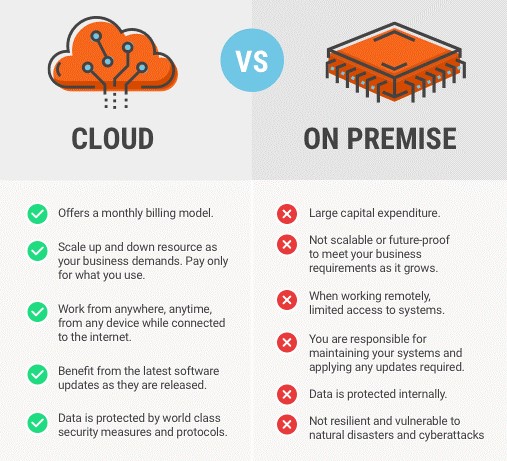
**What is Cloud Computing?**

[Cloud computing](https://www.simplilearn.com/tutorials/cloud-computing-tutorial/what-is-cloud-computing) is a technology that provides access to various computing resources over the internet. All you need to do is use your computer or mobile device to connect to your cloud service provider through the internet. Once connected, you get access to computing resources, which may include server less computing, virtual machines, storage, and various other things.

**Advantages of cloud computing:**

****

## What is Microsoft Azure?

[Azure is a cloud computing platform](https://www.simplilearn.com/putting-the-cloud-to-work-for-your-business-with-microsoft-azure-article) and an online portal that allows you to access and manage cloud services and resources provided by Microsoft. These services and resources include storing your data and transforming it, depending on your requirements. To get access to these resources and services, all you need to have is an active internet connection and the ability to connect to the Azure portal.

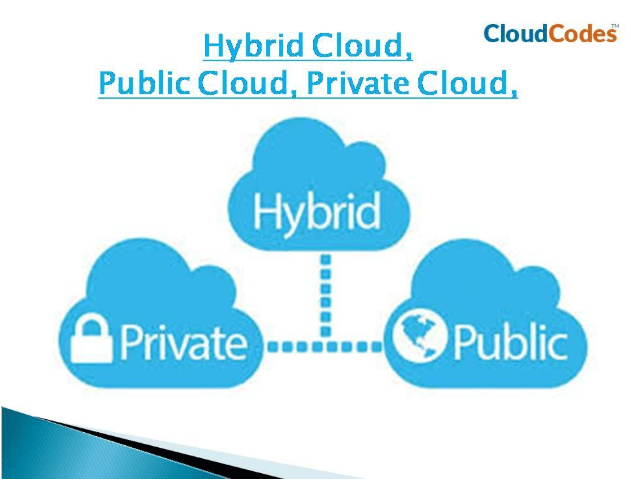
Things that you should know about Azure:

* It was launched on February 1, 2010, significantly later than its main competitor, AWS.
* It’s free to start and follows a pay-per-use model, which means you pay only for the services you opt for.
* Interestingly, 80 percent of the Fortune 500 companies use Azure services for their cloud computing needs.
* Azure supports multiple programming languages, including Java, Node Js, and C#.
* Another benefit of Azure is the number of data centers it has around the world. There are 42 Azure data centers spread around the globe, which is the highest number of data centers for any cloud platform. Also, Azure is planning to get 12 more data centers, which will increase the number of data centers to 54, shortly.

**Azure deployment methods:**

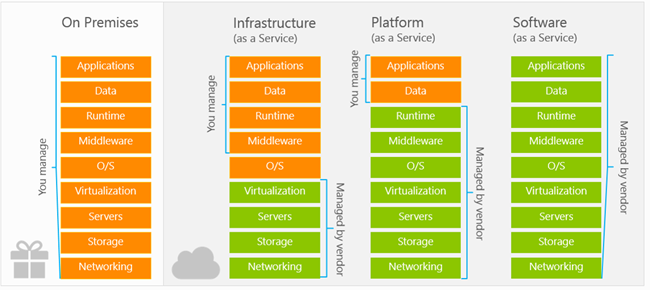
The cloud isn’t one thing—cloud computing can be categorized into three general types:

* **Public cloud** is cloud computing that’s delivered via the internet and shared across organizations.
* **Private cloud** is cloud computing that is dedicated solely to your organization.
* **Hybrid cloud** is any environment that uses both public and private clouds.

**Source Link:** [**https://www.bmc.com/blogs/public-private-hybrid-cloud/**](https://www.bmc.com/blogs/public-private-hybrid-cloud/)

**Cloud computing services:**

When talking about the topic of cloud computing services, you must have heard of these concepts, *Infrastructure as a Service (IaaS)*, *Platform as a Service (PaaS)*, and *Software as a Service (SaaS)*. They are the three major service categories provided by cloud providers. It is important to understand what they are and what the difference between them is because they will appear on various occasions where cloud computing services are discussed.

****

**Infrastructure as a Service (IaaS)**

As you can see in the image above, IaaS is the most flexible category of cloud services. Instead of buying hardware, with IaaS, you rent IT infrastructure servers and virtual machines (VMs), storage, networks, and operating systems from Microsoft on a pay-as-you-go basis, and you are responsible for managing the operating systems, data, and applications.

Therefore, Azure virtual machines are Infrastructure as a Service (IaaS) and Azure page blobs are the backbone of the virtual disks platform for Azure IaaS.

**Platform as a Service (PaaS)**

As you can see in the image above, a PaaS solution requires less user management and does not provide access to the operating system. That means that the PaaS is a complete development and deployment environment in the cloud and provides a framework that developers can build upon to develop or customize cloud-based applications.

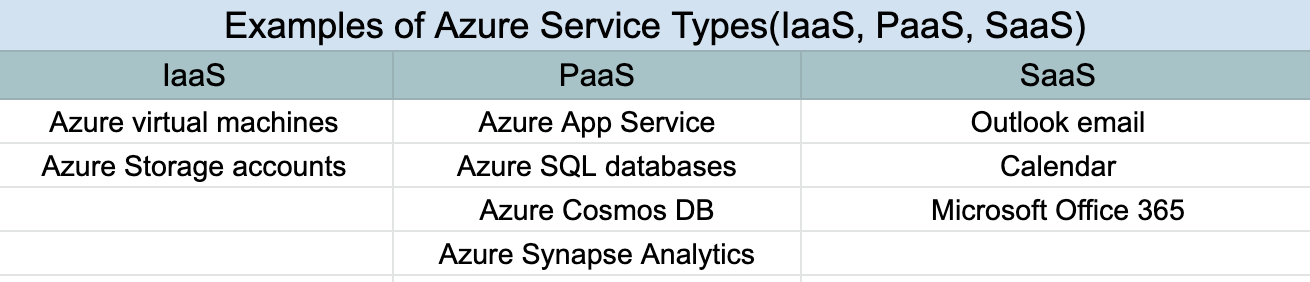
For example, the Azure Web Apps service provides an environment for you to host your web applications but you don't have to access the virtual machine and the operating system. And Azure SQL Database is a fully managed platform as a service (PaaS) database engine that handles most of the database management functions such as upgrading, patching, backups, and monitoring without user involvement.

**Software as a Service (SaaS)**

A SaaS solution requires the least management. Microsoft is responsible for managing everything, and you just use the software. SaaS allows you to connect to and use cloud-based apps over the Internet. When you are implementing a SaaS solution, you are responsible for configuring the SaaS solution. Common examples are Outlook email, calendar, and office tools (such as Microsoft Office 365).

**Summary**

And as a summary, here is a table that shows several examples of Azure services belonging to these three categories.



## The List of Microsoft Azure Services:

## Solved: Azure Cloud Service types ? - THWACK

### 1) Azure Application Services

Azure application services help you develop, deploy, and maintain various applications on the Azure platform. These include [**Azure AI**](https://azure.microsoft.com/en-in/overview/ai-platform/)**,**[**Azure Analytics**](https://azure.microsoft.com/en-in/product-categories/analytics/)**,**[**Azure IoT**](https://azure.microsoft.com/en-in/product-categories/iot/)**,**[**Azure Active Directory (AD)**](https://azure.microsoft.com/en-in/services/active-directory/)**,**[**Azure Media Services**](https://azure.microsoft.com/en-in/services/media-services/)**, and**[**Azure Scheduler**](https://azure.microsoft.com/en-in/services/scheduler/).

Let’s look at the three most notable application services: Azure AI, Azure Analytics, and Azure IoT.

#### 1.1 Azure AI

Microsoft Azure enables your AI teams to have quick access to AI services and helps you engage with customers, empower your employees, and optimize business processes across the organization. Your developers can create interactive applications on Azure’s AI Platform which offers AI capabilities like NLP, speech recognition, voice synthesis, face detection, video indexing, and text analytics.

Further, Microsoft Azure Bot Service releases intelligent agents on different platforms (website, facebook messenger, skype, email, SMS, etc.) that can be used to create [intelligent chat applications](https://victorops.com/blog/automated-chatops-in-incident-response).

#### 1.2 Azure Analytics

You can process and analyze large volumes of data to make well-informed critical business decisions with Azure’s analytics suite. It allows your analytics team to run complicated queries across huge sets of data and get actionable insights.

Backed by [Apache](https://www.apache.org/), the analytics solution offers an interactive real-time workspace to your data engineers, data scientists, and business analysts working on shared projects that enable them to quickly perform real-time analysis of large data sets on varied topics.

#### 1.3 Azure Internet of Things (IoT)

Azure IoT provides your organization with platform and services that can help you develop IoT solutions without reinventing the wheel. Azure’s IoT Central (SaaS) offers pre-configured solutions while “Azure IoT solution accelerators” offer more flexibility to develop a custom solution for your organization. Azure IoT Hub - is the core platform (PaaS) supporting Azure IoT Central and Azure IoT solution accelerators.

Further, Azure IoT offers numerous SDKs and integration support for connecting a range of IoT devices and gather insights from the incoming data with advanced analytics and visualization.

### 2) Azure Data Services

Azure data services store and manage data on cloud. Microsoft Azure comes with a range of data services: [Azure Storage](https://azure.microsoft.com/en-in/product-categories/storage/), [Azure SQL Database](https://azure.microsoft.com/en-in/services/sql-database/), [Azure Document DB](https://azure.microsoft.com/en-in/resources/videos/introduction-to-azure-documentdb/), [Azure StorSimple](https://azure.microsoft.com/en-in/services/storsimple/), and [Azure Redis Cache](https://docs.microsoft.com/en-us/azure/azure-cache-for-redis/).

Below is an overview of Azure Storage, one of the most prevalent data services provided by Microsoft Azure.

#### 2.1 Azure Storage

Microsoft Azure’s data storage service offers secure and scalable cloud storage for both, structured and unstructured data. Using Azure Storage, your data security experts can ensure safe integration of your on-premise data with the cloud data.

Azure Storage supports multiple storage options such as blob, queue, file, and NoSQL. You can store any size of data, from a small size blob to a large file, as per your business requirement without affecting the production environment.

### 3) Azure Development Services

Microsoft Azure offers multiple [development tools and services](https://azure.microsoft.com/en-in/tools/) to enhance the overall software development and deployment process. Here’s a quick look at one of the essential services by Microsoft i.e. Azure DevOps, enabling continuous delivery to customers.

#### 3.1 Azure DevOps

Azure DevOps automates your software delivery process and boosts your software development by providing an alternative to the self-managed CI/CD servers and open source DevOps tools. [Microsoft Azure DevOps](https://azure.microsoft.com/en-in/product-categories/devops/) technology can help you in hassle-free delivery with its fast and reliable tools. It is an extensible platform which can augment your existing tools and IDEs and offer hundreds of integrations in the Marketplace.

### 4) Azure Compute Services

Azure compute services are the hosting services responsible for hosting and running the application workloads. These include [Azure Virtual Machines (VMs)](https://azure.microsoft.com/en-in/services/virtual-machines/), [Azure Container Service](https://azure.microsoft.com/en-in/overview/containers/), [Azure App Services](https://azure.microsoft.com/en-in/services/app-service/), [Azure Batch](https://azure.microsoft.com/en-in/services/batch/), and [Azure ServiceFabric](https://azure.microsoft.com/en-in/services/service-fabric/).

Here’s a brief note on two of the most frequently used Azure Compute Services: *Azure Virtual Machines (VMs)* and *Azure Container Service*:

#### 4.1 Azure Virtual Machines (VMs)

A Microsoft Azure Virtual Machine (VM) is an on-demand, scalable computing resource. You don’t need to buy any physical hardware and bear its maintenance cost; you have the flexibility of virtualization. Your cloud administrators only need to select the operating system, configure the required resources, and create the web server – all this gets done within a few minutes.

#### 4.2 Azure Container Service

Azure helps you leverage the modern container-based development practices and micro services architecture. You can migrate your .NET applications to micro services using Windows Server containers with Azure Service Fabric. Further, you can use [Azure Kubernetes Service](https://azure.microsoft.com/en-us/services/kubernetes-service/) to scale and orchestrate Linux Containers.

You can choose between Docker Hub and Azure Container Registry to store your images and deploy to any preferred target. Moreover, it simplifies the configuration process and optimizes it for the cloud. The major advantage is that it consumes less space as compared to VMs and starts instantly; hence speeding up the processes.

### 5) Azure Network Services

Azure Network Services refer to those services that perform networking operations within Azure and between Azure and on-premise infrastructure. These include [**Azure Virtual Network**](https://azure.microsoft.com/en-in/services/virtual-network/)**,**[**Azure ExpressRoute**](https://azure.microsoft.com/en-in/services/expressroute/)**,**[**Azure-provided DNS**](https://azure.microsoft.com/en-in/services/dns/)**,**[**Azure Traffic Manager**](https://docs.microsoft.com/en-us/azure/traffic-manager/)**and**[**Azure Content Delivery Network (CDN)**](https://docs.microsoft.com/en-us/azure/cdn/).

Following is a brief outline on the Azure Content Delivery Network (CDN) service:

#### 5.1 Azure Content Delivery Network (CDN)

Azure CDN allows distribution of large files by providing high bandwidth content and reducing the file download time. It caches the file to an Azure data centre closest to the user’s geographical location. Hence, it lowers the latency to a great extent and decreases the load on application by relieving it from the whole process of delivering the content. The end-users derive the main benefit out of this service as it offers a better experience to them.

**Azure free trail account creation:**

Please go through the mentioned link and create the azure free trail account.

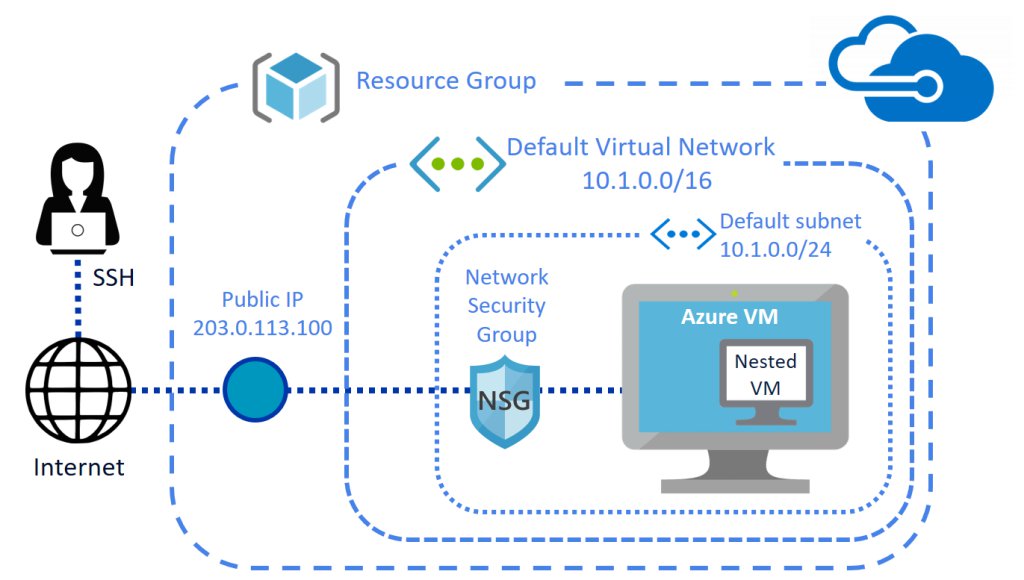
[**https://k21academy.com/microsoft-azure/create-free-microsoft-azure-trial-account/**](https://k21academy.com/microsoft-azure/create-free-microsoft-azure-trial-account/)

**Azure Virtual Network**

Azure Virtual Network (VNet) is the fundamental building block for your private network in Azure. VNet enables many types of Azure resources, such as Azure Virtual Machines (VM), to securely communicate with each other, the internet, and on-premises networks. VNet is similar to a traditional network that you'd operate in your own data center, but brings with it additional benefits of Azure's infrastructure such as scale, availability, and isolation.

## Why use an Azure Virtual network?

Azure virtual network enables Azure resources to securely communicate with each other, the internet, and on-premises networks. Key scenarios that you can accomplish with a virtual network include - communication of Azure resources with the internet, communication between Azure resources, communication with on-premises resources, filtering network traffic, routing network traffic, and integration with Azure services.



**Azure Vnet features:**

1. **Communication with the Internet directly**
2. **Communication with internet using NAT**
3. **Filter network traffic (NSG)**
4. **Monitor network traffic**
5. **VNET Peering**
6. **Secure**
7. **Simple**
8. **Reliable**

**Read below for detailed explanation:**

### Communicate with the internet

All resources in a VNet can communicate outbound to the internet, by default. You can communicate inbound to a resource by assigning a public IP address or a public Load Balancer. You can also use public IP or public Load Balancer to manage your outbound connections.

### Communicate between Azure resources

Azure resources communicate securely with each other in one of the following ways:

* **Through a virtual network**: You can deploy VMs, and several other types of Azure resources to a virtual network, such as Azure App Service Environments, the Azure Kubernetes Service (AKS), and Azure Virtual Machine Scale Sets.
* **Through a virtual network service endpoint**: Extend your virtual network private address space and the identity of your virtual network to Azure service resources, such as Azure Storage accounts and Azure SQL Database, over a direct connection. Service endpoints allow you to secure your critical Azure service resources to only a virtual network.
* **Through VNet Peering**: You can connect virtual networks to each other, enabling resources in either virtual network to communicate with each other, using virtual network peering. The virtual networks you connect can be in the same, or different, Azure regions.

**Communicate with on-premises resources**

You can connect your on-premises computers and networks to a virtual network using any combination of the following options:

* **Point-to-site virtual private network (VPN):** Established between a virtual network and a single computer in your network. Each computer that wants to establish connectivity with a virtual network must configure its connection. This connection type is great if you're just getting started with Azure, or for developers, because it requires little or no changes to your existing network. The communication between your computer and a virtual network is sent through an encrypted tunnel over the internet.
* **Site-to-site VPN:** Established between your on-premises VPN device and an Azure VPN Gateway that is deployed in a virtual network. This connection type enables any on-premises resource that you authorize to access a virtual network. The communication between your on-premises VPN device and an Azure VPN gateway is sent through an encrypted tunnel over the internet
* **Azure ExpressRoute:** Established between your network and Azure, through an ExpressRoute partner. This connection is private. Traffic does not go over the internet.

### Filter network traffic

You can filter network traffic between subnets using either or both of the following options:

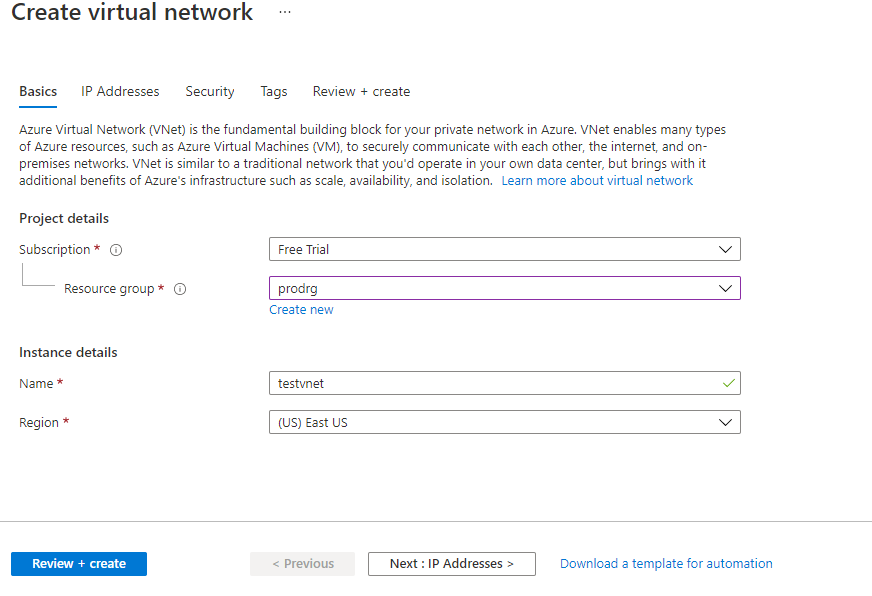
* **Network security groups:** Network security groups and application security groups can contain multiple inbound and outbound security rules that enable you to filter traffic to and from resources by source and destination IP address, port, and protocol.
* **Network virtual appliances:** A network virtual appliance is a VM that performs a network function, such as a firewall, WAN optimization, or other network function. To view a list of available network virtual appliances that you can deploy in a virtual network, see [Azure Marketplace](https://azuremarketplace.microsoft.com/marketplace/apps/category/networking?page=1&subcategories=appliances).

### Virtual network integration for Azure services

Integrating Azure services to an Azure virtual network enables private access to the service from virtual machines or compute resources in the virtual network. You can integrate Azure services in your virtual network with the following options:

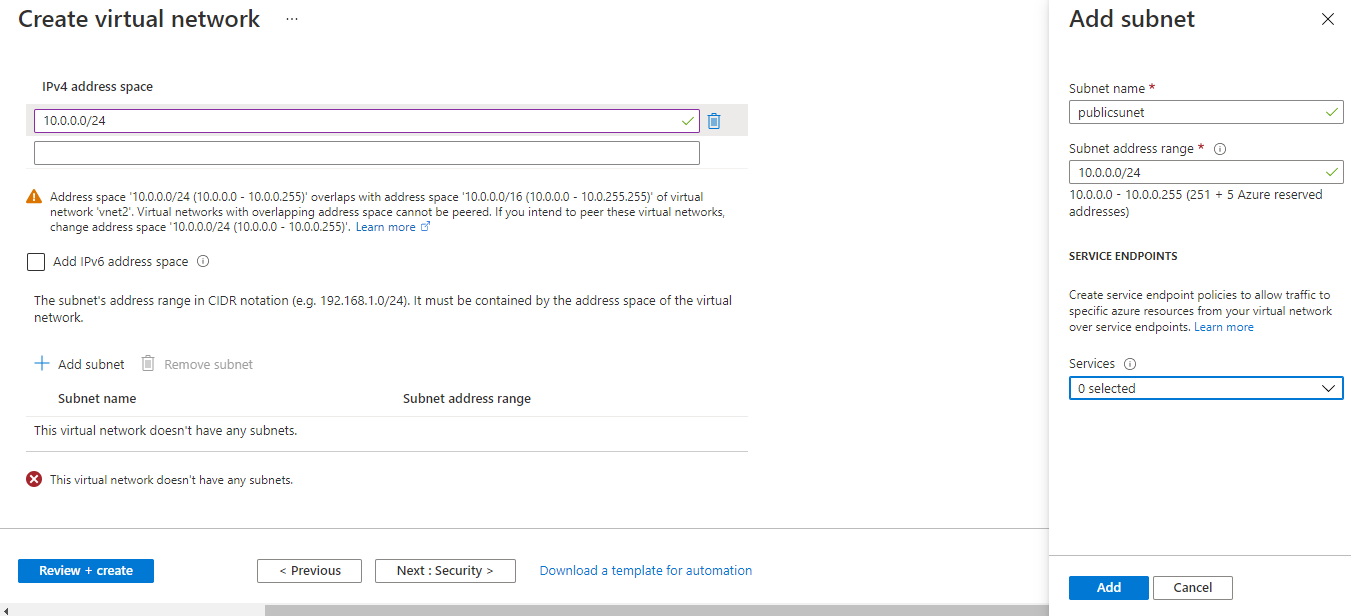
* Deploying [dedicated instances of the service](https://docs.microsoft.com/en-us/azure/virtual-network/virtual-network-for-azure-services) into a virtual network. The services can then be privately accessed within the virtual network and from on-premises networks.
* Using [Private Link](https://docs.microsoft.com/en-us/azure/private-link/private-link-overview) to access privately a specific instance of the service from your virtual network and from on-premises networks.
* You can also access the service using public endpoints by extending a virtual network to the service, through [service endpoints](https://docs.microsoft.com/en-us/azure/virtual-network/virtual-network-service-endpoints-overview). Service endpoints allow service resources to be secured to the virtual network.

**Create a virtual network**

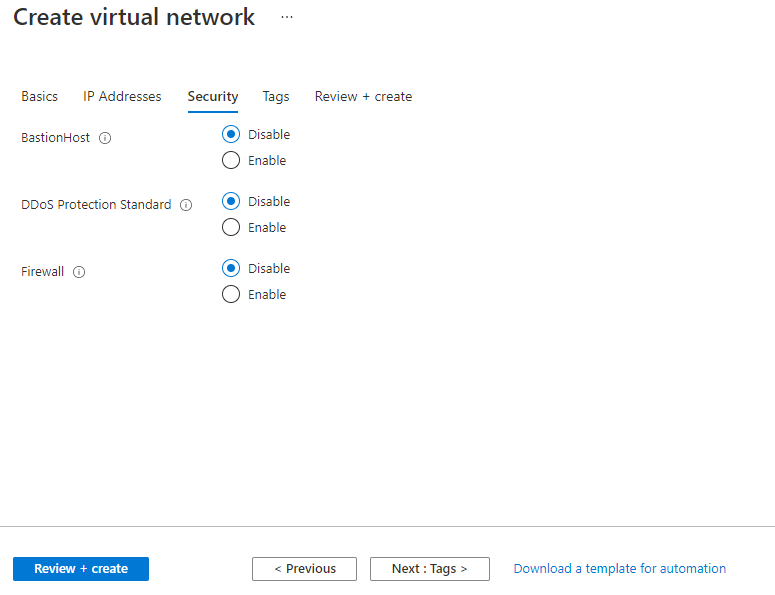


1. From the Azure portal menu, select **Create a resource**. From the Azure Marketplace, select **Networking** > **Virtual network**.
2. In **Create virtual network**, enter or select this information:

|  | |
| --- | --- |
| **Setting** | **Value** |
| Subscription | Select your subscription. |
| Resource group | Select **Create new**, enter *myResourceGroup*, then select **OK**. |
| Name | Enter *myVirtualNetwork*. |
| Location | Select **East US**. |



1. Select **Next: IP Addresses**, and for **IPv4 address space**, enter *10.1.0.0/16*.
2. Select **Add subnet**, then enter *myVirtualSubnet* for **Subnet name** and *10.1.0.0/24* for **Subnet address range**.



1. Select **Add**, then select **Review + create**. Leave the rest as default and select **Create**.
2. In **Create virtual network**, select **Create**.

**Create vnet by using azure cli:**

--> Install Azure cli in your system.

--> Run the command prompt in system as administrator.

--> command: az login

--> Copy the link and code displayed on the command prompt and regiter for Azure CLI.

--> https://aka.ms/devicelogin -- Code displayed.

--> cls - to clear the screen.

--> Azure commands always starts with (az).

**Create azure Resource group**

az group create --name RG2 --location eastus

**Command to create a VNET.**

az network vnet create --name vnet2 --resource-group RG2 --subnet-name public

**Command to add extra subnet to existing VNET.**

az network vnet subnet create --name private --address-prefix 10.0.1.0/24 --vnet-name vnet2 --resource-group RG2

**16. Azure - Confgure VNET using Azure PowerShell:**

--> Run the Windows Poweshell ISE as adminstrator.

--> First You need to install package for azure (azurerm(new) or azure(old)).

**Command to install azurerm package using powershell**

install-package -name azurerm --> command: install-module -name azurerm

Set-ExecutionPolicy -ExecutionPolicy RemoteSigned

login-AzureRmAccount --> provide the credentials to login.

**Create Azure Resource Group**

New-AzureRmResourceGroup -Name RG3 -Location

**Vnet Creation:**

New-AzureRmVirtualNetwork -name vnet3 -ResourceGroupName RG3 -Location eastus -AddressPrefix 10.0.0.0/16

**Add sunet:**

New-AzureRmVirtualNetworkSubnetConfig -name public -AddressPrefix 10.0.1.0/24

**command to store the information in variable.**

$vent2 = Get-AzureRmVirtualNetwork -name vnet3 -ResourceGroupName RG3

**Command to Add subnet to VNET.**

Add-AzureRmVirtualNetworkSubnetConfig -name public -VirtualNetwork $vent2

--> Provide the AddressPrefix : 10.0.1.0/24

----------------------------------------

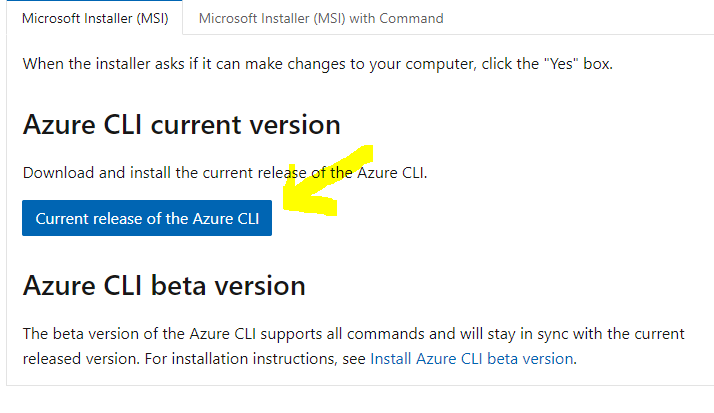
## Prerequisites

1. Use the Bash environment in [Azure Cloud Shell](https://docs.microsoft.com/en-us/azure/cloud-shell/quickstart).

[Launch Cloud Shell in a new window](https://shell.azure.com/)

1. If you want to use local, download the cli by clicking below link.

<https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?tabs=azure-cli>



1. If you prefer, [install](https://docs.microsoft.com/en-us/cli/azure/install-azure-cli) the Azure CLI to run CLI reference commands.

If you're using a local installation, sign in to the Azure CLI by using the [az login](https://docs.microsoft.com/en-us/cli/azure/reference-index" \l "az_login) command. To finish the authentication process, follow the steps displayed in your terminal. For additional sign-in options, see [Sign in with the Azure CLI](https://docs.microsoft.com/en-us/cli/azure/authenticate-azure-cli).

Run [az version](https://docs.microsoft.com/en-us/cli/azure/reference-index?" \l "az_version) to find the version and dependent libraries that are installed. To upgrade to the latest version, run [az upgrade](https://docs.microsoft.com/en-us/cli/azure/reference-index?" \l "az_upgrade).

## Create a new Windows virtual machine:

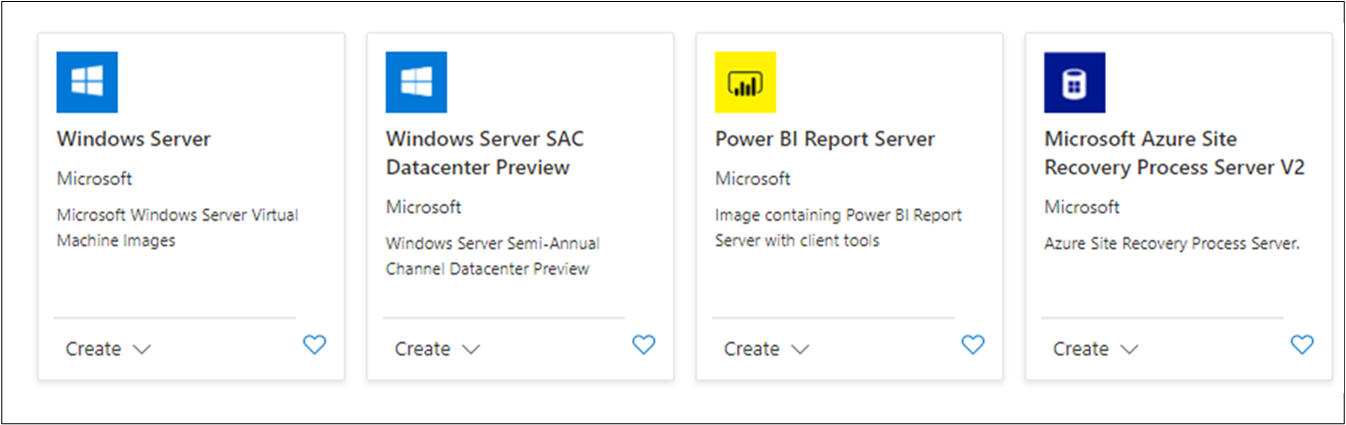
We can create Windows VMs with the Azure portal, Azure CLI, or Azure PowerShell. The easiest approach is the portal because it walks you through the required information and provides hints and helpful messages during the creation of the VM.

1. Sign into the [Azure portal](https://portal.azure.com/learn.docs.microsoft.com) using the same account you activated the sandbox with.
2. On the Azure portal menu or from the **Home** page, select **Create a resource**.
3. Select the **See all** link next to the Azure Marketplace heading. This option displays all available images on the Azure Marketplace.

Select **Microsoft** as the filter value from the Publisher filter options. In the search box, enter **Windows Server** and then action the search by pressing Enter.



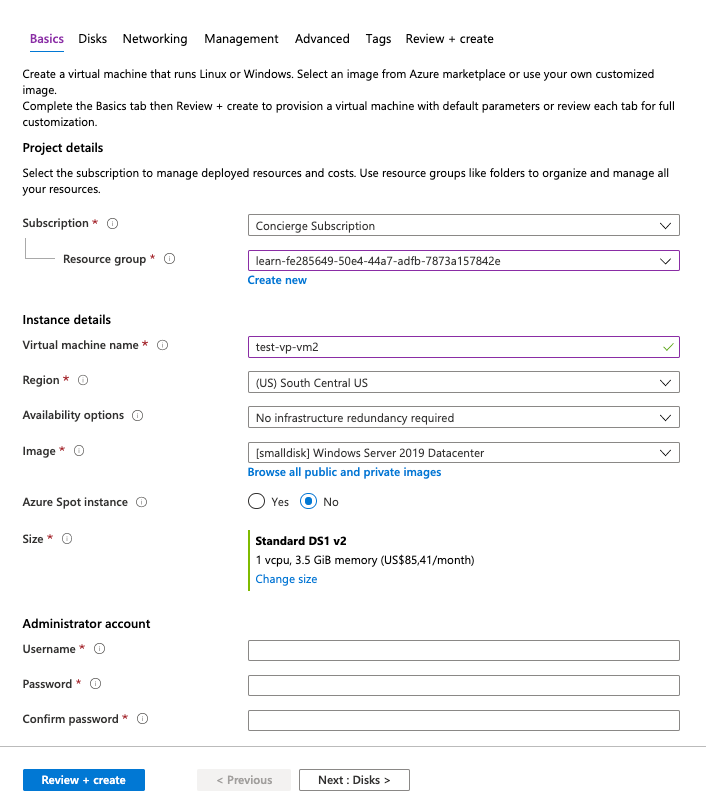
1. The result will show several Windows Server related images. Select the **Windows Server** image.



1. There are several Windows Server versions we can select from to create our VM. In the Windows Server image overview panel, from the **Select a plan** dropdown, select the **[smalldisk] Windows Server 2019 Datacenter** option.
2. Select **Create** to start configuring the VM.

## Configure the VM settings

We'll start with the **Basics** section.



### Configure basic VM settings

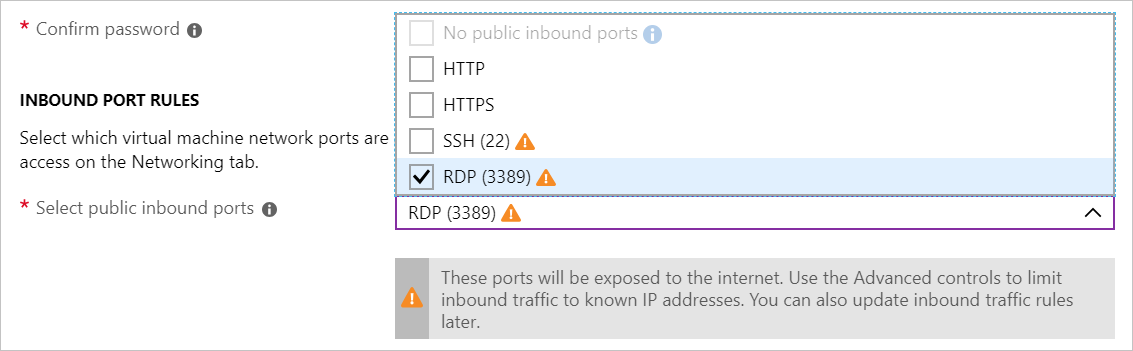
As you change settings and tab out of each free-text field, Azure will validate each value automatically and place a green check mark next to it when it's good. You can hover over error indicators to get more information on issues it discovers.

1. Select the **Subscription** that should be billed for VM hours.
2. For **Resource group**, select any resource group name.
3. In the **Instance Details** section, enter a name for your VM, such as **test-vp-vm2**
   * It's best practice to standardize your resource names so you can easily identify their purpose. Windows VM names are a bit limited - they must be between 1 and 15 characters, cannot contain non-ASCII or special characters, and must be unique in the current resource group.
4. Select a region close to you from the locations list.
5. Leave **Availability options** as "No Infrastructure redundancy required". This option is used to ensure the VM is highly available by grouping multiple VMs together a set to deal with planned or unplanned maintenance events or outages.
6. Ensure the image is set to "[smalldisk] Windows Server 2019 Datacenter - Gen1". You can open the dropdown to see all the options available.
7. The **Size** field is not directly editable and has a DS1 default size. Select the **Change size** link to explore other VM sizes. The resulting dialog allows you to filter based on # of CPUs, Name, and Disk Type. Select "Standard DS1 v2" (normally the default) when you are done. That will give the VM 1 CPU and 3.5 GB of memory.

**Tip**

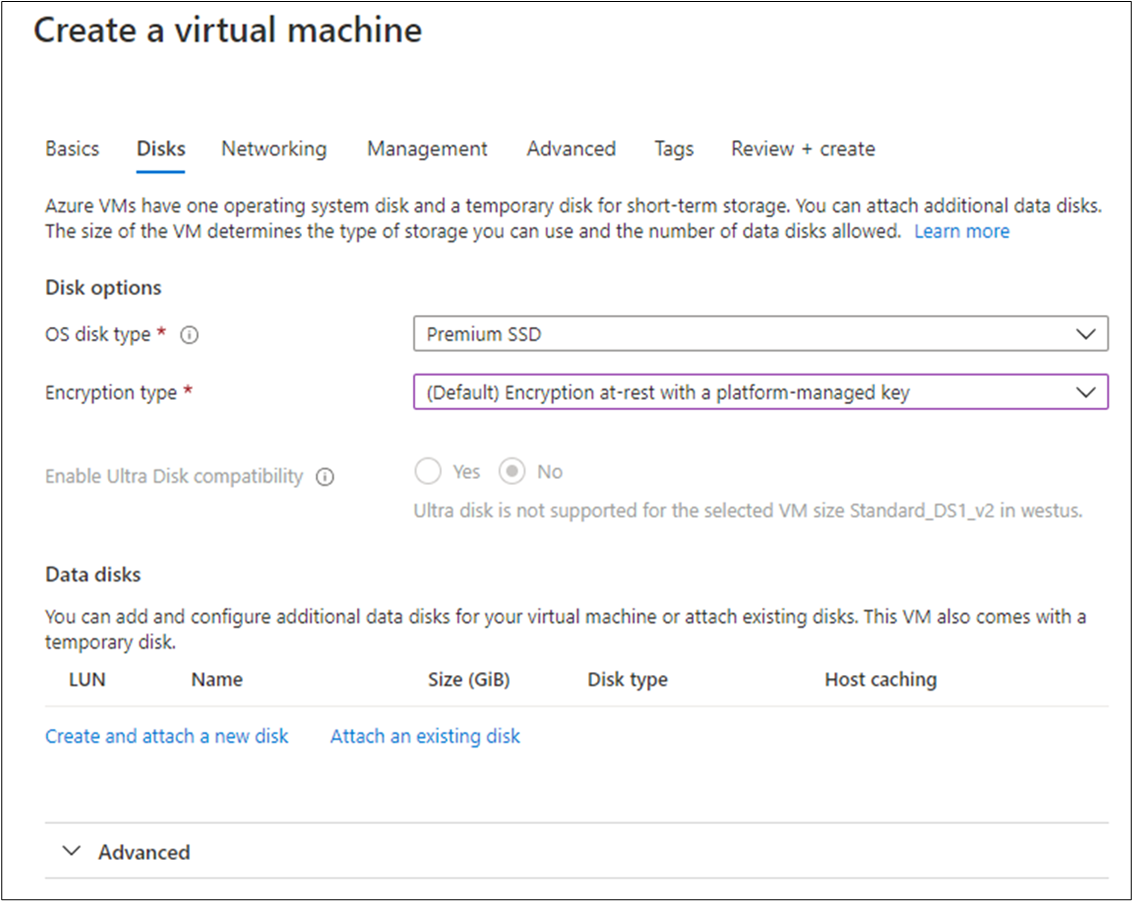
You can also just slide the view to the left to get back to the VM settings as it opened a new window off to the right and slid the window over to view it.

1. In the **Administrator Account** section, set the **Username** field to a username you will use to sign in to the VM.
2. In the **Password** field, enter a password that's at least 12 characters long. It must have three of the following: one lower case character, one uppercase character, one number, and one special character that is not '\' or '-'. Use something you will remember or write it down, you will need it later.
3. Confirm the **password**.
4. In the **Inbound port rules** section, for **Public inbound ports**, select Allow selected ports. Because this is a Windows VM, we want to be able to access the desktop using RDP. For **Select inbound ports**, select RDP (3389) from the dropdown. As the note in the UI indicates, we can also adjust the network ports after we create the VM.



## Configure Disks for the VM

1. Select **Next: Disks**to move to the Disks tab.

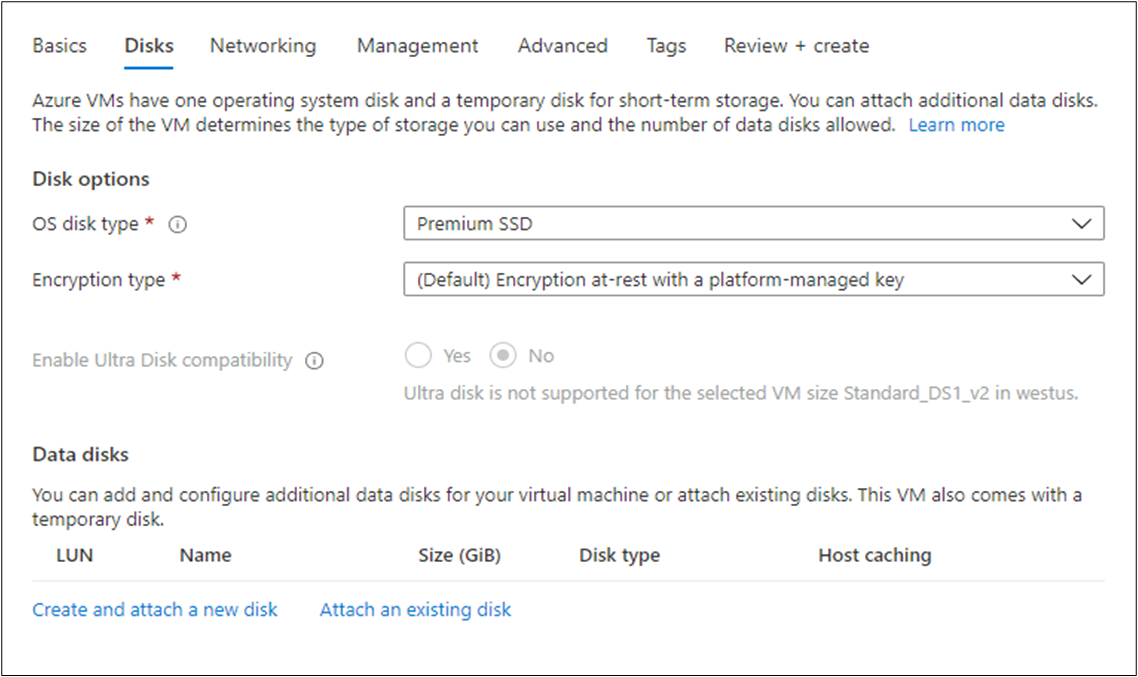


1. Select Premium SSD for the **OS disk type**.
2. For **encryption type**, select the (Default) Encryption at-rest with a platform-managed key from the dropdown.

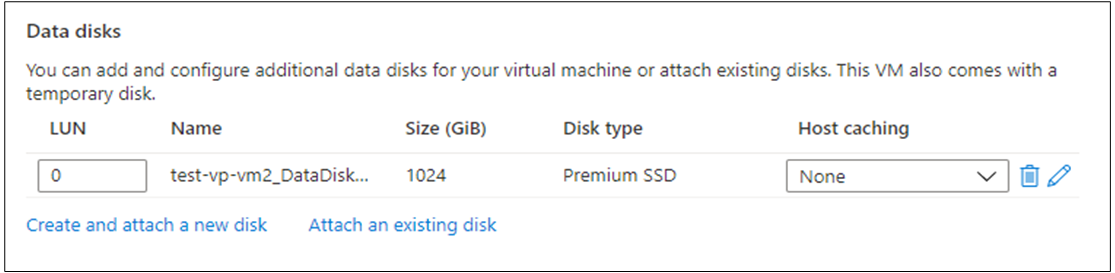
### Create a data disk

Recall we will get an OS disk (C:) and Temporary disk (D:). Let's add a data disk as well.

1. Select the **Create and attach a new disk** link in the **Data disks** section.



1. You can take all the defaults: Name; None (empty disk); 1024 GiB Premium SSD; (Default) Encryption at-rest with a platform-managed key; and No for **Enable shared disk**. Note that here is where we could use a snapshot, or Storage Blob to create a VHD.
2. Select **OK** to create the disk and go back to the **Data disks** section.
3. Under **Data disks**, there should now be a new disk in the first row.



## Configure the network

1. Select **Next: Networking** to move to the Networking section.
2. In a production system, where we already have other components, we'd want to utilize an existing virtual network. That way our VM can communicate with the other cloud services in our solution. If there isn't one defined in this location yet, we can create it here and configure the:
   * **Subnet**: First subnet to subdivide the address space - it must fit within the defined address space. Once the VNet is created, you can add additional subnets.
   * **Public IP**: Overall IPV4 space available to this network.
3. Let's change the default ranges to use the 172.xxx IP address space. Select **Create new** under **Virtual network**. The **Create virtual network** pane appears.

In the **Address space** section:

* + Change the **Address range** field to be 172.16.0.0/16 to give it the full range of addresses

In the **Subnets** section:

* + Leave the **Subnet name** set as default.
  + Change the **Address range** field to be 172.16.1.0/24 to give it 256 IP addresses of the space.

1. Select **OK**.

**Note**

By default, Azure will create a virtual network, network interface, and public IP for your VM. It's not trivial to change the networking options after the VM has been created so always double-check the network assignments on services you create in Azure.

## Finish configuring the VM and create the image

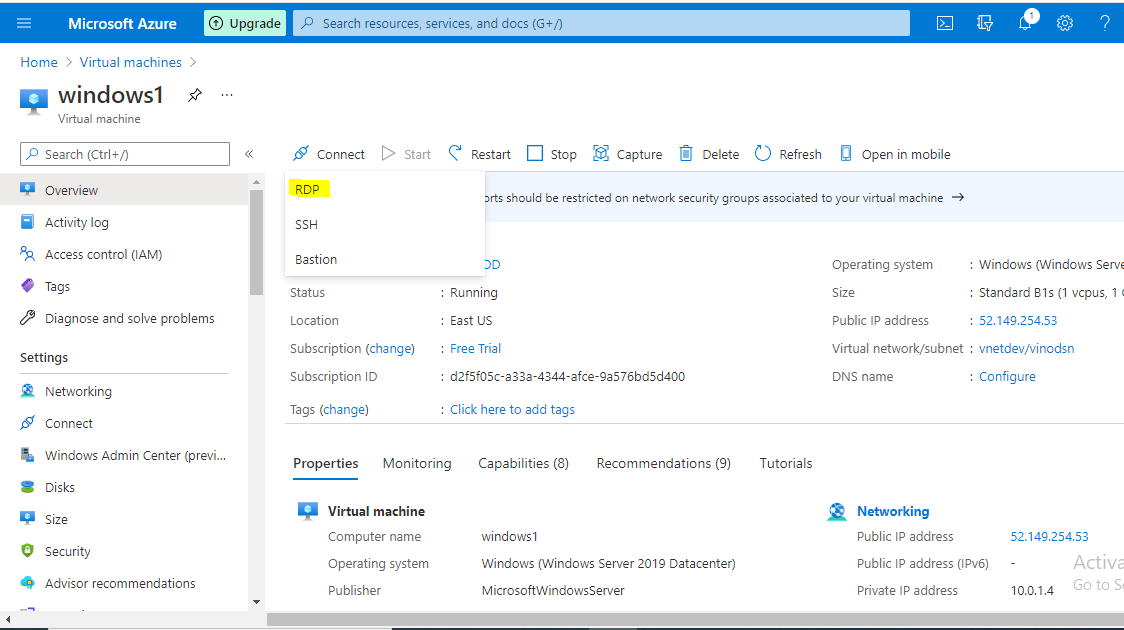
The rest of the options have reasonable defaults and there's no need to change any of them. You can explore the other tabs if you like. The individual options have an (i) icon next to them that will show a help bubble to explain the option. This is a great way to learn about the various options you can use to configure the VM.

1. Select **Review + create**. The system will validate your options and give you details about the VM being created.
2. Select **Create** to create and deploy the VM. The Azure dashboard will show the VM that's being deployed. This may take several minutes.

While that's deploying, let's look at what we can do with this VM.

**Connect to the windows machine with RDP:**

* Go to the Azure server overview
* click on connect
* select RDP
* download the RDP file
* give the username and password (which you have given while creating vm)



**Resize the VM Disk size:**

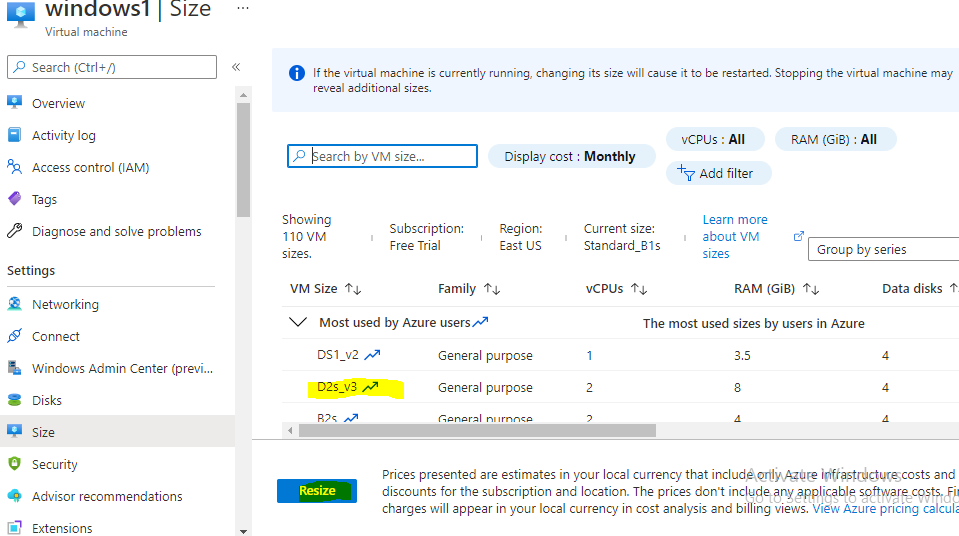
After you create a virtual machine (VM), you can scale the VM up or down by changing the VM size. In some cases, you must deallocate the VM first. This can happen if the new size is not available on the hardware cluster that is currently hosting the VM.

If your VM uses Premium Storage, make sure that you choose an **s** version of the size to get Premium Storage support. For example, choose Standard\_E4**s**\_v3 instead of Standard\_E4\_v3.

## Use the portal

1. Open the [Azure portal](https://portal.azure.com/).
2. Open the page for the virtual machine.
3. In the left menu, select **Size**.
4. Pick a new size from the list of available sizes and then select **Resize**.

If the virtual machine is currently running, changing its size will cause it to be restarted. Stopping the virtual machine may reveal additional sizes.



**Azure VM Status:**

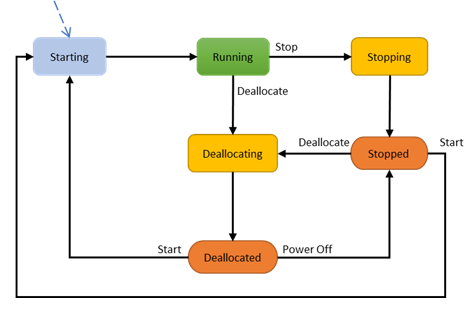
If your VM is not running, it will have one of two states – Stopped, or Stopped (deallocated). Essentially, if something is “allocated” – you’re still paying for it. So while deallocating a virtual machine sounds like a harsh action that may be permanently deleting data, it’s the way you can save money on your infrastructure costs and eliminate wasted Azure spend with no data loss.

## Azure’s Stopped State

When you are logged in to the operating system of an Azure VM, you can issue a command to shut down the server. This will kick you out of the OS and stop all processes, but will maintain the allocated hardware (including the IP addresses currently assigned). If you find the VM in the Azure console, you’ll see the state listed as “Stopped”. The biggest thing you need to know about this state is that you are still being charged by the hour for this instance.

## Azure’s Deallocated State

The other way to stop your virtual machine is through Azure itself, whether that’s through the console, Powershell, or the Azure CLI. When you stop a VM through Azure, rather than through the OS, it goes into a “Stopped (deallocated)” state.  This means that any non-static public IPs will be released, but you’ll also stop paying for the VM’s compute costs. This is a great way to save money on your Azure costs when you don’t need those VMs running, and is the state that ParkMyCloud puts your VMs in when they are parked.

****

**Azure Disk snapshot:**

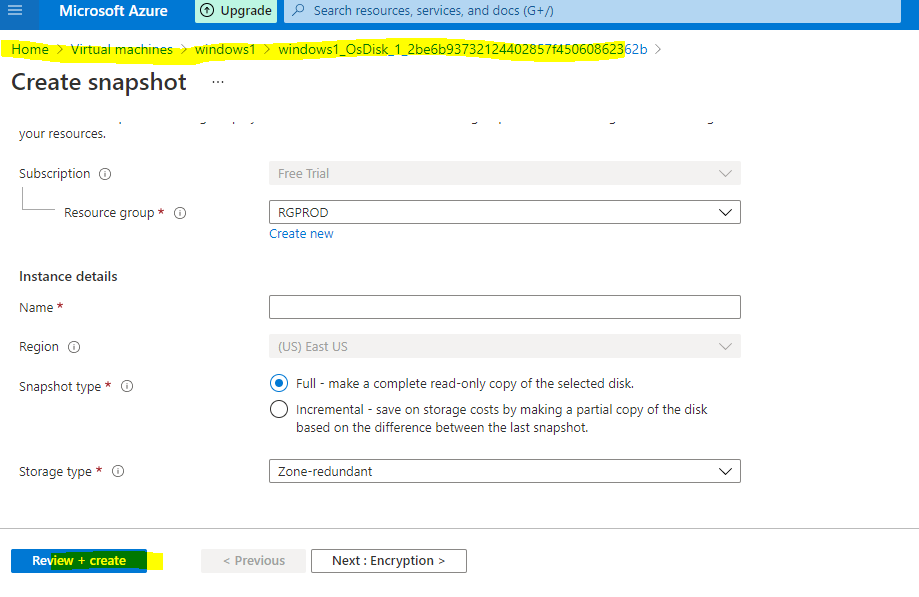
Take a snapshot of an OS or data disk for backup or to troubleshoot VM issues. A snapshot is a full, read-only copy of a VHD.

## Use Azure portal

1. Sign in to the [Azure portal](https://portal.azure.com/).
2. Starting in the upper-left, click **Create a resource** and search for **snapshot**. Select **Snapshot** from the search results.
3. In the **Snapshot** blade, click **Create**.
4. Enter a **Name** for the snapshot.
5. Select an existing resource group or type the name for a new one.
6. For **Source disk**, select the managed disk to snapshot.
7. Select the **Account type** to use to store the snapshot. Use **Standard HDD** unless you need it stored on a high performing SSD.
8. Click **Create**.

## Next steps

Create a virtual machine from a snapshot by creating a managed disk from the snapshot and then attaching the new managed disk as the OS disk.



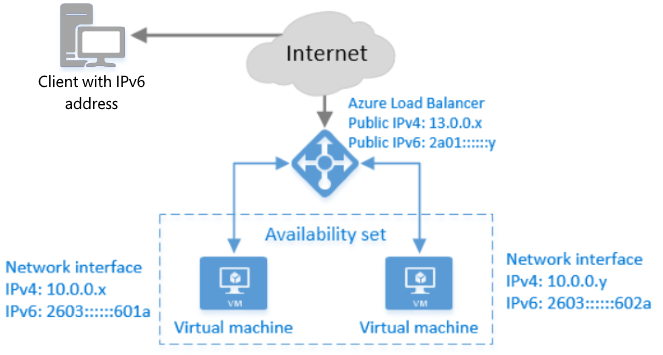
**Azure Load balancer:**

Load balancing refers to evenly distributing load (incoming network traffic) across a group of backend resources or servers.

Azure Load Balancer operates at layer 4 of the Open Systems Interconnection (OSI) model. It's the single point of contact for clients. Load balancer distributes inbound flows that arrive at the load balancer's front end to backend pool instances. These flows are according to configured load-balancing rules and health probes. The backend pool instances can be Azure Virtual Machines or instances in a virtual machine scale set.

A [**public load balancer**](https://docs.microsoft.com/en-us/azure/load-balancer/components#frontend-ip-configurations) can provide outbound connections for virtual machines (VMs) inside your virtual network. These connections are accomplished by translating their private IP addresses to public IP addresses. Public Load Balancers are used to load balance internet traffic to your VMs.

An [**internal (or private) load balancer**](https://docs.microsoft.com/en-us/azure/load-balancer/components#frontend-ip-configurations) is used where private IPs are needed at the frontend only. Internal load balancers are used to load balance traffic inside a virtual network. A load balancer frontend can be accessed from an on-premises network in a hybrid scenario.

****

## Why use Azure Load Balancer?

With Azure Load Balancer, you can scale your applications and create highly available services. Load balancer supports both inbound and outbound scenarios. Load balancer provides low latency and high throughput, and scales up to millions of flows for all TCP and UDP applications.

Key scenarios that you can accomplish using Azure Standard Load Balancer include:

* Load balance [**internal**](https://docs.microsoft.com/en-us/azure/load-balancer/quickstart-load-balancer-standard-internal-portal) and [**external**](https://docs.microsoft.com/en-us/azure/load-balancer/quickstart-load-balancer-standard-public-portal) traffic to Azure virtual machines.
* Increase availability by distributing resources [**within**](https://docs.microsoft.com/en-us/azure/load-balancer/tutorial-load-balancer-standard-public-zonal-portal) and [**across**](https://docs.microsoft.com/en-us/azure/load-balancer/tutorial-load-balancer-standard-public-zone-redundant-portal) zones.
* Configure [**outbound connectivity**](https://docs.microsoft.com/en-us/azure/load-balancer/load-balancer-outbound-connections)for Azure virtual machines.
* Use [**health probes**](https://docs.microsoft.com/en-us/azure/load-balancer/load-balancer-custom-probe-overview) to monitor load-balanced resources.
* Employ [**port forwarding**](https://docs.microsoft.com/en-us/azure/load-balancer/tutorial-load-balancer-port-forwarding-portal) to access virtual machines in a virtual network by public IP address and port.
* Enable support for [**load-balancing**](https://docs.microsoft.com/en-us/azure/virtual-network/virtual-network-ipv4-ipv6-dual-stack-standard-load-balancer-powershell) of [**IPv6**](https://docs.microsoft.com/en-us/azure/virtual-network/ipv6-overview).
* Standard load balancer provides multi-dimensional metrics through [Azure Monitor](https://docs.microsoft.com/en-us/azure/azure-monitor/overview). These metrics can be filtered, grouped, and broken out for a given dimension. They provide current and historic insights into performance and health of your service. [Insights for Azure Load Balancer](https://docs.microsoft.com/en-us/azure/load-balancer/load-balancer-insights) offers a preconfigured dashboard with useful visualizations for these metrics. Resource Health is also supported. Review [**Standard load balancer diagnostics**](https://docs.microsoft.com/en-us/azure/load-balancer/load-balancer-standard-diagnostics) for more details.
* Load balance services on [**multiple ports, multiple IP addresses, or both**](https://docs.microsoft.com/en-us/azure/load-balancer/load-balancer-multivip-overview).
* Move [**internal**](https://docs.microsoft.com/en-us/azure/load-balancer/move-across-regions-internal-load-balancer-portal) and [**external**](https://docs.microsoft.com/en-us/azure/load-balancer/move-across-regions-external-load-balancer-portal) load balancer resources across Azure regions.
* Load balance TCP and UDP flow on all ports simultaneously using [**HA ports**](https://docs.microsoft.com/en-us/azure/load-balancer/load-balancer-ha-ports-overview).

**Create azure internal Load balancer:**

When you create an internal load balancer, a virtual network is configured as the network for the load balancer.

A private IP address in the virtual network is configured as the frontend (named as **LoadBalancerFrontend** by default) for the load balancer.

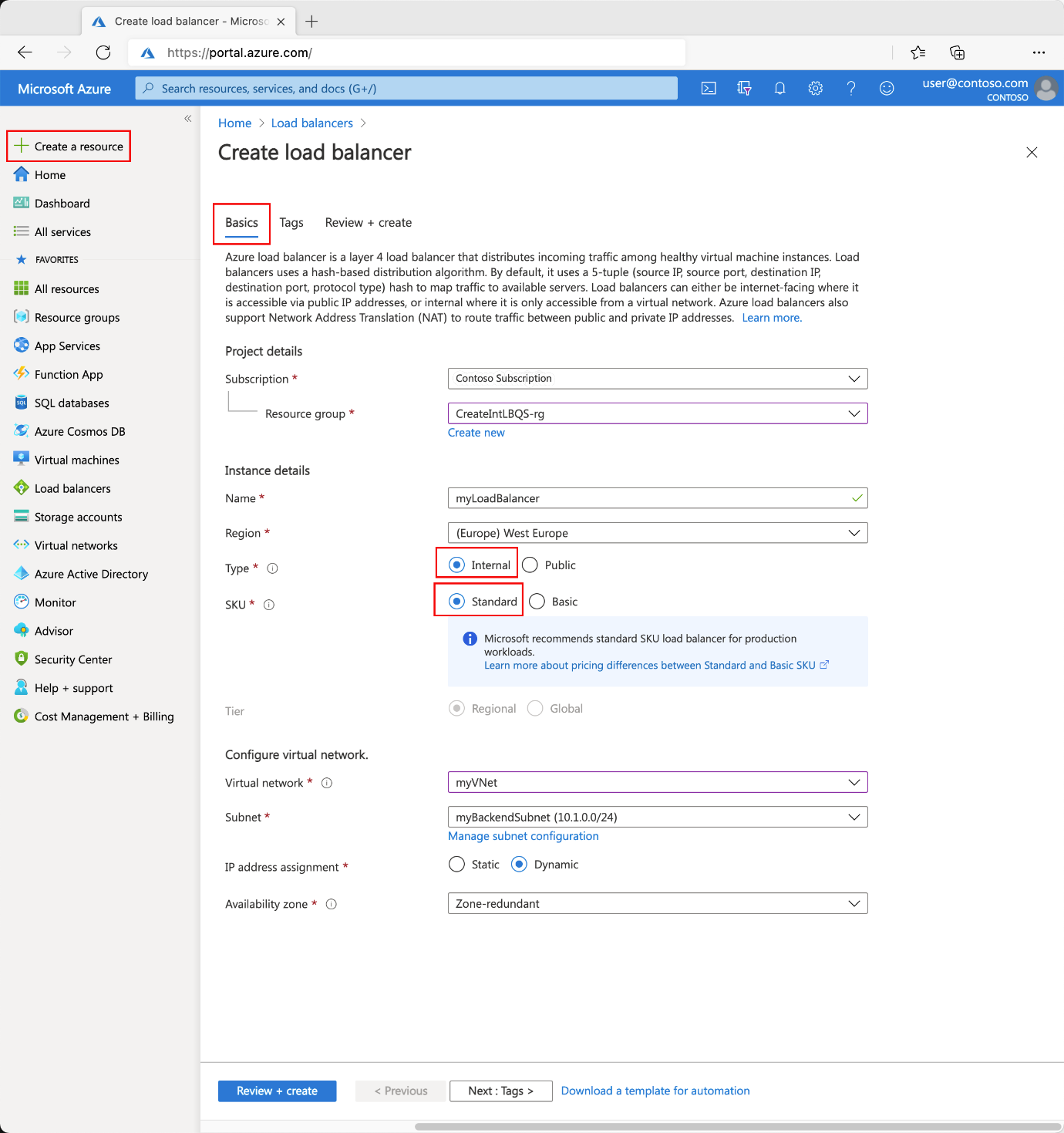
The frontend IP address can be **Static** or **Dynamic**.

**Create azure vnet as explained earlier**

## Create load balancer

1. Select **Create a resource**.
2. In the search box, enter **Load balancer**. Select **Load balancer** in the search results.
3. In the **Load balancer** page, select **Create**.
4. On the **Create load balancer** page enter, or select the following information:

|  | |
| --- | --- |
| **Setting** | **Value** |
| Subscription | Select your subscription. |
| Resource group | Select **CreateIntLBQS-rg** created in the previous step. |
| Name | Enter **myLoadBalancer** |
| Region | Select **(Europe) West Europe**. |
| Type | Select **Internal**. |
| SKU | Select **Standard** |
| Virtual network | Select **myVNet** created in the previous step. |
| Subnet | Select **myBackendSubnet** created in the previous step. |
| IP address assignment | Select **Dynamic**. |
| Availability zone | Select **Zone-redundant** |

1. Accept the defaults for the remaining settings, and then select **Review + create**.
2. In the **Review + create** tab, select **Create**. 

## Create load balancer resources

In this section, you configure:

* Load balancer settings for a backend address pool.
* A health probe.
* A load balancer rule.

### Create a backend pool

A backend address pool contains the IP addresses of the virtual (NICs) connected to the load balancer.

Create the backend address pool **myBackendPool** to include virtual machines for load-balancing internet traffic.

1. Select **All services** in the left-hand menu, select **All resources**, and then select **myLoadBalancer** from the resources list.
2. Under **Settings**, select **Backend pools**, then select **Add**.
3. On the **Add a backend pool** page, for name, type **myBackendPool**, as the name for your backend pool, and then select **Add**.

### Create a health probe

The load balancer monitors the status of your app with a health probe.

The health probe adds or removes VMs from the load balancer based on their response to health checks.

Create a health probe named **myHealthProbe** to monitor the health of the VMs.

1. Select **All services** in the left-hand menu, select **All resources**, and then select **myLoadBalancer** from the resources list.
2. Under **Settings**, select **Health probes**, then select **Add**.

|  | |
| --- | --- |
| **Setting** | **Value** |
| Name | Enter **myHealthProbe**. |
| Protocol | Select **HTTP**. |
| Port | Enter **80**. |
| Interval | Enter **15** for number of **Interval** in seconds between probe attempts. |
| Unhealthy threshold | Select **2** for number of **Unhealthy threshold** or consecutive probe failures that must occur before a VM is considered unhealthy. |
|  |  |

1. Leave the rest the defaults and Select **OK**.

### Create a load balancer rule

A load balancer rule is used to define how traffic is distributed to the VMs. You define the frontend IP configuration for the incoming traffic and the backend IP pool to receive the traffic. The source and destination port are defined in the rule.

In this section, you'll create a load balancer rule:

* Named **myHTTPRule**.
* In the frontend named **LoadBalancerFrontEnd**.
* Listening on **Port 80**.
* Directs load balanced traffic to the backend named **myBackendPool** on **Port 80**.

1. Select **All services** in the left-hand menu, select **All resources**, and then select **myLoadBalancer** from the resources list.
2. Under **Settings**, select **Load balancing rules**, then select **Add**.
3. Use these values to configure the load-balancing rule:

| **TABLE 7** | |
| --- | --- |
| **Setting** | **Value** |
| Name | Enter **myHTTPRule**. |
| IP Version | Select **IPv4** |
| Frontend IP address | Select **LoadBalancerFrontEnd** |
| Protocol | Select **TCP**. |
| Port | Enter **80**. |
| Backend port | Enter **80**. |
| Backend pool | Select **myBackendPool**. |
| Health probe | Select **myHealthProbe**. |
| Idle timeout (minutes) | Move the slider to **15** minutes. |
| TCP reset | Select **Enabled**. |

1. Leave the rest of the defaults and then select **OK**.

Then create the virtual machines as explained above.

**What is Azure AD?**

Azure Active Directory is Microsoft’s multi-tenant, cloud-based identity and access management service. It’s the digital infrastructure that allows your employees to sign in and access external resources held in Office 365 and an ever-growing list of other SaaS applications, as well as those held on a corporate network or intranet.

Either way, it enables both on-premises and cloud-based users to access the same apps and resources, simultaneously benefitting from features such as single sign-on (SSO), multi-factor authentication (MFA), conditional access and more.

More importantly, it provides a single place from which to manage your identity, security and compliance controls across your entire IT estate.

## What does Azure AD do?

Azure AD provides different benefits depending on what you’re using it for.

For IT admins, it allows complete control over access to applications and resources utilising security controls like MFA and conditional access. They can also use [Azure AD’s built-in governance controls](https://thirdspace.net/blog/understanding-azure-ad-identity-governance/) to apply automated lifecycle management and privileged access limitations.

In addition to this, Azure AD also provides admins with [the ability to automate provisioning between Windows Server Active Directory and cloud apps like Office 365](https://thirdspace.net/events/webinar-how-to-connect-apps-user-provisioning/).

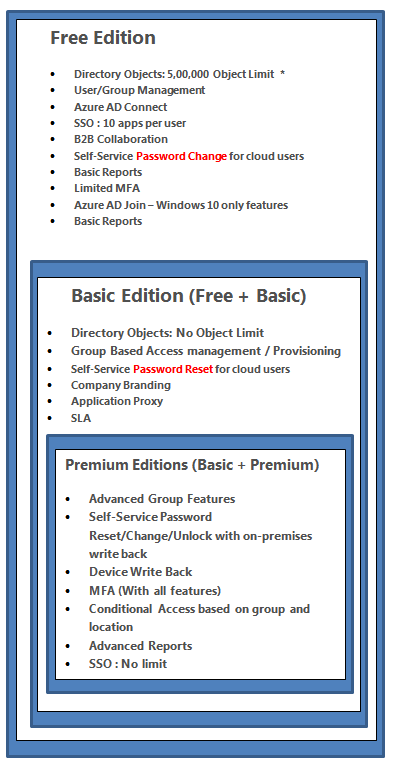
For developers, Azure AD can be used as a standards-based approach to enabling features like SSO and for personalising the app experiences using existing organisation data through APIs.

If you’re a user or employee, Azure AD means quick and easy access to work resources, on a multitude of devices, from almost anywhere on the planet.

**Types of Azure Active Directory:**

* **Basic editioin**
* **Premium edition1**
* **Premium edition 2**

**Note: Ignore the names on screenshots**

****

**How does it work?**

Azure AD, as the name suggests, is a directory – a container for your user names, credentials and access rights (typically to information-based resources).

### Cloud-only or hybrid

Azure AD can be operated in ‘cloud-only’ mode, allowing your users to sign in to their Windows PCs using the cloud directory service. Alternatively, if you, like many organisations, are still tied to on-premise legacy infrastructure, Azure AD can use your local Active Directory as a master for account data and operate in a variety of hybrid modes.

### Threat detection

Whether in cloud-only or hybrid mode, Azure AD effectively acts as your ‘front door’ for sign-ins. A key benefit of doing so allows you to take advantage of state-of-the-art security measures, such as assessing the threat level of the user attempting access and being able to mitigate that threat – for example, adding two-factor authentication.

### Single sign-on compatibility

One of the most attractive advantages of using Azure AD is its ability to enable single sign-on (SSO) and it supports third-party application integration to help achieve this.

Applications can connect using standard ‘modern auth’ protocols – SAML or OpenID Connect. Application and group assignments (including dynamic groups) in Azure AD determine who has access to what.

Single sign-on means that users will be able to access all of the applications they need by signing in only once using a single user account hosted in Azure AD. Once signed in, they can access those applications without being required to authenticate a second time.

Azure AD has been designed to enable easy integration with many of today’s popular SaaS applications, enabling users to either single sign-on to applications directly or discover and launch them from a portal, such as Office 365 or the Azure AD access panel.

## What are the benefits of using Azure AD?

Azure AD offers a plethora of incentives for adoption, hence why [it’s used by 95% of the Fortune500](https://azure.microsoft.com/en-us/overview/azure-vs-aws/).

Again, this is driven by its incredible flexibility. Whilst Azure AD is optimised for Microsoft applications, it is also highly compatible with apps developed outside the house that Bill built.

This open standards approach has allowed Azure AD to become the core mechanism by which an organisation can manage all of its different apps, devices and users across multiple tenants.

**Azure AD’s key benefits largely fall into five categories:**

### 1. One place for identity and access management

Azure AD is the heart of your organisation’s IT, giving you one place to go for managing user identities and permissions. You can assign users to groups individually or using rules driven by attributes, and you can use groups to assign licences and application access. You have all the control in one place.

### 2. One identity for all applications

Whilst your users’ Azure AD identities are perfect for signing into Microsoft applications, it is also highly compatible with apps developed everywhere else. Of those 251 million monthly users mentioned earlier, 44 million of those use Azure AD to regularly access 3rd party applications, streamlining the process and increasing productivity.

### 3. Security

Organisations want to protect their resources from malicious or accidental harm – and to protect their users from identity theft. Azure AD achieves these aims with a range of measures, including threat detection, conditional access, multi-factor authentication, privileged identity management (PIM), and more.

### 4. Ease of use

Getting access to resources should be easy for end-users. Single sign-on, using the same sign in for Windows and all your applications, means less fuss with credentials, and fewer demands on the IT help desk.

### 5. Collaboration

Azure AD allows you to invite external (guest) users into your directory to assign access, while their credentials are managed by their organisation’s IT department.

This gives you immediate and easy collaboration options while not having to worry about user lifecycle.

## Azure AD Key features

Having all of your disparate environment united under Azure AD offers some significant functionality option and features:

#### Application management

Manage both cloud and on-premises apps, single sign-on, the MyApps portal, and any SaaS apps.

#### Authentication

Whether this be providing self-service password reset, calibrating MFA requirements, or enabling smart lockout, you can get really granular with your authentication settings (especially when used in conjunction with conditional access) for increased security and control.

#### Business-to-business (B2B)

Manage guest users and partners, providing them the access you’re willing to allow.

#### Business-to-customer (B2C)

Offer custom sign in and sign up experiences, allowing customers to manage their profiles within your applications.

#### Device management

Control how your network is accessed by on-premises and external devices, utilising Intune to keep data secure.

#### Hybrid identity

Most organisations aren’t ready to go cloud-only yet, but using Azure AD Connect allows you to take advantage of Azure AD’s features - even if you’re running some on-premise applications and some in the Cloud.

#### Identity governance

To ensure that your identity ecosystem remains healthy, Azure AD has some built-in governance features that allow you to manage identity and access lifecycles and set privileged access conditions.  
  
These controls are designed to enable organisations to ensure that the correct users have the corresponding levels of access and monitor what they’re doing with it. One of the key benefits of good governance is being able to audit and verify the effectiveness of the applied controls.

#### Identity protection

Azure AD Identity Protection utilises security information drawn from across Microsoft’s digital empire to detect and remedy identity-based risks, automating a large part of the process of identifying and addressing security concerns.  
  
These risks can then be further investigated through the Azure AD portal.

#### Reports and monitoring

Azure AD also features monitoring and reporting capabilities to help you gain insights into your environment. You can run diagnostics and view logs which can then also be applied to third-party SIEM tools ([or Microsoft’s own Azure Sentinel](https://thirdspace.net/blog/definitive-guide-microsoft-azure-sentinel-everything-need-know/)) to take a deeper dive into your data.

**Azure Storage Account:**

An Azure storage account contains all of your Azure Storage data objects: blobs, files, queues, tables, and disks. The storage account provides a unique namespace for your Azure Storage data that is accessible from anywhere in the world over HTTP or HTTPS. Data in your Azure storage account is durable and highly available, secure, and massively scalable.

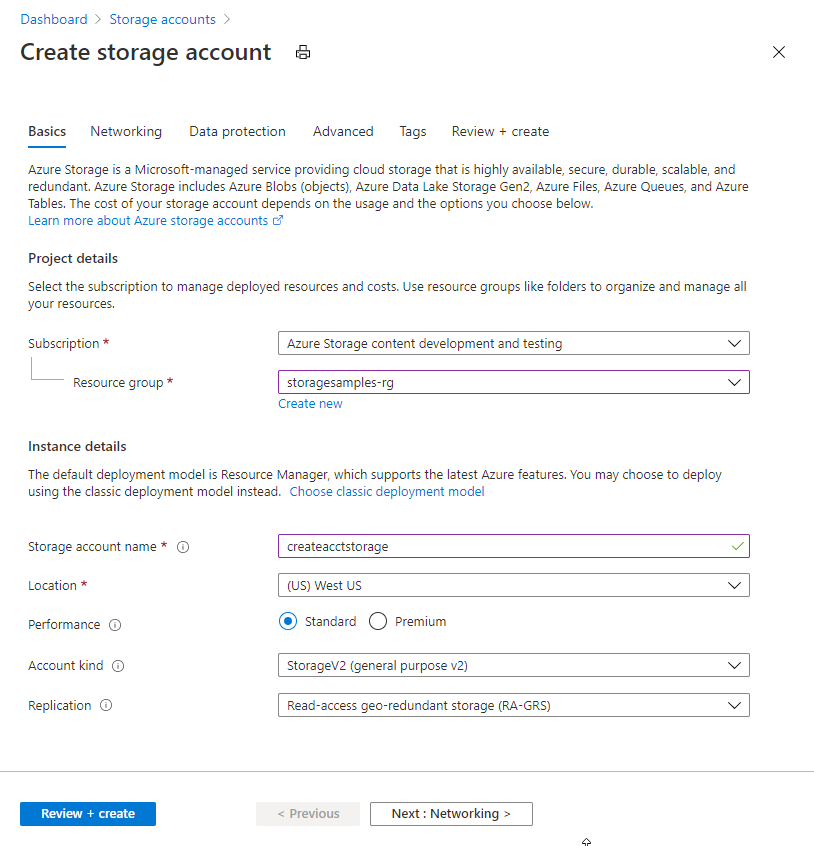
A **general-purpose v2** storage account provides access to all of the Azure Storage services: blobs, files, queues, tables, and disks. The steps outlined here create a general-purpose v2 storage account, but the steps to create any type of storage account are similar. For more information about types of storage accounts and other storage account setting

**Create azure storage account on portal:**

To create a general-purpose v2 storage account in the Azure portal, follow these steps:

1. On the Azure portal menu, select **All services**. In the list of resources, type **Storage Accounts**. As you begin typing, the list filters based on your input. Select **Storage Accounts**.
2. On the **Storage Accounts** window that appears, choose **Add**.
3. On the **Basics** tab, select the subscription in which to create the storage account.
4. Under the **Resource group** field, select your desired resource group, or create a new resource group.
5. Next, enter a name for your storage account. The name you choose must be unique across Azure. The name also must be between 3 and 24 characters in length, and may include only numbers and lowercase letters.
6. Select a location for your storage account, or use the default location.
7. Select a performance tier. The default tier is Standard.
8. Set the **Account kind** field to Storage V2 (general-purpose v2).
9. Specify how the storage account will be replicated. The default replication option is Read-access geo-redundant storage (RA-GRS
10. Additional options are available on the **Networking**, **Data protection**, **Advanced**, and **Tags** tabs. To use Azure Data Lake Storage, choose the **Advanced** tab, and then set **Hierarchical namespace** to **Enabled**.
11. Select **Review + Create** to review your storage account settings and create the account.
12. Select **Create**.

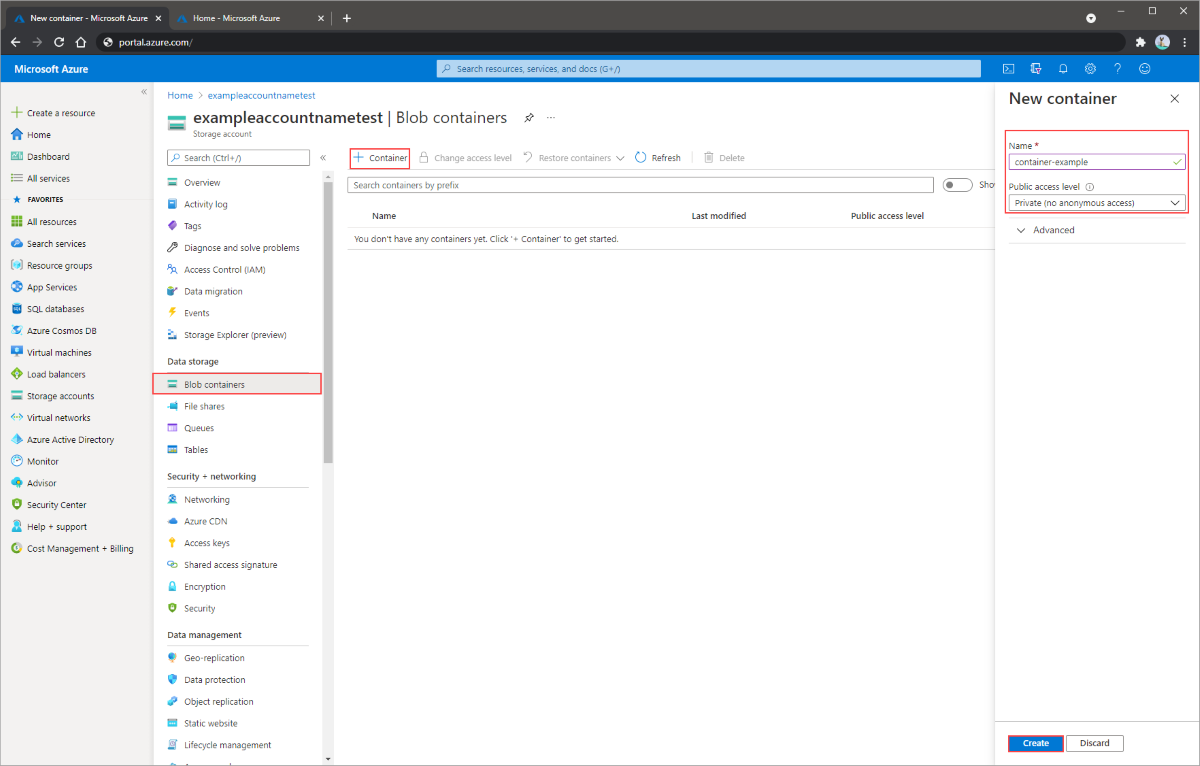
The following image shows the settings on the **Basics** tab for a new storage account:



## Create a container

To create a container in the Azure portal, follow these steps:

1. Navigate to your new storage account in the Azure portal.
2. In the left menu for the storage account, scroll to the **Data storage** section, then select **Blob containers**.
3. Select the **+ Container** button.
4. Type a name for your new container. The container name must be lowercase, must start with a letter or number, and can include only letters, numbers, and the dash (-) character. For more information about container and blob names, see [Naming and referencing containers, blobs, and metadata](https://docs.microsoft.com/en-us/rest/api/storageservices/naming-and-referencing-containers--blobs--and-metadata).
5. Set the level of public access to the container. The default level is **Private (no anonymous access)**.
6. Select **OK** to create the container.

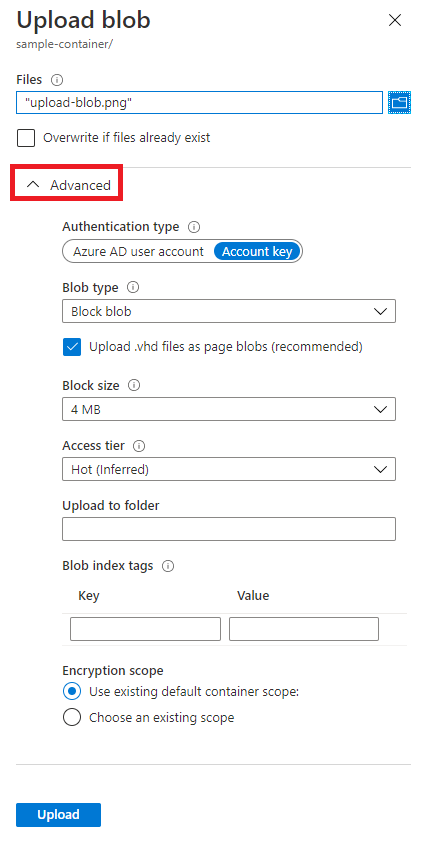
[](https://docs.microsoft.com/en-us/azure/storage/blobs/media/storage-quickstart-blobs-portal/create-container.png#lightbox)

## Upload a block blob

Block blobs consist of blocks of data assembled to make a blob. Most scenarios using Blob storage employ block blobs. Block blobs are ideal for storing text and binary data in the cloud, like files, images, and videos. This quickstart shows how to work with block blobs.

To upload a block blob to your new container in the Azure portal, follow these steps:

1. In the Azure portal, navigate to the container you created in the previous section.
2. Select the container to show a list of blobs it contains. This container is new, so it won't yet contain any blobs.
3. Select the **Upload** button to open the upload blade and browse your local file system to find a file to upload as a block blob. You can optionally expand the **Advanced** section to configure other settings for the upload operation.

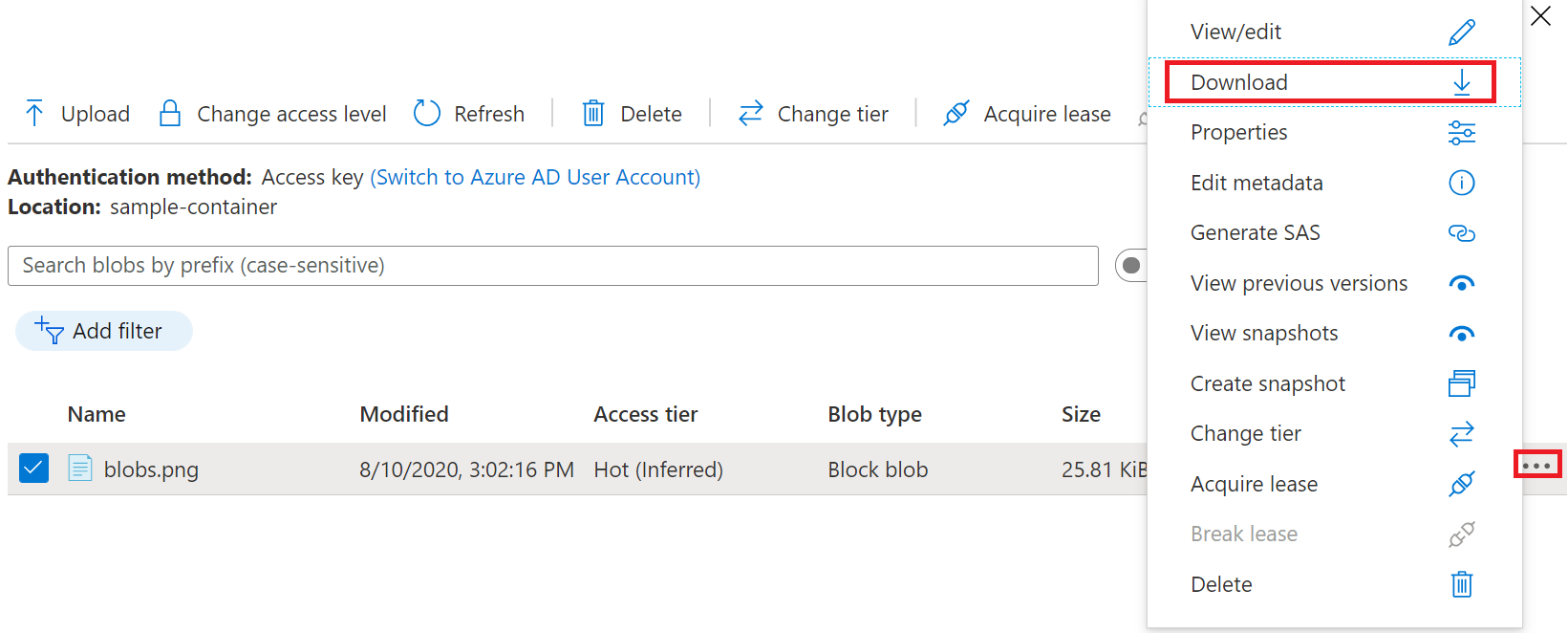


1. Select the **Upload** button to upload the blob.
2. Upload as many blobs as you like in this way. You'll see that the new blobs are now listed within the container.

## Download a block blob

You can download a block blob to display in the browser or save to your local file system. To download a block blob, follow these steps:

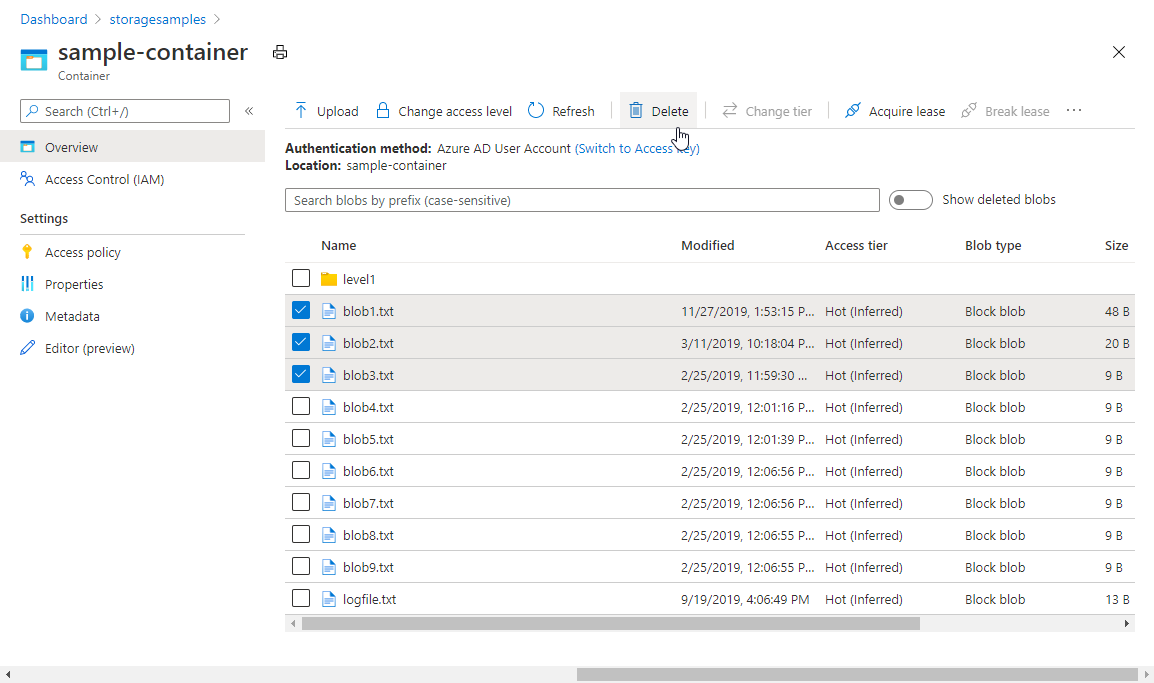
1. Navigate to the list of blobs that you uploaded in the previous section.
2. Right-click the blob you want to download, and select **Download**.



## Delete a block blob

To delete one or more blobs in the Azure portal, follow these steps:

1. In the Azure portal, navigate to the container.
2. Display the list of blobs in the container.
3. Use the checkbox to select one or more blobs from the list.
4. Select the **Delete** button to delete the selected blobs.
5. In the dialog, confirm the deletion, and indicate whether you also want to delete blob snapshots.



## Clean up resources

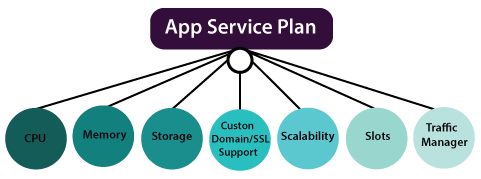
To remove all the resources you created in this quickstart, you can simply delete the container. All blobs in the container will also be deleted.

To delete the container:

1. In the Azure portal, navigate to the list of containers in your storage account.
2. Select the container to delete.
3. Select the **More** button (**...**), and select **Delete**.
4. Confirm that you want to delete the container.

**App Service plan**

An app service plan denotes a set of features and capacity that we can share across multiple apps in the same subscription and geographical region. A single or dual app can be configured to run on the same computing resources.



**Each App Service plan defines:**

* Region (West US, East US, etc.)
* Number of VM instances
* Size of VM instances (Small, Medium, Large)
* Pricing tier
  + ***Shared compute****:* Free and shared, the two basic tiers, runs an app over the same Azure VM as other App Service app runs, including apps of different customers.
  + ***Dedicated compute****:* Basic, Standard, Premium, and PremiumV2 tiers run apps on a fixed Azure VM.
  + ***Isolated:*** This tier runs dedicated Azure VMs on dedicated Azure Virtual Networks, which provides network isolation on top of computing isolation to your apps.
  + ***Consumption****:* It is only available to function apps. It scales the functions dynamically, depending on the workload.

**Environment features**

* **Development frameworks:** App Service supports a variety of development frameworks, including ASP.NET, classic ASP, node.js, PHP, and Python- all of which run as extensions within IIS.
* **File access**
  + ***Local drives*** - Operating system drive (D:\drive), an application drive and user drive (the C:\ drive)
  + ***Network drives*** - Each customer's subscription has a reserved directory structure on a specific UNC share within a data center.
* **Network access:** The application code can use TCP/IP and UDP based protocols to make outbound network connections to access Internet endpoints that expose external services.

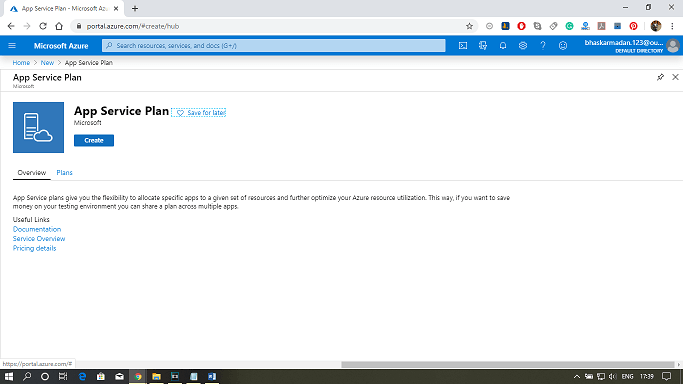
**Web apps Overview**

Azure App Service Web Apps is a service for hosting web applications. The key feature of App Service Web Apps.

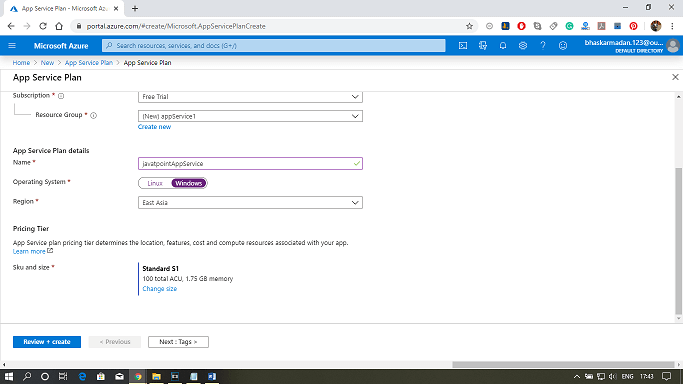
* Multiple language and frameworks
* DevOps optimizations
* Security & Compliance
* Application template
* Visual Studio integration

**Creating App Service Plan in Azure Portal**

**Step 1:** Click on *create a new resource* and search for App Service Plan to create it.



**Step 2:** Fill-in all the required details and select the SKU size, as shown in the figure below. Then click on create.



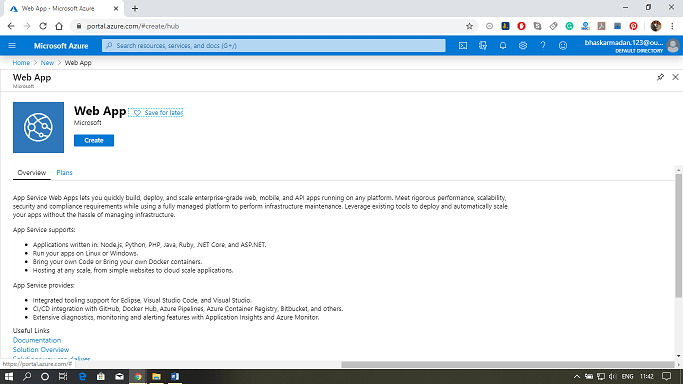
**Step 3:** Your app service plan will be created. You can now explore and modify it as per your requirement.

# **Azure Web App**

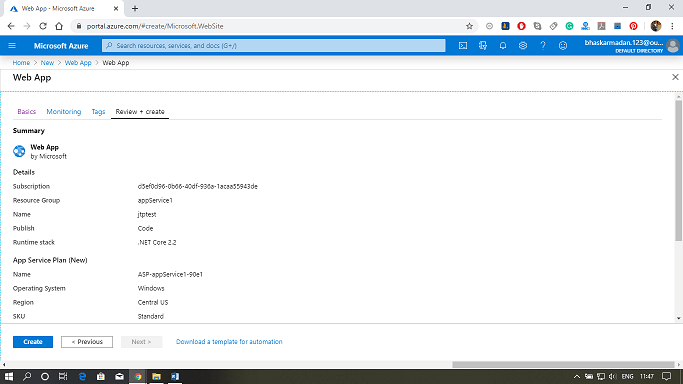
Azure Web App service lets us quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. It helps us to meet rigorous performance, scalability, security, and compliance requirements while using a fully managed platform to perform infrastructure maintenance.

**Creating a Web App and deploying an application into Azure web App from visual studio**

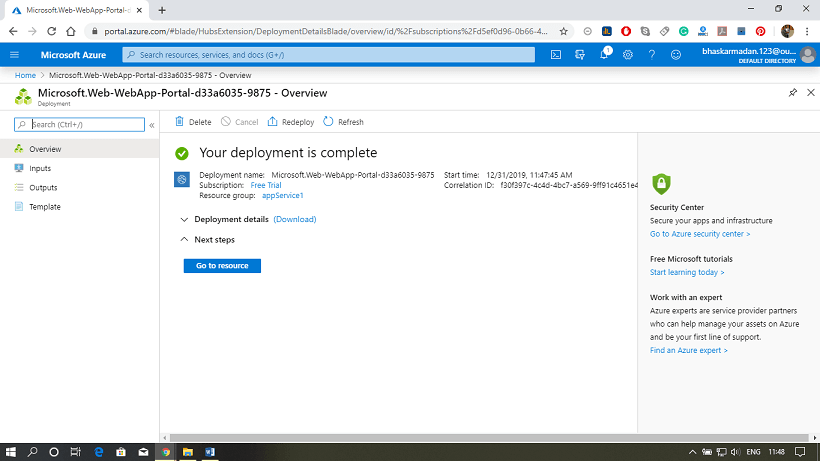
**Step 1:** Click on create a resource and type in the web app. After that, click on the web app and then click on Create.



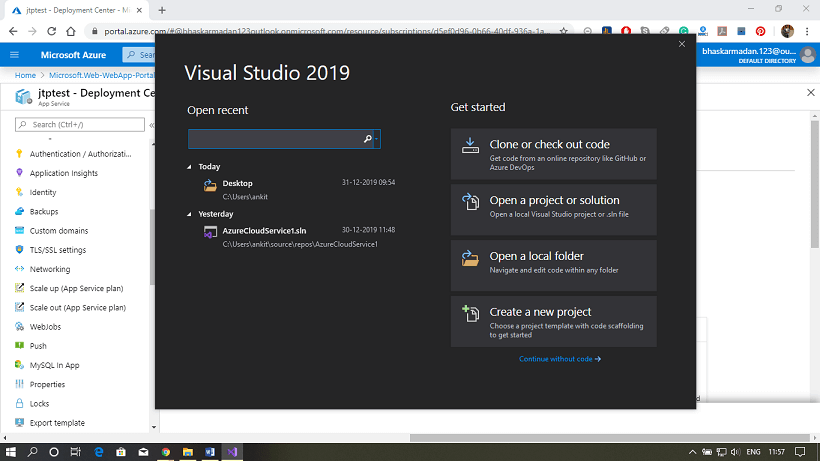
**Step 2:** You are now on the Web App creation page. Fill-in, all the required details, then click on review+create.



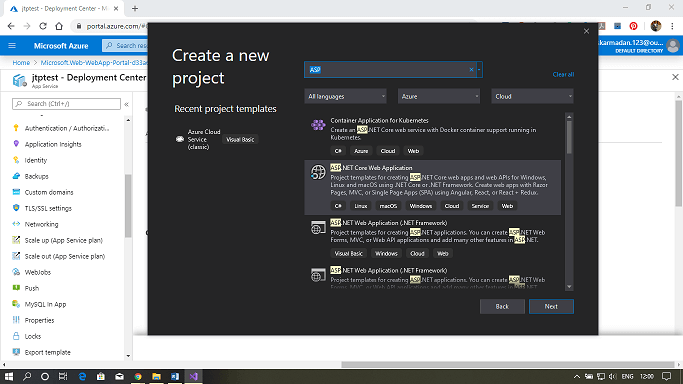
**Step 3:** Click on create, then you will be redirected to the following page.



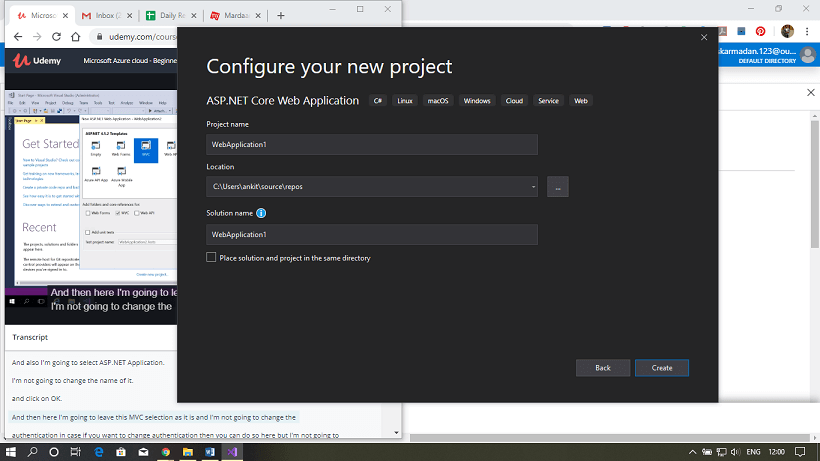
**Step 4:** Open Visual Studio, then click on Create a new project.



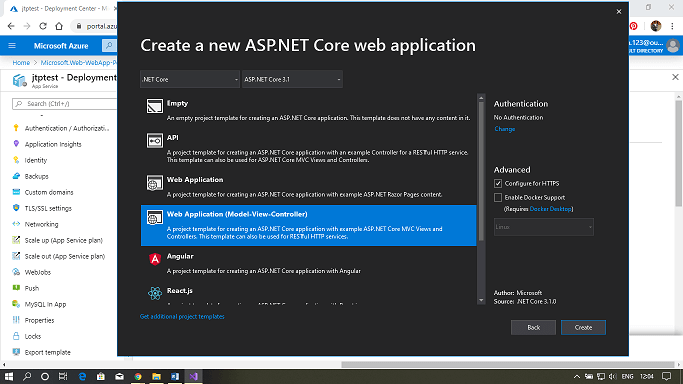
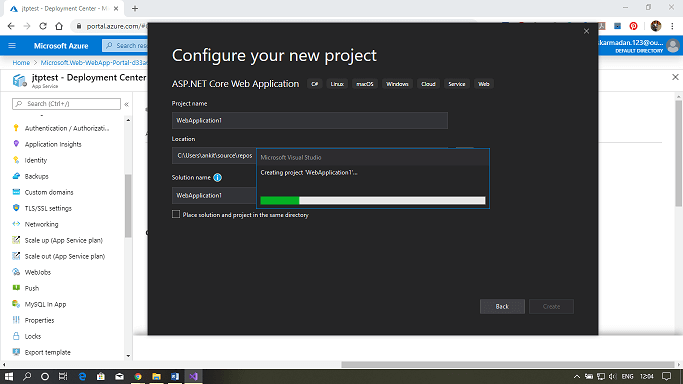
**Step 5:** Search for ASP.net web application and click on it.



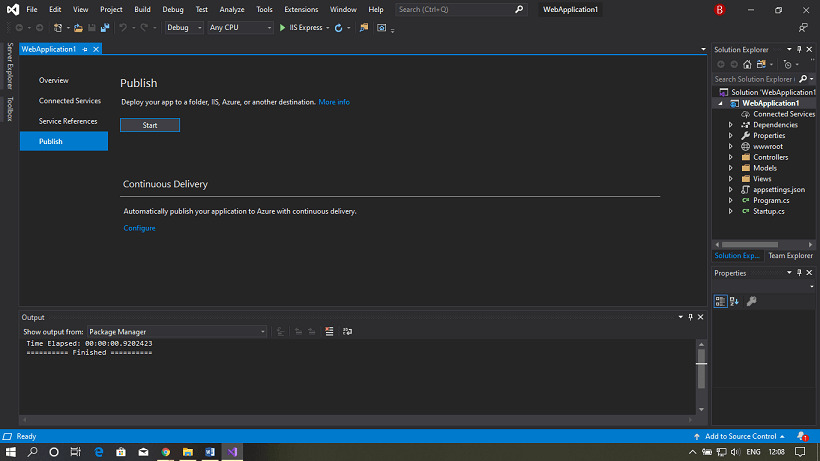
**Step 6:** Now, configure your project and click on create.



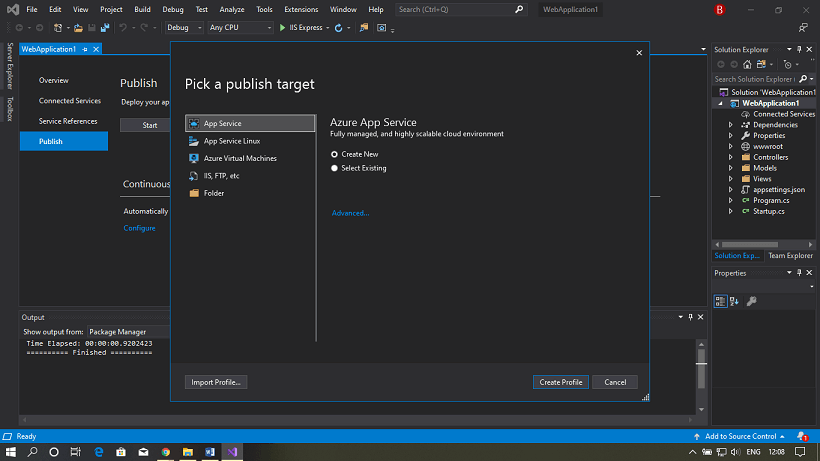
**Step 7:** Now select the Web application (Model View Controller) option from available templates. Then Click on create.

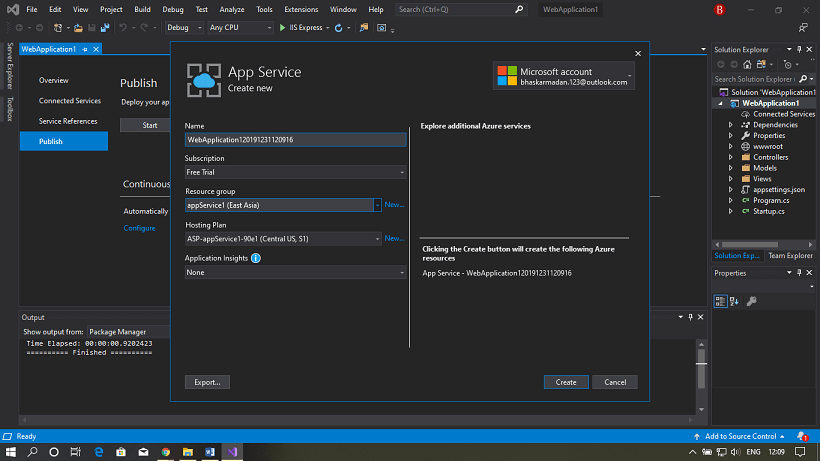
**Step 8:** Your project will be created, now click on publish to configure it with the Azure portal.



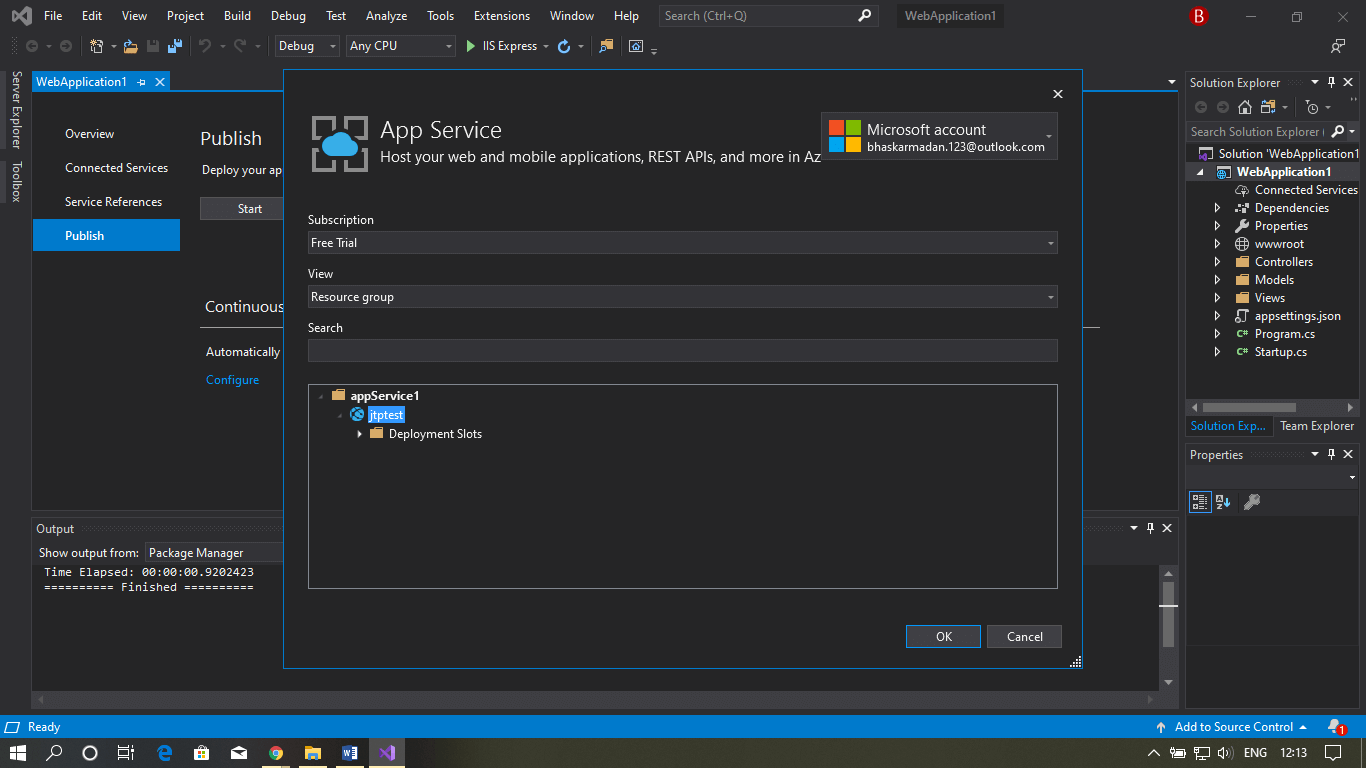
**Step 9:** Here, either you can create a new service plan, or you can use an existing one.



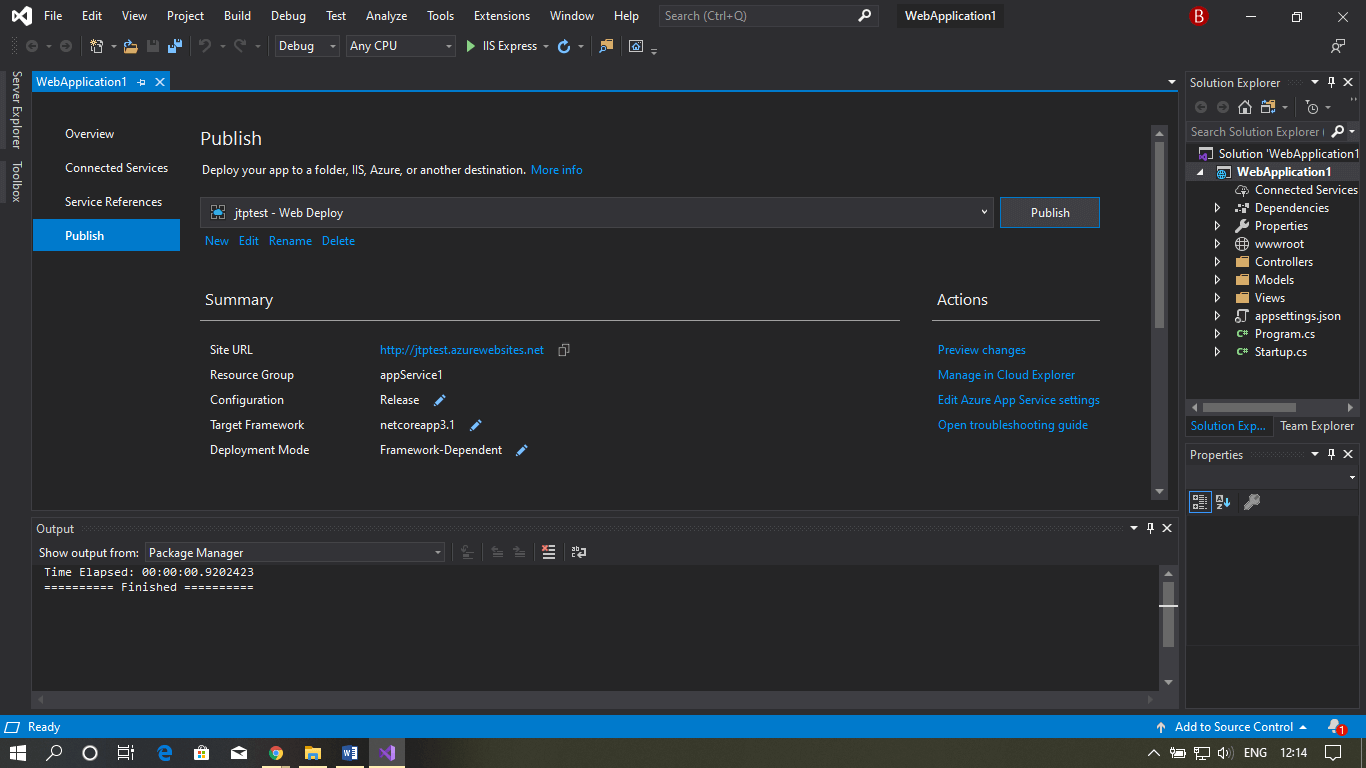
**Step 10:** Let's see how to create a new one, click on Create new. Fill all the details and click on create a profile.



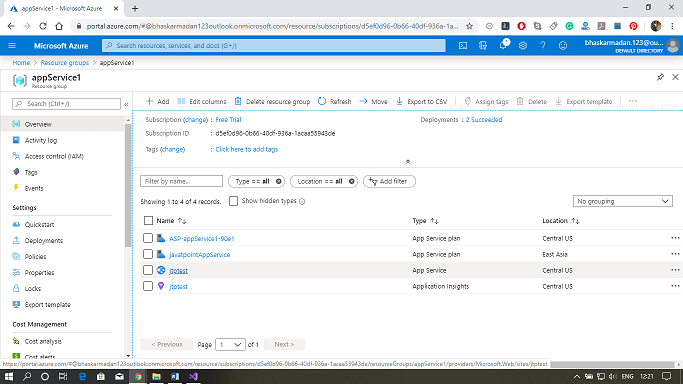
**Step 11:** As we have already created an App service previously, so we are using that one here. Go back and click on *Select existing*.



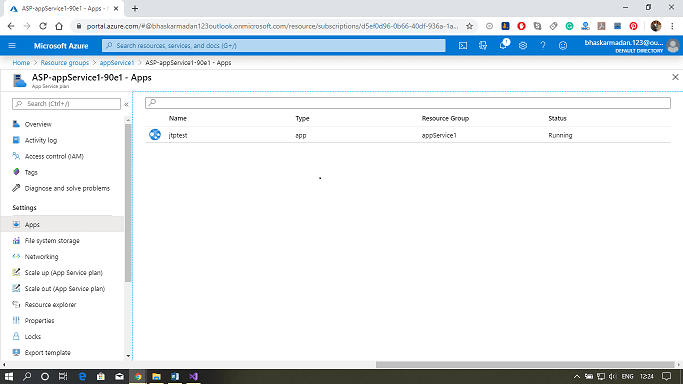
**Step 12:** Now, click on the file name and then click on, Ok.



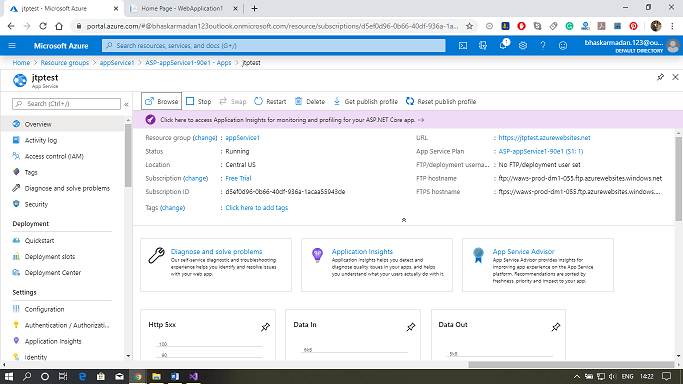
**Step 13:** Now go to the Azure portal and click on the storage account.



**Step 14:** Click on the app section, here you can view the web app that you have created.



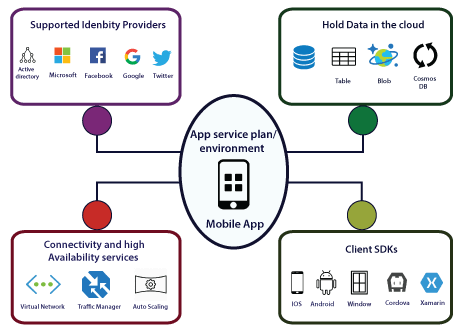
**Step 15:** Click on the browse button above to see that your Web app is working or not.



# Azure Web App

# **Azure Mobile App**

We can deploy our mobile backend services on Azure using Azure Mobile apps. By implementing our mobile backend service on Azure, our mobile backend will be able to communicate with different Azure services. We can able to take advantage of various features that are provided by Azure Mobile Apps.



## Features and services:

**Data Storage:** Our mobile backend can be able to store the data or access the data of Azure SQL database Azure table storage, Azure Blob storage, and Cosmos DB. And also, we can add offline sync features to our mobile app.

**Client-Side development:** Once we host our mobile backend services on Azure, there must be a way to consume those services. For that purpose, Microsoft provided several client SDK's depending upon the platform.

**Authentication and Authorization:** We can integrate our mobile app with different authentication providers. So we can integrate with Azure active directory, Microsoft, Facebook, Google, and Twitter. We can integrate this service without any code.

**Connectivity:** In terms of connectivity to apps that are hosted in a virtual network. Our mobile app will be able to connect to a virtual network either using point to site VPN or by hosting our mobile app backend service into an app service environment, which will automatically get connected to the virtual network.

**Availability Service:** We can use the traffic manager to provide higher resilience even during the data center failures. We can also use auto-scaling to scale our mobile backend services as more number of users are about to start using our app.

**Notification Hub:** Using the notification hub, we can enable push notifications to different platforms, and also we can use mobile engagement using which we can understand what exactly the user is doing with our app.

**How to create Mobile apps backend development**

**Step 1:** Log in to the Azure portal and create a new Azure mobile app backend.

**Step 2:** Configure the mobile app backend.

**Step 3:** Define a table controller.

**Step 4:** Create the Data Transfer Object (DTO) class.

**Step 5:** Configure a table reference in the Mobile DbContext class.

**Step 6:** Create a table controller.

**Step 7:** Define a custom API controller.

**Mobile Client-side development**

**Step 1:** Based on mobile OS, download the client-side SDK.

**Step 2:** Reference the MicrosoftAzureMobile (IOS) in your client code.

**Step 3:** Create MSClient (IOS) reference and start accessing data from tables.

**Step 4:** For Custom API's, use MSClient.invokeAPI to call custom API.

## Mobile offline data sync

Mobile offline data sync is a client and server SDK feature of Azure Mobile Apps that makes it possible to create apps that work without a network connection.

**Sync Table**

* To access the "/tables" endpoint, Azure Mobile client SDKs provide an interface such as MSTable. However, this will fail if the client devices do not have a network connection.
* To support offline use, our app should instead use the sync table APIs such as MSSyncTable. All the CRUD operations will happen at a local store.

The local store is the data persistence layer on the client device (Windows, Xamarin, and Android). It is based on SQLite, whereas on iOS, it is based on core data. In offline synchronization, the sync can be a push, pull, implicit pushes, or Incremental sync.

**Webjobs:**

WebJobs is a feature of [Azure App Service](https://docs.microsoft.com/en-us/azure/app-service/) that enables you to run a program or script in the same instance as a web app, API app, or mobile app. There is no additional cost to use WebJobs.

**Types:**

* **Continuous**
* **Triggered**

| **Continuous** | **Triggered** |
| --- | --- |
| Starts immediately when the WebJob is created. To keep the job from ending, the program or script typically does its work inside an endless loop. If the job does end, you can restart it. | Starts only when triggered manually or on a schedule. |
| Runs on all instances that the web app runs on. You can optionally restrict the WebJob to a single instance. | Runs on a single instance that Azure selects for load balancing. |
| Supports remote debugging. | Doesn't support remote debugging. |

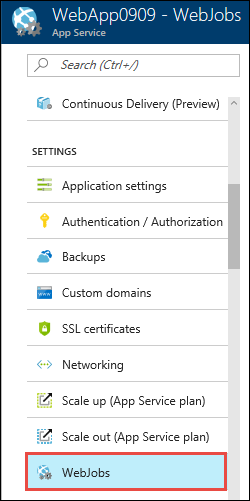
**Supported file types for scripts or programs**

The following file types are supported:

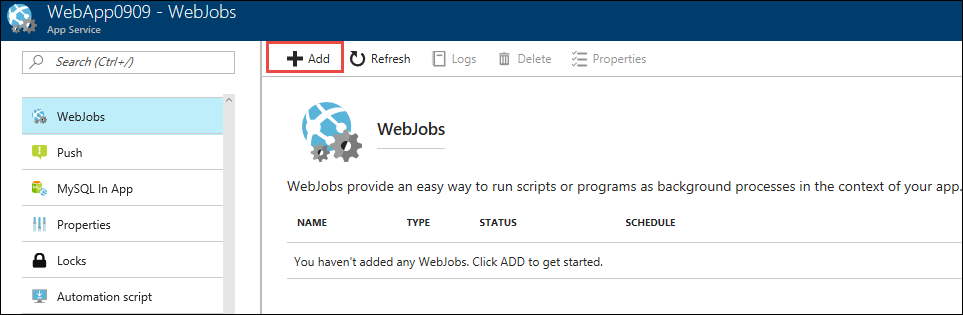
* .cmd, .bat, .exe (using Windows cmd)
* .ps1 (using PowerShell)
* .sh (using Bash)
* .php (using PHP)
* .py (using Python)
* .js (using Node.js)
* .jar (using Java)

In the [Azure portal](https://portal.azure.com/), go to the **App Service** page of your App Service web app, API app, or mobile app.

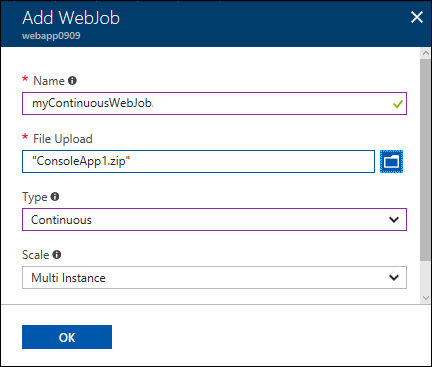
1. Select **WebJobs**.



1. In the **WebJobs** page, select **Add**.



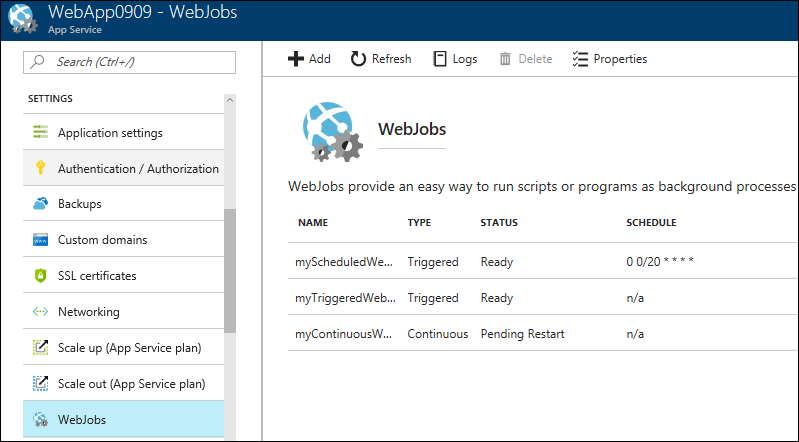
1. Use the **Add WebJob** settings as specified in the table.



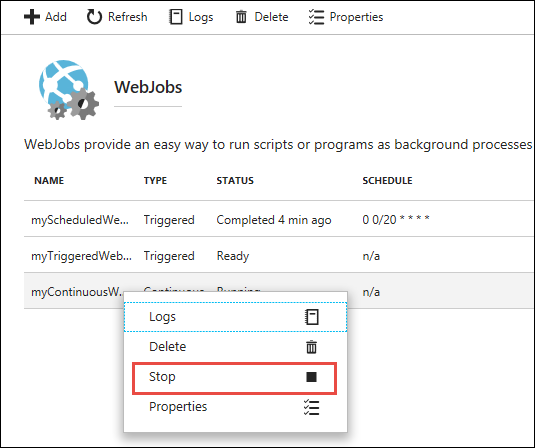
| **Setting** | **Sample value** | **Description** |
| --- | --- | --- |
| **Name** | myContinuousWebJob | A name that is unique within an App Service app. Must start with a letter or a number and cannot contain special characters other than "-" and "\_". |
| **File Upload** | ConsoleApp.zip | A .zip file that contains your executable or script file as well as any supporting files needed to run the program or script. The supported executable or script file types are listed in the [Supported file types](https://docs.microsoft.com/en-us/azure/app-service/webjobs-create#acceptablefiles) section. |
| **Type** | Continuous | The [WebJob types](https://docs.microsoft.com/en-us/azure/app-service/webjobs-create" \l "webjob-types) are described earlier in this article. |
| **Scale** | Multi instance | Available only for Continuous WebJobs. Determines whether the program or script runs on all instances or just one instance. The option to run on multiple instances doesn't apply to the Free or Shared [pricing tiers](https://azure.microsoft.com/pricing/details/app-service/?ref=microsoft.com&utm_source=microsoft.com&utm_medium=docs&utm_campaign=visualstudio). |

1. Click **OK**.

The new WebJob appears on the **WebJobs** page.

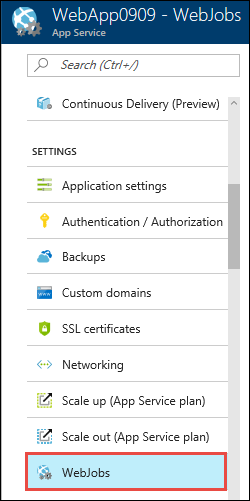


1. To stop or restart a continuous WebJob, right-click the WebJob in the list and click **Stop** or **Start**.

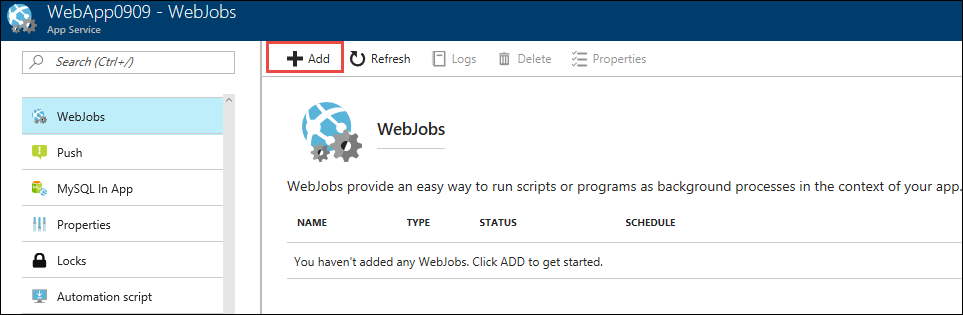


## Create a manually triggered WebJob

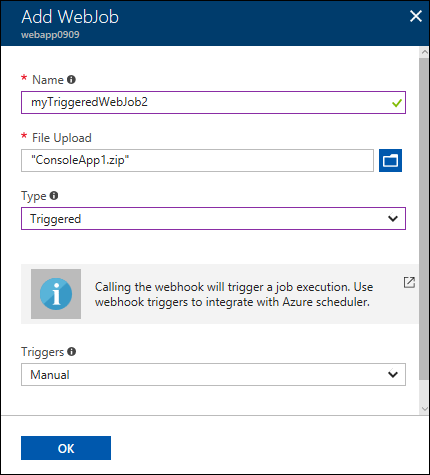
1. In the [Azure portal](https://portal.azure.com/), go to the **App Service** page of your App Service web app, API app, or mobile app.
2. Select **WebJobs**.



1. In the **WebJobs** page, select **Add**.

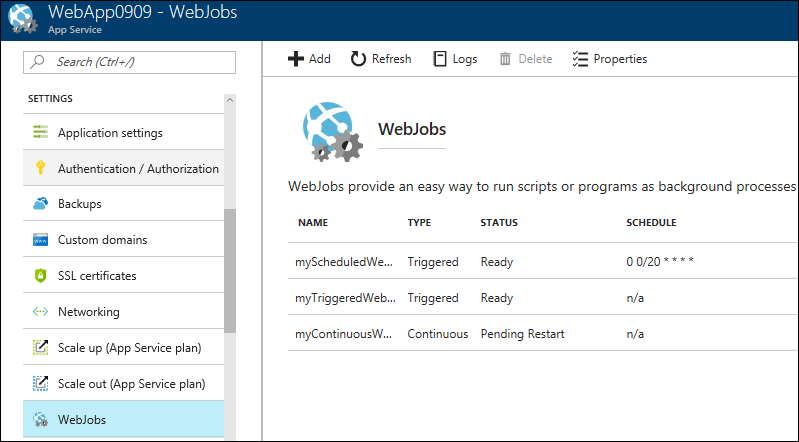


1. Use the **Add WebJob** settings as specified in the table.

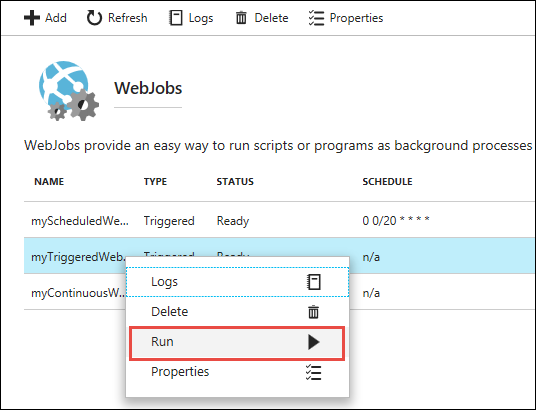


|  | | |
| --- | --- | --- |
| **Setting** | **Sample value** | **Description** |
| **Name** | myTriggeredWebJob | A name that is unique within an App Service app. Must start with a letter or a number and cannot contain special characters other than "-" and "\_". |
| **File Upload** | ConsoleApp.zip | A .zip file that contains your executable or script file as well as any supporting files needed to run the program or script. The supported executable or script file types are listed in the [Supported file types](https://docs.microsoft.com/en-us/azure/app-service/webjobs-create#acceptablefiles) section. |
| **Type** | Triggered | The [WebJob types](https://docs.microsoft.com/en-us/azure/app-service/webjobs-create" \l "webjob-types) are described earlier in this article. |
| **Triggers** | Manual |  |

1. Click **OK**.
2. The new WebJob appears on the **WebJobs** page.

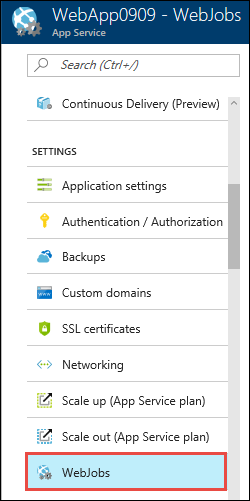


1. To run the WebJob, right-click its name in the list and click **Run**.

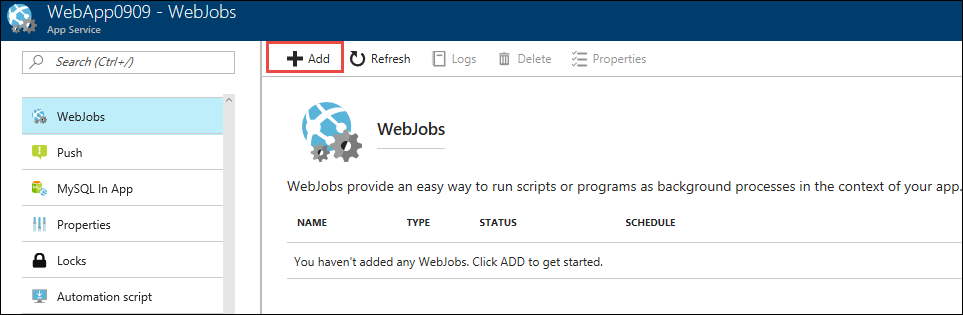


## Create a scheduled WebJob

1. In the [Azure portal](https://portal.azure.com/), go to the **App Service** page of your App Service web app, API app, or mobile app.
2. Select **WebJobs**.

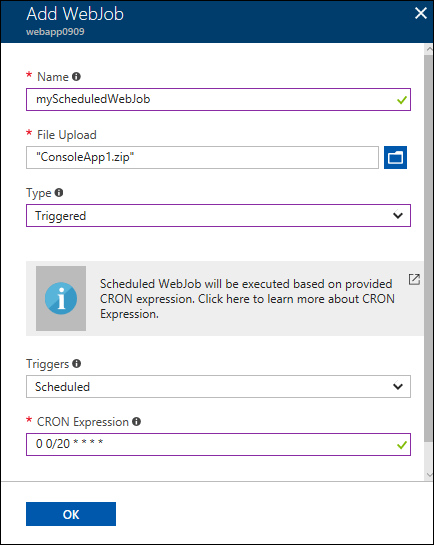


1. In the **WebJobs** page, select **Add**.



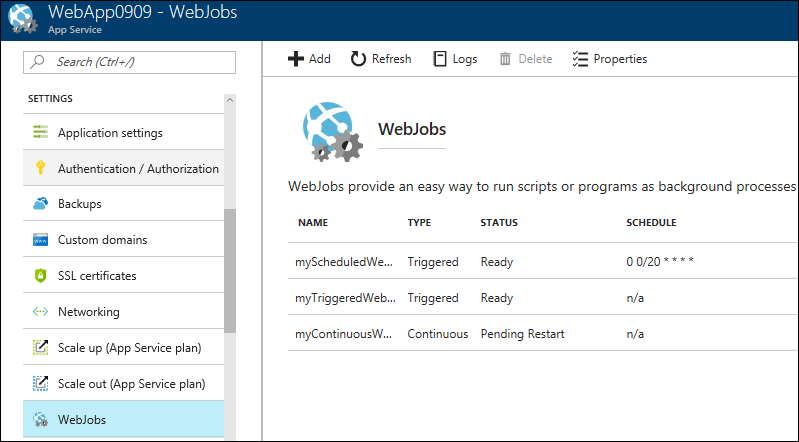
1. Use the **Add WebJob** settings as specified in the table.

|  | | |
| --- | --- | --- |
| **Setting** | **Sample value** | **Description** |
| **Name** | myScheduledWebJob | A name that is unique within an App Service app. Must start with a letter or a number and cannot contain special characters other than "-" and "\_". |
| **File Upload** | ConsoleApp.zip | A .zip file that contains your executable or script file as well as any supporting files needed to run the program or script. The supported executable or script file types are listed in the [Supported file types](https://docs.microsoft.com/en-us/azure/app-service/webjobs-create#acceptablefiles) section. |
| **Type** | Triggered | The [WebJob types](https://docs.microsoft.com/en-us/azure/app-service/webjobs-create" \l "webjob-types) are described earlier in this article. |
| **Triggers** | Scheduled | For the scheduling to work reliably, enable the Always On feature. Always On is available only in the Basic, Standard, and Premium pricing tiers. |
| **CRON Expression** | 0 0/20 \* \* \* \* | [CRON expressions](https://docs.microsoft.com/en-us/azure/app-service/webjobs-create#ncrontab-expressions) are described in the following section. |



1. Click **OK**.

The new WebJob appears on the **WebJobs** page.



## NCRONTAB expressions

You can enter a [NCRONTAB expression](https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-timer#ncrontab-expressions) in the portal or include a settings.job file at the root of your WebJob .zip file, as in the following example:

JSONCopy

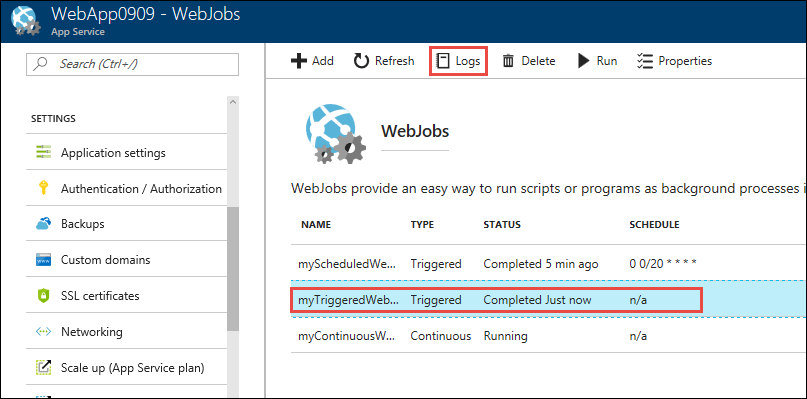
{

"schedule": "0 \*/15 \* \* \* \*"

}

## View the job history

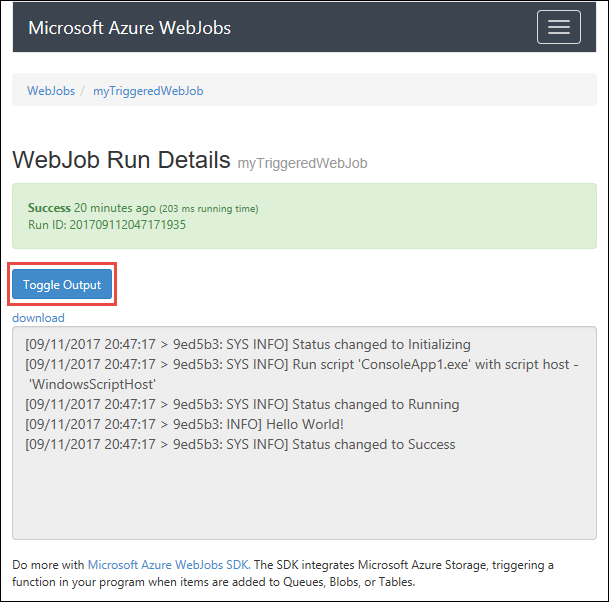
1. Select the WebJob you want to see history for, and then select the **Logs** button.



1. In the **WebJob Details** page, select a time to see details for one run.



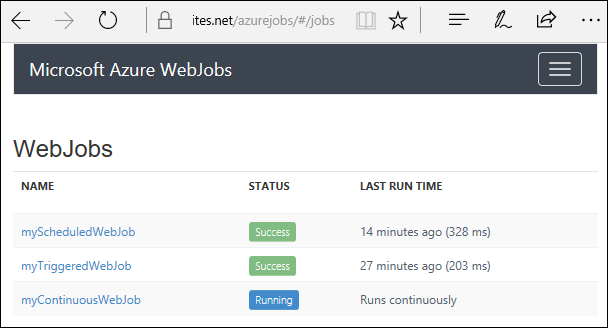
1. In the **WebJob Run Details** page, select **Toggle Output** to see the text of the log contents.



To see the output text in a separate browser window, select **download**. To download the text itself, right-click **download** and use your browser options to save the file contents.

1. Select the **WebJobs** breadcrumb link at the top of the page to go to a list of WebJobs.





**Azure API Apps and API Management**

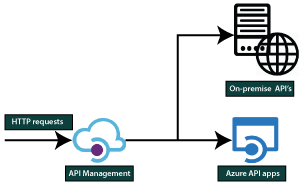
The API apps features make it easy to develop, host, and consume APIs in the cloud and on-premises. The advantage of hosting APIs in Azure API apps is that we will get enterprise-grade security and simple access control, automatic SDK generation, and seamless integration with Logic Apps. Logic Apps are system workflows that you can build within Azure. And as a part of the workflow, each activity needs to interact with the functionality exposed by a different system. By having those interfaces hosted in Azure, it makes it easy to integrate with the logic apps also.

## Features of API apps

* **Bring our own existing API as-is:** API can be developed in any language framework supported by App Service such as C#, Java, PHP, Node.js, etc.
* **Easy Consumption:** There is integrated support for Swagger API. By enabling swagger, we are making it easy for others to consume our APIs, and also we will provide excellent visibility of APIs to developers.
* **Simple access control:** Protect an API app from unauthenticated access with no changes to your code.
* **Visual Studio Integration**
* **Integration with Logic apps**

## API Management

* API Management is all about managing APIs. We can put an API Management frontend on an API to monitor and throttle usage, manipulate input and output, consolidate several APIs into one endpoint, and so forth. The APIs being managed will be hosted anywhere.
* API Apps is about hosting APIs, whereas API management is about managing APIs. Let?s see how API management works.



At a very high level, firstly, when http or https request comes. It will come to API management and the API management based on the location of the API. Then it will forward that request to either Azure API apps or on-premises apps. But when it is forwarding that request it can throttle, it can also monitor and manipulate the inputs and outputs.

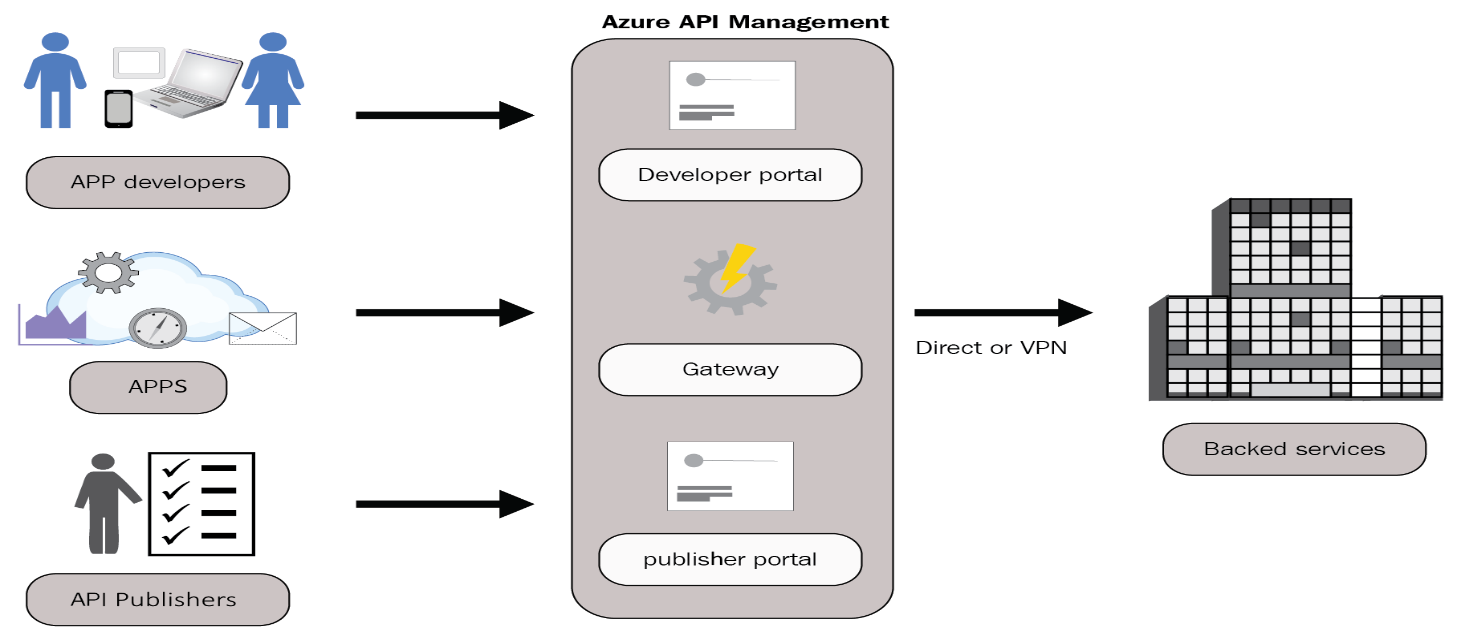
### API Management portals

* The API management portal is where developers can learn about APIs, view and call operations, and subscribe to products.
* Content within the developer portal is modified via the publisher portal, which is accessible from the Azure portal. To reach there, click on the Publisher portal from the service toolbar of our API Management instance.
* The dashboard of the developer portal can be customized by adding custom content, customizing styles, and adding our branding.

### API management concepts

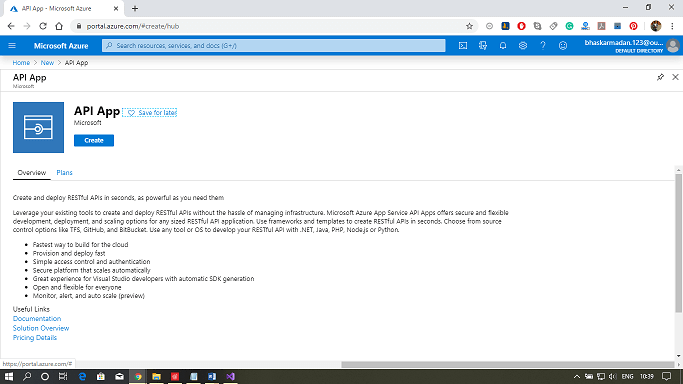
The API management concept is the crucial thing that we need to remember.

* **APIs and operations:** Each API represents a set of actions available (might be CRUD operation) to developers.
* **Products:** This is how APIs are surfaced to developers. Each product can contain multiple APIs.
* **Groups:** It is used to manage the visibility of APIs so we can have three types of groups.
  + An administrator group member can manage API management service instances, creating the APIs, operations, and products that are used by developers.
  + Developers? group members are authenticated customers that build applications using APIs.
  + Guests are the unauthenticated developer portal users. Guests are our prospective customers who will come and consume/trail. They will view the APIs and see whether it fits into their requirements or not.
* **Policies:** It is a very powerful capability of API management that allows the publisher to change the behavior of the API through configuration, such as throughput.

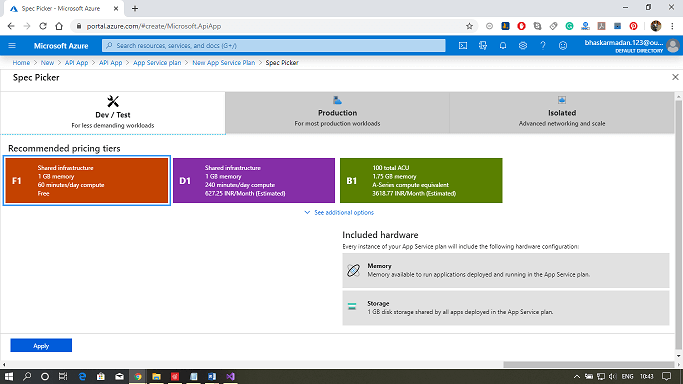


## Creating an API using Azure Portal

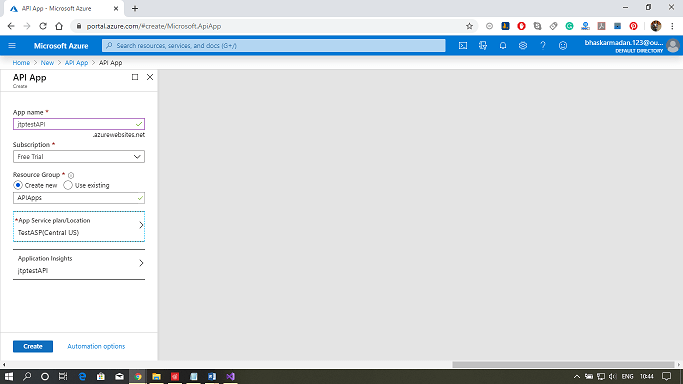
**Step 1:** Click on create a resource. After that type in API apps and click on create.



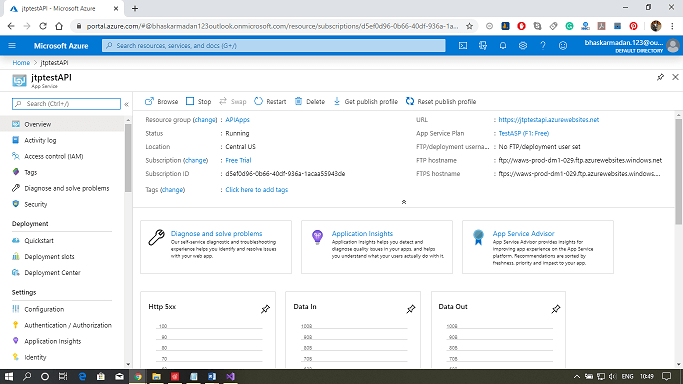
**Step 2:** Now, assign a name to your API app and select the resource group. After that, select the service architecture according to your requirements.



**Step 3:** Finally, click on create.

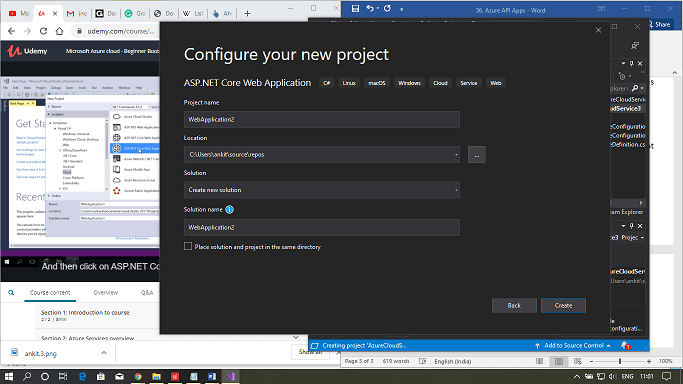


**Step 4:** Your API app will be successfully created.

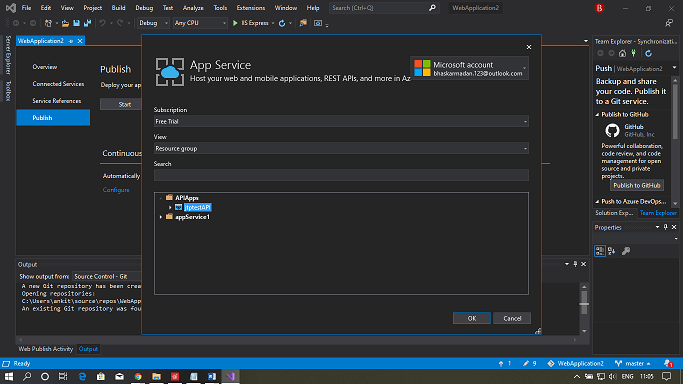


## Publishing an API using Visual Studio

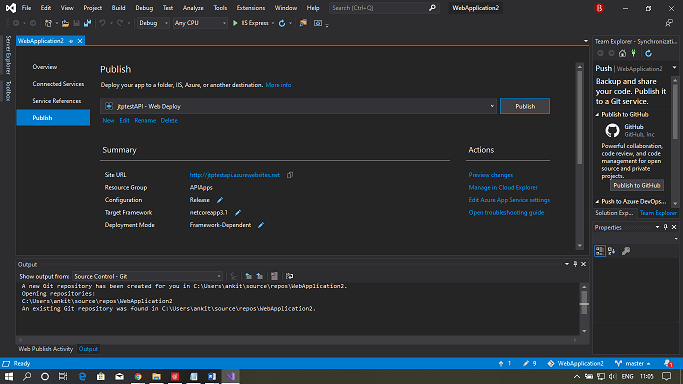
**Step 1:** Create a new web app project in Visual Studio. As shown in the following figure.


**Step 2:** Click on publish, then click on select existing. After that, click on the publish.



**Step 3:** Your web app is successfully posted to the Azure portal.



**Scaling**

There are two ways we can scale the app services.

**Scale-up:** It means we can get more CPU, memory, disc space, and also an extra feature like dedicated virtual machines, custom domains, certificates, staging slots, auto-scaling, and many other features based on the pricing tier we select when we are scaling up our app service plan.

**Scale-out:** It means we will increase the number of VM instances that run our app so we can scale out to any number of instances based on the pricing tier. But, if we go for app service environments in an isolated tier, then we can scale out to a hundred instances.

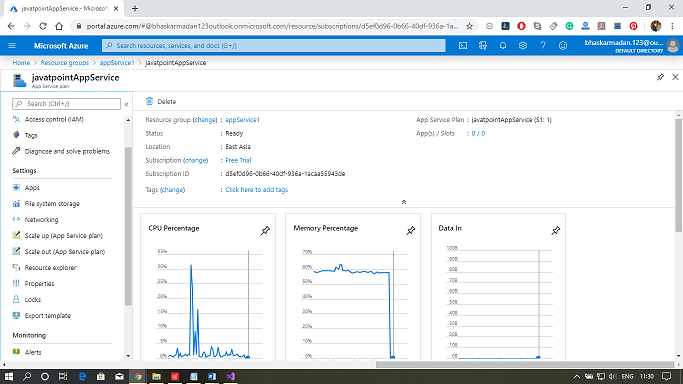
Apart from this, another important thing that we need to remember about scaling is Auto Scaling. There are many ways that we can scale our app services.

* Automatically
* Manually
* Pre-set Matric
* Scheduled

**Scaling the App Service using Azure Portal**

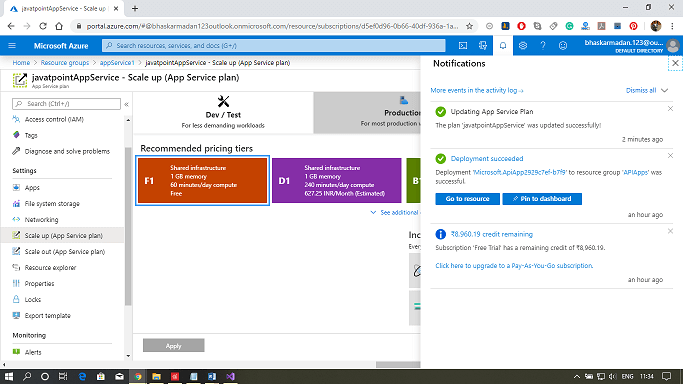
on some predefined conditions.

**Step 1:** Open your already created app service or create a new one.



**Step 2:** Now, click on scale-up on the left toolbar.

**Step 3:** Select the pricing tier then click on apply.



**Step 4:** It is successfully deployed, and similarly, you can scale out and set up it for auto-scaling based on some predefined conditions.

**Back up your app in Azure**

The Backup and Restore feature in [Azure App Service](https://docs.microsoft.com/en-us/azure/app-service/overview) lets you easily create app backups manually or on a schedule. You can configure the backups to be retained up to an indefinite amount of time. You can restore the app to a snapshot of a previous state by overwriting the existing app or restoring to another app.

**What gets backed up**

App Service can back up the following information to an Azure storage account and container that you have configured your app to use.

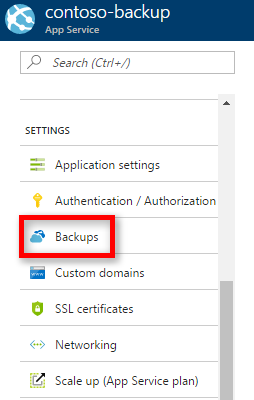
* App configuration
* File content
* Database connected to your app

## Requirements and restrictions

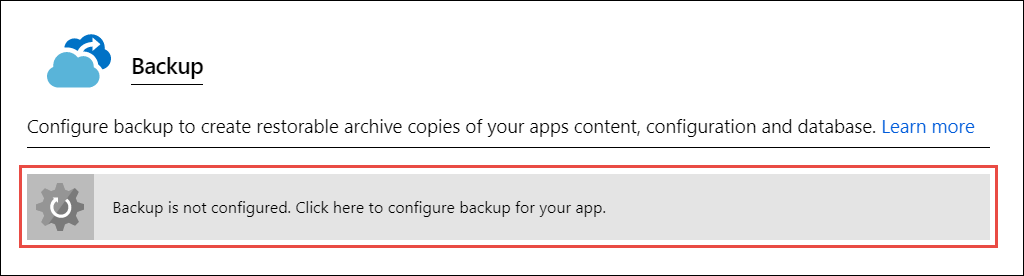
* The Backup and Restore feature requires the App Service plan to be in the **Standard**, **Premium** or **Isolated** tier. For more information about scaling your App Service plan to use a higher tier, see [Scale up an app in Azure](https://docs.microsoft.com/en-us/azure/app-service/manage-scale-up). **Premium** and **Isolated** tiers allow a greater number of daily back ups than **Standard** tier.
* You need an Azure storage account and container in the same subscription as the app that you want to back up. For more information on Azure storage accounts, see [Azure storage account overview](https://docs.microsoft.com/en-us/azure/storage/common/storage-account-overview).
* Backups can be up to 10 GB of app and database content. If the backup size exceeds this limit, you get an error.
* Backups of TLS enabled Azure Database for MySQL is not supported. If a backup is configured, you will encounter backup failures.
* Backups of TLS enabled Azure Database for PostgreSQL is not supported. If a backup is configured, you will encounter backup failures.
* In-app MySQL databases are automatically backed up without any configuration. If you make manually settings for in-app MySQL databases, such as adding connection strings, the backups may not work correctly.
* Using a firewall enabled storage account as the destination for your backups is not supported. If a backup is configured, you will encounter backup failures.

## Create a manual backup

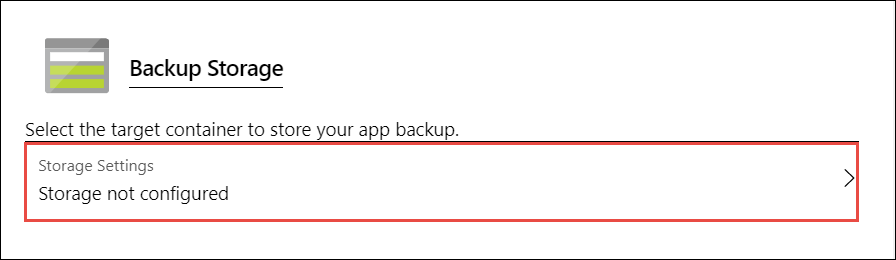
1. In the [Azure portal](https://portal.azure.com/), navigate to your app's page, select **Backups**. The **Backups** page is displayed.



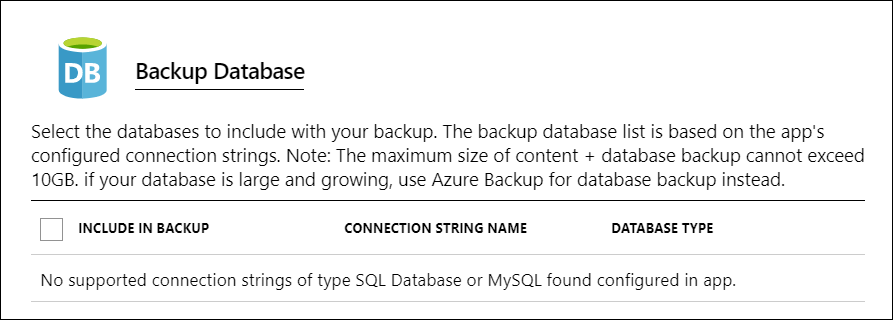
1. In the **Backup** page, select **Backup is not configured. Click here to configure backup for your app**.



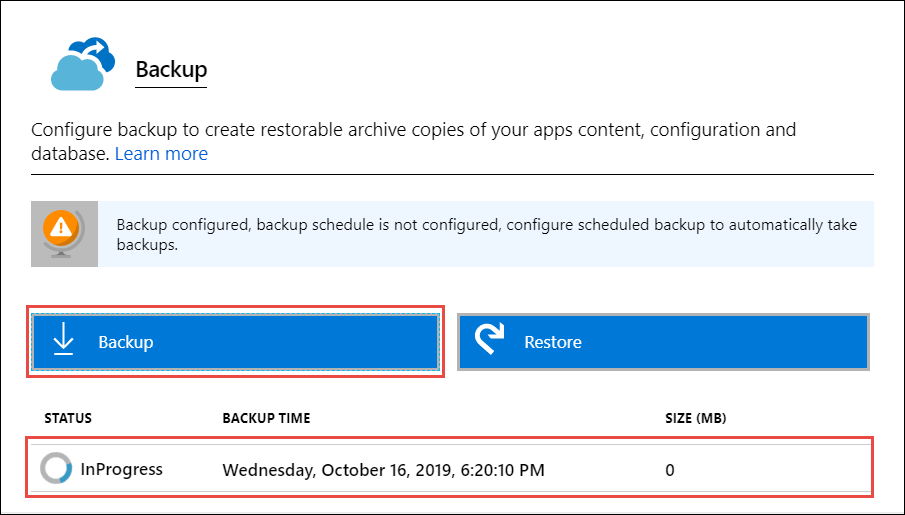
1. In the **Backup Configuration** page, click **Storage not configured** to configure a storage account.



1. Choose your backup destination by selecting a **Storage Account** and **Container**. The storage account must belong to the same subscription as the app you want to back up. If you wish, you can create a new storage account or a new container in the respective pages. When you're done, click **Select**.
2. In the **Backup Configuration** page that is still left open, you can configure **Backup Database**, then select the databases you want to include in the backups (SQL Database or MySQL), then click **OK**.



1. In the **Backup Configuration** page, click **Save**.
2. In the **Backups** page, click **Backup**.

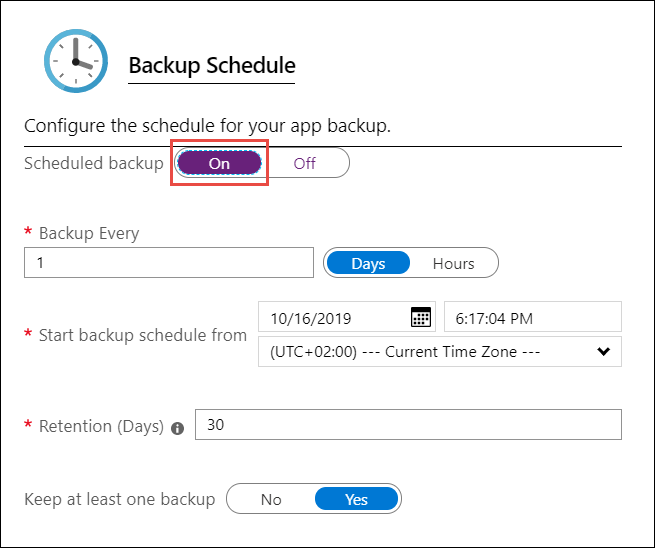


You see a progress message during the backup process.

Once the storage account and container is configured, you can initiate a manual backup at any time. Manual backups are retained indefinitely.

## Configure automated backups

1. In the **Backup Configuration** page, set **Scheduled backup** to **On**.



1. Configure the backup schedule as desired and select **OK**.

## 

## Configure Partial Backups

Sometimes you don't want to back up everything on your app. Here are a few examples:

* You [set up weekly backups](https://docs.microsoft.com/en-us/azure/app-service/manage-backup#configure-automated-backups) of your app that contains static content that never changes, such as old blog posts or images.
* Your app has over 10 GB of content (that's the max amount you can back up at a time).
* You don't want to back up the log files.

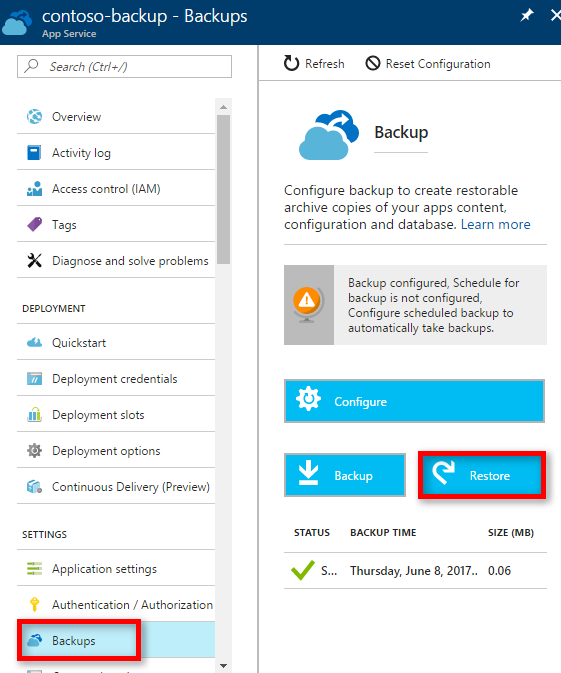
Partial backups allow you choose exactly which files you want to back up.

# **Restore an app in Azure**

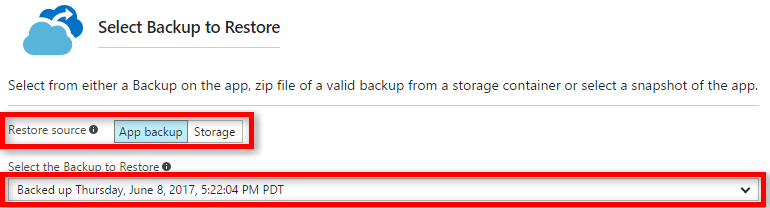
This article shows you how to restore an app in [Azure App Service](https://docs.microsoft.com/en-us/azure/app-service/overview) that you have previously backed up (see [Back up your app in Azure](https://docs.microsoft.com/en-us/azure/app-service/manage-backup)). You can restore your app with its linked databases on-demand to a previous state, or create a new app based on one of your original app's backups. Azure App Service supports the following databases for backup and restore

**Restore an app from an existing backup**

1. On the **Settings** page of your app in the Azure portal, click **Backups** to display the **Backups** page. Then click **Restore**.

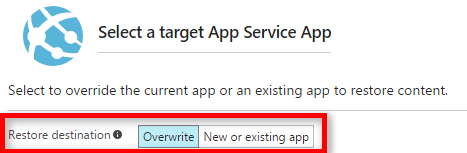


1. In the **Restore** page, first select the backup source.



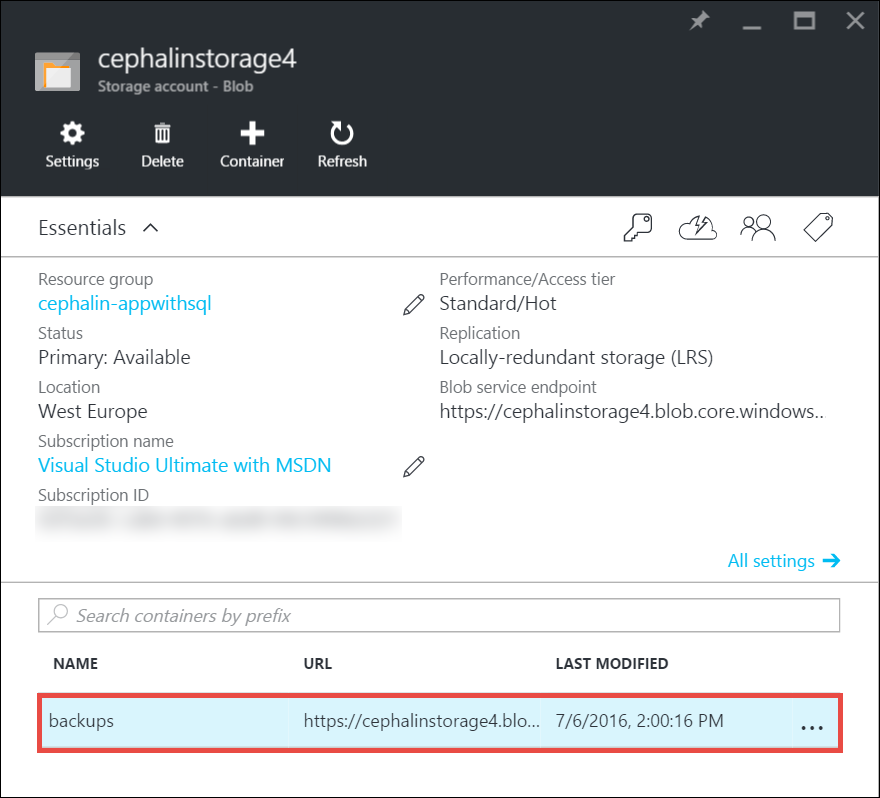
The **App backup** option shows you all the existing backups of the current app, and you can easily select one. The **Storage** option lets you select any backup ZIP file from any existing Azure Storage account and container in your subscription. If you're trying to restore a backup of another app, use the **Storage** option.

1. Then, specify the destination for the app restore in **Restore destination**.



**Download or delete a backup from a storage account**

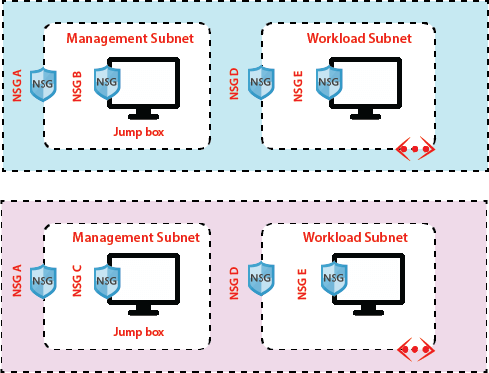
1. From the main **Browse** page of the Azure portal, select **Storage accounts**. A list of your existing storage accounts is displayed.
2. Select the storage account that contains the backup that you want to download or delete. The page for the storage account is displayed.
3. In the storage account page, select the container you want



1. Select backup file you want to download or delete.
2. Click **Download** or **Delete** depending on what you want to do.

# **Azure Network Security group**

A network security group consists of security rules that allow or deny inbound/Outbound network traffic to or from different types of Azure resources that we will host in our Azure virtual network. And we can apply the network security group at different levels. For example:



## Security rule properties:

**Name:** The name of the network should be unique within the network security group.

**Priority:** Security rules are processed in priority order with a lower number has the highest priority.

**Source or Destination:** (The IP address, CIDR (Classless inter-domain routing) block, service tag, or application security group) The ability to specify multiple individual IP addresses and ranges in a rule is referred to as augmented security rules.

**Protocol:** TCP, UDP, etc.

**Port range:** we can specify an individual or range of ports

**Action:** Allow or Deny

**Service Tags**

Service tag represents a group of IP address prefixes to help minimize complexity for security rule creation. We cannot create our service tag, nor specify which Ip address is included within a tag. Microsoft manages the address prefixes encompassed by the service tag, and automatically updates the service tag as an address change.

Earlier, if we want to allow communication to Azure service from our virtual machine, we need to configure IoT of outbound rules because Microsoft is providing list of IP addresses for each service you need to configure those list of IP addresses in our NSG rule to allow outbound connection from our virtual machine to that particular service and also in case if Microsoft is changing the addresses you need to change your rules.

Using service tags will simplify your NSG rules a lot, for example:

**Storage:** This tag denotes the IP address space for the Azure Storage service. If you specify Storage for the value, traffic is allowed or denied to storage.

**SQL:** This tag denotes the address prefixes of the Azure SQL Database, Azure Database for MySQL, Azure Database for PostgreSQL, and Azure SQL Data Warehouse services.

**Azure CosmosDB:** This tag denotes the address prefixes of the Azure Cosmos Database services.

**AzureKeyVault:** This tag denotes the address prefixes of the Azure KeyVault service. If you specify AzureKeyVault for the value, traffic is allowed or denied to AzureKeyVault.

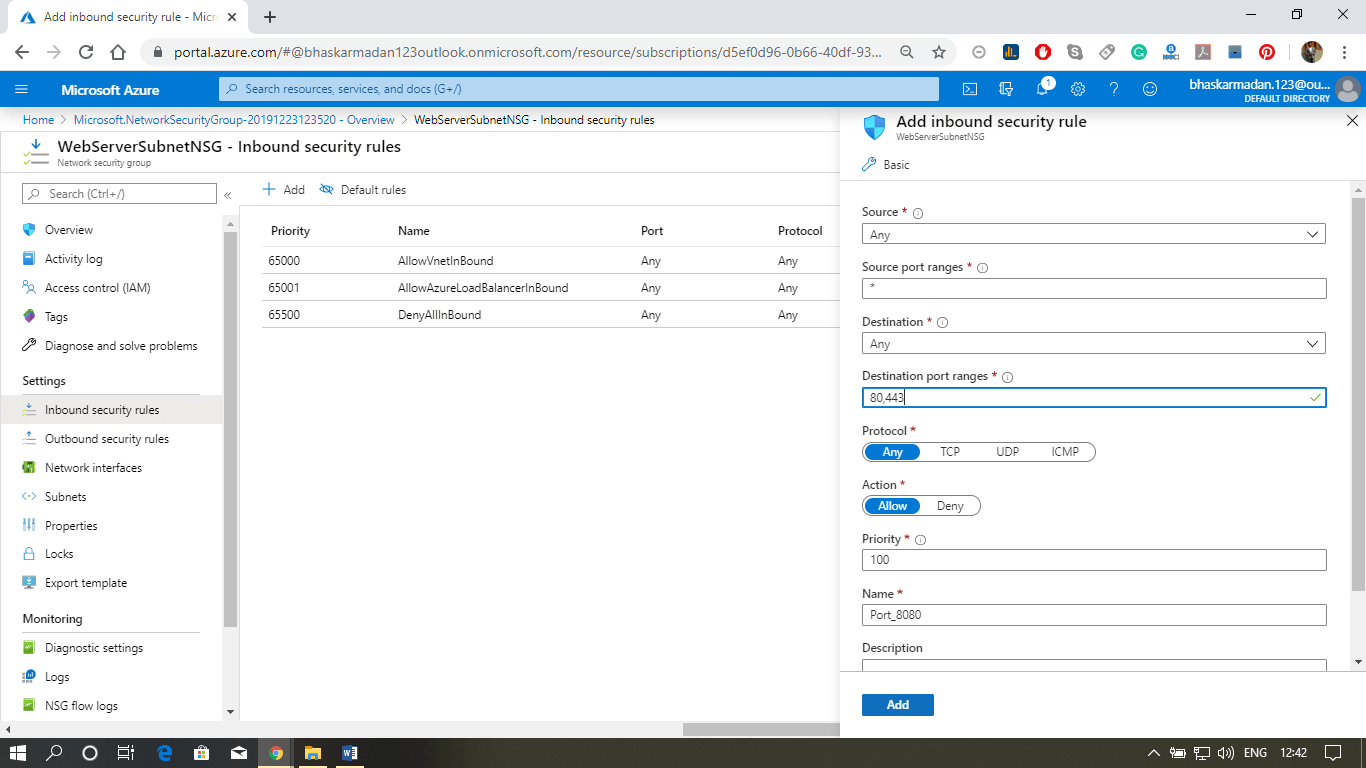
**EventHub:** This tag denotes the address prefixes of the Azure EventHub service. If you specify EventHub for the value, traffic is allowed or denied to EventHub.

## Default Rules

Some default rules are created by default when we create NSG. There are two types of default rules.

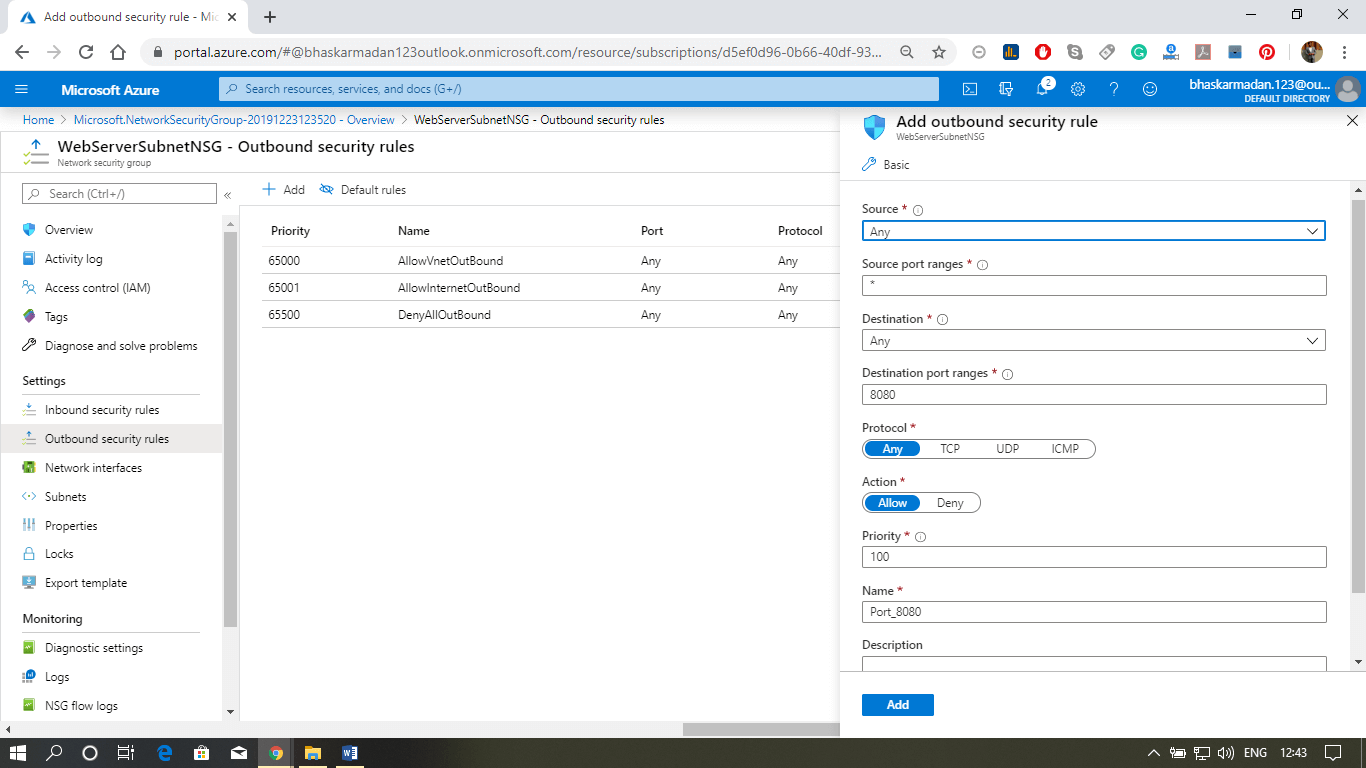
**Inbound Security rules**

* **AllowVNetInbound:** Traffic is allowed from any resources within the VNet
* **AllowAzureLoadBalancerInbound:** Any traffic originating from Azure load-balancer to any of the virtual machines within the network is permitted.
* **DenyAllInbound:** By default, virtual machines in the virtual network can communicate with each other, and also Azure load balancer can communicate with virtual machines within the virtual network
* load balancer can communicate with virtual machines within the virtual network.
* **Step 2:** Now, you are on the Network Security Group creation page. Select the resource group, fill the name, select the region, and click on review+create.
* load balancer can communicate with virtual machines within the virtual network.



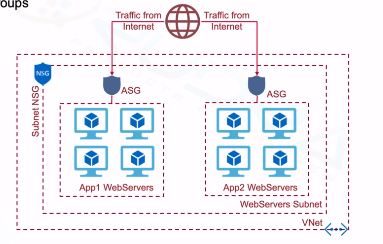
**Outbound Security rules**

* **AllowVNetOutBound:** Traffic is allowed through any resources within the VNet
* **AllowInternetOutBound:** Traffic originating from any resources in the VNet to the Internet is allowed.
* **DenyAllOutBound:** By default, virtual machines in a virtual network can communicate with each other, and also Azure load balancer can interact with the virtual machine within the virtual network.



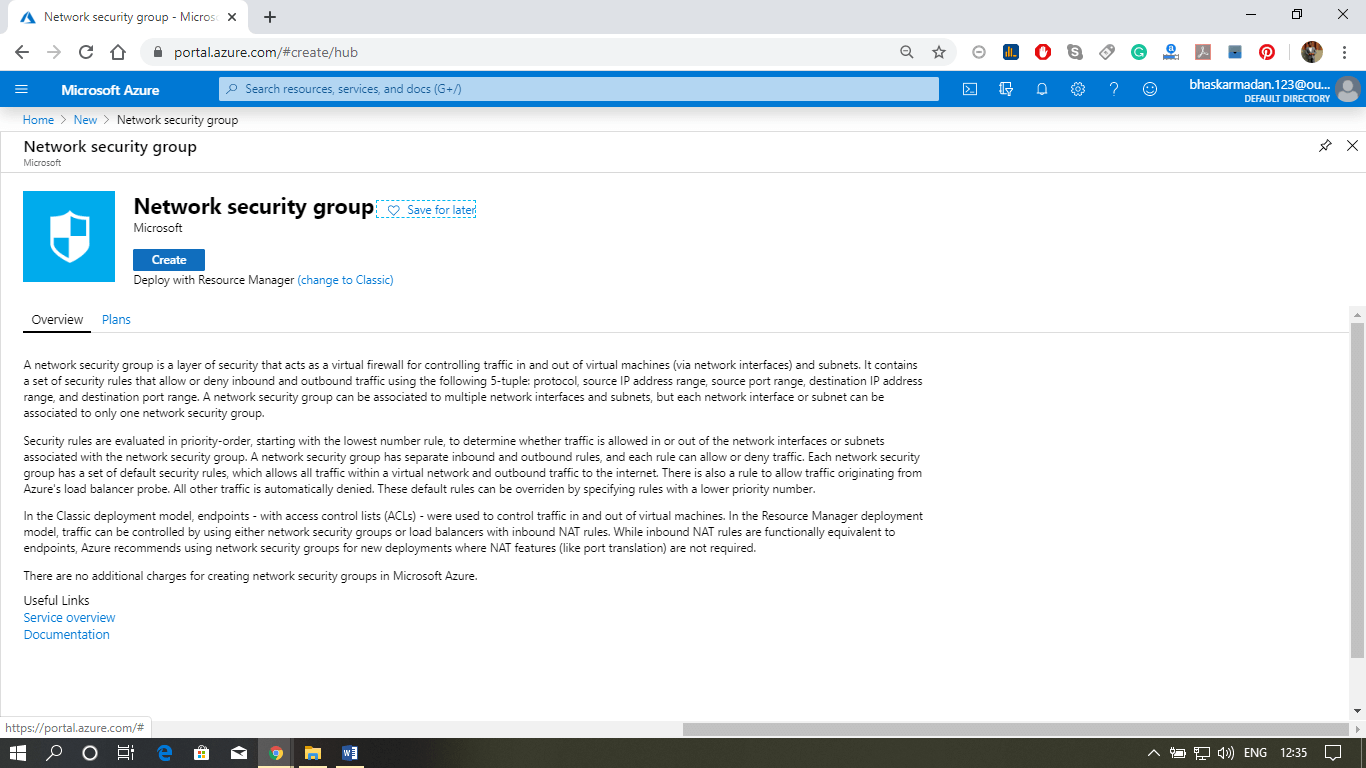
**Application Security Groups**

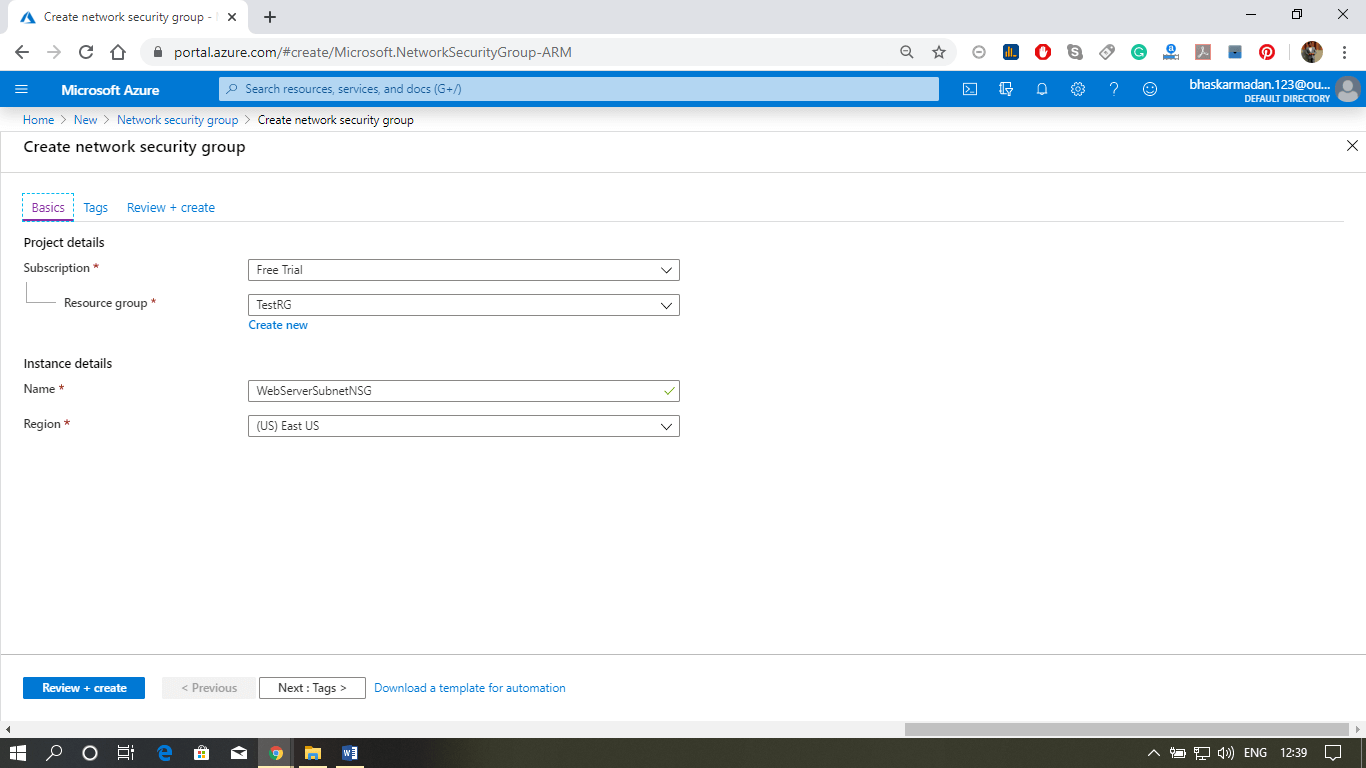
Application security groups enable you to configure network security as a natural extension of an application's structure, allowing you to group virtual machines and define network security policies based on those groups. For example -



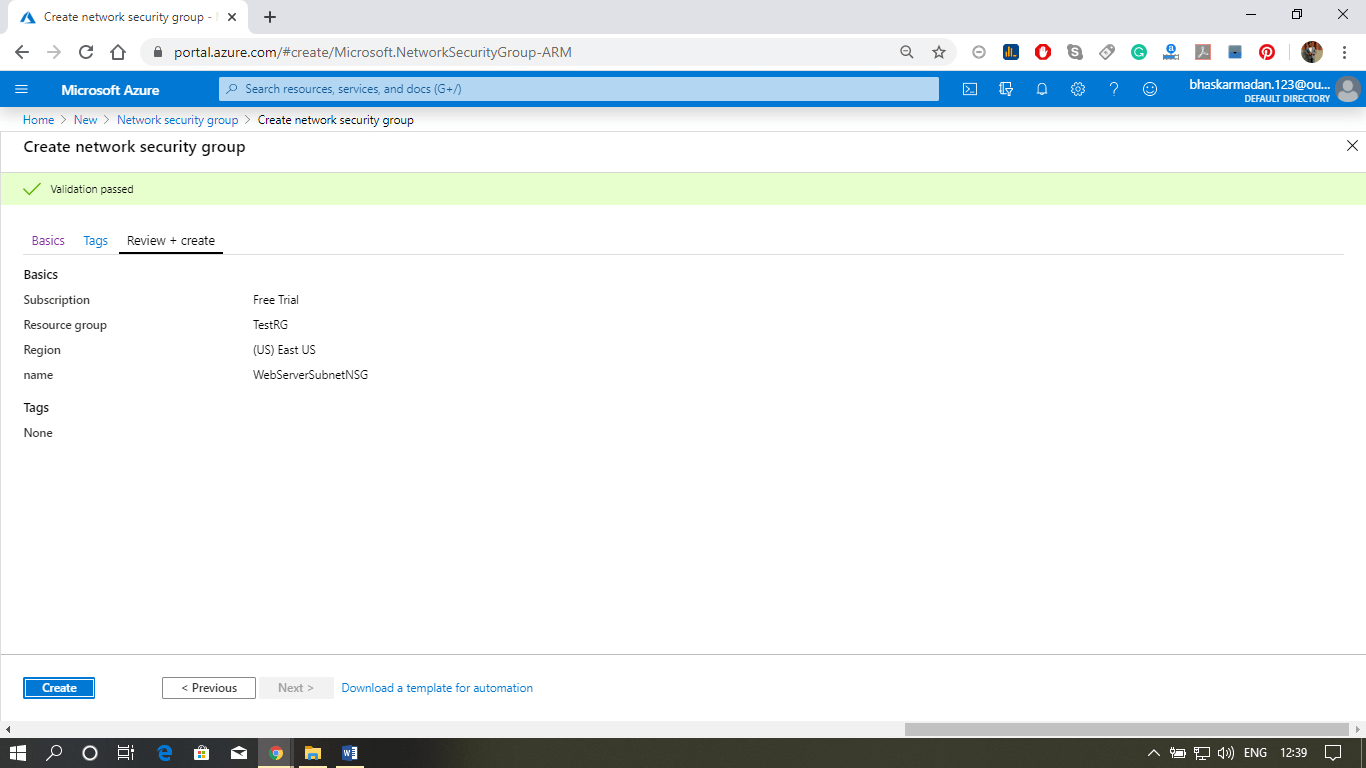
**Configuring an NSG at Subnet and VM level**

**Step 1:** Click on create a resource button and type-in Network Security Group. Then select Network Security Group, and click on create button.

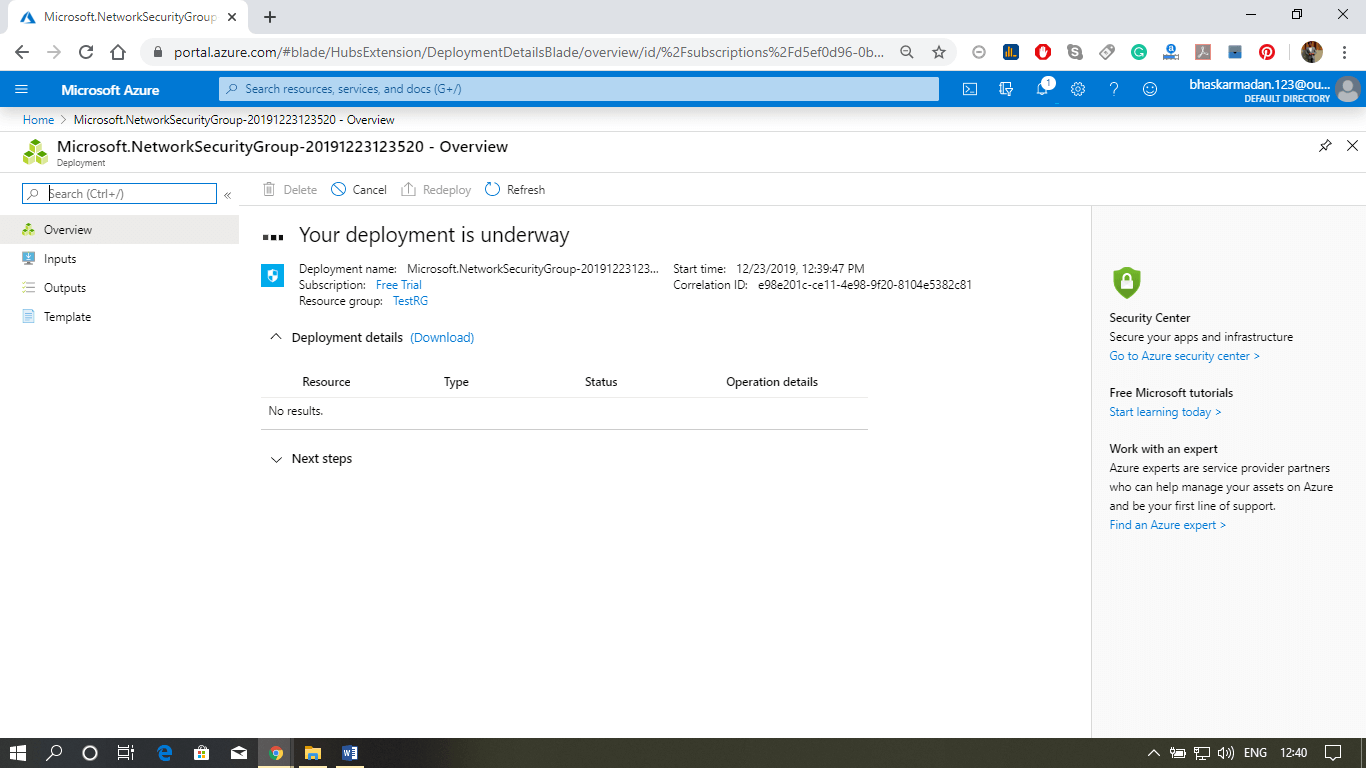




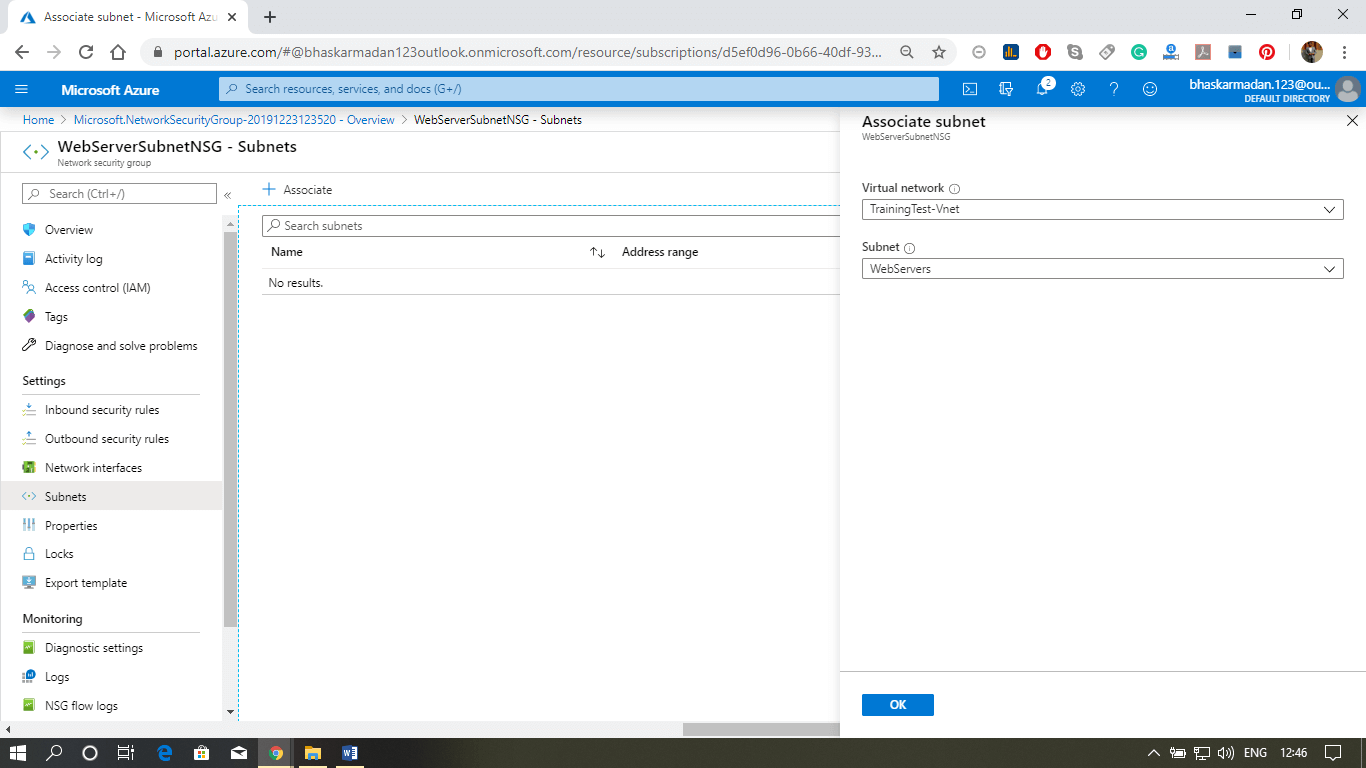
**Step 2:** Now, you are on the Network Security Group creation page. Select the resource group, fill the name, select the region, and click on review+create.



**Step 3:** Your NSG is created, now we will associate this NSG with the subnet.



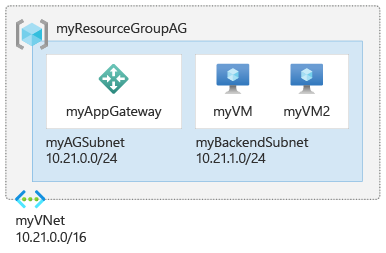
**Step 4:** Click on the subnet, then click on add Associate. Select the virtual network and subnet with which you want to associate this NSG.



**Step 5:** Finally, click on the ok button. Your NSG is now associated with the subnet.

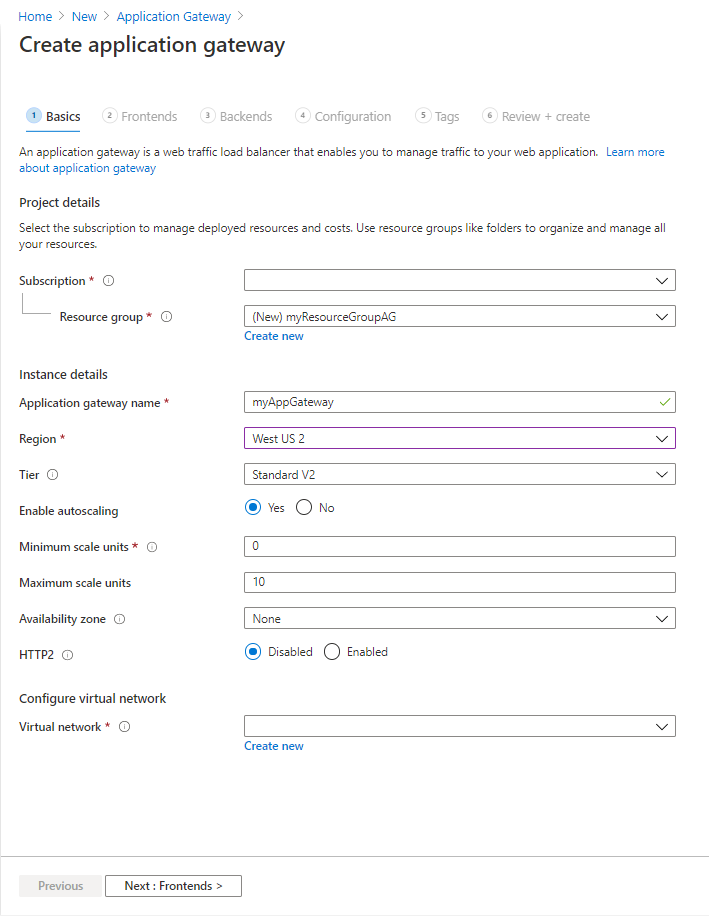
**Application Gateway :**

The application gateway directs application web traffic to specific resources in a backend pool. You assign listeners to ports, create rules, and add resources to a backend pool. For the sake of simplicity, this article uses a simple setup with a public front-end IP, a basic listener to host a single site on the application gateway, a basic request routing rule, and two virtual machines in the backend pool.



**Basics tab**

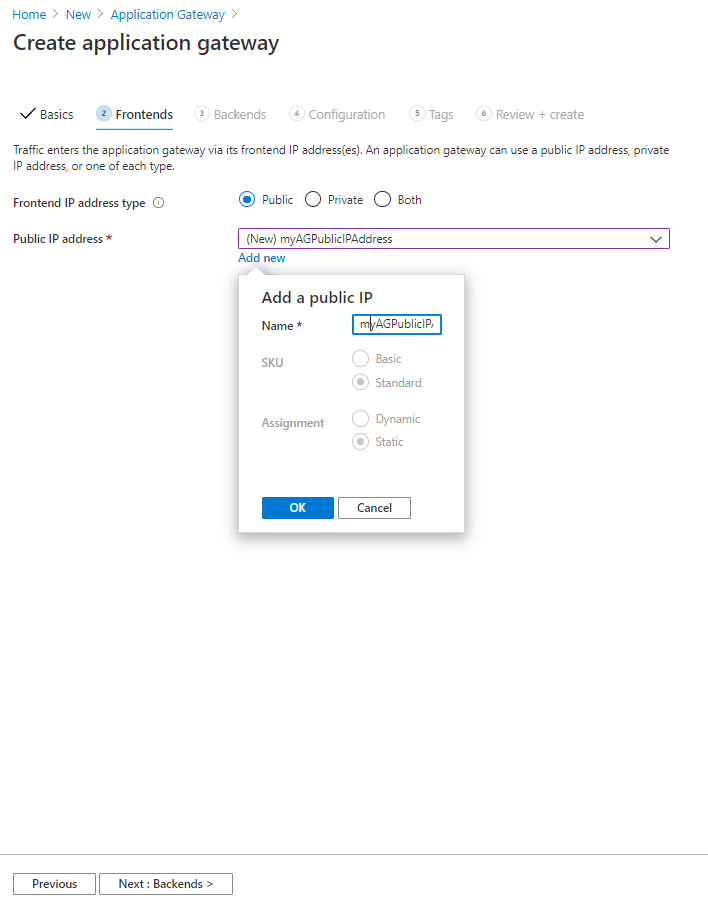
1. On the **Basics** tab, enter these values for the following application gateway settings:
   * **Resource group**: Select **myResourceGroupAG** for the resource group. If it doesn't exist, select **Create new** to create it.
   * **Application gateway name**: Enter *myAppGateway* for the name of the application gateway.



1. For Azure to communicate between the resources that you create, it needs a virtual network. You can either create a new virtual network or use an existing one. In this example, you'll create a new virtual network at the same time that you create the application gateway. Application Gateway instances are created in separate subnets. You create two subnets in this example: one for the application gateway, and another for the backend servers.
2. Under **Configure virtual network**, create a new virtual network by selecting **Create new**.
3. On the **Basics** tab, accept the default values for the other settings and then select **Next: Frontends**.

**Frontends tab**

1. On the **Frontends** tab, verify **Frontend IP address type** is set to **Public**.  
   You can configure the Frontend IP to be Public or Private as per your use case. In this example, you'll choose a Public Frontend IP.
2. Select **Add new** for the **Public IP address** and enter *myAGPublicIPAddress* for the public IP address name, and then select **OK**.

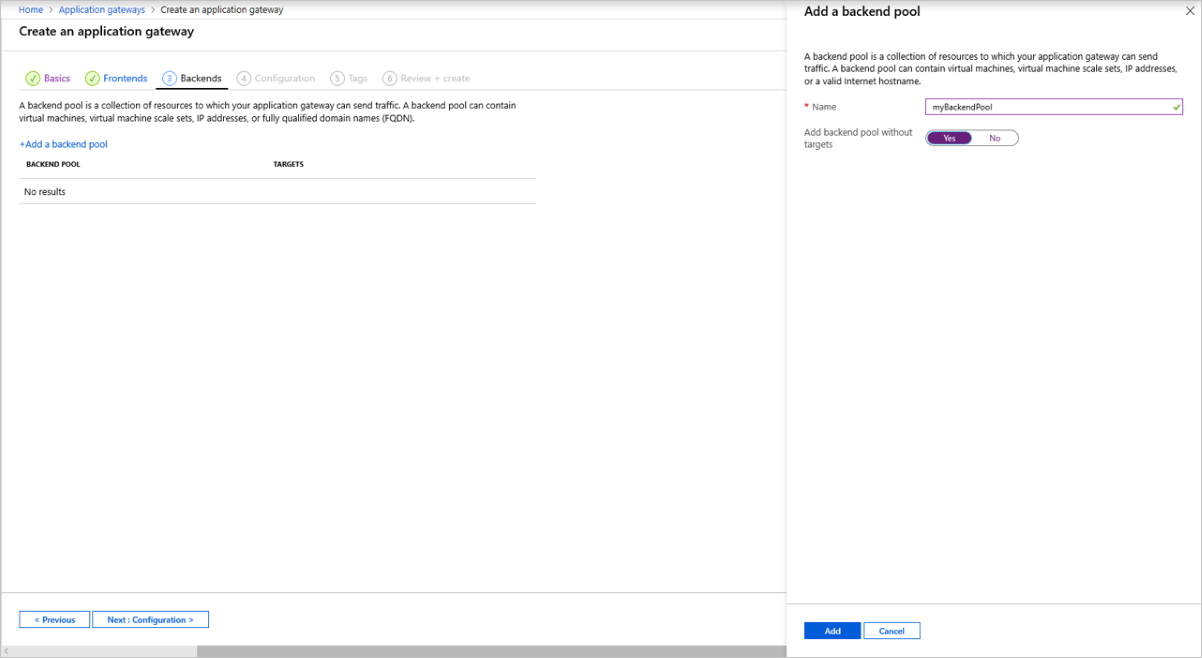


1. Select **Next: Backends**.

**Backends tab**

The backend pool is used to route requests to the backend servers that serve the request. Backend pools can be composed of NICs, virtual machine scale sets, public IP addresses, internal IP addresses, fully qualified domain names (FQDN), and multi-tenant back-ends like Azure App Service. In this example, you'll create an empty backend pool with your application gateway and then add backend targets to the backend pool.

1. On the **Backends** tab, select **Add a backend pool**.
2. In the **Add a backend pool** window that opens, enter the following values to create an empty backend pool:
   * **Name**: Enter *myBackendPool* for the name of the backend pool.
   * **Add backend pool without targets**: Select **Yes** to create a backend pool with no targets. You'll add backend targets after creating the application gateway.
3. In the **Add a backend pool** window, select **Add** to save the backend pool configuration and return to the **Backends** tab.



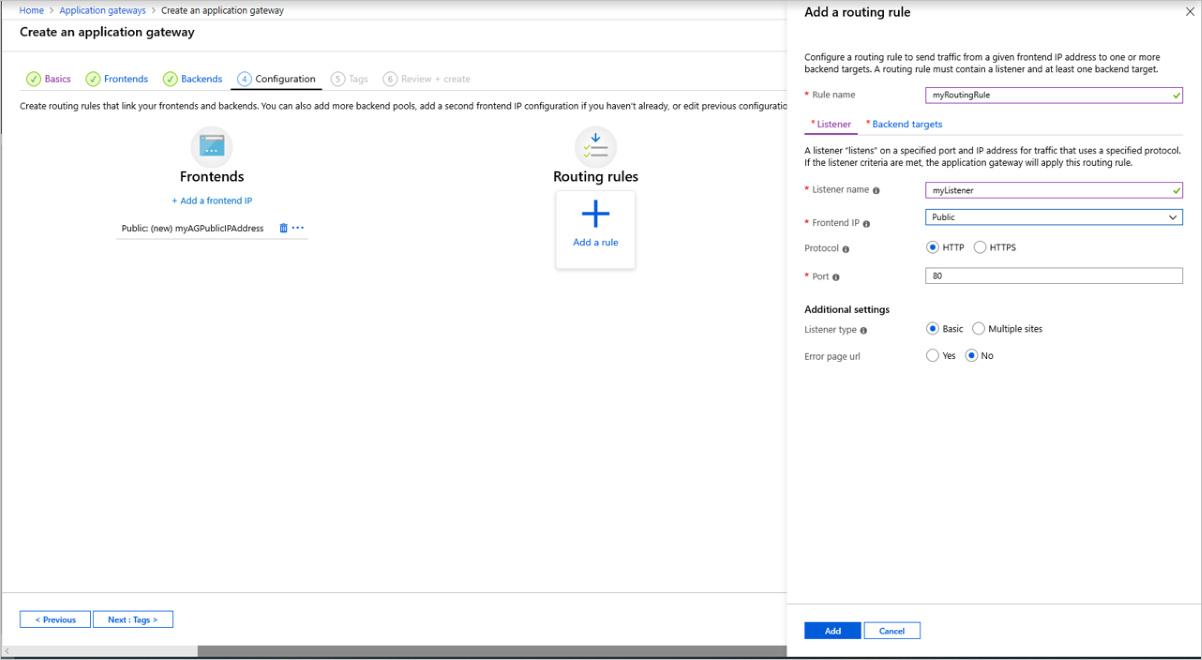
1. On the **Backends** tab, select **Next: Configuration**.

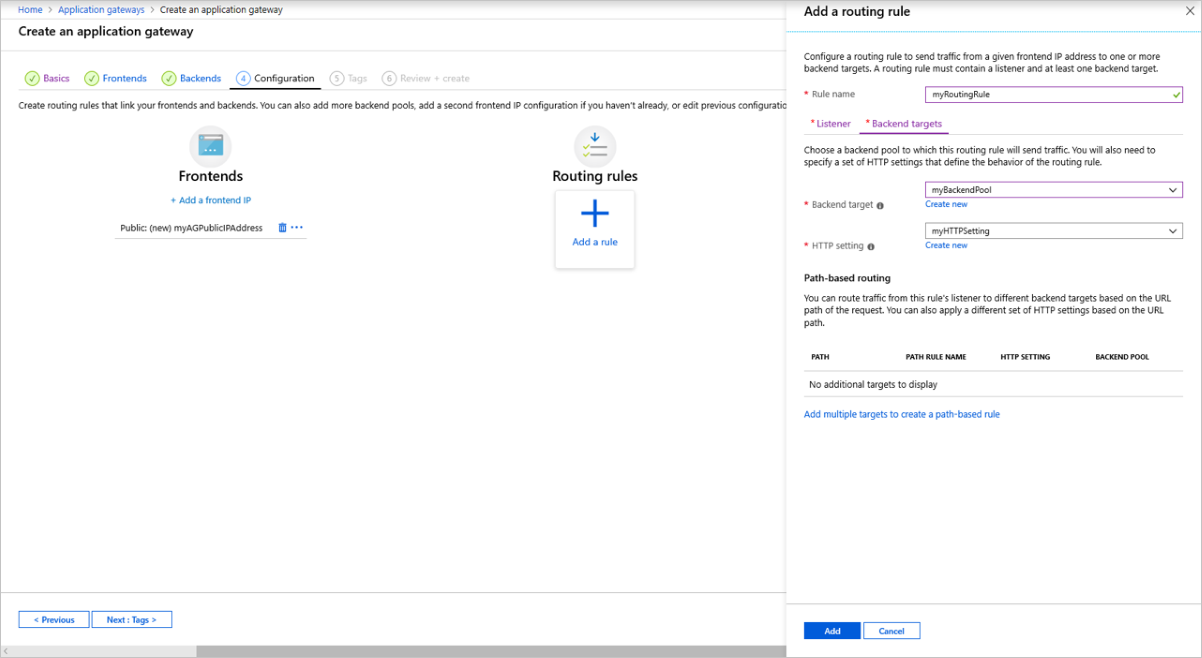
**Configuration tab**

On the **Configuration** tab, you'll connect the frontend and backend pool you created using a routing rule.

1. Select **Add a routing rule** in the **Routing rules** column.
2. In the **Add a routing rule** window that opens, enter *myRoutingRule* for the **Rule name**.
3. A routing rule requires a listener. On the **Listener** tab within the **Add a routing rule** window, enter the following values for the listener:
   * **Listener name**: Enter *myListener* for the name of the listener.
   * **Frontend IP**: Select **Public** to choose the public IP you created for the frontend.

Accept the default values for the other settings on the **Listener** tab, then select the **Backend targets** tab to configure the rest of the routing rule.



1. On the **Backend targets** tab, select **myBackendPool** for the **Backend target**.
2. For the **HTTP setting**, select **Add new** to add a new HTTP setting. The HTTP setting will determine the behavior of the routing rule. In the **Add an HTTP setting** window that opens, enter *myHTTPSetting* for the **HTTP setting name** and *80* for the **Backend port**. Accept the default values for the other settings in the **Add an HTTP setting** window, then select **Add** to return to the **Add a routing rule** window.
3. On the **Add a routing rule** window, select **Add** to save the routing rule and return to the **Configurati**
4. Select **Next: Tags** and then **Next: Review + create**.

**Review + create tab**

Review the settings on the **Review + create** tab, and then select **Create** to create the virtual network, the public IP address, and the application gateway. It may take several minutes for Azure to create the application gateway. Wait until the deployment finishes successfully before moving on to the next section.

**Add backend targets**

In this example, you'll use virtual machines as the target backend. You can either use existing virtual machines or create new ones. You'll create two virtual machines as backend servers for the application gateway.

To do this, you'll:

1. Create two new VMs, *myVM* and *myVM2*, to be used as backend servers.
2. Install IIS on the virtual machines to verify that the application gateway was created successfully.
3. Add the backend servers to the backend pool.

**Install IIS for testing**

In this example, you install IIS on the virtual machines to verify Azure created the application gateway successfully.

1. Open Azure PowerShell.

Select **Cloud Shell** from the top navigation bar of the Azure portal and then select **PowerShell** from the drop-down list.

1. Run the following command to install IIS on the virtual machine. Change the *Location* parameter if necessary:

Azure PowerShellCopy

Set-AzVMExtension `

-ResourceGroupName myResourceGroupAG `

-ExtensionName IIS `

-VMName myVM `

-Publisher Microsoft.Compute `

-ExtensionType CustomScriptExtension `

-TypeHandlerVersion 1.4 `

-SettingString '{"commandToExecute":"powershell Add-WindowsFeature Web-Server; powershell Add-Content -Path \"C:\\inetpub\\wwwroot\\Default.htm\" -Value $($env:computername)"}' `

-Location EastUS

1. Create a second virtual machine and install IIS by using the steps that you previously completed. Use *myVM2* for the virtual machine name and for the **VMName** setting of the **Set-AzVMExtension** cmdlet.

**Add backend servers to backend pool**

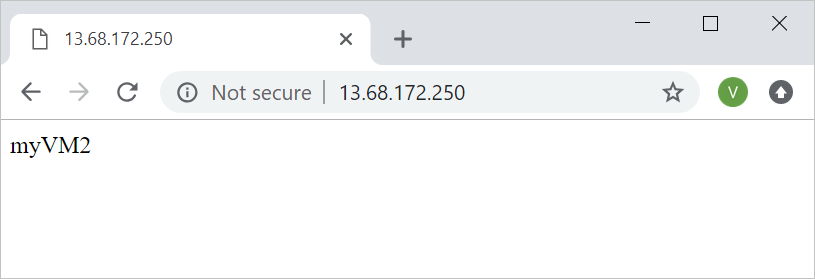
1. On the Azure portal menu, select **All resources** or search for and select *All resources*. Then select **myAppGateway**.
2. Select **Backend pools** from the left menu.
3. Select **myBackendPool**.
4. Under **Backend targets**, **Target type**, select **Virtual machine** from the drop-down list.
5. Under **Target**, select the **myVM** and **myVM2** virtual machines and their associated network interfaces from the drop-down lists.
6. Select **Save**.
7. Wait for the deployment to complete before proceeding to the next step.

**Test the application gateway**

Although IIS isn't required to create the application gateway, you installed it in this quickstart to verify if Azure successfully created the application gateway.

Use IIS to test the application gateway:

1. Find the public IP address for the application gateway on its **Overview** page. Or, you can select **All resources**, enter *myAGPublicIPAddress* in the search box, and then select it in the search results. Azure displays the public IP address on the **Overview** page.
2. Copy the public IP address, and then paste it into the address bar of your browser to browse that IP address.
3. Check the response. A valid response verifies that the application gateway was successfully created and can successfully connect with the backend.

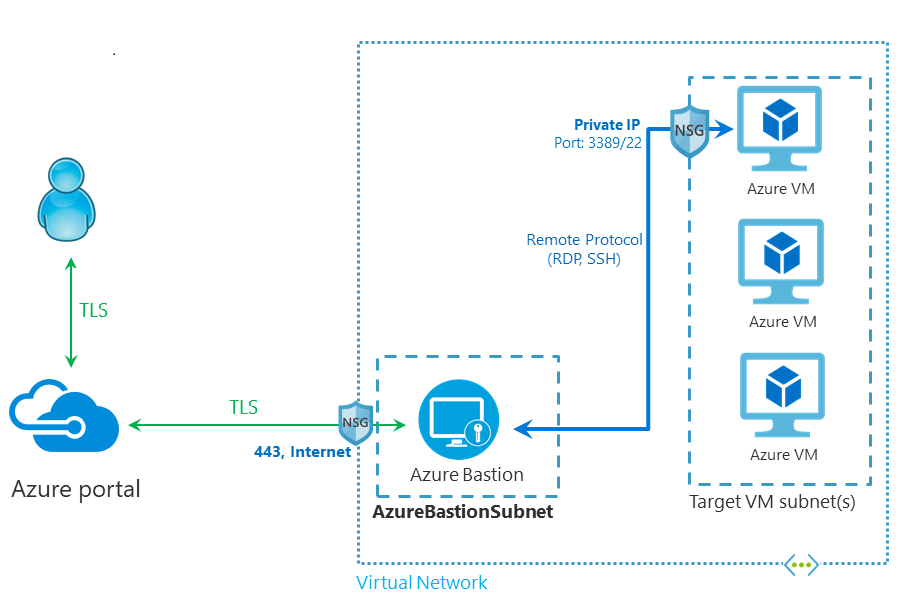


Refresh the browser multiple times and you should see connections to both myVM and myVM2.

**Azure Bastion**

Azure Bastion is a service you deploy that lets you connect to a virtual machine using your browser and the Azure portal. The Azure Bastion service is a fully platform-managed PaaS service that you provision inside your virtual network. It provides secure and seamless RDP/SSH connectivity to your virtual machines directly from the Azure portal over TLS. When you connect via Azure Bastion, your virtual machines do not need a public IP address, agent, or special client software.

Bastion provides secure RDP and SSH connectivity to all of the VMs in the virtual network in which it is provisioned. Using Azure Bastion protects your virtual machines from exposing RDP/SSH ports to the outside world, while still providing secure access using RDP/SSH.



## Key benefits

* **RDP and SSH directly in Azure portal:** You can get to the RDP and SSH session directly in the Azure portal using a single click seamless experience.
* **Remote Session over TLS and firewall traversal for RDP/SSH:** Azure Bastion uses an HTML5 based web client that is automatically streamed to your local device. You get your RDP/SSH session over TLS on port 443, enabling you to traverse corporate firewalls securely.
* **No Public IP required on the Azure VM:** Azure Bastion opens the RDP/SSH connection to your Azure virtual machine using private IP on your VM. You don't need a public IP on your virtual machine.
* **No hassle of managing Network Security Groups (NSGs)**: Azure Bastion is a fully managed platform PaaS service from Azure that is hardened internally to provide you secure RDP/SSH connectivity. You don't need to apply any NSGs to the Azure Bastion subnet. Because Azure Bastion connects to your virtual machines over private IP, you can configure your NSGs to allow RDP/SSH from Azure Bastion only. This removes the hassle of managing NSGs each time you need to securely connect to your virtual machines. For more information about NSGs, see [Network Security Groups](https://docs.microsoft.com/en-us/azure/virtual-network/network-security-groups-overview#security-rules).
* **Protection against port scanning:** Because you do not need to expose your virtual machines to the public Internet, your VMs are protected against port scanning by rogue and malicious users located outside your virtual network.
* **Protect against zero-day exploits. Hardening in one place only:** Azure Bastion is a fully platform-managed PaaS service. Because it sits at the perimeter of your virtual network, you don’t need to worry about hardening each of the virtual machines in your virtual network. The Azure platform protects against zero-day exploits by keeping the Azure Bastion hardened and always up to date for you.

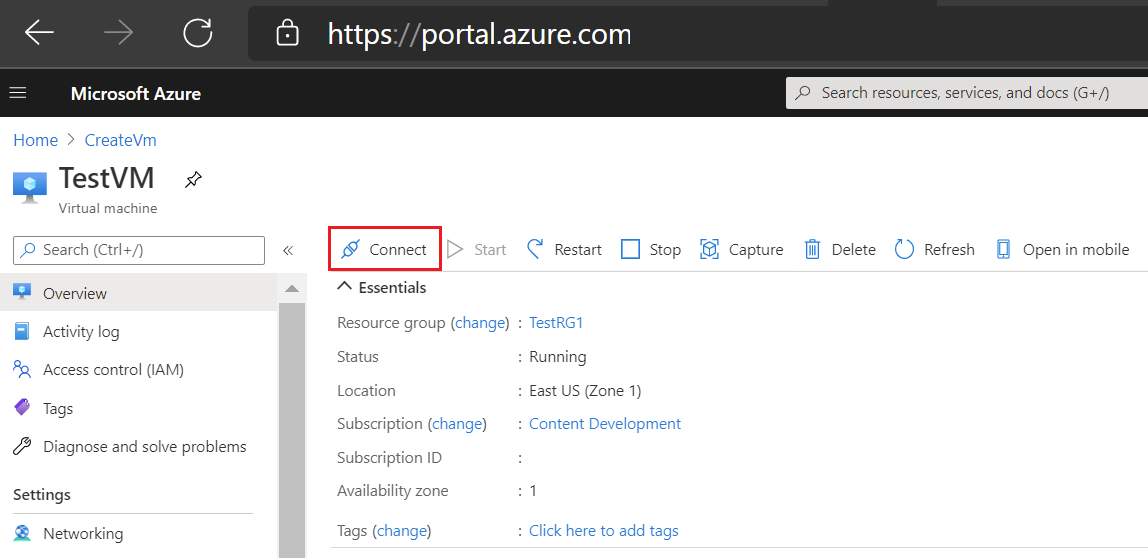
# **Configure Azure Bastion from VM settings**

Here we can configure Azure Bastion based on your VM settings in the Azure portal, and then connect to a VM via private IP address. Once the service is provisioned, the RDP/SSH experience is available to all of the virtual machines in the same virtual network. The VM doesn't need a public IP address, client software, agent, or a special configuration. If you don't need the public IP address on your VM for anything else, you can remove it. You then connect to your VM through the portal using the private IP address.

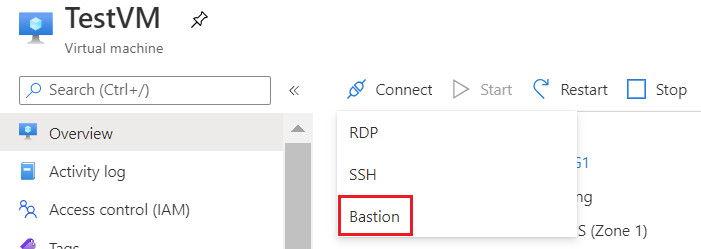
## Create a bastion host

There are a few different ways to configure a bastion host. In the following steps, you'll create a bastion host in the Azure portal directly from your VM. When you create a host from a VM, various settings will automatically populate corresponding to your virtual machine and/or virtual network.

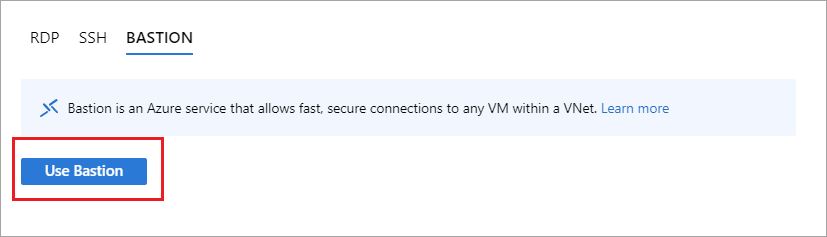
1. Sign in to the [Azure portal](https://ms.portal.azure.com/).
2. Navigate to the VM that you want to connect to, then select **Connect**.

[](https://docs.microsoft.com/en-us/azure/bastion/media/quickstart-host-portal/vm-connect.png#lightbox)

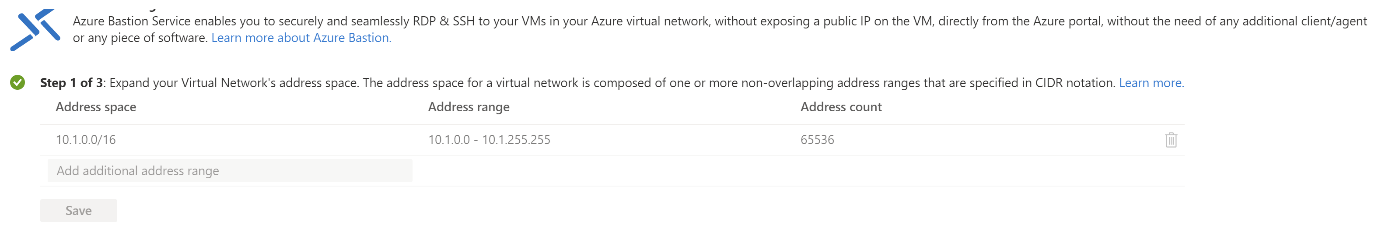
1. From the dropdown, select **Bastion**.

[](https://docs.microsoft.com/en-us/azure/bastion/media/quickstart-host-portal/bastion.png#lightbox)

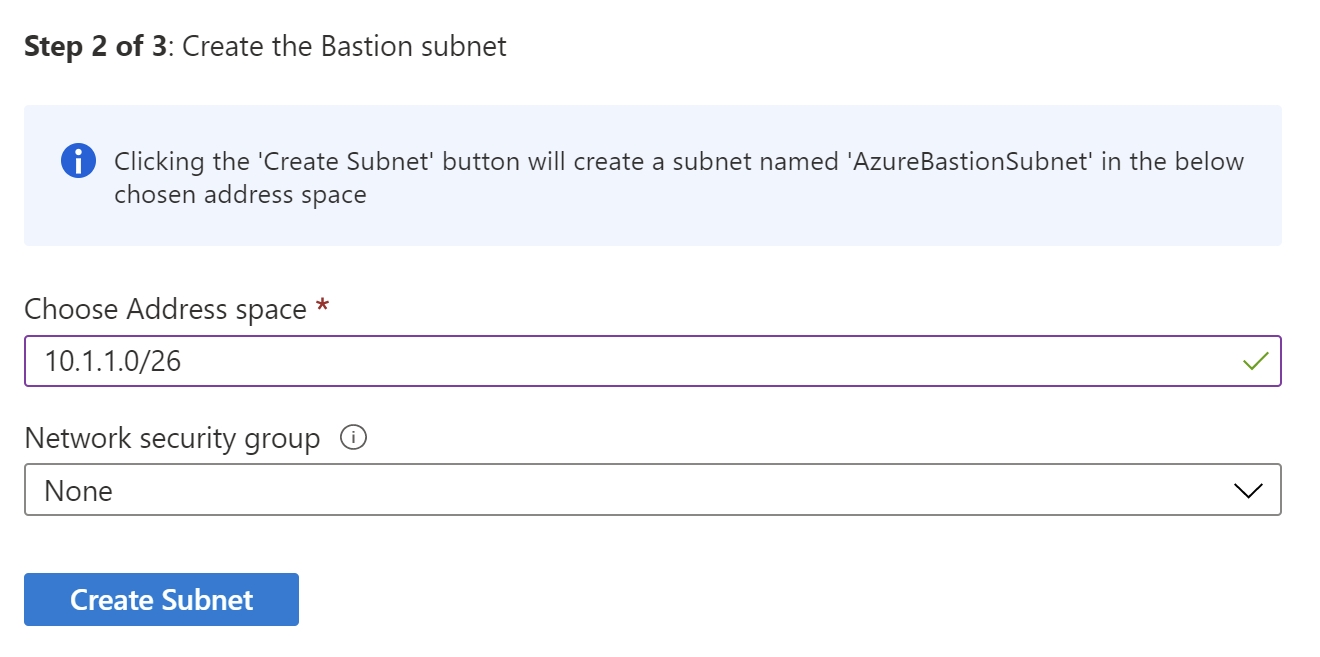
1. On the **TestVM | Connect page**, select **Use Bastion**.



1. On the **Connect using Azure Bastion** page, **Step 1**, the values are pre-populated because you are creating the bastion host directly from your VM.

[](https://docs.microsoft.com/en-us/azure/bastion/media/quickstart-host-portal/create-step-1.png#lightbox)

1. On the **Connect using Azure Bastion** page, **Step 2**, configure the subnet values. The AzureBastionSubnet address space is pre-populated with a suggested address space. The AzureBastionSubnet must have an address space of /26 or larger (/25, /24, etc.). We recommend using a /26 so that host scaling is not limited. When you finish configuring this setting, click **Create Subnet** to create the AzureBastionSubnet.



1. After the subnet creates, the page advances automatically to **Step 3**. For Step 3, use the following values:
   * **Name:** Name the bastion host.
   * **Tier:** The tier is the SKU. For this exercise, select **Standard** from the dropdown. Selecting the Standard SKU lets you configure the instance count for host scaling. The Basic SKU doesn't support host scaling. For more information, see [Configuration settings - SKU](https://docs.microsoft.com/en-us/azure/bastion/configuration-settings#skus).
   * **Instance count:** This is the setting for host scaling. Use the slider to configure. If you specify the Basic tier SKU, you are limited to 2 instances and cannot configure this setting. For more information, see [Configuration settings - host scaling](https://docs.microsoft.com/en-us/azure/bastion/configuration-settings#instance). Instance count relies on the Standard SKU. In this quickstart, you can select the instance count you'd prefer, keeping in mind any scale unit [pricing](https://azure.microsoft.com/pricing/details/azure-bastion) considerations.
   * **Public IP address:** Select **Create new**.
   * **Public IP address name:** The name of the Public IP address resource.
   * **Public IP address SKU:** Pre-configured as **Standard**.
   * **Assignment:** Pre-configured to **Static**. You can't use a Dynamic assignment for Azure Bastion.
   * **Resource group:** The same resource group as the VM.
2. After completing the values, select **Create Azure Bastion using defaults**. Azure validates your settings, then creates the host. The host and its resources take about 5 minutes to create and deploy.