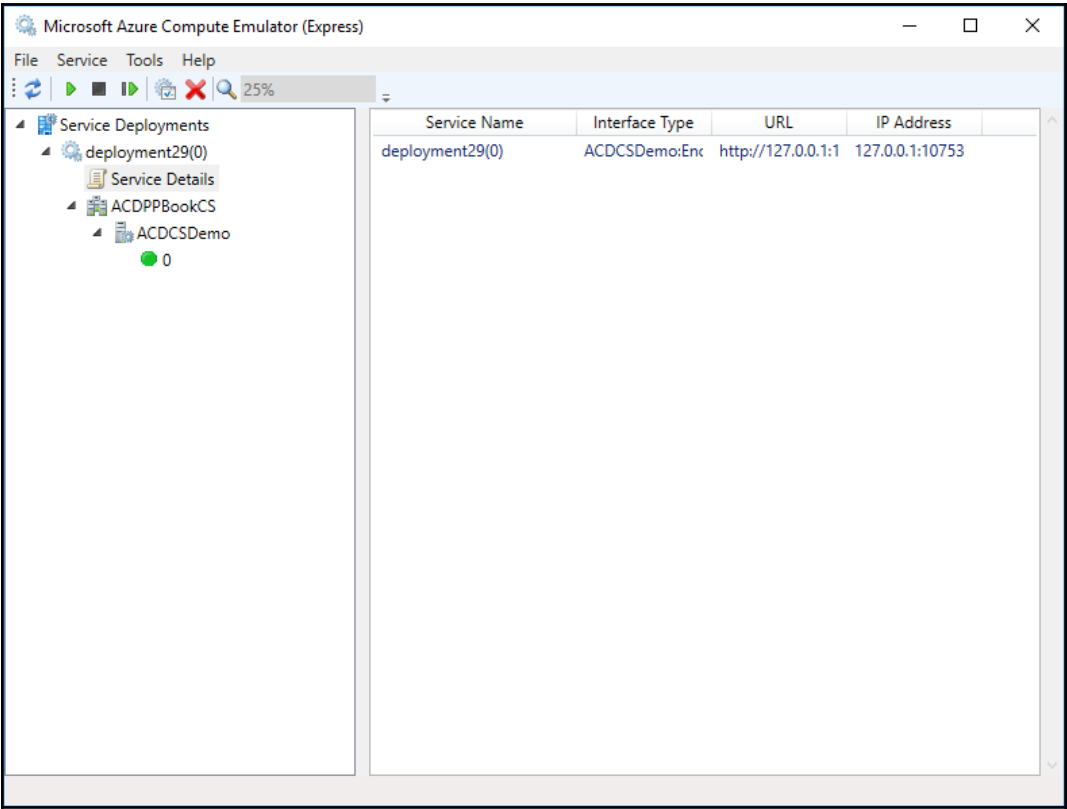
27. You can access the **Microsoft Azure Debugging Environment** over the task bar from your PC. Look for the Azure icon:



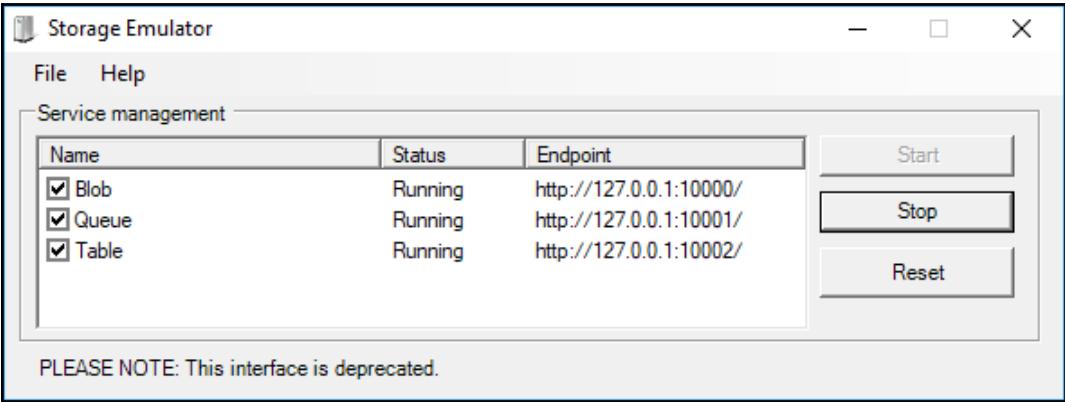
28. Please right-click on the icon:



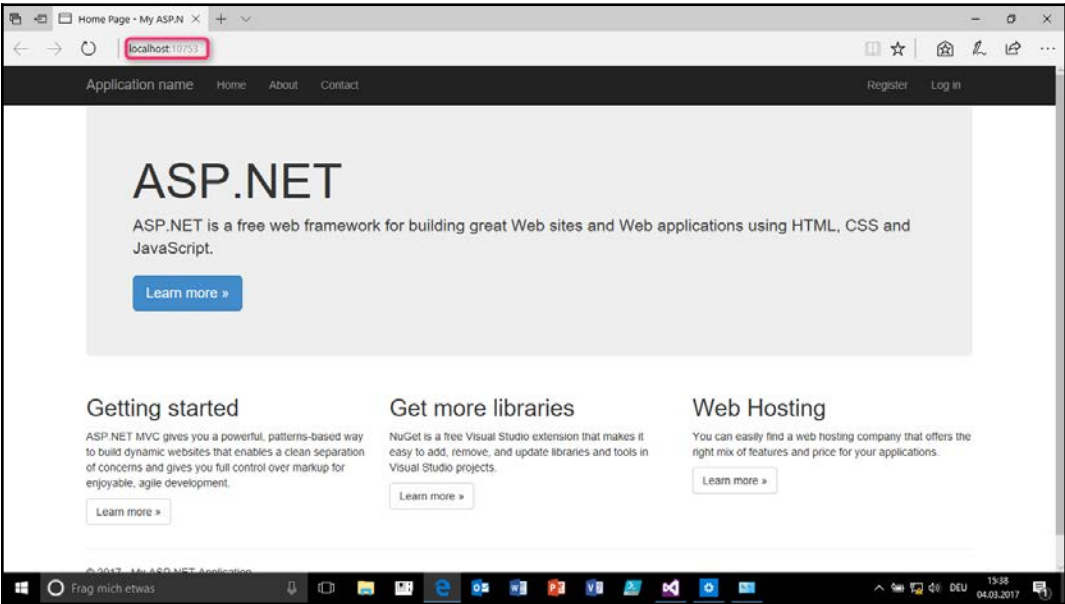
29. Here is the **Microsoft Azure Compute Emulator (Express)**:



30. And here is the **Storage Emulator**:



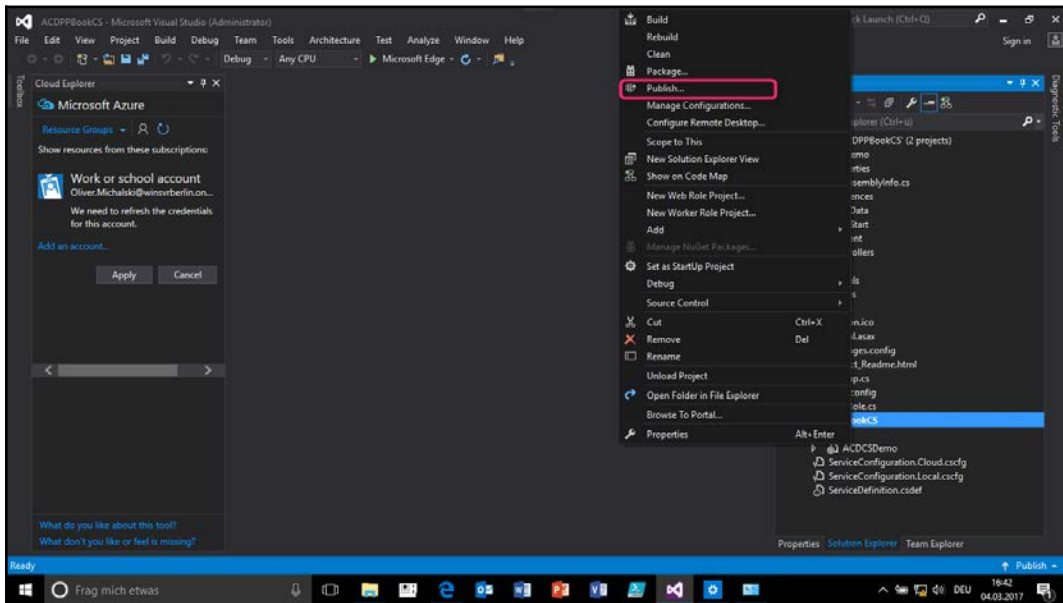
31. At the same time, your web application is loaded into the browser.





The Microsoft Azure Debugging Environment, with its tools, helps you develop your service, for example, providing you with debugging capabilities. Since the focus of our book is on the implementation of Azure solutions, we will not pursue this further. More information can be found here: <https://docs.microsoft.com/en-us/azure/vs-azure-tools-debug-cloud-services-virtual-machines>.

32. The last step is deployment to the Azure platform. There are two possibilities for this. The basis for both ways is a right-click on the role project. For the first option, select the **Publish** command:



33. Now an extensive dialogue is opened. Numerous inputs are expected here:

1. On the first page, **Sign in** you must connect with your Azure account. Sign In with the same credentials that you use to login to the Azure Portal:

Publish Azure Application

Microsoft Azure Publish Sign In

Sign in

Settings

Diagnostics

Summary

Microsoft account
oliver-michalski@t-online.de

Choose your subscription:
Microsoft Azure Sponsorship

[Online privacy statement](#)

< Previous

Next >

Publish

Cancel

2. On the second page, **Settings | Common Settings**, most of the information are already assigned. However, you should change the **Environment** setting to **Staging** and ensure that **Build Configuration** is set to **Release**:

Publish Azure Application

 **Microsoft Azure Publish Settings**

Sign in
Settings
Diagnostics
Summary

Common Settings | Advanced Settings

Cloud Service:
acdppbookcs2 (West Europe)

Environment:
Staging

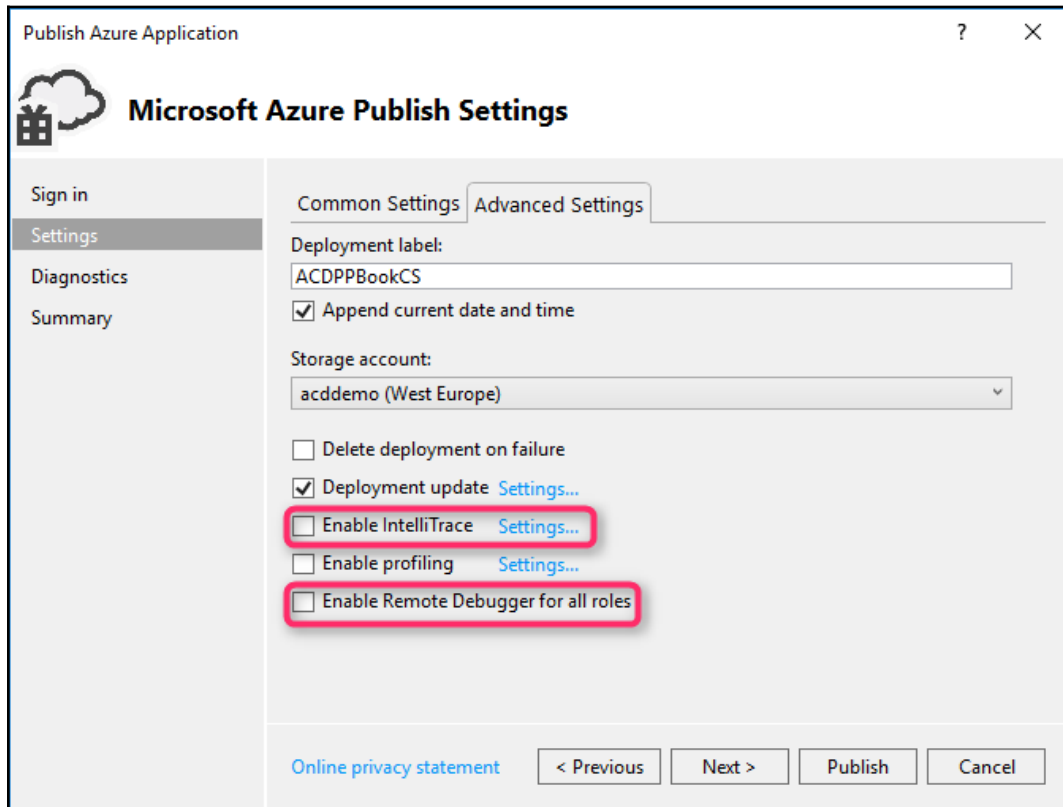
Build configuration:
Release

Service configuration:
Cloud

☐ Enable Remote Desktop for all roles [Settings...](#)
☐ Enable Web Deploy for all web roles (requires Remote Desktop)

[Online privacy statement](#) < Previous Next > Publish Cancel

- On the third page, **Settings | Advanced Settings**, the data are also already assigned and can be adopted unchanged. On this page, you have also the possibility to activate remote debugging for later operations. Simply mark the checkboxes **Enable IntelliTrace** and **Enable Remote Debugger** for all roles:



The screenshot shows the 'Publish Azure Application' dialog box with the 'Advanced Settings' tab selected. The 'Deployment label' is 'ACDPPBookCS' and 'Append current date and time' is checked. The 'Storage account' is 'acddemo (West Europe)'. The 'Enable IntelliTrace' and 'Enable Remote Debugger for all roles' checkboxes are highlighted with red boxes. The 'Enable Remote Debugger for all roles' checkbox is also highlighted with a red box.

Publish Azure Application

Microsoft Azure Publish Settings

Sign in
Settings
Diagnostics
Summary

Common Settings Advanced Settings

Deployment label:
ACDPPBookCS

☒ Append current date and time

Storage account:
acddemo (West Europe)

☐ Delete deployment on failure

☒ Deployment update [Settings...](#)

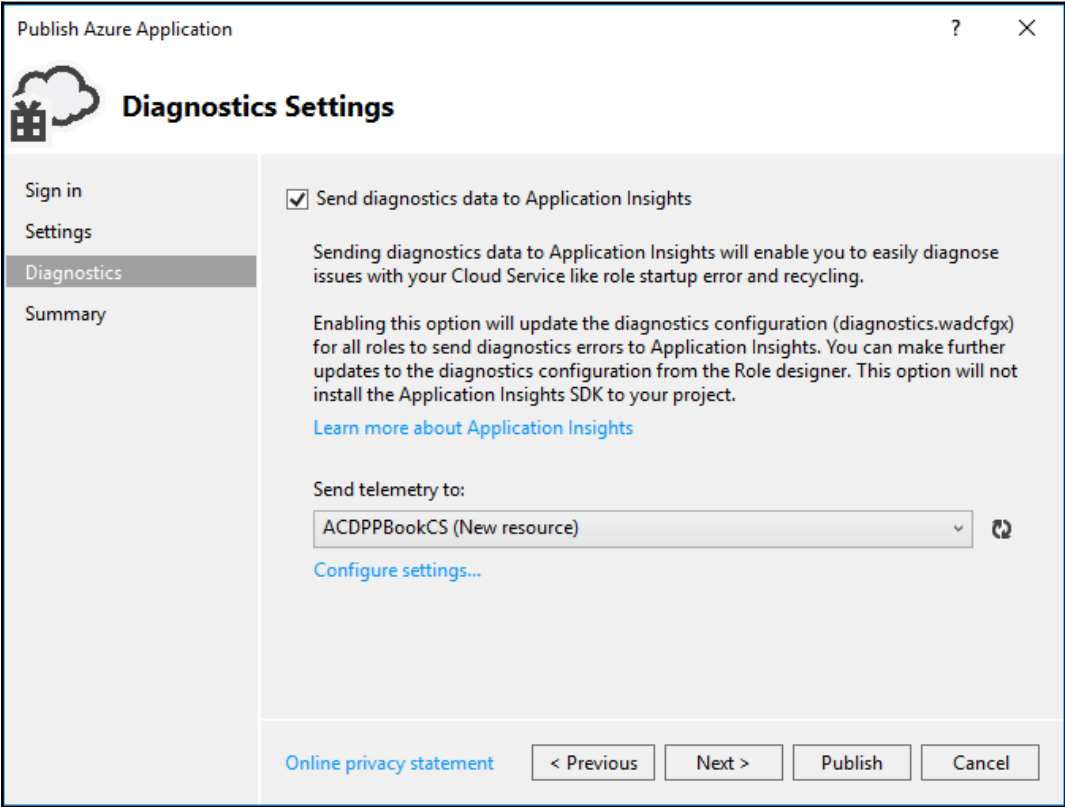
☐ Enable IntelliTrace [Settings...](#)

☐ Enable profiling [Settings...](#)

☐ Enable Remote Debugger for all roles

[Online privacy statement](#) < Previous Next > Publish Cancel

4. The **Diagnostics Settings** page is something special, since you can link your Cloud Services to Application insights:



Publish Azure Application

 **Diagnostics Settings**

Sign in
Settings
Diagnostics
Summary

☒ Send diagnostics data to Application Insights

Sending diagnostics data to Application Insights will enable you to easily diagnose issues with your Cloud Service like role startup error and recycling.

Enabling this option will update the diagnostics configuration (diagnostics.wadcfgx) for all roles to send diagnostics errors to Application Insights. You can make further updates to the diagnostics configuration from the Role designer. This option will not install the Application Insights SDK to your project.

[Learn more about Application Insights](#)

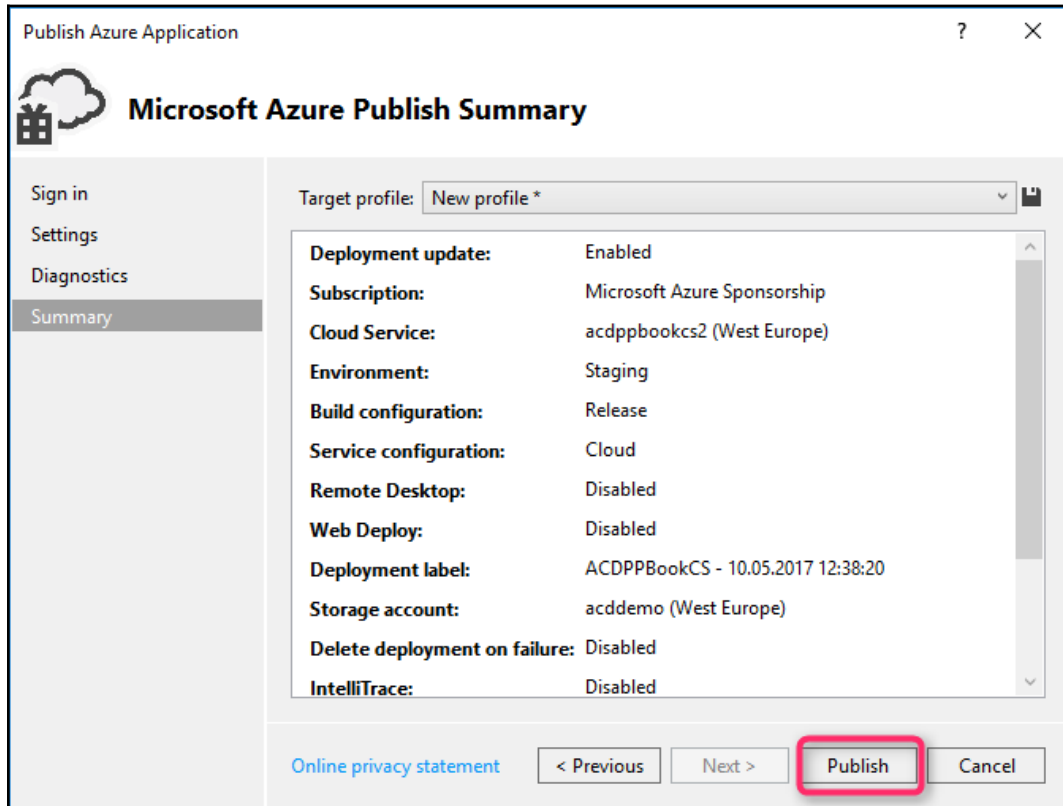
Send telemetry to:

ACDPPBookCS (New resource) 

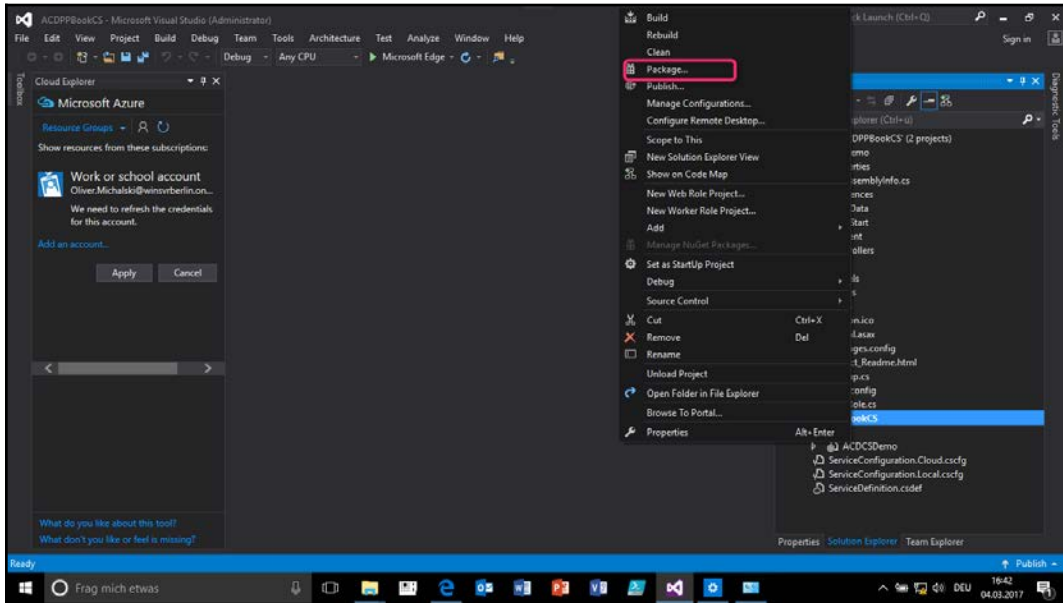
[Configure settings...](#)

[Online privacy statement](#)

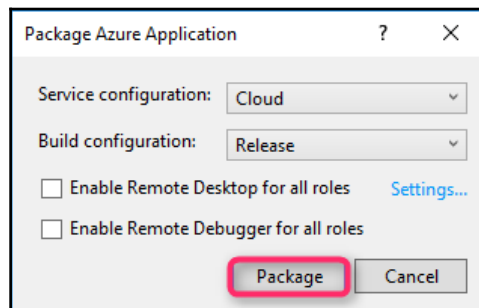
5. Finally, if you are using this path for your deployment, you only need to press the **Publish** button on the **Summary** page:



34. Let's now turn to the second option (the classical way). For this option, you must select the **Package...** command:



35. This time only a simple window opens, where you must press the **Package** button:



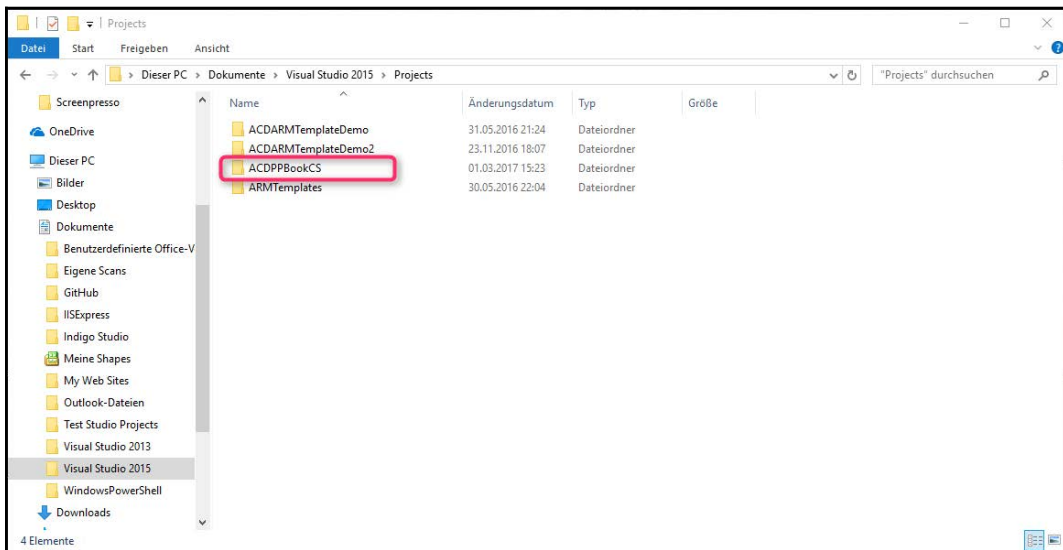
The question arises: what is behind it?

Simple answer-in the initial phase of the Azure platform, there were no publishing procedures. Cloud Services were manually packaged and then uploaded with Azure PowerShell or the Azure management portal.

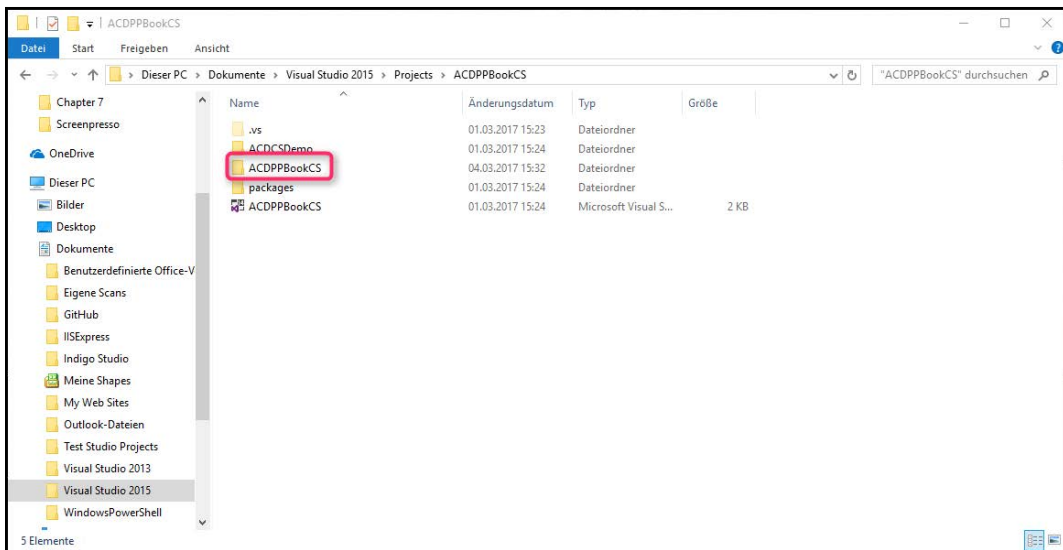
This process is taken up again, only this time the packing as a background action takes place.

Next question: where can we find the new service package?

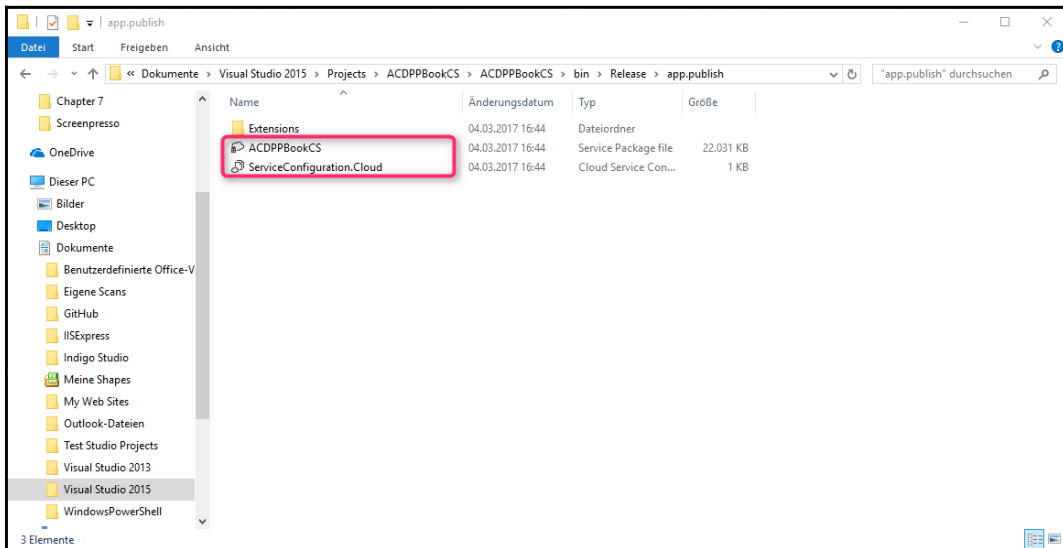
1. Go to the `Projects` folders of Visual Studio. There you can open your Cloud Service solution:



2. Now open the role project:



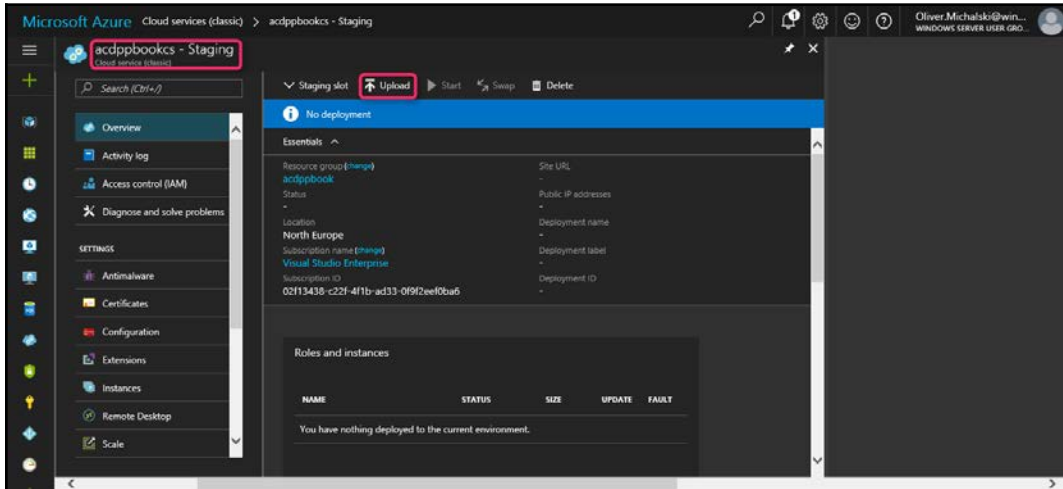
3. Now you must opened the bin folder, then the folder that matches to your build (Release or Debug) There, you can find the service package and also the Service Configuration File in the app.publish subfolder :



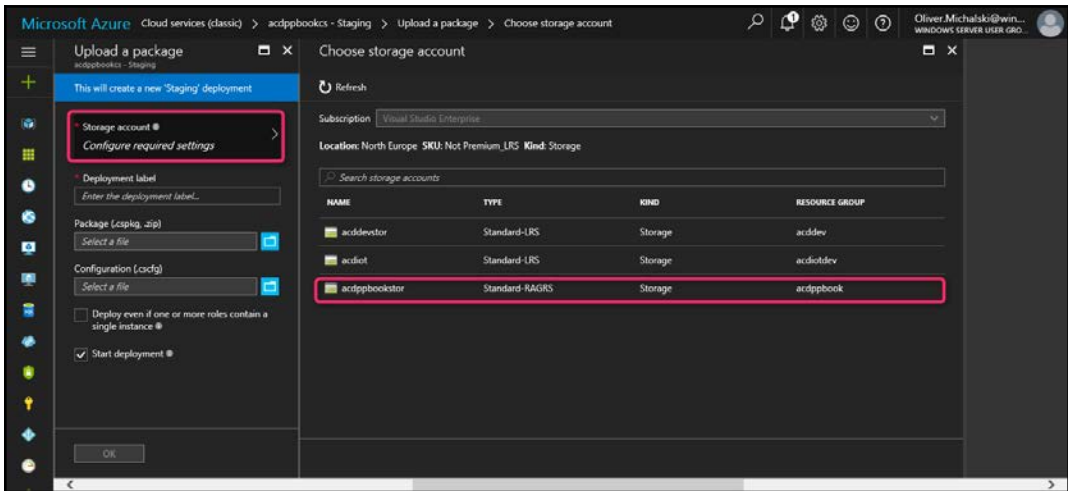
4. Now it's time to return to the Azure platform, where we stopped before, the dashboard of our Cloud Service.

First please make sure you are in the **Staging** slot:

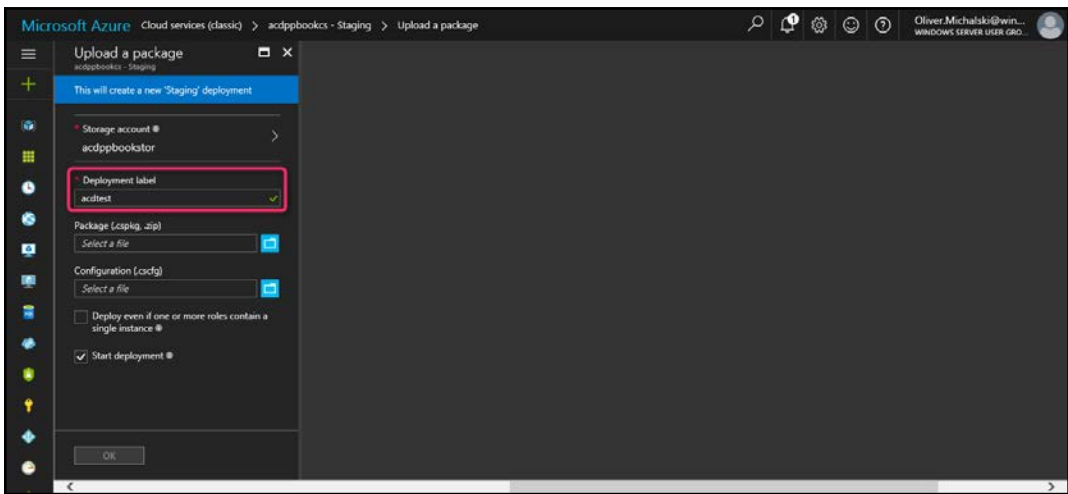
1. In the upper navigation bar, you will find the **Upload** button. Please click on it:



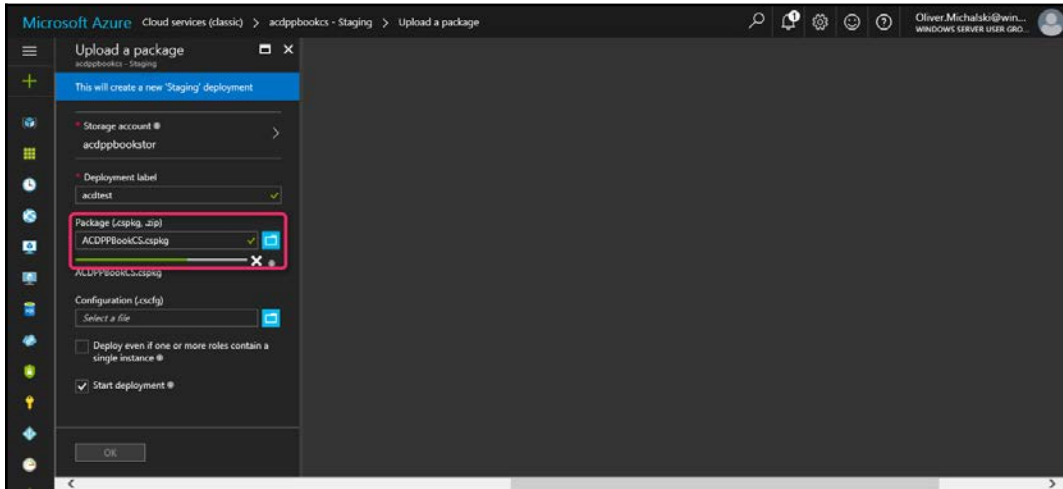
2. Now open the **Upload a package** dialog. Here you must first select an Azure **Storage account** (you can find information about creating a Storage account, in Chapter 5, *Implementing and Securing Azure Storage Accounts*), where you want to upload the service package:



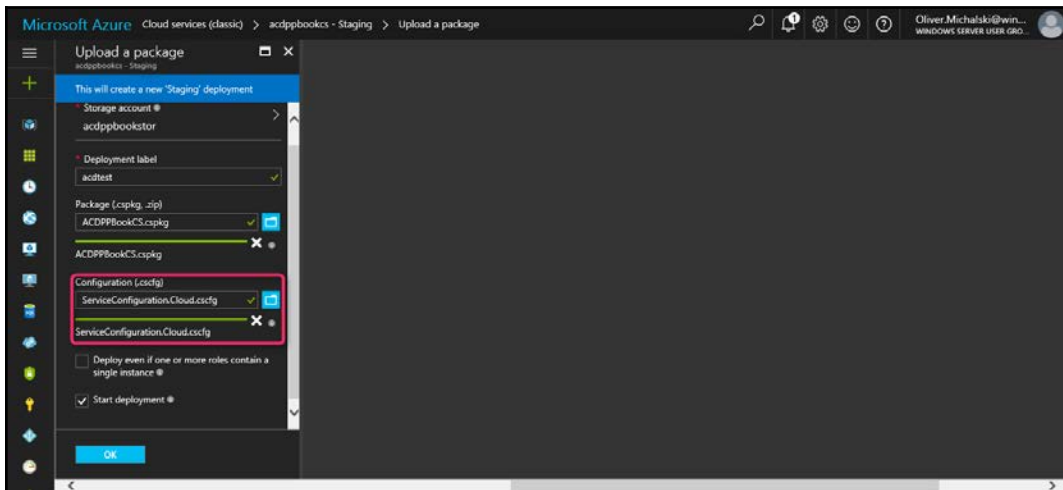
3. Then type a name for the **Deployment label**:



4. First upload the service package and after this the Service Configuration File:

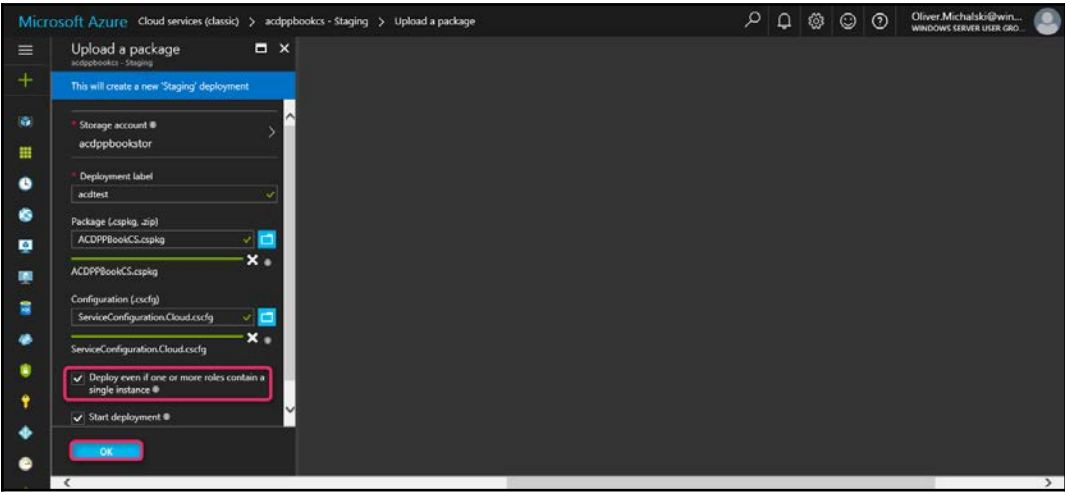


5. The green progress bar gives you information about the status of the upload:

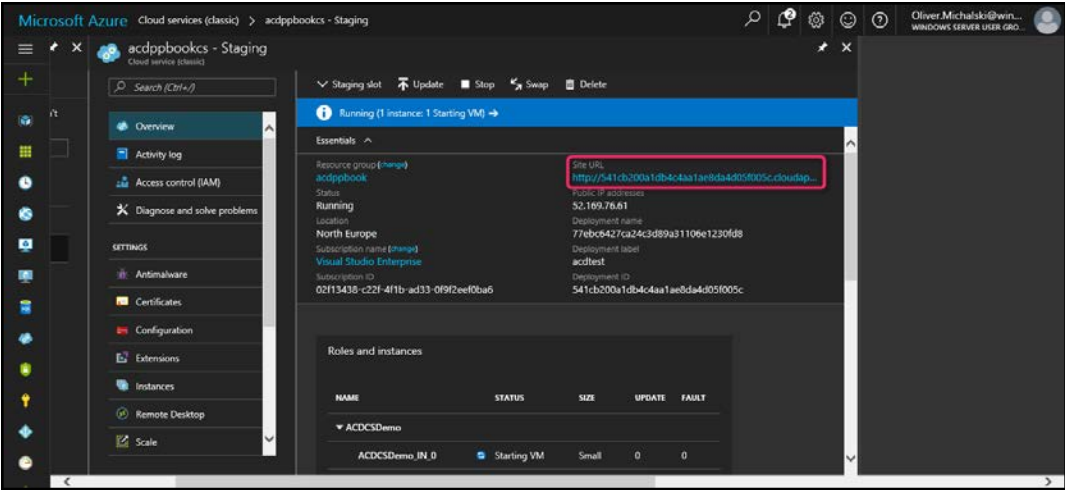
**Attention!**

Since we use only one role instance in our demo, please ensure that the checkbox has been marked in the following screenshot. Otherwise the deployment process would abort with an error.

6. The last action is to press the **OK** button:



7. Ready! Your Cloud Service is now available on the Azure platform:



Summary

In this chapter, we learned all about the Azure Cloud Service offer.

In the first part of this chapter, we learned the basic knowledge of the Azure Cloud Service, all about the Cloud architecture and the difference between Azure Cloud Service and Azure App Services.

Then in the second part of this chapter, there followed a detailed description of the create process.

In the next chapter, we will continue with the newest service of the available Azure PaaS offers, the Azure Container Service.

8

Implementing Azure Container Service

Azure Container Services (ACS) is the newest part of the Azure platform and at present, at least partially in preview status (for example, in the area Windows container).

Azure Container Services is a new variant of the classical Azure IaaS offer from Azure (described in *Chapter 6, Planning and Deploying Virtual Machines in Azure*), and use also virtual machines as the technological base.

Two versions of the Azure IaaS are offered, and both are based on virtual machines?

Yes, but Azure Container Services has some advantages over the older solution due to its additional functionality.

I will now take a closer look at these advantages and introduce Azure Container Services in detail.

In this chapter, you will learn:

- The basic knowledge about the Azure Container Services area
- The workflow when creating your first container service and the necessary steps for working with the service afterwards

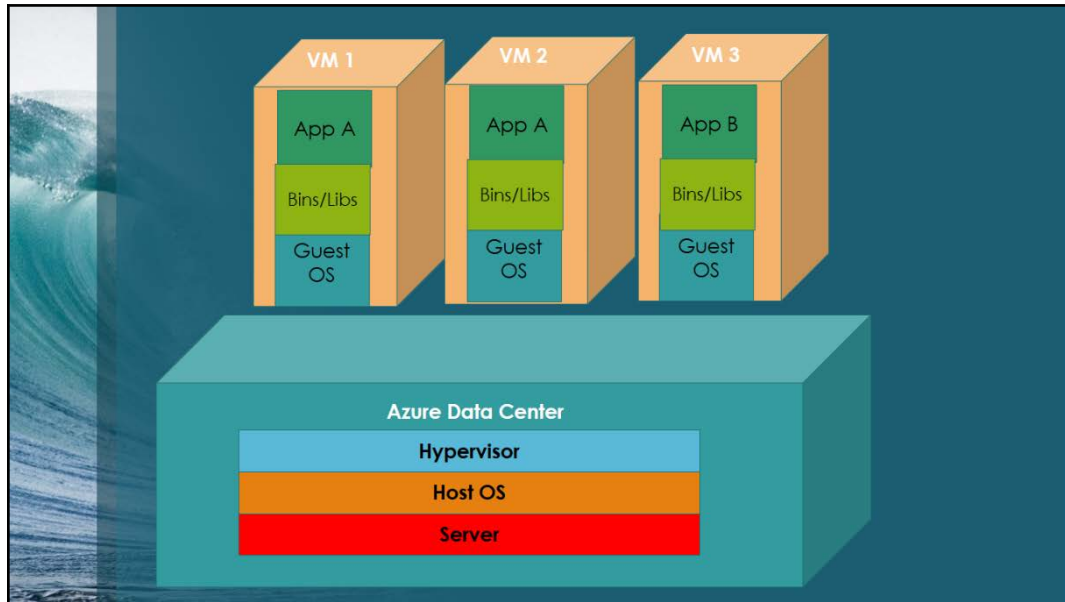
What is an Azure Container Service?

An Azure Container Service helps you to create, configure, or manage a cluster of VMs. These clusters are preconfigured to run applications in so-called containers.

Ok, this is a complete answer, but we should nevertheless, take a deeper look.

Let's start with a comparison between the two Azure IaaS offers.

The following diagram shows you the classic Azure IaaS offer, or in other words, Windows or Linux VMs on Azure:



As you can see, the offer consists of the following two layers:

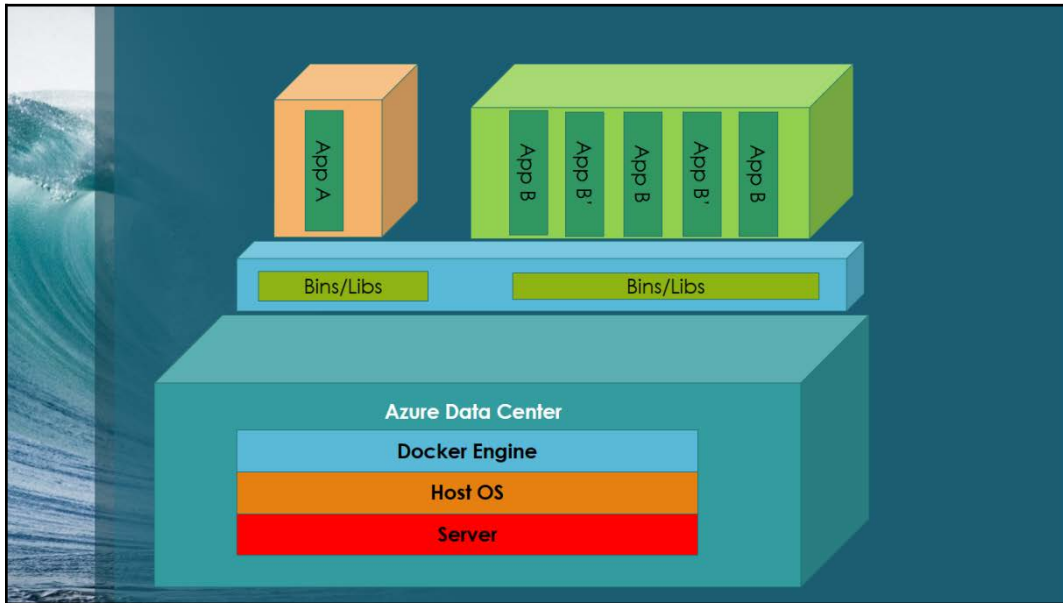
- The infrastructure in the **Azure Data Center**
- The actual VMs

The area infrastructure is easily explained: VMs are an abstraction of physical hardware running on a server and turn one server into many servers. The hypervisor allows you to run several VMs on a single machine.

Now we come to the area VMs—each VM includes one or more apps, the necessary libraries and binaries, and a full guest operating system.

A VM can reach a size in the high GB range and can therefore be very slow when booting.

The next diagram shows you the new Azure IaaS offer, or in other words, the IaaS based on container solution:



At first glance, this image looks like the first image. But there are some differences.

In the infrastructure layer, for example, you will find a container engine instead of the hypervisor. In ACS, the container engine is a **Docker Engine**. Docker is currently the leading open source project for container solutions.

In good Azure tradition, the ACS Engine is an open source project. If you want to examine the engine internals, or if you need to adjust your needs, you can find the complete source code at <https://github.com/Azure/acs-engine>.

Let's come back to the subject. The container engine is the runtime environment for containers (partially comparable to the Guest OS of the VMs). With their help, multiple containers can run on the same machine and share the OS kernel with other containers that run as isolated processes in their own user space.

The next difference in the new offer, you will find in the provision of the necessary libraries and binaries. They are provided by the container engine in an intermediate layer for an individual or as a shared resource for multiple containers.

Understanding Containers

The term has already been mentioned several times, so it is time to ask: what is a container?

Simple answer: a container in our mind is an instance of a Docker image. A **Docker image** is an ordered collection of root filesystem changes and the corresponding execution parameters.



When you scale your Azure Container Service, you typically run multiple containers from the same image.

A container is defined not only by a Docker image, but also by the following components:

- A so-called Dockerfile is a text file that contains necessary information about the structure of the container
- Something singular like a process, a service, or microservice (for example, an Azure Service Fabric) or an app

Containers take up less space than VMs (typically in a low MB range), and start almost instantly.

Now we have gained our first insights into the container issue. To understand the following, however, we must clarify a few more terms.

Cluster

The first term is cluster or Docker cluster.

A cluster bundles several Docker hosts and represents them as a single virtual host. A cluster is thus able to scale very easily and this with an unlimited number of hosts.

Why do I tell you this?

When I speak later about the creation of your first Azure Container Service, it means the creation of a cluster and not the creation of a container application.

Orchestrator

The next term is orchestrator.

An orchestrator is a management tool for Docker clusters and Docker hosts, and enables users to manage their images, containers, and hosts through a GUI.

The orchestrator allows users to administer container networking, configurations, load balancing, service discovery, high availability, host management, and is responsible for running, distributing, scaling, and healing workloads across a collection of nodes.

When you create your ACS cluster, you also decide about the orchestrator you want to use.

The following orchestrators are currently available:

- DC/OS
- Kubernetes
- Docker swarm

Which orchestrator should I choose?

Microsoft does not recommend a specific orchestrator. If you have experience with one of the orchestrators, you can continue to use this experience in ACS.

However, there are trends that indicate that **Datacenter Operating System (DC/OS)** is good for Big Data and IoT workloads. Kubernetes is suitable for cloud-native workloads and Docker swarm is known for its integration with other Docker tools and for its easy learning curve.

Let's take a deeper look at the three orchestrators.

Mesosphere DC/OS

Mesosphere DC/OS is a distributed operating system based on the Apache Mesos distributed systems kernel.

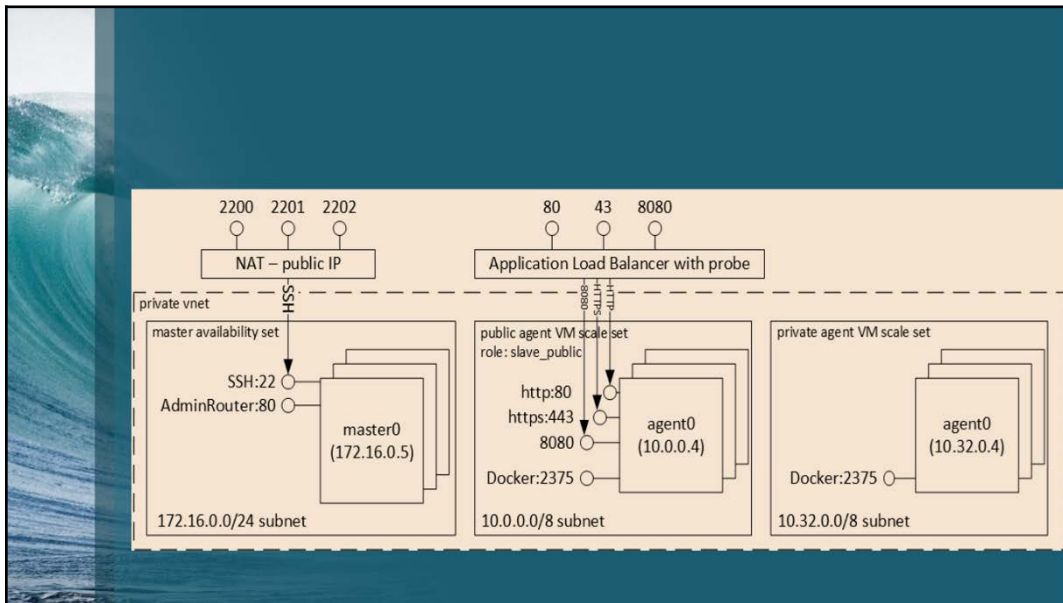
DC/OS (and Apache Mesos) include the following features:

- Scalability
- Fault-tolerant replicated master and slaves using Apache ZooKeeper
- Support for Docker containers

- Native isolation between tasks with Linux containers
- Scheduling for multiple resources (memory, CPU, disk, and ports)
- Java, Python, and C++ APIs
- A web UI for viewing cluster state (Apache Mesos UI)

By default, DC/OS includes the marathon orchestration platform for scheduling workloads and mesosphere universe, a collection of services that can be added to your service.

The following diagram shows the architecture of an ACS cluster using DC/OS:



For further information on mesosphere DC/OS please visit <https://dcos.io/>. You can find the complete source code at <https://github.com/dcos>.

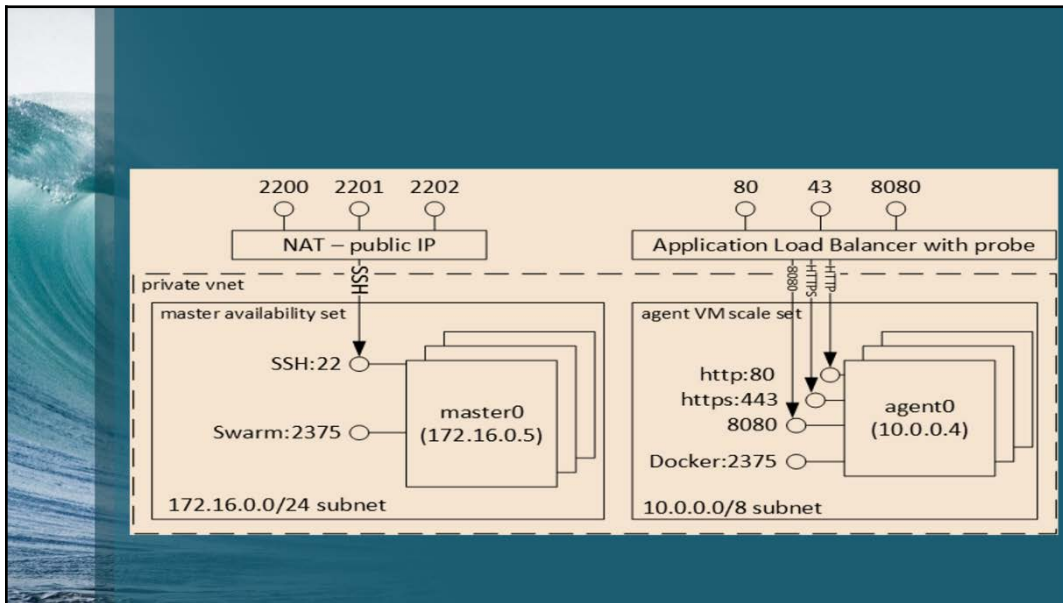
Docker Swarm

Docker swarm offers a system-specific clustering feature for Docker. Because Docker handles Docker swarm's standard API, any tool that already communicates with a Docker daemon can use swarm.

Supported tools for managing of containers on a swarm cluster are:

- Dokku (an ALM tool)
- Docker CLI
- Docker compose
- Jenkins

The following diagram shows the architecture of an ACS cluster using Docker swarm:



For further information on Docker Swarm please click visit <https://www.docker.com/>. The complete source code you can find at <https://github.com/docker>.

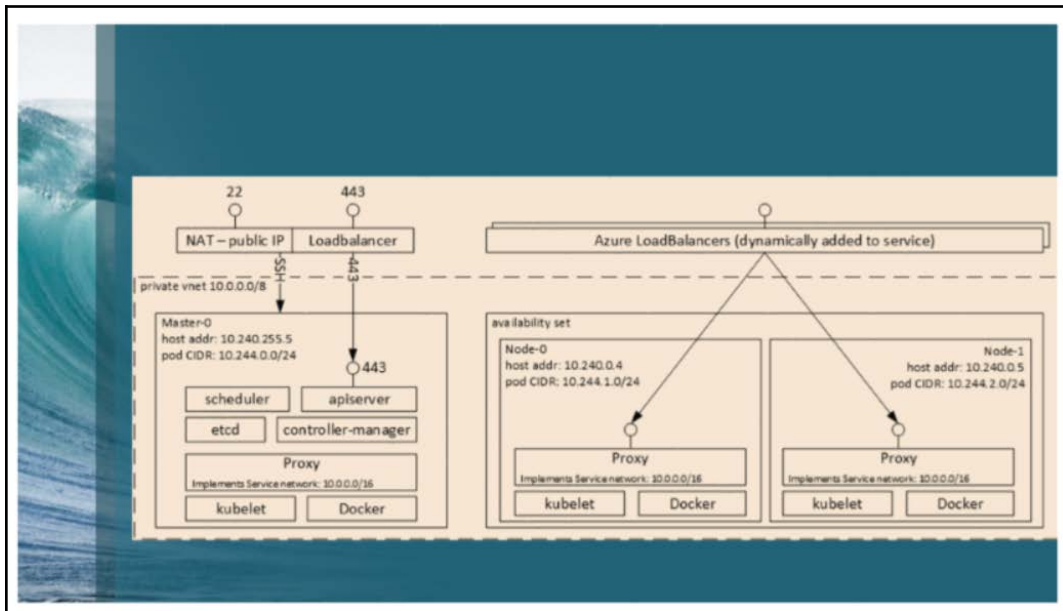
Kubernetes

Kubernetes is a popular production-grade container orchestrator tool. Kubernetes automates deployment, scaling, and management of containerized applications.

Kubernetes includes the following features:

- Horizontal scaling
- Service discovery and load balancing
- Secrets and configuration management
- API-based automated rollouts and rollbacks
- Self-healing

The following diagram shows the architecture of an ACS cluster using Kubernetes:





For further information on Kubernetes please visit <https://kubernetes.io/>. The complete source code you can find at <https://github.com/kubernetes>.

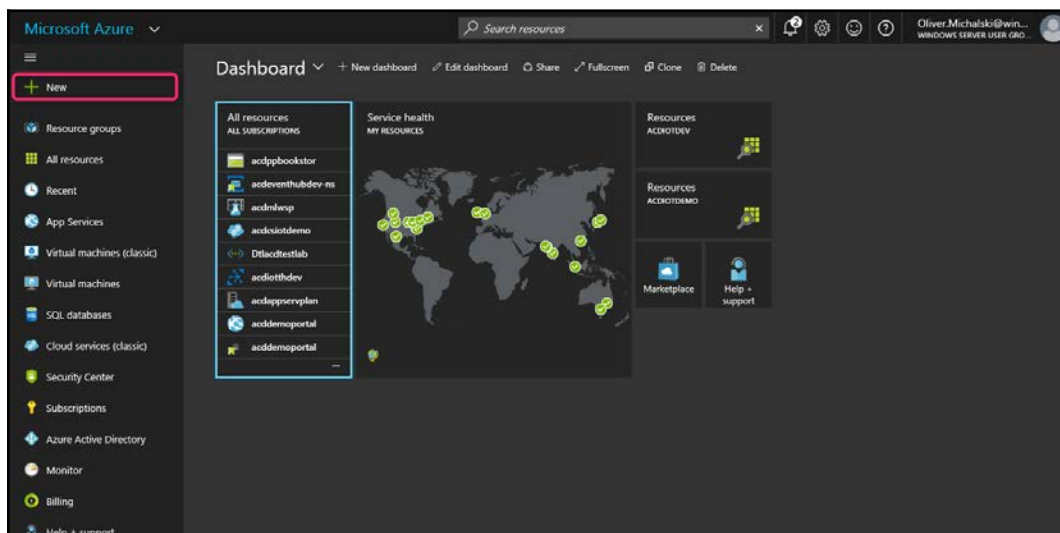
All three orchestrators are open source projects with frequent version changes. These version changes are considered by ACS but unfortunately, there is currently no possibility to upgrade an existing deployment.

Creating your First Azure Container Service

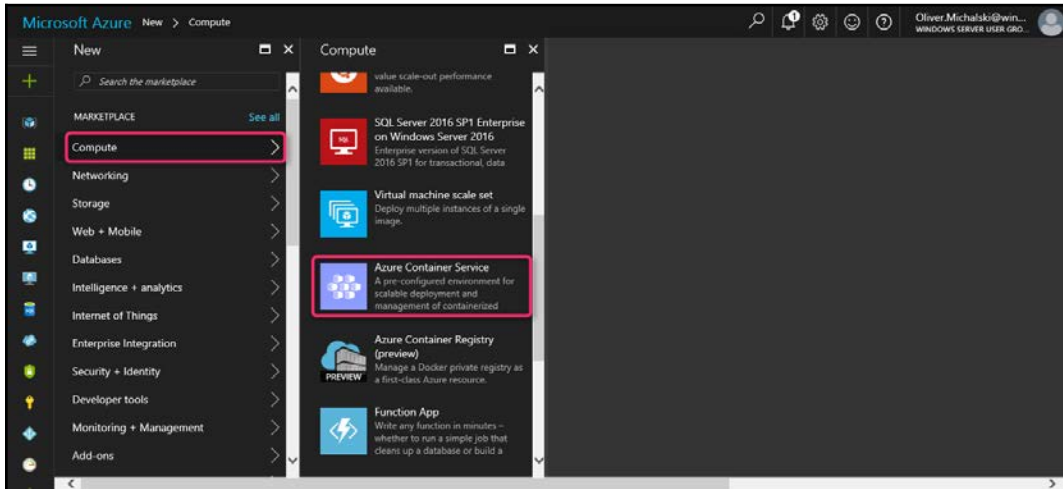
A preliminary remark: the following description is only an example. However, the described procedure applies in the same way to all supported configurations.

Let's start:

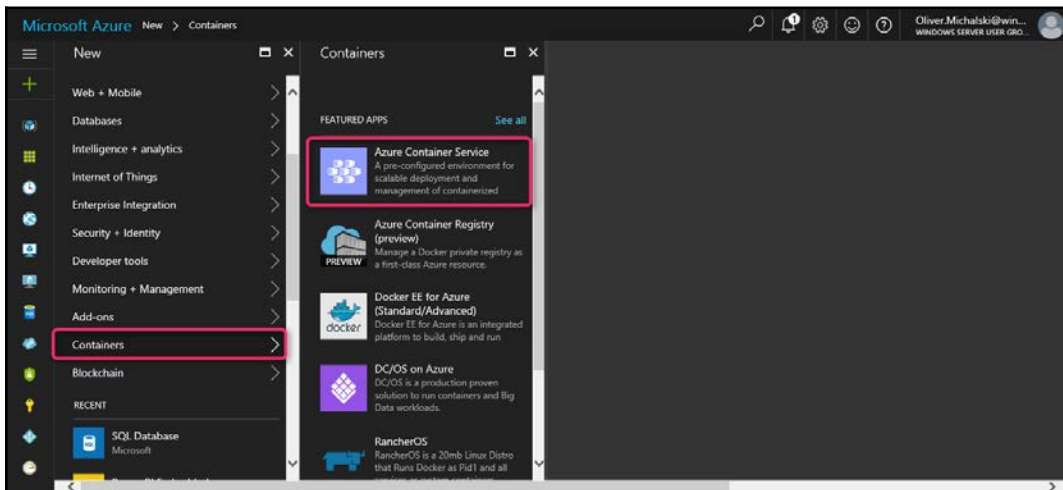
1. Open your Azure management portal at <https://portal.azure.com>.
2. In the portal, click **New**:



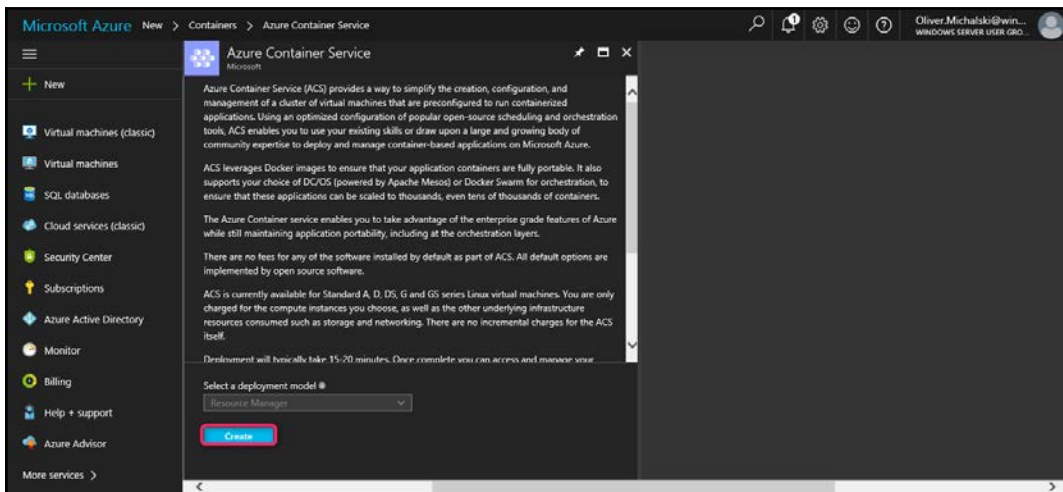
3. Now click **Compute**, scroll down the list, and then click **Azure Container Service**:



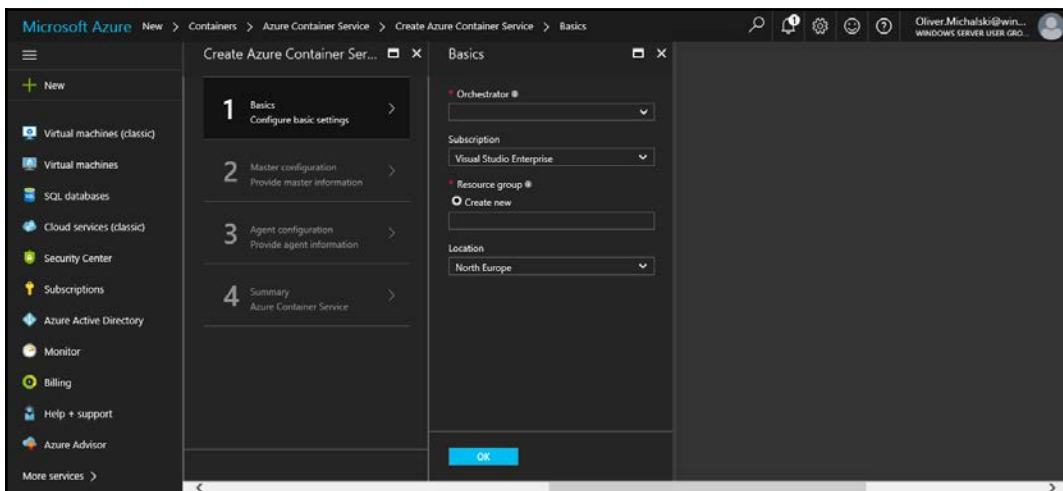
4. Alternatively, you can also click **Containers**, and then click **Azure Container Service**:



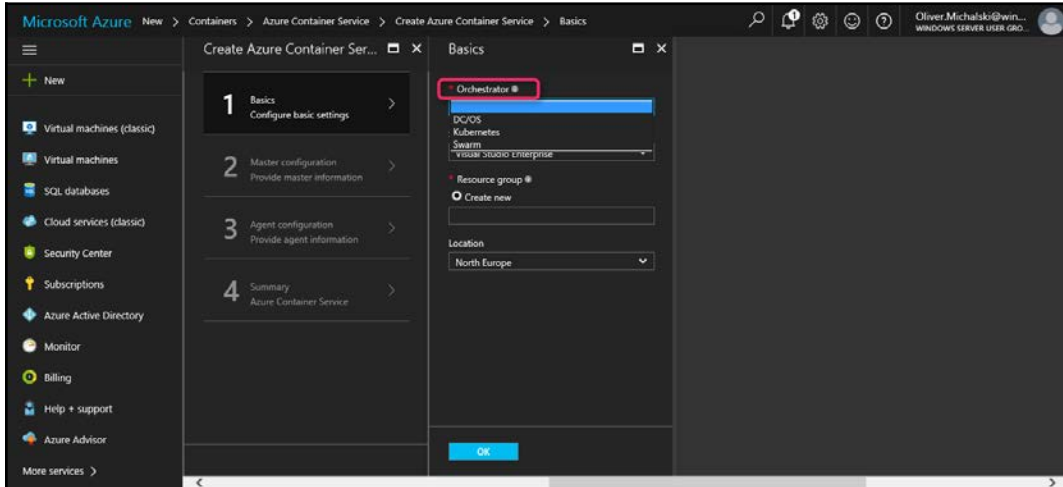
5. Now open the info page of the offer, press the **Create** button here:



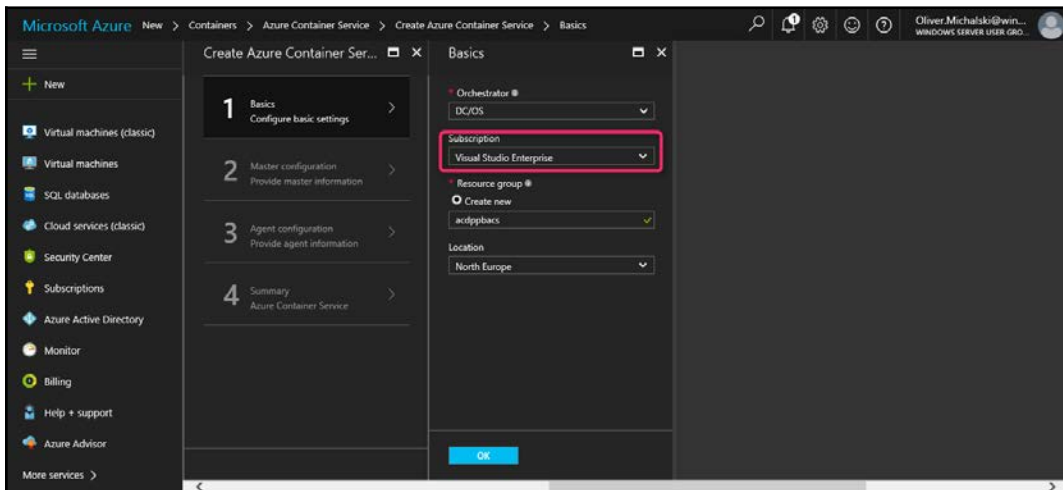
6. Now open the **Create Azure Container Service** dialog:



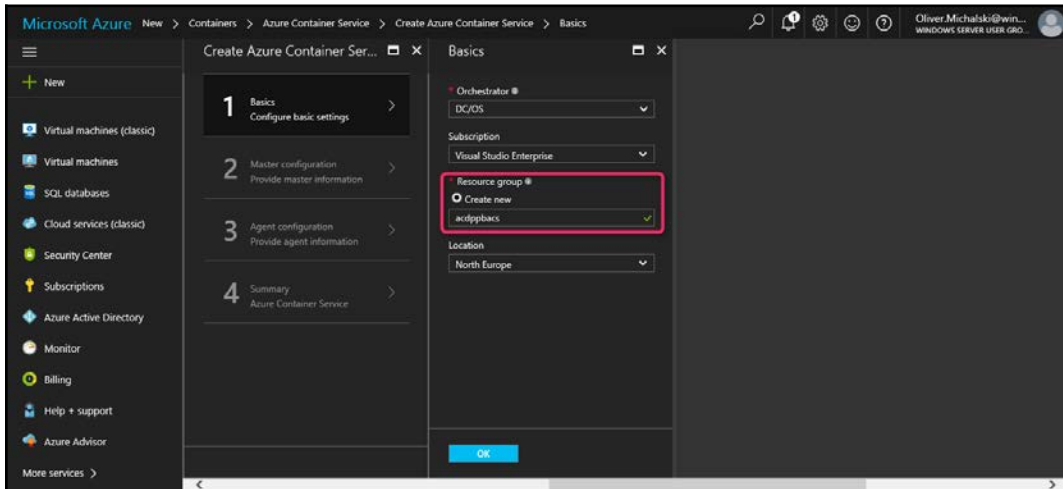
7. The **Create Azure Container Service** consists of four parts. In part 1, **Basics**, you must first select an **Orchestrator**. For the following demo, I chose DC/OS:



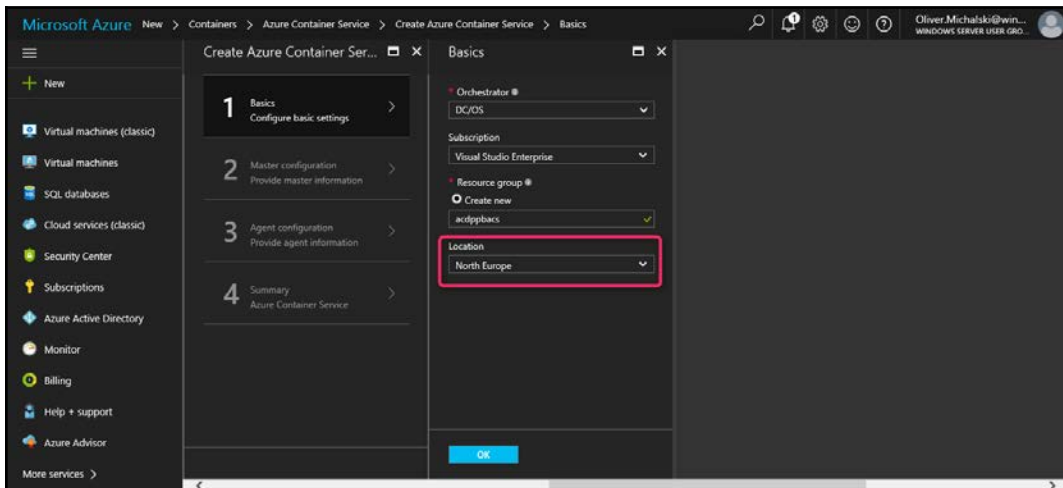
8. Next, choose a **Subscription** (Use the default subscription):



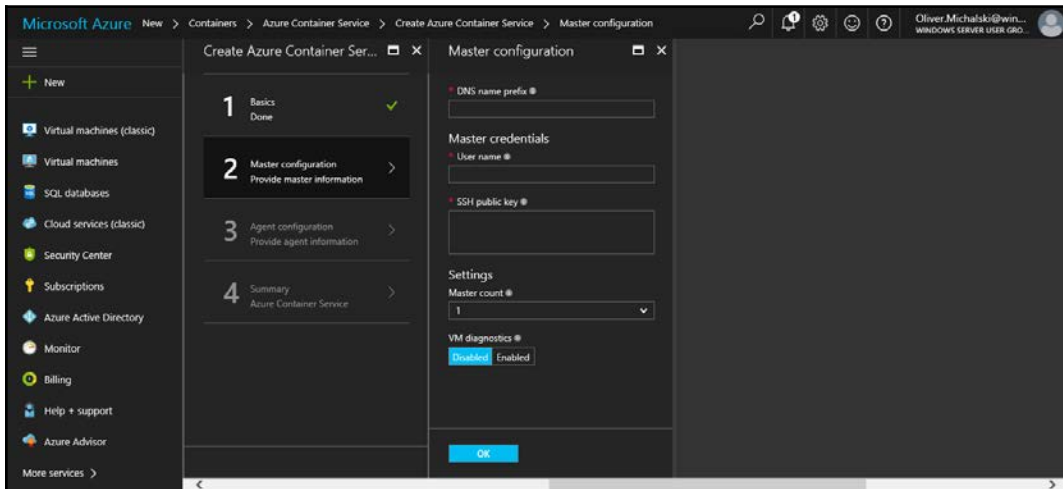
9. In the **Resource group** section, you only have the option to create a new **Resource group** (of course there is a reason, as we shall see later). Type a name of your choice for the **Resource group** in the text field:



10. In the **Location** list, select the same location you have been using for the Azure Resource groups:

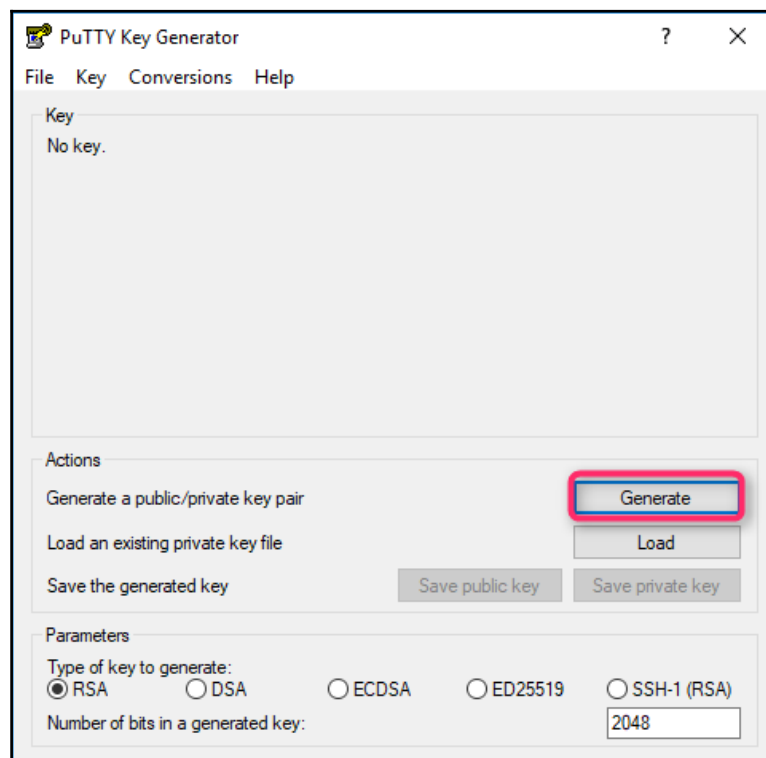


11. Once you have made your settings, press the **OK** button.
12. Now open part 2, **Master configuration** from the **Create Azure Container Service** dialog:

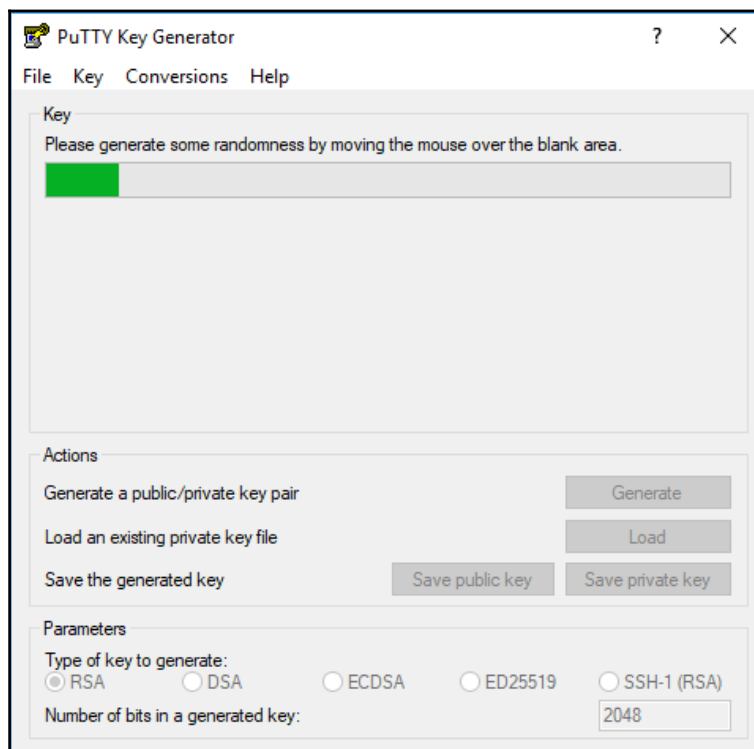


13. Type a unique name for the service you are creating in the **DNS name prefix** box. If the name of the service is unique, you can see the green tick.
14. In the **Master credentials** section type a **User name** for your admin, and then copy and paste an **SSH public key** in the following box. (If you do not already have an SSH key, you should work through the following points. Otherwise, you can go directly to *step 21*).
15. For the work with SSH, you need a key pair. The so-called **public key** is stored on the server with which you want to connect. The so-called **private key** then serves as authentication against the server.
16. In principle, there are several possibilities to generate a key pair. These vary depending on your OS and so the following is just one example.

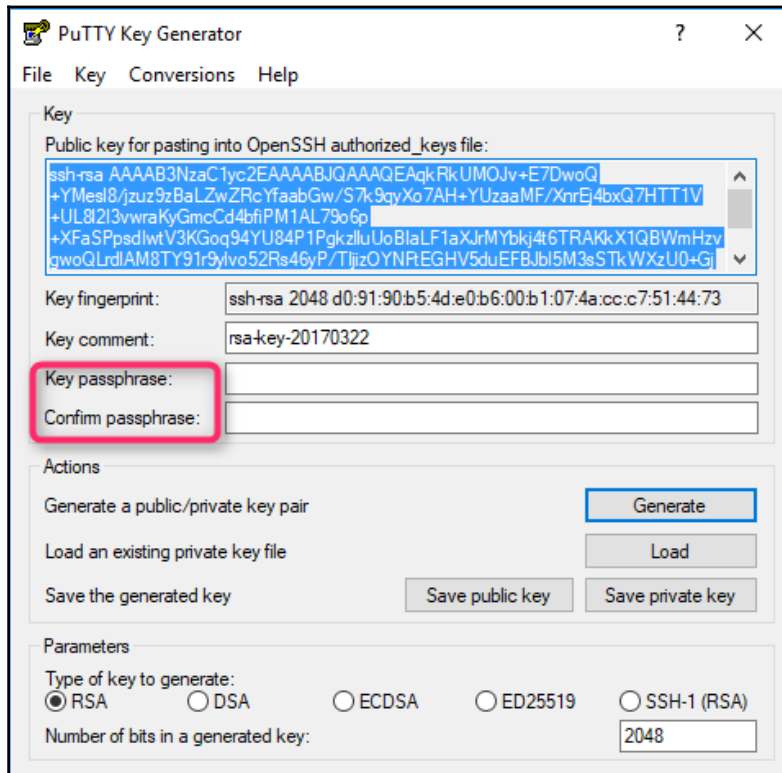
17. Starting out of the premise that you use a PC with a Windows OS, I recommend using PuTTY. You can find an Installer for PuTTY at <http://www.putty.org/>. Download and install PuTTY.
18. Now search for the PuTTYgen tool.
19. Open the tool and click the **Generate** button:



20. The generation process starts. You can speed up the process by hovering over the empty area:



21. Once the key pair has been generated, you must enter a passphrase (a type of password, only longer) for the private key and confirm by typing again:



The screenshot shows the PuTTY Key Generator window. The 'Key' section displays the public key for pasting into the OpenSSH authorized_keys file, the key fingerprint, and the key comment. The 'Key passphrase' and 'Confirm passphrase' fields are highlighted with a red box. The 'Actions' section includes buttons for 'Generate', 'Load', 'Save public key', and 'Save private key'. The 'Parameters' section shows the type of key to generate (RSA selected) and the number of bits in a generated key (2048).

PuTTY Key Generator

File Key Conversions Help

Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAqkRkUM0Jv+E7DwoQ  
+YMesl8/jzuz9zBaLZwZRcYfaabGw/S7k9gyXo7AH+YUzaaMF/XnrEj4bxQ7HTT1V  
+UL8l2l3vvwraKyGmcCd4bfiPM1AL79o6p  
+XFaSPpsdlwtV3KGoq94YU84P1PgkzlluUoBlalF1aXJrMYbkj4t6TRAKkX1QBWmHzv  
gwoQLrdlAM8TY91r9ylvo52Rs46yP/TijzOYNREGHV5duEFBjbl5M3sSTkWXzU0+Gj
```

Key fingerprint: ssh-rsa 2048 d0:91:90:b5:4d:e0:b6:00:b1:07:4a:cc:c7:51:44:73

Key comment: rsa-key-20170322

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair

Load an existing private key file

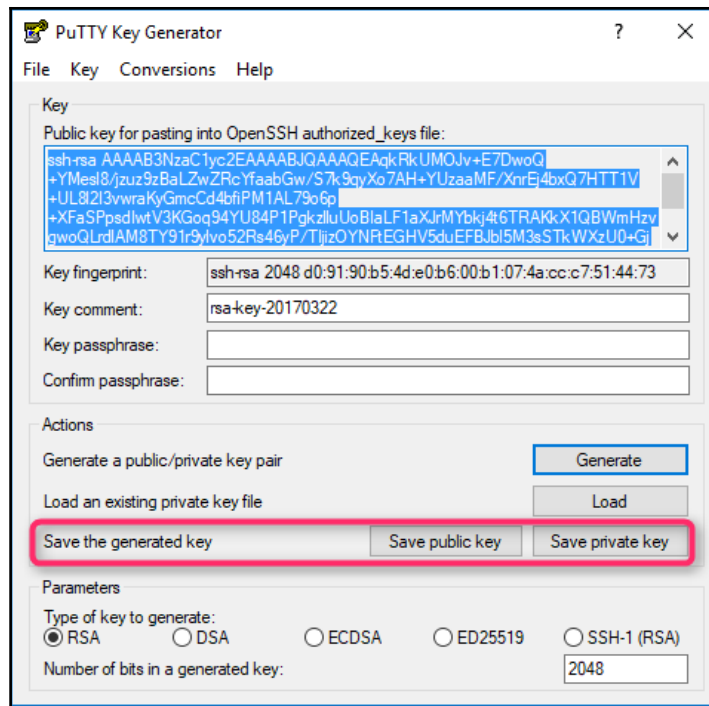
Save the generated key

Parameters

Type of key to generate:
☒ RSA ☐ DSA ☐ ECDSA ☐ ED25519 ☐ SSH-1 (RSA)

Number of bits in a generated key:

22. The last step, save your SSH keys in the local file system:

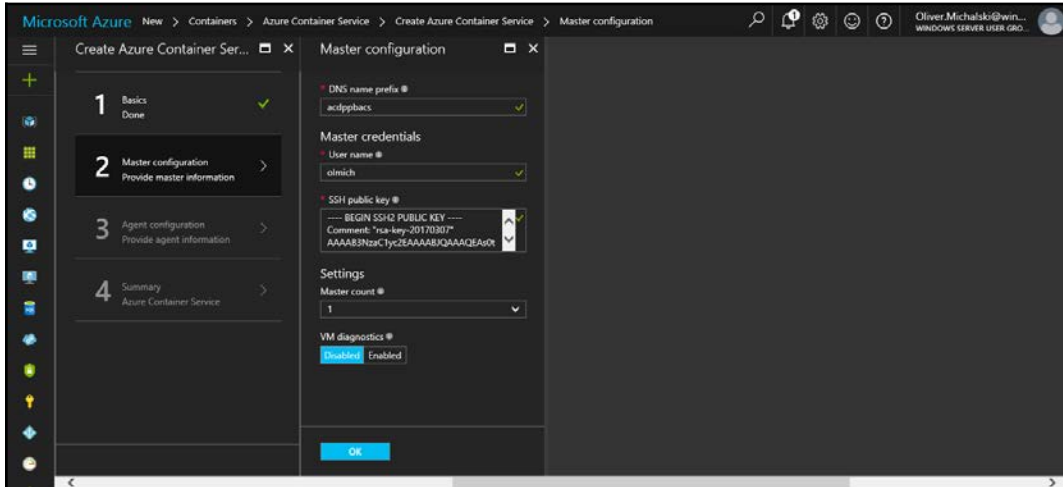


The key pair is not only good for working with ACS, but can also be used in other fields, for example, each Linux VM or also GitHub.

23. Now we are back in the **Master configuration** of the **Create Azure Container Service** dialog. In the dialog window, you will also find two optional settings:

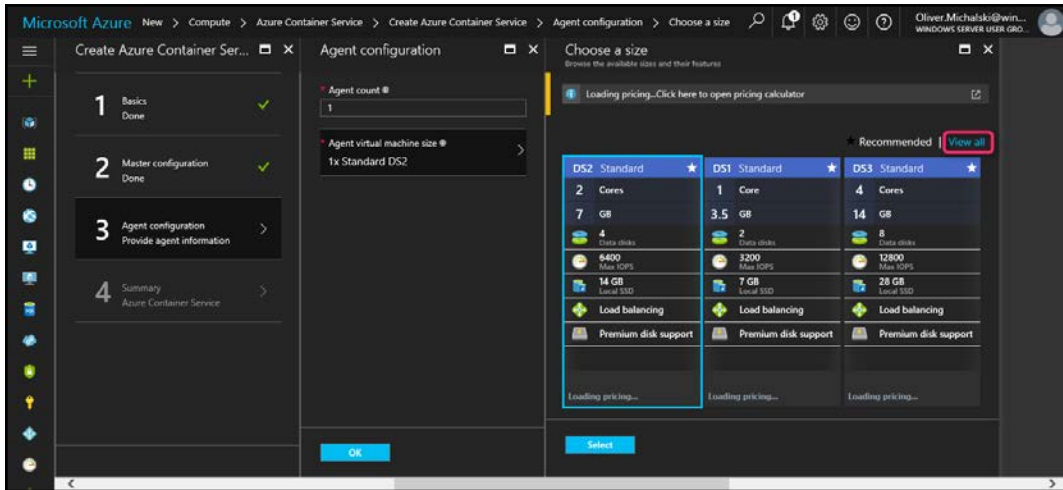
- **Master count:** Here you can decide how many master nodes are to be created

- **VM diagnostics:** here you can decide whether the diagnostics feature is enabled or not

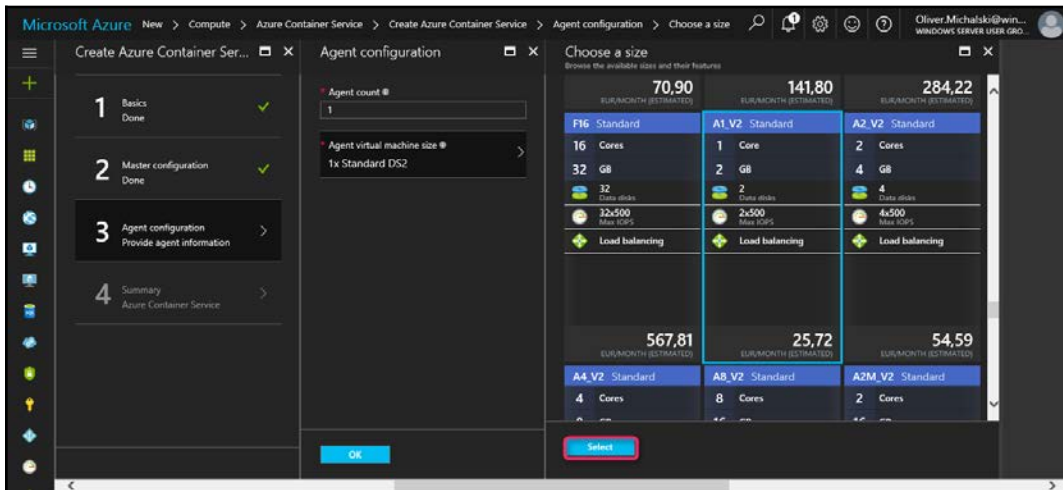


24. Once you have made your settings, press the **OK** button.
25. Now open part 3, **Agent configuration** of the **Create Azure Container Service** dialog.
26. In this dialog window, you will also find two settings:
 - **Agent count:** Here you can decide how many agent nodes are to be created
 - **Agent virtual machine size:** Here you can decide which Azure Series is used

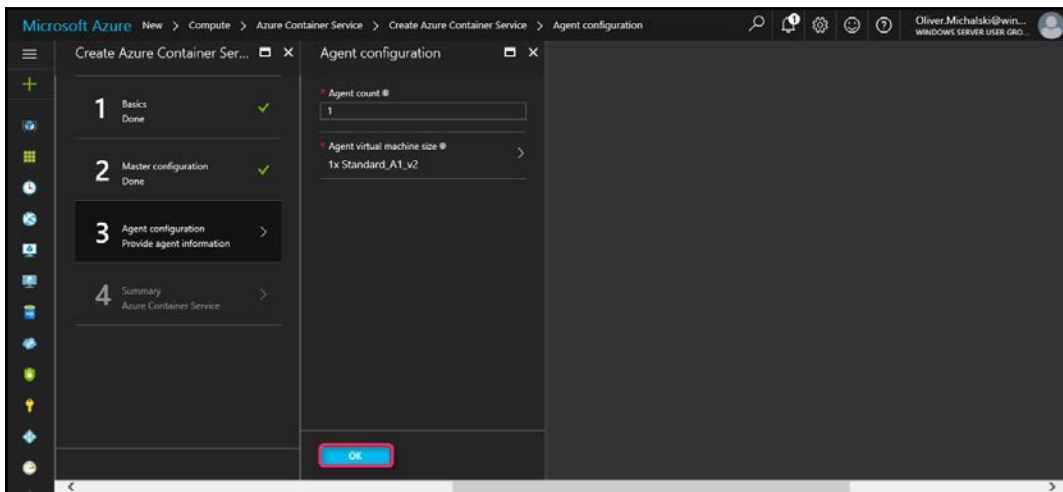
27. The setting for **Agent virtual machine size** needs a closer look. Once you click on the setting, you will first see the so-called recommendations. Attention: these are usually the most expensive solutions. Therefore, you should now click on the **View all** link:



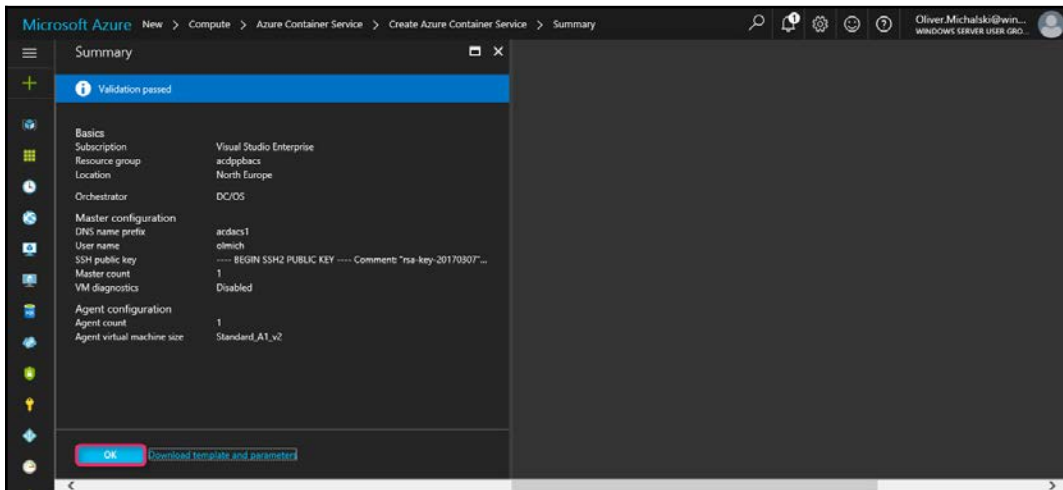
28. Now a much longer list opens. Here you can find a suitable offer. For our small demo, I decided to use the instance A1_V2. This is currently the cheapest deal. Press the corresponding tile once and then press the **Select** button:



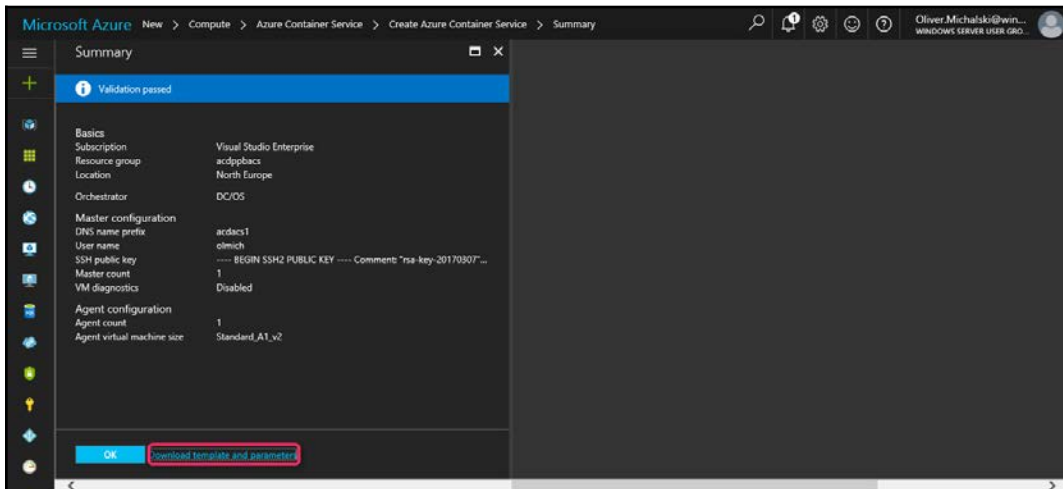
29. Once you have made your settings, press the **OK** button:



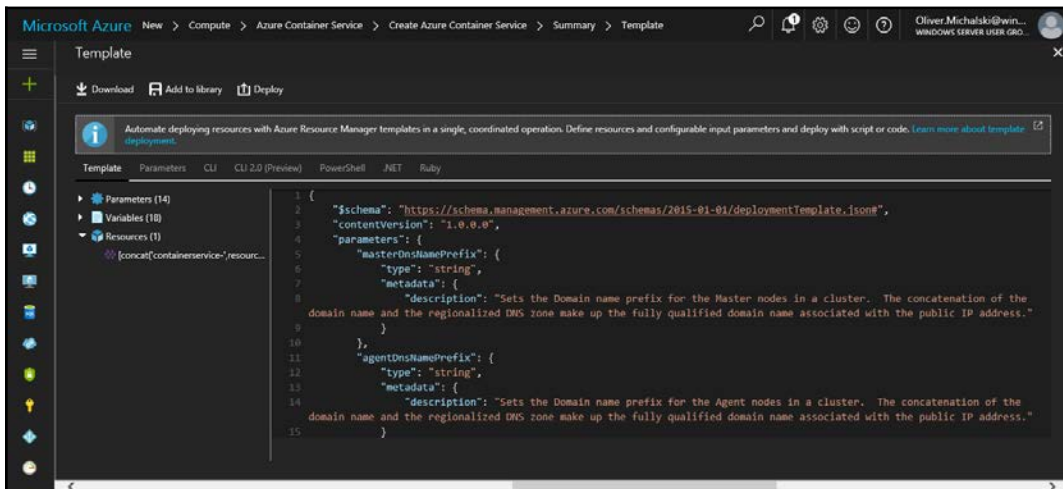
30. Now open part 4, **Summary** from the **Create Azure Container Service** dialog. Here you can now press the **OK** button and the deployment starts:



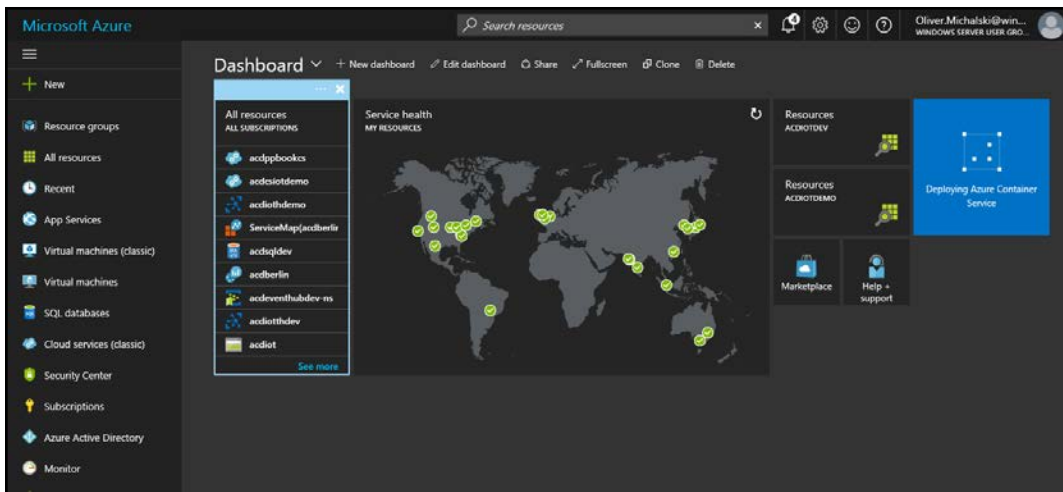
31. You can also download an ARM template for future use. Simply click on the **Download template and parameters** link:



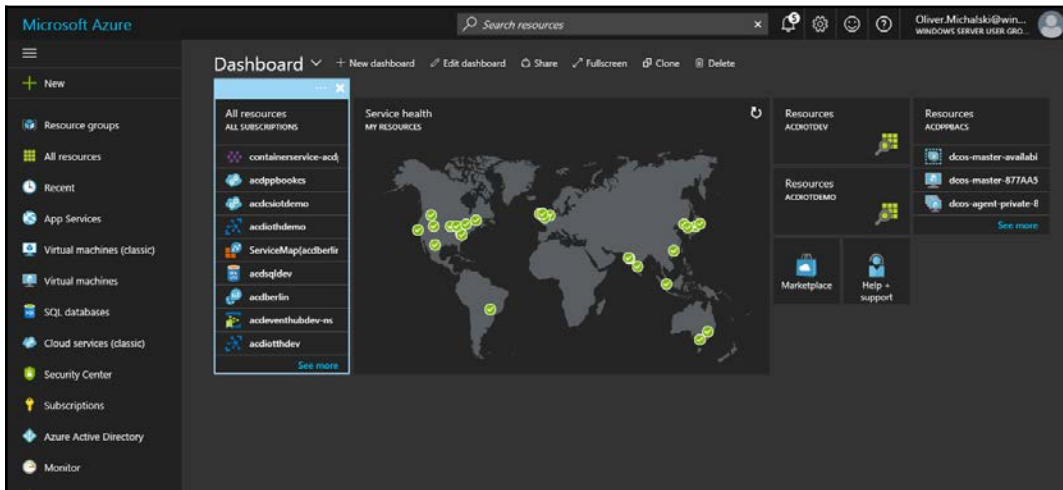
32. Now a **Template** editor window opens. This editor is similar in functionality to the template editor, which I described in *Chapter 2, Azure Resource Manager and Tools*, but does not allow any direct changes to the template:



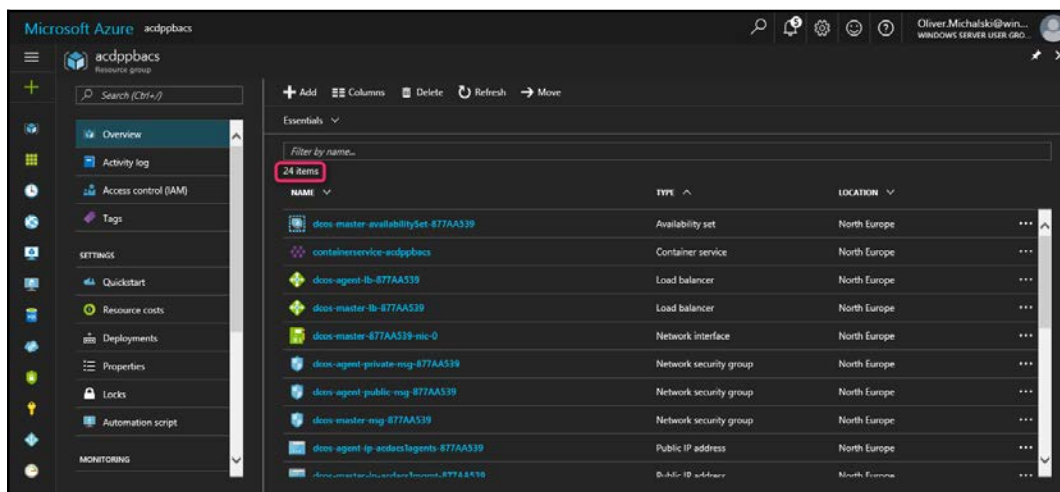
33. Once you have started your deployment, you can take a break:



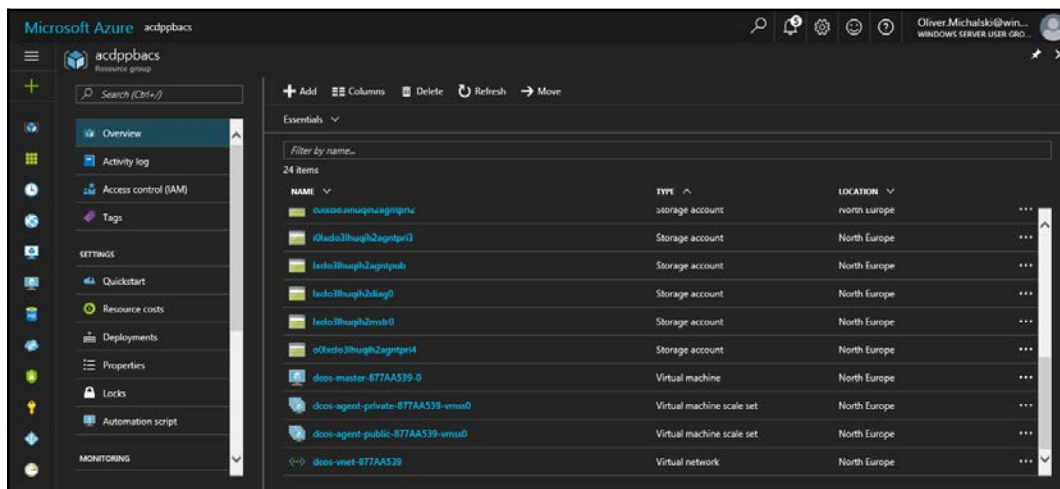
34. Twenty minutes later, you should see a result on the dashboard. Please click the **See more** link in the new tile:



- This opens the **Resource group** dashboard of your new ACS cluster. Now you can also see why you could only create a new **Resource group** before. Instead of just one new object, you will find a total of **24 items** (container service, load balancer, network interface, network security group, virtual machine, virtual machine scale set, virtual network, and storage accounts):



36. To see all elements of the Resource group, you must scroll down the list





I have important information for you, even if no costs are incurred for the Azure container service, significant costs are incurred for the remaining resources. In my first attempts with ACS, I very quickly consumed the Azure credits of my MSDN subscription. So, keep an eye on cost developments.

How to Work with Your Azure Container Service Cluster

Now we have created a new ACS cluster. But we also want to work with it. I will now introduce the basic approach.

As an important prerequisite, to work with the cluster, you need the Master FQDN (a URL). The Master FQDN can be found in the **Essentials** section of the dashboard of your container service.

NAME	VM SIZE	VM COUNT	FQDN
agentpool	Standard_A1_v2	1	acdacs1agents.northeurope.cloudapp.azure.com

Now that we know the important prerequisites, we can go further.

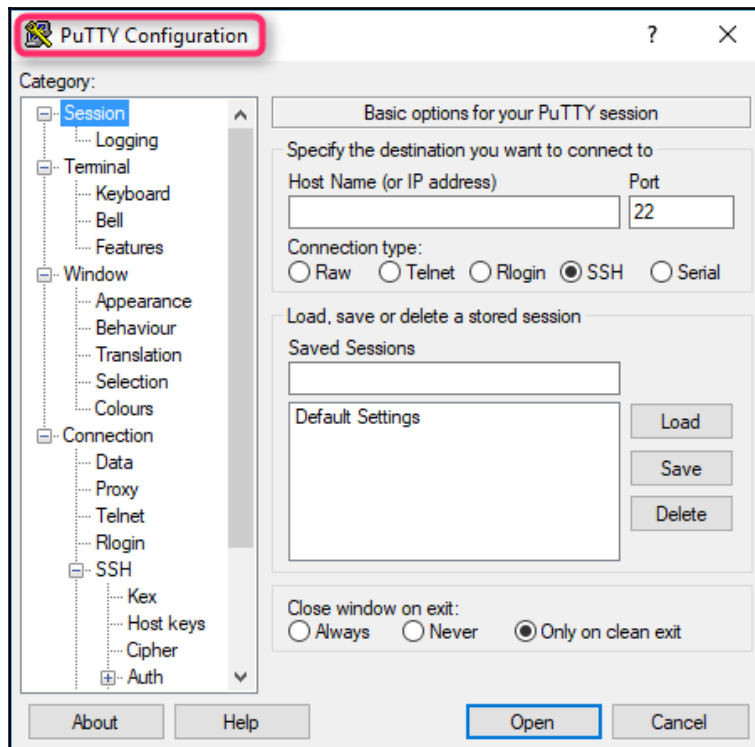
Each of the three available orchestrators provides you with a work surface. To work with these UIs, you must first create an SSH tunnel.

You don't know how to create an SSH tunnel?

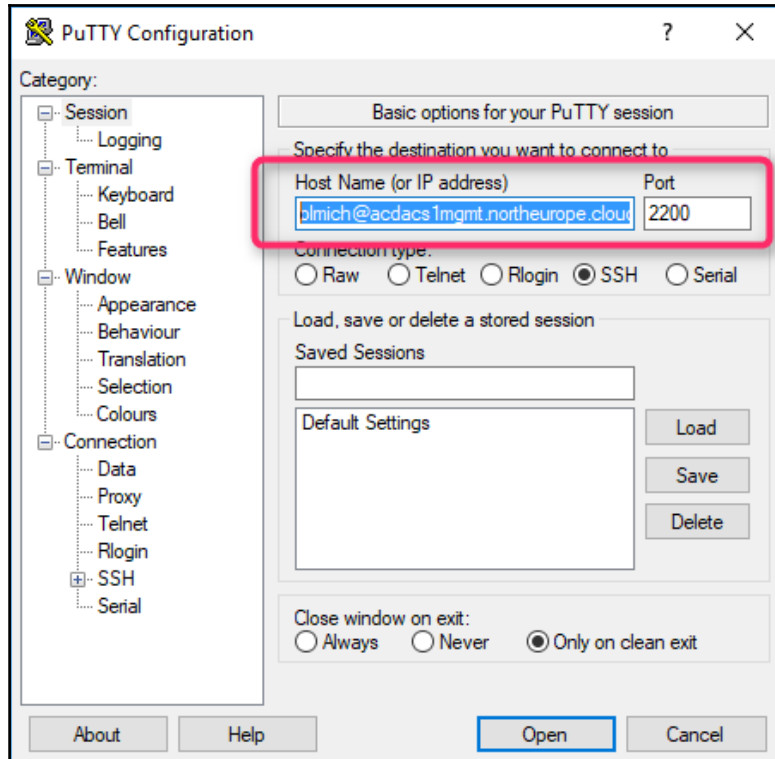
Then I want to bring you closer to the procedure by the way of an example. I assume that you have followed my earlier advice and have installed the PuTTY toolset. I also assume that your SSH key pair is available.

Everything okay? Then let's start:

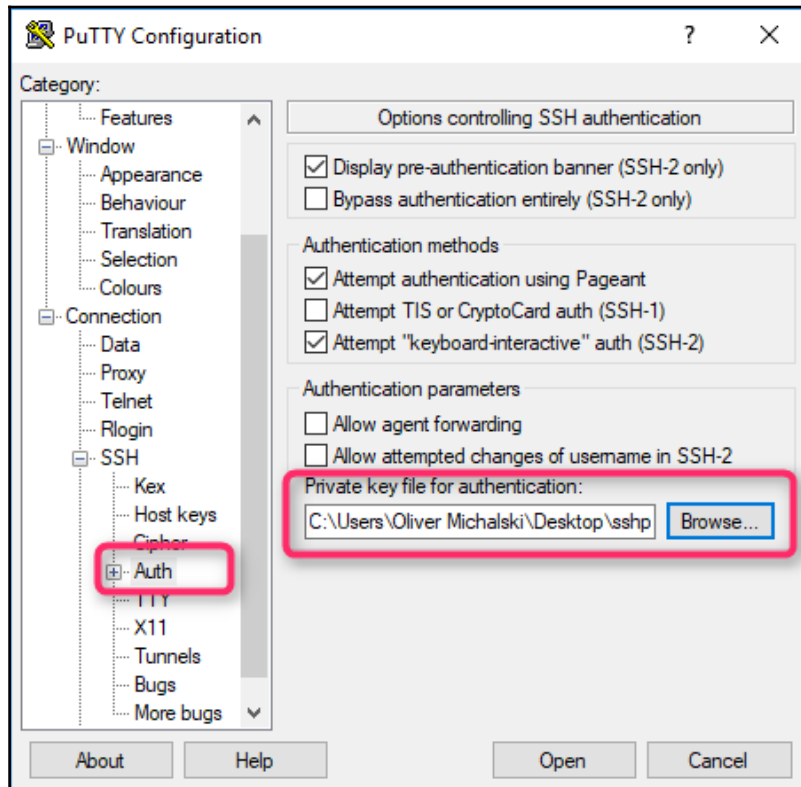
1. Search for the **PuTTY** tool and then open the tool:



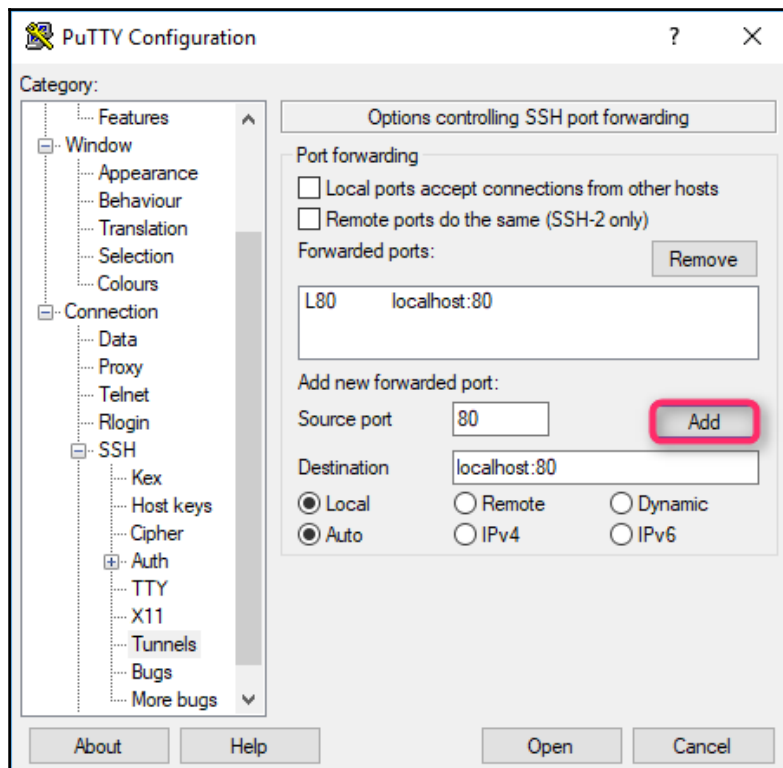
2. On the first page, fill in the **Host Name** field. The hostname is composed of the admin username (from your cluster) and the Master FQDN and has the format `adminusername@masterfqdn`. Change the **Port** to 2200:



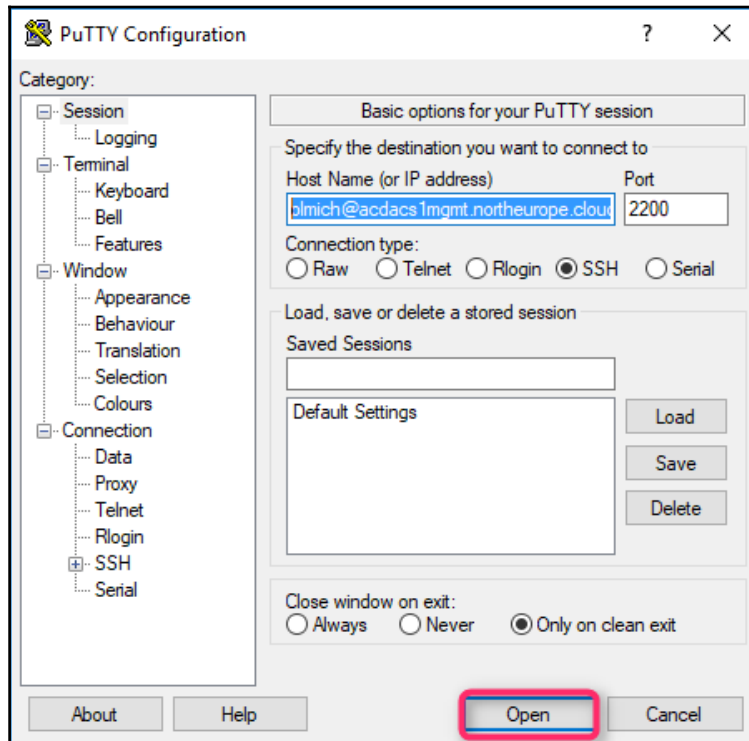
3. Now switch to the **Connection | SSH | Auth** site. Here you enter the path to your SSH private key:



4. Move to the **Connection | SSH | Tunnels** site. In the **Add** new forwarded port section, type 80 in the **Source port** field and `localhost:80` in the **Destination** field. Finally press the **Add** button:



5. Now go back to the first page. Press the **Open** button and the SSH tunnel is built up:



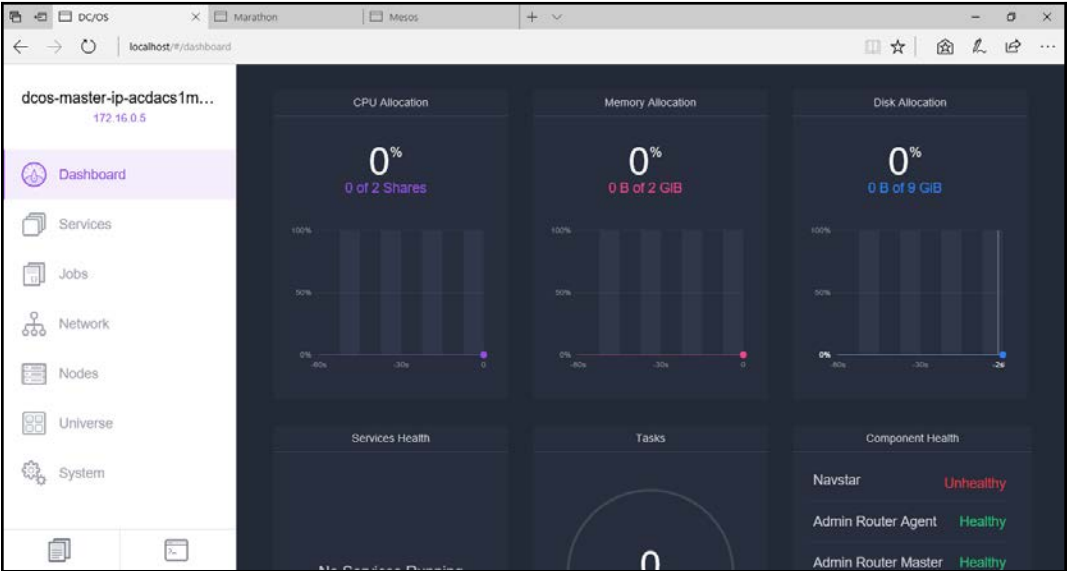
Have you created your SSH tunnel now?

If yes, we will look now at the work surfaces once. Remember, since we have created our cluster based on DC/OS, it is the UIs of DC/OS. The UIs of the other types are very similar.

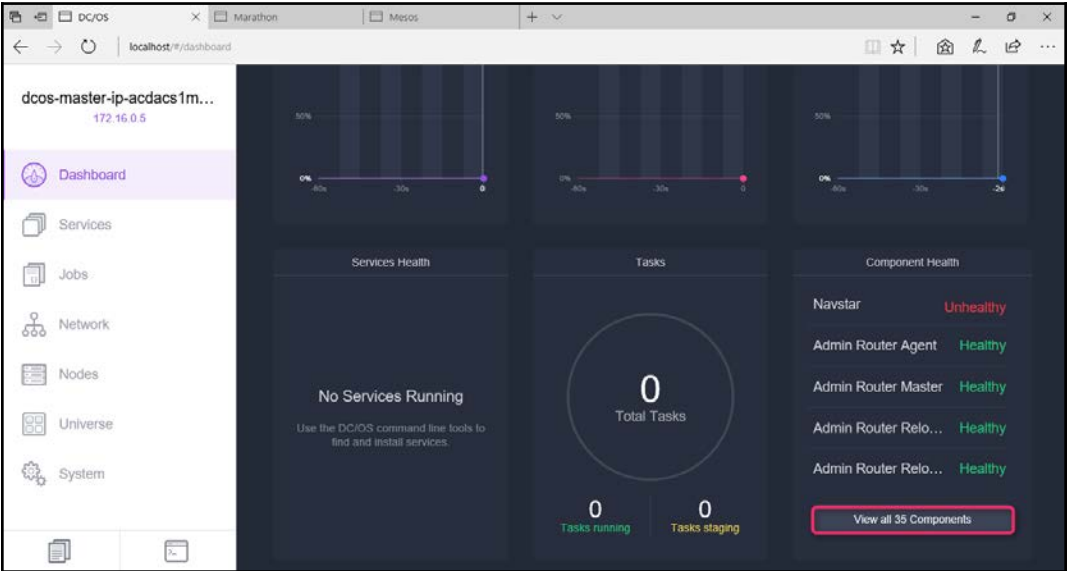
Let's start with the DC/OS dashboard. To reach this UI, enter the following URL into the browser of your choice:

```
http://localhost:80/
```

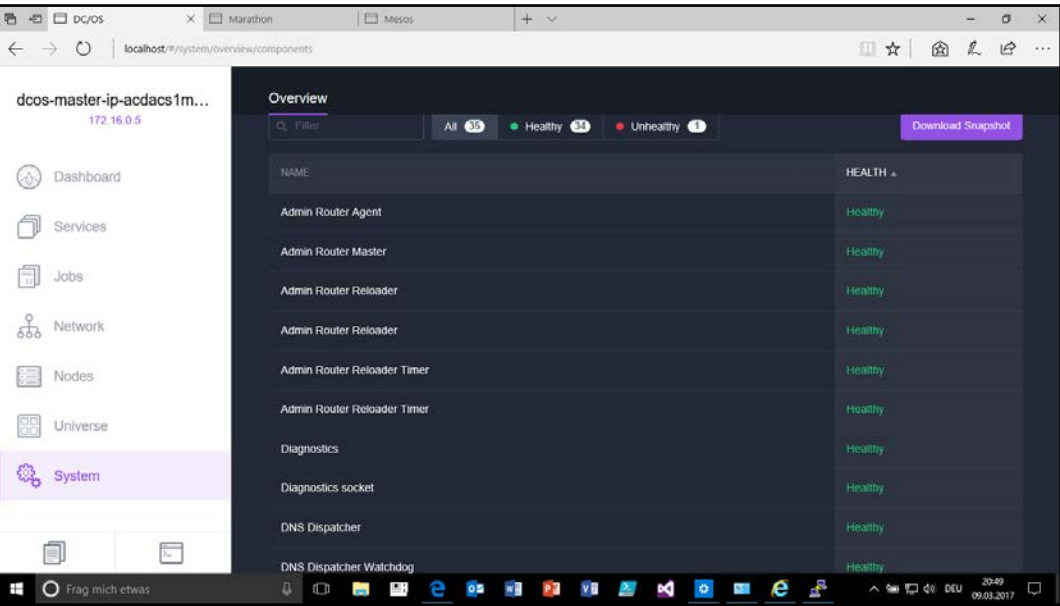
With the **DC/OS** dashboard, you can monitor the performance indicators of your cluster or display the health status of individual components:



If you want to see the health status of individual components, click the **View all 35 Components** button in the **Component Health** tile:

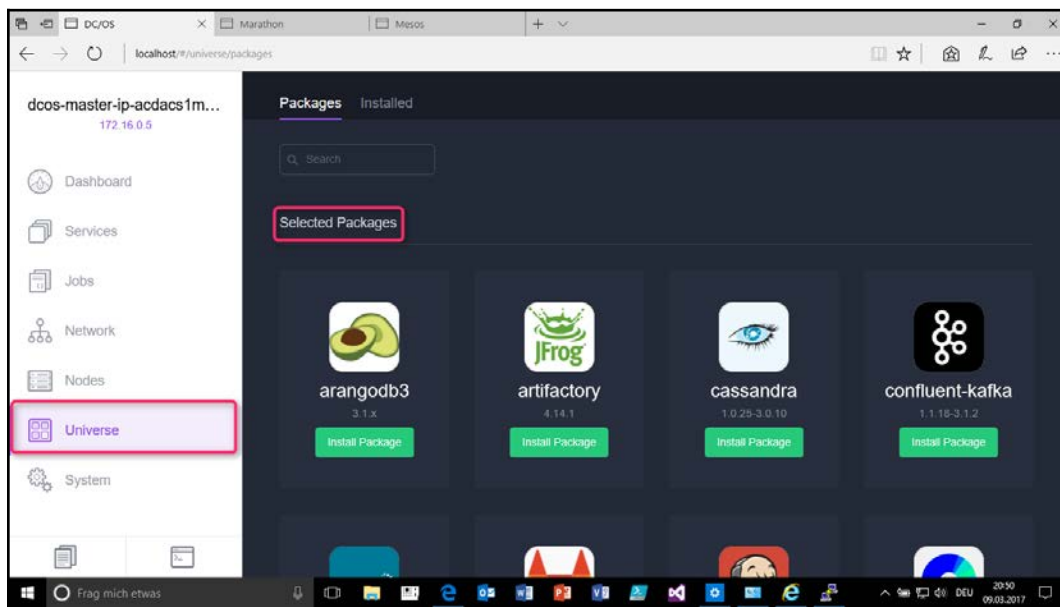


A detailed list with the corresponding status information will open:

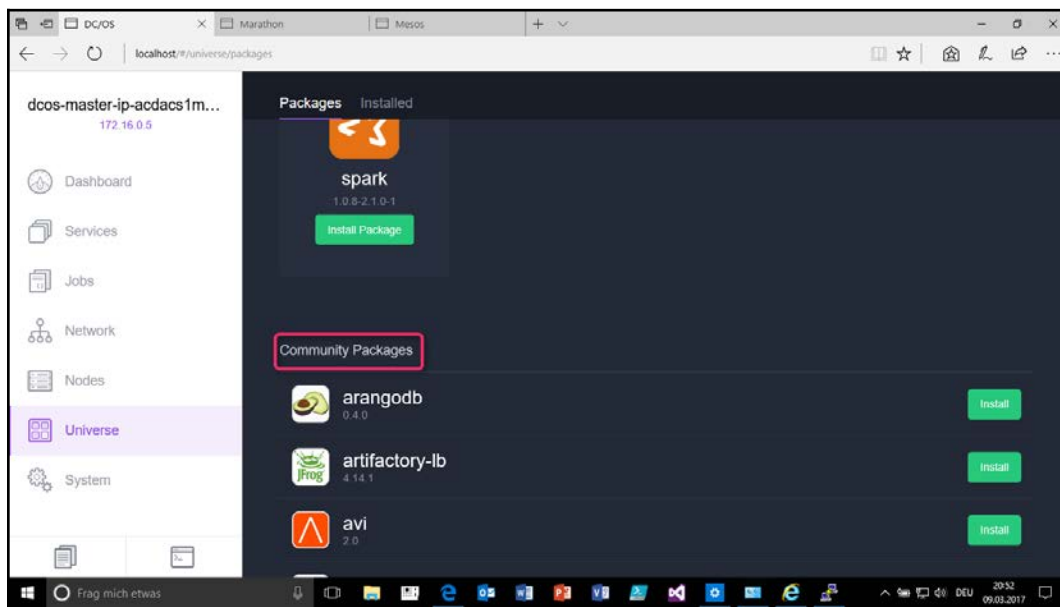


In the **DC/OS** dashboard you will also find another interesting application. Simply press the **Universe** button in the navigation area. This starts mesosphere universe.

Mesosphere universe is a package repository that contains services like Spark, Cassandra, Jenkins, and many others that can be easily deployed onto your cluster:



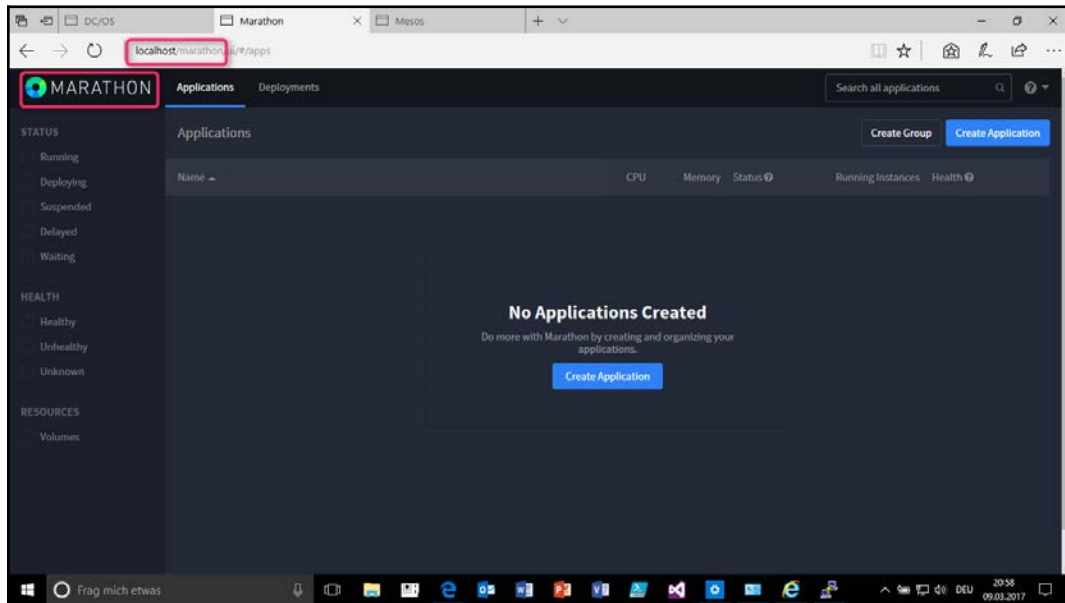
In addition to the solutions provided by mesosphere, there are still solutions from the community. Just scroll the page down:



The next area is the **MARATHON** orchestration platform. To reach this UI, enter the following URL into the browser of your choice:

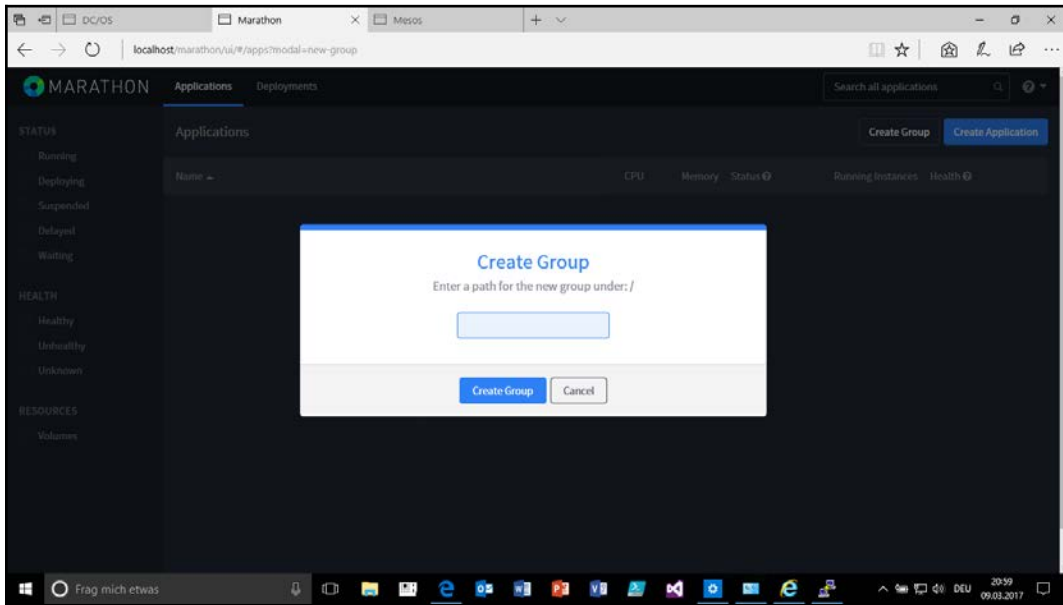
```
http: //localhost:80/marathon/
```

With this UI, you can start a new container and other types of application in the cluster. In addition, the UI also provides information on executed containers and applications and is constantly on-going for planning tasks:

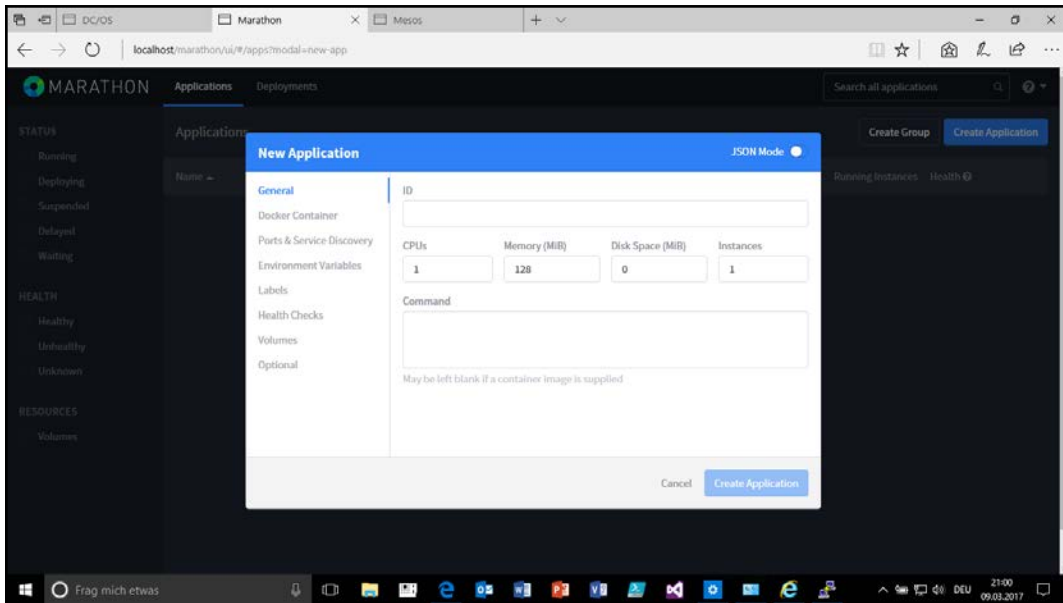


Two short examples:

- The following screenshot shows the dialog for creating a group. A group in DC/OS is a collection of apps (services, and so on) that are related to each other (for example, over the organization):



- The next screenshot shows the dialog for creating a **New Application**. An application is a long-running service that may have one or more instances that map one to one with a task:



Internally, the user creates an application by providing an application definition (a JSON file). Marathon then schedules one or more application instances as tasks depending on how much the definition is specified.

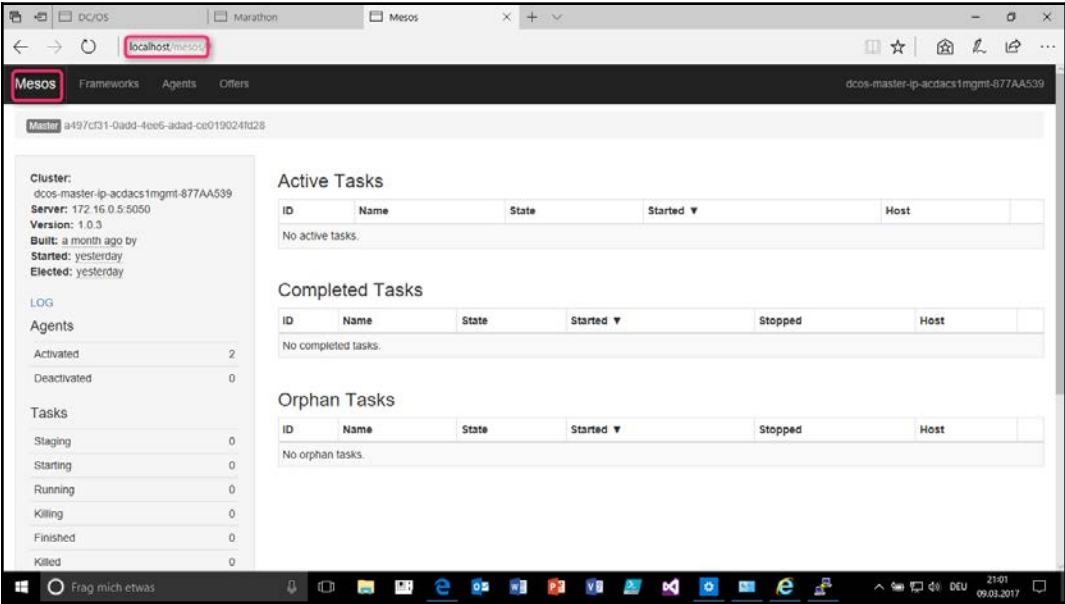


In the original concept of DC/OS, there is still another part, the pods. A pod is a special case of an application. A pod is also a long-running service that may have one or more instances but map one to many with colocated tasks. Pod instances may include one or more tasks that share resources (for example, IPs, ports, or volumes). Pods are currently not supported by ACS.

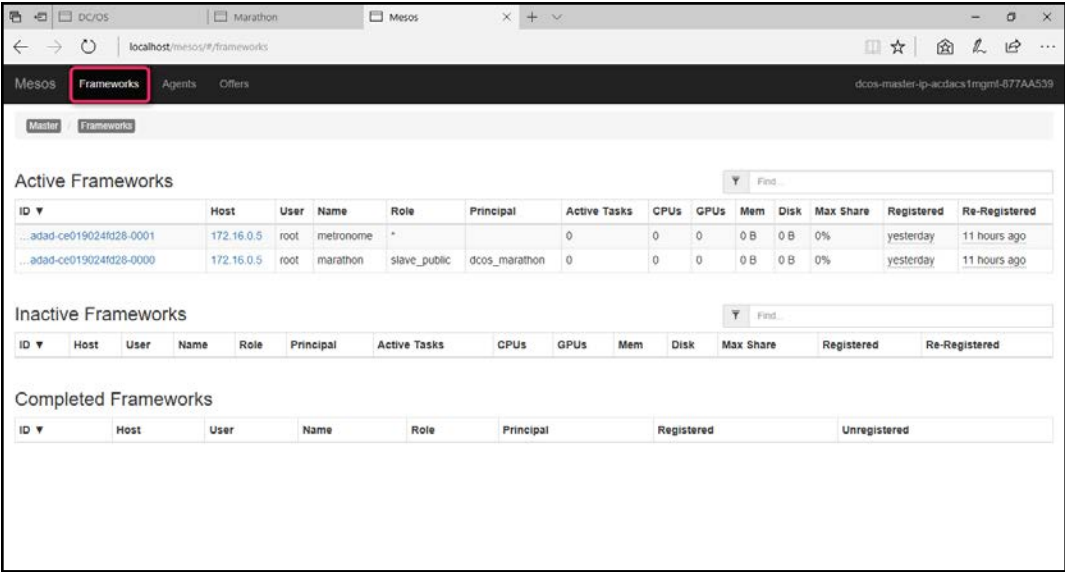
The last area is the Mesos. Mesos is a web UI for viewing cluster state, and above all tasks. To reach this UI, enter the following URL into the browser of your choice:

```
http: //localhost:80/mesos/
```

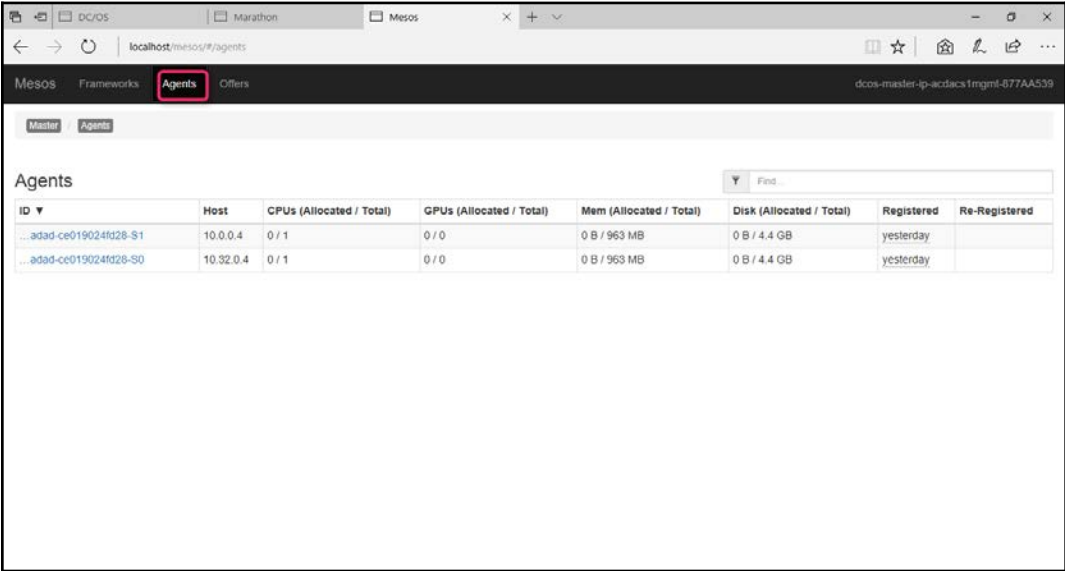
The following screenshot shows the UI for **Tasks**:



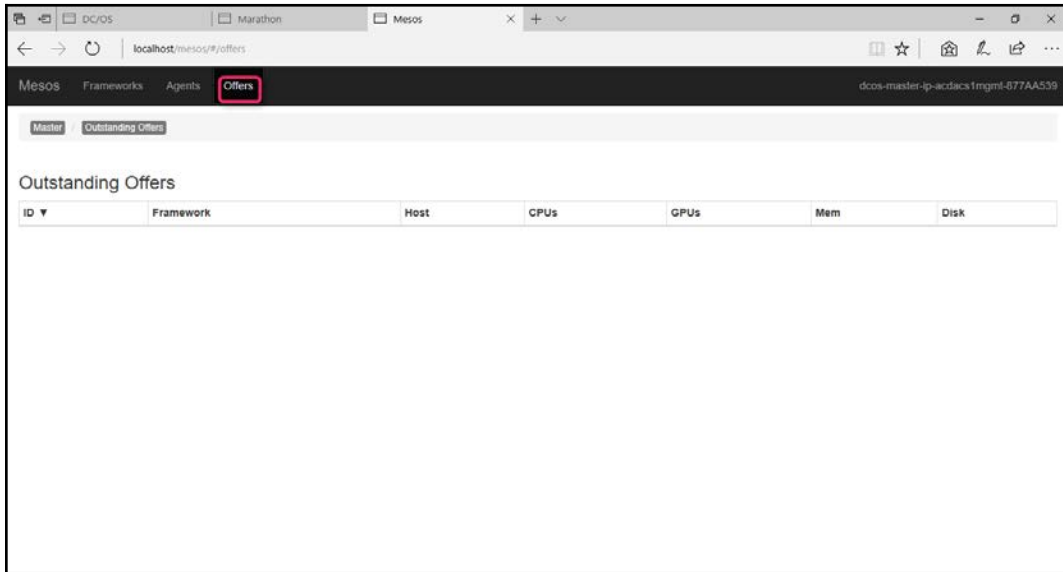
The next screenshot shows the UI for **Frameworks**. A framework running in Mesos consists of two components: a scheduler that registers with the master to be offered resources, and an executor process that is launched on agent nodes to run the tasks:



The next screenshot shows the UI for **Agents** and its conditions:



The last screenshot shows the UI for **Offers**. An offer in Mesos is simple a resource (for example, container), and is assigned to a framework for processing:



Summary

In this chapter, we learned all about the Azure Container Services and therefore the basics for implementing a modern Azure IaaS solution.

In the first part of this chapter, we learned the basic knowledge of the Azure container service and the basic terminology.

Then in the second part of this chapter, there followed a description of the create process and a short introduction in working with ACS.

In the next chapter, we will continue with the basics of Azure security (for example Azure AD).

9 Implementing Azure Security

In previous chapters, we learned how to set up and manage Azure infrastructure services. These included Resource groups, Azure Active Directory, virtual networks, and storage accounts.

Security is a very major and important section and all these services should be created with security in mind. As cloud security is a little different from the classic security we know from our on-premise infrastructures, this chapter will provide a brief overview of security.

You should consider scalability, durability, and high availability depending on the scenario and target you try to achieve.

Within this chapter, you will learn about most of the basic Azure security offerings. You will learn how and when to implement and integrate security on storage and compute level. We will also take a look at (hybrid) identity security.

First we will take a look at Identity Security in Azure, as it provides the basis for most of the other techniques that we will talk about.

After that, Azure Resource (group) security will be discussed. Finally after talking about some Azure networking and storage security, we will delve into the logging and monitoring capabilities of Azure.

This chapter aims to provide an overview of the possibilities of Azure IaaS security. There are some Azure security topics, which are just briefly discussed and others that are not even mentioned. As every security technology in this chapter will probably fill a small book on its own, this chapter is about basic implementations.

We will not talk about any **Operation Management Suite (OMS)** or **Enterprise Mobility Suite (EMS)** related security techniques in this chapter.

Azure Key Vault should be used whenever possible for key management. It can be leveraged for storage encryption, Azure Active Directory applications and many more. Key Vault is not part of this chapter, because it's a very comprehensive feature that would need a chapter alone.

In this chapter, we will explore the most built-in infrastructure and Identity security functions. There will also be a section about Logging.

Azure Identity Security

Security boundaries are often thought of like firewalls, IPS/IDS systems, or routers. Also, logical setups such as DMZs or other network constructs are often referred to as boundaries.

But in the modern world, where many companies support dynamic work models that allow you to **bring your own device (BYOD)** or are heavily using online services for their work, the real boundary is the identity.

Identities are often the target of hackers as they resell them or use them to steal information. To make the attacks as hard as possible, it's important to have a well-conceived and centralized identity management. The Azure Active Directory provides that and a lot more to support your security strategy and simplify complex matters such as monitoring of privileged accounts or authentication attacks.

Azure Active Directory

Azure Active Directory (AAD) is a very important service that many other services are based on. It's not a directory service like many may think of when they hear the name Active Directory. The AAD is a complex structure without built-in **Organizational Units (OUs)** or **Group Policy Objects (GPOs)**, but with a very high extensibility, open web standards for authorization and authentication, and a modern, (hybrid) cloud-focused approach to identity management.

Azure Active Directory Editions

Chapter 3, Deploying and Synchronizing Azure Active Directory already described the synchronization between on-premise and Azure AD very well. As a result, we will not go deeper into AD Connect.

Just to recapture *Chapter 3, Deploying and Synchronizing Azure Active Directory*, here are the major differences between the AAD pricing tiers. This is important to remember, as there are two (in our opinion) very important security features that are only available in premium P2. We will focus on security-related features that were not handled in *Chapter 3, Deploying and Synchronizing Azure Active Directory*.

The following table describes the differences between the four Azure Active Directory editions:

Services	Common	Basic	P1	P2
Directory Objects	X	X	X	X
User/Group Management (add/update/delete)/ User-based provisioning, Device registration	X	X	X	X
Single Sign-On (SSO)	X	X	X	X
Self-Service Password Change for cloud users	X	X	X	X
Connect (Sync engine that extends on-premises directories to Azure Active Directory)	X	X	X	X
Security/Usage Reports	X	X	X	X
Group-based access management/ provisioning		X	X	X
Self-Service Password Reset for cloud users		X	X	X
Company Branding (Logon Pages/Access Panel customization)		X	X	X
Application Proxy		X	X	X
SLA 99.9%		X	X	X
Self-Service Group and app Management/Self-Service application additions/Dynamic Groups			X	X
Self-Service Password Reset/Change/Unlock with on-premises write-back			X	X
Multi-factor authentication (Cloud and On-premises (MFA Server))			X	X
MIM CAL + MIM Server			X	X
Cloud App Discovery			X	X
Connect Health			X	X
Automatic password rollover for group accounts			X	X
Identity Protection				X
Privileged Identity Management				X

Overview of Azure Active Directory editions

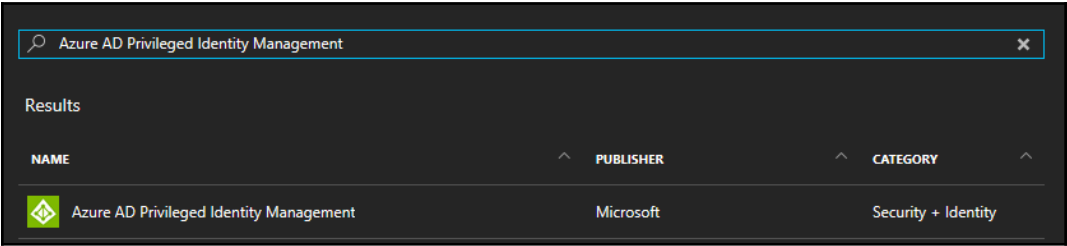
Privileged Identity Management

With the help of Azure AD **Privileged Identity Management (PIM)**, the user can access various capabilities. These include the ability to view which users are Azure AD administrators and the possibility to enable administrative services on demand such as Office 365 or Intune. Furthermore, the user is able to receive reports about changes in administrator assignments or administrator access history.

The AAD PIM allows the user to monitor the access in the organization. Additionally, it is possible to manage and control the access. Resources in Azure AD and services such as Office 365 or Microsoft Intune can be accessed.

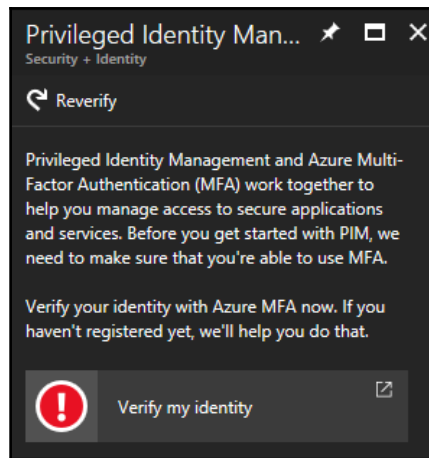
Lastly, the user can get alerts, if accesses to privileged roles are granted. Let's take a look at the PIM dashboard. To use Azure PIM, it needs to be activated first:

- 1. Azure AD PIM should be found in the search easily:



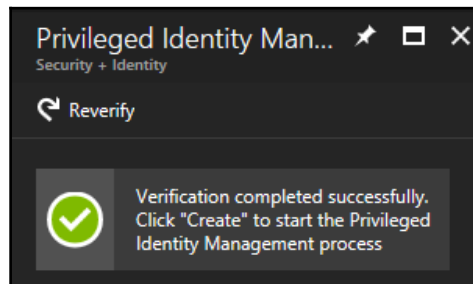
Azure AD PIM in the marketplace

- 2. After clicking on Azure AD PIM in the marketplace, PIM will probably ask you to re-verify MFA for security reasons. To do this, the MFA token needs to be typed in, after clicking on **Verify my identity**:



Re-verify identity for Azure AD PIM setup

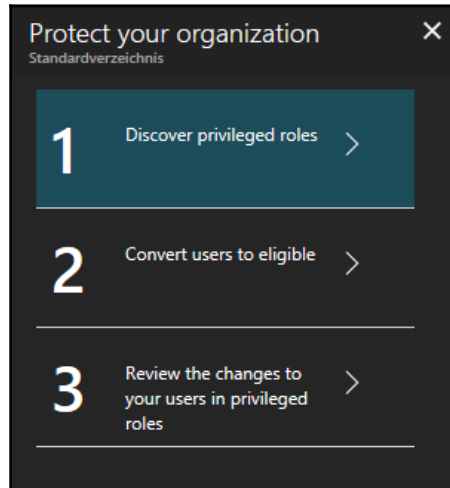
3. After a successful verification you will get an output as illustrated here:



Successful re-verification

Now the initial setup will start. The setup guides the user through the task of choosing all accounts in the tenant that have privileged rights. It is also possible to select them if they are eligible for requesting privileged role rights.

4. If the wizard is completed without choosing any roles or user as eligible, it will by default assign the security administrator and privileged role administrator roles to the first user that does the PIM setup. Only with these roles, it is possible to manage other privileged accounts and make them eligible or grant them rights:



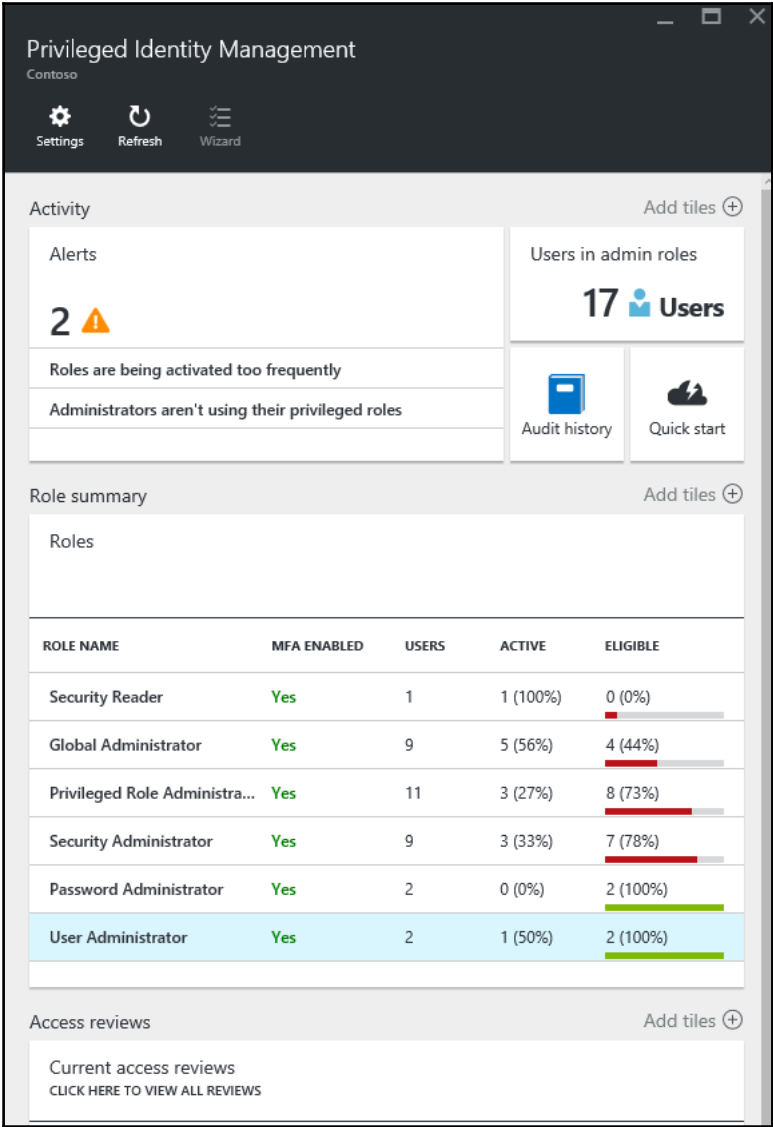
Setup wizard for Azure AD PIM

5. In the following screenshot the tasks related to **Privileged Identity Management** are illustrated:



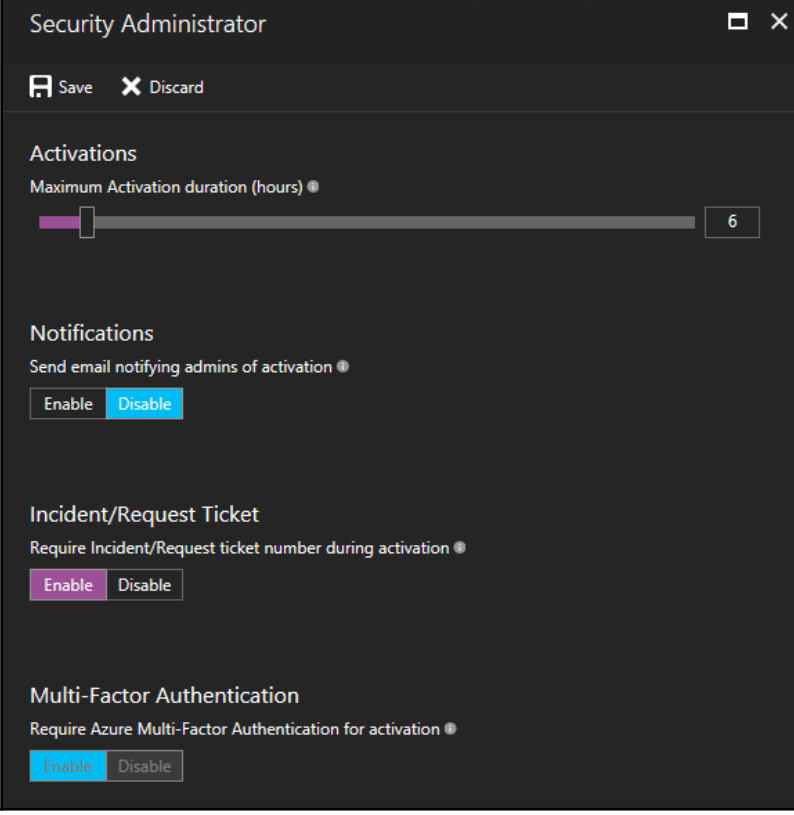
Azure AD PIM main tasks

6. As this subscription looks very boring after enabling Azure AD PIM, we chose to show a demo picture from Microsoft that shows how Azure AD PIM could look in a real-world subscription:



Azure AD PIM dashboard (https://docs.microsoft.com/en-us/azure/active-directory/media/active-directory-privileged-identity-management-configure/pim_dash.png)

7. It's now possible to manage all chosen eligible privileged accounts and roles through Azure AD PIM. Besides removing and adding eligible users to Azure AD PIM, there is also a management of privileged roles, where the role activation setting is available. This setting is used to make privileged roles more transparent, trackable, and to implement the **just-in-time (JIT)** administration model. This is the activation setting blade for the **Security Administrator**:



The screenshot shows the 'Security Administrator' role activation settings. At the top, there are 'Save' and 'Discard' buttons. The settings are organized into four sections: 'Activations' with a slider for 'Maximum Activation duration (hours)' set to 6; 'Notifications' with a toggle for 'Send email notifying admins of activation' set to 'Enable'; 'Incident/Request Ticket' with a toggle for 'Require Incident/Request ticket number during activation' set to 'Enable'; and 'Multi-Factor Authentication' with a toggle for 'Require Azure Multi-Factor Authentication for activation' set to 'Enable'.

Role activation settings for the role Security Administrator

It's also possible to use the rich monitoring and auditing capabilities of Azure AD PIM and never to lose track of the use of privileged accounts and to track misuse easily.

Azure AD PIM is a very useful security feature and it is even more useful in combination with Azure AD Identity Protection.

Identity Protection

Azure AD identity is a service that provides a central dashboard that informs administrators about potential threats to the organization's identities. It is based on behavioral analysis and it provides an overview of risks levels and vulnerabilities.

The Azure AD anomaly detection capabilities are used by Azure AD Identity Protection to report suspicious activities. These enable one to identify anomalies of the organization's identity in real time, making it more powerful than the regular reporting capabilities. This system will calculate a user's risk level for each account, giving you the ability to configure risk-based policies in order to automatically protect the identities of your organization. Employing these risk-based policies among other conditional access controls provided by AAD and EMS enables an organization to provide remediation actions or block access to certain accounts.

The key capabilities of Azure Identity Protection can be grouped into two phases.

Detection of Vulnerabilities and Potential Risky Accounts

This phase is basically about automatically classifying suspicious sign-in or user activity. It uses user-defined sign-in risk and user risk policies. These policies are described later.

Another feature of this part is the automatic security recommendations (vulnerabilities) based on Azure provided rules.

Investigation of Potential Suspicious Events

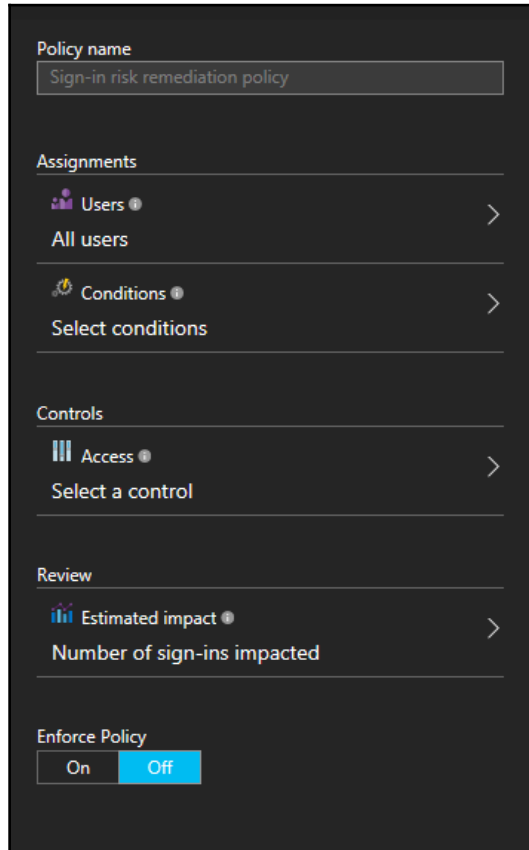
This part is all about investigating the alerts and events that are triggered by the risk policies. So basically, a security-related person needs to review all users that got flagged based on policies and take a look at the particular risk events that triggered this alert and so contributed to the higher risk level. It's also important to define a workflow that is used to take the corresponding actions.

It also needs someone who regularly investigates the vulnerability warnings and recommendations and estimates the real risk level for the organization.

It's important to take a closer look at the risk policies that can be configured in Azure AD Identity Protection.

We will skip the **Multi-factor authentication registration** configurations here. For more details on MFA, read the next paragraph. Just because it can't be said often enough, *we highly recommend enforcing MFA for all accounts!*

The two policies we can configure are user risk policy and sign-in risk policy. The options look quite similar, but the real differentiation happens in the background:



The screenshot shows the configuration interface for a 'Sign-in risk remediation policy' in Azure Security Center. The interface is dark-themed and includes the following sections:

- Policy name:** A text box containing 'Sign-in risk remediation policy'.
- Assignments:**
 - Users:** A section with a purple icon, showing 'All users' with a right arrow.
 - Conditions:** A section with a yellow icon, showing 'Select conditions' with a right arrow.
- Controls:** A section with a blue icon, showing 'Select a control' with a right arrow.
- Review:** A section with a blue icon, showing 'Estimated impact' with a right arrow and 'Number of sign-ins impacted' below it.
- Enforce Policy:** A toggle switch at the bottom, currently set to 'Off' (highlighted in blue) with 'On' as an alternative option.

Sign-in risk policy

In the following diagram, user risk policy view is illustrated:

The screenshot displays the configuration page for a 'User risk remediation policy'. The interface is organized into several sections: 'Policy name' (containing the text 'User risk remediation policy'), 'Assignments' (with expandable options for 'Users' (showing 'All users') and 'Conditions' (showing 'Select conditions')), 'Controls' (with an expandable 'Access' option showing 'Select a control'), 'Review' (with an expandable 'Estimated impact' option showing 'Number of users impacted'), and an 'Enforce Policy' section at the bottom with 'On' and 'Off' toggle buttons. The 'Off' button is currently selected and highlighted in blue.

User risk policy

The main differentiation between Sign-in and User risk policies is where the risk events are captured. The Sign-in policy defines what happens when a certain account appears to have a high number of suspicious sign-in events. This includes sign-in from an anonymous IP address, logins from different countries in a time frame where it would not be possible to travel to the other location, and a lot more.

On the other hand, User risk policies trigger a certain amount of events that happen after the user was already logged in. Leaked credential or abnormal behavior are example risk events.



Microsoft provides a guide to simulating some risks to verify that the corresponding policies trigger and events got recorded. This guide is provided at this address

<https://docs.microsoft.com/en-us/azure/active-directory/active-directory-identityprotection-playbook>.

The interesting thing after choosing users is the **Conditions** setting. This setting defines the threshold of risk events that is required to trigger the policy. The different options for the threshold are **Low and above**, **Medium and above**, and **High**. When **High** is chosen, it needs much more risk events to trigger the policy, but it also has the lowest impact on users. When **Low and above** is chosen, the policy will trigger much more often, but the likelihood of false positives is much higher, too.

Finding the right balance between security and usability is one more time the real challenge.

The last option provides a preview of how many users will be affected by the new policy. This helps to review and identify possible misconfigurations of earlier steps.

Multi-factor Authentication

Authentications that require more than a single type of verification method are defined as two-step verifications. This second layer of sign-in routine can critically improve the security of user transactions and sign-ins. This method can be described as employing two or more of the typical authentication factors, defined as either one, something you know (password), two, something you have (a trusted device such as a phone or a security token) or three, something you are (biometrics).

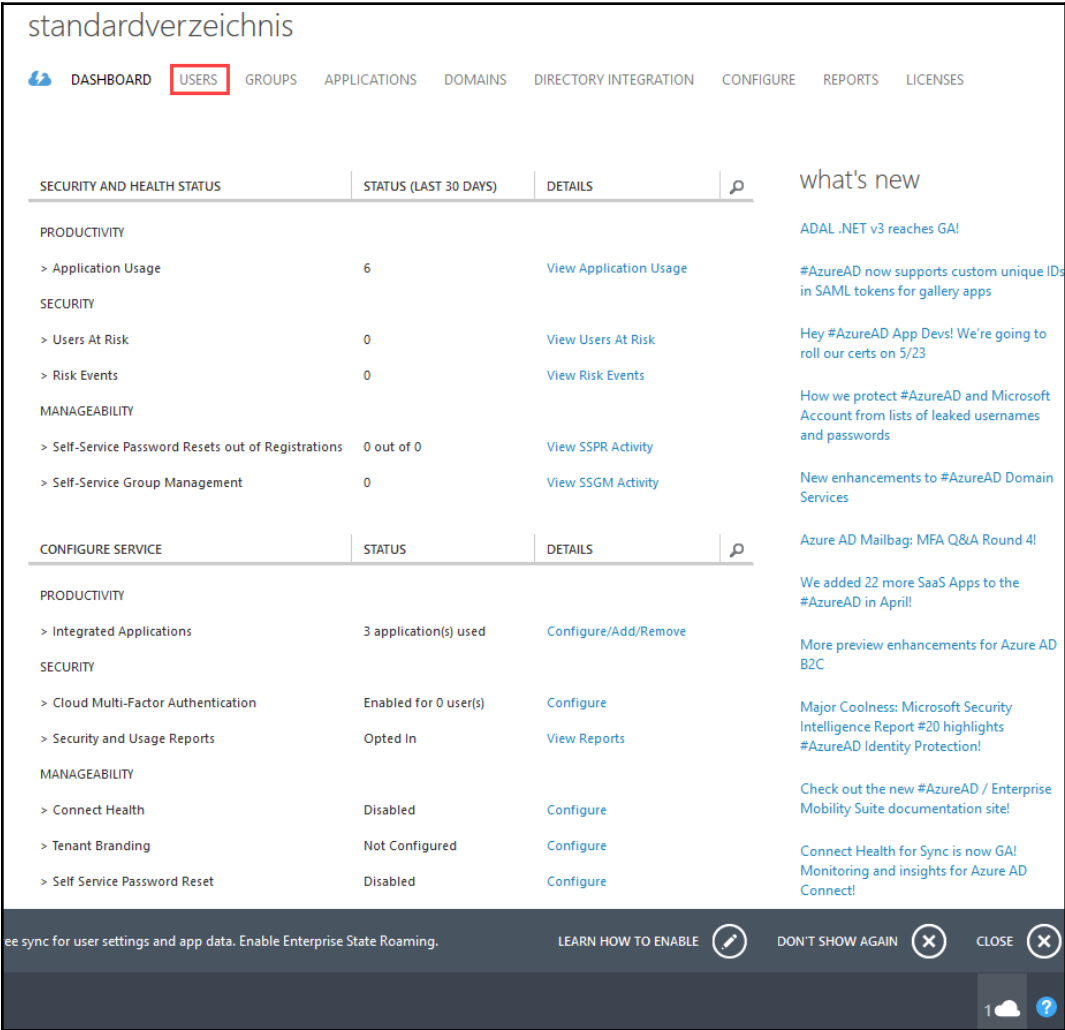
Microsoft uses Azure MFA for this objective. The user's sign-in routine is kept simple, but Azure MFA improves safe access to data and applications. Several verification methods such as phone call, text message, or mobile app verification can help you to strengthen the authentication process.



There are two pricing models for Azure MFA. One is based on users, the other is based on usage. Take your time and consider both models. Just go to the pricing calculator and calculate both to compare them.

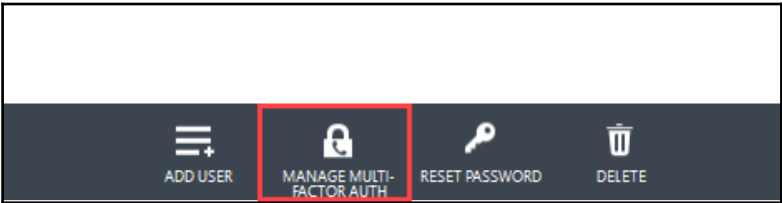
Now we will see how easy it is to activate MFA in Azure:

- 1. First sign in to the old Azure portal (<https://manage.windowsazure.com>). After choosing the right Azure Active Directory, click on **USERS**:



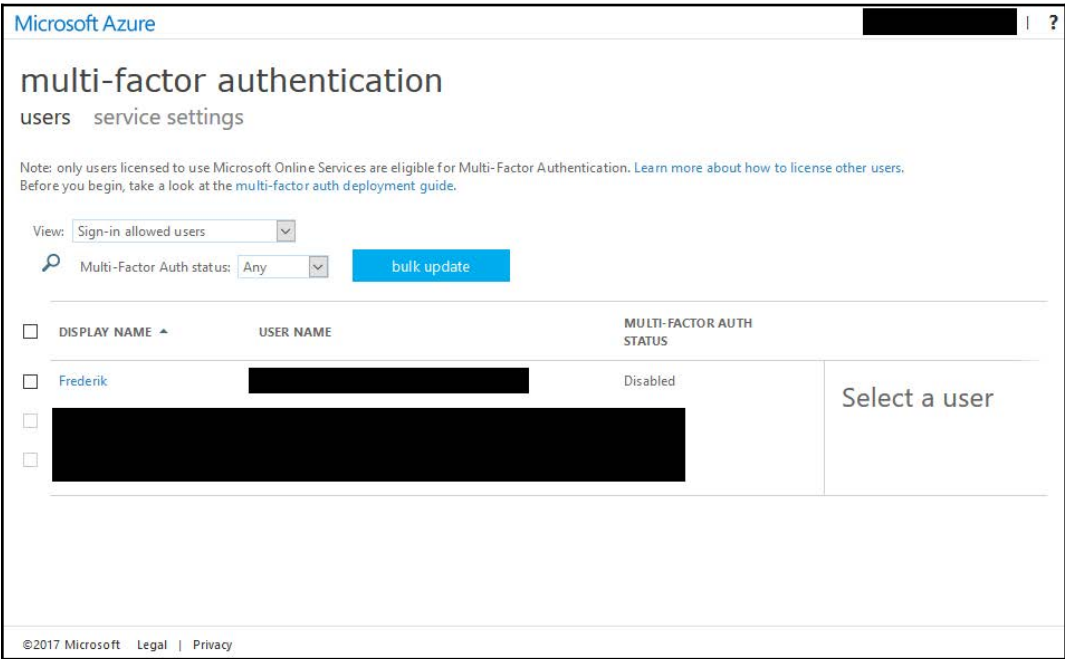
Azure Active Directory in the old Azure portal

2. At the **USERS** page click on **MANAGE MULTI-FACTOR AUTH** at the very bottom:



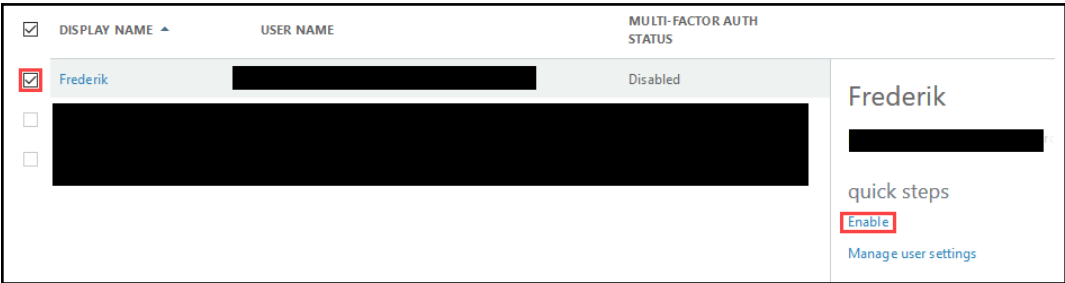
Multi-factor authentication settings

3. Now a redirection to the MFA portal should take place. In this portal, the MFA management takes place. We will use the demo user **Frederik** to show the process of activating MFA:



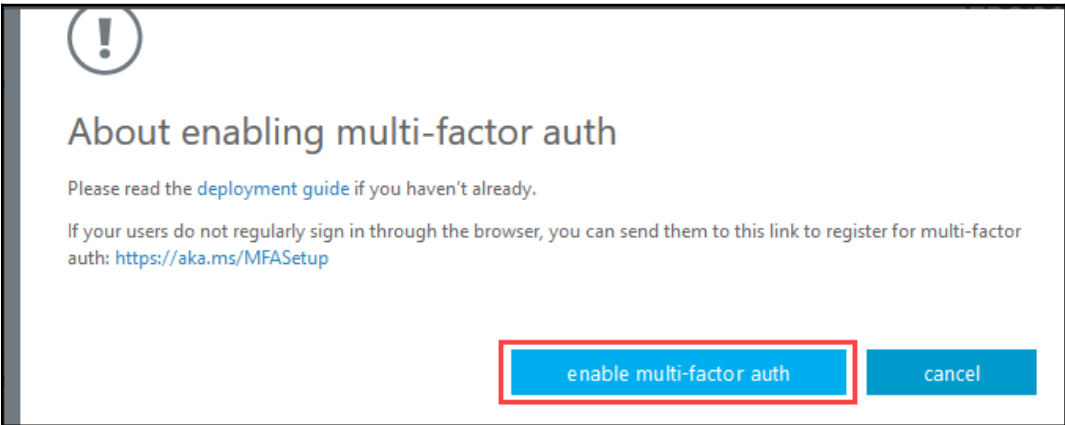
MFA portal

4. Just choose the user that needs to be enabled for MFA and press **Enable**:



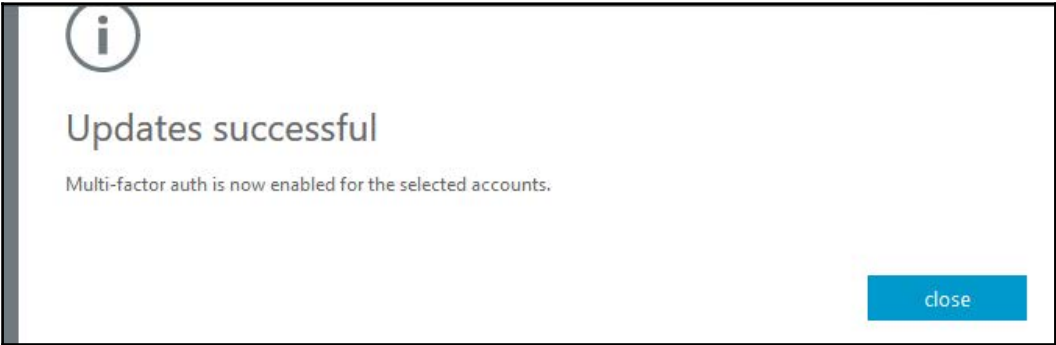
Choosing users for MFA and enabling them is enabled for MFA.

5. After confirming the change, it takes a few seconds and the user is enabled for MFA.



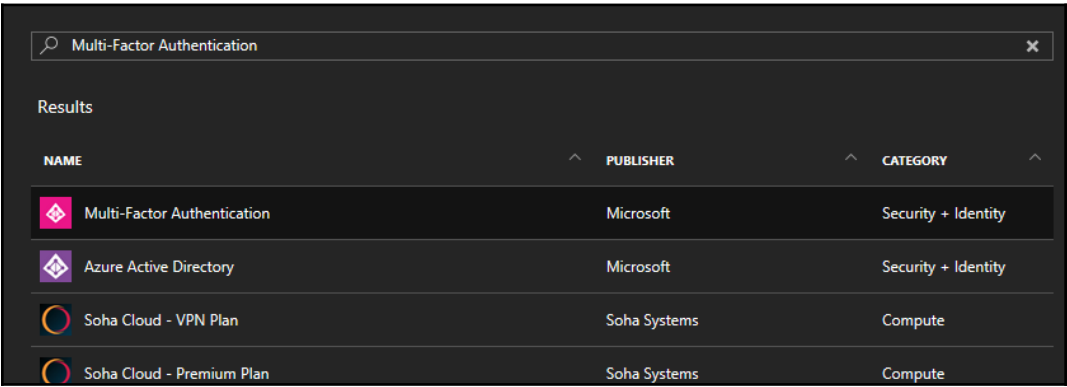
Do you really want to enable MFA?

6. Search for the Multi-Factor Authentication in the search box and then click on it:



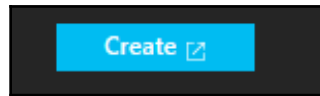
Confirmation for enabling MFA

7. To change the billing model for MFA, a new MFA provider needs to be created to replace the existing one. For this, the **Multi-Factor Authentication** resource should be created from the marketplace:



MFA provider in the marketplace

8. Click on **Create** button to proceed:



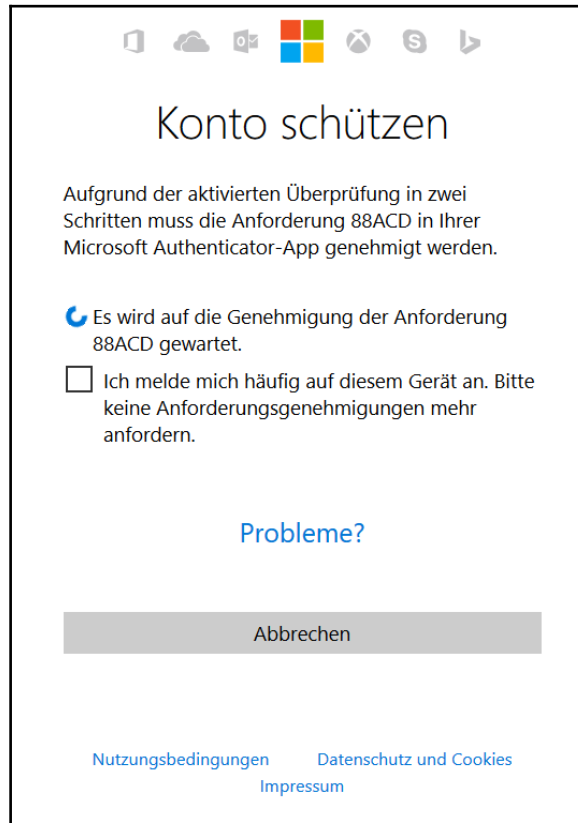
Redirection at creation

9. The little arrow in the box indicates a redirection to the old portal. In the old portal, the MFA provider page directly opens up. There is an exclamation mark next to the usage model. This and the text at the bottom warns that it's not possible to change the usage model afterwards:

New MFA provider

There are many more features related to Azure Active Directory and Identity Security that we are not able to discuss in this book. We encourage you to take a look at Azure AD Connect Health, Azure AD Cloud App Discovery, and Azure Information Protection (as part of EMS). It's important to know what services are offered and what advantages they could offer your company.

This is an example dialogue that will be shown after typing the password, when MFA is enabled and the Authenticator-App was chosen as the main method:



MFA with authenticator-app

Conditional Access

Another important security feature that is based on Azure Active Directory is conditional access. Although it's much more important when working with Office 365 it is also used to authenticate against Azure AD applications.

A conditional access rule grants or denies access to a certain resource based on location, group membership, device state, and the application the user tries to access.

After creating access rules that apply to all users who use the corresponding application, it's also possible to apply a rule to a security group or the other way around, and exclude a group from applying. There are scenarios with MFA, where this could make sense.



Currently, conditional access is completely managed in the old Azure portal (<https://manage.windowsazure.com>). There is a conditional access feature in the new Azure portal, but it is still in preview and not supported for production.

The administrator is also able to combine conditional access policies with Azure AD **Multi-factor authentication (MFA)**.

This will combine the MFA policies with those of other services such as Identity Protection or the basic MFA policy. This means that even if a user is per group excepted from authenticating with MFA to an application, all the other rules still apply. So if there is an MFA policy configured in Identity Protection that enforces MFA, the user still needs to log in using MFA.

In the old portal, the conditional access feature is configured on an application basis, in the new portal the conditional access rules are configured and managed in the Azure Active Directory resource:

multi-factor authentication and location based access rules

ENABLE ACCESS RULES ☐ OFF ☒ ON ?

APPLY TO ☒ ALL USERS ☐ GROUPS ?

☐ Except

RULES

☒ Require multi-factor authentication ?

☐ Require multi-factor authentication when not at work

☐ Block access when not at work

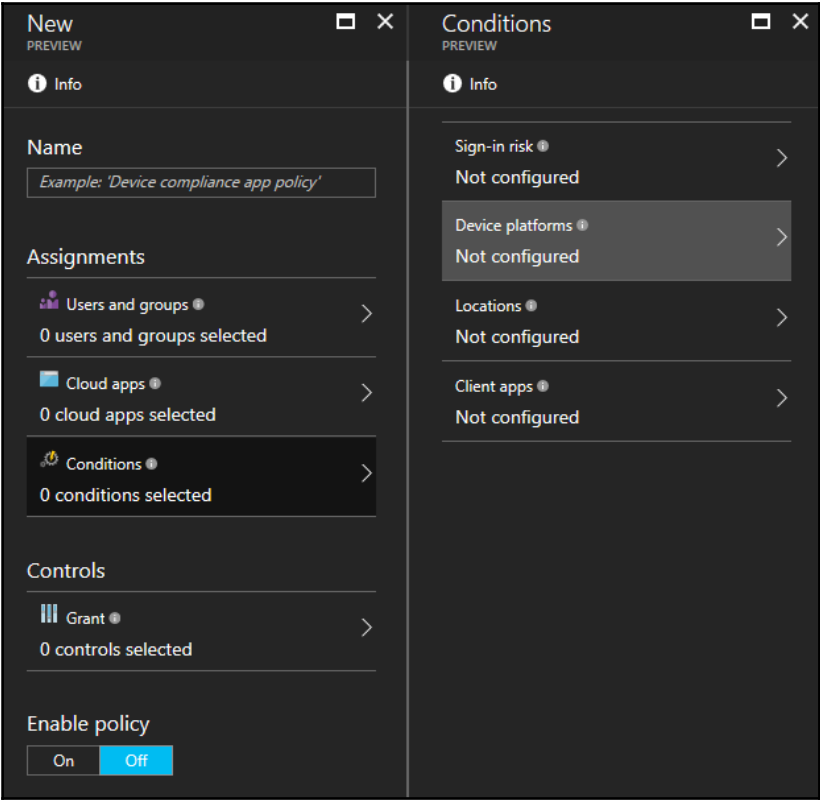
[Click here to define/edit your work network location](#)

device based access rules

ENABLE ACCESS RULES ☐ OFF ☐ ON ?

Per application management old Azure Portal

Following screenshot shows the view of new Azure AD:



Central management in Azure Active Directory in the new portal

Resource Security

The subscription is the lowest level of identity verification (for Global Administrator/Co-Administrator) and the connected Azure AD takes control of all the identities and the assigned roles and rights.

These are the basics for the **Role-Based Access Control (RBAC)** on Azure Resources. With the release of the new portal, the concept of an Azure Resource was introduced. Every service you create in Azure is part of a resource group or is itself a resource.

This is the place where access control takes place in Azure.

Role-Based Access Control

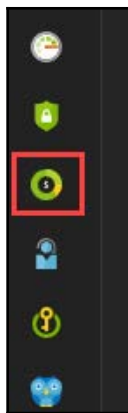
Azure RBAC enables you to use detailed access management for Azure. Employing RBAC makes it possible to make specific choices about how user access is granted in order to perform their tasks (for example, SQL Administration). Two ways of control are possible when using the fine-grained RBAC:

RBAC role assignments are bound to a specific subscription or resource (group). Giving access to a specific Resource, does not imply access to any other resource in the same subscription.

The role can narrow the access even further within the scope of an assignment. These roles can be on a general level, such as owner, but also specific such as virtual machine reader.

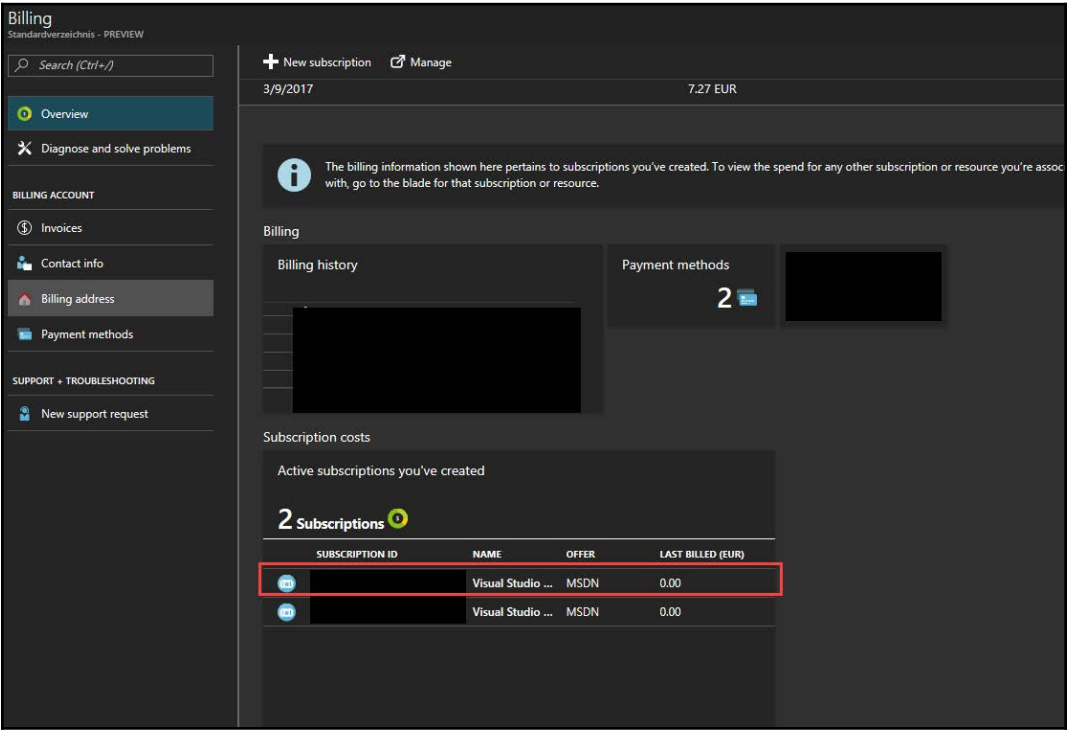
We will take a look at RBAC roles as well as role assignment:

1. First, navigate to the **Billing** blade:



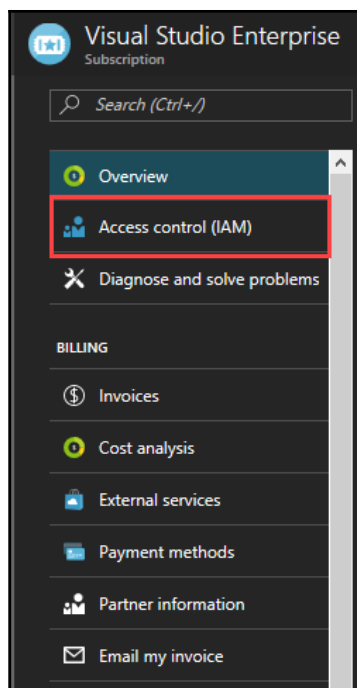
Billing button in Azure portal

2. After clicking on the **Billing** option, you will have a new window named **Billing**:



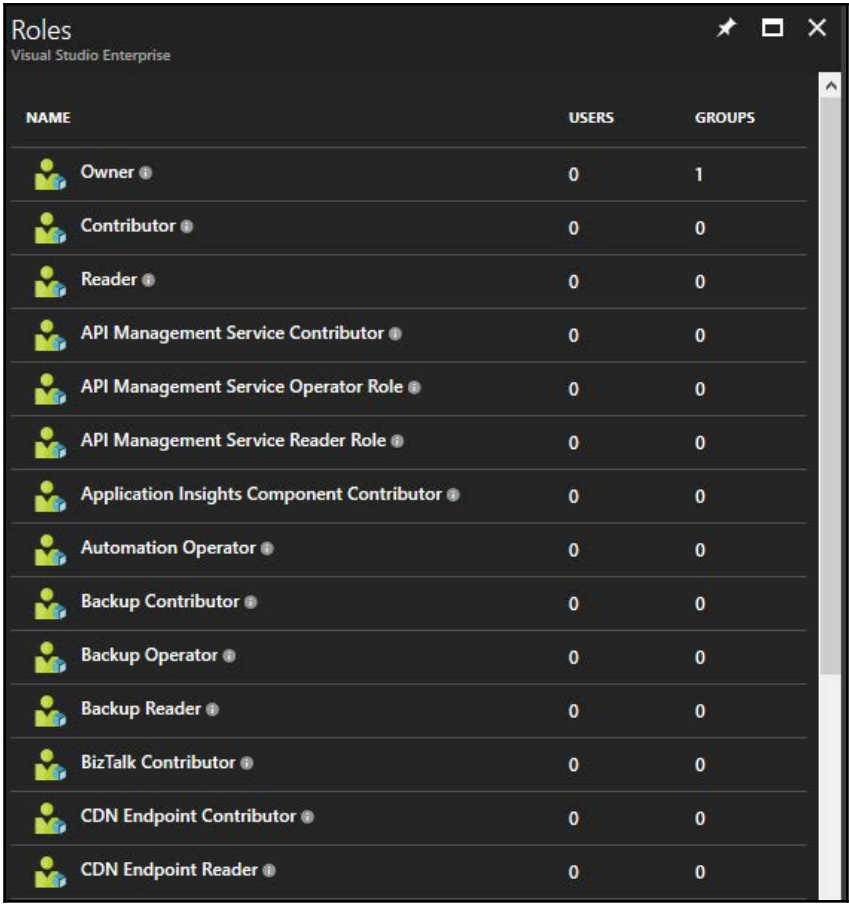
Billing dashboard

3. Now select the subscription that needs to be managed:



Subscription management

4. In the Subscription management, choose **Access control (IAM)**. You will be presented with an overview of users and groups connected to that subscription and their assigned roles. To see which roles are existent in your subscription, click on **Roles**:

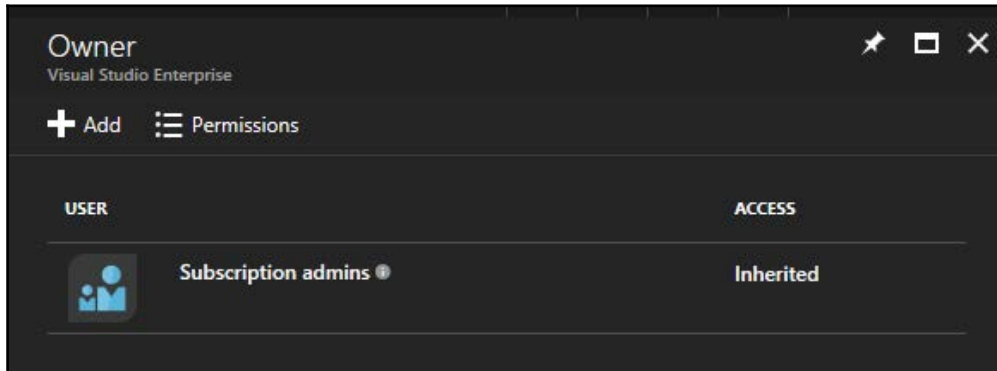


The screenshot shows a window titled 'Roles' from Visual Studio Enterprise. It contains a table with three columns: 'NAME', 'USERS', and 'GROUPS'. The table lists 14 different roles, each with a user icon and a help icon. The 'USERS' column shows 0 for all roles except 'Owner', which shows 1. The 'GROUPS' column shows 1 for 'Owner' and 0 for all other roles.

NAME	USERS	GROUPS
 Owner ⓘ	0	1
 Contributor ⓘ	0	0
 Reader ⓘ	0	0
 API Management Service Contributor ⓘ	0	0
 API Management Service Operator Role ⓘ	0	0
 API Management Service Reader Role ⓘ	0	0
 Application Insights Component Contributor ⓘ	0	0
 Automation Operator ⓘ	0	0
 Backup Contributor ⓘ	0	0
 Backup Operator ⓘ	0	0
 Backup Reader ⓘ	0	0
 BizTalk Contributor ⓘ	0	0
 CDN Endpoint Contributor ⓘ	0	0
 CDN Endpoint Reader ⓘ	0	0

List of existing roles

If you have not yet configured custom roles, a list of all built-in roles and their assigned users and groups will be displayed. To assign users or groups to that role click at the role and use the **Add** button at the top to add new users or groups to the role. This assigns the role and the associated rights to the user or to all users of the group. In the following screenshot, the default role of **Owner** gets assigned to all subscription admins:



Assigned users and groups of the Owner role

Custom roles must be created programmatically through one of the existing interfaces. Azure PowerShell is a good way for beginners to achieve that task. First, someone has to define the exact rights that the custom role should provide. These are very granular rights for single resources and they look as follows:

```
"Microsoft.Network/*/read"  
"Microsoft.Compute/*/read"  
"Microsoft.Compute/virtualMachines/restart/action"
```

Now, these are used to create a JSON-based role definition file that will be uploaded to Azure for role creation. To do that, use the following cmdlet:

```
New-AzureRmRoleDefinition -InputFile "C:temproleDefinition.json"
```



To find out how exactly custom role creation works, go to <https://docs.microsoft.com/en-us/azure/active-directory/role-based-access-control-custom-roles>.

Resource Tags and Policies

Although tagging is more a governance than a security topic, we should at least explain what tags are and how they are used to improve security. Tags are used to organize resources in Azure. They can be added to a Resource group as well as to single resources. They provide up to 15 key value fields to add some information to the object. This could be a billing related field such as a department or project or just general information. These tags could also be used as a method for filtering events and applying policies to certain resources.

Examples of tags could be `startup-time`, `shutdown-time`, `customer`, `cost-center`, `accountable`. In general, tags are mainly used to help in organizing/automating and to improve the governance.

Azure resource manager policies are a way to control resource deployment to, for example, meeting compliance rules. This could be for example a rule, that requires resources to be deployed in a specific region. A resource policy is basically a JSON-based if-then statement that is checked at resource deployment. Resource policies can be used to:

- Append
- Deny
- Audit

when a specific condition is met. The audit feature is extremely interesting for security investigations. When suspicious activities happen in a Resource group or a resource, audit policies could help with logged events for several actions. It could also help to track the actions of suspicious users.



For more information on Resource manager policies, use this link
<https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-manager-policy>.

Azure Network Security

Azure network security is based on classic network security. Just because the hardware underneath is managed and the whole networking is virtualized doesn't mean that all classic network security principles are invalid.

Network Security Groups

A very basic feature when it comes to network security in Azure is the network security groups. It would be too much to call them firewalls, but they fulfill basic firewall activity. They are best compared to access lists on a Layer 3 network switch. They consist of inbound and outbound network security rules. These rules control the network traffic flow and can be given several parameters at creation.

Priority

The **Priority** of a rule defines the order in which the rules are applied to traffic. The rules are processed in order, what means that a lower number means a higher priority.

Source

There are a few different setting for source traffic. The source traffic needs to be categorized first. An important setting is the **Tag** categorization. This makes it possible to choose from the **Source** from **Internet**, **AzureLoadBalancer**, and **VirtualNetwork**. Where **Internet** means the public IP address space, **VirtualNetwork** the private and virtual address space, and **AzureLoadBalancer** the traffic that originates from deployed load balancers to determine the service health.

The last setting is the **Protocol**. There are many service templates for common services such as RDP, SMTP, or FTP. This makes it easy to quickly create some rules in the portal.

Destination

Destination has the same settings as **Source** besides the **Protocol** setting, as this is already defined by the **Source** setting.

Note that all port and protocol settings get disabled when choosing a service template, because the standard ports are used. When you create a rule for a network service that doesn't use the standard ports, use a custom setting. This is how a new network rule looks:

The screenshot shows the 'Add inbound security rule' dialog for 'AzureDemoVM-nsg'. The 'Basic' tab is selected. The rule configuration is as follows:

- Name:** denyRdpIn (with a green checkmark)
- Priority:** 1100 (with a green checkmark)
- Source:** Any (selected from a dropdown menu)
- Protocol:** TCP (selected from a dropdown menu)
- Source port range:** * (empty field)
- Destination:** Any (selected from a dropdown menu)
- Destination port range:** 3389 (with a green checkmark)
- Action:** Allow (selected from a dropdown menu)

New inbound security rule

Network security groups can be associated with multiple network interface cards and multiple subnets. That means that all security rules defined in that security group apply to the complete network traffic in all associated subnets.

By default, all subnets are routed. To implement basic security based on NSGs, it's best practice to implement traffic flow control between subnets first. After connecting to an on-premise network via VPN or ExpressRoute (see *Chapter 4, Implementing Azure Networks*), it's also very important to ensure that all unnecessary traffic from the Internet is blocked and that all Internet-facing services/virtual machines are isolated by, for example, a DMZ.

Network Virtual Appliances

Although there are already extensive built-in network protection capabilities against threats from the outside, like DDoS and other common attack patterns, these are mainly protecting Azure as a platform. That means, that when one customer is affected by a DDoS attack, no other customer will be influenced by that. But that also means, that you still have to protect yourself against DDoS attacks against your own instances.

When it comes to advanced network filtering or routing scenarios Azure has *network virtual appliances* in place. These are virtual appliances that are deployable from the marketplace. They are third-party solutions for different scenarios from vendors such as Barracuda, Cisco, or F5. If none of the marketplace appliances meet the requirements, it's possible to create an NVA based on a VM. With that method, a free firewall solution such as pfSense could be used for several scenarios including test environments.

To use appliances in Azure, network traffic needs to be routed through them. This is done with **user-defined routes (UDRs)**. User-defined routes can be deployed to control traffic flow in virtual networks.

Common scenarios for network virtual appliances are:

- Separation of a corporate and Azure network (for example, additional scanning of traffic that's gone through a VPN)
- VPN to Azure-NVAs can also be used to set up a VPN tunnel instead of using the Azure VPN Gateway
- WAN optimizer architectures
- Specific scanning such as web application filtering, additional DDoS protection, IPS/IDS scenarios, or other detailed traffic analysis

User-defined routes are also used to configure forced tunneling. Forced tunneling is used to control outbound Internet traffic and it helps to meet compliance rules.

With the help of UDRs, outbound traffic is routed back to the company network and is then redirected to the Internet. A central internet breakout helps in cases where it's important to apply policies for antivirus scanning, data leakage protection, or other security measures.



To learn more about user-defined routes, use the following link <https://docs.microsoft.com/en-us/azure/virtual-network/virtual-networks-udr-overview>.

For available third-party appliances, check out this marketplace link <https://azuremarketplace.microsoft.com/de-de/marketplace/apps/category/networking?page=1&subcategories=appliances>.

Azure Storage Security

Azure Storage is the base for nearly all IaaS services. It's a highly scalable, available, and fully virtualized cloud storage solution that can be used to provide storage space in many ways.

When talking about storage security it's very important to remember that all earlier discussed security measures such as RBAC or strict identity management should already be deployed.

After that is ensured, it's useful to leverage the special security abilities of Azure Storage. These include:

- Shared access signatures
- Data encryption in transit
- Storage account encryption at rest
- Virtual Hard Disk encryption
- Advanced analytics

Key Management

Azure uses 512 bit strings as storage account keys. Paired with the storage account name it enables the user to access the objects in the storage account, for example, blobs, entities within a table, queue messages, and files on an Azure Files share. The access to the data plane of a Storage account is based on the controlling of the access to the Storage account keys.

Two keys, here defined as `Key 1` and `Key 2` in the PowerShell cmdlets and the Azure portal, are assigned to each storage account. By employing the Azure portal, PowerShell, Azure CLI, .NET Storage Client Library, or the REST API these can be manually regenerated if desired.

Regenerating your Storage account keys can have several reasons. One option would be to regenerate them in regular time intervals to improve the security of your organization, especially if someone was able to hack into your account or obtain the keys from a hardcoded piece of software. This would give the (evil) person full access to your Storage account. If several different SAS have been compromised, you should replace all of them, especially if they are based on the same Storage account key.

Make sure that you have a list of all your applications and Azure services, which are possibly dependent on the Storage account before you regenerate your keys. For example, if you use a Storage account in combination with Azure Media Services, make sure to re-sync the access keys in your Media Services after regenerating the key. This also applies to other services as well, such as the storage explorer. It is important to notice that the VHD files of your VMs stored in your storage account are not affected by this action. After regenerating your keys in the Azure portal, it will take up to 10 minutes to be synchronized across all storage services.

The following part contains the general process of how to change your keys. It is assumed that currently `Key 1` is in use and the goal is to change to `Key 2`. Just reverse the key names for the process if you are currently employing `Key 2`.

First, you should regenerate `Key 2` in the Azure portal to make sure it is secure.

You can change each application to use the new key and publish it as part of the migration process. After you have completed this, keep in mind to regenerate the old key, so it doesn't work anymore.

Storing the key in Azure Key Vault and making the application retrieve from the service is another possible option. An advantage of this method is that the applications will not need to be redeployed, because they can get the key from the updated Azure Key Vault automatically. It is also possible to cache the key in your memory and only read it from the Azure Key Vault if it fails when using it.

Another level of security is added by deploying Azure Key Vault, because it avoids the key from being hardcoded in the configuration file, making it vulnerable to several attack vectors.

Using Azure Key Vault also gives you the ability to control access to your keys with the use of Azure Active Directory. Applications that require access to keys from Azure Key Vault can be granted the information, while making sure that other applications don't have this privilege without the specific permission of you.

Shared Access Signatures

It is important to remember that sharing your storage account key grants complete access to that person. Someone could potentially replace your files with virus-infected versions or steal the data. So, granting someone unlimited access to your storage account is a very risky practice. A shared access signature consists of query parameters as a part of the URL, aiming to provide information about the access parameters and the period for which access is granted.

Shared access signatures can only be created programmatically or within the Azure storage explorer.

The storage service uses input query parameters from the received request and creates a signature based on the method of the calling program. The service checks if the two signatures are identical. In case of a positive result, the storage then validates other parameters, such as storage service version, correct time and date in the specified window, if the access requested corresponds to the request made, and so on.

There are two types of shared access signatures.

Service-Level SAS

A service-level SAS enables the user to access specific resources in a Storage account. In this kind of SAS, this could, for example, be retrieving a list of blobs in a container, the download of a blob, the update of an entity in a table, or uploading a file to a file share.

Account-Level SAS

An account-level SAS is capable of all service-level SAS plus several additions. Multiple options can be chosen, which are not available in service-level SAS, giving you the ability to create file shares, tables, queues, and containers. It is possible to specify access to several services at once, for example, enabling you to give someone access to blobs and files in your storage account at the same time.

Remember that all SAS that are signed with a specific storage access key are invalidated, as soon as the access key is re-generated.

Storage Encryption

As customer and compliance needs including encryption at every stage, there are several ways to achieve encryption at rest in Azure.

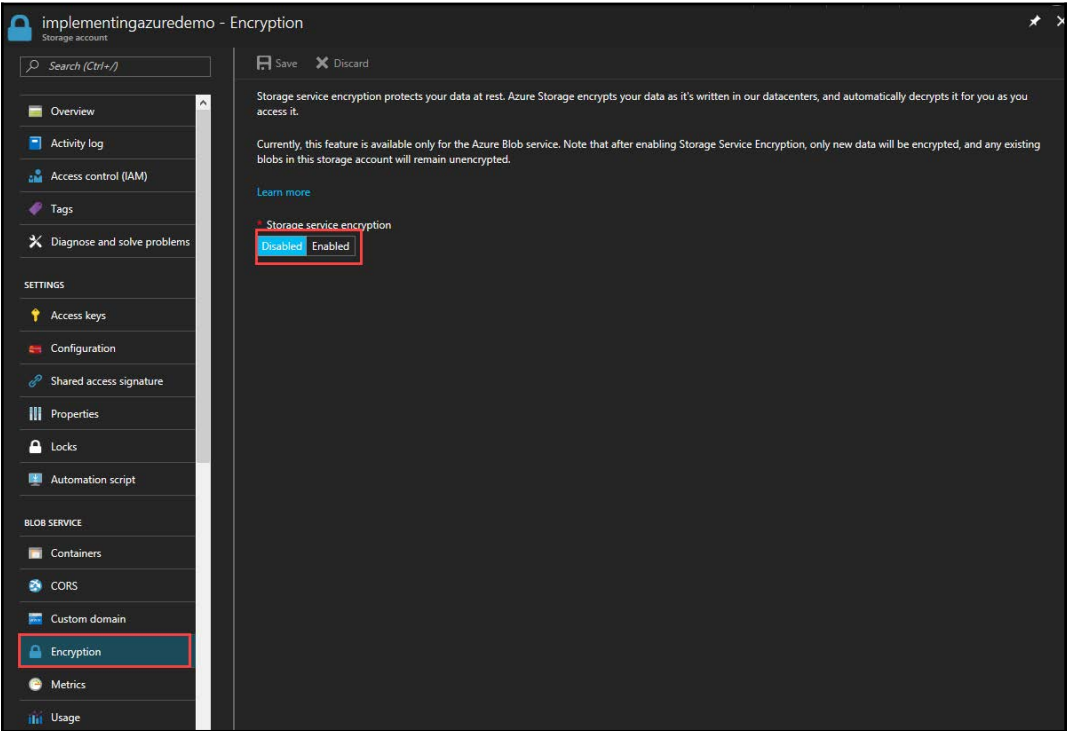
Storage Account Encryption

In order to protect and safeguard your data to meet organizational security and compliance commitments, Azure **Storage Service Encryption (SSE)** is used. Prior to persisting to storage and retrieval, Azure Storage encrypts and respectively decrypts your data. Currently, SSE only takes care of blobs and files (Preview) that are stored on a Storage account.



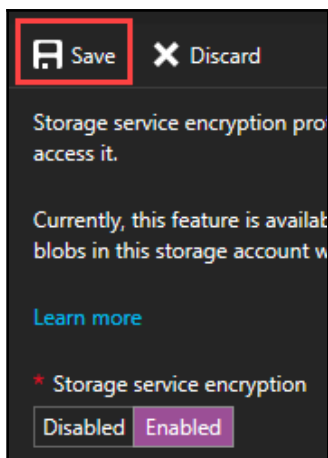
Currently, there is a preview feature called SSE for Azure File Storage. This feature uses the same techniques as SSE for blobs to protect the Azure storage file service.

1. The following screenshots provide guidance on how to enable Storage account encryption on any supported storage account:



Blob encryption

2. Now click **Enabled** and **Save**:



Enabling storage encryption

That's it. Storage encryption is enabled now.



It's important to remember that the process of SSE is completely transparent. That means that all keys, as well as encryption and decryption, is managed by Microsoft. Because of that, there is no possibility to receive, back up, or use your own keys for the AES-256 encryption taking place.

Azure Disk Encryption

It is fully supported to enable and use Azure SSE for **Virtual Hard Disks (VHD)** that are used by virtual machines. But as SSE is only used on data that is added after the encryption was turned on, the basic VHD would stay unencrypted.

SSE has full support for marketplace resources, so what basically happens is that new writes from the virtual machine to the storage account will be encrypted, while the VHD used for enrolment stays untouched.

To meet compliance and security requirements, Microsoft introduced a service that explicitly takes care of virtual machines in Azure.

This service is called Azure disk encryption and it uses internal system technologies to encrypt the virtual disks of virtual machines. This software is BitLocker in Windows and DM-Crypt in Linux VMs.

Virtual machine disk encryption can currently only be enabled using PowerShell with the following steps:

1. At first, we need a service principal in the Azure AD. Afterwards, an Azure Key Vault is needed to store the key encryption key.
2. After these are set up, the principal needs to be connected to the Key Vault for managing keys.
3. After that is done and the Azure AD application is set up for certificate-based authentication, this PowerShell cmdlet is used to enable disk encryption for a certain VM:

```
Set-AzureRmVMDiskEncryptionExtension -ResourceGroupName  
$resourceGroup -VMName $vm -AadClientID $adApplication -  
AadClientCertThumbprint $certThumb -DiskEncryptionKeyVaultUrl  
$encryptionUrl -DiskEncryptionKeyVaultId $kvId
```

Because many technologies involved in this process are changing very fast, we did not mention the whole process here.



To find the current implementation and learn how to implement disk encryption, please use this link

<https://docs.microsoft.com/en-us/azure/security/azure-security-disk-encryption>.

Logging and Monitoring

Logging and monitoring are the most important techniques in today's threat landscape. Logs and monitoring data is the basis for later analysis and behavioral analysis of network traffic, user actions, failed processes, and more. The more data you have, the more likely you are to find an anomaly or a pattern that can be used for automated threat analysis.

But as data is nothing without former analysis, the visualization and analytic tools are as important as the behavioral threat detection tools.

Azure Logs

Azure collects lots of logs for you by default. There is a central dashboard for viewing logs and events called **Monitor**. The Monitor is still in preview, but in our opinion, it is the best-organized place to manage logs and diagnostics.

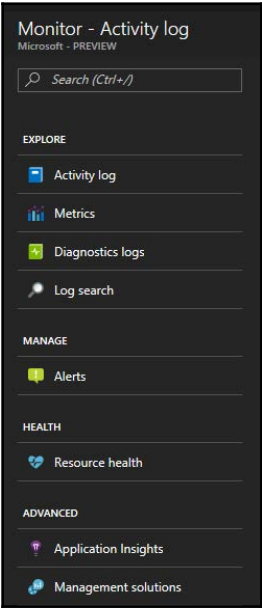
There are three main categories of logs that are collected. These are under the **EXPLORE** point in the Azure Monitor.

- **Activity log:** The activity log keeps track of all actions that are happening on Azure. It logs all activities no matter if they were initiated by users or other resources including resource creation, change of resources and resource deletion. Activity logs can be analyzed using different Azure Services. These include PowerBI, Event Hub or just save them to a storage account to handle them otherwise.
- **Metrics:** Metrics are basic telemetry logs for resources like Network traffic, CPU percentage, context switches etc.. It depends on the monitoring settings that are configured for the resources if all metrics are available. The metrics that can be collected from the fabric have the [Host] prefix.

Metrics can also be used as a trigger for alerts. These alerts can be used to trigger events like a custom script, an alert mail or a WebHook when a certain threshold is reached.

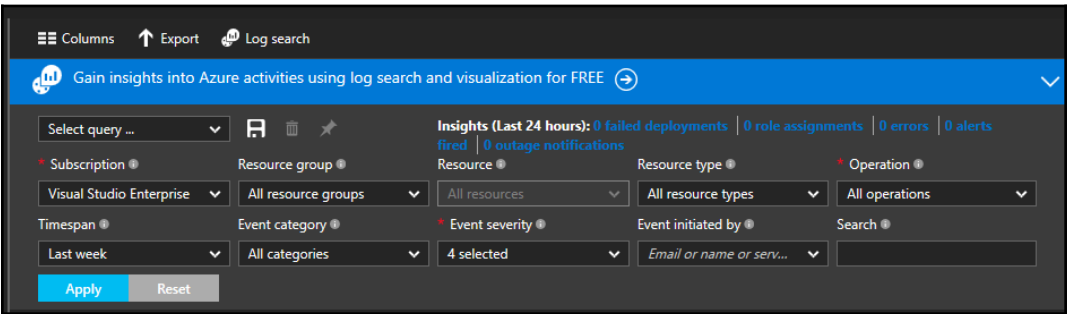
- **Diagnostic log:** Diagnostic Logs contain very specific logs for individual resources. For example, when diagnostic logging on a network security group is activated, the diagnostic logs contain events connected to NSGs and their rules as well as several counters.

Diagnostic Logs can also be used as a source for analysis by Operation Management Suite Log Analytics.



Azure monitor management blade

In the following screenshot log filter options are shown:



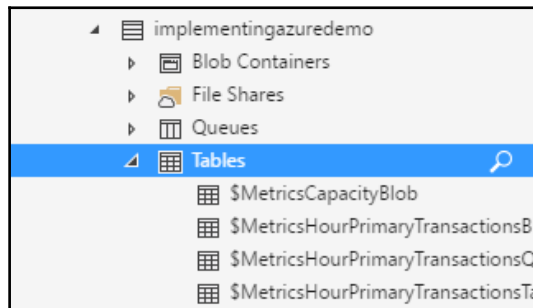
Log filtering options

There is a special kind of logs for storage accounts. These logs are saved directly in the storage account itself. They can be accessed by PowerShell, REST API or by using the Azure Storage Explorer that was introduced in Chapter 5, *Implementing and Securing Azure Storage Accounts*.

To view the metrics recorded for the Storage account `implementingazuredemo` you need to log in, provide an SAS or Access Key first. Now you should be able to see the content of the storage account. Expand the tables node and there will be the metric tables for several counters.

These are

- Capacity metrics
- Transaction Metrics Blob
- Transaction Metrics Queue
- And Transaction Metrics Table.



Storage account metrics

Azure Security Center

Azure Security Center is an offering, that helps users to protect their environments and helps to detect suspicious events and vulnerabilities.

It's able to monitor Azure Virtual Machines, Azure SQL Databases, Azure Virtual Networks, and many more PaaS and IaaS services.

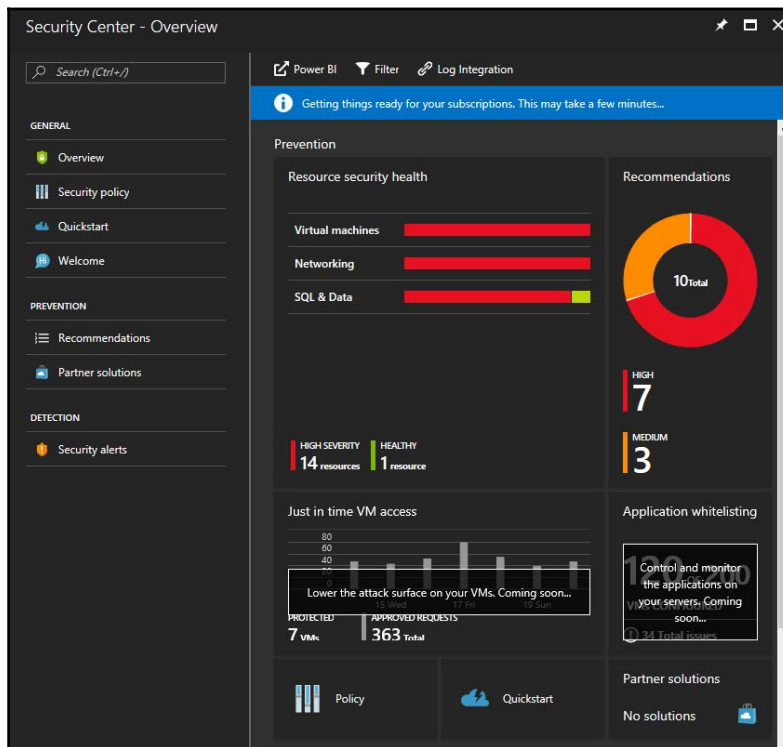
Based on the collected metrics it's possible to detect threats and increase the transparency of critical actions based on rulesets and threat analytics.

With the help of the security center, the Security Operator always has an overview of the current security threats and the state of security measures as well as vulnerabilities across all Azure resources.



All collected data is stored in your own storage account, that is created per region on policy creation.

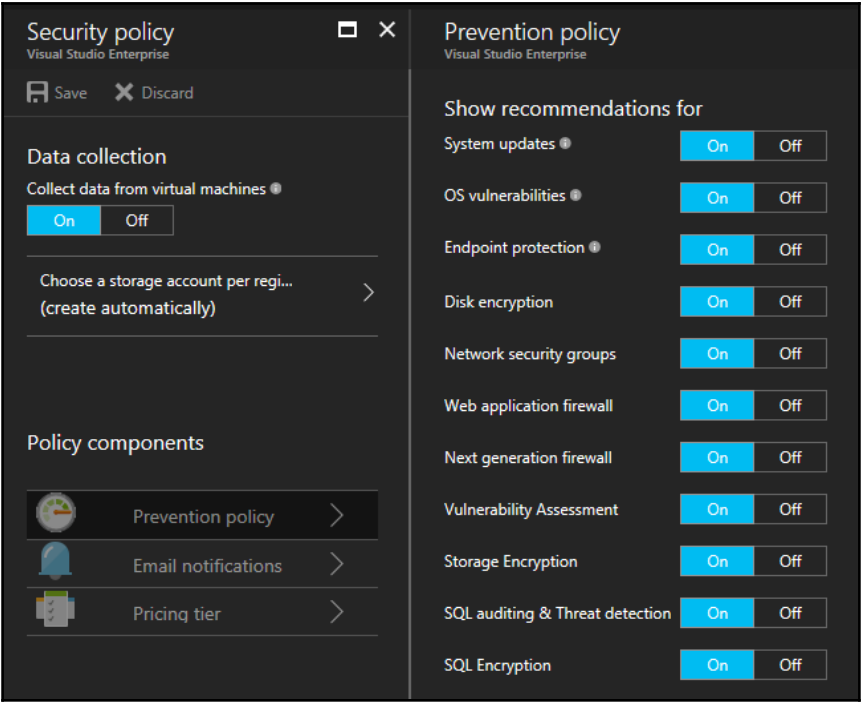
When opening the Security Center by clicking on the icon at the left, it opens the dashboard. At first time launch, the security center will tell you, that it is working with limited function. This is because just the basic version of the Security Center is free. For evaluating, it is recommended to use the free test ride, because many of them, in our opinion, are very valuable, functions are only available in the paid version.



Azure Security Center dashboard

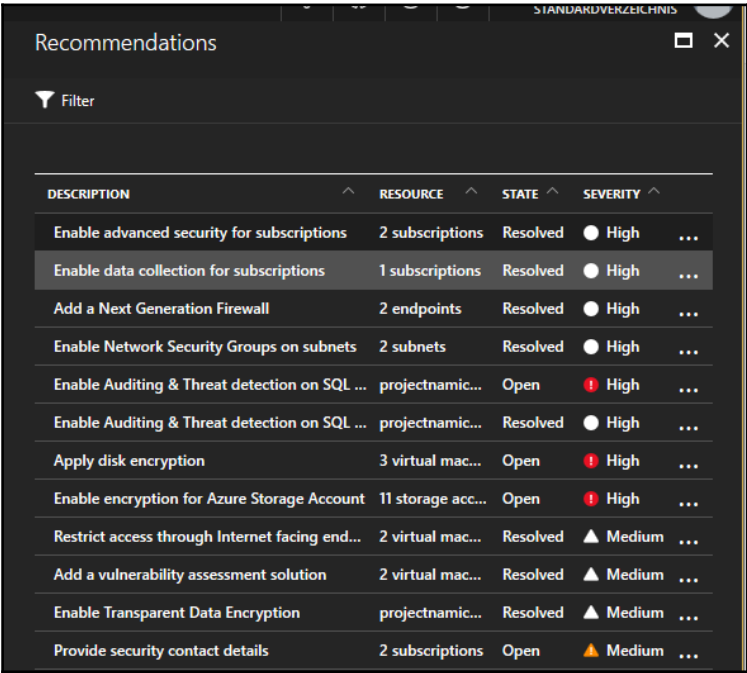
After launching the dashboard, you will realize, that it works on a subscription base. What that means is that the Security Center provides recommendations based on policies that are applied to single subscriptions. To change a policy just click on **Security policy** then click on the subscription that you want to change.

In the subscription blade, it's also possible to choose a pricing plan, activate the free trial, configure notification option and, what we are here for, configure the **Prevention policy**:



Configure the Prevention policy

After it configures which resources should be collected data for, recommendations will show up in **Recommendation** as well as potential security alerts, as long as you use the standard pricing:



DESCRIPTION	RESOURCE	STATE	SEVERITY	
Enable advanced security for subscriptions	2 subscriptions	Resolved	● High	...
Enable data collection for subscriptions	1 subscriptions	Resolved	● High	...
Add a Next Generation Firewall	2 endpoints	Resolved	● High	...
Enable Network Security Groups on subnets	2 subnets	Resolved	● High	...
Enable Auditing & Threat detection on SQL ...	projectnamic...	Open	! High	...
Enable Auditing & Threat detection on SQL ...	projectnamic...	Resolved	● High	...
Apply disk encryption	3 virtual mac...	Open	! High	...
Enable encryption for Azure Storage Account	11 storage acc...	Open	! High	...
Restrict access through Internet facing end...	2 virtual mac...	Resolved	▲ Medium	...
Add a vulnerability assessment solution	2 virtual mac...	Resolved	▲ Medium	...
Enable Transparent Data Encryption	projectnamic...	Resolved	▲ Medium	...
Provide security contact details	2 subscriptions	Open	⚠ Medium	...

Security Center recommendations

Summary

Let's summarize this chapter. You should have learned the basics of Azure security. This means understanding principles and possibilities of implementation and maintaining secure Azure environments.

It's very important to create your architecture with security in mind and Microsoft Azure does help a lot in taking complexity away and focusing on the important parts of IaaS security.

In the next chapter, we will see some example architecture from real-world examples.

10

Azure Design Patterns

In the previous chapter, we got an overview of Azure cloud and explained some of the important concepts related to it. This chapter is about Azure cloud patterns related to virtual networks, storage accounts, regions, and availability sets. These are important constructs that affect the final architecture delivered to customers in terms of cost, efficiencies, and overall productivity. This chapter will also briefly discuss cloud patterns that help in implementing scalability and performance within the overall architecture.

In this chapter, we'll cover the following topics:

- Azure virtual network design
- Azure storage design
- Azure zones, regions, and availability sets
- Azure design patterns related to the following:
 - Messaging
 - Performance
 - Scalability

Azure Zones and Regions

Azure cloud is backed up by large data centers interconnected into a single large network. The data centers are grouped together based on their physical proximity to Azure regions. For example, a couple of data centers in Western Europe together are available to Azure users as the West Europe region. Users cannot control which data center that is, the exact data center for their deployment. They can inform Azure about the region and Azure will pick an appropriate data center.

Choosing an appropriate region is an important architectural decision as it affects the following:

- Availability of resources
- Data and privacy compliance
- Performance of applications
- Cost of running applications

Availability of Resources

Not all resources are available in every Azure region. If application architecture demands a resource and it is not available in a region, choosing that region will not help. Instead, a region should be chosen based on the availability of all resources required by the application. It might be so that the resource might not be available while developing the application architecture and could be on the Azure roadmap to make it available subsequently.

For example, log analytics is not available in all regions. Data transfer to log analytics would incur egress network cost if the application is deployed in a region that does not have log analytics (previously known as **operational management suite**). Similarly, Azure key vault provides services to only resources that are co-located in the same region. Another example is that an Azure virtual network can host virtual machines, load balancers from same regions. Virtual machines from different regions cannot be part of the same virtual network.

Data and Privacy Compliance

Each country has its own rules for data and privacy compliance. Some countries are very specific that their and their citizen's data cannot be stored in any other country. These are legal requirements and should be taken into architectural consideration for an application.

Performance of Applications

The performance of an application is dependent on the network route that was taken by requests and response from users. If an Azure region is near the users of the application, users will get better performance from the application compared to users far away from the Azure region. An application deployed in Western Europe for users in south-east Asia will not perform as well as an application deployed at East Asia region for users in in south-east Asia.

Cost of Running Applications

The cost of Azure services differs from region to region. A region with an overall cheaper cost should be chosen. There is a complete chapter on cost management in this book and it should be referred to for more details from the cost perspective.

Virtual Network

A **virtual network** should be thought of as a physical LAN network in your office or home. Conceptually, they both are the same, although the Azure virtual network is implemented as a software-defined network backed up by a giant physical network infrastructure.

A virtual network is required for hosting a virtual machine. They provide a secure communication mechanism between Azure resources to connect to each other. They provide an internal IP address to them, access, and connectivity to other resources including virtual machines on the same virtual network, routing of requests, and connectivity to other networks.

A virtual network is contained within a resource group and is hosted within a region, for example, West Europe. Virtual network cannot span multiple regions, but it can span all data centers within a region. For connectivity across regions, virtual networks can be connected using the VNET-to-VNET connectivity. The virtual network also helps in connecting to on-premise data centers enabling hybrid cloud. There are multiple types of VPN technologies available to connect to on-premise data centers such as site-to-site VPN and point-to-site VPN. There is also a dedicated connectivity available between Azure virtual network and on-premise network using **ExpressRoute**.

A virtual network is free of charge. Every subscription can create up to 50 virtual networks across all regions. However, it can be increased by reaching out to Azure support. There will be no charge for data transfer within a VNET



Information about networking limits is available at <https://docs.microsoft.com/en-us/azure/azure-subscription-service-limits#networking-limits>.

Architectural Considerations for Virtual Networks

Virtual networks like any other resource can be provisioned using ARM templates, REST APIs, PowerShell, and CLI. It is quite important to plan the network topology at the beginning stage itself to avoid troubles later in the development life cycle. This is because once the network is provisioned and resources start using it, it is difficult to change it without having a downtime. For example, for moving a virtual machine from one network to another will require the virtual machine to shut down.

- **Regions:** The virtual network is an Azure resource and is provisioned within a region such as West Europe. Applications spanning multiple regions will need separate virtual networks, one per region, and they also need to be connected using the VNET-to-VNET connectivity. There is a cost associated with VNET-to-VNET connectivity for both inbound and outbound traffic. However for a network without any VNET-TO-VNET connectivity, there are no charges for inbound (ingress) data—only charges are associated with outbound data. Also, there is no charge for data transferred within virtual network.
- **Dedicated DNS:** Virtual networks by default use Azure provided DNS to resolve names within a virtual network and it also allows name resolution on the internet. If the application wants a dedicated name resolution service or wants to connect to the on-premise data centers, it should provide its own DNS server and they should be configured within the virtual network for successful name resolution.
- **Network performance:** Every virtual network has a defined throughput, and it is wise to create a separate virtual network to have a consistent performance instead of putting all network related resources on a single network. This might clog the network.

- **Number of virtual networks:** The number of virtual networks is affected by the number of regions, bandwidth usage by services, cross-region connectivity, and security.
- **Number of subnets in each virtual network:** Subnets provide isolation within a virtual network. It is also a security boundary. Network security groups can be associated with subnets thereby restricting or allowing specific access to IP addresses and ports. Application components having separate security and accessibility requirements should be placed within separate subnets.
- **IP ranges for networks and subnets:** Each subnet has an IP range. The IP range should not be large enough to be underutilized them while other subnets are getting suffocated because of lack of IP addresses. It should have just enough IP addresses to meet current and future requirements. This should be considered after understanding the future IP address needs of the deployment.

Planning should be done for IP addressing and ranges for Azure networks, subnets and on-premise data centers. There should not be an overlap for seamless connectivity and accessibility.

- **Monitoring:** Monitoring is an important architectural facet and is a must to be included in the overall deployment architecture. **Azure Network Watcher** provides logging and diagnostic capabilities with insights into network performance and health. It provides capabilities to watch both inbound or outbound packets to and from virtual machines, whether they are allowed or denied, next hop flow for validating the configuration of user-defined routes, an audit of network security, capturing packets along with subscription limits. It also provides diagnostic logs for all the network resources in a resource group.

Network performance can be monitored through log analytics. The network performance monitor management solution provides network monitoring capability. It monitors the health, availability, and reachability of networks. It is also used to monitor connectivity between public cloud and on-premises and subnets hosting various tiers of a multi-tiered application.

- **Security considerations:** Virtual networks are one of the first components that is accessed by any resource on Azure. Security plays an important role to allow or deny access to the resource. **Network security groups (NSG)** are the primary means of enabling security for virtual networks. They can be attached to virtual network subnets and every inbound and outbound flow is constrained, filtered, and allowed to them.

User-defined routing (UDR) and IP forwarding also helps in filtering and routing requests to resources on Azure.



You can read more about UDR and forced tunneling at <https://docs.microsoft.com/en-us/azure/virtual-network/virtual-networks-udr-overview>.

Resources can also be secured and protected by deploying network appliances such as barracuda, F5, and other third-party components.



You can read more about them at <https://azure.microsoft.com/en-in/solutions/network-appliances/>.

- **Deployment:** Virtual networks should be deployed in its own dedicated resource groups. Network administrators should have owner permission on this resource group, while developers or team members should have contributor permissions for allowing them to create another Azure resource in other resource groups consuming services from the virtual network.

It is also good practice to deploy resources with static IP addresses in a dedicated subnet, while dynamic IP address related resources can be on another subnet.

Policies should be created so that only network administrators can delete the virtual network, and it should also be tagged for billing purposes.

- **Connectivity:** Resources in a region on a virtual network can talk seamlessly. Even resources on other subnets within a virtual network can talk to each other without any explicit configuration. Resources in multiple regions cannot use the same virtual network. The boundary of a virtual network is within a region. To make resources available across regions, communicate with each dedicated gateway, which is required on each end to facilitate conversation. The networks on each end are connected to other networks through these gateways.

However, resources on Azure also at times have connectivity needs to on-premise data centers. Azure virtual networks can connect to on-premise data centers using VPN technology and ExpressRoute. In fact, one virtual network is capable of connecting to multiple on-premise data centers and other Azure regions in parallel. As a best practice, each of these connections should be in their dedicated subnets within a virtual network.

Benefits of Virtual Networks

Virtual networks are a must for deploying any meaningful IaaS solution. Virtual machines cannot be provisioned without virtual networks. Apart from being almost a mandatory component in solutions, it brings great architectural benefits and the most important among them are the following:

- **Isolation:** Most application components have separate security and bandwidth requirements and have different life cycle management. Virtual networks help in creating isolated pockets for these components that can be managed independently of other components with the help of virtual networks and subnets.
- **Security:** Filtering and tracking who is accessing resources is an important feature provided by virtual networks. They can stop access to a malicious IP address and port.
- **Extensibility:** Virtual networks act such as a private LAN on the cloud. It can also be extended into WAM by connecting other virtual networks across the globe and can be extended as an extension to on-premise data centers.

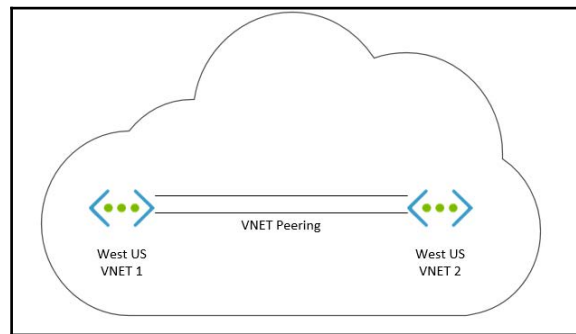
Virtual Network Design

In this section, we will consider some of the popular design and usage of virtual networks.

There can be multiple usages of virtual networks. A gateway can be deployed at each virtual network endpoint to enable security and transmit packets with integrity and confidentiality. A gateway is a must when connecting to on-premise networks, however, it is optional when using virtual network peering.

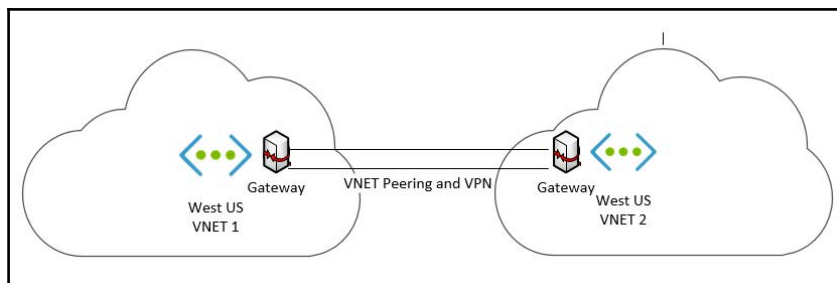
Connecting to Resources within the Same Region and Subscription

Multiple virtual networks within the same region and subscription can be connected to each other. With the help of virtual network peering, both networks can be connected and would use the Azure private network backbone for transmitting packets among each other. Virtual machines and services on these networks can talk to each other, subject to network traffic constraints.



Connecting to Resources within the Same Region in Another Subscription

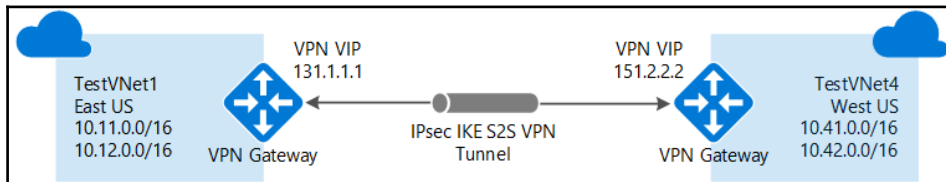
This usage is very much like the previous one, however, the additional complexity is that another network is on another subscription. If both the subscriptions belong to the same tenant, then they can be connected using virtual network peering as discussed before. It would also use Azure's private network backbone for transmitting packets among each other.



Gateways can be used along with peering

Connecting to Resources in Different Regions in Another Subscription

In this scenario, the traffic will pass through the public network and the virtual network peering should be deployed along with gateways to encrypt the traffic using VPN.

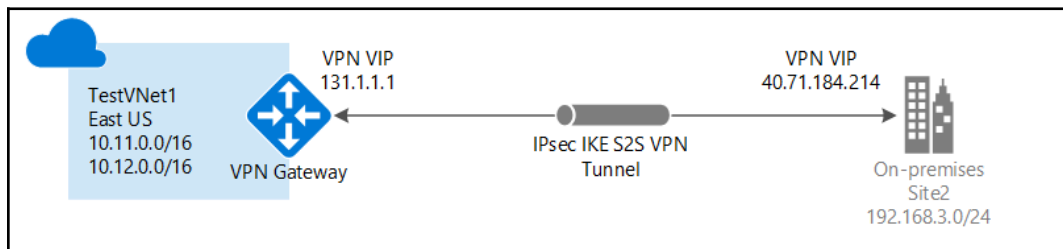


Connecting to On-Premise Data Centers

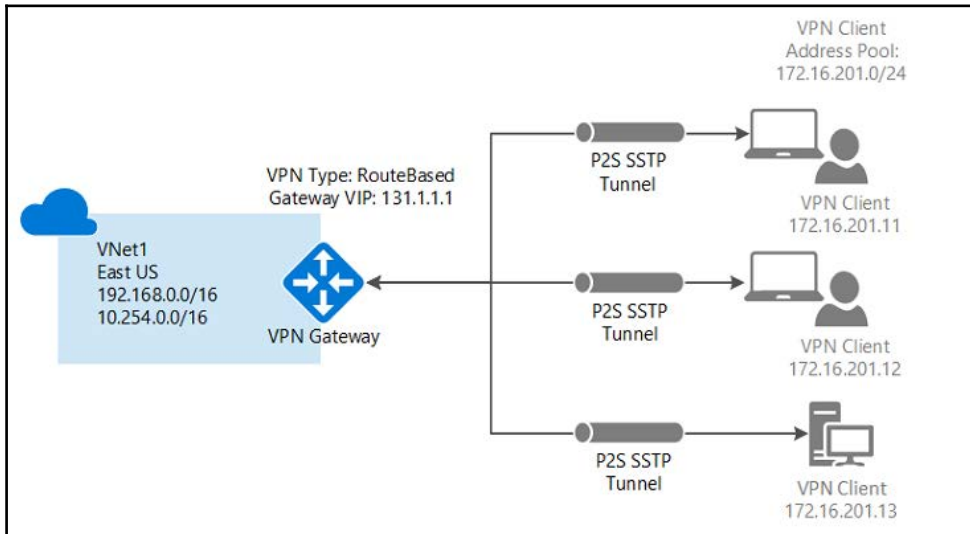
Virtual networks can be connected to on-premise data centers such that both Azure and an on-premise data center becomes a single **Wide Area Network (WAN)**.

Connecting on-premise network needs deployment on gateways and virtual private networks on both sides of the network. There are three different technologies available for this purpose:

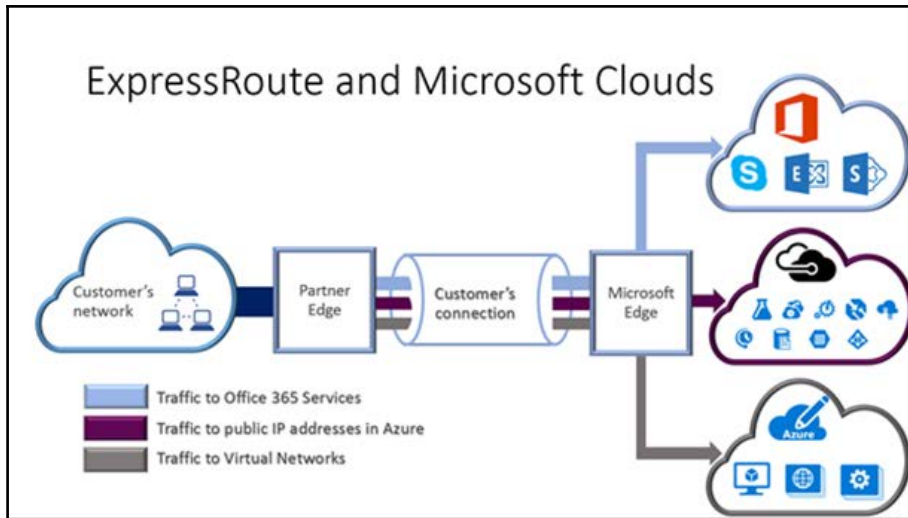
- **Site to site VPN:** This should be used when both Azure network and on-premise should be connected to form a WAN where any resource on both networks can access any other resource on them irrespective of Azure or an on-premise data center. VPN gateways are required to be available on both sides of networks for security reasons. Also, Azure gateways should be deployed on their own subnets on the virtual network connecting to on-premise data centers. Public IP addresses must be assigned to on-premise gateways for Azure to connect to it over the public network.



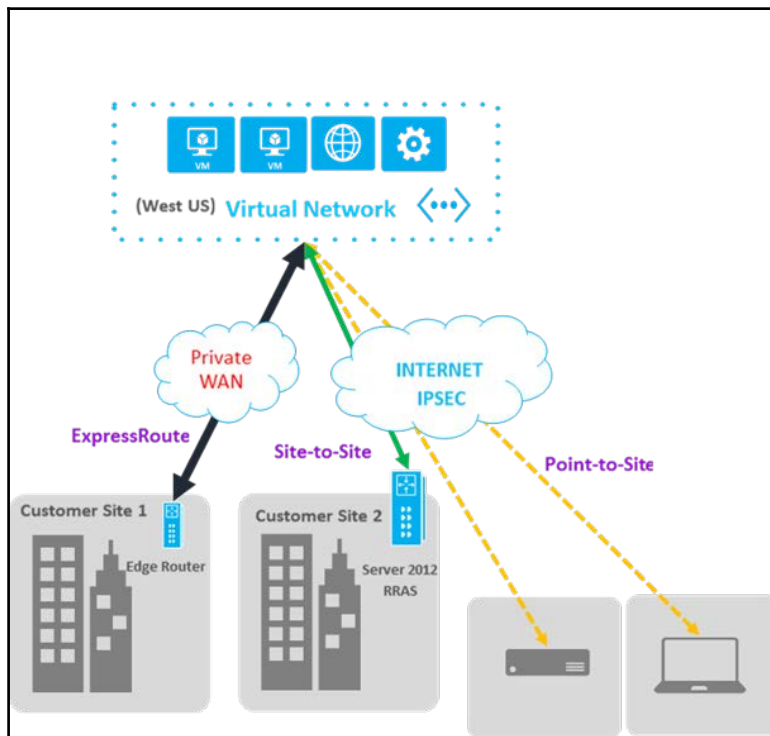
- **Point to site VPN:** This is similar to site-to-site VPN connectivity, however, there is a single server or computer attached to the on-premise data center. It should be used when there are very few users or clients that would connect to Azure securely from remote locations. Also, there is no need for public IP and gateway on the on-premise side in this case.



- **ExpressRoute:** Both site-to-site and point-to-site VPN work using the public internet. They encrypt the traffic between the network using VPN and certificates technology. However, there are applications that are deployed in hybrid mode. Some of its resources are hosted on Azure and others on on-premise data center. Even though resources are hosted on Azure, these resources should not use public internet for connectivity to on-premise data center. Azure ExpressRoute is the best solution for them, although a costly option compared to Site to site and point to site VPN connectivity. It is highly secure and reliable connectivity providing much greater speed and reduced latency compared to other VPN technologies. This is because the traffic never uses public internet but rather used dedicated connections with service providers. Azure ExpressRoute helps in extending on-premises networks into Azure over a dedicated private connection facilitated by a connectivity provider



The following figure shows all three types of hybrid networks:



It is a good practice for virtual networks to have separate subnets for each logical component having separate deployments from security and isolation perspectives.

Storage

Azure provides durable, highly available, and scalable storage solutions through storage services.

Storage is used for persisting data for long-term needs. Azure storage is available through the internet from almost every programming language.

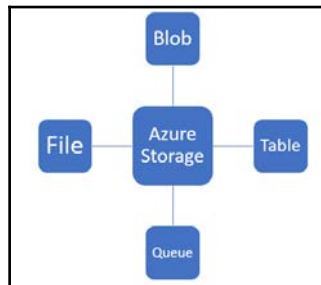
Storage Categories

Storage has two categories of storage accounts:

- A standard storage performance tier that allows you to store tables, queues, files, blobs, and Azure virtual machine disks.
- A premium storage performance tier supporting Azure virtual machine disks at the time of writing. Premium storage provides higher performance and IOPS compared to standard general storage. Premium storage is currently available as data disks for virtual machines are backed up by SSD's.

Storage Types

Azure provides four types of general storage services as shown in the following diagram:



- **Blob storage:** This type of storage is best suitable for unstructured data such as documents, images, and other kinds of files. Blob storage can be hot or cold. Hot means that content within the blob storage is accessed frequently compared to the cold tier.
- **Table storage:** It is a NoSQL key-attribute data store. It should be used for structured data. The data is stored as entities.
- **Queue storage:** It provides reliable message storage.
- **File storage:** This is shared storage based on SMB protocol. It is typically used for storing and sharing files. This also stores unstructured data, but the main distinction is that it is sharable on the SMB protocol.

These four storage types cater to different architectural requirements and cover almost all types of data storage facilities.

Storage Features

Azure storage is elastic. It means that you can store as little as a few megabytes to as large as petabytes of data. You do not need to pre-block the capacity and it will grow and shrink automatically. Consumers just need to pay for the actual usage of storage.

Azure storage is secure. It can only be accessed using the SSL protocol. Moreover, access should be authenticated. Azure storage provides the facility to generate an account level **Secure Access Signature (SAS)** token that can be used by storage clients to authenticate themselves. It is also possible to generate individual service level SAS tokens for blobs, queues, table, and files. Data stored in Azure storage can be encrypted. This is known as secure data at rest. Azure disk encryption is used to encrypt the OS and data disks in IaaS virtual machines. **Client-side Encryption (CSE)** and **Storage Service Encryption (SSE)**--are both used to encrypt data in the Azure storage. SSE is a storage account-wide setting that ensures that data is encrypted while writing data to storage and is decrypted while reading by storage engine. This ensures that no application changes are required to enable SSE. In CSE, client applications can use storage SDK to encrypt data before sending and writing to Azure storage. The client application can later decrypt it while reading. This provides both data in transit and data at rest security. CSE is dependent on secrets from the Azure key vault. There is another service Azure disk encryption used to encrypt the OS and data disks in IaaS virtual machines.

Azure storage is highly available and durable. What this means is that Azure always maintains multiple copies of Azure accounts. The location and number of copies depend on the replication configuration. Azure provides four replication settings. These settings have implications on both cost as well as availability during disaster. Locally redundant storage is cheapest and provides least availability compared to others while Read access geo-redundant is costliest and provides high availability.

- **Locally redundant storage: Locally redundant storage (LRS)** replicates and maintains three copies of storage within same data center of a region. It means data is highly available within a data center only. Storage will be lost or unavailable if for any reason this data center goes down. It follows the synchronous pattern for write operations meaning write to all replicas is regarded as a success. A write request is successful only after it is written to all three replicas. However, Azure ensures fault tolerance from storage disk and rack perspective. It does so by placing the three replicas in different fault and upgrade domains. Fault and upgrade domains are discussed in detail in the next chapter.
- **Zone-redundant storage: Zone-redundant storage (ZRS)** is costlier than locally redundant storage because apart from three copies in the same data center, it also stores data into another data center as well within a region. Since multiple data center is involved, to maintain performance and latency SLA writes to all storages happens asynchronously.
- **Geo-redundant storage: Geo-redundant storage (GRS)** provides better durability and availability compared to zone redundancy by replicating storage to another region apart from local redundancy. This is also one of the costliest options but provides disaster recovery at the region level.
- **Read-access geo-redundant storage: Read-access geo-redundant storage (RAGRS)** is similar to Geo-redundancy but it additionally provides read-only access to replicas.

Architectural Considerations for Storage Accounts

Storage account should be provisioned within the same region as that of other application components. This would help in using the same data center network backbone without incurring any network charges.

Each of the Azure storage services has scalability targets for capacity (GB), transaction rate, and bandwidth. A general storage account allows 500 TB of data to be stored. If there is a need to store more than 500 TB of data, then either multiple storage accounts should be created, or premium storage should be used. General storage performs max at 20000 IOPS or 60 MB of data per second. Any requirement for higher IOPS or data managed per second will be throttled. If this is not enough for applications from the performance perspective, either premium storage or multiple storage accounts should be used.

The size of a virtual machine determines the size and capacity of data disks available. While higher sized virtual machines have data disks with higher IOPS capacity, the max capacity will still be limited to 20000 IOPS and 60 MB per second. It is to be noted that these are maximum numbers and so generally lower levels should be taken into consideration when finalizing storage architecture.



For more information about bandwidth and storage performance details, check out the link <https://docs.microsoft.com/en-us/azure/storage/common/storage-performance-checklist>.

Azure storage accounts should be enabled for authentication using SAS tokens. They should not be allowed for anonymous access. Moreover, for blob storage, different containers should be created having separate SAS tokens generated based on different types and categories of clients accessing those containers. These SAS tokens should be periodically regenerated to ensure that these keys cannot be guessed or cracked by anyone.

Generally, blobs fetched for blob storage accounts should be cached and it can be determined that data in a cache is stale by comparing the blob's last modified property to re-fetch the latest blob.

Azure storage account provides concurrency features to ensure that the same file and data is not modified simultaneously by multiple users. It provides the following:

- **Optimistic concurrency:** It allows multiple users to modify data simultaneously, but while writing checks if the file or data has changed. If it has, it informs the users to re-fetch the data and performs the update again. This is the default concurrency for table service.
- **Pessimistic concurrency:** When an application tries to update a file, it places a lock that explicitly denies any updates to it by other users. This is the default concurrency for file services when accessed using SMB protocol.
- **Last writer wins:** In this, the updates are not constrained, and the last user updates the file irrespective of what was read initially. This is the default concurrency for both queue, blob, and file (when accessed using REST) services.

Design Patterns

Design patterns are proven solutions to known design problems. They are reusable solutions that can be applied to problems. They are not reusable codes or designs that can be incorporated as is within a solution. They are documented descriptions and guidance to solve a problem. The problem might manifest itself in a different context and design patterns can help solve them. Azure provides numerous services with each service providing a specific feature and capability. Using these services is straightforward, but creating solutions weaving multiple services can be a challenge. Moreover, achieving high availability, super-scalability, reliability, performance, and security for a solution is not a menial task. Azure design patterns provide ready solutions to these problems and provide solutions that can be tailor-made to individual solutions. They help in making highly available, scalable, reliable, secure, and performance-centric solutions on Azure. Although there are many patterns and some of the patterns are dealt with in detail in the subsequent chapters, some of the messaging, performance, and scalability patterns are mentioned in this chapter. Also, links are provided for the detailed description of these patterns. These design patterns deserve a complete book by themselves. They have been mentioned here to make the readers aware of their existence and provide links for deeper information.

Messaging Patterns

Messaging patterns help in connecting services in a loosely coupled manner. What it means is that services never talk to each other directly. Instead, a service generates and sends a message to a broker (generally a queue) and any other service that is interested in that message can pick it and process it. There is no direct communication between the sender and receiver service. This decouples not only makes services and overall application more reliable, but also more robust and fault tolerant. Receivers can receive and read messages at their own capable speed.

Messaging helps in creating asynchronous patterns. Messaging involves sending messages from one entity to another. These messages are created and forwarded by a sender, stored at durable storage, and finally consumed by recipients.

The top architectural concerns addressed by messaging are the following:

- **Durability:** Messages are stored in durable storage and the application can read it later after they come up in case of prior failure
- **Reliability:** Messages help in implementing reliability with design since messages are not lost because they are persisted in disk
- **Availability of messages:** The messages are available for consumption by applications after restoration of connectivity and prior downtime

Azure provides service bus queues and topics to implement messaging patterns within applications. Storage queue can also be used for the same purpose.

Choosing between Azure service bus queue and storage queue is to decide between how long the message should be stored, size of the message, latency, and cost.

Azure service bus provides support for 256 KB message size, while storage queue provides support for 64 KB message size. Azure service bus can store messages for an unlimited period, while storage queue can store messages for seven days.

The cost and latency are higher in case of service bus queues.

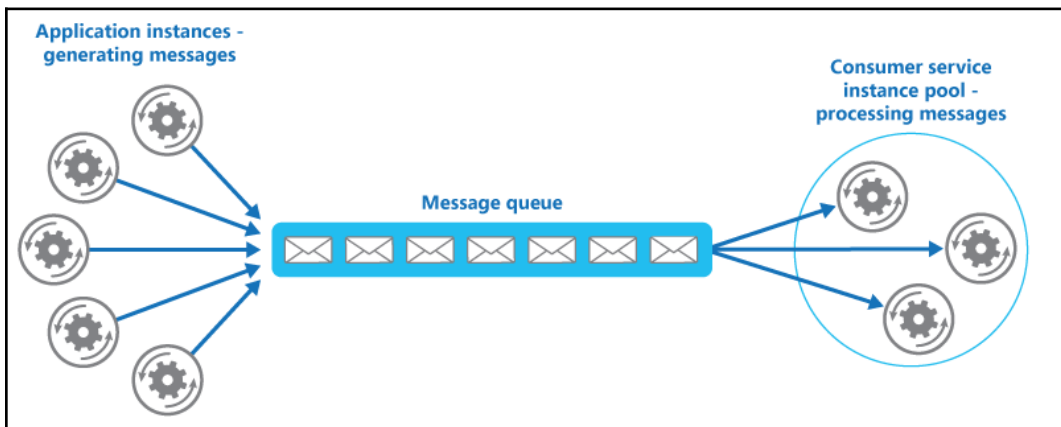
Depending on application requirements and needs the preceding factors should be employed before deciding the best queue.

Competing Consumers

A single consumer of messages works in a synchronous manner unless the application itself implements the logic of reading messages asynchronously. Competing consumers pattern helps in implementing a solution in which multiple consumers are ready to process the incoming message and they compete to process it. This helps in architecting solutions that are highly available and scalable. It is scalable because with multiple consumers it is possible to process a higher number of messages in the smaller period. It is highly available because there could be at least one or more consumers to process the messages even if some consumers crash.

This pattern should be used when each message is independent of other messages. The messages by themselves contain complete information for a consumer to complete a task. This pattern should not be used if there is any dependency among messages. The consumers should be able to complete the tasks in isolation. Also, this pattern is applicable if there is a variable demand for services. Additional consumers can be added or removed based on demand.

A message queue is required for implementing competing consumers pattern. In this pattern, patterns from multiple sources pass through a single queue that is connected to multiple consumers on another end. These consumers should delete the message after reading it so that it is not re-processed again.





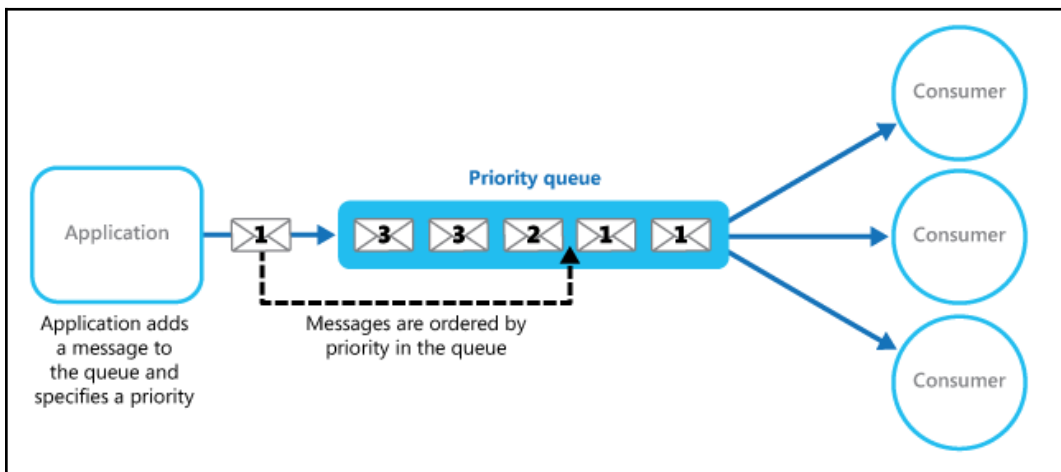
More about this pattern can be read at <https://docs.microsoft.com/en-us/azure/architecture/patterns/competing-consumers>.

Priority Queue

There is often a need when messages with higher importance should be given priority in processing compared to general messages with lower priority. This pattern is important for applications that provide different **service level agreements (SLA)** to consumers, which provide services based on different plans and subscriptions.

Queues follow first-in, first-out patterns. Messages are processed in sequence. However, with the help of this pattern, it is possible to fast-track the processing of certain messages due to their higher priority. There are multiple ways to implement it. If the queue provides the capability of assigning priority and based on priority re-order messages, then even a single queue is enough to implement this pattern.

However, if the queue does not have the capability to re-order messages, then separate queues can be created for different priorities and each queue can have separate consumers associated with it.



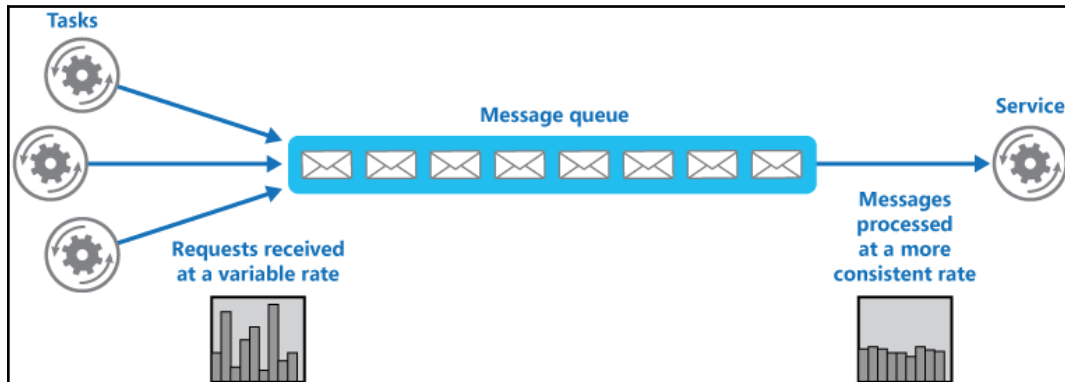
In fact, this pattern can reuse competing consumer patterns if needed to fast-track processing of messages from each queue using multiple consumers.



More about this pattern can be read at <https://docs.microsoft.com/en-us/azure/architecture/patterns/priority-queue>.

Queue-Based Load Leveling Pattern

There are times when the load on an application cannot be determined at all times. Although there is consistent and predictable demand for application for most of the times, there are times when this load can go very high leading to failure of service or providing reduced performance or non-availability. Queue-based load leveling pattern can help during such scenarios. In this pattern, a queue is maintained and all request for the service is stored as messages within this queue. The queue acts as a highly available and durable temporary storage that then sends messages to service at a controlled speed thereby reducing disruption at the service end. The same has been shown in the next image. There are multiple tasks sending messages to message queue. The queue stores the messages and ensures that the service gets these messages at a speed consistent with the resources available at the service end.



This pattern ensures that there is no unnecessary scaling up and out of resources by provisioning more instances to meet higher service demand. It has a direct impact on cost as well due to predictable usage and instances of resources.

High availability and better scalability are other advantages derived by implementing this pattern.

Performance and Scalability patterns

Performance and scalability go hand in hand. Performance refers to the responsiveness of a system to execute any action within a given time interval, while scalability is the ability of a system to handle increases in load without impact on performance. In this section, a couple of design patterns related to performance and scalability are described.

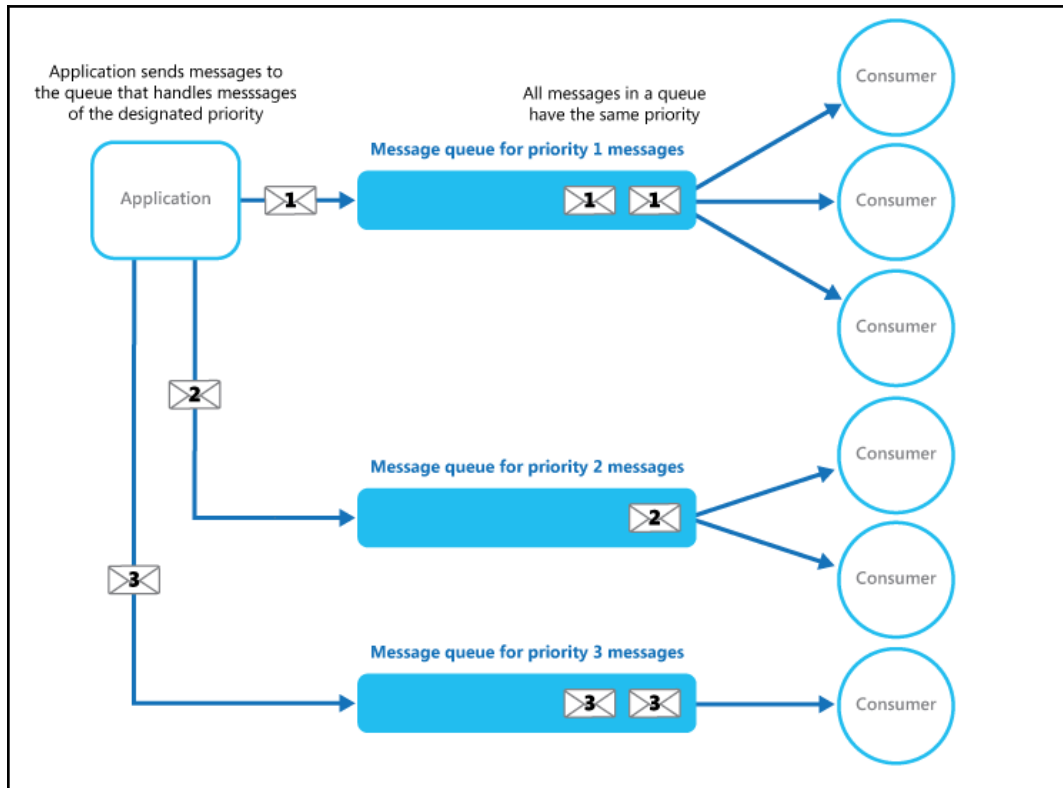
The Command and Query Responsibility Segregation (CQRS) Pattern

CQRS is a generic pattern that has applicability in any scenario that has data stored in a data store and it should be accessed in a way to increase the overall performance and responsiveness of the application.

Data operations can broadly be classified into read and write operations. There are multiple ways to read and write to the data store and there is often a data access layer and component responsible for carrying out these operations. This data access component has information about connecting the data store and performs both read and write operations. Performing both operations from within a single interface can be challenging from performance perspective especially if the data is large and the read-write ratio is skewed.

Command and Query Responsibility Segregation (CQRS) is a pattern that helps in implementing read and write operations using different interfaces. It means the components implementing the read operations are separate from write operations and can be individually deployed on a separate instance. This helps in providing dedicated resource capacity to them. This pattern also helps in scenarios where execution time for read and write operations is significantly large and consume more resources

CQRS not only helps in improving the performance of the application, but it also helps in design and implementation between multiple teams. Due to its nature of using separate models, CQRS pattern is not a great fit if using model and scaffolding generation tools.



More about this pattern can be read at <https://docs.microsoft.com/en-us/azure/architecture/patterns/cqrs>.

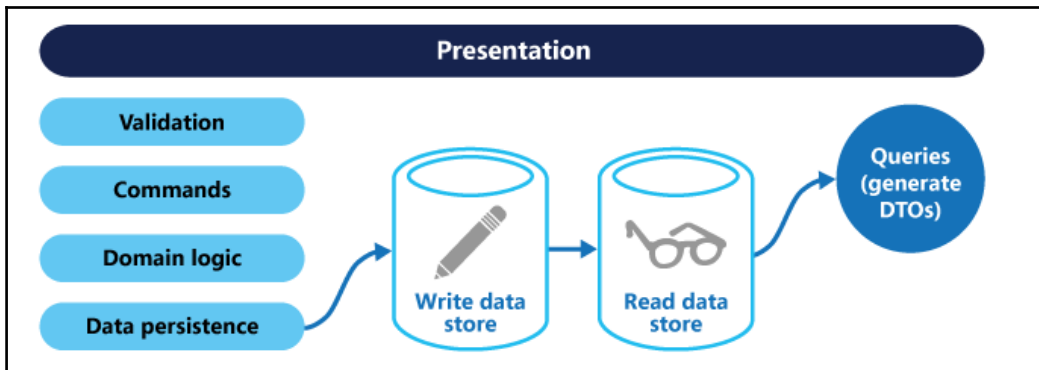
Throttling Pattern

At times, there are applications that have very stringent--SLA requirements from performance and scalability perspective irrespective of a number of users consuming the service. In such circumstances, it is important to implement throttling patterns because they help in limiting the number of requests allowed to be executed. The load on applications cannot be predicted accurately for all circumstances. When the load on application spikes, throttling helps in reducing pressure on the servers and services by controlling the resource consumption.

This pattern should be used when meeting SLA is a priority for applications, to prevent some users to consume more resources than allocated, to optimize spikes and burst in demand, and to cost optimize the resource consumption. These are valid scenarios for applications built to be deployed on the cloud.

There can be multiple strategies used for throttling an application. The throttling strategy can reject newer requests once the threshold is crossed, or it can let the user know that the request is in the queue and will get an opportunity to be executed once the number of requests gets reduced.

The following diagram illustrates implementing throttling in a multi-tenant system where each tenant is allocated a fixed resource usage limit. Once they cross this limit, any additional demand for resources is constrained, thereby maintaining enough resources for other tenants.



More about this pattern can be read at <https://docs.microsoft.com/en-us/azure/architecture/patterns/throttling>.

Other Patterns

Some of the other important patterns are described in this section.

Retry Pattern

The retry pattern is an extremely important pattern to make applications and services more resilient to transient failures. Think about a situation that you are trying to connect and use the service, and the service is not available due to any reason. If the service is going to come up in short duration, it makes sense to keep retrying for a successful connection. This will make the application more robust, fault tolerant, and bring instability. In Azure, most of the components are running on the internet and that internet connection can produce transient faults intermittently. Because of these failures that rectify within seconds, an application should not be allowed to crash. The application should be designed in a manner that it can re-try to use the service repeatedly during failures. It should also stop retrying if even after a certain number of retries, the service is not available.

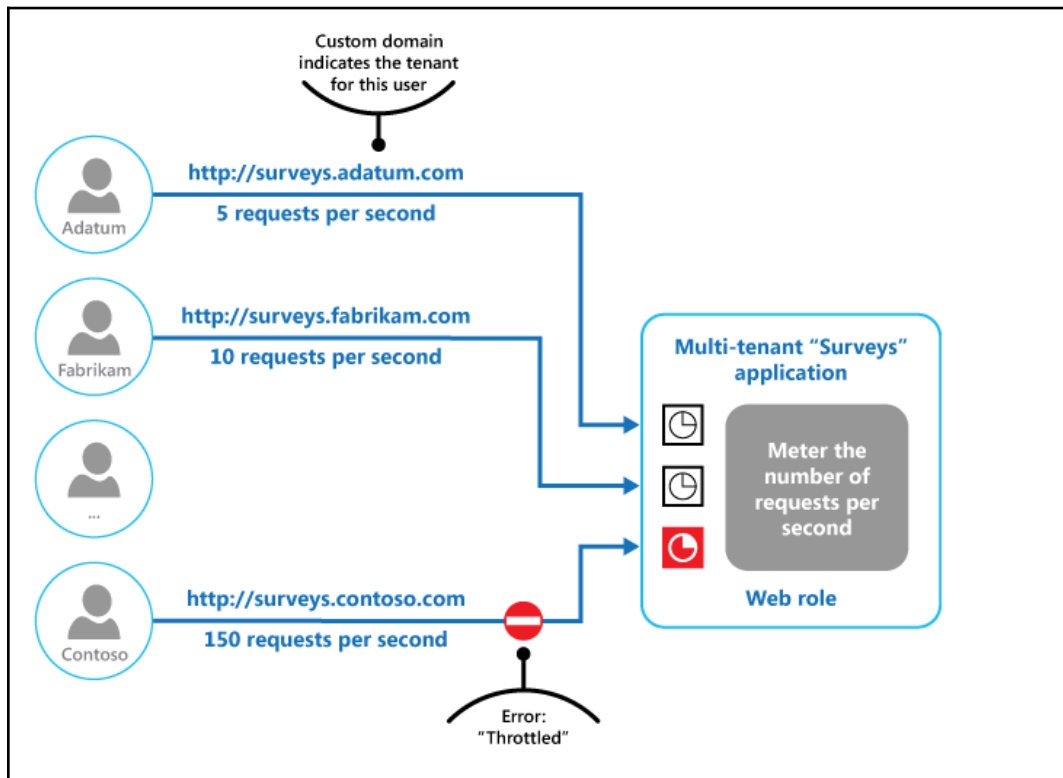
This pattern should be implemented when an application could experience transient faults as it interacts with a remote server or accesses a remote resource. These faults are expected to be short-lived, and repeating a request that has previously failed could succeed on a subsequent attempt.

The retry pattern can adopt different retry strategies based on the nature of errors and application:

- **Retry a fixed number of times:** It denotes that the application will try to communicate to the service a fixed number of times before it can determine a failure and raise an exception. For example, retry three times to connect to another service. If it is successful in connecting within these three tries, the entire operation is successful, or after the expiry of this count, it will raise an exception.
- **Retry based on schedule:** It denotes that the application will try to communicate to the service repeatedly for a fixed number of seconds or minutes and wait for a fixed number of seconds or minutes before retrying. For example, retrying every three seconds for 60 seconds to connect to another service. If it is successful in connecting within these tries, the entire operation is successful, or after the expiry of 60 seconds, it will raise an exception.

- **Sliding and delaying the retry:** It denotes that the application will try to communicate to the service repeatedly based on schedule and keeps adding an incremental delay in subsequent tries. For example, retrying for a total of 60 seconds where the first retry happens after one second, the second try happens after two seconds from the last retry, the third one after four seconds from the last try, and so on. This helps in reducing the overall number of retries.

The retry pattern is depicted by means of a picture where the first request gets HTTP 500 as a response message, the second retry again gets HTTP 500 as a response, and finally, the request is successful and gets HTTP 200 as a response message.



More about this pattern can be read at <https://docs.microsoft.com/en-us/azure/architecture/patterns/retry>.

Circuit Breaker Pattern

This is an extremely useful pattern. Think again about a situation that you are trying to connect to and use the service, and the service is not available due to any reason. If the service is not going to come up in a short duration, it is no use to keep retrying for connection. Moreover, keeping other resources occupied while retrying wastes a lot of resources that could potentially be used elsewhere.

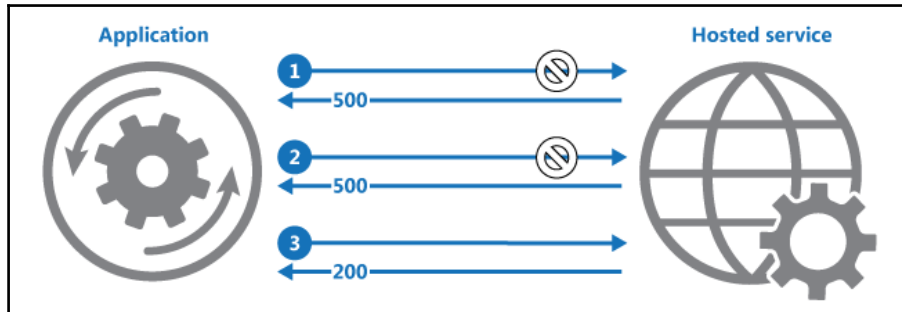
The circuit breaker pattern helps eliminate such wastage of resources. It can prevent applications from repeatedly trying to connect and use a service that is not available. It also helps applications to detect whether the service is up and running again and eventually allows applications to connect to it.

To implement the circuit breaker pattern, all requests to the service should pass through another service. This service that acts as a proxy to the original service. The purpose of this proxy service is to maintain a state machine and acts as a gateway to the original service. There are three states that it maintains. There can be more states that could be included based on application requirements.

The minimal states needed to implement this pattern are the following:

- **Open:** It denotes that the service is down, and the application is shown an exception immediately instead of allowing it to retry or wait for the timeout. When the service is alive again, the state is transitioned to Half-Open state.
- **Closed:** This state denotes that the service is healthy and that the application can go ahead and connect to it. Generally, a counter is maintained for maintaining the number of failures before it can transition to the Open state.
- **Half-Open:** At some point, as the service is up and running, this state allows a limited number of requests to pass through it. This state is a litmus test checking if the request that passes through it is successful or not. If it is successful, the state is transitioned from Half-Open to Closed. This state can also implement a counter to allow a certain number of requests to be successful before it can transition to the Closed state.

The preceding mentioned states and their transition is shown here:



More about this pattern can be read at <https://docs.microsoft.com/en-us/azure/architecture/patterns/circuit-breaker>.

Summary

There are numerous services available on Azure and most can be combined to create real solutions. It was difficult to write about design patterns in 20 pages where there are hundreds of services and multiple ways to bring them together. This chapter explained the three most important services from Azure--regions, storage, and networks. They almost form the backbone of every solution deployed on any cloud. This chapter provided details about these services and how their configuration and provisioning can affect design decisions. The important considerations for both storage and networks were detailed in this chapter. Both networks and storage provide lots of choices and it is important to choose an appropriate configuration based on requirement. Finally, some of the important design patterns related to messaging such as competing consumers, priority queues, and load leveling were described. Patterns related to CQRS and throttling were illustrated and other patterns such as retry, and circuit breaker were discussed.

Next chapter will be an interesting chapter discussing High Availability options available in Azure for deploying workloads.

11

Designing High Availability

Running applications and systems that are available to users for consumption whenever they need is one of the topmost priorities for CIO's. They want their applications to be operational, functional, and continue to be available to their customers even when some untoward event happens, and this is the theme for this chapter—Designing High Availability. *Keeping the lights on* is the common metaphor used for High Availability. Achieving High Availability for applications is not an easy task and organizations have to spend considerable time, energy, resources, and money to achieve this. And even when using them, there are still chances and risks that their implementation does not produce the desired results. Azure provides a lot of High Availability features for **virtual machines (VMs)** and PaaS services. In this chapter, we will go through the architectural and design features provided by Azure for ensuring High Availability for applications and services.

In this chapter, we'll cover the following topics:

- Azure availability sets:
 - Fault domain
 - Upgrade domain
- Azure load balancer
- Azure application gateways

High Availability

High Availability is one of the major architectural concerns for any architect. It forms one of the core non-functional technical requirements for any serious service and its deployment. High Availability refers to the feature of a service or application that keeps it operational on a continuous basis, meeting or surpassing its promised defined **Service Level Agreement (SLA)**. Users are promised certain SLA on the availability of a service. The service should be available for consumption based on its SLA. For example, an SLA can have 99% availability for an application for the entire year (assuming 365 days). It means it should be available for consumption by users for 361.35 days. If it goes less than this, there is a breach of SLA. Most mission-critical applications define their High Availability SLA with five nines for a year. It means the application should be up, running, and available throughout the year, but it can be down and not available only for 5.2 hours.

It is important to note here that High Availability is defined in terms of time that is yearly, monthly, weekly, and so on and it could even be a combination of these.

A service or application is made up of multiple components and these components are deployed on separate tiers and layers. Moreover, it is deployed on an operating system and hosted on a physical or virtual machine. It consumes network and storage services for various purposes. It might even be dependent on external systems. For these services or applications to be highly available it is important that networks, storage, operating system, virtual or physical machines, each component of the application is architected and designed keeping SLA and High Availability in mind. There should be a definite application lifecycle process that should be used to ensure High Availability is baked from the start of application planning until its introduction to operations. It also involves architecting for redundancy. There should be redundant resources included in the overall application and deployment architecture to ensure that if one goes down, the other takes over and serves the requests of the customer.

SLA

SLA is defined on Wikipedia as:

"A service-level agreement is an agreement between two or more parties, where one is the customer, and the others are service providers. Particular aspects of the service - quality, availability, responsibilities - are agreed between the service provider and the service user. The most common component of SLA is that the services should be provided to the customer as agreed upon in the contract".

Factors Affecting High Availability

Planned maintenance, unplanned maintenance, and application deployment architecture are the major factors affecting High Availability of an application.

Planned Maintenance

Planned maintenance refers to the process of keeping the application and its surrounding ecosystem comprising of platforms, frameworks, software, operating system, and host and guest drivers up to date with the latest stable releases. It is important to patch software's, drivers, and operating systems with the latest updates as it helps in keeping the environment healthy from a security, performance, and future-ready perspective. Not upgrading an environment is not an option and is a fact of life. Even applications should be upgraded with enhanced functionality, bugs, and hotfixes. Every organization plans for environment and application upgrades and typically these involve shutting down and restarting the application and operating system. It might also involve starting of the physical host operating system that in turn will reboot all guest virtual machines running on top of it.

Unplanned Maintenance

As the name suggests, unplanned maintenance refers to maintenance that cannot be planned and is ad hoc in nature. These refer to hardware failures such as storage corruption, network or router failure, power loss, and a host of other failures due to hardware and software. Bugs in underlying platform bringing the application down are also part of unplanned maintenance.

Application Deployment Architecture

Application architecture plays a crucial role to ensure High Availability of an application. An application whose components are deployed on a single machine is not highly available. When the machine reboots, the application is not available to its users. Similarly, depending on a single instance of a resource can become a single point of failure from a High Availability point of view. Each component of application should be designed in a manner that it can be deployed on multiple machines and redundancy should not be a bottleneck. Some software provides features related to High Availability and are not dependent on host operating systems or other third-party tools. SQL server availability groups is an example of such features.

High Availability versus scalability

High Availability is different than scalability although both are serious architectural concerns. Scalability refers to flexibility and elasticity to add more or reduce resources to existing deployment to accommodate more users than normal without comprising on application performance. Scalability indirectly helps in making an application highly available. However, it does not mean scalability eventually leads to High Availability. High Availability is an architectural concern that is not dependent on the number of users, while scalability rules are determined by a number of users consuming the service. High Availability could be a requirement even if there are very few users. High Availability talks about services being present and operational as and when users demand its consumption. It is a function of consumption based on SLA. Scalability is the topic of the next chapter and it will be discussed in more detail.

High Availability versus Disaster Recovery

High Availability is again different from disaster recovery; however, the difference could be very subtle. High Availability is a function of the application being in a consumable state as and when the user asks for it. So, it is designed for operations that are before a disaster, while disaster recovery is a function that comes into the picture after a disaster. Disaster recovery refers to the architecture implementation through which services are up and running after a disaster, while High Availability takes care of availability before a disaster. Disaster recovery includes data backup, archived and dormant servers across continents, while High Availability consists of load balancers, distribution of load, active-passive, and active-active redundancy.

Azure High Availability

Achieving High Availability that meets high SLA requirements is a tough requirement. Azure provides lots of features that enable High Availability for applications from the host and guest operating system to applications using its PaaS. Architects can use these features to get High Availability in their applications using configuration instead of building these features by scratch or being dependent on third-party tools.

In this section, we will see features and capabilities provided by Azure to make applications highly available. Before we get into the architectural and configuration details, it is important to understand Azure High Availability related concepts.

Concepts

The fundamental constructs provided by Azure to attain High Availability is through:

- Availability sets
- Fault domain
- Update domain
- Availability Zones

Availability Sets

High Availability in Azure is primarily achieved through **redundancy**. Redundancy means that there is more than one resource instance, such that in the event of a failure for a resource, the other takes over. But just having more similar resources does not make them highly available. For example, having more than one virtual machine does not make these virtual machines highly available. Azure provides availability set resources to tag and brings these resources together to form a group. All virtual machines in the availability set become highly available because they are placed on separate physical racks in the Azure data center and each virtual machine is updated one at a time instead of all at the same time. Availability sets provide fault domain and updates domain to achieve this and this is discussed in the next section. In short, availability sets provides redundancy at data center level similar to locally redundant storage.

Fault Domain

When a virtual machine is provisioned and assigned to an availability set, it is assigned a fault domain. With **Azure Resource Manager (ARM)**, each availability set has two or three fault domains by default depending on Azure regions. Some provide two, while others provide three fault domains in an availability set. Fault domains are non-configurable by users. When multiple virtual machines are created, they are placed on separate fault domains. If more than five virtual machines are provisioned on an availability set, they are placed in a round-robin fashion on five fault domains. Fault domains are related to physical racks in the Azure data center. Fault domains provide High Availability from unplanned maintenance due to hardware, power, and network failure. Since a single virtual machine is only placed on a rack, other virtual machines continue running in case the rack in consideration snaps off.

Update Domain

Fault domain takes care of unplanned maintenance and update domain handles planned maintenance. Each virtual machine is also assigned an update domain. There can be as many as 20 update domains in a single availability set. Update domains are nonconfigurable by users. When multiple virtual machines are created, they are placed on separate update domains. If more than 20 virtual machines are provisioned on an availability set, they are placed in a round-robin fashion on these update domains. Update domains take care of planned maintenance.

Availability Zones

This is relatively a new concept introduced in Azure and very similar to zone redundancy we saw for storage accounts. Availability Zones provides high availability within a region by placing virtual machines instances on separate data centers within a region. Availability Zones are applicable to virtual machines, managed disks, virtual machine scale sets and load balancers. This was a gap in Azure for a long time and has been removed recently from computing high availability perspective. More information about Availability Zones is available at <https://docs.microsoft.com/en-us/azure/availability-zones/az-overview>.

Load Balancing

Load balancing, as the name suggests, refers to the process of balancing the load among virtual machines and applications. With one virtual machine, there is no need of load balancer because the entire load is on a single virtual machine and there is no other virtual machine to share the load. However, with multiple virtual machines containing the same application and service, it is possible to distribute the load among them through load balancing. Azure provides a couple of resources for enabling load balancing:

- **Load balancers:** Azure load balancer helps in architecting solutions with high availability. Within the TCP stack, it is a layer 4 transport level load balancer. It is a layer 4 load balancer that distributes incoming traffic among healthy instances of services defined in a load-balanced set. Level 4 load balancers work at transport level and have network level information such as IP address and port to decide the target for the incoming request. Load balancers are discussed in detail later in this chapter.

- **Application gateways:** Azure application gateways deliver High Availability to your applications. It is a layer 7 load balancer that distributes the incoming traffic among healthy instances of services. Level 7 load balancers can work at the application level and has application level information such as cookies, HTTP, HTTPS, and sessions for the incoming request. Application gateways are discussed in detail later in this chapter.

Virtual Machine High Availability

Virtual machines provide compute capabilities. It provides processing power and hosting for applications and services. If the application is deployed on a single virtual machine and that machine is down, then even the application is not available. If the application is composed of multiple tiers and each tier is deployed in its own single instance of a virtual machine, even a downtime in a single virtual machine can render the application non-available. Although Azure tries to make even single virtual machine deployments highly available by placing on different racks as soon as it can figure out, Azure does not ensure or guarantee any SLA on them. Azure provides SLA for those virtual machines that are grouped together in an availability set. It provides 5 nines (99.999%) SLA for the availability of virtual machines if they are part of an availability set and more than two virtual machines are on that availability set.

Computing High Availability

Applications demanding High Availability should be deployed on multiple virtual machines on the same availability set. If applications are composed of multiple tiers, then each tier should have a group of virtual machines on their dedicated availability set. In short, if there are three tiers of an application, there should be three availability sets and minimum six virtual machines (two in each availability set) to make the entire application highly available.

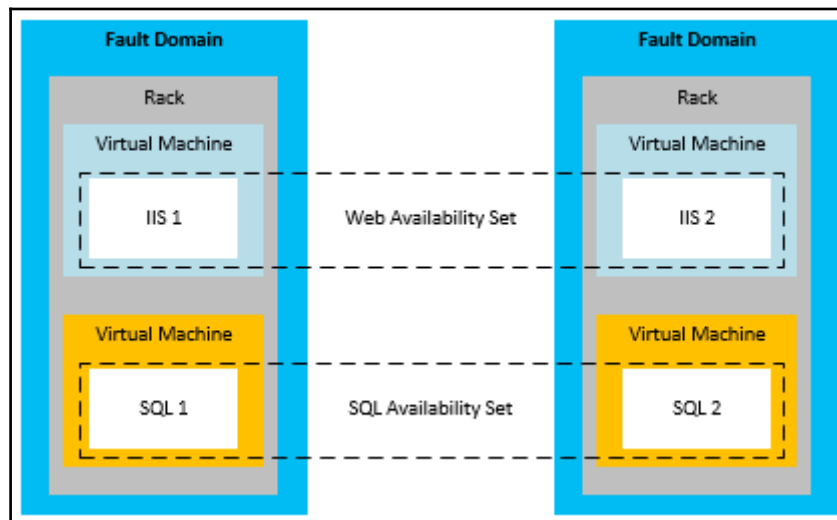
How does Azure provide SLA and High Availability to virtual machines in an Availability set with multiple virtual machines in each availability set? This is the question that might be coming to mind.

Here comes the use of concepts that we considered before: fault and update domain.

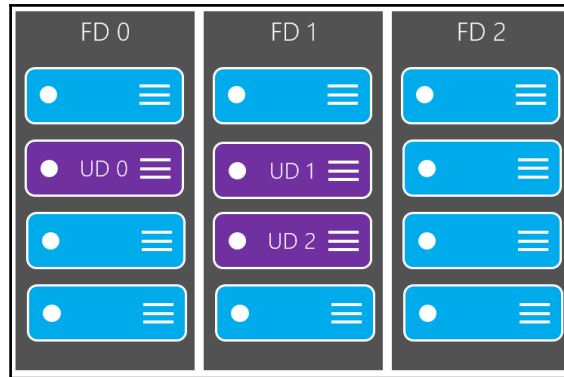
When Azure sees multiple virtual machines in an availability set, it places those virtual machines on a separate fault domain. In other words, these virtual machines are placed on separate physical racks instead of the same rack. This ensures that at least one virtual machine continues to be available even if there is a power, hardware, or rack failure. There are two or three fault domains in an availability set and depending on a number of virtual machines in an availability set, the VMs are placed on separate fault domains or repeated in a round robin fashion. This ensures that High Availability is not impacted because of failure of the rack.

Azure also places these VMs on a separate update domain. In other words, Azure tags these VMs internally in such a way that these virtual machines are patched and updated one after another such that any reboot in an update domain does not affect the availability of the application. This ensures that High Availability is not impacted because of the virtual machine and host maintenance.

With the placement of virtual machines in separate fault and update domains, Azure ensures that not all of them are down at the same time and are alive and available for serving requests even though they might be undergoing maintenance or facing physical downtime challenges.



The previous image shows four virtual machines (2 IIS and 2 SQL related). Both IIS and SQL virtual machines are part of their availability set. The IIS and SQL virtual machines are on separate fault domain and different racks in the data center. They would also be on separate upgrade domains.



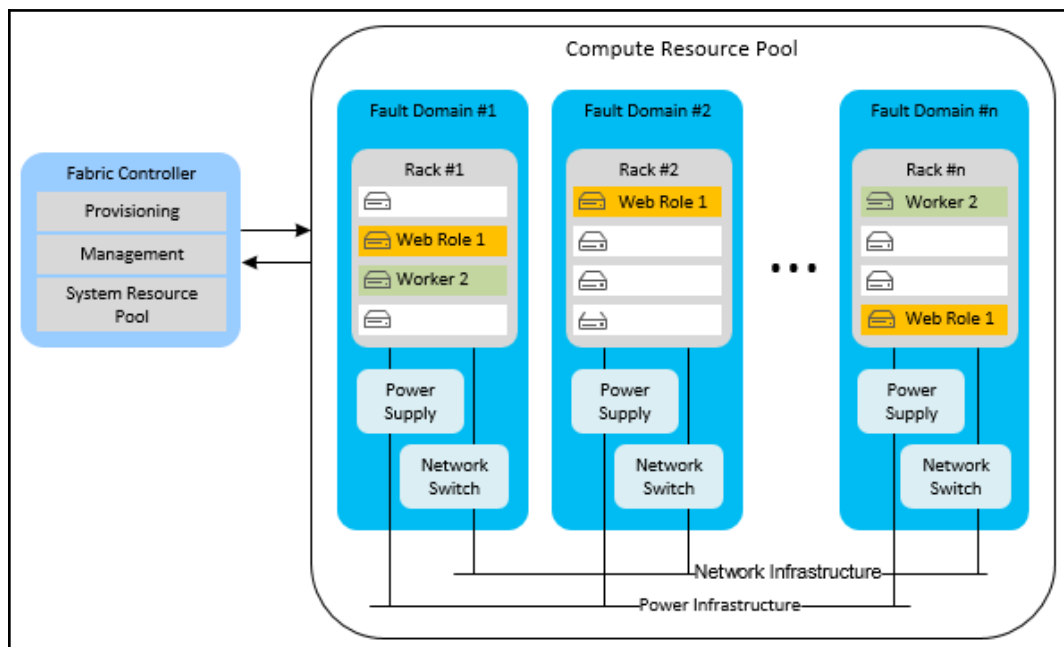
Storage High Availability

Virtual machines are backed up by storage accounts by storing their VHD files on them. While availability sets provide High Availability to compute instance, it does not ensure High Availability of VHD files for virtual machines stored in storage accounts. The VHD files for all VMs might be placed on the same storage cluster and any cluster failure can render all virtual machines non-available or less available than required. In short, it is not only computed services that need to be highly available, but even storage accounts storing VHD files should be placed on separate clusters such that in an event of failure, at least one or some virtual machines continue to be available both from a computer and storage perspective.

Azure provides managed disks as a concept that provides disk management facilities. Managed disks provide better reliability for availability sets by ensuring that the disks of VMs in an availability set are sufficiently isolated from each other to avoid single points of failure. It does this by automatically placing the disks in different storage clusters. If a storage cluster fails due to hardware or software failure, only the VM instances with disks on those stamps fail. Each virtual machine VHD in an availability set should be placed in a separate storage account. Although, virtual machines from different availability sets can be placed in a storage account.

PaaS High Availability

Azure provides app services and cloud services for hosting managed platforms. Services and applications can be deployed on top of them. They provide flexibility, elasticity, and economies to create and deploy applications. These platforms are managed by Azure and users do not interact with base infrastructure on which they are deployed. They bring in a higher level of abstraction compared to IaaS by letting developers concentrate on their business problem and using the platform to fast track their development and deployment process. This alleviates them to manage, operate, and monitor the base infrastructure. When an application is deployed in-app services or cloud services, Azure provisions virtual machines that are not visible to users. The applications are deployed on these virtual machines and Azure fabric controller is responsible for provisioning, managing, and monitoring them. The fabric controller monitors the status of the hardware and software of the host and guest machine instances. When it detects a failure, it maintains SLAs by automatically relocating the VM instances. When multiple cloud service role instances are deployed, Azure deploys these instances to different fault and update domains.



The previous diagram shows that PaaS services with multiple virtual machines instances deploy these web and worker roles on separate fault domains. Deploying on separate fault domains means deploying on separate racks within a data center. It also means that these services have separate network switches and power supply ensuring that even if one of the racks undergoes maintenance, there is disruption of power supply to the rack or failure of the network switch, there are other instances available to serve customer's requests.

Application High Availability

High Availability can be inbuilt within the software used for applications or it can be built from ground-up within applications. One example of the High Availability feature provided by software is SQL server always on availability groups. They help in keeping databases highly available.

Azure Services also have an inbuilt High Availability mechanism. In Azure SQL, data is replicated synchronously within the region. Active geo-replication allows up to four additional database copies in the same region or different regions. Azure storage has its own mechanism to make data available by replicating to multiple data centers and regions.

Load Balancing

Azure provides two constructs to provision load balancers. It provides a level-4 load balancer that works at the transport layer within TCP OSI stack and a level-7 load balancer that works at application and session level.

Although both application gateways and load balancer provide basic features of balancing the load, they serve different purposes. There are use cases in which the application gateway makes more sense to deploy compared to load balancer.

Application gateway provides the following features that are not available in the Azure load balancers:

- **Web application firewall:** This is an additional firewall on top of operating system firewall and has the capability to peek into incoming messages. This help in identifying and protecting from common web-based attacks such as SQL injection, cross-site scripting attacks, and session hijacks.

- **Cookie-based session affinity:** Load balancers distribute incoming traffic to services instances that are healthy and relatively free. A request can be served by any service instance. There are applications that need advance features in which all subsequent requests following the first request should be processed by same service instance. This is known as cookie-based session affinity. Application gateway provides cookie-based session affinity to keep a user session on the same service instance using cookies.
- **Secure Sockets Layer (SSL) offload:** Encryption and decryption of request and response data are performed by SSL and is generally a costly operation. Web servers should ideally be spending its resources on processing and serving requests rather than encryption and decryption of traffic. SSL offload helps in transferring this cryptography process from the web server to load balancer thereby providing more resources to web servers serving users. The request from the user is encrypted but gets decrypted at application gateway instead of the web server. The request from application gateway to web server is unencrypted.
- **End to end SSL:** While SSL offload is a nice feature for a certain application, there are certain mission-critical secure applications that need complete SSL encryption and decryption even if traffic passes through load balancers. Application gateway can be configured for an end to end SSL cryptography as well.
- **URL-based content routing:** Application gateway are also useful to redirect the traffic to different servers based on the URL content of incoming requests. This helps in hosting multiple services alongside other applications.

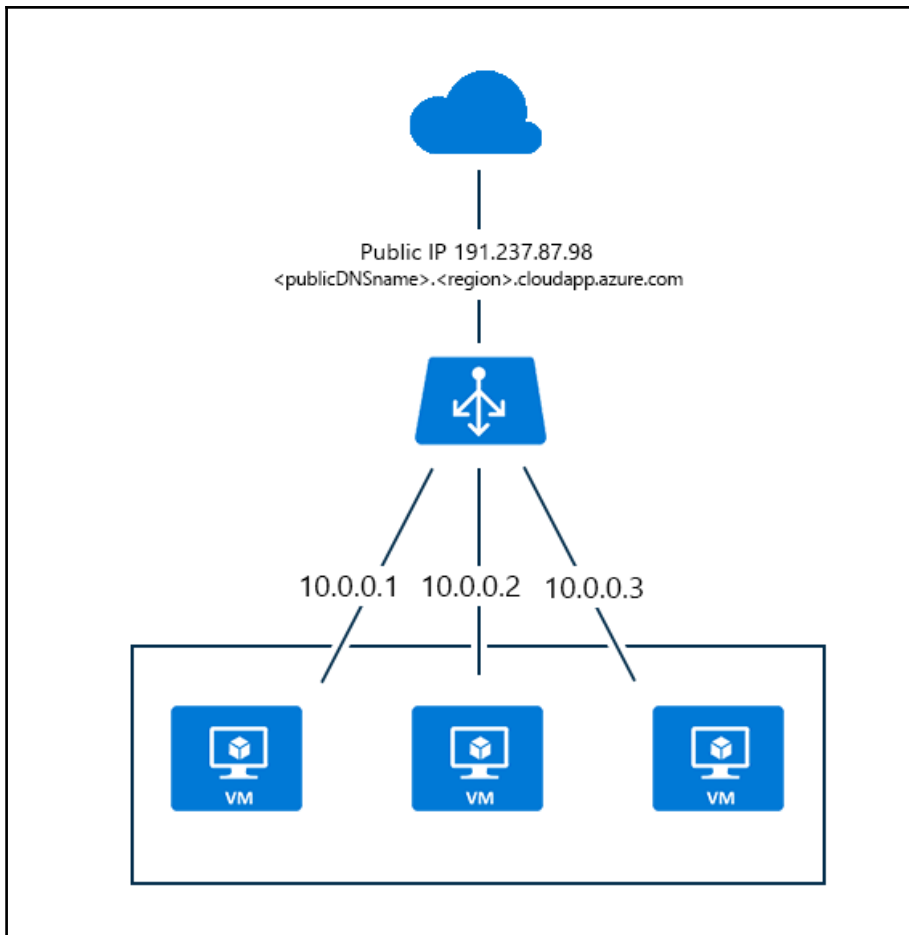
Azure Load Balancers

The Azure load balancer distributes incoming traffic based on transport level information available to it. It relies on the following:

- Originating IP address
- Target IP address
- Originating port number
- Target port number
- Type of protocol--TCP or HTTP

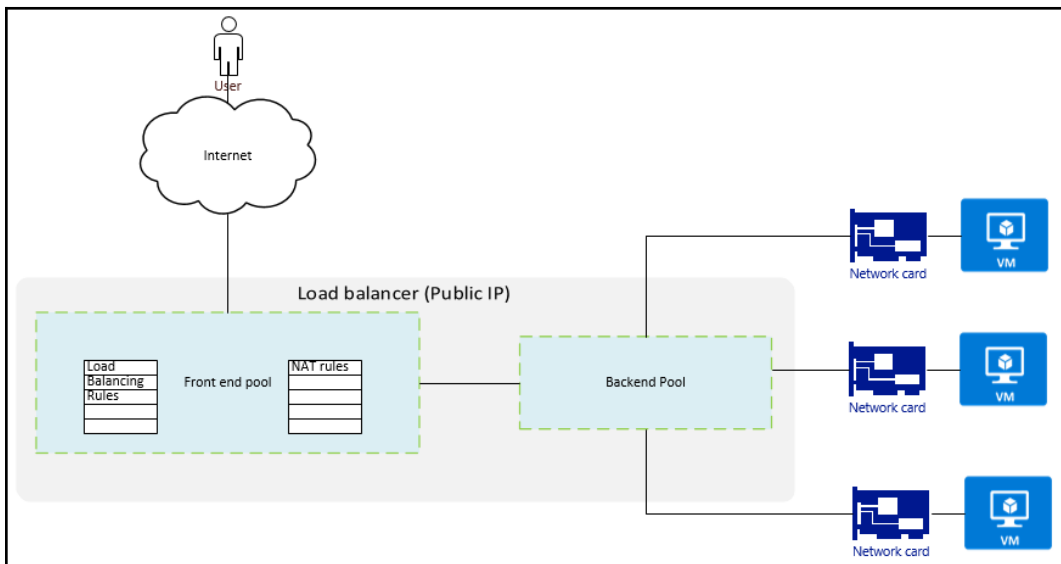
Public Load Balancing

In this configuration, load balancers are assigned a public IP address. Assigning a public IP address ensures that the load balancer can accept requests coming in from the internet. Without a public IP address, it is not possible to access the resource from the internet. The load balancer can be configured with load balancing rules. Load balancing rules work at the port level. It accepts source and destination ports map them together such that whenever a load balancer receives a request for the source port, the request is forwarded to a virtual machine from a group of virtual machines attached to the load balancer on the destination port. This is shown in the following diagram:



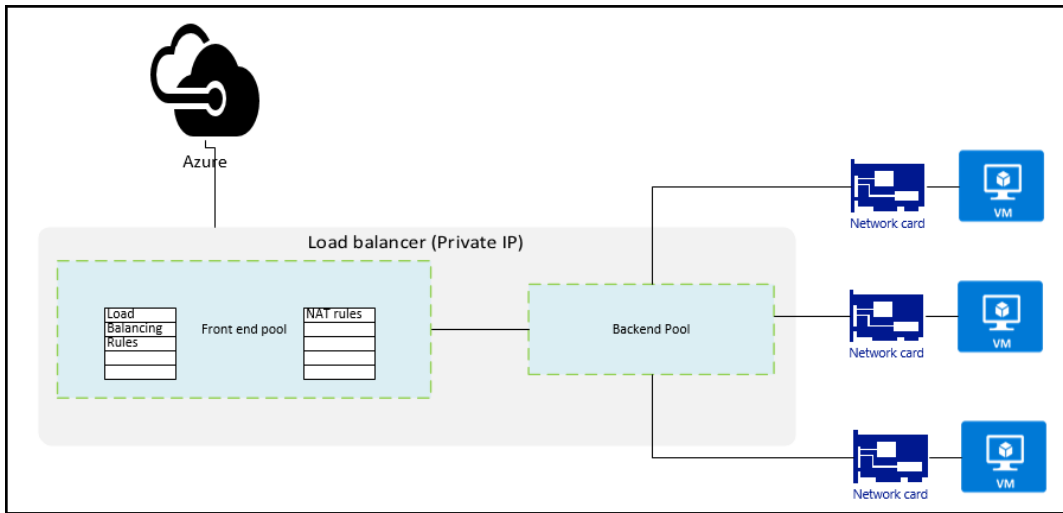
But how does this entire thing work? How is a public IP address assigned to a load balancer? What does the load balancer contain? How is it configured with load balancer rules? How does the load balancer send requests to the virtual machines? How does the virtual machine know that it is attached to the load balancer and more? The answer to all these questions is visible in the following diagram.

In this configuration, the load balancer is assigned a public IP address. The load balancer is accessible from the internet and can accept client requests. The load balancer can be configured with load balancing and NAT rules. Both NAT and load balancing rules are part of the frontend configuration. The frontend configuration sends client requests to one of the IP address available in the backend pool. These IP addresses are associated with the network interface card, which in turn is attached to virtual machines.

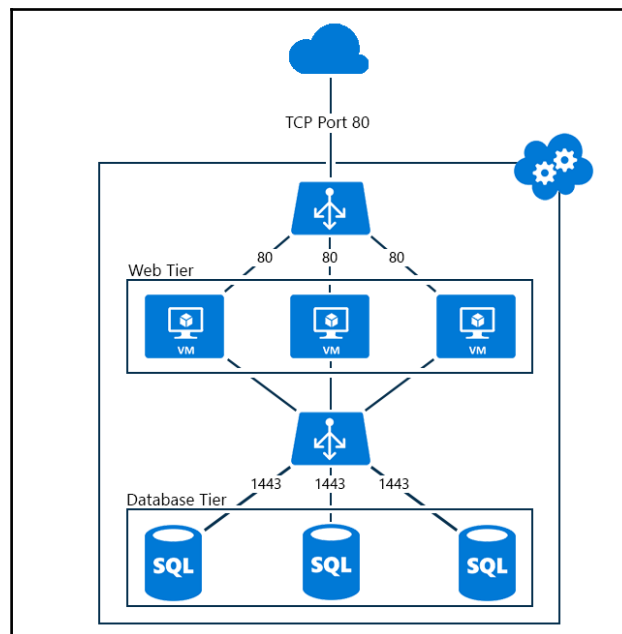


Internal Load Balancing

The following diagram shows the working of an internal load balancer. You can see that the request comes from resources from Azure itself as it is not accessible on the internet. In this configuration, the load balancer is assigned a private IP address. The load balancer is only accessible within the virtual network to which it is attached. It cannot be accessed through the internet. The rest of its configuration is such as public load balancer. The load balancer can be configured with load balancing and NAT rules.



The following diagram shows how multiple load balancers can be deployed to create solutions. In this, there is a public load balancer that accepts client requests and an internal load balancer for the database tier. The database tier virtual machines are not accessible on the internet, but only through the load balancer on port 1433.



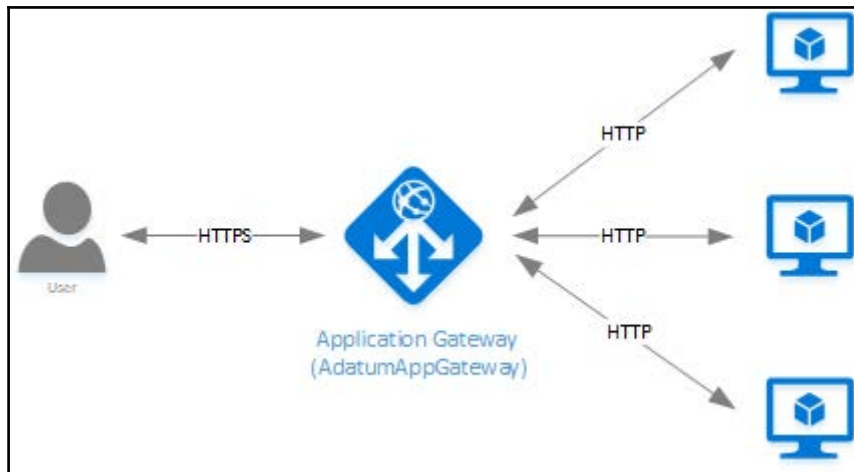
Port Forwarding

At times, there is a need where a request should always redirect to a virtual machine. Azure load balancer helps us achieve this with the NAT rules. NAT rules are evaluated after load balancing rules are evaluated and none of its rules are satisfied. NAT rules are evaluated for each incoming request and once it finds them, it forwards the request to that virtual machine through a backend pool. It is to be noted that a virtual machine cannot register the same port for both port forwarding using NAT and load balancing rules.

Azure Application Gateways

Azure load balancer helps us to enable solutions at the infrastructure level. However, there are times when advance services and features are required from the load balancer itself. These advance services include SSL termination, sticky sessions, advanced security, and more. Azure application gateways are built on top on Azure load balancers to provide these additional features. The Azure application gateway is a level 7 load balancer that works with the application and session payload in a TCP OSI stack. Application gateways have more information compared to the Azure load balancer to take decisions on request routing and load balancing between servers. Application gateways are managed by Azure and are highly available.

An application gateway sits in between the users and virtual machines, as shown in the following figure:



Application gateways are internally implemented using virtual machines. **Internet information service (IIS)** is installed and configured with **Application Request Routing (ARR)** on these virtual machines. These gateways can be installed on multiple virtual machines providing High Availability for the gateways themselves. Although not visible, Azure load balancers distribute loads among multiple application gateway servers. Creating an application gateway needs an internal or public IP address and that is used by users to send requests to it. This public IP or internal IP is provided by the Azure load balancer working at the transport level (TCP/UDP) and having all incoming network traffic being load balanced to the application gateway worker instances. The application gateway then routes the HTTP/HTTPS traffic based on its configuration whether it's a virtual machine, cloud service, internal, or an external IP address.

An application gateway is similar to Azure load balancer from a configuration perspective with additional constructs and features. It provides frontend IP, protocol, certificate and port configuration, backend pool, port, session affinity, and protocol configuration.

Azure Traffic Manager

After having a good understanding of both the Azure load balancer and the application gateway, it's time to get into the details of Traffic Manager. Azure load balancers and application gateways are much-needed resources for High Availability within a data center and region; however, to achieve High Availability across regions and data centers, there is a need for another resource and that is Traffic Manager. Traffic Manager helps us to create highly available solutions that span multiple geographies, regions, and data centers. Traffic Manager is not similar to load balancers. It uses DNS to redirect requests to an appropriate endpoint determined by their health and configuration. Traffic Manager is not a proxy or a gateway. Traffic Manager does not see the traffic passing between the client and the service. It simply redirects the request based on most appropriate endpoints.

Azure Traffic Manager enables you to control the distribution of traffic across your application endpoints. An endpoint is any internet-facing service hosted inside or outside of Azure.

Endpoints are internet facing reachable public URLs. Applications are provisioned within multiple geographies and Azure regions. Applications deployed to each region has a unique endpoint referred by **DNS CNAME**. These endpoints are mapped to Traffic Manager endpoint. When a Traffic Manager is provisioned, it gets an endpoint by default with a `.trafficmanager.net` URL extension.

When a request arrives at the Traffic Manager URL, it finds the most appropriate endpoint out of its list and redirects the request to it. In short, Traffic Manager acts such as a global DNS to identify the region that will serve the request.

However, how does Traffic Manager know which endpoints to use and redirect the client request to? There are two aspects that the Traffic Manager implements to determine the most appropriate endpoint and region.

First, Traffic Manager actively monitors the health of all endpoints. It can monitor the health of virtual machines, cloud services, and app services. If it determines that health of an application deployed to a region is not suitable for redirecting traffic, it redirects the requests to a healthy endpoint.

Second, the Traffic Manager can be configured with routing information. There are four traffic routing methods available in Traffic Manager:

- **Priority:** Should be used when all traffic should go to a default endpoint, and backups are available in case the primary endpoints are unavailable.
- **Weighted:** Should be used to distribute traffic across endpoints evenly or according to defined weights.
- **Performance:** Should be used for endpoints in different regions and users should be redirected to the closest endpoint based on their location. This has direct impact on network latency.
- **Geographic:** This should be used to redirect users from a specific geography to an endpoint (Azure, external, or nested) available in that geography or nearest to that geography. Examples include complying with data sovereignty mandates, localization of content and user experience, and measuring traffic from different regions.

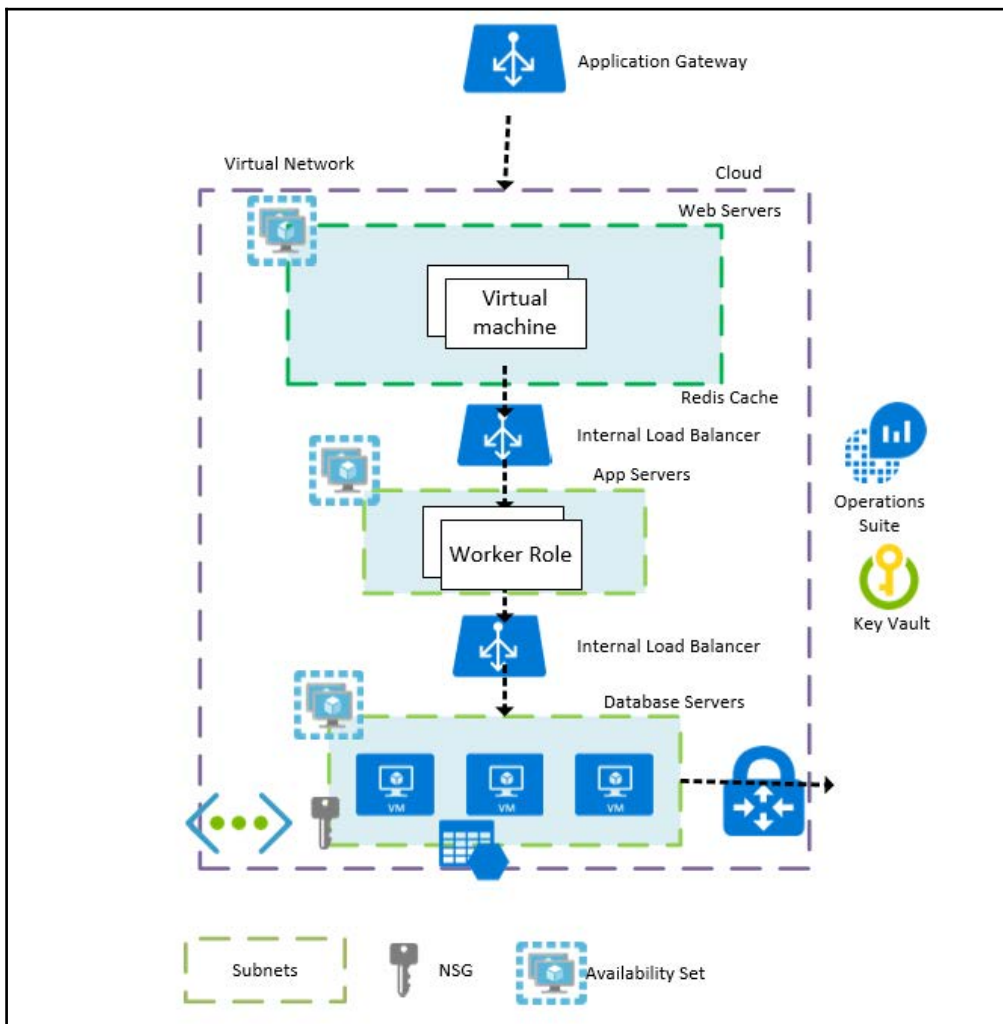
It is to be noted that after the Traffic Manager determines a valid healthy endpoint, clients connect directly to the application.

Architectural Considerations for High Availability

In this section, we will go through some of the architectures for High Availability.

High Availability within Azure regions

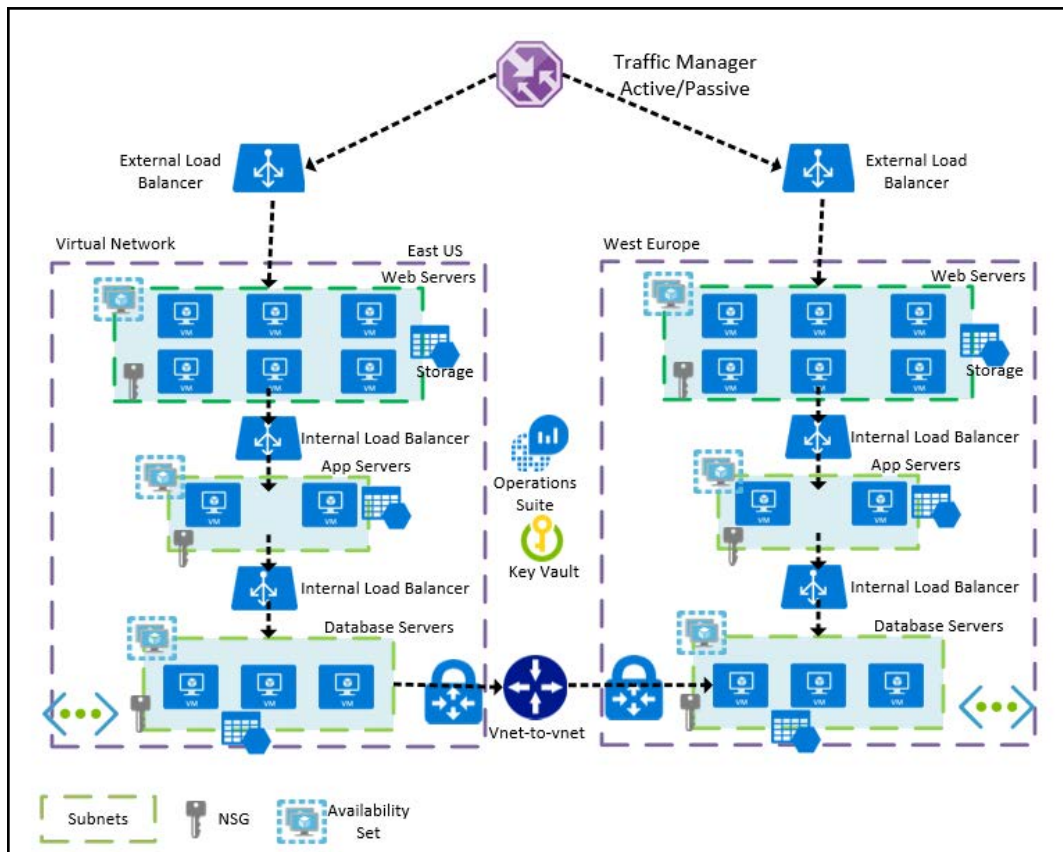
The architecture shown next, shows High Availability deployment within a single Azure region. High Availability is designed at the individual resource level. In this architecture, there are multiple virtual machines at each tier connected through either application gateway or load balancer and they are part of an availability set. Each tier is associated with an availability set. These virtual machines are placed on separate fault and update domains. While the web servers are connected to application gateways, the rest of the tiers such as application and database tiers have internal load balancers. While the web servers are connected to application gateways, the rest of the tiers such as application and database tiers have internal load balancers.



High Availability across Azure regions

The architecture shown next, shows similar deployments on two different Azure regions. Both the regions have the same resources deployed. High Availability is designed at individual resource level within these regions. There are multiple virtual machines at each tier connected through the load balancer and they are part of the availability set. These virtual machines are placed on separate fault and update domains. While the web servers are connected to external load balancers, the rest of the tiers such as application and database tiers have internal load balancers. It is to be noted that application load balancers could have been used for web servers and application tiers instead of Azure load balancer if there is a need for advanced services such as session affinity, SSL termination, advance security using WAF, and path-based routing. Databases in both the regions are connected to each other using VNET peering and gateways. This is helpful in configuring log shipping, SQL Server AlwaysOn, and other data synchronization techniques.

Endpoints of load balancers from both the regions are used to configure Traffic Manager endpoints and traffic is routed based on priority load balancing method. Traffic Manager helps in routing all requests to the East US region and failover to West Europe in case of non-availability of the first region.



Best Practices

This section describes High Availability best practices. They have been categorized into application, deployment, data management, and monitoring.

Application High Availability

An application should be built keeping High Availability as one of the important architectural concerns. Some of the important application related High Availability practices are mentioned next:

- An application should implement appropriate exception handling to gracefully recover and inform stakeholders about the issue
- An application should try to perform the same operation again in the fixed interval for a certain number of times before exiting in an event of an error or exception
- An application should have inbuilt timeout capability to decide that an exception cannot be recovered from
- Maintaining logs and writing logs for all errors, exceptions, and execution should be adopted within the application
- Applications should be profiled to find their actual resource requirements in terms of compute, memory and network bandwidth for a different number of users



Please refer to <https://docs.microsoft.com/en-us/azure/architecture/checklist/availability> for knowing more about application and rest of high availability best practices.

Deployment

Deployment strategy to a large extent affects the availability of application and overall environment. Some of the important things to be taken into consideration are the following:

- Deploy multiple instances of Azure resources including multiple instances for virtual machines, cloud services, and other resources
- Deploy virtual machines on availability sets or availability zones. They cannot be used together
- Deploy multiple instances of virtual machines across multiple regions
- Create multiple environments and keep at least one of them in standby mode

Data Management

Some of the important data related best practices for High Availability are the following:

- If possible, store data on Azure provided services such as Azure SQL, Cosmos DB, and table storage
- Use storage accounts that are based on a geo-redundant type
- Ensure that data is replicated to multiple regions and not only within zone or data center
- Take periodic backups and conduct restore tests frequently
- If storing data in virtual machines, ensure that there are multiple virtual machines and they are either on availability sets or availability zones
- Use keys and secrets to data stores in Azure key vault

Monitoring

Some of the important monitoring related best practices for High Availability are the following:

- Use OMS (log analytics) for monitoring the environment and enable log auditing
- Use application insights to capture telemetry information from the custom application and environment related to compute, storage and network and other log information
- Ensure alerts are configured on OMS for issues related to availability of environment and application
- Visit Azure monitor frequently to gather recommendation related to high availability

Summary

High Availability is an important and crucial architectural concern. Almost every application and every architect tries to implement High Availability. Traditionally, High Availability was implemented from ground-up without any support from platforms. Azure is a mature platform that understands the need of High Availability for applications and in this bid, provides resources to implement High Availability from granular to data center level. High Availability is not an afterthought and should be part of the application life cycle development from the planning phase itself. Azure provides availability sets that ensure virtual machines are placed on separate fault and update domains. Fault domains ensure that unplanned rack level disruption does not change the availability of an application, while update domains take care of planned maintenance. Even Azure PaaS services use fault and update domains behind the scenes to keep the application up and running. Azure internal and external load balancer is level-4 transport level constructs provided by Azure to distribute load among multiple instances of the application. Similarly, application gateways are level-7 application and session level load balancers that provide advanced functionalities such as SSL termination and sticky sessions. Traffic Managers are global DNS level High Availability constructs to make applications across geographies highly available. Next chapter will focus on scalability which is another important facet for overall architecture on Azure.

12

Implementing Scalability

Running applications and systems that are available to users for consumption is an important consideration for architects for any serious application. However, there is another equally important application feature that is one of the top priorities for architects and it is the scalability of the applications. Imagine situations in which applications are deployed and obtain great performance and availability with few users, but both availability and performance degrade as users start increasing. There are times when an application under normal load performs well but degrades in performance with the increase in the number of users. This happens especially if there is a sudden increase in the number of users and the environment is not built for such a large number of users. To accommodate such spikes in the number of users, you might have provisioned the hardware and bandwidth for handling spikes. The challenge with this is that the additional capacity is not used for a majority part of the year and does not provide any return on investment. They are provisioned for use only during few festivals or offers. I hope you are getting the problems architects are trying to solve. All these problems are related to capacity sizing and scalability of an application. The focus of this chapter is to understand scalability as an architectural concern and check out services provided by Azure for implementing scalability.

In this chapter, we'll cover the following sections:

- Scalability
- Scalability in IaaS and PaaS solutions
- Basics of virtual machine scale sets
- Architecture of VMSS
- Auto scaling in VMSS
- Auto scaling rules

- Scaling up and down
- Scaling out and in
- Automation related to VMSS

Scalability

Capacity planning and sizing are one of the top priorities for architects for their applications and services. Architects must find a balance between buying and provisioning too many resources versus fewer resources. Having fewer resources can lead to not being able to serve all users, turning them off to competition, while having more resources can hurt the budgets and return on investments because most of these resources remain unused most of the time. Moreover, the problem is amplified with a varied level of demand during different times. It is almost impossible to predict the number of users for the application round the clock and year. However, it is possible to find an approximate number using past information and continues monitoring.

According to Wikipedia (<https://en.wikipedia.org/wiki/Scalability>), scalability refers to:

"Scalability is the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged to accommodate that growth. For example, a system is considered scalable if it is capable of increasing its total output under an increased load when resources (typically hardware) are added."

Scalability refers to the ability in application deployment, process, and technology to handle a growing number of users and providing them with the same level of performance when there are fewer users. Scalability might refer to serving more requests without degradation of performance or it might refer to handling larger and more time-consuming work without any loss of performance in both cases.

Capacity planning and sizing exercises should be undertaken by architects at the very beginning of the project during the planning phase to provide scalability to applications.

Some applications have stable demand patterns while it is difficult to predict others. Scalability requirements are known for stable demand applications while it is a more involved process for variable demand applications. Auto scaling, a concept we will review in the next section, should be used for such applications whose demands cannot be predicted.

Scalability versus Performance

It is quite easy to get confused between scalability and performance architectural concerns because scalability is all about ensuring that no matter the number of users consuming the application, all get the same pre-determined level of performance.

Performance relates to application features that ensure that the application caters to predefined response time and throughput, scalability refers to having provision for more resources on a need basis to accommodate more users without sacrificing performance.

It is better to understand them using an analogy. Speed of a train refers to the performance for railway systems, however, accommodating more trains to run in parallel with the same or higher speed will be referred to as scalability of the railway network.

Azure Scalability

In this section, we will see the features and capabilities provided by Azure to make applications highly available. Before we get into the architecture and configuration details, it is important to understand Azure High Availability related concepts.

Concepts

The fundamental constructs provided by Azure to attain High Availability is through:

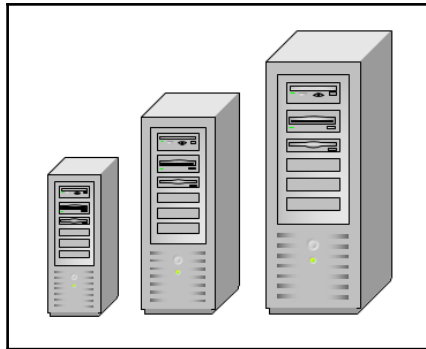
- Scaling
- Scaling up and down
- Scaling out and in
- Auto scaling
- Rolling updates

Scaling

Scaling refers to the transformation that either increases or decreases the units of resources used to serve requests from users. Scaling can be automatic or manual. Manual scaling requires an administrator to manually initiate the scaling process, while automatic scaling refers to an automatic increase or decrease of resources based on events available from the environment and ecosystem such as memory and CPU availability. Scaling can be up or down and out or in, which will be explained later in this section.

Scaling Up

Scaling up of a virtual machine or service refers to adding up to additional resources to existing servers such as CPU, memory, and disks. It is to increase the capacity of existing physical hardware and resources.



Scaling Down

Scaling down of a virtual machine or service refers to the removal of existing resources from existing servers such as CPU, memory, and disks. It is to decrease the capacity of existing physical and virtual hardware and resources.

Scaling Out

Scaling out refers to the process of adding additional hardware in terms of additional servers and capacity. This typically involves adding new servers, assigning them IP addresses, deploying applications on them, and making them part of the existing load balancers such that traffic can be routed to them. Scaling out can be automatic or manual as well. However, for better results automation should be used.



Scaling In

Scaling in refers to the process of removal of the existing hardware in terms of existing servers and capacity. This typically involves removing existing servers, de-allocating their IP addresses, removing them from existing load balancer configuration such that traffic cannot be routed to them. Like scaling out, scaling in can be automatic or manual.

Auto Scaling

Auto scaling refers to the process of either scaling up/down or scaling out/in dynamically based on application demand and it happens using automation. Auto scaling is helpful because it ensures that deployment always consists of a correct and ideal number of server instances. Auto scaling helps in building applications that are fault tolerant. It not only helps in scalability, but also makes applications highly available. Finally, it provides the best cost management. Auto scaling helps in having the optimal configuration for server instances based on demand. It helps in not over provisioning servers that are underutilized or removes servers that are not needed anymore after scaling out.

PaaS Scalability

Azure provides app services for hosting managed applications. App services is a PaaS offering from Azure. It provides the web and mobile platform. Behind these web and mobile platform is a managed infrastructure that is managed by Azure on behalf of its users. Users do not see or manage the infrastructure; however, they have the capability to extend the platform and deploy their applications on top of it. With this, architects and developers can concentrate on their business problems instead of worrying about the base platform and infrastructure provisioning, configuration, and troubleshooting. Developers have the flexibility to choose any language, operating system, and frameworks to develop their applications. App services provide multiple plans and based on chosen plans, capabilities of scalability are available. App services provide five plans. They are as follows:

- **Free:** It uses shared infrastructure. It means multiple applications will be deployed on the same infrastructure from the same or multiple tenants. It gets 1 GB of storage free of cost. No scaling facility is available in this plan.
- **Shared:** It also uses shared infrastructure and gets 1 GB of storage free of cost. Additionally, custom domains are also provided as an extra feature. No Scaling facility is available in this plan.
- **Basic:** It has three different **stock keeping units (SKU)**--B1, B2, and B3. They have increasing units of resources available to them in terms of CPU and memory. In short, they provide higher configuration of virtual machines backing these services. Additionally, they provide storage, custom domains, and SSL support. The basic plan provides basic features for manual scaling. There is no automatic scaling available in this plan. Maximum three instances can be used for scaling out of the application.
- **Standard:** It also has three different SKU--S1, S2, and S3. They have increasing units of resources available to them in terms of CPU and memory. In short, they provide higher configuration of virtual machines backing these services. Additionally, they provide storage, custom domains, and SSL support similar to the basic plan. It also provides traffic manager, staging slots, and one daily backup as an additional feature on top of the basic plan. The standard plan provides features for automatic scaling. Maximum 10 instances can be used for scaling out of the application.

- Premium:** It also has three different SKU--P1, P2, and P3. They have increasing units of resources available in them in terms of CPU and memory. In short, they provide higher configuration of virtual machines backing these services. Additionally, they provide storage, custom domains, and SSL support similar to the basic plan. It also provides traffic manager, staging slots, and 50 daily backups as an additional feature on top of the basic plan. The premium plan provides features for automatic scaling. Maximum 20 instances can be used for scaling out of the application.

PaaS Scaling Up and Down

Scaling up and down of services hosted in-app services is quite simple. The Azure app services menu items to scale up, which opens a new blade with all plans and their SKU's listed. Choosing a plan and SKU will scale up or down the service.

The screenshot displays the Azure App Service Scale Up/Down blade. On the left is a navigation pane with options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, SETTINGS (Apps, File system storage, Networking, Scale up (App Service plan), Scale out (App Service plan), Properties, Locks, Automation script), and OBSERVE (Resource explorer). The main area shows a grid of plans categorized into Premium (P1, P2, P3) and Standard (S1, S2, S3). Each plan card lists its core count, RAM, storage, and various features like custom domains, SSL support, instance scaling, staging slots, backups, and traffic manager. Premium plans offer up to 20 instances, while Standard plans offer up to 10 instances. Pricing is shown in INR/MONTH (ESTIMATED).

P1 Premium		P2 Premium		P3 Premium	
1	Core	2	Core	4	Core
1.75	GB RAM	3.5	GB RAM	7	GB RAM
250 GB Storage		250 GB Storage		250 GB Storage	
Custom domains / SSL SNI Incl & IP SSL Support		Custom domains / SSL SNI Incl & IP SSL Support		Custom domains / SSL SNI Incl & IP SSL Support	
Up to 20 instance(s) * Subject to availability		Up to 20 instance(s) * Subject to availability		Up to 20 instance(s) * Subject to availability	
20 slots Web app staging		20 slots Web app staging		20 slots Web app staging	
50 times daily Backup		50 times daily Backup		50 times daily Backup	
Traffic Manager Geo availability		Traffic Manager Geo availability		Traffic Manager Geo availability	
14,752.68 INR/MONTH (ESTIMATED)		29,505.37 INR/MONTH (ESTIMATED)		59,010.73 INR/MONTH (ESTIMATED)	

S1 Standard		S2 Standard		S3 Standard	
1	Core	2	Core	4	Core
1.75	GB RAM	3.5	GB RAM	7	GB RAM
50 GB Storage		50 GB Storage		50 GB Storage	
Custom domains / SSL SNI Incl & IP SSL Support		Custom domains / SSL SNI Incl & IP SSL Support		Custom domains / SSL SNI Incl & IP SSL Support	
Up to 10 instance(s) Auto scale		Up to 10 instance(s) Auto scale		Up to 10 instance(s) Auto scale	
Daily Backup		Daily Backup		Daily Backup	
5 slots Web app staging		5 slots Web app staging		5 slots Web app staging	
Traffic Manager Geo availability		Traffic Manager Geo availability		Traffic Manager Geo availability	

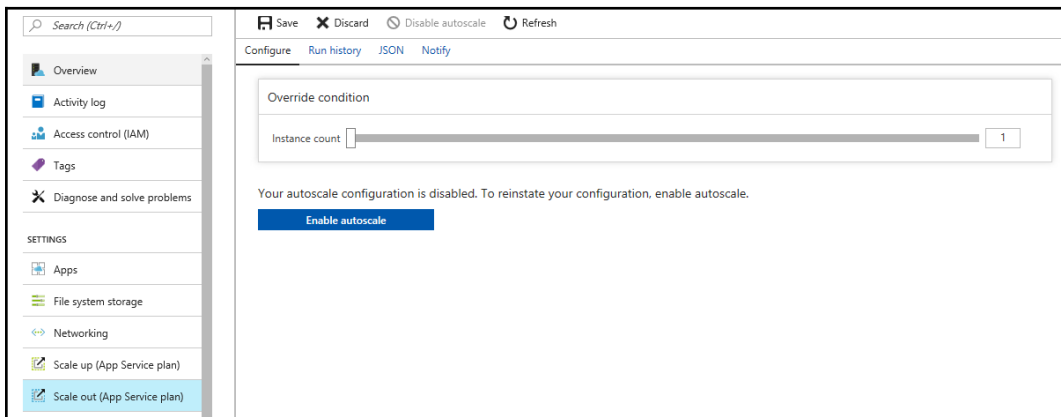
A 'Select' button is located at the bottom of the plan grid.

PaaS Scaling Out and In

Scaling out and in of services hosted in-app services is also quite simple. The Azure app services menu items to scale out, which opens a new blade with scaling configuration options.

By default, auto scaling is disabled for both premium and standard plans. It can be enabled using **Scale Out** menu item and by clicking on the **Enable autoscale** button.

Manual scaling does not need configuration but auto scaling helps in configuring with the help of following properties.



- **Mode of scaling:** Based on some metric such as CPU or memory usage or just scale to specify the number of instances.
- **When to scale:** Multiple rules can be added that determine when to scale out and in. Each rule can determine the criteria such as CPU or memory consumption, whether to increase or decrease instances, how many instances to increase or decrease at a time. At least one rule for scale out and one rule for scale in should be configured. Threshold definition helps in defining the upper and lower limits cross which should trigger auto scale--either increase or decrease the number of instances.

- **How to scale:** Specifies how many instances to create or de-provision in each scale out or in.

The screenshot shows the Azure Autoscale configuration page for an App Service plan. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, SETTINGS, Apps, File system storage, Networking, Scale up (App Service plan), Scale out (App Service plan), Properties, Locks, Automation script, OBSERVE, Resource explorer, and Alerts. The main content area is titled 'Default Auto created scale condition' and includes a 'Resource group' dropdown set to 'demoRG'. Below this, the 'Scale mode' is set to 'Scale based on a metric'. A 'Rules' section contains a single rule: 'Scale out and scale in your instances based on metric. For example, add a rule that increases instance count by 1 when CPU percentage is above 70%'. The 'Instance limits' section shows 'Minimum' set to 1, 'Maximum' set to 1, and 'Default' set to 1. The 'Schedule' section states 'This scale condition is executed when none of the other scale condition(s) match'. On the right, the 'Criteria' section is configured with 'Time aggregation' set to 'Average', 'Metric name' set to 'CPU Percentage', 'Time grain statistic' set to 'Average', 'Operator' set to 'Greater than', 'Threshold' set to 70, and 'Duration (in minutes)' set to 10. The 'Action' section is configured with 'Operation' set to 'Increase count by', 'Instance count' set to 1, and 'Cool down (minutes)' set to 5. An 'Add' button is at the bottom right.

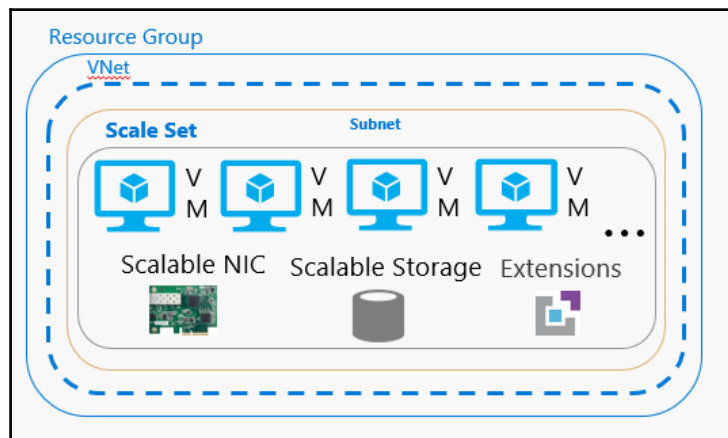
IaaS Scalability

There are users who want to have complete control over the base infrastructure, platform, and application. They prefer to consume IaaS solutions compared to PaaS solutions. For such customers, when they create virtual machines they are also responsible for capacity sizing and scaling. There is no out-of-the-box configuring for manual or auto-scaling of virtual machines. These customers will have to write their own automation scripts, triggers, and rules to achieve auto scaling. With virtual machines comes the responsibility to maintain them as well. Patching, updates, and upgrades of virtual machines is the responsibility of owners. Architects should think about both planned as well as unplanned maintenance. How these virtual machines should be patched, its order, grouping, and other factors must be thought through to ensure that both scalability and availability of an application is not comprised. To help alleviate such problems, Azure provides virtual machine scale sets as a solution.

Virtual Machine Scale Sets

Virtual machine Scale Sets (VMSS) are an Azure compute resource that you can use to deploy and manage a set of identical VMs. With all VMs configured in the same way, scale sets are designed to support true auto scale, and no pre-provisioning of VMs is required. It helps in provisioning multiple identical virtual machines connected to each other through a virtual network and subnet.

VMSS creates a set that can be created, configured, and managed as a unit. All virtual machines are part of this unit and any changes made are applied to the unit, which in turn applies it to virtual machines using a pre-determined algorithm.



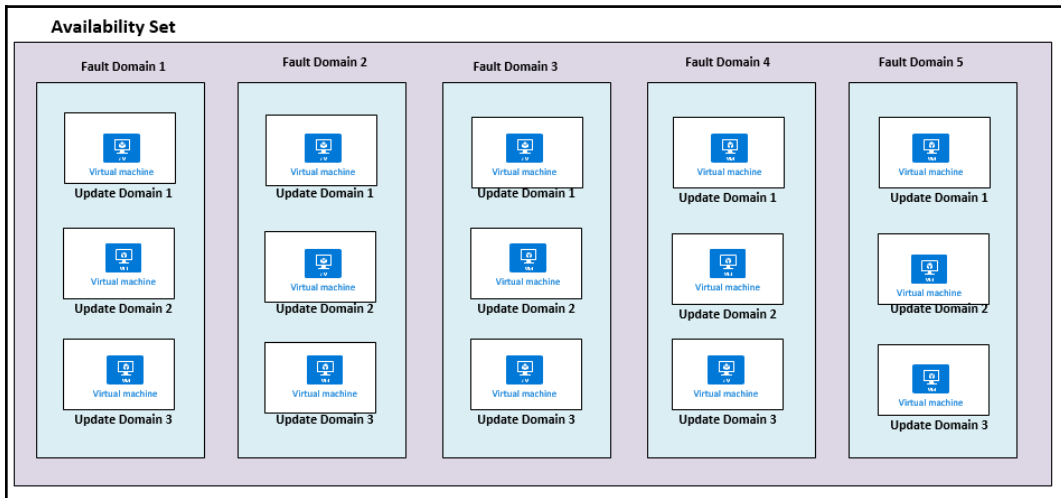
It allows these virtual machines to be load balanced using Azure load balancer or application gateways. All the virtual machines could be either Windows or Linux operating system. They can run automated scripts using a PowerShell extension and they can be managed centrally using desired state configuration. They can be monitored as a unit and individually using log analytics as well.

VMSS can be provisioned from the Azure portal, Azure command-line interface, Azure resource manager templates, REST API's, and PowerShell cmdlets. It is possible to invoke REST API's and Azure CLI from any platform, environment, operating system, and any language.

Already a lot of Azure services use VMSS as its underlying architecture. Major among them are Azure batch, Azure service fabric, and Azure container services. Azure container services, in turn, provisions Kubernetes and DC/OS on these virtual machine scale sets.

VMSS Architecture

VMSS allows for the creation of up to 1000 virtual machines in a scale set when using a platform image and 100 virtual machines if using a custom image. If the number of virtual machines are less than 100 in a scale set, they are placed in a single availability set; however, if they are greater than 100, multiple availability sets are created, known as placement groups and virtual machines are distributed among these availability sets. We know from the last chapter that virtual machines in an availability set are placed on separate fault and update domains. Availability sets related to VMSS have five fault and update domains by default. VMSS provides a model that holds metadata information for the entire set. Changing this model and applying changes impact all virtual machine instances. This information includes maximum, minimum virtual machine instances, operating system SKU and version, the current number of virtual machines, fault and update domain, and more.



VMSS Scaling

Scaling refers to the increase or decrease in compute and storage resources. VMSS is a feature rich resource that makes scaling easy and efficient. It provides auto scaling that helps in scaling up or down based on external events and data such as CPU and memory usage.

Horizontal versus Vertical Scaling

Scaling can be horizontal or vertical or both. Horizontal scaling is another name for scaling out and in, while vertical scaling is about scaling up and down.

Capacity

VMSS have capacity property that determines the number of virtual machines in a scale set. VMSS can be deployed with zero as value for this property. It will not create a single virtual machine; however, if you provision VMSS by providing a number for the capacity property, those number of virtual machines are created.

Auto Scaling

Automatic scaling of virtual machines in VMSS refers to the addition or removal of virtual machine instances based on configured environments to meet the performance and scalability demands of an application. Generally, in the absence of VMSS, this is achieved using automation scripts and runbooks.

VMSS helps in this automation process with the help of configuration. Instead of writing scripts, VMSS can be configured for automated scaling up and down.

Auto scaling consists of multiple integrated components to achieve its end goal. Auto scaling continuously monitors the virtual machines and collects telemetry data from it. It stores this data and combines it together and evaluates it against a set of rules to determine whether it should trigger auto scale. The trigger could be to a scale out or scale in. It could also be for scale up or down.

Autoscale uses diagnostics logs for collecting telemetry data from virtual machines. These logs are stored in storage accounts as diagnostics metrics. Autoscale also uses the insight monitoring service that reads these metrics, combines them together, and stores them into its own storage account.

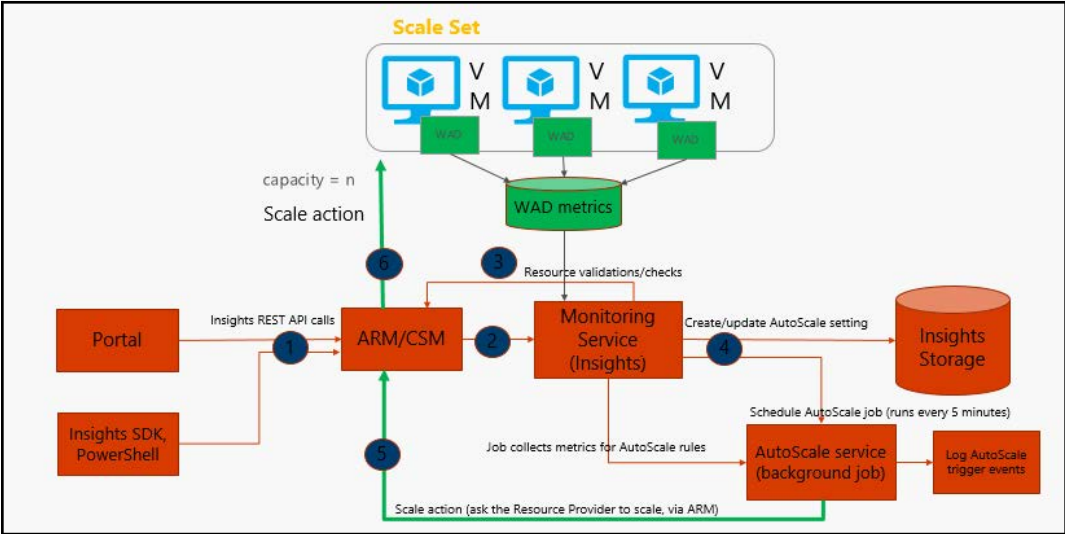
Auto scale background jobs continually run to read the insights storage data, evaluate them based on all the rules configured for auto scaling, and executes the process of auto scaling, if any of the rules or combination of rules returns positive. The rules can take into consideration metrics from guest virtual machine as well as host server.

The rules are defined using the following properties. These properties descriptions are available at <https://docs.microsoft.com/en-us/azure/virtual-machine-scale-sets/virtual-machine-scale-sets-autoscale-overview>.

Rule	Description
<code>metricName</code>	This value is the same as the performance counter that you defined in the <code>wadperfcounter</code> variable for the diagnostics extension. In the preceding example, the thread count counter is used.
<code>metricResourceUri</code>	This value is the resource identifier of the VMSS. This identifier contains the name of the resource group, the name of the resource provider, and the name of the scale set to scale.
<code>timeGrain</code>	This value is the granularity of the metrics that are collected. In the preceding example, data is collected on an interval of one minute. This value is used with <code>timeWindow</code> .
<code>statistic</code>	This value determines how the metrics are combined to accommodate the automatic scaling action. The possible values are <code>--average</code> , <code>min</code> , and <code>max</code> .
<code>timeWindow</code>	This value is the range of time in which instance data is collected. It must be between 5 minutes and 12 hours.
<code>timeAggregation</code>	This value determines how the data that is collected should be combined over time. The default value is <code>average</code> . The possible values are <code>--average</code> , <code>minimum</code> , <code>maximum</code> , <code>last</code> , <code>total</code> , and <code>count</code> .
<code>operator</code>	This value is the operator that is used to compare the metric data and the threshold. The possible values are <code>--Equals</code> , <code>NotEquals</code> , <code>GreaterThan</code> , <code>GreaterThanOrEqual</code> , <code>LessThan</code> , and <code>LessThanOrEqual</code> .
<code>threshold</code>	This value is the value that triggers the scale action. Be sure to provide a sufficient difference between the threshold values for the scale-out and scale-in actions. If you set the same values for both actions, the system anticipates constant change, which prevents it from implementing a scaling action. For example, setting both to 600 threads in the preceding example doesn't work.
<code>direction</code>	This value determines the action that is taken when the threshold value is achieved. The possible values are <code>increase</code> or <code>decrease</code> .

Rule	Description
type	This value is the type of action that should occur and must be set to ChangeCount.
value	This value is the number of virtual machines that are added to or removed from the scale set. This value must be 1 or greater.
cooldown	This value is the amount of time to wait for the last scaling action before the next action occurs. This value must be between one minute and one week.

The auto scale architecture is shown in the following diagram:



Autoscale can be configured for scenarios that are more complex than general metrics available from environments. For example, scaling could be based on any of the following event:

- Scale on a specific day
- Scale on a recurring schedule such as weekends
- Scale differently on weekdays and weekends
- Scale during holidays that is, one of the events
- Scale on multiple resource metrics

These can be configured using the schedule property of insights resources that help in registering rules.

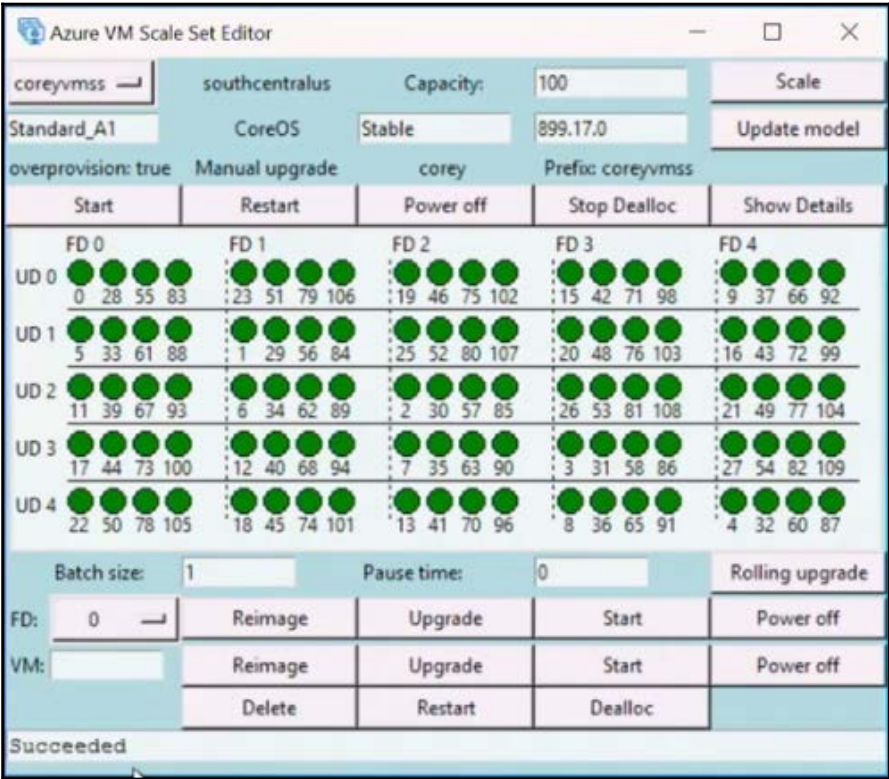
Architects should ensure that at least two actions--scale out and scale in, should be configured together. Scaling in or scaling out configuration will not help achieve scaling benefits provided by VMSS.

Upgrades

After VMSS and applications are deployed, they need to be actively maintained. Planned maintenance should be conducted periodically to ensure that both environment and application is up to date with latest bits and the environment is current from a security and resilience point of view.

Upgrades can be associated with applications, the guest virtual machine instance, or to the image itself. Upgrades can be quite complex because they should happen without affecting the availability, scalability, and performance of environments and applications. To ensure that updates can take place one instance at a time using rolling upgrade methods, it is important that VMSS supports and provides capabilities for these advanced scenarios.

There is a utility provided by the Azure team to manage updates for VMSS. It's a Python-based utility that can be downloaded from <https://github.com/gbowerman/vmssdashboard>. It makes REST API calls to Azure to manage scale sets. This utility can be used for starting, stopping, upgrading, and reimaging for virtual machines on a fault domain or group of virtual machines.



Application Updates

Application updates in VMSS should not be executed manually. It must be executed as part of release management and pipeline using automation. Moreover, the update should happen one application instance at a time, not affecting the overall availability and scalability of the application. Configuration management tools such as desired state configuration should be deployed to manage application updates. DSC pull server can be configured with the latest version of the bits and they should be applied on a rolling basis to each instance.

Guest Updates

Updates to virtual machine are the responsibility of administrator. Azure is not responsible to patch guest virtual machines. Guest updates are in the preview mode and users should control patching manually or using custom automation such as runbooks and scripts. However, rolling patch upgrades are in preview and can be configured in the ARM template using upgrade policy as shown here:

```
"upgradePolicy": {  
  "mode": "Rolling",  
  "automaticOSUpgrade": "true" or "false",  
  "rollingUpgradePolicy": {  
    "batchInstancePercent": 20,  
    "maxUnhealthyUpgradedInstanceCount": 0,  
    "pauseTimeBetweenBatches": "PT0S"  
  }  
}
```

Image Updates

VMSS can update the OS version without any downtime. OS update involves changing the version or SKU of the OS or changing the URI of a custom image. Updating without downtime means updating virtual machines one at a time or in groups (such as one fault domain at a time) rather than all at once. By doing so, any virtual machines that are not being upgraded can keep running.

Scaling Best Practices

In this section, we will go through some of the best practices that applications should implement to take advantages of scalability capability provided by VMSS.

Prefer Scaling Out

Scaling out is a better scaling solution compared to scaling up. Scaling up or down means re-sizing of virtual machine instances. When a virtual machine is resized, it generally needs to be restarted, which has its own disadvantages. First, there is a downtime for the machine. Second, if there are active users connected to the application on that instance, they might face unavailability of application or they might even have lost transactions. Scaling out does not impact existing virtual machines. It provisions newer machines and adds them to the group.

Bare Metal versus Dormant Instances

Scaling new instances can take two broad approaches. Either create the new instance from scratch, which means to install applications, configure, and test, while on the other hand, there can be dormant instances sleeping and can be started when they are needed due to scalability pressure on other servers.

Configuring Maximum and Minimum Number of Instances Appropriately

Setting a value of two for both minimum and maximum instance count with current instance count being two, no scale action can occur. There should be an adequate margin between the maximum and minimum instance counts, which are inclusive. Autoscale always scales between these limits.

Concurrency

Applications are designed for scalability to focus on concurrency. An application should use asynchronous patterns to ensure that client requests do not wait indefinitely for acquiring resources if resources are busy serving other requests. Implementing asynchronous patterns in code ensures that threads do not wait for resources and systems gets exhausted of all available threads. Applications should implement the concept of timeouts if there are intermittent failures expected.

Stateless

Applications and services should be designed to be stateless. Scalability can become a challenge to achieve with stateful services and it is quite easy to scale stateless services. With state comes the requirement of additional components and implementation such as replication, centralized or decentralized repository, maintenance, and sticky sessions. All these are impediments in the path to scalability. Imagine a service maintaining active state on a local server. No matter the number of requests on the overall application or on an individual server, the subsequent requests must be served by the same server. Subsequent requests cannot be processed by other servers. This makes scalability implementation a challenge.

Caching and CDN

Applications and services should take advantage of caching. Caching helps eliminate multiple subsequent calls to either databases or filesystem. This helps in making resources available and free for more requests. **Content Distribution Network (CDN)** is another mechanism for caching static files such as images and JavaScript libraries. They are available on servers across the globe. They also make resources available and free for additional client requests. This makes applications highly scalable.

N+1 Design

N+1 design refers to building redundancy within overall deployment for each component. It means to plan for some redundancy even when it is not required. It could mean additional virtual machines, storage, and network interface card.

Summary

Scalability is an important and crucial architectural concern. Almost every application and every architect try to implement scalability along with availability and performance. Azure is a mature platform that understands the need of scalability for applications and provides scalability options for both PaaS as well as IaaS solutions. PaaS app services can be configured to auto scale and virtual machines can be deployed on a scale set to take advantage of scaling. Scaling can be up/down/out/in. Similar to High Availability, scalability is not an afterthought and should be part of the application lifecycle development from the planning phase itself. Scaling can be vertical by increasing the size of virtual machine instance or can be horizontal in which additional servers are added to the existing set. VMSS provides a model that holds metadata information for the entire set. Changing this model and applying changes impact all virtual machine instances. This allows upgrading, resizing, stopping, and starting virtual machines on a rolling basis. Finally, some of the important best practices were covered in this chapter related to scalability.

Next chapter deals with security which is the most important architectural concern for cloud deployments.

13

Designing IoT Solutions

So far, we have been dealing with architectural concerns and their solutions on Azure in general. This chapter is not based on generalized architecture. This chapter is about one of the most disruptive technologies of this century. This chapter will get into the details of IoT architecture on Azure

This chapter will specifically cover the following sections:

- Azure and IoT
- Azure IoT overview
- Device management
 - Registering devices
 - Device to IoT hub communication
- Scaling IoT solutions
- High availability of IoT solutions
- IoT protocols
- Using message properties for routing messages

IoT

IoT is an abbreviation and when expanded is **Internet of Things**. There are two important keywords in this phrase--Internet and Things. To understand it better, let's go back a few centuries in history to relate to the current emergence of this technology.

In the nineties, internet was invented and became available generally to everyone although the penetration was less. During this time, almost everyone moved towards having a presence on the internet and started creating static web pages. These static web pages were showing details about various aspects. Eventually, the static content became dynamic and content was generated on the fly based on context. In almost all cases, a browser was needed for accessing the internet. There were a plethora of browsers, but without them consuming the internet was a challenge.

During the first decade of this century, there was an interesting development happening--the rise of handheld devices in the form of mobile phones and tablets. Mobile phones started becoming cheaper by the day and available ubiquitously. The hardware and software capabilities of these handheld devices were improving considerably. So much so, people started using browsers on mobile devices rather than desktops. But there was a subtle change that was noticeable and it was the rise of mobile apps. The mobile apps were downloaded from some store and were connected to the internet to talk to backend systems. Towards the end of the last decade, there were millions of apps available with almost every conceivable functionality build in them. The backend system for these apps was built on the cloud so that they could be scaled rapidly. This was the age of connecting applications and servers.

But, was this the pinnacle of innovation? What was the next evolution in making this better for everyone at large? If you look closely there was another paradigm taking center stage. This was IoT. When apps connected to backend systems, it was basically a device that had compute, storage, and was connected to the internet making requests. How about connecting every device connected to the internet? Instead of just mobile and tablets connected to the internet, there can be other devices that can be connected to the internet. These devices were available in select markets, were costly, not available to masses, and had limited hardware and software capabilities. However, during the first part of the current decade commercialization of these devices started in big scale. These devices were becoming smaller and smaller, more capable from hardware and software perspective, had more storage and compute power, could connect to the internet on various protocols, and could be attached to almost anything. This is the age of connecting devices to server, applications, and to other devices.

This led to the formulation of the idea and applications that could change the way industries were operating. Newer solutions that were unheard of before started becoming reality. Now, these devices could be attached to anything and they could get information and send it to a backend system that could assimilate that information from all the devices and either take proactive action or report the happenings.

Examples of IoT are vehicle tracking systems that track all vital parameters of a vehicle and send details to centralized data store for analysis, smart city services, such as tracking pollution levels, temperature, street congestion, and so on, and agriculture-related activities related to soil fertility, humidity, and so on.

IoT Architecture

Before getting into Azure and its features and services related to IoT, it is important to understand various components needed to create end -to-end IoT solutions.

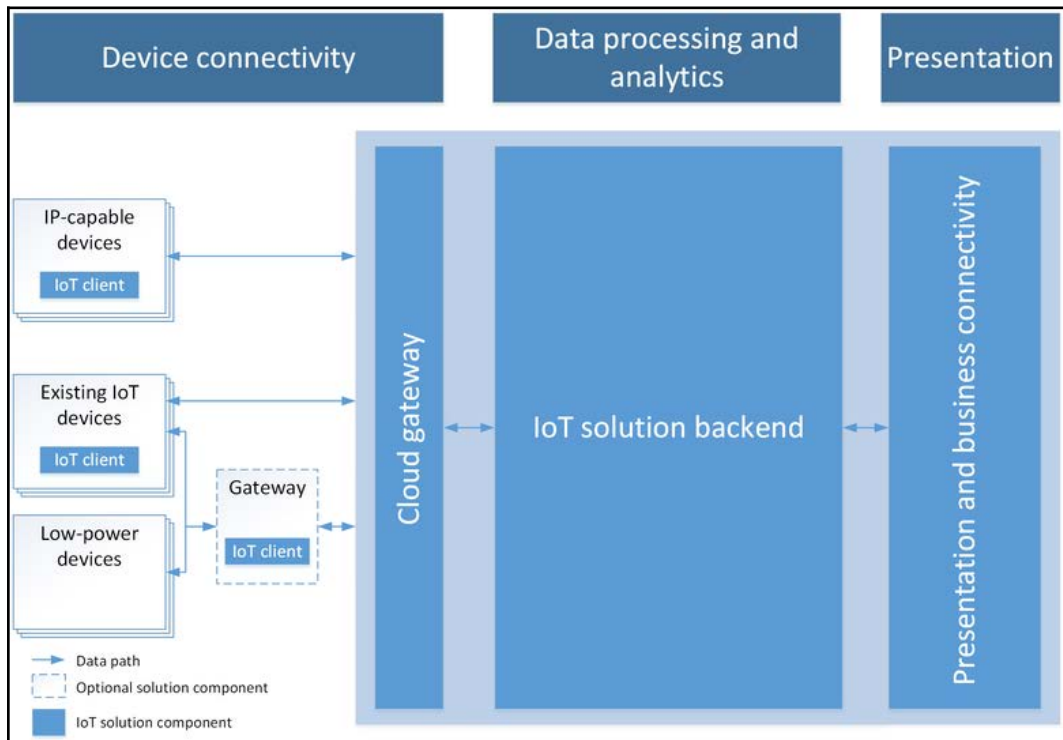
Imagine IoT devices across the globe sending millions of messages every second to a centralized database. Why is this data collected? This data is collected to extract rich information about events, anomalies, outliers, and happenings within those devices and objects they are connected to.

Let's understand this in more detail.

IoT architecture can be divided into distinct phases:

- Connectivity
- Identity
- Capture
- Ingestion
- Storage
- Transform
- Analytics
- Presentation

The next images show a generic IoT based architecture. Data are generated or collected by devices and send over to cloud gateways. The cloud gateway, in turn, sends the data to multiple backend services for processing. Cloud gateways are an optional component. They should be used when devices themselves are not capable of sending requests to backend services either because of their resource constraints or lack of reliable network. These Cloud gateways can collate data from multiple devices and send to backend services. Data processes by backend services are then shown as insights or dashboards to users.



Connectivity

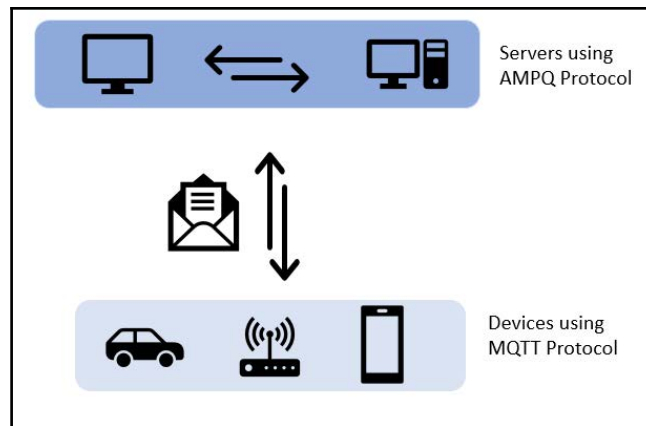
IoT devices need to communicate. There are various connectivity types. It could be between devices in a region, between devices and a centralized gateway, devices to IoT platform, and more.

In all cases mentioned before, IoT devices need connectivity capability. This capability could be in terms of internet connectivity, bluetooth, infrared, or any other near device communication.

Some devices might not have the capability to connect to the internet. In those cases, they can connect through other means to a gateway, which in turn has connectivity to the internet.

IoT devices use protocols to send messages. The major protocols among these are **Advanced Message Queuing Protocol (AMQP)** and **Message Queue Telemetry Transport (MQTT)**.

Devices data should be sent to IT infrastructure. MQTT protocol is a device to server protocol that devices can use to send telemetry and other information to servers. Once the server receives the message through the MQTT protocol it needs to transport the messages to other servers using a reliable technology based on messages and queues. AMQP is the preferred protocol to move messages between servers in the IT infrastructure in a reliable and predictable manner.



The servers getting initial messages from IoT devices should send messages to other servers for processing, such as saving to logs, evaluation, analytics, and presentation purposes.

Some devices do not have capabilities to connect to the internet or do not support the protocols understood by IT protocols. As mentioned before, IoT devices have various capabilities and some of the features that affect their capabilities are mentioned next. To enable these devices to work with the IoT platform and the cloud, intermediate gateways help in adapting them for sending information to cloud. Gateways help in on-boarding devices that connectivity and networks are slow and not consistent, devices use protocols that are not standard, and their capabilities are limited in terms of resources and power.

In such circumstances when devices need additional infrastructure to participate and connect to backend services, client gateways can be deployed. These gateways receive messages from near devices and forward and push them to IT infrastructure and the IoT platform for further consumption. These gateways are capable of protocol translation if required.

Identity

IoT devices should be registered with the cloud platform. Not every device should be allowed to connect to the cloud platform. The devices should be registered and be assigned an identity. The device should be sending the identity information while connecting and sending information to the cloud. In case the device fails to send this identity information, the connectivity should fail. We will see later in this chapter about how to generate an identity for a device using simulation application.

Capture

IoT devices should be able to capture the information from itself and the ecosystem around it. It should have the capability for example to read the moisture content in air or in soil. The information can be captured based on frequency, which could be as low as per second. Once the information is captured, it should be able to send it across to the IoT platform for processing. If a device does not have the capability to connect to the IoT platform directly, it can connect to an intermediary and cloud gateways to push the captured information. The size of captured data and frequency is the most important aspect for the device. Whether the device has local storage and temporarily store the captured data is another important aspect that should be taken care of. The device can work in an offline mode if there is local storage available. Even mobile devices sometimes act as IoT devices connected to various instruments and have the capability to store data.

Ingestion

The data captured and generated by devices should be sent to an IoT platform that is capable of ingesting and consuming that data to extract meaningful information and insights out of it. The ingestion service is an important and crucial service because its availability and scalability affects the throughput of incoming data. If data starts getting throttled due to scalability issues or data is not able to ingest due to availability issues, data would be lost and might get biased or skewed.

Storage

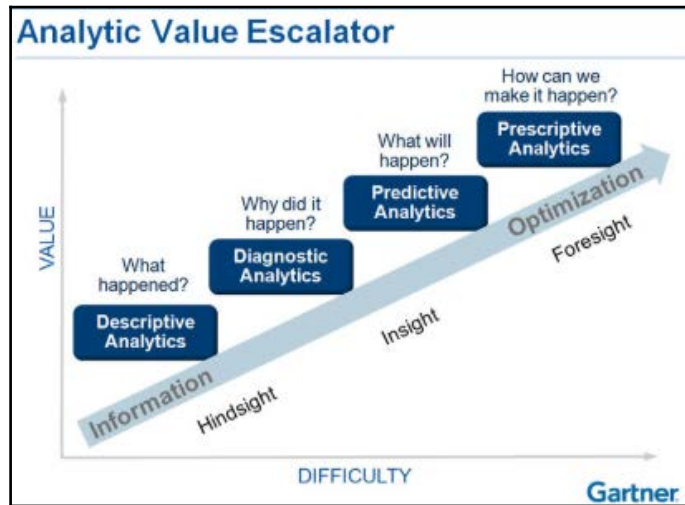
IoT solutions generally deal with millions and billions of records spanning terabytes and petabytes of data. This is valuable data that can provide insights on operations and its health. This data needs to be stored such that analytics can be performed on it. The storage should be readily available for analytics applications and service to consume it. They should provide adequate throughput and latency from a performance perspective, be highly available and scalable, and secure in nature.

Transform

IoT solutions are generally data-driven and have considerable high volume. Imagine each car having a device and each one is sending messages every five seconds. If there are a million cars sending messages, it will be equal to 288 million messages per day and 8 billion messages per month. Together this data has lots of hidden information and insights; however, making sense of this kind of data is quite difficult by mere viewing. The data captured and stored can be consumed for solving business problems. Depending on the nature of the problem, not all data captured is of importance. There could be a subset of data that should be used for solving a problem. The data captured and stored could not be consistent as well. To ensure that the data is consistent, not biased or skewed, appropriate transformation should be executed on it for making it readily consumable. This process is typically known as transformation. Here, data is filtered, sorted, removed, enriched, and transformed to a structure; further downstream components and applications can consume readily.

Analytics

The data transformed in the previous step becomes the input and source of data for analytics. Depending on the nature of business and problem at hand there are different multiple types of analytics that can be performed on transformed data.



The preceding figure is taken from *Gartner* and it very well depicts the different types of analytics that can be performed:

- **Descriptive Analytics:** This type of analytics helps in finding patterns and details about statuses of IoT devices and overall health. This is generally the first stage of analytics that identifies and summarizes the data for further consumption by more advanced analytics. It will help in finding information such as summarization, statistics related to probability, deviation, and other simple statistical concepts.
- **Diagnostic Analytics:** This type of analytics is more advanced than descriptive analytics. It builds on top of descriptive analytics and tries to answer queries about why things happened. It tries to find the root cause of the events that happened. It tries to find answers using advanced concepts, such as hypothesis and correlation.

- **Predictive Analytics:** This type of analytics tries to predict things that have a high probability of happening in future. It's about prediction based on past data. Regression is one of the examples based on past data, it could, for example, predict the price of a car, stock in the stock market, when the next tire will burst and so on.
- **Prescriptive/Cognitive Analytics:** This analytics are at the highest maturity levels and help take actions automatically. This analytics help in identifying the most appropriate action that should be executed to ensure that the health of devices and solution do not degrade and proactive measures can be undertaken. This helps in avoiding and eliminating the problems from the root.

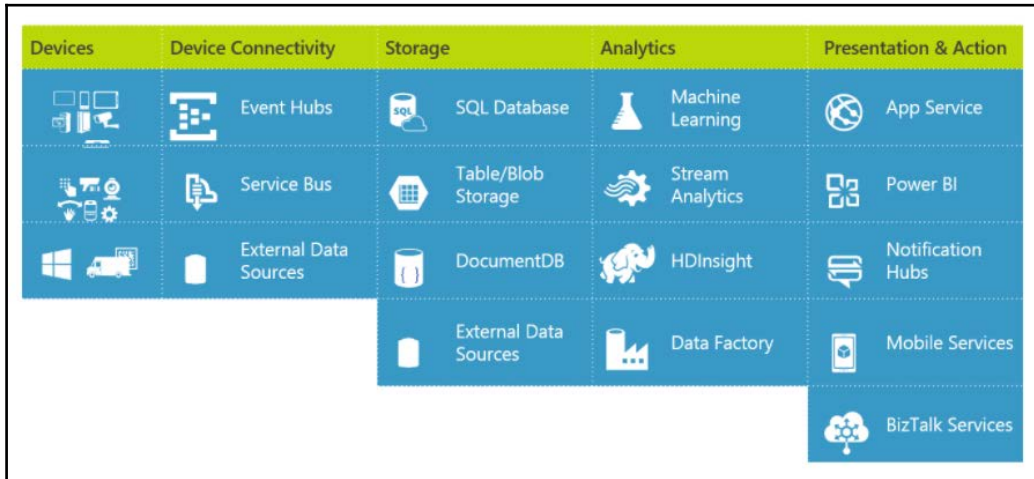
Presentation

Analytics help in identifying answers, patterns, and insights based on data available from the past. These insights also need to be available to all stakeholders in different forms and format in which they can understand. It should be readily consumable and available in multiple formats. Appropriate dashboards and reports can be generated--statistically or dynamically, and then can be presented to stakeholders. The stakeholders can consume these reports for further action and improve continuously on their solution.

Azure IoT

After learning details about the various stages that help in creating end-to-end IoT solutions. Each of these stages are crucial and their implementation is a must for its success. Azure provides lots of services for each of these stages. Apart from these services, Azure provides IoT hubs--the core IoT service and platform that is capable of hosting complex, highly available, and scalable IoT solutions.

We will dive deep into IoT hubs after going through other services. This section will detail in brief each of these services.



Identity

Azure IoT hubs also provide services for authenticating devices. IoT hubs provide interface for generating unique identity hashes for each device. When devices send their messages containing this hash, the IoT hub can identify them after verification of its own database for the existence of such hashes.

Capture

Azure provides IoT gateways that enable IoT hubs non-compliant devices to get adapted and enable pushing of data. There are local or intermediary gateways deployed near devices such that multiple devices can connect to a single gateway to send their information. Similarly, multiple such clusters of devices with a local gateway can be deployed. There can be a cloud gateway deployed on the cloud itself, capable of accepting data from multiple sources and ingesting them for IoT hubs.

Ingestion

Azure provides an IoT hub that becomes a single point of contact for devices and other applications to send data. In other words, ingestion of IoT messages is the responsibility of the IoT hub service. There are other services, such as event hubs and service bus messaging infrastructure that can provide ingestion of incoming messages; however, the benefits and advantages of IoT hubs for ingesting IoT data far outlast compared to event hubs and service bus messaging. In fact, IoT hubs have been made specifically for the purpose of ingesting IoT messages within the Azure ecosystem such that the rest of the services and components can act on them.

Storage

Azure provides multiple types of storage for storing messages from IoT devices. These storage accounts include storing relational data, schema-less NoSQL data, and blobs:

- **SQL database:** SQL database provides storing relational data, JSON, and XML documents. It provides rich SQL query language and it uses a full-blown SQL server as a service. Data from devices can be stored in SQL databases if data is well defined and expectation is that the schema will not undergo changes frequently.
- **Azure storage:** Azure storage provides table and blob storage. Table storage helps in storing data as entities where schema is not important. It is an implementation of NoSQL databases. Blobs help in storing files in containers as blobs.

- **CosmosDB/DocumentDB:** DocumentDB is a full blown enterprise scale NoSQL database, available as a service capable of storing schema less data. It is a truly distributed database that can span continents providing high availability and scalability of data.
- **External data sources:** Apart from Azure services, customers can bring their own datastores, such as SQL server on Azure virtual machines, and can use them for storing data in relational format.

Transform and Analytics

- **Data factory:** Azure data factory is a cloud-based data integration service that allows us to create data-driven workflows in the cloud for orchestrating and automating data movement and data transformation. Azure data factory helps to create and schedule data-driven workflows (called pipelines) that can ingest data from disparate data stores, process/transform the data by using compute services such as **Azure HDInsight Hadoop, Spark, Azure Data Lake Analytics, and Azure Machine Learning**, and publish output data to data stores such as Azure SQL data warehouse for **business intelligence (BI)** applications to consume. It's more of an **Extract-and-Load (EL)** and then **Transform-and-Load (TL)** platform rather than a traditional **Extract-Transform-and-Load (ETL)** platform.
- **Azure HDInsight:** Microsoft and Hortonworks have come together to help companies by offering the big data analytics platform on the Azure cloud service. HDInsight is a high-powered, fully managed cloud service environment powered by Apache Hadoop and Apache Spark using Microsoft Azure HDInsight. It helps in accelerating workloads seamlessly with Microsoft and Hortonworks' industry leading big data cloud service.
- **Azure stream analytics:** This is a fully managed real-time data analytics service that helps in performing computation and transformation on streaming data. Stream analytics can examine high volumes of data flowing from devices or processes, extract information from the data stream, and look for patterns, trends, and relationships.
- **Machine learning:** Machine learning is a data science technique that allows computers to use existing data to forecast future behaviors, outcomes, and trends. Using machine learning, computers learn without being explicitly programmed. Azure machine learning is a cloud predictive analytics service that makes it possible to quickly create and deploy predictive models as analytics solutions.

It provides a ready-to-use library of algorithms to create models on an internet-connected PC, and deploy predictive solutions quickly.

Presentation

After appropriate analytics have been conducted on data, the data should be presented to stakeholders in a format that is consumable by them. There are numerous ways insights from data can be presented. This includes presenting data using web applications deployed using Azure app services, sending data to notification hubs that can then notify mobile applications and more. However, the ideal approach for presenting and consuming insights is by way of using **Power BI** reports and dashboards. Power BI is a Microsoft visualization tool for rendering dynamic reports and dashboard on the internet accessible for anywhere on any network.

IoT Hubs

IoT projects are generally complex in nature. The complexity arises because of high volume of devices and data, devices are embedded across the world, monitoring and audit of devices, storage of data, transformation and analytics on petabytes of data, and finally taking action based on insights. Moreover, these projects are of long duration, have long gestation periods, and their requirements keep changing because of timelines.

If any enterprise wants to embark on a journey for IoT projects, sooner than later it would be realized that these are not easy problems to solve. These projects need large hardware in terms of compute and storage and services that can work with such high volumes of data.

IoT hub is a platform that is built to help ease and enable IoT projects for faster, better, and easier delivery. It provides all the necessary features and services for the following:

- Device registration
- Device connectivity
- Field gateways

- Cloud gateways
- Implementation of industry protocols such as AMQP and MQTT
- Hub for storing incoming messages
- Routing of messages based on message properties and content
- Multiple endpoints for different types of processing
- Connectivity to other services on Azure for analytics and real-time analytics and more

Protocols

Azure IoT hub natively supports communication over the MQTT, AMQP, and HTTP protocols. In some cases, devices or field gateways might not be able to use one of these standard protocols and will require protocol adaptation. In such cases, custom gateway can be deployed. A custom gateway can enable protocol adaptation for IoT hub endpoints by bridging the traffic to and from the IoT hub.

Device Registration

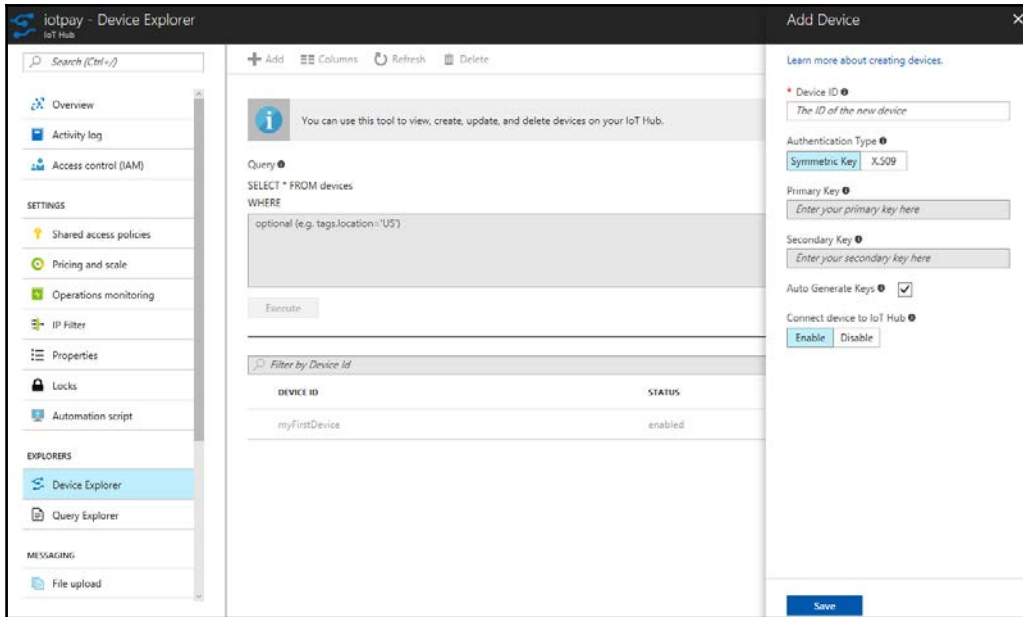
Devices should be registered before they can send messages to the IoT hub. Registration of devices can be done manually using Azure portal or can be automated using IoT hub SDK. Azure provides sample simulation applications, with the help of which it becomes easy to register virtual devices for development and testing purposes. There is also a Raspberry Pi online simulator that can be used as a virtual device and then obviously there are other physical devices that can be configured to connect to the IoT hub.

To simulate a device from a local PC generally used for development and testing purposes, there are tutorials available on Azure documents for multiple languages. They are available at <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-get-started-simulated>.

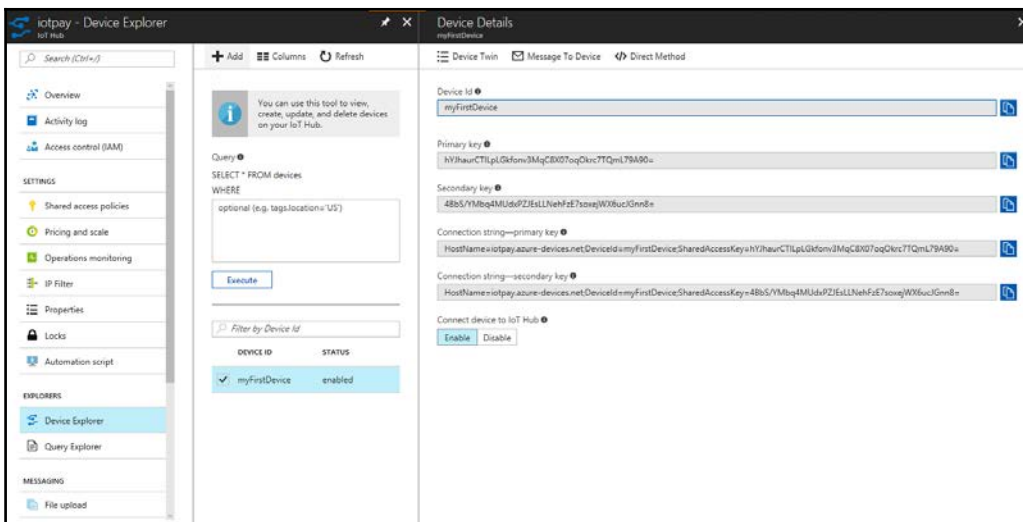


Raspberry Pi online simulator is available at <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-raspberry-pi-web-simulator-get-started> and for using physical devices to be registered with IoT hub, steps mentioned at <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-get-started-physical> should be used.

For manually adding a device using the Azure portal, IoT hub provides a **Device Explorer** menu and it can be used for configuring a new device.



After the device identity is created, primary key connection string for IoT hub should be used in each device to connect to it.



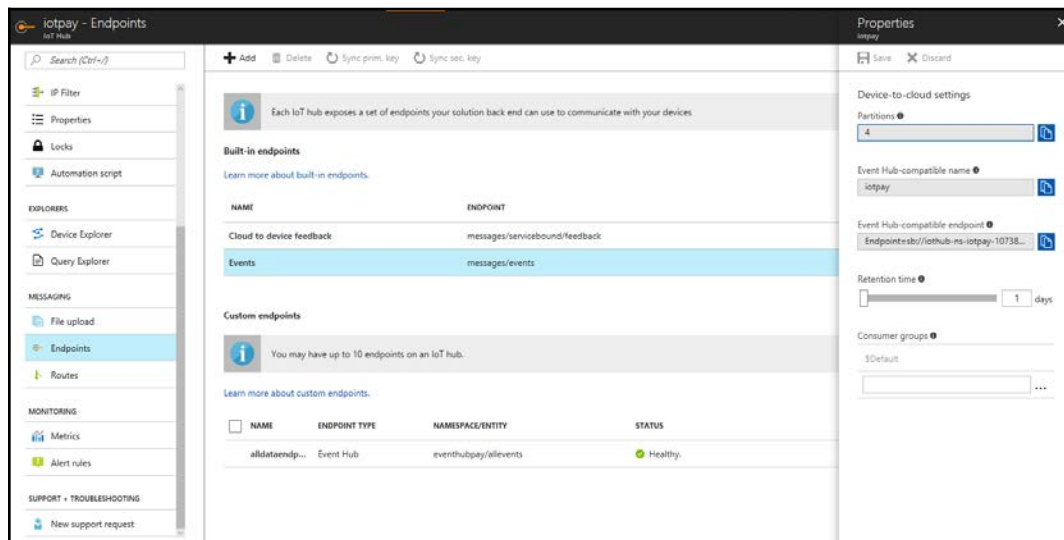
Message Management

After devices are registered with the IoT hub, they can start interacting with it. The interaction could be from device to cloud or from cloud to device.

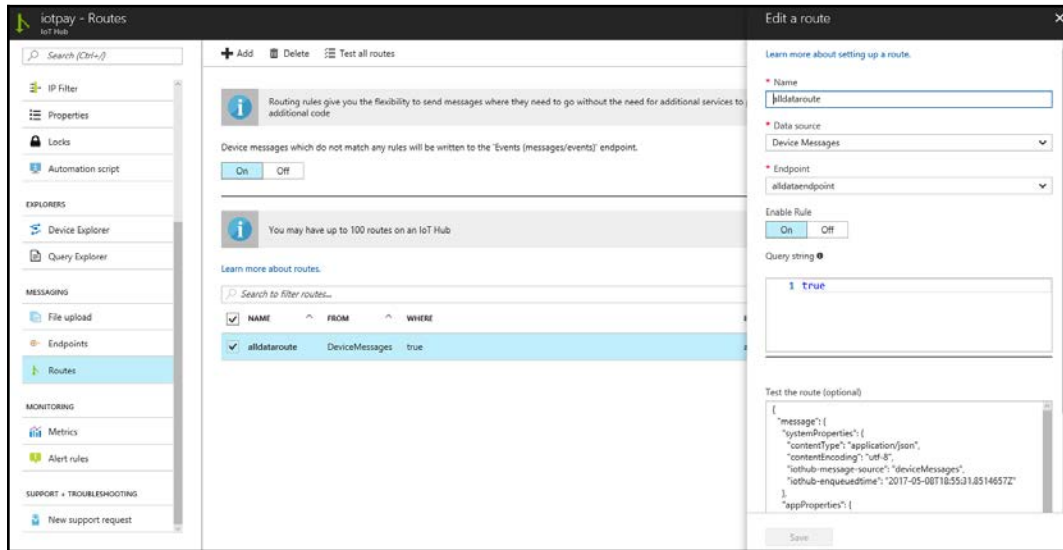
Device to Cloud Messaging

One of the best practices that must be followed in this communication is that although the device might be capturing a lot of information, only data that is of any importance should be transmitted to the cloud. The size of the message is very important in IoT solutions because of the inherent nature that IoT solutions are generally very high in volumes. Even 1 KB of extra data can result in a gigabyte of storage and processing wasted. Each message has properties and actual payload. Properties define the metadata for the message. This metadata contains data about the device, identification, tags, and other properties that are helpful in routing and identifying messages.

Devices or cloud gateways should connect to IoT Hubs to transfer data. IoT hubs provide public endpoints that can be utilized by devices to connect and send data. IoT hub should be considered as the first point of contact for backend processing. IoT hub is capable of further transmit and routing of these messages to multiple services. By default, the messages are stored in Event Hub. Multiple Event Hub can be created for different kinds and types of messages.



Messages can be routed to different endpoints based on message header and body properties and the same is shown in next image.



Message in IoT hub stays within for 7 days by default. Their size can go up to 256 KB.

There is a sample act that acts as a simulator for sending messages to the cloud. It is available in multiple languages and the C# version is at

<https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-csharp-csharp-c2d>.

Cloud to Device Messaging

Azure IoT hub is a managed service providing bi-directional messaging infrastructure. Messages can be sent from cloud to devices and based on the message the devices can act on them.

There are three types of cloud to device messaging patterns:

- Direct methods require immediate confirmation of the result. Direct methods are often used for interactive control of devices such as opening and closing garage shutters. It follows the request-response pattern.

- Twin's desired properties for long-running commands intend to put the device into a certain desired state. For example, set the telemetry send interval to 30 minutes. Device twins are JSON documents that store device state information (metadata, configurations, and conditions). IoT hub persists a device twin for each device in the IoT hub.
- Cloud-to-device messages for one-way notifications to the device app. This follows the fire and forget pattern.

Security

Security is an important aspect in IoT based applications. IoT based applications comprise of devices that use public internet for connectivity to backend applications. Securing devices, backend applications, and connectivity from malicious users and hackers should be considered a top priority for the success of these applications.

Security in IoT

IoT applications are primarily built around the internet, and security should play a major role to ensure that the solution is not comprised of an identity, confidentiality, and integrity point of view. Some of the important security decisions affecting IoT architecture are the following:

- Devices using HTTP versus HTTPS rest endpoints--rest endpoints secured by certificates ensure that messages transferred from device to cloud and vice versa are well encrypted and signed. The messages should make no sense to an intruder and should be extremely difficult to crack.
- If devices connected to a local gateway, the local gateway should connect to the cloud using a secure HTTP protocol.
- Devices should be registered to cloud IoT hubs before they can send any messages.
- The information passed to cloud should be persisted into storage that is well protected from a confidentiality, integrity, and identity perspective. Appropriate SAS tokens or connection strings that are stored in the Azure Key vault should be used for the connection.
- Azure key vault should be used to store all secrets, passwords, and credentials including certificates.

Scalability

Scalability for IoT hub is a bit different than other services. In IoT hub, there are two types of messaging:

- **Incoming:** Device to cloud messages
- **Outgoing:** Cloud to device messages

And both need to be accounted for in terms of scalability.

IoT hub provides a couple of configuration options during provision time to configure scalability. These options are also available post provisioning and can be updated to better suit the solution requirements in terms of scalability.

The scalability options available for IoT hub are:

- The Sku edition that is the size of the IoT hub
- Number of units

Sku Edition

The Sku in IoT hub determines the number of messages it can handle per unit per day and this includes both incoming as well as outgoing messages. There are four Sku's defined. They are as follows:

- **Free:** It allows for 8000 messages per unit per day and allows both incoming and outgoing messages. Maximum 1 unit can be provisioned. This edition is suitable for gaining familiarity and testing out the capabilities of the IoT Hub service.
- **Standard (S1):** It allows for 400,000 messages per unit per day and allows both incoming and outgoing messages. Maximum 200 units can be provisioned. This edition is suitable for a small number of messages.
- **Standard (S2):** It allows for six million messages per unit per day and allows both incoming and outgoing messages. Maximum 200 units can be provisioned. This edition is suitable for a large number of messages.

- **Standard (S3):** It allows for 300 million messages per unit per day and allows both incoming and outgoing messages. Maximum 10 units can be provisioned. This edition is suitable for a very large amount of messages.

Choose your pricing and scale tier

S1 Standard	S2 Standard	S3 Standard
400k messages/unit/day	6M messages/unit/day	300M messages/unit/day
Device-to-cloud telemetry	Device-to-cloud telemetry	Device-to-cloud telemetry
Cloud-to-device messaging	Cloud-to-device messaging	Cloud-to-device messaging
200 units maximum	200 units maximum	10 units maximum
3,304.81 INR PER IOT HUB UNIT	33,048.13 INR PER IOT HUB UNIT	330,481.25 INR PER IOT HUB UNIT

F1 Free

8k messages/unit/day

Device-to-cloud telemetry

Cloud-to-device messaging

1 unit

IoT hub cannot transition between free and paid tiers.

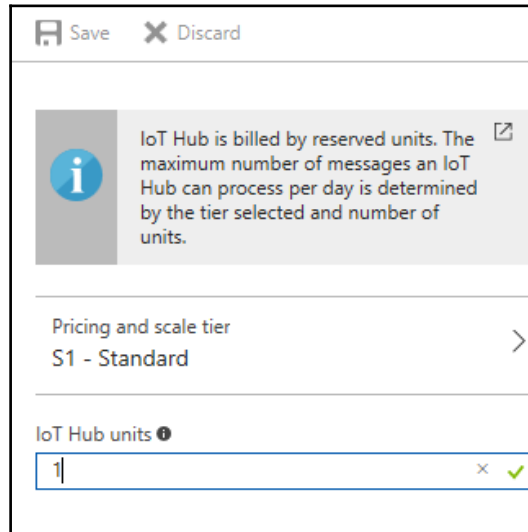
Select

An astute reader would have noticed that Standard S3 Sku allows for a maximum of 10 units only compared to other standard units that allow for maximum 200 units. This is directly related to the size of the machines running behind. The size and capability of virtual machines for Standard S3 is significantly higher compared to other Sku's where the size remains same.

Units

Units define the number of instance of each SKU running behind the service. For example, 2 units of Standard S1 Sku will mean that the IoT hub is capable of handling $400K * 2 = 800K$ messages per day.

Units increase the scalability of the application.



The screenshot shows a configuration window for IoT Hub with the following elements:

- Buttons: Save, Discard
- Information box: IoT Hub is billed by reserved units. The maximum number of messages an IoT Hub can process per day is determined by the tier selected and number of units.
- Section: Pricing and scale tier
- Value: S1 - Standard
- Section: IoT Hub units
- Input field: 1
- Buttons: x, ✓

High Availability

IoT hub is a PaaS service from Azure. Customers and users do not directly interact with the underlying number and size of virtual machines on which IoT hub service is running. Users decide on the region, the Sku of the IoT hub and number of units for their application. The rest of the configuration is determined and executed by Azure behind the scenes. Azure ensures that every PaaS service is highly available by default. It does so by ensuring that multiple virtual machines provisioned behind the service are on separate racks in the data center. It does this by placing those virtual machines on an availability set and on a separate fault and update domain. This helps in high availability for both planned as well as unplanned maintenance. Availability sets take care of high availability at data center level.

To achieve high availability across multiple data centers in a region or across regions, customers should take additional action to provision additional IoT hub services in different regions, ensure that devices are registered at all IoT hub services, and each have the same identifier. Create a separate monitoring application on each region such as a web application, all tied up using a traffic manager and continually monitoring the traffic manager endpoint for current location unavailability to redirect traffic to other IoT hubs, and write logic to route the device messages to the new IoT hub.

Summary

IoT is one of the biggest upcoming technologies of this decade and is already disrupting the industries with the type of solutions they can create. Things that sounded impossible are possible to track, monitor, and take actions remotely. IoT hub acts as a platform that eases creating and delivering IoT solutions to the customer in a faster, better, and cheaper way. It provides implementation of all industry protocols such as MQTT and AMQP, along with field gateways that can adapt non-standard devices. It provides high availability, scalability, and security features to both messages and overall solution. It provides connectivity to a large number of Azure services, helps in routing messages to multiple different endpoints--each capable for processing messages. IoT can fast-track the entire development lifecycle and help faster go-to-market strategies for companies.

Azure provides numerous data storage options and services. Next chapter will be a primer on data storage and services on Azure.

14

Designing and Implementing Data Solutions

Data is the new currency is becoming true these days. Any organization that has any considerable data is sitting on a pile of intelligence and insights that can turn the future around. During the last decade there have been innovations that have contributed massively to data explosion. The first among them is the popularity of social platforms. There are platforms, such as Facebook, LinkedIn, and Twitter that have millions if not billions of users across the globe, generally this means millions of content every day. There is lots of wisdom and insights into this social networking content. Another big data changer paradigm is IoT. IoT devices are generating billions of messages every day for a variety of industries. Hardware is becoming cheaper; this further fuels its usage for IoT scenarios. It is also quite clear that traditional methods and tools are not appropriate to load such large volume of data and generate insights from it. They are not made for big data analysis. Availability of such large volumes of data led to innovation in storage of large data. Big data systems, such as Hadoop, have become quite popular for storing large volumes of data, advance analytics, and streaming live data systems then became the focus of innovation. Machine learning, artificial intelligence, and deep learning have taken center stage for finding insights from big data.

Data is characterized by volume, velocity, and varied format. There is a need for a variety of services that can come together to create complete solutions for data that is of huge volume, has high velocity and is available in multiple formats. These services should be enterprise class and scale with high availability, performance-centric, and secure. Azure is a mature platform for handling the complete life cycle of data--from storage to analytics to visualization--all are available with multiple choices for each stage in the life cycle.

In this chapter, we will go into detail about the following:

- Azure SQL
- Azure SQL stretching and sharding
- Azure NoSQL Cosmos DB
- Azure SQL versus Cosmos DB
- Azure data factory
- Azure stream analytics
- Data Lake
- Table storage
- SQL data warehousing

Azure SQL

Azure SQL is a cloud database as a service available in Azure. Microsoft's flagship database technology SQL Server has been available for more than 20 years and Microsoft provides the same on the cloud and provides complete database instances to its users. Azure SQL is a completely managed, relational database service providing higher productivity, faster and easier provisioning, and cheaper alternatives. It provides almost all features of a full-blown SQL Server without any need for managing infrastructure and licenses.

Azure SQL is based on a SQL Server database engine and provides a relational database as its core feature. Data is stored in a tabular format, in rows and columns. A combination of columns forms the schema for the table and rows are stored in a tabular form. Azure SQL is one of the most popular data destinations on Azure. Almost every application needs data to be persisted over time and Azure SQL provides data stores for these applications.

Azure SQL is a transaction based database. It means that it provides support for transactions. There are four major tenets of transactions, also known as **ACID properties**. They are as follows:

- Atomicity
- Consistency
- Isolated
- Durable

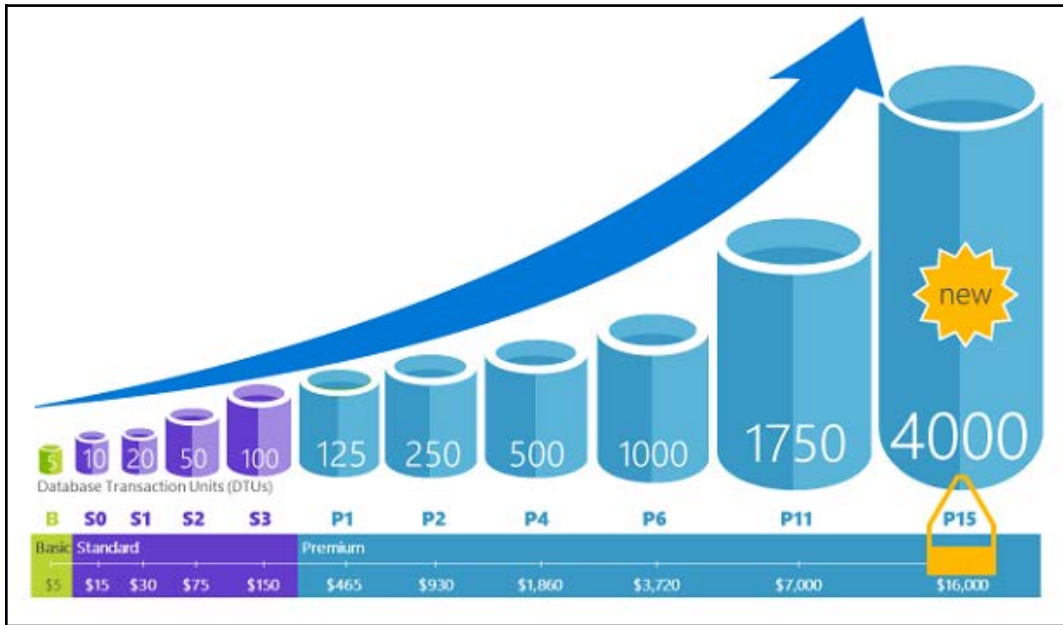
Azure SQL is composed of two important components:

- **Azure SQL logical server:** This is equivalent to a SQL Server instance in the on-premise data center or Azure virtual machines. A server is a physical and logical boundary consisting of multiple databases. It provides a security boundary with its own set of users, groups, and permissions. Users must be authenticated with the server before they can access databases within it. The server provides all the management functions and features for managing and maintaining Azure SQL.
- **Azure SQL databases:** This is the core component of Azure SQL because it is here that the data is stored. Each database is composed of multiple tables with each table consisting of columns and columns together holding rows.

Azure SQL columns provide multiple datatypes to support varied types of data. It includes general datatypes, such as char, varchar, integer, and so on, and more advanced ones, such as JSON, XML, and Spatial. Tables can be normalized such that both writing and reading data can be extremely fast, however it adds redundancy of data. It also allows the creation of multiple types of indexes on columns--clustered, non-clustered, covering, unique, and so on to improve the performance of queries.

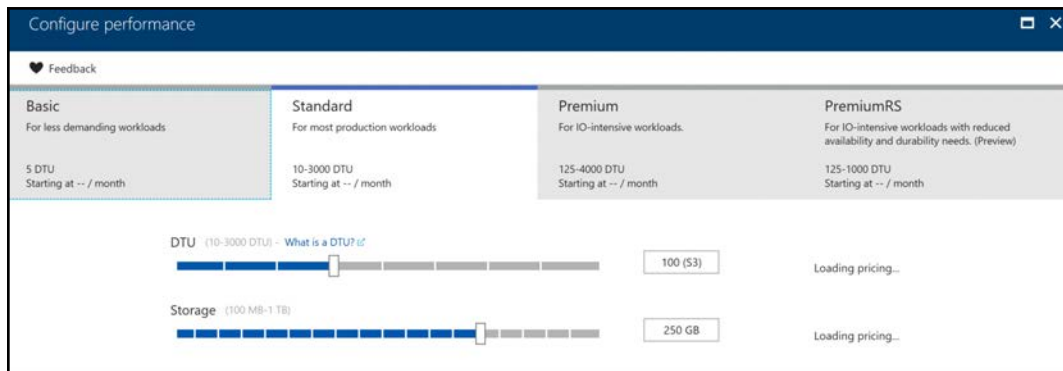
It provides **Transact-SQL** as a language to issue data manipulation and data definition statements. The SQL data engine takes the heavy lifting of parsing, tokenizing, storing, finding an optimal execution plan, and actual execution of queries. It also interacts with the storage engine to fetch data from storage to memory and vice-versa. It provides and implements transactional capabilities and provides management features, such as logs, jobs, agents, backup, restore, and always-on.

Azure SQL performance is measured in **Database Transaction Units (DTU)**. For Azure to guarantee a minimum performance **service level agreement (SLA)**, a combination of compute, memory, and storage resources are calculated together to form DTU. Each DTU confirms and guarantees certain levels of performance.



Azure SQL provides four levels of performance. They are as follows:

- **Basic:** This is for the less demanding workloads providing 5 DTUs of performance
- **Standard:** This is for general production workloads and provides anything between 10 to 3,000 DTUs
- **Premium:** This is for IO-intensive workloads providing DTUs between 125 to 4,000
- **PremiumRS:** This is again for IO-intensive workloads but with reduced availability and durability and provides between 125 to 1,000 DTUs



Users should decide the best DTU needs based on their application requirements. They can start with low DTUs and eventually scale up with increased demand for their application. Scaling of DTU adds more compute, memory, and IOPS resources without stopping database. The scale up and down happens transparently without affecting the availability of the database. Storage can also be scaled up and down based on requirements.

Azure SQL Availability

Azure ensures that data is highly available. Even in case of disaster or hardware failure, data is still available, and it provides multiple capabilities to ensure high-availability.

Azure SQL performs automatic backup of the user's database. These databases are backed up in a manner to minimize the time to recover in case of a disaster. It performs a full weekly backup, hourly differential backups, and backup of transaction logs every five minutes. These backups are then stored in a geo-redundant storage which keeps multiple copies across geographies. In case of a disaster, Azure support can help get these backups and restore them.

Azure SQL also provides geo-replication of databases with up to four readable secondary databases in different regions. All these primary and secondary databases are continuously synchronized using asynchronous replication. It also provides capability for auto-failover to another database in another region.

Azure SQL provides point-in-time recovery and restores databases as well.

Azure SQL Security

Azure provides advance secure features to ensure that data is never comprised.

Azure SQL transparently encrypts and decrypts all data, backups, log files using **Transparent Data Encryption (TDE)** automatically. Applications are not aware of this behind the scene encryption and decryption. They use data as they do normally, however, data at rest in a SQL Server cannot be comprehended by anyone having access to them using database tools. The encryption keys can be managed and stored in the Azure key vault.

Azure SQL also takes care of protecting data in flight that is getting transmitted over a network. It provides an **Always Encrypted** feature that keeps data encrypted even while in storage at rest, during query processing, and on network.

Another great security feature provided by Azure SQL is **dynamic data masking**. It helps in exposing data to non-privileged users by hiding sensitive data in query results. This also works transparently without changing the application.

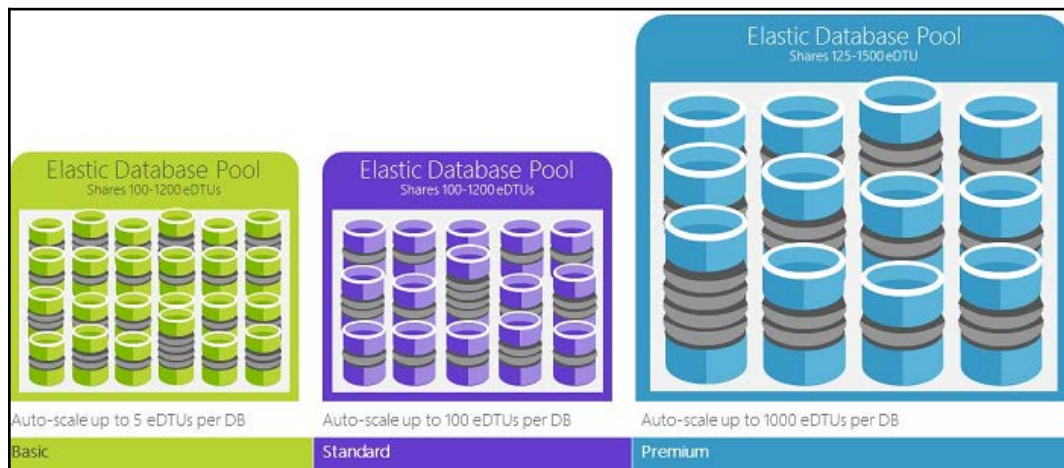
Azure SQL users and groups can be managed using the Azure **Active Directory (AD)**. Users from Azure active directory can be providing access to server and database and this simplifies the overall management of identities by centralizing and out sourcing it.

Azure SQL security goes down from the server to rows in tables. It provides row-level security wherein users can access only those rows that they are authorized to.

It also allows access to those IP addresses that are whitelisted explicitly by the administrator. It also provides advance auditability in terms of who is accessing the server and database logging for every writing and reading data activity.

Elastic Pools

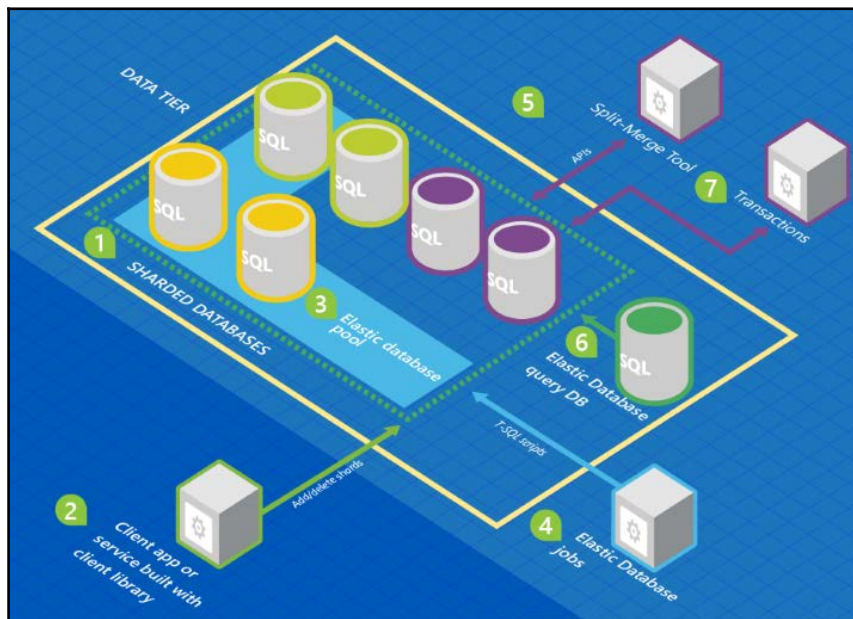
Azure DTUs are great to guarantee a minimum level of performance and resource consumption, however, Azure SQL dedicated the resources for a given DTU for a SQL database and remains the same even if the resources are under-utilized. In case, there are large numbers of databases, each database will have its own dedicated DTU, although some of the databases might not be utilized at all or remain under-utilized. Azure provides a better alternative for such scenarios of having a large number of databases. Azure provides elastic pools that help in hosting multiple databases on the same SQL Server and puts them on the same pool. Each pool has dedicated DTUs assigned to them known as **eDTU**. Within this pool there might be databases that consume more resources than the estimated, while there are others that do not consume many resources. An elastic pool helps in aggregating the DTUs utilized by each of these databases in the pool and ensures that over-utilized databases are not throttled and under-utilized databases are not over charged. The cost of eDTU gets amortized over multiple databases in spite of some databases having a sudden jump in demand. Elastic pools help in reducing the overall cost for multiple databases while ensuring that demanding databases get more resources in times of need.



Azure SQL Scaling Out

Azure SQL is a versatile platform and provides features to cater to different needs. Azure scale out is one such feature, also known as **sharding**. Imagine a situation of having a very large database containing millions of rows. The performance of queries is degrading because of a large number of writes and reads have to traverse through large recordsets. It is a better design to break this large database into multiple smaller databases and distribute data among them. However, to write such a controller or manager is a difficult and herculean task. Azure provides scaling out features to cater to this need of horizontally partitioning data between multiple databases. Client applications connect to the primary Azure database that is responsible for querying data from appropriate databases, collating the information and send it back to the client application. The same process is followed for write operations as well. Azure SQL also provides the facility to merge back these shards into one database.

The table must have partitioned horizontally based on pre-determined keys and the primary database has to maintain details about the range of keys with each database. There are additional management tasks that are also performed by this primary database. The image shown next shows a typical sharding strategy. Each color shows a database that has been partitioned. While some of them are elastic pools, the rest of them consume their dedicated DTUs.



Azure portal does not provide the necessary user interface to shard databases. The scaling out and sharding is done using REST API, PowerShell, or any programming language consuming the Azure SQL SDK and elastic database client library.



The tools for managing sharding are available at <https://docs.microsoft.com/en-us/azure/sql-database/sql-database-elastic-scale-get-started>.

These tools consume Azure SQL SDK and elastic database client libraries to provide different options to manage the process of sharding and merging.

It provides options for splitting and merging databases, creating a new shard, adding databases to shards, executing queries against all shards, and deleting the shard. The previous image shows these client applications interacting with Azure SQL.

Stream Analytics

Stream analytics is a fully managed data and event processing engine that helps in providing real-time analytics on the data getting streamed. While Cosmos DB and Azure SQL provide data that is already available in their storage, stream analytics provides analytics computation on data that is getting streamed live as they are ingested. They are not queries against data stored in permanent storage. This is an extremely powerful capability for getting real-time insights instead of finding insights much after the event occurred.

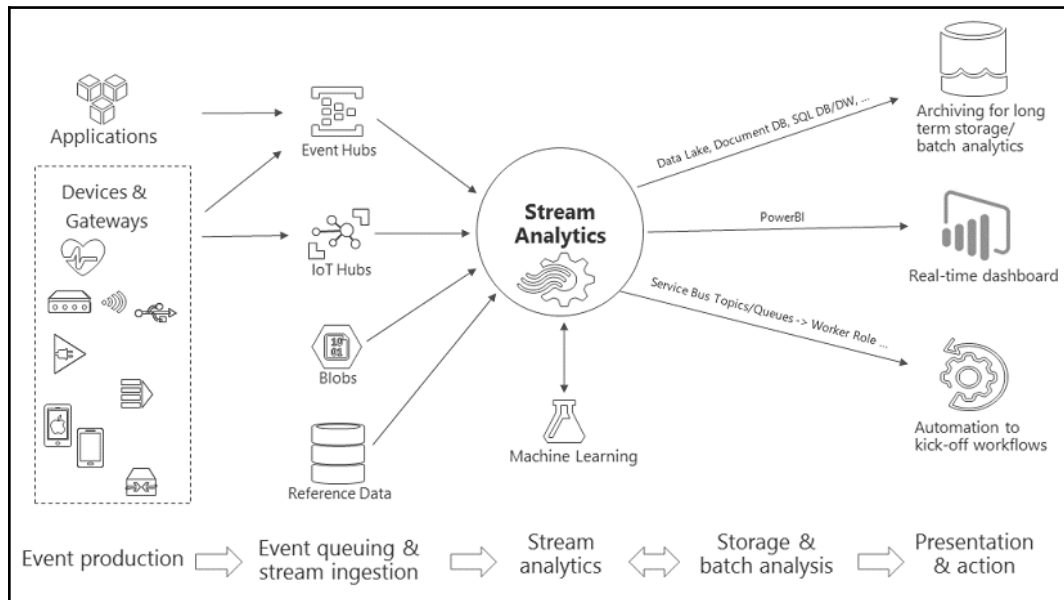
The beauty of stream analytics lies in the fact that the data can come from anywhere--applications, IoT devices, other Azure services, such as event hubs or any other system.

The other fundamental core tenet of stream analytics is the processing of a large volume of data. It has been created to process a huge quantity of data and extract insights and information out of it in real-time.

It also allows necessary integration with other systems and services to act on the extracted insights, such as sending notifications, raising alerts, logging into multiple log analytics services, and reporting using visualization tools.

The crux of stream analytic lies in the fact that data of a high volume, velocity, and multiple format can be processed by a single service in real-time and means actions can be taken on the same.

The architecture of stream analytics is shown in the following diagram:



Stream analytics provides **Extract-Transform-Load (ETL)** functionality for real-time processes. It defines a job that is configured with incoming data along with source of data, transformations, and the output data log with destinations.

There are many use cases for implementing stream analytics. They are as follows:

- Many industries need real-time analytic solutions, such as parking system, highway toll-booth systems, movie ticket sales, and so on
- Social media sentiment analysis
- Web traffic analysis
- Log analytics for security and other breaches and more
- Infrastructure monitoring

There are multiple components that are part of the Azure stream analytics service.

Data Sources

Stream analytics works on data. Data is extracted from the source location and after transformation, data is loaded into the target location. Both the source and destination locations are termed as data sources in Azure analytics parlance. There are data sources, such as Azure event hubs and IoT hubs that can provide data to Azure analytics. There can be other data sources, such as IoT devices that can send data directly to stream analytics and ones that are on-premise or on different cloud platforms. The data sources could either be a place where the data or the event is generated or it could be from storage, such as Azure storage, IoT hubs, and so on.

Data Integration

Data sources are providers and receivers of data. However, the glue that helps connecting Azure stream analytics with them is data integration. Azure stream analytics provides integration for both source and destination data sources. This integration queries data within the specified time-range and provides them to the stream analytics engine for processing.

Data Transformation

Incoming data is processed to gather insights. This process of transformation involves data to be filtered, augmented, and enriched. Not every data that comes in might be of interest. There might be data that is incorrect or missing. To ensure that insights are not biased and skewed, data needs to be brought into a stage that will provide accurate results. This is the job of transformation. Azure stream analytics provides SQL type query language that helps in grouping, aggregation, joining, and filtering data according to business requirements.

Stream Analytics Engine

This is the core component of stream analytics. It is here that the stream analytics jobs are executed and every component interacts together to ETL the data. This engine keeps running and continuously executes jobs over time-series windows to ensure that real-time insights can be gathered.

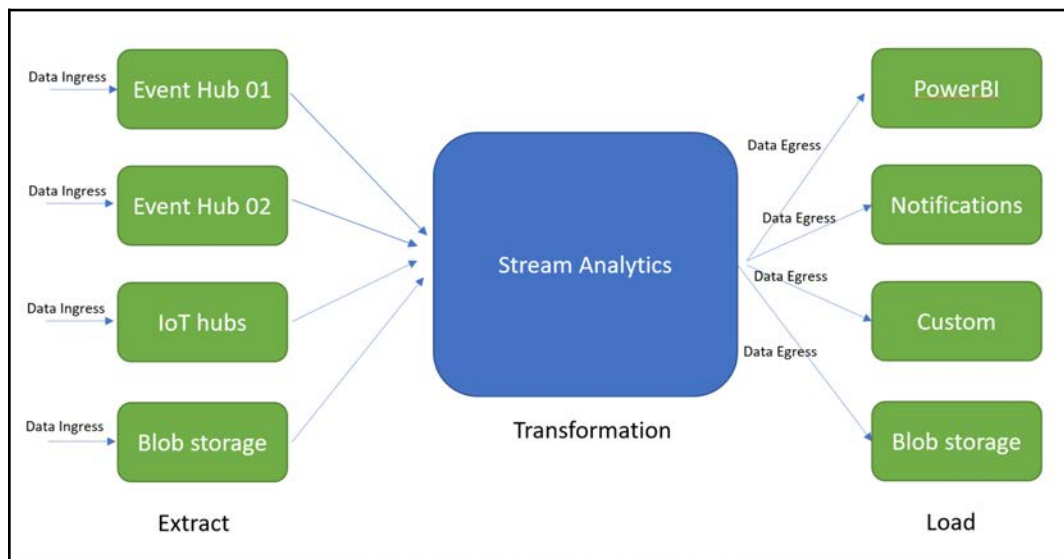
Storage and Presentation

The analytics engine can also help act on the gathered insights. It can send data to durable storage to Azure blob storage or present data using visualization tools, such as Power BI or use it to send notifications.

Architecture

Azure stream analytics is a mature platform and service for real-time analytics. It provides features for multiple inputs and outputs. In the image shown next, there are multiple **Event Hub**, **IoT hubs**, and **Blob storage** as input data sources. These can be joined together, grouped, and aggregated within the same **Stream Analytics** job and insights generated.

There is the **Data Egress** (out-going) cost in Azure, however, the cost is not applicable within a region. If data transfer happens within a region, the egress cost is not applicable. It is applicable when data transfer happens between regions. It is for this reason, as a good practice data sources in Azure should ideally be located within the same region as that of Azure stream analytics.



On the Azure portal, the Azure stream analytics dashboard provides the number of inputs and outputs.



Azure Data Factory

Data is generally stored in flat files, relational databases, NoSQL databases and other locations in multiple formats, such as JSON, XML, CSV, text files, and binary.

Moreover, each data store defines its own model to store data. An employee ID in one data store is named `EMPIDID` in another data store and `EID` in the third data store within the same organization. There are multiple facets for the same data within different data stores. Azure data factory is a data integration service that helps in authoring orchestration and workflow scenarios for the ETL of data. Azure data factory works with data from any location-cloud, on-premise, and works at the cloud scale. It works with large volumes of data with varied formats.

The workflows in Azure data factory are known as data pipelines and Azure data factory helps in creating, deploying, monitoring, and scheduling these data pipelines easily. Azure data factory is a managed PaaS service providing high availability, scalability, and fault tolerance to data pipelines.

Azure data factory can ingest from multiple locations and load the final transformed data to multiple different target locations and types. From the onset, data factory looks like just another ETL service, however, there is a big difference. The main difference between data factory and other ETL tools is based on **transformation**. The transformation executed by data factory is based on compute services, such as HDInsight Hadoop, Azure Data Lake analytics, Spark, and Azure machine learning, whereas the transformation executed by other ETL tools are related to transformation of rows and columns, filtering and aggregation of rows and column, and enrichment of data. Azure data factory prepares the data by filtering, aggregating, grouping, joining, and enriching data, and then applies transformation using compute services. The output of this transformation are insights, information that can further be consumed by visualization tools, such as Power BI and other custom applications.

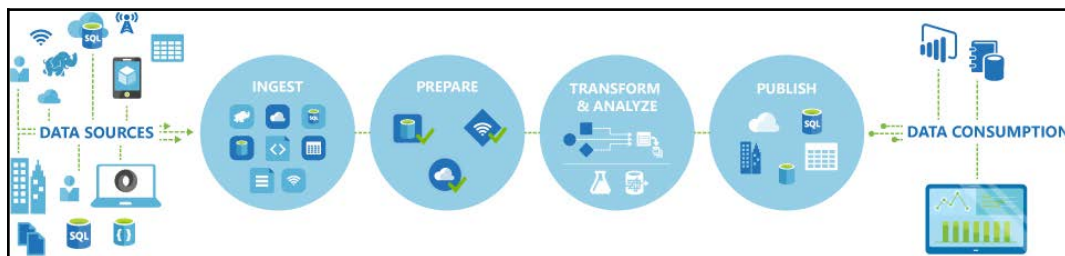
Azure data factory provides connectors for almost every major data store including SAP, Oracle, MongoDB, Cassandra, and supports multiple types of protocols and drivers for connectivity to these data stores including ODBC, OLEDB, ODATA, and HTTP.

The data factory pipelines can be scheduled to run daily, hourly, and weekly.

The main difference between Azure data factory and Azure stream analytics is that while Azure data factory works on a snapshot of data that is already available on durable storage, Azure stream analytics works on transient real-time data that is processed as and when it is generated.

There are two versions of Azure data factory--V1 and V2. V2 is currently in preview and only available in East US and East US 2 regions. V1 is generally available and is available in East US, West US, West Central US, and North Europe. As a good practice, data sources in Azure should ideally be located within the same region as that of the Azure data factory.

The architecture of Azure data factory is shown in the following image:



There are multiple components that are part of the data factory service.

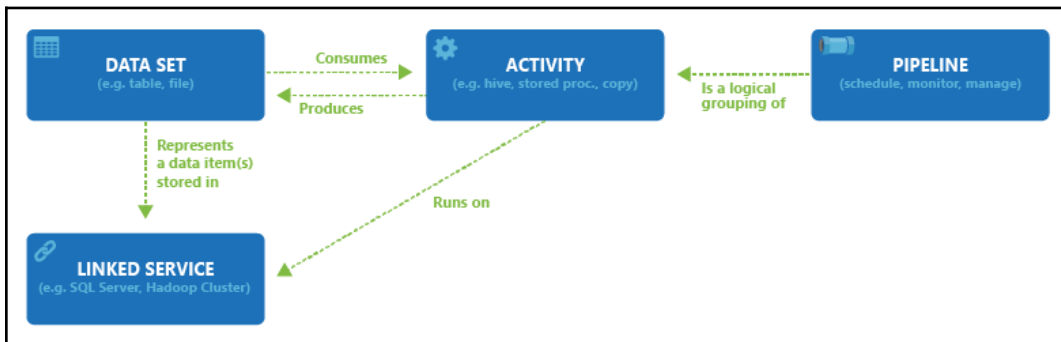
Data Sources

Data should first be ingested within the data factory. Data can be either be on cloud or on-premise. The data needs to move within Azure data factory. Azure data factory provides linked services and dataset constructs for connecting and fetching data into the data factory.

Datasets refer to actual data stored at the data source or data location. It could be a comma separated value file on the on-premise server or a file in a container in the Azure storage account as blob or in the Azure SQL. The possibilities are endless. Linked services refer to the process of connecting to the data source by using a data source specific connection string and using it to fetch datasets into the data factory. For example, if you are copying data from the Azure blob file to Azure SQL, two datasets, and two linked services are needed. One linked service would consist of the connection string to the Azure storage account while the other would contain the connection string for Azure SQL. The dataset related to Azure blob would consist of files containing the data, and the dataset related to Azure SQL would consist of tables where data would be written. The data pipeline uses activities (for example, copy activity) to connect to data sources using linked services and copy datasets from source location to data factory.

Data factory consists of multiple pipelines and each pipeline consists of activities. Pipelines are made up of multiple activities where each activity is responsible for executing a single responsibility. For example, copy activity is responsible for copying data from a data location using linked services and datasets to another data location using target linked services and datasets.

The relationship between activities, pipeline, linked services, and dataset is shown next.





The dataset types available in the data factory are available at <https://docs.microsoft.com/en-us/azure/data-factory/v1/data-factory-create-datasets#Type>.

Azure data factory pipelines can be executed manually or scheduled to run periodically.

Data Transformation

Now that the data is available in the data store, it can be processed using compute services, such as Azure HDInsight Hadoop, Spark, Data Lake analytics, and machine learning. The data output from transformations can be published for further consumption.

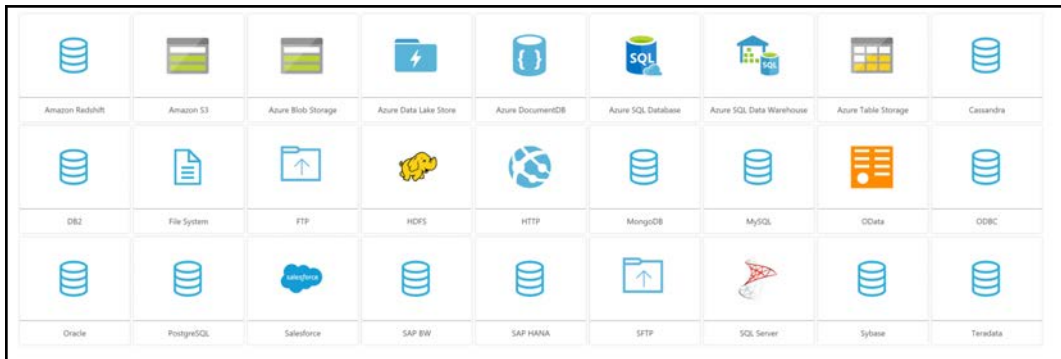
Publish and Presentation

The transformation output can be stored in durable storage, such as Cosmos DB, Azure SQL, Azure data warehouse, or any custom analytics tools. Also, Power BI can be connected to the durable storage for creating reports, dashboards, and visualization.

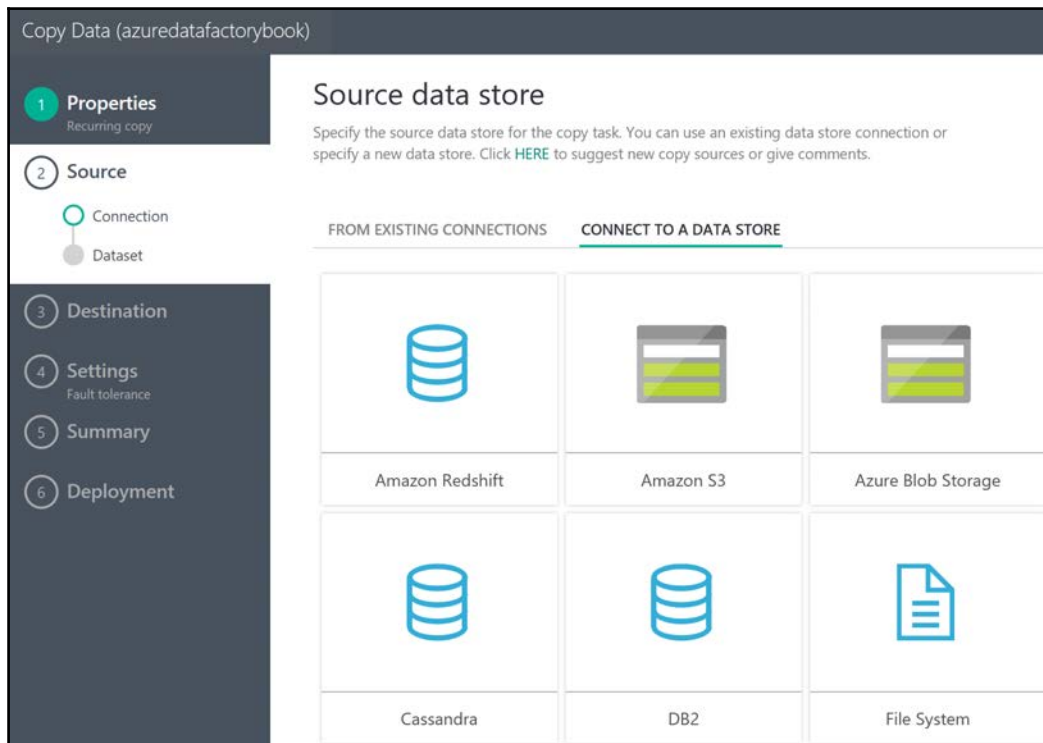
Using Data Factory

This section will show step-by-step details about how to use Azure data factory to create datasets, linked services and pipelines, and transfer data from one storage account to another.

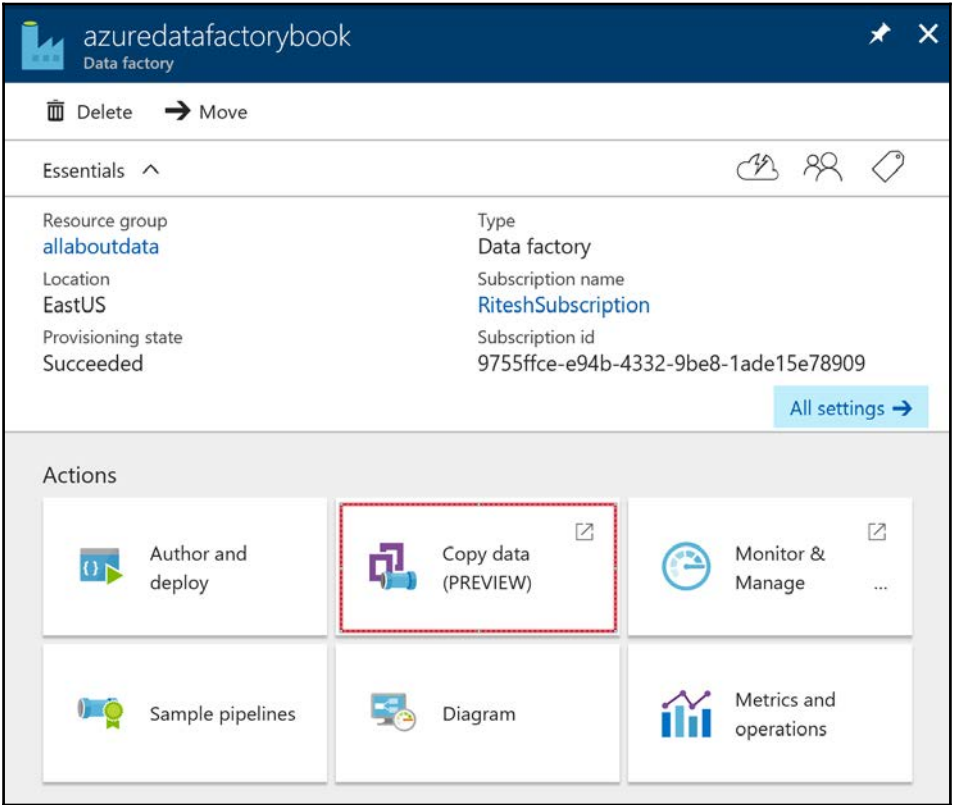
Data factory provides numerous data stores from which data can be read and transferred to. Data factory provides enormous flexibility in terms of choosing a source and target data store. The next screenshot shows the number and type of data stores available in the data factory. You can choose any of them as a source and target.



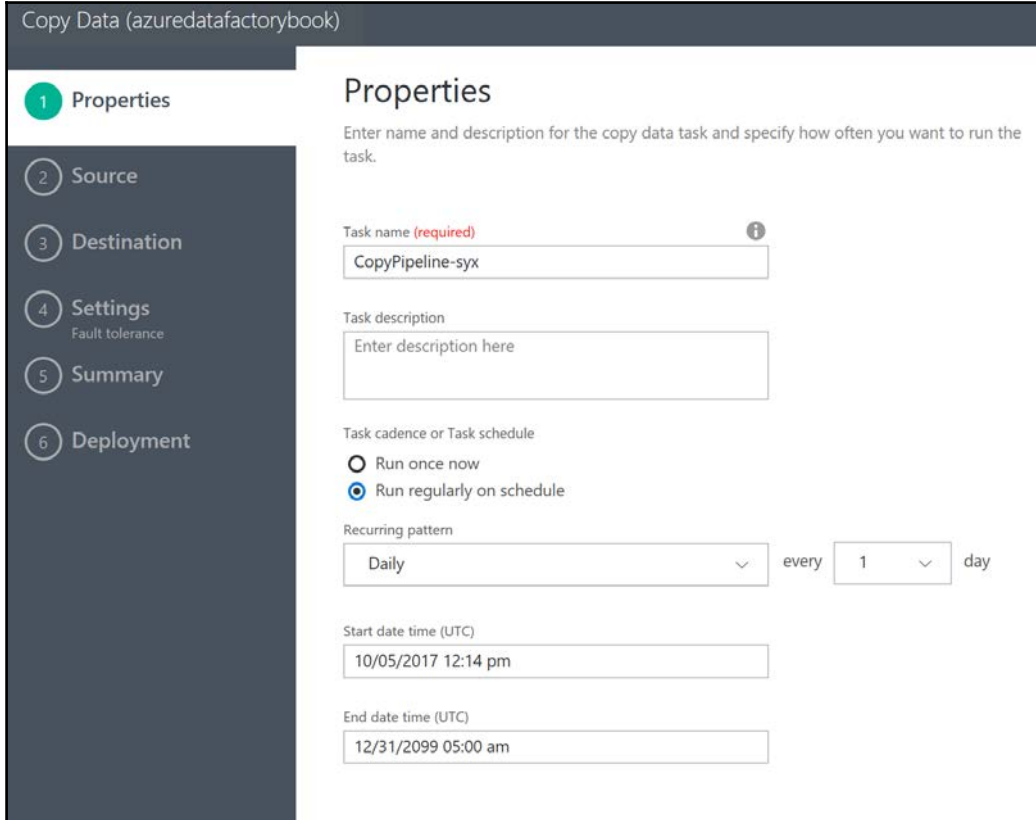
Azure data factory also provides a wizard interface for identifying a source data store and its data along with target data store and data. The same is shown in the next screenshot:



1. The first step is to create a data factory resource. After creation, click on the **Copy data** button:



2. This will open a browser window and navigate to <https://datafactory.azure.com/>. This URL provides a wizard-based user interface to configure the data factory:



Copy Data (azuredatabafactorybook)

1 Properties

2 Source

3 Destination

4 Settings
Fault tolerance

5 Summary

6 Deployment

Properties

Enter name and description for the copy data task and specify how often you want to run the task.

Task name (required) i
CopyPipeline-syx

Task description
Enter description here

Task cadence or Task schedule
☐ Run once now
☒ Run regularly on schedule

Recurring pattern
Daily every 1 day

Start date time (UTC)
10/05/2017 12:14 pm

End date time (UTC)
12/31/2099 05:00 am

3. Fill in all the details, such as the name of the task, scheduling information, recurring pattern, start date, and end date. Click on **Next**.

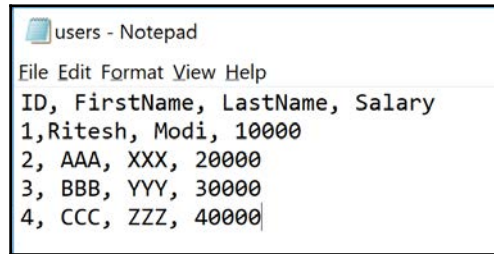
4. Select **Azure Blob Storage** as the source data store. This could be any data source and the next step that the wizard will ask for is the value for constructing the connection string and linked service to connect to it. This is shown in the next screenshot. Click on the **Next** button:

The screenshot shows the 'Specify the Azure Blob storage account' step in the Azure Data Factory wizard. The left sidebar contains a navigation pane with steps 1 through 6: Properties (Recurring copy), Source, Destination, Settings (Fault tolerance), Summary, and Deployment. The 'Source' step is selected, and a green checkmark indicates the 'Connection' is complete, while the 'Dataset' is still pending. The main content area has the title 'Specify the Azure Blob storage account' and contains four required fields: 'Connection name' (text input with 'Source-BlobStorage-syx'), 'Account selection method' (dropdown menu with 'From Azure subscriptions'), 'Azure subscription' (dropdown menu with 'Select all'), and 'Storage account name' (dropdown menu with 'incomingfiles'). Each field has a red '(required)' label and an information icon.

5. The next step asks the user to select the file in the source data store. This activity creates a dataset for the data factory:

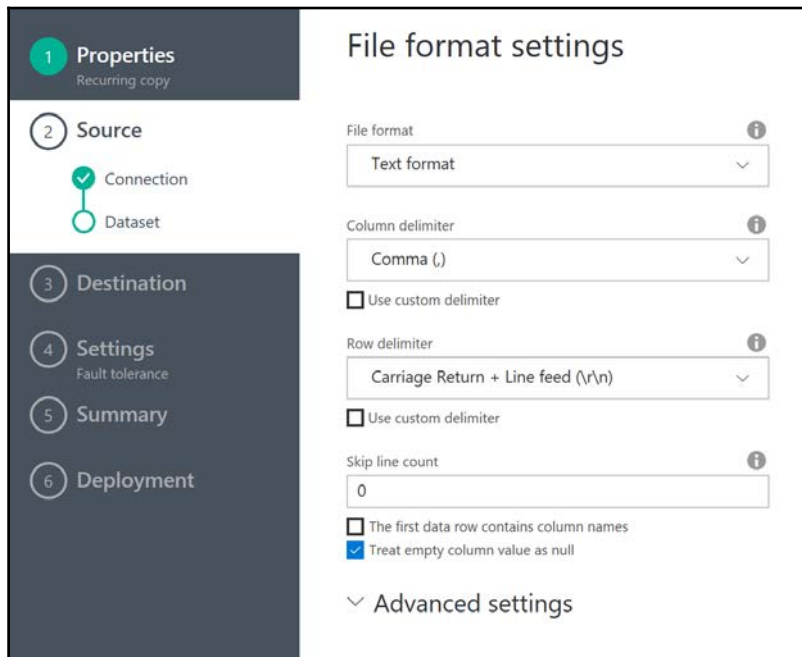
The screenshot shows the 'Choose the input file or folder' step in the Azure Data Factory wizard. The left sidebar is identical to the previous screenshot, with 'Source' selected and 'Connection' completed. The main content area has the title 'Choose the input file or folder' and a subtitle 'Select or specify a source folder or file to be copied to the destination data store.' Below this is a 'File or folder' text input field. A list of files and folders is displayed below the input field: 'orders>' (with an upward arrow icon), 'cChoco.zip' (with a zip file icon), and 'users.csv' (with a CSV file icon). The 'users.csv' file is highlighted with a dark blue background.

6. The `users.csv` file is a simple file containing some representative data however, in a real-life scenario, millions and billions of records with terabytes of data might be available.
7. The `users.csv` looks as shown next. It has a header as the first row and the rest of the rows contain comma-separated data. Choose the compression type if the file is already compressed.



```
File Edit Format View Help
ID, FirstName, LastName, Salary
1,Ritesh, Modi, 10000
2, AAA, XXX, 20000
3, BBB, YYY, 30000
4, CCC, ZZZ, 40000
```

8. Clicking on **Next** will identify the file format and show that to the user. It is here where the user can change the settings for the file format. This is shown in the next screenshot:



1 Properties
Recurring copy

2 Source
Connection
Dataset

3 Destination

4 Settings
Fault tolerance

5 Summary

6 Deployment

File format settings

File format 
Text format

Column delimiter 
Comma (,)
☐ Use custom delimiter

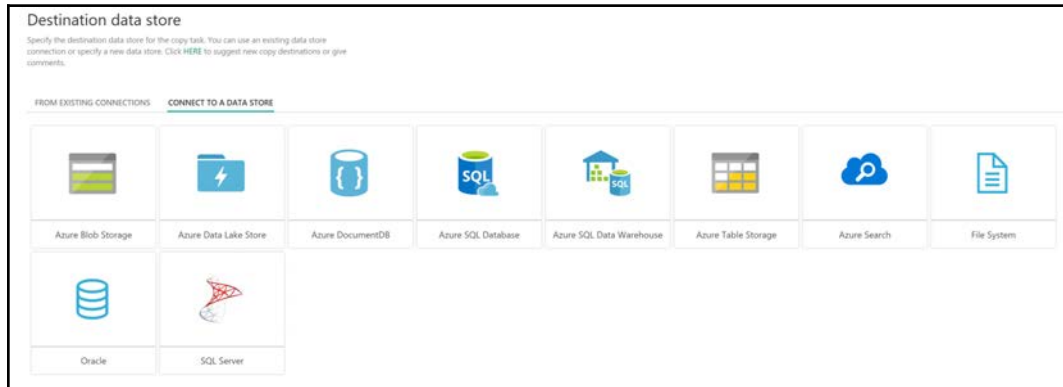
Row delimiter 
Carriage Return + Line feed (\r\n)
☐ Use custom delimiter

Skip line count 
0

☐ The first data row contains column names
☒ Treat empty column value as null

Advanced settings

9. Clicking on **Next**, it will ask for the **Destination data store** configuration. This is similar to the source data store. Choose an appropriate target data store as shown next:



10. In our sample, we are selecting **Azure Blob Storage** but users are advised to play with all kinds of data stores, such as **Azure Data Lake Store**, **Azure DocumentDB**, **Azure SQL Data Warehouse**, **Azure Table Storage**, **Azure Search**, **Oracle**, and **SQL Server**.
11. Specify the connection string details for the target data store as shown next and then click on **Next**.

Specify the Azure Blob storage account

Connection name *(required)* ⓘ
Destination-BlobStorage-syx

Account selection method *(required)* ⓘ
From Azure subscriptions ▼

Azure subscription *(required)* ⓘ
Select all ▼

Storage account name *(required)*
test2451835d ▼

12. Provide appropriate details for the target dataset, such as container name and file name as shown in the next screenshot and click on **Next**:

Choose the output file or folder

Specify a folder that will contain output files or a specific output file in the destination data store.

Folder path

orders

Filename

users.csv

You can use variables in the folder path to copy data from a folder that is determined at runtime. Make sure that you select a folder with that structure using the Browse button first. The supported variables are: {year}, {month}, {day}, {hour}, {minute} and {custom}. See [Data Movement Activities](#) article for details about these variables.
Example: inputfolder/{year}/{month}/{day}.

Compression type

None

Copy behavior

Merge files

13. Configure the **File format** setting for the destination data store as shown in the next screenshot and click on **Next**:

File format settings

File format 

Text format 

Column delimiter 

Comma (,) 

☐ Use custom delimiter

Row delimiter 

Carriage Return + Line feed (\r\n) 

☐ Use custom delimiter

☐ Add header to file 

^ Advanced settings

Escape character 

Quote character 

Null value 

\N

Encoding name 

14. Configure the tolerance and performance (using parallel writes) settings as shown next and click **Next**:

1 Properties
Recurring copy

2 Source
Azure Blob Storage

3 Destination
Azure Blob Storage

4 Settings

5 Summary

6 Deployment

Settings

More options for data movement

^ Fault tolerance settings

Error handling for incompatible rows between source and destination ⓘ

Actions

Abort copy activity on first incompatible row (cc) ▾

^ Performance settings

^ Advanced settings

Parallel copy ⓘ

Cloud units

Auto ▾

15. Validate all the details as shown in the **Summary** window. This is shown in the next screenshot, then click on **Next**:

Summary

You are running scheduled pipeline to copy data from Azure Blob Storage to Azure Blob Storage.

**Azure Blob Storage orders**
Region: West Central US

Copy Run Time Region: South Central US

**Azure Blob Storage orders**
Region: South Central US

Properties

Task name

CopyPipeline-syx

Task description

<no description has been provided>

Task cadence

Daily, every 1 day between Thu, 05 Oct 2017 12:14:16 GMT and Thu, 31 Dec 2099 05:00:00 GMT

Edit

Source

Connection

Account: incomingfiles

Connection name

Source-BlobStorage-syx

Dataset name

InputDataset-syx

Blob path

orders

Region

West Central US

Minimum size in MB

Incremental updates

Not enabled

Data delay

00:00:00

Maximum retry

Retry interval

00:00:00

Retry timeout

00:00:00

Edit

Destination

Connection

Account: test2451835d

Connection name

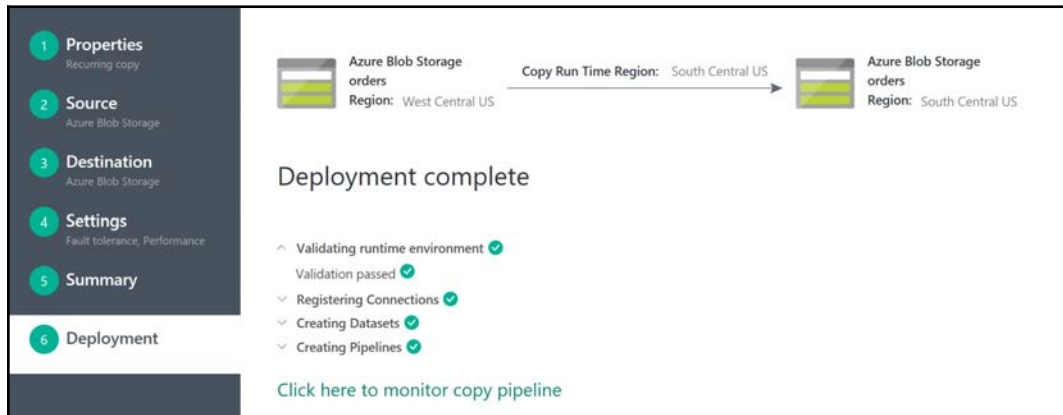
Destination-BlobStorage-syx

Edit

Previous

Next

16. This will start the deployment process. The next screenshot shows that the deployment is successful and the data was transferred from the **West Central US** to the **South Central US** from one **Azure Blob Storage** to another.
17. This will be a scheduled activity if scheduling was configured earlier.



Azure Data Lake

Azure Data Lake is an Azure hosted service providing end-to-end services for storing big data, transforming it, and performing analysis on top of it. It comprises multiple services:

- Azure Data Lake store
- Azure Data Lake analytics

It provides enterprise grade scalability, performance, and security for data storage and processing. It helps in the massive execution of parallel workloads and is a much-sought service for data scientists and analysts.

Azure Data Lake store

Azure Data Lake store provides storage for big data. It provides petabyte scale of storage and is highly optimized for storing data that has high volume, velocity, and comprises multiple types and format. Data of various format and size can be co-stored within the same Data Lake.

Traditional storage, such as filesystems, relational databases, and other storage are general purpose storage accounts and are not suitable for storage of big data that is accessed for analytics purposes. The repository should be optimized for faster and efficient querying and should be able to store data in raw as well as processed format. Azure Data Lake has been created specifically for this purpose.

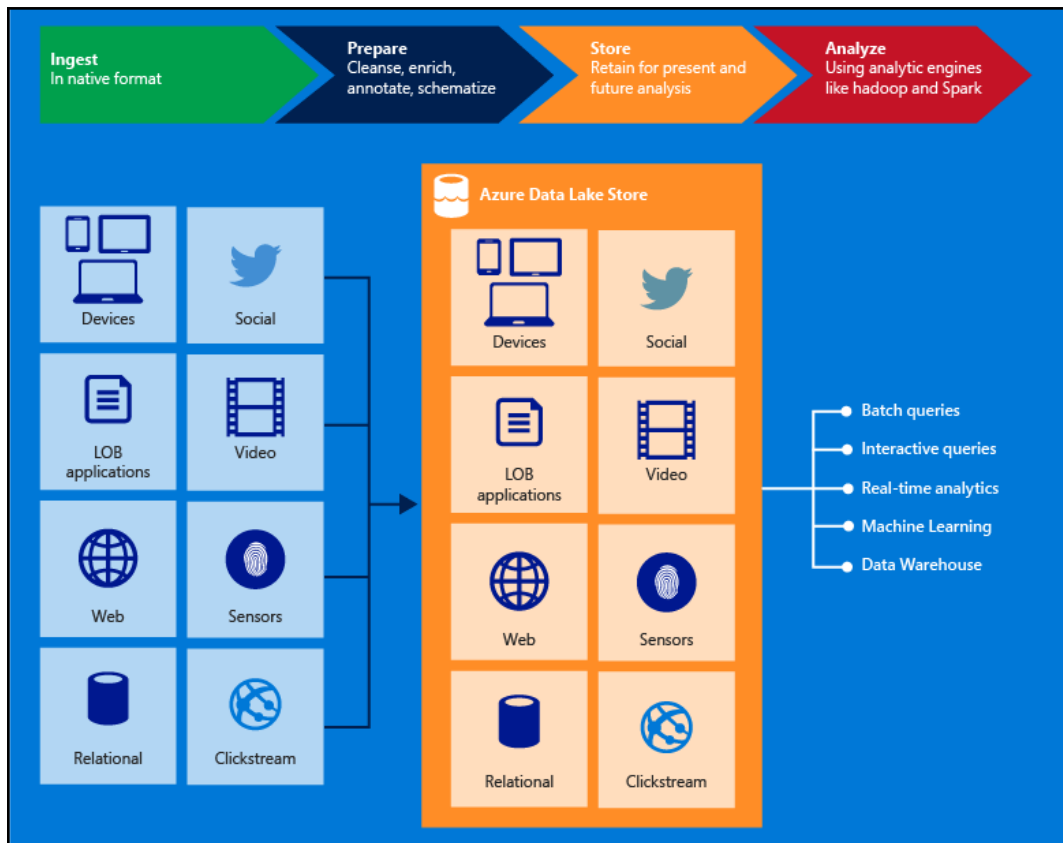
Azure Data Lake is accessible by a variety of tools and services including HDInsight Hadoop, Azure machine learning, stream analytics, data factory, event hubs, and more.

Data comes from a variety of sources and devices, such as social media, IoT devices, line of business applications, relational and non-relational databases, web scraping, and many more.

It is currently available only at three Azure locations--East US 2, North Europe, and Central US.

Eventually, Azure Data Lake is a hierarchical file-based storage just as a Windows, Linux, or Hadoop filesystem that has been optimized for efficient querying and storage for big data workloads.

The data in Azure Data Lake is highly available as it is replicated into multiple replicas and stored in different data centers.



Data Lake Security

Security aspects are supremely important when dealing with data. Azure Data Lake is a secure repository, access to which is managed by Azure AD. Users, groups, and service applications that are provisioned and enabled in Azure AD are allowed to access the Azure Data Lake store. Moreover, after authentication the identity should have adequate permissions to perform activities on the Azure Data Lake store. **Role-Based Access Control (RBAC)** from **Azure Resource Manager (ARM)** is responsible for enforcing permission checks and authorization for an identity. There is additional authorization security available in terms of **Access Control Lists (ACLs)**. The ACLs can be applied with read, write, and execute permissions of folders, sub-folders, and files.

Similar to Azure SQL, Data Lake store can only be accessed by those IP addresses that are explicitly allowed by whitelisting them in the Data Lake store configuration. Since, Azure Data Lake is frequently accessed by services in Azure itself, additional configuration is available to provide access to them without knowing their IP addresses.

Data stored in Azure Data Lake store can be encrypted. Data at rest can be in an encrypted form, such that it cannot be read by anyone having access to it. Similar to SQL Azure TDE, the data in Data Lake store can be encrypted before storing and decrypted after retrieval transparently, without the application knowing about it. It is good practice to keep data encrypted at rest. The encryption and decryption keys can be stored in the Azure key vault.

Data Lake Performance

Performance is another very important aspect for solutions based on Data Lake store. Since Data Lake store is a repository, it should be storage intensive providing high throughput in terms of higher IOPS for reads and writes. Traditional storages does not consume the entire throughput available from the storage and hardware. Instead, Azure Data Lake store provides higher throughput by parallelizing the access to data and consuming the bandwidth available from hardware.

It is to be noted that for end-to-end solutions, it is just not the performance of Azure Data Lake store that is important, in fact the performance of all data sources from where data is ingested into Azure Data Lake store is also important along with services that accesses data for analytical purposes.

Azure Data Lake Analytics

Azure Data Lake analytics is one of the flagship services from Azure to execute analytics jobs. Azure Data Lake stores took care of storing big data in an optimized way. However, for an end-to-end scenario, the data in storage is useful if it is read by a service and that service can perform analytics on it and gain meaningful insights. The analytics service is compute intensive in nature because of deep processing of a large quantity of data. To setup an environment for such a service involves provisioning and configuring virtual machines, tune them for performance and maintain them. Azure Data Lake analytics provides platform as a service capability for running analytics jobs. It means users no more need to think about provisioning and configuring infrastructure and hardware, rather, they can concentrate on writing their analytics job and focus on the business problem.

Azure Data Lake analytics provides its own SQL-style language U-SQL that has a capability to integrate code generally written using programming languages, such as C#. It's a versatile language that can query multiple types of databases, such as Azure SQL, IaaS SQL Server, and Azure SQL data warehouse.

Another important benefit, apart from low maintenance, is that it provides the pay-as-you-go model from a cost perspective. Users pay only when the analytic jobs are running and not otherwise. There is no charge for provisioning Azure Data Lake analytics if it is not running.

Azure Data Lake analytics is a scalable service. It means that when a job executes in Azure Data Lake analytics, it provisions compute and memory resources behind the scene and dynamically keeps scaling out as more demand for resource happens. After the job completes, the resources are scaled down automatically. All this happens transparently without the knowledge of the user.

Azure Data Lake analytics just as Azure Data Lake store, is integrated with Azure AD for authentication purpose. Users, groups, and service applications available in Azure AD can access Azure Data Lake analytics. ARM's RBAC is also applied from an authorization perspective to determine the permissions applicable for users and Azure Data Lake analytics.

Azure SQL Data Warehouse

Azure SQL is a highly scalable, durable, performance centric, highly available service for transactional data. This has traditionally been called **Online Transaction Processing (OLTP)** applications. There is another set of applications popularly known as the **Online Analytics Processing (OLAP)**. Typically, OLAP applications are based on a data warehouse that stores massive amounts of data in a format readily consumable by analytics engine as the **SQL Server Analysis Services (SSAS)**. The data warehouse stores data that is de-normalized for efficient querying.

Data warehouses are categorized into read major data stores. The write to this data store happens periodically contrary to OLTP applications where both writes and reads are comparable. Generally, applications that provide reporting, visualization, and analytics services connect to a data warehouse to load and generate analytical insights out of it.

Date warehouses are also characterized by their massive need for storage and compute power. They generally consist of terabytes and petabytes of data and perform aggregation and grouping of millions and millions of records.

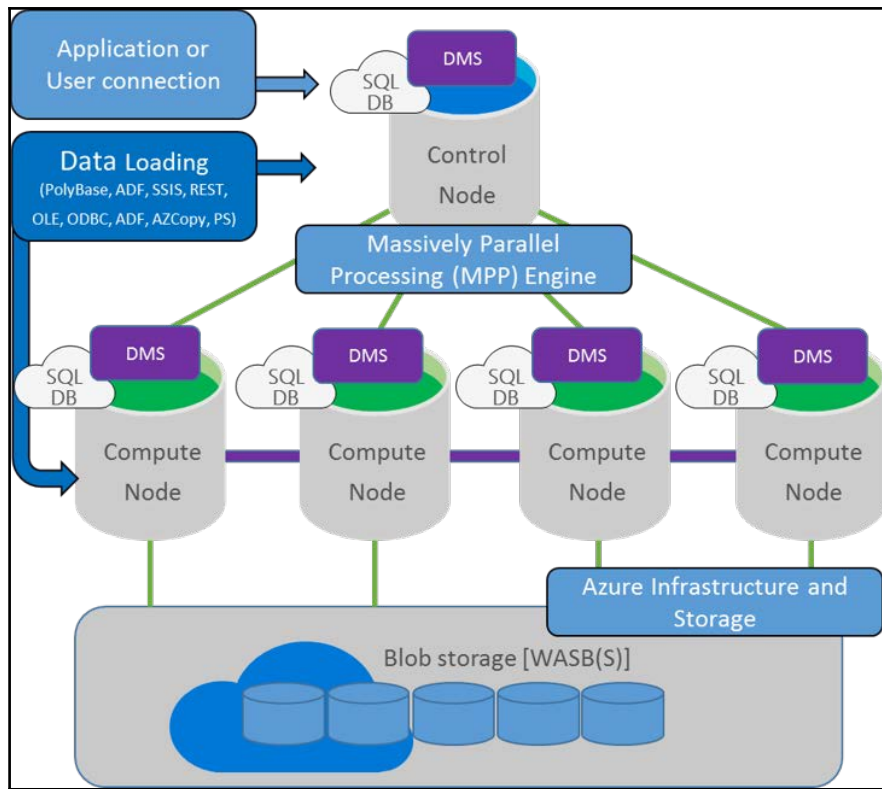
Azure provides SQL data warehouse as a service on the cloud that is massively scalable, has high adjustable performance, is highly available and secure. It is built on top of the Azure SQL Server and databases and so it gets all goodness and features from it. It also stores data into Azure blob storage services that helps in storing a virtually unlimited amount of data into them.

The architecture of Azure SQL data warehouse is shown next. Azure SQL data warehouse comprises of two kinds of nodes:

- **Control node:** This node is responsible for all interaction with users and applications interacting with Azure SQL data warehouse. It is the main brain and engine behind the execution of Azure SQL data warehouse. It is responsible for coordinating with all worker or compute nodes for storage and compute requirements. It also helps in distributing the load between them.
- **Compute nodes:** These nodes are the actual workhorses of Azure SQL data warehouse. They are the ones that store and retrieve data from Azure blob storage and provide compute power by executing SQL statements against the data. They implement a technology called **PolyBase** that helps in executing SQL statements against data stored in the non-relational stores, such as Azure blob storage service.

Both control and compute nodes are based on Azure SQL database and host on the Azure logical SQL Server.

There is another important component called **Data Movement Service (DMS)** available on all nodes in Azure SQL data warehouse irrespective of control and compute nodes. This is the glue that actually helps in the movement of data, combining, and aggregating data from multiple compute nodes, and presenting uniform data to the requestor.



Some of the important characteristics of Azure SQL data warehouse are as follows:

- **Massively Parallel Processing (MPP)** system providing features to implement parallel data warehousing.
- The interaction is based on Transact-SQL. Ability to connect using SQL drivers and tools.
- Available virtually at all Azure locations to help data sovereignty and governance.
- Completely elastic from a storage perspective. Virtually unlimited storage available.
- The compute layer is different from the storage layer. This helps in scaling them independently.

- Provides the querying capacity and power of an SQL Server.
- Provides all benefits of Azure SQL:
 - FIP-based firewall access control list
 - Always on scalability and availability
 - Horizontal sharding
 - Pooling for cost optimization
 - Security

An important aspect of Azure SQL data warehouse is adjustable performance configuration. The performance is defined using a **Data Warehouse Unit (DWU)**. Each DWU provides SLAs in terms of compute, IOPS, and memory. The cost of Azure SQL data warehouse goes higher with a higher configuration for DWU. This is similar to the DTUs provided by Azure SQL. Users can decide the best performance requirements for their application and accordingly adjust the performance scale. There are two different ways performance can be adjusted. They are as follows:

- **Optimized for Elasticity:** According to Azure this separates compute and storage for independent, elastic sizing of resources. It is designed for frequent scaling operations to manage the compute spend and cloud burst scenarios. It offers the lowest starting scale point.
- **Optimized for Compute (Preview):** It is coming soon.

Configure Performance

♥ Feedback

Optimized for Elasticity

Separates compute and storage for independent, elastic sizing of resources. Designed for frequent scaling operations to manage compute spend and cloud "burst" scenarios. Offers lowest starting scale point.

Starting at 99.94 INR / hour

[Learn more about performance tiers](#)

Scale your system

(DW400)

Optimized for Compute (Preview)

Automatically tiers storage to deliver enhanced performance for workloads with consistent compute requirements. Designed for intensive query performance and compute scalability needs.

Starting at 49.97 INR / hour

[Learn more about pricing](#)

Price of your system

399.78 INR / hour

400 DWU

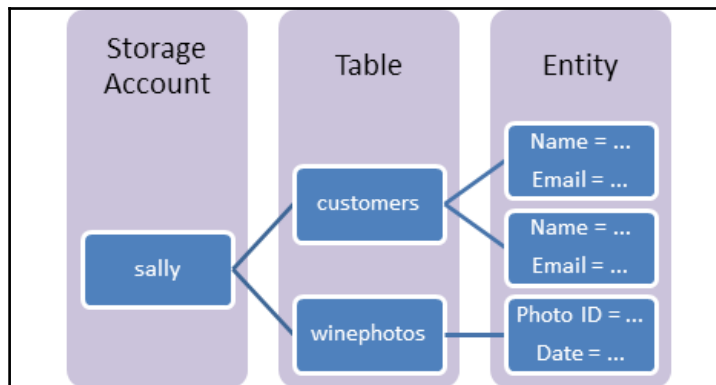
Azure SQL data warehouse can be integrated to a host of services on Azure as the following:

- Machine learning
- Power BI
- Stream analytics
- Data factory

Table Storage

Azure table storage service is available as part of the Azure storage account. Azure storage provides multiple types of storage and table service is one among them. Azure table service is a fully managed, secure, scalable, highly available petabyte-scale storage service that helps in storing data in the NoSQL format. As we know, NoSQL helps in building schema-less design, Azure table service is also schema-less. It helps in storing documents using key-attribute pairs. What this means is that table service does not enforce any schema on entities. Entities can have different properties for each entity at the same time without modification.

Data in Azure tables is stored as entities. Data is stored in tabular format where entities form the rows and properties form individual columns. An entity is very similar to a row in a relational database and each table service consists of multiple entities and each entity consists of a pair of keys that help uniquely identify the entity. The relationship between table service, entities, and the key-value store is shown next. The entities are referred using partition and row keys. The values for both partition and row keys should come from the data itself.



Partitions should be thought of dividing a table into separate logical structure for fast and easy querying. A partition is uniquely identified using storage account name, table name and partition name. It is very important the partition and row keys are decided appropriately. They can be a maximum 1 KB in size each. Their combination should be unique across the table so that any entity can be uniquely identified, and they are also indexed. Partitions help to scale tables into terabytes without sacrificing query performance. It is very similar to sharding in Azure SQL.



Details about designing tables in Azure is available at <https://docs.microsoft.com/en-us/azure/cosmos-db/table-storage-design-guide>.

It is to be noted that table services are part of the storage account. Each general storage account has a maximum limit of 20,000 operations/second. Based on requirements of operations this might be an overkill or over-utilized. If these numbers of operations are not enough, Cosmos DB should be considered. Another aspect is the storage limit in table service which is maxed at 500 TB. For storage greater than 500 TB, Cosmos DB should be considered.

Another constraint of table services is that it allows indexing only on partition and row keys. This makes them less efficient when querying based on other columns in table service. Writing complex queries using joins are difficult with table service.

Table services stores data as properties in tabular format whereas Cosmos DB stores data as JSON-document objects and the whole object can be indexed instead of just a few select columns.

Summary

Data is one of the most crucial elements for a solution. Data has value and they need to be protected and secured. The nature of data has changed over the last few years. Data has grown exponentially in terms of volume, and the velocity of generation of data has also increased manifold. Data is also stored on-premise and on the cloud, in PaaS and IaaS services and there are a ton of drivers and protocols to access them. The data can be in any format and so parsing and understanding data across all formats is a challenge by itself. Azure provides rich services to handle all these newer facets of data. Azure provides Azure SQL, SQL on IaaS, elastic pools, sharding, data factory, stream analytics, Data Lake store and analytics, data warehouse, machine learning, and Power BI to architect end-to-end data solutions from ingestion of data to processing, transformation, loading, and analytics. This chapter was an introduction to these services on Azure and they should be read further using the Azure documentation.

Next chapter will discuss one of most emerging deployment patterns on cloud known as Serverless.

15

Designing and Implementing Serverless Solutions

Serverless is one of the hottest buzzwords in technology these days and everyone wants to ride this bandwagon. Serverless brings a lot of advantages in overall computing, software development processes, infrastructure, and technical implementation. There is a lot going on in the industry--at one end of the spectrum is infrastructure as a service and serverless at the other end. There is PaaS and containers in between. I have met many developers and it seems to me that there exists some level of confusion among them about IaaS, PaaS, containers, and serverless computing. Also, there is much confusion about use cases, applicability, architecture, and implementation for the serverless paradigm. Serverless is a new paradigm that is changing not only technology but also culture and processes within organizations.

This chapter will concentrate on the following topics related to serverless:

- What is serverless architecture?
- What are Azure functions?
- Creating your first Azure function
- Types of Azure functions
- Creating a connected function
- Creating an event-driven function
- Creating a solution comprising of multiple functions

In this chapter, we will go into detail about the following:

- Azure billing
- Invoicing

- Usage and quotas
- Usage and billing API
- Azure pricing calculator
- Best practices

A Short History of Serverless

Before we understand serverless, its architecture, and implementation, it is important to understand its history and how it evolved.

At the beginning there were physical servers. Although users had complete control over physical servers, there were lots of disadvantages:

- Long gestation period between ordering and actual deployment of the server
- Capital intensive in nature
- Waste of resources
- Lower return on investment
- Difficult to scale out and up

A natural evolution to physical servers was virtualization. Virtualization refers to the creation of virtual machines on top of physical servers and deploying applications within them. Virtual machines provide advantages in terms of the following:

- No need to procure physical hardware
- Comparatively easier to create newer virtual machines
- Complete isolation of environments
- Lower costs compared to physical servers

However, virtualization had its own set of disadvantages. They are the following:

- Still dependent on physical procurement of the server for scaling out after a number of virtual machine instances.
- Still costly because of human and hardware dependence.
- Wastage of compute resources. Each virtual machine runs a complete operating system in it.
- High reliance and costs in maintenance.

The next evolution was IaaS from cloud providers. Instead of procuring and managing data centers and infrastructures, the strategy was to create virtual machines, storages, and networks on the cloud. The cloud provides software-defined infrastructure services and hides all the details related to physical servers, networks, and storage. This had some advantages:

- No capital expenditure. Only operational expenses.
- No gestation time to create new virtual machines. New virtual machines can be provisioned within minutes rather than hours.
- Flexible size of virtual machines.
- Easier scaling up and out of virtual machines.
- Completely secure.

The disadvantages of virtual machines on the cloud are the following:

- Requires active monitoring and auditing.
- Requires active maintenance of virtual machines.
- Scalability, high-availability, and performance of virtual machines should be managed by users. Any degradation and subsequent improvement is the user's responsibility.
- Costly option because users pay for the entire machine whether it is used or not.

The cloud also provides another pattern for deploying applications, popularly known as PaaS. PaaS provides abstraction from the underlying infrastructure in terms of virtual machines, virtual networks, and storage on the cloud. It provides a platform and an ecosystem where users do not need to know a thing about the infrastructure at all; they can bring and deploy their application on these platforms. There are definite advantages using PaaS compared to IaaS but there are still possible better options.

The main disadvantages of PaaS are the following:

- PaaS applications are deployed on virtual machines behind the scenes and the payment model is not granular. It is still at the deployment level.
- It still demands monitoring for scaling out and in.
- Users still need to identify the requirements for their platform. There are limited options available for different types of platform. Azure provided the Windows environment for a long time and only recently was Linux available. Moreover, installation of packages, utilities, and software is the responsibility of its users.

Further, a new paradigm emerged known as containers, which was primarily popularized by Docker. Containers provide a lightweight, isolated, and secure environment that have all the benefits of virtual machines minus their disadvantages. They do not have a dedicated operating system and instead rely on a base server operating system. Containers come in both IaaS and PaaS patterns. Containers provide a large number of advantages:

- Faster provisioning of environments
- Consistent and predictable creation of environments
- They help in creating microservices architectures
- Rich ecosystem with advance services from Kubernetes, Swarm, and DC/OS

The disadvantages of containers are the following:

- They require active monitoring and auditing.
- They require active maintenance.
- Scalability, high-availability, and performance of virtual machines should be managed by users or advance tools, such as Kubernetes. Any degradation and subsequent improvement is the user's responsibility.

And this leads to serverless as the current state of evolution for deployment and consumption of applications.

Serverless

Serverless refers to a deployment model in which users are responsible for only their application code and configuration. Customers of serverless do not have to bother about the underlying platform and infrastructure and can concentrate on solving their business problems by writing code against them.

Serverless does not mean there are no servers. Code and configuration needs some server to run them. Serverless is from the customer's perspective. From the customer perspective, serverless means that they do not see the server at all. They do not care about the underlying platform and infrastructure. They do not need to manage and monitor them. Serverless provides an environment that can scale up and down, out and in, automatically, without the customers even knowing about it.

All operations related to platforms and infrastructures happen behind the scenes, transparently, without the user knowing about them. Customers are provided with performance **service level agreements (SLA)** and Azure ensures that it meets that SLA irrespective of the number or size of the servers required.

Customers are required to bring in their code only and the rest of the artifacts needed to run the code is the responsibility of the cloud provider.

Principles of Serverless Technology

Serverless technology is based on the following principles.

Lower Cost

Cost is based on the actual consumption of the compute resources and power. There is no cost associated to them if there is no consumption.

Event-Driven

Serverless functions should be able to execute based on certain events happening. The event should trigger the execution of the function. In other words, serverless should allow functions to decouple themselves from other functions and instead rely on the firing of certain events in which they are interested in.

Single Responsibility

Serverless should implement a single functionality and responsibility and it should do that well. Multiple responsibilities should not be coded or implemented within a single function.

Execute Quickly

Serverless functions should not take a long time to complete a job. They should be able to execute quickly and return back.

Azure Functions or Functions-as-a-Service (FaaS)

Azure provides *functions* as a resource. They are serverless implementations from Azure. With Azure functions, code can be written in any language the user is comfortable with and an Azure function will provide a runtime to execute them. Based on the language chosen, an appropriate platform is provided to users for bringing their own code. They are a unit of deployment and can automatically be out and in. When dealing with functions, users cannot view the underlying virtual machines and platform but Azure functions provide a small window to see them in terms of the **Kudu** console.

Azure Functions Runtime, Bindings, and Triggers

There are two main components of Azure functions.

Azure Function Runtime

The core and brain of Azure functions is its Azure runtime. The pre-cursor to Azure functions is Azure web jobs. The code for Azure web jobs also forms the core for Azure functions. There are additional features and extensions added to Azure web jobs to create Azure functions. The function runtime is the magic behind making the function work. Azure functions are hosted within Azure app services. Azure app services load Azure runtime and either wait for an external event to occur or for any HTTP requests. On arrival of a request or occurrence of a trigger, it loads the incoming payload, reads the function's `function.json` file to find the function's bindings and trigger, maps the incoming data to incoming parameters, and invokes the function with parameter values. Once the function completes its execution, the value is again passed backed to Azure function runtime by way of an outgoing parameter defined as binding in the `function.json` file. The function runtime returns the values to the caller. Azure function runtime acts as a glue that enables the entire functioning of functions.

Azure Function Binding and Triggers

If Azure function runtime is the brain of Azure functions, then function bindings and triggers are the heart of Azure functions. Azure functions promote loose-coupling and high cohesion between services using triggers and bindings. Generally, the application implements code using imperative syntax for incoming and outgoing parameters and return values. This generally results in hardcoding the incoming parameters. Since Azure functions should be capable of invoking any function defined, they implement a generic mechanism to invoke functions by means of triggers and binding.

Binding refers to the process of creating a connection between the incoming data and the Azure function, mapping the data types. The connection could be a single direction from runtime to the Azure function, Azure function to runtime for return values, or multi-direction--the same binding can transmit data between Azure runtime and Azure functions. Azure functions use a declarative way to define bindings.

Triggers are a special type of binding through which functions can be invoked based on external events. Apart from invoking the function, it also passed the incoming data, payload, and metadata to the function.

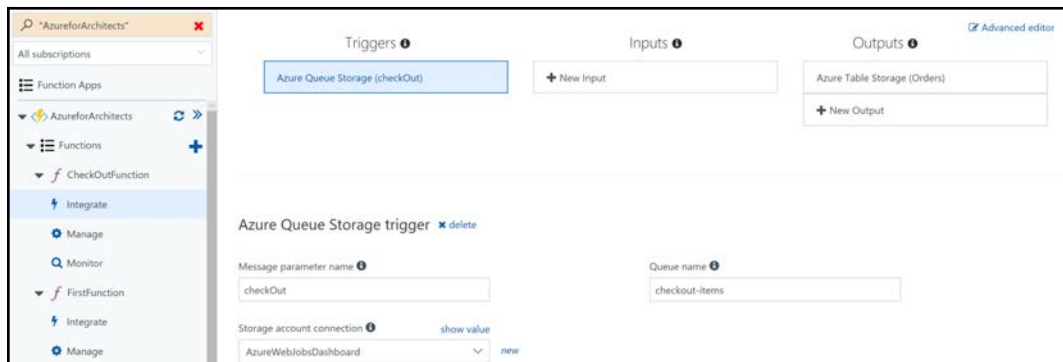
Bindings are defined in the `function.json` file:

```
{
  "bindings": [
    {
      "name": "checkout",
      "type": "queueTrigger",
      "direction": "in",
      "queueName": "checkout-items",
      "connection": "AzureWebJobsDashboard"
    },
    {
      "name": "Orders",
      "type": "table",
      "direction": "out",
      "tableName": "OrderDetails",
      "connection": "<<Connection to table storage account>>"
    }
  ],
  "disabled": false
}
```

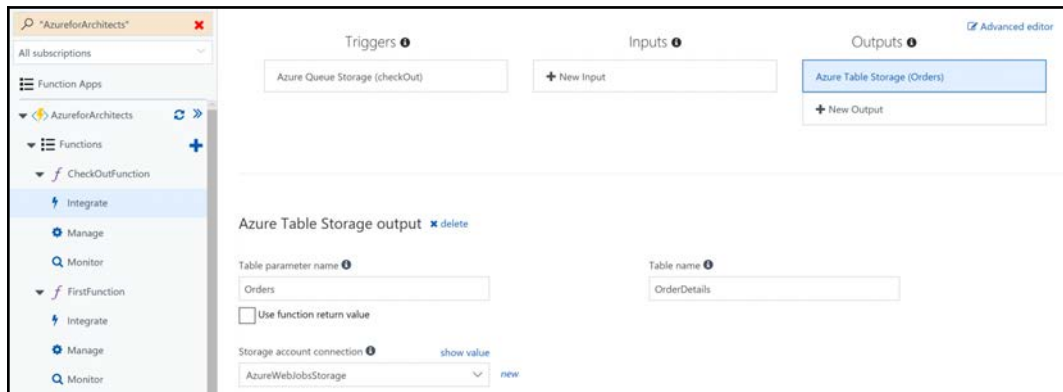
In this example, a trigger is declared that invokes the function whenever there is a new item in the storage queue. The type is of `queueTrigger`, the direction is `in`, `queueName` is `checkout-items`, and details about the target storage account connection and table name are also shown. All these values are important for the functioning of this binding. The name `checkOut` can be used within the functions code as a variable.

Similarly, a binding for return values is declared. Here the return value is named as `Orders` and the data is the output from the Azure function. It writes the return data into the Azure table storage using the connection string provided.

Both bindings and triggers can be modified and authored using the **Integrate** tab in Azure functions. Behind the scene, the `function.json` file is updated. The trigger `checkOut` is declared, as shown here:



And the output **Orders** is shown next:



The authors of Azure functions do not need to write plumbing code to get data from multiple sources. They just decide the type of data expected from the Azure runtime. This is shown in next code segment. Notice the checkout is available as a string to the function. Functions provide multiple different types to be able to send to a function. For example, a queue binding can provide the following:

- **Plain old simple object (POCO)**
- String
- Byte[]
- CloudQueueMessage

The author of the function can use any one of these datatypes and Azure function runtime will ensure that a proper object as a parameter is sent to the function:

```
using System;
public static void Run(string checkOut, TraceWriter log)
{
    log.Info($"C# Queue trigger function processed: { checkOut }");
}
```

It is also important to know that in the previous images, the storage account names are `AzureWebJobsStorage` and `AzureWebJobsDashboard`. These are keys that are defined within the Azure functions `appSettings` setting.

For more information on Azure bindings and triggers, refer to the following link: <https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-storage-queue>.

Azure Function Proxies

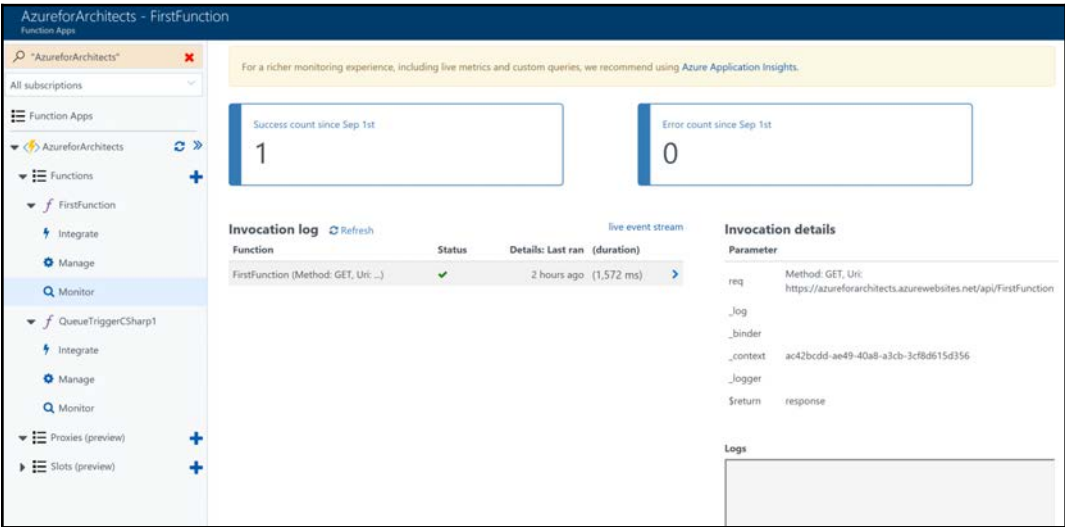
Azure function proxies are one of the latest additional features to Azure functions. After starting using Azure functions, there will be a time when there might be lots of function implementation and it will be difficult to integrate them together in a workflow by clients. Instead of letting clients weave these functions together, Azure proxies can be used. Proxies help in providing clients with a single function URL and then in turn invoke multiple Azure functions behind the scenes to complete workflows. It is important to understand that proxies are applicable in those cases where functions accept requests on demand instead of driven by events. These internal functions connected to a proxy can be within a single function app or on multiple separate apps.

Proxies get requests from clients, convert, override, and augment the payload and send them to backend internal functions. Once they get a response from these functions, they can again convert, override, and augment the response and send it back to the client.

More information about Azure functions can be found at <https://docs.microsoft.com/en-us/azure/azure-functions/functions-proxies>.

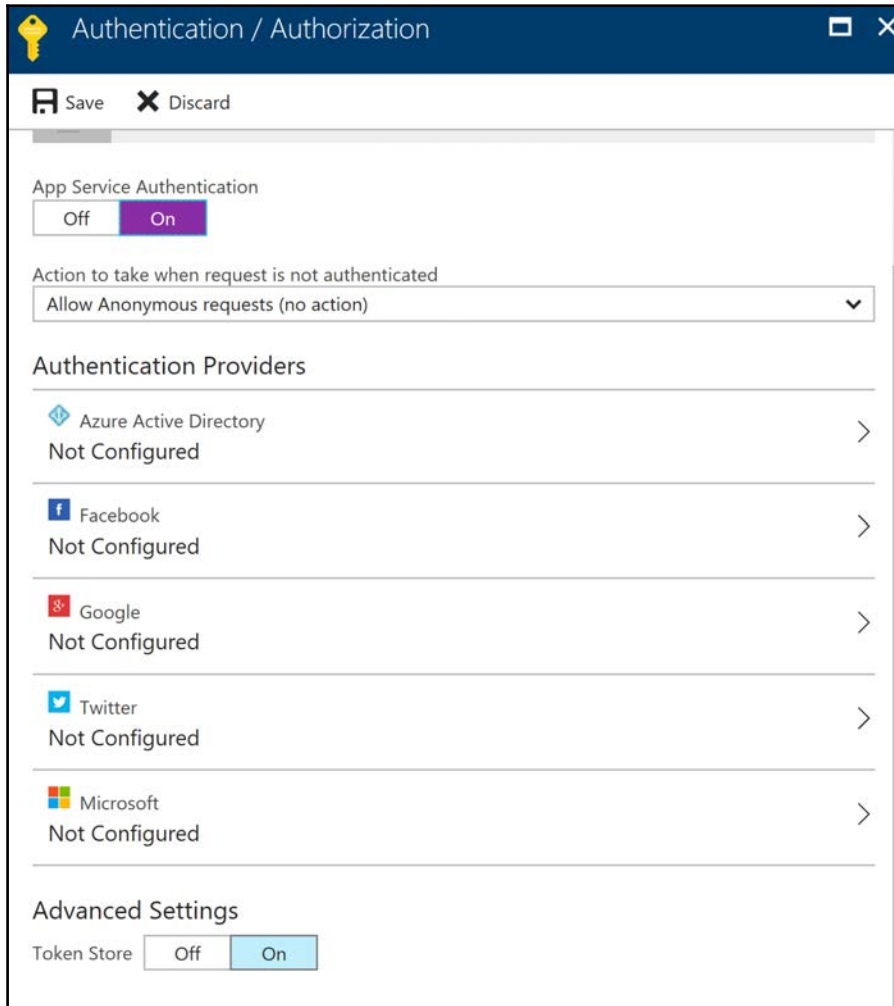
Monitoring

Complete log information is provided for each request or trigger in the **Monitor** tab of Azure functions. This helps in identifying issues and auditing any risks, bugs, or exception in Azure functions.



Authentication and Authorization

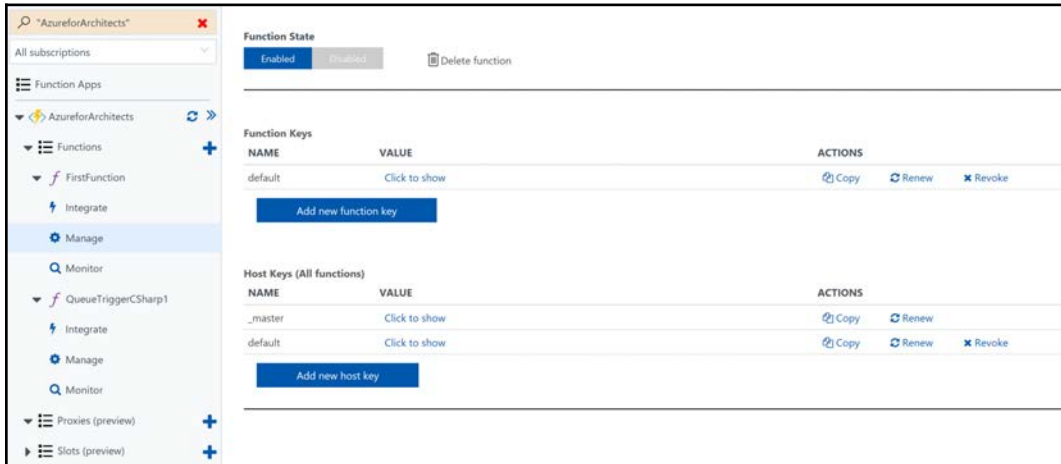
Azure functions rely on Azure app services for their authentication needs. App services have rich authentication features where clients can use OpenConnectID for authentication and OAuth for authentication. Users can get authenticated using Azure AD, Facebook, Google, Twitter, or Microsoft accounts.



The screenshot shows the 'Authentication / Authorization' configuration page for an Azure App Service. The page has a dark blue header with a yellow key icon and the title 'Authentication / Authorization'. Below the header, there are 'Save' and 'Discard' buttons. The main content area is divided into several sections:

- App Service Authentication:** A toggle switch is currently set to 'On' (purple).
- Action to take when request is not authenticated:** A dropdown menu is set to 'Allow Anonymous requests (no action)'.
- Authentication Providers:** A list of providers, each with a status of 'Not Configured' and a right-pointing chevron:
 - Azure Active Directory
 - Facebook
 - Google
 - Twitter
 - Microsoft
- Advanced Settings:** A 'Token Store' toggle switch is currently set to 'On' (light blue).

Azure functions that are based on HTTP can also incorporate usage of keys that should be sent along with HTTP requests. These keys are available from the **Manage** tab in the functions.



Function keys allow authorization to individual functions. These keys should be sent as part of the HTTP request header.

Host keys also allow authorization to all functions within a function app. These keys should be sent as part of the HTTP request header.

The default keys are used when using function and host keys. There is an additional host key named **_master** that helps in administrative access to the Azure function runtime API.

Azure Function Configuration

Azure functions provide configuration options at multiple levels. They provide configuration for the following:

- The platform itself
- The function app services

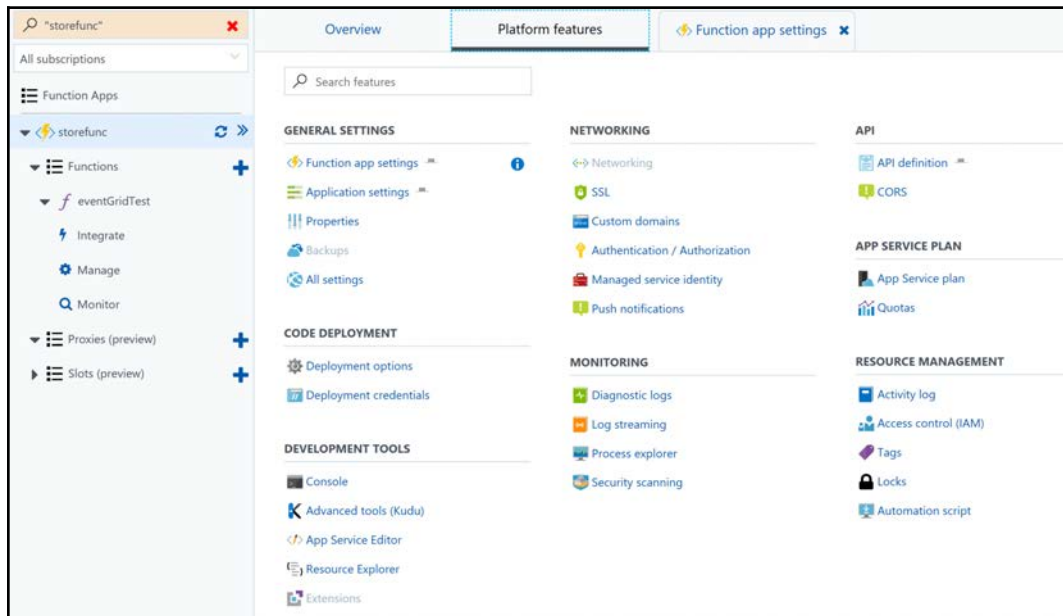
These settings affect every function contained by them. More information about these settings are available at <https://docs.microsoft.com/en-us/azure/azure-functions/functions-how-to-use-azure-function-app-settings>.

Platform Configuration

Azure functions are hosted within Azure app services, so they get all of its features. Diagnostic and monitoring logs can be configured easily using platform features. Furthermore, it provides options for assigning SSL certificates, using a custom domain, authentication, and authorization (using Azure AD, Facebook, and other OAuth and OpenConnectID protocols) as part of the networking features.

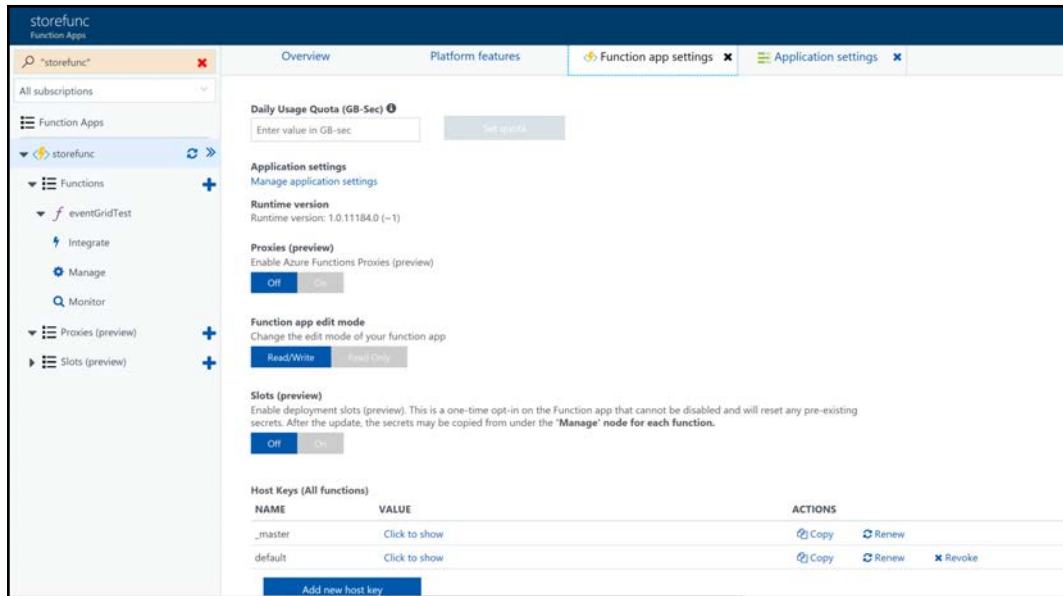
Although customers are not concerned about the infrastructure, operating system, filesystem, platform on which the functions actually execute, the Azure function provides the necessary tooling in place to peek within the underlying system to change and tweak them. The console and Kudu console are tools used for this purpose. They provide a rich editor to author Azure functions and edit their configuration.

Azure functions, just like app services, allow storing configuration information within the `web.config` application setting section, which can be read on demand.



App Services Function Settings

These settings affect all functions. Application settings can be managed here. Azure function proxies can be enabled and disabled. We will discuss proxies later in this chapter. They also help in changing the edit mode of a function app and the deployment to slots.



Azure Function Cost Plans

Azure functions are based on Azure app services and provide a better costing model for the users. There are two cost models:

- **Consumption plan:** This is based on the actual consumption and execution of functions. This plan calculates the cost based on the compute usage during actual consumption and execution of the function. If a function is not executed, there is no cost associated with it. However, it does not mean that performance is compromised in this plan. Azure functions will automatically scale out and in based on demand to ensure basic minimum performance levels are maintained. A function execution is allowed 10 minutes for its completion.

- **App service plan:** This plan provides complete dedicated virtual machines behind the scene to functions and so the cost is directly proportional to the cost of the virtual machine and its size. There is a cost associated with this plan even if functions are not executed at all. Function code can run for as long it is needed to complete. There is no time limit. Within the app service plan, the function runtime goes idle if not used within a few minutes and can be awoken only using a HTTP trigger. There is an **Always On** setting that can be used for not letting the function runtime to go idle. Scaling is either manual or based on auto-scale settings.

Azure Functions Advantages

Serverless computing is a relatively new paradigm in server technology and helps organizations convert large functionalities into smaller discrete on-demand functions that can be invoked and executed through automated triggers and scheduled jobs. They are also known as FaaS, in which organizations can focus on their domain challenges instead of the underlying infrastructure and platform. It also helps in democratizing solution architecture into smaller reusable functions deriving higher value for their investments.

There is a plethora of serverless compute platforms available on the internet. Some of the important ones are:

- Azure functions
- AWS Lambda
- IBM OpenWhisk
- Iron.io
- Google Cloud functions

In fact, every few days there is a new framework getting introduced and it is becoming increasingly difficult for enterprises to decide and choose the framework that works best for them.

Azure provides a rich serverless environment known as Azure functions and I would like to point out a few features that it supports to make them enterprise-ready:

- **Large choice of triggers:** Triggers act as events and execute serverless compute functions automatically. These triggers pass the raw data, the action, and event important to the functions. The serverless compute platform should provide different types of triggers enabling enterprises with multiple choices for their solution architecture. It should also provide facilities to execute the functions periodically as a scheduled task.
- **Webhooks:** A webhook helps in advanced integration with third-party services. It places hooks so that any change in a third-party service will start executing the serverless compute functions. It provides a rich infrastructure that can collaborate with important services such as version control systems and social media platforms.
- **On-demand execution:** The functions should not only be executed based on triggers but also have a mechanism to be executed manually. This brings flexibility during the development and testing of the functions.
- **Unlimited scalability:** A serverless compute platform should scale out and in functions dynamically based on their demand. Serverless compute functions are generally reachable through the internet and it is difficult to predict the number of users that could potentially connect to them. It is very important that they scale up automatically with more instances during high demand and scale down when demand reduces. This would help enterprises to reduce capital expenditure on an infrastructure that remains idle for a large part of the year.
- **High availability:** This is a no-brainer. Serverless functions should be highly available. Enterprises do not like unplanned downtime and it is extremely important for serverless functions to be available all the time (24x7x365). If a serverless compute platform does not guarantee more than five nines in terms of availability (99.999%), enterprises should look for another platform.

- **Security:** The platform should provide an appropriate infrastructure for providing security to serverless functions. There should be provision to authenticate, authorize, and maintain the confidentiality and identity of the users. Moreover, the platform should provide provisions to authenticate using the new age OAuth-based authorization protocols. The platform should also allow **Cross-Origin Resource Sharing (CORS)**. CORS allows JavaScript code running in a browser on an external host to interact with serverless functions for functions accessible through the internet.
- **Industry standards:** The platform should use industry standards rather than proprietary technology. JSON is becoming the de-facto standard for message payload exchange on REST endpoints. It should support both HTTP and HTTPS protocols. This allows cross-platform applications to invoke and consume these services.
- **Monitoring, logging, and auditing:** Any serverless platform should provide infrastructure to monitor functions at the operational level, generate alerts, and inform teams to take corrective actions on them. The platform should provide a rich logging mechanism to capture activities in functions and provide telemetry metrics to its owner. This should include availability, scalability, and performance details among others. The logs should be easily accessible in the form of dashboards, making an audit exercise easy for administrators. Enterprises want to optimize time, effort, and money spent on keeping serverless functions operational. They should choose a serverless platform that provides rich operational information through monitoring, logging, and audit capabilities.
- **DevOps friendly:** Enterprises have caught up big time with DevOps. Large enterprises have already adopted DevOps; the small and medium enterprises are not far behind. Every company wants to deploy technologies in their landscape and ecosystem that are automatable from continuous integration, deployment, and delivery processes. The platform should not only provide automation endpoints and utilities but it should also be easy for developers to configure DevOps-related practices for function development.

- **Language support:** A serverless platform should provide multiple language options for the development of functions. Gone are the days when enterprises used a single popular language for all their development activities. Today, companies deploy languages known well to their developers. They also use heterogeneous languages in their landscape for higher productivity and affinity to the type of application getting developed. It could be C#, JavaScript, Bash, PowerShell, or a combination of any such technologies.
- **Advance tools for authoring, testing, and deployment:** Tools should make developing, testing, and debugging easy for the function developer. Apart from providing quick start wizards, tools should increase developer productivity by providing an integrated environment for developing and debugging, executing unit and integration tests, the ability to mock objects, and help in automated deployments. The authoring editor should provide templates and scaffolding code to fast-track development. It should also provide emulators that help in testing the function without hosting on the real environment.
- **Fault tolerant:** Serverless functions should not crash or die due to errors or exceptions in user code or server infrastructure. They should log the exceptions, inform the user, and elegantly continue to execute the next request.
- **Long running activities:** A serverless platform should provide flexibility to run long running activities. After the long running activities complete their execution, the serverless platform should pick up the execution and return to the user.
- **Ability to configure:** Although a serverless compute means no access to the underlying platform, there should be adequate touchpoints through which administrators can peep and change the configuration of the underlying platform. This will help enterprises to troubleshoot, conduct advance deployments, and tweak the platform according to their needs.
- **Set quotas:** The serverless compute platform should have flexibility to determine quotas for execution. This would help in constraining mindless usage of a serverless compute.
- **Multiple hosting plans:** The platform should provide options for multiple usage plans depending upon the resource consumption and executions. Enterprises can choose the plan that best fits their needs and optimize their expenditure while ensuring cost benefits.

- **Rich integrations:** The serverless compute platform should be able to integrate with other services, such as cognitive services, analytics services, communication services, and other transaction-based services, to provide rich and contextual services to customers. This will help enterprises to create solutions that are innovative and relevant to the industry.
- **Massive parallelism:** The serverless platform should be capable of executing requests synchronously as well asynchronously. This eventually helps in writing patterns that can help in massive scalability of functions by optimally utilizing the platform resources.
- **Multi-level configurations:** The platform should provide options to configure at the individual function level, function grouping level and platform level. This will help enterprises to change the overall behavior of their functions, including platform behavior, with minimal effort.

Azure Functions Use Cases

There are many valid use cases for using and implementing Azure functions.

Implementing Microservices

Azure functions help in breaking down large applications into smaller discreet functional code units. Each unit is treated independent of the other and evolves in its own life cycle. Each such code unit has its own compute, hardware, and monitoring requirements. Each function can be connected to all other functions. These units are weaved together by orchestrators to build complete functionality. For example, in an e-commerce application, there can be individual functions (code units) each responsible for listing catalogs, recommendations, categories, subcategories, shopping carts, checkouts, payment types, payment gateways, shipping addresses, billing addresses, taxes, shipping charges, cancellations, returns, emails, SMS, and so on. Some of these functions are brought together to create use cases for e-commerce applications such as product browsing, checkout flow, and so on.

Integration Between Multiple Endpoints

Azure functions can build overall application functionality by integrating multiple functions. The integration can be based on the triggering of events or it could be on a push basis. It helps in decomposing large monolithic applications into small components.

Data Processing

Azure functions can be used for processing incoming data in batches. They can help in processing data coming in multiple formats such as XML, CSV, JSON, text, and so on. They can also run conversion, enrichment, cleaning, and filtering algorithms. In fact, multiple functions can be used, each doing either conversion or enrichment, cleaning or filtering. Azure functions can also be used to incorporate advance cognitive services such as **optical character recognition (OCR)**, computer vision, image manipulation, and conversion.

Integrating Legacy Applications

Azure functions can help in integrating legacy applications with newer protocols and modern applications. Legacy applications might not be using industry standard protocols and formats. Azure functions can act as a proxy for these legacy applications, accept requests from users or other applications, convert the data into a format understood by a legacy application, and talk to it on protocols it understands. This opens a world of opportunity for integrating and bringing old and legacy applications into the mainstream portfolio.

Scheduled Jobs

Azure functions can be used to execute continuously periodically for certain application functions. These application functions can range from periodically taking backups, restoring, running batch jobs, exporting and importing data, bulk emails, and so on.

Communication Gateways

Azure functions can be used in communication gateways such as using notification hubs, **short messaging service (SMS)**, and email.

Types of Azure Functions

Azure functions can be categorized into three different types:

- **On demand functions:** These are functions that are executed when they are explicitly called or invoked. Examples of such functions include HTTP-based functions and webhooks.
- **Scheduled functions:** These functions are like timer jobs and execute functions on fixed intervals.
- **Event-based functions:** These functions are executed based on external events. For example, uploading a new file to Azure blob storage generates an event which can start execution of the Azure function.

Creating Your first Azure Function

Azure functions can be created using the Azure portal, PowerShell, Azure CLI, and REST API's. Steps to create a function using the ARM template is already detailed at <https://docs.microsoft.com/en-us/azure/azure-functions/functions-infrastructure-as-code>. In this section, Azure functions will be provisioned using the portal.

Azure functions are hosted within Azure app services. Users create a new function app which in turn creates an app service plan and app service. The app service plan is configured based on the following:

- **Name:** Name of app service. The name should be unique within the `.azurewebsites.net` domain.
- **Location:** Location for hosting the Azure function app service.

- **Hosting plan:** This is also known as the pricing plan. Here, two options, as discussed previously, are available--consumption and app services plan.
- **Resource group name:** The name of the resource group containing both the app service plan and app service.
- **Storage account:** Azure functions needs an Azure storage account to store their internal data and logs.
- **Enable application insights:** To capture telemetry information from Azure functions.

The screenshot shows the 'Function App' creation wizard in the Azure portal. The window has a title bar with 'Function App' and a 'Create' button. The form is divided into several sections, each with a red asterisk indicating a required field. The 'App name' field contains 'AzureforArchitects' with a green checkmark and '.azurewebsites.net' as a suffix. The 'Subscription' dropdown is set to 'RiteshSubscription'. The 'Resource Group' section has radio buttons for 'Create new' and 'Use existing', with 'Use existing' selected and 'storerg2' in the dropdown. The 'Hosting Plan' dropdown is set to 'Consumption Plan'. The 'Location' dropdown is set to 'South Central US'. The 'Storage' section has radio buttons for 'Create New' and 'Select Existing', with 'Create New' selected and 'azureforarchitead2b' in the text field with a green checkmark. At the bottom, there is a toggle for 'Application Insights' set to 'Off'. A 'Pin to dashboard' checkbox is also present. At the very bottom, there is a blue 'Create' button and a link for 'Automation options'.

Function App
Create

* App name
AzureforArchitects ✓
.azurewebsites.net

* Subscription
RiteshSubscription ▼

* Resource Group ⓘ
☐ Create new ☒ Use existing
storerg2 ▼

* Hosting Plan ⓘ
Consumption Plan ▼

* Location
South Central US ▼

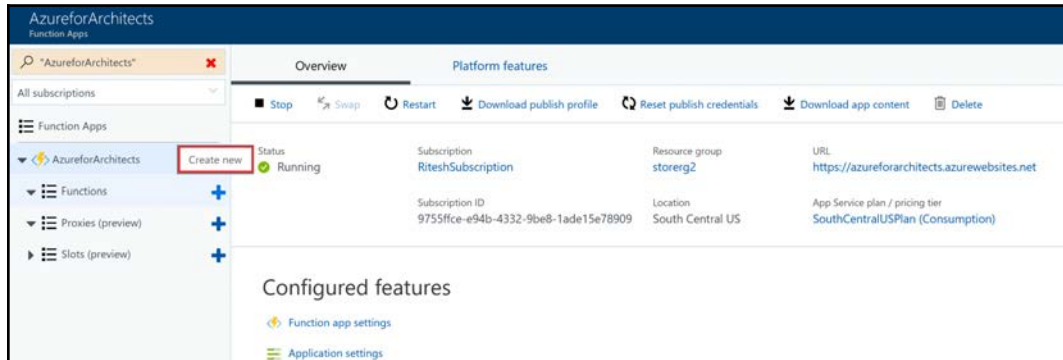
* Storage ⓘ
☒ Create New ☐ Select Existing
azureforarchitead2b ✓

Application Insights ⓘ

☐ Pin to dashboard

Create Automation options

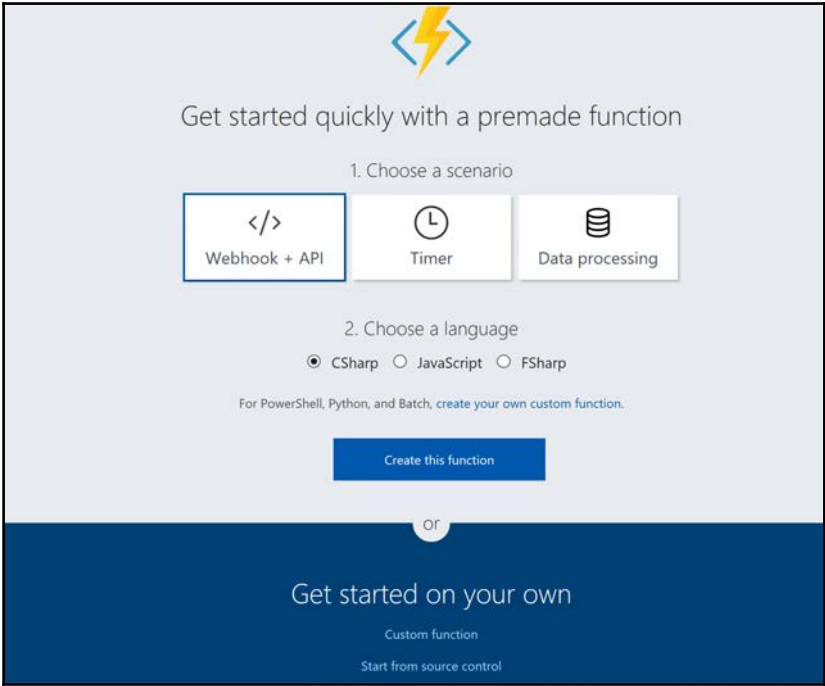
Creating an Azure function app will provide a dashboard after provisioning.



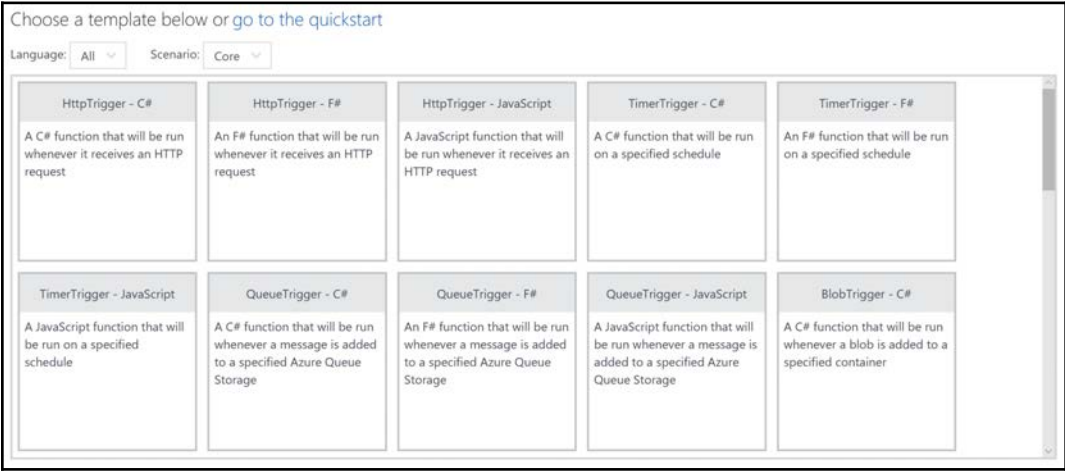
Clicking on the plus button next to the functions will show a wizard for creating a new function. This wizard shows quick templates for creating the **Webhook + API**, **Timer**, and **Data processing** functions. It will create the necessary scaffolding code and structure for getting started. This scaffolding code is also generated for default bindings and triggers. It also allows for the selection of a language for implementation of the function. Azure functions allow the following:

- C#
- JavaScript
- F#
- PowerShell
- Bash
- Python languages

This wizard also allows for creation of custom functions from scratch.



There are many more options provided for creating a custom function.



In this section, a custom Azure function will be created using the PowerShell language, which can be executed whenever it receives a HTTP request. Provide an appropriate name, in this case `FirstFunction`, and choose **Anonymous** as the authorization level. Authorization levels will be discussed later in this chapter.

Choose a template below or [go to the quickstart](#)

Language: PowerShell Scenario: Experimental

HttpTrigger - PowerShell	TimerTrigger - PowerShell	QueueTrigger - PowerShell
(Preview) A PowerShell function that will be run whenever it receives an HTTP request	(Preview) A PowerShell function that will be run on a specified schedule	(Preview) A PowerShell function that will be run whenever a message is added to a specified Azure Queue Storage

This template is experimental and does not yet have full support. If you run into issues, please file a bug on our [GitHub repository](#).

Name your function

HTTP trigger

Authorization level ⓘ

Anonymous

Creating this function provides a complete function authoring integrated environment along with some code. This code gets the raw content from the incoming `req` parameter, which is filled up by Azure runtime with incoming data (query string, form values, and so on), and it converts it in the PowerShell object. There could be multiple types of data within this incoming parameter, and a single `name` value is extracted out of it.

Azure functions runtime also creates a new variable for every incoming querystring with an HTTP request and appends it with `$req_query_`. So sending the name as querystring will provide an out-of-the-box variable, `$req_query_name`, containing the value for the name querystring and will be consumed within the function. The response for this function is written to a file that the Azure function runtime reads and sends back to the browser.



This function that can be invoked using an HTTP request from the browser. The URL for this function is available from the environment and is composed of the function app name along with the function name. The format is `https://<<function app name>>.azurewebsites.net/api/<<function name>>`. In this case, the URL will be `https://azureforarchitects.azurewebsites.net/api/FirstFunction`.

For sending parameters to this function, additional query string parameters can be appended at the end of the URL. For example, for sending name parameters to this function, the `https://azureforarchitects.azurewebsites.net/api/FirstFunction?name=ritesh` URL can be used. The output of the function is shown in the following figure:



For HTTP-based functions, the Azure function already provides triggers and binding within the `function.json` file. This file is used for defining all function level triggers and bindings and there is one associated with every function.

A screenshot of the Azure portal's function editor interface. At the top, there's a header bar with the filename 'function.json', a blue 'Save' button, a 'Run' button with a play icon, and a link '</> Get function URL'. Below this is a code editor with a light blue background and line numbers from 1 to 16 on the left. The code is a JSON object defining function bindings. It has a 'bindings' array with two objects. The first object is for an 'httpTrigger' with name 'req', direction 'in', and auth level 'anonymous'. The second object is for an 'http' binding with name 'res' and direction 'out'. The 'disabled' property is set to false.

```
1 {  
2   "bindings": [  
3     {  
4       "name": "req",  
5       "type": "httpTrigger",  
6       "direction": "in",  
7       "authLevel": "anonymous"  
8     },  
9     {  
10      "name": "res",  
11      "type": "http",  
12      "direction": "out"  
13    }  
14  ],  
15  "disabled": false  
16 }
```

The HTTP template creates a trigger for all incoming requests. The trigger invokes the Azure function and passes in the entire incoming data and payload as a parameter named as `req`. This parameter is available within the Azure function. The response from the function is a binding that takes output from the `res` variable from the Azure function and sends it back to the HTTP channel as a response.

Creating an Event-Driven Function

In this example, an Azure function will be authored connected to the Azure storage account. The storage account has a container for holding all blob files. The name of the storage account is `incomingfiles` and the container is `orders`:

Blob service

incomingfiles

+ Container

Refresh

Storage account

incomingfiles

Status

Primary: Available

Location

West Central US

Subscription (change)

RiteshSubscription

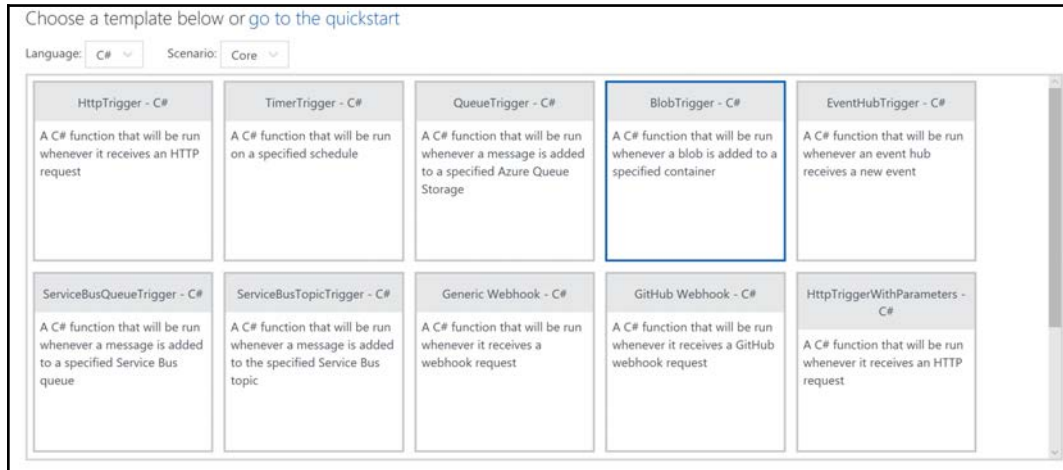
Subscription ID

9755ffce-e94b-4332-9be8-1ade15e78909

Search containers by prefix

NAME
orders

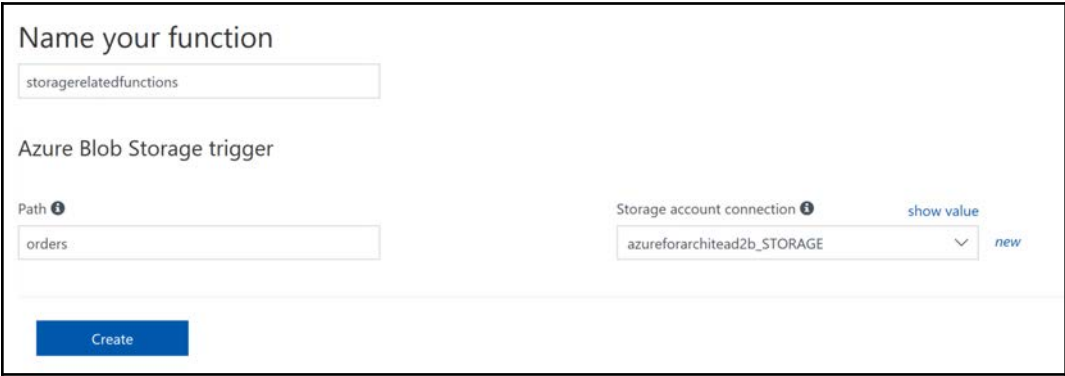
Create a new Azure function from portal.



As of now, this Azure function does not have connectivity information to the storage account. Azure functions need connection information for the storage account and that is available from the **Access keys** tab in the storage account. The same information can be obtained using the Azure function editor environment. In fact, it allows for the creation of a new storage account from the same editor environment.

This can be added using the new button beside the **Storage account connection** input type. It allows for selecting an existing storage account or creating a new storage account. Since I already have a couple of storage accounts, I am reusing them. Readers should create a separate Azure storage account. Selecting a storage account will update the settings in the **appsettings** section with the connection string added to it.

Ensure that a container already exists within the blob service of the target Azure storage account. The path input refers to the path to the container. In this case, the orders container already exists within the storage account. The **Create** button will provision the new function monitoring the storage account container.



The screenshot shows the 'Name your function' page in the Azure portal. The 'Name' field contains 'storagerelatedfunctions'. Below it, the 'Azure Blob Storage trigger' section is visible. The 'Path' field contains 'orders'. The 'Storage account connection' dropdown menu is set to 'azureforarchitead2b_STORAGE', with a 'show value' link and a 'new' button next to it. A blue 'Create' button is at the bottom left.

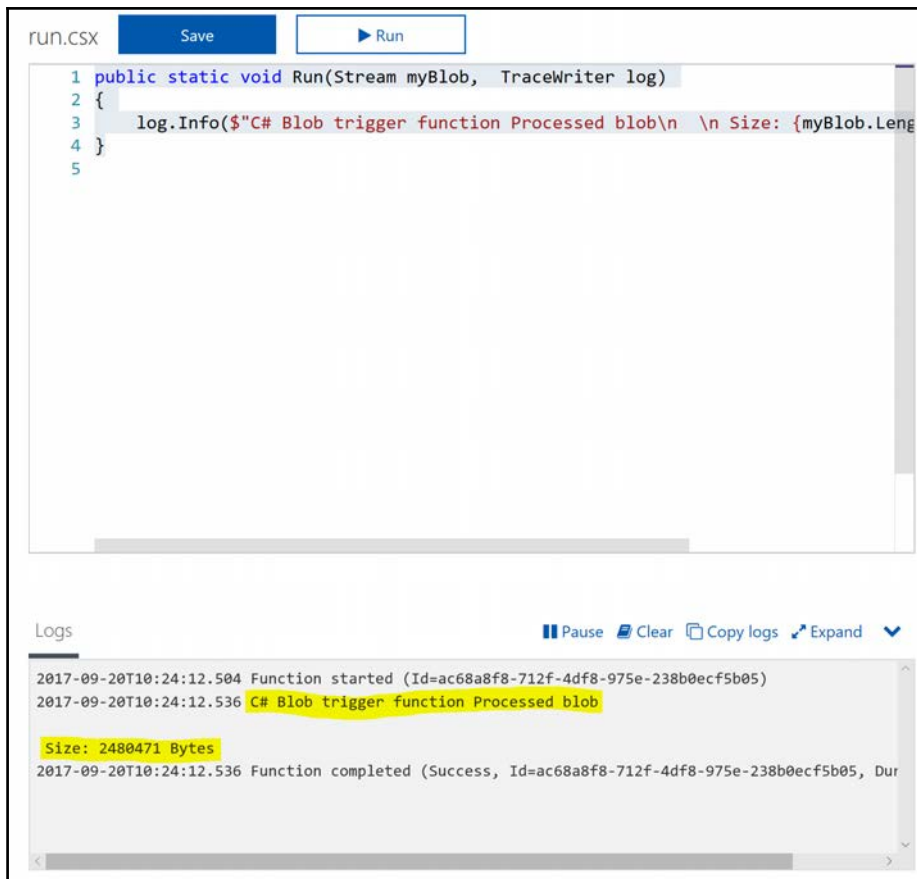
The code for the Azure function is as follows:

```
public static void Run(Stream myBlob, TraceWriter log)
{
    log.Info($"C# Blob trigger function Processed blob\n  \n Size
{myBlob.Length} Bytes");
}
```

The bindings are shown next:

```
{
  "bindings": [
    {
      "name": "myBlob",
      "type": "blobTrigger",
      "direction": "in",
      "path": "orders",
      "connection": "azureforarchitead2b_STORAGE"
    }
  ],
  "disabled": false
}
```

Now, uploading any blob file in the orders container should trigger the function:



The screenshot shows a Visual Studio Code editor window with a file named `run.csx`. The code is a C# function that takes a `Stream` and a `TraceWriter` as input. It logs a message indicating that a blob trigger function has processed a blob and its size. The function is named `Run` and is static. The code is as follows:

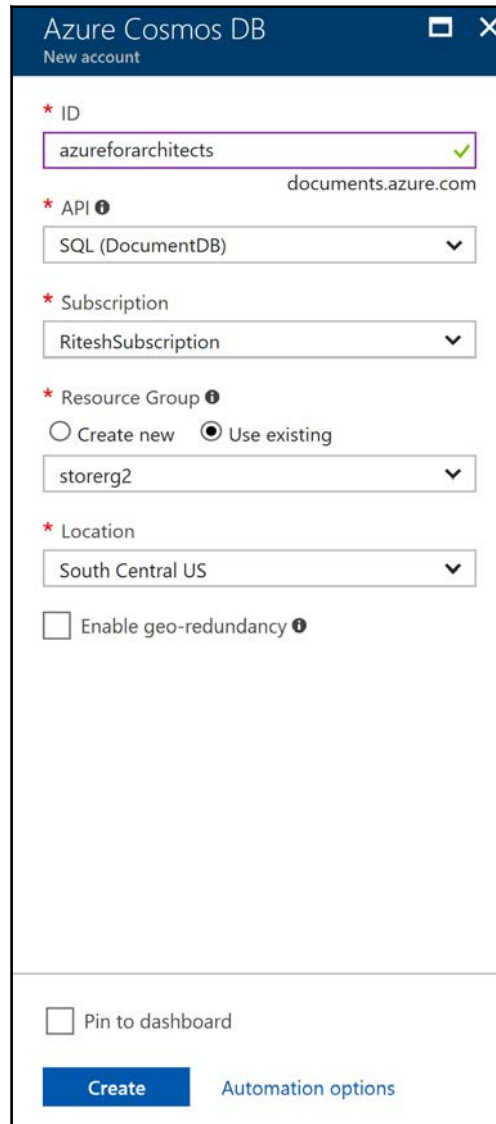
```
1 public static void Run(Stream myBlob, TraceWriter log)
2 {
3     log.Info($"C# Blob trigger function Processed blob\n   Size: {myBlob.Length} Bytes");
4 }
5
```

Below the code editor, there is a "Logs" panel. It shows the execution logs of the function. The logs indicate that the function started at 2017-09-20T10:24:12.504 and completed at 2017-09-20T10:24:12.536. The log message is "C# Blob trigger function Processed blob Size: 2480471 Bytes".

Creating a Connected Architecture with Functions

A connected architecture with functions refers to creating multiple functions, whereby the output of one function triggers another function and provides data for the next function to execute its logic. In this section, we will continue with the previous scenario of the storage account. In this case, the output of the function getting triggered using Azure storage blob files will write the size of the file to Azure Cosmos DB.

The configuration of Cosmos DB is shown next. By default, there are no collections created in Cosmos DB. A collection will automatically be created while creating a function that will get triggered when Cosmos DB gets any data.



The screenshot shows the 'Azure Cosmos DB' 'New account' configuration window. The window has a dark blue header with the title 'Azure Cosmos DB' and a subtitle 'New account'. Below the header, there are several configuration fields, each marked with a red asterisk (*):

- ID:** A text input field containing 'azureforarchitects' with a green checkmark icon to its right.
- API:** A dropdown menu showing 'SQL (DocumentDB)'.
- Subscription:** A dropdown menu showing 'RiteshSubscription'.
- Resource Group:** A section with two radio buttons: 'Create new' (unselected) and 'Use existing' (selected). Below the radio buttons is a dropdown menu showing 'storerg2'.
- Location:** A dropdown menu showing 'South Central US'.
- Enable geo-redundancy:** A checkbox that is currently unchecked.

At the bottom of the window, there is a checkbox labeled 'Pin to dashboard' which is also unchecked. Below this, there is a blue 'Create' button and a link labeled 'Automation options'.

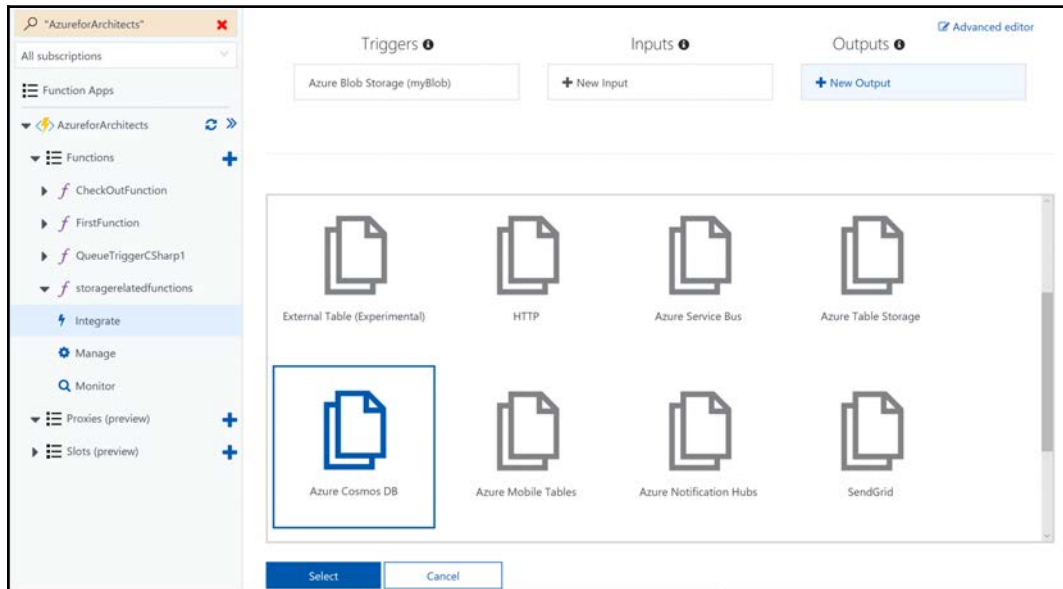
Create a new database `testdb` within Cosmos DB and a new collection named `testcollection` within it. You need both the database and collection name while configuring Azure functions.

The screenshot shows the 'Add Collection' dialog box in the Azure Cosmos DB portal. The dialog has a dark blue header with the title 'Add Collection' and window control icons. The main content area is white and contains several sections, each with a red asterisk and an information icon:

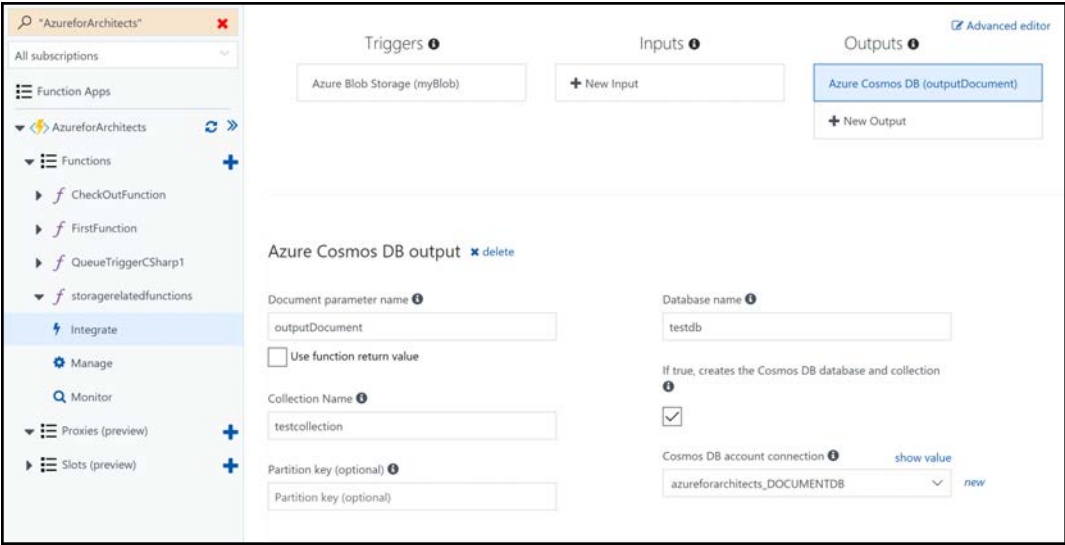
- Collection Id**: A text input field containing 'testcollection' with a green checkmark on the right.
- STORAGE CAPACITY**: Two radio buttons. 'Fixed (10GB)' is selected and highlighted in purple. 'Unlimited*' is the other option.
- INITIAL THROUGHPUT CAPACITY (RU/s)**: A text input field containing '5000' with a green checkmark on the right. To the right of the input are minus and plus icons. Below the input, it says 'Between 400 and 10000 RU/s' and 'Estimated hourly spend 0.400 USD'.
- PARTITION KEY**: A text input field that is empty with a green checkmark on the right.
- DATABASE**: Two radio buttons. 'Use existing' is selected. Below them is a dropdown menu showing 'testdb'.

At the bottom of the dialog is a blue 'OK' button.

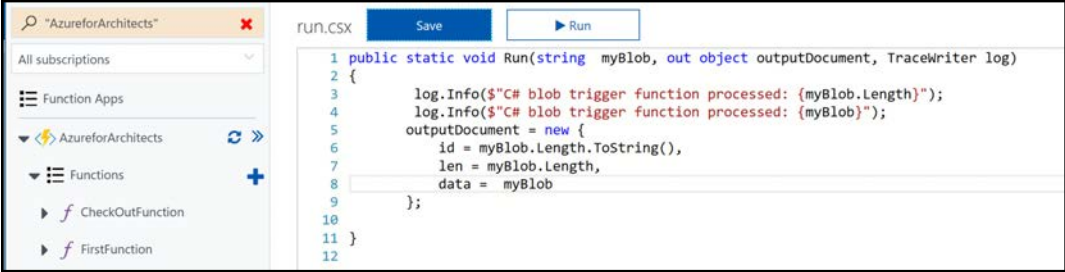
It's time to revisit the `storagerelatedfunctions` function and change its binding to return the size of the data for the uploaded file. This returned value will be written to Cosmos DB. This will demand a change to the bindings as well with an additional one responsible for capturing output values. This binding will eventually write to the Cosmos DB collection. Navigate to the **Integrate** tab and click on the **New Output** button following the **Outputs** label and select **Azure Cosmos DB**.



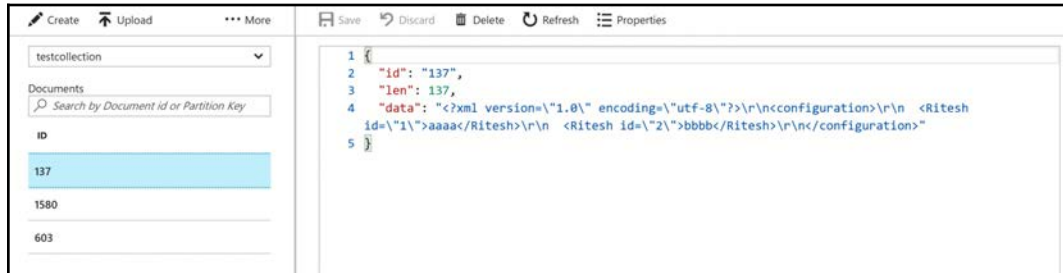
Provide the appropriate names for the database and collection (check the checkbox to create the collection if it does not exist), click on the **New** button to select our newly created Azure Cosmos DB, and leave the parameter name as `outputDocument`.



Modify the function as shown next.



Now uploading a new file to the orders collection in the Azure storage account will execute a function that will write to the Azure Cosmos DB collection. Now another function can be written with the newly created Azure Cosmos DB as a trigger binding. It will provide the size of files and the function can act on it. This is shown next:



Summary

Serverless is a new deployment unit in today's world. The evolvement of functions from traditional methods has led to designing loosely coupled, independently evolving, self-reliant architecture that was only a concept in earlier days. Functions are a unit of deployment and provide an environment where users do not need to manage the environment at all. All they have to care about is the code written for the functionality. Azure provides a mature platform for hosting functions and integrating them seamlessly based on events or on demand. Nearly every resource in Azure can participate in an architecture composed of Azure functions. The future is functions, as more and more organizations want to stay away from managing infrastructures and platforms. They want to offload this to cloud providers. It is an essential feature to master by every architect dealing with Azure.

Next chapter is an interesting one as it deals with Azure Resource Manager concepts such as policies, locks, and tags in details.

16

Designing Policies, Locks, and Tags

Azure is a versatile cloud platform. Customers can not only create and deploy their application, they can also actively manage and govern their environments. Clouds generally follow a pay-as-you-go paradigm where a customer subscribes for an Azure subscription and virtually can deploy anything on it. It could be as small as a small basic virtual machine to thousands of virtual machines with higher Sku's. Azure will not stop any customer from provisioning any resources or constraint them in terms of numbers. Within an organization there could be a large number of people having access to the Azure subscription. There is a need to have a governance model in place such that only the required adequate resources are provisioned by people who have rights to create them. Azure provides resource management features, such as Azure **Role based Access Control (RBAC)** and policies and locks for managing and providing governance for resources.

Another major aspect of governance is cost, usage, and information management. Organization management would always want to be kept updated about their cloud consumption and cost. They would like to identify which team, department, or unit is using what percentage of their total cost. In short, they want to have reports based on various dimensions about consumption and cost. Azure provides a tagging feature that can help provide this kind of information to the management on the fly.

In this chapter, we will cover the following topics:

- Azure RBAC
- Azure policies
- Azure locks
- Azure tags

Azure Tags

Tags in a dictionary is defined as *a label attached to someone or something for the purpose of identification or to give other information*. Azure allows tagging of resource groups and resources with name-value pairs. Tagging helps in logical organization and categorization of resources. Azure also allows tagging 15 name-value pairs for resource groups and its resources. Although resource group is a container for resources, tagging resource group does not mean tagging of its constituent resources. Resource groups and resources should be tagged based on their usage, which will be explained later in this section. Tags work at a subscription level. Azure accepts any name-value pairs and so it is important for an organization to define both the names and their possible values.

But why is tagging important? In other words, what problems can be solved using tagging. Tagging has the following benefits:

- **Categorization of resources:** An Azure subscription is used by multiple departments and roles within an organization. It is important for the management to identify the owners of these resources. Tagging helps in assigning identifiers to resources that can represent department or roles.
- **Information management for Azure resources:** Again Azure resources in a subscription can be provisioned by anyone having access to it. Organizations would like to have a proper categorization of resources in terms of their information management policies. The policies can be based on application life cycle management, such as development, testing, and production environments. It could be based on the usage--internal or external facing. It could be based on their priorities and more. Each organization has their own way of defining information categories and for them Azure is no different. Tags help in organizing resources logically.
- **Cost management:** Tagging in Azure can help in identifying resources based on their categorization. Queries can be executed against Azure to identify cost per category according to information management categories defined. For example, the cost of resources in Azure for developing an environment for the finance department and the marketing department can be easily ascertained. Moreover, Azure also provides billing information based on tags. This helps in identifying which teams, departments, or groups are consuming.

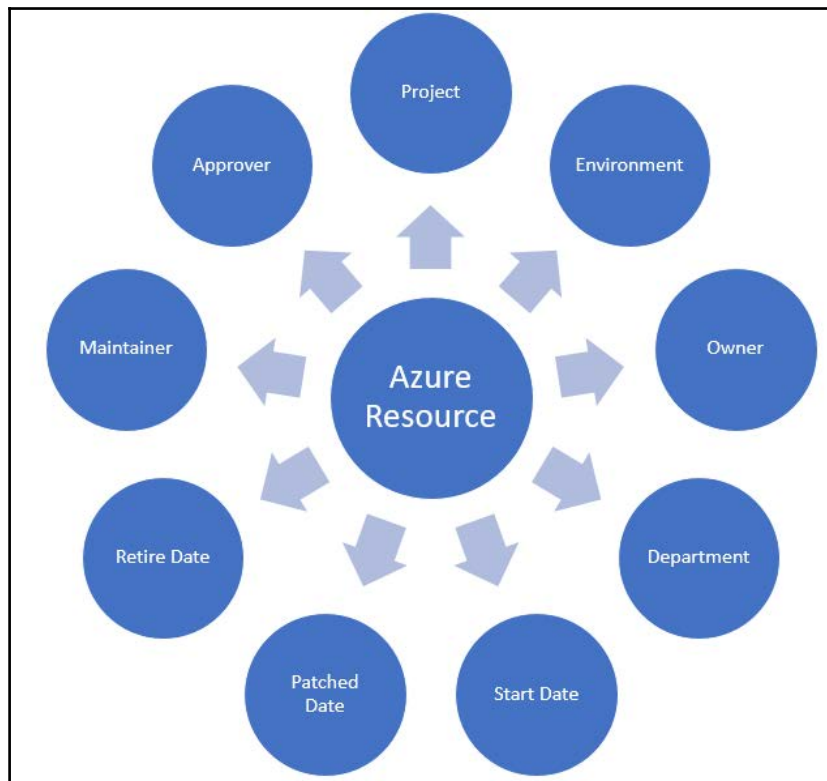
Tags have certain limitations in Azure.

Azure allows a maximum of 15 tag name-value pairs to be associated with a resources and resource groups.

Tags are non-inheritable. Tags applied to a resource group do not apply to the individual resources within it. However, it is quite easy to forget to tag resources while provisioning them. Azure policies are the mechanism to ensure that tags are tagged with the appropriate value during provision time. We will consider details of policies later in this chapter. A complete section is devoted to policies.

Tags can be assigned to resources and resource groups using PowerShell, Azure CLI 2.0, Azure resource management templates, Azure portal, and Azure resource manager REST API's.

A sample information management categorization using Azure tags that can be reused is shown here. In this example, department, project, environment, owner, approver, maintainer, start-date, retire-date, and patched-date name-value pairs are used to tag a resource group or resources. It is extremely easy to find all the resources for a particular tag or a combination of tags using PowerShell, Azure CLI, or REST API's.



Tags with PowerShell

Tags can be managed using PowerShell, Azure Resource Manager templates, portal, and REST API's. In this section, PowerShell will be used to create and apply tags. PowerShell provides a cmdlet for retrieving and attaching tags to resource groups and resources.

- To retrieve tags associated with a resource using PowerShell, the `Find-AzureRmResource` cmdlet can be used:

```
(Find-AzureRmResource -TagName Dept -TagValue Finance).Name
```

- To retrieve tags associated with a resource group using PowerShell, the following command can be used:

```
(Find-AzureRmResourceGroup -Tag @{ Dept="Finance" }).Name
```

- To set tags to a resource group, the `Set-AzureRmResourceGroup` cmdlet can be used:

```
Set-AzureRmResourceGroup -Name examplegroup -Tag @{ Dept="IT";  
Environment="Test" }
```

- To set tags to a resource, the `Set-AzureRmResource` cmdlet can be used:

```
Set-AzureRmResource -Tag @{ Dept="IT"; Environment="Test" } -  
ResourceName examplevnet -ResourceGroupName examplegroup
```

Tags with the ARM Template

Azure resource manager template provides also helps in defining tags for each resource. It can be used to assign multiple tags to each resource. This is shown as follows:

```
{  
  "$schema":  
    "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",  
  "contentVersion": "1.0.0.0",  
  "resources": [  
    {  
      "apiVersion": "2016-01-01",  
      "type": "Microsoft.Storage/storageAccounts",  
      "name": "[concat('storage', uniqueString(resourceGroup().id))]",  
    },  
  ],  
}
```



```
    "location": "[resourceGroup().location]",
    "tags": {
      "Dept": "Finance",
      "Environment": "Production"
    },
    "sku": {
      "name": "Standard_LRS"
    },
    "kind": "Storage",
    "properties": { }
  }
]
```

In the previous example, a couple of tags--`Dept` and `Environment` are added a storage account resource using Azure resource manager templates.

Resource Groups versus Resources

It is a must that architects decide the taxonomy and information architecture for Azure resources and resource groups. They should identify the categories on which resources will be classified based on query requirements. However, they must also identify whether tags should be attached to individual resources or resource groups.

If all resources within a resource group need the same tags, then it is better to tag the resource group rather than tagging each resource. It is important to take the queries on tags into consideration before finalizing whether tags should be applied at the resource level or the resource group level. If the queries relate to individual resource types across subscription and resource groups then assigning tags to resources make more sense, however, if identifying resource groups are enough in queries, then tags should be applied to resource groups.

Azure Policies

The last section talked about applying tags for Azure deployments. Tags are great for organizing resources; however, there is one more factor and question that was not discussed. How do organizations enforce tags to be applied for every deployment? There should be automated enforcement of Azure tags to resources and resource groups. There is no check from Azure to ensure that appropriate tags will be applied to resources and resource groups. Now, this is not just related to tags—it could be related to any configuration for any resource on Azure. For example, all resources should be deployed only in East US and no resources can be deployed elsewhere.

You might have guessed by now that this section is all about formulating a governance model on Azure. Governance is an important element for Azure because it helps in bringing discipline and cost under control. It also ensures that everyone accessing the Azure environment are aware of organization priorities and processes. It helps in defining the organizational conventions that helps in managing resources.

Each policy can be built using multiple rules and multiple policies can be applied to subscription and resource groups. After the rules are executed and when they are satisfied, it executes an action. The action could be to deny the on-going transaction, audit the transaction, which means writing to logs and allowing it to complete, and append metadata to the transaction if the transaction is missing it.

Examples of policies could be related to the naming convention of resources, tagging of resources, the types of resources that can be provisioned, location of resources, or any combination of these rules.

Built-in Policies

Azure provides infrastructure to create custom policies, however, it also provides some out-of-box policies that are frequently needed and used for governance. These policies relate to the allowed location, allowed resource types, and tags. More information for these built-in policies can be found at

<https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-manager-policy>.

Policy Language

Azure policies use JSON language to define and describe policies.

There are two steps in policy adoption. The policy should be defined and then it should be applied and assigned. Policies have scope and can be applied at the subscription and resource group level.

Policies are defined using `if...then` blocks, which is similar to any popular programming language. The `if` block is executed to evaluate the conditions and based on the result from those conditions, the `then` block is executed:

```
{
  "if": {
    "<condition> | <logical operator>"
  },
  "then": {
    "effect": "deny | audit | append"
  }
}
```

Azure policies not only allow simple `if` conditions, but multiple `if` conditions can be joined together logically to create complex rules. These conditions can be joined using **AND**, **OR**, and **NOT** operators.

- The AND syntax requires all conditions to be true
- The OR syntax requires one of the conditions to be true
- The NOT syntax inverts the result of the condition

The AND syntax is shown next. It is represented by the `allOf` keyword:

```
"if": {
  "allOf": [
    {
      "field": "tags",
      "containsKey": "application"
    },
    {
      "field": "type",
      "equals": "Microsoft.Storage/storageAccounts"
    }
  ]
},
```

The OR syntax is shown next. It is represented by the `anyOf` keyword:

```
"if": {
  "anyOf": [
    {
      "field": "tags",
      "containsKey": "application"
    },
    {
      "field": "type",
      "equals": "Microsoft.Storage/storageAccounts"
    }
  ]
},
```

The NOT syntax is shown next. It is represented by the `not` keyword:

```
"if": {
  "not": [
    {
      "field": "tags",
      "containsKey": "application"
    },
    {
      "field": "type",
      "equals": "Microsoft.Storage/storageAccounts"
    }
  ]
},
```

In fact, these logical operators can be combined together as follows:

```
"if": {
  "allOf": [
    {
      "not": {
        "field": "tags",
        "containsKey": "application"
      }
    },
    {
      "field": "type",
      "equals": "Microsoft.Storage/storageAccounts"
    }
  ]
},
```

This is very similar to `if` conditions in popular programming languages such as C# and Node.js

```
If ("type" == "Microsoft.Storage/storageAccounts") {  
    Deny  
}
```

It is to note that there is no `allow` action, although there is a `Deny` action. It means that policy rules should be written with `deny` in perspective. Rules should evaluate the conditions and return `Deny` if they return `true`.

Allowed Fields

The fields that are allowed in policies are as follows:

- Name
- Kind
- Type
- Location
- Tags
- Tags.*
- Property aliases

Azure Locks

Locks are a mechanism to stop certain activities on resources. RBAC provides rights to users/groups/application in a certain scope. There are out-of-the-box RBAC roles, such as owner, contributor, and reader. With the contributor role, it is possible to delete or modify a resource. How can such activities be prevented despite the user having a contributor role? Here enters Azure locks.

Azure locks can help in two ways:

- It can lock resources such that they cannot be deleted even if you have owner access
- It can lock resource in such a way that neither it can be deleted, nor its configuration is modifiable

This is typically very helpful for resources in the production environment where resources should not be modified or deleted accidentally.

Locks can be applied at subscription, resource group, and individual resource levels. Locks follow inheritance between subscription, resource group, and resources. Applying a lock at the parent level will ensure that those at the child level also inherit it. Even resources you add later inherit the lock from the parent. The most restrictive lock in the inheritance takes precedence. Applying a lock at the resource level will also not allow deletion of a resource group containing the resource.

Locks are applied only to operations that help in managing the resource instead of operations that are internal to the resource. Users either need `Microsoft.Authorization/*` or `Microsoft.Authorization/locks/*` RBAC permissions to create and modify locks.

Locks can be created and applied through the Azure portal, Azure PowerShell, Azure CLI, ARM templates, and REST API.

Creating a lock using the ARM template is seen as follows:

```
{
  "$schema":
    "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "lockedResource": {
      "type": "string"
    }
  },
  "resources": [
    {
      "name": "[concat(parameters('lockedResource'),
        '/Microsoft.Authorization/myLock')]",
      "type": "Microsoft.Storage/storageAccounts/providers/locks",
      "apiVersion": "2015-01-01",
      "properties": {
        "level": "CannotDelete"
      }
    }
  ]
}
```

Creating and applying a lock to the resource using PowerShell is seen as follows:

```
New-AzureRmResourceLock -LockLevel CanNotDelete -LockName LockSite `
  -ResourceName examplesite -ResourceType Microsoft.Web/sites `
  -ResourceGroupName exampleresourcegroup
```

Creating and applying a lock to the resource group using PowerShell is seen as follows:

```
New-AzureRmResourceLock -LockName LockGroup -LockLevel CanNotDelete `
  -ResourceGroupName exampleresourcegroup
```

Creating and applying a lock to the resource using Azure CLI is seen as follows:

```
az lock create --name LockSite --lock-type CanNotDelete \
  --resource-group exampleresourcegroup --resource-name examplesite \
  --resource-type Microsoft.Web/sites
```

Creating and applying a lock to the resource group using Azure CLI is seen as follows:

```
az lock create --name LockGroup --lock-type CanNotDelete \
  --resource-group exampleresourcegroup
```

Azure RBAC

Azure provides authentication using Azure AD for its resources. Once authenticated, it needs to be evaluated and decided when the identity should be allowed to access any resource, all resources, or just the selected resource intended for that user. This activity has traditionally been known as authorization. Authorization evaluates whether the given identity has the necessary permissions to access the resource and if it can perform the intended operation. Anybody having access to an Azure subscription should be given just enough permissions so that the job can be performed. There should not be more than the required permissions assigned to identities to ensure that the attack surface remains minimum.

Authorization is popularly also known as Role-based Access Control. RBAC in Azure refers to the assigning of permissions to identities (users/groups/applications) at a scope. The scope could be subscription, resource group, or individual resources.

RBAC helps in creation and assignment of different permissions to different identities. This helps in segregating duties within teams rather than everyone having every permission. It helps in making people responsible for their job because others might not even have access to perform it. It is to be noted that providing access as higher scope automatically ensures that the child resources inherit those permissions. For example, providing read access on a resource group ensures that all resources within it will be with read permissions for the given identity.

Azure provides three general purpose built-in roles. They are as follows:

- Owner role which has full access to all resources
- Contributor role which has access to read/write resources
- Readers role which has only read permissions on resources

There are more roles provided by Azure, but they are resource specific. Examples include network contributor and security manager.

To get all roles provided by Azure for all resources, execute the `Get-AzureRmRoleDefinition` command in the PowerShell console.

Each role definition has certain allowed and not allowed actions. For example, owner role has all actions permitted and none of the actions are prohibited. Prohibited actions take precedence on all actions:

```
PS C:\Users\rimodi> Get-AzureRmRoleDefinition -Name "owner"
Name                : Owner
Id                  : 8e3af657-a8ff-443c-a75c-2fe8c4bcb635
IsCustom            : False
Description         : Lets you manage everything, including access to
resources.
Actions             : {*}
NotActions          : {}
AssignableScopes    : {/}
```

Each role comprises of multiple permissions. Each resource provides a list of operations. The operation supported by a resource can be obtained using the `Get-AzureRmProviderOperation` cmdlet. This cmdlet takes the name of the provider and resource for retrieving the operations:

```
Get-AzureRmProviderOperation -OperationSearchString
"Microsoft.Insights/*"
```


This will result in the following output:

```
PS C:\Users\rimodi> get-AzureRmProviderOperation -
OperationSearchString "Microsoft.Insights/*" | select operation
Operation
-----
Microsoft.Insights/Register/Action
Microsoft.Insights/AlertRules/Write
Microsoft.Insights/AlertRules/Delete
Microsoft.Insights/AlertRules/Read
Microsoft.Insights/AlertRules/Activated/Action
Microsoft.Insights/AlertRules/Resolved/Action
Microsoft.Insights/AlertRules/Throttled/Action
Microsoft.Insights/AlertRules/Incidents/Read
Microsoft.Insights/MetricDefinitions/Read
Microsoft.Insights/eventtypes/values/Read
Microsoft.Insights/eventtypes/digestevents/Read
Microsoft.Insights/Metrics/Read
Microsoft.Insights/LogProfiles/Write
Microsoft.Insights/LogProfiles/Delete
Microsoft.Insights/LogProfiles/Read
Microsoft.Insights/Components/Write
Microsoft.Insights/Components/Delete
Microsoft.Insights/Components/Read
Microsoft.Insights/AutoscaleSettings/Write
Microsoft.Insights/AutoscaleSettings/Delete
Microsoft.Insights/AutoscaleSettings/Read
Microsoft.Insights/AutoscaleSettings/Scaleup/Action
Microsoft.Insights/AutoscaleSettings/Scaledown/Action
Microsoft.Insights/AutoscaleSettings/providers/Microsoft.Insights/Metr
icDefinitions/Read
Microsoft.Insights/ActivityLogAlerts/Activated/Action
Microsoft.Insights/DiagnosticSettings/Write
Microsoft.Insights/DiagnosticSettings/Delete
Microsoft.Insights/DiagnosticSettings/Read
Microsoft.Insights/LogDefinitions/Read
Microsoft.Insights/Webtests/Write
Microsoft.Insights/Webtests/Delete
Microsoft.Insights/Webtests/Read
Microsoft.Insights/ExtendedDiagnosticSettings/Write
Microsoft.Insights/ExtendedDiagnosticSettings/Delete
Microsoft.Insights/ExtendedDiagnosticSettings/Read
```

Custom Roles

Roles are created by combining multiple permissions. For example, a custom role can consist of operations from multiple resources as follows:

```
$role = Get-AzureRmRoleDefinition "Virtual Machine Contributor"
$role.Id = $null
$role.Name = "Virtual Machine Operator"
$role.Description = "Can monitor and restart virtual machines."
$role.Actions.Clear()
$role.Actions.Add("Microsoft.Storage/*/read")
$role.Actions.Add("Microsoft.Network/*/read")
$role.Actions.Add("Microsoft.Compute/*/read")
$role.Actions.Add("Microsoft.Compute/virtualMachines/start/action")
$role.Actions.Add("Microsoft.Compute/virtualMachines/restart/action")
$role.Actions.Add("Microsoft.Authorization/*/read")
$role.Actions.Add("Microsoft.Resources/subscriptions/resourceGroups/read")
$role.Actions.Add("Microsoft.Insights/alertRules/*")
$role.Actions.Add("Microsoft.Support/*")
$role.AssignableScopes.Clear()
$role.AssignableScopes.Add("/subscriptions/c276fc76-9cd4-44c9-99a7-4fd71546436e")
$role.AssignableScopes.Add("/subscriptions/e91d47c4-76f3-4271-a796-21b4ecfe3624")
New-AzureRmRoleDefinition -Role $role
```

How is it Different from RBAC?

Locks are not the same as RBAC. RBAC helps in allowing or denying permissions on resources that are intrinsic to the resource. These permissions relate to performing operations such as read, write, and update on resources. Locks, on the other hand, relates to disallowing permissions to configure or delete the resource.

Examples of Implementing Azure Governance Features

In this section, we will go through a sample architecture implementation for a fictitious organization that wants to implement Azure governance and cost management features.

Background

Company Inc is a worldwide company that is implementing a social media solution on Azure IaaS. They use web servers and application servers deployed on Azure virtual machines and networks. Azure SQL server acts as the backend database.

Role-Based Access Control

The first task is to ensure that the appropriate teams and application owners can access their resources. It is recognized that each team has different requirements. For the sake of understanding, Azure SQL is deployed in a separate resource group compared to the Azure IaaS artifacts.

The administrator assigns the following roles for the subscription:

Role	Assigned to	Description
Owner	The administrator	Manages all resource groups and subscription.
Security manager	Security administrators	This role allows users to look at the Azure security center and the status of the resources.
Contributor	Infrastructure management	Managing virtual machines and other resources.
Reader	Developers	Can view resources, but cannot modify. Developers are expected to work on their development/testing environments.

Summary

Governance and cost management is one of the top priorities for companies moving to the cloud. Having an Azure subscription with pay-as-you-go can harm the company budget because anyone having access to the subscription can provision as many resources they feel like. Some resources are free while others are expensive. It is important for organizations to keep control on their cloud costs. Tags help in generating billing reports. These reports could be based on departments or projects, or owners or any other criteria deemed fit. While cost is important, governance is equally important. Azure provides locks, policies, and RBAC to formulate governance. Policies ensure that resource operations can be denied or audited, locks ensure that resources cannot be modified or deleted, and RBAC ensures that employees have optimal permissions for performing their jobs. With all these four features, companies can have sound governance and cost control on their Azure deployments.

DevOps on Azure will be the focus on next chapter in which some of its important concepts and implementation techniques will be discussed.

17

DevOps on Azure

Software development is a complex undertaking comprising of multiple processes, tools and involves people from different departments. They all need to come together and work in a cohesive manner. With so many variables, the risks are high while delivering to the end customers. One small omission or misconfiguration and the application might come crashing down. This chapter is about adopting and implementing practices that reduce this risk considerably and ensure that high quality software can be delivered to the customer again and again.

Before getting into the details about DevOps, let's understand problems faced by software companies that are addressed by DevOps.

- Organizations are rigid and do not welcome change
- Rigid and time-consuming processes
- Isolated teams working in silos
- Monolithic design and big bang deployments
- Manual execution
- Lack of innovation

In this chapter, we will go detail about the following:

- DevOps
- DevOps practices
- Configuration management
- Continuous integration
- Continuous deployment
- Continuous delivery
- Visual Studio Team Services

- DevOps preparation
- DevOps for PaaS solution
- DevOps for virtual machine (IaaS) based solutions
- DevOps for container (IaaS) based solutions
- Azure automation
- Azure tools for DevOps

What is DevOps?

Today there is no consensus in the industry regarding the definition of DevOps. Every organization has formulated their own definition of DevOps and have tried to implement it. They have their own perspective and think they have implemented DevOps if they have their automation in place, configuration management enabled, by using agile processes or any other combination.

DevOps is about the delivery mechanism of software systems. It is about bringing people together, making them collaborate and communicate, working together towards a common goal and vision. It is about taking joint responsibility, accountability, and ownership. It is about implementing processes that foster collaboration and a service mindset. It enables delivery mechanisms that bring agility and flexibility within the organization. Contrary to the popular belief, DevOps is not about tools, technology, automation. These are enablers that help in collaboration, implementation of agile processes and faster and better delivery to the customer.

There are multiple definitions available on the internet for DevOps and they are neither wrong nor incorrect. DevOps does not provide a framework or methodology. It is a set of principles and practices that, when employed within an organization, engagement, or project, achieves the goal and vision of both DevOps and the organization. These principles and practices do not mandate any specific process, tools and technologies and environment. DevOps provides the guidance which can be implemented through any tool, technology, and process, although some of the technology and processes might be more applicable to achieve the vision of DevOps principles and practices.

Although DevOps practices can be implemented in any organization that provides services and products to customers, going forward in this book, we will look at DevOps from the perspective of software development and the operations department of any organization.

So, what is DevOps? DevOps is defined as:

It is a set of principles and practices bringing both developers and operations teams together from the start of the software system for faster, quicker, and efficient end-to-end delivery of the software system to the end customer, again and again in a consistent and predictable manner reducing time to market, thereby gaining competitive advantage.

Read out loudly the preceding definition of DevOps and if you look at it closely, it does not indicate or refer to any specific processes, tools or technology. It is not prescribing any particular methodology or environment.

The goal of implementing DevOps principles and practices in any organization is to ensure that the demands of stakeholders (including customers) and expectations are met efficiently and effectively.

The customer's demands and expectations are met when:

- Customer gets the features they want
- Customer gets the feature when they want
- Customer gets faster updates on features
- Quality of delivery is high

When an organization can meet the above expectations, customers are happy and remain loyal to the organization. This in turn increases the market competitiveness of the organization which results in a bigger brand and market valuation. It has a direct impact on the top and bottom line of the organization. The organization can invest further on innovation and customer feedback, bringing about continuous changes to its system and services to stay relevant.

The implementation of DevOps principles and practices in any organization is guided by its surrounding ecosystem. This ecosystem is made of the industry and domains the organization belongs to.

DevOps is based on a set of principles and practices. We will look into details about these principles and practices later in this chapter. The core principles of DevOps are the following:

- Agility
- Automation
- Collaboration
- Feedback

The core DevOps practices are the following:

- Continuous integration
- Configuration management
- Continuous deployment
- Continuous delivery
- Continuous learning

DevOps is not a new paradigm, however it is gaining a lot of popularity and traction in recent times. Its adoption is at its highest level and more and more companies are undertaking this journey. I purposely mentioned DevOps as a journey because there are different levels of maturity within DevOps. While successfully implementing continuous deployment and delivery are considered as the highest level of maturity in this journey, adopting source code control, agile software development is considered as a beginning.

One of the first thing DevOps talks about is *breaking the barriers between the developers and the operations team*. It brings about the close collaboration aspect between multiple teams. It is about breaking the mindset that the developer is responsible for writing the code only and passing it on to operations for deployment once it is tested. It is also about breaking the mindset that operations have no role to play in development activities. Operations should influence the planning of the product and should be aware of the features coming up as release. They should also continually provide feedback to the developers on the operational issues such that they can be fixed in subsequent releases. They should influence the design of the system for better operational working of the system. Similarly, the developers should help the operations in deployment of the system and solve incidents when they arise.

The definition talks about *faster, quicker, and efficient end to end delivery of systems to stakeholders*. It does not talk about how fast, quick, or efficient the delivery should be. It should be fast or quick enough depending on the organization domain, industry, customer segmentation, and more. For some organizations, fast enough could be quarterly while for others it could be weekly. Both types are valid for DevOps point of view and they can deploy relevant processes and technologies to achieve the same. DevOps does not mandate it. Organizations should identify the best implementation of DevOps principles and practices based on their overall project, engagement, and organization vision.

The definition also talks about *end-to-end delivery*. It means that from the planning and delivery of the system to the services and operations should be part of the DevOps implementation. The processes should be such that it allows for greater flexibility, modularity, and agility in the application development life cycle. While organizations are free to use the best fit process--**Waterfall, Agile, Kanban**, and more, typically, organizations tend to favor agile processes with iterations-based delivery. This allows for faster delivery in smaller units which are far more testable and manageable compared to large big delivery.

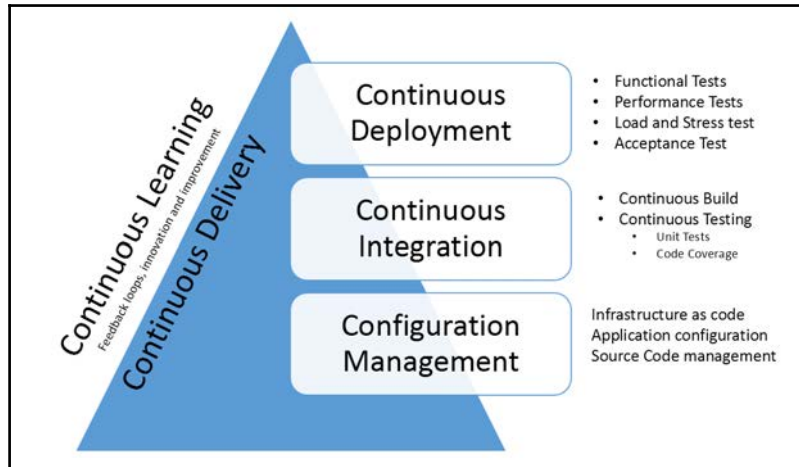
DevOps talks about *end customers again and again in a consistent and predictable manner*. This means that organizations should continually deliver to customers with newer and upgraded features using automation. We cannot achieve consistency and predictability without the use of automation. Manual work should be reduced to none to ensure a high level of consistency and predictability. The automation should also be end to end, to avoid failures. This also indicates that the system design should be modular allowing faster delivery as they are reliable, available, and scalable. Testing plays a great role in consistent and predictable delivery.

The end result of implementing the before-mentioned practices and principles is that the organization is able to meet the expectations and demands of customers. The organization is able to grow faster than the competition and further increase the quality and capability of their product and services through continuous innovation and improvement.

DevOps Practices

DevOps consists of multiple practices each providing distinct functionality to the overall process. The following figure shows the relationship between them. Configuration management, continuous integration, and continuous deployment form the core practices that enables DevOps. When we deliver software services combining these three services, we achieve continuous delivery. Continuous delivery is a capability and maturity of an organization dependent on the maturity of configuration management, continuous integration, and continuous deployment. Continuous feedback at all stages forms the feedback loop that helps in providing superior services to the customers. It runs across all DevOps practices.

Let's deep dive into each of these capabilities and DevOps practices:



Configuration Management

Business application and services need an environment on which they can be deployed. Typically, the environment is an infrastructure comprising multiple servers, compute, network, storage, containers and many more services working together such that business applications can be deployed on top of them. Business applications are decomposed into multiple services running on multiple servers either on-premise or on the clouds and each service has its own configuration along with requirements related to infrastructure configuration. In short, both infrastructure and application is needed to deliver systems to customers and both of them have their own configuration. If the configuration drifts, the application might not work as expected leading to downtime and failure. Moreover, as ALM process dictates the use of multiple stages and environment, an application would be deployed to multiple environments with different configurations. The application will be deployed to the development environment for developers to see the result of their work. The application will be deployed to multiple test environments with different configurations for functional tests, load and stress tests, performance tests, integration tests and more, it would also be deployed to the pre-production environment to conduct user acceptance tests and finally onto the production environment. It is important that an application can be deployed to multiple environments without undertaking any manual changes to its configuration.

Configuration management provides a set of processes and tools and they help in ensuring that each environment and application gets its own configuration. Configuration management tracks configuration items and anything that changes from environment to environment should be treated as a configuration item. Configuration management also defines the relationships between the configuration items and how changes in one configuration item will impact the other configuration item.

Configuration management helps in the following:

- **Infrastructure as Code:** When the process of provisioning of infrastructure and its configuration is represented through code and the same code goes through the application life cycle process, it is known as Infrastructure as Code. Infrastructure as code helps in automating the provisioning and configuration of infrastructure. It also represents the entire infrastructure in code that can be stored in a repository and version controlled. This allows the users to use the previous environment configurations when needed. It also enables provisioning of an environment multiple times in a consistent and predictable manner. All environments provisioned through this way are consistent and equal in all ALM stages.
- **Deployment and configuration of application:** Deployment of an application and its configuration is the next step after provisioning of infrastructure. Examples of application deployment and configuration is to deploy a `webdeploy` package on a server, deploy a SQL server schemas and data (bacpac) on another server, change the SQL connection string on the web server to represent the appropriate SQL server. Configuration management stores values for the application configuration for each environment on which it is deployed.

The configuration applied should also be monitored. The expected and desired configuration should be consistently maintained. Any drift from this expected and desired configuration would render the application as not available. Configuration management is also capable of finding the drift and re-configuring the application and environment to its desired state.

With automated configuration management in place, nobody in the team has to deploy and configure the environments and applications on production. The operations team is not reliant on the development team or long deployment documentation.

Another aspect of configuration management is source code control. Business applications and services comprise code and other artifacts. Multiple team members work on the same files. The source code should be up to date at any point of time and should be accessible by only authenticated team members. The code and other artifacts by themselves are configuration items.

Source control helps in collaboration and communication within the team since everybody is aware of what the other person is doing and conflicts are resolved at an early stage.

Configuration management can be broadly divided into two categories:

- Inside the virtual machine
- Outside the virtual machine

The tools available for configuration management inside the virtual machine are discussed next.

Desired State Configuration

Desired State Configuration (DSC) is a new configuration management platform from Microsoft build as an extension to PowerShell. DSC was originally launched as part of WMF 4.0. It is available as part of **Windows Management Framework (WMF)** 4.0 and 5.0 for all Windows server operating systems preceding Windows 2008 R2. WMF 5.1 is available out of the box on Windows Server 2016, and Windows 10.

Chef, Puppet, and Ansible

Apart from DSC, there are a host of configuration management tools, such as Chef, Puppet, and Ansible supported by Azure. Details about these tools are not covered in this book.

The tools available for configuration management outside of a virtual machine are:

Azure Resource Manager Templates

ARM templates are the primary means of provisioning resources in ARM. ARM templates provide a declarative model through which resources, their configuration, scripts, and extensions are specified. ARM templates are based on **JavaScript Object Notation (JSON)** format. It uses the JSON syntax and conventions to declare and configure resources. JSON files are text based, user friendly, and easily readable files. They can be stored in a source code repository and have version control on them. They are also means to represent infrastructure as code that can be used to provision resources in Azure resource groups again and again, predictably, consistently, and uniformly. A template needs a resource group for deployment. It can only be deployed to a resource group and the resource group should exist before executing template deployment. A template is not capable of creating a resource group.

Templates provide the flexibility to be generic and modular in their design and implementation. Templates provide the ability to accept parameters from users, declare internal variables, help in defining dependencies between resources, links resources within the same or different resource groups, and also execute other templates. They also provide scripting language type expressions and functions that make them dynamic and customizable at runtime.

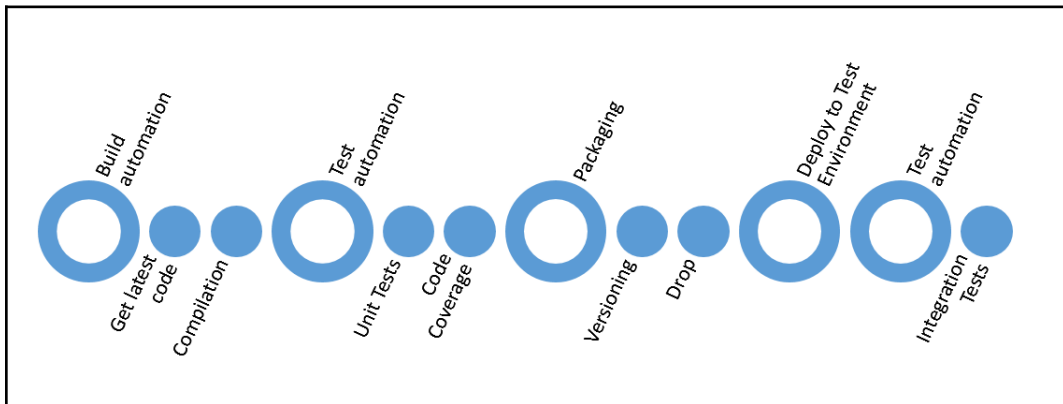
Continuous Integration

Multiple developers write code that is eventually stored in a common repository. The code is normally checked in or pushed to the repository when the developer has finished developing his feature. This can happen in a day or might take days or weeks. Some of the developers might be working on the same feature and they might also follow the same practices of pushing/checking-in code in days or weeks. This can cause issues with the quality of code. One of the tenet of DevOps is to fail fast. Developers should check-in/push their code to the repository often and compile the code to check if he/she has not introduced any bug and that the code is compatible with the code written by his/her fellow member. If the developer does not follow this practice, then the code on their machine will grow very large and will be difficult to integrate with other's code. Moreover, if the compile fails, it is difficult and time consuming to fix the issues arising out of it.

Continuous integration solves these kinds of challenges. Continuous integration helps in compilation and validation of the code pushed/checked-in by a developer by taking it through a series of validation steps. Continuous integration creates a process flow consisting of multiple steps. Continuous integration is comprised of continuous automated build and continuous automated tests. Normally the first step is compilation of the code. After successful compilation, each step is responsible for validating the code from a specific perspective. For example, unit tests can be executed on the compiled code, code coverage can be executed to check which code paths are executed by unit tests. These could reveal if comprehensive unit tests are written or there is scope to add further unit tests. The end result of continuous integration is deployment packages that can be used by continuous deployment for deploying them to multiple environments.

Developers are encouraged to check-in their code multiple times in a day instead of doing so after days or weeks. Continuous integration would initiate the execution of the entire pipeline immediately as soon as the code is checked-in or pushed. If compilation succeeds, code tests and other activities that are part of the pipeline are executed without error, the code is deployed to a test environment and integration tests are executed on it. Although every system demands its own configuration of continuous integration, a minimal sample continuous integration is shown in following figure.

Continuous integration increases the productivity of the developers. They do not have to manually compile their code, run multiple types of tests one after another, and then create packages out of it. It also reduces the risk of getting bugs introduced in the code and the code does not get stale. It provides early feedback to the developers about the quality of their code. Overall, the quality of deliverables is high and deliverables are delivered faster by adopting continuous integration practice:



Build Automation

Build automation consists of multiple tasks executing in sequence. Generally, the first task is responsible for fetching latest source code from the repository. The source code might comprise multiple projects and files. They are compiled to generate artifacts like executable, dynamic link libraries, assemblies and more. Successful build automation reflects that there are no compile time errors in code.

There could be more steps in build automation depending on the nature and type of project.

Test Automation

Test automation consists of tasks that are responsible for validating different aspects of code. These tasks are related to testing code from a different perspective and are executed in sequence. Generally, the first step is to run a series of unit tests on the code. Unit testing refers to the process of testing the smallest denomination of a feature validating its behavior in isolation from other features. It can be automated or manual however the preference is towards automated unit testing.

Code coverage is another type of automated testing that can be executed on the code to find out how much of code is executed while running the unit tests. It is generally represented as a percentage and refers to how much code is testable through unit testing. If the code coverage is not close to 100%, it is either because the developer has not written unit tests for that behavior or the uncovered code is not required at all.

Successful execution of test automation resulting in no significant code failure should start executing the packaging tasks. There could be more steps in test automation depending on the nature and type of project.

Packaging

Packaging refers to the process of generating deployable artifacts such as MSI, NuGet and `webdeploy` packages, database packages, versioning them, and storing them at location such that they can be consumed by other pipelines and processes.

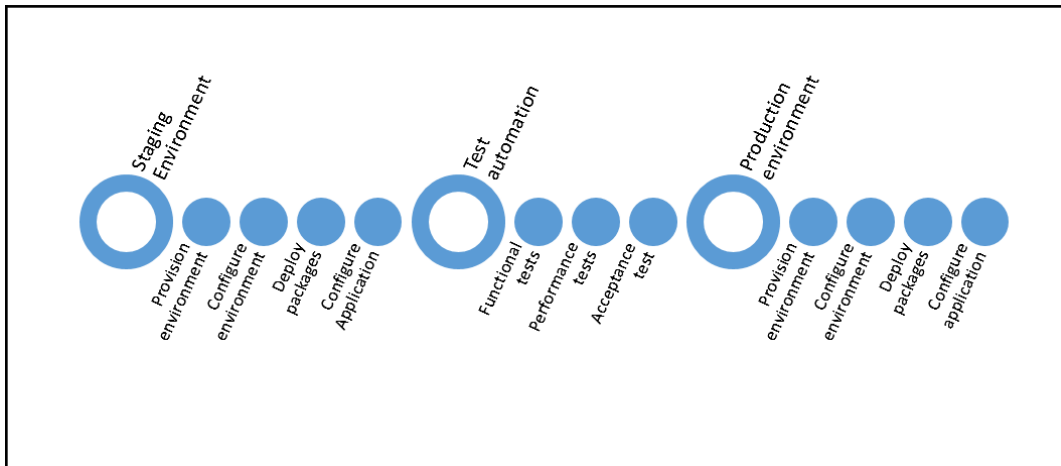
Continuous Deployment

By the time the process reaches continuous deployment, continuous integration has ensured that we have fully working bits of an application that can now be taken through different continuous deployment activities. Continuous deployment refers to the capability of deploying business applications and services to pre-production and production environments through automation. For example, continuous deployment could provision and configure the pre-production environment, deploy application to it and configure. After conducting multiple validations, such as functional tests, performance tests on pre-production environment, the production environment is provisioned, configured and the application is deployed to production environments through automation. There are no manual steps in the deployment process. Every deployment task is automated. Continuous deployment can provision the environment and deploy the application for a bare metal deployment while it could reuse the existing environment and conduct only the application deployment if the environment already exists. It is always better to conduct bare metal green field deployment however; business justification can make the demand for brown field deployments.

All the environments are provisioned through automation using Infrastructure as Code. This ensures that all environments, whether it's development, test, pre-production, production and any other environment is the same. Similarly, the application is deployed through automation ensuring that it is also deployed uniformly across all environments. The configuration across these environments could be different for the application.

Continuous deployment is generally integrated with continuous integration. When continuous integration has done its work by generating the final deployable packages, continuous deployment kicks in and starts its own pipeline. This pipeline is called the **release pipeline**. Release pipeline consists of multiple environments with each environment consisting of tasks responsible for provision of environment, configuration of environment, deploying applications, configuring applications, executing operational validation on environments and testing the application on multiple environments. We will look at release pipeline in greater details in the next chapter and also the chapter on continuous deployment.

Employing continuous deployment provides immense benefits. There is a high level of confidence in the overall deployment process which helps in faster and risk-free releases on production. The chances of anything going wrong comes down drastically. The team would have lower stress levels and rollback to the previous working environment is possible if there are issues in the current release:



Although every system demands its own configuration of release pipeline, a minimal sample of continuous deployment is shown in the preceding figure. It is important to note that generally provisioning and configuring of multiple environments is part of the release pipeline and approvals should be sought before moving to the next environment. The approval process might be manual or automated depending on the maturity of the organization.

Test Environment Deployment

The release pipeline starts once the drop is available from continuous integration and the first step it should take is to get all the artifacts from the drop. After which, it might create a completely new bare metal test environment or reuse an existing one. This is again dependent on the type of project and nature of testing planned to be executed on this environment. The environment is provisioning and configured. The application artifacts are deployed and configured.

Test Automation

After deploying an application, a series of tests can be performed on the environment. One of the tests executed here is functional tests. Functional tests are primarily aimed at validating the feature completeness and functionality of the application. These tests are written from requirements gathered from the customer. Another set of tests that can be executed are related to scalability and availability of the application. This typically includes load test, stress tests, and performance tests. It should also include operational validation of the infrastructure environment.

Staging Environment Deployment

This is very similar to the test environment deployment with the only difference is that the configuration values for the environment and application would be different.

Acceptance Tests

Acceptance tests are generally conducted by stakeholders of the application and this can be manual or automated. This step is a validation from the customer point of view about the correctness and completeness of application functionality.

Deployment to Production

Once the customer provides its approval, the same steps as that of the test and staging environment deployment are executed, with the only difference that the configuration values for the environment and application are specific to the production environment. A validation is conducted after deployment to ensure that the application is running according to expectation.

Continuous Delivery

Continuous delivery and continuous deployment might sound similar to many readers; however, they are not the same. While continuous deployment talks about deployment to multiple environments and finally to the production environment through automation, continuous delivery practices is the ability to generate application packages in a way that are readily deployable in any environment. For generating artifacts that are readily deployable, continuous integration should be used to generate the application artifacts, a new or existing environment should be used for deploying these artifacts, conduct functional tests, performance tests and user acceptance tests through automation. Once these activities are successfully executed with no errors, the application package is referred as readily deployable. Continuous delivery comprises continuous integration along with deployment to an environment for final validations. It helps in getting feedback faster from both the operations as well as from the end user. This feedback can then be used to implement subsequent iterations.

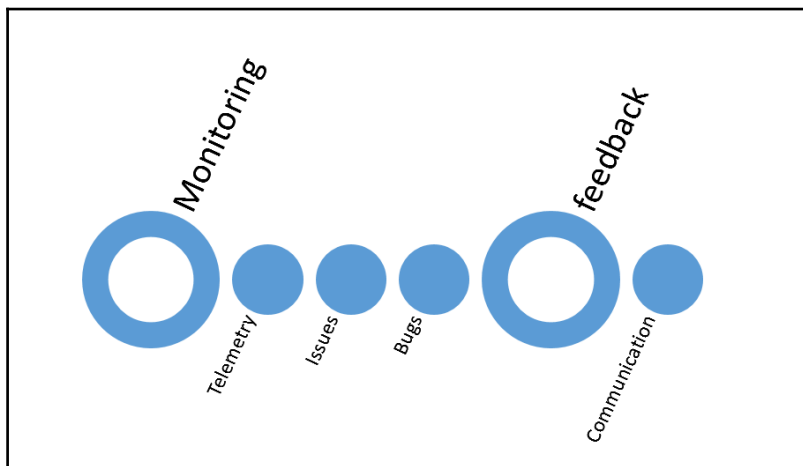
Continuous Learning

With all the before mentioned DevOps practices, it is possible to create great business applications and deploy them automatically to the production environment, however, benefits of DevOps will not last for long, if continuous improvement and feedback principles are not in place. It is utmost important that real-time feedback about the application behavior is passed on as feedback to the development team from both end users and the operations team.

Feedback should be passed to the teams providing relevant information about what is going well and importantly, what is not going well.

Applications should be built with monitoring, auditing, and telemetry in mind. The architecture and design should support these. The operations team should collect the telemetry information from the production environment, capture any bugs and issues and pass it on the development team such that they can get fixed in subsequent releases. The same has been shown in the following figure.

Continuous learning helps in making the application robust and resilient to failures. It helps in making sure that the application is meeting the requirements of the consumers:



Visual Studio Team Services

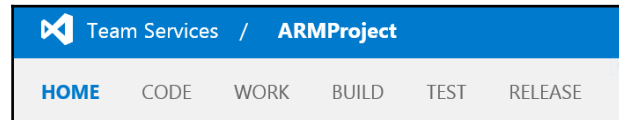
Now, it's time to focus on another revolutionary online service, **Visual Studio Team Services (VSTS)**, that enables continuous integration, continuous deployment and continuous delivery seamlessly. In fact, it would be more appropriate to call it a suite of services available under a single name. VSTS is a PaaS provided by Microsoft and hosted on the cloud. The same service is available as **Team Foundation Services (TFS)** on-premise. All examples used in this book uses VSTS.

According to Microsoft, VSTS is a cloud-based collaboration platform that helps teams in sharing code, tracking work and shipping software. VSTS is the new name and earlier it was known as **visual studio online (VSO)**. VSTS is an enterprise software development tool and service that enables organizations in providing automation facilities to their end-to-end application life cycle management process from planning to deployment of application and getting real-time feedback from software systems. This increases the maturity and capability of an organization to deliver high quality software systems to their customers again and again.

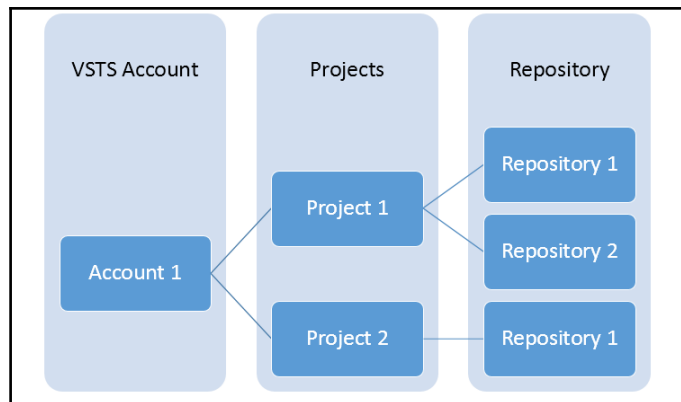
Successful software delivery involves efficiently bringing numerous processes and activities together. These include executing and implementing various agile processes, increasing collaboration among teams, seamless, and automatic transition of artifacts from one phase of the ALM to another phase, and deployments to multiple environment. It is important to track and report on these activities to measure and take action and improve delivery processes. VSTS makes it simple and easy. It provides a whole suite of services that enables the following:

- Collaboration among every team member by providing a single interface for the entire application life cycle management.
- Collaboration among development teams using source code management services
- Collaboration among test teams using test management services
- Automation validation of code and packaging through continuous integration using build management services
- Automating validation of application functionality, deployment, and configuration of multiple environments through continuous deployment and delivery using release management services
- Tracking and work item management using work management services

The following figure shows all the services available from the VSTS top navigation bar:



A project in VSTS is a security boundary and logical container that provides all the services we mentioned in the previous section. VSTS allows for the creation of multiple projects within a single account. By default, a repository is created with the creation of a project, however, VSTS allows for the creation of additional repositories within a single project. The relationship between the VSTS account, project and repository is shown in the following figure:



Relationship between VSTS Account, projects and repositories

VSTS provides two types of repositories

- GIT
- **Team Foundation Version Control (TFVC)**

It also provides the flexibility to choose between GIT or TFVC source control repository. There can be a combination of TFS and TFVC repositories available within a single project.

Team Foundation Version Control

TFVC is the traditional and centralized way of implementing version control in which there is a central repository and developers work on it directly in connected mode to check-in their changes. If the central repository is offline or not available, developers cannot check-in their code and have to wait for it to be online and available. Other developers can see only the checked in code. Developers can group multiple changes into a single change set for checking-in code changes that are logically grouped to form a single change. TFVC locks the code files that are undergoing edits. Other developers can read the locked-up file but they cannot edit it. They must wait for the prior edit to complete and release the lock before they can edit. The history of check-ins and changes are maintained on the central repository while the developers have the working copy of the files but not the history.

TFVC works very well with large teams that are working on the same projects. This allows control over the source code at a central location. It also works the best when the project is for a long duration since the history is managed at a central location. TFVC has no issues working with large and binary files.

GIT

GIT on the other hand is a modern distributed way of implementing version control where the developers can work on their own local copies of code and history in offline mode. Developers can work offline on their local clone of code. Each developer has a local copy of code and entire history and they work on their changes with this local repository. They can commit their code to the local repository. They can connect to the central repository for synchronization of their local repository on a need basis. This allows every developer to work on any file since they would be working on their local copy. Branching in GIT does not create another copy of the original code and is extremely fast to create.

GIT works well with a smaller team. With larger teams, there is a substantial overhead to manage multiple pull requests to merge the code on a central repository. It also works best for smaller duration project as the history would not get too large to be downloaded and manageable on every developer's local repository. Branching and merging is a breeze with advance options.

GIT is the recommended way of using source control because of the rich functionality it provides. We will use GIT as the repository for our sample application in this book.

Preparing for DevOps

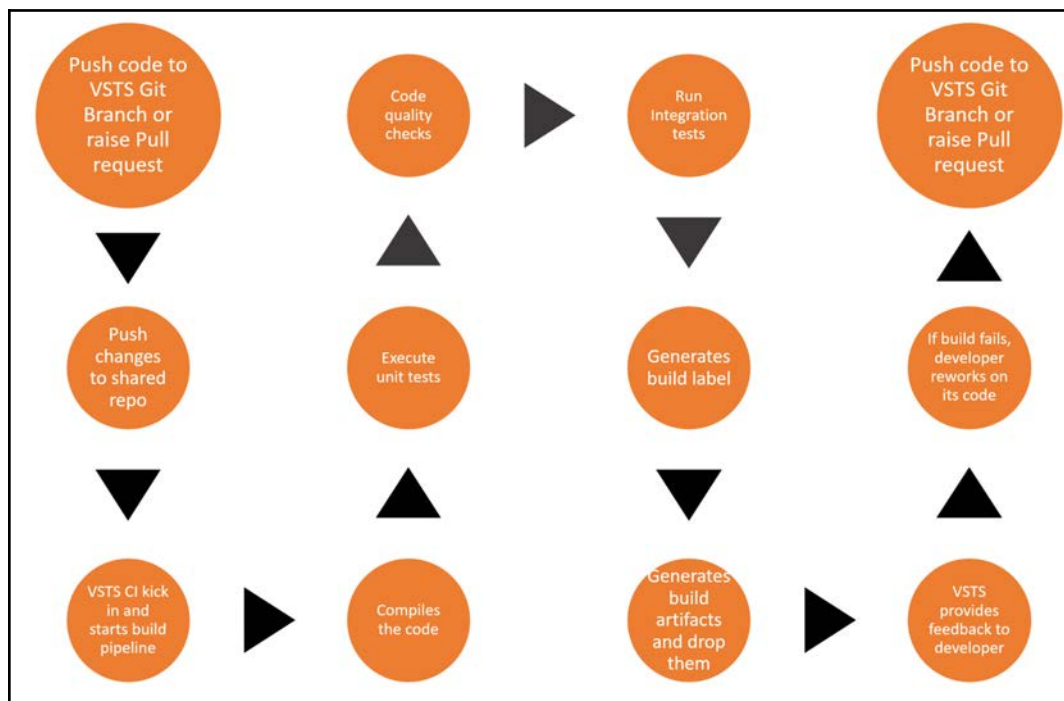
In this chapter from henceforth, focus will be on process and deployment automation using different patterns in Azure. These include the following:

- DevOps for IaaS solutions
- DevOps for PaaS solutions
- DevOps for container based solution

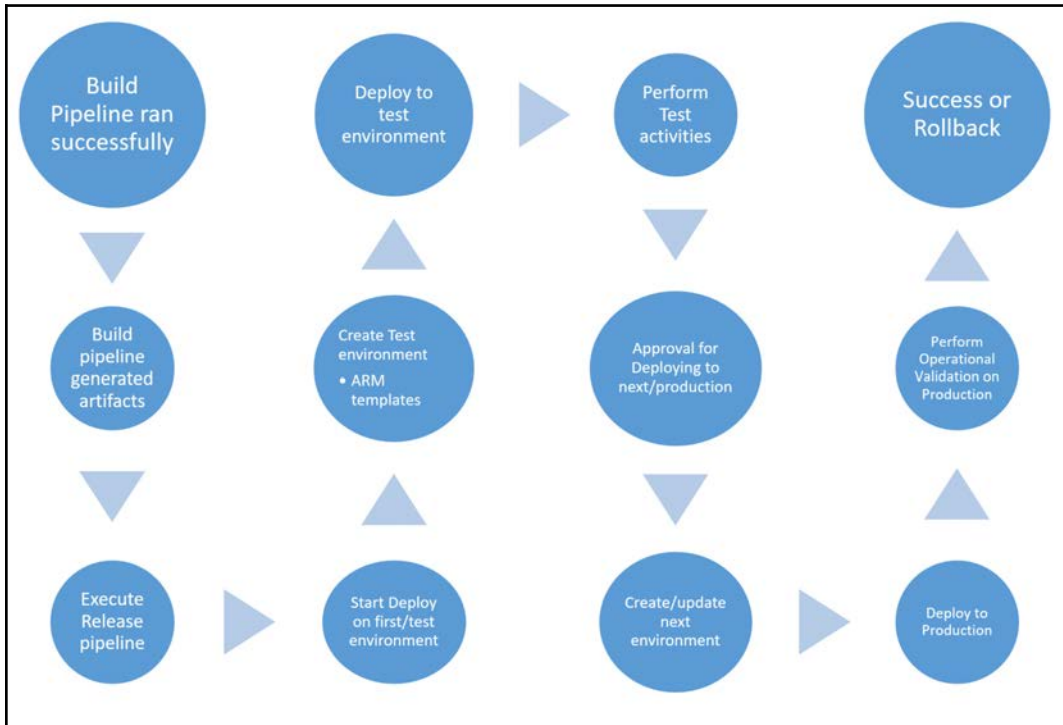
Generally, there are common shared services not unique to any one application. Their services are consumed by multiple applications from different environments such as development, testing, and production. The life cycle of these common shared services is different from other applications. Therefore, they have different version control accounts, a different codebase, build and release management. They have their own cycle of plan-design-build-test and release.

The resources that are part of this group are provisioned using ARM templates, PowerShell, and DSC configurations.

The overall flow for building these common components is shown next:



The release process is shown in the following diagram:



To get on the DevOps journey, it is important to understand and provision the common components and services before starting any software engagement, product, or service.

Provision VSTS Account

A version control system is needed to collaborate at code level. VSTS helps in providing both centralized and decentralized versions of control systems. VSTS also provides orchestration services for building and executing build and release pipelines. It is a mature platform to organize all DevOps related version control, build and release and work-item related artifacts. After account is provisioned, a VSTS project should be created to hold all project related artifacts.

A VSTS account can be provisioned by visiting <https://www.visualstudio.com>.

Provision Azure Key Vault

It is not advisable to store secrets, certificates, credentials, and other sensitive information in code configuration files in files, databases or any other general storage system. It is advised to store this important data into a vault that is specifically designed for storing secrets and credentials. Azure key vault provides such a service. Azure key vault is available as a resource and service from Azure.

Provisioning a Configuration Management Server

A configuration management server providing storage for configurations and applying those configurations to different environments is always a good strategy for automating deployments. DSC on custom virtual machines, DSC from Azure automation, Chef, Puppet, Ansible are some of the options and can be used on Azure seamlessly for both Windows as well as Linux environments. This book uses DSC as a configuration management tool for all purposes and it provides a pull server that holds all configuration documents (MOF files) for the sample application. It also maintains the database of all virtual machines and containers that are configured and registered with pull server to pull configuration documents from it. The local configuration manager on these target virtual machines and containers periodically checks the availability of new configurations as well as drifts in current configuration and reports back the same to the pull server. It also has inbuilt reporting capabilities that provides information about nodes that are compliant as well as those which are non-compliant within a virtual machine. A pull server is a general web application hosting DSC pull server endpoint.

Provisioning Log Analytics

Log analytics is an audit and monitoring service provided by Azure to get real time information about all changes, drifts, events occurring within virtual machines and containers. It provides a centralized workspace and dashboard for IT administrators for viewing, searching, and conducting drill-down searches on all changes, drifts and events occurring on these virtual machines. It also provides agents that are deployed on target virtual machines and containers. Once deployed, these agents start sending all changes, events and drifts to the centralized workspace.

Azure Storage Account

Azure storage is a service provided by Azure to store files as blobs. All scripts and code for automating the provisioning, deployment and configuration of infrastructure and sample application are stored in a VSTS `git` repository and are packaged and deployed in an Azure storage account. Azure provides PowerShell script extension resources that can automatically download DSC and PowerShell scripts and execute them on virtual machines during execution of Azure resource manager templates. This storage acts as a common storage across all deployments for multiple applications.

Images

Both virtual machine and container images should be built as part of the common services build and release pipeline. Tools such as Packer, Docker build can be used to generate these images.

Monitoring Tools

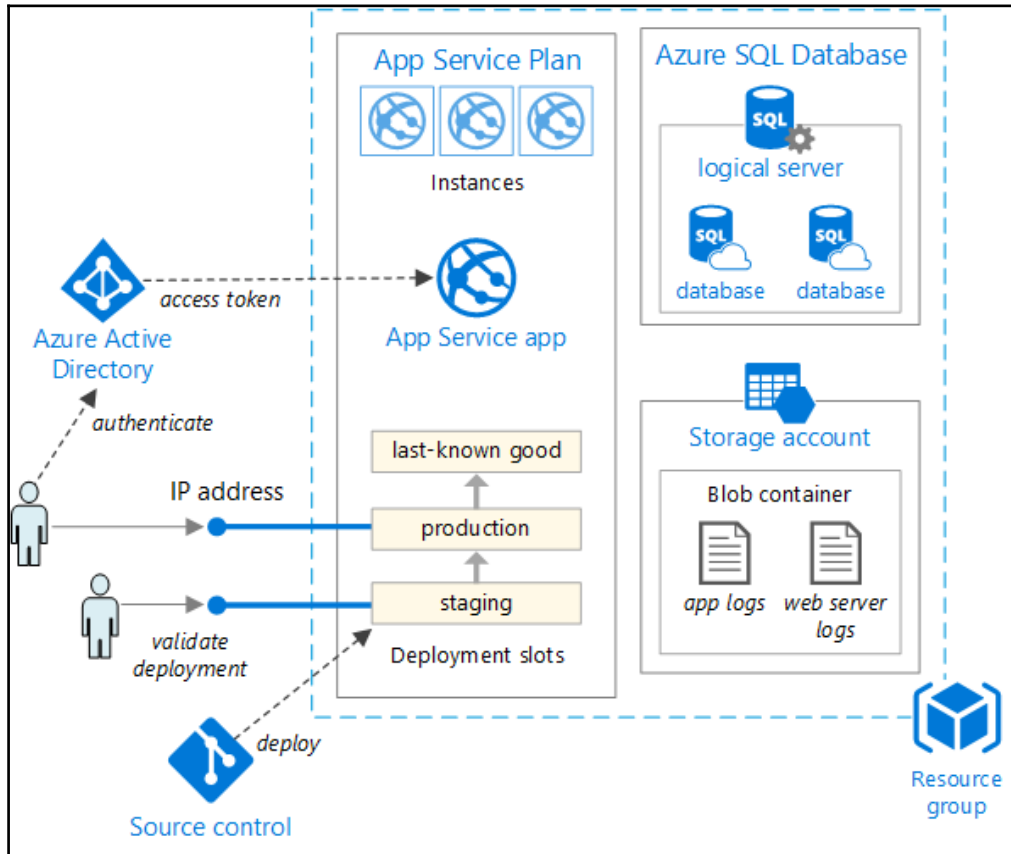
All monitoring tools, such as Azure Monitor, application insights, log analytics, OMS, system center operations manager should be provisioned and configured during the release pipeline of common services.

Management tools

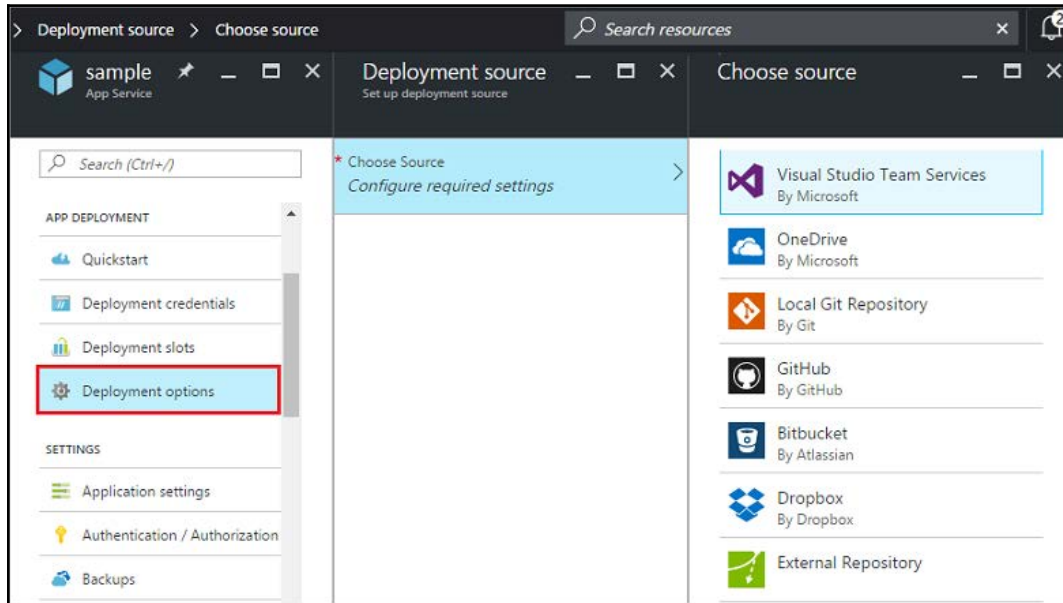
All management tools, such as Kubernetes, DC/OS, Docker Swarm, ITIL tools should be provisioned at this stage.

DevOps for PaaS Solutions

The typical architecture for Azure PaaS app services is based on the solution shown next:



The architecture shows some of the important components, such as Azure SQL, storage accounts, version control system, and more that participate in Azure app services based cloud solution architecture. These artifacts should be created using Azure resource manager templates. These ARM templates should be part of the overall configuration management strategy. It can have its own build and release management pipelines very similar to the one shown in the previous section:



The ARM template should also configure continuous deployment by configuring **Deployment options**.

Azure App Services

Azure app service provides managed hosting services for cloud solutions. It is a fully managed platform for provisioning and deploying cloud solutions. App services takes away the burden of creating and managing infrastructure and provides minimum **service level agreements (SLA)** for hosting your cloud solutions. They are open and let users decide the choice of language they want to use for building their cloud solutions and flexible to host the cloud solution on either Windows or Linux operating systems. Creating an app service and hosting cloud solutions provision virtual machines behind the scene that are completely managed by Azure, and users do not see them at all. Multiple different types of cloud solutions such as web application, mobile backend API's, API endpoints, containers can be hosted seamlessly on Azure app services.

Deployment Slots

Azure app service provides deployment slots that makes deployment to them seamlessly and easy. There is a production and staging slot and it lets users swap them easily. This helps in first deploying the custom cloud solution to staging and after all checks and tests, they can be swapped to production if found satisfactory. However, in case of any issue in production after swapping, the previous good values from the production environment can be reinstated by swapping again.

Azure SQL

Azure SQL is SQL PaaS service provided by Azure to host databases. Azure provides a secure platform to host databases and takes complete ownership to manage the availability, reliability and scalability of the service. With Azure SQL, there is no need to provision custom virtual machines, deploy a SQL server and configure it. Instead, Azure team does these behind the scenes and manages them on our behalf. It also provides a firewall service enabling security and only an IP address allowed by the firewall can connect the server and access the database. The virtual machines provisioned to host web applications have distinct Public IP addresses assigned to them and they are added to Azure SQL firewall rules dynamically. Azure SQL server and its database is created while executing the Azure resource manager template. It is to be noted that generally management virtual machines should execute such tasks.

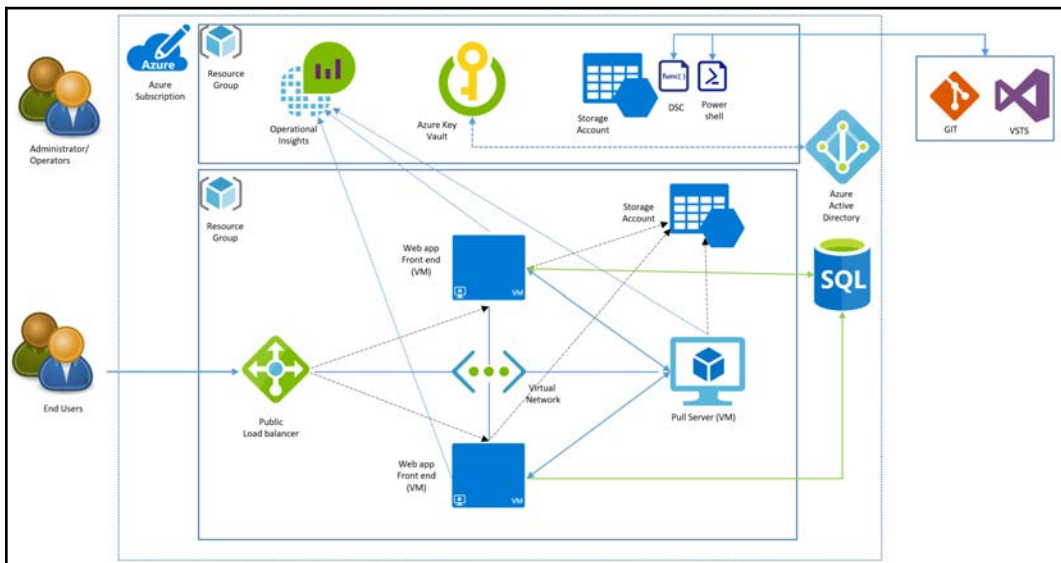
Build and Release Pipeline

Azure app services provide out of box services for continuous deployment and they should be used for deploying custom cloud solutions as shown next. Details about account, repository, branch should be provided. Azure app services would connect to the repository and pull the latest source code from the provided branch and execute the build steps of compiling and generating deployable artefacts. It will also deploy to staging deployment slots automatically.

Visual Studio at the developer's end should be configured to connect to the version control system. The build process should start by Azure app services, whenever a developer checks in or pushes his code to the repository branch.

DevOps for Virtual Machine (IaaS) Based Solutions

The typical architecture for IaaS virtual machine based solution is shown next:



Azure Virtual Machine

Azure virtual machines hosting web applications, application servers, database, and other services are provisioned using ARM templates. Each virtual machine has a single network card with a public IP assigned to it. They are attached to a virtual network and have a private IP address from the same network. The public IP for virtual machines is optional since they are attached to a public load balancer. These virtual machines are based on a Windows 2016 server image. Operational insights agents are installed on virtual machines for monitoring the virtual machines. PowerShell scripts are also executed on these virtual machines downloaded from a storage account available in an other resource group to open relevant firewall ports, download appropriate packages and install local certificates to secure access through PowerShell. The web application is configured to run on the provided port on these virtual machines. The port number for the web application and all its configuration is pulled from the DSC pull server and dynamically assigned.

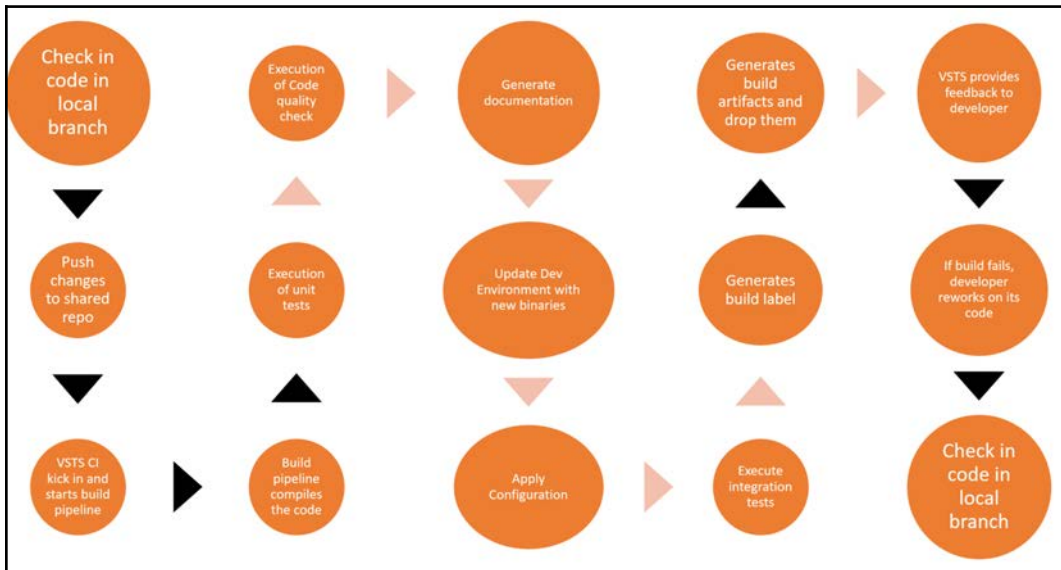
Azure Public Load Balancer

A public load balancer is attached to some of the virtual machines for sending requests to them in a round robin fashion. This is generally needed for front-end web applications and APIs. A public IP address and DNS name can be assigned to load balancer such that it can serve internet requests. It accepts HTTP web requests on different port and routes the same to the virtual machines. It also probes on certain ports on HTTP protocols with some provided application paths. **Network Address Translation (NAT)** rules can also be applied such that they can be used to login into the virtual machines using remote desktops.

An alternative resource to Azure public load balancer is the Azure application gateway and depending on the scenario, can be used and deployed.

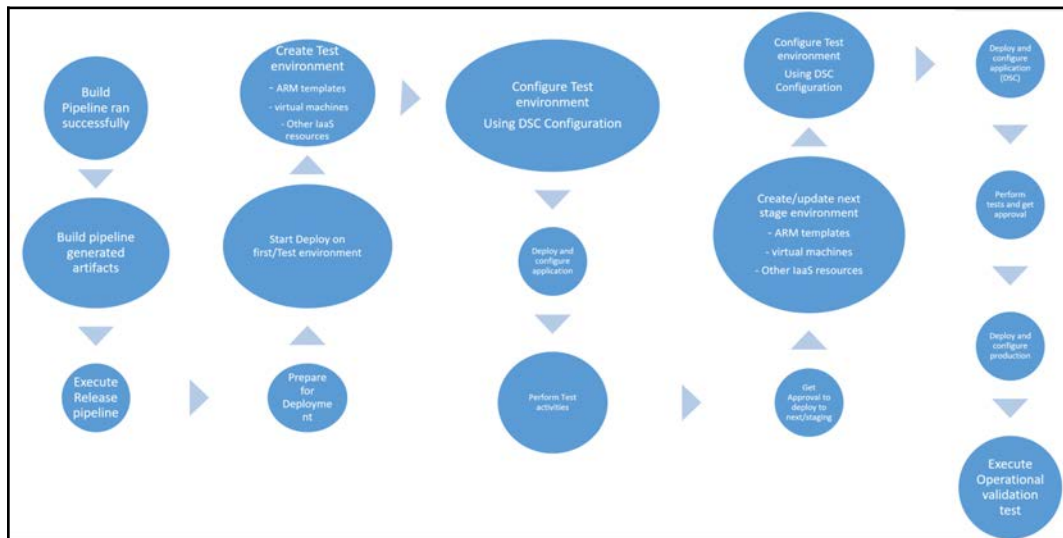
Build Pipeline

A typical build pipeline for an IaaS virtual machine based solution is shown next. A release pipeline starts when a developer pushes his code to the repository. The build pipeline starts automatically as part of continuous integration. It compiles and builds the code, executes unit tests on it, code quality checks and generates documentation from code comments. It deploys the new binaries into the dev environment (note that development environment is not newly created), changes configuration, executes integration tests and generated build labels for easy identification. It then drops the generated artifacts to a location accessible by the release pipeline. In case there are issues during the execution of any step in this pipeline, the same is communicated to the developer as part of build pipeline feedback such that they can re-work and submit their changes again. The build pipeline should fail or pass based on the severity of issues found and that is dependent on organization to organization:



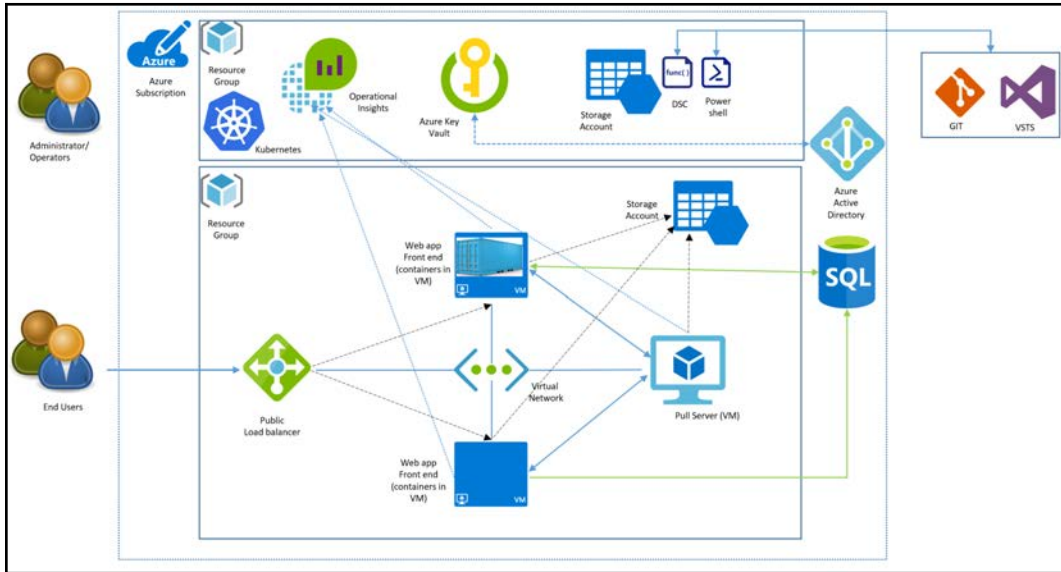
Release Pipeline

A typical release pipeline for an IaaS virtual machine based deployment is shown next. A release pipeline starts after completion of the build pipeline. The first step for the release pipeline is to gather the artifacts generated by the build pipeline. They are generally deployable assemblies, binaries and configuration documents. The release pipeline executes and creates or updates the first environment which generally is a test environment. It uses Azure Resource Manager templates to provision all IaaS and PaaS services and resources on Azure and configures them as well. They also help executing scripts and DSC configuration after virtual machines are created as post-creation steps. This helps configuring the environment within the virtual machine and the operating system. At this stage, application binaries from the build pipeline are deployed and configured. Different automated tests are performed to check the working on solution and if found satisfactory, the pipeline moves to deployment to next environment after obtaining necessary approvals. The same steps are again executed on the next environment including the production environment. At last, operational validation tests are executed on production to ensure that the application is working as expected and there are no deviations. At this stage, if there are any issues or bugs, they should be rectified and the entire cycle should be repeated; however if it does not happen within a stipulated timeframe, the previous known snapshot should be restored on the production environment to minimize downtime:



DevOps for Container Based (IaaS) Solutions

The typical architecture for IaaS container based solution is shown next:



Containers

Containers is also a virtualization technology, however, they do not virtualize physical servers. Instead, containers is an operating system level virtualization. What it means is that containers share the operating system kernel provided by its host among themselves along with the host. Running multiple containers on a host (physical or virtual) share the host operating system kernel. There is a single operating system kernel provided by the host and used by all containers running on top of it.

Containers are also completely isolated from their host and other containers much like a virtual machine. Containers uses Windows storage filter drivers and sessions isolation for providing isolation of operating system services, such as filesystems, registry, processes, and networks. Each container gets its own copy of operating system resources.

Docker

Docker provides management features to Windows containers. It comprises two executables:

- Docker daemon
- Docker client

Docker daemon is the workhorse for managing containers. It is a Windows service responsible for managing all activities on the host related to containers. Docker client interacts with Docker daemon and is responsible for capturing inputs and sending them across to Docker daemon. Docker daemon provides the runtime, libraries, graph drivers, engines to create, manage and monitor containers, and images on the host server. It also provides capabilities to create custom images that are used for building and shipping applications to multiple environments.

DockerFile

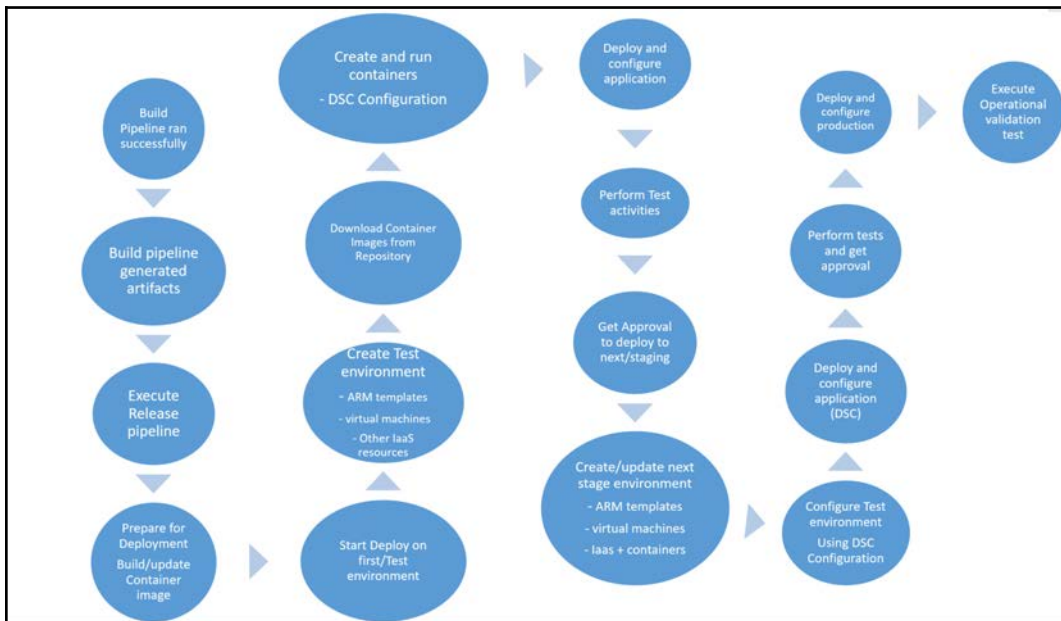
DockerFile is a primary building block for creating Windows container images. It is a simple text based human readable file without any extension and is named DockerFile. Although there is a mechanism to name it differently, generally it is named as DockerFile. DockerFile contains instructions to create a custom container image from base image. These instructions are executed sequentially from top to bottom by the Docker daemon, the engine behind all activities related to Windows containers. The instructions refer to the command and their parameters understood by Docker daemon. DockerFile enables infrastructure as code practices by converting the application deployment and configuration into instructions that can be versioned and stored in a source code repository.

Build Pipeline

There is no difference from the build perspective between the container and a virtual machine based solution. The build step remains the same. Please refer to the earlier section for build pipeline details.

Release Pipeline

A typical release pipeline for an IaaS container based deployment is shown next. The only difference between this and the release pipeline is container image management and creation of containers using DockerFile and Docker compose. Advance container management utilities, such as Docker Swarm, DC/OS, and Kubernetes can also be deployed and configured as part of the release management. However, it is to be noted that these container management tools should be part of the shared services release pipeline as discussed earlier:



Azure Automation

Azure automation is Microsoft's platform for all automation implementation with regard to cloud, on-premise and hybrid deployments. Azure automation is a mature automation platform that provides rich capabilities in terms of the following:

- Defining assets, such as variables, connections, credentials, certificates, and modules
- Implementing runbooks using Python, PowerShell scripts, and PowerShell workflows
- Providing user interfaces to create runbooks
- Managing the full runbook life cycle including build, test, and publish
- Scheduling of runbooks
- Ability to run runbooks anywhere--on cloud or on-premise
- DSC as a configuration management platform
- Manage and configure environments, machines--Windows and Linux, applications, and deployment
- Ability to extend Azure automation by import custom modules

Azure automation provides a DSC pull server that helps in creating a centralized configuration management server consisting of configurations for nodes/virtual machines and their constituents.

It implements the hub and spoke pattern wherein nodes can connect to the DSC pull server and download configurations assigned to them and re-configure themselves to reflect their desired state. Any changes and deviation within these nodes are auto-corrected by DSC agents the next time they run. This ensures that administrators do not need to actively keep monitoring the environment to find any deviations.

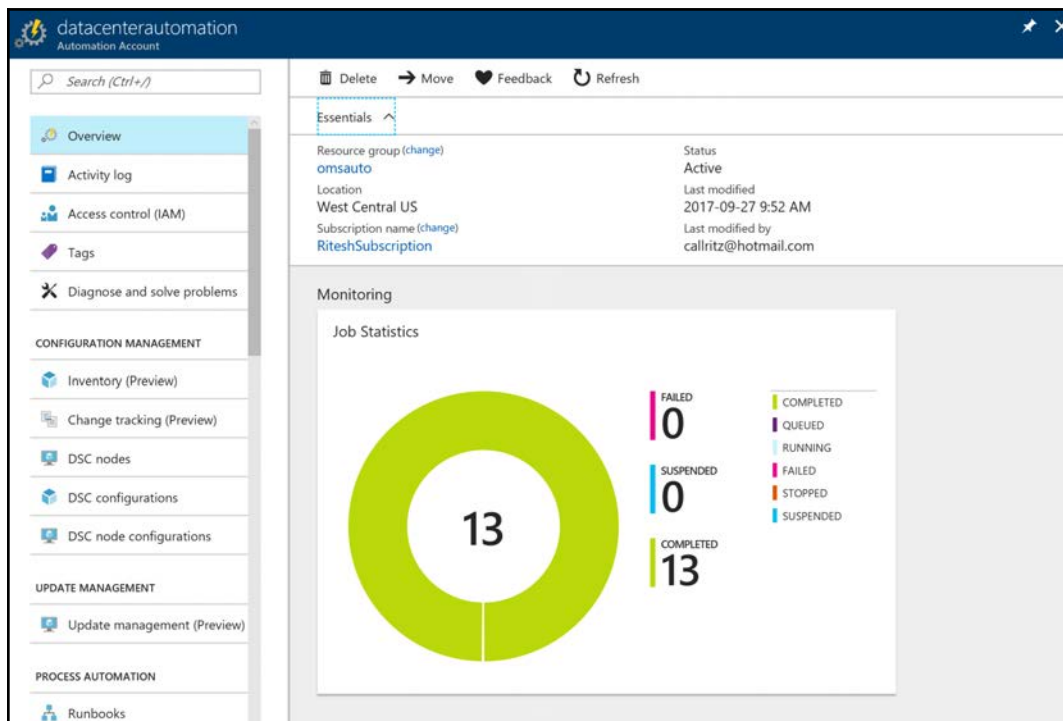
DSC provides a declarative language in which you define the intent and configuration but not how to run and apply those configurations. These configurations are based on PowerShell language and eases the process for configuration management.

In this section, we will look into a simple implementation of using Azure automation DSC to configure a virtual machine to install and configure web-server (IIS) and create an `index.htm` file that informs users that the website is under maintenance.

Provision Azure Automation Account

Create a new Azure automation account from Azure portal or PowerShell within an existing or a new resource group. Astute readers will find in the next image that Azure automation provides menu items for DSC. It provides the following:

- **DSC nodes:** It lists down all the virtual machines and containers that are enlisted with the current Azure automation DSC pull server. These virtual machines and containers are managed using configurations from the current DSC pull server in consideration.
- **DSC configurations:** It lists down all the raw PowerShell configurations imported and uploaded to the DSC pull server. They are in human readable format and are not in a compiled state.
- **DSC node configurations:** It lists down all compiles of DSC configurations available on the pull server to be assigned to nodes--virtual machines and containers. A DSC configuration produces MOF files after compilations and they are eventually used to configure nodes.



Author DSC Configuration

The next step is to write a DSC configuration using any PowerShell editor to reflect the intent of configuration. For this sample, a single configuration `ConfigureSiteOnIIS` is created. It imports the base DSC module `PSDesiredStateConfiguration` consisting of resources used within the configuration. It also declares a node `webserver`. When this configuration is uploaded and compiled, it will generate a `DSCConfigurationNodes` named `ConfigureSiteOnIISwebserver`. This configuration then can be applied to nodes.

The configuration consists of a few resources. These resources configure the target node. The resources install a web-server, ASP.NET and framework and creates an `index.htm` file within `inetpub\wwwroot` directory with content to show that the site is under maintenance. For more information about writing DSC configuration, refer to <https://docs.microsoft.com/en-us/PowerShell/dsc/configurations>.

```
Configuration ConfigureSiteOnIIS {
    Import-DscResource -ModuleName 'PSDesiredStateConfiguration'
    Node WebServer {
        WindowsFeature IIS
        {
            Name = "Web-Server"
            Ensure = "Present"
        }
        WindowsFeature AspDotNet
        {
            Name = "net-framework-45-Core"
            Ensure = "Present"
            DependsOn = "[WindowsFeature]IIS"
        }
        WindowsFeature AspNet45
        {
            Ensure = "Present"
            Name = "Web-Asp-Net45"
            DependsOn = "[WindowsFeature]AspDotNet"
        }
        File IndexFile
        {
            DestinationPath = "C:\inetpub\wwwroot\index.htm"
            Ensure = "Present"
            Type = "File"
            Force = $true
            Contents = "<HTML><HEAD><Title> Website under
construction.</Title></HEAD><BODY> `
                <h1>If you are seeing this page, it means the website is
under maintenance and DSC Rocks !!!!!</h1></BODY></HTML>"
        }
    }
}
```

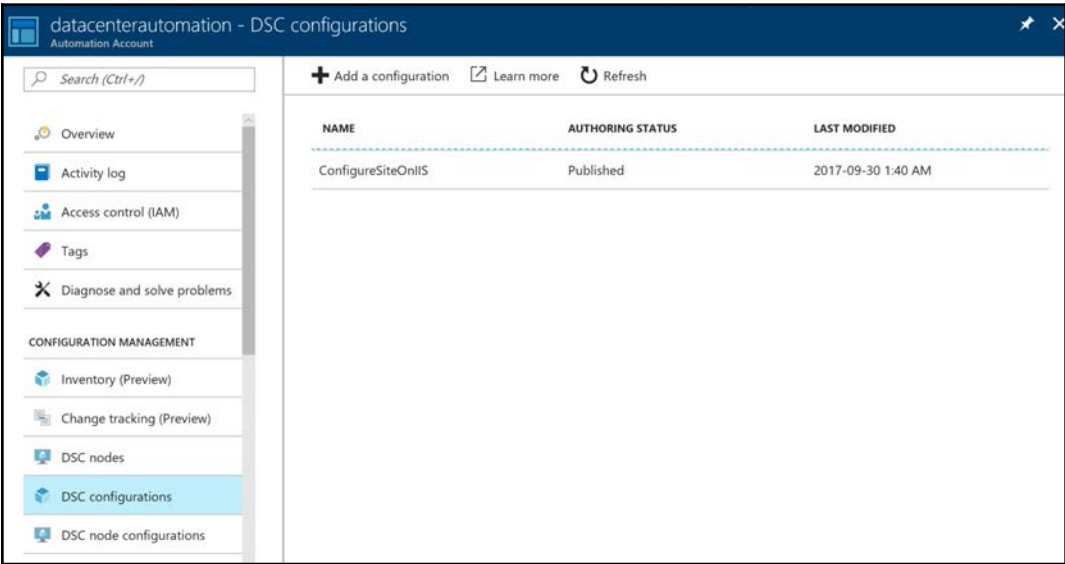
```
    }  
  }  
}
```

Importing DSC Configuration

The DSC configuration is still not known to Azure automation. It is available on some local machines. It should be uploaded to Azure Automation DSC Configurations. Azure automation provides `Import-AzureRmAutomationDscConfiguration` cmdlet to import the configuration to Azure automation.

```
Import-AzureRmAutomationDscConfiguration -SourcePath  
"C:\DSC\AA\DSCfiles\ConfigureSiteOnIIS.ps1" -ResourceGroupName  
"omsauto" -AutomationAccountName "datacenterautomation" -Published -  
Verbose
```

The DSC configuration on Azure after applying configuration to node should look as shown next:

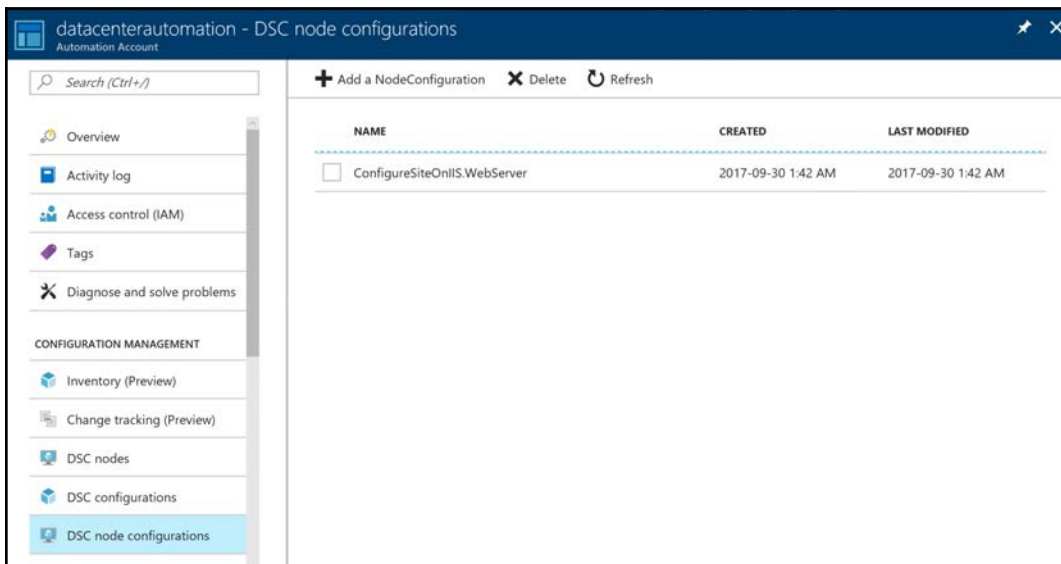


Compiling DSC Configuration

After the DSC configuration is available in Azure automation, it can be asked to be compiled. Azure automation provides another cmdlet for the same. Use `Start-AzureRmAutomationDscCompilationJob` cmdlet to compile the imported configuration. The configuration name should match exactly the name of the configuration uploaded. Compilation creates a MOF file named after the configuration and node name together, which in this case is `ConfigureSiteOnIIS.webserver`.

```
Start-AzureRmAutomationDscCompilationJob -ConfigurationName
ConfigureSiteOnIIS -ResourceGroupName "omsauto" -AutomationAccountName
"datacenterautomation" -Verbose
```

The DSC nodes configuration on Azure after applying the configuration to the node should look as shown next:



Assigning Configuration to Nodes

Now, the compiled DSC configurations can be applied to nodes. Use `Register-AzureRmAutomationDscNode` to assign the configuration to a node. The `NodeConfigurationName` parameter identifies the configuration name that should be applied to the node. This is a powerful cmdlet which can also configure the DSC agent that is `localconfigurationmanager` on nodes before they can download configurations and apply the same. There are multiple `localconfigurationmanager` parameters that can be configured and details about the same are available at

<https://docs.microsoft.com/en-us/PowerShell/dsc/metaconfig>.

```
Register-AzureRmAutomationDscNode -ResourceGroupName "omsauto" -  
AutomationAccountName "datacenterautomation" -AzureVMName testtwo -  
ConfigurationMode ApplyAndAutocorrect -ActionAfterReboot  
ContinueConfiguration -AllowModuleOverwrite $true -  
AzureVMResourceGroup testone -AzureVMLocation "West Central US" -  
NodeConfigurationName "ConfigureSiteOnIIS.WebServer" -Verbose
```

The DSC nodes on Azure after applying the configuration to the node should look as shown next:

datacenterautomation - DSC nodes

Automation Account

Search (Ctrl+F)

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems

CONFIGURATION MANAGEMENT

Inventory (Preview)
Change tracking (Preview)
DSC nodes
DSC configurations
DSC node configurations

+ Add Azure VM + Add Non-Azure Learn more Refresh Enable Log Search

Azure VMs managed by the configuration service do not incur a cost if they have the VM DSC extension version greater than 1.0.0 installed. Learn more about this price change and how to upgrade the extension.

DSC nodes: Search nodes... Status: 7 selected Node configuration name: 2 selected VM DSC extension ver: 2 selected

NAME	STATUS	NODE CONFIGURATION	LAST SEEN	VM DSC EXT
testtwo	✓ Compliant	ConfigureSiteOnIIS.WebS...	2017-09-30 2:34 AM	Yes

Browse the Server

If appropriate, Network security groups and firewalls are opened and enabled for port 80 and a public IP is assigned to the virtual machine, the default website can be browsed using the IP address. Otherwise, log in to the virtual machine that is used for applying the DSC configuration and navigate to `http://localhost`.

It should show a page as shown next:



This is the power of configuration management that without writing any significant code, authoring a configuration once can be applied multiple times to the same and multiple servers and be assured that they will be running in the desired state without any manual intervention.

Azure for DevOps

As mentioned before, Azure is a rich and mature platform providing the following:

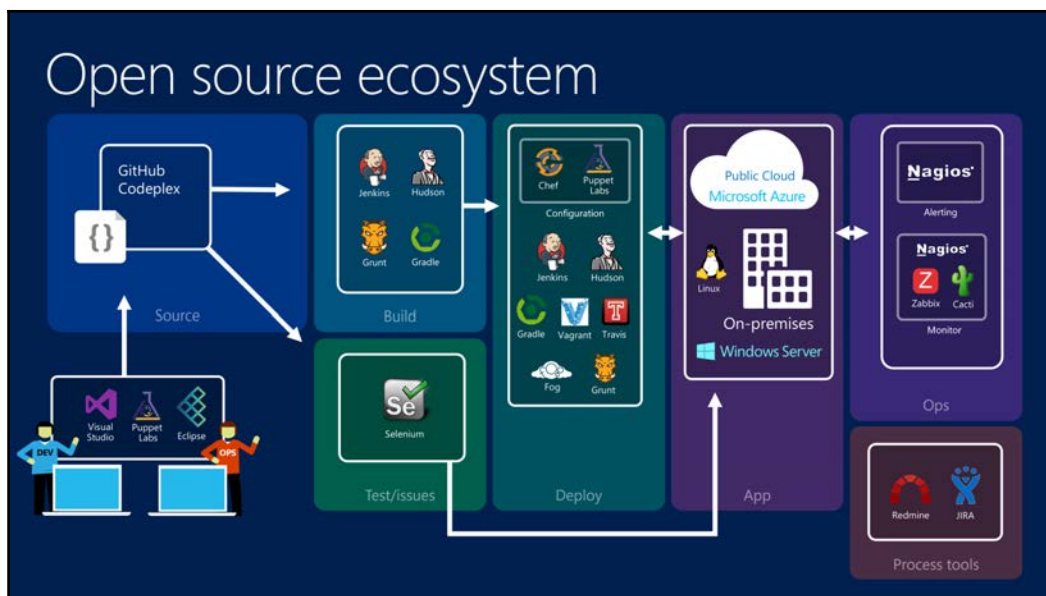
- Multiple choices of languages
- Multiple choices of operating system
- Multiple choices of tools and utilities
- Multiple patterns for deploying solutions (virtual machines, app services, containers, micro-services)

With so many options and choices, Azure is the following:

- **Open cloud:** It is open for both open source, Microsoft and non-Microsoft products, tools, and services
- **Flexible cloud:** It provides flexibility to both users and developers to feel comfort in using whatever their existing skillset and knowledge is based on
- **Unified management:** Provides seamless monitoring and management features

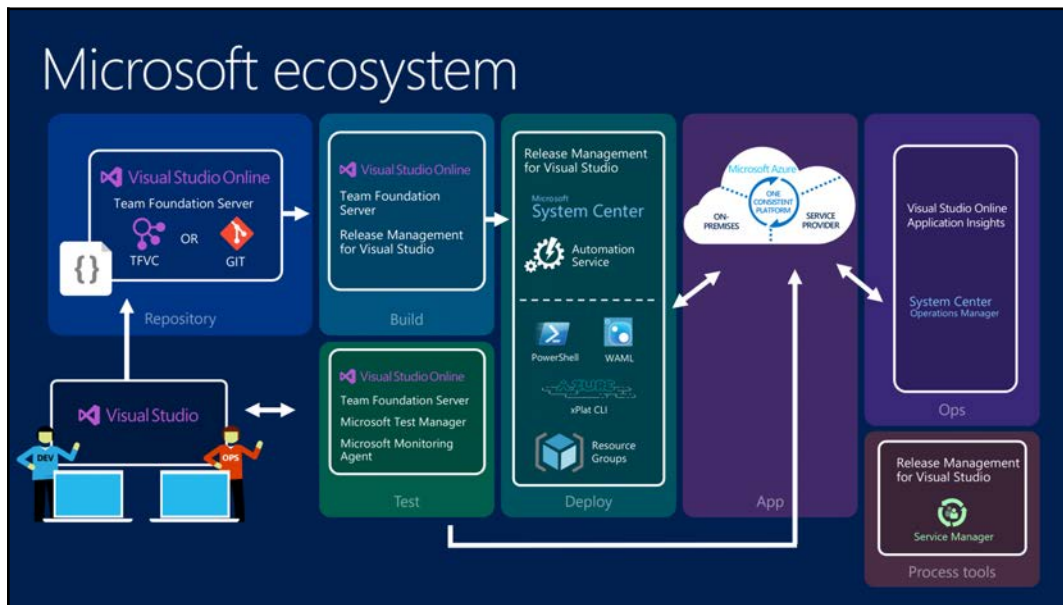
All the features mentioned before are important for successful implementation of DevOps for any project or engagement. The next image shows the open source tools and utilities that can be used for different phases in application life cycle management and overall DevOps. This is just a small representation of all tools and utilities and there are much more options available.

- Jenkins, Hudson, Grunt, and Gradle tools for constructing build pipeline
- Selenium for testing
- Chef, Puppet, Jenkins, Hudson, Grunt and Gradle and more for deployment or configuration management
- Nagios for alerting and Monitoring
- Jira and Redmine for managing processes



The next image shows Microsoft tools and utilities that can be used for different phases in the application life cycle management and overall DevOps. This is again just a small representation of all tools and utilities and there are much more options available.

- VSTS build orchestration for constructing a build pipeline
- Microsoft test manager, Pester for testing
- Desired State Configuration, PowerShell, ARM templates and more for deployment or configuration management
- Log analytics, application insights, and **System Center Operations Manager (SCOM)** for alerting and monitoring
- VSTS and **System Center Service Manager** for managing processes



Summary

DevOps is gaining lots of traction and momentum in the industry. Most organizations have realized its benefits and are looking forward to implementing DevOps. This is happening while most of them are moving to the cloud. Azure as a cloud model is rich and mature in providing DevOps services, making it easy for such organizations to implement DevOps. In this chapter, DevOps was discussed at length along with its core practices, such as configuration management, continuous integration, continuous delivery, and deployment. We also discussed different PaaS based, virtual machines IaaS, and a container IaaS based cloud solution along with their respective Azure resources, build and release pipelines. Configuration management was the core part of this chapter and we discussed DSC services from Azure automation and using pull servers to configure virtual machines automatically. Finally, the openness and flexibility of Azure regarding varied open source choice of languages, tools, and operating system was discussed.

Cost is one of the most important drivers to move to cloud and there are opportunities to save and manage cost even on cloud. Next chapter on cost management provides details about cost administration and minimization on Azure.

Other Books You May Enjoy

If you enjoyed this book, you may be interested in these other books by Packt:



Hands-On Azure for Developers

Kamil Mrzygłód

ISBN: 978-1-78934-062-4

- Implement serverless components such as Azure functions and logic apps
- Integrate applications with available storages and containers
- Understand messaging components, including Azure Event Hubs and Azure Queue Storage
- Gain an understanding of Application Insights and other proper monitoring solutions
- Store your data with services such as Azure SQL and Azure Data Lake Storage
- Develop fast and scalable cloud applications



Hands-On Machine Learning with Azure

Thomas K Abraham, Parashar Shah, Jen Stirrup, Lauri Lehman and Anindita Basak

ISBN: 978-1-78913-195-6

- Discover the benefits of leveraging the cloud for ML and AI
- Use Cognitive Services APIs to build intelligent bots
- Build a model using canned algorithms from Microsoft and deploy it as a web service
- Deploy virtual machines in AI development scenarios
- Apply R, Python, SQL Server, and Spark in Azure
- Build and deploy deep learning solutions with CNTK, MMLSpark, and TensorFlow
- Implement model retraining in IoT, Streaming, and Blockchain solutions
- Explore best practices for integrating ML and AI functions with ADLA and logic apps

Leave a Review - Let Other Readers Know What You Think

Please share your thoughts on this book with others by leaving a review on the site that you bought it from. If you purchased the book from Amazon, please leave us an honest review on this book's Amazon page. This is vital so that other potential readers can see and use your unbiased opinion to make purchasing decisions, we can understand what our customers think about our products, and our authors can see your feedback on the title that they have worked with Packt to create. It will only take a few minutes of your time, but is valuable to other potential customers, our authors, and Packt. Thank you!

Index

A

A-Series virtual machines

- about 253

- Azure A-series basic 253

- Azure A-series standard 253

Access Control List (ACL) 242

Access Control Lists (ACLs) 577

access keys 243, 244

accounts

- adding, to Azure Active Directory (AAD) 88, 89, 90, 95, 99, 102, 103, 109

- Cloud accounts 88

- Hybrid accounts 88

ACID properties 551

ACS Engine

- reference link 377

Active Directory (AD) 554

Active Directory Federation Services (ADFS) 110

ADFS Proxy 114

ADFS Server 114

Advanced Message Queuing Protocol (AMPQ) 531

affecting factors, High Availability

- application deployment architecture 486

- planned maintenance 486

- unplanned maintenance 486

analytics

- about 534

- diagnostic analytics 534

- predictive analytics 535

- prescriptive analytics 535

App service 15

app services, plans

- basic 513

- free 513

- premium 514

- shared 513

- standard 513

application 485

application gateways 499

Application High Availability 494, 505

Application Request Routing (ARR) 500

architectural considerations, High Availability

- High Availability across Azure regions 503

- High Availability within Azure regions 502

architectural considerations, storage accounts 471

architectural considerations, virtual network

- connectivity 462

- dedicated DNS 460

- deployment 462

- IP range for subnets 461

- IP ranges for networks 461

- monitoring 461

- network performance 460

- number of subnets in virtual network 461

- number of virtual networks 460

- region 460

- security considerations 461

ARM template

- authoring 63, 64

- creating 65, 66, 67, 68, 69, 70

- deployment, exporting 50, 51

- example 51, 53, 54, 55, 57

- modifying 57, 58, 59, 60, 61, 62, 63

- working 49, 50

auto scaling 512

automation account 17

availability sets 488

- about 265

- fault domain 265

- update domain 265

Availability Zones

- reference 489
- AzCopy tool
 - about 243
 - reference link 243
- Azure A-Series basic
 - versus Azure A-Series standard 253
- Azure A-Series standard
 - versus Azure A-Series basic 253
- Azure Active Directory (AAD)
 - about 76, 77, 104, 416
 - accounts, adding 88, 89, 90, 91, 102
 - Azure AD basic 78
 - Azure AD Connect environment, installing 118, 123, 129
 - Azure AD Connect prerequisites, installing 114
 - Azure AD Connect, infrastructure 130, 132
 - editions 416, 417
 - groups, adding 88, 89, 90, 91, 99, 102, 103, 109
 - identity protection 423
 - multi-factor authentication 426, 427, 428, 429, 430, 431, 432, 433, 434
 - options 77
 - potential risky accounts 423
 - potential suspicious events, investigation 423, 424, 425, 426
 - Privileged Identity Management (PIM) 418, 419, 420, 421, 422
 - vulnerabilities, detection 423
- Azure Active Directory premium P2 77
- Azure AD Connect
 - about 105
 - reference link 109
 - synchronization rule editor 105
 - synchronization service 106
 - synchronization service key manager 107
 - synchronization service web service configuration manager 108
- Azure Application Building Blocks 20
- Azure application gateways 162
- Azure automation
 - about 670
 - account, creating 671
 - configuration, assigning to nodes 675
 - DSC configuration, compiling 674
 - DSC configuration, importing 673
 - DSC configuration, writing 672
 - server, browsing 676
- Azure bindings
 - reference 594
- Azure CLI
 - reference link 18
- Azure Cloud Service
 - about 295, 296
 - architecture 296
 - Azure series, selecting 314
 - cloud services package 297
 - creating 322, 324, 325, 326, 328, 329, 331, 332, 335, 337, 340, 342, 343, 345, 346, 348, 350, 352, 353, 354, 356, 358, 360, 361, 363, 365, 367, 369, 370, 371, 373
 - exploring 299
 - Guest OS, selecting 312
 - roles 297
 - service configuration file 297, 308
 - service definition file 300
 - service endpoint 298
 - update level, selecting 312
 - versus Azure PaaS offerings 312
- Azure Command-Line Interface (CLI) 17
- Azure connections 155
- Azure Container Service cluster
 - working 399, 401, 403, 404, 406, 408, 410, 411, 413
- Azure Container Service
 - about 375, 378
 - cluster 378
 - creating 383, 384, 385, 387, 388, 391, 392, 394, 395, 396, 397, 398
 - orchestrator 379
- Azure Data Center 376
- Azure data factory
 - about 561, 562
 - data sources 563
 - data transformation 564
 - publish and presentation 564
 - reference 567
 - using 564, 566, 567, 568, 570, 571, 572, 573, 574, 575

- Azure Data Lake Analytics 538, 578, 579
- Azure Data Lake store 576
- Azure Data Lake
 - about 575
 - performance 578
 - security 577, 578
- Azure Data Services 20
- Azure Execution Model 20
- Azure ExpressRoute circuit
 - configuring 185, 187, 188, 189
- Azure ExpressRoute
 - about 147, 149, 158
 - Azure global Points of Presence 150
 - National Azure Points of Presence 152
 - Premium SLA 153
 - Standard SLA 153
- Azure for DevOps 676, 677, 678
- Azure function configuration
 - about 597
 - app services function settings 599
 - platform configuration 598
- Azure function cost plans
 - about 599
 - app service plan 600
 - consumption plan 599
- Azure function proxies 594
- Azure functions use cases
 - communication gateways 606
 - data processing 605
 - integration between multiple endpoints 605
 - legacy applications integration 605
 - microservices implementation 604
 - scheduled jobs 605
- Azure functions
 - about 591
 - advantages 600, 601, 602, 603, 604
 - authentication 596
 - authorization 596
 - binding 592
 - creating 606, 607, 608, 609, 610, 612
 - event-based functions 606
 - monitoring 595
 - on demand functions 606
 - reference 595
 - runtime 591
 - scheduled functions 606
 - triggers 593
 - types 606
- Azure governance features
 - background 636
 - implementing, examples 636
 - role-based access control 636
- Azure HDInsight 538
- Azure HDInsight Hadoop 538
- Azure High Availability
 - about 487
 - availability sets 488
 - Availability Zones 489
 - fault domain 488
 - load balancing 489
 - update domain 489
- Azure Identity Security
 - about 416
 - Azure Active Directory (AAD) 416
- Azure IoT
 - about 535
 - analytics 538
 - capture 536
 - DocumentDB 538
 - external data sources 538
 - identity 536
 - ingestion 537
 - presentation 539
 - storage 537
 - transform 538
- Azure Load Balancer (ALB) 301
- Azure load balancer
 - about 159
 - automatic reconfiguration 160
 - hash-based distribution 160
 - port forwarding 160
 - service monitoring 161
- Azure load balancers
 - about 489, 495
 - internal load balancer 497
 - port forwarding 499
 - public load balancer 496
- Azure local gateway 146
- Azure locks 630, 631, 632
- Azure logs

- about 451, 452, 453
- activity log 451
- diagnostic log 451, 452
- metrics 451
- Azure Machine Learning 538
- Azure network security
 - about 441
 - network security groups (NSGs) 441
 - network virtual appliances 443, 444
- Azure Network Watcher 461
- Azure networking
 - architecture 211, 214
 - Azure application gateways 162
 - Azure connections 155
 - Azure DNS 164
 - Azure ExpressRoute 147, 149
 - Azure load balancer 159
 - Azure local gateway 146
 - Azure routes 156
 - Azure third-party network devices 157, 158
 - Azure Traffic Manager 163
 - Azure virtual network gateway, setting up with Azure ExpressRoute 183
 - Azure virtual network gateway, setting up with MPLS 183
 - Azure virtual network site-to-site VPN, setting up 171
 - Azure virtual networks (VNet) 136, 137
 - Azure VNet peering 139
 - Azure VNet peering, setting up 192
 - Azure VNet, setting up 165, 167, 169, 171
 - Azure VPN gateways 139, 142, 144, 145
 - components 136
 - custom routes, configuring 199, 200, 202
 - limits 135
 - setting up 164
 - third-party network appliances, configuring 204, 206, 209
- Azure PaaS offerings
 - versus Azure Cloud Services 312
- Azure policies
 - about 627
 - allowed fields 630
 - built-in policies 627
 - language 628, 629
- Azure PoPs
 - reference link 152
- Azure Portal
 - about 17
 - URL 17
- Azure PowerShell
 - reference link 18
- Azure pricing calculator
 - reference link 249
- Azure RBAC
 - about 632, 633
 - custom roles 635
- Azure regions
 - about 16, 457
 - availability of resources 458
 - cost of running applications 459
 - data and privacy compliance 458
 - performance of applications 459
 - references 16
- Azure resource explorer 72, 73
- Azure resource group
 - creating 23, 24, 25, 26
 - resource, adding to 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40
- Azure Resource Manager 15
- Azure resource manager (ARM)
 - about 21
- Azure Resource Manager (ARM)
 - about 139
- Azure resource manager (ARM)
 - functionalities 21, 22
- Azure Resource Manager (ARM)
 - reference link 147
- Azure resource manager (ARM)
 - tagging 40, 41, 42, 43, 44, 45
 - working 22, 23
- Azure Resource Manager portal
 - virtual machine, accessing 285, 287
 - virtual machine, deploying 267, 269, 271, 274, 276, 279, 280, 283
- Azure Resource Manager Template Visualizer (ArmViz) 64
- Azure Resource Manager templates
 - about 646, 647
- Azure Resource Manager

- about 488, 577
- Azure resource
 - locking 46, 47, 48, 49
- Azure resources 15
- Azure routes 155
- Azure scalability
 - about 510
 - auto scaling 512
 - IaaS scalability 516
 - PaaS scalability 513
 - scaling 511
 - scaling down 511
 - scaling in 512
 - scaling out 512
 - scaling up 511
- Azure SDK
 - URL 18
- Azure security center 453, 454, 455, 456
- Azure series
 - nutshell 322
 - selecting 314
 - Series A 314
 - Series D 316
 - Series F 318
 - Series G 318
 - Series H 320
 - Series N 319
- Azure Service Definition Schema 300
- Azure Service Manager (ASM) 139, 165
- Azure SQL availability 553
- Azure SQL data warehouse
 - about 579
 - characteristics 581
 - compute nodes 580
 - control node 580
- Azure SQL databases 551
- Azure SQL logical server 551
- Azure SQL security 554
- Azure SQL
 - about 550, 551
 - elastic pools 555
 - performance, levels 552
 - scaling out 556, 557
- Azure Storage Account 291
- Azure Storage Explorer
 - Azure Storage services, exploring 245
- Azure storage security
 - about 444
 - key management 444, 445, 446
 - shared access signatures 446
 - storage encryption 447
- Azure Storage services
 - about 223
 - access keys 243, 244
 - Blob storage services 224, 226, 228
 - exploring, with Azure Storage Explorer 245
 - File storage services 234, 237, 238, 241, 242
 - pricing 249
 - Queue storage services 231, 233
 - Table storage services 229, 230
- Azure storage
 - Geo-redundant storage (GRS) 470
 - Locally redundant storage (LRS) 470
 - Read-access geo-redundant storage (RAGRS) 470
 - Zone-redundant storage (ZRS) 470
- Azure stream analytics 538
- Azure stream analytics service
 - architecture 560, 561
 - data integration 559
 - data sources 559
 - data transformation 559
 - presentation 560
 - storage 560
 - stream analytics engine 559
- Azure tags
 - about 623, 624
 - resource groups, versus resources 626
 - with ARM template 625
 - with PowerShell 625
- Azure third-party network devices 157, 158
- Azure Traffic Manager 163
 - about 500
 - traffic routing methods 501
- Azure triggers
 - reference 594
- Azure virtual machine types
 - A-Series virtual machines 253
 - about 252

- Basic A-Series 252
- Compute intense A-Series 252
- D and DS-Series 252
- D-Series virtual machines 255
- DS-Series virtual machines 255
- F and FS-Series 252
- F-Series virtual machines 257
- FS-Series virtual machines 257
- G and GS-Series 252
- G-Series virtual machines 258
- GS-Series virtual machines 258
- H-Series 252
- LS-Series 252
- LS-Series virtual machines 261
- N-Series 252
- NC-Series virtual machines 260
- NV-Series virtual machines 260
- Standard A-Series 252
- Azure virtual network gateway
 - configuring 184
 - connection, configuring between Azure ExpressRoute circuit and Azure virtual network gateway 190, 192
 - setting up, with Azure ExpressRoute 183
 - setting up, with MPLS 183
- Azure virtual network site-to-site VPN
 - Azure virtual network gateway, configuring 173, 174, 177
 - connection, configuring between local and virtual network gateways 179, 181, 183
 - local network gateway, configuring 171
 - setting up 171
- Azure virtual networks (VNet) 137
 - about 136
 - setting up 165, 167, 169, 171
- Azure VNet peering 139
 - configuring 195, 196, 198
 - deployment, preparing 193, 194
 - setting up 192
- Azure VPN gateways 142, 144, 145
 - about 139
 - URL, for guide 139
- Azure zones 457

B

- Backend network 137
- benefits, virtual network
 - extensibility 463
 - isolation 463
 - security 463
- blob storage 469
- BLOB storage 14
- Blob storage account
 - cool access tier 219
 - hot access tier 219
 - premium storage performance tier 218
 - standard storage performance tier 218
- Blob storage services 224, 226, 228
- Border Gateway Protocol (BGP) 147
- business intelligence (BI) 538

C

- capture 532, 536
- circuit breaker pattern
 - about 482
 - closed state 482
 - half-open state 482
 - open state 482
 - reference 483
- Client-side Encryption (CSE) 469
- cloud characteristics
 - about 11
 - broad network access 11
 - measured service 11
 - on-demand self-service 11
 - rapid elasticity 11
 - resource pooling 11
- cloud deployment models
 - about 9
 - community cloud 9
 - hybrid cloud 10
 - private cloud 10
 - public cloud 9
- cloud service models
 - about 8
 - Infrastructure as a Service (IaaS) 8
 - Platform as a Service (PaaS) 8
 - Software as a Service (SaaS) 8

- Cloud Service Package 297
- Code Assets 297
- Command and Query Responsibility Segregation (CQRS) pattern 477
 - about 478
 - reference 478
 - throttling pattern 479
- Common Internet File System (CIFS) 235
- Compute Unified Device Architecture (CUDA) 319
- concurrency features, Azure storage account
 - last writer wins 472
 - optimistic concurrency 472
 - pessimistic concurrency 472
- configuration management, DevOps
 - about 644, 645
 - Ansible 645
 - Azure Resource Manager templates 646
 - Chef 645
 - Desired State Configuration (DSC) 645
 - Puppet 645
- connected architecture
 - creating, with functions 616, 619, 621
- connectivity 530, 531, 532
- containers
 - advantages 589
 - disadvantages 589
- Content Distribution Network (CDN) 526
- continuous delivery, DevOps 651
- continuous deployment, DevOps
 - about 649
 - acceptance tests 651
 - deployment to production 651
 - staging environment deployment 651
 - test automation 650
 - test environment deployment 650
- continuous integration, DevOps
 - about 646, 647
 - build automation 648
 - packaging 648
 - test automation 648
- continuous learning, DevOps 652
- Cosmos DB 616
- Cross-Origin Resource Sharing (CORS) 602
- custom role creation

- reference link 439
- custom routes
 - configuring 199, 200, 202

D

- D-Series virtual machines 255
- data factory 538
- Data Movement Service (DMS) 580
- Data Warehouse Unit (DWU) 582
- Database Transaction Units (DTU) 552
- Datacenter Operating System (DC/OS) 379
- design patterns
 - about 472
 - circuit breaker pattern 482
 - Command and Query Responsibility Segregation (CQRS) pattern 477, 478
 - messaging patterns 473
 - performance and scalability patterns 477
 - retry pattern 480, 481
- Desired State Configuration (DSC) 645
- DevOps for PaaS solutions
 - about 660, 661
 - Azure app services 662
 - Azure SQL 662
 - build pipeline 663
 - deployment slots 662
 - release pipeline 663
- DevOps, for container based (IaaS) solutions
 - about 667
 - build pipeline 668
 - containers 667
 - Docker 668
 - DockerFile 668
 - release pipeline 669
- DevOps, for virtual machine (IaaS) based solutions
 - about 663
 - Azure public load balancer 664
 - Azure virtual machine 664
 - build pipeline 665
 - release pipeline 666
- DevOps
 - about 639, 640
 - configuration management 643, 644
 - continuous delivery 651

- continuous deployment 649
- continuous integration 646, 647
- continuous learning 652
- core principles 640
- practices 641, 643
- preparing for 656
- disaster recovery
 - versus High Availability 487
- disk encryption, implementing
 - reference link 450
- distance-vector routing protocol 147
- DNS CNAME 500
- DNS settings
 - changing 288
- Docker Engine 377
- Docker swarm
 - reference link 381
- DS-Series virtual machines 255
- dynamic data masking 554

E

- eDTU 555
- Enterprise Mobility Suite (EMS) 415
- event-based functions 606
- event-driven function
 - creating 613, 614, 615
- ExpressRoute 459, 466
- Extract-and-Load (EL) 538
- Extract-Transform-Load (ETL) 558

F

- F-Series virtual machines 257
- fault domain 488
- file storage 14, 469
- File storage services 234, 237, 238, 241, 242
 - file access 242
 - storage limit 242
- first in first out (FIFO) 232
- forced tunneling
 - reference 462
- Frontend network 137
- FS-Series virtual machines 257
- functions as a service (FaaS) 591
- functions
 - connected architecture, creating with 616,

619, 621

G

- G-Series virtual machines 258
- gateway subnet 136
- geo-redundant storage (GRS) 221
- Geo-redundant storage (GRS) 470
- GIT 655
- Group Policy Objects (GPOs) 416
- groups
 - adding, to Azure Active Directory 109
 - adding, to Azure Active Directory (AAD) 88, 89, 90, 91, 93, 95, 103
- GS-Series virtual machines 258
- Guest OS
 - selecting 312

H

- H-Series virtual machines 259
- high availability 548
- High Availability
 - about 485
 - affecting factors 486
 - Application High Availability 494
 - architectural considerations 501
 - best practices 504, 505
 - computing 490, 491, 492
 - data related best practices 506
 - deployment related best practices 505
 - monitoring related best practices 506
 - PaaS High Availability 493
 - Storage High Availability 492
 - versus disaster recovery 487
 - versus scalability 487
 - virtual machine High Availability 490
- high performance computing (HPC) 321
- hybrid networks 467

I

- IaaS scalability 516
- identity 532, 536
- Identity Protection 79
- Information Technology (IT) 7
- Infrastructure services, Microsoft Azure
 - networking 14

- operating system 14
- server compute 14
- storage 14
- ingestion 533, 537
- internal load balancing 497
- Internet information service (IIS) 500
- Internet of Things (IoT) 14, 528, 529
- Internet Service Provider (ISP) 147
- IoT architecture
 - about 529, 530
 - analytics 534
 - capture 532
 - connectivity 530, 531, 532
 - identity 532
 - ingestion 533
 - presentation 535
 - storage 533
 - transform 533
- IoT hubs
 - about 539, 540
 - device registration 540, 541
 - message management 542
 - protocols 540
 - reference link 540
 - scalability 545
 - security 544
- IP settings
 - changing 288

J

- JavaScript Object Notation (JSON) 15, 646
- just-in-time (JIT) 422

K

- key management 444, 445, 446
- Kubernetes
 - reference link 382
- Kudu console 591

L

- lift-and-shift migrations 235
- load balancer probe 301
- load balancing 489, 494
- LoadBalancerProbe element 301
- Locally redundant storage (LRS) 470

- locally redundant storage (LRS) 220
- logging
 - about 450, 451
 - Azure logs 451, 452, 453
- LS-Series virtual machines 261

M

- machine learning 538
- managed disks 263
- Massively Parallel Processing (MPP) 581
- mesosphere DC/OS
 - reference link 379
- message management
 - about 542
 - cloud, to device messaging 543, 544
 - device, to cloud messaging 542, 543
- Message Queue Telemetry Transport (MQTT) 531
- messaging patterns
 - about 473
 - architectural concerns 473
 - competing consumers 474
 - priority queue 475
 - queue-based load leveling pattern 475, 476
- Microsoft Account Entity Management (MIM) 108
- Microsoft Azure
 - about 12
 - automation 17
 - automation tools 18
 - Azure Portal 17
 - Azure regions 16
 - key concepts 15
 - Platform Services 13
 - Resource Manager 15
 - resources 15
 - REST APIs 18
 - services, overview 12
 - URL, for free trial 14
- monitor 451
- monitoring
 - about 450, 451, 461
 - Azure security center 453, 454, 455, 456
- Multiprotocol Label Switching (MPLS) 10, 147

N

National Institute of Standards and Technology (NIST) 11

NC-Series virtual machines 260

Network Address Translation (NAT) 664

network security groups (NSGs)

about 441

destination 441, 442, 443

source 441

network security groups (NSGs)

source 441

network virtual appliances 443, 444

NetworkConfiguration element 310

networking limits

reference 460

NetworkTrafficRules element 307

NV-Series virtual machines 260

O

on demand functions 606

Online Analytics Processing (OLAP) 579

Online Transaction Processing (OLTP) 579

Open Computing Language (OpenCL) 319

Operation Management Suite (OMS) 415

operational management suite 458

optical character recognition (OCR) 605

orchestrator, Azure Container Service

Docker swarm 381

kubernetes 382

mesosphere DC/OS 379

Organizational Units (OUs) 416

P

PaaS High Availability 493

PaaS scalability 513

scaling down 514

scaling in 515

scaling out 515

scaling up 514

PaaS

disadvantages 588

path vector protocol 147

peering types

microsoft peering 149

private peering 148

public peering 148

performance and scalability patterns 477

performance

versus scalability 510

physical servers

disadvantages 587

Plain old simple object (POCO) 594

Platform Services, Microsoft Azure

analytics 14

CDN 13

data 13

developer services 13

hybrid operations 14

integration 13

Internet of Things (IoT) 14

management 13

media 13

mobile 13

security 13

service compute 13

web 13

Point of Presence (PoP) 149

PolyBase 580

port forwarding 499

premium storage 468

Premium Storage account

about 249

requisites 249

presentation 535, 539

principles, serverless technology

event driven 590

execute quickly 590

lower cost 590

single responsibility 590

priority queue pattern

about 475

reference 475

private key 388

Privileged Identity Management (PIM) 79, 418, 419, 420, 421, 422

probes

guest agent probe 161

HTTP custom probe 161

TCP custom probe 161

- protocols 540
- public key 388
- public load balancing 496, 497

Q

- queue storage 469
- Queue storage services 231, 233
- queue-based load leveling pattern 476

R

- rapid elasticity 11
- Raspberry Pi online simulator
 - reference link 540
- read-access geo-redundant storage (RA-GRS) 221
- Read-access geo-redundant storage (RAGRS) 470
- redundancy 488
 - about 220
 - geo-redundant storage (GRS) 221
 - locally redundant storage (LRS) 220
 - read-access geo-redundant storage (RA-GRS) 221
 - zone-redundant storage (ZRS) 220
- release pipeline 649
- replication 220
- resource manager policies
 - reference link 440
- resource security
 - about 434, 435
 - resource policies 440
 - resource tags 440
 - Role-Based Access Control (RBAC) 435, 436, 437, 438, 439
- resources, for enabling load balancing
 - application gateways 490
- retry pattern
 - about 480
 - reference 481
 - strategies 480
- Role-Based Access Control (RBAC) 22, 435, 436, 437, 438, 439, 577
- roles
 - about 297
 - WebRoles 297

- WorkerRoles 297

S

- scalability
 - about 509, 545
 - reference 509
 - SKU edition 545
 - units 547
 - versus High Availability 487
 - versus performance 510
- scaling down 511
- scaling in 512
- scaling out 512
- scaling up 511
- scaling
 - about 511
 - best practices 524, 525, 526
- scheduled functions 606
- Secure Access Signature (SAS) 469
- security
 - about 544
 - in IOT 544
- Server Message Block (SMB) 234
- serverless technology
 - principles 590
- serverless
 - about 589
 - history 587, 588
- service 485
- service configuration file
 - about 297, 309
 - NetworkConfiguration element 310
 - role element 309
- service definition file
 - about 297, 300
 - LoadBalancerProbe element 301
 - NetworkTrafficRules element 307
 - WebRole element 301, 303
 - WorkerRole element 305
- service endpoint 298
- service level agreement (SLA) 475, 485, 552, 590, 662
- Service Level Agreement (SLA) 220
- sharding 556
- shared access signatures (SAS) 243

- shared access signatures
 - about 446
 - account-level SAS 447
 - service-level SAS 446
- short messaging service (SMS) 606
- Single Sign-On (SSO) 78
- Sku edition
 - about 545
 - free 545
 - Standard (S1) 545
 - Standard (S2) 545
 - Standard (S3) 546
- source NAT (SNAT) 161
- Spark 538
- SQL Server Analysis Services (SSAS) 579
- standard storage 468
- stock keeping units (SKU) 513
- storage account
 - about 216
 - architectural considerations 471
 - BLOB storage account 218
 - concurrency features 472
 - general purpose storage account 218
- storage encryption
 - about 447
 - Azure disk encryption 449, 450
 - storage account encryption 447, 448, 449
- Storage High Availability 492
- Storage Service Encryption (SSE) 447, 469
- storage services
 - about 468
 - blob storage 469
 - file storage 469
 - queue storage 469
 - table storage 469
- storage types
 - BLOB storage 14
 - file storage 14
- storage
 - about 468, 537
 - Azure storage 537
 - CosmosDB 538
 - premium storage 468
 - SQL database 537
 - standard storage 468

- stream analytics
 - about 557
 - architecture 558
 - use cases 558
- System Center Operations Manager (SCOM) 678
- System Center Service Manager 678

T

- table storage 469, 583, 584
- Table storage services 229, 230
- tables, in Azure
 - reference 584
- tagging
 - benefits 623
- Team Foundation Services (TFS) 653
- Team Foundation Version Control (TFVC) 655
- third-party network appliances
 - configuring 206, 209
- throttling pattern
 - reference 479
- Traffic Manager 163
- traffic routing methods, Traffic Manager
 - performance 501
 - priority 501
 - weighted 501
- Transact-SQL 551
- transform 533
- Transform-and-Load (TL) 538
- transformation 562
- Transparent Data Encryption (TDE) 554

U

- update domain 489
- update level
 - selecting 312
- upgrades, Virtual machine scale sets (VMSS)
 - about 522
 - application updates 523
 - guest updates 524
 - image updates 524
- user-defined routes
 - reference link 444
- User-defined routing (UDR)
 - about 462

reference 462

V

Virtual Hard Disks (VHD) 449

virtual IP (VIP) 161

virtual machine extensions 262

virtual machine High Availability 490

Virtual machine scale sets (VMSS)

about 517

architecture 518

auto scaling 519, 520, 522

capacity property 519

horizontal scaling, versus vertical scaling 519

scaling 519

upgrades 522

virtual machine

accessing, in Azure Resource Manager portal
285, 287

common scenarios 289

communication traffic, optimization 289

deploying, in Azure Resource Manager portal
267, 269, 271, 274, 276, 279, 280, 283

on-demand usage, for calculations 290, 292

on-premises servers, disaster recovery 292,
294

virtual machines

advantages 587

disadvantages 588

Virtual Network (VNet) 167

virtual network, connecting to on-premise data
centers

point to site VPN 466

site to site VPN 465

virtual network

about 459

architectural considerations 460, 461, 462,
463

benefits 463

design 463, 464, 465

virtualization

disadvantages 587

Visual Studio Dev Essentials

about 14

URL 14

visual studio online (VSO) 653

Visual Studio Team Services (VSTS)

about 653

GIT 655

Team Foundation Version Control (TFVC)
655

VPN device

reference link 142

VPN Gateway 141

W

WebRole element 298, 301, 303

Wide Area Network (WAN) 465

Windows Azure Pack (WAP) 10

Windows Management Framework (WMF) 645

Windows Remote Management (WinRM) 115

WorkerRole element 298, 305

Z

Zone-redundant storage (ZRS) 470

zone-redundant storage (ZRS) 220