MS AZURE ADMINISTRATION

Contents

Introduction to Cloud Computing 8
Cloud computing 8
Types of cloud computing service models 9
Types of cloud deployment models 11
What is Microsoft Azure? 14
Overview of Microsoft Azure 15
Azure basics 15
Azure accounts versus Azure subscriptions 15
Azure Resource Manager (ARM) versus Azure Service Manager (ASM) 16
Azure global infrastructure 17
Availability Sets versus Availability Zones 19
Azure tools 19
Azure Portal 19
Azure command-line interface (Azure CLI) 20
Azure Cloud Shell 21
Azure PowerShell 22
Azure SDK 23
Azure RESTful API 23
ARM templates 24
Azure developer tools 24
Overview of Microsoft Azure core services 25
Azure Compute services – IaaS versus PaaS 25
Azure Networking 26
Azure Storage 26
Data and analytics services 27
Backup services and disaster recovery 28
Administrative roles and role-based access control 28
Further reading 29
Implementing and Managing Azure Virtual Machines 30
The principles of Azure VMs 30
Planning and deploying Azure VMs 31

Identifying the workloads 31
Choosing the appropriate Azure VM sizing 31
Azure VM storage options 33
Managed disks versus unmanaged disks 33
Azure reserved VM instances (RIs) versus pay–as–you–go instances 34
Deploying an Azure VM 34
Creating Azure VMs 34
Connecting to Azure VMs 51
Connecting to a Windows Azure VM via Remote Desktop Protocol (RDP) 52
Connecting to a Linux Azure VM via Secure Shell (SSH) 52
Configuring Azure VMs in security 53
Restricting access to Azure VMs from the internet using NSG 53
Managing Azure VMs with VM Agent and VM extensions 55
Configuring the availability and scalability of Azure VMs 57
Scaling Azure VMs 57
Managing Azure VM's availability 67
Implementing and Managing Containers in Azure 73
The principle of containers and microservices 73
Containers versus container clusters 75
Docker basics 76
Container registry 77
Dockerizing your web application in Azure 78
Preparation work 78
Implementing Azure Container Registry 79
Pushing your Docker image in ACR 81
Deploying your Dockerized application with CI/CD capabilities 84
Clustering solutions with Azure ACS in Azure 84
An overview of container cluster solutions working with ACSs 85
Implementing three types of orchestrators of Azure ACS in Azure 89
Docker Swarm 89
Kubernetes 93
Implementing and managing a Kubernetes cluster with AKS 99

Implementing and Managing Azure Virtual Networks 105
Planning and designing Azure virtual networks 105
Analyze network requirements 105
Determine the type of connectivity 107
Determine the address space 107
Assigning static, public, and private IP addresses 108
Implementing Azure virtual networks 110
Creating an Azure virtual network 110
Updating the Azure virtual network 115
Delete Azure virtual network 117
Managing Azure virtual networking 118
Routing network traffic 118
Filtering the network traffic 120
Distributing network traffic 121
Design and implement cross-premise and multisite connectivity 129
Point-to-site virtual private network (VPN) over IKEv2 or SSTP 130
Site-to-site and multisite virtual private network (IPsec/IKE VPN tunnel) 132
VNet-to-VNet virtual private network (IPsec/IKE VPN tunnel) 133
Virtual network peering (VNet Peering) 134
ExpressRoute 140
Configuring Hybrid Connections for App Service 143
Configuring multi-region applications with Azure Traffic Manager 145
Creating a Traffic Manager profile 146
Adding endpoints to the Traffic Manager profile 148
Managing Traffic Manager profiles 150
Integrating Azure services with an Azure virtual network 151
Restrict network access to PaaS resources using a service endpoint 153
Integrating a web app in App Service with an Azure virtual network 153
Configuring accelerated networking to improve your networking performance 155
Managing Azure Identities 156
Implementing and managing Azure Active Directory (Azure AD) 157
Managing identities via Azure Active Directory admin center 157

Creating an Azure Active Directory via the Azure Portal 159
Creating and managing Azure AD users 162
Creating Azure AD groups and managing user groups 164
Enabling Multi-Factor Authentication for users 165
Using bulk update for custom user profile properties 167
Managing devices 168
Add a custom domain 169
Conditional access 171
Configuring self-service password reset 172
Configuring privileged identity management 174
Configuring Azure AD identity management 174
Leveraging Microsoft Graph other than Azure AD Graph API 175
Integrating applications with Azure AD 175
Creating an Azure AD B2C directory 176
Managing Azure AD B2C directory 178
Implementing Business to Business (B2B) collaboration 180
Integrating applications with Azure AD 181
Implementing federation and social identity provider authentication 183
Configuring SAML-based SSO for an application with Azure AD 186
Managing hybrid identities 186
Configuring Azure AD Connect and synchronization services 186
Managing domains with Azure AD domain services 188
Implementing SSO in hybrid scenarios 190
Monitoring on-premises identity infrastructure and synchronization services 190
Planning and Implementing Azure Storage, Backup, and Recovery Services 191
Implementing and managing Azure Storage 192
An overview of Azure Storage services 192
Implementing Azure Storage services 192
Managing Azure Storage services 214
Implementing hybrid storage solutions 222
Moving data to and from Azure Storage 222
Implementing data storage services 225

SQL Database 226
Azure Database for MySQL 226
Azure Database for PostgreSQL 227
Database-managed instances 227
Azure SQL Data Warehouse 228
Cosmos DB 228
Configuring Content Delivery Network 229
Implementing a business continuity and disaster recovery (BCDR) strategy in Azure 231
Planning a BCDR strategy 232
BCDR in Azure 232
Implementing Azure Backup 233
Planning and implementing Azure Site Recovery 243
Managing Azure Operations and Automation 247
Implementing Azure Automation 247
An overview of runbooks 248
Creating an Automation account 248
Creating or importing PowerShell runbooks 251
Managing PowerShell runbooks 257
An overview of Desired State Configuration 261
Implementing PowerShell Desired State Configurations 262
Managing PowerShell Desired State Configurations 263
Other excellent configuration management tools 264
Implementing Azure Automation-based cloud management 264
Integrating Azure Automation with Web Apps 265
Comparing Azure Automation with Azure Functions 265
Integrating with Event Grid 266
Integrating with Logic Apps 268
Runbook gallery 269
Implementing monitoring solutions in Azure 270
Core monitoring 271
Deep application monitoring 275
Deep infrastructure monitoring 276

Shared capabilities 277
Implementing Azure VMs monitoring solutions 278
Configuring ARM VM monitoring 278
Configuring alerts 279
Configuring diagnostic and monitoring storage location 280
Implementing Log Analytics (OMS) solutions 282
Creating an OMS workspace 282
Collecting and searching across data sources from multiple systems 284
Transforming Azure activity data and managed resource data 285
Building custom visualizations with view designer 287
Sending data to Log Analytics with the HTTP Data Collector API 288
IT Service Management Connector 288

Introduction to Cloud Computing

Information technologies have evolved significantly over the last decade. For those of us in the professions of creating, building, and developing within these changes, cloud computing is one of the fastest-growing areas in our world. **Cloud**is the keyword of our new age, and it will be a fundamental part of everything in our future.

As an IT professional, I believe that being successful in IT is really about being a lifelong student, which means that we must constantly learn new skills and different platforms. Thus, this exam guide is not only a tool to help IT professionals get certified, but it will also to help them to get hands-on experience through practice labs. I would love to share my experience on how to build our knowledge of Microsoft Azure from the conceptual level to the operational level.

Cloud computing

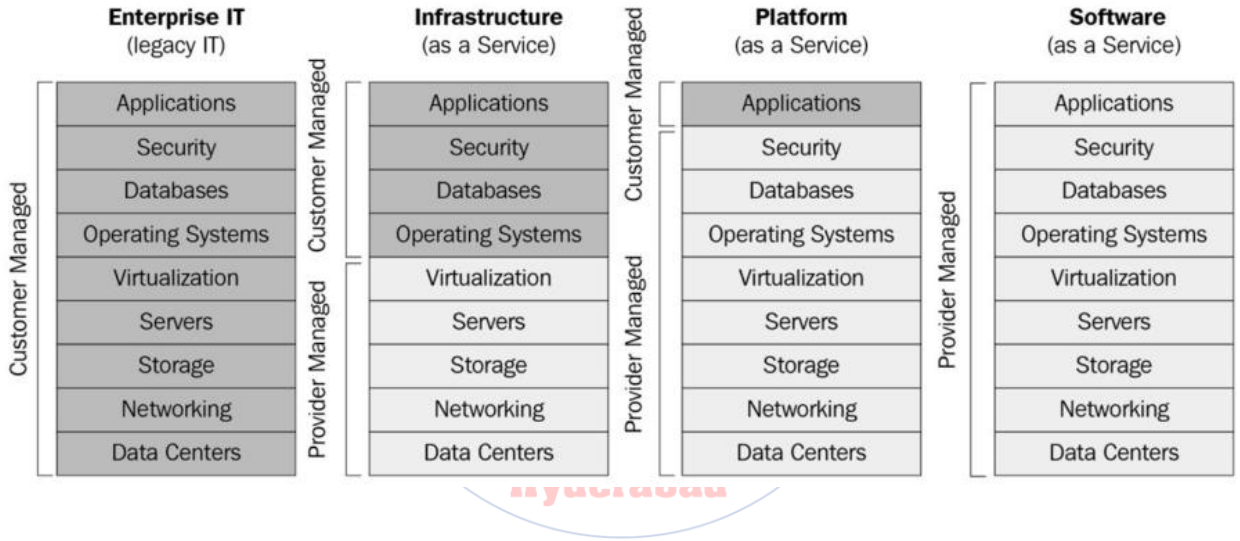
Cloud computing has been a star since it was born; it appears with big data, **Internet of Things** (**IoT**), and **Artificial Intelligence** (**AI**) in our conversations.

Modern cloud computing services providers such as AWS, Microsoft Azure, and Google Cloud Platform, generally consist of four basic components: **compute, network, database, and storage**. These providers provide computer services such as virtual machine, storage services to store objects/files on the cloud, provide the different cloud-based databases to store data, and network service to deploy virtual networks. Nowadays, the popular cloud providers seem more ambitious than we thought they were. They don't limit themselves to act as infrastructure that supports application deployments but they also provide the management services to support DevOps, monitoring, logging, and alarming, backup and restore in time on the cloud and integration tools to build CI/CD pipelines. More and more, it provides advanced services support such as the **ETL** (**extract, transform and load**) processing and data analytics, **Machine Learning** (**ML**), AI, and also IoT services to communicate

with IoT devices. Theoretically, cloud computing technology can help us do everything we want in the cloud.

Types of cloud computing service models

Predominantly, cloud computing is built into three types of cloud computing service models: **Infrastructure as a Service** (**IaaS**)**, Platform as a Service** (**PaaS**), and **Software as a Service** (**SaaS**). Each service offering provides a different level of virtualization and management responsibilities. The following is a screenshot showing which responsibilities each service model provides for:



Different cloud computing service models

IaaS – Infrastructure as a Service

As shown in the preceding screenshot, the IaaS service model provided is generally the capability of infrastructure level to the users; the cloud providers managed the hardware and infrastructure such as virtual servers, storage, networks, connectivity, operating systems and other fundamental computing resources. Based on the IaaS offering, users should manage them with administration works such as installing the patches, updates, and configurations. One of the best examples of this model is the virtual machine in the cloud.

PaaS – Platform as a Service

The PaaS services model provides the capability that comprises deployed and configured IT resources in a ready state based on the IaaS model. Users don't need to care about the infrastructure level and even the administration work they face when they're in the IaaS model. It directly provides the environment with the specified runtime. What users need to do is focus their work on the application level.

SaaS – Software as a Service

Compared to PaaS, SaaS has a more advanced virtualization level, which is widely accessed over the internet and directly used by users. The most common example is Google's **G Suite** and Microsoft's Visual Studio Team Services (which is also known as **Visual Studio Online** or **VSTS**).

X as a Service in the cloud

Based on these models, there are also some extension concepts which are known as **X as a service** such as:

- **Database as a Service** (**DBaas**): A managed database service in the cloud that aims to offer the database layer to the applications; the cloud provider manages the complex database environments.
- **Container as a Service** (**CaaS**): A managed service model that provides the container-based virtualization technology to let users manage and deploy containers; and applications as well as container clusters in the cloud.
- **Messaging as a Service** (**MaaS**): A messaging service in the cloud that allows sending and receiving messages through a queue. Originally implemented for the purpose of resolving queue-based load-leveling problems for a service whose peaks in demand make services or applications in the cloud overload and therefore unable to respond to requests in a timely manner. The queue acts as a buffer, storing the message until it's retrieved by the service. Applications or services in the cloud retrieve the messages from the queue and process them.
- **Logic as a Service** (**LaaS**): Also known as serverless. It gives little control over the infrastructure, the related infrastructure is managed by the cloud providers, and users can focus themselves on coding and configuring settings. Great examples include Azure Functions and Azure Logic Apps.
- **Identity as a service** (**IDaaS**): Supplies cloud-based authentication or identity management to enterprises or organizations. The goal is to ensure if a user has access to

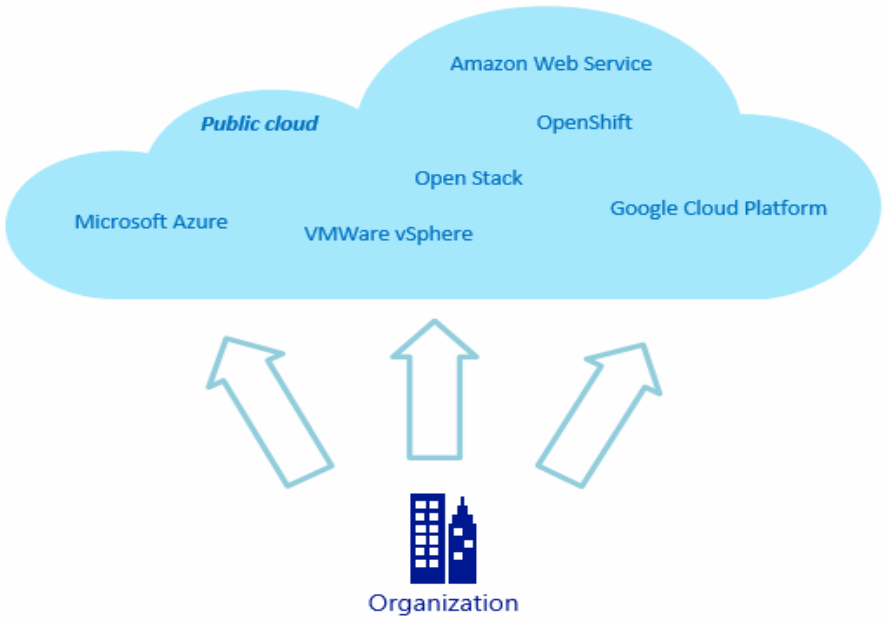
cloud applications or services and which type of access they could have to cloud applications or services.

There are some other service models such as **Disaster Recovery as a Service** (**DRaaS**), **Data as a Service** (**DaaS**), **Big Data as a Service** (**BDaaS**), **Log as a Service** (**LaaS**), and more than the mentioned models here. We believe, as cloud computing is one of the fastest-growing technologies, more and more services will appear and may serve together for future cloud computing platforms.

Types of cloud deployment models

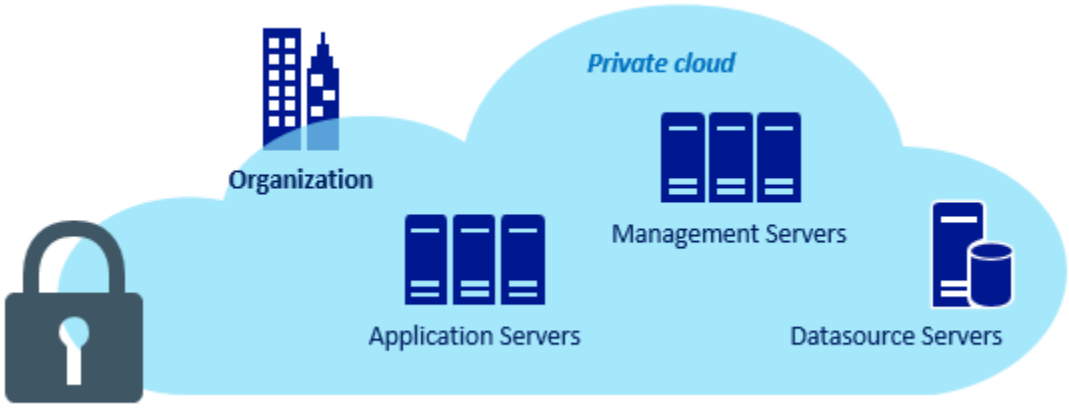
Typically, within cloud computing, it is possible to build our model as one of the four types of cloud deployment model: public cloud, private cloud, hybrid cloud, and community cloud. Each of them is defined for different levels of management, such as where the IT resource is located and security reasons.

The **public cloud** is a publicly accessible cloud environment provided by a cloud provider such as Microsoft Azure, **Amazon Web Service** (**AWS**), or **Google Cloud Platform** (**GCP**). These platforms manage the IT resources in their data center and are responsible for the security of these IT resources. With the help of the Internet, users can access cloud services on the **public cloud** from anywhere around the world. The following is an image showing how the **public cloud** works:



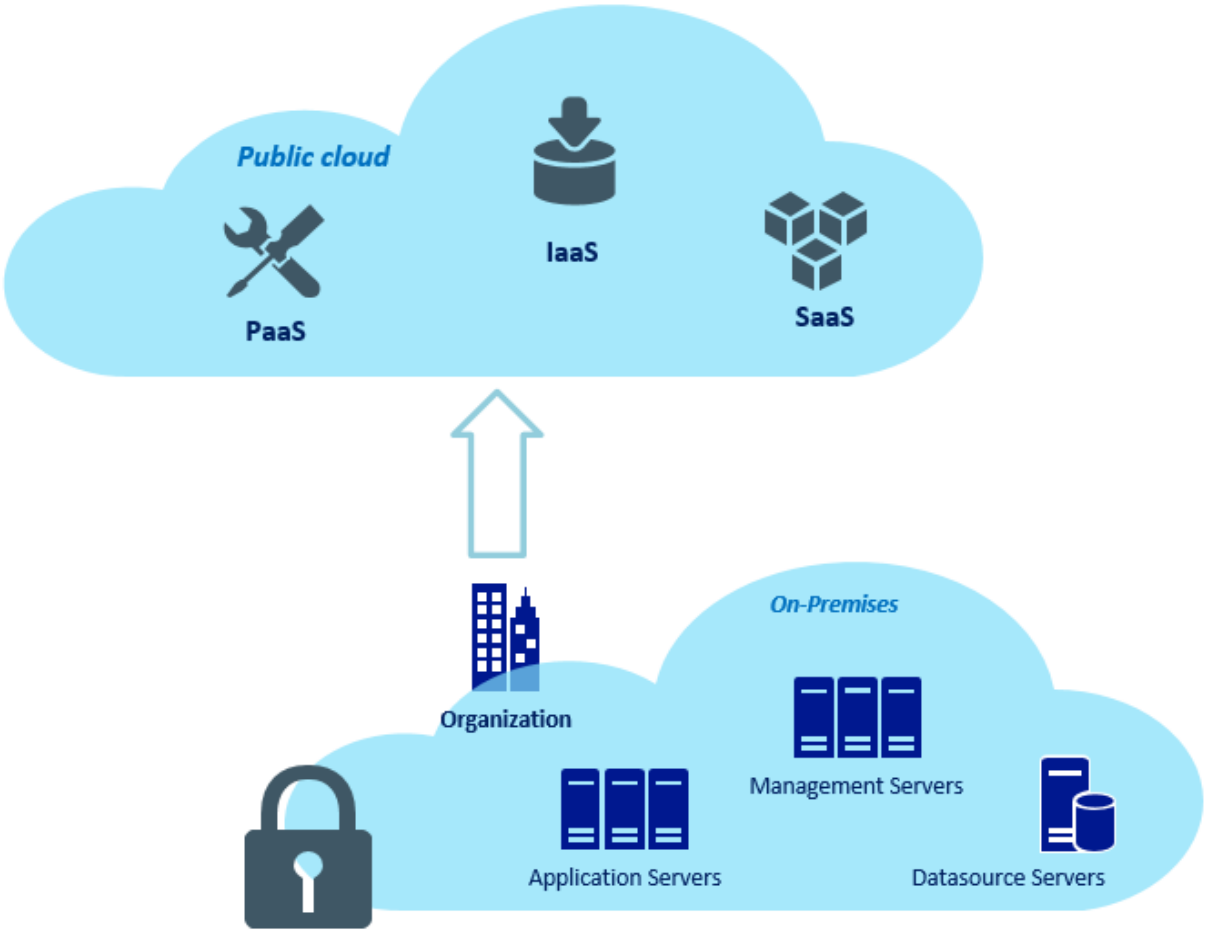
Public cloud

As compared to the **public cloud**, a **private cloud** is more critical in security and can only be accessed it from the internal network where the infrastructure is hosted. IT resources in private clouds are managed by companies' or **organizations**' own data center. Users can access it only when they're on the internal network. The following is an image showing how a **privatecloud**works:



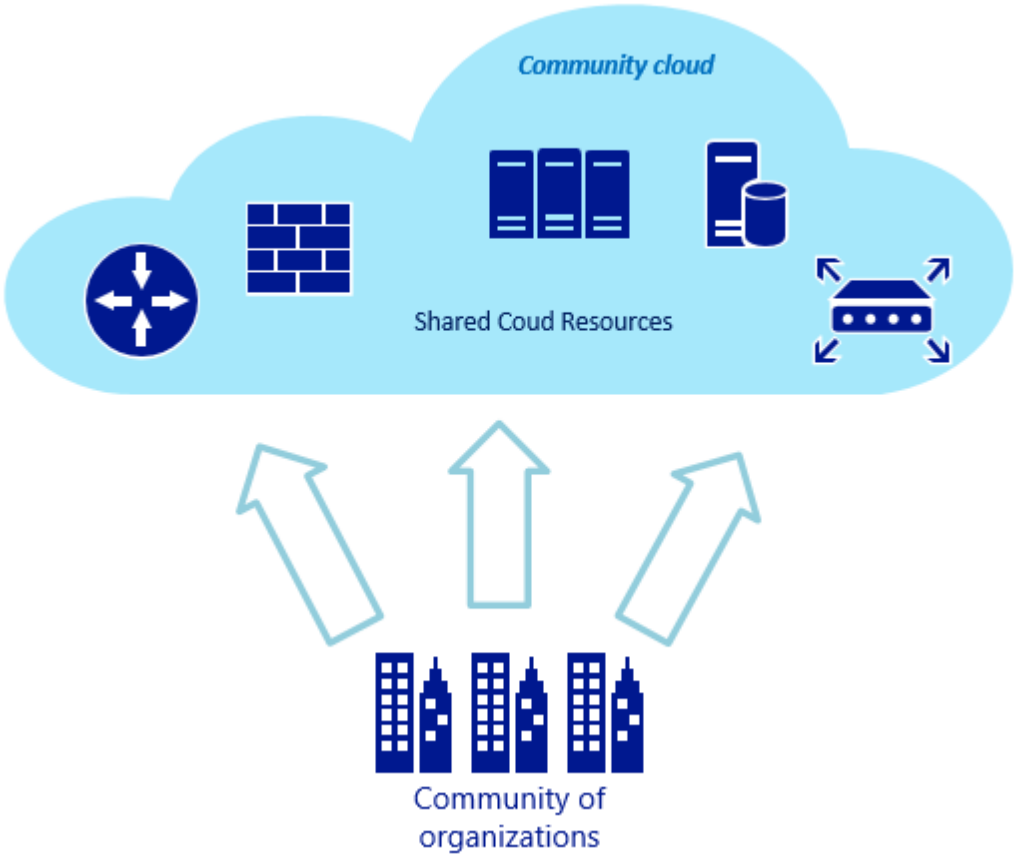
Private cloud

Based on the **public cloud** and **private cloud**, a **hybrid cloud** is intended to combine both of them in the same scenario. Generally, a network connection (dedicated or private) should be established between the **private cloud** and the **public cloud**. Hence, it is important to define which IT resources are on-premises and in the cloud and how do they work together. Be careful, as a hybrid cloud is intended to be for short-term configurations. If we are in a transition stage, a hybrid cloud is the most common cloud deployment model. The following is an image showing how a hybridcloud works:



Hybrid cloud

A **community cloud** is similar to a **public cloud**, but it is intended to be limited to a specific member of the community (or authorized access by the community); the IT resources can be located in the community's data center and shared by several members. The general scenario is several applications consume the common IT resources, but in this context, the **community cloud** usually ensures a dedicated connection between each consumer and IT resource, as illustrated in the following diagram:



Community cloud

What is Microsoft Azure?

Microsoft Azure was announced in October 2008 and then released on February 1, 2010 as Windows Azure. By March 25, 2014, it was renamed Microsoft Azure. The Microsoft Azure public cloud platform offers IaaS, PaaS, and SaaS services to enable businesses worldwide to create, deploy, and operate cloud-based applications and infrastructure services.

One of the reasons why Microsoft Azure is popular and fast-developing in the current market is because it is easy to work along with other Microsoft solutions such as Microsoft System Center, and can be leveraged together to extend an organization's current data center into a hybrid cloud that expands capacity and provides capabilities beyond what could be delivered solely from an on-premises standpoint.

Overview of Microsoft Azure

In this chapter, we'll introduce the core concepts of Microsoft Azure, including the different ways to access Microsoft Azure; introduce different Azure tools; indicate how to install and configure them; and provide an overview of Azure core services.

The following are the topics that we will cover in this chapter:

- Azure account and Azure subscription
- The classic model and Azure Resource Manager deployment model of Microsoft Azure
- Azure regions, global data centers, Availability Set, and Availability Zone
- Introduction to different tools to access Microsoft Azure
- An overview of Azure Compute, networking, storage, data and analytics, and backup and disaster recovery services
- Administrative roles and **role-based access control** (**RBAC**)

Azure basics

Microsoft Azure is a cloud platform launched by Microsoft that helps individuals and organizations provision, deploy, and operate cloud-based services and IT assessment

Azure accounts versus Azure subscriptions

When you're starting to use Azure, you'll create an Azure account. Microsoft lets users start Azure with a free Azure account. You can use the address given here to open your first Azure account: https://azure.microsoft.com/en-us/free/.

You can find details about all the service limits when you're using the Azure free account here at https://azure.microsoft.com/en-us/free/free-account-faq/.

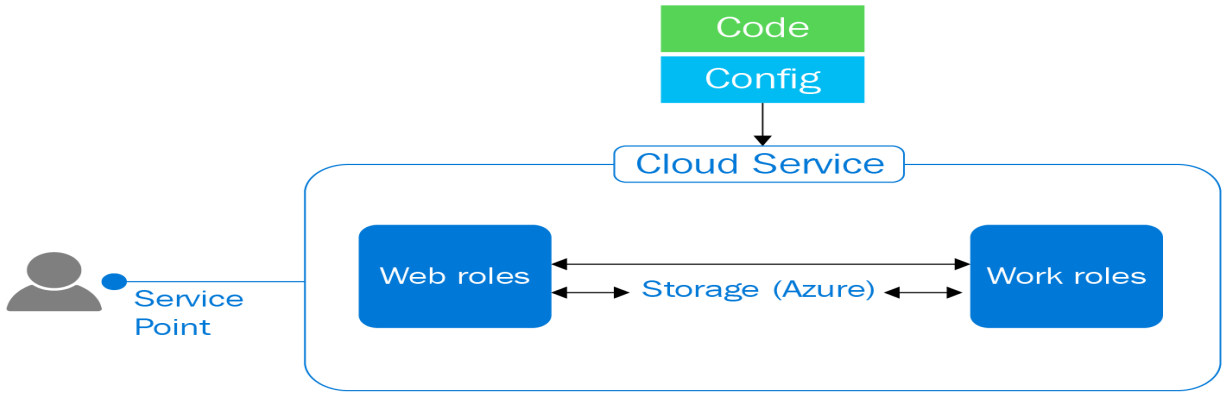
An Azure account contains one or more subscriptions. A subscription contains the details of services and billing use within an account. You can check the subscription in your account at https://account.azure.com/subscriptions/.

In Azure, the maximum number of services and resources is applied on per-subscription and per-region levels.

Azure Resource Manager (ARM) versus Azure Service Manager (ASM)

Microsoft Azure also has two different deployment models: Azure classic deployment and Azure Resource Manager deployment.

The **classic deployment**, which is also known as Azure Service Management model, is a historical deployment model, wherein each resource that existed in the cloud is independent and has no connection with another cloud. The following diagram shows the Azure Service Management model:



Azure classic deployment model

The IT resources for hosting virtual machines are provided by a **Cloud Service**, which is required in this model so that it acts as a container to host these VMs. There is also a **network interface card** (**NIC**) and an IP address, which are allocated by Azure linked with this VM.

Another deployment model is the **Azure Resource Manager** deployment model, ARM, which is different from classic deployment model. It lets you deploy, manage, and monitor all of the IT resources in the cloud such as virtual machines, storage accounts, virtual networks, or a database with a logical group, which is known as a **resource group**. The advantage of resource groups is organizing the resources in a logical way, and all the resources in the same resource group share the same life cycle, which means you can deploy, update, or delete them in a **one-click** way with only one single operation. Another great advantage is that resources deployed with ARM model can also be provisioned by a JSON-based template, which defines the dependencies between the deployed resources and the connection with the different resource groups. ARM deployment mode was inspired by **Infrastructure as Code** (**IaC**) which will be explained in the coming section. While interacting with the ARM, you can use command-line tools such as Azure PowerShell or Azure CLI, and you can also use ARM RESTful APIs.

Azure Resource Manager is not only a new deployment technique but also provides a consistent management layer for the different tasks you perform through different Azure tools, which we'll discuss in the upcoming section of this chapter.

When we start to deploy each a new resource in Azure, it will be necessary to specify the deployment model if this resource exists in two models. Take note of the fact that the Resource Manager deployment model and classic deployment model are not completely compatible with each other. ARM model is strongly recommended by Microsoft while deploying new IT solutions.

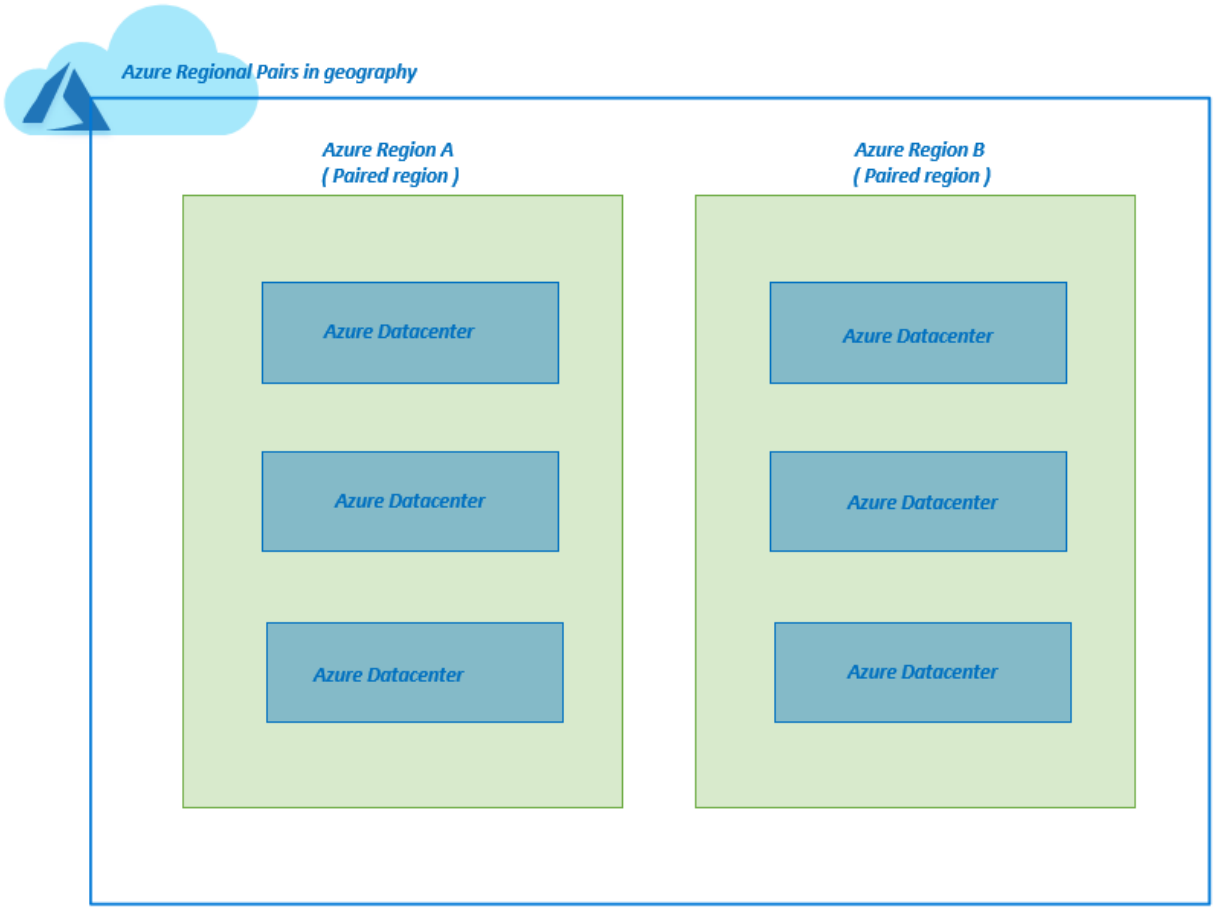
Azure global infrastructure

Azure services are hosted in physical Microsoft-managed data centers throughout the world. As of the time writing this , Azure was generally available in 54 regions and in over 140 countries around the world. To view the latest available regions of Azure global infrastructures, please check the following page at https://azure.microsoft.com/en-us/global-infrastructure/regions/.

Azure operates in multiple geographies around the world. An Azure geography is a defined area of the world that typically contains two or more regions and preserves data residency and compliance boundaries.

Whenever you create a new Azure resource, you must select an Azure region to determine the data center where the service will run. In Azure, a region is a set of data centers. Microsoft deployed their data centers within a latency-defined perimeter and then connected through a dedicated regional low-latency network.

One of the greatest difference between Azure regions and AWS regions is that the data centers are located in multiple geographic areas. Each Azure region is paired with another region within the same geography to form a regional pair. The only exception is Brazil South, which is paired with a region outside its geography:



Azure paired regions

Take note that you can specify the region where you want to host deployed resources in Azure, but not all Azure services are available from every region. Check https://azure.microsoft.com/en-gb/global-infrastructure/services/ to know more about Azure products available by region.

Availability Sets versus Availability Zones

In Azure, there are two concepts when we talk about availability, namely, which is the type of availability that is set and the zone which it is available.

An **Availability Set** is used to make sure the deployed VMs in Azure are distributed across multiple isolated hardware nodes in a cluster so that only a subset of your VMs is impacted in the case of failure, but your overall solution remains available and operational, it can provide SLA of 99.95%.

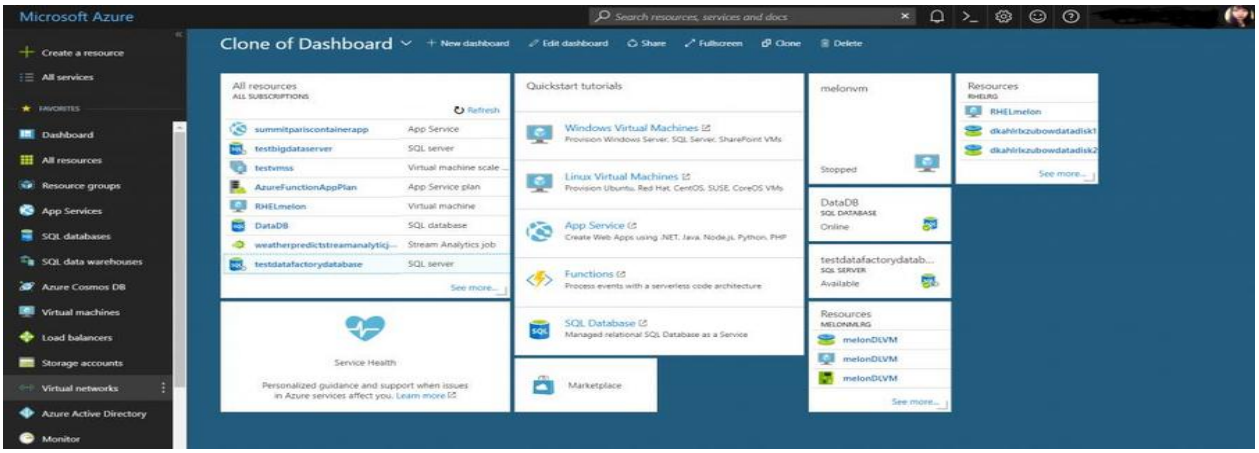
An **Availability Zone** consists of fault-isolated locations within an Azure region. One Availability Zone can consist of one or more data centers, it provides redundant power, cooling, and networking. Availability Zones are a great fit for mission-critical applications with requirements for intra-region resilience and fault tolerance in the case of failures of the data center., it can provide SLA of 99.99%.

Azure tools

Microsoft Azure provides different command-line tools and development tools to facilitate building, debugging, deploying, diagnosing, and especially managing scalable and elastic apps in Azure. Let's take a look at each of them.

Azure Portal

We can use https://portal.azure.com to navigate to Azure's new portal as shown as next:



Azure Portal

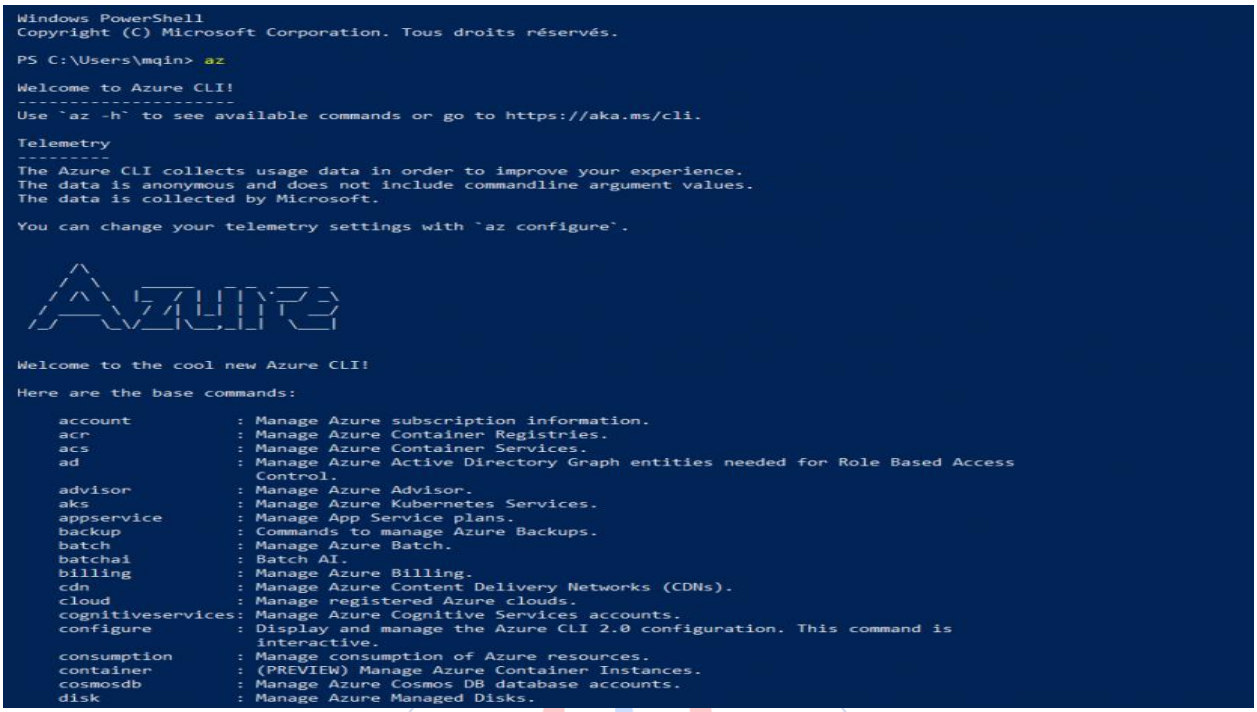
Azure command-line interface (Azure CLI)

Azure CLI, which was previously named Azure xPlat CLI, is an open-source, cross-platform, shell-based command-line interface designed for scripting and automating the creation and management of resources in Azure. Azure CLI works on Windows, Linux, macOS, and Docker containers. To install and configure Azure CLI under different OS, check the following page at https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest.

You can test whether the installation was successful using the following command from your command line:

```
az --help
```

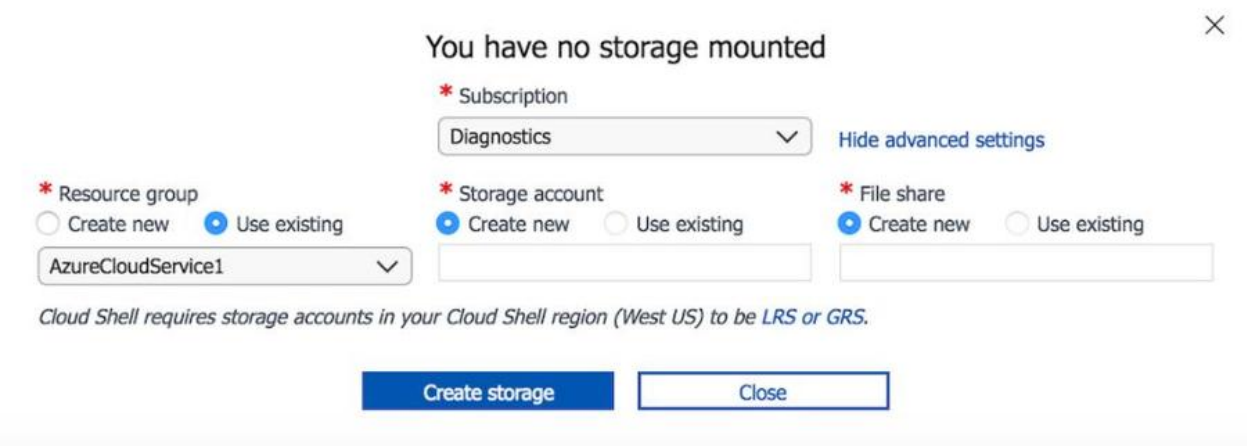
The following screenshot shows sample output:



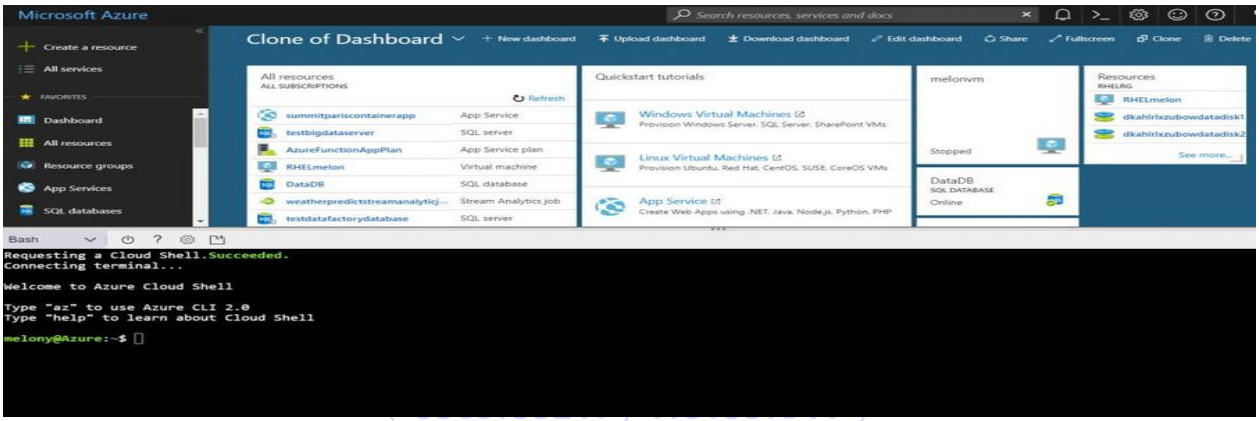
Azure Cloud Shell

Azure Cloud Shell is an interactive browser-based shell command line to manage cloud resources in Azure. Cloud Shell comes preinstalled with some popular command-line tools and language support such as Azure CLI, Bash, npm, mvn, git, and Docker.

The first time you launch Cloud Shell from the Azure Portal, it will create a resource group, storage account, and file share on your behalf; this is a one-time step and will be automatically attached for all sessions:



You can access Cloud Shell from shell.azure.com or via the Azure Portal (the following screenshot shows accessing Cloud Shell via Azure Portal):



Azure PowerShell

Azure PowerShell is one of the most powerful tools developed by Microsoft, and contains a set of modules within PowerShell that provide cmdlets to manage cloud resources in Azure. Azure PowerShell contains two different modes, which are defined using the Azure Service Manager or Azure Resources Manager modules, and which we explained previously. Azure PowerShell works on Windows, macOS, and Linux. To install and configure Azure PowerShell check the address: https://docs.microsoft.com/en-us/powershell/azure/install-azurerm-ps?view=azurermps-5.5.0.

Azure SDK

Azure SDK helps developers to deploy infinitely scalable applications and APIs, configure diagnostics, create and manage app service resources, and integrate data from Visual Studio. Currently, Azure SDK is available in many popular development languages such as .NET, Java, Node.js, Python, Ruby, PHP, JavaScript, and Swift.

To download Azure SDK in different programming languages, check https://azure.microsoft.com/en-au/downloads/.

Microsoft Azure provides also some Azure code samples at https://azure.microsoft.com/en-us/resources/samples.

Azure RESTful API

Most Azure service REST APIs have client libraries that provide a native interface for using Azure services. **Representational State Transfer** (**REST**) APIs are service endpoints that support sets of HTTP operations (methods), which provide create, retrieve, update, or delete access to the service's resources. The Azure RESTful API is available in .NET, Java, Node.js, Python, and Azure CLI 2.0 SDK. Azure also provides a great way to secure your REST requests by registering your client application with **Azure Active Directory** (**Azure AD**). You can refer to the page of REST API browser which is currently in the preview state from https://docs.microsoft.com/en-us/rest/api/.

Similar to Azure PowerShell, Access APIs endpoint of Azure Resource Manager uses https://management.azure.com/ and the Azure classic deployment model uses https://management.core.windows.net/. Both of them should add an api-version query-string parameter at the end, which is mandatory. Below is an example of the Get VM scale set in current subscription:

```
GET
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroup
Name}/providers/Microsoft.Compute/virtualMachineScaleSets/{vmScaleSetName}/virtualmac
hines/{instanceId}?api-version=2017-12-01
```

ARM templates

An ARM template is a JSON file that defines the required resources in a declarative format to deploy IT solutions in a quick way in Azure, which is an excellent implementation of Infrastructure as a Code. It uses the definition file to define the infrastructure for your IT resource in the cloud. Similar to CloudFormation for AWS, ARM templates for Microsoft Azure make the infrastructure version possible. It is a simple and effective way to manage infrastructure resources in the cloud. The best practice recommended by Microsoft is to implement different ARM templates for different environments such as test, staging, and production.

You can find many useful sample ARM templates at: https://azure.microsoft.com/en-us/resources/templates/.

Or go to GitHub at: https://github.com/Azure/azure-quickstart-templates.

Azure developer tools

There are some other related Azure developer tools to support development in Azure with Microsoft solution or other open source solutions such as Visual Studio Tools for Azure, PowerShell Tools for Visual Studio, Storage Explorer, Docker Tools, and Azure Service Fabric Tools. Check the following address to know more about installation and configuration of these tools: https://azure.microsoft.com/en-in/tools/.

Overview of Microsoft Azure core services

Microsoft Azure is broken down into several high-level groupings of services. So far, there are more than 100 services in Microsoft Azure. They're generally grouped as Azure Compute, Azure Networking, Azure Storage, Azure Data and Analytics services, Azure Backup, and Azure Disaster Recovery.

You can take a quick look at the following link to take a global view of the latest Azure services: https://azure.microsoft.com/en-us/resources/infographics/azure/.

Microsoft also provides a search page so that users can use it to browse the latest products in different Azure categories as follows: https://azure.microsoft.com/en-us/services/.

Azure Compute services – IaaS versus PaaS

Azure provides different hosting models such as running applications on virtual servers or containers, or in a serverless computing environment, and each provides a different set of services. The following are some of the hosting models:

- **Virtual Machines** are the most important IaaS offering providing by Microsoft Azure. Different from physical machines, a virtual machine is a machine based on virtualization technology; they can be Windows-based or Linux-based VMs.
- **Virtual Machine Scale Sets** is a managed VM pool that contains a set of identical VMs. All VMs in VM scale sets with the same configuration is designed to improve scalability and availability.
- **App Service** contains PaaS offerings such as web apps, mobile apps, API apps, and logic apps in the same app service plan to provide a managed hosted environment.
- **Cloud Services** is a deployment solution with more control of the OS than App Service, there are two versions, IaaS cloud services; and PaaS cloud services.
- **Service Fabric** is a PaaS service which is designed for building packaging, deploying, and managing scalable and reliable microservices.

- **Azure Functions** is a **Function as a Service** (**FaaS**) service which provides serverless functions in the cloud. They are designed for deploying small pieces of code in different languages and in the cloud without managing the infrastructure.
- **Azure Batch** allows running large-scale parallel and high-performance computing applications efficiently in the Azure cloud.
- **Azure Container Service** (**ACS**) is designed for deploying and managing fully portable application containers and container orchestrators solutions such as Docker Swarm, DC-OS, as well as Kubernetes in the Cloud. Microsoft Azure also has a managed Kubernetes managed cluster which is known as **Azure Kubernetes Service** (**AKS**).

Azure Networking

Azure Networking provides the following connections to connect your virtual machines, PaaS cloud services, and on-premise infrastructures:

- **Azure Virtual Networks**: This enables you to deploy isolated networks in the cloud to securely connect Azure resources to each other
- **Azure ExpressRoute**: This helps you to create a dedicated high-speed connection from your on-premise data center to Azure
- **VPN Gateway**: The virtual private network gateway is used to send network traffic between Azure virtual networks and on-premise locations, and also between virtual networks within Azure
- **Traffic Manager**: This provides DNS-level load balancing for applications that need to be high-availability
- **Load Balancer**: This provides level 4 load balancing features distributing network traffic to your applications
- **Azure DNS**: This is a domain name resolution service to manage DNS records for both Azure services and external resources

Azure Storage

Azure Storage is a cloud storage service that is durable, available, and scalable. Azure storage provides the following four types of storage:

- **Blob storage**: This is an object-based storage used to store documents, media files, or even application installers
- **Table storage**: This is a NoSQL key-attribute data store that is designed for semi-structured data, which allows for rapid development and fast access to large quantities of data
- **Queue storage**: This provides reliable messaging for workflow processing and for communication between components of Azure services
- **File storage**: This offers shared storage for legacy applications using the standard SMB protocol

Data and analytics services

Azure provides different databases and analytics services to help migrate your data to the cloud. Azure takes care of scalability, backup, and high availability through the following services:

- **SQL Database**: This is a managed database service, which is different from AWS RDS, which is a container-based service with a different database engine.
- **Azure SQL Data Warehouse**: This is a cloud-based, scaled-out data warehouse in the cloud, which can process massive volumes of data, both relational and nonrelational.
- **CosmosDB**: This is designed as a globally distributed database, which allows you to use key-value, graph, column-family, and document data in one service. Multi-model and globally distributed is the most important aspect of CosmosDB. It independently and elastically scales storage and throughput at any time, anywhere across the globe, making it perfect for your serverless applications.
- **Azure Redis Cache**: This is a distributed, managed cache designed for building highly scalable and responsive applications by providing super-fast access to your data.
- **Azure Machine Learning**: This enables you to apply statistical models to data and perform predictive analytics in the cloud.
- **Azure Search**: This provides a fully managed search service in the cloud.
- **Azure Data Factory**: This is a cloud-based data integration service that orchestrates and automates the movement and transformation of data within the data pipeline.
- **Azure Data Lake Store**: This is an enterprise-wide hyper-scale repository for big data analytics workloads, which is naturally integrated with HDFS to support Hadoop-based analytics.

Backup services and disaster recovery

Azure provides a business continuity plan that includes disaster recovery for all your major IT systems without the expense of secondary infrastructure. The following are the core services:

- **Azure StorSimple**: This is an integrated storage solution that manages storage tasks between on-premise devices and Azure cloud storage in the case of failure
- **Azure Backup**: This provides a cloud-based back up and works with ASR to restore your data in the Azure cloud
- **Azure Site Recovery** (**ASR**): This orchestrates replication of on-premise virtual machines and physical servers in the Azure cloud and restores the backup

Administrative roles and role-based access control

Microsoft uses RBAC to let users manage permission levels on resources in Azure. It is strongly recommended to apply the least privilege principle for each role, which defines a set of permitted actions. For more information refer to: https://docs.microsoft.com/en-us/azure/billing/billing-add-change-azure-subscription-administrator.

Azure provides three subscription-level administrative roles which have basic access management permissions:

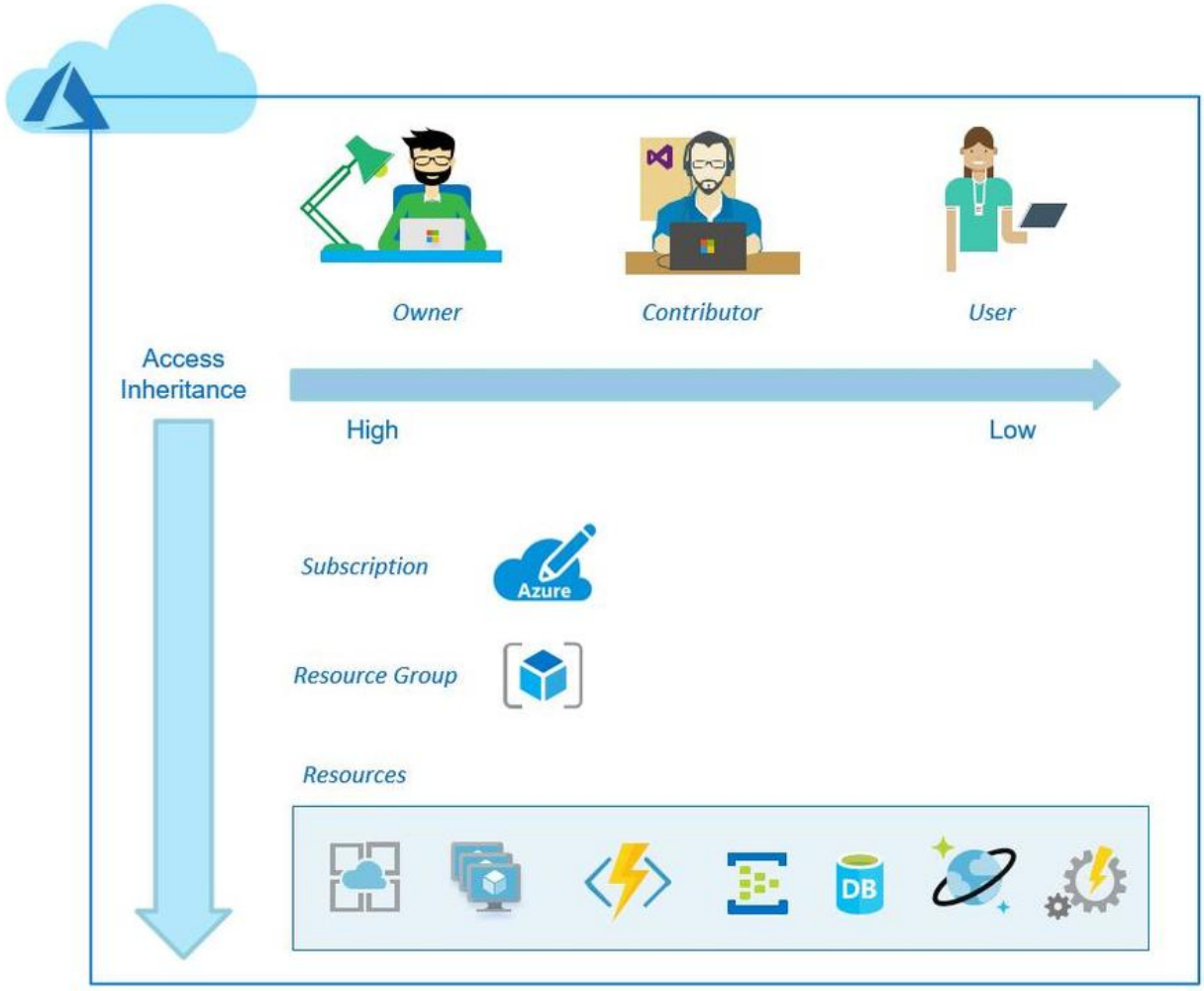
- Account administrator
- Service administrator
- Co-administrator

Azure RBAC is the latest access control system, and is recommended by Microsoft. It offers fine-grained access management, which has three basic roles that apply to all resource types in Azure:

- The **Owner** has full access to the defined resources, including the right to delegate access to others for these resources
- The **Contributor** can create and manage defined Azure resources, but can't grant access to others for these resources

- The **User** (reader) can view existing defined Azure resources

Each of these three roles has respected the scope, as shown in the following image:



The level and scope of management access with RBAC

Further reading

For more on the topics we have covered in this chapter, read the following links:

Adopting Azure is the first stage in organizational maturity for an enterprise. By the end of this stage, people in your organization can deploy simple workloads to

Azure: https://docs.microsoft.com/en-us/azure/architecture/cloud-adoption-guide/adoption-intro/overview

When deploying enterprise workloads to the cloud, Azure Virtual Datacenter is a great approach that helps IT organizations and business units balance governance but with developer agility. I recommend you read: https://azure.microsoft.com/en-us/resources/azure-virtual-datacenter/en-us/

The following is a great guide on how to identify and plan the migration of applications and servers to Azure using the lift and shift method, minimizing any additional development costs while optimizing cloud hosting options in Microsoft Azure: https://azure.microsoft.com/en-us/resources/azure-virtual-datacenter-lift-and-shift-guide/en-us/

The naming rules and restrictions for Azure resources and a baseline set of recommendations for naming conventions: https://docs.microsoft.com/en-us/azure/architecture/best-practices/naming-conventions#naming-rules-and-restrictions

Implementing and Managing Azure Virtual Machines

Azure VMs, which are **virtual machines** (**VMs**) provided by Microsoft Azure, have many advantages over traditional on-premise physical computers. In Azure, we can choose the deployment model, which can be classic or ARM VMs, before deploying them. Since Microsoft retired all the resources in the ASM model in the latest updates of the exam and has become more and more focused on ARM, in this book, all the resources that we deploy in Azure are with the**Azure Resource Manager** (**ARM**) model.

n the following chapter, we'll learn the basics of Azure VMs—that is, how to plan and deploy them with the Azure Portal—and the ARM template via Azure Portal, Visual Studio, Azure PowerShell, and Azure CLI.

The principles of Azure VMs

Microsoft Azure provides cloud-based VMs to help users deploy their workloads with more control of the system, such as to host custom services and applications, which is more flexible

than other Azure services. This module introduces how to plan, deploy, and monitor Azure VMs in different ways.

Planning and deploying Azure VMs

Compared to a traditional on-premises virtual machine, a cloud-based VM allows you to quickly scale up and down as requested; you don't need to buy and maintain the physical hardware that runs the virtual machine. Another great advantage is that you only pay for what you use in the cloud. In practice you, still need to maintain the virtual machine by configuring, patching, and maintaining the software that runs on the VM. However, its flexible pricing makes it more efficient in terms of time and costs.

Identifying the workloads

Before deploying a virtual machine in Azure, we will always start by identifying the workloads, whether the best deployment solution for the target workloads is on Azure VM or maybe on other Azure offerings.

In real life, organizations thinking about migrating their existing application to Azure quickly should not only take into account the technical concerns but also the financial aspects. Besides, certain types of workloads are a great fit for hosting in an Azure IaaS environment, for example, when you need a high flexibility to control your OS and don't mind higher administration efforts than other PaaS offerings in Azure.

However, not every application is always a suitable fit for the cloud, as the following case :

- Certain low volumes or limited growth workloads where it might be cheaper to run the service or applications on commodity hardware on-premises
- Certain regulated environment workloads where the type of data is more sensitive or credentials requested by an organization needs to be kept on-premises or using other private cloud platforms such as Open Stack or the extension of the public cloud such as Azure Stack or VMware cloud on AWS.

Choosing the appropriate Azure VM sizing

If you decide to go further with Azure VM, you should choose the SKU of the virtual machine or the size of virtual machine of the Azure virtual machine provided by Microsoft Azure. In

Azure, the SKU or sizing is based on a variety of options for the number and speed of its processors (VCPU), amount of memory (RAM), the number of data disks you can attach to it, the maximum size of a temporary disk, IOPS, and the type of disks for the operating system. Generally, when the VM sizes support premium storage, which uses **solid-state drive** (**SSD**), the maximum aggregate disk I/O performance would be better than standard storage with a **hard disk drive** (**HDD**).

Virtual machines are available in several different sizes. When your requirements change, it is easy to resize the VM, which means you can use more advanced VM configurations, such as a more powerful CPU or larger RAM.

You can choose the appropriate size depending on your technical requirements. Try to balance the appropriate size of VMs and the number of VMs in your project. In real life, very often, the final decision on the size of VMs and number of instances for DevTest or the production environment would be made after a period of workload testing.

The following are the available categories of Azure virtual machines that are available so far:

Series	VM Size Family	User case
A Series	Entry-level economical VMs for dev/test	Development and test servers, low traffic web servers, small to medium databases, servers for proof-of-concepts, and code repositories.
B Series	Base level performance for workload, ability to burst CPU performance up to 100% of the CPU	Ideal for workloads that do not need the full performance of the CPU continuously, like web servers, small databases and development and test environments
D Series	General purpose compute	most applications, relational databases, in-memory caching, and analytics
Dv2 Series	Next generation general purpose compute	Ideal for applications that demand faster CPUs, better local disk performance or higher memories, and offer a powerful combination for many enterprise-grade applications
E Series	Optimized for in-memory hyper-threaded applications	SAP HANA, SAP S/4 HANA, SQL Hekaton and other large in-memory business critical workloads
F Series	Compute optimized virtual machines	Batch processing, web servers, analytics, and gaming
G Series	Memory and storage optimized virtual machines	Large SQL and NoSQL databases, ERP, SAP, and data warehousing solutions
H Series	High performance virtual machines	High performance computing, batch processing, analytics, molecular modeling, and fluid dynamics
L Series	Storage optimized virtual machines	NoSQL databases such as Cassandra, MongoDB, Cloudera, and Redis. Data warehousing applications and large transactional databases are great use cases as well
M Series	Largest memory optimized virtual machines	SAP HANA, SAP S/4 HANA, SQL Hekaton and other large in-memory business critical workloads requiring massive parallel compute power
N Series	GPU enabled virtual machines	Graphics rendering, video editing, remote visualization, high performance computing, and analytics

Azure VM storage options

Every Azure VM generally has two disks:

- An **operating system disk**, which is registered as a SATA drive and labeled as the C: drive for Windows machines and the repository /dev/sda for Linux machines, by default.
- A **temporary disk**, which is labeled as the D: drive for Windows that stores the pagefile.sys file. For a Linux machine, the repository /dev/sdb is the temporary disk.

Besides these two disks, Azure VMs can be attached to a number of **data disks**. The operating system disk is created from a VM image; the operating system disk and data disks are **virtual hard disks** (**VHDs**) stored in a page blob in an Azure storage account.

Managed disks versus unmanaged disks

Azure **unmanaged disks** are the disks created and managed by service administrators. Azure **Managed Disks** are the disks that allow Microsoft Azure to handle the disk management of the IaaS VMs.

Compared to unmanaged disks, Azure-managed disks provide better scalability while scaling the VMs with VMSS (scale sets) and breaks the limit of IOPS per storage account, that is, Azure has a limit of 20,000 IOPS per storage account which will impact the number of VMs that can be created per storage account. Azure Managed Disks are recommended by Microsoft for storing persistent storage of data while creating Azure VMs.

Both of them have standard and premium pricing tiers. A standard tier is based on HDD. A premium tier is based on high-performance SSD to support I/O intensive workloads.

Unmanaged disks are available for **locally redundant storage** (**LRS**), **zone-redundant storage** (**ZRS**), **geo-redundant storage** (**GRS**), and **read-access geo-redundant storage** (**RA-GRS**). At the time of writing this book managed disks are only available for LRS.

Azure reserved VM instances (RIs) versus pay–as–you–go instances

Compared to pay-as-you-go prices, Azure also provides a way to cut down VM costs in a significant way by purchasing the **reserved VM instances** (**RIs**), which link to a 1-year or 3-year engagement on both Windows and Linux VMs.

Purchasing Azure-reserved VM instances can help users save up to 72% VMs costs. Additionally, it accumulates the **Azure Hybrid Benefit**, which can help users save up to 82% VM costs.

Deploying an Azure VM

To facilitate the deployments of different workloads using an Azure virtual machine, Microsoft Azure offers different ways to release the deployment. Users can deploy their virtual machines via Azure Portal, Azure PowerShell, Azure CLI, Azure Cloud Shell, or ARM Template.

The deployment usually starts from choosing an OS image from the Azure Marketplace. The Azure Marketplace provides images of various Microsoft and Linux operating systems, such as CentOS, Debian, Ubuntu, and so on, and also provides preconfigured products with a **ready-to-use** image. Microsoft and Microsoft's third-party partner provides various popular image solutions, such as Windows Server 2016 Data Center, Red Hat Enterprise, SUSE Linux Enterprise, the Data Science Virtual Machine, and so on.

Creating Azure VMs

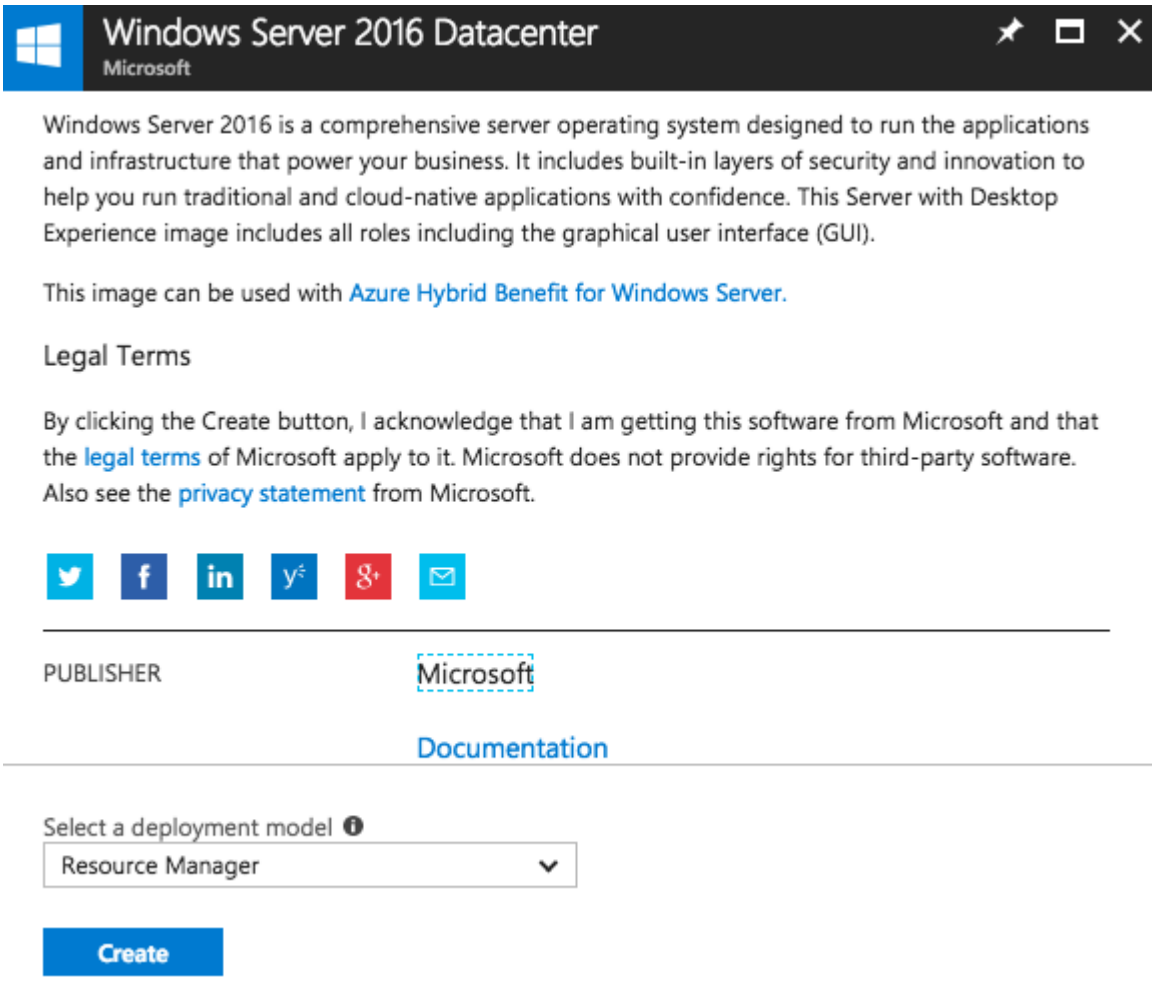
There are many ways to create an Azure VM, for example, via Azure Portal, Azure PowerShell, or Azure CLI.

Creating Azure VMs via the Azure Portal

Let’s take a look at how to create an Azure VM via the Azure Portal with a Windows image and a Linux image in the Azure Marketplace.

Creating an Azure VM with a Windows image

Via the Azure Portal, click on Create a resource and then on the Compute option, then choose the appropriate Windows image (**Windows Server 2016 Datacenter**, which is the latest one). The following screenshot shows the first page that will appear when you start to deploy an Azure Windows VM; be careful to choose the ARM as a deployment model, which will register your resource with Azure Resource Manager:

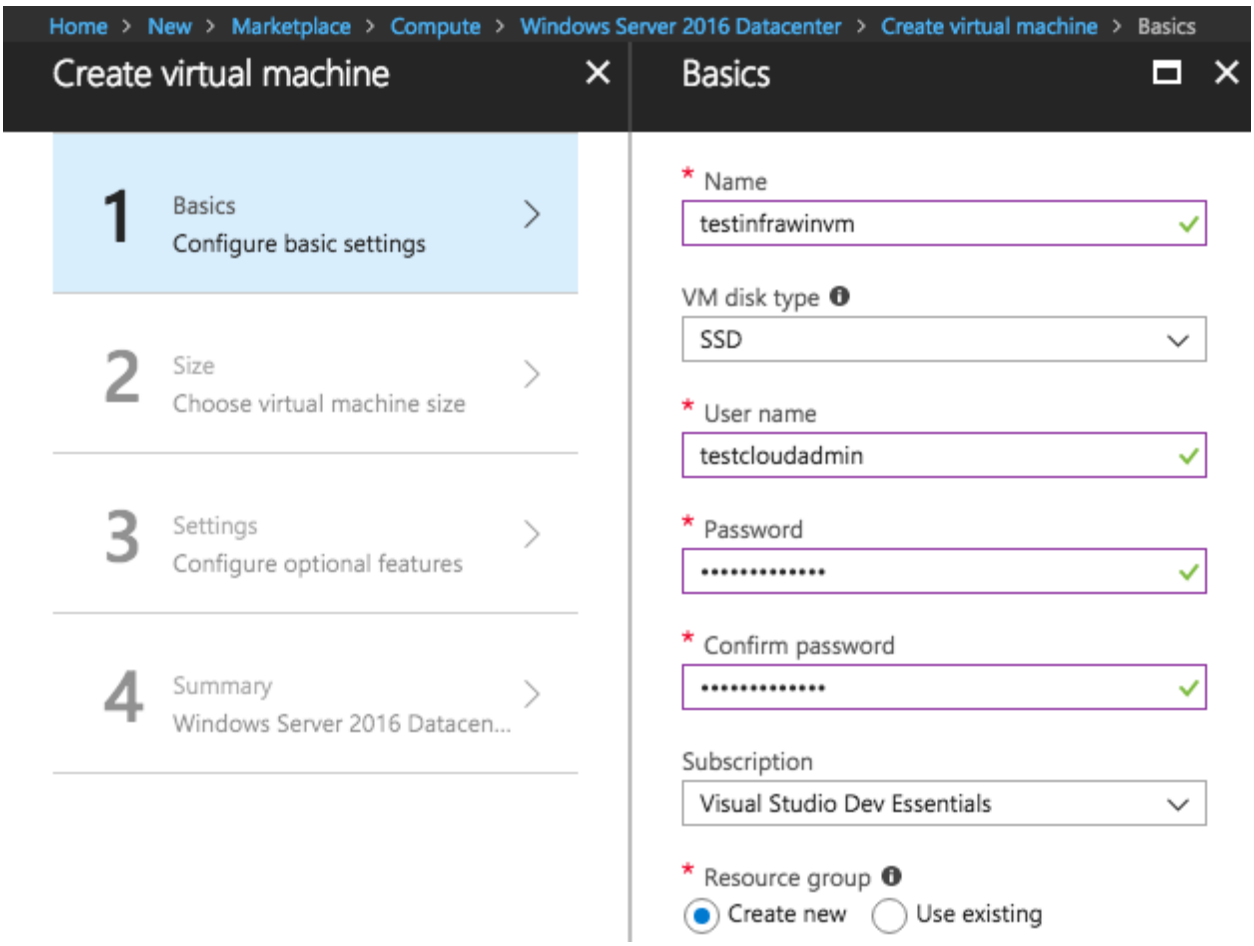


Choosing the appropriate deployment model to start deploying a windows VM

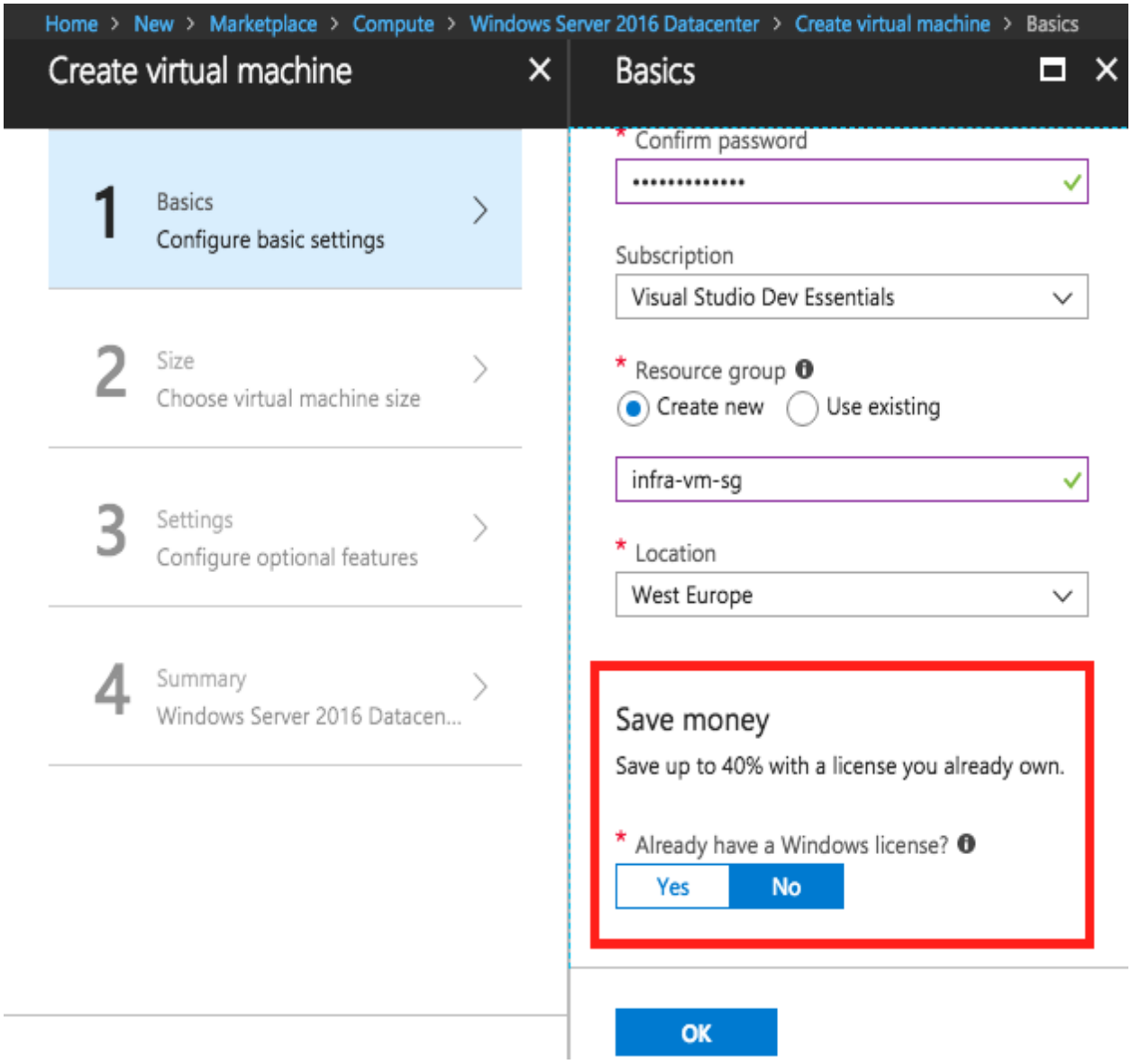
Then, you should fill in the necessary information in the **Basics** blade. The VM Disk type will affect the proposed pricing plan in the next step, which lets users choose the appropriate size of the VM. Azure provides the following types of disks:

- The **Premium disks (SSD)** are backed by SSDs, provide consistent, low-latency performance, and are ideal for I/O-intensive applications and production workloads
- The **Standard disks (HDD)** are backed by magnetic drives and are designed for applications with infrequently accessed data

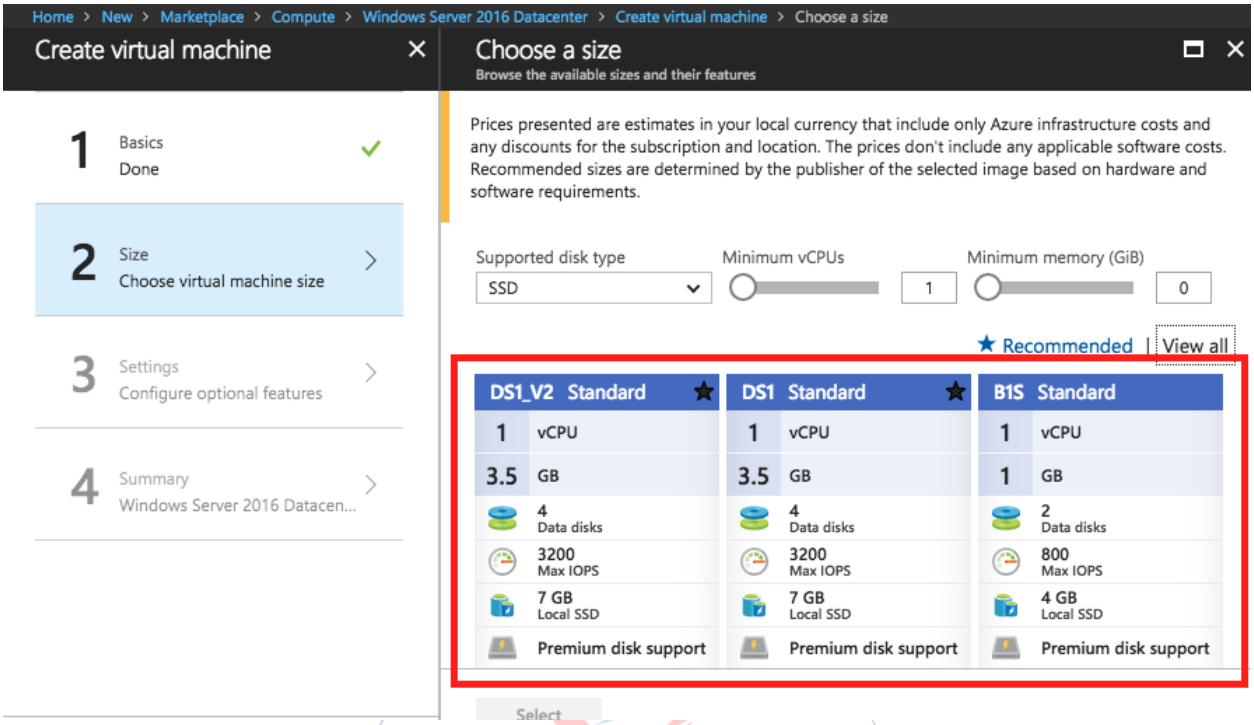
The **User name** and **Password** will be used to connect to the virtual machine. Usernames can be a maximum of 20-characters long, and the password must be at least 12-characters long and should have lower characters, upper characters, at least a digit, and a special character. While creating Azure VMs, users should pay attention to how to choose the most appropriate resource group, the subscription of the organization, and the location closest to your geography to reduce latency. The following screenshot is an example of the information in the **Basics** blade:



All Microsoft software installed in the Azure virtual machine environment must be licensed correctly. Microsoft Azure provides Azure Hybrid Benefit for the Windows server, which allows users to use on-premises Windows Server licenses and run Windows virtual machines on Azure at a reduced cost—this offer allows users to save up to 40% in costs. To obtain the benefits of Azure Hybrid, just confirm that you already have an on-premise license, as indicated in the following screenshot:

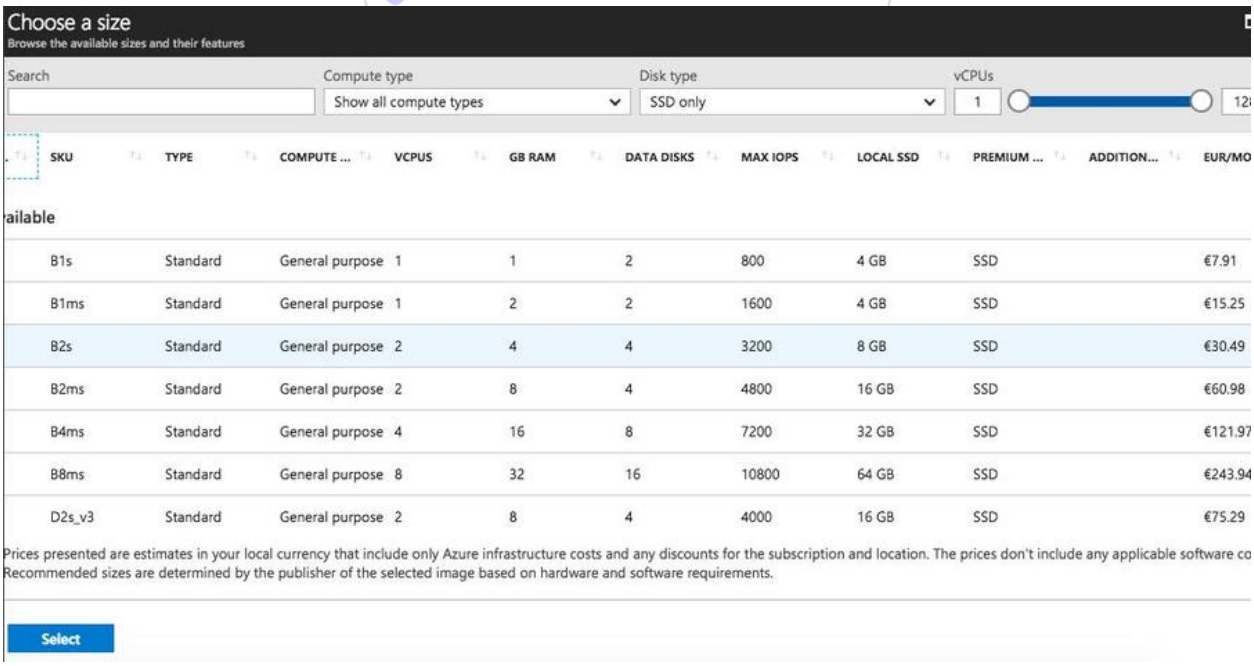


Microsoft Azure provides different pricing solutions with a range of predefined configuration options that correspond to different VM sizes. The different VM sizes indicate the different numbers and speed of its processors, different amounts of memory, a maximum number of network adapters or data disks that users can attach to it, and the maximum size of a temporary disk. As shown in the following screenshot, users should choose an initial VM size while deploying a new VM in Azure:



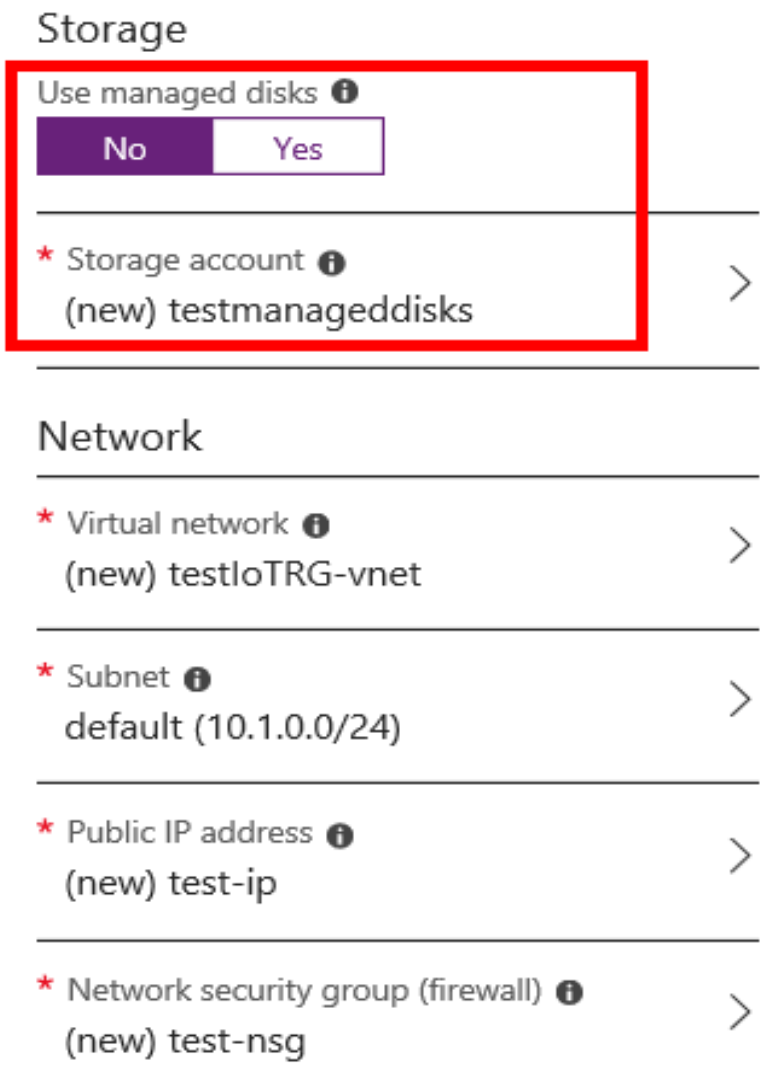
Choosing an Azure VM initial size

Here is a list view of the same:



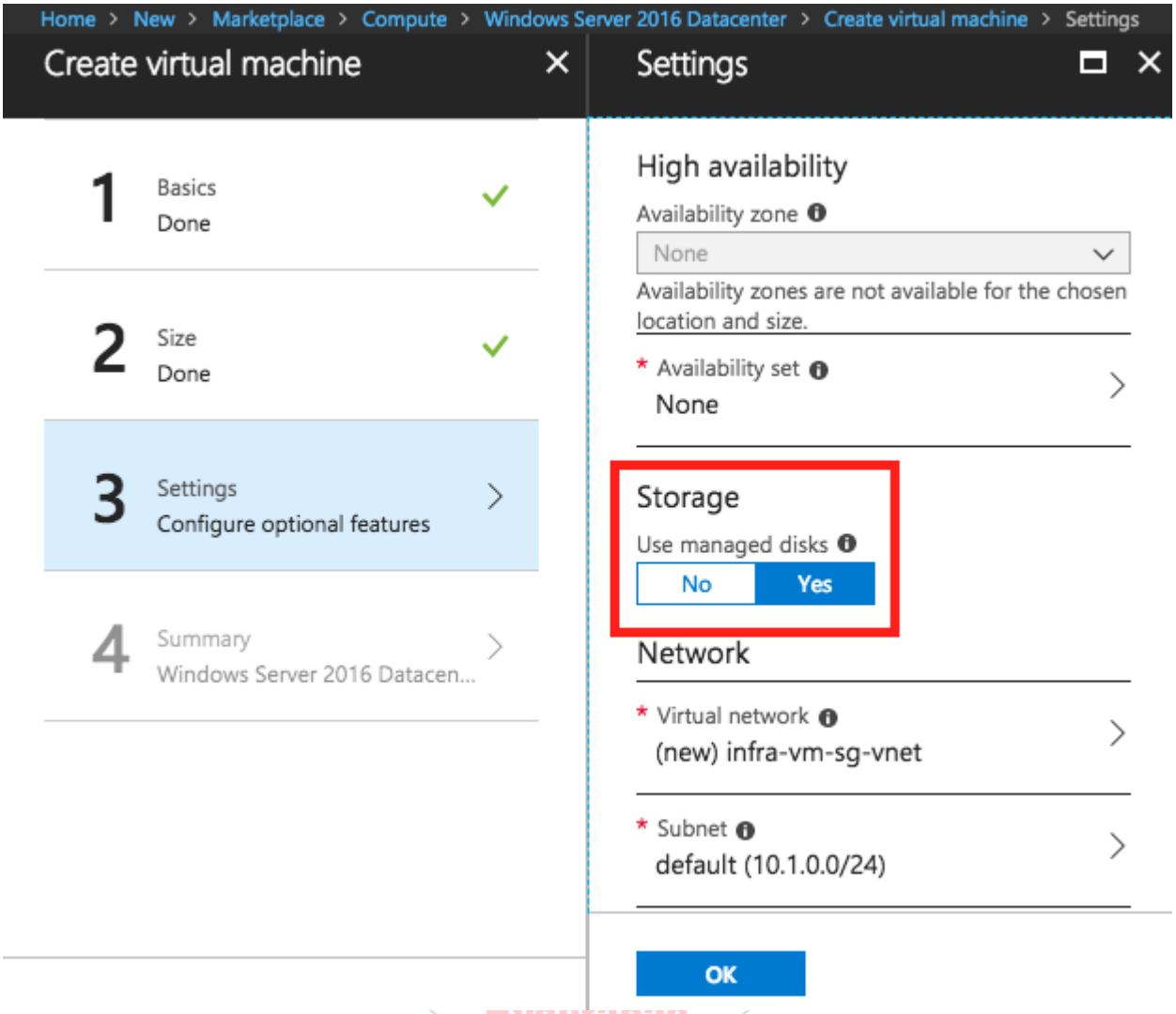
Choosing an Azure VM initial size (list view)

In the **settings** step, if you're not going to use the managed disk, you should specify a storage account to store disks, as shown in the following screenshot:



Choosing storage account while using non-managed disks

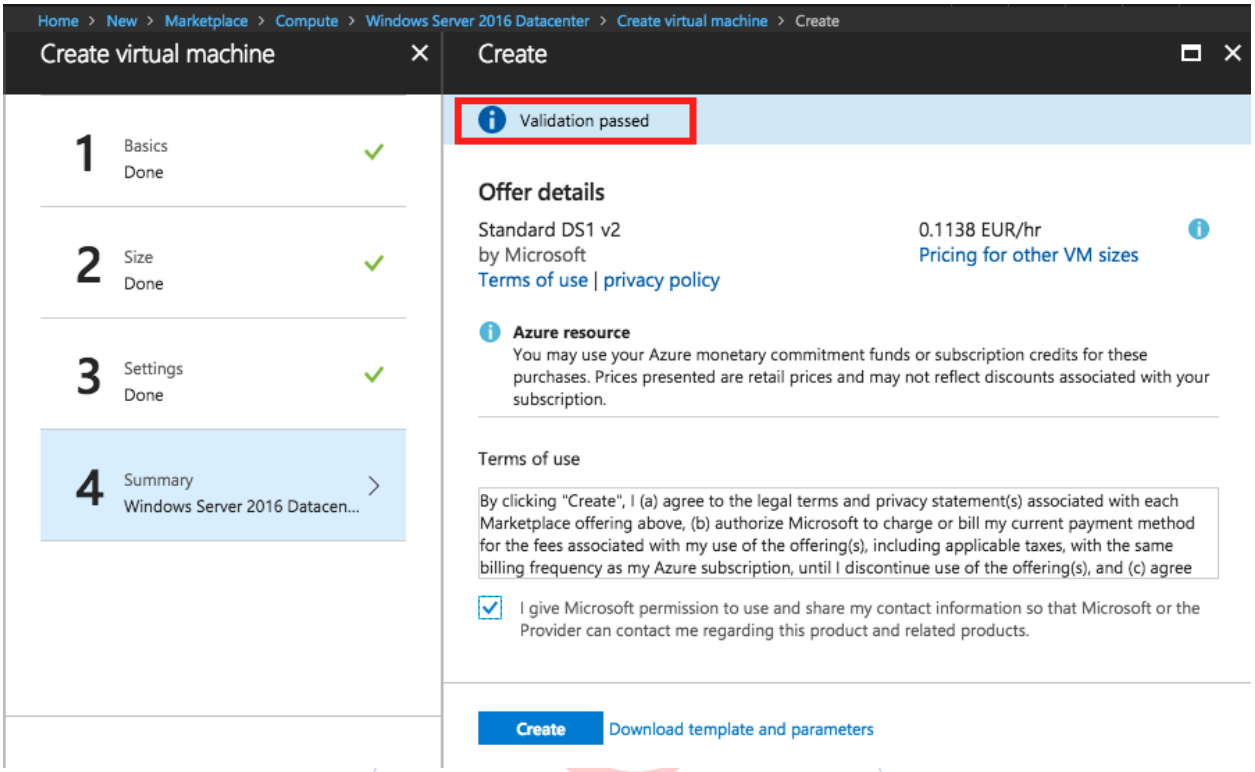
Select your storage option:



Choosing whether to use managed disks in Settings step

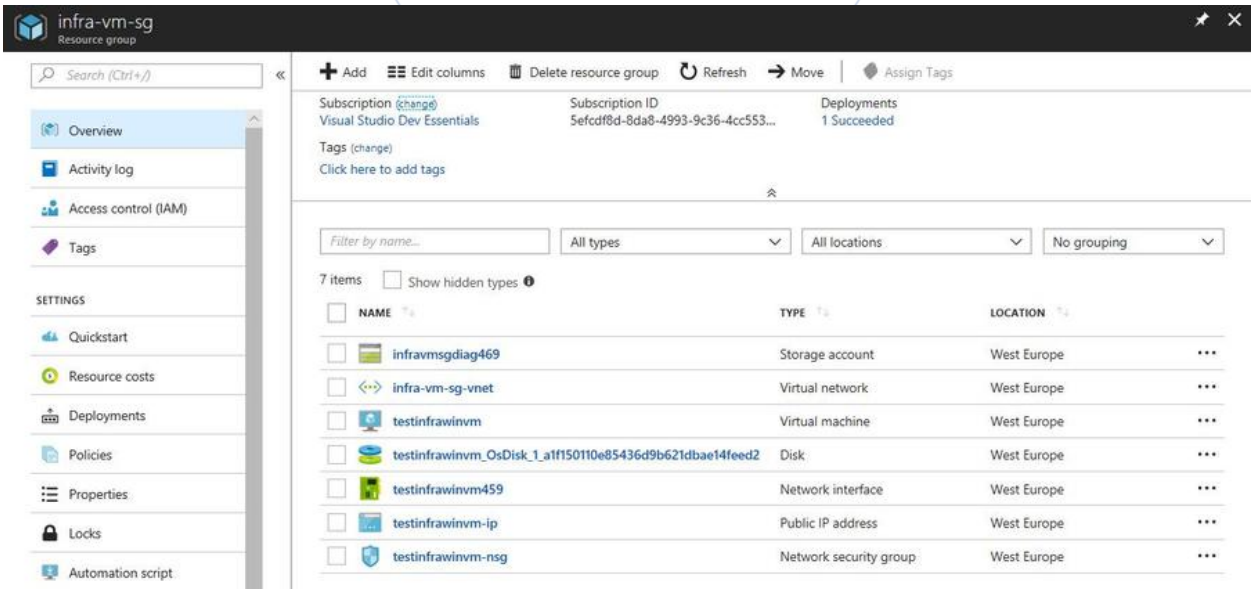
There are many options for creating a new Azure VM, such as **Virtual Network** (**VNet**), Subnet, and **Networks Security Group** (**NSG**). Microsoft Azure manages a default configuration for a predefined VM template that is ready to use. You can change these options while creating a new VM, depending on your intentions and requirements.

After filling in all the necessary information, Microsoft Azure will summarize and validate these details, as follows:



Azure VM deployment validation

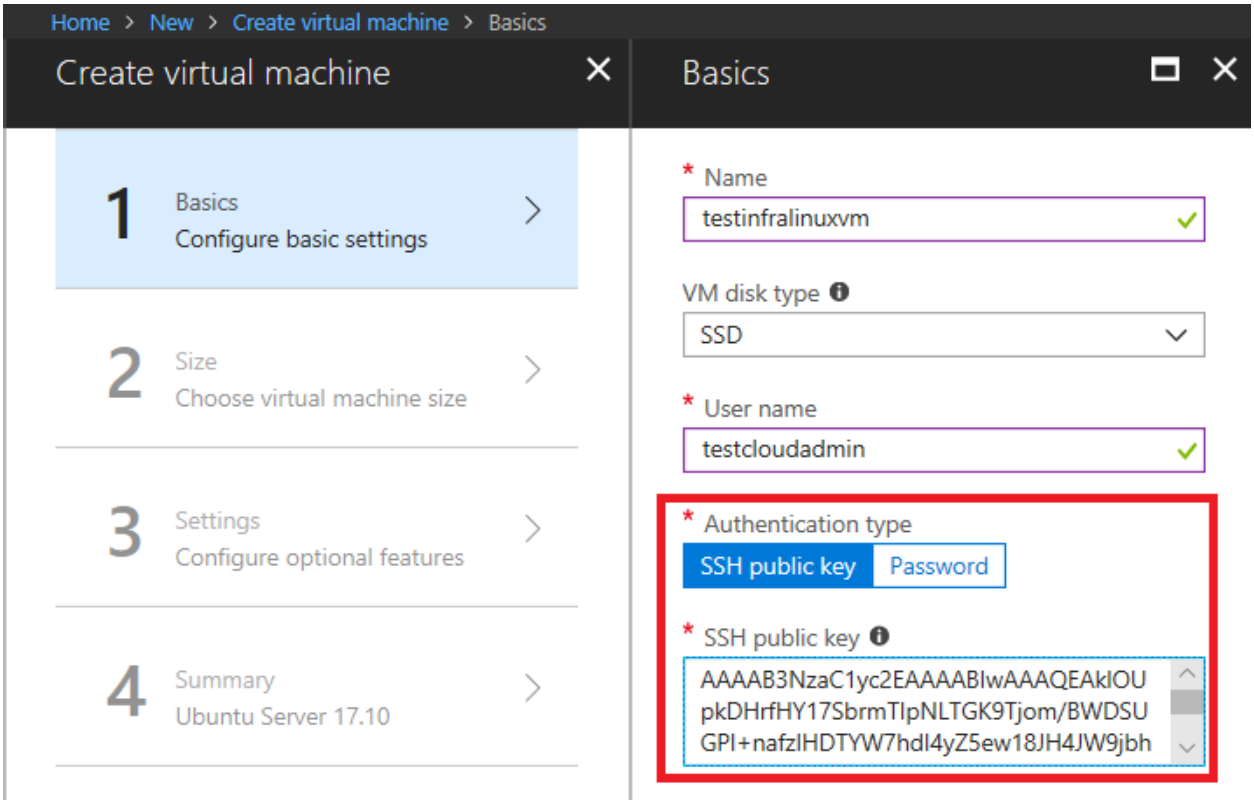
After deploying a Windows-based VM in Azure, the related resources such as vnet, nsg, and NIC are shown:



Related resources after a successful Azure VM deployment

Creating an Azure VM with a Linux distribution image

While creating a Linux-based VM, Microsoft provides almost the same options. A little different from the approach to create a Windows-based VM, a Linux-based VM is the authentication type, which means that Azure allows users to choose between the password-based and SSH public key–based authentication types while creating Linux-based Azure VMs, as follows:



Choose between the password-based and SSH public key–based authentication

Users should provide an RSA public key in the single line format (starting with ssh-rsa) or multiline PEM format (the multiline SSH key must begin with ---- BEGIN SSH2 PUBLIC KEY ---- and end with ---- END SSH2 PUBLIC KEY ----). You can generate SSH keys using ssh-keygen on Linux and macOS X, or PuTTYGen on Windows.

If you're using macOS or Linux, go to the following link to create your RSA key:

https://docs.microsoft.com/en-us/azure/virtual-machines/linux/mac-create-ssh-keys

If you're a Windows users, don't worry, go to the following link to get your RSA public key:

https://docs.microsoft.com/en-us/azure/virtual-machines/linux/ssh-from-windows

We will use the Azure Service Manager to deploy resources in Azure with the classic deployment model and the Azure Resource Manager to deploy resources in Azure with the ARM deployment model. That's why all the PowerShell commands in the ARM model are prefixed with RM.

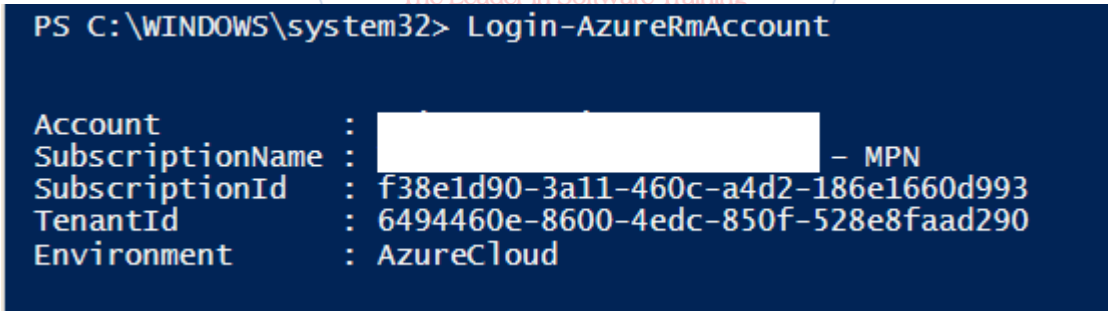
Let's begin now:

1. To sign in to Azure via Azure PowerShell or Windows PowerShell ISE, use the following command:

```
Login-AzureRmAccount
```

   Running the preceding command will display a popup to let you put in your Azure account name and password to pursue the authentication in Azure.

2. After logging into Azure with Azure PowerShell successfully, you'll get the following output:



   Output after login in Azure with Azure PowerShell successfully

3. Now, to get the list of Azure subscriptions associated with your account, use the following command:

```
Get-AzureRmSubscription
```

   You will get the following output after executing the preceding command:

```
PS C:\WINDOWS\system32> Get-AzureRmSubscription

Name      : Visual Studio Dev Essentials
Id        : 5efcdf8d-8da8-4993-9c36-4cc5534b9563
TenantId  : 6494460e-8600-4edc-850f-528e8faad290
State     : Disabled

Name      :
Id        : f38e1d90-3a11-460c-a4d2-186e1660d993
TenantId  : 6494460e-8600-4edc-850f-528e8faad290
State     : Enabled

Name      : Microsoft Azure
Id        : 659714ac-e6c2-4fc2-84b8-e2cf5d1a6eba
TenantId  : 8265bf70-d0b6-438b-b01d-fa623087ca84
State     : Enabled

Name      :
Id        : d9c16e03-bbea-45ed-b211-c99049536d51
TenantId  : 933c9cbe-35d3-4416-abbd-ddd1bca5879c
State     : Enabled

Name      :
Id        : 0e95efb5-2252-441b-953e-9754cc7d5433
TenantId  : 933c9cbe-35d3-4416-abbd-ddd1bca5879c
```

List of all subscriptions in the current tenant

4. If it is not the target subscription, use the following command to choose the target subscription:

Select-AzureRmSubscription -SubscriptionName "<subscription name>"

5. To set the subscription context, using the following command:

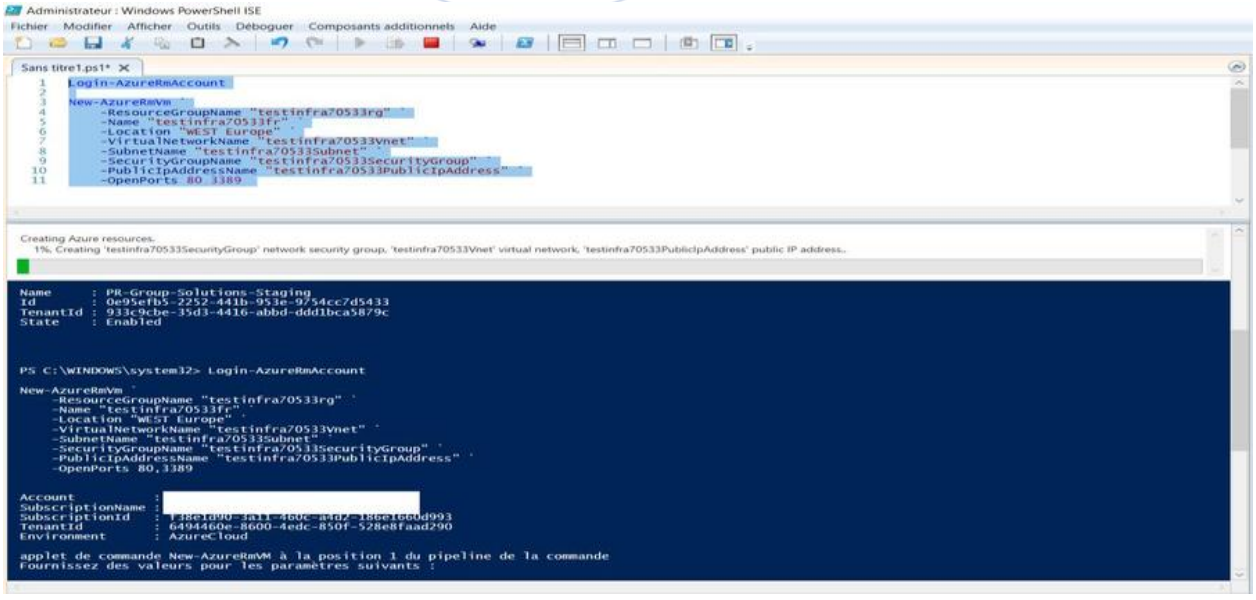
Set-AzureRmContext -SubscriptionId "<subscription id>"

6. To start the deployment of the Windows image, we should collect the related information regarding the deployment:
    1. The virtual network and its subnet
    2. Public IP address (optional)
    3. Network interface card (NIC)
    4. NSG with a rule allowing inbound RDP traffic (open inbound traffic for the port 3389 of your Azure VM)
    5. OS admin credentials (it is recommended to store them in a variable)

7. Then, create the Azure VM using the following command:

```
New-AzureRmVm
    -ResourceGroupName "testinfra70533rg"
    -Name "testinfra"
    -Location "WEST Europe"
    -VirtualNetworkName "testinfra70533Vnet"
    -SubnetName "testinfra70533Subnet"
    -SecurityGroupName "testinfra70533SecurityGroup"
    -PublicIpAddressName "testinfra70533PublicIpAddress"
    -OpenPorts 80,3389
```


After authentication, it will start to create resources in Azure, as shown in the following screenshot:



Creating Azure VMs via an Azure CLI

To create an Azure VM using a Windows image, for example, a Windows Server 2016 image via the Azure CLI, similar to Azure PowerShell, we should configure the login and subscription that we will use in Azure. You can use the following command:

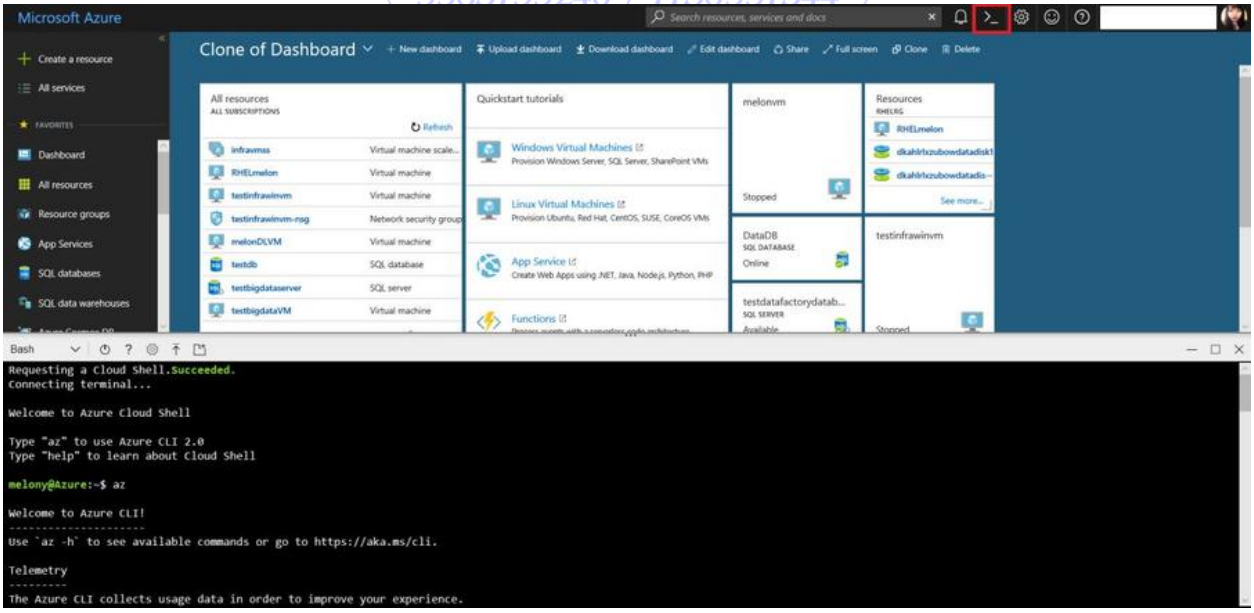
```
az login
```

Use the following command to set your subscription, and where you want, to deploy your Azure VM:

```
az account set –subscription <subscription name>
```

A great way to access Azure CLI is to use Azure Cloud Shell, which will let you always work with the latest Azure CLI commands without worrying about installing updates.

Launching the Cloud Shell via Azure Portal, as shown in the following screenshot:



Launching a Cloud Shell via Azure Portal

After launching Cloud Shell successfully, create a resource group using the following command:

```
az group create --name <resource groupname> --location <Azure region>
```

Specify the resource group that will host the Azure VM, its location following your intention. The example output will look like what's shown here in the following screenshot:

```
melony@Azure:~$ az group create --name testinfravmazurecli --location 'West Europe'
{
  "id": "/subscriptions/f38e1d90-3a11-460c-a4d2-186e1660d993/resourceGroups/testinfravmazurecli",
  "location": "westeurope",
  "managedBy": null,
  "name": "testinfravmazurecli",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null
}
```

Creating resource group via Azure CLI

To create the Azure VM, use the following command:

```
az vm create --resource-group <resource groupname> --name <VM name> --image <Azure Marketplaceimage> --generate-ssh-keys
```


Currently, the valid images contain CentOS, CoreOS, Debian, openSUSE-Leap, RHEL, SLES, UbuntuLTS, Win2016Datacenter, Win2012R2Datacenter, Win2012Datacenter, and Win2008R2SP1.

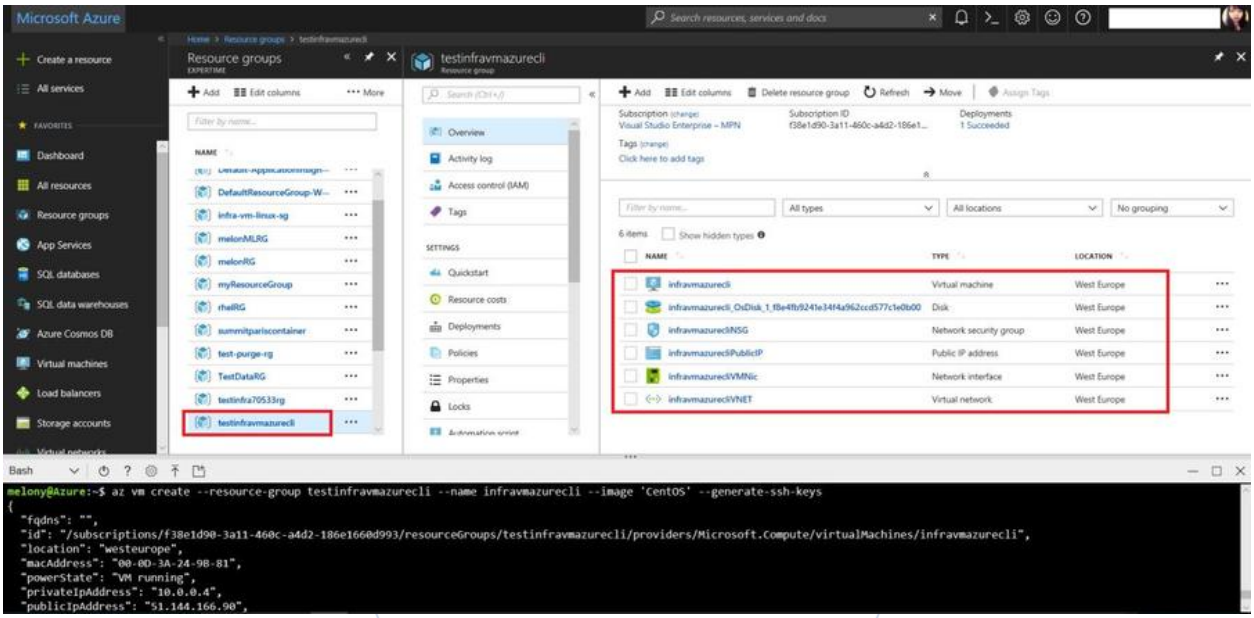
The following is an example command I launched to create a Azure VM by using the Azure CLI:

```
az vm create --resource-group testinfravmazurecli --name infravmazurecli --image 'CentOS' --generate-ssh-keys
```

The output of the preceding commands will be as follows, which means we have created a Linux VM via Azure CLI successfully:

```
melony@Azure:~$ az vm create --resource-group testinfravmazurecli --name infravmazurecli --image 'CentOS' --generate-ssh-keys
{
  "fqdns": "",
  "id": "/subscriptions/f38e1d90-3a11-460c-a4d2-186e1660d993/resourceGroups/testinfravmazurecli/providers/Microsoft.Compute/virtualMachines/infravmazurecli",
  "location": "westeurope",
  "macAddress": "00-0D-3A-24-9B-81",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.4",
  "publicIpAddress": "51.144.166.90",
  "resourceGroup": "testinfravmazurecli",
```

When returning to Azure Portal, we can find the resources that have been deployed in Azure successfully, as shown in the following screenshot:



Creating Azure VMs via the ARM template

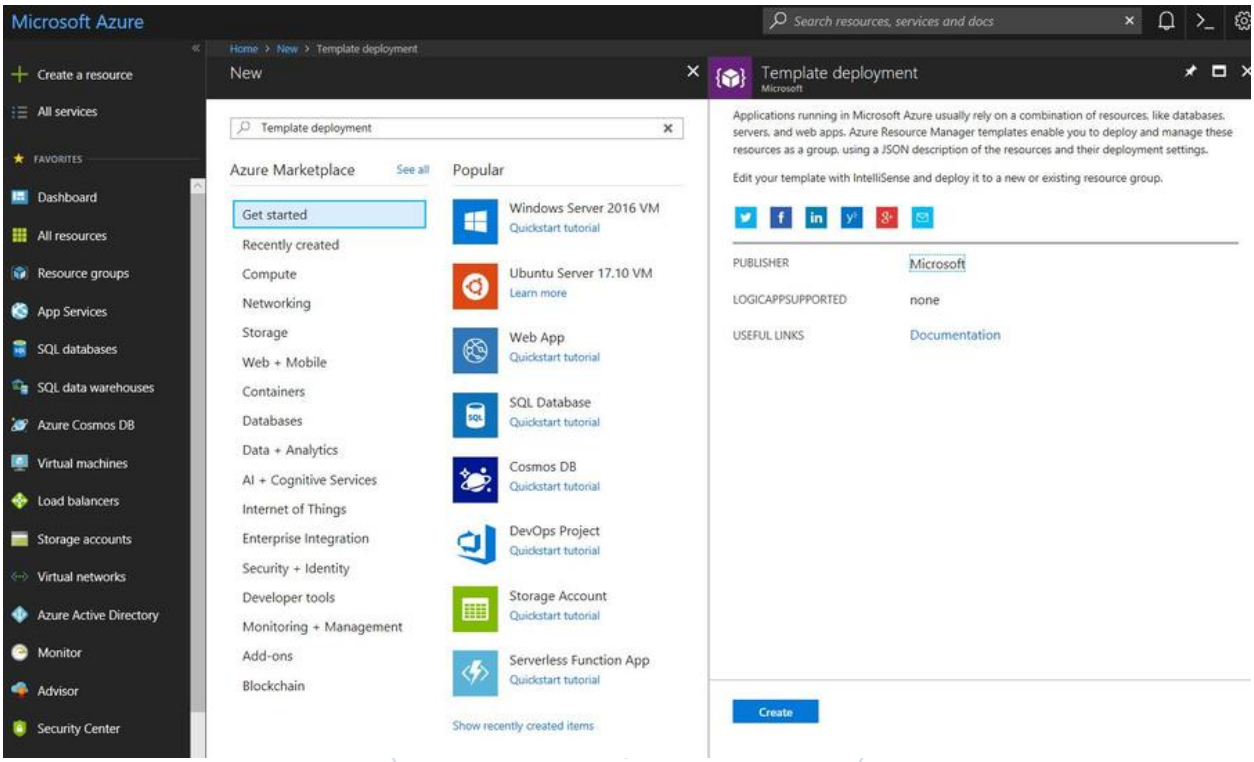
You can also create VMs using Azure Resource Manager templates, which facilitate the deployment process. Microsoft Azure provides different ways to deploy ARM templates, such as Azure Portal, Visual Studio, and Visual Studio Code. This capability is provided by Azure Resource Manager, which makes it possible to use a formatted **JSON file** and include definitions of all the Azure Resource Manager resources that are part of the deployment.

An ARM Template contains $schema, contentVersion, parameters, variables, resources, and outputs. $schema is defined by Microsoft Azure which is mandatory; other elements, except contentVersion, and resources, are optional. The following is a sample template to create a Linux-based Azure VM using managed disks:

```
"resources": [
   {
    "apiVersion": "2018-06-01",
    "type": "Microsoft.Network/publicIPAddresses",
    "name": "myPublicIPAddress",
    "location": "[resourceGroup().location]",
    "properties": {
     "publicIPAllocationMethod": "Dynamic",
     "dnsSettings": {
      "domainNameLabel": "testinfradns"
     }
    }
   },
   {
    "apiVersion": "2018-04-01",
    "type": "Microsoft.Network/virtualNetworks",
    "name": "myVNet",
    "location": "[resourceGroup().location]",
    "properties": {
     "addressSpace": { "addressPrefixes": [ "10.0.0.0/16" ] },
     "subnets": [
      {
       "name": "mySubnet",
       "properties": { "addressPrefix": "10.0.0.0/24" }
      }
     ]
    }
   },
   {
    "apiVersion": "2018-04-01",
    "type": "Microsoft.Network/networkInterfaces",
    "name": "myNic",
    "location": "[resourceGroup().location]",
    "dependsOn": [
     "[resourceId('Microsoft.Network/publicIPAddresses/', 'myPublicIPAddress')]",
     "[resourceId('Microsoft.Network/virtualNetworks/', 'myVNet')]"
    ],
    "properties": {
     "ipConfigurations": [
      {
       "name": "ipconfig1",
       "properties": {
        "privateIPAllocationMethod": "Dynamic",
        "publicIPAddress": { "id":
"[resourceId('Microsoft.Network/publicIPAddresses','myPublicIPAddress')]" },
```

```
          "subnet": { "id": "[variables('subnetRef')]" }
         }
        }
       ]
      }
     },
     {
      "apiVersion": "2018-04-01",
      "type": "Microsoft.Compute/virtualMachines",
      "name": "myVM",
      "location": "[resourceGroup().location]",
      "dependsOn": [
       "[resourceId('Microsoft.Network/networkInterfaces/', 'myNic')]"
      ],
      "properties": {
       "hardwareProfile": { "vmSize": "Standard_DS1" },
       "osProfile": {
        "computerName": "myVM",
        "adminUsername": "[parameters('adminUsername')]",
        "adminPassword": "[parameters('adminPassword')]"
       },
       "storageProfile": {
        "imageReference": {
"publisher": "OpenLogic",
         "offer": "CentOS",
         "sku": "7.4",
         "version": "latest"
        },
        "osDisk": {
         "name": "myManagedOSDisk",
         "caching": "ReadWrite",
         "createOption": "FromImage"
        }
       },
       "networkProfile": {
        "networkInterfaces": [
         {
          "id": "[resourceId('Microsoft.Network/networkInterfaces','myNic')]"
         }
        ]
       }
      }
     }
    ]
```

With the ARM Template, you can deploy it directly via Azure Portal. Start from **Create a resource** and then search **Template deployment** terms, and start to deploy ARM Template, as follows:



To deploy an ARM Template via Azure CLI, use the following commands:

```
az login

az group create --name TestInfra70533RG --location "West Europe"
az group deployment create \
  --name TestInfra70533Deployment \
  --resource-group TestInfra70533RG \
  --template-file storage.json \
  --parameters storageAccountType=Standard_GRS
```

Connecting to Azure VMs

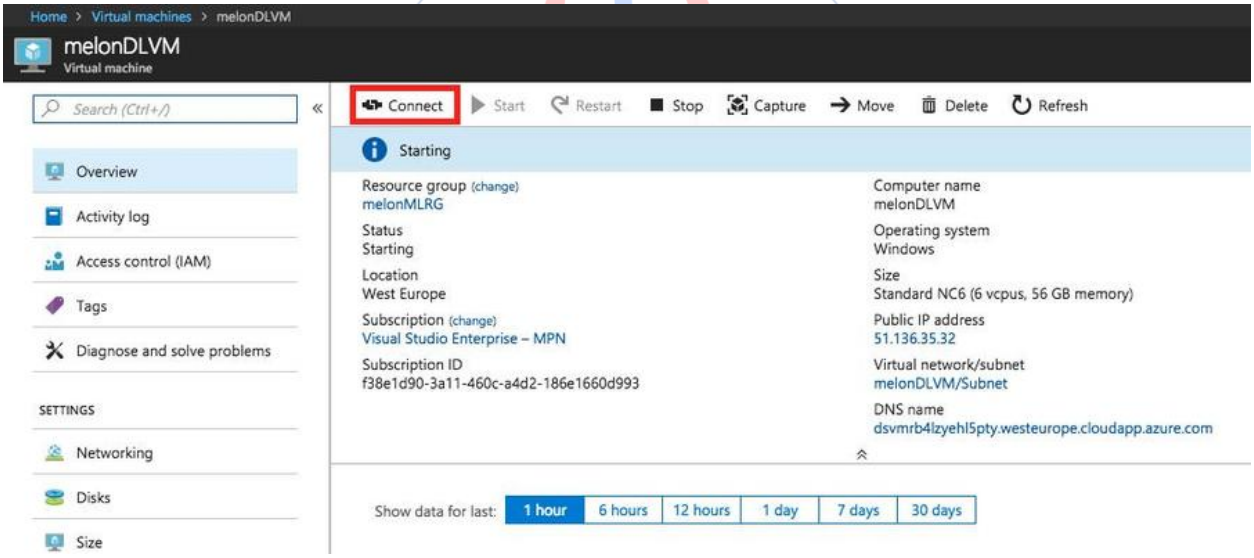
The default network security group that is attached to the virtual network of the deployed VM has the following NAT rules:

- Regarding Windows-based VMs: A rule which allows connectivity from the internet to port 3389
- Regarding Linux-based VMs: A rule which allows connectivity from the internet to port 22

Microsoft Azure provides three ways to connect to an Azure VM. The possible approaches are mentioned in the upcoming sections.

Connecting to a Windows Azure VM via Remote Desktop Protocol (RDP)

In the **Overview** blade, there is a **Connect** menu, as shown in the following screenshot, which allows you to download an RDP file after clicking on it; you can use it to connect to Windows Azure:

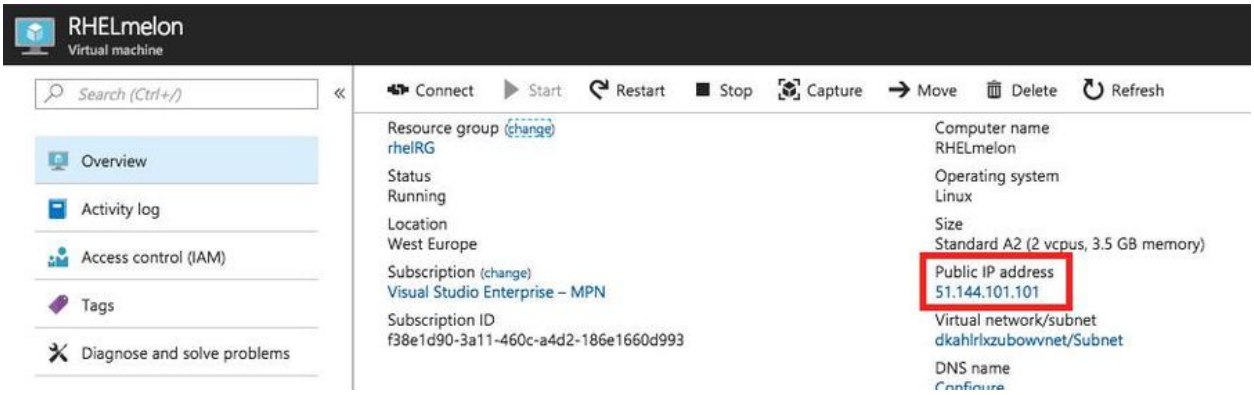


Connecting to a Linux Azure VM via Secure Shell (SSH)

Using the command line, you can connect to an Azure Linux VM:

```
ssh <yourAdminUsername>@<PublicIPOfYourVM>
```

You can find the **Public IP address** of your VM in the **Overview** blade:



The following is an example regarding how to connect to an Azure Linux VM using the command line:

```
melony — testcloudadmin@RHELmelon:~ — ssh testcloudadmin@51.144.101.101 — 88×...
MelonyQins-MacBook-Pro:~ melony$ ssh testcloudadmin@51.144.101.101
The authenticity of host '51.144.101.101 (51.144.101.101)' can't be established.
ECDSA key fingerprint is SHA256:t+L03cEKtu5mJOsq6SVrVTUEo+PjXBRWTzsBdoAUfhI.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '51.144.101.101' (ECDSA) to the list of known hosts.
Password:
Last login: Thu Mar  8 20:36:23 2018
[testcloudadmin@RHELmelon ~]$
```

Configuring Azure VMs in security

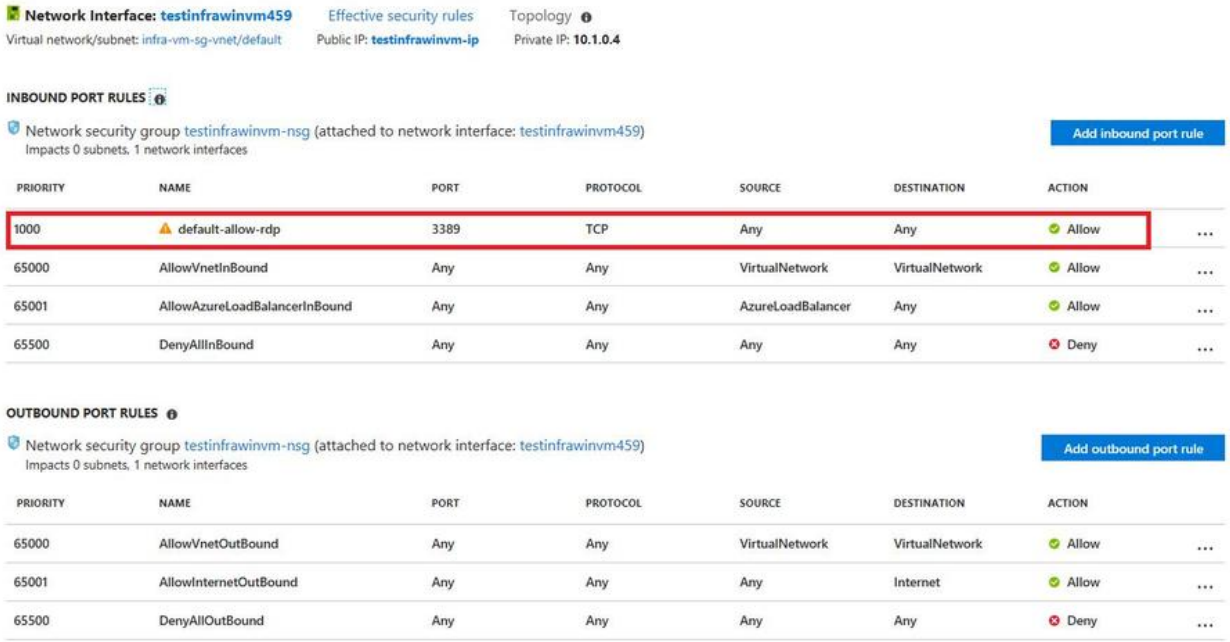
Microsoft Azure is engaged to guarantee data privacy and sovereignty, and enables their customers to control Azure-hosted data securely. The solutions provided by Microsoft take into account the potential business needs of their customers and give their customers the flexibility to choose the solution that fits best. To make sure that the Azure VM is secure, there are two aspects we always need to take care of. We will explain these in the following two subsections.

Restricting access to Azure VMs from the internet using NSG

An NSG, which is also known as a **network security group,** based on the Azure Virtual network, contains a list of security rules that allow or deny network traffic to resources connected to the same VNet. Besides, NSGs can be associated with subnets, individual VMs in the classic deployment model, or even individual network interfaces attached to VMs, such as

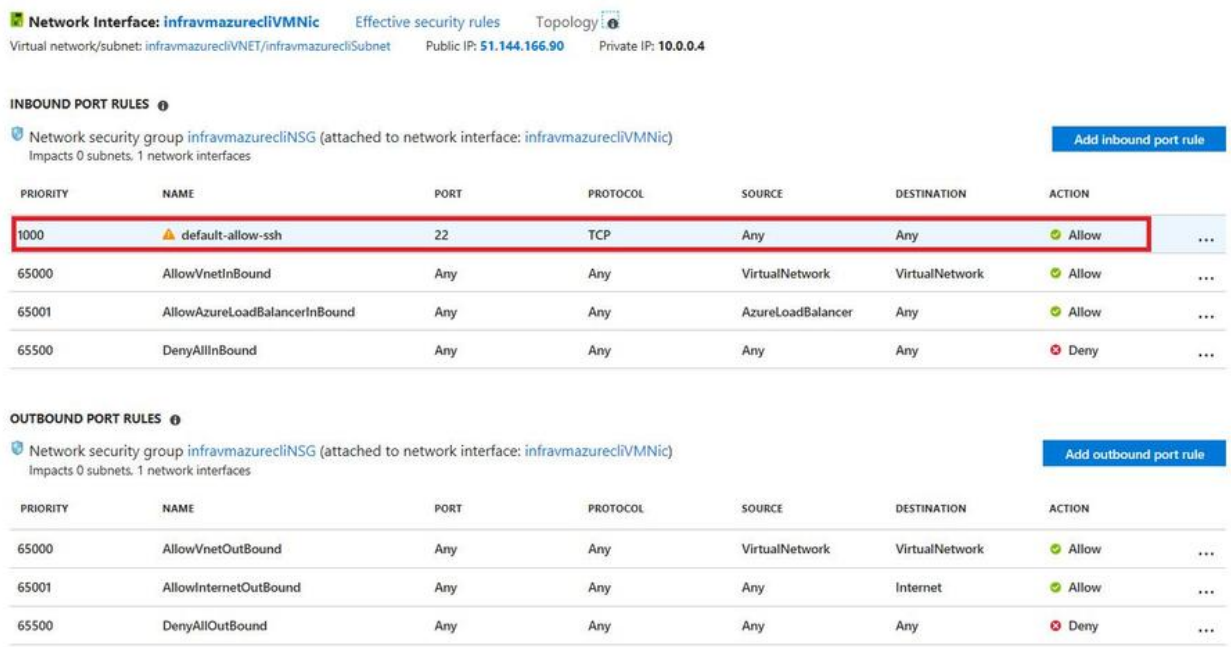
the **Resource Manager**. An example is that when an NSG is associated to a subnet, the rules apply to all resources connected to the subnet.

As we explained in the **Connecting to Azure VM** section of this chapter, to connect to an Azure VM successfully, there is a rule to allow inbound traffic in port 3389 for Windows-based VMs as follows:



Inbound port rule in NSG of Windows-based Azure VM

For port 22 Linux-based VMs, there is also an inbound port rule for port 22, as follows:



Network Interface: infravmazurecliVMNic Effective security rules Topology

Virtual network/subnet: infravmazurecliVNET/infravmazurecliSubnet Public IP: 51.144.166.90 Private IP: 10.0.0.4

INBOUND PORT RULES

Network security group infravmazurecliNSG (attached to network interface: infravmazurecliVMNic)
Impacts 0 subnets, 1 network interfaces

Add inbound port rule

PRIORITY	NAME	PORT	PROTOCOL	SOURCE	DESTINATION	ACTION
1000	default-allow-ssh	22	TCP	Any	Any	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

OUTBOUND PORT RULES

Network security group infravmazurecliNSG (attached to network interface: infravmazurecliVMNic)
Impacts 0 subnets, 1 network interfaces

Add outbound port rule

PRIORITY	NAME	PORT	PROTOCOL	SOURCE	DESTINATION	ACTION
65000	AllowVnetOutBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowInternetOutBound	Any	Any	Any	Internet	Allow
65500	DenyAllOutBound	Any	Any	Any	Any	Deny

Managing Azure VMs with VM Agent and VM extensions

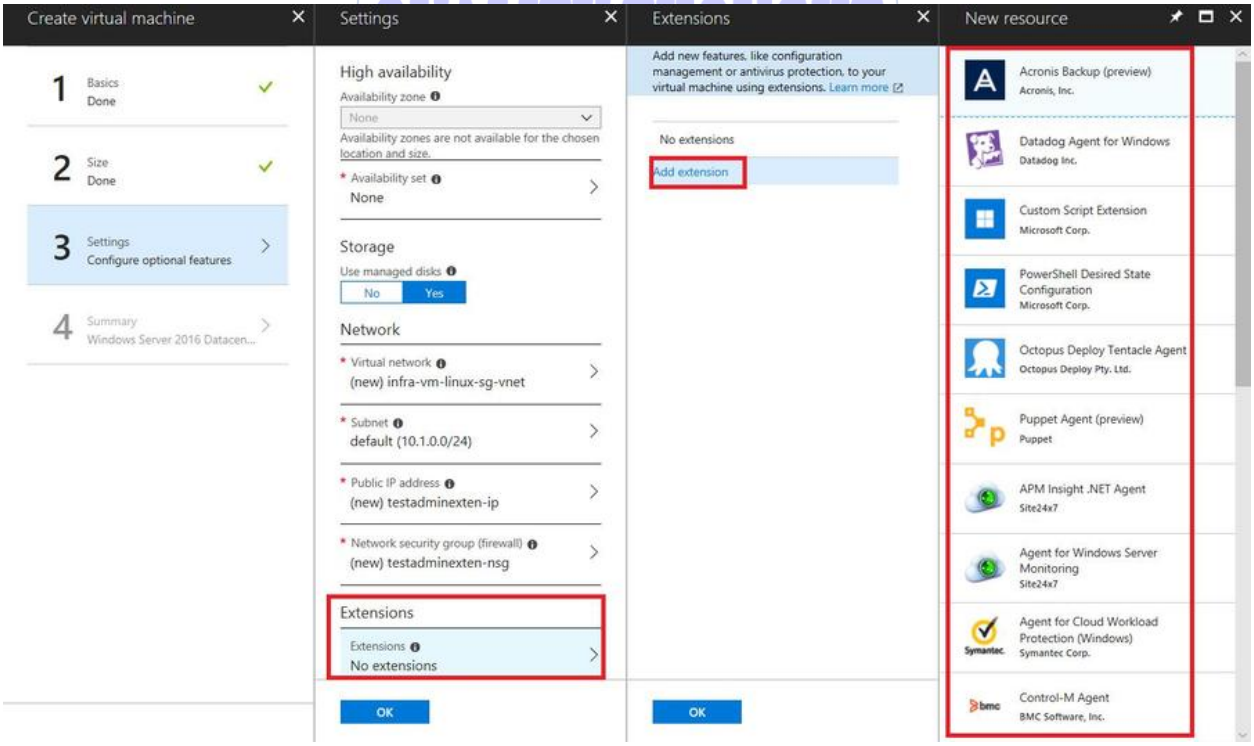
Azure offers a couple of methods that simplify and enhance the management of both Windows and Linux Azure VMs. Users can manage an Azure VM via Azure Portal, RESTful API, Azure PowerShell, Azure CLI, and so on. It is also possible to connect to Azure VMs when it is necessary to interact with an OS running within the VM.

When deploying Azure VMs, sometimes, we need to implement platform-specific configurations in VM. Azure provides a way to configure the OS and workloads running in the VM using a software component, which is known as the **Azure VM Agent**. The VM Agent supports **VM extensions**, which implement an additional functionality, especially in the areas of management, monitoring, and security terms.

Based on the VM Agent, users can add VM extensions. The following are some commonly used VM extensions:

- **Azure VM Access extension** enables you to reset local administrative credentials, fix misconfigured RDP settings on Windows VM, reset the admin password or SSH key, fix misconfigured SSH settings, create a new sudo user account, and check disk consistency on Linux VM.
- **Chef Client and Puppet Enterprise Agent** integrate Windows and Linux VMs into cross-platform Chef and Puppet enterprise management solutions.
- **Custom Script extension for Windows and Linux** makes it possible to run custom scripts within Azure VMs to apply custom configuration settings during VM provisioning. The extension supports any scripting language that the OS supports, such as Python or Bash.
- **DSC extension for Windows and Linux** implements a script-based or template-based configuration of OS components and applications.
- **Docker extension** facilitates automatic installation of Docker components, including the Docker Daemon, Docker Client, and Docker Compose on Linux VMs, and simplifies the process of implementing and managing containerized workloads in a significant way.

You can find out how to install the extension for your Azure VM by choosing **Extensions** while deploying a new VM; alternatively, after deploying, you can add the extension, as described in the following screenshot:



If your Azure VM has been already created, you can also go to the **Extension** blade and try to deploy a new extension by clicking **Add extension**.

Configuring the availability and scalability of Azure VMs

Let's take a look at managing Azure VMs regarding two aspects: availability and scalability. While implementing the Azure VM, it is important to make sure that workloads based on Azure are resilient and deal with all possible hardware failures.

Scaling Azure VMs

Generally, there are two kinds of scaling in the cloud:

- **Vertical scaling** is also called as **scale up**. Vertical scaling increases the capacity of existing hardware or software by adding compute resources, such as CPU memory-based processing power to a server, to make it faster. In this context, it means that users can scale by changing the VM's size.
- **Horizontal scaling** is also called as**scale out**. Horizontal scaling is used to increase the number of multiple entities so that they can handle more incoming requests while scaling in the case of peak time, such as Black Friday. In this context, it means that users can scale by increasing or decreasing the number of VMs that reside in the same Availability Set and share their load through internal or external load balancing. To implement horizontal scaling of Azure VMs, Azure virtual machine scale sets (scale sets) would be a great choice.

In terms of improving the availability of Azure VMs, horizontal scaling is more desirable than vertical scaling because changing the VMs' size will cause the shutdown of Azure VMs.

A possible approach to improve Azure VM’s scalability is presented in subsequent sections.

Configuring scale up by resizing Azure VMs

After deploying an Azure VM, it is possible to resize it when needed via Azure Portal, Azure CLI, or Azure PowerShell. At the Azure Portal, go to the Azure VM that you’ve deployed. There is a **Size** option in the blade and click on it. You’ll note the potential Azure VMs size for your consideration.

After choosing the size you want, you can click on **Select** to start a resize deployment. You can achieve the same results via Azure CLI and Azure PowerShell.

You can check the link https://docs.microsoft.com/en-us/azure/virtual-machines/windows/resize-vm regarding the Azure documentation to learn how to resize a Windows Azure VM by using Azure PowerShell. In addition, you can learn to resize a Linux Azure VM by using Azure CLI. This is provided at the following link: https://docs.microsoft.com/en-us/azure/virtual-machines/linux/change-vm-size.

Configuring scale out by deploying ARM VM scale sets (VMSS) and configuring ARM VMSS auto-scale

Microsoft Azure provides facilities with the workloads, such as deploying a set of identical VMs while they have identical configurations and deliver the same functionality to support a service or application using **virtual machine scale sets** (**VMSS**), or VM scale sets, for short. With VM scale sets, users can manage the scalability of VMSS by increasing or decreasing the number of VMs or resizing the VMSS which will resize the instances in VMSS, in an easy way. Another consideration is to facilitate the management of the availability of VMs in the VM pool which can deal with incoming requests in a flexible manner.

There are two basic ways to configure VMs deployed in a scale set:

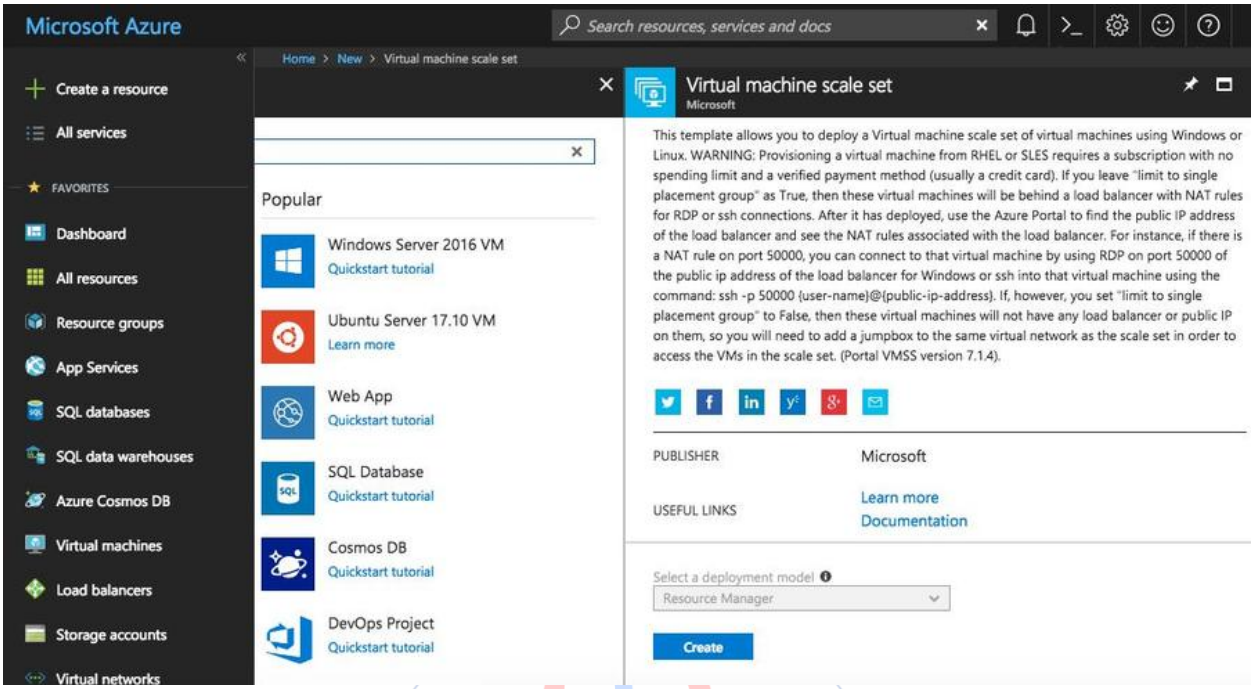
- Building your custom image before provisioning VMs with VMSS
- Configuring the VM using VM extensions after provisioning them

VMSS integrates the Azure load balancer or application gateway to handle dynamic distribution of network traffic across multiple VMs. It supports NAT rules so that users can connect to individual instances in VMSS.

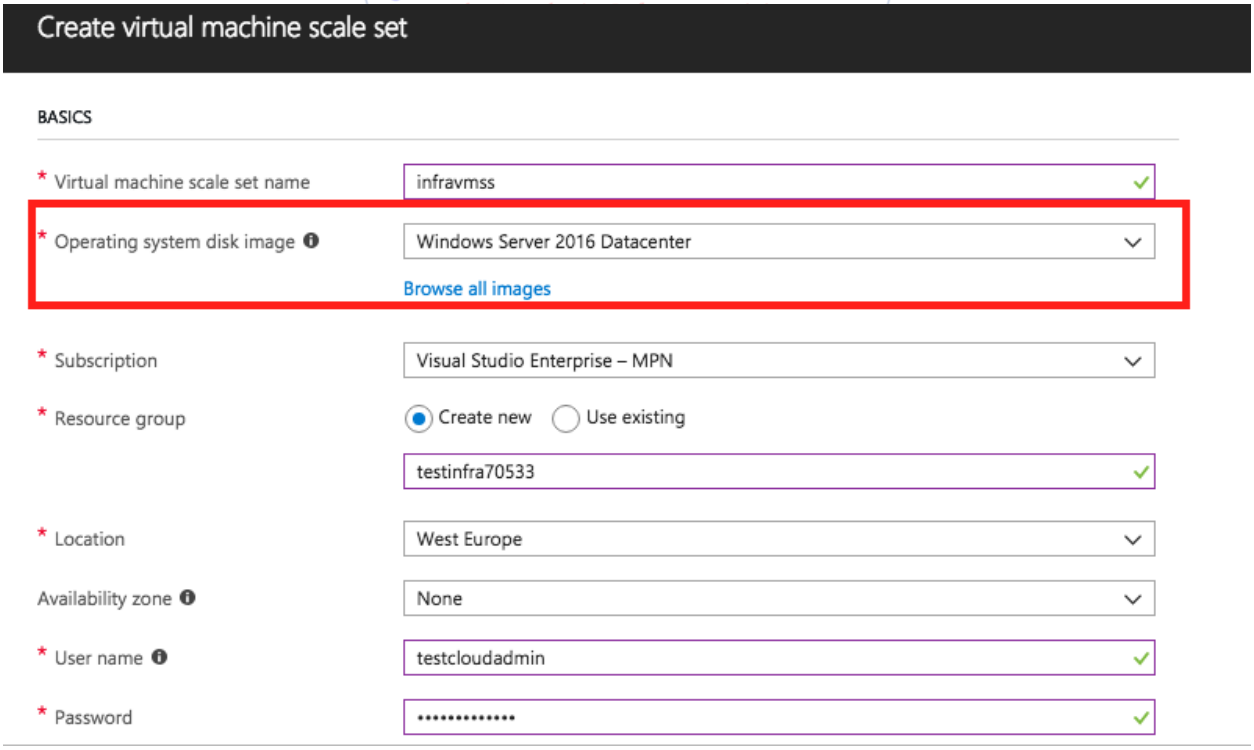
Deploying VMSS via Azure Portal

To create a VMSS via Azure Portal perform the following steps:

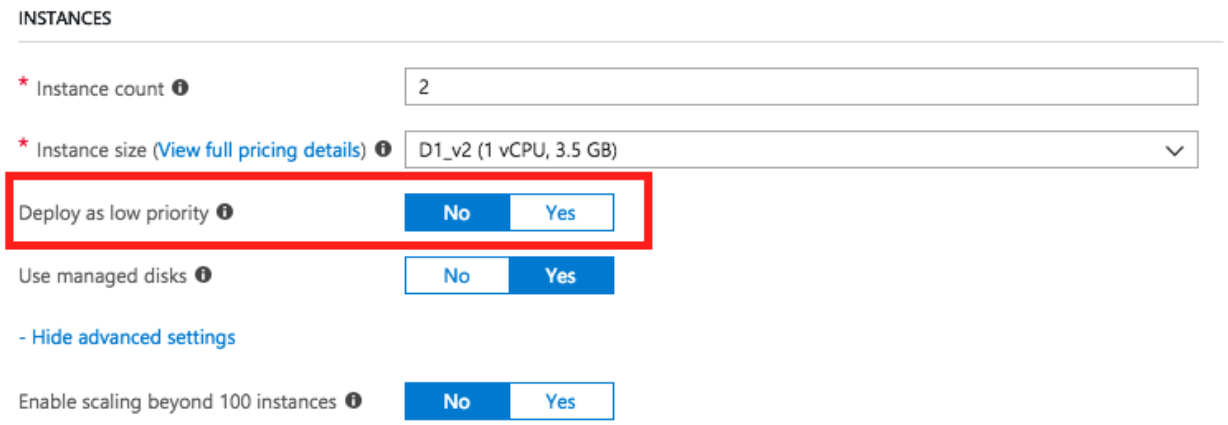
1. You can search Virtual machine scale set, then click on **Create**. As shown in the following screenshot, VMSS is only available in the ARM model, which means youcan only select**Resource Manager**as the deployment model:



2. While filling in the information in the **Basic** blade, you can choose the operating system disk image you want to deploy in your dedicated virtual machine.
3. Microsoft Azure provides thousands of OS images in the Azure Marketplace. Similar to the user name and password, it will be applied to every deployed instance in VMSS:

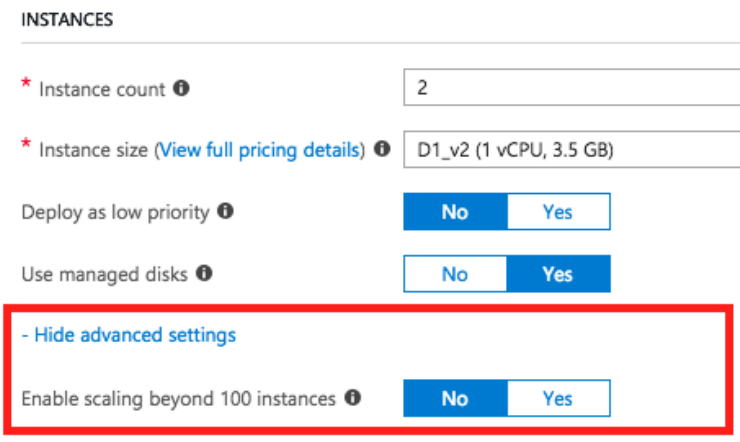


4. Click on **Browse all images**—you'll note a list of OS images from the Azure Marketplace.
5. In the **INSTANCES** section, you can deploy your scale set as low priority (as shown in the following screenshot), which can help you save up to 80% over usual on-demand costs, especially while deploying stateless workloads, batch processing jobs, or large compute workloads, because the VMs in the scale set may be evicted when they're not in use:



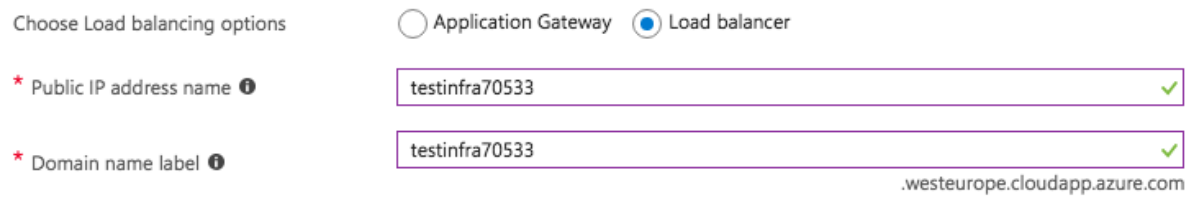
Note the instance count, which is the number of virtual machines in the scale set. It ranges from 1 to 100, which is much less than the maximum size of VMSS with the capacity of 1,000 VMs because, by default, the placement Group (not to scale more than 100 instances) was set to No. This means that the scale set will be limited to one placement group with a maximum capacity of 100.

6. Choosing **Yes** allows the scale set to span Placement Groups. This will enable the scaling beyond 100 and changes the availability guarantees of the scale set at the same time:



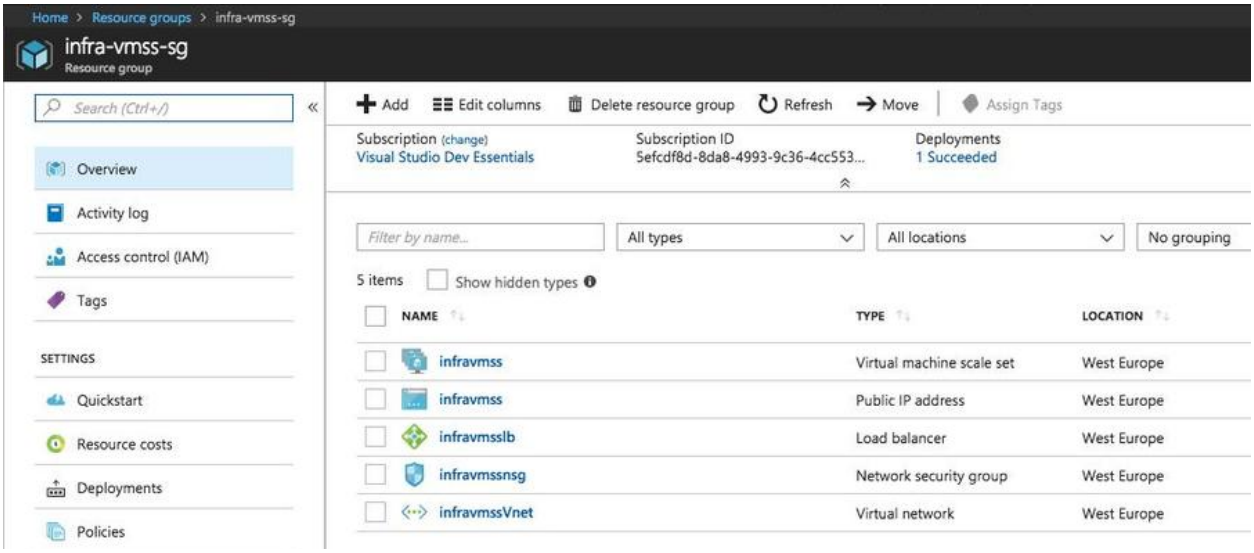
7. In the networking section, there are two options to manage web traffic while deploying VMSS: **Application Gateway**and **Load balancer**.
8. Choose **Azure Load balancer** for VMSS as a load balancer, which allows you to scale web applications and improve high availability. In this case, you should fill in the public IP address name and FQDN for the load balancer in front of the scale set.

FQDN, **fully qualified domain name**, is a domain name that specifies its exact location in the tree hierarchy of the **Domain Name System** (**DNS**), which must be unique across all of Azure. You should enter an appropriate name that adapts to the individual situation. The following is an example of this:



Azure Application Gateway also acts as a web traffic load balancer. It is a dedicated virtual appliance providing the **application delivery controller** (**ADC**) as a service. It is an OSI layer 7 (application layer) load balancer, which can perform more specific functions than the traditional OS Layer 4 load balancer.

Once you've confirmed that all the information you have entered on the **Summary**blade is correct, you can click on **OK**to start the scale set deployment. The deployment will last about a couple of minutes. After the deployment, we can see that there are some related resources in the same resource group of the VMSS. We have created a public IP for the load balancer and a virtual network for our VMSS, as shown in the following screenshot:



Deploying VMSS using the ARM Template

You can also deploy the VMSS using the ARM Template. You can define your VMSS as follows:

```
{
"type":"Microsoft.Compute/virtualMachineScaleSets",
"name":"[variables('namingInfix')]",
"location":"[resourceGroup().location]",
"apiVersion":"2018-04-01",
"dependsOn":[
"[concat('Microsoft.Network/loadBalancers/', variables('loadBalancerName'))]",
"[concat('Microsoft.Network/virtualNetworks/', variables('virtualNetworkName'))]"
],
"sku":{
"name":"[parameters('vmSku')]",
"tier":"Standard",
"capacity":"[parameters('instanceCount')]"
},
"properties":{
"overprovision":"true",
"upgradePolicy":{
"mode":"Manual"
},
"virtualMachineProfile":{
"storageProfile":{
"osDisk":{
"createOption":"FromImage",
"caching":"ReadWrite"
```

```
},
"imageReference":"[variables('imageReference')]"
},
"osProfile":{
"computerNamePrefix":"[variables('namingInfix')]",
"adminUsername":"[parameters('adminUsername')]",
"adminPassword":"[parameters('adminPassword')]"
},
"networkProfile":{
"networkInterfaceConfigurations":[
{
"name":"[variables('nicName')]",
"properties":{
"primary":true,
"ipConfigurations":[
{
"name":"[variables('ipConfigName')]",
"properties":{
"subnet":{
"id":"[concat('/subscriptions/', subscription().subscriptionId,'/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/virtualNetworks/',
variables('virtualNetworkName'), '/subnets/', variables('subnetName'))]"
},
"loadBalancerBackendAddressPools":[
{
"id":"[concat('/subscriptions/', subscription().subscriptionId,'/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('loadBalancerName'), '/backendAddressPools/', variables('bePoolName'))]"
}
],
"loadBalancerInboundNatPools":[
{
"id":"[concat('/subscriptions/', subscription().subscriptionId,'/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Network/loadBalancers/',
variables('loadBalancerName'), '/inboundNatPools/', variables('natPoolName'))]"
}
]
}
}
]
}
}
]
}
}
}
```

```
}
```

Deploying VMSS using Azure CLI

You can also deploy VMSS using the following Azure CLI command (replace the words between # with your own):

```
az vmss create -n #vmssname# -g #resourcegroupname# --instance-count #instancenumber# --image #vmimage# --data-disk-sizes-gb #gbsize#
```

To learn more about how to manage VMSS using Azure CLI, check out the following link: https://docs.microsoft.com/en-us/azure/virtual-machine-scale-sets/virtual-machine-scale-sets-manage-cli.

Deploying VMSS using Azure PowerShell

You can also deploy VMSS using the following Azure PowerShell command:

```
AzureRmVmss `
 -ResourceGroupName #resourcegroupname# `
 -Location #location# `
 -VMScaleSetName #vmssname# `
 -VirtualNetworkName #VnetName `
 -SubnetName #subnetname# `
 -PublicIpAddressName #publicIpAddressName# `
 -LoadBalancerName #lbname# `
 -UpgradePolicyMode #upgrademodel#
```

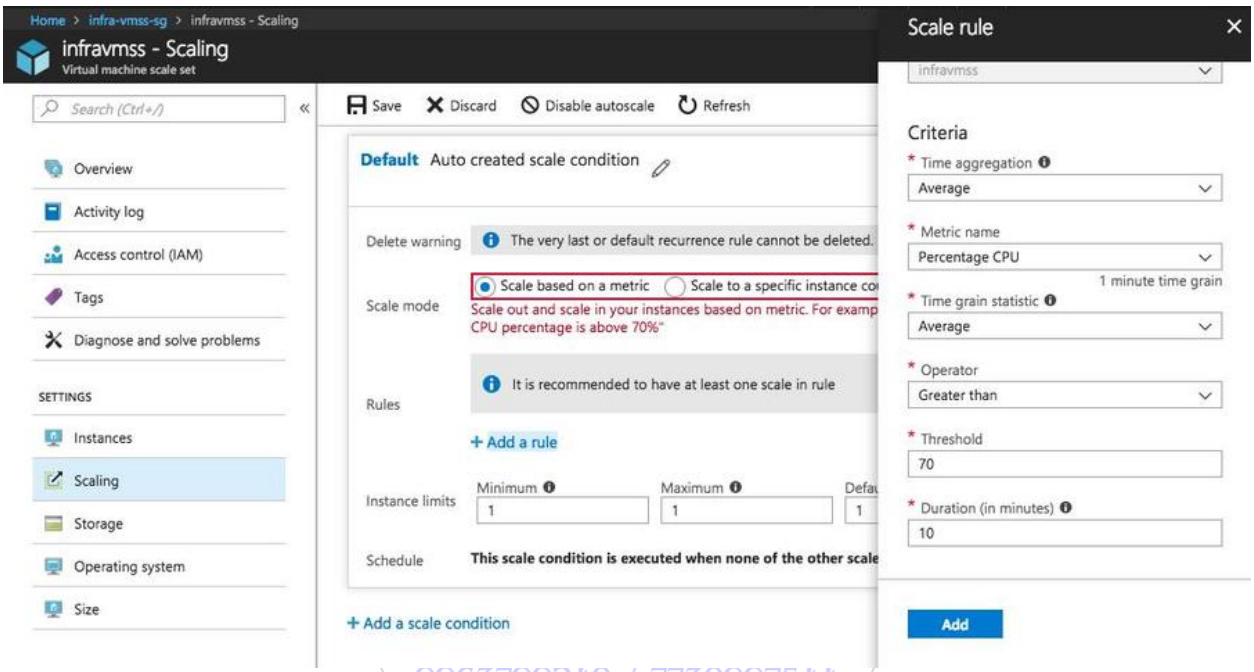

To learn more about how to manage VMSS using PowerShell, check out the following link: https://docs.microsoft.com/en-us/azure/virtual-machine-scale-sets/virtual-machine-scale-sets-manage-powershell

Configuring ARM VMSS autoscale

Even after the creation of VMSS, it is possible to adjust the autoscale condition for the VM in the deployed VMSS. You can also enable autoscale. There are two scale modes, as follows:

- Scale based on a metric
- Scale to a specific instance count

Configure the scaling rules, such as the **Minimum** or **Maximum** number of VMs and the CPU percentage threshold, which will take effect while specifying condition matching so that VMSS achieves scaling out or down. Make sure that every condition meets your intentions and finally click on **OK**:



It is also possible to use Azure Resource Explorer to preview your autoscaling condition ARM Template. You should add an autoscaling setting in the template, as follows:

```
{
"type":"Microsoft.Insights/autoscaleSettings",
"apiVersion":"2015-04-01",
"name":"autoscalewad",
"location":"[resourceGroup().location]",
"dependsOn":[
"[concat('Microsoft.Compute/virtualMachineScaleSets/', variables('namingInfix'))]"
],
"properties":{
"name":"autoscalewad",
"targetResourceUri":"[concat('/subscriptions/',subscription().subscriptionId, '/resourceGroups/',
resourceGroup().name, '/providers/Microsoft.Compute/virtualMachineScaleSets/',
variables('namingInfix'))]",
"enabled":true,
"profiles":[
{
```

```
"name":"Profile1",
"capacity":{
"minimum":"1",
"maximum":"10",
"default":"1"
},
"rules":[
{
"metricTrigger":{
"metricName":"Percentage CPU",
"metricNamespace":"",
"metricResourceUri":"[concat('/subscriptions/',subscription().subscriptionId,
'/resourceGroups/', resourceGroup().name,
'/providers/Microsoft.Compute/virtualMachineScaleSets/', variables('namingInfix'))]",
"timeGrain":"PT1M",
"statistic":"Average",
"timeWindow":"PT5M",
"timeAggregation":"Average",
"operator":"GreaterThan",
"threshold":60
},
"scaleAction":{
"direction":"Increase",
"type":"ChangeCount",
"value":"1",
"cooldown":"PT1M"
}
},
{
"metricTrigger":{
"metricName":"Percentage CPU",
"metricNamespace":"",
"metricResourceUri":"[concat('/subscriptions/',subscription().subscriptionId,
'/resourceGroups/', resourceGroup().name,
'/providers/Microsoft.Compute/virtualMachineScaleSets/', variables('namingInfix'))]",
"timeGrain":"PT1M",
"statistic":"Average",
"timeWindow":"PT5M",
"timeAggregation":"Average",
"operator":"LessThan",
"threshold":30
},
"scaleAction":{
"direction":"Decrease",
"type":"ChangeCount",
"value":"1",
```

```
"cooldown":"PT5M"
}
}
]
}
]
}
}
```

You can find a great example in GitHub for Linux-based VMSS in the following link: https://github.com/Azure/azure-quickstart-templates/tree/master/201-vmss-ubuntu-autoscale

You can also find a great example in GitHub for Windows-based VMSS in the following link: https://github.com/Azure/azure-quickstart-templates/tree/master/201-vmss-windows-autoscale

Managing Azure VM's availability

In terms of resilience in Azure, there are three major areas it encompasses:

- High availability
- Disaster recovery
- Backup

Some readers may be wondering, we usually talk about availability as being 99.95%, 99.9% — what does it mean? Actually, these percentages of availability are the major cause of failure that Microsoft Azure can guarantee. After calculation, they may mean that the downtime of a Azure VM may be affected in order to meet its **Service Level Agreements** (**SLA**) as shown in the following table:

Availability	Downtime per year	Downtime per month	Downtime per week	Downtime per day
90%	36.52 days	3.04 days	16.80 hours	2.40 hours
95%	18.26 days	1.52 days	8.40 hours	1.20 hours
99%	3.65 days	7.30 hours	1.68 hours	14.40 minutes
99.5%	1.83 days	3.65 hours	50.40 minutes	7.20 minutes
99.9%	8.77 hours	43.83 minutes	10.08 minutes	1.44 minutes
99.95%	4.38 hours	21.91 minutes	5.04 minutes	43.20 seconds
99.99%	52.59 minutes	4.38 minutes	1.01 minutes	8.64 seconds
99.999%	5.26 minutes	26.30 seconds	6.05 seconds	0.86 seconds

You can see that when availability is at 99.95%, we'll have over 4 hours of downtime per year; when availability is 99.99%, the downtime reduces within 1 hour per year, which may be acceptable by most applications. When availability is 99%, which seems like a good number but after calculation, it means over 3 days per year when the application won't run. This situation cannot be accepted by mission-critical applications. When there is a problem, there is a solution, and Microsoft is striking to improve the resilience of applications by using major levels to manage the SLA (service-level agreement) of VMs:

- Single VM (based on premium storage): SLA 99.9%
- Availability Set: SLA 99.95%
- Availability Zone: SLA 99.99%
- Region Pairs

Here is Microsoft's latest infographic regarding different levels of availability provided by Azure services: https://azurecomcdn.azureedge.net/mediahandler/files/resourcefiles/azure-resiliency-infographic/Azure_resiliency_infographic.pdf

There are several general approaches to achieving high availability across regions pairs:

- **Active/passive with hot standby** means that when the traffic goes to the primary region, the VMs in the secondary region are allocated and running at all times.
- **Active/passive with cold standby** means that when the traffic goes to the primary region, but the VMs in the secondary region is not allocated until needed for failover and it will take time to be allocated in case of fail-over.

- **Active/active** means the primary and secondary region are both active, and requests can be distributed by load balancing between them. The healthy status will be determined by healthy

There are two types of events, planned maintenance and unplanned maintenance, in Azure that will affect the availability of Azure virtual machines.

Planned maintenance is when VMs are restarted due to Microsoft updates on the underlying platform. **Unplanned maintenance** is when there is a hardware failure.

To learn more about the SLA of VMs, check out the following link: https://azure.microsoft.com/en-us/support/legal/sla/virtual-machines/v1_8/

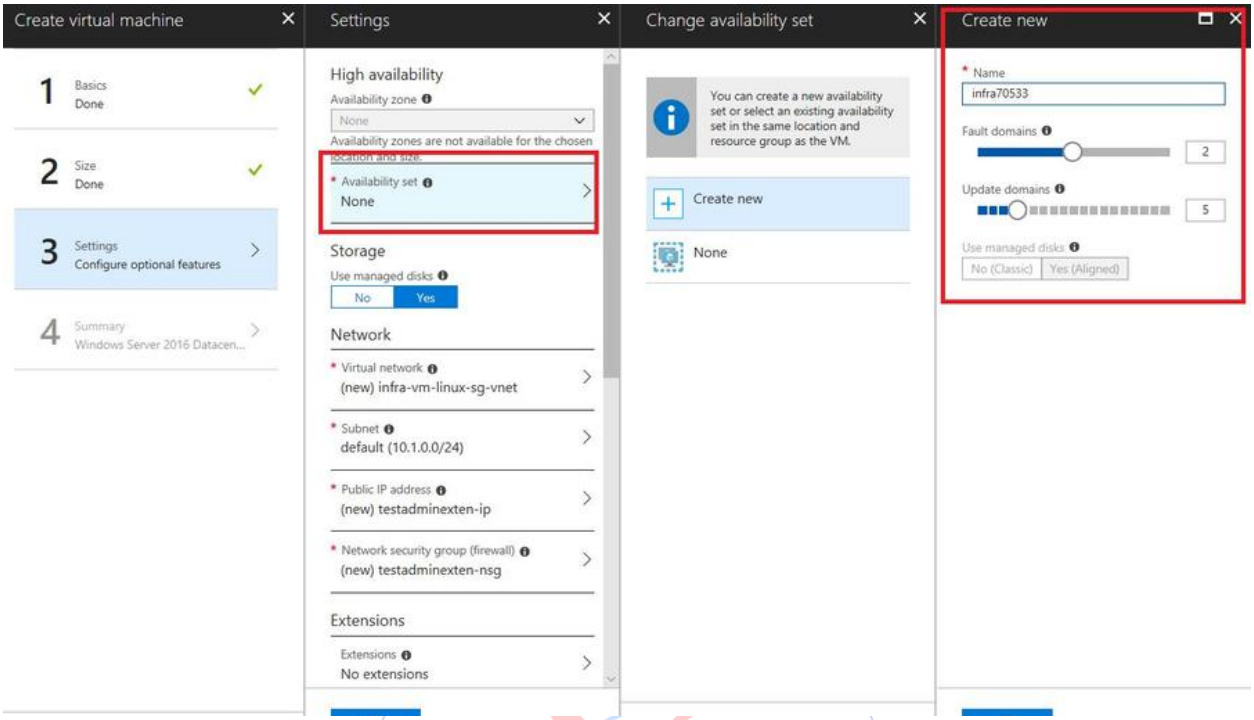
Now, in the subsequent sections, we'll be putting theory into practice.

Improving Azure VM's availability using Availability Sets or Availability Zones

To provide redundancy to your application, Microsoft recommends that you group two or more virtual machines in an Availability Set, which is a logical grouping of two or more virtual machines. This configuration ensures that, during a planned or unplanned maintenance event, at least 2 virtual machines in the Availability Set will meet the 99.95% Azure SLA.

When creating Availability Sets, Microsoft recommends the following best practices:

- For redundancy, configure multiple virtual machines in an Availability Set
- Configure each application tier into separate Availability Sets
- Combine a load balancer with Availability Sets



Another fact is that if you have two or more instances deployed across two or more Availability Zones (AZ is only available in some regions and for some Azure services mentioned in the following link, **https://docs.microsoft.com/en-us/azure/availability-zones/az-overview** at the moment) in the same Azure region, the SLA will be at least 99.99 %. You can choose the Availability Zone while creating a new VM from the drop-down menu.

Looking into the VM's availability by converting a Windows virtual machine from unmanaged disks to managed disks

As we explained previously, since managed disks compared to unmanaged disks break the limits of IOPS per storage account, Microsoft recommends that you convert the VMs to use managed disks through the Azure Managed Disks service. This means that the best practice is to convert both the OS disk and any attached data disks of an Azure VM. This approach provides better availability than unmanaged disks. In Azure, for any single instance of Azure VM using premium storage (SSD) for all Operating System Disks and Data Disks, the will meet SLA at least 99.9%.

Improving Azure VM's availability by combining a load balancer with Availability Sets

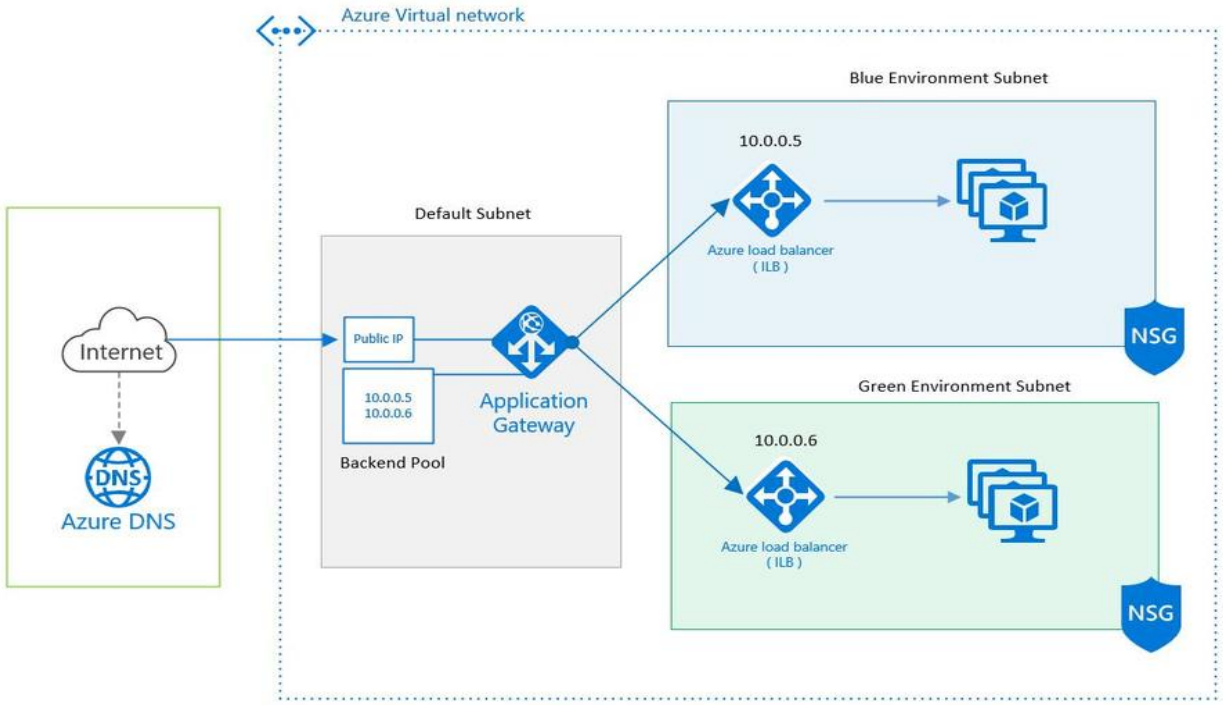
While deploying multiple Azure VMs to improve availability purposes, we usually combine a load balancer that allows distributing traffic between multiple virtual machines. The load

balancer is usually integrated with a health check process, which means it attempts connections or sends requests to test the VMs periodically. It routes requests to the VMs that are available. Microsoft recommends combining the Azure load balancer with an Availability Set to get the most application resiliency.

Improving Azure VM's availability by implementing blue-green deployment in Azure

The main principle of blue/green deployments, which are also known as A/B deployments, is to deploy two identical environments, which are configured in the same way. Generally, while one environment is live and in use by users, the other environment stays idle. When downtime occurs, this architecture allows you to redirect the incoming traffics to the idle configuration, which runs the original version with the help of a load balancer. The aim is to reduce downtime during production deployments.

The general B/G architecture in Azure contains one resource group for the green environment, which usually contains an application in the old version deployed in a VMSS, and one resource group for the blue environment, with an application in the newer version deployed in a VMSS. All the resources are in the same virtual network (VNet); the green and blue environment is on a different subnet. The Application Gateway receives all incoming traffic and distributes it to the backend load balancers. Application Gateway contains two addresses in its backend pool, which are the frontend of two load balancers. Each VMSS has an internal load balancer with a private frontend IP address so that it can distribute the incoming traffic across the backend VMs. The following is an example of designing a B/G deployment in Azure with VMSS; an Azure Load Balancer, which is a Level 4 load balancer; and an Application Gateway, which is a Level 7 load balancer.



Improving Azure VM's availability by implementing multi-region deployments with ARM Templates

Microsoft also recommends managing availability by deploying your infrastructure to a multi-region. In the case of regional outage that affects the primary region, a DNS-level load balancer, which is a Traffic Manager in Azure, provides capabilities to fail over to the secondary region. The multi-region architecture helps if an individual subsystem of the application fails. To learn more regarding the multi-region architecture, refer to the examples in the Azure reference architecture center.

Run Linux VMs in multiple regions for high availability using the following link:

https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/virtual-machines-linux/multi-region-application

Run Windows VMs in multiple regions for high availability using the following link:

https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/virtual-machines-windows/multi-region-application

Implementing and Managing Containers in Azure

As cloud computing technology has become the new norm, it has helped in the quick growth of virtualization technology, which has changed the IT industry in a significant way. Containers and container clusters in the cloud are becoming more widely spoken of today.

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

- Implementing the Azure Container Registry in Azure
- Creating and managing container images with Docker
- Deploying and managing clusters of containers using **Azure Container Service** (**ACS**), along with open source and non-Microsoft container orchestration solutions, such as Docker Swarm, Kubernetes, and DC/OS
- Deploying a Kubernetes cluster with **Azure Kubernetes Service** (**AKS**) in Azure
- Migrating container workloads to, and from, Azure
- Monitoring Kubernetes using Microsoft **Operations Management Suite** (**OMS**)

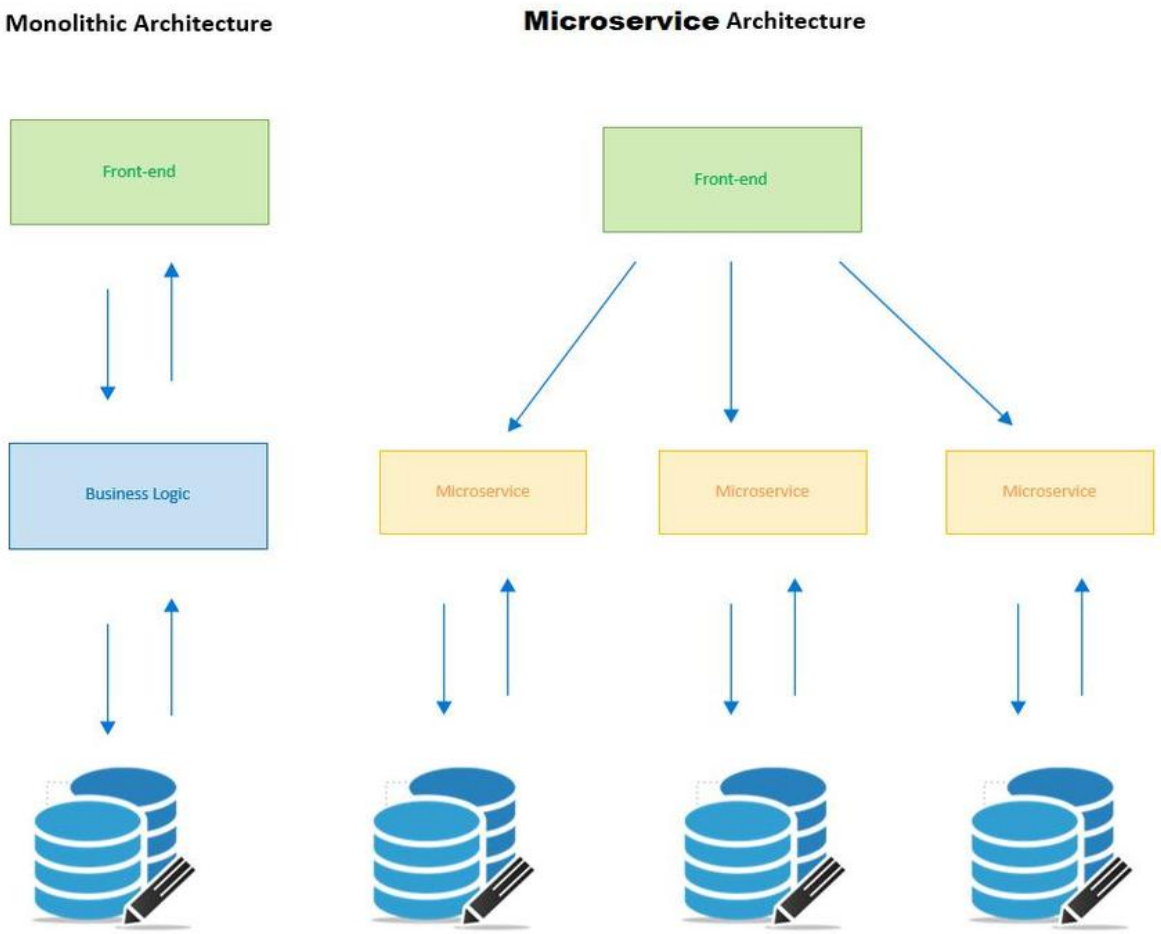
The principle of containers and microservices

Before discussing containers, we cannot ignore the term **microservice**. Microsoft defined microservices architecture as a whole system that contains a collection of small, autonomous services.

The following are the three main reasons for building the modern applications with the microservice architecture:

- With independent, autonomous modules, it is possible make each module scale at their own space
- Using different technologies in the same application has become possible, for example, a RESTful API service can be developed both in Node.js and in the .NET web API in the same application
- High backwards compatibility makes client-side applications evolve at their own pace

The difference between a traditional monolithic architecture, or N-tier applications (such as the front web tier, middleware business logic, and backend data tier), and a microservice architecture can be explained with the following schema:



As we can see in the preceding schema, the microservice has classified an application into individual modules, and each of them has its own data tier that makes sure that a microservice can work as an independent module. Each module can be scaled at their own pace. In case of failure of one microservice, other microservices won't be affected—the system will be still working, all we need to do is to restore the failed modules.

With the help of containers that allow us to run the applications in an isolated virtualized environment, there will be no more challenges such as administering the patch of the operating system, and managing the dependencies of applications. Containers aim to enforce the

portability and agility of applications; that is why it is a powerful technology to build applications in a microservice architecture.

Containers versus container clusters

Containers is definitely a popular keyword in the IT industry, and is a higher level of virtualization technology compared to virtual machines. Containers act as a **recipe** in the kitchen, and they manage to virtualize the operating system and the related infrastructure. We can add applications with anything related to it, such as all dependencies, libraries, and other binaries and configuration files needed to run the application inside, bundled into one package. Definitely, containers enforce the portability of an application, it is a great to resolve the problem of how to run the application while moving from one computing environment to another.

Containers act as a deployment unit when we need to deploy multiple container clusters. It was initially designed for the development and staging environment and will be ready for production environments soon. One of the greatest examples of containerization is Docker.

We'll explain the basics of Docker and provide a demonstrative example while working with Azure in the following section.

Containers provide an effective way to orchestrate your software, operating system, and hardware configurations to provide a typical running environment to make your application run the way you want it to. However, while running an application with numerous instances, such as hundreds, thousands, or even more container clusters, managing all these clusters becomes a really challenging question. Notable examples of container-clustering solutions are Kubernetes, Docker Swarm, and DO/OS.

We'll explain the basics of container-clustering solutions and practice demos while working with Azure in the following section.

Docker basics

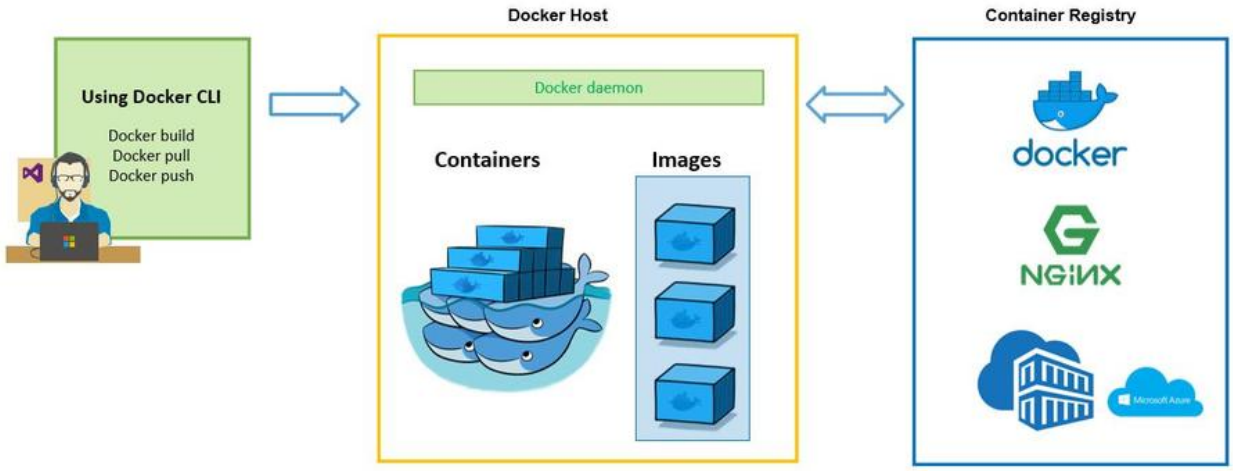
Docker is the world's leading software container platform available for developers, DevOps, and businesses to help them build, deliver, and run any application on independent infrastructures. Docker uses a client-server architecture. Docker can be built into three important parts: **Docker Client**, **Docker Host** (with Docker Daemon), and **Docker Registry**. Each part has its own responsibilities:

- **Docker Client** is where a Docker environment should be installed to build Docker images with the target application
- **Docker Host** is a managed host with Docker Daemon (also known as dockerd, which is the persistent process that manages containers)
- **Docker Registry** provides or stores different Docker images

The Docker Client can communicate with Docker Daemon using the RESTful API over UNIX sockets or a network interface in the following ways:

- Docker Client can run on the same system with Docker Daemon
- Docker Client connects to a remote Docker Daemon

This can be done as shown in the following picture (from **Docker Documentation**):

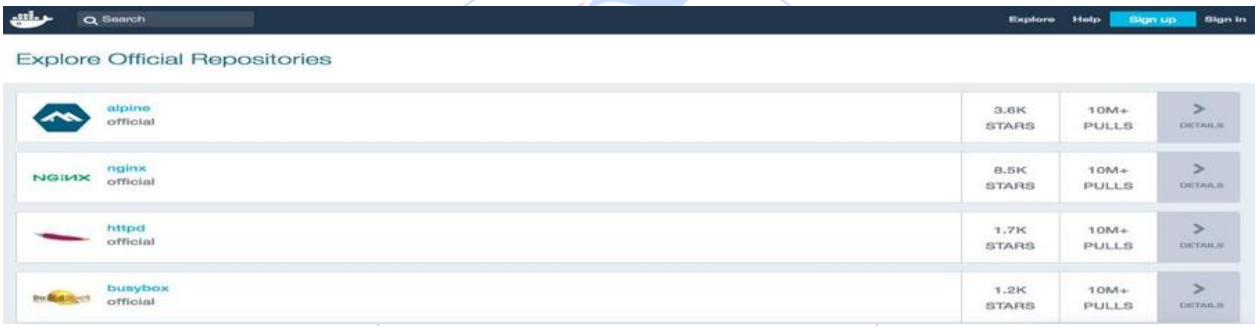


Container registry

The container registry provides different Docker images in the marketplace. We will provide examples of some famous open communities for Docker images in this section.

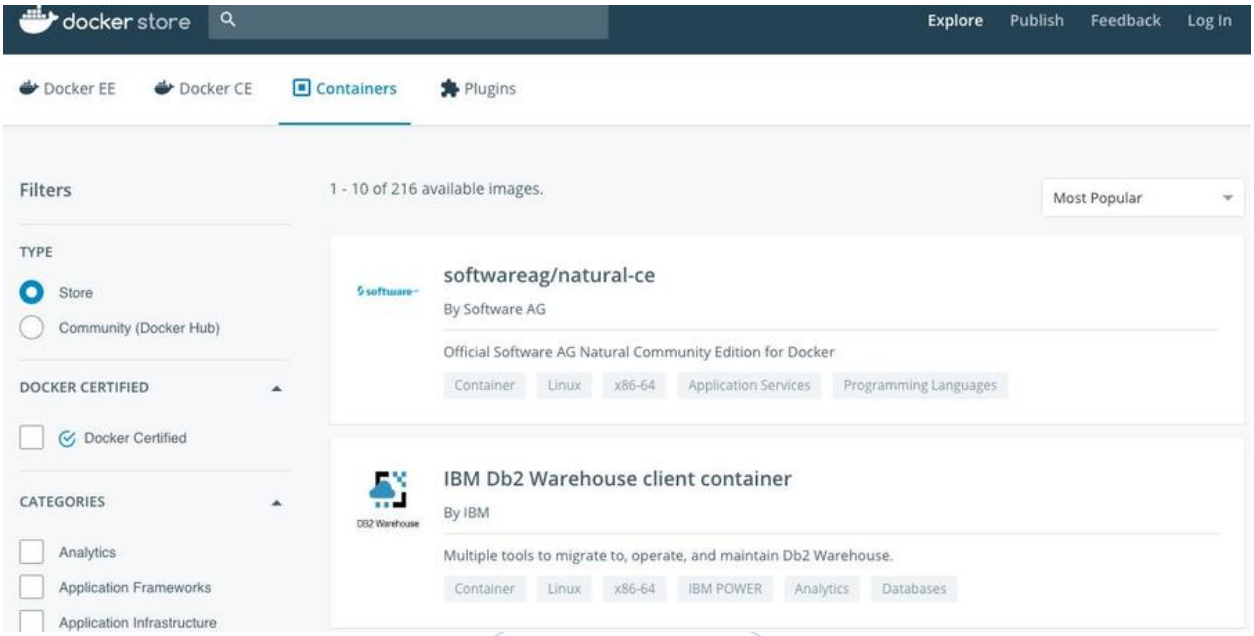
Docker Hub

Docker Hub provides public and private registries for Docker images that are built by other communities. Using Docker Hub, users can also upload their own Docker-built images to Docker Hub. It provides a webhook to support dev-test pipeline automation.Navigate to the official site of Docker Hub at https://hub.Docker.com/ to find what you need:



Docker Store

The Docker Store is a common Docker registry. It provides especially trusted and enterprise-ready containers, plugins, and Docker editions. The official site of Docker Store is https://store.docker.com/, as shown in the following screenshot:



Nginx

Another repository that contains official Docker images for Nginx is on GitHub. You can find some official Nginx Dockerfiles at the following repository: https://github.com/nginxinc/docker-nginx.

Dockerizing your web application in Azure

Microsoft Azure provides a couple of capabilities to help developers and organizations to Dockerize their application.

Preparation work

Before working with Azure, users should make sure that they have installed the Docker environment and Docker CLI tools correctly:

1. Users can download Docker and install it using the following useful links that have guidance for installation:
    - Install Docker for Windows: https://docs.docker.com/docker-for-windows/install/
    - Install Docker for Mac: https://docs.docker.com/docker-for-mac/install/

If you have an older version of Windows, you may need to install Docker Toolbox before installing Docker from https://docs.docker.com/toolbox/overview/.

After installing Docker, a whale will be displayed in the notification area, which shows that Docker has started, and users can access Docker from a terminal or console from your OS. The following is a simple command to verify that Docker is ready in your host.

Run the following command to list all the available Docker commands:

docker --version

Here is a screenshot of the output (in macOS):

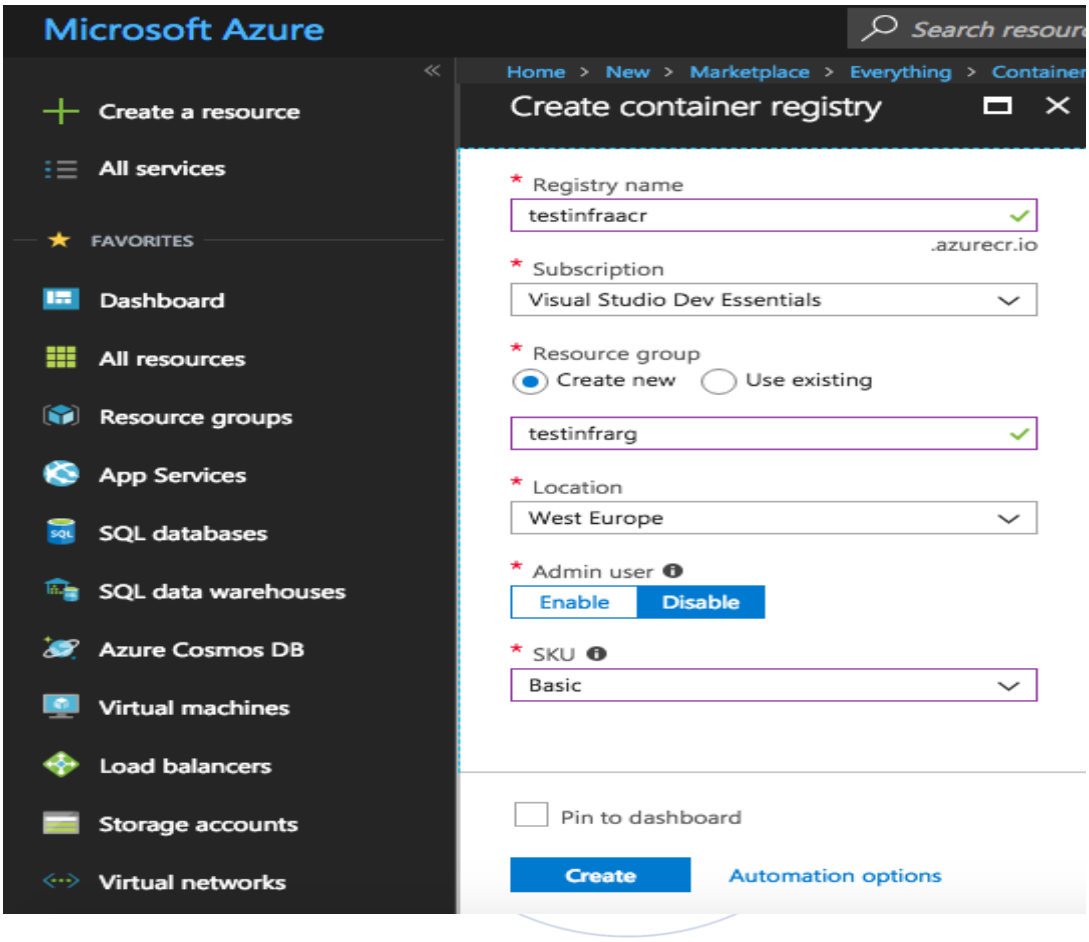
```
MelonyQins-MacBook-Pro:~ melony$ docker -v
Docker version 17.12.0-ce, build c97c6d6
MelonyQins-MacBook-Pro:~ melony$ docker --version
Docker version 17.12.0-ce, build c97c6d6
MelonyQins-MacBook-Pro:~ melony$ docker

Usage:  docker COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default "/Users/melony/.docker")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level ("debug"|"info"|"warn"|"error"|"fatal") (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default "/Users/melony/.docker/ca.pem")
```


Implementing Azure Container Registry

Azure Container Registry is a private registry performed by Microsoft Azure to help developers host Docker-formatted images.

Azure Container Registry integrates well with orchestrators hosted in Azure Container Service, such as Docker Swarm, DC/OS, and Kubernetes. Users can benefit from using open source CLI tools, and this made it possible to maintain Windows and Linux container images in a single Docker Registry.

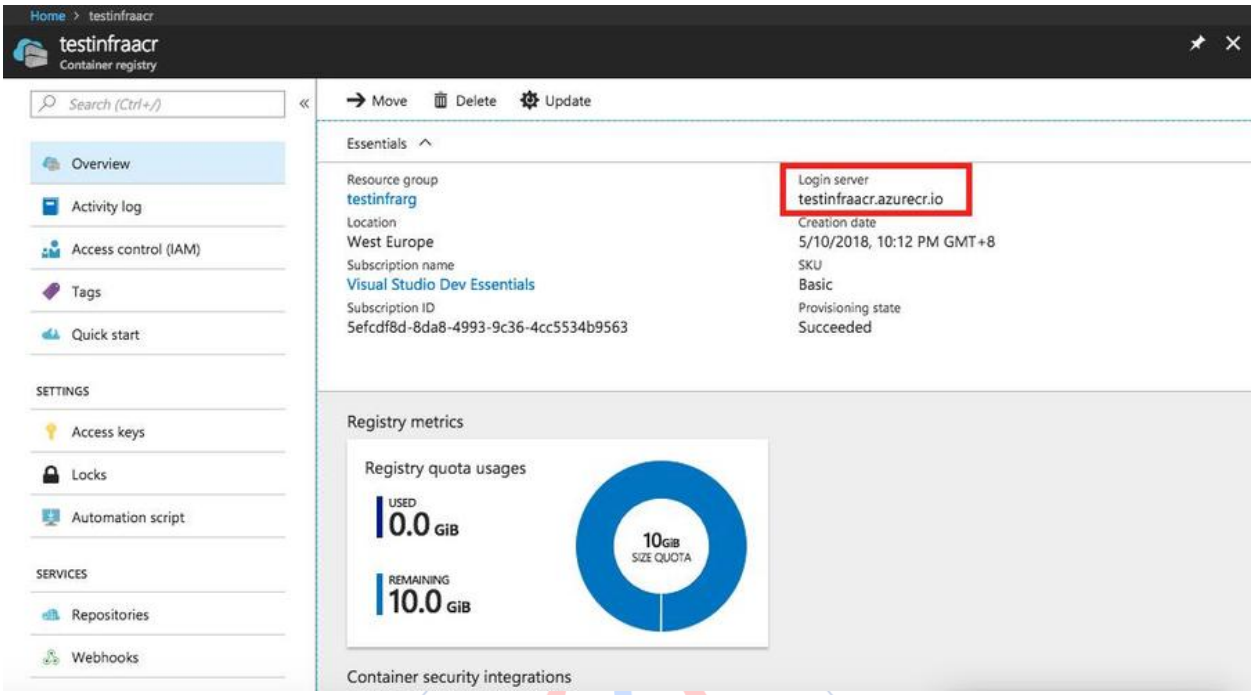
To create an Azure Container Registry, go to Azure portal, click on **Create a resource**, then search for Container Registry and choose it. Then, fill in the necessary information in the blade:



Creating the Container Registry

There are currently three types of capabilities that is, Basic, Standard, and Premium. All SKUs provide the same programmatic capabilities, but a higher SKU will provide more available storage, total webhooks, and a geo-replication feature (which is only available for Premium SKU), for example.

After creating ACR successfully via Azure Portal, you can return to the resource group that you've created and search for the created container registry, as follows:



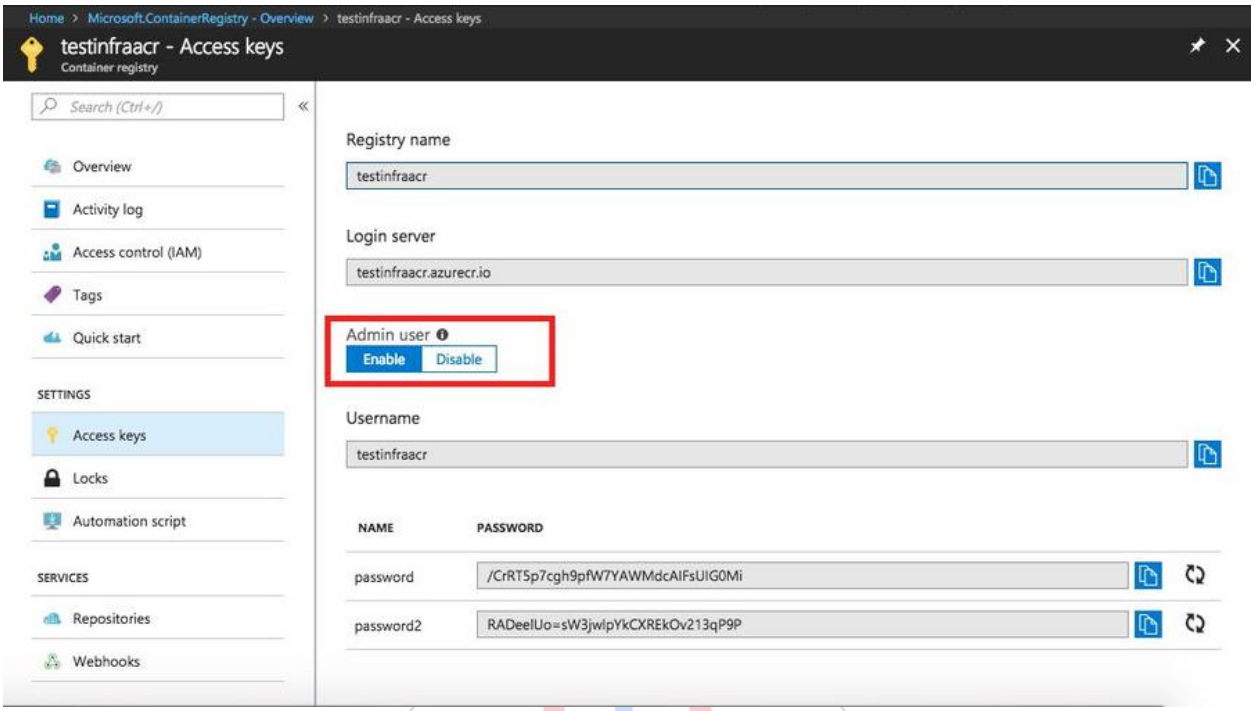
Creating the Azure Container Registry

The following is the URL of the login server that you'll use to log in:

```
#nameofregistry#.azurecr.io
```

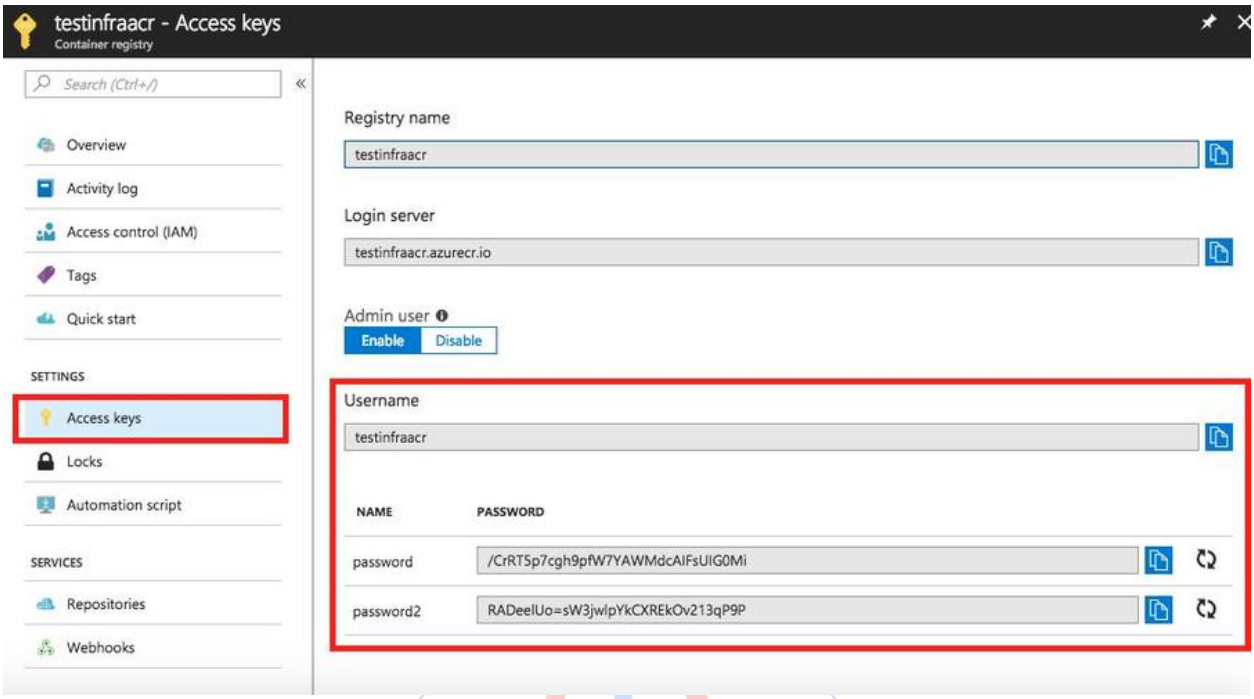
Pushing your Docker image in ACR

If you haven't enabled the **Admin user** while creating an ACR, you can navigate to the **Access keys** blade and click on **Enable** (as shown in the following screenshot):



Enabling the admin user in the Access Key blade

Then, you can use your username and password on the **Admin user** section to log in to your ACR registry and manage your credentials here (as shown in the following screenshot). If you lose your password, you can regenerate it by clicking on the refresh icon, and your current password will immediately become invalid and not recoverable:



Before pushing your local Docker image to ACR, use the following command to log in to Docker (replace the words between # with your own):

docker login myContainerRegistry.azurecr.io -u #username# -p #password#

If you log in to Azure container registry, you'll get the following output:

```
melonyqins-macbook-pro:Github melony$ docker login testinfraacr.azurecr.io -u testinfraacr -p /CrRT5p7cgh9pfW7YAW
dcAIFsUIG0Mi
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Warning: failed to get default registry endpoint from daemon (Cannot connect to the Docker daemon at unix:///var/
un/docker.sock. Is the docker daemon running?). Using system default: https://index.docker.io/v1/
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?
melonyqins-macbook-pro:Github melony$ docker login testinfraacr.azurecr.io -u testinfraacr -p /CrRT5p7cgh9pfW7YAW
dcAIFsUIG0Mi
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded
```

Tag the locally built image to the ACR repository:

docker tag #yourImageId
yourContainerRegistry.azurecr.io/starterapp:latest

Then, push it to ACR using the following commands:

```
docker push yourContainerRegistry.azurecr.io/starterapp:latest
```

If you pushed your Docker images to ACR successfully, you'll get a message stating that it is Pushed in the output, as shown in the following example:

```
melonyqins-macbook-pro:Github melony$ docker push testinfraacr.azurecr.io/starterapp:latest
The push refers to repository [testinfraacr.azurecr.io/starterapp]
f999ae22f308: Pushed
latest: digest: sha256:8072a54ebb3bc136150e2f2860f00a7bf45f13eeb917cca2430fcd0054c8e51b size: 524
```

If you still want to verify that your images are in the registry, you can also use the following commands to verify it in the cloud shell or your local machine if you've already installed Azure CLI:

```
az acr repository list -n yourContainerRegistry
```


Deploying your Dockerized application with CI/CD capabilities

Currently, Microsoft Azure provides two ways to deploy the Dockerized application:

- Web App for Containers
- Azure Containers instances

Clustering solutions with Azure ACS in Azure

Nowadays, there are a couple of container orchestrators that help us to simplify the management of container clusters, in order to improve an application's scalability and resilience. The common objective of these tools is to let users handle the entire cluster as a

single deployment, which also extends the life cycle management capabilities to complex workloads with multiple containers deployed on a cluster of machines.

The open source container orchestrators are popular in the market, such as Docker Swarm, Kubernetes, and Mesosphere's DC/OS.

As a fast-growing cloud provider, Microsoft Azure implemented a container service known as ACS for all of these popular container orchestrators in Azure. This makes the containerization application fully portable by leveraging Docker images. These applications can be scaled to thousands or even more containers.

To implement any container cluster, Microsoft Azure provides a cloud-based service, which is known as Azure Container Service.

An overview of container cluster solutions working with ACSs

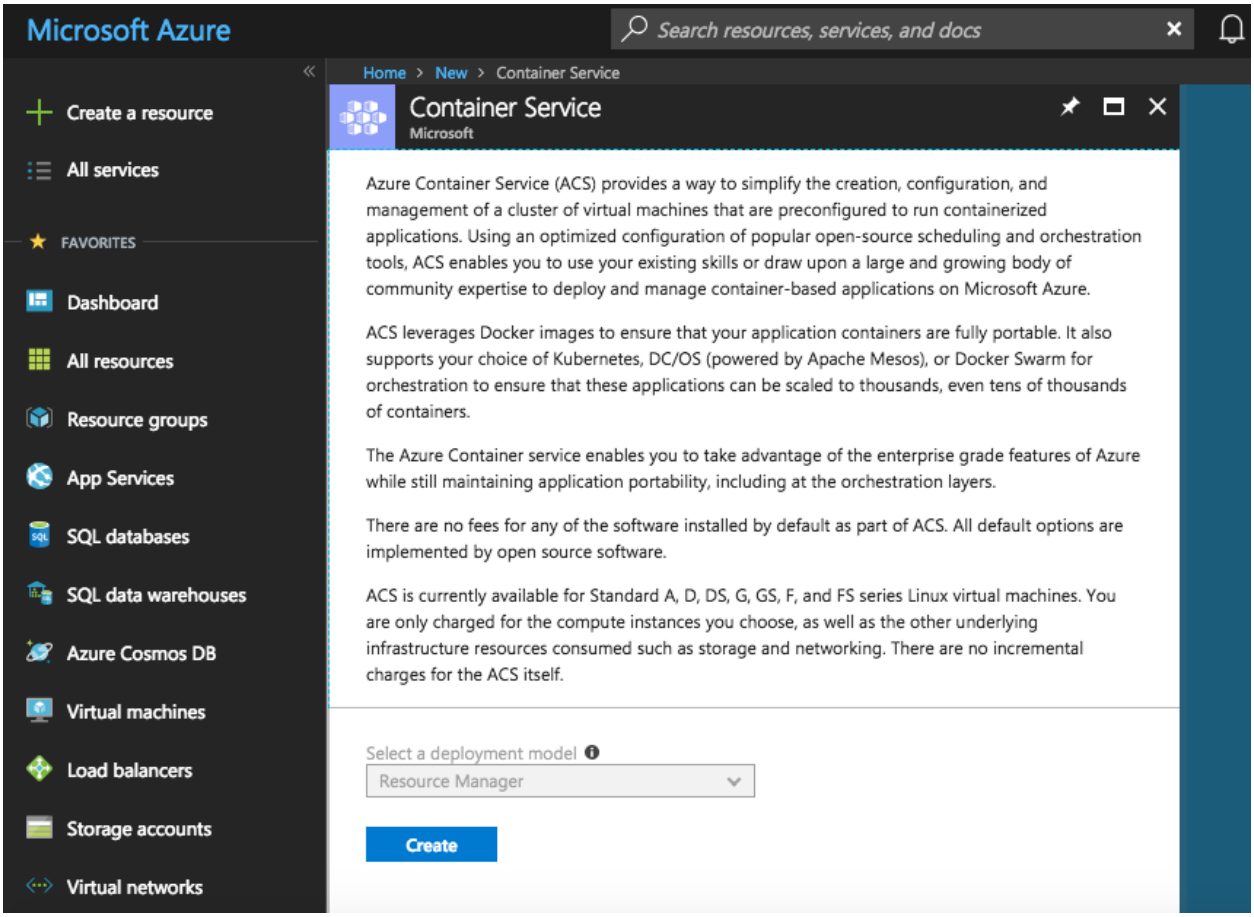
The goal of ACS is to simplify the creation, configuration, and management of a container cluster in the Azure cloud using an optimized configuration of popular open source scheduling and orchestration tools. ACS implements three kinds of popular open source orchestrators, such as Kubernetes, DC/OS (datacenter and operating system, which is powered by Apache Mesos), and Docker Swarm. You can use them for orchestration in Azure.

When you're using ACS, Microsoft Azure only charges for the compute instances and the underlying infrastructure resources consumed, such as storage or networking. There are no fees for any of the software installed by default as part of ACS.

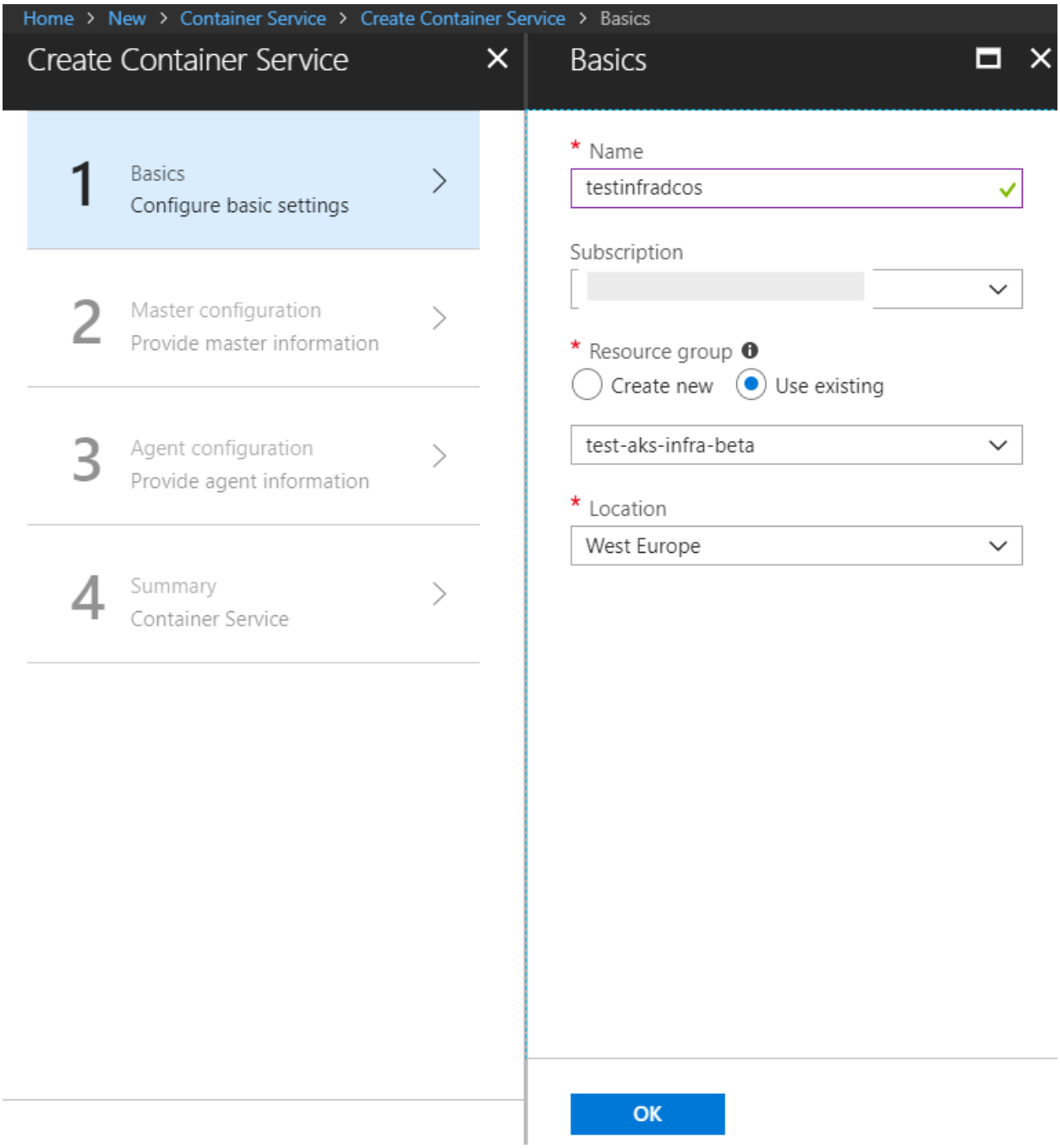
We can directly deploy an ACS cluster via the portal, use the Azure CLI, or deploy an ARM (Azure Resource Manager) template.

Creating ACS via the Azure Portal

In the Azure Portal, click on **Create a resource**. You can search for container services and then choose it to start to create a container service in Azure, as shown in the following screenshot:



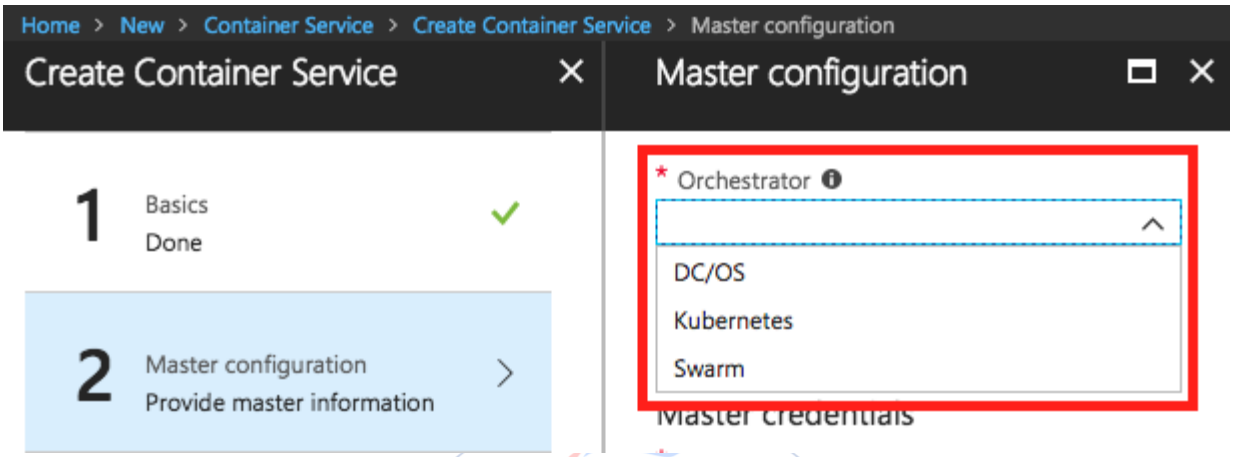
In the **Basic** blade, you can specify the name of the cluster and subscription that you want to use for this resource and define the right resource group and resource location. Then, go to **Master configuration**, which is important to identify the type of your orchestrator:



Choosing your target orchestrator

To choose **Master configuration**, you can select the type of container cluster that you want to deploy in ACS. You can choose from among the following three types of orchestrator supported by Microsoft Azure: Kubernetes, Docker Swarm, and DC/OS (datacenter operating system). As they have different architectures, your choice will change the type of credentials

you need in the **Master configuration**. All the orchestrators will need a RSA key, and you may also need a service principal if you are going to deploy a Kubernetes cluster. We'll explain this in the next section:



Generating your keygen

In **Master configuration**, under the section **Master credentials**, you have several ways to generate your keygen. Take a look at a way to generate keygen using the following command via cloud shell:

```
ssh-keygen -t rsa -b 2048
```

The preceding command will generate a public key and a private key with the name that you specified while executing the commands. The following is a sample output to inform the system that you have created your key pairsuccessfully:

```
melony@Azure:~$ ssh-keygen -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/home/melony/.ssh/id_rsa): testkeygenk8s
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in testkeygenk8s.
Your public key has been saved in testkeygenk8s.pub.
The key fingerprint is:
SHA256:B6Iu3YjKpLWTOstbBqrGnjeKSy4RQxC0N2Cow0GKZi8 melony@cc-cdaace85-2228111606-9zfhv
The key's randomart image is:
+---[RSA 2048]----+
|B*               |
|=oo              |
|*+.o  . .        |
|B.o .. . .       |
| E ..   S .      |
|o o+ o    .      |
|o++++ .          |
|%=*=             |
|%&=..            |
+----[SHA256]-----+
melony@Azure:~$
```

The cloud shell uses an Azure file storage to persist files across sessions, which was specified when you were starting it the first time. You can use Bash commands, such as ls, to display your files and folders in the current repository as follows:

```
melony@Azure:~$ ls
clouddrive  testkeygenk8s  testkeygenk8s.pub
```

Use the cat command if you want to show the content of your private key:

Check public key

Use the cat command if you want to show the content of your public key

Implementing three types of orchestrators of Azure ACS in Azure

There are three container clusters solutions supported by ACS. Now, let's get an overview on each of them.

Docker Swarm

Swarm is the native clustering for Docker, which also means you can use Docker in swarm mode, which is a way to manage a cluster of Docker Engines. Based on the **docker-native** principle, any tools or containers that work with Docker run equally well in Docker Swarm. Originally, Docker Swarm performed a resilient zero single-point-of-failure architecture, secured by default with automatically generated certificates and backward compatibility with existing components.

As an excellent orchestrator, Docker Swarm can be installed and configured in an easy way. As a **docker-native** orchestrator, Docker Swarm can deploy container clusters faster than Kubernetes or other orchestrators, especially in very large clusters or contexts, which requires fast reaction times to scaling on demand. However, for every plus, there is a minus. As it is naturally designed to extend Docker support, the functionalities are limited by the Docker API

that works with the core Docker Engine. That is why it can't support specific complex operations that aren't supported by Docker.

Docker Swarm architecture

There are two types of nodes in Docker Swarm: the manager node and worker node. The concept of a node is an instance of the Docker Engine in the swarm. It is possible to run one or more nodes on a single physical computer or virtual machine in the cloud. It is also possible to run distributed nodes across multiple physical machines and VMs in the cloud.

The two nodes have the following different roles in swarm mode:

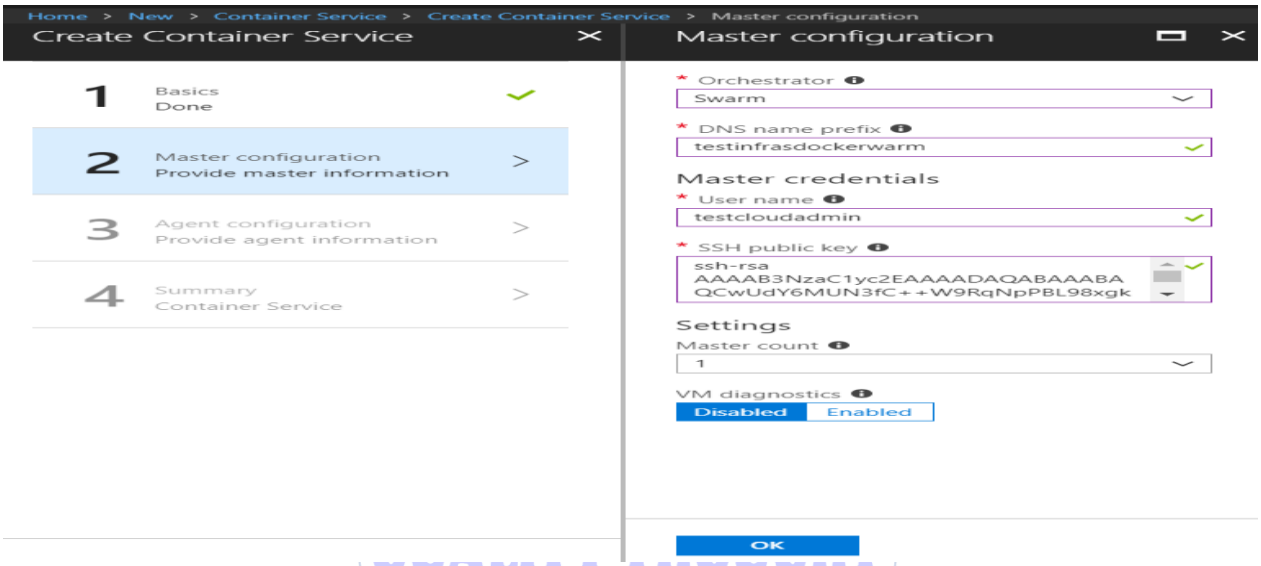
- **The manager** node is in charge of dispatching tasks to the worker node. A task is a unit of work that carries a Docker container and the commands to run inside the container.
- **The worker** node runs swarm services and receives and execute tasks dispatched from manager nodes.

A swarm consists of multiple Docker Hosts, which run in swarm mode and act as managers and workers. A Docker Host can perform as a manager, worker, or both. If a worker node becomes unavailable, Docker is in charge of scheduling that node's tasks to other nodes. This architecture also ensures availability while deploying an application with Docker Swarm. The following diagram shows the **Docker Swarm** architecture:

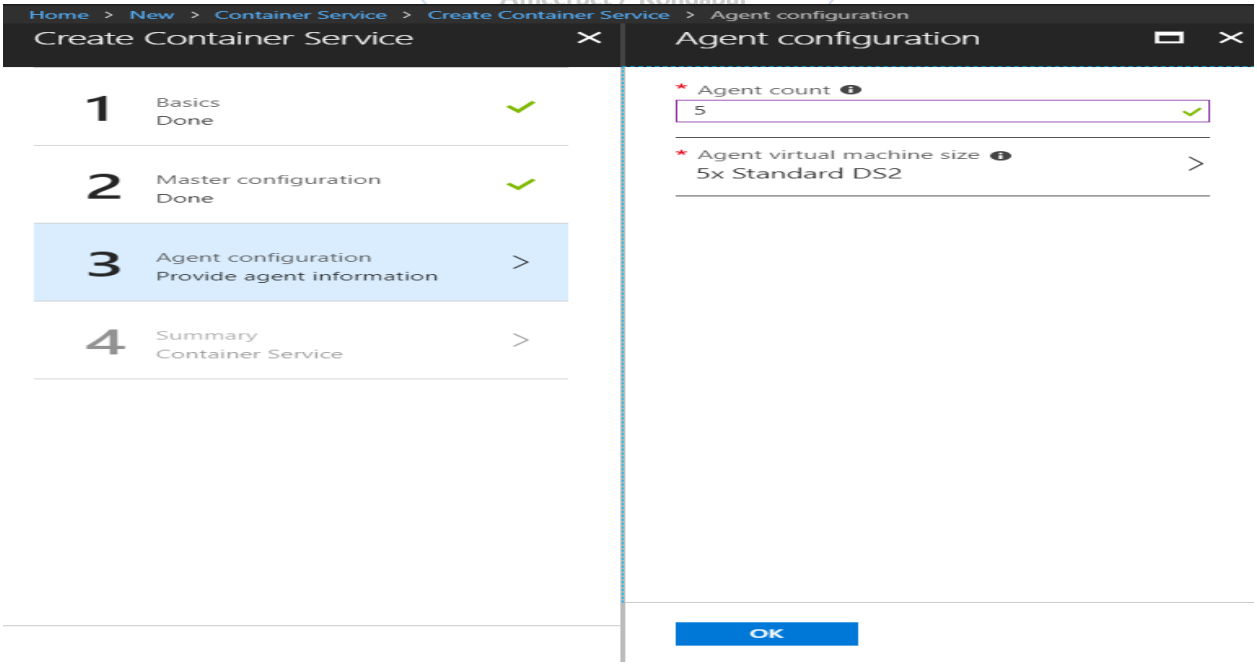
In action – implementing and managing an ACS Docker Swarm cluster

If you're creating an ACS Docker Swarm cluster via the Azure Portal, follow these steps:

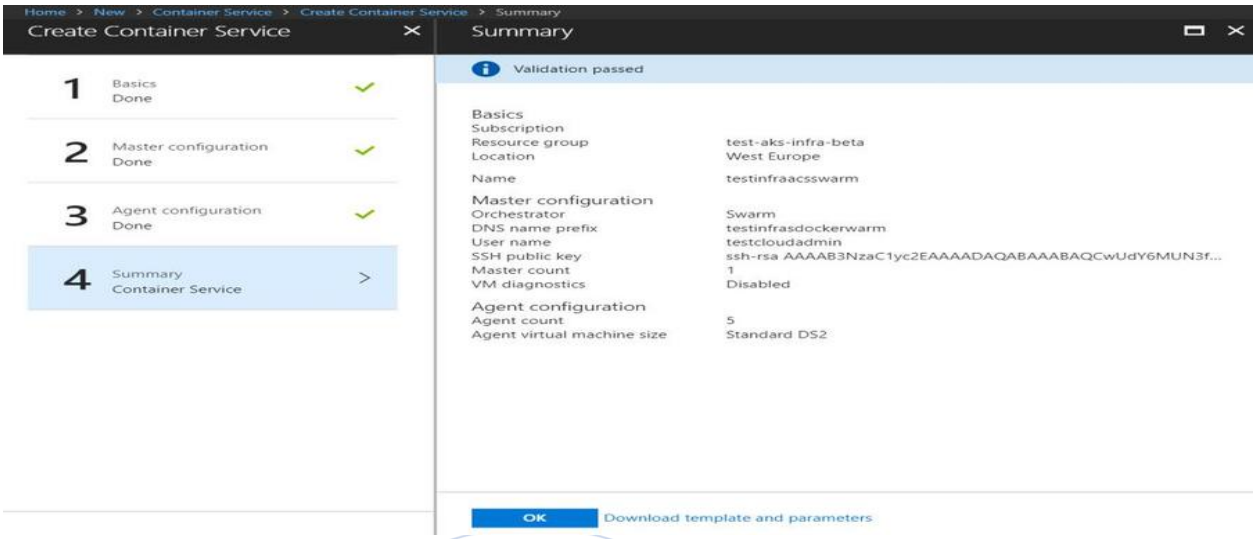
1. Start by creating a **Container Service**, then choose your **Orchestrator** as Swarm in the **Master configuration** step.
2. In the **Master configuration** page, you can specify the **VM** size and the number of nodes you need as the master node:



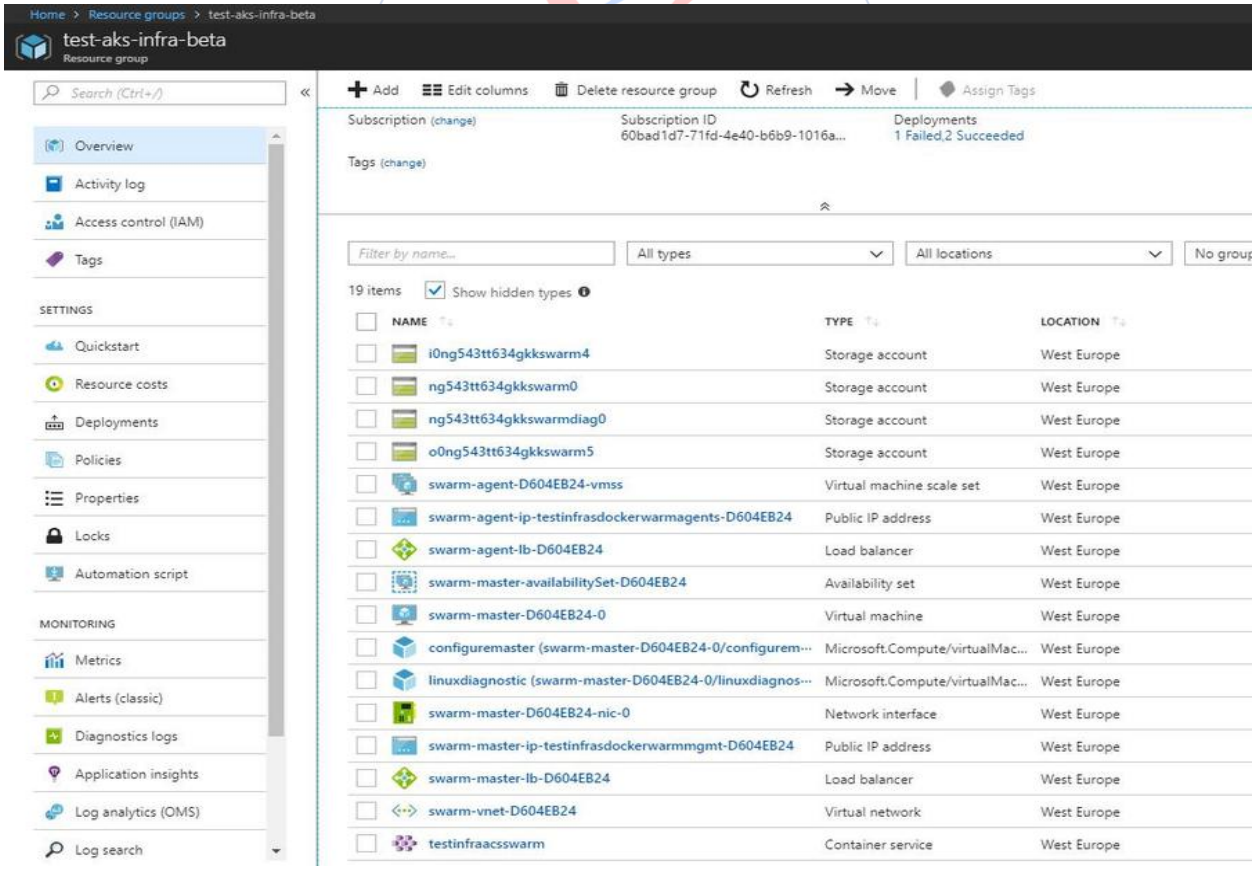
3. Similarly, you can also specify the **Agent count** and the VM size in the **Agent configuration** page, as shown in the following screenshot:



4. The following is a **Summary** page that is shown after all the information is filled in:



5. The deployment usually takes several minutes. After creating an ACS Docker Swarm cluster successfully, go to the resource group; you can note the deployed resource, as follows:



6. To create an ACS Docker Swarm, you can use the following command:

```
az acs create --name #yourSwarmCluster# --orchestrator-type dockerce --resource-group #yourResourceGroup# --generate-ssh-keys
```

Kubernetes

Kubernetes is an open source platform for container deployment automation, scaling, and operations across clusters of hosts. It aims to provide the components and tools to relieve the burden of running applications in public and private clouds by grouping containers into logical units. The advantage of Kubernetes is flexibility, environment agnostic portability, and easy scaling.

Kubernetes requires a configuration file to configure such as etcd, flannel, the **Docker Engine**, **the cluster configuration** such as the IP addresses of the nodes, which determines which role each node is going to take, and how many nodes there are in total before starting the deployment.

The Kubernetes architecture

Traditionally, the Kubernetes architecture is as follows:

The orchestrators are designed to track and monitor the health of the containers and hosts. In the event of a node failure, orchestrators launch a replacement. We call the mechanism to detect whether the application is operating correctly a **health check**.

Kubernetes supports a user-implemented application health check, which is performed by using kubelet**.**

The following two ways are used to implement Kubernetes in Azure:

- Deploying the Kubernetes cluster using ACS.
- Deploying the Kubernetes cluster using Azure Kubernetes Service. This service was in preview at the time of writing this book.

In action – implementing an ACS Kubernetes cluster

To create an ACS Kubernetes, start by creating a resource group using the following command:

```
az group create --name test-infra70533 --location westeurope
```

Creating a new AKS Kubernetes cluster

To create an AKS Kubernetes cluster, use the following commands:

```
az acs create --orchestrator-type kubernetes --resource-group test-infra70533 --name infrak8scluster --generate-ssh-keys
```

Alternatively, you can use the following command:

```
az aks create -g test-infra70533 -n infrak8scluster --generate-ssh-keys
```

The `az acs create` command is used to create a Kubernetes cluster in ACS in the resource group `test-infra70533`. The `--generate-ssh-keys`parameter is used to generate new SSH keys. If you want to use your own SSH keys, you can replace the preceding command with the following command:

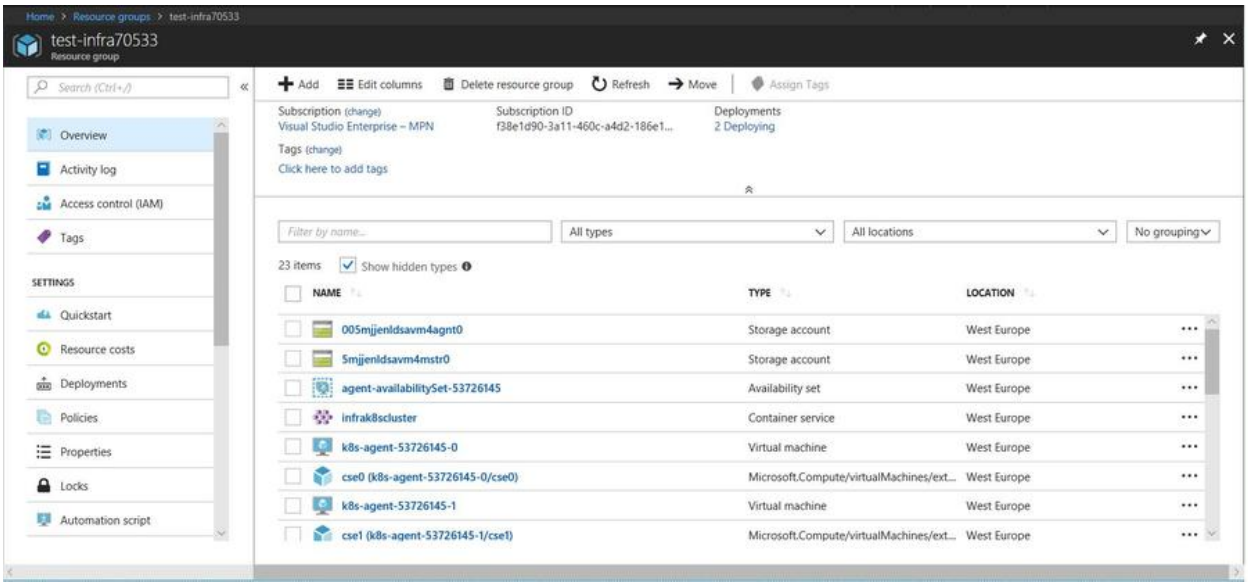
```
az aks create -g test-infra70533 -n infrak8scluster --ssh-key-value /path/to/publickey
```

After several minutes, the command completes and returns **JSON formatted** information about the cluster, which means our Kubernetes cluster has been created successfully. The following is an output information example:

```
{
 "id":              "/subscriptions/f38e1d90-3a11-460c-a4d2-186e1660d993/resourceGroups/test-infra70533/providers/Microsoft.Resources/deployments/azurecli1526992751.50577269674",
 "name": "azurecli1526992751.50577269674",
 "properties": {
  "additionalProperties": {
   "duration": "PT13M12.0422159S",
   "outputResources": [
    {
     "id":        "/subscriptions/f38e1d90-3a11-460c-a4d2-186e1660d993/resourceGroups/test-infra70533/providers/Microsoft.ContainerService/containerServices/infrak8scluster",
     "resourceGroup": "test-infra70533"
    }
   ],
```

```
    "templateHash": "15580770358025216932"
  },
  "correlationId": "96ba0389-9d85-4685-b95f-c6b7bb216af8",
  "debugSetting": null,
  "dependencies": [],
  "mode": "Incremental",
  "outputs": {
    "masterFQDN": {
      "type": "String",
"value": "infrak8scl-test-infra70533-f38e1dmgmt.westeurope.cloudapp.azure.com"
    },
    "sshMaster0": {
      "type": "String",
  "value": "ssh azureuser@infrak8scl-test-infra70533-
f38e1dmgmt.westeurope.cloudapp.azure.com -A -p 22"
    }
  },
  "parameters": {
    "clientSecret": {
      "type": "SecureString"
    }
  },
```


When something similar to the preceding output is returned by Cloud Shell, you can go to the resource group to check whether the resources-related AKS Kubernetes are available, as shown in the following screenshot:



In action – managing an ACS Kubernetes cluster

To connect to the Kubernetes cluster, use the kubectl command, which is Kubernetes, command-line client.

Connecting to a Kubernetes cluster

To configure kubectl to connect to your Kubernetes cluster, you should run the **az acs kubernetes get-credentials** command:

```
az acs kubernetes get-credentials --resource-group=test-infra70533 --name=infrak8scluster
```

The preceding step allows you to download credentials and configures the Kubernetes CLI to use them. Then, you can use the kubectl get command (as shown in the following screenshot) to return a list of the cluster nodes so that you can verify the connection to your cluster:

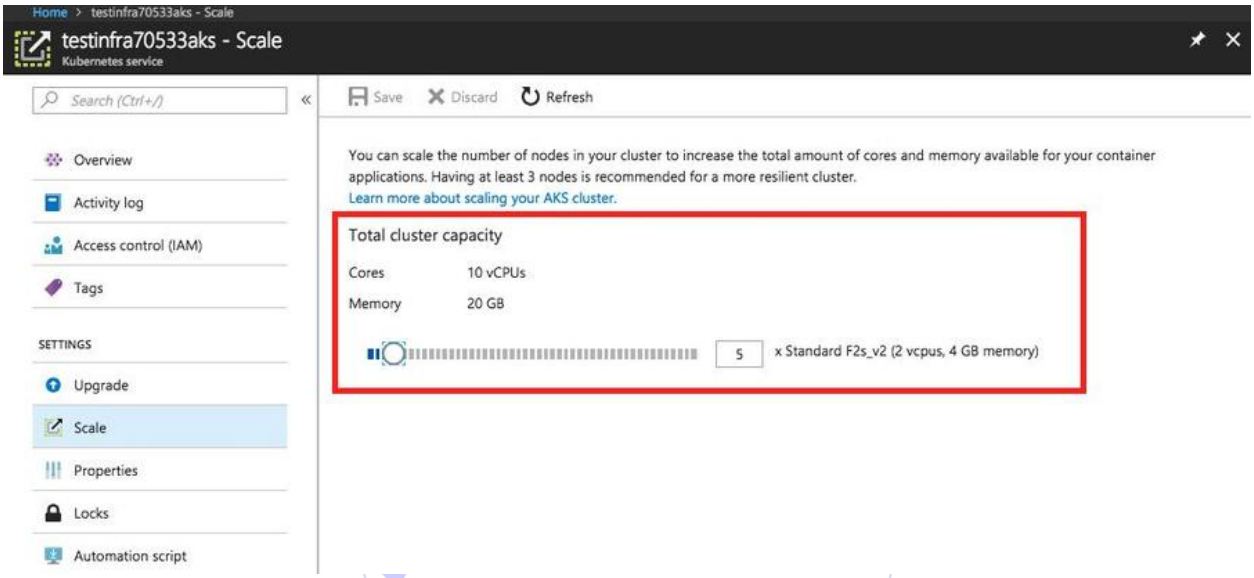
```
kubectl get nodes
```

Scaling the cluster nodes

The az aks scale command is used to scale the cluster nodes, as follows:

az aks scale --name#yourAKSCluster# --resource-group #yourResourceGroup# --node-count 1

Note that the number of the agent node of Kubernetes in the Azure Portal increased from three to five nodes, as shown in the following screenshot:

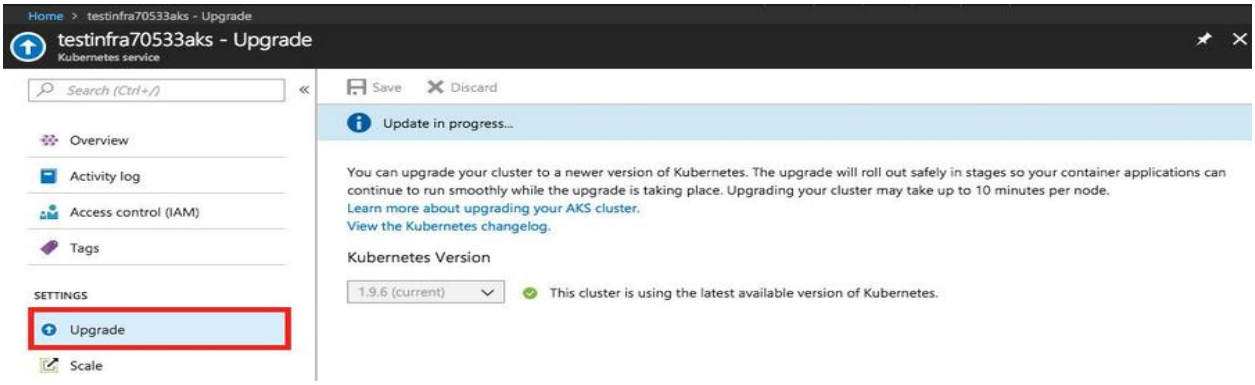


Upgrading an AKS cluster

The az aks get-upgrades command is used to scale the cluster nodes, as follows:

az aks get-upgrades --name #yourAKSCluster#--resource-group #yourResourceGroup# --output table

Another way to upgrade your Kubernetes cluster is via Azure Portal:



Deploying applications to a Kubernetes cluster

You can run your application using the kubectl create command, as follows:

```
kubectl create -f #yourappliactionmanifestfile#.yml
```

Deleting Kubernetes clusters

When the cluster is no longer needed, you can use the az group delete command to remove the resource group, container service, and all related resources:

```
az group delete --name test-infra70533 --yes --no-wait
```

Implementing and managing a Kubernetes cluster with AKS

Another way to deploy a Kubernetes cluster is to create a Kubernetes cluster using AKS. ACS implemented the popular container orchestrator-managed Kubernetes in Azure. It simplifies how to create, configure, and manage a container cluster and many useful features, such as the following:

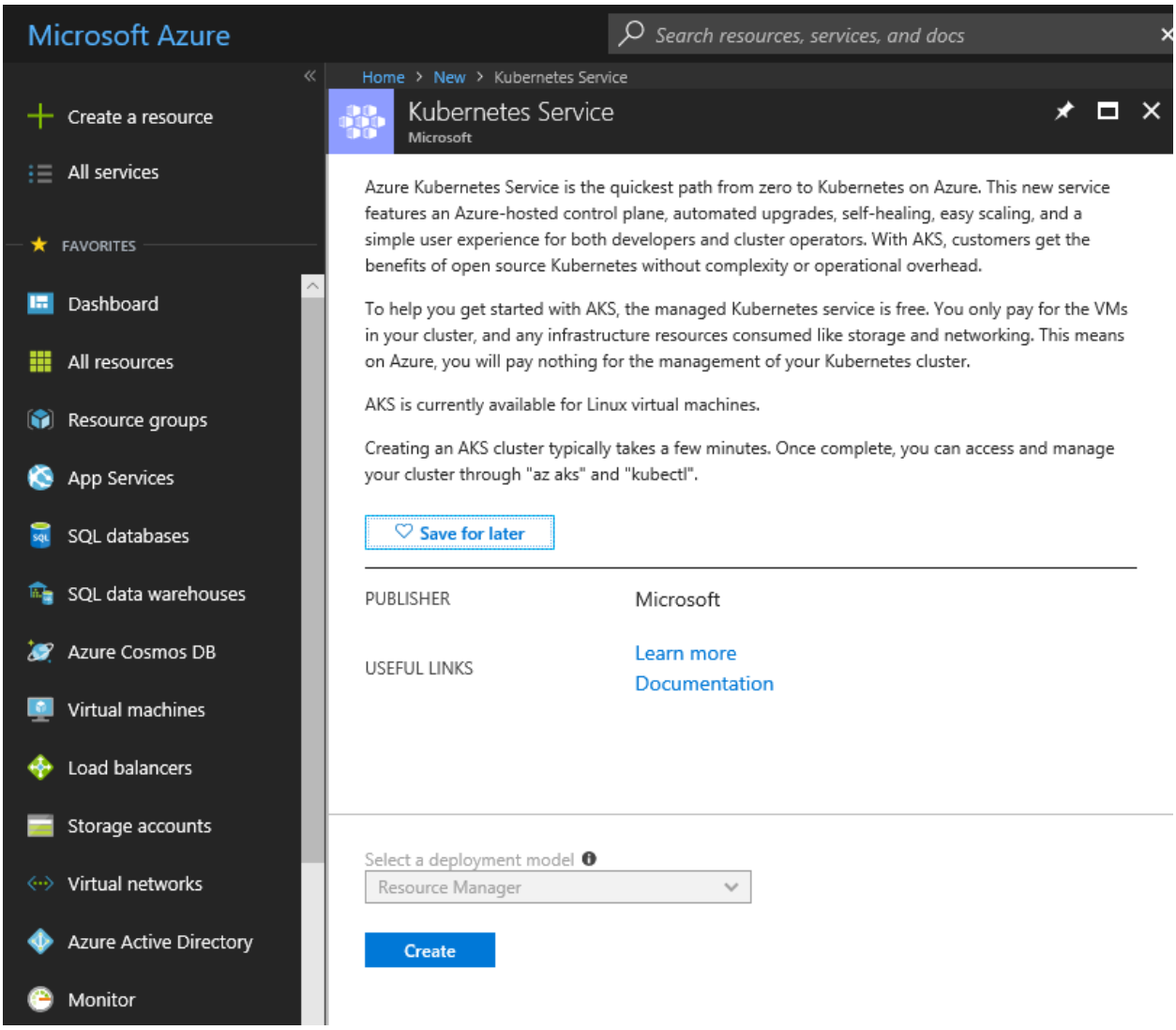
- Easy management of containers even when there are more than 100 instances
- Easy scaling
- Supports popular operating systems, such as Linux and Windows
- Easy rollout and rollback
- Can be combined with batch processes or cron jobs
- Automatic bin packing (depends on GPU / CPU usage, for example)

Using AKS, you can maintain application portability through Kubernetes and the Docker image format, and focus on building a containerized application. Azure will handle the rest of your work, such as container deployment, cluster configuration, and health monitoring.

Regarding pricing in AKS, users pay only for the agent nodes within the clusters, but not for masters, which is in charge of controlling tasks.

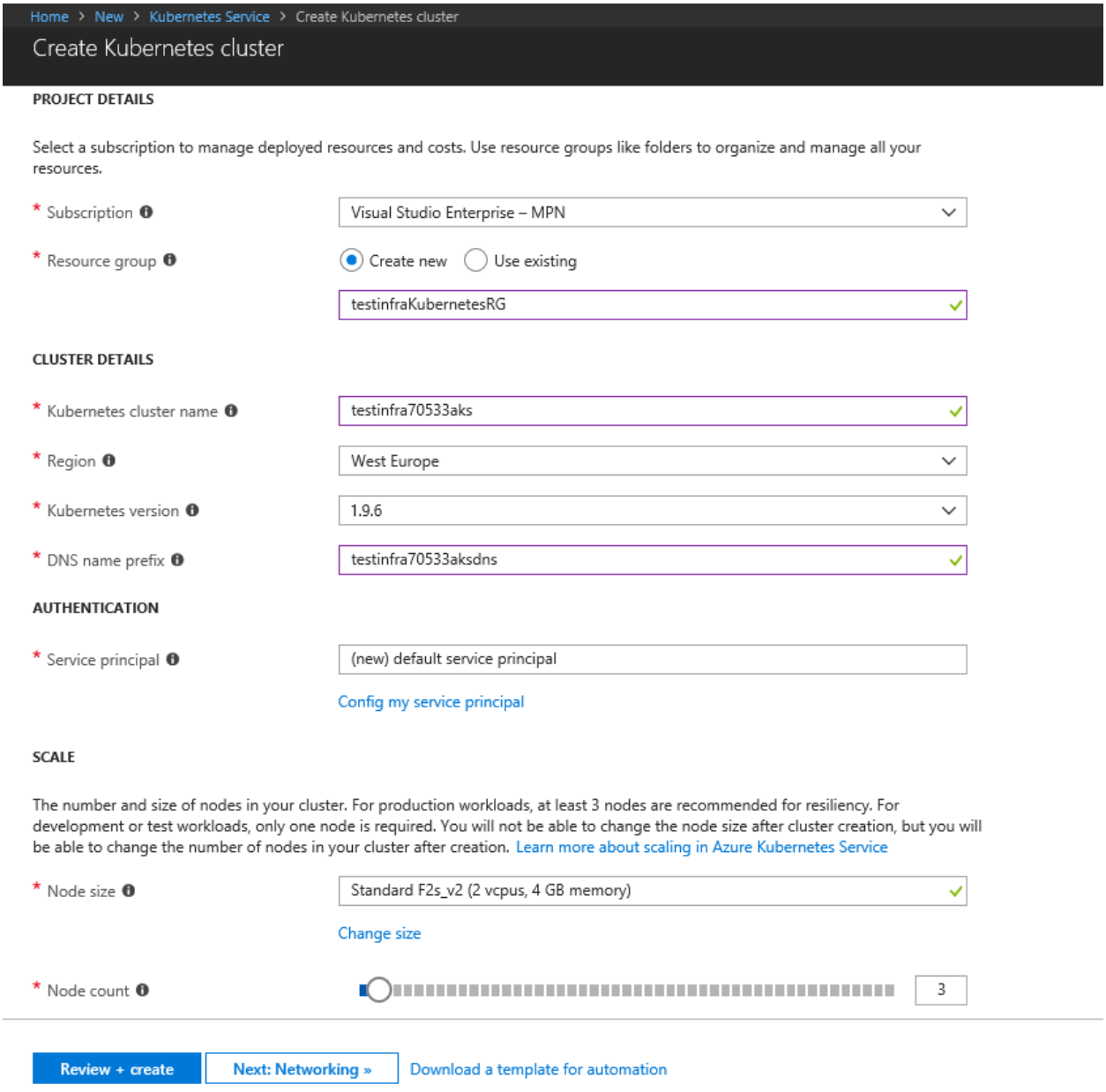
Creating AKS via the Azure Portal

From Azure Portal, click on **Create a resource** and search for kubernetes services. Choose it to start creating an AKS service in Azure, as follows:

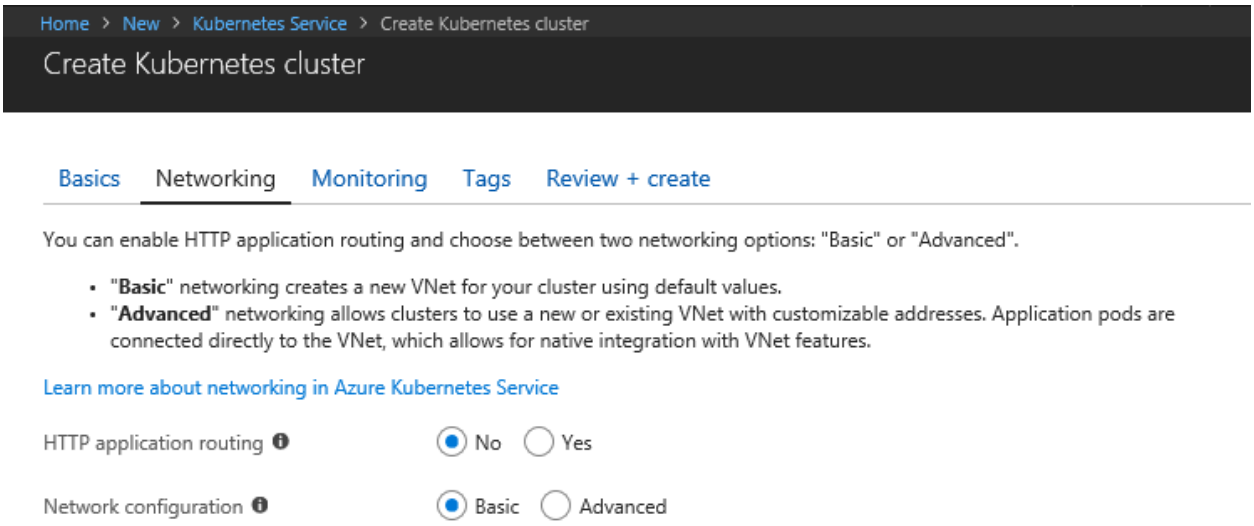


From the **Basics** blade, you can choose the version of Kubernetes, the size of machine you'll deploy with Kubernetes, and the number of nodes. Microsoft recommends that you deploy at least three nodes to improve the resilience of your application in production. You can deploy

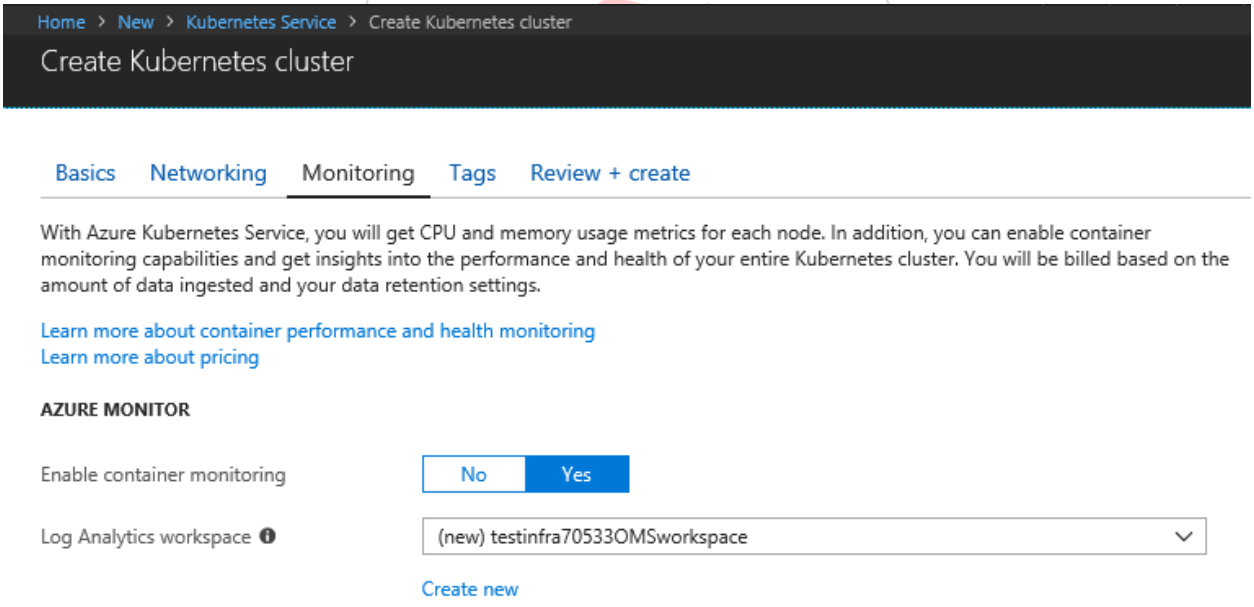
only one node for test purposes or in the development environments as described in the following screenshot:



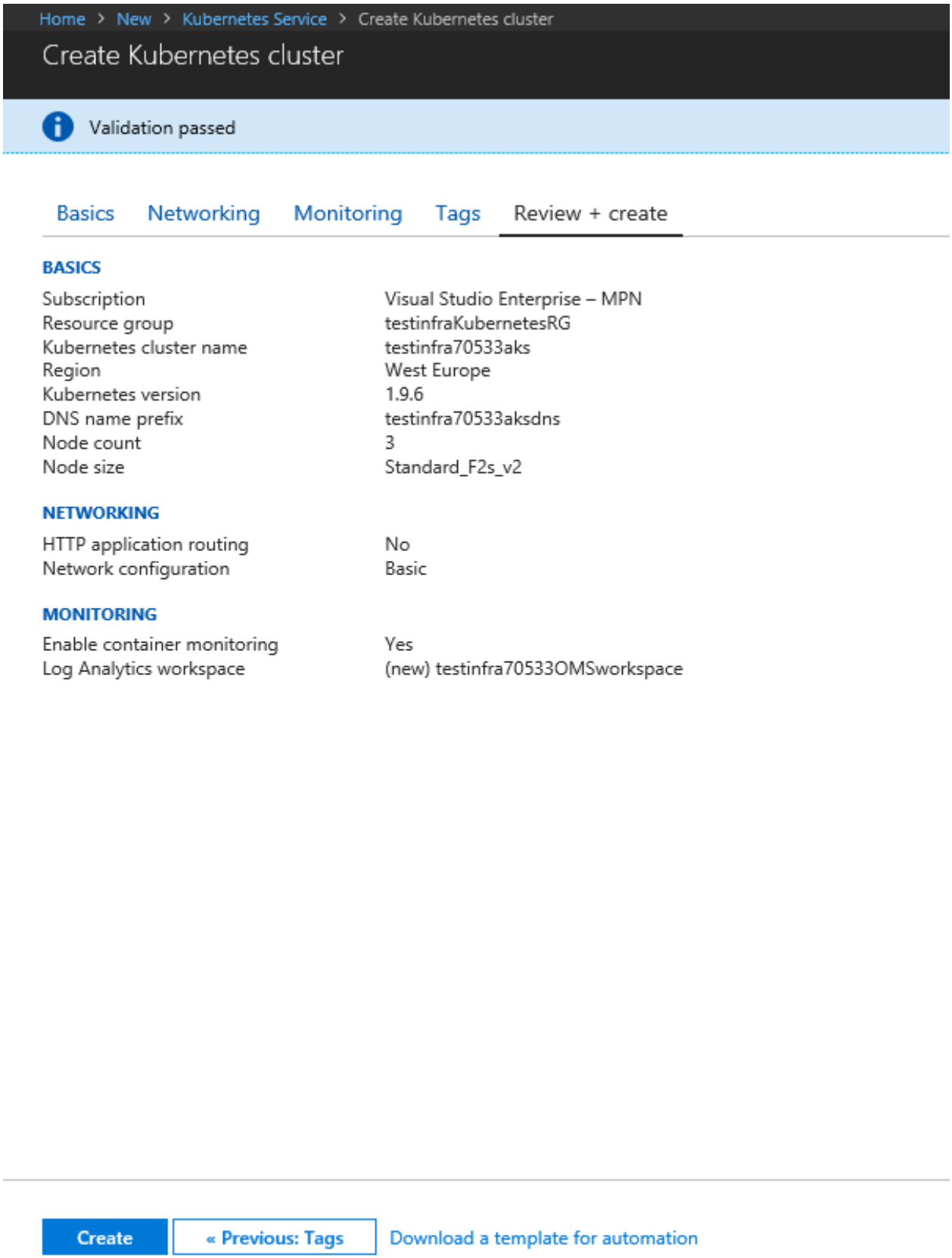
AKS supports two kinds of network configuration: **basics** and **advanced**. The difference between them is the advanced networking places your pods in an Azure Virtual Network (VNet) that you can configure, providing them with an automatic connectivity to the VNet resources:



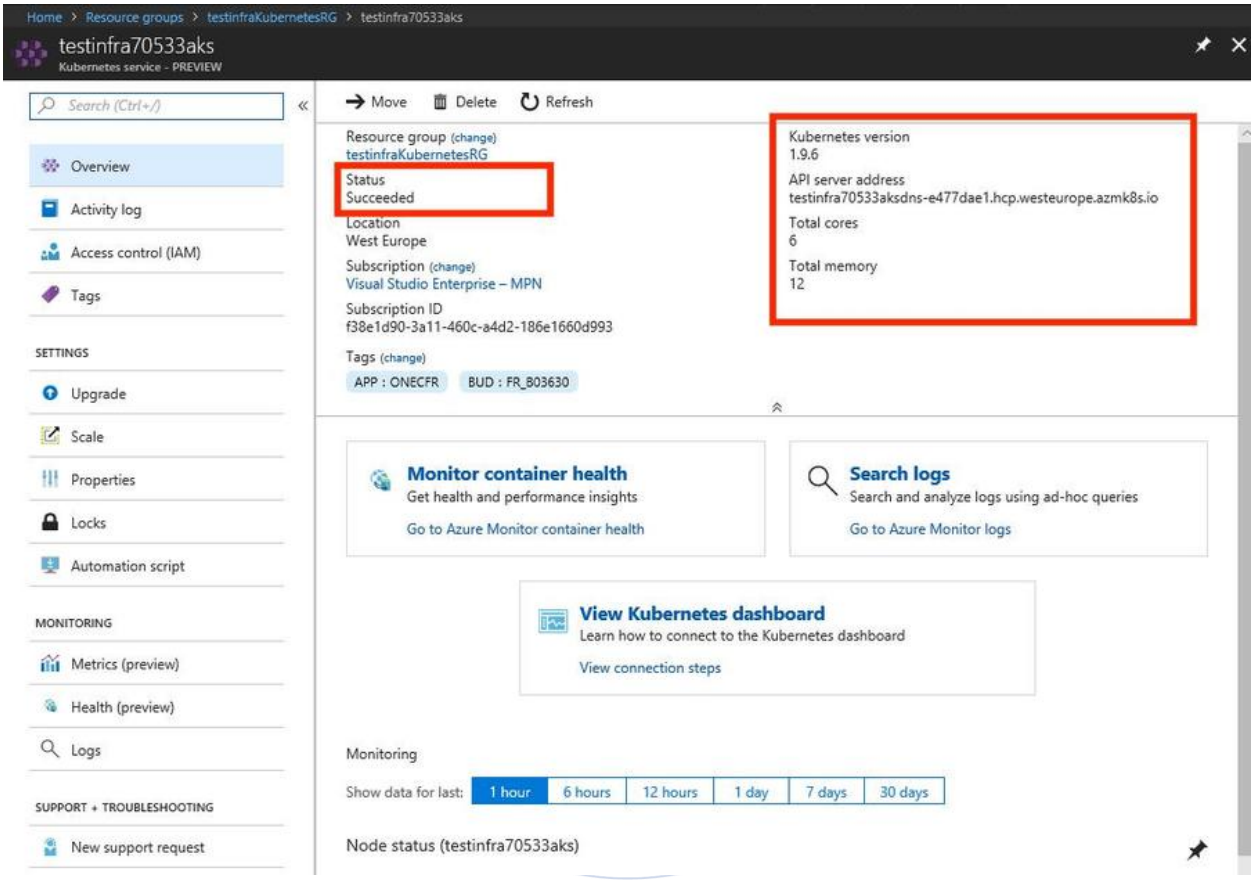
Kubernetes in Azure supports working with OMS (**Log Analytics**) to perform the infrastructure-level monitoring strategy. You can get some basics metrics to monitor the nodes, such as CPU and memory usage, and the health of each node:



Finally, there is a summary of the information that you've filled in before deploying. All the information will be validated by Microsoft Azure:


Home > New > Kubernetes Service > Create Kubernetes cluster
Create Kubernetes cluster
Validation passed
Basics
Networking
Monitoring
Tags
Review + create
BASICS
Subscription
Visual Studio Enterprise – MPN
Resource group
testinfraKubernetesRG
Kubernetes cluster name
testinfra70533aks
Region
West Europe
Kubernetes version
1.9.6
DNS name prefix
testinfra70533aksdns
Node count
3
Node size
Standard_F2s_v2
NETWORKING
HTTP application routing
No
Network configuration
Basic
MONITORING
Enable container monitoring
Yes
Log Analytics workspace
(new) testinfra70533OMSworkspace
Create
« Previous: Tags
Download a template for automation

Usually, the deployment will take 3 - 5 minutes to complete. After a successful deployment, you can go to the **Overview** blade of Kubernetes Service and note that the status is **succeeded**. You can also check some other information of your deployed Kubernetes Cluster, such as Kubernetes version and API server address:



At the moment, ACS and AKS coexist in Azure. However, Kubernetes is winning over all of these competitors. For many reasons, AKS is becoming more and more important in Microsoft's roadmap.

Implementing and Managing Azure Virtual Networks

Azure networking provides different components in the cloud to help customers create and manage virtual private networks in Azure; it also enables connecting to other virtual networks or their own existing on-premise networks.

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

- The basic concepts and components of Azure networking
- How to implement Azure virtual networks
- How to manage Azure virtual networks
- Traffic filtering and routing
- How to configure a multi-region strategy with Azure Traffic Manager
- How to configure different types of Azure virtual network hybrid and multisite connectivity
- How to monitor Azure networking services
- How to secure Azure virtual networks

Planning and designing Azure virtual networks

Networking management is a critical and important topic within an organization. Networking connects people, applications, and services. These resources keep businesses running.

Analyze network requirements

Designing a network can be a challenging task. The first step is to understand networking requirements. The main considerations in planning and designing networking solutions are as follows:

- **Connectivity**: This determines the types of connection, public cloud, private cloud, or cross-premise connectivity. The common elements of Microsoft Cloud connectivity will help you to do a check before starting. Here is the link: https://docs.microsoft.com/en-us/office365/enterprise/common-elements-of-microsoft-cloud-connectivity.

- **Scalability**: This determines whether the designed network can grow to involve new users, new services, and new applications without affecting the existing services.
- **Availability**: This determines whether the designed network is consistent with reliable performance and offers reasonable response times from and to any host within the network.
- **Security**: This determines the location of security devices, filters, and firewall as well as compatibility with security requirements within the organization.
- **Manageability**: This determines whether the network can be managed effectively and efficiently.

These considerations are becoming less challenging when working with Azure virtual networking in Microsoft Azure. The greatest advantage when working with Azure is that we can make our concepts and design a reality in a "one-click" way. **Go cloud** made a significant difference for organizations which are on the road to digital transformation. When they're at the transition stage, while moving to the cloud customers usually need networking functionality similar to when they were using an on-premise deployment. Microsoft Azure networking components offer a range of functionalities and services that can help organizations design and manage their cloud networking resource. Among all the Azure networking services, Azure virtual network plays a key role.

An **Azure virtual network**, also called a **VNet**, defines an organization's network in the cloud. It is a logical isolation of the Azure cloud. Within each VNet, there are one or more subnets. The subnets facilitate segmentation of networks, providing a way of controlling communication between network resources.

Different from the traditional understanding of networks, a subnet acts as an address range within a VNet. They can be secured by Network Security Groups (NSGs), which we'll cover in this chapter. So, it is very important to define the address space of a VNET. Each VNet that you connect to another VNet must have a unique address space. Each VNet can have one or more public or private address ranges assigned to its address space.

You can get more inspiration from the Azure documentation about how to plan and design virtual networks in Azure as follows:

https://docs.microsoft.com/en-us/azure/virtual-network/virtual-network-vnet-plan-design-arm

Determine the type of connectivity

Before we begin designing a virtual network within the organization, it is important to know whether the transformation will happen in-cloud only, interconnected cloud-only, or cross-premise and interconnected cross-premise as well as which services are in the cloud or on-premise. A cross-premise network should have hybrid network connections between Azure and the on-premise network. These connections can be realized through an Azure gateway, ExpressRoute, and more. We'll cover it later in this chapter.

Determine the address space

In Azure, the address space is defined by the network prefix notation or **classless Inter-Domain Routing** (**CIDR**), which is compact for allocating IP addresses and IP routing prefix. The notation is constructed with an IP address, a slash ('/'), and a decimal number: Here is an example of CIDR: 10.0.0.0/16.

In Azure, you can use any range in RFC 1918. In Azure, the smallest supported CIDR is /29, and the largest is /8.

In a single Subnet, Azure reserves the first and last IP addresses of each subnet for protocol conformance, as well as the x.x.x.1-x.x.x.3 addresses of each subnet.

You can consider the following recommendations to define the size of your subnet:

Number of virtual machines	Subnet Size
Up to 3	/29
Up to 11	/28
Up to 27	/27
Up to 59	/26
Up to 123	/25

For a number of technical reasons, Azure doesn't recommend users add the following address ranges:

- Multicast: 224.0.0.0/4
- Broadcast: 255.255.255.255/32
- Loopback: 127.0.0.0/8
- Link-local: 169.254.0.0/16
- Internal DNS: 168.63.129.16/32

Assigning static, public, and private IP addresses

IP addresses are like our name at home or at work, but are an identity on the internet. Assigning these IP addresses can be done by using Azure Portal, Azure CLI, or AzurePowerShell.

Public IP versus private IP

We can take an example as public IP is your official name and private IP is our nickname. The translate job between your official name and your nickname is done by a Network Address Translation (NAT), which is a process in which the router translates the private IP Address into a public IP so that you'll be eligible to enter the internet through your **Internet Service Provider** (**ISP**).

The Internet Assigned Numbers Authority (IANA) reserves some IP address blocks for usage as private IP addresses, as follows:

- 192.168.0.0 - 192.168.255.255 (about 65,536 IP addresses)
- 172.16.0.0 - 172.31.255.255 (1,048,576 IP addresses)
- 10.0.0.0 - 10.255.255.255 (16,777,216 IP addresses)

To know more about how to create, change, or delete a public IP address, check here: https://docs.microsoft.com/en-us/azure/virtual-network/virtual-network-public-ip-address.

Dynamic IP versus static IP

A dynamic IP is an IP address that is constantly changing. A static IP, on the other hand, is one that remains the same. In most cases, dynamic IP is used thanks to the Dynamic Host Control Protocol (DHCP), which is a protocol used to provide automatic, rapid, and central management for the distribution of IP addresses within a network.

DNS for resources in Azure VNets

DNS, which is the abbreviation of **Domain Name System**, is responsible for translating a public hostname, such as a website, public portal, or internal service name such as intranet portal, to its IP address. A simple example may help you understand better; www.pack.com can be translated by DNS to a public address such as 191.239.213.197. Within the Azure virtual network, it is possible to use custom DNS as well as Azure DNS.

Azure DNS provides reliable and secure name resolution using the Microsoft Azure infrastructure. Azure DNS supports all the common record types such as A, AAAA, CNAME, MX, NS, PTR, SOA, SRV, and TXT records.

Azure has public Azure-provided hostnames as well as DNS Private Zones, which provides name resolution both within a virtual network and between virtual networks. To know more about Azure DNS Private Zones scenarios, check the following link: https://docs.microsoft.com/en-us/azure/dns/private-dns-scenarios.

You can manage your DNS records using the same credentials, APIs, tools, and billing as your other Azure services via the Azure Portal, Azure PowerShell cmdlets, and Azure CLI.

Applications requiring automatic DNS management can integrate with the service via the REST API and SDKs. You can know more about it by going to the following link: https://docs.microsoft.com/EN-US/azure/virtual-network/virtual-networks-name-resolution-for-vms-and-role-instances.

Implementing Azure virtual networks

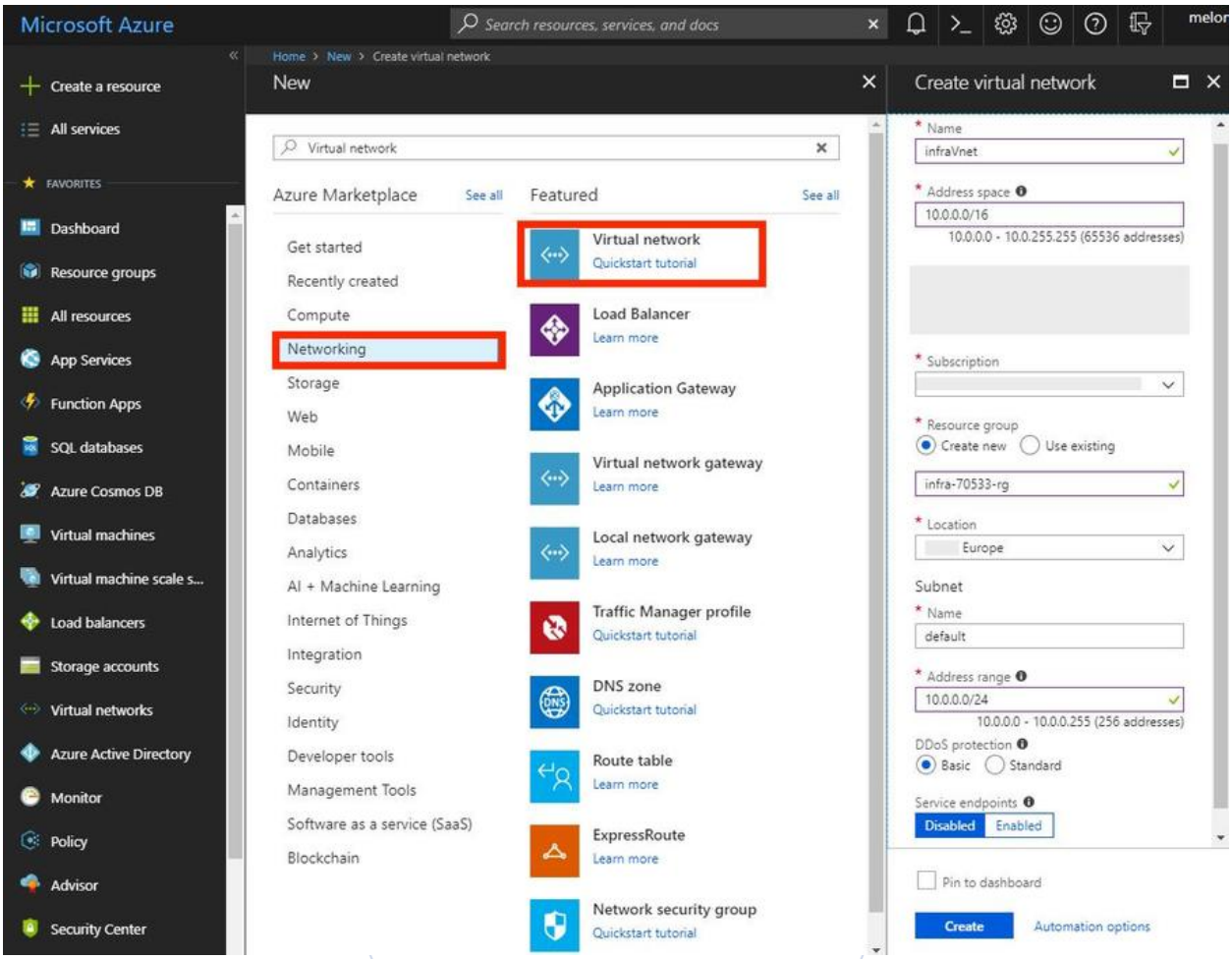
To create an Azure virtual network, you can use the different ways covered in upcoming subsections.

Creating an Azure virtual network

You can create a virtual network via Azure Portal, Azure CLI, PowerShell as well as ARM template.

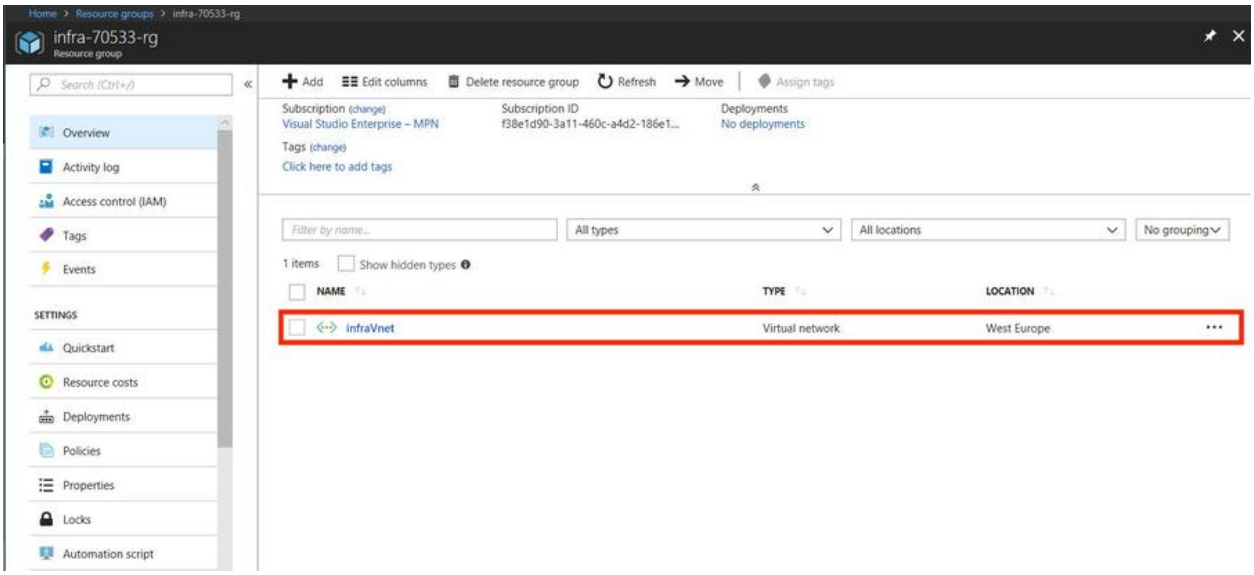
Via Azure Portal

To create a virtual network, go to Azure Portal and click on **Create a resource**. Then, in the **Networking** category, click on **Virtual network**. You can start filling in the basic information, as shown in the following screenshot:

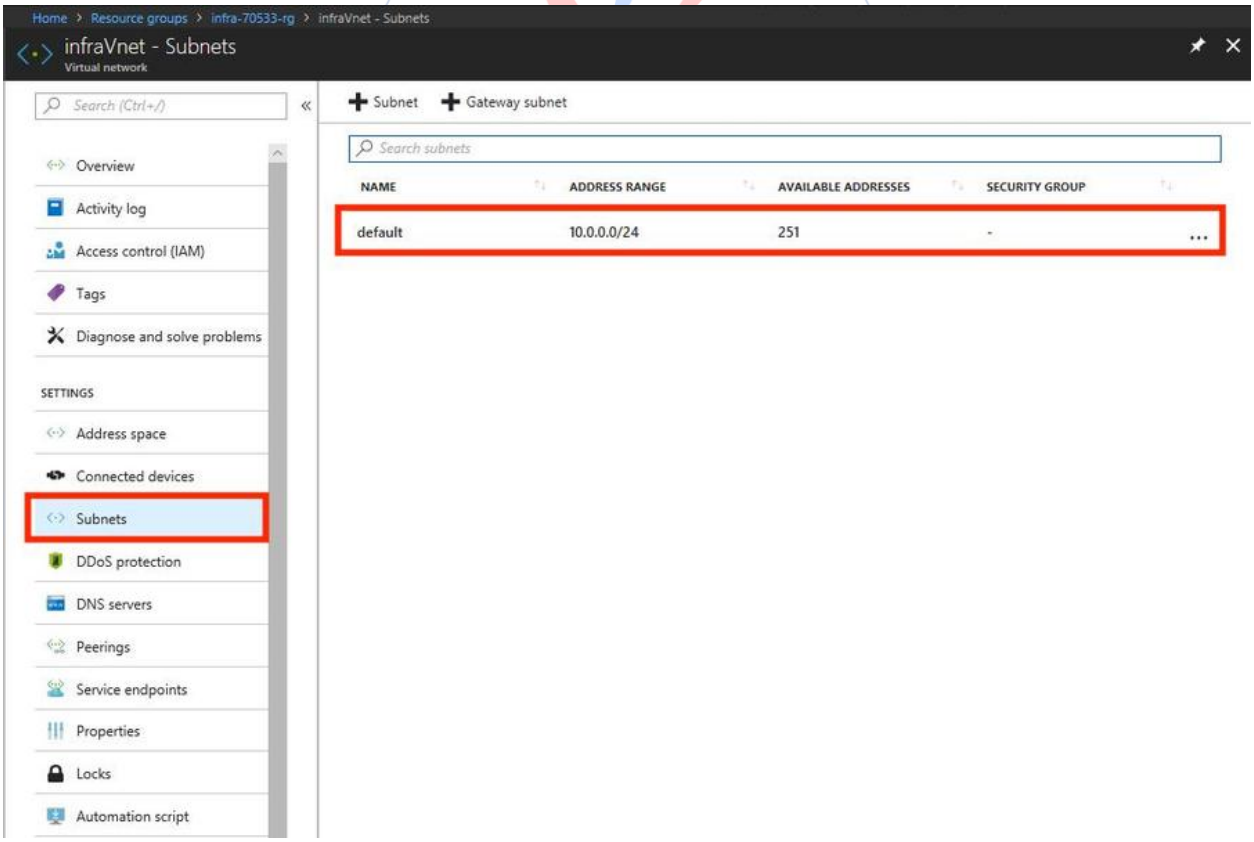


It is very important to specify the address range of your VNet and Subnets. The address range for the subnets or the whole virtual network must be specified with the CIDR notation, defined in RFC1918; within the same Azure virtual network, the address range of the subnets cannot overlap with each other. Every time you create a new Azure virtual network, it will create a default subnet. Finally, you can click on **Create**. The deployment will take just a few minutes.

After creating a VNet successfully, you can go to the **Resource group**. You can see that your VNet is deployed as shown in the following screenshot. You can find information regarding the virtual network such as address space and DNS servers in the **Overview** blade:



If you go to the VNet, then click on the **Subnets** blade. You will see that the default subnet is in the VNet, as shown in the following screenshot:

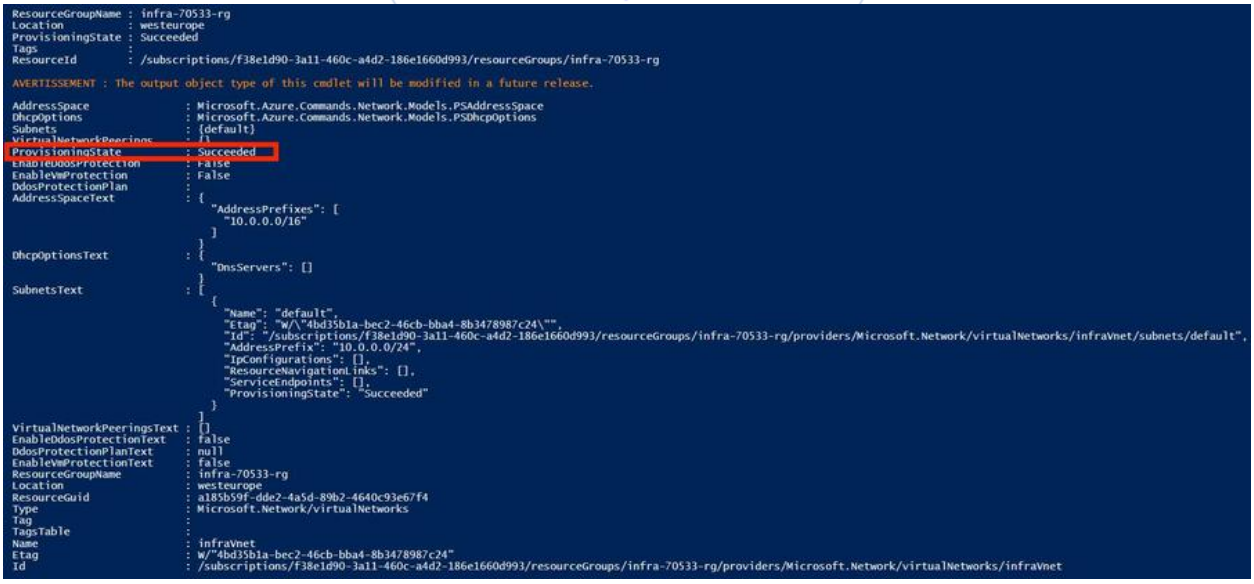


Via Azure PowerShell

You can also create a virtual network using PowerShell. The commands are as follows:

```
# Create a resource group.
$resgroupname = "infra-70533-rg"
$location = "West Europe"


New-AzureRmResourceGroup -Name $resgroupname -Location $location

# Create a new virtual network
$virtualNetwork = New-AzureRmVirtualNetwork -ResourceGroupName $resgroupname -Location $location -Name infraVnet -AddressPrefix 10.0.0.0/16

# Define subnet configuration
$subnetConfig = Add-AzureRmVirtualNetworkSubnetConfig -Name default -AddressPrefix 10.0.0.0/24 -VirtualNetwork $virtualNetwork

#Apply the designed configuration to new virtual network
$virtualNetwork | Set-AzureRmVirtualNetwork
```


If everything is okay, you will see output similar to the following screenshot:



Via Azure CLI

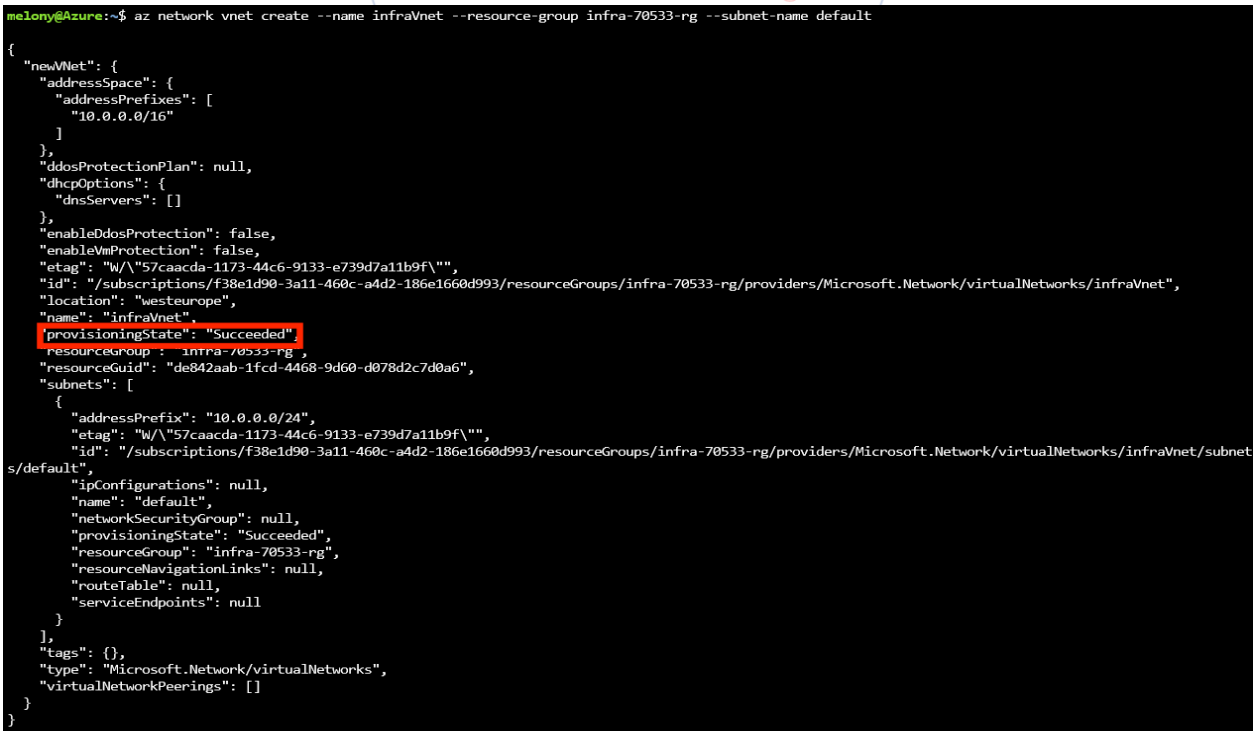
To achieve the same result, you can also use Azure CLI as shown in the following code. You can start by creating a new resource group ():

```
az group create --name #yourresourcegroup# --location #yourlocation#
```

You can create a new Azure VNet by the following command:

```
az network vnet create \
--name #youvnetname# \
--resource-group #yourresourcegroup# \
--subnet-name #yoursubnetname,here is default#
```

If everything is okay, you will see an output similar to the following screenshot. If ProvisioningState is marked Succeeded as shown below, we have created a Azure VNet by using Azure CLI:

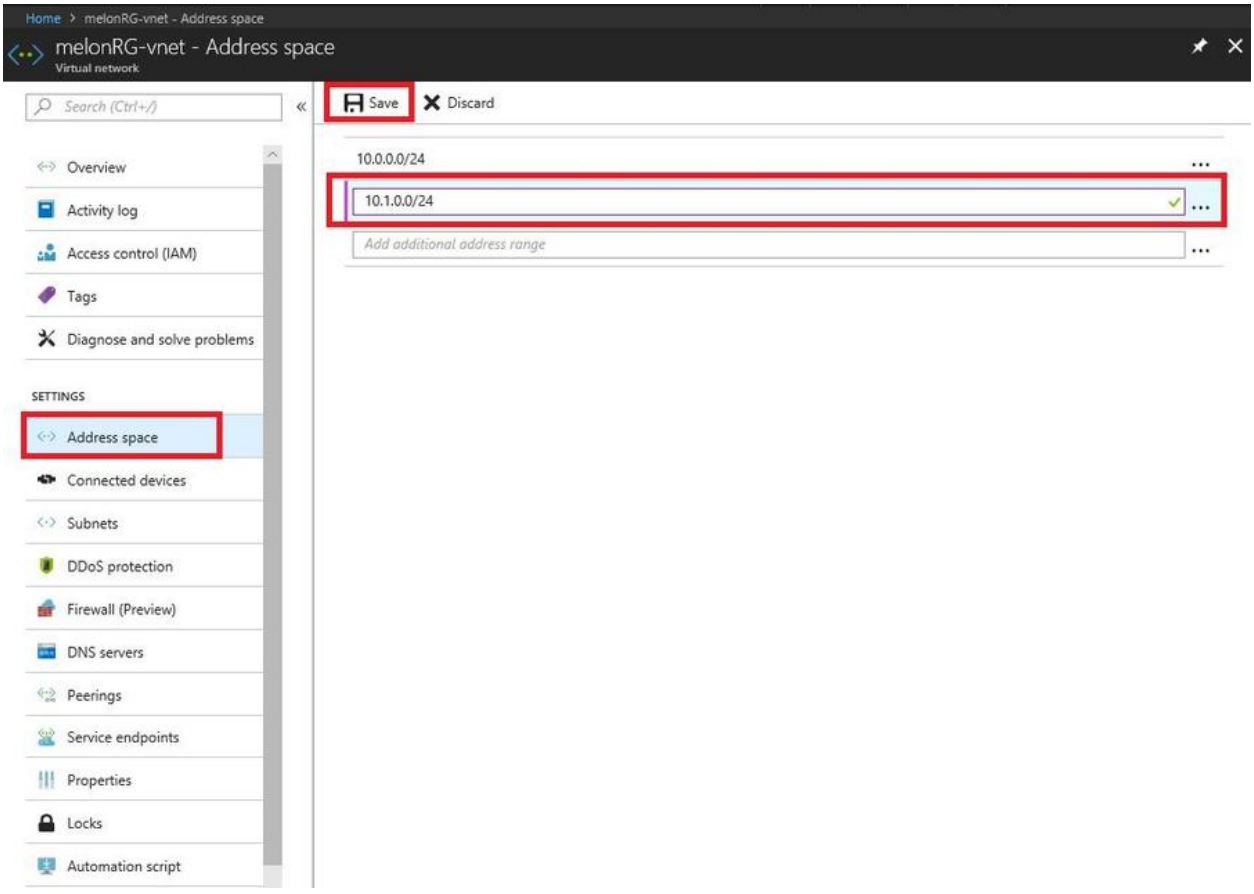


Updating the Azure virtual network

It is possible to add or remove address ranges and change DNS servers after creating the Azure virtual network. You can do that using Azure Portal, Azure CLI, or PowerShell.

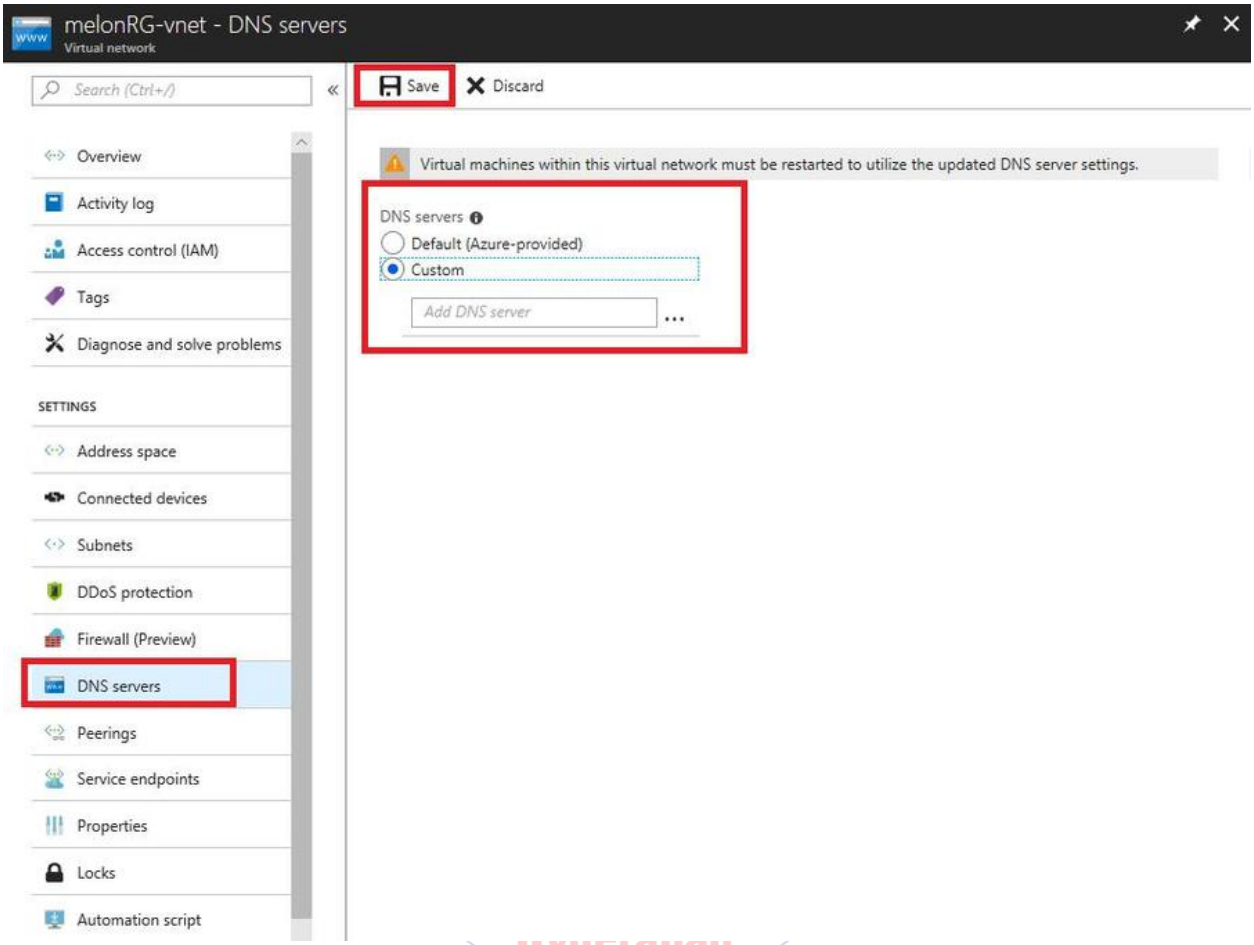
Via Azure Portal

To add or remove the address range, you can go to the **Address space** blade of the Azure virtual network and add or remove a space range, as shown in the following screenshot:



Name resolution for the devices connected to the current virtual network is managed by Azure DNS by default. Users can decide whether they want to choose the Azure internal DNS server or a custom DNS server.

To set a DNS server, you can go to the **DNS server** blade to use Azure default DNS or a custom server, as shown in the following screenshot:



Via Azure CLI

You can use the following Azure CLI cmdlet to update the virtual network:

```
az network vnet update -g #yourresourcegroupname# -n #VNetname@ --dns-servers 10.1.0.5
```

There are some other settings as well that can be updated using the same command. You can check the following link for more information: https://docs.microsoft.com/en-us/cli/azure/network/vnet?view=azure-cli-latest#az-network-vnet-update.

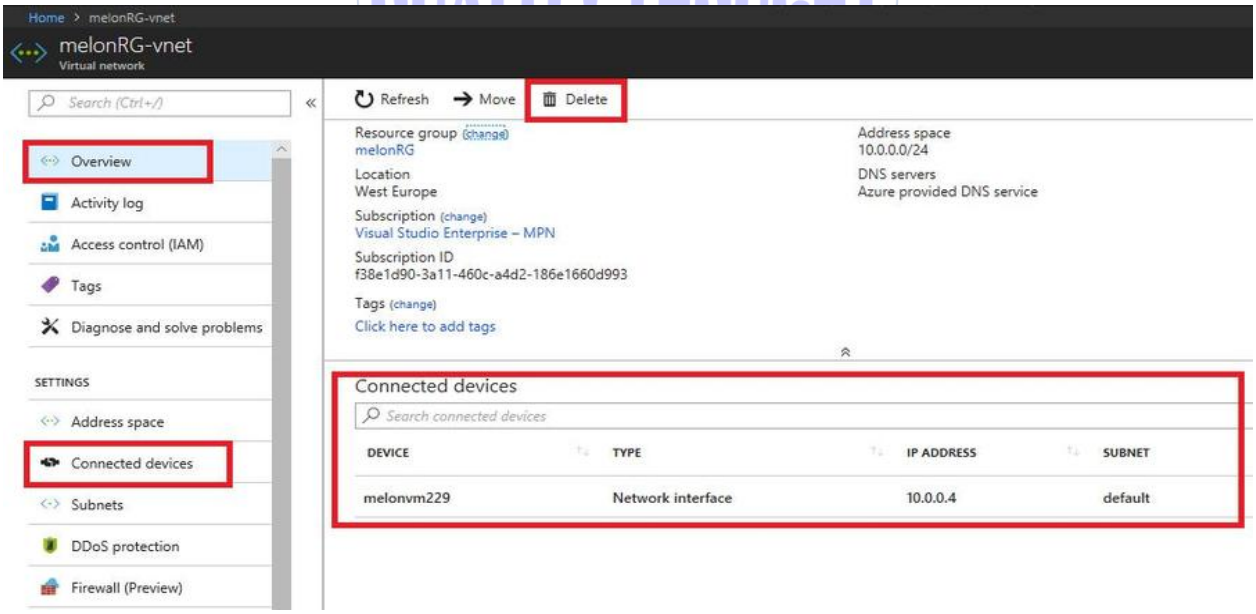
Via Azure PowerShell

The same result can be achieved using the Set-AzureRmVirtualNetwork cmdlet, set the expected state of a virtual network:

```
$virtualNetwork = New-AzureRmVirtualNetwork -Name MyVirtualNetwork -ResourceGroupName
TestResourceGroup -Location westeurope -AddressPrefix "10.0.0.0/16"
$virtualNetwork | Set-AzureRmVirtualNetwork
```

Delete Azure virtual network

When a virtual network is no longer needed, you can delete it via Azure Portal by clicking on **Delete** in the **Overview**blade. Just to remind you, when we said a VNet is no longer in use we meant there are no devices connected with this VNet. You can check the connected devices in the **Overview** or **Connected devices** blades in the virtual network (as shown in the following screenshot):



You can also use the following Azure CLI cmdlet to delete a virtual network:

```
az network vnet delete -g MyResourceGroup -n myVNet
```

Alternatively, you can delete the current virtual network using Azure PowerShell:

```
Remove-AzureRmVirtualNetwork -Name #virtualnetworkname# -ResourceGroupName #resourcegroupname#
```

Managing Azure virtual networking

Network traffic from the internet, on-premise, or any other cloud providers to Azure can be routed, filtered, and distributed thanks to the different PaaS services of Azure networking. In this section, we'll discuss different Azure networking components to route, filter, and distribute networking traffic.

Routing network traffic

In Azure, there are a couple of options to route network traffic between subnets or connected VNets in Azure, on-premise, and the internet. Typically, you can optionally use **User-defined routes** (**UDR**) to override Azure's default routing or using **Border gateway protocol** (**BGP**) routes through a network gateway.

User-defined routes

Azure routes traffic between Azure, on-premise, and the internet. Azure automatically creates a route table containing a set of routes, which specifies how to route traffic within a virtual network for each subnet within an Azure virtual network and the system default routes will be added to the table.

User-defined routes will be very useful when you want to override some of Azure's system routes with custom routes and add additional custom routes to route tables. Azure routes outbound traffic from a subnet based on the routes in the route table of the subnet. The relationship between the route table and subnet is **one to many**, which means each route table

can be associated to multiple subnets, but a subnet can only be associated to a single route table.

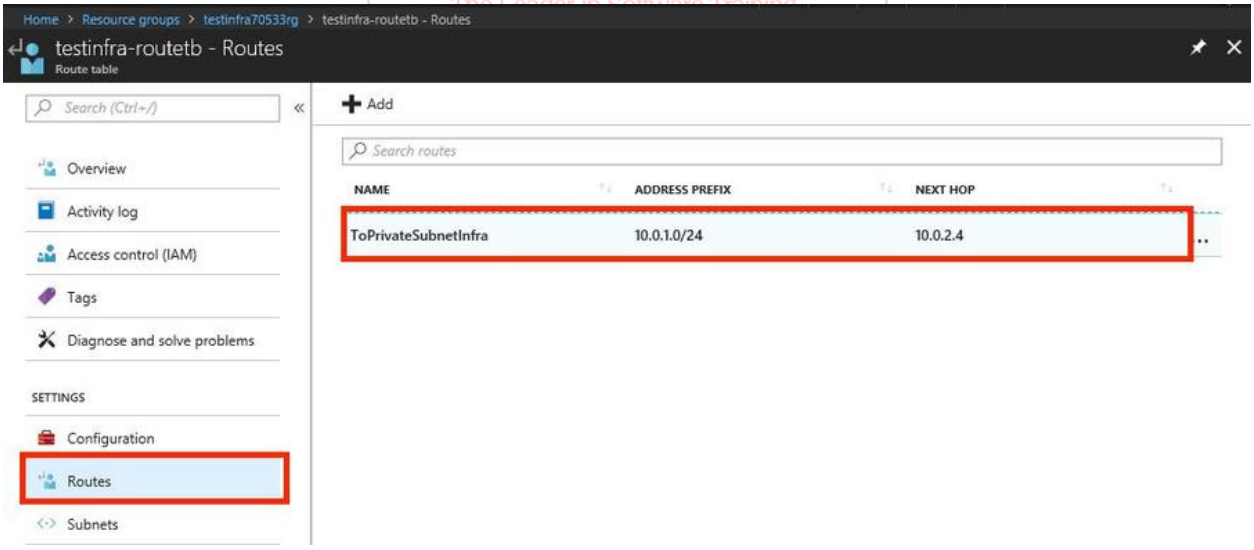
Border gateway protocol

Border gateway protocol (**BGP**) is a standard routing protocol commonly used on the internet to exchange routing and reachability information between two or more VNets. Users can connect the virtual network to an on-premise network using an Azure VPN Gateway or ExpressRoute connection. You can know more about BGP by checking:https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-bgp-overview.

Managing routes in a route table

To know more about how to create a route table, delete a route table, and manage routes in a route table, you can go to the following link: https://docs.microsoft.com/en-us/azure/virtual-network/manage-route-table.

You can use the Azure Portal to check an existing user-defined route by clicking on the **Routes** blade of the created route table, as shown in the following screenshot:



You can also use the following Azure CLI `az network route-table route show` cmdlet to consult the routes in the route table:

```
az network route-table route list -g testinfra70533rg --route-table-name testinfra-routetb
```

In Azure PowerShell, the Get-AzureRmRouteConfig cmdlet is also able to check the route config in the route table. The following is a sample command:

```
Get-AzureRmRouteTable -ResourceGroupName "testinfra70533rg" -Name "testinfra-routetb " | Get-AzureRmRouteConfig -Name "ToPrivateSubnetInfra"
```

Filtering the network traffic

Inbound and outbound network traffic between subnets in Azure can be filtered by source IP address and port, destination IP address and port, and protocol using **Network security groups** (**NSG**) or **Virtual Network Appliances**(**VNA**).

Network security groups

A network security group (NSG) contains a list of security rules that allow or deny network traffic access to resources connected to Azure Virtual Networks (VNet). NSGs can be associated to subnets, individual VMs (classic), or individual network interfaces (NIC) attached to VMs (Resource Manager). When an NSG is associated to a subnet, the rules apply to all resources connected to the subnet. Traffic can further be restricted by also associating an NSG to a VM or NIC.

Virtual Network Appliances

A **Virtual Network Appliance** (**VNA**) is a VM running software that performs a network function, such as a firewall. VNAs are also available that provide WAN optimization and other network traffic functions. NVAs are typically used with user-defined or BGP routes. You can also use a NVA to filter traffic between VNets.

Distributing network traffic

As you probably know, there are different options such as Azure Load Balancer, Traffic Manager, and Application Gateway to distribute network traffic using Microsoft Azure.

These three options work on different layers of the OSI model. They have different feature sets. Users can use these services individually or combine their methods depending on their needs to build the optimal solution. The three options are explained as follows:

- **Azure Load Balancer**: This works at the transport layer, which is level 4 of the OSI model. It provides network-level distribution of traffic across instances generally in the same Azure region. Users can configure public and internal load-balanced endpoints and define rules to map inbound connections to backend pool destinations using TCP and HTTP health-probing options to manage service availability.
- **Traffic Manager**: This is another load-balancing solution that is included within Azure. It works at the DNS level. You can use Traffic Manager to load-balance between endpoints that are located in different Azure regions. Users can configure this load-balancing service to use different traffic distribution methods such as priority, weighted performance, or geographic routing methods.
- **Application Gateway:** This is a load balancer that works at level 7 of the OSI model, which is also known as the application layer. It provides load-balanced solutions for network traffic that is based on the HTTP protocol. It uses routing rules as application-level policies that can offload Secure Sockets Layer (SSL) processing from load-balanced VMs. Similar to ALB, the Application Gateway can be configured as an internet-facing gateway, an internal-only gateway, or a combination of both.

These three options work differently from each other, have different feature sets, and support different scenarios. You can use these services individually or combine their methods, depending on your needs, to build the optimal solution.

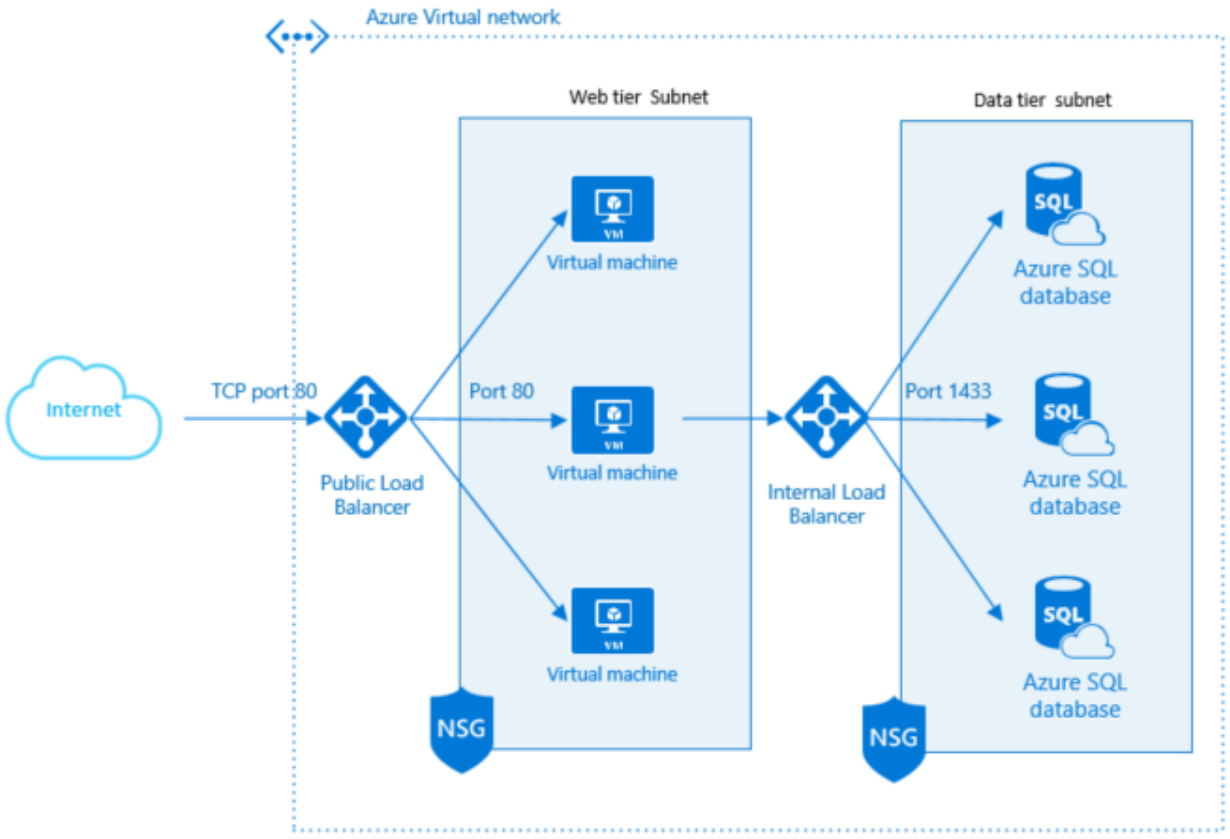
Configure external and internal load balancing

Load Balancer distributes new inbound traffic to arrive at the load balancer's frontend to the backend pool instances based on rules and health probes.

There are two kinds of load balancer:

- **Internet-facing load balancer** or **public load balancer**: This helps to distribute the incoming internet traffic to web front VMs
- **Internal load balancer**: This helps to distribute traffic across VMs inside a virtual network such as data tier VMs

You can combine both types of load balancer using the following schema:



Combine Public Load Balancer and Internal Load Balancer in the same scenario

Azure Load Balancer is available in two SKUs, Basic and Standard, based on scalability, availability, pricing, and other features.

To know more about how to create a load balancer in the basic tier, you can check the following links:

To use the Azure Portal: https://docs.microsoft.com/en-us/azure/load-balancer/quickstart-create-basic-load-balancer-portal.

To use Azure CLI: https://docs.microsoft.com/en-us/azure/load-balancer/quickstart-create-basic-load-balancer-cli.

To use Azure PowerShell: https://docs.microsoft.com/en-us/azure/load-balancer/quickstart-create-basic-load-balancer-powershell.

Load balancing in the standard tier provides a higher level of availability and scalability. It can distribute incoming requests across multiple Azure VMs. There is some more configuration work to do while creating the Standard Load balancer. You can check the following links for more information.

Creating a standard load balancer via the Azure Portal: https://docs.microsoft.com/en-us/azure/load-balancer/quickstart-load-balancer-standard-public-portal.

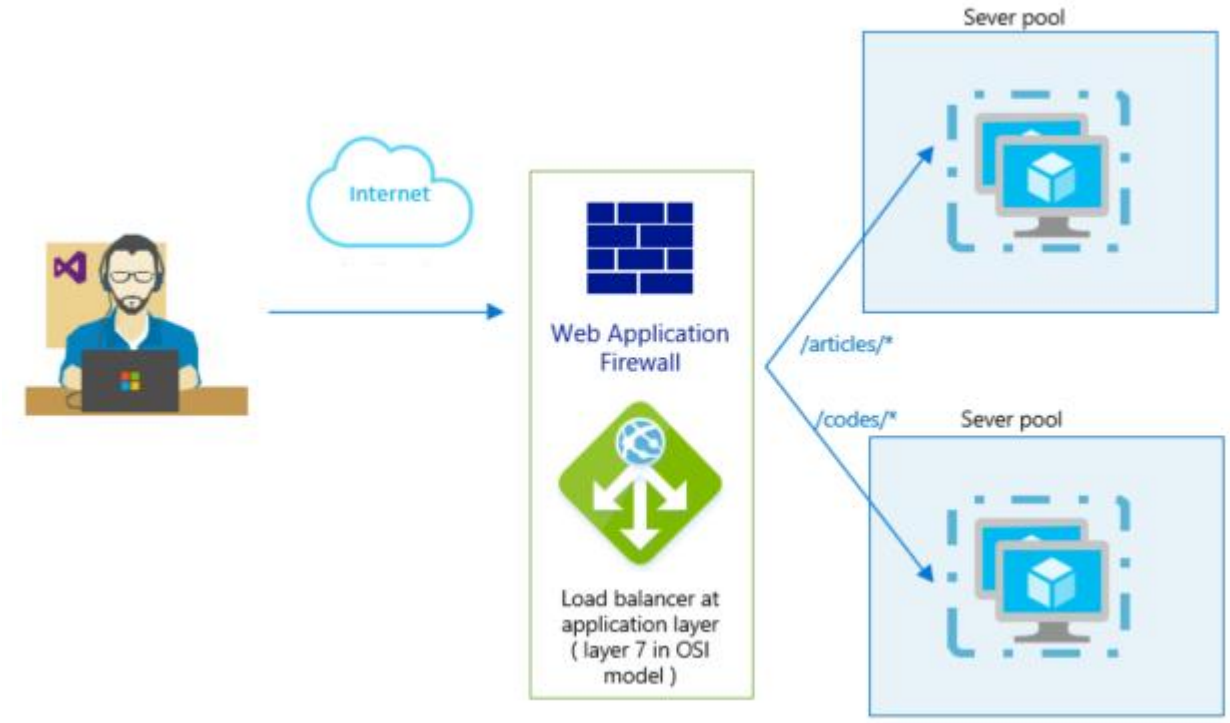
Creating a standard load balancer via Azure CLI: https://docs.microsoft.com/en-us/azure/load-balancer/quickstart-load-balancer-standard-public-cli.

Creating a standard load balancer via Azure PowerShell: https://docs.microsoft.com/en-us/azure/load-balancer/quickstart-create-standard-load-balancer-powershell.

Implementing an Application Gateway

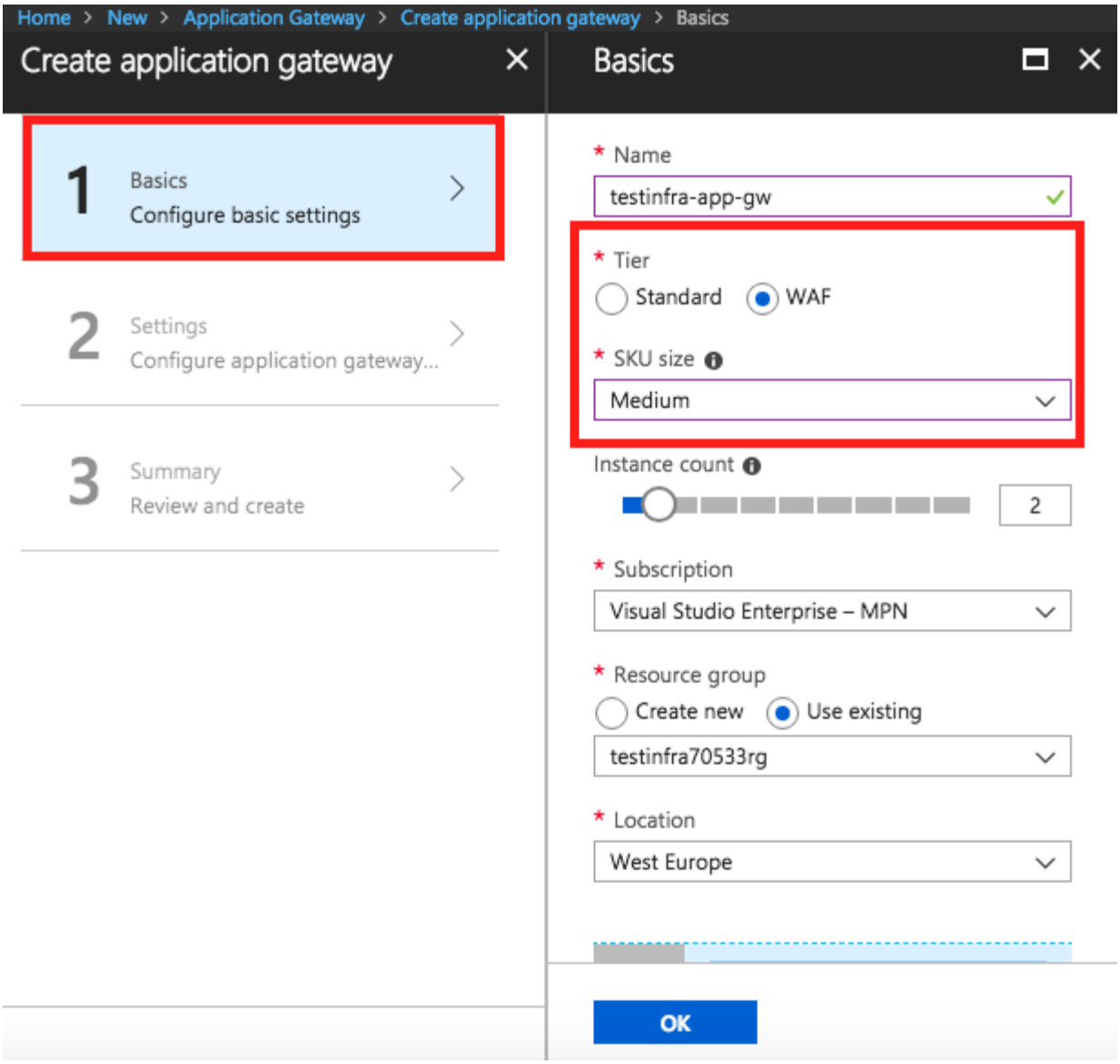
We can create an Application Gateway with a **web application firewall** (**WAF**) feature using Azure Portal. The WAFuses rules from the **Open Web Application Security Project** (**OWASP**) core rule sets to protect your application. These rules include protection against attacks such as SQL injection, cross-site scripting attacks, and session hijacks.

You can see how the Application Gateway works in the following schema. It provides **URL Path-Based Routing**, which allows us to route traffic to the backend server pools based on the URL paths of the request.

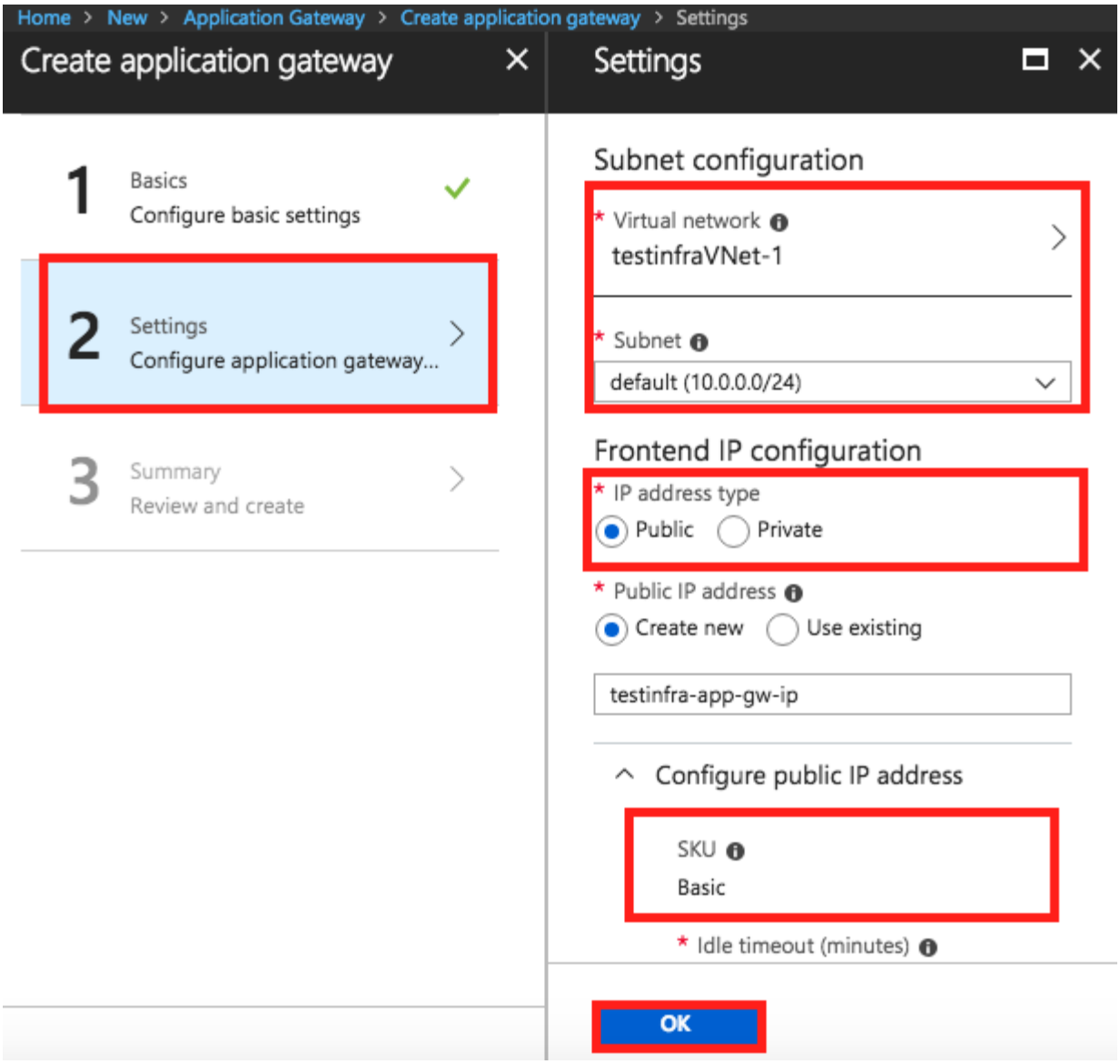


Application Gateway with URL Path-Based Routing and WAF feature

To create an **Application Gateway** via Azure Portal, you can click on**Create a resource**and find **Application Gateway** in the Networking category. After clicking on **Create**, you'll see a form, as shown in the following screenshot:



Choose the tier with **WAF** and **Medium** size. Note that the existing virtual network you'll choose in the next step and the public IP address must be in the same location as your Application Gateway. You can click on **OK** if everything looks good and go to the 2nd step to configure the Application Gateway. You'll then see the following screenshot:



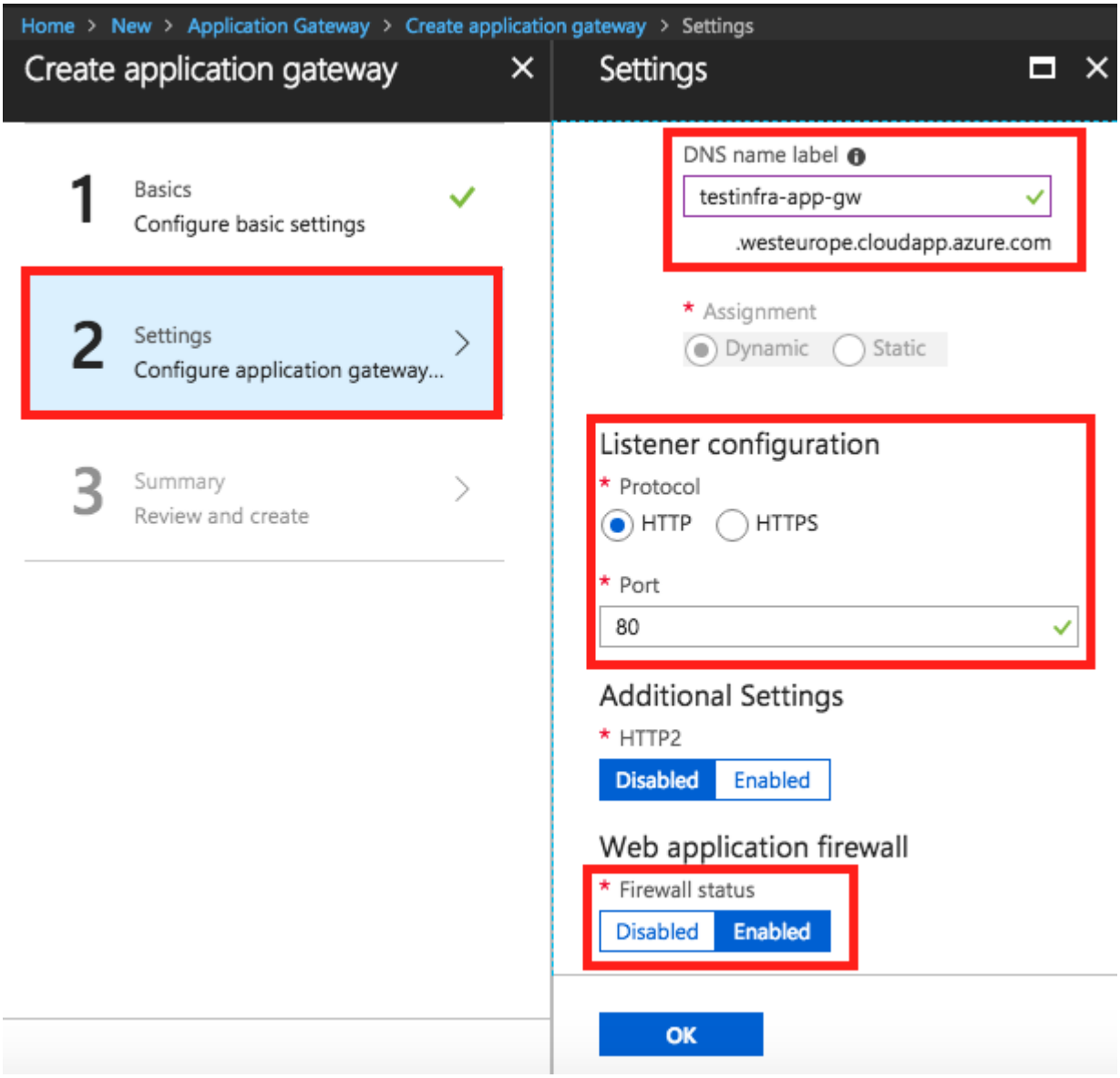
In this step, choose an existing virtual network or create a new virtual network which is in the same location as the Application Gateway. Then choose your frontend IP type. We'll choose Public IP in our case, since the traffic is coming from the internet. As indicated in the preceding screenshot, the SKU for public IP addresses will be defined as BASIC since only the basic tier can be used with an Application Gateway.

There are some other interesting ones in the same step. You should enter a DNS name label for your Application Gateway, which is actually an A record for your public IP address that will

be registered with Azure-provided DNS servers. In our case, the FQDN of our Application Gateway will look like this:

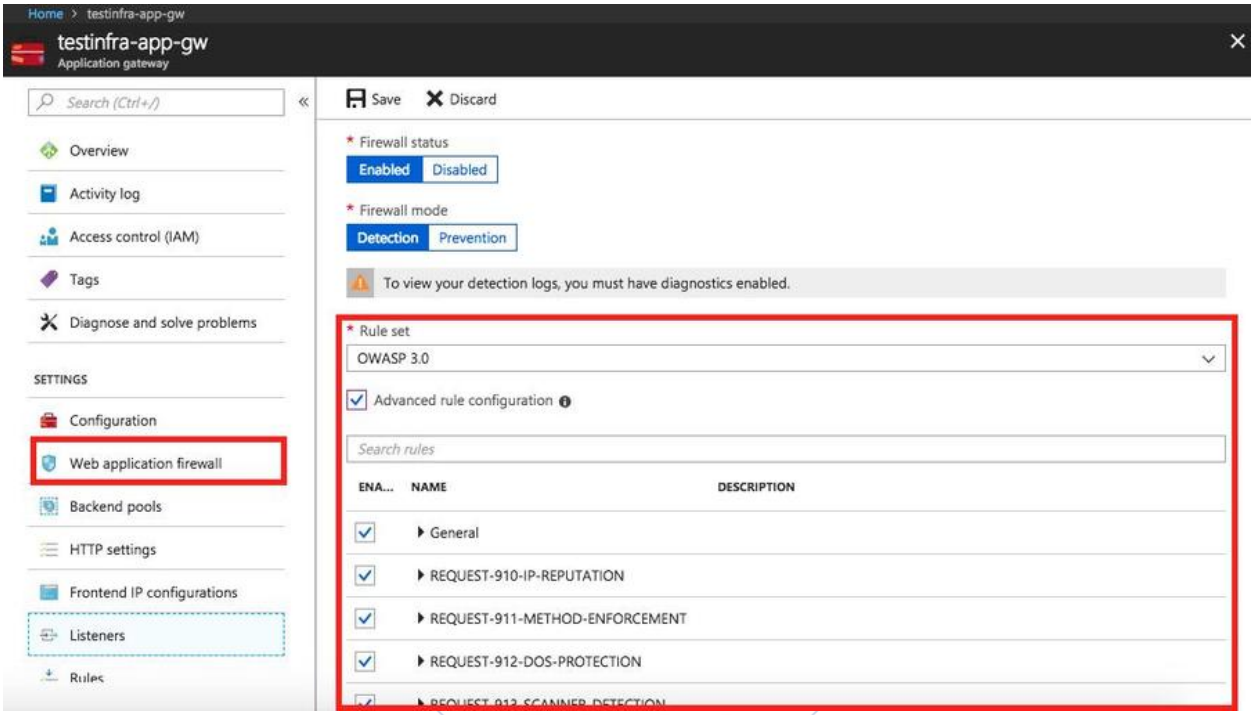
> **testinfra-app-gw.westeurope.cloudapp.azure.com.**

In this step, you should also choose the protocol of your Application Gateway listener, HTTP or HTTPS. Make sure that you have **Enabled** the WAF, as shown in the following screenshot:

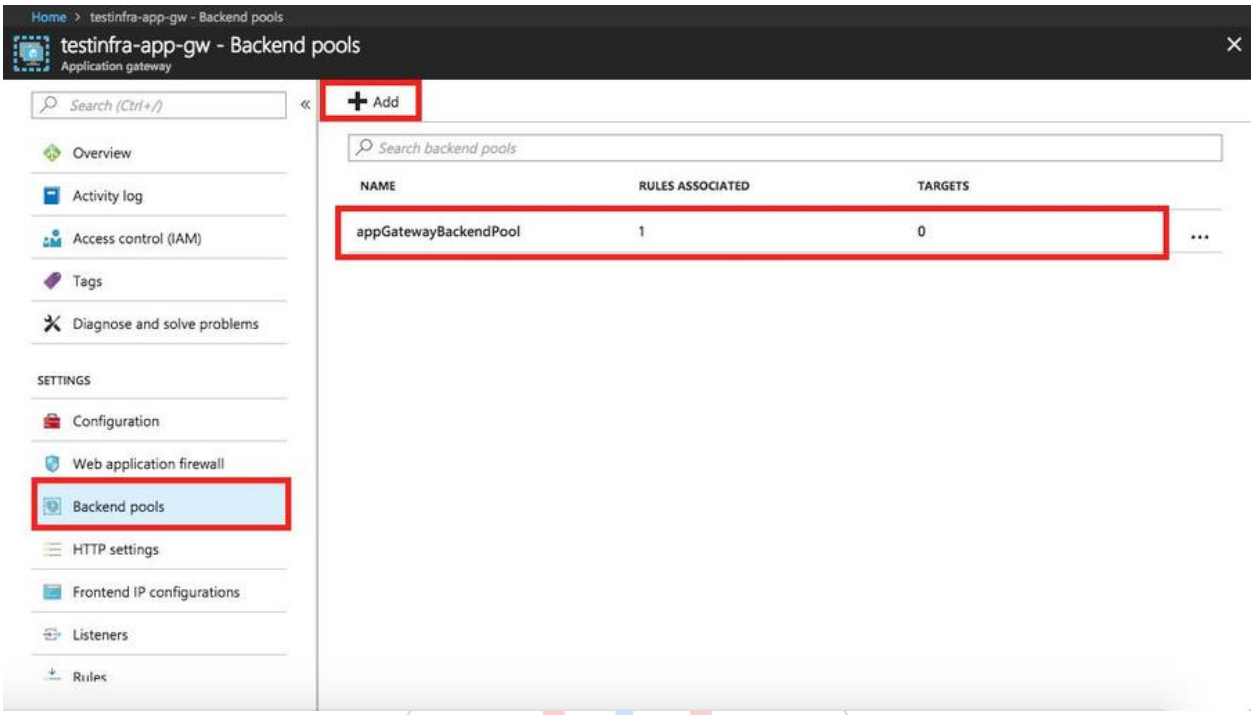


Finally, you'll have a summary page with everything you have entered into the form. Then you can click on OK. The deployment of the Application Gateway will take a couple of minutes.

After creating the Application Gateway successfully, you can go to the **Web application firewall** blade to choose the rule set that you want to use or do some other advanced rule configuration:



You can also go to **Backend pools** blade to add, modify, or delete the Azure **VM,VM Scale Sets**, **IP address**, or **FQDN** in the backend configuration, as shown in the following screenshot:



To create an Application Gateway via Azure PowerShell, you can refer to the following link: https://docs.microsoft.com/en-us/azure/application-gateway/tutorial-restrict-web-traffic-powershell.

Combining Azure load balancing services

Microsoft Azure provides three types of load balancing service to manage network traffic that is distributed. They can be used individually or the methods can be combined depending on your needs. To know more about how to combine Azure Load balancing solution, you can check the following link: https://docs.microsoft.com/en-us/azure/traffic-manager/traffic-manager-load-balancing-azure.

Design and implement cross-premise and multisite connectivity

Microsoft Azure provides capabilities to work as an extension in the cloud of on-premise datacenter for organizations by configuring the network connectivity between the on-premise existing environment and Azure virtual networks.

Microsoft Azure provides the following connectivity options to implement cross-premise and multisite connectivity with Azure virtual networks:

- Point-to-site VPN
- Site-to-site VPNs
- VNet-to-VNet
- VNet peering
- ExpressRoute

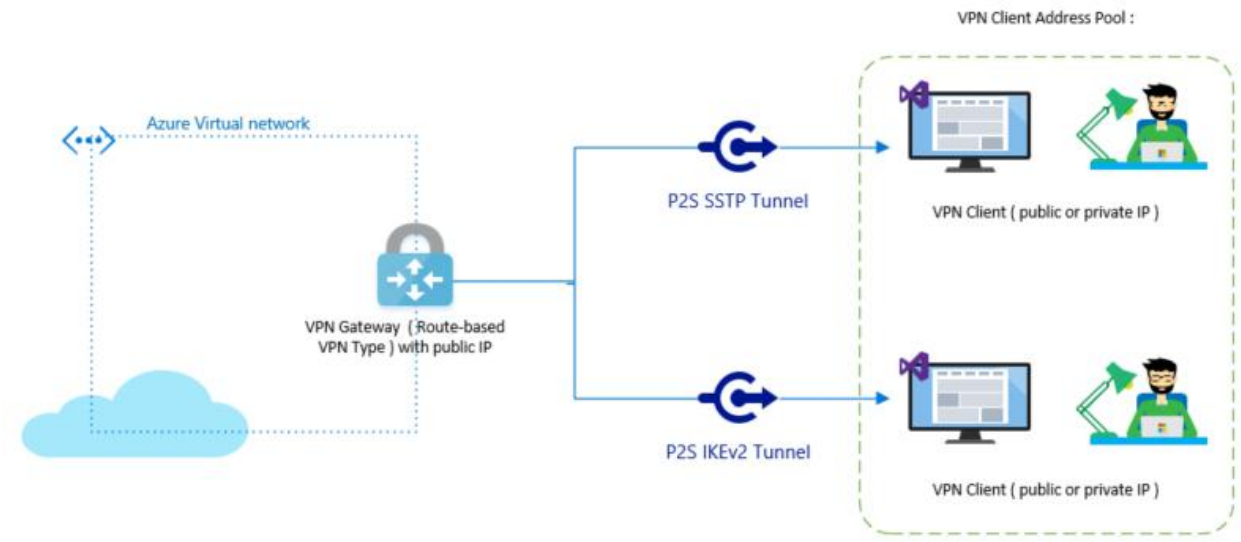
Point-to-site virtual private network (VPN) over IKEv2 or SSTP

Point-to-site virtual private network (VPN) is a solution used to establish a secure connection between an Azure virtual network and a single client computer. The general use case for P2S is to establish connections between individual computers to connect to Azure Virtual Network from a remote location, such as some offsite employer who wants to work from home.

There are two kinds of protocol that can be used to establish a P2S connection:

- **Secure Socket Tunneling Protocol** (**SSTP**): This is a proprietary SSL-based VPN protocol, which can be used to connect from Windows devices (Windows 7 and above)
- **IKEv2 VPN**: This is a standard-based IPsec VPN solution, which can be used to connect from Mac devices (MAC OS X10.11 and above)

The following schema shows what P2S looks like:



As we can see from the schema, P2S also needs a VPN Gateway, which is a virtual network gateway in Azure with route-based VPNs. Each client compute needs to use self-signed certificates (root and client certificate) before connecting to Azure.

Users can generate a certificate using Azure PowerShell or using make cert. You can see how to generate and export certificate for P2S at the following links.

Using Azure PowerShell: https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-certificates-point-to-site.

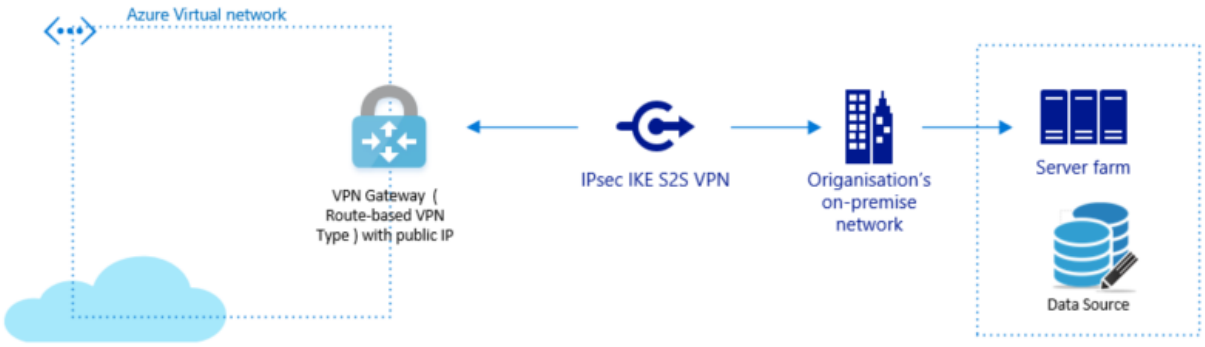
Using make cert: https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-certificates-point-to-site-makecert.

For more information on how to implement P2S via Azure Portal, click on the following link: https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-howto-point-to-site-resource-manager-portal.

You can also implement P2S with Azure PowerShell as indicated in the following link: https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-howto-point-to-site-rm-ps.

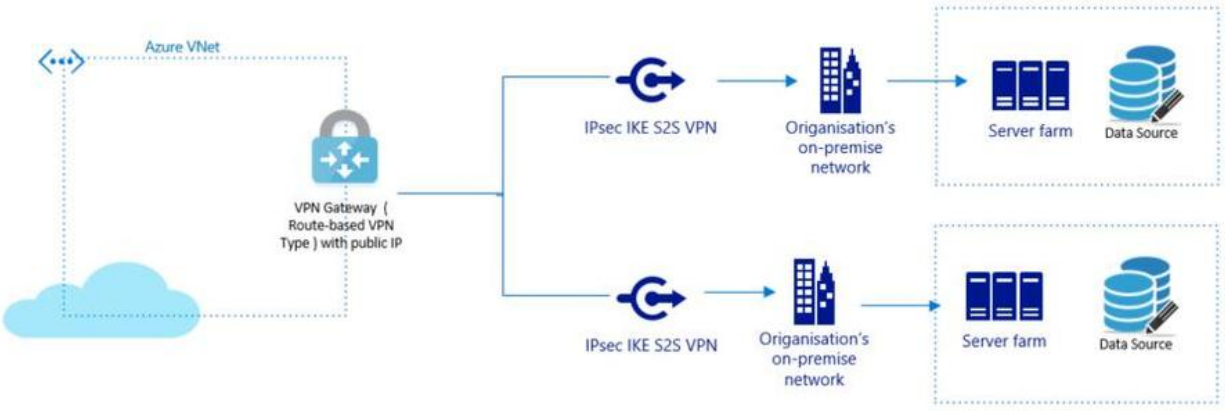
Site-to-site and multisite virtual private network (IPsec/IKE VPN tunnel)

A Site-to-Site VPN gateway connection is a solution used to connect on-premise to an Azure virtual network. This connection is over an IPsec/IKE (IKEv1 or IKEv2) VPN tunnel. To establish this connection, you should have a VPN gateway located on-premise with an externally facing public IP address. This type of connection is shown in the following schema:



Site-to-Site VPN in Azure

While creating more than one VPN connection in Route-Based VPN type so that users can connect to multiple on-premise sites, we called it multisite VPN this is a variation on the S2S connection. This type of connection is shown in the following schema:



Multisite VPN in Azure

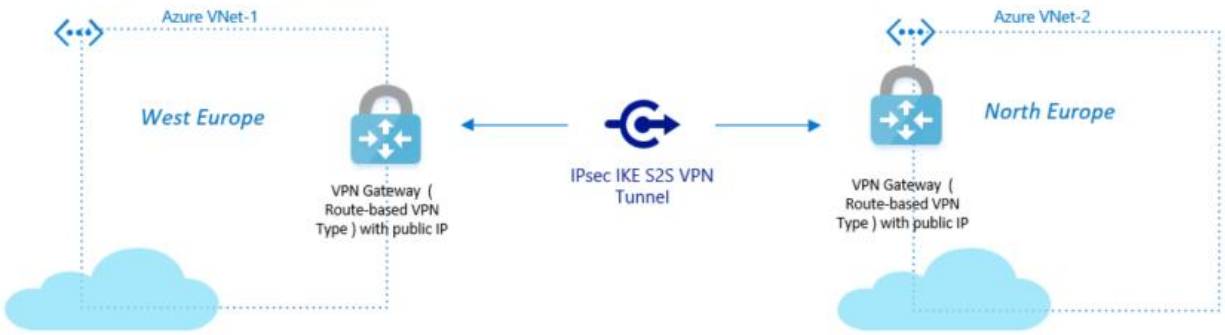
For more information on how to implement S2S via the Azure Portal, click on the following link: https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-howto-site-to-site-resource-manager-portal.

You can also do the same using Azure PowerShell: https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-create-site-to-site-rm-powershell.

Another great way is to use Azure CLI: https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-howto-site-to-site-resource-manager-cli.

VNet-to-VNet virtual private network (IPsec/IKE VPN tunnel)

VNet-to-VNetconnection is a solution used to combine cross-premise connectivity with inter-virtual network connectivity. It also uses the IPsec/IKE VPN tunnel to connect the VNet in the same or different regions, subscriptions, or even deployment models (the classic deployment model and the Azure resource manager deployment model). The following schema shows how to establish a VNet-to-VNet VPN connection:



VNet-to-VNet connection in Azure

For more information on how to implement VNet-to-VNet via the Azure Portal, click on the following link: https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-howto-vnet-vnet-resource-manager-portal.

You can also do this using Azure PowerShell: https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-vnet-vnet-rm-ps.

Another great way is to use Azure CLI: https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-howto-vnet-vnet-cli.

To know how to connect classic VNets to Resource Manager VNets to allow resources located in the separate deployment models via Azure Portal, you can check the following link: https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-connect-different-deployment-models-portal.

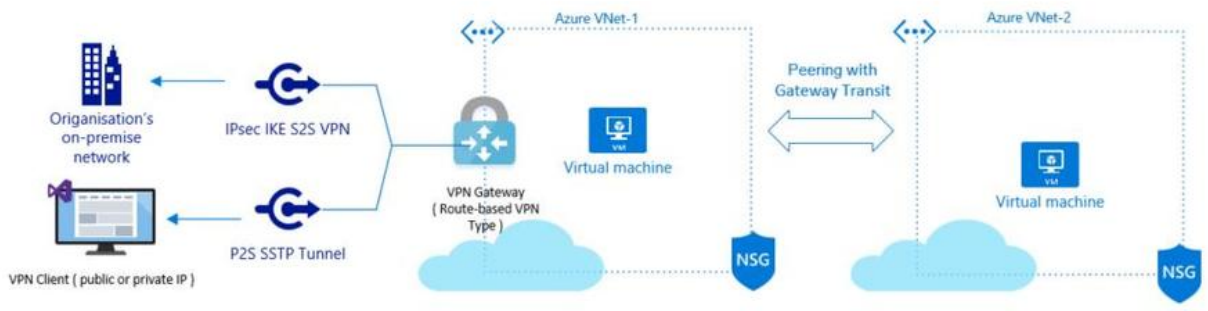
It is also possible to use Azure PowerShell to do that. You can go to the following link for more information: https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-connect-different-deployment-models-powershell.

Virtual network peering (VNet Peering)

Virtual network peering (VNet Peering) is an option provided by Microsoft Azure to connect two Azure virtual networks without using VPN gateways. There are two types of virtual network peering:

- **VNet peering**: This is a way to connect VNets within the same Azure region
- **Global VNet peering**: This is a way to connect VNets across different Azure regions

An example of VNet peering is described in the following schema:

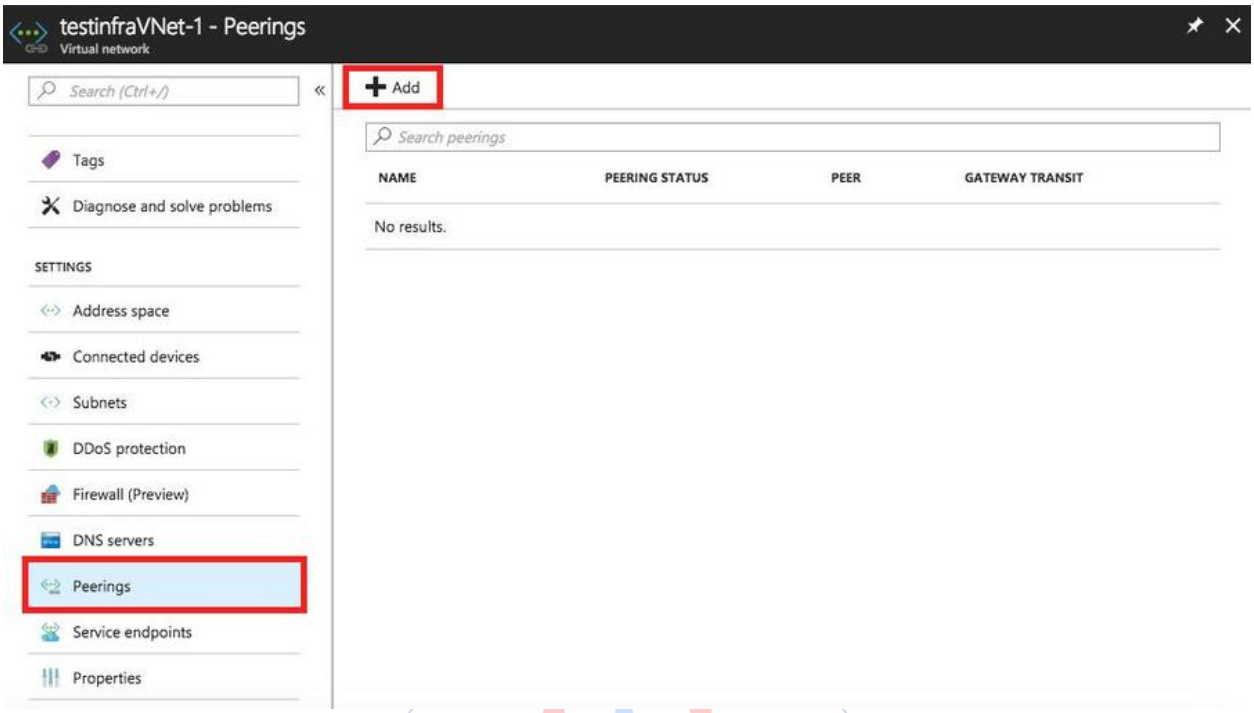


As we can see from this schema, **VNet-1** and **VNet-2** have been connected by a VNet peering. **VNet-1** has a **VPN Gateway** to connect with another organization using S2S VPN and another client compute using **P2SVPN**. The **VPN Gateway** has enabled the Gateway transit peering property, which enables **VNet-2** to use the **VPN Gateway** in the peered virtual network, which is **VNet-1** for cross-premise or VNet-to-VNet connectivity. Connectivity applies to both **VNet -1** and **VNet-2**.

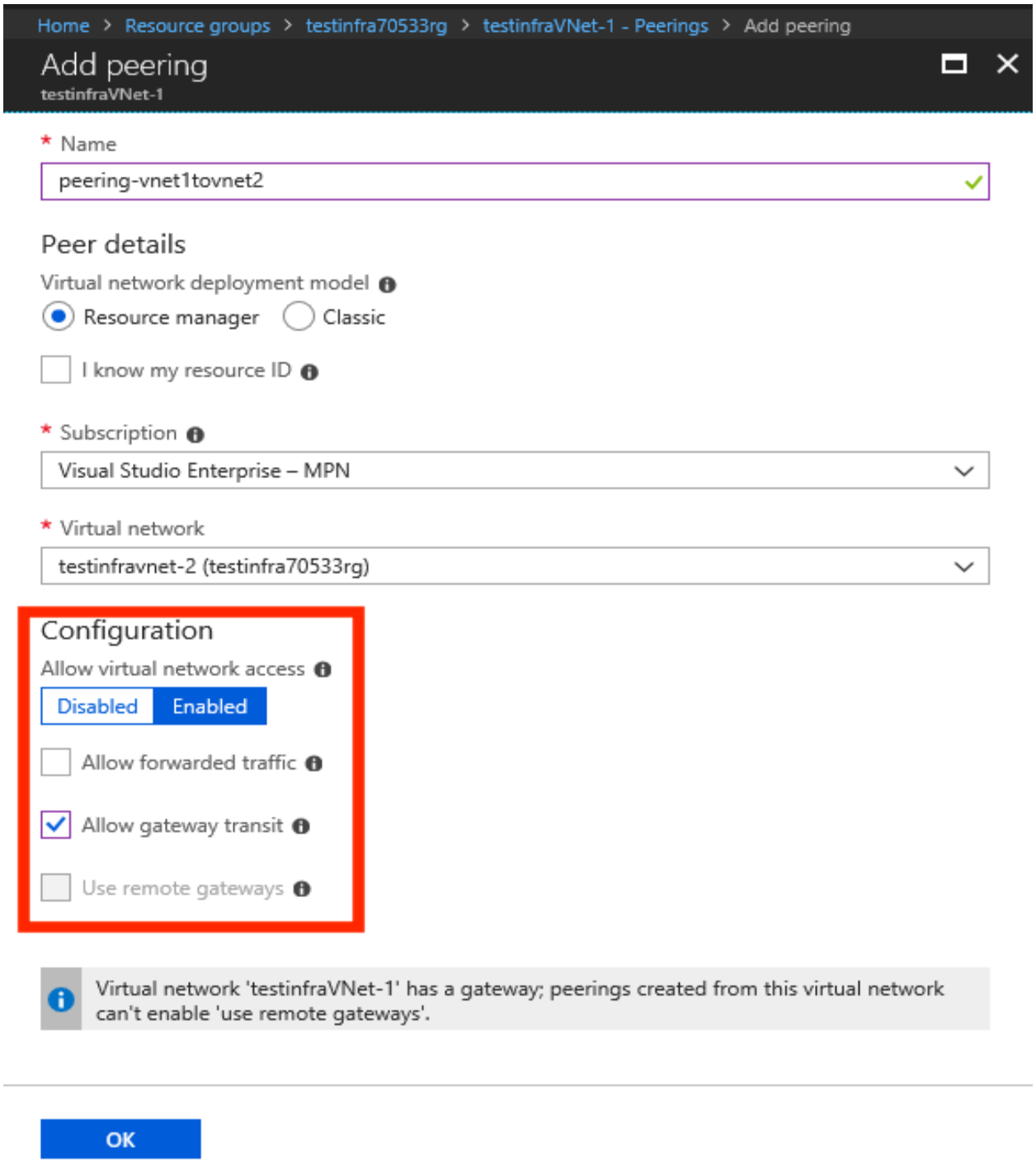
The greatest advantage of VNet peering is that Network traffic between peered virtual networks is private and routed through the Microsoft backbone infrastructure. The bandwidth and latency across the VNets are the same in the same region as if the resources were connected to the same VNet. When creating the peering, there is no downtime to resources in either virtual network. The peering is also possible to happen to peer one virtual network created through Azure Resource Manager to a virtual network created through the classic deployment model that exists in the same or different subscriptions.

Creating VNet peering via Azure Portal

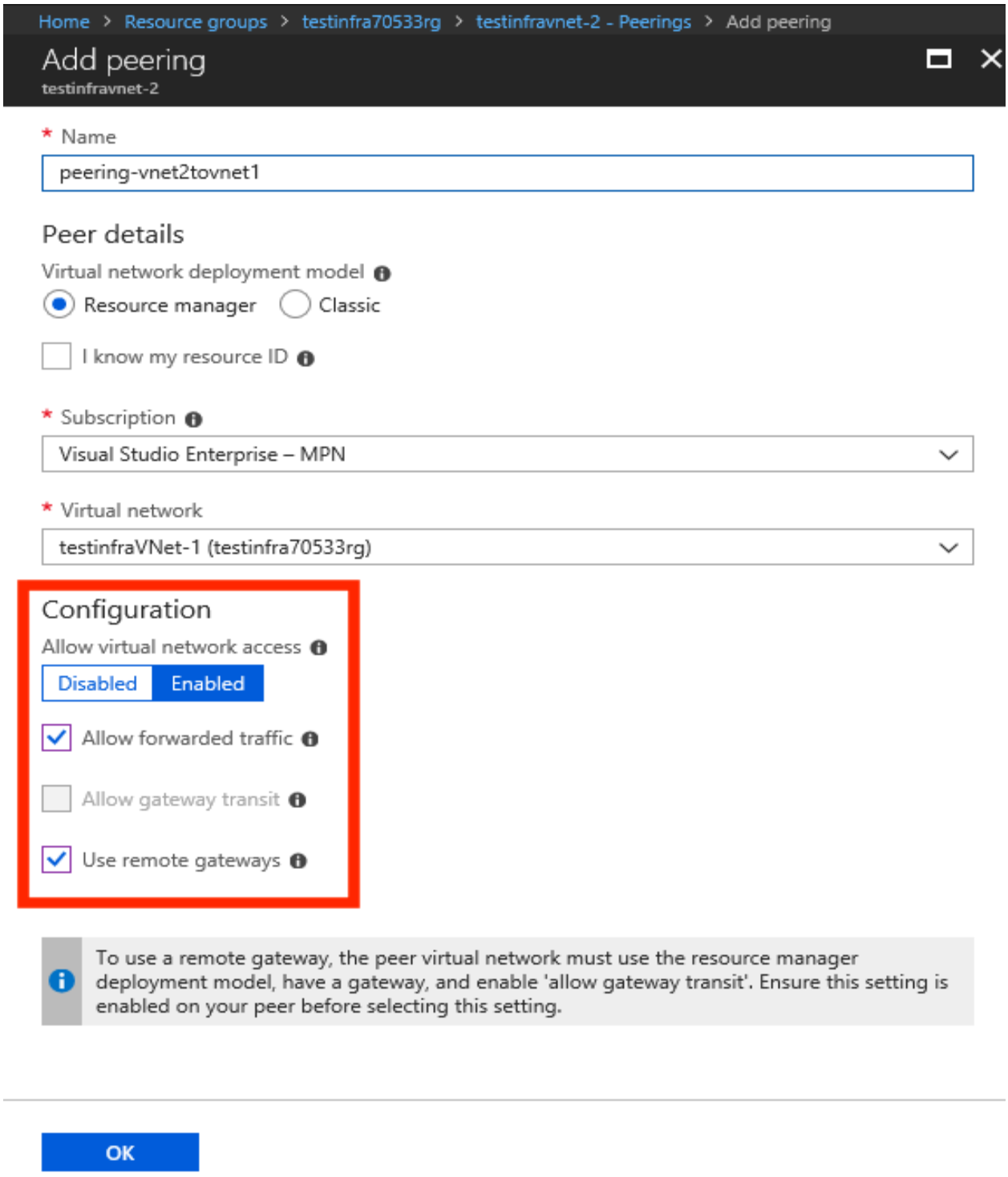
Now, let's implement a peering connection between VNet-1 and VNet-2. To add a VNet peering via Azure Portal, you can go to the **Peerings** blade of VNet-1, and then click on **Add** to add a new peering:



After clicking on Add, the Add peering page will be displayed. In this creation form, we should create the peering between VNet-1 to VNet-2, and select Enable gateway transit in the configuration section, as shown in the followingscreenshot. Make sure that we already have a virtual network gateway was in the gateway Subnet of the current VNet so that we can choose to use the remote gateway in the VNet-2 when we create the peering between VNet-2 to VNet-1:



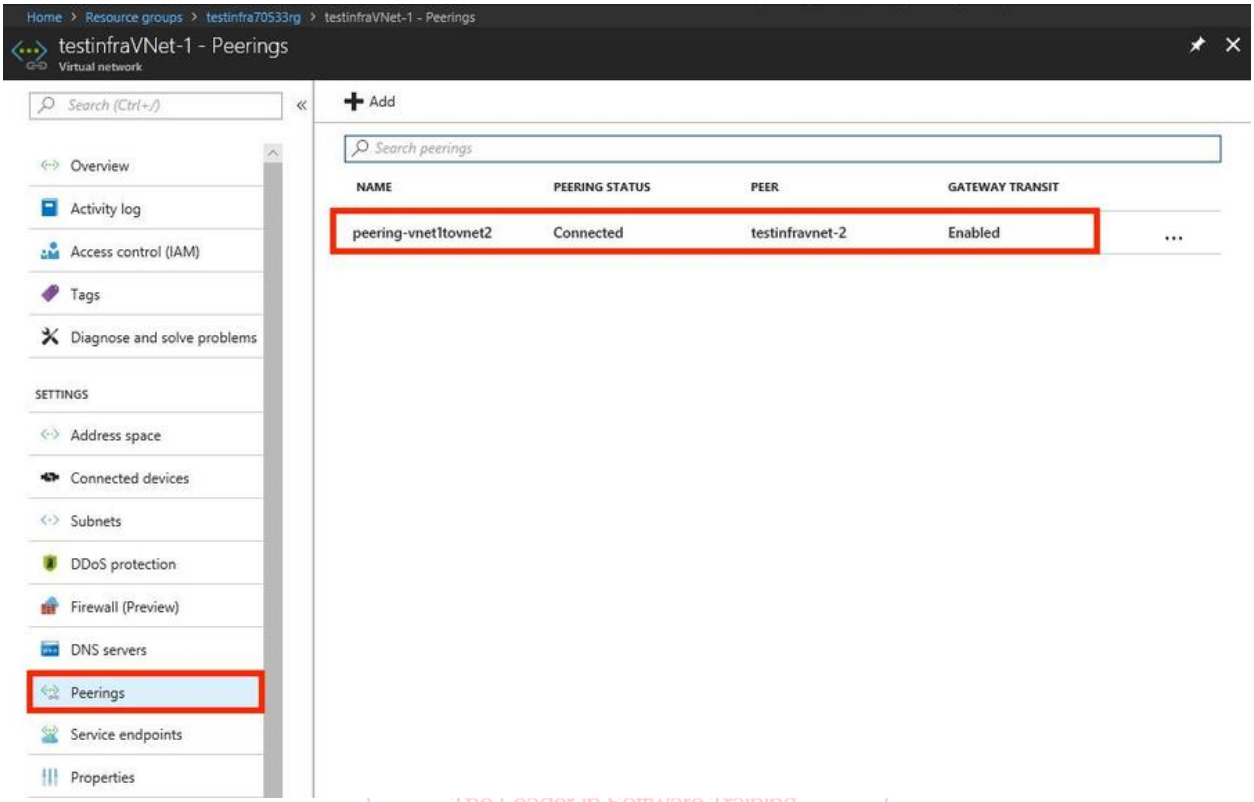
As explained in the previous step, we can go to the Peering blade for VNet - 2 and add a peering in the same way, as shown in the following screenshot. Enabling **Allow forwarded traffic** allows the traffic go to other peered VNets in a transitive way:



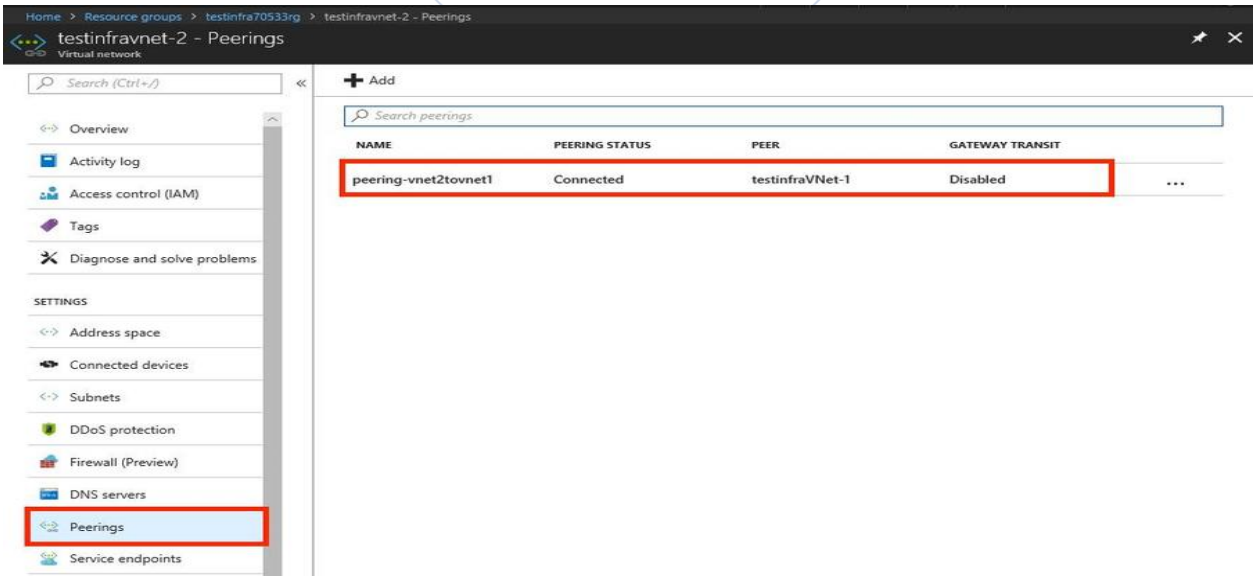
Verify the peering connection

Finally, after clicking on OK and waiting for two peering connections to be created, you can go to the Peering blade of each VNet to verify whether two VNets have been connected

successfully. If everything is going well, you'll be able to see the peering status of VNet- 1, as shown in the following screenshot:



The peering status of VNet-2 is shown in the following screenshot:



ExpressRoute

ExpressRoute is an Azure service that lets you create private dedicated connections that do not go over the public internet between Microsoft Clouds such as **Microsoft Azure**, **Microsoft Office 365**, **Microsoft Dynamics 365**, and the organization's IT environment.

To connect an on-premise infrastructure to Microsoft Cloud, users should order an ExpressRoute circuit through a connectivity provider. In Azure, a single ExpressRoute circuit represents a logical connection; it is also possible to order multiple ExpressRoute circuits and each connection can be in the same or different regions, and can be connected to your premises through different connectivity providers.

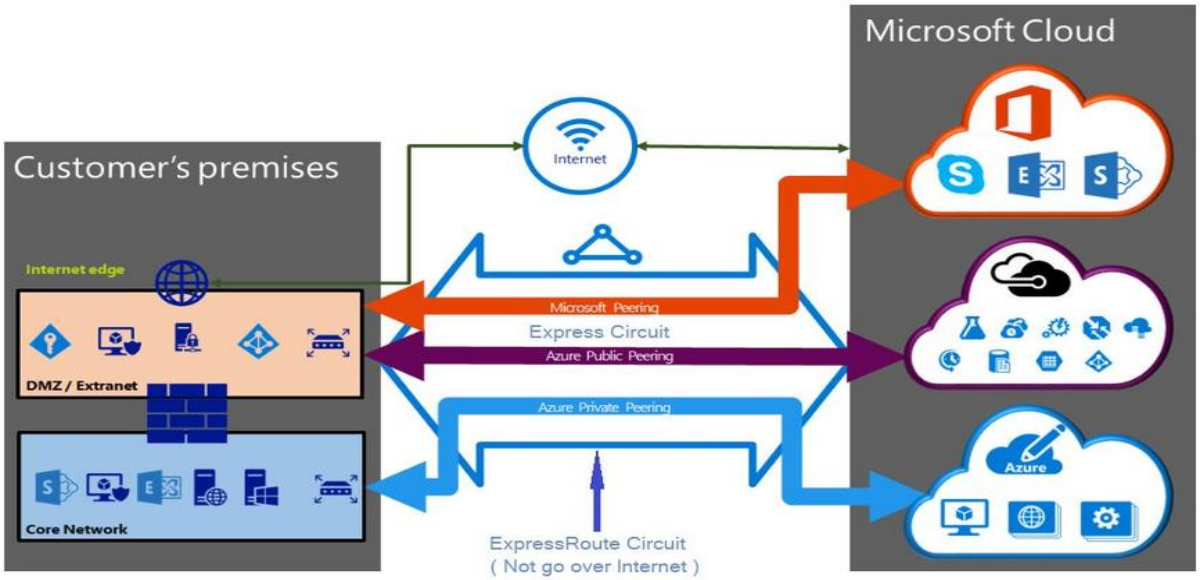
The following are the three connectivity models in Azure:

- Co-located at a cloud exchange
- Point-to-point Ethernet connections
- Any-to-any (IPVPN) networks

An ExpressRoute circuit has the following routing domains:

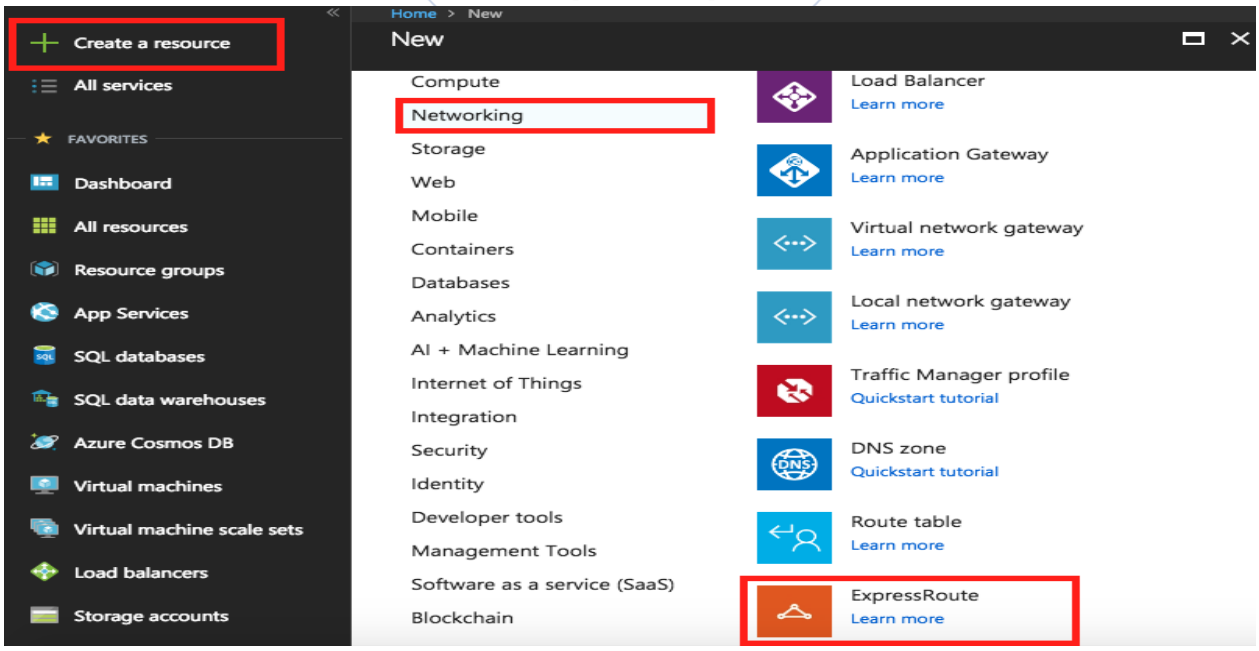
- Azure public peering
- Azure private peering
- Microsoft peering

The following schema shows the connectivity between on-premise and Microsoft Cloud through multiple routing domains:

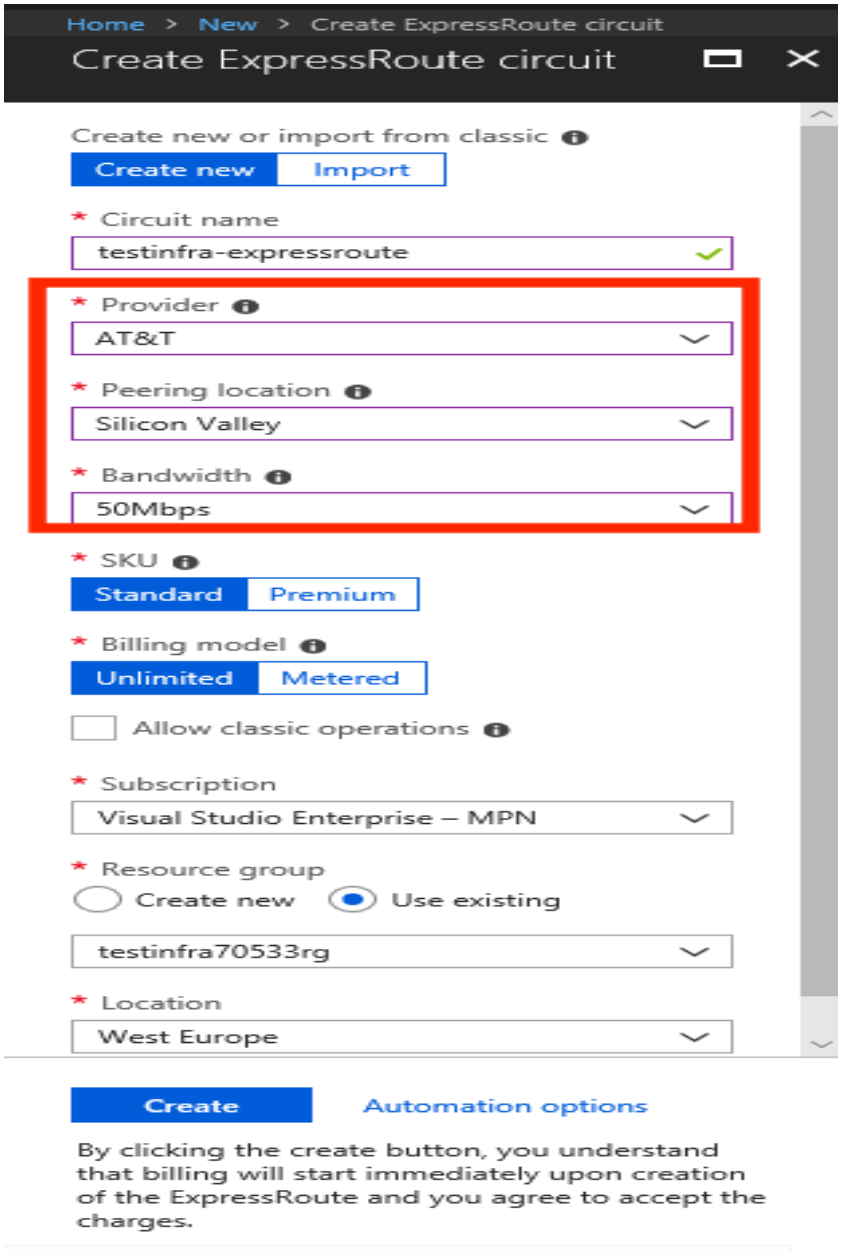


ExpressRoute circuit connecting on-premise and Microsoft Cloud (https://github.com/MicrosoftDocs/azure-docs/blob/master/articles/expressroute/expressroute-circuit-peerings.md)

To create an **ExpressRoute** circuit, you can go to the Azure Portal and click on **Create a resource**; you'll find **ExpressRoute** in the **Networking** category, as shown in the following screenshot:



After clicking on ExpressRoute, you'll see the **Create ExpressRoute circuit** page where there is a creation form. You should choose a connectivity provider and a peering location, which is the physical location from which you are peering with Microsoft. You can also choose the available bandwidth based on your previous selection. If everything is okay, you can click on **Create**; the deployment will last for a couple minutes:



After creating an ExpressRoute circuit successfully, the next step is to create the routing configuration, and finally to link a VNet to an ExpressRoute circuit. For more information

on how to do that, you can check the following links to know different ways to achieve your objective:

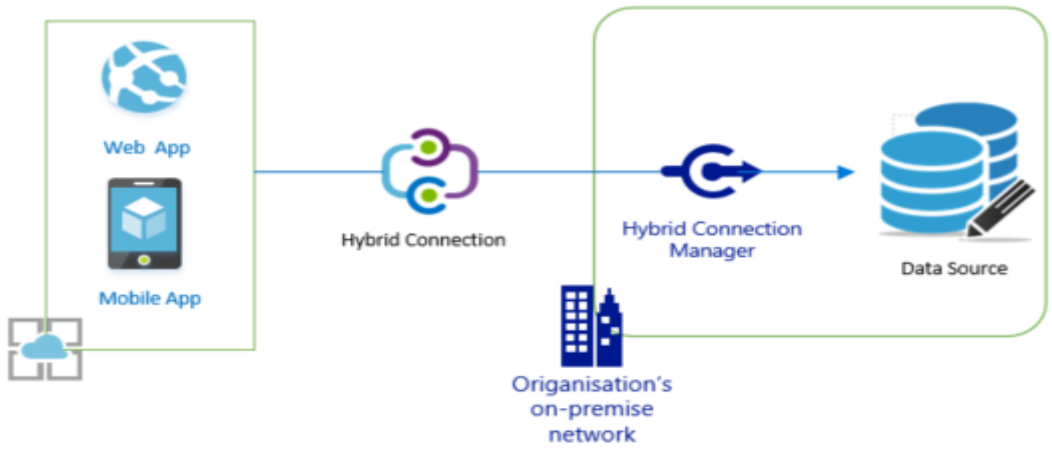
- Via Azure Portal: https://docs.microsoft.com/en-us/azure/expressroute/expressroute-howto-routing-portal-resource-manager
- Via Azure PowerShell: https://docs.microsoft.com/en-us/azure/expressroute/expressroute-howto-routing-arm
- Via Azure CLI: https://docs.microsoft.com/en-us/azure/expressroute/howto-routing-cli

For more information on how to link a VNet to a ExpressRoute circuit, you can check the different ways to do that at the following links:

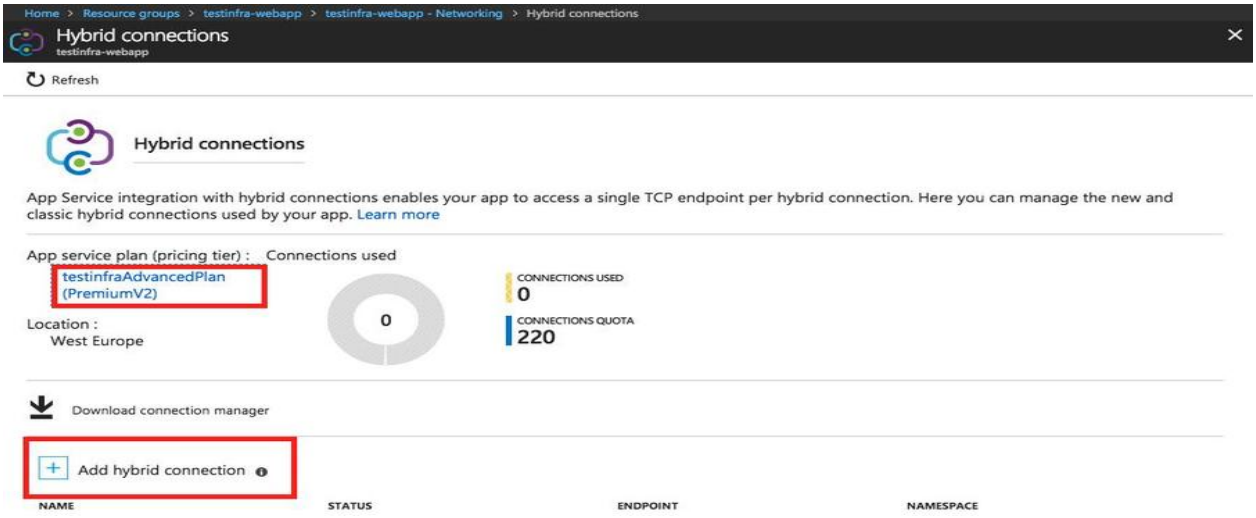
- Via Azure Portal: https://docs.microsoft.com/en-us/azure/expressroute/expressroute-howto-linkvnet-portal-resource-manager
- Via Azure PowerShell: https://docs.microsoft.com/en-us/azure/expressroute/expressroute-howto-linkvnet-arm
- Via Azure CLI: https://docs.microsoft.com/en-us/azure/expressroute/howto-linkvnet-cli

Configuring Hybrid Connections for App Service

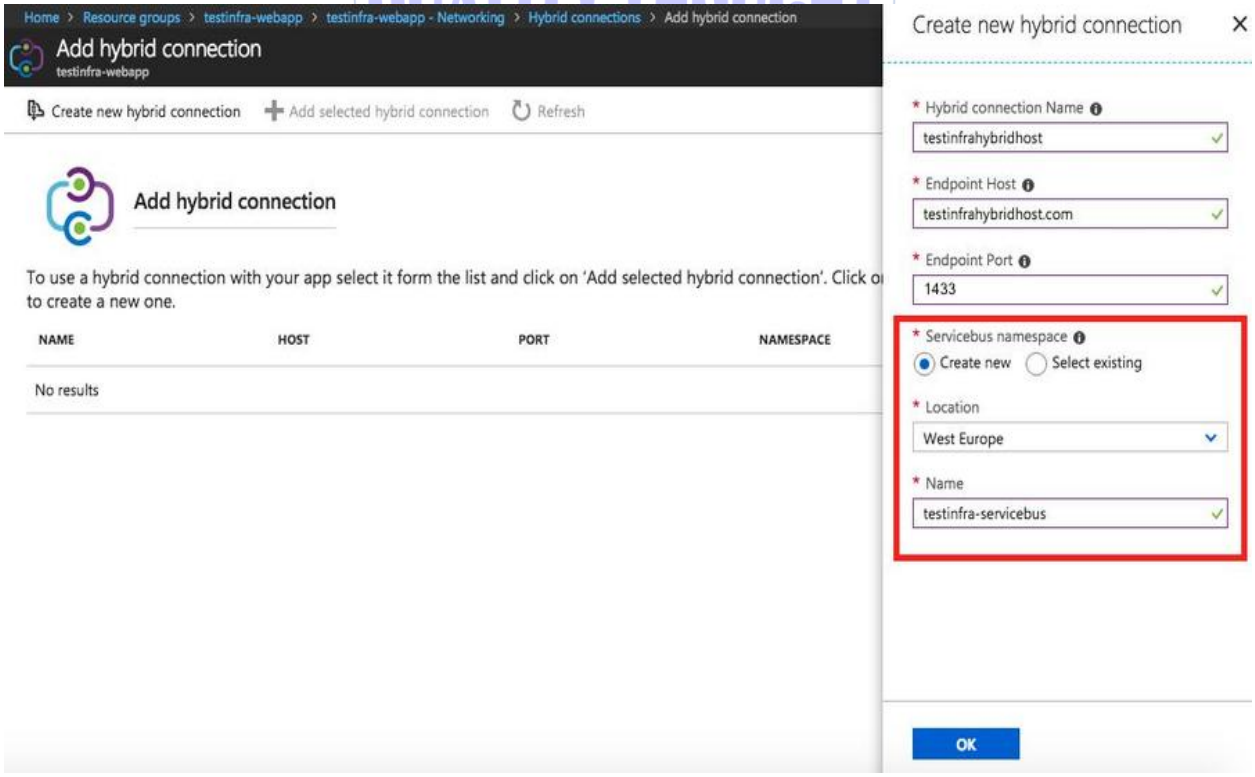
Hybrid Connections is a capability provided by Azure App Service to let web apps and mobile apps in App Service access on-premise systems and services securely. The following schema shows how Hybrid Connections works in Azure:



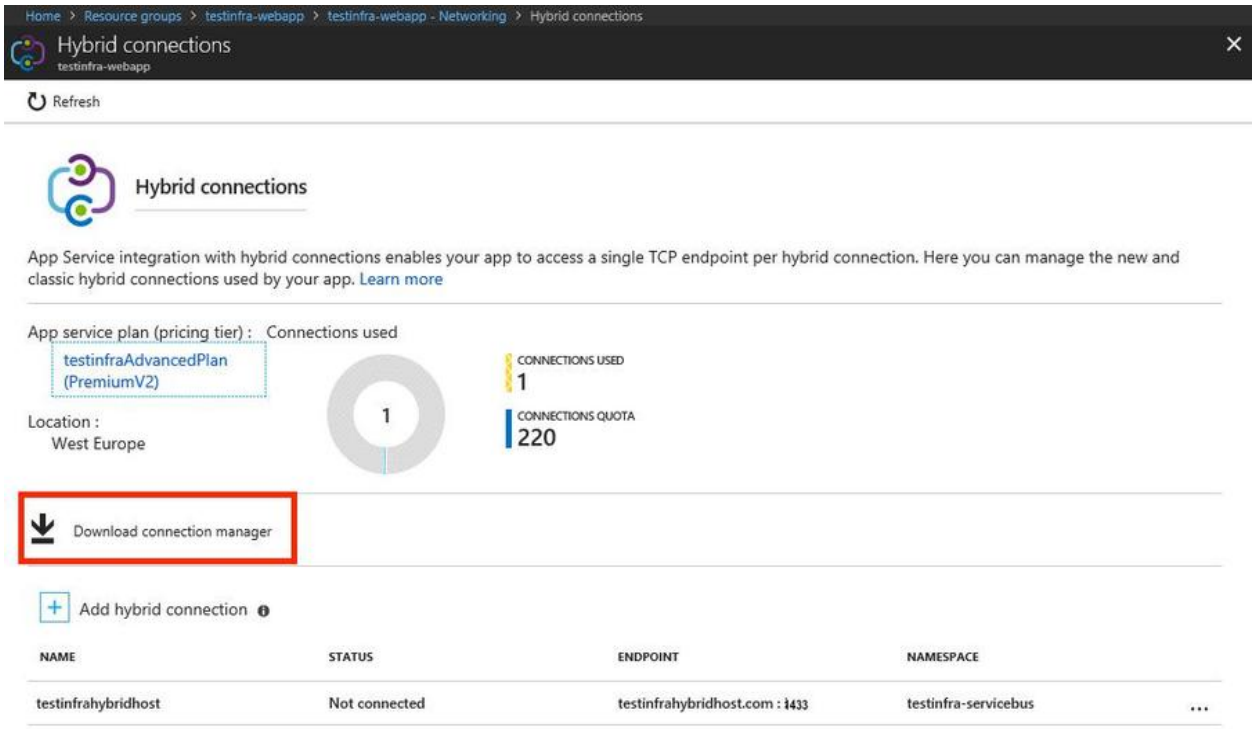
To create a hybrid connection, you can go to the Azure Portal and select your web app. Then, go to the **Networking**blade and click on **Configure your Hybrid Connection endpoints**; you will be provided with a screen as shown in the following screenshot:



Click on **Add hybrid connection**; you should fill in the endpoint and endpoint port. Each hybrid connection is attached to a service bus namespace in one Azure region. Microsoft recommends you use an existing service bus namespace in the same region, or create a new service bus namespace as the target web app, to reduce network latency, as shown in the following screenshot:



After a hybrid connection has been created successfully, you can go to the **Networking** blade and then click on Hybrid Connections. You should download the Hybrid connection Manager and install it on the on-premise resource so that the Hybrid Connections are established:



Configuring multi-region applications with Azure Traffic Manager

Traffic Manager is a DNS-level load-balancing solution that is included within Azure. It uses the following four traffic-routing methods to direct client requests to the most suitable service endpoint:

- **Priority**: This is a method to distribute traffic to the primary location but it will direct the traffic to a secondary location in the case of failure of the primary region.
- **Weighted**: This is a method to distribute traffic across a set of endpoints according to weights. It can be evenly distributed as well depending on the user's configuration.
- **Performance**: This is a method to distribute traffic depending on the location with the lowest network latency.
- **Geographic**: This is a method to distribute traffic depending on which geographic location the DNS query originates from.

In Azure, all the Traffic Manager profiles perform the following two features:

- Monitoring the health of endpoints

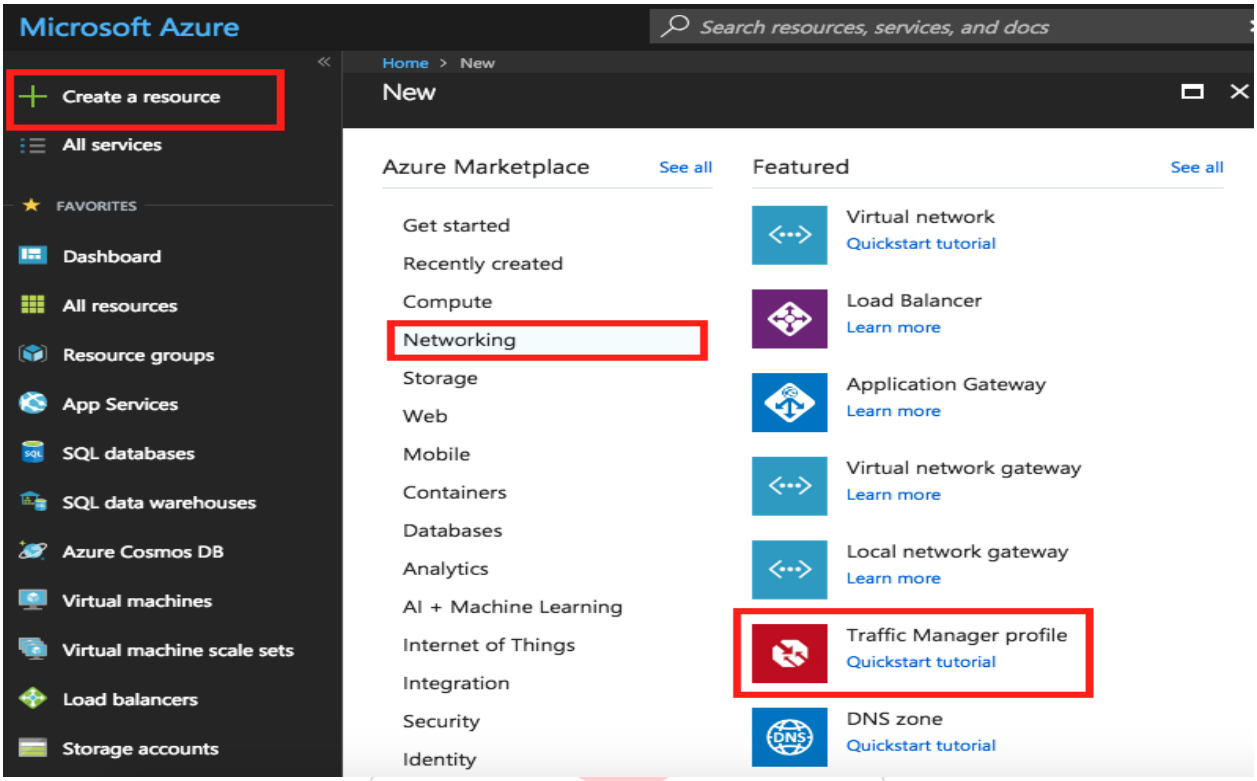
- Handling endpoint failover automatically

In the multi-region scenario, the Traffic Manager can be configured with priority methods and it routes incoming requests to the primary region and fails over to the secondary region in the event a failure occurs in the primary region; for example, the application running in the primary region becomes unavailable.

Another settings is about Health probes. Traffic Manager uses an HTTP probe to monitor the availability of configured endpoint linked to the Traffic Manager profile. The main responsibility of a health probe is to check the availability of each region. It can send a request to a specified URL path via the defined protocol (HTTP or HTTPS) and port to check for uptime and determine whether the instances in this region (current endpoint) are healthy by getting a 200 response or unhealthy if they get a non-200 response within a determined period of time (failing these, it will throw a timeout). After several retries, if the requests still fail, the Traffic Manager will consider the current endpoint as a failure and will fail over to the other endpoint.

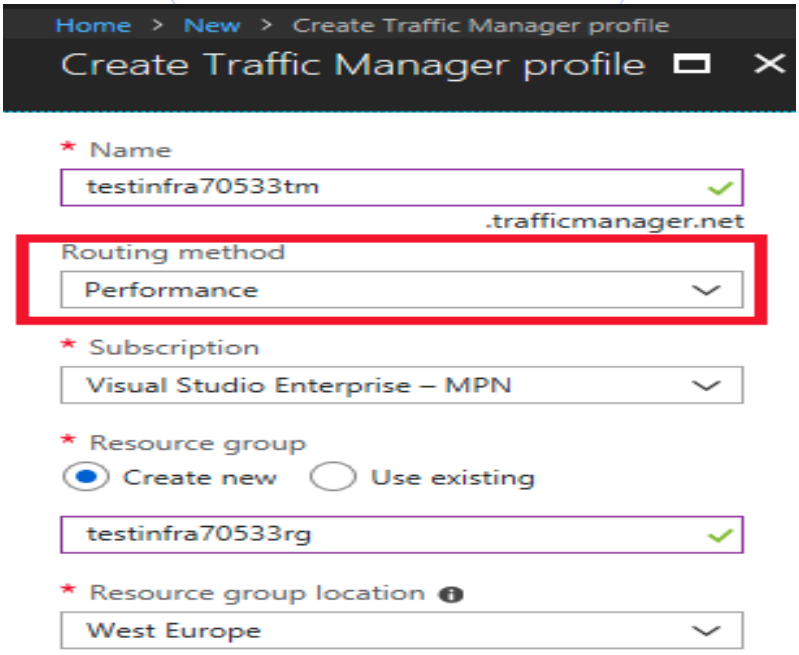
Creating a Traffic Manager profile

You can create a new Traffic Manager by clicking on **Create a resource** via the Azure Portal. Then, in the **Networking** category, you'll find **Traffic Manager profile**, as shown in the following screenshot:



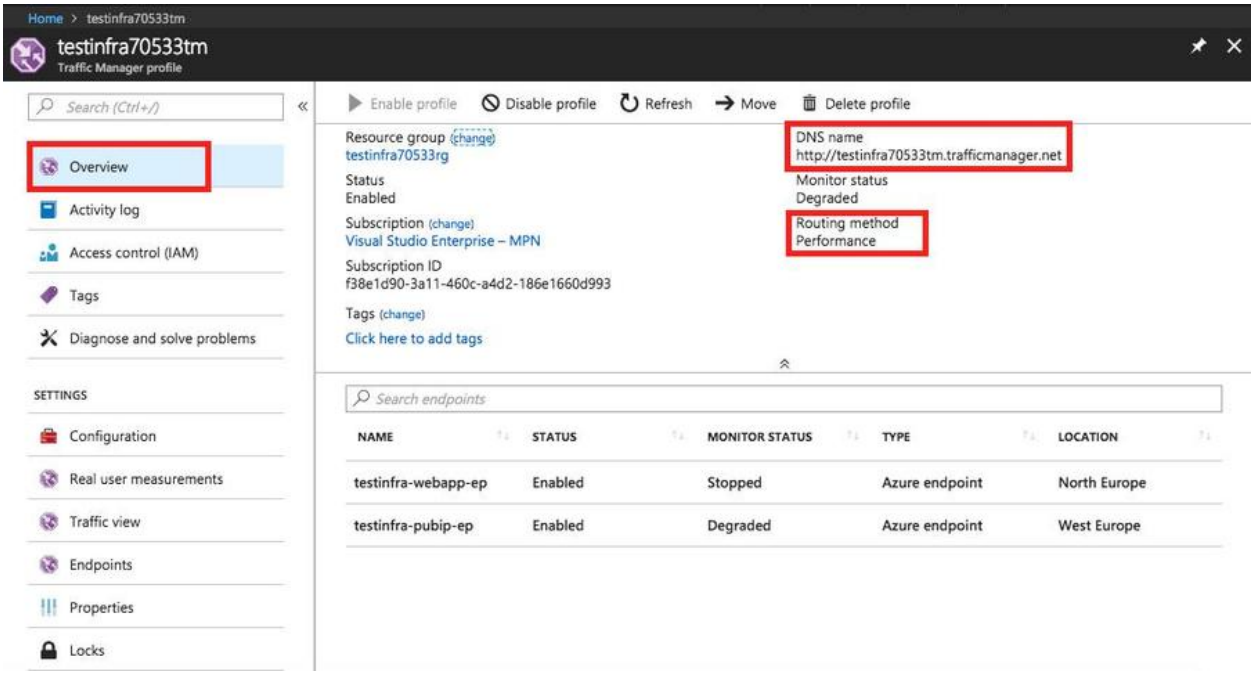
Creating Traffic Manager profile via Azure Portal

In the basic information form, you can enter the name of the Traffic Manager profile and choose a routing method for the Traffic Manager:



After clicking on **Create**, the **Traffic Manager profile** will be created in a few minutes. After creating the a Traffic Manager profile, you can go to the **Overview** blade and share information such as routing methods you have chosen previously. The URL of a Traffic Manager profile named testinfra70533 looks likes this: http://testinfra70533tm.trafficmanager.net.

The following screenshot shows all the available information that you can see via the **Overview** blade:



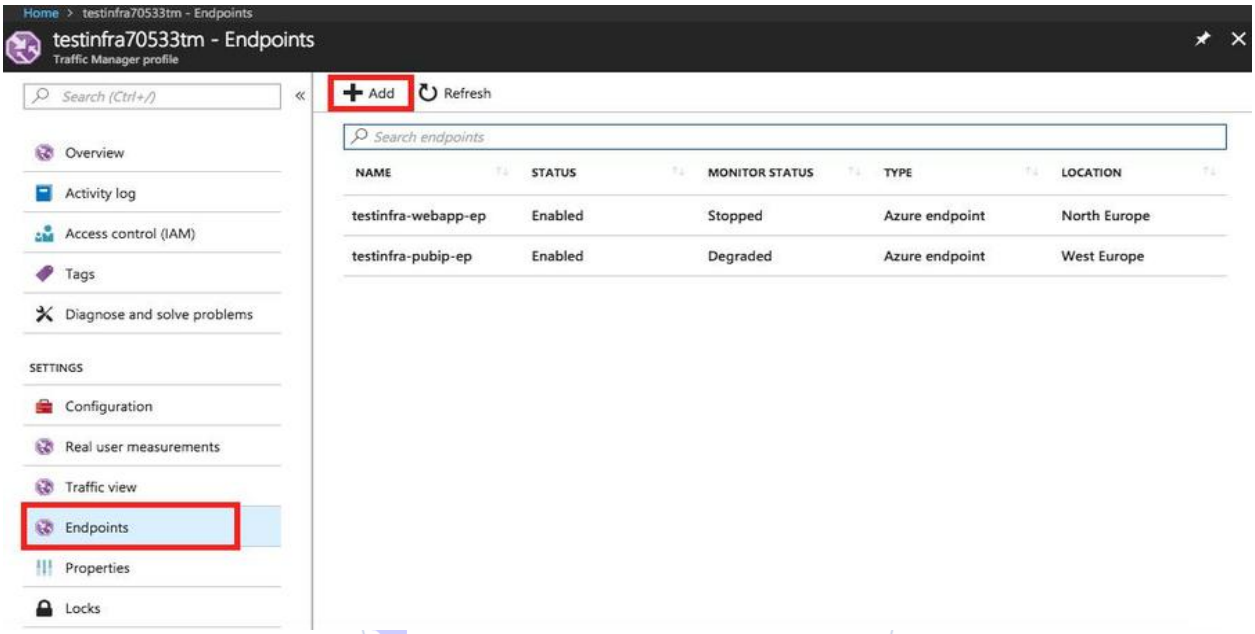
Adding endpoints to the Traffic Manager profile

After creating a Traffic Manager profile, you'll be able to add endpoints to the Traffic Manager. Currently, there are the following three types of endpoint supported by Traffic Manager:

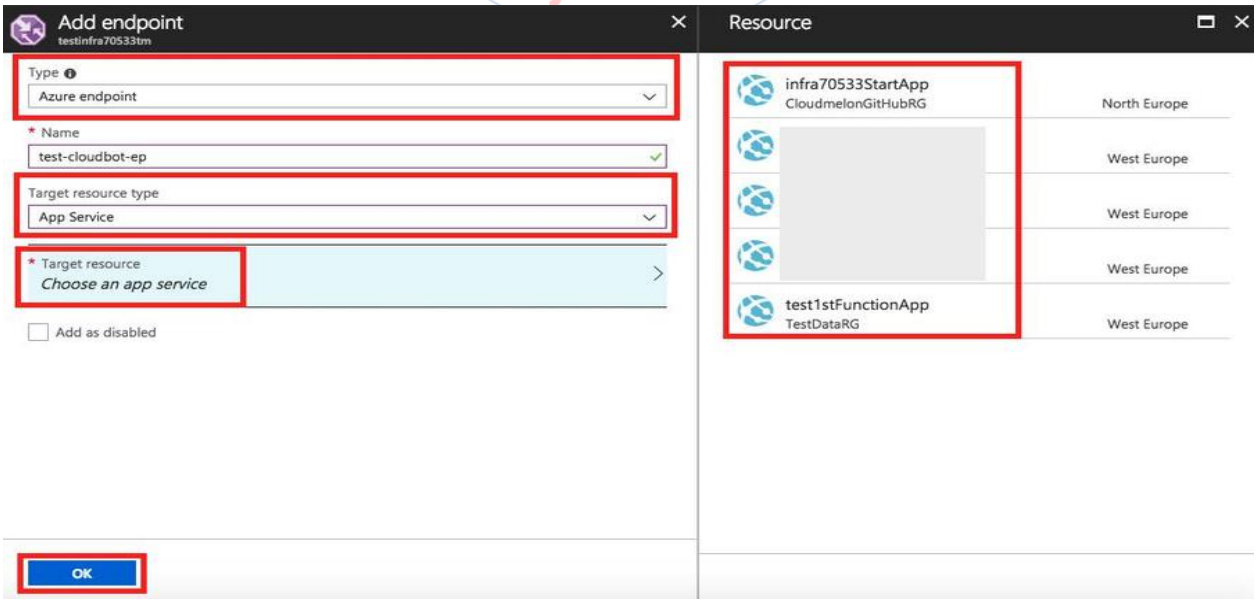
- **Azure endpoints**: These are used for different IaaS services, PaaS services, or Public IP addresses within Azure.
- **External endpoints**: These are used for different services hosted on-premise or by other hosting providers to resume any fully-qualified domain name (FQDN) outside Azure.

- **Nested endpoints**: This is an endpoint type combining a parent endpoint and child endpoint in the same scenario, which is a more complex deployment scenarios.

While working with Azure Traffic Manager, it is possible to combine different types of endpoint in a single Traffic Manager profile as shown in the following screenshot. Additionally, each profile can contain multiple endpoint types. You can add an endpoint by clicking on **Add**:



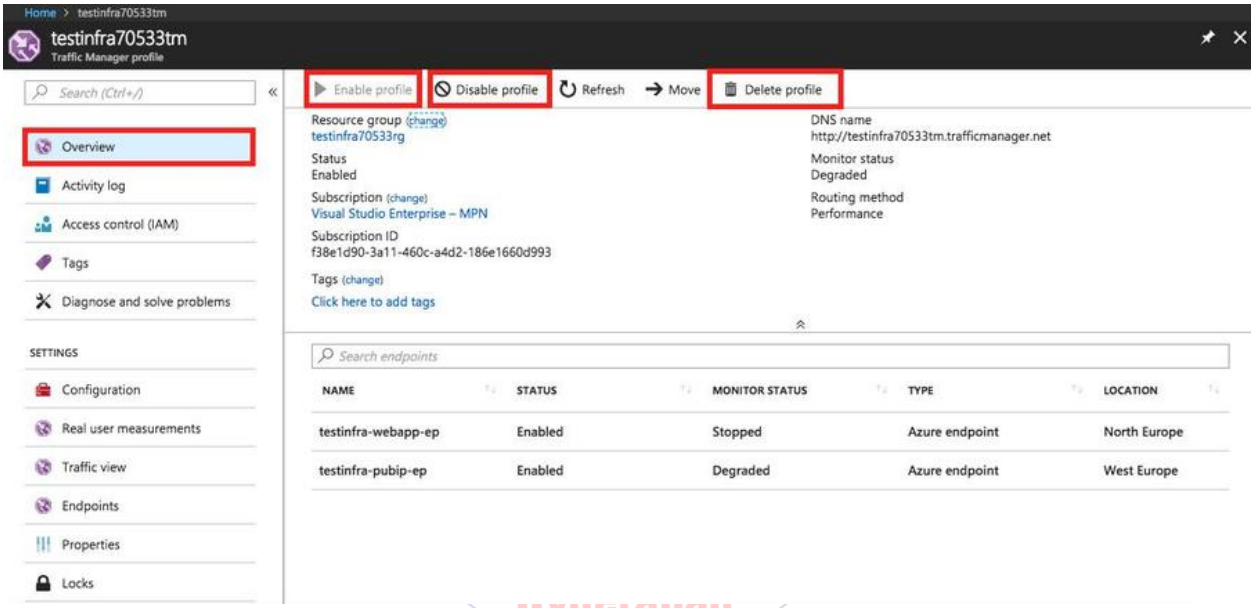
For example, while creating an Azure endpoint, you can choose a type such as cloud service, public IP address, or App Service, and then specify the target source, as shown in the following screenshot:



After you click on **OK**, within a few seconds an endpoint will be added in the Traffic Manager profile.

Managing Traffic Manager profiles

To manage created Traffic Manager profiles, let's go back to the **Overview** blade. Here you can select **Enable profile**, **Disable profile**, or **Delete profile** for current Traffic Manager profile:

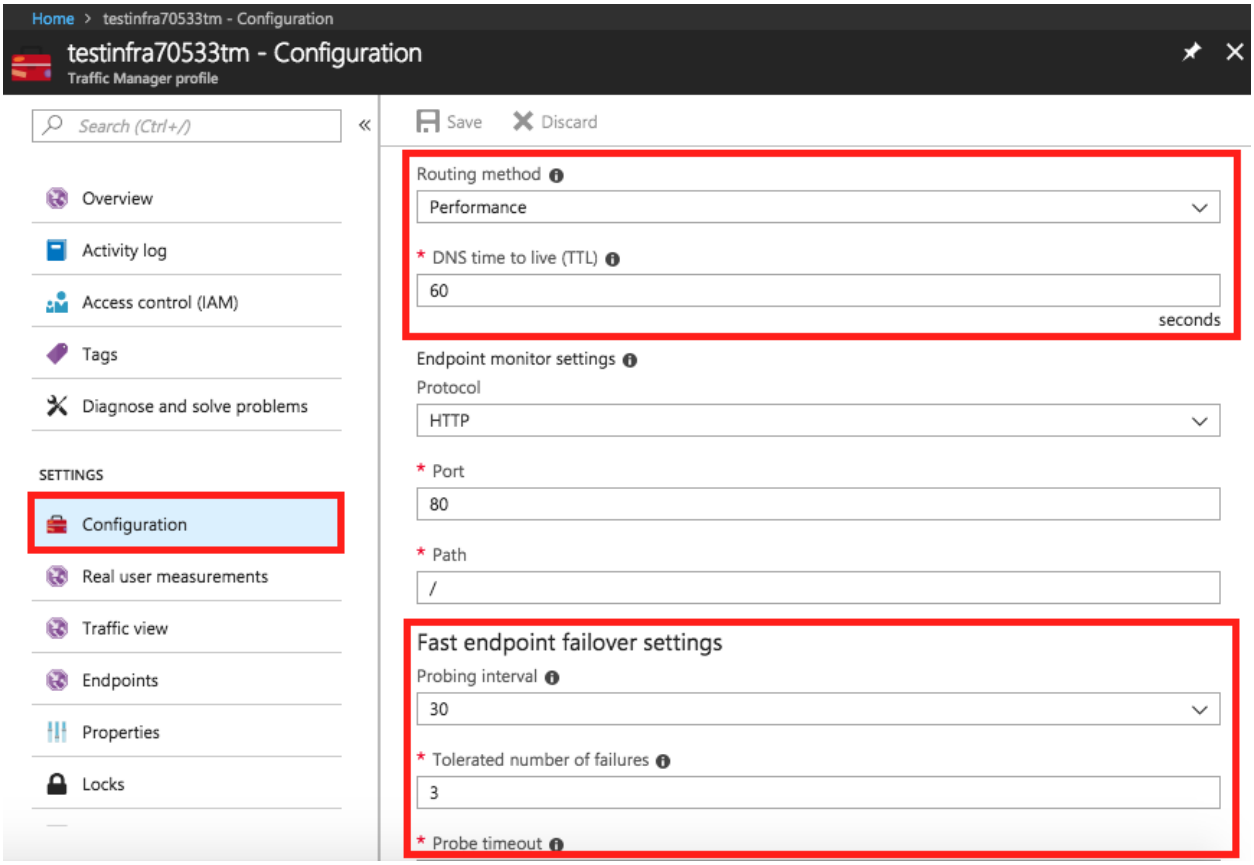


If you go to the **Configuration** blade in the Traffic Manager, you can also modify the configured routing method while creating the Traffic Manager. You can also define the value of the DNS **time to live** (**TTL**) as the live time of the client's local caching name server. After this period, the local caching name server will query the Traffic Manager system to update DNS entries.

Additionally, if you want to improve application resilience using a multi-region configuration with Traffic Manager, some interesting settings in the Fast endpoint failover settings section will definitely help you:

- **Probing interval**: This represents the time interval between endpoint health probes.
- **Tolerated number of failures**: This defines the number of health probe failures tolerated before an endpoint failure is triggered. It can be any number from 0 to 9.
- **Probe timeout**: This defines the time required before an endpoint health probe times out. This value must start from 5 and should be smaller than the probing interval value.

The following screenshot shows which settings are included in the **Configuration** blade for the Traffic Managerprofile:

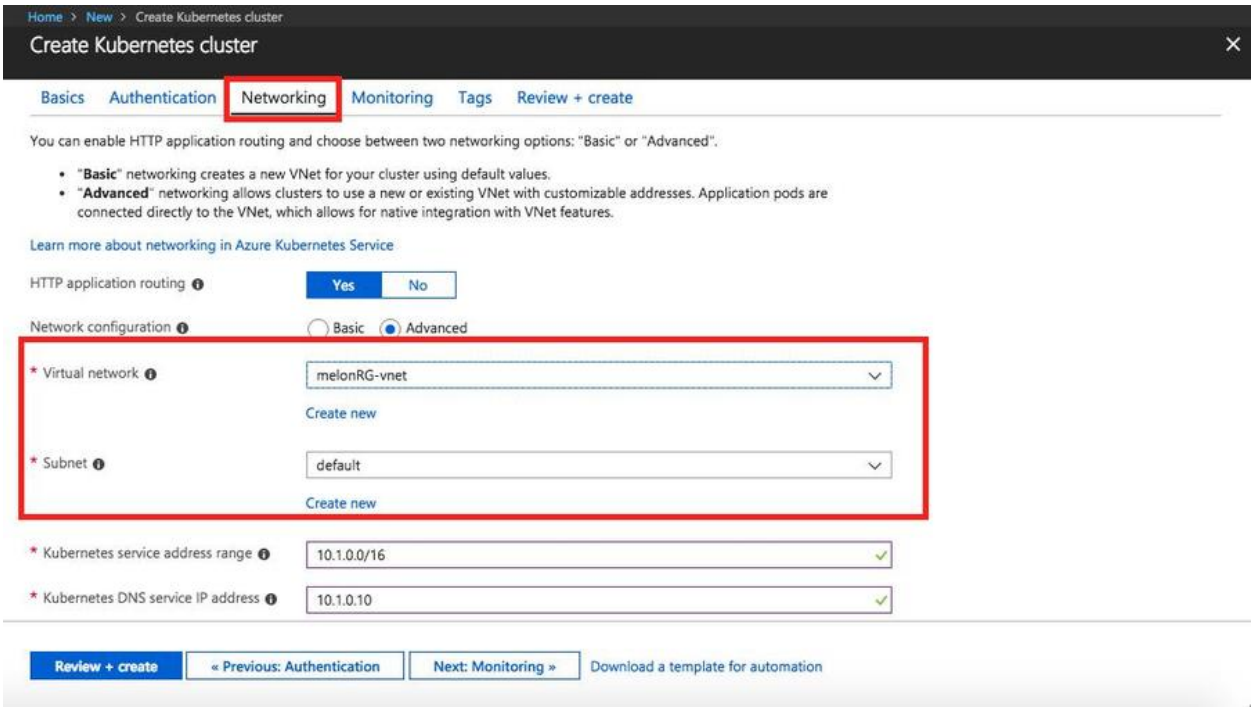


Integrating Azure services with an Azure virtual network

For most customers, a top concern when migrating their data and services to the cloud is how to restrict access control at the network level. Microsoft Azure provides a way to allow private access from instances of an Azure service deployed in the virtual network, by integrating Azure services with Azure Virtual Network. By definition, to integrate an Azure service means to guarantee communication between Azure services in the following two ways:

- Deploying dedicated instances of the Azure service into a **virtual network** while creation so that these services can be guaranteed private access within the virtual network
- Extending a virtual network to the Azure service through **Azure Network service endpoints** to allow access to individual service instances

There is a wide range of Azure services that can be deployed in Azure virtual networks, such as Azure Virtual machines, Virtual machine scale sets, Azure Kubernetes Services (AKS), and Azure Batch. A service such as AKS can be integrated into a virtual network. You can configure it to an existing VNet or create a new VNet for it during the first creation, as shown in the following screenshot:



Configuring Azure Virtual Network while creating a new AKS cluster

It is possible to deploy an Azure service into a subnet within an Azure virtual network that has also other Azure-integrated Azure Services and secure that service in the subnet with NSG.

Additionally, to reach out to these Azure services from on-premise, we can use different cross-premise connectivity options, and use a public IP address from the internet.

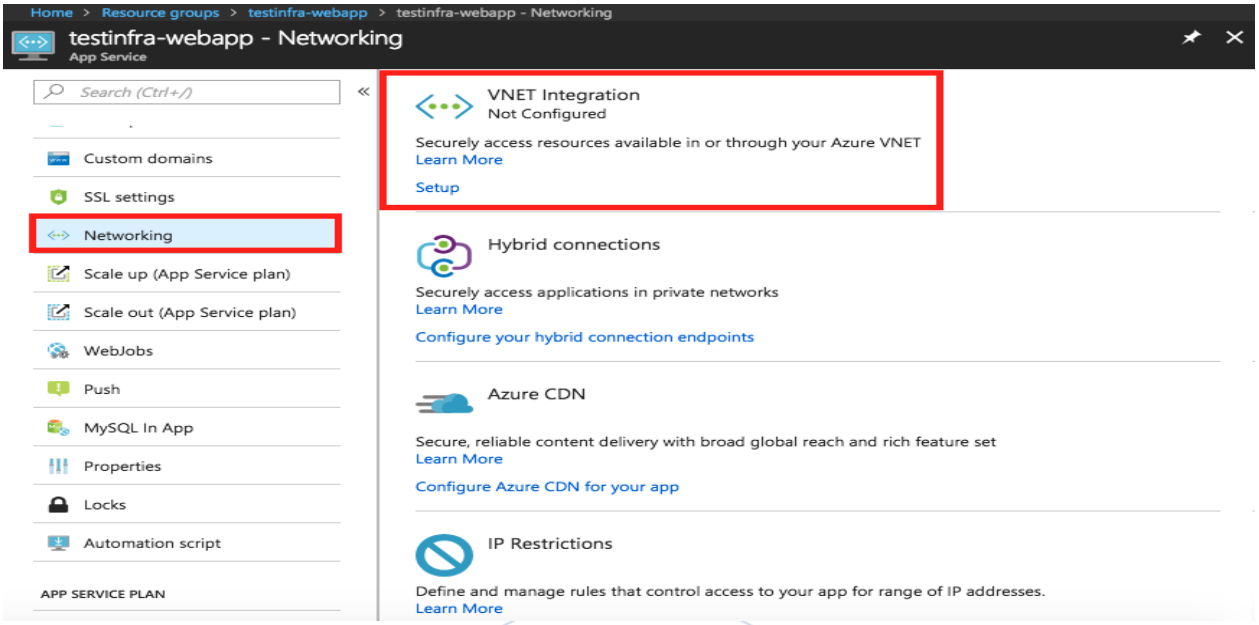
Restrict network access to PaaS resources using a service endpoint

Virtual network service endpoints is an amazing option in Azure to limit network access to some Azure service resources to a virtual network subnet in Azure. It allows you to use the private address space of the virtual network to access Azure services over a direct connection. Azure guarantees your traffic from the VNet to the Azure service always remains on the Microsoft Azure backbone network. This option is available to PaaS Azure services such as Azure Storage and Azure SQL Database for all Azure regions. For more information about this feature, you can go to the following link: https://azure.microsoft.com/en-us/updates/?product=virtual-network.

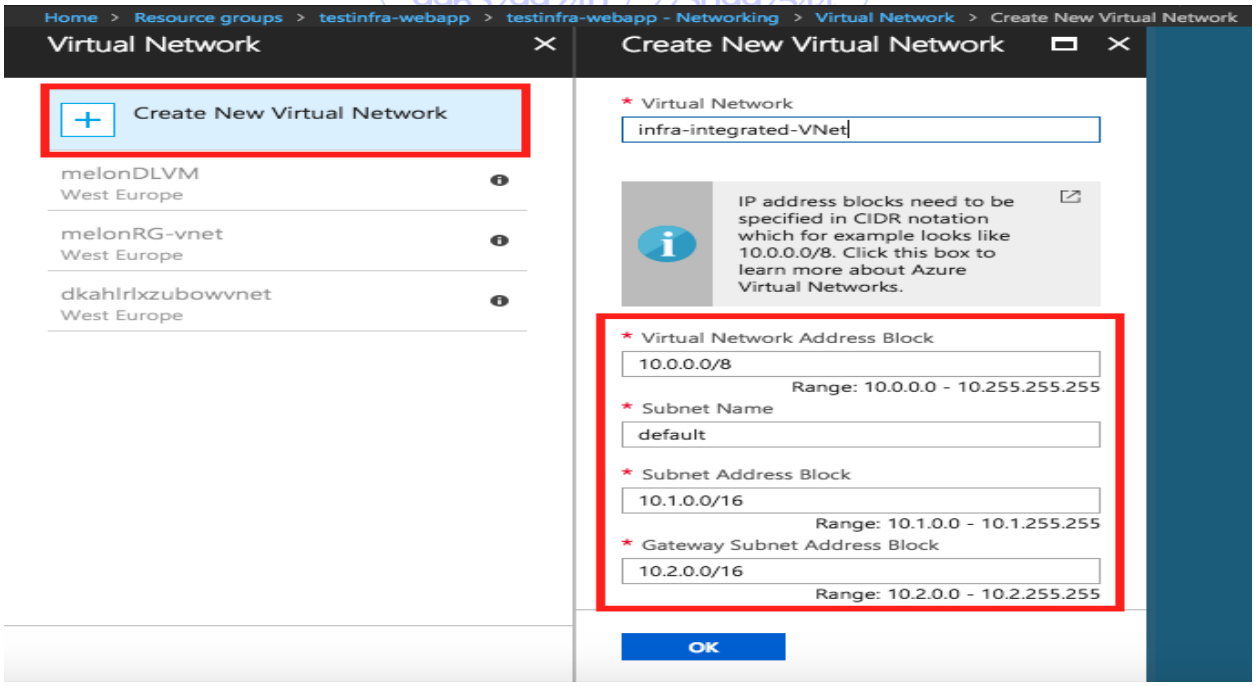
Integrating a web app in App Service with an Azure virtual network

VNet integration allows a web app hosted in the App Service plan to have access to resources in an Azure virtual network; users can hosted multiple Azure resources in an Azure virtual network with the control of access from the internet or on-premise using a variety of VPN connectivities.

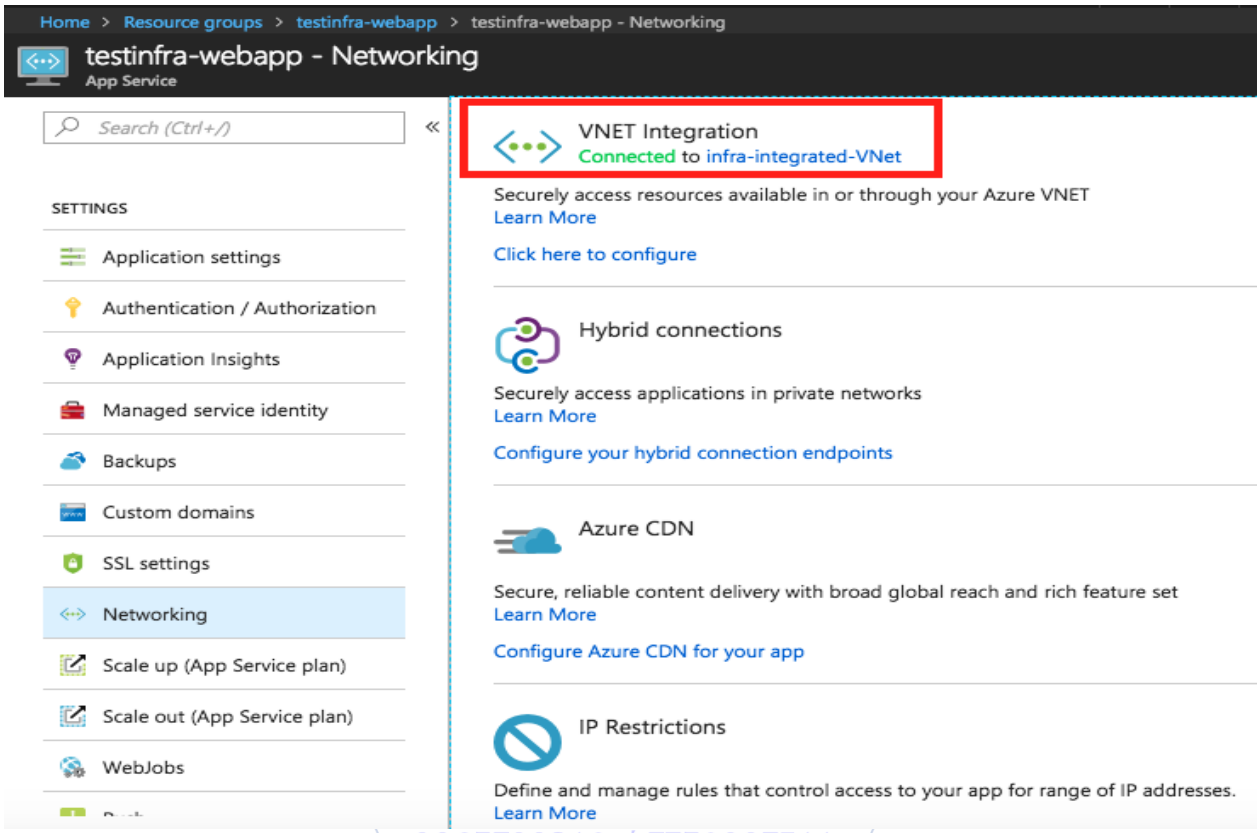
For a web app that has been created, you can use the VNet Integration options to connect it to a new or existing Azure virtual network. You can go to the **Networking** blade for the web app and choose the VNet Integration option, as shown in the following screenshot:



Note that this feature is only available in the Standard and, Premium, and Isolated Pricing Plans. So if you're using the Standard or Premium plans, you can click on Setup to start to configure a VNet for your web app. If you want to integrate your web app with an existing VNet, the VNet would have a point-to-site VPN enabled with a Dynamic routing gateway so that it can be connected to an app; in the other case that P2S VPN with a static routing gateway or without any gateway, you cannot integrate it with your web app. In this case, you should create a new VNet. As shown in the following screenshot, you can create a new virtual network for your web app by specifying the address block and subnet for this VNet:

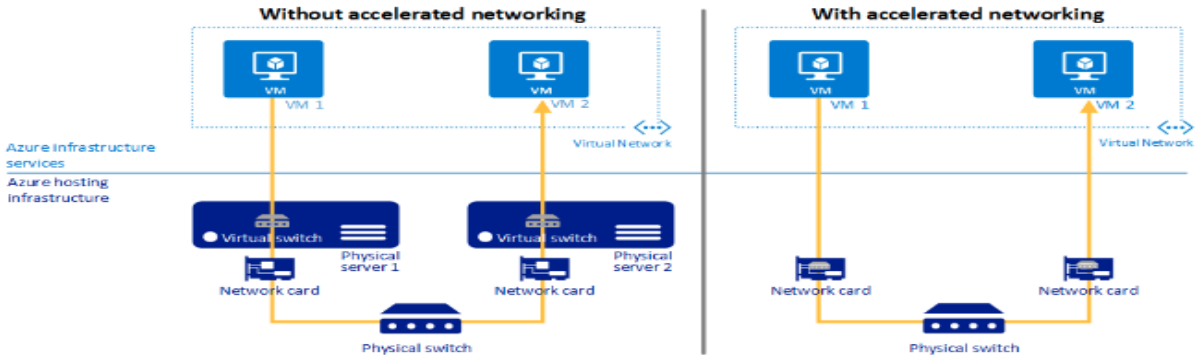


The deployment usually takes a couple minutes. If your web app has been integrated with the newly created VNet, you'll see something similar to the following screenshot, while checking the **Networking** blade of your web app:



Configuring accelerated networking to improve your networking performance

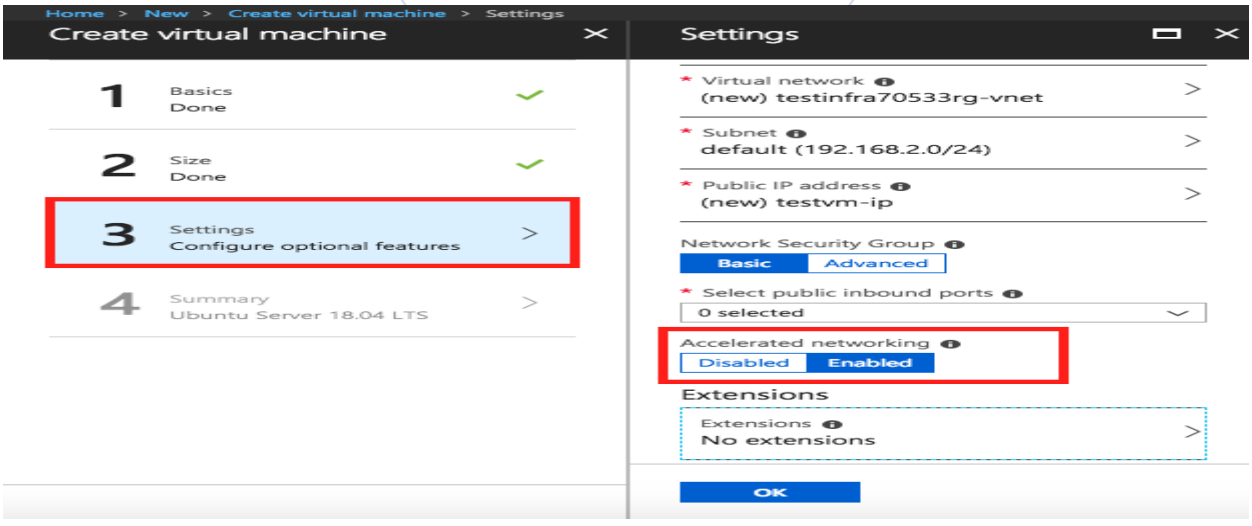
Accelerated networking is a pretty cool, free feature for Azure VMs and VM Scale Sets. It enables **single root I/O virtualization** (SR-IOV) for a VM or VMSS, which will greatly improve its networking performance. The reason why this feature is pretty cool is because it prevents all the inbound and outbound traffic from traversing the host and the virtual switch. With accelerated networking, network traffic arrives at the **Azure Network interface** (**NIC**) of the VMs and instances in VM Scale Sets then forwarded directly to the VM because all the network policies that the virtual switch applies are now offloaded and applied in hardware, as shown in the following screenshot:



The benefits from accelerated networking are reduced latency and CPU utilization, which significantly improves network performance.

Accelerated networking is supported on most general-purpose and compute-optimized such as **D/DSv2**, **F/Fs**, **D/DSv3**, **E/ESv3**, **Fsv2**, and **Ms/Mms** for Azure virtual machines and instances in VM scale sets.

To use this free feature, you can enable it while creating a new Azure VM only if you choose the VM size support this feature previously. If that is the case, you can enable this feature via the Azure Portal as shown in the following screenshot:



Managing Azure Identities

Azure AD is a cloud-based identity service provided by Microsoft Azure that allows you to secure access to cloud-based and on-premises applications and services.

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

- Implementing and managing Azure AD
- Integrating applications with **Azure Active Directory** (**Azure AD**)
- Integrating **Active Directory Domain Services** (**ADDS**)
- Implementing **Active Directory Federation Services** (**ADFS**)

Implementing and managing Azure Active Directory (Azure AD)

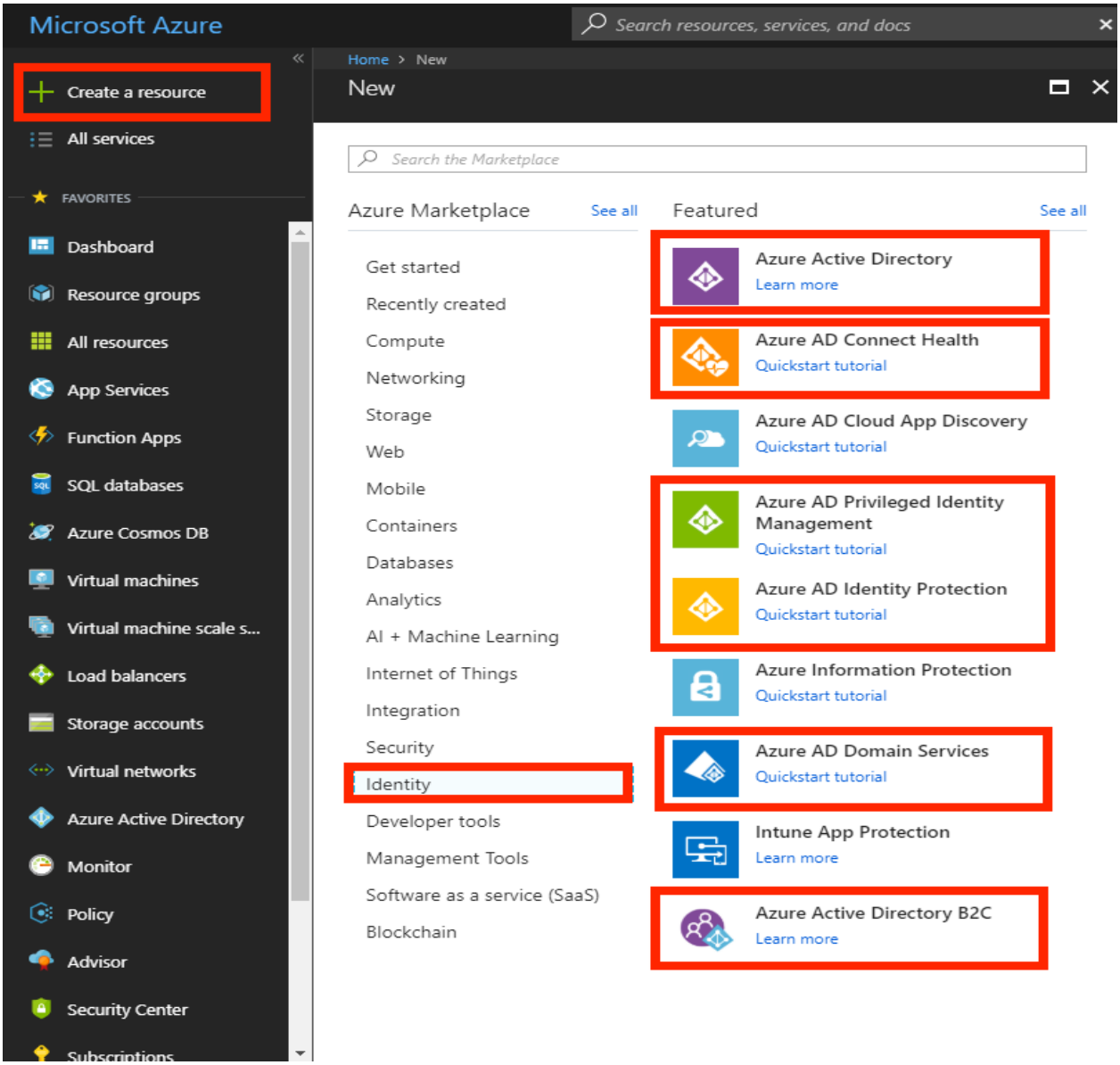
Azure AD is a cloud-based multi-tenant identity solution that provides the following capabilities:

- Managing users, groups, and roles within organizations
- Providing multifactor authentication for both on-premises and cloud-resident resources
- Providing control access functionality control to enterprise-level applications
- Providing identity protection and managing identity devices
- Integrating **Single Sign-On** (**SSO**) functionality with a cloud-based **Software as a Service** (**SaaS**) applications and on-premises apps

Azure AD offers a rich, standards-based platform with many capabilities to help enterprises and organizations build cloud-based IDaaS solution in Azure.

Managing identities via Azure Active Directory admin center

If you go to the Azure Portal, click on **Create a resource,** and then on the **Identity** category, you'll find a couple of Azure AD-related services (as shown in the following screenshot):



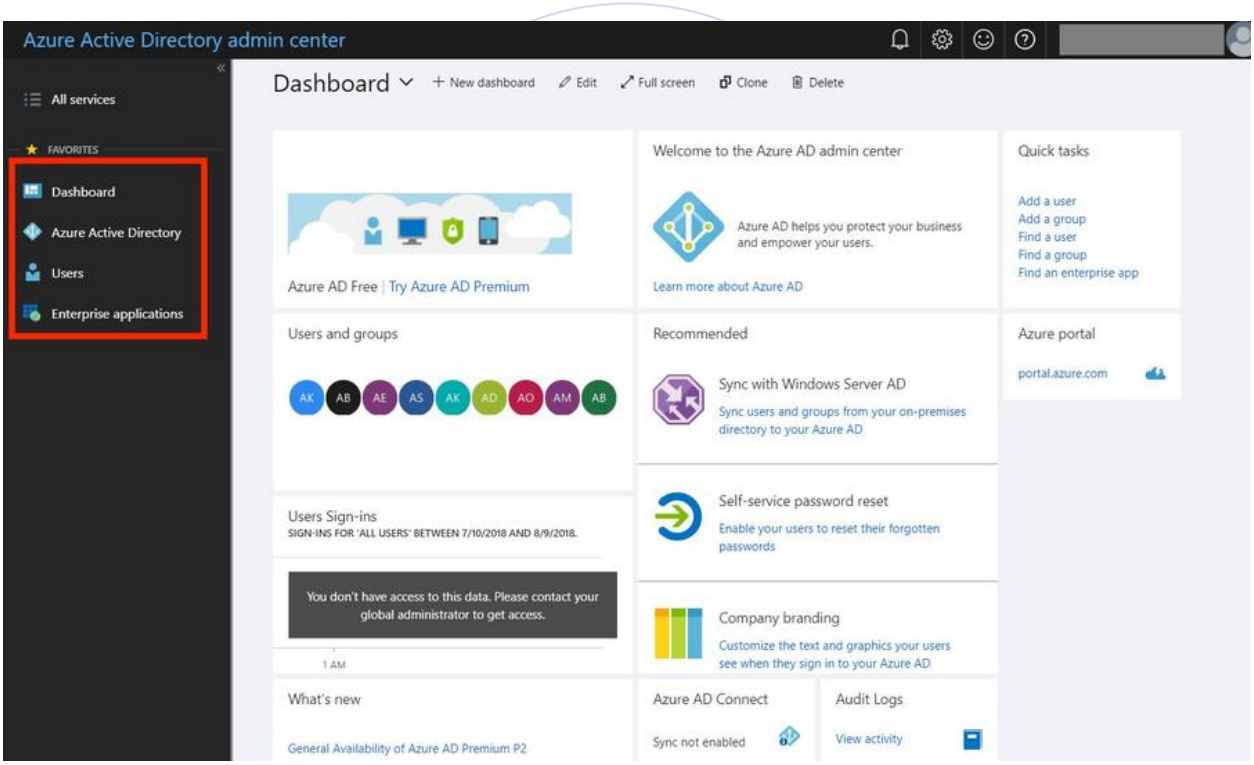
The following services are different capabilities provided by the Azure IDaaS solution:

- **Azure Active Directory**: It is a cloud-based directory, which is multi-tenant and supports identity management, application access management, and identity protection in a single solution.
- **Azure AD Identity Protection**: It is a feature provided by Azure that allows us to detect suspicious actions and configure responses automatically to protect the organization's identities.
- **Azure AD Privileged Identity**: It is an Azure Service to manage, control, and monitor access in Azure AD, Azure Resources, Office 365, Microsoft Intune, or other Microsoft online services within organization.

- **Azure AD Domain Services**: It provides managed domain features, such as domain join, LDAP, and Kerberos/NTLM authentication, which can extend the traditional Windows Server Active Directory.
- **Azure AD Connect Health**: It is a Azure service to monitor on-premises identity infrastructure and synchronization services.
- **Azure Information Protection (AIP)**: It is a cloud-based solution that protects an organization's documents and emails using labels based on rules and conditions.

In Azure, there is a portal that was specially created for Azure AD, and you can input the following link into your browser to access it: https://aad.portal.azure.com/

Have a look at the following Azure AD portal:

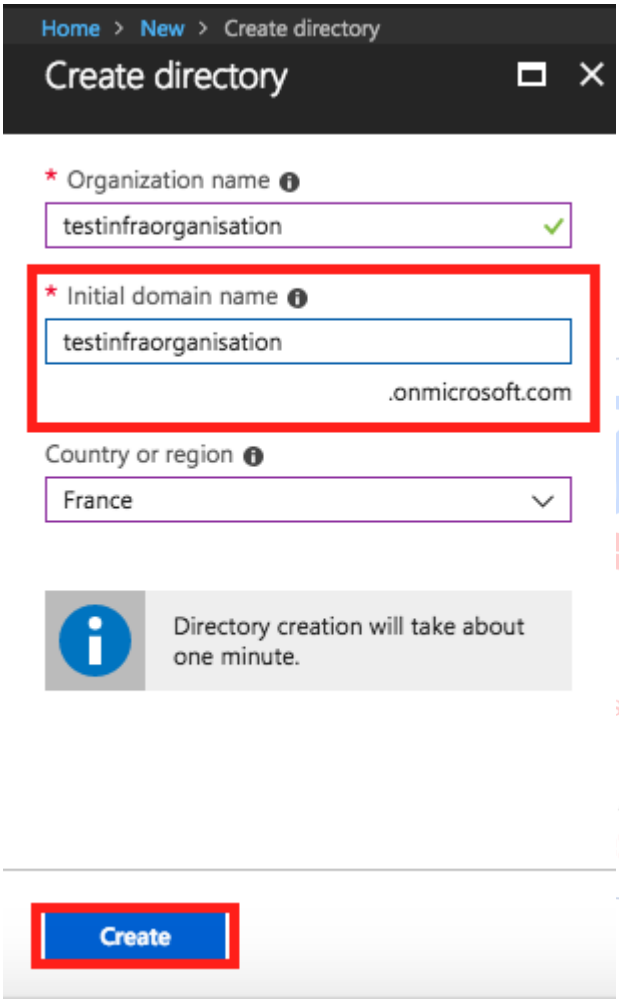


Creating an Azure Active Directory via the Azure Portal

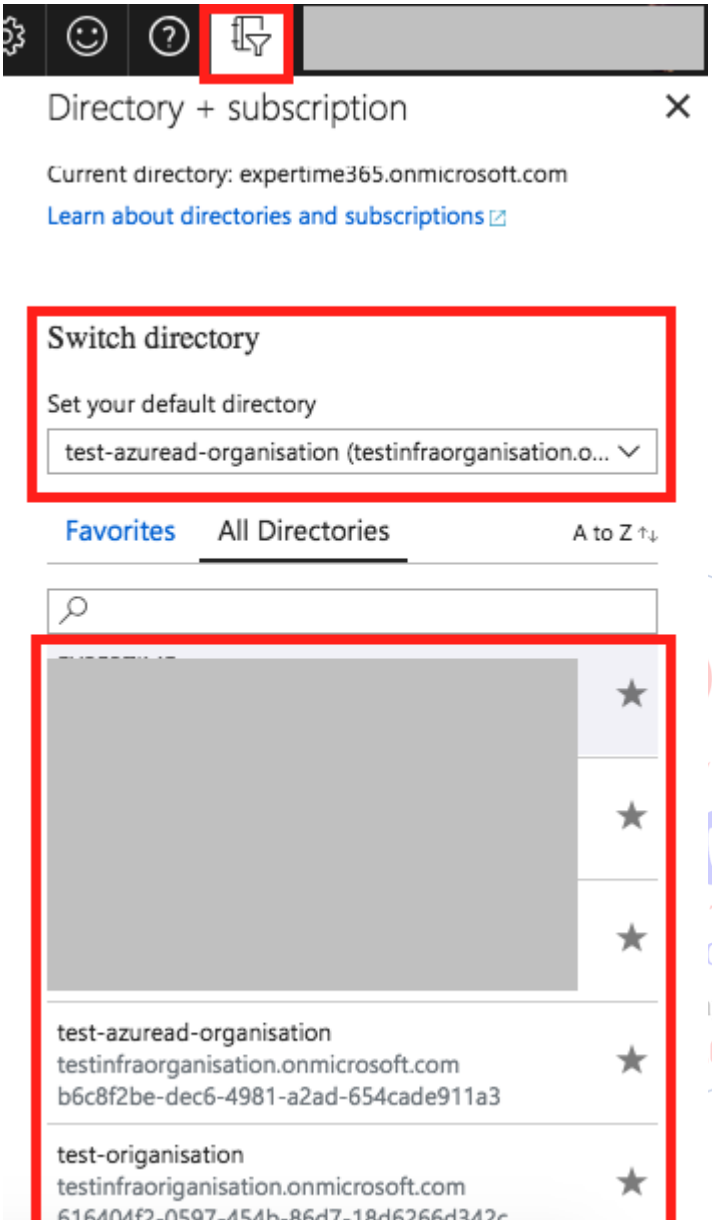
Creating an Azure active directory is very simple. Go to the Azure Portal, search for **Azure active directory**, and then click on **Create**. You'll see the **Create directory** form, then fill in your organization's name and the initial domain name (as shown in the following screenshot).

Generally, your directory will look like the following URL: #youroganisationname#.onmicrosoft.com.

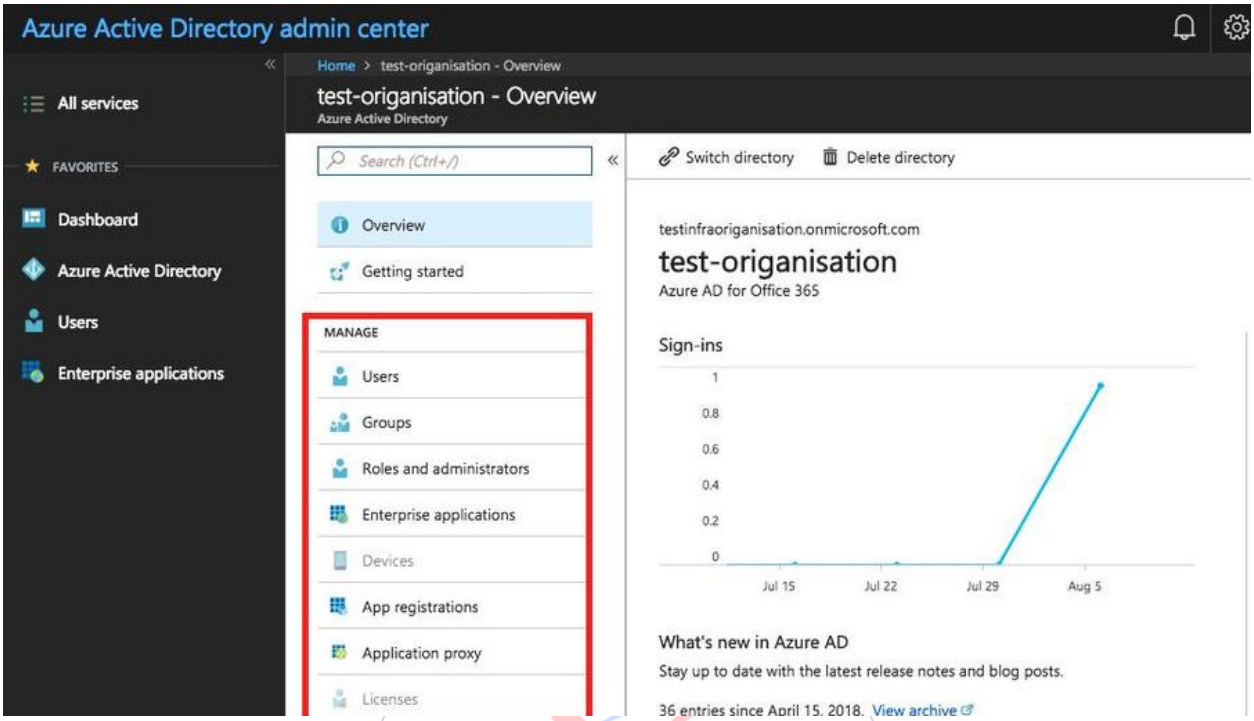
Then, click on **Create**, and it will take up to 1 minute to create a new directory successfully:



After creating an Azure AD directory, go to the top of the Azure Portal and find the **Switch directory** button (as shown in the following screenshot) to switch to the target directory:



After switching a directory successfully, you can go the Azure AD portal of this directory, shown in the following screenshot. You can manage the users, groups, and roles of current Azure AD and additionally integrate applications with the Azure AD directory:

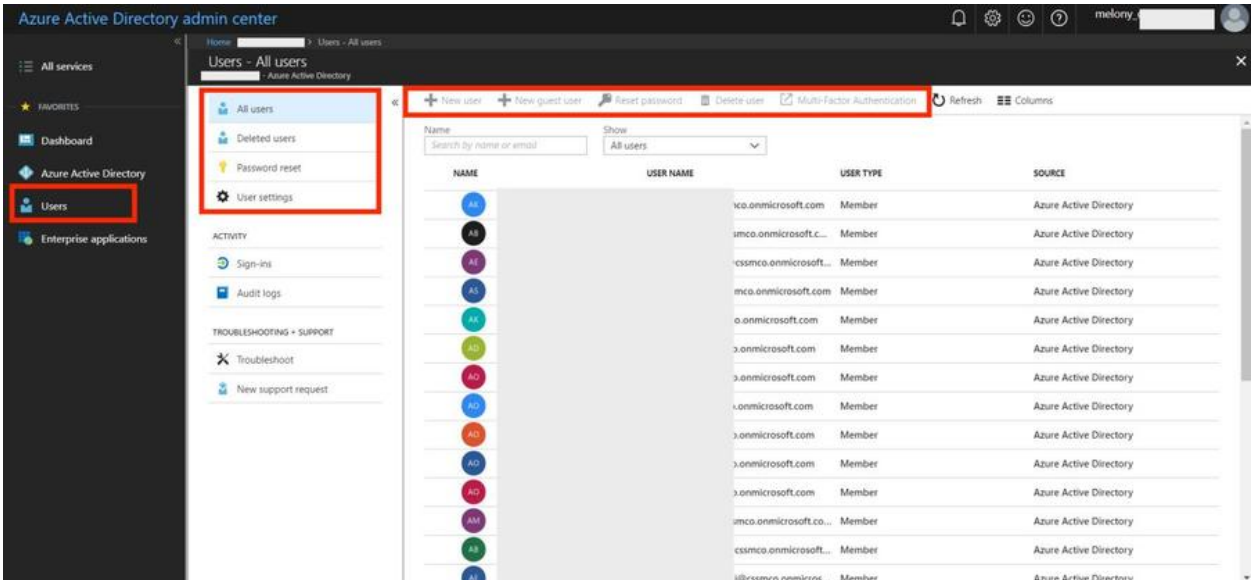


Creating and managing Azure AD users

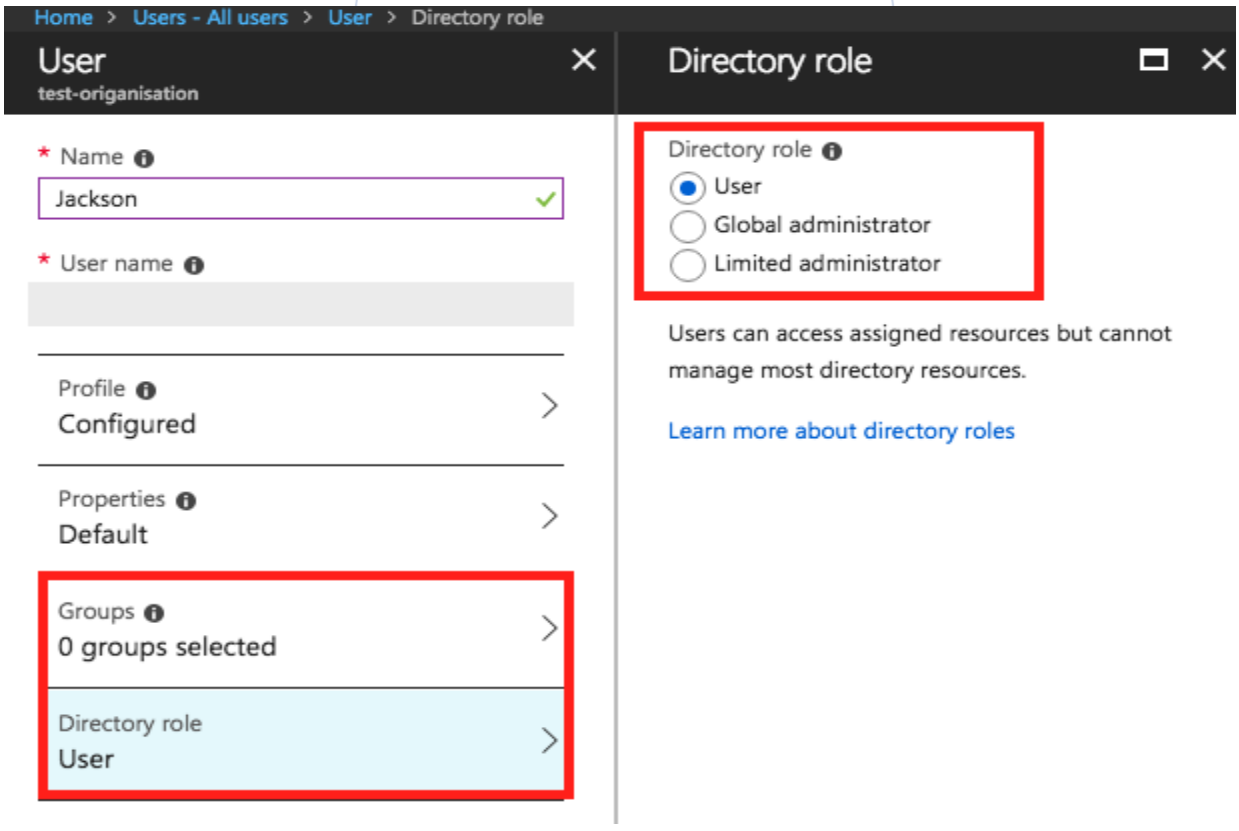
In the Azure AD portal, we can take advantage of the following capabilities:

- Creating a new user and guest user
- Deleting a user
- Resetting the password
- Configuring MFA for an existing user

After clicking on **All users**, you can as shown in the following screenshot with all the users in the current directory. At the top of the users list, you can find all these buttons to help you manage the users in your organization:

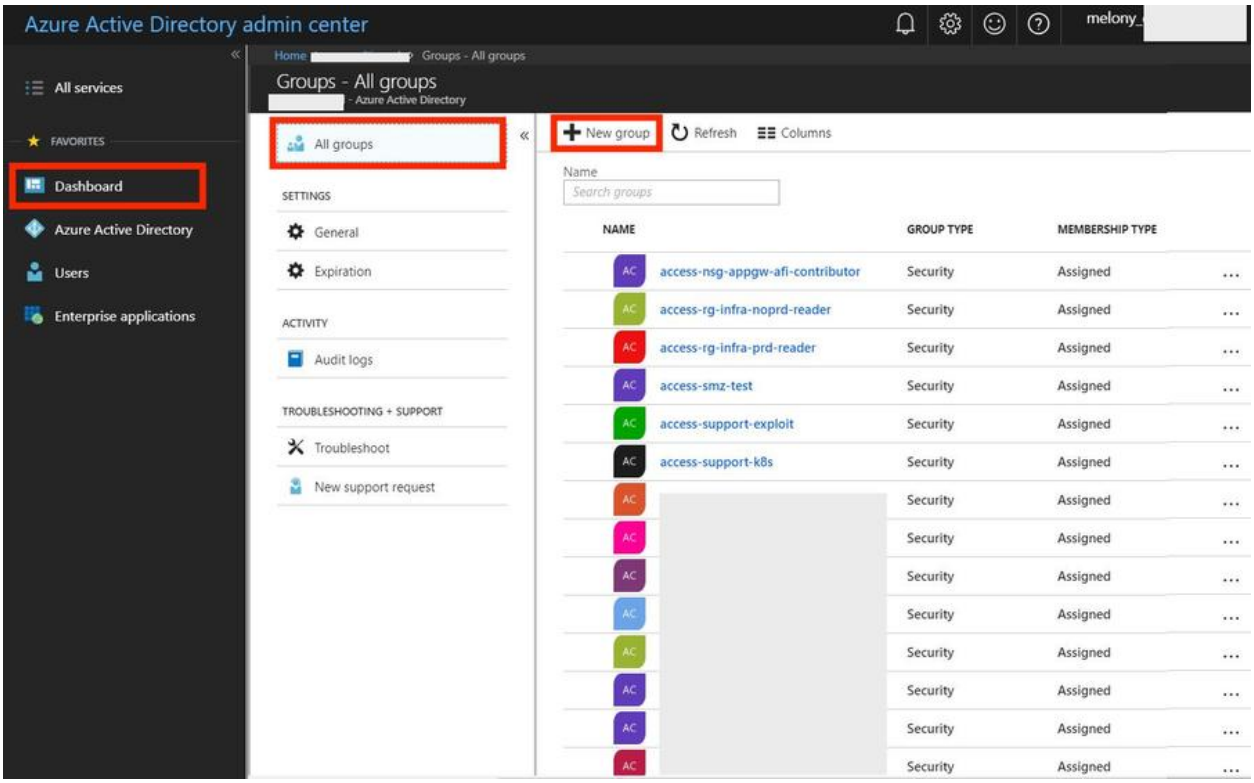


To add a new user, click on **+ New user** and then you'll see the page for user creation. On this page, you can configure the user group and its role. By default, the profile is only available as **Azure Active Directory**:

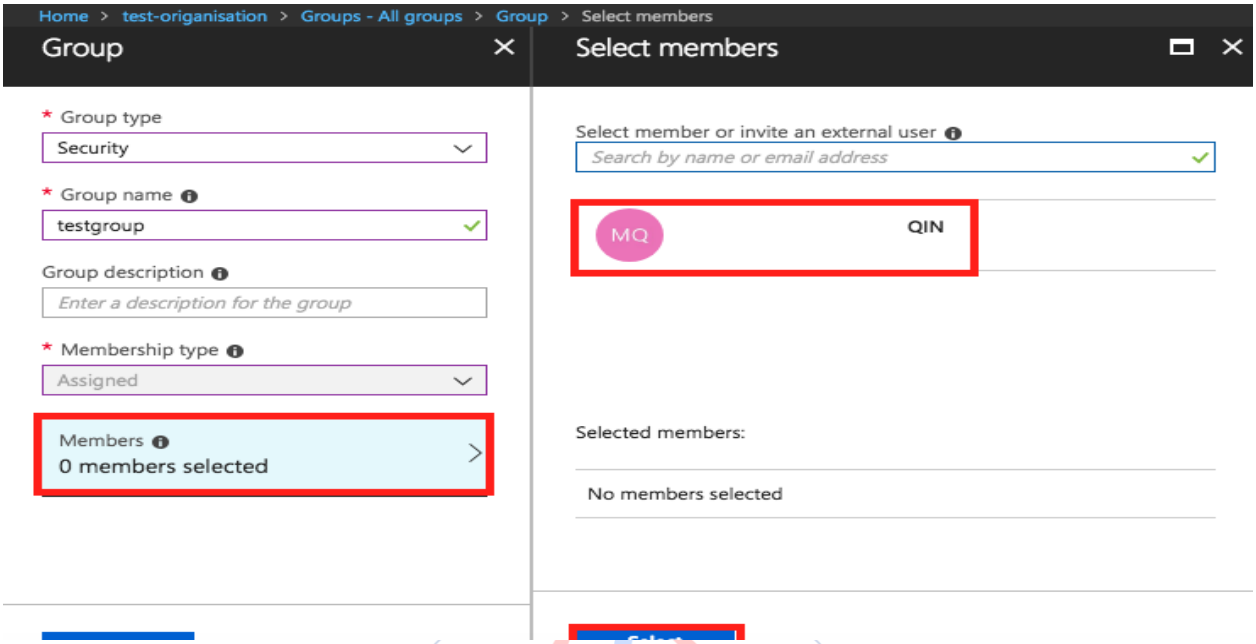


Creating Azure AD groups and managing user groups

You can go to the Azure AD portal and click on **Azure active directory**, then click on the **All groups** blade. Go to a new page, which has all the available groups within the organization (as shown in the following screenshot; you can manage your users in different group types, such as users in security part) and click on **+ New group** to create a new group:



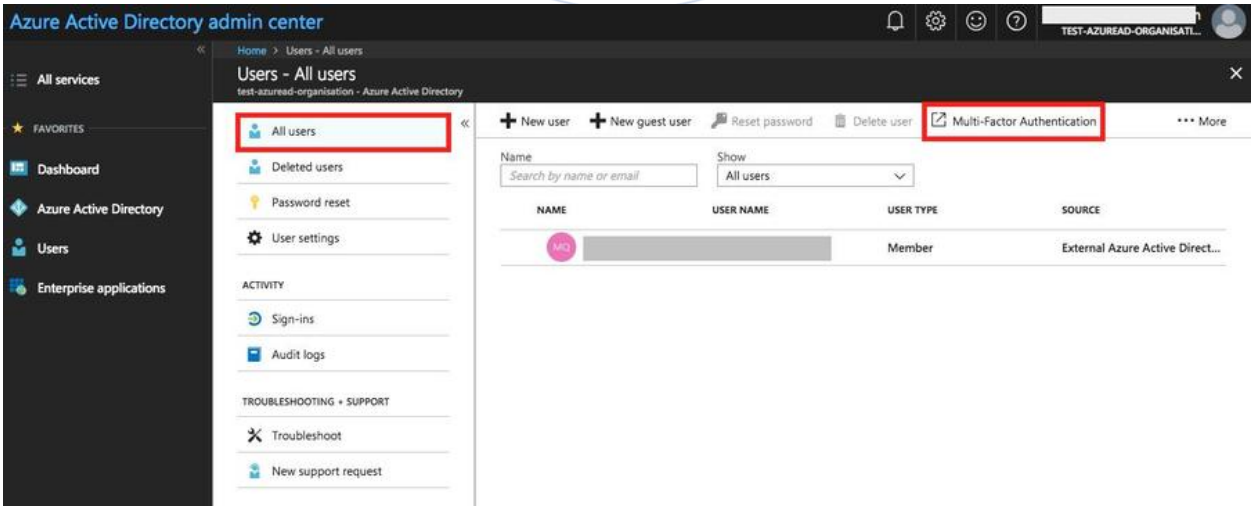
Then, fill in the group type that has office 365 or security type and group name. You can choose members in the member list or invite an external user to the group by clicking on **Members** and then **Select**, as shown in the following screenshot. Finally, click on **Create** to create a new Azure AD group:



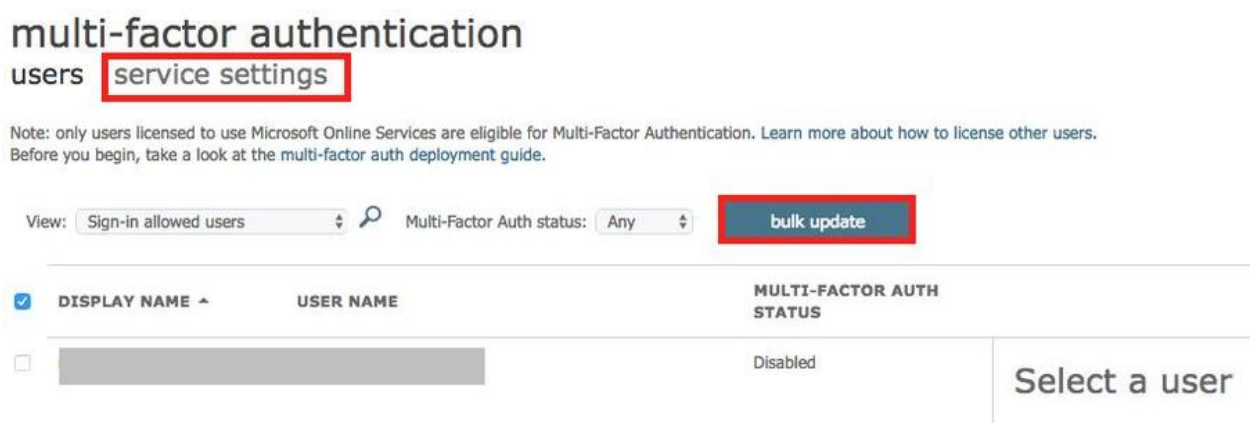
Enabling Multi-Factor Authentication for users

To enable the MFA feature, you should have **Azure AD Premium licenses** so that you have access to full-featured use of MFA in the Azure cloud or the on-premises Azure MFA server.

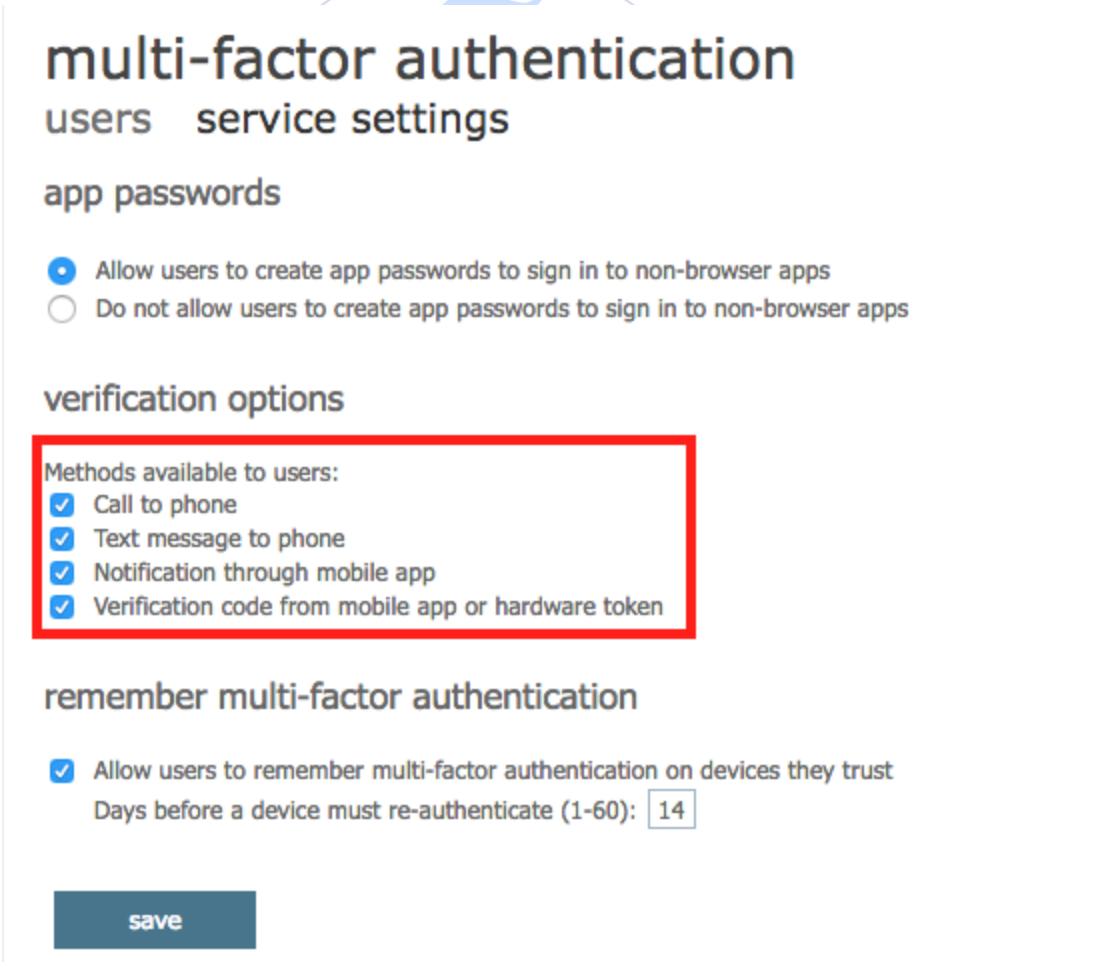
Make sure you meet all the prerequisites, then you can click on **Multi-Factor Authentication** to configure it (as shown here):



You'll see another page in another tab of your browser. Click on **service settings**:



In the **service settings** tab, you can configure all the information related to MFA, choose the available ways to users such as text message, phone, notification through push-up or verification code (as follows):

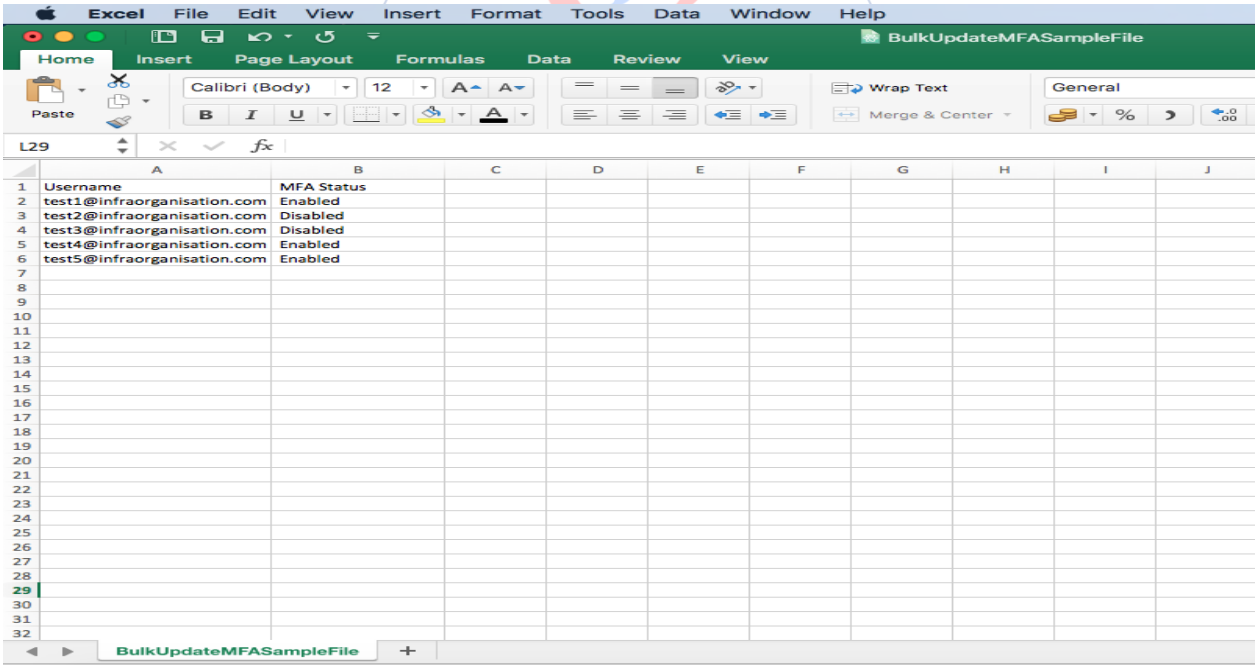


Using bulk update for custom user profile properties

When you want to configure the MFA feature for multiple users, it is possible to use the **bulk update** feature. To do this, you should click on the **bulk update** button in the users tab of the previous page. Then, you'll see a popup, shown here, where you can upload a .csv file:

a CSV file to enable MFA for multiple users

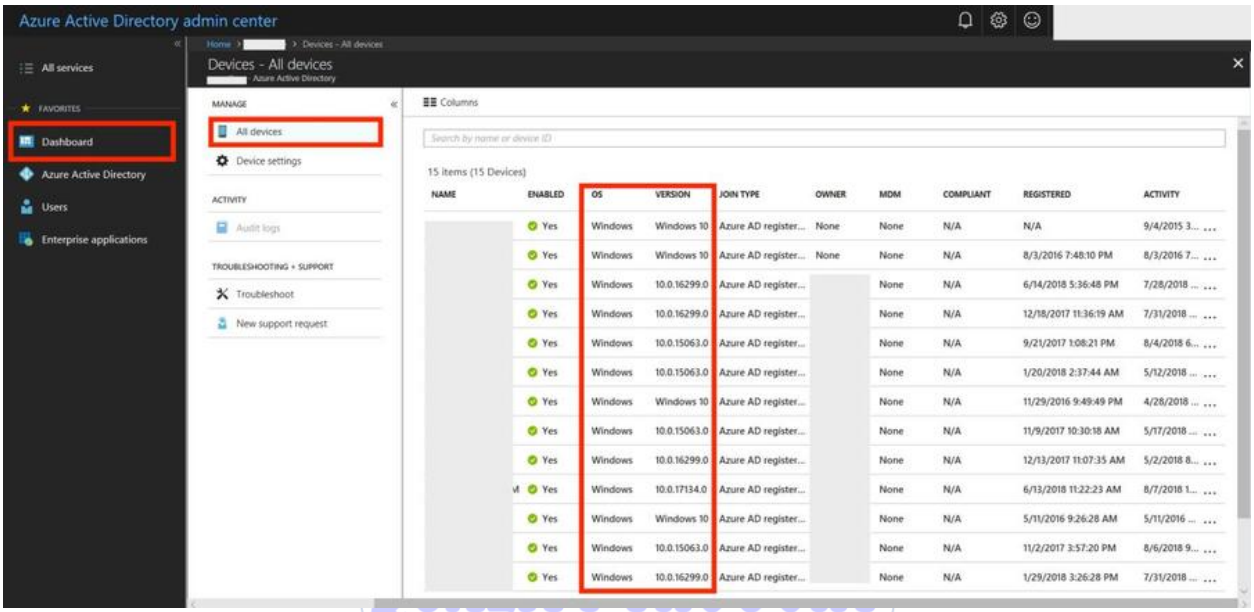
The example .csv file you'll upload is as follows:



How to use a CSV file to configure MFA for multiple users

Managing devices

Azure AD allows you to manage single sign-on for devices, apps, and services from anywhere. Go to the Azure AD portal and click on **Devices** to manage all the devices that have been registered in the repository (as shown here):



Azure AD provides support for these devices for the **Bring Your Own Device** (**BYOD**) scenario so that the users in the directory that is with work or school account can work with different devices, such as on laptop, tablet, or mobile, and across different OSes, such as Windows 10, iOS, Android, and macOS.

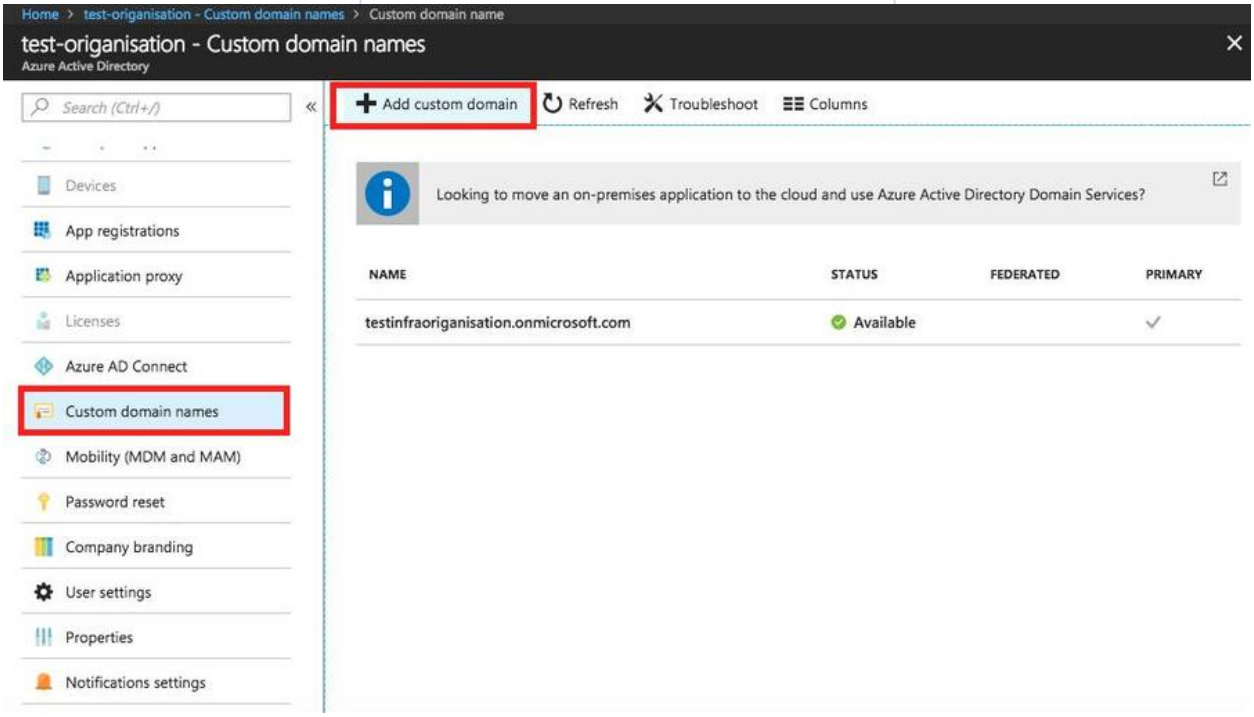
You may be wondering how did this device list come, if you want to know more about how to configure a new Windows 10 device joining Azure AD, you can refer to the following link: https://docs.microsoft.com/en-us/azure/active-directory/devices/azuread-joined-devices-frx.

Add a custom domain

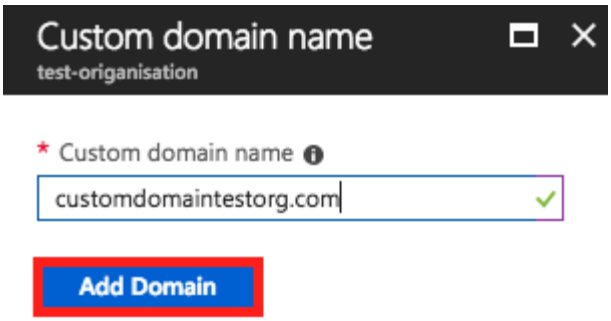
When creating a new Azure AD directory, we configured an initial domain that is in the form of organisationname.onmicrosoft.com; the domain name cannot be changed or deleted, but the major user may not be familiar with this domain. Azure provides a friendly feature to let an administrator add a custom domain name for the Azure AD directory. After adding the custom domain name, the organisationname.onmicrosoft.com URL becomes the following:

youname.organisationname.com

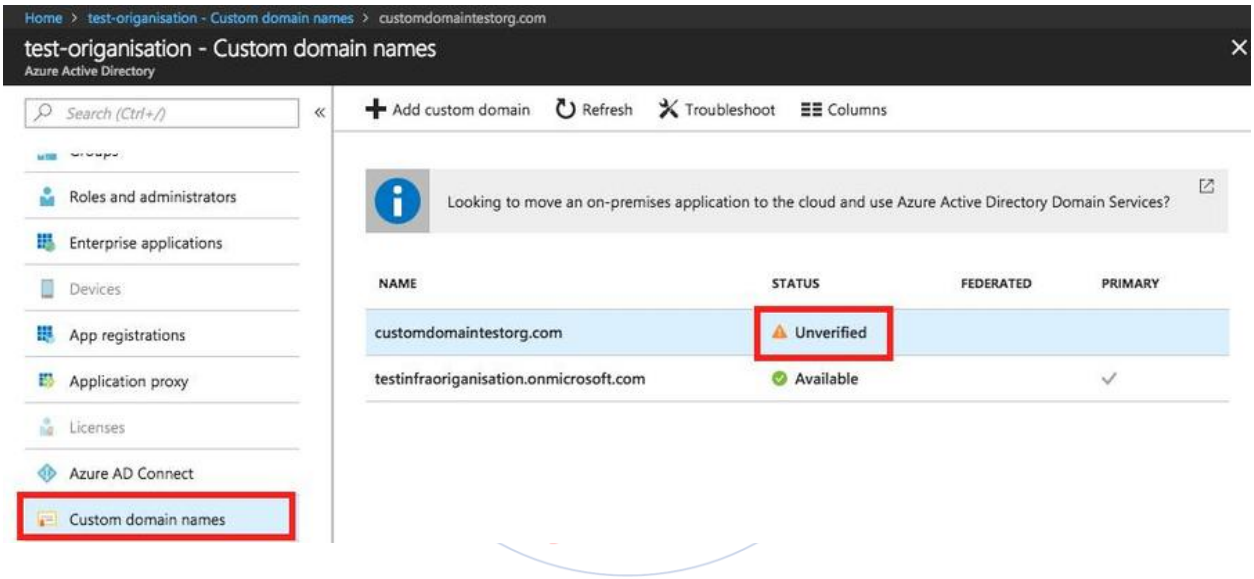
An example is test@qualitythought.com. Here, test is your name or the name of anyone else, and qualitythought is the domain name; it can be another domain name that you expected. To add a custom domain name, you need to go to the Azure active directory page and click on the **Custom domain names** blade, then click on the **+ Add custom domain**button (shown in the following screenshot):



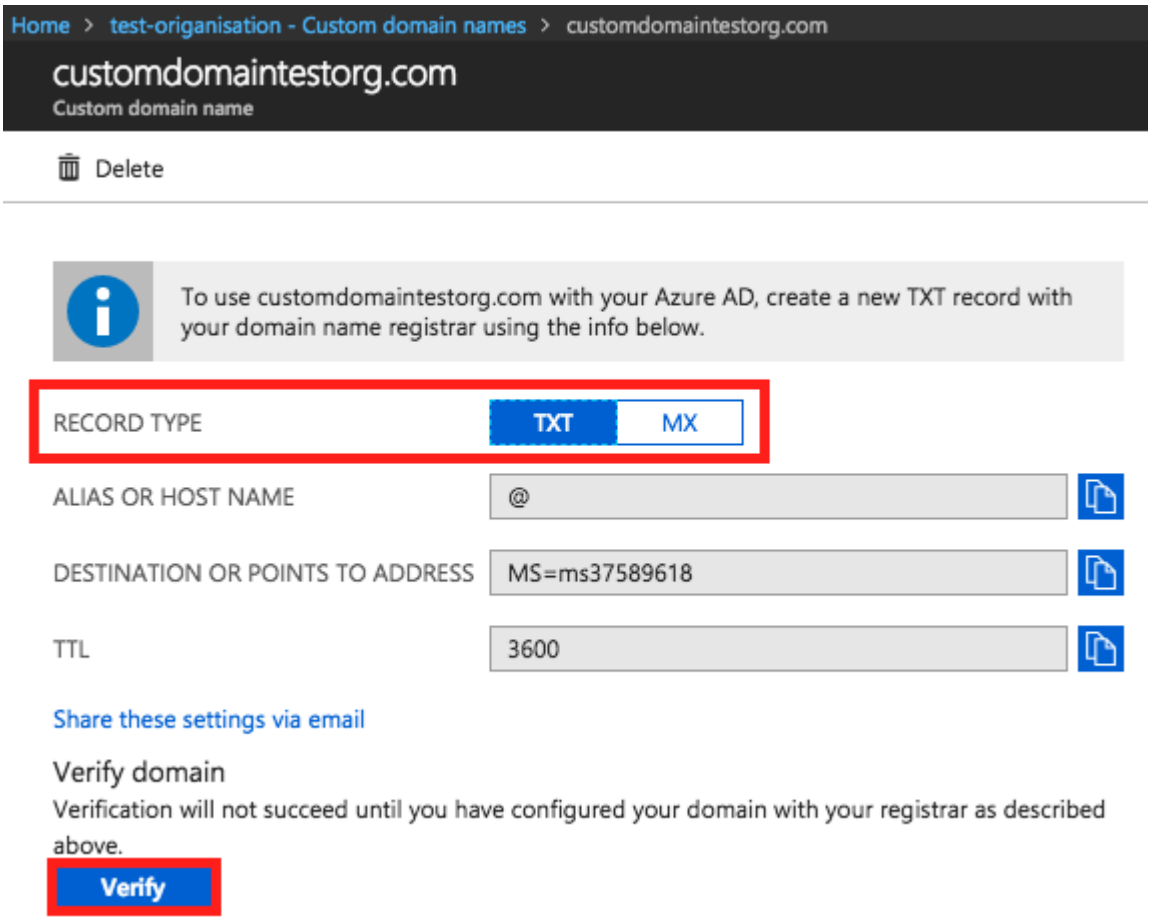
Then, you'll get a create **Custom domain name** page, and you can fill in the page **Custom domain name** and add the target domain in this page (as shown in the following screenshot):



After creating this domain, it may not be operational yet. Go back to the **Custom domain names** page. As you can see as the following screenshot, the status of this domain is marked as **Unverified**. This means that this domain name should be verified by Azure:



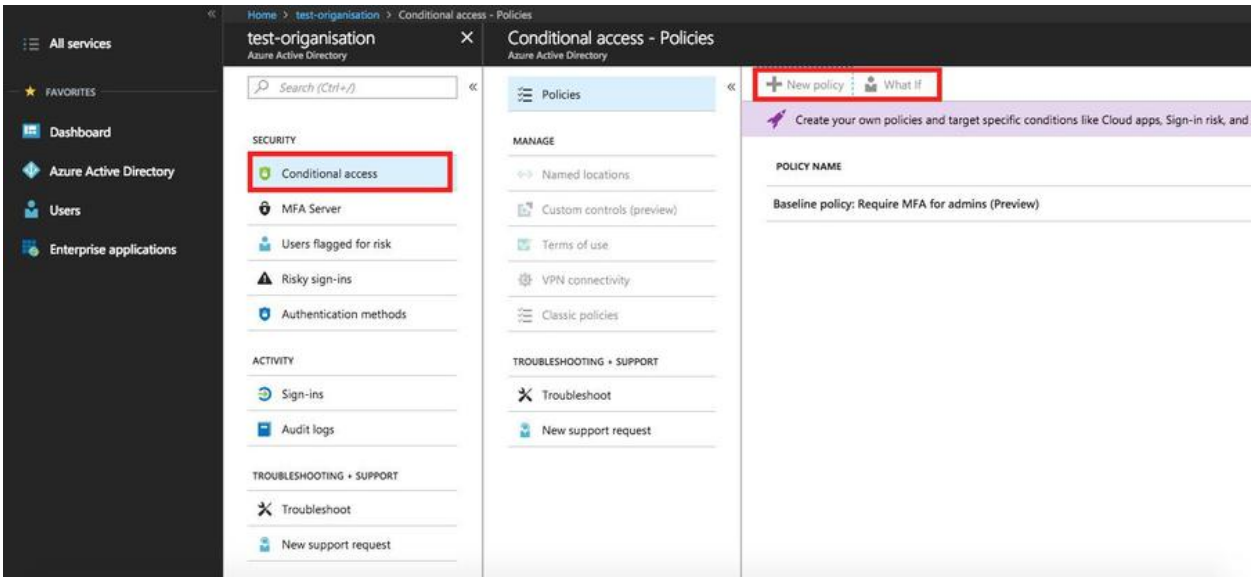
Click on this domain label, then go to the verify page. Choose both the types of DNS records, TXT or MX, as record types here. The MX record specifies where the emails for your domain should be delivered, and the TXT record is used to store text-based information related to your domain. We choose the TXT record here and click on **Verify**, as shown in the following screenshot. If everything goes well, after a few minutes or so, the newly created custom domain will be operational:



Conditional access

Azure Active Directory also provides a very cool ability to control access to cloud native applications based on conditions. The conditions defined in the condition policy that defined **when does this happen and what to do when this happens**.

You can define a new condition policy via the Azure Portal; go to the Azure AD portal and click on the **Conditional access** blade (as shown in the following screenshot):

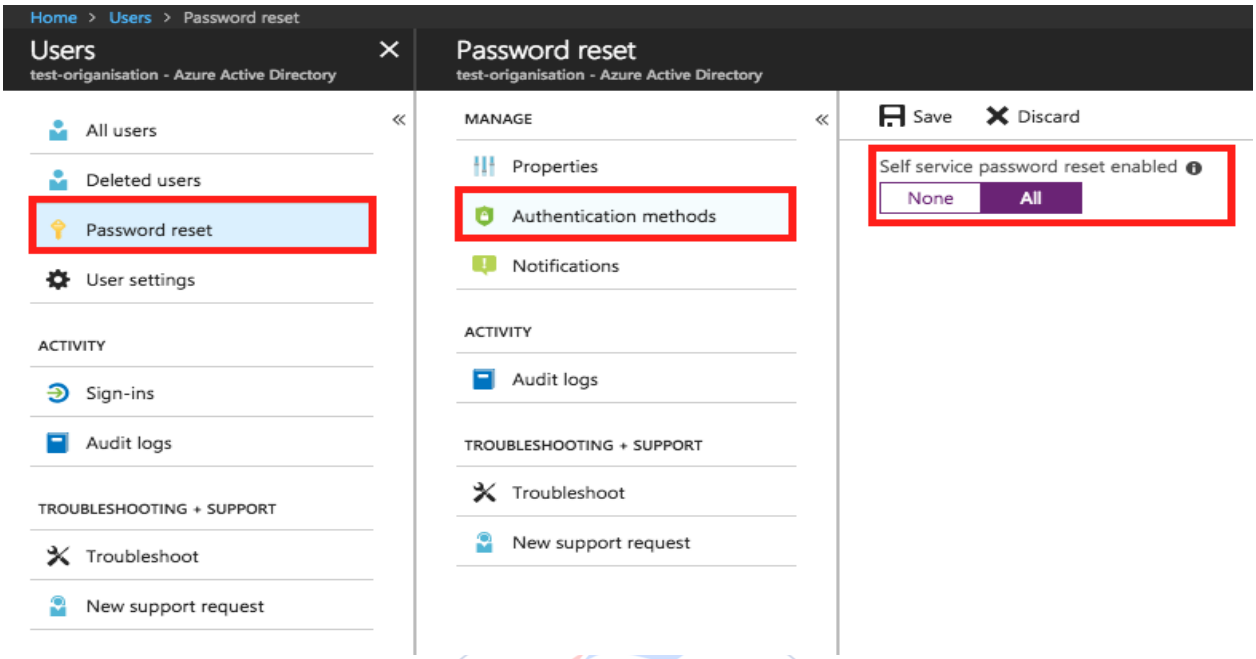


To find out more about how to define conditions, refer to the following link: https://docs.microsoft.com/en-us/azure/active-directory/conditional-access/conditions.

Configuring self-service password reset

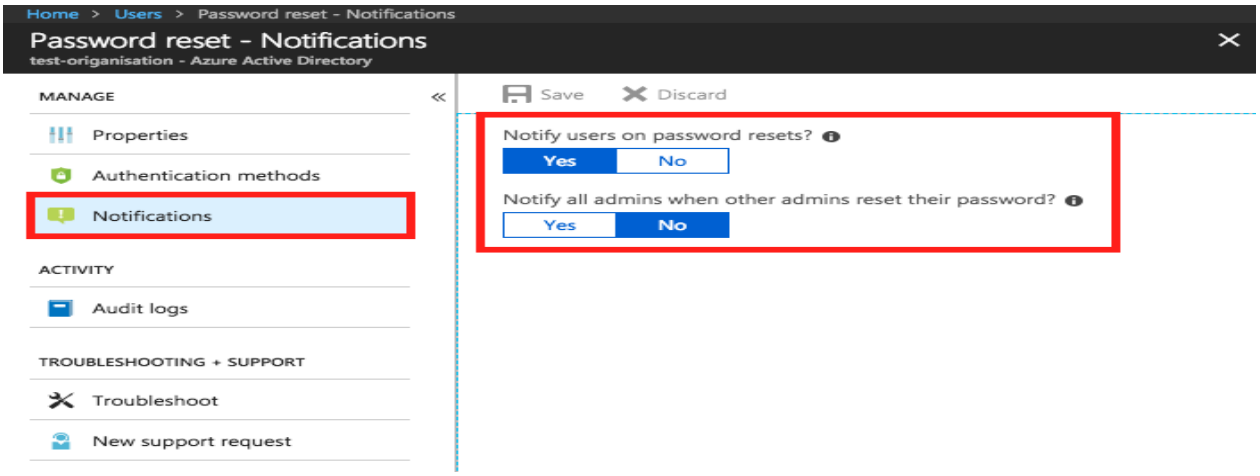
Within an organization, IT administrators usually have a couple of regular and important tasks, such as managing users and their identities. One of the best practices for managing cloud security is to enable users to reset their passwords or unlock their accounts by configuring **self-service password reset** (**SSPR**).

To enable this feature, go to the Azure AD portal and click on **Users**, where you can see the **Password reset** blade. Set **Self-service password rest enable** as all value in the authentication methods blade and then click on **Save**, as shown in the following screenshot:



After enabling the self-service password reset feature, Go back to the authentication methods blade to configure self-service details such as the number of methods required to reset and the methods available to users, which include mobile phone, office phone, or email. Note that it would be great to also set the notifications to inform administrators and users so that they know the password has been reset successfully.

Go to the **Notifications** blade to configure the notification to users and administrators, as shown in the following screenshot:



Set notifications to notify users and administrators on password reset

You may be wondering, if we still have our existing on-premises directory, how do we get both Azure AD and our on-premises Active Directory Domain Services (AD DS) environment to be managed in a centralization way? Azure provides a very cool feature, **password writeback**, to help you synchronize password changes in this kind of hybrid scenario. After password writeback has been enabled, as it is a part of Azure AD Connect, it will send password changes back to an existing on-premises directory from Azure AD in a secure way.

To find out more about how to enable password writeback for your hybrid environment, check the following link: https://docs.microsoft.com/en-us/azure/active-directory/authentication/tutorial-enable-writeback

Configuring privileged identity management

The Azure AD also provides the Azure AD privileged identity management capability. With this feature, you can manage, control, and monitor on-demand and just-in-time administrative access in Azure AD, Azure Resources, Office 365, Microsoft Intune, or other Microsoft Online Services within an organization. To find out more about how to configure it, checkout the following link: https://docs.microsoft.com/en-us/azure/active-directory/privileged-identity-management/pim-configure.

Configuring Azure AD identity management

Azure AD identity protection is a great feature allows you to detect suspicious actions and configure responses automatically to protect the organization's identities. You can enable it by creating Azure AD Identity Protection via Azure Portal.

Refer to the following link to find out more about how to configure Azure AD Identity Management in Azure: https://docs.microsoft.com/en-us/azure/active-directory/identity-protection/overview.

Leveraging Microsoft Graph other than Azure AD Graph API

The **Azure Active Directory Graph API** (**Azure AD Graph API**) provides a programmatic way to access Azure AD using RESTful APIs. Developers can program call the the Azure AD Graph API to perform **create, read, update, and delete**(**CRUD**) operations on Azure AD data and objects, such as creating a new user in a directory and then getting the target user's detailed properties. Microsoft recommends us to use Microsoft Graph instead of the Azure AD Graph API to access Azure Active Directory objects. **Microsoft Graph** is the API in **Microsoft 365**, which is a more powerful identity API than the AAD Graph API,. It allows to connect to **Office 365, Windows 10**, and **Enterprise Mobility** in a secure way. To know more about Microsoft Graph, Checkout the following link: https://developer.microsoft.com/en-us/graph.

Integrating applications with Azure AD

Azure AD **Business-to-Consumer** (**B2C**) is a dedicated Azure AD tenant provided by Microsoft Azure that enables individual customers working with organizations to get access to custom web applications, mobile apps, and desktop applications.

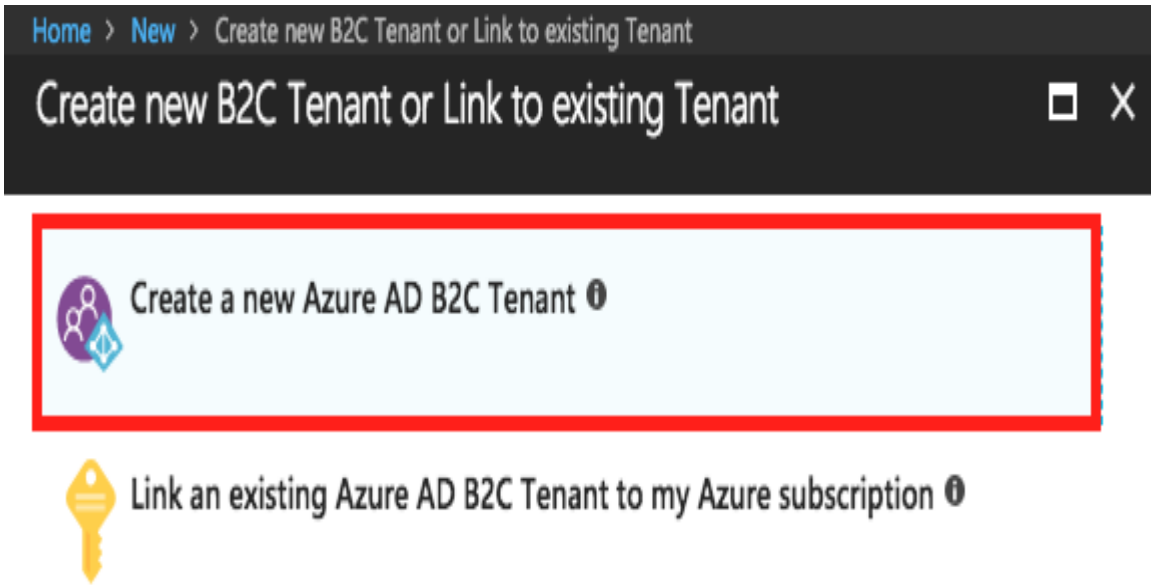
Azure Active Directory B2C helps organizations and enterprises worldwide to connect to their customers and serve their applications with a high level of cloud-based identity protection.

Azure AD B2C supports popular protocols such as**OpenID Connect**,**OAuth 2.0**, and**SAML**. The accounts used by Azure AD B2C can be created directly in the Azure B2C tenant or provided by

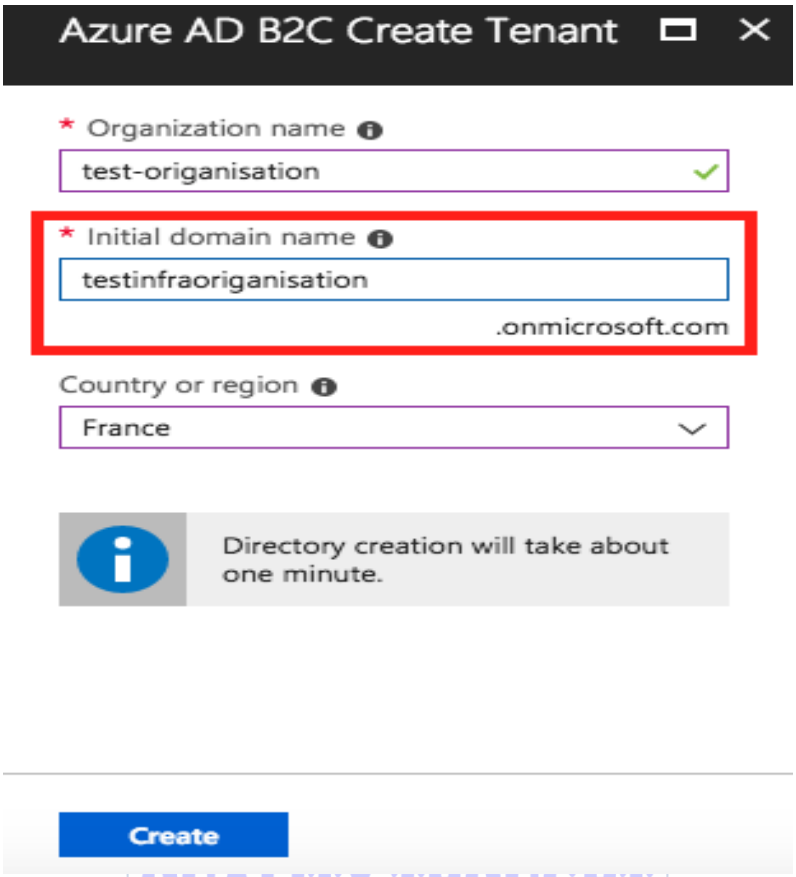
famous social media identities, such as Facebook, Google, Amazon, LinkedIn, and Twitter.

Creating an Azure AD B2C directory

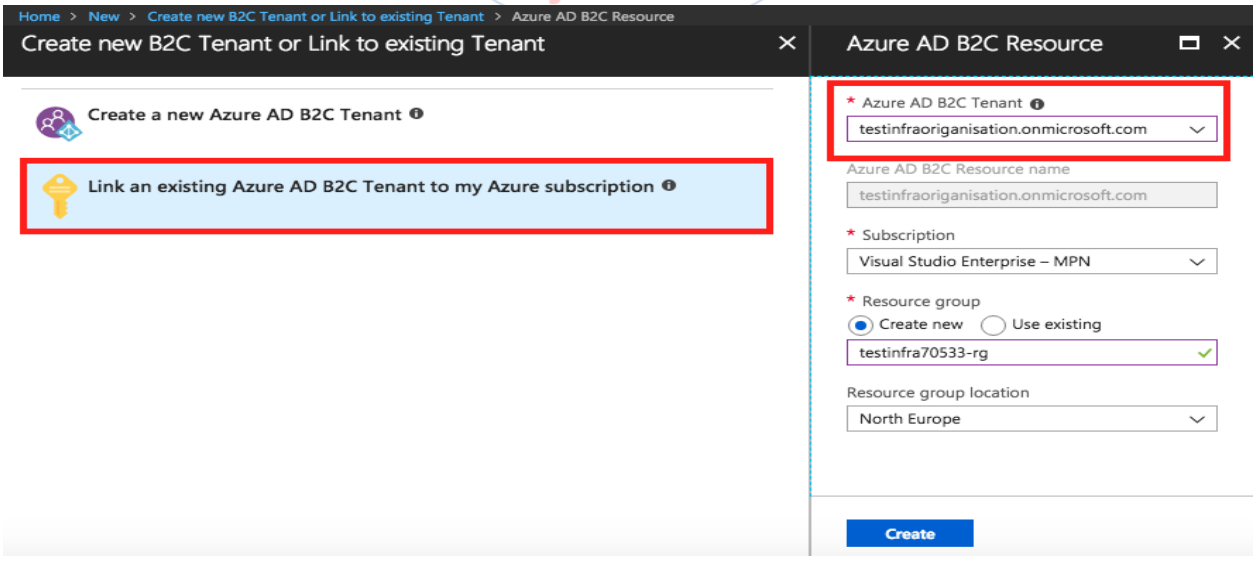
To create a Azure AD B2C directory, go to the Azure Portal and click on **Create a new resource**, then choose **Identity** repository and **Azure Active Directory B2C** and click on **Create**; you can create a new Azure AD B2C Tenant or link an existing Azure AD B2C Tenant to a Azure Subscription, as shown here:



Here, we'll create a new Azure AD B2C tenant. To create a new Azure AD directory, fill in the organization name and initial domain name, then choose a location before clicking on **Create**, as shown in the following screenshot:

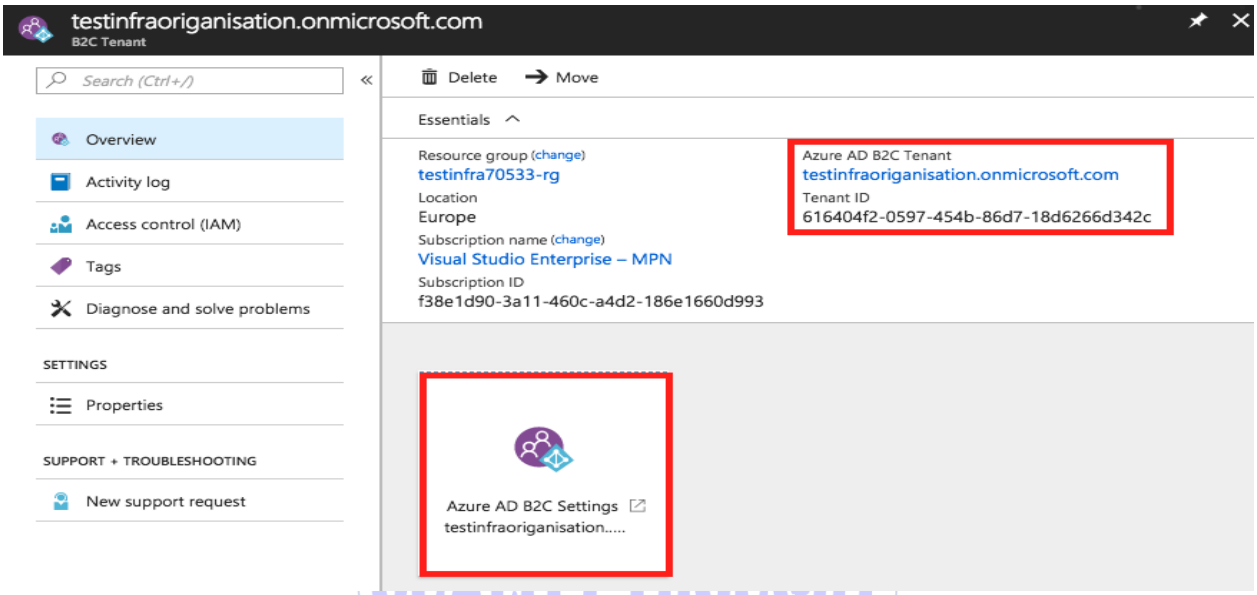


After creating an Azure AD B2C tenant, you can link it to an Azure subscription by choosing it and filling in other information such as resource group, subscription, and location, then click on **Create**. Azure will manage the remaining work:

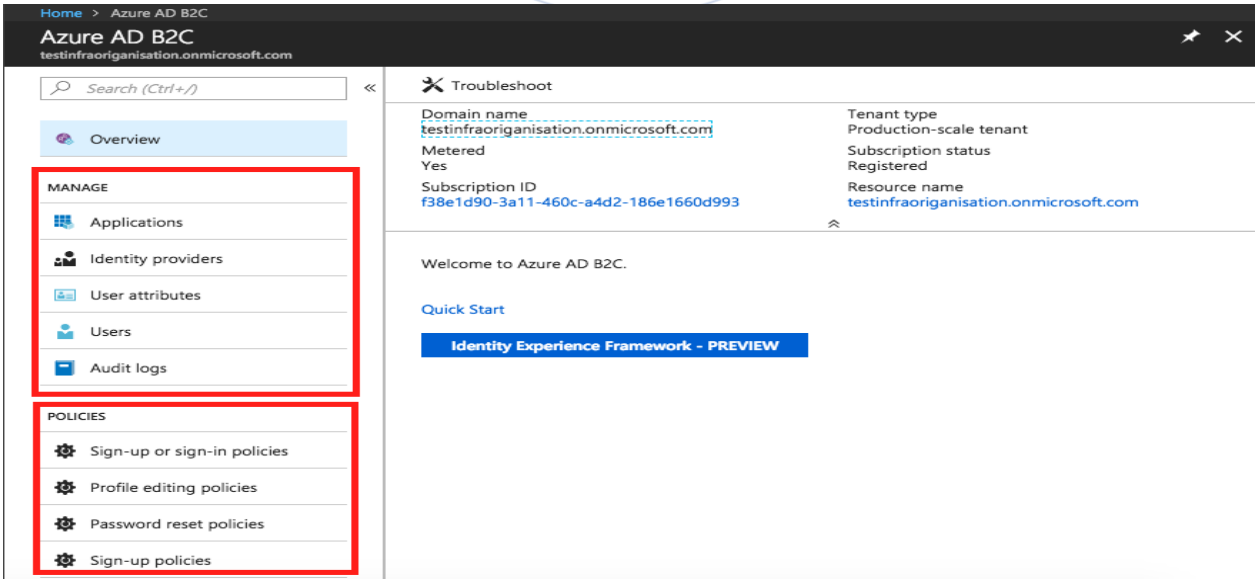


Managing Azure AD B2C directory

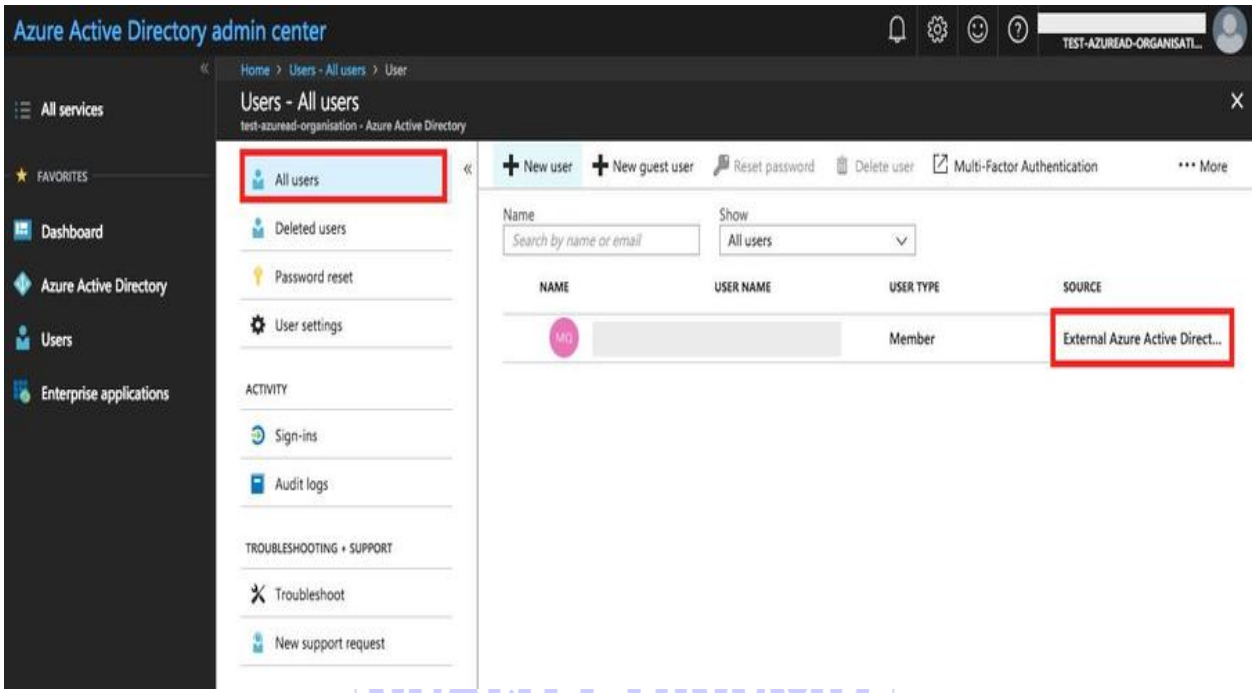
After creating a B2C directory, go to the resource and check the **Overview** blade, where you can get overall information on the B2C tenant and click on **Azure AD B2C Settings** to manage this tenant, as shown in the following:



You'll see the following page after clicking on **Azure AD B2C Settings**, where we can manage all the users and groups in this directory, as well the application linked to the current directory:



If we go back to the Azure AD portal the current Azure B2C directory and click on **All users**, you can see that the users in this directory are referenced as **ExternalAzure Active Directory source**:



Azure B2C also supports the use of built-in policies to create a wonderful login experience within minutes. It can build custom policies and integrate with CRMs, databases, marketing analytics tools, and other account verification systems.

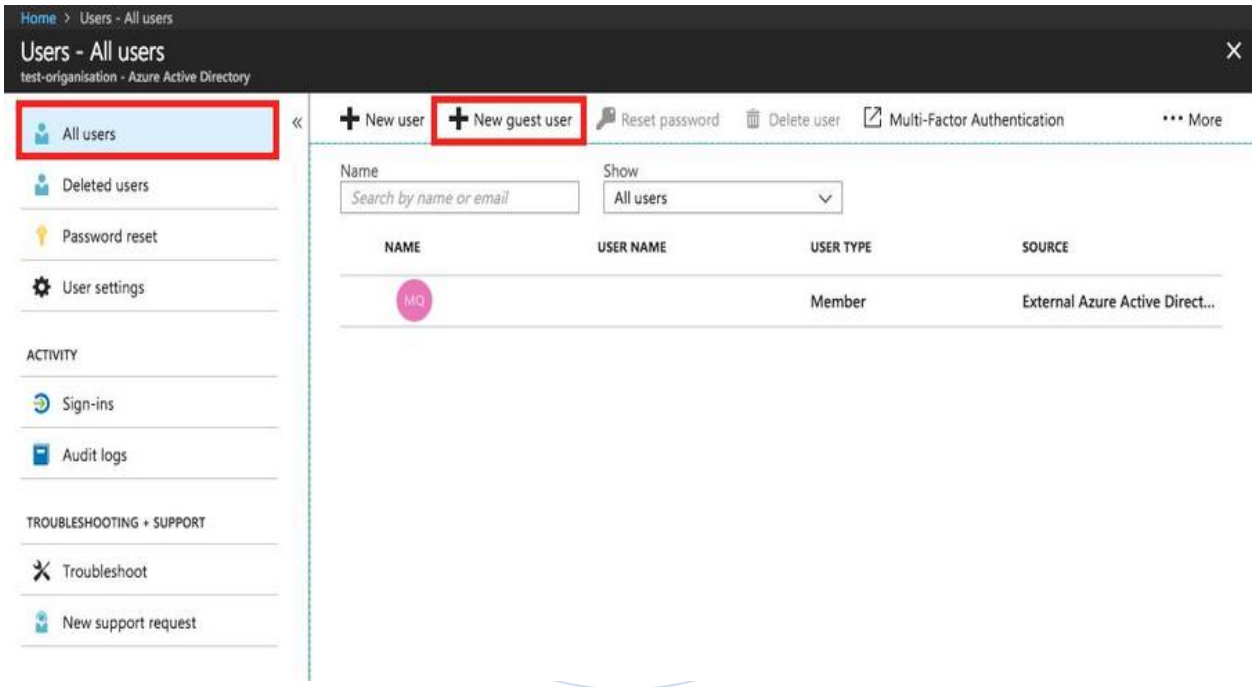
If you want to know more about how to use built-in policies with Azure B2C, which can be applied in most general scenarios, check out the following link: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-reference-policies.

For more complex scenarios, you can use custom policies, which are still in preview when this book was written. You can get more information from here: https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-get-started-custom.

Implementing Business to Business (B2B) collaboration

Azure AD **Business to Business** (**B2B**) collaboration capabilities enable any organization using Azure AD to work safely and securely with users from other partner organizations, using school or work email.

You can enable B2B collaboration by adding guest users to your organization in Azure AD as shown here:



Then, you'll see a **New Guest User** page, where you can add the email address of your partner and add a message in the email invitation, then click on **Invite**

Once you click on **Invite**, users will receive an invitation with a redemption URL, and then they can review and accept the privacy terms.

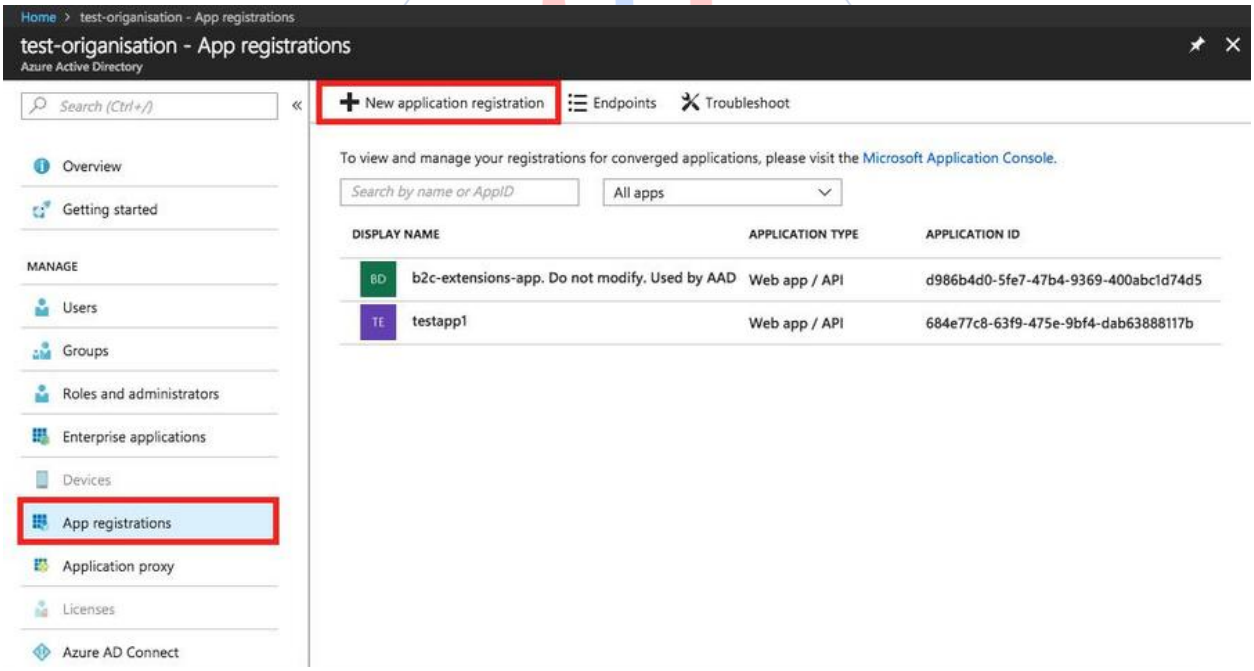
It is also possible to add B2B collaboration guest users without an invitation; check out the following link more information: https://docs.microsoft.com/en-us/azure/active-directory/b2b/add-user-without-invite.

Integrating applications with Azure AD

There are millions of enterprise-level **software-as-a-service** (**SaaS**) applications are boosting IT market. Application providers and enterprise developers want a way to integrate line-of-business applications with a cloud-based IDaaS solution such as Azure AD to provide secure sign-in, authorization, and SSO capabilities.

In Azure, to integrate a web application with Azure AD must first be registered in an Azure AD tenant. The registration process includes giving Azure AD a URL to specify where it's located and to send replies after a user is authenticated.

To register an application, go to the Azure AD portal and click on **App registrations**, and then register a new application by clicking **+ New application registration**, as shown here:



In the **Create** form, you should choose the **Application type**, which is **Web app / API or local app**, then fill in the **Sign-on URL** and click on **Create**:

After a few seconds, you can see your application has been registered successfully. As shown in the following screenshot, **Application ID** is your application identity, which is known by the Azure AD tenant

Microsoft also recommends to use Azure AD **Managed Service Identity** (**MSI**) as your application identity, as it simplifies creating an identity for code. To know more about Azure AD Managed Service Identity, you can check the following link: https://docs.microsoft.com/en-us/azure/active-directory/managed-service-identity/overview.

If you have an existing application that has its own account system and it may require to support other kinds of sign-ins and from other cloud providers, in that case, you may need to sign in any Azure AD user by using the multitenant application pattern. Check out the following link to have more information: https://azure.microsoft.com/fr-fr/resources/samples/active-directory-dotnet-native-desktop/.

You can also integrate Windows desktop applications, universal application with Azure AD by using ARM template. The following are some excellent sample ARM templates on GitHub which can help you:

- Integrating Azure AD into a Windows desktop application using interactive authentication:

  https://azure.microsoft.com/fr-fr/resources/samples/active-directory-dotnet-native-desktop/

- Integrating a Windows Universal application with Azure AD:
  https://azure.microsoft.com/en-gb/resources/samples/active-directory-dotnet-windows-store/

To integrate Azure AD with a web application using OpenID connect and WS-Federation, it is also possible to use an ARM template, check out the following links:

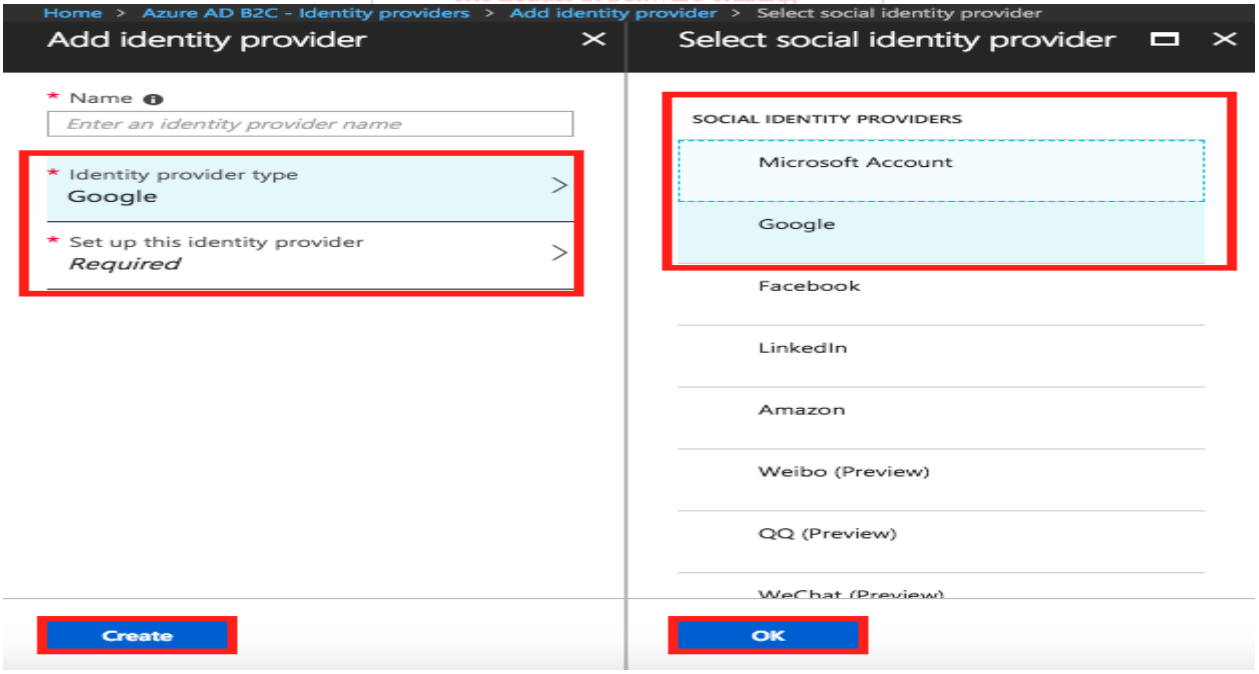
- Integrate Azure AD into a web application using OpenID Connect:

  https://azure.microsoft.com/fr-fr/resources/samples/active-directory-dotnet-webapp-openidconnect/

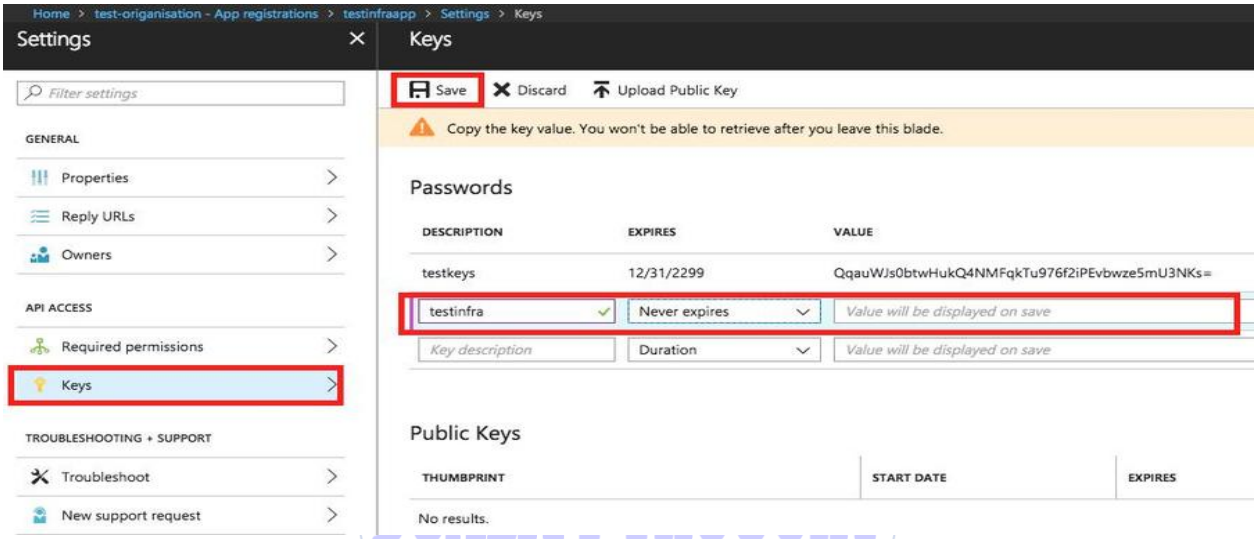
- Integrating a web app with Azure AD using WS-Federation:
  https://azure.microsoft.com/en-gb/resources/samples/active-directory-dotnet-webapp-wsfederation/

Implementing federation and social identity provider authentication

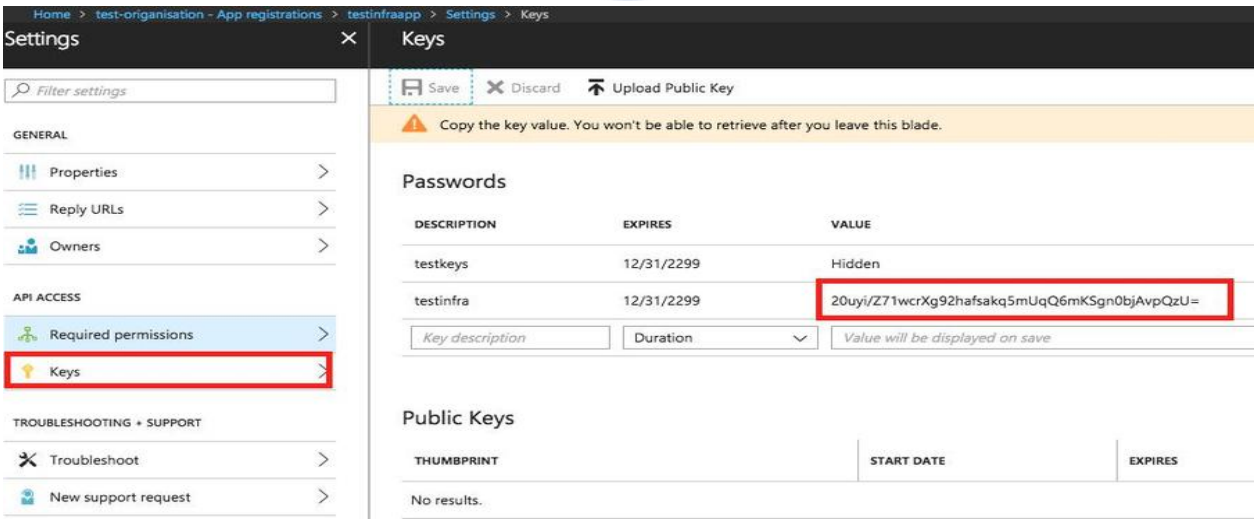
Most websites may have different customers across different social media provided to configure federation with public consumer identity providers such as Facebook, Google, and Twitter. Go to your Azure AD B2C tenant and click on **Identity providers** to add an identity provider, as shown in the following screenshot; you have a wide range of identity providers to choose from, such as **Microsoft Account**, **Google**, **Facebook**, **Twitter**, and **LinkedIn**. There are also some popular Chinese social media organizations (currently in preview), such as **WeChat**, **Weibo**, and **QQ**. Choose which you expect and click on **OK**:



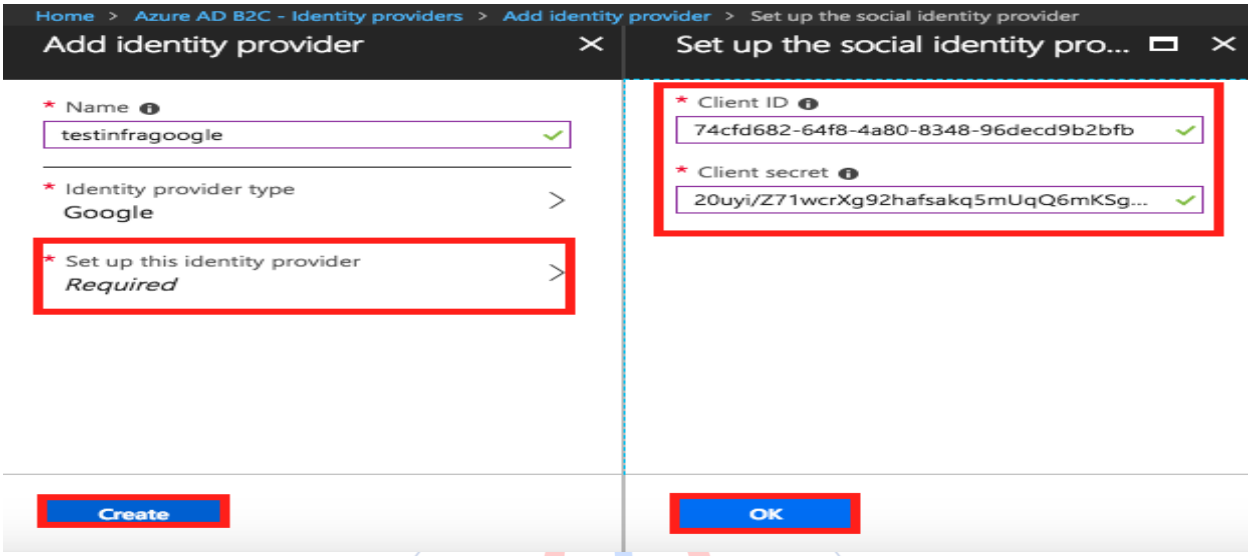
After selecting the identity provider type, you may also need to set up this identity provider using the Application ID that we mentioned in the previous section while registering a web application in the Azure AD tenant. You also need an application secret to set this up. We can get this secret by clicking on **Settings** in the registered application, and then go to the **Keys** blade and file in the description of keys and the expiration period, as shown in the following screenshot:



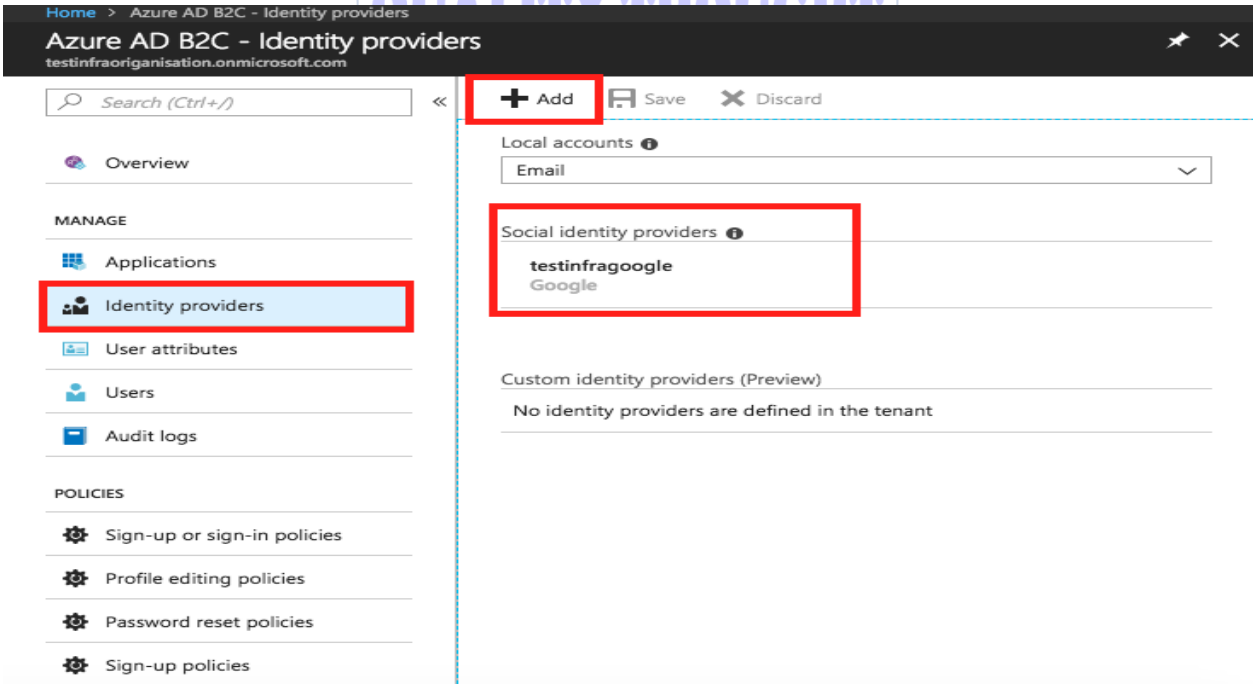
After filling in the description and duration, where you can choose 1 year, 2 years, or never, click on **Save**; after a few seconds, the value of your key will appear, as shown here, you can copy this value to use in the next step:



Here, use your Application ID to fill in **Client ID** and the copied secret in the **Client secret** field, then click on **OK**to finish the setup of the identity provider:



Go back to the **Identity providers** page, where you can see that the Google identity provider has been created successfully, as follows:



Configuring SAML-based SSO for an application with Azure AD

SSO means being able to access all the applications and resources that you need by signing in only once and once, signed in, you can access all of the applications with a single account and without typing in a password for a second time.

Check out the following link to configure SAML-based SSO for an application with Azure AD: https://docs.microsoft.com/en-us/azure/active-directory/manage-apps/configure-single-sign-on-portal.

Managing hybrid identities

At the transition stage of a cloud-first digital transformation, we need to have a cloud-based identity solution that will work in different hybrid scenarios. Azure AD services provide the following capabilities:

- Using **Azure AD Connect** to integrate your on-premises Windows Server Active Directory with Azure AD
- Using **Azure AD Domain Services** (**AD DS**) to provide managed domain services, such as domain join, group policy, LDAP, and Kerberos/NTLM
- Using **Active Directory Federation Services** (**AD FS**) to federate identities and more advanced scenarios, such as SSO or on-premises MFA

Configuring Azure AD Connect and synchronization services

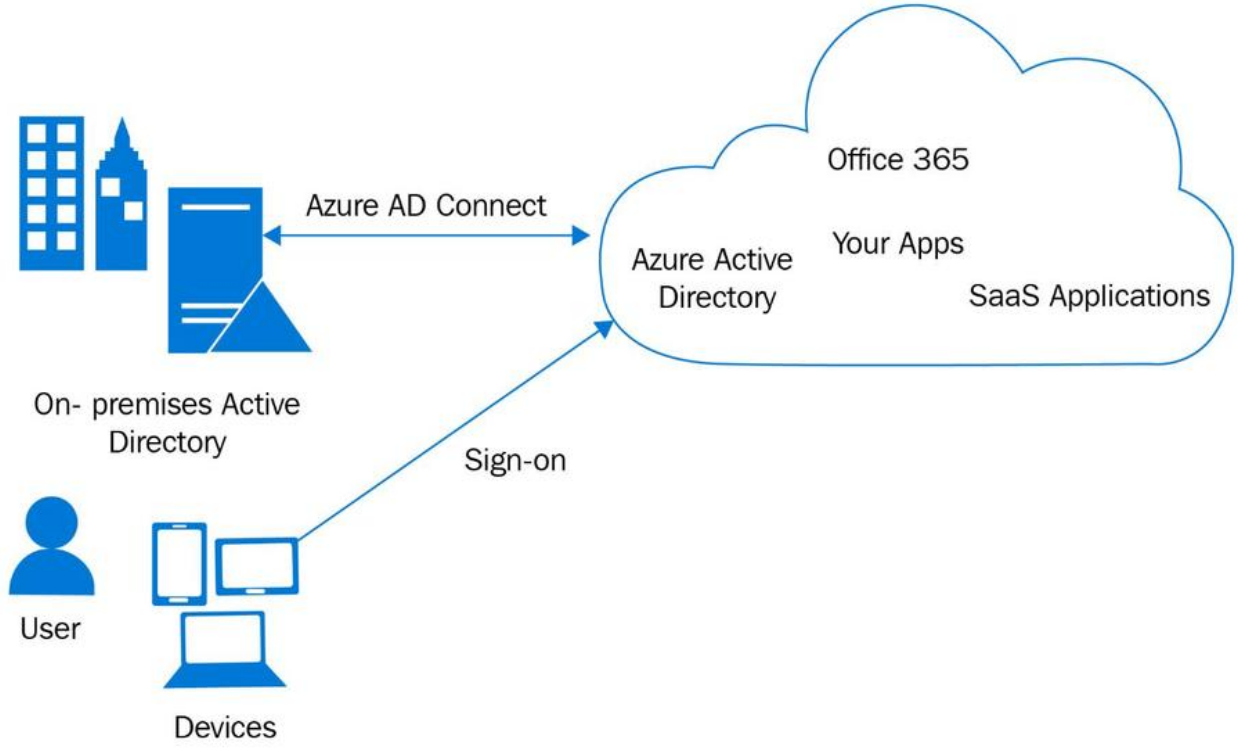
Azure AD Connect is an excellent way to connect the on-premises identity directory with Azure AD and Office 365, as well as the other SaaS applications integrated with Azure AD.

Azure AD Connect is made up of the following three primary components:

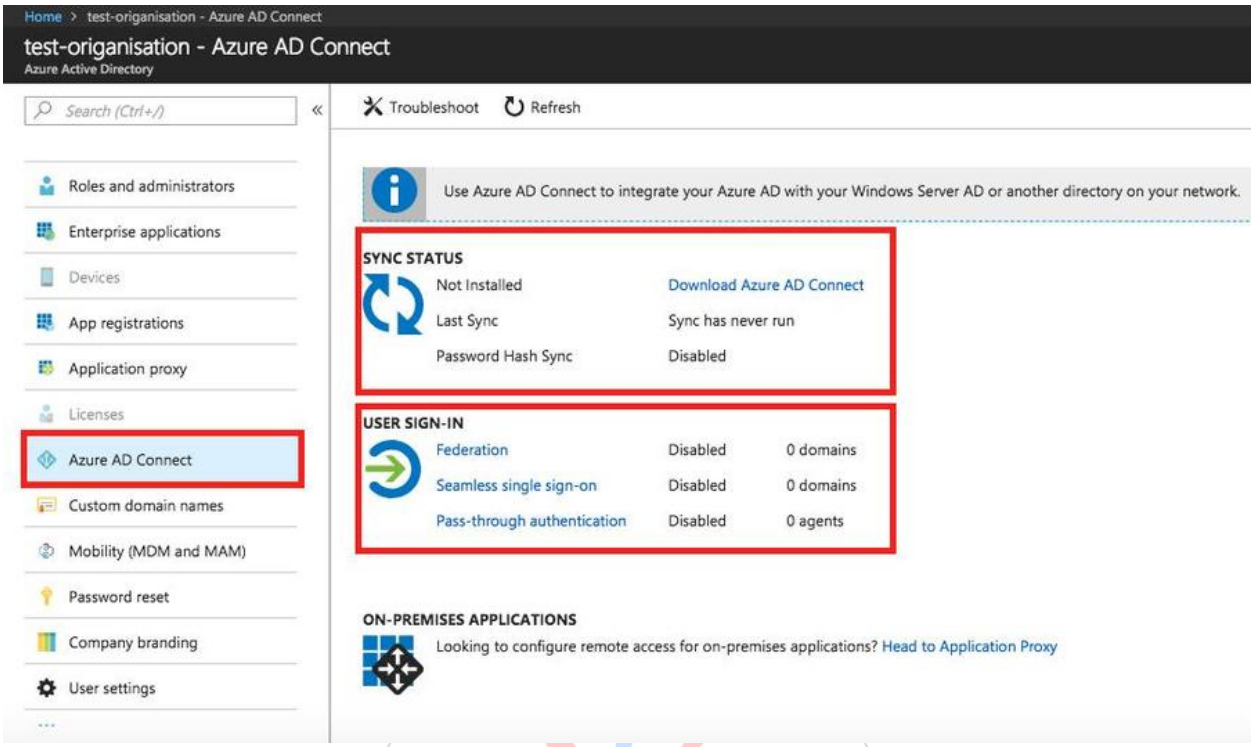
- **Azure AD Connect sync** is an **Azure Active Directory Connect synchronization** services in charge of all operations related to synchronizing identity data between your on-premises environment and Azure AD. It replaces old connectors, such as DirSync and Azure AD Sync.

- **Active Directory Federation Services** component, which is optional and provides the cloud identity federation feature.
- The **Azure AD Connect Health** is a monitoring component to help users gain insights on their on-premises identity infrastructure.

To show how Azure AD Connect can work, is as referred in the following schema, Azure AD connect on-premise Active Directory and Azure Active Directory to provide a friendly identity experience to users:



Go the Azure AD portal and click on the **Azure AD Connect** blade to configure all these features:

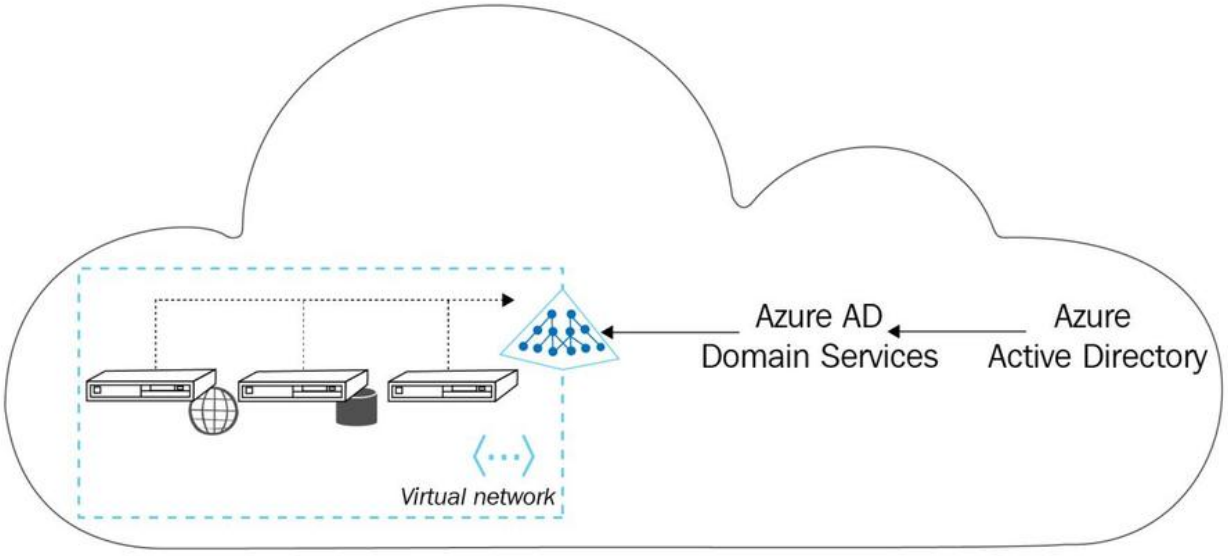


Download AD Connect from the following link: https://www.microsoft.com/en-us/download/details.aspx?id=47594.

To find out more about how to configure Azure Connect's synchronization and federation features, check out the following link: https://docs.microsoft.com/en-us/azure/active-directory/connect/active-directory-aadconnect.

Managing domains with Azure AD domain services

Azure AD domain services provide managed domain services in an Azure virtual network, such as domain join, group policy, LDAP, and Kerberos/NTLM authentication that can integrate Windows Server Active Directory with Azure AD:



AD DS stores information about usernames, passwords, phone numbers, and so on, and guarantees that other users can access this information under authorization.

If you want to join your on-premises Active Directory domain-joined devices to Azure AD, you can accomplish this by referencing the following link to configure hybrid Azure AD joined devices step by step: https://docs.microsoft.com/en-us/azure/active-directory/devices/hybrid-azuread-join-manual-steps.

Before Azure AD DS, we generally use an S2S VPN connection or ExpressRoute to connect on-premises identity server to Azure cloud, or may deploy an Azure VM to run Active Directory. Another way is to use Dir sync to synchronize on-premises identity with Azure cloud. Until now, using Azure AD Connect is still an alternative way, both Azure AD connect and AD DS can most adaptive in different scenarios, you can know more about how to compare them by checking the link below: https://docs.microsoft.com/en-us/azure/active-directory-domain-services/active-directory-ds-compare-with-azure-ad-join.

Implementing SSO in hybrid scenarios

To implement SSO in a hybrid environment, you can use **seamless SSO**; it will need the user's device to be domain-joined. Check out the link https://docs.microsoft.com/en-us/azure/active-directory/connect/active-directory-aadconnect-sso to know how to configure it step by step.

SSO and secure remote access for web applications hosted on-premises can be implemented using **Azure AD Application Proxy**. Azure AD Application Proxy is a lightweight agent that facilitates the flow of traffic from the Application Proxy service in the cloud to your on-premises environment.

To know more about how to enable Azure AD Application Proxy, check out the following link: https://docs.microsoft.com/en-us/azure/active-directory/manage-apps/application-proxy-enable.

Monitoring on-premises identity infrastructure and synchronization services

Azure provides a couple of components to help users monitor their on-premises identity infrastructure and synchronization services such as Office 365 and other Microsoft Online Services.

The key identity components are the following:

- **Active Directory Federation Services** (**AD FS**) servers
- Azure AD Connect servers (which contain a Sync Engine)
- Active Directory domain controllers, and so on.

Monitoring these key components will help an administrator or enterprise architect to make informed decisions. You should download and install Azure AD Connect Health Agents to get health and usage information about your on-premises services.

You can download these agents from the following links:

- Downloading Azure AD Connect Health Agent for AD FS: http://go.microsoft.com/fwlink/?LinkID=518973
- Downloading Azure AD Connect (configures Azure AD Connect Health agent for sync): http://go.microsoft.com/fwlink/?linkid=615771
- Downloading Azure AD Connect Health Agent for AD DS: http://go.microsoft.com/fwlink/?LinkID=820540

You can also get information on configuring monitoring for these components by checking out the following links:

- Monitoring AD FS using Azure AD Connect Health: https://docs.microsoft.com/en-us/azure/active-directory/connect-health/active-directory-aadconnect-health-adfs
- Monitoring Azure AD Connect sync with Azure AD Connect Health: https://docs.microsoft.com/en-us/azure/active-directory/connect-health/active-directory-aadconnect-health-sync
- Using Azure AD Connect Health with AD DS: https://docs.microsoft.com/en-us/azure/active-directory/connect-health/active-directory-aadconnect-health-adds

You can see the views of alerts, performance monitoring, usage analytics, and other information in one place using the Azure AD Connect Health portal. Here is the URL: https://aka.ms/aadconnecthealth.

Planning and Implementing Azure Storage, Backup, and Recovery Services

Microsoft Azure provides different storage capabilities; there are four core storage services such as blobs, tables, queues, and file shares. Otherwise, Microsoft Azure also provides hybrid storage solutions such as StorSimple as well as cross-premise transfer options. It also offers capabilities to facilitate recovery and to assist customers with implementing their **business continuity and disaster recovery** (**BCDR**) strategy using Azure Backup and Azure Site Recovery.

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

- Implementing and managing Azure Storage services
- Configuring Azure **Content Delivery Network** (**CDN**)
- Introducing Azure data storage services

- Implementing the BCDR strategy with Azure Backup and Azure Site Recovery
- Introducing Azure StorSimple and other Azure Hybrid storage

Implementing and managing Azure Storage

Microsoft Azure provides various storage options that allow users to store files, messages, tables, and any other type of information; data stored in Azure Storage can be used by web applications, mobile apps, desktop applications, and various types of custom solution. From a conceptual point of view, Azure Storage options are applied in the following scenarios:

- Object-based storage for virtual machines such as Azure Blobs and file shares
- Semi-structured data storage such as table storage
- Storing or processing large numbers of messages using queue storage
- Hyper-scale repository for big data analytic workloads using Azure Data Lake Store

An overview of Azure Storage services

Azure Storage is a managed service within Azure which aims to provide cloud-based storage, which is secure and scalable, with different levels of availability. Azure Storage includes the following data services:

- **Azure Blobs** is an object-based storage for text and binary data
- **Azure Files** is a managed file shared storage for cloud-based as well as on-premise deployments
- **Azure Queues** stores and exchanges messages between Azure services and applications
- **Azure Tables** is designed for massive semi-structured data or NoSQL storage, which is the cheapest in Azure in comparison with Cosmos DB

Implementing Azure Storage services

To implement Azure Storage services, we should start by creating a storage account. A **storage account** is a unique namespace, where users can store and access data objects in cloud storage.

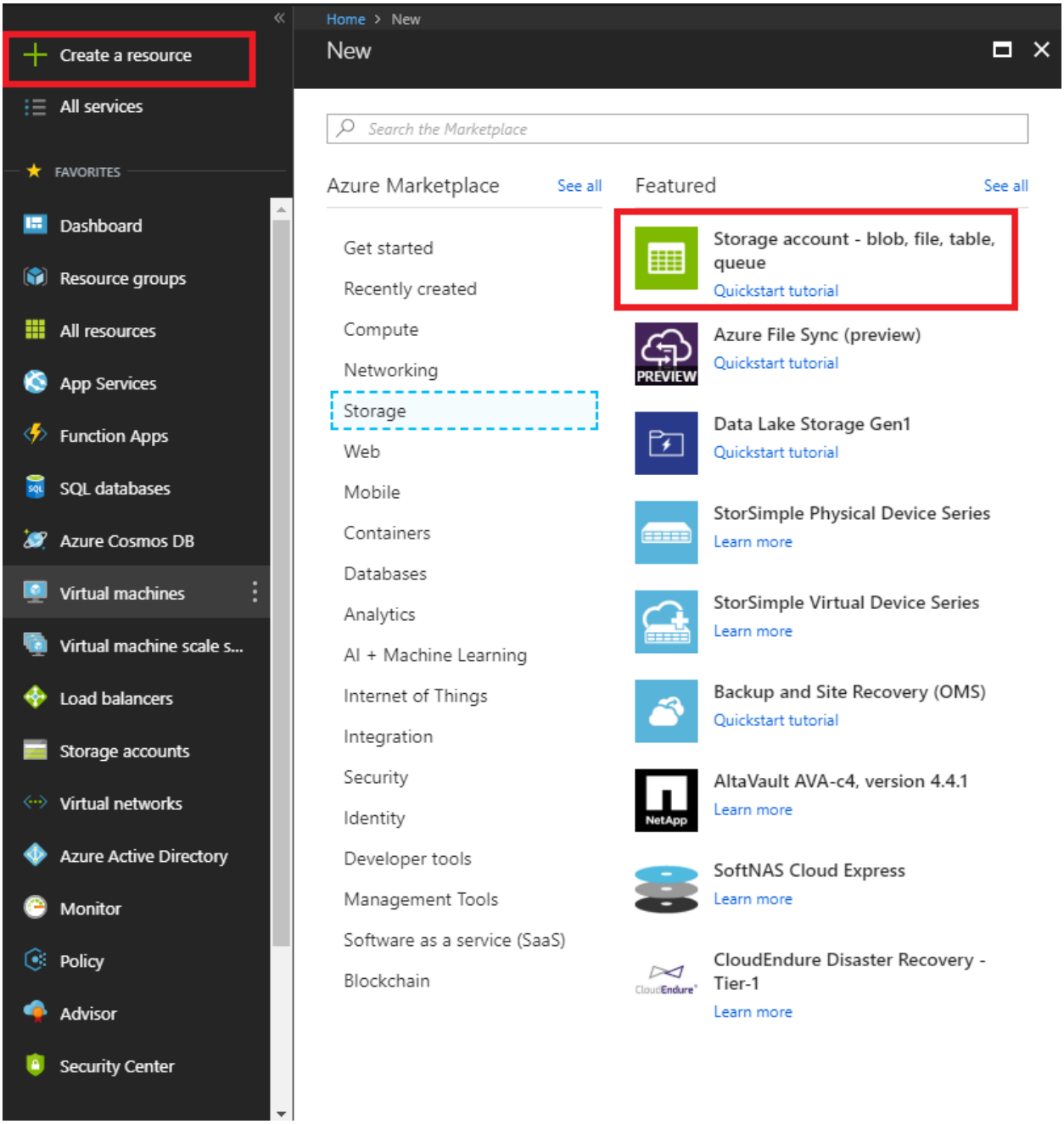
There are three kinds of storage account, which are explained as follows:

- **General-purpose v1** (**GPv1**): This is a unique namespace to host blobs, tables, queues, and files without access tier support
- **General-purpose v2** (**GPv2**): This is a unique namespace to host blobs, tables, queues, and files, and supports new features such as access tiers
- **Blob storage**: This is a unique namespace to host blobs

The general quota for storage accounts is limited to 200 in a single Azure subscription. Users can request to increase the quota by contacting Azure support.

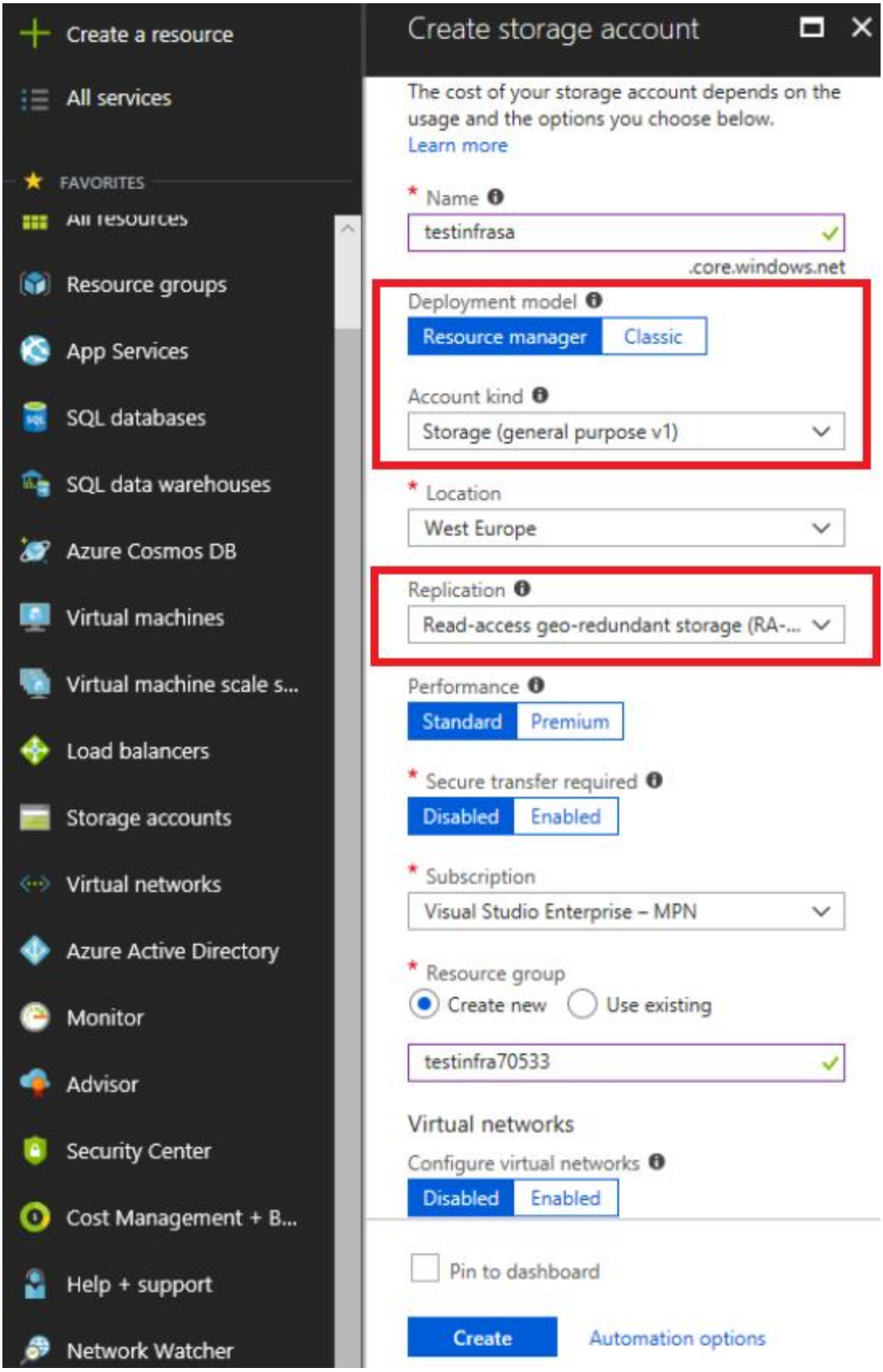
Creating a storage account

To create a storage account via the Azure portal, click on **Create a resource**, then you'll find **Storage account**, as shown in the following screenshot:



Click on **Storage account,** fill in a **Name,** and choose a type of storage account that is compatible with Azure resource in classic model and the Azure Resource Manager model.

You can choose a type of storage account and set a replication method for the storage as shown in the following screenshot:

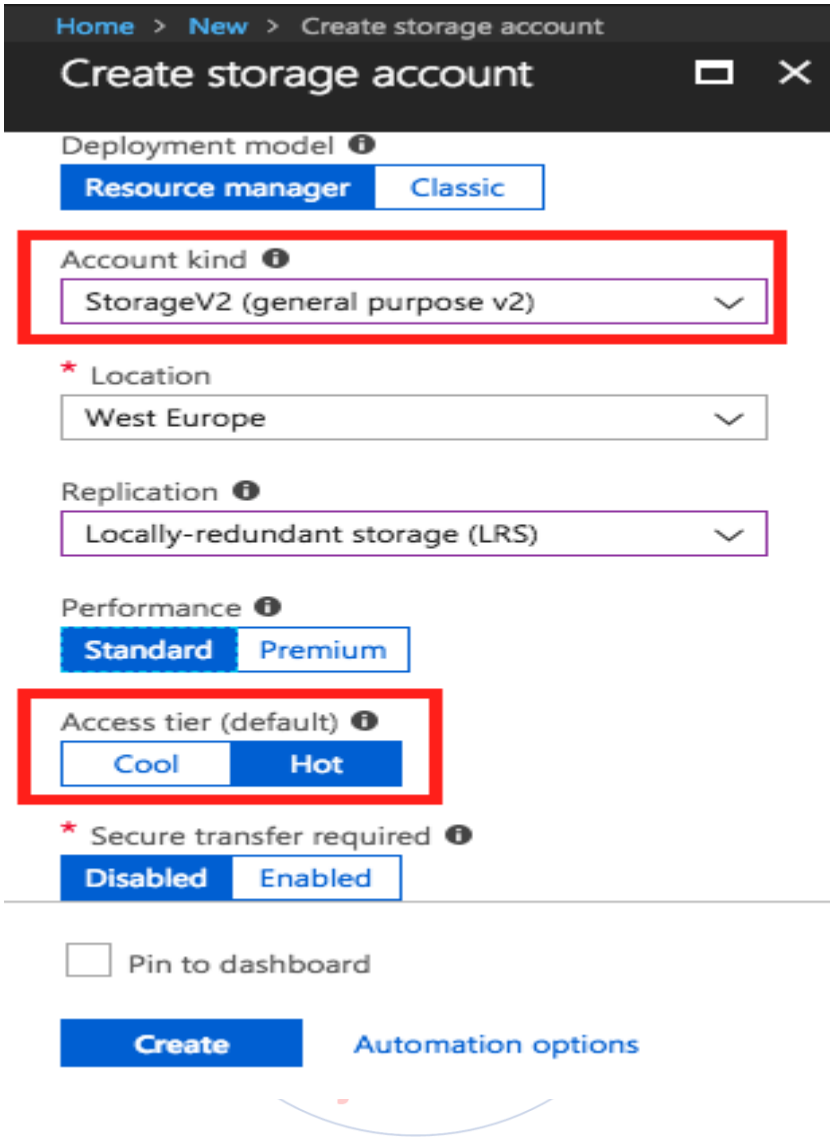

Create a resource
All services
FAVORITES
All resources
Resource groups
App Services
SQL databases
SQL data warehouses
Azure Cosmos DB
Virtual machines
Virtual machine scale s...
Load balancers
Storage accounts
Virtual networks
Azure Active Directory
Monitor
Advisor
Security Center
Cost Management + B...
Help + support
Network Watcher
Create storage account
The cost of your storage account depends on the usage and the options you choose below.
Learn more
* Name
testinfrasa
.core.windows.net
Deployment model
Resource manager
Classic
Account kind
Storage (general purpose v1)
* Location
West Europe
Replication
Read-access geo-redundant storage (RA-...
Performance
Standard
Premium
* Secure transfer required
Disabled
Enabled
* Subscription
Visual Studio Enterprise – MPN
* Resource group
Create new
Use existing
testinfra70533
Virtual networks
Configure virtual networks
Disabled
Enabled
Pin to dashboard
Create
Automation options

There are currently four replication methods in Azure Storage:

- **Locally-redundant storage** (**LRS**): The target is to provide at least 99.999999999% (11 times 9) durability of objects over a given year by replicating your data within a datacenter in the region. It is supported by the GPv1, and GPv2, blob storage account types.
- **Zone-redundant storage** (**ZRS**): The target is to offer durability for storage objects of at least 99.9999999999% (12 times 9) over a given year; this option replicates data in the storage account synchronously across three storage clusters, which is physically separated from the others and resides in its own **availability zone** (**AZ**) in a single region.
- **Geo-redundant storage** (**GRS**): This is designed to provide at least 99.99999999999999% (16 times 9) durability of objects over a given year; this option replicates data to a secondary region in the case of failure of the primary region, which is not recoverable from a regional outage or a disaster. However, the data in the secondary region is available to be read only when Microsoft initiates a failover from the primary to the secondary region.
- **Read-access geo-redundant storage** (**RA-GRS**): This is the same as GRS, but provides read-only access to the data in the secondary location as well as the primary region. This means users can read from the secondary region regardless of whether Microsoft initiates a failover from the primary to the secondary region.

Note that the primary region for the account is chosen by users while creating a new storage account; however, the paired secondary region cannot be changed, and is determined by Microsoft. Check the following link to know more about paired regions in Azure: https://docs.microsoft.com/en-us/azure/best-practices-availability-paired-regions.

You can choose a **Replication** method that is most compatible with your requirements. If you choose the GPv2 account type, you can choose the access tier type as shown in the following screenshot:



Choosing access tier for the storage account

There are two types of access tier, based on the data access frequency:

- **Hot access tier**: This is for data most frequently accessed.
- **Cool access tier**: This is for data less frequently accessed.
- **Archive access tier**: This is optimized for storing data that is rarely accessed. It can be stored for 180 retention days. Archive storage has the lowest storage costs but involves more costs to retrieve the data.

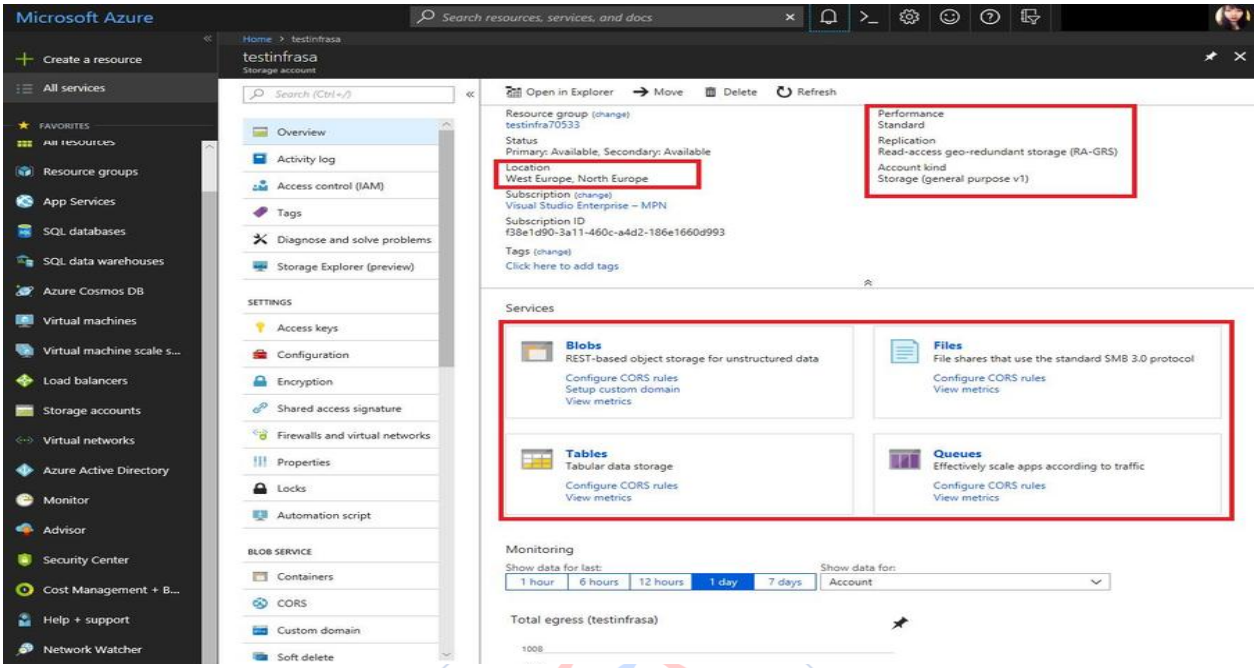
The access tiers are designed for storing data at a lower cost. Data residing in this account will automatically inherit that access tier setting. The hot and cool storage tiers can be set **at the account level;** archival storage can be set **at the object level** as well as the two others.

The general storage account has two performance tiers:

- **Standard storage performance tier**: This hosts Tables, Queues, Files, Blobs and Azure VM disks.
- **Premium storage performance tier**: This currently only supports Azure VM disks. The performance settings can't be changed after the storage account is created. Note that Azure VMs, which use premium storage for all disks, will be guaranteed a 99.9% SLA.

The secure transfer required setting means, after enabling this setting, all the requests to the storage account should be considered as secure connections, otherwise, it would be rejected by Azure. As an example, when this setting is enabled, any requests using HTTP will be rejected when you're calling the data in the blob storage by programming methods using the Azure RESTful API.

After filling in all the information, you can click on **Create**. The deployment usually takes a few seconds. After creating the storage account successfully, you can go to your storage account (as shown in the following screenshot):



In the **Overview** blade, you can find the information related to your storage account, including **Resource group**, **Location**, **Performance** tier, **Replication**, and **Account kind**.

You can also create a storage account using Azure PowerShell as explained in the following link: https://docs.microsoft.com/en-us/azure/storage/blobs/storage-samples-blobs-powershell.

To get more information about creating a storage account using the Azure CLI, go to: https://docs.microsoft.com/en-us/azure/storage/blobs/storage-samples-blobs-cli.

```
{
  "name": "string",
  "type": "Microsoft.Storage/storageAccounts",
  "apiVersion": "2018-10-01",
  "sku": {
    "name": "string"
  },
  "kind": "string",
  "location": "string",
  "tags": {},
  "identity": {
    "type": "SystemAssigned"
  },
  "properties": {
```

```
    "customDomain": {
      "name": "string",
      "useSubDomain": boolean
    },
    "encryption": {
      "services": {
        "blob": {
          "enabled": boolean
        },
        "file": {
          "enabled": boolean
        }
      },
      "keySource": "string",
      "keyvaultproperties": {
        "keyname": "string",
        "keyversion": "string",
        "keyvaulturi": "string"
      }
    },
    "networkAcls": {
      "bypass": "string",
      "virtualNetworkRules": [
        {
          "id": "string",
          "action": "Allow",
          "state": "string"
        }
      ],
      "ipRules": [
        {
          "value": "string",
          "action": "Allow"
        }
      ],
      "defaultAction": "string"
    },
    "accessTier": "string",
    "supportsHttpsTrafficOnly": boolean
  }
}
```

You can also find an example in the Azure quickstart template repository: https://github.com/Azure/azure-quickstart-templates/tree/master/101-storage-account-create.

Implementing Azure Blob storage

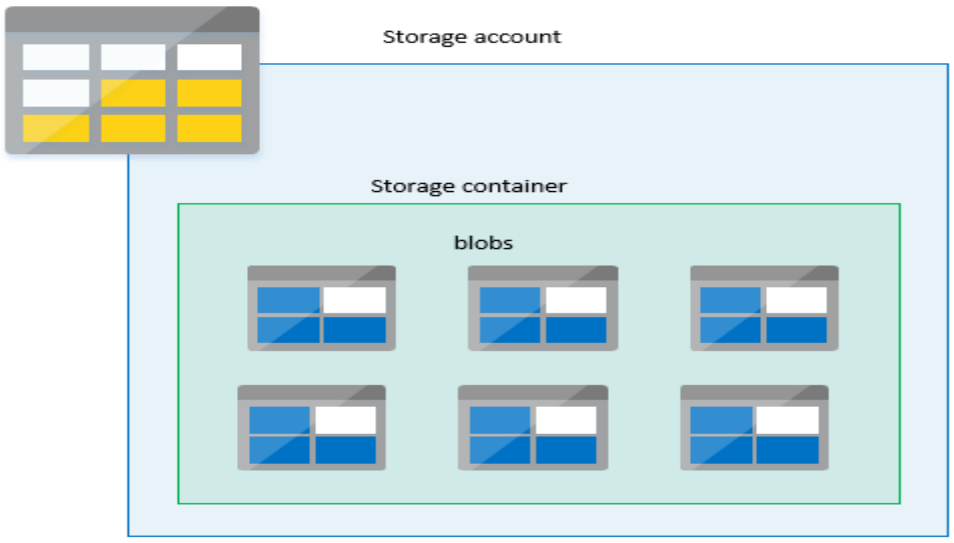
Azure Blob storage is an object-based storage solution in Microsoft Azure. To implement an Azure Blob, you should create a storage account, then a container, and finally the blob.

There are three types of blob, which are listed as follows:

- **Block blobs**: These are designed for storing text and binary data
- **Append blobs**: These are optimized for append operations, ideal for logs
- **Page blobs**: These are designed to store VHD files for Azure VMs

A container acts as an organizer, and contains a set of blobs. All the blobs reside within a container. There is no limit to the number of containers within the storage account and also no limit on the number of blobs in the containers.

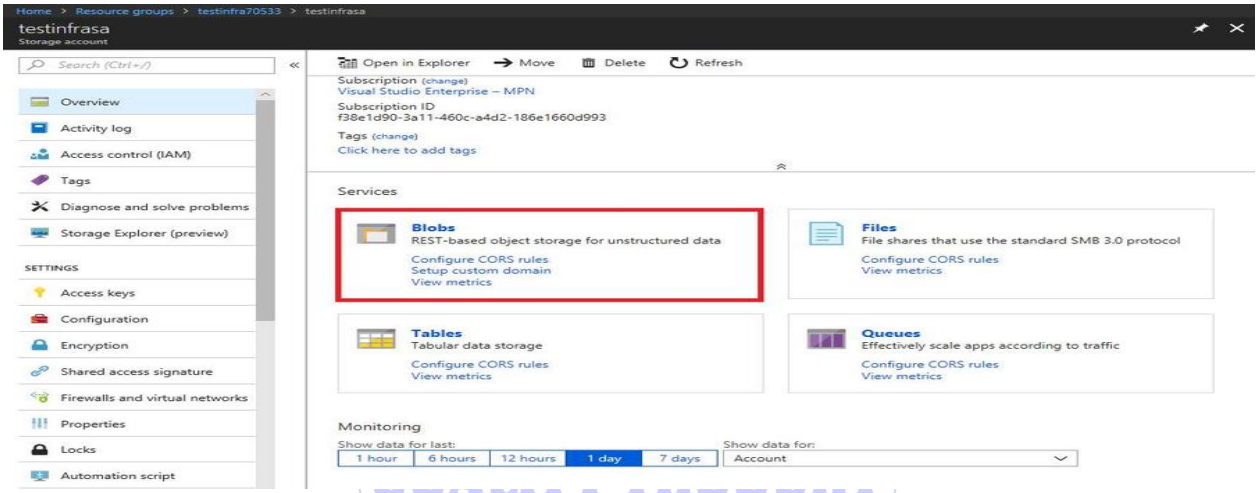
You can define the relationship between **Storage account**, **Storage container,** and **blobs** using the following schema:



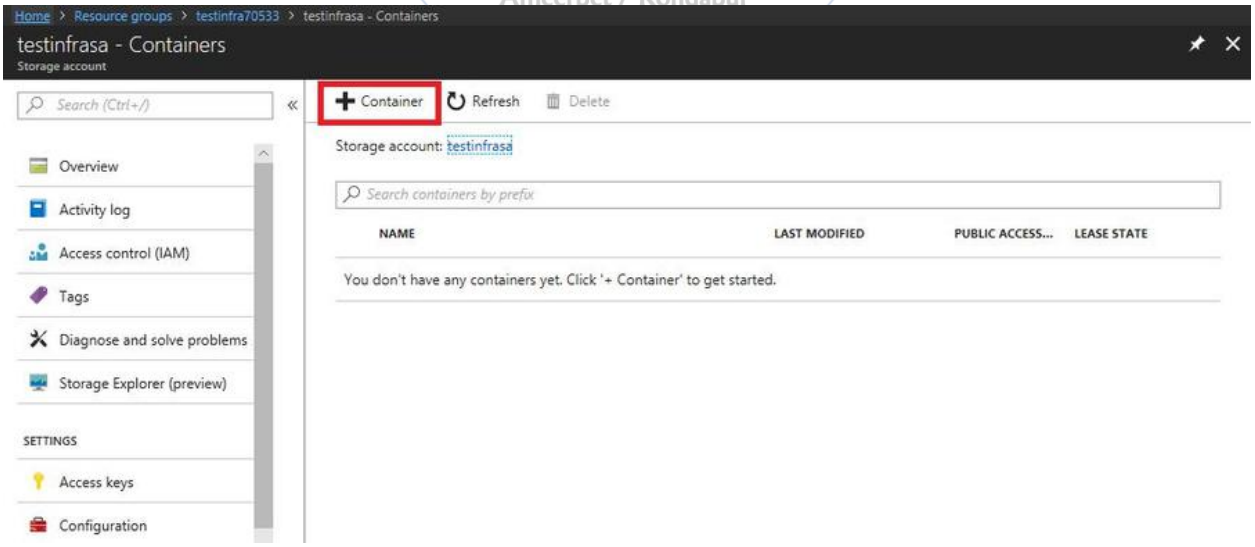
The relationship between resources

A storage account may have one root container, which acts as a default container for the current storage account. The root container is created by default and is named $root. A text file resides in the root container and can be referenced in the following manner: https://storageaccountname.blob.core.windows.net/blob.txt.

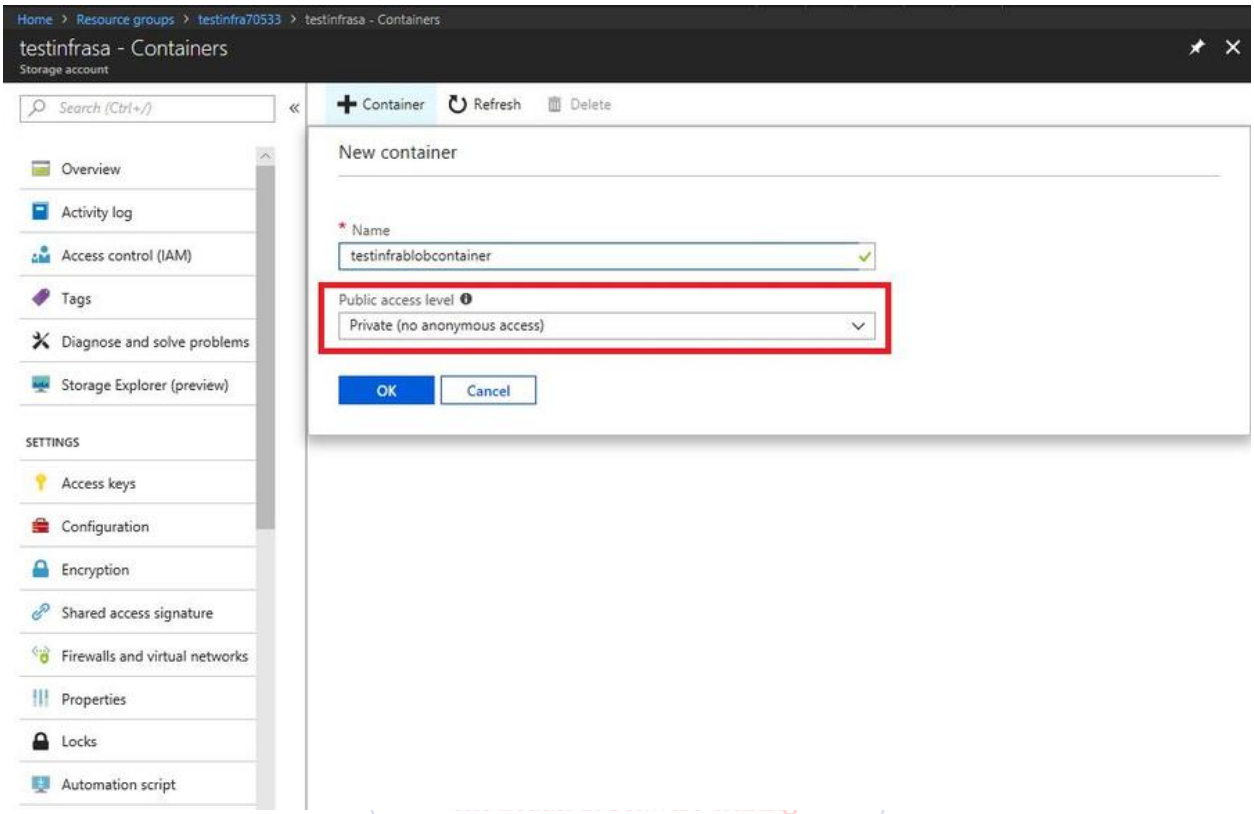
To create a new blob, go to your storage account and click on **Blobs** as shown in the following screenshot:



Then, click on **+ Container** to create a new container of your blob as shown in the following screenshot:



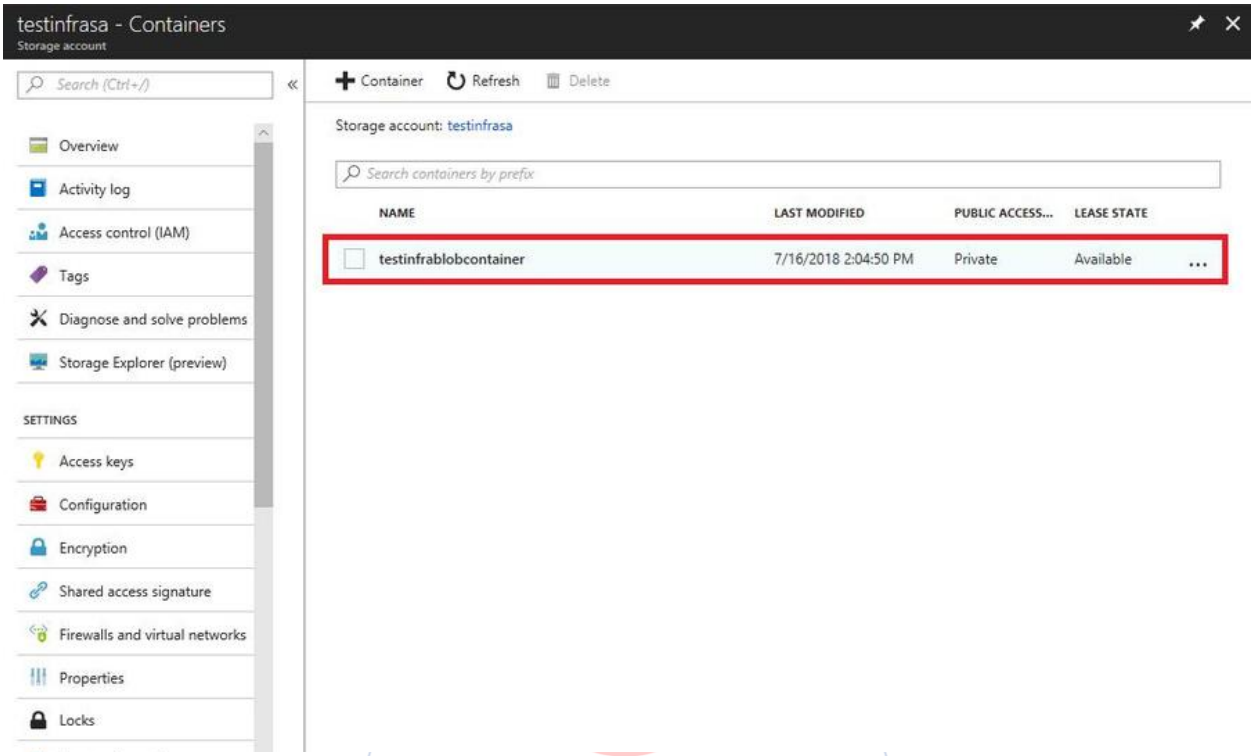
To create a new container, you should choose a permission. By default, a container is set to Private (no anonymous access), which means all the blobs within the container can only be accessed by the storage account owner (as shown in the following screenshot):



However, there are three permissions that are required to configure a container, as follows:

- **Private**: This is used to set all the blobs that can be accessed only by the storage account owner
- **Blob**: This allows anonymous read access within the container, but not for container data
- **Container**: This allows anonymous read access to containers and blobs

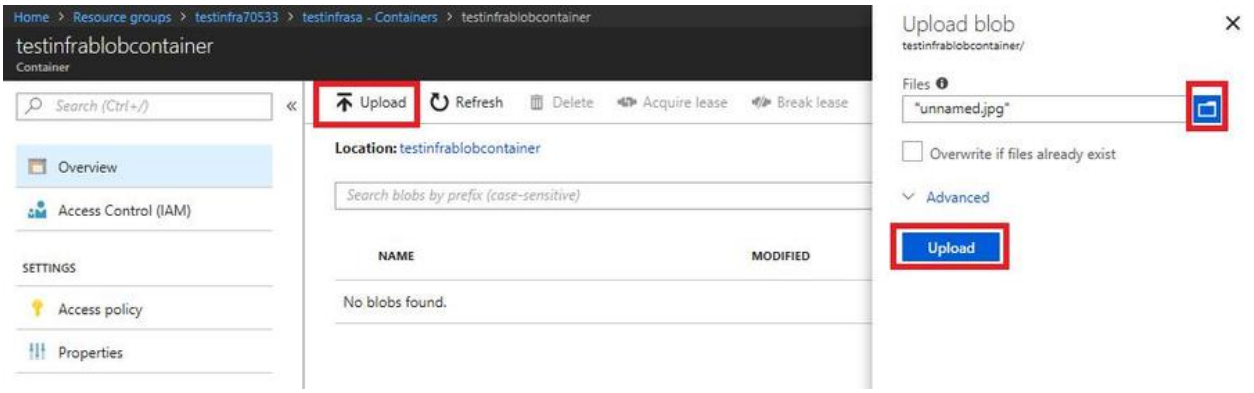
After clicking on **OK**, you'll see that a container with private access has been created in the storage account.



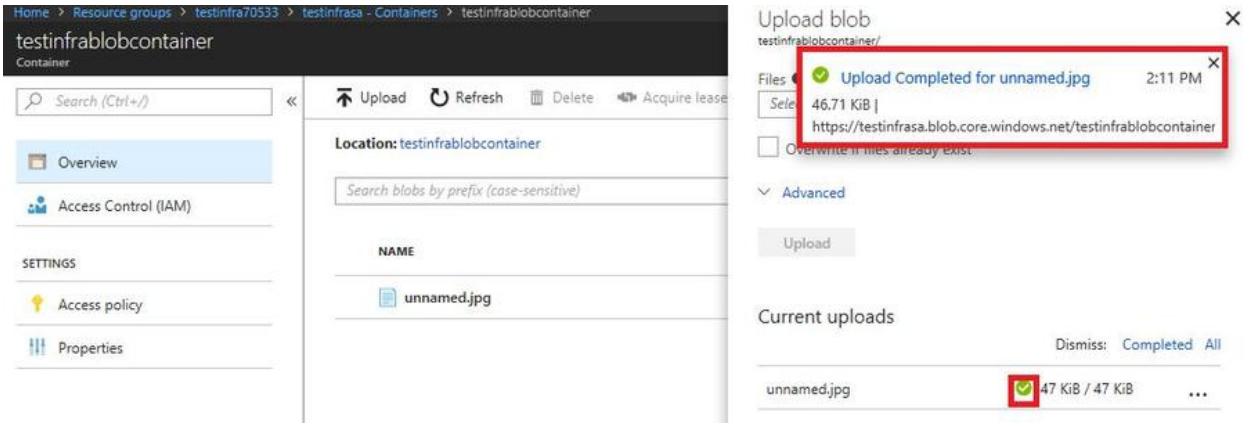
Managing Azure Blob storage

Go to the container to manage your blob storage.

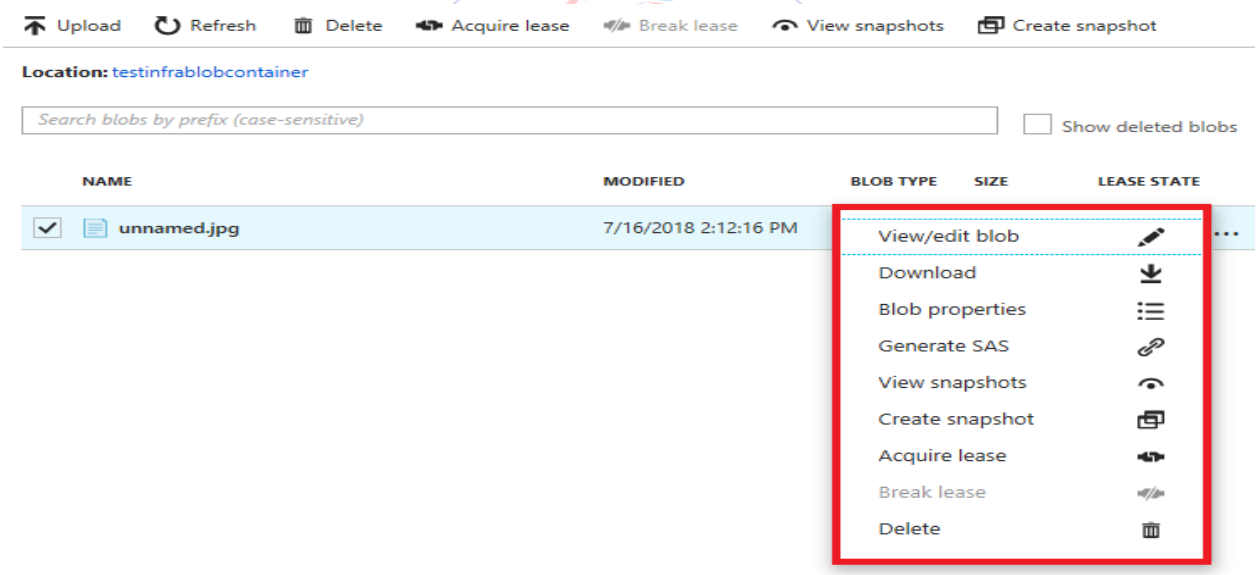
To add a blob, you can click on **Upload** in the **Overview** blade; a popup will be displayed. Then, click on the file icon to choose a file in your PC and click on **Upload**, as shown in the following screenshot:



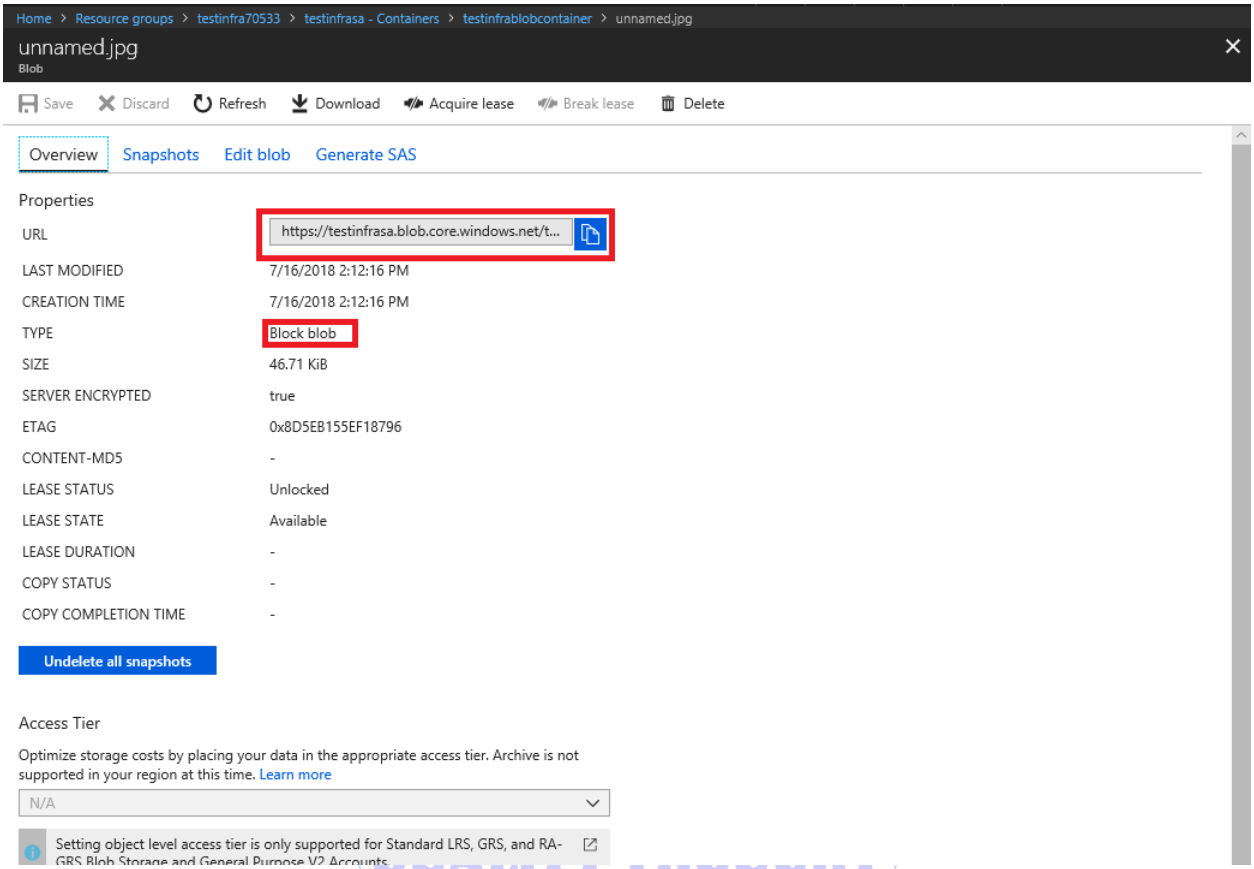
After clicking on **Upload**, you can see that your blob has been uploaded successfully:



You can also use other management functions provided by Azure by right-clicking on the blob (as shown in the following screenshot):



Click on **Blob properties**; you can find the endpoint of the blob as shown in the following screenshot:

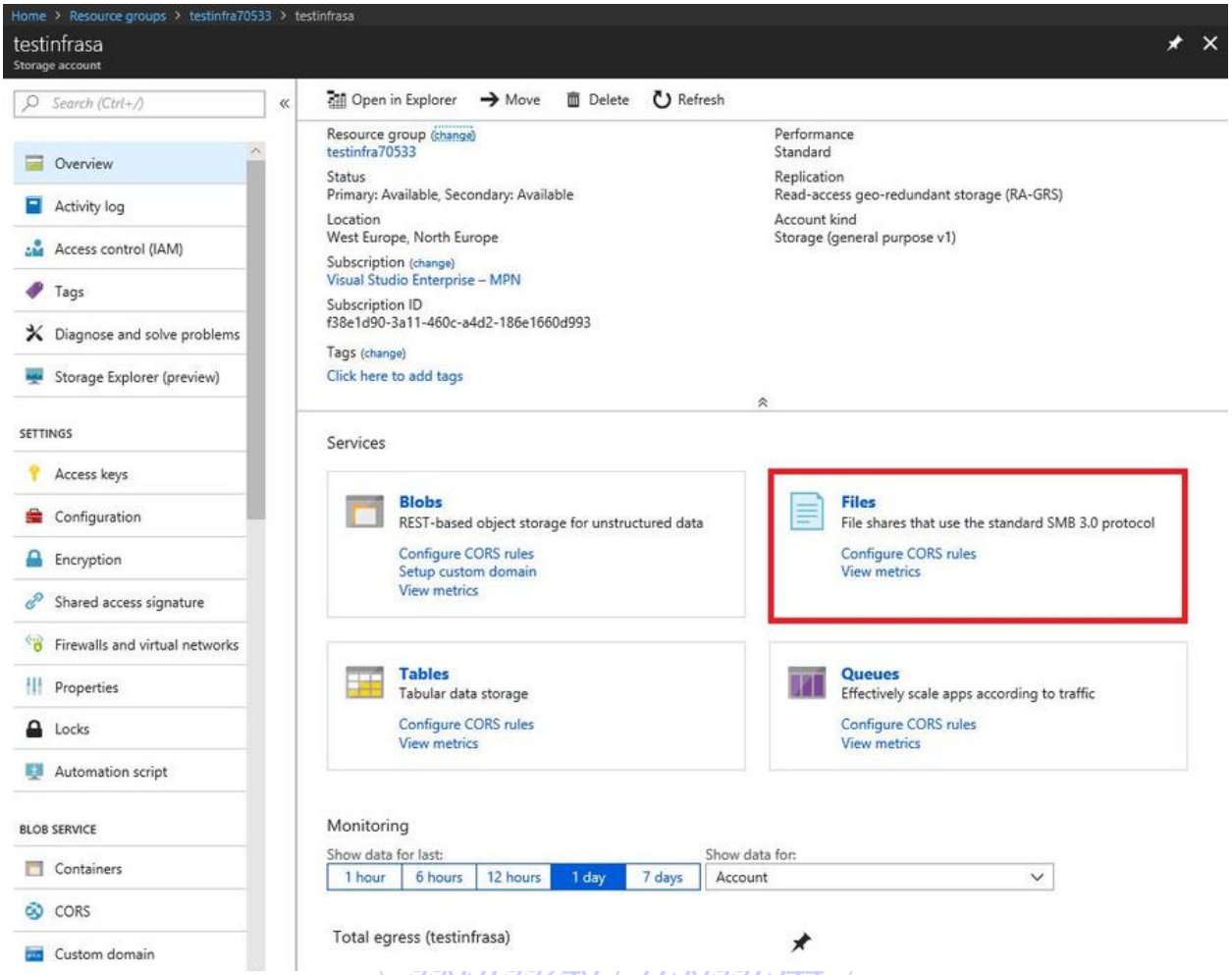


The default endpoints for blobs in the storage account are as follows: http://storageaccountname.blob.core.windows.net.

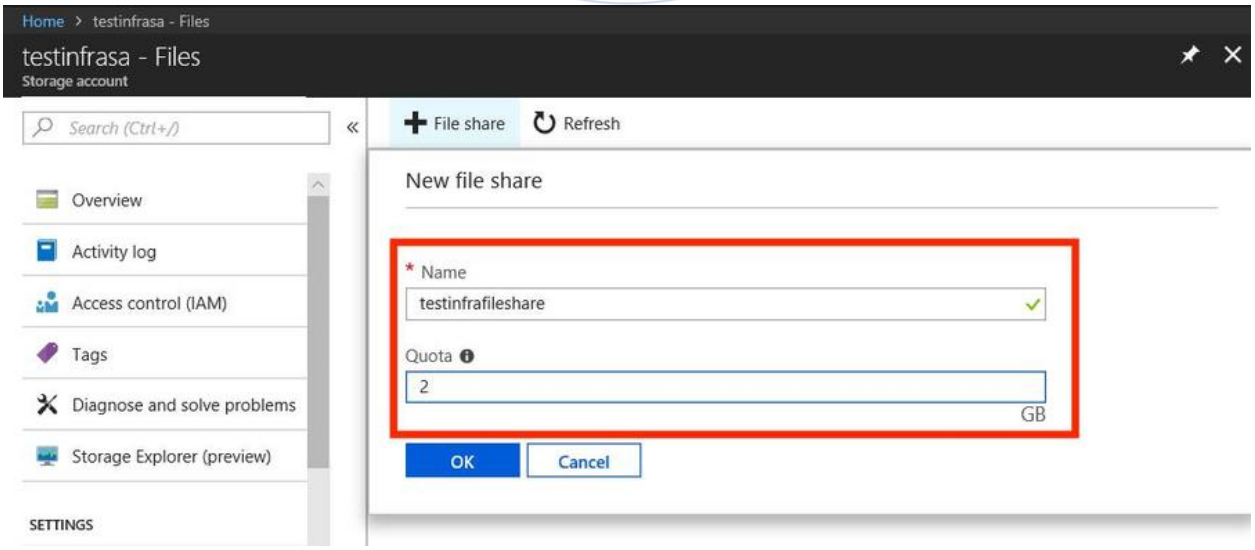
Implementing Azure Files storage

Azure Files provides managed file shares in Azure cloud, which is accessible via the **Server Message Block** (**SMB**) protocol. It is possible to be mounted as a file share by Azure VMs in the cloud or even VMs on-premise. Azure files can be used by different operating systems such as Windows, Linux, and macOS.

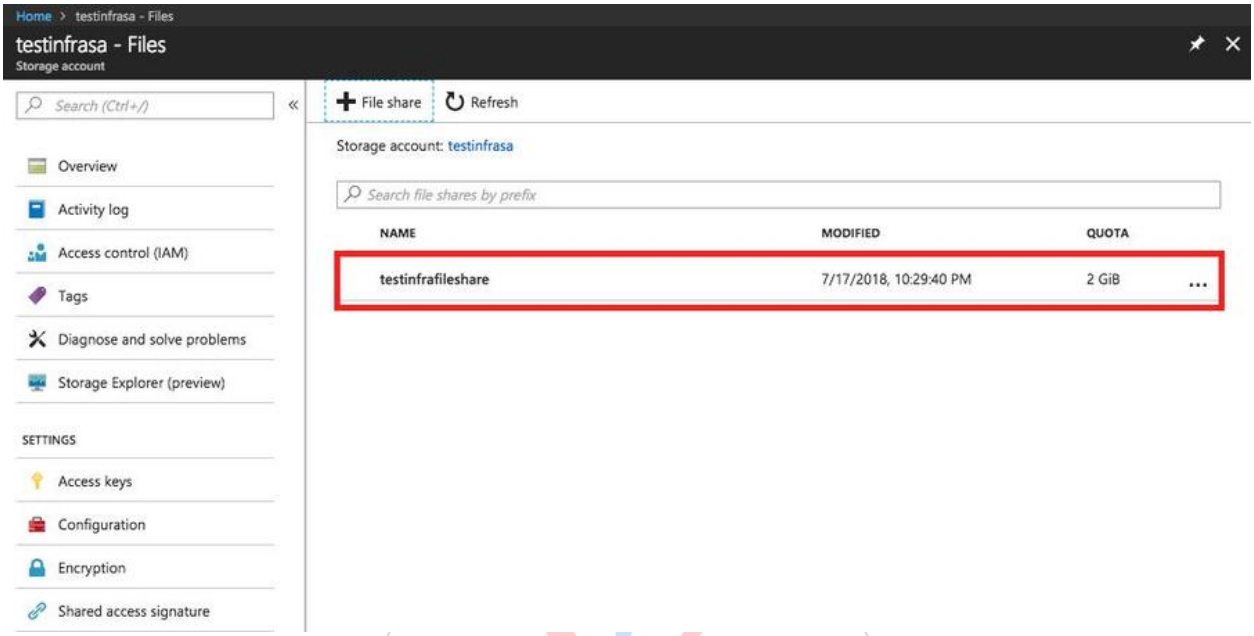
To create a file share, go to the storage account and click on **Files** as shown in the following screenshot:



Then, click on **+ File share** to add a new file share. You should provide the name of the file share quota, which is limited to 5120 GB, as shown in the following screenshot:



After a few seconds, you'll see that a file share has been deployed successfully, as shown in the following screenshot:



To create a file share through the Azure CLI, you can refer to the following link: https://docs.microsoft.com/en-au/azure/storage/files/storage-how-to-create-file-share#create-file-share-through-command-line-interface-cli.

To create file share through PowerShell, you can refer to the following link: https://docs.microsoft.com/en-au/azure/storage/files/storage-how-to-create-file-share#create-file-share-through-powershell.

Azure File share can be used by different operating systems such as Windows, Linux, and macOS.

For Windows, you can refer to: https://docs.microsoft.com/en-au/azure/storage/files/storage-how-to-use-files-windows.

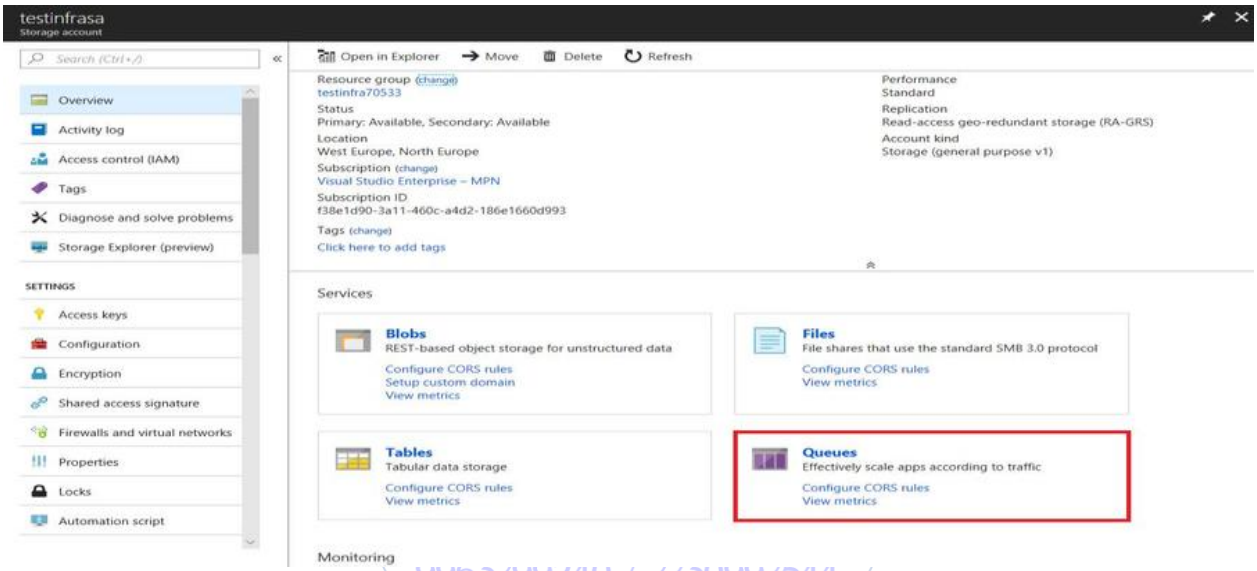
For Linux, you can refer to: https://docs.microsoft.com/en-au/azure/storage/files/storage-how-to-use-files-linux.

For MacOS, you can refer to: https://docs.microsoft.com/en-au/azure/storage/files/storage-how-to-use-files-mac.

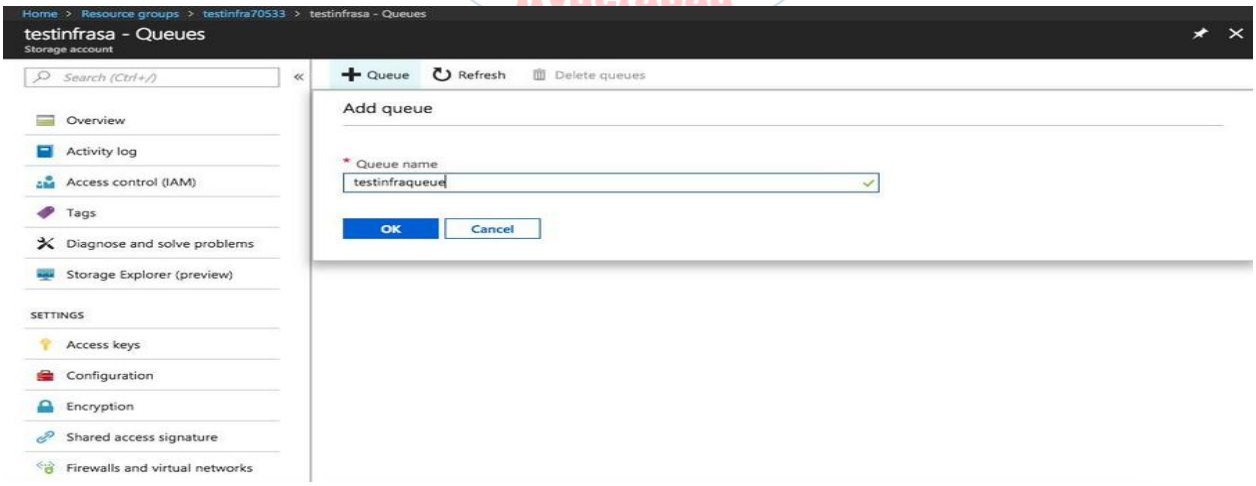
The default endpoints for the blobs in the storage account are as follows: https://#youstorageaccountname#.file.core.windows.net/#fiesharename#.

Implementing Azure Queue storage

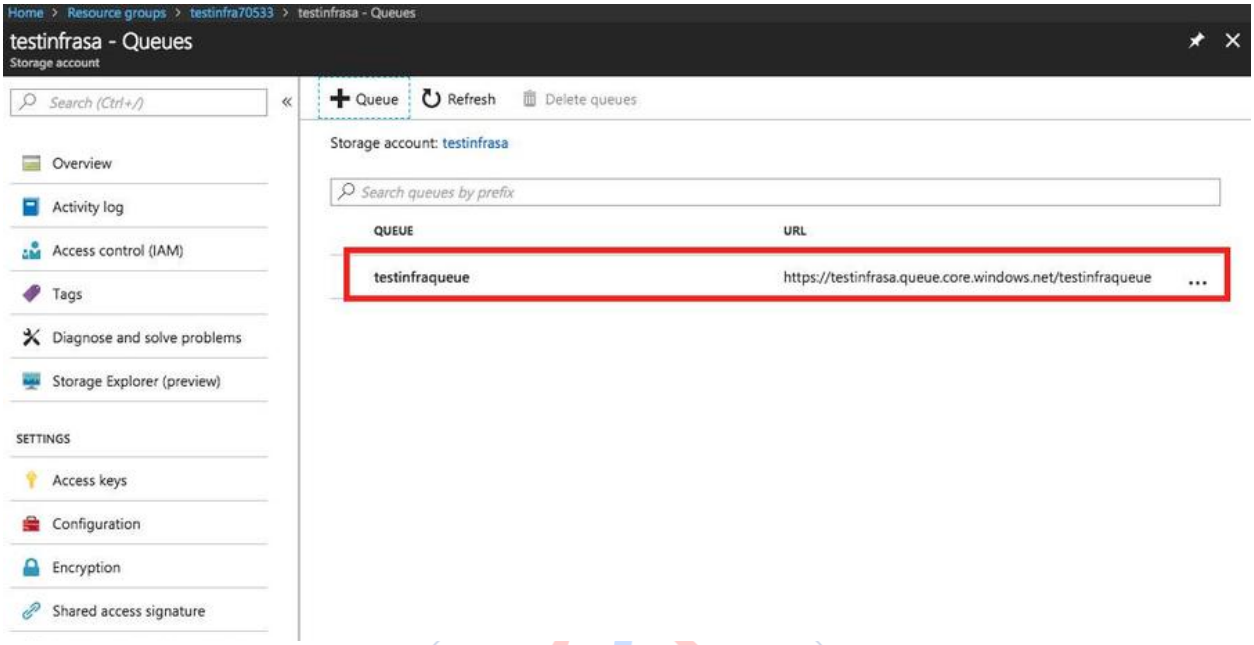
Azure Queue storage is designed for storing large numbers of messages using the HTTP or HTTPS protocols. To create a queue storage, go to the storage account and click on **Queues**, as shown in the following screenshot:



Then, click on **+ Queue** to add a new queue storage, as shown in the following screenshot:



Click **OK.** After a few seconds, you'll find that a queue storage has been deployed successfully, as shown in the following screenshot:



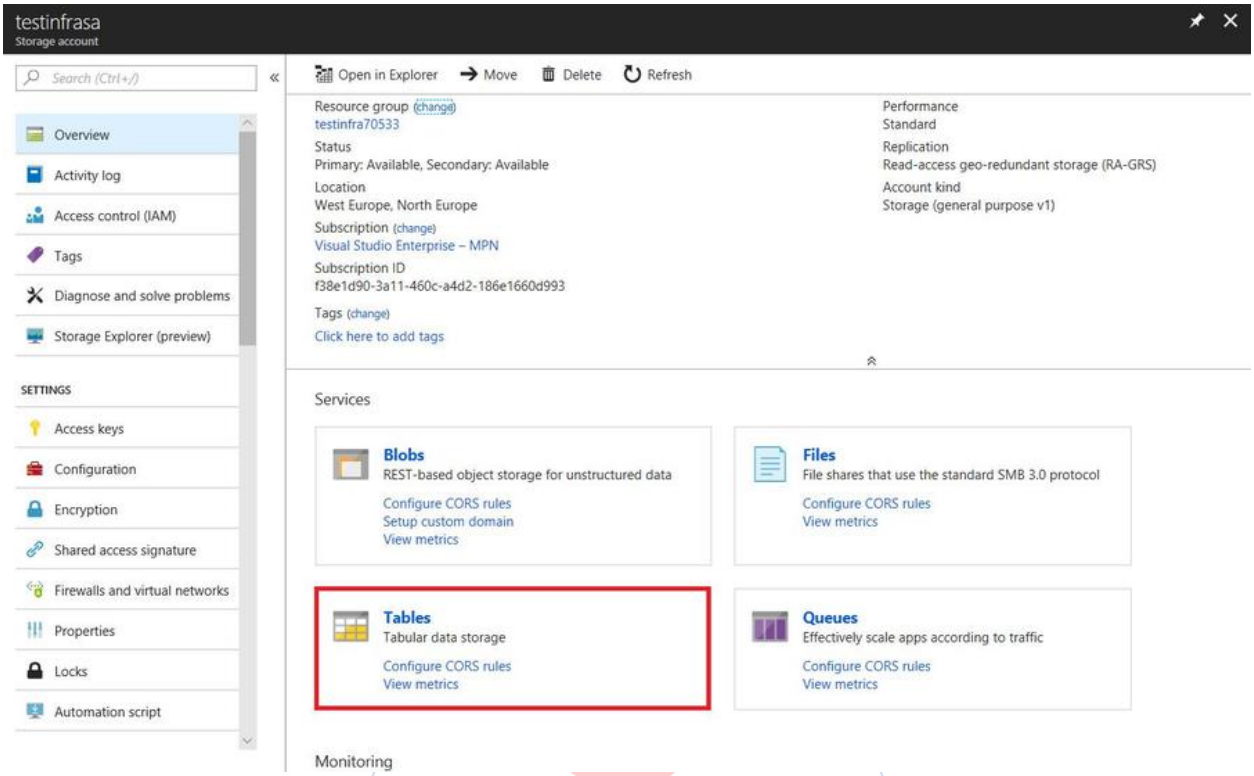
A new queue storage has been created successfully

The default endpoints for the file share in the storage account are as follows: https://#youstorageaccountname#.queue.core.windows.net/#queuename#.

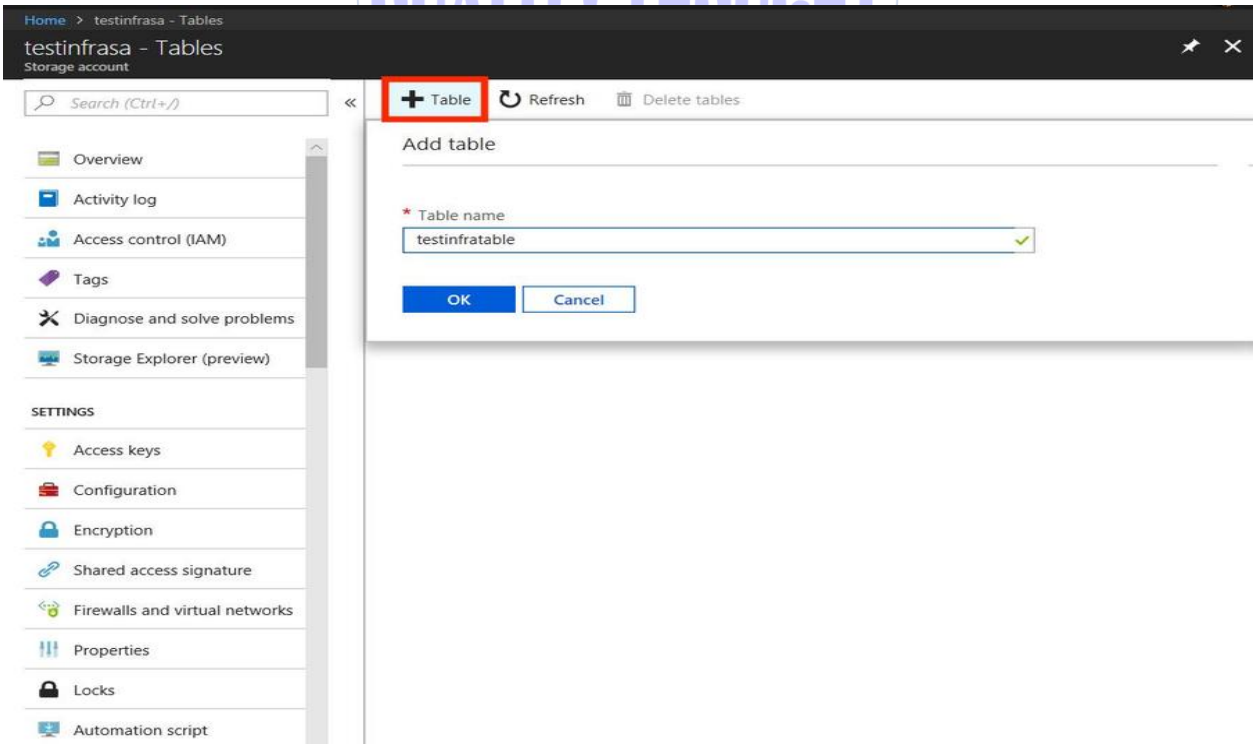
For a comparison between Storage queues and Service Bus queues, refer to: https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-azure-and-service-bus-queues-compared-contrasted.

Implementing Azure Table storage

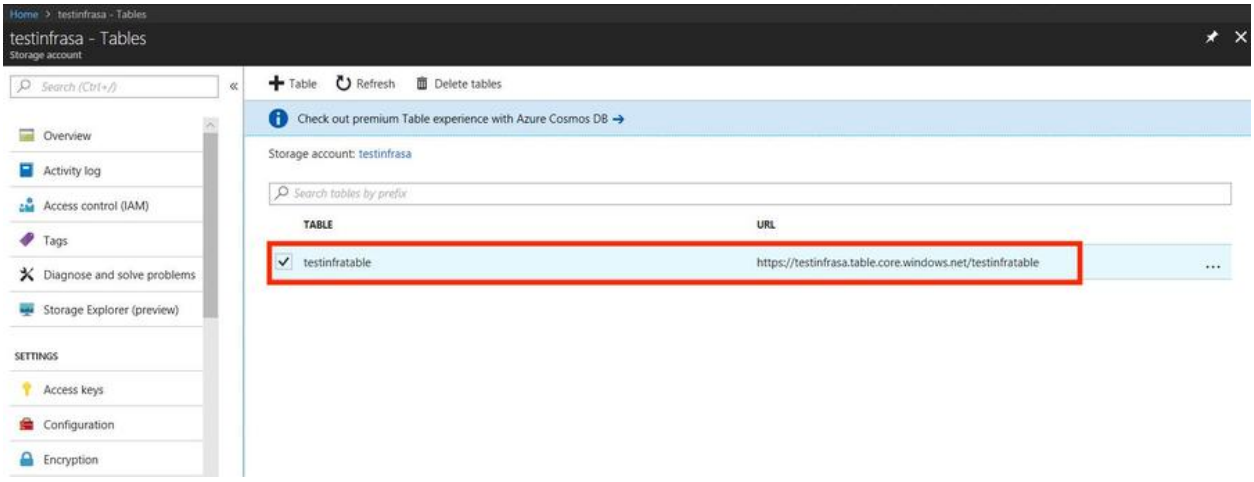
Azure Table storage is designed to store schema-less or NoSQL data in the cloud; it provides a key/attribute store. To create a table storage, go to the storage account and click on **Tables**, as shown in the following screenshot:



Then, click on **+ Table** to add a new table storage, as shown in the following screenshot:



After a few seconds, you'll find that a table storage has been deployed successfully as shown in the following screenshot:

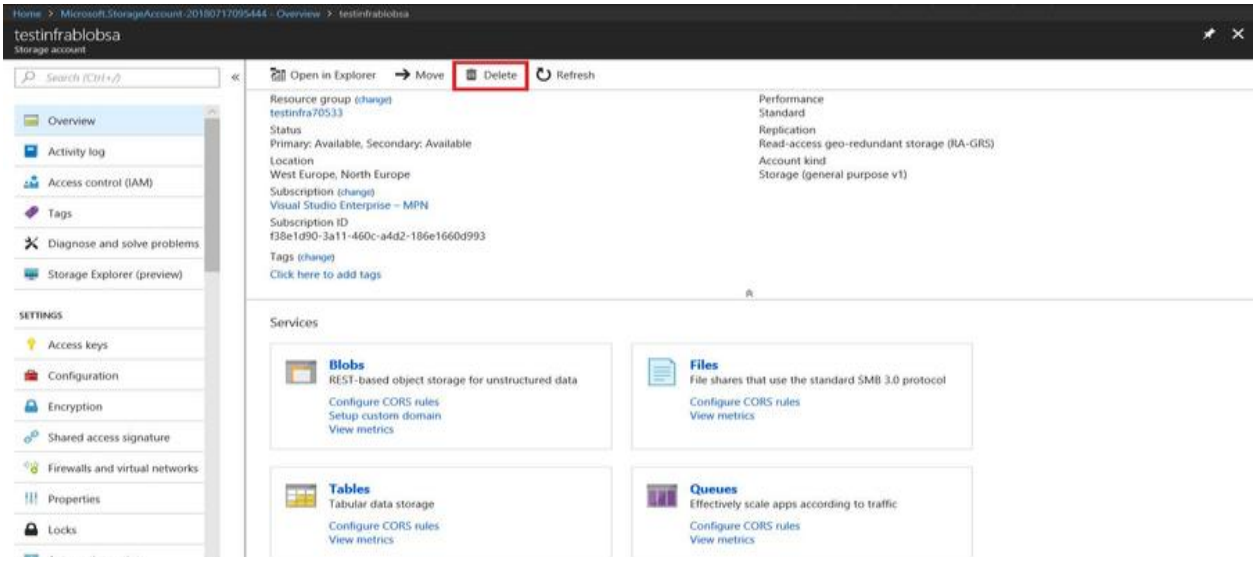


Azure Table storage accounts use the following URL: http://#storageaccountname#.table.core.windows.net/#tablestoragename#.

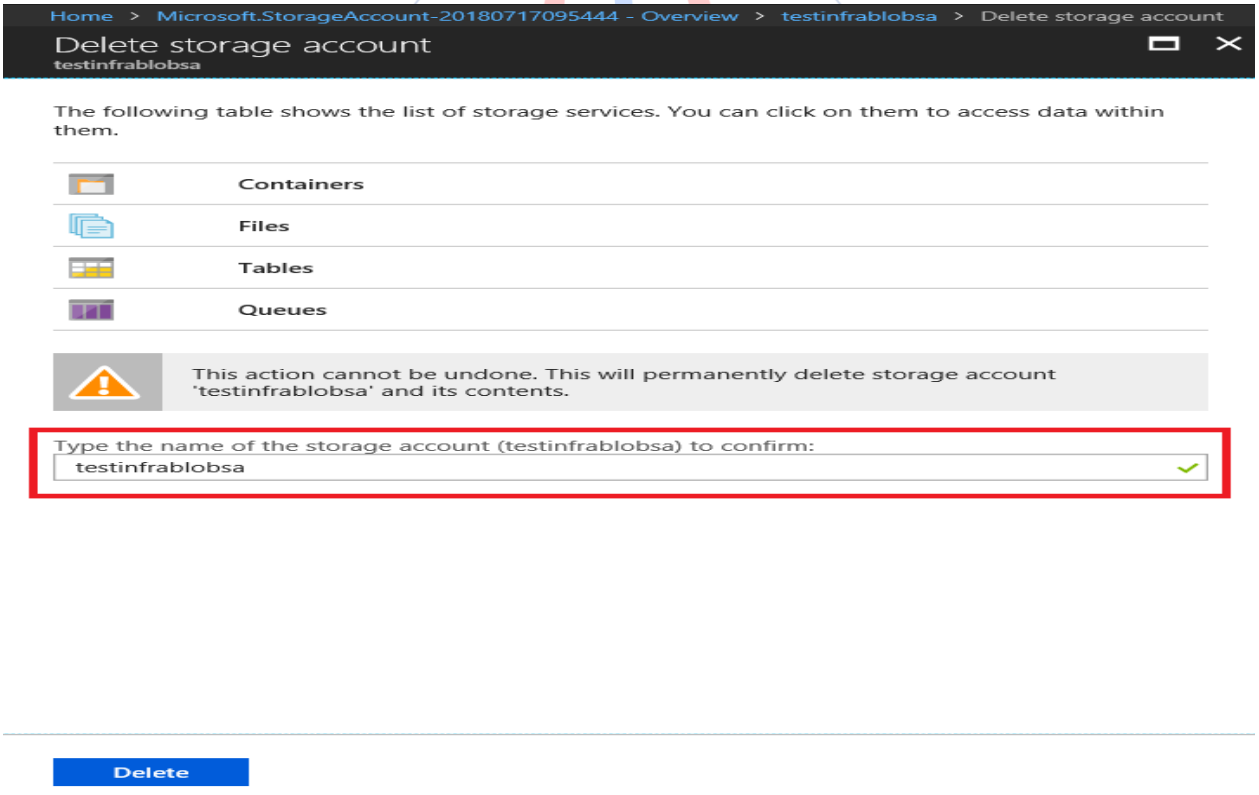
Azure Cosmos DB Table API accounts use the following URL: http://#storageaccountname#.table.cosmosdb.azure.com/#tablestoragename#.

Deleting a storage account

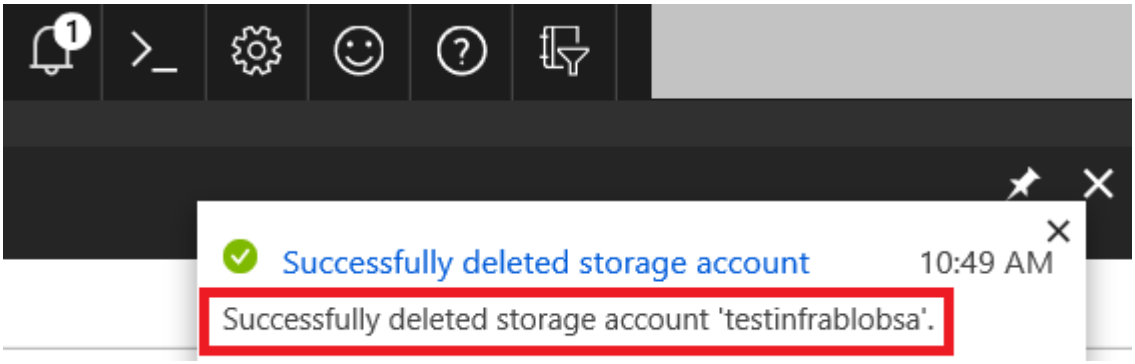
To delete a storage account, you can find the following in the **Overview** blade in the storage account:



After clicking on **Delete**, Azure will request you to type the storage account name:



Then, click on **Delete**. The operation will take a couple of seconds. A notification will launch to show that the operation was successful (as shown in the following screenshot):



Managing Azure Storage services

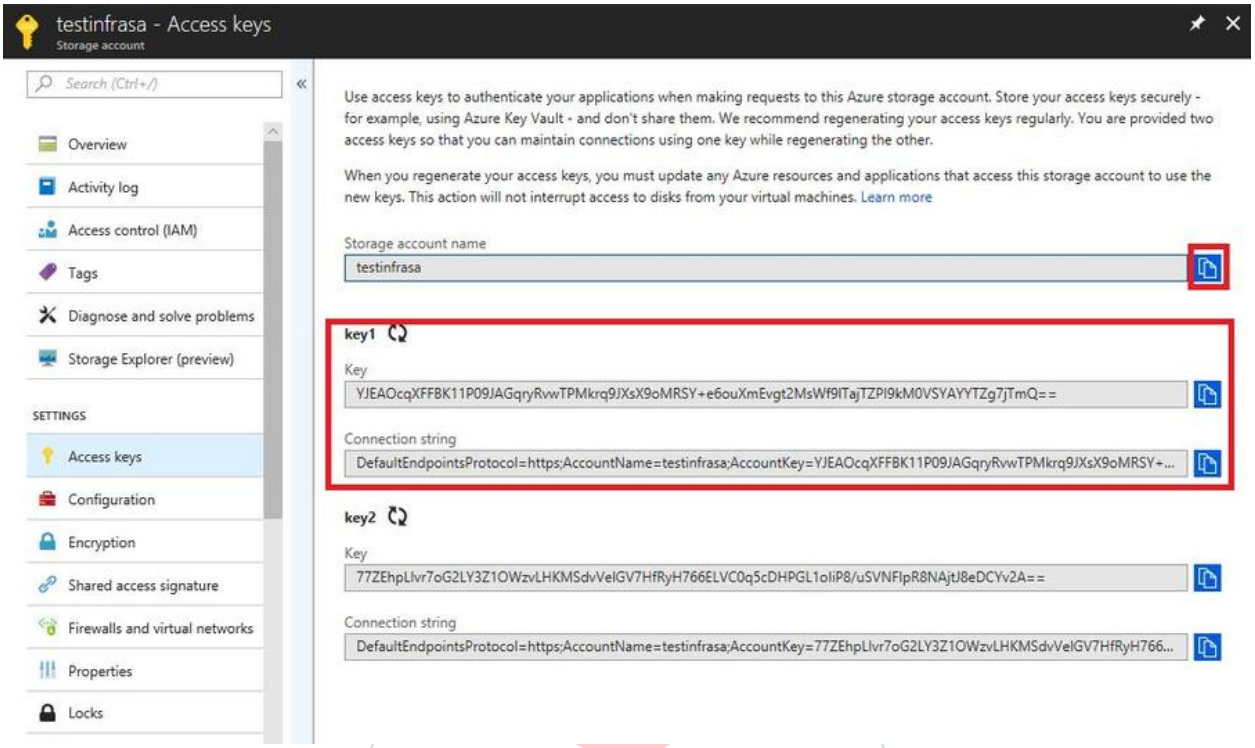
Let's see how we can manage Azure Storage services.

Managing access to storage

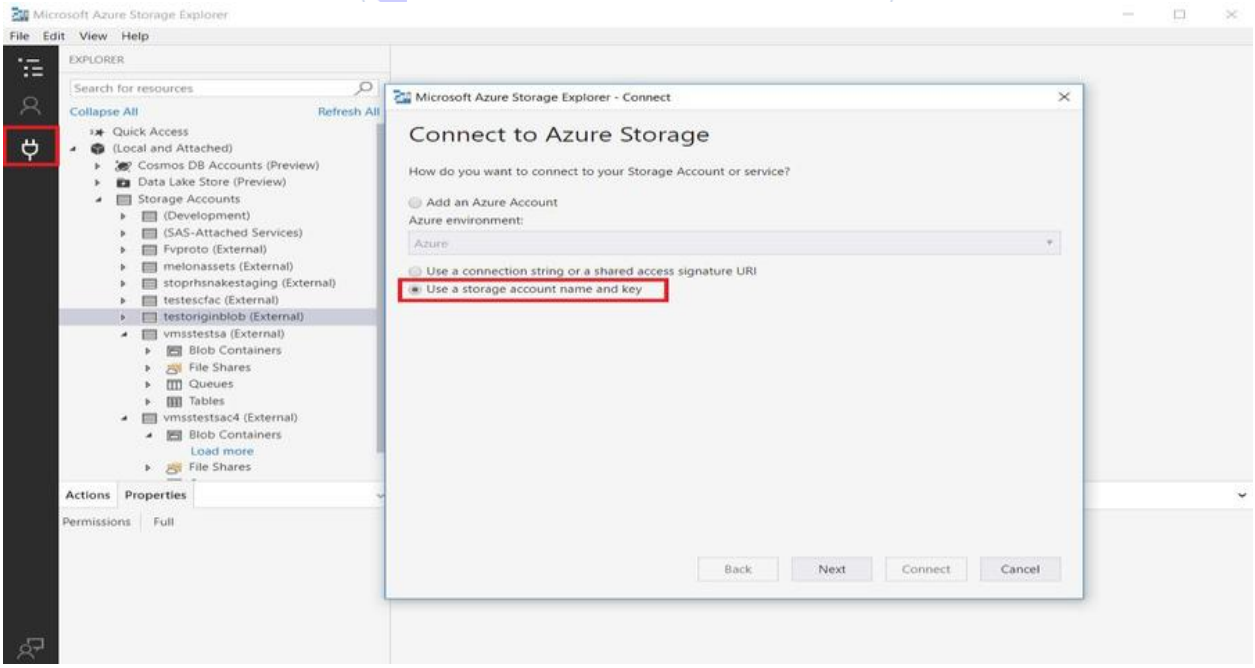
Here we'll introduce two ways to access the storage account:

Accessing storage using access keys

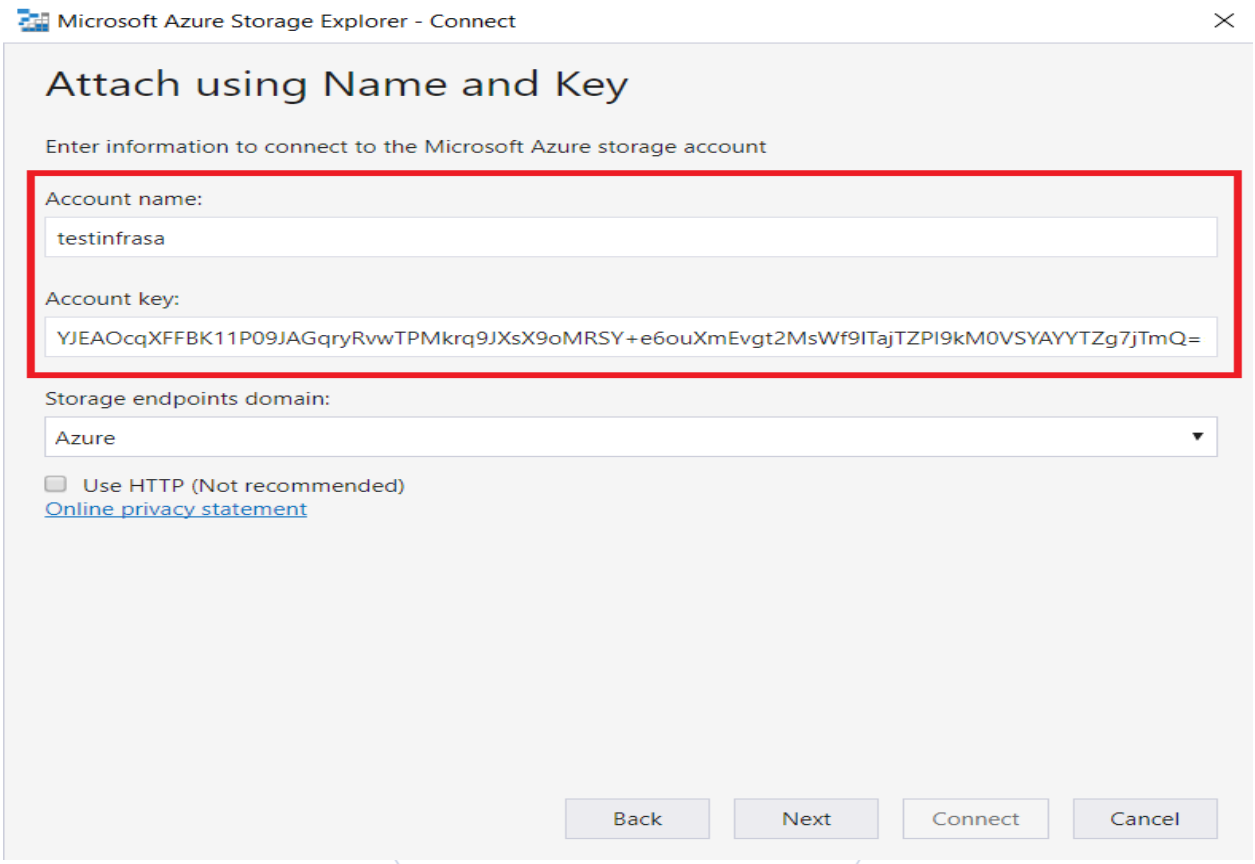
You can access your storage account using the storage account name and access keys. You can go to the **Access keys**blade of your storage account to get this information (as shown in the following screenshot):



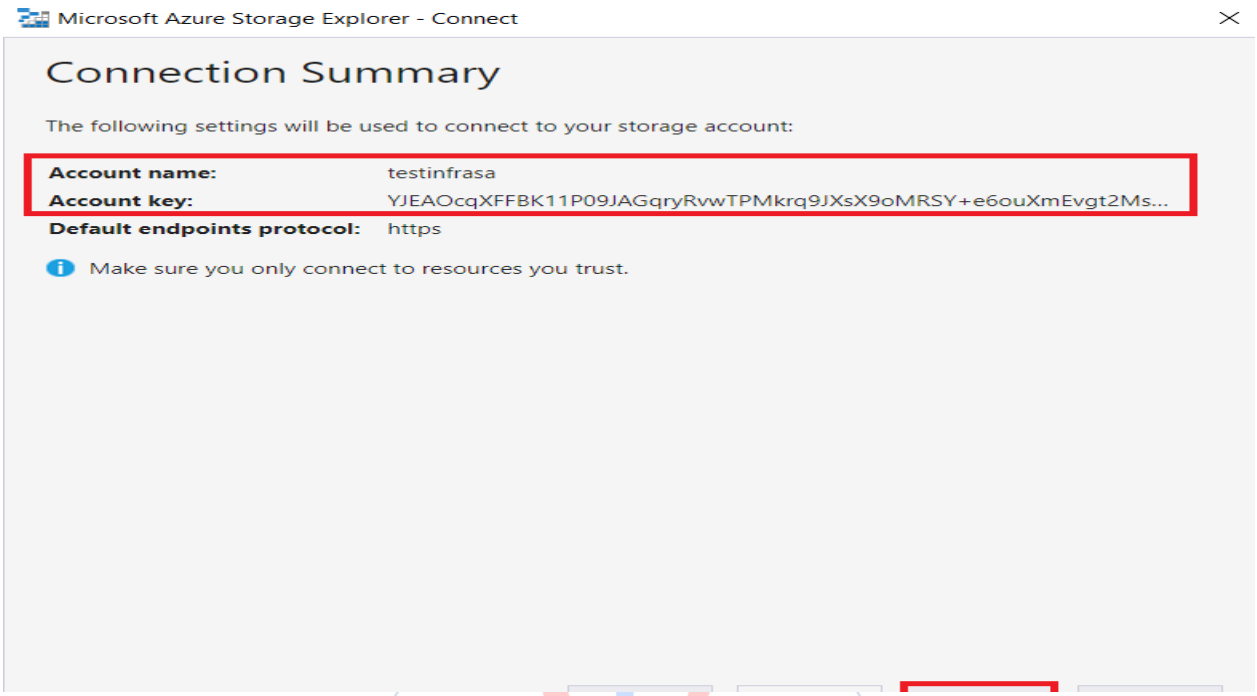
You can use this information to access Azure Storage using Storage Explorer (as shown in the following screenshot):



Select **Use a storage account name and key** and paste your storage **Account name** and **Account key** (**Key1** or **Key2** from the previous screen) as shown in the following screenshot:



Click on **Next** and you will see the following summary page:



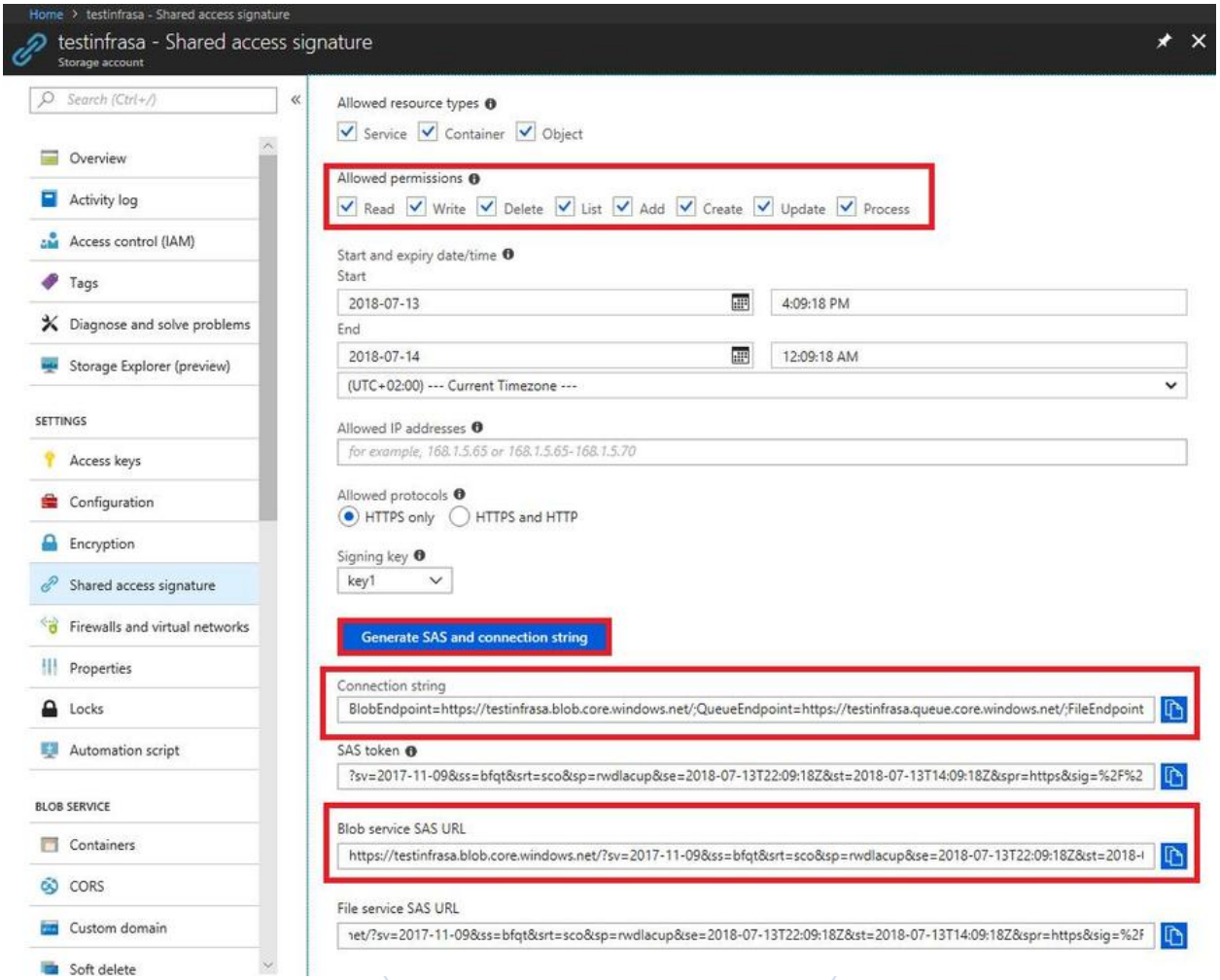
Finally, click on **Connect**. You'll be able to access your storage in Azure.

Accessing storage using shared access signatures

A **shared access signature** (**SAS**) is a URI that grants restricted access rights to Azure Storage. You can generate it at the storage account level and single-file level. As mentioned earlier, an account-level SAS can delegate access to multiple storage services, which are hosted within such as blobs, files, and queues.

Users can use a shared access signature URI to the customers who need a specified period of time to access the resources without informing the storage account name and access key.

You can go to the **Shared access signature** blade of your storage account to get this information (as shown in the following screenshot), where you can generate your SAS and configuration string:

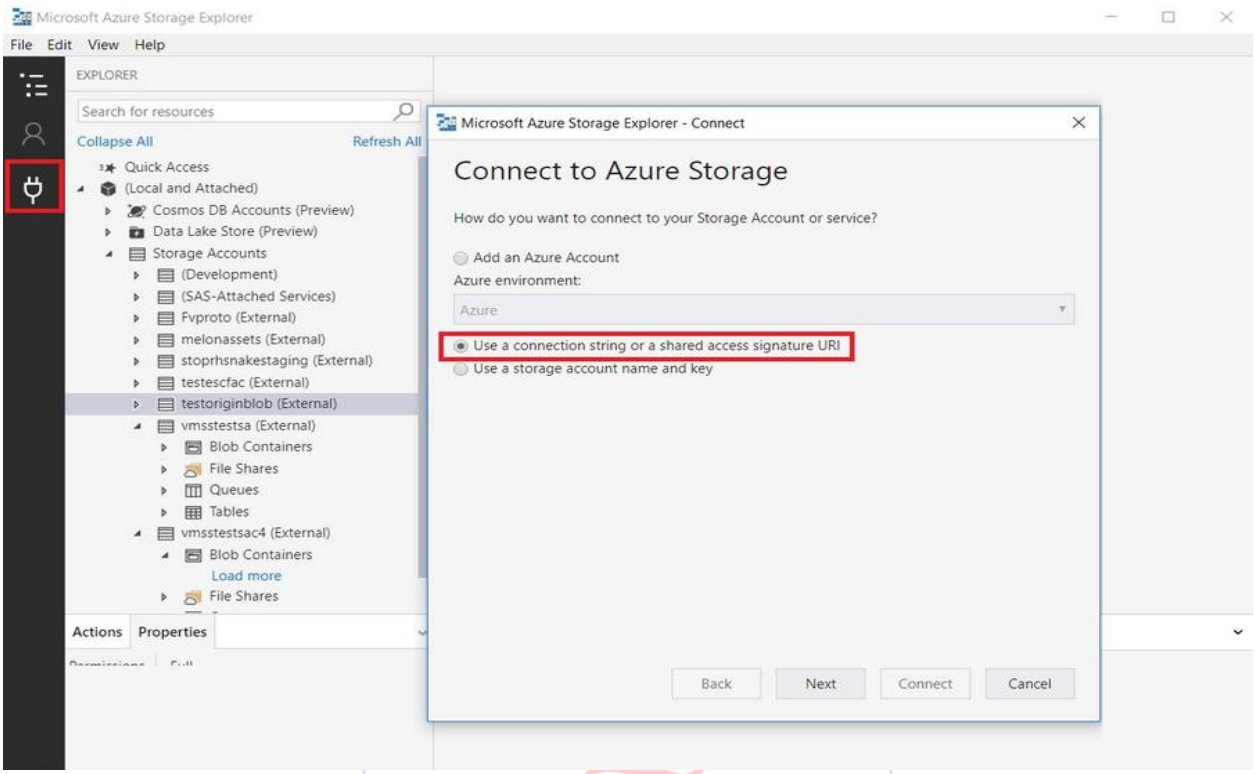


Generating Shared access signature and connecting the string via the Azure Portal

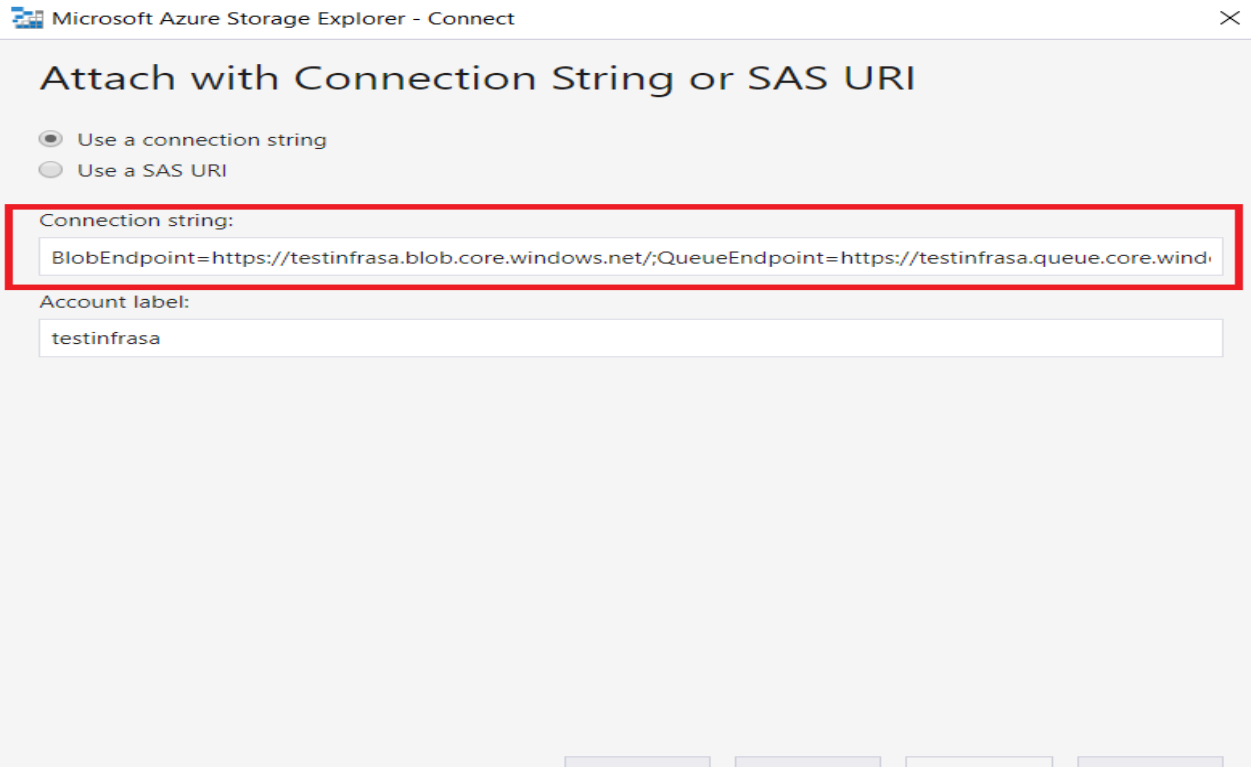
You can configure the following information according to your requirements:

- The allowed permission
- Validate period (start time and end time)
- Allowed IP addresses
- Allowed protocol

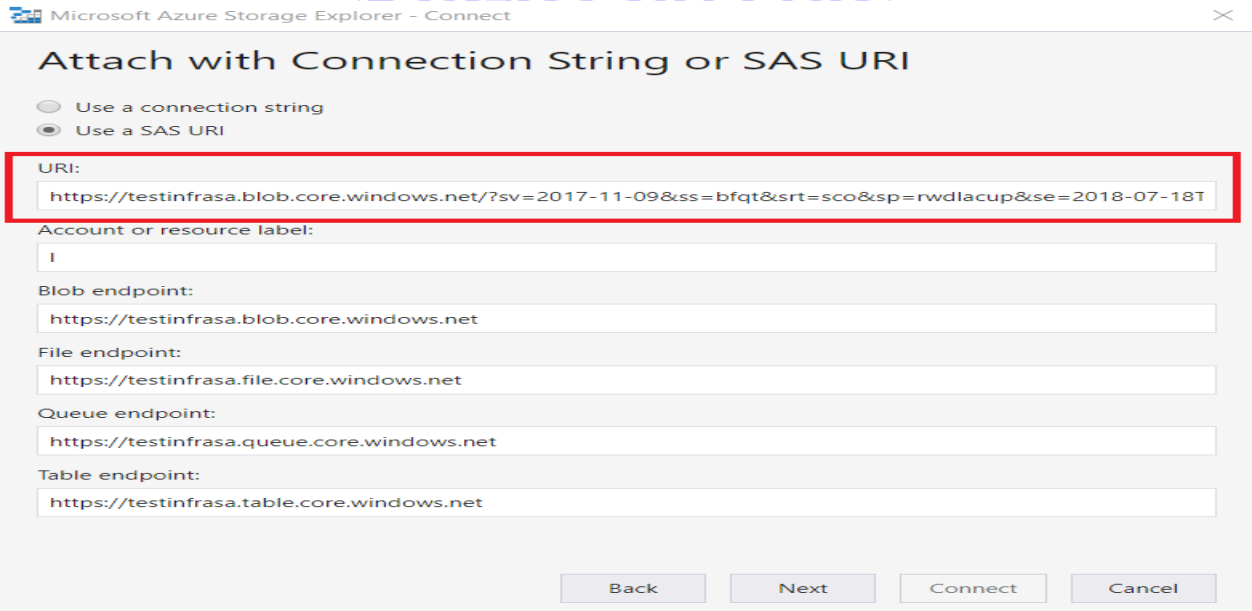
While using Storage Explorer to access your storage in Azure, you should choose use connection string or SAS URL, as shown in the following screenshot:



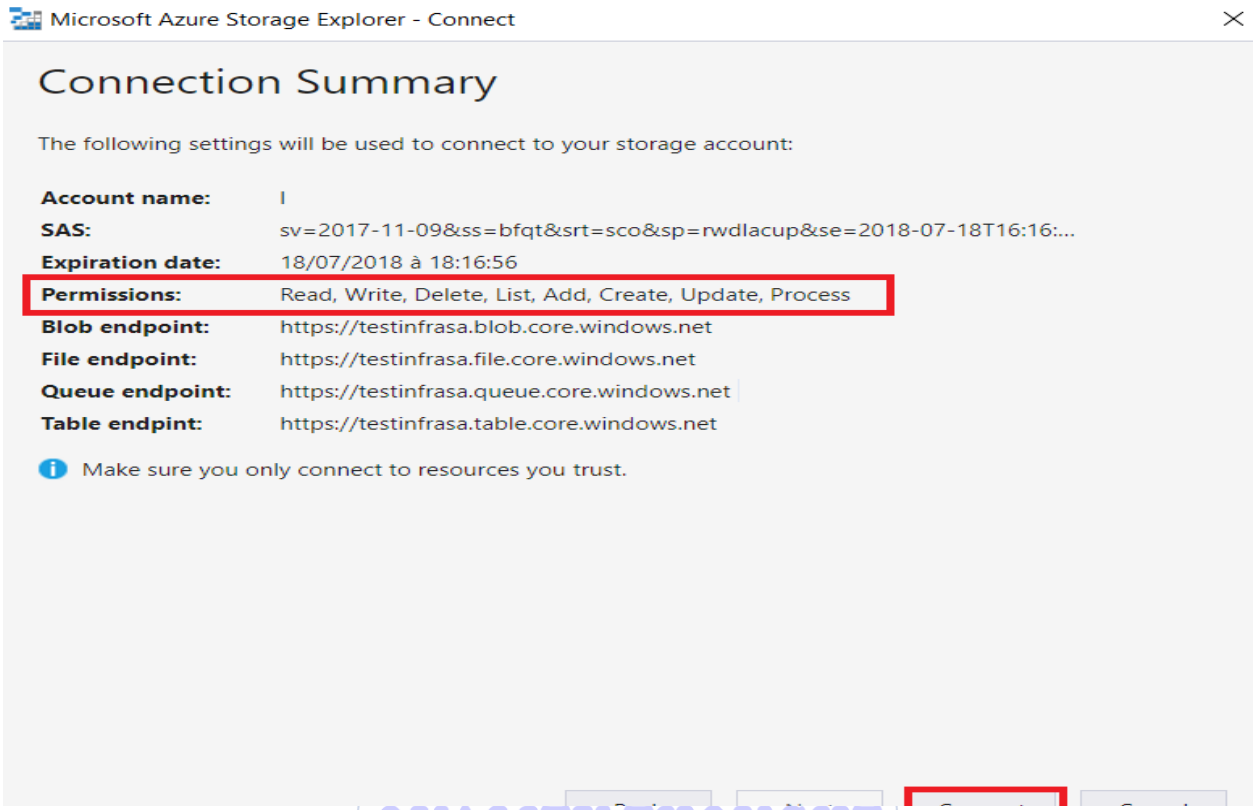
You can use connection string or SAS URL, as shown in the following screenshot:



Filling shared access signature is shown in the following screenshot:



Then click on **Next**. You'll see a summary page as shown in the following screenshot, with the exact permissions you've requested, the validation period, and so on before validating your connection information. Click on **Connect**:



Setting and retrieving properties and metadata

Objects in Azure Storage support the following two types of data (apart from the data they contain):

- **System properties**: These are defined by Microsoft and exist on each storage resource, and some of them are read-only and cannot be set
- **User-defined metadata**: This is the metadata that users can specify on a given resource in the form of a name-value pair

Users can set these values using Azure RESTful APIs. Refer to the following URL for more information: https://docs.microsoft.com/en-us/rest/api/storageservices/set-container-metadata.

The following URL shows the usage of Azure Storage Client Library for .NET: https://docs.microsoft.com/en-us/azure/storage/blobs/storage-properties-metadata.

Implementing hybrid storage solutions

Microsoft Azure offers a range of Azure Storages in hybrid scenarios such as StorSimple and Azure File Sync. Let's take a look at each of them.

Implementing Azure StorSimple

Microsoft Azure StorSimple is an integrated storage that manages storage tasks between an on-premise virtual array running in a hypervisor and cloud storage in Azure. Azure StorSimple virtual array is an excellent fit for storing infrequently accessed archival data.

To deploy the StorSimple Device Manager service for StorSimple Virtual Array, you can refer to the following URL: https://docs.microsoft.com/en-us/azure/storsimple/storsimple-virtual-array-manage-service.

Implementing Azure File Sync

Use Azure File Sync to centralize your organization's file shares in Azure Files, while keeping the flexibility, performance, and compatibility of an on-premise file server. Azure File Sync transforms the Windows Server into a quick cache of your Azure File share. You can use any protocol that's available on Windows Server to access your data locally, including SMB, NFS, and FTPS. You can have as many caches as you need across the World.

To deploy an Azure File Sync, you should have an Azure Storage account and an Azure File share that are in the same region that you want to deploy Azure File Sync in.

To make sure your region has Azure File Sync available, refer to the following URL: https://docs.microsoft.com/en-us/azure/storage/files/storage-sync-files-planning#region-availability.

Moving data to and from Azure Storage

There are a wide range of options to help users move data to and from Azure Storage.

Transferring data with AzCopy

AzCopy is a command-line utility for transferring data to and from Azure Storage. AzCopy is available on Windows and Linux. It can be used for:

- Copying data to and from Microsoft Azure Blob and File storage within a storage account
- Copying data between different storage accounts

To know more about transferring data with the AzCopy on Windows, you can refer to the following URL: https://docs.microsoft.com/en-us/azure/storage/common/storage-use-azcopy?toc=%2fazure%2fstorage%2ffiles%2ftoc.json.

To know more about transferring data with AzCopy on Linux, you can refer to the following URL: https://docs.microsoft.com/en-us/azure/storage/common/storage-use-azcopy-linux?toc=%2fazure%2fstorage%2ffiles%2ftoc.json.

Azure Storage Data Movement Library

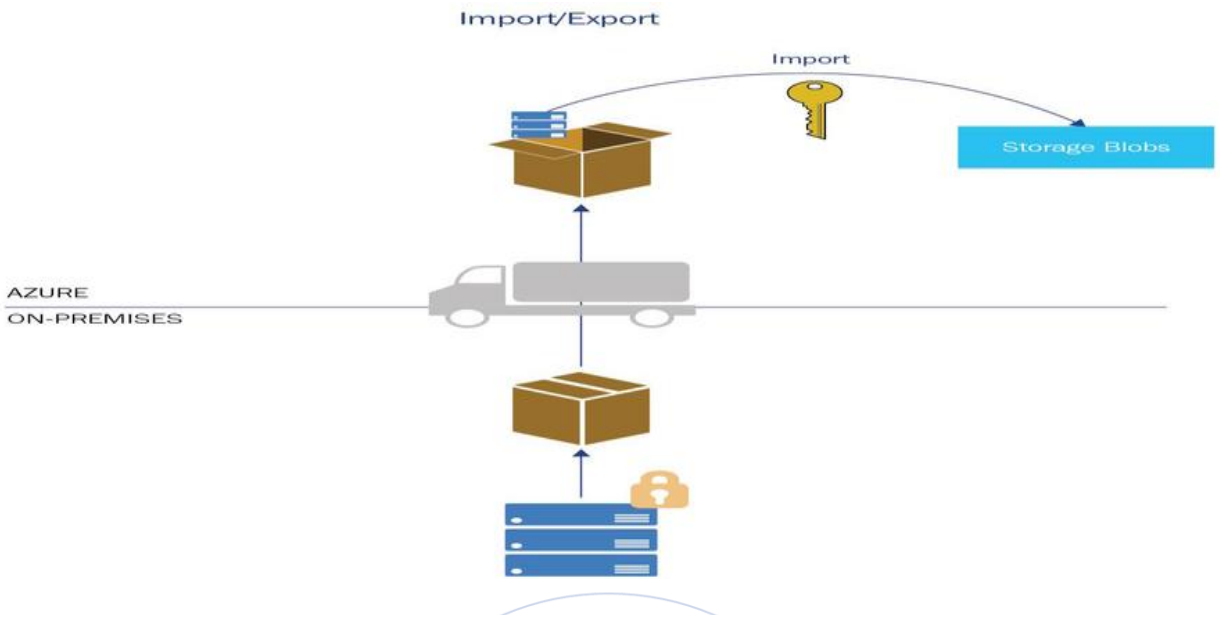
Azure Storage Data Movement Library (DML) for .NET is an open source project that exposes the core data movement framework, which powers AzCopy. It is designed for high-performance copying of data to and from Azure.

Cross-premise data transfer

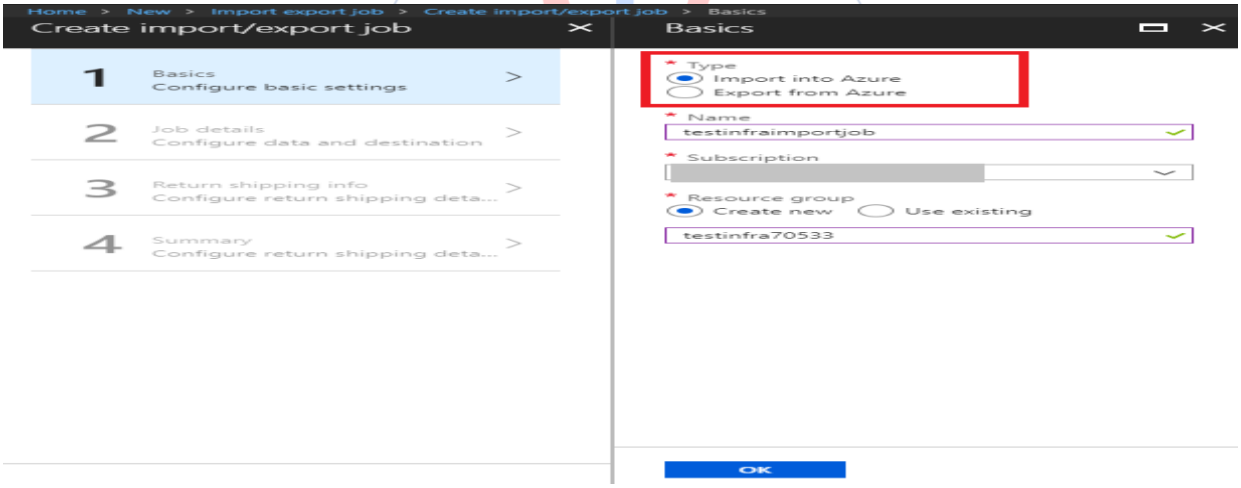
Microsoft Azure provides a couple of cross-premise data transfer options; let's take a look at each of them.

Azure Import/Export

Azure Import/Export is a service you can use to securely import large sets of data from on-premise datacenters or other cloud datacenters to Azure Blob storage (the general purpose v1 type and Azure Files type) by shipping disk drives Azure data center , or in reverse, to transfer data from Azure Blob storage to disk drives and then shipping to customers' on-premise sites. The following schema shows how it works:



You should create an import or export job via Azure Portal and fill in all the shipping information, as shown in the following screenshot:



Azure Data Box

The Azure Data Box tool help a users to transfer large amounts of data (such as terabytes of data) to Azure in a secure and quick way. Users are able to order the Data Box directly through the Azure Portal. As described by Microsoft, after filling data, users should return it to Azure Data Center so that the data can be uploaded in Azure. To see how Data box works, refer to the following screenshot:

Azure Data Box Disk

Similar to Azure Data Box, which is a portable, secure, quick, and simple way to move large datasets into Azure, Azure Data Box Disk is a lower-capacity (and easier to move) choice. After filling data, users should also return it to Azure Data Center so that the data can be uploaded in Azure. The following shows Data Box Disk and a flowchart explaining how it works:

Implementing data storage services

Azure offers a variety of **Platform as a Service** (**PaaS**) services, also known as **Database as a Service** (**DBaaS**) which, removes the need for you to manage the underlying operating system and database-server platform.

SQL Database

Azure SQL Database is a managed database service that is different from AWS RDS, which is a container service. As a PaaS offering, this frees you from performing updates and maintenance tasks, and includes built-in features that provide fault tolerance and scalability.

SQL Database offers logical servers that can contain single or multiple SQL databases. SQL Database has two different pricing models:

- vCore-based purchasing model
- DTU-based purchasing model

SQL Database also provides options such as columnstore indexes for extreme analytic analysis and reporting, and in-memory **online transaction processing** (**OLTP**) for extreme transnational processing. Microsoft manages all patching and updating work and all the underlying infrastructure.

Azure Database for MySQL

Azure Database for MySQL is a relational database service fully managed by Microsoft Azure, and is based on the MySQL Community Edition database engine. It provides users with built-in high availability and dynamic scaling as well as unparalleled security and compliance with a flexible pricing model.

To know more about creating an Azure Database for MySQL Server using Azure CLI, you can take a look at the following URL: https://docs.microsoft.com/en-us/azure/mysql/quickstart-create-mysql-server-database-using-azure-cli.

To know more about creating an Azure Database for MySQL server using the Azure portal, you can refer to the following URL: https://docs.microsoft.com/en-us/azure/mysql/quickstart-create-mysql-server-database-using-azure-portal.

Azure Database for PostgreSQL

Azure Database for PostgreSQL is a relational database service fully managed by Microsoft Azure for developers based on the community version of the open-source PostgreSQL database engine. Users get built-in high availability and capability to scale in seconds and benefit from its unparalleled security and compliance as well as a flexible pricing model.

To know more about creating an Azure Database for PostgreSQL server in the Azure portal, you can take a look at the following URL: https://docs.microsoft.com/en-us/azure/postgresql/quickstart-create-server-database-portal.

To know more about creating an Azure Database for PostgreSQL using the Azure CLI, you can refer to the following URL: https://docs.microsoft.com/en-us/azure/postgresql/quickstart-create-server-database-azure-cli.

Database-managed instances

Azure SQL Database Managed Instance is a Platform as a Service database offer, which provides nearly 100% of the SQL Server features. It enables users to migrate the on-premise SQL Server instance, which would be 100% compatible. Every instance is fully isolated from the other customer instance and assigned a private IP address.

For more information, you can refer to the followingURL: https://blogs.msdn.microsoft.com/sqlserverstorageengine/2018/03/07/what-is-azure-sql-database-managed-instance-2/.

Azure SQL Data Warehouse

Azure SQL Data Warehouse is a **massively parallel processing** (**MPP**) distributed database system that helps users focus on simply loading and querying a database without thinking about the underlying infrastructure and OS configuration.

Azure SQL Data Warehouse is actually a distributed system using nodes that work together to supply the data for any queries.

There are several differences between Azure SQL Database and Azure SQL Data Warehouse. SQL DB is for OLTP and make applications with individual updates, inserts, and deletes; SQL DW is for **online analytical processing** (**OLAP**) and is an approach to answering **multi-dimensional analytical** (**MDA**) queries swiftly in computing systems.

Cosmos DB

In July 2017, Microsoft announced Azure Cosmos DB, which is the next big leap in globally distributed, at-scale, cloud databases. Azure Cosmos DB is Microsoft's globally distributed, multi-model database. With the click of a button, Azure Cosmos DB enables you to elastically and independently scale throughput and storage across any number of Azure's geographic regions. It offers throughput, latency, availability, and consistency guarantees with comprehensive **service level agreements** (**SLAs**), something no other database service can offer.

Cosmos DB exposes multiple well-defined consistency models. It allows developers to store and query their data in its original form. It exposes graphs, documents, key-values, column-family data models, and will enable others. The multi-model and multitude of APIs to access and query data includes SQL and various popular OSS APIs. It also allows multi-API capabilities and removes friction, allowing you to build with any data model and API. Cosmos DB is all about providing intelligent choices to developers and enabling you to build planet scale apps.

Configuring Content Delivery Network

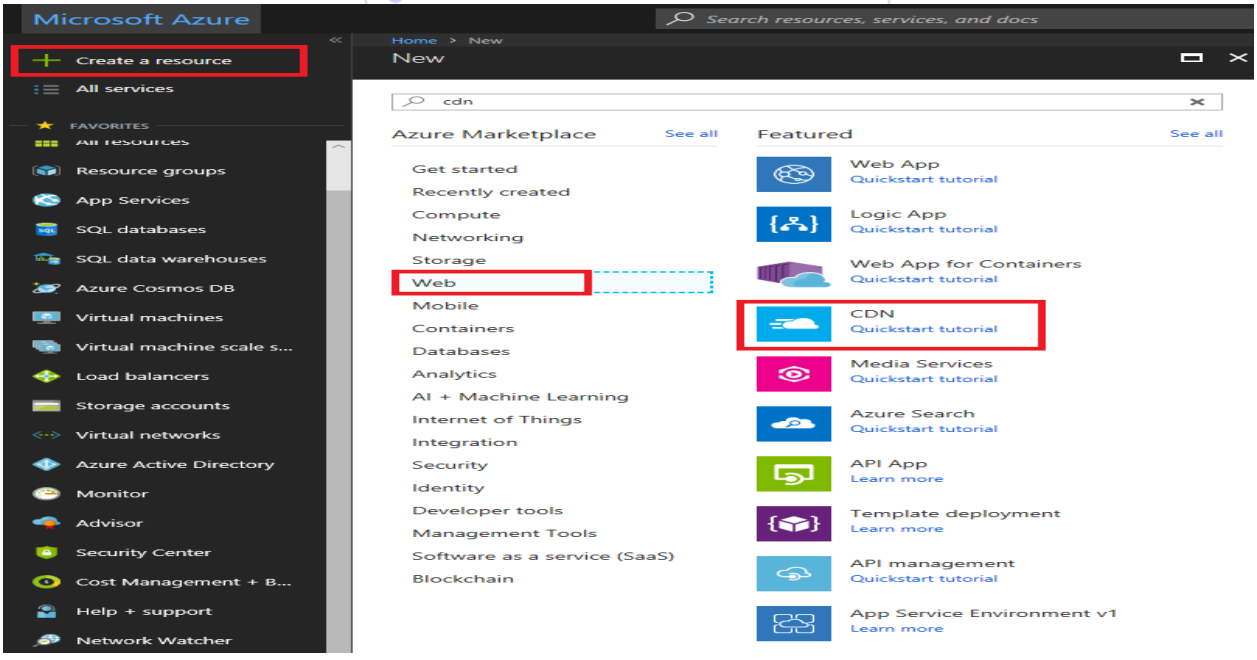
Content Delivery Network (**CDN**) is a distributed network that delivers web content to users based on their geographic location by choosing the closest edge servers in **point-of-presence** (**POP**) locations, so that the latency of distribution is maximum-ally reduced. Azure CDNs carry a significant portion of the world's internet traffic. It provides users with a global solution for delivering high-bandwidth content hosted in Azure or any other location.

CDN is used for:

- Delivering **static content** by caching files such as images, style sheets, documents, files, client-side scripts, and HTML pages
- Accelerating what to serve **dynamic content** using CDN POPs

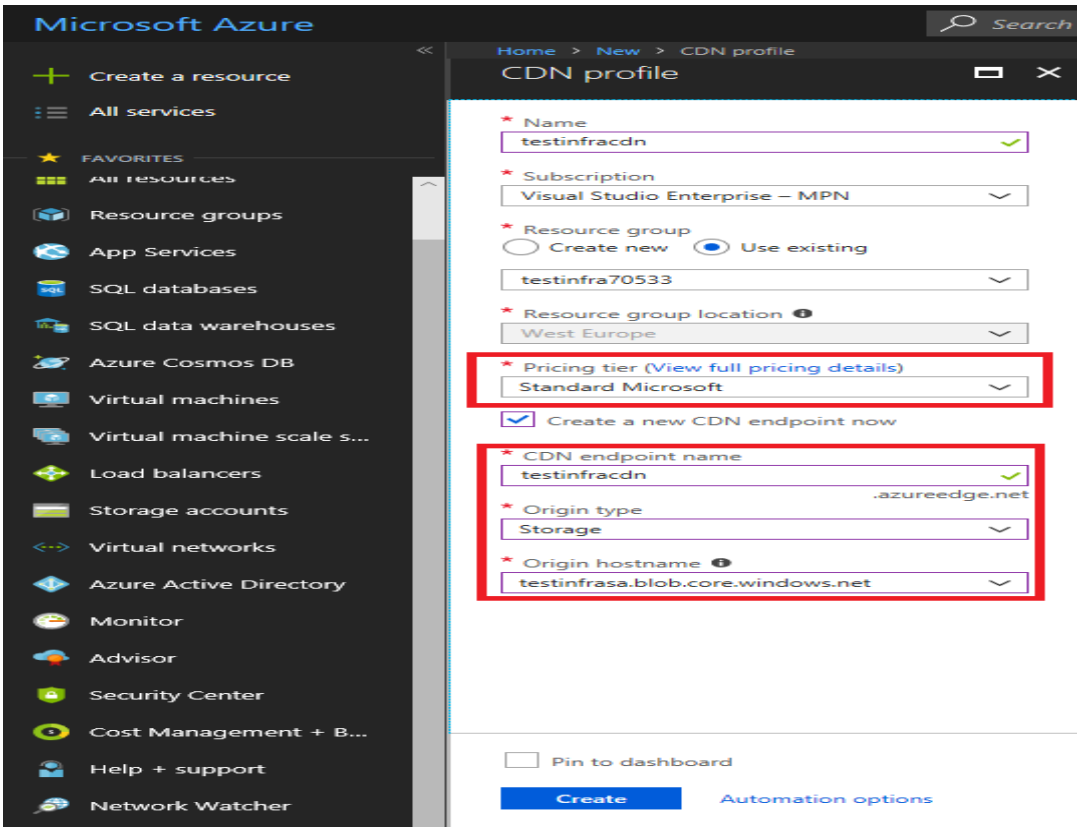
Creating a CDN profile

To implement Azure CDN, you should start by creating a CDN profile. You can go to the Azure Portal, click on **Create a resource**, then find cdn in the web category, as shown in the following screenshot:

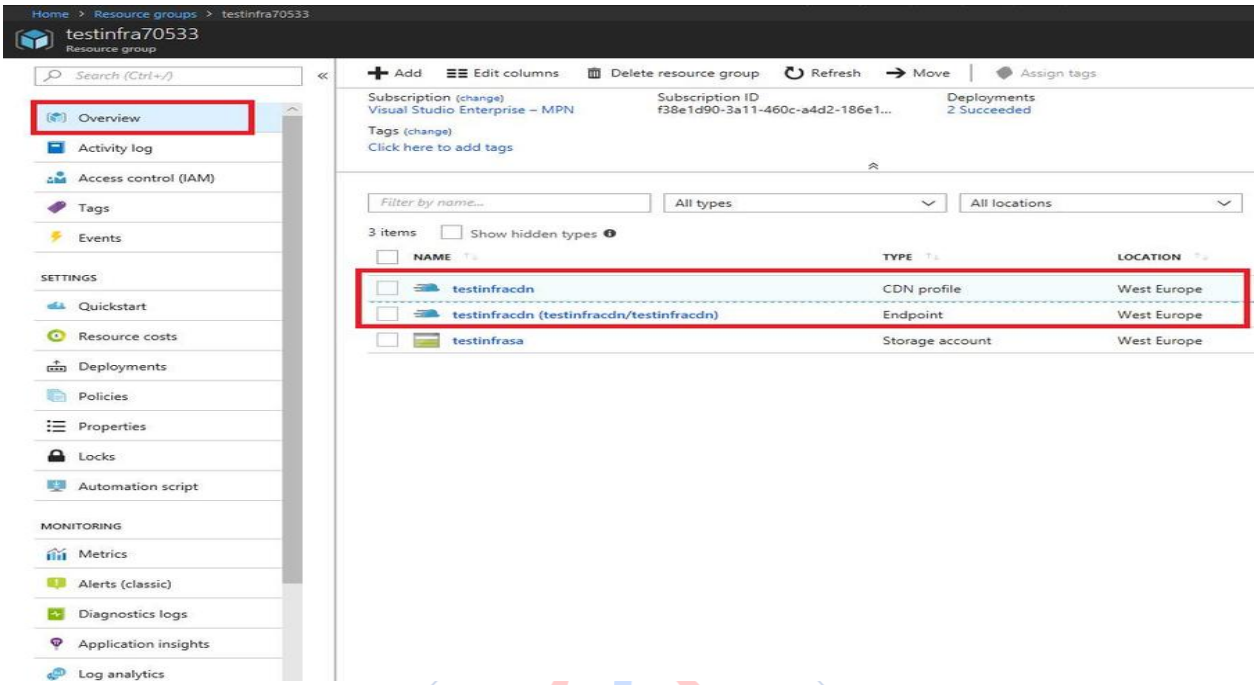


When configuring the CDN endpoint, you should choose the CDN pricing tier and the origin server. You can choose a pricing tier from Standard Verizon (S1), Standard Akamai (S2), to Standard Microsoft (S3). For more details about the pricing tier of Azure CDN, refer to the following URL: https://azure.microsoft.com/is-is/pricing/details/cdn/.

Azure supports custom CDN endpoints of any origin. You can even create an origin in your own datacenter, an origin provided by third-party cloud providers. In the following we'll point the origin server to the host of the storage account:



After clicking on **Create**, Azure will create a **CDN profile** as well as the **Endpoint** that you've created previously, as shown in the following screenshot:



Custom domains over HTTPS

We can also add custom domain mapping to your CDN endpoint and enable custom domain HTTPS. To use this feature of the Azure CDN Premium offering from Verizon, you can refer to the following URL: https://docs.microsoft.com/en-us/azure/storage/blobs/storage-https-custom-domain-cdn.

Implementing a business continuity and disaster recovery (BCDR) strategy in Azure

BCDR is a popular term nowadays. As an organization, the requirement is to adopt a business continuity and disaster recovery (BCDR) strategy that keeps the data on-premise and on the cloud so it's secure, and keeps all apps and workloads up and running even when planned and unplanned outages occur.

Planning a BCDR strategy

Every business continuity planning or disaster recovery plan potentially begins with a general business analysis, which contains some critical terms that we need to understand: These terms are listed as follows:

- **Recovery time objective** (**RTO**): This is the maximum acceptable length of time that your application can be offline
- **Recovery point objective** (**RPO**): This is the maximum acceptable length of time during which data might be lost from your application due to a major incident
- **Uptime**: This is a measure of the time a system runs over a given period of time per year
- **Downtime**: This is the period of time that a system fails to provide or perform its primary function as expected

BCDR in Azure

Cloud providers such as Microsoft Azure provide capabilities that support availability and a variety of disasterrecovery services to adapt to different scenarios by building Azure recovery services to contribute to an enterprise-level BCDR strategy. The following are two disaster recovery services:

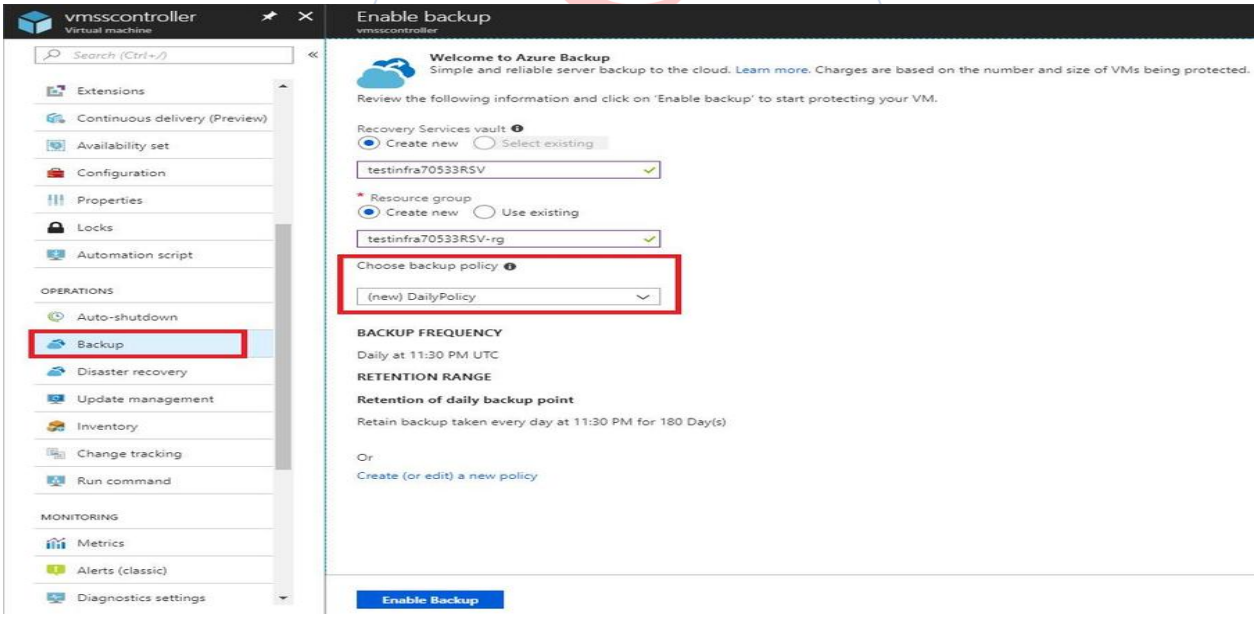
- **Azure Backup service**: This keeps your data safe and recoverable by backing it up to Azure.
- **Site Recovery service**: This is used to replicate workloads running on physical and virtual machines from a primary site to a secondary region. Users can failover to a secondary location if an outage occurs at the primary region and users can fall back to it when the primary region is back to normal.

Implementing Azure Backup

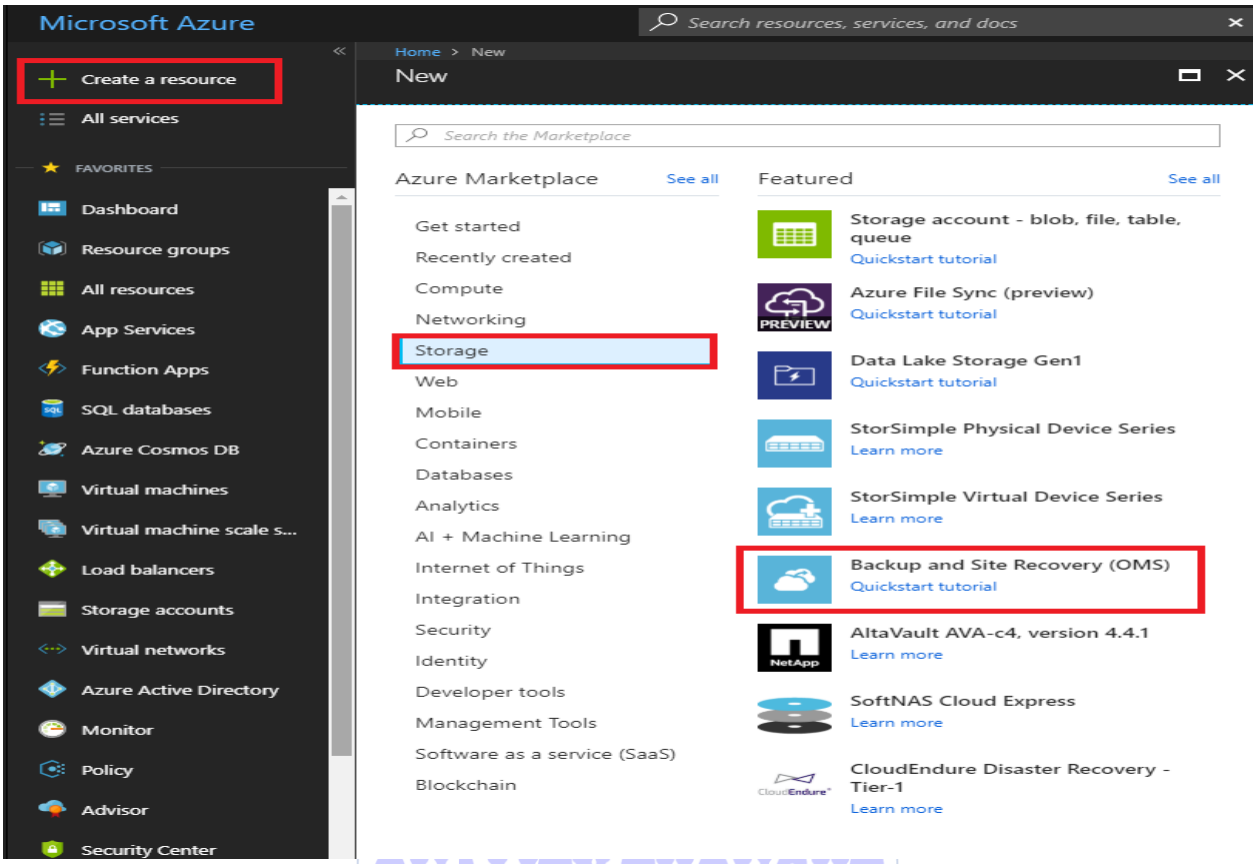
Azure Backup is a cloud service that allows users to backup and restore data in Azure. Azure Backup offers a couple of components or agents to help users deploy depending on what they want to protect. With these components or agents, Azure Backup can back up files, folders, Hyper-V or VMware virtual machines on-premise, Microsoft SQL Server, Microsoft Sharepoint, Microsoft exchange, or Azure IaaS VM.

To know more about the different components of Azure Backup, you can refer to the following URL: https://docs.microsoft.com/en-us/azure/backup/backup-introduction-to-azure-backup.

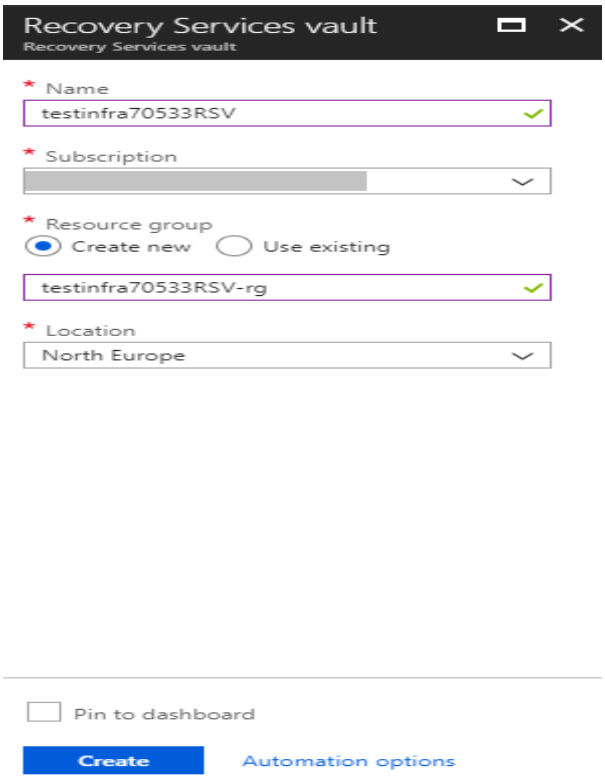
To implement **Azure Backup**, you can enable it directly in the **Backup** blade of your Azure VM instance, and create a new (or use an existing) Recovery Service vault, as shown in the following screenshot:



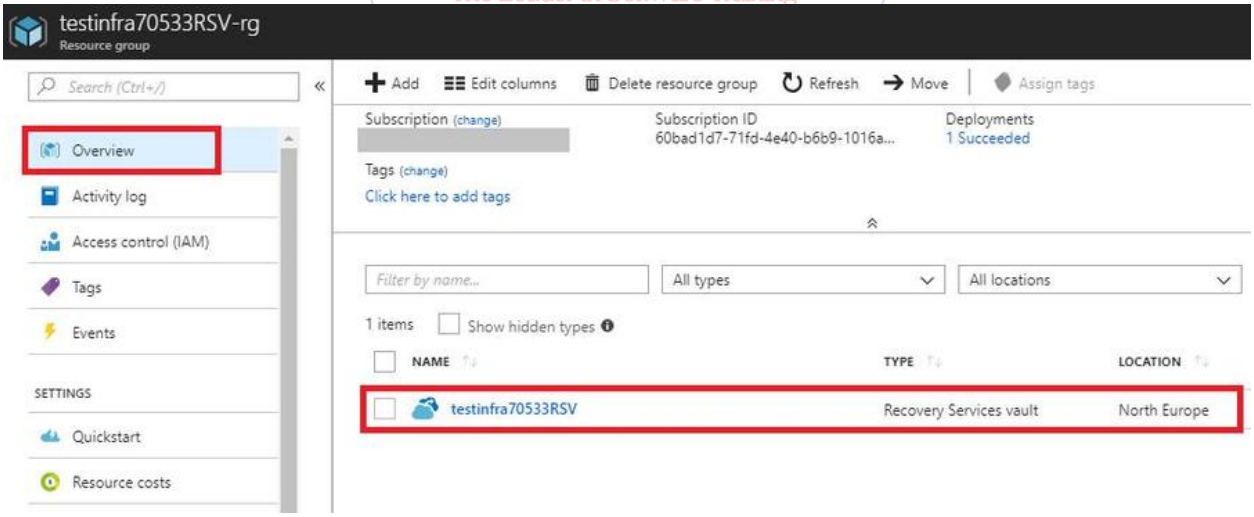
Alternatively, you can go to the Azure Portal and click on **Create a resource**. Then, go to the **Storage** category and select the **Backup and Site Recovery** (**OMS**) option:



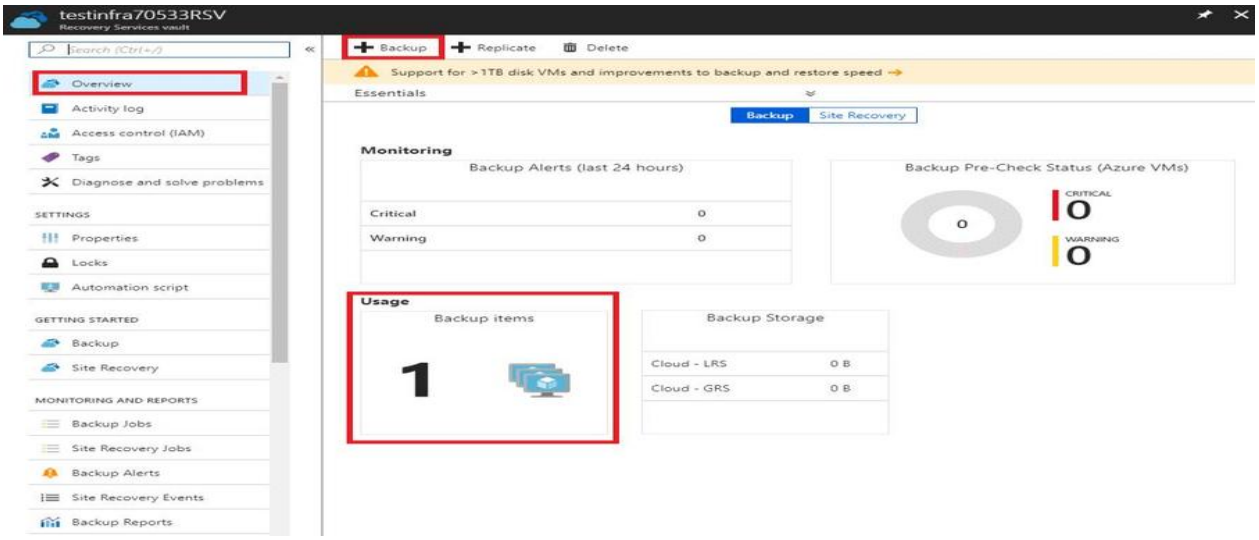
Then, you can fill complete the **Recovery Services vault** dialog to create a new instance of the RS vault:



After clicking on **Create**, you can go to the resource group where the recovery services vault has been created:



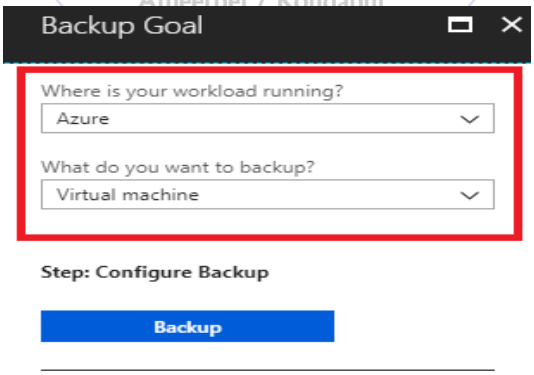
The RS Vault launches. You can see that there is a backup item that has been activated if you created your RS vault by enabling **Backup** for Azure VM, as shown in the following screenshot:



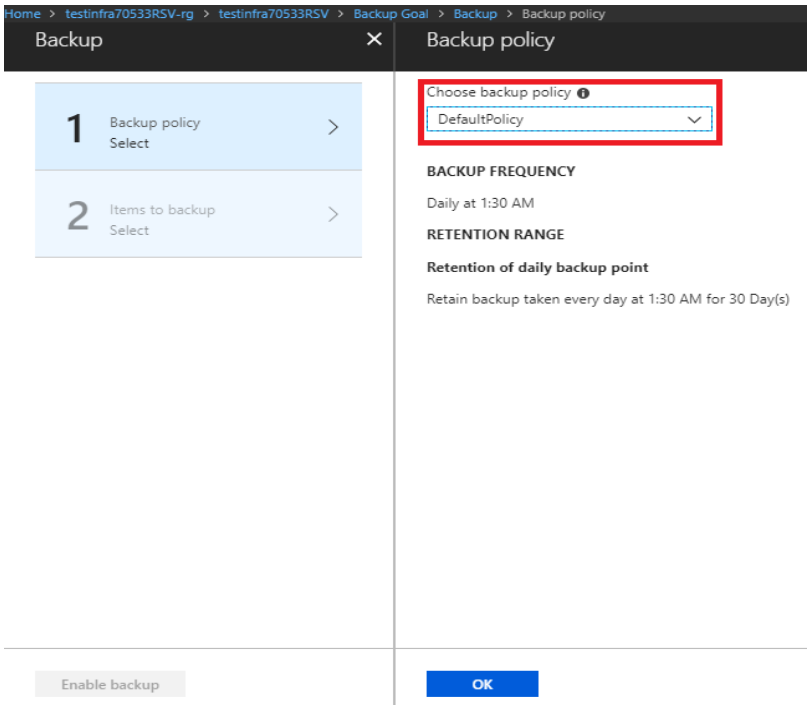
At this stage, by clicking on **Backup**, you can start to back up one or more additional Azure VMs in the cloud. As shown in the screenshot, you should firstly be clear about your objectives in creating a backup. This could be done by answering the following questions:

- Where is your workload? In the cloud or on-premise?
- What do you want to back up? Azure VMs, File share, or SQL Server in Azure VM?

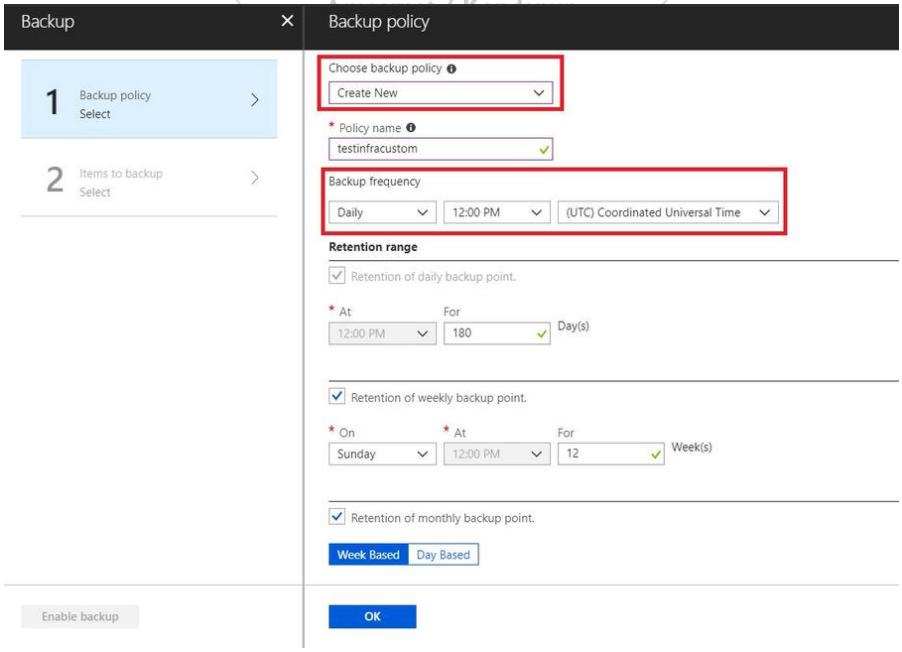
Refer to the following screenshot:



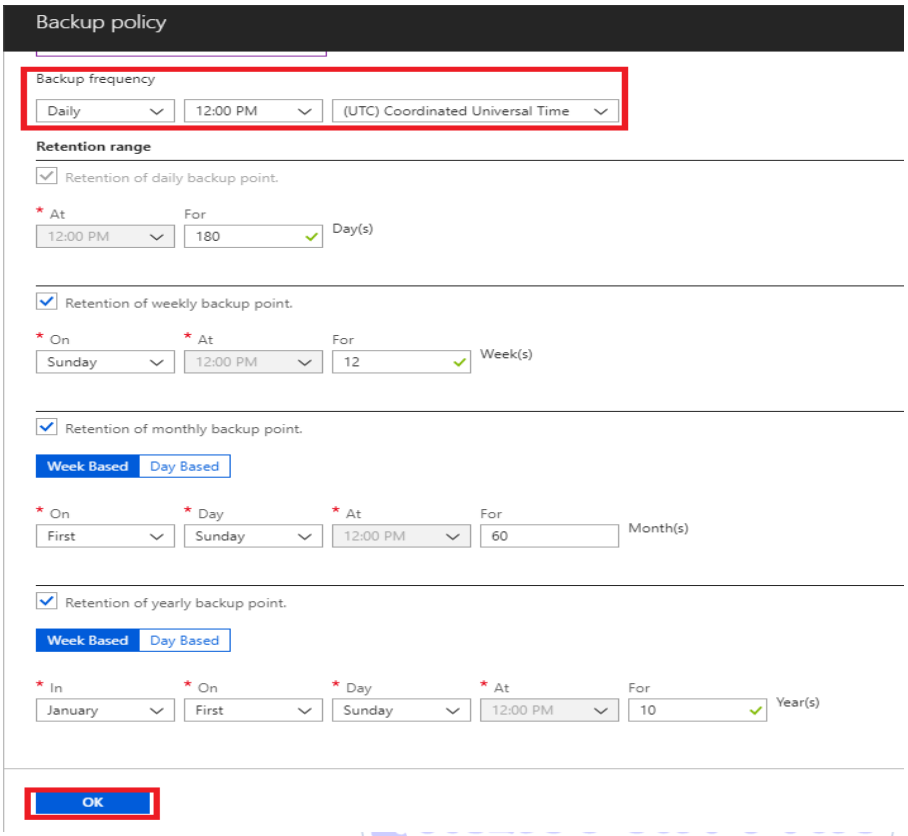
After choosing your backup goal, Azure workload, and Azure VM, for example, you can go to Azure Backup Policy to create a new policy or use an existing policy (the default or other pre-existing policy). As shown in the following screenshot, you can select the **DefaultPolicy** or create a new one:



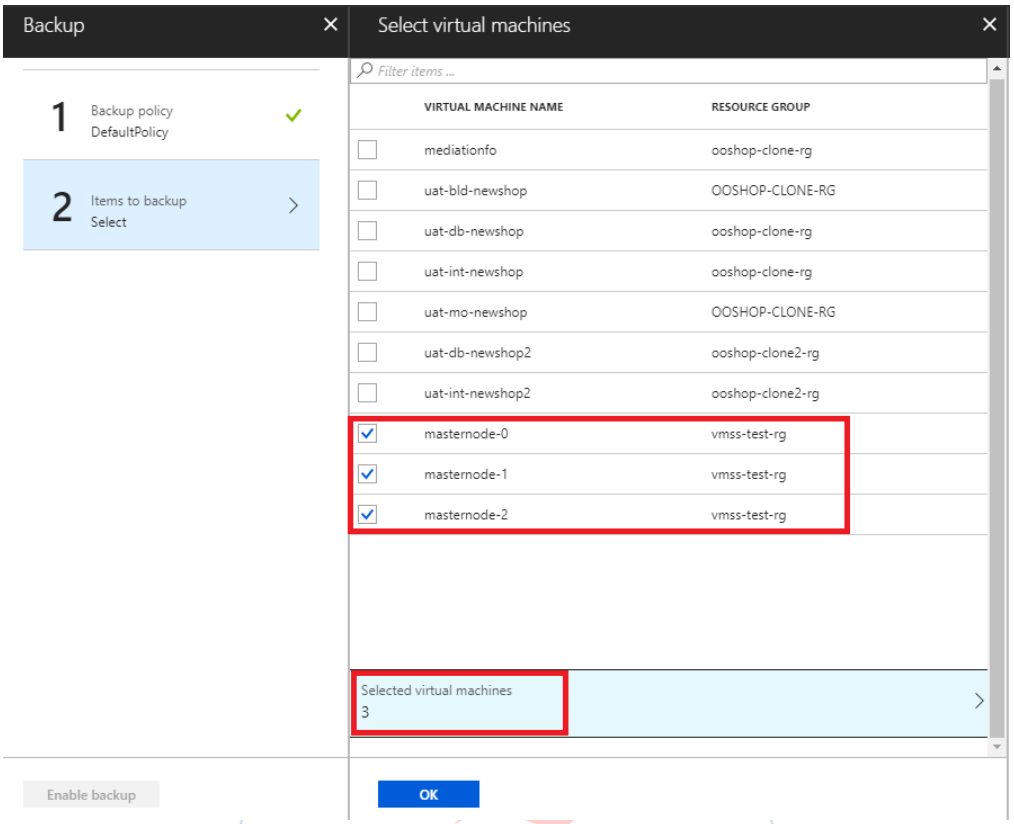
In order to create a new backup policy, you should choose a **Backup frequency** and **Retention** range, which address most your requirements, as shown in the following screenshot:



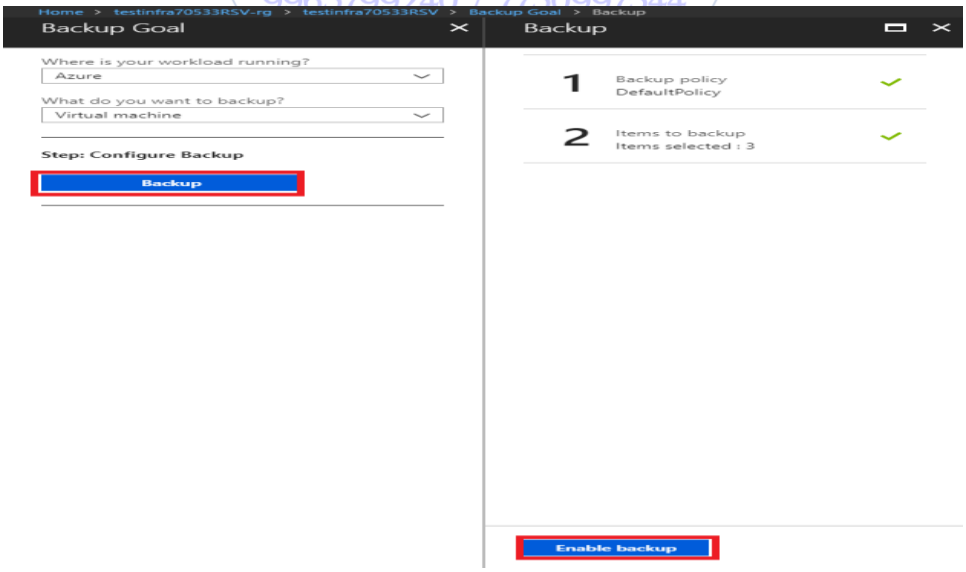
The **Retention range** is set as illustrated in the following screenshot:



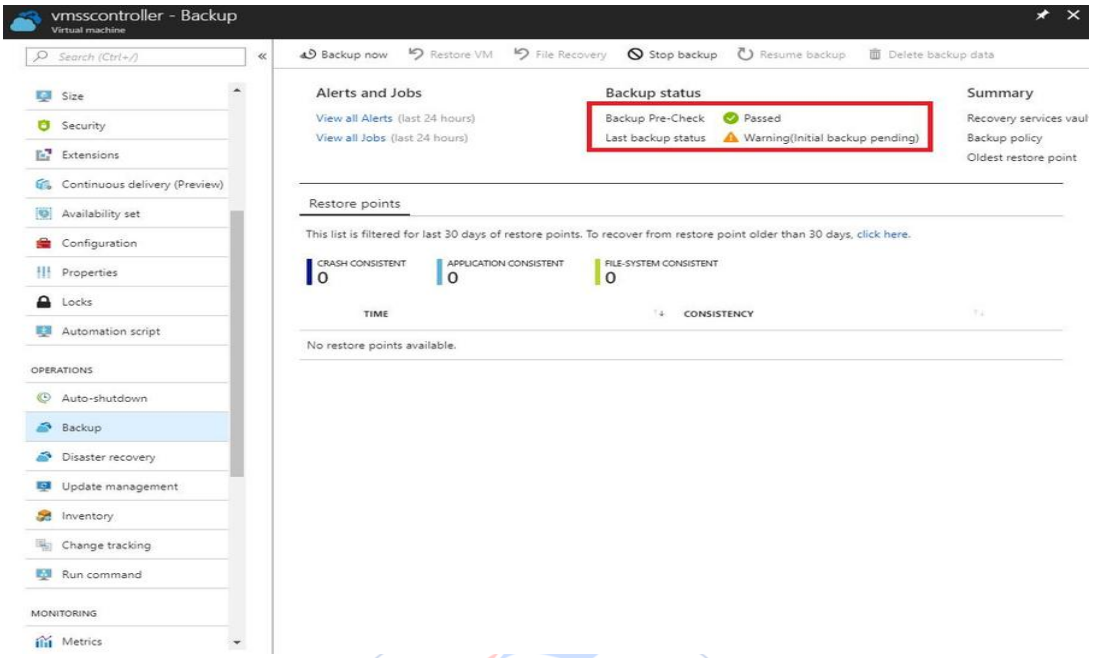
Now click on **OK**. You can go to the next step to choose items to back up. You can choose one or multiple items, here we choose 3 items; from the same resource group, **Selected virtual machines** displayed as 3 as shown in the following screenshot:



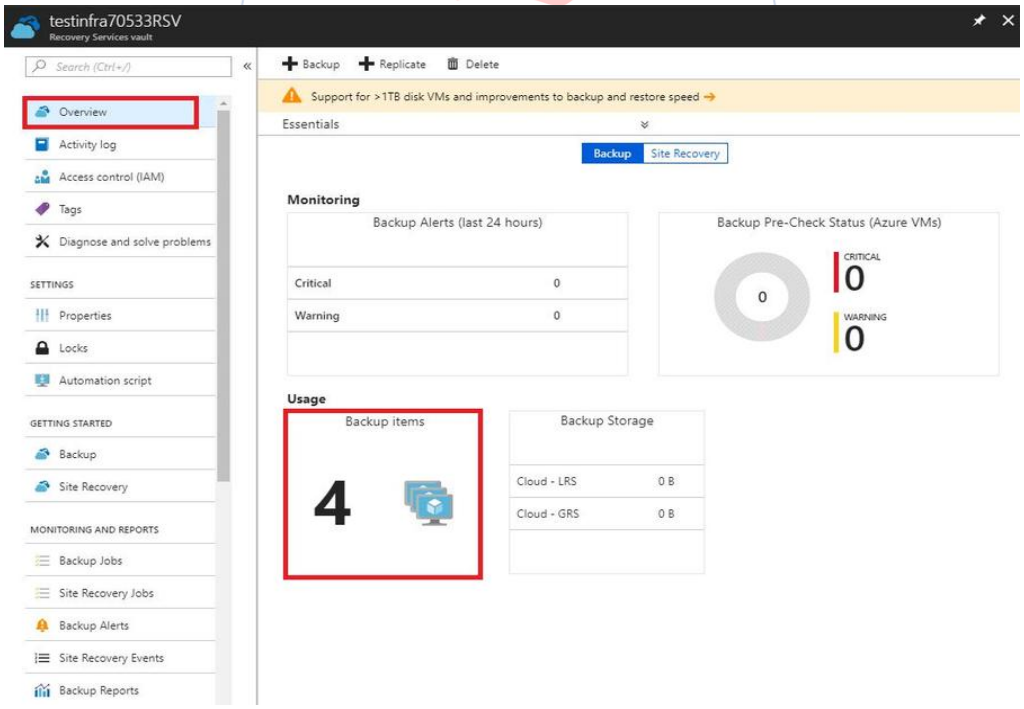
When you click on **OK** and then on **Enable backup**, the backup process will start, as shown in the following screenshot:



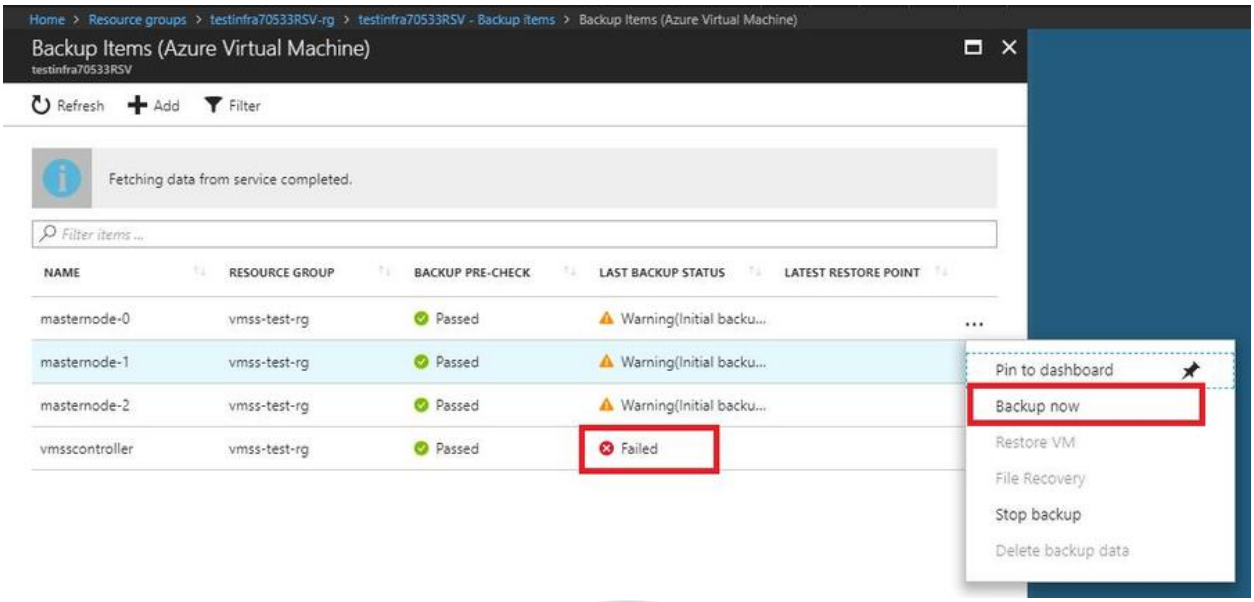
It will take a few minutes to complete a backup of the VM. But you can check the backup status of the VM by going to its **Backup** blade, as shown in the following screenshot:



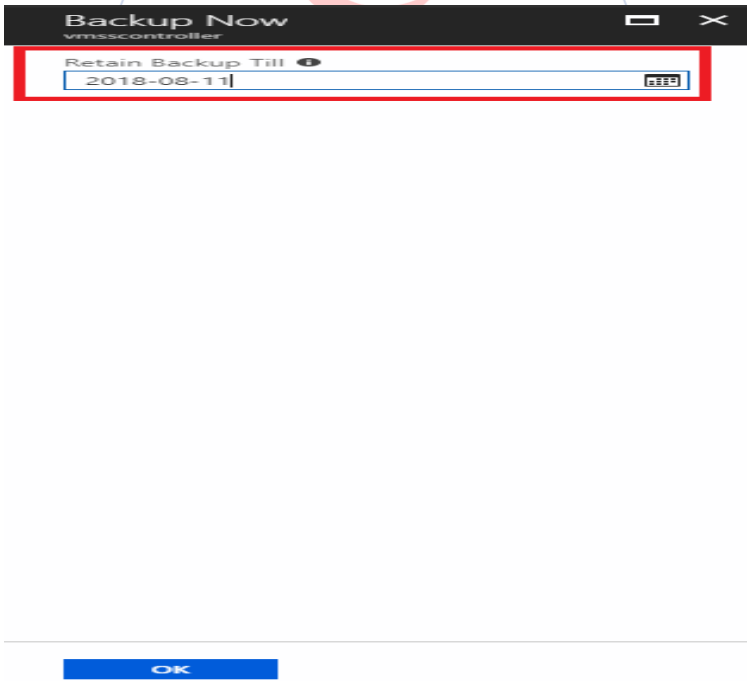
Alternatively, you can go to the **Overview** blade in the RS vault, as shown in the following screenshot, and click on **Backup items** to check the status of all the items:



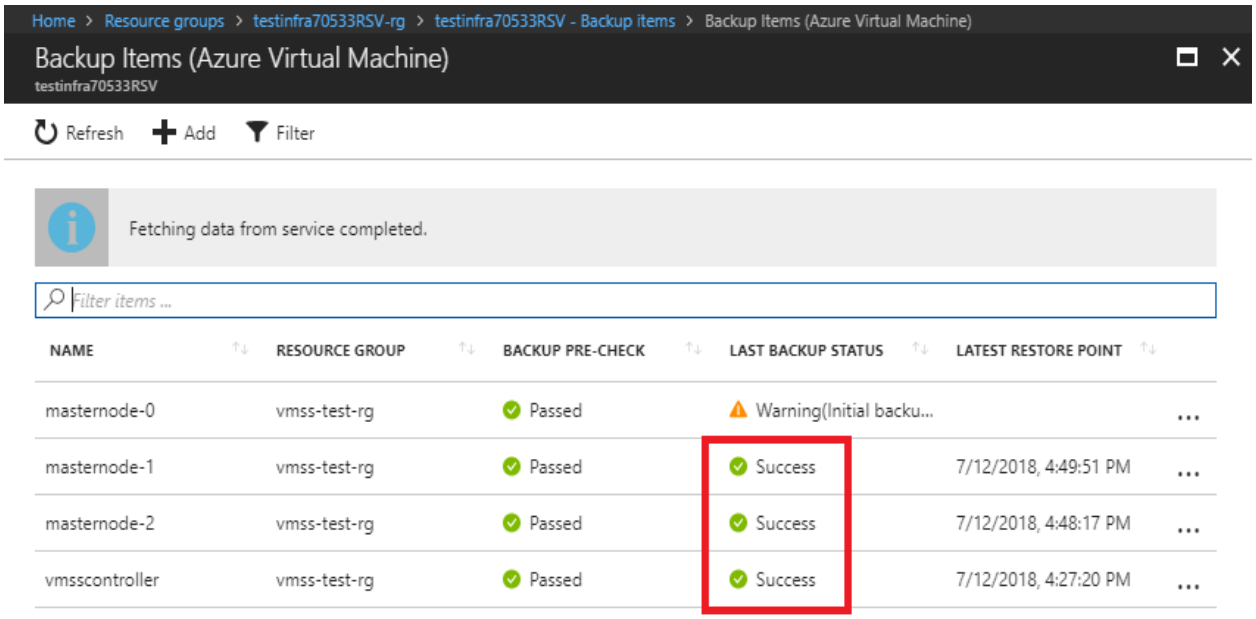
As you can see in the following screenshot, a Failed message will be displayed if the backup fails. You can right-click the item and select **Backup now** to launch a backup progress:



If you want to accept the backup retention policy of 30 days, just use the default **Retain Backup Till** date, as shown in the following screenshot:



After a couple of minutes, you can go to your RS vault to recheck the backup status of your VMs as shown in the following screenshot, which means the last backup (for three VMs) was successful:



You can also achieve the same result using Azure PowerShell as shown in the following URL: https://docs.microsoft.com/en-us/azure/backup/quick-backup-vm-powershell.

You can also use the Azure CLI by referring to the documentation at the following URL: https://docs.microsoft.com/en-us/azure/backup/quick-backup-vm-cli.

To create a backup vault via the ARM template, refer to the following code:

```
{
    "type": "Microsoft.RecoveryServices/vaults",
    "apiVersion": "2018-01-10",
    "name": "[parameters('vaultName')]",
    "location": "[parameters('location')]",
    "sku": {
      "name": "RS0",
      "tier": "[parameters('skuTier')]"
    },
    "properties": {}
  }
```

There are also a couple of useful Azure Resource Manager templates for Azure Backup, which you can find at the following URL: https://docs.microsoft.com/en-us/azure/backup/backup-rm-template-samples.

Planning and implementing Azure Site Recovery

Azure Site Recovery (**ASR**) can manage replication for the following points:

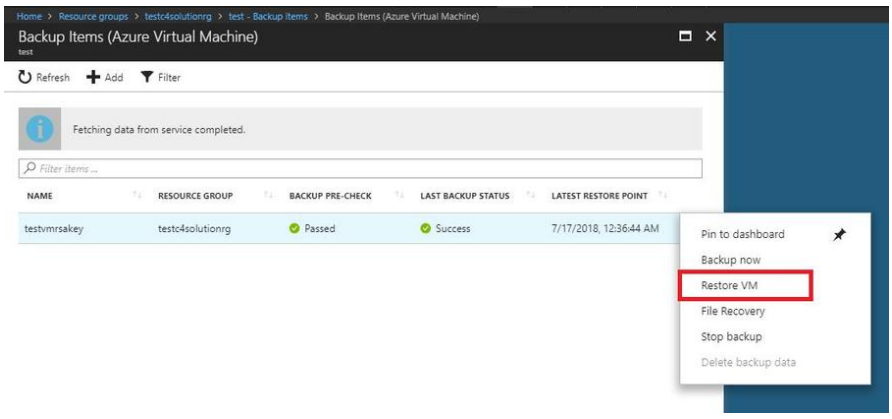
- Migrating on-premise Hyper-V VMs, VMware VMs, and physical servers to Azure
- Migrating Azure VMs between Azure regions
- Migrating AWS Windows-based Instances to Azure IaaS VMs

ASR is a powerful Azure service, but is simple to use. It provides an on-click restore facility and can replicate any workload running on a machine that's supported for replication. Refer to the following link for more information: https://docs.microsoft.com/en-us/azure/site-recovery/site-recovery-overview.

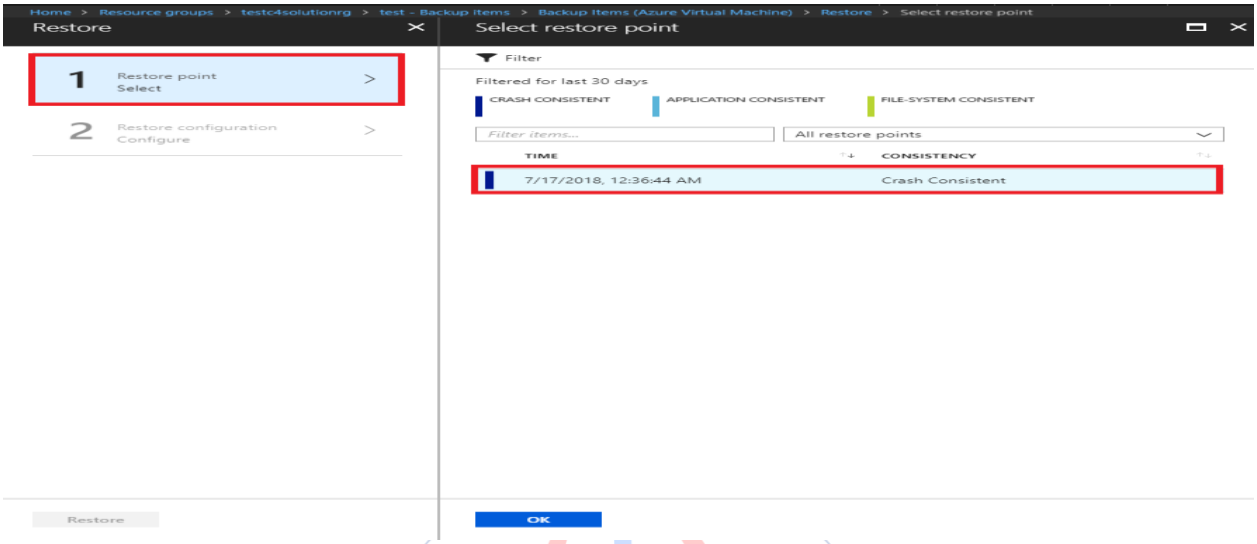
ASR can be used in simple-to-very complex use cases. Now let's take a look at two general use scenarios in the upcoming subsections.

Restoring VMs with ASR

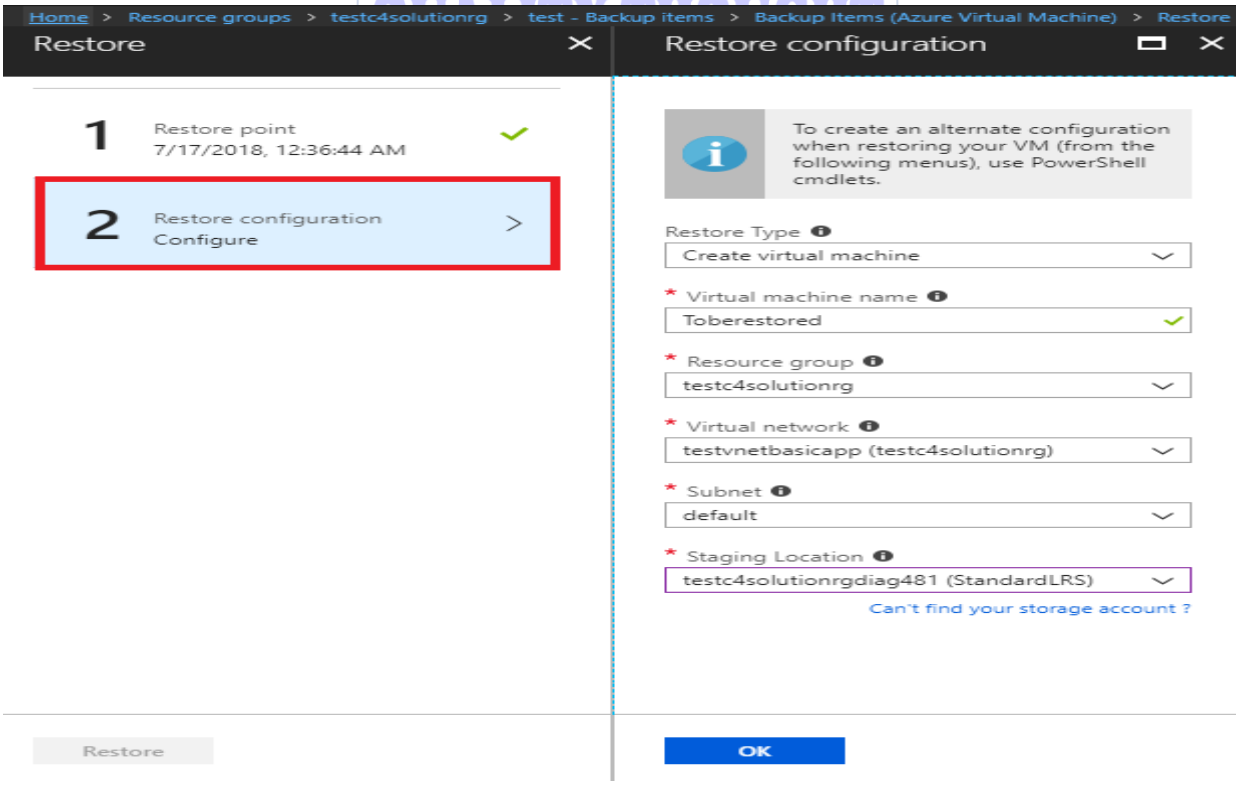
When you go back to check backed-up VMs, you can restore them in just one click (as shown in the following screenshot):



What you need to do is select which **Restore point** you should recover, as shown in the following screenshot:

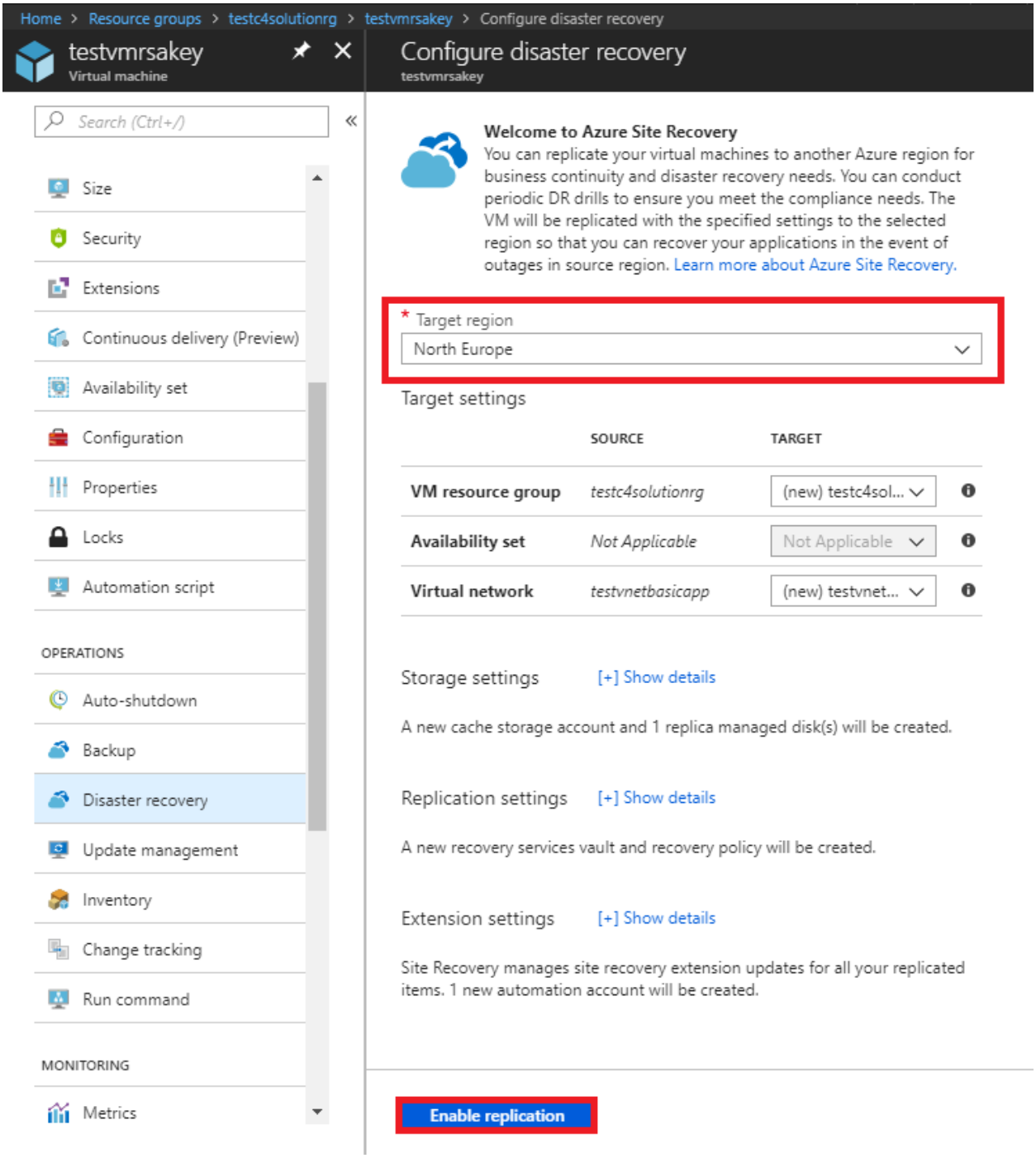


Simply configure where you'll restore your backup. The operation will take a couple of minutes, but it is simple to use since all the manipulations can be done via the Azure portal:

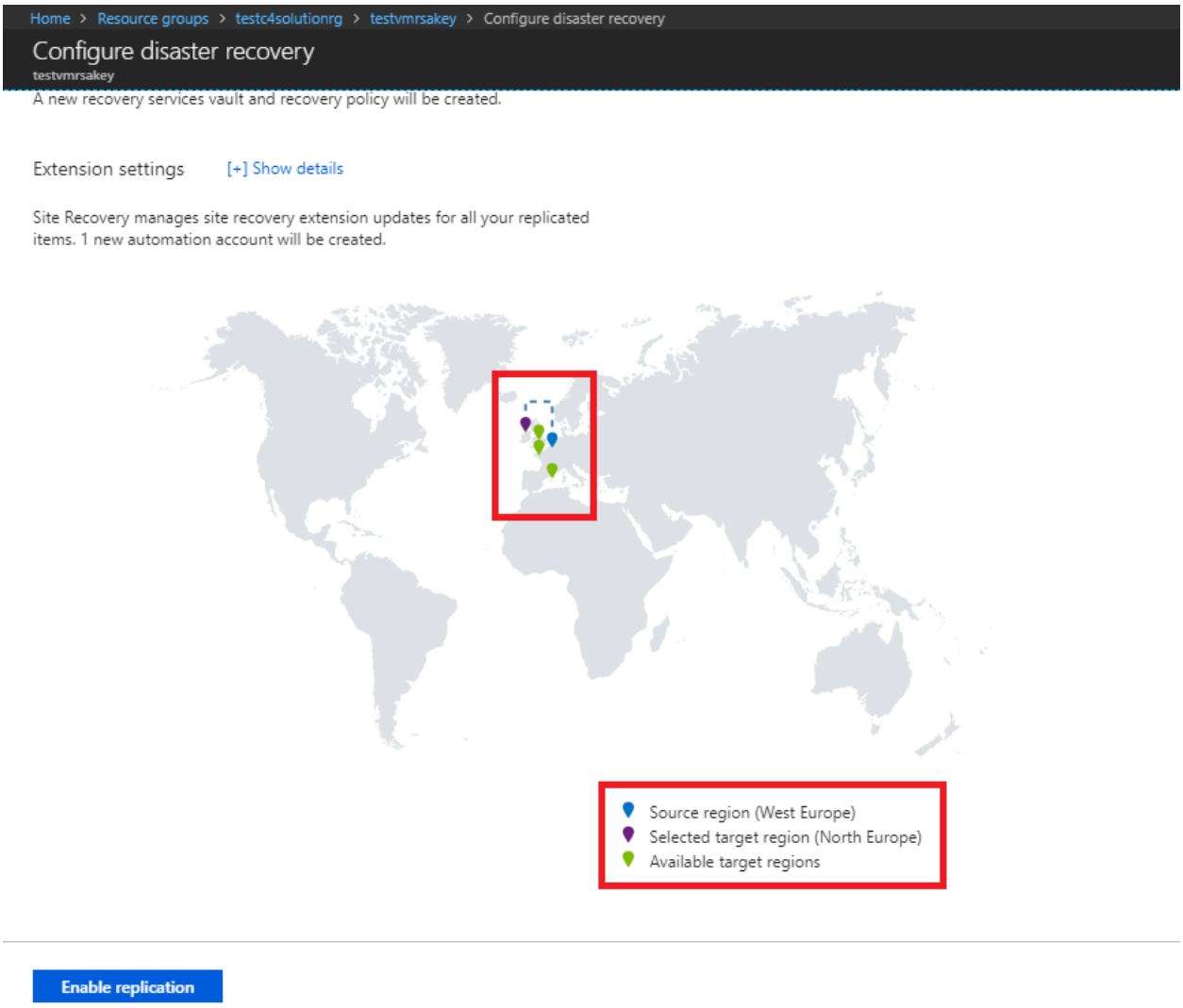


Replicating workloads to another region

Go to the **Disaster recovery** blade of Azure VM and choose a **Target region** to replicate. By default, it is in the paired region of the region where the resource you choose was deployed:



You can create a new recovery service vault and configure a recovery policy. It will display the Azure regions in which you can replicate your workloads, as shown in the following screenshot:



Managing Azure Operations and Automation

Azure Automation manages the life cycle of the infrastructure and applications in Azure by using runbooks, it also allows the use of **Desired State Configurations** (**DSC**) to configure Windows-based and Linux distribution-based machines at the infrastructure and application level across hybrid environment. It can also work with CI/CD tools, such as Jenkins and **Visual Studio Team Services** (**VSTS**).

In this chapter, you will learn the following topics:

- Automation services and their main characteristics
- An overview of monitoring solutions in Azure
- Implementing and configuring Log Analytics (OMS)
- The different types of runbooks that Azure Automation supports
- Publishing and executing runbooks within Azure Automation

Implementing Azure Automation

Azure Automation is an automation and configuration service in Azure. It provides a wide range of capabilities during deployment, operations, and decommissioning of workloads and resources:

- **Process automation** makes it possible to automate frequent, time-consuming cloud management tasks along with reducing errors and lowering operational costs.
- **Configuration management** allows one to manage DSC resources in Azure Automation and apply configurations to virtual or physical machines across hybrid environments.
- **Update management** allows one to update different operating systems such as Windows and Linux across hybrid environments and provides the visibility of update compliance across Azure, on-premises, and other clouds.
- **Shared capabilities** are a set of shared resources that make it easier to automate and configure environments at scale, such as role-based access control, variables, and credentials.

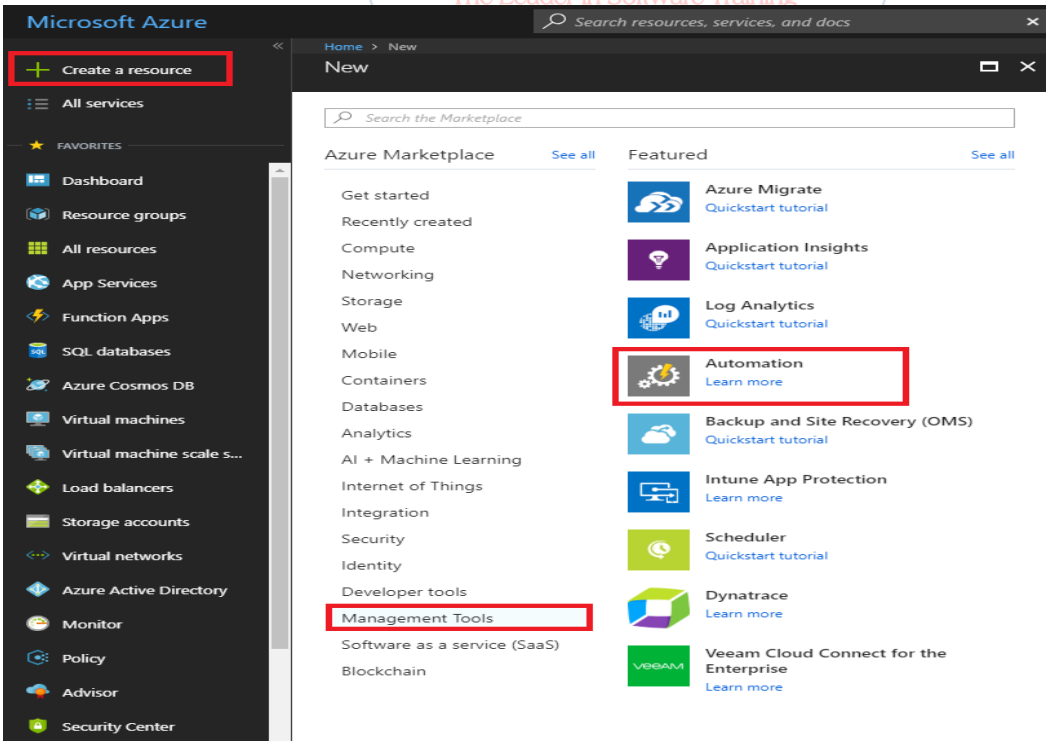
An overview of runbooks

Azure Automation supports a couple of types as described in the following:

- **Graphical runbook** can be created and edited completely in a graphical editor via Azure Portal
- **Graphical PowerShell Workflow runbook** is based on Windows PowerShell Workflow and is created and edited completely in the graphical editor via Azure Portal
- **PowerShell runbook** is based on Windows PowerShell script
- **PowerShell Workflow runbook** is based on Windows PowerShell Workflow
- **Python runbook** is a code snippet in Python

Creating an Automation account

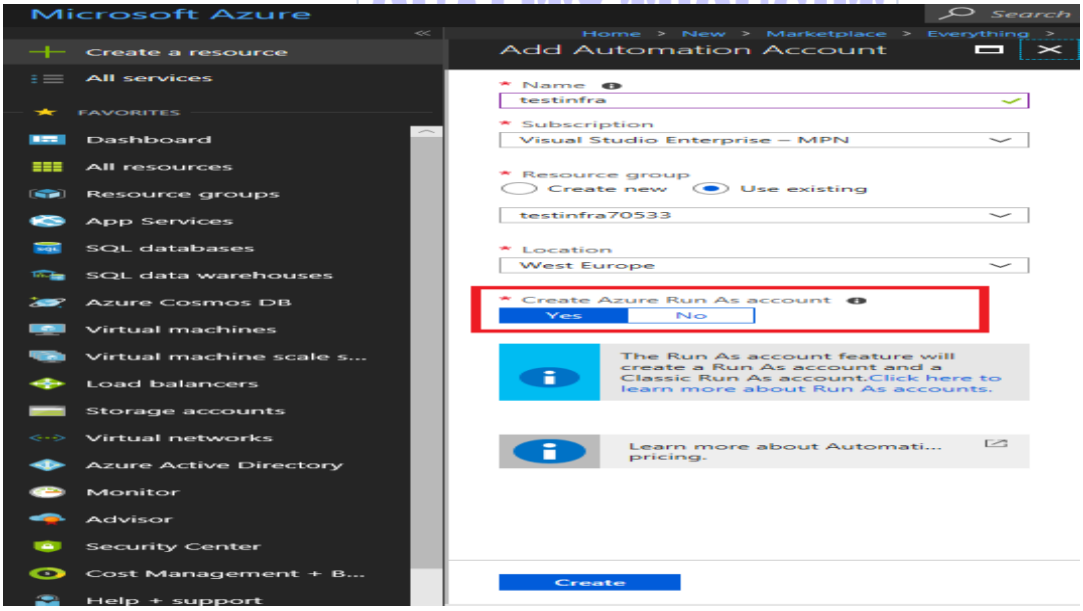
To create an Automation account, you should go to the Azure Portal and click on **Create a resource**, then you can find **Automation** in the management tool category (as shown in the following screenshot):



When you create a new Automation account via the Azure Portal, you can create an Azure Automation account by setting **Run As account** as **Yes**; this feature allows you to create a **Run As account** and a **Classic Run As account**with some useful resources such as sample script showing how to authenticate with Azure Automation or uses certificate which were automatically included for you. Azure automation use a new service principal that allows us to assign the Contributor role-based access control (RBAC) role in the subscription by default. Using this feature, users can authenticate with Azure when you're managing ARM resources using runbooks and made it possible to automate the use of global runbooks that configured in Azure alerts. To resume, there are three tasks while creating a **Run As account**:

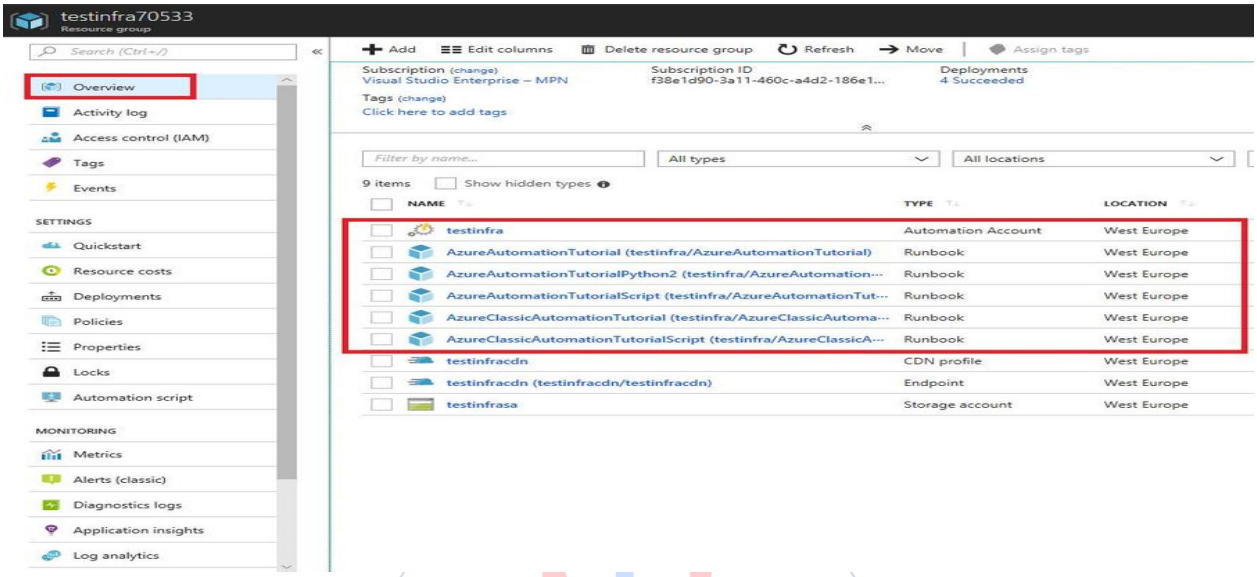
- Creating a new service principal in **Azure Active Directory** (**Azure AD**)
- Creating a one-year lifespan certificate that authenticates with Azure, so you can manage Azure Resource Manager resources from runbooks
- Assigning the Contributor Role-Based Access Control so that your role can manage Azure Resource Manager resources using runbooks

Look at the following screenshot:



After clicking on **Create**, the deployment will be launched and it will last a couple of minutes. After receiving a notification that the deployment is successful, you can go to the same

resource group and check the created resource; you can see that an Automation account has been created with some default runbooks (as shown here):



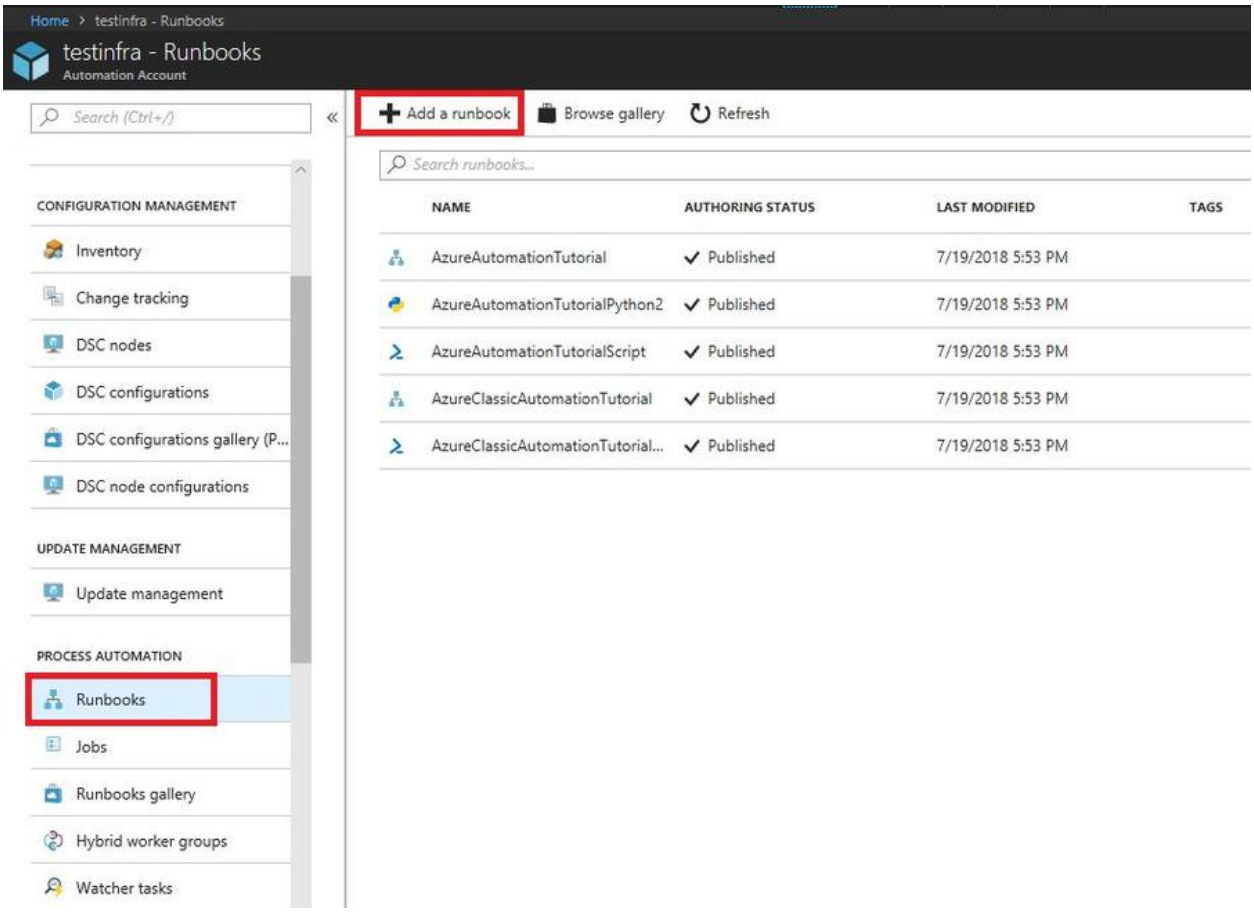
To find out how to create a Automation account using PowerShell, check the following link: https://docs.microsoft.com/en-us/azure/automation/automation-create-runas-account.

You can also create an Automation account using an ARM template. The example of an ARM template is as follows:

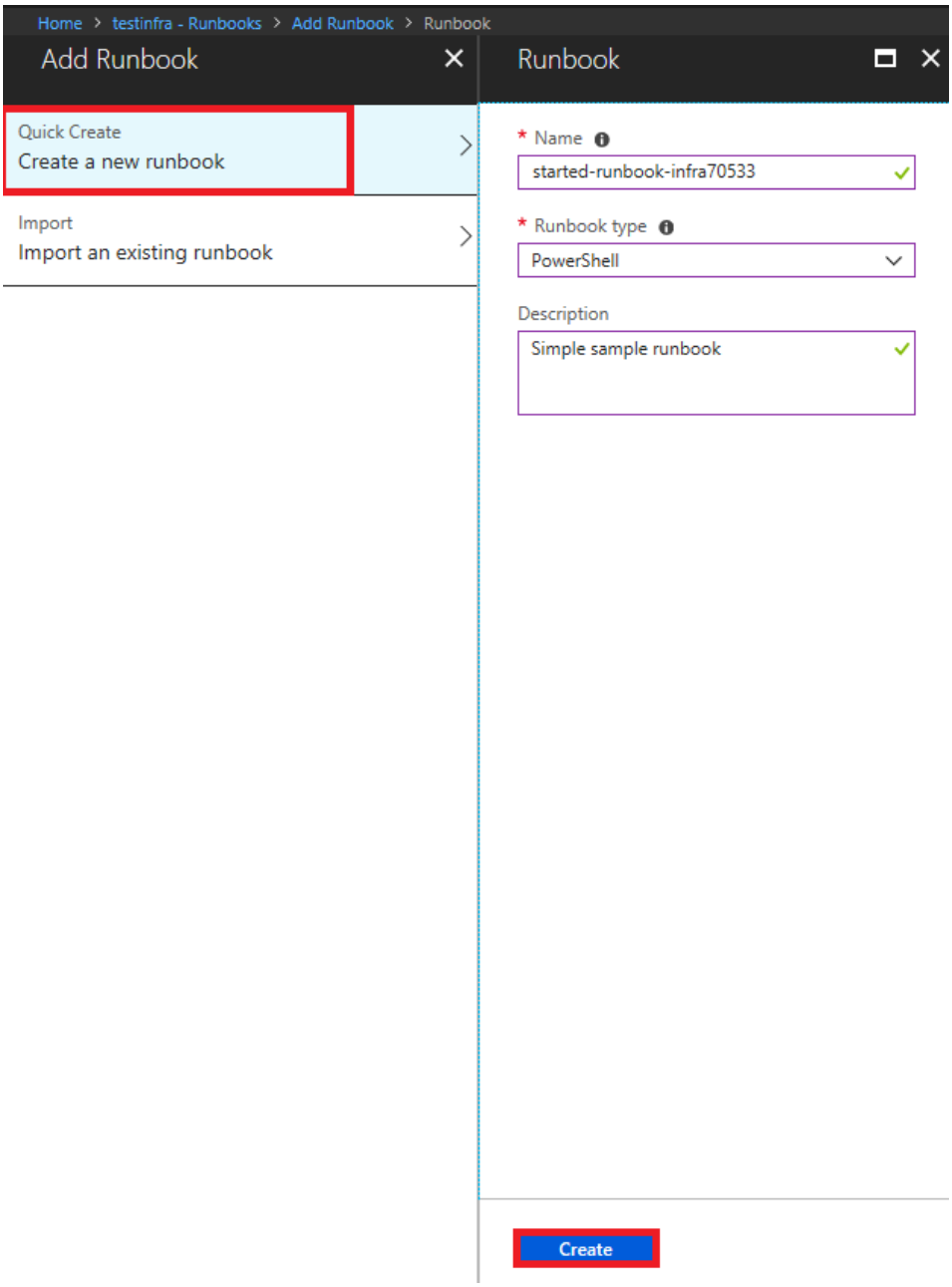

```
{
  "name": "string",
  "type": "Microsoft.Automation/automationAccounts",
  "apiVersion": "2018-01-15",
  "properties": {
    "sku": {
      "name": "string",
      "family": "string",
      "capacity": "integer"
    }
  },
  "location": "string",
  "tags": {}
}
```

Creating or importing PowerShell runbooks

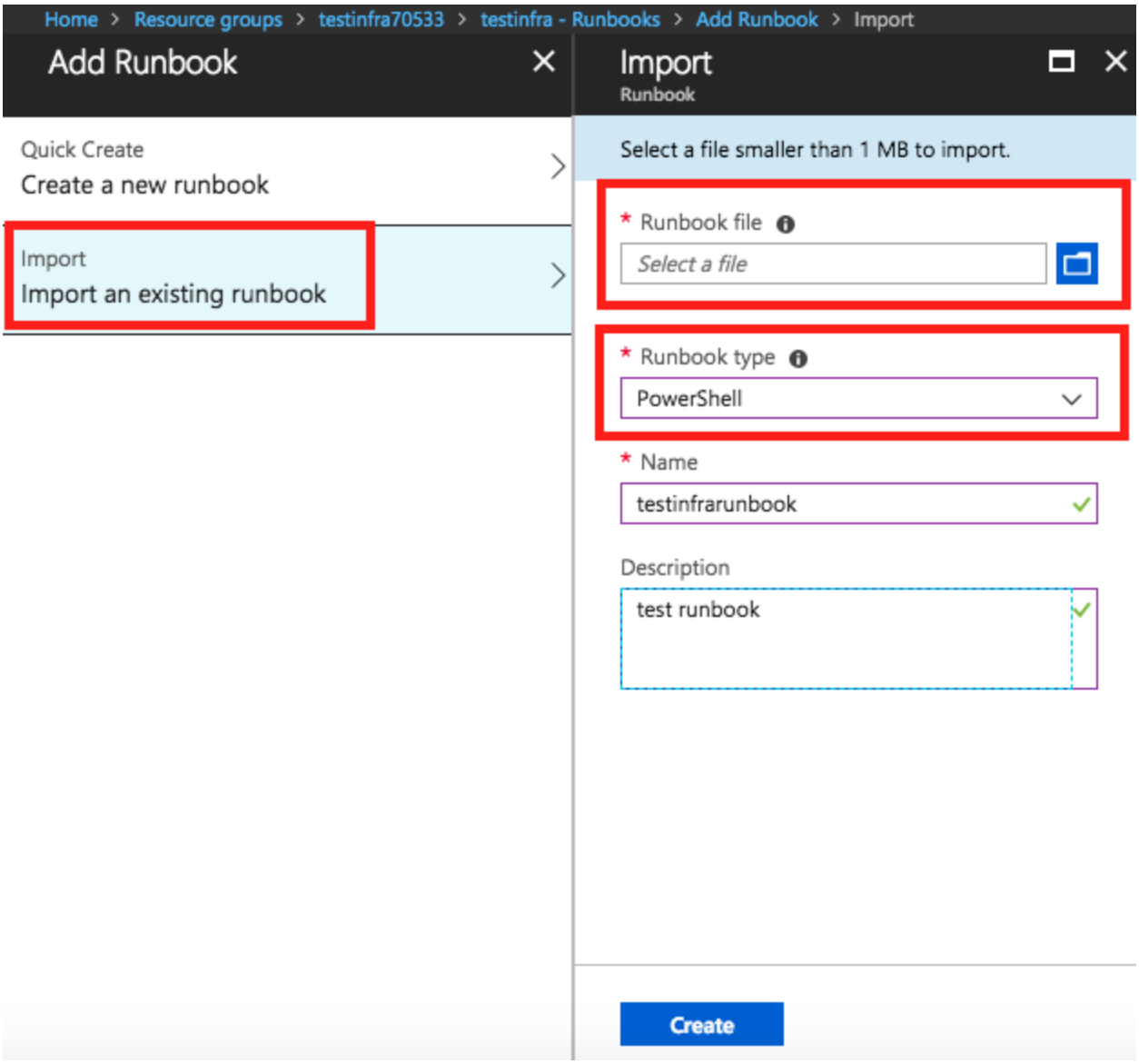
To create or import a new runbook, you can go to the **Runbooks** blade of **Automation Account** and click on **Add a runbook**, shown as follows:



You can fill in the basic information and choose an appropriate **Runbook type**, as shown in the next screenshot:



If you want to import a runbook, you should upload your runbook and choose the right type of runbook, shown as follows:



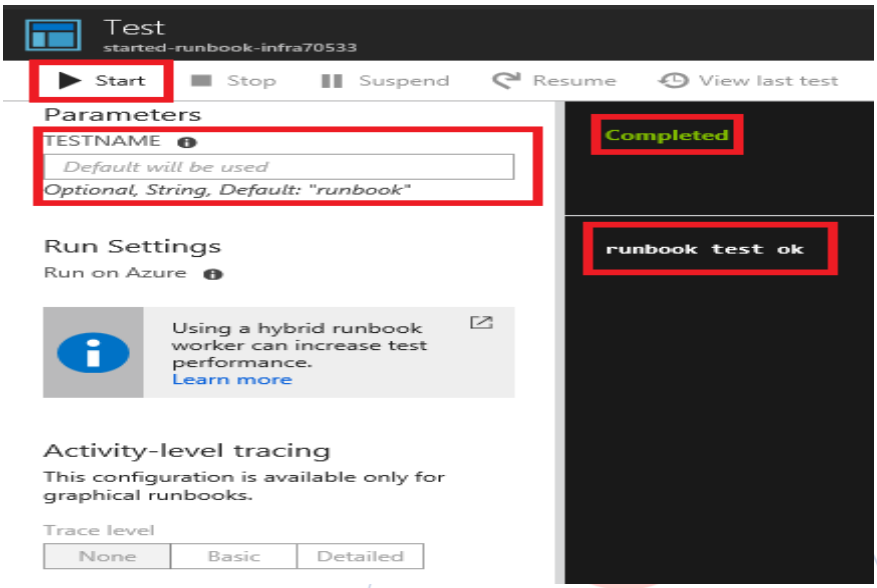
After clicking on **Create**, a new runbook will be deployed, but the runbook won't be operational instantly. You can go to the **Runbooks** blade of the Automation account, and you can see (as shown in the following screenshot), the authoring status is marked as **New**, which means that we should publish our runbooks to make it operational:

Here, we're going to test a simple sample runbook which was created previously to help you understand how to publish and test a newly created runbook. The sample runbook is as follows:

```
param
(
   [Parameter(Mandatory=$false)]
   [String] $testname = "runbook"
)
Write-Output ("$testname test ok")
```


You can find this sample runbook in the GitHub repository of this book that was given in the **Technical requirements**section of this chapter. After completely editing the content of this runbook, you can click on **Test pane** to test it (as shown in the following screenshot):

We can see that we can input the parameters in the test panel before clicking on **Start** to launch a new test as shown in the following screenshot. It is possible to set run settings to run the runbook on Azure or Hybrid environment, as shown in the next screenshot:

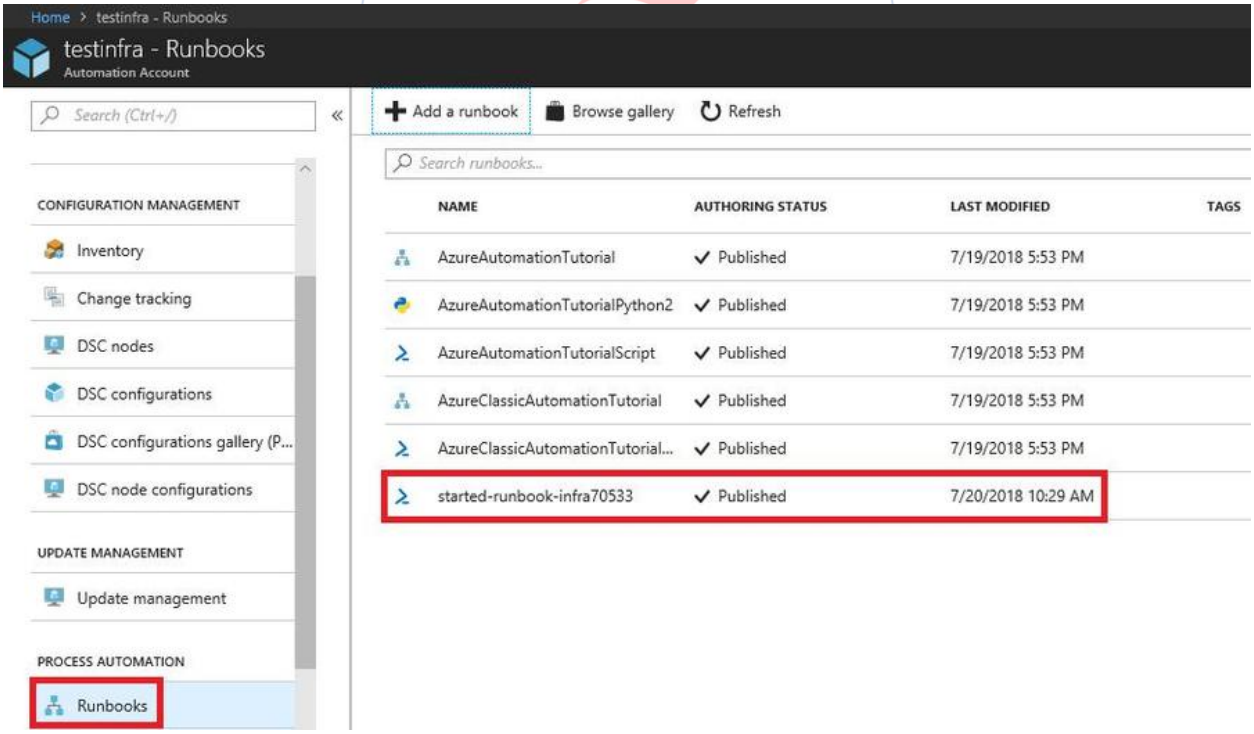


If you make sure that everything is OK, you can publish your runbook so that it is operational. You can publish a runbook in the following two ways:

- Click on the **Publish** button
- Right-click and choose **Publish** in the menu

Look at the following screenshot:

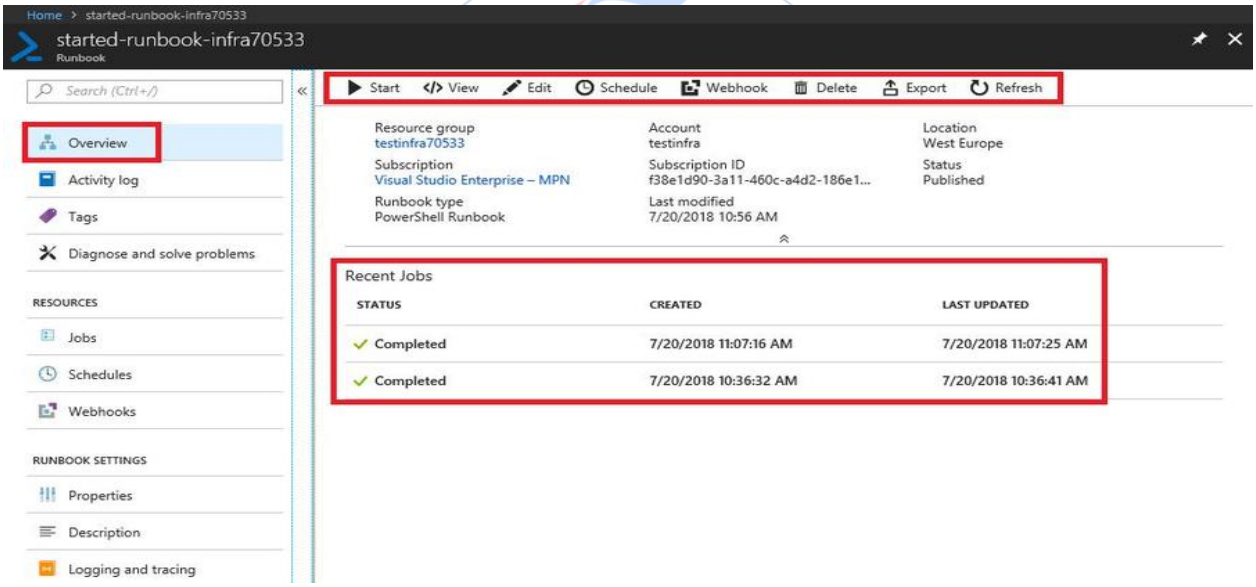
To make sure that you have published your runbook successfully, you can go back to the **Runbooks** blade of your Automation account and check the current status of the runbook. As shown in the following screenshot, the status **Published** means this runbook has been published successfully:



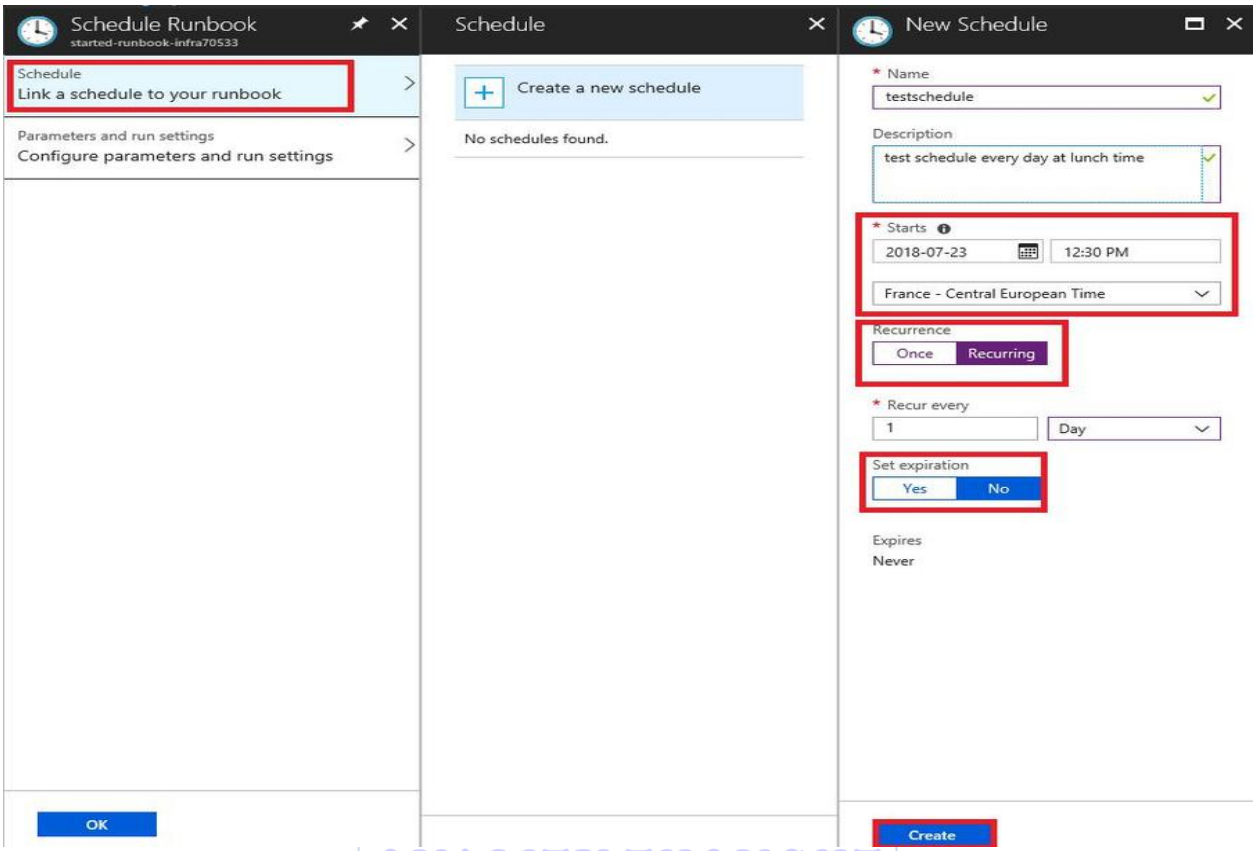
You can check the following link to know more about how to author the graphic runback in Azure Automation: https://docs.microsoft.com/en-us/azure/automation/automation-graphical-authoring-intro.

Managing PowerShell runbooks

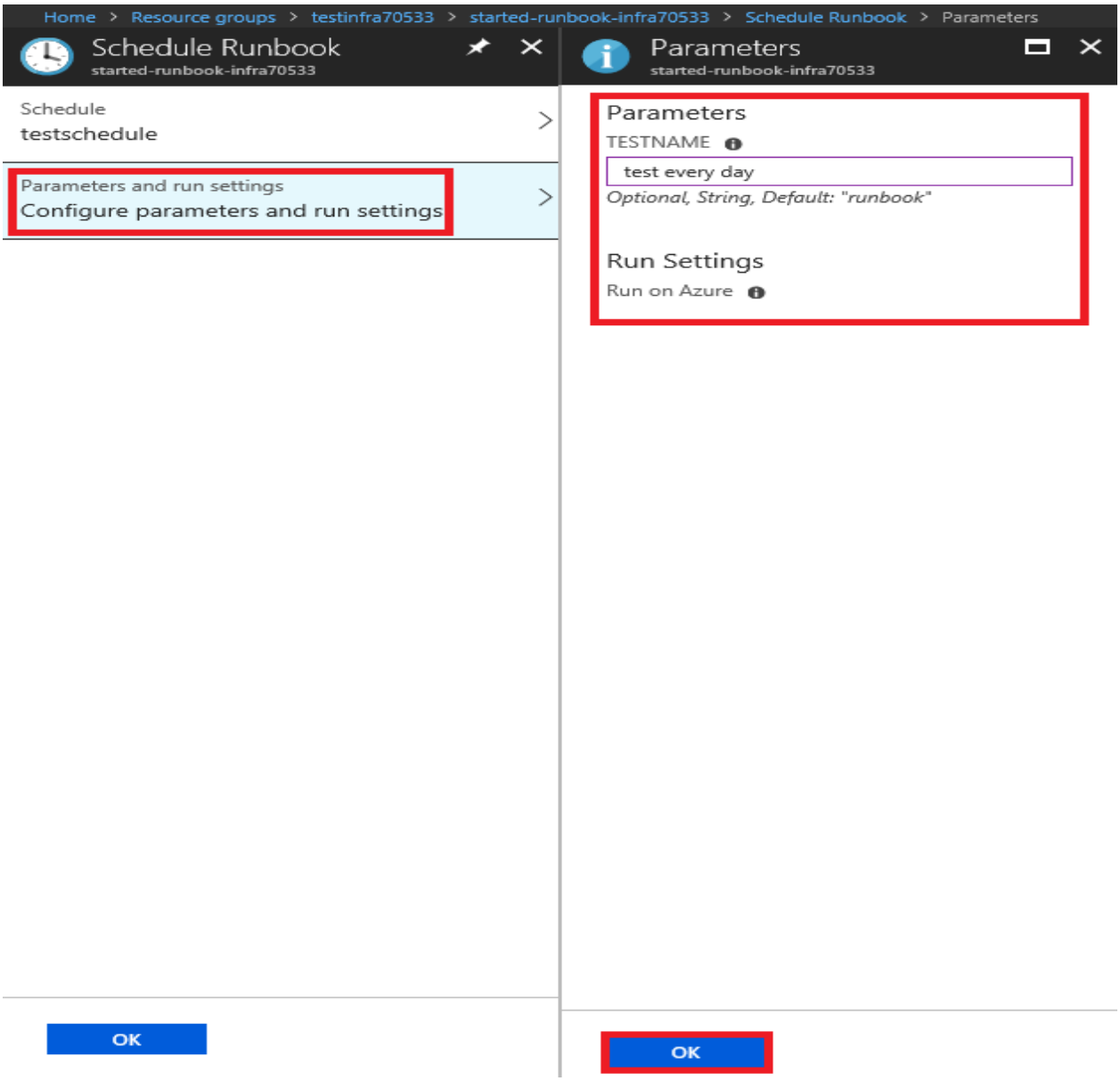
To manage a PowerShell runbook, you may go to the **Overview** blade of the runbook, and you can use the toolbar (as shown in the following screenshot) to manage your runbook. Even after publishing, you can edit (by clicking on **Edit**) an existing runbook and test it by clicking on **Start**. In the **Recent Jobs** section, you can find the historical record of the execution of the current runbook:



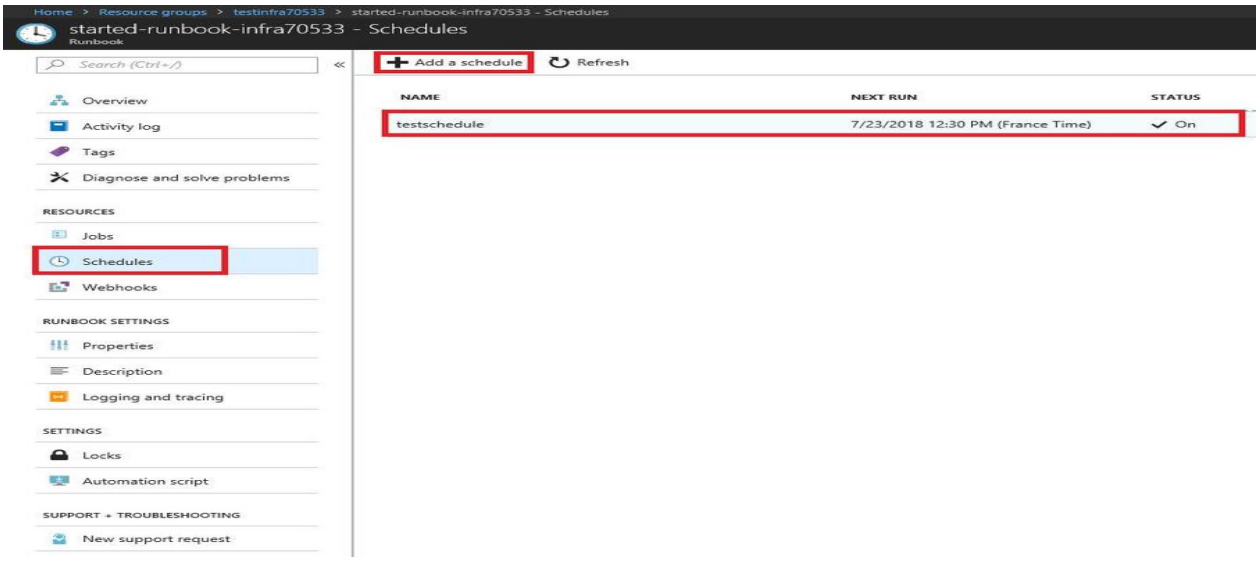
You can also configure a **Schedule** or **webhook** to trigger your runbook. To create a schedule, you can click on **Schedule** or go to the **Schedule** blade to click on **Create a new schedule**, and you can start to create a new schedule or use an existing schedule in the list (as follows):



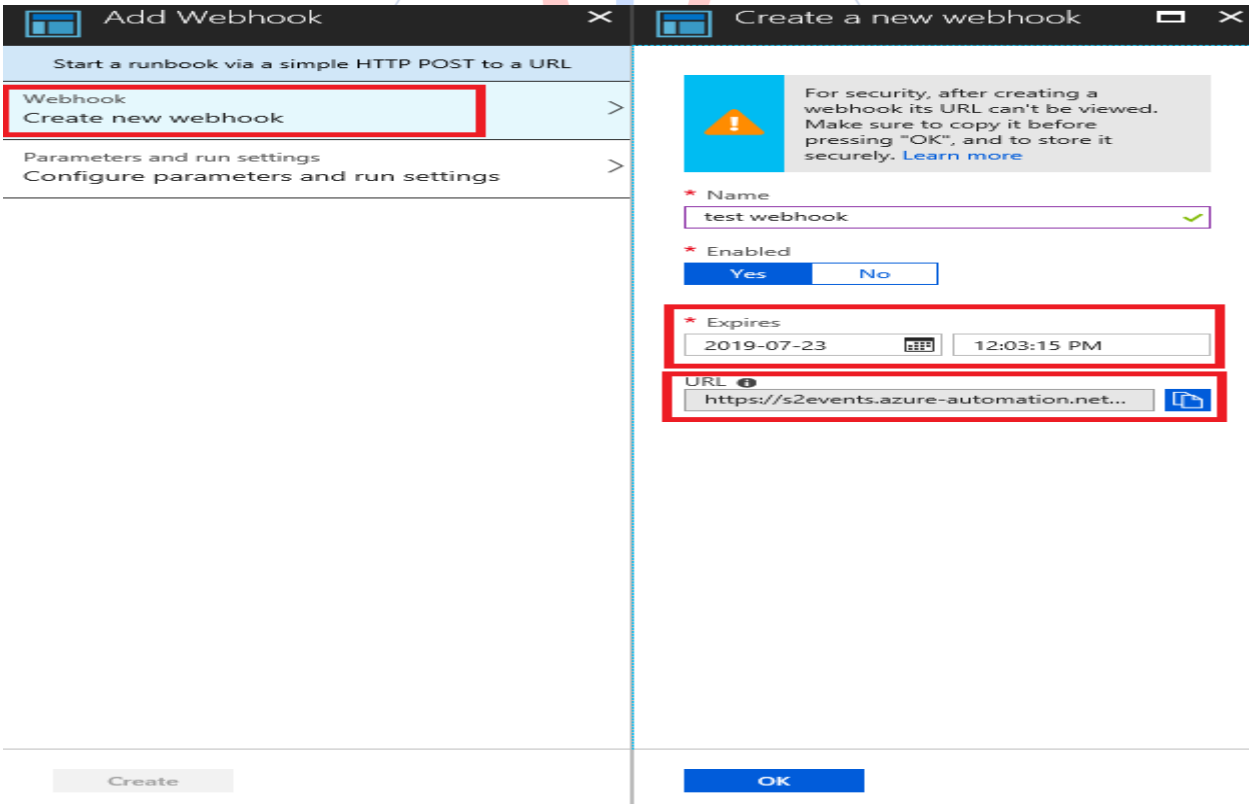
It is possible to schedule a runbook by defining it is a job to run only once or repeatedly. If you're choosing it as a repeat job, you can choose the frequency of repeat and an expiration date if you choose recurring recurrence. Then, you can also specify the input parameter when needed and click on **OK**, as follows:



You can check the **Schedules** blade to know whether your schedule has been created successfully and what the current status of the schedule is, as follows:



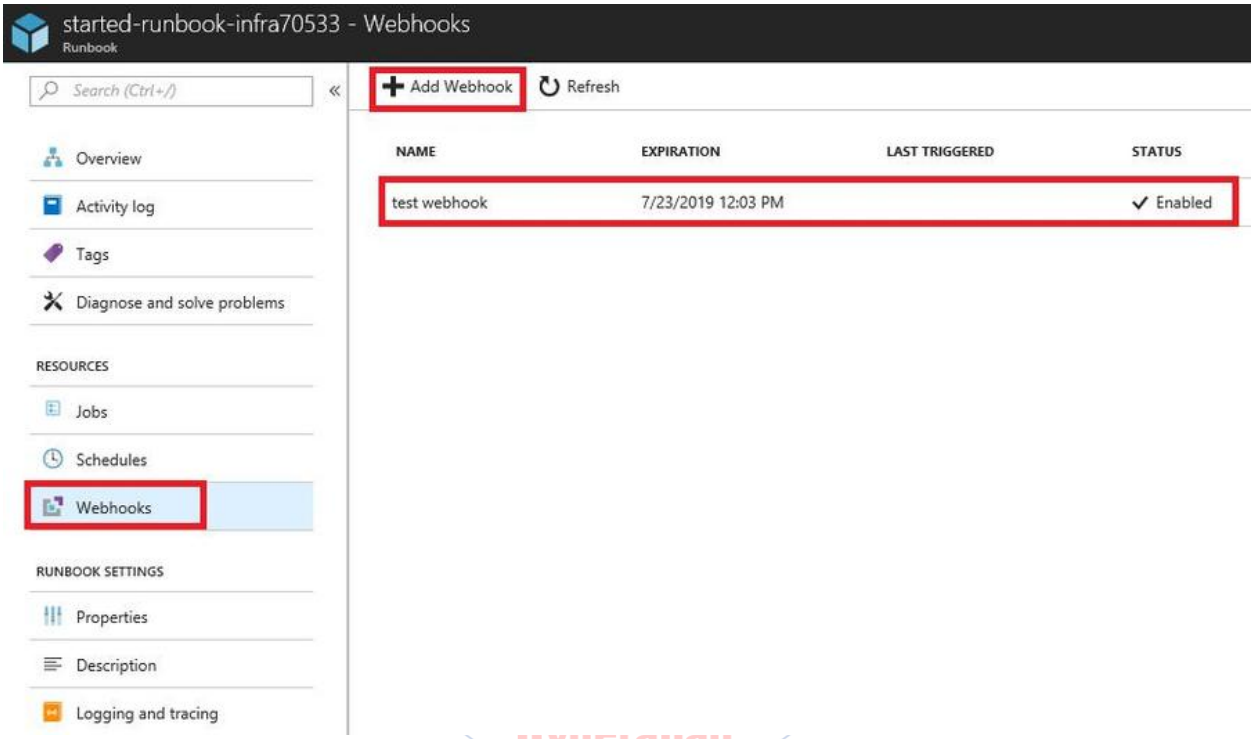
Similarly, to create a **Webhook**, you should start by clicking on **Webhook** and set a status for **Webhook** as well as the expiration date, as follows:



Here is a sample URL generated by Azure while creating a Webhook via the Azure Portal:

https://s2events.azure-automation.net/webhooks?token=buL5eq4XMJCP4c%2bVQqqbU%2fG3jnb9NjCC0RpRiHC1zgs%3d

You can check the **Webhooks** blade to know if your Webhook has been configured successfully (as shown here):



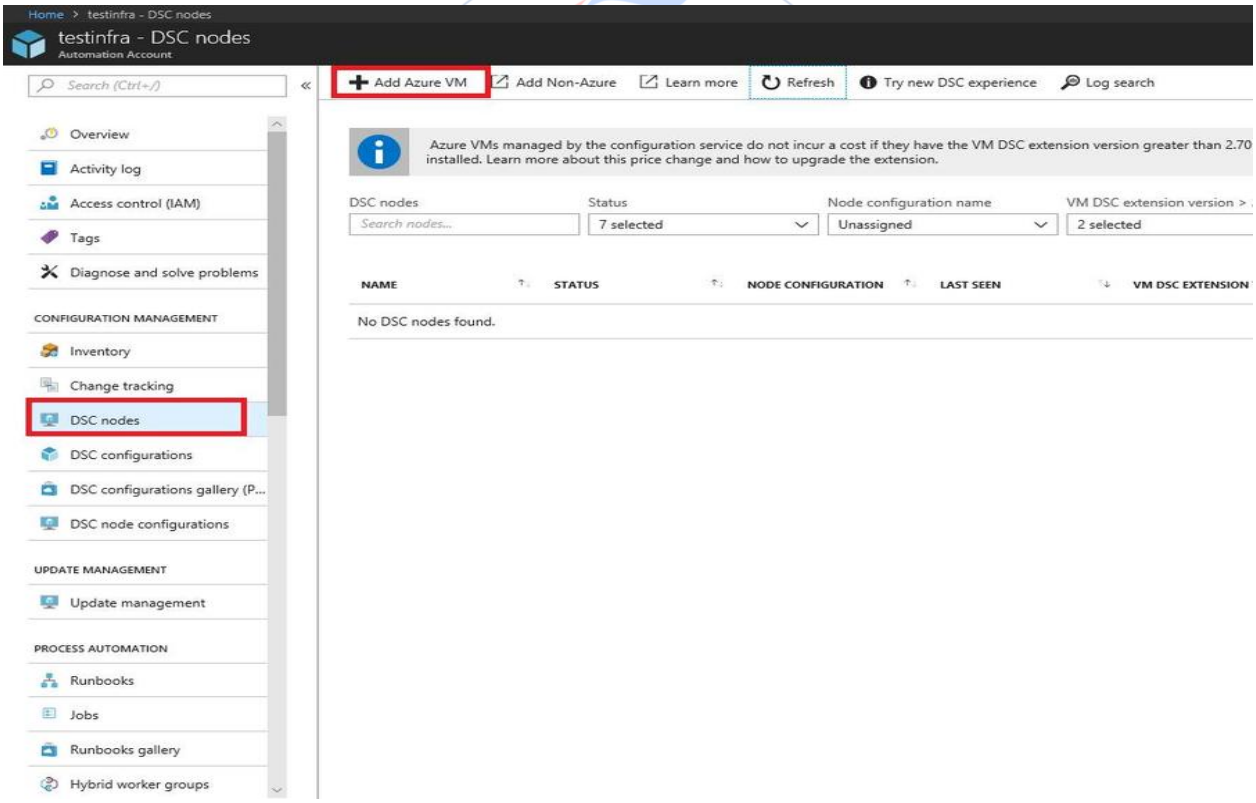
An overview of Desired State Configuration

Azure Automation **Desired State Configuration** (**DSC**) is a configuration management service in Azure. A configuration is a document that describes an environment with specific characteristics. Azure Automation focuses on **configurations**; it can configure physical and virtual machines on premises, or in any cloud provider other than Azure, it considers a single machine as a DSC node, and makes it possible to scale across thousands of based on a declarative configuration.

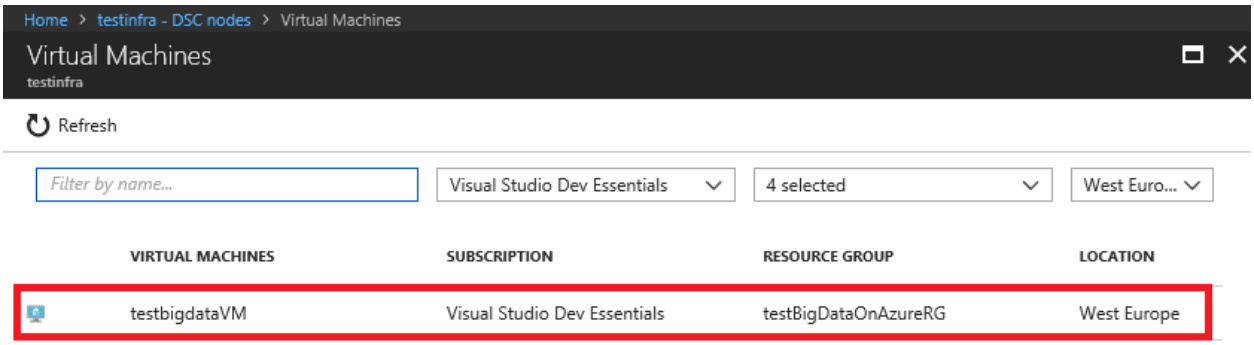
Azure Automation DSC can be used to manage a wide range of machines on premises or in the cloud such as Azure VMs (both classic and ARM-based), AWS instances, or any other cloud providers. Both physical and virtual machine, across different operating systems such as Windows or Linux.

Implementing PowerShell Desired State Configurations

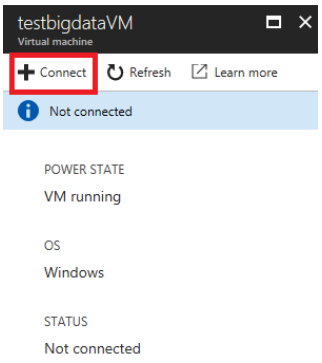
To implement the desired state configuration, you can go to the **DSC nodes** blade of your Automation account. This blade is where you virtualize all the DSC nodes in the same configuration. You can add the Azure VM or non-Azure resource here; to add a VM in Azure, you can click on **Add Azure VM**, as shown in the following screenshot:



Then, choose your target machine in the list, as shown here :



Click on **Connect** if there is already an Azure Desired State Configuration (DSC) VM extension that has been deployed in the target VM, and you'll be able to connect it:



Managing PowerShell Desired State Configurations

To import Automation DSC Configurations, you can use the `Import-AzureRmAutomationDscConfiguration` command to import a Desired State Configuration into Azure Automation. The example command is as follows (replace the words between # with your own values):

```
Import-AzureRmAutomationDscConfiguration -AutomationAccountName #yourautomationaccountname# -ResourceGroupName #yourresourcegroupname# -SourcePath #DSCscriptrepo# -Force
```

Users need to generate DSC node configurations using the Import-AzureRmAutomationDscConfigurationcommand to import an MOF configuration file into Azure Automation as a DSC node configuration. The following is a sample command:

Other excellent configuration management tools

In our world, there are a couple of tools used for configuration management, such as Chef, Puppet, and Ansible. When we talk about configuration management tools, it means a management tool that is designed to deploy, configure, and manage servers. Let's take a quick look at each of them:

- **Chef** is a popular configuration tool in the Linux world. It has master-agent architecture, meaning the server runs on the master machine and, with a client, runs as a client agent on each client machine. Besides master and agent, there is a controller machine that contains all the configurations. Users can use that machine to push all the configurations to central Chef server. Chef is a great choice for users who know Git and Ruby well.
- **Puppet** is also a famous term in the Linux world and has a master-agent architecture. Similar to Chef, Puppet server runs on the master machine, and Puppet clients run as an agent on each client machine. In addition, there is a certificate signing between the agent and the master.
- **Ansible** is an open source automation tool that allows to bootstrap machine and generates environment, different to master-agent architecture. It uses SSH connection to log in to client systems or the nodes you want to configure. It is an awesome tool in the configuration world, and as it is based on Python; it is easy to use for Python developers to personalize some features. The greatest advantage of Ansible is that it is SSH-based; hence, it doesn't require installing any agents on remote nodes.

Implementing Azure Automation-based cloud management

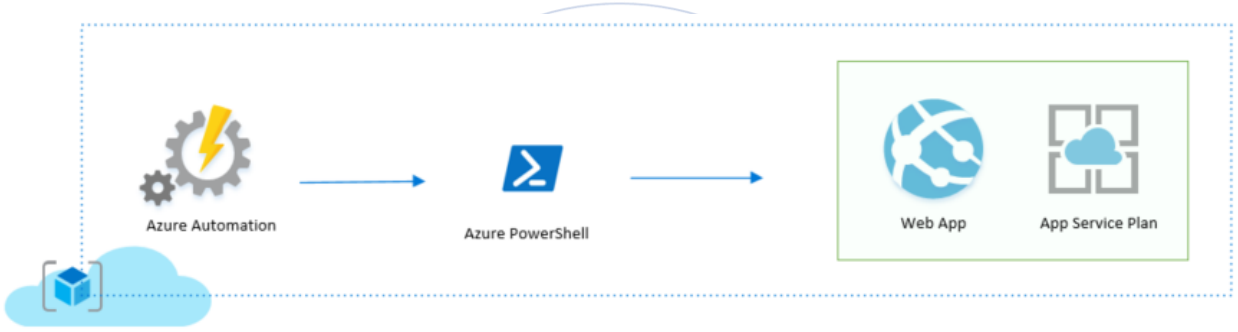
Azure Automation is not only a cloud-based automate service. It can also be used for simplifying cloud management through process automation. We'll introduce some scenarios to show how Azure automation interacts with other Azure services.

Integrating Azure Automation with Web Apps

We know that Web App is a PaaS offering provided in Azure. As Azure PowerShell provides a wide range of commands to manage Web App in an App Service plan, it is also possible to use Azure Automation to interact with Azure Web App. The general scenario is:

- To stop or start Web App automatically
- Scale the App Service plan from a single instance to multiple instances
- Scale Web App in multiple regions with a high-availability architecture
- To create a one-time or scheduled backup of Web App

I am using the following diagram to show these scenarios:



You can find the most useful PowerShell sample with the following link: https://docs.microsoft.com/en-us/azure/app-service/app-service-powershell-samples.

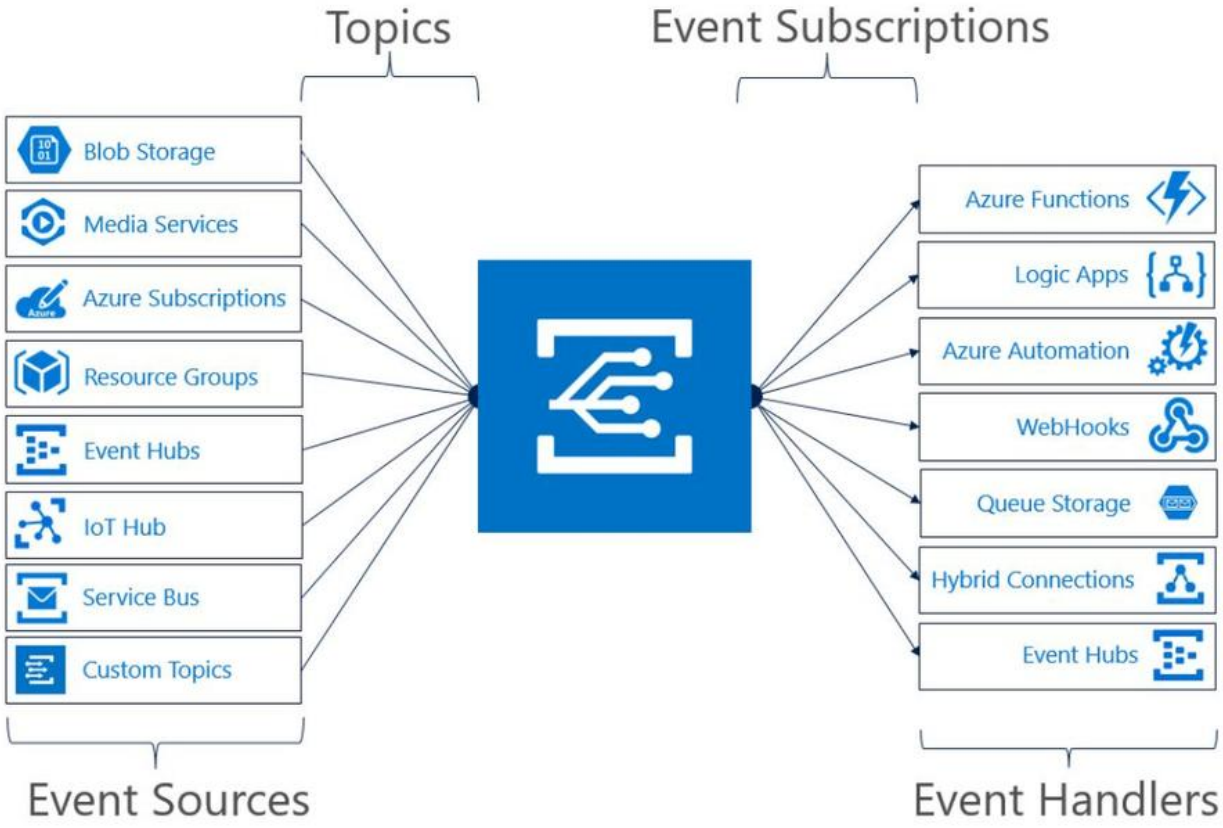
Comparing Azure Automation with Azure Functions

Both Azure Automation and Azure Functions can run PowerShell Scripts on Azure. They both support Webhooks and can be scheduled. However, Azure Functions are far richer in terms of triggers and liaisons, and they support a wide range of languages to write code running in the cloud not limited to PowerShell. As a Function as a Service (FaaS), Azure Functions can be used to build microservices at scale.

Here is the link to find out more about the triggers and bindings of Azure functions: https://docs.microsoft.com/en-us/azure/azure-functions/functions-triggers-bindings.

Integrating with Event Grid

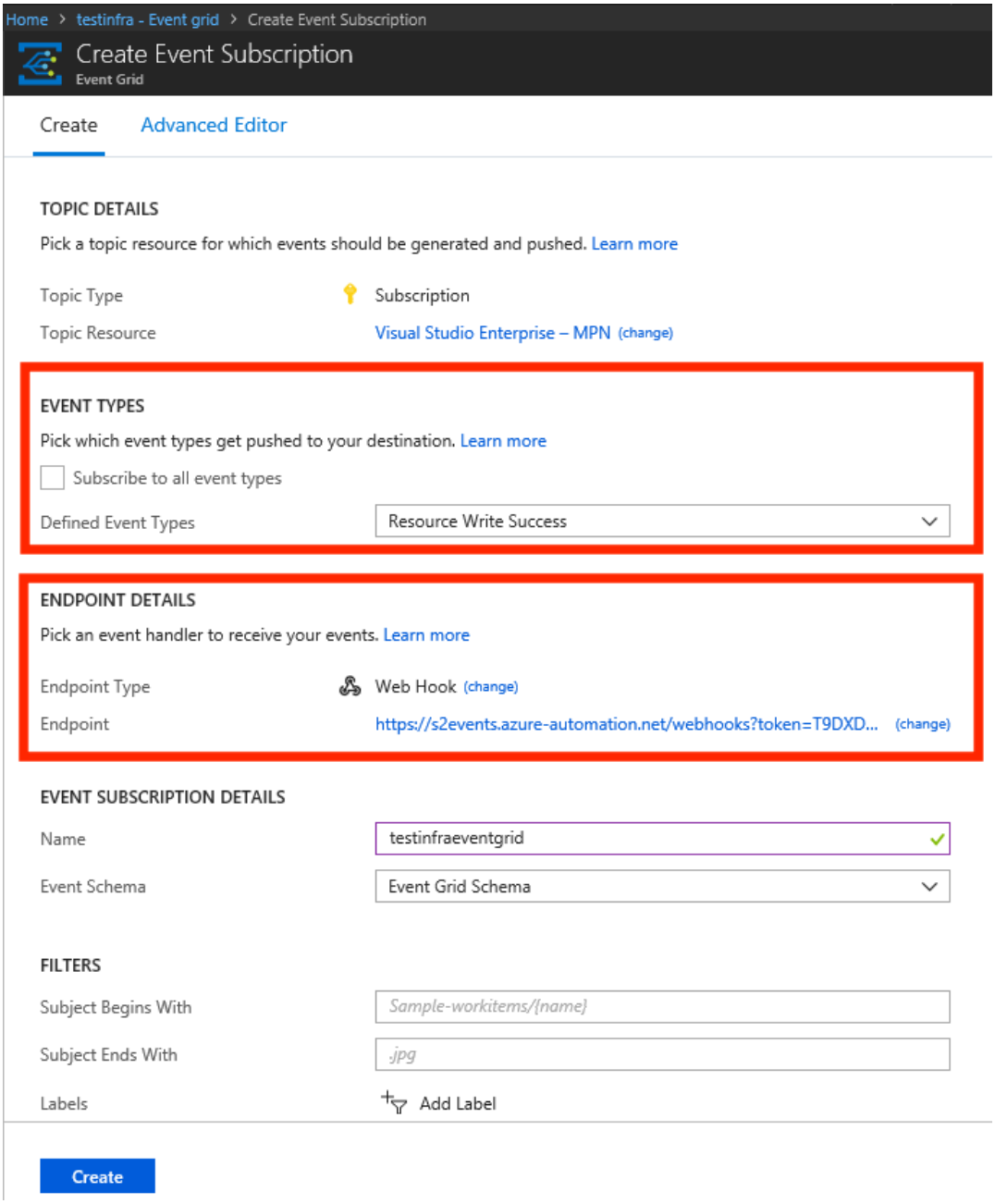
Azure Event Grid is a cloud-based publisher and subscriber service that helps users to build event-based architecture for their application. Users can select Azure resource that is subscribed and give the event handler or **WebHook**endpoint to send the event to. Azure Event Grid is focusing on providing the capabilities of Automation and integration with other Azure Services to build powerful cloud-based solutions, as shown in the following diagram:



Event Grid (https://github.com/MicrosoftDocs/azure-docs/blob/master/articles/event-grid/overview.md)

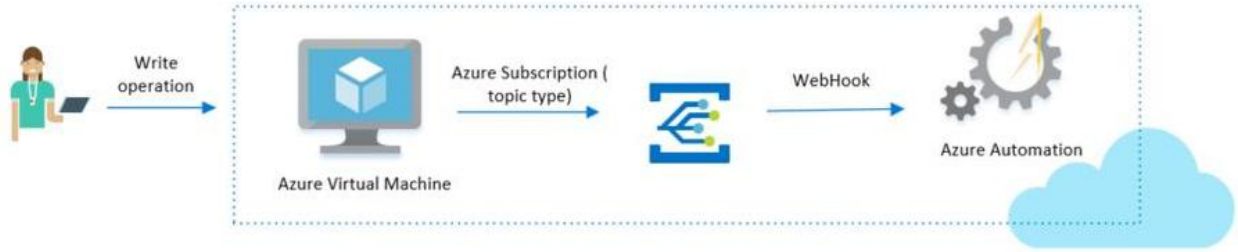
Event grids can be integrated with a wide range of other Azure Services, thanks to its pub-sub pattern. The first thing you should do is create an **Event Subscription**. This step is quite important because you can define the event type that means the kind of events as source, also,

the endpoint type that is the **trigger** for this event. You can see how to create an Event Grid subscription as follows:



For example, you can use Event Grid to catch all the write operations on an **Azure Virtual Machine** and then trigger an **Azure Automation** script that can log in to all the write operations with a text file by using **WebHook**. Here is a diagram of this interaction. You can

do more powerful things with Event Grid while being integrated with other Azure Services, shown as follows:



To trigger Stream big data into the Azure data warehouse, refer to the following link:

https://docs.microsoft.com/en-us/azure/event-grid/event-grid-event-hubs-integration

Service Bus is a cloud-based multitenant service that allows you to connect applications through the cloud. You can check the following link to find out more on how Event Grid responds to a Service Bus event:

https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-to-event-grid-integration-example?toc=%2fazure%2fevent-grid%2ftoc.json

Integrating with Logic Apps

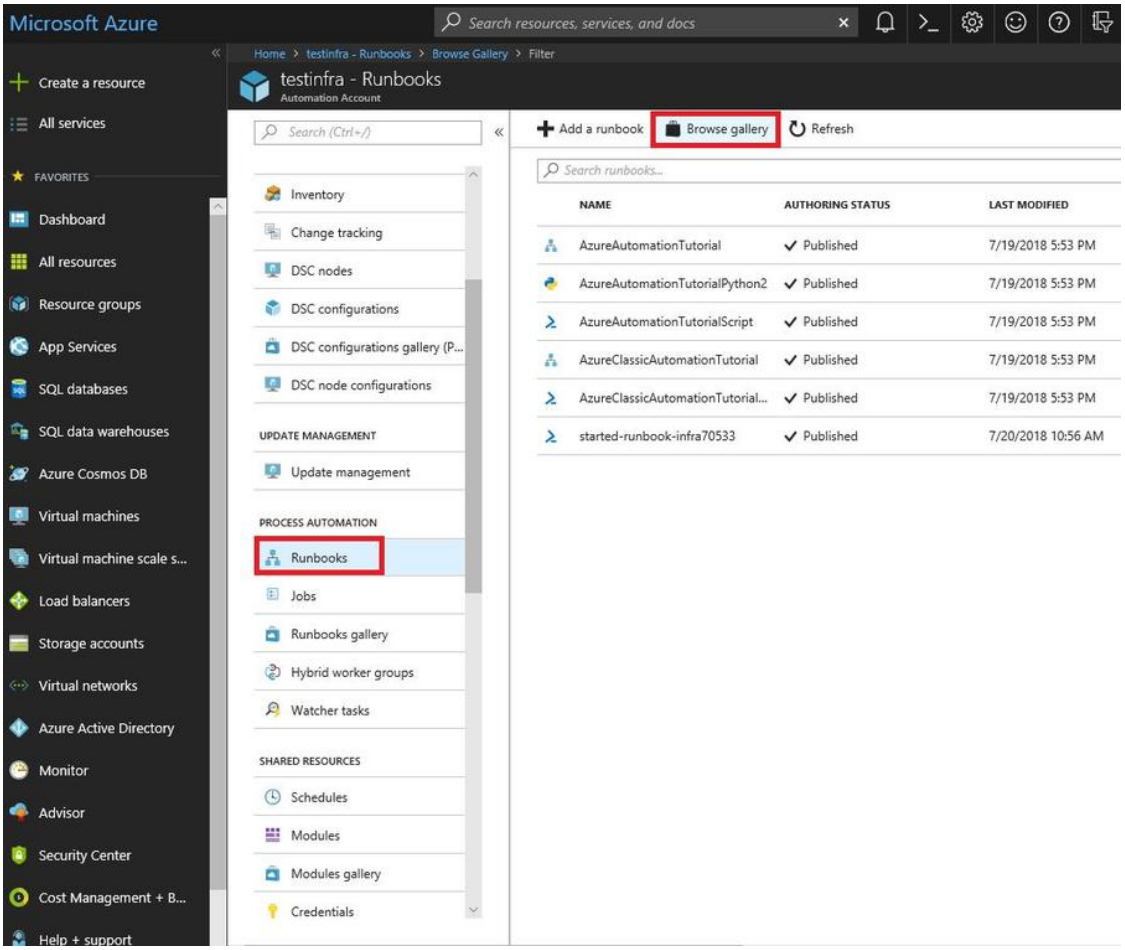
Logic Apps is a great PaaS offering to implement scalable logical workflows for cloud-native applications. Especially for those who are more comfortable designing workflows with graphic interfaces, it provides a visual designer to model and automate the process and the workflow with a couple of steps. Every Logic Apps workflow begins with a trigger and can execute the combinations of actions with conditional logic, in parallel, and sequentially.

For a great post about how Azure Automation works together with Logic Apps, please check this link:

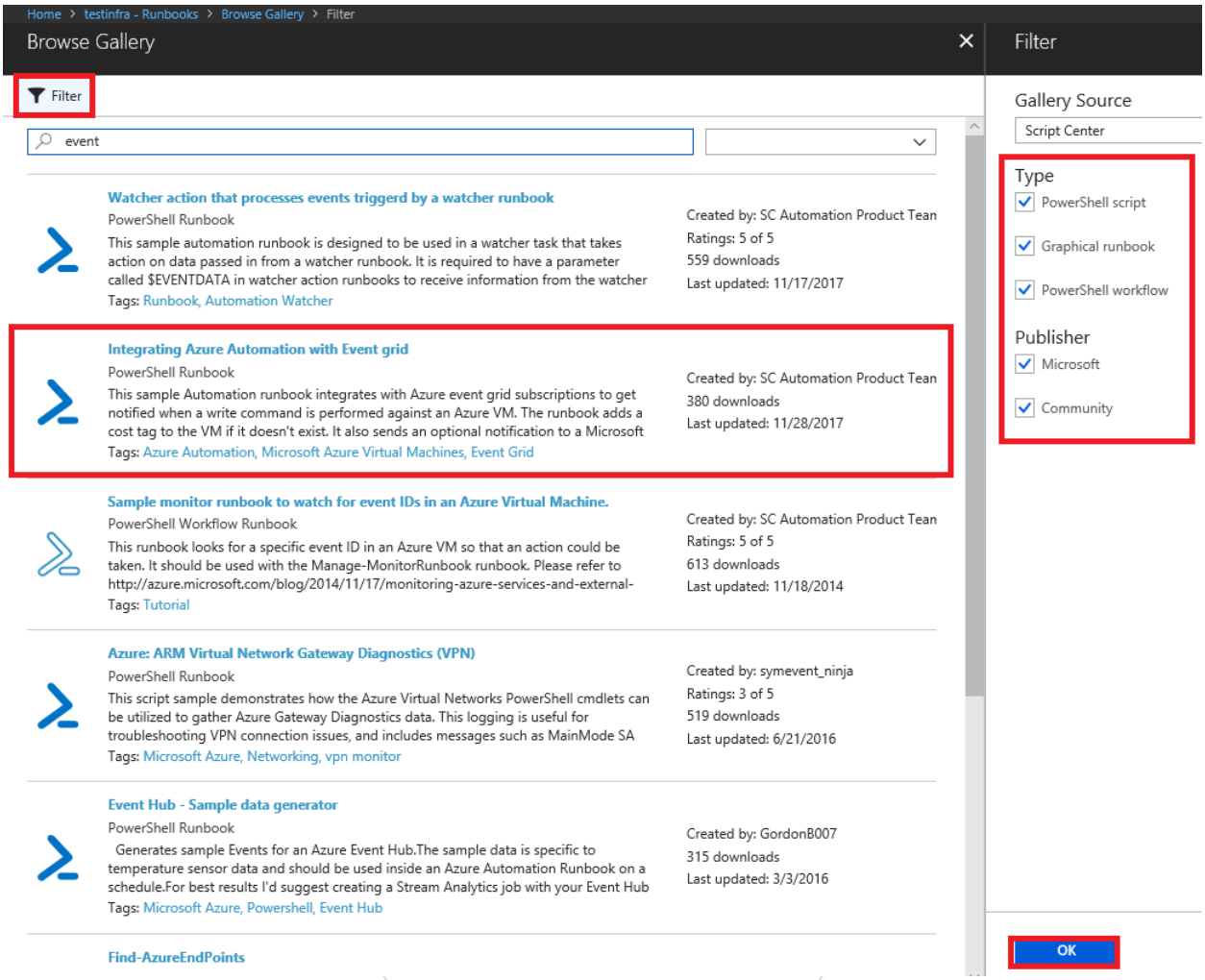
https://blogs.technet.microsoft.com/stefan_stranger/2017/06/23/azur-logic-apps-schedule-your-runbooks-more-often-than-every-hour/

Runbook gallery

To know more about how Automation interacts with other Azure services, you can use existing runbooks in Runbooks gallery, which you can find in the **Runbooks** blade, and click on **Browse gallery**, as follows:



In the gallery, you can find the runbook conform to your search criteria using **Filter**; it will help you find the runbook you need in an effective way. See the following screenshot



Implementing monitoring solutions in Azure

Monitoring in Azure covers the performance, health, and availability of your Azure resources to help users analyze issues and detect problems in case of failure.

Azure includes a couple of services performing monitoring tasks; they can work individually on telemetry or work together to provide a complete monitoring strategy for your application. There are the following two modes of monitoring in Azure:

- **Core monitoring** aims to provide fundamental and required monitoring across Azure resources.

- **Deep monitoring** includes services that go beyond core monitoring. It provides capabilities that collect and analyze data at a deeper level. There are two types: **deep application monitoring** and **deep infrastructure monitoring**.

The following diagram shows a conceptual view of the components that work together to provide monitoring of Azure resources:

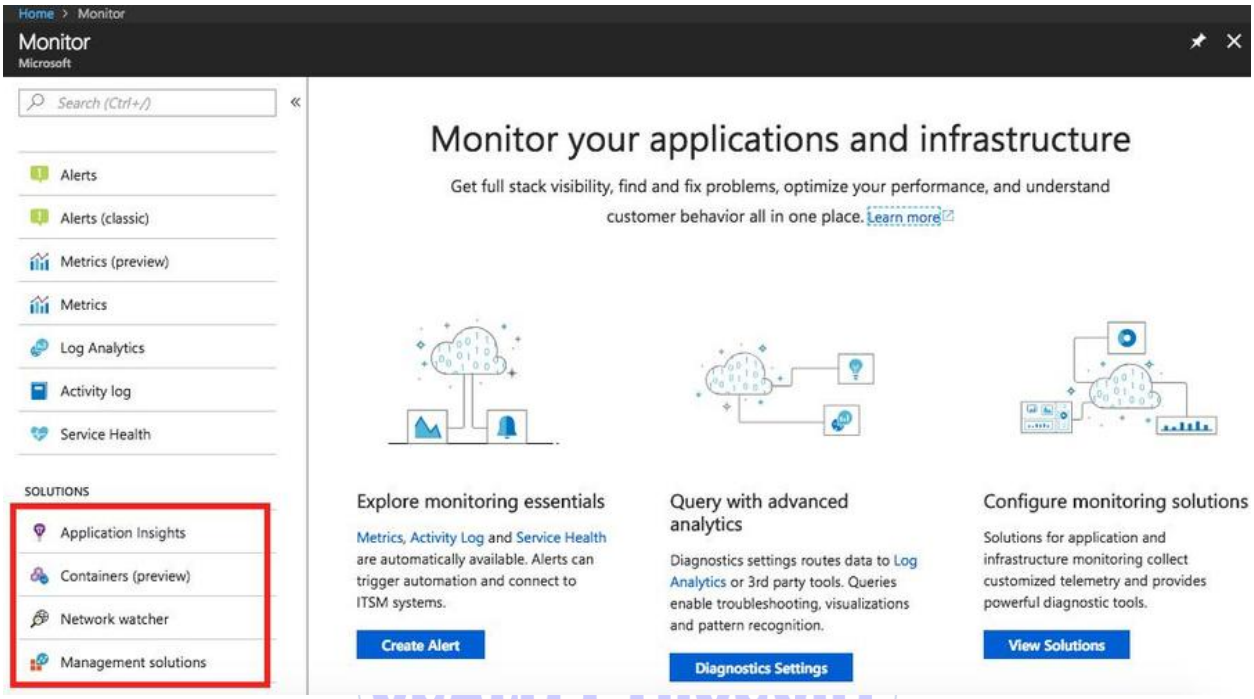
Core monitoring

In this section, let's take a look at each service that performs fundamental and required monitoring across Azure resources.

Azure Monitor

Azure Monitor is a monitoring solution for applications and infrastructure in Azure where users can get full stack visibility, get helping finding problems and resolutions, and understand customer behavior. It provides base-level infrastructure metrics and logs for many services such as cloud service, virtual machine, virtual machine scale sets, and service fabric in Microsoft Azure.

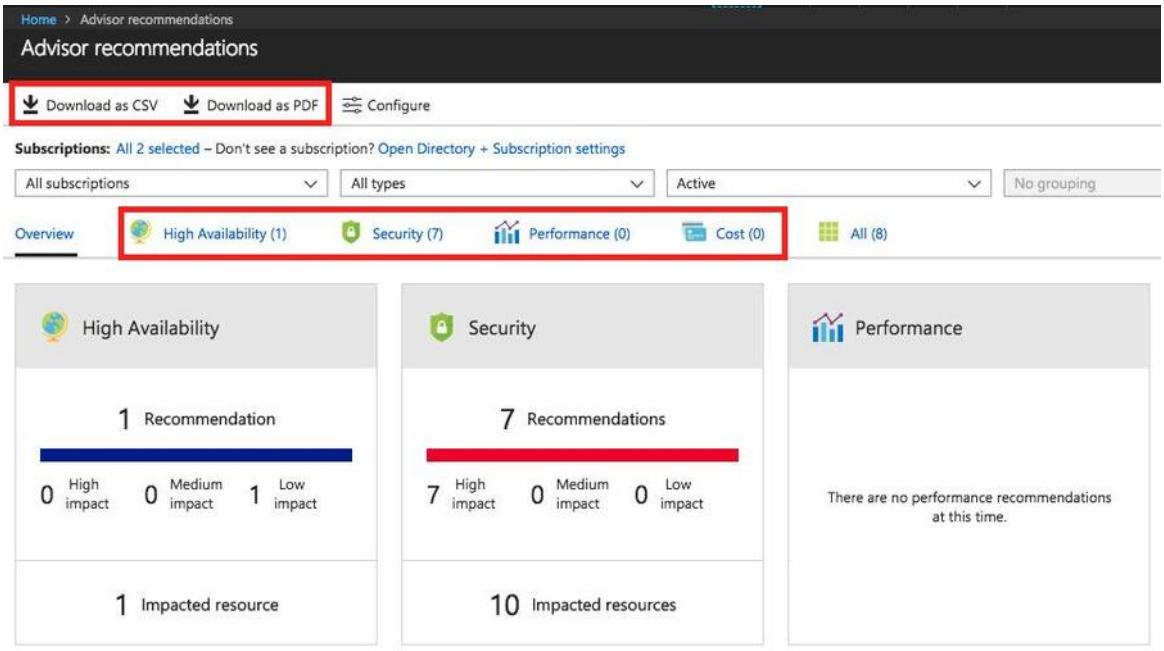
It also supports monitoring container solutions in Azure and connections to **Application Insights**, **Network watcher**, and **Management solutions** (OMS workspace) as well, shown as follows:



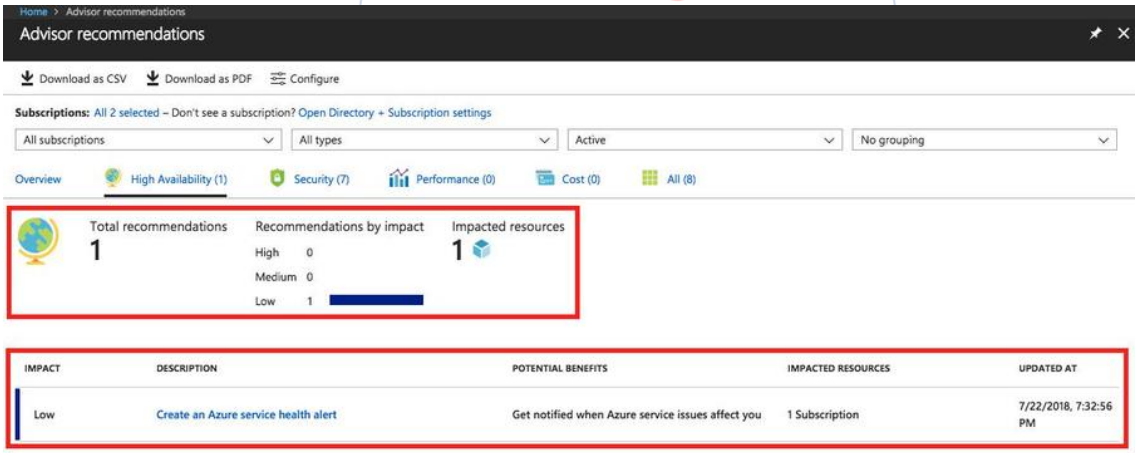
Azure Monitor collects most application-level metrics and logs such as **Application Logs**, **Windows Event Logs**, **.NET Event Source**, **IIS Logs**, and **Customer Error Logs** using diagnostics extension.

Azure Advisor

Azure Advisor is a personalized advisor that Azure provides for you. If want to to know more about Azure, it will be your best friend on your cloud journey. Azure Advisor helps you follow best practices to optimize your Azure deployments and analyzes your existing resource configuration, usage telemetry, and so on. It recommends a best solution in terms of cost, performance, high availability, and security and support to export these recommendations by PDF or CSV file, as shown in the following screenshot:

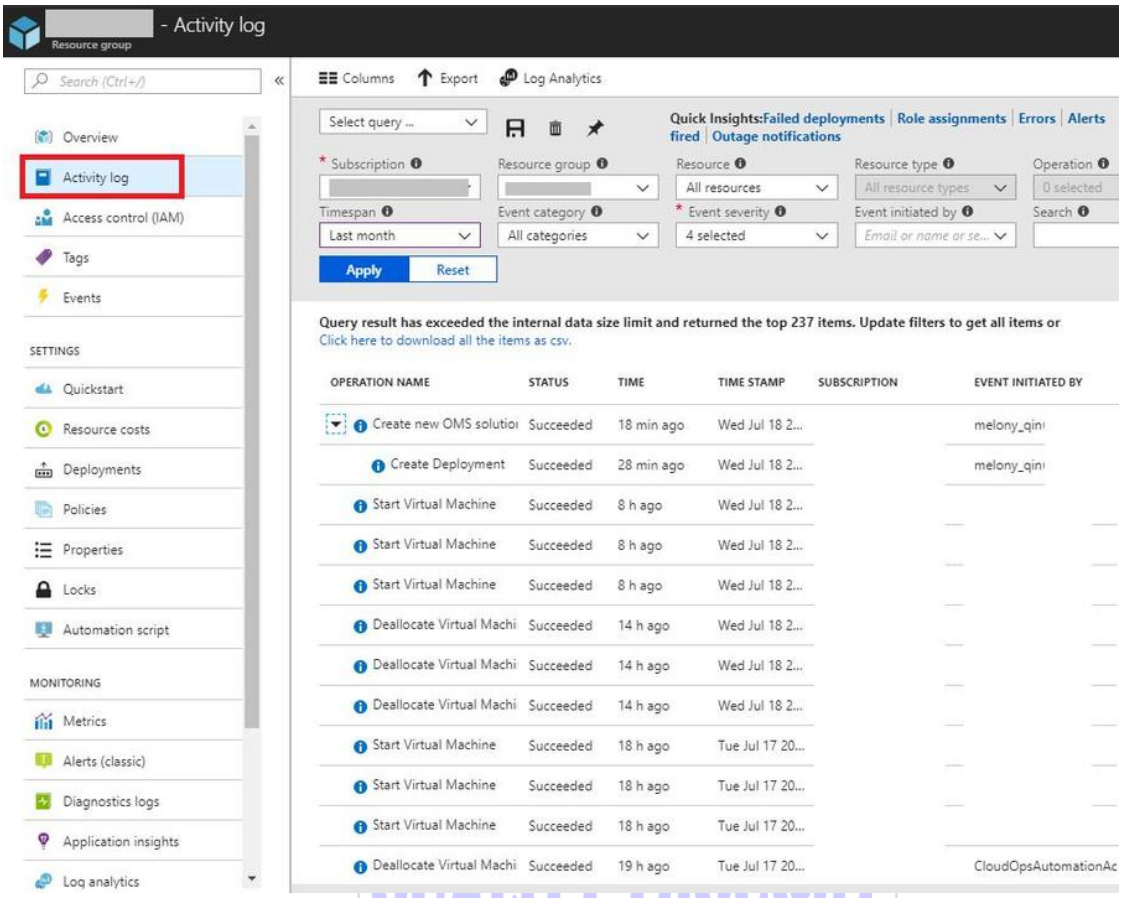


As the picture shows, you can click on the recommendation such as **High Availability**, and Advisor will give you further details about its recommendation, as shown here:



Activity log

Activity log is a log to trace events such as configuration changes, incidents, and operations related to the dedicated Azure resource. Users can view logs from the Azure Portal, as shown in the following screenshot:



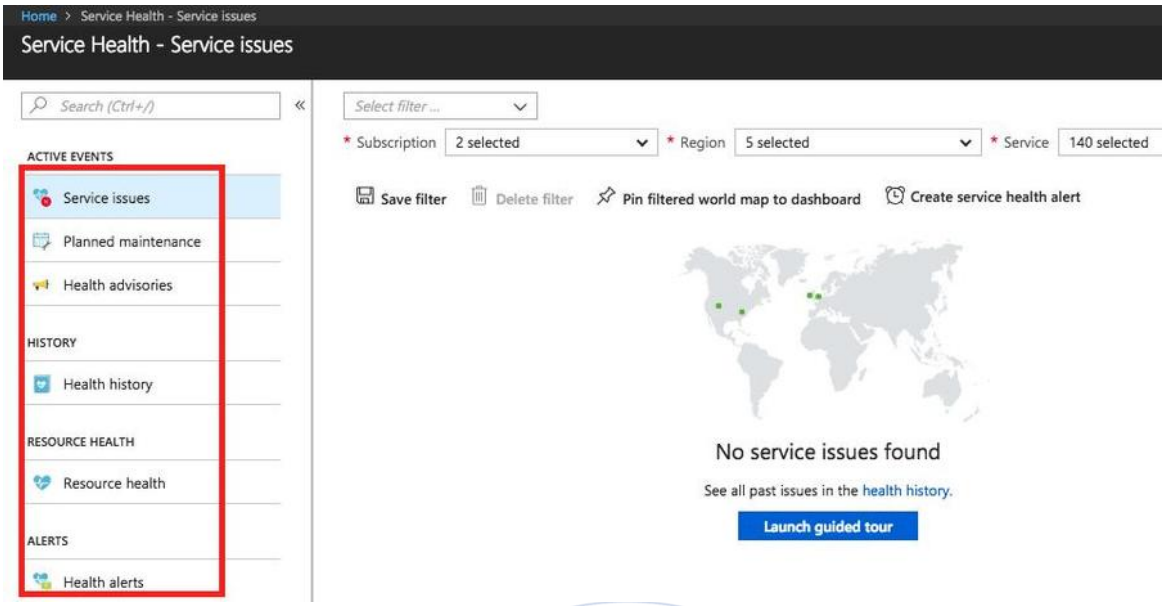
Service health

Service health can help users to receive guidance of service issues and performs notification of information about the resources under users' subscriptions. This information is the subclass of activity log events, which means the same information can also be found in the activity log. There is a wide range of classes of service health notifications in terms of **Action required**, **Assisted recovery**, **Incident**, **Maintenance**, **Information**, and **Security**.

To know how to create activity log alerts on service notifications, you can refer to:

https://docs.microsoft.com/fi-fi/azure/monitoring-and-diagnostics/monitoring-activity-log-alerts-on-service-notifications

After creating a notification, you can check it at Azure Portal by clicking on **All services** and search Service health, you can go to the dashboard of **Service Health** where you can get the global vision of this service (as shown in the following screenshot):



Deep application monitoring

Application Insights plays a key role in deep application monitoring solutions provided by Azure; it can run on top of the Guest OS in the compute model. In the previous chapter, we introduced how to monitor Web App in App Service Plan using Application Insights. Actually, Azure Application Insights can not only be used for monitoring applicationshosted in App Service Plan (API App, mobile app, and so on) but also for Azure Functions.

Azure Functions has been integrated with Azure Application Insights since April 2017. Users can find out how to configure Functions to send telemetry data to Application Insights and how it works through the following link:

https://docs.microsoft.com/en-us/azure/azure-functions/functions-monitoring

Application Insights also supports monitoring Docker-based applications in Azure; you can check here to get more information:

https://docs.microsoft.com/en-us/azure/application-insights/app-insights-docker?toc=%2fazure%2fmonitoring%2ftoc.json

Deep infrastructure monitoring

In this section, let's take a look at each service that performs infrastructure-level monitoring in depth across Azure resources.

Log Analytics

Log Analytics is also a very important role in Azure monitoring. It focuses on collecting data across Azure resources into a single repository, and it allows queries using **Log Analytics query language** (**KQL**) and analyzes collected data. Azure services such as Application Insights, Azure Security Center, Azure Monitor, Management Solutions, and agents installed on virtual machines in the cloud or on-premises can store data in the Log Analytics data store. Log Analytics acts as a core monitoring service in Azure. We'll dive deeper into it later in this chapter.

Management solutions

Management solutions are based on the monitoring data collected by Log Analytics and analyze them so that it can provide a global vision for a particular application or service. To know more about how to use and install the management solutions go to https://docs.microsoft.com/en-us/azure/monitoring/monitoring-solutions.

Network monitoring

There are a variety of tools that work together to provide a comprehensive monitoring solution in Azure or on-premise from various aspects of networking, and are as follows:

- **Network Watcher** performs monitoring, metrics, and enables or disables logs for resources in an Azure VNet. It stores data in Azure metrics and diagnostics for further analysis.
- **Network Performance Monitor** (**NPM**) is a monitoring solution that monitors connectivity across public clouds and on-premises data centers in the cloud.
- **ExpressRoute Monitor** is a subcapability of NPM, focusing on monitoring the end-to-end connectivity and performance over Azure ExpressRoute circuits.
- **DNS Analytics** is a service based on DNS servers that monitors the security, performance, and operations-related insights.

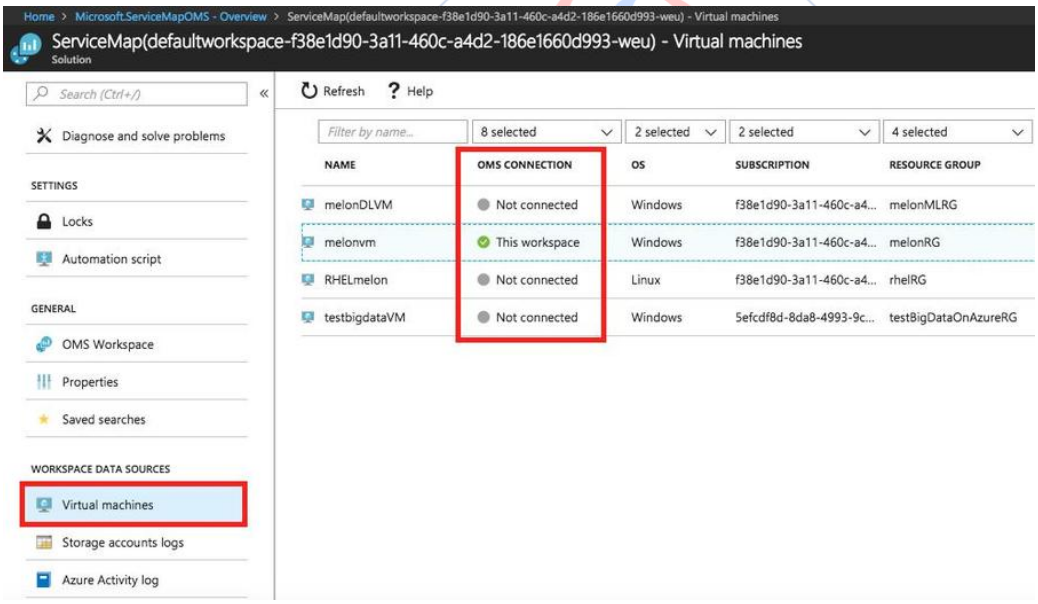
- **Service Endpoint Monitor** is a cloud-based service to monitor the reachability of applications and detects performance bottlenecks across cloud or on-premise data centers.

To know more about the networking monitoring solution in Azure, you can check out the following link:

https://docs.microsoft.com/en-us/azure/networking/network-monitoring-overview

Service Map

Service Map is similar to Application Map in Application Insights. It integrates events, performance data, and management solutions in Log Analytics. It provides insight for different processes within virtual machines and dependencies on other machines and external processes, as shown in the following screenshot:



Shared capabilities

Shared capabilities of monitoring strategy in Azure is realized by some shared functionalities, such as **alerts**, **dashboards**, and **metrics** that provide a basic understanding view of the current status of Azure resources. An excellent monitoring solution helps users understand better how do their solutions work and detect the exception in case of issue. It also performs proactive notification of critical issues so that users can take care of them before the real problem comes.

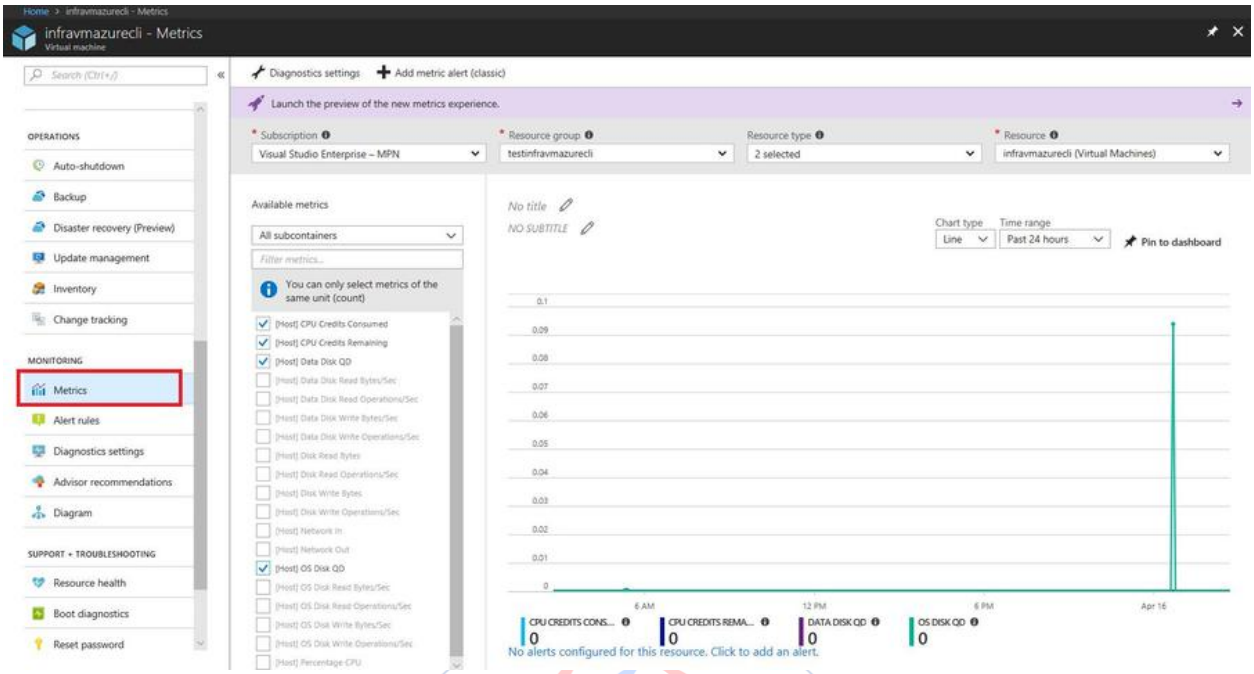
Implementing Azure VMs monitoring solutions

Like most Azure services, Azure VMs enable you to track their performance, availability, and usage. This data is available directly from the Azure Portal. You can also collect Azure VM metrics and diagnostics via Azure PowerShell and Azure CLI scripts. In addition, you can collect this data programmatically via the REST API and Azure SDKs.

Configuring ARM VM monitoring

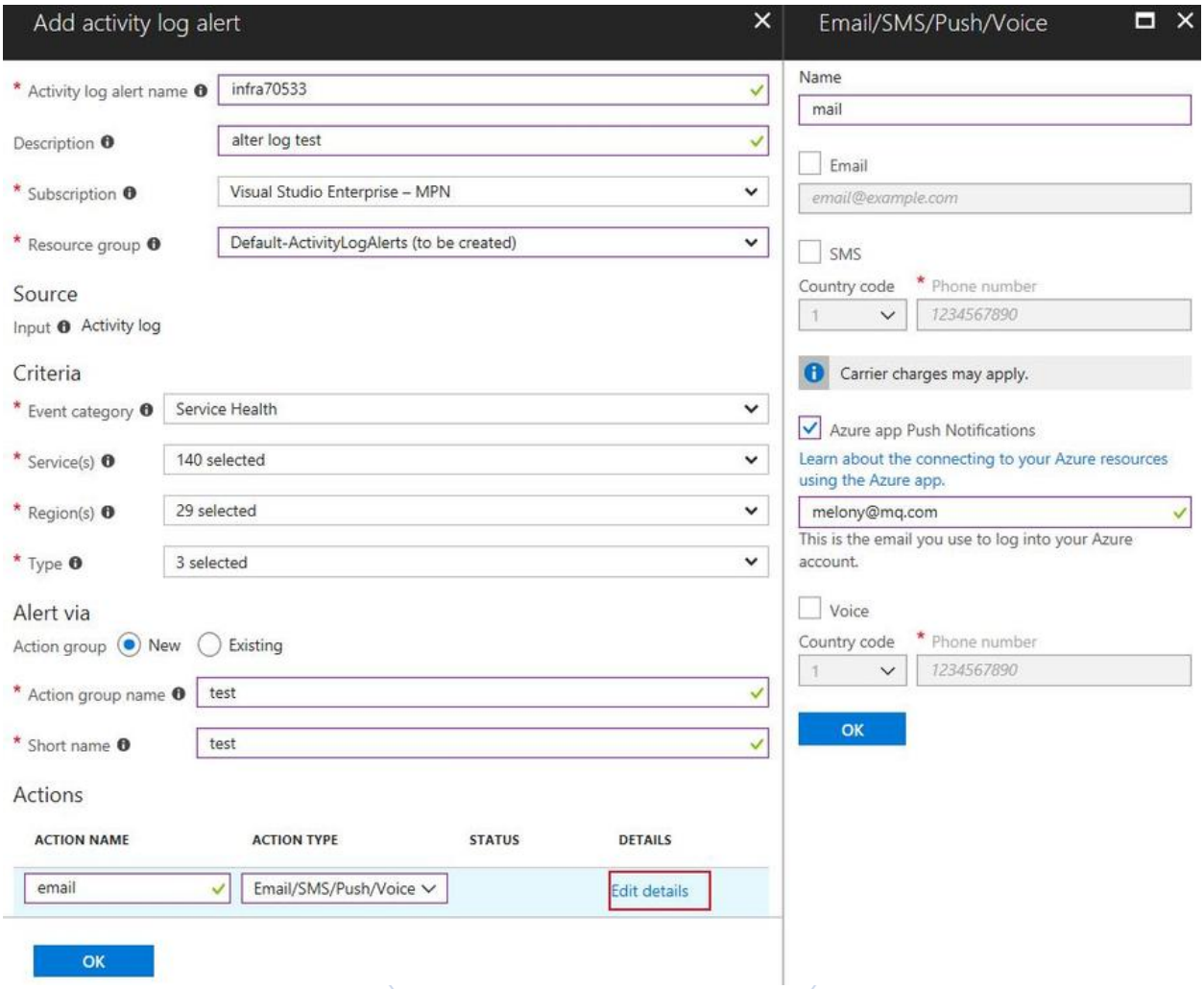
Azure VM monitoring provides metrics such as percentage**CPU**, **Network** in and out, disk read bytes, disk write bytes, disk read operations per seconds, and disk write operations per seconds, as shown in the following screenshot:

Azure also provides a way to query metrics in the **Metrics** blade of Azure VM, shown as follows:



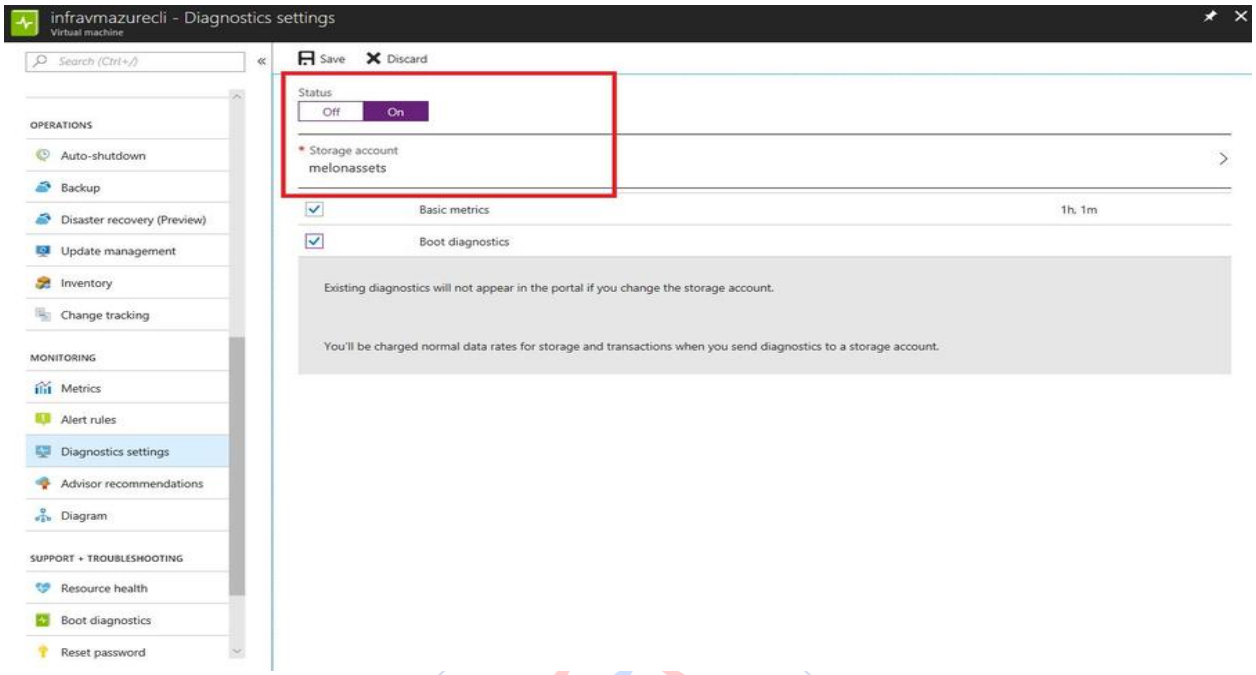
Configuring alerts

Azure provides alert rules that allow users to trigger notifications based on metrics-based criteria that they had to specify. In Azure, each rule includes a metric, condition, threshold, and time period that collectively determine when to raise an alert. You can configure your email address to get the alert notification. It also supports the Webhook, which is a route alert to an arbitrary HTTP or HTTPS endpoint. With alerts, it is possible to configure a response using an Azure Automation runbook, as shown in the following screenshot:

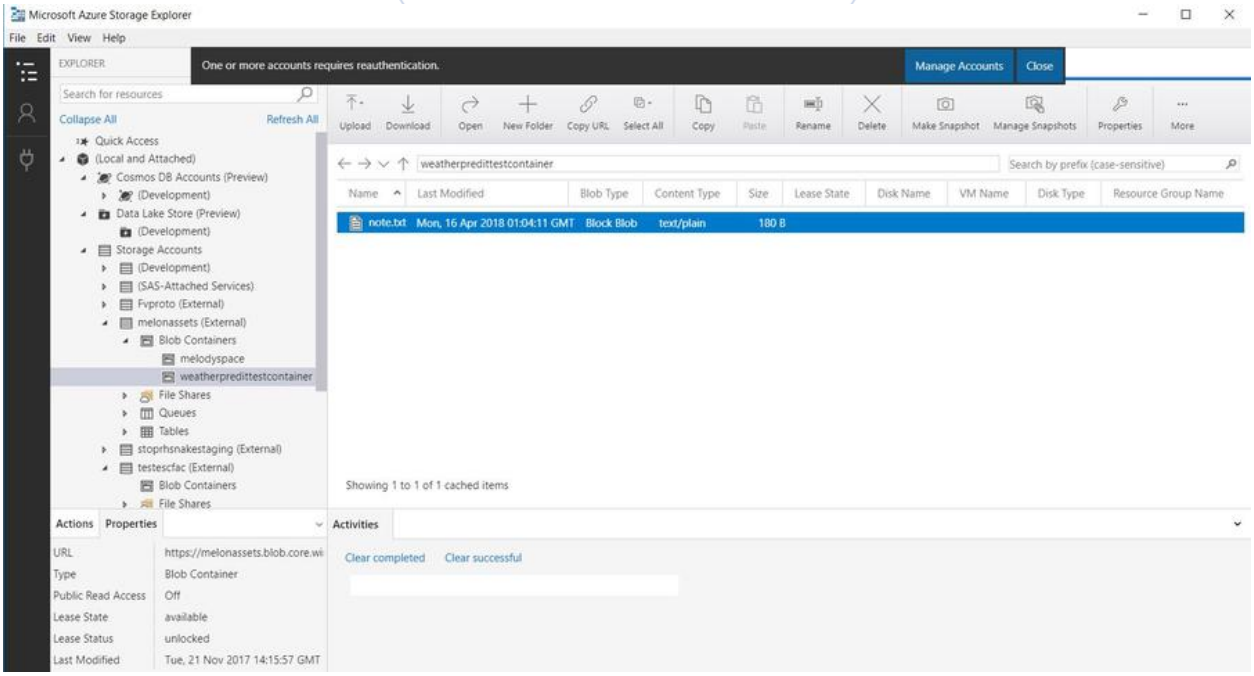


Configuring diagnostic and monitoring storage location

Azure provides insight into the performance and state of an Azure VM's OS by enabling diagnostics. For a Windows-based Azure VM, after enabling it, Azure will be able to collect data and monitor them via basic metrics including CPU usage, memory usage, network in and out , and so on, to enable **Boot diagnostics**, Azure will create the logs such as event logs, IIS logs and failed request logs, crash dumps. To enable diagnostics, you must configure a standard storage account so that the collected data can be stored; you can do it by setting the **Status** as **On** value and choosing the storage account that you want to use, as shown in the next screenshot:



To view and analyze diagnostics and logs, you can use a tool, such as **Azure Storage Explorer**, that provides access to tables and blobs in the Azure Storage account that is hosting collected data. It is also possible to export the data into an Excel file or any other business intelligence application (such as Power BI) for further analysis, as shown in the next screenshot:

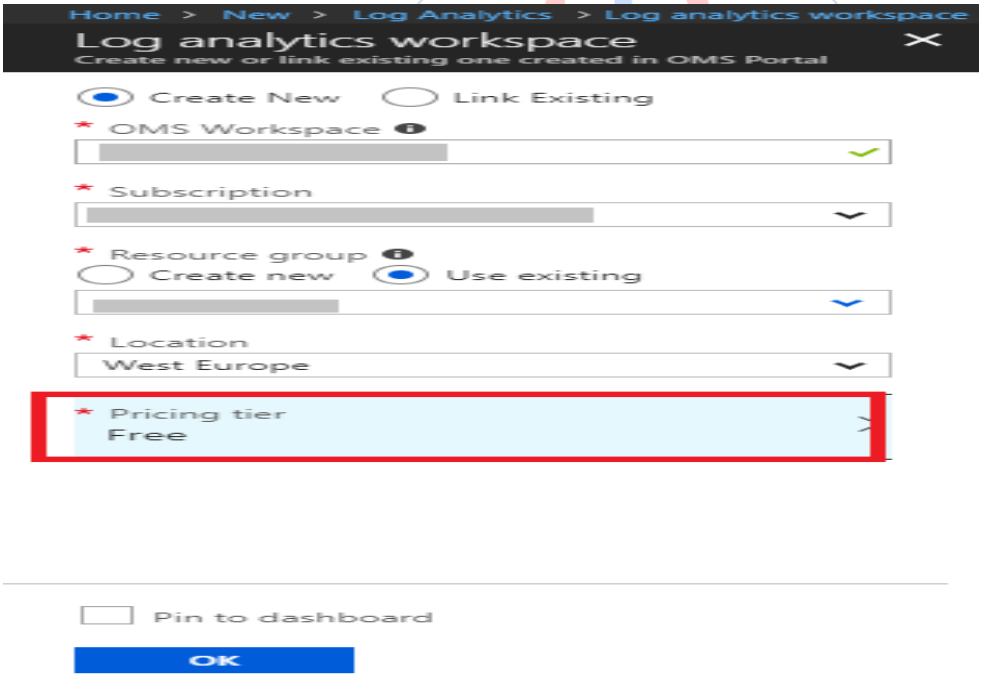


Implementing Log Analytics (OMS) solutions

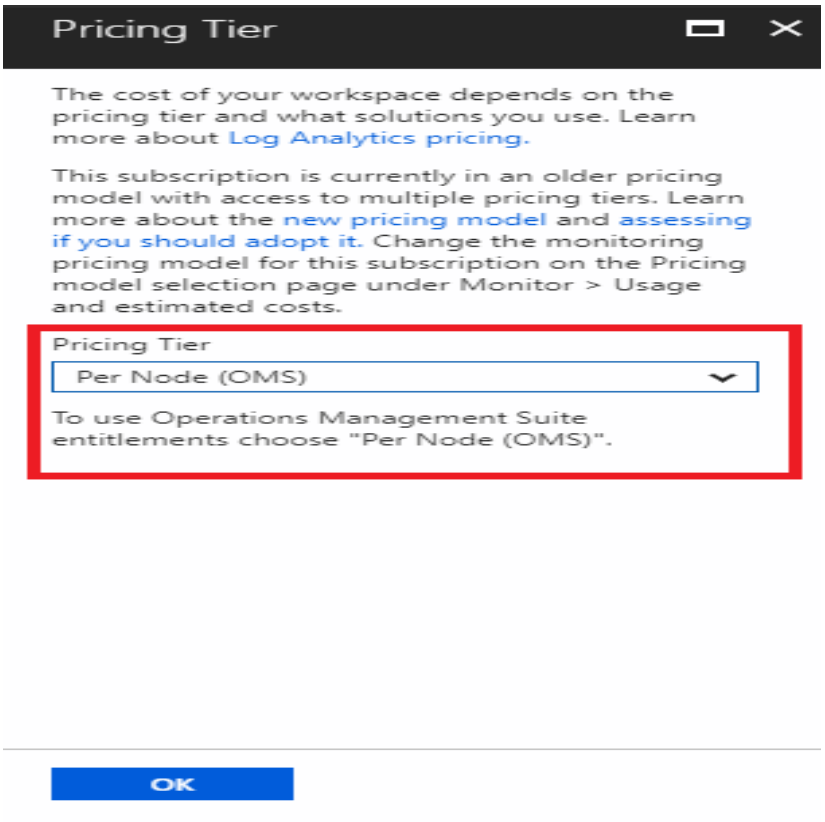
To enable Log Analytics (OMS), you can go to Azure Portal by searching Log Analytics or enable it via at the resource group level.

Creating an OMS workspace

You may start from creating an OMS workspace where you should fill in basic information such as workspace name, resource group, and subscription name and choose the right pricing tier, as shown in the following screenshot:



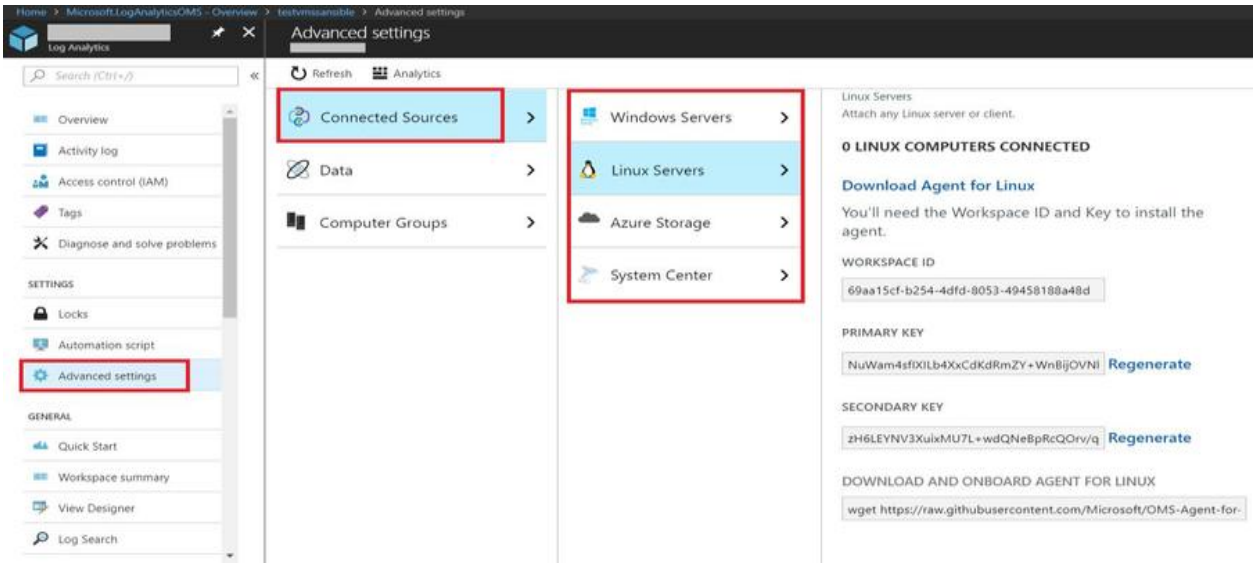
To choose the pricing tier of OMS solution, note the following screenshot:



As shown you may have some options:

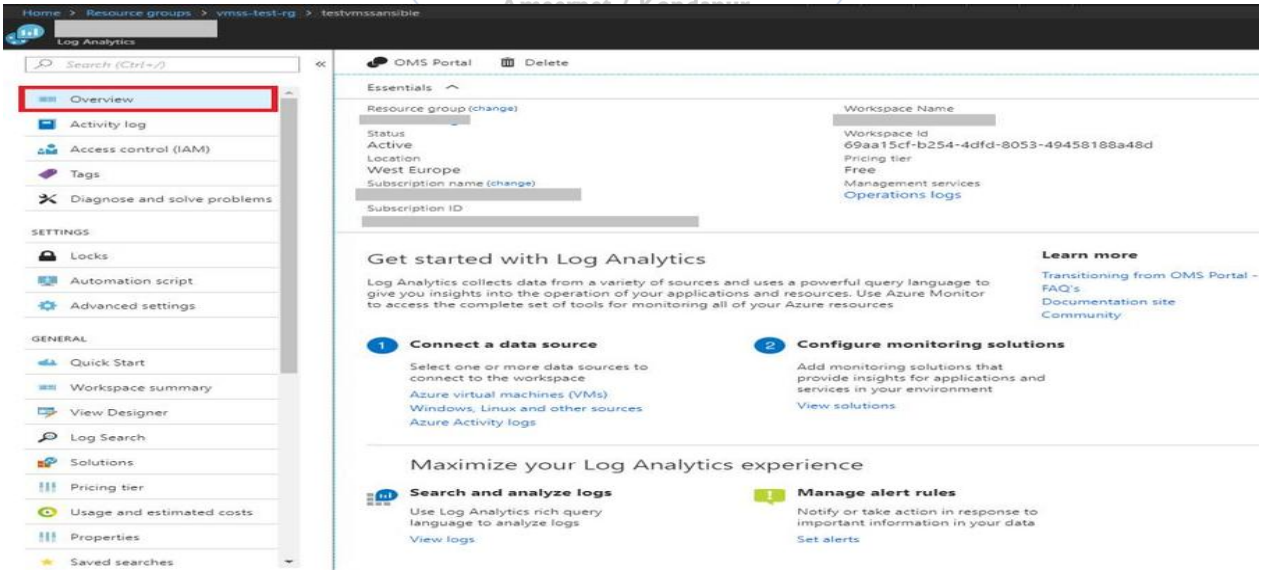
- **Free**: The old free tier has a 500 MB limit on the amount of data collected daily and doesn't allow for data retention periods longer than 7 days. The new pricing model does not have any limits on the amount of data collected daily and allows you to retain your log data for up to 2 years.
- **Per node** (**OMS**): A node is any physical server or virtual machine that is managed by the Insight and Analytics service such a Log Analytics, Service Map, or Network Performance Monitor. For details, refer to https://azure.microsoft.com/en-us/pricing/details/insight-analytics/
- **Per gigabyte** (**GB**): Log Analytics is billed per gigabyte (GB) of data ingested into the service. As of writing this book, the first 5 GB of data ingested to the Azure Log Analytics service every month is free but every GB of ingested data (even after the first 5 GB will be retained at no charge only for the first 31 days)

If you want to know if there are any services under monitor by current workspace of Log Analytics, check the connected Azure resource by clicking **Connected Sources** as shown below:

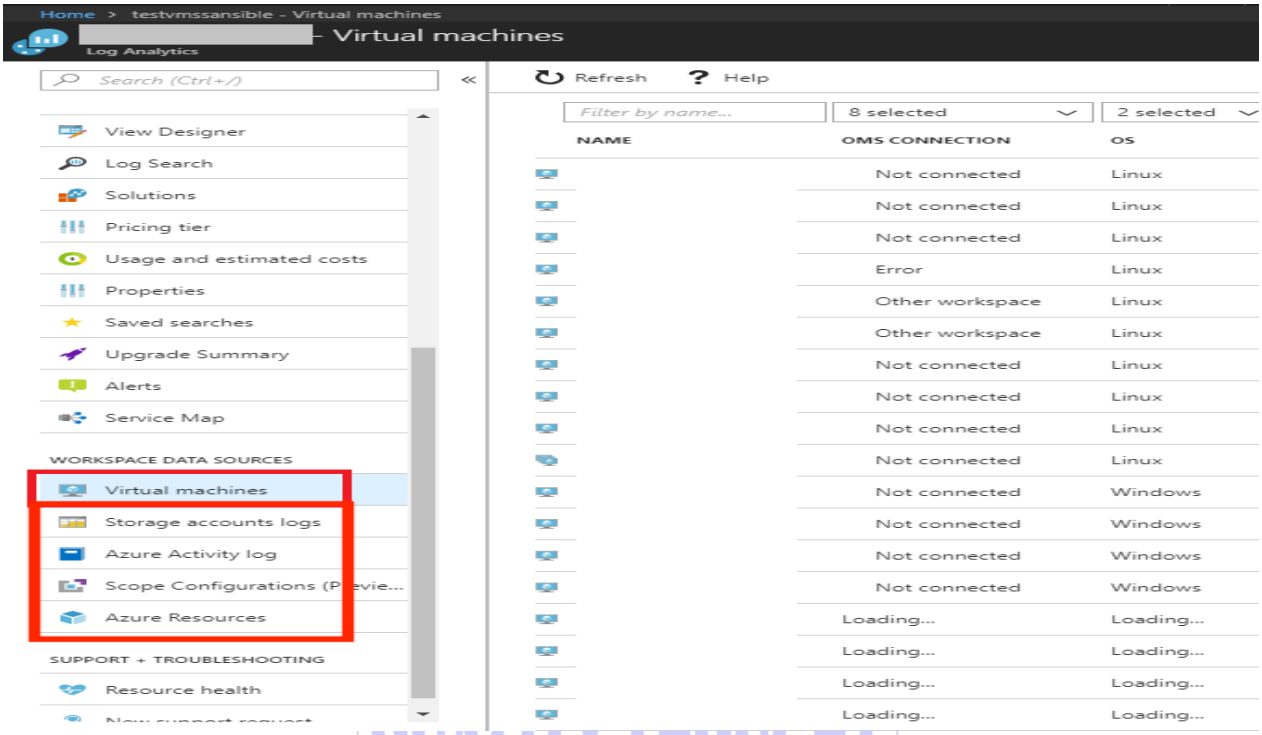


Collecting and searching across data sources from multiple systems

Log Analytics collects data from configured connected sources and stores it in Log Analytics workspace. To configure the Data Sources, you can go to the **Overview** blade and click on **Connect a data source** (as shown in the next screenshot); Log Analytics also allows you to maximize your Log Analytics experience by configuring search and analyze logs and manage alert rules so that you have a notification and will be able to take action in case issues arise:



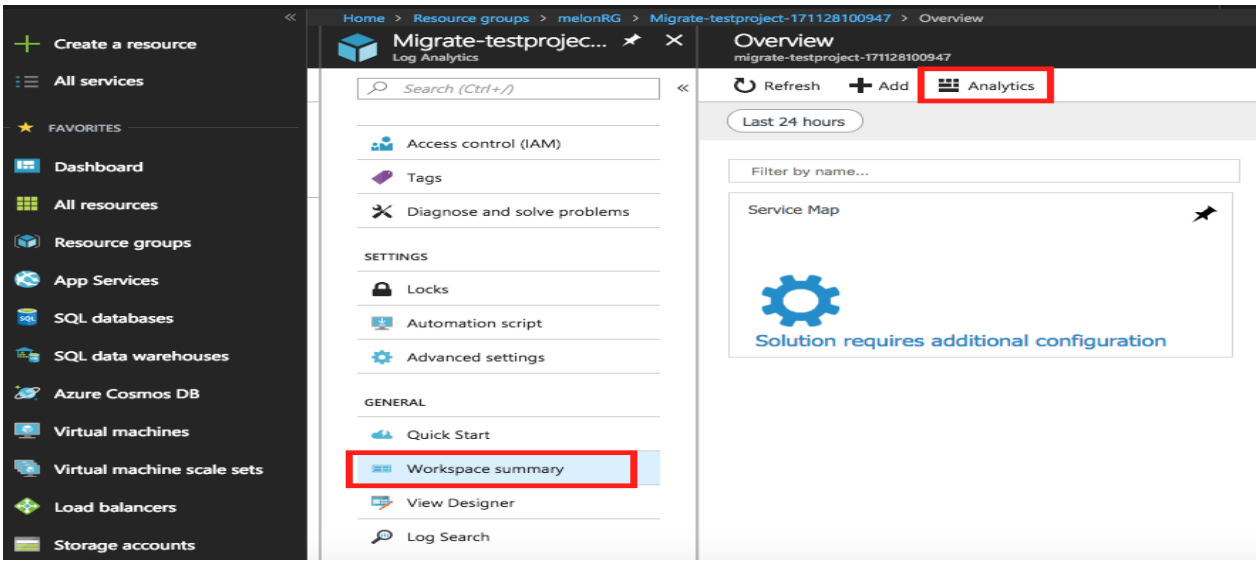
You can also go to the **WORKSPACE DATA SOURCES** section and configure the data source (as shown here):



Transforming Azure activity data and managed resource data

The Activity Log helps users analyze and search the Azure activity log across Azure subscriptions. The Activity Log focuses on providing insights into the operations level, with information regarding what, who, and when for any write operations, such as PUT, POST, and DELETE made for the resources across a subscription. Log Analytics offers the **Log search** functions to let users find the information they need in an effective way. Another way to find the needed monitoring information is to use queries. You can go to a Log Analytics portal to write these queries, and you can find this information in the output.

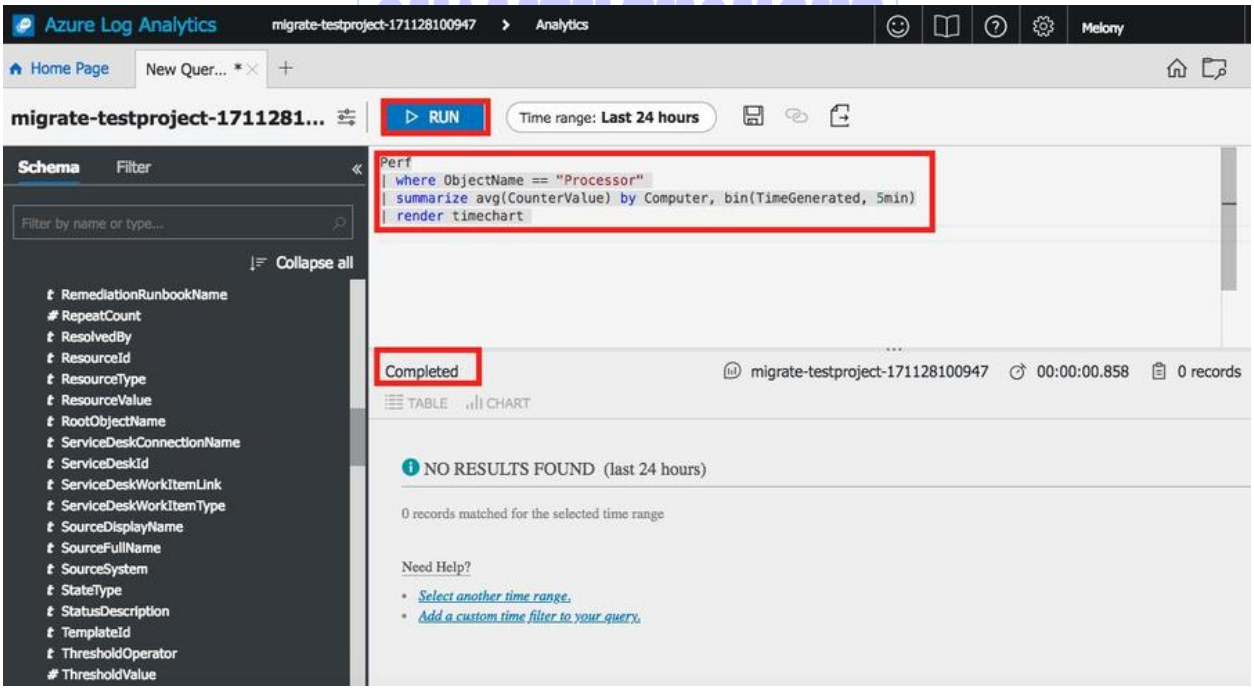
You can find your Log Analytics Portal (as shown in the following screenshot) by clicking on **Workspace summary** and then **Analytics**:



Here is a sample URL for the Log Analytics Portal:

https://portal.loganalytics.io/subscriptions/…

The portal contains a query section and output section, as follows:



Here is an example of a query that means to find all the updated operation times with more than 1 hour in your current workspace and display them with classification:

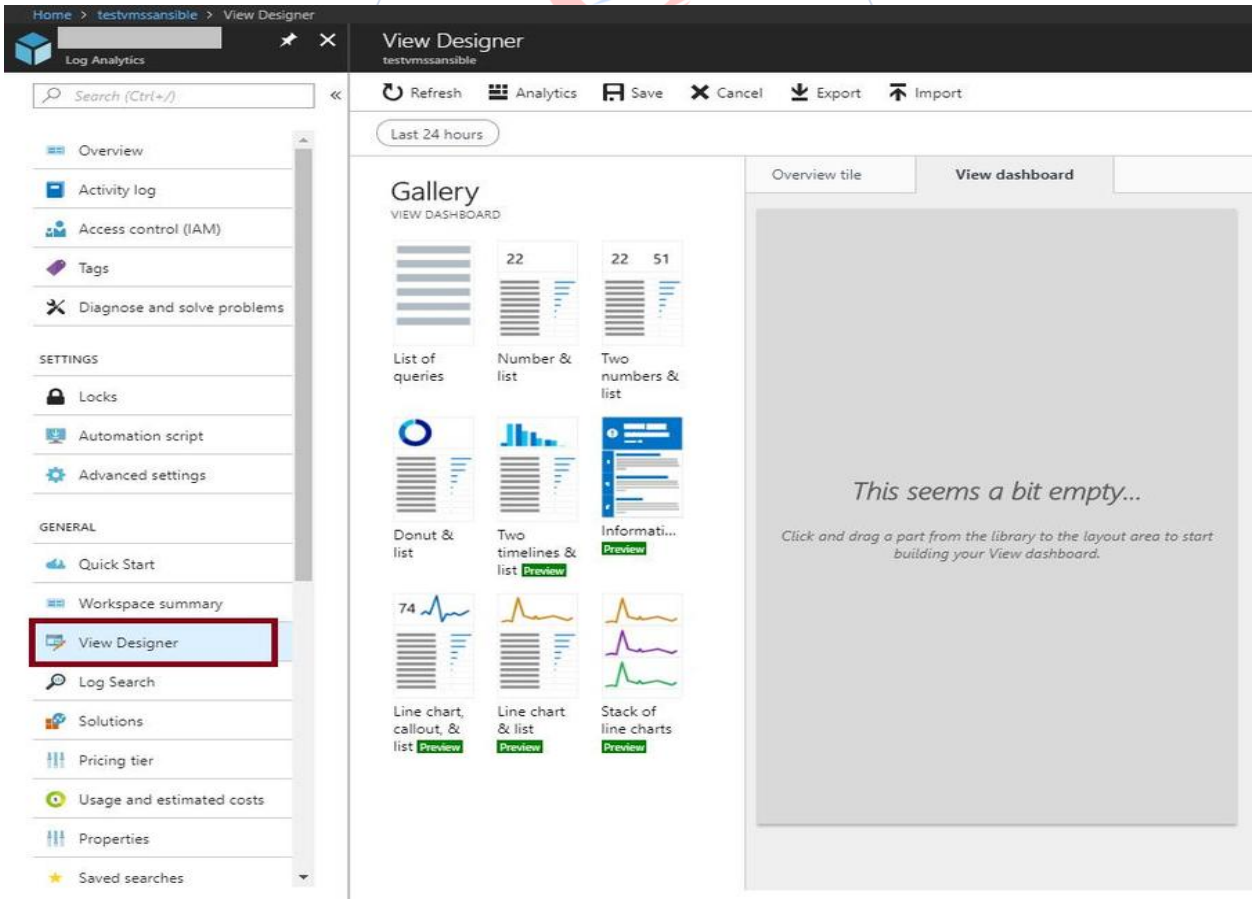
```
union Update, workspace("testinfra-workspace").Update
| where TimeGenerated >= ago(1h)
| summarize dcount(Computer) by Classification
```

To know more about log search and how to write queries, please check the following link:

https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-log-search

Building custom visualizations with view designer

Log Analytics makes it possible to collect events from text files and display them as your request. To do this, you can use **View Designer** in Log Analytics; go to your Log Analytics and click on the **View Designer** blade as shown in the following screenshot:



Build custom visualizations by using view designer

You can also check the following links to know more about:

- How to define a custom log file:https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-data-sources-custom-logs
- How to add your own searchable fields to extend existing records in Log Analytics:https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-custom-fields

Sending data to Log Analytics with the HTTP Data Collector API

You may also need to send or collect data to Log Analytics from a REST API client. To know more about how to use HTTP Data Collector API, checkout the following link:

https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-data-collector-api

IT Service Management Connector

The **IT Service Management Connector** (**ITSMC**) is a service in Azure to help users to connect Azure and **IT Service Management** (**ITSM**).

The ITSM connector provides connection between Azure and ITSM tools to help users resolve troubleshooting issues that are detected by Azure services, such as Azure Monitor and Log Analytics that reside in an ITSM service. These tools allow you to create work items in ITSM tools, based on your Azure alerts, such as metric alerts, Activity Log alerts, and Log Analytics alerts.

ITSMC supports connections with the ITSM tools, such as ServiceNow, System Center Service Manager, Provance, and Cherwell.

You can create an ITSMC via Azure Portal, and you only need to search for the term IT Service Management Connector. Fill in the related information and you can create it right

away. Thereafter you can manage it with Log Analytics or an other mentioned Azure service, shown as follows: