Modernize the Connected Healthcare Device Platform in azure.

In this Blog, we will look how we can use Azure services to create a platform for health care devices offers multiple applications for all type of health care device management. We will use Azure Kubernetes cluster, Azure container registry, Cosmos, SQL, Azure Webapp, Application gateway, Storage account, Key vault and Service bus.

Problem statement

Company provides solution or platform for healthcare devices where any company having there healthcare devices can use the platform to maintain their devices this solution gives able to monitor the devices through communicative dashboards also enable them to verify the logs Realtime and deploy software updates. Company wanted to modernize the solution with services provided by Azure cloud and to reduce the capital expenditure.
Scalability in minimum time if the sales of devices are high in volume, they wanted to adopt Pay as you go model which can save there cost and also enables high availability. They also wants to ensure data protection and security to ensure we meet HIPPA and HITRUST for healthcare.

The existing solution was built couple of years ago which was deployed on Virtual machines in on-premises environment. This solution consists of different business critical applications developed based on Java & Nodejs. The problem with on-prem environment was as follows:

1.There was a high scope for customer acquisitions in future, but solution was not scalable.
2. On premises environment required lot of efforts to deploy each application manually and updates in different virtual machine and also we had to install lot of dependency softwares and licenses for example: - Java,Nodejs and Mysql.
3. The current solution does not have high availability.
4. Each application requires different set of VM and storage which increases manual efforts to maintain infrastructure.

Design considerations and Solution

Design consideration:-

This section explains the design considerations for deployment of this solution on Azure cloud.

- The platform should be extensible to support future digital technology road map.
- Applications should be hosted centralized.
- Reduce Manual efforts to maintain the platform by automating deployment using azure devops and cost effectiveness.
- By leveraging PAAS services from Azure we able to achieve high scalable platform
- High availability and redundancy.
- Use Azure services to provide risk detection from threats.

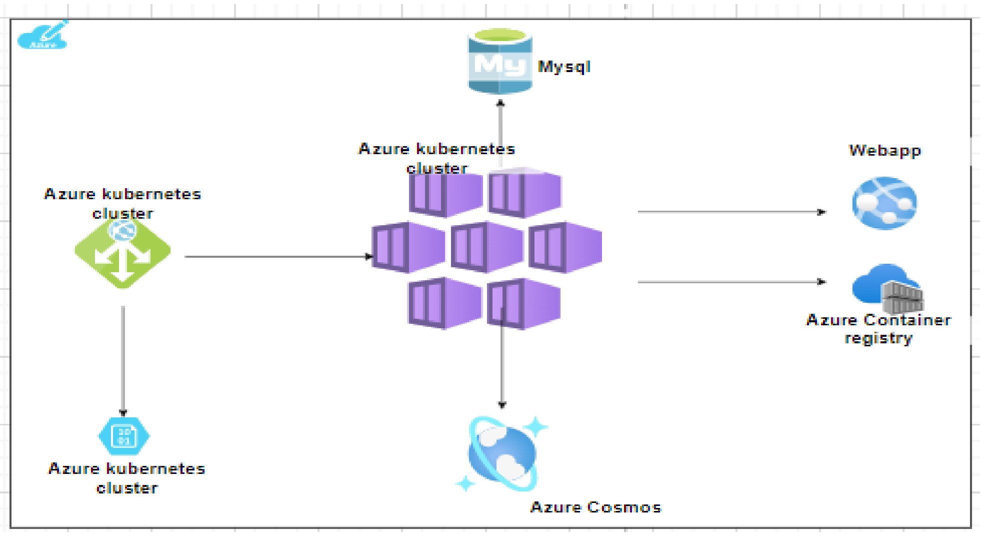
Solution

ACR - Azure Container Registry is to store docker images we can convert from java, npm or any other and will be integrated with AKS, We can enhance the feature of availability zone while creating.
AKS – Applications are hosted on Azure Kubernetes Cluster which provides high scalability, high availability with zones, we will create services which creates container/pods.
Webapp - For hosting UI we have used webapp to host static page UI.
Application gateway - For load balancing between multiple application we will use Application gateway with ingress controller of AKS to route the traffic between multiple applications hosted in AKS acting as backend pool.

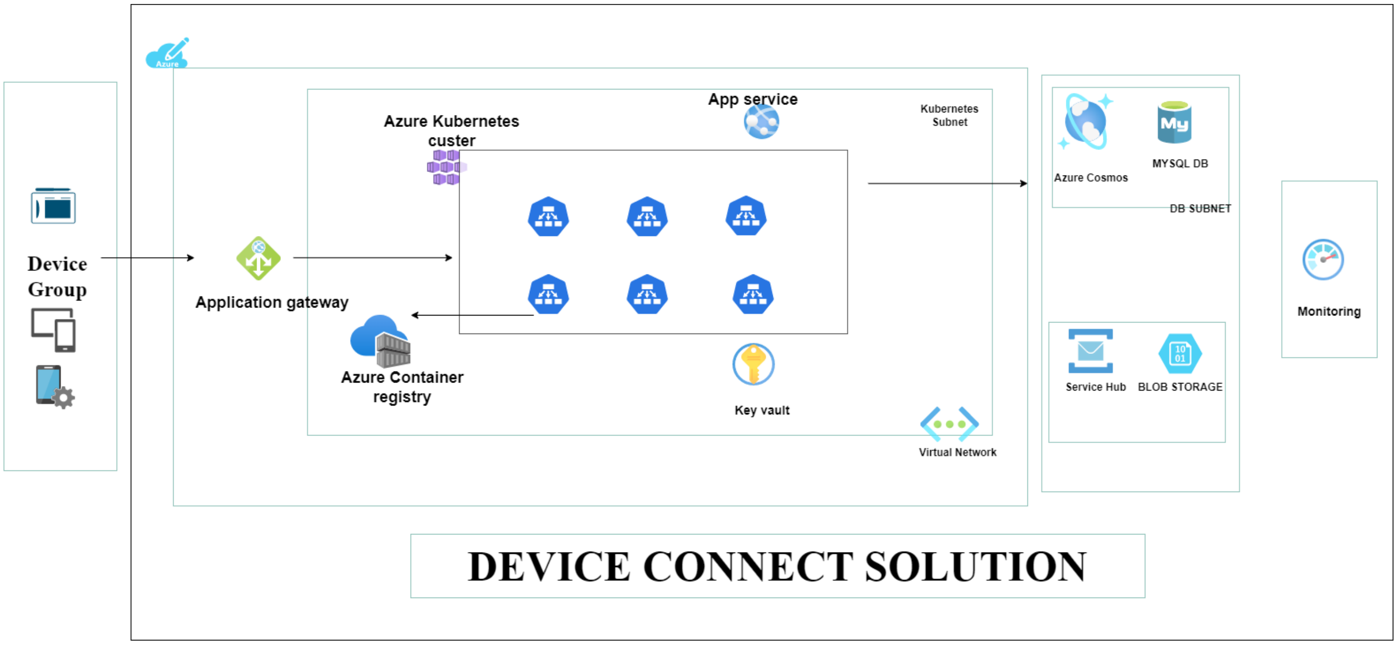
Modernize the Connected Healthcare Device Platform in azure.

Key vault - To store the Secrets for example DB username password or certificates and secrets we will use key vault.
Cosmos – We will use Cosmos to store the platform data and we will enable backup.
SQL - We will use SQL to store devices data for example device id we can enable geo redundancy backup for
Storage – We will use blob to store static error pages.
Service Bus – To queue if any files get uploaded it will process the request and store it to SQL
Azure Policy – We can use azure policies to detect compliance of the resources to meet HIPPA and HITRUST.

Solution Block



3.Proposed Architecture.



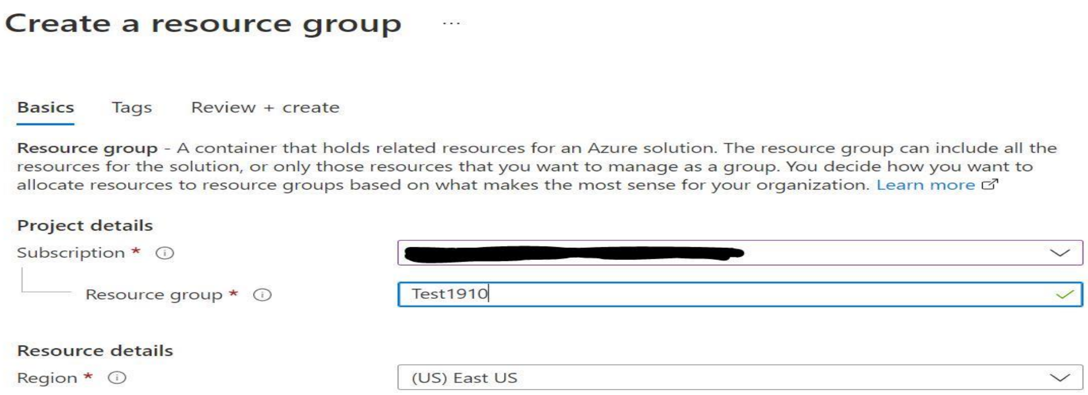
Modernize the Connected Healthcare Device Platform in azure.

Implementation Steps: -

Step 1: Resource Group Creation

Pre-requisites:
An Azure account with an active subscription.

- Login to https://portal.azure.com
- Expand the menu on the left corner and then click Resource Groups.
- Enter a unique Azure Resource Group name, choose the Azure subscription and
- location.
- Click Create. Resource Group is now created.



Create a resource group ···

Resource group - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. Learn more

Project details

Subscription * : [redacted]

Resource group * : Test1910

Resource details

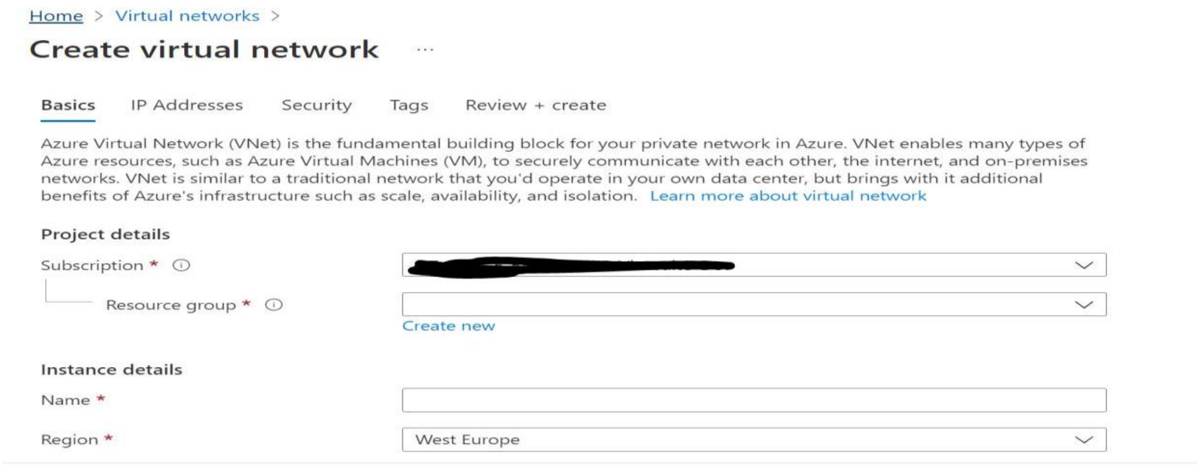
Region * : (US) East US

Step 2: Virtual Network Implementation.

Log-in to https://portal.azure.com/

- Expand the menu on the left corner and then click More Services and search for Virtual
- Networks
- Select +Create and follow below steps to create virtual network
- Azure resources can communicate with the rest of the other networks in the parent
- subscription using Vnet peering.

The network communication is restricted by applying Network Security Groups on Subnets



Create 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. Learn more about virtual network

Project details

Subscription * : [redacted]

Resource group * :

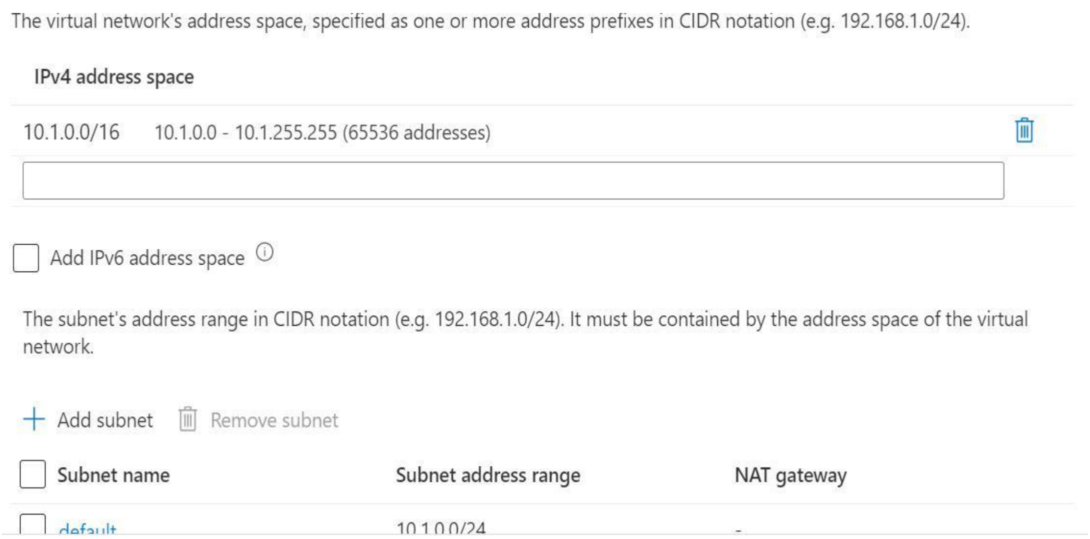
Create new

Instance details

Name * :

Region * : West Europe

Add the ip range.



The virtual network's address space, specified as one or more address prefixes in CIDR notation (e.g. 192.168.1.0/24).

IPv4 address space

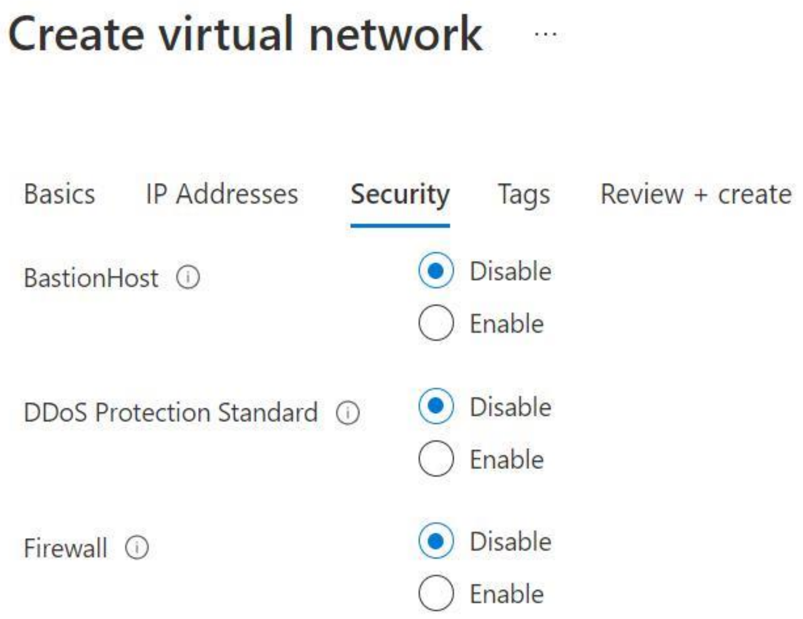
10.1.0.0/16 10.1.0.0 - 10.1.255.255 (65536 addresses)

□ Add IPv6 address space ⓘ

The subnet's address range in CIDR notation (e.g. 192.168.1.0/24). It must be contained by the address space of the virtual network.

□ Subnet name	Subnet address range	NAT gateway
□ default	10.1.0.0/24	-

Enable firewall or DDOS.



Create virtual network

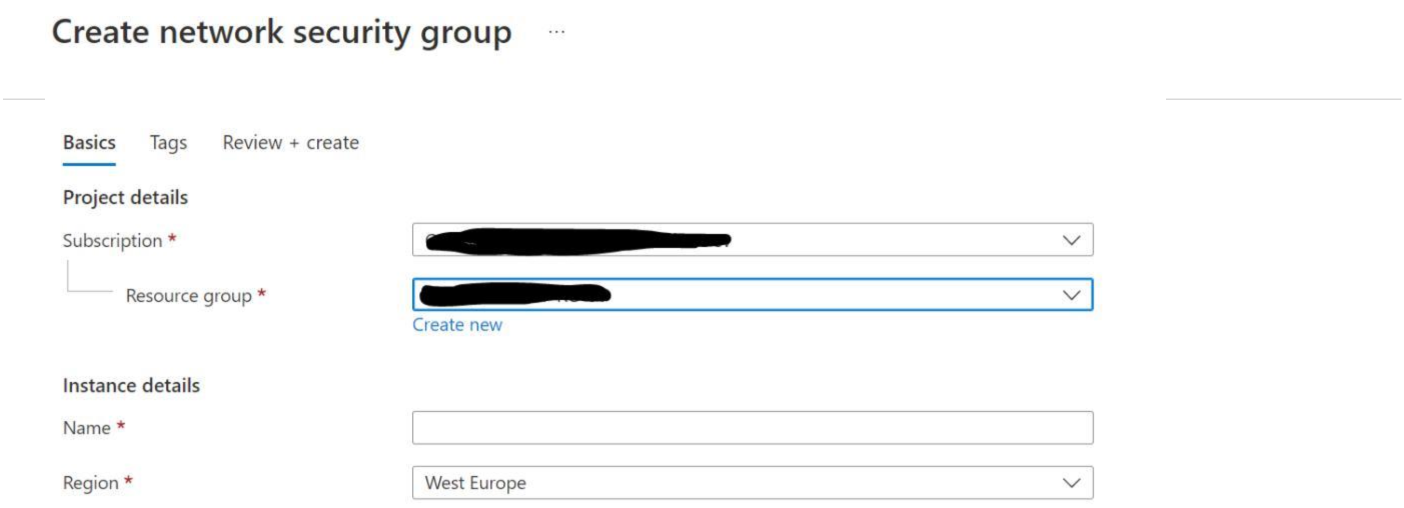
BastionHost ⓘ ● Disable ○ Enable

DDoS Protection Standard ⓘ ● Disable ○ Enable

Firewall ⓘ ● Disable ○ Enable

Add security groups:-

- In the search box at the top of the portal, enter *Network security group*.
- Select Network security groups in the search results.
- In the Create network security group page, under the Basics tab, enter or select the following value



Create network security group ···

Project details

Subscription * [redacted]

Resource group * [redacted]

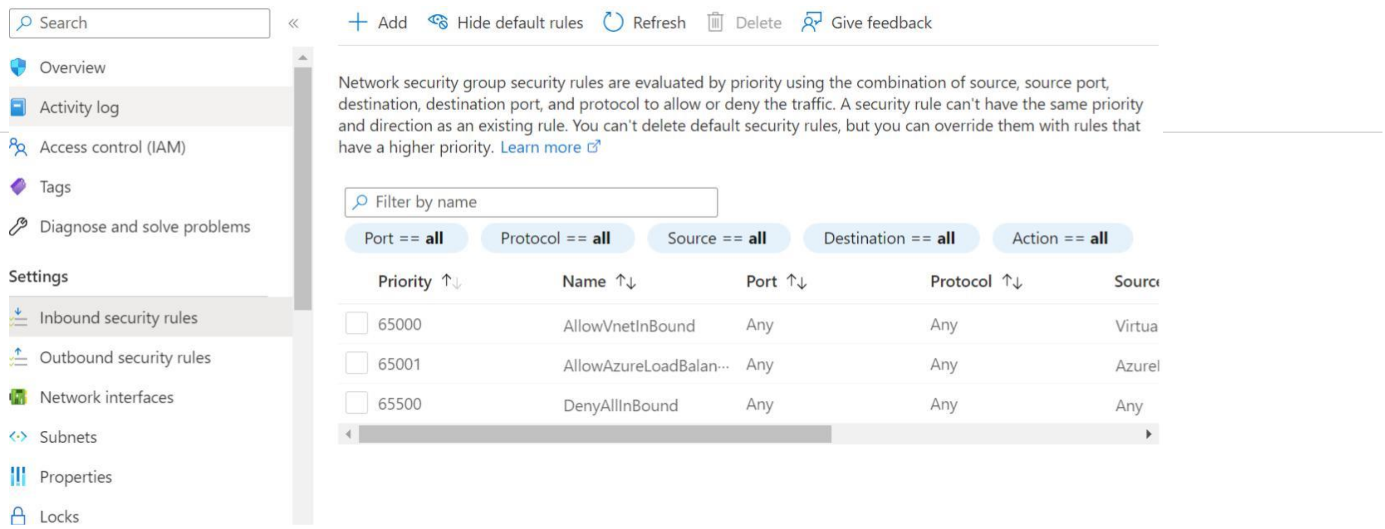
Create new

Instance details

Name *

Region * West Europe

- To add inbond and outbond rules please click and add rule for 80,8080,443.



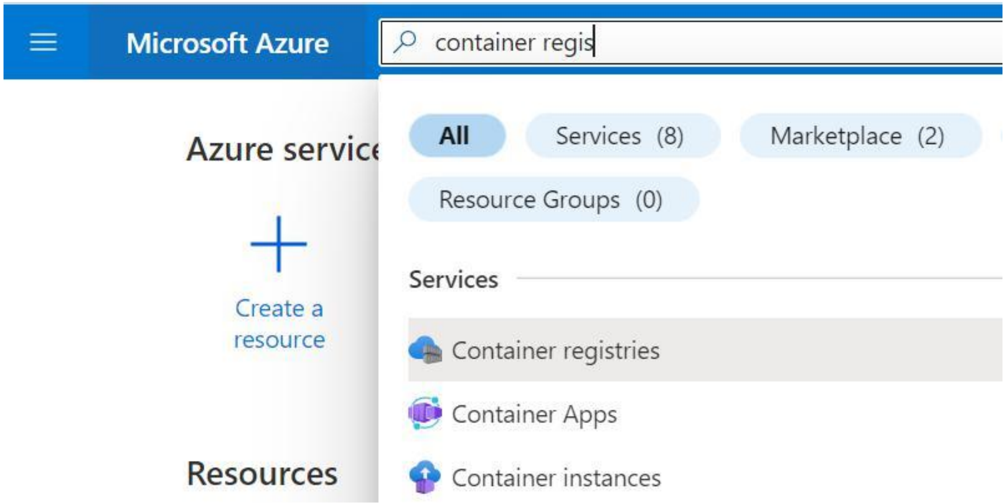
Network security group security rules are evaluated by priority using the combination of source, source port, destination, destination port, and protocol to allow or deny the traffic. A security rule can't have the same priority and direction as an existing rule. You can't delete default security rules, but you can override them with rules that have a higher priority. Learn more

	Priority ↑↓	Name ↑↓	Port ↑↓	Protocol ↑↓	Sourc
□	65000	AllowVnetInBound	Any	Any	Virtua
□	65001	AllowAzureLoadBalan···	Any	Any	Azurel
□	65500	DenyAllInBound	Any	Any	Any

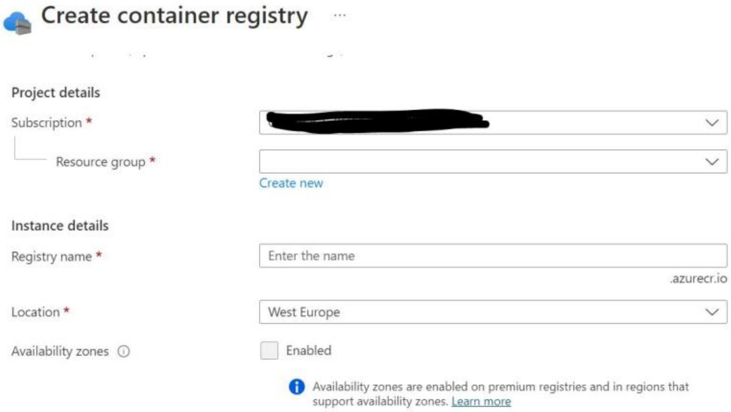
Step 3: Create Container registry.

We need container registry to act as central repository to store the images or code, We can fetch the code from ACR from other resources securely.

- In the search box at the top of the portal, enter *container registry*.



- Click Container registries and enter create.
- In the Create network security group page, under the Basics tab, enter or select the following values:



Project details

Subscription * [redacted]

Resource group *

Create new

Instance details

Registry name * Enter the name .azurecr.io

Location * West Europe

Availability zones ⓘ □ Enabled

Availability zones are enabled on premium registries and in regions that support availability zones. Learn more

Once Azure container registry is created we can push the images to push the image please use commands:-

- **Using Az CLI**

```
#az login
#az acr login --name myregistry
```

- **Using Docker commands**

```
# docker login <myregistry.azurecr.io>
```

- Tag your image

```
# docker tag device:1.1 myregistry.azurecr.io/samples/device:1.1
```

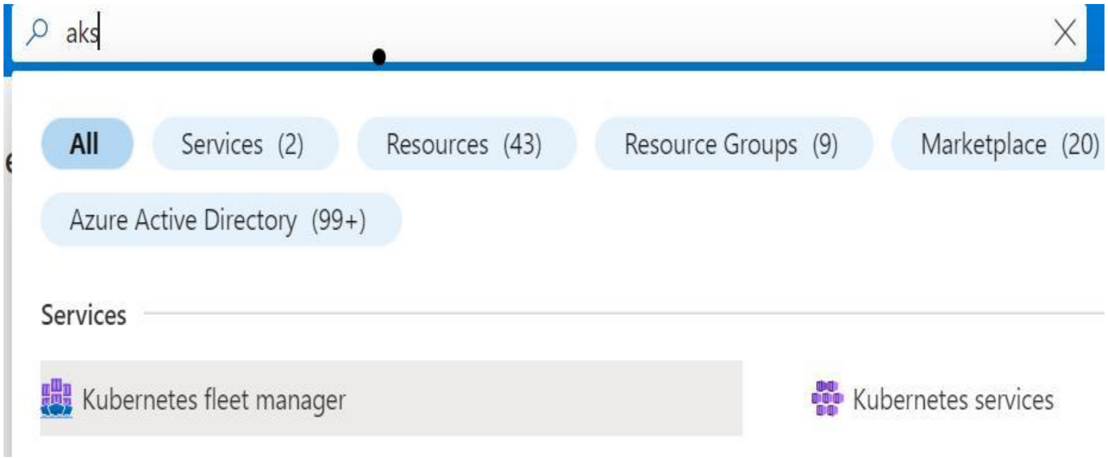
- **To push any image build locally in your computer**

```
#docker push myregistry.azurecr.io/samples/device:1.1
```

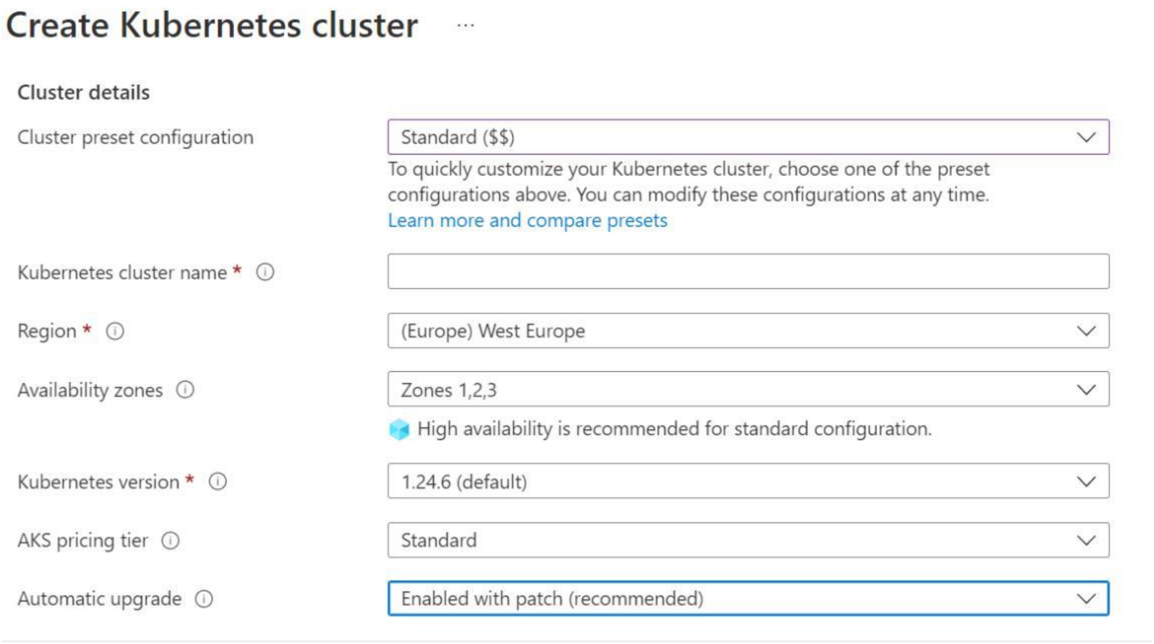
Once pushed you will see the output like below.

Step 4:- Create AKS and integrate with ACR.

- In the search box at the top of the portal, enter *AKS and select Kubernetes*.
- Select Create Kubernetes Cluster.



- Provide name for your cluster and select high availability and zone which provides you fault tolerance. We can select up to 3 zones which means your nodes/vm will be created in three different Data center which are miles away from each other.



Create Kubernetes cluster ···

Cluster details

Cluster preset configuration: Standard ($$)

To quickly customize your Kubernetes cluster, choose one of the preset configurations above. You can modify these configurations at any time.
Learn more and compare presets

Kubernetes cluster name *

Region *: (Europe) West Europe

Availability zones: Zones 1,2,3

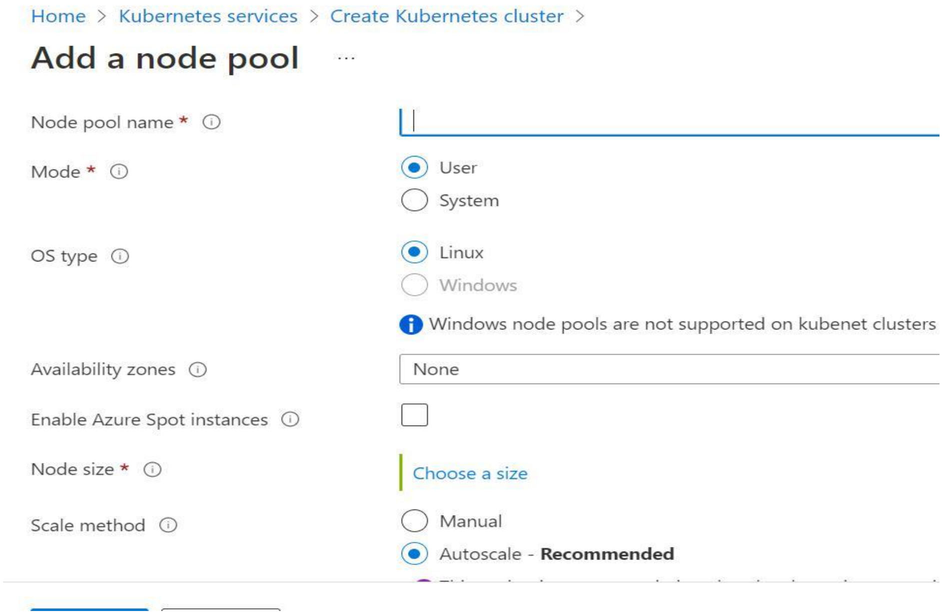
High availability is recommended for standard configuration.

Kubernetes version *: 1.24.6 (default)

AKS pricing tier: Standard

Automatic upgrade: Enabled with patch (recommended)

- Create the Node pool, Select the name and VM size with OS type as Linux and Auto scale to provide scalability in future in case if the utilization increases node pool will automatically add the VM which can save downtime.



Home > Kubernetes services > Create Kubernetes cluster >

Add a node pool ...

Node pool name * ⓘ

Mode * ⓘ ● User ○ System

OS type ⓘ ● Linux ○ Windows

ⓘ Windows node pools are not supported on kubenet clusters

Availability zones ⓘ None

Enable Azure Spot instances ⓘ □

Node size * ⓘ Choose a size

Scale method ⓘ ○ Manual ● Autoscale - **Recommended**

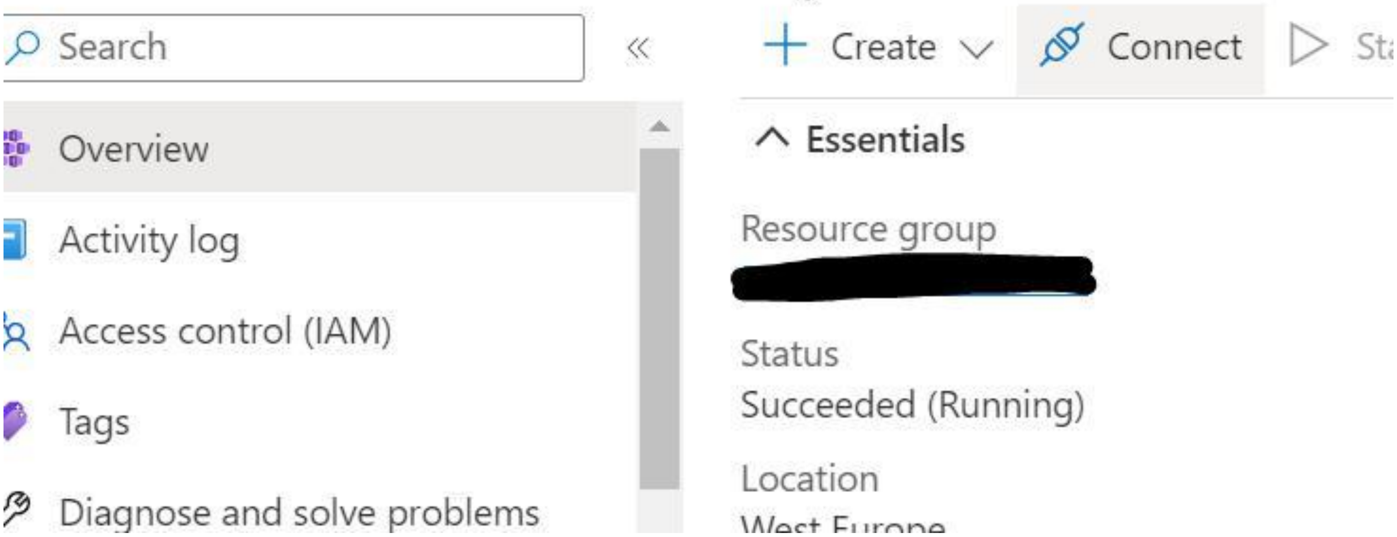
Integrate with ACR

Using Azure CLI.

```
# az aks update -n <Clustername> -g <Resource grupname> --attach-acr <ACR name>
```

To host application please follow below steps.

- Connect to AKS,Select connect in overview of AKS and connect using the credentials.



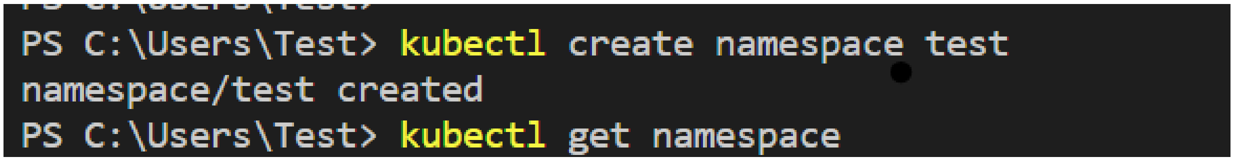
Resource group
[redacted]

Status
Succeeded (Running)

Location
West Europe

- Create Namespace

Create the namespace with below command and verify also



```
PS C:\Users\Test> kubectl create namespace test
namespace/test created
PS C:\Users\Test> kubectl get namespace
```

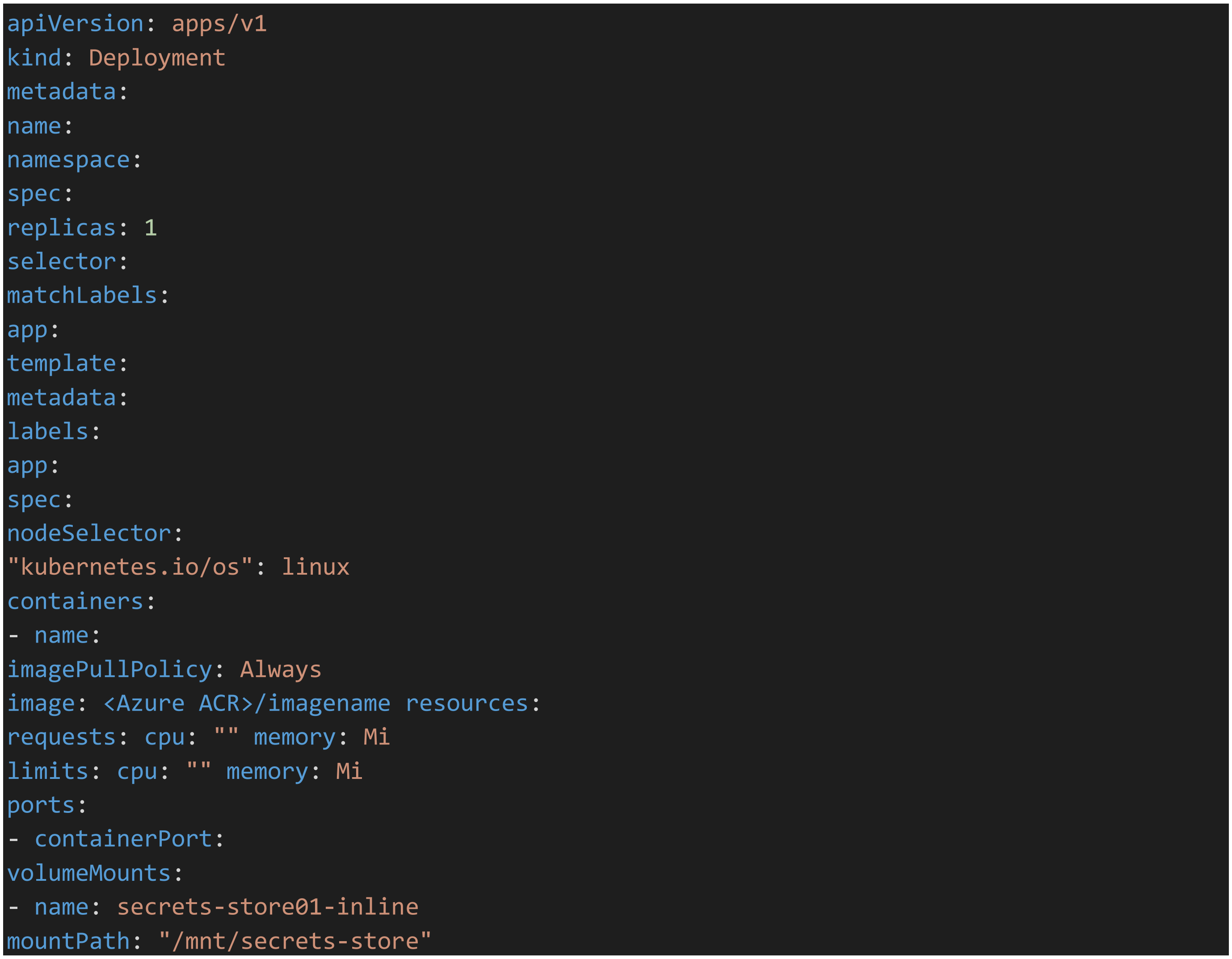
- Create deployment of application by deployment.yaml

#cd <folder in local computer where you have store deployment.yaml

#kubectl apply -f <testdeployment.yaml

Please find one of the sample codes.

- Fill the name of your deployment.
- Fill the name of your azure ACR and image name for application.
- CPU and Memory as per your usage.
- Add the key vault to add secrets



```
apiVersion: apps/v1
kind: Deployment
metadata:
name:
namespace:
spec:
replicas: 1
selector:
matchLabels:
app:
template:
metadata:
labels:
app:
spec:
nodeSelector:
"kubernetes.io/os": linux
containers:
- name:
imagePullPolicy: Always
image: <Azure ACR>/imagename resources:
requests: cpu: "" memory: Mi
limits: cpu: "" memory: Mi
ports:
- containerPort:
volumeMounts:
- name: secrets-store01-inline
mountPath: "/mnt/secrets-store"
```

```
readOnly: true
env:
- name:
valueFrom:
secretKeyRef:
name:
key:
```

- Verify the pods by running the below commands

  #Kubectl get pods -A

- Create ingress by deploying ingress.yaml file

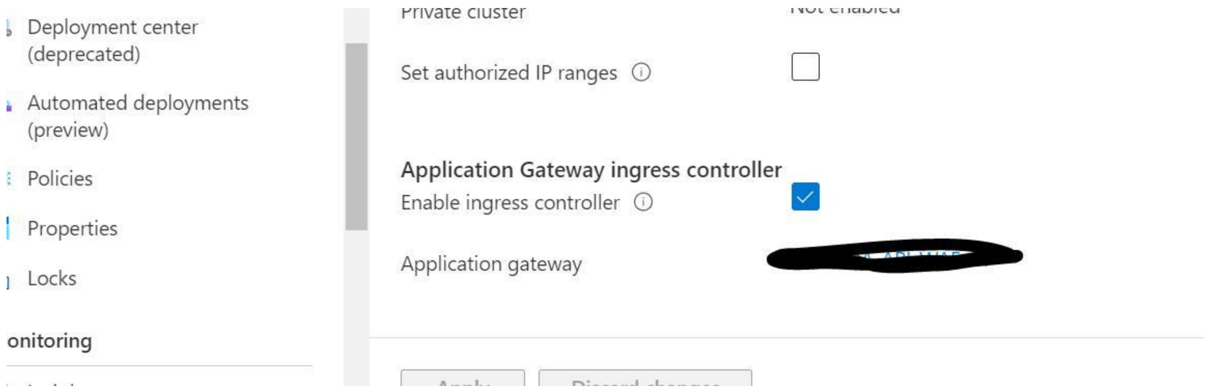
  Sample code

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  labels:
    app:
  name:
  namespace:
  annotations:
    appgw.ingress.kubernetes.io/request-timeout: "300"
    appgw.ingress.kubernetes.io/ssl-redirect: "true"
    kubernetes.io/ingress.class: azure/application-gateway
    appgw.ingress.kubernetes.io/health-probe-status-codes: "200-502"

spec:
  tls:
  - hosts:
    - <host url>
  rules:
  - host: <host url>
    http:
      paths:
      - backend:
          service:

   name:      # Name of your below service
            port:
              number: 80         # Port where your below service is listening on
        path: /                  # Path where ingress listens
        pathType: Prefix
```

- Upgrade an existing AKS cluster with Azure Key Vault Provider for Secrets Store CSI Driver support,This will enable to use keyvault when pod tried to get secrets from keyvault.

  #az aks enable-addons --addons azure-keyvault-secrets-provider --name myAKSCluster --resource-group myResource Group.

- Attach Application gateway as ingress controller in networking. Once application gateway is created,this step you can perform once application gateway is created.


Deployment center (deprecated)
Automated deployments (preview)
Policies
Properties
Locks
onitoring
Apply
Discard changes

Private cluster Not enabled

Set authorized IP ranges ☐

Application Gateway ingress controller

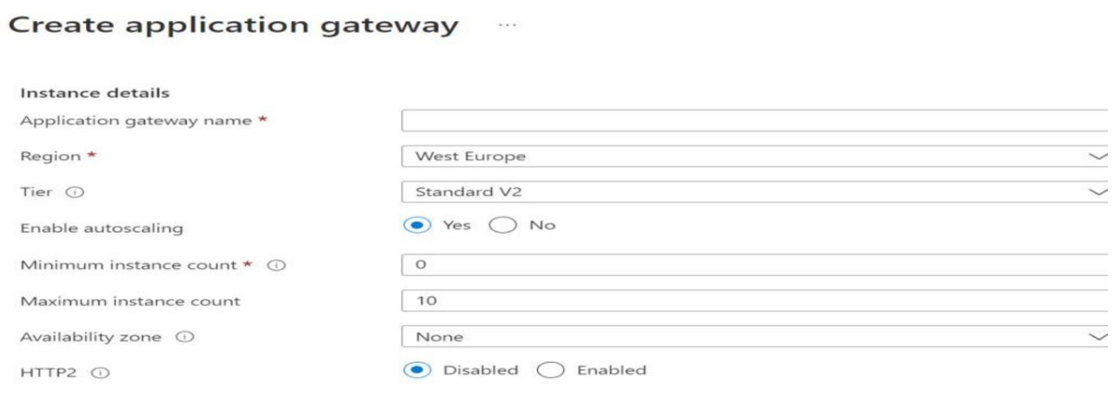
Enable ingress controller ☑

Application gateway [redacted]

Step 5 :- Application Gateway

In the search box at the top of the portal, enter Application gateway.

Select Create and select tier V2.



Create application gateway

Instance details

Application gateway name *

Region * West Europe

Tier Standard V2

Enable autoscaling ● Yes ○ No

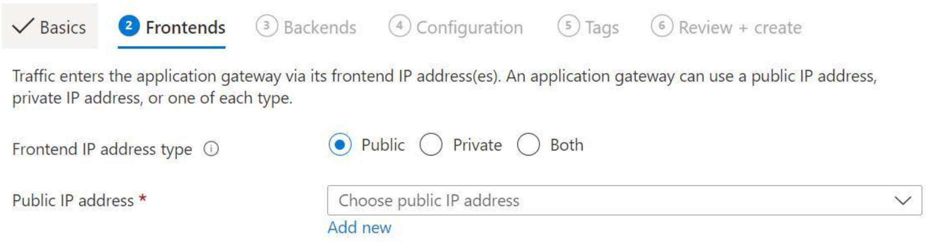
Minimum instance count * 0

Maximum instance count 10

Availability zone None

HTTP2 ● Disabled ○ Enabled

- In front end choose public ip.



✓ Basics | 2 Frontends | 3 Backends | 4 Configuration | 5 Tags | 6 Review + create

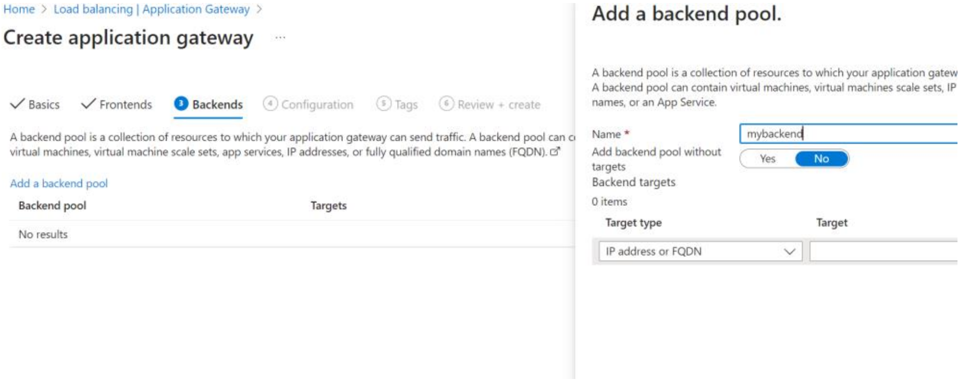
Traffic enters the application gateway via its frontend IP address(es). An application gateway can use a public IP address, private IP address, or one of each type.

Frontend IP address type ⓘ ● Public ○ Private ○ Both

Public IP address * Choose public IP address

Add new

On the Backends tab, select Add a backend pool.



Home > Load balancing | Application Gateway >

Create application gateway ···

✓ Basics ✓ Frontends 3 Backends 4 Configuration 5 Tags 6 Review + create

A backend pool is a collection of resources to which your application gateway can send traffic. A backend pool can c
virtual machines, virtual machine scale sets, app services, IP addresses, or fully qualified domain names (FQDN).

Add a backend pool

Backend pool	Targets
No results	

Add a backend pool.

A backend pool is a collection of resources to which your application gatew
A backend pool can contain virtual machines, virtual machines scale sets, IP
names, or an App Service.

Name * mybackend

Add backend pool without targets Yes [✓ No]

Backend targets

0 items

Target type	Target
IP address or FQDN	

- 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

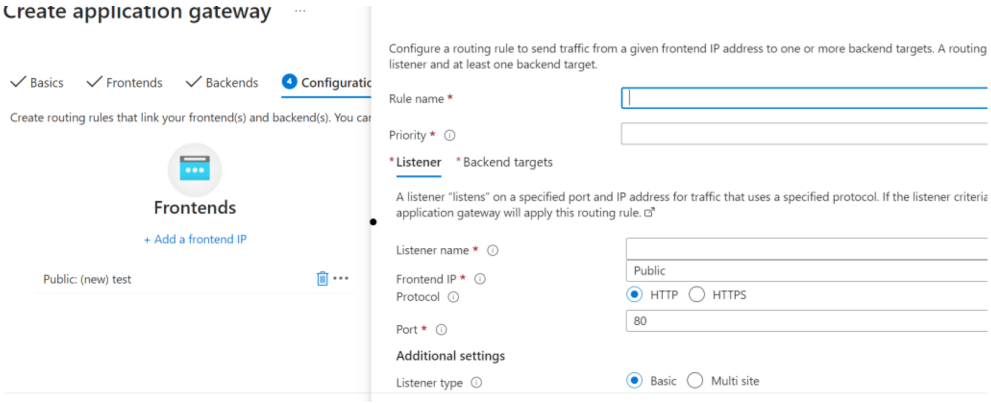
- Add backend pool without targets: Select Yes to create a backend pool with no targets. You'll add backend targets after creating the application gateway. Backend pool will be added once you add ingress.In the Add a backend pool window, select Add to save the backend pool configuration and return to the Backends tab.

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

- Select Add a routing rule in the Routing rules column.

- 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 or you can leave it as we add



Create application gateway ...

✓ Basics ✓ Frontends ✓ Backends 4 Configuratio

Create routing rules that link your frontend(s) and backend(s). You car

Frontends

+ Add a frontend IP

Public: (new) test

Configure a routing rule to send traffic from a given frontend IP address to one or more backend targets. A routing listener and at least one backend target.

Rule name *

Priority *

*Listener *Backend targets

A listener "listens" on a specified port and IP address for traffic that uses a specified protocol. If the listener criteria application gateway will apply this routing rule.

Listener name *

Frontend IP * Public

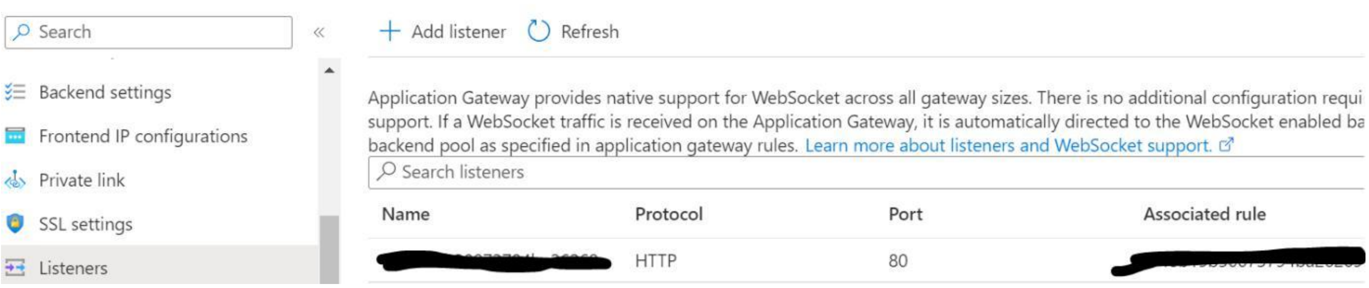
Protocol ● HTTP ○ HTTPS

Port * 80

Additional settings

Listener type ● Basic ○ Multi site

- 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.

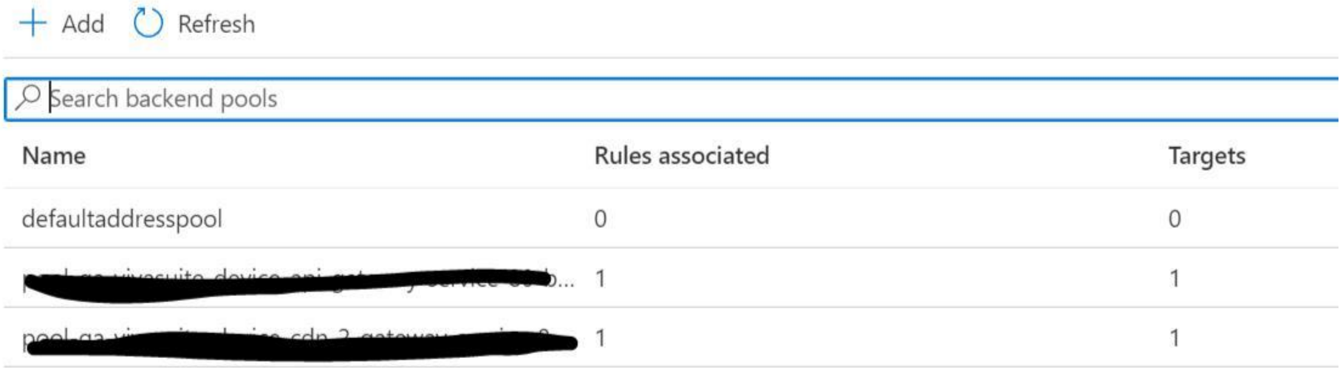


Application Gateway provides native support for WebSocket across all gateway sizes. There is no additional configuration requi support. If a WebSocket traffic is received on the Application Gateway, it is automatically directed to the WebSocket enabled ba backend pool as specified in application gateway rules. Learn more about listeners and WebSocket support.

Search listeners

Name	Protocol	Port	Associated rule
[redacted]	HTTP	80	[redacted]

- Verify all the setup and run the pods.
- Once you have run the pod you can verify in backend pool of the application gateway.



Name	Rules associated	Targets
defaultaddresspool	0	0
[redacted]	1	1
[redacted]	1	1

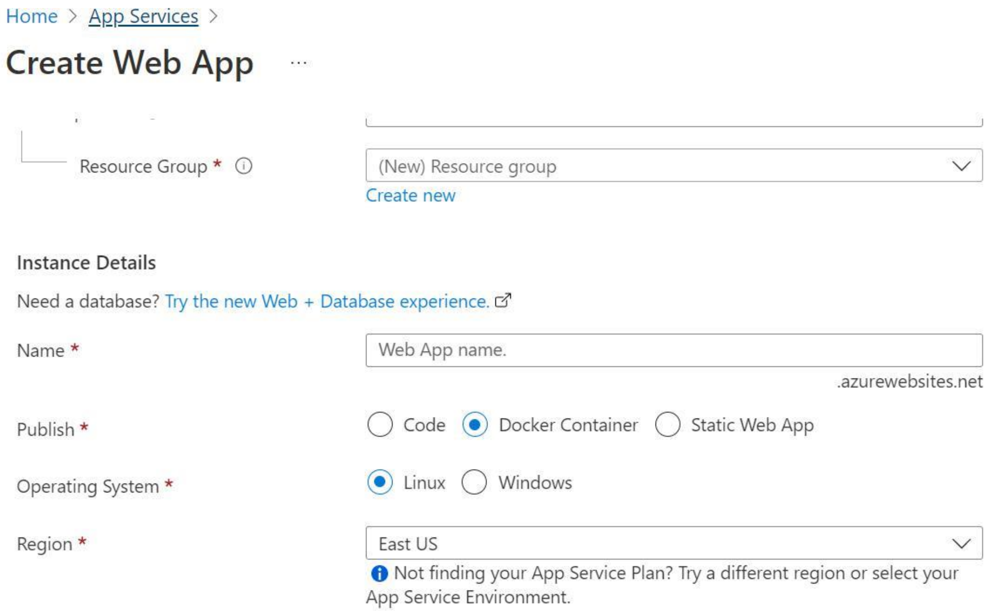
Verify the Workflow by testing the page and registering device using API.

- Open the domain <Yourdevicename..com>

Step 6:- Azure webapp and Appservice plan.

Webapp is used to host the static UI oage ,You have feasibility to route traffic directly to Webapp or call from aks container.

- In the search box at the top of the portal, enter *container registry*.
- Select Create.
- Provide name for your webapp,Select Docker container with Linux and Appservice plan.



Home > App Services >

Create Web App ...

Resource Group * (New) Resource group

Create new

Instance Details

Need a database? Try the new Web + Database experience.

Name * Web App name.

.azurewebsites.net

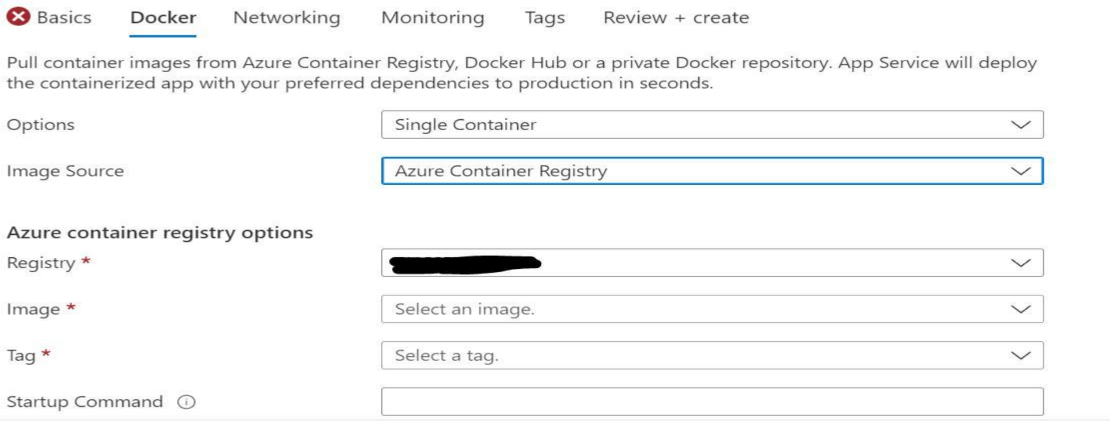
Publish * ○ Code ● Docker Container ○ Static Web App

Operating System * ● Linux ○ Windows

Region * East US

Not finding your App Service Plan? Try a different region or select your App Service Environment.

- Select the Image source as ACR and provide Image name of the UI which webapp will fetch from ACR and create docker Webapp and start the webapp.



Basics | Docker | Networking | Monitoring | Tags | Review + create

Pull container images from Azure Container Registry, Docker Hub or a private Docker repository. App Service will deploy the containerized app with your preferred dependencies to production in seconds.

Options: Single Container

Image Source: Azure Container Registry

Azure container registry options

Registry * [redacted]

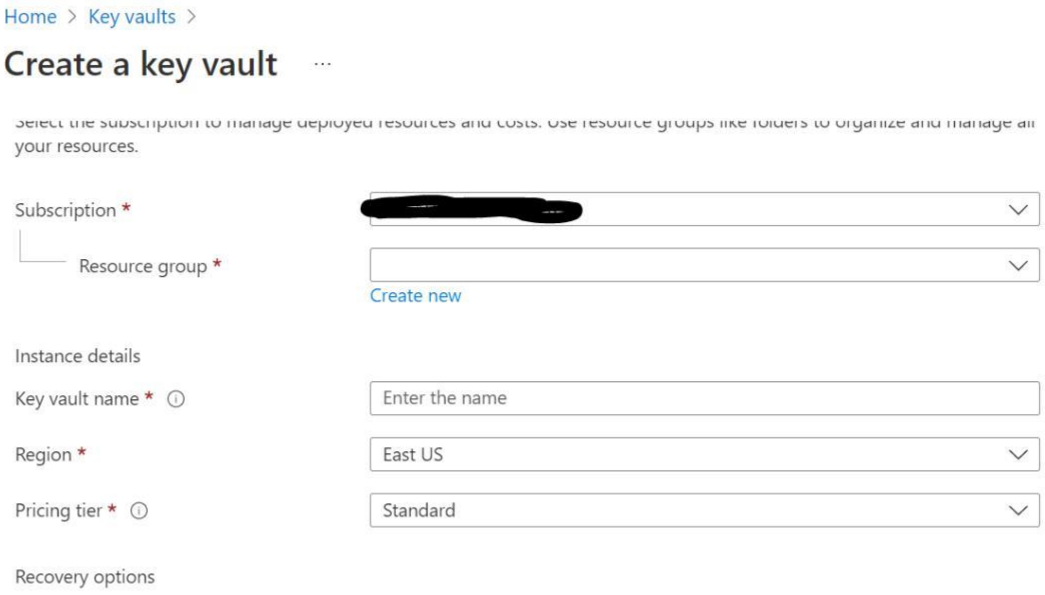
Image * Select an image.

Tag * Select a tag.

Startup Command

Step 5: Create Keyvault.

- In the search box at the top of the portal, enter *Keyvault* .
- Select Create keyvault.



Home > Key vaults >

Create a key vault ···

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

Subscription * [redacted]

Resource group *

Create new

Instance details

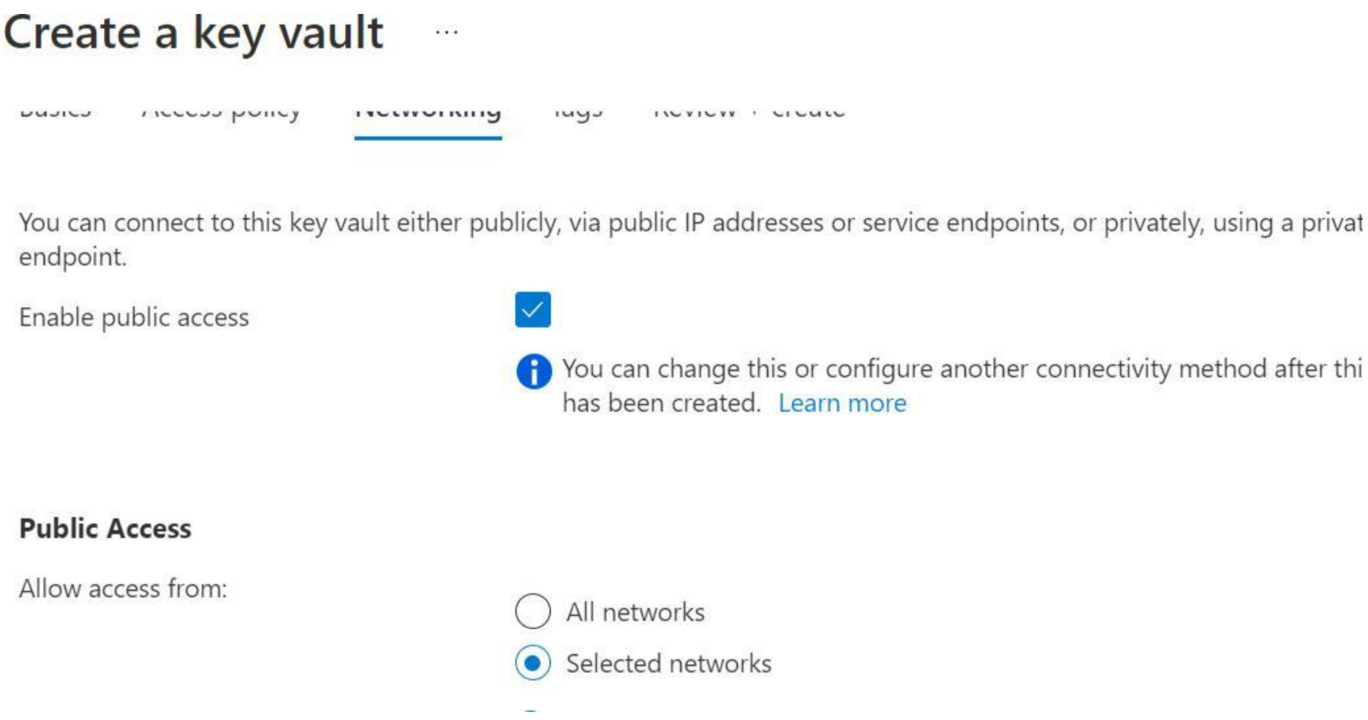
Key vault name * ⓘ Enter the name

Region * East US

Pricing tier * ⓘ Standard

Recovery options

- In Keyvault networking please add subnet which is required.



Create a key vault ···

Basics Access policy Networking Tags Review + create

You can connect to this key vault either publicly, via public IP addresses or service endpoints, or privately, using a privat endpoint.

Enable public access ☑

ⓘ You can change this or configure another connectivity method after thi has been created. Learn more

Public Access

Allow access from:

○ All networks

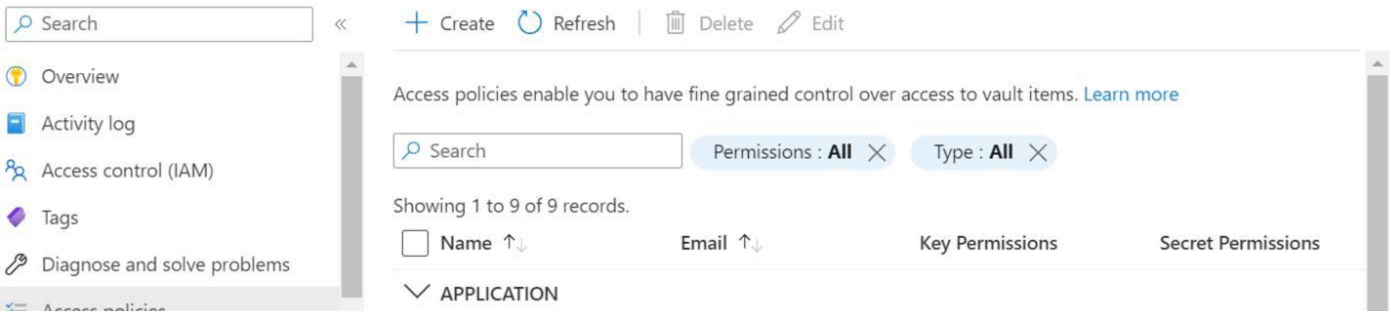
● Selected networks

- In access policies please add Node pool ID of AKS so that pods will have access to fetch secrets or certificates from keyvault.

To get the node pool id please run below commands

```
#az aks nodepool list --cluster-name --resource-group
```

Add the node pool Id in access policies of Keyvault to read secrets and certificates.

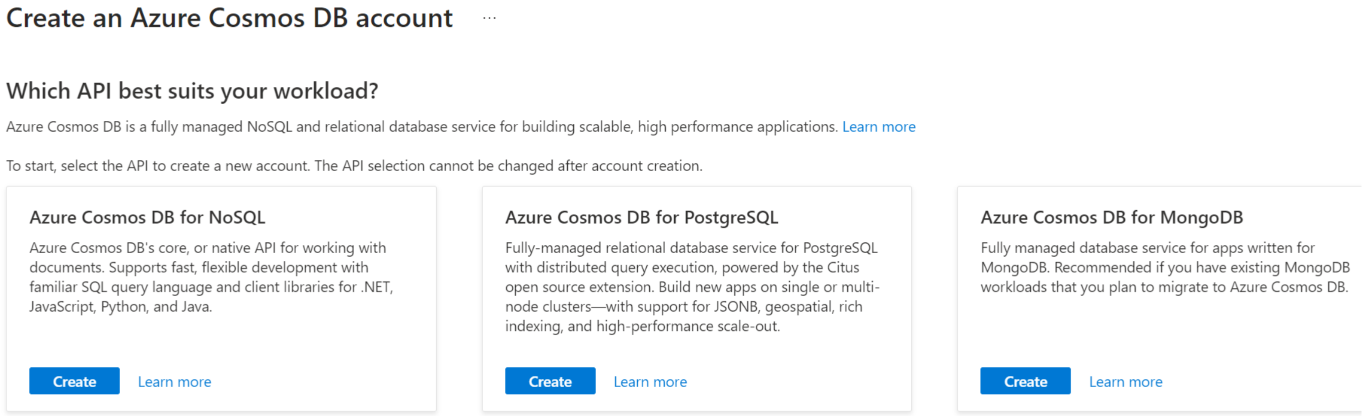


Access policies enable you to have fine grained control over access to vault items. Learn more

Showing 1 to 9 of 9 records.

Step 6:Create Cosmos DB.

- In the search box at the top of the portal, enter *Cosmos*
- Select Create Cosmos for mongodb.



Create an Azure Cosmos DB account ...

Which API best suits your workload?

Azure Cosmos DB is a fully managed NoSQL and relational database service for building scalable, high performance applications. Learn more

To start, select the API to create a new account. The API selection cannot be changed after account creation.

Azure Cosmos DB for NoSQL

Azure Cosmos DB's core, or native API for working with documents. Supports fast, flexible development with familiar SQL query language and client libraries for .NET, JavaScript, Python, and Java.

Create Learn more

Azure Cosmos DB for PostgreSQL

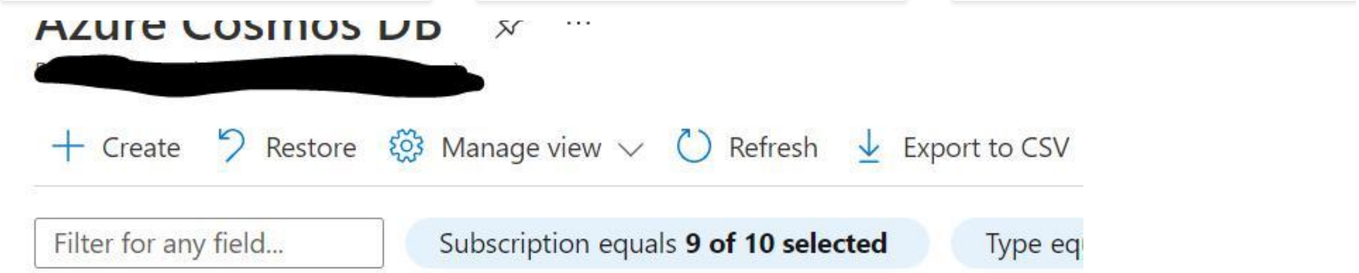
Fully-managed relational database service for PostgreSQL with distributed query execution, powered by the Citus open source extension. Build new apps on single or multi-node clusters—with support for JSONB, geospatial, rich indexing, and high-performance scale-out.

Create Learn more

Azure Cosmos DB for MongoDB

Fully managed database service for apps written for MongoDB. Recommended if you have existing MongoDB workloads that you plan to migrate to Azure Cosmos DB.

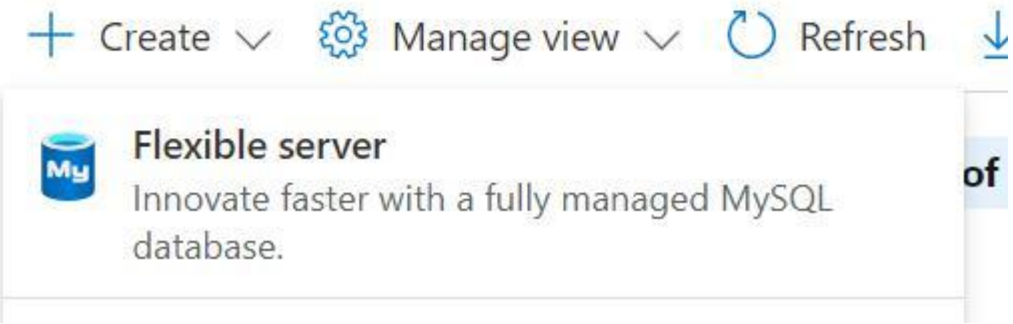
Create Learn more



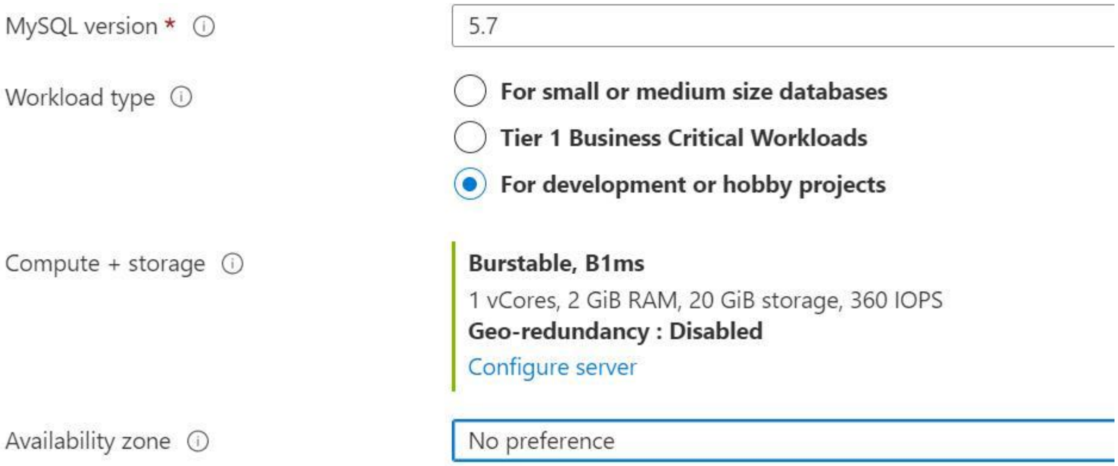
[redacted]

Step 7: Create Mysql flexible server.

- In the search box at the top of the portal, enter *Mysql Flexible server* .
- Select Create flexible server.



- Select Geo-Redundancy for high availability.



MySQL version * 5.7

Workload type
○ For small or medium size databases
○ Tier 1 Business Critical Workloads
● For development or hobby projects

Compute + storage
Burstable, B1ms
1 vCores, 2 GiB RAM, 20 GiB storage, 360 IOPS
Geo-redundancy : Disabled
Configure server

Availability zone No preference

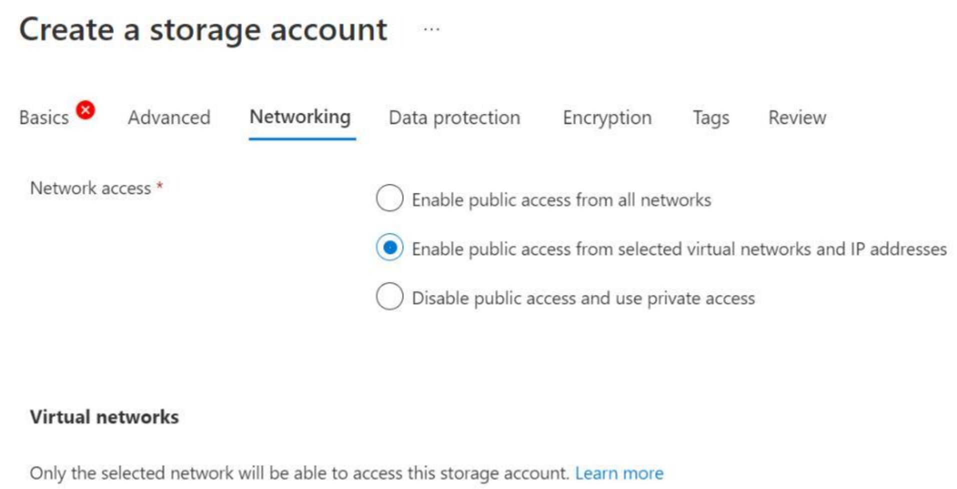
- Select Private network to access from your Virtual network.

Note :- Due to restriction I cannot share the code base to integrate SQL and cosmos with applications

Step 8 : Create Storage.

We need storage account to connect to listeners to fetch error pages or any other static page for UI.

- In the search box at the top of the portal, enter *Storage*.
- Select Create storage account.
- Enable public access from selected IP.



Create a storage account ···

Network access *

○ Enable public access from all networks

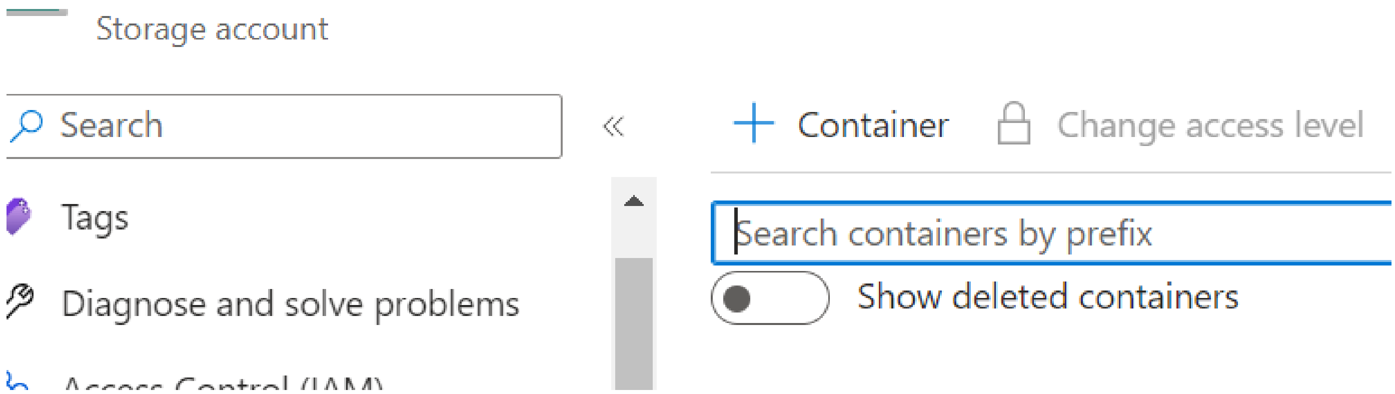
● Enable public access from selected virtual networks and IP addresses

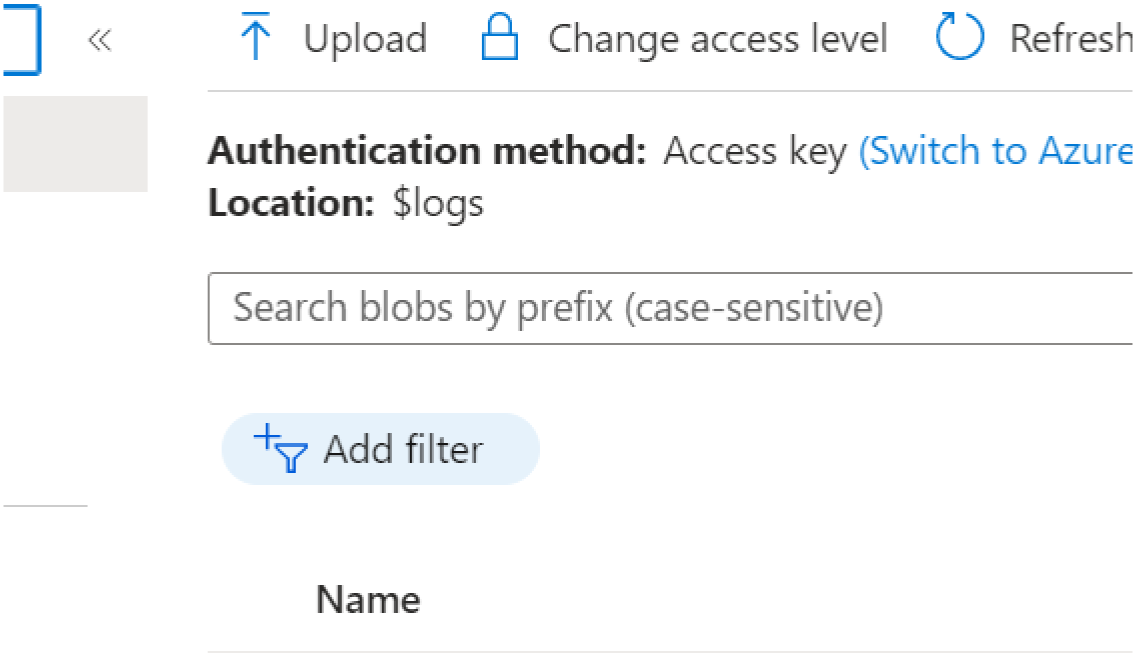
○ Disable public access and use private access

Virtual networks

Only the selected network will be able to access this storage account. Learn more

- Create container from left panel and add the file of static webpage on container using upload button in container.



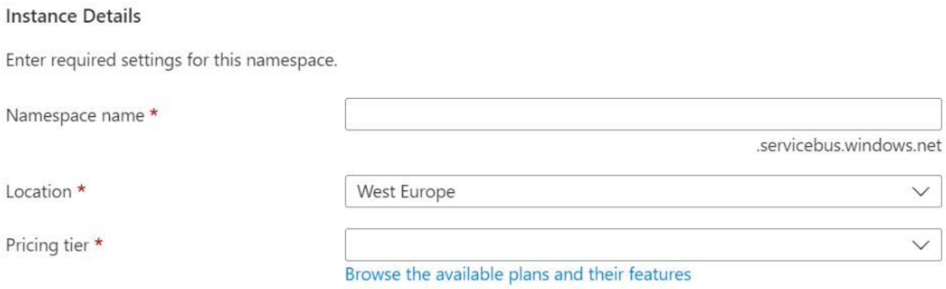


Authentication method: Access key (Switch to Azure
Location: $logs

Step 9 : Create Service Bus.

We can use service bus as queue for the incoming requests.

- In the search box at the top of the portal, enter Service *Bus*
- Select Create service bus.



Instance Details

Enter required settings for this namespace.

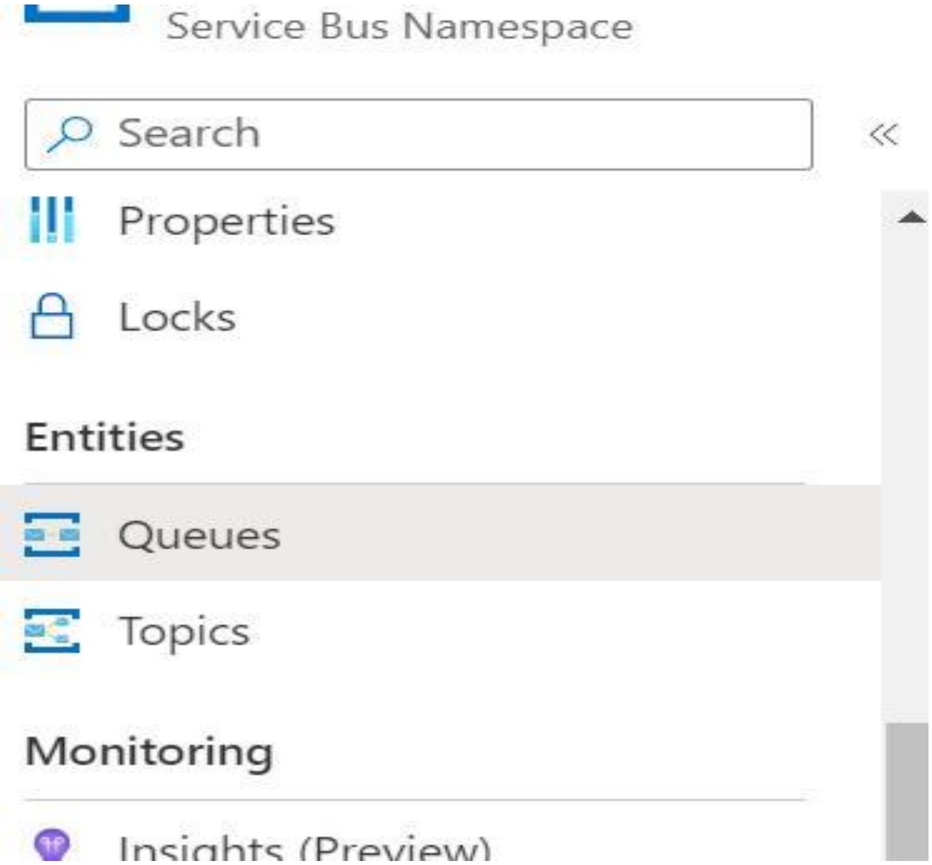
Namespace name * .servicebus.windows.net

Location * West Europe

Pricing tier *

- In Networking select private network to access the resource internally.

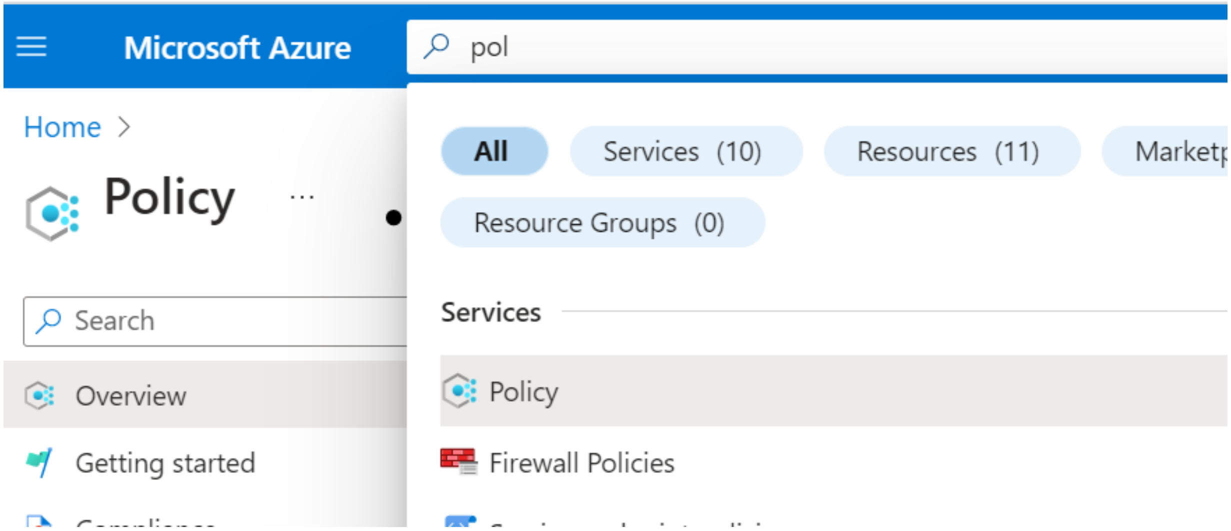
- Once Namespace is ready, Select queue and topics from left panel and create.



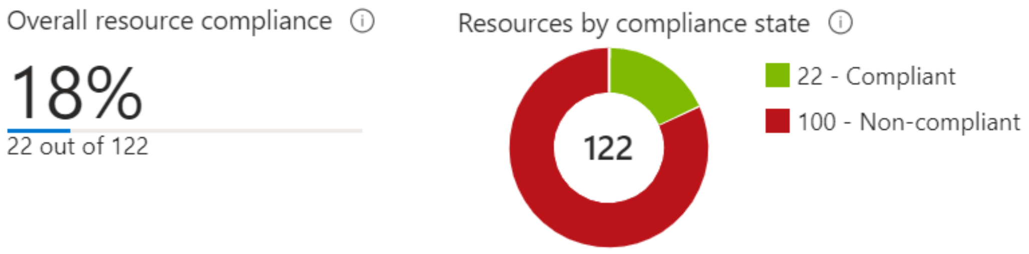
Step 9 : Azure policies

We can use service bus as queue for the incoming requests.

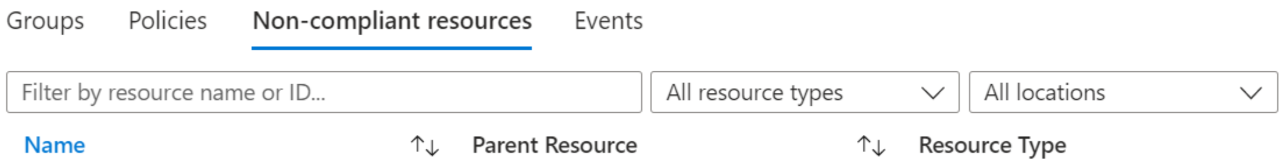
- In the search box at the top of the portal search policy
- Select subscription in the scope.



- You can verify the compliance score.


Overall resource compliance ⓘ
18%
22 out of 122
Resources by compliance state ⓘ
122
🟩 22 - Compliant
🟥 100 - Non-compliant

- Check the non-compliant resources in the section.



5.Knowledge Sharing and Best Practices:

- We can achieve upto 99.99999 percent High availabitlity using Azure service by using the concept of Zoning.
- We can host our health care service application using azure services like AKS,SQL,ACR and webapp which provides high availability and scalability.
- We can also enhance IOT devices to use azure services and host applications.
- Data is fully secured by using Encryption provided by Azure which saves overhead cost also.
- Use keyvault to secure secrets for the applications to be hosted in the pods.

6.Business Benefits:

- We have utilized 100% of Azure PaaS components in implementing solution to reduce the administrative (Scalability and Availability) overhead.
- Out of the box Azure Kubernetes Services helped in simplifying and deploying of microservices based connected healthcare solution.
- The insights from the solution will help us in identifying real time faults in healthcare devices and managing the device in efficient way.

7.Conclusion

In this blog I have demonstrated how you can use AKS to host multiple application and route the traffic from application gateway.

The solution is to migrate from legacy or other resource of Azure to new technology and resources which reduces manual efforts and also provide high availability and scalability.

The solution is a classic example of PAAS services implementation especially Azure Kubernetes Service (AKS).

8.References:

Azure Kubernetes cluster: -
https://learn.microsoft.com/en-us/azure/aks/intro-kubernetes

Azure application gateway:-
https://learn.microsoft.com/en-us/azure/app-service/overview