Multi-Cloud for Architects

Grow your IT business by means of a multi-cloud strategy



Florian Klaffenbach, Markus Klein
and Suresh Sundaresan

Packt
www.packt.com

Multi-Cloud for Architects

Grow your IT business by means of a multi-cloud strategy

Florian Klaffenbach
Markus Klein
Suresh Sundaresan

Packt>

BIRMINGHAM - MUMBAI

Multi-Cloud for Architects

Copyright © 2019 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: Vijin Boricha
Acquisition Editor: Heramb Bhavsar
Content Development Editor: Sharon Raj
Technical Editor: Prashant Chaudhari
Copy Editor: Safis Editing
Project Coordinator: Drashti Panchal
Proofreader: Safis Editing
Indexer: Mariammal Chettiyar
Graphics: Tom Scaria
Production Coordinator: Nilesh Mohite

First published: January 2019

Production reference: 2260219

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

ISBN 978-1-78862-376-6

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 a technology solutions professional at Microsoft. He is a well-known expert 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, such as technical community manager and solution expert at Dell, and solutions architect at CGI Germany. He is also one of Packt's authors and has worked on books such as *Implementing Azure Solutions*, first and second edition. He spends his free time with his wife and little son, and is currently waiting for his second son.

I want to thank my fiancée, Jennifer, and my children, Alexander and Maximilian who were born while this book was being written. Although not an easy task, I found great support in my whole family who supported me by investing their valuable spare time and countless sleepless nights.

I would also like to extend my gratitude to everyone at Packt Publishing for their continuous support.

Markus Klein works as a technology solution specialist at Microsoft Germany, specializing in Azure and Hybrid Azure scenarios. He has been passionate about Microsoft technology for more than 20 years, starting with System Center, Service Provider Foundation, KATAL, Azure Pack and Azure, and later the Azure Stack. Before joining Microsoft, he worked as an architect at Microsoft Cloud Partners. In 2007, he founded a cloud community and was the co-founder of some Azure meetups, too. He was 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.

I want to thank my wife Melanie and may daughter Indra; they invested their valid spare time and lent me valuable support. I would also like to thank everyone at Packt Publishing for their help.

Suresh Sundaresan, a postgraduate from the prestigious Indian Institute of Science, is passionate about competency-building in the areas of cloud computing and storage technologies. After holding the position of Group Head Competency Development first at Wipro and then at IBM India, Suresh joined Oracle as a cloud architect for their Managed Cloud Services division. After a brief stint at Oracle, Suresh founded Cloudsiksha, a start-up dedicated to cloud education. He has now trained more than 2,000 engineers in cloud computing and DevOps. His clients include major multi-national corporations such as Philips, JPMC, Wells Fargo, Accenture, Wipro, TCS, Adobe, Epsilon, and many more. Suresh is AWS Solution Architect Professional certified.

About the reviewers

Toni Willberg is a cloud infrastructure subject matter expert with over 20 years of professional IT experience. He started his career in 1997 as a software developer and Linux system admin. Currently, Toni works with Microsoft as a cloud solution architect, helping Microsoft partners to build their cloud competences and practices.

Gajanan Chandgadkar has more than 12 years of IT experience. He has spent more than six years in the US, helping large enterprises architect, migrate, and deploy applications in AWS and Azure. He's been running production workloads on AWS for over six years and on Azure for the past year. He is a certified solutions architect professional and a certified DevOps professional with more than seven certifications in trending technologies. Gajanan is also a technology enthusiast who has extensive interest and experience in different topics, such as application development, container technology, and Continuous Delivery. Currently, he is working with Happiest Minds Technologies as DevOps architect and has worked with Wipro Technologies Corporation in the past.

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: Introduction to Cloud Computing 5
- **The history of cloud computing** 6
- **Characteristics of cloud computing** 8
- **Cloud models** 10
  - Cloud service models 10
    - Classic data centers 11
    - Infrastructure as a Service 11
    - Platform as a Service 12
    - Software as a Service 12
  - Cloud deployment models 13
    - Private clouds 13
    - Hybrid clouds 14
    - Public clouds 14
    - Community clouds 14
- **Definition of a multi-cloud** 15
- **Summary** 17

Chapter 2: Microsoft Public Cloud Offerings 18
- **Microsoft Azure** 18
- **Microsoft 365** 19
- **The strategy behind Microsoft's cloud offerings** 19
- **Microsoft data centers and global network backbone** 20
- **The Microsoft Azure service portfolio** 24
  - Computing 24
  - Network 24
  - Storage 25
  - Mobile 26
  - Databases 26
  - Data and analytics 27
  - AI and cognitive services 28
  - Internet of things 28
  - Enterprise integration 28
  - Security and identity 29
  - Developer tools 29
  - Monitoring and management 29
  - The Azure Marketplace 30
- **The Microsoft 365 service portfolio** 30
  - Microsoft Office 365 31
  - Windows 10 as a service 33

Enterprise mobility and security 33
Microsoft Trust Center 33
Summary 34

Chapter 3: Amazon Web Services and Its Public Cloud Service 35
Regions and availability zones 36
AWS Edge Locations 37
AWS services 37
AWS compute services 37
AWS storage services 40
AWS database services 42
Networking 43
DevOps 45
Security, identity, and compliance 48
Monitoring and management tools 50
Integrating an application 51
Analytics 52
Container service 53
Migration services 54
Machine learning 55
Augmented reality and virtual reality 57
Mobile 57
Internet of Things 59
Game development 60
Customer engagement 61
Business productivity 61
Desktop and app streaming 62
Media services 62
Summary 64

Chapter 4: Microsoft Azure Stack and Private Cloud Services 65
Microsoft Cloud and Azure Stack 66
Resource providers 67
Azure Stack's core management services 68
Authorization management 68
Subscriptions, plans, and offers 69
Plan 69
Offer 70
Subscription 70
Events 70
Working with Azure Stack 70
Azure Stack portal 70
PowerShell commandlets 71
Azure Stack CLI 72
Azure Stack API 72
Sizing Azure Stack 73

Azure Stack services 75
Infrastructure as a Service 75
Platform as a Service 76
Serverless computing on Azure Stack 77
Azure design scenarios 78
Sensitive data 78
Regulations 78
ROI and TCO 79
Starting an Azure Stack experience 79
Testing Azure Stack in public Azure 82
Summary 82

Chapter 5: OpenStack and Its Private Cloud Service 83
The history of OpenStack 84
OpenStack architecture 84
Control plane 86
OpenStack services 86
OpenStack compute service – Nova 87
OpenStack bare-metal service – Ironic 89
OpenStack identity service – Keystone 89
OpenStack block storage service – Cinder 90
OpenStack object storage service – Swift 91
OpenStack networking – Neutron 92
OpenStack image service – Glance 93
OpenStack dashboard – Horizon 94
OpenStack telemetry service – Ceilometer 94
OpenStack orchestration service – Heat 95
OpenStack backup/restore and disaster recovery service – Freezer 96
OpenStack container service – Zun 97
OpenStack load-balancing solution – Octavia 97
OpenStack database service – Trove 98
OpenStack DNS service – Designate 99
OpenStack container orchestration – Magnum 99
OpenStack messaging system – Zaqar 100
OpenStack data-processing service – Sahara 101
OpenStack resource-reservation service – Blazar 102
OpenStack cluster service – Senlin 102
Computing inside storage objects – Storlets 103
Application data protection – Karbor 103
Resource optimization service – Watcher 103
Key manager service – Barbican 104
Root-cause analysis service – Vitrage 104
OpenStack search service – Searchlight 104
OpenStack file-sharing service – Manila 105
Summary 106

Chapter 6: Network Interconnects to the Cloud 107
Connectivity providers 107
Network service providers 107
Internet service providers 108
Internet exchange points 108
Exchange providers 108
Cloud and content providers 109
Technology and terms 110
Autonomous Systems 110
Border Gateway Protocol routing 111
Open Shortest Path First 111
Private peering or private network interconnect 111
Internet direct peering or public peering and remote peering 111
Decentralized and centralized routing 112
Common connectivity options for cloud providers 114
Encryption via the internet 114
Virtual private network tunnels via the internet 114
Private network interconnect via Ethernet, MPLS, or an IPVPN provider 114
What matters the most 115
Latency and bandwidth 115
Data in transit and data that's stored 116
Summary 117

Chapter 7: An Overview of Network-Based Interconnect 118
Public interconnect and points of presence 118
Interconnect via internet (HTTPs) 121
Azure VPN 123
VPN with network virtual appliances in Azure 124
Private network interconnects 127
Azure ExpressRoute 127
Azure ExpressRoute Global Reach 129
Azure ExpressRoute Direct 131
Mixed interconnect with software-defined WANs 131
Azure Virtual WAN 133
Summary 134

Chapter 8: An Overview of Network-Based Interconnecting AWS 135
A VPC 136
Subnets 137
Internet gateways 139
Routing tables 139
NAT gateways 141
The Network Access Control List (NACL) 141
Building a VPC 142
Step 1 – Creating a VPC 143

Step 2 – Creating the subnets 143
Step 3 – Creating an internet gateway 143
Step 4 – Creating a routing table 143
Step 5 – Launching instances and associating Elastic IP 146
Step 6 – Creating a NAT gateway 146
Interconnecting to an Amazon VPC 147
Connecting to an Amazon VPC using a hardware VPN 148
Interconnecting using Direct Connect 149
Interconnecting using Direct Connect and a VPN 150
AWS VPN CloudHub 152
Software VPN 153
Connecting without a VPN 154
Interconnecting between Amazon VPCs 155
VPC peering 155
Software VPN 156
Software-to-hardware VPN 158
Hardware VPN 158
Direct Connect 159
Interconnecting to AWS services from inside a VPC 159
VPC endpoints – the gateway endpoint 160
VPC endpoints – the interface endpoint 161
Endpoint services 162
Transit Gateway 163
Summary 165

Chapter 9: Interconnecting Different Cloud Solutions 166
Interconnecting cloud solutions in the network layer 166
Interconnecting via the internet 167
Virtual Private Networks 169
Interconnecting via private network interconnects 169
Interconnecting cloud solutions in the identity layer 171
Multi-cloud templates and management 174
Summary 176

Chapter 10: Identity and Access Management in Microsoft Azure 177
Directory services in general 177
Active Directory basics 178
Azure Active Directory basics 179
Azure Active Directory 183
Azure AD users 188
Azure AD groups 189
Azure AD roles 190
Azure AD devices 191
Azure AD design guidelines 191
Designing a hybrid cloud identity management solution 193
Active Directory Federation Services 195

Seamless single sign-on 196
Pass-through authentication 196
Custom domain names 198
Designing a hybrid cloud access management solution 200
Self-service password reset 200
Multi-factor authentication 202
Summary 206

Chapter 11: An Overview of Identity and Access Management in AWS 207
How IAM works 208
The principal 208
Resources 208
Actions or operations 209
Authentication 209
Authorization 209
Requests 210
Users 210
Root users 210
IAM users 211
Federated users 212
SAML 2.0-based federation 212
Web identity federation 215
Groups 216
Roles 218
AWS service roles 219
AWS service roles for EC2 instances 219
AWS service-linked roles 219
Delegation 220
Cross-account access roles 221
Access management 221
Evaluating a policy 223
Policy types 223
Identity-based policies 223
Resource-based policies 224
Writing a JSON policy 224
Managing policies 227
AWS managed policies 227
Customer-managed policies 228
Inline policies 230
Summary tables 231
Summary 231

Chapter 12: Clustered Application Servers 232
Technical requirements for cross-cloud application servers 232
Networking connectivity between different clouds 233
Single identity management solutions for all servers 233

Supported applications for georedundancy 233
Examples of clustered application servers 235
Microsoft SQL Server 235
Microsoft Exchange Server 237
Supporting cross-cloud implementations using geo load balancers 240
Azure Traffic Manager 240
AWS Route 53 241
Managing multi-cloud virtual machines for clustered application servers 243
Troubleshooting cross-cloud application servers 245
Summary 245

Chapter 13: Mixed IaaS, PaaS, and SaaS in Multi-Cloud Environment 246
Design guidelines for multi-cloud solutions 246
Definition of cloud services 247
Infrastructure as a Service 247
Platform as a Service 248
Software as a Service 248
Containerization 248
Cloud Foundry 249
Docker 250
OpenShift 251
Microservices 252
Docker Services as a Service 253
Best practices 253
Best practices 253
A real-world design scenario 254
Solution design 255
Preparations 255
Networking 258
Identity management 259
Modern workplace services 263
Infrastructure services 264
Setting up new cloud services 267
Development environment 268
Cost management 269
Monitoring and security 270
Summary 271

Another Book You May Enjoy 272

Index 274

Preface

Public cloud offerings are becoming increasingly interesting for customers. But some of them are not willing to go with a single-vendor login, and are therefore looking at a two-vendor approach. This will most likely mean the use of **Amazon Web Services** (**AWS**) and Microsoft Azure. However, the major challenge facing architects and administrators is to implement these solutions appropriately.

The focus of this book is to get an idea as to how a multi-cloud design approach appears, and how it can be validated and then designed. This book offers a helping hand and describes a number of approaches in terms of architecting your solution, based on real-world design scenarios.

Who this book is for

If you are a cloud architect, solutions architect, system/network administrator, or a DevOps engineer aware of cloud solutions and keen to successfully architect them to your organization, then this book is for you.

What this book covers

`Chapter 1`, *Introduction to Cloud Computing*, covers the basics of cloud computing and provides explanations of potential cloud scenarios.

`Chapter 2`, *Microsoft Public Cloud Offerings*, describes the Microsoft Azure public cloud offerings and the basic ideas behind them.

`Chapter 3`, *Amazon Web Services and its Public Cloud Service*, talks about the AWS public cloud offerings and their basic designs.

`Chapter 4`, *Microsoft Azure Stack and Private Cloud Services*, covers the Azure stack, the hybrid cloud solution for Microsoft Azure in your data center.

`Chapter 5`, *OpenStack and its Private Cloud Service*, describes the OpenStack solution as a private cloud stack and their typical implementation scenarios.

`Chapter 6`, *Network Interconnects to the Cloud*, talks about the underlying network interconnectivity to make sure that resources distributed between Microsoft Azure and AWS can reach one another.

`Chapter 7`, *An Overview of Network-Based Interconnect Microsoft*, describes the Microsoft Azure inter-connectivity technologies.

`Chapter 8`, *An Overview of Network-Based Interconnecting AWS*, covers networking connectivity basics in AWS.

`Chapter 9`, *Interconnecting Different Cloud Solutions*, describes how to interconnect AWS and Azure networking technologies.

`Chapter 10`, *Identity and Access Management in Microsoft Azure*, shows the technical design of Azure Active Directory for the purpose of in-depth identity and access management.

`Chapter 11`, *An Overview of Identity and Access Management in AWS*, covers the networking topology and technology within AWS.

`Chapter 12`, *Clustered Application Servers*, shows how to set up application servers in clustered environments that are distributed between AWS and Azure.

`Chapter 13`, *Mixed IaaS, PaaS, and SaaS in Multi-Cloud Environments*, covers a mixed scenario of SaaS, IaaS, and PaaS in a multi-cloud environment based on Microsoft Azure and AWS.

To get the most out of this book

You should have a basic knowledge of Azure and AWS. Reading through this book will afford you a deep understanding of how things should be designed, based on a multi-cloud approach, and will guide you through each design step. The book also describes how some approaches really work in practice. In addition, the book should provide a solid background in terms of the consulting approaches undertaken by you or your customers.

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: "All traffic from the subnet whose destination is `10.0.0.0/16` must be routed within the VPC (local)."

A block of code is set as follows:

```
 {
"Version": "2012-10-17",
"Statement":
{
"Effect": "Allow",
"Action": "s3:ListBucket",
"Resource": "arn:aws:s3:::mybucket"
}
}
```

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

```
 {
"Version": "2012-10-17",
"Statement":
{
"Effect": "Allow",
"Action": "s3:ListBucket",
"Resource": "arn:aws:s3:::mybucket"
}
}
```

Bold: Indicates a new term, an important word, or words that you see on screen. Here is an example: "The security of the subnets is provided via **network access control lists** (**NACL**)."



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
Introduction to Cloud Computing

Cloud services have grown more and more popular over the last decade. They have become one of the biggest trends in business, and recently got an own drive in the economy.

In the early 2000s, cloud computing wasn't a widely used phrase, but the concept of shared resources (or GRID computing), as well as data centers with massive computing power, already existed. Later in that same decade, the word cloud became a buzzword, referring to nearly anything that was not tangible or online. However, the real rise in cloud computing, with all of its different service models, happened when the big IT companies started to offer cloud services. Companies such as Amazon, Microsoft, and Salesforce got into the cloud business.

With these companies came a massive increase in cloud offerings, services, and even private cloud solutions, and IT infrastructures and solutions became more and more complex. It is now almost impossible to find architects that are able to understand all of the solutions and their behaviors, and it is even harder to connect the solutions together.

But, first things first; when you are out in the field having discussions with people, you hear many thoughts about the cloud and what it means to different people. Depending on what person you ask, you will get answers that suggest the cloud started with mainframes, or that the cloud is only virtualization with a nice GUI.

Due to such complications and the development that cloud computing has undergone over the last decade, I don't want to start the chapter like every other book about cloud computing: with cloud models. Yes, we will discuss models, but it will be a bit later in the chapter. First, I want to delve in slightly more depth into the history of shared resources and cloud computing. I will try to prove that everyone is partially correct with his or her opinion, but only the current iteration of shared resource computing should be named the cloud.

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

- The history of cloud computing
- The difference between shared resources and the cloud
- Cloud service models
- Cloud deployment models
- Cloud characteristics

The history of cloud computing

Cloud computing is one of the oldest terms in information technology. As early as the beginning of the 1990s, some people in the IT industry prophesied that computers would be distributed over the network; they suggested that cloud computing would emerge as soon as the technology was ready. By the end of the 1990s, the time had come, with the development of the multitenant architecture.

In 1995, Gesellschaft für Mathematik und Datenverarbeitung mbH, from Germany, known today as Fraunhofer FIT, presented the **Basic Support for Cooperative Work** (**BSCW**), a system that would be called a cloud today. Users were able to upload web-based documents to folders and share them with others, like the functionally included in Facebook, Dropbox, or OneDrive today. Since 1996, BSCW has been offered commercially by the Fraunhofer spin-off OrbiTeam.

However, the term cloud computing was mainly defined by fast-growing internet companies, such as Amazon, Microsoft, Google, and Yahoo. Because of the rapid growth of their user bases, these companies faced the problem of a constantly growing number of servers that would need to provide sufficient performance, even at peak load times. As an example, during that time (and even today), Amazon had high peaks during Christmas, and over the rest of the year most resources weren't used efficiently enough. For Amazon, the peak load in 2006 was higher than the basic load in daily business by a factor of ten. This meant that Amazon's Christmas business was only using 10 percent of its data center capacity and was wasting a lot of money on power, cooling, and space for their servers.

To address this problem, Amazon decided to use their service-oriented architecture and the services that had been designed and established to cope with a very fluctuating, often very high number of users, and transform it into a product that could be offered to outside customers. For Amazon, that meant that peak loads would be distributed among all of the users of their infrastructure.



Statistically, it is easier to predict the decisions of a high number of people than to predict the decision of a single person. In cloud computing, it is the same. The more users you have in your cloud infrastructure, the more efficient the infrastructure becomes, and the better you can optimize and predict the workload.

At the beginning of the 2000s, this step was a logical consequence for Amazon, because the company had already switched to small, fast-moving, **two-pizza teams**. These teams were responsible for implementing new functionalities based on the existing cloud infrastructure's implementation.



Fun fact: The two-pizza teams were based on the **2 Pizza Rule** of Amazon's CEO Jeff Bezos. According to Bezos, if a team couldn't be fed with two pizzas, it was too big.

The scaling effects of the cloud services thus became the basis of cloud computing itself, which was offered publicly. Amazon, with its daughter company Amazon Web Services, is now the world's largest cloud provider.

A prerequisite for the use and dissemination of cloud computing services is a fast broadband connection, with high bandwidth and low latency. With such connectivity, it makes no difference whether data is stored locally on a PC, or in remote servers in a cloud. As a result, the increasing importance of cloud computing for private users to the market supply is coupled with equally fast, reliable, and cost-effective DSL and LTE connections.

Cloud computing still faces many issues, not from the technology side of things, but from a political and psychological perspective. Even in highly technological countries such as Germany, cost-efficient and reliable broadband internet connections aren't always available, or must be built with high costs and with a high political influence. Then, there are tax issues; for example, customers that move their servers to the cloud and transfer CAPEX to OPEX costs have less depreciation. These forces them to pay higher taxes on their earnings. The antique data security and privacy laws, from a time when letters were brought by men or women on horses, are also delightful. That are problems that must be solved by the governments in old Europe, but for most of them, this is unknown territory, and no one wants to set foot on that ground. These decisions (or non-decisions) can slow down the adoption of cloud computing in countries with old and traditional economies, but these decisions can give countries with new and fast moving economies, such as Iceland, Brazil, or Singapore, the opportunity to move up in the global market.

Looking at society, cloud computing is changing our conduct and how we work with information. We are used to accessing our data, infrastructure, and applications at any time, from anywhere. We are building new solutions much more quickly and we collaborate with each other to build new products or to finish projects. Cloud computing is becoming the engine of the future.

Characteristics of cloud computing

When we look at the current cloud offerings, we can see different definitions of cloud computing.

When people discuss VMware, for example, they also use the name cloud computing for their products, but a deeper look at their solutions shows the characteristics of grid computing, which is similar to cloud computing, but is basically distributed virtual workloads.

Grid computing can be described as follows. In grid computing, resources from multiple computers and multiple locations are collected, in order to reach a goal or fulfill a duty. When you think of a grid, you can consider it as a distributed system with static workloads that involve a high amount of files. Grid computing is different from conventional, high-performance computing systems or clusters. In those grid clusters, each node is set to perform a different task or to run a different application. In general, grid computers are more heterogeneous and geographically dispersed than clusters. A single grid can be dedicated to an application, but commonly, grids are used for multiple purposes. Grids are a form of distributed computing in which a super virtual computer is built from a couple of computers that act together to perform complex and long-term tasks. For certain applications, distributed (or grid) computing can be understood as a special kind of parallel computing that relies on whole compute units, consisting of CPUs, memory, storage, power supplies, network interfaces, and so on, connected to a public or private network by common network interconnection standards, such as Ethernet or InfiniBand.

In my personal opinion, the most appropriate definition of cloud computing comes from the **National Institute of Standards and Technology** (**NIST**). NIST describes cloud computing as follows:

- **On-demand, self-service**: An automated deployment of resources that the consumer orders through an interface, such as a consumer portal.
- **Broad network access**: Providing the 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:** This is also known as rapid scaling, and it describes the ability to scale resources in a massive way. The automatic and fast assignment and reassignment of resources, and the rapid up and down scaling of single instances, are key terms involved in rapid elasticity. The adjustment of web server resources, depending on the demand, is an example of rapid elasticity.
- **Measured service**: All usage data for consumer resources is monitored and reported, to make it available for consumers and the cloud provider. This is one of the requirements for minute-based billing.

From a cloud provider perspective, I would personally add some additional points to that definition, as follows:

- **Automated and self-maintaining data center**: The data center of a cloud provider must be fully automated, and the intervention of the provider must be at a minimum or zero touch level.
- **Software defined**: In my opinion, most of the infrastructure within a cloud provider data center must be software-defined. There is no longer a need for storage or network appliances. The only requirements are OSI Layer 2 switches, which can be managed via a command-line interface, and servers that support the **Intelligent Platform Management Interface** (**IPMI**). Everything that's on top, such as routers or the firewall, should run as **Network Virtual Appliances** (**NVAs**).
- **High density and efficiency**: Cloud data centers should work with a very high density in packaging computing and network resources, in order to increase efficiency in energy, cooling, and space.
- **Forward thinking**: A cloud provider should use technology that makes data centers more efficient and environmentally friendly. Cloud providers should also improve their service offerings in such a way, and be innovative when it comes to new services.



Thanks to the openness of some cloud providers, we get hints about what is coming in the future, such as the initiative from Microsoft and Facebook to build the longest and fastest transatlantic cable ever built. The MAREA cable has a capacity of 160 Tbits per second and a length of 6.600 kilometers; read more about it at `https://en.wikipedia.org/wiki/MAREA`.

Recently, we have seen a lot of technologies coming to the market that are inspired by cloud providers and their needs, such as **Remote Directory Memory Access** (**RDMA**), or new server designs based on server architectures, such as the Open Compute Project Olympus: `http://www.opencompute.org/wiki/Server/ProjectOlympus`

Cloud models

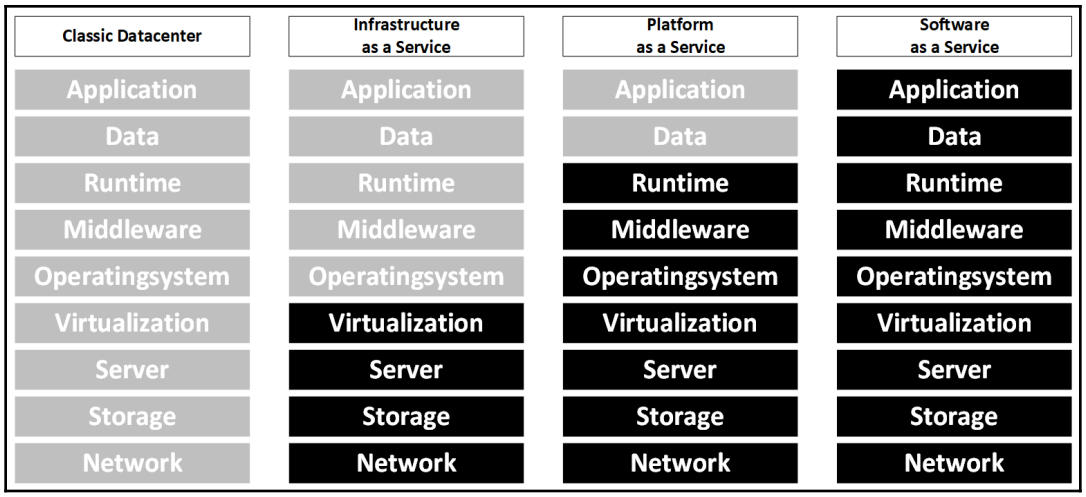
With our definition of cloud computing, two types of models were referred to: the cloud service model and the cloud deployment/availability model. They can be described as follows:

1. The cloud service model refers to the offerings of a cloud provider. Does it only offer the infrastructure, or does it offer fully managed software?
2. The cloud deployment model refers to who can access the cloud provider's offerings. Is it available to everyone, is it available to a special group of people, or is it completely private?

We will go over these models in the next sections.

Cloud service models

When considering cloud computing in terms of cloud service models, there are four main categories. The first is our classic data center, the second is Infrastructure as a Service, the third is Platform as a Service, and the last is **Software as a Service** (**SaaS**):



With regards to the IT industry, some IT vendors and companies have created new as-a-service definitions to fit their products, for example, **Desktop as a Service** (**DaaS**), which is mostly used by virtual desktop providers to push their clouds, or IT as a Service to find a cloud definition for IT outsourcing in the new age. There are many more versions out there, but when the terms are broken down, each of these buzzwords only delivers one of the four major cloud service models.

Classic data centers

Classic data centers are what we have known for decades. We have our servers, storage, and network, and we are in charge of the power supply, the air conditioning, the security, and the building itself. We have the highest amount of flexibility, because we are able to choose even the smallest screw that we want to use; but, we are also responsible for everything at the top. The current development in data center technology is going in the direction of software-defined data centers, in order to gain benefits from the cloud within a classic data center, and to reduce dependencies on hardware components. Most of the classic data centers do not offer a lot of self-service, and are only limited to automation and zero-touch administration.

Infrastructure as a Service

Infrastructure as a Service (**IaaS**) refers to a model in which the cloud provider gives the consumer the possibility to create and configure resources from the computing layer upwards. This includes virtual machines, networks, appliances, and many other infrastructure-related resources.

IaaS gives the customer the flexibility to create a solution, but the customer has many responsibilities. He is in charge of the backup and configuration of his virtual machine. He needs to ensure the availability of the application, which includes clustering or failover. Depending on the cloud provider, the customer may have to configure network parameters, like firewall rules, network and subnet masks, and so on. On the other hand, he can use any application or solution that supports the operating system and virtualization platform.

Typical examples of IaaS cloud providers are Microsoft Azure, Amazon Web Services, Google Cloud Platform, and Alibaba Cloud.

Platform as a Service

Platform as a Service (**PaaS**) gives the consumer an environment from the operating system upwards. So, the consumer is not responsible for the underlying IaaS infrastructure. Examples include operating systems, databases, or development frameworks. This gives the customer the opportunity to only focus on the development of his application solution and the data that he needs to work with. He is no longer responsible for backups, except for the data or redundancy of his solution. Backing up data is handled by the cloud provider. On the other hand, functionality and flexibility are often very limited, and not every dirty trick that you might try in an IaaS solution is available in PaaS solutions; for example, there could be a limitation in SQL. In Microsoft SQL Server, there are options to work with the database, that are not available in Microsoft Azure SQL.

Examples of PaaS cloud solutions and providers include Microsoft Azure, Amazon Web Services, and IBM Bluemix.

Software as a Service

Software as a Service (**SaaS**) is the model with the lowest levels of control and required management, but with the most limitations. SaaS solutions give the customer minimal opportunities to customize solutions. If you want to use an SaaS solution, there is no option to change the behavior or anything outside the given limitations. In simple terms, you take it or leave it. An SaaS application is accessible to multiple clients and customers, and the owning consumer doesn't have any control over the backend, except for some application-related management tasks. Therefore, the customer gets a high amount of scalability, and is not in charge of backups, updates, redundancy, and so on. All of that must be provided by the cloud provider along with SaaS solutions.

Examples of SaaS cloud solutions are Microsoft 365 Services (such as Office 365 or Dynamics), Salesforce, Dropbox, and Google Docs.

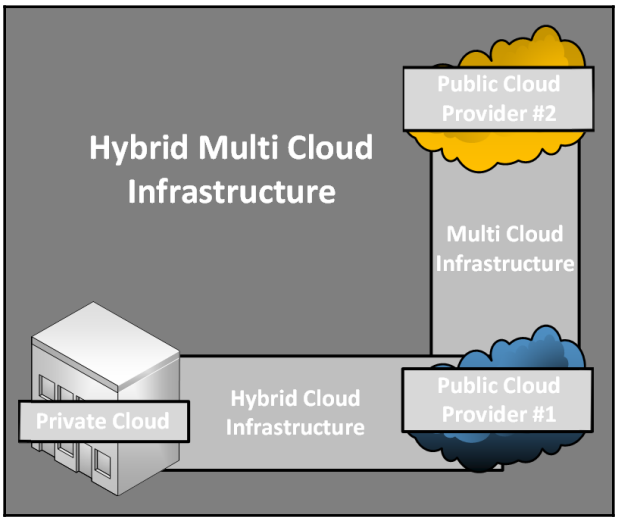


Sometimes, it is not really easy to decide what kind of cloud service you should use. It is mostly based on what perspective you look into the cloud environment from. Let me explain this with an example. Suppose that you are a hosting service provider that runs VMware and some automation to deploy hosts and tenant infrastructures. You use classic storage and network solutions, without any software-defined layer in between. For you, this is a classic data center with some automation on top. To your customer, who is clicking through a web portal to deploy a virtual machine, it might look like an IaaS offering.

Cloud deployment models

Furthermore, there are a number of deployment models for cloud computing that need to be discussed. These deployment models cover nearly all of the common cloud computing provider scenarios. They describe groups of consumers that are able to use the services of the cloud provider.

In the following diagram, you can see a schematic view of the dependencies between the cloud models:



Looking at the distribution model, there are also special forms of cloud distribution models. The most common one is the community cloud. A community cloud is only accessible by a specified group. The groups can be connected by location, membership of an organization, or for reasons of compliance.

Private clouds

A **private cloud** describes an environment/infrastructure built and operated by a single organization, for internal use. These offerings are specifically designed for the different units in the organization. Examples include Microsoft Azure Stack, or OpenStack, if they are used for internal deployments.

Hybrid clouds

A **hybrid cloud** combines private and public clouds. It is defined as a private cloud environment at the consumer's premises, as well as a public cloud infrastructure that the consumer uses. Therefore, there is no difference if you combine major resources together, such as web servers in the cloud, and keep your data on-premise. A hybrid cloud is already provided when you synchronize your identities from your Active Directory to Azure AD to use Office 365 or other SaaS cloud solutions. Every time you sync, bind, or merge local resources, with or to public cloud resources, you create a hybrid cloud solution.

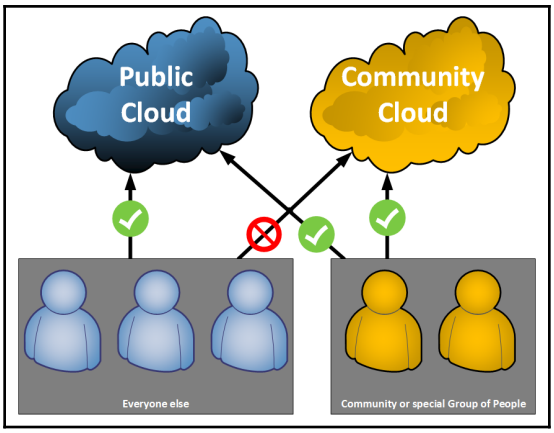
Public clouds

A **public cloud** describes a cloud computing offering that can be accessed by the public. This includes individuals, as well as companies. Examples of public clouds include Microsoft Cloud Services (such as Microsoft Azure or Office 365), Salesforce, Dropbox, and Amazon AWS.

Community clouds

A **community cloud** is a special form of public cloud that is only accessible by members of a special community, such as governmental institutions, regions, or users with the same commercial interests. They have almost the same characteristics as public clouds, except for the access and authorization required to use the services.

The following diagram illustrates public cloud and community cloud access:



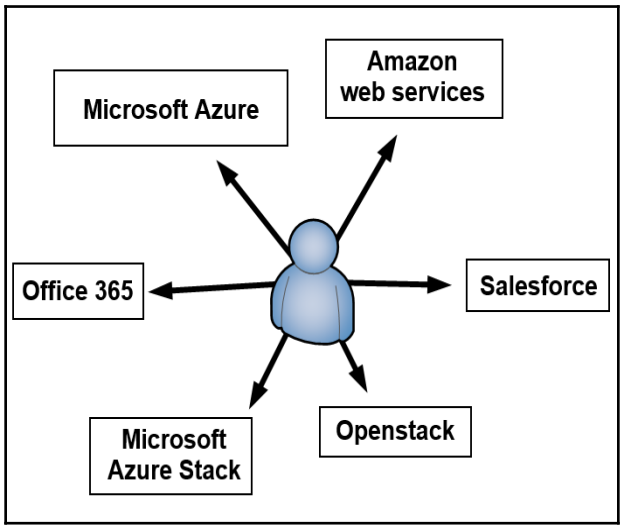
Examples of community clouds include Microsoft Cloud Germany, which is only accessible by companies with a place of business within the **European Free Trade Association** (**EFTA**) region, and Microsoft Azure DoD, which was built for the United States **Department of Defense** (**DoD**) and the governmental institutes attached to the DoD, such as the United States Army or Homeland Security.

Definition of a multi-cloud

To understand multi-clouds, you need to first be aware of what a cloud service is. At this stage of the book, you have already had some insights into the history of cloud computing, cloud models, and cloud characteristics. Now, you should be able to identify the cloud services that you already use in your company, and which ones you might use in the future.

Multi-cloud means that you or your company are not only using the services of one cloud provider, but are also using different solutions from different cloud providers. Examples include using Microsoft Office 365 for business collaboration, Salesforce for customer relationship management, and AWS Route 53 for GeoDNS and GeoIP.

The following diagram shows a person or company surrounded by multiple cloud providers:



Why would we use multiple cloud providers, rather than one provider that fits all? There are different reasons why someone would choose a multi-cloud solution. Let me explain the most common reasons, as follows:

- **Redundancy**: You don't want to build your environment on only one cloud provider, because one can fail, as has happened with AWS in the past. So, you want to keep the business running with the services of the other cloud providers. This is mostly related to using IaaS or PaaS. For the most part, redundancy is not possible with SaaS if the cloud provider does not support hybrid environments.
- **The solution does not fit your needs**: Mostly, when you are choosing a cloud solution, you look at whether it fits your needs. You mostly look at 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. Often, you will see that 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 weak in another. This means that they don't offer the services you want; for example, you might 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 that Salesforce does not offer at the moment, which means that you could include Microsoft Azure Active Directory in your environment to achieve your goal.
- **Your departments use a cloud service as shadow IT:** I have seen shadow IT in almost every company that I have seen over the past 12 years of my work experience. This means that a department uses a solution outside the IT controlled area or solution field, managing the application itself without IT knowing about it. Often, those solutions become business-critical, and C-level management forces IT to take over the solution and provide support for it. In cases of easily accessible cloud solutions, this issue increases dramatically. This is because the IT departments aren't fast enough to deploy an appropriate on-premises solution, or the user thinks, *"OK, I only need a credit card? Let's try it!"*

The key goal in building 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, network, and application infrastructure. That's what we will discuss in the next chapters in this book.

Summary

In this chapter, you learned about the history of cloud computing, as well as its current state. You also learned about cloud models and some characteristics of cloud computing. You should now be able to identify cloud solutions, and you should know the basics of multi-cloud solutions.

In the next chapter, we will discuss the Microsoft Cloud portfolio as a starting point for our coverage of the main cloud providers and solutions.

2
Microsoft Public Cloud Offerings

When most people think of Microsoft, they think of products like Microsoft Windows, Windows Server, or Exchange. In 2008, Microsoft started its cloud journey, and since then, it has changed its whole strategy and company culture to support that cloud strategy. For over ten years now, Microsoft has been evolving its cloud portfolio, and it has become one of the major cloud providers for enterprises around the world.

In this chapter, we will dive into the Microsoft Cloud portfolio, taking a look at the history of Microsoft Azure and Microsoft 365 and discussing the strategies behind Microsoft's cloud services/offerings.

The following topics will be covered in this chapter:

- Microsoft Azure
- Microsoft 365
- The strategy behind Microsoft's cloud offerings
- The Microsoft data centers
- The Microsoft Azure service portfolio
- The Microsoft 365 service portfolio

Microsoft Azure

Microsoft Azure is Microsoft's portfolio of integrated cloud services, built for developers and IT professionals. Azure depends on high bandwidth and low latency global networks, connecting Microsoft data centers around the globe. Every service that Microsoft offers to its customers is delivered out of these Azure data centers and built from Azure services.

Microsoft's journey with Azure began in 2008, when Raymond E. Ozzie, the CTO of Microsoft at the time, announced Windows Azure, during his session at the Professional Developers Conference in Los Angeles. During the conference, Azure was made available as a public preview and began its first services.

In the beginning, Azure was more of a development platform for developers than a full cloud framework, like it is now.

Microsoft 365

As Microsoft's Software as a Service grew, they bundled and rebranded their services and named them Microsoft 365, to match the cloud-first strategy. Microsoft 365 includes different SaaS offerings, like Office 365, Windows, and Enterprise Mobility and Security.

Microsoft's first attempt was a product called Business Productivity Online Suite (BPOS), available for small- and medium-sized customers. Later, in October, 2010, Microsoft rebranded BPOS into Office 365, and made it available to all Microsoft customers.

After that attempt, Microsoft started to offer more products as SaaS solutions, including Microsoft Intune and the Identity Protection Service.

The strategy behind Microsoft's cloud offerings

If you look at Microsoft's strategy and efforts, you will learn that Microsoft always goes by the cloud-first strategy.

Everything that Microsoft builds and develops is first made for Azure and Microsoft's other cloud offerings. Nearly every feature or product that Microsoft develops originates in Azure or M365.

Examples of these cloud-born products are Windows Server 2016 and Exchange 2018. Features such as Storage Spaces Direct, Hyper-V containers, Hyper-V shielded VMs, and storage replication, as well as new features, such as zero downtime patching, were first tested and proven in Azure.

The best example of this technological strategy is Storage Spaces Direct. Microsoft never used SAN in their Azure environment; they still stored everything on the local disk. With the general availability of persistent storage for virtual machines, back in 2012, Microsoft mastered Storage Spaces Direct, and from then on, it has implemented it into the Windows server platform.

Microsoft data centers and global network backbone

With the goal of reaching every Microsoft cloud customer globally within a 22-millisecond network round-trip, Microsoft maintains a large number of data centers and currently has the second largest network backbone in the world.

With regard to Microsoft data centers and regions, Microsoft currently has 54 regions available. Every region consists of two or more data centers.

The Azure region of North Europe, which is located near Dublin, currently depends on 13 data centers, with more than one million physical hosts. If you are interested, Microsoft offers guided tours of some of their data centers to their customers.

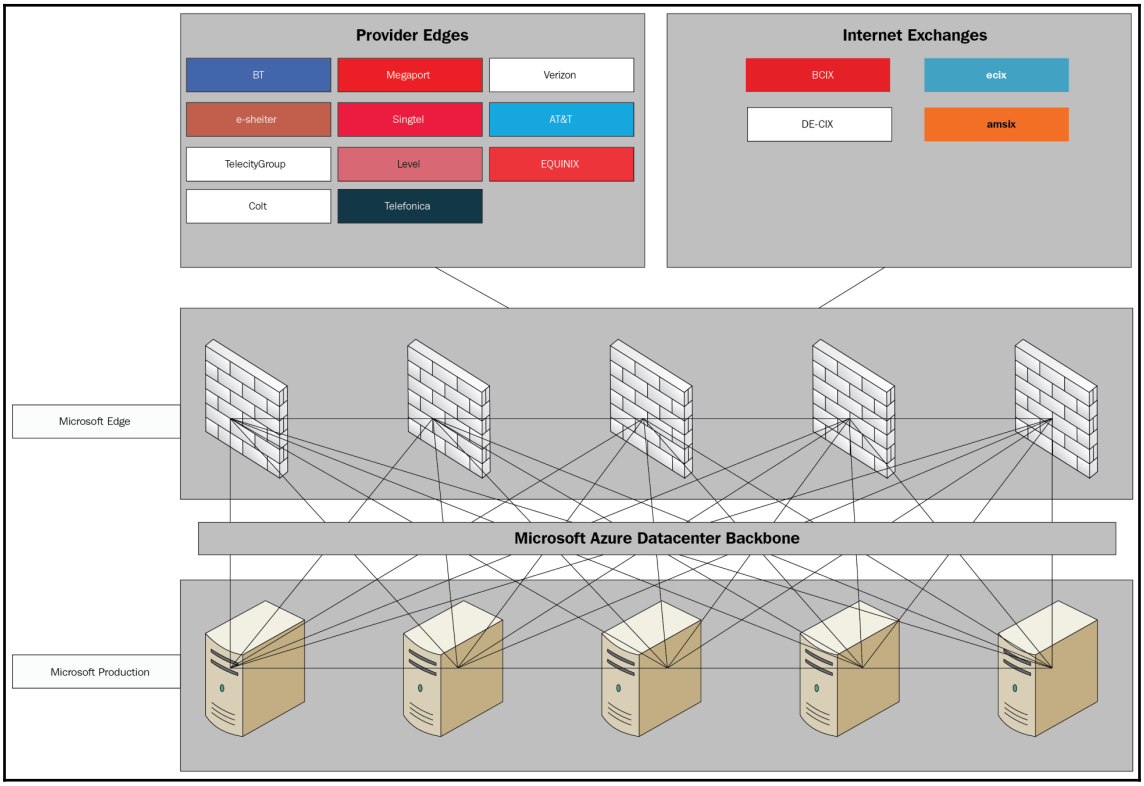
Microsoft divides its regions into geopolitical zones, which are EMEA, the Americas, and Asia.

Currently, there are regions on the map where the data centers are not connected to the Microsoft global data center backbone. Microsoft calls those regions Sovereign Clouds. The Sovereign Clouds were built by Microsoft to fit customer or governmental needs, for things like special compliance and data privacy laws. At the moment, the following Sovereign Clouds are available:

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

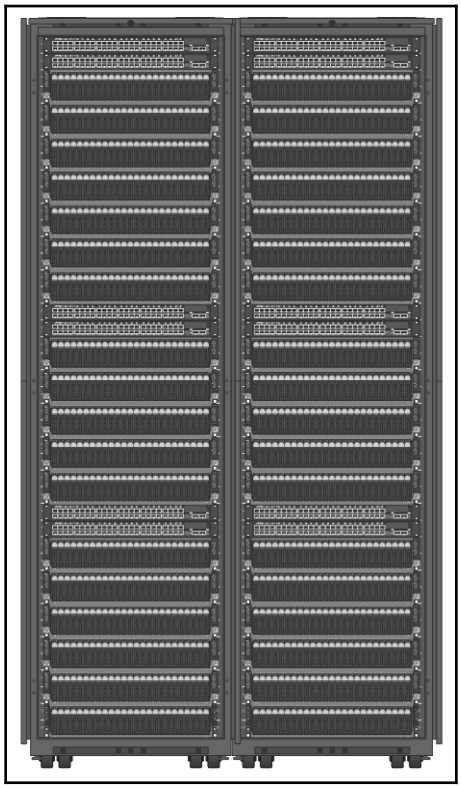
Microsoft Cloud Germany is special among the Sovereign Clouds. Due to customer demands, Microsoft built Microsoft Cloud Germany differently. Microsoft does not operate the cloud in Germany itself; they use a data trustee to operate the cloud for them. Microsoft Azure staff and Microsoft employees are not allowed to enter the data centers or lay hands on the servers or framework. Everything is operated by the trustee, from hardware maintenance to updates of the framework. Currently, Microsoft does not onboard new customers to those regions, having made the announcement of new global regions in Germany.

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

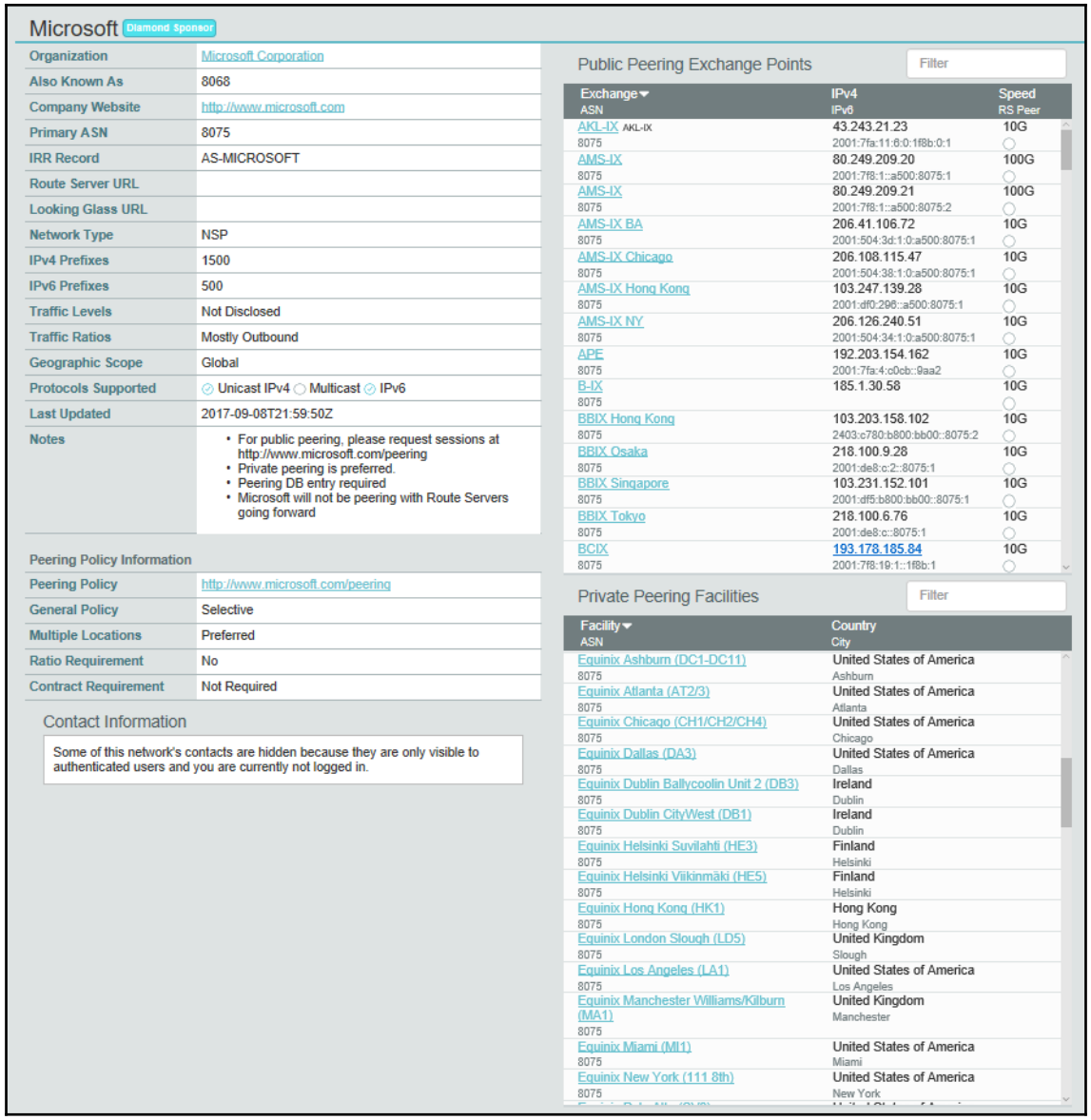


Edge and production sites are connected via the Microsoft backbone, the second largest and fastest full mashed provider backbone in the world, with its own sea cables, like the MAREA cable which runs from Bilbao, Spain, to Virginia, USA.

In building its backbone, Microsoft acts differently than other cloud providers. Microsoft builds its own cables, or leases cables for 20 or more years, from providers like euNetworks GmbH. Microsoft operates the backbone itself, and runs a fully software-defined network and infrastructure for its backbone, using firewall appliances built for network function virtualization. If you ever get the chance to see a server rack that connects to the Microsoft backbone or represents a Microsoft Edge site, it will probably look like the following screenshot:



If you are interested in learning about the data center locations for Microsoft, you can leverage different resources, such as `https://www.peeringdb.com/`. Microsoft's public peering ASN is 8075, and with that, you can get the address of the public and private peering facilities that Microsoft maintains, as shown in the following screenshot:



Microsoft Diamond Sponsor

Organization	Microsoft Corporation
Also Known As	8068
Company Website	http://www.microsoft.com
Primary ASN	8075
IRR Record	AS-MICROSOFT
Route Server URL	
Looking Glass URL	
Network Type	NSP
IPv4 Prefixes	1500
IPv6 Prefixes	500
Traffic Levels	Not Disclosed
Traffic Ratios	Mostly Outbound
Geographic Scope	Global
Protocols Supported	Unicast IPv4 Multicast IPv6
Last Updated	2017-09-08T21:59:50Z
Notes	• For public peering, please request sessions at http://www.microsoft.com/peering • Private peering is preferred. • Peering DB entry required • Microsoft will not be peering with Route Servers going forward

Peering Policy Information

Peering Policy	http://www.microsoft.com/peering
General Policy	Selective
Multiple Locations	Preferred
Ratio Requirement	No
Contract Requirement	Not Required

Contact Information

Some of this network's contacts are hidden because they are only visible to authenticated users and you are currently not logged in.

Public Peering Exchange Points — Filter

Exchange / ASN	IPv4 / IPv6	Speed / RS Peer
AKL-IX AKL-IX 8075	43.243.21.23 2001:7fa:11:6:0:1f8b:0:1	10G
AMS-IX 8075	80.249.209.20 2001:7f8:1::a500:8075:1	100G
AMS-IX 8075	80.249.209.21 2001:7f8:1::a500:8075:2	100G
AMS-IX BA 8075	206.41.106.72 2001:504:3d:1:0:a500:8075:1	10G
AMS-IX Chicago 8075	206.108.115.47 2001:504:38:1:0:a500:8075:1	10G
AMS-IX Hong Kong 8075	103.247.139.28 2001:df0:296::a500:8075:1	10G
AMS-IX NY 8075	206.126.240.51 2001:504:34:1:0:a500:8075:1	10G
APE 8075	192.203.154.162 2001:7fa:4:c0cb::9aa2	10G
B-IX 8075	185.1.30.58	10G
BBIX Hong Kong 8075	103.203.158.102 2403:c780:b800:bb00::8075:2	10G
BBIX Osaka 8075	218.100.9.28 2001:de8:c:2::8075:1	10G
BBIX Singapore 8075	103.231.152.101 2001:df5:b800:bb00::8075:1	10G
BBIX Tokyo 8075	218.100.6.76 2001:de8:c::8075:1	10G
BCIX 8075	193.178.185.84 2001:7f8:19:1::1f8b:1	10G

Private Peering Facilities — Filter

Facility / ASN	Country / City
Equinix Ashburn (DC1-DC11) 8075	United States of America Ashburn
Equinix Atlanta (AT2/3) 8075	United States of America Atlanta
Equinix Chicago (CH1/CH2/CH4) 8075	United States of America Chicago
Equinix Dallas (DA3) 8075	United States of America Dallas
Equinix Dublin Ballycoolin Unit 2 (DB3) 8075	Ireland Dublin
Equinix Dublin CityWest (DB1) 8075	Ireland Dublin
Equinix Helsinki Suvilahti (HE3) 8075	Finland Helsinki
Equinix Helsinki Viikinmäki (HE5) 8075	Finland Helsinki
Equinix Hong Kong (HK1) 8075	Hong Kong Hong Kong
Equinix London Slough (LD5) 8075	United Kingdom Slough
Equinix Los Angeles (LA1) 8075	United States of America Los Angeles
Equinix Manchester Williams/Kilburn (MA1) 8075	United Kingdom Manchester
Equinix Miami (MI1) 8075	United States of America Miami
Equinix New York (111 8th) 8075	United States of America New York

If you are a network provider or you have your own peering ASN and you are connected to an internet exchange, you can request a direct peering with Microsoft ASN 8075, via `https://www.microsoft.com/Peering/PeeringForm`.

The Microsoft Azure service portfolio

Microsoft Azure consists of around 600 products and solutions from Microsoft, and various third-party partners. The categories for those solutions will be covered in the following sections.

Computing

Virtual machines and serverless computing are the services that a cloud provider must offer today. The following list introduces the computing services that Microsoft offers:

- **Virtual machines**: Virtual machines are deployed based on the Windows Server or Linux image in Azure. Customers can select images from a marketplace or use their own customized images.
- **Virtual machine scale sets**: VM scale sets allow us to build highly scalable applications by easily deploying and managing sets of identical VMs.
- **Functions**: Azure Functions is an event driven, compute-on-demand (or serverless computing service) in Azure. Functions allow for using Azure serverless compute resources to build HTTP endpoints accessible by mobile and IoT devices.

Network

The network is one of the most critical services for a cloud provider and cloud customers. Without a network, the cloud can't work. Microsoft is currently the second largest network service provider globally, and with the following services, Microsoft allows customers to leverage and connect to their backbone:

- **Virtual networks**: Azure virtual networks are private networks in Azure that have full control over IPs, DNS, security rules, and traffic. You can connect a virtual network to an on-premises network by leveraging a VPN tunnel or using ExpressRoute for a private connection, via Ethernet or MPLS.

- **Load balancers**: Azure load balancers distribute public (internet) and private network traffic among different service instances in cloud services or virtual machines, in a load balancer group.
- **VPN Gateway**: The Azure VPN Gateway establishes cross-premises connections between virtual networks within Azure and on-premises IT infrastructures, using IPsec security standards.
- **Azure DNS**: Azure DNS allows for hosting DNS domains alongside your Azure apps and managing DNS records by using an Azure subscription. Azure DNS allows you to manage public and private DNS zones.
- **Content delivery network**: This allows for the delivery of high bandwidth content to users around the globe, with low latency and high availability, via a robust network of global data centers leveraging the Microsoft Edge site and peering at the global internet exchanges.
- **ExpressRoute**: Azure ExpressRoute allows for creating private connections between Azure data centers and customers on-premises, or colocation infrastructures, by leveraging last mile networks from connectivity providers like Level 3, euNetworks, Interxion, Equinix, or AT&T.
- **Azure DDoS Protection**: Azure DDoS Protection enables always-on monitoring and automatic network attack mitigation for an Azure-based infrastructure.
- **Network Watcher**: Network Watcher is a service to monitor and diagnose conditions at the network level. Network diagnostics and visualizations with Network Watcher will take packet captures on a VM, if an IP flow is allowed or denied on a virtual machine. It monitors and reports how the packet is routed from a VM, and gives insights into an Azure network topology.

Storage

No Microsoft service is complete without storage. Microsoft offers different kinds of storage solutions for different purposes, as follows:

- **Blob storage**: Object storage for unstructured data, based on REST.
- **Queue storage**: Storage for apps with reactive scaling, according to traffic.
- **File storage**: These are file shares that are used to deliver a standard SMB 3.0 protocol from the cloud. Azure File offers a service called Azure File Sync, which enables a OneDrive or Windows Server distributed filesystem replication or DFS-R like service with an Azure backend.
- **Disk storage**: Secure and persistent disk options for virtual machine storage.
- **StorSimple**: Azure StorSimple is a unique hybrid cloud storage appliance for storage, archiving, and disaster recovery.

- **Azure Backup**: Azure Backup protects data, shared files, folders, and virtual machine workloads.
- **Azure Site Recovery**: Azure Site Recovery protects workloads and applications by the replication and recovery of private clouds, virtualized or server infrastructures for disaster recovery.

Mobile

To enable developers to easily create web and mobile apps, Microsoft offers the following services:

- **App Service**: Azure App Service allows us to create applications for any platform or device and deploy them on a flexible, scalable, and reliable hyperscale infrastructure.
- **Web Apps**: Web Apps allows us to create and deploy web applications. Web Apps is a classic service, and will be replaced by App Service.
- **Mobile Apps**: Mobile Apps allows us to build and host the backend for mobile applications.
- **Logic Apps**: This automates access and data usage across clouds, without having to write program code.
- **API Apps**: This allows us to build and consume cloud APIs.
- **Azure Search**: Azure Search is a manageable service for adding search capabilities to web and/or mobile applications, with no need for the typical complex integration of full-text searches. Azure Search is based on the Microsoft Bing search engine.
- **API Management**: Azure API Management allows for publishing APIs to developers, partners, and employees, in a secure and scalable way.

Databases

Databases are the backbone of many Microsoft services. To enable customers to use these database services, too, Microsoft has started to offer the following services:

- **Azure Database for PostgreSQL**: Azure Database for PostgreSQL provides a managed database service for app developments and deployments that have a need for the PostgreSQL database.

- **SQL Server Stretch Database**: With SQL Server Stretch Database, customers can dynamically stretch warm and cold transactional data from Microsoft SQL Server to Azure.
- **Redis Cache**: Azure Redis Cache is based on the open source Redis Cache, giving customers access to a secure, dedicated cache for Azure applications.

To get a full view of the Microsoft database services, visit the product overviews at `https://azure.microsoft.com/en-us/services/`.

Data and analytics

To support businesses in big data analytics and data warehousing, Microsoft has designed several services, as follows:

- **HDInsight**: Azure HDInsight is a Hadoop-based service that enables Apache Hadoop solutions in Azure.
- **Azure Bot Service**: Azure Bot Service enables the rapid development of intelligent bots. Azure Bot Service combines the Microsoft Bot Framework and Azure Functions.
- **Data Lake Analytics**: The Data Lake Analytics service is a distributed analytics service, built on Apache YARN, to analyze large amounts of data.
- **Data Lake store**: The Data Lake store provides a single-object based repository where data of any size, type, and speed can be captured simply, without changes to applications.
- **Data Factory**: Azure Data Factory is a service that produces trusted information from raw data from cloud or on-premises sources.
- **Power BI Embedded**: Power BI Embedded provides stunning, fully interactive data visualizations in customer-facing apps.
- **Log Analytics**: Azure Log Analytics collects, correlates, and visualizes machine data, such as event logs, network logs, performance data, and so on, from both on-premises and cloud assets.
- **Azure Analysis Services**: This is an enterprise-grade analytics engine.
- **Dynamics 365 for Customer Insights**: This transforms customer data into actionable insights.
- **Event Hubs**: Azure Event Hubs allow for elastic-scale telemetry and event ingestion, with durable buffering and sub-second end-to-end latency for devices and events.

AI and cognitive services

With the Microsoft Cortana suite, Microsoft offers a fully-integratable AI and cognitive service suite:

- **Machine Learning Services**: End-to-end, scalable, trusted AI platform, with experimentation and model management.
- **Machine Learning Studio**: Azure Machine Learning lets customers design, test, operationalize, and manage predictive analytics solutions.
- **Azure Batch AI**: You can train deep learning and other AI models with Batch AI in parallel, using any framework.

Internet of things

Like IBM Bluemix, Microsoft offers a lot of services that are focused on the **internet of things** (**IoT**), like the following:

- **IoT Hub**: Connect, monitor, and control IoT assets running on a wide set of operating systems and protocols.
- **IoT Edge**: Extend intelligence from Azure to edge devices.
- **Time Series Insights**: Instantly explore and analyze time series data in IoT solutions.
- **Event Grid**: Build reactive, event-driven apps with an event routing service.
- **Notification Hubs**: A scalable push notification engine for sending millions of messages.
- **Event Hubs**: Azure Event Hubs allow for elastic-scale telemetry and event ingestion, including durable buffering and sub-second end-to-end latency for devices and events.

Enterprise integration

With Service Bus, Microsoft also provides services that can help customers to integrate their cloud applications into their enterprises:

Service Bus: Azure Service Bus is a messaging infrastructure that operates between applications, allowing for the exchange of messages, for improved scale and resiliency.

Security and identity

With the following points, you will get a little introduction to the security and identity services that Microsoft provides with their cloud services:

- **Key Vault**: Azure Key Vault provides the opportunity for an easy and cost-effective way to safeguard keys and other secrets in Azure, by using **hardware security modules** (**HSMs**).
- **Azure Active Directory**: Azure Active Directory provides identity management and access control applications for users, groups, devices, and applications in Azure and third-party applications.
- **Azure Active Directory B2C (AAD B2C)**: Azure Active Directory B2C is a comprehensive identity management solution for applications that are facing customers and consumers. AAD B2C integrates identities into any platform and makes them accessible from any device.

Developer tools

Microsoft also offers a number of development tools to manage and develop cloud apps, and enables you to easily create a DevOps environment from scratch. Some of these tools are as follows:

- **Azure DevTest Labs**: Azure DevTest Labs is an IaaS solution that allows us to quickly create environments to deploy and test applications.
- **Application Insights**: Detects, triages, and diagnoses issues and behaviors in web apps and services.
- **HockeyApp**: HockeyApp distributes beta versions, collects live crash reports, and gets feedback from users.

Monitoring and management

The following points will give you a brief overview of the services that Microsoft offers for monitoring and management:

- **Azure Advisor**: Azure Advisor is a personalized recommendation engine that analyzes customer workloads and follows Azure best practices, giving advice to optimize those workloads.
- **Azure Resource Manager**: Manages app resources and Azure backend interactions.

- **Automation**: Azure Automation lets customers create, deploy, monitor, and maintain resources in Azure environments automatically.
- **Scheduler**: Azure Scheduler invokes actions that call HTTP/S endpoints or post messages to a storage queue, for any schedule.
- **Azure Monitor**: Real-time monitoring of data for any Azure resource.
- **Cloud Shell**: Azure Cloud Shell is a browser-based shell running in Azure, to help manage Azure resources.
- **Azure Service Health**: This enables particular guidance and support when issues in Azure services affect customers and users.
- **Cost Management**: This enables transparency and accuracy in spending in Azure.
- **Azure Policy**: This turns on built-in policies, or builds custom policies to enable security and management.

The Azure Marketplace

Through the Azure Marketplace, Microsoft enables their partners to offer their solutions to end users, to deploy then in Azure. Those solutions can be simple application offerings, like WordPress or Apache Server, as well as highly complex applications, like SAP, network appliances, or Citrix deployments.

The following are some examples of Azure Marketplace apps and solutions:

- Windows Server 2008, R2 SP1
- Web App
- Virtual Network
- SQL database
- CHEF
- Barracuda NextGen firewall

To stay up to date on every current Azure solution, you can visit the Azure Service catalog at `https://azure.microsoft.com/en-us/services/`.

The Microsoft 365 service portfolio

As we mentioned earlier in this chapter, Microsoft has bundled different services together, with the name Microsoft 365. In this section, we will give you a little introduction to those services.

Microsoft Office 365

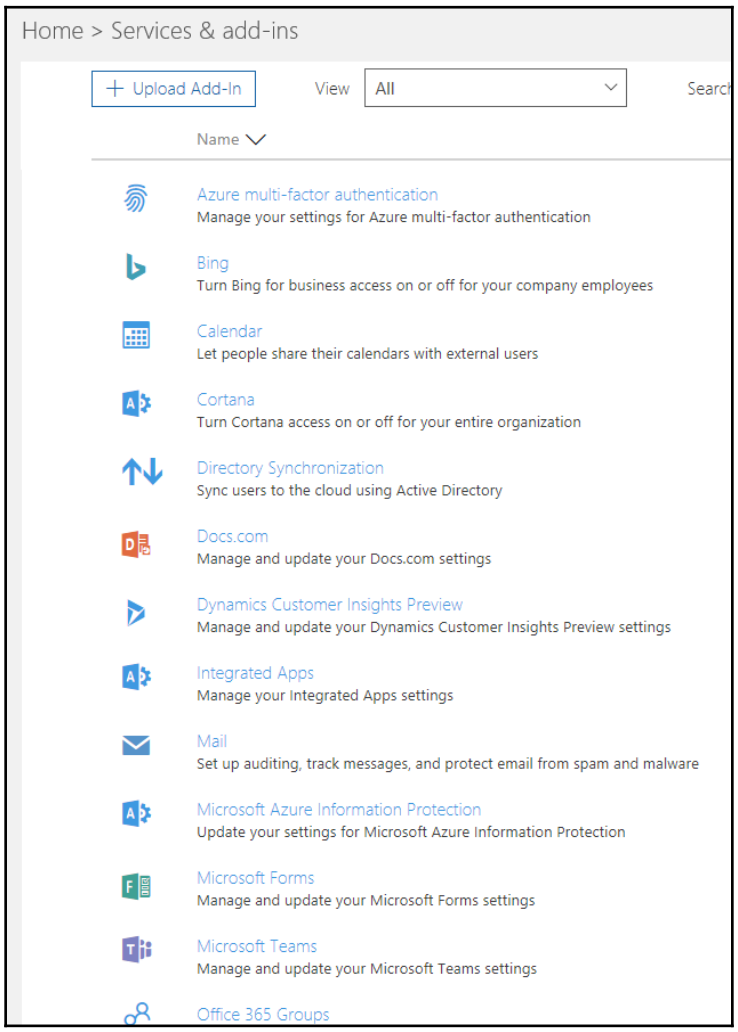
Microsoft Office 365 is a cloud-based Software as a Service offering that is designed to provide customers with Microsoft application solutions, like Exchange, SharePoint, Project, Office products, and so on, hosted by Microsoft.

The services offered by the Office 365 portfolio are as follows:

- **Office 365 Platform Service**: The Microsoft platform and frontend to deliver and maintain the Office 365 service.
- **Office Applications service**: The service that provides the latest version of the Office desktop applications, such as Word, Excel, and PowerPoint.
- **Office Online service**: Office Online (previously known as Office Web Apps) opens Word, Excel, OneNote, PowerPoint, and Visio documents in a web browser, and enables customers to read or edit Office documents without an installed application.
- **Exchange Online service**: Microsoft Exchange Online is a hosted messaging solution that delivers Microsoft Exchange Server as a cloud-based service.
- **Exchange Online Archiving service**: Microsoft Exchange Online Archiving is an archiving solution for Microsoft Exchange Server 2013, Microsoft Exchange Server 2010 (SP2 and later), or Exchange Online.
- **OneDrive for Business service**: OneDrive for Business (formerly SkyDrive Pro) is a personal home drive, online and in the cloud. Users can store work files across multiple devices with ease and security. Users can share files with business colleagues and edit Office documents together. Users can also sync files with a local computer, using the OneDrive for Business sync app.
- **SharePoint Online service**: With Microsoft, SharePoint Online organizations can create and manage custom, team-focused, and project-focused sites, for collaboration.
- **Skype for Business Online service**: Skype for Business Online gives access to information about user presence, and enables instant messaging, audio and video calling, online meetings, and web conferencing services.
- **Workplace Analytics service**: Microsoft Workplace Analytics, or DELVE, analyzes how your teams work together, so that behaviors can be shown and possible improvements for individuals or teams can be identified.
- **Power BI service**: Power BI is a suite of business analytics tools that enables data visualization and analytics.
- **Project Online service**: Project Online is a flexible, online solution for project management.

- **Yammer service**: Yammer is a secure and private enterprise social network.
- **Microsoft Dynamics 365 (online) service**: Dynamics 365 combines **Customer Relationship Management** (**CRM**) and **Enterprise Resource Planning** (**ERP**) into one cloud service.

Office 365 offers customers additional Microsoft third-party services, as shown in the following screenshot:



Windows 10 as a service

Having changed Windows 10 to the current branch update cycle, Microsoft has turned your Windows operating system into a cloud service, as well. In terms of delivering a cloud service, it's not what you might think; Microsoft is actually moving away from selling software licenses. Microsoft is also starting to sell Windows as a subscription-based service, like they already do with Microsoft Office.

Enterprise mobility and security

As we explained earlier in the chapter, the EM+S Suite is combined with different Microsoft cloud products; the following list provides a brief introduction to these products:

- **Microsoft Identity Manager (MIM)**: Microsoft Identity Manager 2016 builds on identity and access management, like the former version, called **Federal Identity Manager** (**FIM**) 2010 R2.
- **Cloud App Security**: Cloud App Security is a comprehensive solution that allows for taking full advantage of cloud applications, but keeps the owner in full control, through improved visibility into all of the activities in relation to the application. It also increases the protection of critical data across cloud applications.
- **Microsoft Intune**: Intune is a cloud-based service in **Enterprise Mobility Management** (**EMM**), like System Center Configuration Manager (as an on-premises product).

Microsoft EM+S also includes the Azure Active Directory Premium 1 plan.

Microsoft Trust Center

Microsoft is very open when it comes to their cloud environment; they have built a collection on a website they call the Trust Center. Here, Microsoft provides valuable information about **Corporate, External, and Legal Affairs** (**CELA**), data center and application certifications, and any necessary information for compliance and security.

Microsoft Trust Center for Azure: `https://www.microsoft.com/en-us/trustcenter/default.aspx`

Microsoft Trust Center for Office 365: `https://products.office.com/en-us/business/office-365-trust-center-welcome`

Summary

In this chapter, you learned about the Microsoft cloud portfolio and got some insight into the Microsoft Azure infrastructure. You also saw how a Microsoft data center operates, and how Microsoft builds up their data centers. You should now know most of Microsoft's Azure portfolio, and should have more insight into their data center operations.

With that knowledge, you should be able to compare Microsoft cloud solutions to other providers, and to find services that fit your needs.

In the next chapter, we will discuss the Amazon Web Services cloud portfolio, the second of the main cloud providers that we will cover in this book.

3
Amazon Web Services and Its Public Cloud Service

Amazon Web Services, or **AWS** as it is popularly known, is the world's leading cloud services provider. It has consistently appeared as a leader in the Gartner Magic Quadrant report for cloud infrastructure. AWS has grown from offering a few services in 2006, to offering more than 1,000 services today. It has consistently introduced new services, keeping in mind the demands of the market, and it has also consistently reduced their price, thereby ensuring that more and more people migrate to the cloud. AWS now incorporates various generic as well as specialized areas, such as computing, storage, databases, big data, IoT, mobile development, machine learning, and others. AWS also offers a free tier for those who want to try out the cloud before migrating to it. AWS is currently seen as the market leader in the **Infrastructure as a Service** (**IaaS**) space and is considered to be way ahead of its rivals in terms of its market share.

We will cover the following topics in this chapter:

- The regions and availability zones of AWS
- AWS's edge locations
- Details of the various services offered by AWS

Regions and availability zones

To understand how AWS provides its services, we need to understand the concept of regions and availability zones. AWS has its infrastructure spread across the world in various locations. A **region** is a geographical location—for example: **Oregon** is a region, **Frankfurt** is a region, and **Tokyo** is a region. These are called **geographical regions**. AWS also has another kind of region, called a **local region**. A local region is a single data center that will complement a geographical region. Local regions are mainly used by clients who want to replicate their data or application over larger geographical distances. At the time of writing, AWS infrastructure is available in 19 geographical regions and one local region. Each region is completely independent and isolated from the other regions.

Each region consists of multiple availability zones. Each availability zone is physically separated and isolated from the others. All availability zones within a region are connected to one another via highly redundant, low-latency, high-speed networks. Architects can use the fact that availability zones are isolated from one another and design their infrastructure in such a way that it is highly available and fault-tolerant. Replicating this across different availability zones in a geographical region will help to increase redundancy and fault tolerance. The number of availability zones present in each geographical region varies, but there is a minimum of two availability zones per geographical region. If there is a local region, there is only a single availability zone present.

The following table shows the regions and the n**umber of availability zones** that each contains:

Region	Num of Availability Zones
Virginia	6
Oregon	3
N.California	3
Ohio	3
Paris	3
Tokyo	4
Singapore	3
Mumbai	2
Ireland	3
Frankfurt	3
London	3
Seoul	2
Beijing	2
Ningxia	3
US Gov Cloud (US West)	3
US Gov Cloud (US East)	3
Sao Paulo	3
Stockholm	3
Sydney	3
Osaka(local)	1
Canada Central	2

The following are the new regions that will be coming soon:

- Bahrain
- Hong Kong (SAR, China)
- Sweden
- AWS GovCloud (US East)

AWS Edge Locations

AWS is also a **Content Delivery Network** (**CDN**) provider. To deliver content to end users from a location close to them, AWS has a network of edge locations spread out across the world. These locations help with content delivery and ensure that the content is delivered to end users with low latency. This Amazon service is called CloudFront. CloudFront uses a global network, which is spread across 56 cities in 24 countries.

AWS services

AWS offers more than 1,000 services, and it keeps adding services at regular intervals. The way AWS has gone about building its portfolio is very impressive. AWS has been expanding in areas where it is the leader, and it has also expanded into newer territories. In the following sections, we will try to give you an idea of the various services offered by AWS and to whom they would be useful.

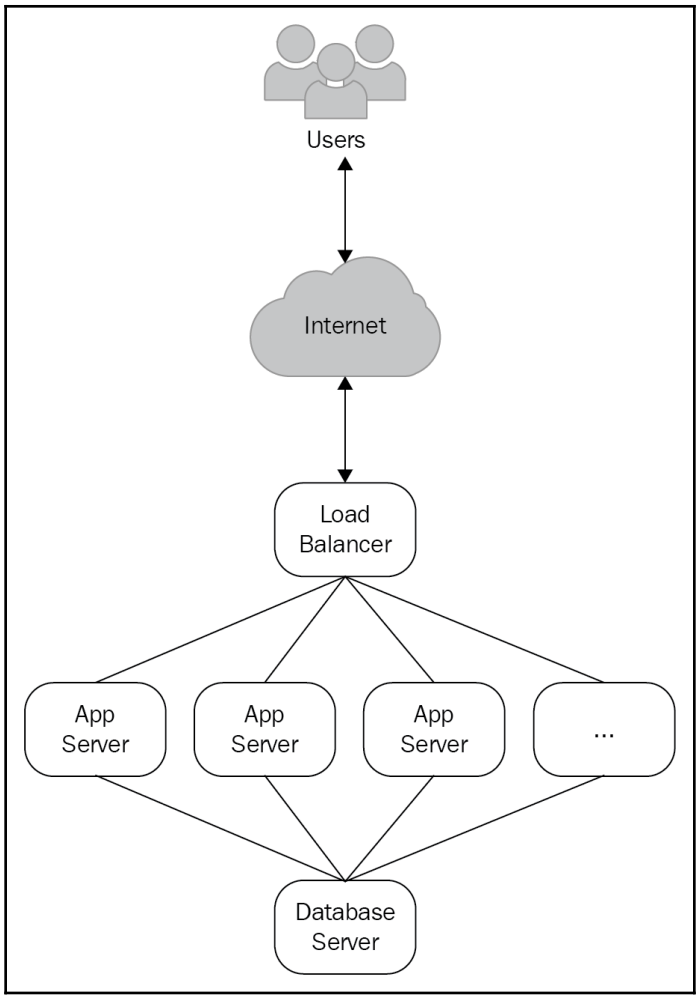
AWS compute services

A **compute service** is one of the most generic services expected from any cloud. Amazon has the basic EC2 service, which provides a computer on the cloud, along with a range of other compute services. We will look at each one in detail in the following list:

- **Elastic Compute Cloud (EC2)**: Elastic Compute Cloud, more commonly known as EC2, is one of the most important and most-used services of AWS. EC2 provides a virtual machine to the users on the cloud, called an **instance**. EC2 also provides various instance types for a variety of workloads. An instance type determines the number of vCPUs, the amount of memory and storage, and the network speed. Users can select an instance type based on their application's needs:

  - The instance could be single-tenanted or multi-tenanted. By default, AWS provides a multi-tenanted instance. A dedicated instance and a dedicated host are two options available for obtaining a single-tenanted instance.
  - EC2 also has a bare-metal offering. In this case, the user will be provided with a bare-metal computer and not a virtual machine. The user will have complete control over the hardware.

- **AWS Lambda**: Lambda is the serverless computing service of AWS. In serverless computing, the user code is run without the user having to provision any hardware or manage any servers. Lambda allows the user code to run when required, with the provisioning and scaling of servers being taken care of by AWS.
- **Amazon Lightsail**: Lightsail offers a **virtual private server** (**VPS**). We can view Lightsail as a prepackaged EC2 instance. Lightsail provides a VPS, which is a virtual machine with an SSD drive of fixed capacity, free data transfer allowance, a static IP, and DNS management. This is provided at a set monthly fee. Running websites, web applications, blogs, and e-commerce sites are some of the use cases of Lightsail with AWS.
- **AWS Elastic Load Balancer**: AWS provides users with three types of load balancers: Classic Load Balancer, Network Load Balancer, and Application Load Balancer. The role of the load balancer is shown in the following diagram:



- **Classic Load Balancer**: This was the first load balancer provided by AWS. It is a basic load balancer that balances the load across the multiple EC2 instances attached to it. Before the advent of **Virtual Private Cloud** (**VPC**), Amazon had the EC2-Classic network. Amazon recommends this load balancer for use in the EC2-Classic network.

- **Network Load Balancer**: This is a layer-4 load balancer, which has high throughput and low latency, and is capable of handling millions of requests per second. The targets for this load balancer could be EC2 instances, Docker containers, and IP addresses. This load balancer uses a static IP per availability zone. It is integrated with various AWS services
- **Application Load Balancer**: This is a layer-7 load balancer. This load balancer can route traffic to the target, based on the content of the request. Typical targets include EC2 instances, Docker containers, and IP addresses. This load balancer is used to load balance HTTP and HTTPS applications.

AWS storage services

Storage is another generic service that is expected from all cloud vendors. AWS provides services for block storage, file storage, and object storage. Additionally, AWS also provides Storage Gateway solutions, which are hybrid cloud solutions. We will look at each one in detail in the following list:

- **Simple Storage Service (S3)**: Simple Storage Service, or S3, as it is popularly known, was one of the first services introduced by Amazon for the general public. S3 is an object-based storage service. You can store an unlimited number of objects in S3. It has multiple classes of storage, each of which has a different cost. The standard class has the highest SLA of 11 nines of durability and is designed for four nines of availability. Objects in S3 are accessed using REST APIs. S3 is a natural destination for cloud backup solutions.
- **Glacier**: Glacier is an archival storage solution from Amazon. It is a very low-cost solution, with the cost per GB being less than a cent for long-term storage. Objects in Glacier have a durability of 11 nines. AWS offers different levels of retrieval: expedited, standard, and bulk. Expedited retrieval retrieves objects within 1-5 minutes, and the standard is between 5 and 12 hours, with the bulk taking more than 12 hours to retrieve an object. Each retrieval type has a different cost.
- **Elastic Block Store (EBS)**: EBS is a block storage solution from AWS. EBS disks, also called **volumes**, are disks that are attached to EC2 instances. EBS volumes come in different varieties: throughput optimized, magnetic, general-purpose SSD, and provisioned IOPS. The maximum EBS volume size is 16 TB. EBS volumes support snapshots for backing up. All snapshot backups are stored in S3. Volumes can be attached and detached at any point in time.

- **Instance Store**: This is another block storage service from AWS. It is temporary block-level storage for the instance, where the storage is provided by the disks that are physically attached to the bare metal on which the VM resides.
- **Elastic File System (EFS)**: EFS is a file storage service from AWS. EFS is based on NFS v4.1, and instances that run Linux can concurrently mount EFS. EFS has no space limits and can store an infinite number of files. Computers in data centers can mount EFS over Direct Connect.
- **Storage Gateways**: Storage Gateways is a hybrid cloud solution, enabling on-premises equipment to connect to AWS storage. Storage Gateway can be installed as a **Virtual Appliance** (**VA**) on-premises, or it can be deployed on an AWS EC2 instance. There are three types of Storage Gateways: a volume gateway, a file gateway, and a tape gateway. These are outlined in the following list:
    - **Volume gateway**: A **v**olume gateway provides cloud storage as iSCSI volumes that can be mounted by application servers in the data center. Volume gateways come in two types: cached volume gateways and stored volume gateways. In a cached volume gateway, the data is stored in Amazon S3. The local disk stores a cache of the data for faster access. In a storage volume gateway, data is stored locally, and at regular intervals a snapshot of the data is taken and stored in S3.
    - **File gateway**: A file gateway allows you to access files from S3 using NFS v4.1. In other words, the file gateway provides a file interface to S3, allowing the application servers in the data center to store and retrieve objects to and from S3, using a file interface.
    - **Tape gateway**: A tape gateway provides a **Virtual Tape Library** (**VTL**) interface and allows the files to be stored in Glacier for archival purposes.

AWS database services

Another important service that all cloud vendors offer is a database service. The AWS database service includes relational databases, NoSQL databases, in-memory databases, and data warehousing. Let's cover each of these services in detail by going through the following list:

- **Relational database service (RDS)**: Amazon RDS is a managed database service. AWS sets up the VM with a database of your choice, performs regular backups of the database, and allows users to scale CPU, memory, storage, and IOPS seamlessly.
    - RDS can set up databases such as MySQL, MariaDB, PostgreSQL, Oracle, Amazon Aurora, and Microsoft SQL Server.
    - RDS can be set up in a multi-availability zone mode. This will ensure that the database is fault tolerant and highly available. AWS RDS performs a synchronous replication and automatic failover in multi-availability zone mode.
    - RDS also supports read replicas for MySQL, MariaDB, Amazon Aurora, and PostgreSQL.
- **DynamoDB**: DynamoDB is the NoSQL database solution from AWS. This is a fully managed service where AWS sets up, provisions, scales, replicates, and backs up the database. All data is stored on SSD and spread over multiple servers to provide the required performance. The data in the SSD is replicated across multiple availability zones automatically. This ensures that the data is highly available. All data can be backed up on demand. DynamoDB tables can store and retrieve any amount of data. The user can scale up or scale down the throughput capacity requirement at any point in time.
- **Elastic Cache**: AWS Elastic Cache is a distributed, in-memory caching service. As with other database services, AWS takes care of setting up the in-memory cache. Elastic Cache also scales the cache whenever required. Redis and Memcached are the two engines that are supported by Elastic Cache.
- **Amazon RedShift**: If you require a petabyte-scale data warehousing solution, then Amazon RedShift is an option for you. RedShift is a petabyte-scale, fully managed data warehousing solution from AWS. RedShift takes care of all the complexity involved in setting up the petabyte-scale data warehouse and enables users to analyze their data by using their own business intelligence tools.

Networking

The ability to build their own network in the cloud would be a major requirement for all enterprises. In addition to this, multiple methods of connecting to the cloud are also desirable. AWS offers a virtual private cloud, as well as Direct Connect for its users. AWS also offers DNS services and CDN services. Let's look at each of these in detail by going through the following list:

- **VPC**: **Virtual Private Cloud** (**VPC**), is one of the most important constructs in AWS. VPC allows you to replicate an on-premises-like data center structure within AWS. Each VPC is region-specific and is contained within a region. VPC allows users to create subnets, with each subnet being contained within an availability zone. Subnets can be designed to be accessible from the internet. These are called public subnets. Subnets that cannot be accessed from the internet are known as private subnets. Public subnets are connected to internet gateways provided by AWS, and this enables access from the internet. Private subnets can access the internet via network gateways for IPv4 connections and via egress-only gateways for IPv6 connections.
    - AWS also provides a VPN gateway for VPC. This allows the connection between an on-premises network and AWS VPC. VPCs can be connected to one another through a process called **VPC peering**. VPC peering works across regions—that is, VPCs in two different regions can be connected to each other.
    - AWS provides for VPC endpoints. A VPC endpoint enables instances within a private VPC to access AWS services such as S3 and DynamoDB (which have an internet endpoint) through the internal network of AWS. This means that EC2 instances in a private network can access S3 or DynamoDB without the help of a NAT gateway.
    - The security of the subnets is provided via **Network Access Control Lists** (**NACL**). An NACL functions as a firewall at the subnet level. Additional instance-level security is provided by security groups, which are firewalls at the instance level.

- **Direct Connect**: **Amazon Direct Connect** (**ADC**) is used to establish a private dedicated connection between an on-premises network and the AWS network. This helps to improve network performance and reduces bandwidth cost. Users also get a much better network experience compared to connecting over the internet.

- **Route 53**: Route 53 is a highly available and scalable **Domain Name System** (**DNS**) service. Route 53 helps users with the following three tasks:
    - Registering the user domain
    - Traffic routing from the internet to the user's domain
    - Performing a health check of user resources

  Route 53 also provides the following routing policies:

    - Simple
    - Weighted round robin
    - Failover
    - Latency
    - Geolocation
    - Geoproximity
    - Multivalue answer



Since Route 53 is a DNS service, it can also be configured to act as a global load balancer by adopting suitable routing policies.

- **CloudFront**: CloudFront is a CDN solution from AWS. It is mainly used to speed up the delivery of static and dynamic web content. Its content could be video files, audio files, image files, and so on. Delivery of the content is done via CloudFront edge locations. When data is requested by a user, the user is routed to the nearest edge location so that data can be served from that location, resulting in low latency.



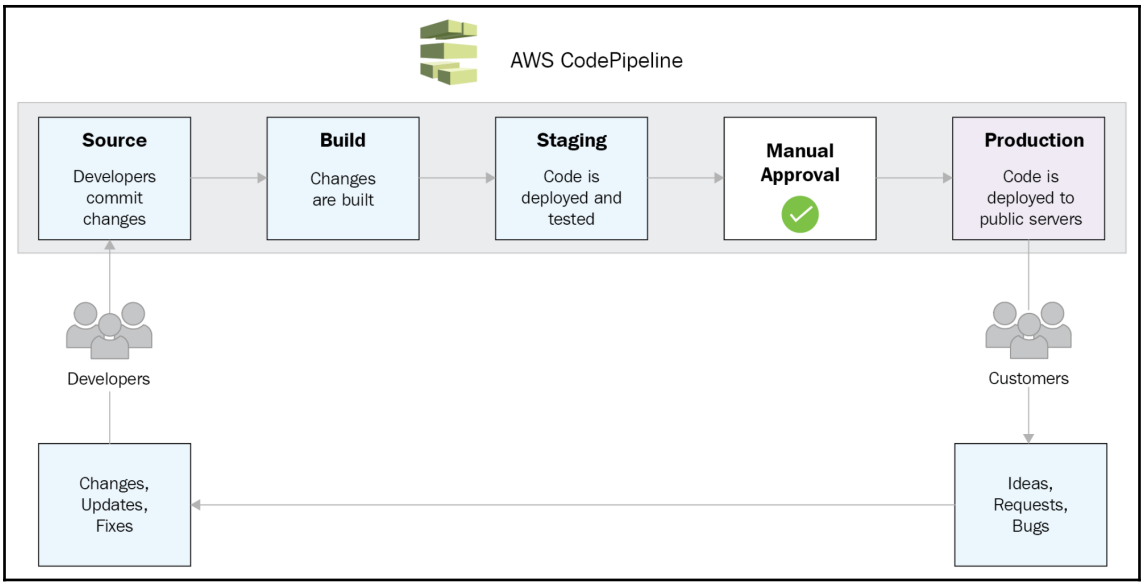
If the location already has the data, then data is delivered to the user from that location; otherwise, the edge location fetches the data from the origin server and delivers the data to the user. It also caches the data, which can then be served for later requests.

DevOps

AWS has many services that can help with DevOps. Though the AWS console itself doesn't categorize them as DevOps, we will put them in this category, to understand them better. The following list shows the services in more detail:

- **AWS OpsWorks:** The configuration management service of AWS is called **OpsWorks**. There are three modules in OpsWorks:
    - **AWS OpsWorks for Puppet Enterprise:** Puppet works in the client-server mode. AWS OpsWorks for Puppet Enterprise creates a puppet master server for the user and manages the master server. The master server stores all the puppet modules, modules being instructions for how the infrastructure must be configured. Once the master server is set up, all the client nodes can be configured. The client nodes could be EC2 instances or on-premises equipment.
    - **AWS OpsWorks for Chef Automate**: Chef also works in client-server mode. AWS OpsWorks for Chef Automate creates a Chef master server and manages the master server. These servers include the Chef Automate premium features in them. Like modules in Puppet, Chef has cookbooks that help with configuring the infrastructure. The nodes in the infrastructure, as well as the cookbooks on the Chef server, can be managed using Chef tools such as Knife and Test Kitchen.
    - **AWS OpsWorks Stacks**: A group of resources that are needed to run an application is called a **stack** in AWS OpsWorks parlance. For example, to run a web application, we may require EC2 instances, database instances, load balancers, and so on. When all these resources are gathered together, they are called a stack. AWS OpsWorks is a service that allows users to create, manage, and destroy stacks and applications. Users can deploy and monitor their application on stacks using AWS OpsWorks Stacks. Within the stack, we can create multiple layers. Each layer is a grouping of resources that achieves a well-defined aim. For example, users can have a database layer for the database servers and a web layer containing the load balancer and the web servers. Layers use Chef recipes for various tasks, such as installing software, deploying applications, running scripts, and so on.

- **Elastic Beanstalk**: This is a **Platform as a Service** (**PaaS**) feature from AWS. Elastic Beanstalk allows users to quickly deploy applications written in PHP, Python, Ruby, Node.js, Java, and .NET, without worrying about provisioning the hardware. The users just have to upload their application while Elastic Beanstalk takes care of provisioning the instances, scaling instances when required, provisioning load balancers, and monitoring the health of the application.
- **CloudFormation**: This is a template-based resource-provisioning solution. Users create a template, providing the details of all the infrastructure items they need. CloudFormation then uses this template and provisions and configures the resources that are requested. This allows for repeatable and dependable infrastructure deployment.
- **AWS CodeCommit**: This is a version-control service from AWS. This is a scalable and secured source control system for hosting private Git repositories.
- **AWS CodeBuild**: This managed build service from AWS eliminates the need for users to maintain their own build servers. Instead, they can use build servers on the cloud utilizing the services of AWS CodeBuild. AWS CodeBuild compiles the code, performs unit tests, and generates the final artifacts. This service also takes care of provisioning, scaling, and maintaining the build servers. Prepackaged build environments for most programming languages and build tools, such as Apache Maven, Gradle, and so on, are available.
- **AWS CodeDeploy**: This service enables automated application deployments. The applications can be deployed on EC2 instances, on-premises instances, or as serverless Lambda functions. You can deploy various types of application content, including executables, code, configuration files, packages, scripts, multimedia files, and so on.
- **AWS CodePipeline**: **Continuous Integration and Continuous Deployment** (**CICD**) is an important part of DevOps. AWS CodePipeline helps in providing continuous delivery. You can use CodePipeline to automate the steps required for the release of software. For example, we can build a pipeline that integrates Git, AWS CodeBuild, and AWS CodeDeploy so that any new commit to Git is automatically built and deployed immediately:

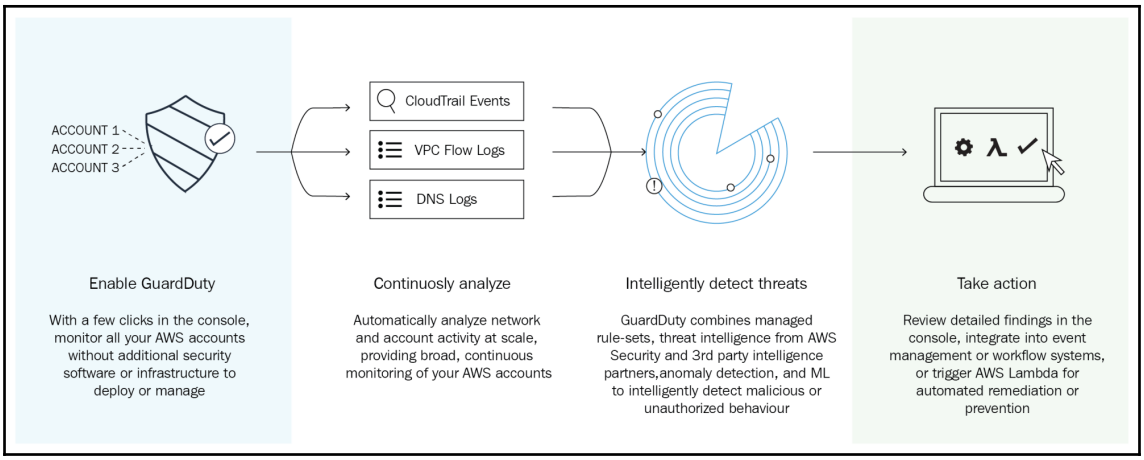


- **AWS Cloud9**: This is a cloud-based IDE that helps users to write, debug, and run their code, using only a browser. Users do not need to install any local IDE, as you get a code editor, a debugger, and a terminal with AWS Cloud9. More than 40 programming languages, including popular ones such as Python, PHP, and JavaScript are supported by AWS Cloud9. AWS Cloud9 connects to AWS EC2 instances to run code.
- **AWS CodeStar**: This AWS service allows users to create and manage software development projects on AWS. Users can quickly develop, build, and deploy applications on AWS using CodeStar. CodeStar provides project templates that can be used to quickly build a project. Each template contains the tools and infrastructure required for the development, building, and deployment. These include source control, build servers, and deployment servers. CodeStar is integrated with other AWS tools, such as CodeCommit and CodePipeline.

Security, identity, and compliance

Security in the cloud is a major concern for all cloud users. Managing the identity of users, encryption in flight and at rest, using certificates for websites, and mitigating **Distributed Denial of Service** (**DDoS**) attacks are just some cloud users' security requirements. The following list shows the AWS services for each of these areas:

- **Identity and Access Management** (**IAM**): It is important that only authorized users are able to access AWS Services. IAM helps with managing access to AWS resources. Using IAM, users and groups can be created. Fine-grained access policies can be applied to users and groups, **Multifactor Authentication** (**MFA**) can be enabled, and federated users can be managed.
- **AWS Key Management Service** (**KMS**): KMS allows users to create and manage encryption keys. Once created, users can use the key to encrypt data. To protect keys securely, KMS uses FIPS 140-2 validated hardware modules. Multiple AWS services, such as EBS, S3, DynamoDB, and so on, are integrated with KMS.
- **AWS CloudHSM**: This is a cloud-based **hardware security module** (**HSM**). Users can generate and use their own encryption keys on a FIPS 140-2 Level-3 compliant HSM. CloudHSM supports industry-standard APIs such as PKCS#11, **Java Cryptography Extensions** (**JCE**), and Microsoft **CryptoNG** (**CNG**) libraries.
- **AWS WAF and Shield: WAF** stands for **Web Application Firewall**. The AWS WAF monitors the HTTP and HTTPS requests that are sent to CloudFront and the load balancers. The WAF can then be used to allow or block requests. Shield is used to minimize the effects of DDoS attacks.
- **AWS Secrets Manager**: This security and compliance service allows users to manage their secrets. Secrets could be API keys, database credentials, and so on. Using AWS Secrets Manager, users can rotate secrets safely. Users and applications can retrieve secrets with an API call to this service.
- **AWS Guard Duty**: This is an intelligent threat-detection system. This service continuously monitors your AWS infrastructure to check whether your account has been compromised. It detects both unusual API calls and unauthorized deployments. When a threat is detected, a security alert is raised and delivered to CloudWatch Events. This process is shown in the following diagram:



- **AWS Inspector**: This is an automated security assessment service. AWS Inspector checks your application for any security vulnerabilities. It also checks for any deviations from security best practices. Once the assessment is done, a detailed report of the vulnerabilities and deviations is produced. This gives security auditors an insight into the security testing that has been done, thereby helping to demonstrate compliance with security standards.
- **Amazon Macie**: Using machine learning, Amazon Macie discovers, classifies, and protects sensitive data in AWS. As of now, Macie is available to be used on data stored in Amazon S3. Once sensitive data is identified, Macie monitors it continuously, generating alerts when it perceives that there has been any authorized access risk or data leak.
- **AWS Certificate Manager**: This service provides users with the ability to provision, manage, and deploy SSL/TLS certificates for use in AWS services and the users' internally connected resources.
- **AWS Directory Service**: This service is a managed Microsoft AD server on the AWS cloud. Applications that integrate with Microsoft AD can make use of this service.

Monitoring and management tools

Monitoring infrastructures on the cloud is another important task for all system operators. In addition, they will also need to keep track of all activities, that are performed for compliance purposes. Let's take a look at how AWS supports users' efforts in monitoring and managing:

- **CloudWatch**: Cloudwatch is the most important monitoring tool of AWS. CloudWatch is a metrics repository. It collects metrics for all AWS services utilized by the user, and it stores the metrics for a certain period of time. Users can define and store custom metrics using CloudWatch. CloudWatch allows users to generate alarms based on these metrics, and it also monitors the system to check for **events** (state changes to the infrastructure) and allows users to build workflows to mitigate against these events. It is also responsible for log management and storage.
- **CloudTrail**: As the name indicates, CloudTrail is a service that shows us the trail of all API usage within our account. Any action taken from the console or CLI, or using SDK, is logged. This ensures that we get to see the complete history of all the actions taken in our account. It gives us information about who took what action against which resource and when the action was taken. This service helps organizations with their governance, auditing, and compliance.
- **AWS Config**: This service continuously monitors the system to check for changes to the configuration of our infrastructure. All configurations are recorded. This enables users to check how the current configuration deviates from the desired configuration. This allows users to check how the compliance of the configuration conforms to the internal guidance, thereby helping with compliance auditing and security analysis.
- **AWS Trusted Advisor**: This AWS service checks your infrastructure and provides advice on how to optimize cost, improve security, and increase performance.

Integrating an application

AWS allows developers to build distributed applications based on AWS services, using the SDK provided by AWS. The following list shows the services that help developers with their application's integration:

- **Simple Queue Service (SQS)**: Queues are used by developers building distributed systems. Queues help developers to decouple the various components of the system. This loose coupling of components helps with scaling the system. SQS is a hosted queue service from AWS that is highly available. SQS offers a standard queue that does not guarantee that messages will be delivered in sequence and a **First-In-First-Out** (**FIFO**) queue that guarantees the sequential delivery of messages. SQS also has dead-letter queues.
- **Amazon MQ**: Apache ActiveMQ is a popular open source messaging system. Amazon MQ provisions the required hardware and manages ApacheMQ on its servers. Users can use the industry standard APIs for messaging, such as JMS, NMS, AMQP, STOMP, MQTT, and WebSocket.
- **Simple Notification Service (SNS)**: SNS is a **publisher–subscriber** (**pub–sub**) messaging system, as well as a mobile notification service. Users subscribe to certain topics, using endpoints of their choice and SNS coordinates delivery to those subscribed endpoints. SNS can also be used to fan out messages to a large number of subscribers.
- **Simple Workflow Service (SWF)**: One of the important challenges that developers face when developing a distributed system is deciding how to coordinate work across multiple components of the system. SWF helps with coordinating work across distributed components, helping developers to quickly develop their application.
- **AWS Step Functions**: We saw how SWF coordinates work across the distributed components of a system. AWS Step Functions also helps with coordinating work across distributed components and microservices, using visual workflows. AWS Step Functions provides a graphics console. Users can use this console to visualize their application components as a series of steps. AWS Step Functions is a part of the AWS Serverless platform. This makes it easy to orchestrate AWS Lambda functions.

Analytics

These days, data analytics is a very important tool for all enterprises. The ability to analyze large amounts of data and/or the ability to analyze streaming data has a great bearing on the growth of a business and its profitability. AWS provides various tools for data analysis, as shown in the following list:

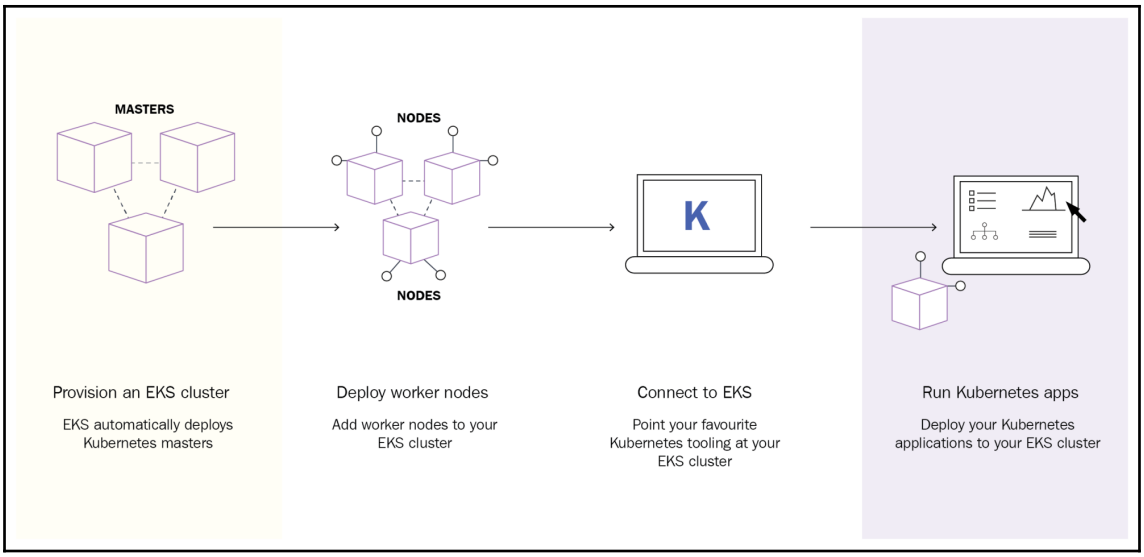
- **Elastic Map Reduce (EMR)**: If you are a user who wants to use the Hadoop framework for your application, EMR is the service for you. EMR is a managed Hadoop framework. AWS takes care of setting up and scaling the Hadoop framework. Users can also run other frameworks, such as Apache Spark, Presto, Flink, and HBase on Amazon EMR.
- **Kinesis:** This is a service for the collection and analysis of streaming data in real time. Kinesis consists of the following four modules:
    - **Kinesis Video Streaming**: This captures and analyses video-streaming data.
    - **Kinesis Data Streaming:** This captures and analyses data streams.
    - **Kinesis Firehose**: This helps in capturing streaming data and storing the data in AWS data stores, such as S3.
    - **Kinesis Data Analytics**: This allows users to analyze data streams with standard SQL queries.
- **Data Pipeline**: This is a data workflow orchestration service. This allows users to process data and move data reliably across AWS services, as well as on-premises data sources, at specified intervals.
- **AWS Glue**: This is a fully managed **extract**, **transform**, and **load** (**ETL**) service. AWS Glue generates code that performs data transformation and executes the data-loading process. Once the code is ready, it can be run on an Apache Spark environment, which is fully managed by AWS Glue.
- **Amazon QuickSight**: This is a business analytics service. This allows users to build a visualization of data. Users can perform ad hoc analysis of data. This helps with obtaining quick insights into data. QuickSight helps you to perform analytics without needing to buy any hardware or expensive BI software.
- **Athena**: If users have stored data in Amazon S3, then this data can be analyzed using standard SQL. Athena is the service that makes this interactive querying of data stored in S3 happen. Athena is a serverless service, so there is no infrastructure that needs to be set up to use it.

- **CloudSearch**: This is a managed service that provides the search functionality for your website or application. Users don't need to set up any search servers of their own; AWS CloudSearch sets up the servers and provides the search functionality as a service.
- **Amazon Elasticsearch**: This is a managed service that implements Elasticsearch. There are built-in integrations with Kibana, Logstash, and AWS services, such as VPC, KMS, CloudWatch, Kinesis Firehose, Lambda, and Cognito.

Container service

Docker has started playing a very important role in microservices architecture. Docker orchestration is inevitable for any complex microservices application. Let's have a look at what AWS provides for Docker containers and Docker orchestration by going through the following list:

- **Amazon Fargate**: This service allows users to run their containers without needing to manage any infrastructure. That is, users don't need to provision for the hardware on which the container runs; AWS Fargate does that for the users. Users just need to specify their CPU and memory and launch their application, which should be packaged in a container. AWS Fargate then takes care of setting up the requested hardware and launching the application. AWS Fargate is integrated with AWS ECS and AWS EKS.
- **Amazon ECS**: **ECS** stands for **Elastic Container Service**. This is a container orchestration service from AWS that supports Docker containers. Users can use this service to orchestrate their containers, instead of loading and managing their own container orchestration software.
- **Amazon EKS**: This stands for **Elastic Container Service for Kubernetes**. In the realm of Docker orchestration software, Kubernetes is becoming very popular. The EKS service of AWS is a managed Kubernetes service. With this service, users need not manage their own Kubernetes clusters; AWS takes care of provisioning, patching, and upgrading Kubernetes servers. Any application currently running on a Kubernetes cluster is fully compatible with AWS EKS. The role of Amazon EKS is shown in the following diagram:



Migration services

One of the questions that every enterprise that wants to move to the cloud asks is this: *How difficult is migration*? migration of an application involves migrating servers, migrating data, migrating databases, and migrating the application itself. The following list shows the tools that are provided by AWS for migration:

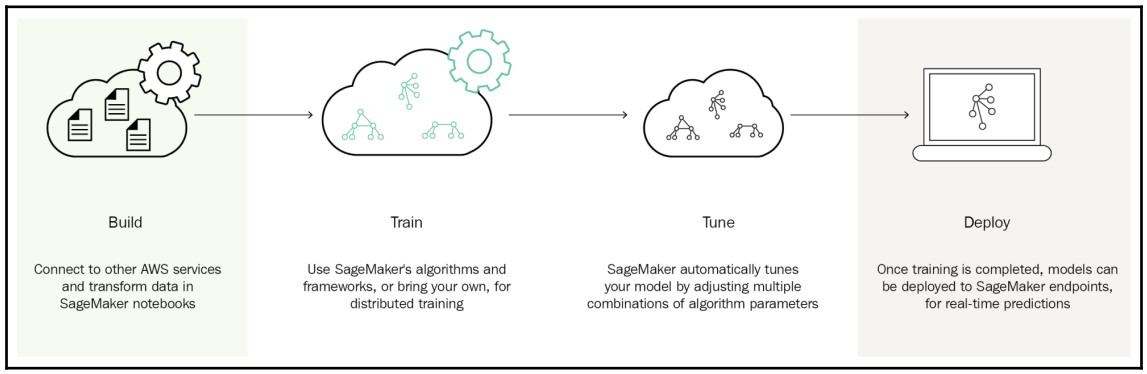
- **Server Migration Service (SMS)**: If your enterprise is using VMware or HyperV, you can use the SMS to automate the migration of VSphere Microsoft Hyper-V/SCVMM virtual machines to AWS. SMS performs incremental replication of the virtual machine, converting them into **Amazon Machine Images** (**AMI**). The AMIs can then be used to start instances on AWS.
- **Database Migration Service (DMS)**: Migrating a database is a critical part of migrating an application. The DMS service from AWS helps with migrating databases between on-premises and cloud storage. Relational databases, NoSQL databases, data warehouses, and other data stores can be easily migrated using this service.
- **Amazon Snowball**: If you need to transport a large amount of data to AWS, then the Amazon Snowball service will come in handy. The Snowball service uses a snowball appliance (50 TB or 80 TB in size), to which data can be transferred. The appliance is then shipped to Amazon, which uploads the data into AWS.

- **Application Discovery Service**: This is a profiling service. It profiles the application running in your data center and also collects data related to to the application, such as configuration and usage data, as well as system performance. Application discovery can happen in two ways: agentless or agent based. Once a discovery is performed, discovered servers can be grouped based on the applications they are hosting, and their migrations can be tracked. This way, we can track the migrations of all applications from on-premises to AWS.

Machine learning

Machine learning is one of the current hot topics, with more and more companies and products embracing it. AWS provides a great platform for machine learning and has some amazing products to help users with various areas, such as speech recognition, video recognition, translation, transcription, and others. Let's look at some of AWS's machine-learning products in more detail:

- **Amazon SageMaker**: This is a platform for machine learning. This service enables data scientists to build their machine-learning models, train the models with sample data, and then deploy the model to production. The models are refined later, based on the inferences generated and the new data obtained in production. SageMaker has hosted Jupyter notebooks, integrated with S3, which allows developers to visualize their data that's stored in S3. SageMaker also provides the most common ML algorithms. It can run TensorFlow and Apache MXNet in Docker containers. When users want to train their models, SageMaker makes it possible by easily scaling up to petabyte scale. Once trained, the model can be deployed using SageMaker on an auto-scaling cluster. This process is shown in the following diagram:



- **Amazon Lex**: This is a service that enables developers to build chatbots using both voice and text interfaces. Lex utilizes deep-learning functionalities of automatic speech recognition to convert speech to text. Lex uses natural-language understanding to understand the intent of text. All these factors ensure that the chatbots are lifelike. With Lex, developers are now able to access the same deep-learning technologies that were used to build Amazon Alexa.
- **Amazon Polly**: This is part of the language services provided by AWS using machine learning. AWS Polly is a text-to-speech conversion service. Deep learning is used to produce speech that is very close to a human voice.
- **Amazon Translate**: This is another language service from Amazon. This service is a neural machine translation service that translates supported languages and English. Using deep-learning technologies, it provides high-quality translations. Currently, six languages are supported: Arabic, Simplified Chinese, French, German, Spanish, and Portuguese. More languages will be added in due course.
- **Amazon Transcribe**: This is a speech-to-text service. This service is very useful in cases where voice calls need to be transcribed or when subtitles need to be added to videos. Amazon Transcribe uses the audio files stored in S3, analyzes them, converts the speech to text, and returns a text file to the user, containing the transcribed speech.
- **Amazon Comprehend**: This is a natural-language processing service used to discover relationships and insights in text. For example, Amazon Comprehend can scan a lot of text—say, from multiple websites—and find out how a particular product has been rated. Using machine learning, Amazon Comprehend can identify key words, understand how positive or negative the text is, identify brand names or people's names, and much more. The results of Amazon Comprehend can be used in a variety of applications, be it analyzing customer support incidents, identifying which feature of your product your customers love the most by analyzing social media feeds, and so on.
- **Amazon Rekognition**: This is a video and image-analysis service. This service enables the recognition of people, text, activities, objects, and so on. Rekognition can be used in various tasks and services, such as comparing faces for facial recognition, user verification, people counting, and so on. It can also be used to detect inappropriate content. Rekognition uses image files and video files stored in S3 and analyzes them.

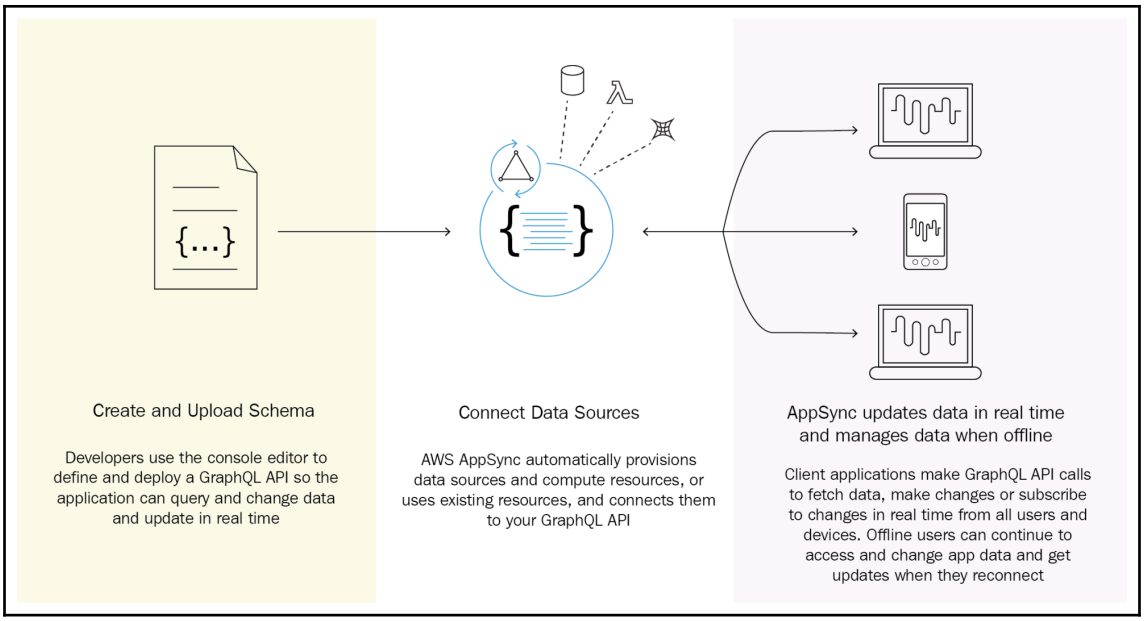
Augmented reality and virtual reality

Augmented reality (**AR**) and **virtual reality** (**VR**) are becoming a reality in today's world (forgive the bad joke). AWS has the Sumerian service for AR and VR, helps users to create VR and AR, as well as 3D applications. The main benefit of using Sumerian is that it does expect the user to be an expert in graphics programming. Sumerian allows users to create immersive environments and lifelike characters. Sumerian is platform-agnostic: once applications are built using Sumerian, they can be run on Android and iOS devices, Oculus Rift, HTC Vive, and HTC Vive Pro.

Mobile

Mobile development requires various services. AWS intends to ease the work of mobile developers by offering various services, including mobile-testing services, authentication services, and others. These services are described in the following list:

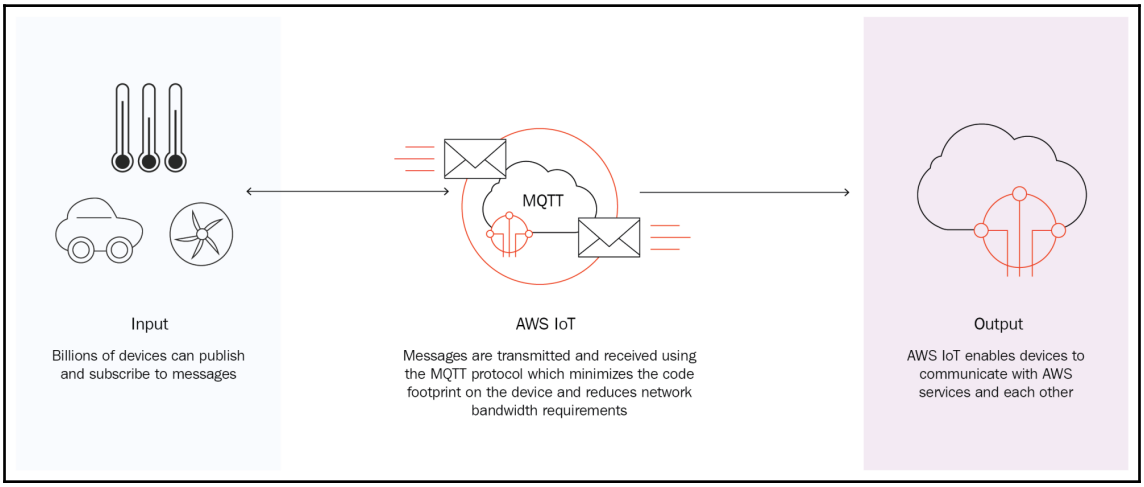
- **AppSync:** This is a service for web and mobile app developers. This service ensures the automatic updating of data for web and mobile applications in real time. If a user is offline, the data is updated as soon as the user comes online. AppSync uses an open standard query language called GraphQL. Using GraphQL, applications can request data from the cloud. AppSync is also very useful for developing a collaborative application that depends on the real-time updating of shared data. It can also be used in applications where data is sourced from various channels. AppSync's role and function are shown in the following diagram:



- **AWS Device Farm**: This is an app-testing service. This service allows apps that have been developed in Android, iOS, or on the web to be tested against many devices. The user can select their choice of real phones and devices. AWS Device Farm allows direct access to these private devices from the host machine. This is similar to attaching devices directly to the host machine. The device screen is displayed on the browser. The user can then interact with the device screen and test the new functionality or reproduce bugs.
- **Amazon Cognito**: Cognito is a service for mobile and web app developers. Cognito provides an easy way for applications to add users, handle user sign in, and ensure that there is access control. Cognito also supports user login using any social identity providers, such as Facebook, Google, Amazon, and enterprise identity providers, such as Microsoft AD. If they use Amazon Cognito, users can create their apps without worrying about authentication, which will be taken care of by Amazon Cognito.

Internet of Things

The **Internet of Things** (**IoT**) is another growth area. For those involved in developing IoT solutions, AWS has a lot to offer. The IoT platform of AWS and its other tools will come in handy for IoT developers. The following diagram and list show the roles and functions that AWS IoT and its associated services can provide:



- **IoT Core**: This service is a cloud service that allows devices to connect to the cloud as well as other devices. IoT Core supports billions of devices and trillions of messages. IoT Core can route these messages to AWS services or to other devices. It supports HTTP, WebSockets, and MQTT protocols. IoT Core also provides end-to-end encryption throughout all points of connection. Using IoT Core, data can be processed and transformed based on business logic.
- **IoT 1-Click**: As you saw earlier, AWS Lambda is a serverless service that enables us to run Lambda functions without provisioning hardware. IoT 1-Click enables simple devices to run Lambda functions. You can associate a Lambda function with a device or a group of devices. When triggered, the corresponding Lambda function is executed. Lambda functions can be used to perform actions such as notifying support, tracking assets, and so on.
- **IoT Device Management**: This service is used to perform multiple tasks such as device onboarding, remotely managing devices, searching for specified devices, and organizing devices based on specified criteria. Device attributes can be added to the AWS IoT registry in bulk, and the devices can be put into service. It also allows users to form device groups so that access policies can be applied at group level. This service also enables us to search and locate a particular device among the entire fleet of devices in near real time.

- **IoT Analytics**: As would be expected, the volume of data gathered from IoT devices is generally very large. Users will need to set up their own platform to run analytics on gathered data. The IoT Analytics platform allows users to run sophisticated analytics on huge amounts of data, without having to set up their own platform. IoT Analytics can also filter and transform the data before storing it. You can then run queries on this data, using the built-in SQL Query engine.
- **Greengrass**: This is a service that brings cloud functionality to IoT devices. Greengrass Core is a piece of software installed on supported IoT devices (devices that run Linux on ARM or x86 architecture) that allows for the local execution of Lambda code, enabling devices to respond to local events. So, an action can be taken at the device level even when the device has lost connection to the cloud, using Greengrass. Devices with Greengrass Core can connect to devices that run Amazon FreeRTOS or that have AWS IoT SDK installed. These can form a group and can keep communicating with each other locally, even if the connection to the cloud is lost.
- **Amazon FreeRTOS**: This is an operating system for microcontrollers that is based on the open source FreeRTOS operating system. Amazon has extended the OS by supplying software libraries that make it easy for low-power devices to connect to AWS cloud services such as IoT Core or to AWS Greengrass-enabled devices.

Game development

Amazon provides the GameLift service, which is intended to help game developers. This service has the following features:

- **Amazon GameLift:** This is a managed service that deploys game servers on the cloud for game developers, freeing them from the headache of scaling and managing servers. AWS GameLift not only deploys servers, but also scales them based on their load and manages them servers. Developers can concentrate on building high-quality games without bothering about the infrastructure. Using its worldwide distributed infrastructure, GameLift ensures a low-latency gaming experience for players from around the world.

Customer engagement

All apps need to engage with the client; it is the most effective way to grow a business. Amazon's services—Amazon Connect, Amazon PinPoint, and **Simple Email Service** (**SES**)—all help app developers to develop apps that have good customer engagement. The following list looks at these services in more details:

- **Amazon Connect**: Amazon Connect is a cloud-based contact center. Using a GUI, users can quickly design contact flows, manage agents, and track performance metrics. Amazon Connect is based on the same technology that Amazon uses internally for its customer service. For users, there is no upfront payment or infrastructure that needs to be set up. They just pay for the time that they use the contact center (by the minute).
- **Amazon Pinpoint**: This service enables users to connect to their customers using multiple channels, such as SMS, email, and mobile push. Using the campaign management feature of this service, users can send SMS, email, or mobile push messages to customers. The service also tracks the way in which customers react to messages, such as how they opened the message and how many clicked on links, as well as how many messages were delivered.
- **Amazon Simple Email Service (SES)**: As the name indicates, SES is a service for sending and receiving emails.

Business productivity

AWS helps to increase the productivity of businesses by providing some effective services for them, such as Amazon Chime, Alexa for Business, WorkDocs, and WorkMails. The following list looks at these services in more detail:

- **Amazon Chime**: This is a communication service. You can use Amazon Chime for video conferencing, online meetings, voice calls, sharing data, and chat. Meetings can call you and you can join them with a single click.
- **Alexa for Business**: Amazon Alexa can be used as an intelligent assistant at work. It can help employees manage their schedules, keep track of their to-do lists, and set reminders. Alexa can start meetings and control conference room settings, all using voice commands.
- **Amazon WorkDocs**: This is a fully managed file collaboration and management service. Files are stored in a central location. Users can collaborate to comment on a file, share files, and access a files from any device.

- **Amazon WorkMail**: This is a managed business email and calendar service that allows users to check their mail using their favorite mail clients, including Microsoft Outlook, native iOS, and Android mail application, or any mail application that supports IMAP. This service removes the need for enterprises to provision and manage on-premises email servers.

Desktop and app streaming

Virtual desktops are being used effectively by various companies these days. AWS also provides a desktop as a service solution. It also provides an app-streaming solution. The following list looks at these two solutions in more detail:

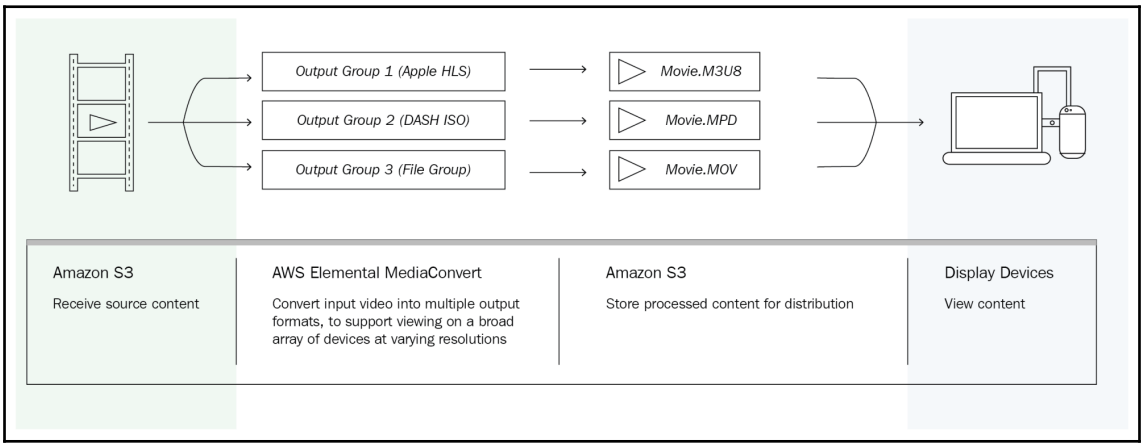
- **Amazon WorkSpaces**: This is a desktop as a service solution from AWS. Virtual Microsoft-based desktops can be provisioned for users in AWS cloud. Users can then access their applications and documents from anywhere and at any time using any of the supported devices.
- **AppStream 2.0**: This is a service that enables desktop applications to be streamed to a web browser that supports HTML 5. Let's say an organization has a desktop application that needs to be supported. Let's also say that the organization now wants to ensure that this application is available on a browser. In this case, code has to be rewritten for the browser. If AppStreaming is used instead, the desktop application will be streamed to an HTML-compatible web browser, and thus, users can access the application from a browser.

Media services

More and more videos are being generated and consumed regularly across the world. The consumption of media happens on various devices. There is a need to convert media so that it can be consumed by different devices. AWS has the following services for video transcoding, live streaming, video storage, and more:

- **Elastic Transcoder**: Transcoding is the act of converting media from one format to another so that the media can be played on a variety of devices, from `.wav` to `.mp3`, from 1080p to 720p, and so on. The Elastic Transcoder service is a transcoding service on the cloud that is provided by Amazon.

- **AWS Elemental MediaConvert**: This is a file-based video-transcoding service that has broadcast features built in. The input video content is stored in Amazon S3. AWS Elemental MediaConvert uses this as a source and converts it into various video formats as requested by the user. The converted videos are stored in S3. This process is shown in the following diagram:



- **Amazon Elemental MediaLive**: This is a broadcast-grade live video-processing service. Live video streams are encoded in real time. The input, which is a large-sized live video, is compressed to a smaller version for distributing to viewers. This service removes the need for users to set up their own broadcast-grade video-processing infrastructure.
- **Elemental MediaPackage**: This service takes input from Elemental MediaLive. It packages the incoming video stream into different distribution formats based on endpoint configurations. The formats help with implementing popular features, such as rewind, pause, and so on. This service also protects the content, using DRM.
- **Elemental MediaTailor**: This service allows video providers to insert individually targeted ads into their videos without compromising the video quality or the broadcast quality of the service.
- **Elemental MediaStore**: This is a storage service provided by AWS that is optimized for video. This service can be the origin of all video-based workflows.

Summary

AWS started with very few services and, in the space of 12 years, has expanded its services many times over. Currently, AWS is the leading IaaS service provider. AWS offers various services in the areas of computing, storage networking, databases, monitoring, security and identity management. Furthermore, AWS offers specialized services in the areas of big data, machine learning, game development, mobile development, IoT, media services, business productivity, and more. It is clear from the way that AWS has been innovating that we will be seeing more services from AWS in the near future.

In the next chapter, we will learn about OpenStack, which allows people to build a cloud of their own.

4
Microsoft Azure Stack and Private Cloud Services

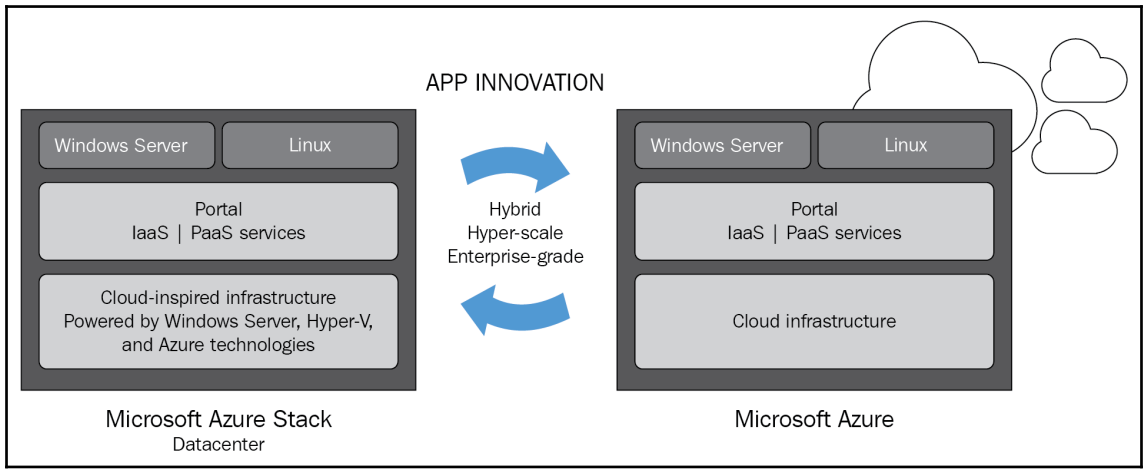
In this chapter, you will learn about the Microsoft Hybrid Cloud solution, which provides Azure in your data center. As you probably already know, you have the option not just to consume public Azure, but you can also order hardware (currently offered by seven different hardware vendors) with Azure services on top and run it in the data center of your choice. This solution is called Azure Stack.

The topics that we will look at in this chapter are as follows:

- Microsoft Cloud and Azure Stack
- Resource providers
- Azure Stack's core management services
- Working with Azure Stack
- Sizing Azure Stack
- Azure Stack services
- Azure Stack design scenarios

Microsoft Cloud and Azure Stack

When Microsoft started working on Azure, the decision was to go with a private stack. The reason for this was to prepare an infrastructure that could run in Azure in the future. As this solution is an extension of Azure, there is no system center, as this was the management solution of Azure Pack in the past. The following diagram shows a technical overview of Azure Stack compared to Microsoft's public Azure:



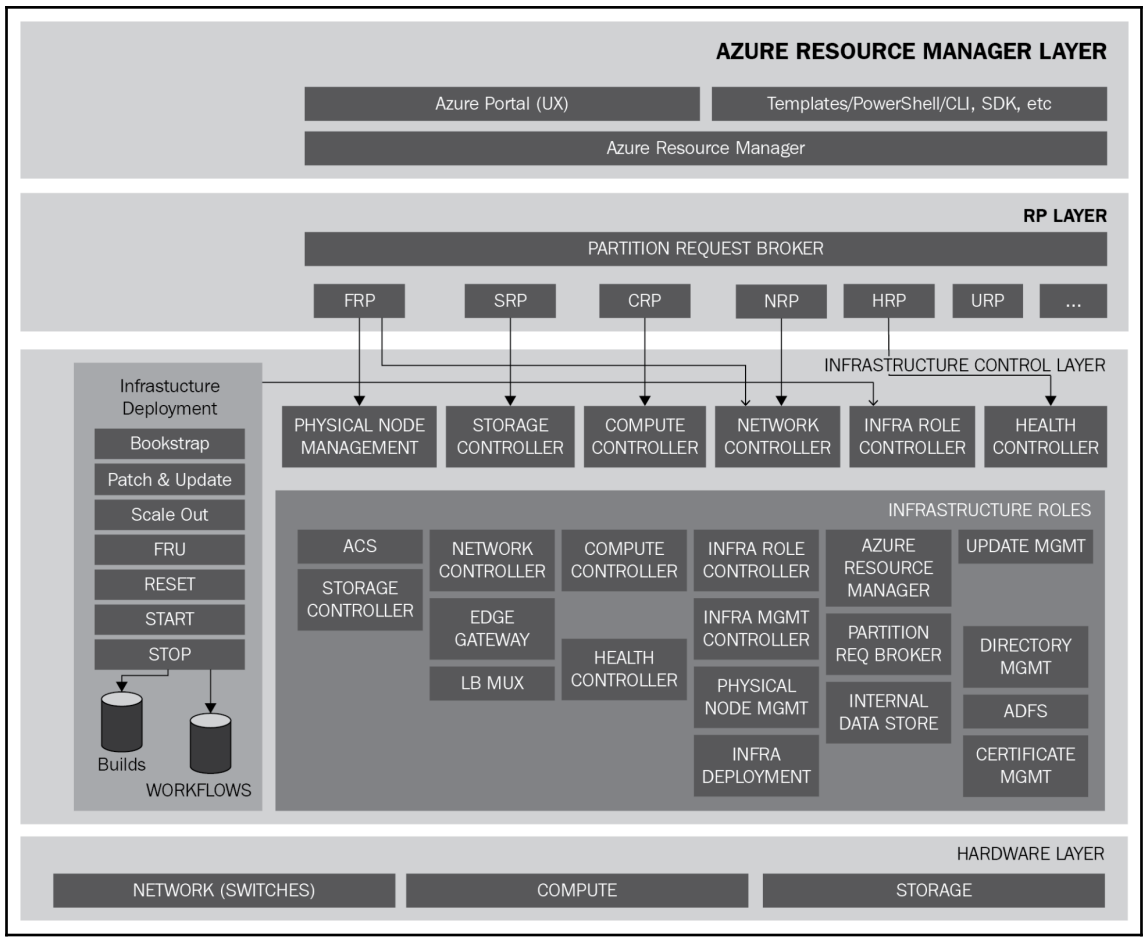
The only difference between Azure Stack and Azure is the cloud-inspired infrastructure. In Azure, there are tens of thousands of servers that are the minimum set; with Azure Stack, the number is quite smaller. The technology involved in Azure Stack is based on Windows Server 2016 and uses Windows Hyper-V as a virtualization solution. The general technology is based on Azure.

For stability and functionality reasons, Microsoft provides Azure Stack as an integrated system. This means that it is a set of servers with Azure Stack installed on them, and it can only be ordered through a hardware vendor. At **General Availability** (**GA**), the hardware OEMs are HPE, Dell EMC, and Lenovo. For the first quarter of 2018, Huawei, Cisco, and Intel/Wortmann announced an Azure Stack integrated system too. In addition to this, there will be a single-host development toolkit available (called the **Azure Stack Development Kit** (**ASDK**)). You can run it as a proof of concept on one generic server as soon as it fulfills the hardware requirements.

Because of this integrated solution, Azure Stack itself consists of only software-defined parts, including **Software-Defined Networking** (**SDN**) and **Software-Defined Storage (SDS)**. To name the specific underlying technologies, the SDN solution is based on VxLAN technology and the storage solution is based on **Storage Spaces Direct** (**S2D**).

Resource providers

The technical basis of Azure Stack is a piece of technology called the resource provider. Each resource provider adds features or functionalities to Azure Stack. Resource providers are an easy way to maintain and extend a solution, as shown in the following diagram:



Azure Stack consists of different resource providers. The three main ones are as follows:

- **Storage Resource Provider** (**SRP**)
- **Compute Resource Provider** (**CRP**)
- Network Controller (NRP)

We need to differentiate between them and the following additional resource providers:

- **Fabric Resource Provider** (**FRP**)
- **Health Resource Provider** (**HRP**)
- **Update Resource Provider** (**URP**)

Azure Stack's core management services

By default, Azure Stack offers the following core management services that you may already know from Azure:

- The authorization management service
- Subscriptions, plans, and offers
- Gallery
- Events
- Monitoring
- Usage

Authorization management

Azure Stack authorization leverages the Azure authorization management service. The following authorization options are possible:

- **Disconnected scenario (using ADFS and local accounts)**: For example, on ships crossing the ocean or on construction sites with no internet access
- **Connected scenario (leveraging Azure Active Directory (AAD))**: For example, a service provider offering Azure Stack as a service or in every other scenario where connection to AAD is acceptable

The connected scenario fully supports **Azure Active Directory** (**Azure AD**), which is a multitenant, cloud-based identity management service, and therefore provides full support of multitenancy. But it requires proper internet connectivity. This makes life quite easy.

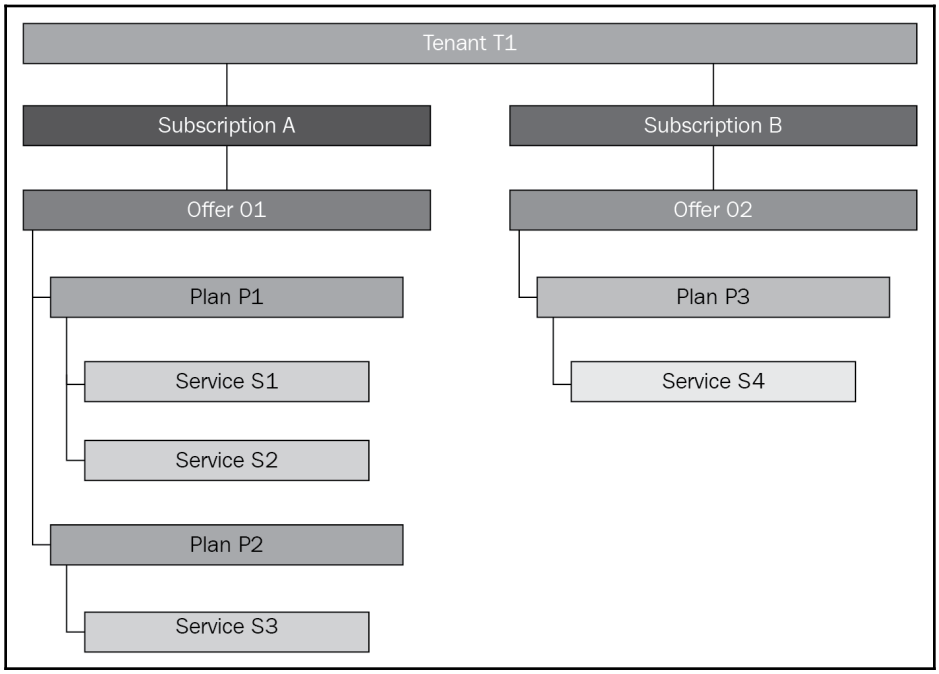
If the customer does not want an internet-connected scenario, the disconnected solution may fit, but it does not support multitenancy. This is quite often fitting for customers who need to put Azure Stack in a dedicated networking environment without any internet connectivity.



The connected scenario will allow you to have the full features of Azure Stack. Only if this is not possible or undesirable should the disconnected scenario be an acceptable option.

Subscriptions, plans, and offers

Azure Stack works with the same concepts as Azure regarding tenant management, namely subscriptions, plans, and offers. These are shown in the following diagram:



Plan

A plan is a description of a product offered by Azure Stack consisting of different services—for example, a website plan or a virtual machine plan. It is good practice to include the different levels of quality of a service in the plan—for example, bronze, silver, gold, or platinum. A plan is—more or less—the description of the products you will offer in Azure Stack.

Offer

An offer defines the various products that you are offering to your customers based on Azure or Azure Stack.

Subscription

The subscription is the brackets for everything. A user is given access control of Azure Stack with their username and password. This account is linked to an offer that is related to the plan. Setting up subscriptions can be done using the portal or by an operator that has the required permissions.

Events

The service called the event service provides information about a deployment taking place. It provides a log file for a specific resource. As in Azure, all services can be accessed via an API using PowerShell or any other coding languages.

Working with Azure Stack

Azure Stack can be managed using the same options as Azure, which are as follows:

- Azure Stack portal
- PowerShell cmdlets
    - Azure CLI
- API

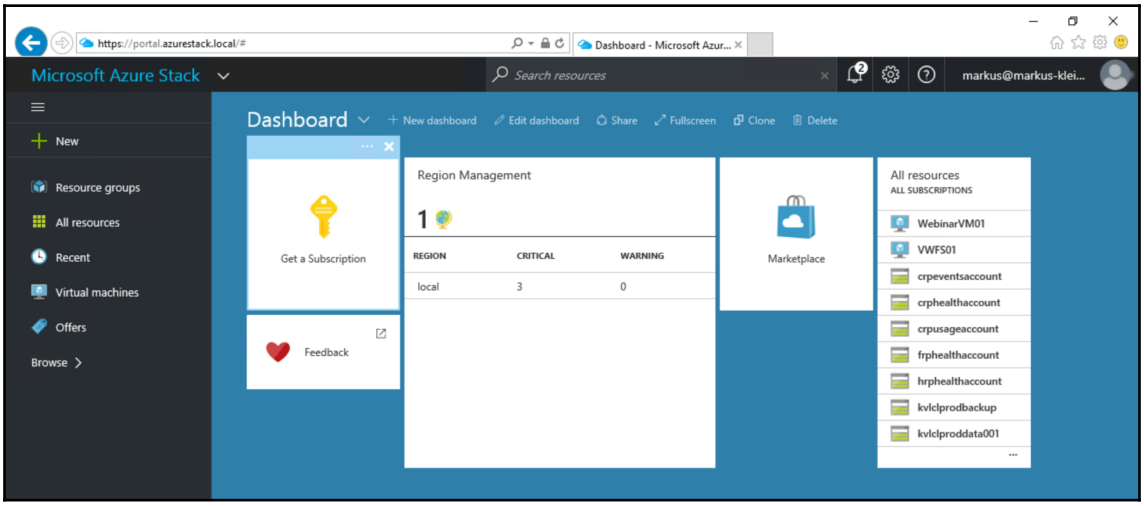
Depending on what you need, you should use one or more of these options to work with Azure Stack.

Azure Stack portal

As Azure and Azure Stack share the same portal UI, it is quite easy to get started with Azure Stack if you already know Azure. The portal provides access to the available services, the marketplace, and even governance features, such as RBAC.

As the portal works based on the role you log on with, you will have more or fewer features available, depending on the account you log on with.

The following screenshot shows what the Azure Stack portal looks like:



The Azure Stack portal looks more-or-less the same as the Azure Public one. This means that if you can work with one, you can use the other too.

PowerShell commandlets

As well as the portal, we have the option to work with Azure Stack using PowerShell (such as public Azure Hast). Microsoft provides a wide feature set of PowerShell `cmdlets` that have nearly the same experience and syntax that public Azure has. As scripting provides a basic documentation and a reusable set of services, this is the preferred way to work with Azure Stack.



To install Azure Stack PowerShell, you can go to the following link for a set of instructions that will walk you through the process. This will be useful, as the API versions of Azure Stack and Azure will be different, and therefore you need to have the appropriate versions in mind when you start the process:
`https://docs.microsoft.com/en-us/azure/azure-stack/azure-stack-powershell-install`.

Azure Stack CLI

The third option to work with Azure Stack is the Azure Stack command-line interface, which is a toolset that can be installed on any client. It is available for Windows, Mac, and Linux.



The download locations for Azure Stack CLI are as follows:

Windows: `http://aka.ms/azstack-windows-cli`
Mac: `http://aka.ms/azstack-mac-cli`
Linux: `http://aka.ms/azstack-linux-cli`

The CLI gives you another option to work with Azure Stack in a different, non-PowerShell-based way.

Azure Stack API

The fourth option to connect to and work with Azure Stack is by using the API, which is the same API that Azure uses.



Instructions to install the Azure **Software Development Kit** (**SDK**) can be found at `https://docs.microsoft.com/en-us/azure/azure-stack/asdk/asdk-download`.

SDKs are available for the following:

- .NET
- Java
- Node.js
- PHP
- Python
- Ruby
- Mobile
- Media

Sizing Azure Stack

The smallest footprint of Azure Stack in production is a four-node setup, which includes the **Top-of-the-Rack** (**ToR**) switches and possibly the aggregate switch. If you will be scaling resources later, you will have to order another rack and add it to your Azure stack. If the first rack is missing an aggregate switch, then this needs to be added too. One thing you need to have in mind is that the portal and the Azure resource manager will always be in your first rack for version 1. This means that you will need to take special care of your first rack because if you lose it, your complete Azure Stack management is gone. For this, I would recommend that you have a working and tested backup-and-restore process in place to make sure that if something happens with the first rack, you will be able to restore it as soon as possible.

A good question to bear in mind is what your hardware OEM will need to set up your Azure Stack deployment as expected. There is a deployment collection Excel file that you will get from the hardware vendor you have chosen. All information in this document needs to be collected and then used as the basis for the deployment.

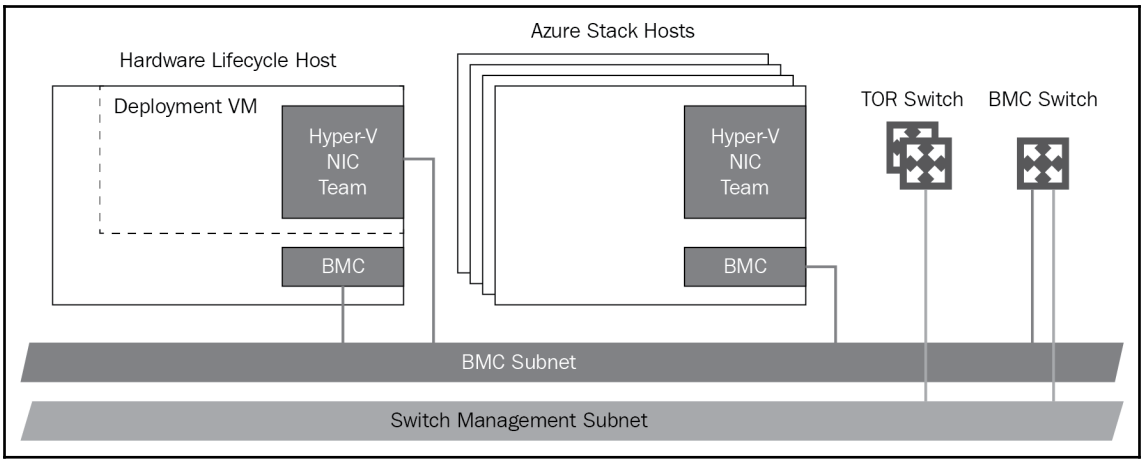
So, we need to prepare all these details in our planning of the design. Let's start first with the sizing of the solution. The following questions will help you get an outline of the design:

- How many end customers are you planning to onboard to the solution today and within the next five years? What kind of customers are you offering your services to, keeping in mind that you have a buffer of 30–50% for unexpected growth?
- How many resources do your offerings require per service implementation? Think about a gold, silver, and bronze offering for performance, and keep in mind that you should have a buffer too, if unexpected an load will run in these 30–50% services.
- Are you willing to offer highly available solutions to the end customers (fault domains)? If so, how many, and in which different fire locations would you like to place the stacks? Which service will be highly available? How would this affect the connectivity? Do we need **Quality of Service** (**QoS**)? Think about offering different solutions as non-HA or HA.
- How many regions are you planning? (For GA, only one region is planned; later on, additional ones will be possible, too.)
- Are you planning to offer your services on a national or an international market? This will give you an idea of the compliance regulations you will need to map.

- What's your monitoring concept for the solution? Do we need to add Azure Stack to an existing monitoring service (such as Nagios, the System Center Operations Manager, or the Operations Management Suite)? What would this mean for the existing load? How should we connect this if the installation of an agent is not possible because the system is locked down?
- A backup solution needs to be added, so what does this mean for your resources? Do you need to add a certain percentage of resources for the placement of the backup?

A lot of Azure Stack customers would ask whether they would need to run the complete project with the hardware OEM. This is really not the case, as the consulting company you are currently working with might have a deeper knowledge of Azure Stack than the hardware supplier. The consulting company can help you prepare your Azure Stack project and decide which hardware OEM would be the one you want to go with. The company could even help you fill in the Excel sheet. Later, when the hardware is up and running, your consulting company could help you prepare the offerings with Azure Stack and even create the ARM templates with or for you.

The networking part of Azure Stack looks as follows:



The best thing would be to have a workshop to define a basic concept of Azure Stack in your data center. During this workshop, the answers to the previous questions should be the basis of the discussion. Once we have answered these questions (including the ones relating to sizing), we can go on to describe the big picture of your Azure Stack deployment.

After deciding how many hosts the footprint would go live at the start, the next question is always about the regions. For version 1, we could set up just one region; if we needed more, there are two basic options we could go with:

- Make sure your second data center is well connected and that you have all management in your primary data center
- Set up a different Azure Stack installation in your second data center and deploy all ARM templates and the portal again without any connection to the primary one

For future releases, we can expect to have connectivity and single points of deployment for more than one region.

Azure Stack services

Azure Stack provides **Infrastructure as a Service** (**IaaS**) and **Platform as a Service** (**PaaS**) solutions, but what is the difference between them In this section, we will discuss the concepts of IaaS, PaaS, and infrastructure as code.

Infrastructure as a Service

Azure Stack is not just another virtualization replacement. This means that it should not be considered just for IaaS solutions. In this case, a Hyper-V cluster with virtual machine manager could be a better solution.

If you are considering running Azure Stack, it needs to return the investment, and in the best cases, you can earn money with an Azure Stack. Looking at a "lift-and-shift" scenario, this means moving all VMs on the premises to the cloud—1:1. The result will most often be that it is more expensive compared to your on-premise solution. Therefore, you should bear in mind that each VM will be paid based on the pay-as-you-go model. This means that a VM should be shut down for the time it is not used to decrease the costs.

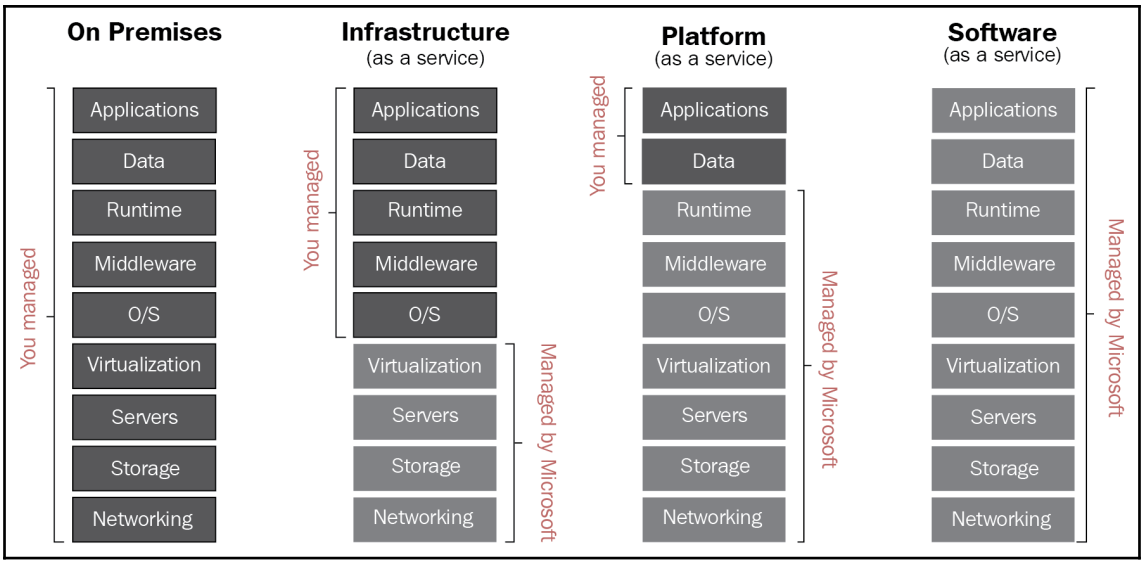
But it is this model the one you are really willing to use? Do you run VMs in your data center and power them on only if you need them? From a pay-as-you-go model perspective, this will reduce the costs and drive your cloud solution to be more economical, from a business perspective.

Platform as a Service

Costs will start to decrease when you start to use multitenant solutions on a PaaS basis.



PaaS means that when it comes to shared responsibility in cloud environments, you are not responsible for the VM and the OS, but, instead, you are running your services (for example, an SQL database) on an existing SQL server cluster. You are responsible for the database and the cloud provider is responsible for running the VM, the OS, and the basic application (in this case, the SQL server) itself.



The preceding example of a SQL server is a good way to explain the business aspects a little more. In general, the SQL licensing model is expensive. On top of this, you also need high-class resources to run a powerful database for an application. If you could just use an existing SQL server with your database, this would certainly cost less per database, and if you think about your IT infrastructure, how many SQL servers are running with only a single database on them? How many resources could be set free if you started using these databases on a single SQL server cluster?

If we go a step further and move this example to a private-cloud environment with hybrid connectivity, you will have to rethink the story a little bit. If you are running the Azure Stack environment, this means that you will have run the SQL server environments yourself. If you are using an Azure Stack environment in a service provider, you would not have to be responsible for running the environment. This needs to be considered when planning your hybrid cloud environment.

As Azure Stack owns the concept of resource providers, you will notice that each additional resource provider will follow the same concept to interact with the ARM technology. In general, you will see that each new PaaS solution from Azure Stack will bring their own virtual machines (with a minimum of one) with their own web service and some ARM integrations.

For GA, you will find the following three additional resource providers with PaaS solutions:

- Microsoft SQL Resource Provider
- MySQL Resource Provider
- App Service Resource Provider

Serverless computing on Azure Stack

Azure Stack fully supports serverless computing. This is where cloud computing could evolve in the future. As in public Azure, this feature is called Azure Functions.



Azure Functions provide a way to run code in a cloud environment and get results, but do not prepare any environment for it; just throw the code to the cloud, wait to get the results, and only pay for the time you use this service.

You could use these functions in the following use cases:

- Timer-based scenarios (for example, cron-jobs in databases for cleaning up)
- Event procession, based on WebHook (for example, exporting newly discovered data in a blob storage to an SQL database)
- SaaS event processing (for example, using Outlook email to export an attachment to OneDrive)
- Serverless web application architectures (for example, transferring webpage inputs to an automation script)

- Serverless mobile backends (for example, creating a thumbnail for a picture once it has been taken)
- Real-time stream processing (for example, transforming the data of thousands of devices and saving the important data to a database)
- And many more

Azure design scenarios

One common question is this: Do we really need a hybrid cloud? The answer depends on the following concerns:

- Data sovereignty
- Sensitive data
- Regulations
- Customizations
- Legacy support

A hybrid cloud is, in general, a business- or law-driven cloud solution. Often, it is not required by technology, but it is required for other reasons. This is not true in a disconnected scenario, where there is no internet connectivity.

Sensitive data

For companies working with personal data (for example, health care companies and doctors), there are countries around the world where this data must not be saved somewhere in the cloud or in countries other than the one the company in question is based in.

Regulations

In some countries, there are government requirements that define the physical location of data in specific locations. If this is the case, Azure Stack as a hybrid cloud solution could help.

ROI and TCO

A common question is this: Do I need Azure Stack in my data center? Regarding ROI and TCO calculations, existing solutions are great from a TCO perspective. The hardware has already been bought, which is great, given that new hardware requires bigger budgets. Looking at the costs of managing a solution, Azure Stack is easier to handle as the additional training for on-premises technology is not required anymore, as Azure Stack behaves like Azure—it is Azure—and if you can work with one, you can work with the other one too.

Starting an Azure Stack experience

If Azure Stack can fit the customer's needs, the next step would be to deploy Azure Stack, using the Azure Stack Development Toolkit. The hardware requirements are as follows:

Hardware

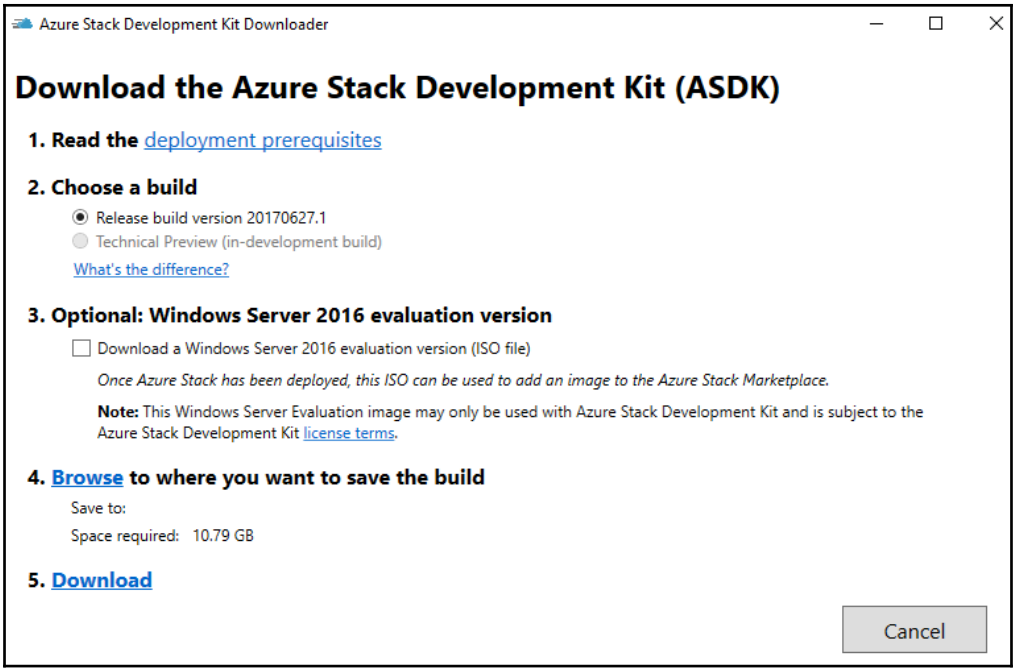
Component	Minimum	Recommended
Disk drives: Operating System	1 OS disk with minimum of 200 GB available for system partition (SSD or HDD)	1 OS disk with minimum of 200 GB available for system partition (SSD or HDD)
Disk drives: General development kit data*	4 disks. Each disk provides a minimum of 140 GB of capacity (SSD or HDD). All available disks will be used.	4 disks. Each disk provides a minimum of 250 GB of capacity (SSD or HDD). All available disks will be used.
Compute: CPU	Dual-Socket: 12 Physical Cores (total)	Dual-Socket: 16 Physical Cores (total)
Compute: Memory	96 GB RAM	128 GB RAM (This is the minimum to support PaaS resource providers.)
Compute: BIOS	Hyper-V Enabled (with SLAT support)	Hyper-V Enabled (with SLAT support)
Network: NIC	Windows Server 2012 R2 Certification required for NIC; no specialized features required	Windows Server 2012 R2 Certification required for NIC; no specialized features required
HW logo certification	Certified for Windows Server 2012 R2	Certified for Windows Server 2012 R2

*You will need more than this recommended capacity if you plan on adding many of the marketplace items from Azure.

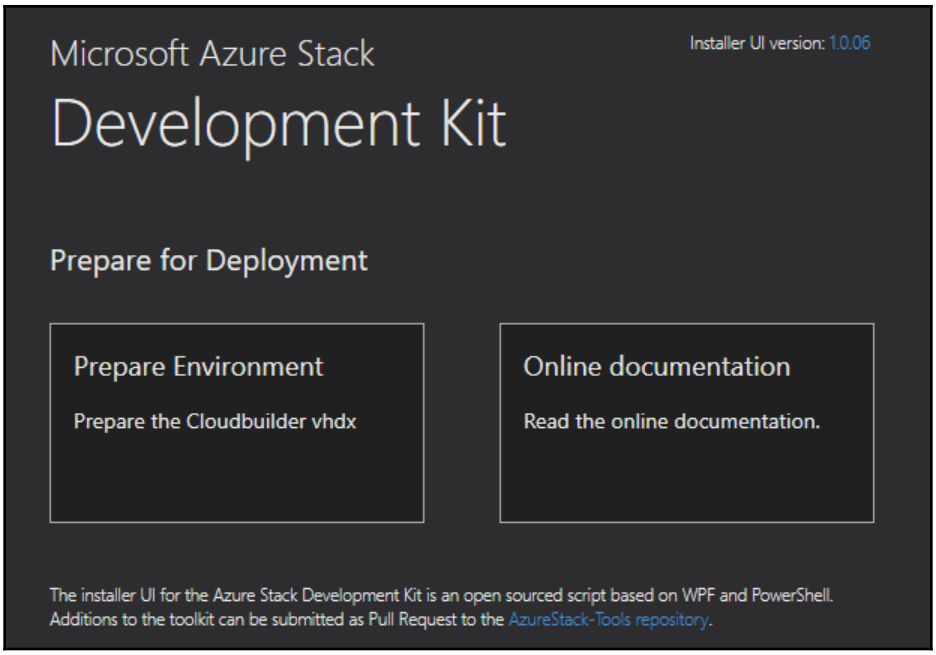
Data disk drive configuration: All data drives must be of the same type (all SAS, all SATA, or all NVMe) and capacity. If SAS disk drives are used, the disk drives must be attached via a single path (no MPIO, multi-path support is provided).

The next steps would then be as follows:

1. Download the most recent version of the ASDK at `https://azure.microsoft.com/en-us/overview/azure-stack/development-kit/?v=try`, as shown in the following screenshot:



2. Extract the download (`CloudBuilder.vhdx`).
3. Prepare the installation by running the ASDK Installer, as shown in the following screenshot:



This will guide you through the installation.

4. After the reboot into the `CloudBuilder.vhdx`, just restart the same tool and add the following information for a proper setup:
    - The path should be `CloudBuilder.vhdx`
    - Optional storage drivers
    - `Computername`
    - `Timezone`
    - DHCP or static IPv4 address

After this, the installation will start.



A successful installation of ASDT should take 4-5 hours. If something fails, bear in mind that the `-rerun` parameter is always the first thing to look for when troubleshooting.

Optionally, you can then install the resource providers, and Azure Stack will be ready to go for testing.

Testing Azure Stack in public Azure

If someone wants to start without dedicated hardware, it is possible to prepare Azure public with an Azure policy to behave in the same way as Azure Stack. This can be set up for a complete subscription or a dedicated resource group.



Further information on how to configure Azure to behave in the same way as Azure Stack can be found at `https://github.com/yagmurs/AzureStack-VM-PoC`.

Summary

As you have seen in this chapter, with Azure Stack, you can run the same Azure technology from Microsoft public Azure in your data center. This gives you a way to run services on-premises while using the same administration and operation concepts from public Azure. Although Azure Stack has fewer services than public Azure, it is a good entry point to bringing the cloud on-premises. As Azure Stack is an agile product, you will get updates and/or new features each month. Regarding connecting to public Azure, Azure Stack provides two different connection designs: disconnected or connected. As a result, there is nearly no design left where it does not fit.

5
OpenStack and Its Private Cloud Service

In earlier chapters, we learned about the public cloud services from Microsoft and Amazon. With these services, the service providers, Microsoft and Amazon, build the cloud. Users then use the services provided by the cloud.

But how is the cloud built? Can companies build their own cloud? How can a company build a public cloud? Well, to build a cloud, a lot of tools are required. OpenStack, a free and open source platform, provides the necessary tools to build and maintain a cloud. The cloud that is built can be a public or a private cloud.

So, what is the problem that OpenStack is trying to solve? OpenStack's mission statement gives us a clear idea of what it sets out to achieve:

> *The OpenStack mission is "to produce a ubiquitous Open Source Cloud Computing platform that is easy to use, simple to implement, interoperable between deployments, works well at all scales, and meets the needs of users and operators of both public and private clouds."*

The following topics will be covered in this chapter:

- The history of OpenStack
- OpenStack architecture
- OpenStack services

Let's look at the history of OpenStack first, and then proceed to check out the various services OpenStack offers to its users.

The history of OpenStack

The origins of OpenStack lie in **Rackspace**, a cloud company. Some years ago, Rackspace was using its own infrastructure code to manage its hosting infrastructure. In 2010, Rackspace wanted to rewrite the infrastructure code. At this point in time, Rackspace considered making the code widely available by open sourcing it. Anso Labs, which was contracting for NASA, had released the beta version of its product, **Nova**. Nova was a cloud computing fabric controller and was written in Python. The resources of both these companies merged, and thus OpenStack was formed. The early code was a combination of Rackspace's cloud file platform and NASA's Nebula platform. NASA made Nebula open source, and Rackspace did the same with its cloud software.

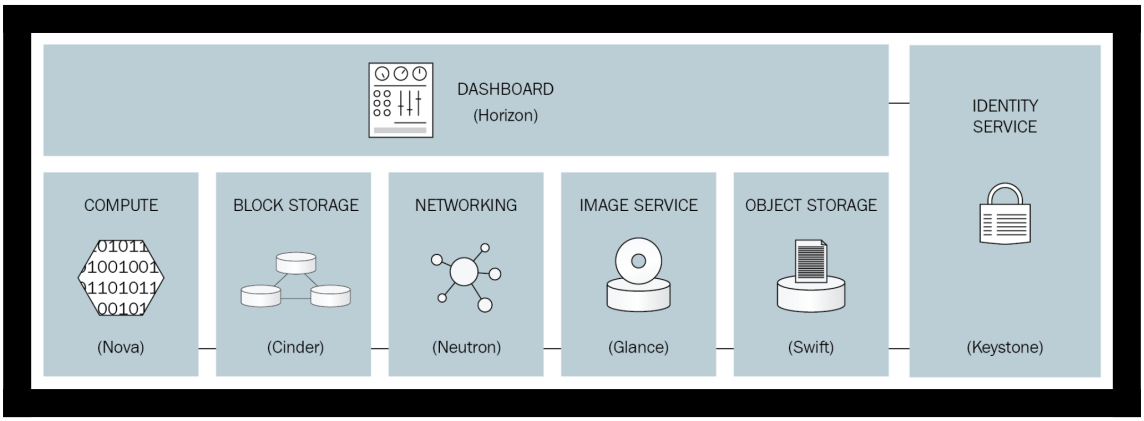
On July 13 and 14, 2010, the first OpenStack Design Summit was held in Austin, Texas. On July 21 of the same year, the project was officially announced at Portland, Oregon, at OSCON.

At the OpenStack conference in 2011, which was held in Boston, Rackspace declared that there would be an OpenStack Foundation, which would be in charge of project governance and would be responsible for the ownership of the OpenStack trademark. The OpenStack Foundation was jointly created by 27 companies. The formation of the OpenStack Foundation was formally announced on April 12, 2012.

OpenStack was initially started with two projects: Nova, from NASA, and Cloud File, from Rackspace. The community started growing, and new projects were added to OpenStack. Around the end of 2012, the concept of **incubation** was introduced. The idea was to grow projects, for inclusion in OpenStack.

OpenStack architecture

OpenStack has a modular architecture. You could say that OpenStack is a combination of various projects, each aimed at providing a certain functionality, as shown in the following diagram:



The following table gives the names of some of the major projects and the service they provide, from an OpenStack perspective:

Service	**Project name**	**Description**
Compute	Nova	This a critical service. This service is responsible for the life cycle of a virtual instance, starting from the provisioning of a **virtual machine** (**VM**), all the way to terminating that VM. This service acts as the brain of OpenStack and manages all VMs.
Block storage	Cinder	Cinder provides a block-storage service. The major goal of this service is to provide persistent storage to virtual machines. The service allows users to manage these block devices easily.
Networking	Neutron	This service is used to manage networks. This API-driven service allows users to define their own network and attachments to them. Assigning IP addresses to VMs, managing firewalls, and VLAN management are some of the functions that Neutron provides.
Object storage	Swift	Object storage allows for the storage and retrieval of objects using RESTful API calls. Swift has a scale-out architecture that allows people to store a large number of files. Fault tolerance is provided in Swift by way of replication.

Dashboard	Horizon	One of the key requirements of a cloud is that it should support self-service. Horizon provides a web-based self-service portal for OpenStack. This allows users to interact with various services, such as Nova, Neutron, Cinder, and so on, using the web interface. Users can use the web interface to manage instances, configure access controls, and so on.
Identity service	Keystone	The basic aim of this service is to provide authentication and authorization for other OpenStack services. A central directory of users mapped to various OpenStack services is provided by Keystone.
Image service	Glance	When virtual machines are being provisioned, they need virtual machine disk images. Glance is the service that stores and provides the virtual machine disk images.
Telemetry	Ceilometer	This service monitors cloud usage and provides usage data that can be used for billing. We can monitor and meter the OpenStack cloud, using this service for billing, benchmarking, scalability, and statistical purposes.

These are the most common services of OpenStack. There are other, higher-level services that we will talk about later.

Control plane

The web interface, the messaging service, the database, and the API service form a part of the control plane. OpenStack uses the messaging service to communicate with all its services. The protocol used for the message-queuing service is the **Advanced Message Queuing Protocol** (**AMQP**). This protocol supports plug-in components, so the backend can be provided by message-queuing services such as Rabbit MQ, Qpid, and ZeroMQ.

The databases used can be MySQL, MariaDB, or PostgreSQL.

OpenStack services

As pointed out previously, OpenStack provides various services, and each service is dealt with as a separate project. The most popular OpenStack projects are as follows:

- Nova, for computing
- Neutron, for networking

- Cinder, for block storage
- Swift, for object storage
- Glance, for image services
- Keystone, for identity services

There are various other projects that provide a variety of services. In this section, we will be looking at all of these services in detail.

OpenStack compute service – Nova

The aim of the compute service is to launch, monitor, and manage the cloud compute infrastructure—in this case, VM. Nova is the OpenStack project that provides a way to provision, launch, and manage VM. You can create a VM using Nova. In conjunction with another project called **Ironic**, Nova can also be used to launch a bare-metal server. Nova has a limited capability for running system containers, and it provides its services by running a set of daemons on top of existing Linux servers. Nova allows users to create , scale , reboot , and delete instances. They can do this via the UI or by using the CLI.

Nova works in conjunction with other services. These services are needed for Nova to have its basic functions:

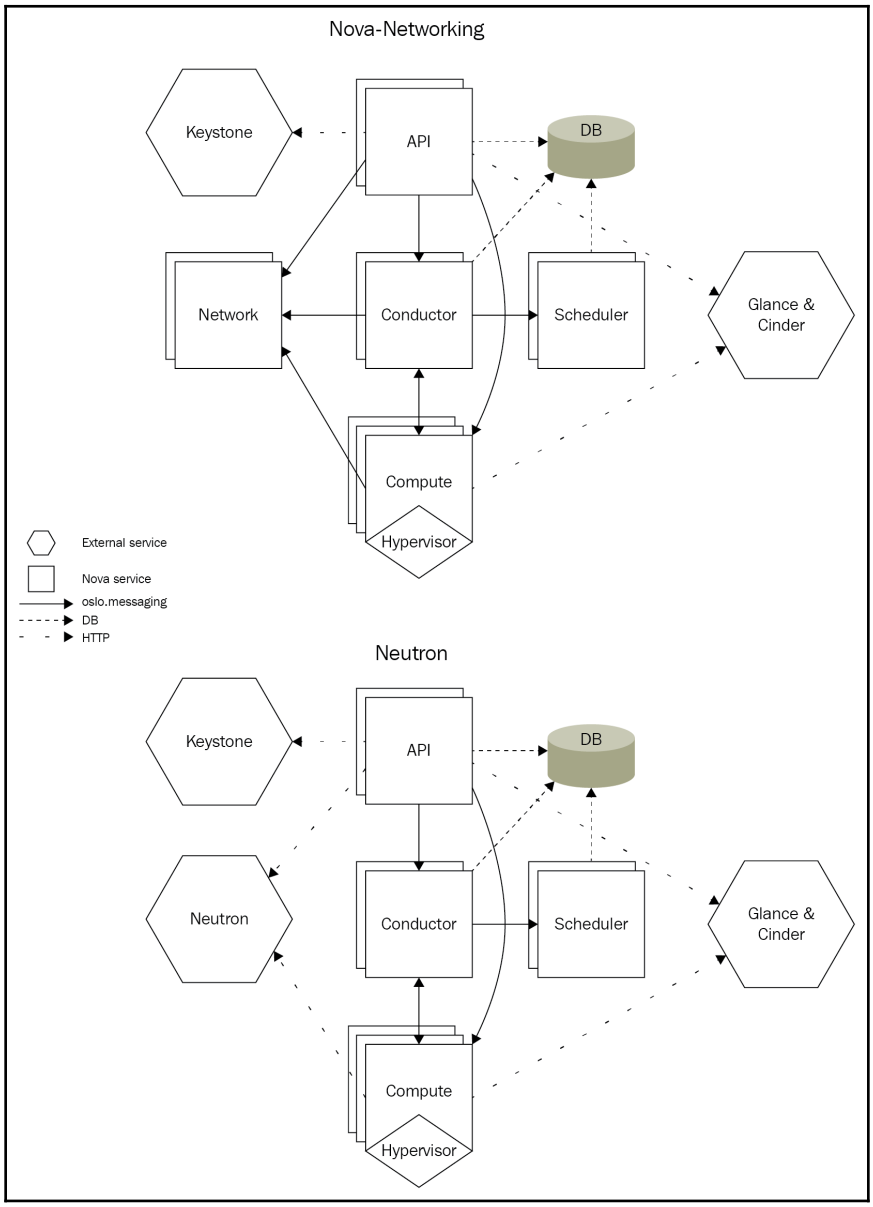
- **Keystone**: This is needed to provide the identity and authentication of users using Nova.
- **Glance**: This is needed to obtain the images that the virtual instances use to be launched.
- **Neutron**: This is needed to build the network to which the instances will connect to once they start.

Nova also interacts with Cinder to obtain block storage volumes and attaches these volumes to the virtual instances.

The end users can interact with Nova by either using the tools or by using the APIs directly. The tools that can be used include the following:

- Horizon, the WebUI provided by OpenStack.
- OpenStack CLI, called the OpenStack client, which can be used for Nova, as well as for other OpenStack services.
- Nova Client, which can be used for some very advanced features. While Nova Client is supported, OpenStack recommends using the OpenStack CLI whenever possible.

The Nova project exposes all the end-user features via REST APIs. So, if the end user wants to build complex logic or an automation, they can do this by using the APIs directly, or by using an API via an SDK. The following diagram gives us an idea of how Nova interacts with other OpenStack projects, such as Keystone, Neutron, and Glance:



OpenStack bare-metal service – Ironic

Ironic is an OpenStack service that provisions bare-metal-for users. Ironic integrates with other OpenStack services, such as Nova, Keystone, Swift, Neutron, and Glance.

To manage the bare-metal servers, Ironic uses both common remote management tools, such as PXE and IPMI, and vendor-specific remote management tools. Ironic provides a unified interface to a heterogeneous fleet of servers.

This service is made up of the following three components:

- **The Ironic API**: This is a RESTful API that processes application requests.
- **Ironic conductor**: This manages the bare-metal server. It can start and stop bare-metal servers using IPMI or other vendor-specific remote-management tools. It is also responsible for provisioning the bare-metal servers and deploying them.
- **Ironic Python Agent**: This provides the Ironic conductor with remote access for band hardware control and hardware introspection.

OpenStack identity service – Keystone

Keystone is the identity and access management component of OpenSource. It is the equivalent of the **Identity and Access Management** (**IAM**) service of AWS. This service is used to authenticate all the users of OpenStack. Keystone also provides service discovery and multitenant authorization.

Keystone supports various protocols, such as LDAP, OpenID Connect, SAML, OAuth, and SQL. Various types of authentication, such as username/password and token-based systems, are supported by Keystone.

Internally, Keystone is made up of the following services:

- **Identity service**: This service provides the data of the users and groups. It also provides the credential-validation service. This service may store the user data internally or interact with other services such as LDAP. Here, users represent the individuals who use the API. A *group* is a collection of users.
- **Resource service**: The main purpose of this service is to provide data about projects and domains. *Projects* (also referred to as *tenants* earlier) are collections of objects. Every resource in OpenStack must be owned by a project. For example, a user is a resource, and a user must be part of a project. Every project should be a part of a domain. So, a domain is a container that contains projects.

- **Assignment service**: This is responsible for assigning roles to projects or domains. A *role* is the authorization that an end user can obtain. The role determines which action a user can perform. An individual user or a group can be assigned a role.
- **Token service**: Once the user's credentials are validated, the token service manages the tokens.
- **Catalog service**: This is for discovering endpoints. This service provides an endpoint registry.
- **Policy service**: This provides a rules-based authorization engine and also a rule-management interface open.

OpenStack block storage service – Cinder

Cinder is the block storage service of OpenStack. Cinder is used to create and provide storage volumes. Once created, the volumes can be attached to an instance. Cinder takes care of all block storage management. Cinder is the equivalent of the **Elastic Block Store** (**EBS**) service of Amazon AWS. Using Cinder, a volume can only be attached to one instance at a time.

The block storage service contains the following components:

- **Backend storage devices**: Whenever a volume is created and provided to a user, the storage must come from a physical device. As such, there is a need for backend storage devices on which the services can be stored. By default, the LVM on a local volume group is used. These volumes are called **Cinder volumes**. In addition to this, Cinder also provides support for using other storage devices, such as external RAID arrays.
- **Volumes**: This is the equivalent of a hard disk. These volumes are persistent block storage devices. A volume is an allocated resource that can be attached to an instance. We have two types of volumes: volumes that store the OS image and volumes that are used to boot. These are known as **root volumes**. We can also have secondary volumes that are not root volumes.
- **Snapshots**: These are point-in-time copies of a volume. Snapshots are read-only. A snapshot is created from a volume. Once a snapshot is created, we can recreate the volume from the snapshot.
- **Backups**: These are archived copies of volumes. The backups are stored in the object storage service of OpenStack, called Swift.

OpenStack object storage service – Swift

Swift is OpenStack's object storage solution. Object storage is different than block storage, in the sense that object storage cannot be attached to servers and objects live in a flat namespace. Object storage is typically used to store a large number of objects. OpenStack Swift is the equivalent of AWS's **Simple Storage Service** (**S3**), which is also an object store.

OpenStack Swift is a distributed object-storage service. This means that there is no central point of control. This ensures that the service is scalable and redundant. According to OpenStack documentation, "*Swift is a highly available, distributed, eventually consistent object/blob store*." Using OpenStack Swift, users can store their unstructured data in a highly available object store at a very low cost.

Let's try to understand each component of the architecture by going through the following list:

- **Proxy server**: The main job of the proxy server is to handle all the incoming requests. Since Swift is an object store, each object stored in Swift has a URL associated with it. When a request is received by the proxy server, it determines which storage server the data resides in, based on the URL of the object. The proxy server coordinates the responses and handles all failures. Proxy Servers are generally kept behind a load balancer, and at least two proxy servers are used for high-availability purposes. The proxy servers are so designed that they have a **shared-nothing architecture**. This enables us to add more proxy servers when workloads increase.

- **Ring**: The main purpose of the ring is to map objects names to a physical location on disks. There are three types of ring: an account ring, a container ring, and an object ring. Whenever there is a need to perform an operation on an object, the ring needs to be contacted, to determine the physical location of the object.
- **Zones**: Zones are basically used to isolate failure boundaries, which is required when replicating objects. For example, if we have three zones, we can replicate the object in these three zones so that the failure of any one zone will not affect the availability of the object.
- **Accounts and containers**: Each account and container is a SQLite database. This database is replicated across the cluster. The account database stores information about the list of containers in that account. The container database stores data about the list of objects in that container.

- **Replication**: This process ensures that the multiple copies of an object are consistent. It constantly compares the multiple copies to ensure that each copy of the object is the latest one.

OpenStack networking – Neutron

Networking is a key component of any cloud. Neutron is the project that provides the networking service for OpenStack.

Neutron allows users to define network connectivity and addressing in the cloud. It allows users to create a software-defined network infrastructure. Users can create networks, subnets, switches, and routers. Neutron also allows the creation of firewalls and a VPN.

The following OpenStack services interact with Neutron:

- The Keystone service is used to authenticate any API request that is received.
- In the case of the compute service (Nova), Neutron is used to provide the plug for each of the network interface cards on a VM to a particular network.
- The dashboard service, Horizon, is used to create and manage all the network services.

OpenStack's compute service (Nova) is one of the major users of Neutron. Using Neutron, the user is able to configure the network and the subnets. The user can then instruct other services such as Nova to attach devices to this network. Nova uses this to provide network connectivity for its VMs.

Neutron is made up of the following three components:

- The Neutron server
- A database
- Plugin agents

The plugins for devices from various manufacturers are available. The following is a sample list of manufacturers whose equipment plugins are available to use with Neutron:

- Arista Networks
- Avaya
- Big Switch
- Brocade
- BNC
- Cisco

- Freescale
- IBM
- Juniper
- NEC
- Citirix Netscaler
- VMware

A complete list of supported plugins can be obtained at `https://wiki.openstack.org/wiki/Neutron_Plugins_and_Drivers`.

OpenStack image service – Glance

For an instance to start, there needs to be an operating system image available. This image is then used by the compute service to start a VM. **Glance** is an OpenStack registry that contains these images. Glance is a service that stores, registers, discovers, and retrieves VM images. Using RESTful APIs, users can query Glance to obtain metadata or to retrieve images. The images themselves can be stored in different locations: in a simple filesystem, or in an object store such as OpenStack Swift. Glance is a central service for OpenStack. Without the service, a compute service such as Nova will not know where to obtain the image from.

The images stored in Glance need to be generalized; host keys, IP, and MAC addresses, as well as local user accounts will be removed. This is to ensure that the same image can be used multiple times to launch a VM. The utility called `cloud-init` can be used (as well as other tools) to provide host-specific information.

Users can build and store their own images in Glance. One way to do this is to start a virtual machine, load the required software, and then create an image. Before creating the image, users need to ensure that all host-specific information is removed. Generally, the `cloud-init` is always included in the image.

Glance is made up of the following components:

- **Glance API**: This component accepts all API calls for image discovery, image retrieval, and image storage.
- **Database**: The database is used to store image metadata. The choice of database is left to you. The common databases used are MySQL or SQLite.
- **Image files repository**: Glance supports various kinds of repositories. The repository could be a normal filesystem. It could also be an object store, such as Swift or a VMware datastore, and so on.

- **Metadata definition service**: This is a service that provides common APIs for vendors, users, and admins. This service can be used to define custom metadata.

OpenStack dashboard – Horizon

Horizon is a service that implements a web UI-based dashboard for OpenStack services. Nova, Keystone, Swift, and other services use Horizon.

Horizon was built based on the following key design principles:

- **Core support**: Horizon provides support for all the core OpenStack projects. There are three central dashboards in Horizon: the users, system, and settings dashboards. These three dashboards ensure out-of-box support for the core OpenStack services.
- **Extensible**: An extensible service ensures that new features and components can be added to Horizon by anyone.
- **Manageable**: The core database of the service must be easily manageable. It should be simple and easy to navigate the database.
- **Consistent**: Since Horizon provides the UI for various services, we need to maintain the same UI experience in each service by providing a consistent interface across various services.
- **Stable**: Since projects develop with time, it is very important to ensure backward compatibility. Backward compatibility is what gives stability to any product. Horizon's design philosophy is to ensure that backward compatibility is maintained for its core components and reusable components.
- **Usable**: A key design principle is to ensure that an excellent UI is provided that people will love to use.

OpenStack telemetry service – Ceilometer

One of the most important properties of the cloud is that it is a measured service. This means that the cloud bills you based on your usage. To do this, it is imperative that usage data is collected and used for billing.

Monitoring services on the cloud is another important task that users undertake. Raising alarms when thresholds are breached and reacting to any state-change events are activities that are performed on a regular basis. Collecting data metrics of the services that are used is another regular activity.

OpenStack Ceilometer is a telemetry service that helps perform these activities. It is a data-collection service. The data collected by Ceilometer can be used for billing, raising alarms, resource tracking, and much more. It can be used with all the core OpenStack components. We can think of Ceilometer as being the equivalent of CloudWatch in AWS.

The Ceilometer service performs the following functions:

- It monitors the notifications sent by various services and collects metering data and events.
- It continuously polls metering data related to OpenStack services.
- Once data is collected, Ceilometer publishes it to different targets, such as message queues and data stores.

The Ceilometer service has the following three components:

- A compute agent that runs on each compute node and polls for resource utilization data.
- A central agent that runs on a central server and polls for the statistics of resources that are not tied to a compute node.
- A notification agent that runs on a central server and builds metering and event data. This is later published to designated targets.

OpenStack orchestration service – Heat

Building your own infrastructure in the cloud can be a complex process, and if the same infrastructure is needed for another of your projects, building it from the bottom up is a waste of time. To build consistent infrastructure for complex cloud projects, an orchestration service is used. **Heat** is the orchestration service for OpenStack. Heat allows users to launch complex applications based on a template. This template-based orchestration is an example of infrastructure as code. Heat can be seen as the equivalent of CloudFormation in AWS. In fact, Heat supports CloudFormation templates as well.

A Heat template is used to describe the infrastructure in a text file, which is in a human-readable format. Various resources can be described in the template, including servers, IPs, security groups, volumes, and so on. Along with the resources, the template also specifies the relationship among the resources—for example, which volume should be attached to which instance.

The template can be used to manage the complete life cycle of the infrastructure. When changes need to be made to the infrastructure, we can alter the template file, and Heat will ensure that the appropriate infrastructure changes are made. While Heat is primarily an infrastructure orchestration tool, it integrates well with software configuration management tools, such as Puppet, Chef, and Ansible.

OpenStack backup/restore and disaster recovery service – Freezer

For any enterprise, backing up data and having a clear **disaster recovery** (**DR**) plan is essential. When moving to the cloud, users have to ensure they have a good backup strategy, as well as a good DR plan. Freezer is an OpenStack service that provides distributed backup/restore and DR services. Freezer is designed to be multi-OS compatible (Linux, Windows, BSD, and OSX).

With Freezer, we can perform the following tasks:

- Block-based backup
- Point-in-time actions
- Incremental file-based backup
- Backup synchronization across nodes

Freezer has the following components:

- **The Freezer API service**: This service accepts the API calls made by the end user and responds to them.
- **The Freezer Scheduler service**: As the name indicates, this service is for scheduling jobs. Not only does it schedule jobs, it also fetches, updates, and deletes backup jobs.
- **The Freezer Agent service**: This service is responsible for executing a backup job. It calls the scheduler to ensure that backup/restore jobs are executed.
- **The Freezer DR service**: This service performs disaster recovery in the case of any compute node failure.

OpenStack container service – Zun

Containers are being used extensively now. Microservices are built using containers. OpenStack's container service is called **Zun**. This service allows containerized workloads to be provisioned and managed on OpenStack without the need to manage servers or clusters.

Zun requires some other OpenStack services to operate. The services required are as follows:

- Keystone
- Neutron
- kuryr-libnetwork

Zun also integrates with other OpenStack services, such as Cinder, Heat, and Glance.

OpenStack load-balancing solution – Octavia

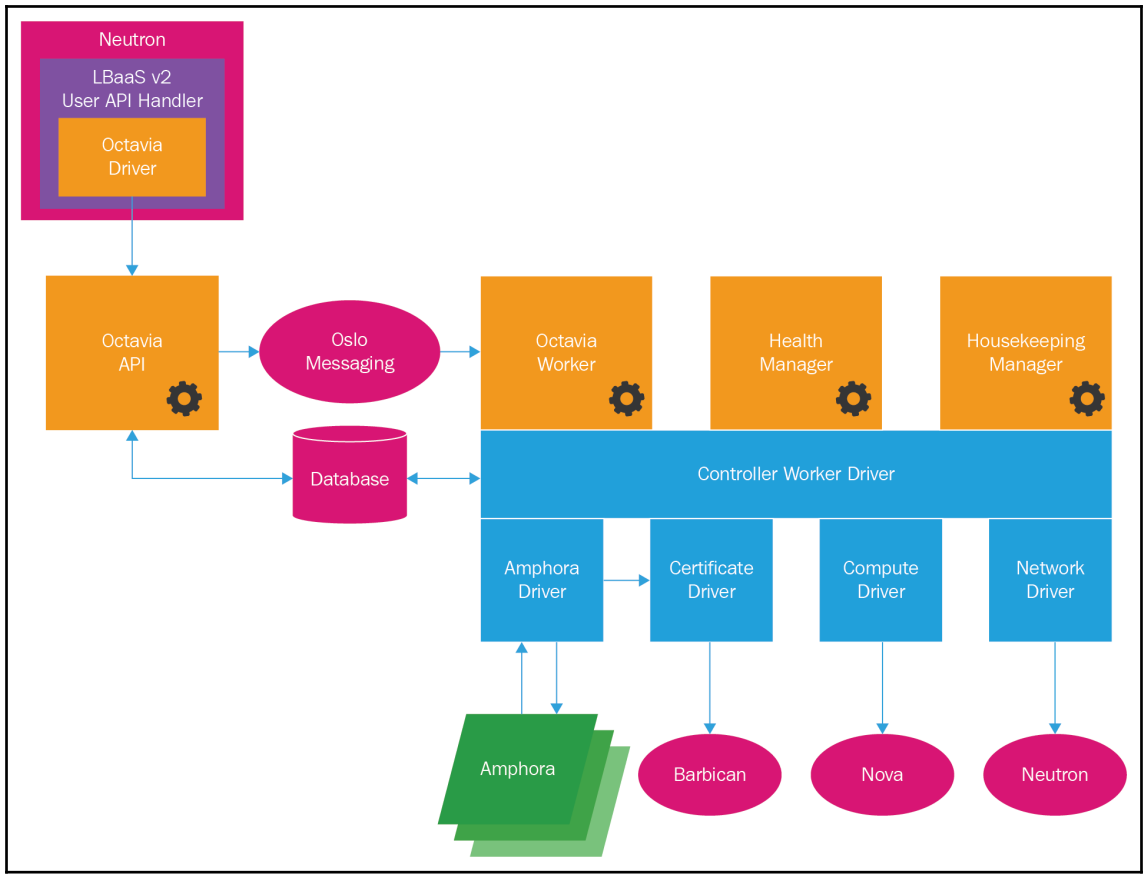
Load balancers are very important components in any cloud. Load balancing, coupled with autoscaling, is used to ensure that the load on an application is correctly dealt with. **Octavia** is the load-balancing service of OpenStack.

Octavia spins up VMs, containers, and bare-metal servers to horizontally scale and take care of balancing the load. This is an on-demand service. The collections of the fleet of virtual machines, containers, and bare-metal servers is called **amphorae**.

Octavia uses other OpenStack projects to provide its services. The projects used by Octavia include the following:

- **Nova**: To bring up and manage the life cycle of instances on demand
- **Neutron**: To ensure connectivity between the amphorae, external networks, and the tenant environment
- **Keystone**: For authentication
- **Barbican**: To manage TLS certificates
- **Glance**: To store the VM images of amphorae
- **Oslo**: For communication between Octavia's controller components

The following diagram gives you an idea of the various components of Octavia:



OpenStack database service – Trove

A database is an important component of any big enterprise application. All cloud vendors should ensure that they have a good database as a service available for their users. **Trove** is a database as a service for OpenStack.

Trove is designed to run entirely on OpenStack. The overall idea of the service is to allow users to quickly use the features of a relational database without having to worry about administrative tasks, which can be complex. This service allows users to provision multiple database instances.

From a design point of view, Trove was designed to run a single-tenanted database within a Nova instance. Trove places no restrictions on how Nova should be configured.

Trove is made up of the following components:

- **The Trove API**: This is a RESTful API service. This allows users to provision and manage Trove instances. The RESTful API supports JSON and XML.
- **The Trove task manager**: This service is responsible for provisioning the Trove instances. The management of the instance life cycle is performed using this service. All operations on the database instance are also performed by this service.
- **The Trove guest agent**: This is an agent that runs on the guest instance. All operations of the database itself are performed using this agent. The agent listens for RPC messages and then performs the requested operations.
- **The Trove conductor**: This is also an agent, but unlike the guest agent, this runs on the host. This interacts with the guest instances to get updated information, such as the instance status, the status of a backup job, and so on.

OpenStack DNS service – Designate

DNS is an important service that many services rely on. **Designate** is the DNS-as-a-service solution from OpenStack. A variety of DNS servers are supported by Designate, including Bind9 and PowerDNS. Diesignate is a multitenanted service that uses a REST API for domain and record management, and can be configured to autogenerate records for Nova and Neutron actions.

OpenStack container orchestration – Magnum

With the increased popularity of microservices and containers, container orchestration has become a very important service. OpenStack's Magnum is a container-orchestration engine. It takes care of deploying containers and managing them.

A **cluster** forms part of the basic architecture of Magnum. A cluster is a collection of nodes. Work is scheduled on these nodes. Along with these clusters, Magnum also provides a cluster template. Objects can store cluster information in this template so that these templates can be used to create new clusters in a consistent manner.

The main features of Magnum are as follows:

- It provides an abstraction for clusters.
- It is integrated with popular container-orchestration software such as Kubernetes, Docker Swarm, and Mesos for its backend technology.
- It is integrated with Keystone to provide multitenant security.
- It is also integrated with Neutron for Kubernetes's multitenant network security.
- To provide volume services for the containers, it is integrated with Cinder.

OpenStack messaging system – Zaqar

Zaqar is a messaging service aimed at web and mobile developers. It is a multitenanted cloud messaging and notification service.

Using either the REST APIs or the WebSocket APIs provided by Zaqar, multiple components of a web or mobile application can communicate with one another. The scalability and security of the messaging engine is completely taken care of by Zaqar.

Some of the key features that are provided by Zaqar include the following:

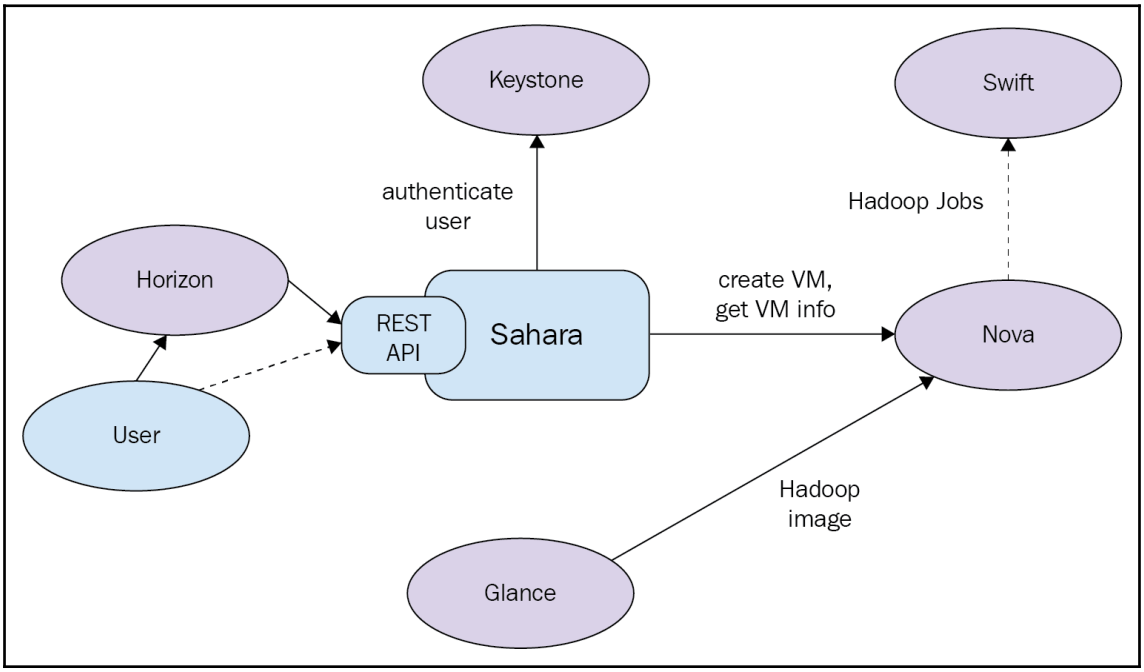
- Users have a choice of two communication transports, both of which have Keystone support. One option is to use the HTTP-based RESTful API. Many developers prefer HTTP-based RESTful APIs. The other option is to use WebSocket-based APIs. This is based on a persistent connection. Bear in mind that HTTP does not use persistent connections, and WebSocket APIs are better at reducing network traffic and network delays, since they can send multiple requests and responses over a single connection.
- Users can have multitenanted queues based on Keystone IDs.
- Common patterns, such as point-to-point and even broadcasting and task distribution, are supported.
- Subscription to the queues is supported.
- Various notification types are supported, such as email notifications, webhook notifications, and WebSocket notifications.

OpenStack data-processing service – Sahara

In the field of data analytics, there are multiple data-processing frameworks. Apache Hadoop is a popular data-processing framework. It is an implementation of MapReduce. Hadoop works on clusters, and services such as Amazon **Elastic MapReduce** (**EMR**) provide managed Hadoop clusters. **Sahara** is an OpenStack project that aims at providing users an easy way to provision and manage clusters for frameworks such as Hadoop, Spark, and Storm.

The user is asked to specify various parameters such as the cluster topology, the version of the framework, node type, and so on. Once the user specifies these parameters, Sahara will then provision the required clusters in a matter of minutes. Using Sahara, an already-existing cluster can be scaled out or scaled in by adding or removing nodes.

Sahara interacts with various other OpenStack components. The following diagram shows you some of the various component interactions with Sahara:



OpenStack resource-reservation service – Blazar

In many projects, the users may want certain resources dedicated to the project for a certain amount of time. Initially, there will be a need to reserve hardware resources, such as a compute host for a project. There might also be a need to reserve VMs. **Blazar** was created to address these requirements. A project could then expand beyond the compute host and VM reservation to allow users to reserve (or *lease*) cloud resources for a certain period of time. The cloud resource reserved could be a Nova compute instance, a Cinder volume, a compute host, and so on.

There are different kinds of leases, such as the following:

- **Immediate lease**: The resources are provisioned and reserved immediately, or not at all
- **Reservation with retries**: If the reservation fails, it will be retried for a configurable number of times
- **Best-effort reservation**
- **Scheduled reservation**: A lease is created if Blazar thinks that it can reserve the requested resources at a later point in time

OpenStack cluster service – Senlin

The clustering service of OpenStack is called **Senlin**. The main aim of Senlin is to orchestrate clusters.

Senlin provides an API that users can use to do the following:

- Add a node to the cluster
- Remove a node from the cluster
- Attach a policy to the cluster
- Detach a policy from the cluster

The policies that can be attached/detached include load-balancing policies, scaling policies, health policies, creation policies, deletion policies, and so on.

Computing inside storage objects – Storlets

Storlets allows users to run computations near the data in a secure manner. Storlets basically extend the Swift service.

A storlet is compiled and packaged code—in other words, an executable. Once the compilation and packaging is done, the packaged file, say a JAR file, is loaded to Swift. The packaged file is loaded as any other object would be. Once the file is uploaded, it can then be invoked in Swift, with the objects in Swift as the targets.

Application data protection – Karbor

Karbor is an application data protection service. Karbor provides a framework for protecting the data and metadata that make up an OpenStack-deployed application.

With the full protection option, many resources are protected. Each of these resources is protected using a different plugin in Karbor. The resources that are protected include the following:

- VMs
- VM images
- Storage volumes
- Networks
- Projects

Resource optimization service – Watcher

Optimization is one of the key concerns for any cloud service provider. A cloud service provider would want to optimize the costs of the data center and increase the operational efficiency. **Watcher** is a resource optimization service of OpenStack that helps in achieving the optimization goals.

Watcher provides an optimization loop that comprises measuring metrics, processing events, profiling, an optimization process, and a process to apply the action plan. This framework can help realize goals, such as reducing operational costs, increasing system performance using techniques such as VM migration, and increasing energy efficiency.

Watcher supports a pluggable architecture. This means that users can develop their own optimization algorithm using their data metrics and profilers. These can then be inserted into the Watcher framework.

Key manager service – Barbican

Barbican is OpenStack's key manager service. It helps users to store their cryptographic keys. Barbican provides secure storage, provisioning, and management of secret data. Users can store keys such as symmetric keys, asymmetric keys, certificates and binary data in this service.

Root-cause analysis service – Vitrage

Vitrage is a root-cause analysis service for OpenStack. This service organizes, analyzes, and expands OpenStack's alarms and events. The aim of the service is to provide insights into the root cause of a problem. It also helps in finding out the existence of a problem before the problem happens so as to proactively ensure that the appropriate action is taken to avoid the problem from occurring.

At a high level, the functionalities of Vitrage are as follows:

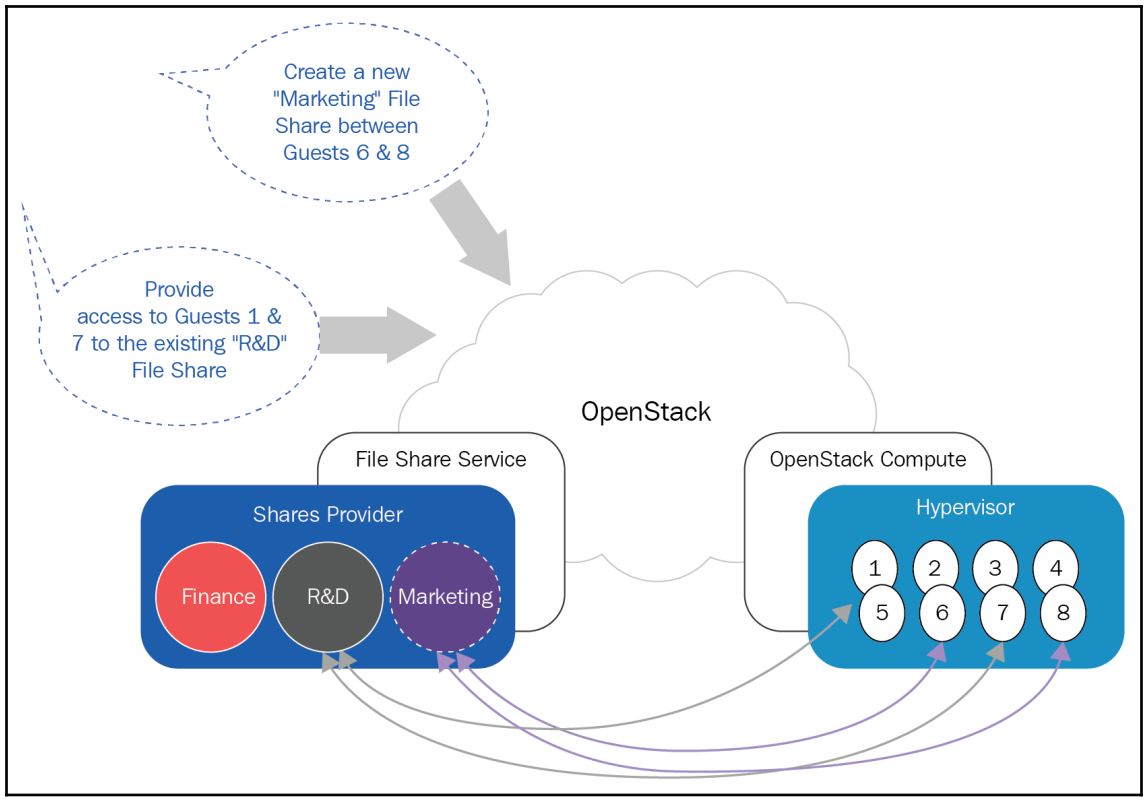
- Physical-to-virtual entity mapping
- Deducing an alarm or modifying a state, based on the current state of the system
- Root-cause analysis for alarms and events
- Providing plugins for Horizon for all the aforementioned features

OpenStack search service – Searchlight

Providing indexing and search capabilities across OpenStack resources is the aim of OpenStack's **Searchlight** project. OpenStack Searchlight uses ElasticSearch, a search server based on Apache Lucene. On top of this, Searchlight adds authentication and role-based access control to provide the appropriate protection of data.

OpenStack file-sharing service – Manila

OpenStack Manila is a file-sharing service. This allows multiple systems to share the same filesystem. As with shared filesystems, Manila is also based on the concept of **shares**. A share is the basic unit allocated by Manila. This is a readable, writable filesystem that can be mounted by multiple systems. The following diagram shows how the shared filesystem works:



Summary

In this chapter, we learned that OpenStack is a free and open source platform that allows users to build and maintain a cloud of their own. We also learned that OpenStack's architecture is made up of multiple projects. Each project is dedicated to providing a particular service: Nova for the compute service, Neutron for networking, Cinder for block storage, Swift for object storage, and so on. All these components help users build their own private clouds. Users can now build a highly scalable and modular private cloud on their own.

The popularity of OpenStack is due to various factors. Firstly, it is open source. This enables enterprises to build a cost-effective private cloud. Secondly, its design is modular. This enables newer services to be added easily and effectively. Thirdly, there is a large open source community working on OpenStack, and this is helpful for developers when it comes to understanding and resolving any issues in OpenStack. Finally, OpenStack architecture is built to be highly scalable. This is an important aspect to consider for any enterprise. Because of these factors, OpenStack can be used to build a cloud effectively.

In the next chapter, we will learn about the Microsoft hybrid cloud solution that provides Azure in your data center.

6
Network Interconnects to the Cloud

One of the most crucial elements that cloud usage depends on is ease and consistency of connectivity to cloud services. It is important that users of the cloud know how to connect to the different cloud providers. In this chapter, we will take a look at cloud connectivity and the various options that are available. Then, we will look at how cloud providers distribute their services.

You will learn about the tools that you will need to get more information about your cloud provider's network infrastructure and the key features that your cloud provider and network provider need to provide to make your cloud connectivity experience a success.

The following topics will be covered in this chapter, in detail:

- Connectivity providers
- The technology and terms related to cloud networks
- Connectivity options for cloud providers

Connectivity providers

When working in the field of clouds, you will come across a lot of cloud providers. These providers vary in their areas of expertise and types of business. In the next section, you will learn how to differentiate between these providers.

Network service providers

Network service providers basically sell bandwidth or network access directly to an internet backbone or an internet service provider. Usually, they also allow for access to their network access points.

Often, these providers are also referred to as backbone providers or internet providers.



With its backbone and cloud strategy, Microsoft is currently the second largest network service and backbone provider available. Currently, the largest network provider in the globe is AT&T.

Internet service providers

Internet service providers are organizations that provide access to the internet and internet services. The internet services offered by ISPs are internet access, internet transit, domain name registration, web hosting, Usenet services, and collocations.



Some examples of internet providers are AT&T, Google Fiber, 1&1 Versatel, and Verizon.

Internet exchange points

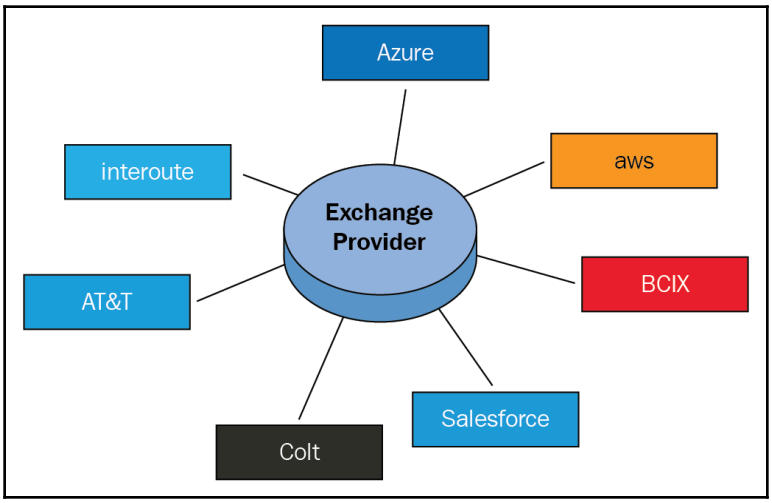
Internet exchanges offer a layer-2 platform and switch technology in different cities and locations around the globe. Network providers can use this platform to connect to other networks via public peering, and can extend their networks to third-party networks and services. In this way, internet exchanges help to connect and built the internet and reduce the costs for network providers.

Currently, there are more than 500 internet exchanges available around the globe. The six largest internet exchanges related to data traffic are DE-CIX (Frankfurt), AMS-IX (Amsterdam), IX-BR (Brazil), LINX (London), MSK-IX (Moscow), and DATAIX (Saint Petersburg). To see a good map of all global internet exchanges, you can visit website: `https://www.internetexchangemap.com/`.

Exchange providers

An exchange provider basically does the same thing as an internet exchange. The difference between an exchange and an internet provider is that an exchange provider enables private peering while internet exchanges focus on public peering.

The interconnect model of an exchange provider states that they want to connect to as many network and cloud providers as possible. The following diagram shows a graphical representation of the interconnect model:





Many small network providers prefer connecting to a cloud provider via an exchange provider, because exchange providers scale much faster than the network providers. Some examples of exchange providers include Equinix, Interxion, Megaport, and euNetworks.

Cloud and content providers

Content providers publish data on the internet as their major business or usage model. These content providers use internet exchanges and network providers to publish the content. You can think of a content provider as an information provider.

Sites like Netflix and Website Hoster are content providers; however, a cloud provider can also be seen as a content provider, because it helps customers to publish their content.

Technology and terms

When you work with cloud connectivity and cloud networks, you will often hear certain terms that are not very familiar to people that are used to working with classic on-premise environments. Most of these terms are encountered when dealing with internet service providers or network providers.

In the next section, you will learn about the most common terms.

Autonomous Systems

Autonomous Systems (**AS**) are collections of connected IP routing prefixes under the control of one or more network providers or operators.

An autonomous system needs to be controlled by a single entity. These entities are commonly internet service providers or large organizations with independent connections to multiple networks.

According to the new definition, defined under RFC 1930, multiple organizations can run Border Gateway Protocol routing. These organizations use a private AS number, not assigned by regional internet registries, to connect to an ISP. The ISP then connects the organization to the internet. Even if the ISP supports multiple autonomous systems, the internet only sees the routing policy of the one ISP. That ISP must have an official registered **Autonomous System Number**, or **ASN**.



You can find the RFC definition at `https://tools.ietf.org/html/rfc1930`.

The following table shows the ASNs for the most common cloud, content, and network providers:

Provider	Primary ASN	Additional ASN
Microsoft	8075	12076
Amazon Web Services	16509	38895
Netflix	2906	40027
Salesforce	14340	45422
Level 3	3549	3356

Border Gateway Protocol routing

The **Border Gateway Protocol** (**BGP**) is the successor to the Exterior Gateway Protocol. BGP is designed for routing between AS. BGP supports a metric and is able to make intelligent routing decisions. BGP can work with the internal routing protocol OSPF. Route aggregation is mostly used and supported in **Classless Inter-Domain Routing** (**CIDR**).

Open Shortest Path First

Open Shortest Path First (**OSPF**) is an IP-based routing protocol for networks using a **Link State Routing** (**LSR**) algorithm. OSPF falls into the category of **Interior Gateway Protocols** (**IGPs**).

Simply put, OSPF establishes a routing path based on the shortest path between the source and destination. By using this method, OSPF establishes a route with as little network latency as possible.

Private peering or private network interconnect

Private peering is the direct interconnection between two networks. This connection leverages a medium OSI, layer 1 or 2, with dedicated bandwidth. Private peering is not shared by any other network or third party. Nowadays, most private interconnections are established at carrier hotels or carrier-neutral collocation facilities called meet-me rooms. These are facilities in which a direct cross-connection can be provided between participants that are located in the same building or at the same data center campus. These network-to-network interconnects usually have much lower costs than telecommunication circuits.



A **telco circuit** is any line or provider that information or data is transmitted through.

Internet direct peering or public peering and remote peering

Public peering is accomplished across an exchange point, internet exchange, or a layer 2 access technology called a network access point. At these locations, multiple providers interconnect with one or more other carriers across a single physical port.

Public peering allows networks to interconnect, or "peer", with many other networks. Public peering, is often seen as offering a lower capacity than private peering because there are third-party costs for the internet exchange, but it allows for the connection to a larger number of networks. It also allows new network or content providers to send traffic to other networks without private peering.

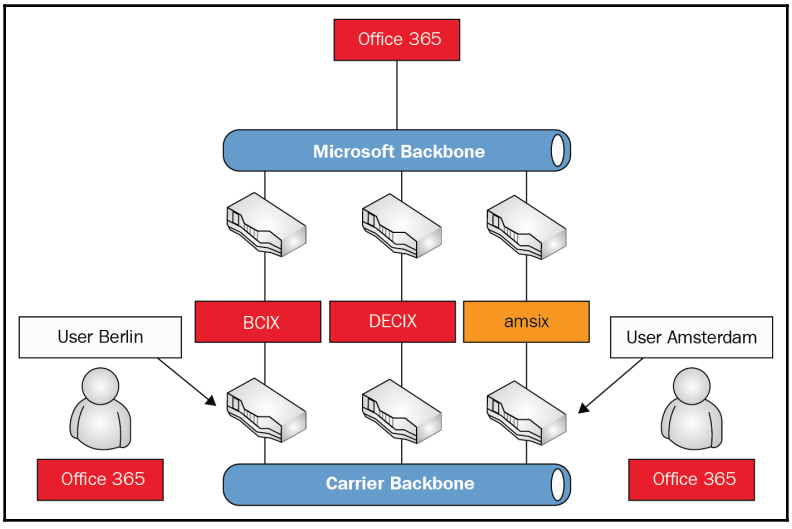


Most of the internet exchange points are commercially organized and offer carrier-neutral peering. The two most important conferences for those peers are the Global Peering Forum and the European Peering Forum.

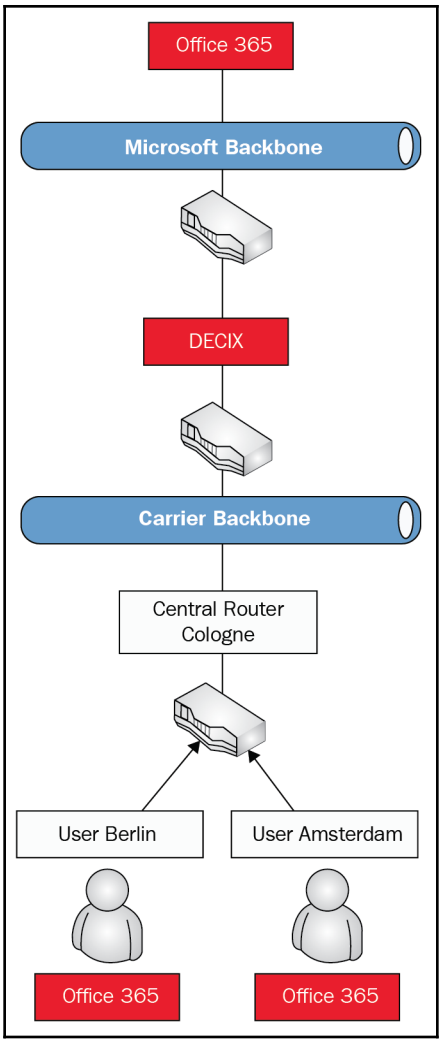
Decentralized and centralized routing

When looking at the network routing in these networks, most internet service providers and network providers follow two routing rules, as described in the following list:

- **Decentralized routing**: With decentralized routing, a provider first routes traffic that is determined for other networks, on the shortest path to the destination network. These providers usually have many private and/or public peering's to other providers. Those providers often offer good latencies with their connections to other networks and cloud providers, but because of the high amount of peering points, greater dark fiber networks, and lots of networking equipment, these providers are usually a bit more expensive. The following diagram shows an example of such a decentralized routing arrangement:



- **Centralized routing**: With centralized routing, a provider routes first traffic within its network and then redirects traffic for other predetermined networks as a bundle, to a lower number of peering locations with other network providers. This makes their networks very cost-efficient, but often results in higher latencies with other networks or services. The following diagram shows an example of such centralized routing:



Common connectivity options for cloud providers

Most cloud providers basically offer the same solutions to connect to their cloud services. In the next section, we will take a brief look at those solutions.

Encryption via the internet

The first type of connectivity that we will look at is direct connectivity to a cloud service via the internet. Most cloud service providers use HTTP and SSL, via port 443, to enable access. In those HTTP sessions, they encapsulate different protocols—such as the **Remote Desktop Protocol** (**RDP**)—mail traffic, API access, or regular website access.

Virtual private network tunnels via the internet

A **Virtual Private Network** (**VPN**) extends a private network across the internet and other public networks and enables companies to send and receive data across those shared networks. Devices connected to a VPN are directly connected to the private network of the organization. When we look at VPN tunnels over the internet, we can see that these tunnels have high limitations to their SLA, routing, performance, or high availability. You will get an SLA for most sites that use a VPN tunnel (as is the case with Azure, for example), for the virtual network gateways, or for internet access from your internet provider, but you will have nothing regarding the SLA in between when using the internet. The internet is still the best option.

Private network interconnect via Ethernet, MPLS, or an IPVPN provider

The primary service that all cloud providers offer is a private WAN connection via a network, possibly also including exchange providers. These private connections offer a direct connection to cloud services and cloud applications. These connections are mostly based on layer-2 Ethernet, layer-3 MPLS, or IPVPN connections to the cloud data center of the provider.

When working with an enterprise-grade cloud environment, these direct connections are necessary to provide low latency and highly available network connections to cloud resources. Some examples of these services are Microsoft ExpressRoute and AWS Direct Connect.

What matters the most

When working in a hosted environment or with cloud providers, there are some facts or that matter more than others. First, latency and bandwidth—they define how many cloud services you can consume and how fast you can reach your data. The second factor is the data that is stored and the data that is in transit. This information describes where your data is right now.

Latency and bandwidth

The best way to describe latency and bandwidth is by comparing the data flow to plumbing. Latency describes how long your pipeline is and how far the water needs to travel from end to end. The bandwidth refers to how much water you can pump through the pipe.

You will always need a combination of good latency and good bandwidth. Your pipeline needs to be big enough to let all of your data through, and needs to be short to provide better access to your services. For example, if you want to connect your toilet cistern with the toilet bowl, you will need to connect it directly, and not through the kitchen sink. You also wouldn't use a drinking straw as the outlet pipe if you needed to flush the water, which needs a waste pipe.

The following diagram illustrates bandwidth and latency:





In general, if you move a service to the cloud, you can say that, without WAN optimization, the service needs the same bandwidth as it needs on premise. There are options available to minimize this traffic, but it depends on the solution and protocols that are used.



For latency, there is a small, empirical formula. Latency is about 1 millisecond per 200 kilometers, which is nearly the speed of light. In the field, you will mostly see about 1 millisecond per 150 kilometers.

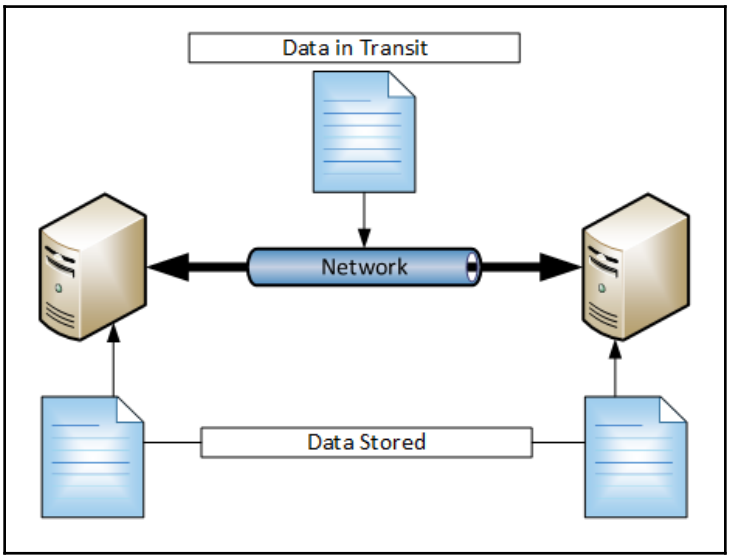
Data in transit and data that's stored

When you work with hosted services or clouds, you need to know the difference between data in transit and data stored.

With your data that is stored, you know and are able to define where your data is stored (for example, the data center or geographical region).

For data in transit, it is nearly impossible to define, because the data is traveling across carrier and data center networks and always takes the shortest available route. Data in transit does not know any borders. If the shortest route goes from a source in Germany to a destination in Norway via Russia, then the data will be transferred that way.

The following diagram shows a graphical explanation of both data types:



Summary

In this chapter, we covered the basics of cloud connectivity, the different providers, peering, and routing mechanisms. You should now be able to decide on the right provider to use to connect your different cloud providers and partners. You should also be able to decide whether you should use a connection via the internet or a private interconnect, and which routing method you should use between the different providers.

In the next chapters, you will learn how the different cloud providers put these connectivity solutions into practice.

7
An Overview of Network-Based Interconnect

In this chapter, we will take a closer look at Microsoft's connectivity options and how you can connect to Microsoft Cloud Services. You will learn about Microsoft's points of presence for internet peering and products such as Microsoft Azure ExpressRoute and Azure Virtual WAN.

The following topics will be covered in this chapter:

- Public interconnect and points of presence
- Private network interconnect
- Mixed interconnect with software-defined WAN

Public interconnect and points of presence

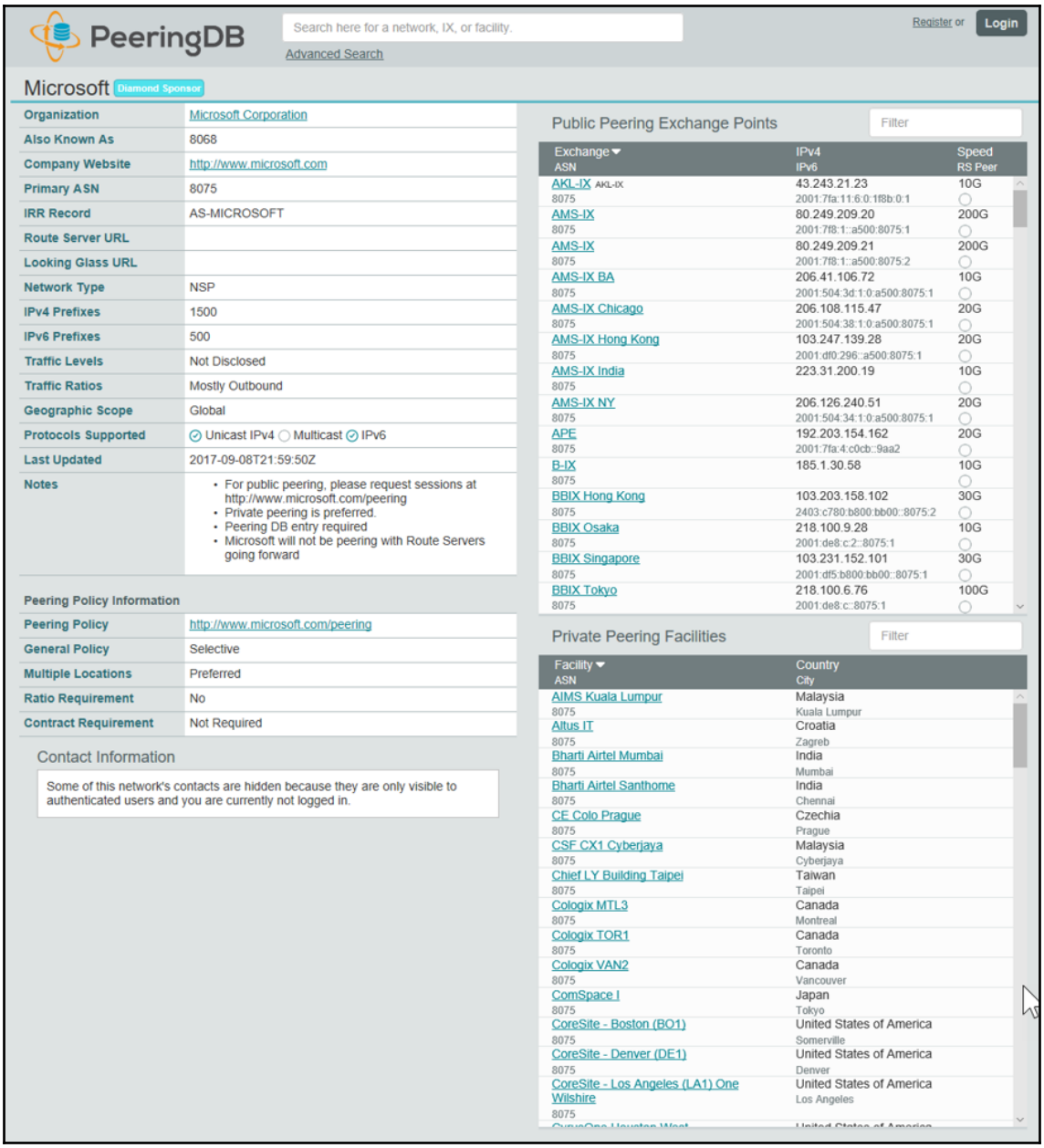
First, you need to know about Microsoft's points of presence. Microsoft is the second-largest network provider on the globe, and this is also reflected in their PoPs.

For public peering, Microsoft currently offers over 160 locations, and for private peering, they are available in more than 50 locations. Public and private peering is all handled via ASN 8075.



You can find the peering locations by going to `https://www.peeringdb.com/net/694`.

The following screenshot shows the entry on `http://peeringdb.com` for Microsoft AS8075:



Microsofts ExpressRoute's PoPs are also known, and can be found in the Microsoft documentation. The peering that AS Microsoft is using for ExpressRoute is known as ASN 12076.



You can find the peering locations and the available providers by looking at the documentation at `https://docs.microsoft.com/en-us/azure/expressroute/expressroute-locations`.

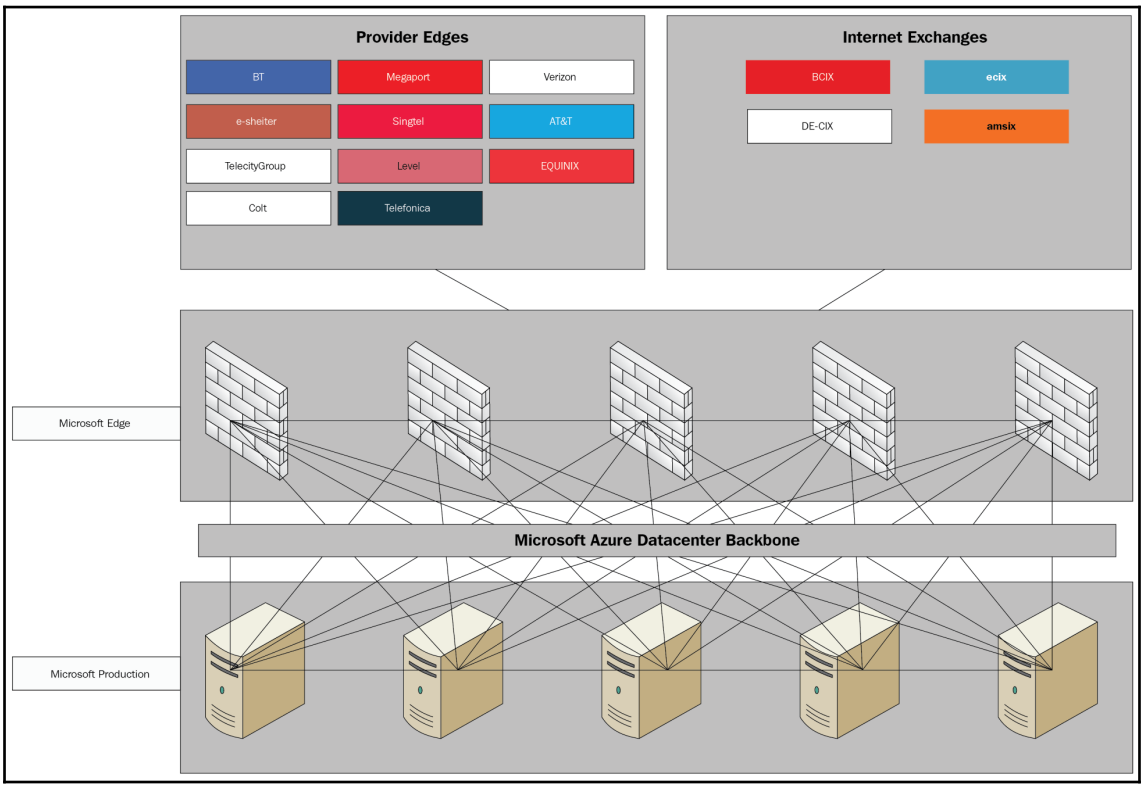
Microsoft has a very restricted and selective peering policy, which means that Microsoft does not peer with routeservers and unknown peers. Microsoft also offers only **Private Network Interconnects** (**PNIs**) to other network providers. Customers are not allowed to privately peer. Anyway, Microsoft still agrees to publicly peer with customers. A customer only needs to fit into Microsoft's peering policies, which basically includes the following rules:

- Peering takes place at an internet exchange
- Public peering AS should be used
- Public peering network should have at least a /24 mask
- There should be an entry in `http://peeringdb.com`
- The NOC should be reachable 24/7



To get more information about peering with Microsoft, and about the policies for PNI, you can visit `https://peering.azurewebsites.net/peering`. There, you can also request public peering with Microsoft AS 8075.

The following diagram shows you a schematic view of the structure behind a Microsoft PoP or Edge site:

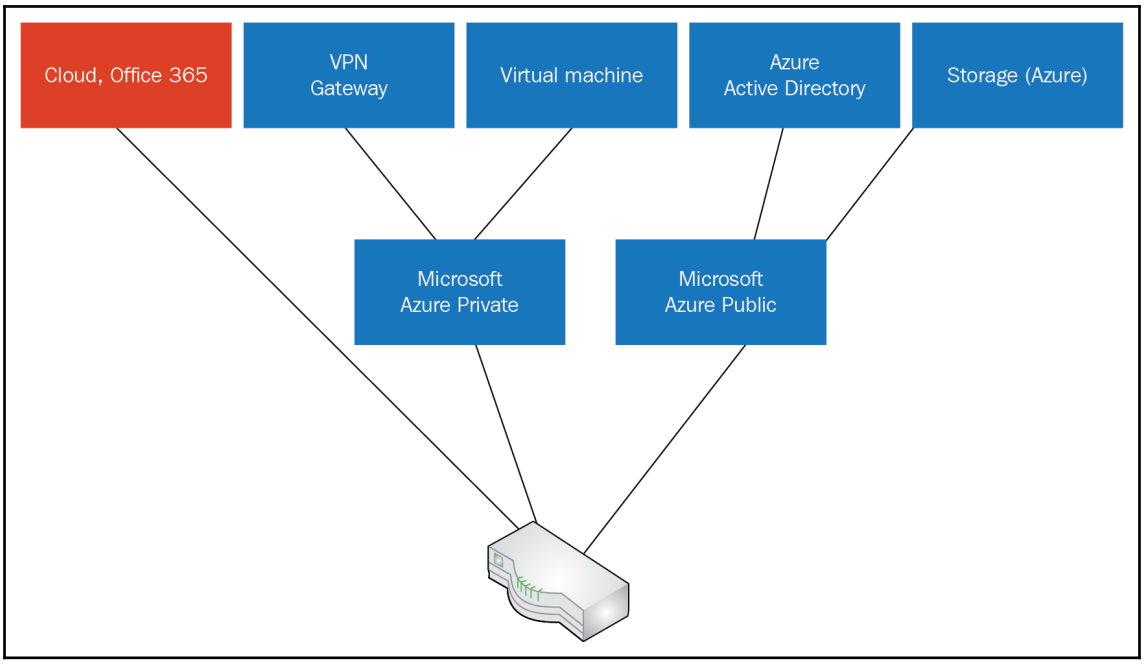


Interconnect via internet (HTTPs)

In general, Microsoft platforms and cloud services are available on a public IP only. There is no option, for example, to reach Office 365 via a private network, even if you use ExpressRoute. This also means that Microsoft cloud services communicate with one another on a public IP, even in the Microsoft global backbone.

Let's look at an example of a big layer-3 routing device, handling all internal and external requests from Microsoft, ensuring that Microsoft service communications stay on the backbone and are reachable for external access.

The following diagram shows a schematic view of this topology:



If a customer wants to restrict the access to their Azure tenant from the internet and only wants to allow certain IPs or IP ranges to have access, then they need to use an Azure Active Directory feature known as **conditional access**.



For more details on conditional access, you should consult the Azure documentation at `https://docs.microsoft.com/en-us/azure/active-directory/conditional-access/`.

Every piece of traffic you send or receive from Microsoft via the internet is secured, and at least 256-bit encrypted or more, a military standard of encryption. Microsoft also protects this traffic; you cannot open and inspect those packages sent from or to Microsoft cloud services without triggering a security mechanism that results in package drops. With every opened package, Microsoft suspects a "man-in-the-middle" attack and will prevent corrupted traffic from passing.

Microsoft also secures its endpoints from DDoS attacks by default, which means that all customers using Microsoft IP addresses have this protection.

Most of the Microsoft services are built for the internet, as you can see from the following statement from Microsoft documentation:

> ⓘ Note
>
> Office 365 was created to be accessed securely and reliably via the Internet. Because of this, we recommend ExpressRoute for specific scenarios. For information about using ExpressRoute to access Office 365, visit Azure ExpressRoute for Office 365.

There is only one scenario left: to get ExpressRoute to connect to Microsoft 365 services, as shown in the following screenshot (this can be found in the documentation at `https://docs.microsoft.com/en-us/office365/enterprise/azure-expressroute?redirectSourcePath=%252farticle%252f6d2534a2-c19c-4a99-be5e-33a0cee5d3bd`):

> ⓘ Note
>
> Microsoft authorization is required to use ExpressRoute for Office 365. Microsoft reviews every customer request and authorizes ExpressRoute for Office 365 usage when a customer's regulatory requirement mandates direct connectivity. If you have such requirements, please provide the text excerpt and web link to the regulation which you interpret to mean that direct connectivity is required in the ExpressRoute for Office 365 Request Form to begin a Microsoft review. Unauthorized subscriptions trying to create route filters for Office 365 will receive an error message.

Azure VPN

Azure virtual network gateways are core routers within an Azure virtual network. They connect an Azure network to different kinds of interconnect options. Those options are site-to-site VPN, point-to-site VPN, Azure virtual WAN, or ExpressRoute.

Every VNet can have at least one VPN gateway. VPN gateways are available in different service option with different features and available services. You can find a detailed list about the gateways in the Microsoft documentation at `https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-about-vpngateways#gwsku`.



Since Ignite 2018, Microsoft has extended this offering to network gateways. Addressing customer needs regarding better SLAs for gateways, it started to offer zone-redundant virtual network gateways for ExpressRoute and VPN. Those gateways are placed into different Azure data centers with separated power supplies, cooling, and data center environments. This prevents those gateways from data center outages and failures.

With Microsoft VPN or virtual network gateways, you have the following options to connect to your on-premises environment:

- **Policy based**: IPSec IKEv1, single-site connection with static routing
- **Route based**: IPSec IKEv2, multisite connection with static routing and BGP

Depending on the devices you connect to on-premises, you can either choose policy or route based. To known what your device supports, you can ask the vendor or check the Microsoft documentation at `https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-about-vpn-devices`.



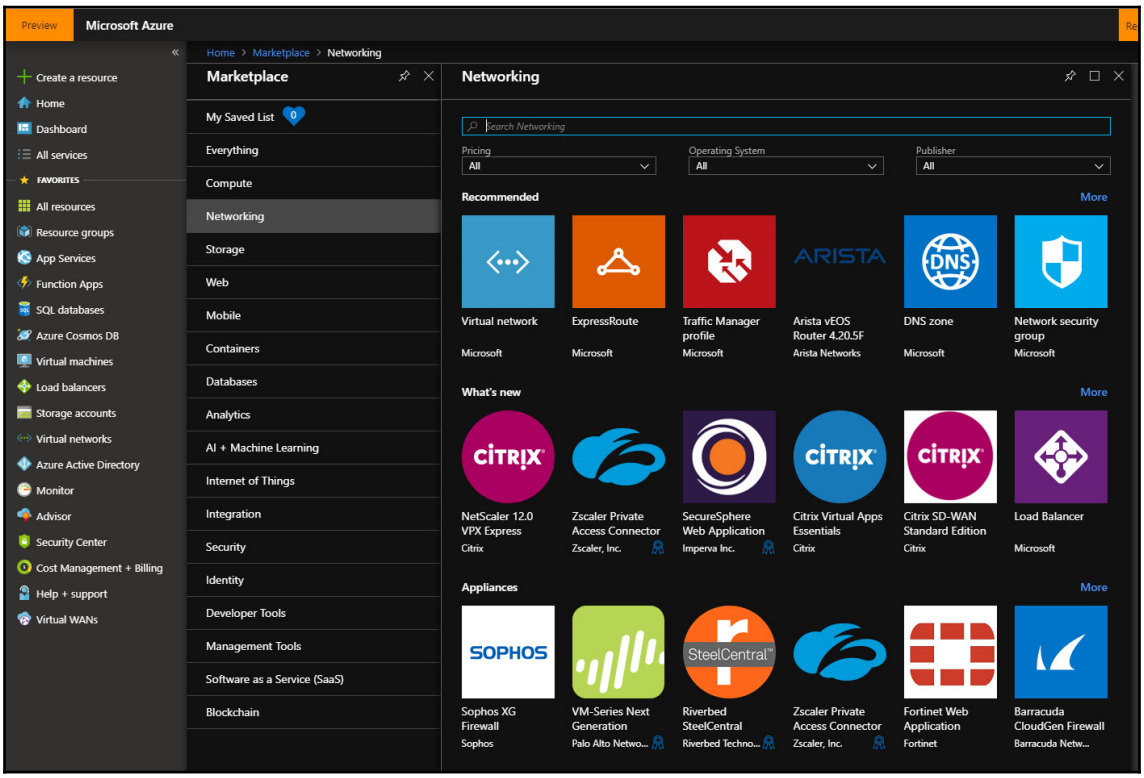
I would only use a route-based option. Most of the devices we have out there are able to connect to a route-based gateway or the route-based gateway, can emulate a policy-based option. Go to `https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-ipsecikepolicy-rm-powershell` for more details.

VPN with network virtual appliances in Azure

In addition to Microsoft's own offering regarding VPN and virtual gateways, it is also possible to integrate network virtual appliances, such as those available from Palo Alto, Barracuda, CheckPoint, and many other companies, into your virtual-machine environment on Azure. These VMs behave like their physical counterparts in most the cases, and enable more options regarding VPNs. As with Barracuda, you can also use a TINA Tunnel for VPN, a Cisco Miraki SDWAN, and so on.

NVAs are a good extra option if Microsoft has no network offering that fits your cloud strategy.

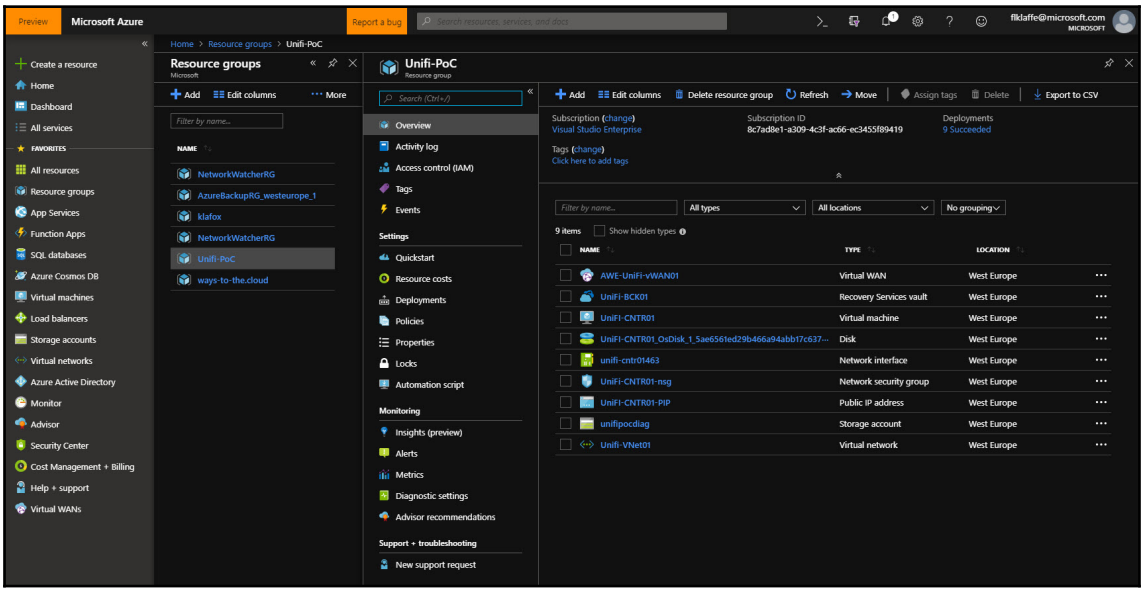
To find these products, you need to take a look at the Microsoft Azure Marketplace and order it via a third-party solution, as shown in the following screenshot:



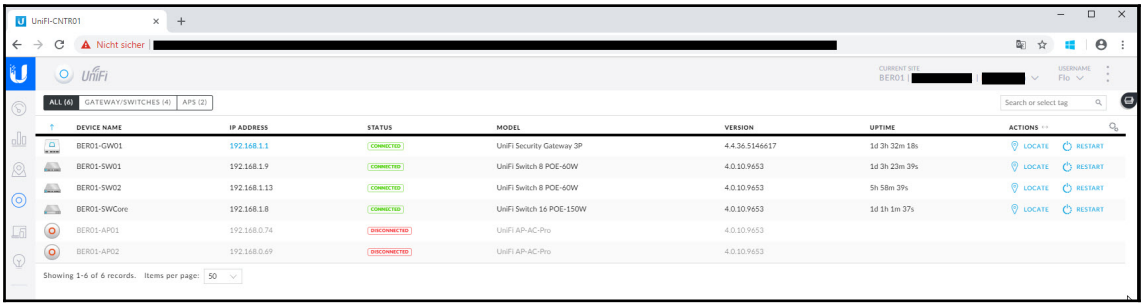


When ordering a third-party product from the Microsoft Azure Marketplace, you mostly have the choice between the "pay-as-you-go" and the "bring-your-own-license" options. From my experience, pay as you go is good for testing and short-term usage; if you want to use the appliances for a longer period, it is better to use the bring-your-own-license model. The prices offered by the marketplace have no discount, but third parties are often able to give you a discount on the license when ordering it directly via their distributions.

You can also set up your own network appliances on Azure as you can with Ubiquiti Unifi, where you can set up a Unifi controller on Azure or AWS to connect your locations and hardware to the cloud. The following screenshot shows a setup on Azure:



Hosting such network controller devices in cloud environments is a very interesting option to reduce the hardware footprint for management on premises, as shown in the following screenshot:



Private network interconnects

When talking about private network interconnects, we're mainly talking about connecting directly to cloud services without touching any public networks or internet breakouts. This means that you connect through a private network, such as an MPLS, layer-2 networks, or a colocation, to a cloud provider using a connectivity partner, such as Verizon, Equinix, Megaport, or AT&T. The following part of this chapter will look at Microsoft's options for private interconnects, and will explain how to use these technologies.

Azure ExpressRoute

When working with ExpressRoute, we are working with a very common technology that is commonly used by **Internet Service Providers** (**ISPs**). This technology is called **Multiprotocol Label Switching** (**MPLS**) or ISP IP VPN.

MPLS is a data-transfer technology for widely used networks that routes data from one network to another. The technology works using the shortest path labels and does not use long network addresses. This reduces overhead and long routing tables. MPLS can use the encapsulation of different network protocols and supports nearly every commonly used access technology (for example: T1/E1, frame relay dark fiber).

The routing in these wide-area networks is based on **Border Gateway Protocol** (**BGP**) routing. BGP is a standard, and is available with every enterprise layer-3 network gateway device. It has been designed to exchange the routing and availability information of different systems in WANs and other internet connectivity systems. BGP makes routing decisions based on the path, network policies, and rule set configured by a network provider or administrator. BGP then makes the core routing decisions for these networks.

Normally, you see MPLS when connecting a range of offices or data centers to very complex routing or mashed networks among network sites. MPLS also does not terminate **Quality of Service** (**QoS**) settings at the gateway; all settings can be transported from one network site to another network site.

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

When you configure your settings, you will be able to configure the different kinds of peering. The two options for peering are as follows:

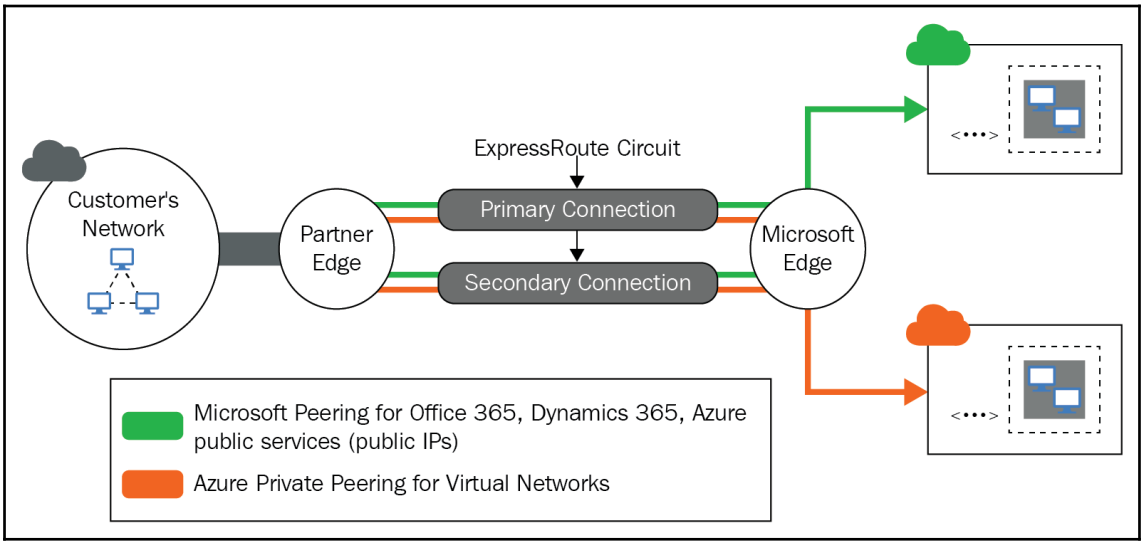
- **Private peering**: In general, a customer will use IaaS or other vNet-enabled cloud services. These services can be connected by using a private peering domain. This is an extension of a trusted vNet in Microsoft Azure and an existing tenant infrastructure. Peering can be configured as unidirectional or bidirectional.

- **Microsoft peering**: Within Microsoft peering, there are services such as Azure Storage, SQL databases, and websites that are offered on public IP addresses. You will be able to connect the Microsoft public peering Azure IP ranges to your DMZ or internal network and connect to all Azure services without having to connect through the internet. You enable bidirectional connectivity among your WAN and Microsoft cloud services.

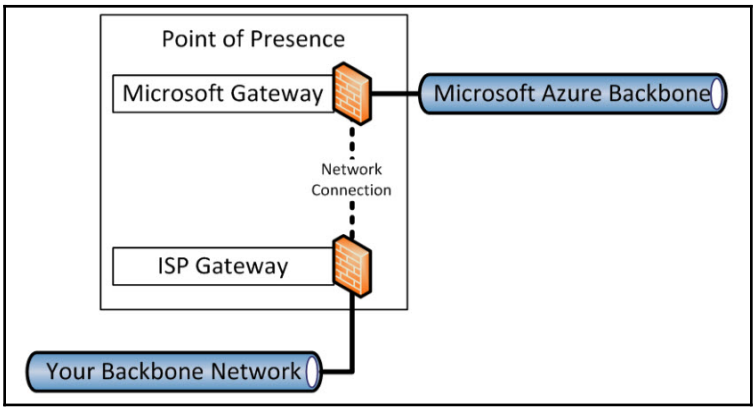


Because of the different deployment and distribution strategies from Azure and Microsoft 365 services, it is not recommended that you use ExpressRoute for Microsoft 365. You should only use ExpressRoute for Azure services and/or as your global MPLS backbone interconnect. You can find the Microsoft official statement regarding this at `https://docs.microsoft.com/en-us/office365/enterprise/azure-expressroute`.

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



What basically happens is that your ISP connects your network to the network of Microsoft. These connections happen at most of the point of presence, meet-me locations, or private network interconnect hubs all over the globe. The following diagram shows how this happens within the Azure data center:

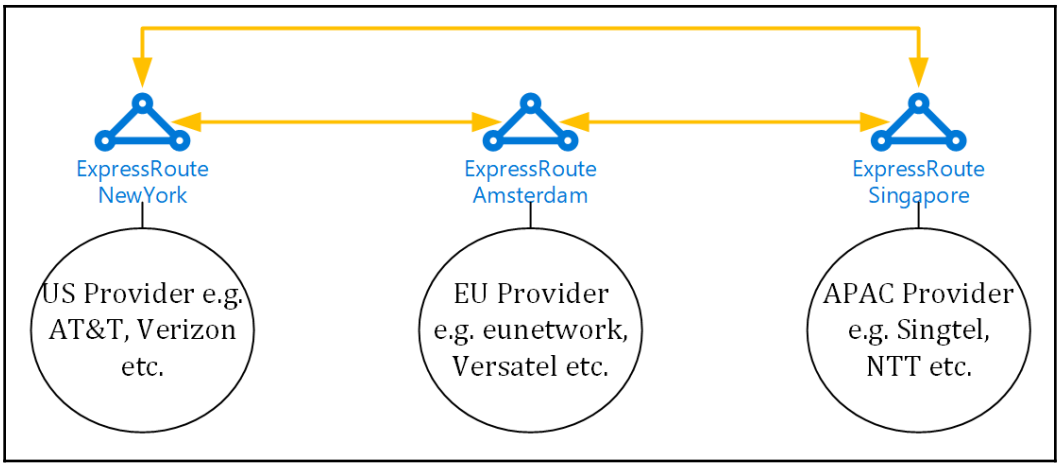


Microsoft offers ExpressRoute in two service levels: Standard SLA and Premium SLA. The premium offering expands the standard offering in the following ways:

- Increased routing table limit from 4,000 routes to 10,000 routes for private peering.
- A higher count of VNets that can be connected to the ExpressRoute circuit (default is 10).
- Global connectivity over the Microsoft core network with ExpressRoute Premium Circuit. This enables you to link a VNet in one geopolitical region with an ExpressRoute circuit in another region. For example, you can link a VNet created in Singapore to an ExpressRoute circuit created in New York.
- Connectivity to Office 365 services and CRM Online.

Azure ExpressRoute Global Reach

Express Route Global Reach provides a link between ExpressRoute circuits to enable a virtual private backbone between a company's on-premises network, using the Microsoft global backbone. This service makes Microsoft a backbone provider for customers and telecommunication providers. You can even integrate it into an existing ExpressRoute configuration. The following diagram shows a schematic overview of this arrangement:



With ExpressRoute Global Reach, customers get the option to always choose the best and most cost-effective local provider to use to connect to Azure. Customers also gain flexibility when choosing providers, and are able to easily exchange providers or build cheap WANs of their own. ExpressRoute Global Reach is also an option for smaller ISPs and network providers, because they can use ExpressRoute to use Microsoft cables across the Atlantic and Pacific oceans, instead of renting expensive dark fiber cables of their own.

ExpressRoute Global Reach is currently available in the following regions:

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

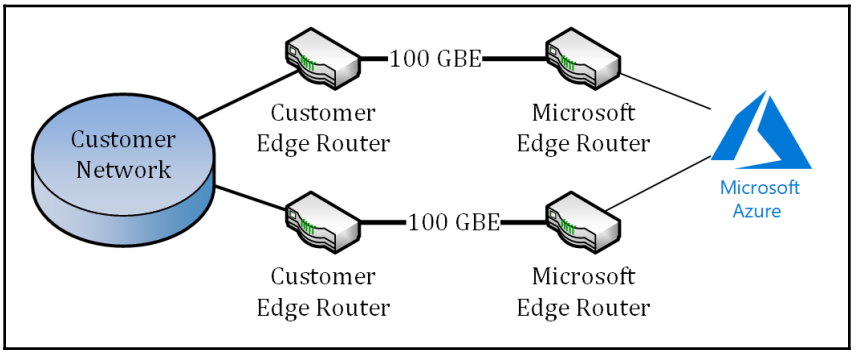


Microsoft is currently working to make more regions available. With ExpressRoute Global Reach, Microsoft is the second-largest telecom provider in the world, and there are therefore some government regulations and restrictions that need to be cleared first.

Azure ExpressRoute Direct

ExpressRoute Direct is a service that can connect to Microsoft's global network at peering locations around the world without needing to connect to other network providers. ExpressRoute Direct provides a redundant 100 Gbps connectivity, which supports active/active connectivity and scalability.

The following diagram shows a schematic view of how ExpressRoute Direct works:



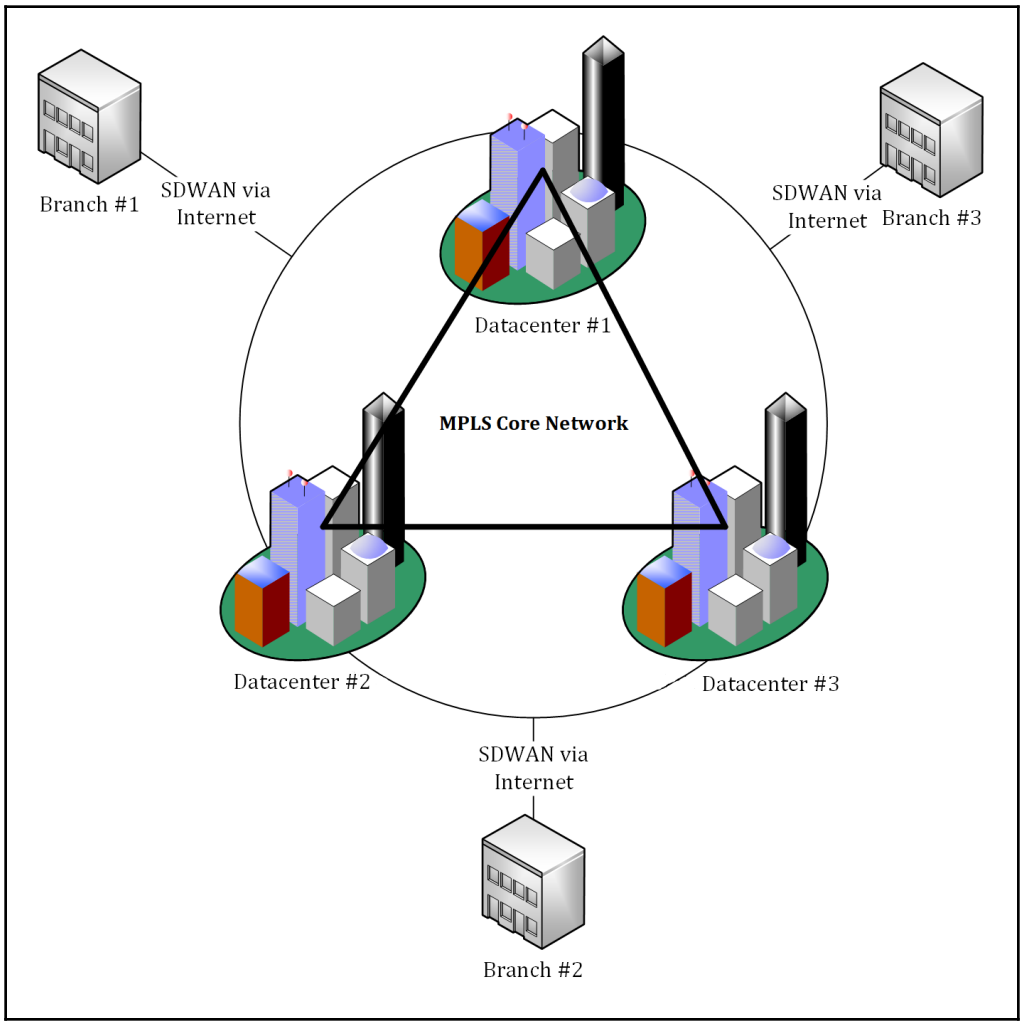
ExpressRoute Direct was built for customers who have very high bandwidth needs, as well as network providers who want to use the Microsoft global backbone and sea cables to extend their own network to new regions.

Mixed interconnect with software-defined WANs

Software-defined WAN (**SD-WAN**), is an acronym for **Software-Defined Networking** (**SDN**) and **Network Function Virtualization** (**NFV**) in a **wide area network** (**WAN**). The implementation of an SD-WAN should make the management and operation of a wide area network easier by separating the networking layer from the control and transport layer. This concept is nearly the same as software-defined networking implementations, where you use virtualization technology to improve data center network management and flexibility and minimize operational overhead.

A key feature of an SD-WAN is that it allows organizations to build higher-performance WANs using less costly and better commercially available internet access or by combining network technologies such as MPLS with other access technologies, such as DSL or fiber internet connections. This enables businesses to partially or wholly replace expensive private WAN-connection technologies, such as MPLS, or use those technologies for premium workloads, such as data center connections, and use cheaper technologies to connect branches.

The following diagram shows a schematic view of this arrangement:





Many companies are changing from using an MPLS network to connect their branches, to using a more efficient SD-WAN network with high-performance internet breakouts. VeloCloud did some research into this, which you can find at `http://www.velocloud.com/sd-wan-resources/infographics/top-6-sd-wan-trends`.

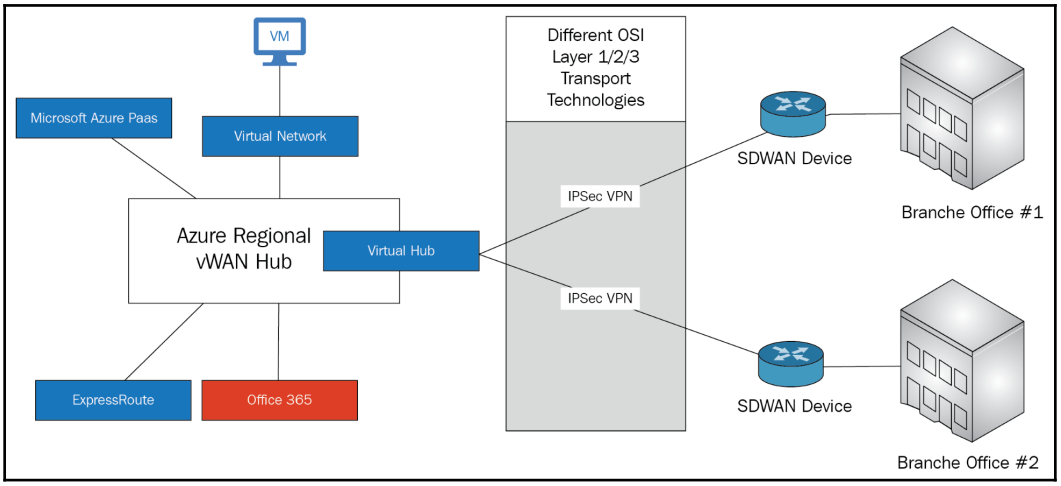
Azure Virtual WAN

Azure Virtual WAN is a network service from Azure that provides optimized, managed, reportable, and automated branch-to-branch connectivity through a public or a private network with Azure and the Microsoft global backbone. You build an SD-WAN aggregation in the Azure Virtual WAN hub. Virtual WAN lets customers connect and configure SD-WAN devices in branches to communicate with the Azure vWAN Hub. This can be through the manual configuration of (for example) devices that do not support automated deployment and configuration, or by using one of Microsoft's preferred partner appliances and devices that can be managed automatically.

You can find a list of preferred and already generally available devices and partners at `https://docs.microsoft.com/en-us/azure/virtual-wan/virtual-wan-locations-partners#connectivity-through-preferred-partners`.

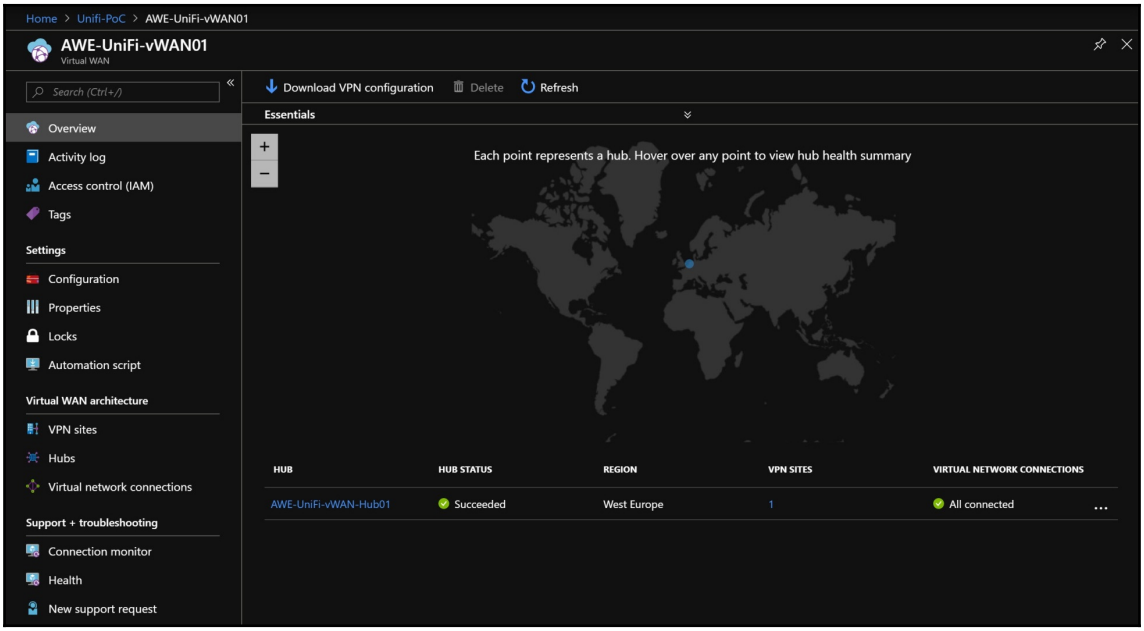
Using preferred partner devices allows you to auto configure the devices, thereby simplifying the management and deployment and enrolling new branches in a few hours, instead of days or weeks.

The following diagram shows a schematic view of the vWAN options:



The Azure WAN dashboard provides troubleshooting insights that help you to find failures and issues in your environment. With the dashboard, you can keep even a large-scale connectivity network under control.

The following screenshot shows the Azure WAN dashboard:



Summary

In this chapter, we took a deeper look at Microsoft's network architecture and products that help us to connect our cloud environments. We learned about connecting through the internet and the options for private interconnection. We also learned about the Microsoft global backbone and peering strategy, which should help you to decide the right peering and connectivity options for your future projects.

You should now be able to name Microsoft's interconnect products and limitations, as well as the possible solutions for most of the situations you will face when using a multi-cloud solution.

8
An Overview of Network-Based Interconnecting AWS

In this chapter, you will learn how AWS Cloud Solutions can be interconnected.

The following topics will be covered:

- **The Virtual Private Cloud** (**VPC**)
- Building a VPC
- Interconnecting to Amazon VPC
- Interconnecting between Amazon VPCs
- Interconnecting to AWS Services from inside a VPC
- Transit Gateway

To understand network-based interconnection to Amazon Webservices, we must understand the concept of the **VPC**. In the current AWS scheme of things, all connects happen to instances running inside the VPC. So, let's first understand the VPC. Once we are clear about it, then we can go ahead and look at all the interconnects that are possible.

A VPC

Let's say we want to deploy a three-tier application. The three tiers are as follows: web tier, application tier, and data tier. When such a deployment happens, the web tier is generally connected to the internet, whereas the application tier and data tier are hidden from the internet. In other words, the web tier can be public, whereas the other tiers must be private. (In actual large deployments, even the web tier is private, with a load balancer as the frontend for the client). So, we need to have a way of building subnets that can be controlled by us; that is, we should be in a position to make a subnet public or private. This is what a VPC sets out to accomplish.

A VPC is a virtual network. We can also say that a VPC is an isolated network, in the sense that if you build two VPCs in your account, each one is isolated from the other and instances in one VPC cannot communicate with instances in another VPC, by default.

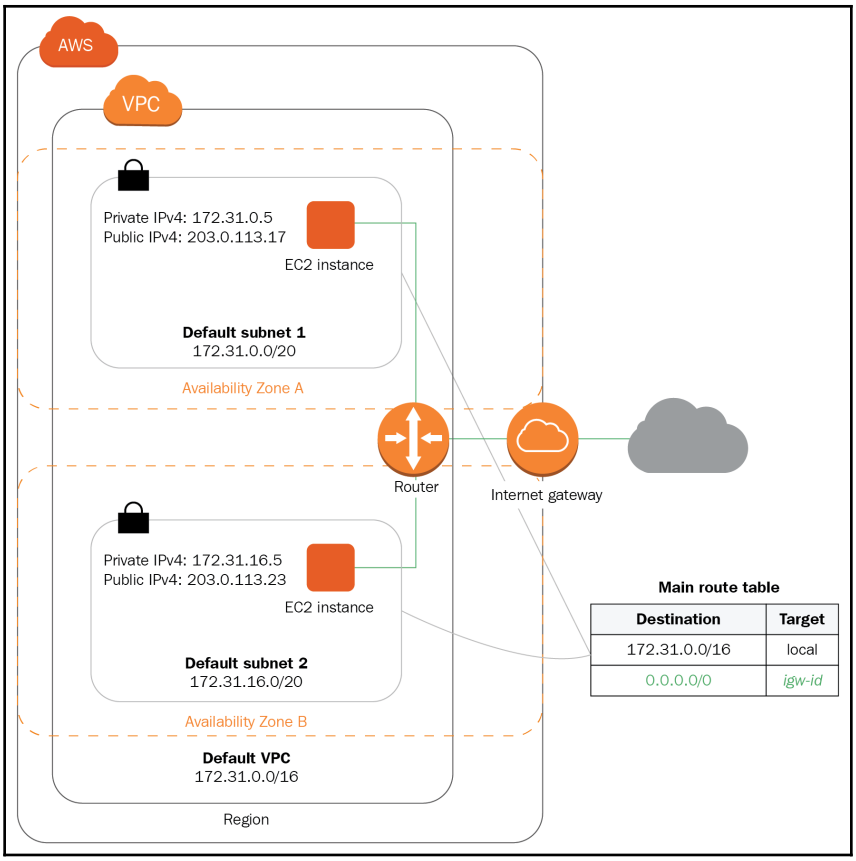
When you build a VPC, you first specify an IP address range. The maximum size of the CIDR range that can be attached to a VPC is `/16` (as per the CIDR notation), and the minimum size is `/28`. This VPC will belong to the region in which it is created and will span across all the availability zones in that region. AWS also provides you the ability to attach more CIDR ranges to the VPC. In total, we can attach five CIDR ranges to a VPC, each of which can be of a maximum of `/16` in size.

Once we create the VPC, we can then go ahead and create subnets inside the VPC, attach an internet gateway to the VPC, launch a NAT gateway, and perform other actions.

There are two kinds of VPCs in AWS: one called a **default VPC**, and another called a **non-default VPC**. A default VPC is set up by Amazon and is available to you in each of the regions once you sign up and log into the console. Any instance that is launched in a default VPC will be assigned a public IP. The user can then log into the system using the public IP, using protocols such as RDP or SSH.

A non-default VPC is a user-created VPC. If there are non-default VPCs, when an instance is launched in a subnet, the instance is not provided with a public IP. The instance only has a private IP attached to it. We will discuss the non-default VPCs more in the coming sections.

Here is a diagram of a default VPC:



Subnets

A **subnet** is a range of IP addresses within the VPC. AWS allows users to create a maximum of `200` subnets within a VPC. A subnet exists in an availability zone, and a subnet cannot span across availability zones.

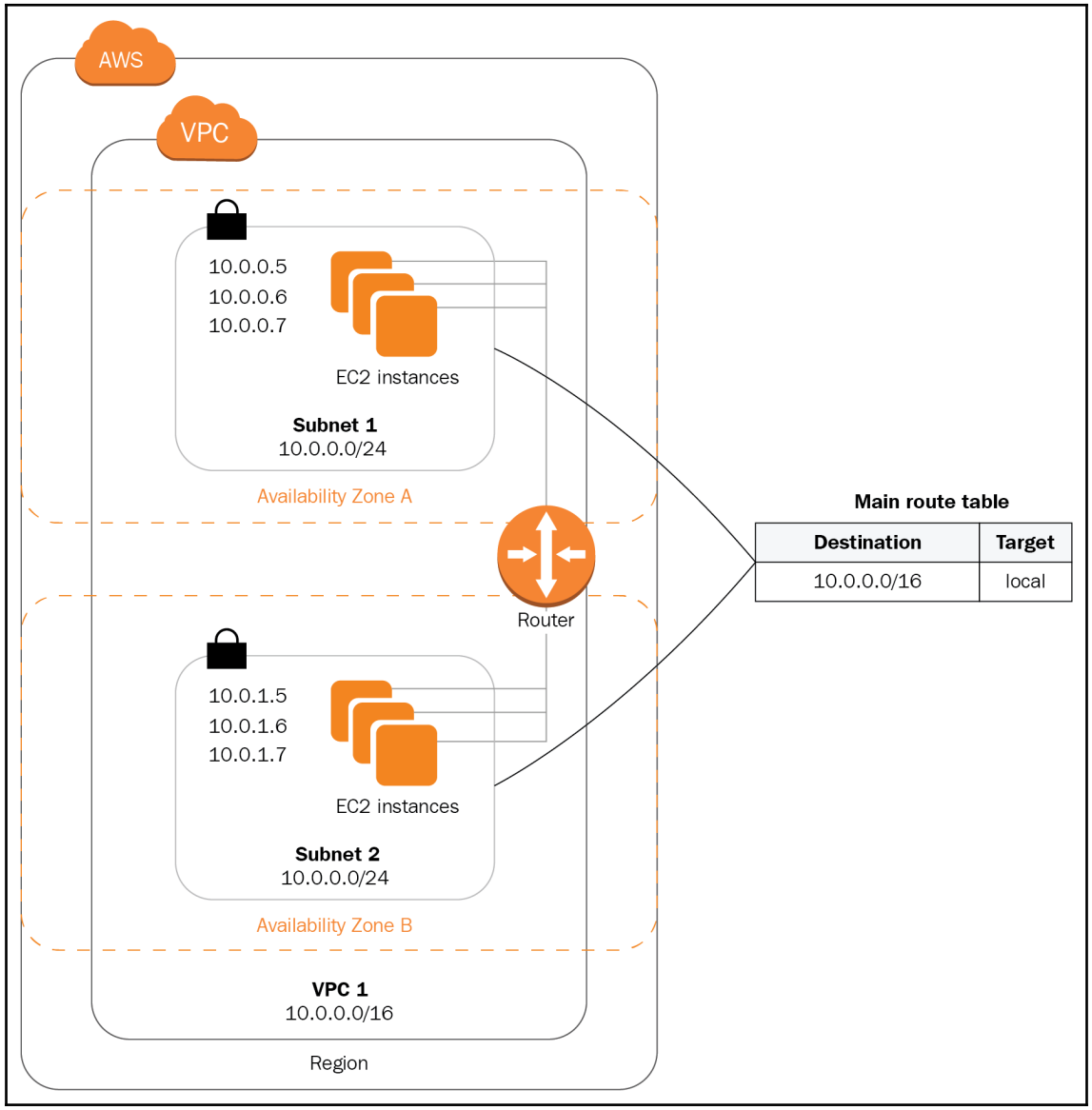
The maximum size of a subnet is `/16` and the minimum size of the subnet is `/28`.

In the default VPC, AWS creates one subnet per availability zone. These subnets are all public subnets. Additionally, every instance that is started within these public subnets will get a public IP.

In the non-default VPC, each subnet is created by the user and the subnet is created as a private subnet. Any instance started within the subnet will not have a public IP by default.

AWS automatically connects all subnets within a VPC using routing tables. This means that instances within different subnets of the same VPC can communicate with one another using the private IP address of the instance.

Here is a diagram of a non-default VPC with two subnets:



Internet gateways

As we saw earlier, a VPC is an isolated network. It means that a VPC is available only within the AWS infrastructure, and there is no way to connect to any instance inside the VPC. If we need to connect to any instance inside the VPC from the internet, we need to have an internet gateway, which will allow traffic from the internet to enter the VPC.

Amazon provides an internet gateway at no cost. Each VPC can be connected to, at most, one internet gateway. Once the internet gateway is requested, we then attach the internet gateway to the VPC of our choice. Once this is done, we can route traffic to and from the desired subnets to the internet gateway via routing tables. This will then make the subnets public subnets. The instances in these subnets can then access the internet, and the instances can be accessed from the internet, provided they have an Elastic IP attached to them.

The high availability and scaling of the internet gateway is taken care of by AWS. This is a managed service of AWS.

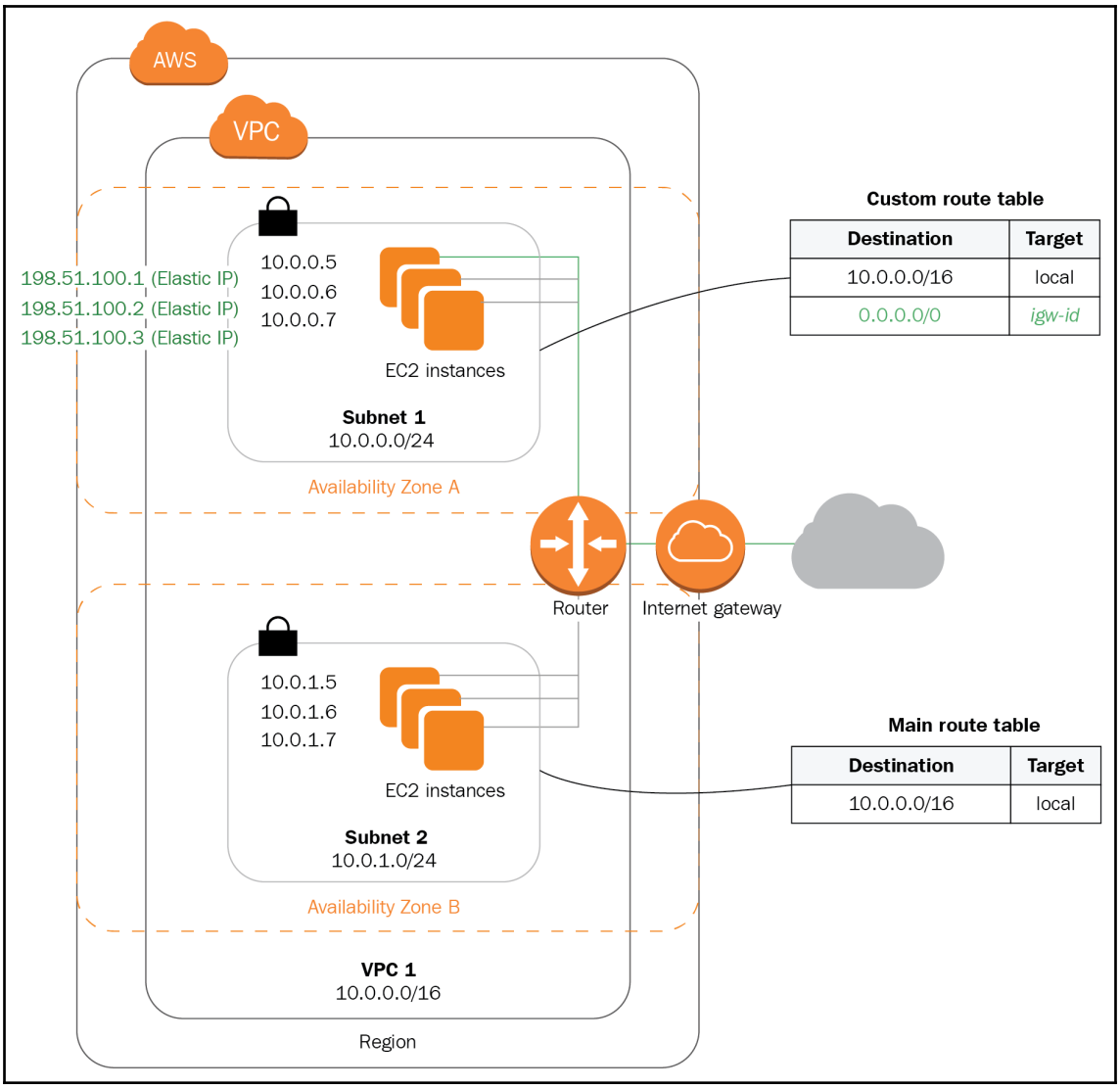
Routing tables

Routing tables are the most important part of a VPC . All connections between subnets and the external world happen by way of entries in the routing table.

Whenever a VPC is created, AWS automatically creates a routing table, called the **main routing table**. This routing table is attached to all subnets that are created. This default routing table has an entry that enables connections between various subnets.

Users can create their own routing table and attach it to any of the subnets in place of the main routing table. For example, if a user want to make one of the subnets public, then the can create a routing table in which the packets going out are sent to the internet gateway. Then, this routing table can be attached to the relevant subnet. Similarly, routing table changes allow the use of the NAT gateway and the VPC endpoint. As was said earlier, for all interconnections, the routing table is the most important entity.

Here is a diagram of a non-default VPC with an internet gateway and routing tables:



NAT gateways

As we saw earlier, subnets can be of two types: a public subnet and a private subnet. An instance inside the public subnet, if it has a public IP, can access the internet, and the instance can be accessed from the internet. The instance in a private subnet cannot access the internet, and the instance cannot be accessed from the internet.

There are times when we need internet connectivity for instances running inside private subnets. For example, assume you are running a database server inside a private subnet. This database server may need to download security patches and updates. We would need to access the internet to do so. The challenge is to access the internet without having a public IP, and without converting this subnet to a public subnet. In other words, the instances inside a private subnet should be able to reach the internet, but they should not be accessible from the internet. This is done using NAT gateway.

NAT gateways are used to provide internet access to instances in the private subnet. A NAT gateways is a service provided by Amazon. AWS takes care of the high availability and scaling of NAT gateways.

NAT Gateway works by sending requests on behalf of private instances to the internet and passing the responses back to the private instance. The connection between a private subnet and the NAT Gateway is achieved using the routing table.

The Network Access Control List (NACL)

Network Access Control List (**NACL**) is a collection of firewalls that are placed in front of every subnet. AWS has different kinds of firewalls: web access firewalls, security groups, and NACLs. Security groups are instance firewalls. They protect an instance. An NACL, on the other hand, protects the subnet.

By default, the main NACL is created by AWS when a non-default VPC is created. This NACL is attached to each of the subnets created. The default NACL allows all incoming and outgoing traffic.

An NACL protects the subnets based on the rules specified. An NACL is a stateless firewall, so rules need to be clearly specified for both incoming and outgoing traffic. Each rule has a rule number associated with it, and an NACL processes the rules in ascending order of the rule numbers. Rule numbers can start from `1` and can go up to `65,535`. The processing stops if one of the rules meets the condition.

Unlike security groups, an NACL allows for explicit `DENY`. So, we can both `ALLOW` and `DENY` traffic. This comes in handy in more than one situation.

Users can create their own NACL and attach it to the desired subnet in place of the default NACL. For example, if we want only SSH and MySQL traffic to enter a subnet that hosts a MySQL database server, we can write the appropriate NACL and attach it to the subnet, replacing the default NACL.

A sample NACL table is given here:

Rule#	Type	Protocol	Port range	Source	Action	Comments
100	SSH	TCP	`22`	`192.168.1.10`	`ALLOW`	Allow SSH connection
200	HTTP	TCP	`80`	ALL	`ALLOW`	Allow HTTP for all
300	HTTPS	TCP	`443`	ALL	`ALLOW`	Allow HTTPS for all
400	SSH	TCP	`22`	ALL	`DENY`	`DENY` SSH for all
*	All Traffic	ALL	ALL	`0.0.0.0/0`	`DENY`	`DENY` everything else

Building a VPC

We have seen the various components of the **VPC**. To understand how each component fits into the overall scheme of things, let's build a sample VPC. This will further improve our understanding of VPCs and will enable us to clearly understand the interconnects to AWS.

We will now look at how to create a VPC with the following properties:

- **VPC IPv4 address range**: `10.0.0.0` to `10.0.255.255` (that is, `10.0.0.0/16`)
- Two subnets
- **First subnet IP range**: `10.0.0.0 to 10.0.0.255` (that is, `10.0.0.0/24`)
- **Second subnet IP range**: `10.0.1.0` to `10.0.1.255` (that is, `10.0.1.0/24`)
- Create an internet gateway
- Modify routing tables to make the first subnet a public subnet
- Create a network gateway
- Modify the routing table to enable instances in the second subnet to access the internet
- Modify NACL rules

Step 1 – Creating a VPC

To create a VPC, we have to specify its IP address range. In our example, we are building a VPC of size `/16`, so we will specify our IP address range to be `10.0.0.0/16`. Furthermore, we will give a name to this VPC, which will enable us to identify it in the dashboard.

Step 2 – Creating the subnets

We now create two subnets using the following steps:

1. Provide a name for the first subnet to identify it.
2. Input the CIDR of the first subnet as `10.0.0.0/24`.
3. Provide a name for the second subnet.
4. Input the CIDR of the second subnet as `10.0.1.0/24`.

We have now created two subnets. Both subnets will be private by default. Our aim is to make the first subnet a public subnet. For that, we need to create an internet gateway.

Step 3 – Creating an internet gateway

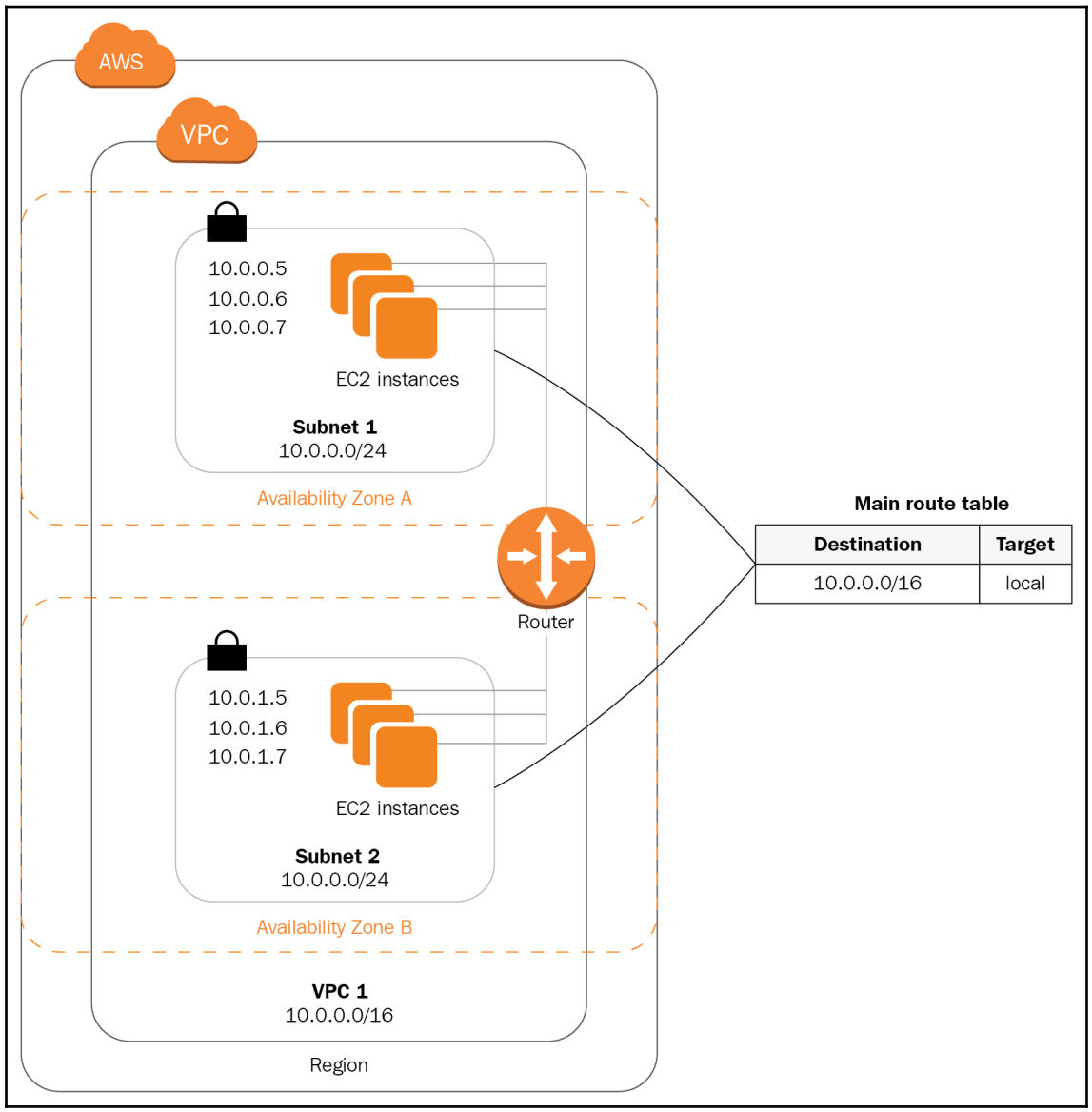
AWS provides the internet gateway. To create an internet gateway, the only parameter we need to input is the name of the gateway. Once that parameter has been input, the internet gateway is created.

When the internet gateway is created, it is in a detached state. It has not yet been attached to any VPC. We now attach the internet gateway to our VPC.

Step 4 – Creating a routing table

As stated earlier, our aim is to make the first subnet a public subnet. To do that, we need to create a new routing table and attach it to the first subnet.

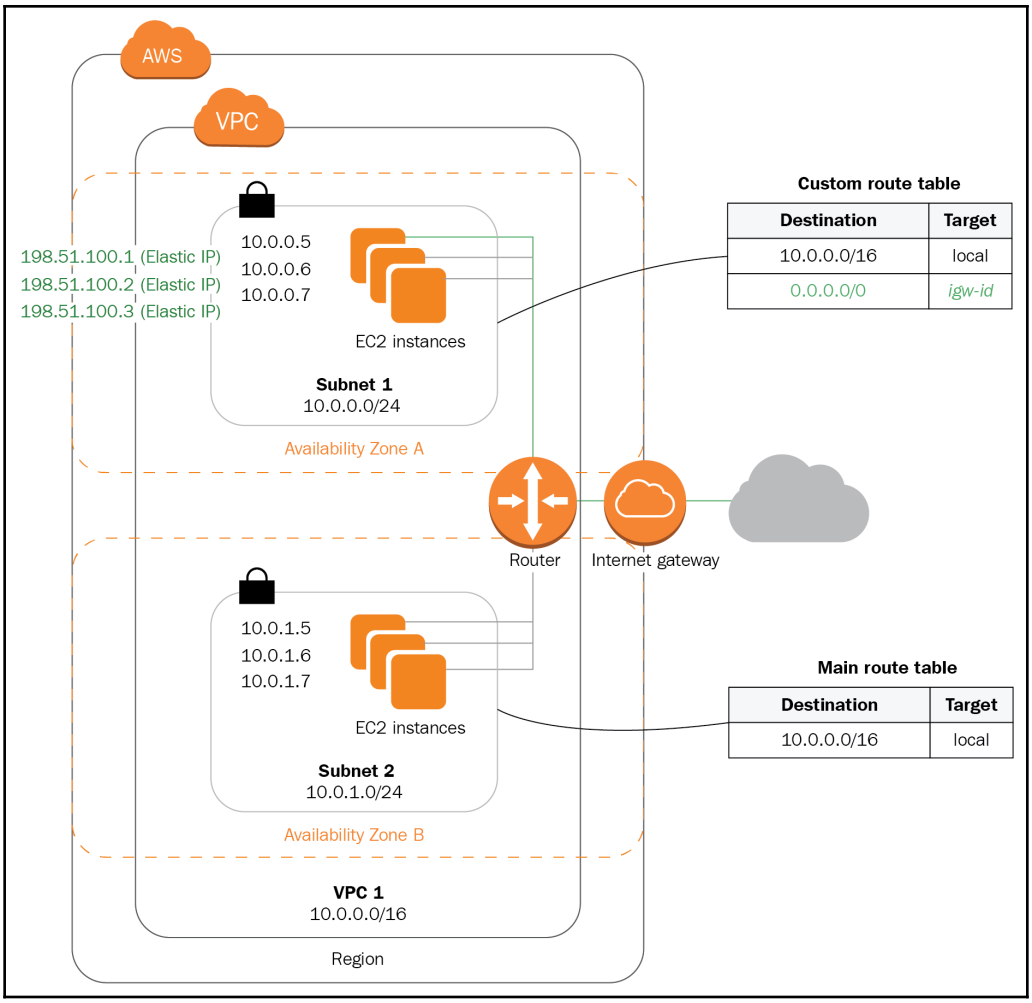
The entry in the main route table is given in the following diagram. This main route table is attached to all subnets by default:



We now create a new route table. We provide a name for the table. The entry in our table should enable all traffic, other than local traffic, to be sent to the internet gateway. In other words, the table must implement the following:

- All traffic out of the subnet whose destination is `10.0.0.0/16` must be routed within the VPC (that is locally).
- All traffic to any other destination must be routed to the internet gateway.

This is how our new routing table will look:



Now we need to associate the routing table with the first subnet. Once the association is done, the first subnet will become a public subnet.

Step 5 – Launching instances and associating Elastic IP

We can now launch instances in either the public subnet or the private subnet. In AWS, we do this by choosing the appropriate VPC and the subnet in the Configure Instance Details screen when the instances are launched using the console.

The instances by default will not have any public IP address associated with them. To provide a public IP address, we need to request AWS for an Elastic IP. AWS will provide up to five Elastic IPs per region per customer. (If more Elastic IPs are needed, we need to raise a request.)

Once the Elastic IP is allocated, we need to associate it with an instance. We will associate an Elastic IP with an instance in the public subnet. Once an Elastic IP has been associated with an instance, we can connect to that instance in the public subnet.

Step 6 – Creating a NAT gateway

The NAT gateway is provided by AWS. The scaling and high availability of the NAT gateway is taken care of by AWS. The only parameter that needs to be provided by the user is the name of the NAT gateway. The user must also indicate whether a new EIP is needed for NAT gateway or whether the user wants to associate one of their EIPs to the NAT gateway. NAT gateway must be started in the public subnet.

As we saw earlier, a NAT gateway is used to provide internet access to instances in the private subnet, which have only a private IP address and don't have a public IP address. To do so, we need to associate a different routing table to the private subnet.

The routing table to be created will have the following entries:

- All traffic out of the subnet whose destination is `10.0.0.0/16` must be routed within the VPC (that is, locally).
- All traffic to any other destination must be routed to the NAT gateway.

Once the routing table is created, we need to change the subnet associations, so that this routing table is attached to the private subnet.

With these steps, we have created a non-default VPC.

Interconnecting to an Amazon VPC

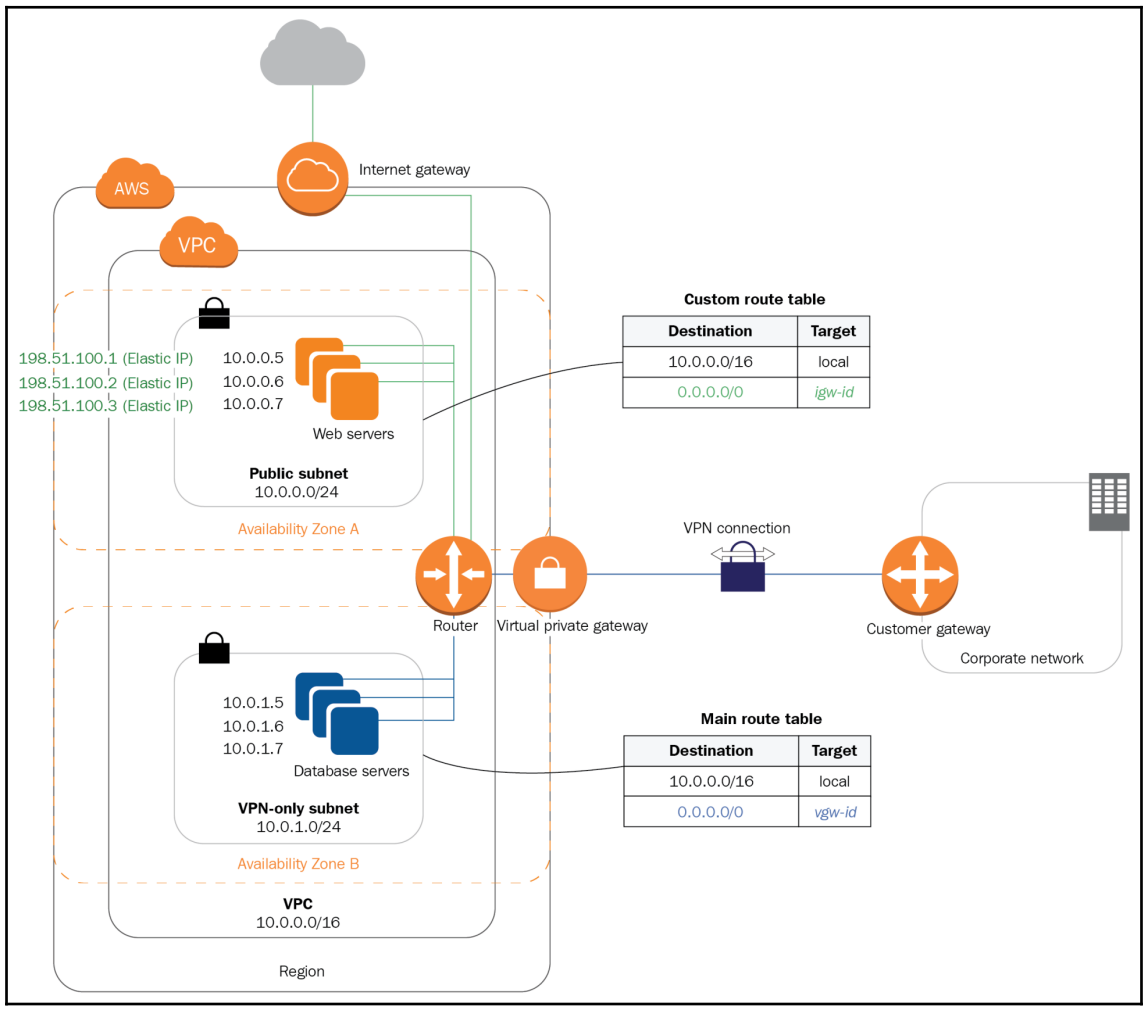
There are multiple connectivity options for connecting users to an Amazon VPC. Furthermore, we can connect VPCs to one another. We will look at all these options in the coming sections.

The end user to VPC inter-connectivity options would include the following:

- Using a hardware VPN
- Using a software VPN
- Using Direct Connect
- Using VPN CloudHub
- Connecting without a VPN

Connecting to an Amazon VPC using a hardware VPN

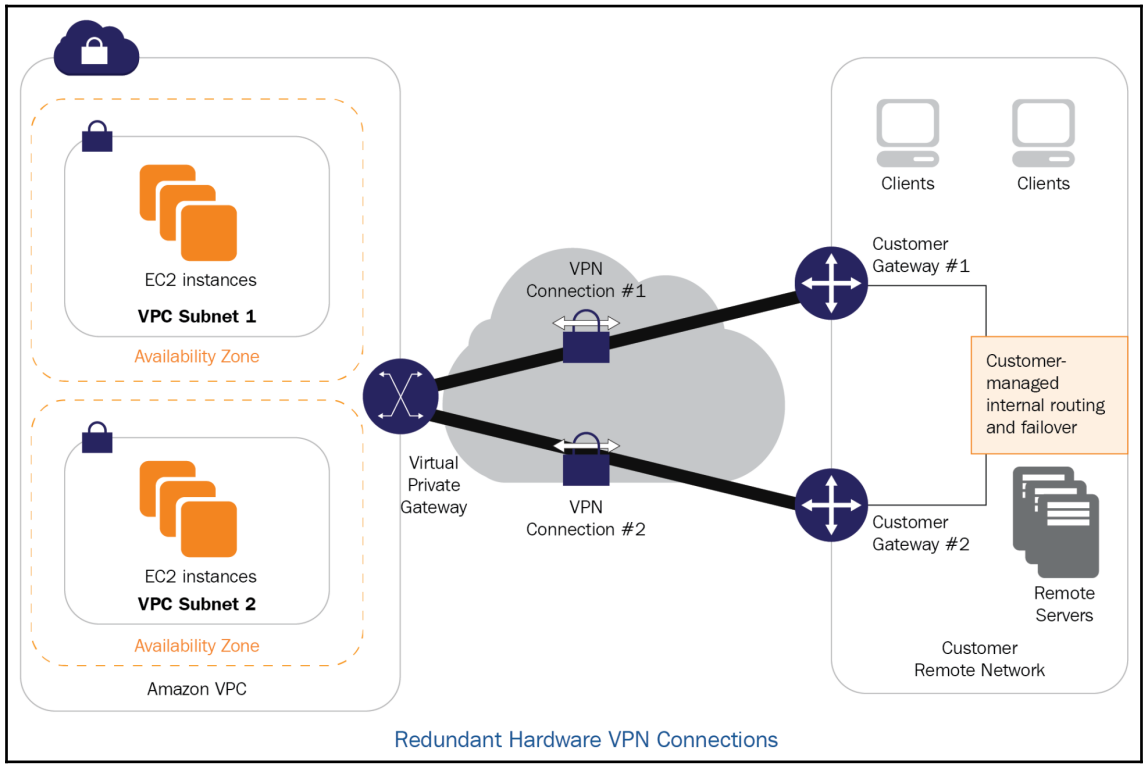
In this type of interconnect, the end-user premises has a hardware VPN, and on the Amazon side, and there is a **Virtual Private Gateway** (**VGW**) provided by AWS. Using these, the customer network can be connected to the VPC in AWS. The connection is shown in this diagram:



This option enables the creation of an IPsec connection between the customer network and the VPC. The VGW is completely managed by Amazon and has two endpoints in two different availability zones. These redundant endpoints ensure the high availability of the VPN connection.

The VGW supports both static and dynamic routing options, enabling the customer to choose according to their preference. In the case of dynamic routing, BGP peering is used to exchange routing information between the customer network and the VPC. BGP allows users to specify routing policies, weights, and priorities. This enables users to influence the path taken between the user network and AWS.

As stated earlier, the VGW has redundant end points. Amazon encourages users to also use multiple gateways so that redundancy can be implemented on the user's end as well. A diagram of a redundant connection using a VPN and VGW is given here:



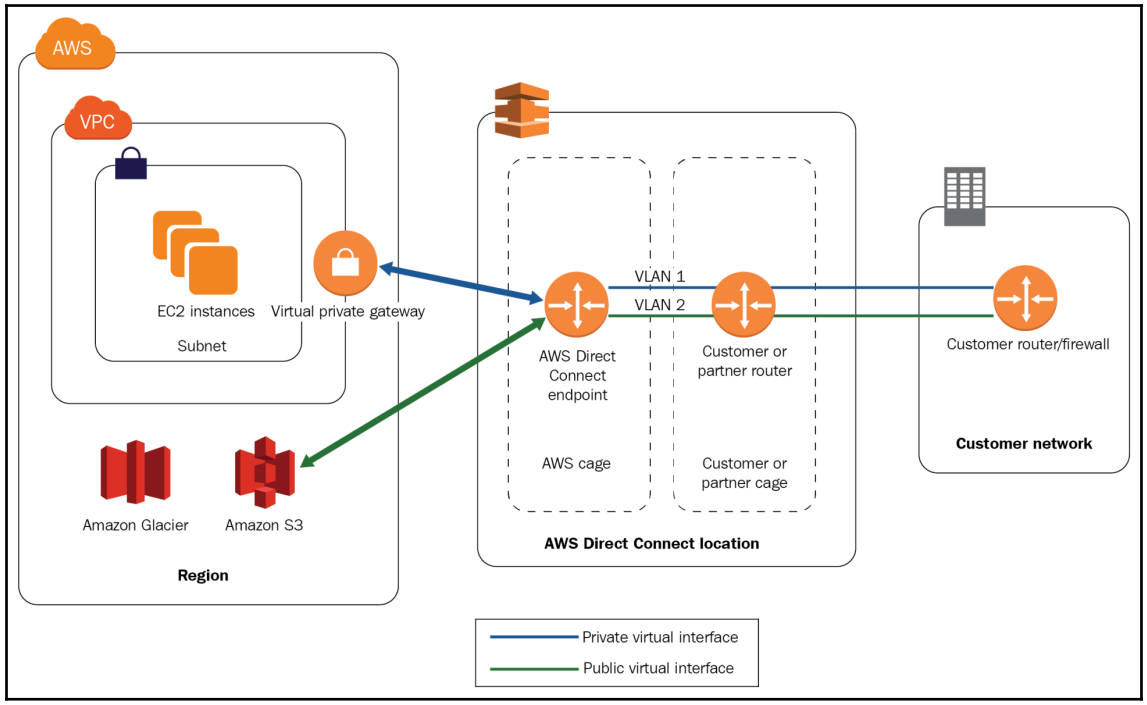
Redundant Hardware VPN Connections

Interconnecting using Direct Connect

Amazon's Direct Connect allows users to establish a direct connection between their network and the Amazon network. This is a private connection and does not use the internet. With Direct Connect, it is possible to transfer data over this private connection instead of using the internet. This helps with increasing the speed of transfer. This will also provide the users with a consistent network experience.

AWS Direct Connect enables users to establish a 1 Gbps or 10 Gbps connection. AWS uses industry standard VLANs to access resources such as EC2 instances. The EC2 instances in the VPC are accessed using their private IPS.

A diagram detailing Direct Connect is given here:



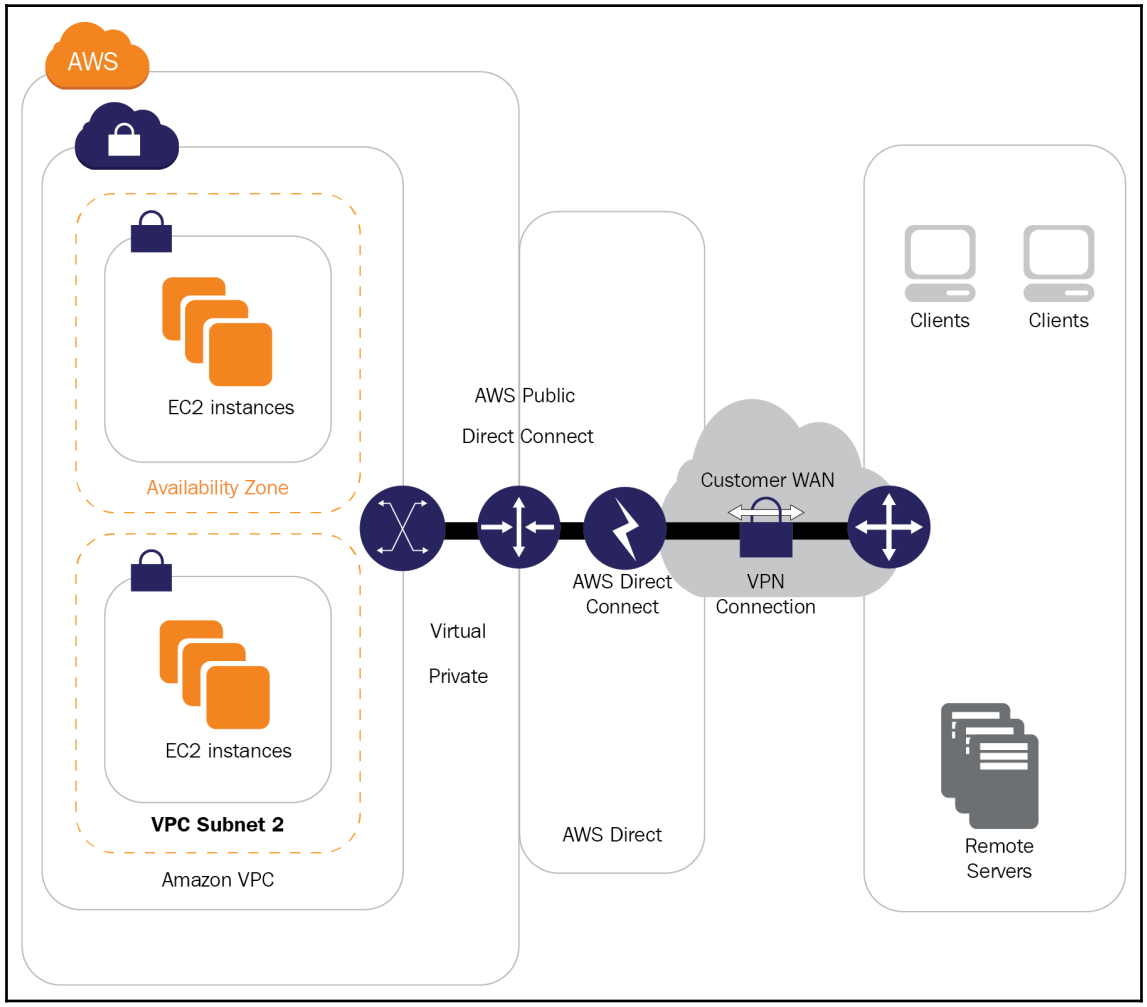
Interconnecting using Direct Connect and a VPN

Amazon Direct Connect can be combined with VGWs to provide a private IPsec connection between the AWS VPC and the customer network.

Direct Connect can be configured with two kinds of interfaces: a public interface and private interface. The public interface allows the user networks systems to access Amazon services that have a public endpoint, such as S3, for example. In this case though, when a public endpoint is being accessed, the connectivity is via Direct Connect and not over the internet. The private interface is used to access resources with private IPs, such as EC2 instances, for example.

This solution provides both the benefits of a private IPsec connection and Direct Connect.

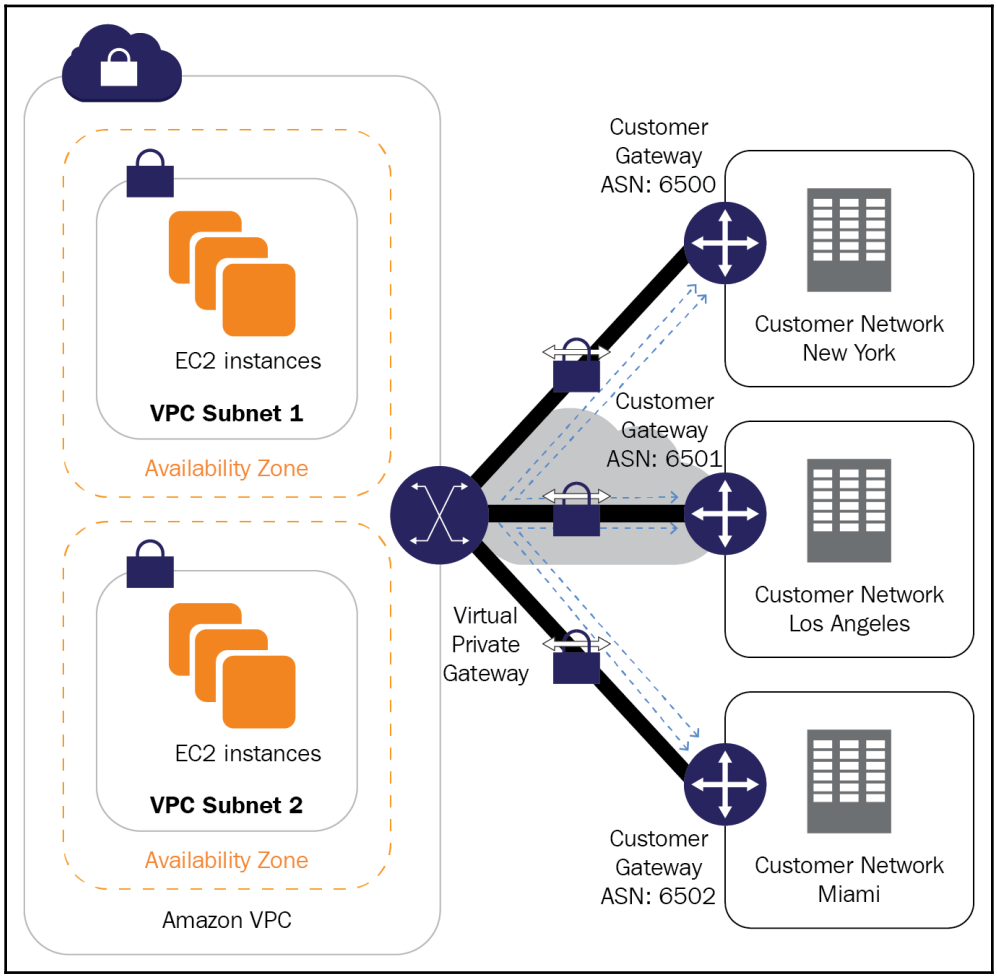
A sample network diagram of this configuration is given here:



AWS VPN CloudHub

Let's assume that the customer has multiple locations, and hence, multiple networks. AWS VPN CloudHub allows customers to create secure connections between multiple locations. AWS VPN CloudHub operates on a spoke-and-hub model. This can be used with or without a VPC. This interconnect can be used by customers who have multiple offices and would like to use a hub-and-spoke design, using their internet connections. These connections can be used as either a primary or secondary connection between various offices.

The next diagram gives you a clear idea of this hub-and-spoke design:



AWS VPN CloudHub uses the VGW with multiple user gateways, each gateway having a distinct BGP ASN. The user gateway advertises its BGP ASN using the VPN. This is received by the VGW and re-advertised over all the peer BGP connections. Once this is done, data can then be transferred from one site to another. We must keep in mind that each spoke must have a unique ASN, and there should be no overlapping IP address between any two sites. Additionally, each spoke connection can be used to send and receive data from the site to the VPC, as if a standard hardware VPN connection was used.

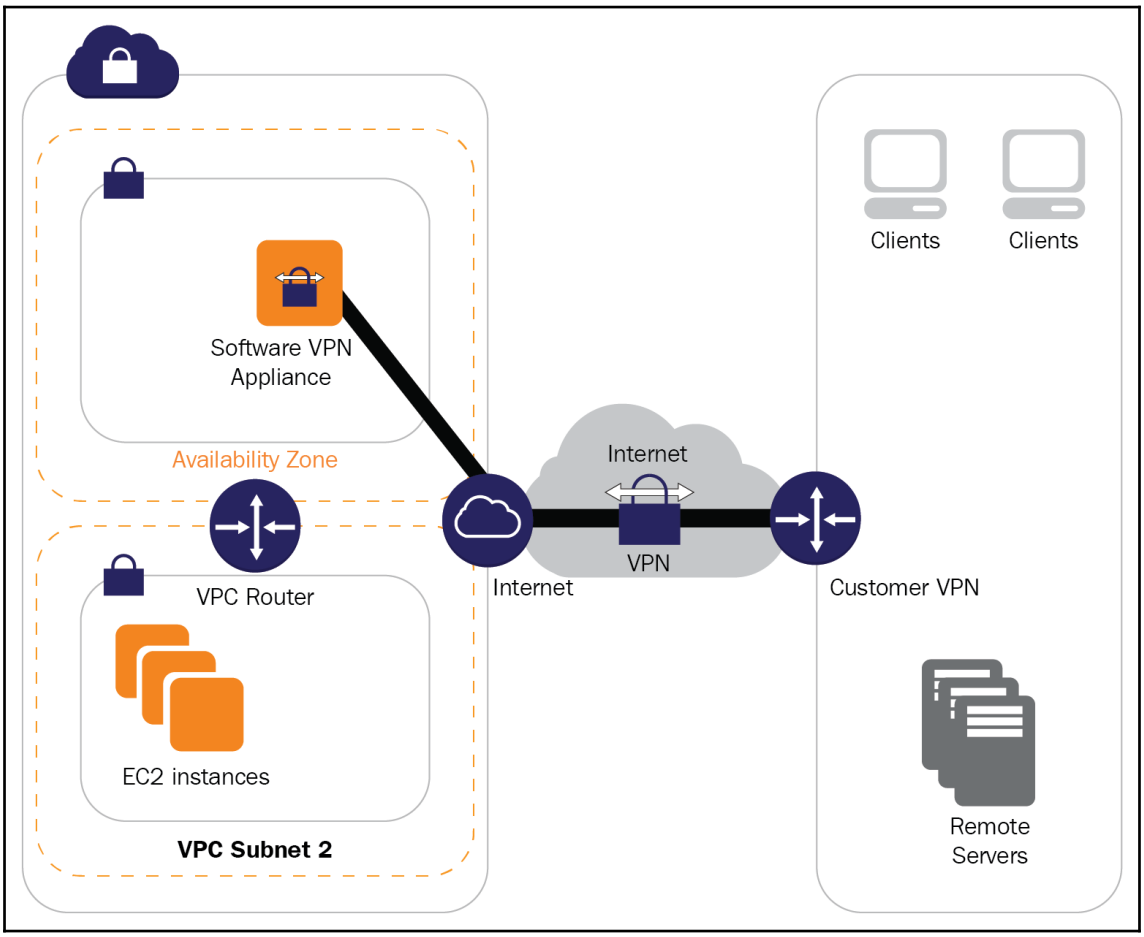
Software VPN

In the case of hardware VPN, we saw that Amazon manages one side of the VPN, the VGW. The user manages the other end of the connection, which is the user's VPN equipment. If the user wants to manage both sides of the connection, then a software VPN can be used for this purpose. The need for managing both sides of the connection may arise due to either compliance requirements or when the user is using gateway equipment that is not supported by Amazon.

The software VPN is basically a software appliance. This software appliance will run on an EC2 instance. The way to obtain such an appliance is to look into the Amazon Marketplace, select the appropriate **Amazon Machine Image** (**AMI**), that is, the software VPN appliance, and use the AMI to start an EC2 instance. Software VPN appliances are available from multiple vendors in the Amazon Marketplace. Vendors include Check Point, Astaro, Microsoft, and so on. Open source products such as OpenVPN are also available. It must be borne in mind that the user is responsible for managing the appliance. The user needs to take care of patching and upgrading.

In the case of using a single software VPN appliance, it can potentially become a single point of failure. It is the user's responsibility to ensure the high availability and fault tolerance of the software VPN appliance.

The following diagram gives the configuration of the connectivity when a software VPN is used:



Connecting without a VPN

If an EC2 instance is in a default VPC, or the EC2 instance is in a public subnet of a non-default VPC, then these can be accessed using SSH or RDP, provided the instances have a public IP attached to them.

In the case of Linux instances, we can SSH to the instances if we have a key pair that is associated with the instance. The appropriate ports must be opened in the Security Group and NACL. Similarly, if we are running Windows instances, we can access the instance using RDP. Here, we need the key pair to obtain the password for the Windows instance. Once the password is obtained, we can log into the instance, provided the security group and NACL are opened for RDP traffic.

In these cases, we are leveraging the internet gateway to access the systems. There is no VGW involved, and there is no VPN connection involved either.

Interconnecting between Amazon VPCs

In this section, we will look at the various methods of connecting multiple VPCs. A customer can have multiple VPCs for various reasons: security, billing, needing cloud infrastructure in more than one region, and so on.

When using multiple VPCs and planning to connect them to one another, the users must ensure that there are no overlapping IPs between any of the VPCs. Hence, it is important for the user to plan the IP allocation right at the beginning, and assign a contiguous CIDR range for each VPC.

We can connect VPCs using the following techniques:

- VPC peering
- Hardware VPN
- Software VPN
- Software-to-hardware VPN
- Direct Connect

Let's look at each of these in detail

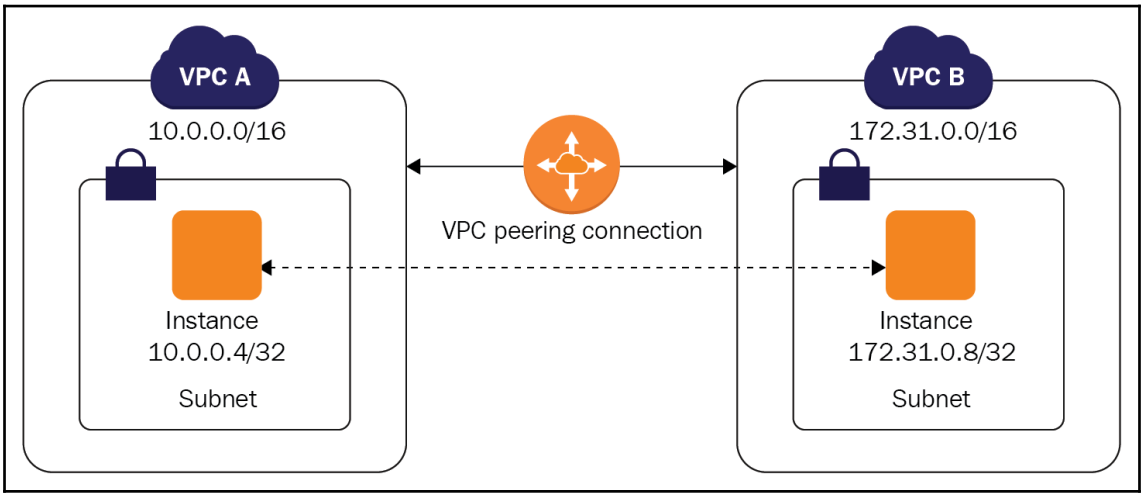
VPC peering

The easiest and simplest way to connect two VPCs is by VPC peering. The peered VPCs can be in the same region or may be in different regions. For peering to happen, there should be no overlapping IPs between the two VPCs.

A peering connection allows the instances in the peered VPCs to communicate with each other using private IP addresses. The peering can be between VPCs of the same account or of different accounts.

AWS does not use any extra resources, such as a VGW, to create the peering connection. There is no dependence on any hardware for creating this connection. We make changes in the routing table to take advantage of peering. Protection can be provided by security groups and network access control lists.

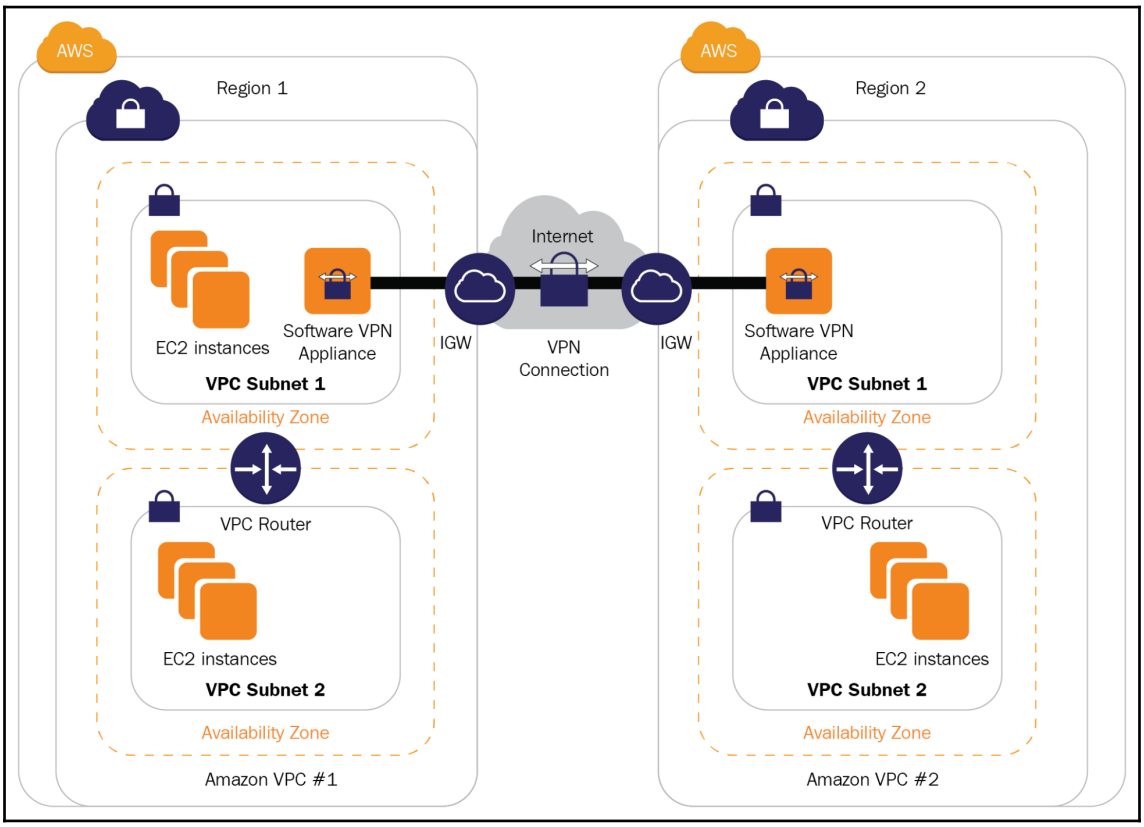
VPC Peering is not transitive. If VPC A is connected to VPC B and VPC B is connected to VPC C, it doesn't mean that instances in VPC A can communicate to instances in VPC C directly. This is because of the non-transitive nature of VPC Peering. In this case, if communication must happen between VPC A and VPC B, then they must be directly peered:



Software VPN

If the user does not want peering and wants to control both ends of the connection by using their preferred VPN software, then using a software VPN is an option that can be explored. (This option was also used earlier to connect VPCs that were in different regions, as VPC peering was initially not available across regions.)

The following diagram gives you the configuration for VPN software based on a VPC-to-VPC connection:

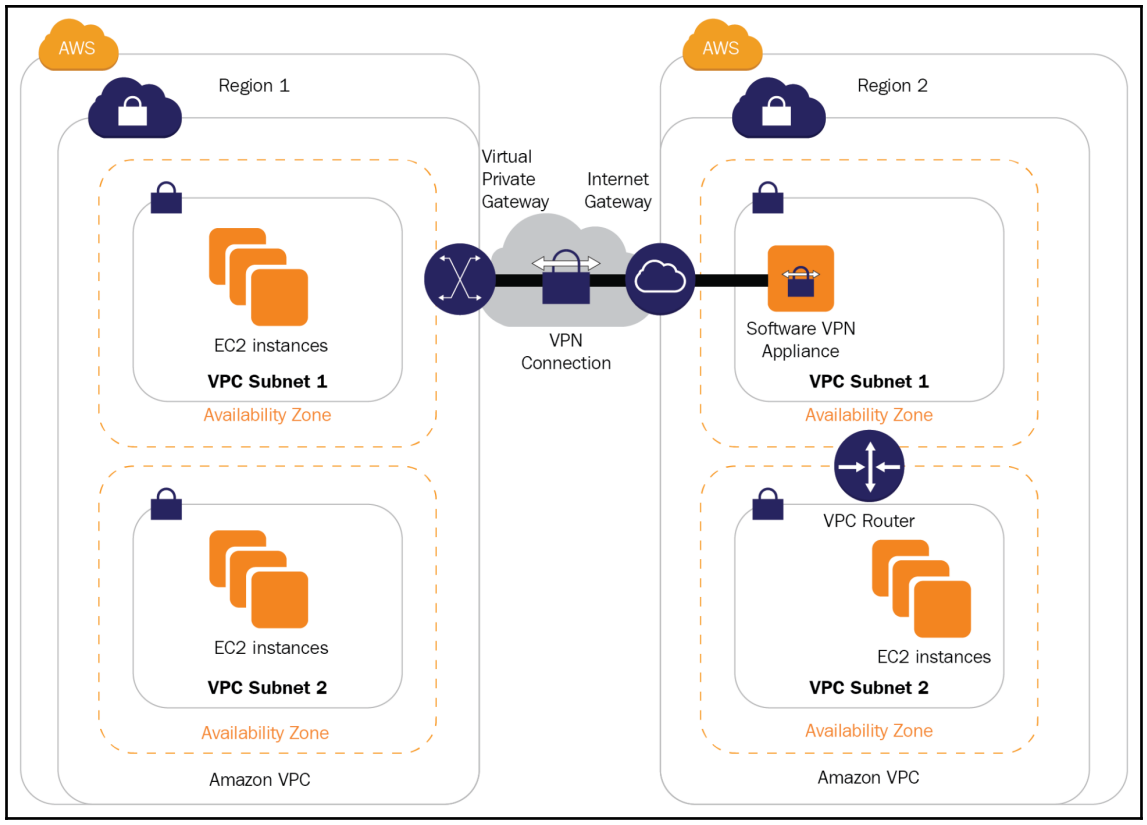


As can be seen from the diagram, we have a software-based VPN appliance in both the VPCs. Since the connection has to happen over the internet, the appliance is placed in the public subnet and leverages the internet gateway for connectivity purposes.

As stated in one of the earlier subsections, the user is free to use any software VPN appliance of their choice. These can be obtained from the Amazon Marketplace. Managing the appliance, and ensuring high availability and fault tolerance, are the responsibility of the user.

Software-to-hardware VPN

In this architecture pattern, we have the software VPN appliance in one of the VPCs, and in the other VPC, we leverage the VGW. This architecture is shown here:



This pattern was earlier suggested by AWS when there was a need to connect VPCs that were in different regions. That was because VPC peering was restricted to VPCs within the same region. With VPC peering now available across regions, this particular pattern will be useful only if there is a need to use a software VPN appliance in one of the VPCs for compliance reasons. Otherwise, VPC peering is the preferred method.

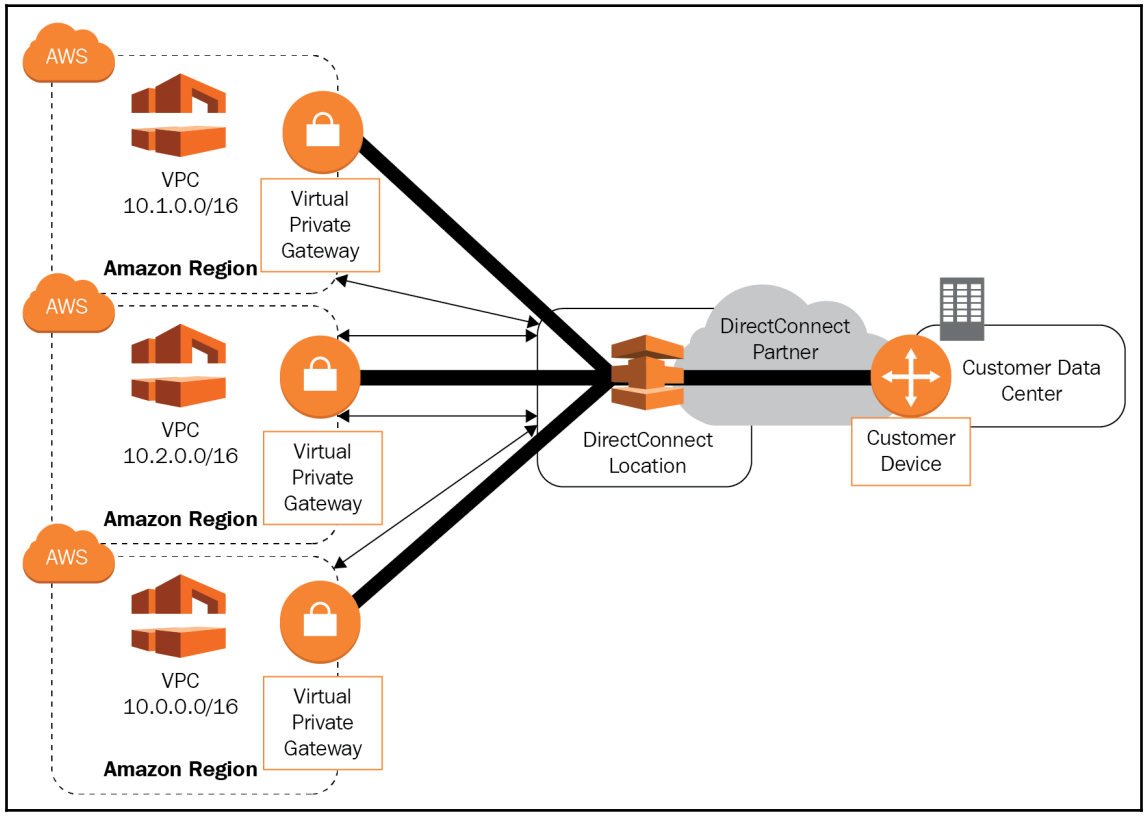
Hardware VPN

The hardware VPN architecture leverages two VPN connections. Each connection is from the VPC to the customer data center.

This too was one of the suggested patterns when inter-region VPC peering was not available. With inter-region VPC peering now available, VPC peering is a better option than this.

Direct Connect

AWS **Direct Connect** customers can leverage Direct Connect for VPC-to-VPC communication. The user can divide the direct physical connection into various logical connections. These logical connections can then be used to route traffic between various VPCs. The following diagram provides the details:



As can be seen from the diagram, the WAN backbone is leveraged for the VPC connection.

Interconnecting to AWS services from inside a VPC

In the previous sections, we have seen how to connect an external data center to a VPC, as well as how to connect two VPCs together. In this section, we will see how we can access some of the AWS services from within an Amazon VPC. This is done using the concept of endpoints and Endpoint Services.

There are two kinds of VPC endpoints that allow instances in a VPC to access AWS services without needing an NAT gateway or an internet gateway. The kinds of VPC endpoints are as follows:

- **The gateway endpoint**
- **The interface endpoint**

VPC endpoints – the gateway endpoint

As the name indicates, a gateway endpoint is a gateway. This gateway enables the instances within the VPC to connect to some AWS services. The AWS services that can currently be accessed using a gateway endpoint are S3 and DynamoDB.

S3 and DynamoDB have an internet end point. Before the gateway was introduced, the only ways to reach S3 or DynamoDB from inside a VPC was to either leverage the internet gateway if the instance was in the public subnet, or through a NAT gateway if the instance was in a private subnet. With the introduction of the gateway endpoint, instances inside a VPC can reach S3 and DynamoDB using the gateway endpoint. There is no need for an internet gateway or an NAT Gateway to access S3 and DynamoDB from within a non-default VPC.

To use the gateway endpoint, the user needs to update the routing table and point to the gateway endpoint as the target for routes to S3 and DynamoDB.

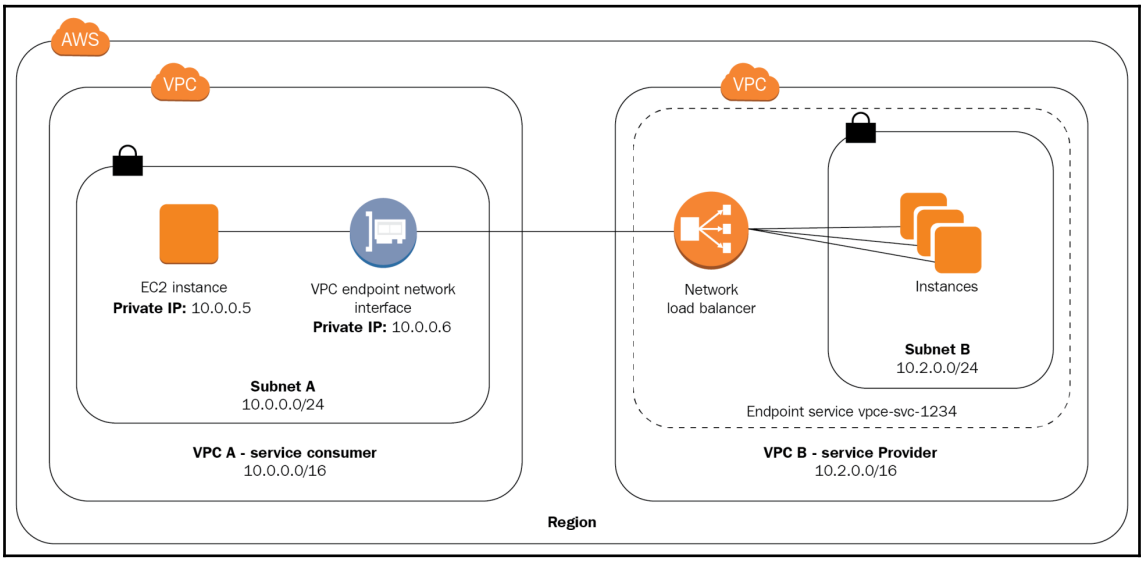
VPC endpoints – the interface endpoint

Before we understand the Interface endpoint, we need to understand **AWS PrivateLink**. AWS PrivateLink is a service provided by AWS and using which we can connect services across VPCs, across accounts, and connect to services from on-premises. In all these cases, the traffic is only within the Amazon network, and the traffic does not flow out to the internet. This ensures our traffic is secure. If AWS PrivateLink is used, we eliminate the need for using an internet gateway, NAT gateway, and Elastic IPs.

Interface endpoints are VPC endpoints that allow a user to connect to services that are powered by AWS PrivateLink. The types of services could be as follows: AWS services, services hosted by other AWS accounts, and supported AWS Marketplace services. The following services are currently supported by interface endpoints:

- Amazon API Gateway
- Amazon CloudWatch
- Amazon CloudWatch Events
- Amazon CloudWatch Logs
- AWS CodeBuild
- Amazon EC2 API
- Elastic Load-Balancing API
- AWS Key Management Service
- Amazon Kinesis Data Streams
- Amazon SageMaker Runtime
- AWS Secrets Manager
- AWS Service Catalog
- Amazon SNS
- AWS Systems Manager
- Endpoint services hosted by other AWS accounts
- Supported AWS Marketplace partner services

Using an interface endpoint is different than using a gateway endpoint. In a gateway endpoint, we create an entry in the routing table, enabling the appropriate traffic to be routed via the gateway. In the case of an Interface endpoint, we create an interface endpoint for a service in the VPC. This, in turn, creates an **Elastic Network Interface** (**ENI**) in the subnet. This ENI has a private IP address associated with it. Through this private IP address, we can access the service for which the interface was created. This is shown in the following diagram:



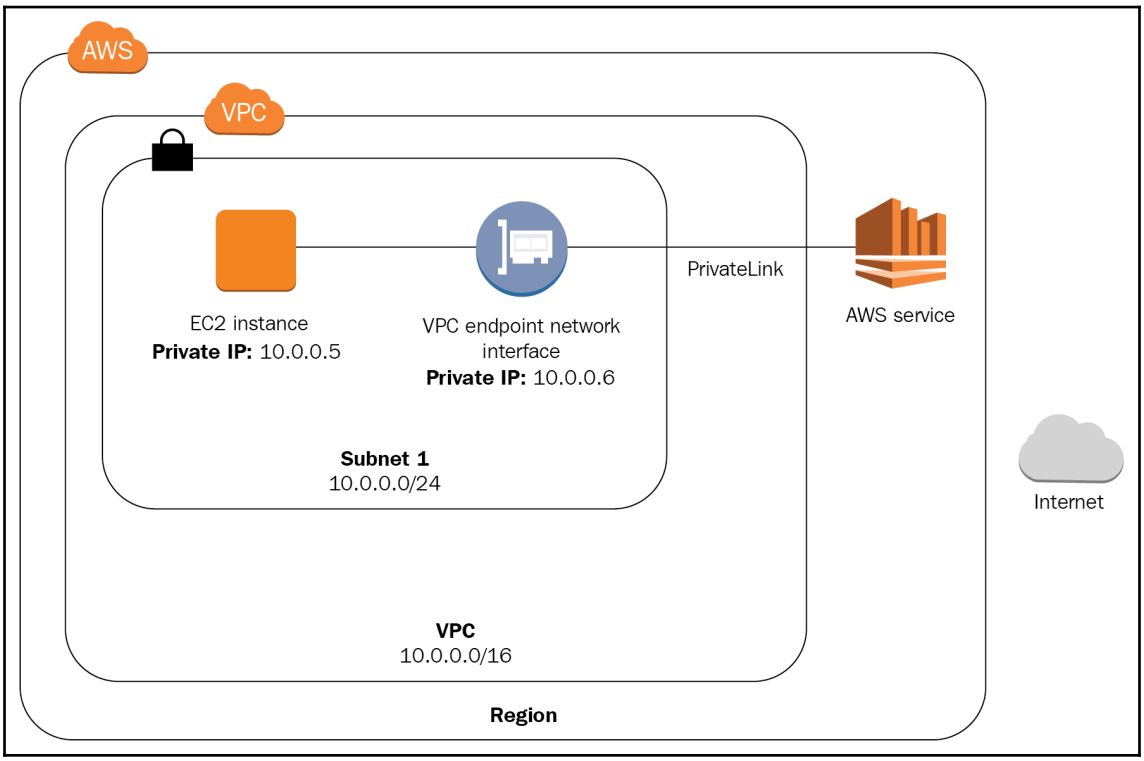
Endpoint services

Users can also enable their services for users in other VPCs using PrivateLink and Endpoint Services. Once the service is enabled as an Endpoint service, other VPC users can access this service using an Interface endpoint.

To allow our services to be accessed via an interface endpoint, we need to take the following steps:

- Create a Network Load Balancer and enable it for all the availability zones in which you want the service to be available.
- Attach your instance hosting the application to the Network Load Balancer.
- In the VPC Endpoint Service configuration, specify the Network Load Balancer.

Once this is done, others can access this service using the interface endpoint. This is shown in the following diagram:



Transit Gateway

Assume you have multiple VPCs in your AWS infrastructure. You also have a requirement that these VPCs are peered. Furthermore, you want to connect to these VPCs from your data center as well. All these can be achieved by using peering connections and VPN connections, but the complexity increases when the number of VPCs become large. This is because of the following reasons:

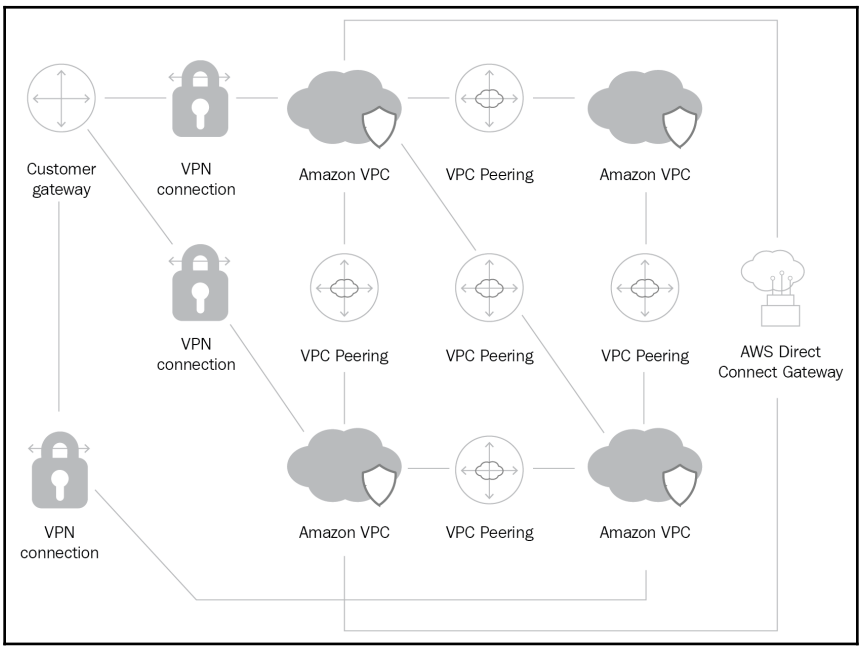
1. The Peering connections are non transitive.
2. We have to make one VPN connection per VPC, and we want to connect to VPCs from our data center.

Let's assume that you have four VPCs, all of which you want peered. Furthermore, you want to connect to all VPCs from your data center. This will mean that you have to create six peering connections and four VPN connections. That means you now have to manage 10 connections. We need to understand that these are point-to-point connections and we cannot manage all these connections from a central point. The complexity increases when the number of VPCs increases and you start connecting VPCs across various accounts. To manage all the connections from a central point, AWS has introduced the transit gateway.

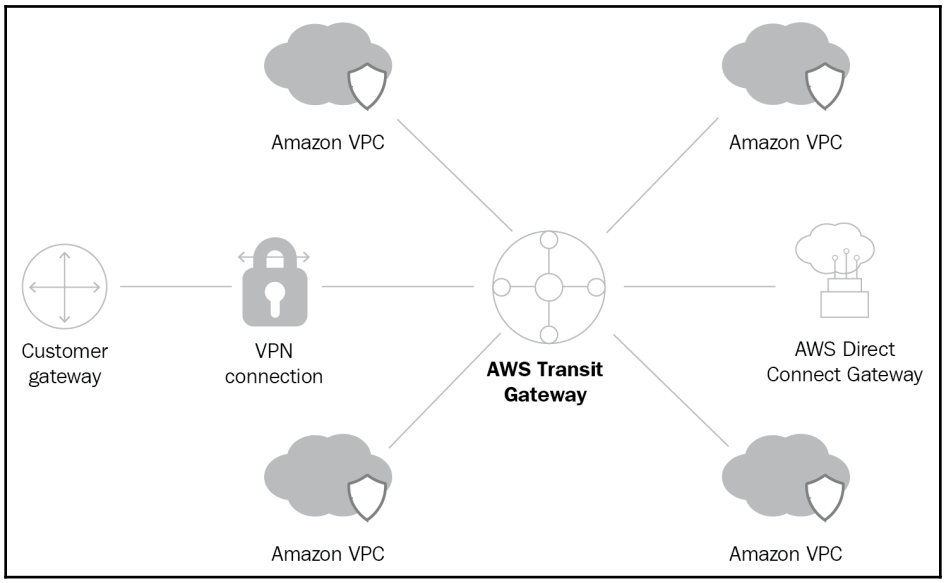
When you use a transit gateway, you can connect your VPC or the data center to the transit gateway. That is the only connection needed. The transit gateway is the hub, and the VPCs/data centers are the spokes. So, now we need to manage only one connection per VPC or data center. The routing between various networks can now be achieved from the transit gateway. Once a VPC is connected to the transit gateway, it automatically becomes available to all other networks that are connected to the transit gateway. Using a transit gateway allows us to scale our network easily.

The following two diagrams give us an idea of how a transit gateway simplifies the complex connectivity needs. The first diagram shows how we achieve connectivity without a transit gateway. The second diagram shows how connectivity is achieved with a transit gateway.

The following shows VPCs and connectivity with no transit gateway:



The following shows VPCs and connectivity with a transit gateway:



Summary

In this chapter, we have learned about the various network interconnects to AWS. We have seen how we can build a VPC and how we can connect to this VPC using VPN hardware, VPN software, or Direct Connect. We also saw how VPC peering was done, and we learned about the VPC endpoint, VPC endpoint services, and the transit gateway.

In the next chapter, we will talk about Network Interconnect, for on-premises and hosted cloud solutions.

9
Interconnecting Different Cloud Solutions

In this chapter, you will get some more insight into how to interconnect different cloud providers. Therefore, you will use the knowledge that you gained in the other chapters. We will discuss how to interconnect multi-cloud services in the network layer and the identity layer, so that you can build a unified, best of breed cloud solution. I will also give you a little insight into multi-cloud deployment management, which is often called **Infrastructure as Code** (**IaC**).

The following topics will be covered in this chapter:

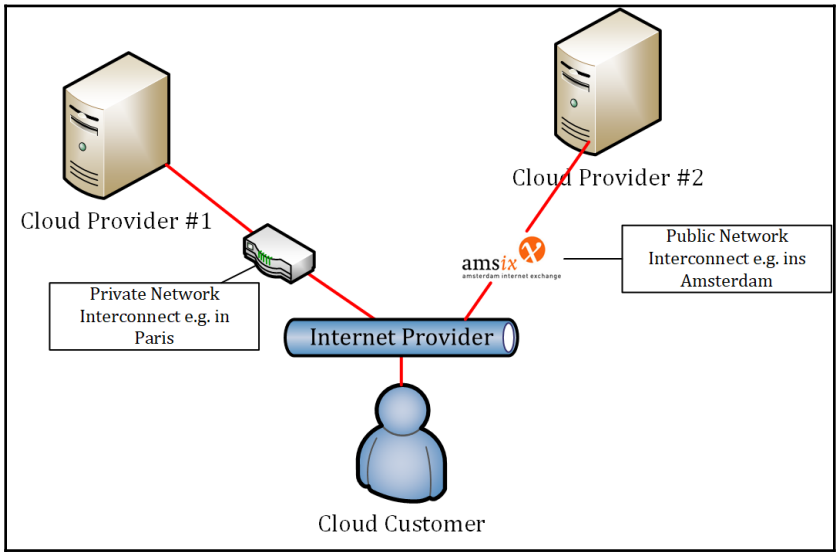
- Interconnecting cloud solutions in the network layer
- Interconnecting cloud solutions in the identity layer
- Multi-cloud infrastructure templates and management

Interconnecting cloud solutions in the network layer

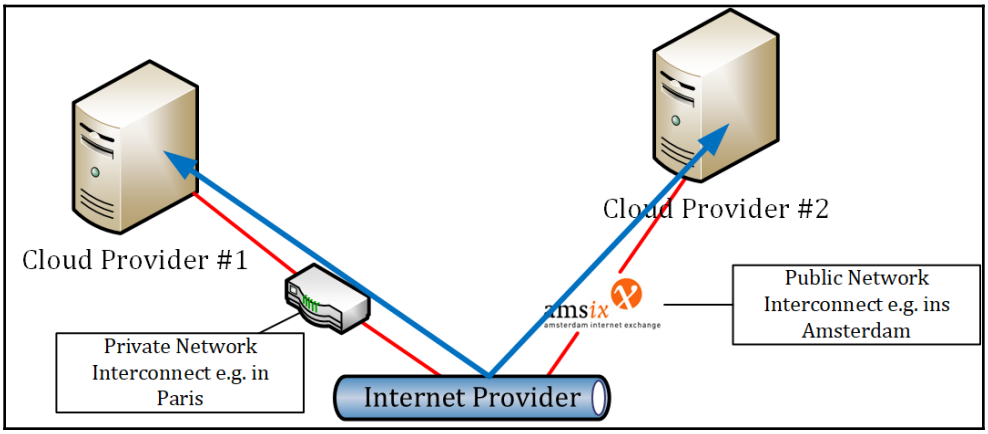
Before trying to interconnect cloud providers, you need to know that most cloud providers, including AWS, Salesforce, and Microsoft, do not peer with one another. They are basically competitors in the same market, and because their internal policies are not public to customers, they don't provide any direct connections to each other. Therefore, you will need to optimize the interconnection between all of the providers.

Interconnecting via the internet

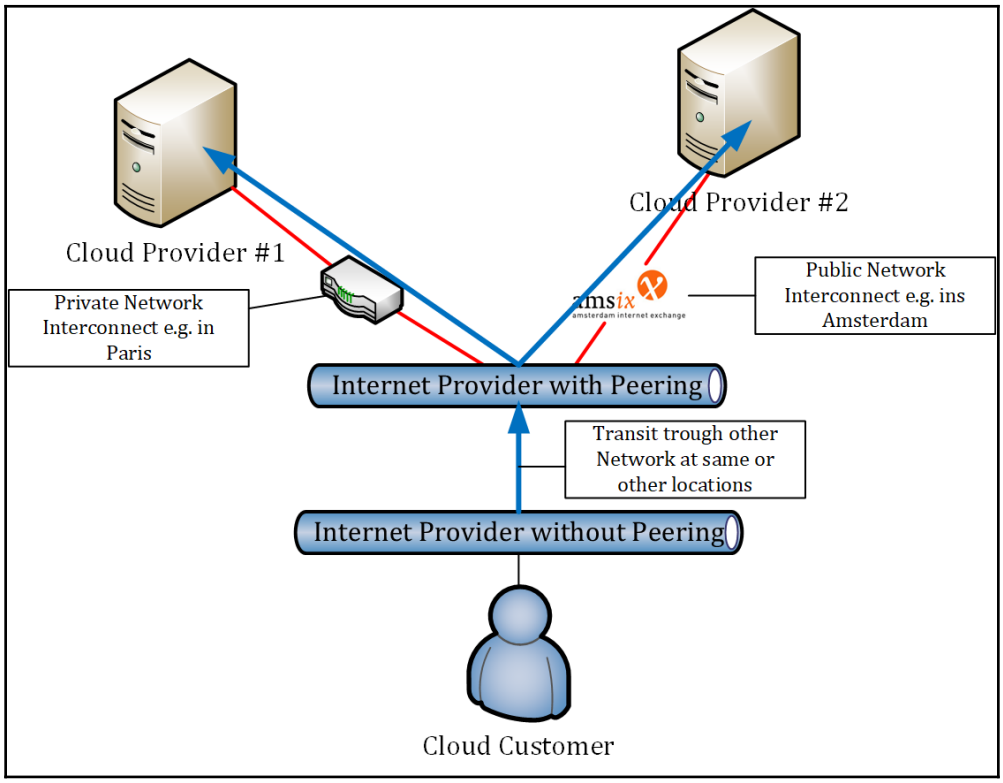
When using regular internet access, the interconnections are mostly done with your internet provider, which has a direct peer to the cloud provider. The following diagram illustrates such an interconnection:



In this case, the routing from one cloud provider to another looks as follows:



The same happens when you use different internet providers. Both of the internet providers either peer with each other or need a transit network to reach each other. The following diagram shows you a schematic view of a direct peered provider network to network interconnect and one with transit:





The important thing, when using the internet to connect cloud solutions (either on-premises or hosted), is to choose providers with good interconnections to each other and to the cloud provider of your choice. A well peering, low latency, and widely distributed peering infrastructure at a network provider is crucial for a good cloud experience and effective usage for cloud customers. The basic principle here is: latency kills the cloud.

Virtual Private Networks

When you are using **Virtual Private Networks** (**VPNs**) to connect to cloud solutions via the internet, you connect your cloud services to your network like additional VPN sites. For most IaaS scenarios, that is totally fine, but when you are using PaaS or SaaS solutions, they often have no VPN endpoints, and the security comes from encryption and identity and access management.

Cloud services are built on **zero trust networks**, which means that the security does not come from an IP or network security; it comes from protocols and identity. To learn more about zero trust architectures, I highly recommend the article at `https://www.paloaltonetworks.com/cyberpedia/what-is-a-zero-trust-architecture`.

Nevertheless, you are able to connect cloud providers with each other via VPNs. For most cloud providers, you would use standard VPN protocols, like IPSec.

The following article shows the setup for connecting AWS to Azure: `https://blogs.technet.microsoft.com/canitpro/2016/01/11/step-by-step-connect-your-aws-and-azure-environments-with-a-vpn-tunnel/`.

Interconnecting via private network interconnects

When interconnecting cloud solutions and providers, you will often want to integrate and interconnect them via your private, wide-area network. This has the same challenges as internet-based interconnection; for example, there is no direct peering between the cloud providers. You cannot configure an Azure ExpressRoute circuit to connect to AWS Direct Connect, and vice versa. In addition to that, your network providers need to have peering with the cloud provider in the right location, and you need to be able to route between the providers. You need to create the interconnection by yourself, or along with your network provider.

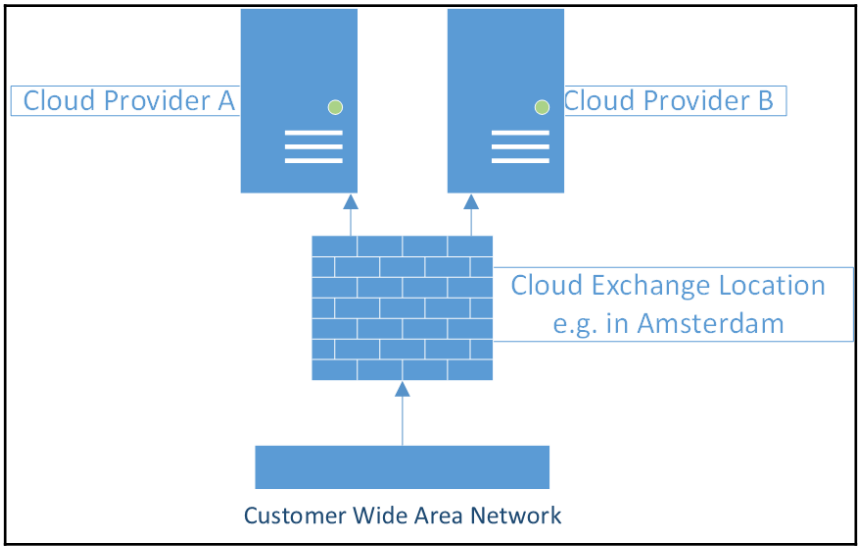
The two most common use cases in this field are as follows:

- In the first one, you have a Layer 3 network, like with MPLS, and you can directly interconnect to the cloud providers; the routing is done by you or your provider. You only need to choose the right interconnection locations, to keep the locations close together and optimize data transfer.

- The second option is to own a Layer 2 network, where routing is done by you, and the network is basically hardwired from location to location. Here, you need an additional colocation, like a colocation at a cloud exchange hub, to exchange data.

There are providers on the market, such as Megaport, that offer **virtual cloud routers**, enabling customers to build a direct interconnection by themselves, without any need for colocations or hardware. An implementation of Megaport is shown at `https://www.megaport.com/megaport-cloud-router/`.

For both scenarios, you will need an interconnection point. The architectural diagram might look as follows:





I personally recommend getting all of the cloud providers, your network provider, and your teams at one table, in order to discuss a solution for your wide-area network.

Interconnecting cloud solutions in the identity layer

When building a multi-cloud and multi-identity, single sign-on scenario for all solutions becomes more and more necessary. No user wants to sign in to cloud solutions and applications with twenty different logins, and—also very important—you, as an administrator, don't want to maintain twenty different identity and access systems and user databases.

Normally, you will want a single-identity system, where you can manage access to all of the cloud platforms and on-premises resources for every user, and may be able to add multi-factor authentication or fulfill European Union **General Data Protection Regulation** (**GDPR**) requirements.

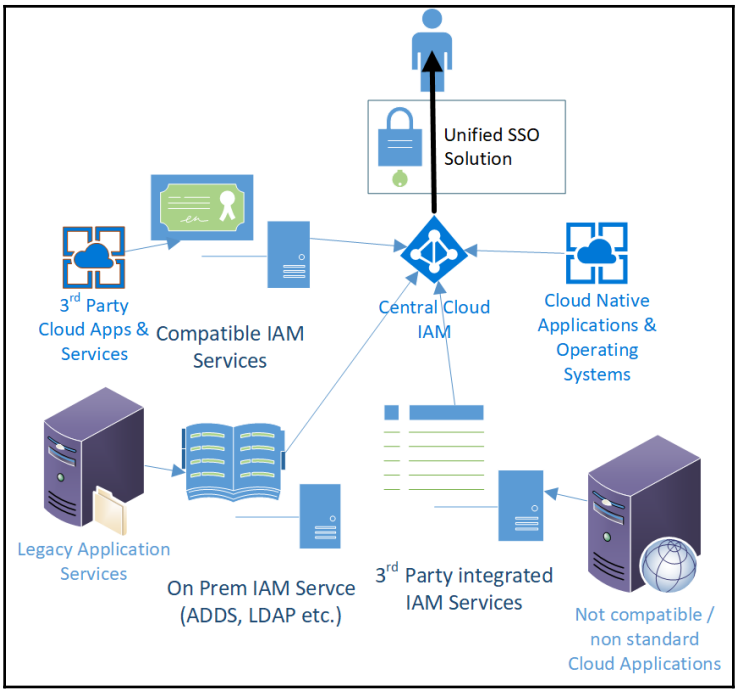
Generally, every cloud provider has its own identity solutions. For Microsoft, you have Azure Active Directory in the cloud and Windows Server Active Directory Domains and Federation Server on premises; for Google, it is Google Cloud IAM, and so on.

You need to get these solutions together when you are thinking about a central cloud identity solution. The first thing is to streamline it to your basic needs, like the following:

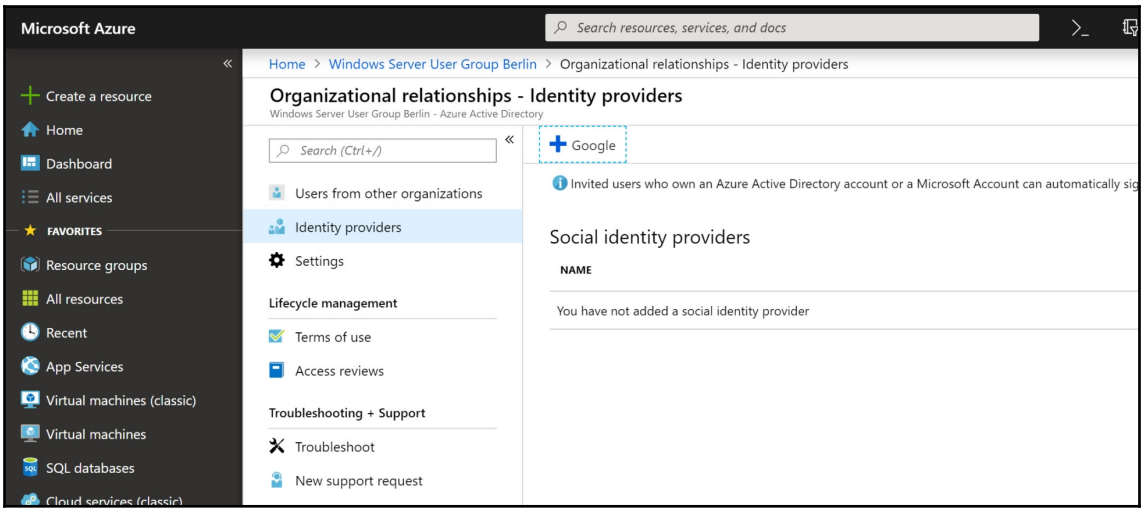
- Single sign-on
- Multi-factor authentication
- SAML, XAML, oAuth
- Integration of Office 365, Windows, LDAP, or Facebook

Now, check the solutions that are available on the market. To be honest, you will not find a solution that will fit all of the requirements one hundred percent, but you can find a solution that fits ninety percent and has an API with which you can connect third-party products, in order to get the last ten percent.

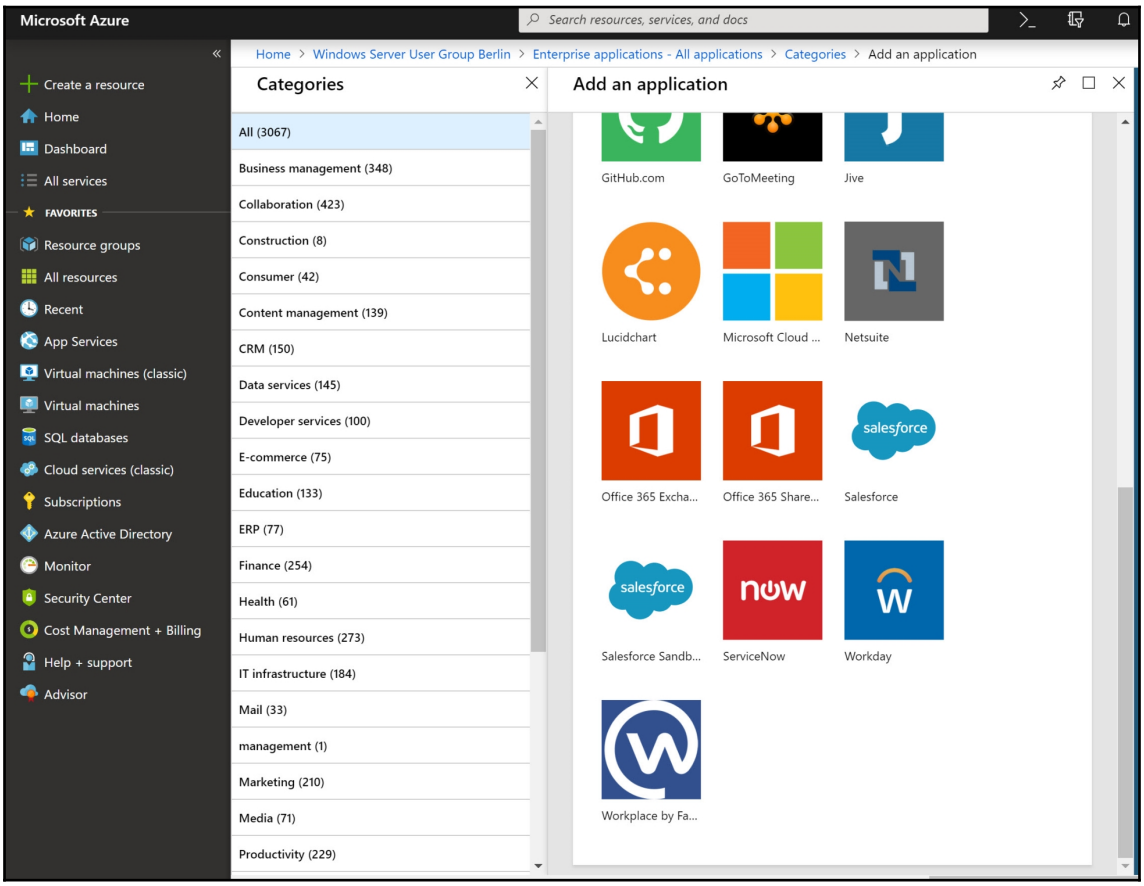
Let me illustrate this as a basic diagram:



Azure Active Directory is a good example of such an identity solution. Azure Active Directory can integrate different identity providers, like Google, as shown in the following screenshot:



You can also add additional enterprise applications, like AWS or Salesforce, as shown in the following screenshot:





An integration of Amazon is explained in the following tutorial: `https://docs.microsoft.com/en-us/azure/active-directory/saas-apps/amazon-web-service-tutorial`.

With Azure AD B2C, you can also add different identity providers, like Facebook.

With that, you can fulfill nearly every identity integration possible. You will achieve a very flexible and scalable solution, where you can add and remove cloud and application services, and will also be able to replace and switch identity providers, if necessary.



A strange scenario that I had in the past was as follows. A marketing department wanted to have centralized management for their Twitter and Facebook accounts, without sharing passwords with trainees or temps. They wanted to sign in with their Active Directory domain accounts and manage the access through a centralized IAM system. We built such a solution on top of Azure Active Directory. Azure AD was federated with Windows Server Active Directory, and we integrated Facebook, Twitter, and so on, through APIs. That gave us single sign-on for those platforms and central management with Azure AD.

Azure is only one example of those IAM systems. As we already mentioned, there is also Google, as well as companies like JumpCloud. JumpCloud is specialized in multi-cloud identity management, with on-premises components and different operating systems, like Linux or macOS.

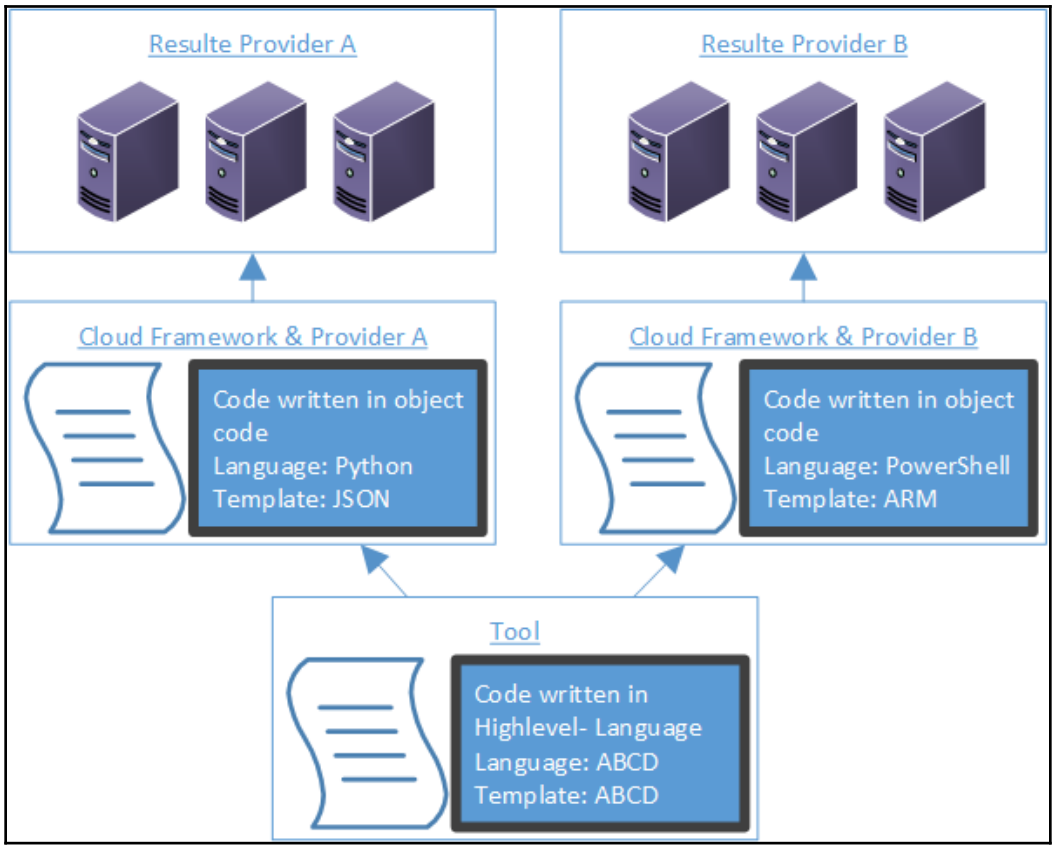
Multi-cloud templates and management

When you are working with a multi-cloud environment, the most challenging point is to keep your code consistent across all of the cloud platforms. This means that you can replicate configurations and environment deployments without maintaining many code files and different development engines and languages.

So, you need to implement a unique platform to maintain the code. There are tools on the market that can help you with that. Some of those tools are Terraform and SaltStack.

When working with or searching for those tools, you will often hear the buzzword Software as Code. Software as Code basically means that the code is written in a high-level language and is readable by people, but not systems or clouds themselves. The source code script must be converted into object code, scripts, or machine language, executable in the specific cloud in which it can be deployed.

The following diagram shows a schematic workflow of such tools:



Because we cannot cover how to use those tools within this book, we will provide you with some links to entry-level tutorials and training for the most common tools, in the following table:

Tool	Link
Terraform	`https://www.terraform.io/intro/vs/cloudformation.html`
SaltStack	`https://docs.saltstack.com/en/getstarted/`
Chef	`https://www.chef.io/chef/get-chef/`
Ansible	`https://docs.ansible.com/ansible/latest/user_guide/intro_getting_started.html`

Summary

Having finished this chapter, you should now have an idea of the challenges involved in interconnecting cloud solutions in the network, identity, and API layers. You have learned how to solve those challenges, and have seen the options that are available to support your needs.

In the next chapters, we will work on a number of concrete implementation scenarios that you may face within the field.

10
Identity and Access Management in Microsoft Azure

In this chapter, we will discuss **Identity and Access Management** (**IAM**), including access to resources within a tenant environment. You will learn how to handle identity on-premises and in the cloud. The key concepts of the two are basically the same, but the difference is in the details, as always. You will learn how to design a model for working within hybrid cloud environments, in order to create a relationship between both of them.

The following topics will be covered in this chapter:

- Directory services, in general
- Designing a hybrid cloud identity management solution
- Designing a hybrid cloud access management solution

Directory services in general

A **directory service** maps the names of the resources in the network to their corresponding addresses. It provides a central location for all infrastructure information, like the location, user accounts, volumes, files, devices, groups, and a lot of other objects. The directory service provides the central database for all of this information, and it is the critical resource for the network operating system; if it fails, no directory resources can be found anymore. Therefore, this service should be designed for high redundancy. The name of the directory is mapped to a name resolution service, to easily identify the resources and their attributes. The directory service could also provide permissions and authorization and authentication information.

If we take a look into IT history, every networking environment working with central user management is based on a directory service. In the past, we had **Novell Directory Services** (**NDS**), and even **NT-Domain Services** (**NTDS**); since Windows Server 2000, the Microsoft on-premises directory service has been called **Active Directory** (**AD**). The cloud-based version is named **Azure Active Directory** (**AAD**).

As most of you already know how Active Directory works, I will only provide a short summary, to remind you of the most important topics.

Active Directory basics

AD is a hierarchical directory service that allows for the definition of the following:

- **Root domains**: A **forest domain** is the root (and the first Active Directory domain) for all other existing domains. It defines the name (based on DNS domain naming, like `company1.com`). It is possible to have different DNS domain names in one root domain (like `company2.org`).
- **Child domains**: A child domain is defined as each sub-domain of an existing root domain or other child domain (such as `europe.company1.com` or `germany.europe.company.com`). This provides flexible but complex domain designs. The best practice for a domain design is to always keep things simple, and to not go deeper than two to three hierarchies.
- **Organizational units (OU)**: An organizational unit is the best way to organize complexity in a flat directory design, as it provides a container for all objects (users, computers, or groups), and even supports OU nesting. This means that one OU can be placed underneath another one.

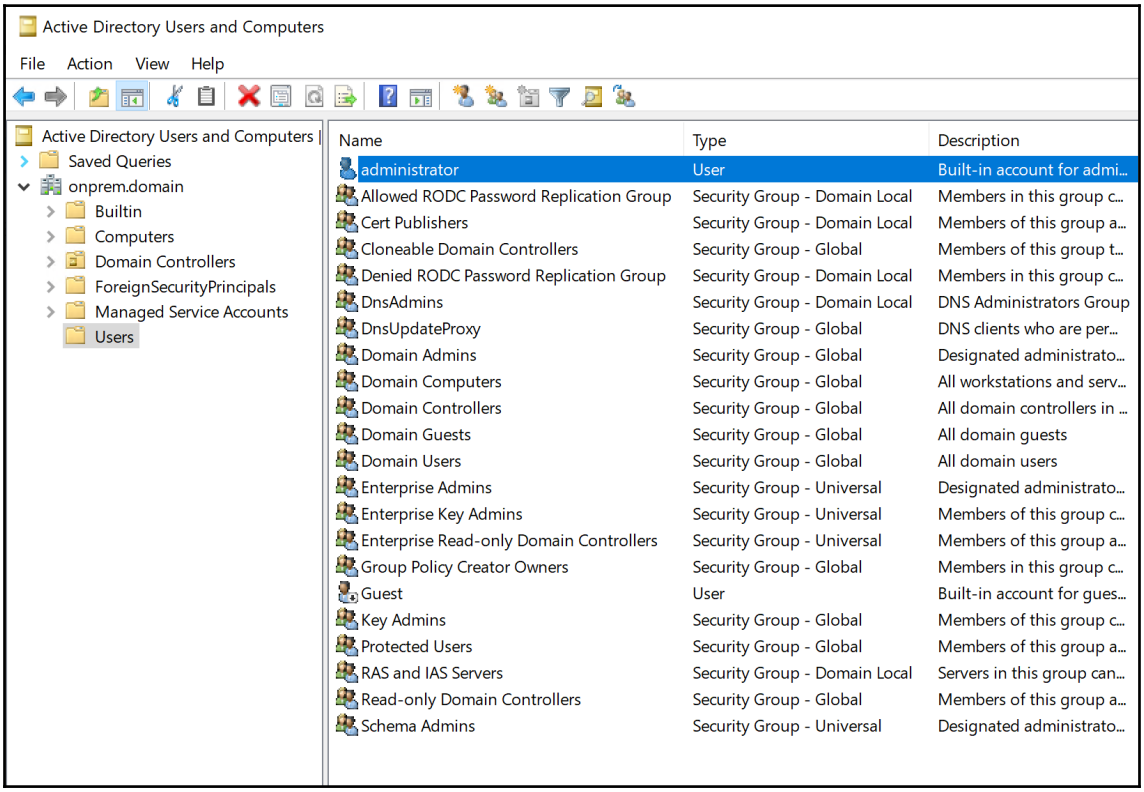


To keep it simple, the best practice for an on-premises directory service design is to have a minimum number of domains and to set up hierarchies based on OUs.

You can set up domains while setting up domain controllers, which is a part of adding the Active Directory Services role to a Windows server. As AD relies on DNS, you would either set the appropriate DNS server, or create it while deploying your AD domain itself.

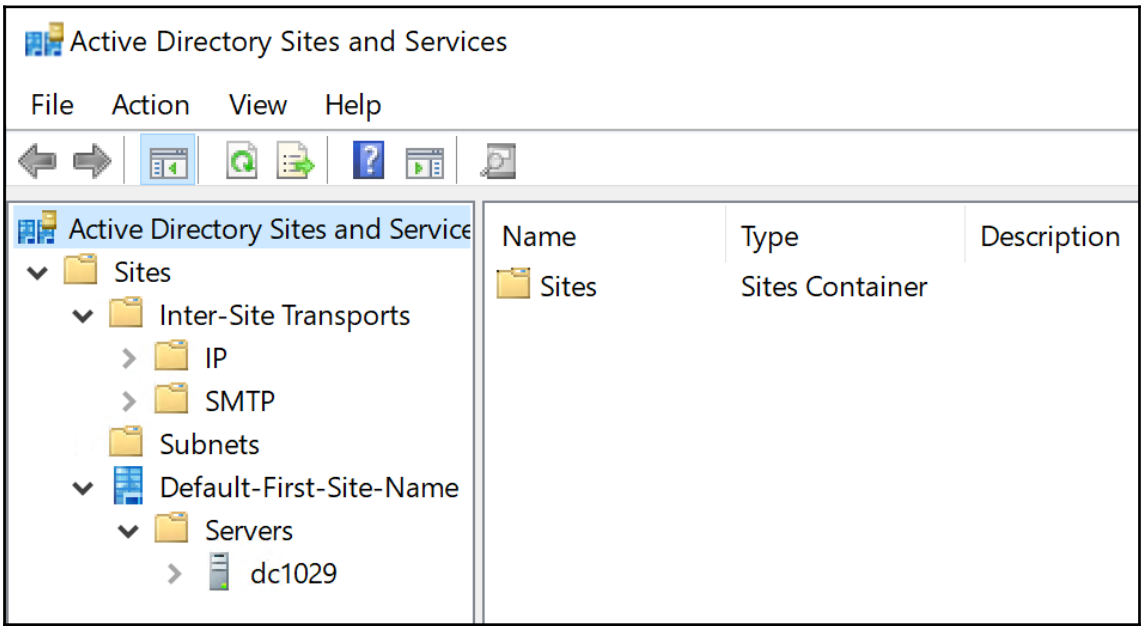
Azure Active Directory basics

Having successfully deployed a first (or an additional) domain controller, you will find the corresponding administration tools on the server. If you do not want to administer Active Directory directly on a DC, you can install them on an administrative workstation. In this case, you will need to install the **Remote Server Administration Tools** (**RSAT**):



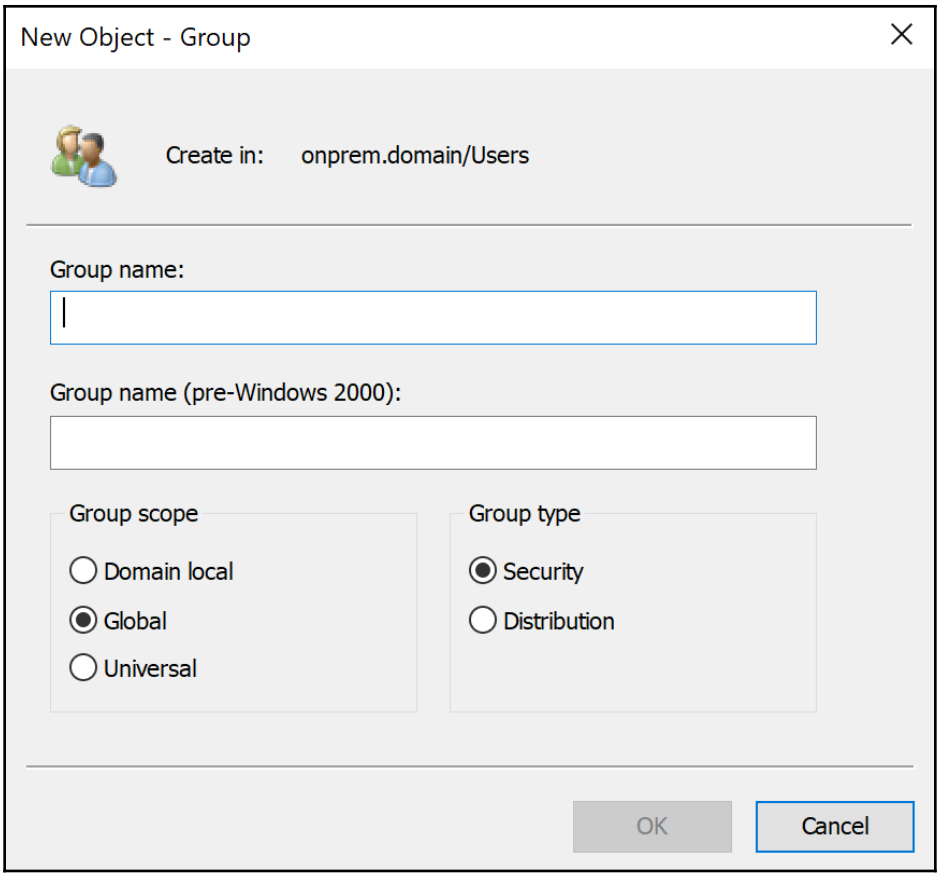
The management snap-in **Active Directory Users and Computers** provides a UI-based administration point for user and computer accounts, or even groups.

A user account is based on two unique identifiers: **Global Unique Identifier** (**GUID**) and **Security Identifier** (**SID**). The GUID is unique in a complete Active Directory Forest, and the SID is unique in a domain:



Active Directory Sites and Services is the management point for configuring route-based synchronization between IP subnets and their corresponding domain members. You can even define the protocol for synchronization (which can be either IP or SMTP).

If you have existing user accounts and need to organize them (for example, to enable access to a file or folder), you should create groups and add the users to those groups to provide access, as shown in the following screenshot:



As you can see, there are different options when creating a group: the **Group scope** and the **Group type**.

A group scope can be the following:

- **Domain local**: A **domain local group** is only accessible from within a local domain, and not outside, in the tree or the forest. Members of a domain local group can be users or other domain local groups from the same domain. Domain local groups are saved on each domain controller in a forest.

- **Global**: A **global group** is available outside of each domain, to provide cross-domain access. Members of a global group can be users, domain local groups, or other global groups, from the same or other domains in that AD forest.
- **Universal**: A **universal group** is available outside of each domain, and is being saved on each **Global Catalog** (**GC**) server in a domain. A GC can answer directly to access controls without asking other domain servers.

A group type can be the following:

- **Security**: A **security group** can be used to allow or disallow access to resources, and can be email enabled, too.
- **Distribution**: A **distribution group** can only be used for email delivery.



The principle to follow for allowing access to resources is defined as $U \rightarrow AG \rightarrow DL \rightarrow P$.

This means that a user should be added to a global group; the global group(s) can then be added to domain local groups, which are deriving the access control to resources.

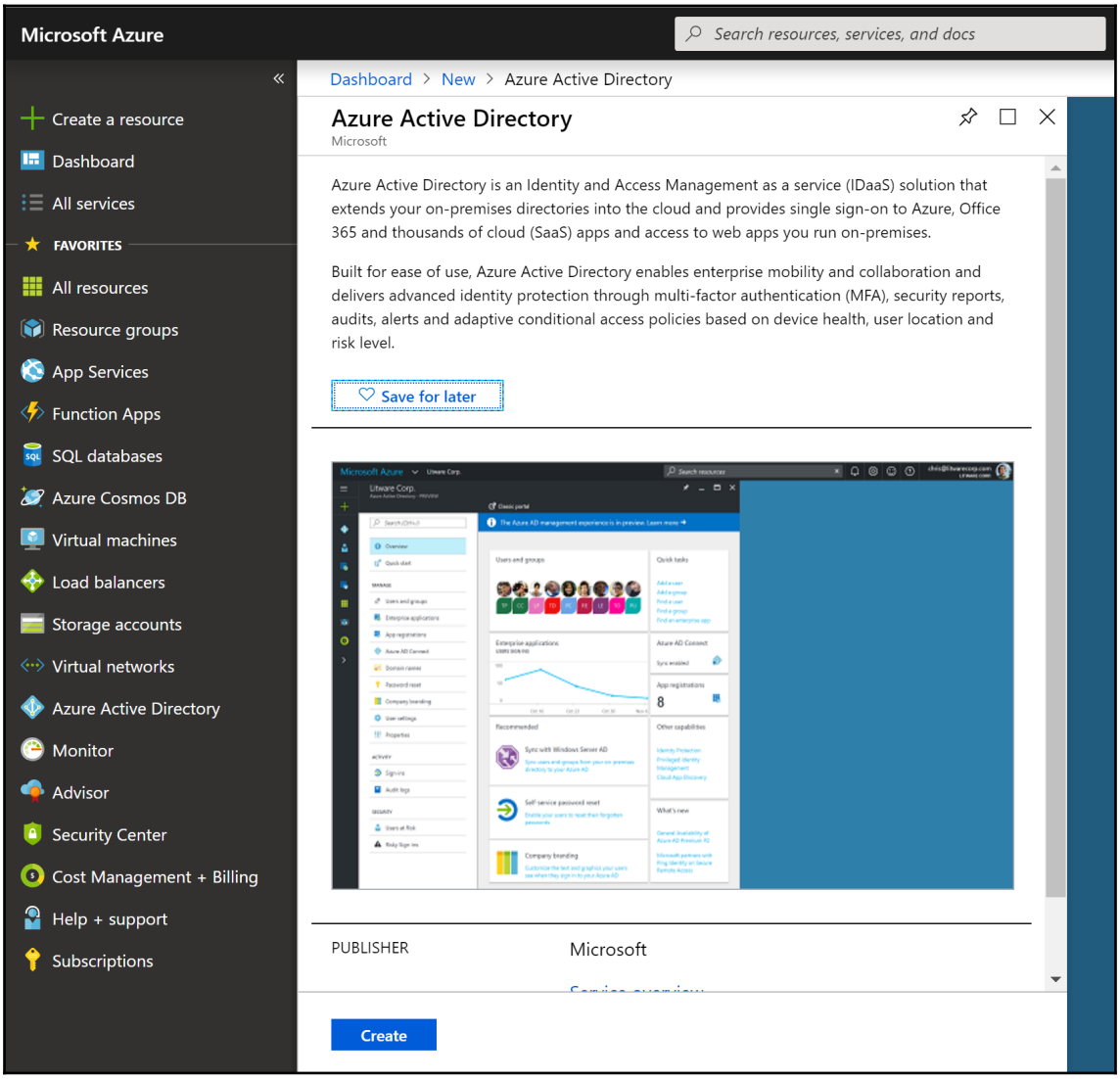
As on-premises Active Directory is backward compatible and can support up to Windows NT as co-existing domain controllers, there is a way to enable or disable support for different Windows versions.



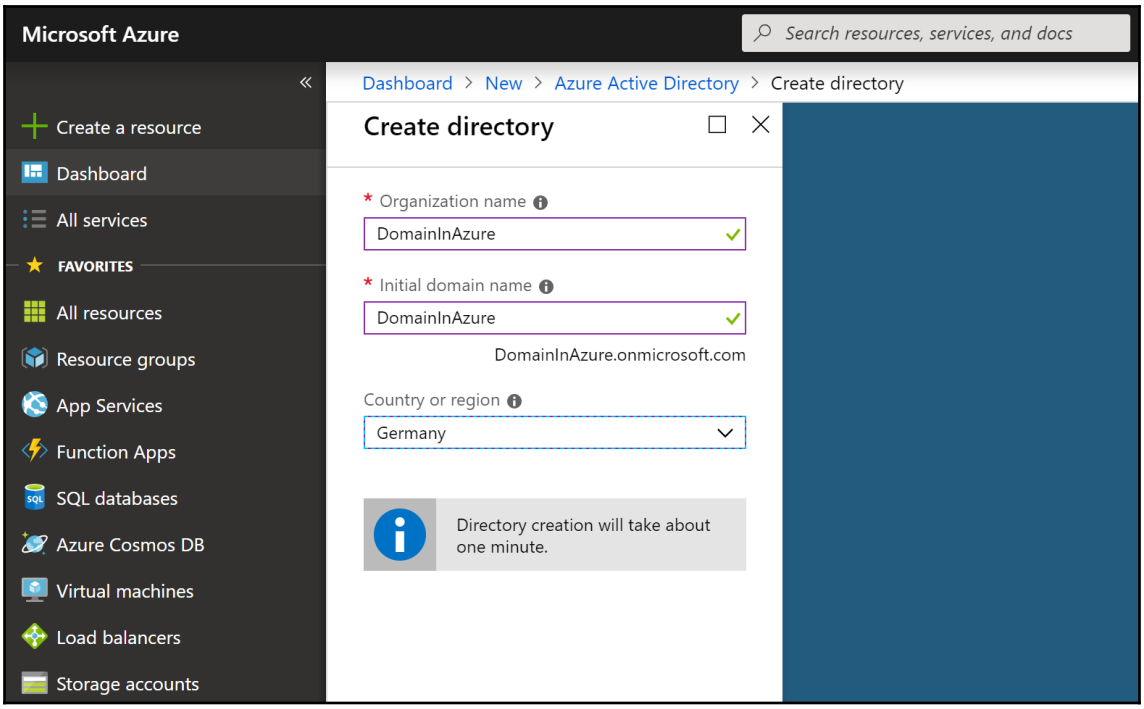
There is a domain functionality level and a forest functionality level. The domain functionality level defines whether there are specific DCs on a specific version of Windows in the domain. The forest functionality level defines the version of supported DCs in a forest. This means, for example, that in Windows Server 2016, only functionality level DCs with the operating system version Windows Server 2016 (or higher) are supported.

Azure Active Directory

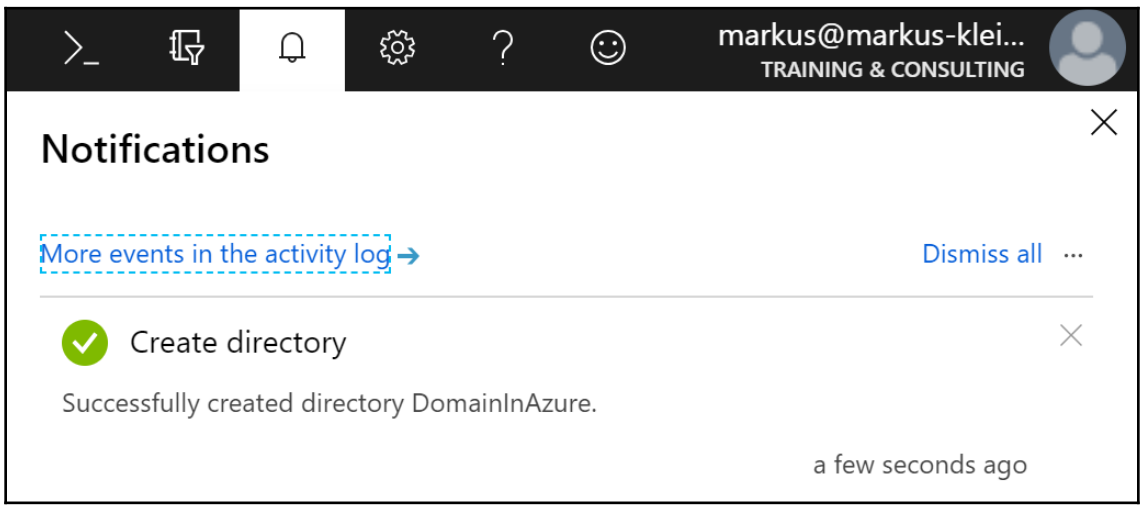
Azure Active Directory is the cloud-based version of Active Directory. In comparison, it is a flat design that only allows for one level of hierarchy. To allow for a complex design, however, there are ways to link different Azure ADs to each other:



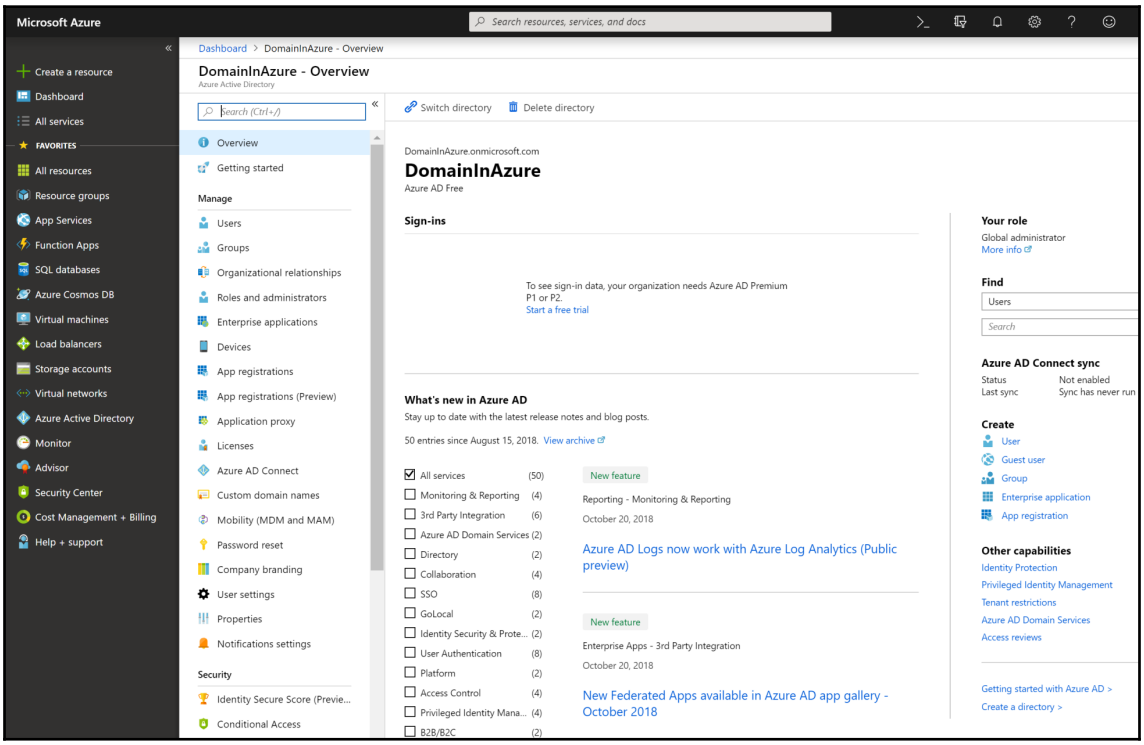
Each Azure AD is unique, and its name can exist only once:



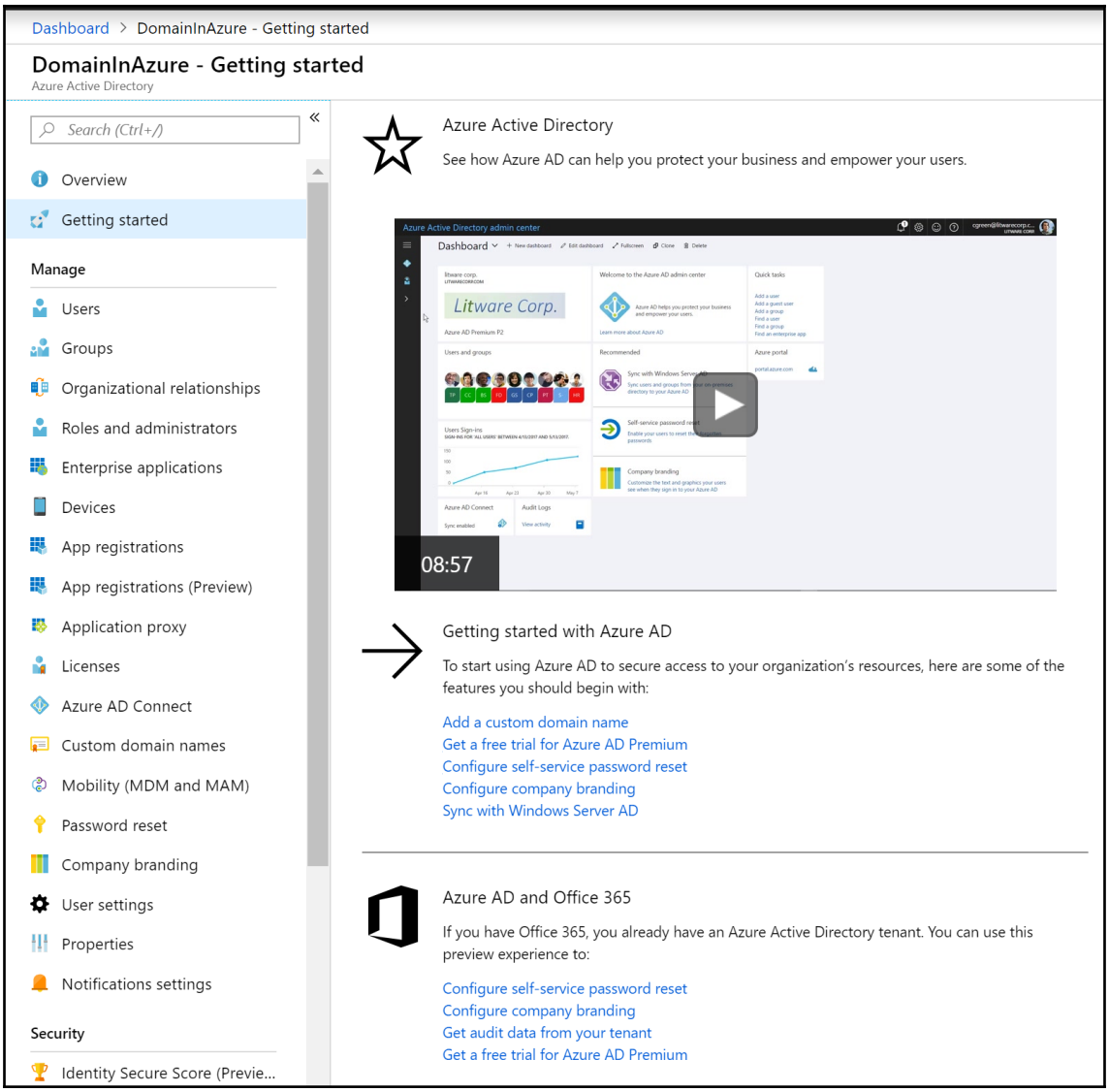
In addition, you can see that each domain name ends with `onmicrosoft.com` as a suffix:



Having created the directory, you will get the notification shown in the preceding screenshot. Azure AD is a worldwide service, in general, but you will have the chance to define in which public Azure area it will reside (for example, Europe):



By default, there is no relationship between an Azure AD and the Azure subscription. An Azure AD can exist even without an Azure subscription. This is most likely the case when you are using Office 365 without any Azure services enabled. The one thing that every Azure public cloud service shares is Azure Active Directory:



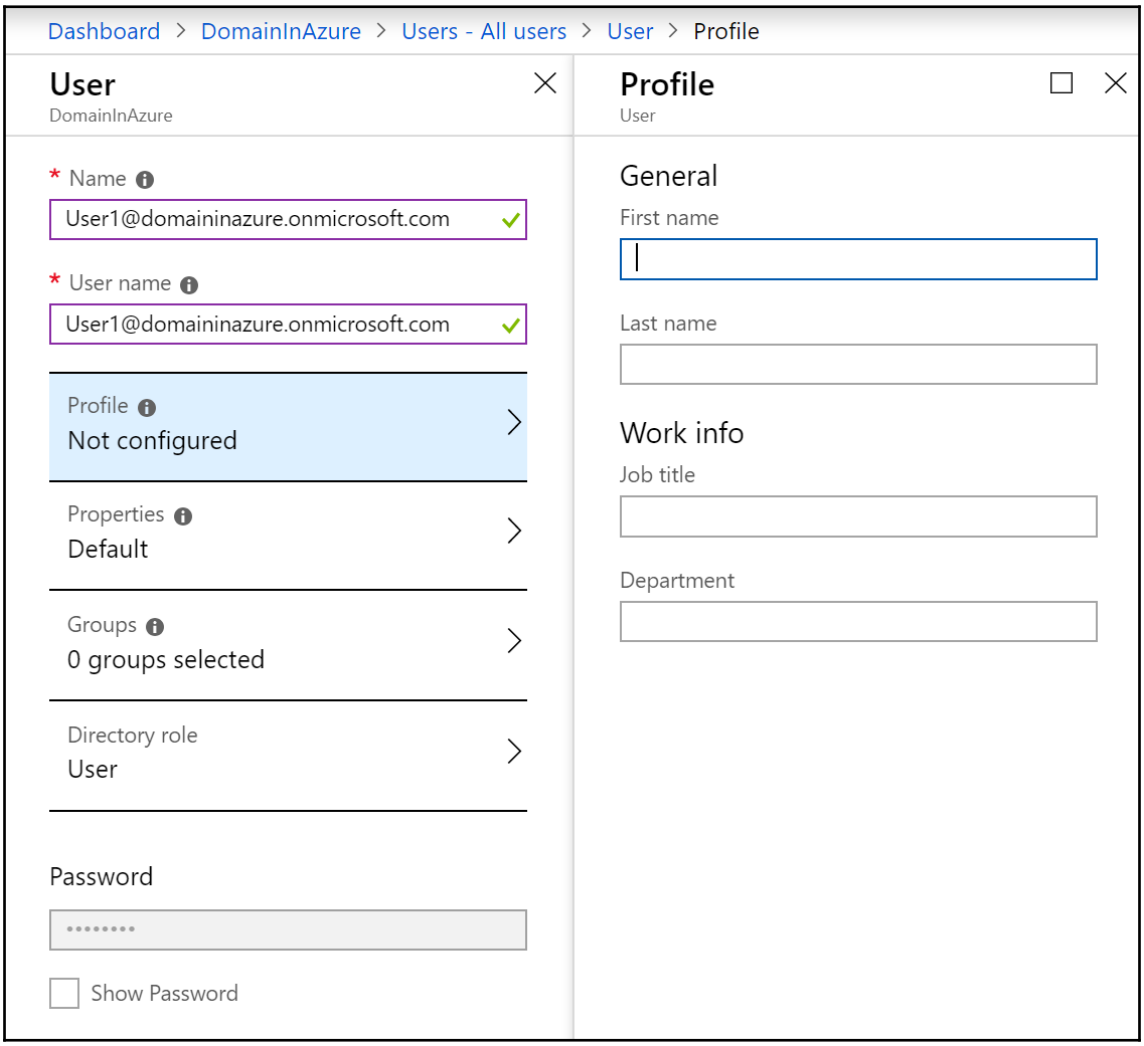
Azure Active Directory is available in different SKUs, as shown in the following screenshot:

- **Azure Active Directory Free.** Provides user and group management, on-premises directory synchronization, basic reports, and single sign-on across Azure, Office 365, and many popular SaaS apps.
- **Azure Active Directory Basic.** In addition to the Free features, Basic also provides cloud-centric app access, group-based access management, self-service password reset for cloud apps, and Azure AD Application Proxy, which lets you publish on-premises web apps using Azure AD.
- **Azure Active Directory Premium P1.** In addition to the Free and Basic features, P1 also lets your hybrid users access both on-premises and cloud resources. It also supports advanced administration, such as dynamic groups, self-service group management, Microsoft Identity Manager (an on-premises identity and access management suite) and cloud write-back capabilities, which allow self-service password reset for your on-premises users.
- **Azure Active Directory Premium P2.** In addition to the Free, Basic, and P1 features, P2 also offers Azure Active Directory Identity Protection to help provide risk-based conditional access to your apps and critical company data and Privileged Identity Management to help discover, restrict, and monitor administrators and their access to resources and to provide just-in-time access when needed.

As you can see, the SKUs differ in the services that are available to use. And, as is always the case with Azure, it is possible to switch from one SKU to another. With Azure AD licenses, this is per user.

Azure AD users

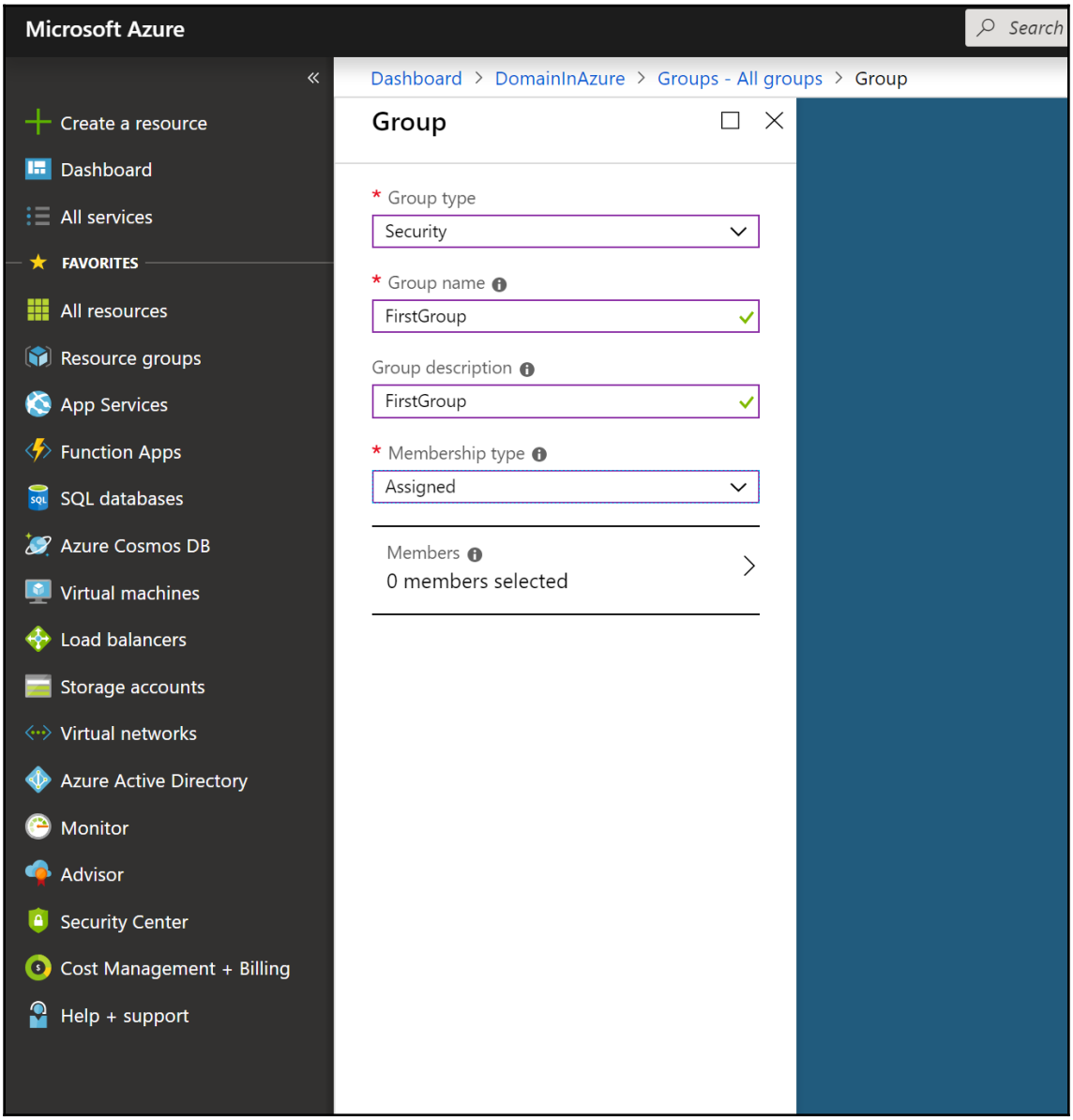
As you can see in the following screenshot, creating a user in Azure AD is quite simple:



As soon as you have created it, you can basically use it worldwide in Azure Public or even Azure Stack.

Azure AD groups

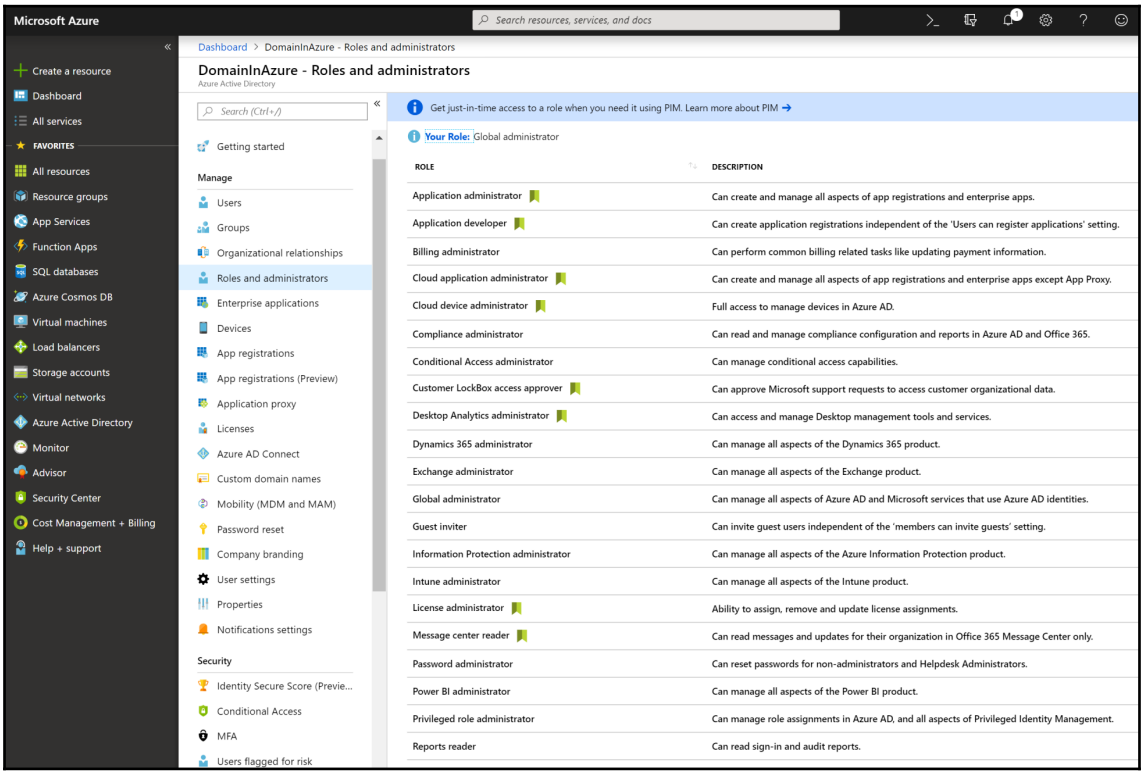
If you need to organize different users, there is a way to create groups in Azure Active Directory, too. Navigate to **Dashboard** | **DomainAzure** | **Groups - All groups** | **Group**, as shown in the following screenshot:



The membership type of a group can be **Assigned** (to a specific user), or **Dynamic**. **Dynamic** means that it is based on user properties.

Azure AD roles

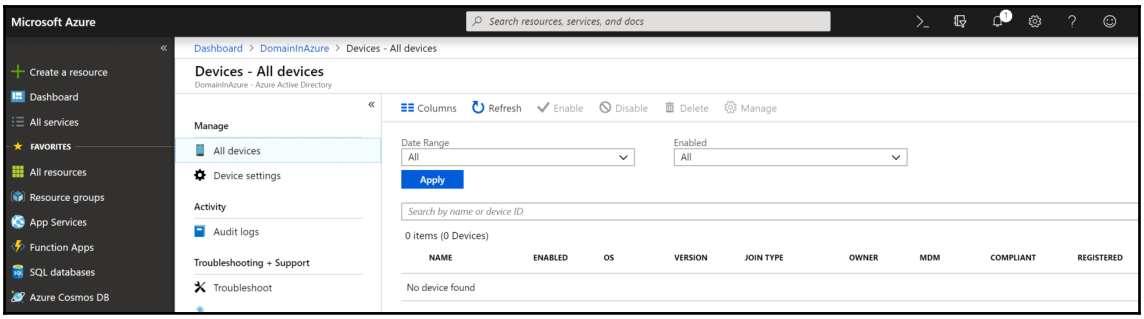
Azure AD permissions most often rely on role definitions. An Azure AD role defines which Azure service, in which quality, can be used by a specific user account, as shown in the following screenshot:



You can even define fine-grained permissions in custom roles, if the preexisting ones do not fulfill your needs.

Azure AD devices

Azure provides an overview of all of the devices that are joined to Azure AD, but device management is part of a specific service, called **Intune**:



So, joining an Azure AD does not allow you to enable or disable options, as compared to group policies in on-premises Active Directory environments.

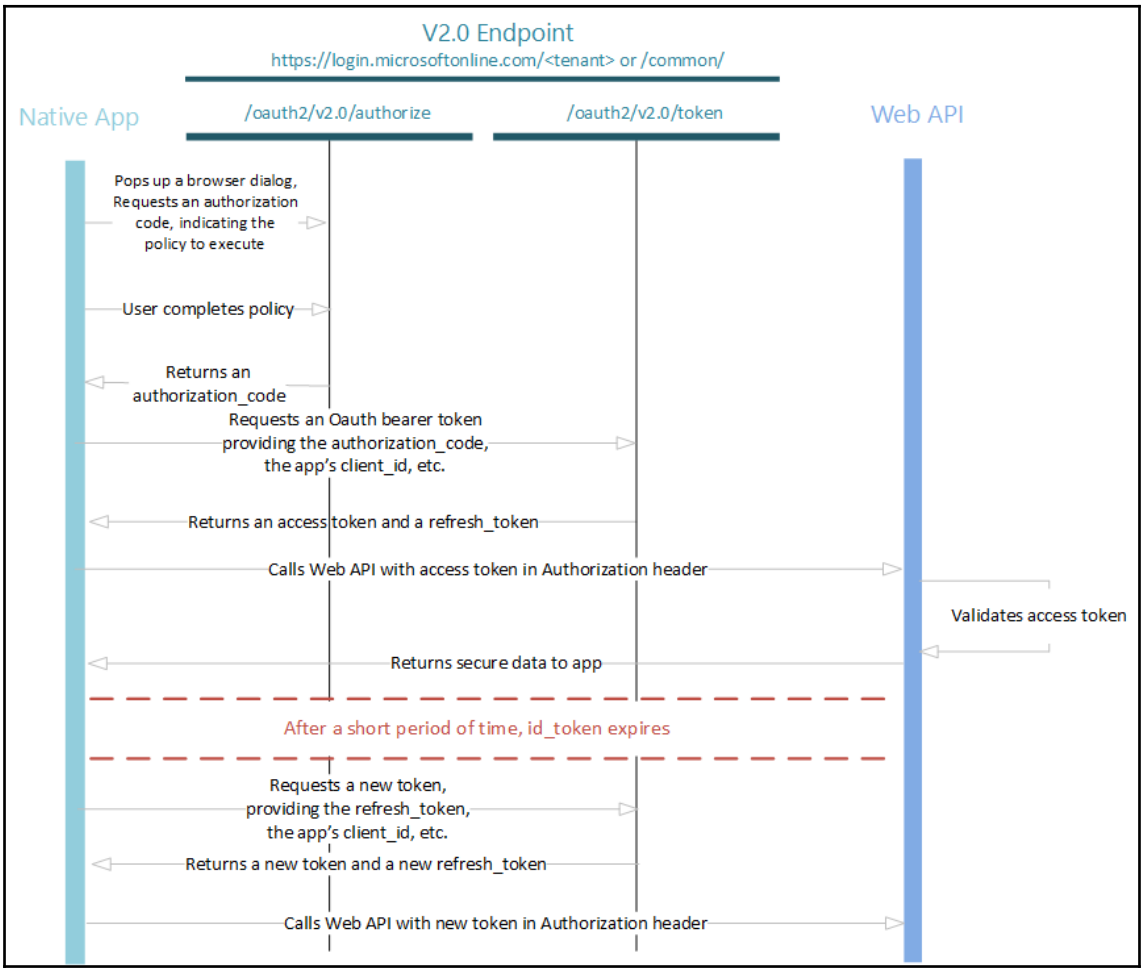
Azure AD design guidelines

It is beneficial to set up Azure AD environments, because Azure AD is, by default, as follows:

- Flat and simple
- Easy to set up
- Not hierarchical

Azure AD relies on oAuth 2.0, which is a default and standardized authentication method.

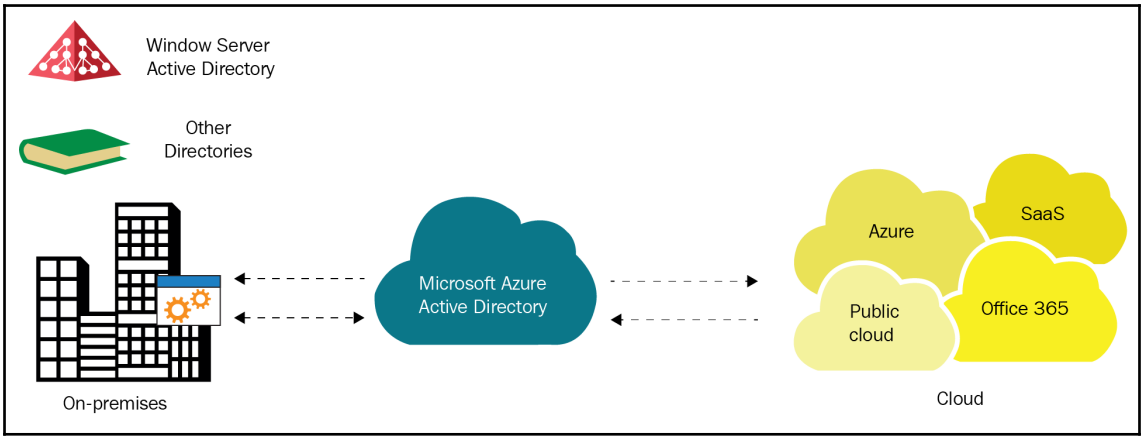
oAuth is an open standard used to delegate access. In general, it is for internet users, and is used to grant access to websites or applications on websites without password exchanges. This authentication mechanism is used by companies like Twitter, Microsoft, Google, and Amazon, to grant access to their services provided via the internet:



At first, oAuth 2.0 may seem complex. But the more you understand it, the more flexible it will be, and, as it is designed for web-based authentication by default, there is no need to translate it into something else.

Designing a hybrid cloud identity management solution

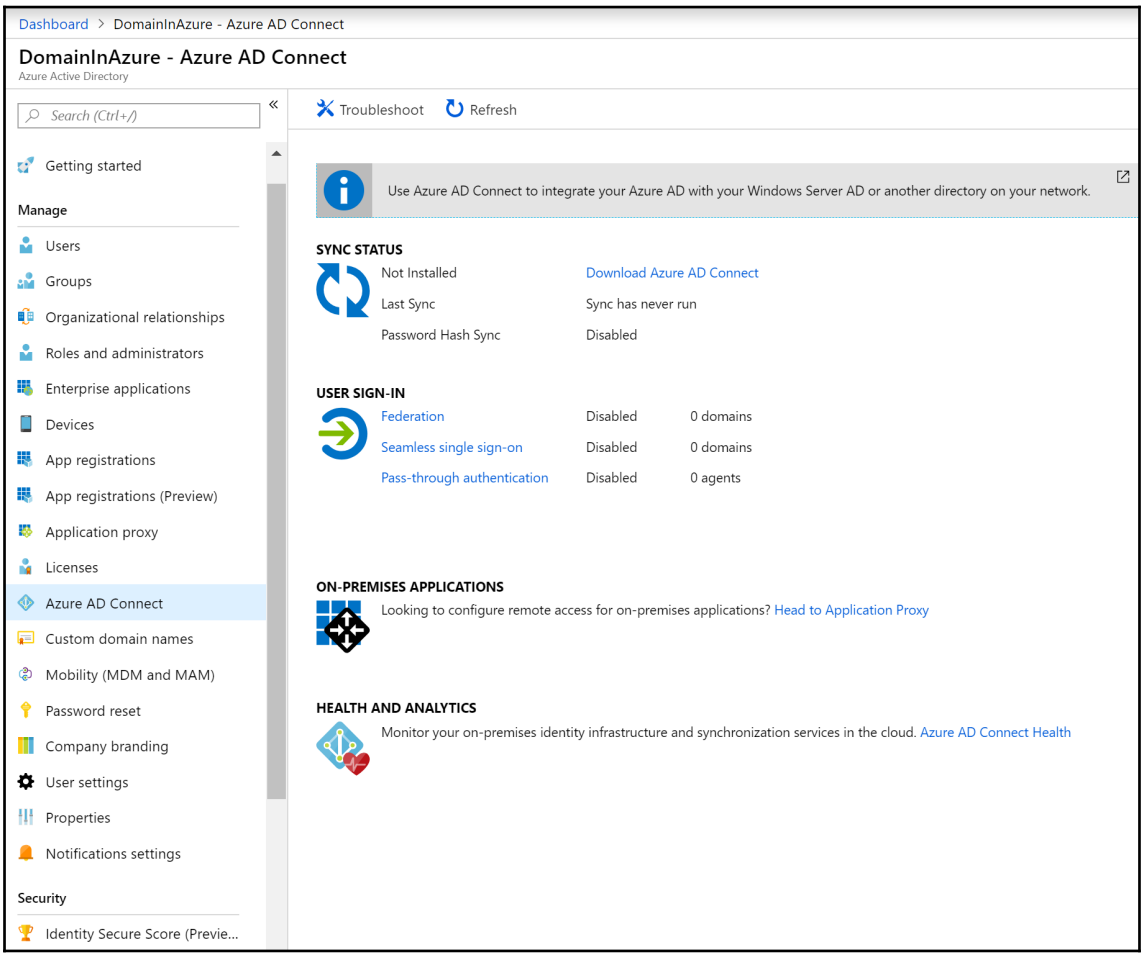
As nearly all of us are coming from something that already existed on-premises, there will be a need to design a hybrid authentication mechanism and bring the on-premises and cloud networks together:



There are three ways to synchronize accounts and/or password hashes between the on-premises network and the cloud, as follows:

- Federation
- Seamless single sign-on
- Pass-through authentication

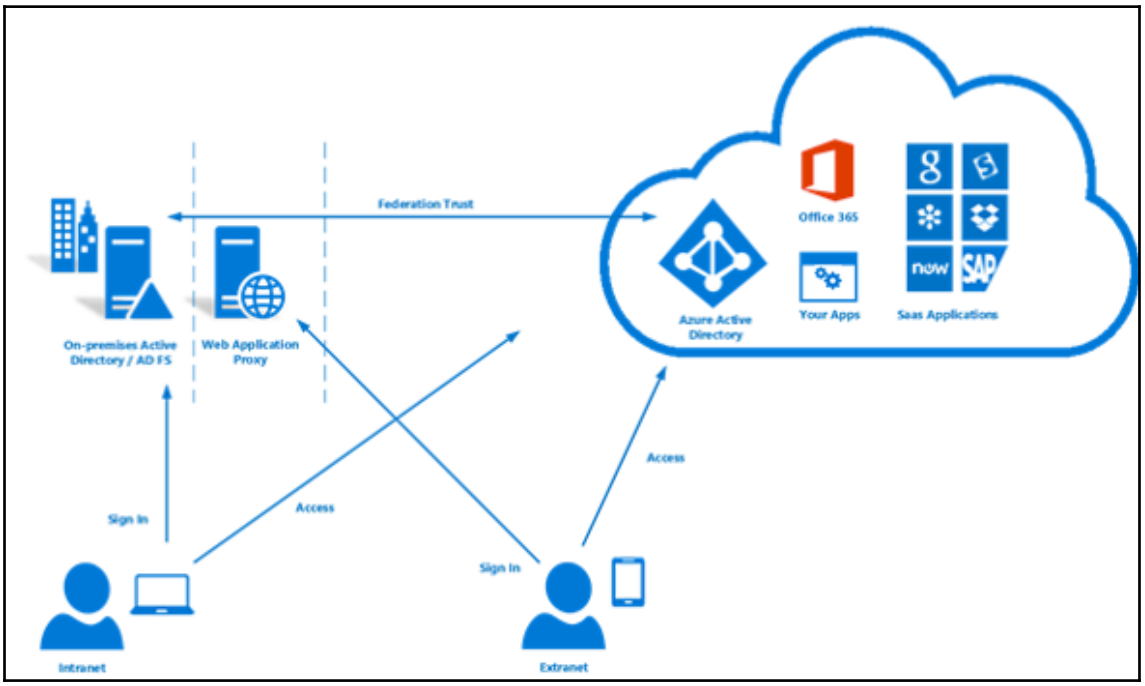
You can find all of them in the Azure AD wizard, which is easy to work through. We just need to understand how they work:



All three of the options are served by the ADConnect tool that is freely available within Azure, for a seamless authentication approach.

Active Directory Federation Services

The basis for **Active Directory Federation Services** (**ADFS**) authentication is that only the user account will be synchronized to Azure, using ADConnect. No password will ever be available in the cloud, so security will stay on-premises, as shown in the following diagram:

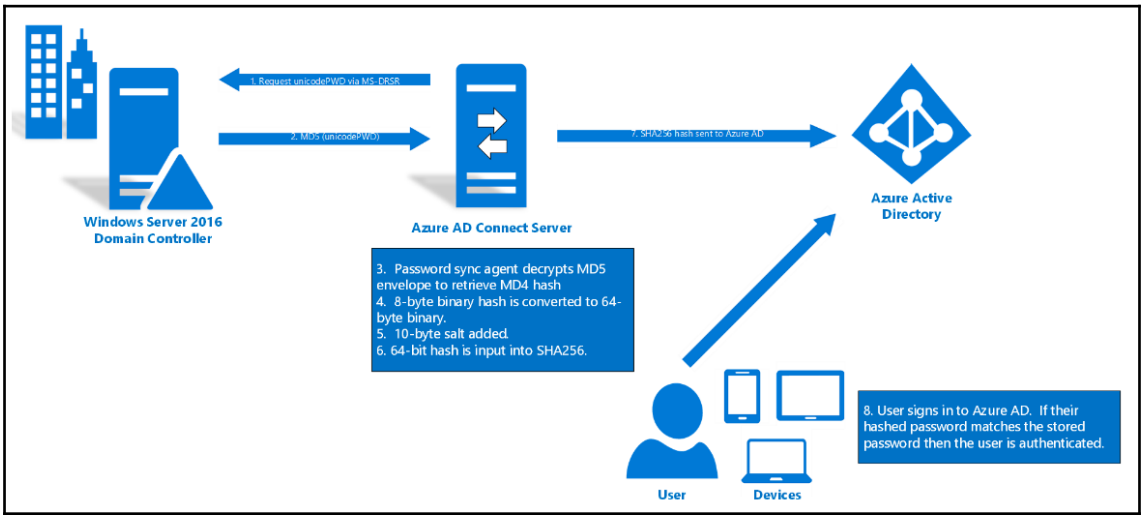


From a user perspective, the UI flow during the authentication will break. At first, you will need to enter your account in an Azure login form. Then, Azure will recognize that the domain is linked to the on-premises environment, and therefore, it will connect back to the on-premises environment (ADFS server). Now, a second authentication window, for creating the login token, will appear. This token will be passed back to Azure.

This authentication is really secure, but if the on-premises connectivity fails, no authentication will be available in Azure services.

Seamless single sign-on

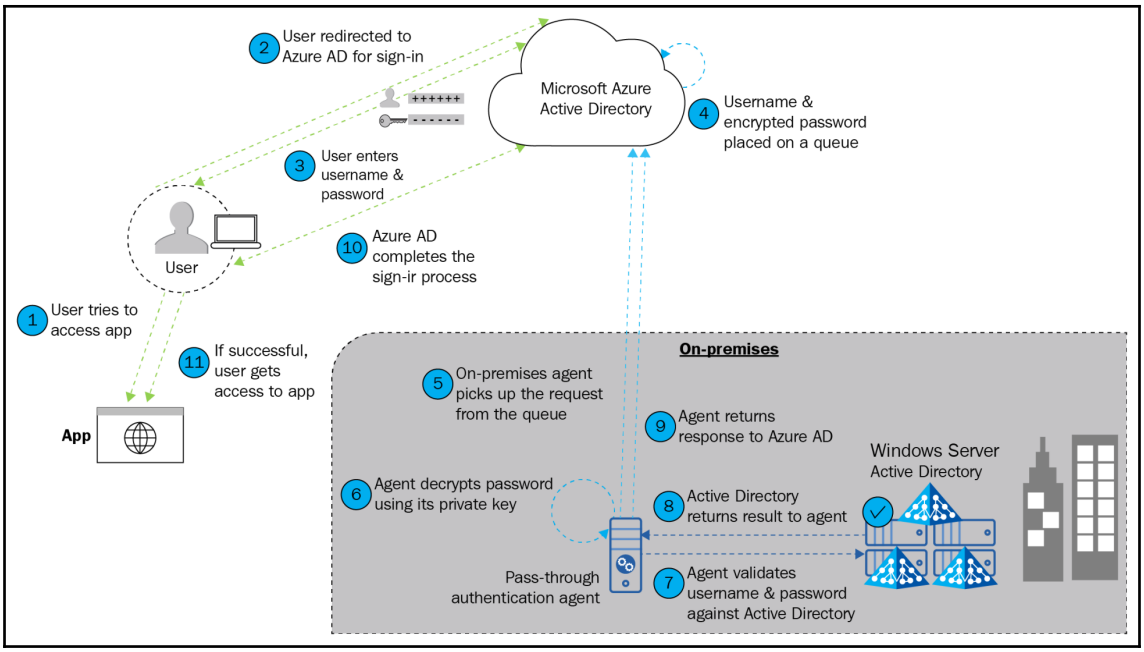
The easiest alternative to enabling access with one account and password, on-premises and in the cloud, is the seamless single sign-on feature. As you can see in the following diagram, the password hash on-premises will be hashed again, using SHA256, and then synchronized to Azure AD:



From an end user perspective, this is the easiest one. There is only one UI for authentication. From a security perspective, a lot of companies' security policies do not allow for using this feature. This is due to the theoretical risk of passwords getting saved in the cloud. But, as it is a hashed password, that is not true at all, from a technical perspective.

Pass-through authentication

The newest approach is the **Pass-through Authentication** (**PTA**) mechanism. It does not require saving the passwords in the cloud. As there is an option to have more than one PTA agent, it is highly available out of the box, without any additional resources:

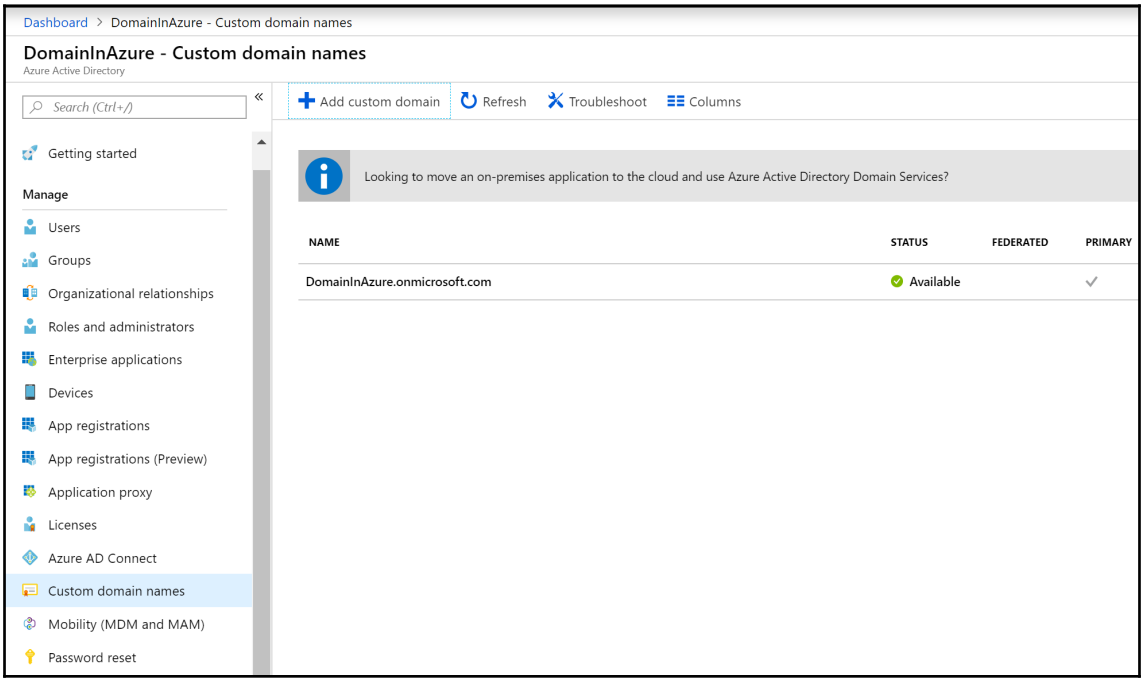


PTA is designed to work with a queuing mechanism. Each authentication request is queued in the cloud. Each server that is running the PTA agent connects from the on-premises network to the cloud, looking for authentication requests. It takes the next one in the queue and authenticates the request in the on-premises environment by returning the directory response to Azure, in order to complete the sign-in process.

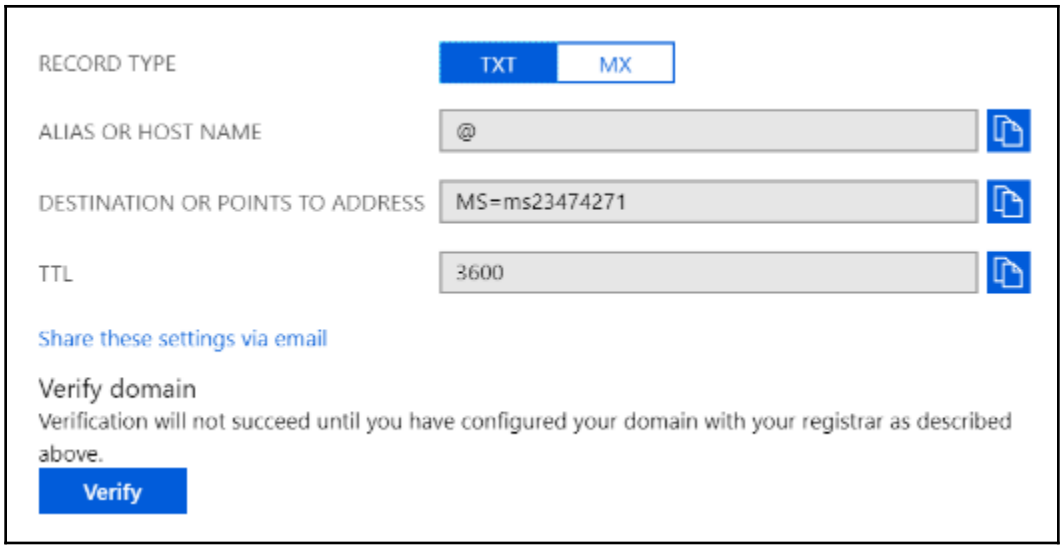
From a user perspective, this technology is as easy as synchronizing password hashes to the cloud, but with the security enhancement that the authentication stays on-premises.

Custom domain names

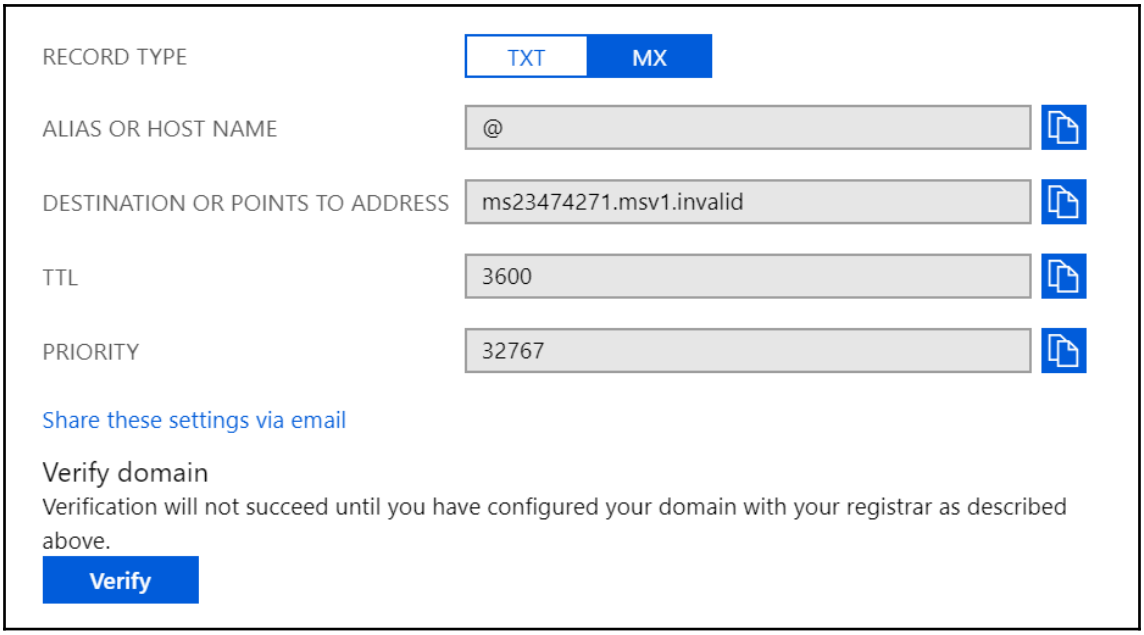
To enable authentication with an account that is on the same domain name as the on-premises environment, you will need to prepare your domain to be available in Azure as a custom domain name that is mapped to the `xxx.onmicrosoft.com` one:



Therefore, you will need to add the custom domain to Azure. This is done by adding a custom `.txt` or `.mx` entry to the publicly available domain, as you can see in the following screenshot:



After modifying the public DNS entries, you can hit the **Verify** button:



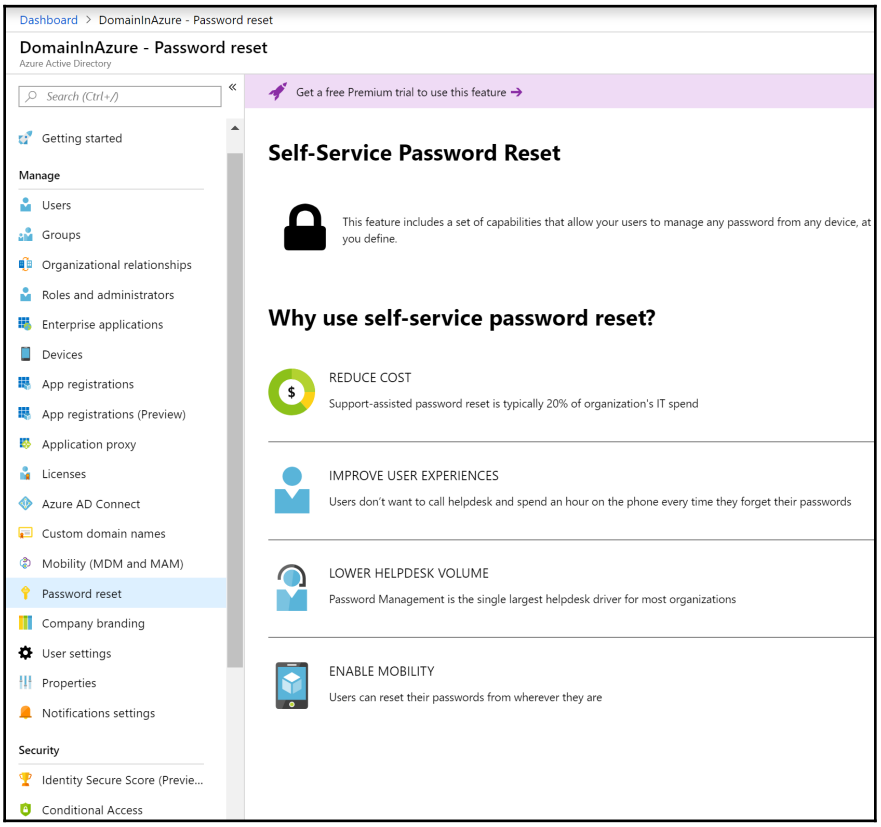
Keep in mind that it can take up to 48 hours for public DNS propagation.

Designing a hybrid cloud access management solution

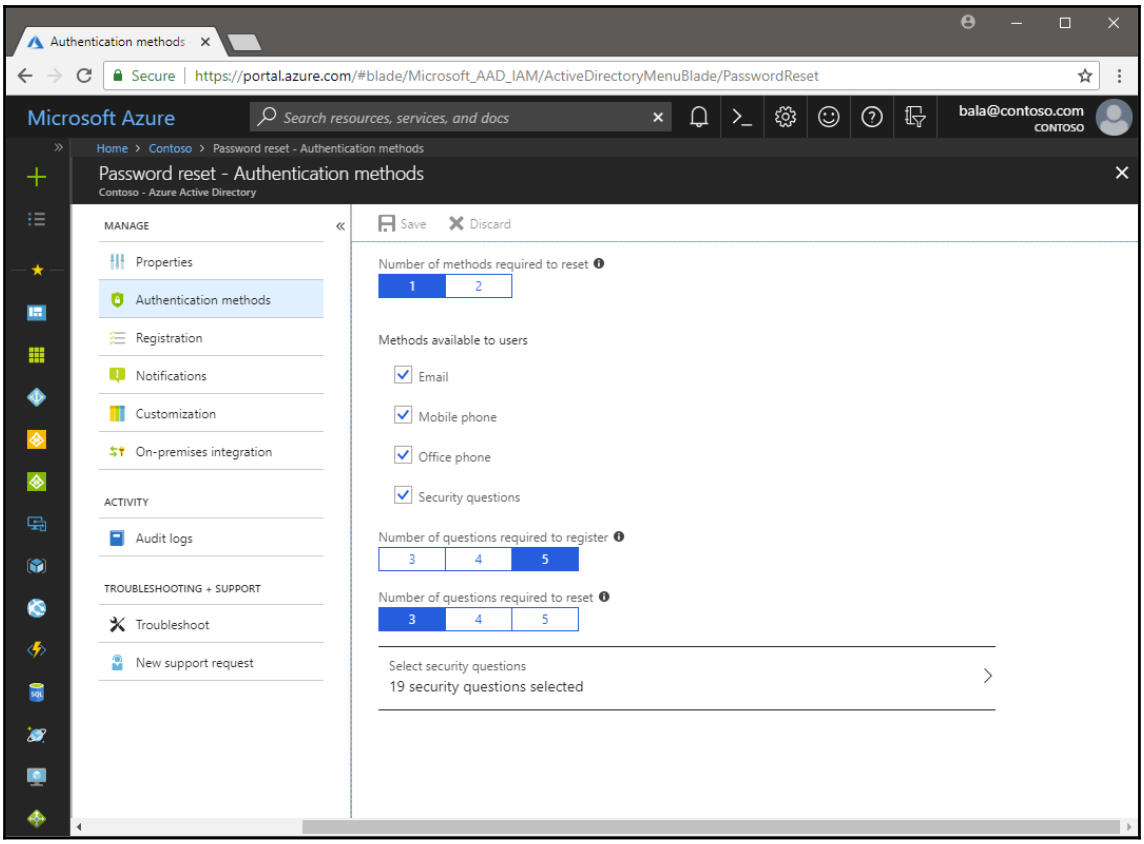
As soon as the Azure AD environment is up and running (and, if required, a hybrid authentication approach is in place), you and your employees will be able to use the new services in the cloud. We will take a look at two useful features: self-service password reset and multi-factor authentication.

Self-service password reset

Azure AD provides an easy and out-of-the-box UI for the self-service of password resets for once in a month or once per quarter. This is one of the most important tasks of user help desk:



To enable this feature, you will need to define the authentication mechanisms, as follows:

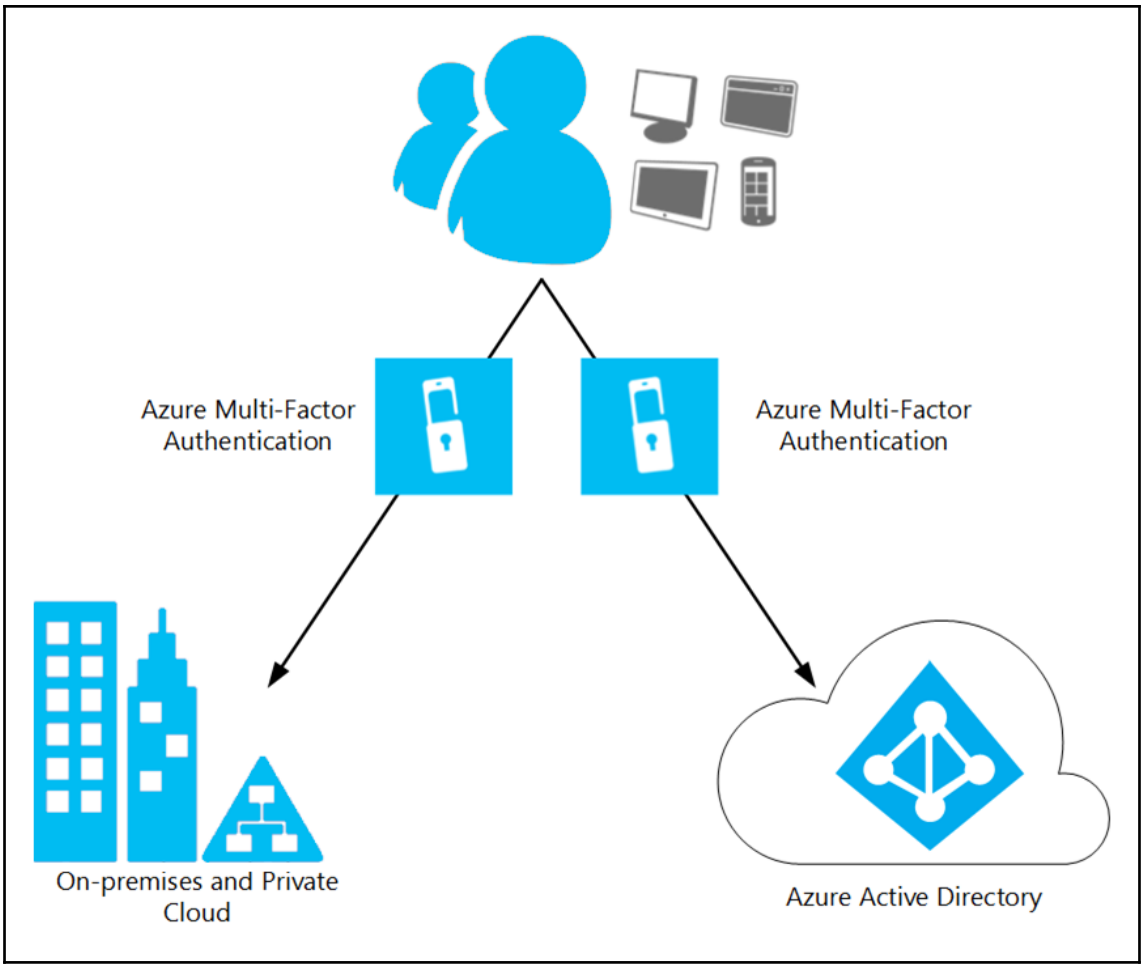


Having provided this configuration, everything will now be up and running.

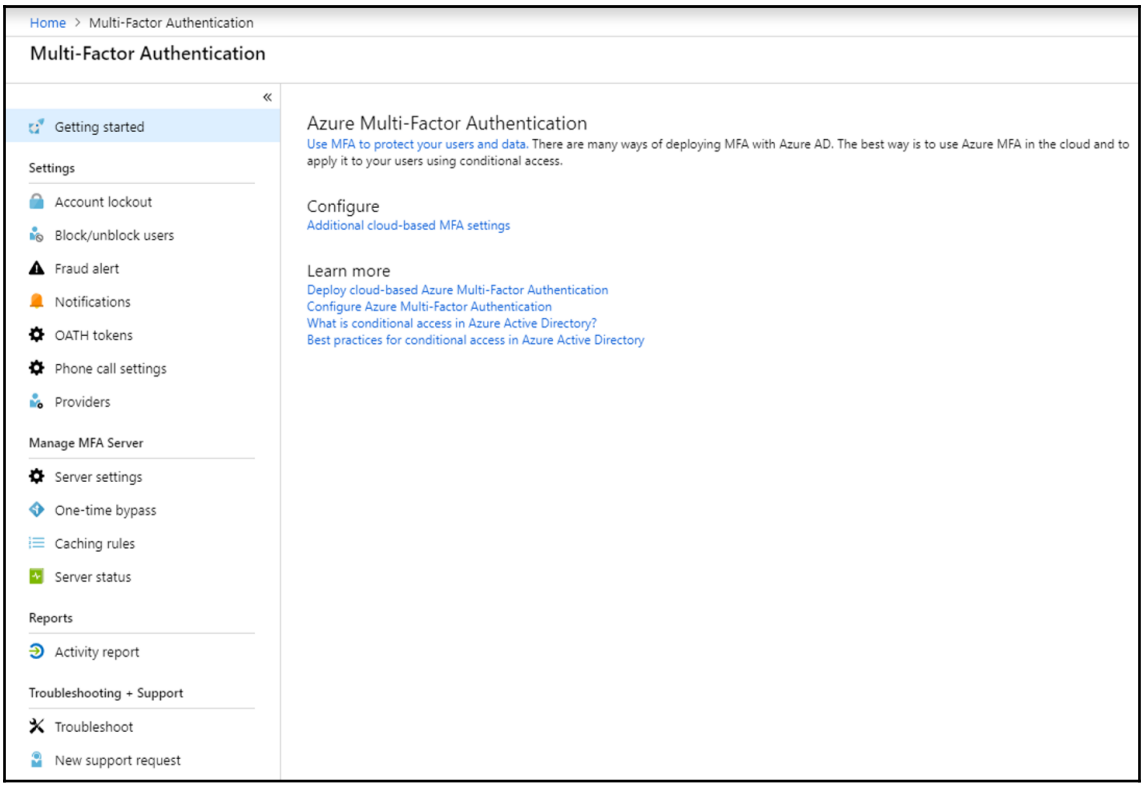
Now, if a user loses his password, he will just have to go to the following URL to use the newly enabled reset procedure: `https://aka.ms/ssprsetup`

Multi-factor authentication

Setting up a service for more than one factor of authentication can be complex. Using Azure AD, however, it is very simple and easy to enable, as you can see in the following diagram:

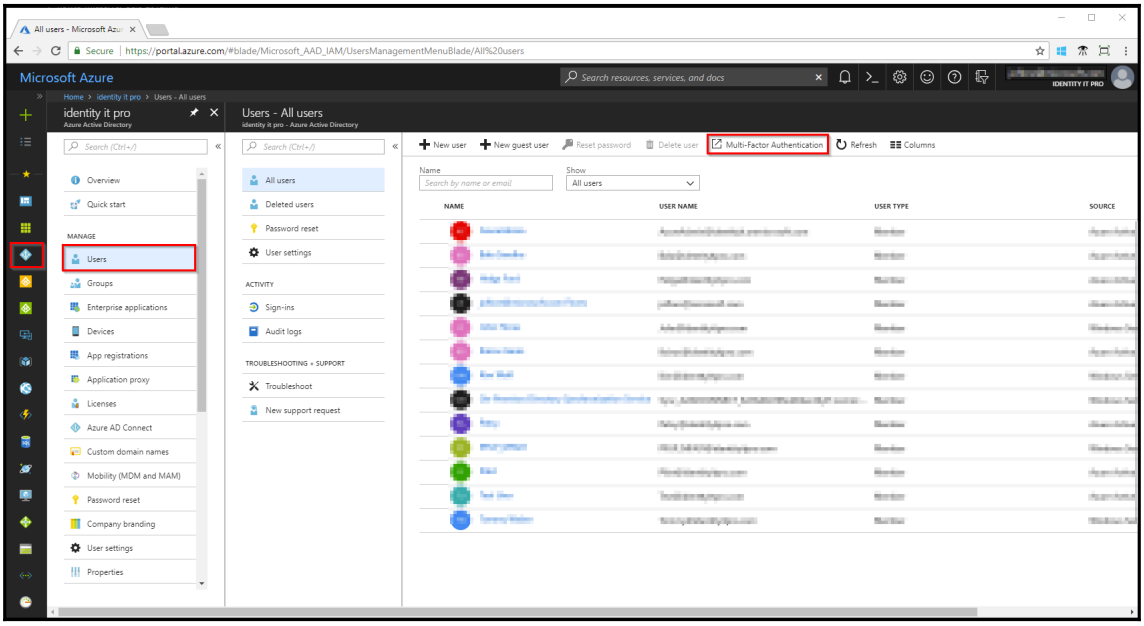


If you own the proper SKU of Azure AD, the process will be just a few clicks away:

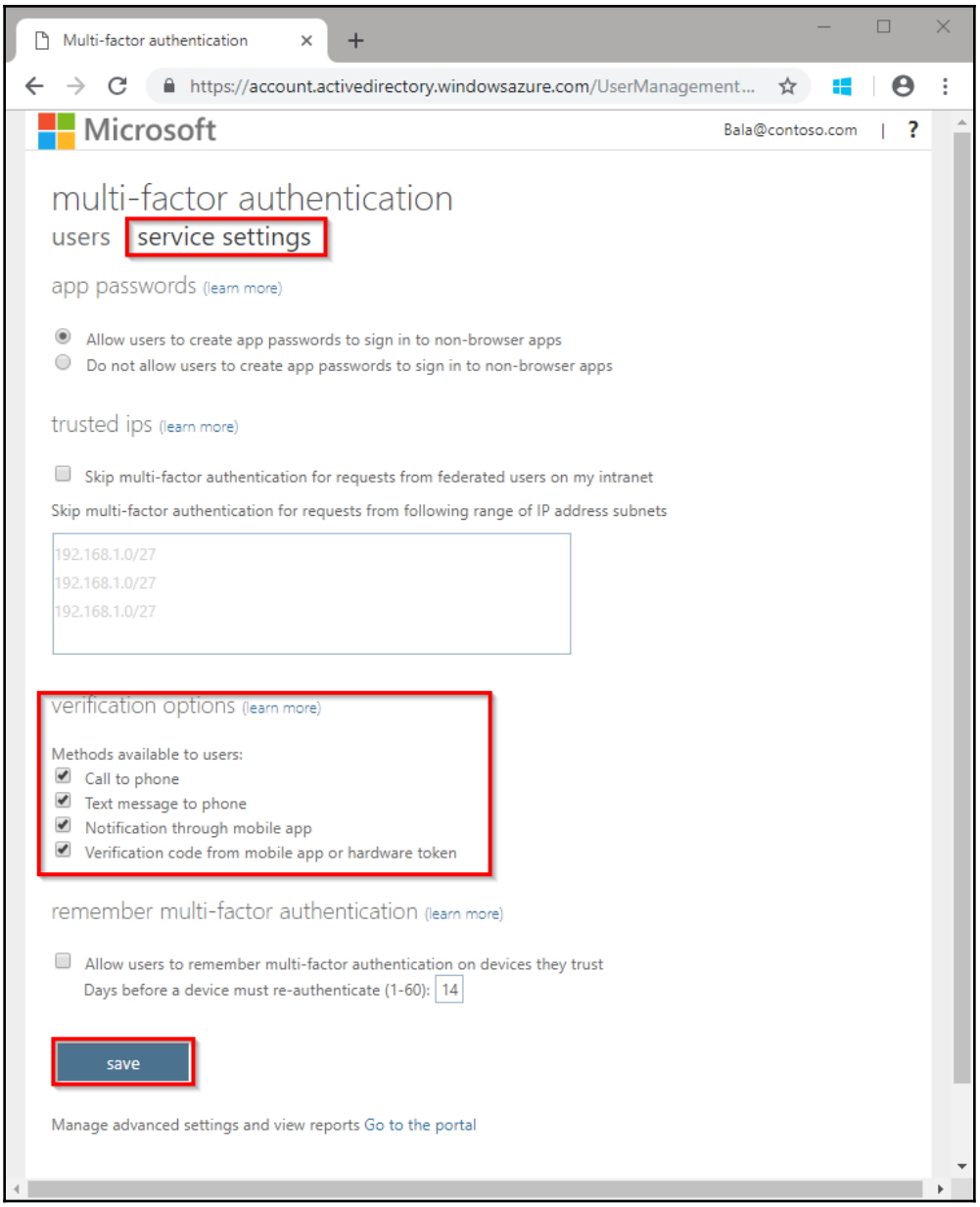


As you can see, Azure MFA includes fraud detection. This means that if a user logs in in different regions of the world (let's say, America and Asia) within minutes of each other, it will appear as potential fraud, and the user will be blocked automatically.

MFA can be enabled per user from the Azure Portal, as follows:



This leads to a specific page, with the options for MFA itself, as follows:



Now, Azure MFA is up and running for the chosen user accounts.

Summary

As you learned in this chapter, there are various options available to support hybrid authorization and authentication using Azure Active Directory and on-premises Active Directory services.

Regardless of your specific design, as soon as you rely on Azure AD, there will be many options available for connecting to these services, whether on-premises or in the cloud. There are even ways to enable Azure AD authentication from third-party cloud vendors (`https://docs.microsoft.com/en-us/azure/active-directory/saas-apps/amazon-web-service-tutorial`).

As an architect, the choice is yours. As soon as you decide to go with Azure Active Directory, you will be in a situation to support tons of authentication providers, using one authentication mechanism. If Azure Stack is a part of your solution, you can enable it to use Azure AD in connected mode, or even to use ADFS in disconnected mode.

With identity management, the best foundation for a public cloud solution has been built. Now, we will dive deeper into the specific services themselves.

11
An Overview of Identity and Access Management in AWS

AWS's **Identity and Access Management** (**IAM**) is a service that helps users to access all of the resources of AWS in a secure fashion. IAM is responsible for authenticating users, and also for ensuring that they have the required authorization to access a particular resource.

IAM provides various services to its users. Some of the important services provided by IAM include the following:

- Creating users and groups
- Creating roles
- Policies for providing granular permissions to users
- Multi-Factor Authentication
- Identity federation
- Integration with most AWS services

All of these services are provided to the user free of charge.

In the following sections, we will look at each one of these services in detail.

How IAM works

To understand all of the services of IAM, it is necessary for you to know how it works. As you saw earlier, IAM is responsible for controlling authentication and authorization for your account. The IAM infrastructure that provides us with authentication and authorization has various components. The components of the IAM infrastructure are as follows:

- A principal
- Action or operations
- Resources
- The request
- Authentication
- Authorization

The principal

When we speak about authorization, we mean that some entity is trying to access a resource, and IAM has to ensure that the entity has the necessary permission to access that resource. The entity that requests access to a resource is called a **Principal** in IAM.

As you can see, any user can be a Principal. Similarly, roles and applications, and with federated users, can be Principals. (We will talk about roles and federated users later in this chapter.)

In essence, any entity that requests for an action or operation on an AWS resource is a Principal.

Resources

Every AWS service will have resources associated with it. Resources can be thought of as objects within a service, upon which some operation can be performed. For example, an S3 bucket is a resource. Users can create a bucket, delete a bucket, and so on. An EC2 instance is another example of a resource. Users can launch an instance, stop an instance, terminate an instance, and so on.

Depending on the service, you can have a number of operations on a resource. An EBS volume will have snapshot operations, AMI operations, and other operations attached to it. The actions that can be performed on a resource are clearly documented. If the user tries to perform an unrelated action on a resource, the action will fail.

Actions or operations

As we stated earlier, every AWS resource has actions or operations defined within it. The exact operations that are allowed are dependent on the service. Operations can include things such as viewing the resource, creating the resource, editing the resource, deleting the resource, and so on. Some examples include: listing S3 buckets, attaching a different role to an EC2 instance, copying a snapshot to another region, or deleting an RDS instance.

Authentication

AWS uses two kinds of authentication. One is when you log in to the console. A user logging in from the console must provide the username and password. Another is when the user uses the CLI or AWS APIs. If the user is using the AWS CLI or the AWS APIs (either directly, or via **software development kits** (**SDKs**)), the user must provide the access key and secret key. The access key and the secret key are provided to the user during the user creation process, provided that the user is created as a programmatic user.

If users want extra security, they can opt for multi-factor authentication. You will learn about multi-factor authentication later in this chapter.

Authorization

Whenever we create entities, such as users or roles, we attach policies to them. The policies dictate which resources of AWS the particular entity is authorized to access. There are different kinds of policies, such as user policies, resource policies, and policy boundaries. (We will look at each of them in detail in the *Policy* section, later on.)

IAM authorizes access to a resource based on the policy. Generally, for all users (except the root user), access is denied to all resources. So, the policy attached to a user has to explicitly allow the user to access a resource. The policy can also explicitly deny access to a resource, in which case the user will not be allowed access to the said resource.

Requests

Whenever a Principal wants to access a resource, either via a console, a CLI, or using an API, it sends a request to AWS. The request contains the following:

- The action to be performed
- The resource upon which the action must be performed
- Details about the Principal
- Environment data, such as the IP address, and so on

Using the information that is provided in the request, AWS forms a request context. Using this context object, AWS evaluates whether the request can be authorized.

Users

IAM allows us to create users. We need users for greater control and security. Organizations require multiple users, with each set of users needing to perform select tasks. IAM allows us to give the required permissions to each user to allow them to perform their tasks, and nothing more. The policies of IAM allow us to provide fine-grained permissions to users.

In AWS, we have three kinds of users: root users, IAM users, and federated users. We will discuss each of these users separately.

Root users

An AWS **root user** is created whenever someone creates an AWS account. When creating an AWS account, we provide our email as the login name. When the account is created and you log in using your email, it means that you are using your root account. The root account credentials are the email ID that you use to log in and the password.

The root account has the maximum permissions. In fact, you have unrestricted access to all AWS services when you use the root account. The root account has access to all of the billing information and credit card and debit card details. You can add more credit cards and debit cards to the account, modify the payment methods, and so on, using the root account. You can also change your password using this account.

The best practice, as suggested by AWS, is to use this root account to create other users, with an admin user among them. This admin user can then be used for day-to-day activities. AWS suggests that you do not use the root account for day-to-day activities. It goes without saying that since the root account is a very powerful account and provides access to all of the resources, the credentials of the root account must not be shared with anyone.

IAM users

In the previous section, we indicated that the root account should not be used for day-to-day operations, and that we must create other users and use them to log in. IAM allows us to create users, and these users are called **IAM users**.

In the IAM scheme of things, users provide the identity part of IAM. As you know, in the case of access to any resource, we need to answer two questions: an authentication question, and an authorization question. Who is accessing the resource, and does that person have the rights to access the resource? Users answer the who part, and you authenticate the users.

These users are not separate accounts. They are users within the account. So, when resources are consumed by the users, the billing will happen at the account level. While creating IAM users, we can create them as console users or programmatic users, or both. If they are console users, we create a password for them, using which they can log in. If the users need to use the **CLI** or write programs that use the AWS APIs, then they will need programmatic access. In this case, the users are provided with an access key and a secret key. You can create users not only for individual people, but also for applications.

AWS suggests that once you have created an AWS account, you should log in using the root credentials and immediately create an admin account. This account will have access to all of the resources, except billing. Then, use this account to log in and perform the day-to-day operations.

It is also a best practice to ensure that the root account needs **MFA**. IAM supports MFA, and we can leverage this to provide additional security for our root account.

Federated users

Most companies have their own methods for authenticating their users, such as, say, an active directory. In such cases, these users can be federated into AWS, and we do not need to create IAM users for them. Similarly, users with web identities can log in to AWS using their web identities, and there is no need to create these users using IAM. In these cases, we are using **Identity Providers** (**IdPs**) to provide us with authentication.

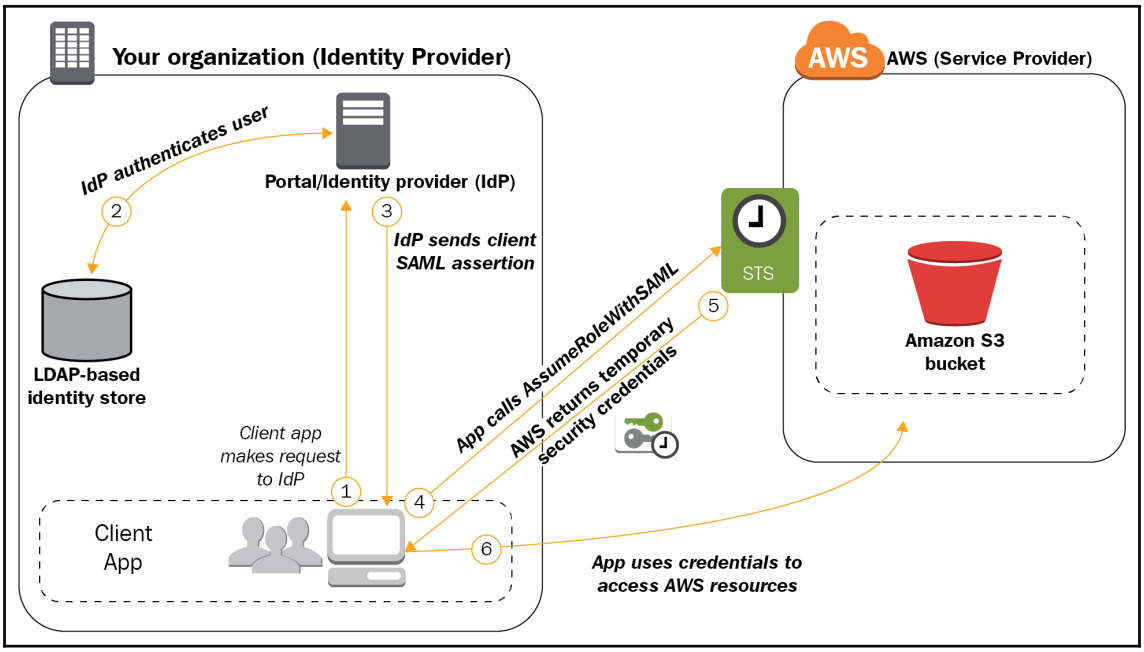
AWS supports IdPs that are compatible with **OpenID Connect** (**OIDC**) or **Security Markup Assertion Language 2.0** (**SAML 2.0**). To use an IdP, we create a trust relationship between the IdP and our AWS account.

SAML 2.0-based federation

SAML 2.0 is an open standard. This standard is used by many identity providers. AWS supports SAML 2.0, and we can federate users from IdP into AWS using SAML 2.0.

Since AWS supports federation with IdP using SAML 2.0, we can use the single sign-on feature to log in to AWS and use the AWS resources, or we can access the resources via the AWS APIs. This can be done without having to create users in AWS. This simplifies the administrative tasks, since we don't need to create for all users in our organization in AWS.

In the following diagram, we provide you with an example of how an SAML 2.0-based federation works:

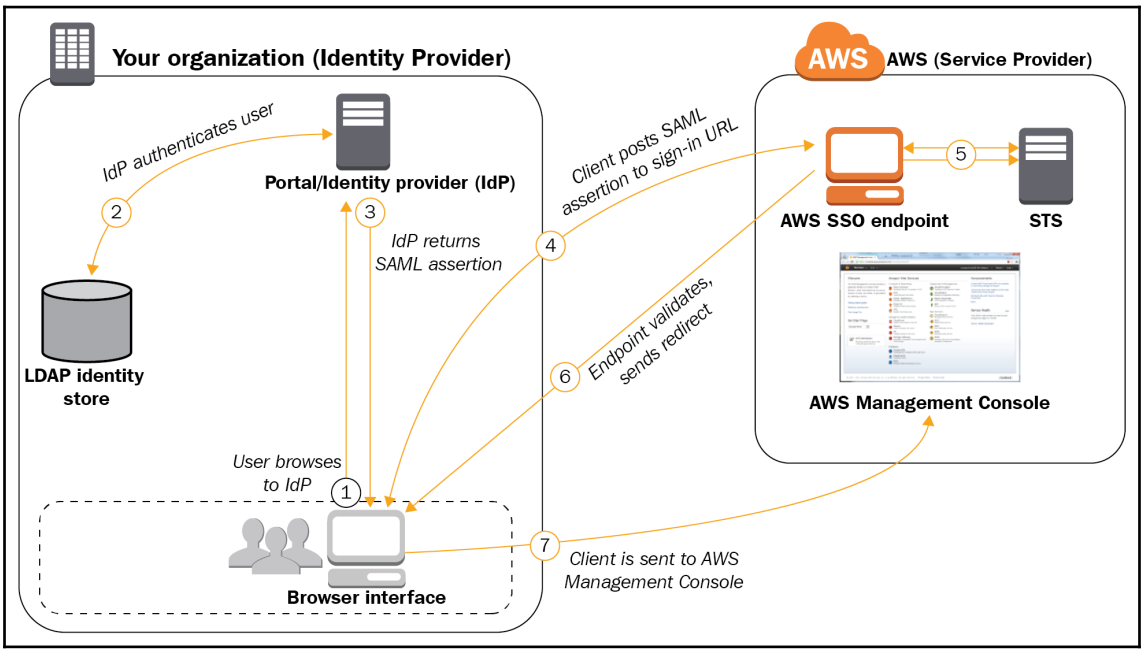


Let's assume that we want the users in our organization to back up their laptop data every day, into a particular bucket in **Simple Storage Service** (**S3**). To access S3, the user must have the required permission. A standard method would be to create a user in AWS and provide the permission to the user to access S3. Instead, we now use Identity Federation to ensure that the user can access the S3 bucket without us having to create a user in AWS.

We can outline the steps that enable this to happen as follows:

1. A laptop user in your organization uses an app to authenticate themselves with the organization's IdP.
2. The user is authenticated by the IdP.
3. The IdP constructs an SAML assertion with info about the user and sends it back to the app.
4. The app now contacts the AWS **Simple Token Service** (**STS**), passing this assertion along with other data, such as the identity of the IdP, and so on.
5. The STS responds to the app with temporary credentials.
6. Using these temporary credentials, the app now calls the S3 API, which enables the user to store data in an S3 bucket.

Federated users can not only use the API, but can also log in to the AWS Console using the corporate directory. The following diagram provides the details:



The following are the steps that are taken to log in to the AWS Console using a federated user:

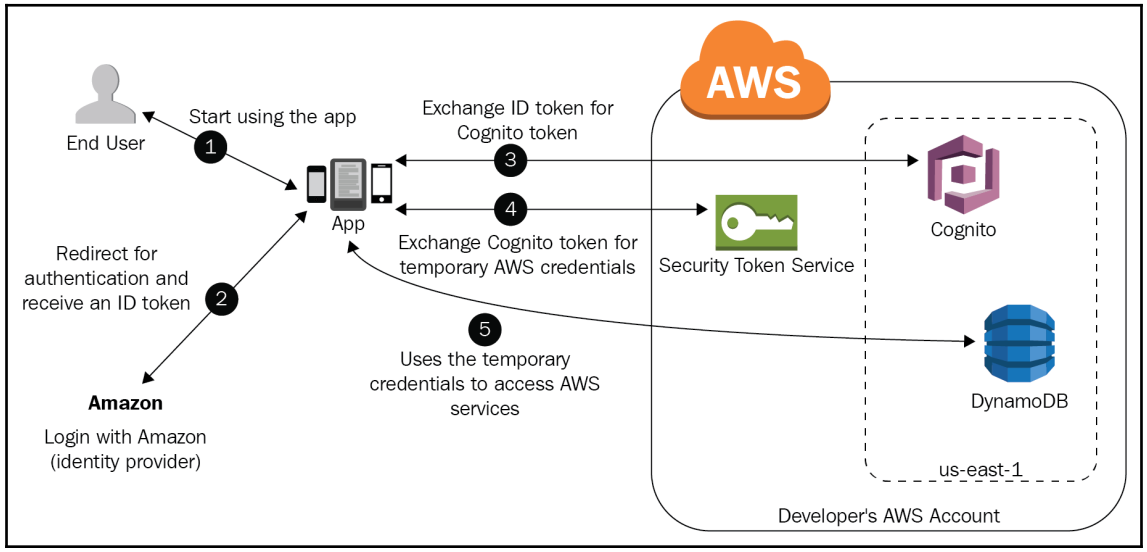
1. Typically, an organization will have its own portal, in which it will provide an option for the user to log in to the AWS Console.
2. The user will input their credentials (the username and login, as per the corporate directory) into this portal.
3. The credentials will then be verified against the corporate directory.
4. An SAML assertion will be returned by the IdP.
5. The portal will send the assertion, along with the user data, to the client browser.
6. The browser will now be redirected to an AWS **Single Sign-On** (**SSO**) endpoint, and the SAML assertion will be posted here.
7. The endpoint will then communicate with AWS STS on the behalf of the user and obtain temporary credentials and a sign-in URL that uses those credentials.
8. The sign-in URL will then be sent back to the client browser.
9. The client browser will then be redirected to the AWS Management Console.

Web identity federation

In an earlier section, you saw how we can use the corporate directory as the identity provider and access AWS resources without creating users in IAM. Now, you will see how we can use some well-known identity providers such as Amazon, Google, or Facebook to access AWS resources without having to create IAM users. Any IdP that is OpenID Connect-compatible can be used.

Let's look at a sample use case. Let's assume that you are developing a mobile app. Further, assume that this mobile app needs to access some AWS resource, such as an S3 bucket or a DynamoDB table. To access these resources, we need the credentials (the access key and secret key). An AWS best practice is to not embed the key in the mobile. Rather, the app must request temporary credentials and use those credentials to access the resources. This is very similar to what was discussed in the earlier section.

One way of using the Web Identity Federation is to use Amazon's service called **Cognito**. The following diagram provides the steps that are taken when Cognito is used for Web Identity Federation:



The steps taken for Web Identity Federation are as follows:

1. The user is asked to sign in when the user starts the app.
2. The credentials are then authenticated with the IdP (in this case, Amazon). Upon successful validation, the IdP sends an authentication token back to the app.

3. The app then contacts Cognito with this token, and, in exchange, receives a Cognito token.
4. The app now contacts the AWS STS, passing the Cognito token. In return, it gets temporary credentials from the STS.
5. The app then accesses the required resources using this temporary token. The role that is associated with the temporary credentials will determine which services can be accessed by the app. (We will discuss roles further in the coming sections.)

You can do Web Identity Federation without using Cognito. The steps remain the same, except that in place of contacting Cognito and obtaining a token, we use the `AssumeRoleWithWebIdentity` API provided by AWS and request temporary credentials.

The app can cache the temporary credentials, so that you don't need to access STS frequently. Generally, the temporary credentials are valid for a period of one hour. Before the credentials expire, the app has to make another call to `AssumeRoleWithWebIdentity` to obtain a new set of credentials.

Groups

Groups are a concept that is used to provide ease of administration. A group can be created, and permissions can be attached to that group. Later, users can be added to the group. The users thus added will get all of the permissions available for the group. While the group does have permissions, it is not an entity that you can log in with. It is available to ensure that the administration of users is easy.

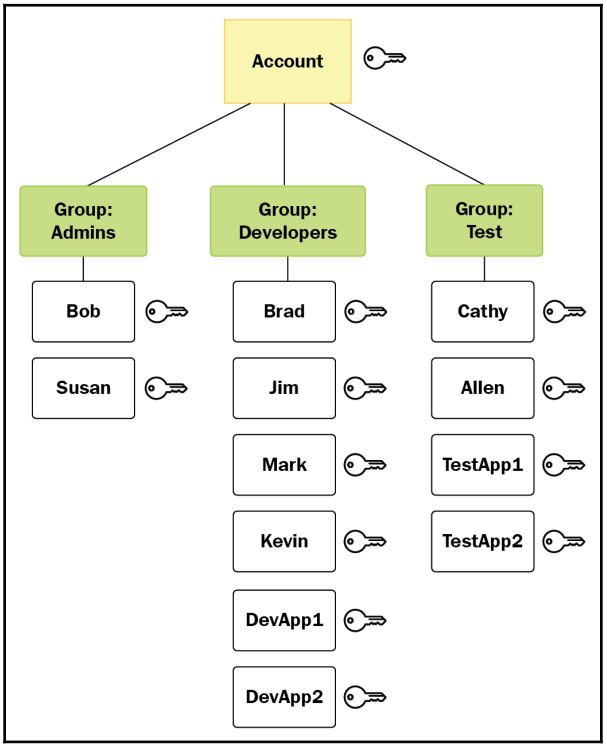
Let's suppose that you have around six administrators that are basically backup administrators. These users require access to the compute service, EC2, and to the storage service, S3. In such a case, we can create a group called Backup Administrators. Then, we can provide this group with permission to access EC2 instances and S3 buckets. Later, we can add all six administrators to this group. Now, all six administrators will have permission to access EC2, and to also access S3. You don't need to give these permissions to each user. Thus, using a group makes it easy for you to manage the permissions of users, and also helps you to logically group users together.

When a new user is added, we can add the user to the group, and if the role of a particular user changes from, say, a Backup Administrator to a Network Administrator, then the user can be removed from the Backup Administrator group and added to the Network Administrator group. This means that we don't need to keep editing the permissions of each user every time a new user is added, or when the role of a user changes.

Groups have certain properties, which are listed as follows:

- You can add a user to a group, but you cannot add a group to another group. In other words, groups cannot be nested.
- There is no restriction that a user can only be a part of one group. A user can be present in multiple groups, but there is an upper limit to how many groups a user can be a member of.
- Similarly, a group can contain many users.
- There is a limit to the number of groups that you can have per account.

The following diagram gives an idea of the relationship between users and groups:



Roles

Before looking at what a role is in AWS, let's try to understand why we need the concept of roles. Let's look at an example. If you are a user, and you are a developer who develops using AWS APIs, you need an access key and a secret key to access AWS resources from your program. You can store these keys on your laptop (in a configuration file). While this would help you during your development phase, it can become a problem during the deployment phase. You would want to install your application in an EC2 instance in the AWS cloud. The keys would now need to be stored in this EC2 instance in order for your program to work. Unlike your laptop, the EC2 instance may be used by more than one person. This means that storing your credentials (both the access key and the secret key) in the EC2 instance is a security risk. Using roles will mitigate this risk.

A role, like a group or a user, has policies attached to it. Unlike a user, this role can be assumed by any user/service that needs it. In the case of roles, they are not provided with permanent credentials. Rather, each role is provided with temporary credentials whenever a user assumes a role.

Roles are used to provide access to users and AWS services that may not normally have access to certain services. For example, in the earlier example, assume that you are accessing S3 from your program. Then, the EC2 instance upon which your program is running can attach itself to a role that has permissions to access S3. Once that is done, temporary credentials can be provided that can be used to access S3. This way, the program running in the EC2 instance can access S3 without having to store credentials in the EC2 instance.

You can also use roles to provide permissions for a user in one AWS account to access the AWS resources of another AWS account. You can use this to provide permissions for a mobile app to access AWS resources without having to store the credentials in the mobile phone. Users whose identities are not defined in AWS, but are defined elsewhere, such as in a corporate directory, can be provided with permission to access AWS resources.

Now, we have come to the question of who can play a role. Roles can be played by the following:

- A user in the same AWS account
- A user from a different AWS account
- An AWS Service, such as Lambda or EC2
- A user authenticated by an external IdP compatible with SAML 2.0, OpenID Connect, or a custom-built identity broker

Let's look at some of the important terms related to roles in the next subsections.

AWS service roles

Let's assume that you are using the AWS serverless service called Lambda. In this case, AWS sets up the servers and runs the code provided by you. In other words, Lambda is a service that executes the code on your behalf. As a part of the service, Lambda may need the login details for another AWS service, called **CloudWatch**. Lambda requires permission to send logs to Cloudwatch. So, Lambda needs to be attached to a role that allows it to send logs to CloudWatch on your behalf. Such roles are called **AWS service roles**.

The permission for such a role would change based on the service that was used. This role can only provide access to resources within the account, and cannot provide access to resources in another account.

AWS service roles for EC2 instances

This is a special kind of service role. When the EC2 instance is launched, we can attach a role to the instance. This role needs to give permission to access the resources needed by the application, which will run in the EC2 instance. The EC2 is given temporary credentials. These are then provided to the application that is running in the EC2 instance.

The role can be attached to the EC2 instance before it starts, or it can be attached to a running instance.

AWS service-linked roles

These are special roles that are automatically linked to a service. The service needs this role (and the permission associated with the role) to function. For example, the EC2 Autoscaling service is used to increase or decrease the number of instances used, based on the load. For the EC2 Autoscaling service to function, it has to get alarms from the CloudWatch service. It should also be capable of launching, as well as terminating, an EC2 instance. So, the EC2 Autoscaling service can only function if it has a set of permissions for doing these tasks on your behalf. (Note that the EC2 Autoscaling needs more permissions than what are stated here.) So, EC2 Autoscaling will be attached to a predefined AWS service-linked role.

The following URL gives you the various AWS services; in the service-linked role column, if you find a **YES**, then a service-linked role is attached to the service. Clicking on the **YES** provides you with details about the permissions given to the role. For more information, go to `https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-services-that-work-with-iam.html`.

Delegation

Assume that you have two teams in your organization that are working as a part of the same project, and further, assume that both of the teams have their own AWS accounts. It is very possible that as a part of the project, one team may require use of the AWS resources controlled by the other team. Similarly, you may have clients that have their own AWS accounts, and that they will need to access some of the AWS resources that you control. For example, the client may need to access files that you have stored in an S3 bucket in your account.

When such requirements arise, we can use the concept of **delegation**. Delegation involves setting up a trust relationship between two accounts; one is the account that owns the AWS resource, and the other is the account that needs access to the said AWS resource. The two accounts can be accounts within the same organization, or accounts in two different organizations.

To delegate, we need to create an IAM role that delegates the permissions. The IAM role needs to be attached to two policies: one policy that details the permissions granted to the user playing this role, and a trust policy that details the trusted account members who can play this role.

When the user takes on this role, they will have to give up existing permissions. The user will now only have the permissions that are attached to this role. Once the user stops playing the role, the original permissions will be restored to the user.

Cross-account access roles

Cross-account access roles provide access in one account to a trusted principal in another account. While roles are the primary means to grant cross-account access, in the case of some services, access to the resource can be provided directly to a different account by using the resource-based policy. An example of a service for which the resource-based policy works is S3. In this case, we can attach a resource policy to an S3 bucket, which provides access to a user in a different account. S3, Glacier, **Simple Notification Service** (**SNS**), and **Simple Queue Service** (**SQS**) are some of the AWS services that provide resource-based policies.

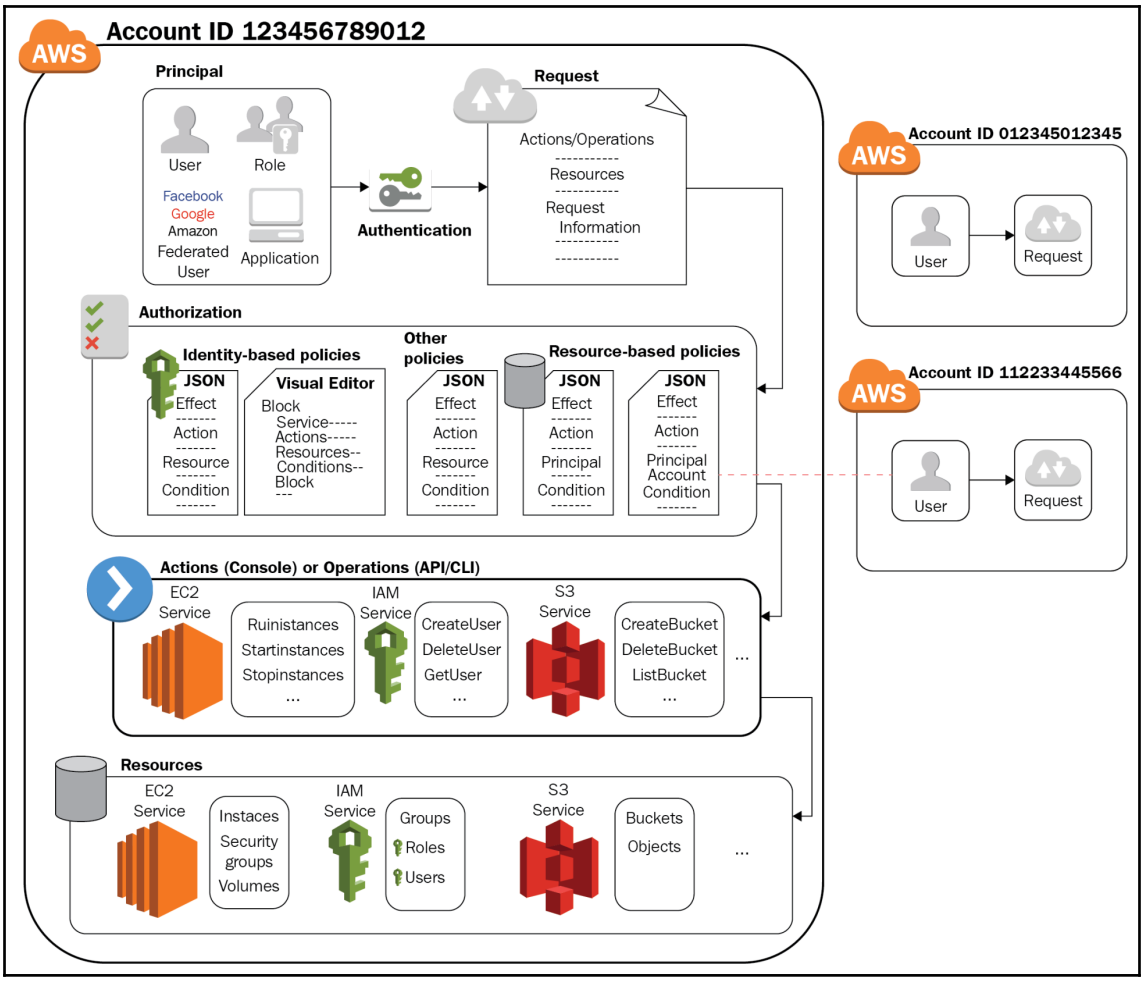
Access management

Access management is one of the key functions of IAM. While the identity answers the question of authentication ("*Is this person who they claim to be?*"), access management answers the question of authorization ("*Is this person authorized to perform the activity on a particular resource?*").

To define what actions can be performed by a user, AWS uses the concept of permission policies and permission boundaries. A policy is a JSON file. In this file, we detail what activities can be performed; that is, what permissions are given. This file is then attached to a user. The user now gets the permission present in the policy document. The permission boundary defines the maximum permission that a Principal can have (a Principal being an IAM user, IAM role, or AWS organization).

To illustrate how the identity and access management parts of IAM work together, let's suppose that we want to access a resource. Which principal wants to access the resource is first authenticated. Once authenticated, the IAM service checks whether the principal has the necessary authorization. If the policy allows for the principal to access the resource, then the access is allowed. You will see more details about the policies in the coming subsections.

The following diagram details the various components of IAM:



We will now talk about policies and policy evaluation.

Evaluating a policy

As we stated earlier, every principal will have a policy document attached to it. You can attach more than one policy to the principal. The policies can both `ALLOW` and `DENY` access. If a `DENY` exists in the policy, it is called an explicit `DENY`.

The policies are evaluated in the following fashion:

- By default, all of the requests are denied. (The root user is the exception. For the root user, all of the requests are allowed)
- An explicit `ALLOW` in the policy will override the default and allow access
- A permission boundary, or the assumption of a role, will override the `ALLOW`
- If there is an explicit `DENY`, then the explicit `DENY` will override all `ALLOW` instances

Policy types

AWS has different policy types:

- **Identity-based policy:** These are policies that we attach to users, groups, and roles.
- **Resource-based policy:** These are policies that we attach to a resource (such as, say, an S3 bucket).
- **Organization Service Control Policy (SCP):** This is used to apply a permission boundary to the AWS organization or an organization unit.
- **Access Control Lists (ACL):** These detail that the principal can access a resource. They are attached at the resource level. These ACLs are not JSON-based. They are very similar to resource-based policies.

Identity-based policies

As we stated earlier, identity-based policies are those that are attached to a Principal, such as a user, a group, or a role. The policy is written as a JSON document and is attached to the Principal. Policies state which permissions the Principal has, and for which resources.

You can use either managed policies or inline policies. In the case of managed policies, you can use the policy to attach it to multiple users, groups, or roles in the account. For our ease, AWS has created some standard managed policies. These are available to us, and we can attach them to any Principal that we want. These are called **AWS managed policies**.

We can write and manage our own policies. We call this a **customer-managed policy**. While the AWS managed policies are very generic, the customer-managed policies can be very specific. For example, the AWS managed policy would give us full permission for S3 access or read-only permission for S3 access. In some cases, this may be what we require, but in many cases, we would need to control it at the bucket level. In such cases, we can use our own policies (customer-managed policies) for this purpose. As a customer, we can also create and manage a policy that is directly embedded into a single user, role, or group. We call this an **inline policy**.

Resource-based policies

An identity-based policy gives permission to a Principal to perform certain actions on a resource. While this is necessary, we may also want to control our resources. For example, if we have some important files loaded into a particular S3 bucket, we may want to strictly control who can have access to this bucket. In such cases, we can use a resource-based policy.

A **resource-based policy** controls access to the resource. It details which principal can perform what action on the resource, under what conditions. As with an identity-based policy, we can create the policy in the form of a JSON document and attach it to the resource.

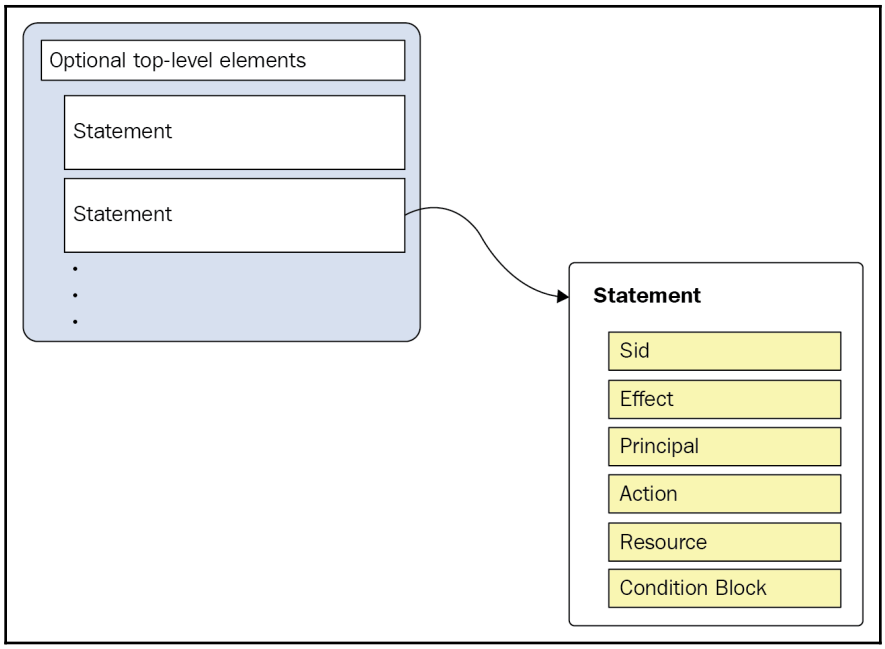
There are a couple of important points to remember here; one is that we must clearly state who is the Principal that can access the resource. In the case of an identity-based policy, this is not needed, since the policy will be attached to the Principal. Second, resource-based policies are always inline policies. That is, you don't have any managed resource policies, and for each resource, you have to have an inline policy.

Writing a JSON policy

In AWS, policies (other than ACL policies) are written as JSON documents. These are then attached to users, groups, roles, resources, or an AWS organizational unit. AWS provides a policy-generation tool, as well as a visual editor, which we can use from the console.

A policy document contains two parts, as follows:

- Policy-wide information at the beginning of the document (this is optional)
- One or more individual statements



This version is an example of policy-wide information. This is the policy document version that we are going to use. The 2012-10-17 version is the current version.

An individual statement is made up of the following elements:

- **Sid**: This is a statement ID that can be used to distinguish between multiple statements in the policy document (this is optional).
- **Effect**: This takes one of two values: `ALLOW` or `DENY`. This tells us whether the access to the resource is allowed or denied.
- **Action**: The list of actions allowed or denied on the resource.
- **Resource**: This specifies the resource on which the `Effect` and `Action` apply.
- **Principal**: The user, role, account, or federated user for whom the action is allowed or denied for the stated resource. If the policy is being attached to a user, group, or role, this field is not required. In such a case, the user or role becomes the Principal automatically.
- **Condition**: Here, you can specify the conditions under which the access should be allowed or denied. For example, we can allow access to a particular S3 bucket only if the request comes from a specific IP address. This field is optional.

Let's look at a few examples.

Here is a policy that allows the `List Bucket` operation on a bucket called `mybucket`:

```
{
     "Version": "2012-10-17",
      "Statement":
                {
                     "Effect": "Allow",
                       "Action": "s3:ListBucket",
                       "Resource": "arn:aws:s3:::mybucket"
                }
}
```

Now, let's take a look at a policy that has more than one statement. In this case, we will make up a policy that has two statements. You can then extrapolate it to a policy that can have more than two statements.

In the policy given next, we are providing permission to access `CloudWatch` for `GetMetric Data` and `PutMetric Data`. We are also providing permission for `ListBucket` for all S3 buckets in our account:

```
{
 "Version": "2012-10-17",
 "Statement": [
   {
     "Sid": "VisualEditor0",
     "Effect": "Allow",
     "Action": [
       "cloudwatch:PutMetricData",
       "cloudwatch:GetMetricData",
     ],
   "Resource": "*"
  },
   {
     "Sid": "VisualEditor1",
     "Effect": "Allow",
     "Action": "s3:ListBucket",
     "Resource": "arn:aws:s3:::*"
   }
 ]
 }
```

Managing policies

Earlier, we discussed AWS managed policies, customer-managed policies, and inline policies. In this section, we will look at each of these in more detail.

AWS managed policies

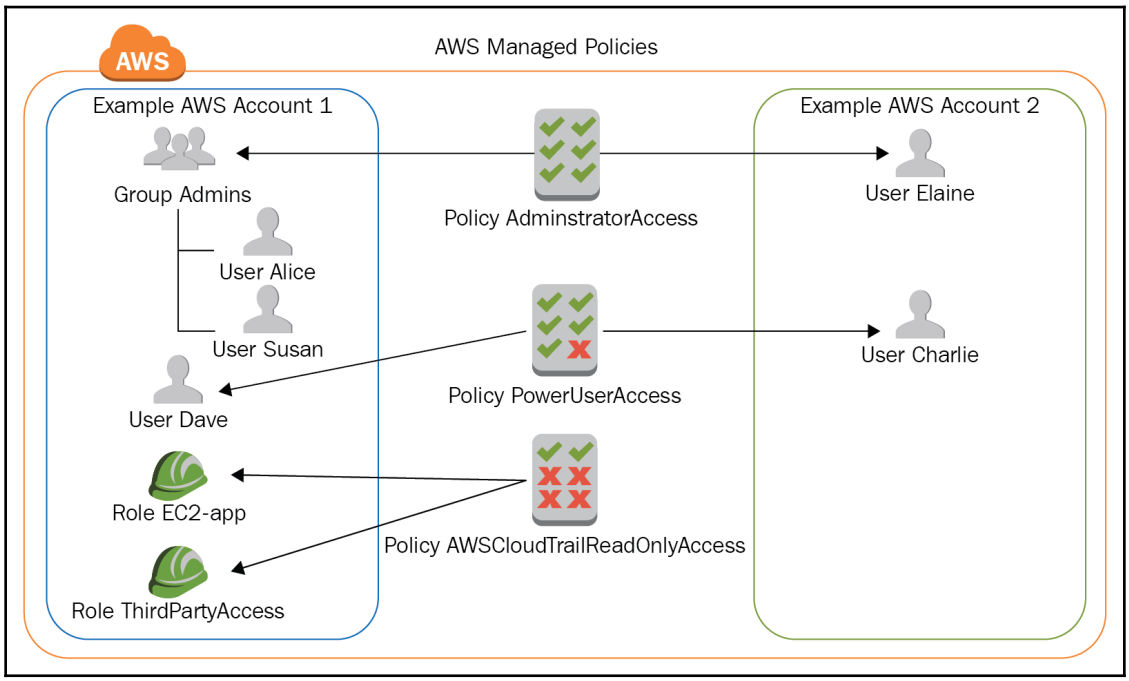
AWS creates certain policies, and maintains those policies. They are called AWS managed policies. These policies are also sometimes called **standalone policies**, and they have their own **Amazon Resource Name** (**ARN**) associated with them.

The idea of AWS managed policies is to provide policies for some common use cases. For example, suppose that you want to provide a backup administrator full access to the EC2 service and to an S3 service; then, we have the full permission policies available as an AWS managed policy. We do not need to write this policy, and we can use the AWS managed policy as it is. Similarly, if you want to provide read-only access to certain services, such as EC2, there is an AWS managed policy, the `AmazonEC2ReadOnlyAccess` policy, which provides you with the necessary permissions.

Another advantage of AWS managed policies is that there are many policies that are defined for various job functions in the IT industry. For example, we have an administrator role, in which the administrator needs access to all of the resources. There is an administrator access policy available for this job function. The advantage that we have is that if AWS adds a new service, then the administrator access is automatically updated to include permissions to access that new service.

As the name indicates, an AWS managed policy is managed by AWS. This means that the user cannot change any permissions that are defined in these policies. Only AWS can update the permissions. When such a thing happens, it impacts all of the principals to which the policy has been attached. As we stated earlier, when a new service is introduced, then the permissions for that service are automatically added to the relevant policies. If you had administrative access, then you will not have administrative access to the new service. If you had read-only access to all services, you will now have read-only access to a new service, as well.

The following diagram shows how AWS managed policies are attached to various Principals:



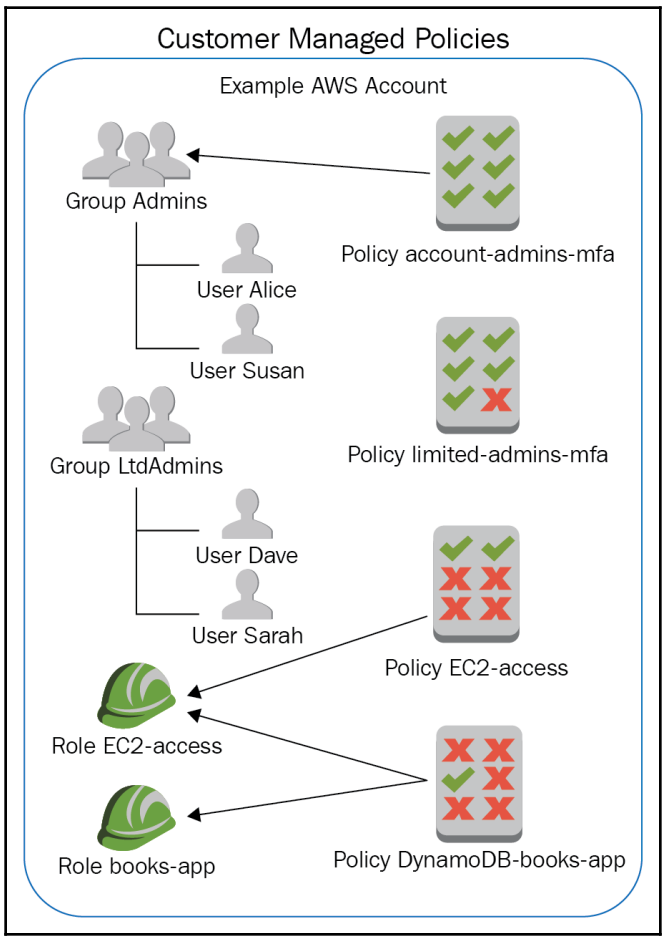
Customer-managed policies

You have seen the AWS managed policy feature, and how it helps us in common use cases and in providing policies for certain job functions. At the same time, we may need to provide more granular permissions to our resource. This can be done by writing and managing our own policies. These are called customer-managed policies. As with AWS managed policies, we can attach the customer-managed policy to multiple entities.

To create a customer-managed policy, we can use the policy-generation tool that is provided by AWS. We can use this tool when logging into the console. We can also copy an AWS managed policy, and edit it to suit our requirements. For example, if we want to provide full access to a particular bucket, we can copy the S3 `FullAccess` policy and change the resource type for S3 to the particular bucket.

As with an AWS managed policy, a customer-managed policy also has its own ARN.

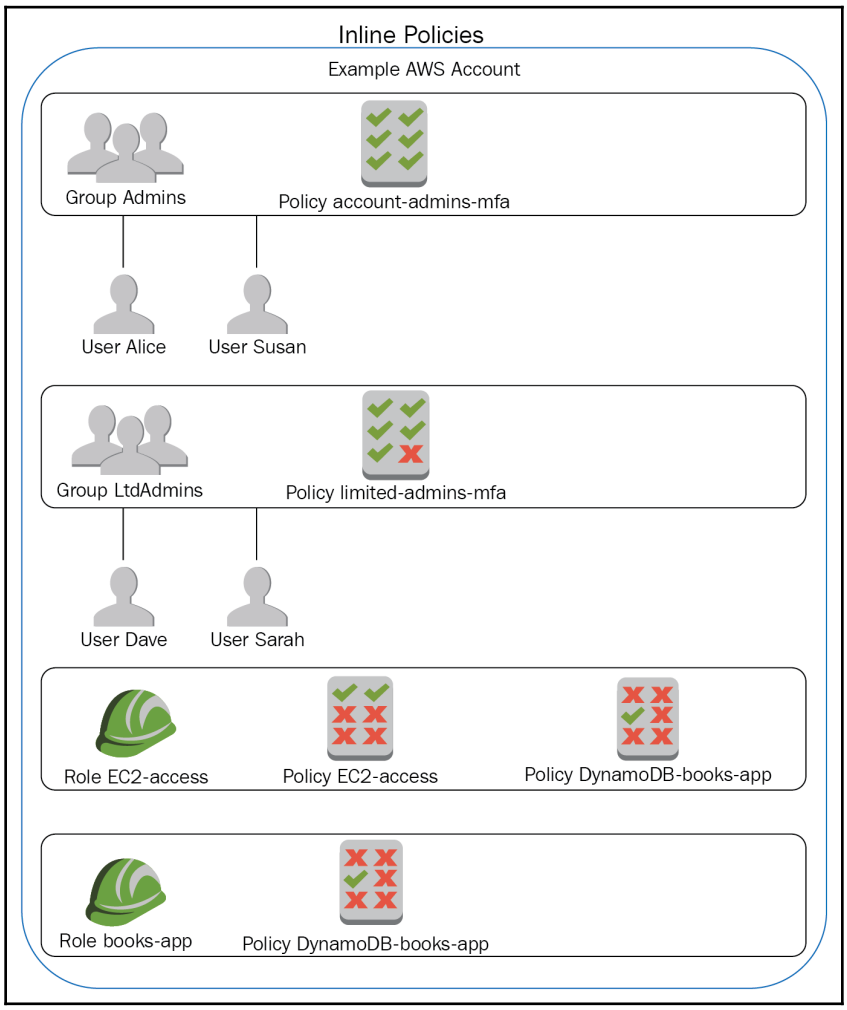
The following diagram shows details of how customer-managed policies can be attached to Principal entities:



Inline policies

Inline policies are the ones that we embedded into a resource. Unlike managed policies, these policies cannot be attached to multiple resources. If another resource needs the same policy, we must make a copy of the policy and embed it into the resource. We can embed the policy into the resource when it is being created, or later on.

The following diagram shows you the inline policies that are applied to various resources:



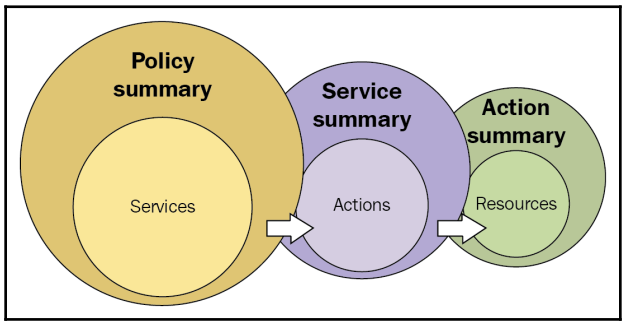
Summary tables

There are three important tables that summarize a policy. The tables are as follows:

- Policy summary tables
- Service summary tables
- Action summary tables

The policy summary table has a list of services. Once we select a service, we get the service summary table. The summary tables list the actions and the permissions for the service. If you select an action, you get the action summary table. This table contains a list of resources and the conditions for the action.

This is illustrated in the following diagram:



Summary

In this chapter, you learned about the IAM of AWS. You saw that there can be multiple identities, such as users, roles, and so on. You also saw that for authentication, IAM uses policies, and that there are various kinds of policies, such as identity-based policies, resource-based policies, and access control lists. We also explained that, apart from ACL policies, the policies are written as JSON documents. We looked at a few examples of policies.

In the next chapter, you will learn how to interconnect various cloud solutions.

12
Clustered Application Servers

In this chapter, we will illustrate how applications are clustered, using different cloud providers and frameworks. You can set up a lot of applications in a highly available way, and spread their workloads between the on-premises and cloud environments. You can also set them up in different cloud environments, if the technical requirements are set. In this chapter, we will describe solutions and how to implement them in such a case to be independent of one cloud vendor and do not run into a vendor login.

The following topics will be covered in this chapter:

- Requirements for cross-cloud application servers
- Examples of clustered application servers
- Managing multi-cloud virtual machines for clustered application servers
- Troubleshooting cross-cloud application servers

Technical requirements for cross-cloud application servers

To design a cross-cloud application server environment, some requirements need to be fulfilled, as follows:

- Network connectivity between the different clouds
- Single identity management solutions for all servers
- Supported applications for georedundancy

Looking into these points a little bit deeper, you can see figure out that most of them were already discussed in earlier chapters. However, we will briefly take a look at these points again.

Networking connectivity between different clouds

No matter what cloud you need to connect to, networking and networking security are always the key. This means that you will need reliable and secured networking connectivity, as there is a possibility that not all of the traffic is encrypted, depending on the software architecture for high availability.

As the Windows cluster environment basically has all of the nodes in one physical network location, each newer release can work with the nodes in physically different locations. So, for example, one part of the server nodes could be in cloud A, and the other one in cloud B. The software requirements are set for applications using different cloud vendors.

As each cloud vendor is running different connection gateways, the most common solution is to have the same network virtual appliance as a VM instance (single, or even double redundant) in each environment, and to design each cloud as an independent data center location. There is no requirement to have direct connectivity (using MPLS) or remote connectivity, but you will need to make sure that the network package round-trips are as quick as possible.

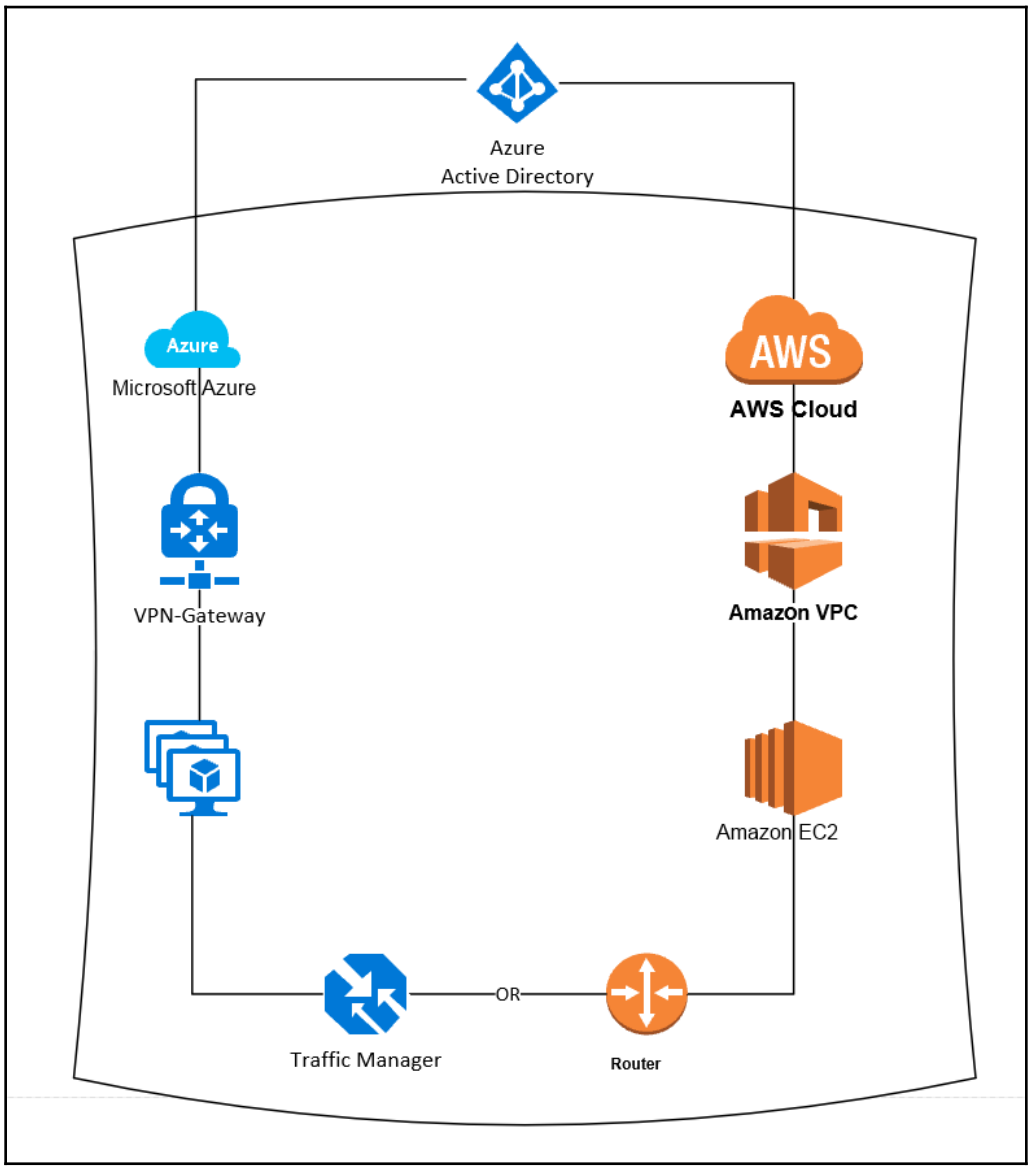
Single identity management solutions for all servers

The second pillar of cross-platform application designs is the IDM solution. As not every cloud vendor offers managed IDM solutions, the valid options are setting them up, for example, Active Directory as VMs and joining all servers in the Cloud into these domain or using a managed IDM solution like Azure AD which is not only support being used in Azure, but also works in other public cloud environments to have all servers join the one Microsoft Azure AD.

Supported applications for georedundancy

Finally, the application itself needs to support georedundancy in its node placement design. The application needs to be designed to work in low latency environments (for example, Microsoft SQL Server or Microsoft Exchange Server); it can also be designed quite easily, by placing a georedundant load balancer (for example, a web server).

A sample design of a cross-cloud virtual machine running clustered application nodes is as follows:



The preceding diagram shows Azure on the left and AWS on the right, both connected to a single-network environment and using Azure AD as a single identity management solution. Whether you choose the left one or the right one for your preferred cloud vendor, the result will be the same, and the service will be reachable from both cloud environments.

Examples of clustered application servers

Now that we have discussed the technical basics, let's take a look at some sample application servers (based on virtual machines) in a multi-cloud design. The samples will be as follows:

- Microsoft SQL Server
- Microsoft Exchange Server
- Web servers

Microsoft SQL Server

Microsoft SQL Server is a robust relational database application server that can run either based on Microsoft Windows Server or in Linux, as an operating system (starting with the 2017 version). Surely, you could set up a single SQL server as a virtual machine in your preferred cloud, we need to look at the high availability features of the application.

With Microsoft SQL Server, the basic keyword for HA is **availability groups**. Using this feature, you can set up your application server to have one or more replicas of the database itself, organized in an availability group. You can design the availability groups based on your needs, and split them between servers, no matter where the virtual machine really lives; then, you can configure database replication. One availability group supports one primary, and up to eight secondary, databases.

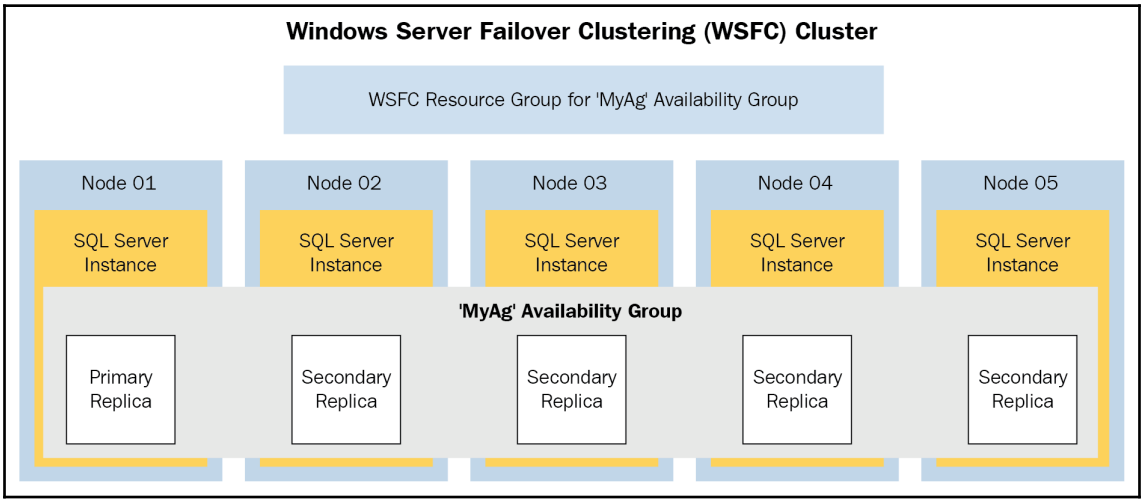
Keep in mind that a secondary database is not equal to a database backup, as the replica contains the same information that the primary database does.

Since the 2017 release, there have been two options for availability groups, as follows:

- **Always on for redundancy**: This means that if a database replica fails, one or more are still available for the application requesting the data.
- **Always on for read optimization**: This means that if an application needs to read from data, it uses its nearest database server. For write operations, the primary database replica needs to be available.

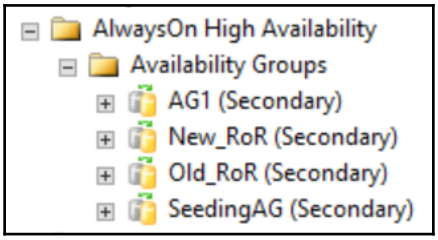
The replica operation itself can be synchronous or asynchronous, depending on the requirements and design of the application working with the database(s).

The following chart illustrates the availability group design of Microsoft SQL Servers:



As you can see in the preceding chart, there are different nodes available. Each of them can either reside in the same cloud, or a different one. Regarding risk management, a single cloud vendor can go offline without affecting the availability of your database servers themselves.

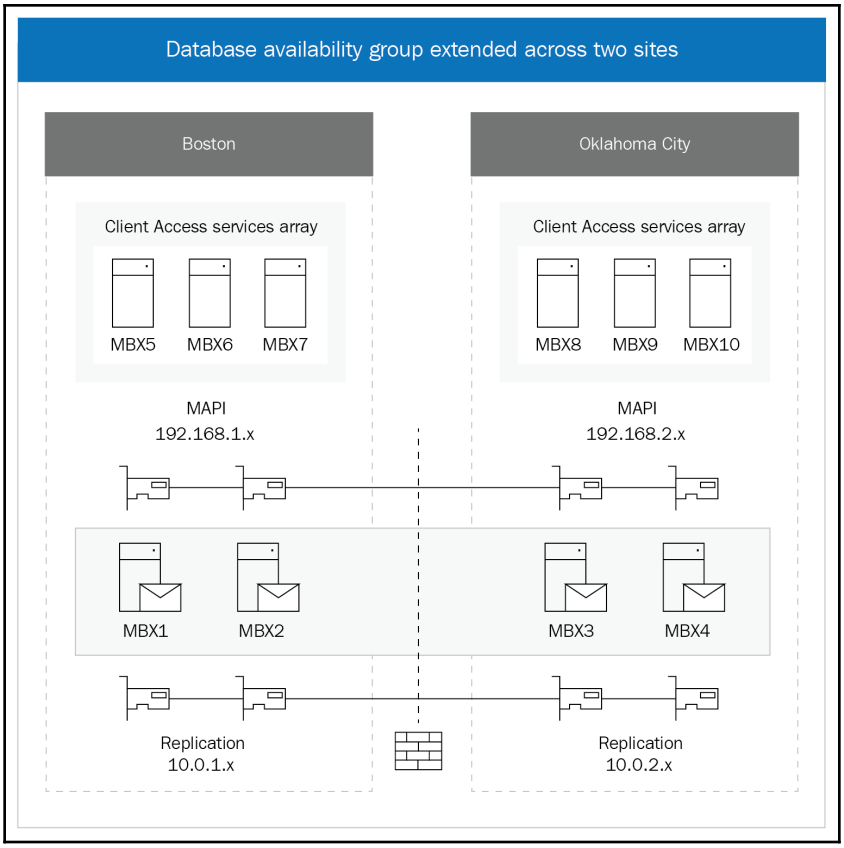
When you take a look at the SQL Server Management Studio, you will see the availability groups and their health statuses, as follows:



Microsoft Exchange Server

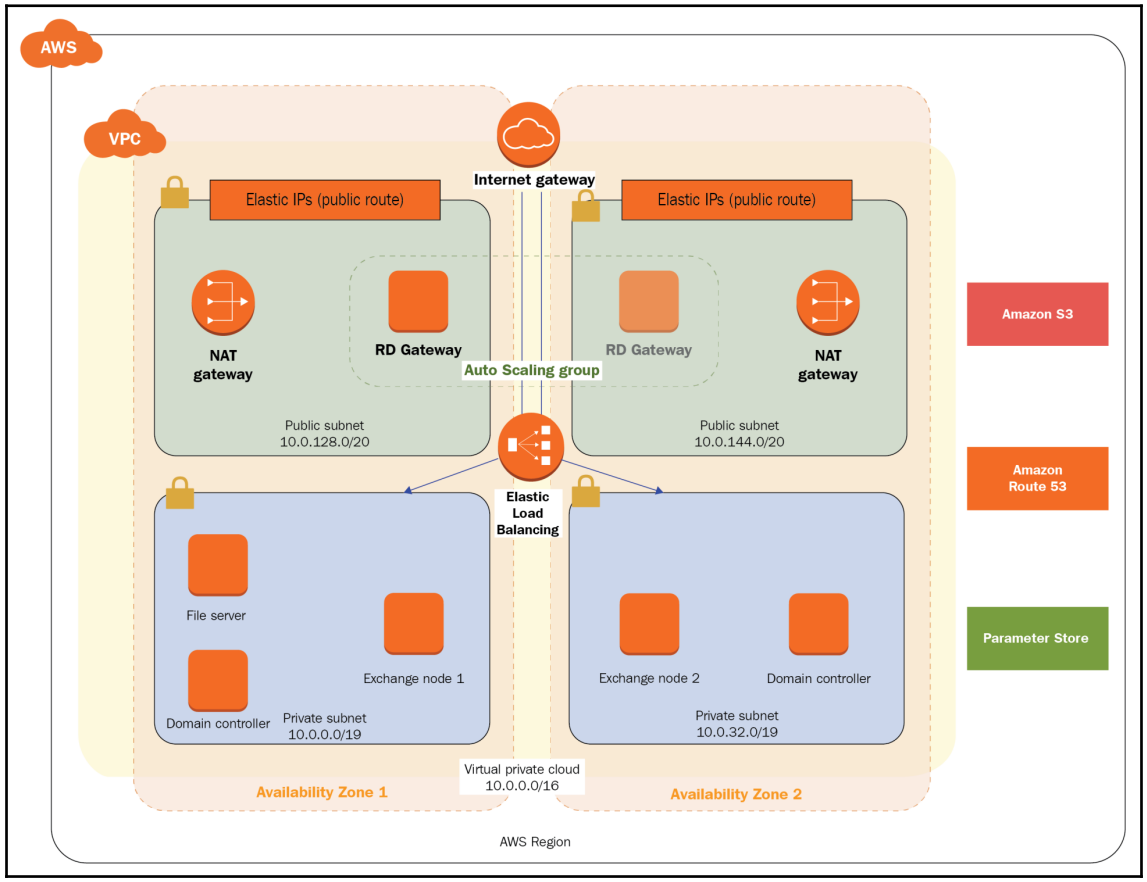
Microsoft Exchange Server is a groupware and email solution that is, in Microsoft cloud technology, the technical basis of the Office 365 SaaS email functionality. If you decide to run Exchange Server on your own, the most recent release is Exchange Server 2019. And, in case it is needed, there is full support to run Exchange Server as virtual machines in your cloud environment.

Of course, it is possible to run a simple VM with all Exchange Services on it, but, like with almost every company groupware solution that requires high availability, there is the possibility to set up Exchange Server environments in a multi-server environment. As Exchange Server has to support low latency network environments by default, it supports running an Exchange Server environment in different networking regions. This means that you can set up some nodes of Exchange Server in cloud A, and the others in cloud B. This feature is called availability groups, too. A typical design would look as follows:

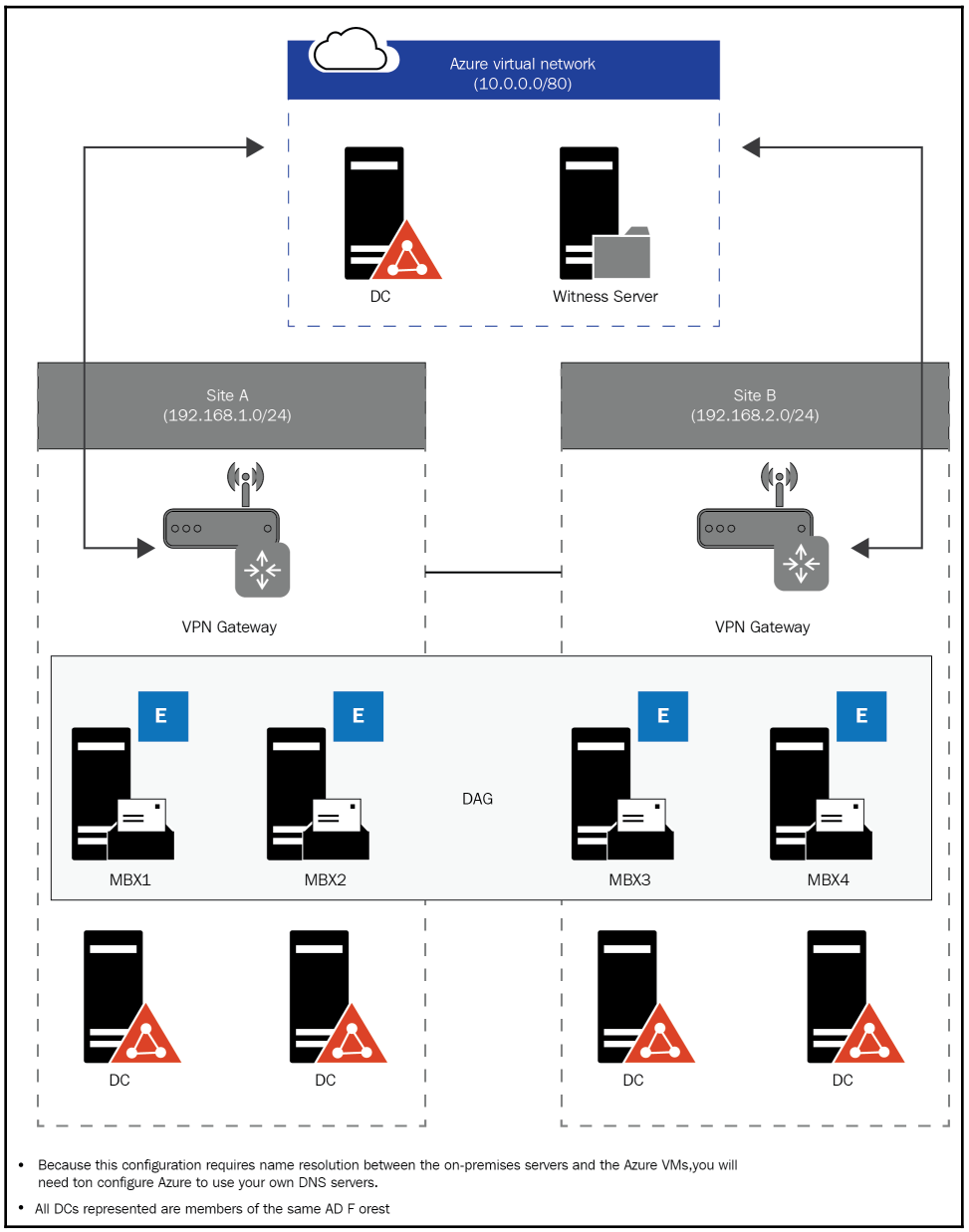


As you can see, the preceding diagram shows two Azure AD sites with redundancy, using **Database Availability Groups** (**DAGs**).

The best practice for running Exchange Server on AWS is illustrated as follows:



There is also a best practice design for Azure in hybrid cloud environments; it looks as follows:



A primary database residing on one Exchange mailbox server could have up to 16 database copies. The placement of these copies is dependent on the customer requirements.

As you have seen, configuring high availability within Exchange Server is the default, and it is quite easy to handle, too, from either the Exchange Server Management Console or from PowerShell.

Supporting cross-cloud implementations using geo load balancers

If an application that needs to be redesigned for a multi-cloud environment works based on port communications, the redesign will be quite easy, as you will just need to set up a georedundant load balancer to support the different cloud targets, and route the traffic correspondingly.

A georedundant load balancer is a more complex solution; it's like a default load balancer that just routes the traffic between different servers in one region or cloud environment. It generally works with the same technology and uses DNS name resolutions for redundancy and traffic routing, but, in comparison to DNS round robin technologies, a load balancer knows the available targets for resolving requests, and can work with technologies such as IP range mapping.

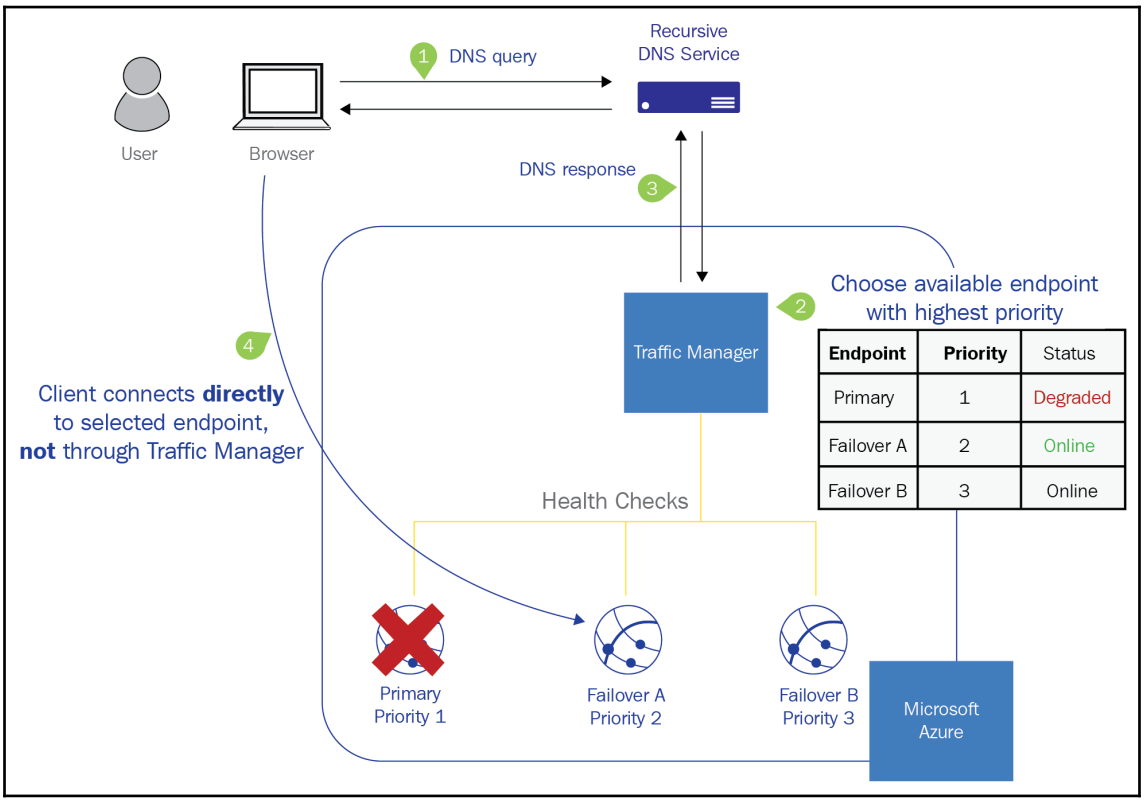
In the next part of this chapter, we will take a look at two solutions: Azure Traffic Manager and AWS Route 53.

Azure Traffic Manager

Azure Traffic Manager is the Microsoft solution for georedundant traffic routing. It is available in each Azure region, and it provides transparent load balancing for services that coexist in different Azure regions, non-Microsoft clouds, or on premises. It provides the following features:

- Flexible traffic routing
- Reduced application downtime
- Improve performance and content delivery
- Distributed traffic over multiple locations
- Support for all available cloud solutions (private and public clouds)

As you can see in the following diagram, Azure Traffic Manager is a flexible solution for traffic routing, and can point to each target that you need in your application design:

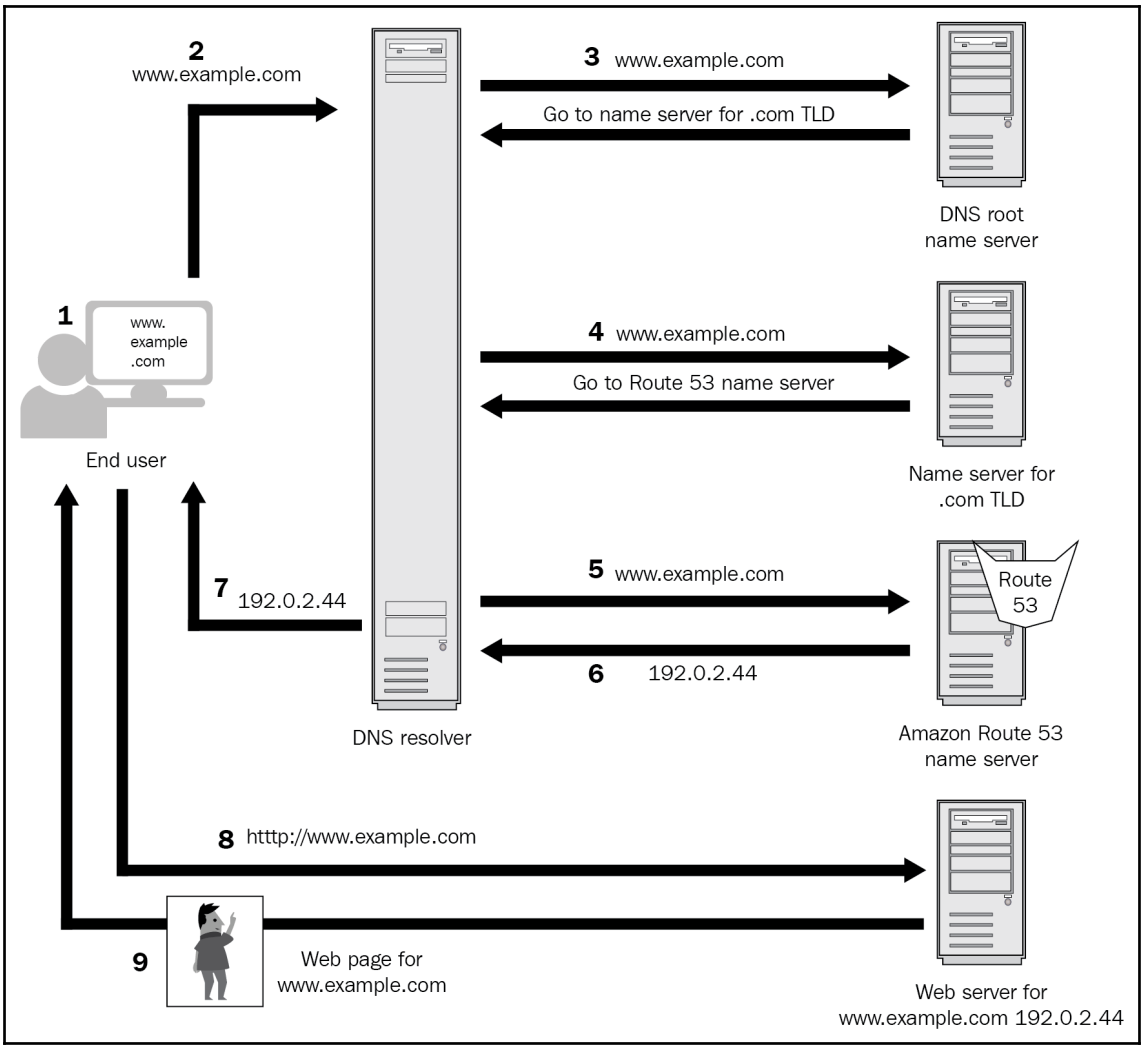


Incoming traffic is routed to the appropriate site using Traffic Manager metrics, and if a site is down or degraded, Traffic Manager routes the traffic to another available site. You can compare Traffic Manager to an intelligent router that knows the origin of the traffic and reroutes the requests to the nearest available service.

AWS Route 53

In AWS, the Route 53 service provides an easy solution for routing traffic based on load and availability. It is a PaaS service, like Azure Traffic Manager, and works based on DNS name resolutions, too.

The technical design works as follows; it is fully integrated into the DNS service:



As you can see, the Route 53 design is quite comparable to Azure Traffic Manager.

If you need to decide which service to use in your design, it is not a technical decision at all, as the technology is nearly the same. Rather, the choice is based on other requirements, involving technology and pricing.

Managing multi-cloud virtual machines for clustered application servers

If you decide to design your applications in a multi-cloud environment, it does not facilitate designing automation and replaying configurations. Azure works with ARM templates and AWS with AWS CloudFormation. Both languages are JSON based, but they are different.

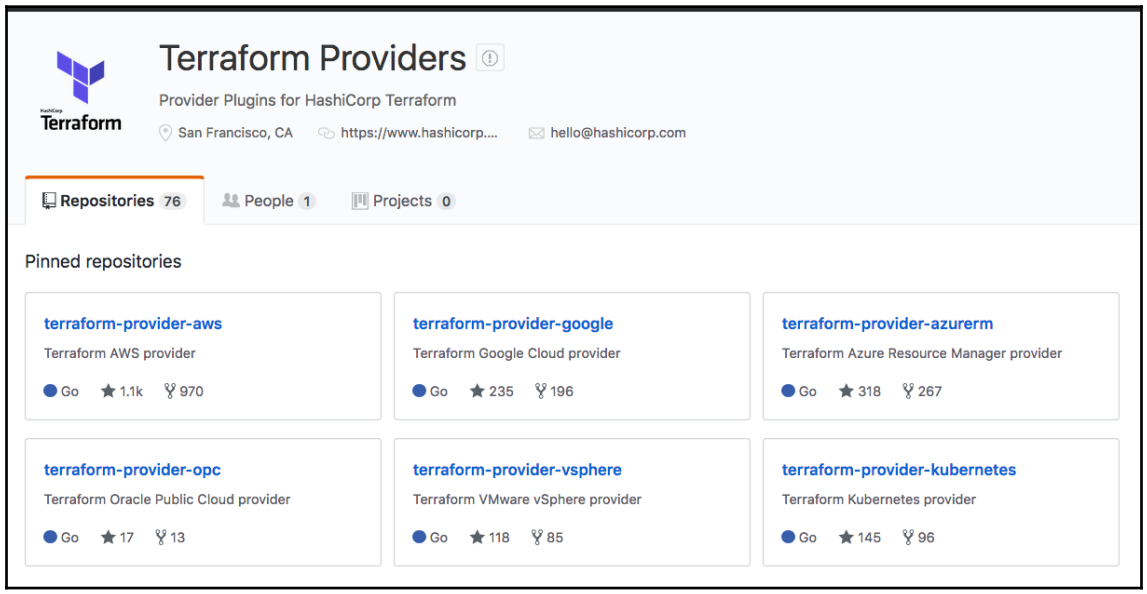
If you plan to use cloud solutions to transform your on-premises solutions, you should think about automation and ways to replay configurations. If you need to deal with two (or even more) different dialects, you will need to set up a process to create and update the corresponding templates.

Therefore, implementing another layer of templating will be required, if you do not want to rely on manual processes. There is a very small number of vendors that provide the technology to avoid relying on different dialects. A common one is Terraform, but Ansible and Puppet are other options. As this book is not about orchestration tools, we will just take a look at the first one.

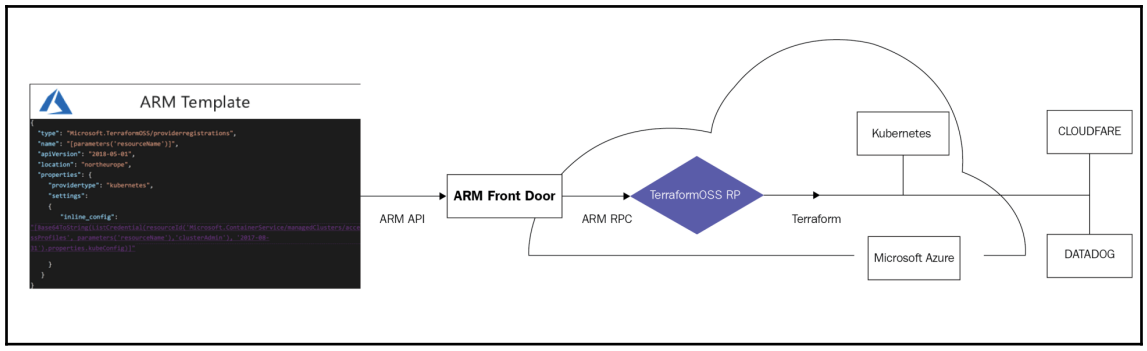
Terraform works based on a language called **Hashicorp Configuration Language** (**HCL**). It is designed for human consumption, so users can quickly interpret and understand the infrastructure configurations. HCL includes a full JSON parser for machine-generated configurations. If you compare HCL to the JSON language, it looks as follows:

```
# An AMI
variable "ami" {
        description = "the AMI to use"
}

/* A multi line comment. */
resource "aws_instance" "web" {
  ami                   = "${var.ami}"
  count                 = 2
  source_dest_check     = false

  connection {
     user = "root"
  }
}
```

Terraform gives us providers to translate the deployments into the corresponding cloud vendor languages. There are a lot of providers available, as you can see in the following screenshot:



A provider works as follows:



If you decide to work with Terraform to make your cloud automation processes smooth and independent of your cloud vendor, you can install it from most cloud vendor marketplaces as a virtual machine.

Troubleshooting cross-cloud application servers

If you have decided on a multi-cloud design for your applications, you will need to have a plan for troubleshooting; network connectivity and having identities between the different cloud environments could be reasons for unavailability issues. Otherwise, the troubleshooting mechanisms are the same ones that you're already familiar with, and they are included in the application servers themselves, in general.

Summary

As you learned in this chapter, if you follow the design guidelines for multi-cloud architectures, it is quite easy to design a multi-cloud environment. If there is a need to change the components in this solution, you can even decide to change services as a part of your solution from one cloud vendor to another.

13
Mixed IaaS, PaaS, and SaaS in Multi-Cloud Environment

This chapter will present a scenario that describes how to use solutions from different cloud providers and frameworks. You will learn how to interact with and create a design to fit into the requirements that will be as transparent as possible to the end customer. We will conclude the chapter by designing a real-world scenario with **Infrastructure as a Service** (**IaaS**), **Platform as a Service** (**PaaS**), and **Software as a Service** (**SaaS**), in multi-cloud environments (private, public, and hybrid).

The following topics will be covered in this chapter:

- Design guidelines for multi-cloud solutions
- Defining cloud services
- Containerization

Design guidelines for multi-cloud solutions

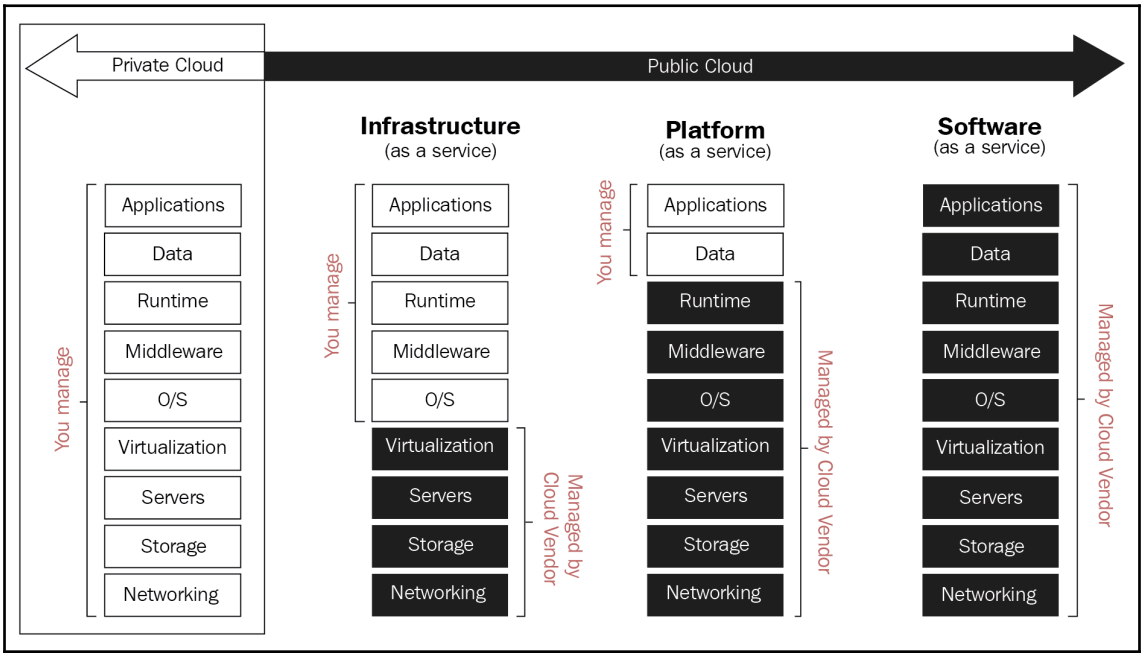
To design a multi-cloud environment, some requirements need to be fulfilled, as follows:

- Network connectivity between the different clouds
- A single identity management solution for all servers
- Supported application for georedundancy

Once these requirements are set properly, you can start your cloud service solution design.

Definition of cloud services

Looking at the various cloud service providers, we can categorize their services by different qualities, as shown in the following diagram:



In the past, the private cloud was quite famous: you were responsible for everything. With the public cloud, that changed. Depending on which solution you choose, the cloud vendor will be responsible for a few of the IaaS, more PaaS, or all SaaS parts of the services. We will look into this in detail in the following sections.

Infrastructure as a Service

The easiest way to use cloud services is to move the virtual machines to a public cloud vendor and run them as is. This is also called **lift and shift**. Everything that goes to the hypervisor is the responsibility of the cloud vendor; your responsibility starts at the VM itself.

In terms of flexibility, IaaS is a nearly 100% flexible, as you can install the operating system that you want with the services/daemons and applications that you need. You are responsible for everything. You will also need to take care of patching and security fixing. You should be aware that this is the most expensive cloud service.

Platform as a Service

With PaaS, there is no need for you, as a customer, to decide on the operating system. With the application release, your service is up and running. You just decide to use a defined service, with defined SLAs (technical and operational) and use it from a marketplace. High availability and redundancy are there by default and scaling up and down is possible whenever you need it. Pricing is mostly based on the amount and quality.

Software as a Service

With SaaS, there is no need for managing the underlying server technology itself. The most important thing is the web frontend. As a customer, you only have to care about the client applications themselves.

Patching, security, high availability, and redundancy are not your responsibility. You can just use the service itself.

This service reduces the amount of administrative tasks on-site, as there are limited tasks to be fulfilled by your employees. In general, you pay per user, and you can easily scale with this.

Containerization

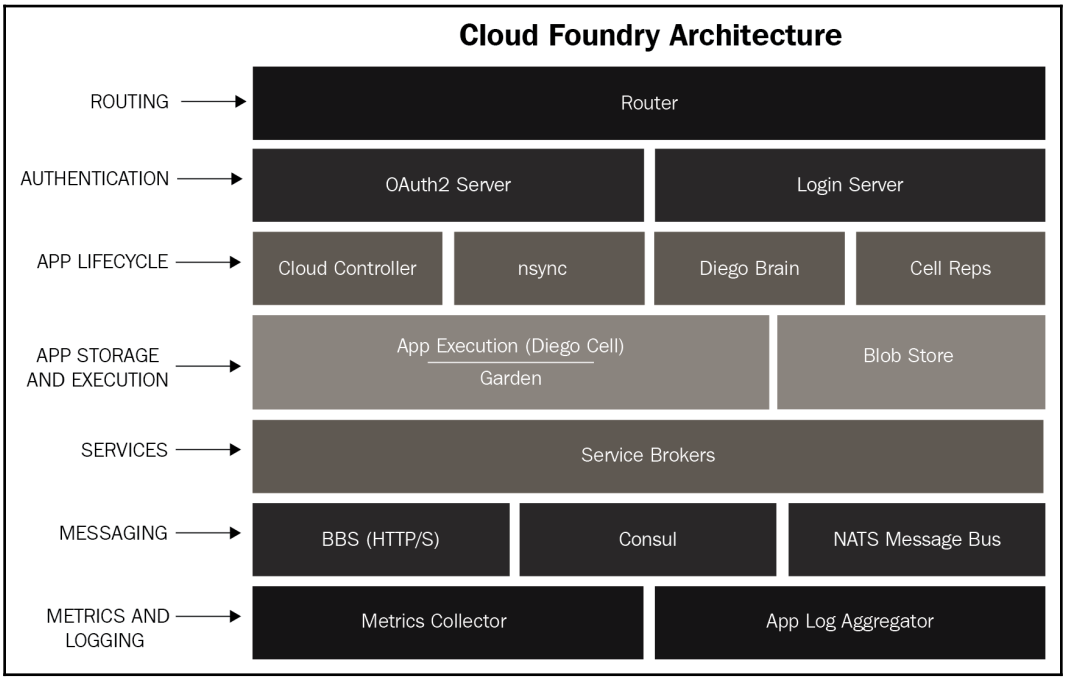
As virtual machines are complex and not easy to manage, there is often a requirement to bring in flexibility to custom IT services. This is where containerization comes into play. The concept of containers is defined as follows: a container is an infrastructure independent sandbox running in a container environment, without an operating system in the container. You can easily move containers between environments, and you can scale by adding another container to an existing environment.

Items that are usually bundled into a container include the following:

- Applications
- Dependencies
- Libraries
- Binaries
- Configuration files
- Container services have been an approach of different solutions in the past and even came from the community. In the meantime, the following container solutions survived.

Cloud Foundry

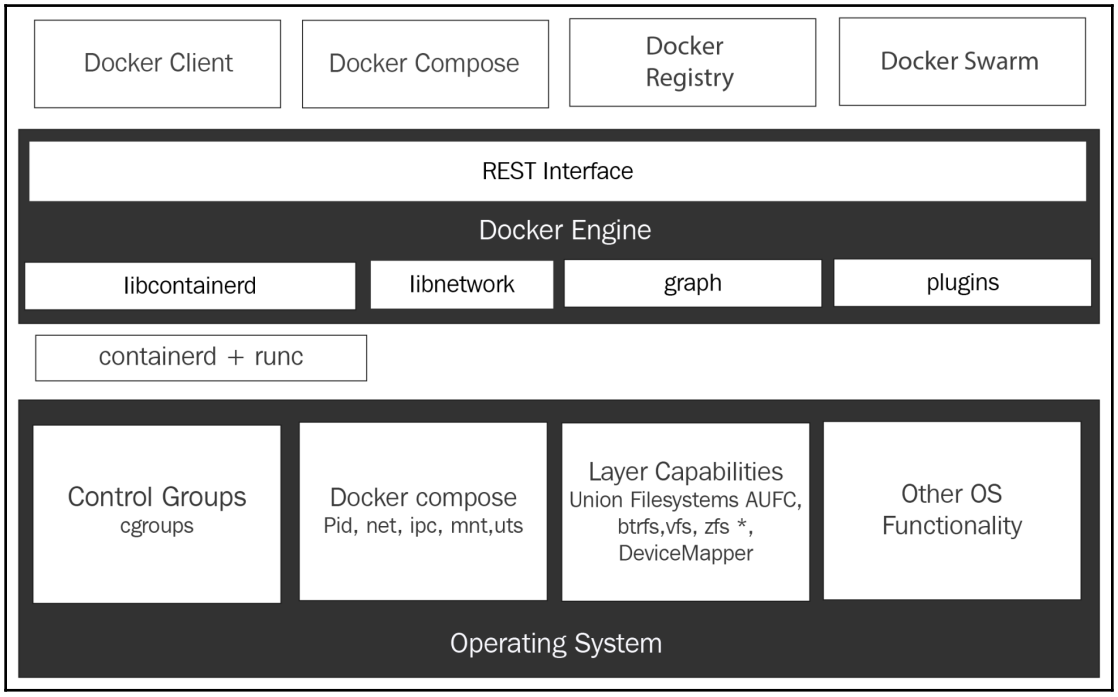
Cloud Foundry was mainly developed by VMware and later by Pivotal. As you can see in the following diagram, the basic design is easy to understand. There are different layers of services that are split to scale and communicate between them:



Cloud Foundry provides the ability to run containers independent of the underlying programming language or framework. It provides so-called service brokers that provide a defined connection to existing PaaS solutions from cloud vendors (for example, MySQL from Azure, DB2 from AWS, and so on). It is completely integrated into the CI/CD pipeline of development, and therefore, it has a lot of users from the DevOps parties.

Docker

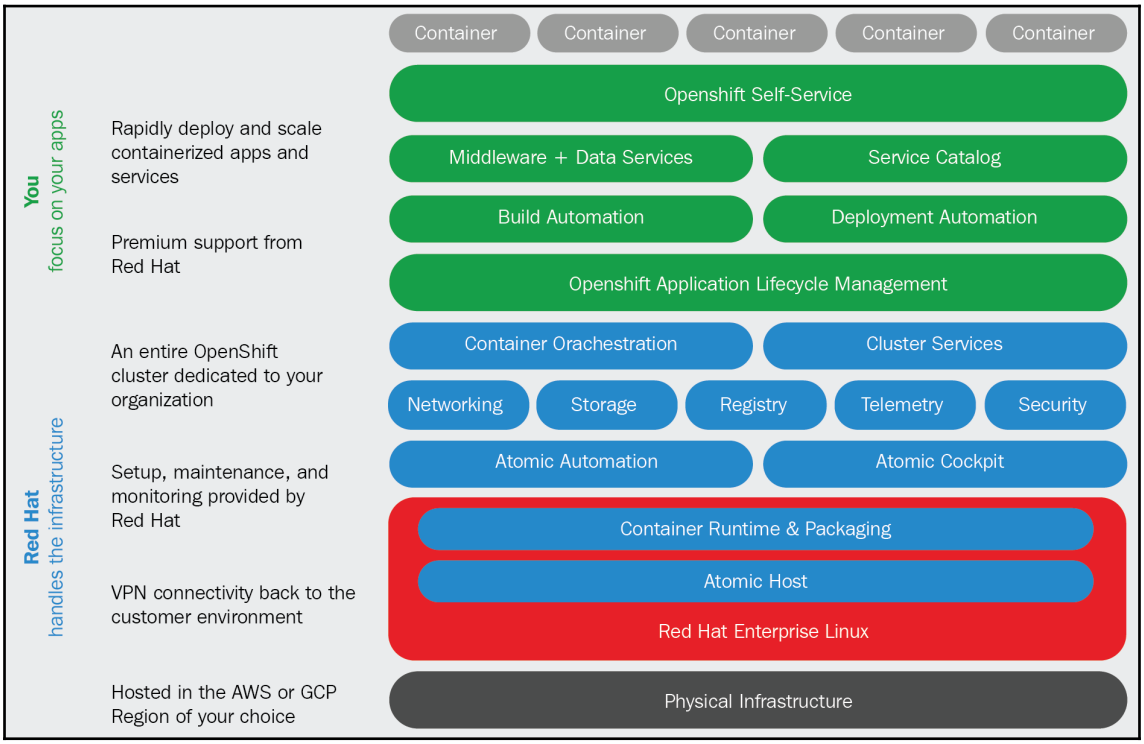
Docker is a software solution that works based on containers. A container is defined as an isolated environment containing applications, tools, configuration files, and libraries. All of them run on a single operating system Kernel without guest operating systems, and we know from virtual machines. If you run container in scale, you will need to define an orchestration solution. In today's Docker environments provided by public cloud vendors, you will find Kubernetes as the management, orchestration, and scaling solution shown as follows:



As you can see in the preceding diagram, there are different layers that ensure that the **Docker Engine** can communicate to the rest of the services and provide a defined API, internally and externally. Each of these layers and design parts is responsible for an important part of the product.

OpenShift

OpenShift is a container solution, with Kubernetes as the orchestrator, that runs on the RedHat operating system. It is owned by IBM:

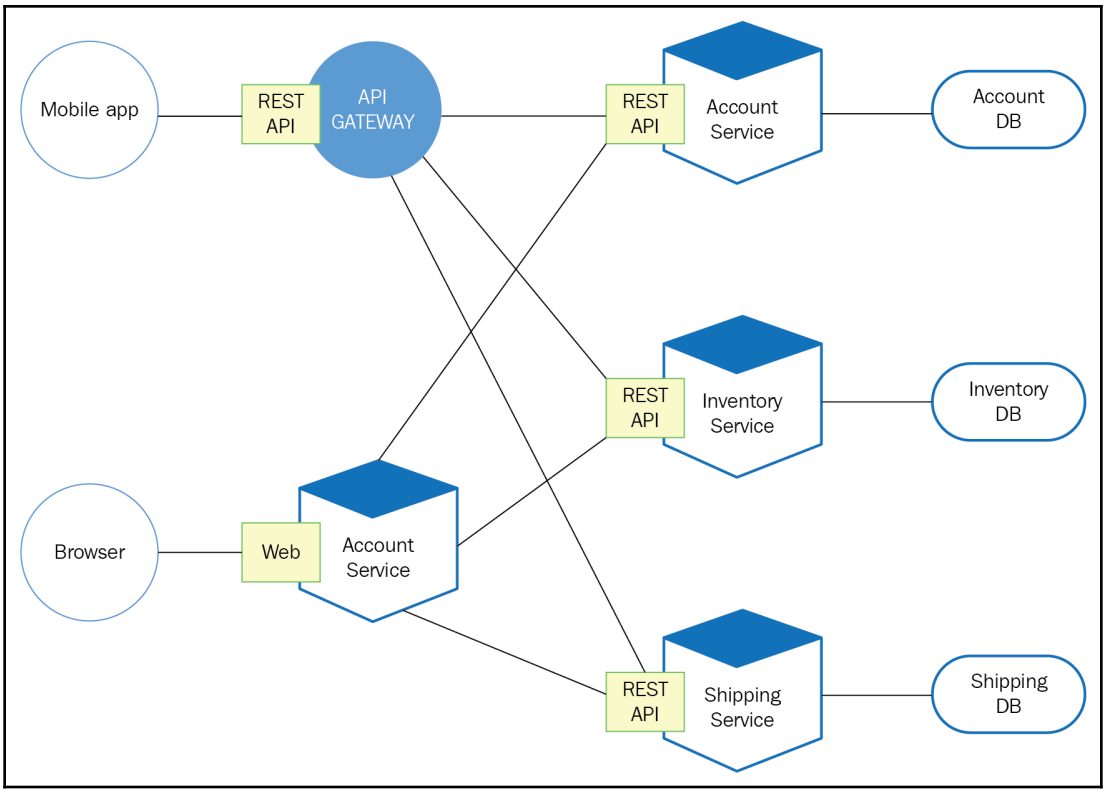


OpenShift is comparable to Docker itself, but it has some modifications that were designed by RedHat. They have been explicitly implemented into the OS itself.

Microservices

The technical basis for working with containers is a microservice application architecture. This means that each application should be sliced into the smallest possible (but scalable) services. These services are then moved to containers. To scale a specific part of an application, another container is switched on, and a load balancer, sitting before the microservices container, is responsible for integrating the new container into the application life cycle.

The concept of microservices is illustrated in the following diagram:



As you can see in the preceding diagram, there is an application frontend, API services, and a background database in this sample design. So we have services talking to the user and acting as the frontend. There is a layer in between for communication and translation and we can find a third layer, which is the database service itself.

Docker Services as a Service

Depending on the cloud vendor, you will find some, or even all, of the following services as PaaS offerings:

- Container registries for hosting the container images
- Container hosts/instances for running the container images
- Container orchestrators (generally based on Kubernetes) to orchestrate the images

Regardless of the overall design of your cloud solution, you will be able to integrate these from different cloud vendors. This means that you can spread the same containers to different clouds, hosted by different vendors, and decide where you would like to place your registry, and where your Kubernetes orchestrator should have its home address.

Best practices

Regarding best practices for your PaaS container design, you should make sure that you find all the required parts for a solution at one or many public cloud vendors. If we set the networking connectivity with good performance and low latency, for example, there will be no need to, place our container hosts in different environments to provide better availability for customers consuming the cloud services. For example, if a customer is consuming his cloud services from Azure, it could connect to your service within the Azure backbone. If it comes from Amazon Web Services, this may be its connectivity target.

Bear in mind that redundancy requirements cannot be solved with more complexity. Even here, Terraforms can help to design a descriptive language that is cloud independent.

Best practices

PaaS services, in comparison to IaaS, do not need the customer to manage and service the underlying infrastructure. The service is designed for scaling and high availability, by default. From a pricing perspective, a PaaS service is much cheaper than a dedicated IaaS service.

Looking at the design of PaaS services, you will find that a lot of them are already container-based but may not be multi-tenant at all. A PaaS solution can even be 1:1 for the relationship of the container and the customer.

Keep all of the points from the previous chapters. The real challenge to design a cloud-based and auto-scale solution is to figure out the service components you need and decide from which cloud vendor you would like to consume it. If the technical requirements for multi-cloud designs are set, you will just need to combine the services to best fit your requirements. In general, you should follow the the guidelines in the following order: first SaaS, then PaaS, and finally, IaaS.

A real-world design scenario

Now let's take a look at a real-world scenario that involves designing a multi-cloud architecture for a mid-size European company with locations in the United States, Japan, and Moscow, in addition to their worldwide headquarters in Munich. They are working in the medical and health care area and decided to completely move every IT service to the cloud, except their data center in Moscow, as their IT team is located there. But even for this location, their goal is to minimize the on-premises servers and even work in the data center environment using public cloud technology, as this would give them the chance to move the last on-premise servers to a public cloud, if somehow, in the future, their data center needs to move.

As of today, the company is running the following services:

- Active Directory on Windows Server 2012 R2 with four domain controllers' in the IT headquarters in Moscow.
- Each location has two domain controllers' on-premises.
- 480 member servers, running Windows Server 2008 and higher.
- 280 Ubuntu 17 servers.
- VMware, as a virtualization layer.
- Each server is a virtual machine; even their firewalls (checkpoint) are virtual machines.
- The company's network is MPLS-based, operated by AT&T. They have a central internet breakout in Moscow.
- There are about 500 SQL server instances running on Windows and Linux. Each of them is already in the most recent release.

- About 50 servers are running custom installations developed by the company's development team. They are using Visual Studio for managing their code.
- About 4200 client devices are being used. Each of them is running Windows 7 and Office 2010. For mobile devices, they are using Apple iPhones.
- The central solution for business services is SAP, which is currently hosted in their local data center in Germany, and in Moscow as a redundant hot standby environment.
- They are currently managing their environment by using system center 2012 R2 tools.
- Their ITSM solution is already running as a SaaS solution in the ServiceNow cloud. This is the only solution that will survive the redesign and even in 2021 and later will run in the ServiceNow cloud.

The CEO has decided to have a cloud-first strategy, and all of the cloud migration must be done by the end of 2021, as all the existing data center contracts would be cancelled by then. In addition, they have already decided to implement a two cloud vendor strategy using Microsoft Azure and Amazon Web Services. AWS should mainly be used by their development team, as the chief of development is a former employee of AWS. The head of IT has decided to move all of the infrastructure services mainly to Microsoft Azure.

Suppose that you are the responsible, external consultant, helping to design the new IT environment.

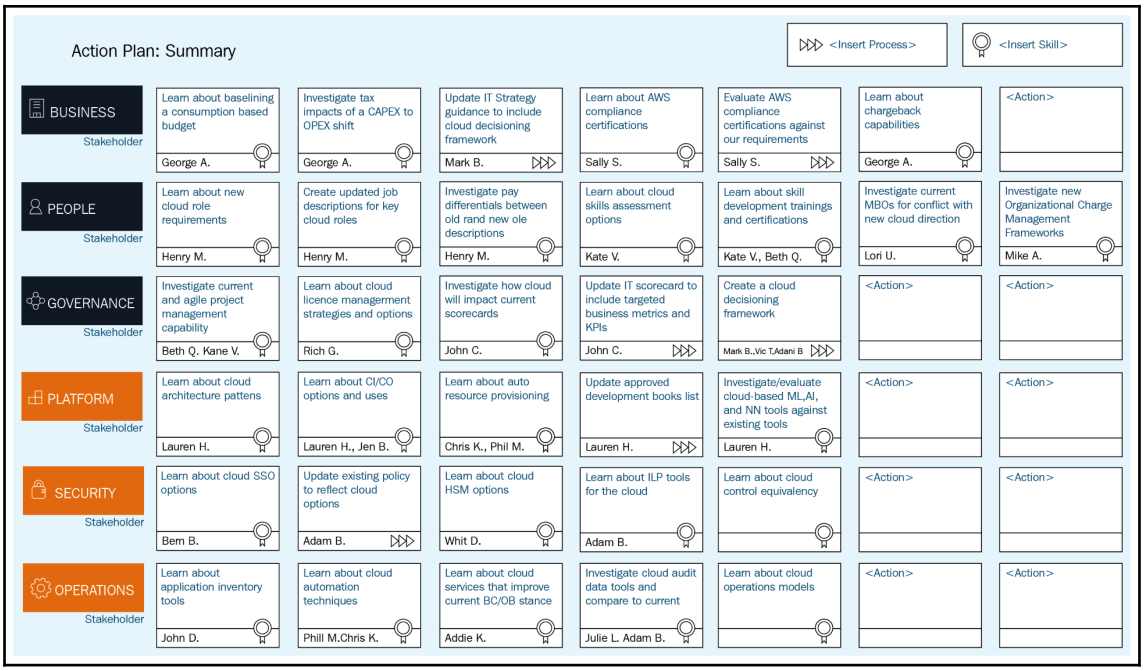
Solution design

This section will describe the project and the final design of the new company's IT environment, based on Azure and AWS technology.

Preparations

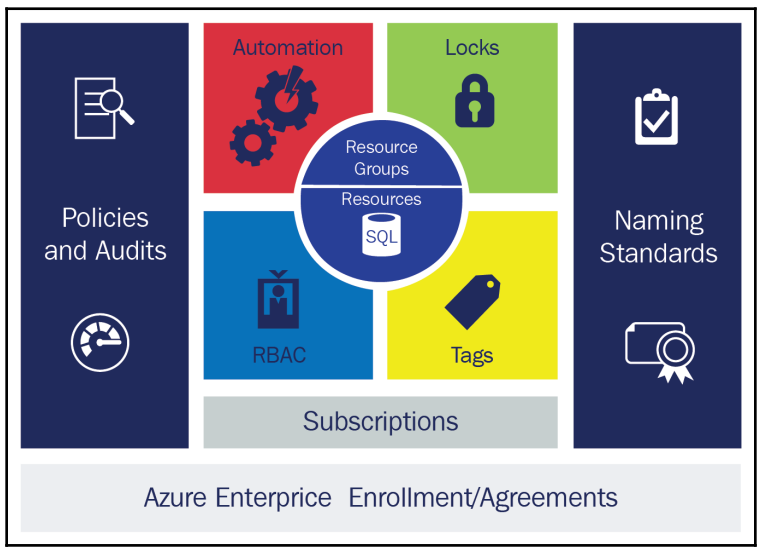
One of the most important steps, when starting to use cloud services is to define the cloud governance. Regarding which cloud vendor you decide to use, basically, they are all the same.

With Amazon Web Services, the Cloud Adoption Framework looks as follows:



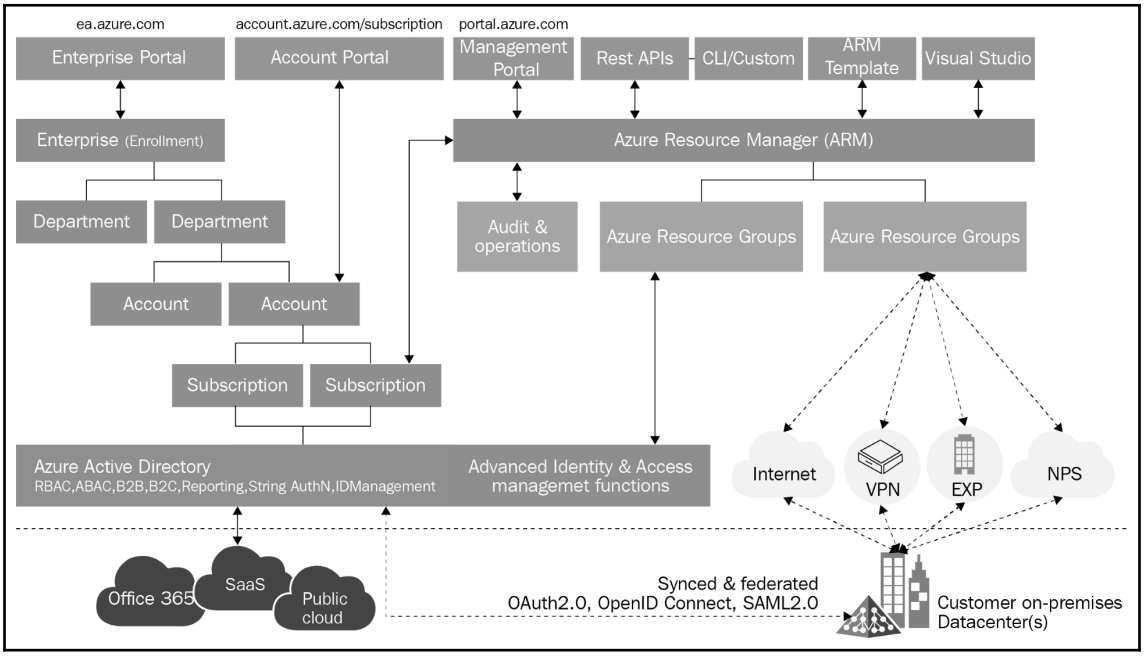
With AWS, as the customer, have to work through each of the points, in order to be happy with your cloud governance.

With Microsoft Azure Services, there is the Azure Governance Scaffold, as follows:



These are the main points that you will need to define your governance and work with Azure properly.

Looking at Azure in more detail, we will need to decide on a concept for the following components:



As you can see in the preceding diagram, there are different portals on the business side (the **Enterprise Portal** and the **Account Portal**), and then a third one to manage the technologies themselves (the **Management Portal**). If you would like to work with code (JSON), the APIS, and CLI, Visual Studio will be your ideal solution to work with.

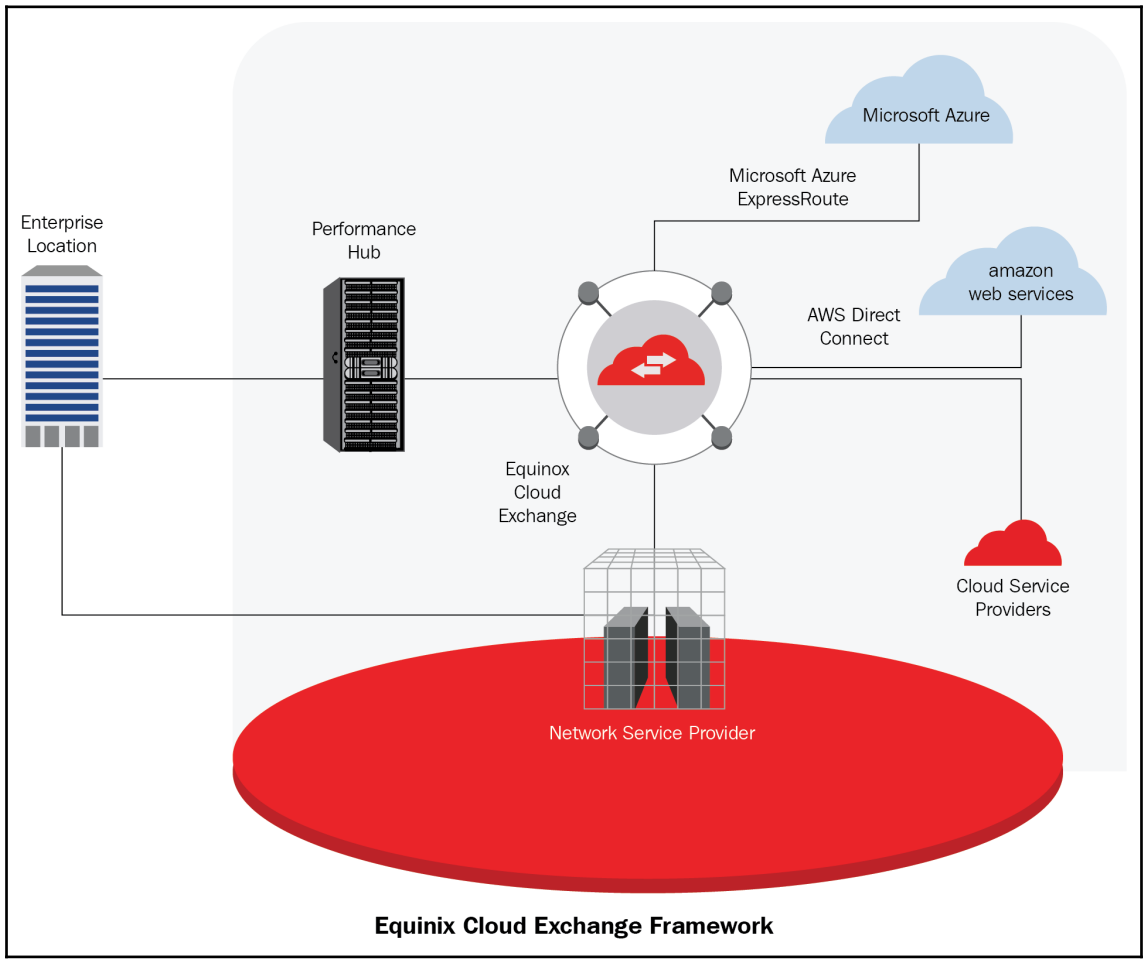
We will need to merge both of the cloud vendors.

For more details on Azure Governance, you should take a look at the following Packt Publishing book: `https://www.packtpub.com/virtualization-and-cloud/implementing-azure-solutions-second-edition`.

For this example, the governance has already been set and we can have a look at the technology itself.

Networking

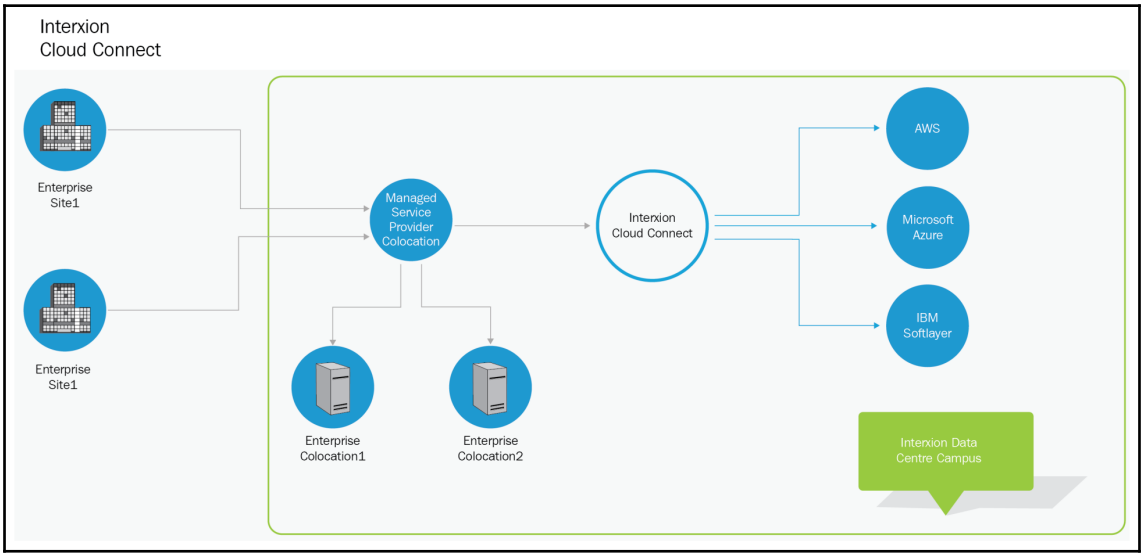
Looking at the networking portion of the multi-cloud design, the company decided to work with a partner that supports multi-cloud connections. This means that they, themselves, do not have to manage connectivity. This is being done via remote peering with their networking partner. Our company decided to go with Equinix and Interxion. The following diagram shows the **Equinix Cloud Exchange Framework**:



Equinix Cloud Exchange Framework

As you can see in the preceding diagram, the customer has connectivity to Equinix and Equinix will provide the customer with connectivity to the cloud vendors of your choice.

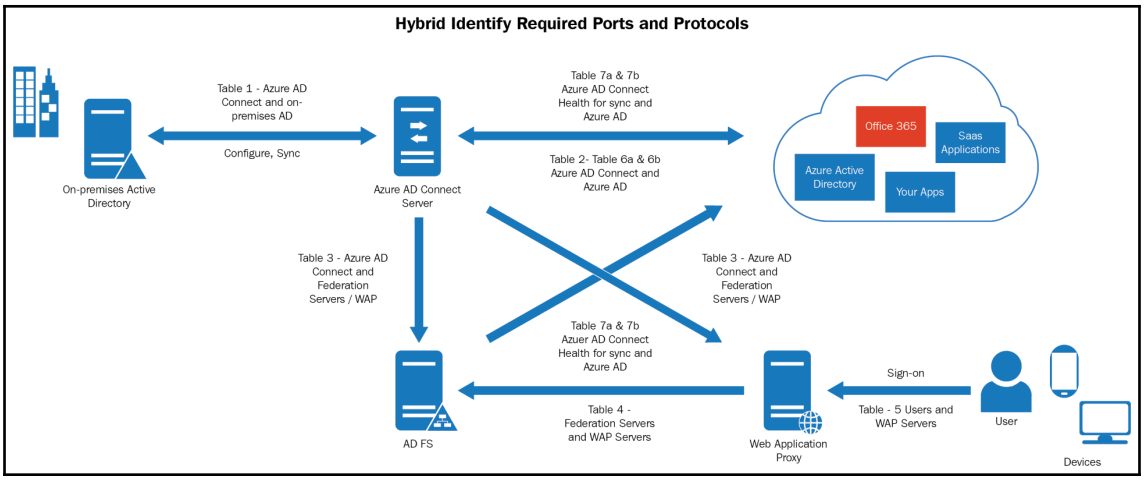
Let's take a look at Interxion:



Interxion works the same way that Equinix does, and it is another vendor to help you solve your multi-cloud networking configurations. The result will be redundancy and independency and even a connection to the local data center in Moscow without any issues to both cloud vendors.

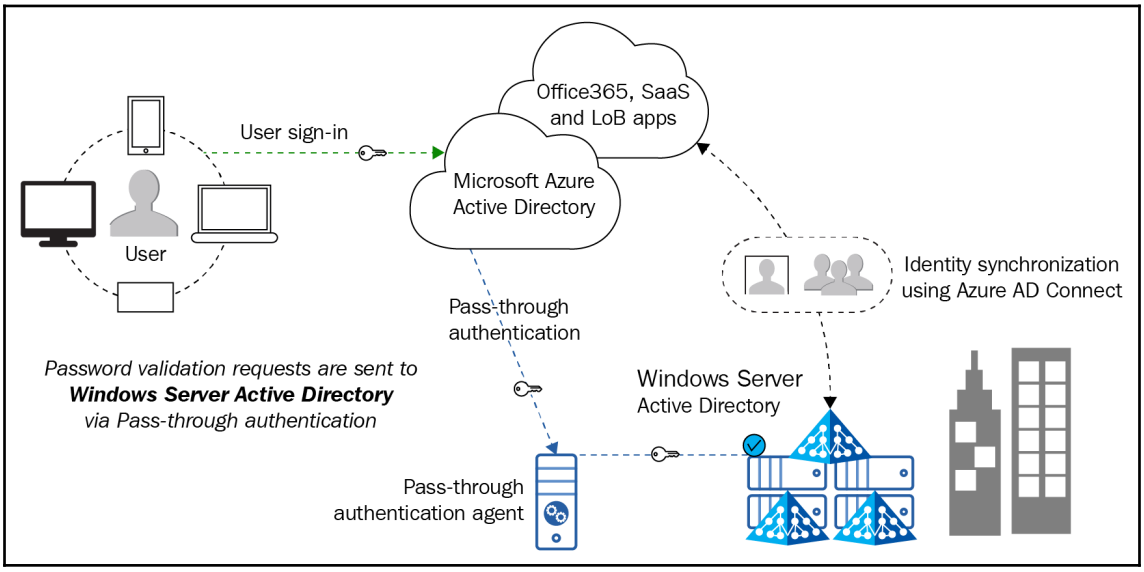
Identity management

The company has decided to have a single identity management solution based on the technology, they already run on both public clouds, which is Azure Active Directory:



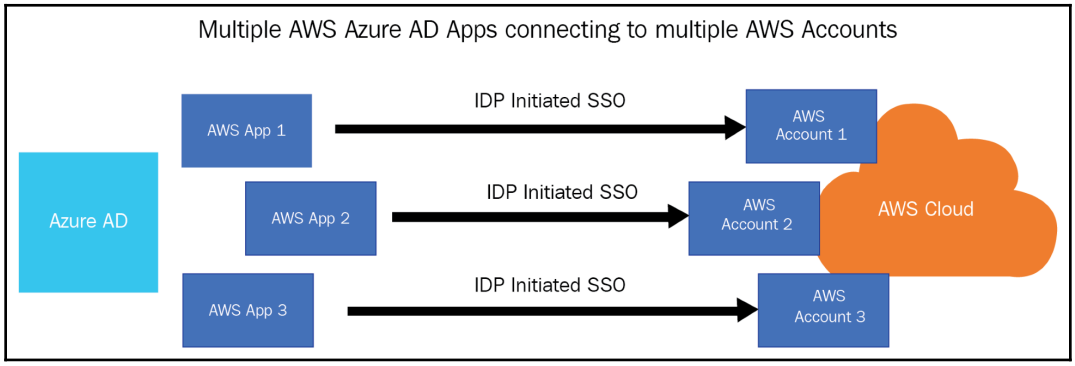
As you can see in the preceding diagram, each cloud service (either public or hybrid, using Azure Stack or a similar service) is using **Azure AD** as a single IDM solution.

Based on their security policies, the company has decided to go with Azure AD Connect, using pass through authentication (PTA):

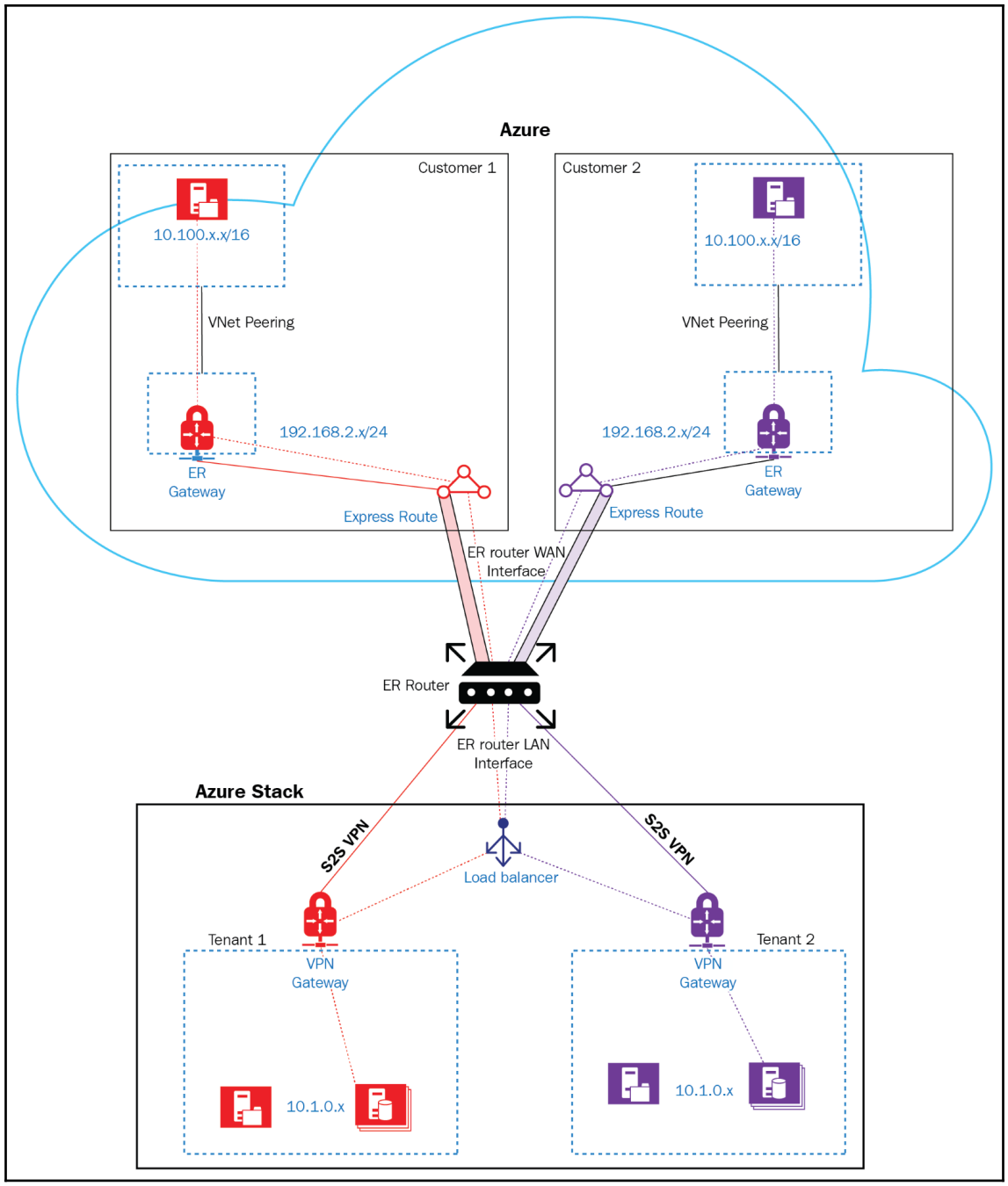


The PTA agent is monitoring the IDM queues in the cloud and authenticating the requests locallys transferring back the authentication token.

As Azure AD works with AWS, too, there is single identity management solution in place, as follows:



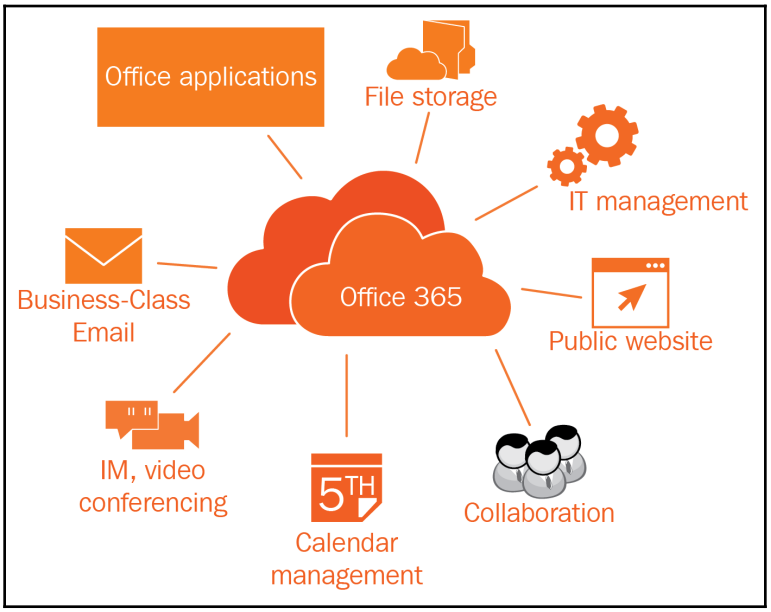
For their on-premises cloud environment, the company has decided to go with Azure Stack in a connected mode, in order to leverage Azure AD, too. The design is illustrated in the following diagram:


Azure
Customer 1
10.100.x.x/16
VNet Peering
192.168.2.x/24
ER
Gateway
Express Route
Customer 2
10.100.x.x/16
VNet Peering
192.168.2.x/24
ER
Gateway
Express Route
ER router WAN
Interface
ER Router
ER router LAN
Interface
Azure Stack
S2S VPN
Load balancer
S2S VPN
Tenant 1
VPN
Gateway
10.1.0.x
Tenant 2
VPN
Gateway
10.1.0.x

As you can see in the preceding diagram, Azure Stack and Azure behave the same way technologically, and can therefore be integrated into the express route configuration as another Azure cloud.

Modern workplace services

With the basic cloud environment in place, the company has decided to go with Office 365 for all workplace services, on all client devices. They will be migrated to Office applications on the client computers, using Office servers as SaaS directly from the cloud. This will be a seamless migration for the user accounts, and everything will work as usual, even when the migration is taking place:

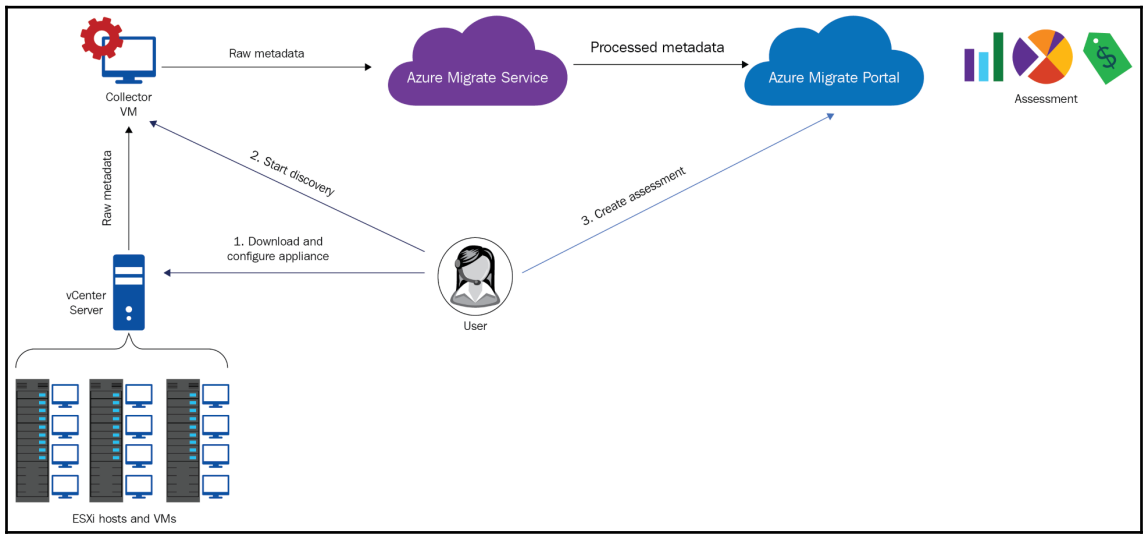


As the Office 365 license can also be used on iPhones, all of the employees will be fine. Regarding the company's on-premises exchange server, Skype, and SharePoint, they will move these to Office 365 completely, and will get rid of the virtual machines that are present today.

Infrastructure services

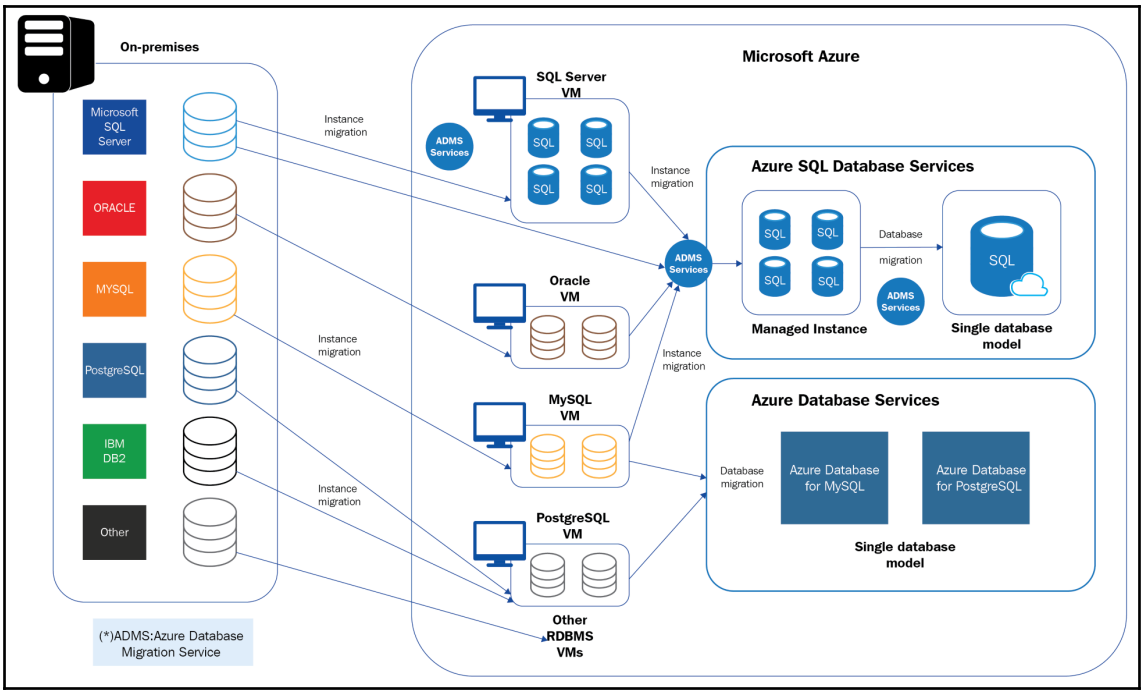
Regarding the existing infrastructure services, you have decided to move most of them to Microsoft Azure, and to prepare the migration by first identifying which solution can exist as a PaaS, and what needs to reside on a VM in IaaS.

To automatically collect all of the required information, you decide to perform an analysis using Azure Migrate, as follows:



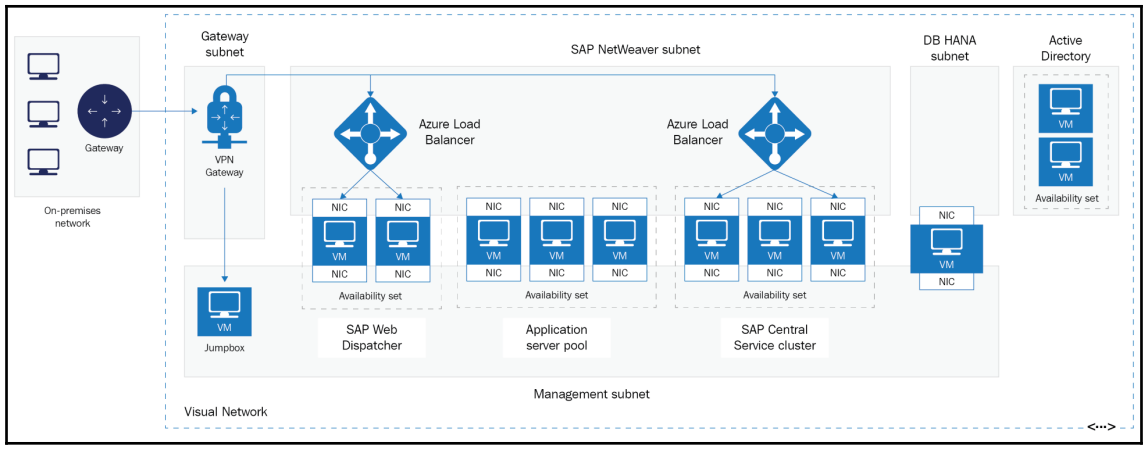
The **vCenter Service** will be connected to Azure and will host the migration service itself. I will be responsible for synchronizing, and later managing, the switch of each VM from on-premises to the cloud.

For all of the SQL services, there is a solution called Azure SQL Migrate, as follows:

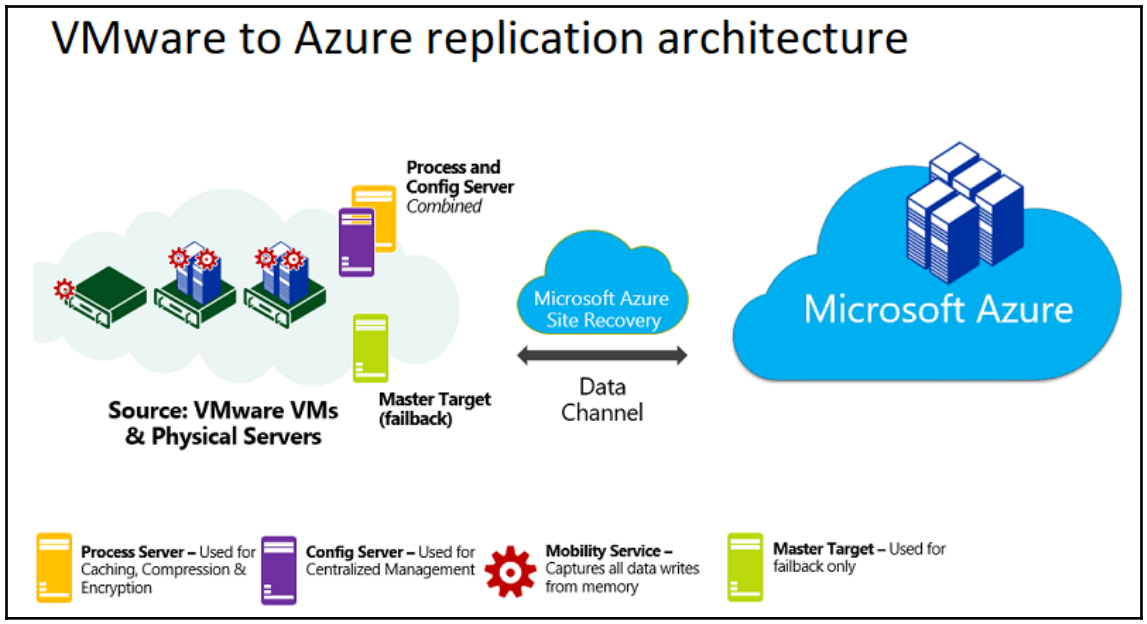


As a result of these cloud services, you will be able to figure out if there are still any virtual machines running SQL. In general, about 80-90% of the SQL servers in the existing design can be moved to PaaS solutions. Using the results of the migration service, you can get an idea of what the cloud SQL solution will look like. It will also help you to work through each migrating step in an informed way.

For the SAP environment that is currently running on-premises, you decide to migrate to SAP HEC on Azure, based on the existing blueprint design of Microsoft, as follows:



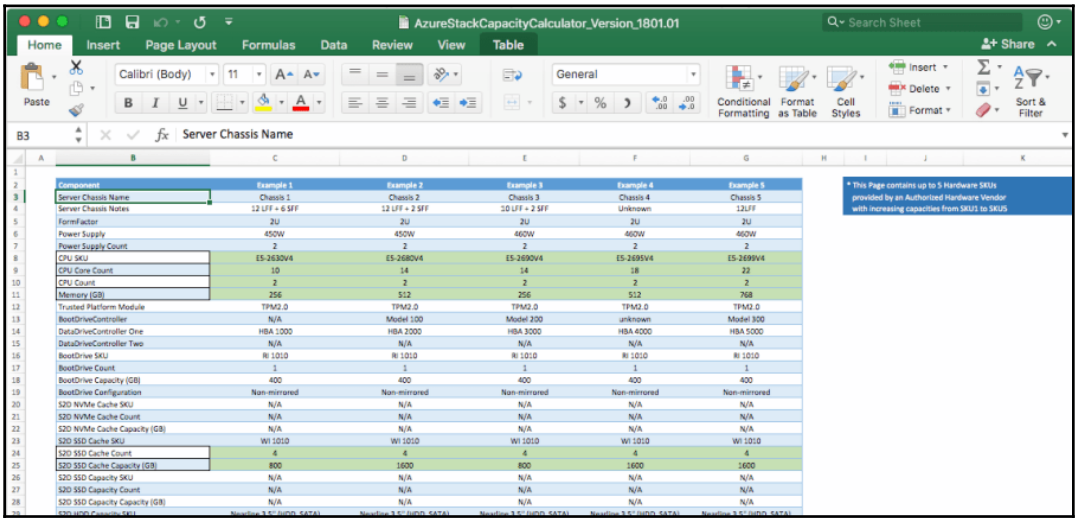
About 68% of the VMs can be migrated to Azure seamlessly, without requiring running them as virtual machines anymore. Another 25% of the VMs need to be migrated to Azure using the lift and shift model. The service that you will need to migrate them to Azure is Azure Site Recovery. This service works as follows:



For all of the VMs that need to run in the on-premises data centers that will be left after the move to the cloud, you decide to use Azure Stack. As Azure Stack is the on-premises solution of Azure, the process to migrate the VM is nearly the same. As the company's CEO has decided that a temporary placement of a virtual machine in a public Azure cloud for migration tasks is fine, you three-step migration:

1. Migrate the VM from the on-premises VMware to Azure, using Azure Site Recovery
2. Move the VM from Azure to Azure Stack, using Azure Storage Explorer
3. Bring the VM online again, on Azure Stack

From a sizing perspective, you decide to use an eight-node Azure Stack environment from the company's favorite hardware vendor. The sizing has been created using the Azure Stack Capacity planner. As you can see in the following screenshot, it is an Excel sheet with input and output parameters:



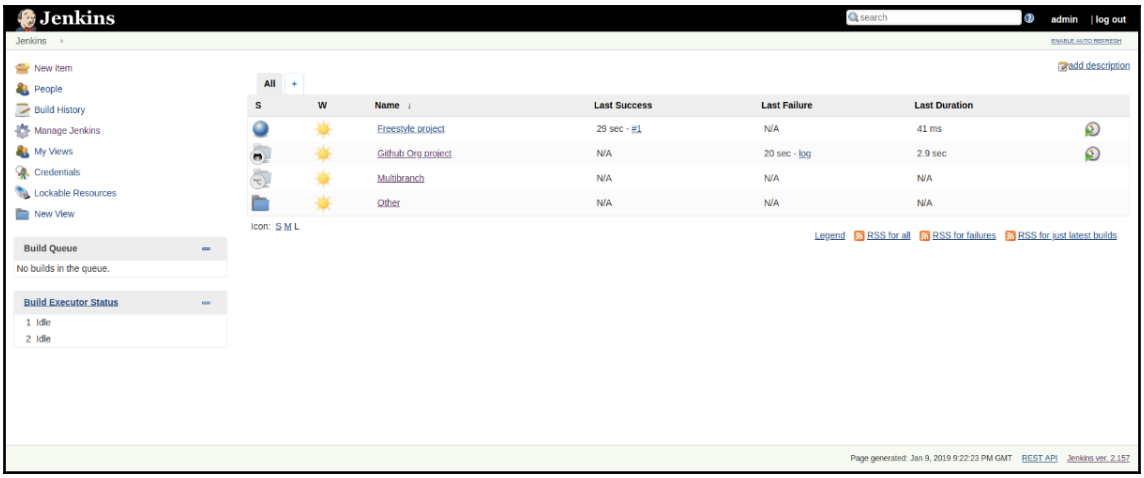
To download the spreadsheet, go to: `https://aka.ms/azstackcapacityplanner`.

Setting up new cloud services

For each new cloud service that will be deployed, the CIO has decided to go with Terraforms, in order to have a unique description language for all of the resources, regardless of the cloud flavor of a specific vendor. Terraforms provides an easy way to automate the deployment, and to be flexible when moving resources, even between the clouds.

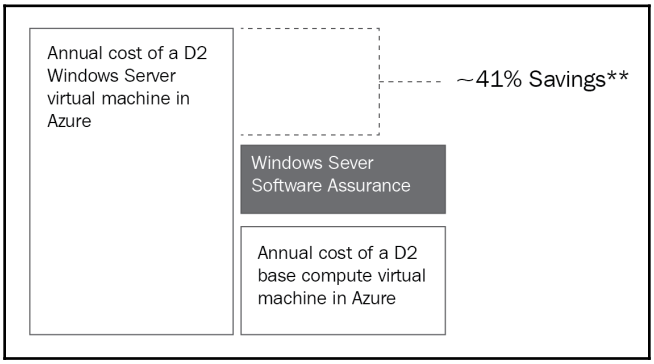
Development environment

As the CDO is a former employee of Amazon Web Services, and as all of the existing development code is in AWS, there is no need for him to change this:



As Jenkins is supported in Azure, too, the development is flexible. The main task is to design the build pipeline using stage environments. If DevOps decides to implement virtual machines for their services, these may also reside on AWS, but due to the underlying single identity and networking design, this really does not matter at all. The only requirement from the CIO is that if the VM is a Windows server, and not Linux, it must be placed on Azure, as in Azure, there is an option to save license costs by using Azure Hybrid Benefits.

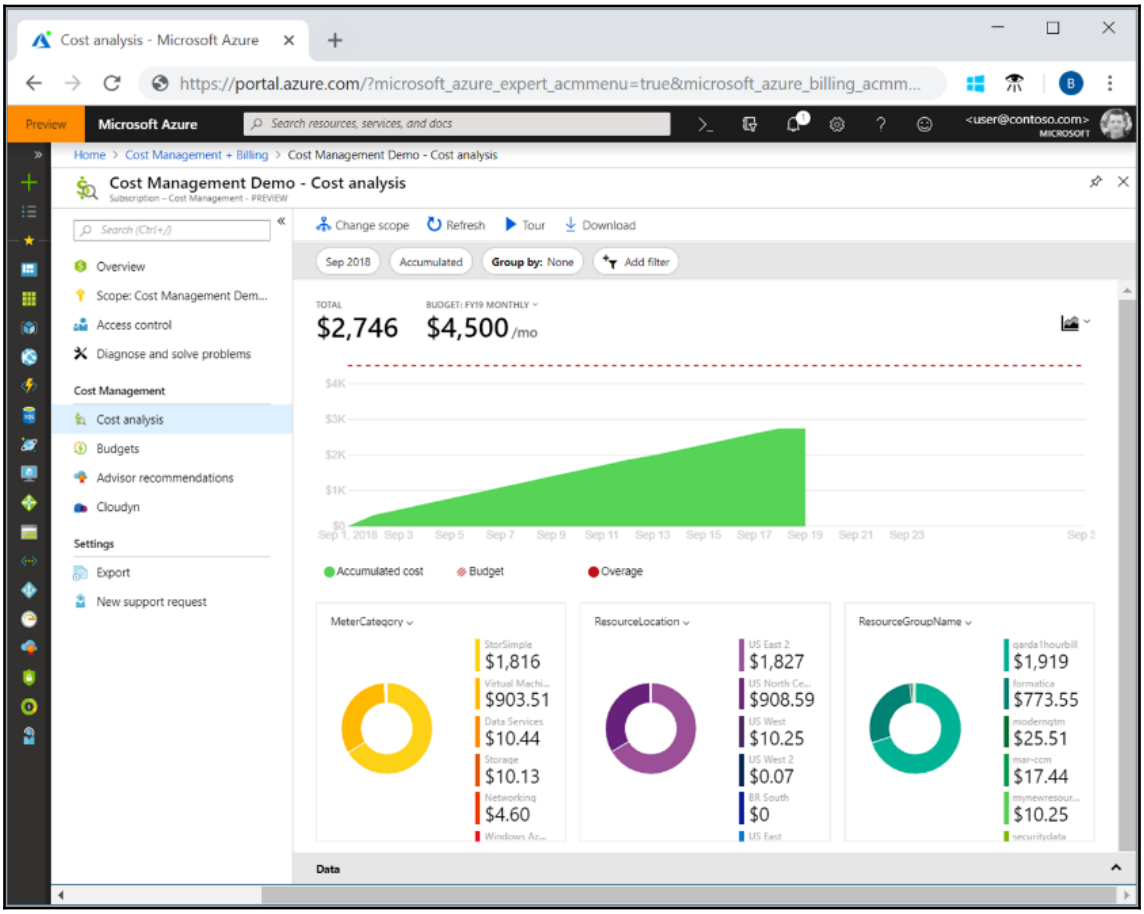
If you want to read more about *Azure Hybrid Benefits*, visit: `https://azure.microsoft.com/en-us/pricing/hybrid-benefit/`:



As you can see in the preceding diagram, there is about a 41% savings using the Hybrid Benefits and reusing the Windows server licenses in the cloud. So, the plan is to demote a server on-premises and enable it in Azure. With this switch of each VM, you will be able to transfer the license itself.

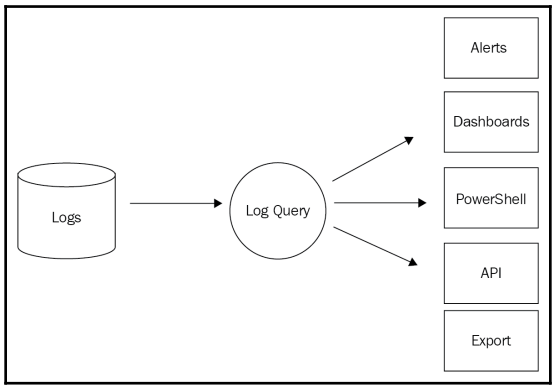
Cost management

Another final factor that you have to bear in mind is cost management. As cloud services are easy to deploy without any complex internal processes, there is a requirement to monitor costs. Therefore, you decide to go with Azure Cost Management, as it provides cost management for Azure, and even for Amazon Web Services:



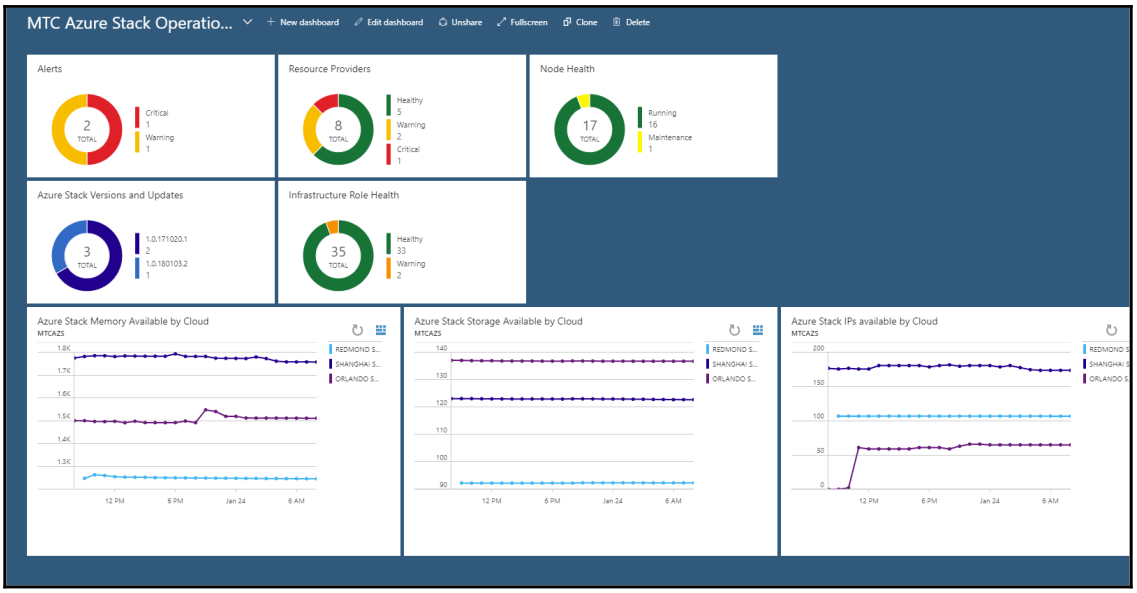
Monitoring and security

Currently, the company is using System Center Operations Manager for monitoring. This will still exist, in order to monitor Azure Stack. Everything in the cloud will be monitored using Azure Log Analytics, as shown in the following diagram:



As you can see in the preceding diagram, the log file collection of each cloud service provides a way to centrally alert and manage the availability of each service.

For all of the Azure Stack monitors, you decide to use the Log Analytics solution for Azure Stack as shown in the following screenshot:



This solution is available on GitHub, at `https://azure.microsoft.com/en-us/resources/samples/azurestack-adminpowershell-omsintegration/`.

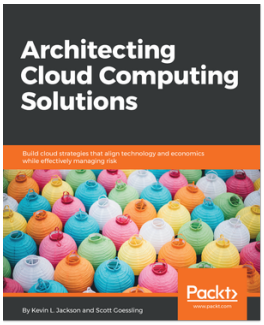
Summary

In this chapter, you learned that in a real-world scenario, designing a multi-cloud architecture using mixed IaaS, PaaS, and SaaS is quite simple if you follow the design guidelines that we have discussed in this book. With this knowledge, you should now be able to design and architect customer environments following a dual cloud approach.

It is important to provide a single networking and identity management solution. If you take into account the cloud vendor that the customer has chosen, this will not be an issue. It will work with each of them. Enjoy designing multi-cloud environments!

Another Book You May Enjoy

If you enjoyed this book, you may be interested in these other books by Packt:



Architecting Cloud Computing Solutions
Kevin L. Jackson, Scott Goessling

ISBN: 978-1-78847-242-5

- Manage changes in the digital transformation and cloud transition process
- Design and build architectures that support specific business cases
- Design, modify, and aggregate baseline cloud architectures
- Familiarize yourself with cloud application security and cloud computing security threats
- Design and architect small, medium, and large cloud computing solutions

Leave a review - let other readers know what you think

Please share your thoughts on this book with others by leaving a review on the site that you bought it from. If you purchased the book from Amazon, please leave us an honest review on this book's Amazon page. This is vital so that other potential readers can see and use your unbiased opinion to make purchasing decisions, we can understand what our customers think about our products, and our authors can see your feedback on the title that they have worked with Packt to create. It will only take a few minutes of your time, but is valuable to other potential customers, our authors, and Packt. Thank you!

Index

A

Access Management
  about 221, 222
  identity-based policy 223
  JSON policy, writing 224, 226
  policy, evaluating 223
  policy, types 223
  resource-based policy 224
AI and Cognitive services, Microsoft Azure service portfolio
  Azure Batch AI 28
  Machine Learning Services 27
  Machine Learning Studio 28
Alexa for Business 61
Amazon Chime 61
Amazon Cognito 58
Amazon Comprehend 56
Amazon Connect 61
Amazon ECS 53
Amazon EKS 53
Amazon Elasticsearch 53
Amazon Fargate 53
Amazon FreeRTOS 60
Amazon GameLift 60
Amazon Lex 56
Amazon Machine Images (AMI) 54
Amazon Macie 49
Amazon MQ 51
Amazon Pinpoint 61
Amazon Polly 56
Amazon QuickSight 52
Amazon Rekognition 56
Amazon SageMaker 55
Amazon Simple Email Service (SES) 61
Amazon Snowball 54
Amazon Transcribe 56
Amazon Translate 56
Amazon VPC, interconnecting to
  about 147
  AWS VPN CloudHub used 152
  Direct Connect and VPN used 151
  Direct Connect used 150
  Hardware VPN used 148, 149
  software VPN 153
  without VPN 154
Amazon VPCs, interconnecting among
  about 155
  Direct Connect 159
  hardware VPN 158
  software VPN 156
  software-to-hardware VPN 158
Amazon Web Services (AWS)
  about 35
  regions and availability zones 36
Amazon WorkDocs 61
Amazon WorkMail 62
Amazon WorkSpaces 62
Application Discovery Service 55
AppStream 2.0 62
AppSync 57
Athena 52
Augmented reality (AR) 57
Autonomous Systems (AS) 110
availability groups 235
AWS Certificate Manager 49
AWS CodeBuild 46
AWS CodeCommit 46
AWS CodeDeploy 46
AWS CodePipeline 46
AWS CodeStar 47
AWS compute services
  about 37
  Amazon Lightsail 38

Application Load Balancer 40
AWS Elastic Load Balancer 38
AWS Lambda 38
Classic Load Balancer 39
Elastic Compute Cloud (EC2) 37
Network Load Balancer 40
AWS Config 50
AWS database services
about 42
Amazon RedShift 42
DynamoDB 42
Elastic Cache 42
Relational database service (RDS) 42
AWS Device Farm 58
AWS Directory Service 49
AWS Glue 52
AWS Guard Duty 48
AWS Inspector 49
AWS OpsWorks 45
AWS Route 53 241
AWS Secrets Manager 48
AWS Services, interconnecting from inside VPC
about 160
Endpoint services 162
Gateway Endpoint 160
Interface Endpoint 161
AWS services
about 37
analytics 52
application integration 51
augmented reality and virtual reality 57
business productivity 61
compute services 37, 40
container service 53
customer engagement 61
database services 42
desktop and app, streaming 62
DevOps 45, 47
game development 60
Internet of Things (IoT) 59
machine learning 55
media services 62
migration services 54
mobile 57
monitoring 50
networking 43
storage services 40
AWS Step Functions 51
AWS storage services
Elastic Block Store (EBS) 40
Elastic File System (EFS) 41
Glacier 40
Instance Store 41
Simple Storage Service (S3) 40
Storage Gateways 41
AWS Trusted Advisor 50
AWS VPN CloudHub 152
AWS WAF and Shield 48
Azure design, scenarios
about 78
regulations 78
ROI and TCO 79
sensitive data 78
Azure ExpressRoute
Azure ExpressRoute Direct 131
Azure ExpressRoute Global Reach 129, 130
Azure Stack
API 72
CLI 72
portal 70, 71
PowerShell commandlets 71
sizing 73, 74, 75
starting 79, 80, 81
testing, in public Azure 82
working with 70
Azure Traffic Manager 240
Azure Virtual WAN 133
Azure VPN 124

B

bandwidth 115
Basic Support for Cooperative Work (BSCW) 6
best practices 253
Border Gateway Protocol (BGP) 111, 127

C

centralized routing 113
cloud computing
about 6
characteristics 8

defining 8
cloud deployment models
about 13
community clouds 14
hybrid clouds 14
private clouds 13
public clouds 14
cloud models
about 10
cloud deployment models 13
cloud service models 10
cloud providers, connectivity options
about 114
encryption, via internet 114
private network interconnect, via Ethernet, MPLS, or IPVPN provider 114
virtual private network tunnel, via internet 114
cloud service models
about 10
classic data centers 11
Infrastructure as a Service (IaaS) 11, 247, 248
Platform as a Service (PaaS) 12, 248
Software as a Service (SaaS) 12, 248
cloud services
defining 247
cloud solutions, interconnecting in identity layer 171, 174
cloud solutions, interconnecting in network layer
about 166
via internet 167, 168
via private network interconnects 169
Virtual Private Networks 169
CloudFormation 46
CloudHSM 48
CloudSearch 53
CloudTrail 50
CloudWatch 50
clustered application servers, examples
about 235
AWS Route 53 241, 242
Azure Traffic Manager 240, 241
cross-cloud implementations supporting, geo load balancers used 240
Microsoft Exchange Server 237, 238, 239, 240
Microsoft SQL Server 235, 236
community clouds 14
compute, Microsoft Azure service portfolio
functions 24
Virtual machine scale sets 24
Virtual machines 24
connectivity providers
about 107
cloud and content providers 109
exchange providers 108
internet exchange points 108
internet service providers 108
network service providers 107
containerization
about 248, 249
best practices 253
Cloud Foundry 249
Docker 250
Docker Services as a Service 253
Microservices 252
OpenShift 251
core management services, Azure Stack
about 68, 69
authorization management 68
events 70
offer 70
plan 69
subscription 70
cross-cloud application servers, technical requisites
about 232
different clouds, networking connectivity 233
georedundancy, supported applications 233, 235
single identity management solutions, for all servers 233
cross-cloud application servers
troubleshooting 245

D

data and analytics, Microsoft Azure service portfolio
Azure Analysis Services 27
Azure Bot Service 27
Data Factory 27
Data Lake Analytics 27
Data Lake store 27
Dynamics 365, for Customer Insights 27

HDInsight 27
Log Analytics 27
Power BI Embedded 27
data in transit 116
data pipeline 52
Database Migration Service (DMS) 54
databases, Microsoft Azure service portfolio
Azure Database, for PostgreSQL 26
Redis Cache 27
SQL Server Stretch Database 27
decentralized routing 112
Desktop as a Service (DaaS) 11
developer tools, Microsoft Azure service portfolio
Azure DevTest Labs 29
HockeyAPP 29
Direct Connect 43
directory services
about 177, 178
Active Directory 178
Active Directory Federation Services 195
Azure Active Directory 179, 180, 182, 183, 184, 187
Azure AD users 188
Azure AD, design guidelines 191
Azure AD, devices 191
Azure AD, groups 189
Azure AD, roles 190
custom domain names 198, 199
hybrid cloud identity management solution, designing 193

E

Elastic Beanstalk 46
Elastic Map Reduce (EMR) 52
Enterprise Mobility Management (EMM) 33

F

Federal Identity Manager (FIM) 33

G

geographical regions 36
Greengrass 60
groups, IAM 216, 217

H

Hashicorp Configuration Language (HCL) 243
hybrid cloud access management solution
designing 200
multi-factor authentication 202, 204
self-service password reset 200, 201
hybrid cloud identity management solution, designing
about 193
Active Directory Federation Services 195
pass-through authentication 196
seamless single sign-on 196
hybrid clouds 14

I

Identity and Access Management (IAM)
about 48, 208
actions or operations 209
authentication 209
authorization 209
principal 208
request 210
resource 208, 209
working 208
Infrastructure as a Service (IaaS) 246
Intelligent Platform Management Interface (IPMI) 9
internet direct peering 111
Internet of Things (IoT) 59
IoT 1-Click 59
IoT Analytics 60
IoT Core 59
IoT Device Management 59

K

Key Management Service (KMS) 48
Kinesis 52

L

latency 115
Link State Routing (LSR) 111

M

managed policies
about 227

AWS managed policies 227
customer-managed policy 228, 229
inline policy 230
summary tables 231
media services
about 62
Amazon Elemental MediaLive 63
AWS Elemental MediaConvert 63
Elastic Transcoder 62
Elemental MediaPackage 63
Elemental MediaStore 63
Elemental MediaTailor 63
Microsoft 365 19
Microsoft 365 service portfolio
about 30, 31
Enterprise mobility and security 33
Exchange Online Archiving service 31
Exchange Online service 31
Office 365 Platform Service 31
Office Applications service 31
Office Online service 31
Power BI service 31
Project Online service 31
SharePoint Online service 31
Skype, for Business Online service 31
Windows 10 as a service 33
Workplace Analytics service 31
Yammer service 32
Microsoft Azure service portfolio
about 24
AI and cognitive services 28
Azure Marketplace 30
compute 24
data and analytics 27
databases 26
developer tools 29
enterprise integration 28
Internet of things 28
mobile 26
monitoring and management 29
network 24
security and identity 29
storage 25
Microsoft Azure
about 18
cloud offerings 19
Microsoft Cloud
and Azure Stack 66
Microsoft Exchange Server 237, 238, 240
Microsoft Identity Manager (MIM) 33
Microsoft SQL Server 235, 236
Microsoft Trust Center 33
Microsoft
data centers 20
global network backbone 20, 24
mixed interconnect
Azure Virtual WAN 133
with software-defined WANs 131
mobile, Microsoft Azure service portfolio
API Apps 26
API management 26
App service 26
Azure Search 26
Logic Apps 26
Mobile Apps 26
Web Apps 26
multi-cloud solutions
design guidelines 246
multi-cloud virtual machines
managing, for clustered application servers 243, 244
multi-cloud
defining 15, 16
management 174
templates 174
Multiprotocol Label Switching (MPLS) 127

N

NAT gateway 141
National Institute of Standards and Technology (NIST) 8
Network Access Control List (NACL) 141
Network Function Virtualization (NFV) 131
Network Virtual Appliances (NVAs) 9
network, Microsoft Azure service portfolio
Azure DDoS Protection 25
Azure DNS 25
Content delivery network 25
ExpressRoute 25
load balancers 25

Network, watcher 25
virtual networks 24
VPN Gateway 25

O

Open Shortest Path First (OSPF) 111
OpenStack services
about 86
application data protection - Karbor 103
bare metal service - Ironic 89
block storage service - Cinder 90
cluster service - Senlin 102
compute service - Nova 87, 88
computing inside storage objects - Storlets 103
container orchestration - Magnum 99, 100
container service - Zun 97
dashboard - Horizon 94
data-processing service - Sahara 101
database service - Trove 98, 99
disaster recovery service - Freezer 96
DNS service - Designate 99
file-sharing-service - Manila 105
identity service - Keystone 89, 90
image service - Glance 93
Key manager service - Barbican 104
load balancing solution - Octavia 97, 98
messaging system - Zaqar 100
Neutron 92
object storage service - Swift 91, 92
orchestration service - Heat 95, 96
resource optimization service - Watcher 103, 104
resource-reservation service - Blazar 102
Root-cause analysis service - Vitrage 104
search service - Searchlight 104
telemetry service - Ceilometer 94, 95
OpenStack
architecture 84, 86
control plane 86
history 84

P

Platform as a Service (PaaS) 12, 246
private clouds 13
private network interconnect (PNI)
about 111, 120, 127
Azure ExpressRoute 127, 129
public clouds 14
public interconnect
about 118, 120
Azure VPN 123
interconnect, via internet (HTTPs) 121, 123
VPN, with network virtual appliances 124
public peering 112

Q

Quality of Service (QoS) 127

R

Real-world scenario design
about 254, 255
cost management 269
development environment 268
identity management 259, 263
infrastructure services 265, 267
modern workplace services 263
monitoring 270
networking 258, 259
new cloud services, setting up 267
preparations 255, 257
security 270, 271
solution design 255
remote peering 112
resource providers 67, 68
roles
about 218
AWS service role 219
AWS service role, for EC2 instance 219
AWS service-linked roles 219
Cross-Account Access roles 221
delegation 220
Route 53 44
routing tables 139

S

Security Assertion Markup Language 2.0 (SAML 2.0) 212
Server Migration Service (SMS) 54
services, Azure Stack
about 75

Infrastructure as a Service (IaaS) 75
Platform as a Service 76, 77
serverless compute 77, 78
Simple Notification Service (SNS) 51
Simple Queue Service (SQS) 51
Simple Workflow Service (SWF) 51
Software as a Service (SaaS) 12, 246
software VPN 153, 157
Software-Defined Networking (SDN) 131
storage, Microsoft Azure service portfolio
Azure Backup 26
Azure Site Recovery 26
Blob storage 25
disk storage 25
StorSimple 25
subnets 137
Sumerian 57

U

users, IAM
about 210, 211
federated users 212, 214
root user 210, 211
web identity federation 215, 216

V

Virtual Private Cloud (VPC)
about 43, 136
building 142
creating 143
default VPC 136
Elastic IP, associating 146
instances, launching 146
internet gateway 139
internet gateway, creating 143
NAT gateway, creating 146
non-default VPC 136
routing table, creating 143, 146
routing tables 139
subnets 137
subnets, creating 143
VPC Peering 156