

Implementing Azure Solutions

Second Edition

Deploy and manage Azure containers and build Azure solutions
with ease



Packt>

www.packt.com

Florian Klaffenbach, Markus Klein, Sebastian Hoppe,
Oliver Michalski and Jan-Henrik Damaschke

Implementing Azure Solutions

Second Edition

Deploy and manage Azure containers and build Azure solutions with ease

Florian Klaffenbach
Markus Klein
Sebastian Hoppe
Oliver Michalski
Jan-Henrik Damaschke



BIRMINGHAM - MUMBAI

Implementing Azure Solutions

Second Edition

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.

Commissioning Editor: Gebin George
Acquisition Editor: Prachi Bisht
Content Development Editor: Abhishek Jadhav
Technical Editor: Swathy Mohan
Copy Editor: Safis Editing
Project Coordinator: Jagdish Prabhu
Proofreader: Safis Editing
Indexer: Pratik Shiroadkar
Graphics: Tom Scaria
Production Coordinator: Shantanu Zagade

First published: May 2017
Second edition: October 2018

Production reference: 1311018

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

ISBN 978-1-78934-304-5

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 customercare@packtpub.com for more details.

At www.packt.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 is currently working as Technology Solutions Professional at Microsoft. He is one of the well-known experts when it comes to hybrid cloud scenarios, cloud connectivity, and cloud environment optimization. Before he started at Microsoft, he worked at several companies in different roles, like as technical Community Manager and Solution Expert at Dell or Solutions Architect at CGI Germany. He is also one of Packt's authors and worked on books like *Implementing Azure Solutions* first and second edition and multi-cloud for architect. He spends his free time with his wife and little son and is currently waiting for his second son.

Markus Klein is working as Technology Solution Specialist at Microsoft Germany, specialized on Azure and Hybrid Azure scenarios. He is passionate about Microsoft technology for more than 20 years, starting with System Center, Service Provider Foundation, KATAL, Azure Pack, Azure, and later Azure Stack. Before joining Microsoft he has been working as Architect at Microsoft Cloud Partners. In 2007 he founded a cloud community and was the co-founder of some Azure meetups, too. He has been recognized as an MVP in Cloud and datacenter management for seven years before he joined Microsoft. You can find him as a speaker at conferences in Europe and abroad. He likes to support the community and is a regular blogger.

Sebastian Hoppe's passion for IT topics started almost 20 years ago with Linux derivatives and Linux based server hosting. About 12 years ago his path led him to software development topics and he co-founded a software development company. Since then he can look back on a broad range of projects and topics and is still passionate about software development topics such as web-development and DevOps. Today he is working as a Lead IT Consultant and Architect for Azure and Office 365 and is involved in many projects in different business sectors. He is also a Microsoft Certified Trainer and likes to share his knowledge in training, workshops and as a speaker on events. You can follow him on twitter @_derhoppe.

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 and July 2017, he has been a Microsoft Most Valuable Professional for Microsoft Azure. Oliver is author (co-author) of *Implementing Azure Solutions* and *Implementing Azure Cloud Design Patterns*, both available from Packt Publishing.

About the reviewer

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 Azure Stack. 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. He was also the author on the first edition of this book *Implementing Azure Solutions*.

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
Technical requirements	8
Service models	8
Deployment models	9
Cloud characteristics	11
Multi-cloud characteristics and models	11
Cloud brokering	14
Best of breed	14
Microsoft Azure	14
Azure services overview	15
Azure basics	17
Azure Resource Manager (ARM)	17
Resources	17
Azure regions	18
Microsoft data center and backbone	19
Azure portal	23
Azure automation	24
Azure automation tools	24
REST APIs	24
Summary	25
Questions	25
Further reading	26
Chapter 2: Azure Resource Manager and Tools	27
Technical requirements	28
Understanding ARM	28
Functionalities provided by ARM	34
Working with ARM	35
Creating an Azure resource group	36
Adding a resource to a resource group	38
First approach – adding a storage account to your resource group	39
Second approach – adding a storage account to your resource group	43
Tagging in ARM	45
Locking Azure resources	49
Azure resource locks	49
Working with ARM templates	53
Exporting a deployment as an ARM template (for IT pros)	54
Example 1 – exporting a resource group to an ARM template	54

Example 2 – exporting a resource (classic) to an ARM template	60
Modifying an ARM template	62
Authoring an ARM template	69
Creating your own ARM template (for developers)	70
Summary	77
Questions	77
Further reading	78
Chapter 3: Deploying and Synchronizing Azure Active Directory	79
Azure AD	80
Azure AD options	82
Azure AD free	82
Azure AD basic	82
Azure AD premium P1	82
Deploying a custom Azure AD	83
Adding accounts and groups to Azure AD	86
Installing Azure AD Connect – prerequisites	106
Installing a basic Azure AD Connect environment	110
Azure AD Connect highly available infrastructure	126
Azure AD conditional access	128
Azure AD DS	130
Summary	130
Questions	130
Further reading	131
Chapter 4: Azure Managed Applications	132
Technical requirements	132
Azure managed applications overview	133
History and purpose of Azure managed applications	133
Scenarios of Azure managed application publishing	133
Azure managed applications architecture	135
Elements of an Azure managed application	135
Security and access of an Azure managed application	136
Creating a managed application definition	137
The file mainTemplate.json	138
The file createUiDefinition.json	145
The UI definition outline	146
The basics step	147
Additional steps in the steps section	147
The outputs section	154
Testing and validation of our template files	154
Testing and validating the mainTemplate.json file	155
Validation and testing against Azure	155
Validation and testing with local scripts	156
Testing and validating the createUiDefinition.json file	158
Validation and testing against Azure	158

Validation and testing with local scripts	160
Publishing a managed application to the internal service catalog	161
Creating the service catalog managed application definition	161
The basics section	163
The package section	164
The authentication and lock level section	164
Deploying a managed application from the internal service catalog	168
The deployment process	169
The managed resource group	176
Publishing a managed application to the Azure marketplace	178
Technical and business requirements	179
Summary	182
Questions	182
Further reading	183
Chapter 5: Implementing Azure Networks	184
Azure networking limits	185
Azure networking components	185
Azure virtual networks (VNet)	185
VNet peering and global VNet peering	188
VNet service endpoints	188
Azure VPN gateways	189
Azure local gateway	196
Azure virtual WAN	197
Azure ExpressRoute	198
Route filter	204
ExpressRoute Direct	204
ExpressRoute Global Reach	204
Azure connections	206
Azure route	206
Azure Firewall	208
Azure third-party network devices	208
Azure load balancer	211
Hash-based distribution	212
Port forwarding	212
Automatic reconfiguration	212
Service monitoring	213
Source NAT	213
Azure Application Gateways and Web Application Firewall	214
Web Application Firewall	215
Azure Traffic Manager	216
Azure DNS	216
Azure DDoS	217
Setting up Azure networks	217
Setting up Azure VNet	218
Setting up Azure virtual network site-to-site VPN	225
Configuring local network gateway	226

Configuring Azure virtual network gateway	228
Configuring connection between local and virtual network gateways	234
Setting up Azure virtual network with MPLS and ExpressRoute	238
Configuring Azure virtual network gateway	239
Configuring Azure ExpressRoute circuit	240
Setting up Azure VNet peering	245
Preparing the deployment	246
Configuring VNet peering	248
Configuring custom routes	252
Common Azure network architectures	257
Summary	260
Questions	261
Further reading	261
Chapter 6: Implementing Azure Storage	262
Storage accounts	263
The Blob storage account	263
General-purpose storage v1 account	265
General-purpose storage v2 accounts	265
Azure File Sync/Storage Sync services	265
Azure Data Lake	266
Replication and redundancy	266
Locally redundant storage (LRS)	266
Zone-redundant storage (ZRS)	266
Geo-redundant storage (GRS)	267
Read-access geo-redundant storage (RA-GRS)	267
Azure Storage services	269
Blob storage services	270
Table storage services	271
Queue storage services	272
File storage services	272
Access keys	274
Exploring Azure Storage with Azure Storage Explorer	276
Premium storage accounts	280
Premium storage requirements	280
Pricing	280
How to deploy a storage account?	281
Summary	290
Questions	291
Further reading	291
Chapter 7: Virtual Machines in Azure	292
Azure VM types	292
A-series VMs	294
D-series and DS-series VMs	295
F-series and FS-series VMs	295

G-series and GS-series VMs	296
H-series VMs	296
NV-series and NC-series VMs	297
NV VMs	297
NC VMs	297
Ls-series VMs	297
B-series VMs	298
E-series VMs	298
M-series VMs	298
SAP HANA Large Instances	298
VM extensions	299
Managed disks	300
Availability sets	302
Availability Zone (AZ)	303
Azure Hybrid Use Benefit (HUB)	304
Azure Reserved Instances (RIs)	305
Accelerated networking for VMs	305
VM serial console	306
Azure Confidential Compute	306
Deploying a VM in Azure	307
Accessing a VM in Azure	318
Changing IP and DNS settings	322
Common scenarios for VMs	324
Optimization of Azure-related communication traffic	324
On-demand usage for calculations	326
Disaster recovery for on-premises servers	328
Summary	329
Questions	330
Further reading	330
Chapter 8: Implementing Azure-Managed Kubernetes and Azure	
Container Service	331
Technical requirements	332
Containers – the concept and basics	332
Microservices – the concept	334
Workloads to run in containers	335
Deploying container hosts in Azure	336
Docker on Linux	336
Windows Server Container VM	337
Azure Container Registry (ACR)	338
ACI	340
Creating a first container in Azure	340
Azure Marketplace containers	344
Creating custom containers	351
Container orchestration	352

The concept of container orchestration	352
Azure Kubernetes Service (AKS)	353
Summary	369
Questions	370
Further reading	370
Chapter 9: Implementing Azure Cloud Services	371
Technical requirements	371
What is an Azure Cloud Service?	372
Understanding the Cloud Service architecture	372
Roles	373
The service endpoint	374
Going deeper into the Cloud Services	376
Service definition file	376
LoadBalancerProbes	377
WebRole	377
WorkerRole	381
NetworkTrafficRules	383
Service configuration file	384
Role	384
NetworkConfiguration	385
Azure Cloud Services versus other Azure PaaS offerings such as	
Azure App Services	387
Selection of a Guest OS and an update level	387
Selection of an Azure series	388
series A	389
series D	390
series E	392
series G	392
series H	393
In a nutshell	393
Creating your first Azure cloud service	393
Part 1	394
Part 2	405
Summary	441
Questions	442
Further reading	442
Chapter 10: Implementing Azure Governance	443
Technical requirements	444
Organizational governance	444
Azure hierarchy	444
Naming standards	447
Technical governance	447
Resource groups	448
Resource tags	451

Resource locks	452
Resource auditing	452
Management groups	453
Azure Policies	457
RBAC	467
Azure tooling	473
Azure Security Center	473
Azure Cost Management	477
Summary	479
Questions	479
Further reading	480
Chapter 11: Azure Hybrid Data Center Services	481
Technical requirements	484
ASDK	484
Preparing the ASDK host	485
Identity management configuration	492
Networking configuration	497
VM design of Azure Stack (ASDK)	502
Azure Stack configuration task	503
Operating Azure Stack	504
Working with the portals	504
Working with PowerShell	506
Working with the CLI	508
Hybrid cloud patterns	509
Configure hybrid cloud connectivity	510
Machine learning solution with Azure Stack	511
Azure stack staged data analysis	511
Azure Stack cloud burst scenario	511
Azure Stack geo-distributed Application	512
Monitoring Azure Stack	512
Summary	514
Questions	515
Further reading	515
Assessments	516
Other Books You May Enjoy	524
Index	527

Preface

Microsoft Azure has numerous effective solutions that shape the future of any business. However, the major challenge that architects and administrators face are implementing these solutions appropriately.

Implementing Azure Solutions, Second Edition will enable you to implement Azure Solutions effectively. The book starts by helping you choose the back-end structure for your solutions. You will then work with Azure toolkit and learn how to use Azure Managed Apps to share your solutions with Azure Service Catalog. The focus will then be on various implementing techniques and best practices such as *Implementing Azure Cloud Services* by configuring, deploying and managing cloud services. Moving on, the book will teach how to work with Azure Managed Kubernetes and Azure Container Services.

By the end of the book, you will get an overview of how enterprise based IoT and Hybrid Cloud solutions are designed in Azure.

Who this book is for

The book is aimed at IT architects, IT professionals and DevOps engineers who plan to implement Azure solutions for their organization.

What this book covers

Chapter 1, Getting Started with Azure Implementation, this chapter will help you understand how the basic services of Azure make up the core of an application running in Azure. We will also give the reader an impression how Azure influences Microsoft's products and product strategy. We will explain the different Cloud Models and Multi Cloud strategies in conjunction with Microsoft Azure too.

Chapter 2, Azure Resource Manager and Tools, this chapter describes the Azure Resource Manager (ARM) concept, the ARM Tools Instrumentation and how it works. We show: Working with the Azure Portal and working with Azure PowerShell. Last, we will also describe the differences between classic deployment and ARM.

Chapter 3, *Deploying and Synchronizing Azure Active Directory*, this chapter will describe how to deploy Azure Active Directory, how to secure it for the next following steps and give some best practice advises when using Azure Active Directory together with other Microsoft Services like Office 365. Within the chapter we will describe how Azure Active Directory Synchronization could be implemented. We will give best practices which Synchronization method is the best for different environments. We will also explain how to secure and filter which accounts and attributes are synced.

Chapter 4, *Azure Managed Applications*, based on the knowledge about ARM templates provided in Chapter 2, *Azure Resource Manager and Tools*, in this chapter you will learn the latest ways to provide your solutions. Azure Managed Applications give you the power to share your solutions with Azure Service Catalog for your enterprise or through Azure Marketplace for your customers.

Chapter 5, *Implementing Azure Networks*, the reader will learn how to deploy and configure virtual networks in Azure and will get some best practice advises about working with subnets and network splitting. We will also provide an overview about routing in Azure and Network Devices in Azure.

Chapter 6, *Implementing Azure Storage*, in this chapter, the readers will get to know how to implement storage accounts in azure, the differences between accounts and give a brief overview about the different usage scenarios.

Chapter 7, *Virtual Machines in Azure*, in this chapter, we will describe how to decide which type of virtual machines we need for an environment and application. How to deploy this virtual machine within a resource group and connect it to network and storage. We will also provide information how to connect to this Machines and put them into Active Directory Domain Services.

Chapter 8, *Implementing Azure-Managed Kubernetes and Azure Container Service*, this chapter will describe the general concept behind containers in Azure work, the need to have Kubernetes as an orchestrator, how AKS works and where Azure Managed Instances make sense, how they are created, deployed and managed.

Chapter 9, *Implementing Azure Cloud Services*, in this chapter the different types and usage scenarios of cloud services would be described. The readers will be shown how to implement basic cloud services.

Chapter 10, *Implementing Azure Governance*, this chapter will describe the basics of Azure Governance including Azure Policies, Azure Role Based Access Control, Resource Tags and will add how to secure an Azure environment using Azure Security Tools.

Chapter 11, *Azure Hybrid Data Center Services*, this chapter will give you an overview how to implement Azure Hybrid DataCenter Services using Azure Stack.

Chapter 12, *Working with Azure Functions*, this chapter will describe the general concept behind Serverless Compute Concept and Azure Functions in Azure work and how they are created, deployed and managed. To read this chapter, go to the link https://www.packtpub.com/sites/default/files/downloads/Working_with_Azure_Functions.pdf.

Chapter 13, *Implementing Monitoring and Telemetry*, within this chapter, we will learn the basics about Monitoring and Telemetry, and how we can use Monitoring and Telemetry for our solutions. To read this chapter, go to the link https://www.packtpub.com/sites/default/files/downloads/Implementing_Monitoring_and_Telemetry.pdf.

Chapter 14, *Azure IoT*, within this chapter we will explain you how to implement an Azure IoT Solution and how to operate within different scenarios. To read this chapter, go to the link https://www.packtpub.com/sites/default/files/downloads/Azure_IoT.pdf.

Chapter 15, *Implementing Azure App Services*, in this chapter you learn the basic knowledge about Azure AppServices. The readers will be shown how to plan, build and deploy app services. To read this chapter, go to the link at https://www.packtpub.com/sites/default/files/downloads/Implementing_Azure_App_Services.pdf.

To get the most out of 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 at no 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, I recommend using the free Visual

Studio Community Edition from <https://www.visualstudio.com/downloads/>.

Download the example code files

You can download the example code files for this book from your account at www.packt.com. If you purchased this book elsewhere, you can visit www.packt.com/support and register to have the files emailed directly to you.

You can download the code files by following these steps:

1. Log in or register at www.packt.com.
2. Select the **SUPPORT** tab.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the book in the **Search** box and follow the onscreen instructions.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR/7-Zip for Windows
- Zipeg/iZip/UnRarX for Mac
- 7-Zip/PeaZip for Linux

The code bundle for the book is also hosted on GitHub at <https://github.com/PacktPublishing/Implementing-Azure-Solutions-Second-Edition>. In case there's an update to the code, it will be updated on the existing GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at <https://github.com/PacktPublishing/>. Check them out!

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: https://www.packtpub.com/sites/default/files/downloads/9781789343045_ColorImages.pdf.

Conventions used

There are a number of text conventions used throughout this book.

CodeInText: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "Not all resource types support the `export template` function."

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:

```
{
  "$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"
      }
    }
  },
}
```

Any command-line input or output is written as follows:

```
New-AzureRmStorageKey -ResourceGroupName "MyResourceGroup" -AccountName
"MyStorageAccount" -KeyName "key1"
```

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: "Browsing through the **File Shares** section, the earlier-created file share can be found."



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: If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at customercare@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.packt.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@packt.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 packt.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, already existed. 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 big IT companies started their cloud offerings. That was Amazon, Google, and Microsoft in particular. As these cloud offerings developed, they enabled companies from start ups 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
- Multi-cloud characteristics and models
- An overview of Azure services

Technical requirements

To start with Microsoft Azure and cloud services, you need an active Azure subscription and an Azure tenant, which will be obtained with the subscription. There are different ways to order such an subscription. The following list provides a few options:

- Microsoft MSDN subscription
- Microsoft Azure free trial
- Microsoft Azure pass
- Microsoft **Enterprise Agreement (EA)** with Azure commitment
- Microsoft Azure cloud solution provider
- Microsoft Azure in open licensing
- Microsoft BizSpark program

Service models

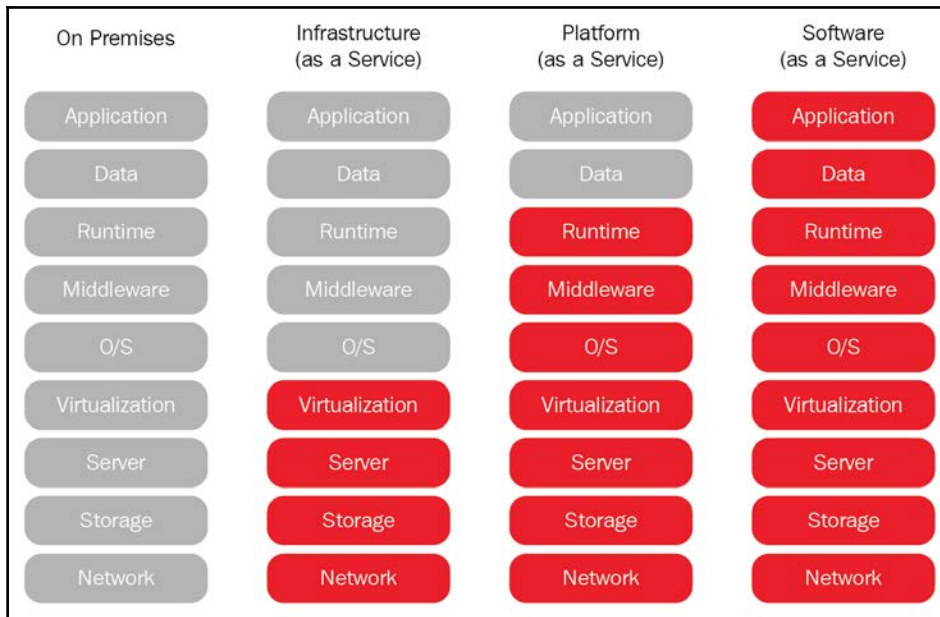
Cloud computing is a new trend model for enabling workloads that use resources from a normally huge resource pool that is operated by a cloud service provider. These resources include servers, storage, network resources, applications, services, or even functions. These can be rapidly deployed, operated,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

Cloud offerings are mainly categorized into the following service models:

- **Infrastructure as a Service (IaaS):** This describes a model where the cloud provider enables the consumer to create and configure resources from the computing layer upwards, without any need to care or know about the hardware layer. That includes virtual machines, networks, appliances, and lots of other infrastructure-related resources and services. The most popular IaaS resources in Azure contain virtual machines, virtual networks (internal and external), container services, and storage.
- **Platform as a Service (PaaS):** This 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):** This is the model with the lowest level 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, the Outlook website, OneDrive, and even the Amazon website itself is a SaaS application with Amazon as its own consumer.

A comparison of service model responsibilities is shown in the following diagram:



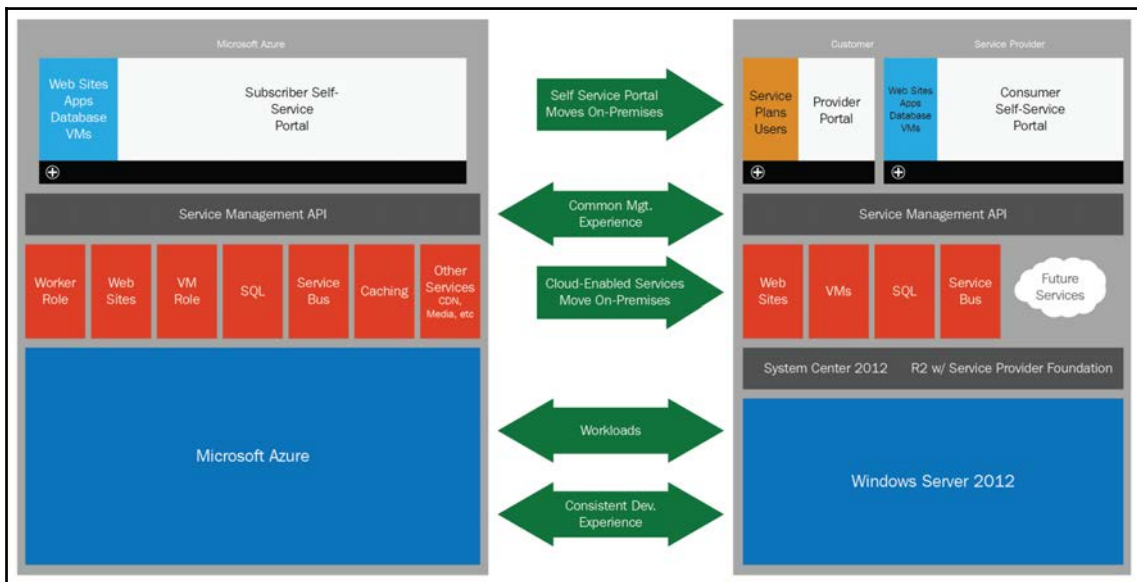
Deployment models

There are also 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:** A 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 the 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 a combination of any other models, such as community and public clouds. Examples are Azure VMs connected to an on-premises infrastructure through Microsoft Azure ExpressRoute or site-to-site VPN.

The following diagram depicts a comparison between Azure (public cloud) and Azure Pack (private cloud):





In the summer of 2017, Microsoft released the new version of the private cloud adoption from Azure Resource Manager. The new version is 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 of the characteristics that the **National Institute of Standards and Technology (NIST)** describes for cloud computing. These are as follows:

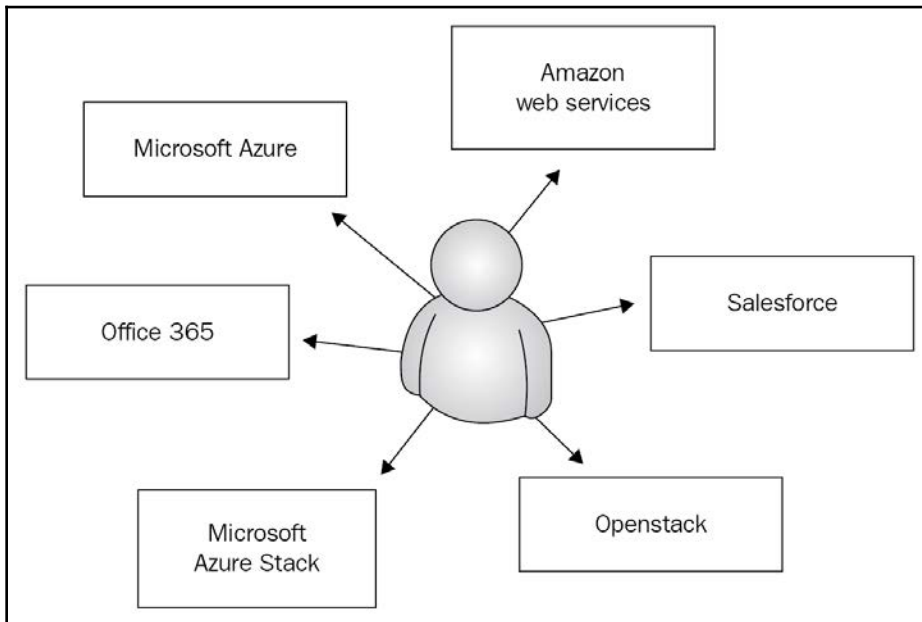
- **On-demand self-service:** This means an automated deployment of resources that a consumer orders through an interface such as a consumer portal.
- **Broad network access:** Providing availability of cloud services through a standardized network interface that is, at best, accessible by several endpoint 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 data usage 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.

Multi-cloud characteristics and models

When defining multi-cloud, you need first to be aware of what a cloud service is. At this stage of this book, you already had some insight into cloud computing and cloud models and characteristics. Now, you should be able to identify the cloud services you already use in your company and that you might use in the future.

Multi-cloud means you or your company are using not only the services of one cloud provider, but different solutions from different cloud providers. That could be an example of using Microsoft Office 365 for business collaboration, Salesforce for CRM, and AWS Area 52 for GeoDNS and GeoIP, or even OpenStack or Azure Stack as your private cloud solution within your data center or co-location.

The following diagram shows a schematic definition of a person or company between multiple cloud providers:



Why use multiple cloud providers and not only one that fits all? There are different reasons why someone chooses a multi-cloud solution. Let me explain the most common reasons in the field:

- **Redundancy:** You don't want to build up your environment on only one cloud provider because one can fail, as happened with AWS in the past. So, you want to keep the business running with the services of another cloud provider. That's mostly a reason when using IaaS or PaaS. Redundancy is mostly not possible with SaaS if the cloud provider does not support hybrid environments.

- **The solution does not fit my needs:** Mostly when choosing a cloud solution, you see whether it fits your need. You mostly look to features such as data center location or performance. Sometimes, a cloud solution from my preferred provider does not fit those needs, so I need to choose another cloud provider with its solution. Often, you see that in Microsoft Dynamics CRM Online versus Salesforce, or your preferred provider does not offer a data center in South Africa. So, you may switch from AWS to Microsoft Azure for that reason.
- **The cloud provider does not offer the service I need:** Often, cloud providers are strong in one field and less so in others. This means they don't offer the services you may want; for example, you use Salesforce and want to have a unified single sign-on solution with Facebook, Twitter, or Instagram for your marketing teams. That's a service Salesforce does not offer at the moment, which means you may want to include Microsoft Azure **Active Directory (AD)** in your environment to achieve your goal.
- **Your departments use a cloud service as shadow IT:** I have seen shadow IT in nearly every company in the last 12 years of my work experience. It means a department uses a solution outside of the IT controlled area or solution field, managing the application itself without IT knowing of it. Often, it happens that those solutions become business critical and C-level management forces IT to take over the solution and support it. In times of easily accessible cloud solutions, this issue increased dramatically. There are mostly two reasons for shadow IT:
 - IT departments aren't fast enough to deploy an appropriate on-premises solution
 - The user thinks, *Okay I only need a credit card? Let's try.*

The key elements to building and performing a successful multi-cloud solution is to build a uniform solution between all of the cloud providers. Those solutions are based on a uniform **Identity and Access Management (IAM)**, network, and application infrastructure.

Within this field, you might see two flavors of multi-cloud.

Cloud brokering

With cloud brokering, you migrate your workload depending on the price and needs from one cloud provider to another. That can be on a day-to-day or more frequent basis. This brokering was the first intention of businesses to save money with the cloud, but in practice, brokering only works with very simple IaaS or very standardized PaaS solutions. Most of the more complex workloads, such as Microsoft Exchange, SAP, and Oracle depend on drivers and you always have different hypervisor solutions between your cloud providers. In addition to that, IaaS workloads are very costly compared with solutions built on PaaS. So, looking down and ahead the timeline, the second multi-cloud model has become more common—**best of breed**.

Best of breed

Best of breed means you choose your cloud provider and a solution that fits for your needs and business requirements, or that is the market leader in a special area, for example, artificial intelligence, **Network as a Service (NaaS)**, collaboration software, or data center distribution. Mostly, that means you will always end up with three or more cloud providers integrated with each other.

Microsoft Azure

When Windows Azure came online for the general public in February 2010, there were only 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 offerings for Linux and other non-Windows services, Microsoft decided in April 2014 to rename Windows Azure to 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 expands data centers all over the world.

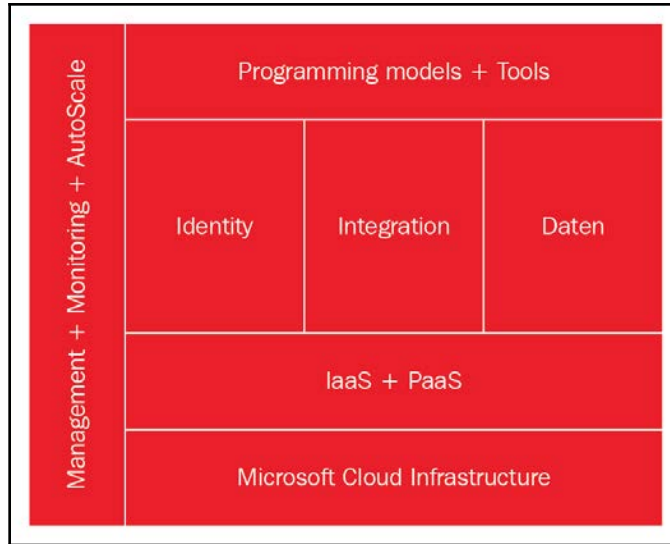


Service updates happen very frequently. That is the reason why you need to keep yourself informed. For example, the database offering you are using could have improved storage or performance capabilities.

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.

The platform services are as follows:

- **Management:** These services include the management portal, the marketplace with the services gallery, and the components to automate things in Azure.
- **Compute services:** Compute services are Azure cloud services that are basically PaaS offerings for developers to quickly build and deploy highly scalable applications. The service fabric and Azure RemoteApp are also in this category.
- **Security:** This contains all of the services that provide identity in Azure, such as Azure AD, multi-factor authentication, and the key vault, which 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, which makes it possible to build your own content delivery network based on Azure. The other is media services that make it very easy to use and process different media with the help of Azure.
- **Web and mobile:** These include all of the services that assist in creating apps or backend services for the web and mobiles, 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 of 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. The **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 of the remaining services that could not clearly be categorized. These include backup, monitoring, and disaster recovery, as well as many others.

The infrastructure services are as follows:

- **Operating system and server compute:** This category consists of compute containers. It includes virtual machine containers and, additionally, container services, which 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, Implementing Azure Storage*.
- **Networking:** This category consists of 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 an 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 (ARM)

In the previous major version of Azure, a deployment backend model called **Azure Service Manager (ASM)** was used. With higher demand on scaling, and being more flexible and standardized, a new model called **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 ARM 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 section 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, such as a VM, a network interface card, a storage account, a 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 an 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, which 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 the resources 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 to. One region consists of at least one highly available data center or data center complex. At the time of writing, 54 regions are distributed all over the world and include community clouds, so-called sovereign regions.

Microsoft also divides its regions into geopolitical zones, which can be found at the following URL: <https://azure.microsoft.com/en-us/global-infrastructure/regions/>.

These sovereign clouds were built by Microsoft to fit customer or governmental needs, such as for special compliance and/or data privacy laws. At the moment, the following sovereign clouds are available:

- Microsoft Azure US **Department of Defense (DoD)**
- Microsoft Azure US Government
- Microsoft Azure China
- Microsoft Cloud Germany

Microsoft Cloud Germany is also special among the sovereign clouds. Because of customer demands, Microsoft built up Microsoft Cloud Germany differently. Microsoft does not operate the cloud in Germany itself; they use a data trustee to operate the cloud for them. Microsoft Azure staff and all Microsoft employees are not allowed to enter the data centers or lay hands on the servers or framework. Everything is operated by the trustee, starting with hardware maintenance up to updates of the framework.



Fun fact: Before Microsoft moved into its data center in Berlin, I used to be allowed to walk straight through the data center with a guide to reach my peer, who is a regional director of the data center provider. Since Microsoft moved into the data center, I can no longer use the shortcut and need to walk around the outside of the building to reach the office of my buddy. So, Microsoft is very serious with their policies.

Regions can also have an impact on the performance and availability of some resources. Some services may not be, or are 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 to see which legal requirements must be met. This could, for example, result in a deployment only in EU regions, or even regions in specific countries:

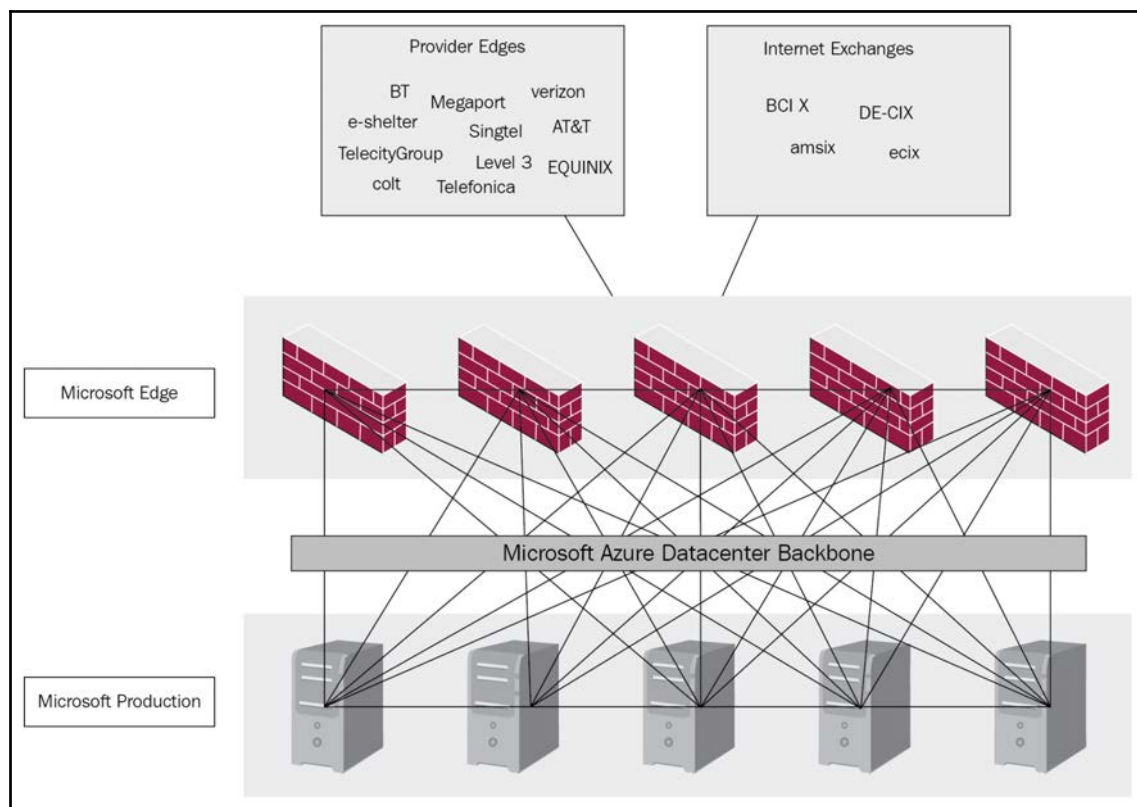
- **Available Azure regions:** <https://azure.microsoft.com/en-us/regions/>
- **Lists all the services available in specific regions:** <https://azure.microsoft.com/en-us/regions/services/>

Microsoft data center and backbone

Microsoft operates two types of following data centers:

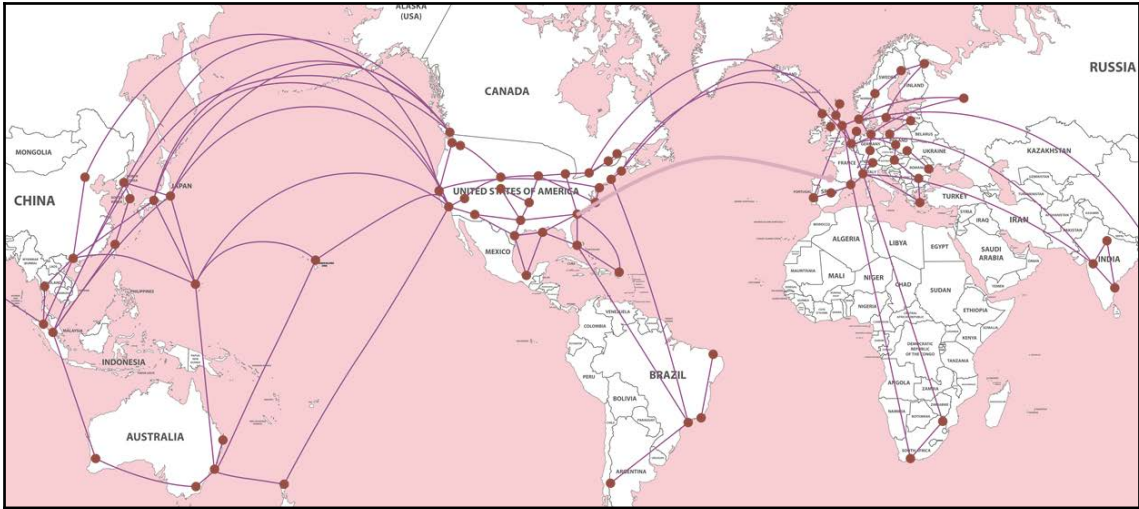
- The first type is the production data center, where Microsoft calculates all workloads of its customers and stores all the data.
- The second type is the edge or delivery site. Those sites connect all Microsoft Cloud services to the internet and Microsoft's customers. Edge sites come in two stages of expansion. The smallest one allows Microsoft public direct peering through the internet. With the second stage of expansion, Microsoft allows customers and providers to establish a private connection to the Microsoft backbone using the Microsoft Azure ExpressRoute service.

The following diagram shows a schematic of the Microsoft data center structure:



Edge and production sites are connected through the Microsoft backbone. Currently Microsoft owns and operates the second largest and fastest full meshed provider backbone of the world.. Microsoft also owns and operates own sea cables such as the MAREA cable from Bilbao (Spain) to Virginia (US).

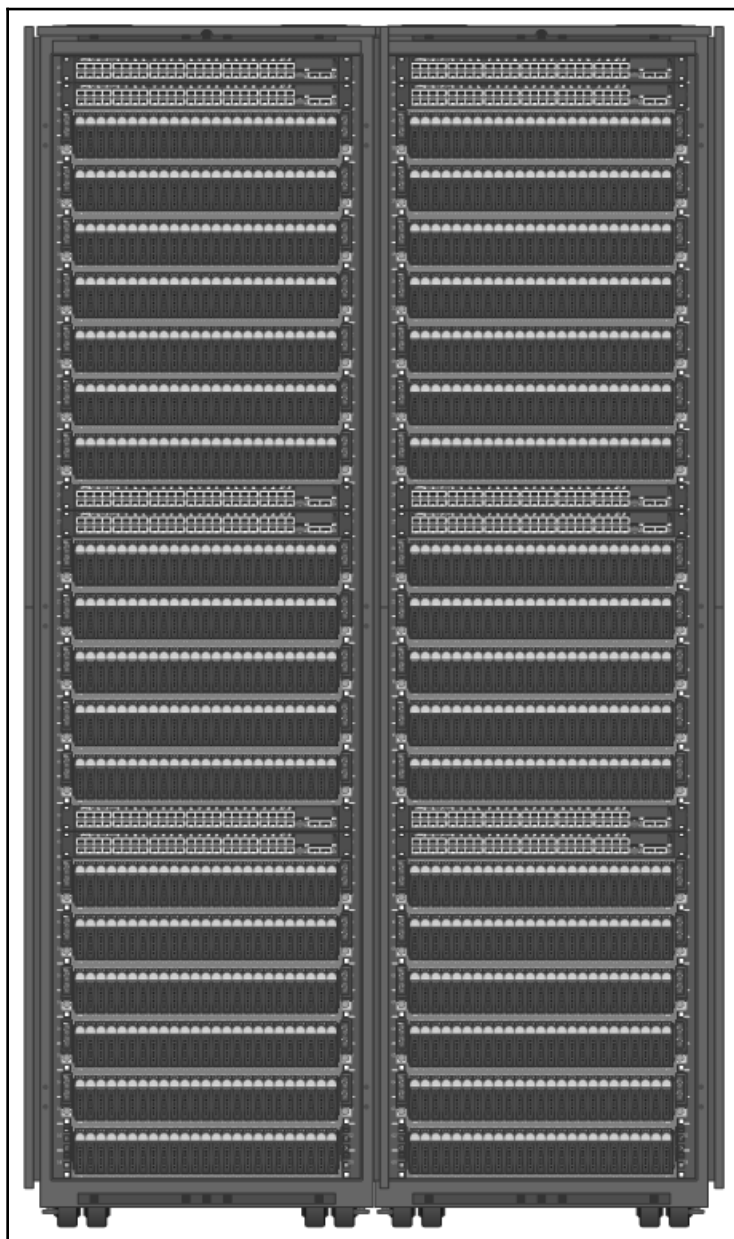
This map shows the current Microsoft Azure backbone with the new MAREA cable:



Fun fact: *What was the hardest thing for the Microsoft backbone teams when building the MAREA cable?* To create and get the purchase order for the submarine approved because of Microsoft processes.

While building its backbone, Microsoft acts differently to the other cloud providers. Microsoft builds its own dark fibre cables or leases dark fibre cables and operates the whole backbone itself. Microsoft runs a fully software-defined network and infrastructure for its backbone, using firewall appliances built for **network function virtualization**.

If you ever have the chance to see a server rack that connects the Microsoft backbone or represents a Microsoft Edge site, it will probably look like this:



If you want to know more about Microsoft regarding data center equipment and software defined, I highly recommend you consult open source and open compute projects. Microsoft is investing highly in these and is very open in the following projects:

- **Microsoft cloud servers:**
 - **Open cloud server platform:** <https://www.opencompute.org/projects/server>
 - **ARM-based cloud server project olympus:** <https://www.opencompute.org/wiki/Server/ProjectOlympus>
- **Microsoft network cards for backbone and cloud services:**
 - **Smart NIC:** <https://www.opencompute.org/wiki/Server/Mezz>
- **Microsoft networking and switch software:**
 - **Project SONiC:** <https://azure.microsoft.com/de-de/blog/sonic-the-networking-switch-software-that-powers-the-microsoft-global-cloud/>



Microsoft also makes heavy use of **Field Programmable Gateway Arrays (FPGAs)**, to make Azure as flexible as possible and adjust the hardware layer as much as possible to the needs of their workloads. If you really want to become an insider in this technology, I would high recommend the session, *Inside Microsoft's FPGA-Based Configurable Cloud*, by Mark Russinovich, CTO of Azure. You can find the session here: https://www.youtube.com/watch?v=v_4Ap1bjwgs.

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 of any rapid deployment plans.

As Azure uses a lot of 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 Azure PowerShell and the Azure **Command-Line Interface (CLI)**.

Both are basically just wrappers around the Azure API to enable everyone not familiar with RESTful APIs, but familiar with their specific scripting language, to use and automate Azure. The Azure PowerShell module provides `cmdlet` for managing Azure services and resources through the Azure API. Azure PowerShell `cmdlet` are used to handle account management and environment management, including creating, updating, and deleting resources. These `cmdlet` work completely the same on Azure, Azure Pack, and the 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, such as 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 an Azure SDK for numerous programming languages (for example, .NET, Java, and Node.js) here at <https://github.com/Azure>.

Summary

In this chapter, we learned about cloud models and what cloud in general means. We now know how Microsoft fits into that ecosystem with its cloud services and their strategy. We also gained some very important insights into Azure and Microsoft regarding their data centers and backbone.

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

Questions

Please answer the following questions:

1. What are the three basic cloud service models?
2. What are four basic cloud deployment models?
3. How is a multi-cloud solution described?
4. What is the difference between a Microsoft Azure Global Region and a Sovereign Region?
5. What is the difference between a Microsoft Edge and production data center?
6. What is the URL of the Azure portal?
7. What is the name of the Microsoft private cloud solution based on ARM?

Further reading

In the following books, you can find more information about what we learned in this chapter:

- **Building Hybrid Clouds with Azure Stack:** <https://www.packtpub.com/virtualization-and-cloud/building-hybrid-clouds-azure-stack>
- **Beginning Serverless Architectures with Microsoft Azure:** <https://www.packtpub.com/virtualization-and-cloud/beginning-serverless-architectures-microsoft-azure>
- **Architecting Microsoft Azure Solutions – Exam Guide 70-535:** <https://www.packtpub.com/virtualization-and-cloud/architecting-microsoft-azure-solutions-exam-guide-70-535>
- **Implementing Azure Cloud Design Patterns:** <https://www.packtpub.com/virtualization-and-cloud/implementing-azure-cloud-design-patterns>

2

Azure Resource Manager and Tools

The Azure platform consists primarily of three parts—**Azure execution model**, which denotes the areas where you can provide your services and applications in the cloud; **Azure Building Blocks**; and **Azure Data Services**, which refers to services that extend the platform to common capabilities and functionalities.

I could actually forgo the description of the platform, because most users only get to see these three parts, but there are still more. Many other services are working under the hood of the platform and ensure its ongoing operation. These services include, for example, **Azure Traffic Manager**, **Azure Load Balancer**, and **Azure Resource Manager (ARM)**. All of these services can be customized using various interfaces for your personal needs.

In this chapter, I'll introduce you to ARM in detail, and we will explore the following topics:

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

Technical requirements

For running containers in a cloud environment, no specific installations are required, as you only need the following:

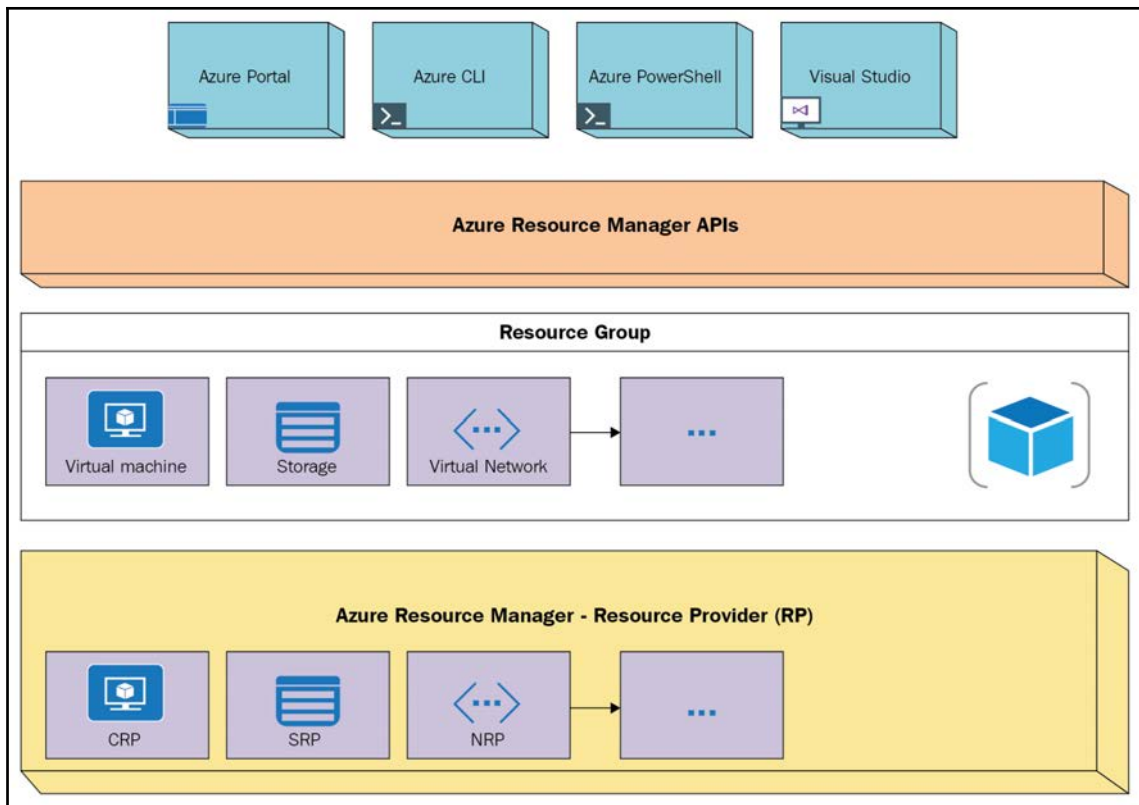
- A computer with an internet browser
- An Azure subscription (if not available, a trial could work too, at <https://azure.microsoft.com/en-us/free/>)
- The code in this chapter can be found here: <https://github.com/PacktPublishing/Implementing-Azure-Solutions-Second-Edition>

Understanding ARM

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 no longer see your components 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, ARM (and ARM tools) was introduced.

Enough of the preliminary remarks. Let's take a look at ARM in detail with the following diagram:



As you can see in the preceding diagram, ARM can be accessed through a variety of different technologies and interfaces. These access options include the following:

- The traditional way, through the Azure portal (version 3.0 and newer)
- The script-based way, through Azure PowerShell (look for PowerShell modules with the `AzureRM` prefix) or through the Azure **Command-Line Interface (CLI)** (cross-platform CLI)
- For developers, through Visual Studio
- For developers, there are also SDKs available (.NET and some other programming languages) and, as with all Azure services, an extensive RESTful API

Let's go through the preceding diagram:

- It consists of one or more Azure resource groups and one or more Azure resources. An Azure resource group is a container (a management unit), that all of the resources of your Azure solution contain. The Azure resource is any form of manageable element available through Azure (for example, a virtual machine, a virtual network, and so on).
- The next section of the diagram is the layer with ARM—the **Resource Provider (RP)**. A resource provider is an internal interface to the platform and offers you numerous operations to handle the resources you need. Each resource type has its own resource provider, for example, a **compute resource provider (CRP)**, **storage resource provider (SRP)**, or **network resource provider (NRP)**.

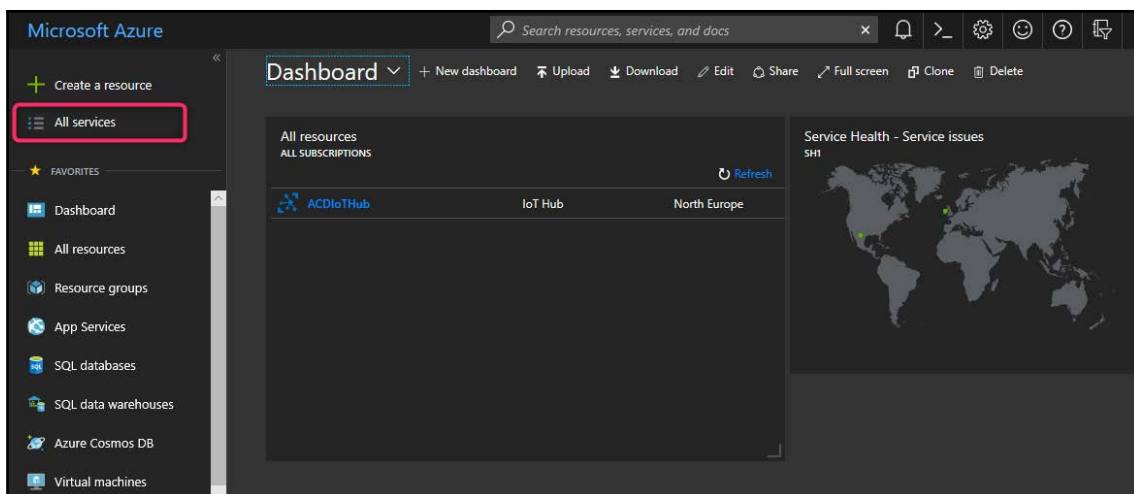
Not shown in the diagram is the template functionality of the ARM. With a so-called ARM template (a text file based on JSON) you determine the details of the creation process of your resource groups or resources or about the configuration settings you require. *Confused about the variety of items?*

For this problem, there is a solution on the Azure platform available—**Azure Resource Explorer**.

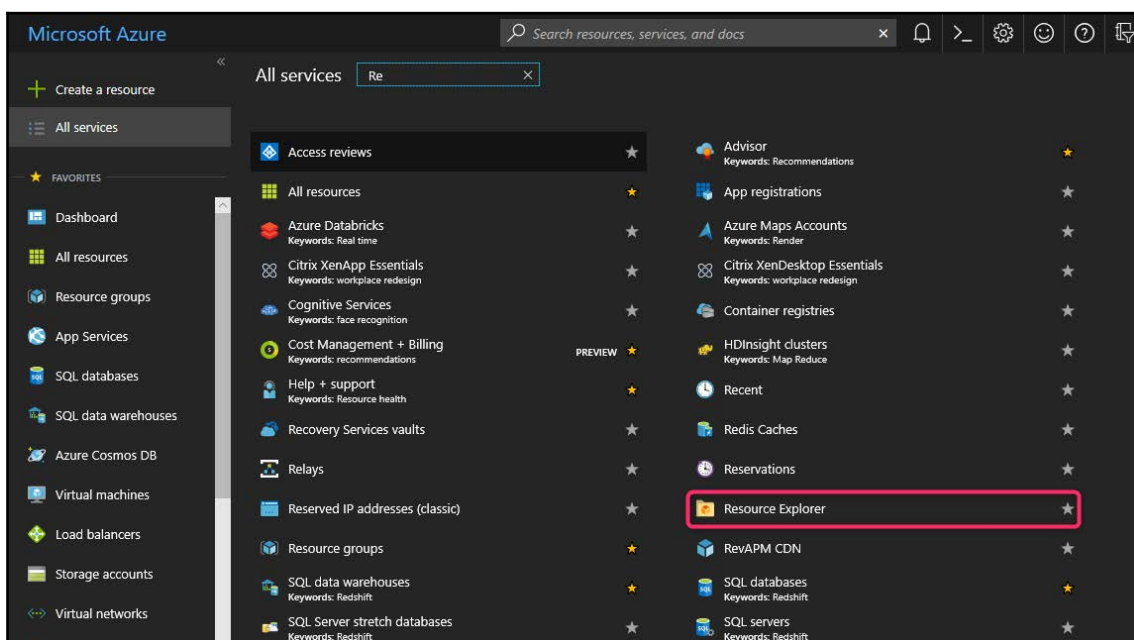
Azure Resource Explorer is a tool for looking at your resources or at the resources of the Azure platform. By using this tool, you can see how the resources are structured and it enables you to view the properties of resources.

How can I find the Azure Resource Explorer? Follow these steps:

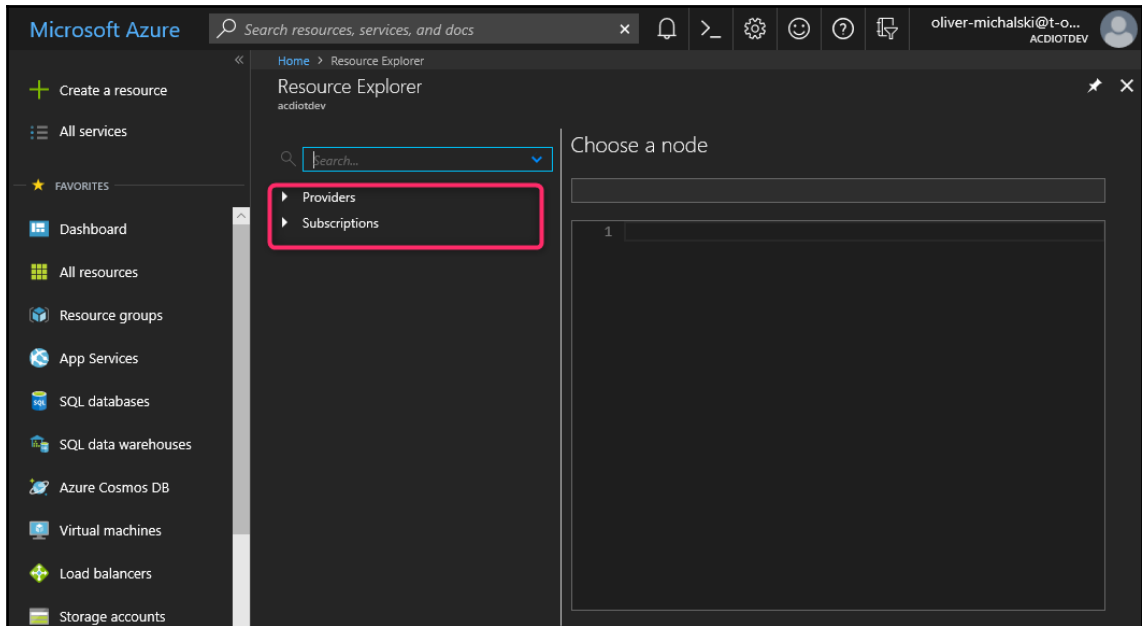
1. Open your Azure Management Portal at <https://portal.azure.com>.
2. In the portal, click on the **All services** option, as shown in the following screenshot:



3. In the list of services, search for **Resource Explorer**, and then click the **Resource Explorer** button:



4. In the **Resource Explorer** tab, you can find the following two nodes:
- **Providers:** A list of all available resource providers from the Azure platform itself
 - **Subscriptions:** A list of resource groups, resources, and resource providers used by yourself:

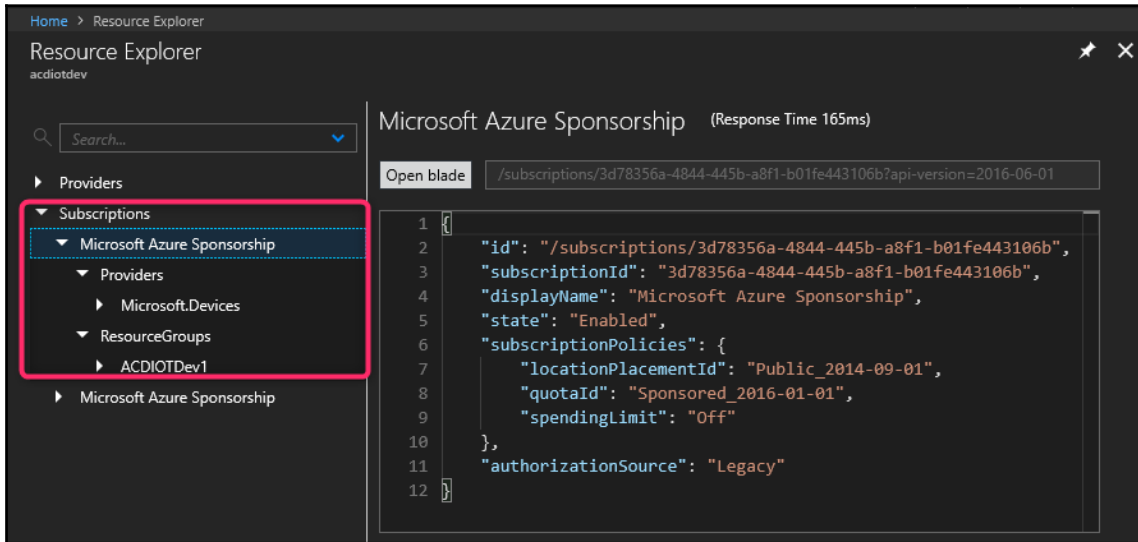


5. By clicking the nodes, you can see the information you want:
 - In the first example for the **Providers** section, note the following:

The screenshot shows the Azure Resource Explorer interface. On the left, the 'Resource Explorer' pane displays a tree of providers. The 'Microsoft.Blueprint' provider is selected and expanded, with its sub-nodes highlighted by a red rectangle. The sub-nodes are: blueprintAssignments, blueprintAssignments/operations, blueprints, blueprints/artifacts, blueprints/versions, and blueprints/versions/artifacts. The main pane on the right shows the details for the 'Microsoft.Blueprint' provider, including the URL '/providers/Microsoft.Blueprint?api-version=2014-04-01-preview' and the JSON response. The JSON response is as follows:

```
1 {
2   "namespace": "Microsoft.Blueprint",
3   "resourceTypes": [
4     {
5       "resourceType": "blueprints",
6       "locations": [],
7       "apiVersions": [
8         "2017-11-11-preview",
9         "2017-11-11-alpha"
10      ]
11    },
12    {
13      "resourceType": "blueprints/artifacts",
14      "locations": [],
15      "apiVersions": [
16        "2017-11-11-preview",
17        "2017-11-11-alpha"
18      ]
19    }
20  ]
21 }
```

- In the second example for the **Subscriptions** section, note the following:



Functionalities provided by ARM

In this section, I would like to give you a brief overview of the functionalities of ARM. 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's take a look at the functionalities:

- There's a access control with Azure **role-based access control (RBAC)**.
- There's logical organization of all the resources of a subscription, with Azure resource tags (for example, for each project and tenant).
- There's improved cost control. You can view the costs for the whole group or for a group of resources with the same tag.

- There's the use of 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 life cycle, 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 ARM

We now know that ARM 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 ARM. Then, in the second part, we will look at working with templates.

Before we begin, I want to introduce some very important facts that are crucial for all workflows:

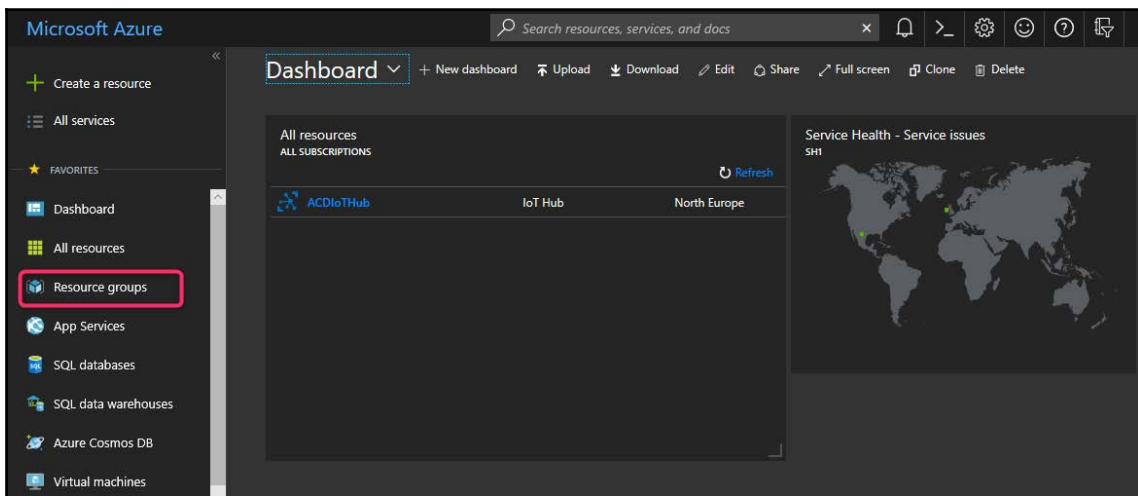
- All of the resources in your resource group have the same life cycle. You will deploy, update, and delete them at the same time.
- Each resource can only exist in one resource group.
- You can add or remove a resource to a resource group at any time. You can also move a resource from one resource group to another.
- A resource group can contain resources that exist in different locations.
- A resource can interact with a resource in another resource group when the two resources are related, but do not share the same life cycle (for example, a web app connecting to a database).

OK, let's start with the first workflow.

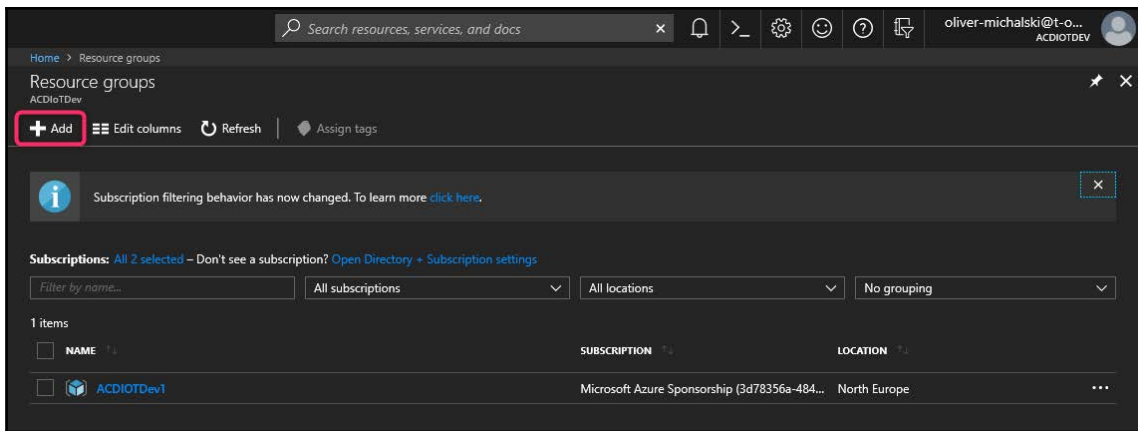
Creating an Azure resource group

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

1. Open your Azure portal at <https://portal.azure.com>.
2. In the portal, click on the **Resource groups** blade, as shown in the following screenshot:

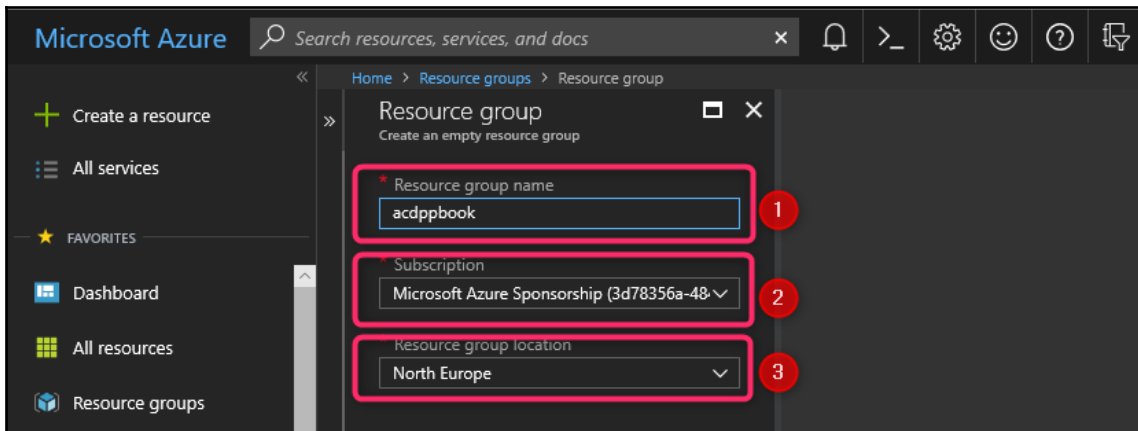


3. On the **Resource groups** blade, click on the **Add** option:

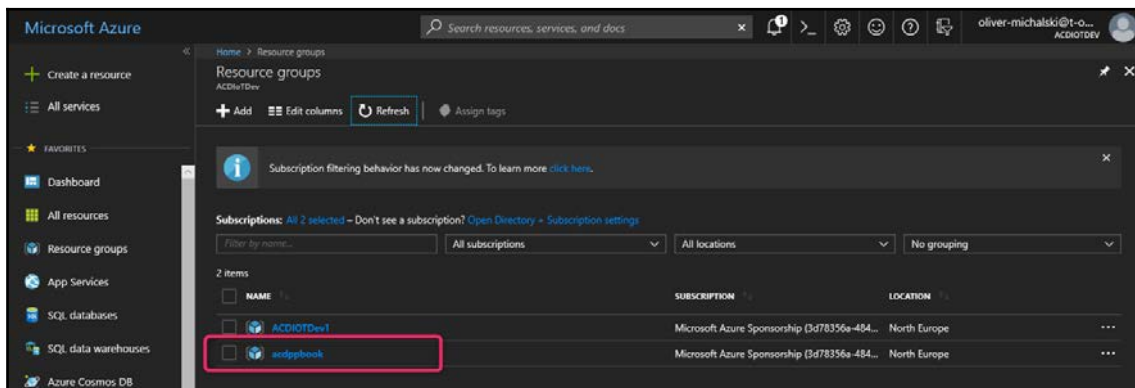


4. On the **Resource groups** blade, type the following values. After that, click on the **Create** button:

- **Resource group name:** acdppbook, as used in the following screenshot (or the name of your choice).
- **Subscription:** Use the default subscription.
- **Resource group location:** Select your preferred location:



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



Adding a resource to a resource group

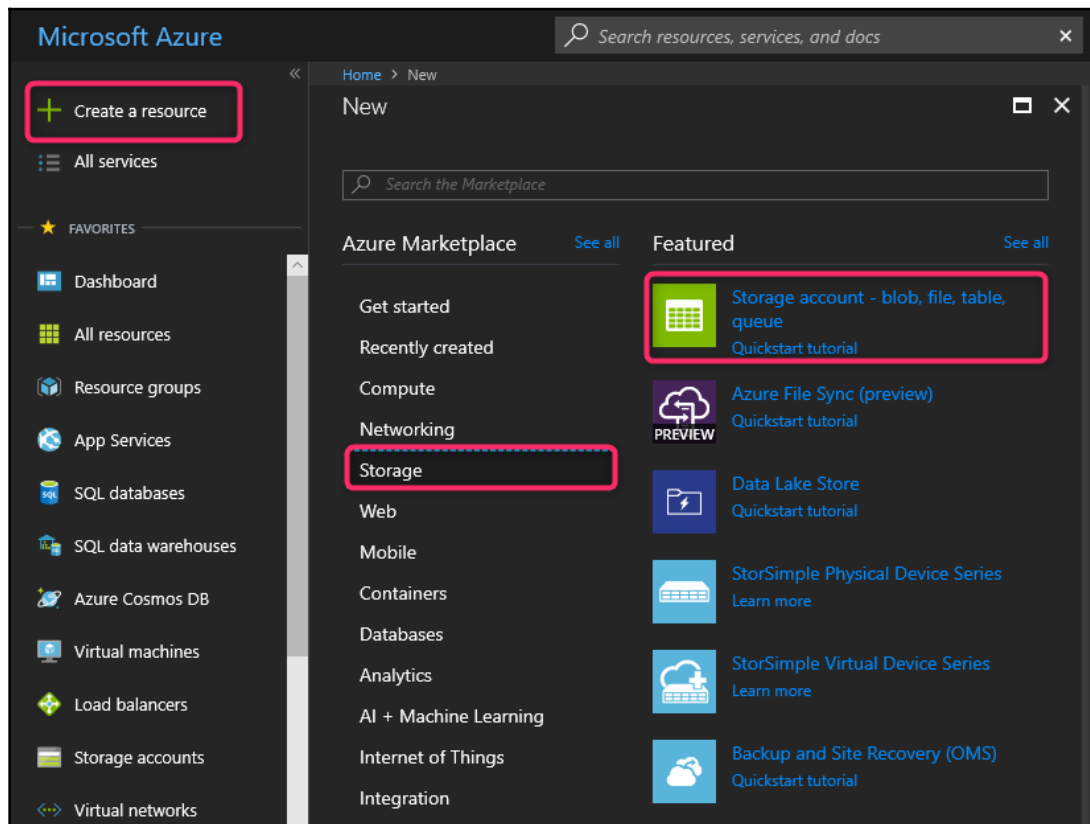
We have just learned how to create an Azure resource group. Now, we'll fill the new resource group with life and add a resource. To complete this process, the Azure platform has a total of two possible approaches. I will now introduce you to them one by one.

For all approaches, I will show the necessary work steps for 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 resource group

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

1. Open your Azure portal at <https://portal.azure.com>.
2. In the portal, click on **Create a resource** | **Storage** | **Storage account - blob, file, table, queue**, as shown in the following screenshot:



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:

Search resources, services, and docs

Home > New > Create storage account

Create storage account

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

* Name ⓘ
acdppbookstoring ✓
.core.windows.net

Deployment model ⓘ
Resource manager Classic

Account kind ⓘ
Storage (general purpose v1) ▾

* Location
North Europe ▾

Replication ⓘ
Read-access geo-redundant storage (RA-GRS) ▾

Performance ⓘ
Standard Premium

* Secure transfer required ⓘ
☐ Pin to dashboard

Create Automation options

4. In the **Location** list, select the same location you have been using for the resource group:

The screenshot shows the Microsoft Azure portal interface. On the left is a sidebar with navigation options: 'Create a resource', 'All services', 'FAVORITES', 'Dashboard', 'All resources', 'Resource groups', 'App Services', 'SQL databases', 'SQL data warehouses', 'Azure Cosmos DB', 'Virtual machines', 'Load balancers', 'Storage accounts', and 'Virtual networks'. The main area displays the 'Create storage account' wizard. The breadcrumb path is 'Home > New > Create storage account'. The wizard title is 'Create storage account'. Below the title, it states: 'The cost of your storage account depends on the usage and the options you choose below. [Learn more](#)'. The form fields are: 'Name' (text input with 'acdppbookstoring' and a '.core.windows.net' suffix), 'Deployment model' (radio buttons for 'Resource manager' and 'Classic'), 'Account kind' (dropdown menu showing 'Storage (general purpose v1)'), 'Location' (dropdown menu showing 'North Europe', highlighted with a red box), 'Replication' (dropdown menu showing 'Read-access geo-redundant storage (RA-...)' and 'Performance' (radio buttons for 'Standard' and 'Premium'). At the bottom, there is a 'Secure transfer required' checkbox and a 'Pin to dashboard' checkbox. The 'Create' button is highlighted in blue, and there is a link for 'Automation options'.

5. In the **Resource group** blade, click the **Use existing** checkbox, then search for and select `acdppbook` in the drop-down list, and then click on the **Create** button, as shown in the following screenshot:

Search resources, services, and docs

Home > New > Create storage account

Create storage account

Replication ⓘ
Read-access geo-redundant storage (RA-... ▼

Performance ⓘ
Standard Premium

* Secure transfer required ⓘ
Disabled Enabled

* Subscription
Microsoft Azure Sponsorship (3d78356a-48-▼

* Resource group
☐ Create new ☒ Use existing

ACDIOTDev1
acdppbook

Configure virtual networks
Disabled Enabled

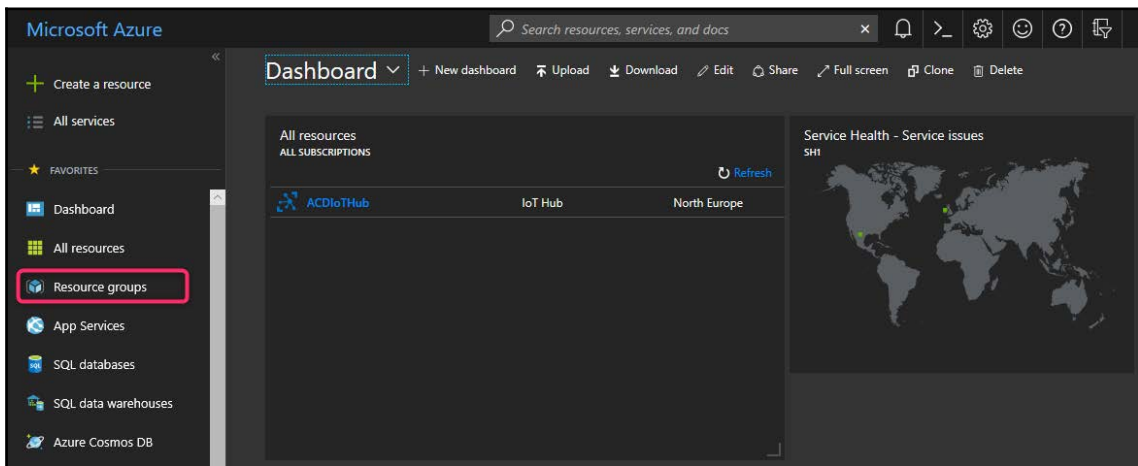
☐ Pin to dashboard

Create Automation options

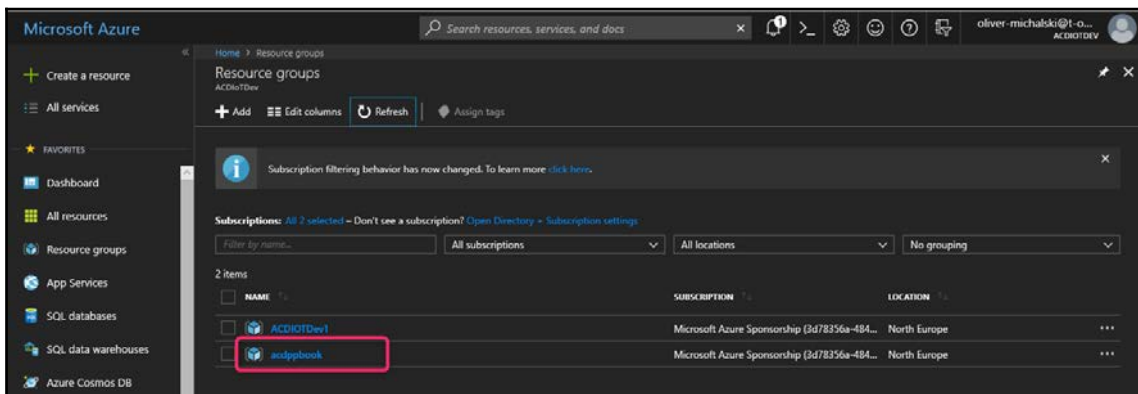
Second approach – adding a storage account to your resource group

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, through the Azure Marketplace. Let's have a look:

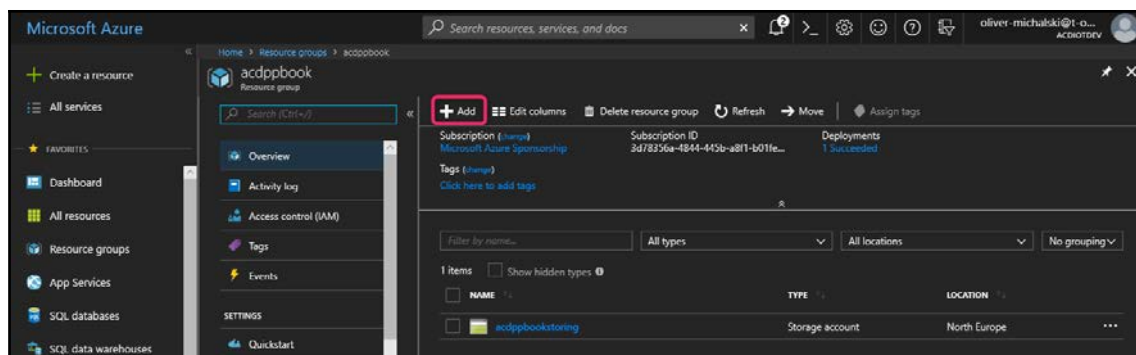
1. In the portal, click on the **Resource groups** blade:



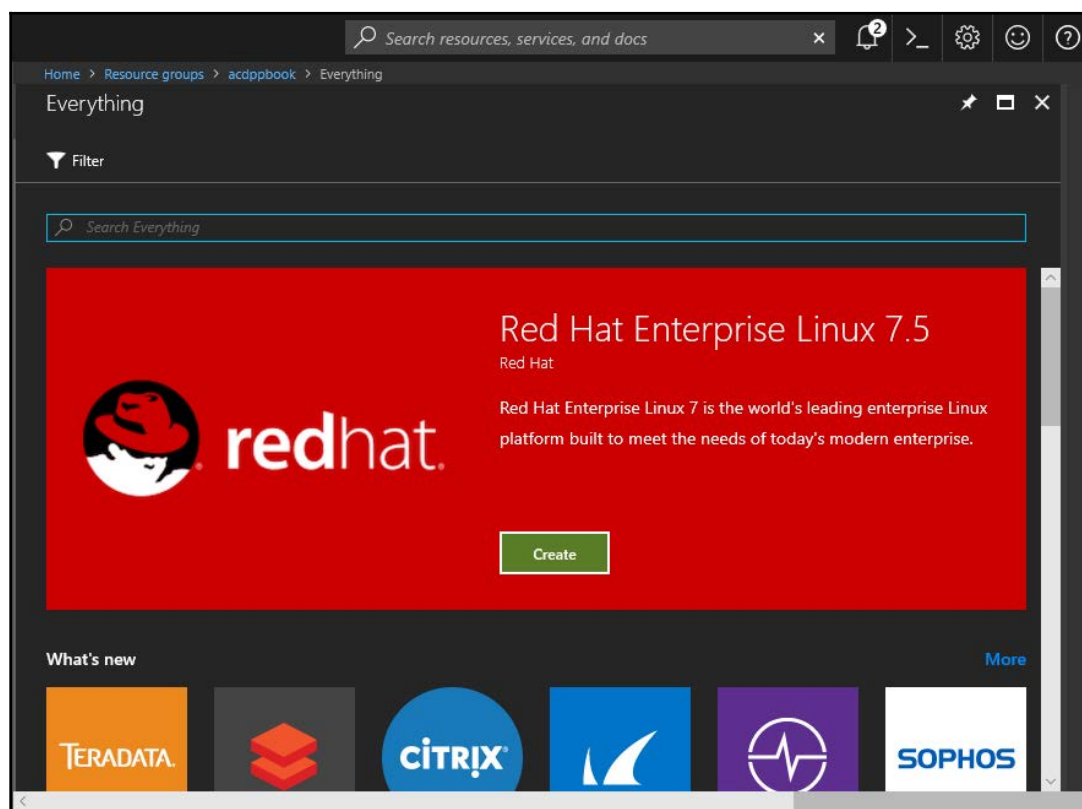
2. On the **Resource groups** blade, click on the `acdppbook` name field:



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:



Tagging in ARM

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

ARM offers a solution for this—**Azure resource tags**. Resource tags are any key/value pairs that appear useful for describing a resource.

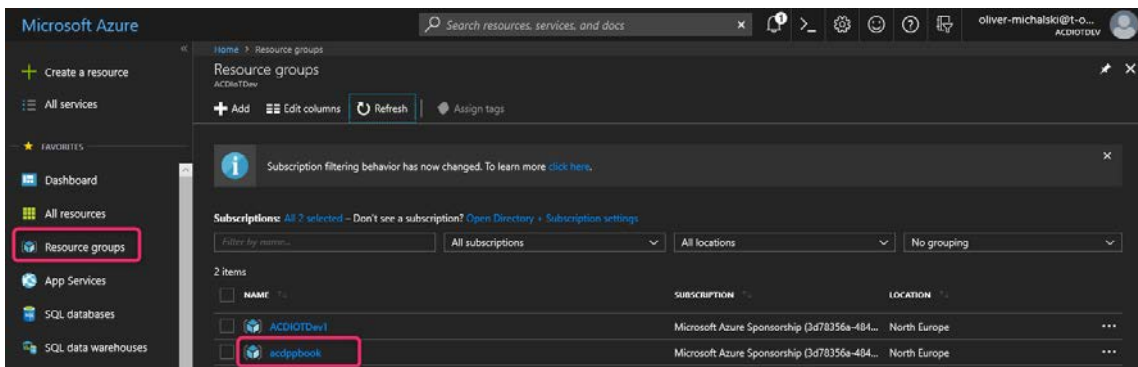
Let's see an example:

Key	Value
Department	Management
Project	PPBook
Tenant	ACD

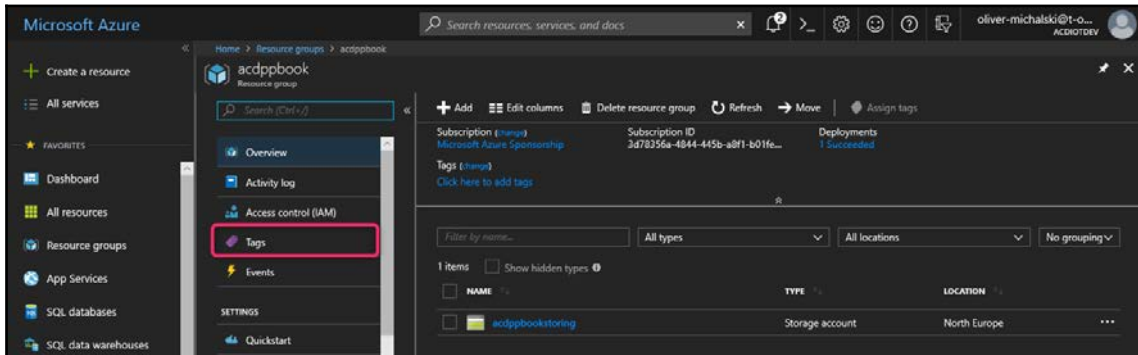
Once you have defined a resource tag, you can use this as a filter in Azure PowerShell or in the Azure Billing APIs (Azure Usage API and Azure RateCard API). Up to 15 tags can be defined per resource.

I 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:

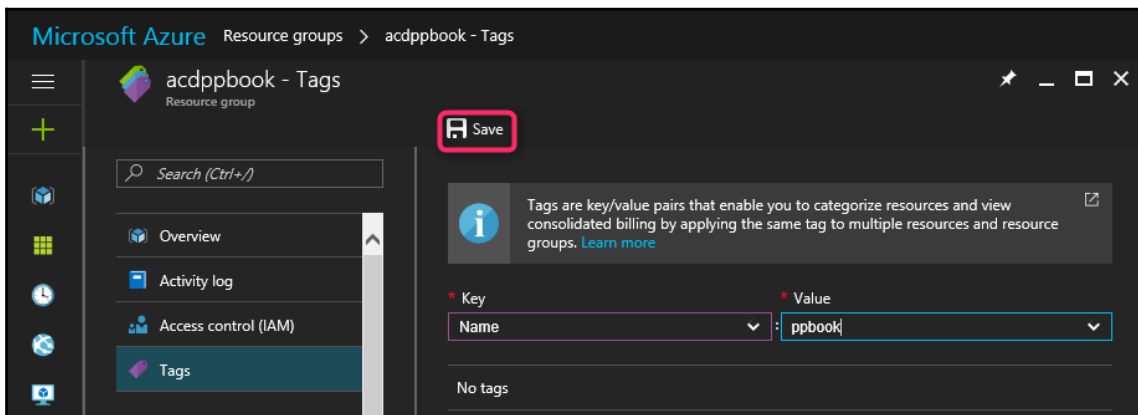
1. Open your Azure portal at <https://portal.azure.com>.
2. In the portal, click on **Resource groups**, and then click on the **Resource groups** blade, then the `acdppbook` name:



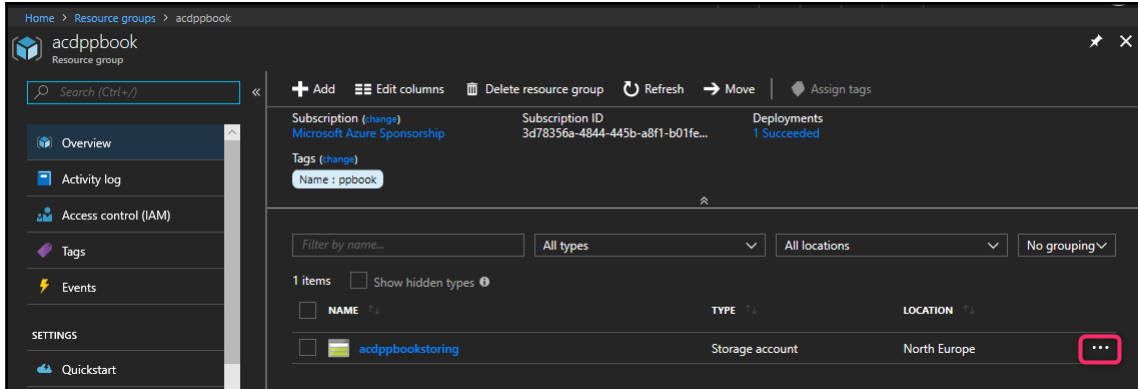
3. In the navigation section on the **Resource groups** dashboard, click on the **Tags** section:



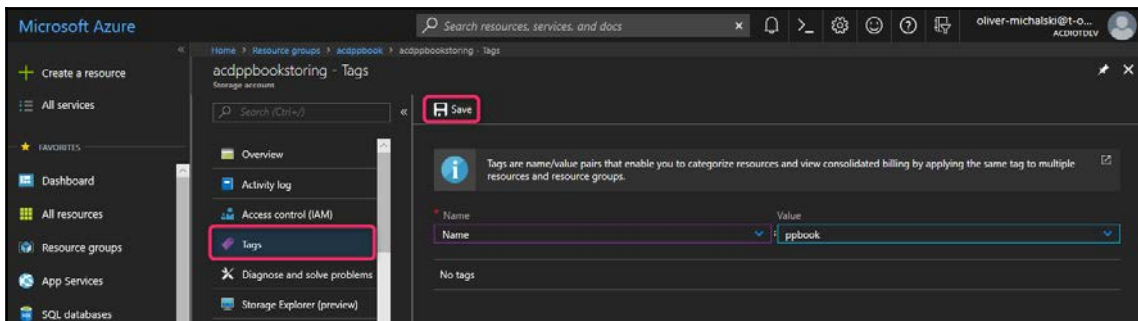
4. Open the **Tags** blade. In the **Name** field, type **Name**, and then in the **Value** field, type **ppbook**. After this, click on the **Save** button:



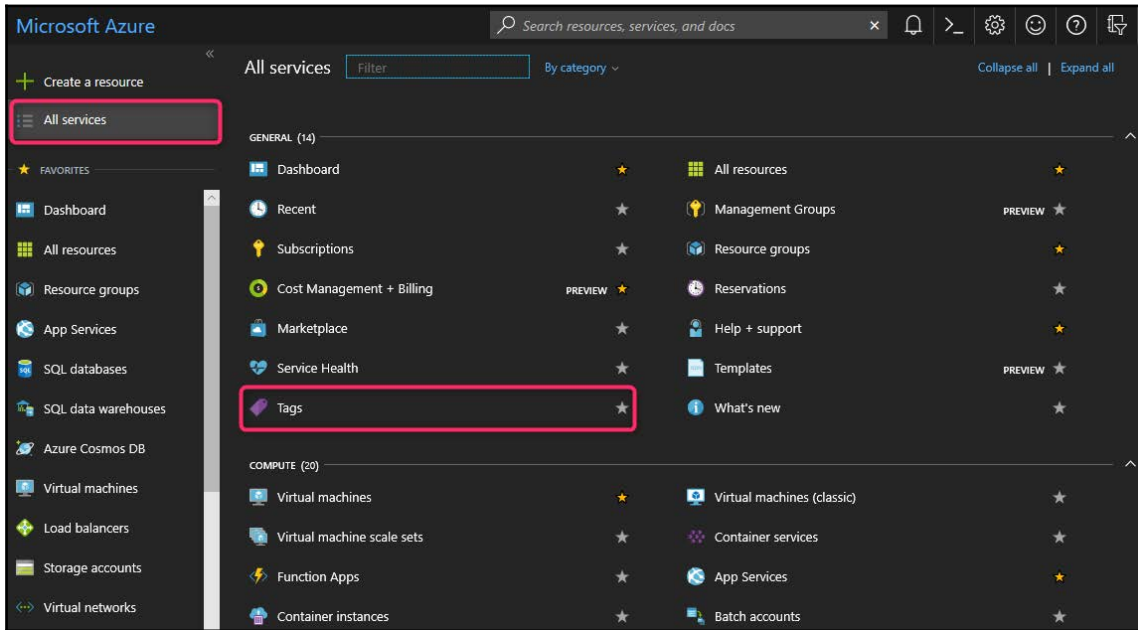
5. Go back to the **Resource groups** dashboard. In the resource grid, select the `acdppbookstoring` row, and then click the `...` button, as shown in the following screenshot:



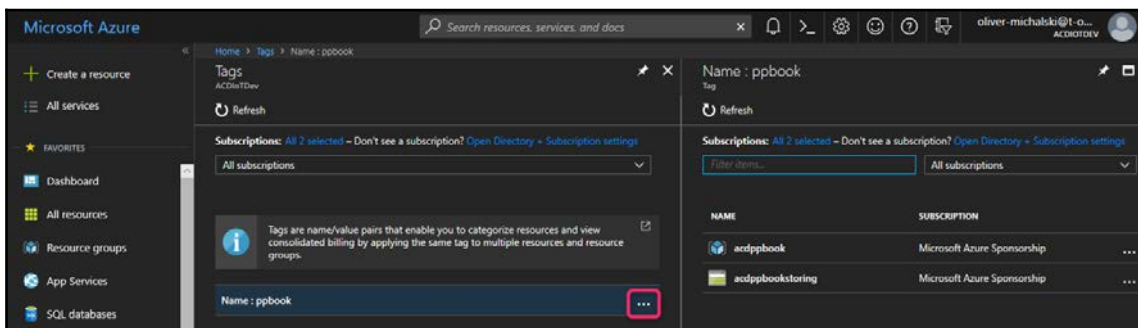
6. In the navigation section of the **Resource groups** dashboard (`acdppbookstoring`), click the **Tags** section. In the **Name** field, type `Name` again, and then in the **Value** field, type `ppbook`. Click on the **Save** button, as shown in the following screenshot:



7. In the portal, click on **All services**, and then click on **Tags**, as shown in the following screenshot:



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



Locking Azure resources

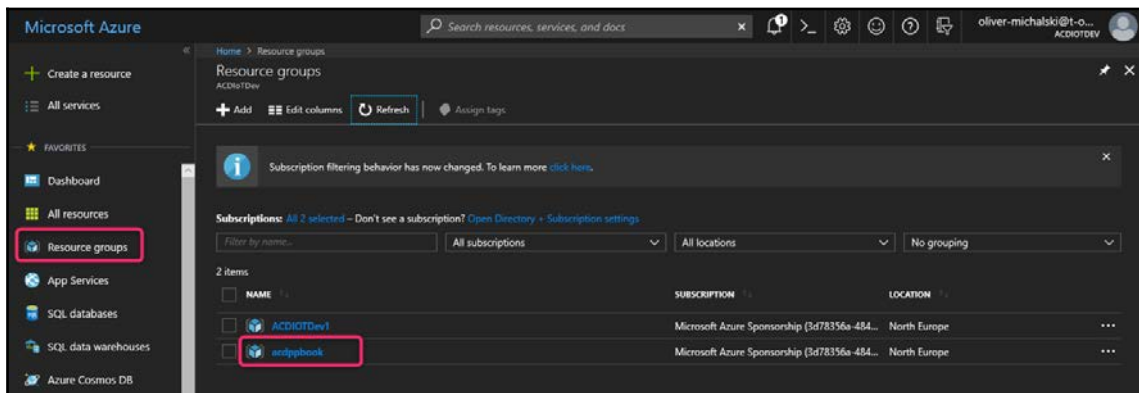
Now you know how to organize your resources, but for working with those resources, there is still another functionality that is important, which I will introduce now.

Azure resource locks

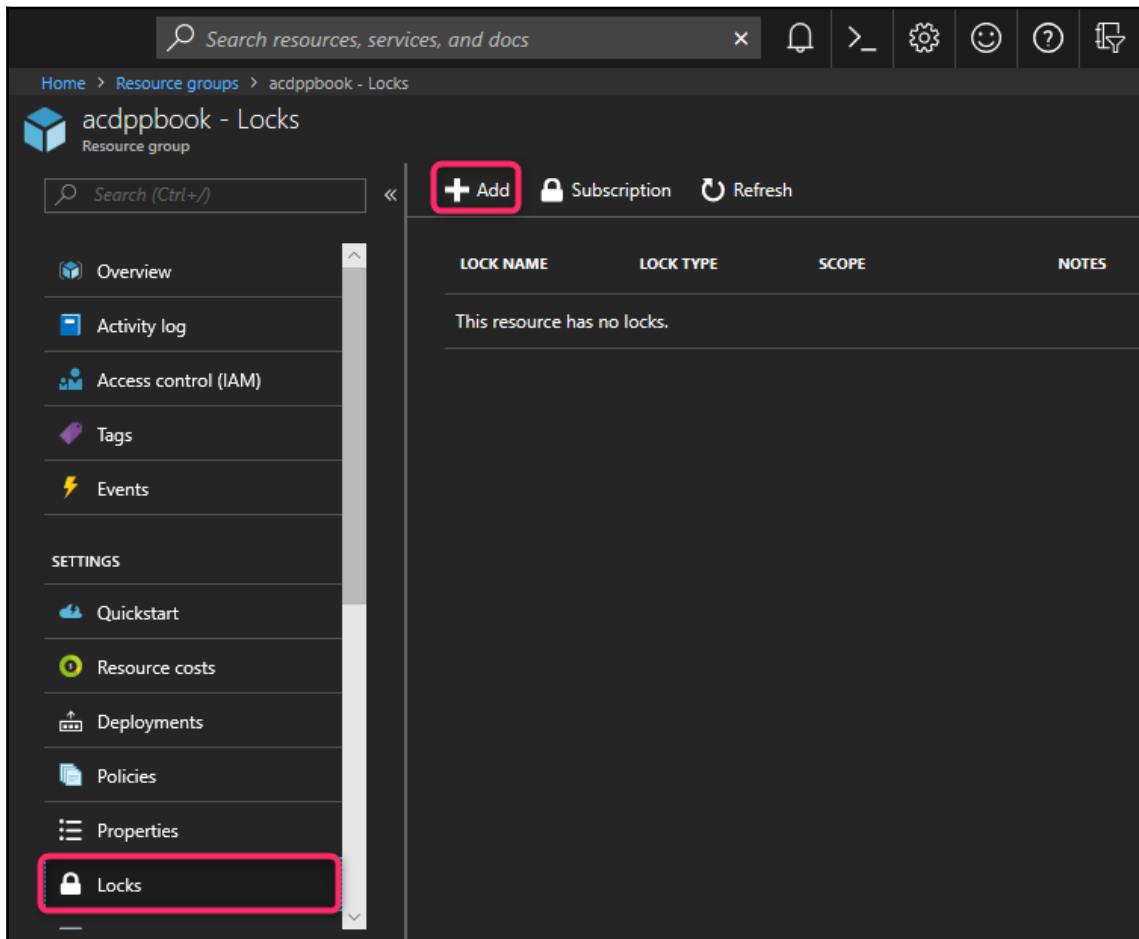
What does this mean? As an administrator, you may need to lock a resource group or resource to prevent other users from accidentally deleting or modifying critical resources. ARM offers a mechanism with two levels (`CanNotDelete` or `ReadOnly`) to be able to make appropriate settings.

Let's take a look at this:

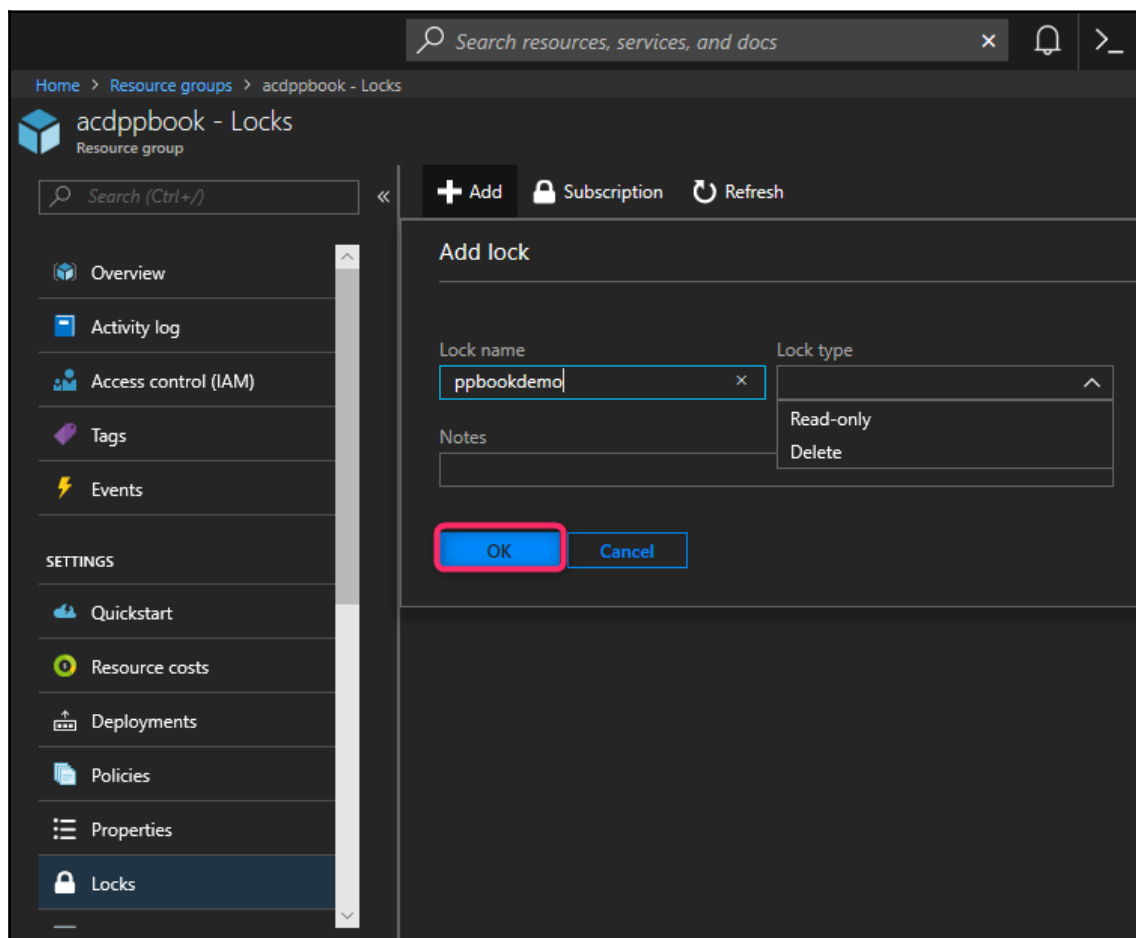
1. In the portal, click on **Resource groups**, and then click on the **Resource groups** blade, then the `acdppbook` name, as shown in the following screenshot:



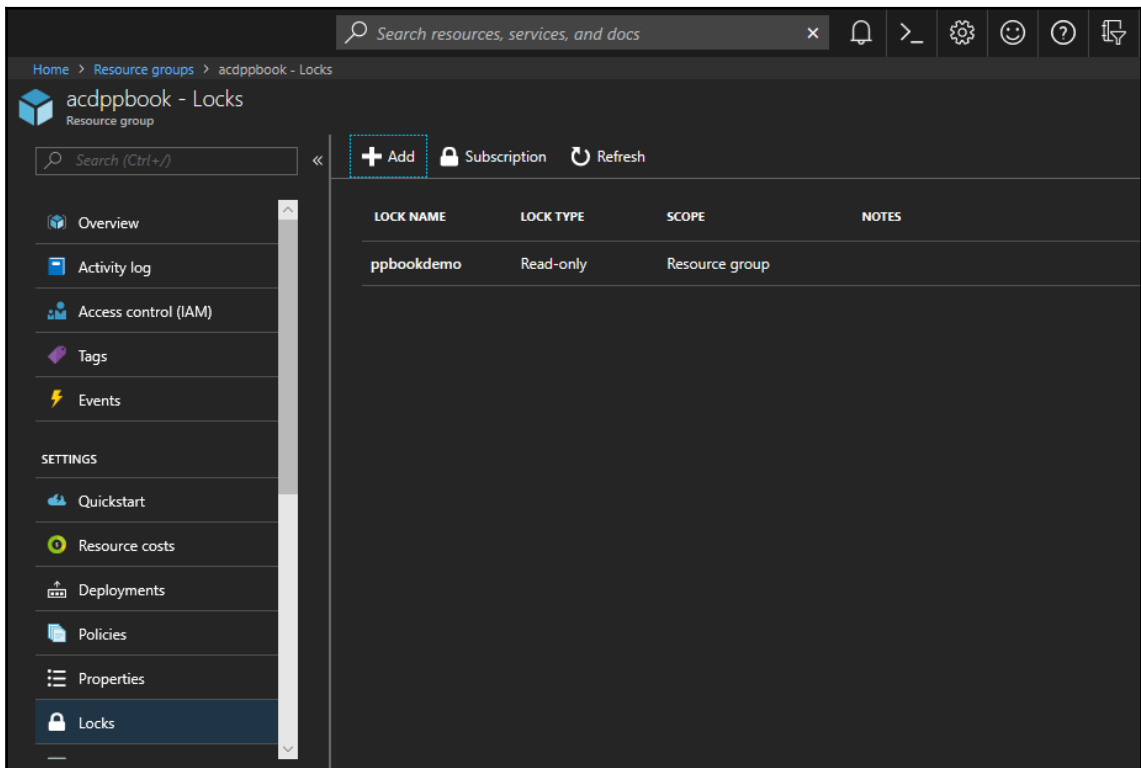
2. In the navigation section on the **Resource groups** dashboard, click on the **Locks** button, then click on the **Locks** blade, followed by clicking on the **Add** button:



- Now, type `ppbookdemo` in the **Lock name** field, select a lock type, and click on the **OK** button:

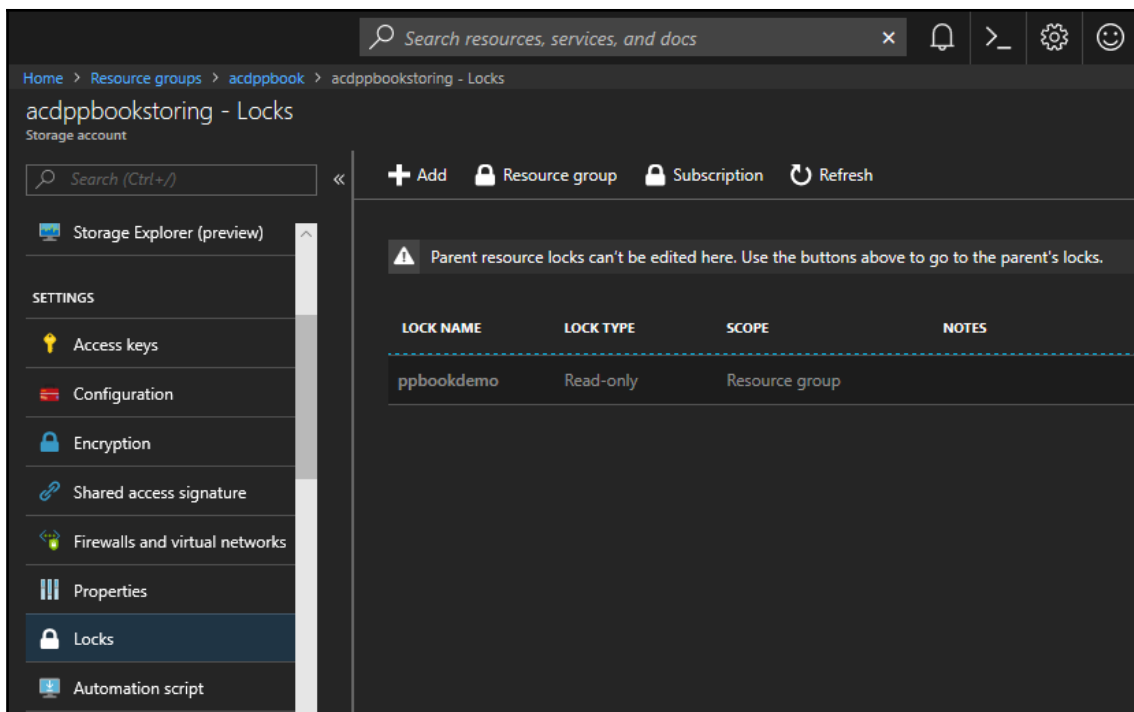


4. Your first lock is ready, as shown in the following screenshot:



Attention! In my example, I have put **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.

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

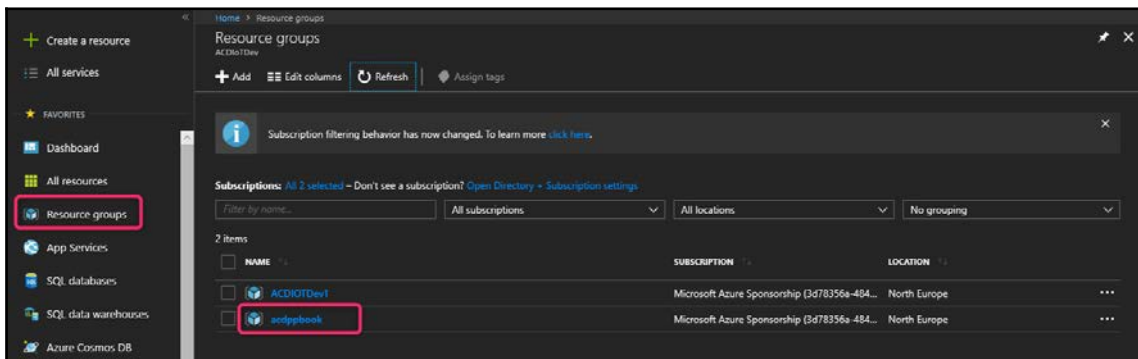
Now, we come to the more advanced section of the explanations on the subject of working with ARM. The content of this section is also divided into two parts:

- In the first part, we consider the issue from the view of an IT professional. IT professionals are less interested in developing their own templates but want to reuse their existing deployments in template form. The corresponding feature in ARM is to export a deployment as an ARM template. This feature allows you to create templates of your deployment and download the templates to secure them in a source code repository, modifying your templates and of course the redeployment.
- The second part (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 often realize demos or customer projects and always roll out the same basic configuration in Azure? If so, the following procedure could make the work easier in future:

1. Open your Azure Management Portal at <https://portal.azure.com>.
2. In the portal, click on **Resource groups**, and then click on the **Resource groups** blade, then the `acdppbook` name, as shown in the following screenshot:

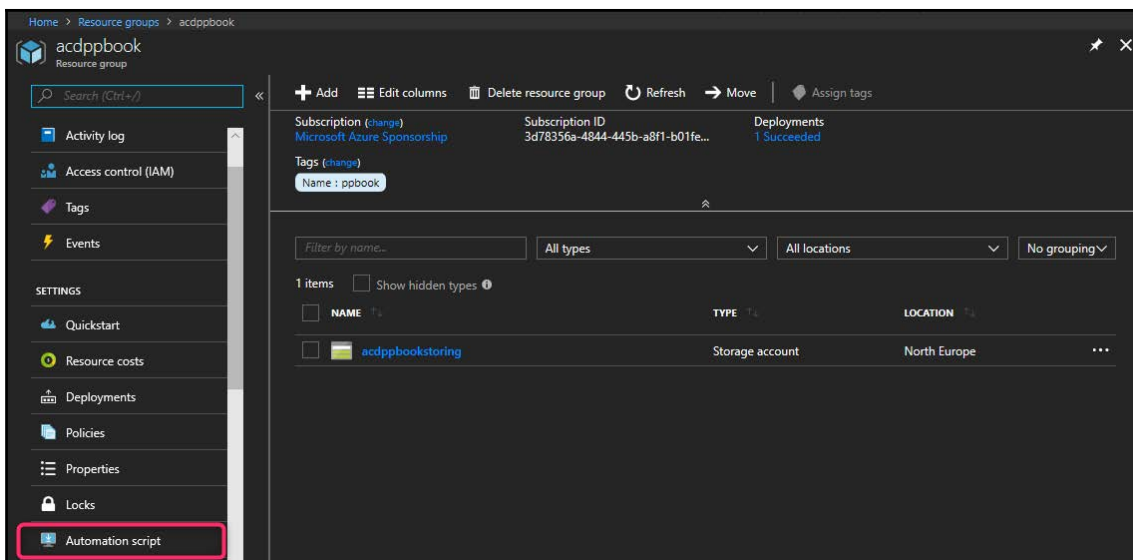


Now, we will export our deployment to an ARM template.

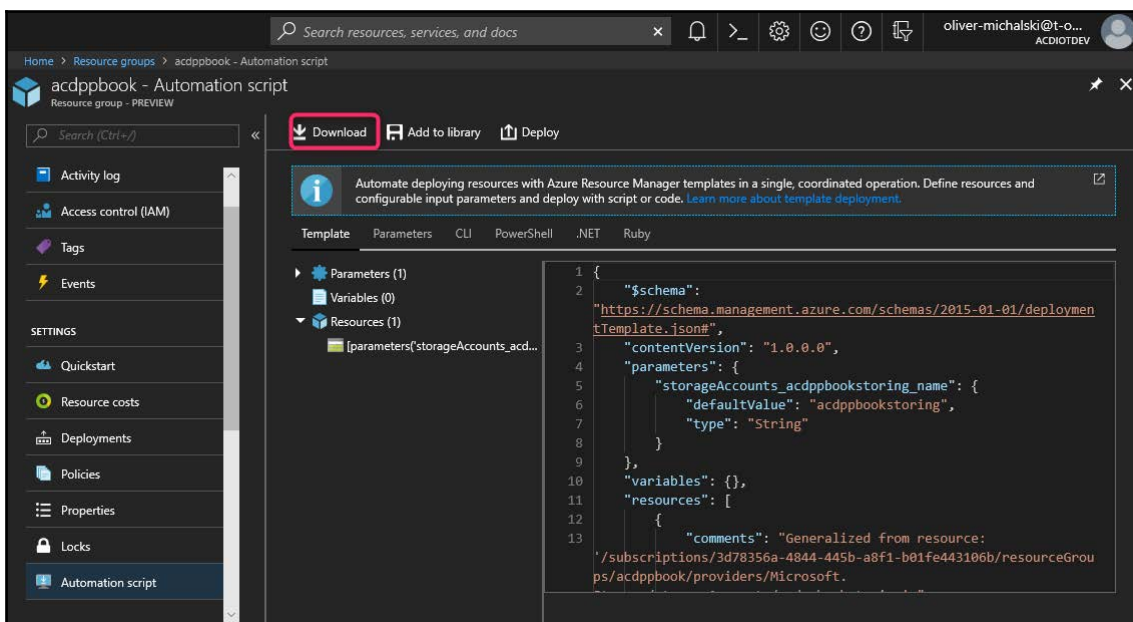
Example 1 – exporting a resource group to an ARM template

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:

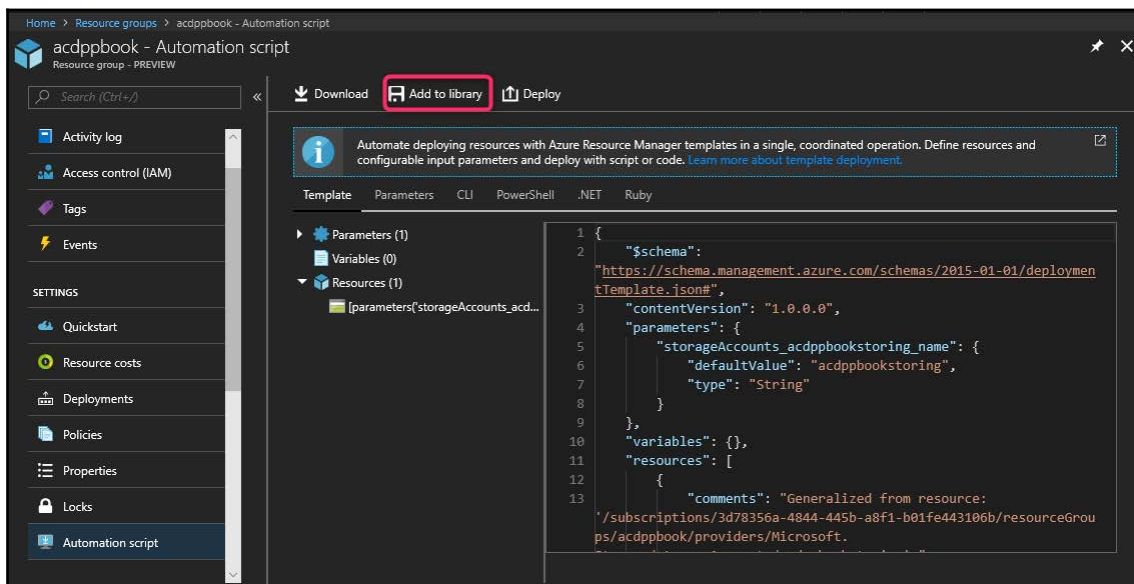
1. In the navigation section of the **Resource groups** dashboard, click on **Automation script**, as shown in the following screenshot:



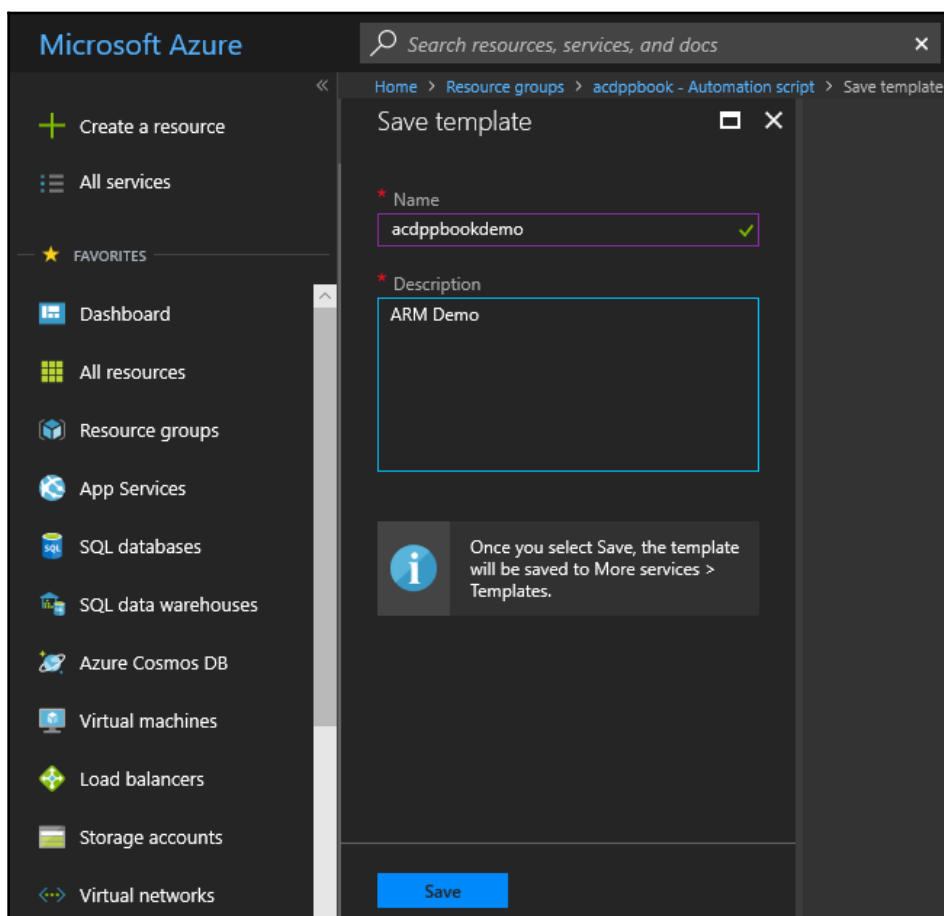
- Now you can see the resource group template. Click on the **Download** button to save the template on your local site and to finalize your work:



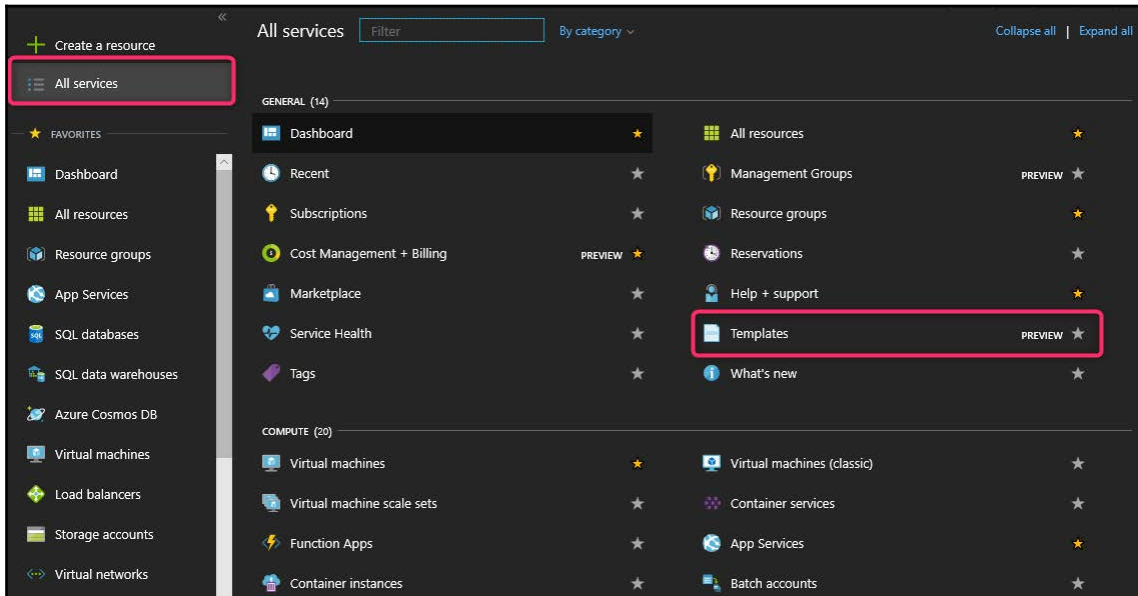
- Alternatively, you also have the option to save your template directly on the Azure platform. Press the **Add to library** button for that:



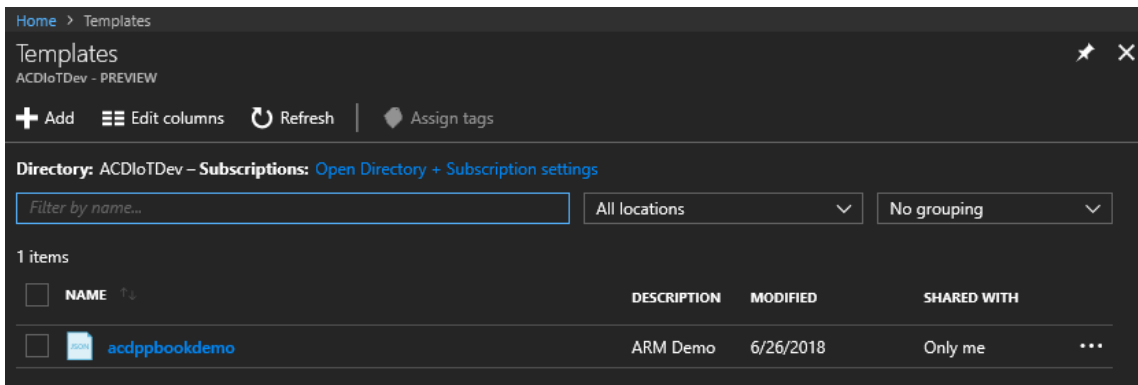
4. Now, open the **Save template** blade. Here, you must type in a value for the **Name** and the **Description** of the template, then click the **Save** button:



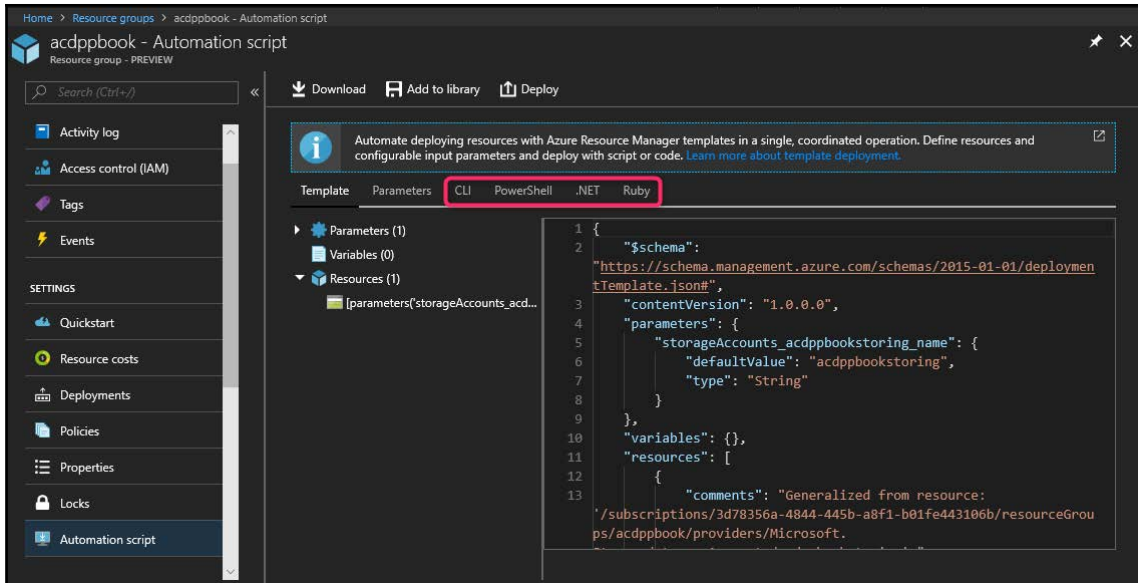
5. If you want to find your template within the Azure platform, click the **All services** list and then the **Templates** option:



6. Open the **Templates** blade with a list of all available templates:



7. Let's go back to the **Automation script** blade. 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:

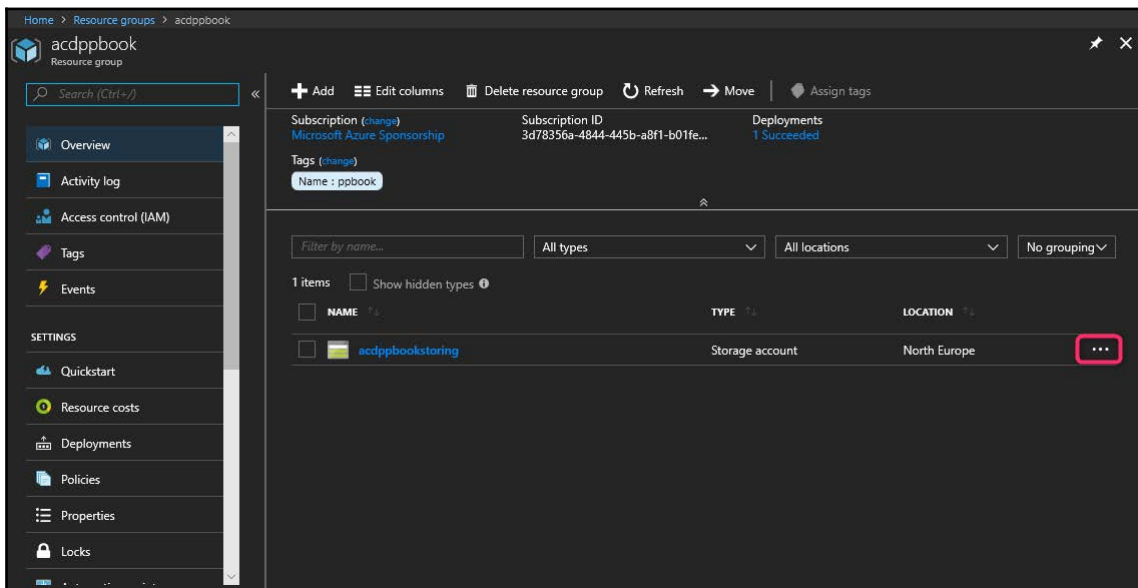


Not all resource types support the export template function. If your resource group only contains a storage account, a virtual machine, or a virtual network, you will not see an error. However, if you have created other resource types, you may see an error stating that there is a problem with the export.

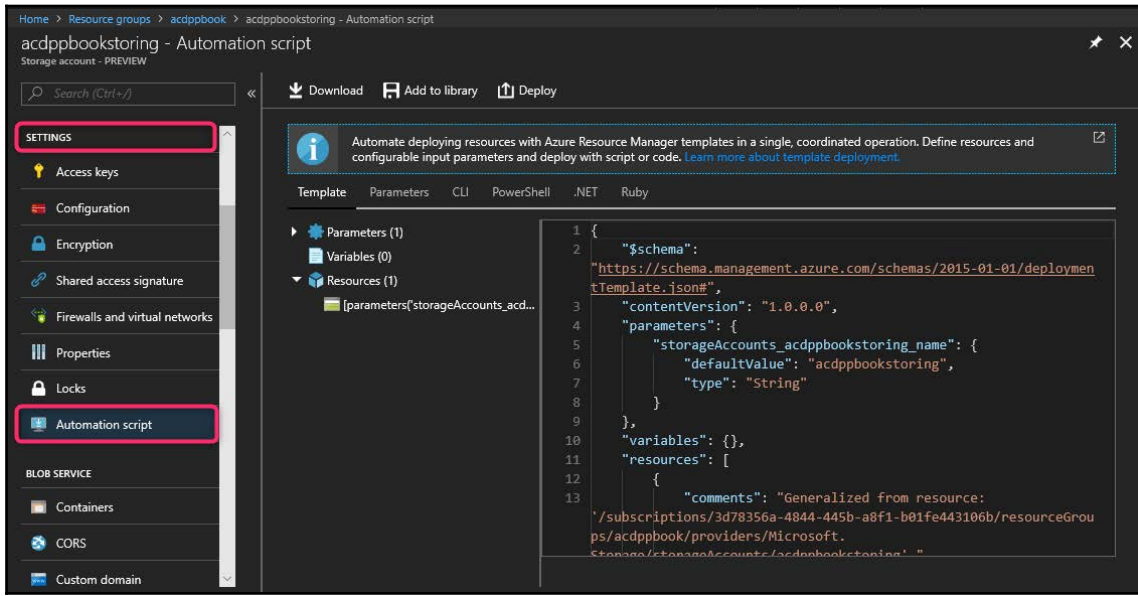
Example 2 – exporting a resource (classic) to an ARM template

In the second example, we want to export a single resource (app, database, and so on), in the so-called classical way. This means that only the most recent version of the resource is exportable. To export a resource in this way, perform the following steps:

1. Go back to the **Resource groups** dashboard for `acdppbook`. In the resource grid, select the `acdppbookstoring` row, and then click the ... field, as shown in the following screenshot:



2. In the **Resource groups** dashboard, click on **Automation script** in the **SETTINGS** area. Now, you can see your template:

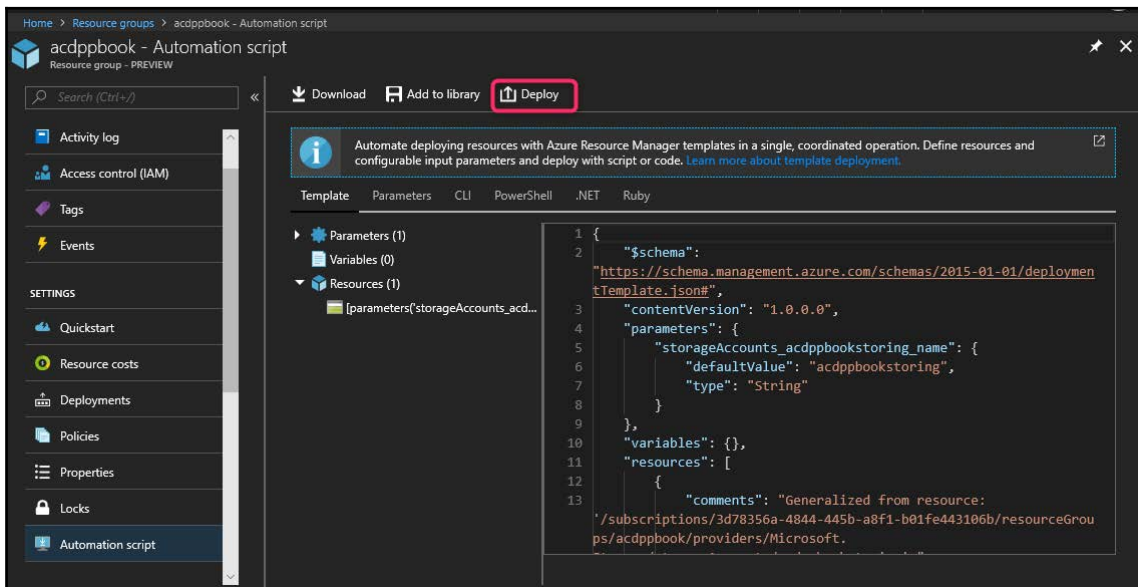


3. Click on **Download** or **Add to library** to save your template, and to finalize your work.

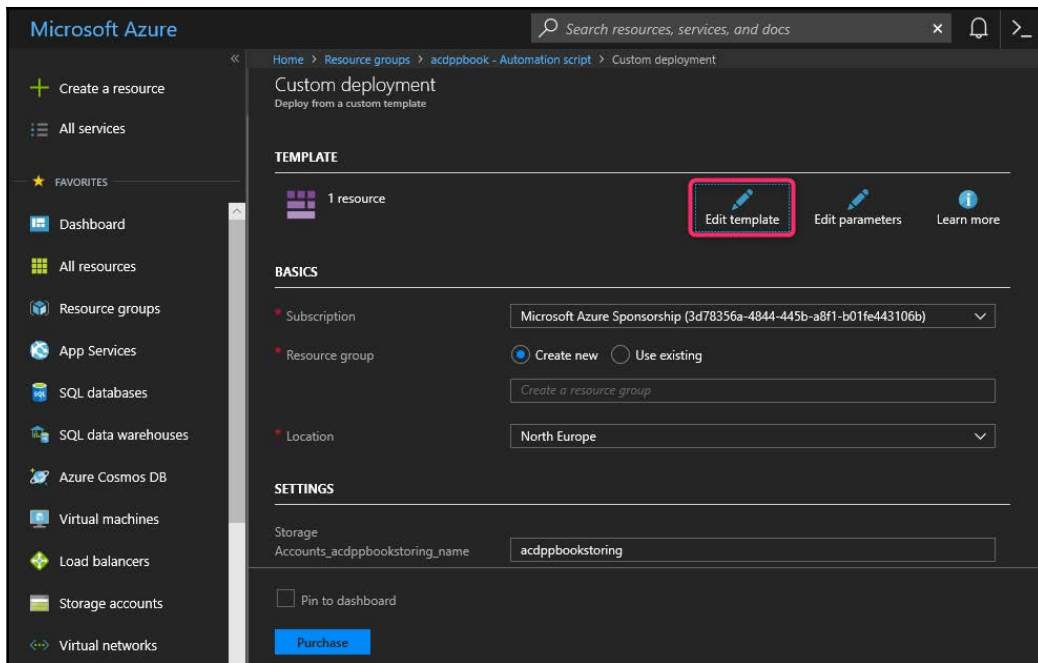
Modifying an ARM template

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

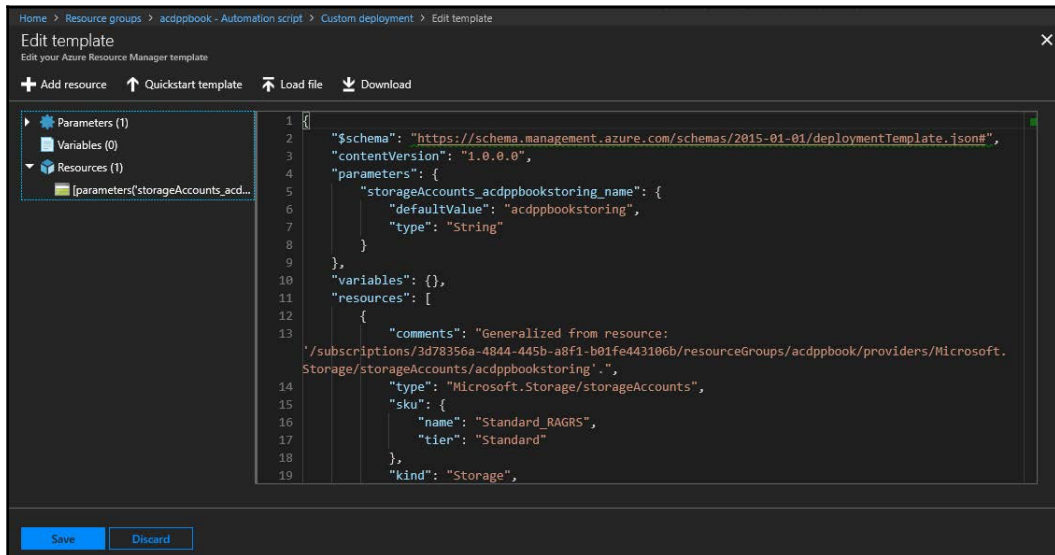
1. Go back to the **Automation script** view. Click on **Deploy**, as shown in the following screenshot:



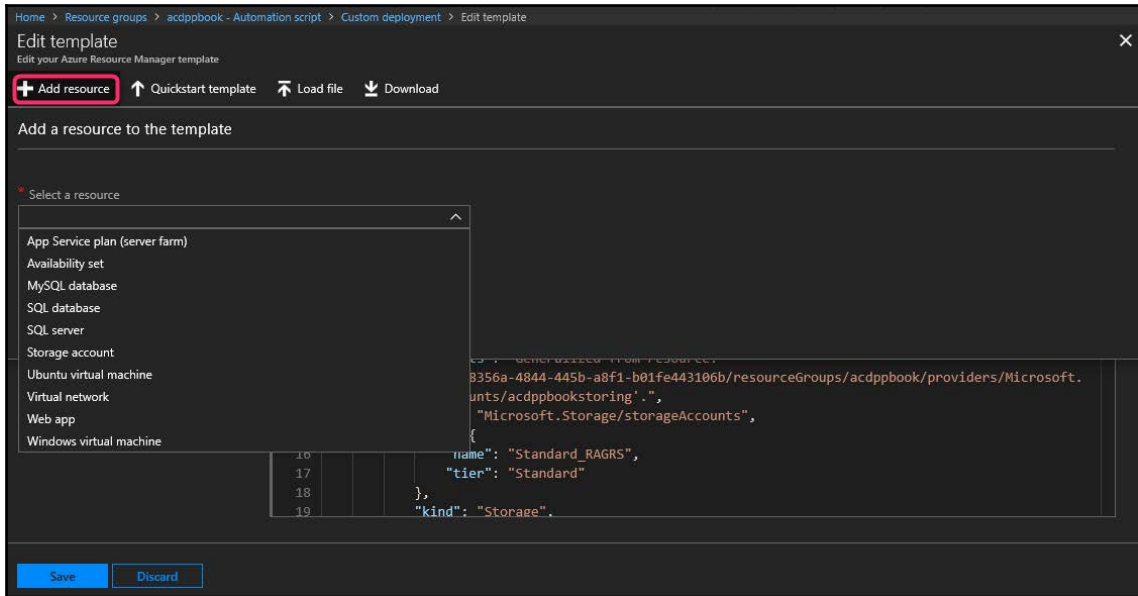
2. Next, the **Custom deployment** blade opens. Note that this is the same environment as used for the Azure Marketplace's template deployment offer, but this time, there is one resource available. To start the working process, you must press the **Edit template** button:



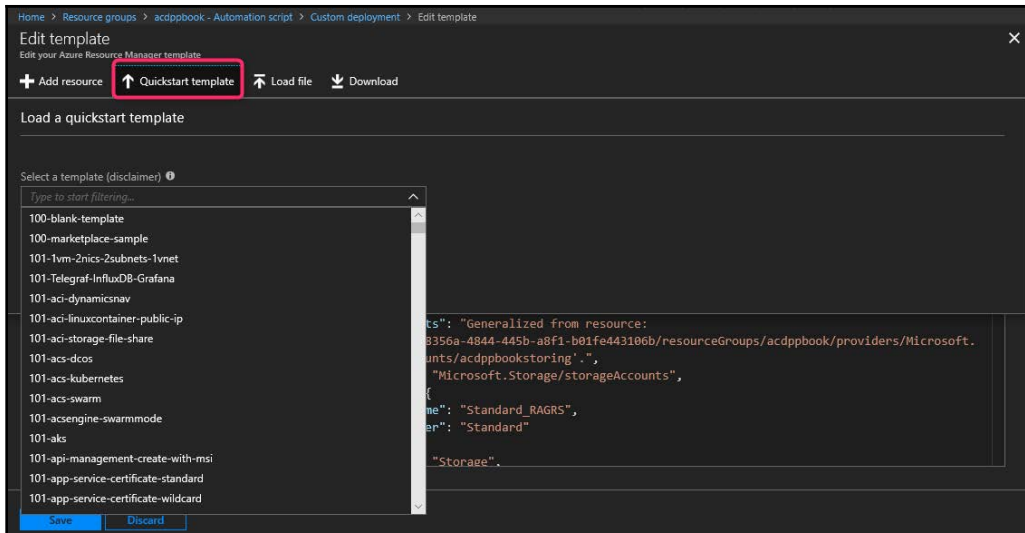
3. The next window is the **Edit template** view. Here, you can edit your template directly in the text:



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

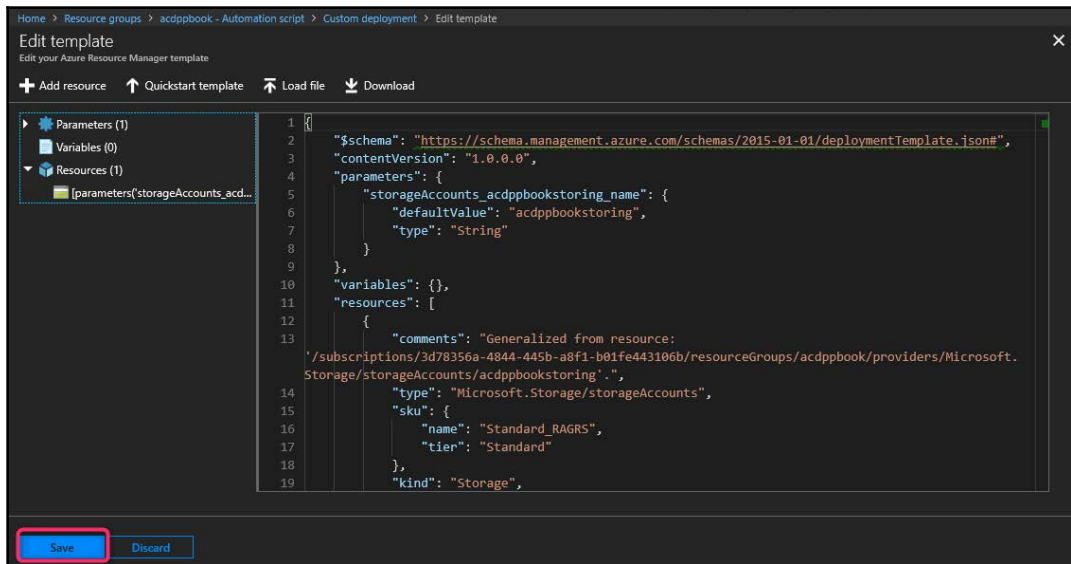


5. You can also take a **Quickstart template** as a reference for your work. Click on the **Quickstart template** button and then select the desired template from the list:



Azure Quickstart templates are a collection of ARM community templates (with solutions for many workloads), and you can find them at <https://github.com/Azure/azure-quickstart-templates>.

6. When everything is OK, press the **Save** button:



7. 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, as shown in the following screenshot:

The screenshot shows the Azure Custom deployment blade. The breadcrumb navigation at the top is: Home > Resource groups > acdppbook - Automation script > Custom deployment. The main heading is "Custom deployment" with the subtitle "Deploy from a custom template".

TEMPLATE

1 resource

[Edit template](#) [Edit parameters](#) [Learn more](#)

BASICS

* Subscription: Microsoft Azure Sponsorship (dropdown arrow)

* Resource group: ☒ Create new ☐ Use existing
Create a resource group (text input)

* Location: North Europe (dropdown arrow)

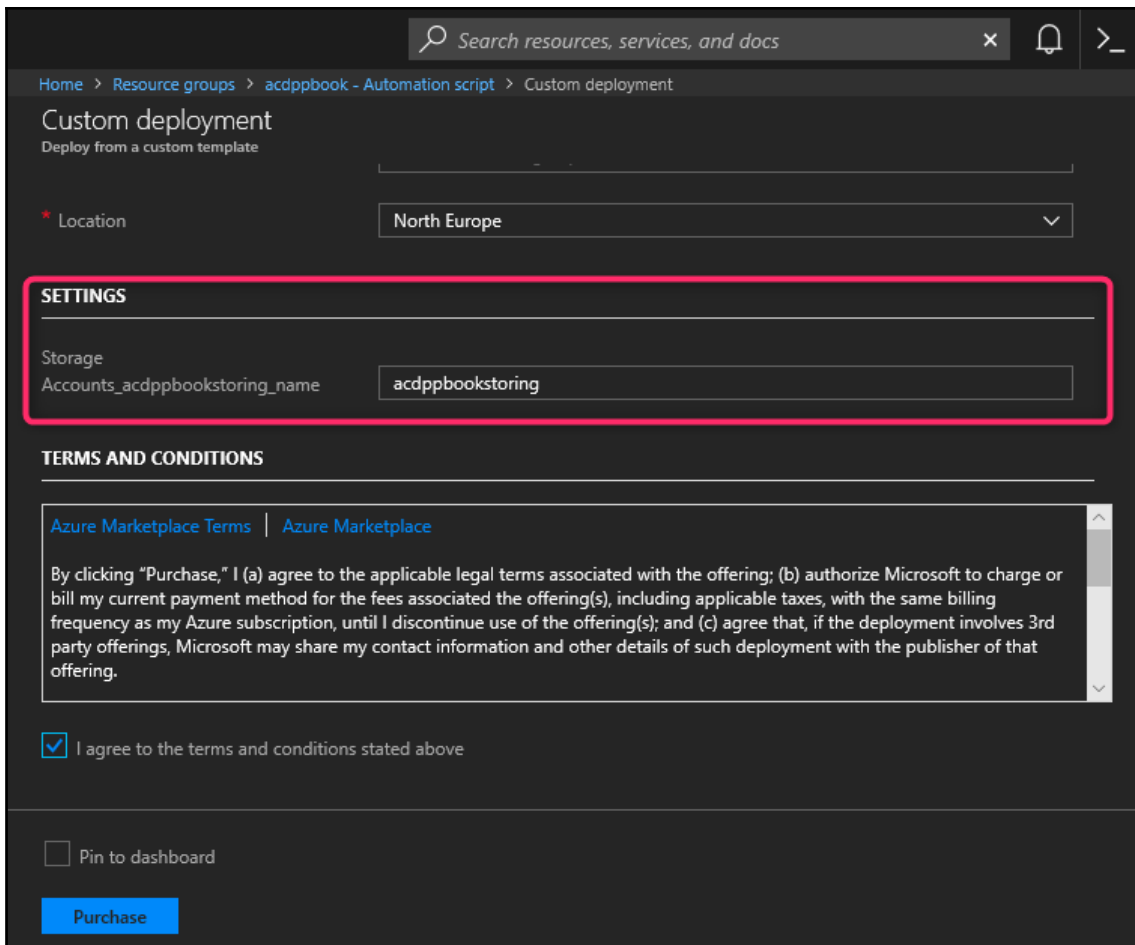
SETTINGS

Storage
Accounts_acdppbookstoring_name: acdppbookstoring

☐ Pin to dashboard

[Purchase](#)

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



The screenshot shows the Azure portal interface for a custom deployment. At the top, there is a search bar and navigation links. The breadcrumb trail is: Home > Resource groups > acdppbook - Automation script > Custom deployment. The main heading is 'Custom deployment' with the subtitle 'Deploy from a custom template'. Below this, there is a 'Location' dropdown menu set to 'North Europe'. The 'SETTINGS' section is highlighted with a red box and contains a 'Storage' field with the label 'Accounts_acdppbookstoring_name' and the value 'acdppbookstoring'. Below the settings, there is a 'TERMS AND CONDITIONS' section with a scrollable text area containing the legal terms. At the bottom, there is a checkbox for 'I agree to the terms and conditions stated above' which is checked, a 'Pin to dashboard' checkbox which is unchecked, and a blue 'Purchase' button.

Search resources, services, and docs

Home > Resource groups > acdppbook - Automation script > Custom deployment

Custom deployment

Deploy from a custom template

* Location: North Europe

SETTINGS

Storage
Accounts_acdppbookstoring_name: acdppbookstoring

TERMS AND CONDITIONS

[Azure Marketplace Terms](#) | [Azure Marketplace](#)

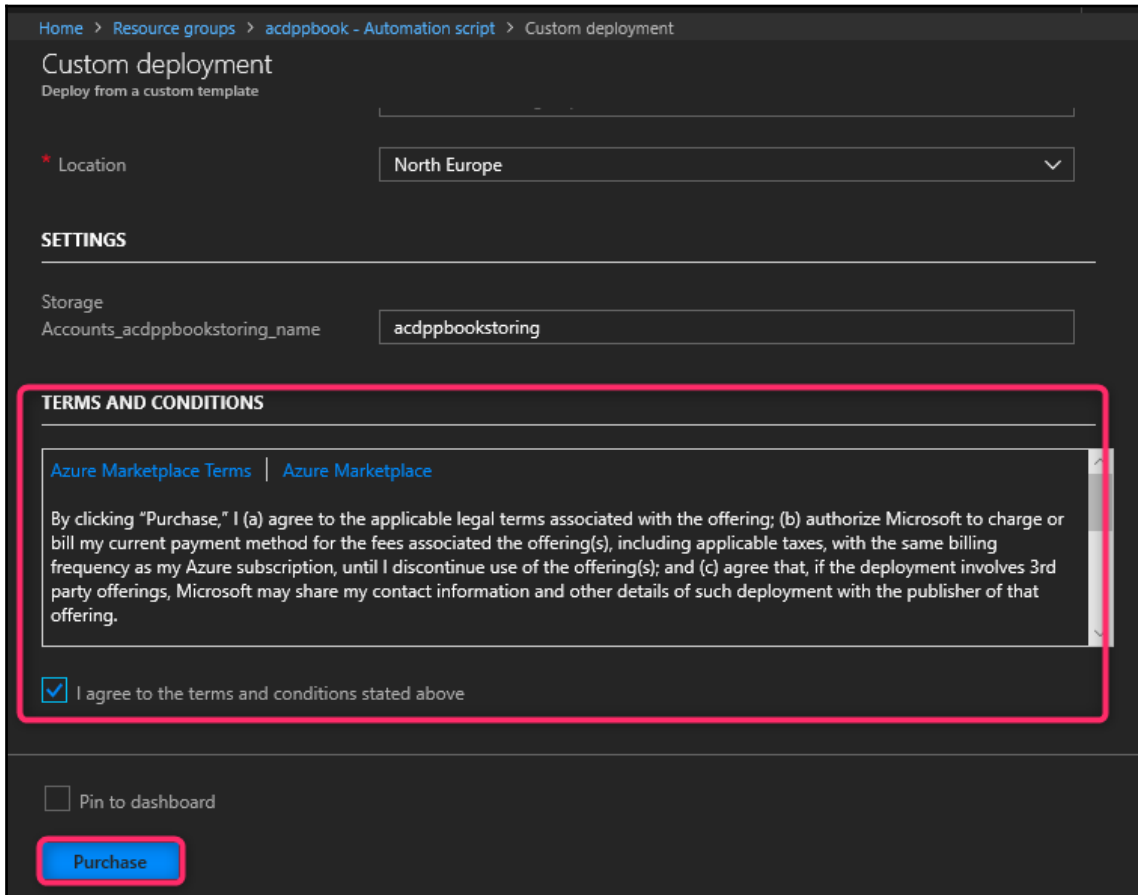
By clicking "Purchase," I (a) agree to the applicable legal terms associated with the offering; (b) authorize Microsoft to charge or bill my current payment method for the fees associated the offering(s), including applicable taxes, with the same billing frequency as my Azure subscription, until I discontinue use of the offering(s); and (c) agree that, if the deployment involves 3rd party offerings, Microsoft may share my contact information and other details of such deployment with the publisher of that offering.

☒ I agree to the terms and conditions stated above

☐ Pin to dashboard

Purchase

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



The screenshot shows the Azure Custom deployment page. The breadcrumb navigation at the top reads: Home > Resource groups > acdppbook - Automation script > Custom deployment. The main heading is "Custom deployment" with the subtitle "Deploy from a custom template". Below this, there is a "Location" dropdown menu set to "North Europe". A "SETTINGS" section follows, with a "Storage" field labeled "Accounts_acdppbookstoring_name" containing the value "acdppbookstoring". The "TERMS AND CONDITIONS" section is highlighted with a red box. It contains links for "Azure Marketplace Terms" and "Azure Marketplace". The text states: "By clicking 'Purchase,' I (a) agree to the applicable legal terms associated with the offering; (b) authorize Microsoft to charge or bill my current payment method for the fees associated the offering(s), including applicable taxes, with the same billing frequency as my Azure subscription, until I discontinue use of the offering(s); and (c) agree that, if the deployment involves 3rd party offerings, Microsoft may share my contact information and other details of such deployment with the publisher of that offering." Below this text is a checked checkbox with the label "I agree to the terms and conditions stated above". At the bottom of the page, there is a "Pin to dashboard" checkbox and a "Purchase" button, which is also highlighted with a red box.



Pressing the **Purchase** button is not really a purchase.

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). I recommend you still use Visual Studio (with an excellent JSON editor).

To prevent unnecessary expense, the **Visual Studio Community Edition** or **Visual Studio Code** is completely sufficient for our purposes.

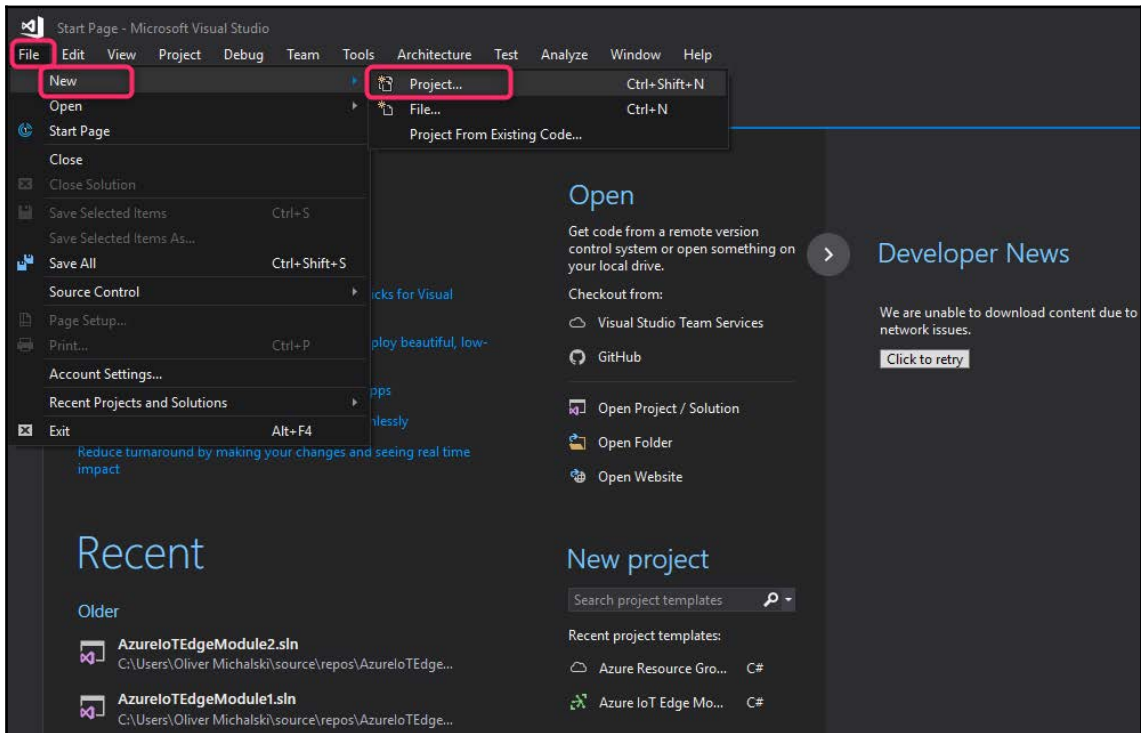


Formerly, another alternative was still available—**Azure Resource Manager Template Visualizer (ArmViz)**. With ArmViz, you can create templates with graphical means or in an editor. However, ArmViz has a weakness—it's an open source project and only available outside the platform. *Formerly?* Unfortunately, yes—the team is currently working on version 2.0 of the ArmViz tool, which does not work 100%. If you still want to try out ArmViz, you will find the tool at <http://armviz.io>.

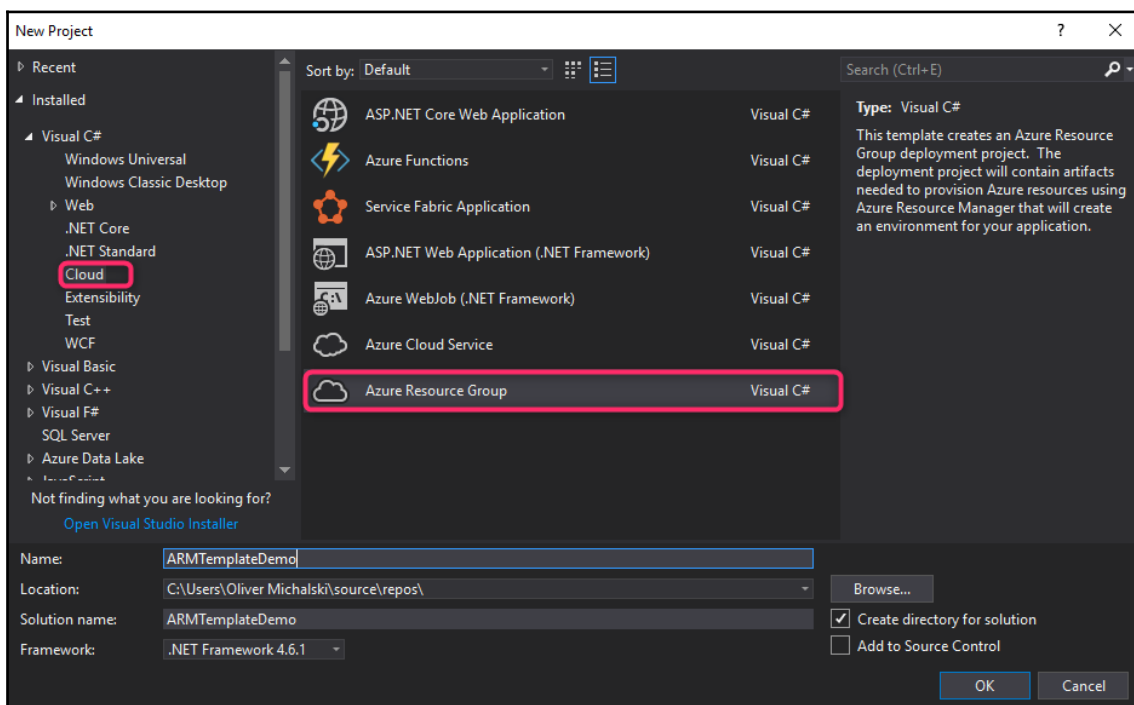
Creating your own ARM template (for developers)

To create an ARM template, perform the following steps:

1. Open your Visual Studio. First, click on the **File** button, then click on **New** and the **Project...** link, as shown in the following screenshot:



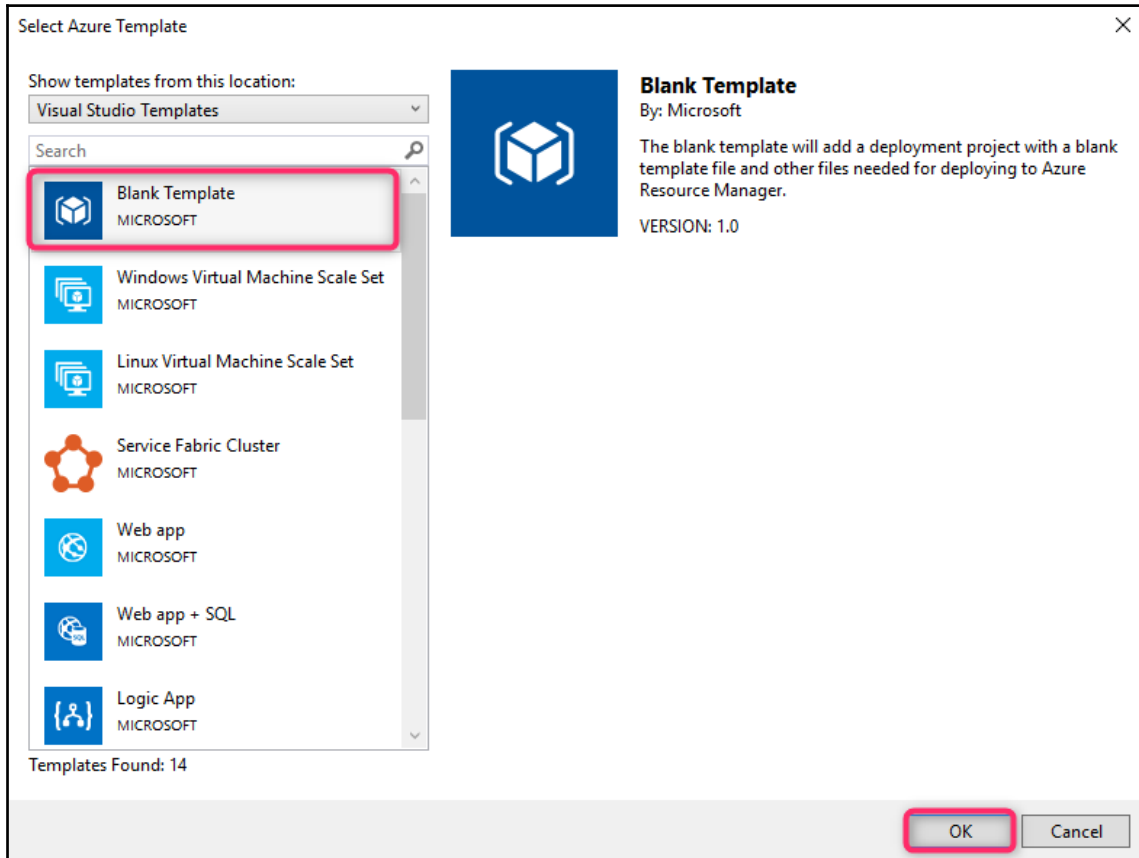
2. The selection dialog opens with the available project templates. The required template, **Azure Resource Group**, can be found in the **Cloud** area:



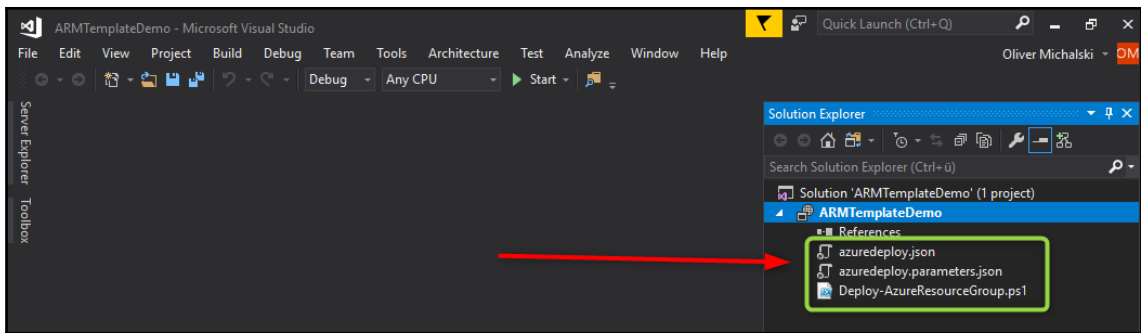
If you use an older version of Visual Studio and do not find the entry there, you have probably forgotten to install the Azure SDK and Azure VS tooling.

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

- Now, another selection dialog opens, this time with a list of available Azure templates. For our demo, we need **Blank Template**. Select the entry and press the **OK** button:



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



The project consists of the following three artifacts:

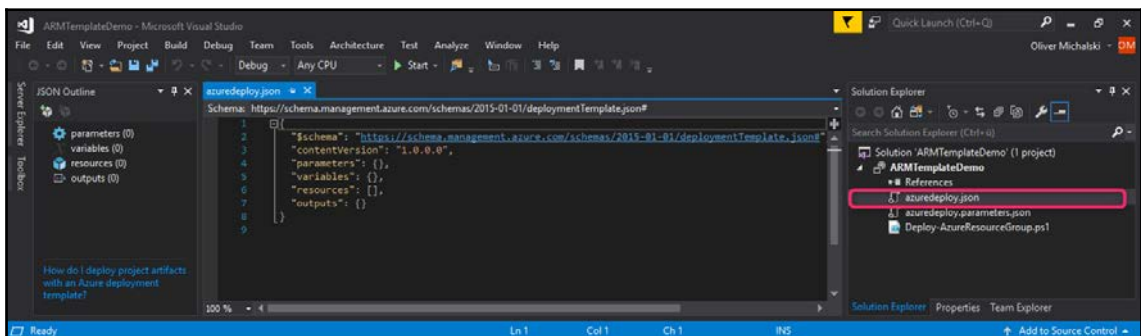
- `azuredeploy.json`: This is the template for your own ARM template
- `azuredeploy.parameters.json`: This is used as a store for all the required parameter values
- `Deploy-AzureResourceGroup.ps1`: This is a PowerShell script that will help you to perform the final deployment



You can change the `azuredeploy` name in any way you like, but the extensions `.json` and `.parameters.json` extensions must be maintained.

Let's have a deeper look:

I would like to start with the `azuredeploy.json` file. Please click on the corresponding entry in the Solution Explorer. Now, you should see the following screen:



In the middle area of the Visual Studio IDE (the editor window) the following code block is now given:

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

This code is the simplest form of an ARM template, but I 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
\$schema	Yes
contentVersion	Yes
parameters	No
variables	No
resources	Yes
output	No

As you can see, the `$schema`, `contentVersion`, and `resources` fields need to be filled in. For the `$schema` and `contentVersion` fields, 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, I 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 I need to point 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 our `azuredeploy.json` sample file looks like this:

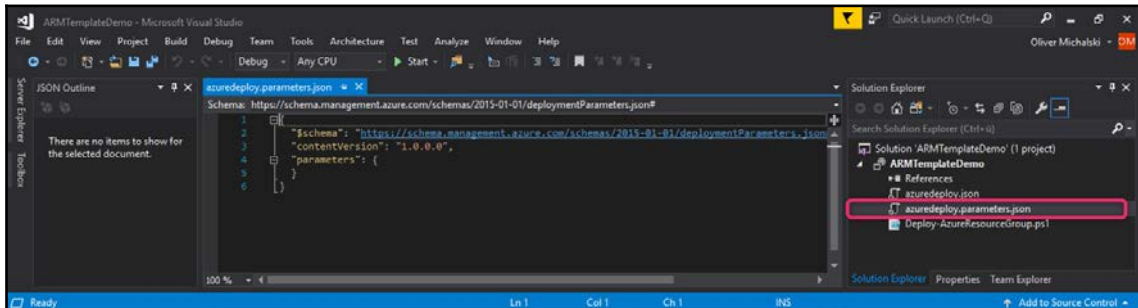
```
{
  "$schema":
    "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.
    json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageAccountName": {
      "type": "string",
      "metadata": {
        "description": "Storage Account Name"
      }
    }
  },
  "resources":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

```

    "accountType": "Standard_LRS"
  }
}
]
}

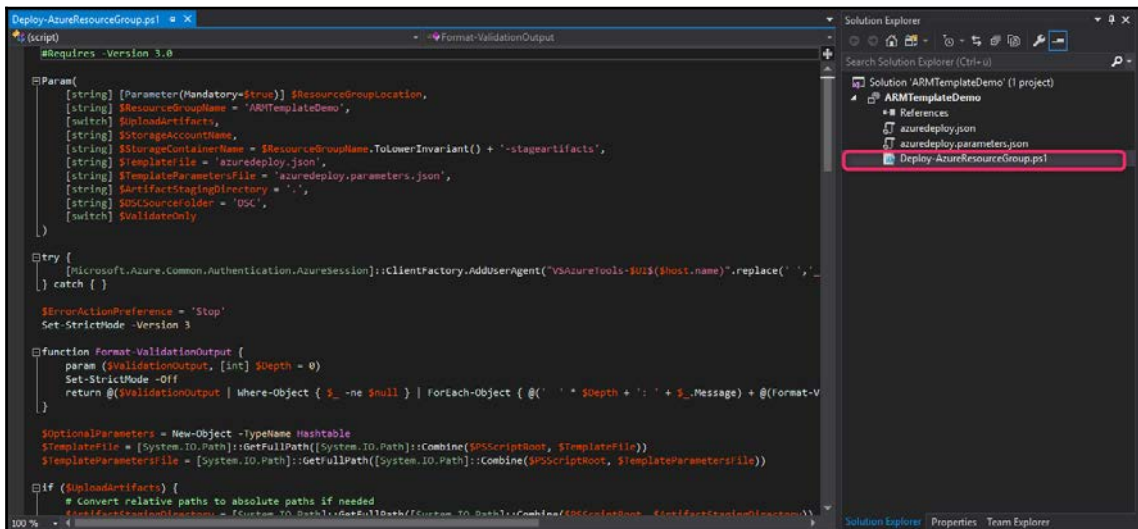
```

The next step in our tour is the `azuredeploy.parameters.json` file. Please click on the corresponding entry in the Solution Explorer. Now, you should see the following screen:



Parameters are automatically adopted during the entire processing of the `azuredeploy.json` file, but can also be inserted manually (for example, to define default values).

The last step in our tour is the `Deploy.AzureResourceGroup.ps1` file. Please click on the corresponding entry in the Solution Explorer. Now, you should see the following screen:



Again, all settings during the entire processing of the `azuredeploy.json` file are automatically taken over and can be manually inserted.

Summary

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

In the first part of this chapter, we saw how to create 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.

The second part of this chapter followed detailed information on ARM templates, divided into two areas for developers and IT professionals.

In the next chapter, we'll use the knowledge we've learned and look at the latest concepts for implementations (**Azure Managed Applications** and **Azure Service Catalog**).

Questions

1. What language are ARM templates in?
2. How does ARM deploy templates?
3. What is the use case for Azure Resource locks?
4. Is there a use case that you cannot fulfill using ARM templates?
5. What is the process to upload custom ARM templates to Azure and deploy them?
6. What are the tools you create ARM templates with?
7. Is ARM available per region or globally?
8. Is there a way to use ARM templates on other clouds? How do you do that?

Further reading

Read the following articles for more information:

- **Azure Resource Manager templates:** https://www.packtpub.com/mapt/book/networking_and_servers/9781786468550/3/ch03lv11sec20/azure-resource-manager-templates
- **Azure MasterClass: Manage Azure Cloud with ARM Templates [Video]:** <https://www.packtpub.com/application-development/azure-masterclass-manage-azure-cloud-arm-templates-video>
- **Azure Resource Manager documentation:** <https://docs.microsoft.com/en-us/azure/azure-resource-manager/>

3

Deploying and Synchronizing Azure Active Directory

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

The Microsoft central instance for identity management for all Microsoft cloud services is Azure **Active Directory (AD)**. Every subscription, contract, security policy, or identity Microsoft provides for their cloud services is based on Azure AD, which is also called Microsoft Tenant.

In this chapter, you will learn the basics of Azure AD, and how you implement Azure AD and hybrid Azure AD when connecting to **Active Directory Domain Services (AD DS)**.

We are going to explore the following topics:

- Azure AD overview
- Azure AD subscription options
- Azure AD deployment
- Azure AD user and subscription management
- How to deploy Azure AD hybrid identities with AD DS
- Azure AD hybrid high availability and non-high availability deployments

Azure AD

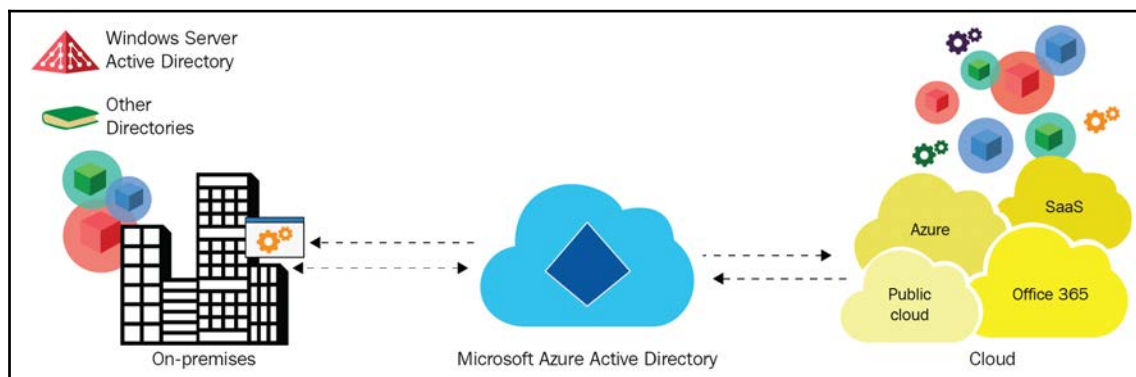
Azure AD is a multi-tenant cloud directory and identity management service developed by Microsoft. Azure AD also includes a full suite of identity management capabilities, including the following:

- 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 AD, 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 (SSO)** and **multi-factor authentication (MFA)** through Azure AD without adding third-party software into its environment.

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

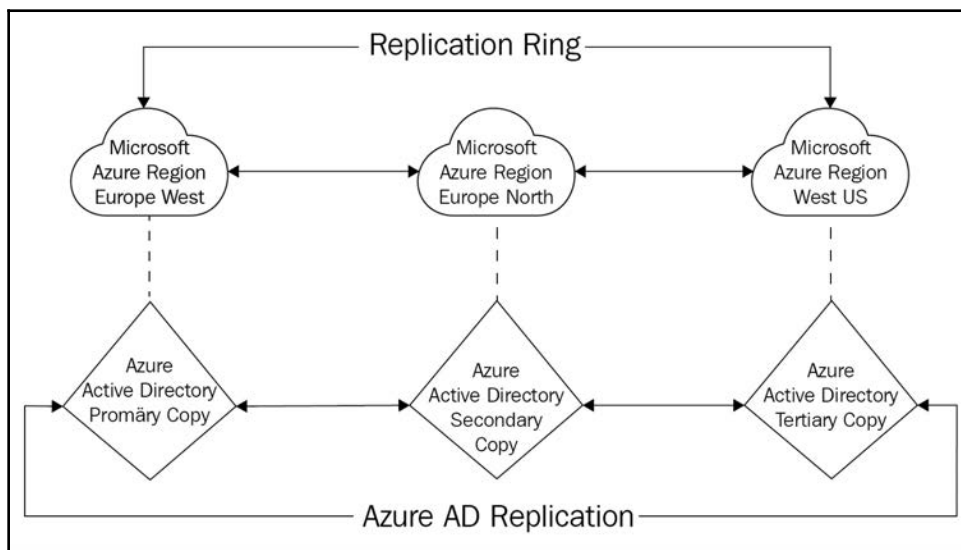
The following diagram describes the general structure of Azure AD in a hybrid deployment with AD DS:



Customers using different Microsoft services, such as Office 365, CRM Online, or Intune, are already using Azure AD for their service. You can easily identify whether you use Azure AD if you have a username such as `user@domain.onmicrosoft.com`. Other top-level-domains, such as `.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 datacenter around the world. Microsoft has implemented automated failover with a minimum of two copies of your Azure directory service in other regional or global datacenters.

Your directory is running in your primary datacenter, but is regularly replicated into another two in your region. If you only have two Azure datacenters in your region, as in Europe, a copy will be distributed to another datacenter in another region:



In regular cases, Microsoft prefers to synchronize Azure AD only within a geopolitical region, such as the European Union or United States. In some cases, where no third region is available, Microsoft replicates a copy outside the geopolitical region. For Europe, that was the case until regions in France became available. Since France has been online and generally available, the replication of Azure AD only runs in regions within the EU.

Azure AD options

There are currently four electable options for Azure AD with different features to use.

Azure AD free

Azure AD free supports common features such as these:

- Up to 5,00,000 directory objects, which could be users, devices, applications, or groups
- User/group management (add/update/delete), user-based provisioning, and device registration
- SSO for up to ten applications per user
- Self-service password changes for cloud users
- Connect and sync with on-premises AD DS
- 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—free and additional group-based access management/provisioning
- Self-service password reset for cloud users
- Company branding (logon pages/access panel customization)
- Application proxy
- Service level agreement of 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 resets for cloud users
- Company branding (logon pages/access panel customization)
- Application proxy
- Service level agreement of 99.9%

Premium features also include:

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

Since Q3/2016, Microsoft has also allowed customers to use the Azure AD P2 plan, which includes all the capabilities in Azure AD premium P1 as well as new identity protection and privileged identity management capabilities. This was an important step for Microsoft to extend its offering for Windows 10 and Windows Server 2016/19 device management with Azure AD.

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

Depending on what you plan to do with your Azure environment, you should choose the right Azure AD option.

Deploying a custom Azure AD

To understand how you deploy Azure AD, you need to understand that Azure AD is directly connected to your Azure subscription. So, the Azure account subscription administrator 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 account formerly known as a Live ID or Microsoft account, or an Azure AD account. This can be created, for example, through Office 365 before adding Azure agreements and subscription payments, or could be an account created and synchronized through the AD DS to Azure. What you shouldn't do is to use a personal account for 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 AD is created when you create an Azure subscription or you subscribe to a Microsoft cloud service such as 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://portal.azure.com/#create/Microsoft.AzureActiveDirectory>.



With effect from January 2018, Microsoft only offers one portal, <https://manage.windowsazure.com>, since the old deployment engine based on **Microsoft Service Manager** was discontinued. The modern portal, <https://portal.microsoft.com>, which is based on the Microsoft resource manager engine (ARM), is the primary and preferred portal.

2. With the new portal, creating an Azure AD becomes quite easy:
 - Select a display name
 - Select your tenant/Azure AD name
 - Select the country where you are legally based

- Click on the **Create** button:

The screenshot shows the Microsoft Azure portal interface. On the left is a dark sidebar with navigation options: 'Create a resource', 'All services', 'FAVORITES', 'Dashboard', 'All resources', 'Resource groups', 'App Services', 'Function Apps', 'SQL databases', 'Azure Cosmos DB', 'Virtual machines', 'Load balancers', 'Storage accounts', 'Virtual networks', 'Azure Active Directory' (selected), 'Monitor', 'Advisor', 'Security Center', 'Cost Management + Billing', and 'Help + support'. The main area is titled 'Create directory' and contains the following fields:

- Organization name**: Bookdemo (with a green checkmark)
- Initial domain name**: bookdemopack (with a green checkmark). Below this, the full domain 'bookdemopack.onmicrosoft.com' is displayed.
- Country or region**: United States (selected from a dropdown menu).

At the bottom of the form is a blue 'Create' button. A grey information box states: 'Directory creation will take about one minute.'



Azure AD B2C is a cloud identity management solution for your consumer-facing web and mobile applications. You can find more information in the Azure documentation at <https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-overview>.

3. The process will take about a minute, and at the end you'll be navigated to your newly created tenant.



The regular way an Azure AD is created is a bit different. Normally, the Azure AD is created by your license or subscription seller, for example, COMPAREX, SoftwareONE, or Ingram Micro. They will create the subscription for you when ordering an Azure subscription, an online **Enterprise Agreement (EA)**, or other Microsoft 365 services. EA customers are normally only allowed to have one Azure AD; if you need to have more than one Azure AD, you need to extend your EA with an additional contract. This contract will allow you to use EA subscriptions in a multi-tenant environment.

To make identity management between Microsoft 365 services, Azure, and other identity services consistent, you should use one tenant for all your users and only split between production and the **dev/test** Azure AD and so on.

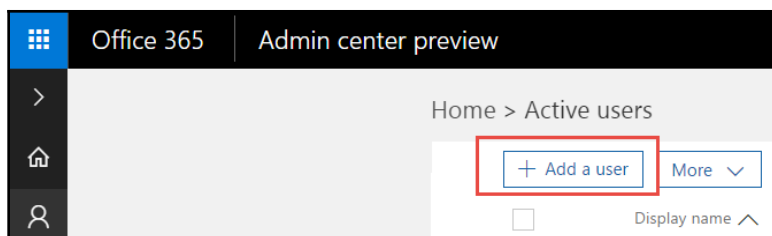
Adding accounts and groups to Azure AD

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

- **Cloud accounts:** Accounts that are created through Azure AD or other Microsoft cloud services, such as Office 365.
- **Hybrid accounts:** Accounts that are created and located in on-premises Microsoft AD DS. Those accounts are deployed through a the Azure AD Connect and synchronization tool.

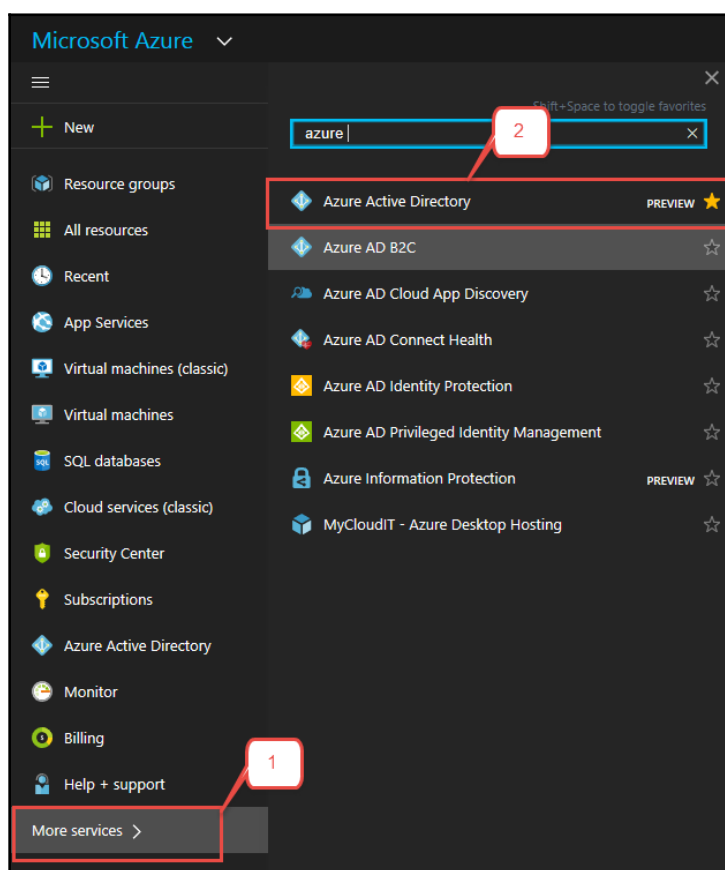
To create cloud accounts, you have several options. Most Azure AD users start with Office 365 and do not natively add users through Azure. If you've used Office 365 before, that would be the simplest for you.

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

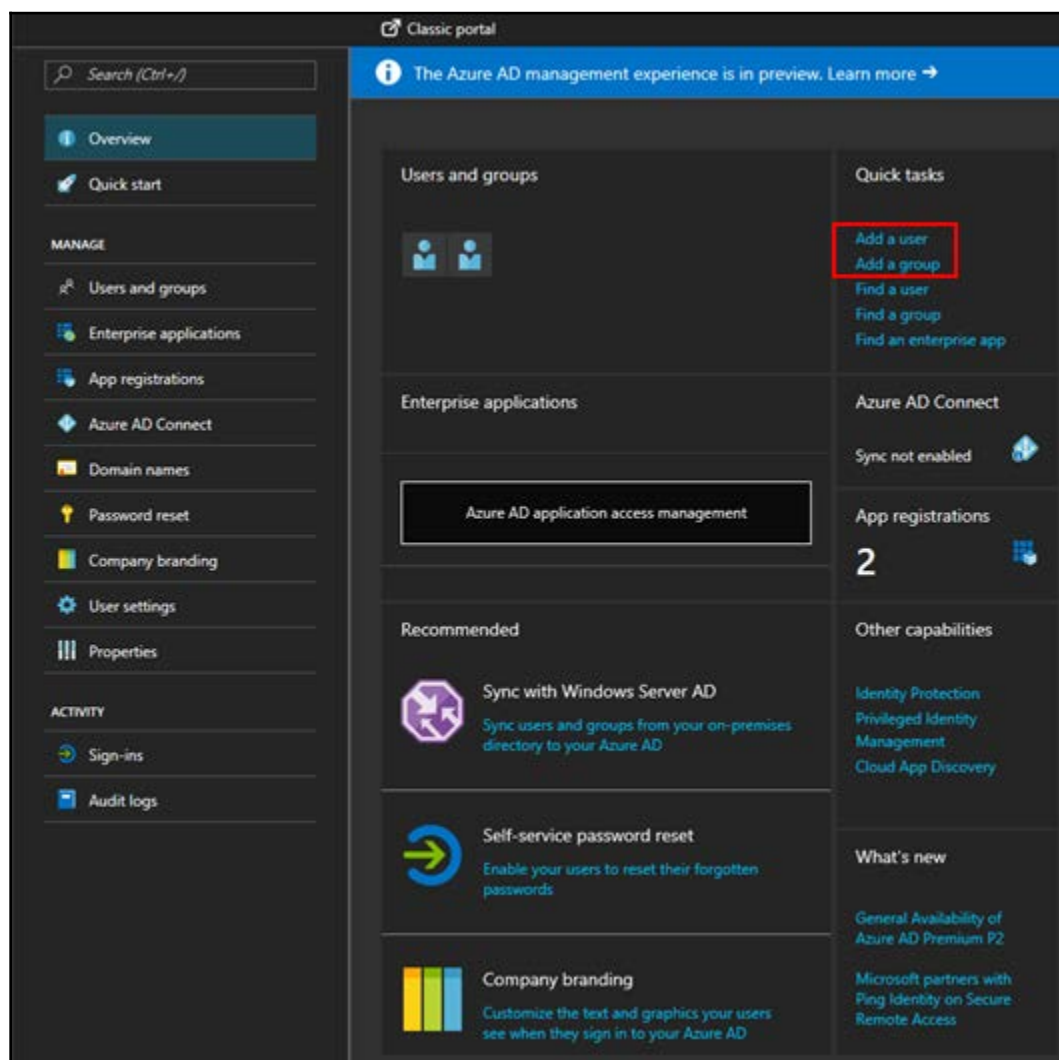


Alternatively, to create new users in Azure AD through the Azure portal, you need to follow these steps:

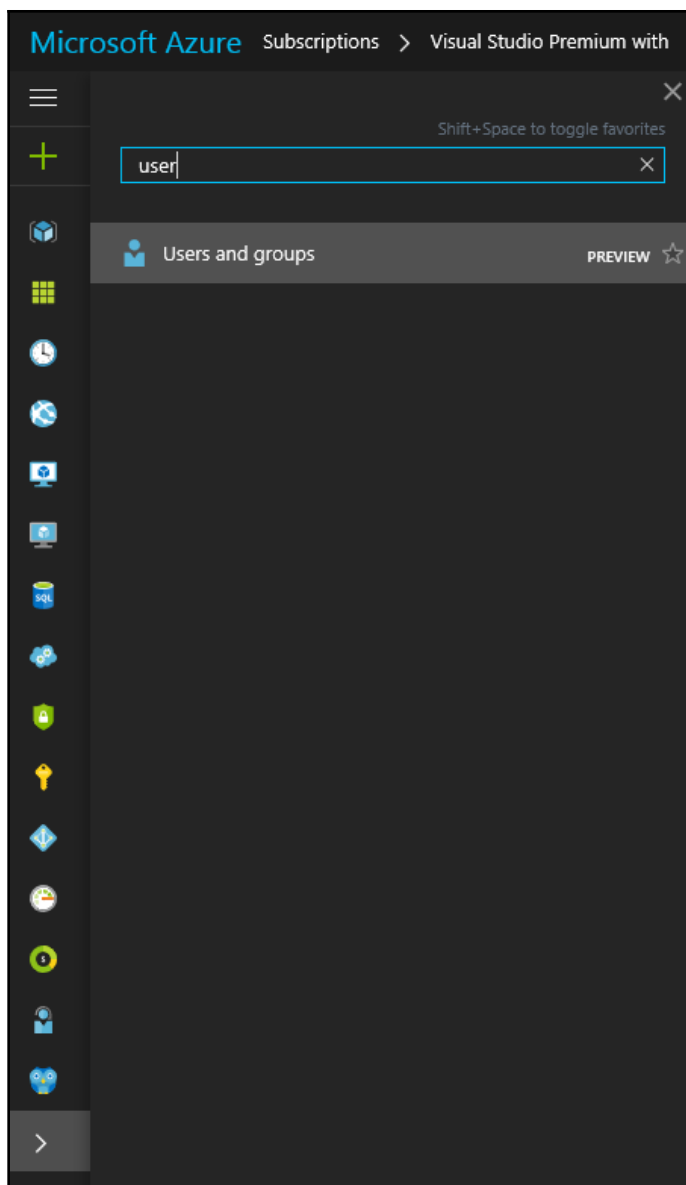
1. Browse to <https://portal.azure.com>.
2. Click on the **More services** option on the sidebar:



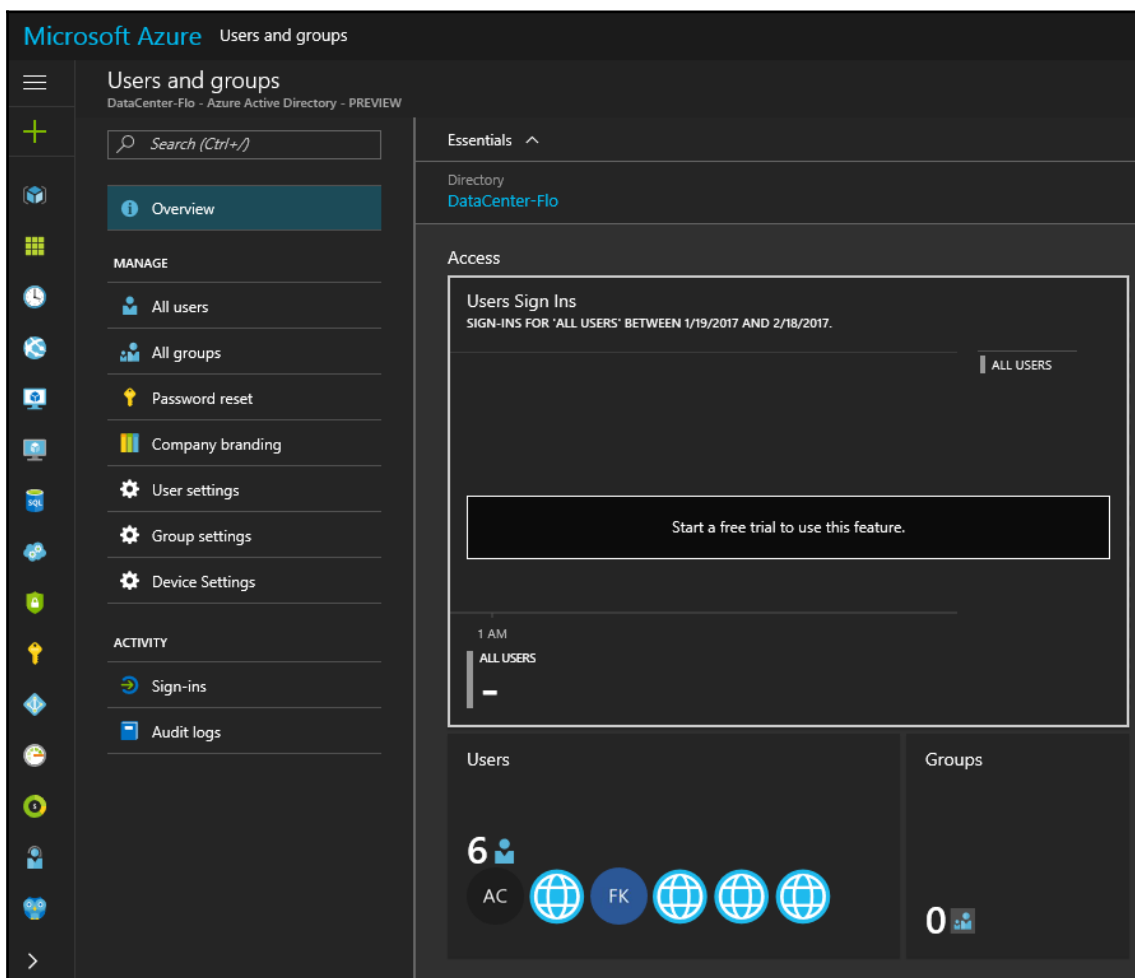
3. The new Azure AD interface does not yet have all features enabled. Currently, it is not possible to create new Azure active directories, but you can perform most user and application operations.
4. In addition to the new portal, Microsoft also extended user management to the Azure AD management portal. At the time of writing this 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 options. The first one would be to open user and group management through the Azure AD interface:



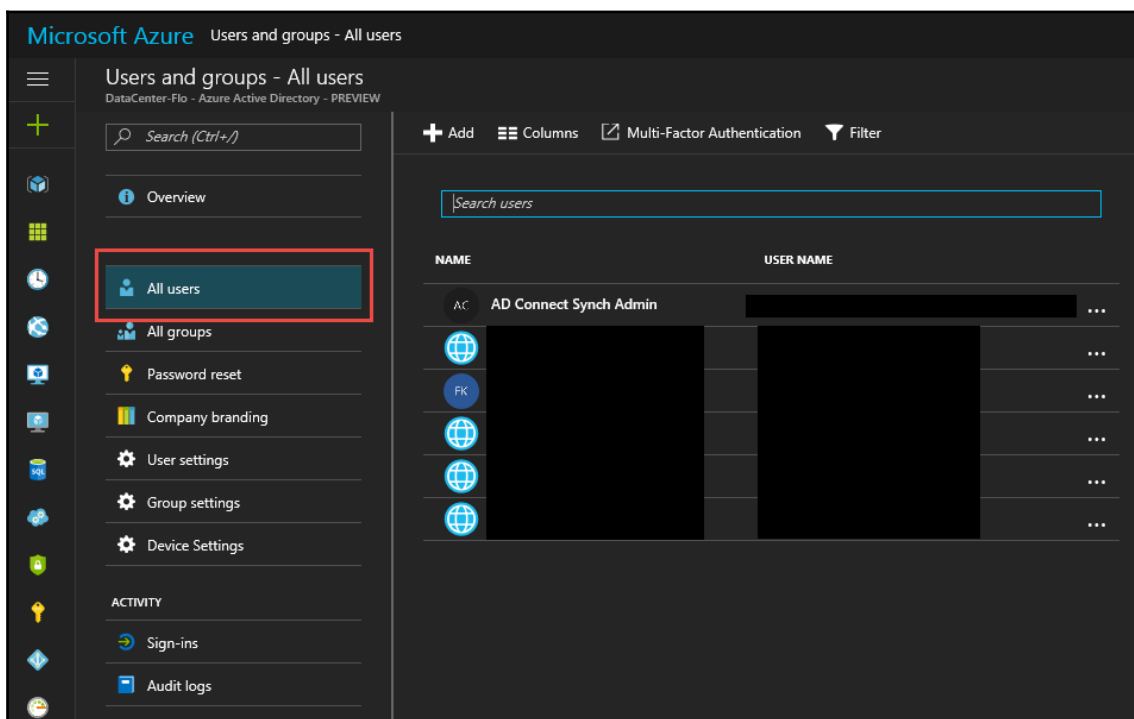
5. The next step is look for **Users and groups** with Microsoft resources:



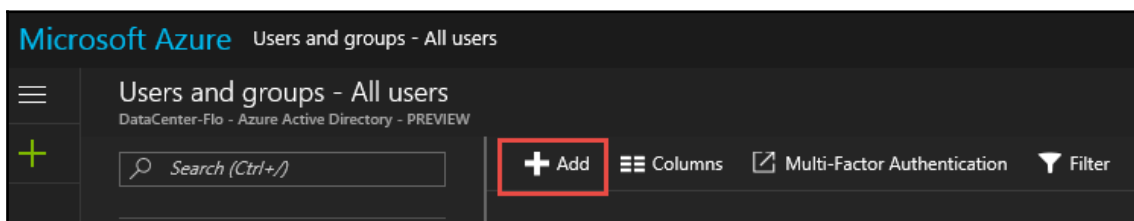
6. Both ways will bring you to the same blade, with options to create users and groups as shown in the following screenshot:



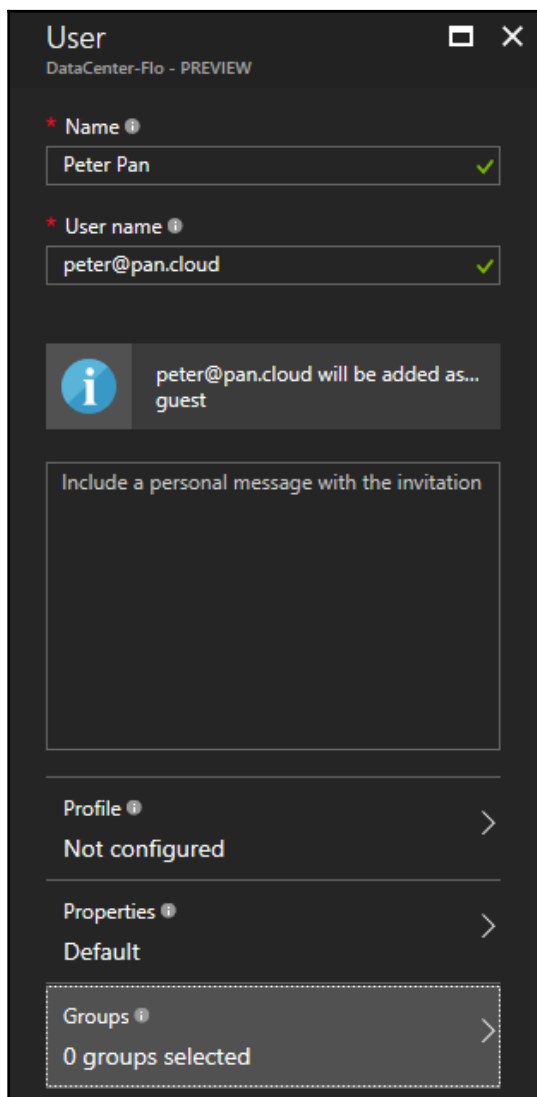
7. To add a user in the new UI, you click on the **All users** section, as shown in the following screenshot:



8. There, you click on the **+Add** button and follow the instructions shown in the following screenshot:



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



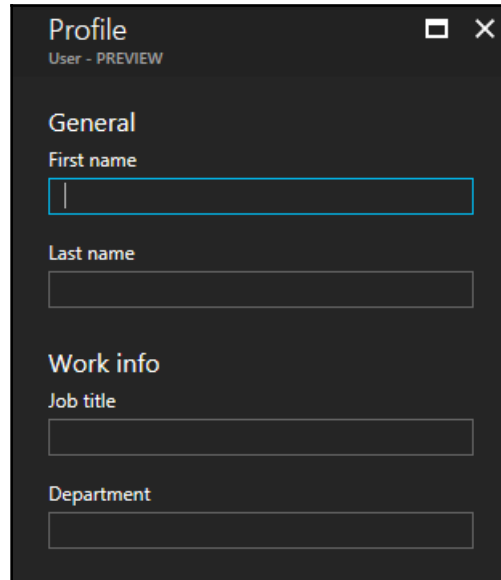
The screenshot shows the 'User' blade in the Azure portal. The title bar indicates 'DataCenter-Flo - PREVIEW'. The form contains the following fields and sections:

- 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 an information icon and the text 'peter@pan.cloud will be added as... guest'.
- Personal message:** A large text area with the placeholder text 'Include a personal message with the invitation'.
- Profile:** A section with the text 'Profile' and 'Not configured', followed by a right-pointing arrow.
- Properties:** A section with the text 'Properties' and 'Default', followed by a right-pointing arrow.
- Groups:** A section with the text 'Groups' and '0 groups selected', followed by a right-pointing arrow.



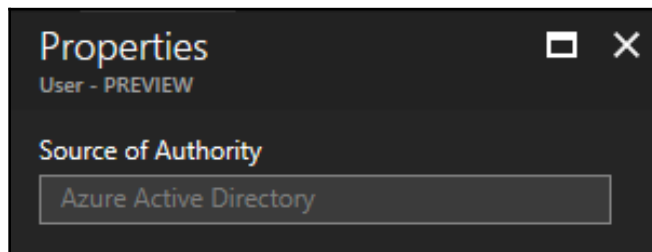
Be aware that with Azure AD free without Office 365, the username must be an email or a Microsoft account (formerly a Microsoft Live account) to be able to receive the invitation from Azure AD.

10. While creating the user, you have different options for pre-staging information about the user, including the **First name**, **Last name**, **Job title**, or **Department** fields as shown in the following screenshot:



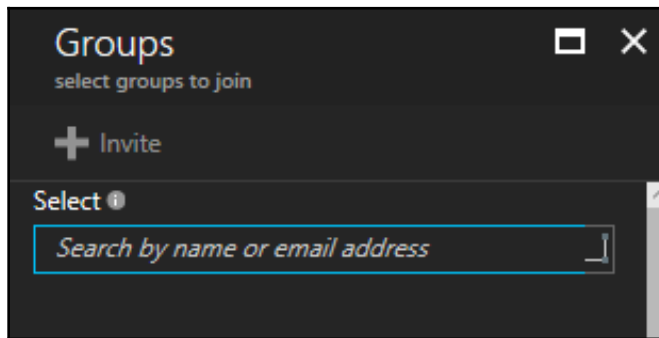
The screenshot shows a 'Profile' window with the subtitle 'User - PREVIEW'. It contains two sections: 'General' and 'Work info'. Under 'General', there are input fields for 'First name' (which is highlighted with a red border), 'Last name', 'Job title', and 'Department'. The 'Work info' section also has input fields for 'Job title' and 'Department'.

11. With a synced AD DS and other joined services, you can change the **Source of Authority** option:

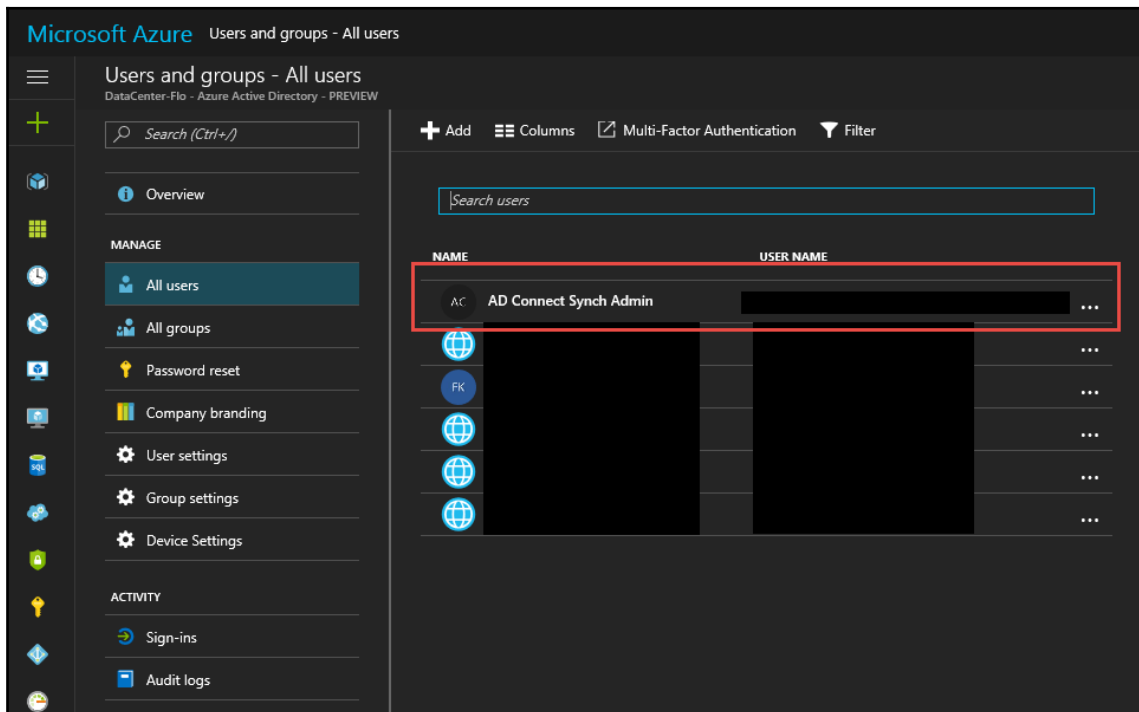


The screenshot shows a 'Properties' window with the subtitle 'User - PREVIEW'. It features a 'Source of Authority' dropdown menu, which is currently set to 'Azure Active Directory'.

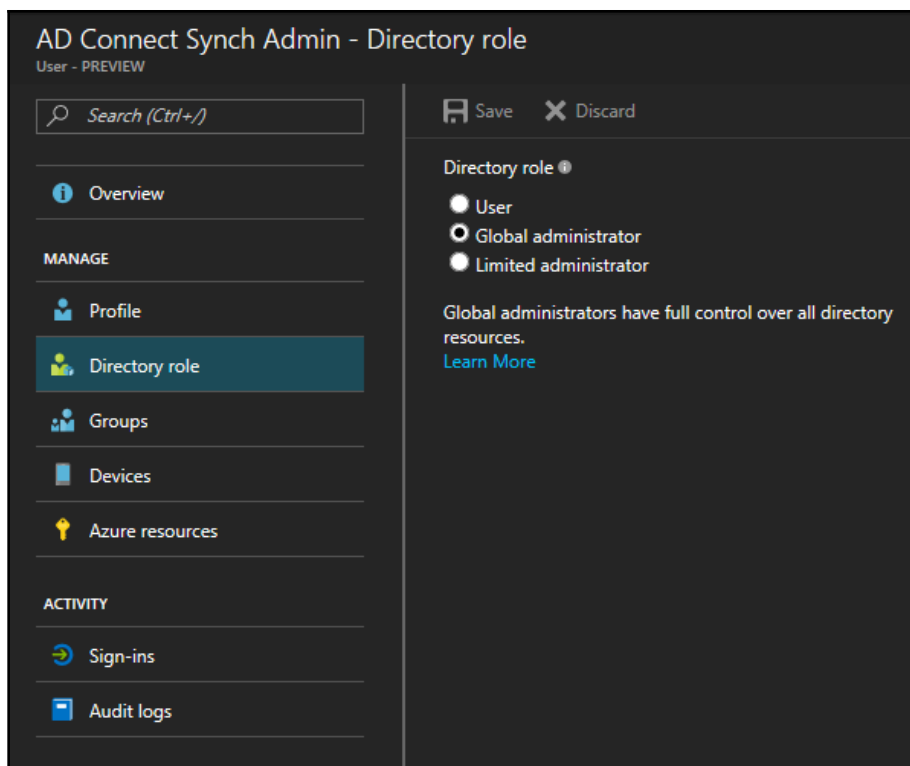
12. You can also join the account directly to Azure AD groups during creation:



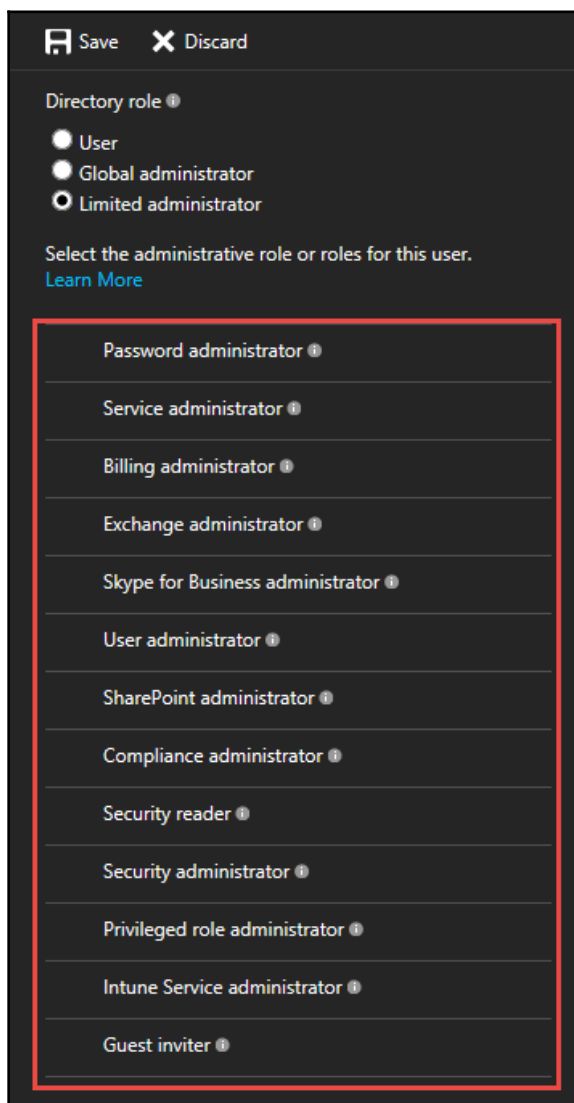
13. The new UI for Azure AD includes a new option to join users as account admins (formerly known as co-administrators). To do this, change the administrator rights of the user by clicking on the relevant user to open the user blade:



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



15. Not every user needs to be global administrator to be able to fulfill their job. Mostly, one or more of the options of a limited administrator should be enough:



Save Discard

Directory role ⓘ

☐ User

☐ Global administrator

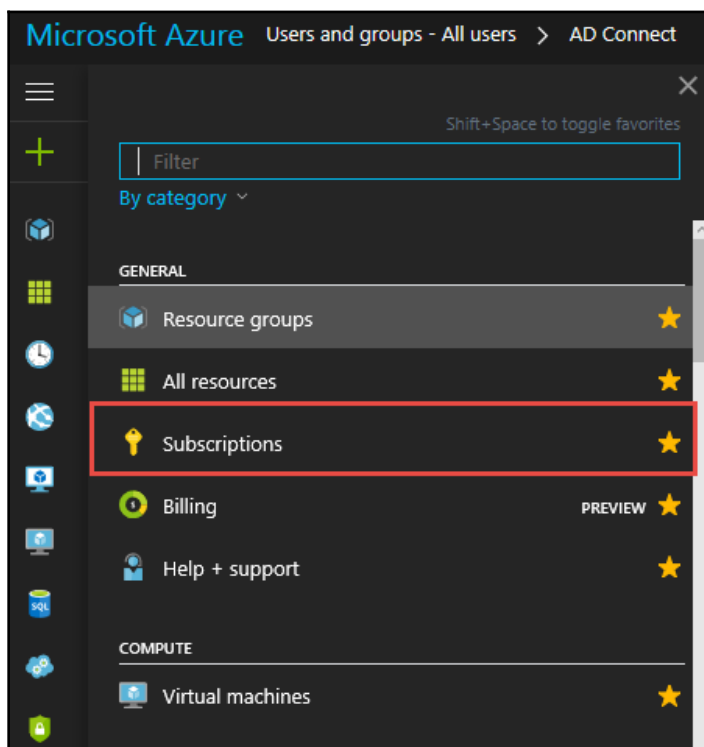
☒ Limited administrator

Select the administrative role or roles for this user.

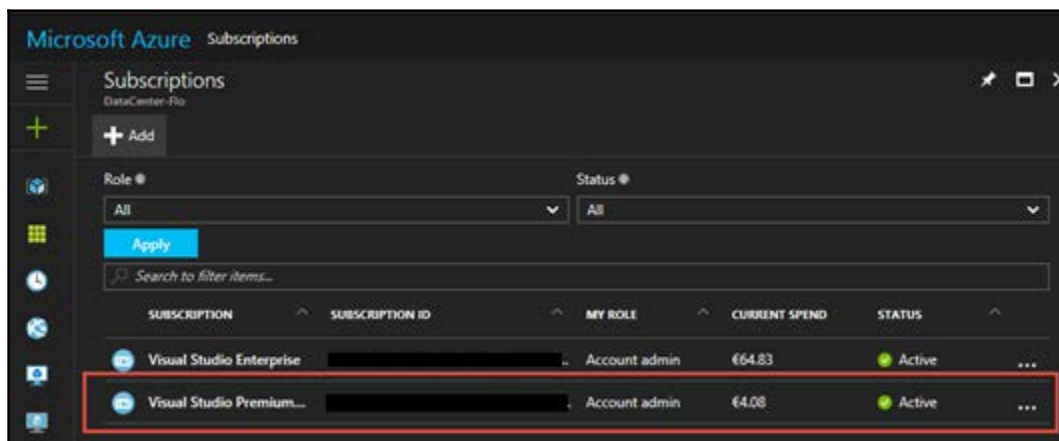
[Learn More](#)

- 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 ⓘ

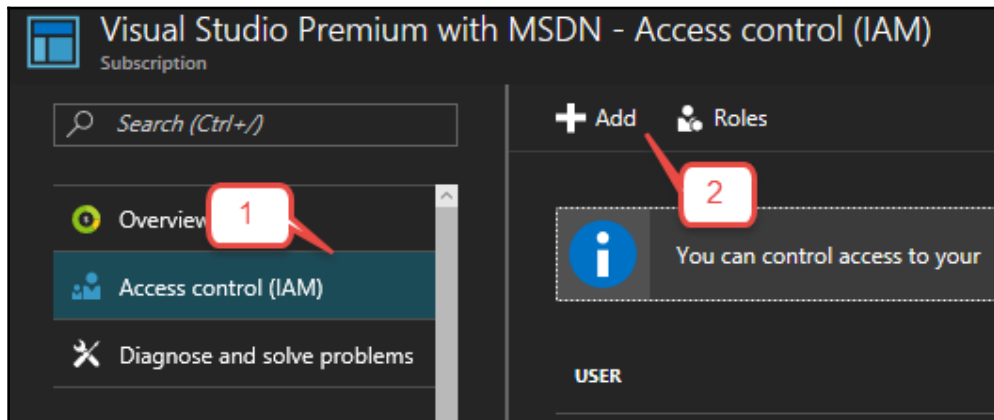
16. After you add the user to their admin role, you need to go to the **Subscriptions** section in the Azure resources:



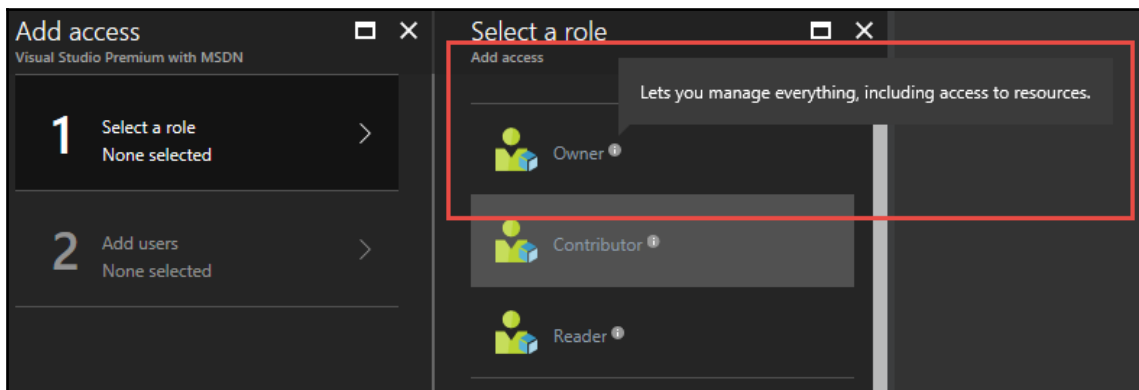
17. There, you select subscriptions for which the user should be the co-administrator:



18. In the following blade, click on the **Access control (IAM)** section and then on the **+Add** button:



19. Now, select a role for the user. To make them a co-administrator, we need to give them **Owner** rights:



As an owner, they have full access to the subscription resources, which will enable them to do all operations that need to be done on a subscription. As a reader, they can see billing and resources for a subscription but can't change things inside the subscription. That is the option most billing tools such as **Microsoft Cloudyn / Azure Cost Management, Cloud Cruiser, or Azure Costs** need to create their statistics.

20. The user is now able to manage the subscription. Next, select the user or group that 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 for 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 through AD DS from your connected on-premises Windows Server AD DS. To do this, you'll need an additional tool named **Azure AD Connect**.



From a security and compliance perspective, never synchronize the administration account to Azure AD. Leave that account for on-premises AD DS only and create cloud-only accounts in Azure AD for administrative work in Azure or Microsoft 365 Services.

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

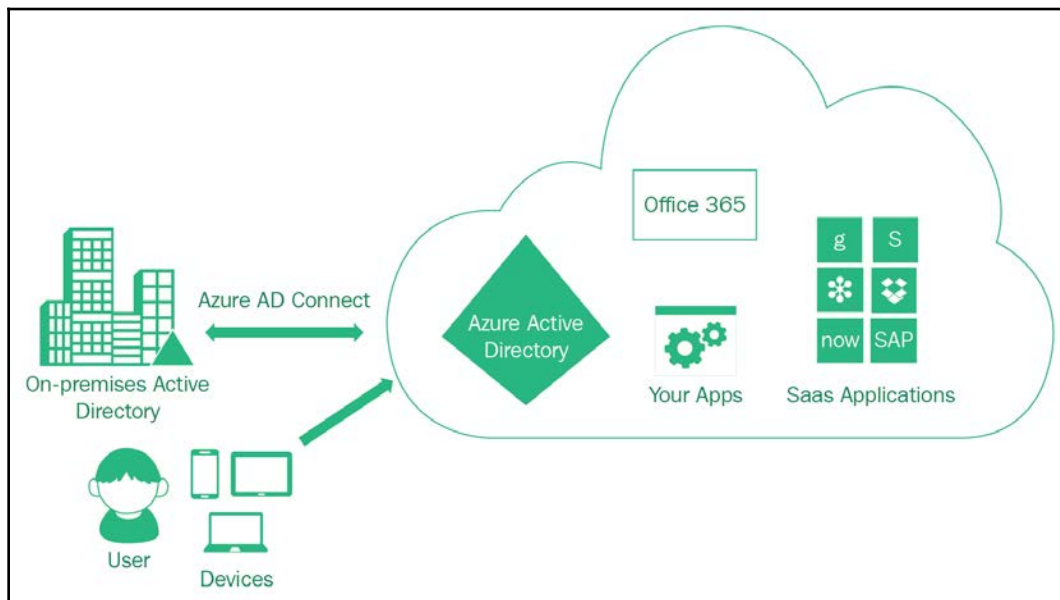


Image source: <https://azure.microsoft.com/en-us/documentation/articles/active-directory-aadconnect/>

Azure AD Connect is the central tool with which to implement hybrid identities in Azure, and enables you to license your software or implement identity and access management tools such as SSO, MFS, or **Active Directory Rights Management Services (AD RMS)**.

Azure AD Connect brings five programs with different purposes:

- **Azure AD Connect:** This is a configuration tool with an integrated and detailed wizard to configure Azure AD and on-premises AD DS synchronization.
- **Synchronization rules editor:** This is a basic tool to configure and customize synchronizations between Azure AD and on-premises AD DS:

Synchronization Rules Editor

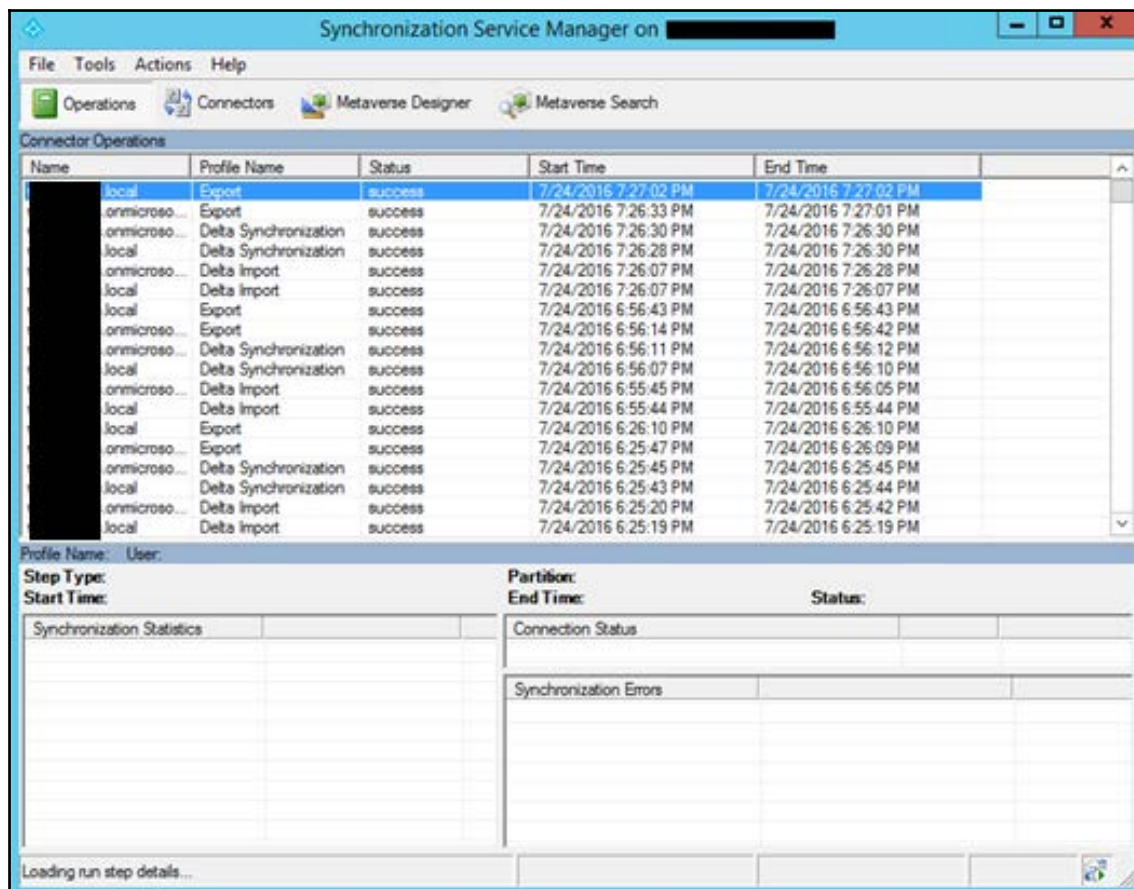
View and manage your synchronization rules

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

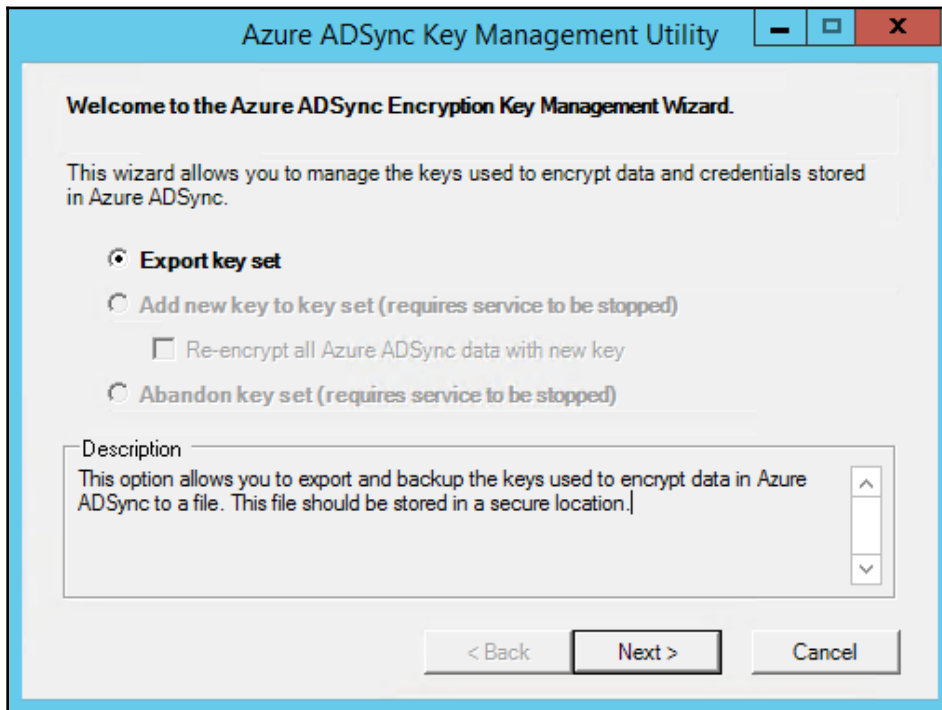
Name	Connector	Precedence	Connector Object Type	Metaverse Object Type
------	-----------	------------	-----------------------	-----------------------

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

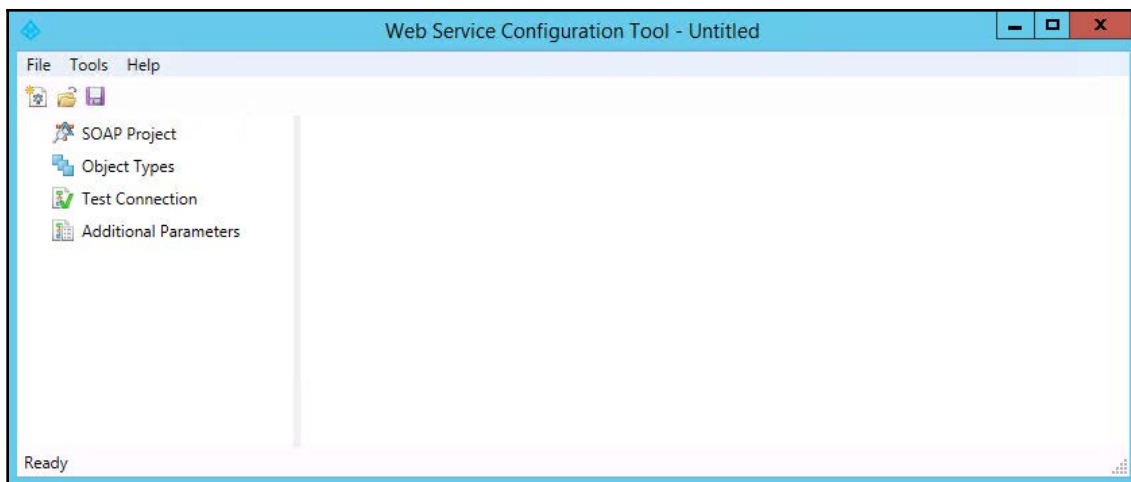
- **Synchronization service:** The synchronization service is a tool to basically monitor and log synchronization between Azure AD and on-premises AD DS. You can supervise the synchronization process:



- **Synchronization service key manager:** Helps you manage security keys to encrypt data transferred between Azure AD and on-premises AD DS:



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



Azure AD Connect and all its 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 users. Each method provides different integration levels and is more or less dependent on 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:

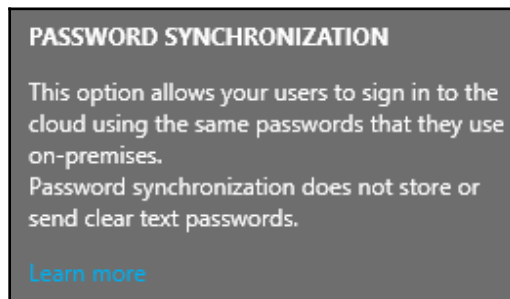
☒ Enable single sign on ? (Preview)

The first solution is to use the **Password Synchronization** option. This option transfers user passwords as hashed values to the cloud. This option is also the one that enables a basic SSO option for your users to Azure AD-based applications in your organization. This 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 fewer 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:

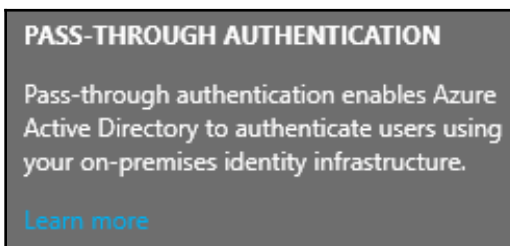


To choose the right sign in option for your organization, please refer a very detailed guide on that topic here: <https://blogs.msdn.microsoft.com/samueld/2017/06/13/choosing-the-right-sign-in-option-to-connect-to-azure-ad-office-365/>.

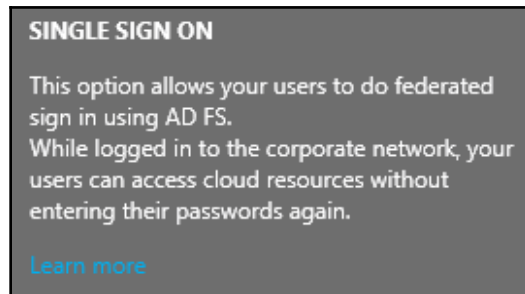
- The following screenshot shows you the tooltip you get when you hover over the question mark for the **Password Synchronization** option:



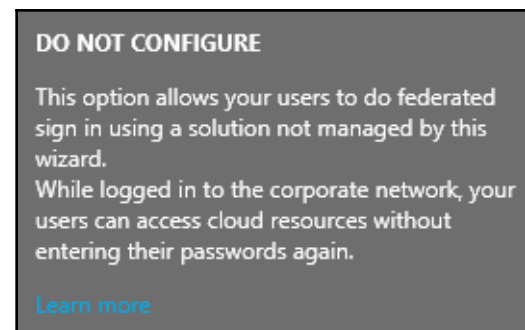
- The second option is to select the new **Pass-through authentication** option. With this option, your AD Connect and Azure AD will not store any identity data, and they will send all requests directly to your environment. The following screenshot shows you the tooltip you get when you hover over the question mark for **Pass-through authentication**:



- The third and most complex solution is to implement **Active Directory Federation Services (AD FS)** with Azure AD, in the interface named federation with AD FS. That will enable full SSO and add MFA. The organizations and implementations. If you want to implement AD FS, you need also to have public and **private key infrastructure (PKI)** and certificates from a trusted agency in place. For AD FS, 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 for **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 the Azure remote app. The following screenshot shows you the tooltip you get when you hover over the question mark for **Do not configure**:



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



Installing Azure AD Connect – prerequisites

For those who haven't worked with AD FS before, federations offer a standardized 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 incoming claims to claims that are understood by its web application, which uses these 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 (DC)** or member server with Windows Server 2008 or newer (see the following table for the appropriate sizing for that machine).

The DC 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, a 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 AD Connect, you need to configure two user accounts. The first one should be a service account with enterprise administrator rights, or for sub-domains with domain administrator rights within your on-premises AD DS. The other one must be a global administrator in Azure AD.

The following screenshot shows 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

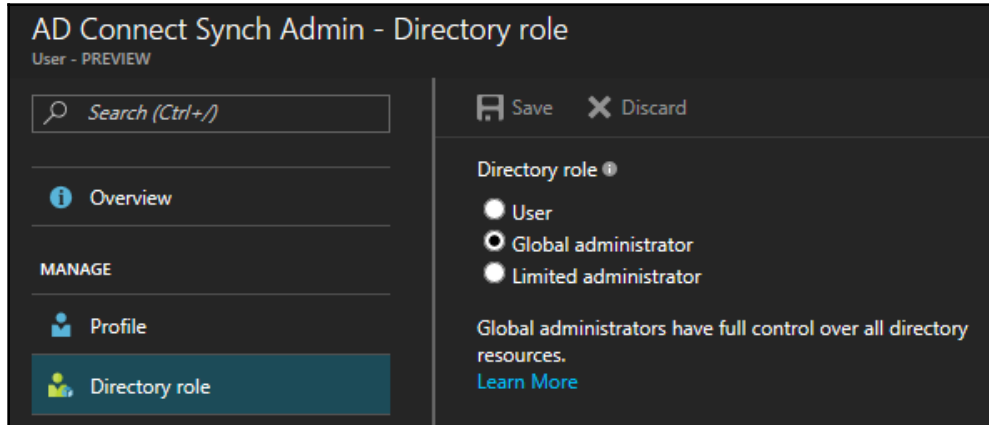
Include a personal message with the invitation

Profile ⓘ
Not configured >

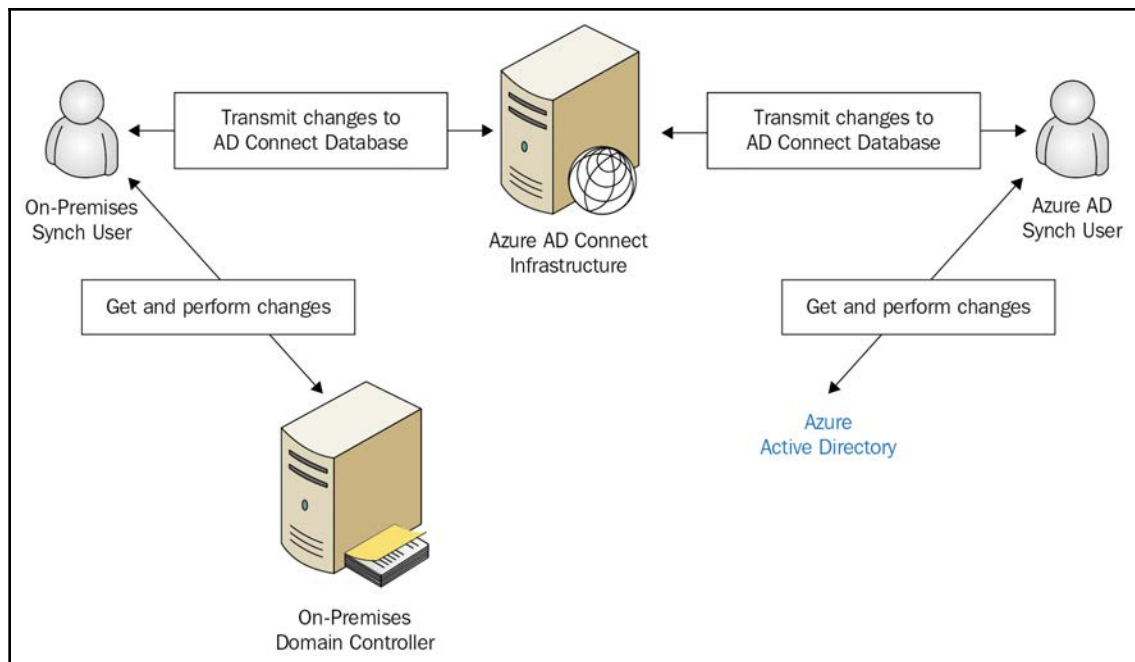
Properties ⓘ
Default >

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:

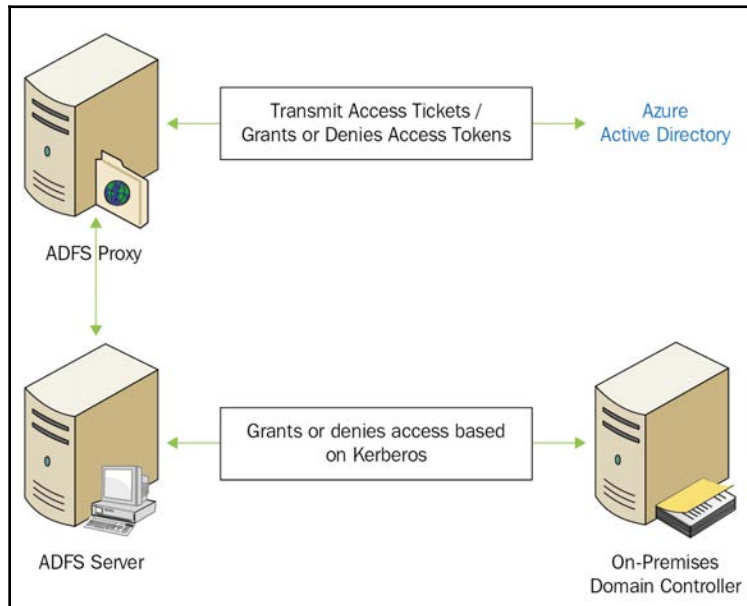


The following diagram shows you the basic workflow behind communication between Azure AD, AD Connect, and on-premises AD DS:



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 AD FS implementation, you need to perform some more steps to get the federation running. For the AD FS implementation, you will need at least one AD FS Server and one AD FS 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 AD FS implementation.

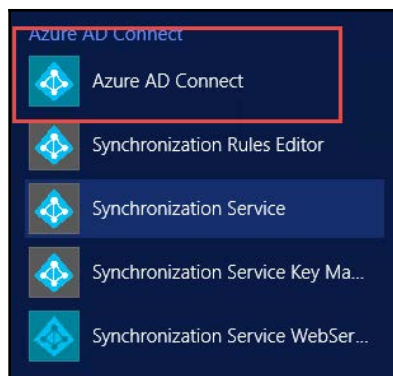
The following diagram reflects the more complex environment:



Installing a basic Azure AD Connect environment

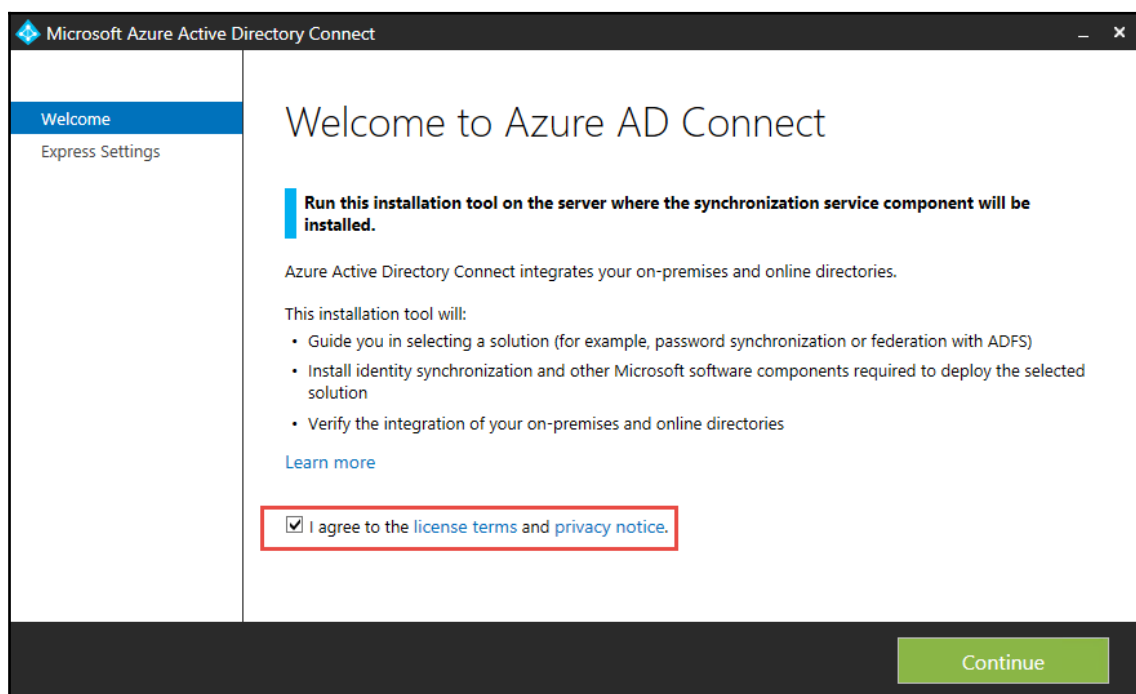
As an example, I will show you how to implement Azure AD Connect with password synchronization. The other options for the setup are similar:

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

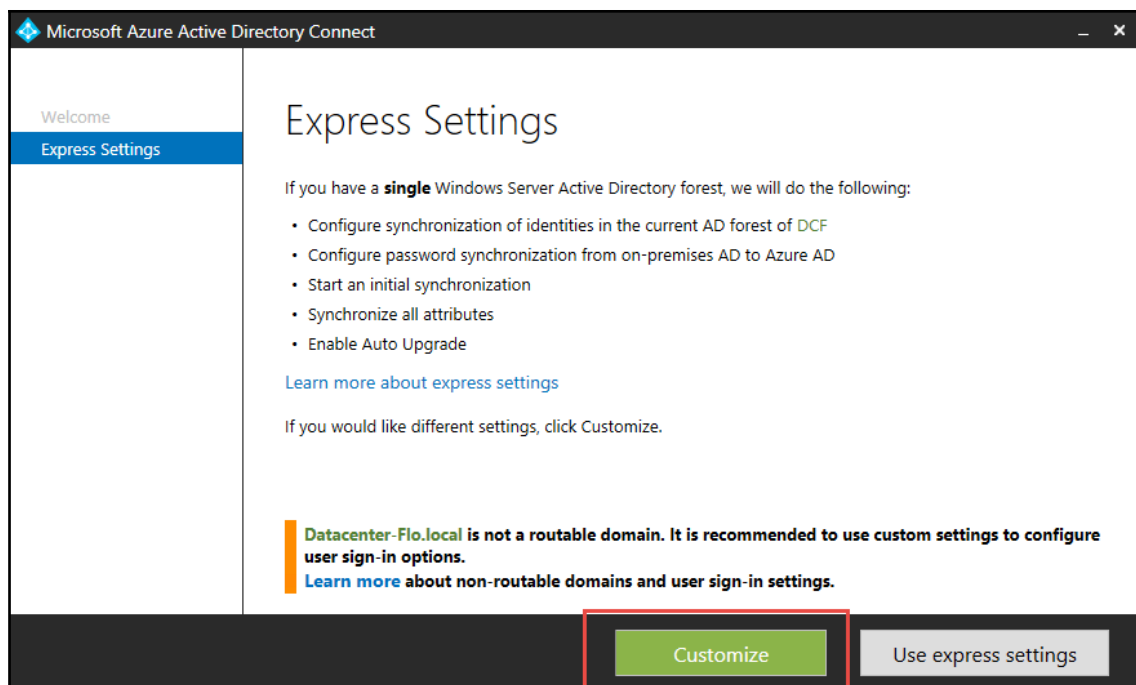


Be aware your AD Connect Server must be joined to the on-premises AD domain.

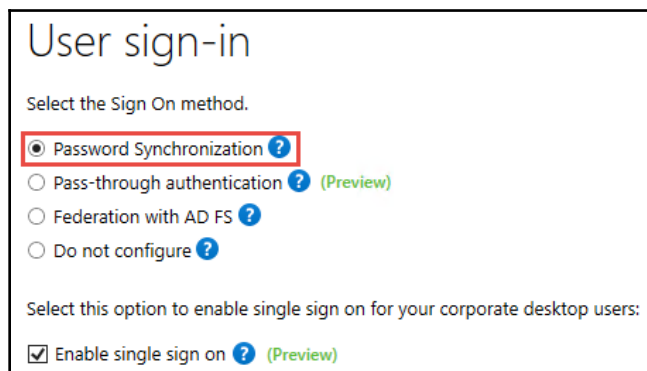
2. Accept the license terms and click on the **Continue** button as shown in the following screenshot:



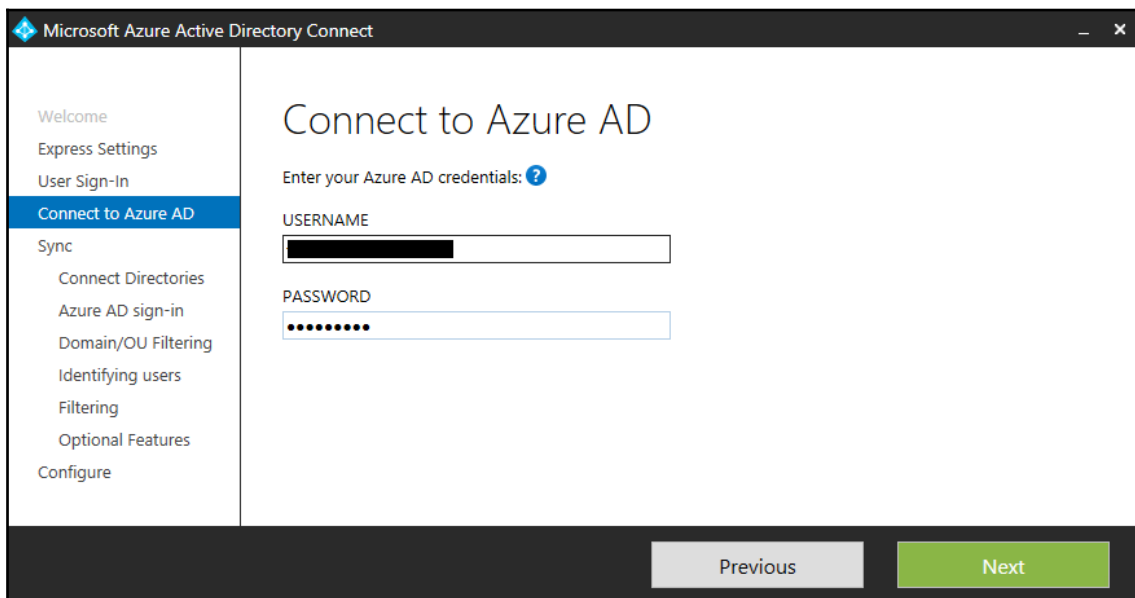
3. Click on the **Customize** button to continue:



4. Choose the **Password Synchronization** option and click on the **Next** button:



5. In the next window, enter the credentials for your Azure AD synch account:



The screenshot shows the 'Microsoft Azure Active Directory Connect' application window. 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 area is titled 'Connect to Azure AD' and contains the text 'Enter your Azure AD credentials: ?'. Below this are two input fields: 'USERNAME' and 'PASSWORD'. The 'USERNAME' field contains a blacked-out string, and the 'PASSWORD' field contains a series of dots. At the bottom right of the window are two buttons: 'Previous' (disabled) and 'Next' (active).

6. On many websites, and also in Microsoft documentation, you will find terms such as **DirSync** and **Azure AD Sync**. Both have no longer been supported since Azure AD Connect reached a production state. 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:



TIP

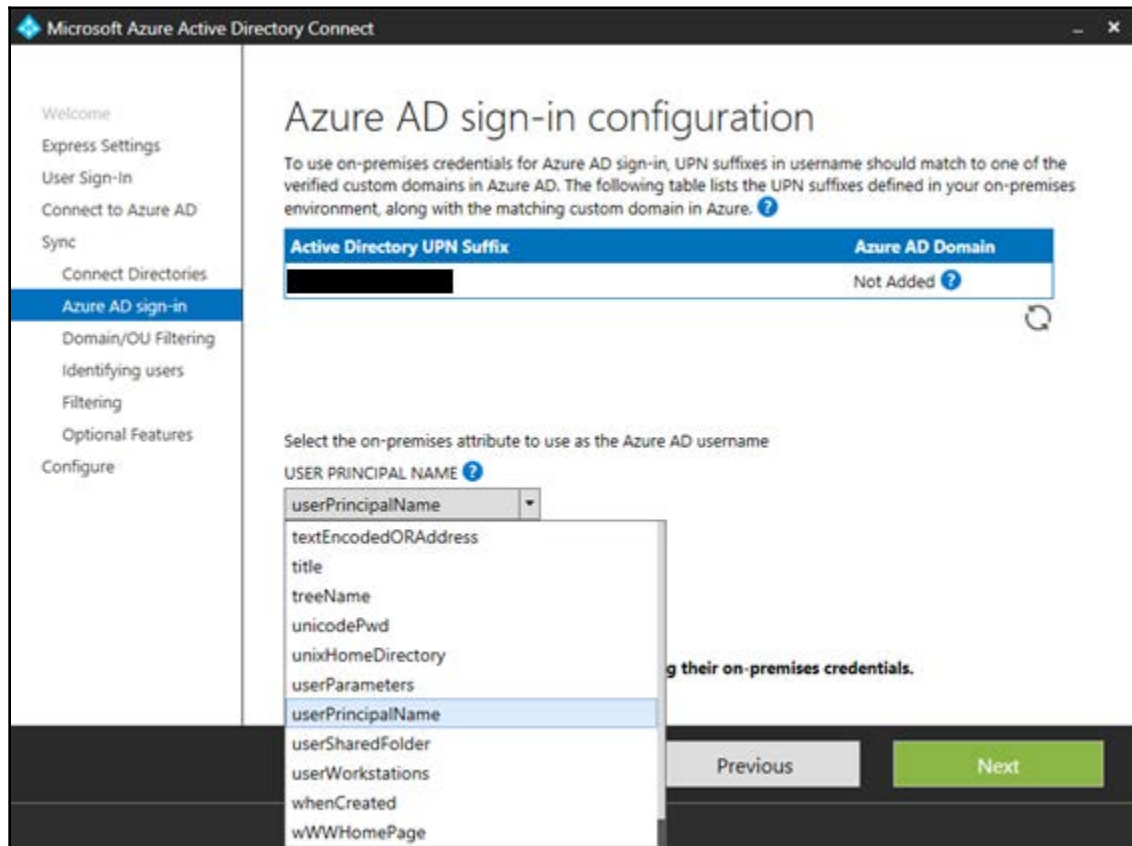
For multi-domain forest AD infrastructures, your on-premises administrator account needs to be an enterprise administrator. If you only have a single domain forest, you can use a regular domain admin.

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



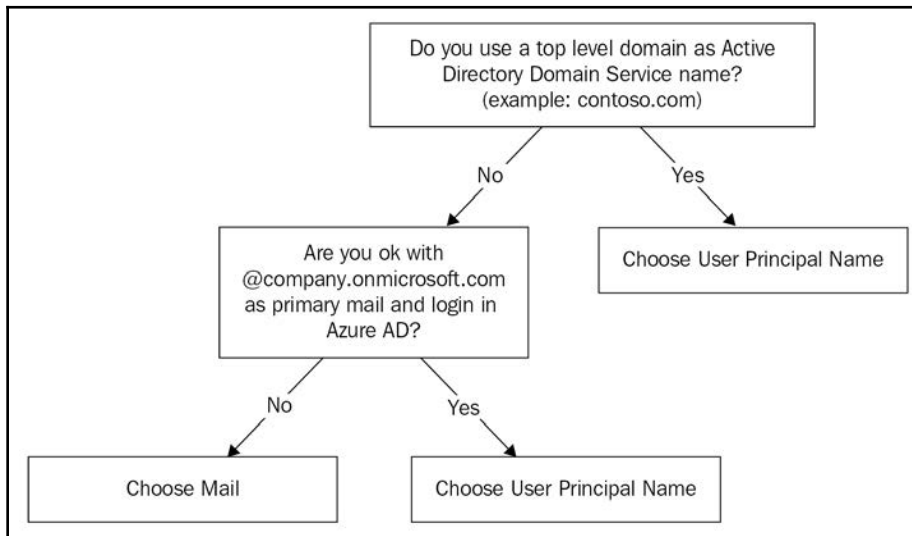
The UPN is an internet-style login name for a user based on the internet standard RFC 822.

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

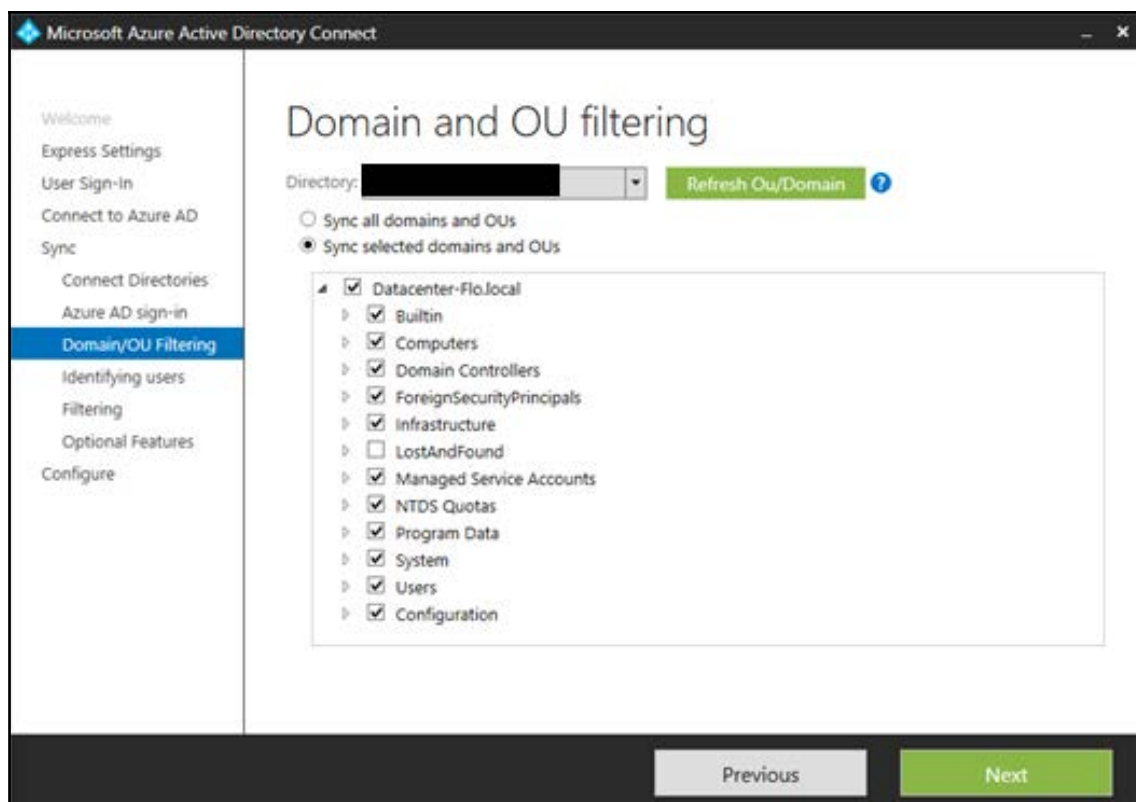


Just from the field, it is not always helpful to choose the principal user. Depending on the domain, sometimes it is better to choose the email instead of the principal user name in Azure AD. That's because the primary email in a synced scenario is always the login name. So, if your login name is `@company.onmicrosoft.com`,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

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



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



12. Now, choose how your users should be identified over your domain. Depending on your decision, you now select the principal user name or email:

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

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 input field with a dropdown arrow]

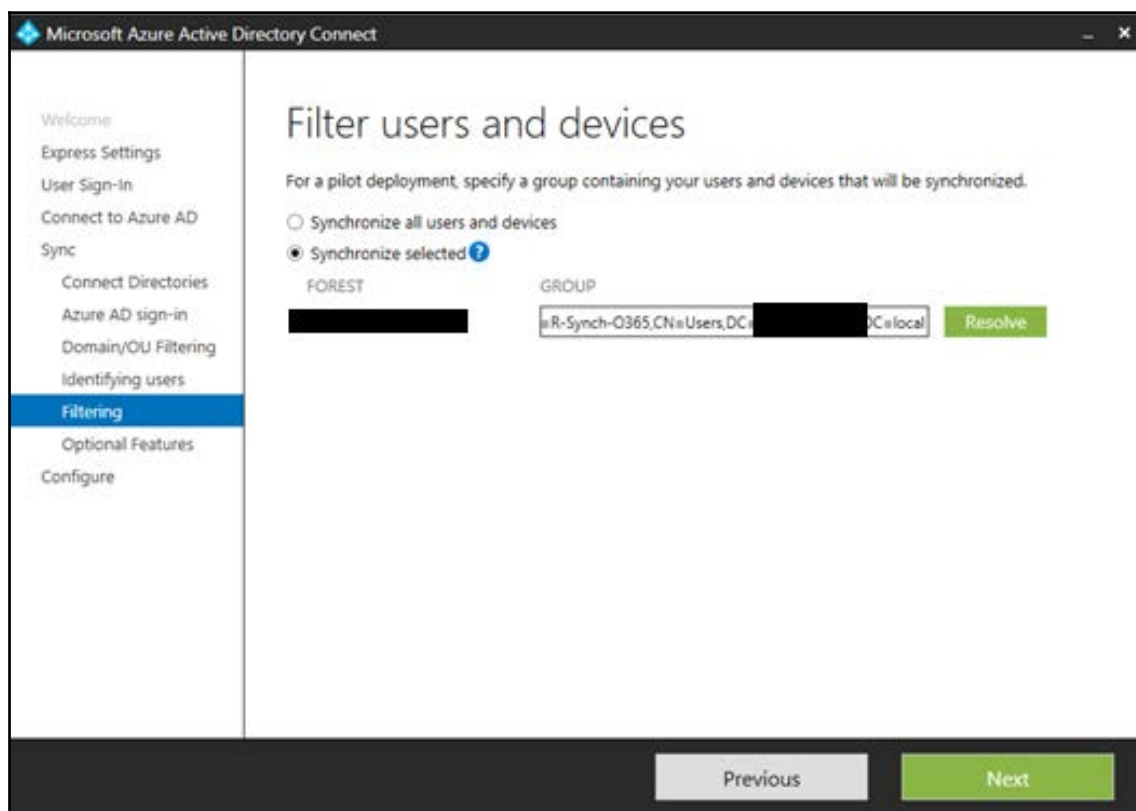
Select how users should be identified with Azure AD.

SOURCE ANCHOR ?

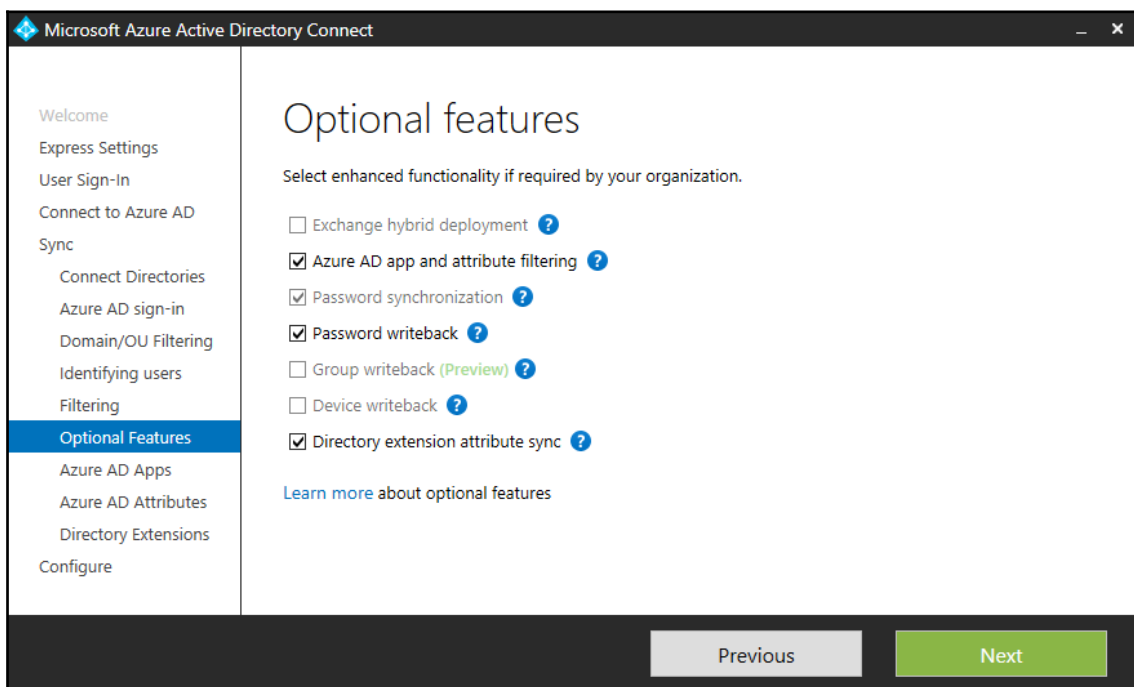
[Dropdown menu showing 'objectGUID']

At the bottom of the window, there are two buttons: 'Previous' (disabled, grey) and 'Next' (active, green).

13. Afterwards, configure the filter for your accounts. You can either synchronize all users and devices or use a filter group. Here, the best way is to have specified groups 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, choose which features you want to have synced in addition to the attributes:

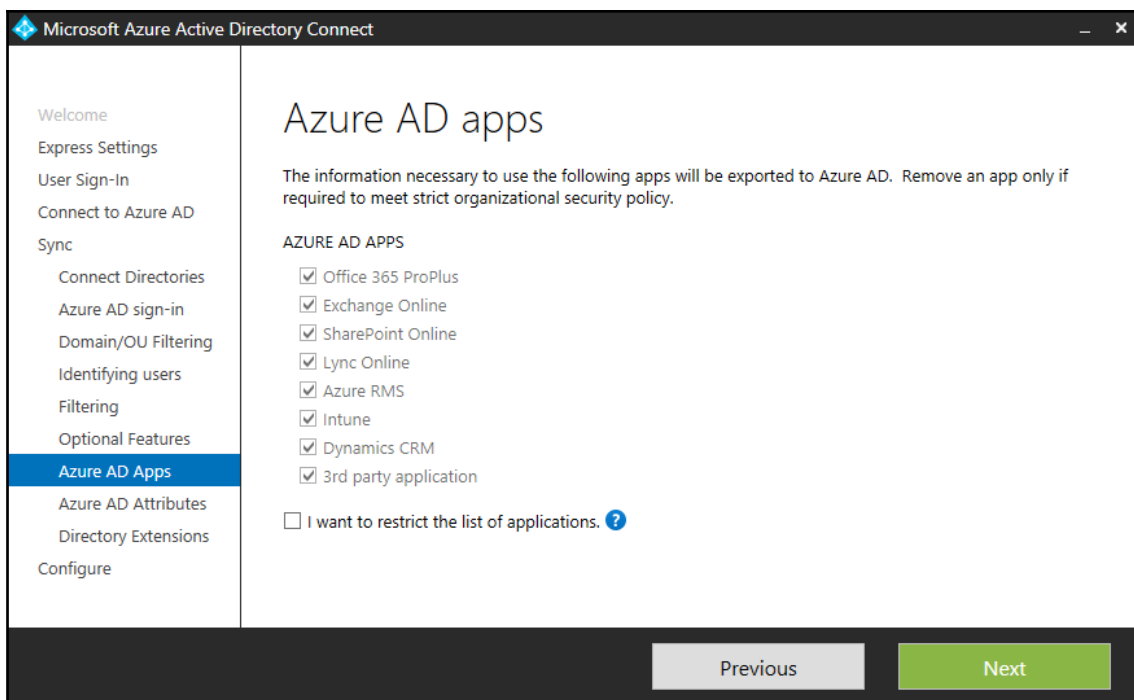


Some, such as the **Exchange hybrid deployment** and **Device writeback** options, are only available with certain AD extensions, or with additional subscriptions such as Azure AD premium or mobile device management such as Intune. Best field practice is to at least sync the **Azure AD app and attribute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

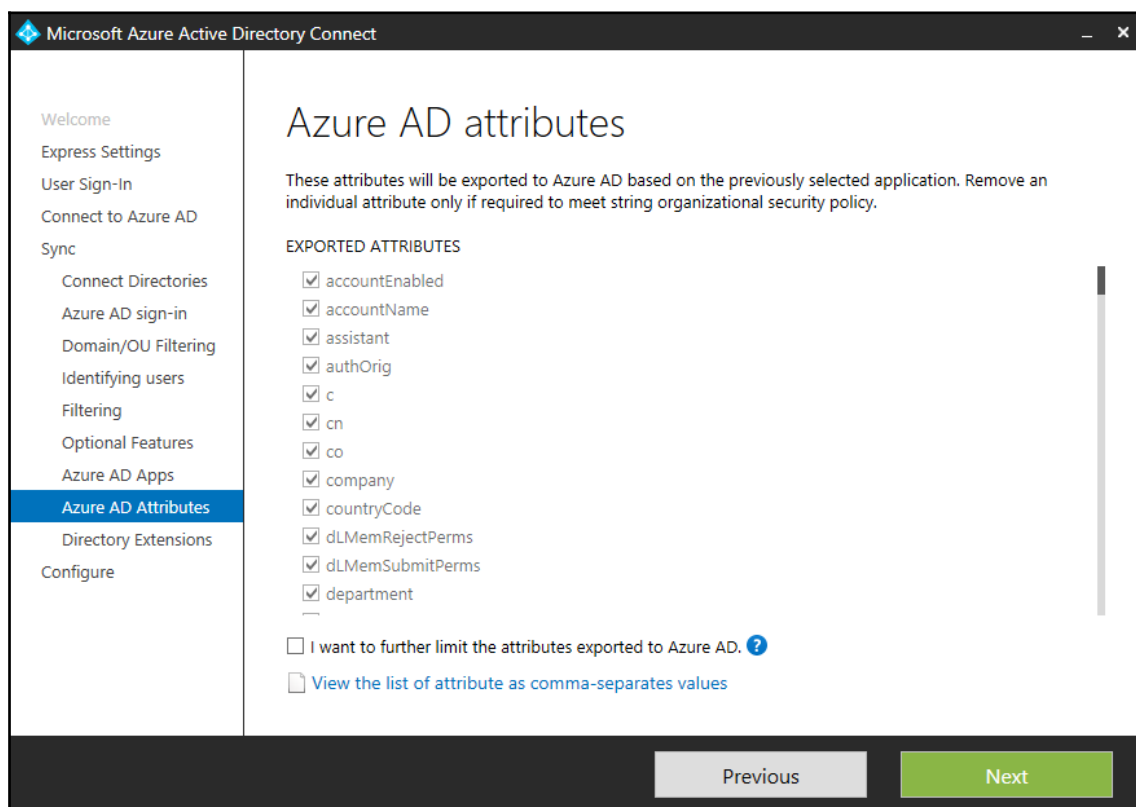


If you are running a Microsoft Exchange within your AD DS, enable the **Exchange hybrid deployment** checkbox. There are certain little challenges with configuring it afterwards. One of common one is broken Exchange mail routing for users who had an Exchange Online mailbox without an on-premises connection before.

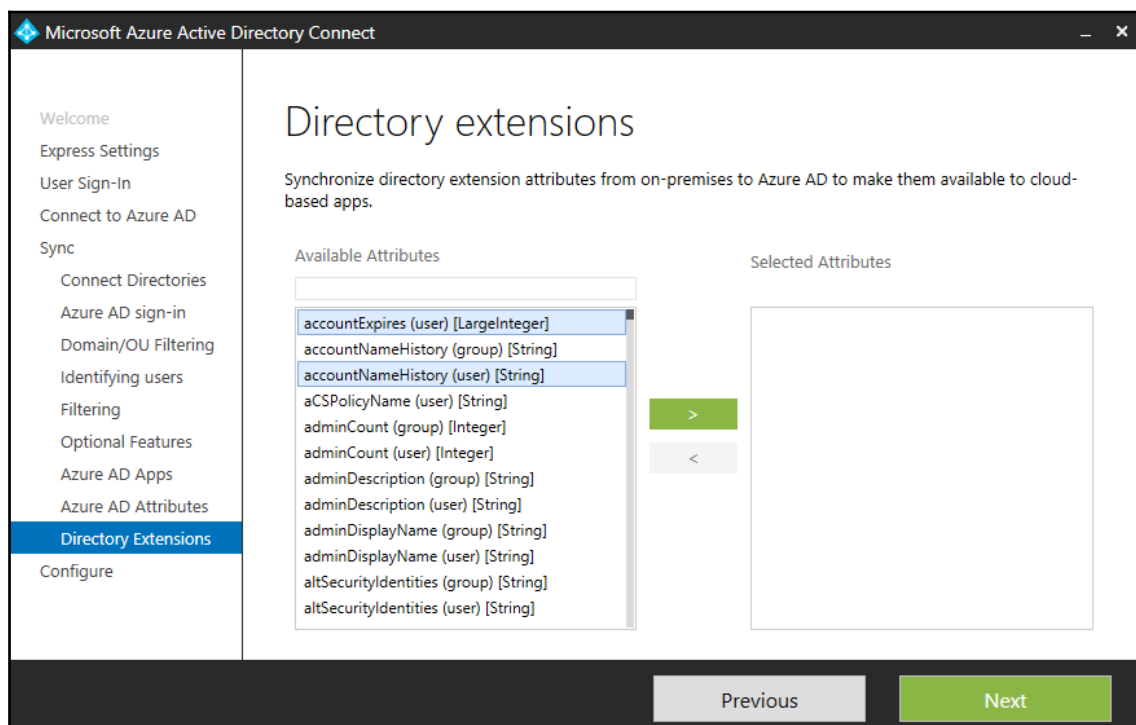
15. In the next few windows, you can decide which Azure AD apps will be synced:



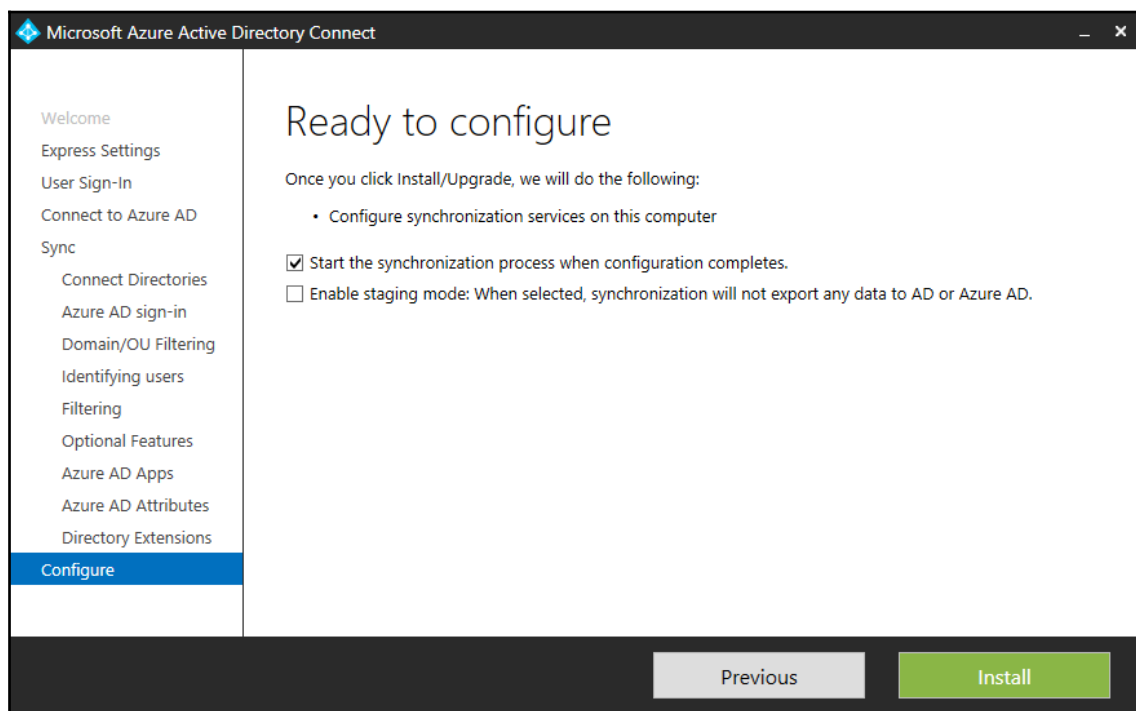
16. Afterward, you can limit the attributes from AD that will be transferred to Azure AD. Microsoft does not recommend limiting attributes:



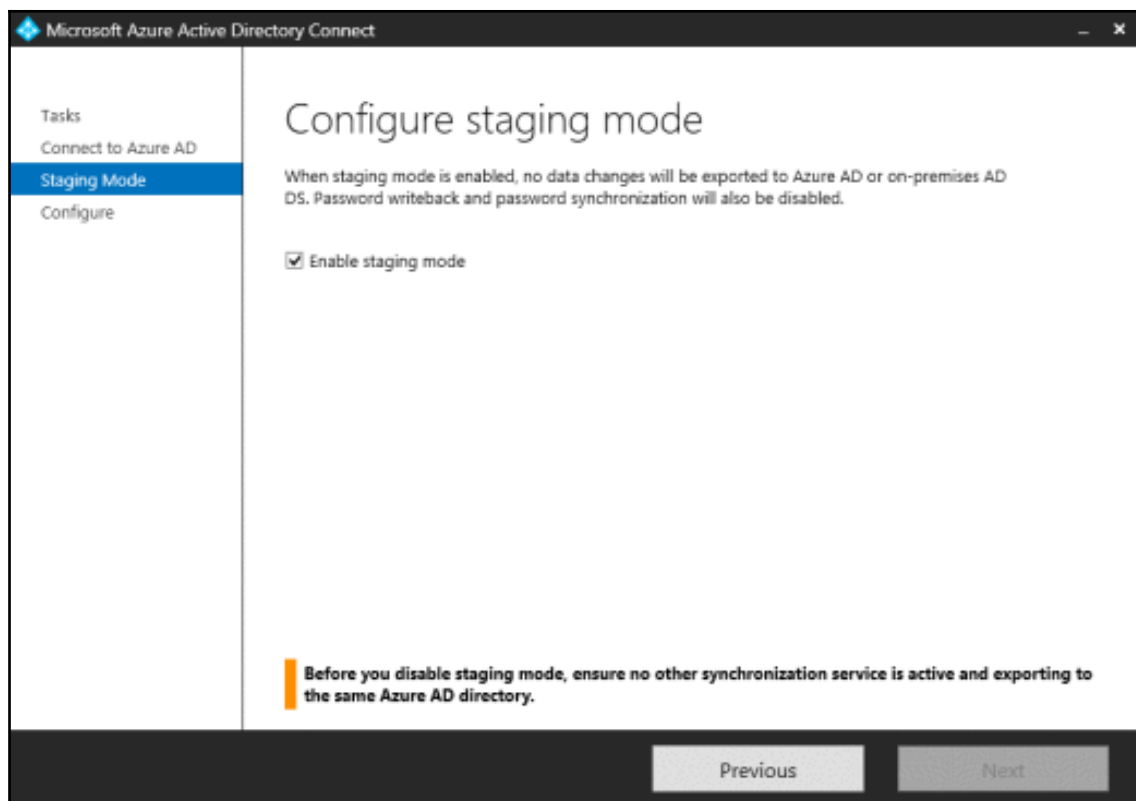
17. Now, you can select which directory extension attributes from your on-premises environment are transferred to Azure AD. This is important for certain applications running in the cloud. In this walk-through, we will not select any attributes:



18. Once you've run through these steps, you can complete the installation. Outside 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 afterward:



19. To disable 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 staging mode:



You have now implemented a simple, SSO-enabled Azure AD and AD synchronization. After you've started synchronization, Azure AD Connect will replicate changes every hour between Azure AD and AD.

You can trigger synchronization manually by using either of the 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 the settings you have configured in Azure AD Connect. It's likely to be what you would do when you start a sync *after* installation.



From a common practice and security standpoint, you should not sync any on-premises domain administrator accounts in Azure AD, and shouldn't give the personal Azure AD account of your IT people any admin rights in Azure AD. Each time, 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 highly available infrastructure

Now you know how to set up a basic AD synchronization without considering the availability infrastructure, 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 *staged mode* to implement it in passive mode. In the case of a failure, you have three days to recover the AD Connect Server. This can be either done through recovering from backup redeployment or switching to the staged, passive AD Connect Server.



So, for placement in either high availability or non-high availability infrastructures, it is recommended that you place systems that are involved in the synchronization of Azure VMs. This is so that you do not transfer as much data through the open internet, thus improves the performance of communication and identity token exchanges between Microsoft Cloud Services. More details about these concepts will be given in the next chapter about Azure networking.

In our high-availability scenario, every active and primary source of synchronization is placed into Azure and configured with either **Availability Zones (AZs)** or **sets**. A connection to on-premises is done through a **virtual private network (VPN)** or **Multiprotocol Label Switching (MPLS)** connection.

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

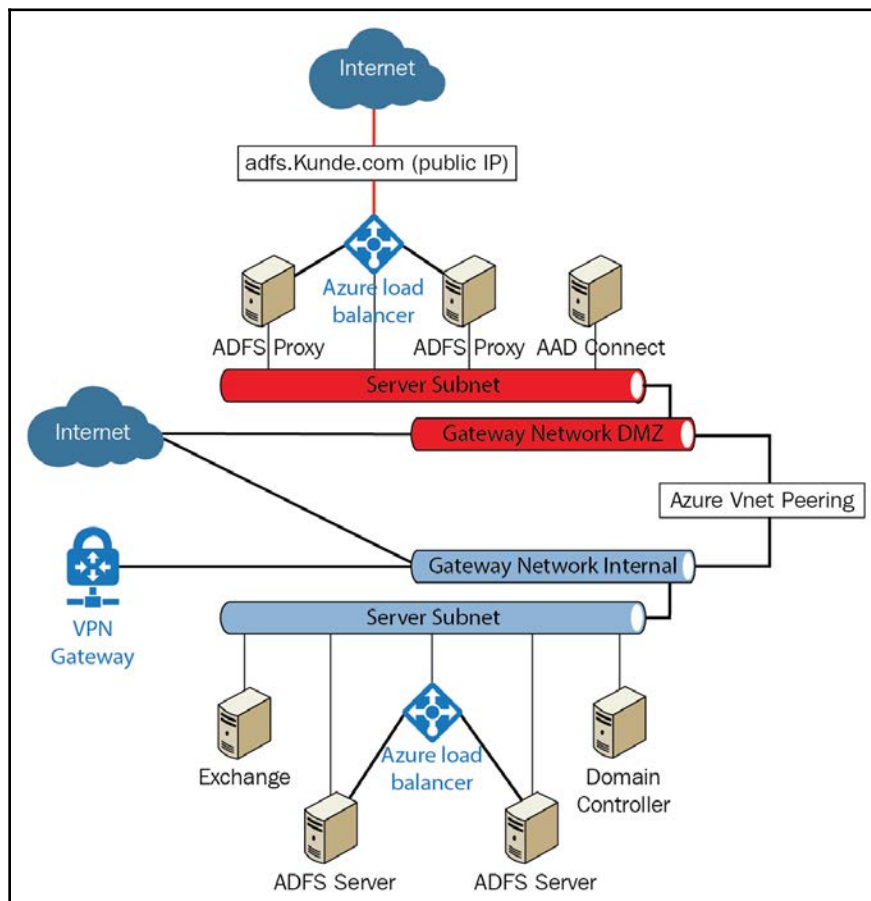
- Two AD DCs with global catalog and DNS
- Two Azure AD Connect servers, one in active and one in staged mode

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

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

- Two AD DCs with global catalog and DNS
- Two Azure AD Connect servers, one in active and one in staged mode
- Two AD federation servers
- Two AD federation proxies

Such an infrastructure could look like the following architectural schema:



Microsoft recently published a very good guide deploying AD FS in Azure. You can find the guide through the following link: <https://docs.microsoft.com/en-us/azure/active-directory/connect/active-directory-aadconnect-azure-adfs>.

Azure AD conditional access

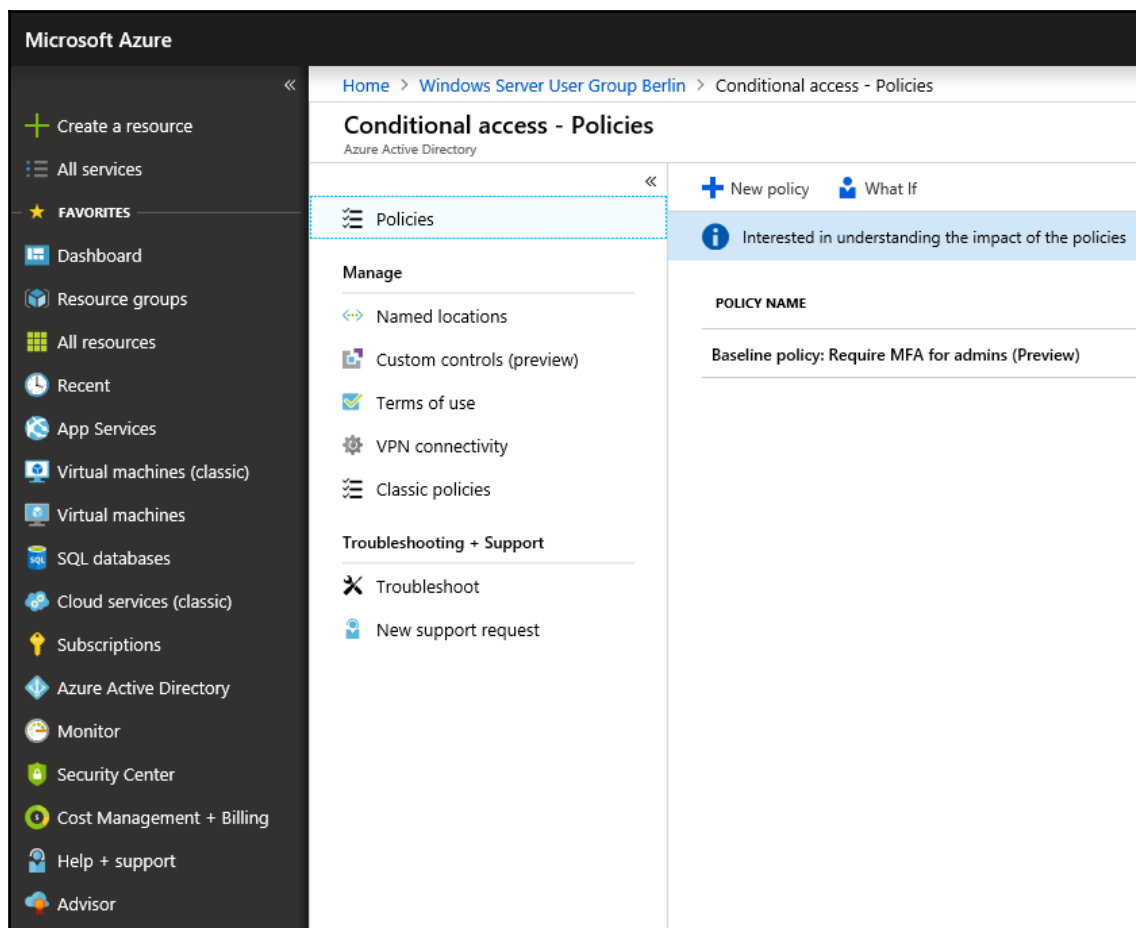
At this point, I want to give some credit to a very important child service of Azure AD. Azure AD conditional access is a very simple way to control and secure access to resources in the cloud and on premises. Azure AD conditional access is a premium feature in Azure AD. You can grade access, for example, by the following conditions:

- **Group membership:** Access based on group membership
- **Location:** Block controls when a user is not on a trusted network, or trigger MFA
- **Device platform:** Use the device platform (iOS, Android, Windows versions) to apply a policy
- **Device-enabled:** Device state (enabled or disabled) is validated during device policy evaluation
- **Sign-in and user risk:** Azure AD Identity Protection for conditional access risk policies



Azure AD conditional access is, for example, the only option to disable access for Azure through the public internet or based on network policies. Even private connections, such as Microsoft ExpressRoute, do not allow limiting access through a network. They always depend on conditional access through Azure AD.

The following screenshot shows you the overview page for Azure AD conditional access:



I would also recommend you consult the documentation to get a deeper look into the capabilities of this service: <https://docs.microsoft.com/en-us/azure/active-directory/conditional-access/overview>.

Azure AD DS

Another service I want to give some credit to is Azure AD DS. This service builds a bridge to services that depend on AD DS or **Lightweight Directory Access Protocol (LDAP)** and Azure AD. Azure AD DS is an AD **Lightweight Directory Services (LDS)** deployed over Azure AD. It allows read access through common AD DS calls and can handle normal domain identity and access. In the majority of use cases, Azure AD DS can replace a classic AD DS.



For the deployment and more, consult the documentation for this service: <https://docs.microsoft.com/en-us/azure/active-directory-domain-services/active-directory-ds-overview>.

Summary

You have learned how Azure AD works and what features each subscription level of Azure AD offers you. You are now also familiar with creating users in Azure AD and syncing them with your on-premises AD. You also saw other options for user sign-in with AD FS and how to make your Azure hybrid environment highly available.

In the next chapter, we will cover Azure Managed Application details.

Questions

1. What are the four options for Azure AD?
2. Which three options can we choose to enable SSO for Azure AD?
3. Which tool do you need to synchronize users, groups, and devices with Azure AD?
4. Can you deploy Azure AD Connect as highly available?
5. How many days do you have to recover an Azure AD Connect Server?
6. Can the `manage.windowsazure.com` portal still be used to manage Azure AD?
7. Which Azure AD service do you need to limit Azure or Microsoft 365 service access through a network?

Further reading

In the following books, you can find more information about what we learned in this chapter:

- **Windows Server 2016, Hybrid Identity, and Access Management Recipes:** <https://www.packtpub.com/networking-and-servers/windows-server-2016-hybrid-identity-and-access-management-recipes-video>
- **Mastering Identity and Access Management with Microsoft Azure:** <https://www.packtpub.com/virtualization-and-cloud/mastering-identity-and-access-management-microsoft-azure>
- **Implementing Azure Cloud Design Patterns:** <https://www.packtpub.com/virtualization-and-cloud/implementing-azure-cloud-design-patterns>
- **Azure for Architects:** <https://www.packtpub.com/virtualization-and-cloud/azure-architects>

4

Azure Managed Applications

Azure offers a variety of marketplace images which can be deployed to the customers environment. Azure managed applications takes a slightly different approach by offering predefined resources and applications to your internal organization on the internal service catalog or for the public marketplace which are then operated and maintained by the publisher.

This approach enables customers or users within your internal organization with less experience in managing the azure platform to securely and reliably operate offers from third-party or the internal service catalog.

This chapter will cover the following topics:

- Azure managed applications overview
- Azure managed applications architecture
- Creating a managed application definition
- Testing and validation of our template files
- Publishing a managed application to the internal service catalog
- Deploying a managed application from the internal service catalog
- Publishing a managed application to the Azure Marketplace

Technical requirements

For editing files like the templates in this chapter, I recommend using Visual Studio (any edition) or Visual Studio Code which can be obtained at <https://visualstudio.microsoft.com>.

Azure managed applications overview

This part of the chapter is dedicated to give you a brief overview of Azure managed applications, the differences between an internal application in the service catalog and a public Azure Marketplace offer.

History and purpose of Azure managed applications

Azure managed applications were first introduced to Azure in mid 2017 as a preview feature and went to GA on November 2017. As an offering unique to Azure, Azure managed applications focus on customers who want to deploy entire applications without the hassle of managing and maintaining the infrastructure behind those applications.

Azure managed applications offer a great way for **Managed Services Providers (MSPs)**, **Independent Software Vendors (ISVs)** and **System Integrators (SIs)** to deliver their applications with an optional full service component. Those partners are now able, to manage and maintain solutions directly in customer environments.

Scenarios of Azure managed application publishing

Azure managed applications can either be published to the internal service catalog or to the public Azure Marketplace. With that said, there are some key differences between the internal service catalog and the Azure Marketplace offering that can be found at: <https://azure.microsoft.com/en-us/blog/azure-managed-application-in-azure-marketplace-under-the-hood/>.

Although there are some differences in publishing managed applications, the process of creation in both scenarios is mostly the same:

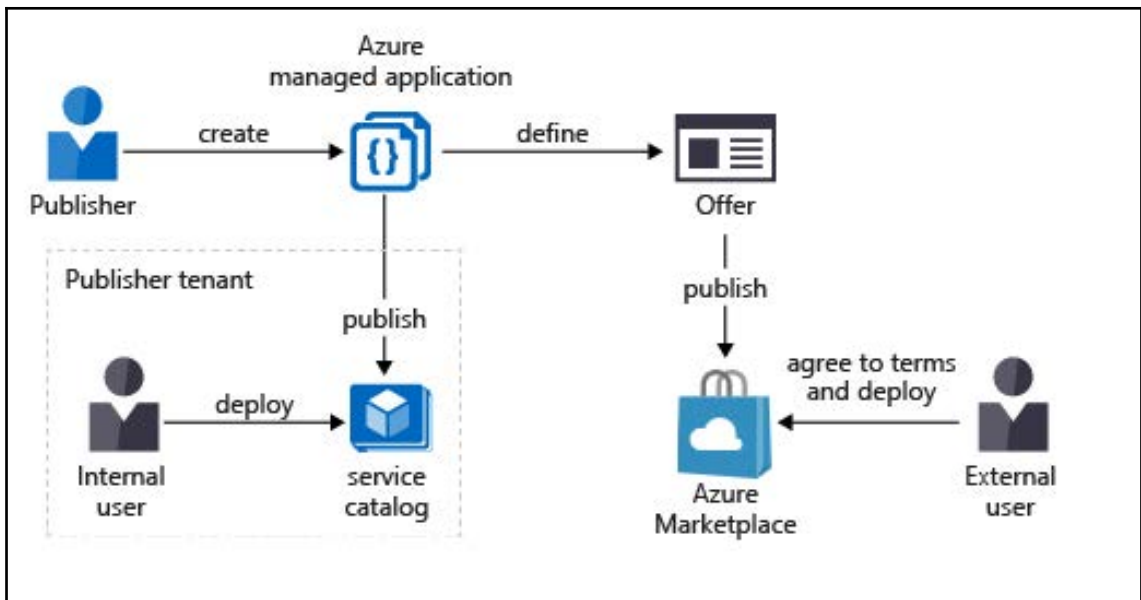


Image source: <https://docs.microsoft.com/en-us/azure/managed-applications/overview>

The publisher, no matter if the publisher is an internal admin or a partner organization, creates an Azure managed application. After the creation, the managed application can be published to the internal service catalog and then be deployed by users within the organization.

For publishing Azure managed applications to the Azure Marketplace, the publisher first needs to create an offer in which he defines some details about the solution. The offer also defines the SKUs and thereby the pricing of the Azure Marketplace Offer. One necessary detail is the support section in which the publisher enables the customer to get support for the application. We will dive deeper into this subject in the *Publishing a managed application to the Azure marketplace* section.

Azure managed applications architecture

This part of the chapter will give you a brief overview of the architectural concept behind Azure managed applications and the key difference to Azure Marketplace solution templates. We will cover the different parts of an Azure managed application and look behind the optional parts that can be deployed using managed applications.

Elements of an Azure managed application

An Azure managed application package consists of two required elements which are then packaged into a `.zip` file:

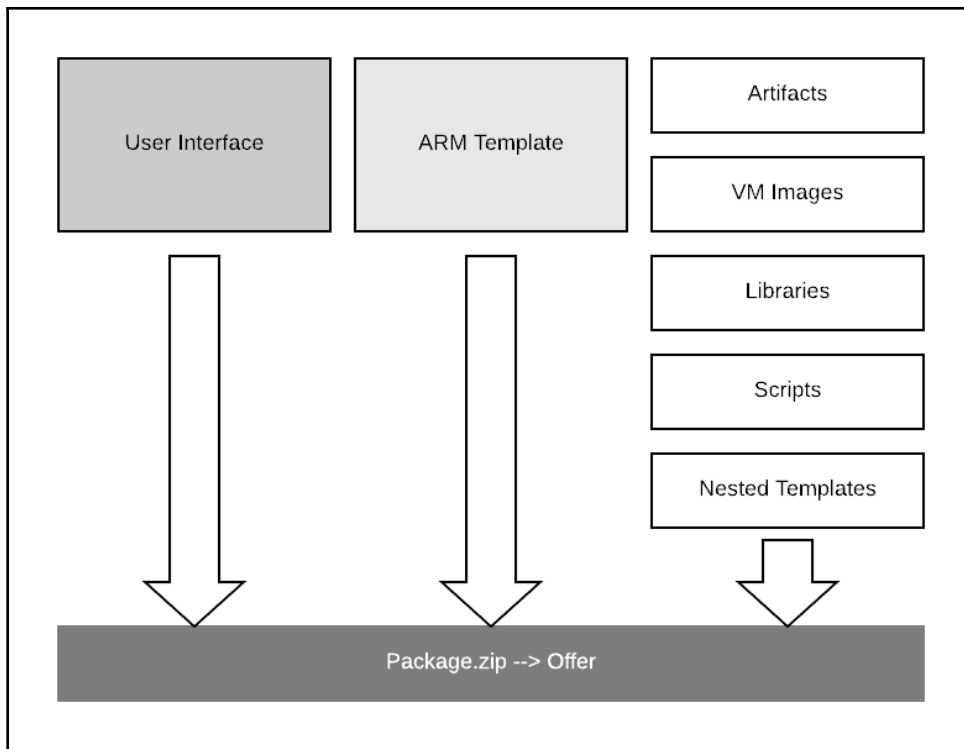
- **Resource template:** The resource template (`mainTemplate.json`) is a `.json` file which has no differences to a standard ARM template as described in Chapter 2, *Azure Resource Manager and Tools*. This file describes the resources and their parameters for deployment.
- **User interface definition:** The user interface definition (`createUiDefinition.json`) is used by the Azure portal to generate the user interface when users are deploying a managed application.



Be aware that the names of these files are case sensitive.

Besides those required parts, a managed application can contain optional elements and files necessary for the ARM template like images, scripts and other artifacts.

Those files are then packaged into a single `.zip` file. The ZIP file must contain the required files on top-level and can contain the optional files (also in folders):



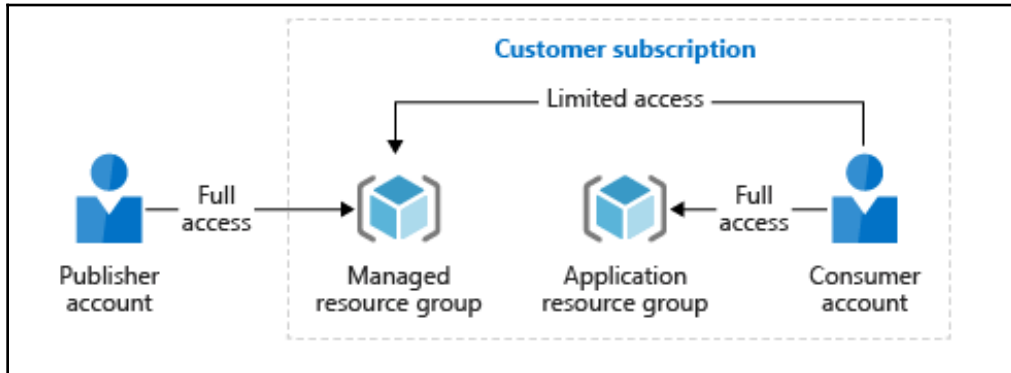
Contents of an Azure managed application definition package

Security and access of an Azure managed application

When an Azure managed application is rolled out to a tenant, it will use two resource groups. The first resource group is considered as **application resource group**, the second as **managed resource group**.

- The application resource group contains the instance of our managed application. The consumer (or internal user) has full access to that resource group for managing the application life cycle. As the user has no access to the resources itself, access to the application resource group is given to gather outputs from the deployment (such as public IP addresses or DNS names) to use the deployed resources (like a VM).

- The managed resource group contains the resources that are required by the Azure managed application itself. Only the specified admins, which are defined when a managed application definition is created, do have write access to this resource group:



Source: <https://docs.microsoft.com/en-us/azure/managed-applications/overview>

While creating the service catalog managed application definition you can either choose to give the user which enrolls your application read-only permission or no permission to the managed resource group. Read-only permission enables the user to view the resources in the managed resource group. We will cover this in detail later in this chapter.

Creating a managed application definition

After we covered the theoretical area of managed applications, let's get more practical in the following part. For your users or customers to roll out either a service catalog managed application or a marketplace managed application, we first need to define the managed application by creating the **app package**.

The app package for the internal service catalog and the marketplace are the same, the difference comes from how these two types of managed applications are published and made available either to your internal users (service catalog managed application) or your customers (marketplace managed application offer). In the previous part we learned, that a managed application consists of two required elements, which we will now discuss in detail.

For this part, I will use the Microsoft managed VM sample for managed applications which can be found at: <https://github.com/Azure/azure-managedapp-samples/tree/master/samples/101-managed-vm>.

This repository is updated and managed by Microsoft. With that said, the code in the repository may change while this book refers to an older version of the code. For that reason, I will also upload the code used in this chapter to our repo at: <https://github.com/PacktPublishing/Implementing-Azure-Solutions-Second-Edition>.

The code from Microsoft is a perfect starting point for anyone who wants to develop an Azure managed application or marketplace template.

The file mainTemplate.json

The first part of a managed application package is made of the resource template (mainTemplate.json).

As described in Chapter 2, *Azure Resource Manager and Tools*, the ARM template itself consists of some required and optional elements. The structure of an ARM template always follows the same markup:

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

Remember: The required parts in this template are `$schema`, `contentVersion` and `resources`. The code which I will use throughout this chapter is located in our repository. We will first have a look at the `mainTemplate.json` file in detail:

In the first two lines, you will find the required values for `$schema` and `contentVersion`:

```
"$schema":
"http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.j
son#",
"contentVersion": "1.0.0.0",
```

The next lines, from line 4 to line 49 contain the parameters for our resources. Parameters are always declared by a name, followed by some definitions:

```
"userName": {
  "type": "string",
  "defaultValue": "azureadmin",
  "metadata": {
    "description": "Specify the OS username"
  }
}
```

The schema of a parameter is defined as follows:

```
"<parameter-name>" : {
  "type" : "<type-of-parameter-value>",
  "defaultValue": "<default-value-of-parameter>",
  "allowedValues": [ "<array-of-allowed-values>" ],
  "minValue": <minimum-value-for-int>,
  "maxValue": <maximum-value-for-int>,
  "minLength": <minimum-length-for-string-or-array>,
  "maxLength": <maximum-length-for-string-or-array-parameters>,
  "metadata": {
    "description": "<description-of-the parameter>"
  }
}
```

Whereas the parameter name and type are required fields, the rest of the definition is optional.

The parameters section is followed by the variables from line 50 to line 58. Variables help you define complex expressions for use throughout your template. In our template, there are only three variables which we use for our **virtual Network ID**, the **subnet reference** and the **image** which is to be deployed:

```
"vnetID": "[resourceId('Microsoft.Network/virtualnetworks', 'vmVnet')]",
"subnetRef": "[concat(variables('vnetID'), '/subnets/',
'subnet1')]",
"osTypeWindows": {
  "imageOffer": "WindowsServer",
  "imageSku": "2016-Datacenter",
  "imagePublisher": "MicrosoftWindowsServer"
}
```

One possible way to define variables is as follows:

```
"variables": {
  "<variable-name>": "<variable-value>",
  "<variable-name>": {
    <variable-complex-type-value>
  },
  "<variable-object-name>": {
    "copy": [
      {
        "name": "<name-of-array-property>",
        "count": <number-of-iterations>,
        "input": {
          <properties-to-repeat>
        }
      }
    ]
  },
  "copy": [
    {
      "name": "<variable-array-name>",
      "count": <number-of-iterations>,
      "input": {
        <properties-to-repeat>
      }
    }
  ]
}
```

So variables can either be just values, which can be created from expressions and functions like `concat` in our sample, or they can be complex types and objects containing one or more properties.



To learn more about variables, visit: <https://docs.microsoft.com/de-de/azure/azure-resource-manager/resource-manager-templates-variables>. To learn more about available functions for an ARM template visit: <https://docs.microsoft.com/de-de/azure/azure-resource-manager/resource-group-template-functions>.

The `resources` section is the main section of our template. Besides the `$schema` and `contentVersion` it is the only mandatory section of the template.

The `resources` section defines, which resources are rolled out by our template. More than that, it defines also which resource has dependencies, and creates an order for deploying the listed resources.

Our template for rolling out a windows VM, defines the following five resources:

- A virtual network
- A **network security group (NSG)**
- A public IP address
- A **network interface card (NIC)**
- A virtual machine

The resources are defined as follows:

For the virtual network:

```
{
  "type": "Microsoft.Network/virtualNetworks",
  "apiVersion": "2017-03-01",
  "name": "vmVnet",
  "location": "[parameters('location')]",
  "dependsOn": [
    "[resourceId('Microsoft.Network/networkSecurityGroups/',
'NSG')]"
    ...
    "[resourceId('Microsoft.Network/networkSecurityGroups/', 'NSG')]"
  ]
}
```

For the NSG:

```
{
  "type": "Microsoft.Network/networkSecurityGroups",
  "apiVersion": "2017-03-01",
  "name": "NSG",
  "location": "[parameters('location')]",
  "properties": {
    "securityRules": [
      {
```

```

        "name": "RDP",
        "properties": {
            ...
            "direction": "Inbound",
            "destinationAddressPrefix": "*",
            "protocol": "Tcp",
            "destinationPortRange": 80,
            "sourcePortRange": "*",
            "priority": 550,
            "sourceAddressPrefix": "*"
        }
    }
]
}
}

```

For the public IP address:

```

{
    "type": "Microsoft.Network/publicIPAddresses",
    "apiVersion": "2017-04-01",
    "name": "[concat(parameters('publicIPAddressName'), 'IP')]",
    "location": "[parameters('location')]",
    "properties": {
        "publicIPAllocationMethod": "Dynamic",
        "dnsSettings": {
            "domainNameLabel": "[toLower(parameters('dnsName'))]"
        }
    }
}

```

For the network interface:

```

{
    "type": "Microsoft.Network/networkInterfaces",
    "apiVersion": "2017-04-01",
    "name": "[concat(parameters('vmNamePrefix'), 'nic')]",
    "location": "[parameters('location')]",
    "dependsOn": [
        "[concat('Microsoft.Network/publicIPAddresses/',
parameters('publicIPAddressName'), 'IP')]",
        "[resourceId('Microsoft.Network/virtualNetworks/',
'vmVnet')]"
    ],
    ...
    "id":
    "[resourceId('Microsoft.Network/publicIPAddresses',
concat(parameters('publicIPAddressName'), 'IP'))]"
    },

```

```

        "subnet": {
            "id": "[variables('subnetRef')]"
        }
    }
}
]
}
}

```

And for the main part, the virtual machine is as follows:

```

{
    "type": "Microsoft.Compute/virtualMachines",
    "apiVersion": "2017-03-30",
    "name": "[concat(parameters('vmNamePrefix'), '-app')]",
    "location": "[parameters('location')]",
    "dependsOn": [
        "[concat('Microsoft.Network/networkinterfaces/',
parameters('vmNamePrefix'), 'nic')]"
    ],
    ...
    "createOption": "FromImage",
    "managedDisk": {
        "storageAccountType": "Standard_LRS"
    },
    "caching": "ReadWrite"
},
{
    "networkprofile": {
        "networkinterfaces": [
            {
                "id":
"[resourceId('Microsoft.Network/networkinterfaces',
concat(parameters('vmNamePrefix'), 'nic'))]"
            }
        ]
    }
}
}

```

All these separate parts depend on each other. For example, the virtual machine cannot be rolled out, while there is no network interface present. For that reason, the virtual machine has a dependency declared, which says it depends on a specific network interface:

```
"dependsOn": [
  "[concat('Microsoft.Network/networkinterfaces/',
    parameters('vmNamePrefix'), 'nic')]"
]
```

You will find dependencies throughout the template. The last section of the template defines our `outputs`. The `outputs` section are values that are returned from a deployment. In an Azure managed application, these outputs are visible to the user deploying the managed application and are an important part of your definition.

In our scenario we will output the **fully qualified domain name (FQDN)** of our VM for the user to make a **Remote Desktop Protocol (RDP)** connection to the machine:

```
"vmEndpoint": {
  "type": "string",
  "value": "[reference(concat(parameters('publicIPAddressName'),
    'IP')).dnsSettings.fqdn]"
}
```

The schema of an output is defined as follows:

```
"outputs": {
  "<outputName>" : {
    "type" : "<type-of-output-value>",
    "value": "<output-value-expression>"
  }
}
```

The `outputs` section are always defined by a type value-pair, where the types that can be used are the same types we use for input parameters.

The file createUiDefinition.json

The second part of a managed application package is made of the user interface definition (`createUiDefinition.json`). If you are deploying an Azure managed application, the user will have to enter some information for the template to be deployed successfully. This information is gathered by the frontend:

The screenshot shows the 'Create Custom VM Managed Application' wizard in the Azure portal. The breadcrumb navigation at the top reads: Home > CustomVM > Create Custom VM Managed Application > Basics. The main title of the wizard is 'Create Custom VM Managed ...'. On the left, there is a vertical list of five steps: 1. Basics (Configure basic settings), 2. VM Credential (Configure the VM credentials), 3. Virtual Machine settings (Configure the virtual machine s...), 4. Endpoint settings (Configure the VM endpoint), and 5. Summary (Custom VM Managed Applicat...). Step 1 is highlighted with a blue background. On the right, the 'Basics' step is active, showing three required fields: 'Subscription' (set to 'Microsoft Partner Network'), 'Resource group' (set to 'Select existing...' with a 'Create new' link below it), and 'Location' (set to 'East US').

Deploying an Azure managed application UI definition

As you can see in the preceding screenshot, the UI design of an Azure managed application deployment is not very different from a marketplace deployment. The design is described in the UI definition file of the managed application.

Like the `mainTemplate.json`, the file `createUiDefinition.json` also has some mandatory and optional elements and follows a predefined markup:

```
{
  "$schema":
    "https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
  "handler": "Microsoft.Compute.MultiVm",
  "version": "0.1.2-preview",
  "parameters": {
    "basics": [ ],
    "steps": [ ],
    "outputs": { }
  }
}
```

The only **optional part** in the `createUiDefinition.json` file is the `$schema` part. The `$schema` is recommended but not necessary for the file to be accepted in a deployment. If you define `$schema` in your file, be aware that the used version of the `$schema` must match the value of the version defined in your template.



In our template, we did include the `$schema`. This is because of the Microsoft testing tools in a later part. The Microsoft testing tools do not accept your template if there is no `$schema` present in the `createUiDefinition.json`!

Let's have a look at our template in more detail and go through the sections included.

The UI definition outline

Generally, the UI template is clustered into one or more steps, which then contain the elements (or inputs and outputs) for the user interface. In the first three lines of the `createUiDefinition.json` file, you will find the specification of the `$schema`, handler and version of our file:

```
"$schema":
"https://schema.management.azure.com/schemas/0.1.2-preview/CreateUIDefinition.MultiVm.json#",
"handler": "Microsoft.Compute.MultiVm",
"version": "0.1.2-preview",
```

The handler for an Azure managed application UI definition is always `Microsoft.Compute.MultiVm`. The latest version is `0.1.2-preview`.

The basics step

The following `parameters` section (from line 4 onwards) is starting with the `basics` step.

```
"basics": [
    {}
]
```

The `basics` step is always the first step of any UI definition file and is the first one generated by the portal. Though we did not declare any elements in the `basics` step, the `basics` step in our definition consists of three elements as you can see in the preceding *Deploying an Azure managed application UI definition* screenshot. Those three elements are automatically added by Azure to the definition to get the minimum amount of information from a user that is needed to deploy any resources in Azure. Those three elements are as follows:

- The subscription
- The resource group to deploy to
- The location of the deployed resources

You do have the opportunity to add user defined elements in the basic section. Be aware, that only elements which are not dependent on either of these three elements are allowed. For example: If you try to add an element of type `Microsoft.Compute.SizeSelector`, this element would not be allowed in the `basics` blade, because the VM sizes depend on the subscription and location and can only be chosen after the selection of this two parameters is made. With some exceptions, only elements from the `Microsoft.Common.*` namespace are allowed in the `basics` step.

Additional steps in the steps section

Following to the `basics` step, the user has the opportunity to define additional steps to gather input from the user. In our template, there are several steps defined.

Let's have a detailed look at the first additional step:

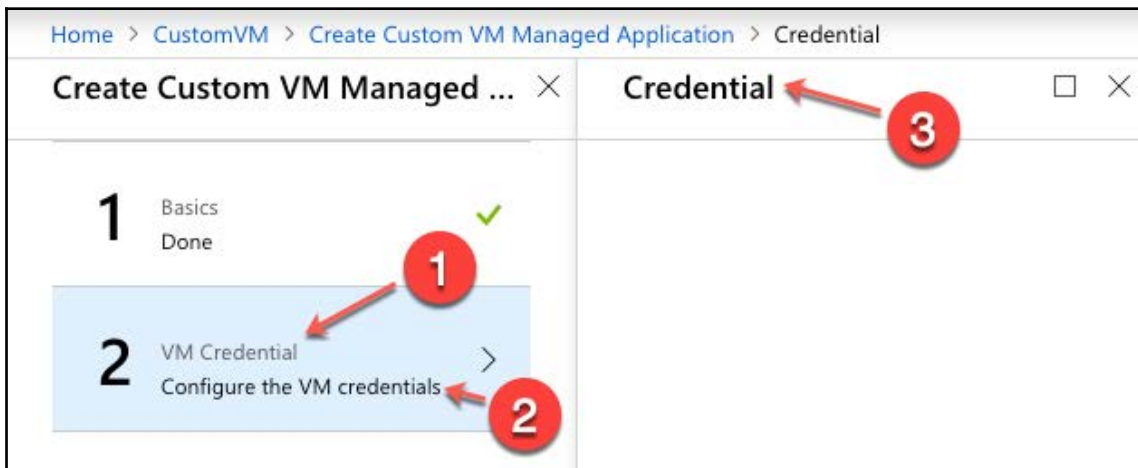
```
"name": "credentialsConfig",
  "label": "VM Credential",
  "subLabel": {
    "preValidation": "Configure the VM credentials",
    "postValidation": "Done"
  },
  ...
},
"osPlatform": "Windows",
```

```
        "constraints": {  
            "required": true  
        }  
    }  
]  
]
```

The steps are defined by the following schema:

```
"name": "stepName",  
  "label": "step label",  
  "subLabel": {  
    "preValidation": "pre-validation label",  
    "postValidation": "post validation label"  
  },  
  "bladeTitle": "blade title",  
  "elements": [  
    {}  
  ]
```

Those parameters of the step, define how the meta information of the step gets displayed in the portal:



Representation of meta information from UI definition in the Azure portal

- The first information in the portal, in the preceding screenshot, with *numberedbullet 1*, is defined in the `label` parameter. This parameter defines the label of our step in the portal.
- The second information in the portal in the preceding screenshot, with *numberedbullet 2* has two functions. When the user currently enters information in this step, the information displayed comes from the `"sublabel"."preValidation"` parameter. After the user has confirmed the input and moves on to the next step, the value displayed comes from the `"subLabel"."postValidation"` parameter.
- The third and last meta information to be displayed in the portal in the preceding screenshot, with *numberedbullet 3* is defined by the `bladeTitle` parameter in our template.

The last section of the Ui definition of a step is formed by the elements. Elements define the input fields and their constraints and limitations as well as how the user enters the information.

In our file, I will use lines 53 to 82 for an explanation of the `elements` section:

```
"elements": [
    {
      "name": "vmNamePrefix",
      "type": "Microsoft.Common.TextBox",
      "label": "Virtual Machine Name prefix",
      "toolTip": "Prefix for the virtual machine",
      "defaultValue": "",
      "constraints": {
        "required": true,
        "regex": "[a-z][a-z0-9-]{2,5}[a-z0-9]$",
        "validationMessage": "Must be 3-5 characters."
      },
      ...
    },
    {
      "constraints": {
        "allowedSizes": [
          "Standard_D1_v2"
        ]
      },
      "osPlatform": "Windows",
      "count": 1
    }
  ]
```

Home > CustomVM > Create Custom VM Managed Application > VM Settings

Create Custom VM Managed ... ✕

VM Settings

1 Basics Done ✓

2 VM Credential Done ✓

3 Virtual Machine settings > Configure the virtual machine s...

* Virtual Machine Name prefix

* Virtual machine size ⓘ >
1x Standard D1 v2

Representation of parameters in the portal

This part of our template is rendered in the portal as shown in the preceding screenshot. As mentioned previously, elements in the Ui definition file are declared having a specific schema:

```
{
  "name": "element1",
  "type": "Microsoft.Common.TextBox",
  "label": "Some text box",
  "defaultValue": "my value",
  "toolTip": "Provide a descriptive name.",
  "constraints": {},
  "options": {},
  "visible": true
}
```

The schema of an element contains required and optional parts. The required values are name, type and label. Although all other parts are optional, some should be included to pass validations and approvals if you are planning to publish your application to the Azure Marketplace. I will go into this more detailed in the *Publishing a managed application to the Azure Marketplace* section.

- The name parameter is used in the outputs and for defining default values of other elements that depend on the input of another parameter.
- The type parameter is a definition of the UI control that should be rendered for the specific element. Depending on the chosen type, the options for validation and supported customization's are different.



The supported element types and their full description can be found at <https://docs.microsoft.com/en-us/azure/managed-applications/create-uidefinition-elements#elements>.

In the mentioned part of our template, we did use two different element types:

- `Microsoft.Common.TextBox`
- `Microsoft.Compute.SizeSelector`

The first element type of `Microsoft.Common.TextBox` has the following schema:

```
{
  "name": "element1",
  "type": "Microsoft.Common.TextBox",
  "label": "Example text box 1",
  "defaultValue": "my text value",
  "toolTip": "Use only allowed characters",
  "constraints": {
    "required": true,
    "regex": "^[a-z0-9A-Z]{1,30}$",
    "validationMessage": "Only alphanumeric characters are allowed, and the value must be 1-30 characters long."
  },
  "visible": true
}
```

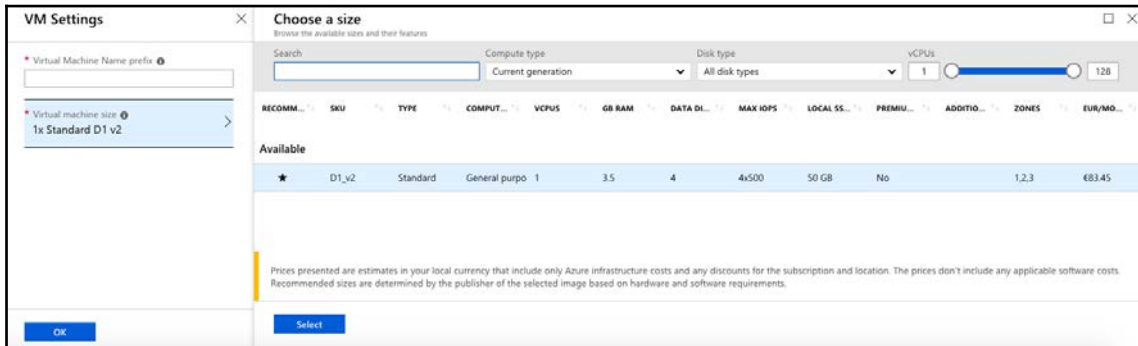
As you can see in the code section of the `Microsoft.Common.TextBox` type sample.

- The defined constraints are one crucial part for elements of this type. The `required` parameter of this element, defines whether a value has to be entered for this element or not. The `required` parameter, if not set, has a default value of `false`.
- The `regex` parameter defines a JavaScript regular expression for determining if a parameter matches the entered regex or not. If the `required` parameter is set to `false` and the `regex` parameter is set to a regular expression, the user must not enter any values for the element, but if he does, the value must follow the regex pattern.
- The `validationMessage` allows to display a hint to the user for entering the correct value for the element.

The second element of type `Microsoft.Compute.SizeSelector` has the following schema:

```
{
  "name": "element1",
  "type": "Microsoft.Compute.SizeSelector",
  "label": "Size",
  "toolTip": "",
  "recommendedSizes": [
    "Standard_D1",
    "Standard_D2",
    "Standard_D3"
  ],
  "constraints": {
    "allowedSizes": [],
    "excludedSizes": [],
    "numAvailabilityZonesRequired": 3,
    "zone": "3"
  },
  "options": {
    "hideDiskTypeFilter": false
  },
  "osPlatform": "Windows",
  "imageReference": {
    "publisher": "MicrosoftWindowsServer",
    "offer": "WindowsServer",
    "sku": "2012-R2-Datacenter"
  },
  "count": 2,
  "visible": true
}
```

As you can see, the second type is very different from our first type. The representation of this type in the portal is shown in the following screenshot:



Representation of the `Mirosoft.Compute.SizeSelector` element in the portal

- Besides the constraints section, which we also saw in the type `Microsoft.Common.TextBox` you get a lot more of customization and configuration options in the type `Microsoft.Compute.SizeSelector` which allow you to configure this element to your needs.
- The parameter `recommendedSizes` allows a configuration of the recommendations on sizing of the virtual machine. In our example, this is set to the `Standard_D1_v2` VM size.
- The constraints section in this type, allows you to configure parameters like `allowedSizes` and `excludedSizes`, to guide the user to choose the right types of VMs for your workloads. Be aware that these two parameters cannot be used simultaneously.
- The `numAvailabilityZonesRequired` and `zones` parameters configure, how many availability zones are required by the region as well as the zone to be used during deployment. This parameter helps in getting a HA deployment of different machines.

In addition to the schema of the first type discussed, we do have some extra sections in this type for defining options and OS image parameters in more detail.

- The options section with the parameter `hideDiskTypeFilter` allows the definition of whether the user is able to switch between the filtering of all disk types or only SSD (premium) disks.
- The `osPlatform` parameter defines the platform for the VM (Windows/Linux) and is used to determine the hardware costs of the VM.

- The `imageReference` is provided for third-party images and determines the software costs of the VM (Linux/Windows).
- The `count` parameter is used to roll out multiple VM's in one step and can be defined either static (min and default value is 1), or dynamically by using an input of a previous step.

All those parts together form the UI representation of our template in the Azure portal.

The outputs section

The `outputs` section of the `createUiDefinition.json` defines the outputs that are then mapped to the template parameters. This section should be considered as the link from your UI to the Azure Resource Manager template.

The `outputs` section is basically a dictionary of `key<value>` pairs, that are put together from static values and the elements of our steps.

The `outputs` section of our template looks as follows:

```
"outputs": {
  "location": "[location()]",
  "vmSize": "[steps('vmConfig').vmSize]",
  "vmNamePrefix": "[steps('vmConfig').vmNamePrefix]",
  "applicationResourceName": "[steps('vmConfig').vmNamePrefix]",
  "userName": "[steps('credentialsConfig').adminUsername]",
  "pwd": "[steps('credentialsConfig').adminPassword.password]",
  "dnsName":
    "[steps('webConfig').dnsAndPublicIP.domainNameLabel]",
    "publicIPAddressName":
      "[steps('webConfig').dnsAndPublicIP.name]"
}
```

As you can see, all necessary parameters of our `mainTemplate.json` are covered by outputs from the `createUiDefinition.json` file.

Testing and validation of our template files

In the previous part of this chapter, we had a deeper look into defining our Azure managed applications template. Because of the complexity and the structure of those templates, Microsoft has decided to give us some help to check our template structure and to test and validate the created template files.

The checks which we are going to see in this part of the chapter are a very important steps in validating our template and getting our template ready for either the Azure internal service catalog or the Azure Marketplace.

Microsoft itself has included these checks into their review of any template submitted for approval, so this is a critical step in delivering an robust Azure managed application template to the marketplace.

Testing and validating the mainTemplate.json file

The testing and validating the `mainTemplate.json` file is as follows:

Validation and testing against Azure

For testing your `mainTemplate.json` file, Microsoft has put together a PowerShell as well as an Azure CLI / Bash script for deploying your artifacts to azure and testing the deployment with a predefined parameters file. Both ways (PowerShell / CLI) have the options to upload the artifacts of the template to azure and to simulate a deployment of the template.

The files needed for this testing can be found at <https://github.com/Azure/azure-quickstart-templates> from the main folder of the repository, download either the `Deploy-AzureResourceGroup.ps1` or the `az-group-deploy.sh` file and place them in a folder of your choice. I will test the files using the PowerShell version of the script, the bash version has some amendments in the switches but is mostly the same.

For Testing it is crucial, that you first login to Azure by starting with an `Add-AzureRmAccount` command to authenticate yourself against Azure for the PowerShell session. After you have successfully logged in, you can use the `Deploy-AzureResourceGroup.ps1` script, to verify your template against Azure.

The `Deploy-AzureResourceGroup.ps1` script needs some parameters to get the validation process started. Those parameters are

- `ArtifactsStagingDirectory` (the directory where your `mainTemplate.json` is located)
- `ResourceGroupLocation` (location to deploy the resources to)

To test your template against Azure, use the following command:

```
.\Deploy-AzureResourceGroup.ps1 -ArtifactStagingDirectory  
.\ManagedAppTemplate\ -ResourceGroupLocation westeurope -ValidateOnly
```

Whereas `ManagedAppTemplate` should be the directory where you located the template files. The output of this command should be:

```
Using parameter file: .\ManagedAppTemplate\.\azuredeploy.parameters.json  
  
Template is valid.
```

If you have an error in your template, the script will give you an output like the following:

```
Using parameter file: .\ManagedAppTemplate\.\azuredeploy.parameters.json  
  
Validation returned the following errors:  
: The specified location '[parameters('location')]' is invalid. A location  
must consist of characters, whitespace, digit, or following symbols '(',')'.  
  
Template is invalid.
```

This indicates, that there is a problem with a location parameter in your template, though the file cannot be validated.

Validation and testing with local scripts

Another possible way of validating your template is to use the scripts provided in the test folder under <https://github.com/Azure/azure-quickstart-templates/tree/master/test/template-validation-tests>. To use these files and test, you first need to install Node.js with npm, but you will find a good installation manual in the repository itself.

After you have installed all necessary files with `npm install`, you are ready to go and validate your template.



If you are using a Mac or Linux system to test your templates, be sure to edit the file `package.json` in the main folder `template-validation-tests` and change lines 7, 9 and 10 by replacing the `\\` with `/` so the referenced files can be found by npm. The files in our repositories are for Windows users and do not include these changes!

The validation itself is very complex and follows the same routine that Microsoft would use to test your marketplace templates against its rules. The template validation can be started by the following command:

```
npm --folder=sample-template run template
```

With the `folder` parameter set to the location of your files. If the test is successful, you should get an output like this:

```
Sebastians-MBP:template-validation-tests sebastianhoppe$ npm --folder=../SimpleManagedTemplate/ run template
Testing template files...

inTemplate.json

template files -
parameters tests -
  ✓ maintemplate.json should have a "parameters" property
  ✓ parameter vmnameprefix that does not have a defaultValue in file mainTemplate.json, must have a corresponding output
in createUiDefinition.json
  ✓ parameter location that does not have a defaultValue in file mainTemplate.json, must have a corresponding output in
createUiDefinition.json
  ✓ parameter vmsize that does not have a defaultValue in file mainTemplate.json, must have a corresponding output in cr
eateUiDefinition.json
  ✓ parameter username that does not have a defaultValue in file mainTemplate.json, must have a corresponding output in
createUiDefinition.json
  ✓ parameter pwd that does not have a defaultValue in file mainTemplate.json, must have a corresponding output in creat
eUiDefinition.json
  ✓ parameter dnsname that does not have a defaultValue in file mainTemplate.json, must have a corresponding output in c
reateUiDefinition.json
  ✓ parameter publicipaddressname that does not have a defaultValue in file mainTemplate.json, must have a corresponding
output in createUiDefinition.json
  ✓ non-null default values must not be provided for secureStrings
  ✓ a parameter named "location" must exist and it must have a defaultValue of resourceGroup().location
  ✓ parameter vmnameprefix must be used in file mainTemplate.json
  ✓ parameter location must be used in file mainTemplate.json
  ✓ parameter vmsize must be used in file mainTemplate.json
  ✓ parameter username must be used in file mainTemplate.json
  ✓ parameter pwd must be used in file mainTemplate.json
  ✓ parameter dnsname must be used in file mainTemplate.json
  ✓ parameter publicipaddressname must be used in file mainTemplate.json
resources tests -
  ✓ location value of resource vmVnet should be an expression or "global"
  ✓ location value of resource NSG should be an expression or "global"
  ✓ location value of resource [concat(parameters('publicIPAddressName'), 'IP')] should be an expression or "global"
  ✓ location value of resource [concat(parameters('vmNamePrefix'), 'nic')] should be an expression or "global"
  ✓ location value of resource [concat(parameters('vmNamePrefix'), '-app')] should be an expression or "global"
  ✓ resourceGroup().location must NOT be used in the template file ../SimpleManagedTemplate/mainTemplate.json.
  ✓ apiVersions must NOT be retrieved using providers().apiVersions[n] in the template file ../SimpleManagedTemplate/mai
nTemplate.json. This function is non-deterministic.
VM
  ✓ VM Image ref must not contain "-preview"

25 passing (20ms)

[mochawesome] Report JSON saved to /.../icatio
ns/Scripts/template-validation-tests/mochawesome-report/mainTemplateTestReport.json

[mochawesome] Report HTML saved to /.../icatio
ns/Scripts/template-validation-tests/mochawesome-report/mainTemplateTestReport.html
```

Template mainTemplate.json test output

Testing and validating the createUiDefinition.json file

Like the testing and validation process of the `mainTemplate.json` file, the `createUiDefinition.json` file can also be tested against the portal itself as well as on the local machine through script.

Validation and testing against Azure

To Validate and test the UI definition against the Azure Portal, a method called `Sideload` can be used. This method gives us the possibility, to test our UI definition in the Azure Portal, as it would look like the user is rolling out our template. The files necessary for this process are also located in the main folder of the `quickstart-templates` GitHub repository at <https://github.com/Azure/azure-quickstart-templates>.

The files needed for this part of the testing are `SideLoad-CreatUiDefinition.ps1` and `sideload-createuidef.sh`. I will also test the files using the PowerShell version of the script, the bash version has some amendments in the switches but is mostly the same.

Like in the previous part, you also need to be logged in to Azure in the PowerShell Window to run the script. When you first run the script `SideLoad-CreatUiDefinition.ps1`, you need to specify the parameter `StorageResourceGroupLocation`. This parameter defines the location of the storage account to be created for uploading the UI definition file. The other parameters all have default values, where the `ArtifactsStagingDirectory` should be specified if the `createUiDefinition.json` file is located in a different directory than the script itself.

The script is started with the following command

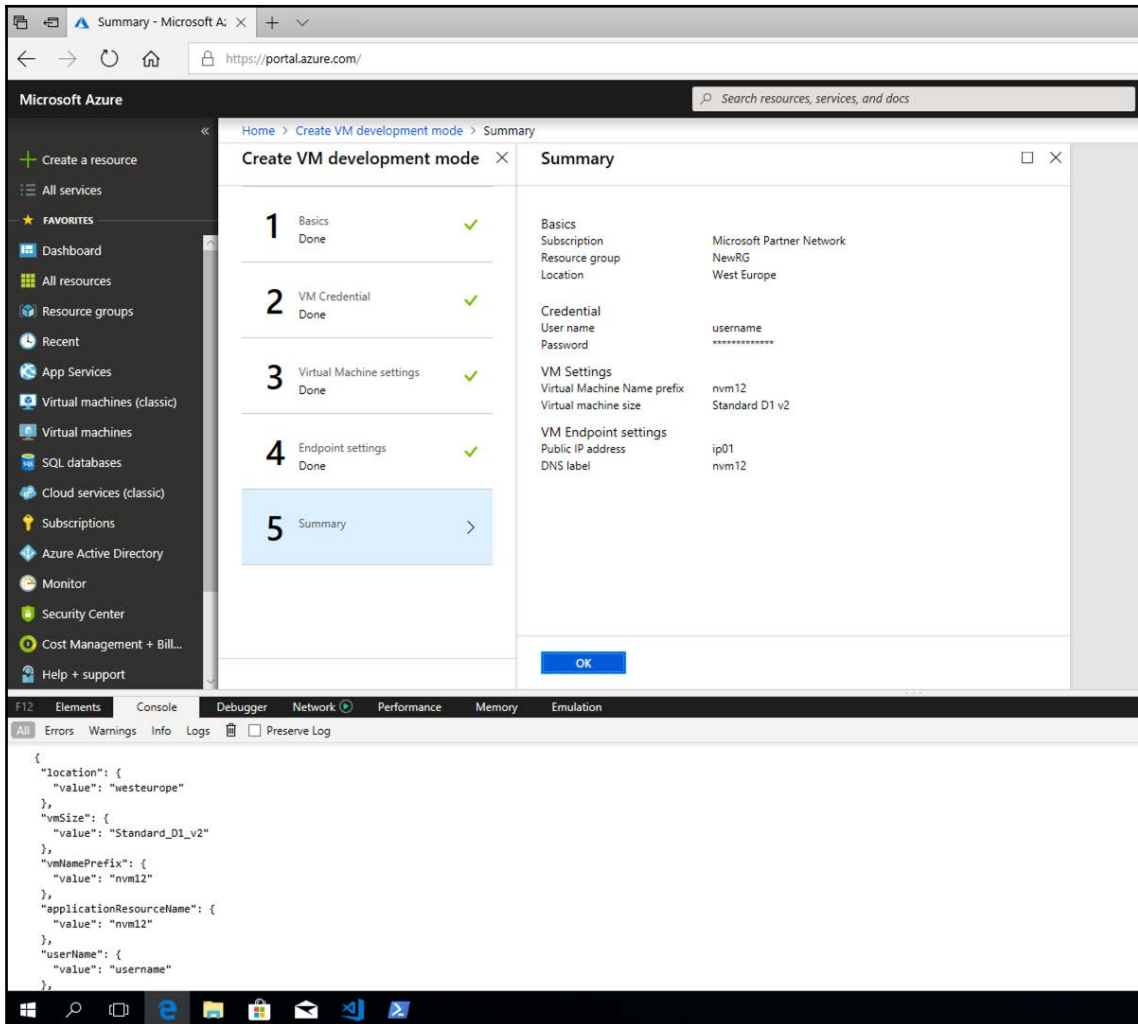
```
.\SideLoad-CreatUiDefinition.ps1 -ArtifactsStagingDirectory
.\ManagedAppTemplate\ -StorageResourceGroupLocation westeurope
```

And should produce the following output:

```
Blob End Point: https://stagea8efc019e8474fa2907.blob.core.windows.net/
Name PublicAccess LastModified ----
https://stagea8efc019e8474fa2907.blob.core.windows.net/createuidef/createUI
Definition.json?sv=2017-07-29&sr=b&sig=TSvlC7GklEjkuxhnhI6x1KrfCnDfhRlZXDgy
ZK7nyiY%3D&se=2018-09-27T19%3A59%3A36Z&sp=r Target URL:
https://portal.azure.com/#blade/Microsoft_Azure_Compute/CreateMultiVmWizard
Blade/internal_bladeCallId/anything/internal_bladeCallerParams/{ "initialDat
```

```
a": {}, "providerConfig": {"createUiDefinition": "https%3A%2F%2Fstagea8efc019e8474fa2907.blob.core.windows.net%2Fcreateuidef%2FcreateUIDefinition.json%3Fs%3D2017-07-29%26sr%3Db%26sig%3DTSv1C7Gk1EjkuxhnhI6x1KrfCnDfhR1zXDgyZK7nyiY%253D%26se%3D2018-09-27T19%253A59%253A36Z%26sp%3Dr"}}}
```

Besides that output, your standard browser should be opened by the script, showing the UI definition which was side loaded to the Azure portal:



Sideloaded UI in the Azure portal with developer console open

As you can see in the preceding screenshot, I also opened the developer console (by pressing *F12* in most browsers), to have an JSON output of my entered parameters. The console will also show any errors (if it hits one).

Validation and testing with local scripts

Like the `mainTemplate.json`, the `createUiDefinition.json` file can also be tested with the scripts located at <https://github.com/Azure/azure-quickstart-templates/tree/master/test/template-validation-tests>.

To test your UI definition against those scripts, simply run the following command:

```
npm --folder=sample-template run createUi
```

With the `folder` parameter set to the location of your files. After running this command, you should get the following output:

```
createUiDefinition.json file -
✓ must have a schema property
✓ handler property value should be 'Microsoft.Compute.MultiVm'
✓ version property value must match schema version
✓ must have parameters and outputs properties
✓ output location must be present in mainTemplate parameters
✓ output vmSize must be present in mainTemplate parameters
✓ output vmNamePrefix must be present in mainTemplate parameters
✓ output applicationResourceName must be present in mainTemplate parameters
✓ output username must be present in mainTemplate parameters
✓ output pwd must be present in mainTemplate parameters
✓ output dnsName must be present in mainTemplate parameters
✓ output publicIpAddressName must be present in mainTemplate parameters
✓ parameters should have basics and steps properties
✓ text box control vmNamePrefix must have a regex constraint
✓ text box control vmNamePrefix must have a validationMessage
✓ location must be in outputs, and should match [location()]

16 passing (15ms)

[mochawesome] Report JSON saved to ns/Scripts/template-validation-tests/mochawesome-report/createUiDefTestReport.json icatio
[mochawesome] Report HTML saved to ns/Scripts/template-validation-tests/mochawesome-report/createUiDefTestReport.html icatio
Sebastians-MBP:template-validation-tests sebastianhoppe$
```

Output of script based test for `createUiDefinition.json`

Publishing a managed application to the internal service catalog

We're now getting to the part, where our template for the managed application is to be published to the internal service catalog.

The internal service catalog is a platform for publication of offers which are designed for users within your tenant. The internal service catalog can contain different managed applications for your users, whereas users can only deploy managed application to which they have been granted access by using RBAC.

Creating the service catalog managed application definition

Creating a service catalog managed application definition is as follows:

[Home](#) > [New](#) > [Service Catalog Managed Application Definition](#) > Service catalog managed application definition

Service catalog managed application definition

Service catalog managed application definition

BASICS

*

Name ⓘ

enter application name

*

Display Name ⓘ

enter application display name

Description ⓘ

enter application description

*

Subscription

MSDN Platforms ▾

*

Resource group ⓘ

☒ Create new

☐ Use existing

*

Location

West Europe ▾

PACKAGE

i

Before you begin, you must have created an Azure managed application and placed all templates into a ZIP file. You will also need to have this package available in Azure blob storage.

For details on creating this package, click here.

↗

*

Package File Uri

AUTHENTICATION AND LOCK LEVEL

*

Lock Level ⓘ

Read Only ▾

Add Authorization

Create

Automation options

Creating a service catalog managed application definition

The service catalog managed application definition can be created in several ways. We will focus on two models:

- Azure portal
- PowerShell

The basics section

The service catalog managed application definition (for shortening I will use `MA-def` as abbreviation) defines your offer in the Internal service catalog. For that reason, you have to define some parameters which will be presented to users rolling out your managed application:

Create service catalog managed application	
Filter by name and description...	
NAME	DESCRIPTION
 Custom VM Managed Application	This Managed Application deploys a custom VM Image

Managed application representation in the internal service catalog

In reference to the *Creating a service catalog managed application definition* screenshot:

- The first parameter defines the name of your managed application. This name is for internal use only and will not be presented in the UI.
- The second parameter defines the display value in the service catalog. This value is shown to your users as the name of your application. The `MA-def` has a display value of **Custom VM Managed Application**.
- The third parameter defines a description. The description has a value of **This Managed Application deploys a custom VM Image**.
- The further parameters define the **Subscription**, **Resource group** and **Location** of your `MA-def`. Those parameters are mandatory as for all Azure resources.

The package section

The package section, with its only parameter `PackageFileUri` defines the location of your app package. To build your application package, you have to ZIP all files created for your Azure managed application and to upload this zip file to Azure Blob Storage.



The `.zip` file must contain only 2 files at top level: `mainTemplate.json` and `createUiDefinition.json`.

All other files need to be included in the specific subfolders. Please keep in mind that some archiving software (like on macOS) does include additional meta files on top level. If you get an error while deploying your package, check your ZIP file for those issues.

The file uploaded to Azure Blob Storage has to get a public access level of blob for the Azure portal to download the file during creation of your `MA-def`.

The authentication and lock level section

In the **AUTHENTICATION AND LOCK LEVEL** section, you need to define who has access to your resources, as well as what the users deploying your managed application can view in their portal. The first **Lock Level** parameter is used to define the access level of users deploying your managed application. There are two options from which you can choose:

- **Read Only**
- **None**

There is no option to grant more rights than **Read Only** to users deploying your managed application, as the definition of an Azure managed application is that the publisher maintains the application. If you plan to grant your users a higher access level, you should consider an Azure Marketplace offer.

The table beneath the **Lock Level** parameter defines the roles and members, which have more access to your managed application:

Home > New > Service Catalog Managed Application Definition > Service catalog managed application definition > Add Authorization > Add Authorization

Service catalog managed application definition

Service catalog managed application definition

enter application name

enter application display name

enter application description

MSDN Platforms

Create new Use existing

CustomVM

West Europe

PACKAGE

Before you begin, you must have created an Azure managed application and placed all templates into a ZIP file. You will also need to have this package available in Azure blob storage.

For details on creating this package, click here.

Package File Uri

AUTHENTICATION AND LOCK LEVEL

Lock Level

Read Only

Add Authorization

ROLE NAME

MEMBER NAME

EMAIL

No Data

CreateAutomation options

Add Authorization

Service catalog managed application definition

Roles

Owner

Members or groups

Select

NAME

OBJECTID

EMAIL

No Data

OK

Add Authorization

Service catalog managed application definition

Invite

Select

Search by name or email address

MAManagedAdmins

Selected

ManagedAdmins

Select

Definition of users and groups for access to the managed application

You can define all preconfigured roles available in Azure and give access to users, as well as groups. For a real-world scenario, there should be no use to configure a single user as an Azure managed application admin—otherwise than for testing purposes.

As you can remember, we talked about the different resource groups that get created during a deployment of an managed application. The first group holds the application itself, as the second resource group holds the resources needed to run the application. Both, lock level and authorization are targeting the latter one for the resources needed to run the application.

After you have completed all parameters and fields, you can click **Create** to deploy the MA-def to the internal service catalog:

[165]

Service catalog managed application definition

Service catalog managed application definition

BASICS

*

Name ⓘ

MyAppDef

✓

*

Display Name ⓘ

My Cool Managed App

✓

Description ⓘ

This is my cool Managed Application

✓

*

Subscription

MSDN Platforms

▼

*

Resource group ⓘ

☒ Create new

☐ Use existing

ManagedAppRG

✓

*

Location

West Europe

▼

PACKAGE

ⓘ

Before you begin, you must have created an Azure managed application and placed all templates into a ZIP file. You will also need to have this package available in Azure blob storage.

For details on creating this package, click here.

🔗

*

Package File Uri

https://managedappstore.blob.core.windows.net/mapp/...

✓

AUTHENTICATION AND LOCK LEVEL

*

Lock Level ⓘ

Read Only

▼

Add Authorization

ROLE NAME	MEMBER NAME	EMAIL
Owner	ManagedAdmins	

Create

Automation options

MA-def with all values included

To deploy your managed application definition through PowerShell, you first need to define some variables and create a resource group to deploy to in your PowerShell script.



The script is included in our repository on GitHub as `Deploy-AzureManagedAppDefinnition.ps1`.

First we create the resource group by the following command:

```
New-AzureRmResourceGroup -Name ManagedAppRG -Location westeurope
```

After creating the resource group, we need to get the ID of our Azure AD group which should be given the owner role on the managed application resources. In my case the group name is `ManagedAdmins`:

```
$managergroupid=(Get-AzureRmADGroup -SearchString ManagedAdmins).Id
```

Then we also need the ID of the `owner` role, to give this role as access level to our group `ManagedAdmins` on the resource group holding our managed application resources:

```
$roleidowner=(Get-AzureRmRoleDefinition -Name Owner).Id
```

The last part of our PowerShell script creates the managed application definition itself:

```
New-AzureRmManagedApplicationDefinition `  
  -Name "MyAppDef" `  
  -Location "westeurope" `  
  -ResourceGroupName ManagedAppRG `  
  -LockLevel ReadOnly `  
  -DisplayName "My Cool Managed App" `  
  -Description "This is my cool Managed Application" `  
  -Authorization "${managergroupid}:$roleidowner" `  
  -PackageFileUri  
  "https://managedappstore.blob.core.windows.net/mapp/app.zip"
```

The output of this PowerShell script should look as follows:

```

Name                               : MyAppDef
ResourceId                         : /subscriptions/.....-...-...-...-
...../resourceGroups/ManagedAppRG/provid
ers/Microsoft.Solutions/applicationDefinitions/MyAppDef
ResourceName                       : MyAppDef
ResourceType                       : Microsoft.Solutions/applicationDefinitions
ResourceGroupName                 : ManagedAppRG
Location                           : westeurope
SubscriptionId                     : .....-...-...-...-.....
Properties                         : @{isEnabled=True; lockLevel=ReadOnly;
displayName=My Cool Managed App;
                                description=This is my cool Managed
Application; authorizations=System.Object[];
                                artifacts=System.Object[]}
ManagedApplicationDefinitionId : /subscriptions/.....-...-...-...-
...../resourceGroups/ManagedAppRG/provid
ers/Microsoft.Solutions/applicationDefinitions/MyAppDef

```

Deploying a managed application from the internal service catalog

We're now going to deploy our newly created managed application from our internal service catalog. This is the part, where your users start to take advantage of your created managed application offerings.

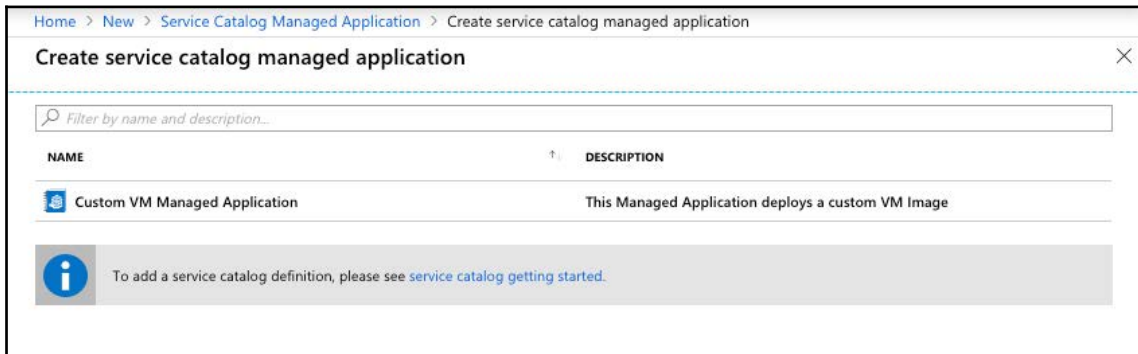
The deployment can be initiated by using the Azure portal. Before your users can deploy your managed application, you have to grant the at least the read rights to your managed application service definition.

For this part of the chapter, I will stick to our example files in the repository and deploy then into a new resource group.

The deployment process

The deployment process for your users is as follows

1. The user logs in to the Azure portal and clicks on **Create a resource** to search for **Service Catalog Managed Application**:



Service catalog managed applications overview

2. After the user clicks on **Create**, the defined values for our managed application have to be entered by the user. As of the type of deployment, the chosen resource group has to be empty (because of the nature of the managed application permissions):

Create Custom VM Managed ... X

1 Basics
Configure basic settings >

2 VM Credential
Configure the VM credentials >

3 Virtual Machine settings
Configure the virtual machine ... >

4 Endpoint settings
Configure the VM endpoint >

5 Summary
Custom VM Managed Applicat... >

Basics □ X

Subscription
Microsoft Partner Network ▼

* Resource group ⓘ
Select existing... ▼
[Create new](#)

* Location
East US ▼

Managed application deployment basic parameters

3. After entering the values of the parameters of the **Basics** section, the user has to enter the credentials for our VM deployment:

The screenshot displays a multi-step wizard titled "Create Custom VM Managed ...". The left sidebar lists five steps: 1. Basics (Done, marked with a green check), 2. VM Credential (selected, highlighted in blue, with a right arrow), 3. Virtual Machine settings (with a right arrow), 4. Endpoint settings (with a right arrow), and 5. Summary (with a right arrow). The main panel on the right is titled "Credential" and contains three input fields, each preceded by a red asterisk and a help icon: "User name", "Password", and "Confirm password".

Entering the parameters for the credentials step

4. Once this step is completed, the values for our VM settings have to be entered:

The screenshot displays the Azure portal interface for creating a custom VM managed application. On the left, a wizard titled 'Create Custom VM Managed ...' shows five steps: 1. Basics (Done), 2. VM Credential (Done), 3. Virtual Machine settings (highlighted), 4. Endpoint settings, and 5. Summary. The 'Virtual Machine settings' step is selected, leading to the 'VM Settings' configuration page on the right. This page contains two required fields: 'Virtual Machine Name prefix' and 'Virtual machine size'. The 'Virtual machine size' is currently set to '1x Standard D1 v2'.

Step	Section	Status
1	Basics	Done
2	VM Credential	Done
3	Virtual Machine settings	Active
4	Endpoint settings	Pending
5	Summary	Pending

VM Settings

- * Virtual Machine Name prefix
- * Virtual machine size **1x Standard D1 v2**

VM settings during deployment

5. The VM settings are then followed by the endpoint configuration for our public IP address:

The screenshot displays the Azure portal interface for creating a custom VM managed application. On the left, a wizard titled 'Create Custom VM Managed Application' shows five steps: 1. Basics (Done), 2. VM Credential (Done), 3. Virtual Machine settings (Done), 4. Endpoint settings (selected, with a right arrow), and 5. Summary (Custom VM Managed Application...). The right pane, titled 'VM Endpoint settings', shows the configuration for a public IP address. It includes a 'Public IP address' field with the value '(new) ip01' and a 'DNS label' field with the value 'westeurope.cloudapp.azure.com'. A red error bar is visible below the DNS label field, indicating a validation issue.

Step	Section	Status
1	Basics	Done
2	VM Credential	Done
3	Virtual Machine settings	Done
4	Endpoint settings	Configure the VM endpoint
5	Summary	Custom VM Managed Application...

VM Endpoint settings

* Public IP address (new) ip01

* DNS label westeurope.cloudapp.azure.com

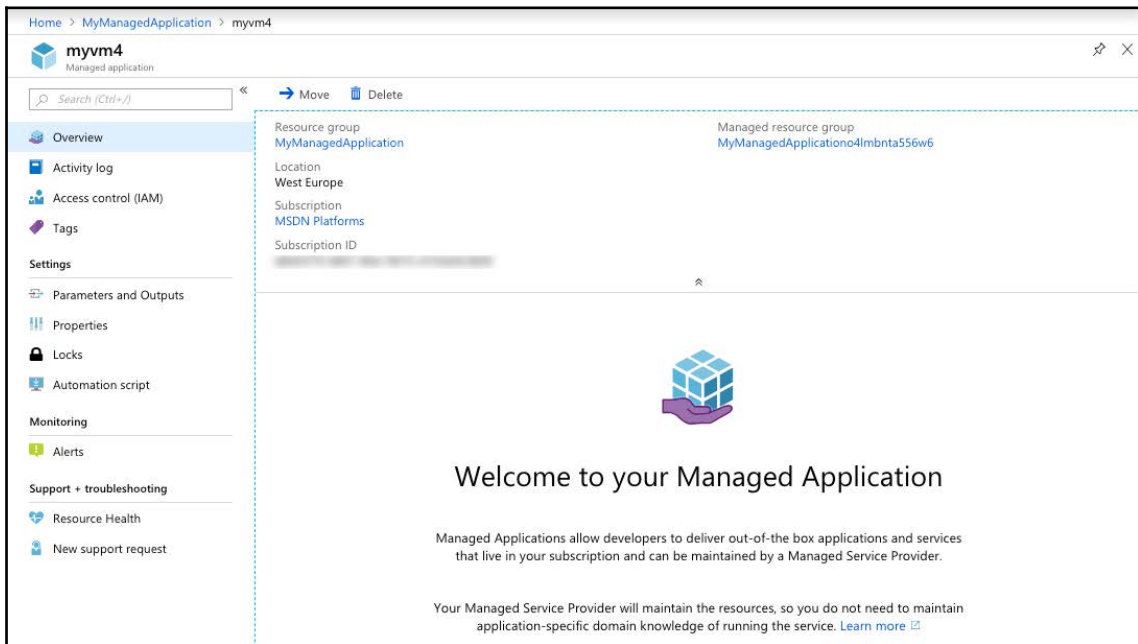
Endpoint settings for the public IP address

- After all values have been entered, a final validation verifies that the entered values are fine and that the template for the managed application can be deployed:

Create Custom VM Managed ...		Summary	
1 Basics Done		Validation passed	
2 VM Credential Done		Basics	
3 Virtual Machine settings Done		Subscription: MSDN Platforms	
4 Endpoint settings Done		Resource group: MyManagedApplication	
5 Summary Custom VM Managed Applicat...		Location: West Europe	
		Credential	
		User name: testuser	
		Password: *****	
		VM Settings	
		Virtual Machine Name prefix: myvm4	
		Virtual machine size: Standard D1 v2	
		VM Endpoint settings	
		Public IP address: ip01	
		DNS label: mymanagedapp	
		OK Download template and parameters	

Final summary for the deployment of the managed application

7. After creating the managed application, you should be able to view your managed application as follows:



Successful deployed Azure managed application

The defined outputs, such as service endpoints or any other necessary information, can be obtained by the **Parameters** and **Outputs** section.

This section, shown in *Final summary for the deployment of the managed application* screenshot, shows all values that were entered for the deployment as well as all outputs generated during deployment:

The screenshot shows the Azure portal interface for a managed application named 'myvm4'. The left sidebar contains navigation links: Overview, Activity log, Access control (IAM), Tags, Settings (Parameters and Outputs, Properties, Locks, Automation script), Monitoring (Alerts), and Support + troubleshooting (Resource Health, New support request). The main content area is titled 'myvm4 - Parameters and Outputs' and displays a search bar and a list of parameters and outputs.

7 parameters

NAME	TYPE	PARAMETER
vmNamePrefix	String	myvm4
location	String	westeurope
vmSize	String	Standard_D1_v2
userName	String	testuser
pwd	SecureString	*****
dnsName	String	mymanagedapp
publicIpAddressName	String	ip01

1 output

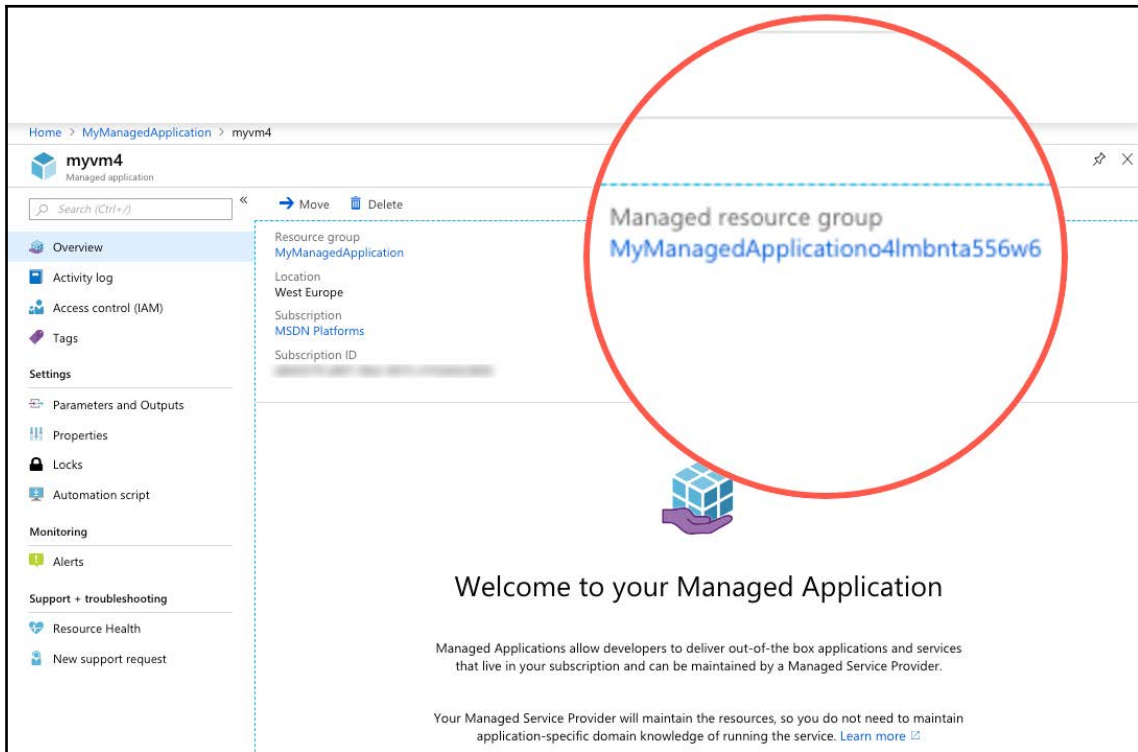
NAME	TYPE	OUTPUT
vmEndpoint	String	mymanagedapp.westeurope.cloudapp.azure.com

Parameters and outputs section of a deployed managed application

The managed resource group

As you saw in the previous section, each deployment of an Azure managed application generates a managed resource group for its resources which are managed and maintained by the author.

This resource group is shown in the managed application resource itself under **Managed resource group** as shown in the following screenshot:



Link to managed resource group after deployment of a managed application

If you open the resource group, you will see that this group contains all resources the managed application is made of:

The screenshot shows the Azure portal interface for a managed resource group named 'MyManagedApplication04lmbnta556w6'. The left-hand navigation pane includes sections for Overview, Activity log, Access control (IAM), Tags, Events, Settings (with sub-items like Quickstart, Resource costs, Deployments, Policies, Properties, Locks, and Automation script), and Monitoring (with sub-items like Insights (preview), Alerts, Metrics, Diagnostic settings, and Advisor recommendations). The main content area shows the 'Overview' tab for the resource group. It displays the subscription (MSDN Platforms), subscription ID, and deployment status (1 Succeeded). Below this, there are filters for name, type, location, and group. A table lists 6 items:

NAME	TYPE	LOCATION
ip01IP	Public IP address	West Europe
myvm4-app	Virtual machine	West Europe
myvm4nic	Network interface	West Europe
myvm4-osDisk	Disk	West Europe
NSG	Network security group	West Europe
vmVnet	Virtual network	West Europe

Resources in the managed resource group

The user who deployed the managed resource will either have no access or read only access to this group, as defined in the managed application definition.

Publishing a managed application to the Azure marketplace

As mentioned in the beginning of this chapter, although the process of creating a managed application is mostly the same, publishing an Azure managed application to the Azure marketplace has some key differences and obstacles.

The Azure marketplace offers users a variety of images from either Microsoft or third-party vendors to choose from. Those images can be deployed with minimal effort and present a very simple solution for building up infrastructure with predefined configuration.

Although using this marketplace offerings is quite simple, the publishing process includes several steps to verify, that the published solution meets the requirements of an Azure marketplace offering.

Please keep in mind, that we are discussing an Azure managed application marketplace offer which has some key differences compared to an Azure marketplace solution.

Technical and business requirements

The technical requirements of your Azure marketplace managed application offering are no different than those of an Azure managed application published to the internal service catalog.

If you worked through the *Testing and validation of our template files* section in this chapter, your technical solution should meet the minimum requirements of an Azure marketplace offering.



To check that you have developed a stable and robust ARM template, please refer to the contribution best practices guide for ARM templates which can be found

at <https://github.com/Azure/azure-quickstart-templates/blob/master/1-CONTRIBUTION-GUIDE/best-practices.md>.

In addition to the technical requirements, there are several business requirements for publishing your application:

- Your company or its subsidiary must be located in a country where sales are supported by the marketplace.
- Your product must be licensed in a way that is compatible with billing models supported by the marketplace.
- Make technical support available to customers in a commercially reasonable manner. The support can be free, paid, or through community support.
- License your software and any third-party software dependencies.
- Provide content that meets criteria for your offering to be listed in the marketplace and in the Azure portal.

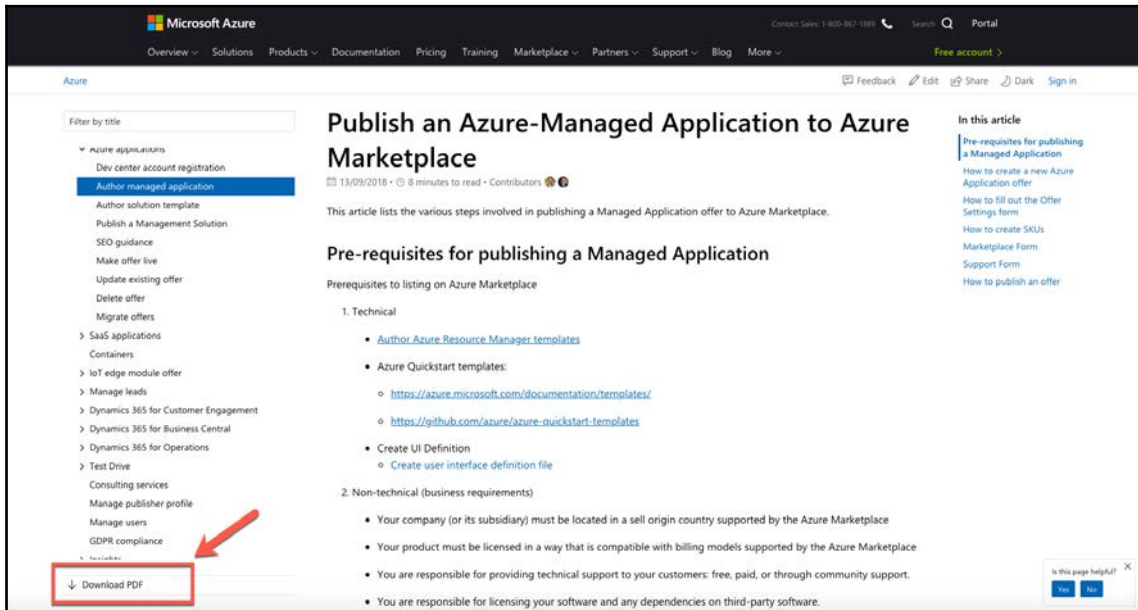
- Agree to the terms of the Azure marketplace participation policies and publisher agreement.
- Agree to comply with the terms of use, Microsoft Privacy Statement, and Microsoft Azure Certified Program Agreement. For more information, refer to: <https://docs.microsoft.com/en-us/azure/managed-applications/publish-marketplace-app>.

To become a publisher in the Azure marketplace, you have to take some further steps for getting access to the publisher portal.

1. First of all, you have to create a Microsoft ID with an email address that belongs to your company and which is not bound to an individual. This account will be granted access to the Microsoft Developer Center and the Cloud Partner portal.
2. The second step is to get in touch with the marketplace onboarding team. This can be done through the Azure Marketplace Nomination Form which can be found at <https://aka.ms/ampnomination>. In the form, select **Managed application** for the question **Solution that you intend to publish**. After the marketplace on-boarding team has reviewed your request, you will receive a voucher code to avoid a registration fee for the Developer Center for which you have to pay otherwise 99\$.
3. The third step is the registration in the Microsoft Developer Center. Microsoft will validate your company by verifying your tax ID and other information. This process can take up to 10 business days to complete.
4. You may now sign in to the Cloud Partner Portal and take the last step to complete your registration—associate your developer account with the marketplace publisher profile.

After these steps, you should be able to publish your managed application to the Azure Marketplace.

To complete the publishing process, and as the Cloud Partner Portal may undergo some changes in UI and handling during the lifetime of this book, I will refer to the Microsoft publishing guide which can be found here at <https://docs.microsoft.com/en-us/azure/marketplace/cloud-partner-portal-orig/cloud-partner-portal-managed-app-publish> or as PDF version here from the link below the article as shown in the following screenshot:



Link for downloading the PDF from the Azure Docs

After you submit your managed application marketplace offering, Microsoft will check your files against the internal publishing guidelines and against automated tests which are in many details the same tests as mentioned in the *Testing and validation of our template files* section.

Microsoft will assist you in the process of getting your solution into the marketplace and will give you feedback on

- **QoS:** How stable is your template and design in terms of quality?
- **Security:** Do you have any security issues in your template (SSH keys, plaintext password)?
- **Code quality:** How well designed and written is your code in terms of code quality?



Please keep in mind that the first two points will, if they do not pass the testing of Microsoft, prevent your solution from being published.

Summary

In this chapter we went through the whole process from an architectural overview of the Azure managed applications services, to the creation of an Azure managed application definition, up to testing and publishing your managed application to the internal service catalog as well as to the Azure marketplace.

You should now be able to create, test and maintain an Azure managed application and to deploy it, whether for your internal users or as a public offering on the Azure marketplace.

Questions

After this chapter, you should be able to answer the following questions:

1. Which tools can be used to publish an Azure service catalog managed application?
2. Which two files are mandatory to create an Azure managed application definition?
3. How many resource groups are used during the deployment of an Azure managed application?
4. What is the purpose of the `mainTemplate.json` file in an Azure managed application definition package?
5. Which three elements are automatically injected in the basics step of a UI definition?
6. How is the method of testing your UI definition in the Azure portal called?
7. What are the possible lock levels of an Azure managed application definition and what is the purpose of this setting?

Further reading

For further reading on the topics of this chapter please refer to the following resources:

- **Azure Marketplace Publishing Guide:** <https://docs.microsoft.com/en-us/azure/marketplace/marketplace-publishers-guide>
- **Azure Managed Applications Documentation:** <https://docs.microsoft.com/en-us/azure/managed-applications/overview>

5 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 topics:

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

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

Azure networking limits

Microsoft offers a wide range of capabilities when it comes to networking in and to Azure. That makes a network solution very flexible but also complex. You have many options to achieve your goal, but to do so you need to keep some limitations in mind. Behind the following link, you can find those Azure limitations: <https://docs.microsoft.com/en-us/azure/azure-subscription-service-limits>.

Azure networking components

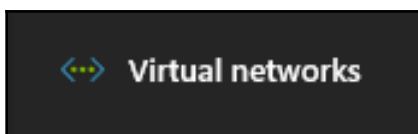
To start with Azure networking, you need to know and understand the components which are needed to set up an Azure solution.

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:

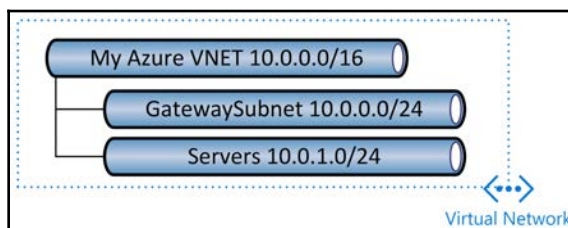


Normally you're setting up a network like you do in your on-premises network. You create a network with an IP range such as `10.0.0.0/16` and split it up into different subnetworks. Every Azure VNet has at least a minimum of two subnets. The first is the **gateway subnet**, which is basically a router 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. I 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.

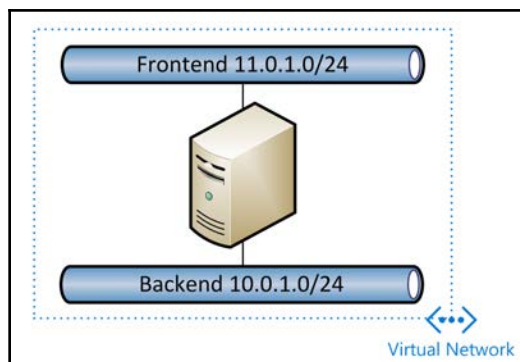


As of September 2016, Azure started to support IPv6 to be used in Azure. The deployment and support of IPv6 is still in progress while writing the book.

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:

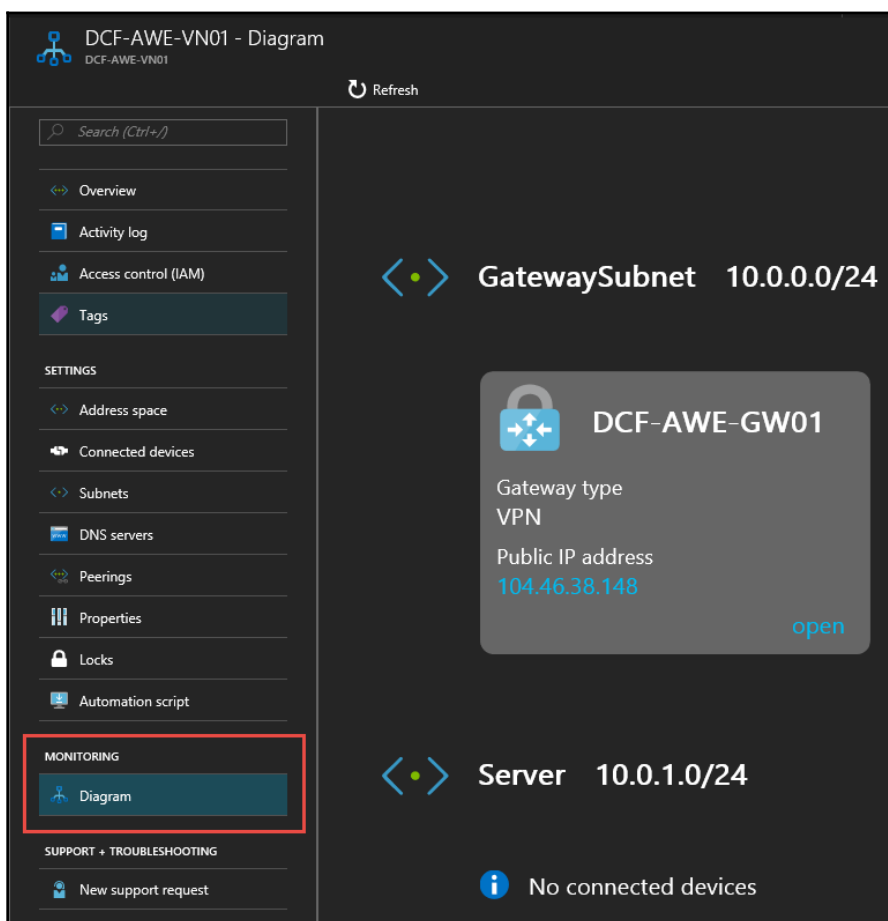


Currently there are only two ways to resolve that issue:

- The first one is to create two subnetworks and put a virtual machine with two network adapters in both networks and route within the virtual machine and prevent default routing with configuration of route filters
- 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 and global VNet peering

There is also an option to connect different Azure VNet works in the same and between Azure region. This option is called Azure VNet peering for VNetworks in the same Azure region and global VNet peering for inter region peering. VNet peering for Azure virtual network lets you directly link two virtual networks 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, I will show you how to set up VNet peering.



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

VNet service endpoints

VNet service endpoints are an extension of a virtual network private address space and a VNet to connect to Azure PaaS services through a direct connection through the Azure backbone to those services. Those private connections and endpoints allow services, virtual machines or users to secure critical and access Azure PaaS service resources only from a virtual network. Traffic from a VNet to the Azure service remains on the Microsoft Azure backbone network at any time, even if the communication between the endpoint and VNet is established through public IPs.

Currently VNet endpoints are GA for following services:

- **Azure Storage:** Generally available in all Azure regions
- **Azure SQL Database:** Generally available in all Azure regions
- **Azure Database for PostgreSQL server:** Generally available in Azure regions where database service is available
- **Azure Database for MySQL server:** Generally available in Azure regions where database service is available

- **Azure Cosmos DB:** Generally available in all Azure public cloud regions
- **Azure Key Vault:** Generally available in all Azure public cloud regions

Following services are currently in preview:

- **Azure SQL Data Warehouse:** Available in preview in all Azure public cloud regions
- **Azure Service Bus:** Available in preview
- **Azure Event Hubs:** Available in preview
- **Azure Data Lake Store Gen 1:** Available in preview

Azure VPN gateways

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

An Azure gateway can serve different purposes:

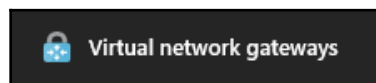
- Internet gateway
- Site-to-site VPN gateway
- Point-to-site VPN gateway
- ExpressRoute gateway
- VNet-to-VNet gateway



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.

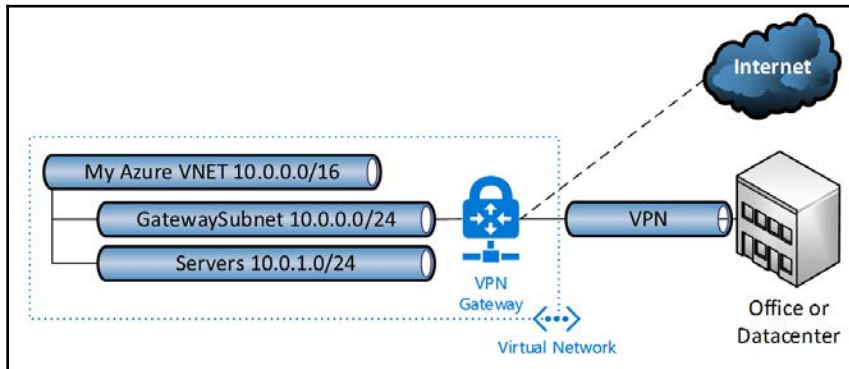
The following table shows a short summary:

	VPN gateway throughput	VPN gateway max IPSEC tunnels	Active - Active VPN	ExpressRoute gateway throughput	VPN gateway and ExpressRoute coexist	Zone redundant
Standard	100 Mbps	10	No	1000 Mbps	Yes	No
High Performance	200 Mbps	30	Yes	2000 Mbps	Yes	No
Ultra High Performance	200 Mbps	30	Yes	9000 Mbps	Yes	No
VpnGw1	650 Mbps	30	Yes	No	No	No
VpnGw1AZ	650 Mbps	30	Yes	No	No	Yes
VpnGw2	1 Gbps	30	Yes	No	No	No
VpnGw2AZ	1 Gbps	30	Yes	No	No	Yes
VpnGw3	1,25 Gbps	30	Yes	No	No	No
VpnGw3AZ	1,25 Gbps	30	Yes	No	No	Yes
ErGw1AZ	No	No	No	1000 Mbps	Yes	Yes with separated VPN gateway
ErGw2AZ	No	No	No	2000 Mbps	Yes	Yes with separated VPN gateway
ErGw3AZ	No	No	No	9000 Mbps	Yes	Yes with separated VPN gateway

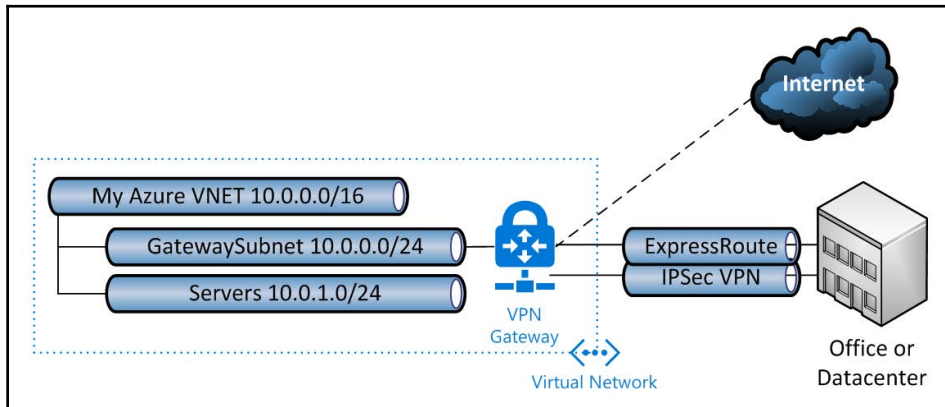


Since Ignite 2018, Microsoft extended his offering around network gateways. The address customer needs regarding better SLAs on gateways, they started to offer Zone-redundant virtual network gateways for ExpressRoute and VPN. Those gateways are placed into different Azure data center with separated power supply, cooling and datacenter environments. That prevents those gateways from datacenter outages and failures. Those Gateways are marked with AZ within der SKU Friendly Name.

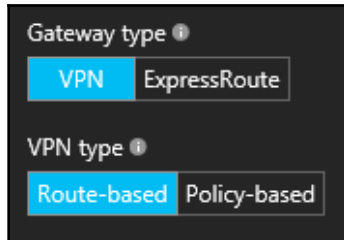
The following diagram shows how the basic **VPN gateway** is connected to your Azure network:



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. I will explain that 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.

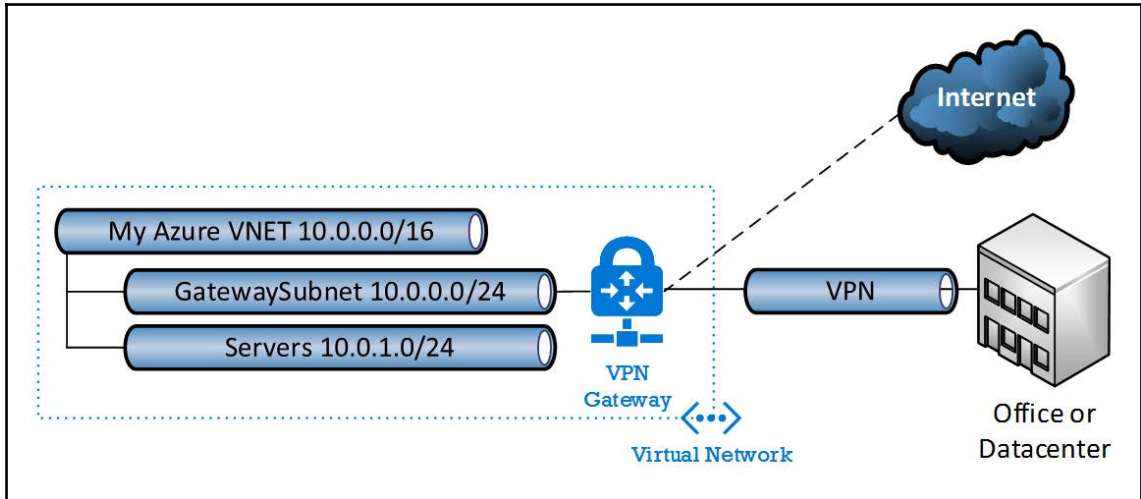
The decision as to which VPN type you need must be done based on your on-premises VPN device. Not every device can speak **Route-based** VPN. Microsoft has published a list of supported devices. You can see them here at <https://azure.microsoft.com/en-us/documentation/articles/vpn-gateway-about-vpn-devices/>.

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

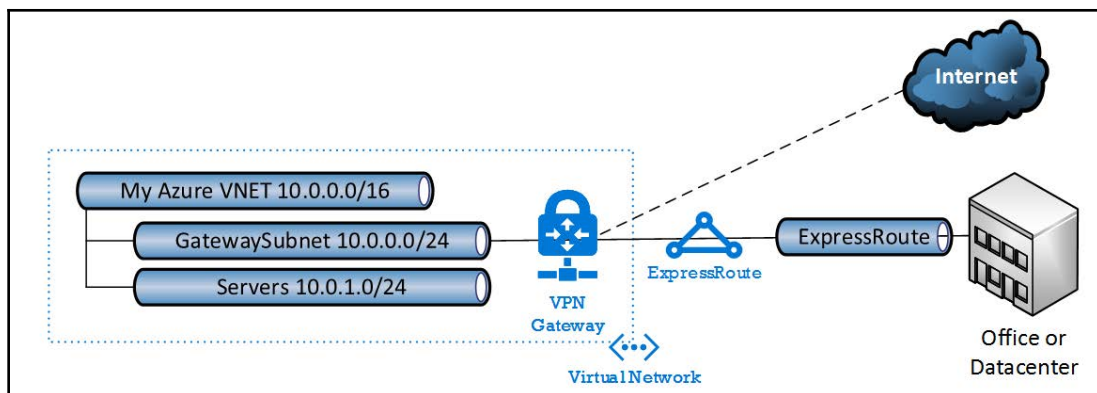
In summary, you can basically have the following gateway configurations:

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

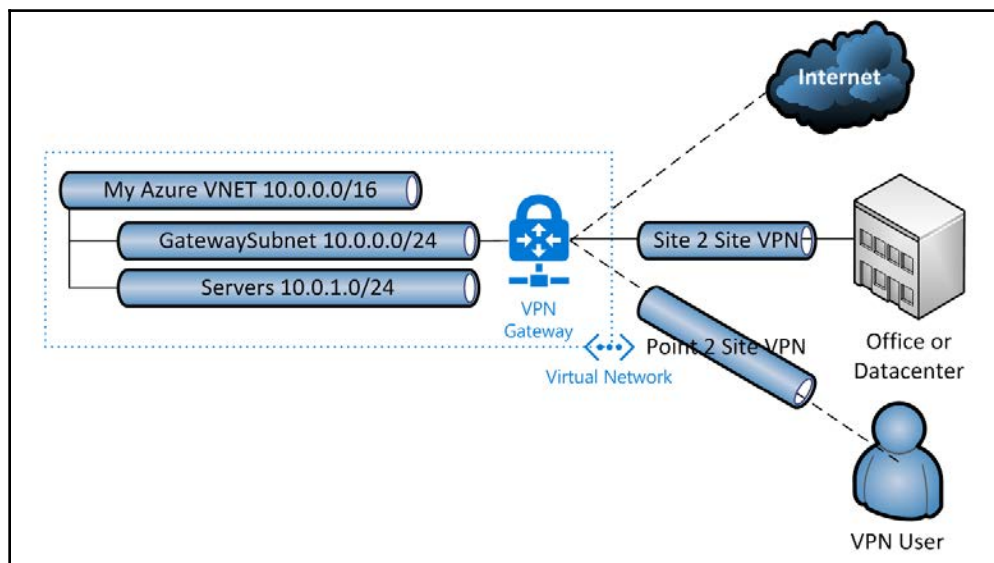


Looking on the current WAN developments and most of the customer infrastructures, a policy-based VPN gateway should only be used if there is absolutely no other option. Most enterprise grade Firewalls are able to work with route-based VPN. Otherwise you can switch to a virtual network device in Azure. Behind the following link you will find a list of devices with information about their available VPN options. <https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-about-vpn-devices>.

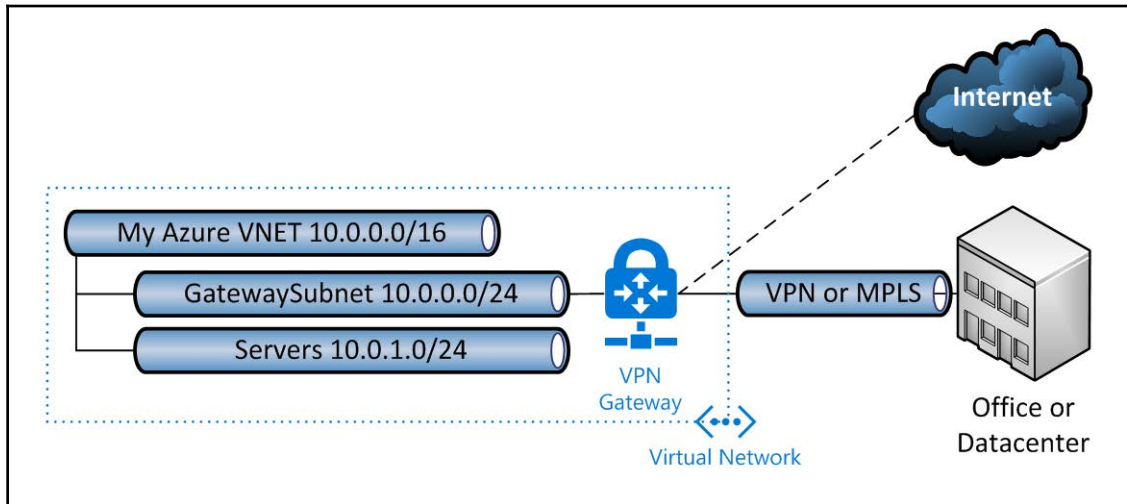
- Route-based standard **VPN gateway** with **ExpressRoute** shown in the following diagram:



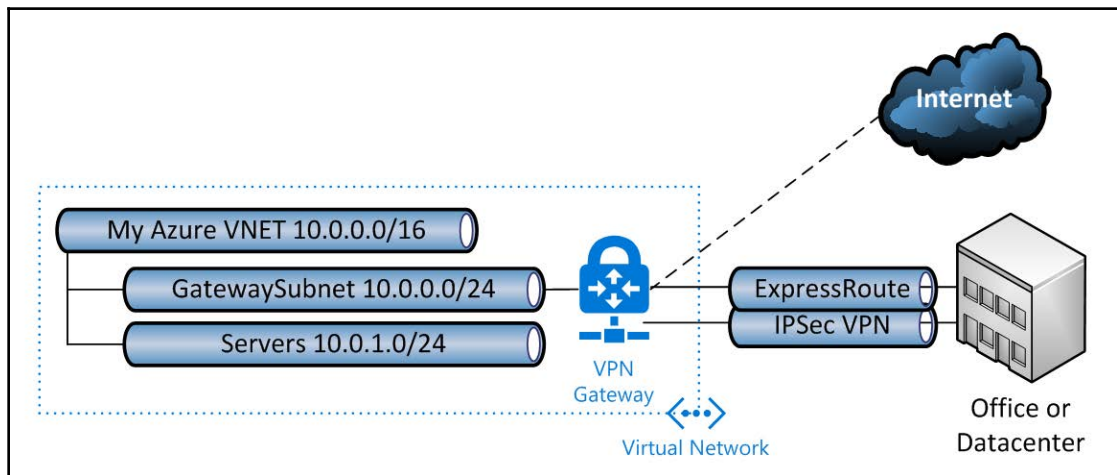
- Route-based basic **VPN Gateway** with a **Site 2 Site VPN** and **Point 2 Site VPN** or a **Route-based** standard or performance **VPN gateway** with a **Site 2 Site VPN** and **Point 2 Site VPN** in shown in the following diagram:



- Route-based standard or performance **VPN gateway** with **Site to Site** or **ExpressRoute** in shown in the following diagram:



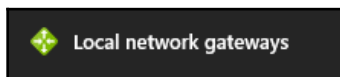
- Route-based standard or performance **VPN gateway** with a site-to-site VPN and **ExpressRoute**:



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 titled "Create local network g..." and has a close button. It contains several fields: "Name" with the value "LocalNetworkGateway" and a green checkmark; "IP address" with an empty field; "Address space" with a list of two ranges: "192.168.10.0/24" and "16.0.0.0/16", each with a three-dot menu icon, and a button "Add additional address range" with a three-dot menu icon; "Subscription" with a dropdown menu showing "Visual Studio Premium with MSDN"; "Resource group" with radio buttons for "Create new" and "Use existing", and a dropdown menu showing "DCF-AWE-RG01"; and "Location" with 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 I 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 virtual WAN

Azure virtual WAN is a networking solution based on SD-WAN that provides an automated branch-to-branch connectivity through the Azure backbone network. Virtual WAN allows to connect and configure branch devices to communicate with the Azure SD-WAN hub. This can be done manually, for not supported devices or by using preferred partner devices, where the configuration can be done automatically.



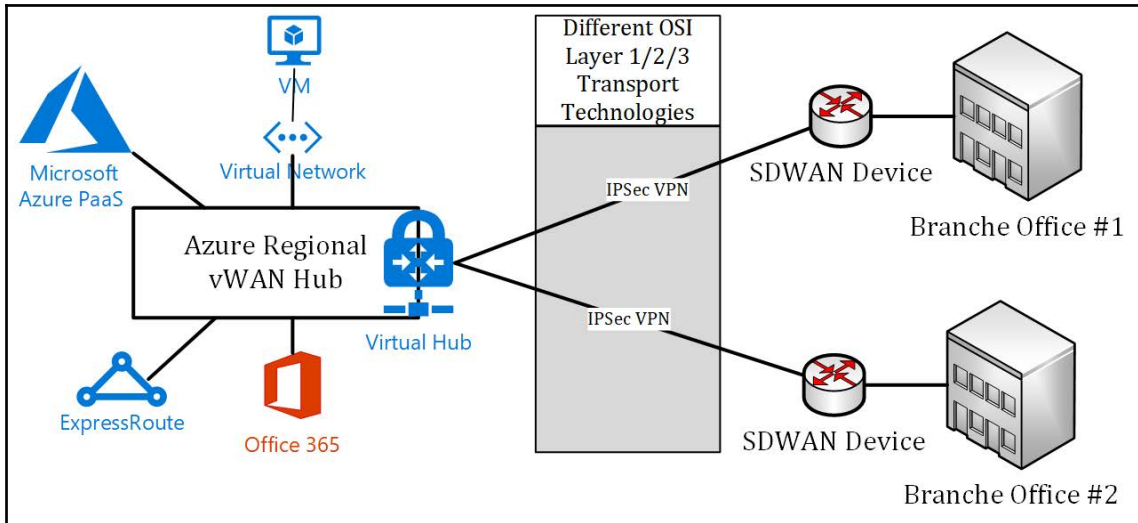
A list of supported devices, partner, and locations can be found here. <https://docs.microsoft.com/en-us/azure/virtual-wan/virtual-wan-locations-partners>.

Using Microsoft partner devices allows to simplification of connectivity and configuration management. The Azure WAN built-in dashboard helps instant troubleshooting insights and gives you a way to get an overview about large-scale connectivity networks.



SD-WAN is an acronym for software-defined networking abstraction layer for **wide area networks (WAN)**. An SD-WAN makes the management and operation of a WAN more simple by decoupling (separating) the WAN hardware from its control mechanism like MPLS, DSL, or even mobile protocols like LTE. This concept of SD-WAN is similar to software-defined networking implementation where virtualization technology improves data center management and operation. SD-WAN is meant to be the future of performant and cost efficient wide area networks.

The following drawing shows a schematic overview about the implementation Azure:



Because Azure vWAN is a very new service and still partly in preview. It is highly recommended to follow the documentation to stay on track about the capabilities and changes. <https://docs.microsoft.com/en-us/azure/virtual-wan/virtual-wan-about>.

Please be aware, even if Microsoft delivers you a very powerful backbone, you still need good last mile providers. Those providers should have a very good latency to Azure and a lot of peering's with Microsoft within the area you want to use them.



Please be careful, not every big provider also has a good local network. Often they need to lease line from local MAN providers or especially in the international business from other providers. That makes those providers very expensive, less performant, and a bad choice for a good cloud connectivity.

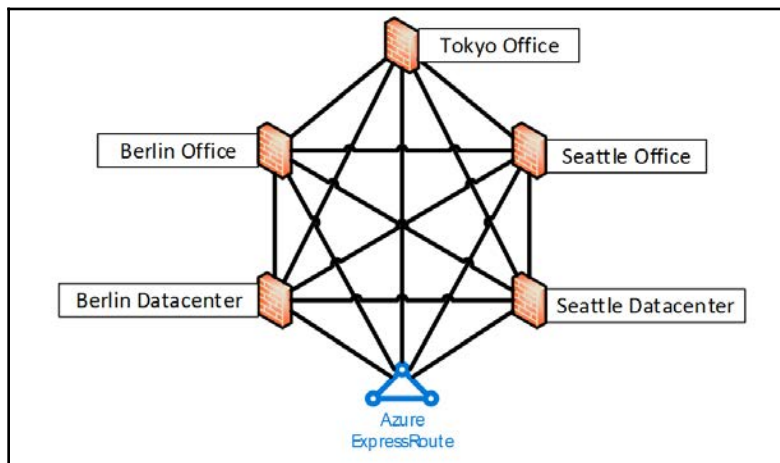
Azure ExpressRoute

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 (QoS)** 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.

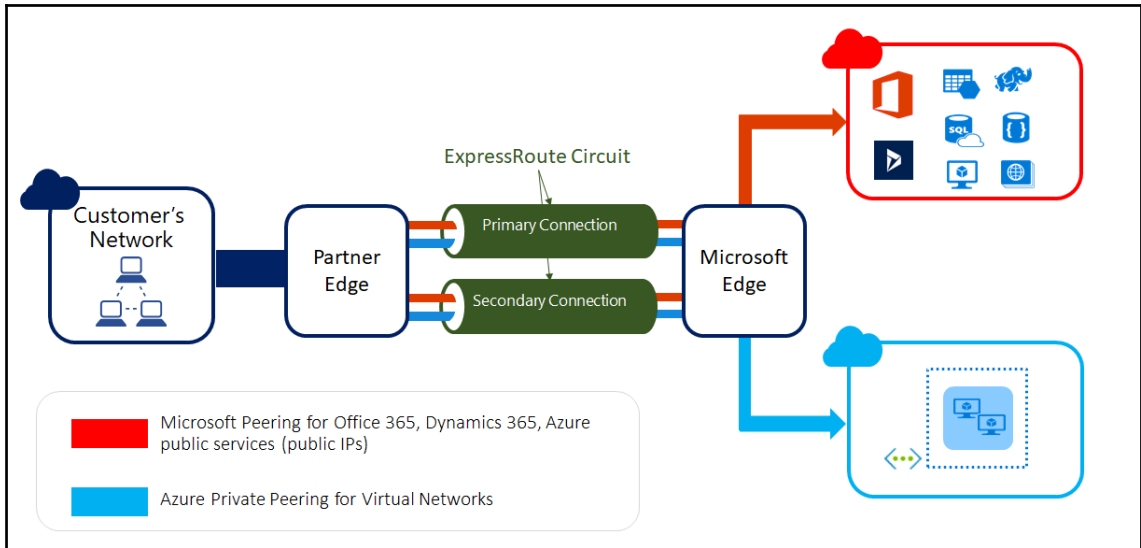
When you configure, you will be able to configure the different peering's. The three peering types are:

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



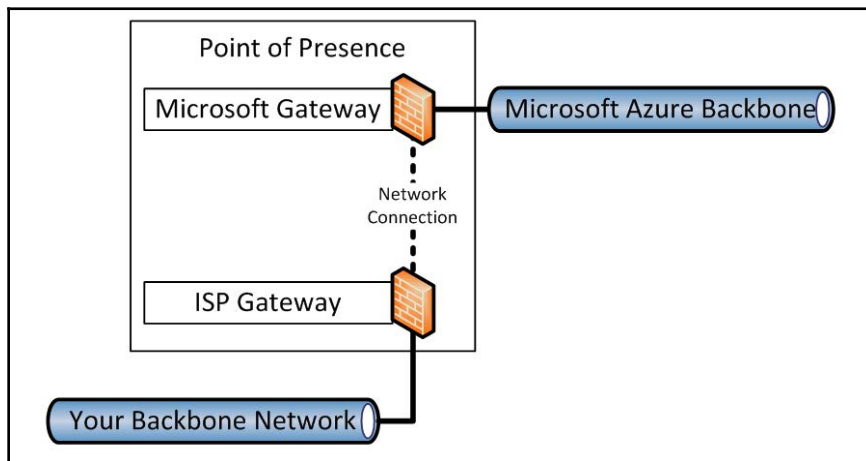
Because of the different deployment and distribution strategy from Azure and Microsoft 365 Services, it is not recommended to use ExpressRoute for Microsoft 365. You should only use ExpressRoute for Azure Services and/or as your global MPLS backbone interconnect.

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)**, **Meet Me Locations** or **Private Network Interconnect** hubs all over the globe. The following diagram shows how this happens within the Azure data center:





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, and so on, 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-nameclpartnersaconnectivity-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:

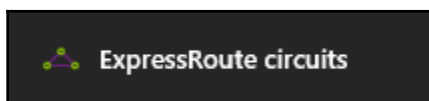
Circuit size	Number of VNet links for standard	Number of VNet links with premium add-on
50 Mbps	10	20
100 Mbps	10	25
200 Mbps	10	25
500 Mbps	10	40
1 Gbps	10	50
2 Gbps	10	60
5 Gbps	10	75
10 Gbps	10	100

ExpressRoute is highly recommended for enterprise environments which need a guarantee for latency and bandwidth for their Azure 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, I will explain how to deploy an ExpressRoute circuit.

Route filter

When Microsoft peering is configured on your ExpressRoute circuit, the Microsoft edge routers establish a pair of BGP sessions with the customer or provider edge routers. In the first place no routes are advertised to a customer network. To enable route advertisements to those networks, a route filter must be associated. A route filter lets identify services a customer want to consume through your ExpressRoute circuit's Microsoft peering. It is essentially a white list of all the BGP community values that should be announced. Once a route filter resource is defined and attached to an ExpressRoute circuit, all prefixes that map to the BGP community values are advertised to the customer network.

ExpressRoute Direct

ExpressRoute Direct provides the ability to connect directly into Microsoft's global network at peering locations across the world. ExpressRoute Direct provides a redundant 100 Gbps connectivity, which supports active/active connectivity and scalability.



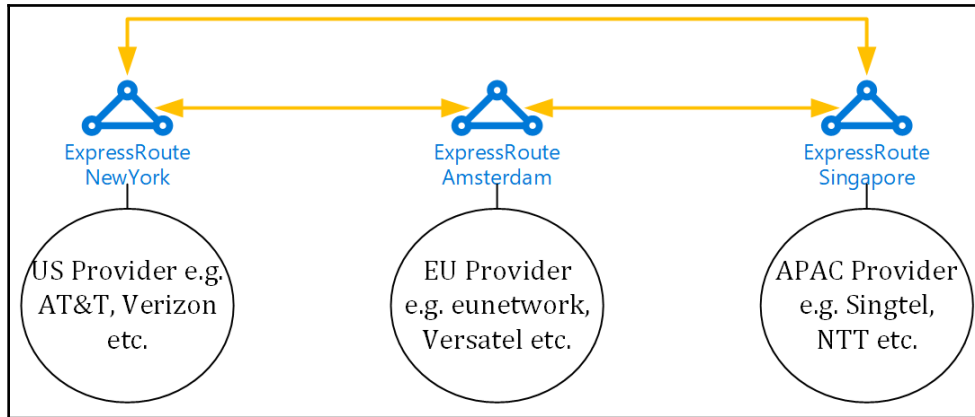
While writing the book, ExpressRoute Direct is in public preview.

ExpressRoute Global Reach

With **ExpressRoute Global Reach**, customers are able to link ExpressRoute circuits together to make a private backbone network between your on-premises networks via the Microsoft global backbone. With that, Microsoft becomes the global backbone provider of the customer.

That enables customers to always choose the best local MPLS or WAN provider in their area, link them to an ExpressRoute and afterwards link the ExpressRoutes together to build a global Backbone Network.

The following diagram shows a schematic overview:



With ExpressRoute Global Reach, Customers get the option to always choose, the best and cost efficient local provider to connect to Azure. Customer gain also flexibility when choosing providers and are able to easily exchange providers or built cheap WAN Networks of their own. It also an option for smaller ISPs and Network providers because they can now use ExpressRoute to use Microsoft cables through the Atlantic and pacific instead of renting expensive dark fiber of their own.

ExpressRoute Global Reach is currently available in following regions:

- Australia
- France
- Hong Kong
- Ireland
- Japan
- Netherlands
- United Kingdom
- United States



Microsoft is currently working to make more regions available. With ExpressRoute Global Reach, Microsoft becomes the second largest telecom provider on the globe and therefore they are a some government regulations and restrictions which need to be cleared first. Afterwards Microsoft will bring the service into those regions first.

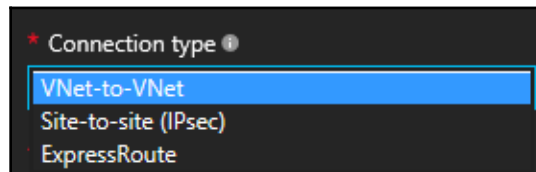
Azure connections

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:

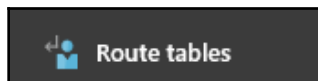


Later on in the chapter, we will use Azure connections to build up a connection between virtual network and MPLS gateways.

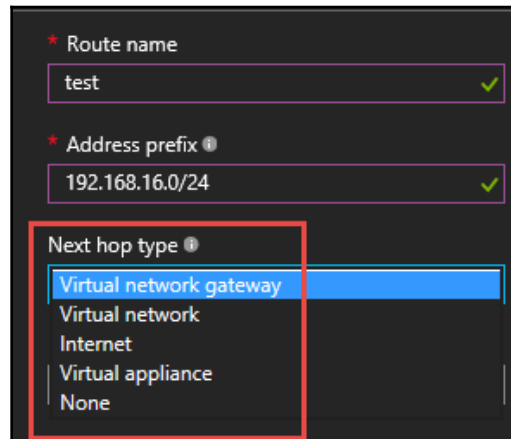
Azure route

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, I 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 Firewall

Azure Firewall is a managed, cloud-based OSI Layer 3 to 7 network security service that protects customer Azure VNet resources. It is a fully stateful firewall as a service with implemented high availability and unrestricted scalability in the cloud.

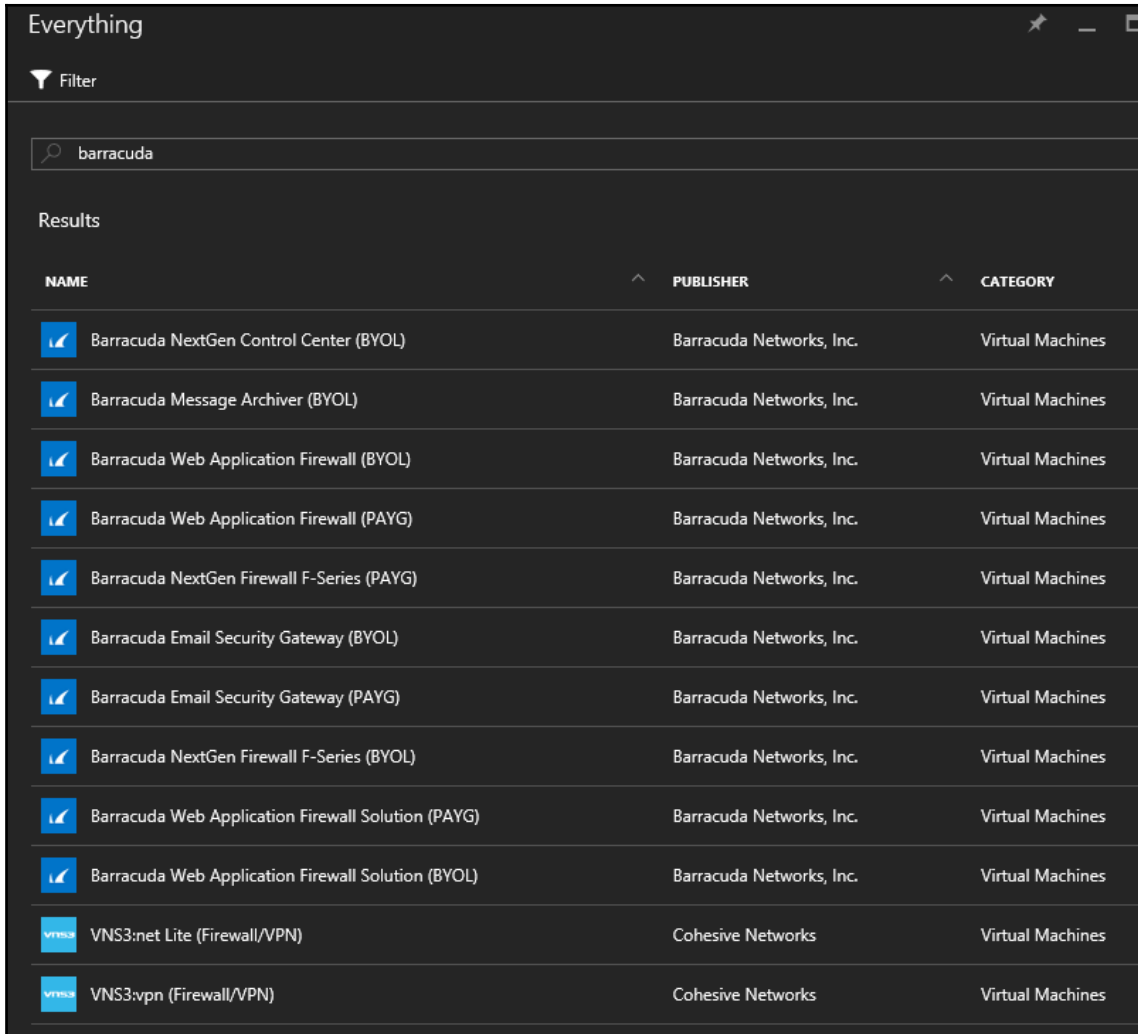
Customers can centrally create, enforce, and log application and network connectivity policies across subscriptions and virtual networks within one Tenant. Azure Firewall uses a static public IP address for a VNet resources, that allows outside firewalls to identify traffic originating from a VNet. The service is fully integrated within the Azure Monitoring services like Azure Monitor, that works as an interface and platform for logging and analytics.

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 in 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`:



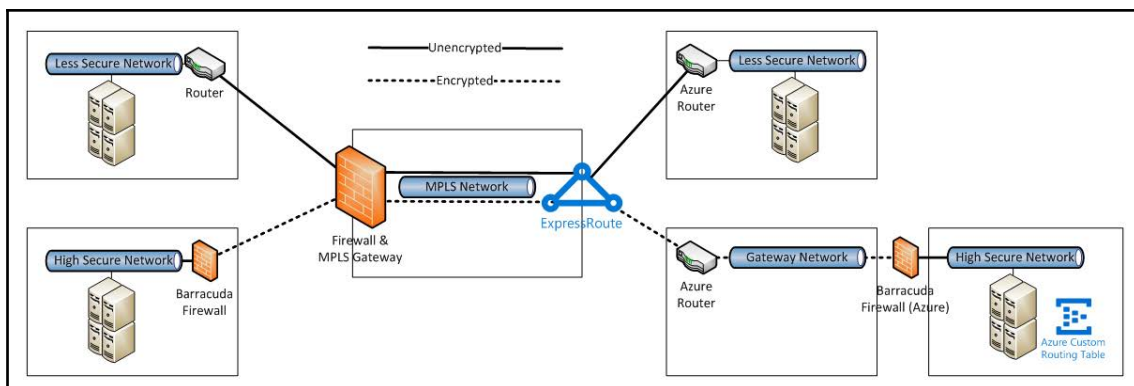
The screenshot shows the Azure Marketplace interface with a search filter set to 'barracuda'. The results table lists various Barracuda products, all categorized as 'Virtual Machines'. The products include NextGen Control Center, Message Archiver, Web Application Firewall (BYOL and PAYG), NextGen Firewall F-Series (PAYG and BYOL), Email Security Gateway (BYOL and PAYG), and VNS3:net Lite and VNS3:vpn from Cohesive Networks.

NAME	PUBLISHER	CATEGORY
Barracuda NextGen Control Center (BYOL)	Barracuda Networks, Inc.	Virtual Machines
Barracuda Message Archiver (BYOL)	Barracuda Networks, Inc.	Virtual Machines
Barracuda Web Application Firewall (BYOL)	Barracuda Networks, Inc.	Virtual Machines
Barracuda Web Application Firewall (PAYG)	Barracuda Networks, Inc.	Virtual Machines
Barracuda NextGen Firewall F-Series (PAYG)	Barracuda Networks, Inc.	Virtual Machines
Barracuda Email Security Gateway (BYOL)	Barracuda Networks, Inc.	Virtual Machines
Barracuda Email Security Gateway (PAYG)	Barracuda Networks, Inc.	Virtual Machines
Barracuda NextGen Firewall F-Series (BYOL)	Barracuda Networks, Inc.	Virtual Machines
Barracuda Web Application Firewall Solution (PAYG)	Barracuda Networks, Inc.	Virtual Machines
Barracuda Web Application Firewall Solution (BYOL)	Barracuda Networks, Inc.	Virtual Machines
VNS3:net Lite (Firewall/VPN)	Cohesive Networks	Virtual Machines
VNS3:vpn (Firewall/VPN)	Cohesive Networks	Virtual Machines

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.

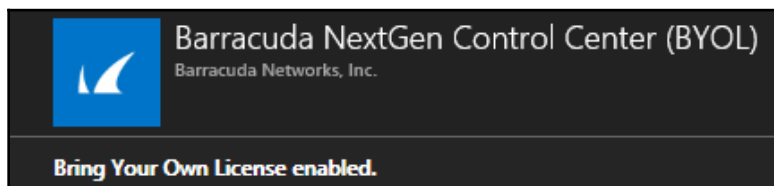
Normally there is no need to implement a third-party device because the Microsoft standard services offer approximately 80% of all services that are needed by most customers. Sometimes there are cases where you need to implement special systems such as a load balancer in Azure. Under certain circumstances, 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:



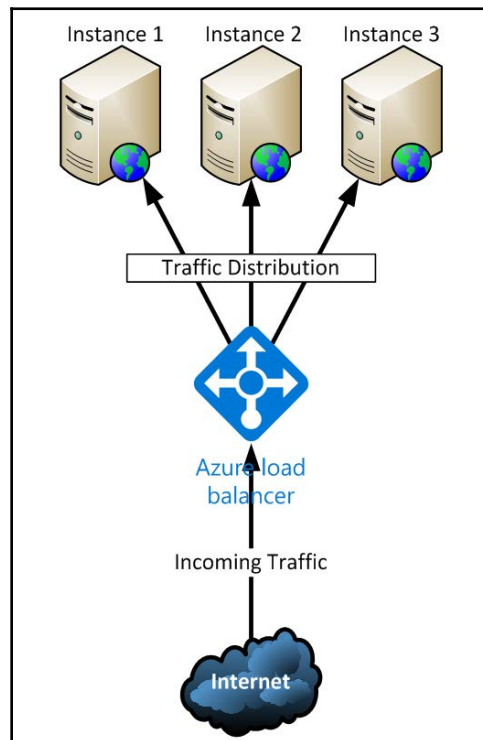
In the text, there should be additional information, as shown in the example screenshot :

- 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 and Web Application Firewall

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**.

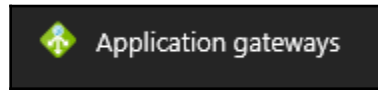
You can create up to 50 application gateways per subscription. Each application gateway can have up to 10 instances each, which makes up to 500 instances depending on the gateway site.

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/>.

Web Application Firewall

Web Application Firewall (WAF) is an operation mode of application gateway that provides centralized protection of customer web applications from most common exploits and vulnerabilities.

WAF is based on rules from the OWASP core rule sets 3.0 or 2.2.9, which can also be customized. It automatically updates it self to include protection against new vulnerabilities and exploits, without any additional configuration.

Following features are currently included in the service:

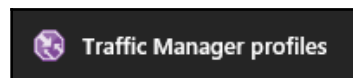
- SQL injection protection
- Cross site scripting protection
- Common Web Attacks Protection like command injection, HTTP request smuggling, HTTP response splitting, or remote file inclusion attack
- Protection against HTTP protocol violations
- Protection against HTTP protocol anomalies like missing host user-agent and accept headers
- Prevention against bots, crawlers, and scanners
- Detection of common application misconfigurations (that is, Apache, IIS, and so on.)

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 United Domains and other DNS providers.

Azure DDoS

Distributed Denial of Service (DDoS) attacks are some of the largest availability and security concerns global IT is facing at that time and while moving their applications to the cloud. A DDoS attacker attempts to exhaust an application's resources as long and as high with the result to make the application unavailable to legitimate users. DDoS attacks can be targeted at any endpoint that is publicly reachable through the internet or even if the attacker was able to corrupt a private network, from private network resources of bigger networks.

Azure DDoS protection together with application design by best practices will provide a defense against DDoS attacks. With Azure DDoS protection provides the following service tiers are available:

- **Basic:** Automatically enabled as part of the Azure platform and services. It includes Always-on traffic monitoring and real-time mitigation of common network-level attacks it provide the same defenses utilized by Microsoft's online services like Office 365, Outlook.com, Xbox, Azure SQL. The entire scale and resources of Azure's global network can be used to distribute and mitigate attack traffic across regions and with that, protect customers and workload against attacks. Protection is either available for IPv4 and IPv6 Azure public IP addresses.
- **Standard:** Provides additional mitigation capabilities over the Basic service tier that are tuned specifically to Azure virtual network resources. It enables customers to customize and optimize their DDoS protection to their Workloads within the VNet.

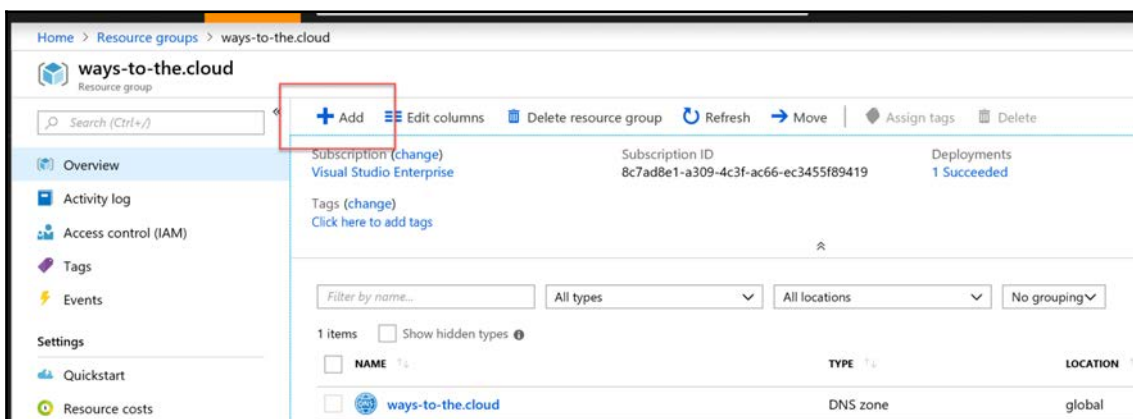
Setting up Azure networks

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.

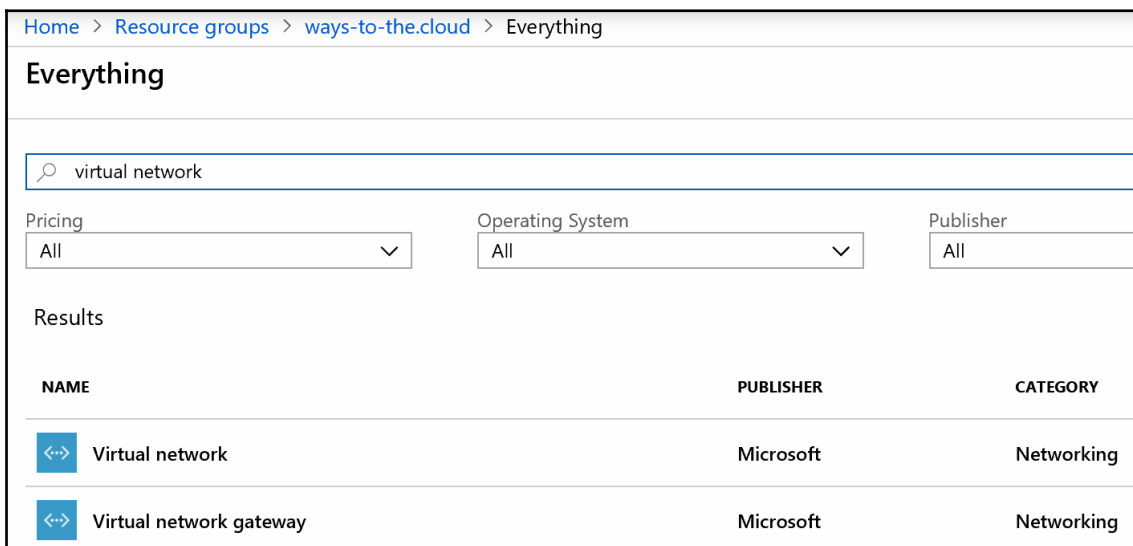
Setting up Azure VNet

The following are the steps to set up Azure VNet:

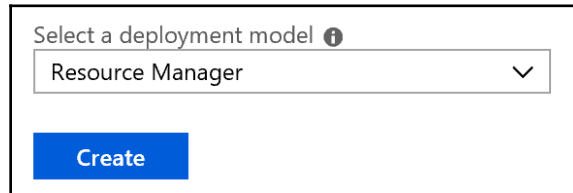
1. First of all, we navigate in our **Resource group** and use **Add** to open the Azure marketplace:



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 some when remove ASM and all resources deployed within it:

A screenshot of a web interface for selecting a deployment model. It features a label "Select a deployment model" with an information icon. Below it is a dropdown menu with "Resource Manager" selected and a downward arrow. At the bottom is a blue "Create" button.

Select a deployment model ⓘ

Resource Manager ▼

Create

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.
 - **DDoS protection:** You can enable DDoS standard for the VNet.
 - **Service endpoints:** You can enable service endpoints for that VNet
 - **Firewall:** You can enable Azure firewall for the VNet.

The rest should be predefined by your resource group:

Create virtual network

*

Name

*

Address space

10.1.0.0/16

10.1.0.0 - 10.1.255.255 (65536 addresses)

*

Subscription

Visual Studio Enterprise

*

Resource group

Select existing...

Create new

*

Location

West Europe

Subnet

*

Name

default

*

Address range

10.1.0.0/24

10.1.0.0 - 10.1.0.255 (256 addresses)

DDoS protection

Basic

Standard

Service endpoints

Disabled

Enabled

Firewall

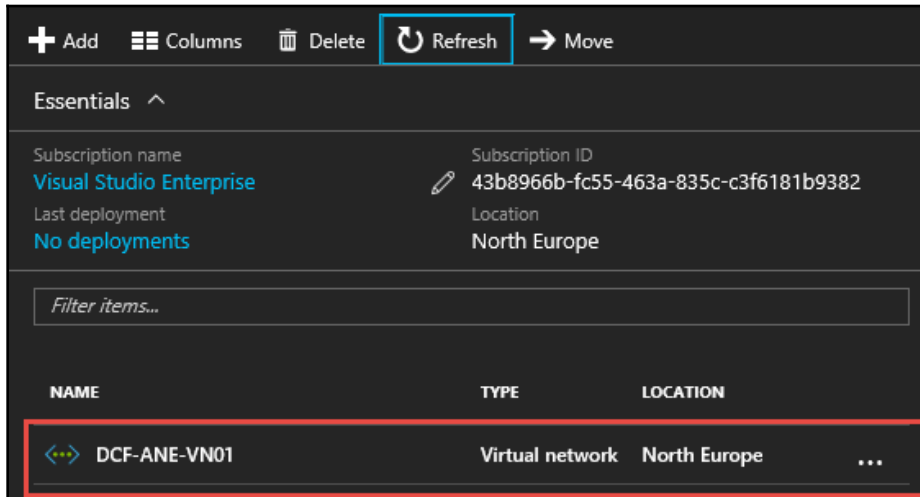
Disabled

Enabled

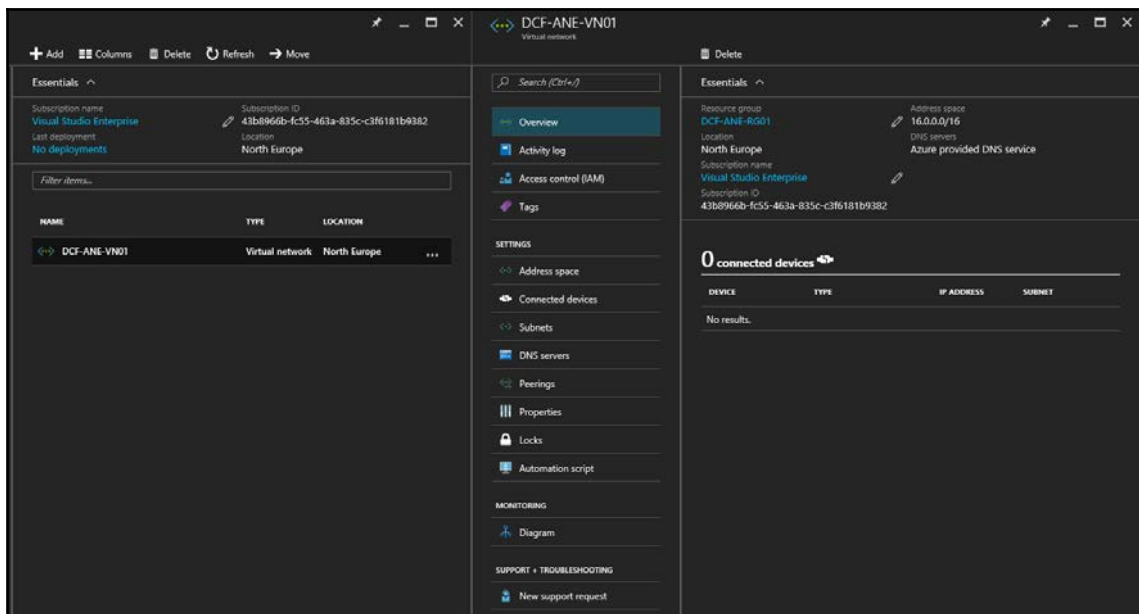
Create

Automation options

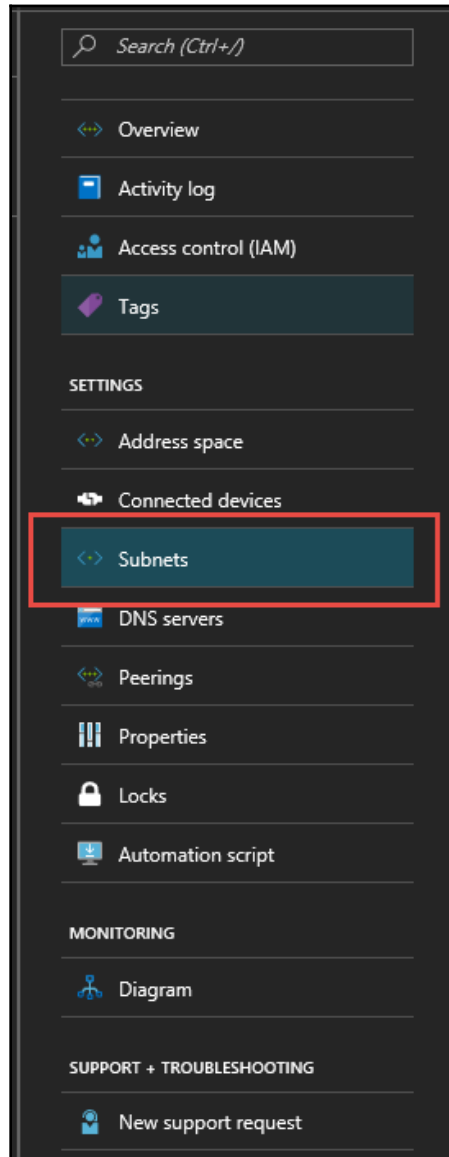
- After the creation, the **Virtual Network (VNet)**, should be listed in your resource group. You maybe need to click on **Refresh**:



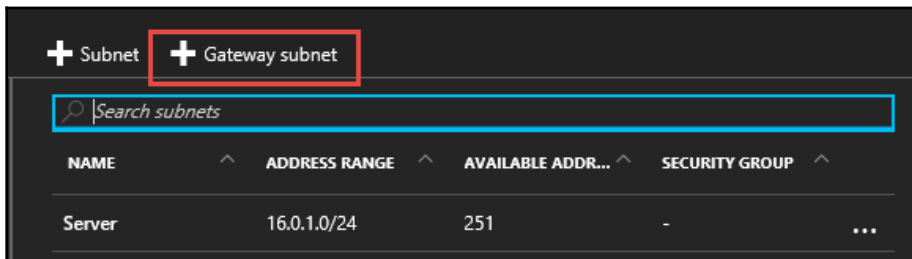
- Now we will add our gateway subnet. Therefore, we select the VNet to open the settings:



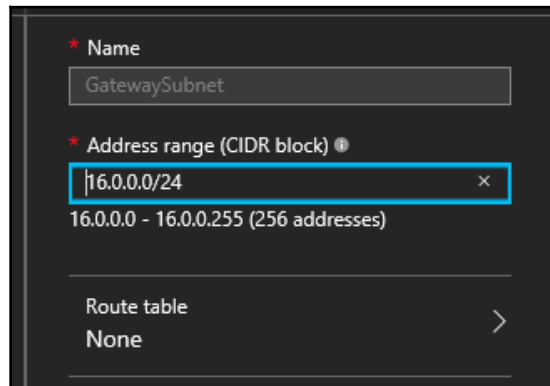
7. Yet we want to add the gateway subnet. Click on **Subnets** to open the subnet blade:



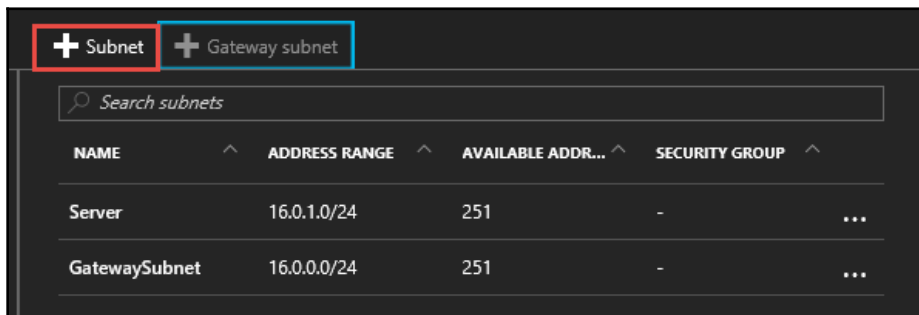
- Afterwards you select the **+ Gateway subnet** button and another blade with the creation details will open:



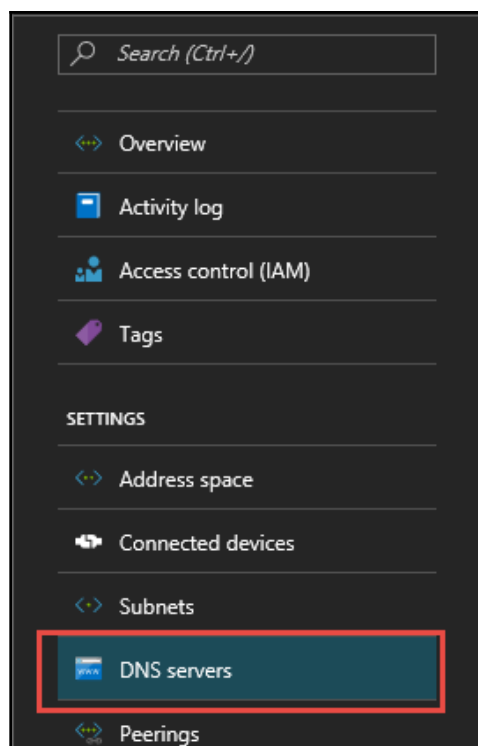
- Within that blade, you need to define the subnet mask that the gateway uses. As explained earlier, we need a minimum of /29 CIDR addresses:



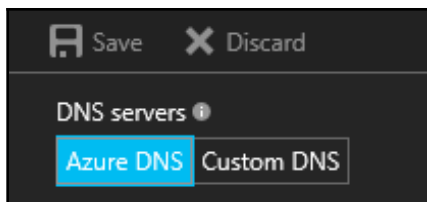
- After clicking **OK**, the gateway subnet will be deployed in Azure. If you want to add more subnetworks, you can use the **Subnet** button:



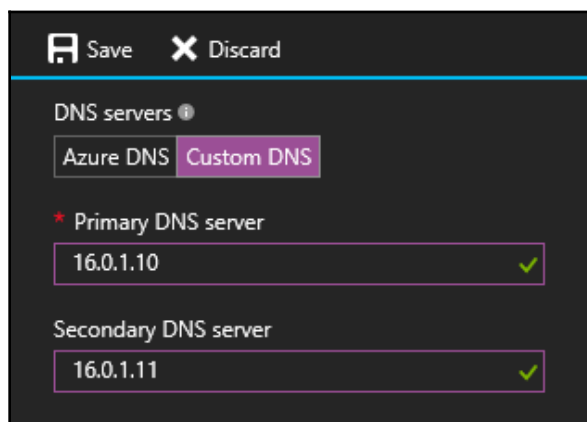
11. In the VNet settings, there is another option that could come in handy. Azure deploys every system with DHCP and by default configures Microsoft DNS servers as default DNS to the virtual machine. In most scenarios, you will need to change this setting to your own DNS server. Therefore, you can do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

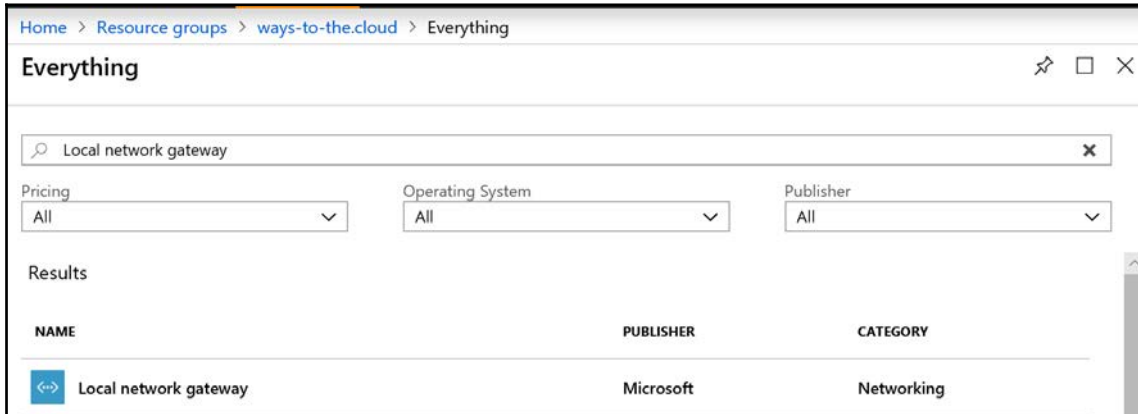
Setting up Azure virtual network site-to-site VPN

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

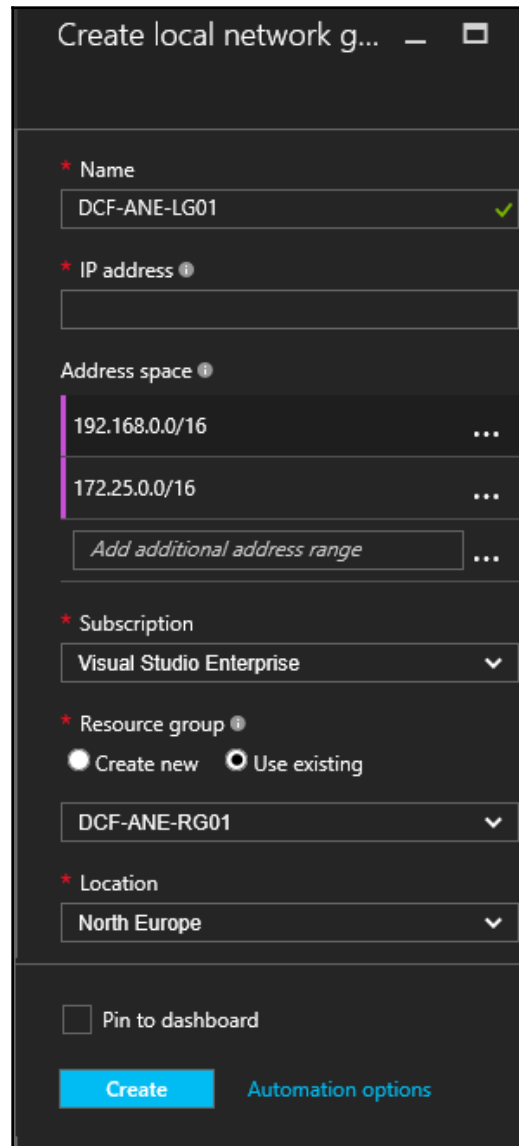
Network gateway follow the given steps:

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 gateway in the Azure marketplace:



2. In the upcoming menu, you need to click on **Create**, afterwards the set up for the local gateway configuration appears:
 - **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 gateway' and includes the following fields and options:

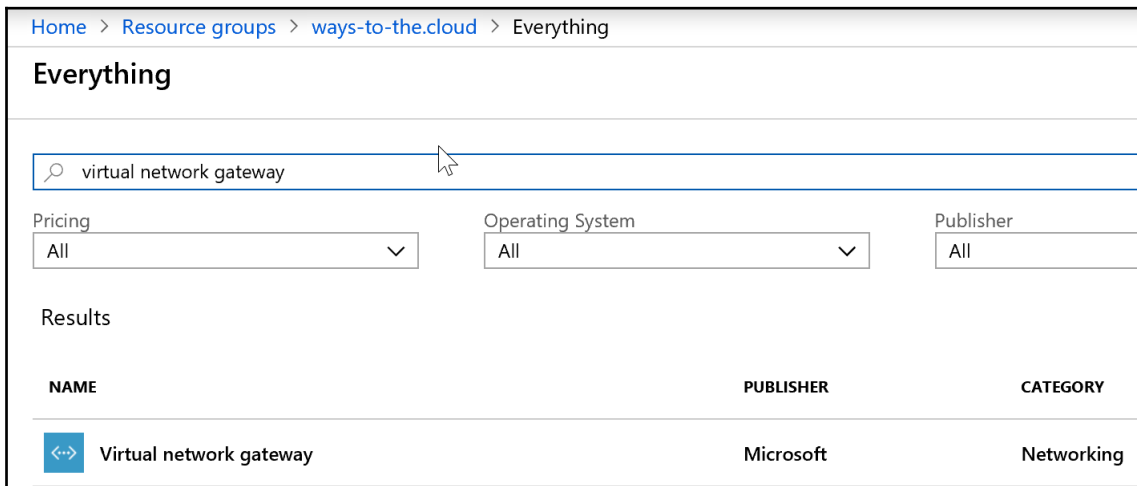
- Name:** DCF-ANE-LG01 (with a green checkmark indicating it is valid).
- IP address:** An empty text box.
- Address space:** A list of address ranges with a vertical scrollbar. The visible ranges are 192.168.0.0/16 and 172.25.0.0/16. There is an 'Add additional address range' button at the bottom of the list.
- Subscription:** Visual Studio Enterprise (selected from a dropdown menu).
- Resource group:** DCF-ANE-RG01 (selected from a dropdown menu). Below this, there are radio buttons for 'Create new' (selected) and 'Use existing'.
- Location:** North Europe (selected from a dropdown menu).
- Pin to dashboard:** An unchecked checkbox.
- Create:** A blue button to create the gateway.
- Automation options:** A link to view automation options.

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:

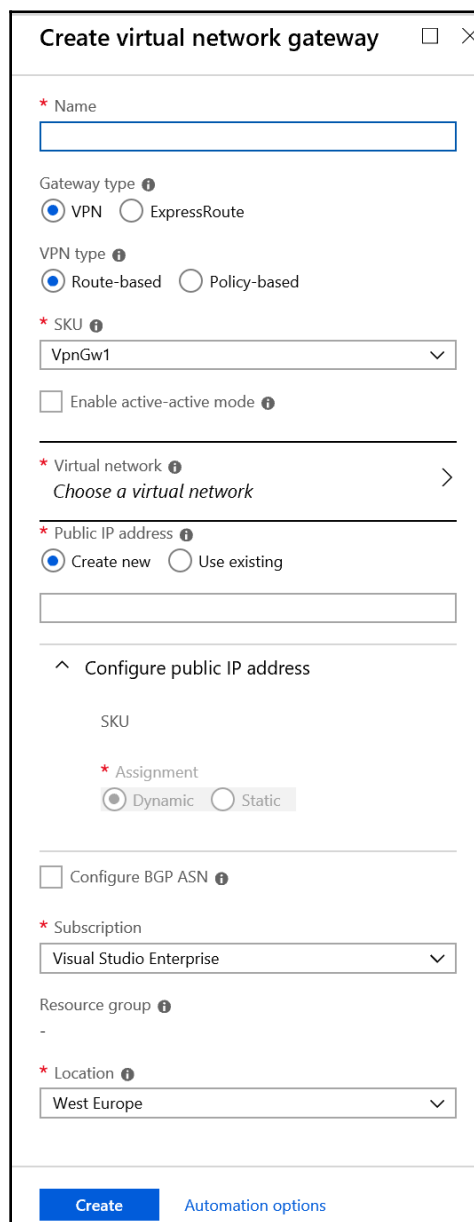
1. At first, we need to deploy the virtual network gateway. Therefore, we click **Add** in our resource group and search after `virtual network gateway` in the Azure marketplace:



The screenshot shows the Azure Marketplace search interface. The breadcrumb navigation at the top reads: [Home](#) > [Resource groups](#) > [ways-to-the.cloud](#) > Everything. Below this, the word "Everything" is displayed. A search bar contains the text "virtual network gateway". Below the search bar are three filter dropdowns: "Pricing" (set to "All"), "Operating System" (set to "All"), and "Publisher" (set to "All"). Under the filters, the word "Results" is shown. Below "Results" is a table with three columns: "NAME", "PUBLISHER", and "CATEGORY". The table contains one result: "Virtual network gateway" by "Microsoft" in the "Networking" category. The "Virtual network gateway" text is preceded by a blue icon with a double-headed arrow.

NAME	PUBLISHER	CATEGORY
 Virtual network gateway	Microsoft	Networking

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



The screenshot shows the 'Create virtual network gateway' blade in the Azure portal. The blade is titled 'Create virtual network gateway' and has a close button (X) in the top right corner. The form contains the following fields and options:

- Name:** A text input field.
- Gateway type:** Radio buttons for **VPN** (selected) and **ExpressRoute**.
- VPN type:** Radio buttons for **Route-based** (selected) and **Policy-based**.
- SKU:** A dropdown menu showing **VpnGw1**.
- Enable active-active mode:** A checkbox that is currently unchecked.
- Virtual network:** A section with a right arrow and the text 'Choose a virtual network'.
- Public IP address:** Radio buttons for **Create new** (selected) and **Use existing**. Below this is an empty text input field.
- Configure public IP address:** A section with a caret icon and the title 'Configure public IP address'. It contains:
 - SKU:** A label.
 - Assignment:** Radio buttons for **Dynamic** (selected) and **Static**.
- Configure BGP ASN:** A checkbox that is currently unchecked.
- Subscription:** A dropdown menu showing **Visual Studio Enterprise**.
- Resource group:** A label with a dash '-' below it.
- Location:** A dropdown menu showing **West Europe**.
- Buttons:** A blue **Create** button and a link for **Automation options**.

- **Name:** Set the name of the virtual network gateway.
- **Gateway type:** You need to choose between VPN and ExpressRoute. In our case we choose VPN.
- **VPN type:** Now you need to decide between **Route-based** and **Policy-based**, it is recommended to use **Route-based** if possible.
- **SKU:** Choose your gateway size, you can also enable active/active mode.
- **Virtual network:** Connect the virtual network gateway to a specific virtual network.
- **Public IP address:** Create a public IP for the gateway. You can also create BGP if necessary .
- **Subscription:** Choose a subscription you want to deploy into.
- **Location:** Choose an Azure region to deploy to. The region must be the same than the virtual network.

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

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/>.

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



In some cases and with some firewall for example, Cisco ASA you need to do some PowerShell to reconfigure the VPN policies to match the vendor specific configuration. The PowerShell commands can be found here. <https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-ipsecikepolicy-rm-powershell#a-name-paramsapart-2---supported-cryptographic-algorithms--key-strengths>.

4. 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>
```

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

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

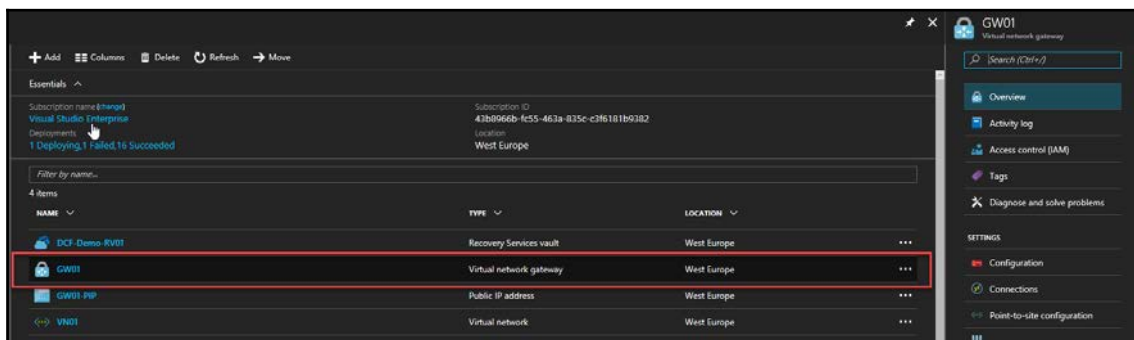
6. PowerShell command example to resize to standard gateway :

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

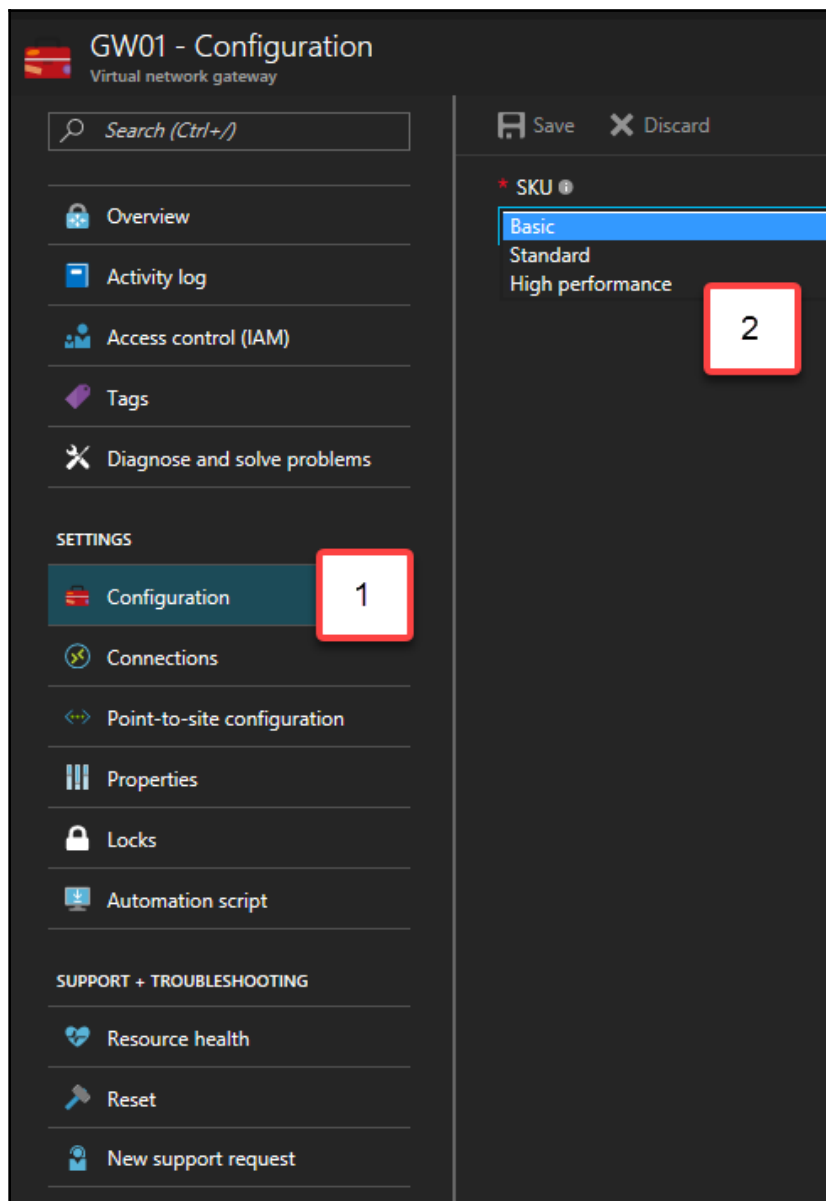
7. The same works also with downsizing a gateway:

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

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

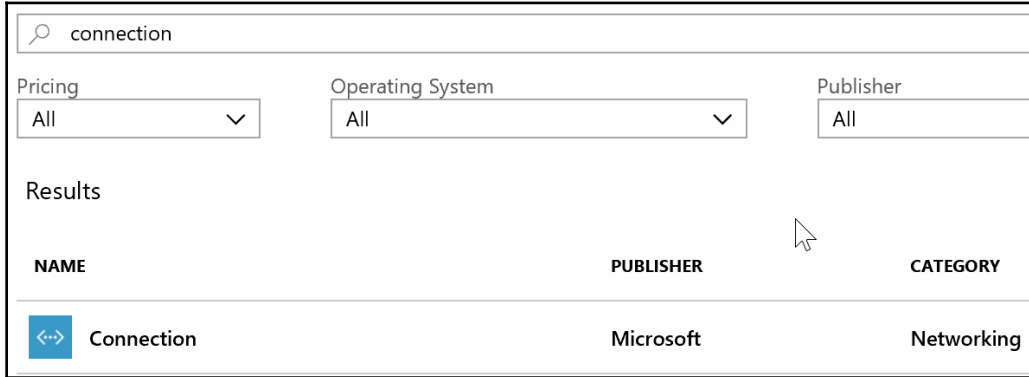


9. Within the detail blade you go to **Configuration** and change the **SKU**. Afterwards you need to save the new SKU. Please be aware that the change of the SKU will take again up to 45 minutes:



Configuring connection between local and virtual network gateways

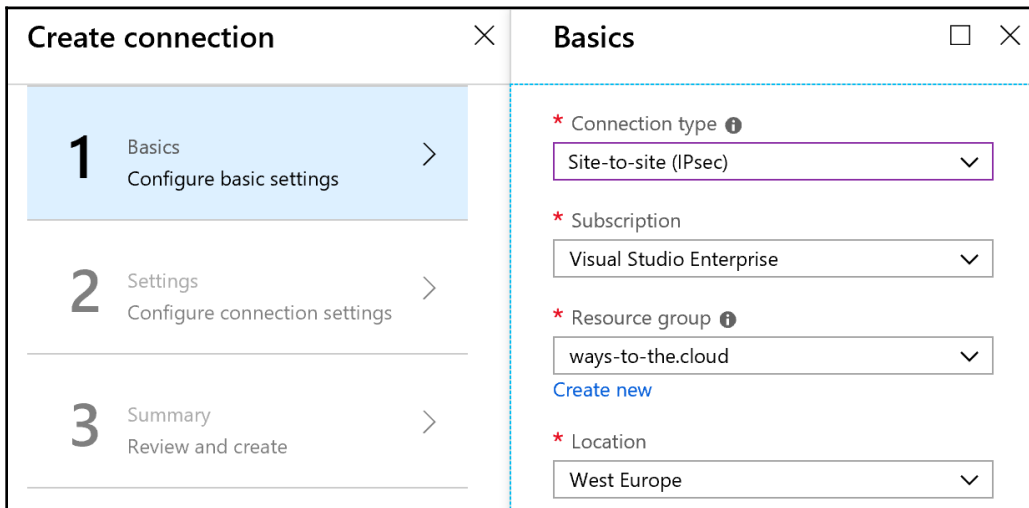
1. Now we need to establish a connection between gateways and enable the routing: Please go back to your resource group and click **Add** again. Now we look for **Connection** in the marketplace. Then select **Connection**:



The screenshot shows the Azure Marketplace search results for the term "connection". At the top, there is a search bar with the text "connection". Below the search bar, there are three filter dropdowns: "Pricing" set to "All", "Operating System" set to "All", and "Publisher" set to "All". The main section is titled "Results" and contains a table with the following columns: "NAME", "PUBLISHER", and "CATEGORY". A single result is displayed: "Connection" by "Microsoft" in the "Networking" category. The "Connection" entry has a blue icon with a double-headed arrow.

NAME	PUBLISHER	CATEGORY
 Connection	Microsoft	Networking

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



The screenshot shows the "Create connection" wizard in Azure. The wizard has three steps: 1. Basics (Configure basic settings), 2. Settings (Configure connection settings), and 3. Summary (Review and create). The "Basics" step is currently selected and highlighted in blue. The "Basics" section contains the following fields:

- Connection type**: A dropdown menu with "Site-to-site (IPsec)" selected.
- Subscription**: A dropdown menu with "Visual Studio Enterprise" selected.
- Resource group**: A dropdown menu with "ways-to-the.cloud" selected. Below this dropdown is a link that says "Create new".
- Location**: A dropdown menu with "West Europe" selected.

3. In the **Settings** phase, we first select the Azure **Virtual network gateway**, **Local network gateway**, **Connection name** and **Shared key**:

The screenshot shows the 'Create connection' and 'Settings' tabs in the Azure portal. The 'Create connection' tab on the left has three steps: 1. Basics (Configure basic settings, marked with a green check), 2. Settings (Configure connection settings, highlighted in blue), and 3. Summary (Review and create). The 'Settings' tab on the right contains the following fields:

- Virtual network gateway**: A dropdown menu with the text 'Choose a virtual network gateway'. This field is highlighted with a blue dashed border.
- Local network gateway**: A dropdown menu with the text 'Choose a local network gateway'.
- Connection name**: A text input field.
- Shared key (PSK)**: A text input field.
- Enable BGP**: A checkbox.

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

DCF-ANE-RG01
Resource group

Search (Ctrl+/)

Overview

Activity log

Access control (IAM)

Tags

SETTINGS

Quickstart

Resource costs

Essentials ^

Subscription name: Visual Studio Enterprise

Subscription ID: 43b8966b-fc55-463a-835c-c3f6181b9382

Last deployment: 9/10/2016 (Succeeded)

Location: North Europe

Filter items...

NAME	TYPE	LOCATION
DCF-ANE-GW01-DCF-ANE-LG01	Connection	North Europe
DCF-ANE-LG01	Local network...	North Europe
DCF-ANE-GW01-PIP	Public IP addr...	North Europe

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

DCF-ANE-GW01-DCF-ANE-LG01
Connection

Search (Ctrl+/)

Overview

Activity log

Access control (IAM)

Tags

SETTINGS

Delete

Essentials ^

Resource group	Data in
DCF-ANE-RG01	0 B
Status	Data out
Connecting	0 B
Location	Virtual network
North Europe	DCF-ANE-VN01
Subscription name	Virtual network gateway
Visual Studio Enterprise	DCF-ANE-GW01 (52.178.159.75)
Subscription ID	Local network gateway
43b8966b-fc55-463a-835c-c3f6181b9382	DCF-ANE-LG01 (91.65.84.40)



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.

- 6. 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).

- 7. 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 full supported and over all available software defined WAN in-dependend 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:

Create virtual network gateway

* Name

Gateway type ⓘ

☐ VPN ☒ ExpressRoute

* SKU ⓘ

Standard

☐ Enable active-active mode ⓘ

* Virtual network ⓘ

Choose a virtual network

* Public IP address ⓘ

☒ Create new ☐ Use existing

Configure public IP address

SKU

* Assignment

☒ Dynamic ☐ Static

☐ Configure BGP ASN ⓘ

* Subscription

Visual Studio Enterprise

Resource group ⓘ

-

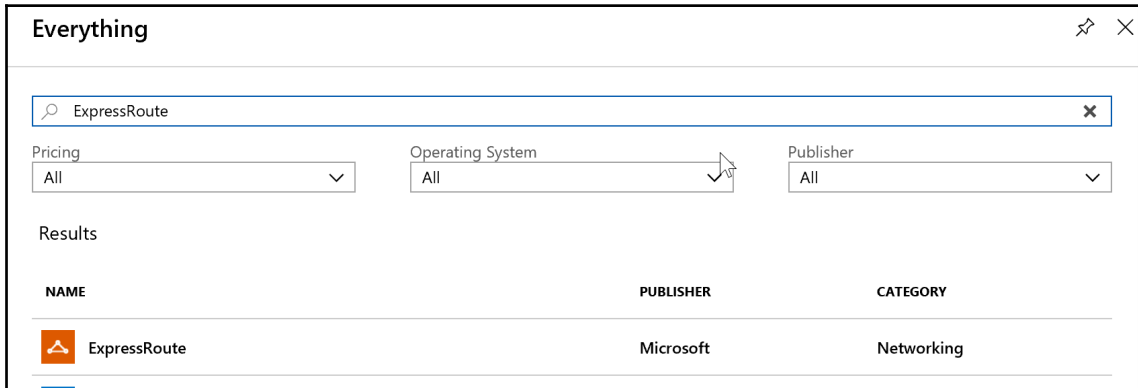
* Location ⓘ

West Europe

Create Automation options

Configuring Azure ExpressRoute circuit

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.

Create ExpressRoute circuit

Create new Import

* Circuit name

* Provider ⓘ

* Peering location ⓘ

* Bandwidth ⓘ

* SKU ⓘ

Standard Premium

* Billing model ⓘ

Unlimited Metered

☐ Allow classic operations ⓘ

* Subscription

Visual Studio Enterprise

* Resource group

Select existing...

[Create new](#)

* Location

West Europe

Create

Automation options



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+F)

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	Provider	Equinix
Circuit status	Enabled	Provider status	Not provisioned
Location	North Europe	Peering location	Amsterdam
Subscription name	Visual Studio Enterprise	Bandwidth	50 Mbps
Subscription ID	43b8966b-fc55-463a-835c-c3f6181b9382	Service key	2c3b3488-2f70-48d6-a2c0-8a0d0e7db148

Peerings

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

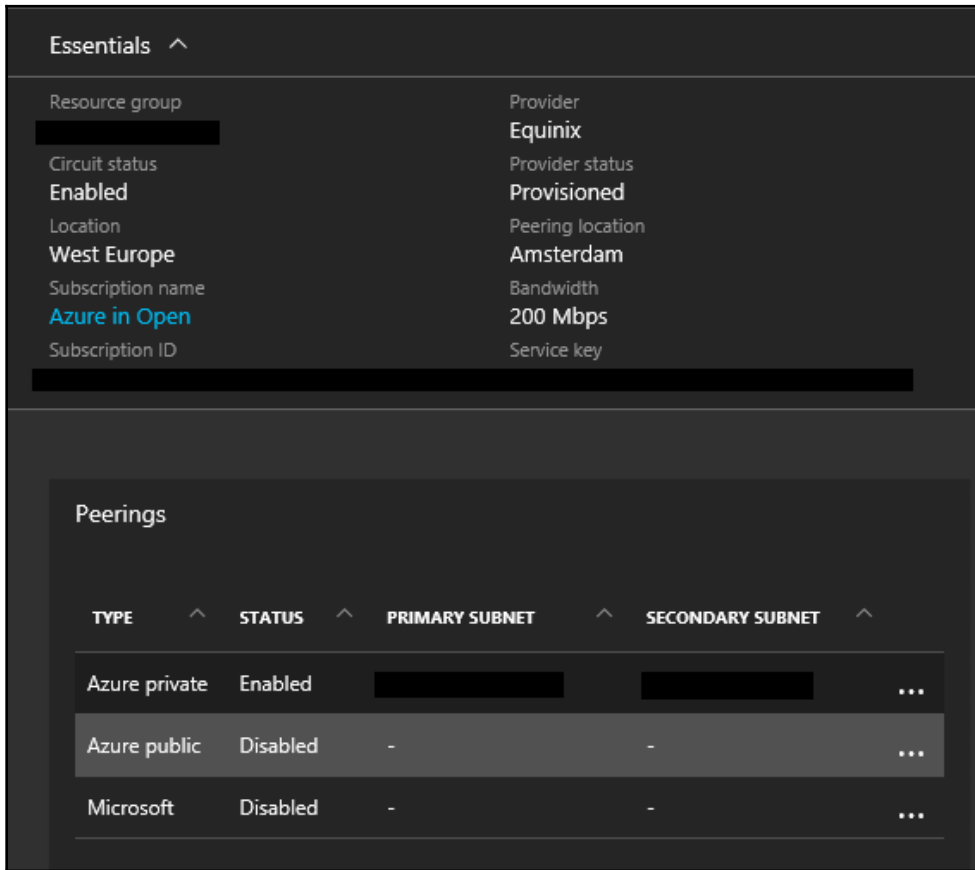
5. After the **Provider** status has changed to **Provisioned**, you can configure the Azure peering's:

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:



The screenshot displays the Azure portal interface for a network circuit. The top section, titled "Essentials", provides key details about the circuit. Below this, the "Peerings" section contains a table listing the configured peering types.

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

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

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, like you already did for VPN in the previous guide.

Setting up Azure VNet peering

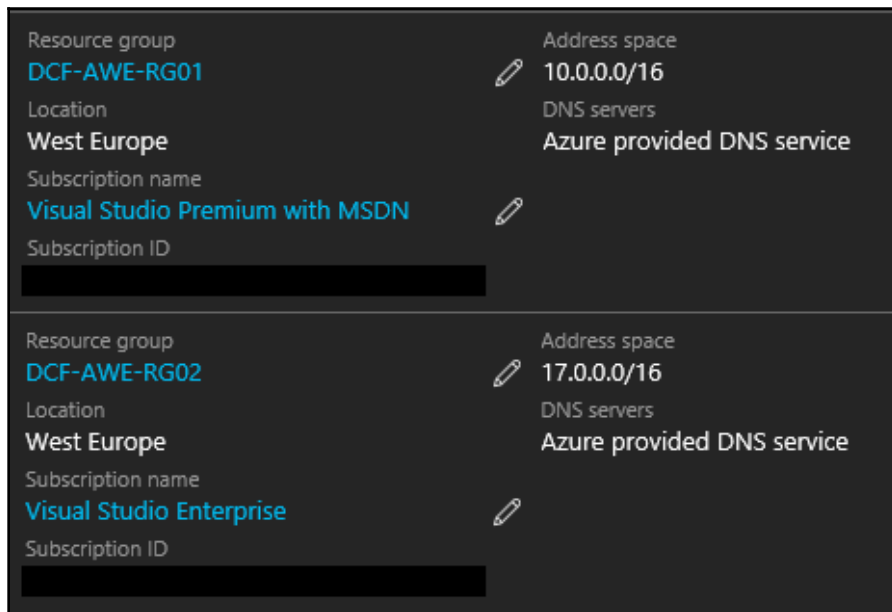
Now let's look at how you implement the new VNet peering. As already mentioned, you can use VNet peering to configure a connection between VNets in the same Azure region via the Azure backbone.

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

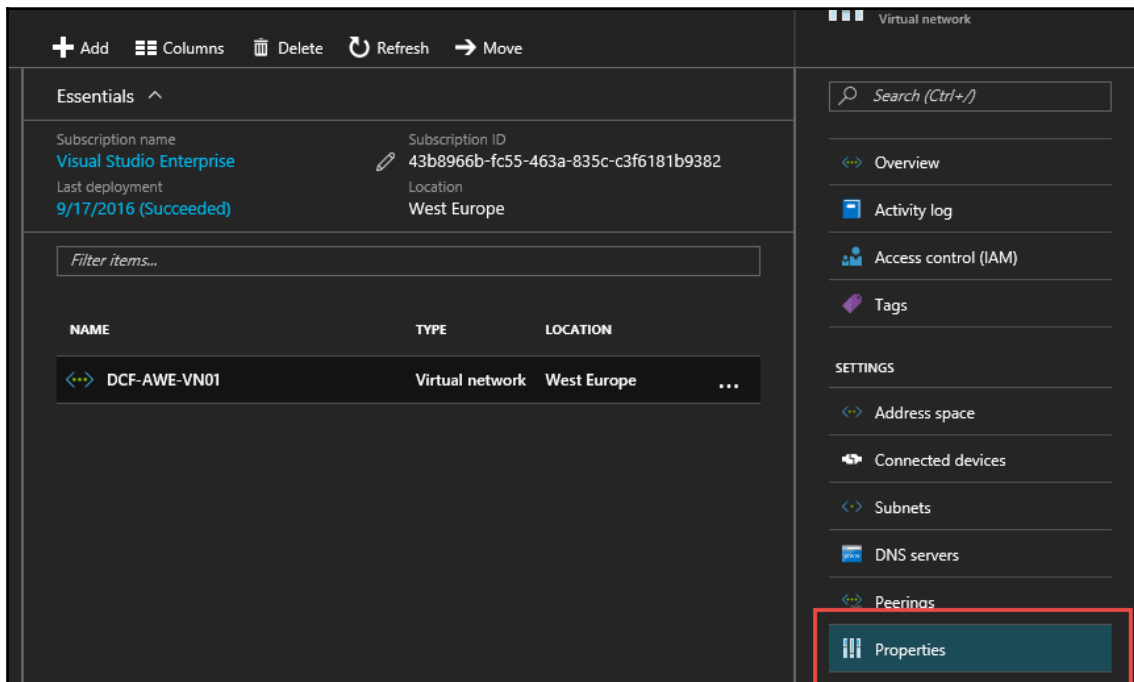
The following are the steps to prepare the deployment:

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

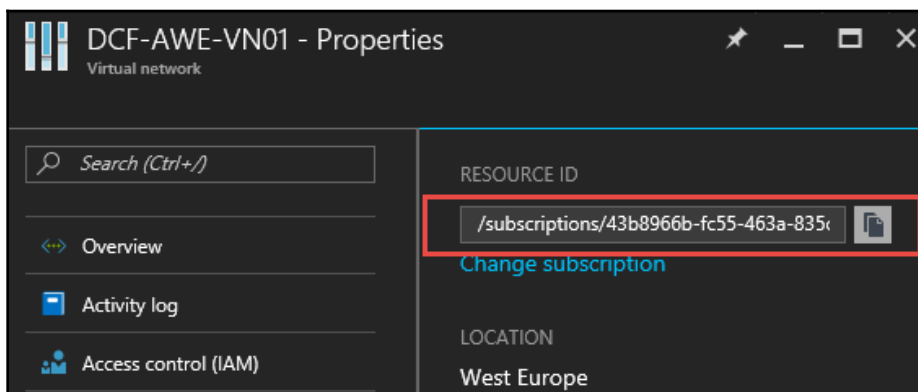


2. Both need to use different IP subnetwork address ranges. In my case, I use 10.0.0.0/16 for **subnet A** and 17.0.0.0/16 for **subnet B**.

- 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

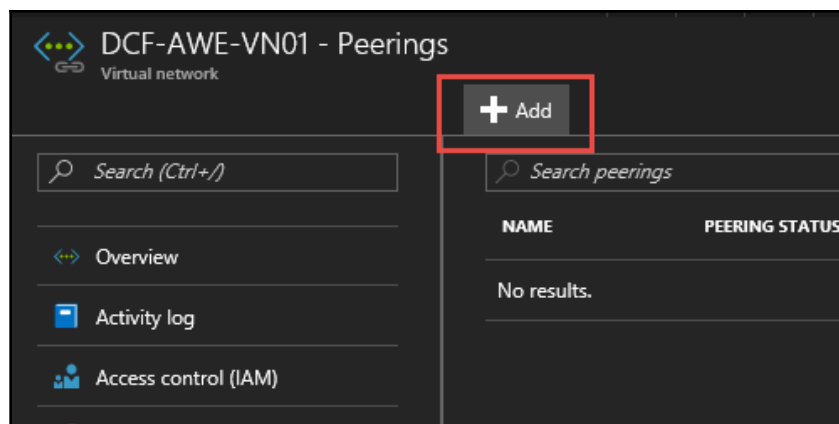
For configuring the VNet peering, perform the following steps:

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**:

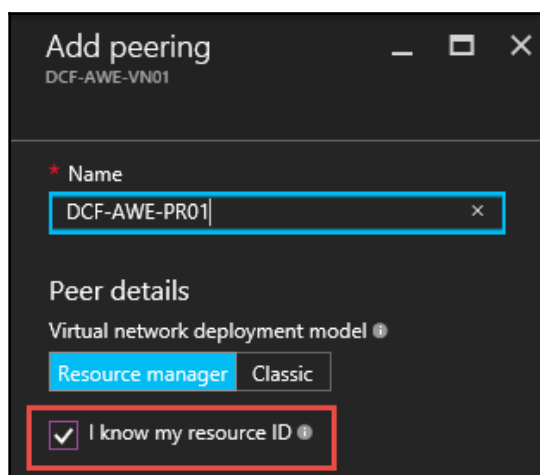
The screenshot displays the Azure portal interface for the 'DCF-AWE-VN01 - Peerings' page. The 'Essentials' section provides details about the subscription, including the name 'Visual Studio Premium with MSDN' and the ID '8a8a0b11-6c01-4199-ba5e-2cad700a1c54'. Below this, a table lists various virtual network resources. The 'DCF-AWE-VN01' resource is highlighted with a red box. On the right side, the 'SETTINGS' sidebar is visible, with the 'Peerings' option selected and highlighted by a red box.

NAME	TYPE	LOCATION
DCF-AWE-GW01-DCF-AWE-LG01	Connection	West Europe
DCF-AWE-LG01	Local network...	West Europe
DCF-AWE-GW01-PIP	Public IP addr...	West Europe
route	Route table	West Europe
DCF-AWE-GW01	Virtual network...	West Europe
DCF-AWE-VN01	Virtual network	West Europe

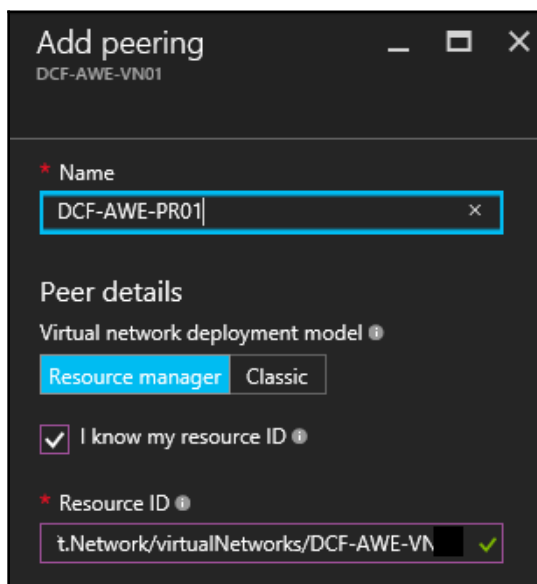
2. In the **Peerings** blade, click on **+Add**:



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:



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

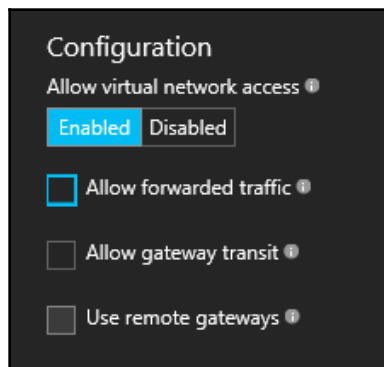
* Name
DCF-AWE-PR01

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:



Configuration

Allow virtual network access ⓘ
Enabled Disabled

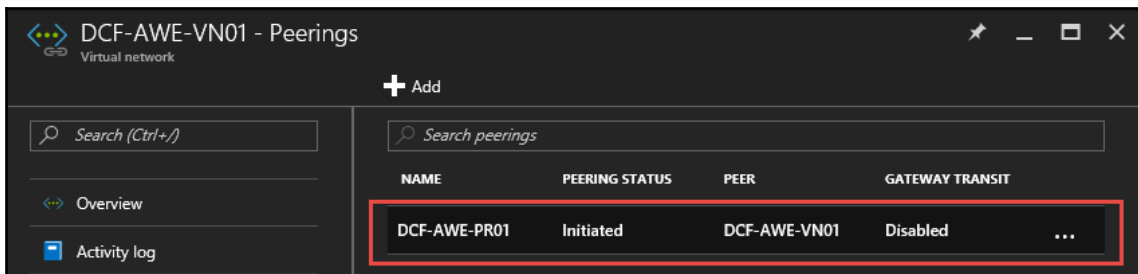
☐ Allow forwarded traffic ⓘ

☐ Allow gateway transit ⓘ

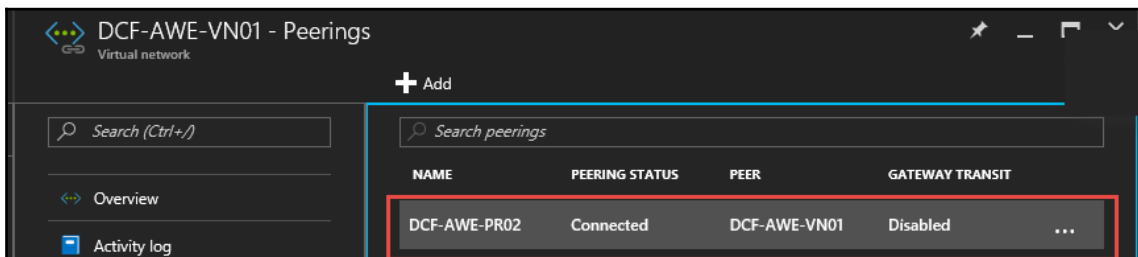
☐ Use remote gateways ⓘ

- **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.

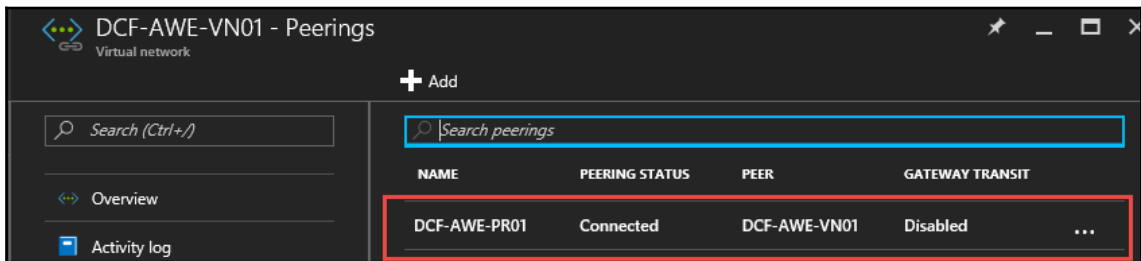
- **Allow forwarded traffic:** Allows traffic not originated from the peered VNet to be accepted or dropped.
 - **Allow gateway transit:** Allows the peer VNet to use your VNet gateway.
 - **Use remote Gateways:** Uses your peer's VNet gateway. The peer VNet must have a gateway configured and a `AllowGatewayTransit` 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:



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

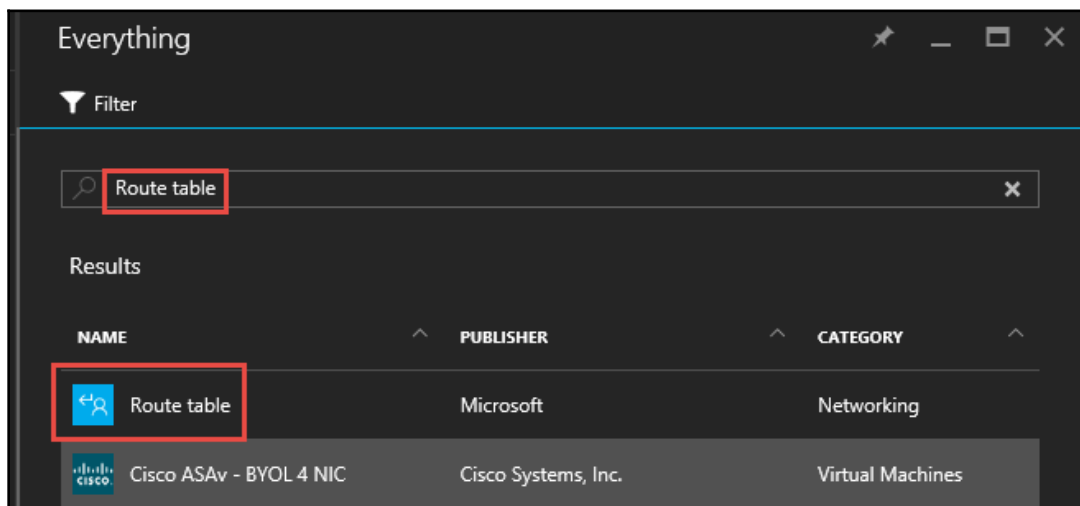


Currently VNet peering is still in preview and under development. Microsoft will extend this service in the near future to support VNet peering with Microsoft Azure public services such as Storage or Azure SQL too.

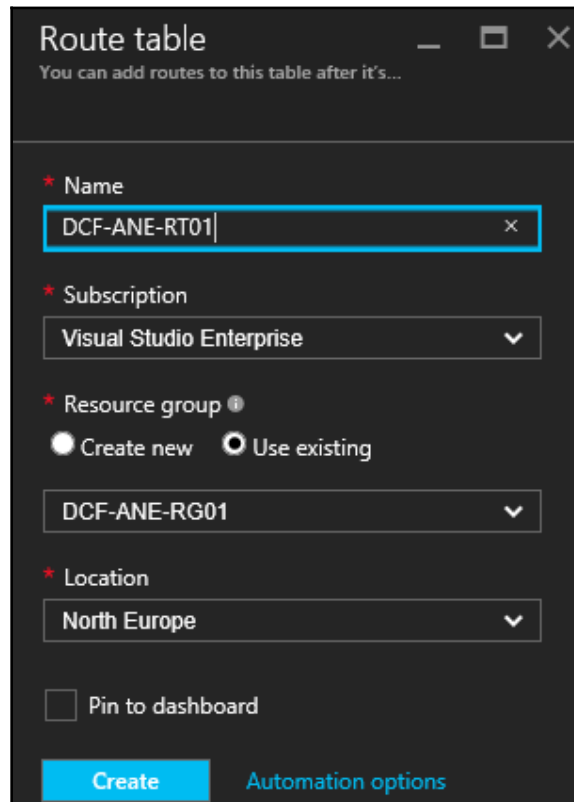
Configuring custom routes

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



The screenshot shows the 'Route table' creation form in the Azure portal. The form is titled 'Route table' with a subtitle 'You can add routes to this table after it's...'. It contains several fields and options:

- Name:** A text input field containing 'DCF-ANE-RT01', highlighted with a red border.
- Subscription:** A dropdown menu showing 'Visual Studio Enterprise'.
- Resource group:** A section with two radio buttons: 'Create new' (selected) and 'Use existing'. Below them is a dropdown menu showing 'DCF-ANE-RG01'.
- Location:** A dropdown menu showing 'North Europe'.
- Pin to dashboard:** An unchecked checkbox.
- Buttons:** A blue 'Create' button and a link 'Automation options'.

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

The screenshot shows the Azure portal interface for the resource group 'DCF-ANE-RG01'. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, SETTINGS (Quickstart, Resource costs, Deployments, Properties, Locks, Automation script), and SUPPORT + TROUBLESHOOTING (New support request). The main area displays the 'Essentials' section with subscription details and a table of resources.

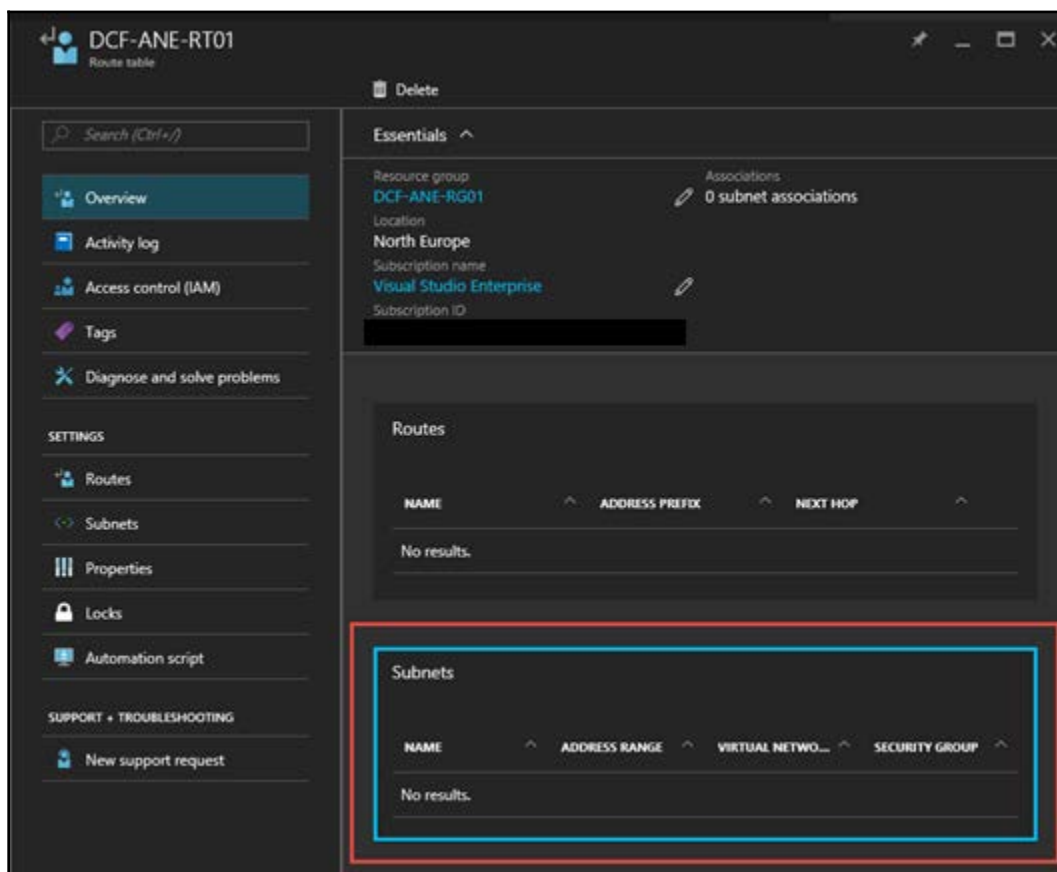
Subscription Details:

- Subscription name: Visual Studio Enterprise
- Subscription ID: 43b8966b-fc55-463a-835c-c3f6181b9382
- Last deployment: 9/14/2016 (Deploying)
- Location: North Europe

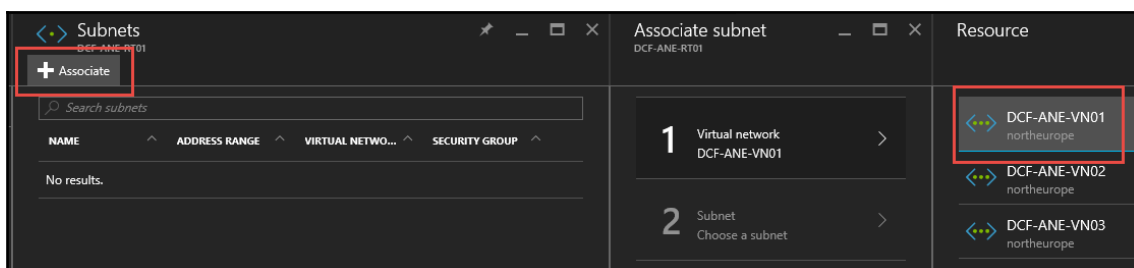
Resources Table:

NAME	TYPE	LOCATION	
DCF-ANE-GW01-DCF-ANE-LG01	Connection	North Europe	...
DCF-ANE-GW01-DCF-ANE-LG02	Connection	North Europe	...
DCF-ANE-EX01	ExpressRoute...	North Europe	...
DCF-ANE-LG01	Local network...	North Europe	...
DCF-ANE-LG02	Local network...	North Europe	...
DCF-ANE-GW01-PIP	Public IP addr...	North Europe	...
DCF-ANE-GW02-PIP	Public IP addr...	North Europe	...
DCF-ANE-GW03-PIP	Public IP addr...	North Europe	...
DCF-ANE-RT01	Route table	North Europe	...

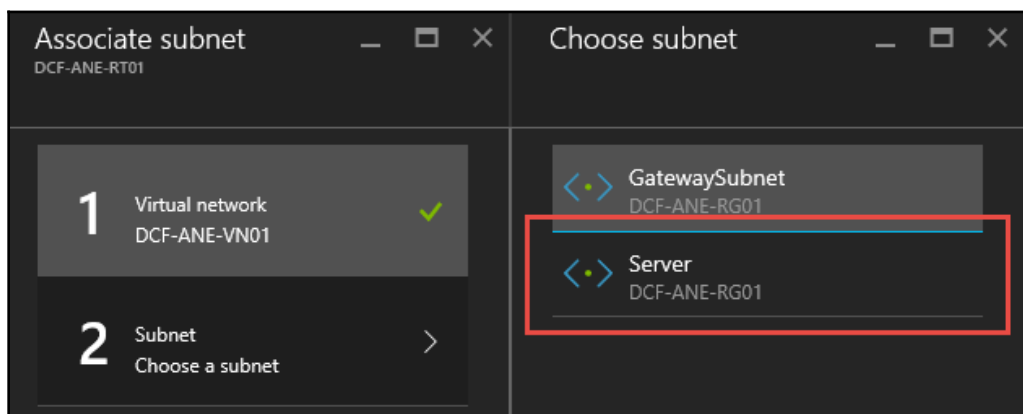
4. The **Settings** blade opens. Here you click first on **Subnets** to open the detail blade to associate subnets to that routing table:



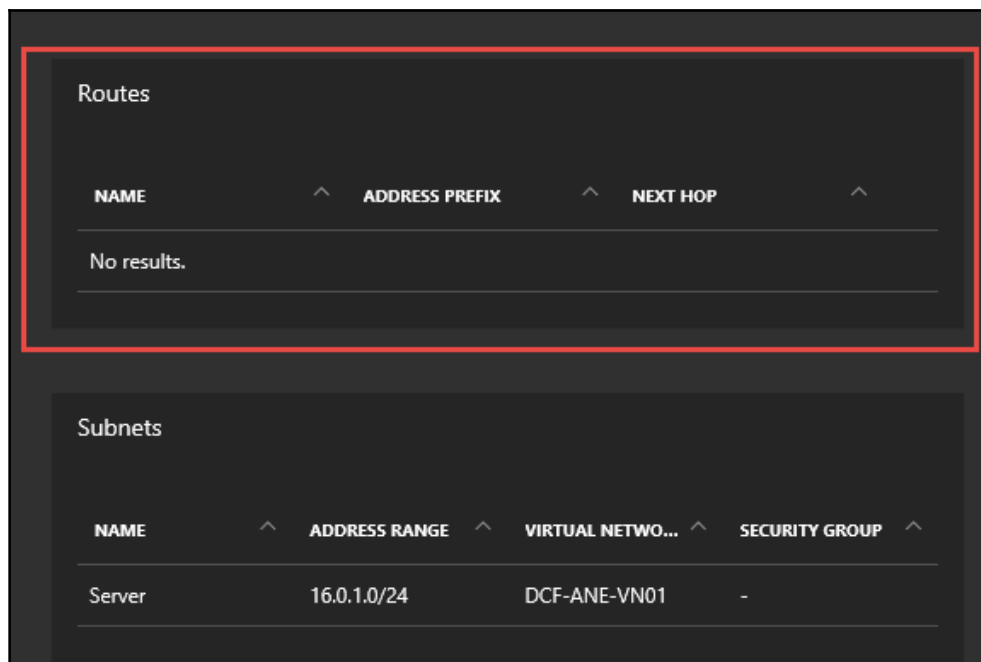
5. In the details blade, you click on **Associate** and add the Azure VNet where the route table should be applied to:



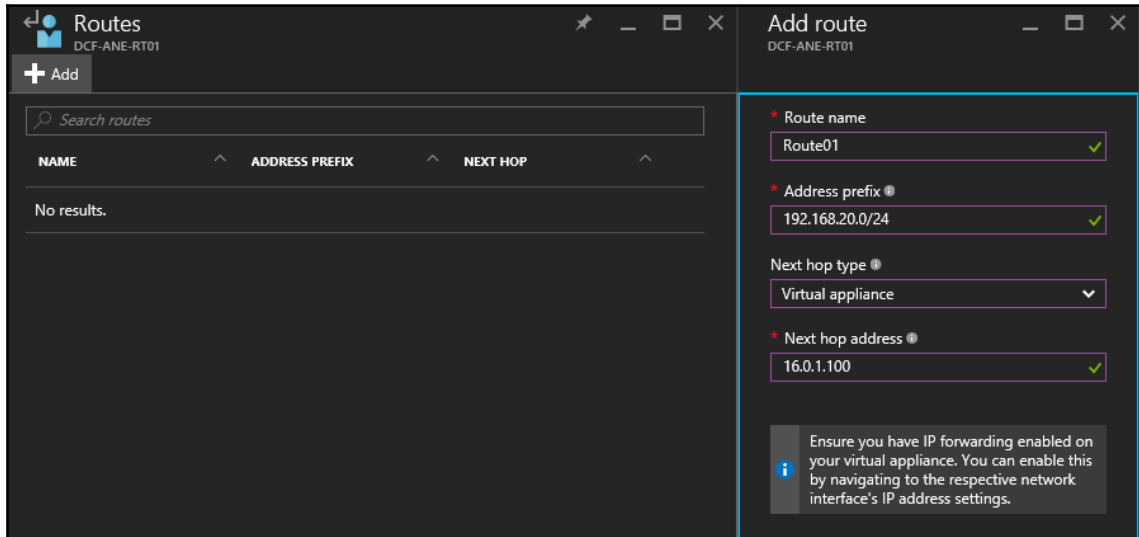
- Then choose the subnet where you want to apply the table to and click **OK** to commit:



- 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:

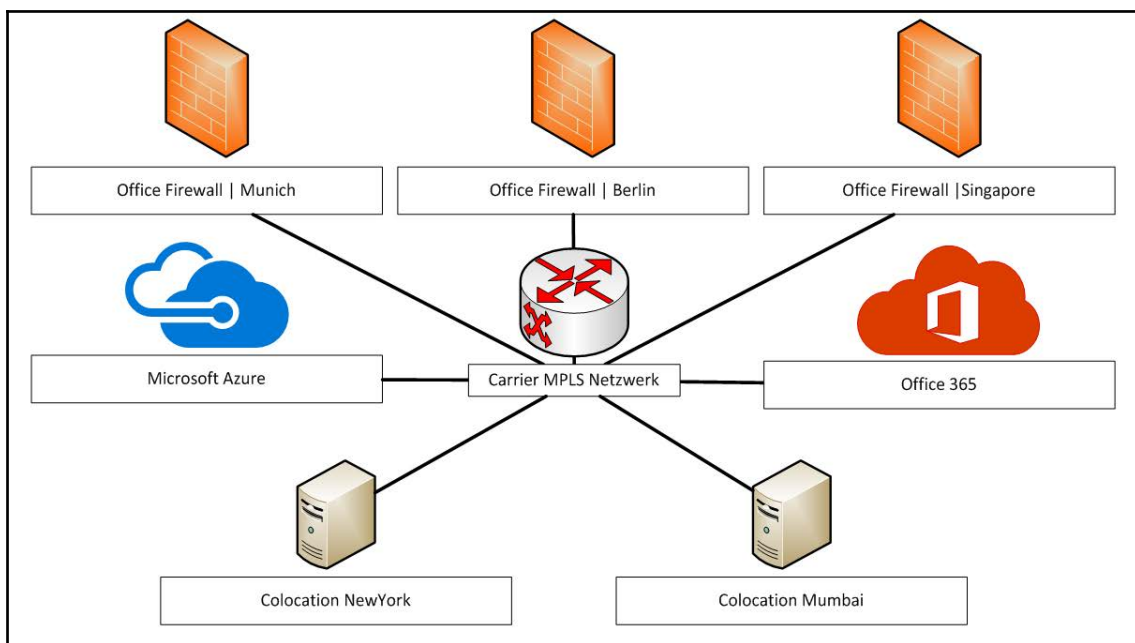


9. It will take a moment until the route will be applied to the subnet.

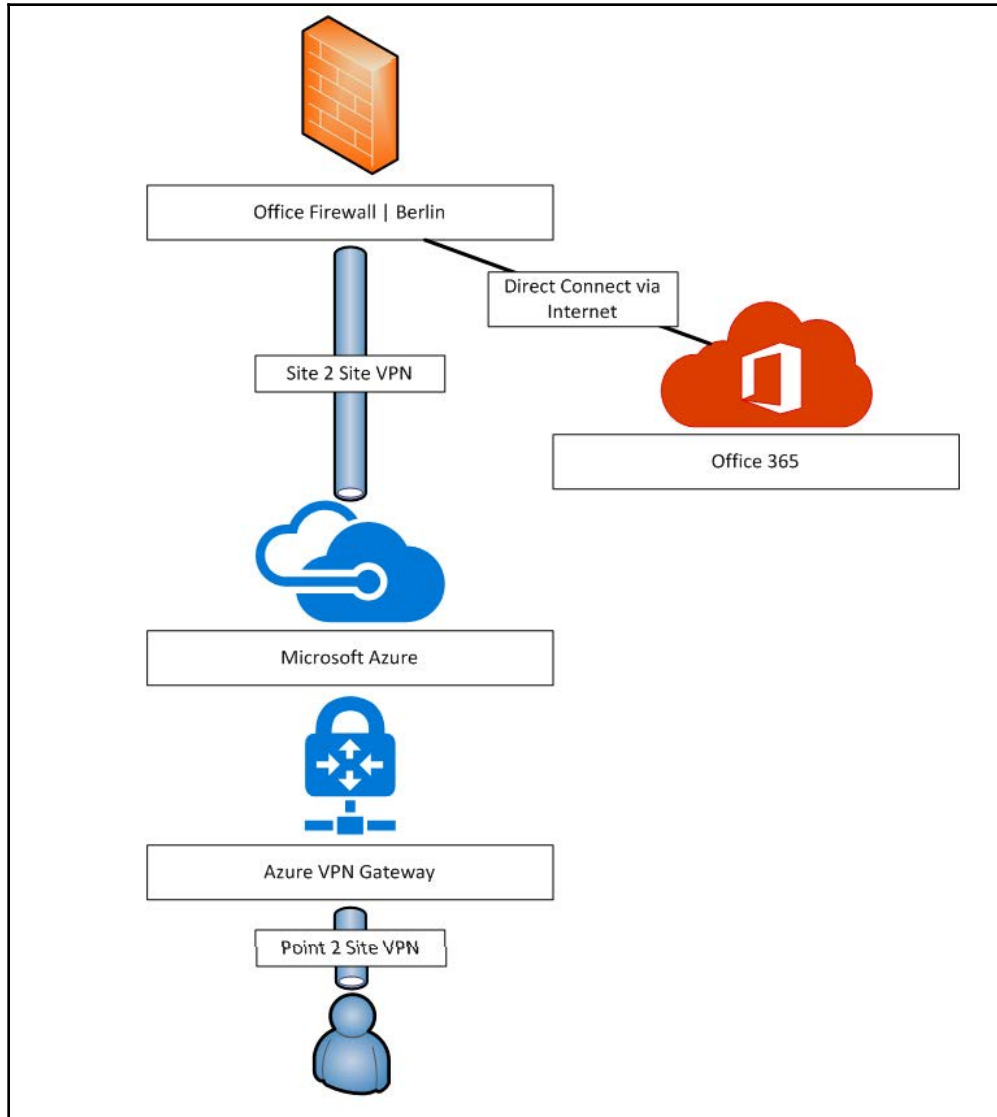
Common Azure network architectures

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.

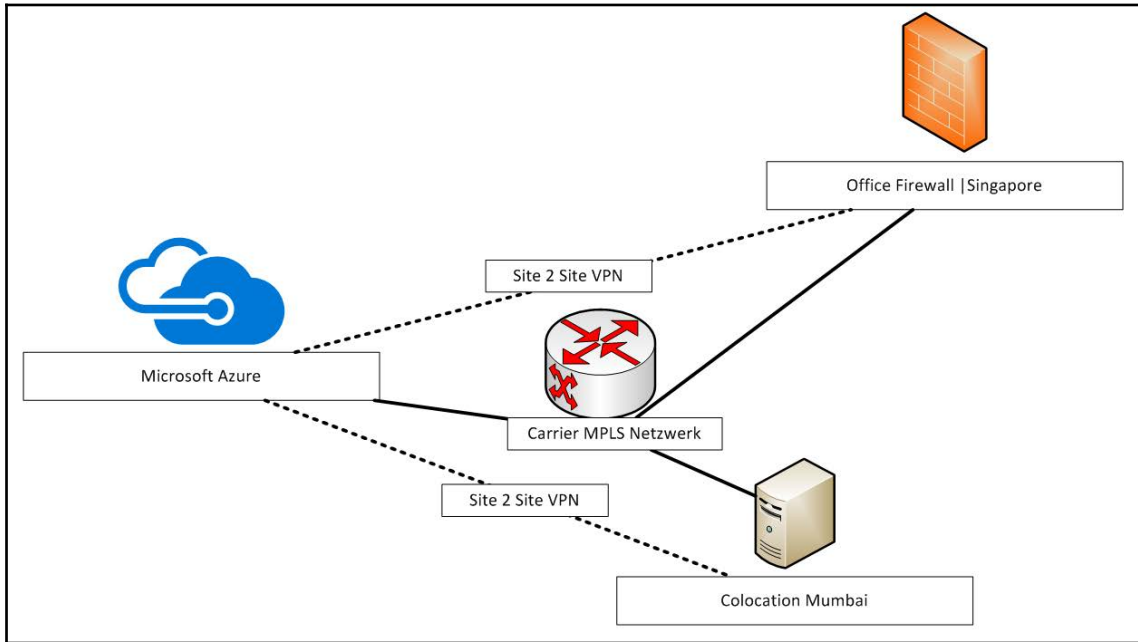
The following diagram shows a short abstract of such an environment:



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.

Questions

1. Does Azure offer only Microsoft network services or 3rd party appliances too?
2. Does a virtual network gateway only support VPN?
3. Azure ExpressRoute is VPN Solution? True or false?
4. What is the current bandwidth for ExpressRoute Direct?
5. Does Azure offer a web application firewall?
6. How is Microsoft SD-WAN solution named?
7. Does Azure support peering of virtual networks across different regions?

Further reading

Within the following books, you can find more information about what we learned in this chapter:

- **Hands-On Cloud Administration in Azure:** <https://www.packtpub.com/virtualization-and-cloud/hands-cloud-administration-azure>
- **Hands-On Networking with Azure:** <https://www.packtpub.com/virtualization-and-cloud/hands-networking-azure>
- **Infrastructure as a Service Solutions with Azure [Video]:** <https://www.packtpub.com/virtualization-and-cloud/infrastructure-service-solutions-azure-video>

6 Implementing Azure Storage

Just like you plan the network in your datacenter or company, you need to do the same in Azure. Nearly every service in Azure is related to 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.

In this chapter, Azure Storage management is discussed. The key takeaways are how and when to implement and integrate the different Azure Storage types.

In this chapter, we will cover the following topics:

- 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 cover deep backup, Azure site recovery, or StorSimple topics. We will set up some basic storage configurations in this chapter.

Storage accounts

Azure Storage implements the following four services:

- **Blob storage:** Blob storage stores unstructured object data, which can be text or binary data.
- **Table storage:** Structured datasets are stored in Table storage. Table storage is a NoSQL key attribute data store, enabling rapid development and fast access to data.
- **Queue storage:** In addition to providing reliable messaging for workflow processing, Queue storage makes communication between segments of cloud services available.
- **File storage:** Using the standard **Server Message Block (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-premise applications can obtain file data.

The 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 as follows:

- **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 to this 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 this 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.
- **Archiving tier:** The archive tier is made for data that can work with several hours of retrieval latency and will remain in the archive for a minimum of 180 days. The archive tier is the most cost-effective opportunity for storage data, but access to that data is more expensive than in the hot or cool tiers.

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 that you have an Azure subscription. The subscription gives 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 **service level agreement (SLA)** for both is 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. You will find more detailed information about the available redundancy options later in the chapter.

General-purpose storage v1 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 accounts* section later in this chapter.

General-purpose storage v2 accounts

General-purpose storage v2 accounts have some of the same features as the v1 version, but they combine them with the tiering options of Blob storage accounts. In addition to that, v2 accounts also offer Geo Redundancy Storage for data stored in that account. V1 accounts only offer **Read Access Geo Redundant Storage (RA-GRS)**.



Geo Redundancy Services enable customers to access and write data from different locations. Azure will take care of data and replication. So, there is no longer any need to replicate data or files through virtual machines or other external services.

Azure File Sync/Storage Sync services

Azure File Sync is an Azure service that enables customers to centralize file shares in Azure files, while keeping the flexibility, performance, and compatibility of a local file server. Azure File Sync enables Windows Servers to perform a quick cache of an Azure file share. Any protocol that's available on Windows Server to access data locally, including SMB, NFS and FTPS, can be used. Those caches have no limit and customers can have as many local file server caches as they need.

To use Azure File Sync, you need at least Windows Server 2012 R2 or later and PowerShell 5.1 installed on the server. Azure File Sync supports also **cloud tiering**, ACL replication between cloud and on-premise, as well as the integration of Azure AD identity management.

Azure Data Lake

Azure Data Lake Storage is a set of capabilities built for big data analytics, on top of Azure Blob storage. It enables customers to interface with data using both file system and object storage paradigms. This makes Data Lake Storage the cloud-based multi-modal storage service, allowing applications to extract analytics value from all customer's available data.

Replication and redundancy

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

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

Locally redundant storage (LRS)

LRS means that the data is held three times in a datacenter in a region. The LRS manages three copies of the customer's data to protect it from hardware failures. LRS does not protect workloads from the failure of a whole datacenter; it only replicates data within an Azure Storage scale Unit from about 1,000 physical machines.

Zone-redundant storage (ZRS)

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 (GRS)

An even higher durability can be achieved with 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 that 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 (RA-GRS)

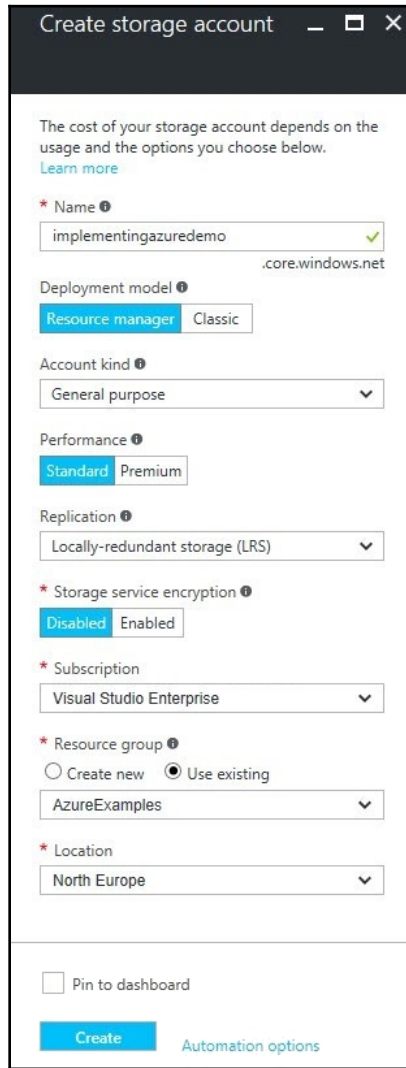
Replication to a secondary geographic location is provided with RA-GRS. The customer holds read access to the data, maintained in the secondary location. Access from the primary and the secondary region is possible. RA-GRS is the default option for your storage account on creation.

The following is an overview of the redundancy options in Azure Storage:

Replication strategy	LRS	ZRS	GRS	RA-GRS
Data is replicated across multiple datacenters	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:



The screenshot shows the 'Create storage account' form in the Azure portal. The form is titled 'Create storage account' and includes a close button. Below the title, there is a note: 'The cost of your storage account depends on the usage and the options you choose below. [Learn more](#)'. The form contains several sections with required fields marked with an asterisk (*):

- Name:** A text input field containing 'implementingazuredemo' with a green checkmark icon. Below the field, the text '.core.windows.net' is displayed.
- Deployment model:** Two buttons: 'Resource manager' (selected) and 'Classic'.
- Account kind:** A dropdown menu showing 'General purpose'.
- Performance:** Two buttons: 'Standard' (selected) and 'Premium'.
- Replication:** A dropdown menu showing 'Locally-redundant storage (LRS)'.
- Storage service encryption:** Two buttons: 'Disabled' (selected) and 'Enabled'.
- Subscription:** A dropdown menu showing 'Visual Studio Enterprise'.
- Resource group:** Radio buttons for 'Create new' and 'Use existing' (selected). Below the radio buttons, a dropdown menu shows 'AzureExamples'.
- Location:** A dropdown menu showing 'North Europe'.

At the bottom of the form, there is a checkbox labeled 'Pin to dashboard' which is unchecked. Below the checkbox is a blue 'Create' button and a link for '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 goal 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. The `Name` parameter is the planned name for the storage account, the `location` parameter is the Azure deployment region, and the `SkuName` parameter is a mix of the performance and replication settings 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

Azure Storage services

As previously mentioned, the difference between Blob storage and general-purpose storage is to be found in the purpose for which they're used. While 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 on 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 that can 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:** Block blobs are utilized for streaming and storing cloud objects. They are best used for storing documents, media files, backups, and so on.
- **Append blobs:** 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. An append blob's field of application consists of logging, in which data has to be written to the end of the blob.
- **Page blobs (disks):** The third type of blob is the page blob. In most cases, page blobs are used to store **Infrastructure as a Service (IaaS)** disks. They support random writes. This means that an Azure IaaS VM **virtual hard disk (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 to or exported from the datacenter. 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.

Azure Storage blobs have access types; those access types define how your blobs can be accessed publicly:

Type	Description
Private	If this type is selected, 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 this type is selected, only blobs can be accessed from the outside with read permissions.
Container	If this type 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.

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, and 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 on 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.

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 individual 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 Queues storage. 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:

- **Azure Queues:** This is a part of Azure Storage and the one that we are working with in this chapter
- **Service Bus queues:** This 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.

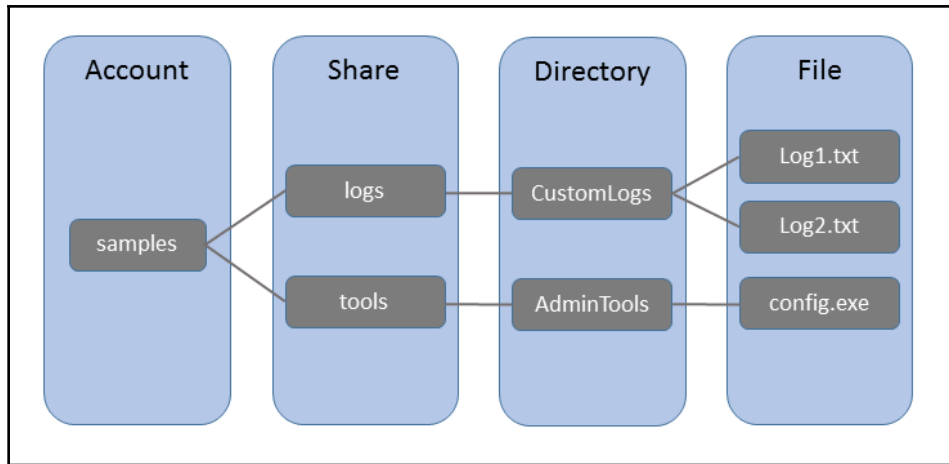


For more information on how to choose the right queue solution, visit <https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-azure-and-service-bus-queues-compared-contrasted>.

File storage services

Currently, the most interesting type of storage for the IT professional is **Azure 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 preceding diagram, 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. But there are also several downsides to the current file share implementation. Currently, there are only two important downsides to consider:

- **Storage limit:** In addition 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.
- **Latency between location and Azure Storage:** Because we are still talking about SMB traffic, and with that a protocol which is not WAN-optimized, we are struggling with latencies. Normally, you say every latency above 16 ms is not feasible for an SMB share. To solve that issue, you should consider using Azure File Sync and keep a file server in your location.

The only ways to authenticate for file share access are access keys and **shared access signatures (SAS)**. An SAS is basically a link 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. An SAS should be used for untrusted external staff, temporary development test, or for customers for tests.

At the time of writing, Microsoft has started to work on a premium version of Azure file shares. The following table shows the most significant changes to standard file shares:

Resource	Standard file share	Premium file share
Minimum size of a file share	(no minimum; pay as you go)	100 GB
Max IOPS per share	1000 IOPS	5120 IOPS baseline 15360 IOPS with burst
Target throughput for single file share	Up to 60 MiBps	Up to 612 MiBps(provisioned)

Access keys will be discussed in the next section.



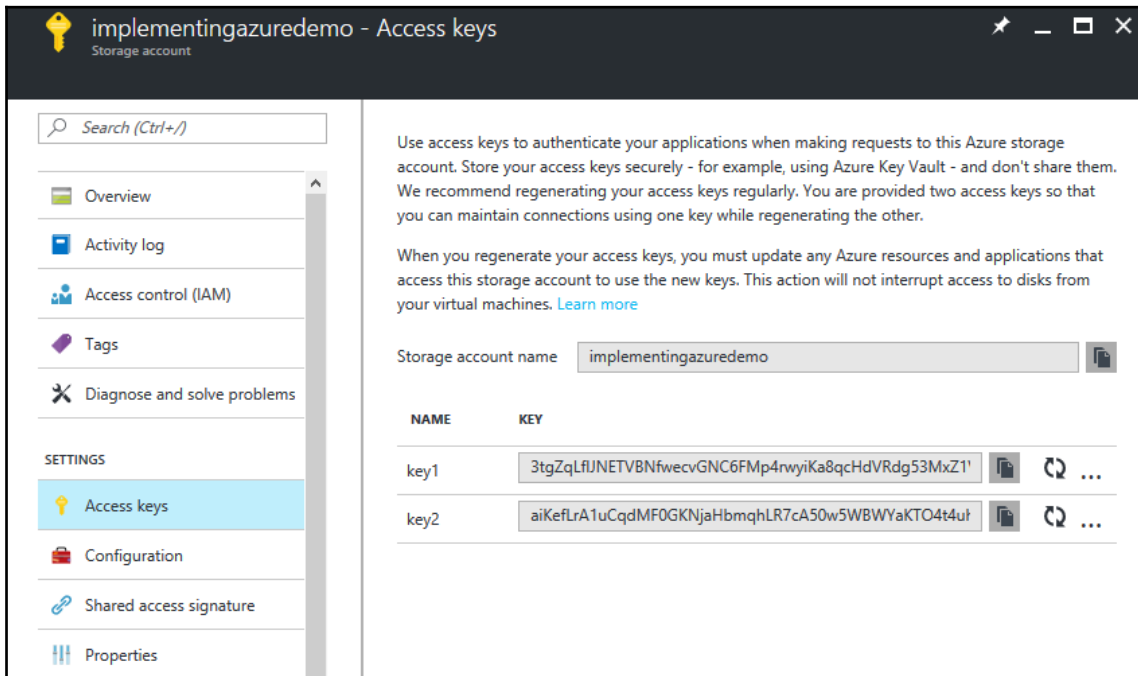
Microsoft released a tool for working with blob, file, and table storage called **AzCopy**. Its main purpose is to transfer data from and to Azure Storage. It can be found at <http://aka.ms/azcopy>.

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 RESTful 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, and for this reason they must be kept safe all the time. An SAS is also created based on the storage accounts access keys. That means that 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 access keys on a regular basis, keys should be rotated to avoid any downtime. Key regeneration does not influence 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:

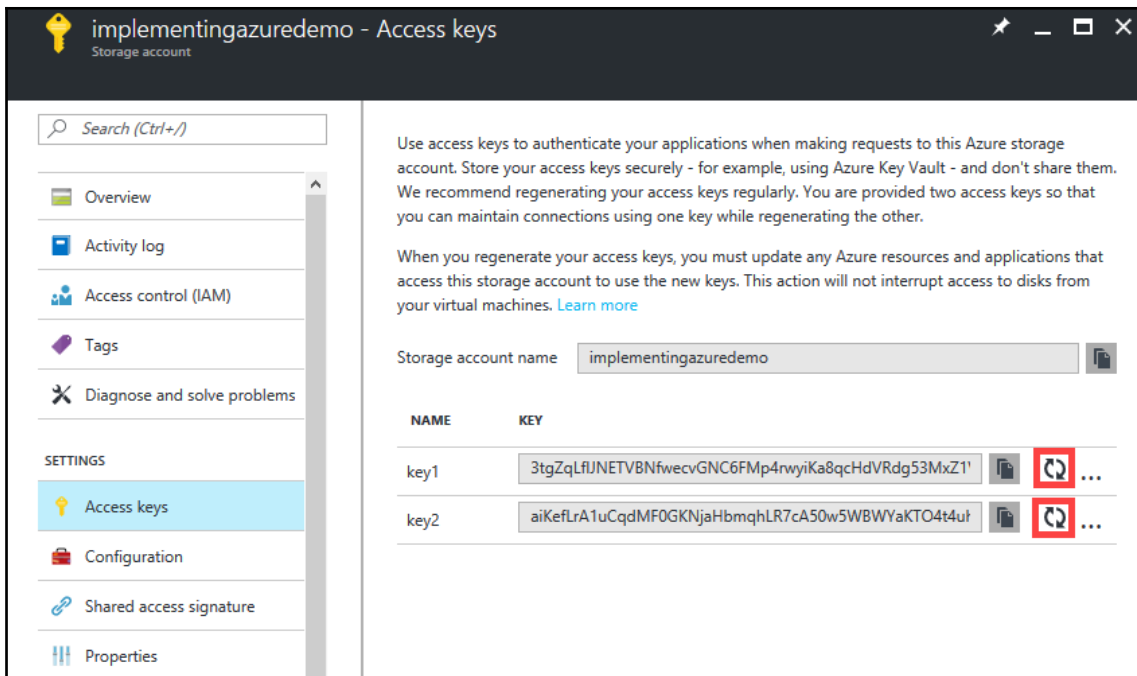


Storage account access key overview

They can also be received with PowerShell with the following command:

```
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 screenshot:



Regenerating keys with the marked buttons

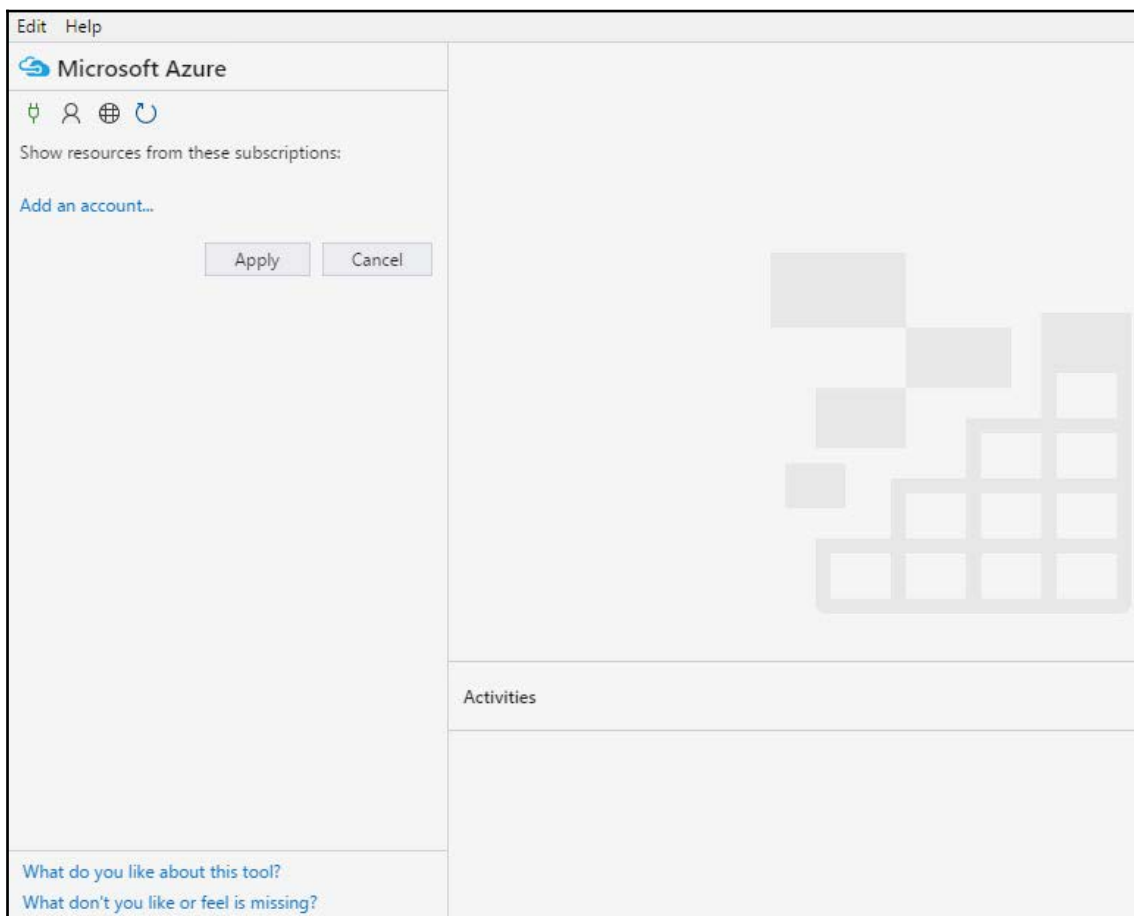
To regenerate a key using PowerShell, the following **cmdlet** is used:

```
New-AzureRmStorageKey -ResourceGroupName "MyResourceGroup" -AccountName
"MyStorageAccount" -KeyName "key1"
```

Exploring Azure Storage with Azure Storage Explorer

Microsoft Azure Storage Explorer is a free, available graphical tool for managing Azure Storage without the portal or PowerShell. Azure 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/>.

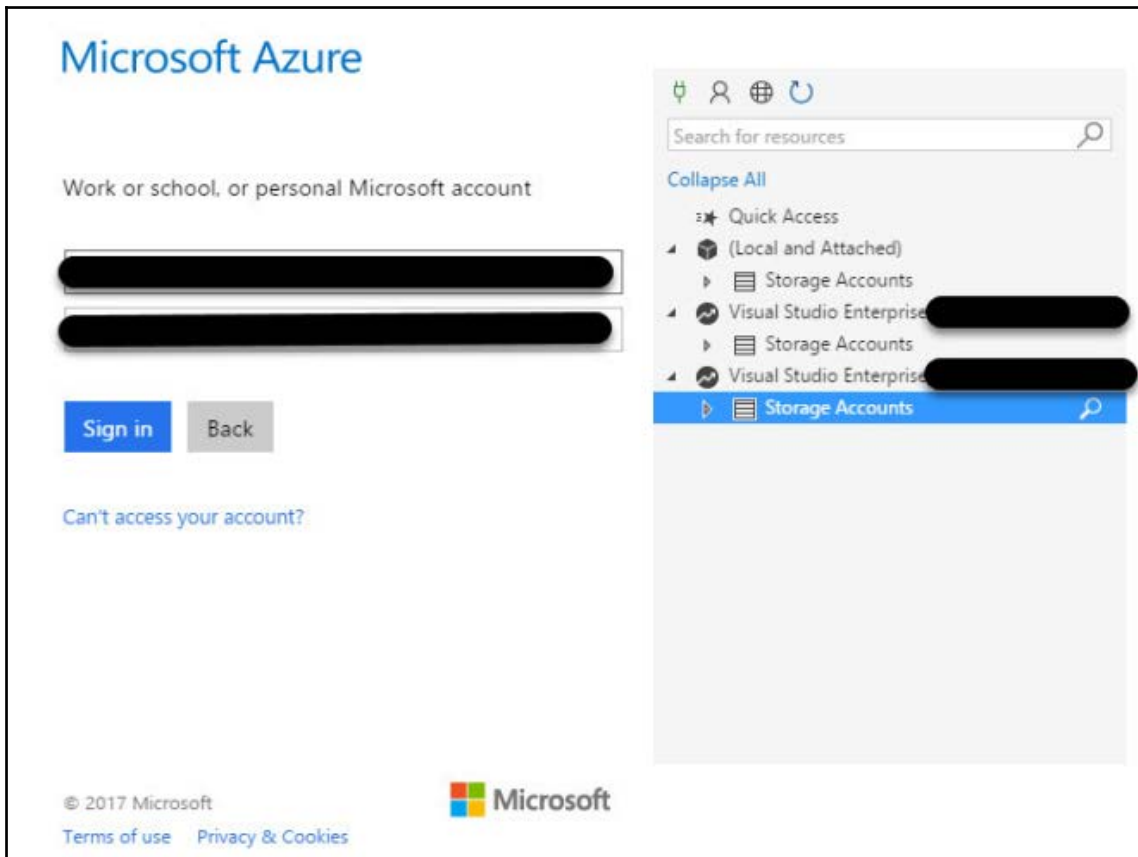
After downloading and installing Azure Storage Explorer, the following screenshot is shown:



Storage explorer dashboard

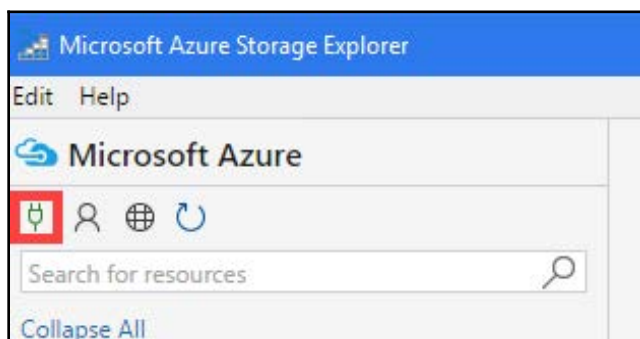
First, a storage account needs to be connected. There are two ways in which one can connect to a storage account:

1. Add a Microsoft or company account. This is the easiest solution. Azure Storage Explorer asks for credentials for a Microsoft or company account that has an active subscription. After typing the username and password, Azure Storage Explorer lets you to select which subscriptions should be managed. By default, all are selected. After clicking on the **Apply** button, the setup is done, and Azure Storage nodes can be explored:



Azure login dialogue (left), and Storage Explorer subscriptions overview (right)

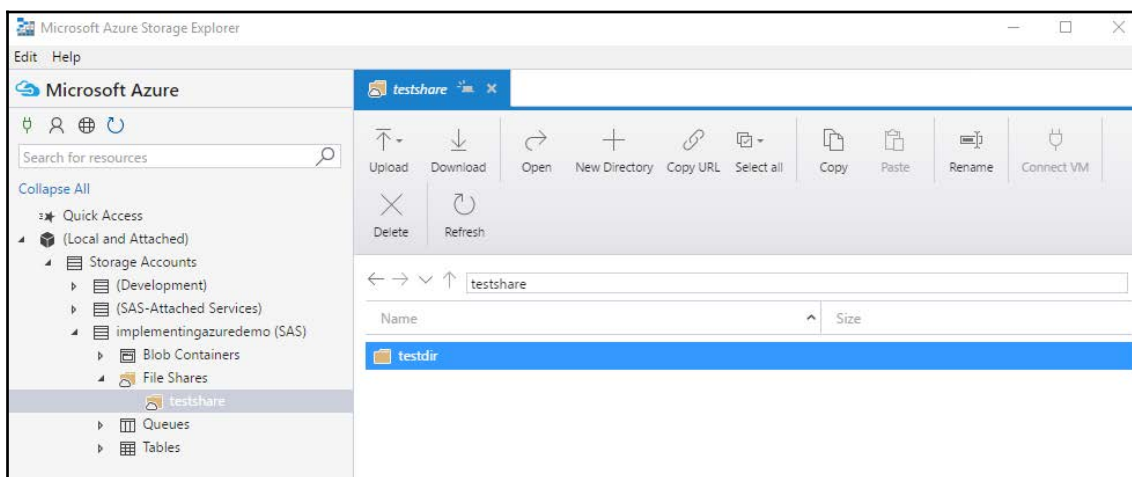
2. Add a connection string, SAS URI, or account. It's also possible to connect to your storage accounts by typing a storage account key or SAS:



Connecting with URI, SAS, or the key button

The plug symbol as shown in the preceding screenshot opens a dialog that 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 Azure Storage Explorer. Browsing through the **File Shares** section, the previously-created file share can be found:



The connected storage account node

Premium storage accounts

When it comes to performance, you can decide between standard or a premium storage account. For most workloads, standard accounts are more than suitable, but in some cases, more I/O-intensive applications need very fast storage. For this use case, the premium storage account was introduced. Premium storage is fully backed by SSD tiers and provides high-performance and 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 **input/output operations per second (IOPS)** and 2,000 MBps disk throughput.

Microsoft examples of 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 from Azure, charges apply.

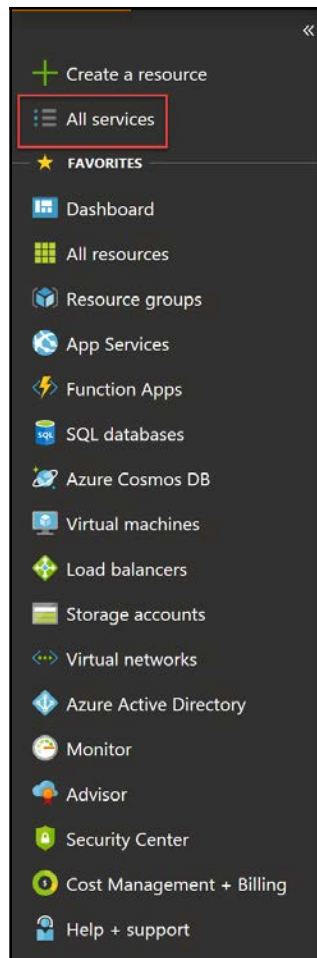


For more information on pricing, use the Azure pricing calculator, located at <https://azure.microsoft.com/en-us/pricing/calculator/>.

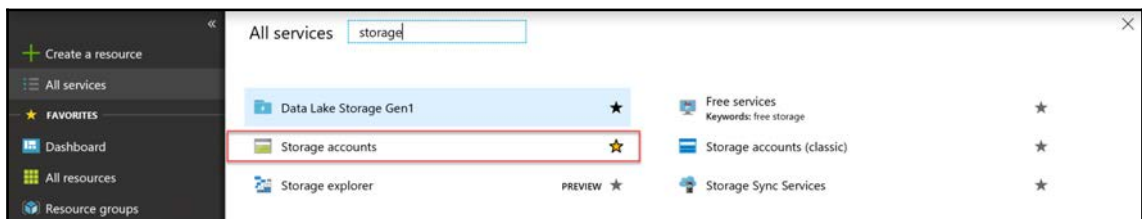
How to deploy a storage account?

Now that we have learned so much about accounts and storage, let's deploy a storage account within our Azure subscription with the following steps:

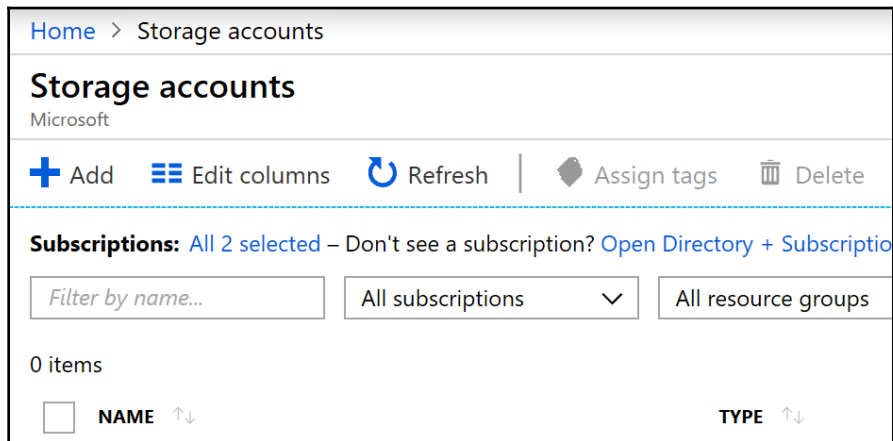
1. Go to the Azure portal and navigate to the **All services** section as shown in the following screenshot:



2. Within the next blade, search for `storage` as shown in the following screenshot:



3. Click on the **Storage accounts** option to open new blade to add a storage account to your subscription. Click on the **+Add** button to continue:



4. In the following blade, we will start with a basic storage configuration:
 - Select the subscription for your deployment
 - Select or create a resource group
 - Name the storage account
 - Select the region where you want to locate the storage account
 - Select your performance tier
 - Select your kind of storage account; in our example we will use a general purpose v2 account
 - Select the replication option

- Depending on your account type, you will have additional options such as the Access Tier, as shown in the following screenshot:

Create storage account

Basics **Advanced** Tags Review + create

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below. [Learn more](#)

PROJECT DETAILS

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

* Subscription Visual Studio Enterprise 1

* Resource group (New) Packt 2
[Create new](#)

INSTANCE DETAILS

The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using the classic deployment model instead. [Choose classic deployment model](#)

* Storage account name ① packt 3

* Location West Europe 4

Performance ① ☒ Standard ☐ Premium 5

Account kind ① StorageV2 (general purpose v2) 6

Replication ① Locally-redundant storage (LRS) 7

Access tier (default) ① ☐ Cool ☒ Hot 8

5. Afterward, click on the **Next: Advanced** > button to continue with the configuration:

Review + create Previous Next : Advanced >

6. Within the next blade, we configure additional security features, such as encryption, VNet access, or access to the namespace from Azure Data Lake Gen2:

Basics **Advanced** Tags Review + create

SECURITY

Secure transfer required ? ☐ Disabled ☒ Enabled

VIRTUAL NETWORKS

Allow access from ☒ All networks ☐ Selected network

? All networks will be able to access this storage account. [Learn more](#)

DATA LAKE STORAGE GEN2 (PREVIEW)

Hierarchical namespace ? ☒ Disabled ☐ Enabled

7. After configuring advanced settings, we only need to add some more tags, for example the cost center. To do that, click the **Next : Tags >** button:

Review + create Previous Next : Tags >

8. As you already know, tags are very important in Azure. They help with automation, development, cost management, and even more. So, it is very important to use those tags and optimize Azure resource management:

Basics Advanced **Tags** Review + create

Tags are name/value pairs that enable you to categorize resources and view consolidated billing by applying the same tag to multiple resources and resource groups. [Learn more](#)

Note that if you create tags and then change resource settings on other tabs, your tags will be automatically updated.

☐	KEY	VALUE	RESOURCE TYPE
☒	Costcenter ✓	0815 ✓	Storage account ▼
☐			Storage account ▼

9. After setting the necessary tags, we only need to validate and review our configuration, and finally create the storage account. To do that, click on the **Next : Review + create >** button:



10. When you have passed the validation, you can click on the **Create** button and the deployment of the account will start:

Create storage account

✓ Validation passed

[Basics](#) [Advanced](#) [Tags](#) [Review + create](#)

BASICS

Subscription	Visual Studio Enterprise
Resource group	(new) Packt
Location	West Europe
Storage account name	packt
Deployment model	Resource manager
Account kind	StorageV2 (general purpose v2)
Replication	Locally-redundant storage (LRS)
Performance	Standard
Access tier (default)	Hot

ADVANCED

Secure transfer required	Enabled
Allow access from	All networks
Hierarchical namespace	Disabled

TAGS

Costcenter	0815 (Storage account)
------------	------------------------

[Create](#) [Previous](#) [Next](#) [Download a template for automation](#)

Microsoft has enhanced the portal experience in the Azure portal. You will now get a response showing the deployment status in the storage **Overview** blade. The status is shown here during deployment:

[Delete](#) [Cancel](#) [Redeploy](#) [Refresh](#)

*** Your deployment is underway

Click the status of your deployment, manage resources, or troubleshoot deployment issues. Pin this page to your dashboard to easily find it next time.



Deployment name: Microsoft.StorageAccount-20181021232041
Subscription: [Visual Studio Enterprise](#)
Resource group: [Packt](#)

DEPLOYMENT DETAILS ([Download](#))
Start time: 21.10.2018 23:48:03
Duration: 22 seconds
Correlation ID: c34f9a90-8844-4fd1-81a7-4484a053d825

RESOURCE	TYPE	STATUS	OPERATION DETAILS
 packt	Microsoft.Storage/storag...	Accepted	Operation details

When the deployment is finished, the status will change and the portal will represent the completed deployment:



Your deployment is complete

[Go to resource](#)

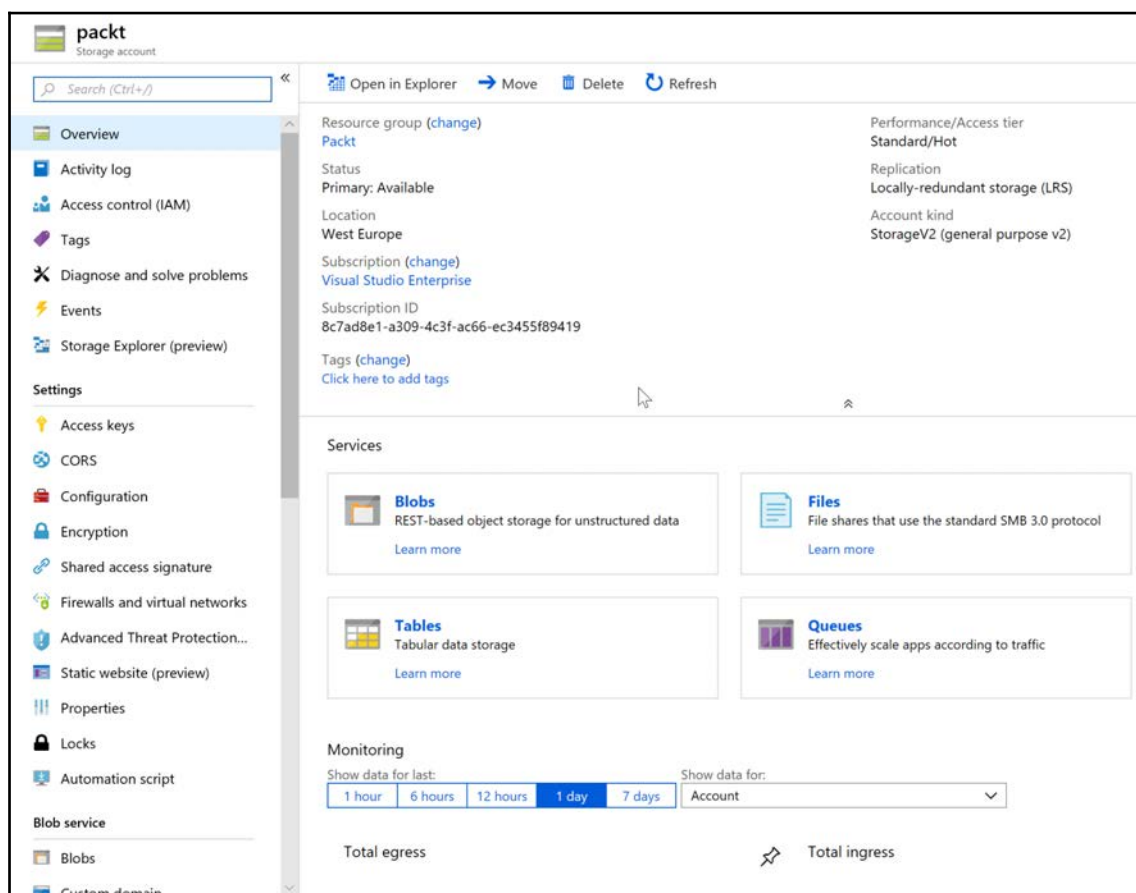


Deployment name: Microsoft.StorageAccount-20181021232041
Subscription: [Visual Studio Enterprise](#)
Resource group: [Packt](#)

DEPLOYMENT DETAILS ([Download](#))
Start time: 21.10.2018 23:48:03
Duration: 33 seconds
Correlation ID: c34f9a90-8844-4fd1-81a7-4484a053d825

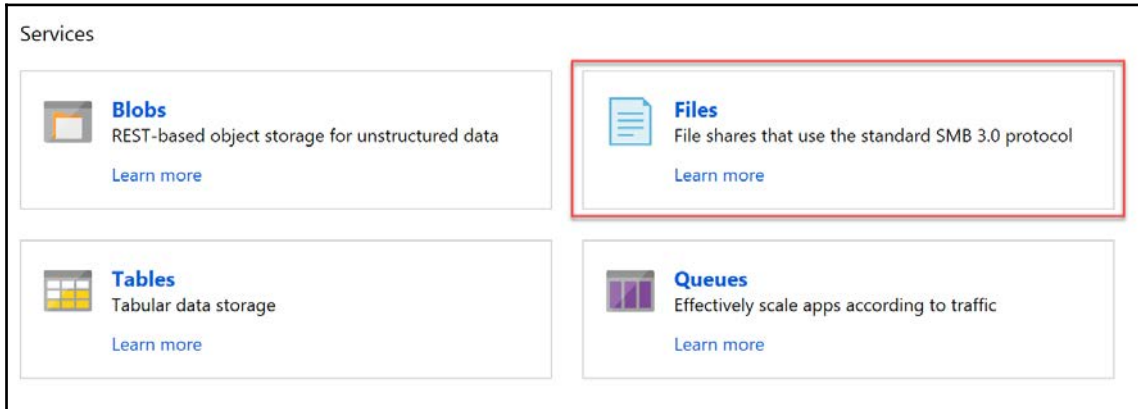
RESOURCE	TYPE	STATUS	OPERATION DETAILS
 packt	Microsoft.Storage/storag...	OK	Operation details

When you now go to the resource, you can create different types of storage service within that newly created storage account:



Lets create an Azure file share as an example:

1. Click on the **Files** section as shown in the following screenshot:



2. In the next blade, click on the **+ File share** button:



3. Now, create a name for the file share and set a quota; remember, the current limit for a share is 5 TB. Afterward, click on the **Create** button:

The screenshot shows a 'File share' creation dialog box. It has a title bar with a close button. Inside, there are two input fields: 'Name' with a red asterisk and a green checkmark, containing the text 'packt'; and 'Quota' with an information icon and a green checkmark, containing the text '500'. Below the 'Quota' field, the unit 'GB' is displayed. At the bottom, there are two blue buttons: 'Create' and 'Discard'.

4. After the file share has been created, you will see the new share in your list of file shares:

 File share  Refresh			
Storage account: packt			
Search file shares by prefix			
NAME	MODIFIED	QUOTA	
packt	22.10.2018 12:06:16	500 GiB	...

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 be familiar with some basic tools and PowerShell cmdlets to interact with Azure Storage.

The next chapter will focus on Azure virtual machines.

Questions

Please answer the following questions:

1. Name the main difference between a general-purpose storage account, v1, and v2.
2. Name the replication options available for Azure Storage.
3. Does storage local redundancy replicate to another Azure datacenter within the region?
4. Which type of storage service does Azure general-purpose storage offer?
5. What is the maximum IOPS for Azure premium file share?
6. What is the minimum Windows Server and PowerShell version for Azure File Sync?
7. Azure Data Lake Storage is a general-purpose storage. True or false?

Further reading

Go through the following links for more information:

- **Learn Azure Storage SDK [Video]:** <https://www.packtpub.com/big-data-and-business-intelligence/learn-azure-storage-sdk-video>
- **Learning Microsoft Azure Storage:** <https://www.packtpub.com/big-data-and-business-intelligence/learning-microsoft-azure-storage>
- **Getting started with Windows Azure Storage [Video]:** <https://www.packtpub.com/virtualization-and-cloud/getting-started-windows-azure-storage-video>

7

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 a **Virtual Machine (VM)** in your environment. To identify the right VM for your services, you need to know the VM types that currently exist and how to implement a VM within your Azure environment.

In this chapter, you will learn the basics about Azure VMs, how to implement VMs, and VM license offerings from the Azure Marketplace.

We are going to explore the following topics:

- VM types and their uses
- License offerings
- How to deploy a VM
- How to change VM settings
- Common VM use cases

We will also set up a VM during this chapter.

Azure VM types

As soon as you start working with VMs, you will notice that Microsoft offers lots of different VMs and is still expanding its offerings. The reason why Microsoft offers so many VM types is easy to explain. Microsoft needs to support different kinds of workloads. VMs are always the basement for every Microsoft service offered out of Azure or other Microsoft Cloud Services. VMs are also the core component for virtual appliances in Azure such as Hadoop, Microsoft Dynamics NAV, third-party network appliances, or open source appliances.

Microsoft currently offers the following types of VMs:

- **Basic A-series:** For testing and development.
- **Standard A-series:** All-round VMs for various workloads.
- **Compute intense A-series:** Designed for high performance computing.
- **D and DS-series:** For enterprise applications and applications with a higher demand for 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 applications.
- **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 VMs include an NVIDIA Tesla M60 and K80. They are built for VDI or GPU computing. To get more information about Azure N-series, please follow the link <https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sizes-gpu>.
- **LS-series:** Build for low latency storage demands above the offerings of GS-series.
- **B-series:** Burstable VMs for workloads that do not need a continuous performance for CPU and memory and burst in their performance.
- **E and ES-series:** Built for memory-intensive workloads.
- **M-series:** These are the largest VMs available in Azure and built for SAP workloads.
- **SAP large instances:** Special systems for running even larger SAP workloads with 20 TB of memory in a single VM.

As you can see, there are some VM types that have an *S* in the name. Those types represent the storage. **S-series** VMs use **solid-state disk (SSD)** 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 VM types are only made different by certain quality of service standards and priorities against other VM types, but there are also connections to their virtualization host. Microsoft mostly uses standard Intel Xeon processors.



When looking into this, you will find an expression for the CPUs used in the Microsoft documentation at <https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-windows-sizes/>.

To get an idea of which type of CPU is used by Microsoft to host your VM, you should look up the virtual hardware, for example, the CPU that is used for your VM. Microsoft updates the VM series along with their hardware and CPU updates. To track these changes, it is recommended to refer to the Microsoft documentation.

A-series VMs

A-series VMs are divided into two categories. The first one is **basic** and the second is **standard**, which is also the class of all other VM series.

A-series VMs are very common and Microsoft deploys them over a variety of hardware types and CPUs. The performance of an A-series VM 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 the following:

- **Availability:** Basic tier VMs are only available in small **A0-A4** instances for testing purposes; standard tier VMs are available on all size instances and regions.
- **Disk IOPS:** Data disk IOPS for basic tier VMs are 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 them to the availability set for high availability, and implement your own load-balancing 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 production.

The standard tier is good for common use such as **Active Directory Domain Services (AD DS)**, **Active Directory Federation Services (AD FS)**, or other basic network and application services.



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 3.20 GHz.



All A-series from **A8** to **A11** are **remote direct memory access (RDMA)** capable.

D-series and DS-series VMs

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 better processor performance with higher limits than the A-series, a higher memory-to-core ratio, and a SSD for the temporary disk.

They are built for operating systems such as Windows Server 2012 R2 or later. These VMs can easily handle workloads from file servers, databases, applications, and web servers.



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 (TBT)** 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 VMs

The 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.

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 VM.

G-series and GS-series VMs

G-series sizes are built to provide the most memory, the highest processing power, and the largest amount of local SSD of any VM size offered by Azure. This extraordinary performance will allow you to deploy 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. Those VMs are partly running on dedicated physical hardware.

H-series VMs

Azure H-series VMs 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 VMs with size H16R and H16RM are RDMA capable.

NV-series and NC-series VMs

The NC and NV sizes are GPU-enabled instances. These are specialized VMs.

NV VMs

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.

NC VMs

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.

Ls-series VMs

The Ls-series is built for workloads that require low latency local storage with high demand on IOPS, such as NoSQL databases or virtualized Windows Server **Storage Spaces Direct (S2D)**. 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 VMs with a size of **Standard_L32s** run on isolated and dedicated hardware.

B-series VMs

The B-series was built as a cost effective way to deploy workloads that do not need the full performance of the CPU continuously and burst in their performance. A B-series VM is running in the low-point and not fully utilizing the baseline performance of the CPU. While not using the full performance of the CPU, the VM instance builds up credits. When the VM has accumulated enough credits, the credits can be used and the VM can burst its usage, up to 100% of the vCPU, for the period of time when the application requires higher CPU performance. A B-series VM is based on Intel® Broadwell E5-2673 v4 2.3 GHz or an Intel® Haswell 2.4 GHz E5-2673 v3 processor vCPU.

E-series VMs

Ev3-series VMs are built on the 2.3 GHz Intel Xeon® E5-2673 v4 (Broadwell) processor and can achieve 3.5 GHz with Intel TBT 2.0. Ev3-series instances are used to run memory-intensive enterprise applications and workloads.

M-series VMs

M-series are the largest VMs provided by Microsoft Azure. With these VMs, you are able to run a single VM with 4 TB of memory and 128 hyper-threaded vCPUs. The vCPU is based on Intel® Xeon® 2.5 GHz E7-8890 v3 processors and meant to run workloads such as SAP HANA and SQL Hekaton.



In May 2018, M-series VMs became officially SAP HANA certified: <https://azure.microsoft.com/en-us/blog/azure-m-series-vm-are-now-sap-hana-certified/>.

SAP HANA Large Instances

Like the M-series, SAP HANA Large Instances are running based on Intel® Xeon® 2.5 GHz E7-8890 v3 processors. The difference with the M-Series is they cannot be booked through the Microsoft Azure Portal; to order those VMs, a customer must get in contact with their Microsoft account team and sign a separate contract for those VMs before they become available. As a benefit, the customer gets VMs that are running with up to 32 TB of memory in multi-node configuration. A single VM offers 20 TB of memory.

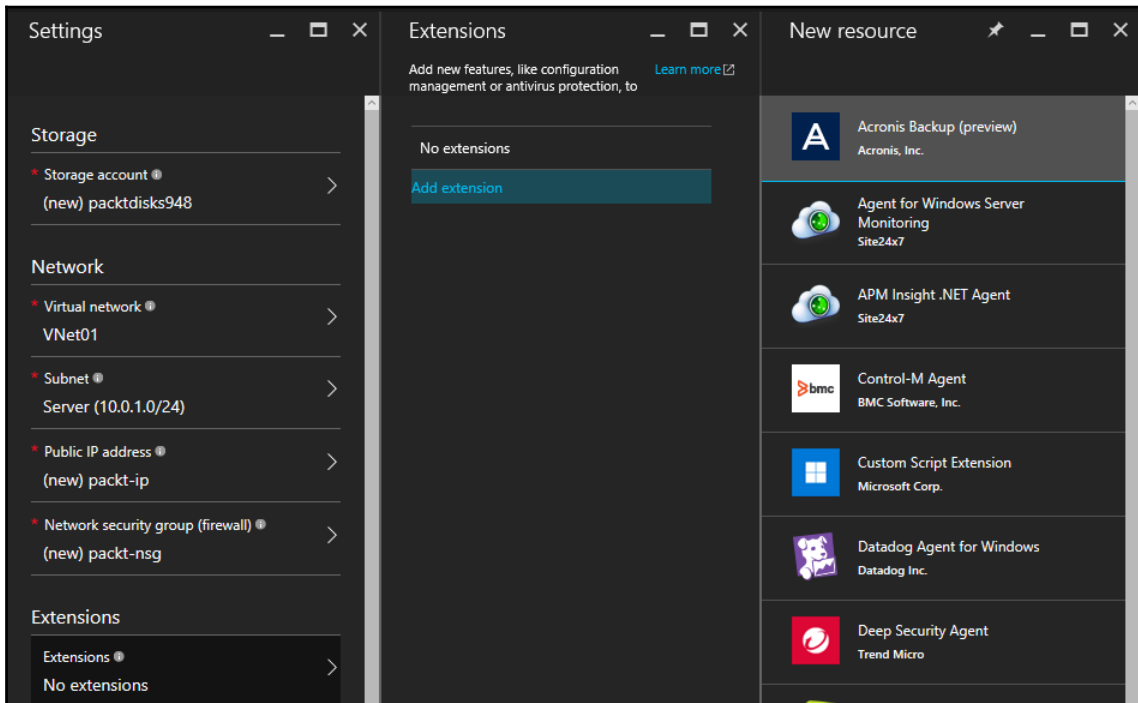


To find more information about SAP HANA on Azure, please visit the Microsoft documentation: <https://docs.microsoft.com/en-us/azure/virtual-machines/workloads/sap/get-started>.

VM extensions

Together with partners and out of its own portfolio, Microsoft offers a wide range of extensions for VMs. Those extensions range from simple anti-malware solutions and backup, to deployment extensions for desired state configuration, to open source plugins such as those for the **Chef server** or **Puppet**.

The following screenshot shows some of the extensions:



The number of extensions for VMs is rapidly growing. To get a full and accurate list, you need to run the following PowerShell command in Microsoft Azure:

```
Get-AzureVMAvailableExtension | Select ExtensionName, Version
```

The VM 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 in your central management, for example, when it comes to anti-malware.

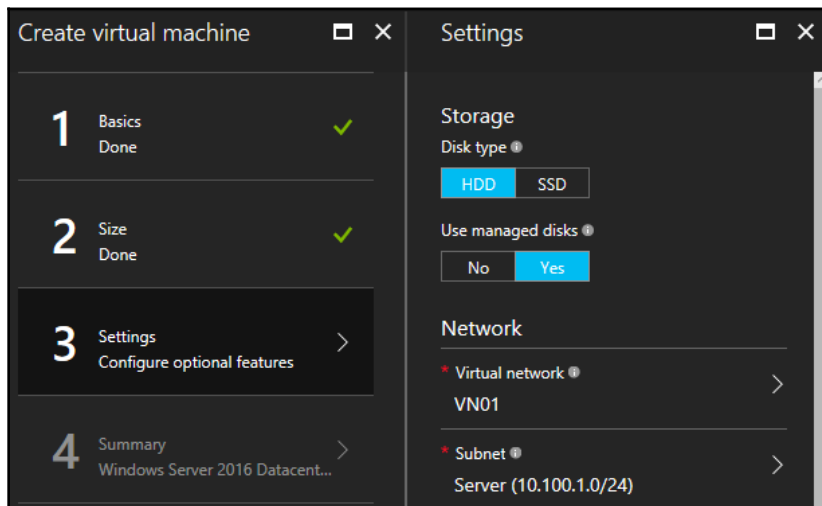


If you are using, for example, anti-malware software that is also offered in Azure, it could be more cost-effective to use the VM extension license instead of buying a new one.

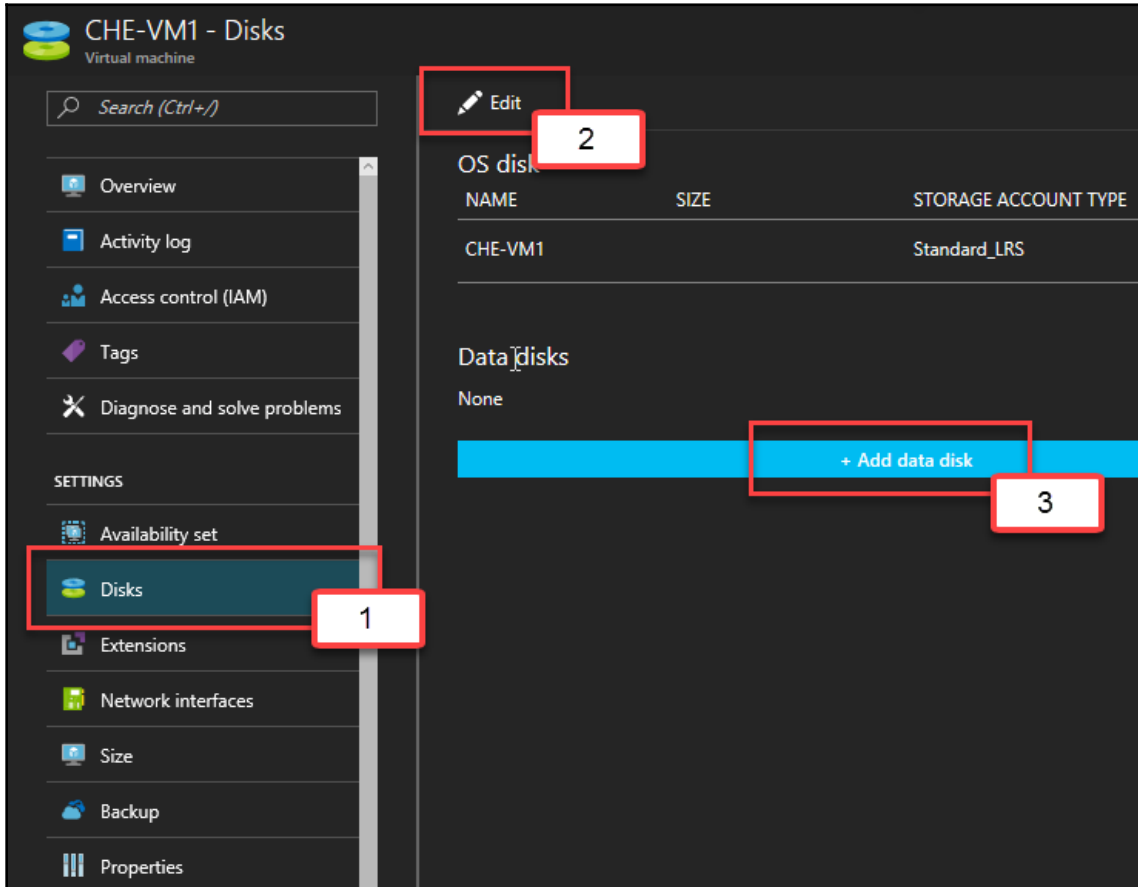
Managed disks

With its progress, Microsoft is changing and evolving the Azure environment, and recently added a new option for VMs 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. To get more info about managed disks, like pricing or currently available disk type, visit following part of the [documentation](https://docs.microsoft.com/en-us/azure/virtual-machines/windows/managed-disks-overview). <https://docs.microsoft.com/en-us/azure/virtual-machines/windows/managed-disks-overview>. To create a VM with managed disks, you can do so during the deployment process, which we will go through later in this chapter:



If you want to manage the disks of a VM, you now can find a configuration blade in the VM settings, where you can edit the managed disks of the VM or click on **+Add data disk**, as shown in the following screenshot:

**TIP**

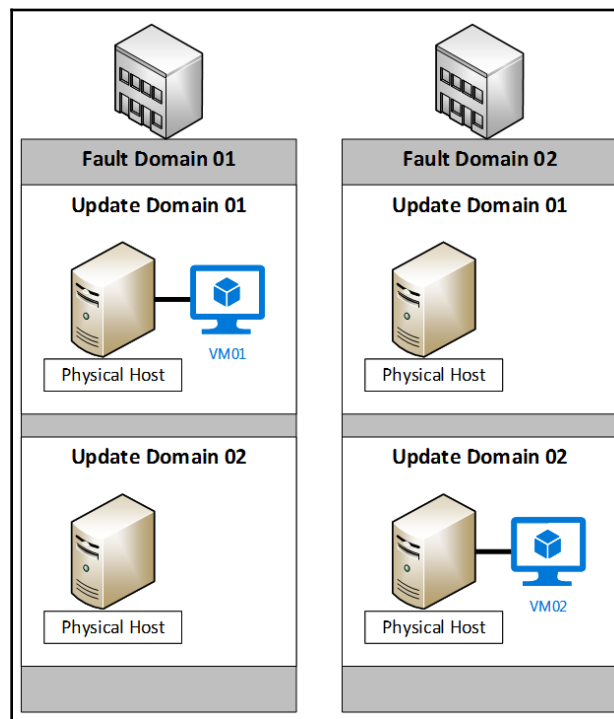
If you are currently working with VM disks on storage accounts and want to migrate to managed disks, Microsoft offers a detailed guide on the Azure documentation website at <https://docs.microsoft.com/en-us/azure/virtual-machines/windows/migrate-to-managed-disks>.

Availability sets

Availability sets are a basic way to let Microsoft Azure know that these two VMs belong to a cluster or application group and are now allowed to go down together. For VMs within a availability Azure manages that these machines run within different fault and update domains of an Azure region:

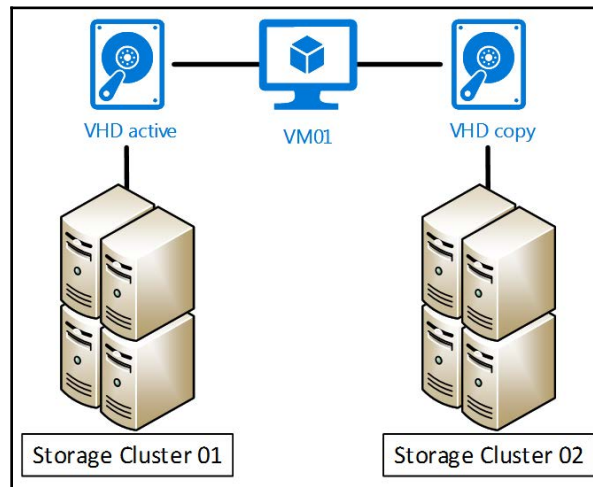
- **Update domain:** Update domain in Azure means that all physical servers in an update domain will get host updates such as firmware, drivers, and OS updates at the same time.
- **Fault domain:** Fault domain in Azure means that all servers in these domains run in the same fire sections, with the same air conditions or electrical source. This means all physical servers within those domains can have an outage at the same time.

The following diagram is a schematic view of the distribution of a VM within an availability set:



With Azure VM managed disks, there is an additional option that comes with availability sets. Azure also takes care that managed disks for VMs within an availability set are placed in different storage clusters of the Azure region.

The following diagram illustrates how those disks are distributed among the storage clusters:



Availability Zone (AZ)

AZs are implemented for high availability to protect applications and data from datacenter failures. AZs are unique physical locations within an Azure region. Currently, you can choose between three zones. Each zone is made up of one or more datacenters, depending on the size of the region. Every zone's data center is equipped with independent power, cooling, and networking.

If one datacenter fails, the other data center, and with that the workloads within that datacenter, are not effected.

This list shows the currently available workloads and applications that support and offer AZs:

- Linux VMs
- Windows VMs
- Virtual machine scale sets
- Managed disks
- Load balancers
- Public IP addresses
- Zone-redundant storage
- SQL databases
- Event hubs
- Service buses
- VPN gateways
- ExpressRoute

Azure Hybrid Use Benefit (HUB)

For Microsoft customers with software assurance, Azure HUB allows you to use on-premises Windows Server and SQL licenses and run workloads such as a VM or Microsoft SQL Server on Azure at a reduced cost. Azure HUB can be used for Windows Server or Microsoft SQL to deploy new VMs with Windows OS or SQL Server.

Looking at Windows Server, you get the following offering when using HUB.

Each two-processor license or each set of 16-core licenses can be used with two instances of up to eight cores, or one instance of up to 16 cores. The Azure Hybrid Benefit for Standard Edition licenses can either be used on-premises or in Azure. The Datacenter Edition benefits allow for simultaneous use on-premises and in Azure.



Currently, there is no automated tracking of HUB licenses in Azure or in any offering, such as **Key Management Service (KMS)** or **Active Directory (AD)** based license activation. So, you need to track their usage by yourself to be prepared for a license audit.

Azure Reserved Instances (RIs)

Azure reservations are built to save customers money for workloads running 24/7 in Azure, and should support Microsoft with capacity planning for their Azure regions. By pre-paying for one year or three years of VMs, SQL database compute capacity, or other Azure resources, the customer gives a commitment to Microsoft for the use of those workloads. As a return service, the customer gets up to a 72% discount on the pay-as-you-go rate. The discount may differ when using other license models, such as an **Enterprise Agreement (EA)** or Cloud Provider License Agreement. Reservations provide a billing discount and do not affect the runtime state of resources. To get more deep dive on hybrid use benefits, take a look on the Microsoft Licensing site for that topic at <https://azure.microsoft.com/en-us/pricing/hybrid-benefit/>.

RIs can also be canceled afterwards, but when doing this, Microsoft currently charges customers a penalty of around 7% of the pre-payment.



With the combination of RIs and Azure HUBs, a customer can save up to 80% on a VM. The saving may be different depending on the size and type of the VM. Normally, the larger the VM, the higher the saving.

Accelerated networking for VMs

Accelerated networking provides low network latency through Azure's in-house programmable hardware, which is based on **field-programmable gate array (FPGA)** and technologies such as **single root I/O virtualization (SR-IOV)**. During Azure's software-defined networking stack movement away from CPUs and into FPGA-based SmartNICs, compute cycles are reclaimed and available again for end user applications, putting less load on the VM, decreasing jitter and inconsistency in latency.

Accelerated networking also enables customers to use more bandwidth, with up to 30 Gbps per VM.



If you are interested in SmartNICs or the next generation, SDN/SDWAN, I would highly recommend Microsoft's research around that topic at <https://www.microsoft.com/en-us/research/publication/azure-accelerated-networking-smartnics-public-cloud/> as well as the session, *Inside Microsoft's FPGA-Based Configurable Cloud*, from Mark Russinovich at https://www.youtube.com/watch?v=v_4Ap1bjwgs.

VM serial console

Newly announced at Ignite 2018, the VM serial console, basically a virtual RS232 interface, should allow customers to access a text-based console for Linux VMs. This serial connection is to the **COM1** serial port of the VM, providing access that is independent of a network or operating system. Access to the serial console can only be done through the Azure portal at the moment and is allowed only for users who have VM Contributor or above access rights to the VM.

Azure Confidential Compute

Azure Confidential Computing protects data while it's processed through the use of secure enclaves. This security capability provide a full data protection at rest, in transit, and in use, which means also during processing in Memory and CPU.

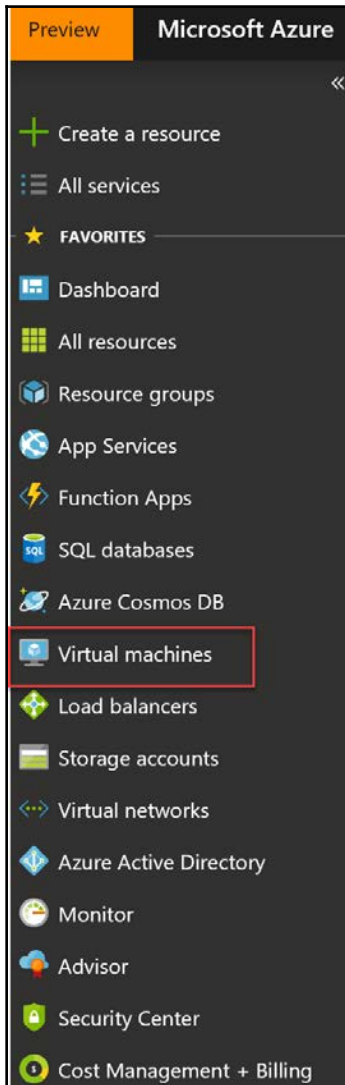


During writing the book, Confidential Compute is in Preview and will be GA around the 1st of November 2018.

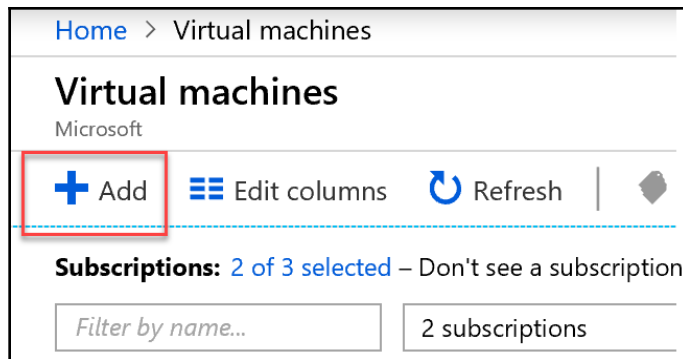
Deploying a VM in Azure

In this part of the chapter, we will deploy a VM by using the **Azure Resource Manager (ARM)** portal. During the following guide, we will configure storage, network, and backup for the VM. Normally, you would prefer to create this before deploying a VM:

1. First, you open the side bar and select **Virtual machines**, as shown in the following screenshot:



2. In the new window, click on the **+Add** button:

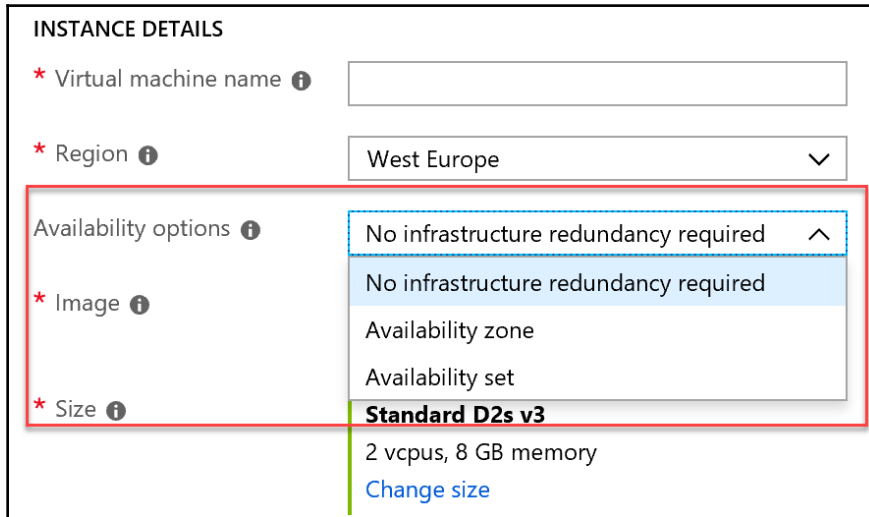


3. Afterwards, you need to select the subscription and resource group or create a new one. Then, you scroll down, select an VM name, and specify the Azure region to deploy to, as shown in the following screenshot:

The screenshot shows the 'INSTANCE DETAILS' form in the Azure portal. The form has several fields with red asterisks indicating required fields. The first two fields, 'Virtual machine name' and 'Region', are highlighted with a red box. The 'Virtual machine name' field is empty. The 'Region' field is set to 'West Europe'. Below these, the 'Availability options' field is set to 'No infrastructure redundancy required'. The 'Image' field is set to 'Ubuntu Server 18.04 LTS'. The 'Size' field is set to 'Standard D2s v3'. There are links for 'Browse all images and disks' and 'Change size'.

INSTANCE DETAILS	
* Virtual machine name ⓘ	
* Region ⓘ	West Europe ▼
Availability options ⓘ	No infrastructure redundancy required ▼
* Image ⓘ	Ubuntu Server 18.04 LTS ▼ Browse all images and disks
* Size ⓘ	Standard D2s v3 2 vcpus, 8 GB memory Change size

4. In the next step, you select the availability option. You can choose between availability set, zone, or nothing. We will choose **No Infrastructure redundancy required** in this step, shown as follows:



INSTANCE DETAILS

* Virtual machine name ⓘ

* Region ⓘ West Europe ▼

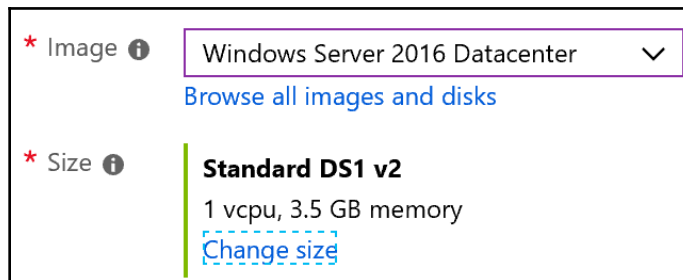
Availability options ⓘ

- No infrastructure redundancy required ^
- No infrastructure redundancy required
- Availability zone
- Availability set

* Image ⓘ

* Size ⓘ **Standard D2s v3**
2 vcpus, 8 GB memory
[Change size](#)

5. Select the image for the OS and the size of the VM:



* Image ⓘ Windows Server 2016 Datacenter ▼
[Browse all images and disks](#)

* Size ⓘ **Standard DS1 v2**
1 vcpu, 3.5 GB memory
[Change size](#)

6. By changing the filters, you can see the other VM sizes too:

Select a VM size
Browse available virtual machine sizes and their features

Search by VM size... [Clear all filters](#)

Size: **Small** Generation: **Current** Family: **General purpose** Premium disk: **Supported**

Showing 12 of 210 VM sizes. | Subscription: Visual Studio Enterprise | Region: West Europe

VM SIZE	OFFERING	FAMILY	VCPUS	RAM (GB)	DATA DISKS	MAX IOPS	TEMPORARY STORA...	PREMIUM DISK SUP...	COST/MONTH (ESTI...
B1ms	Standard	General purpose	1	2	2	1600	4 GB	Yes	€15.25
B1s	Standard	General purpose	1	1	2	800	4 GB	Yes	€7.65
B2ms	Standard	General purpose	2	8	4	4800	16 GB	Yes	€60.98
B2s	Standard	General purpose	2	4	4	3200	8 GB	Yes	€30.49
B4ms	Standard	General purpose	4	16	8	7200	32 GB	Yes	€121.97
D2s_v3	Standard	General purpose	2	8	4	3200	16 GB	Yes	€75.29
D4s_v3	Standard	General purpose	4	16	8	6400	32 GB	Yes	€150.58
DS1_v2	Standard	General purpose	1	3,5	4	3200	7 GB	Yes	€42.66
DS2_v2	Standard	General purpose	2	7	8	6400	14 GB	Yes	€85.33
DS2_v2	Promo	General purpose	2	7	8	6400	14 GB	Yes	€85.33
DS3_v2	Standard	General purpose	4	14	16	12800	28 GB	Yes	€170.66
DS3_v2	Promo	General purpose	4	14	16	12800	28 GB	Yes	€170.66

[Select](#) 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. [View Azure pricing calculator.](#)

7. After configuring the virtual hardware, you need to create your administrative account and password:

ADMINISTRATOR ACCOUNT

* Username ⓘ

* Password ⓘ

* Confirm password ⓘ

8. In the last step of the basic configuration, you enable Azure HUB by changing the default radio button **No** to **Yes** you enable Azure HUB for the VM:

SAVE MONEY

Save up to 49% with a license you already own using Azure Hybrid Benefit for Windows Server. [Learn more](#)

* Already have a Windows license? ☐ Yes ☒ No

9. We are finished with the basic configuration and can click on the **Review + create** button. Azure then works with default configurations. I wouldn't suggest that. So, click on the **Next : Disks >** button, as shown in the following screenshot:

Review + create

Previous

Next : Disks >

10. In the next step, you decide on the disks. With the radio button, you can decide between **managed** or **unmanaged** disks. I personally would recommend the managed disks. Please be aware you can only add as many disks as your VM size is allowed to have. If you need more disks, choose another VM type. You can also attach existing disk, from your Azure disk library, for example from other systems or import services:

Create a virtual machine

[Basics](#)
[Disks](#)
[Networking](#)
[Management](#)
[Guest config](#)
[Tags](#)
[Review + create](#)

Azure VMs have one operating system disk and a temporary disk for short-term storage. You can attach additional data disks. The size of the VM determines the type of storage you can use and the number of data disks allowed. [Learn more](#)

DISK OPTIONS

* OS disk type

Use unmanaged disks ☐ Yes ☒ No

DATA DISKS

You can add and configure additional data disks for your virtual machine or attach existing disks. This VM also comes with a temporary disk.

LUN	NAME	SIZE (GiB)	DISK TYPE	HOST CACHING
Create and attach a new disk	Attach an existing disk			

11. After you attach your disks, click on the **Next : Networking** > button:

[Review + create](#)
[Previous](#)
[Next : Networking >](#)

12. We need to configure our network; if you don't have anything preconfigured, you need to start from scratch, but you have already learned how to do the detailed configuration in the [Chapter 5, Implementing Azure Networks](#) of this book, so we will leave that out. What is important in that configuration is the point about public **inbound ports**. If you have no VPN to the server or VNet and you need to manage the VM OS, you need to explicitly enable RDP or SSH at this point. With a supported VM size, you can also activate accelerated networking. I would suggest to do so for every VM size that supports that feature. It's free and really improves you network performance, and most every modern Linux distribution and Windows Server 2012 R2 or later support that feature:

[Basics](#) [Disks](#) [Networking](#) [Management](#) [Guest config](#) [Tags](#) [Review + create](#)

Configure a new or existing virtual network for your VM as well as how your VM will be accessed on the virtual network. [Learn more](#)

NETWORK INTERFACE

When creating a virtual machine, a network interface will be created for you.

* Virtual network ⓘ

(new) Packt-vnet ▼
[Create new](#)

* Subnet ⓘ

default ▼

Public IP ⓘ

(new) packt-test-ip ▼
[Create new](#)

Network security group

☒ Basic ☐ Advanced

* Public inbound ports ⓘ

☒ None ☐ Allow selected ports

Select inbound ports

Select one or more ports ▼

ⓘ

All traffic from the internet will be blocked by default. You will be able to change inbound port rules in the VM > Networking page.

Accelerated networking ⓘ

☐ On ☒ Off

The selected VM size does not support accelerated networking.

13. After you configure the network, click on the **Next : Management >** button :

Review + create	Previous	Next : Management >
------------------------	-----------------	-------------------------------

14. In the management part of the creation, you can configure basic monitoring, such as boot diagnostics and OS diagnostics, or configure backup, auto-shutdowns, or managed service identity. I would recommend to configure at least boot and OS diagnostics. This information would help you in any support case. To store this information, you need to create a storage account in your tenant; I would recommend to create one storage account for all diagnostic information and use it for more than one machine:

Basics	Disks	Networking	Management	Guest config
Configure monitoring and management options for your VM.				
MONITORING				
Boot diagnostics ⓘ		☒ On ☐ Off		
OS guest diagnostics ⓘ		☐ On ☒ Off		
* Diagnostics storage account ⓘ		(new) acktdiag Create new		
IDENTITY				
Managed service identity ⓘ		☐ On ☒ Off		
AUTO-SHUTDOWN				
Enable auto-shutdown ⓘ		☐ On ☒ Off		
BACKUP				
Enable backup ⓘ		☐ On ☒ Off		

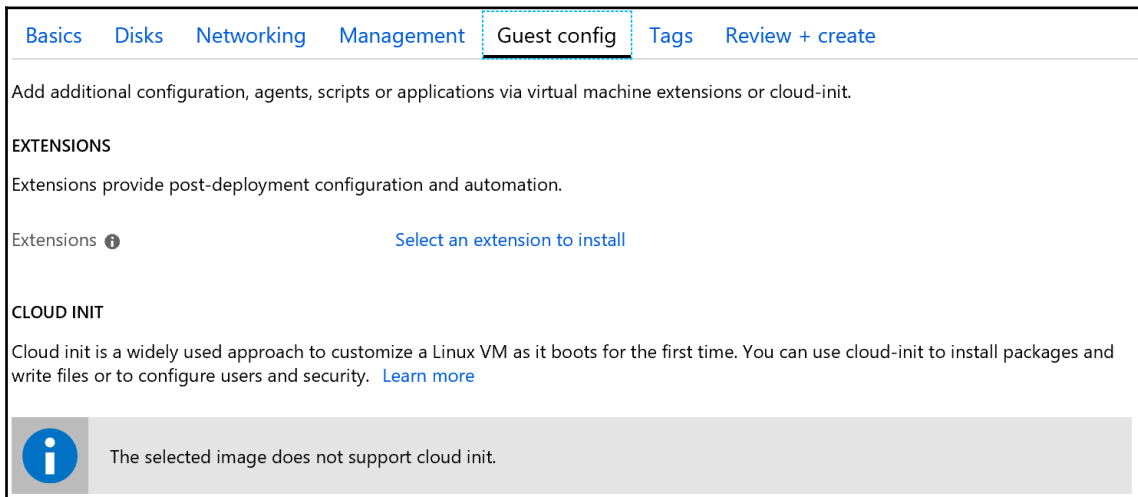


You could now create a backup task for the VM too, but I would recommend to do that later for a bigger group of VMs. That would also help you to define centralized backup policies and makes it easier to get an overview about which machine has a backup and which policies, and which machine has not. Honestly, we are not covering backup and recovery as deeply as it should be in this book, so I would suggest to take an additional look at the Microsoft documentation at <https://docs.microsoft.com/en-us/azure/backup/backup-introduction-to-azure-backup>.

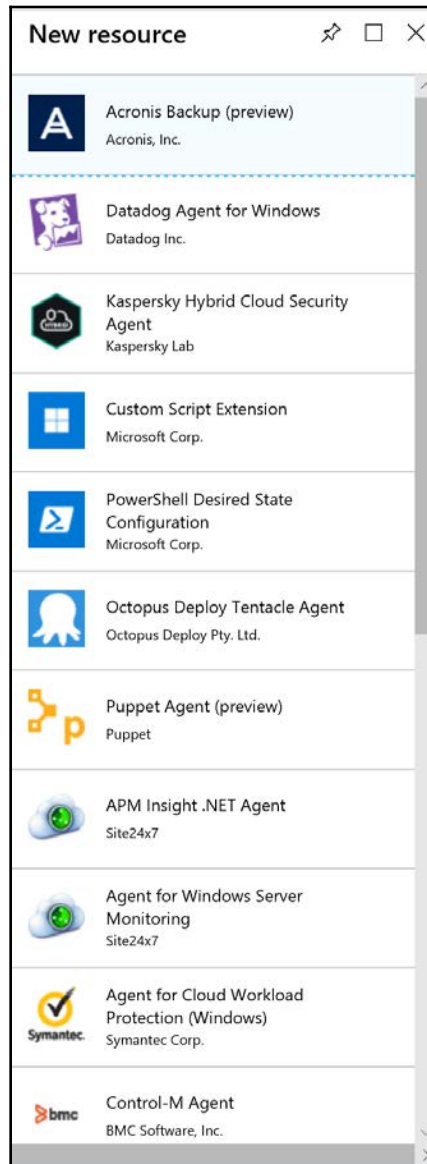
15. When you have finished the management configuration, click on the **Next : Guest config >** button:



16. In the **Guest config** tab, you can add different extensions, for example, for monitoring, anti-virus, or management, or you can do advanced first boot configurations for Linux systems:



17. Looking at the VM extension, you have a wide range of third-party and Microsoft extensions available. Those extensions are also a good option if you do not have client licenses for that software available. Those extension will be added to the VM pricing in nearly all cases as a pay-as-you-go license offering:



18. After you are done with your configuration, click on the **Next : Tags >** button:



19. Tags are very important in Azure and are available not only for VMs but for any resource that can be deployed in Azure. They are equal to labels and help you organize your workloads, or can be used for scripting, automation, and management. You can, for example, set your cost center, departments, or contact information for those resources:

Create a virtual machine

[Basics](#)
[Disks](#)
[Networking](#)
[Management](#)
[Guest config](#)
[Tags](#)
[Review + create](#)

Tags are name/value pairs that enable you to categorize resources and view consolidated billing by applying the same tag to multiple resources and resource groups. [Learn more](#)

Note that if you create tags and then change resource settings on other tabs, your tags will be automatically updated.

☐	KEY	VALUE	RESOURCE TYPE
☐			All resources to be created ▼

20. After you create your necessary tags, you can go to the review screen through the **Next : Review + create >** button:



21. During the review step, ARM will review your configuration and validate it. If you configuration is OK, you can click the **Create** button and your VM will be deployed:



If you have any configuration issues, the deployment engine will notify you about that by showing a warning and a red dot in the configuration step where the error appeared. In our case, I did not set the administrator account and the password during deployment:

Create a virtual machine

!

Validation failed. Required information is missing or not valid.

Basics

Disks

Networking

Management

Guest config

Tags

Review + create

PRODUCT DETAILS

Standard DS1 v2

by Microsoft

[Terms of use](#) | [Privacy policy](#)

Subscription credits apply ⓘ

0.0573 EUR/hr

[Pricing for other VM sizes](#)

TERMS

By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. See the [Azure Marketplace Terms](#) for additional details.

BASICS

Subscription	Visual Studio Enterprise
Resource group	(new) Packt
Virtual machine name	packt-test
Region	West Europe
Availability options	No infrastructure redundancy required
Username	None
Public inbound ports	None

DISKS

OS disk type	Premium SSD
Use unmanaged disks	No

NETWORKING

Create

Previous

Next

[Download a template for automation](#)

Every configuration you make with the wizard can be saved and downloaded as a JSON Template, so you can easily create a library of templates, even if you are not that experienced with JSON-based configuration and deployment:

[Download a template for automation](#)



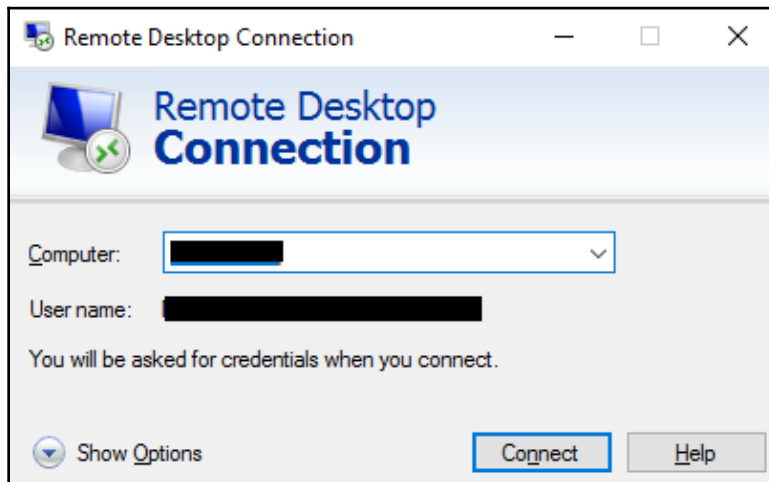
The time needed for deployment depends on your VM size and type. Smaller VMs will take longer to be deployed. Every machine in Azure will be installed and configured from scratch. Azure performs no image-based deployments such as **System Center Virtual Machine Manager (SCVMM)**.

Accessing a VM in Azure

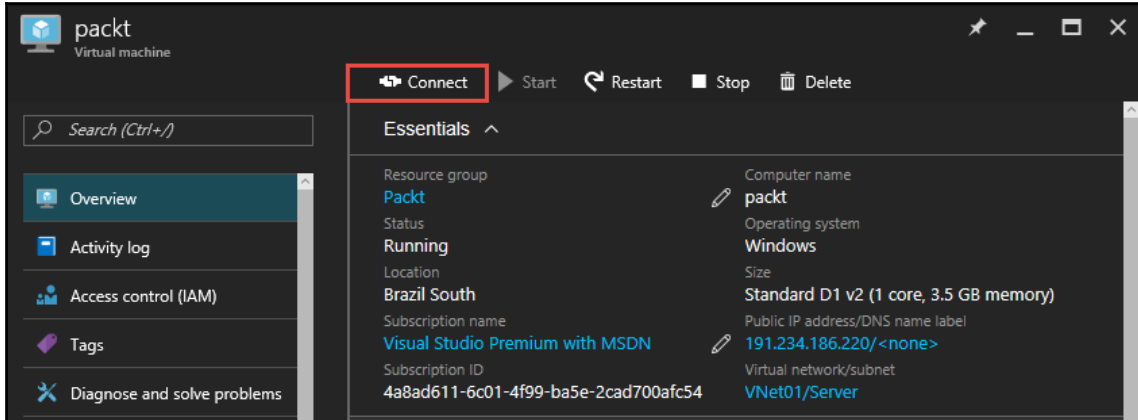
As soon as your VMs are deployed, you can access them.

After you deployed the VM you can connect to VM, like shown in the following guide:

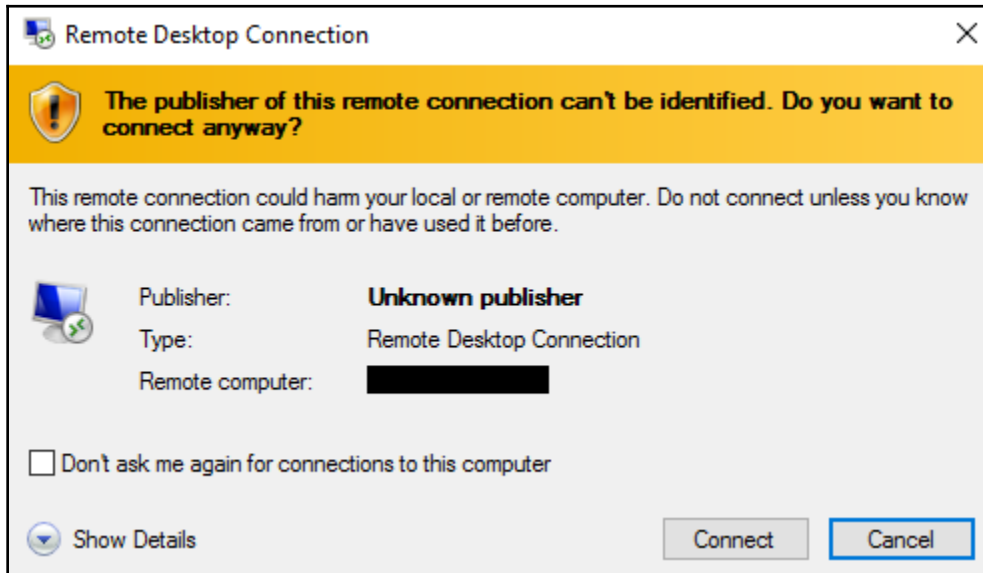
1. You can leverage Microsoft **Remote Desktop Connection** for Windows or SSH tools such as PuTTY for Linux:



2. To connect to your VM, you need to navigate to your VM within your resource groups. In the **Overview** blade, you will see the **Connect** button, as shown in the following screenshot:



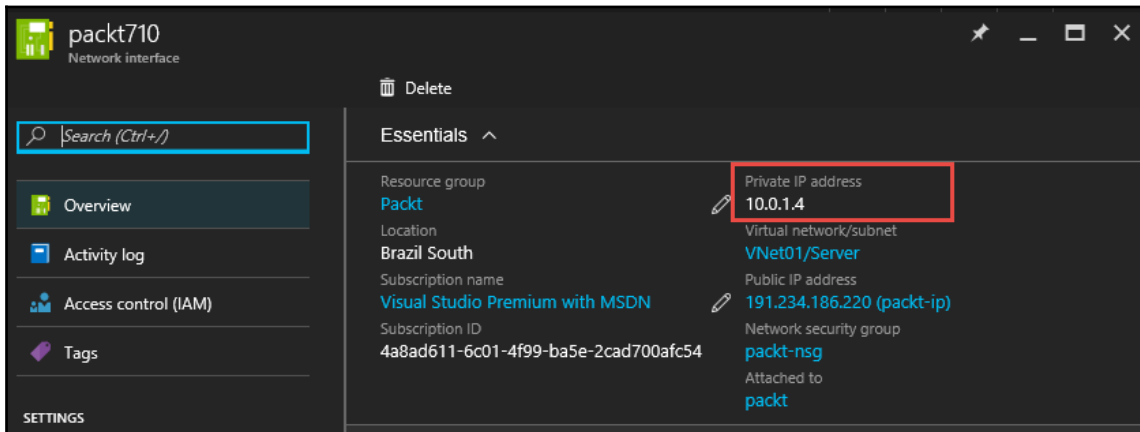
3. For a VM with a public IP, you can download a **Remote Desktop Connection** file and connect directly to the VM:



4. If you have a connection to Azure through VPN or ExpressRoute, you can use the private IP given from Microsoft to your VM. This IP can be found on the virtual network adapter of the VM 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	...

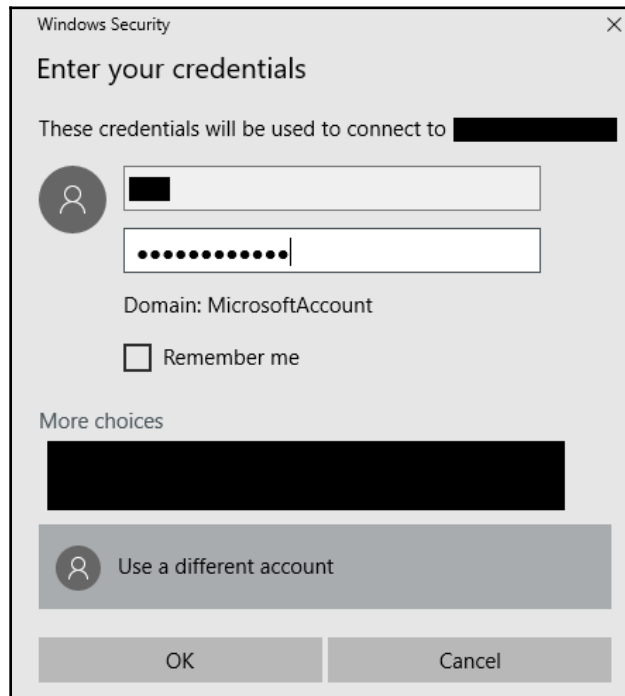
5. There, you also have the **Overview** blade and you can see the IP, as shown in the following screenshot:



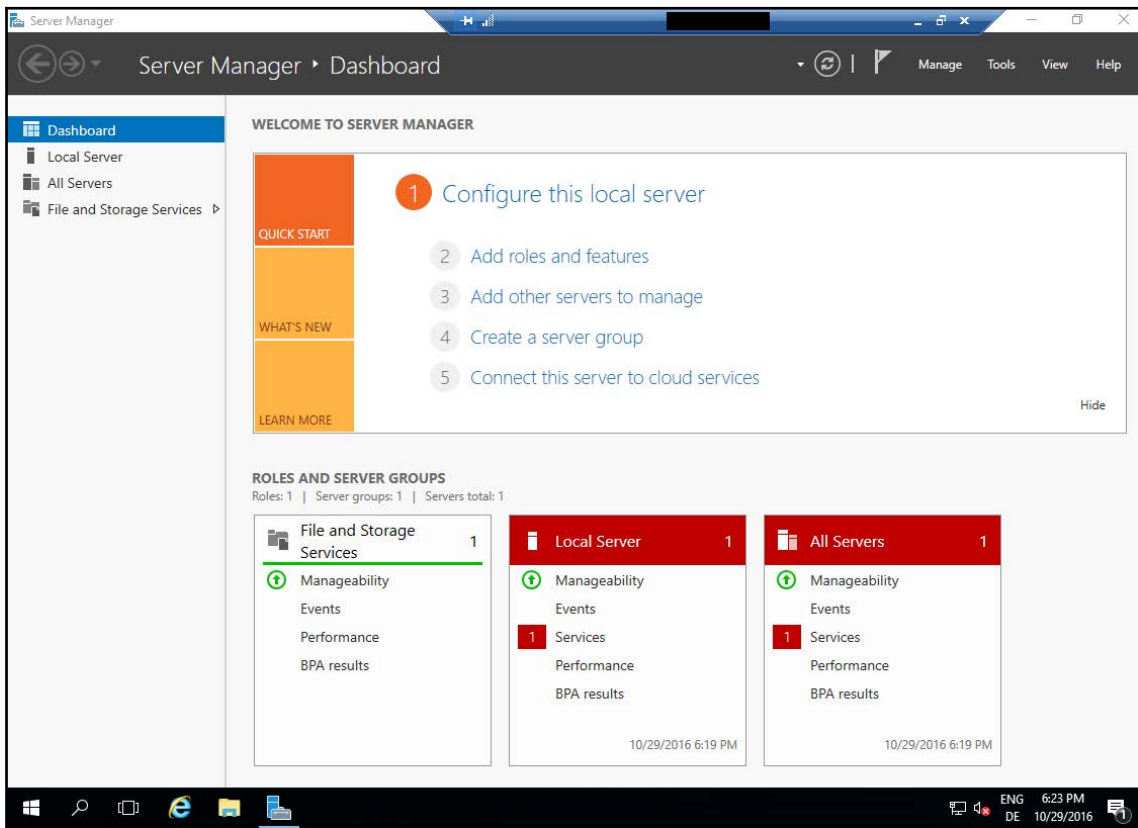
The screenshot shows the Azure portal interface for the network interface 'packt710'. The left sidebar contains a search bar and a list of tabs: Overview (selected), Activity log, Access control (IAM), and Tags. The main content area is titled 'Essentials' and displays various properties of the network interface. The 'Private IP address' is highlighted with a red box and shows the value '10.0.1.4'. Other properties include Resource group (Packt), Location (Brazil South), Subscription name (Visual Studio Premium with MSDN), Subscription ID (4a8ad611-6c01-4f99-ba5e-2cad700afc54), Virtual network/subnet (VNet01/Server), Public IP address (191.234.186.220 (packt-ip)), Network security group (packt-nsg), and Attached to (packt).

Resource group	Packt
Location	Brazil South
Subscription name	Visual Studio Premium with MSDN
Subscription ID	4a8ad611-6c01-4f99-ba5e-2cad700afc54
Private IP address	10.0.1.4
Virtual network/subnet	VNet01/Server
Public IP address	191.234.186.220 (packt-ip)
Network security group	packt-nsg
Attached to	packt

6. After you have started the connection, enter the credentials you used for the local administrator:



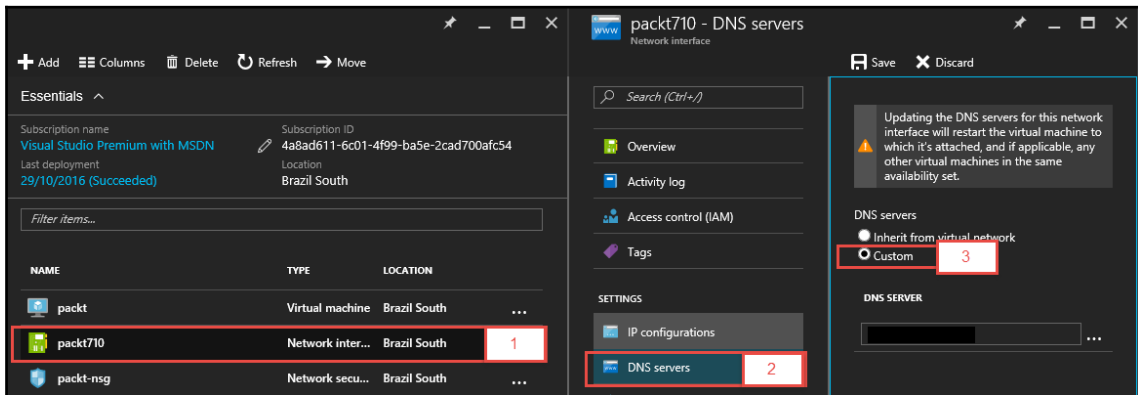
7. Afterwards, you should get the desktop of your VM:



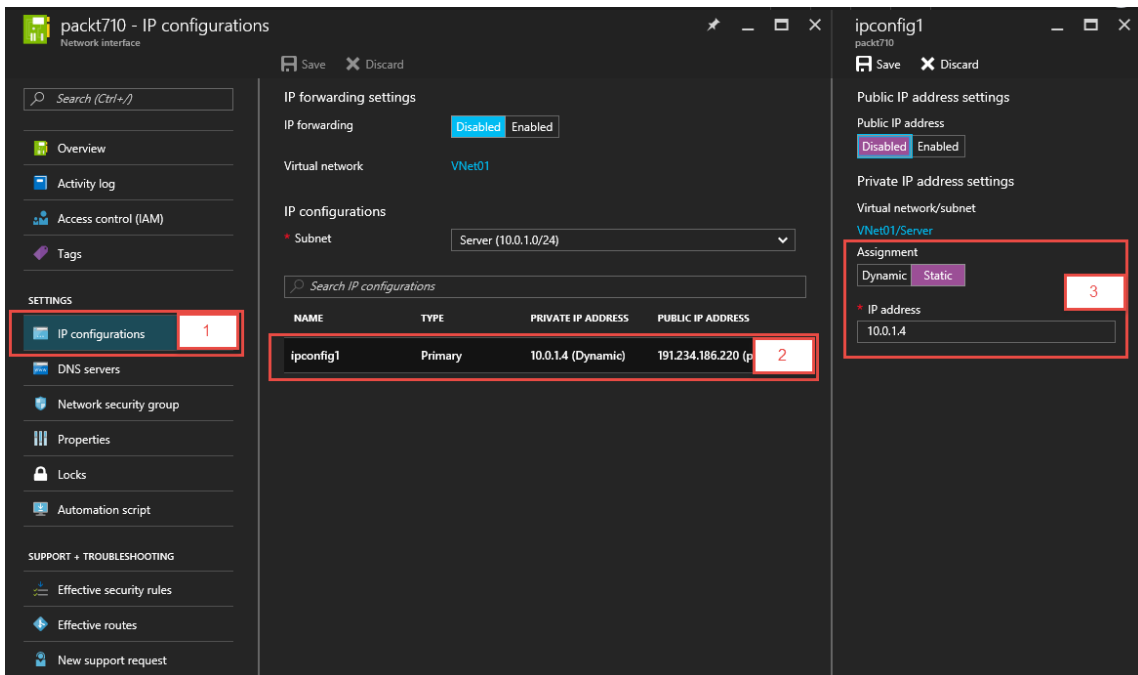
Changing IP and DNS settings

As soon as you are using VMs in an enterprise environment, you may want to change the IPs or DNS servers of a VM, for example, to connect to your domain controller. To change these, you need to navigate back to the network address of the VM.

There, you click on **DNS servers** and afterwards you change the option to **Custom** and enter the IP addresses of your DNS server:



To change the IP, you click on **IP configurations**. Then, click on the **IP configurations** section. After that, you can change the **Assignment** option from **Dynamic** to **Static** and change the IP address to an IP of your choice within the subnet:



Common scenarios for VMs

In the following part of this chapter, you will learn a few common scenarios to start with Azure VMs.

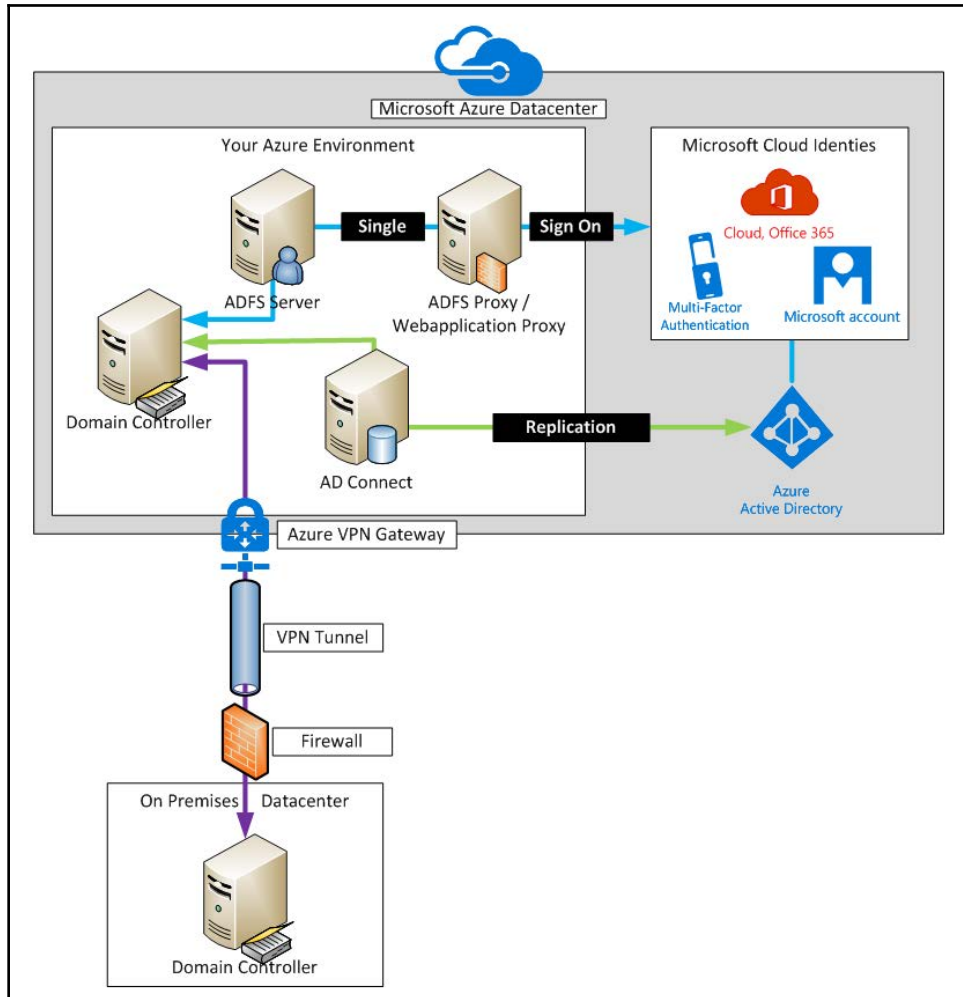
Optimization of Azure-related communication traffic

As you already learned in [Chapter 3, *Deploying and Synchronizing Azure Active Directory*](#), replication traffic for your hybrid identities normally goes over the internet. It's only encrypted using SSL on port 443.

There is an option to optimize security for that traffic by placing the VMs in Azure. They will still communicate with the Azure public IP from Azure AD, but the traffic is handled on the internal switches and router from Microsoft and the traffic doesn't leave the Azure datacenter.

To get the AD account from your on-premises setup, you build up a VPN tunnel or use ExpressRoute to build a secure connection. Afterwards, you place an AD **domain controller (DC)** in Azure and replicate from a bridgehead DC in your on-premises datacenter.

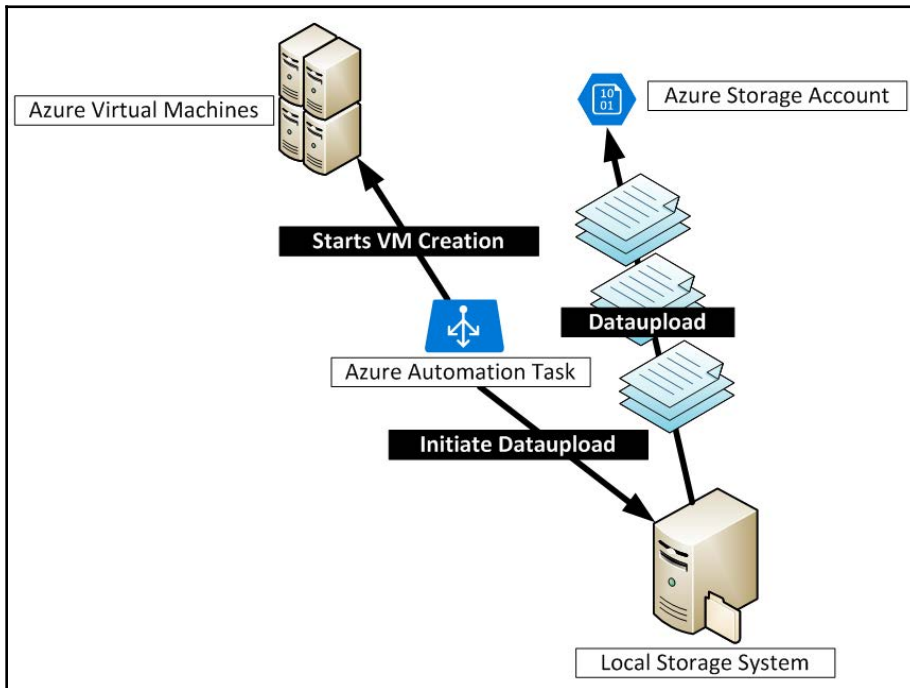
The following diagram shows the concept and VM placement:



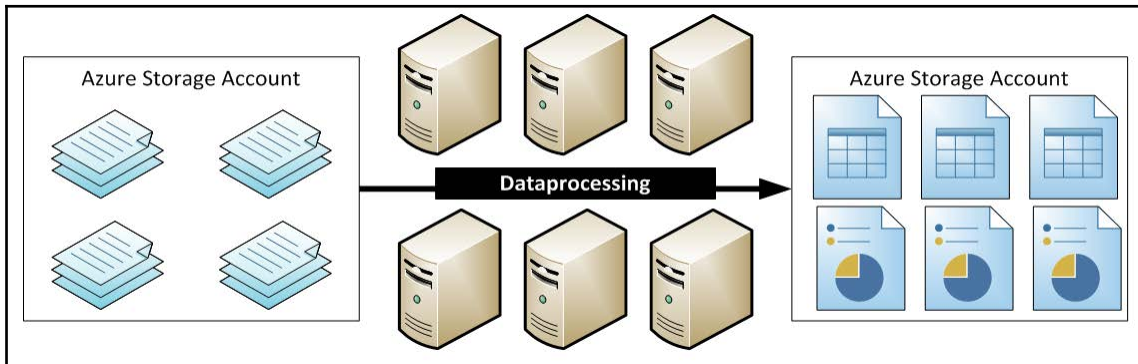
On-demand usage for calculations

Another scenario is to use Azure for workloads that are only temporarily needed; for example, science calculations that need a high amount of calculation capacity and where you have a high investment in the hardware.

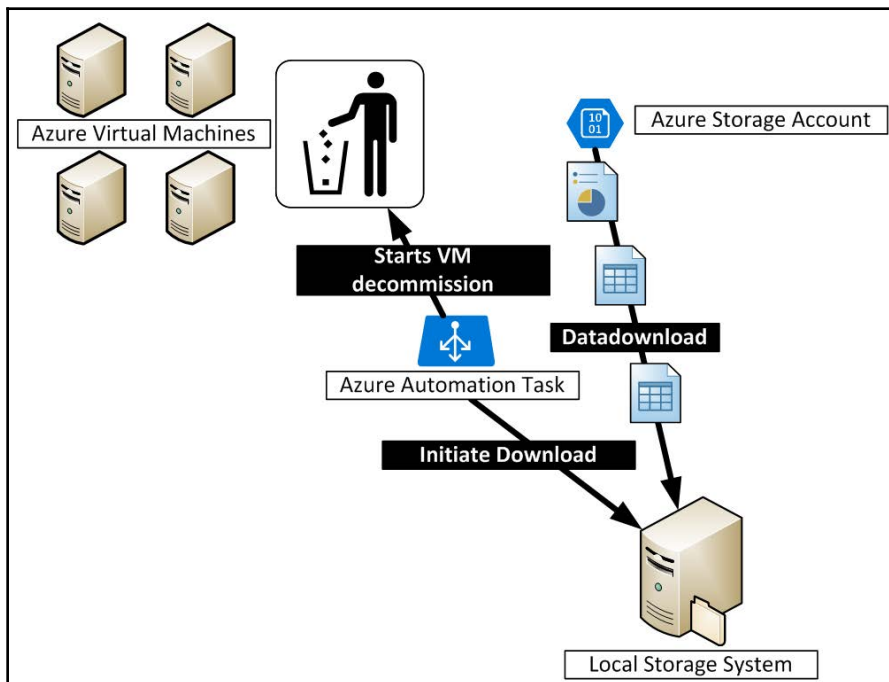
So, *what do you do?* Normally, you create a VM ARM template for those machines. Then, you create a task that triggers your VM deployment and the transfer of the raw data into Azure. The following diagram shows a draft of the workflow:



The deployed Azure VMs run the necessary calculations and deliver the result to 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 VMs. 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 it. The following diagram shows the workflow:

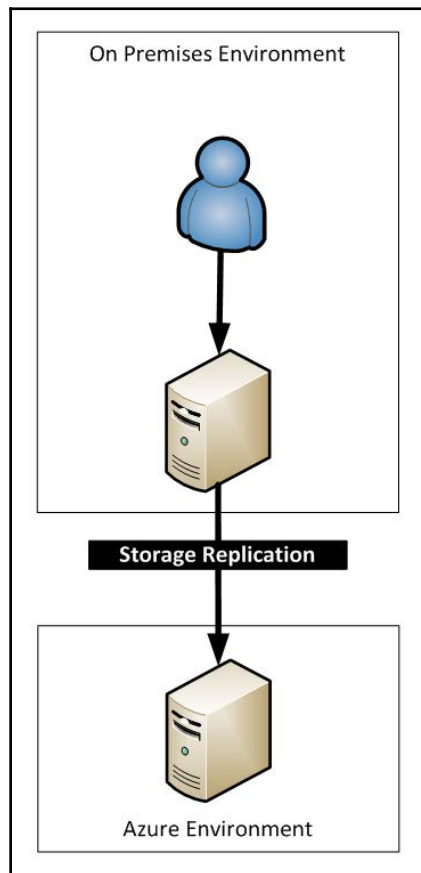


Disaster recovery for on-premises servers

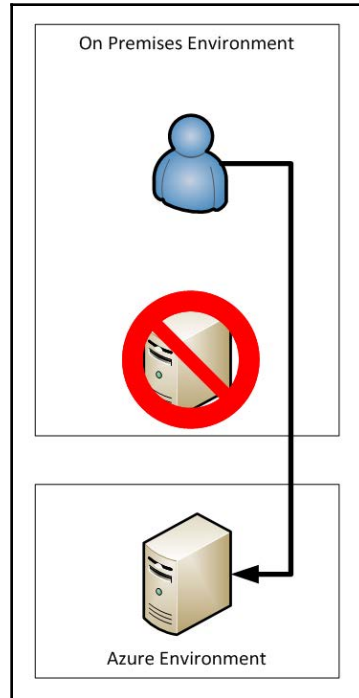
In some cases, it isn't the best choice to place all systems in Azure, for example, when using applications or systems that have a high demand on latency. In those cases, it is necessary to keep the system on-premises, but Azure could still offer support in those situations with disaster recovery or failover options.

For this example, we look at Windows Server 2016 and the new feature storage replication. With storage replication, you can perform asynchronous 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 VMs and decide which VM type and size is the right one for your application. You also know in which scenarios you can benefit from Microsoft VM services and with which scenarios you can start.

In the next chapter, we will take a look at implementing Azure-Managed Kubernetes and Azure Container Service.

Questions

Please answer the following questions:

1. Are B-series VMs always at 100% performance?
2. Are VM extensions always from Microsoft only?
3. Do you need to pre-create a VNet before deploying a VM?
4. Can you set a VM without creating an administrator account and password or SSH key during the deployment?
5. Are their SAP certified VM sizes?
6. Can you set a public IP for a VM or only private IPs?
7. Do you have the option to create a backup during deployment?

Further reading

Within the following books, you can find more information about what we learned in this chapter:

- **Implementing Azure Cloud Design Patterns:** <https://www.packtpub.com/virtualization-and-cloud/implementing-azure-cloud-design-patterns>
- **Implementing Microsoft Azure Infrastructure Solutions: Exam Guide 70-533** <https://www.packtpub.com/virtualization-and-cloud/implementing-microsoft-azure-infrastructure-solutions-exam-guide-70-533>
- **Azure for Architects:** <https://www.packtpub.com/virtualization-and-cloud/azure-architects>
- **Architecting Microsoft Azure Solutions – Exam Guide 70-535:** <https://www.packtpub.com/virtualization-and-cloud/architecting-microsoft-azure-solutions-exam-guide-70-535>

8

Implementing Azure-Managed Kubernetes and Azure Container Service

The next level of virtualization is **containers** as they provide a better solution than virtual machines within Hyper-V, as containers optimize resources by sharing as much as possible of the existing container platform.

This chapter will describe the basics of containers, how to design and implement them, and how to choose the proper solution for orchestrating containers. You will get an overview of how Azure can help you to implement services based on containers and get rid of traditional virtualization stuff with redundant OS resources that need to be managed, updated, backed-up, and optimized.

Containers started with Docker in Linux and came to Windows Server version 2003, without any official feature statement or public announcement because it has been driven by Microsoft Azure, and was a feature that had been used and proofed there for years before Windows Server 2016 Container Services in Windows was officially announced and used.

One of the best things is that a container could be run on Linux and on Windows as a **container host**, without modifications if the service is built up properly, and without breaking the barriers and modifying the general concept of containers.

The following topics will be covered in this chapter:

- Technical design of containers
- Designing microservices
- Container registries
- Container instances
- Container orchestration
- **Azure Container Service (ACS)**

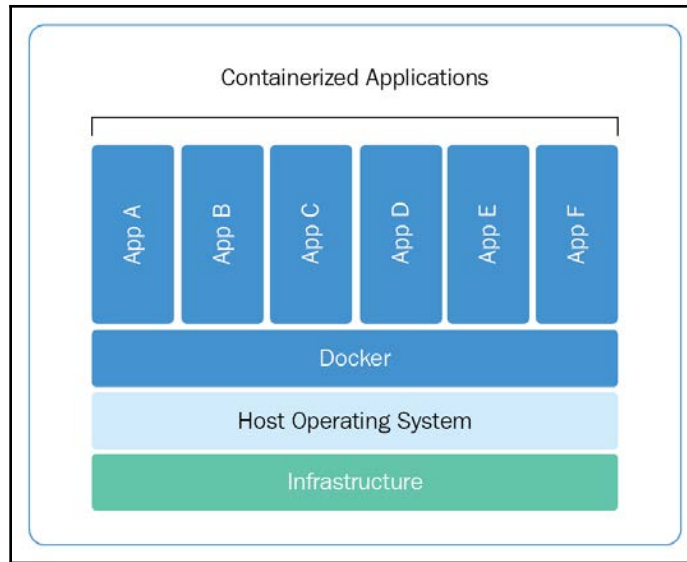
Technical requirements

To run containers in a cloud environment, no specific installations are required, as you only need the following:

- A computer with an internet browser
- An Azure subscription (if not available, a trial could work too: <https://azure.microsoft.com/en-us/free/>)
- The code in this chapter can be found at <https://github.com/PacktPublishing/Implementing-Azure-Solutions-Second-Edition>

Containers – the concept and basics

Containers are a mechanism to run software reliably even when moving them from one computer environment to another. The open sources project Docker in Linux has provided such a service for some years now. It containerizes the application and its dependencies (OS and underlying infrastructure) and abstracts the interaction between each; they isolate applications from each other but use a shared OS. This idea works based on the microservice design of services, because it performs as a service that is independent, flexible, and scalable by default, using predefined APIs for communications:



The basic features of containers are as follows:

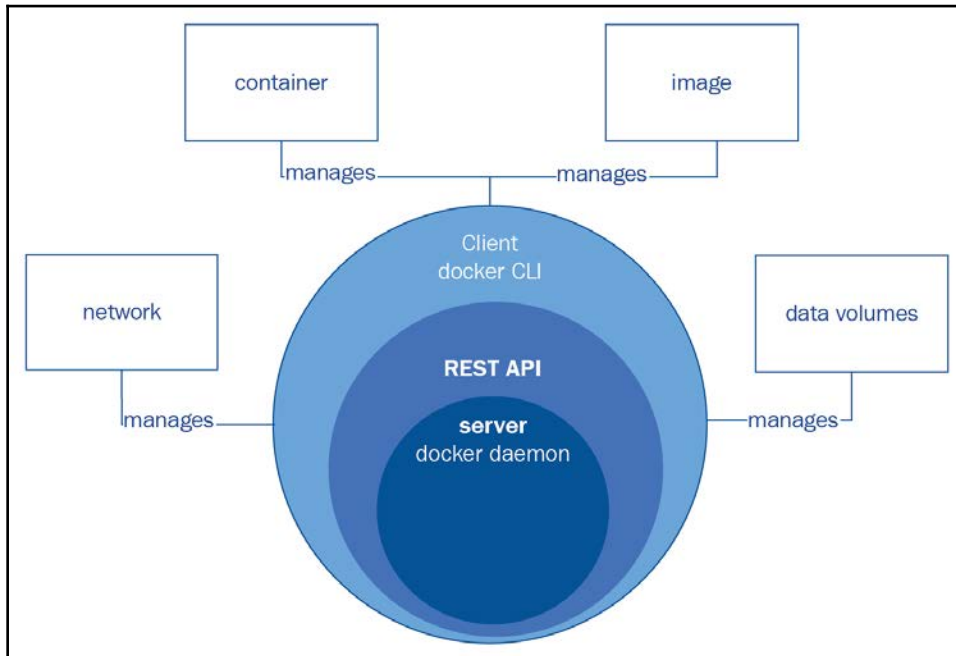
- They are *run-everywhere apps*
- They are developed on a microservice-style architecture
- They enable a higher density of resources

The generic platform for running these containerized applications is Docker, which is available for the following:

- Windows containers
- Docker for Linux
- Docker for Mac
- Boot2Docker
- VirtualBox

In this chapter, we will focus on Linux and Windows containers. With the launch of Windows Server 2016, we had two different version of containers in Windows—Hyper-V and generic containers. As the concept of Hyper-V containers has since been deprecated, we will not waste time on this concept any further.

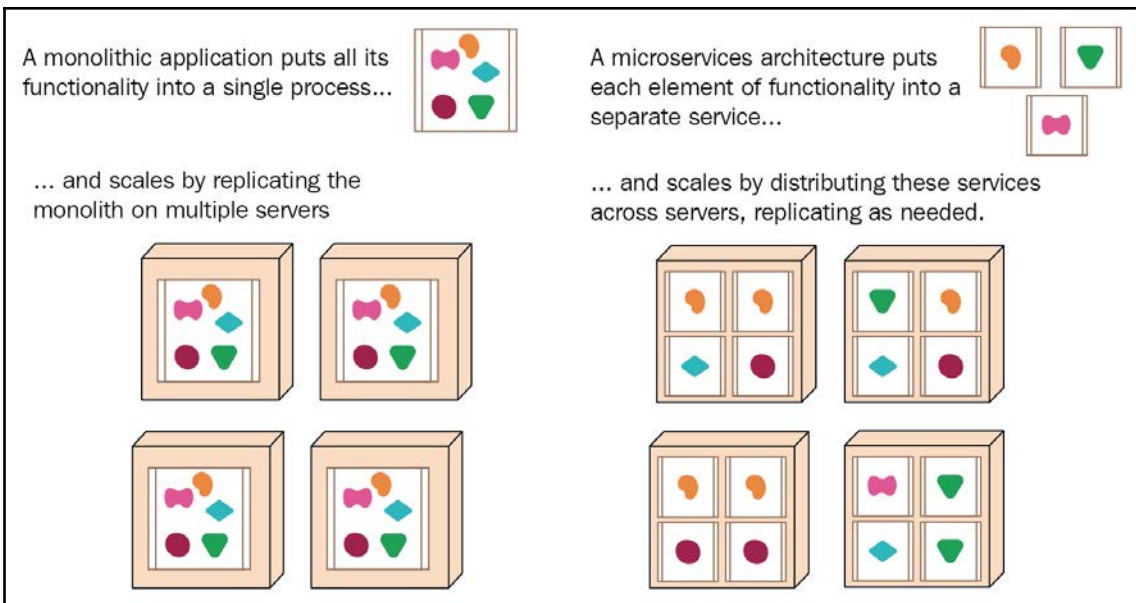
The following image describes the design on container services on a high level:



Microservices – the concept

To design a solution using containers, you will need to design a solution based on the microservices concept. This new architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. This provides an easy mechanism that can even replace technologies, for example, moving from PostgreSQL to MS SQL as you will not have to modify the microservices; just by replacing one microservice, the others will run properly.

The following images describes the concept of container services:



Now, let's combine microservices and containers. The result is a flexible and scalable environment that gives the possibility to spin up additional containers to provide better performance and latency in the app, without any specific configuration changes.

Workloads to run in containers

Docker containers are not the one and only solution for every single workload. There are services that do not fit into this concept, too. The major services that do not fit are as follows:

- Containers are not as fast as *bare-metal* installations. Even though a Docker container has less overhead than VMs, it does not have zero overhead. If your solution needs bare-metal speed, especially from the app point of view, you will have no choice but to run it directly from a bare-metal server, without using containers or virtual machines.
- Containers provide cross-platform compatibility, which means that a container designed to run in a Docker container may run in a Windows container, and vice versa. If you design your container to be flexible with the underlying operating system, you may lose features that often lead software architects to not provide this OS compatibility within containers at all.

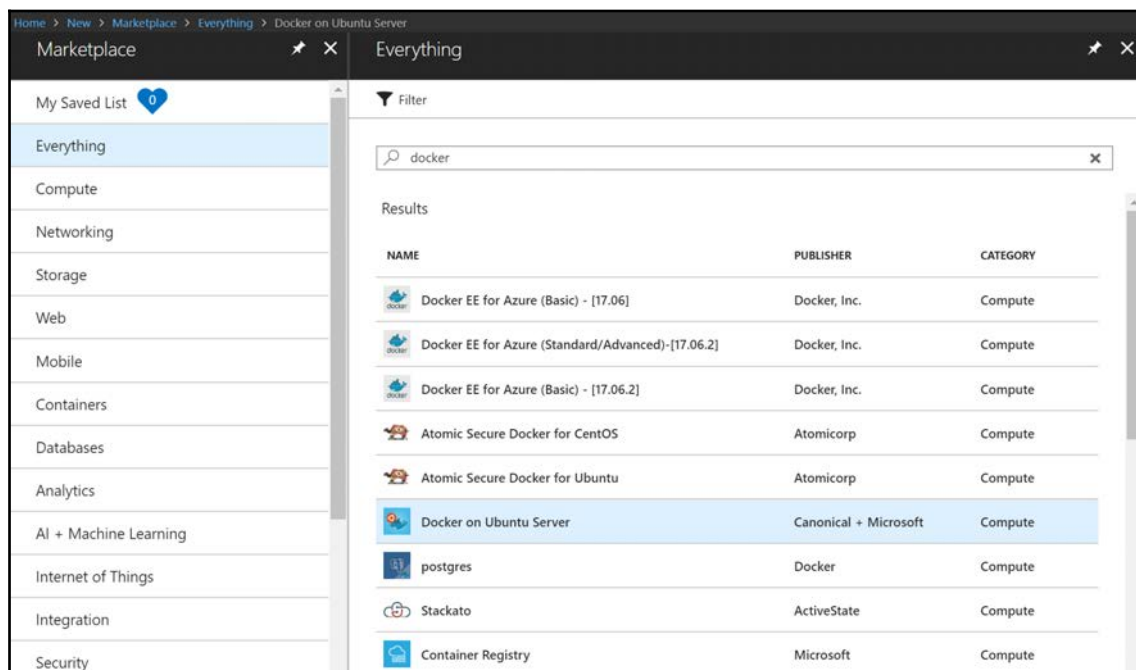
- Containers are not designed to run applications with graphical interfaces. Although some tricks may provide this feature, the result is more than clunky. Docker is not a good solution for applications that require rich interfaces.
- Using Docker may provide a way to improve security, as its concept involves isolating applications from the host system and breaking an application into small microservices. This means that if one microservice is compromised, the others are not necessarily compromised, too. On the other hand, containerization brings additional requirements for security as it is a highly scalable environment, and this means that you will need to monitor the complete solution somehow.

Deploying container hosts in Azure

Deploying a container host in Azure is quite easy, as it is just a VM extension. You just have to choose the appropriate Azure Marketplace item and deploy it as follows.

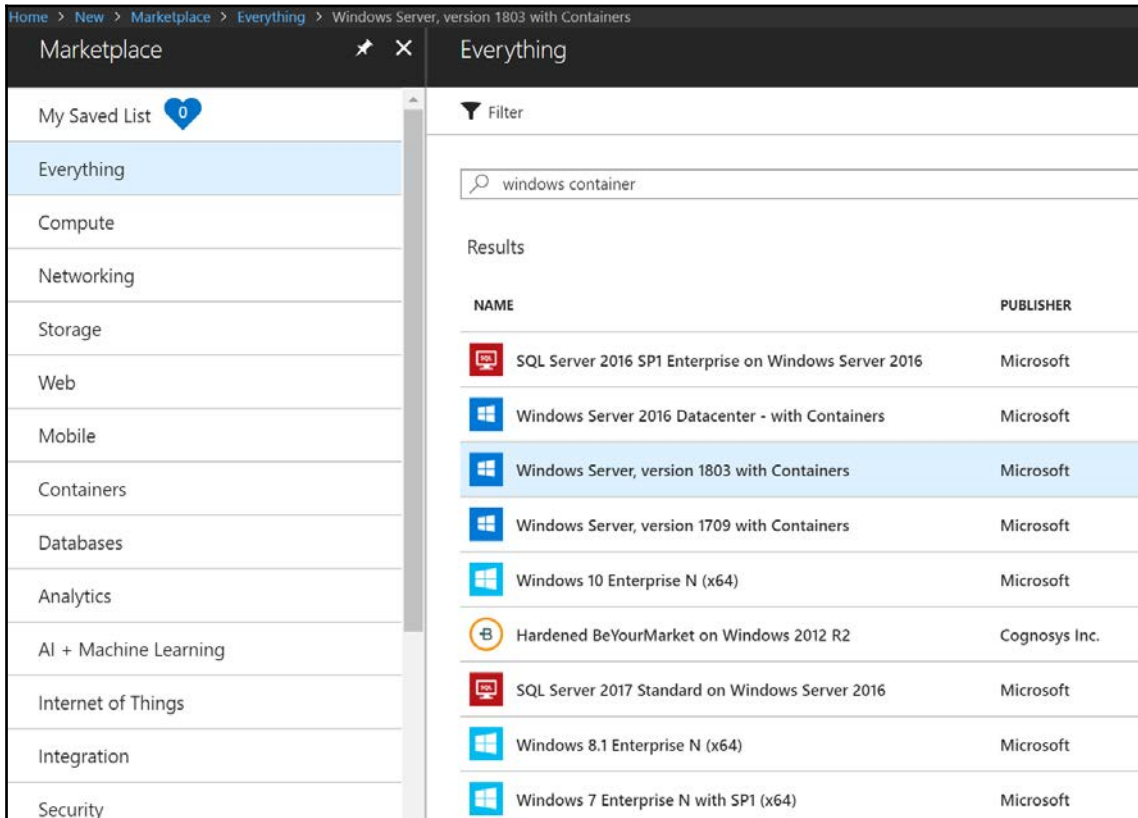
Docker on Linux

As you can see in the following screenshot, there is quite a huge amount of Docker/container stuff that is available in the Azure Marketplace:



Windows Server Container VM

In addition to the huge number of containers, you could even upload your own Azure custom IaaS image with Docker/container services enabled:



The screenshot shows the Azure Marketplace interface. The breadcrumb trail at the top reads: Home > New > Marketplace > Everything > Windows Server, version 1803 with Containers. The left sidebar shows a 'My Saved List' with 0 items and a list of categories: Everything (selected), Compute, Networking, Storage, Web, Mobile, Containers, Databases, Analytics, AI + Machine Learning, Internet of Things, Integration, and Security. The main area displays search results for 'windows container'. A table lists the results with columns for NAME and PUBLISHER.

NAME	PUBLISHER
SQL Server 2016 SP1 Enterprise on Windows Server 2016	Microsoft
Windows Server 2016 Datacenter - with Containers	Microsoft
Windows Server, version 1803 with Containers	Microsoft
Windows Server, version 1709 with Containers	Microsoft
Windows 10 Enterprise N (x64)	Microsoft
Hardened BeYourMarket on Windows 2012 R2	Cognosys Inc.
SQL Server 2017 Standard on Windows Server 2016	Microsoft
Windows 8.1 Enterprise N (x64)	Microsoft
Windows 7 Enterprise N with SP1 (x64)	Microsoft

The way described previously is in general not the way you should provide containers in a cloud environment. With Azure we have easier ways of doing so, but if you need it, you could do it that way. As with Azure, you will have the option to order a container directly in Azure as an **Azure Container Instance (ACI)** or a managed Azure solution using Kubernetes as orchestrator, which we will talk about later in this chapter.



To administer a Docker container from the command line, Azure CLI provides a great implementation: <https://hub.docker.com/r/microsoft/azure-cli/>.

Azure Container Registry (ACR)

If you need to set up a container environment to be used by the developers in your Azure tenant, you will have to think about where to store your container images. In general, the way to do this is to provide a container registry. This registry could reside on a VM itself, but using PaaS services with cloud technologies always provides an easier and more flexible design.

This is where **Azure Container Service (ACS)** comes in, as it is a PaaS solution that provides high flexibility and even features such as replication between geographies.

This means you will need to fill in the following details:

Home > New > Container Registry > Create container registry

Container Registry

Microsoft

Azure Container Registry is a private registry for hosting container images. Using the Azure Container Registry, you can store Docker-formatted images for all types of container deployments. Azure Container Registry integrates well with orchestrators hosted in Azure Container Service, including Docker Swarm, DC/OS, and Kubernetes. Users can benefit from using familiar tooling capable of working with the open source Docker Registry v2.

Use Azure Container Registry to:

- Store and manage container images across all types of Azure deployments
- Use familiar, open-source Docker command line interface (CLI) tools
- Keep container images near deployments to reduce latency and costs
- Simplify registry access management with Azure Active Directory
- Maintain Windows and Linux container images in a single Docker registry

[Save for later](#)

PUBLISHER Microsoft

USEFUL LINKS [Learn more](#)
[Documentation](#)
[Pricing details](#)

[Create](#)

Create container registry

* Registry name
MyContainerRegistry ✓ [azurecr.io](#)

* Subscription
Visual Studio Ultimate bei MSDN

* Resource group
☒ Create new ☐ Use existing
AzureBook ✓

* Location
West Europe

* Admin user ⓘ
[Enable](#) [Disable](#)

* SKU ⓘ
Standard ^
Basic
Standard
Premium

[Create](#) [Automation options](#)

When you create your container registry, you will need to define the following:

- The registry name (ending with `azurecr.io`)
- The resource group the registry sits in
- The Azure location
- The admin user (if you will need to log in to the registry using an account)
- The SKU:
 - Basic
 - Standard
 - Premium

The following table details the features and limits of the basic, standard, and premium service tiers:

Resource	Basic	Standard	Premium
Storage	10 GiB	100 GiB	500 GiB
Max image layer size	20 GiB	20 GiB	50 GiB
ReadOps per minute	1,000	3,000	10,000
WriteOps per minute	100	500	2,000
Download bandwidth MBps	30	60	100
Upload bandwidth MBps	10	20	50
Webhooks	2	10	100
Geo-replication	N/A	N/A	Supported (https://docs.microsoft.com/en-us/azure/container-registry/container-registry-geo-replication)

Switching between the different SKUs is supported and can be done using the portal, PowerShell, or CLI.

If you still are on a classic ACR, the first step would be to upgrade to a managed registry using the following steps at <https://docs.microsoft.com/en-us/azure/container-registry/container-registry-upgrade>.

From the best-practice point of view, you should design your ACR that way, to provide it near to the region where you are planning to deploy the containers themselves.

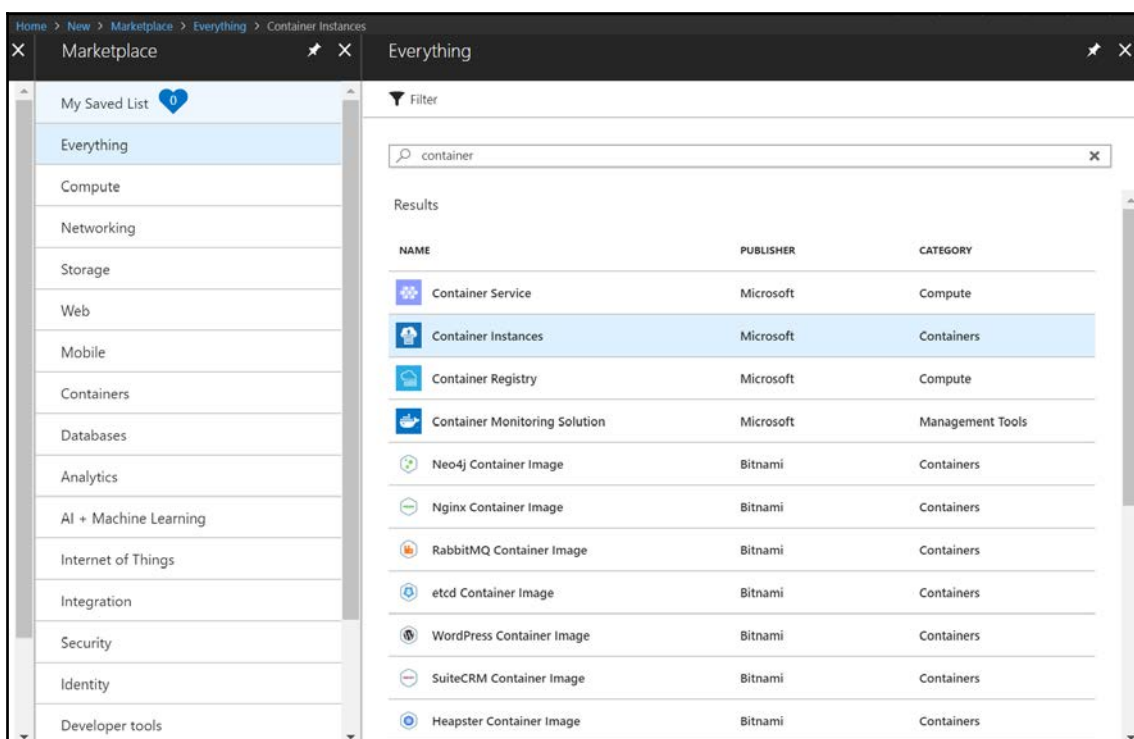
ACI

As mentioned already, running just container instances in Azure is quite easy. By running your workloads in ACI, you don't have to set up a management infrastructure for your containers, you just can put your focus on design and building the applications.

Creating a first container in Azure

Let's create a first simple container in Azure using the portal:

1. Go to **Container Instances** under **New | Marketplace | Everything**, as shown in the following screenshot:



2. After having chosen the **Container Instances** entry in the resources list, you will have to define some properties, which are described as follows:

The screenshot shows the 'Create Container Instances' wizard in the Azure portal, specifically the 'Basics' step. The left sidebar contains navigation links like 'Create a resource', 'All services', and 'FAVORITES'. The main area has a breadcrumb trail: 'Home > New > Marketplace > Everything > Container Instances > Create Container Instances > Basics'. The wizard consists of three steps: 1. Basics (Configure basic settings), 2. Configuration (Specify container requirements), and 3. Summary (Container Instances). The 'Basics' step is active and shows the following fields: 'Container name' (myazurecontainer), 'Container image type' (Public/Private), 'Container image' (microsoft/aci-helloworld), 'Subscription' (Visual Studio Ultimate bei MSDN), 'Resource group' (Create new / Use existing, mycontainer), and 'Location' (West Europe). An 'OK' button is at the bottom right.

At first, we will need to define the Azure container name. Of course, this needs to be unique in your environment. Then, we will need to define the source of the image and to which resource group and region it should be deployed within Azure.

3. As already mentioned, containers can reside on Windows and Linux, because this needs to be defined at first. Afterwards, we will need to define the resources per container:
 - Cores
 - Memory
 - Ports
 - Port protocol

- Restart policy (if the container went offline)

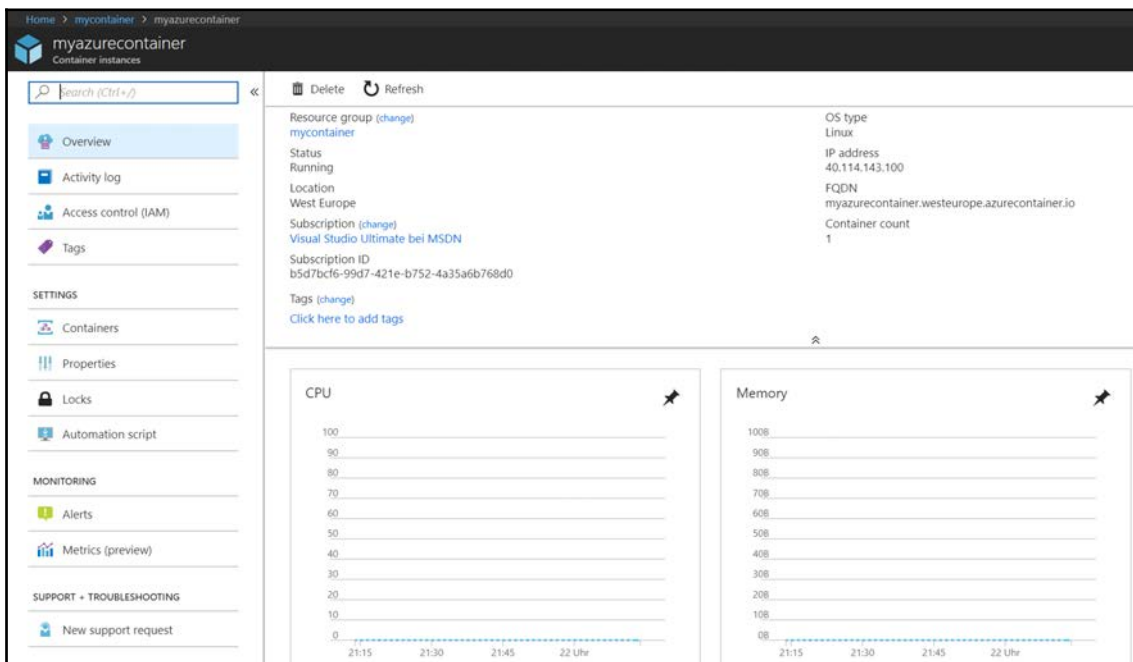
The screenshot displays the Microsoft Azure portal interface for creating container instances. The left sidebar shows the navigation menu with options like 'Create a resource', 'All services', and 'FAVORITES'. The main area is titled 'Create Container Instances' and shows a progress bar with three steps: 1. Basics (Done), 2. Configuration (Specify container requirements), and 3. Summary (Container Instances). The 'Configuration' step is currently active, showing various settings for the container instance.

Configuration Details:

- OS Type:** Windows (selected), Linux (selected)
- Number of cores:** 1
- Memory (GB):** 1.5
- Networking:**
 - Public IP address: Yes (selected), No (selected)
 - DNS name label: myazurecontainer (selected)
 - Port: 80 (selected)
 - Open additional ports: Yes (selected), No (selected)
 - Port protocol: TCP (selected)
- Advanced:**
 - Restart policy: Always (selected)
 - Environment variable: (empty)
 - Add additional environment variables: Yes (selected), No (selected)
 - Command override: (empty)

The 'OK' button is visible at the bottom right of the configuration panel.

4. After having deployed the corresponding container registry, we can start working with the container instance:



5. When hitting the URL posted in the left part, under FQDN, you should see the following screenshot:



After we have finalized the preceding steps, we have an ACI up and running, which means that you are able to provide container images, load them up to Azure, and run them.

Azure Marketplace containers

In the public Azure Marketplace, you can find existing container images that just can be deployed to your subscription. These are pre-packaged images that give you the option to start with your first container in Azure. As cloud services provide reusability and standardization, this entry point is always good to look at first.

1. Before starting with this, we will need to check if the required resource providers are enabled on the subscription you are working with. Otherwise, we will need to register them by hitting the **Register** entry and waiting a few minutes for completion, as shown in the following screenshot:

The screenshot shows the Azure portal interface for a subscription named 'Visual Studio Enterprise - Resource providers'. The left sidebar contains navigation options like 'Subscriptions', 'Settings', 'Resource groups', 'Resources', 'Usage + quotas', 'Policies', 'Management certificates', 'My permissions', 'Resource providers' (highlighted), 'Properties', 'Resource locks', and 'Support + Troubleshooting'. The main content area displays a table of resource providers. A search bar at the top of the table contains the text 'container'. The table lists three providers: Microsoft.ContainerInstance, Microsoft.ContainerRegistry, and Microsoft.ContainerService, all with a status of 'NotRegistered'. Each row has a 'Register' link next to it.

PROVIDER	STATUS	
Microsoft.ContainerInstance	NotRegistered	Register
Microsoft.ContainerRegistry	NotRegistered	Register
Microsoft.ContainerService	NotRegistered	Register

2. Now, we can start deploying marketplace containers such as the container image for WordPress, which is used as a sample, as shown in the following screenshot:

The screenshot shows the 'Subscribe' page for the 'WordPress Container Image' by Bitnami in the Azure Marketplace. The page is divided into two main sections: a left sidebar with product details and a right main area with subscription configuration.

Left Sidebar (WordPress Container Image details):

- PUBLISHER:** Bitnami
- USEFUL LINKS:** [Launching on AKS](#), [README](#), [Get started with ACI](#), [Support](#), [Learn More](#)
- SUPPORT:** <https://bitnami.com/support/azure>
- Available WordPress Container Image versions:** 4, 4-debian-8, 4.9.5, 4.9.6, 4.9.7-debian-8, latest
- Buttons:** Save for later, Create

Right Main Area (Subscription configuration):

- Subscription:** Visual Studio Enterprise
- Azure Container Registry:** Create new (selected) / Use existing. Value: acrweb01
- Resource group:** Create new (selected) / Use existing. Value: acrweb01
- Location:** West Europe
- Tag:** 4.9.6
- Auto-update:** ☐ Auto-update ⓘ
- Info:** Deploying an Azure container registry will incur charges. See pricing details [here](#)
- Offer details:** WordPress Container Image by Bitnami. [Terms of use](#) | [privacy policy](#)
- Subscribe Button:** A blue button labeled 'Subscribe'.

At first, we will need to decide for the corresponding image and choose to create a new ACR, or use an existing one. Furthermore, the Azure region, the resource group, and the tag (for example, version) need to be defined in the following dialog:

The screenshot displays the Azure portal interface for an Azure Container Registry (acrweb01). The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Quick start, Events, SETTINGS (Access keys, Locks, Automation script), SERVICES (Repositories, Webhooks, Replications), and POLICIES. The main content area shows the registry's details and usage.

acrweb01
Container registry

Search (Ctrl+/)

Move Delete Update

Essentials

Resource group: [arcweb01](#)
Location: West Europe
Subscription name: [Visual Studio Enterprise](#)
Subscription ID: f3248cad-78cf-4116-bee1-01151da2d51d

Login server: [acrweb01.azurecr.io](#)
Creation date: 8/17/2018, 2:05 PM GMT+2
SKU: Basic
Provisioning state: Succeeded

Registry usage

Registry quota usage

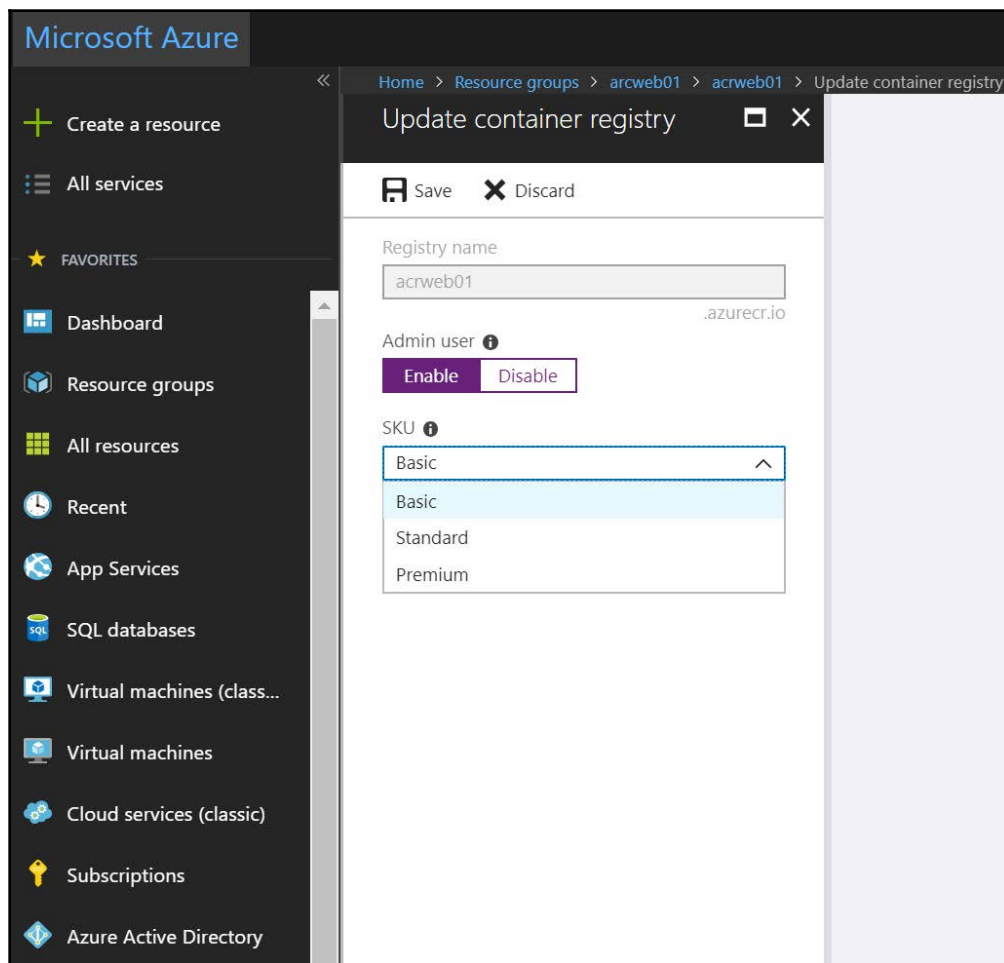
USED: 0.0 GiB
AVAILABLE IN SKU: 10.0 GiB
10 GiB SIZE QUOTA

Container security integrations

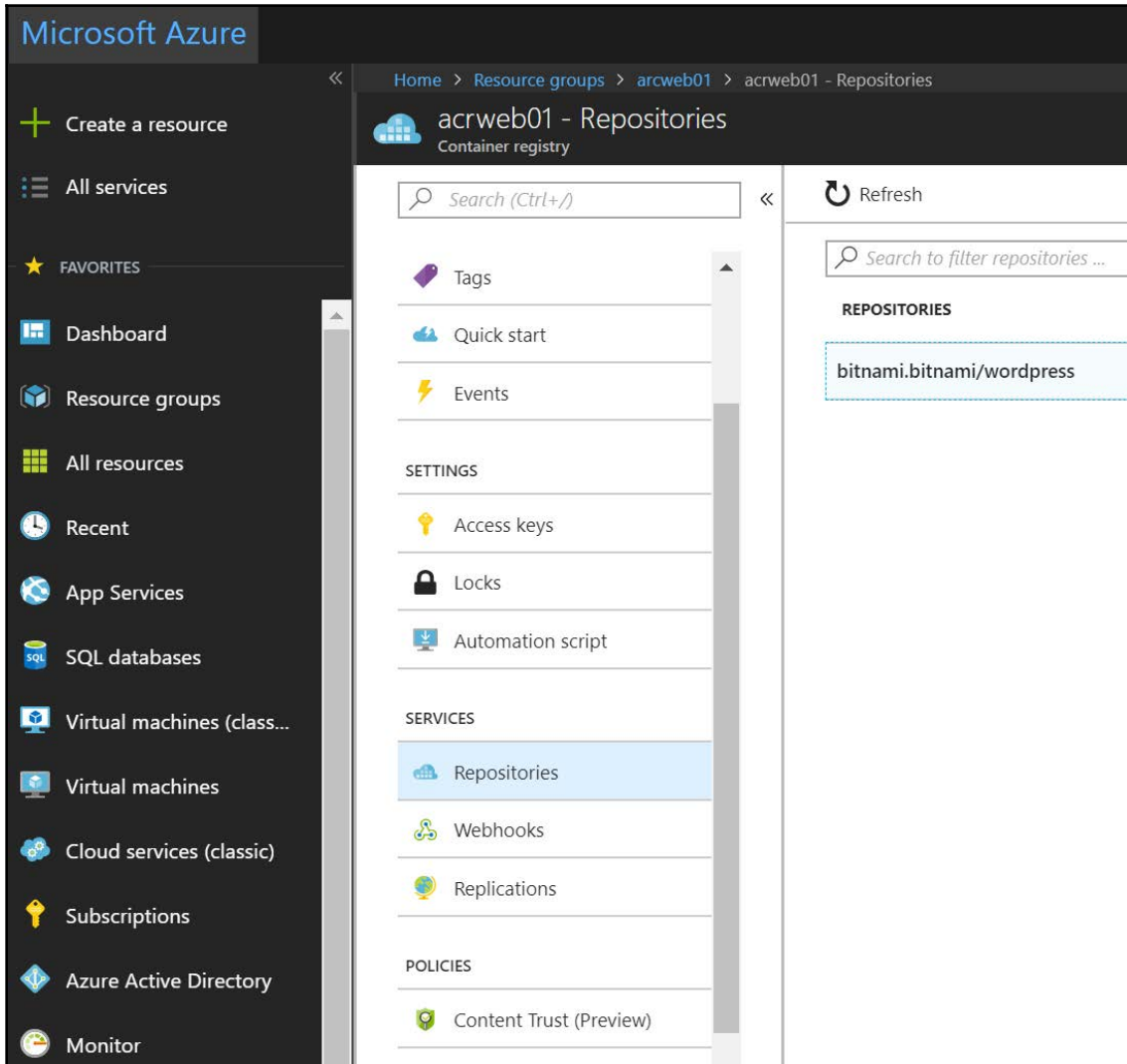
Aqua Security
Aqua provides development-to-production lifecycle controls for securing containerized applications.

Twistlock
Providing vulnerability management and runtime protection across your environments.
[Azure Marketplace](#)

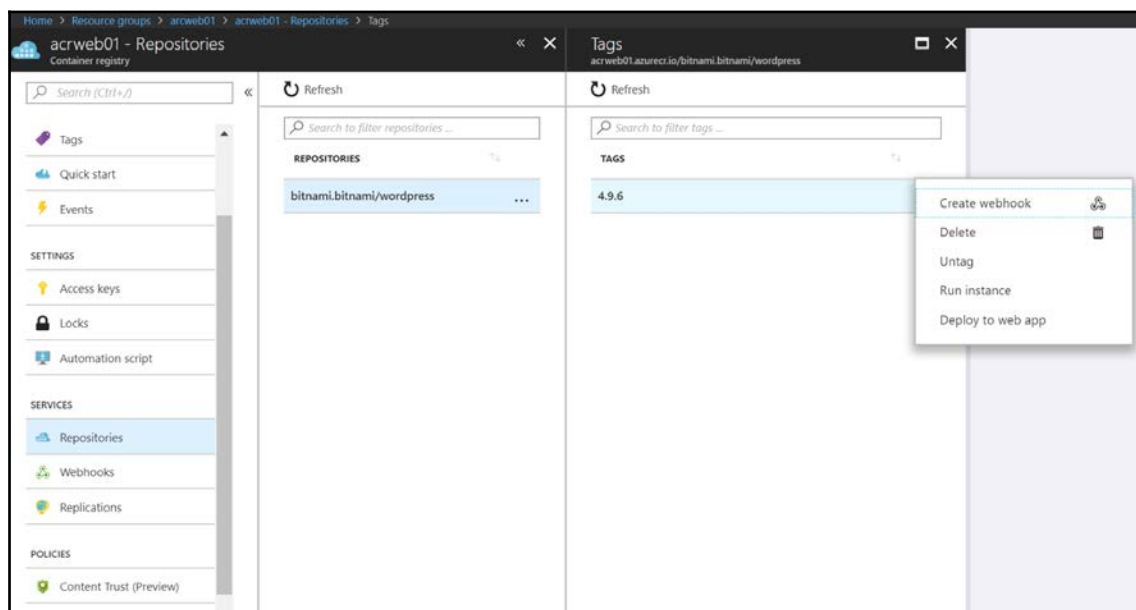
3. Now that the registry is being created, we will need to update the permission settings, also called enable admin registry. This can be done with the **Admin user Enable** button as shown in the following screenshot:



- Regarding the SKU, this is just another point where we can set the priority and define performance. This may take some minutes to be enabled. Now, we can start deploying container images from the container registry, as you can see in the following screenshot with the WordPress image that is already available in the registry:



5. At first, we will need to choose the corresponding container from the registry; right-click the tag version from the **Tags** section:

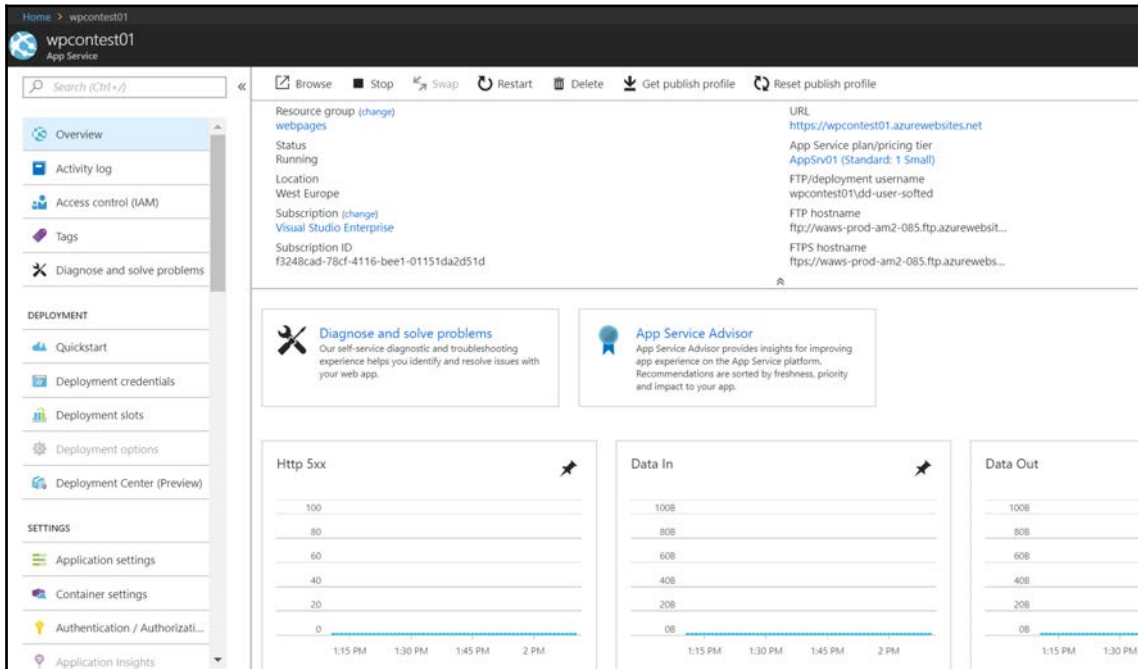


6. Having done that, we will need to hit the **Deploy to web app** menu entry to deploy the web app to Azure:

The screenshot displays three side-by-side panels in the Azure portal interface, each with a close button (X) in the top right corner.

- Web App for Containers:** This panel is titled "Create". It contains several form fields:
 - Site name:** A text box with "wpcontest01" and a green checkmark.
 - Subscription:** A dropdown menu showing "Visual Studio Enterprise".
 - Resource group:** Radio buttons for "Create new" (selected) and "Use existing". Below is a text box with "webpages" and a green checkmark.
 - App service plan/Location:** A light blue box with a right-pointing chevron (>).
 - Image:** A text box with "bitnami.bitnami/wordpress:4.9.6".
 - Info box:** A grey box with an information icon (i) stating: "This enables continuous deployment by creating a webhook named 'wpcontest01120922'".
 - Create button:** A blue button at the bottom left.
- App service plan:** This panel is titled "Select a plan for the web app". It contains:
 - Create new button:** A light blue button with a plus icon (+) and the text "Create new".
 - No result:** Text below the button.
- New app service plan:** This panel is titled "Create a plan for the web app". It contains:
 - App service plan name:** A text box with "AppSrv01" and a green checkmark.
 - Location:** A dropdown menu showing "West Europe".
 - Info box:** A grey box with an information icon (i) stating: "The pricing tier currently defaults to 'S1 Standard'. It can be modified by visiting the [app service plan resource page](#) once the resource has been created, or you can choose an existing plan in your subscription."
 - OK button:** A blue button at the bottom right.

7. As the properties that need to be filled are some defaults for **Web Apps**, it is quite easy to set them:



Finally, the first containerized image for a web app has been deployed to Azure.

Creating custom containers

Having now learned how containers in general work, and how to create the required infrastructure around them (for example, **registries**), let's have a look how to create a customer container from a custom app.

Before creating a container from a custom app, it is quite important to think about the design of containers we talked about earlier in this chapter (for example, no GUI and already microservice).



A good tutorial on creating a custom container from your app can be found at <https://docs.microsoft.com/en-us/azure/container-instances/container-instances-tutorial-prepare-app>.

Container orchestration

In the first part of this chapter, we understood the concept of containers, how they are designed, how you can prepare a service to create your own container, and how to deploy them in an Azure environment.

One of the most interesting topics with regard to containers is that they provide a technology for scaling. For example, if we need more performance on a website that is running containerized, we would just spin off an additional container to load-balance the traffic. This could even be done if we needed to scale down.

The concept of container orchestration

Regarding this technology, we need an orchestration tool to provide this feature set. There are some well-known container orchestration tools available on the market, such as the following:

- Docker swarm
- DC/OS
- Kubernetes

Kubernetes is the most-used one, and therefore could be deployed as a service in most public cloud services, such as in Azure. In general, it provides the following features:

- **Automated container placement:** On the container hosts, to best spread the load between them
- **Self-healing:** For failed containers, restarting them in a proper way
- **Horizontal scaling:** Automated horizontal scaling (up and down) based on the existing load

- **Service discovery and load balancing:** By providing IP-addresses to containers and managing DNS registrations
- **Rollout and rollback:** Automated rollout and rollback for containers, which provides another self-healing feature as updated containers that are newly rolled-out are just rolled back if something goes wrong
- **Configuration management:** By updating secrets and configurations without the need to fully rebuild the container itself

Azure Kubernetes Service (AKS)

Installing, maintaining, and administering a Kubernetes cluster manually could mean a huge investment of time for a company. In general, these tasks are one-off costs and therefore it would be best to not waste these resources. In Azure today, there is a feature called **AKS**, where **K** emphasizes that it is a managed Kubernetes service.

Right after containers became interesting for cloud services, Microsoft launched a first service called which was in general the same, but if you ordered it, Azure created a custom ACS environment (dedicated to your tenant) for templated deployments. With ACS, the choice was just available to the most popular orchestrators—**Mesos**, **Swarm**, and **Kubernetes**. With ACS, you have to pay for the master servers of the orchestrator.

For AKS, there is no charge for Kubernetes masters, you just have to pay for the nodes that are running the containers.

Before you start, you will have to fulfill the following prerequisites:

- An Azure account with an active subscription
- Azure CLI installed and configured
- Kubernetes command-line tool, `kubectl`, installed

- Make sure that the Azure subscription you use has these required resources—storage, compute, networking, and a container service:

Home > New > Marketplace > Everything > Kubernetes Service > Create Kubernetes cluster

Create Kubernetes cluster

Basics Authentication Networking Monitoring Tags Review + create

Azure Kubernetes Service (AKS) manages your hosted Kubernetes environment, making it quick and easy to deploy and manage containerized applications without container orchestration expertise. It also eliminates the burden of ongoing operations and maintenance by provisioning, upgrading, and scaling resources on demand, without taking your applications offline. [Learn more about Azure Kubernetes Service](#)

Project details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

* Subscription ⓘ Visual Studio Enterprise ✓

* Resource group ⓘ ☒ Create new ☐ Use existing
AKS01 ✓

Cluster details

* Kubernetes cluster name ⓘ kubecluster ✓

* Region ⓘ West Europe ✓

* Kubernetes version ⓘ 1.11.1 ✓

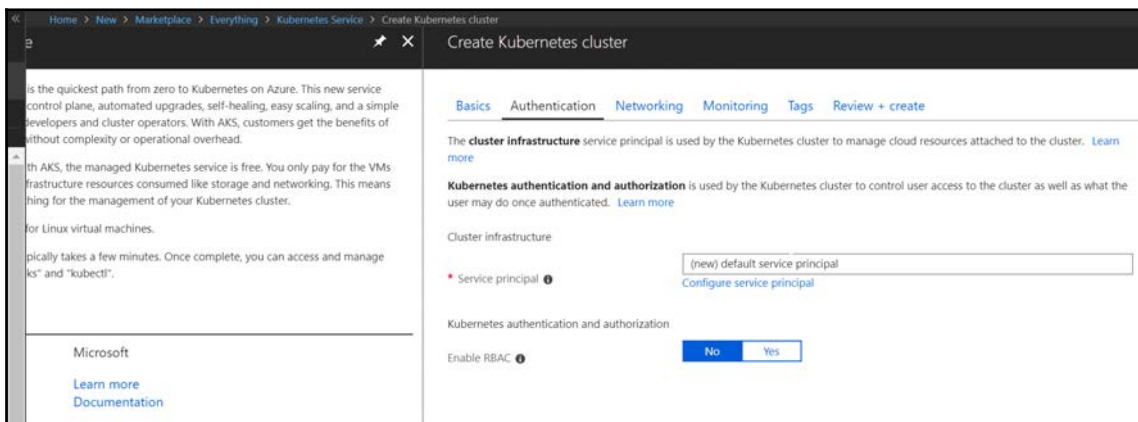
* DNS name prefix ⓘ myaks ✓

Scale

The number and size of nodes in your cluster. For production workloads, at least 3 nodes are recommended for resiliency. For development or test workloads, only one node is required. You will not be able to change the node size after cluster creation, but you will be able to change the number of nodes in your cluster after creation. [Learn more about scaling in Azure Kubernetes Service](#)

[Review + create](#) [Next: Authentication »](#) [Download a template for automation](#)

1. For the first step, you need to choose **Kubernetes service** and choose to create your AKS deployment for your tenant. The following parameters need to be defined:
 - Resource group for the deployment
 - Kubernetes cluster name
 - Azure region
 - Kubernetes version
 - DNS prefix
2. Then, hit the **Authentication** tab, as shown in the following screenshot:

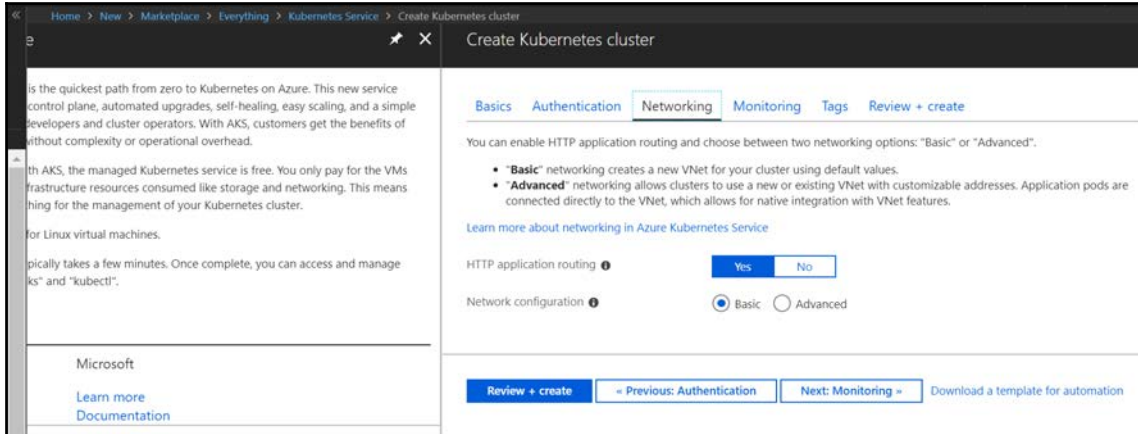


3. On the **Authentication** tab, you will need to define a service principal or choose an existing one, as AKS needs a service principal to run the deployment. In addition, you could enable the RBAC feature, which gives you the chance to define fine-grained permissions based on Azure AD accounts and groups.



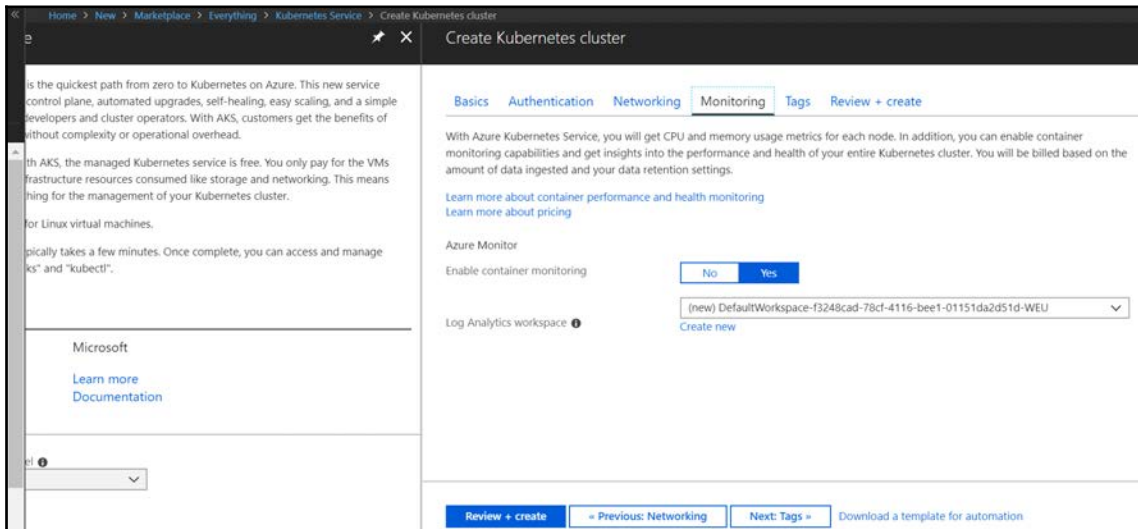
For more information on RBAC with AKS, visit the following URL at <https://docs.microsoft.com/en-us/azure/aks/aad-integration>.

4. On the **Networking** tab, you can choose either to add the Kubernetes cluster into an existing VNET, or create a new one. In addition, the HTTP routing feature can be enabled or disabled:



The screenshot shows the 'Create Kubernetes cluster' wizard in the Azure portal. The 'Networking' tab is selected. The left sidebar contains a description of AKS and a 'Learn more' link. The main content area explains that you can enable HTTP application routing and choose between 'Basic' or 'Advanced' networking options. The 'HTTP application routing' toggle is set to 'Yes'. The 'Network configuration' radio buttons are set to 'Basic'. At the bottom, there are buttons for 'Review + create', '< Previous: Authentication', 'Next: Monitoring >', and a link to 'Download a template for automation'.

5. On the **Monitoring** tab, you have the option to enable container monitoring and link it to an existing Log Analytics workspace, or create a new one:



The screenshot shows the 'Create Kubernetes cluster' wizard in the Azure portal, with the 'Monitoring' tab selected. The left sidebar is the same as in the previous screenshot. The main content area explains that you can enable container monitoring and get insights into the performance and health of your entire Kubernetes cluster. The 'Enable container monitoring' toggle is set to 'Yes'. The 'Log Analytics workspace' dropdown menu is set to '(new) DefaultWorkspace-f3248cad-78cf-4116-bee1-01151da2d51d-WEU'. Below the dropdown is a 'Create new' link. At the bottom, there are buttons for 'Review + create', '< Previous: Networking', 'Next: Tags >', and a link to 'Download a template for automation'.

6. As Azure tags are a major feature and requirement for Azure Governance, which we will talk about later in the Chapter 10, *Implementing Azure Governance*, the following is the source from which to set your required tags:

The screenshot shows the 'Create Kubernetes cluster' wizard in the Azure portal. The breadcrumb navigation at the top reads: Home > New > Marketplace > Everything > Kubernetes Service > Create Kubernetes cluster. The left sidebar contains introductory text about AKS, stating it's the quickest path from zero to Kubernetes on Azure, and mentions that with AKS, the managed Kubernetes service is free, with costs only for VMs and infrastructure resources. It also includes links for 'Learn more' and 'Documentation' under the Microsoft logo. The main content area is titled 'Create Kubernetes cluster' and features a tabbed interface with 'Tags' selected. Below the tabs, a text box explains that tags are name/value pairs for categorizing resources and view consolidated billing, with a 'Learn more' link. A note states that tags will be automatically updated if resource settings change. Below this is a table with two columns: 'KEY' and 'VALUE'. The 'KEY' column has a checkbox icon to its left. At the bottom of the wizard, there are navigation buttons: 'Review + create' (highlighted in blue), '< Previous: Monitoring', 'Next: Review + create >', and a link to 'Download a template for automation'.

KEY	VALUE

7. Finally, the validation will check for any misconfigurations and create the Azure ARM template for the deployment. Clicking the **Create** button will start the deployment phase, which could run for several minutes or even longer depending on the chosen feature, and scale:

Home > New > Marketplace > Everything > Kubernetes Service > Create Kubernetes cluster

is the quickest path from zero to Kubernetes on Azure. This new service control plane, automated upgrades, self-healing, easy scaling, and a simple developers and cluster operators. With AKS, customers get the benefits of without complexity or operational overhead.

With AKS, the managed Kubernetes service is free. You only pay for the VMs infrastructure resources consumed like storage and networking. This means nothing for the management of your Kubernetes cluster.

for Linux virtual machines.

Typically takes a few minutes. Once complete, you can access and manage kubernetes and "kubectl".

Microsoft

[Learn more](#)
[Documentation](#)

Running final validation...

Basics Authentication Networking Monitoring Tags **Review + create**

BASICS

Subscription	Visual Studio Enterprise
Resource group	(new) AKS01
Region	West Europe
Kubernetes cluster name	kubecluster
Kubernetes version	1.11.1
DNS name prefix	myaks
Node count	3
Node size	Standard_DS1_v2

AUTHENTICATION

Enable RBAC	No
-------------	----

NETWORKING

HTTP application routing	Yes
Network configuration	Basic

MONITORING

Enable container monitoring	Yes
Log Analytics workspace	(new) DefaultWorkspace-f3248cad-78cf-4116-bee1-01151da2d51d-WEU

TAGS

Create Previous: Tags Download a template for automation

8. After the deployment has finished, the Kubernetes dashboard is available. You can view the Kubernetes dashboard by clicking on the **View Kubernetes dashboard** link, as shown in the following screenshot:

The screenshot displays the Azure portal interface for a Kubernetes service named 'kubecluster'. The breadcrumb navigation at the top shows 'Home > kubecluster'. The page header includes the 'kubecluster' name and 'Kubernetes service' label, along with a star icon. Below the header, there are three action buttons: 'Move', 'Delete', and 'Refresh'. The main content area is divided into two columns. The left column lists details for the 'Resource group (change) AKS01', including 'Status Succeeded', 'Location West Europe', 'Subscription (change) Visual Studio Enterprise', and 'Subscription ID f3248cad-78cf-4116-bee1-01151da2d51d'. The right column lists 'Kubernetes version 1.11.1', 'API server address myaks-900c6fe3.hcp.westeurope.azmk8s.io', 'Total cores 3', 'Total memory 10.5', and 'HTTP application routing domain 0c67c6a16d994ed4b7a5.westeurope.aksa...'. Below these columns, there are three tiles. The first tile is 'Monitor container health' with a globe icon, describing 'Get health and performance insights' and providing a link 'Go to Azure Monitor container health'. The second tile is 'Search logs' with a magnifying glass icon, describing 'Search and analyze logs using ad-hoc queries' and providing a link 'Go to Azure Monitor logs'. The third tile is 'View Kubernetes dashboard' with a dashboard icon, describing 'Learn how to connect to the Kubernetes dashboard' and providing a link 'View connection steps'.

Home > kubecluster

kubecluster
Kubernetes service

→ Move Delete Refresh

Resource group (change) AKS01	Kubernetes version 1.11.1
Status Succeeded	API server address myaks-900c6fe3.hcp.westeurope.azmk8s.io
Location West Europe	Total cores 3
Subscription (change) Visual Studio Enterprise	Total memory 10.5
Subscription ID f3248cad-78cf-4116-bee1-01151da2d51d	HTTP application routing domain 0c67c6a16d994ed4b7a5.westeurope.aksa...
Tags (change) Click here to add tags	

⌵

Monitor container health
Get health and performance insights
[Go to Azure Monitor container health](#)

Search logs
Search and analyze logs using ad-hoc queries
[Go to Azure Monitor logs](#)

View Kubernetes dashboard
Learn how to connect to the Kubernetes dashboard
[View connection steps](#)

The dashboard looks something like the one shown in the following screenshot:

Kubernetes dashboard

To open your Kubernetes dashboard, complete the following steps:

1

Open Azure CLI version 2.0.27 or later. This will not work in cloud shell and must be running on your local machine. [How to install the Azure CLI](#)

2

If you do not already have kubectl installed in your CLI, run the following command:

```
az aks install-cli
```

3

Get the credentials for your cluster by running the following command:

```
az aks get-credentials --resource-group AKS01 --name kubecuster
```

4

Open the Kubernetes dashboard by running the following command:

```
az aks browse --resource-group AKS01 --name kubecuster
```

Useful links

[What is Kubernetes dashboard?](#)

[Full instructions for opening the Kubernetes dashboard](#)

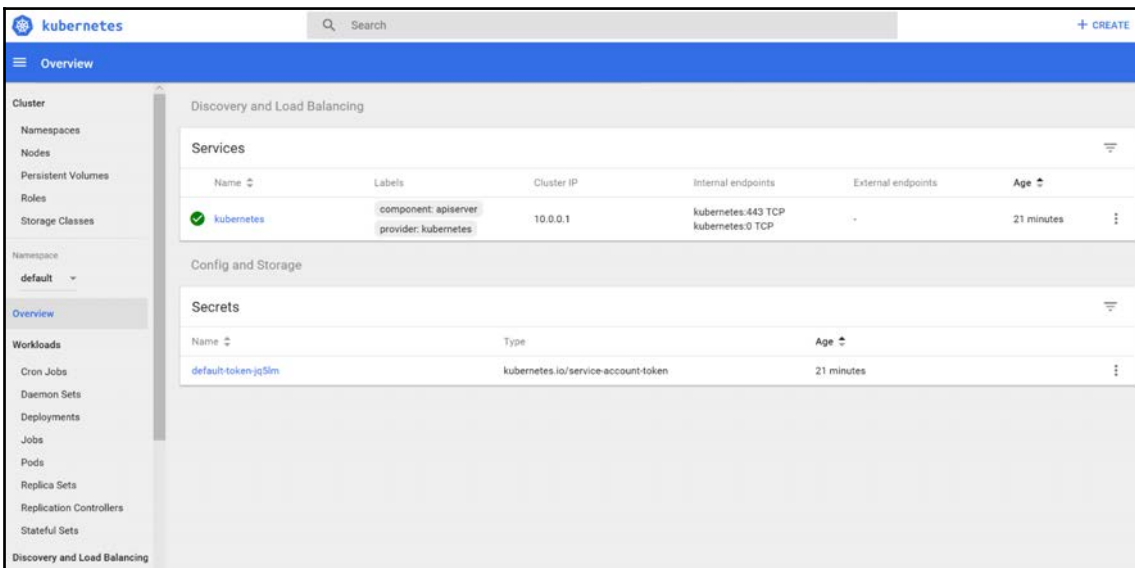
9. As you can see in the preceding screenshot, there are four steps to open the dashboard. At first, we will need to install the Azure CLI in its most current version using the statement that is mentioned in the following screenshot:

```
Administrator: Eingabeaufforderung - az aks install-cli
Microsoft Windows [Version 10.0.17134.228]
(c) 2018 Microsoft Corporation. Alle Rechte vorbehalten.

C:\WINDOWS\system32>az aks install-cli
Downloading client to "C:\Users\maklei\.azure-kubectrl\kubectrl.exe" from "https://storage.googleapis.com/kubernetes-release/release/v1.11.2/bin/windows/amd64/kubectrl.exe"
```

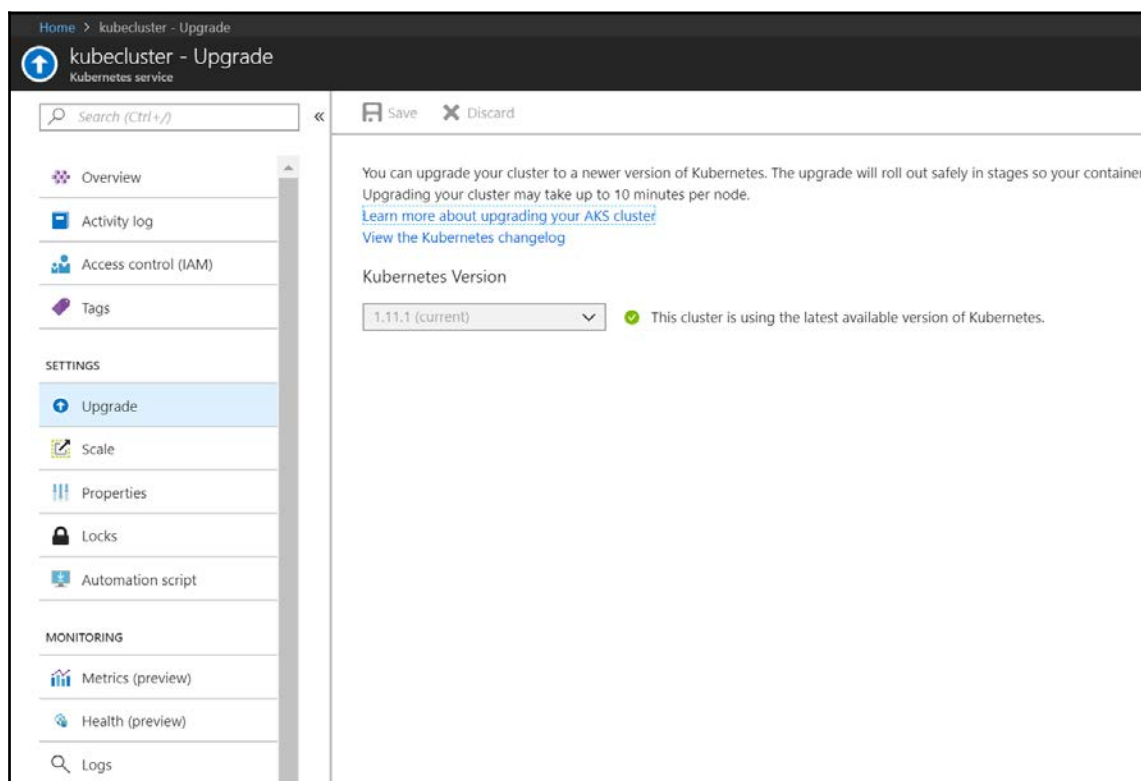
Afterward, the AKS CLI needs to be enabled. It is called `kubectrl.exe`.

10. Finally, after setting all the parameters (and when you have performed steps 3 and 4 from the preceding task list), the following dashboard should open in a new browser window:

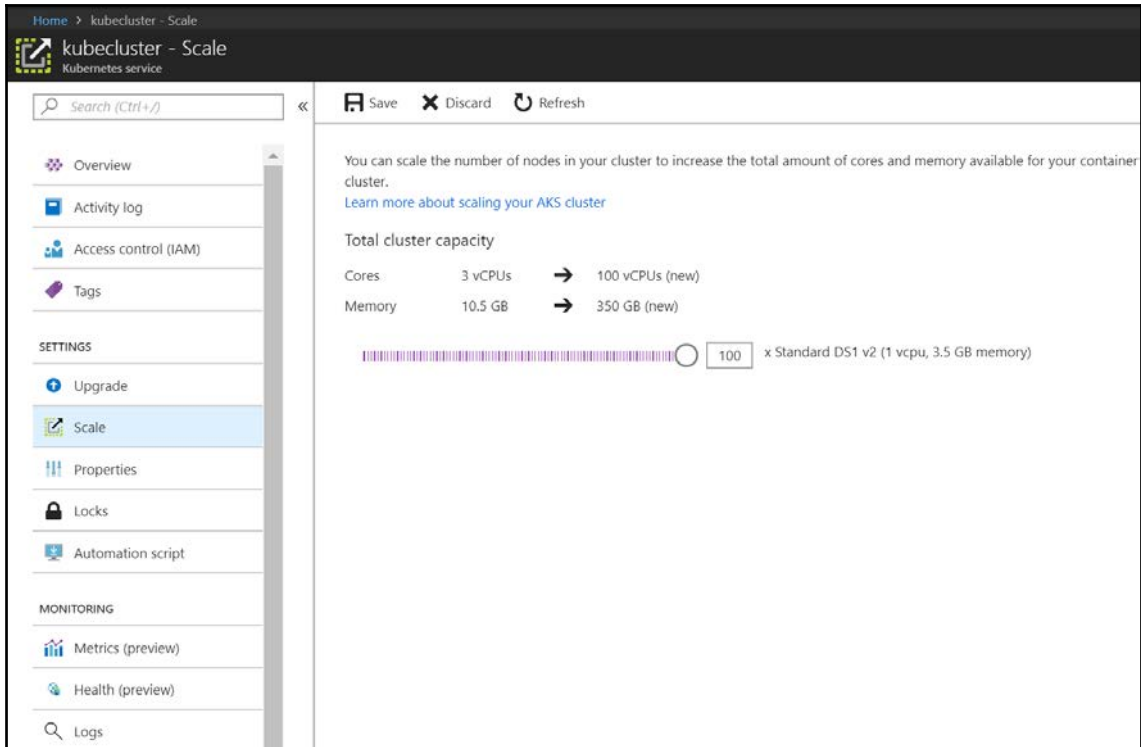


The preceding dashboard provides a way to monitor and administer your Azure Kubernetes environment, in general, from a GUI.

11. If a new Kubernetes version becomes available, you can easily update it from the Azure portal yourself with one click, as shown in the following screenshot:



12. If you need to scale your AKS hosts, this is quite easy too, as you can do it through the Azure portal. A maximum of 100 hosts with 3 vCPUs and 10.5 GB RAM per host is currently possible:



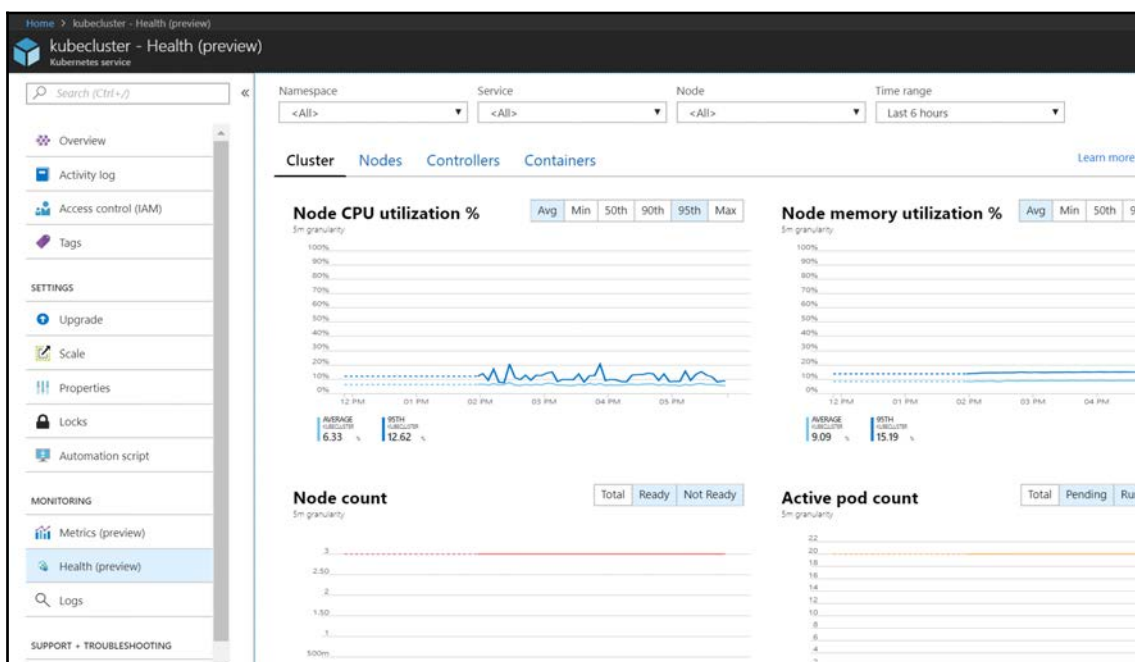
- To enable autoscaler with AKS, the following URL will provide detailed information: <https://docs.microsoft.com/en-us/azure/aks/autoscaler>.
- If you would like to implement AKS using the Azure CLI, the following link is a great source of information: <https://docs.microsoft.com/en-us/azure/aks/kubernetes-walkthrough>.

With all the steps done, you can now upload your containers to your AKS-enabled Docker, and have a huge scalable infrastructure with a minimum of administrative tasks and time for the implementation itself.



If you are working with Terraforms to centrally manage your ARM templates, a valuable description is available at <https://docs.microsoft.com/en-us/azure/terraform/terraform-create-k8s-cluster-with-tf-and-aks?toc=%2Fen-us%2Fazure%2Faks%2FTOC.json&bc=%2Fen-us%2Fazure%2Fbread%2Ftoc.json>.

13. If you need to monitor AKS, the integration with Azure monitoring is integrated completely. By clicking the **Monitor container health** link, you will be directed to the following overview:



The **Nodes** tab provides the following information per node:

The screenshot shows the 'Nodes' tab in the Azure portal for a Kubernetes cluster named 'kubecuster'. The interface includes filters for Namespace, Service, Node, and Time range. A table lists the nodes, and a sidebar provides detailed information for the selected node 'aks-agentpool-3124485-1'.

NAME	STATUS	95TH %	95TH	CONTAINERS	UPTIME	CONTROLLER	TREND 95TH % (1 B)
aks-agentpool-3124485...	Ok	15%	152 mc	17	4 hours	-	
aks-agentpool-3124485...	Ok	7%	67 mc	5	4 hours	-	
aks-agentpool-3124485...	Ok	7%	66 mc	3	4 hours	-	

Node Details (aks-agentpool-3124485-1):

- Node Name: aks-agentpool-3124485-1
- Status: Ready
- Cluster Name: kubecuster
- Kubelet Version: v1.11.1
- Kube Proxy Version: v1.11.1
- Docker Version: 1.13.1
- Operating System: Ubuntu 16.04.5 LTS
- Node IP: 40.114.215.226
- Computer Environment: Azure
- Agent Image: oms

This not only gives a brief overview of the health status, but also the number of containers and the load on the node itself.

14. The **Controllers** view provides detailed information on the AKS controller, its services, status, and uptime:

The screenshot displays the 'kubecluster - Health (preview)' page in the Azure portal. The 'Controllers' tab is active, showing a list of Kubernetes controllers. The table below summarizes the data shown:

NAME	STATUS	95TH % CPU	95TH % MEM	CONTAINERS	RESTARTS	UPTIME
omsagent-rs-55847b4c4 (ReplicaSet)	1	5%	8 mc	1	0	4 hours
tunnelfront-6b689567f (ReplicaSet)	1	4%	41 mc	1	0	4 hours
omsagent (DaemonSet)	3	4%	7 mc	3	0	4 hours
kube-svc-redirect (DaemonSet)	3	3%	25 mc	3	0	4 hours
heapster-6c9777d684 (ReplicaSet)	1	1%	0.9 mc	2	0	4 hours
azureproxy-776884fb9 (ReplicaSet)	1	0.9%	9 mc	1	0	4 hours
addon-http-application-routing-default... (ReplicaSet)	1	0.5%	0.1 mc	1	0	4 hours
kubernetes-dashboard-5fd84f4858 (ReplicaSet)	1	0.5%	0.5 mc	1	0	4 hours
kube-proxy (DaemonSet)	3	0.3%	3 mc	3	0	4 hours
addon-http-application-routing-nginx... (ReplicaSet)	1	0.3%	3 mc	1	0	4 hours
metrics-server-789c47657d (ReplicaSet)	1	0.1%	1 mc	1	0	4 hours
kube-dns-v20-56b5b56d (ReplicaSet)	2	0.1%	0.9 mc	6	0	4 hours

The right sidebar provides details for the selected controller, **omsagent-rs-55847b4c4**:

- Controller Name: omsagent-rs-55847b4c4
- Namespace: kube-system
- Controller Kind: ReplicaSet
- Pod Count: 1
- Container Count: 1
- Service Name: -

15. And finally, the **Containers** tab gives a deep overview of the health state of each container running in the infrastructure (system containers included):

The screenshot shows the 'Containers' tab in the Azure portal for a Kubernetes cluster. The main table lists the following containers:

NAME	STATUS	95TH %	95TH	POD	NODE	RESTARTS	UPT
omsagent	Ok	11%	16 mc	omsagent-5f78b	aks-agentpool-312...	0	4 h
omsagent	Ok	5%	8 mc	omsagent-rs-55847...	aks-agentpool-312...	0	4 h
tunnel-front	Ok	4%	41 mc	tunnelfront-6b6989...	aks-agentpool-312...	0	4 h
redirector	Ok	3%	27 mc	kube-svc-redirect-4...	aks-agentpool-312...	0	4 h
redirector	Ok	3%	26 mc	kube-svc-redirect-x...	aks-agentpool-312...	0	4 h
redirector	Ok	3%	25 mc	kube-svc-redirect-n...	aks-agentpool-312...	0	4 h
heapster-nanny	Ok	1%	0.7 mc	heapster-6c9777d6...	aks-agentpool-312...	0	4 h
omsagent	Ok	1%	2 mc	omsagent-hrpr2	aks-agentpool-312...	0	4 h
omsagent	Ok	1%	2 mc	omsagent-gsvz7	aks-agentpool-312...	0	4 h
heapster	Ok	1%	1 mc	heapster-6c9777d6...	aks-agentpool-312...	0	4 h
azureproxy	Ok	0.9%	9 mc	azureproxy-776884f...	aks-agentpool-312...	0	4 h
addon-http-application-routing-d...	Ok	0.5%	0.1 mc	addon-http-applica...	aks-agentpool-312...	0	4 h

The right sidebar shows details for the 'omsagent' container:

- Container Name: omsagent
- Container ID: df76025e9d4a61405782adbf101c3736e1b27f0539d6423e58e24cd455
- Container Status: running
- Image: oms
- Image Tag: ciprod07312018
- Container Creation Time Stamp: 18.8.2018, 13:55:24
- Start Time: 18.8.2018, 13:55:24
- Finish Time: -
- CPU Limit: 150 mc
- CPU Request: -

16. By hitting the **Search logs** section, you can define your own custom Azure monitoring searches and integrate them in your custom portal:

Home > kubecluster

kubecluster
Kubernetes service

→ Move Delete Refresh

Resource group (change) AKS01	Kubernetes version 1.11.1
Status Succeeded	API server address myaks-900c6fe3.hcp.westeurope.azmk8s.io
Location West Europe	Total cores 3
Subscription (change) Visual Studio Enterprise	Total memory 10.5
Subscription ID f3248cad-78cf-4116-bee1-01151da2d51d	HTTP application routing domain 0c67c6a16d994ed4b7a5.westeurope.aksa...
Tags (change) Click here to add tags	

⌵

Monitor container health
Get health and performance insights
[Go to Azure Monitor container health](#)

Search logs
Search and analyze logs using ad-hoc queries
[Go to Azure Monitor logs](#)

View Kubernetes dashboard
Learn how to connect to the Kubernetes dashboard
[View connection steps](#)

To finally get everything up-and-running, the following to-do list gives a brief overview of all the tasks needed to provide an app within AKS:

1. **Prepare the AKS App:** <https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-prepare-app>
2. **Create the container registry:** <https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-prepare-acr>
3. **Create the Kubernetes cluster:** <https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-deploy-cluster>
4. **Run the application in AKS:** <https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-deploy-application>
5. **Scale the application in AKS:** <https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-scale>
6. **Update the application in AKS:** <https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-app-update>

AKS has the following service quotas and limits:

Resource	Default limit
Max nodes per cluster	100
Max pods per node (basic networking with KubeNet)	110
Max pods per node (advanced networking with Azure CNI)	301
Max clusters per subscription	100

If you already have ACS in place and need to migrate to AKS, the following URL should help: <https://docs.microsoft.com/en-us/azure/aks/acs-aks-migration>.

As you have seen, AKS in Azure provides great features with a minimum of administrative tasks.

Summary

In this chapter, we have talked about all the basics that are required to understand, deploy, and manage container services in a public cloud environment, including the following:

- The concept of containers
- Container registries
- Container environments
- Container orchestrators (including AKS)

Basically, the concept of containers is a great idea and surely the next step in virtualization that applications need to go to. Setting up the environment manually is quite complex, but by using the PaaS approach, the setup procedure is quite simple (because of automation) and allows you to just start using it.

Questions

1. What is the difference between a virtual machine and a container?
2. Describe the concept of microservices and why you should use them, in 2-3 sentences.
3. What is a container registry?
4. Which operating systems can support containers?
5. What is a container orchestrator?
6. Which three container orchestrators do you know?
7. Which kinds of application are not designed to run in a container?

Further reading

Additional resources can be found here at the following links:

- **Docker for Serverless Applications:** <https://www.packtpub.com/virtualization-and-cloud/docker-serverless-applications>
- **Dockerization - Do more with Docker [Integrated Course]:** <https://www.packtpub.com/virtualization-and-cloud/dockerization-do-more-docker-integrated-course>
- **Getting Started with Kubernetes:** <https://www.packtpub.com/virtualization-and-cloud/getting-started-kubernetes>
- **Azure Kubernetes Service (AKS):** <https://docs.microsoft.com/en-us/azure/aks/>

9

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 (PDC)** 2008. Unlike a traditional VM with a hosted IIS, Azure Cloud Services is a **Platform as a Service (PaaS)** offer from Azure. The successor of Azure Cloud Services is **Azure App Service**, which is covered in detail in Chapter 15, *Implementing Azure App Services*.

What does a service like Azure Cloud Services that has been on offer for so long look like? This question is precisely what I will pursue now and we will examine Azure Cloud Services in detail.

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

Technical requirements

I used Visual Studio 2017 Community Edition for my development of our Cloud Service, but you can also use any other edition (Professional/Enterprise) of Visual Studio for this.

Also Visual Studio code (a free and platform independent version of visual studio) is suitable for this purpose but has fewer support for Azure Cloud Services. To prevent unnecessary expense, you can also use the free Visual Studio Community Edition as I did, which is also sufficient for our purposes.

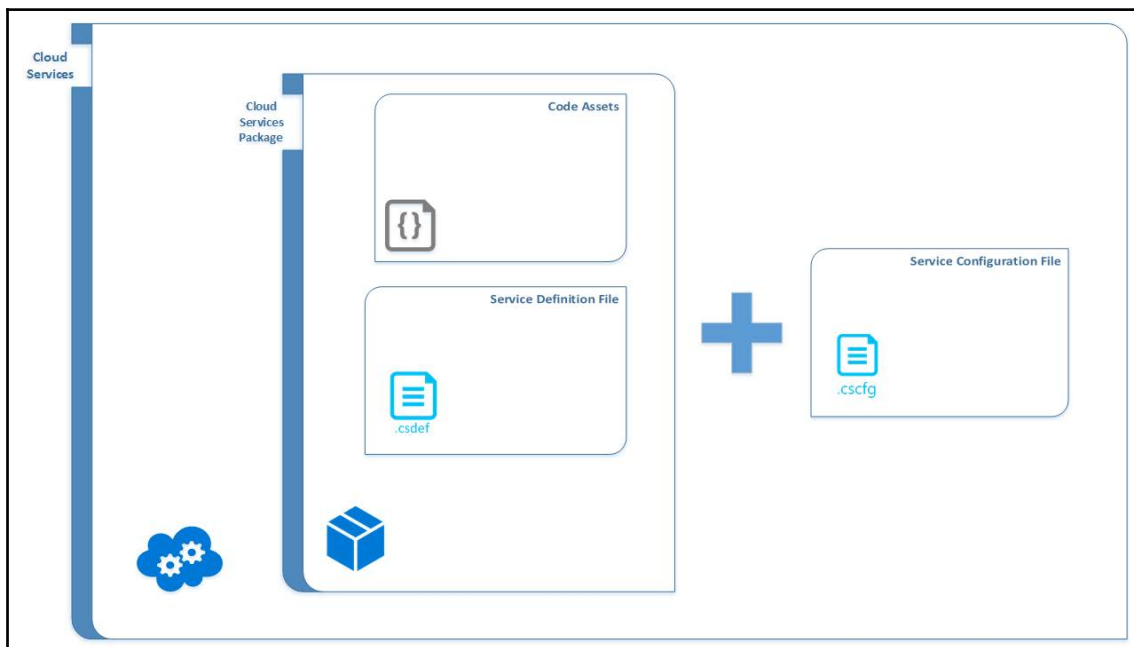
What is an Azure Cloud Service?

An Azure Cloud Service is a highly available, scalable, and multi-layered web app hosted on a Windows VM with an installed IIS.

Although an Azure Cloud Service is hosted on VMs there is a major difference that makes the Azure Cloud Service an PaaS offering. 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 file system 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 machine data disks.

Understanding the Cloud Service architecture

The following diagram shows an brief overview of an Azure Cloud Service architecture:



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 which includes the **Service Definition File** (`ServiceDefinition.csdef`) and the **Code Assets** for the service and the required binary-based dependencies
- **Service Configuration File:** The service configuration file is `ServiceConfig.cscfg`

As you can also 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 run time (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 Azure 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 suitable role instances.

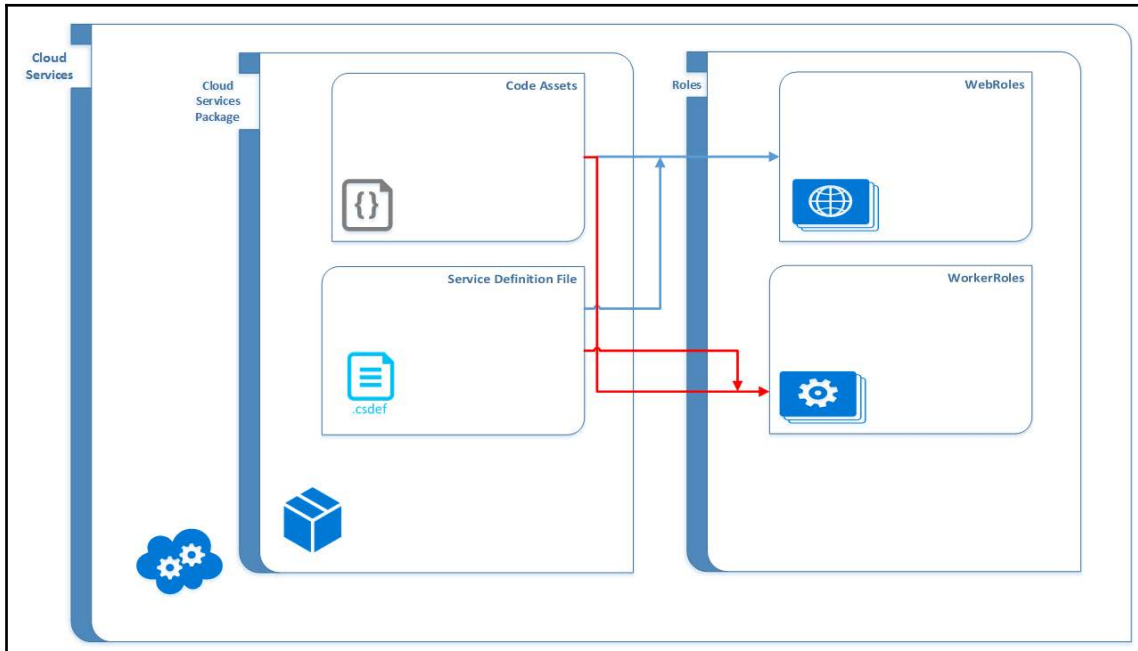
There are currently two options for the roles available:

- **WebRoles**
- **WorkerRoles**

The term *WebRoles* is referred to Cloud Service instances which are running on a Windows VM with an installed IIS and the term *WorkerRoles* refers to Cloud Service instances which are running on a Windows VM without an installed IIS.

Those roles have specific tasks and workload which we will detail in the following part.

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 to 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

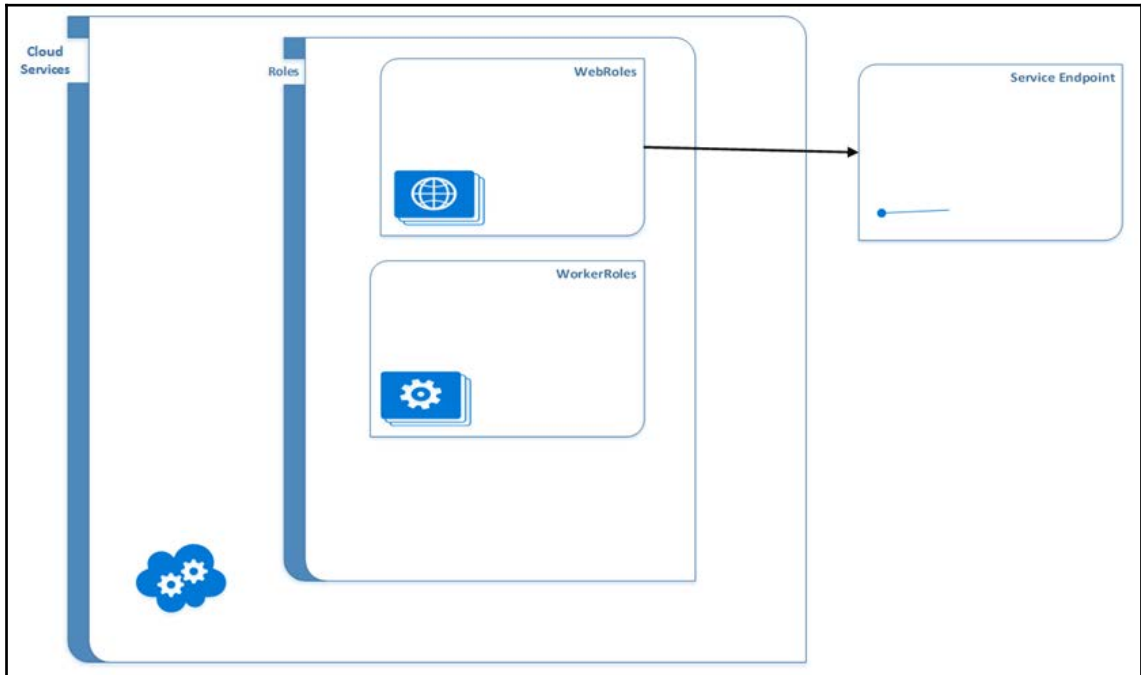
Let's go to the preceding diagram and thus to the last element of the Cloud Service architecture, the service endpoint.

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:



Like the limitations for web and worker Roles, also the endpoints are limited. The limits are instance input endpoints: 25, input endpoints: 25, internal endpoints: 25.

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, I will explain the blueprints of Cloud Services in more detail. Before the question arises, under blueprints of Cloud Services we understood:

- The service definition file (`ServiceDefinition.csdef`)
- The service configuration file (`ServiceConfig.cscfg`)

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/ServiceDefinitio
n" 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 contain at least one item to be valid.

Now we will look at the four elements in detail.

LoadBalancerProbes

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 `LoadBalancerProbes` is not a standalone element, but exists in combination with a `WebRole` or a `WorkerRole`. A `LoadBalancerProbes` may be provided for more than one role.

A template of a `LoadBalancerProbes` 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 `http` as the attribute of `protocol`. 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>
    </Runtime>
  </WebRole>
</ServiceDefinition>
```

```

        </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 I want just to cover the key elements.

The key elements are as follows:

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 <code>Endpoints</code> 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 <code>Endpoints</code> 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 <code>Endpoints</code> 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 ALB. You can define UDP or TCP endpoints.
FixedPort	Specifies the port for the internal endpoint, which enables connections to the ALB.
FixedPortRange	Specifies a range of ports for internal or instance input endpoints, which enables connections to the ALB.
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 run time environment), registering a Component Object Model (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 a `WebRoles` process. `WorkerRole` run on VM instances without installed IIS.

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>
  </WorkerRole>
</ServiceDefinition>
```

```

</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="" 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>
</WorkerRole>
</ServiceDefinition>

```

I 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. To be more specific, it can limit which roles can access the internal endpoints of the specific role.

The `NetworkTrafficRules` element is not a standalone element, but it exists in combination with a `WebRole` or a `WorkerRole`. The element `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
<code>OnlyAllowTrafficTo</code>	Contains a collection of endpoints and the roles that can communicate with them. You can specify multiple nodes of this element.
<code>Destinations</code>	Contains a collection of <code>RoleEndpoint</code> .
<code>RoleEndpoint</code>	Contains a description of an endpoint on a role and allows the communication 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 <code>Destinations</code> node.
WhenSource	Contains a collection of roles that can communicate with the endpoints defined in the <code>Destinations</code> node.
FromRole	Specifies the roles that can communicate with the endpoints defined in the <code>Destinations</code> node. You can specify multiple nodes of this element.

Service configuration file

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, there are three elements in the Azure Services Configuration Scheme. Root element (the top level element) is `<ServiceConfiguration>` with the mandatory attribute `serviceName` and the three optional attributes `osFamily`, `osVersion`, and `schemaVersion`.

The following are additional elements of the Services Configuration Scheme:

- `<Role>`: Defined in the Azure role scheme
- `<NetworkConfiguration>`: Defined in the Azure network configuration scheme

Those two elements are defined as follows:

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>
  </NetworkConfiguration>
</ServiceConfiguration>
```

```

</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 Service 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. Though it might look like the same offering with different names, both services have a unique usage.

While Azure App Service offers a more strict PaaS service, Azure Cloud Services allow the user more control over its underlying resources. 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

You can, if necessary, influence the OS edition (more specific the OS family) and the version (the update level) for the host VM of your Cloud Service.

To include this selection in your configuration, you have to make the following changes:

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 SP1	

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 for our Cloud Services deployment.

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 GBps 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, modelling, and simulations.

The version 1 instance types are as follows:

Instance	Cores	RAM	Temporary disk size
A0	1	0,75 GB	20 GB
A1	1	1,75 GB	225 GB
A2	2	3,5 GB	490 GB
A3	4	7 GB	1000 GB
A4	8	14 GB	2040 GB
A5	2	14 GB	490 GB
A6	4	28 GB	1000 GB

A7	8	56 GB	2040 GB
A8	8	56 GB	1817 GB
A9	16	112 GB	1817 GB
A10	8	56 GB	1817 GB
A11	16	112 GB	1817 GB

Version 2 has the following instances:

Instance	Cores	RAM	Temporary disk size
A1_v2	1	2 GB	10 GB
A2_v2	2	4 GB	20 GB
A4_v2	4	8 GB	40 GB
A8_v2	8	16 GB	80 GB
A2m_v2	2	16 GB	20 GB
A4m_v2	4	32 GB	40 GB
A8m_v2	8	64 GB	80 GB

series D

Strictly speaking, **series D** not only one series, but is divided into series D, Dv2 (version 2) and Dv3 (version 3).

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 Dv3 series is based on a latest generation CPU (Intel Broadwell E5-2673 V4 processor with 2.4 GHz and also Intel Haswell E5-2673 V3 processor with 2.4 GHz) and it achieves a 28% lowered cost level compared to the DV2 series. The Dv3 series is one of the first series to be running in Windows Server 2016 hosts and thereby enables nested virtualization.

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 machine and is directly attached to the host. Be aware that contents on this temporary storage will be deleted when the instance is deallocated.

Instance	Cores	RAM	Temporary disk size
D1	1	3.5 GB	50 GB
D2	2	7 GB	100 GB
D3	4	14 GB	200 GB
D4	8	28 GB	400 GB
D11	2	14 GB	100 GB
D12	4	28 GB	200 GB
D13	8	56 GB	400 GB
D14	16	112 GB	800 GB

The following table shows the available instances of the Dv2 series (version 2):

Instance	Cores	RAM	Temporary disk size
D1v2	1	3.5 GB	50 GB
D2v2	2	7 GB	100 GB
D3v2	4	14 GB	200 GB
D4v2	8	28 GB	400 GB
D5v2	16	56 GB	800 GB
D11v2	2	14 GB	100 GB
D12v2	4	28 GB	200 GB
D13v2	8	56 GB	400 GB
D14v2	16	112 GB	800 GB
D15v2	20	140 GB	1000 GB

The following table shows the available instances of the Dv3 series (version 3):

Instance	Cores	RAM	Temporary disk size
D2v3	2	8 GB	50 GB
D4v3	4	16 GB	100 GB
D8v3	8	32 GB	200 GB
D16v3	16	64 GB	400 GB
D32v3	32	128 GB	800 GB
D64v3	64	256 GB	1600 GB

series E

As the Dv3 series was reduced in terms of RAM amount, a new series, the **series E** was introduced. The series E VMs start a Ev3 (version 3) and are dedicated to a higher RAM to CPU value and therefore optimized for memory intense workloads.

Series E VMs are based on 2.3 GHz Intel Xeon E5-2673 v4 (Broadwell) CPUs and can reach 3.5 GHz with Turbo Boost 2.0.

Instance	Cores	RAM	Temporary disk size
E2v3	2	16 GB	50 GB
E4v3	4	32 GB	100 GB
E8v3	8	64 GB	200 GB
E16v3	16	128 GB	400 GB
E32v3	32	256 GB	800 GB
E64v3	64	432 GB	1600 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 up to 3.1 GHz.

Instance	Cores	RAM	Max. disc size
G1	2	28 GB	384 GB
G2	4	56 GB	768 GB
G3	8	112 GB	1536 GB
G4	16	224 GB	3072 GB
G5	32	448 GB	6144 GB



G5 instances are run in isolation on dedicated hardware that is deployed for only one customer.

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	Maximum 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.

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.

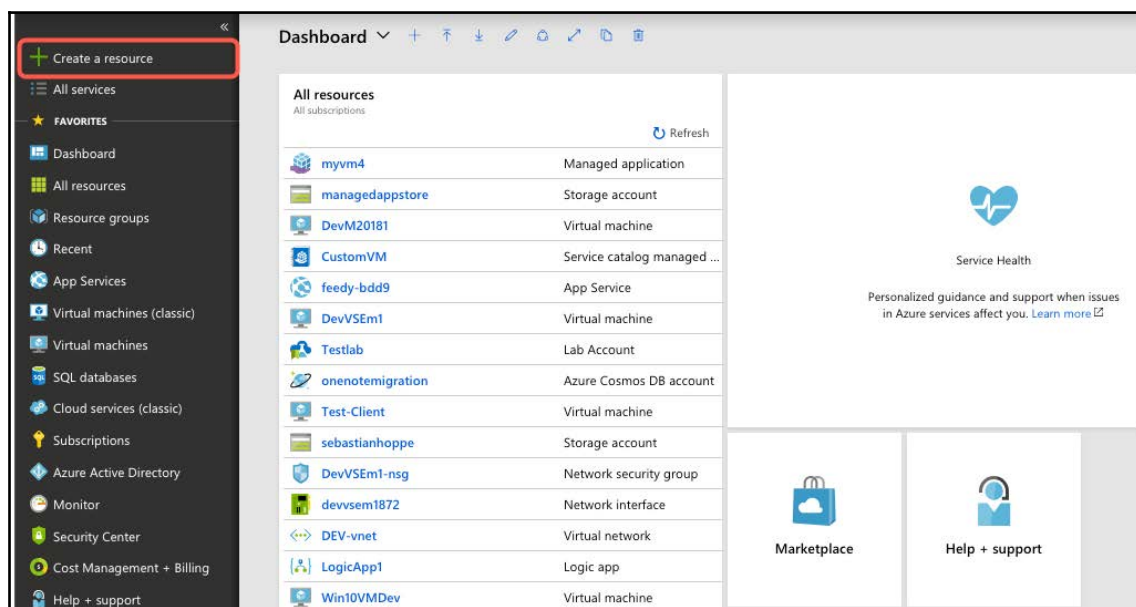
Part 1

As mentioned in the first part, we are doing the necessary preliminary work in the Azure management portal. 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.

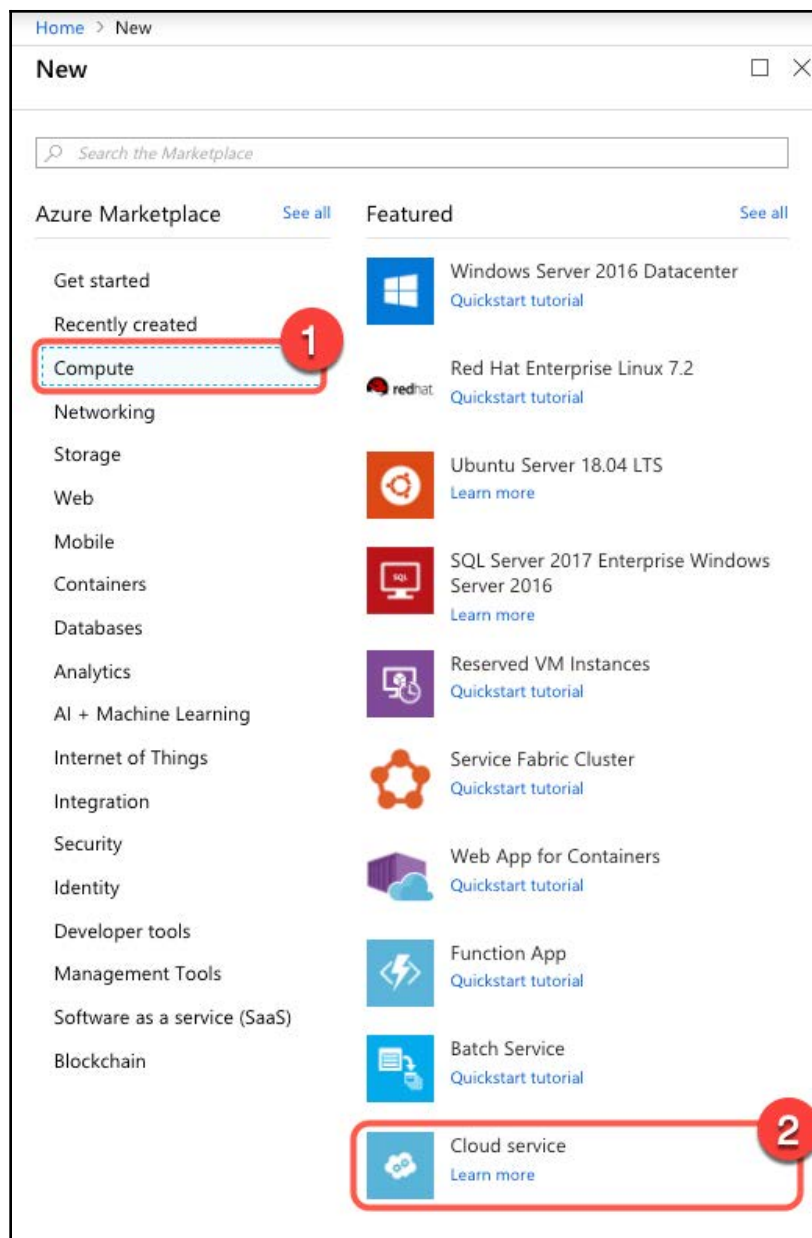
We are now going deeper into the hosting of the container. I will show you how to create a hosting container and how you can work with it.

Let's start:

1. Open your Azure management portal at <https://portal.azure.com>.
2. In the portal, click on **+Create a resource** as shown in the following screenshot:



3. Now click on **Compute**, and then click **Cloud Service**:



4. Now the **Cloud services (classic)** window will appear as shown in the following screenshot:

The screenshot shows the 'Cloud service (classic)' creation window. The breadcrumb navigation at the top reads 'Home > New > Cloud service (classic)'. The window title is 'Cloud service (classic)' with a close button. The form contains the following fields and options:

- DNS name:** A text input field with the placeholder 'Enter the name' and a '.cloudapp.net' suffix.
- Subscription:** A dropdown menu currently showing 'MSDN Platforms'.
- Resource group:** A dropdown menu with 'Select existing...' and a 'Create new' link below it.
- Location:** A dropdown menu currently showing 'West Europe'.
- Package:** A section with the text '(Optional) Select a package' and a right-pointing arrow.
- Certificates:** A section with the text '(Optional) Add certificates' and a lock icon.

At the bottom, there is a blue 'Create' button and a link for 'Automation options'.

5. 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:

Home > New > Cloud service (classic)

Cloud service (classic)

* DNS name
acdppbookcsse ✓
.cloudapp.net

* Subscription
MSDN Platforms

* Resource group
Select existing...
[Create new](#)

* Location
West Europe

Package
(Optional) Select a package >

Certificates
(Optional) Add certificates 🔒

[Create](#) [Automation options](#)

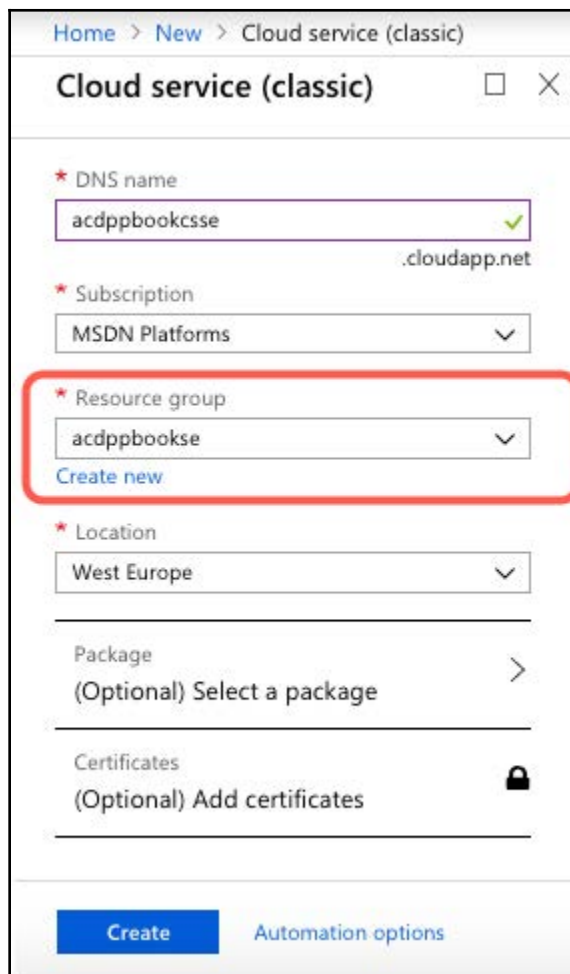
6. Next, choose a **Subscription** (or use the default subscription) as shown in the following screenshot:

The screenshot shows the 'Cloud service (classic)' creation page in the Azure portal. The breadcrumb navigation at the top reads 'Home > New > Cloud service (classic)'. The page title is 'Cloud service (classic)'. The form contains the following fields:

- DNS name:** A text input field containing 'acdppbookcsse' with a green checkmark icon to its right.
- Subscription:** A dropdown menu containing 'MSDN Platforms'. This field is highlighted with a red rectangular box.
- Resource group:** A dropdown menu with 'Select existing...' and a 'Create new' link below it.
- Location:** A dropdown menu containing 'West Europe'.
- Package:** A section with the text '(Optional) Select a package' and a right-pointing chevron icon.
- Certificates:** A section with the text '(Optional) Add certificates' and a lock icon.

At the bottom of the form, there is a blue 'Create' button and a link for 'Automation options'.

7. In the **Resource group** section, select `acdppbookse` from the drop-down list (you can find information about creating this resource group in [Chapter 2, Azure Resource Manager and Tools](#)):



The screenshot shows the 'Cloud service (classic)' creation page in the Azure portal. The breadcrumb navigation at the top reads 'Home > New > Cloud service (classic)'. The page title is 'Cloud service (classic)' with a close button. The form contains several fields:

- DNS name:** A text box containing 'acdppbookcsse' with a green checkmark and '.cloudapp.net' suffix.
- Subscription:** A dropdown menu showing 'MSDN Platforms'.
- Resource group:** A dropdown menu showing 'acdppbookse', which is highlighted with a red rectangular box. Below the dropdown is a 'Create new' link.
- Location:** A dropdown menu showing 'West Europe'.
- Package:** A section with the text '(Optional) Select a package' and a right-pointing arrow.
- Certificates:** A section with the text '(Optional) Add certificates' and a lock icon.

At the bottom of the form, there is a blue 'Create' button and a link for 'Automation options'.

8. In the **Location** list, click on the drop-down list and select the same location you have been using for the Azure resource group:

The screenshot shows the 'Cloud service (classic)' creation window. The 'Location' dropdown is expanded, showing a list of regions. A red circle with the number '1' highlights the dropdown arrow, and a red circle with the number '2' highlights the 'West Europe' option in the list.

Home > New > Cloud service (classic)

Cloud service (classic) □ ×

* DNS name
acdppbookcsse ✓
.cloudapp.net

* Subscription
MSDN Platforms ▾

* Resource group
acdppbookse ▾
[Create new](#)

* Location
West Europe ^
Australia East
Australia Southeast
Brazil South
UK West
West Central US
West Europe

Create Automation options

9. 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

Since both settings are not relevant to our description, we will not be using these options for this tutorial:

Home > New > Cloud service (classic)

Cloud service (classic)

* DNS name
acdppbookcsse ✓
.cloudapp.net

* Subscription
MSDN Platforms

* Resource group
acdppbookse
[Create new](#)

* Location
West Europe

Package
(Optional) Select a package >

Certificates
(Optional) Add certificates 🔒

[Create](#) [Automation options](#)

10. Finally, press the **Create** button as shown in the following screenshot:

Home > New > Cloud service (classic)

Cloud service (classic)

* DNS name
acdppbookcsse ✓
.cloudapp.net

* Subscription
MSDN Platforms

* Resource group
acdppbookse
[Create new](#)

* Location
West Europe

Package
(Optional) Select a package >

Certificates
(Optional) Add certificates 🔒

Create Automation options

11. A few minutes later you will see the result in the **Cloud services (classic)** blade. For moving forward, press the **NAME** field of your new service as shown in the following screenshot:

Home > Cloud services (classic)

Cloud services (classic)

Standardverzeichnis

+ Add Edit columns Refresh Assign tags Start production Stop production Start staging Stop staging Swap More

Subscriptions: All 4 selected – Don't see a subscription? [Open Directory](#) + [Subscription settings](#)

Filter by name... All subscriptions All resource groups All locations All tags No grouping

1 items

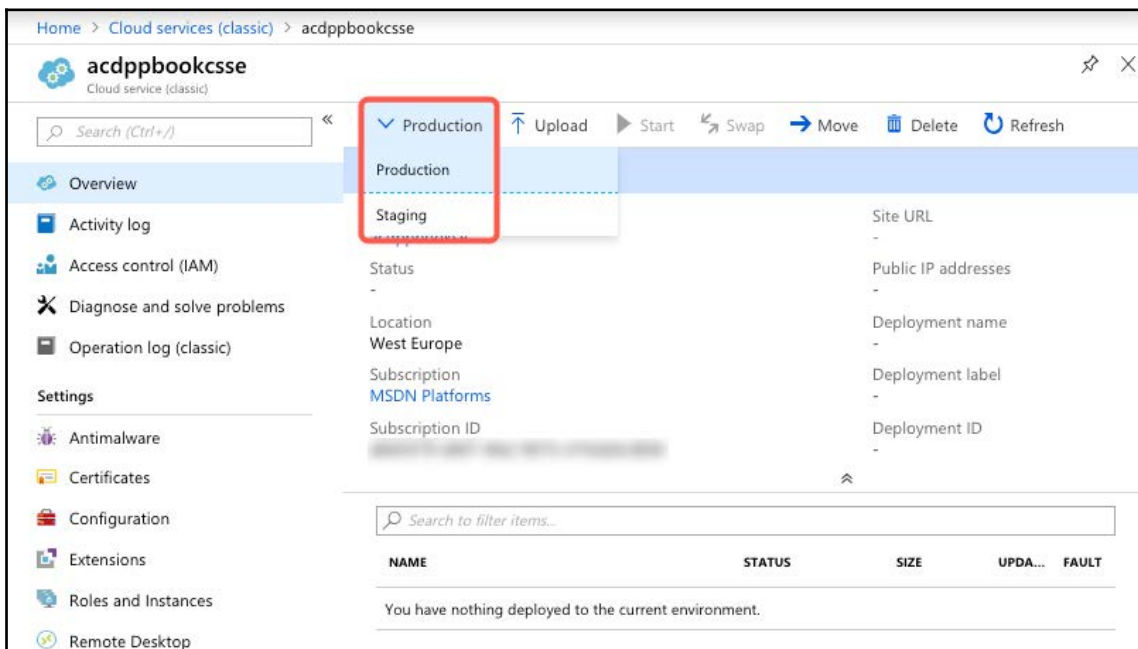
	NAME	RESOURCE GROUP	LOCATION	SUBSCRIPTION
☒	acdppbookcsse	acdppbookse	West Europe	MSDN Platforms

Now the dashboard of your Cloud Service will open. Here we will briefly review the navigation area at the top of the dashboard:

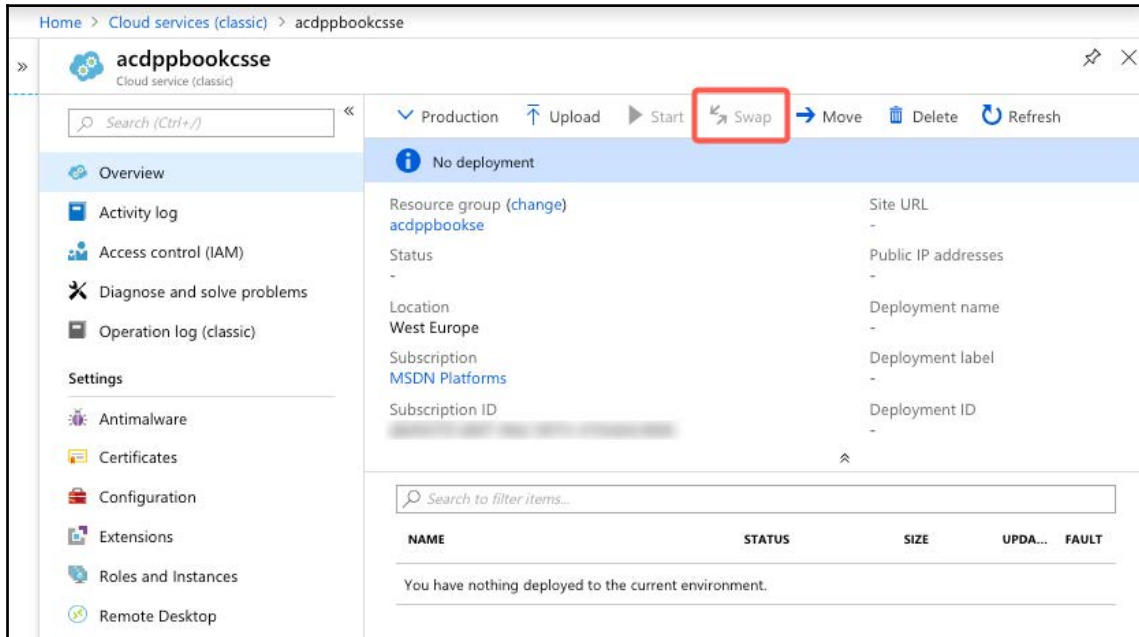
12. The first part of the review concerns the so-called **slots**. Microsoft Azure provides two slots for your later deployments:

- **Production**
- **Staging** (for testing or development)

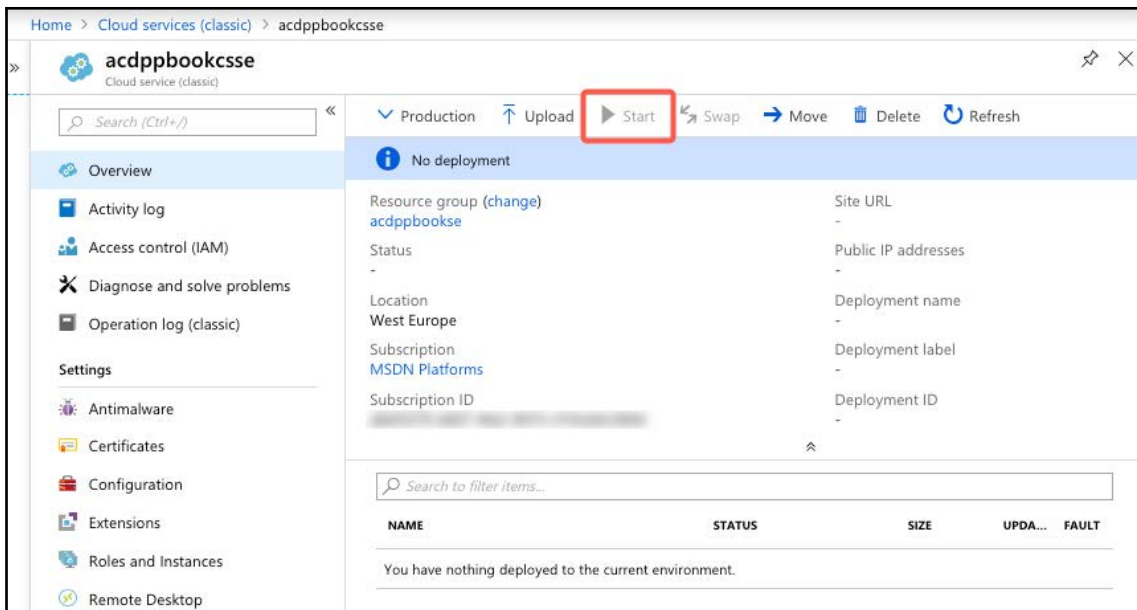
So you have access to these slots, you will find a fold-out list in the navigation bar:



13. As soon as you have completed your tests, and have made the necessary adjustments in the settings, you can use the **Swap** button to move the deployment from the **Staging** slot into the **Production** slot:



14. 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:

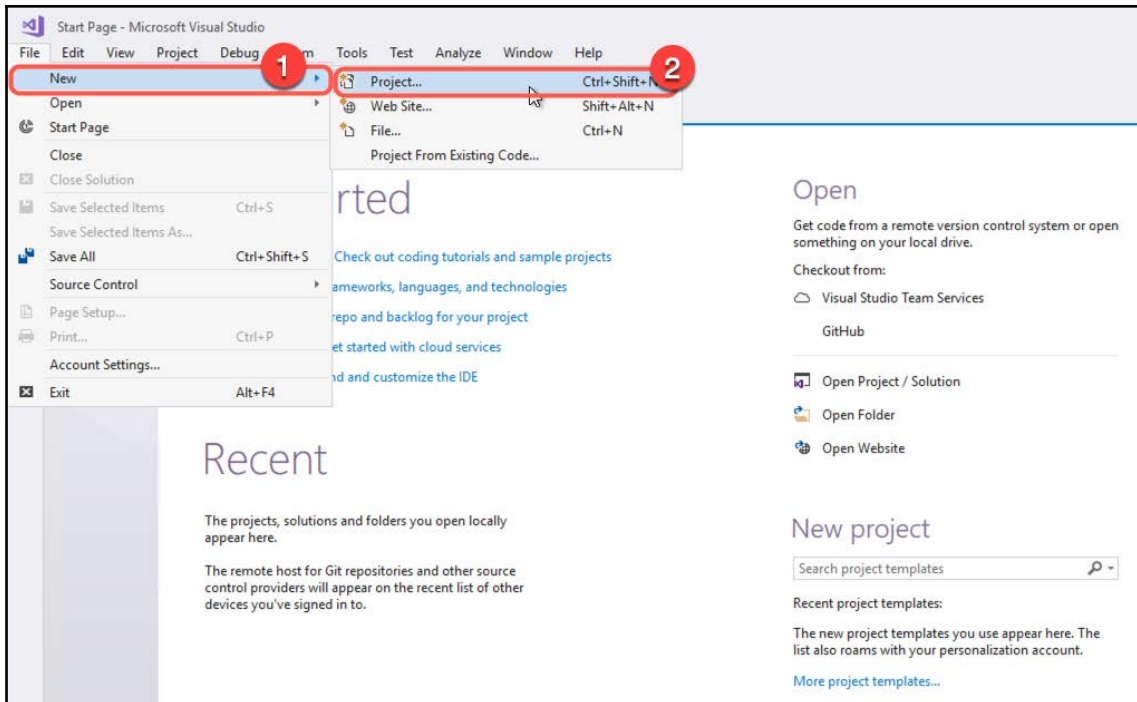


Part 2

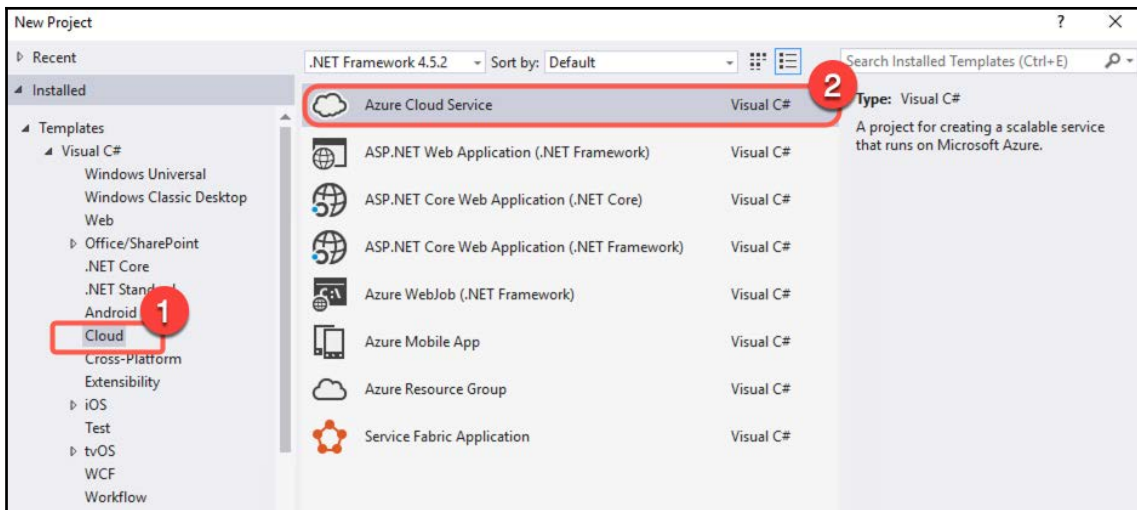
Some important notes—we will now create the Cloud Service implementation as an example workflow in Visual Studio, but depending on your desired programming language, you can also use Eclipse or IntelliJ IDEA.

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...** as shown in the following screenshot:

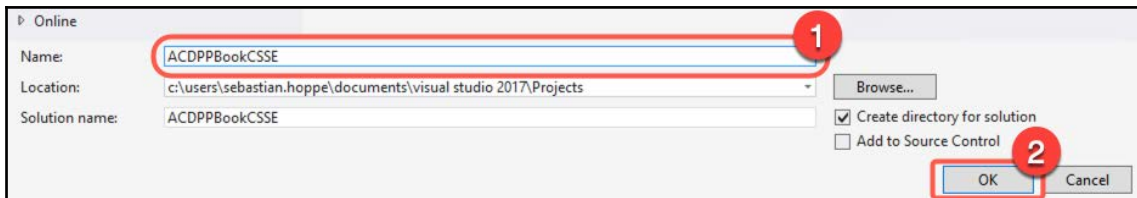


2. Now open the selection dialog with the available project templates. The required template **Azure Cloud Service**, can be found in the **Cloud** area:

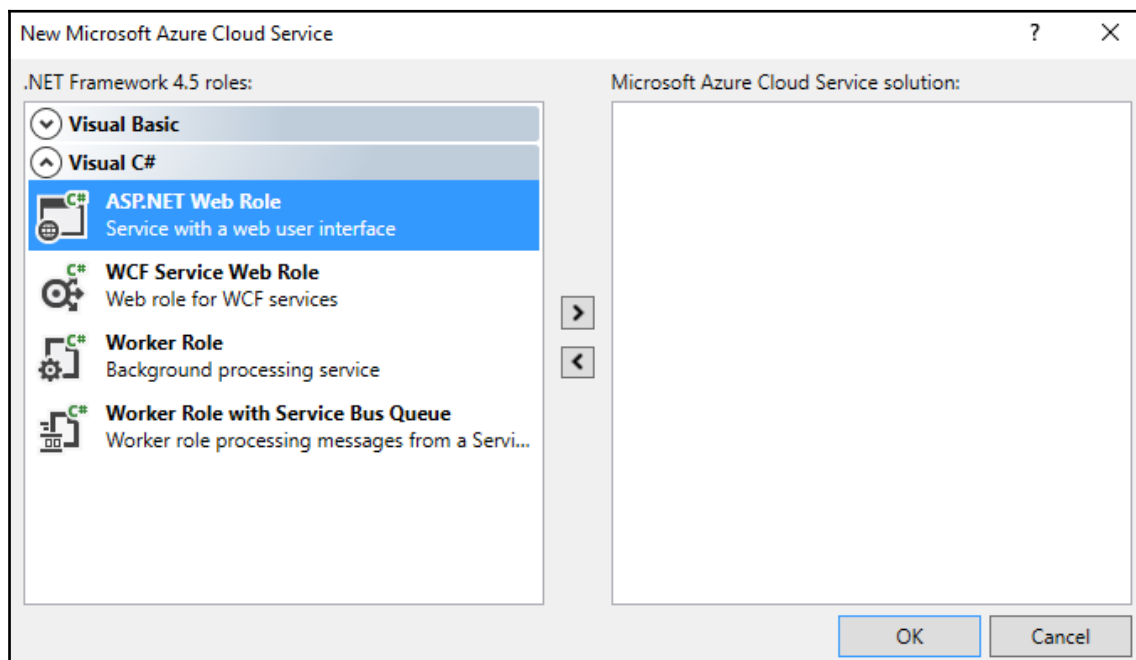
**Attention!**

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, `ACDPPBookCSSE`) and press the **OK** button:

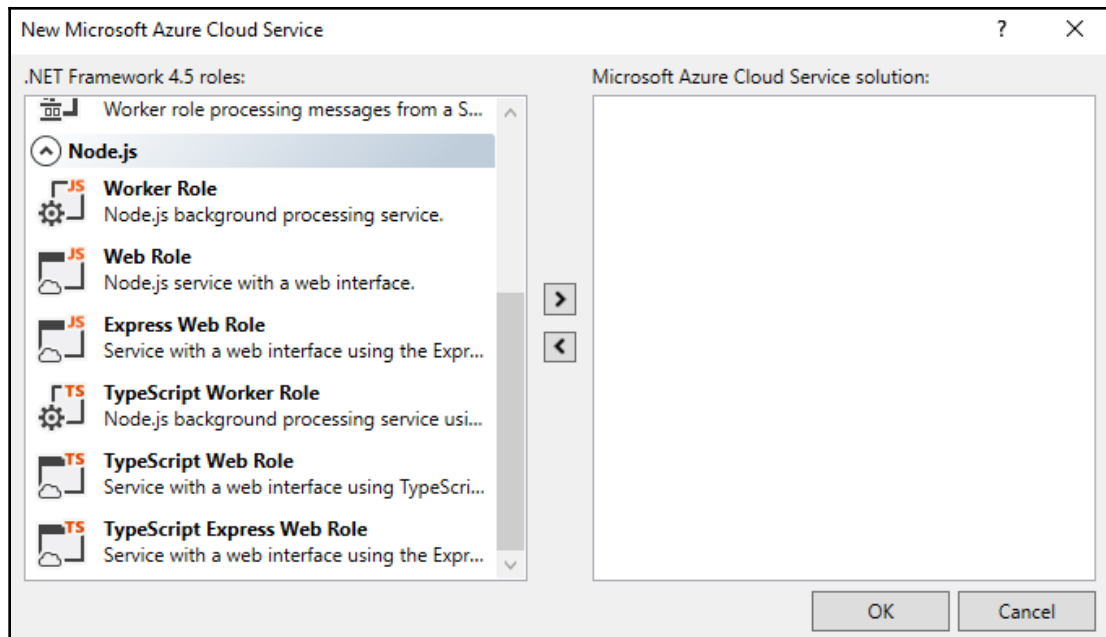


4. After this step, a selection dialog will pop up, showing a list of role templates:

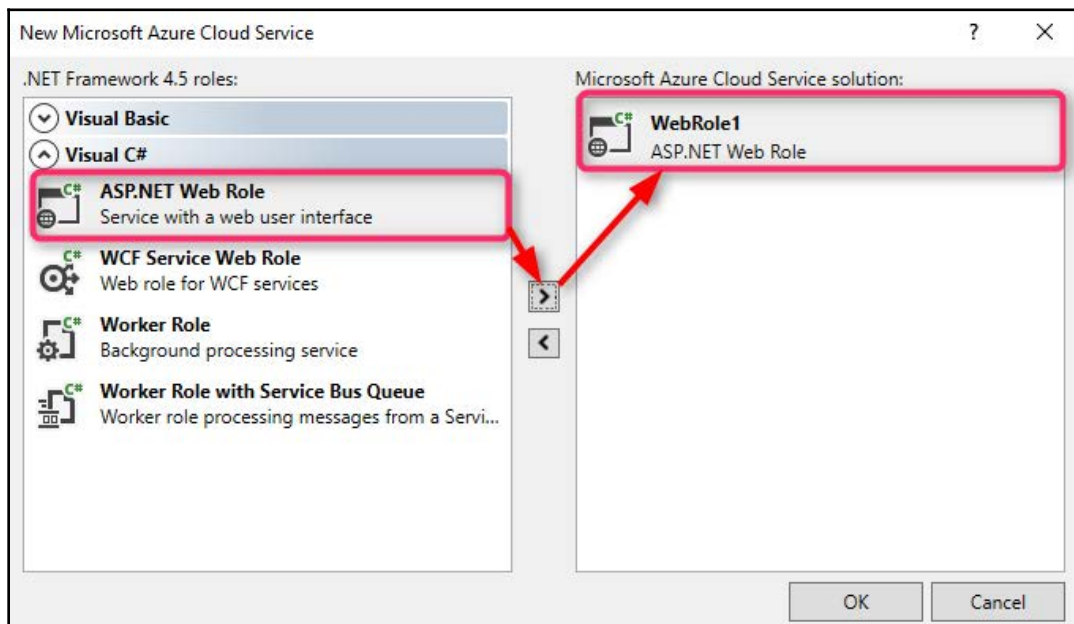


Templates are available for all supported programming languages. Depending on the edition or the chosen installation of Visual Studio, this can also be a more extensive selection.

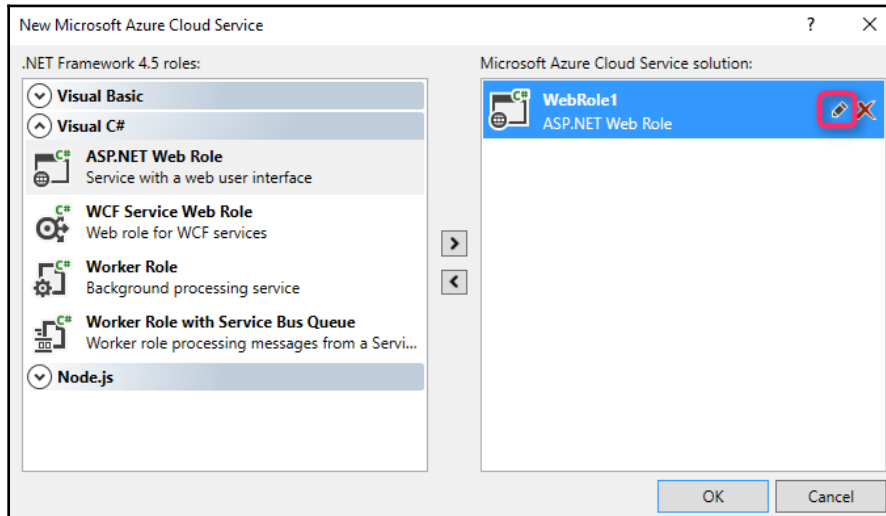
For example, in the following screenshot, you can also see templates for Node.js, Node.js (Express), and TypeScript:



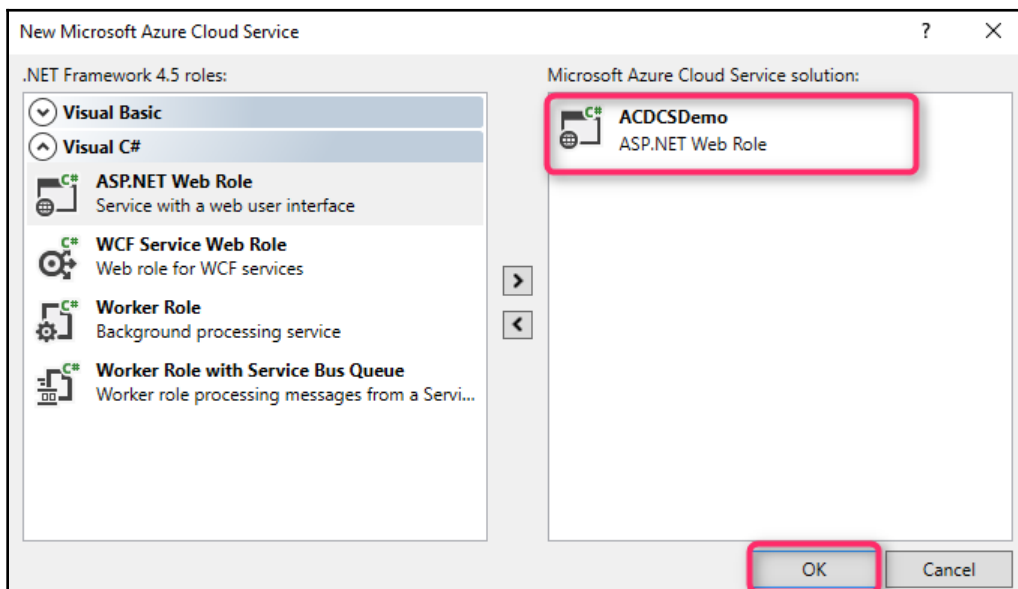
5. For our demo, we will use an **ASP.NET WebRole**. Please select the appropriate role and then press the arrow button as shown in the following screenshot:



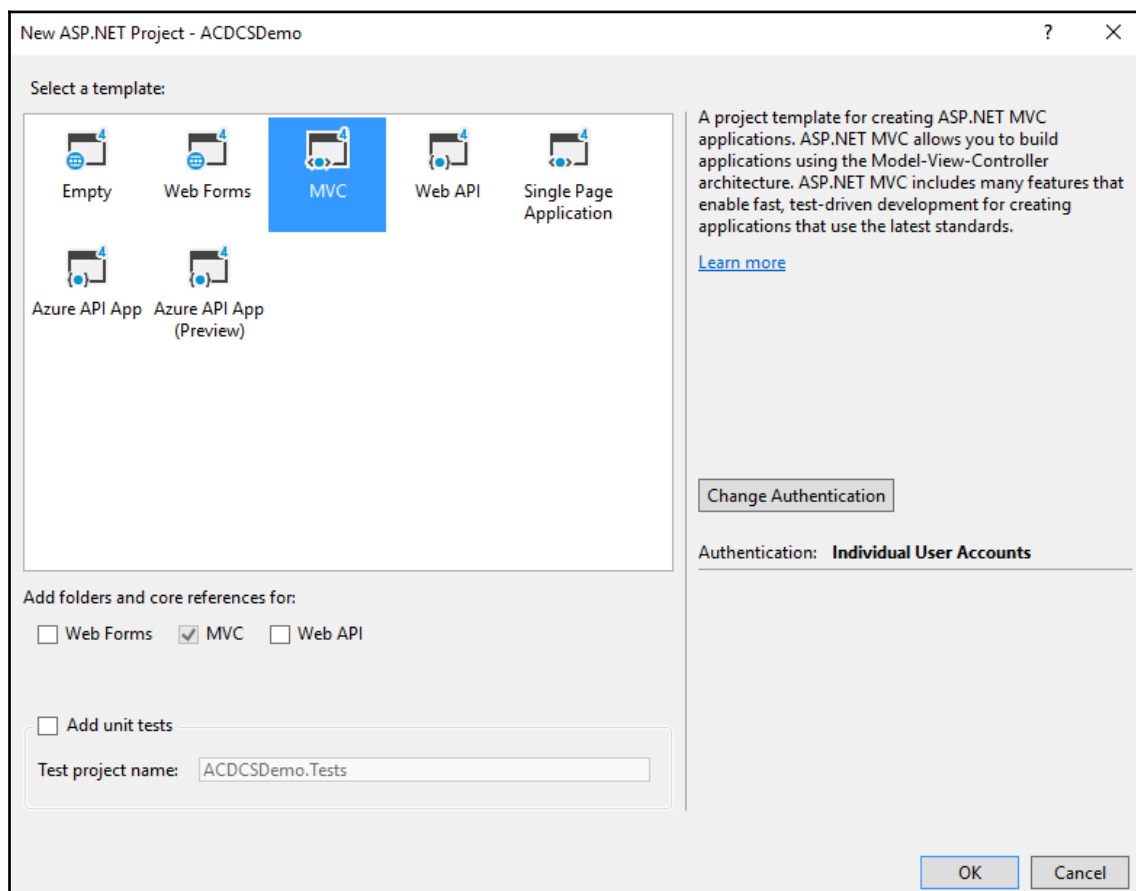
6. **WebRole1** is not a meaningful name. Of course, we want to change that, so please click on the Edit button (button with the pencil icon):



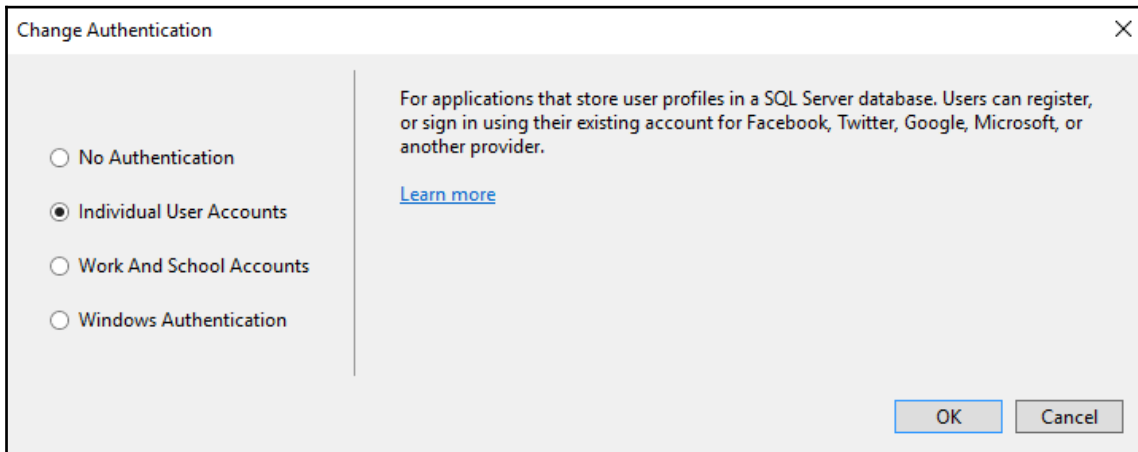
7. Now write the role name of your choice (for example, **ACDCSDemo**) and press the **OK** button:



8. 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:

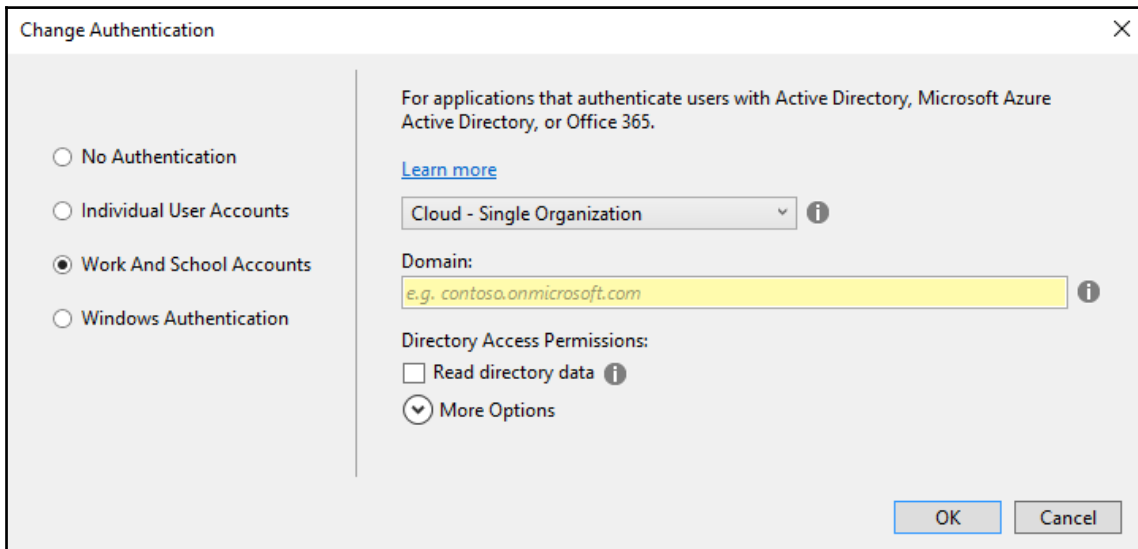


9. If necessary, you can now also change the authentication procedure (simply press the **Change Authentication** button). Since this is not relevant for our demo, I will not pursue it. You can change the authentication to either individual accounts:



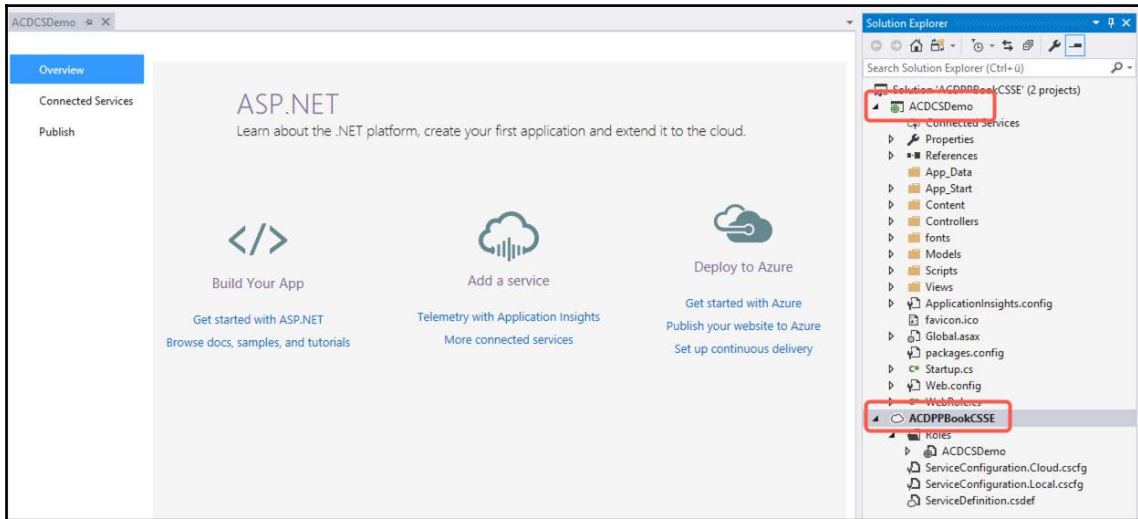
The 'Change Authentication' dialog box is shown. On the left, there are four radio button options: 'No Authentication', 'Individual User Accounts' (which is selected), 'Work And School Accounts', and 'Windows Authentication'. On the right, there is explanatory text: 'For applications that store user 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.

10. Authentication based on an Azure AD as follows:



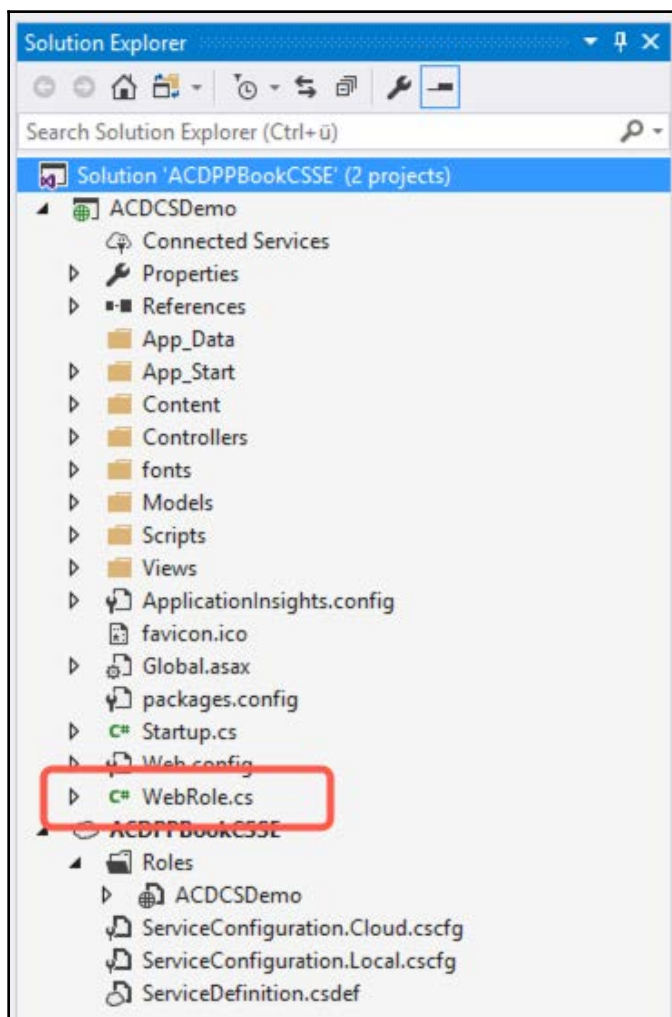
The 'Change Authentication' dialog box is shown. On the left, the 'Work And School Accounts' radio button is selected. On the right, the text reads: 'For applications that authenticate users with Active Directory, Microsoft Azure Active Directory, or Office 365.' Below this is a 'Learn more' link. A 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 are two options: 'Read directory data' (unchecked) and 'More Options' (checked with a dropdown arrow). 'OK' and 'Cancel' buttons are at the bottom right.

11. To continue, press the **OK** button in the project type selection dialog. Then your project solution is created, according to your requirements.
12. 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

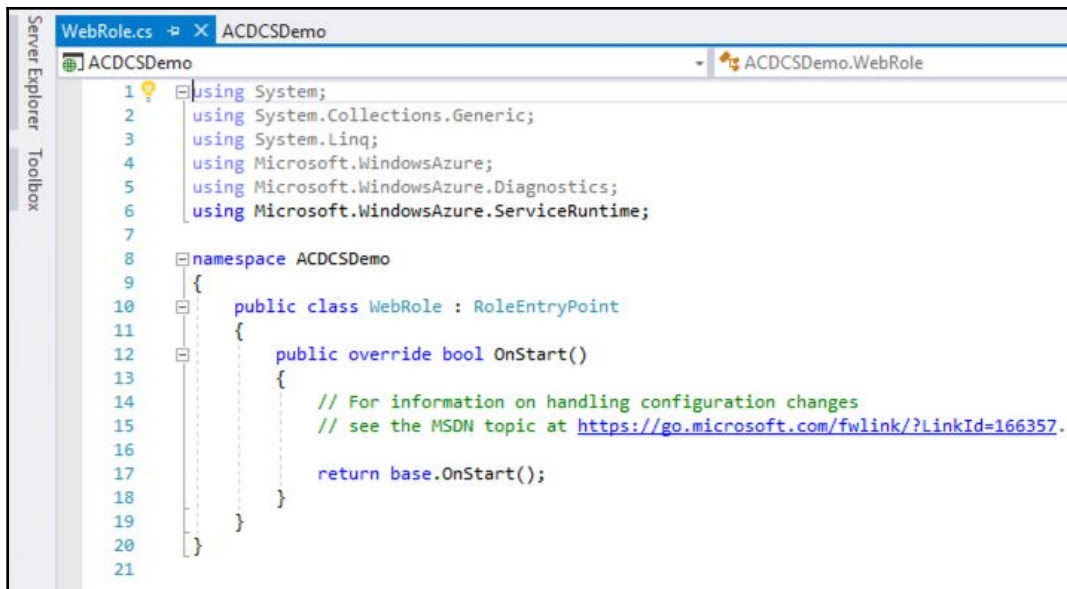


13. Next we will look at the most important components of the solution.

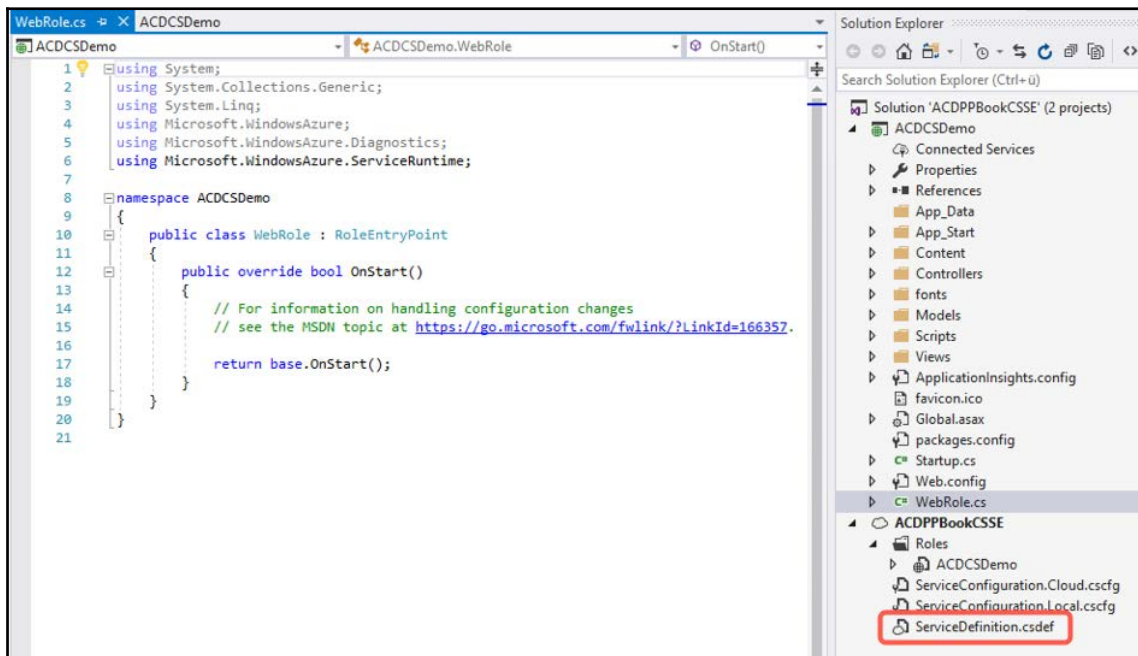
14. In the web project, you will find the file `WebRole.cs`. This is the position where you implement the program logic of your role:



15. A double-click on the filename opens a template, which you can edit according to your needs:



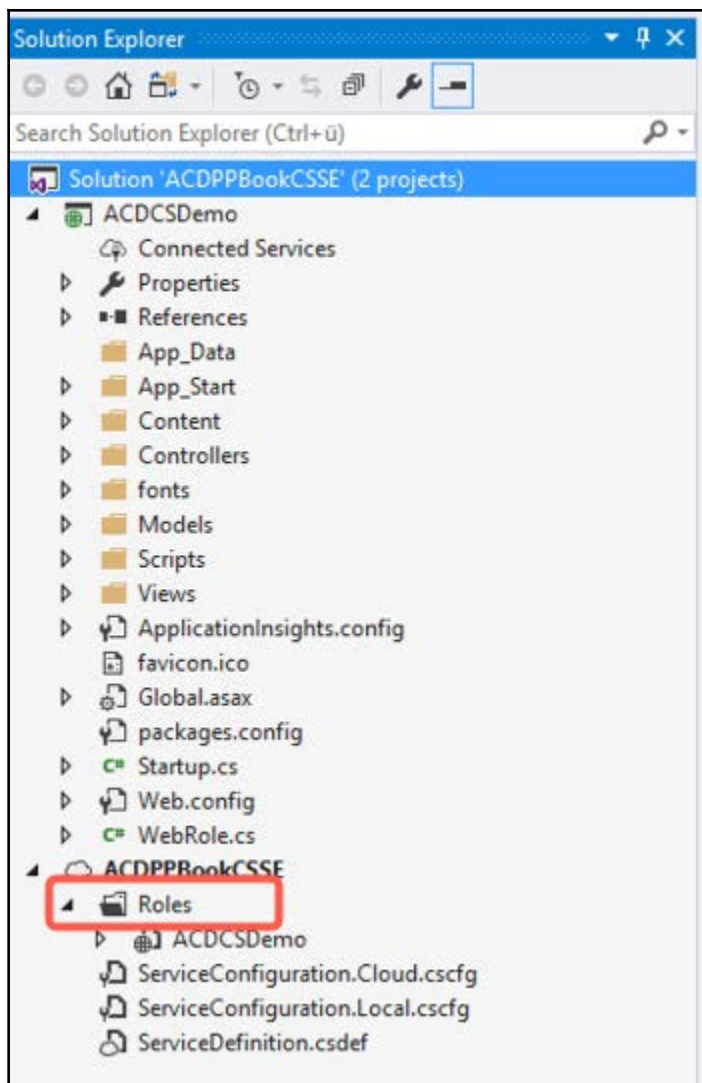
16. In the role project, you will find the templates for the Service Definition File, and the Service Configuration File:



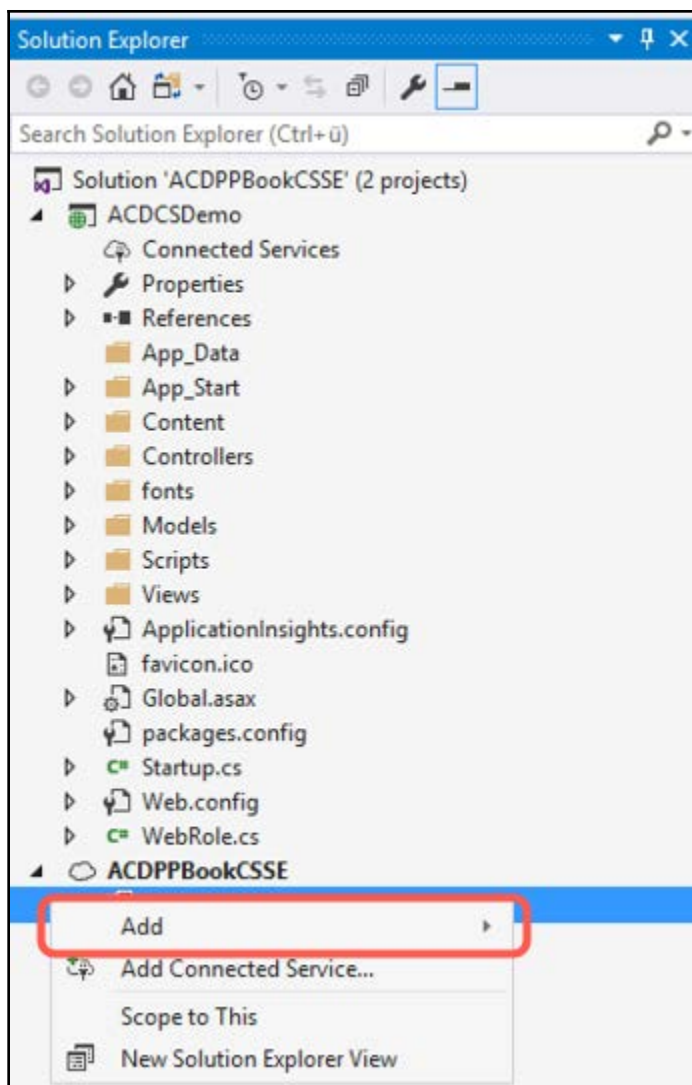
17. The Service Configuration File is available in two variations (Cloud and Local). This allows you to separate configuration settings in the local development cycle:



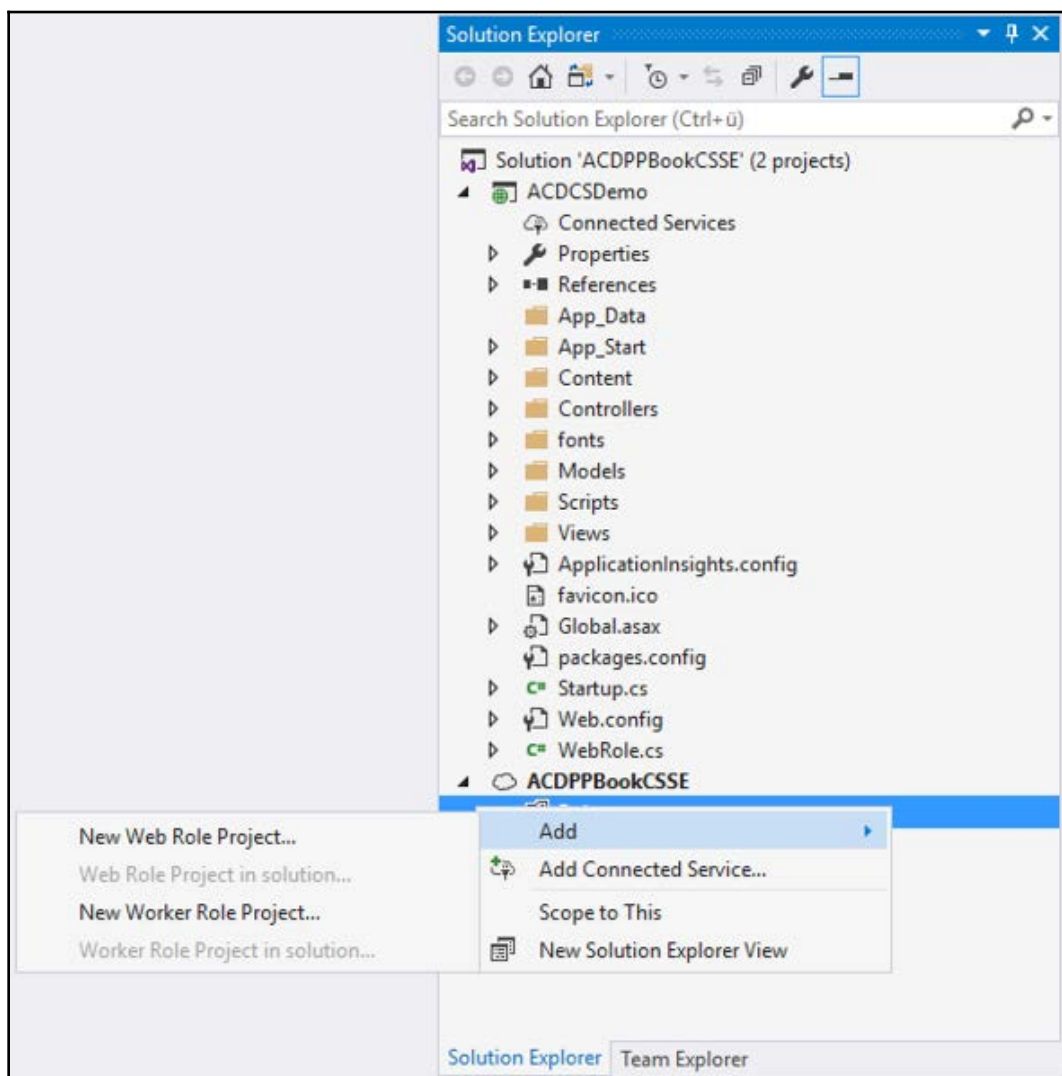
18. In our demo, we have defined only one role. However, a typical Cloud Service solution usually consists of several roles. To define more roles, right-click on the field marked in the following screenshot:



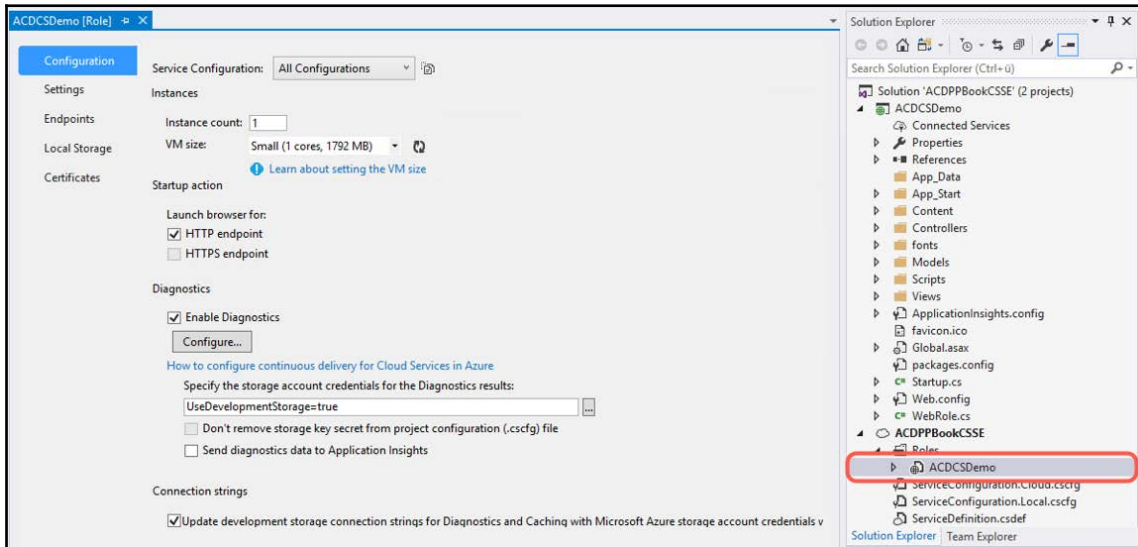
19. Select the **Add** option as shown in the following screenshot:



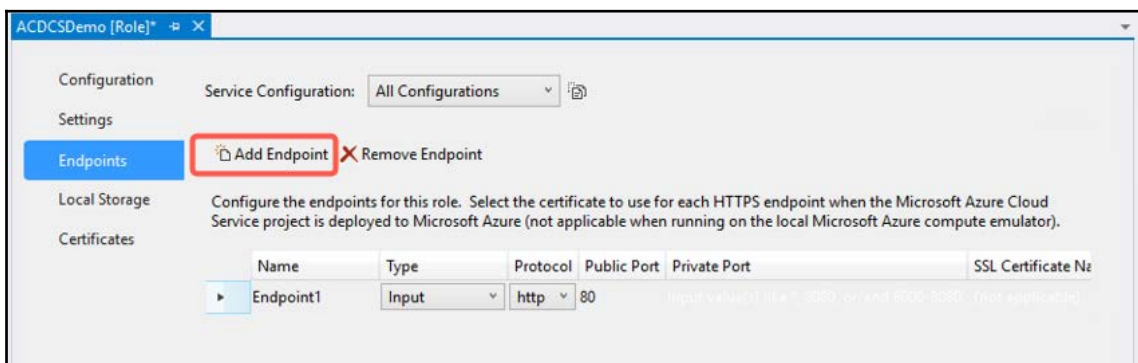
20. Now you can add another role to your solution with the **New Web Role Project...** or **New Worker Role Project...** options:



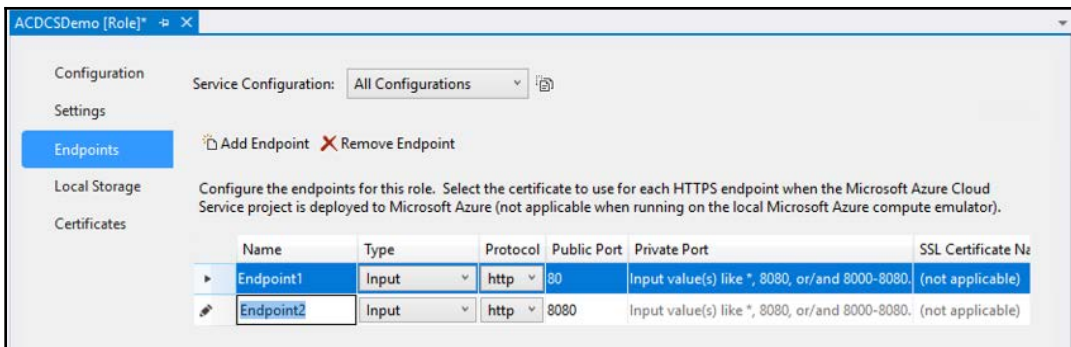
21. Visual Studio also offers opportunities to edit the Service Definition and the Service Configuration File and not only manually. If you right-click on the field marked in the following screenshot, and then click **Properties**, a dialog box with numerous possibilities for individual settings will appear. All the settings you define here are immediately applied to the solution:



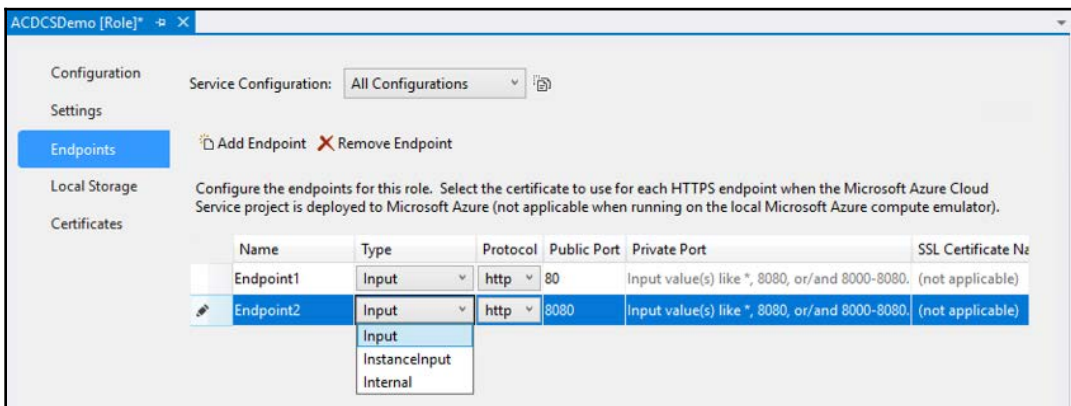
22. For example, open the **Endpoints** page and then press the **Add Endpoint** button as shown in the following screenshot:



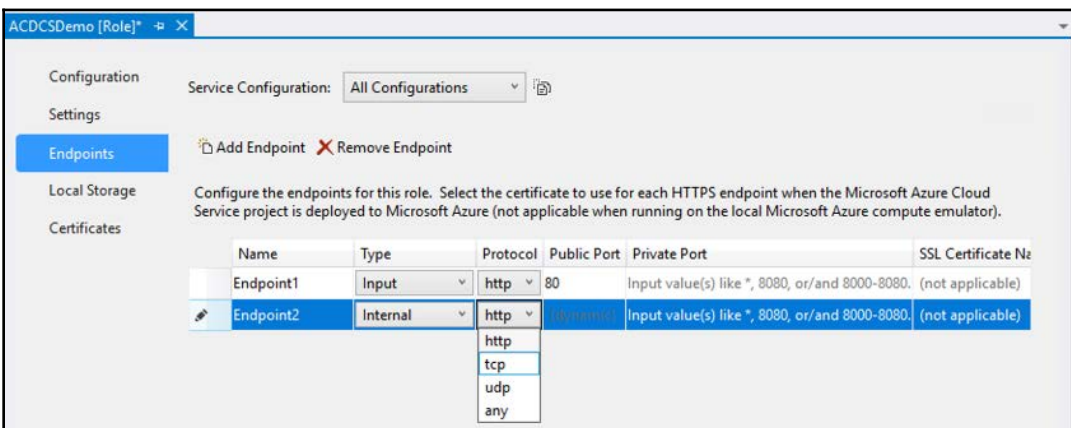
23. Immediately a new endpoint is available:



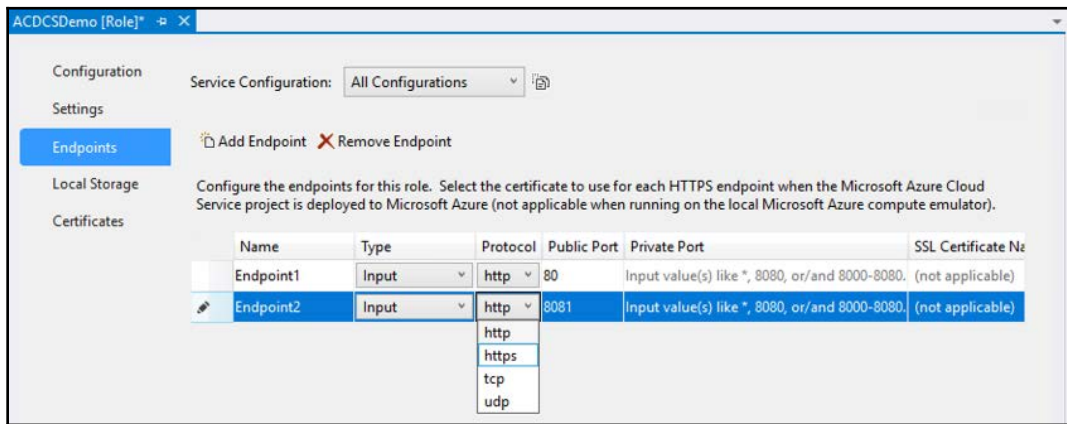
24. Next, select an endpoint type:



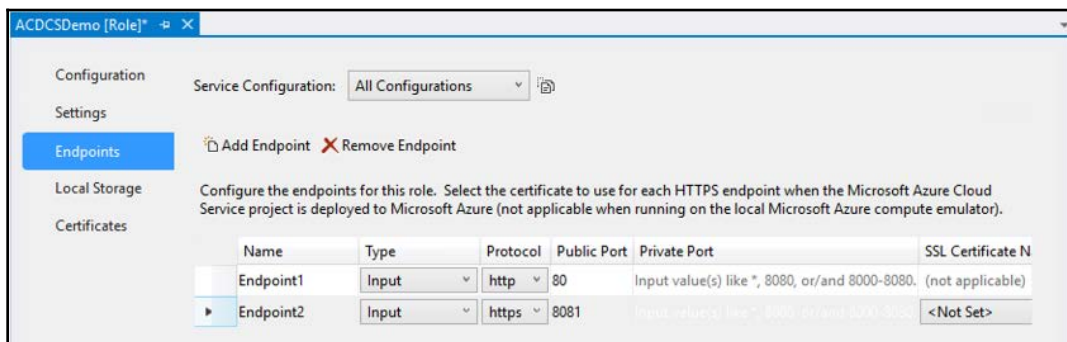
25. And then select a protocol:



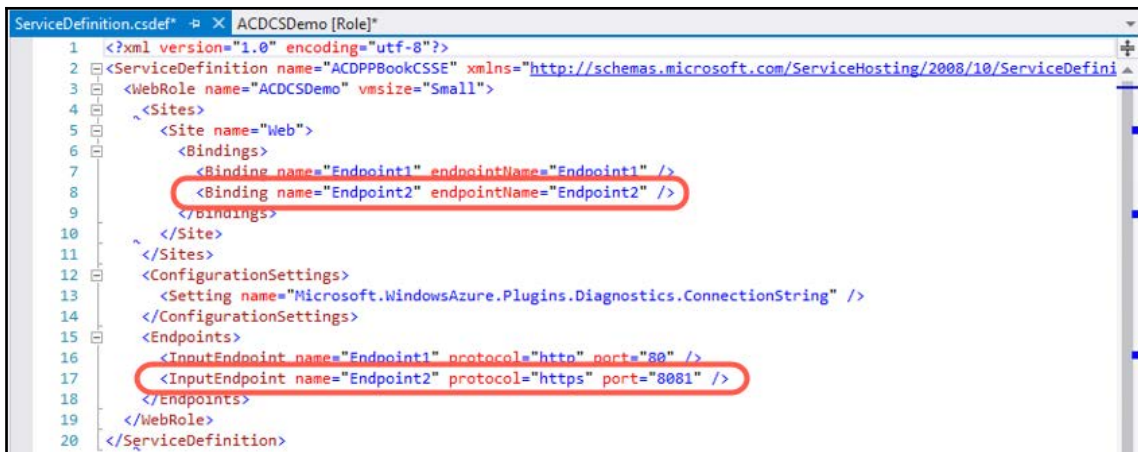
26. As you can see, the range of protocols varies with the different endpoint types:



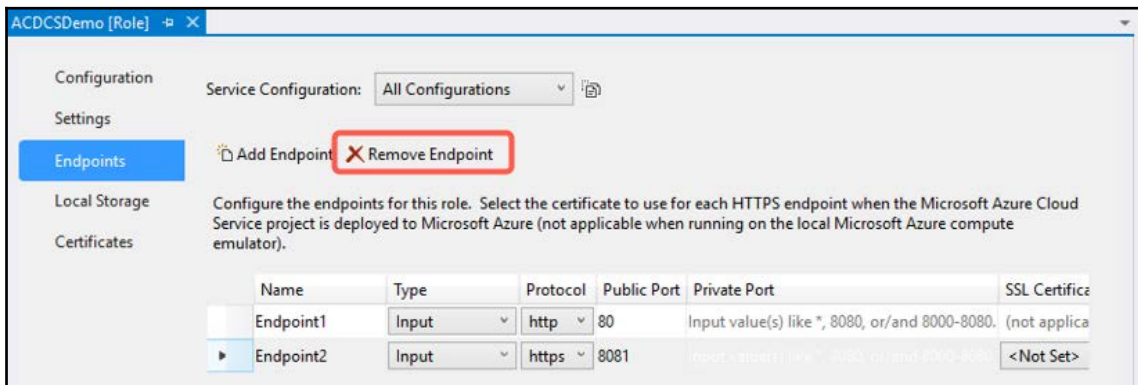
27. Ready, a new endpoint is defined:



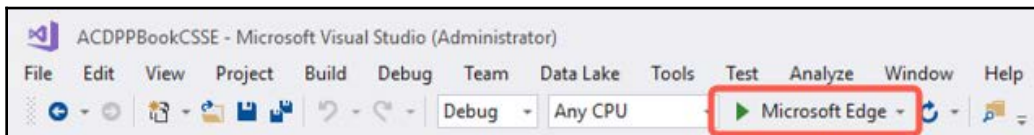
28. If you open the `ServiceDefinition.csdef` file prior to removing the second endpoint, you will see that the changes were automatically adopted. The file should now look as follows:



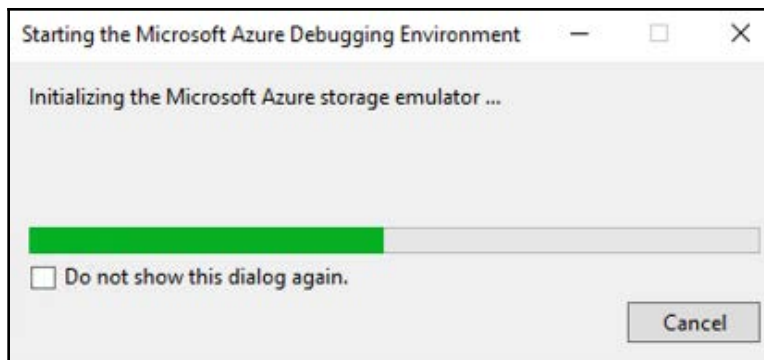
29. Remove the endpoint prior to debugging our application by simply clicking on **Remove Endpoint** while **Endpoint 2** is selected. Otherwise you might get a site showing a certificate error, which we do not want!



30. The next step is the local test run. Please click on the Run button marked in the following screenshot or press the *F5* key:



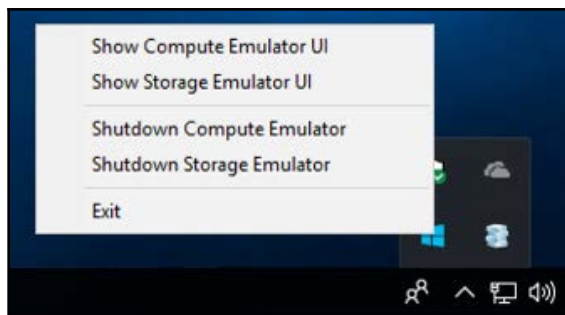
31. Now the **Microsoft Azure Debugging Environment** is starting up:



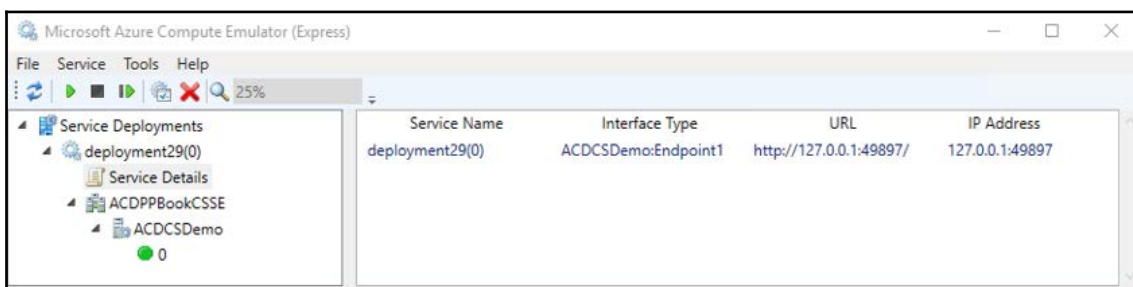
32. You can access the Microsoft Azure Debugging Environment over the task bar from your PC. Look for the Azure icon as shown in the following screenshot:



33. Right-click on the icon and choose either **Show Compute Emulator UI** or **Show Storage Emulator UI**:



34. Here is what the **Microsoft Azure Compute Emulator (Express)** looks like as follows:



35. And here is the Storage Emulator command line tool (I already queried the status):

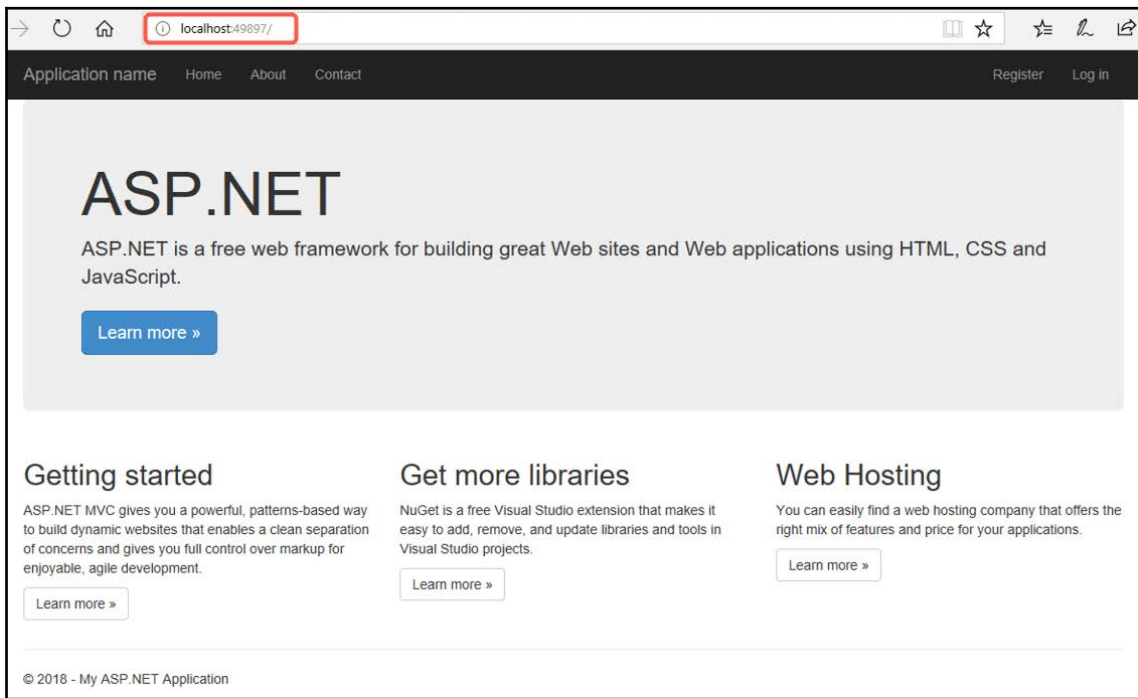
```
Windows Azure Storage Emulator 5.0.0.0 command line tool
Usage:
  AzureStorageEmulator.exe init           : Initialize the emulator database and configuration.
  AzureStorageEmulator.exe start          : Start the emulator.
  AzureStorageEmulator.exe stop           : Stop the emulator.
  AzureStorageEmulator.exe status         : Get current emulator status.
  AzureStorageEmulator.exe clear          : Delete all data in the emulator.
  AzureStorageEmulator.exe help [command] : Show general or command-specific help.

See the following URL for more command line help: http://go.microsoft.com/fwlink/?LinkId=392235

C:\Program Files (x86)\Microsoft SDKs\Azure\Storage Emulator>AzureStorageEmulator.exe status
Windows Azure Storage Emulator 5.0.0.0 command line tool
IsRunning: True
BlobEndpoint: http://127.0.0.1:10000/
QueueEndpoint: http://127.0.0.1:10001/
TableEndpoint: http://127.0.0.1:10002/

C:\Program Files (x86)\Microsoft SDKs\Azure\Storage Emulator>
```

36. At the same time, your web application should be loaded into the browser:

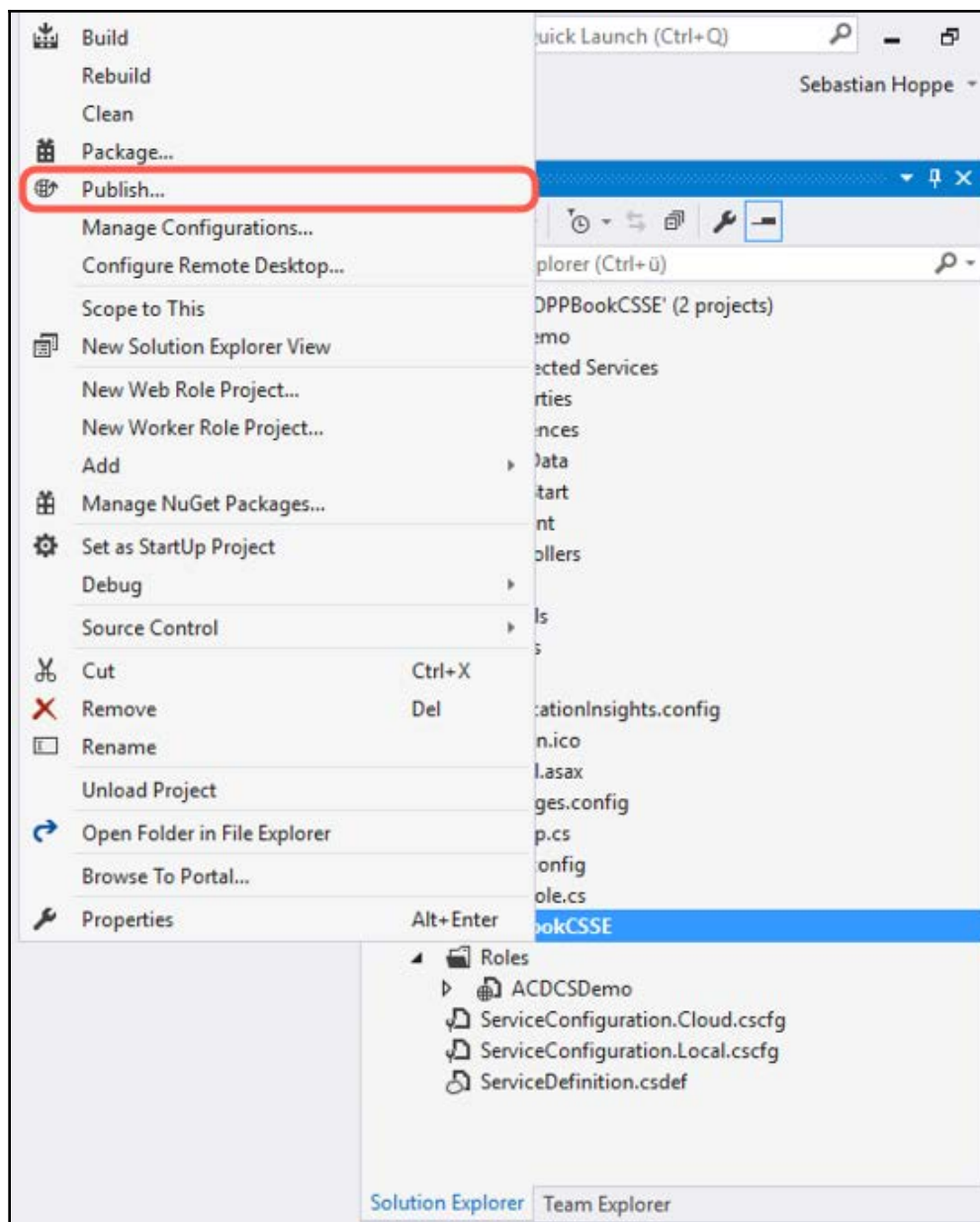


The Microsoft Azure Debugging Environment, with its tools, helps you develop your service, for example, providing you with debugging capabilities. Since the focus of our book is on the implementation of Azure solutions, I will not pursue this further. More information can be found at <https://docs.microsoft.com/en-us/azure/vs-azure-tools-debug-cloud-services-virtual-machines>.

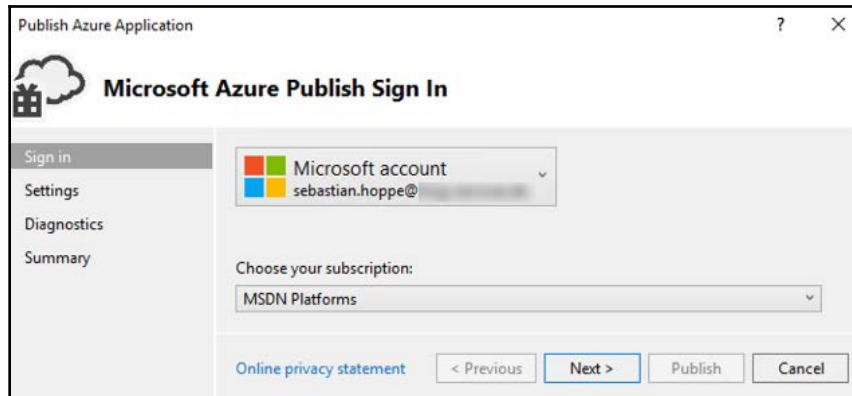
37. The last step is deployment to the Azure platform. There are two possibilities for this:

- Direct **Publish...** to Azure through Visual Studio
- Building a **Package...** file to publish the service

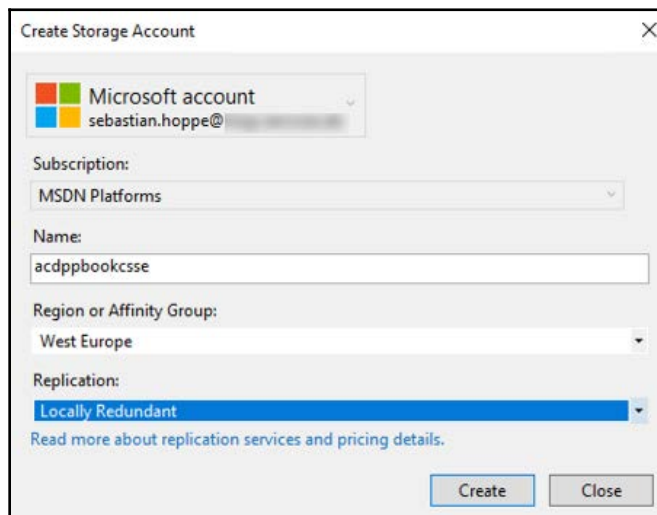
Both ways are triggered through a right-click on the **Role** project. The first way, **Publish...**, is triggered as follows:



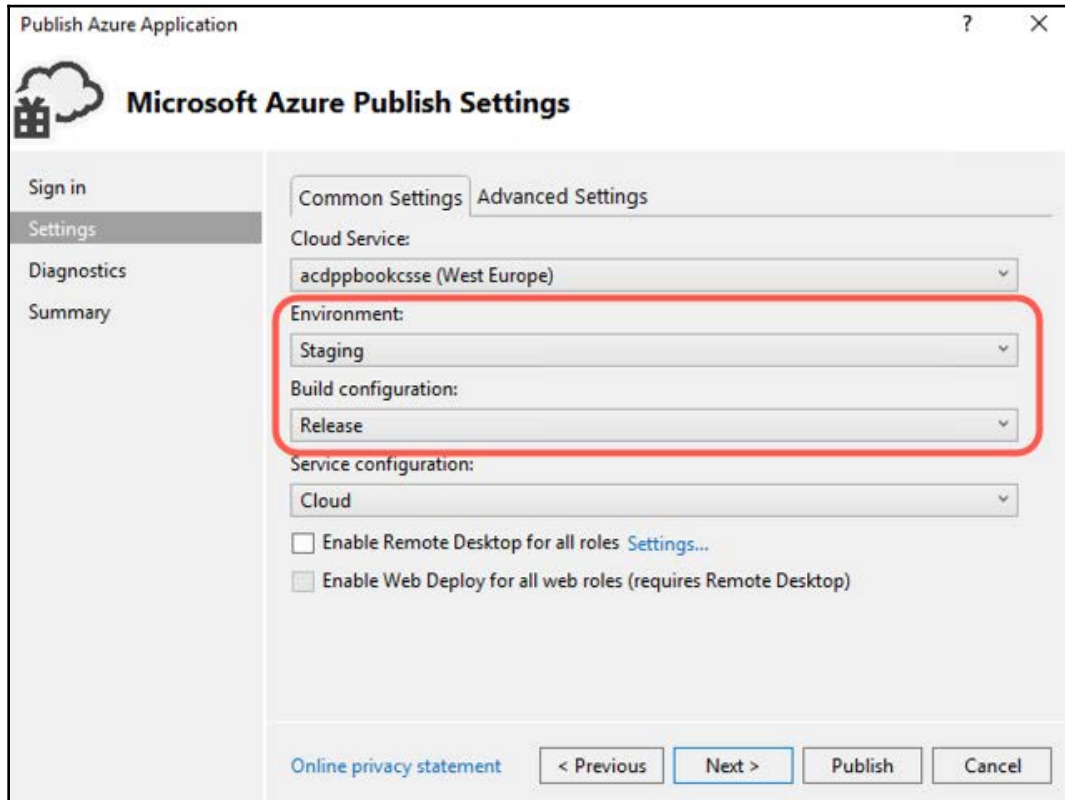
38. Now a dialog for signing in to your Azure Subscription and publishing the service is opened. The following inputs are expected here:
1. On the first page, **Sign in** you must connect with your Azure account. Sign in with the same credentials that you use to login to the Azure portal:



2. If you don't have a storage account in your subscription deployed, a dialog will open for you to create one. In this example, I will create a storage account named `acdppbookcsse` in the **West Europe** region with the replication set to **Locally Redundant**:



3. On the next 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**:



The screenshot shows the "Publish Azure Application" window with the "Microsoft Azure Publish Settings" dialog. The "Common Settings" tab is active. The "Cloud Service" is set to "acdppbookcsse (West Europe)". The "Environment" is set to "Staging" and the "Build configuration" is set to "Release", both of which are highlighted with a red rectangular box. The "Service configuration" is set to "Cloud". There are two checkboxes: "Enable Remote Desktop for all roles" (unchecked) and "Enable Web Deploy for all web roles (requires Remote Desktop)" (unchecked). At the bottom, there is a link for "Online privacy statement" and buttons for "< Previous", "Next >", "Publish", and "Cancel".

4. On the fourth 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 **Enable Remote Debugger** checkbox for all roles:

The screenshot shows the 'Microsoft Azure Publish Settings' dialog box with the 'Advanced Settings' tab selected. The 'Deployment label' is 'ACDPPBookCSSE'. The 'Storage account' is 'acdppbookcsse (West Europe)'. The 'Enable Remote Debugger for all roles' checkbox is checked and highlighted with a red rectangle. Other options include 'Append current date and time' (checked), 'Delete deployment on failure' (unchecked), 'Deployment update' (checked), and 'Enable profiling' (unchecked). Navigation buttons at the bottom include '< Previous', 'Next >', 'Publish', and 'Cancel'. A link for 'Online privacy statement' is also present.

Publish Azure Application

Microsoft Azure Publish Settings

Sign in
Settings
Diagnostics
Summary

Common Settings Advanced Settings

Deployment label:
ACDPPBookCSSE

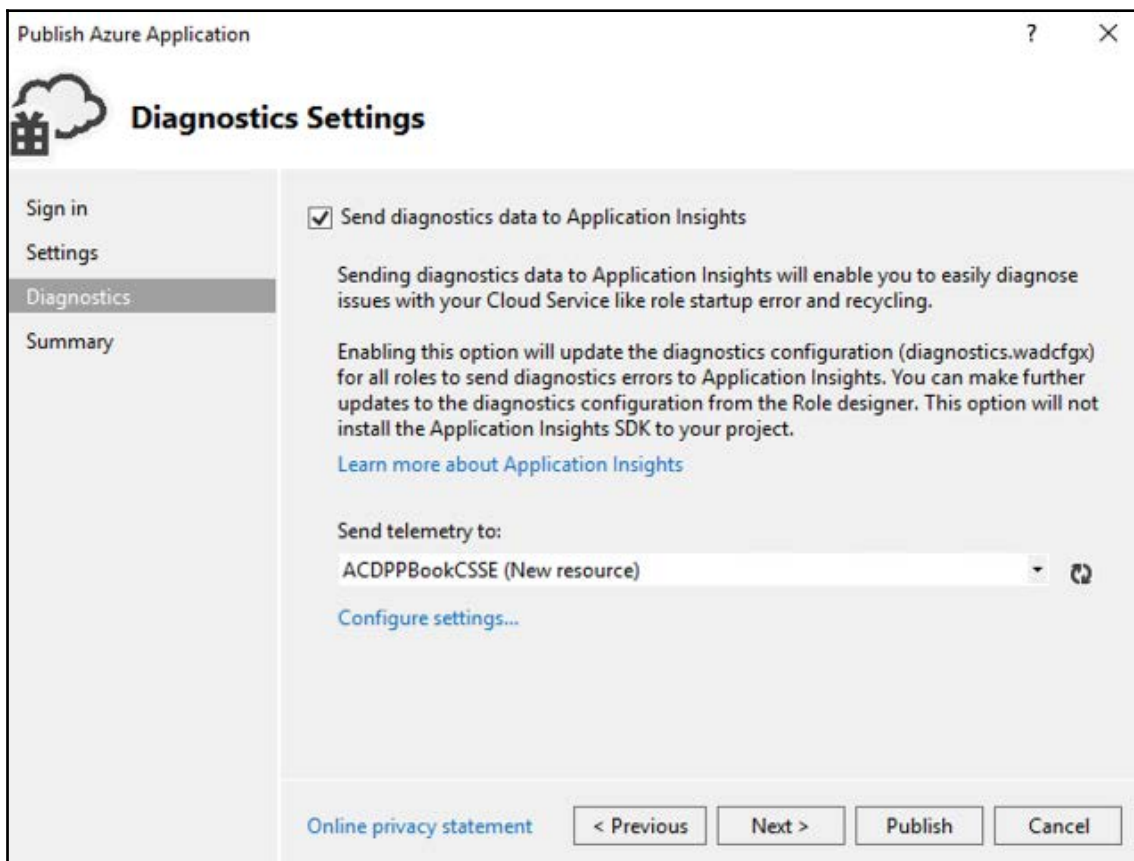
☒ Append current date and time

Storage account:
acdppbookcsse (West Europe)

☐ Delete deployment on failure
☒ Deployment update [Settings...](#)
☐ Enable profiling
☒ Enable Remote Debugger for all roles

[Online privacy statement](#) < Previous Next > Publish Cancel

5. The **Diagnostics Settings** page allows you to link your Azure Cloud Service Application 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](#)

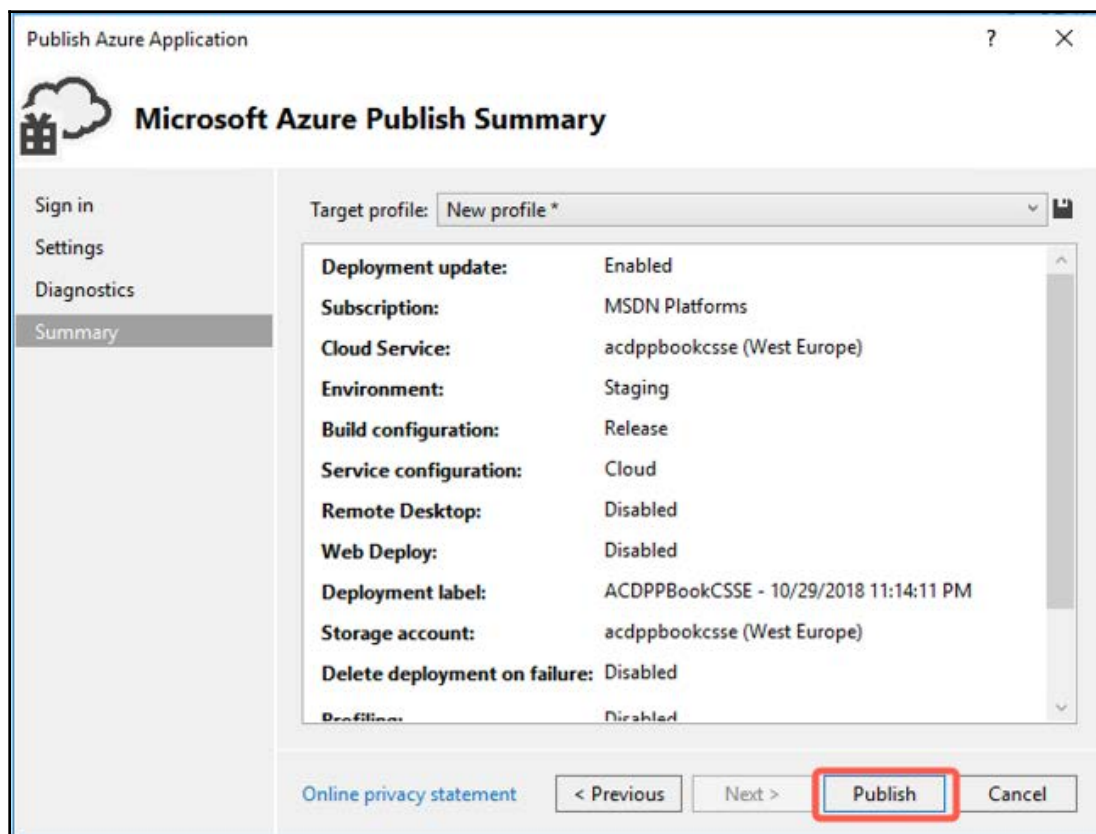
Send telemetry to:
ACDPPBookCSSE (New resource)

[Configure settings...](#)

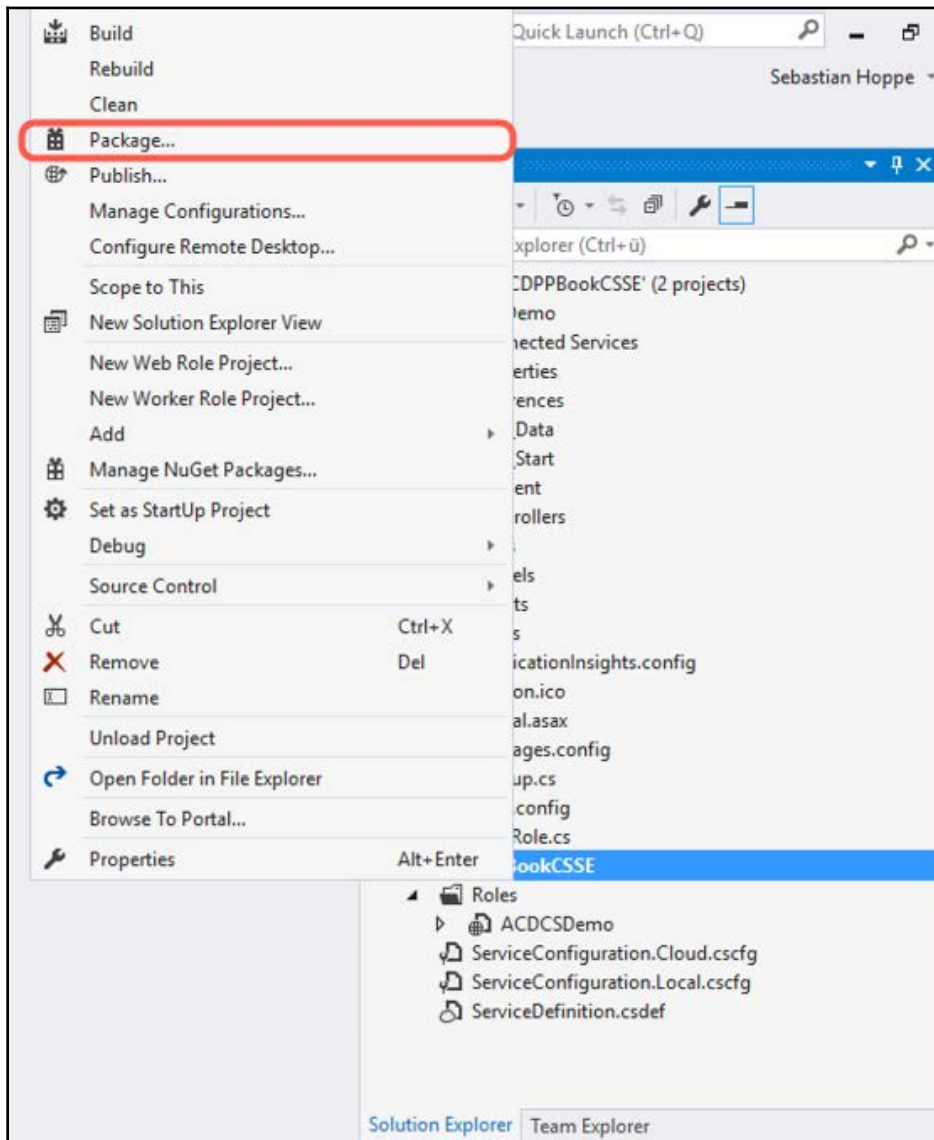
[Online privacy statement](#)

< Previous Next > Publish Cancel

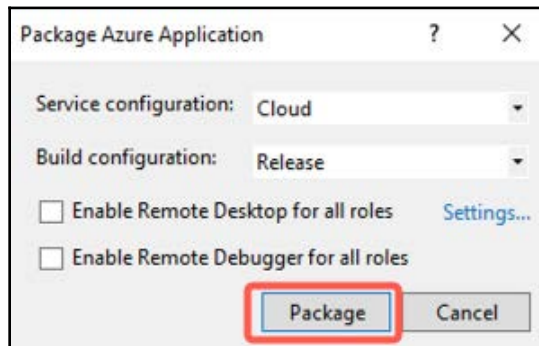
6. Finally, if you are using this path for your deployment, you only need to press the **Publish** button on the **Summary** page:



39. Let's now turn to the second option (the classical way). For this option, you must select the **Package...** option:



40. This time only a tiny dialog window opens, where you must press the **Package** button:

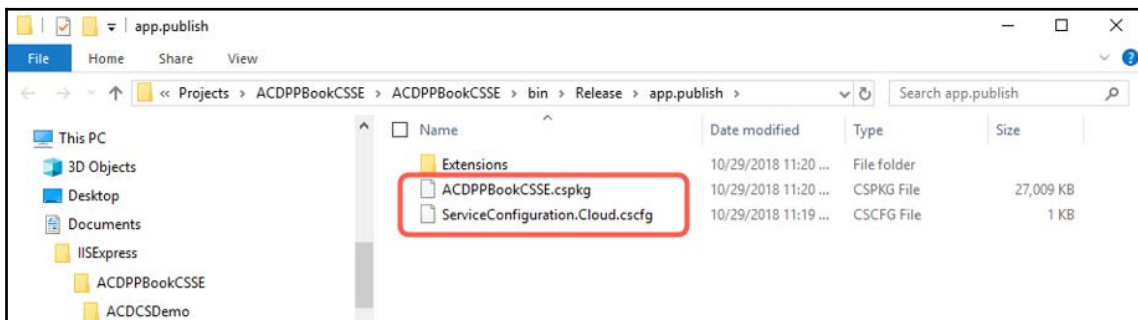


In the initial phase of the Azure platform, there were no publishing procedures. Cloud Services were manually packaged and then uploaded with Azure PowerShell or the Azure management portal.

This process is taken up again, only this time the packing as a background action takes place. So after you did package your Azure Cloud Service, an Explorer window should open and show the contents of your `app.publish` folder.

If this isn't the case go to your Visual Studio Projects folder under your personal Documents and locate your project. You will find the necessary files under `<Projectname>\bin\<Release or Debug>\app.publish`.

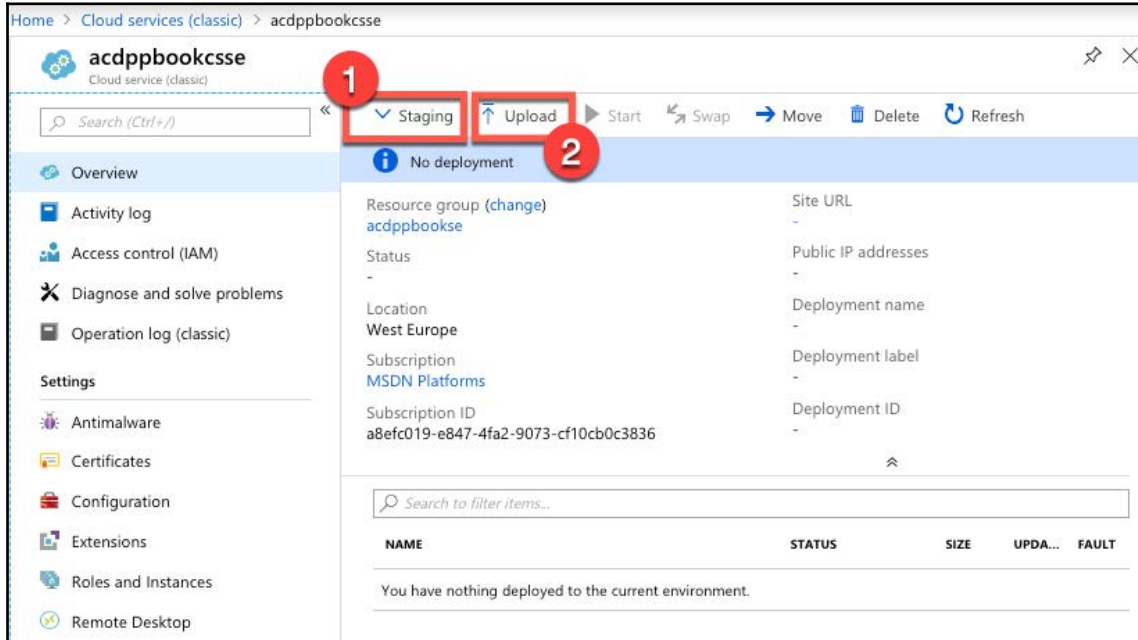
In your `bin` folder you can find the service package and also the Service Configuration File in the `app.publish` subfolder:



Now it's time to return to the Azure platform at <https://portal.azure.com>, 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 as shown in the following screenshot:



2. The **Upload a package** dialog will be opened. First, type a name for the **Deployment label** as shown in the following screenshot:

Home > Cloud services (classic) > acdppbookcsse > Upload a package

Upload a package

acdppbookcsse - Staging

This will create a new 'Staging' deployment

* Deployment label

azbookdeploy ✓

Package/configuration location

☒ from local ☐ from blob

* Storage account ⓘ

Configure required settings >

Upload a package (.cspkg, .zip)

Browse for file... 📁

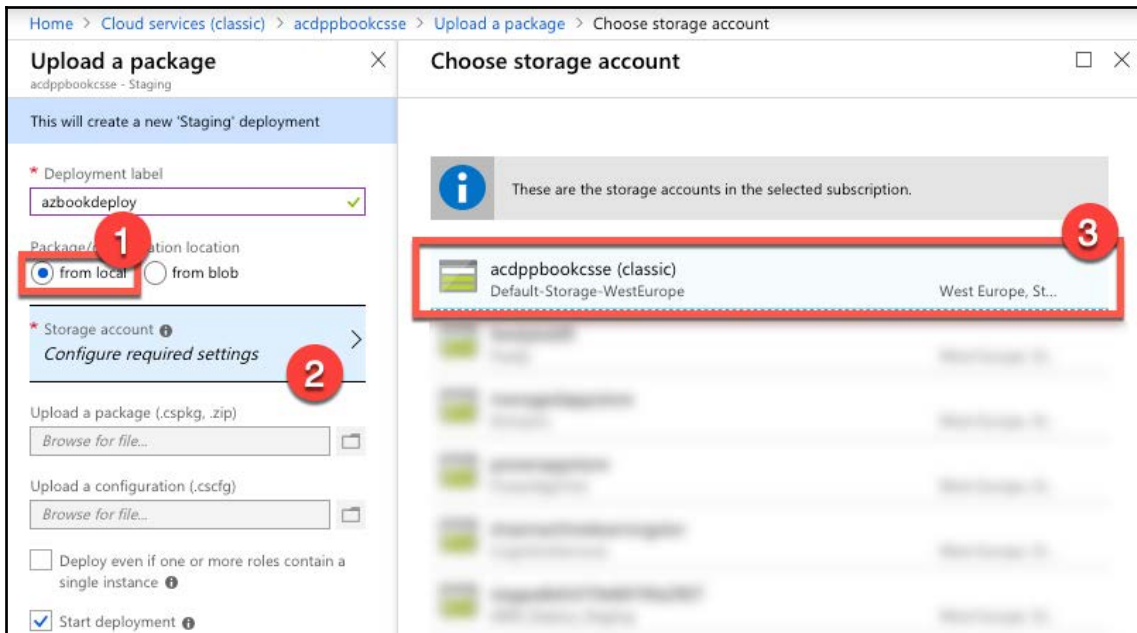
Upload a configuration (.cscfg)

Browse for file... 📁

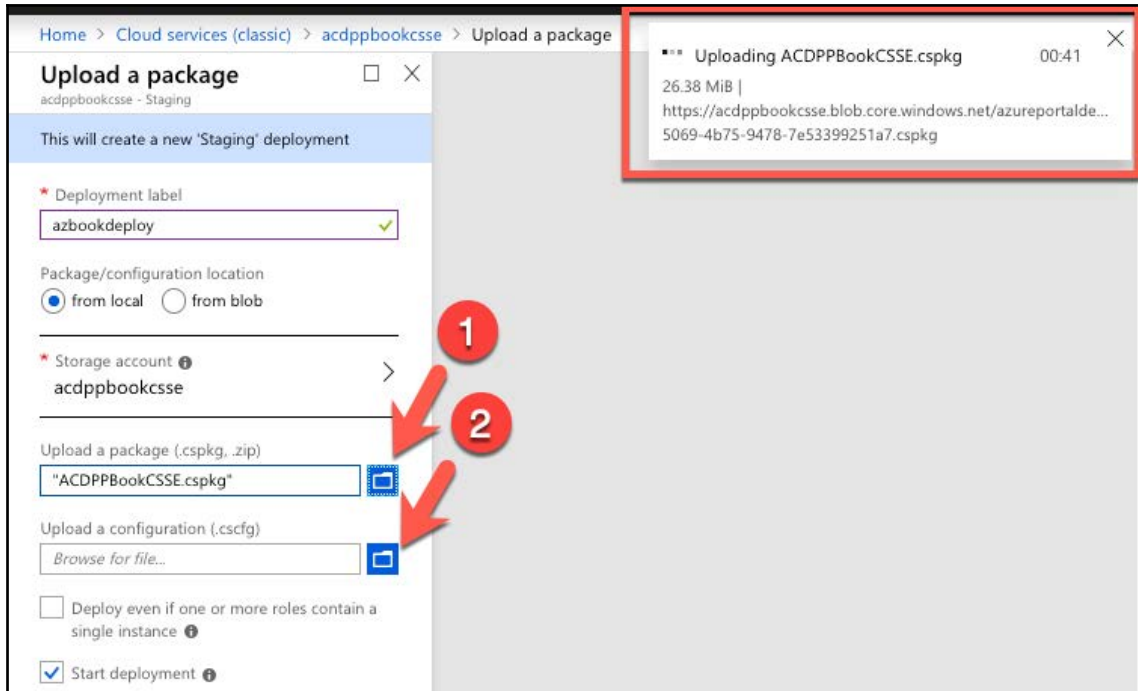
☐ Deploy even if one or more roles contain a single instance ⓘ

☒ Start deployment ⓘ

3. In the next step you must select an Azure Storage account (you can find information about creating a storage account, in [Chapter 6, Implementing Azure Storage](#)), where you want to upload the service package. The location definition (local/blob) selects, whether your package is already placed in a storage account or needs to be uploaded from your local machine—in our scenario, we will use from local:



4. First upload the service package and after this the Service Configuration File. You will notice a dialog showing up in the upper right corner, displaying information about the upload:



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:

Home > Cloud services (classic) > acdppbookcsse > Upload a package

Upload a package

acdppbookcsse - Staging

This will create a new 'Staging' deployment

* Deployment label
azbookdeploy ✓

Package/configuration location
☒ from local ☐ from blob

* Storage account ⓘ
acdppbookcsse >

Upload a package (.cspkg, .zip)
"ACDPPBookCSSE.cspkg" 📁

Upload a configuration (.cscfg)
"ServiceConfiguration.Cloud.cscfg" 📁

☒ Deploy even if one or more roles contain a single instance ⓘ

☒ Start deployment ⓘ

OK

5. The last action is to press the **OK** button as shown in the following screenshot:

Home > Cloud services (classic) > acdppbookcsse > Upload a package

Upload a package

acdppbookcsse - Staging

This will create a new 'Staging' deployment

* Deployment label
azbookdeploy ✓

Package/configuration location
☒ from local ☐ from blob

* Storage account ⓘ
acdppbookcsse >

Upload a package (.cspkg, .zip)
"ACDPPBookCSSE.cspkg" [Browse]

Upload a configuration (.cscfg)
"ServiceConfiguration.Cloud.cscfg" [Browse]

☒ Deploy even if one or more roles contain a single instance ⓘ

☒ Start deployment ⓘ

OK

6. Ready! Your Cloud Service is now available on the Azure platform:

Home > Cloud services (classic) > acdppbookcsse

acdppbookcsse
Cloud service (classic)

Search (Ctrl+F)

Staging Update Stop Swap Move Delete Refresh

Resource group (change)
acdppbookcsse

Status
Running

Location
West Europe

Subscription
MSDN Platforms

Subscription ID
[REDACTED]

Site URL
<http://71fa0988a0604409b8cbd165e0709689.cloudapp.net/>

Public IP addresses
137.117.231.6

Deployment name
ef99a1f1854f973bfd5b3225de97c78

Deployment label
azbookdeploy

Deployment ID
71fa0988a0604409b8cbd165e0709689

Search to filter items...

NAME	STATUS	SIZE	UPDA...	FAULT
ACDCSDemo	Unknown	Small	0	0

Show data for last: 1 hour 6 hours 12 hours 1 day 7 days

CPU percentage

Percentage CPU (Avg)
ACDCSDemo

Disk read and write

Disk Read Bytes/sec
ACDCSDemo

Disk Write Bytes/sec
ACDCSDemo

Network in and out

Network In (Sum)
ACDCSDemo

Network Out (Sum)
ACDCSDemo

Summary

In this chapter, we learned the main parts about the Azure Cloud Service offer. The first part of this chapter was all about the basic knowledge of the Azure Cloud Service, the Cloud architecture and the difference between Azure Cloud Service and Azure App Services.

In the second part of this chapter followed a detailed description how to create an Azure Cloud Service and some information about the deployment process.

In the next chapters, we will also cover implementing Azure Governance.

Questions

After working through this chapter, you should be able to answer the following questions:

1. Which roles can be defined in an Azure Cloud Service?
2. How many roles can be defined in an Azure Cloud Service?
3. What is an Azure Load Balancer probe?
4. What are the valid values for `osFamily` in a Cloud Service?
5. What are the two possibilities to deploy my Cloud Service to the Azure platform?

Further reading

The following book can be used to go further into the topics of this chapter:

- **Implementing Microsoft Azure Infrastructure Solutions: Exam Guide 70-533:** <https://www.packtpub.com/virtualization-and-cloud/implementing-microsoft-azure-infrastructure-solutions-exam-guide-70-533>

10

Implementing Azure Governance

Public cloud services such as Azure provide a wide variety of services that are very easy to implement and consume without the interaction and integration of a company's IT team. This implies the risk that some departments and teams could create an idea to implement their own shadow IT that may not comply with the company's security and regulation policies.

The easiest way to mitigate that risk is for a company to sign an extension to an existing Microsoft **Enterprise Agreement (EA)** to integrate a dedicated part of Azure consumption. This provides the option to access the Microsoft Enterprise portal for Azure and define budgets per team or organization.

After having enabled Azure for a company from the contract side, we are now able to implement the basic guidelines for using Azure; so-called **Azure governance**. Azure governance can be split into **organizational governance** and **technical governance**. This is what this chapter is about.

The following topics will be covered in this chapter:

- Organizational Azure governance
- Technical Azure governance
- Azure tools to support governance

Technical requirements

For running containers in a cloud environment, no specific installations are required, as you only need the following:

- A computer with an internet browser
- An Azure subscription (if not available, a trial could fit too, which can be obtained at <https://azure.microsoft.com/en-us/free/>)

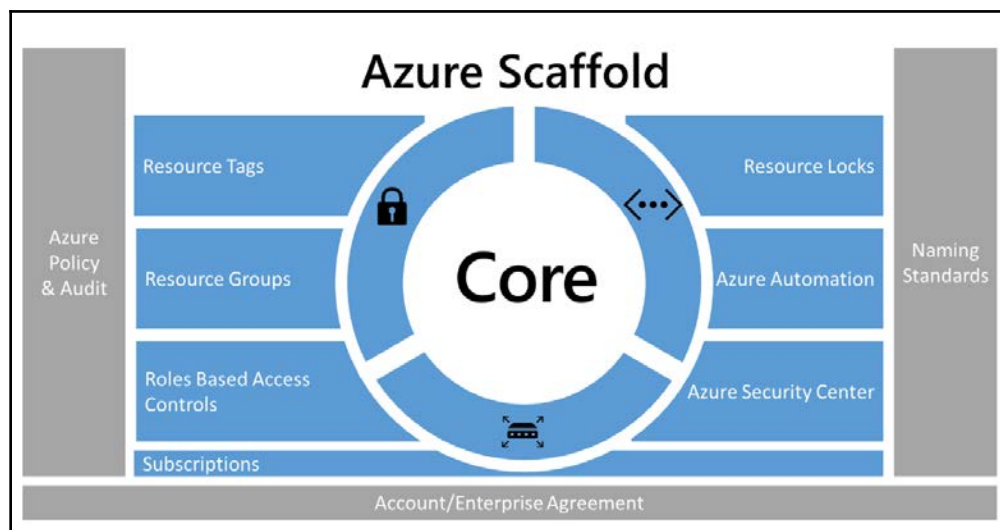
The code in this chapter can be found here at: <https://github.com/PacktPublishing/Implementing-Azure-Solutions-Second-Edition>.

Organizational governance

Organizational governance is all the organizational decisions and stuff that needs to be concluded before defining it using technical tools. So, we will need to define an Azure hierarchy at first, as within Azure there are some options we will need to clarify before going on.

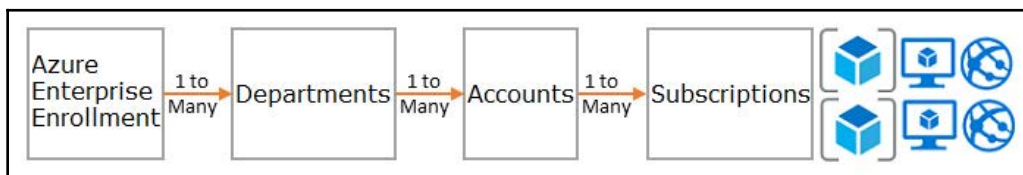
Azure hierarchy

Azure has some vocabulary that we will need to define at first:



The Azure scaffold that you can see in the preceding diagram easily summarizes the components Azure itself relies on. Beyond this, we need a plan for departments, accounts, and subscriptions:

- **Department:** This is the same as the department you know from a company's organizational chart
- **Account:** This is a credential that has access to the Azure tenant portal
- **Subscription:** This in Azure is the root of all resources and defines the limits of them to be deployed at their maximum (for example, cores, RAM, or storage):



The preceding diagram clarifies the relationship between those components. In general, there are three models available on which you can plan your hierarchy:

- **Functional approach:** The functional approach defines the hierarchy based on the departments of a company, which is generally quite easy to define:

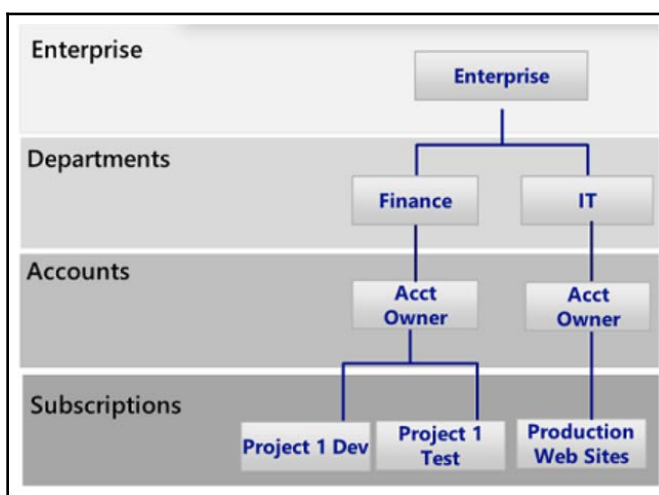


Image source: <https://azure.microsoft.com/en-us/blog/organizing-subscriptions-and-resource-groups-within-the-enterprise/>

- **Business unit approach:** This approach defines the hierarchy based on a company's business units:

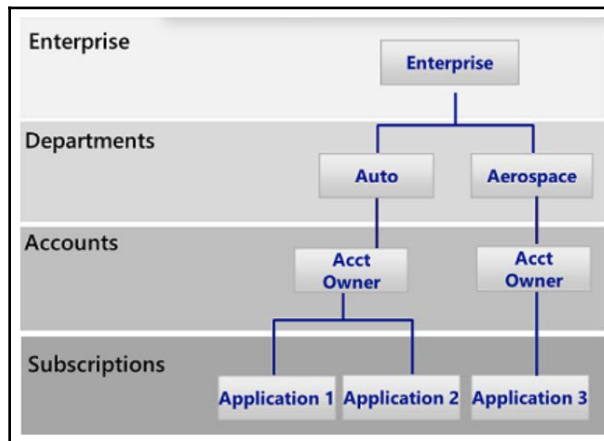


Image source: <https://azure.microsoft.com/en-us/blog/organizing-subscriptions-and-resource-groups-within-the-enterprise/>

- **Geographic approach:** The geographic approach defines the hierarchy based on geographical regions:

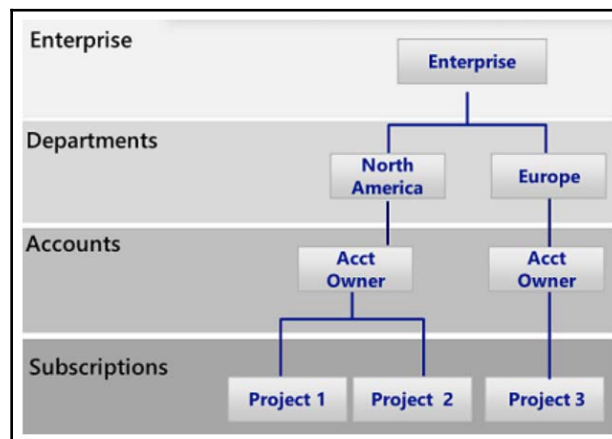


Image source: <https://azure.microsoft.com/en-us/blog/organizing-subscriptions-and-resource-groups-within-the-enterprise/>

The hierarchy approach is the basis for all subscription design and planning.

Naming standards

Having defined the Azure hierarchy, the next step is setting up a naming standard definition for all Azure resources. This is one of the most important steps within the governance, as in the future there may be hundreds and thousands of resources defined in Azure. If no naming definition has been set when starting, the risk of getting lost in there is quite high. There are hundreds of naming definitions—each consultant around the world may have set one—but following the Microsoft best practice would make life easier.

The set of Microsoft naming conventions and patterns can be found at: <https://docs.microsoft.com/en-us/azure/architecture/best-practices/naming-conventions>.

As defined, the following naming conventions should have been set:

```
<Company> <Department (optional)> <Product Line (optional)> <Environment>
```

Structuring your resources this way provides an out-of-the-box solution for documentation, scalability, and generality, and is more or less an unwritten law.

Technical governance

Having now defined the hierarchy and naming conventions and patterns, we will have to have a look at the technical governance feature. This is made up of tools and functionality that provide the technical solution for the topics that have been defined in the *Organizational governance* section.

This means that now we need to talk about the following:

- Resource groups
- Resource tags
- **Role-based access control (RBAC)**
- Policies
- Auditing
- Azure resource logs

Resource groups

An Azure resource group is something provided by **Azure Resource Manager (ARM)**. It gives you the option to group Azure resources; for example, a useful option of resource groups is collecting all resources that belong to one Azure service/solution into a resource group. An example for this may be a virtual machine in Azure consisting of different resources, such as IP address, NIC, storage, CPU, and so on. Collecting all those resources in one resource group provide a great overview even in complex environments.

Resource groups do not support nesting. This means that you cannot place one resource group into another to define hierarchies. It is a one-level organizational option that provides an easy way to set up policies and roles, and even makes removing Azure resources that belong to an Azure solution quite easy, because you just have to delete the complete resource groups and all resources from within are deleted automatically. If you were to do this manually, you would have to have the hierarchy in mind, as each resource belongs to another, and remove them following the hierarchy from the end to the beginning.

Creating a resource group is quite easy from the Azure portal, but can even be done from Azure CLI and Azure PowerShell:

The screenshot shows the Azure portal's search results for 'Resource group'. The search bar at the top contains the text 'resource group'. Below the search bar, a table lists the results. The first result is 'Resource group' by Microsoft, categorized under 'VM Extensions'. Other results include 'Hyperglance up to 2000 resources', 'Hyperglance up to 500 resources', 'Hyperglance up to 1,000 resources', 'Hyperglance up to 100 resources', 'Hyperglance up to 5000 resources', 'Hyperglance up to 250 resources', 'Resource Central - Meeting Room Booking System', and 'AzureStack cloudhub extension'.

On the right side of the screen, a pane titled 'Resource group' by Microsoft provides additional information. It states: 'Resource groups enable you to manage all your resources. Resource groups are enabled by Azure Resource Manager. Resource resources as a logical group which serves as the lifecycle bo within it. Typically a group will contain resources related to group may contain a Website resource that hosts your publ relational data used by the site, and a Storage Account that'. Below this text is a 'Save for later' button. Further down, it shows 'PUBLISHER: Microsoft' and 'USEFUL LINKS: Documentation'. At the bottom of the pane is a 'Create' button.

NAME	PUBLISHER	CATEGORY
Resource group	Microsoft	VM Extensions
Hyperglance up to 2000 resources	Hyperglance	Compute
Hyperglance up to 500 resources	Hyperglance	Compute
Hyperglance up to 1,000 resources	Hyperglance	Compute
Hyperglance up to 100 resources	Hyperglance	Compute
Hyperglance up to 5000 resources	Hyperglance	Compute
Hyperglance up to 250 resources	Hyperglance	Compute
Resource Central - Meeting Room Booking System	Add-On Products	Compute
AzureStack cloudhub extension	Microsoft	AzureStack

Now, let's fill in the properties as shown in the following screenshot:

Home > New > Marketplace > Everything > Resource group > Resource group

Resource group

Create an empty resource group

* Resource group name
MyFirstResourceGroup ✓

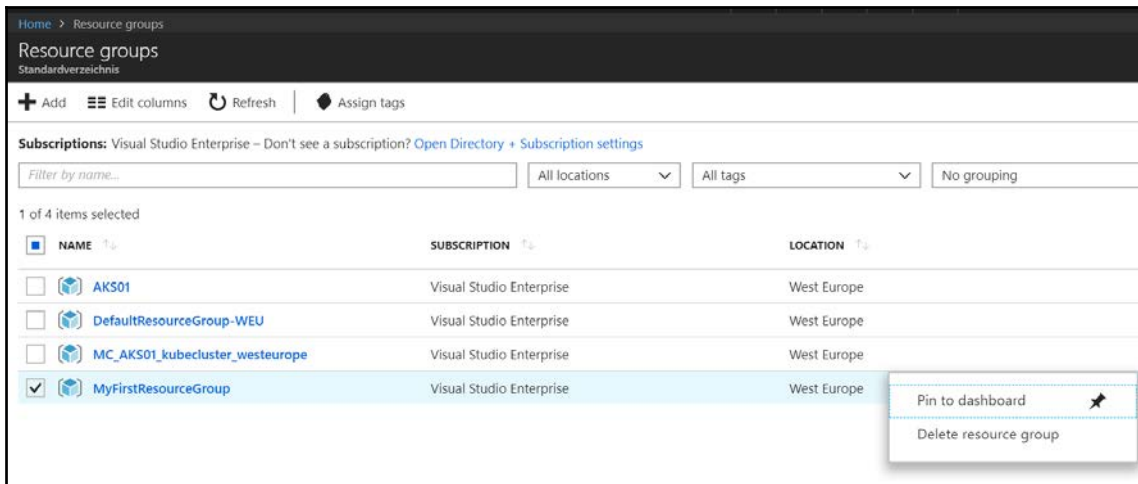
* Subscription
Visual Studio Enterprise ▼

* Resource group location
West Europe ▼

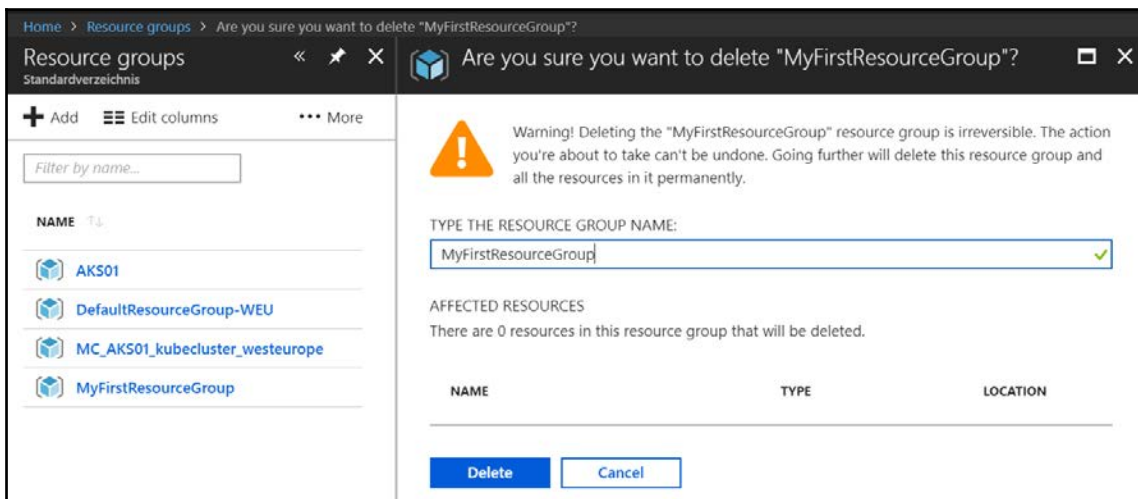
Create

After having defined the unique subscription name, the subscription it belongs to, and the corresponding Azure region, it will be created within seconds.

It can be removed just as easily as the resource group was set up:



After having put in the resource group name, it will be removed, including all resources that are in it:

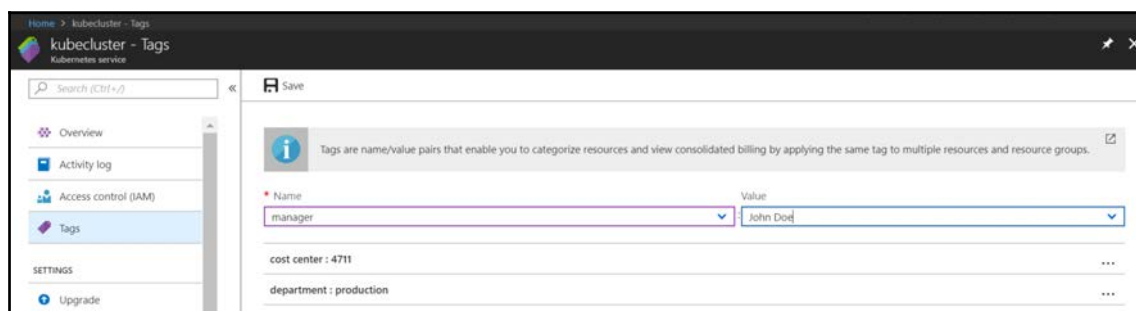


Resource tags

Each Azure resource has different parameters that need to be defined before deploying it. Resource tags provide the option to individually set parameters that are even pre-defined as mandatory. A maximum of 15 tags are available per resource and they have to follow the following pattern:

- Storage tag names have a maximum of 128 characters per tag and 256 characters per tag value
- For all other tags, 256 is the maximum number of characters per tag and 512 is the maximum number of characters per tag value

For applying tags, the minimum role permission is contributor. Tags are shown in the Azure service bill, too. Therefore, it can be used for basic billing filtering, too:



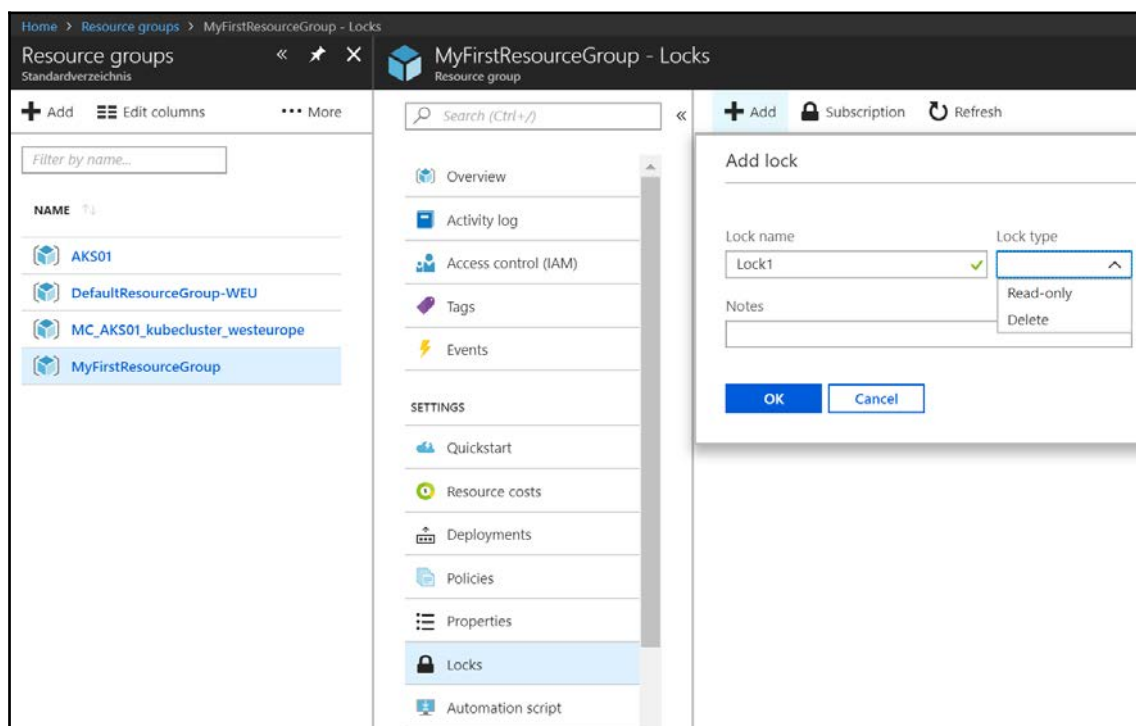
As you have seen, setting up tags is very easy.



For more information on setting up tags using PowerShell or Azure CLI, visit <https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-using-tags>.

Resource locks

Azure resource locks protect resources from unwanted changes or deletions. You can set two states—**Delete** and **Read-only**. The **Delete** state means that the resource can be modified but not deleted, while the **Read-only** state means that the resource cannot be modified at all:



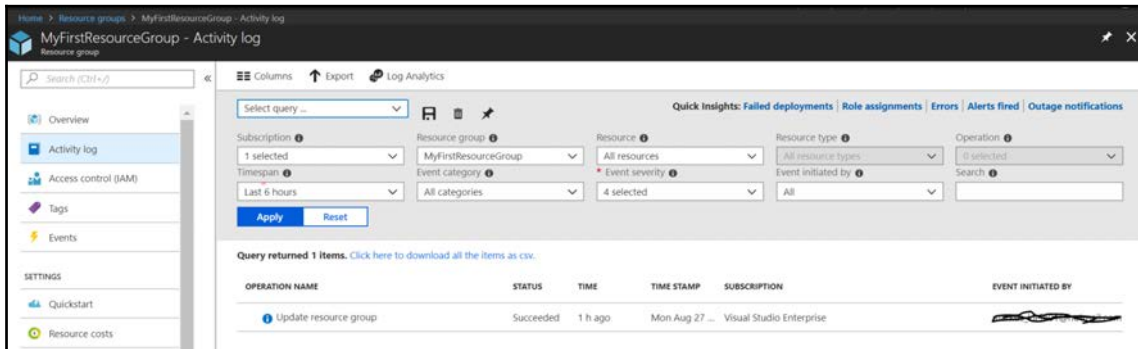
When you are setting the locks through PowerShell or the CLI, the **Delete** lock is called **Do Not Delete**.



For your Azure governance setting, Azure resource locks are quite important. Otherwise, contributors or higher-permitted roles may delete or edit the most important resources of a service by chance. This would mean that a good backup is required.

Resource auditing

Azure resource auditing is the change log for each Azure resource. It provides information on each time you PUT, POST, or DELETE a resource. It does not log the GET call of a resource. Azure auditing log files are retained for 90 days before getting overwritten. If you need to retain them for a longer period of time, a monitoring solution such as Azure Log Analytics can help:



As you can see, Activity Logs are quite easy to work with.

Management groups

The most important preparation in **Azure Active Directory (AAD)** to enable governance is to create the required management groups, as they define the scope of management and administration as the next level under the **Subscriptions** tab. Otherwise, this would be the maximum level of scope depth.

Azure Management Groups is - so to speak the part where all features that are currently available only in the "Enterprise Agreement Portal (EA Portal)", which will be discomissioned during the next months.

This configuration is part of the Azure AD features. You can configure it as shown in the following screenshot:

Home > Default Directory - Properties

Default Directory - Properties

Azure Active Directory

Search (Ctrl+/)

Save Discard

- Organizational relationships
- Roles and administrators
- Enterprise applications
- Devices
- App registrations
- Application proxy
- Licenses
- Azure AD Connect
- Custom domain names
- Mobility (MDM and MAM)
- Password reset
- Company branding
- User settings
- Properties**
- Notifications settings
- Security
- Identity Secure Score (Previe...

* Name
Default Directory ✓

Country or region
Germany

Location
EU Model Clause compliant datacenters

Notification language
English ✓

Global admin can manage Azure Subscriptions and Management Groups
☒ Yes ☐ No

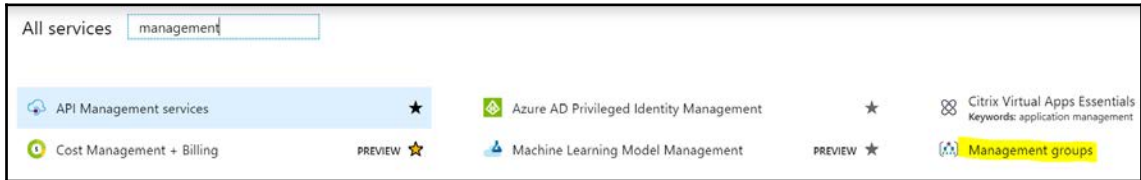
Directory ID
063607a5-2b42-41f3-b9c3-7035811c6415

Technical contact
markus_klein01@hotmail.com ✓

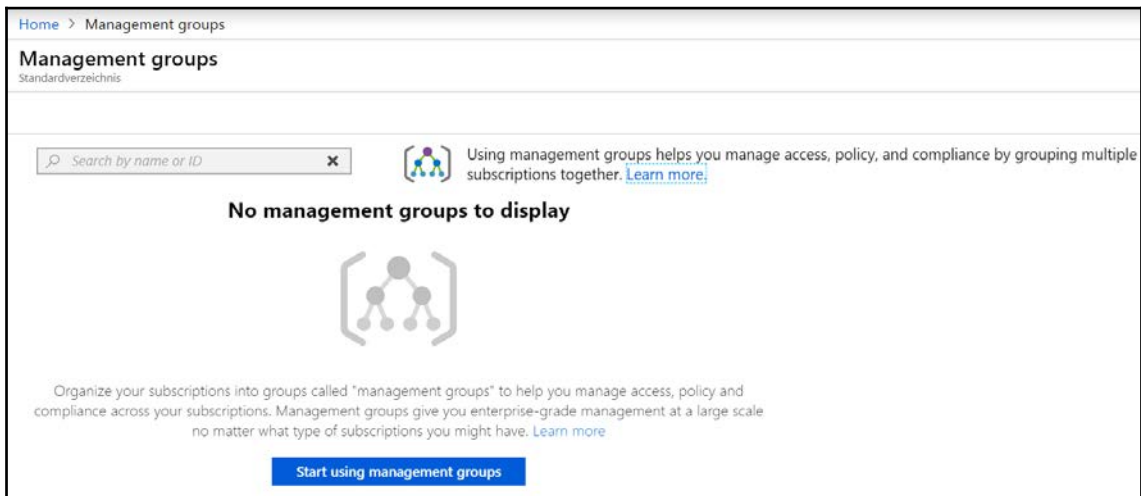
Global privacy contact
✓

Privacy statement URL
✓

This feature needs to be enabled globally from an Azure AD Global Administrator. Afterward, the configuration can continue by searching for **Management groups** in **All services** in the Azure portal:



If no management group is configured, the **Start using management groups** button needs to be pressed in order to continue:

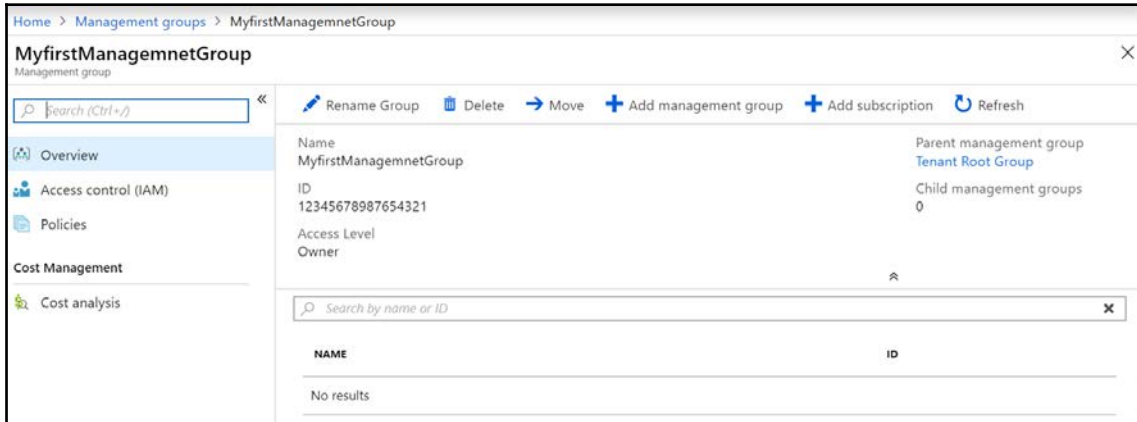


The required management group ID is the unique identifier that needs to be set and cannot be updated afterward. In addition, the display name is required as shown in the following screenshot:

The following table gives an overview on the required RBAC directory role to create, modify, or delete management groups:

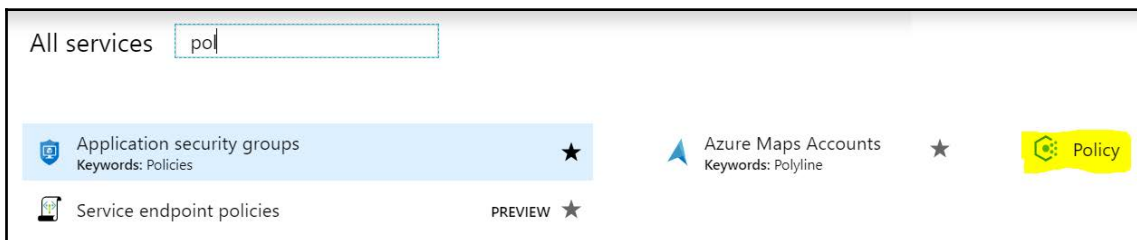
RBAC Role Name	Create	Rename	Move	Delete	Assign Access	Assign Policy	Read
Owner	X	X	X	X	X	X	X
Contributor	X	X	X	X			X
MG Contributor*	X	X	X	X			X
Reader							X
MG Reader*							X
Resource Policy Contributor						X	
User Access Administrator					X		

After the management group is enabled, the following screenshot displays all the required details. Now that the required management groups are available, we can go on with the *Azure Policies* section:

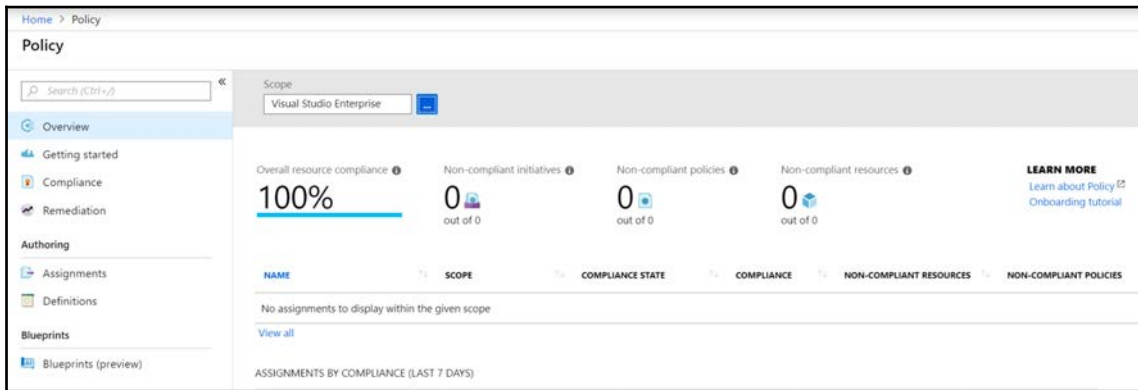


Azure Policies

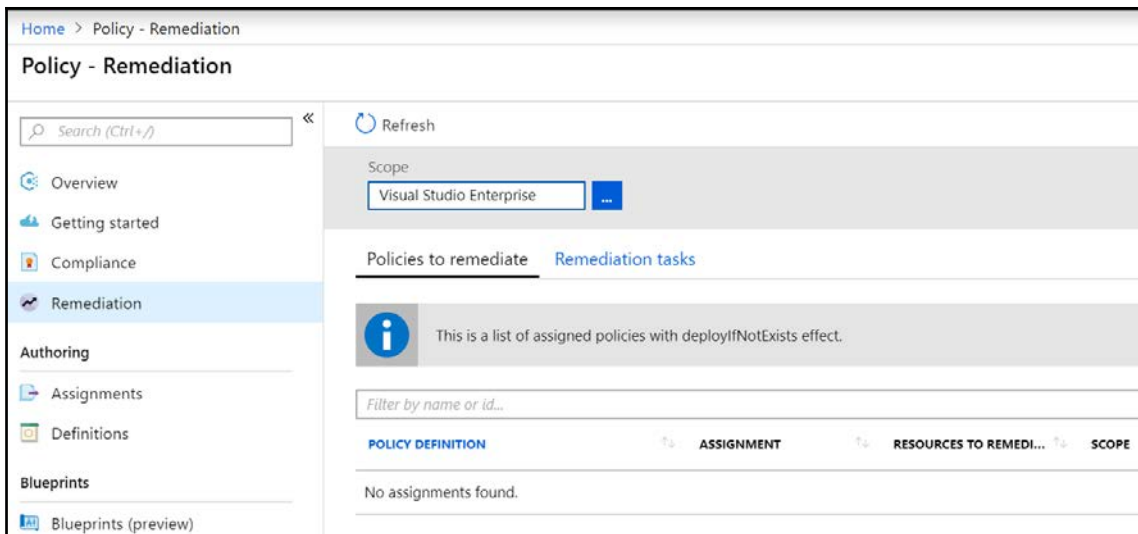
In the first part of this chapter, we talked about naming conventions and further organizational governance configurations. As these are all theoretical definitions, the question always is how to make sure that they are all set; if a configuration does not fulfill the requirements, you run into a problem. Therefore, Azure Policies is a great toolset to configure the theoretical Azure governance in Azure and make sure that all resources comply to them, as if they do not, the appropriate user is unable to deploy the resource and the portal gives an error message. As you can see, Azure Policies can be found under **All services** in the Azure portal:



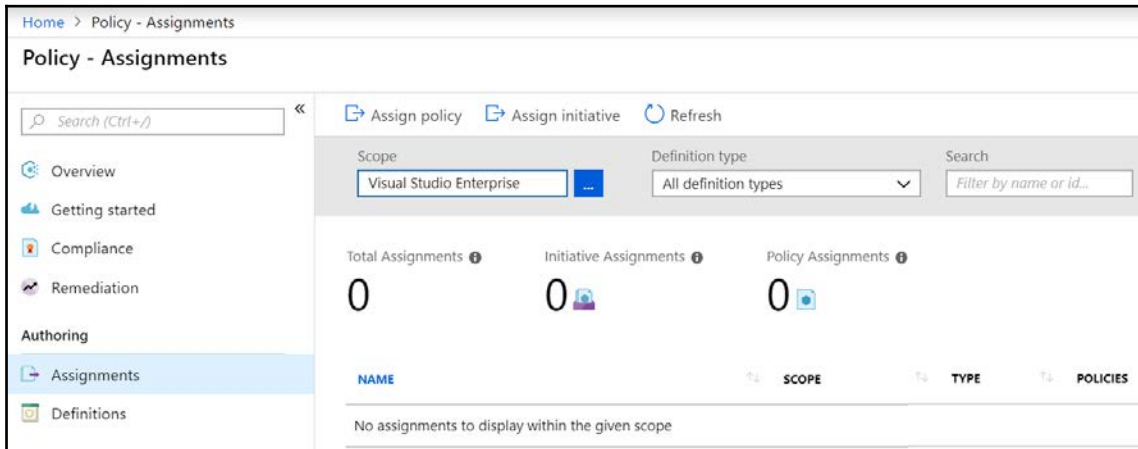
The following screenshot displays a general overview of all Azure Policies and which of the resources are compliant or non-compliant:



The **Remediation** filter displays all issues where, for whatever reason, a policy ran into an issue while being applied to that resource. In general, this view should have no entries:



In the **Authoring** view, you can create policies and assignments. A policy is a definition of rules a resource needs to comply to:



As you can see, there is already a lot of policy definition available that you just need to assign to make your configurations compliant. Best practice is to have a minimum setup for the following policies:

- Allowed Azure regions
- Naming conventions for resources

- Mandatory fields of resources

Home > Policy - Definitions

Policy - Definitions

Search (Ctrl+/)

Overview

Getting started

Compliance

Remediation

Authoring

Assignments

Definitions

Blueprints

Blueprints (preview)

Resources

Resource Graph (preview)

Privacy

User privacy

Initiative definition

Policy definition

Refresh

[Preview]: Deploy Log Analytics Agent for Linux ...	Built-in	Policy	Monitoring
Audit enabling of diagnostic logs in Azure Data ...	Built-in	Policy	Data Lake
Audit VMs that do not use managed disks	Built-in	Policy	Compute
Deploy default OMS VM Extension for Windows...	Built-in	Policy	Monitoring
[Preview]: Audit minimum number of owners fo...	Built-in	Policy	Security Center
[Preview]: Monitor unencrypted VM Disks in Az...	Built-in	Policy	Security Center
Audit resource location matches resource group...	Built-in	Policy	General
[Preview]: Audit Dependency Agent Deploymen...	Built-in	Policy	Monitoring
[Preview]: Deploy VM extension to audit Windo...	Built-in	Policy	Guest Configuration
Audit transparent data encryption status	Built-in	Policy	SQL
[Preview]: Deploy Dependency Agent for Windo...	Built-in	Policy	Monitoring
Audit use of classic virtual machines	Built-in	Policy	Compute
Enforce tag and its value	Built-in	Policy	General
Audit provisioning of an Azure Active Directory ...	Built-in	Policy	SQL
[Preview]: Monitor unprotected web application...	Built-in	Policy	Security Center
Audit enabling of only secure connections to yo...	Built-in	Policy	Cache
[Preview]: Deploy VM extension to audit Windo...	Built-in	Policy	Guest Configuration

[460]

An initiative in Azure Policy is a configuration that requires a task (such as installing agents or enabling configurations):

Home > Policy - Definitions > Initiative definition

Initiative definition

New Initiative definition

BASICS

* Definition location

Visual Studio Enterprise

* Name ⓘ

Enter the name for your initiative

Description ⓘ

Describe the initiative you are authoring

Category ⓘ

☒ Create new ☐ Use existing

Category

POLICIES AND PARAMETERS

Initiatives are composed of one or more policies. Add policies to this Initiative from the list on the right.

Initiative parameters ⓘ

PARAMETER NAME	DISPLAY NAME	TYPE	ALLOWED VALUES
----------------	--------------	------	----------------

Save Cancel

AVAILABLE DEFINITIONS

Type Search

All types Filter by name or L...

Policy Definitions (100)

- + [Preview]: Deploy Log Analytics Agent for Linux VMs
Built-in
Deploy Log Analytics Agent for Linux VMs if the VM image (OS) is in the list defined and the agent is not installed.
- + Audit enabling of diagnostic logs in Azure Data Lake Store
Built-in
Audit enabling of logs and retain them up to a year. This enables you to recreate activity trails for investigation purposes when a security incident occurs or your network is compromised
- + Audit VMs that do not use managed disks
Built-in
This policy audits VMs that do not use managed disks
- + Deploy default OMS VM Extension for Windows VMs
Built-in
This policy deploys OMS VM Extensions on Windows VMs, and connects to the selected Log Analytics workspace
- + [Preview]: Audit minimum number of owners for subscription
Built-in
It is recommended to designate more than one subscription owner in order to have administrator access.

As you can see in the following screenshot, creating your own policy definitions is quite simple, as it is another JSON configuration. A collection of Azure custom policies is available at <https://github.com/Azure/azure-policy>:

Policy definition

New Policy definition

BASICS

* Definition location

* Name ⓘ

Description

Category ⓘ

☒ Create new ☐ Use existing

Category

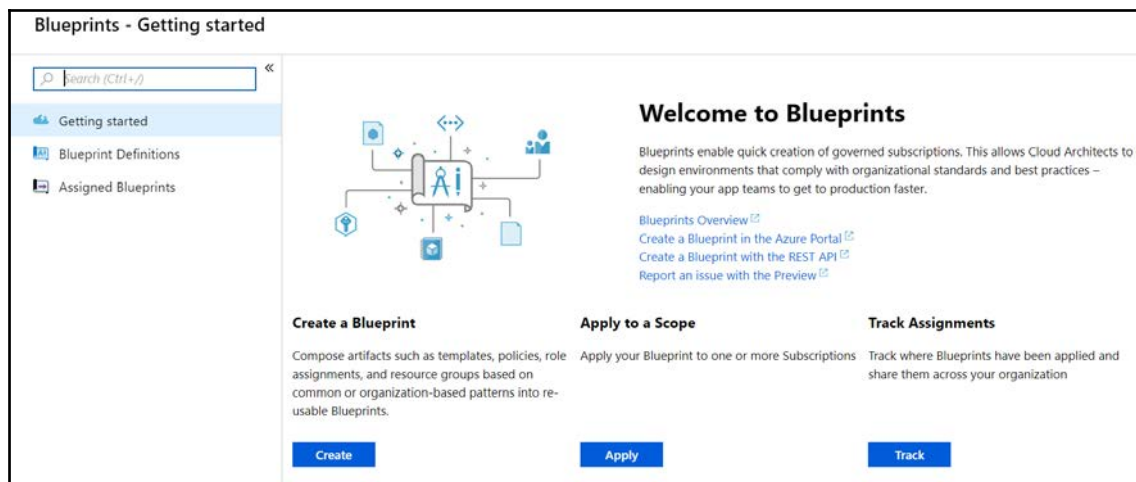
POLICY RULE

↓ [Import sample policy definition from GitHub](#)

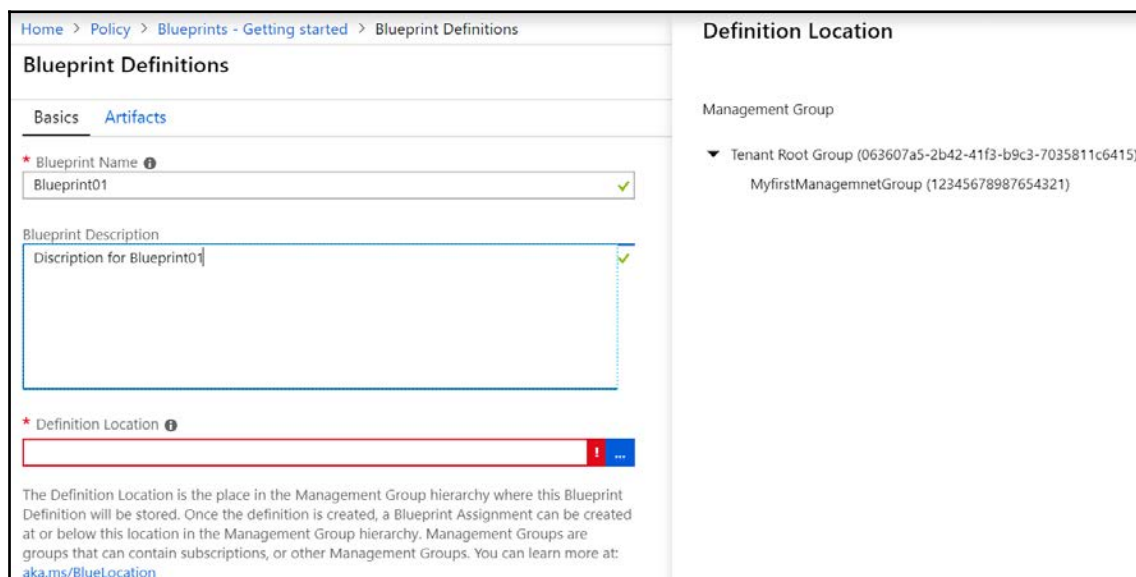
☒ [Learn more about policy definition structure](#)

```
1 {
2   "policyRule": {
3     "if": {
4       "not": {
5         "field": "location",
6         "in": "[parameters('allowedLocations')]"
7       }
8     },
9     "then": {
10      "effect": "audit"
11    }
12  },
13  "parameters": {
14    "allowedLocations": {
15      "type": "Array",
16      "metadata": {
17        "description": "The list of allowed locations for resources.",
18        "displayName": "Allowed locations",
19        "strongType": "location"
20      }
21    }
22  }
23 }
```

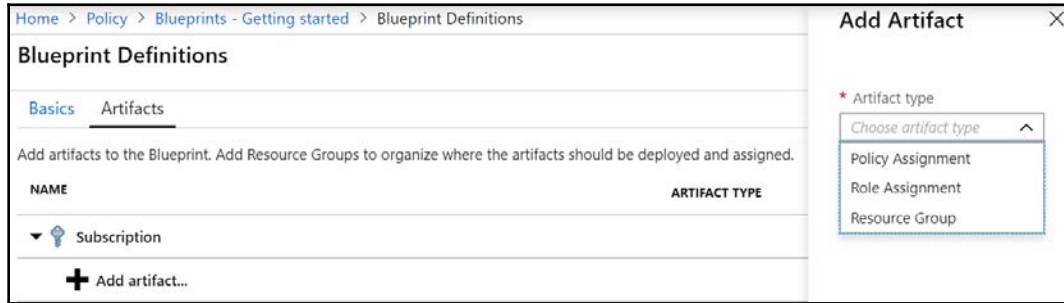
Azure blueprint is a new feature which provides you the feature to define—Azure RBAC, Azure policy, Azure naming conventions and Azure automation as a template. This template can be example and imported. So, Azure Blueprints give you a way to transfer Azure governance templates between subscriptions, and so on:



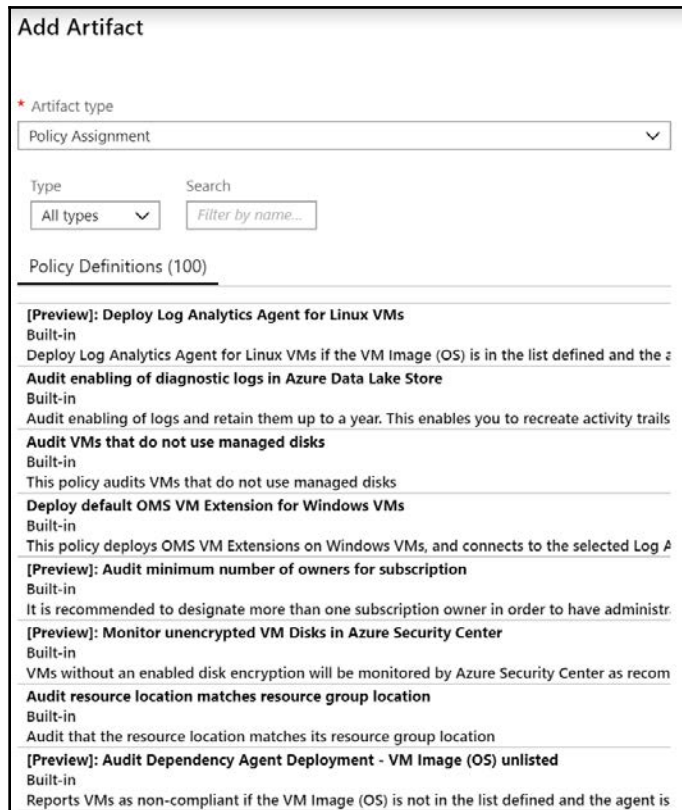
The blueprint definition can be scoped at subscriptions or management groups as a location:



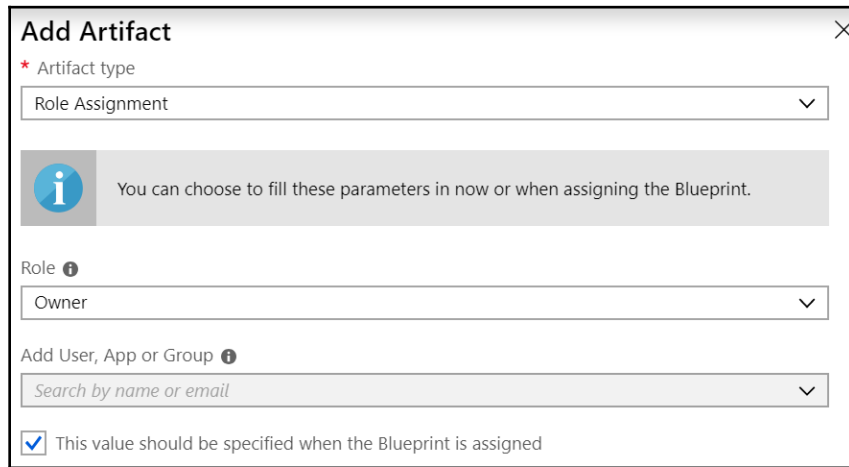
Setting up blueprint definitions also allows you to set up **artifacts**, which can be a policy assignment, a role assignment, or a resource group. These artifacts set up the basics for applying a blueprint:



Artifacts can be enabled already used for the preceding preconfigured list. This makes it easy to use them and you do not need to create your own ones, if you find them on the list:



A **Role Assignment** option of **Artifact type** provides the option to enable directory roles based on users, applications, or groups:



The screenshot shows the 'Add Artifact' dialog box with the 'Artifact type' set to 'Role Assignment'. It includes a message box stating that parameters can be filled now or when assigning the Blueprint. The 'Role' is set to 'Owner', and the 'Add User, App or Group' dropdown is set to 'Search by name or email'. A checkbox is checked, indicating that the value should be specified when the Blueprint is assigned.

Add Artifact ✕

* Artifact type

Role Assignment ▼

 You can choose to fill these parameters in now or when assigning the Blueprint.

Role ⓘ

Owner ▼

Add User, App or Group ⓘ

Search by name or email ▼

☒ This value should be specified when the Blueprint is assigned

The artifact resource group configures the resource group naming convention and the required location:



The screenshot shows the 'Add Artifact' dialog box with the 'Artifact type' set to 'Resource Group'. It includes a message box stating that parameters can be filled now or when assigning the Blueprint. The 'Resource Group Name' is set to 'RG-', and the 'Location' is set to 'West Europe'. Two checkboxes are checked, indicating that the values should be specified when the Blueprint is assigned.

Add Artifact ✕

* Artifact type

Resource Group ▼

 You can choose to fill these parameters in now or when assigning the Blueprint.

Resource Group Name

RG-

☒ This value should be specified when the Blueprint is assigned

Location

West Europe ▼

☒ This value should be specified when the Blueprint is assigned

As the following screenshot demonstrates, you can see the full blueprint configuration and publish it to your Azure environment:

Home > Policy > Blueprints - Blueprint Definitions > Blueprint01

Blueprint01

Blueprints

Publish Blueprint

Edit Blueprint

Delete Blueprint

Name

Blueprint01

Management Group

MyfirstManagemnetGroup

Version

Draft

State

Draft

Description

Discription for Blueprint01

⌵

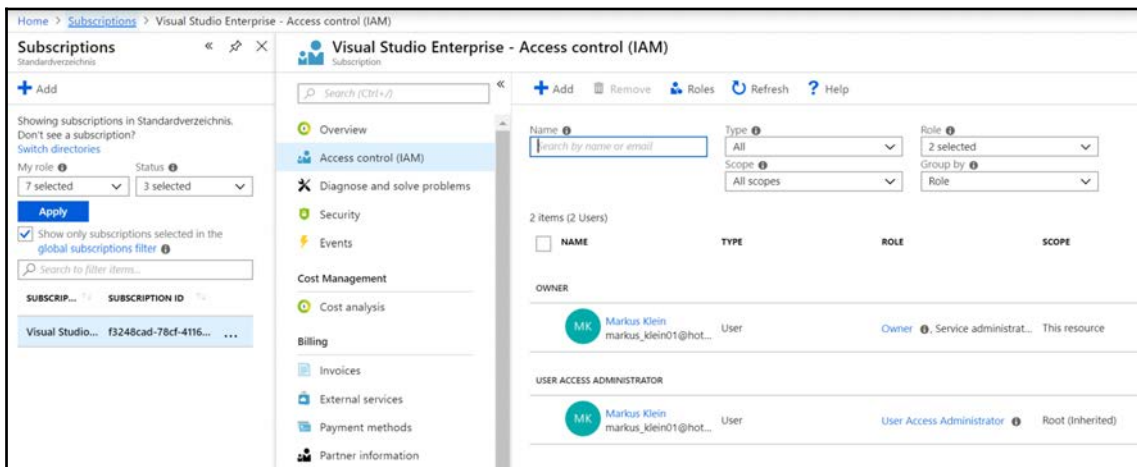
Latest Artifacts

ARTIFACT NAME	RESOURCE TYPE
▼ Assigned Subscription	Subscription
ResourceGroup	Resource Group

As you have seen in this part of the chapter, Azure Policies provide a configuration to enable and apply compliance to your Azure environment. Non-compliant configuration is allowed, and the result is that it cannot be created. Azure Policy is a must-have configuration within Azure environments.

RBAC

Azure identity management is set as AAD. All user configuration and permissions rely on directory roles. You can see all enabled roles when looking at the user itself in Azure AD:



The screenshot shows the Azure portal interface for the 'Visual Studio Enterprise - Access control (IAM)' page. The left sidebar contains navigation links for Subscriptions, Overview, Access control (IAM), Security, Events, Cost Management, Cost analysis, Billing, Invoices, External services, Payment methods, and Partner information. The main content area displays a search bar and filters for roles. Below the filters, a table lists the roles assigned to the user 'Markus Klein' (markus_klein01@hotmail.com). The roles are 'Owner' and 'User Access Administrator'. The 'Owner' role is assigned to 'This resource' (Service administration), and the 'User Access Administrator' role is assigned to 'Root (inherited)'.

NAME	TYPE	ROLE	SCOPE
OWNER			
Markus Klein markus_klein01@hotmail.com	User	Owner	Service administrat... This resource
USER ACCESS ADMINISTRATOR			
Markus Klein markus_klein01@hotmail.com	User	User Access Administrator	Root (inherited)

Around 70 default directory roles are available and can be enabled easily. This list may vary, depending on the enabled resource provided:

Home > Subscriptions > Visual Studio Enterprise - Access control (IAM) > Roles

Roles

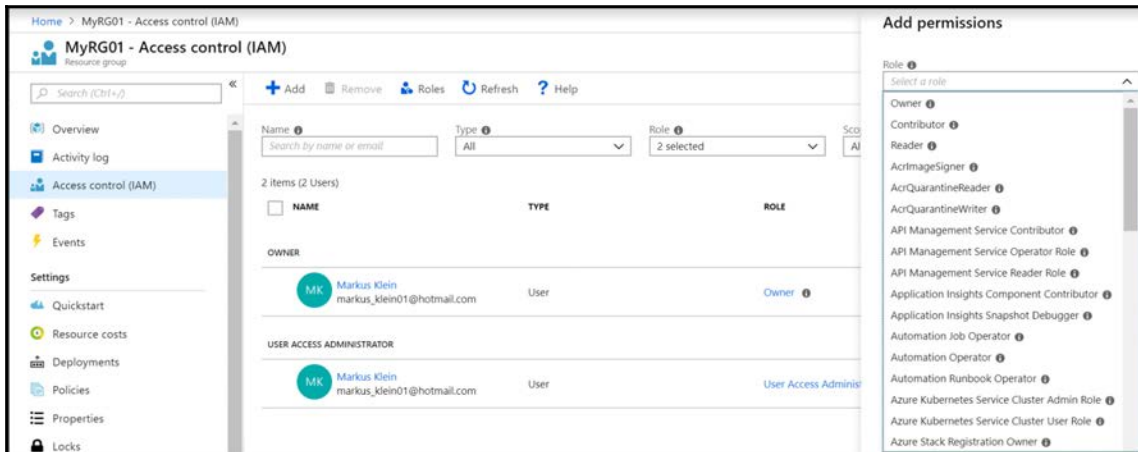
Visual Studio Enterprise

NAME	USERS	GROUPS
 Owner ⓘ	0	1
 Contributor ⓘ	0	0
 Reader ⓘ	0	0
 User Access Administrator ⓘ	1	0
 AcrImageSigner ⓘ	0	0
 AcrQuarantineReader ⓘ	0	0
 AcrQuarantineWriter ⓘ	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
 Application Insights Snapshot Debugger ⓘ	0	0
 Automation Job Operator ⓘ	0	0
 Automation Operator ⓘ	0	0
 Automation Runbook Operator ⓘ	0	0
 Azure Kubernetes Service Cluster Admin Role ⓘ	0	0

By showing the permissions of a role, you will see all features a role member can fulfill in Azure:

Permissions (preview)		
Contributor		
RESOURCE PROVIDER	MANAGEMENT	DATA
 84codes.CloudAMQP	All	--
 AppDynamicsPro AppDynamicsForAzure	All	--
 Aspera.Transfers	All	--
 Auth0 Cloud	All	--
 Azure Data Box	All	--
 Azure Database Migration Service	All	--
 Azure Log Analytics	All	--
 Azure Stack Resource Provider	All	--
 Bot Service Resource Provider	All	--
 Citrix.Cloud	All	--
 CloudSimple Private Cloud IAAS	All	--
 Clouddyn.Analytics	All	--
 Conexlink MyCloudIT	All	--
 Crypteron DataSecurity	All	--
 Domain Services Resource Provider	All	--
 Dynatrace DynatraceSaaS	All	--

Configuring roles in Azure is quite easy, as you will need to set them up through the **Access control (IAM)** entry for a resource or resource group. If your requirement for a role is not available in Azure, you have the chance to set up your own role definition as RBAC. For example, if you will need to have a role that is a contributor, not all features that are allowed by default need to be enabled (for instance, deleting a virtual machine). Therefore, setting up the custom role is a configuration need:



The following JSON shows an example for a custom RBAC role. The `NotActions` define options that are disabled and the `Actions` define the enabled features. This defines the custom directory role, then. The process of setting up custom RBAC roles is as follows:

- Defining all features to be enabled and disabled
- Exporting the role that is nearest to the one you need to apply
- Modifying the JSON as per your requirements
- Importing the new role and enabling it

```
JSON Copy

{
  "Name": "Virtual Machine Operator",
  "Id": "88888888-8888-8888-8888-888888888888",
  "IsCustom": true,
  "Description": "Can monitor and restart virtual machines.",
  "Actions": [
    "Microsoft.Storage/*/read",
    "Microsoft.Network/*/read",
    "Microsoft.Compute/*/read",
    "Microsoft.Compute/virtualMachines/start/action",
    "Microsoft.Compute/virtualMachines/restart/action",
    "Microsoft.Authorization/*/read",
    "Microsoft.Resources/subscriptions/resourceGroups/read",
    "Microsoft.Insights/alertRules/*",
    "Microsoft.Insights/diagnosticSettings/*",
    "Microsoft.Support/*"
  ],
  "NotActions": [
  ],
  "DataActions": [
  ],
  "NotDataActions": [
  ],
  "AssignableScopes": [
    "/subscriptions/{subscriptionId1}",
    "/subscriptions/{subscriptionId2}",
    "/subscriptions/{subscriptionId3}"
  ]
}
```

The following custom role properties are available:

Property	Required	Type	Description
<code>Name</code>	Yes	String	The display name of the custom role. Must be unique to your tenant. Can include letters, numbers, spaces, and special characters. Maximum number of characters is 128.
<code>Id</code>	Yes	String	The unique ID of the custom role. For Azure PowerShell and Azure CLI, this ID is automatically generated when you create a new role.
<code>IsCustom</code>	Yes	String	Indicates whether this is a custom role. Set to <code>true</code> for custom roles.
<code>Description</code>	Yes	String	The description of the custom role. Can include letters, numbers, spaces, and special characters. Maximum number of characters is 1024.
<code>Actions</code>	Yes	String[]	An array of strings that specifies the management operations that the role allows to be performed. For more information, see Actions .
<code>NotActions</code>	No	String[]	An array of strings that specifies the management operations that are excluded from the allowed <code>Actions</code> . For more information, see NotActions .
<code>DataActions</code>	No	String[]	An array of strings that specifies the data operations that the role allows to be performed to your data within that object. For more information, see DataActions (Preview) .
<code>NotDataActions</code>	No	String[]	An array of strings that specifies the data operations that are excluded from the allowed <code>DataActions</code> . For more information, see NotDataActions (Preview) .
<code>AssignableScopes</code>	Yes	String[]	An array of strings that specifies the scopes that the custom role is available for assignment. Currently cannot be set to the root scope (<code>" / "</code>) or a management group scope. For more information, see AssignableScopes and Organize your resources with Azure management groups .

The required permissions to work with custom RBAC roles need to be as follows:

Task	Operation	Description
Create/delete a custom role	<code>Microsoft.Authorization/roleDefinition/write</code>	Users that are granted this operation on all the <code>AssignableScopes</code> of the custom role can create (or delete) custom roles for use in those scopes. For example, Owners and User Access Administrators of subscriptions, resource groups, and resources.
Update a custom role	<code>Microsoft.Authorization/roleDefinition/write</code>	Users that are granted this operation on all the <code>AssignableScopes</code> of the custom role can update custom roles in those scopes. For example, Owners and User Access Administrators of subscriptions, resource groups, and resources.
View a custom role	<code>Microsoft.Authorization/roleDefinition/read</code>	Users that are granted this operation at a scope can view the custom roles that are available for assignment at that scope. All built-in roles allow custom roles to be available for assignment.

As you have learned in this part of the chapter, directory roles provide a way to enable and disable features in your Azure environment. If the default roles do not apply, you can set up your own custom RBAC roles and populate them to Azure.



A more detailed overview for all tasks around RBAC is available at <https://docs.microsoft.com/de-de/azure/role-based-access-control/role-assignments-powershell>.

Azure tooling

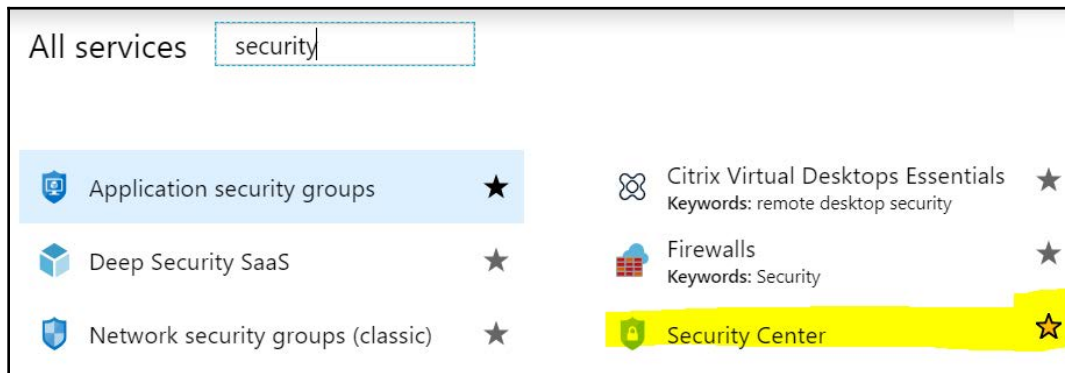
After having populated all custom Azure governance settings for your environment, you will have a toolset to monitor the custom behavior for all settings. The most important tools are the following:

- Azure Security Center
- Azure Cost Management

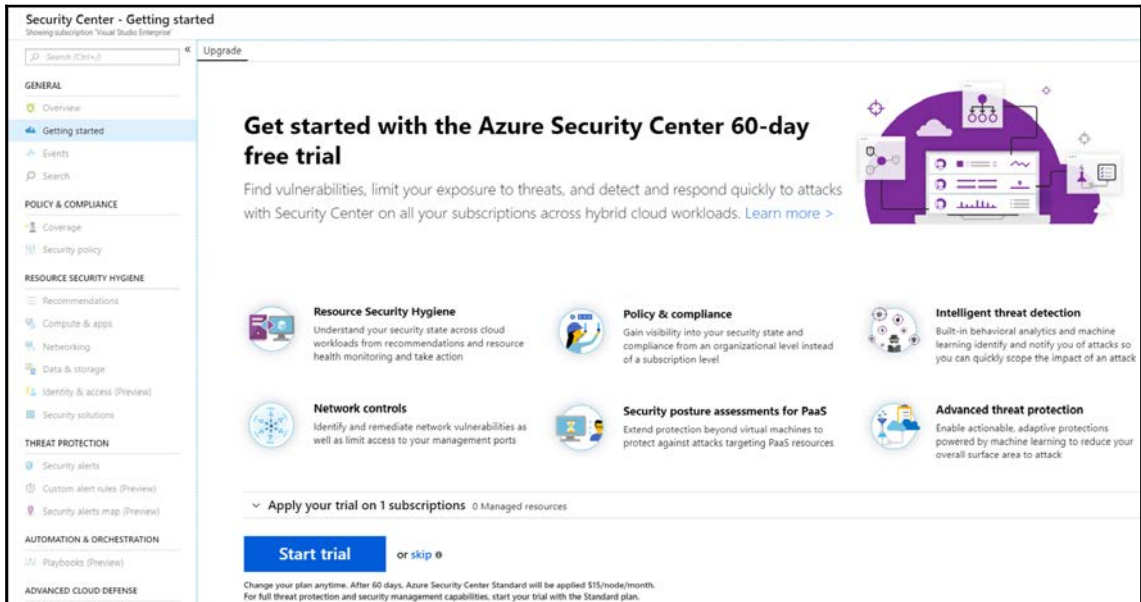
Both these features are part of the third and last part of this chapter.

Azure Security Center

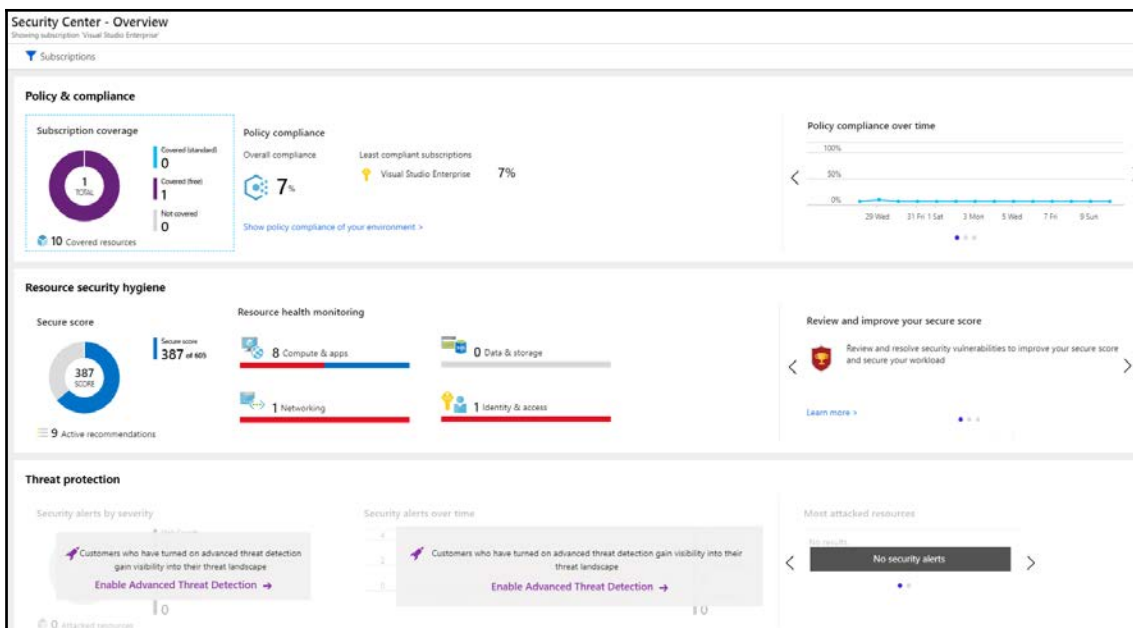
Azure Security Center is a powerful tool set to monitor and alert on security breaches of your Azure services. You can find security center in the **All services** view of the Azure portal:



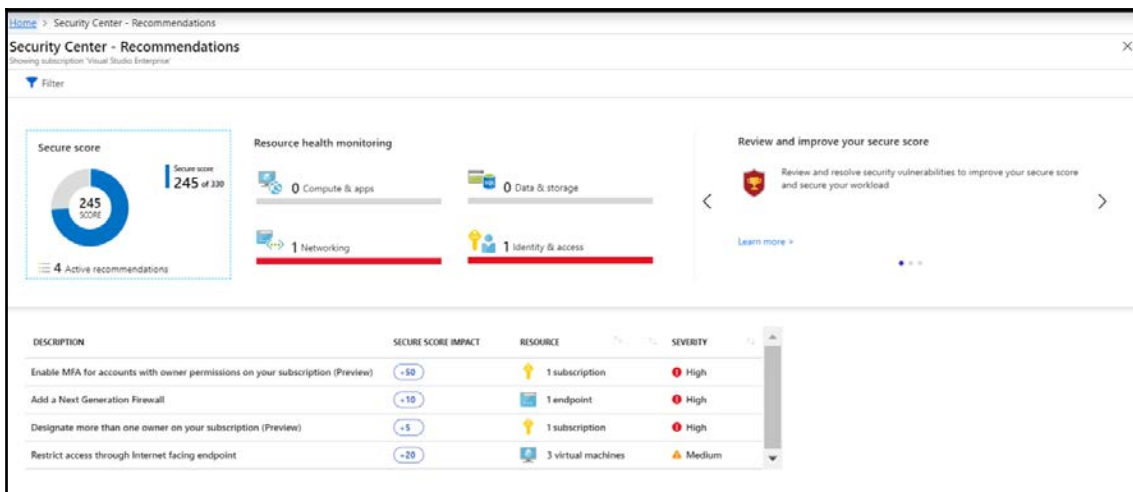
If you would like to start with security center, you will have the chance to test it for about 60 days before you have to pay:



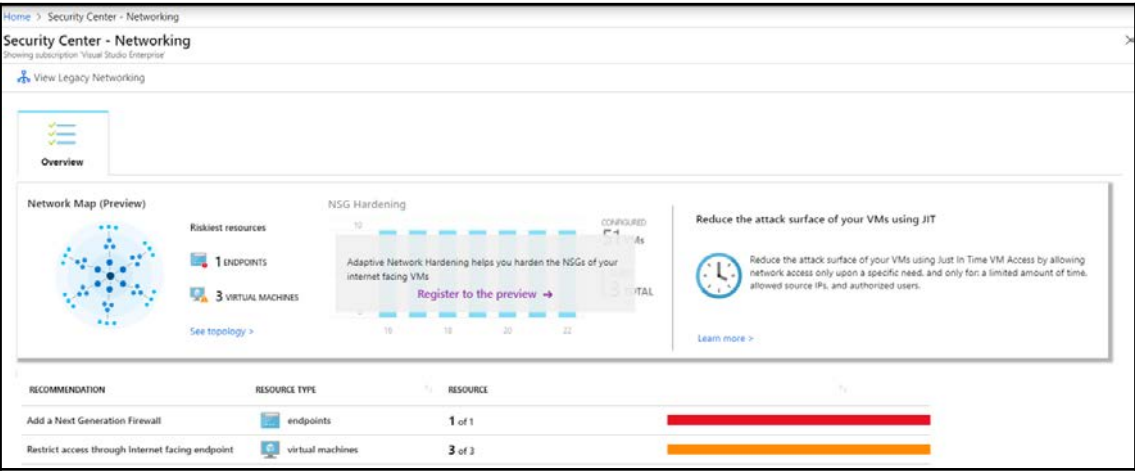
The **POLICY & COMPLIANCE** tab gives an overview of the state of the art for all enabled security features:



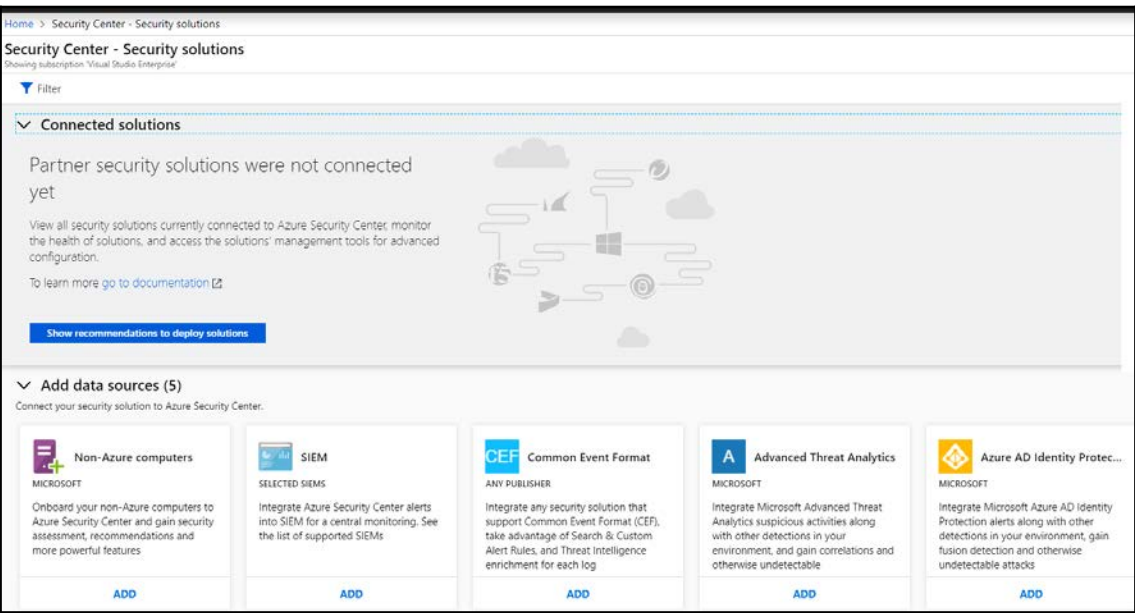
The **RECOMMENDATIONS** tab provides an overview of features that you should enable within your Azure environment. This is more or less the best practice analyzer for Azure Security. You can choose the following tabs to get a more detailed filtered view based on computer, networking, storage, identity, and security, as shown in the following screenshot:



As you can see in the **Networking** tab, there are recommendations for setting up firewall services or restricting resource access:



And finally, the **Connected solutions** and **Add data sources** tab can enrich the features that are enabled by default with additional features to have one view of security configurations in Azure:



Azure Cost Management

The second most important thing with running Azure services is to make sure they comply to the budgets set and make sure that the cheapest service that fits into the technical requirements is in use. Azure Cost Management could help you with that. Once enabled, it provides you an overview with drill-down feature for all services you are using within your Azure subscription:

The screenshot displays the Azure Cost Management + Billing portal. The left sidebar contains navigation links: Overview, Management groups, Cost Management, Diagnose and solve problems, Organization billing, Account overview, Cost analysis, Usage + charges, Credits, Individual billing, Account overview, Subscriptions, Invoices, Payment methods, Support + troubleshooting, and New support request. The main content area is titled 'Cost Management + Billing' and includes a search bar. It features two organization cards: 'Test Enrollment (Direct) (100)' and 'SAB BVT (DO NOT USE) (6328843)'. Below these are sections for 'My subscriptions' and 'Other subscriptions', each with a table of subscription details.

Organizations overview

Organization	CREDITS	CHARGES	ROLE	TYPE
Test Enrollment (Direct) (100)	0.00 USD	77,468.10 USD	Enterprise administrator	Billing account
SAB BVT (DO NOT USE) (6328843)	-	-	Enterprise administrator	Billing account

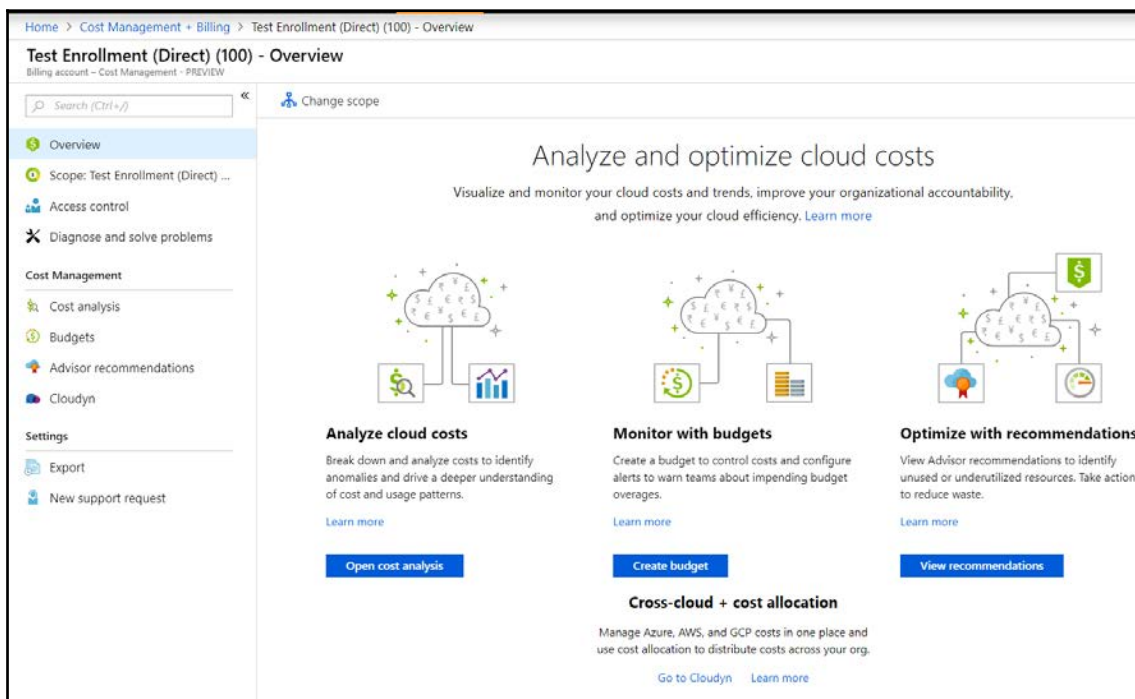
My subscriptions

NAME	SUBSCRIPTION ID	OFFER	STATUS	CURRENT COST
You don't have any subscriptions in the current directory. Don't see a subscription? Switch directories				

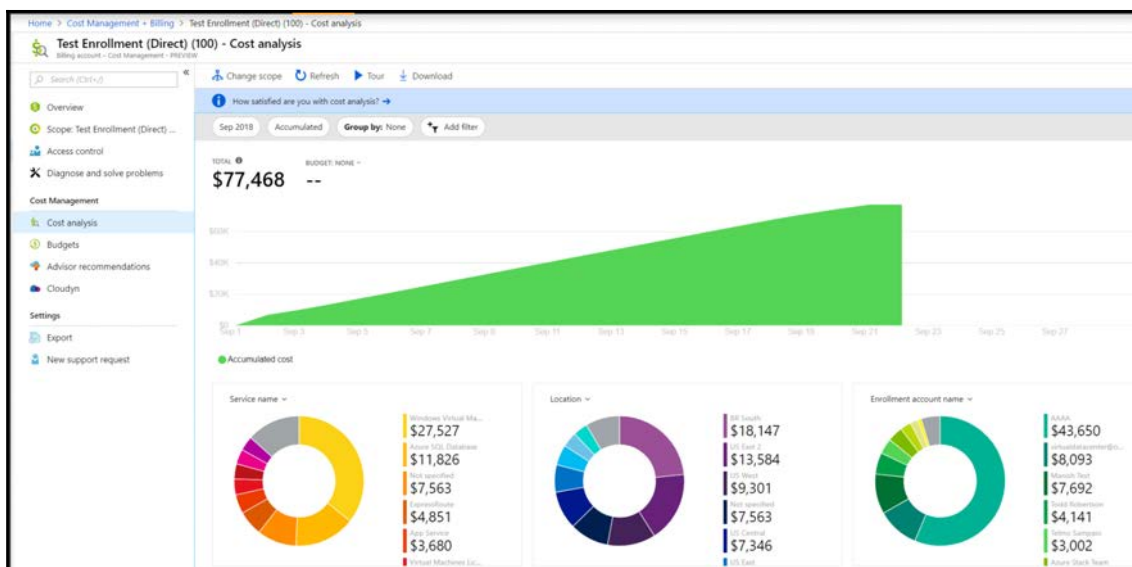
Other subscriptions

NAME	SUBSCRIPTION ID	OFFER	CURRENT COST
Contoso IT - demo	e4272367-5645-4c4e-9c67-3b74b59a6982	Microsoft Azure Internal Consumption	Not available
Trey Research Alpha Lab	586f1d47-9d89-43d5-b196-6e28f8405ff9	Enterprise Agreement	\$83.18
Cost Management Demo	1caaa5a3-2b66-438e-8ab4-bce37d518c5d	Enterprise Agreement	\$3,239.36
Trey Research Alpha Dev	64e355d7-997c-491d-b0c1-8414dcccfc42	Enterprise MSDN Dev/Test	\$1,078.82

In addition, it can analyze your existing services, monitor budgets, and give recommendations on how resources could be configured to be more cost-sensitive (for example, changing the VM type):



The cost analysis feature provides a great management overview on the costs and how they are split between the different services. Finally, you could even enable cost management for AWS or Google Cloud. If you enable this, Azure Cost Management is 1% of the revenue for these third-party clouds; for Azure it is cost-free:



Summary

This chapter gave a deep overview of all Azure services that support your Azure governance. You will need to define rules and then set them up in Azure to fulfill these governance rules. Azure governance is the most important task you will need to start with before starting consuming service. Once services are in place, defining the governance is never good, as existing working methods need to be changed, which mostly results in conflicts. Azure governance is always where to start from.

To summarize all options that are available with Azure, the following table could help:

Azure component	Used for	Used in Azure Blueprint templates
Subscriptions	Billing root	no
Resource groups	Organize resources	yes
Resource tags	Add properties (cost center, and so on)	Yes
Naming conventions	Naming definitions	Yes
Azure RBAC	Set permission roles	Yes
Azure policy	Define governance	Yes
Automation	Task automation	no

Questions

Please answer the following questions:

1. If we talk about Azure governance, what is meant by this?
2. What is the goal for Azure **role-based access control (RBAC)**?
3. How do you make sure that the written governance will be used by the contributors in Azure later on?
4. What are management groups in Azure used for?
5. Which tools could help to support Azure governance?
6. Why is it so important to have Azure governance quite early in your Azure project?
7. What is the difference between Azure Cost Management and Cloudyn?

Further reading

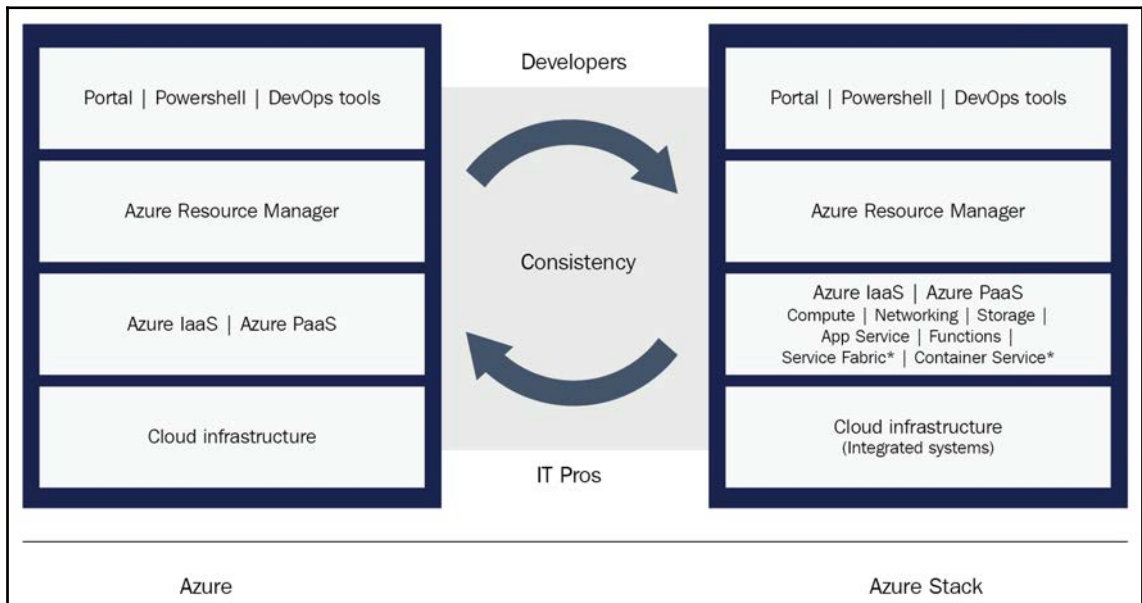
Read the following articles for more information:

- **Azure governance:** <https://azure.microsoft.com/en-us/solutions/governance/>
- **Azure Management - Governance:** <https://docs.microsoft.com/en-us/azure/governance/>
- **The Azure Governance Toolbox – Part 1:** <https://adinermie.com/the-azure-governance-toolbox-part-1-introduction/>

11

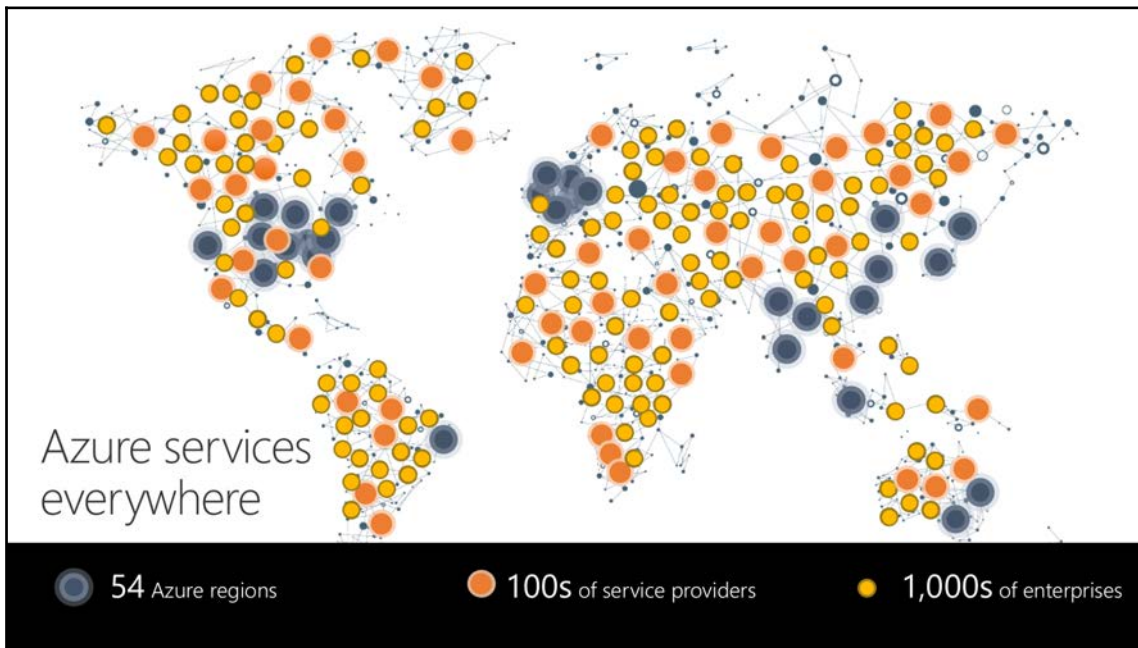
Azure Hybrid Data Center Services

Azure public cloud services provide cloud services all around the world, through a technology based on the following high-level design on the left-hand side. If there are scenarios where Azure does not fit the requirements of a customer and services need to be put on-premise, Microsoft has released a product called Azure Stack that has nearly the same technical design as Azure, but runs a smaller footprint. The high-level design for Azure Stack is as follows on the right-hand side of the following diagram:



As you can see, the only difference is the infrastructure design on the bottom; in Azure this is something special, based on Windows server technology and in Azure Stack it is Windows Server 2016.

The following diagram provides a way to place Azure (and Azure Stack) all around the world:



The following features are available in Azure Stack as of today:

Web and API apps	Serverless computing	Microservices platform	Container orchestration	Pivotal and Open source
				
Azure App Service	Azure Functions	Service Fabric	Kubernetes	Cloud Foundry OpenShift
				
Virtual Machines	Docker Containers	Networking	Storage	Key Vault
Linux and Windows (including VM scale sets)	Linux and Windows	Virtual network, load balancer, VPN gateway	Blobs, tables, queues	Application keys and secrets

In this chapter, we will discuss this hybrid technology a little bit more in depth.



If you want a more detailed reading on this, we would advise that you have a look at this specific book discussing only Azure Stack itself at <https://www.packtpub.com/virtualization-and-cloud/building-hybrid-clouds-azure-stack>.

The chapter will cover the following topics:

- The **Azure Stack Development Toolkit (ASDK)** versus multi-host deployments
- Setting up an ASDK
- Working with Azure Stack
- Monitoring Azure Stack
- Use cases for Azure Stack

Technical requirements

For working with Microsoft Azure Stack, no specific installations are required, as you only need:

- A computer with an internet browser
- An Azure subscription (if not available, a trial could work, too: <https://azure.microsoft.com/en-us/free/>)

Code in this chapter can be found here at <https://github.com/PacktPublishing/Implementing-Azure-Solutions-Second-Edition>.

ASDK

Microsoft Azure Stack can only be ordered with the corresponding hardware, which has been tested and certified. The following are the hardware vendors:

- HPE
- DELL EMC
- Lenovo
- Cisco
- Huawei
- Wortmann
- Fujitsu

The typical scenarios which Azure Stack could fit in are:

- Disconnected/Edge scenarios (for example, running services of Azure on a ship or a plane)
- Data privacy reasons (data could, should, or must not leave the company location)
- Modern cloud app development on-premise

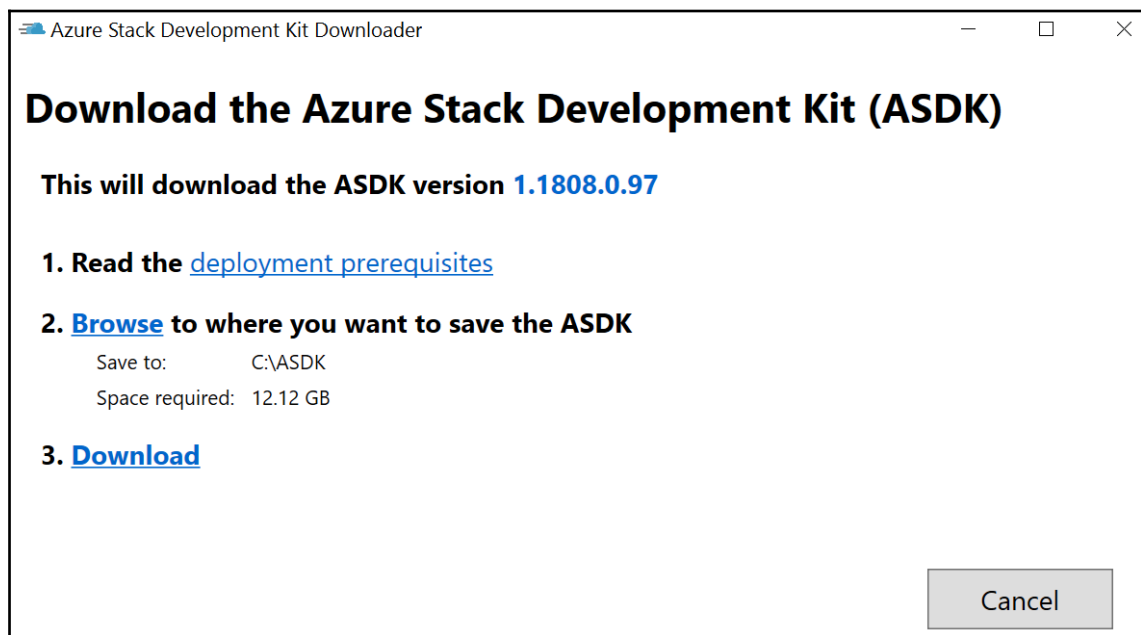
As Azure Stack can only be bought as an integrated system, there is no software available that could be downloaded and place it on dedicated hardware. But for testing **proof of concept (PoC)** purposes, Microsoft offers a trial version of Azure Stack, called **ASDK**. As it is running on one physical host, no performance or availability tests are suitable, but in general all features are available with ASDK, too.

The required hardware for an ASDK is documented here at <https://docs.microsoft.com/en-us/azure/azure-stack/asdk/asdk-deploy-considerations>. To finally double check if all hardware requirements are met, you can run the pre-requisite script from here at <https://go.microsoft.com/fwlink/?LinkId=828735&cid=0x409>.

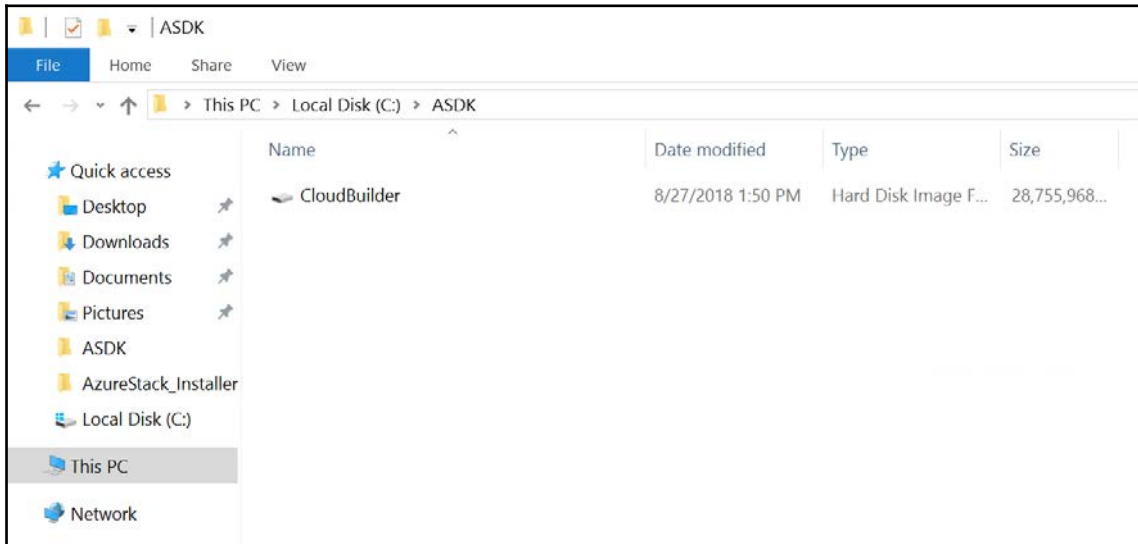
Preparing the ASDK host

The most recent download of the ASDK can be found here at <https://azure.microsoft.com/en-us/overview/azure-stack/development-kit/?v=try>. This is how to download the ASDK:

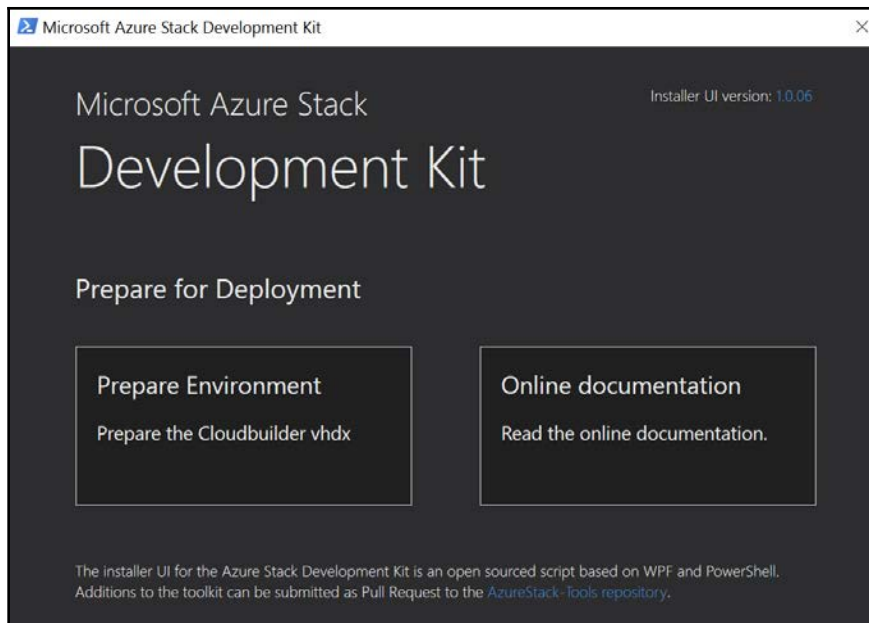
1. The Azure Stack Downloader will help you with a nice UI to make it work:



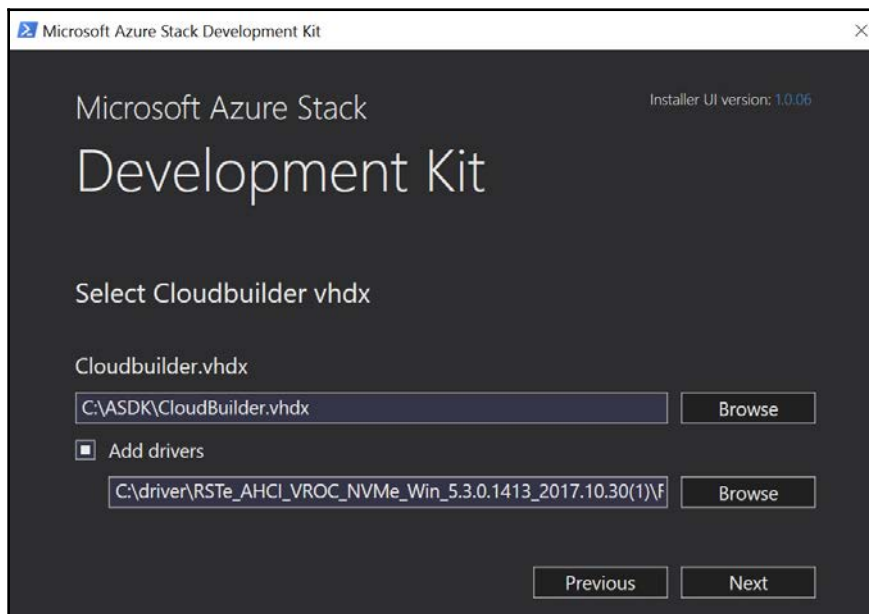
2. Now we will need to extract the files to a folder of your choice on the local hard disk:



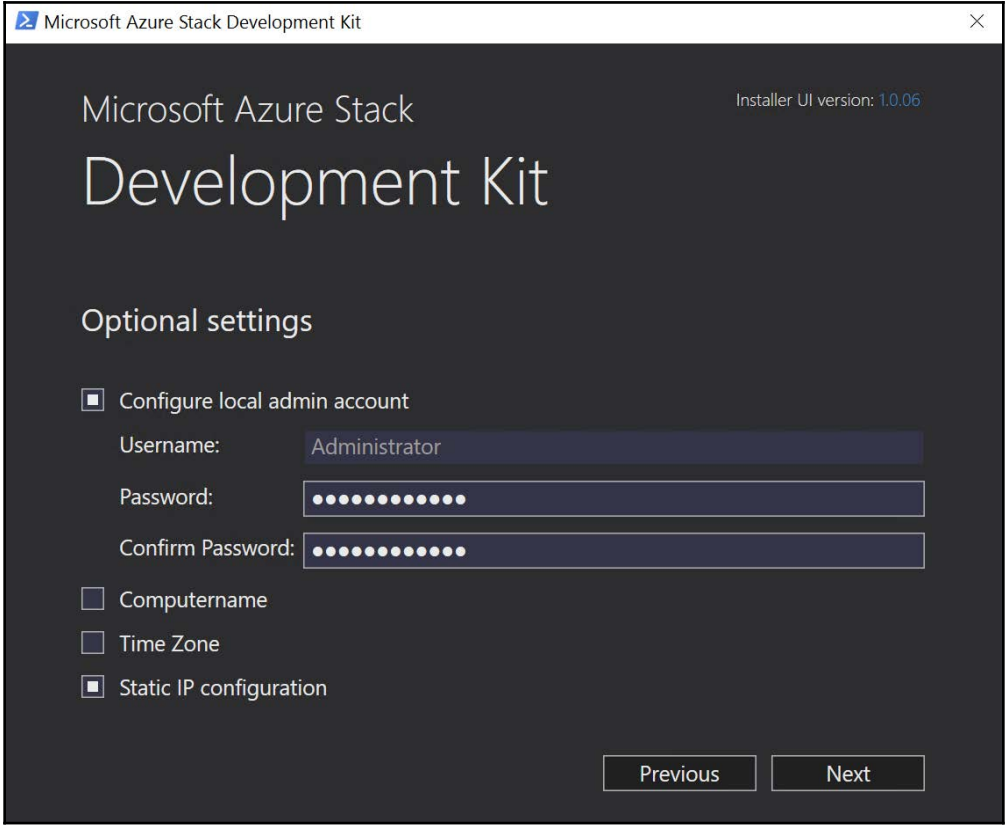
3. The `CloudBuilder.vhdx` is the source that all Azure Stack services boot from. You should place this `.vhdx` file in a destination folder of your choice and start the ASDK installer script, which is available here at <https://docs.microsoft.com/en-us/azure/azure-stack/asdk/asdk-install>.
4. Now let's start with choosing the option on the left and prepare the environment:



5. We will need to choose the correct virtual disk of Azure Stack and load corresponding drivers for the hardware, if needed:



6. The installer now mounts the .vhdx file and boots from it. The next required parameters are the local administrator account and password, which will be set identically for all accounts in Azure Stack. If you are running your ASDK host in a non-DHCP networking environment, you would need to define the networking properties and even the time zone and computer name:



The screenshot shows the 'Microsoft Azure Stack Development Kit' installer window. The title bar reads 'Microsoft Azure Stack Development Kit'. The main window has a dark background with the text 'Microsoft Azure Stack Development Kit' in white. In the top right corner, it says 'Installer UI version: 1.0.06'. Below the title, the section 'Optional settings' is displayed. There are three checkboxes: 'Configure local admin account' (checked), 'Computename' (unchecked), and 'Time Zone' (unchecked). Below the 'Configure local admin account' checkbox, there are three input fields: 'Username:' with the text 'Administrator', 'Password:' with masked characters, and 'Confirm Password:' with masked characters. At the bottom right, there are two buttons: 'Previous' and 'Next'.

Microsoft Azure Stack Development Kit

Installer UI version: 1.0.06

Microsoft Azure Stack Development Kit

Optional settings

☒ Configure local admin account

Username: Administrator

Password: ●●●●●●●●

Confirm Password: ●●●●●●●●

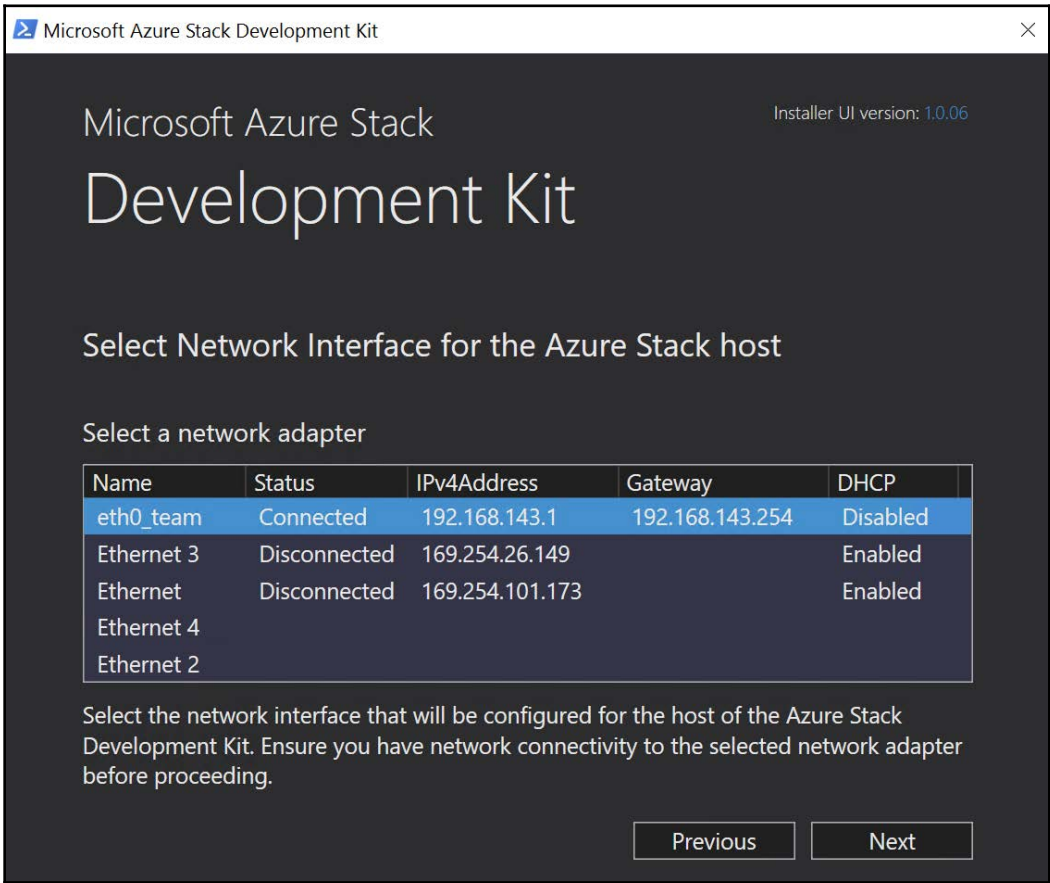
☐ Computename

☐ Time Zone

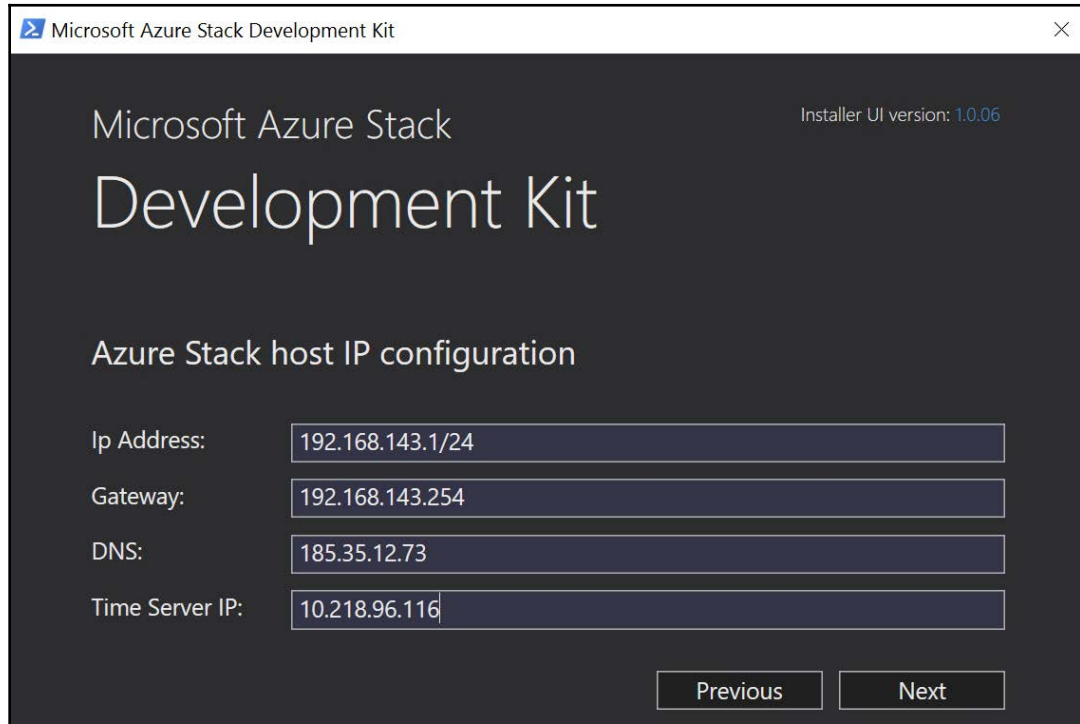
☒ Static IP configuration

Previous Next

- 7. Because ASDK only supports one network card, we will need to choose it. All others will be disabled by default, if they are not already:

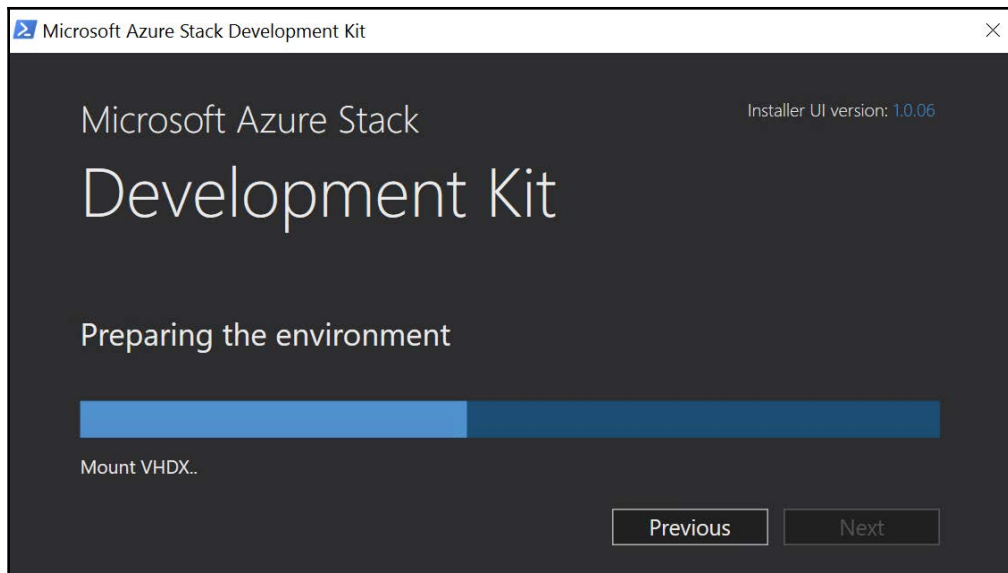


8. If you are deploying ASDK with a fixed IP, it is a must to set a proper time server, as about 80% of all broken ASDK installations are caused by time server issues. If you have no other one available, `time.windows.com` could help you:

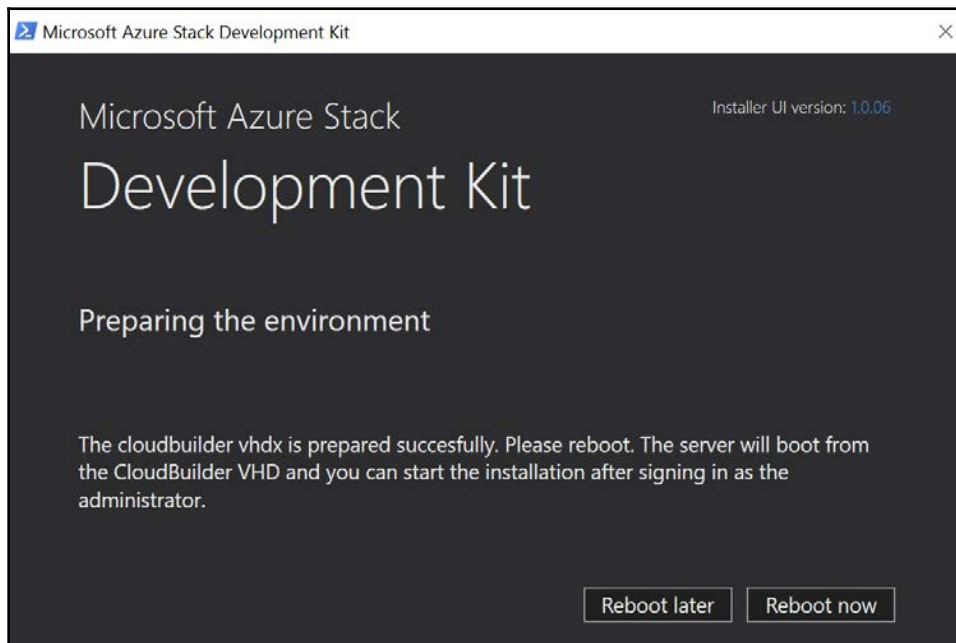


The screenshot shows the 'Microsoft Azure Stack Development Kit' installer window. The title bar reads 'Microsoft Azure Stack Development Kit'. The main window has a dark background with the text 'Microsoft Azure Stack Development Kit' and 'Installer UI version: 1.0.06'. Below this, the section 'Azure Stack host IP configuration' is displayed. It contains four input fields: 'Ip Address' with the value '192.168.143.1/24', 'Gateway' with '192.168.143.254', 'DNS' with '185.35.12.73', and 'Time Server IP' with '10.218.96.116'. At the bottom right, there are 'Previous' and 'Next' buttons.

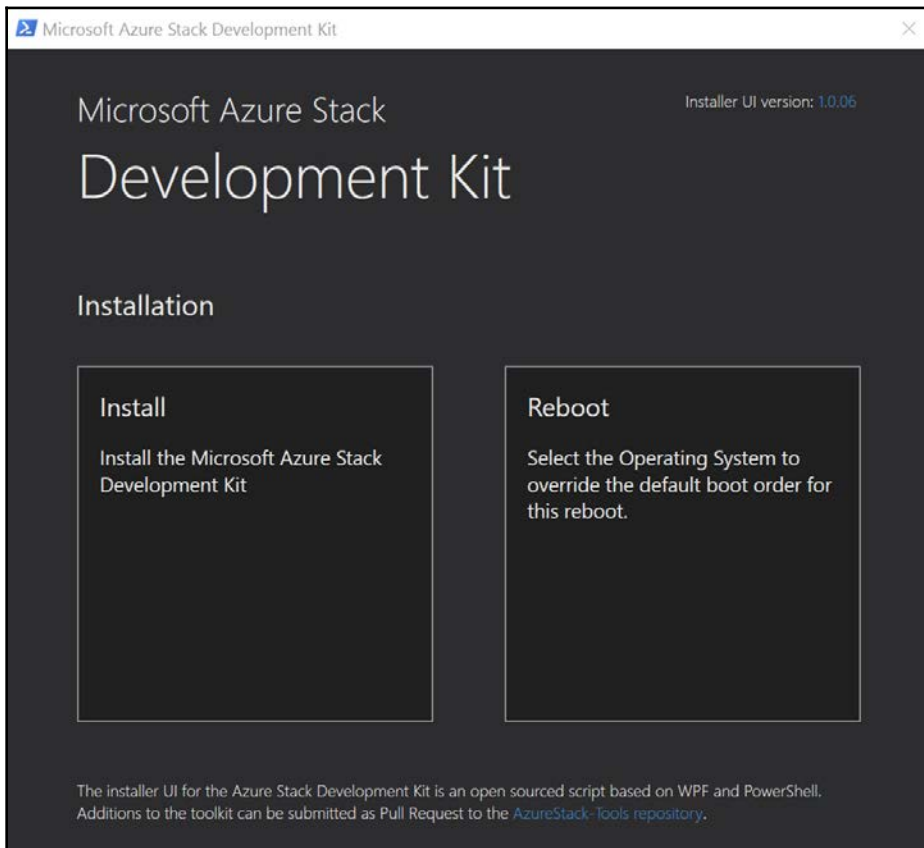
9. Now, the `CloudBuilder.vhdx` is being mounted and a final reboot will bring the system online in the VHDX itself:



10. After the reboot is successfully done, we will need to run the installer again to get to the second phase of the installation:

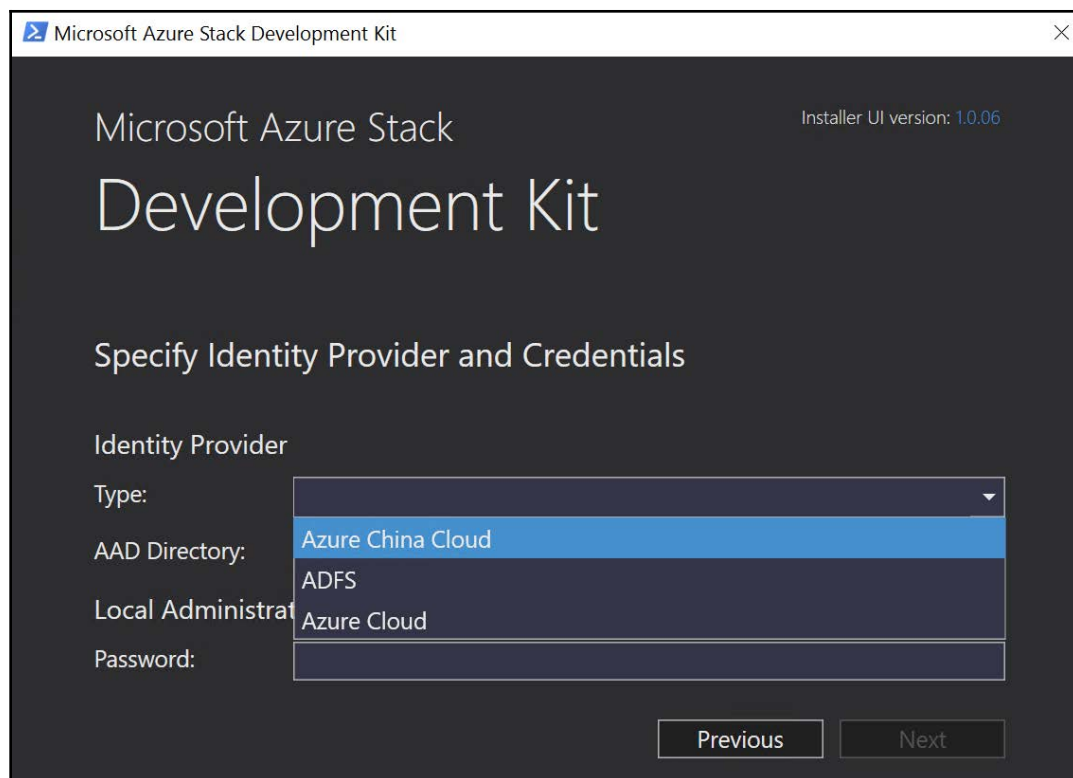


11. Therefore, we will need to hit the **Install** box on the left-side of the following screenshot:



Identity management configuration

During the installation, the appropriate identity management solution (also called **connected mode** or **disconnected mode**) needs to be selected:



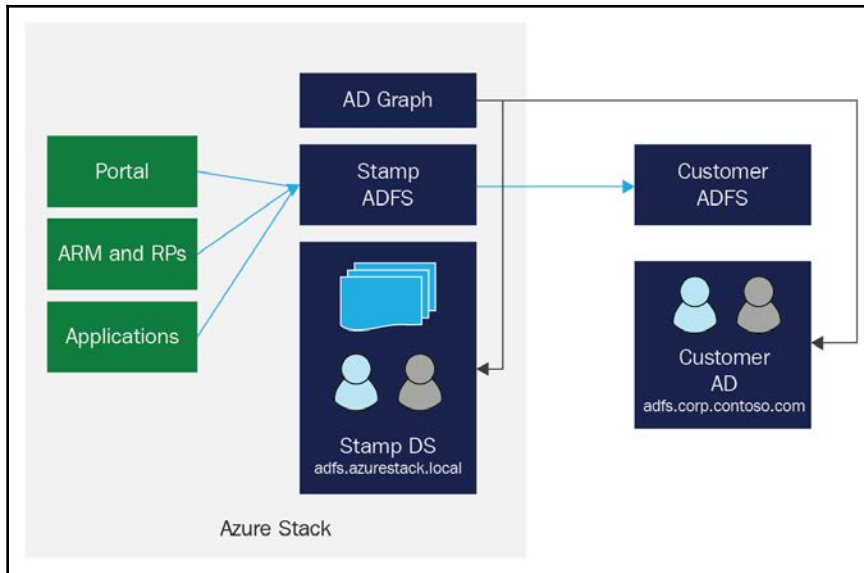
As you can see in the preceding screenshot, Azure Stack is supported to run in:

- **Azure China Cloud**
- **Azure Cloud**
- **ADFS** (although called disconnected mode)

This means that if you run it in connected mode (using Azure Cloud or Azure China Cloud), the identity management is moved to Azure AD. To enable this, you will need to have proper permission (global administrator) in Azure AD. This is because the registration of applications needs to be done during the setup. The following table gives an overview of the tasks in detail:

When	Action	AAD account requirements	App registrations created	SPNs created
Install Azure Stack	Create App registrations and service principals	AAD account with Global Admin role	- Azure Pack Connector - Azure Stack - Azure Stack - Administration - Azure Stack - Bridge - Azure Stack - Compute - Azure Stack - Deployment - Azure Stack - Hubs - Azure Stack - Hubs Administration - Azure Stack - KeyVault - Azure Stack - Monitoring - Azure Stack - Monitoring Administration - Azure Stack - Policy - Azure Stack - Policy Administration - Azure Stack - Portal - Azure Stack - Portal Administration - Azure Stack - RBAC - Azure Stack - RBAC Administration - AzureStack KeyVault Internal	- Azure Stack - Azure Stack - Administration - Azure Stack - Bridge - Azure Stack - Compute - Azure Stack - Deployment - Azure Stack - Hubs - Azure Stack - Hubs Administration - Azure Stack - KeyVault - Azure Stack - Monitoring - Azure Stack - Monitoring Administration - Azure Stack - Policy - Azure Stack - Policy Administration - Azure Stack - Portal - Azure Stack - Portal Administration - Azure Stack - RBAC - Azure Stack - RBAC Administration - AzureStack KeyVault Internal - AzureStack Monitoring Service
Register Azure Stack	Create - A service principal to perform resource actions - A resource group - A registration resource in the created resource group in Azure	For the resource group: subscription owner. For the service principal registration: If "Users can register applications" setting is enabled the any AAD member (without Global Admin role) can create app registrations. Otherwise an Azure AD global admin role is required.	Azure Bridge	Azure Bridge
Install MySQL RP	-	-	-	-
Install SQL RP	-	-	-	-
Install App Service RP	Create App registrations and service principals	AAD account with Global Admin role	App Service	App Service

After the specific registrations have been done, there is no need to have the global administrator permissions anymore. If Azure Stack is using disconnected mode, it relies on ADFS and creates the federation trust during the deployment. This means that ADFS has to be already in place before starting the setup:



For the Microsoft Graph configuration, a service account with read permission in the existing AD needs to be available. The requirements for AD and ADFS are Windows Server 2012, Active Directory 2012, and above.

The technical differences between both modes regarding available services are described in the following table:

	Disconnected from the internet	Connected to the internet
Billing	Must be Capacity Enterprise Agreement (EA) only	Capacity or Pay-as-you-use EA or Cloud Solution Provider (CSP)
Identity	Must be AD FS	Azure AD or AD FS
Marketplace	Supported BYOL licensing	Supported BYOL licensing
Registration	Recommended, requires removable media and a separate connected device.	Automated
Patch and update	Required, requires removable media and a separate connected device.	Update package can be downloaded directly from the Internet to Azure Stack.

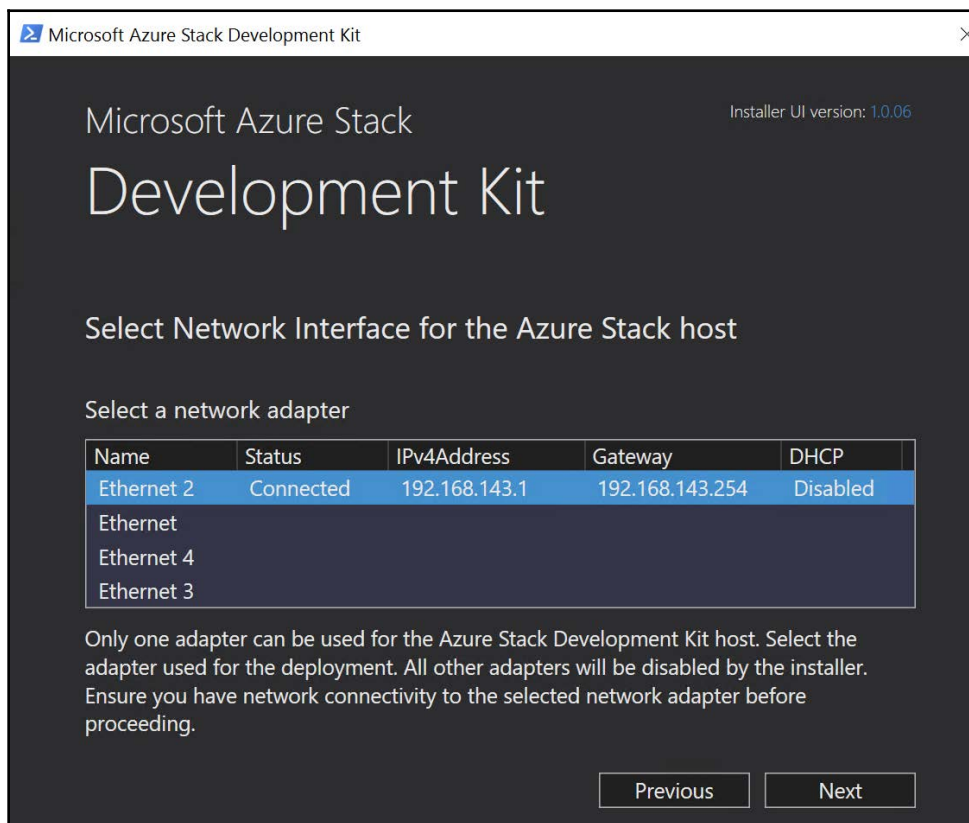


For more details regarding identity configuration, the following URL could help at <https://docs.microsoft.com/en-us/azure/azure-stack/azure-stack-integrate-identity>.

Networking configuration

The next important step during the setup is the networking configuration. With the ASDK, all networking relies on a VM called BGP NAT, which simulates the networking switch in the multi-node deployments. As all outgoing or incoming traffic passes this VM, it also represents the bottleneck of ASDK. As only one network interface card is supported, all other available ones need to be disabled, which will be done during the setup itself. This is how to set up:

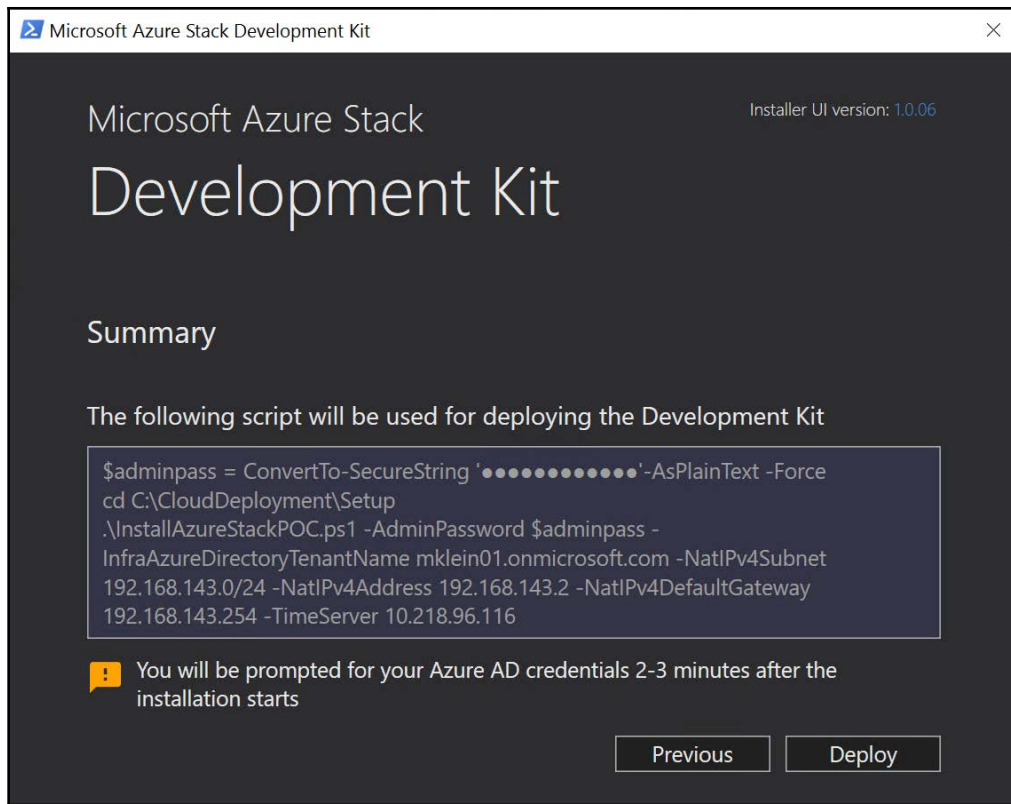
1. In a multi-node environment, the BGP NAT VM is replaced with a physical top of the rack switch:



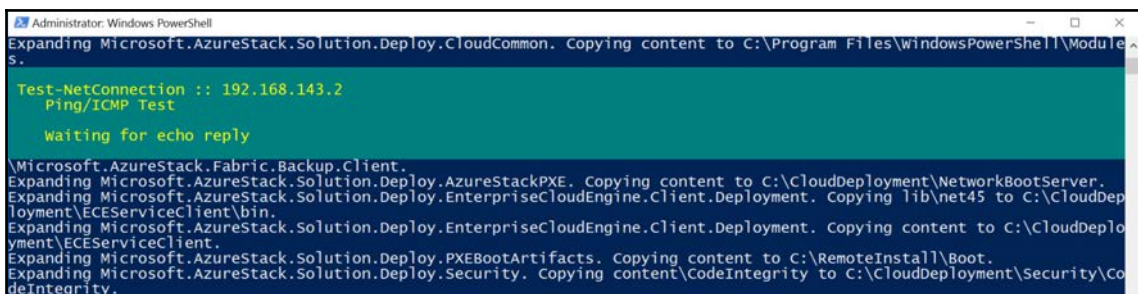
2. For the ASDK, we will need to set up the BGP NAT IP and all other required TCP/IP parameters as shown in the following screenshot:

The screenshot shows the 'Microsoft Azure Stack Development Kit' installer window. The title bar reads 'Microsoft Azure Stack Development Kit'. The main window has a dark background with the text 'Microsoft Azure Stack Development Kit' and 'Installer UI version: 1.0.06'. Below this, the section 'Network Configuration for BGP NAT01' is displayed. There are two radio buttons for network configuration: 'DHCP' (selected) and 'Static'. Below the radio buttons, there are three text input fields: 'Ip Address:' with the value '192.168.143.2/24', 'Gateway:' with the value '192.168.143.254', and 'Time Server IP:' with the value '51.141.32.51'. Below these fields, the section 'Optional Configuration' is displayed, with two text input fields: 'VLAN ID:' and 'DNS Forwarder IP:'. At the bottom right, there are two buttons: 'Previous' and 'Next'.

3. With these parameters filled in, all required information is available and the setup is ready to start:



4. After having hit the **Deploy** button, the setup continues. If you have chosen the connected mode, within the next few minutes an authentication request for AAD will be displayed. If no AAD is in place, no further input is needed:



- Depending on the performance of the physical host, the setup may run for four to five hours or more:

```
Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
VERBOSE: 3> [SQL:Deployment] AzS-Sql01 has finished OS deployment, but is still processing
SetupComplete. - 9/9/2018 11:33:31 AM

Action 'Deployment' 65% Running action plan...
Step 40 - Phase 1 - CreateVMS 0% Management VMS.
Task Cloud - Deployment-Phase1-CreateVMS Running action
Action 'Deployment-Phase1-CreateVMS' 0% Running action plan...
Step 40.41 - (CPI) Create guest VMS 0% Create the management VMS.
Task Cloud\Fabric\ADFS - Deployment Running interface
Task Cloud\Fabric\WAS - Deployment Running interface
Task Cloud\Fabric\FabricRingServices - Deployment Running interface
Task Cloud\Fabric\ACS - Deployment Running interface
Task Cloud\Fabric\SeedRingServices - Deployment Running interface
Task Cloud\Fabric\SQL - Deployment
Running interface
Task Cloud\Fabric\WASPUBLIC - Deployment
Running interface

VERBOSE: 2> [ADFS:Deployment] Testing for presence of
C:\CompleteBootDSCStatus\AzS-ADFS01.1.1808.0.97.xml on WIN-1V1RV61ISN3 - 9/9/2018 11:33:58 AM
VERBOSE: 2> [ADFS:Deployment] AzS-ADFS01 is still being deployed. It will be reachable once OS
deployment is complete and execution of SetupComplete script has ended. - 9/9/2018 11:33:58 AM
VERBOSE: 4> [WAS:Deployment] Testing for presence of
C:\CompleteBootDSCStatus\AzS-WAS01.1.1808.0.97.xml on WIN-1V1RV61ISN3 - 9/9/2018 11:33:58 AM
VERBOSE: 4> [WAS:Deployment] AzS-WAS01 is still being deployed. It will be reachable once OS
deployment is complete and execution of SetupComplete script has ended. - 9/9/2018 11:33:58 AM
VERBOSE: 3> [SQL:Deployment] Testing for presence of
C:\CompleteBootDSCStatus\AzS-Sql01.1.1808.0.97.xml on WIN-1V1RV61ISN3 - 9/9/2018 11:34:01 AM
VERBOSE: 3> [SQL:Deployment] AzS-Sql01 has finished OS deployment, but is still processing
SetupComplete. - 9/9/2018 11:34:01 AM
VERBOSE: 7> [SeedRingServices:Deployment] Testing for presence of
C:\CompleteBootDSCStatus\AzS-ERCS01.1.1808.0.97.xml on WIN-1V1RV61ISN3 - 9/9/2018 11:34:14 AM
VERBOSE: 7> [SeedRingServices:Deployment] AzS-ERCS01 has finished OS deployment, but is still
processing SetupComplete. - 9/9/2018 11:34:14 AM
VERBOSE: 1> [ACS:Deployment] Testing for presence of
C:\CompleteBootDSCStatus\AzS-ACS01.1.1808.0.97.xml on WIN-1V1RV61ISN3 - 9/9/2018 11:34:14 AM
VERBOSE: 1> [ACS:Deployment] AzS-ACS01 is still being deployed. It will be reachable once OS
deployment is complete and execution of SetupComplete script has ended. - 9/9/2018 11:34:14 AM
VERBOSE: 5> [WASPUBLIC:Deployment] Testing for presence of
C:\CompleteBootDSCStatus\AzS-WASP01.1.1808.0.97.xml on WIN-1V1RV61ISN3 - 9/9/2018 11:34:17 AM
```

6. During the first steps of the deployment phase, a Windows cluster will be set up while the ASDK host is rebooting. After this step, it is important to log on as Azurestack\AzureStackAdmin to double check the installation progress:

```
===== ValidateUpdate =====
MpsigStub successfully updated Microsoft Windows Defender (RS1+) using the Platform update
package.

      Original:      Updated to:
Platform: 4.10.14393.2273 4.18.1807.18075
End time: 2018-09-09 21:05:53Z
-----
- 9/9/2018 2:05:53 PM
VERBOSE: [Cloud:UpdateCloudSecurity] Done applying Windows Defender Platform update to individual
nodes - 9/9/2018 2:05:53 PM
VERBOSE: [Cloud:UpdateCloudSecurity] Done applying Windows Defender Platform update - 9/9/2018
2:05:55 PM
VERBOSE: [Cloud:UpdateCloudSecurity] Done updating Common Antimalware Platform (Windows Defender).
- 9/9/2018 2:05:55 PM
VERBOSE: Interface: Interface UpdateCloudSecurity completed. - 9/9/2018 2:05:55 PM
COMPLETE: Task Cloud - UpdateCloudSecurity
VERBOSE: Task: Task completed. - 9/9/2018 2:05:55 PM
COMPLETE: Step 305 - (SEC) Finalize cloud security
VERBOSE: Step: Status of step '305 - (SEC) Finalize cloud security' is 'Success'. - 9/9/2018
2:05:55 PM
VERBOSE: Checking if any of the in progress steps are complete. The following steps are currently
in progress: '305'. - 9/9/2018 2:05:55 PM
VERBOSE: Action: Step 305 completed successfully. - 9/9/2018 2:05:55 PM
VERBOSE: The following steps have completed and will be removed from the collection of in-progress
steps: '305'. - 9/9/2018 2:05:55 PM
VERBOSE: Action plan execution completed for action plan 'Deployment'. - 9/9/2018 2:05:55 PM
VERBOSE: Action: Action plan 'Deployment' completed. - 9/9/2018 2:05:55 PM
COMPLETE: Action 'Deployment'
PS C:\Windows\system32>
```

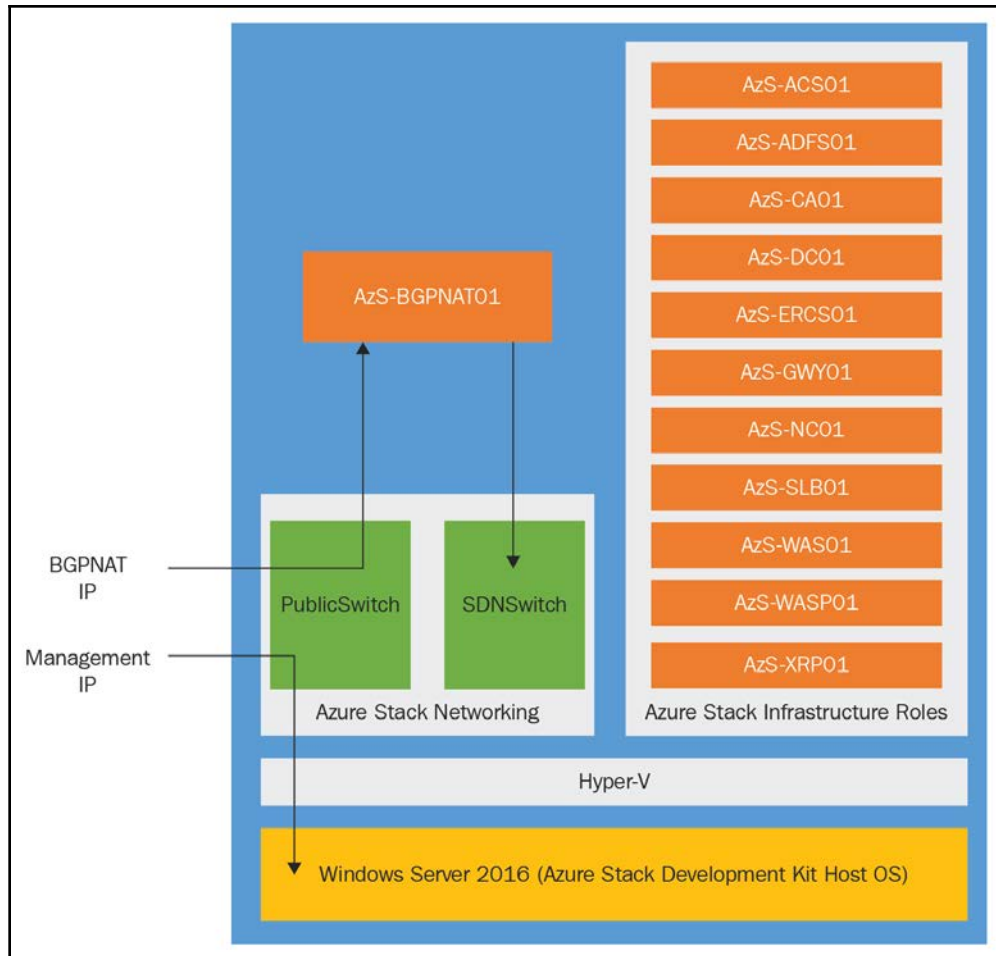
7. If no errors occur, the setup will be finished with the preceding screen. If there are any issues, the installation log files (saved to `.\CloudDeployment\Logs`) are the single point of truth. You can collect them using the following PowerShell command:

```
Get-AzureStackLog -OutputPath C:\AzureStackLogs (or any other
folder)
```

The first and easiest troubleshooting step is to run the setup again using the `/rerun` parameter.

VM design of Azure Stack (ASDK)

After Azure Stack ASDK has been set up properly, the following virtual machines are available on the host:



For the multi-node environments, these VMs are available at minimum redundantly with load balancers in front, but as these are secured by default, you would never see the VMs from the operational perspective.

Azure Stack configuration task

After the installation is finished, there are more tasks before you could call Azure Stack ready for the customers. These include the following:

- Installing Azure Stack PowerShell and AzureRM modules
- Installing Azure Stack tools
- Registering ASDK to Azure
- Adding Microsoft VM Extensions to gallery from marketplace
- MySQL and SQL resource provider installation
- Adding SQL Server and MySQL hosting servers
- App service installation and configuration (certificates generation)
- Setting default quotas and a base plan and offer that contain all deployed services
- Configuring Python and Azure CLI for usage with ASDK

You could either run all these tasks manually (per the description available within the Microsoft Documentation, accessible here at <https://github.com/mattmcspirit/azurestack>) or using a script that has been created by Matt McSpirit.

As it makes configurations quite easy, I would suggest using it every time following the Azure Stack update cycle. It runs for another six to seven hours and a sample should look as follows:

```
.\ConfigASDK.ps1 -azureDirectoryTenantName "contoso.onmicrosoft.com" -  
authenticationType AzureAD -downloadPath "D:\ASDKfiles" -ISOPath  
"D:\WS2016EVALISO.iso" -azureStackAdminPwd 'YourPassword' -VMpwd  
'YourPassword' -azureAdUsername "admin@contoso.onmicrosoft.com" -azureAdPwd  
'YourAADPassword' -registerASDK -useAzureCredsForRegistration -  
azureRegSubId "YourAzureSubscriptionID"
```

If you run into issues while the script is running, you should correct the setup and simply rerun it.

This should be the last step to finally start with Azure Stack testing.

Operating Azure Stack

In this chapter so far, we have discussed what Azure Stack is and how to install the ASDK. Now it is time to have a look at Azure Stack and see how it works.

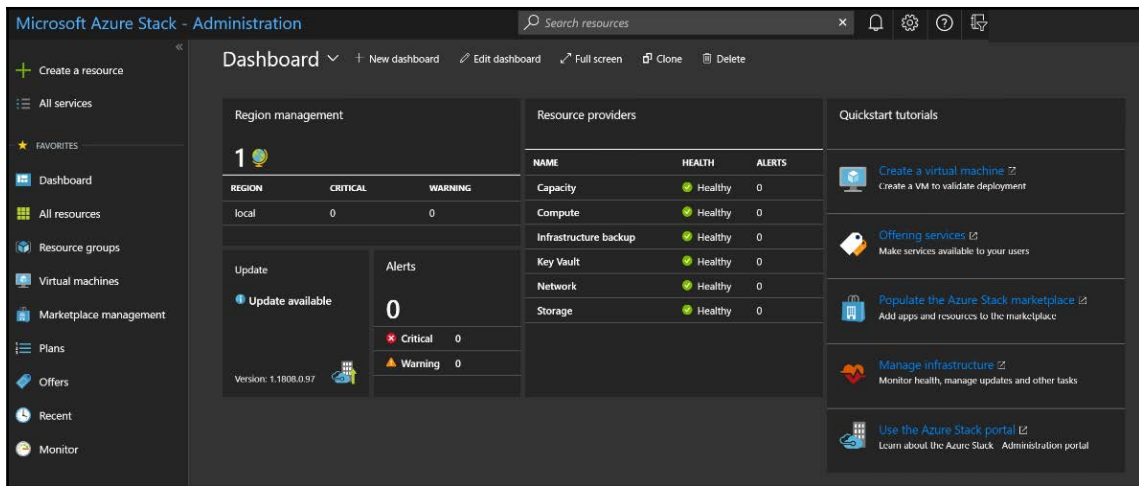
Basically, there are three ways to connect to Azure Stack endpoints and work with them from the operations side:

- Admin portal and tenant portal
- PowerShell
- The Azure (Stack) CLI

Working with the portals

With Azure Stack, you will get an administrative portal, which is lacking with Microsoft Public Azure, as there Microsoft is running Azure. Azure Stack will be operated by your operations team. Therefore, Azure Stack provides an administrative portal that is accessible through the following URL (using ASDK, as with the multi-node environment, is something you have to decide during the setup): `https://adminportal.local.azurestack.external`.

Depending on which scenario Azure Stack has been set up with (connected or disconnected mode), you either have to authenticate against **Azure AD** or as **Cloudadmin**:



This portal is used to fulfil the following tasks:

- Manage the infrastructure (including system health, updates, capacity, and so on)
- Populate the marketplace
- Create user subscriptions
- Create plans and offers

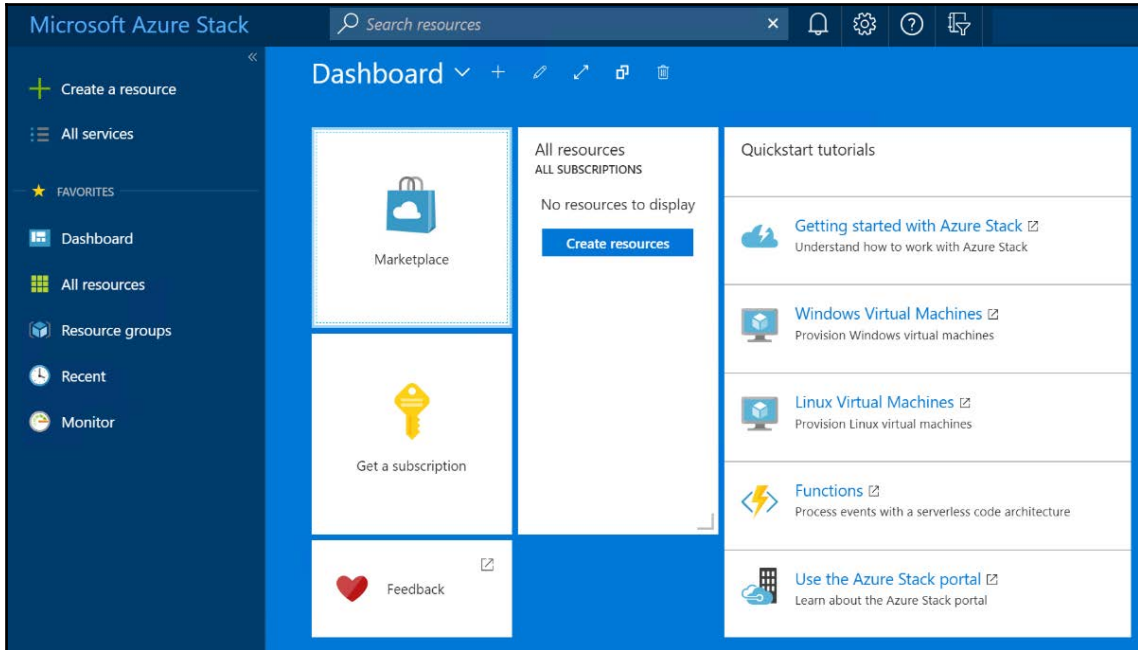
In Azure Stack, a plan is a way to group one or more services into a basic service plan; for example, you could create a plan that includes the compute, network, and storage resource providers. It gives a subscriber the ability to provision virtual machines by then. Resources can be budgeted using **quotas**. A quota can exist per resource.

An **offer** is a group of one or more plans to be presented to a customer.

After having created plans and offers, a user would have to **subscribe** to them. This could be done in two different ways:

- The cloud operator can create the subscription for a user in the administrative portal
- A tenant can subscribe to a public offer using the tenant portal

Now, let's have a look at the tenant portal, which is accessible using the following URL <https://portal.local.azurestack.external/> (in ASDK; in multi-node environments it could be different). This is a screenshot of the tenant portal:



The tenant portal will be used to create resources (virtual machines or PaaS solutions) as a self-service portal.

Working with PowerShell

If your work is not GUI-based, PowerShell could be your friend. You can set up your computer by installing Azure Stack PowerShell upon Azure PowerShell by running the following commands:

1. Install Azure PowerShell (in PowerShell evaluated admin mode):

```
install-module AzureRM
```

2. Install Azure Stack PowerShell (in PowerShell evaluated admin mode):

```
Import-Module -Name PowerShellGet -ErrorAction Stop
Import-Module -Name PackageManagement -ErrorAction Stop
Get-PSRepository -Name "PSGallery"
Register-PSRepository -Default
Set-PSRepository -Name "PSGallery" -InstallationPolicy Trusted
# Install the AzureRM.Bootstrapper module. Select Yes when prompted
to install NuGet
Install-Module -Name AzureRM.BootStrapper
# Install and import the API Version Profile required by Azure
Stack into the current PowerShell session.
Use-AzureRmProfile -Profile 2018-03-01-hybrid -Force
Install-Module -Name AzureStack -RequiredVersion 1.5.0
```

As an example, let's have a look at how to create a storage resource using PowerShell:

```
# Create variables to store the storage account name and the storage
account SKU information
$StorageAccountName = "mystorageaccount"
$SkuName = "Standard_LRS"

# Create a new storage account
$StorageAccount = New-AzureRMStorageAccount `
    -Location $location `
    -ResourceGroupName $ResourceGroupName `
    -Type $SkuName `
    -Name $StorageAccountName
Set-AzureRmCurrentStorageAccount `
    -StorageAccountName $storageAccountName `
    -ResourceGroupName $resourceGroupName

# Create a storage container to store a virtual machine image
$containerName = 'osdisks'
$container = New-AzureStorageContainer `
    -Name $containerName `
    -Permission Blob
```

Everything is possible through PowerShell, so the following link may help to dive deeper into Azure Stack PowerShell: <https://docs.microsoft.com/en-us/azure/azure-stack/user/azure-stack-quick-create-vm-windows-powershell>.

Working with the CLI

A third option to operate Azure Stack is the **Command Line Interface (CLI)**, which is available for:

- Windows
- Linux
- Mac



You can install them using the URL at <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest>.

Now, let's have a look at how to create a virtual machine using the CLI. At first, we would need to register the corresponding environment:

- As administrative environment, you would need to run the following script:

```
az cloud register \
  -n AzureStackAdmin \
  --endpoint-resource-manager
  "https://adminmanagement.local.azurestack.external" \
  --suffix-storage-endpoint "local.azurestack.external" \
  --suffix-keyvault-dns ".adminvault.local.azurestack.external"
\
  --endpoint-vm-image-alias-doc <URI of the document which
  contains virtual machine image aliases>
az cloud set \
  -n AzureStackAdmin
```

- As user environment, the following script could help:

```
az cloud register \
  -n AzureStackUser \
  --endpoint-resource-manager
  "https://management.local.azurestack.external" \
  --suffix-storage-endpoint "local.azurestack.external" \
  --suffix-keyvault-dns ".vault.local.azurestack.external" \
  --endpoint-vm-image-alias-doc <URI of the document which
  contains virtual machine image aliases>
az cloud set \
  -n AzureStackUser
```

- Now, let's log on to the Azure Stack environment with the following command:

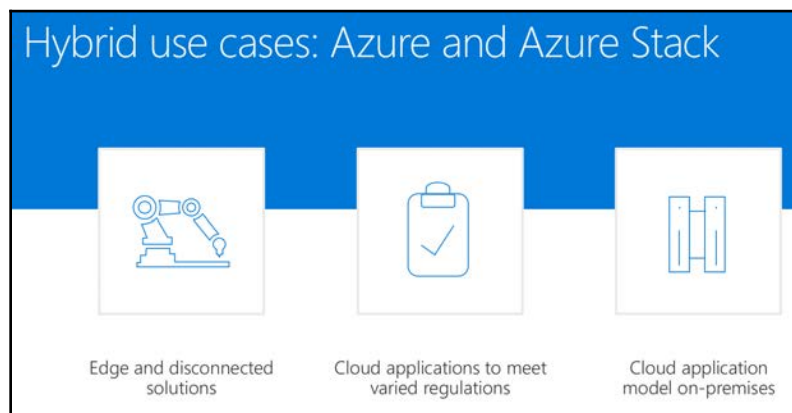
```
az login \  
-u <Active directory global administrator or user account.  
For example: username@<aadtenant>.onmicrosoft.com> \  
--tenant <Azure Active Directory Tenant name. For example:  
myazurestack.onmicrosoft.com>
```

- And finally, let's create a virtual machine:

```
az group create --name myResourceGroup --location local  
az vm create \  
--resource-group "myResourceGroup" \  
--name "myVM" \  
--image "Win2016Datacenter" \  
--admin-username "Demouser" \  
--admin-password "Demouser@123" \  
--use-unmanaged-disk \  
--location local
```

Hybrid cloud patterns

Now that we have an overview of what Azure Stack is, how it works, and how we could work with it, let's go on and discuss some scenarios where Azure Stack could fit. Here are some examples:



Basically, you should in general think about public Azure and how to move your IT services to it. If there is a reason why a service does not fit public Azure, Azure Stack might work.

The general scenarios for Azure Stack are as follows:

- **Edge and disconnected solutions:** Edge and disconnected solutions are everything when not much internet connectivity is available or not intended. Therefore, the disconnected mode is suitable, because with Azure AD an internet connection is really needed.
- **Data sovereignty:** Data sovereignty is the most important reason, as if a customer is not able to, not willing to, or does not have to move specific data to the public cloud, Azure Stack fits completely, as stack lives on the premise, where the data needs to reside.
- **Cloud applications:** Developing cloud applications is often something that needs to be done on the premises, therefore Azure Stack is the one and only option. After a major release or update is available, the application could then be moved from development stage to public Azure.

To make the corresponding use cases for Azure Stack a little bit easier to understand, Microsoft has published some use cases that might fit and has described them in a little bit more detail. These are the following:

- Configure hybrid cloud connectivity
- Machine learning solution with Azure Stack
- Azure Stack staged data analysis
- Azure Stack cloud burst scenario
- Azure Stack geo-distributed applications

While working through these scenarios, you will figure out that there is a step-by-step guide available to easily understand and set up these scenarios.

Let's have a little bit of a deeper look into each of them.

Configure hybrid cloud connectivity

Azure Stack often needs to be integrated into public Azure or other on-premise infrastructure setups. This is where a site-to-site VPN connection needs to be in place to connect the different infrastructure services. As Azure Stack nowadays is working with a shared VPN device for all customers on it, you will need to set up the site-to-site connection between some devices on premise, in Azure, or even using a network virtual appliance from a third party vendor to set up the connectivity needed.



For more information, please refer to: <https://docs.microsoft.com/en-us/azure/azure-stack/user/azure-stack-solution-hybrid-identity>.

Machine learning solution with Azure Stack

This solution is suitable if your company is using a DevOps approach and set up a CI/CD pipeline across stack and public Azure.



For more information, please refer to: <https://docs.microsoft.com/en-us/azure/azure-stack/user/azure-stack-solution-machine-learning>.

Azure stack staged data analysis

This solution could be useful to learn how to set up cross-cloud environments to meet the demand of multi-facility enterprises, as the solution describes a way to easily collect data as a basis for quick decisions with no internet access.

Learn how to use both on-premises and public cloud environments to meet the demands.



For more information, please refer to: <https://docs.microsoft.com/en-us/azure/azure-stack/user/azure-stack-solution-staged-data-analytics>.

Azure Stack cloud burst scenario

This scenario describes an environment where the customer is running its application (for example, a web application) on Azure Stack on-premise and, based on an automatic or manually triggered process (for example, excessive load on the solution), the system auto scales to public Azure through traffic manager to ensure flexibility and scalability.



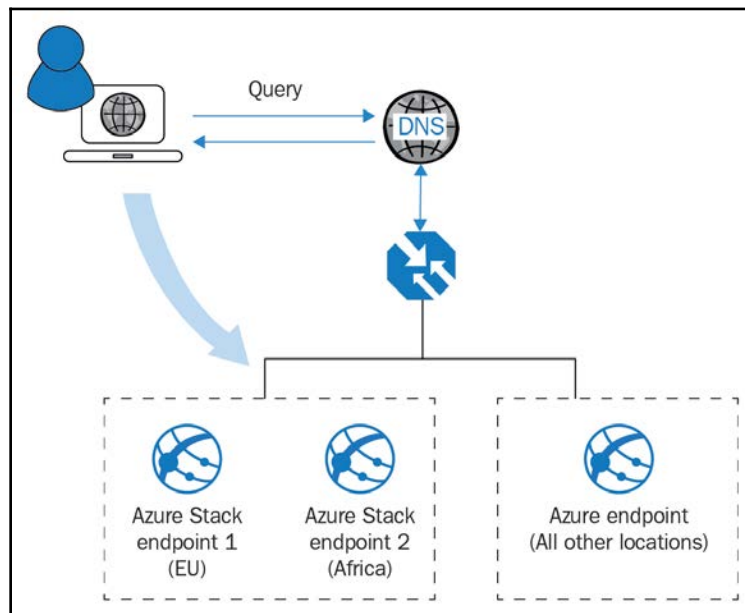
For more information, please refer to: <https://docs.microsoft.com/en-us/azure/azure-stack/user/azure-stack-solution-cloud-burst>.

Azure Stack geo-distributed Application

As you will see in this scenario, an application is being deployed on public Azure and on one or more Azure Stack environments. In combination with using DNS services and Azure Traffic manager, your environment automatically decides the proper cloud to answer an application request. If one is unavailable, it even automatically fails over to another one.



For more information, please refer to: <https://docs.microsoft.com/en-us/azure/azure-stack/user/azure-stack-solution-geo-distributed>.

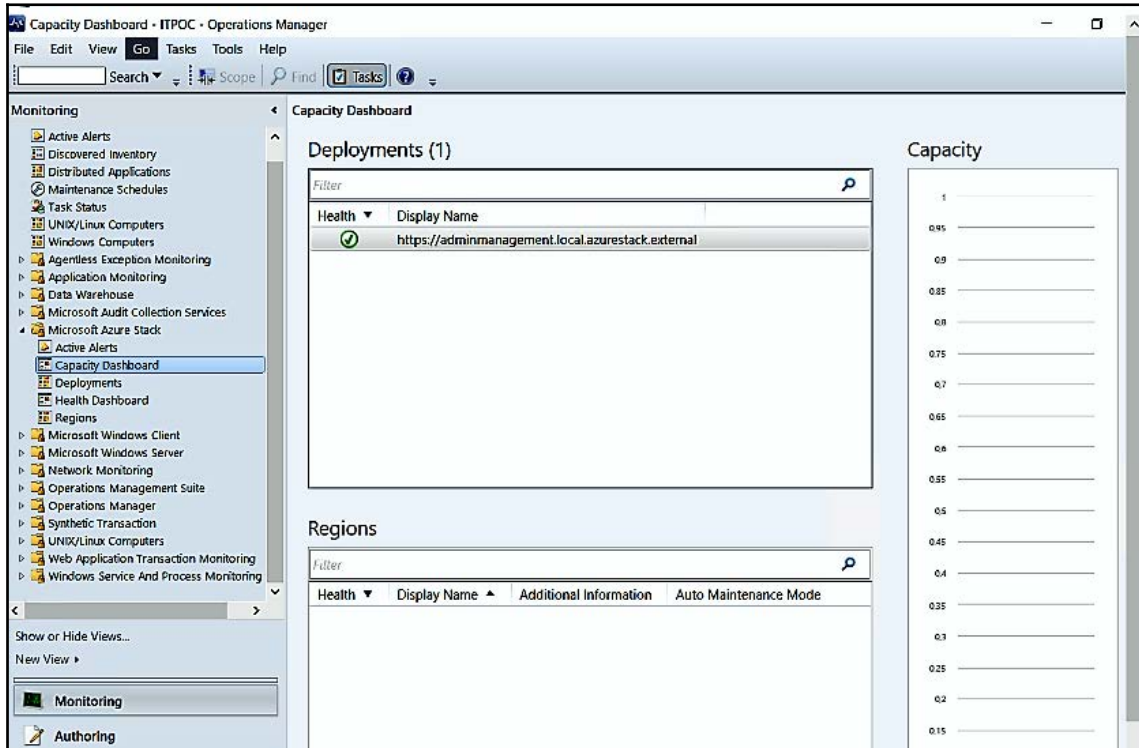


Monitoring Azure Stack

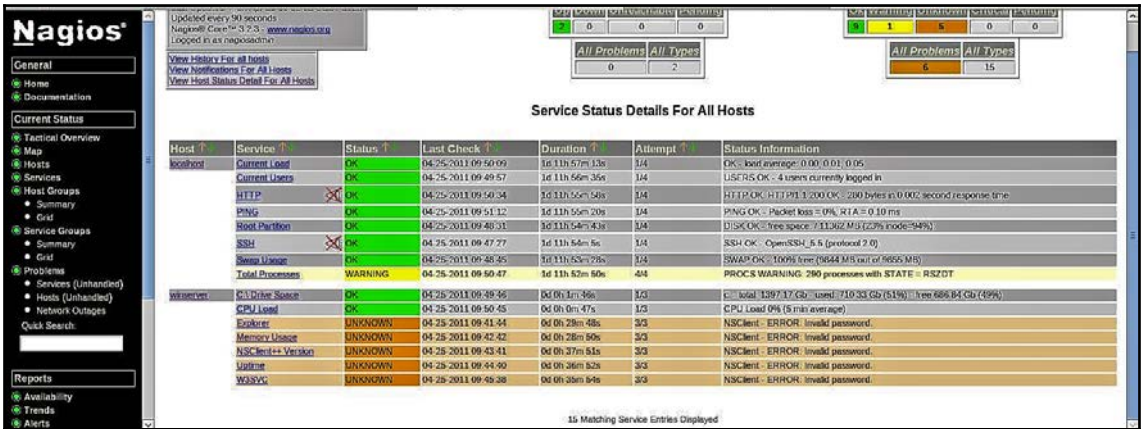
As Azure Stack is an integrated system, the monitoring solution for Azure Stack itself (and not the tenant workloads) has two pillars:

- Hardware monitoring
- Azure Stack monitoring

Hardware monitoring: As the Azure Stack hardware is vendor based, the monitoring of the components works with the hardware vendor-based monitoring solutions that already exist and have been proofed in the based without Azure Stack itself:



Azure Stack monitoring: The Azure Stack software monitoring is based on the corresponding health APIs. If you are running **System Center Operations Manager (SCOM)** on-premise, a SCOM solution is available to monitor multiple Azure Stacks with one SCOM:



If your primary monitoring solution is Nagios based, there is an integration available, too.

Summary

As you have seen in this chapter, a true hybrid cloud environment using Azure Stack and Azure as consistent solutions is possible and quite easy to implement. Azure Stack is a complex service and in general needs to be integrated in the data center if all required teams are part of the project.

Questions

Please answer the following questions:

1. Regarding which use cases should you think about Azure Stack?
2. What is the minimum and maximum size of Azure Stack hosts?
3. How do you buy Azure Stack?
4. What are the services that are available with Azure Stack so far?
5. Which connectivity scenarios are possible with Azure Stack today?
6. Is there a way to test Azure Stack without buying the appropriate hardware?
7. What is the servicing model of Azure Stack?

Further reading

Read the following articles for more information:

- **Building Hybrid Clouds with Azure Stack:** <https://www.packtpub.com/virtualization-and-cloud/building-hybrid-clouds-azure-stack>.
- **What is Azure Stack?:** <https://azure.microsoft.com/en-us/overview/azure-stack/>.

Assessments

Chapter 1: Getting Started with Azure Implementation

1. The three basic cloud service models are as follows:
 1. **Infrastructure as a Service (IaaS)**
 2. **Platform as a Service (PaaS)**
 3. **Software as a Service (SaaS)**
2. The four basic cloud deployment models are as follows:
 1. Public cloud
 2. Private cloud
 3. Hybrid cloud
 4. Community cloud
3. You use one or more cloud provider or solution. Either different public, private or hybrid clouds or you use different cloud service models from different cloud provider.
4. Datacenter of Azure Global Regions are connected to the Azure Global Backbone and Azure Global Active Directory. Sovereign Regions stand alone and built for a dedicated purpose, for example to enforce government restrictions, purposes or to fit customer requests.
5. The Edge Datacenter connects the Microsoft Global Backbone to internet exchanges, network providers and customers. Within a production datacenter the Microsoft Cloud workloads are calculated and operated, a production datacenter is connected to the Microsoft Backbone only.
6. The URL of the Azure portal is `portal.azure.com`.
7. The name of the Microsoft private cloud solution based on ARM is **Microsoft Azure Stack**.

Chapter 2: Azure Resource Manager and Tools

1. ARM templates are written in **JavaScript Object Notation (JSON)**.
2. ARM templates are declarative and define how a resource should look. ARM deploys these resources based on the definition.
3. Azure resource locks are a way to mark an Azure resource as read only or do not delete.
4. Everything that needs to be configured after a resource has been deployed (for example, setting user profile options) cannot be done with ARM templates. Therefore, PowerShell **desired state configuration (DSC)** is an option.
5. Custom ARM templates can be deployed using Azure CLI, PowerShell, or Azure Portal using the custom deployment feature.
6. The tools used to create ARM templates are:
 - Azure Portal
 - Visual Studio
 - Any other JSON editor
 - Simple Notepad
7. ARM is a basic service in Azure that is available globally. Setting the region for deploying a resource is an option for the deployment process itself.
8. As soon as a cloud service is based on Microsoft Azure, ARM templates can be deployed to each of them. The only pre-requisite that has to be met is that the service in general is available in that cloud.

Chapter 3: Deploying and Synchronizing Azure Active Directory

1. The four options for Azure AD are
 - Azure AD free
 - Azure AD basic
 - Azure AD premium P1
 - Azure AD premium P2

2. The three options can we choose to enable SSO for Azure AD are
 - Pass-through authentication
 - Password Synchronization
 - Federation with AD FS
3. User and groups management tool is used to synchronize users, groups, and devices with Azure AD
4. Yes, you can deploy Azure AD Connect as highly available
5. You have three days to recover the AD Connect Server
6. Yes, the `manage.windowsazure.com` is the only portal which is used to manage Azure AD
7. Azure AD attributes is the service we use when we need to limit Azure or Microsoft 365 service access through a network

Chapter 4: Azure Managed Applications

1. For publishing an Azure service catalog managed application the following tools can be used
 - Azure CLI
 - Azure PowerShell
 - Azure Portal
2. To create an Azure managed application definition the mandatory files are
 - `mainTemplate.json`
 - `createUiDefinition.json`

3. During the deployment of an Azure Managed Application two resource groups will be used. One resource group holds the application itself (Application Resource Group) and another resource group is used for the elements of the managed application (managed resource group)
4. The `mainTemplate.json` file is the ARM Template for the Managed Application and defines the resources and dependencies of the Managed Application.
5. The three elements automatically injected in the basic section of an UI Definition file are as follows:
 - The subscription
 - The resource group to deploy to
 - The location of the deployed resources
6. The method of testing your UI Definition in the Azure Portal is called **sideloading**.
7. The lock levels of an Azure managed application definition are **None** and **Read Only**. This setting defines the access level for deploying user to the managed resource group.

Chapter 5: Implementing Azure Networks

1. Out of the Azure Marketplace, Microsoft offers different third party so called Independent Solution Vendor Solutions. Examples for those solutions are Palo Alto and Barracuda Next Gen Firewalls.
2. No, a Virtual Network Gateway is a multi purpose solution and also supports ExpressRoute.
3. False, ExpressRoute is Private Interconnect Solution for MPLS, IPVPN or Private Layer 2/3 connections.
4. Currently there is only one Bandwidth option from 100 GBE available.
5. Yes, the Azure Application Gateway can operate in WAF Mode.
6. Microsoft SD-WAN Solution is called Azure virtual WAN.
7. Yes, with Azure Global VNet Peering, you can peer VNetworks across regions.

Chapter 6: Implementing Azure Storage

1. In version v2 general purpose storage tiering between hot, cold and archive storage is possible
2. The replication options available for Azure Storage are as follows:
 - Local Redundancy
 - Zone Redundancy
 - Geo Redundancy
 - Read-Geo Redundancy
3. No, it only replicates within one Azure Datacenter and Scale Unit
4. The storage services available are as follows:
 - File
 - Blob
 - Storage Que
 - Page
5. 5120 IOPS Baseline / 15260 IOPS with burst
6. The minimum Windows Server and PowerShell Version for Azure File Sync are as follows:
 - Windows Server 2012 R2
 - PowerShell 5.1
7. No, Azure Data Lake is built for Analytics and Big Data

Chapter 7: Virtual Machines in Azure

1. No, B-series VMs are burstable VMs which are running to a minimum of performance when they are not needed and will burst to 100% above CPU performance when needed
2. No, there are many third-party solutions also available
3. No, you don't need, that can also be done during the VM create but it is recommended to preconfigure the VNet and Network before deploying VMs
4. No, that's not possible.
5. Yes, the M-Series and SAP Large Instances
6. You can have a public IP for a VM but it is NATed on the private IP
7. Yes, you have the option to do so but it is more common and recommended to do it afterwards. That will also help to create backup groups of VMs with different or same backup policies

Chapter 8: Implementing Azure-Managed Kubernetes and Azure Container Service

1. A virtual machine is running its own operating system as a guest. A container is using its APIs to work with the underlying host OS.
2. The idea of microservices relies on a concept to split each application into different small services (called microservices) that are independent from each other and have predefined communication ports based on **Representational State Transfer (REST)**. Each microservice does not rely on a used technology and can scale with a load balancer in front.
3. A container registry is the location to store all used containers for a specified solution. In AKS the registry is the only point to load containers from.
4. Containers are supported with Windows Server 2016 and Linux based operating systems
5. A container orchestrator manages all containers in a solutions, scales and restarts them if needed.
6. Kubernetes, Docker Swarm and MESOS
7. UI based applications do not fit into the container based approach.

Chapter 9: Implementing Azure Cloud Services

1. There are currently two roles that can be defined in the Azure Cloud Service:
 - `WebRoles`
 - `WorkerRoles`
2. Currently, two roles can be defined in an Azure Cloud Service.
3. 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**.

4. The valid values for `osFamily` in a Cloud Service are as follows:

OS family	Server OS	Comments
1	Windows Server 2008 SP2	No longer available
2	Windows Server 2008 R2 SP1	
3	Windows Server 2012	
4	Windows Server 2012 R2	
5	Windows Server 2016	

5. The two possibilities to deploy my Cloud Service to the Azure platform are as follows:

- Direct **Publish...** to Azure through Visual Studio
- Building a **Package...** file to publish the service

Chapter 10: Implementing Azure Governance

1. Azure governance means a definition and tooling to make sure that Azure usage will be done based on an organizational definition. This includes Azure Naming Conventions also, as which regions should be used and who has which permissions in Azure itself.
2. RBAC is a role-based definition to match user accounts with the technical function in Azure. By default, more than 70 roles are already predefined, but if you are missing something, you could even set up your own custom role.
3. Azure Policies provide a way to technically define the structure of Azure usage based on naming conventions, regions to be used, needed tagging, and so on.
4. A management group is used to provide a scoping level under the subscription in Azure.
5. Azure Security Center and Azure Cost Management are tools to support Azure governance and have an appropriate reporting on it.
6. If you do not set up governance before starting to use Azure services, you might soon be lost in the middle of nowhere. Reapplying governance later is quite complex and may be not easy to apply. This might even lead to a redeployment of a complete service. Therefore, you should start with governance quite early in your Azure project.

7. The technical basic of Azure governance came from Cloudyn. They have been a start-up company who developed a solution to apply, report, and alert Azure costs and even Google Cloud Platform and Amazon Web Services. Cloudyn is depreciating and the new solution integrated into the Azure portal is Azure Cost Management.

Chapter 11: Azure Hybrid Data Center Services

1. Azure Stack always fits if one of the following scenarios exists:
 - You have a disconnected or partially disconnected scenario
 - Data privacy does not allow you to save data to public Azure
 - Migrating applications to cloud ready/stateless apps on premise
2. The minimum number of hosts is four, the maximum currently is 16 hosts.
3. Azure Stack can only be bought through the seven currently certified hardware vendors.
4. Azure Stack currently supports virtual machines, MySQL, and MSSQL as PaaS, App Services, and Azure Functions. For IoT Edge and Azure Container Services based on Kubernetes, a preview is available for template deployments (as virtual machines).
5. Azure Stack supports the connected scenario using Azure AD as identity provider or the disconnected scenario using ADFS.
6. For testing purposes, the ASDK is available. It is providing all services available on Azure Stack today, but does not provide high availability (as it is running on one host) and performance. In addition, there is no update servicing; you will need to reinstall it once an updated release is available.
7. Azure Stack servicing is available once per month, in general. It included one package to be installed through the Azure administrative portal. As of today, firmware, BIOS, or driver updates are being provided through the hardware vendor's deployment mechanism. This is planned to change in the future, resulting in one update package.

Other Books You May Enjoy

If you enjoyed this book, you may be interested in these other books by Packt:



Implementing Azure Cloud Design Patterns

Oliver Michalski, Stefano Demiliani

ISBN: 978-1-78839-336-2

- Learn to organize Azure access
- Design the core areas of the Azure Execution Model
- Work with storage and data management
- Create a health endpoint monitoring pattern
- Automate early detection of anomalies
- Identify and secure Azure features



Beginning Serverless Architectures with Microsoft Azure

Daniel Bass

ISBN: 978-1-78953-704-8

- Identify the key advantages and disadvantages of serverless development
- Build a fully-functioning serverless application and utilize a wide variety of Azure services
- Create, deploy, and manage your own Azure Functions in the cloud
- Implement core design principles for writing effective serverless code

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

▪

.NET 59

A

A-series VMs

about 294

basic 294

standard 294

accelerated networking

for VMs 305

access keys 274, 275

accounts

adding, to Azure AD 86, 87, 94, 97, 99, 104

Cloud accounts 86

Hybrid accounts 86

Active Directory (AD) 13, 79, 304

Active Directory Domain Services (AD DS) 79,
294

Active Directory Federation Services (AD FS) 105,
294

Active Directory Rights Management Services (AD
RMS) 100

AKS app

references 369

app package 137

app service 18

App Services 327

application resource group 136

ARM template

authoring 69

creating 70, 74, 76

deployment, exporting 54

modifying 62, 64, 67

resource (classic), exporting 60

resource group, exporting 54, 57, 59

working with 53

automation account 24

Automation script 59

availability sets

about 302

fault domain 302

update domain 302

Availability Zone (AZ) 126, 303

AzCopy

reference 274

Azure A-series basic, versus Azure A-series

standard

about 294

availability 294

CPU 294

disk IOPS 294

feature cut 294

price 294

usage 294

Azure A-Series basic

versus Azure A-Series standard 294

Azure A-Series standard

versus Azure A-Series basic 294

Azure Active Directory (AAD)

about 453

accounts, adding 87

Azure AD conditional access

about 128

device platform 128

device-enabled 128

group membership 128

sign-in and user risk 128

Azure AD Connect

about 99, 100

environment, installing 110, 117, 121, 124

infrastructure 126, 127

prerequisites, installing 106, 109, 110

reference 103

- synchronization rule editor 100
- synchronization service 101
- synchronization service key manager 102
- synchronization service web service configuration manager 102
- Azure AD DS 130
- Azure AD Sync 113
- Azure AD, electable options
 - about 82
 - Azure AD basic 82
 - Azure AD free 82
 - Azure AD premium P1 82
- Azure AD
 - about 80, 81, 504
 - accounts, adding 86, 87, 88, 94, 97, 99, 104
 - deploying 83, 86
 - groups, adding 86, 87, 88, 94, 97, 99, 104
- Azure App Service 371
- Azure application gateway 214
- Azure Building Blocks 27
- Azure CLI
 - about 59
 - reference 337
- Azure Cloud Service
 - about 372
 - Azure series, selecting 388
 - creating 393, 394, 396, 400, 401, 402, 404, 405, 407, 408, 410, 412, 413, 415, 419, 420, 421, 422, 424, 425, 426, 429, 433, 435, 436, 440
 - exploring 376
 - Guest OS, selecting 387
 - roles 373
 - Service Configuration File 373
 - service configuration file 384
 - service definition file 376
 - service endpoint 374
 - update level, selecting 387
 - versus Azure PaaS offerings 387
- Azure connections 206
- Azure Container Instance (ACI)
 - about 337, 340
 - container, creating in Azure 340, 343
- Azure Container Registry (ACR) 338
- Azure Container Service (ACS) 332, 338

- Azure Cost Management 99, 477
- Azure Costs 99
- Azure Data Services 27
- Azure execution model 27
- Azure ExpressRoute
 - about 198, 201
 - circuit, configuring 240, 242, 243, 245
 - Premium SLA 202
 - Standard SLA 202
- Azure File Sync/Storage services 265
- Azure Firewall 208
- Azure Free Trial
 - reference 16
- Azure governance 443
- Azure hierarchy
 - about 444
 - business unit approach 446
 - functional approach 445
 - geographic approach 446
- Azure Hybrid Use Benefit (HUB) 304
- Azure Kubernetes Service (AKS) 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369
- Azure Load Balancer (ALB) 27, 377, 521
- Azure load balancer
 - about 211
 - automatic reconfiguration 212
 - hash-based distribution 212
 - port forwarding 212
 - service monitoring 213
 - Source NAT 213
- Azure local gateway 196
- Azure managed applications
 - accessing 136
 - architecture 135
 - elements 135
 - history 133
 - overview 133
 - publishing, scenarios 133
 - purpose 133
 - resource template 135
 - security 136
 - user interface definition 135
- Azure Marketplace containers 344, 345, 346, 347, 348, 349, 350, 351

Azure networking components

- about 185
 - Azure application gateway 214
 - Azure connections 206
 - Azure DDoS 217
 - Azure DNS 216
 - Azure ExpressRoute 199, 201
 - Azure Firewall 208
 - Azure load balancer 211
 - Azure local gateway 196
 - Azure route 207
 - Azure third-party network devices 208, 210
 - Azure Traffic Manager 216
 - Azure virtual networks (VNet) 185, 187
 - Azure virtual WAN 197
 - Azure VPN gateways 189, 191, 193, 195
 - global VNet peering 188
 - VNet peering 188
 - Web Application Firewall (WAF) 214
- ## Azure networking
- architecture 257, 260
 - Azure virtual network gateway, setting up with
 - Azure ExpressRoute 238
 - Azure virtual network gateway, setting up with
 - MPLS 238
 - Azure virtual network site-to-site VPN, setting up 225
 - Azure VNet peering, setting up 245
 - Azure VNet, setting up 218, 221, 223, 225
 - custom routes, configuring 252, 254, 256
 - limits 185
 - setting up 217
- ## Azure PaaS offerings
- versus Azure Cloud Services 387
- ## Azure policies
- 457, 459, 462, 463, 465
- ## Azure PowerShell
- about 59
 - reference 24
- ## Azure pricing calculator
- reference 280
- ## Azure Reserved Instances (RIs) 305
- ## Azure Resource Explorer 30
- ## Azure resource group
- creating 36
 - resource, adding 38

Azure Resource Manager (ARM)

- about 17, 27, 28, 29, 30, 188, 307, 448
 - Azure resource group, creating 36
 - Azure resource group, resource adding 38
 - Azure resource group, storage account adding 39, 41, 43
 - functionalities 34
 - reference 197
 - tagging 45, 47
 - working with 35
- ## Azure Resource Manager Template Visualizer (ArmViz) 69
- ## Azure resource tags 45
- ## Azure resources
- ARM template, authoring 69
 - ARM template, creating 70, 74, 76
 - ARM template, deployment exporting 54
 - ARM template, modifying 62, 64, 67
 - ARM template, working with 53
 - locking 49
- ## Azure route 206
- ## Azure SDK
- reference 25
- ## Azure Security Center 473, 474, 476
- ## Azure series
- nutshell 393
 - selecting 388
 - series A 389
 - series D 390
 - series E 392
 - series G 392
 - series H 393
- ## Azure Service Definition Schema 376
- ## Azure Service Manager (ASM) 17, 188, 219
- ## Azure Stack cloud burst scenario
- about 511
 - reference 511
- ## Azure Stack configuration task 503
- ## Azure Stack Development Toolkit (ASDK)
- about 483, 484
 - host, preparing 485, 486, 488, 489, 491
 - identity management configuration 492, 495, 496
 - networking configuration 497, 499, 501
 - reference 485

- VM design 502
- Azure Stack geo-distributed application
 - about 512
 - reference 512
- Azure Stack PowerShell
 - reference 507
- Azure stack staged data analysis
 - about 511
 - reference 511
- Azure Stack
 - Command Line Interface (CLI), working with 508, 509
 - hardware monitoring 513
 - monitoring 512, 514
 - operating 504
 - portals, working with 504, 506
 - PowerShell, working with 506, 507
 - scenarios 510
- Azure Storage Account 327
- Azure storage account
 - about 263
 - Azure Data Lake storage 266
 - Azure File Sync/Storage services 265
 - Blob storage 263
 - BLOB storage account 263
- Azure Storage account
 - deploying 281, 286, 288, 290
- Azure storage account
 - File storage 263
 - general-purpose storage v1 account 265
 - general-purpose storage v2 account 265
 - Queue storage 263
 - Table storage 263
- Azure Storage Explorer
 - Azure Storage services, exploring 276
 - reference 276
- Azure Storage services
 - access keys 274, 275
 - account type 281
 - Blob storage services 270
 - data egress 281
 - exploring, with Azure Storage Explorer 276
 - File storage services 272
 - location 280
 - pricing 280
- Queue storage services 272
 - replication scheme 281
 - storage capacity 280
 - storage transactions 281
- Table storage services 271
- Azure third-party network devices 208, 210
- Azure tooling
 - about 473
 - Azure Cost Management 477
 - Azure Security Center 473, 474, 476
- Azure Traffic Manager 27, 216
- Azure virtual network gateway
 - configuring 239
 - setting up, with Azure ExpressRoute 238
 - setting up, with MPLS 238
- Azure virtual network site-to-site VPN
 - Azure virtual network gateway, configuring 228, 231
 - connection, configuring between local and virtual network gateways 234, 237
 - local network gateway, configuring 226
 - setting up 225
- Azure virtual networks (VNet)
 - about 185, 187
 - setting up 221, 223, 225
- Azure virtual WAN 197
- Azure VM types
 - A-series VMs 294
 - about 292
 - B-series 293
 - B-series VMs 298
 - Basic A-Series 293
 - Compute intense A-Series 293
 - D and DS-series 293
 - D-series VMs 295
 - DS-series VMs 295
 - E-series 293
 - E-series VMs 298
 - F and FS-Series 293
 - F-series VMs 296
 - FS-series VMs 295
 - G and GS-Series 293
 - G-series VMs 296
 - GS-series VMs 296
 - H-Series 293

- H-series VMs 296
- LS-Series 293
- Ls-series VMs 297
- M-series 293
- M-series VMs 298
- N-Series 293
- NC-series VMs 297
- NV-series VMs 297
- SAP HANA Large Instances 298
- SAP large instances 293
- Standard A-Series 293

- Azure VNet peering
 - configuring 248, 250, 251
 - deployment, preparing 246, 247
 - setting up 246

- Azure VNet
 - setting up 218

- Azure VPN gateways
 - about 189, 191, 194, 195
 - URL, for guide 189

B

- B-series VMs 298
- Backend network 186
- best of breed 14
- BLOB 16

- Blob storage account
 - about 263
 - archiving tier 264
 - cool access tier 264
 - hot access tier 264
 - premium storage performance tier 263
 - standard storage performance tier 263

- Blob storage services
 - about 270
 - append blobs 270
 - block blobs 270
 - page blobs (disks) 270

- Border Gateway Protocol (BGP) 199

C

- Chef server 299
- Citrix Terminal Server 120
- cloud 7
- cloud characteristics

- about 11
- broad network access 11
- measured service 11
- on-demand self-service 11
- rapid elasticity 11
- resource pooling 11

- cloud computing, deployment models
 - about 9
 - community cloud 10
 - hybrid cloud 10
 - private cloud 10
 - public cloud 9

- cloud computing, service models
 - about 8
 - Infrastructure as a Service (IaaS) 8
 - Platform as a Service (PaaS) 8
 - Software as a Service (SaaS) 9

- Cloud Cruiser 99

- Cloud Service 375

- Cloud Service architecture 372

- Cloud Service Package 373

- Cloud Services
 - service configuration file 384

- Cloudadmin 504

- Code Assets 373

- COM1 serial port 306

- Command Line Interface (CLI) 24, 29, 508

- Common Internet File System (CIFS) 272

- Component Object Model (COM) 381

- compute resource provider (CRP) 30

- compute-intensive instances 295

- connected mode 492

- container host 332

- container orchestration, features
 - automated container placement 352
 - configuration management 353
 - horizontal scaling 352
 - load balancing 353
 - rollout and rollback 353
 - self-healing 352
 - service discovery 353

- container orchestration

- about 352

- concept 352

- containers

- about 331
- basics 332, 334
- concept 332, 334
- creating 351
- hosts, deploying in Azure 336
- microservices, concept 334
- workloads, executing 335, 336

- createUiDefinition.json file
 - testing 158
 - testing, against Azure 158
 - testing, with local scripts 160
 - validating 158
 - validating, against Azure 158
 - validating, with local scripts 160

- custom routes
 - configuring 252, 254, 256

D

- D-series VMs 295
- DDoS service tiers
 - basic 217
 - standard 217
- Department of Defense (DoD) 18
- DirSync 113
- disconnected mode 492
- distance-vector routing protocol 199
- Distributed Denial of Service (DDoS) 217
- DNS settings
 - changing 322
- Do Not Delete 452
- Docker
 - on Linux 336
- Domain Controller (DC) 107
- domain controller (DC) 324
- DS-series VMs 295

E

- E-series VMs 298
- Enterprise Agreement (EA) 8, 86, 305, 443
- ExpressRoute 210
- ExpressRoute circuit, deploying
 - ExpressRoute Direct 204
 - ExpressRoute Global Reach 204
 - route filter 204
- ExpressRoute Global Reach 204

F

- F-series VMs 295
- federation 106
- Field Programmable Gateway Arrays (FPGAs)
 - reference 23
- field-programmable gate array (FPGA) 305
- file createUiDefinition.json
 - about 145
 - additional steps 147, 150, 153
 - outputs section 154
 - step 147
 - UI definition outline 146
- file mainTemplate.json 138, 143
- file storage 16
- File storage services
 - about 272
 - location and Azure Storage, latency between 273
 - storage limit 273
- first in first out (FIFO) 272
- Frontend network 186
- FS-series VMs 296
- fully qualified domain name (FQDN) 144

G

- G-series VMs 296
- gateway subnet 186
- general-purpose storage v1 account 265
- general-purpose storage v2 account 265
- geo-redundant storage (GRS) 267
- global VNet peering 188
- groups
 - adding, to Azure Active Directory (AAD) 90
 - adding, to Azure AD 86, 88, 94, 97, 99, 104
- GS-series VMs 296
- Guest OS
 - selecting 387

H

- H-series VMs 296
- high performance computing (HPC) 393
- hybrid cloud connectivity
 - configuring 510
 - reference 510

Hybrid cloud patterns

- about 509, 510
- Azure Stack cloud burst scenario 511
- Azure Stack geo-distributed application 512
- Azure stack staged data analysis 511
- Azure Stack, monitoring 512
- hybrid cloud connectivity, configuring 510
- machine learning solution, with Azure Stack 511

I

- Identity and Access Management (IAM) 13
- image 139
- inbound ports 312
- Independent Software Vendors (ISVs) 133
- information technology (IT) 7
- Infrastructure as a Service (IaaS) 8, 270
- infrastructure services
 - networking 16
 - operating system and server compute 16
 - storage 16
 - versus platform services 15
- input/output operations per second (IOPS) 280
- Internet of Things (IoT) 16
- Internet Service Provider (ISP) 198
- IP settings
 - changing 322

J

- JavaScript Object Notation (JSON) 17

K

- Key Management Service (KMS) 304
- Kubernetes 353
- Kubernetes service 355

L

- lift-and-shift migrations 273
- Lightweight Directory Access Protocol (LDAP) 130
- Lightweight Directory Services (LDS) 130
- load balancer probe 377, 521
- LoadBalancerProbe element 377, 521
- locally redundant storage (LRS) 266
- Ls-series VMs 297

M

- M-series VMs
 - about 298
 - reference 298
- machine learning solution, with Azure Stack
 - reference 511
- machine learning solution
 - with Azure Stack 511
- mainTemplate.json file
 - testing 155
 - testing, against Azure 155
 - testing, with local scripts 156
 - validating 155
 - validating, against Azure 155
 - validating, with local scripts 156
- managed application definition
 - creating 137
 - file createUiDefinition.json 145
 - file mainTemplate.json 138, 143
- managed application, Azure marketplace
 - business requisites 179, 181
 - technical requisites 179, 181
- managed application, from internal service catalog
 - deployment process 169, 175
 - managed resource group 176
- managed application
 - deploying, from internal service catalog 168
 - publishing, to Azure marketplace 178
 - publishing, to internal service catalog 161
- managed disks 300
- managed resource group 136
- Managed Services Providers (MSPs) 133
- management groups 453, 455, 457
- Mesos 353
- Message Passing Interface (MPI) 389
- Microsoft Azure, portal
 - reference 23
- Microsoft Azure, regions
 - references 19
- Microsoft Azure
 - about 14
 - automation 24
 - automation tools 24
 - Azure Resource Manager (ARM) 17

- basics 17
- Microsoft backbone 20
- Microsoft data center 19
- portal 23
- regions 18
- resources 17
- REST APIs 24
- services, overview 15
- Microsoft cloud servers
 - reference 23
- Microsoft Cloudyn 99
- Microsoft Identity Manager (MIM) 83
- Microsoft network cards
 - references 23
- Microsoft networking
 - references 23
- Microsoft Service Manager
 - reference 84
- multi-cloud characteristics 11
- multi-cloud model
 - best of breed 14
 - cloud brokering 14
- multi-cloud models 12
- multi-factor authentication (MFA) 80
- Multiprotocol Label Switching (MPLS) 10, 126, 199

N

- National Institute of Standards and Technology (NIST) 11
- NC-series VMs 297
- Network as a Service (NaaS) 14
- network function virtualization 21
- network interface card (NIC) 141
- network resource provider (NRP) 30
- network security group (NSG) 141
- NetworkConfiguration element 385
- NetworkTrafficRules element 383
- nutshell 393
- NV-Series virtual machines 297
- NV-series VMs 297

O

- organizational governance
 - about 443, 444

- Azure hierarchy 444
- naming standards 447

P

- path vector protocol 199
- peering types
 - Microsoft peering 200
 - private peering 200
- Platform as a Service (PaaS) 8, 371
- platform services
 - analytics and IoT 16
 - compute services 15
 - data 16
 - developer services 16
 - hybrid operations 16
 - integration 15
 - management 15
 - media and CDN 16
 - security 15
 - versus infrastructure services 15
 - web and mobile 16
- Point of Presence (PoP) 201
- Power BI Embedded 327
- premium storage account
 - about 280
 - requisites 280
- private key infrastructure (PKI) 105
- Private Network Interconnect 201
- probes, service monitoring
 - guest agent probe 213
 - HTTP custom probe 213
 - TCP custom probe 213
- Professional Developers Conference (PDC) 371
- proof of concept (PoC) 484
- Puppet 299

Q

- Quality of Service (QoS) 199
- Queue storage services
 - about 272
- Azure Queues 272
- Service Bus queues 272

R

- read-access geo-redundant storage (RA-GRS) 267
- redundancy 266
- Remote Desktop Protocol (RDP) 144
- remote direct memory access (RDMA) 295, 389
- replication
 - about 266
 - geo-redundant storage (GRS) 267
 - locally redundant storage (LRS) 266
 - read-access geo-redundant storage (RA-GRS) 267
 - zone-redundant storage (ZRS) 266
- resource auditing 453
- resource group 18, 448
- resource locks 452
- Resource Provider (RP) 30
- resource tags 451
- role-based access control (RBAC) 34, 447, 467, 470, 473
- roles
 - about 373
 - WebRoles 373
 - WorkerRoles 373
- Ruby 59

S

- S-series 293
- SAP HANA Large Instances 298
- Secure Sockets Layer (SSL) 214
- series A
 - about 389
 - Version 1 389
 - Version 2 389
- series D 390
- series E 392
- series G 392
- series H 393
- Server Message Block (SMB) 263, 272
- service catalog managed application definition
 - authentication 164, 167
 - basics section 163
 - creating 161
 - lock level section 164, 167

- package section 164
- service configuration file
 - about 384
 - NetworkConfiguration element 385
 - role element 384
- Service Definition File 373
- service definition file
 - about 373, 376
 - NetworkTrafficRules element 383
 - WebRole element 377, 379
 - WorkerRole element 381
- service endpoint 374
- service level agreement (SLA) 264
- Service Level Agreement (SLA) 266
- sets 126
- shared access signatures (SAS) 274
- single root I/O virtualization (SR-IOV) 305
- single sign-on (SSO) 80
- slots 403
- Software as a Service (SaaS) 9
- solid-state disk (SSD) 293
- source NAT (SNAT) 213
- Standard_L32s 297
- storage resource provider (SRP) 30
- Storage Spaces Direct (S2D) 297
- subnet reference 139
- Swarm 353
- System Center Operations Manager (SCOM) 514
- System Center Virtual Machine Manager 318
- System Integrators (SIs) 133

T

- Table storage services 271
- technical governance
 - about 443, 447
 - Azure policies 457, 459, 462, 463, 465
 - management groups 453, 455, 457
 - resource auditing 453
 - resource group 448
 - resource locks 452
 - resource tags 451
 - role-based access control (RBAC) 467, 470, 473
- template files
 - testing 154

- validating 154
- Traffic Manager 216
- Turbo Boost Technology (TBT) 295

U

- update level
 - selecting 387
- User Principal Name (UPN) 114

V

- virtual hard disk (VHD) 270
- virtual IP (VIP) 213
- Virtual Machine (VM)
 - about 292
 - accessing, in Azure 318, 320, 322
 - Azure-related communication traffic, optimization 324
 - common scenarios 324
 - deploying, in Azure 307, 309, 311, 315, 316
 - on-demand usage, for calculations 326, 327
 - on-premises servers, disaster recovery 328, 329
- virtual machine extensions 299
- Virtual Network (VNet) 221

- virtual Network ID 139
- virtual private network (VPN) 126
- Visual Studio Code 69
- Visual Studio Community Edition 69
- Visual Studio Dev Essentials
 - reference 17
- VM serial console 306
- VNet peering 188
- VPN device
 - reference 192
- VPN gateway 191

W

- Web Application Firewall (WAF) 214, 215
- WebRole element 374, 377, 379
- wide area networks (WAN) 197
- Windows Azure Pack (WAP) 10
- Windows Remote Management (WinRM) 110
- Windows Server Container VM 337
- WorkerRole element 374, 381

Z

- zone-redundant storage (ZRS) 266