



# **NewgenONE Content Cloud**

## **Configuration and Deployment Guide for Azure**

**Version: 2024.1**

**Newgen Software Technologies Ltd.**

This document contains propriety information of NSTL. No part of this document may be reproduced, stored, copied, or transmitted in any form or by any means of electronic, mechanical, photocopying, or otherwise, without the consent of NSTL.

# Table of contents

<b>1</b>	<b>Preface</b>	<b>3</b>
1.1	Revision history	3
1.2	Intended audience	3
1.3	Documentation feedback	3
1.4	Third-party product information	3
<b>2</b>	<b>Configuring Azure Kubernetes cluster</b>	<b>4</b>
2.1	Creating an Azure Kubernetes cluster	4
2.2	Configuring Azure container registry	8
2.3	Configuring the ACR image scanning	9
2.4	Creating a storage account	11
2.4.1	Creating a BLOB storage	11
2.4.2	Creating an Azure file share	15
2.5	Configuring CDN service	17
2.6	Configuring Azure cache for redis	17
2.7	Adding the Kubeconfig file	19
2.8	Running Kubectl from local machine	19
2.9	Configuration of application gateway ingress controller	20
2.9.1	Creation of an application gateway	20
2.9.2	Installation of an application gateway ingress controller	28
2.9.3	Install Helm	28
2.9.4	ARM authentication using a service principle	28
2.9.5	Add or update Kubeconfig file	29
2.9.6	Install ingress controller using Helm	30
2.10	Configuring the DNS zone	33
2.11	Monitoring the Kubernetes dashboard	35
2.12	Azure monitor for container insights	35
2.13	Configuring the CosmosDB (MongoDB API)	36
2.14	Configuring Azure cognitive search	42
2.15	Configuring the Kafka(Azure Event Hubs)	46
2.16	Configuring the MSSQL database server	48
2.17	Configuring the SendGrid email service	49
<b>3</b>	<b>Deploying NCC containers</b>	<b>51</b>
3.1	Prerequisites	51
3.2	Overview	51
3.3	Deliverables	51
3.3.1	Docker images	52
3.3.2	YAML files	52
3.4	Changing Product's YAML files	55
3.5	Changes in Application Gateway Ingress YAML Files	57
3.6	Changes in Configmap.yaml file	59
3.7	Deploying containers	64
3.8	Configuring the Azure DevOps release pipeline (NCC)	65
3.9	Configuring the Azure DevOps	66

3.10	Configuring the release pipeline.....	67
3.11	Configuring the Dev stage .....	71
3.12	Configuring the UAT stage .....	74
3.13	Configuring the production stage.....	75
<b>4</b>	<b>Creating a new Azure WebApp service.....</b>	<b>78</b>
4.1	Configuring an Azure WebApp service .....	78
4.1.1	Application settings.....	78
4.1.2	General settings .....	79
4.1.3	Path Mappings .....	80
4.2	Creating Build of Web Application.....	80
4.3	Building NCC Microsite .....	80
4.4	Building NCC Micro UI .....	81
4.5	Building NCC Newgen Admin.....	82
4.6	Building NCC Tenant Admin .....	82
4.7	Deploying Web Application to App Service .....	82
4.8	Deploying NCC Microsite .....	83
4.9	Deploying NCC Micro UI .....	83
4.10	Deploying NCC Newgen Admin .....	84
4.11	Deploying NCC Tenant Admin.....	84
4.12	Deploying NCC Help Guides.....	84
4.13	Verifying Deployed Web Applications .....	84

# 1 Preface

This guide describes the deployment of NewgenONE Content Cloud (NCC) 2024.1 deliverables like Docker images and their required configuration files on the Azure Kubernetes Service (AKS).

## 1.1 Revision history

Revision Date	Description
July 2024	Initial publication

## 1.2 Intended audience

This guide is intended for Cloud Administrators, System Administrators, developers, and all other users who are seeking information on the NewgenONE Content Cloud Containers on Azure Kubernetes Service. The reader must be comfortable understanding the computer terminology.

## 1.3 Documentation feedback

To provide feedback or any improvement suggestions on technical documentation, write an email to [docs.feedback@newgensoft.com](mailto:docs.feedback@newgensoft.com).

To help capture your feedback effectively, share the following information in your email.

- Document name
- Version
- Chapter, topic, or section
- Feedback or suggestions

## 1.4 Third-party product information

This guide contains third-party product information about configuring Microsoft Azure CICD Pipeline for Container Deployment on AKS Azure Kubernetes Cluster. Newgen Software Technologies Ltd does not claim any ownership on such third-party content. This information is shared in this guide only for convenience of our users and could be an excerpt from the Azure documentation. For latest information on configuring the Azure Kubernetes Cluster and Azure DevOps refer to the Azure documentation.

## 2 Configuring Azure Kubernetes cluster

This section explains how to configure the Kubernetes Cluster on Azure.

### 2.1 Creating an Azure Kubernetes cluster

#### Prerequisites:

- Signed-in user must have the below roles:
  - At Subscription – Contributor Role
  - At User Access Administrator – Owner Role
- Virtual network and subnet must be created for the Kubernetes cluster.
- Before creating the Azure Kubernetes Cluster (AKS), sign in to the Azure portal at <https://portal.azure.com>

To create an Azure Kubernetes Cluster, perform the below steps:

1. On the Azure portal menu (Home page), select **Create a resource**.

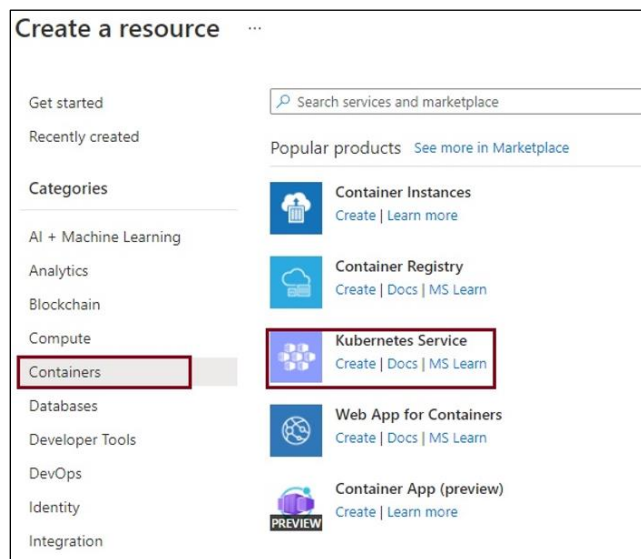


Figure 2.1

2. Select **Containers** and then **Kubernetes Service**. The Create Kubernetes cluster dialog appears.

**Create Kubernetes cluster**

[Basics](#) [Node pools](#) [Access](#) [Networking](#) [Integrations](#) [Advanced](#) [Tags](#) [Review + create](#)

Azure Kubernetes Service (AKS) manages your hosted Kubernetes environment, making it quick and easy to deploy and manage containerized applications without container orchestration expertise. It also eliminates the burden of ongoing operations and maintenance by provisioning, upgrading, and scaling resources on demand, without taking your applications offline. [Learn more](#)

**Project details**

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

Subscription \*

Resource group \*  [Create new](#)

**Cluster details**

Cluster preset configuration    
 To quickly customize your Kubernetes cluster, choose one of the preset

Kubernetes cluster name \*

Region \*

Availability zones    
  High availability is recommended for standard configuration.

AKS pricing tier

Kubernetes version \*

Automatic upgrade

**Primary node pool**

The number and size of nodes in the primary node pool in your cluster. For production workloads, at least 3 nodes are recommended for resiliency. For development or test workloads, only one node is required. If you would like to add additional node pools or to see additional configuration options for this node pool go to the 'Node pools' tab above. You will be able to add additional node pools after creating your cluster. [Learn more about node pools in Azure Kubernetes Service](#)

Node size \*    
  Standard DS2\_v2 is recommended for standard configuration.   
 [Change size](#)

Scale method \*  Manual   
  Autoselect   
  Autoscaling is recommended for standard configuration.

Node count range \*

**Figure 2.2**

3. Under the **Basics** tab, specify the following:

Fields	Description
Subscription	Select a valid Azure subscription.
Resource group	Select or create an Azure Resource group, such as NCC-PT.
Cluster preset configuration	Production Standard
Kubernetes cluster name	Enter a Kubernetes cluster name such as NCC-PT-Cluster.
Region	Select a region into which you want to create an AKS cluster.
Availability zones	There are three availability zones per region that allow us to spread the nodes across different physical locations for high availability. Select the availability zones as per your business requirement. [By default, select all the availability zones].
Kubernetes version	Select the default that is, 1.27.9 (default).
Primary node pool	Select a VM Node size for the AKS nodes and select the number of nodes to be deployed into the AKS cluster. <b>NOTE:</b> The VM size remains the same after the AKS cluster deployment. However, you

Fields	Description
	can adjust the node count.
Scale method	Select the scale method as Autoscale. Autoscaling can help ensure that your cluster is running efficiently with the right number of nodes for the workloads present.

4. Under the **Node** tab, keep the default options and then click **Next: Authentication**. The Authentication page appears.
5. Keep the default options and click **Next: Networking**. The Networking page appears.
6. Select the **Azure CNI** as Network configuration and specify the following options:
  - i. **Virtual network** – Select the created VNet for this AKS cluster deployment that is, **Vnet\_for\_NCC\_PT\_RG**.
  - ii. **Cluster subnet** – Select the subnet into which place both the nodes and containers in the cluster, that is, subnet\_dev (**10.0.0.0/22**).
  - iii. **Kubernetes service address range** – Specify the IP range from which to assign IPs to the internal Kubernetes services. This range must not be connected to this virtual network, or it must not overlap with any Subnet IP ranges. For example: **10.1.0.0/16**. You can reuse this range across different AKS clusters.
  - iv. **Kubernetes DNS service IP address** – An IP address assigned to the Kubernetes DNS service. It must be within the Kubernetes service address range. For example: **10.1.0.10**.

---

**NOTE:**

Avoid using the first IP address in your address range. The first address is used for the `kubernetes.default.svc.cluster.local` address.

---

- v. **Docker Bridge address** – Docker bridge is not used by AKS clusters or the pods themselves, you must set this address to continue to support scenarios such as docker build within the AKS cluster. It is required to select a CIDR for the Docker bridge network address because otherwise, Docker picks a subnet automatically, which conflicts with other CIDRs. You must pick an address space that does not collide with the rest of the CIDRs on your networks, including the cluster's service CIDR and pod CIDR that is, **172.17.0.1/25**. You can reuse this range across different AKS clusters.
- vi. **At the bottom of the screen, select Azure as Network policy** and keep the other settings as default.
7. On the Integration page, keep the default options. At the bottom of the screen, click **Next: Tags**.
8. On the Tags page, keep the default options. At the bottom of the screen, click **Next: Review + create**.
9. On the **Review + create** page, click **Create** once validation passed.
10. Once deployment is complete, click **Go to resource** button.

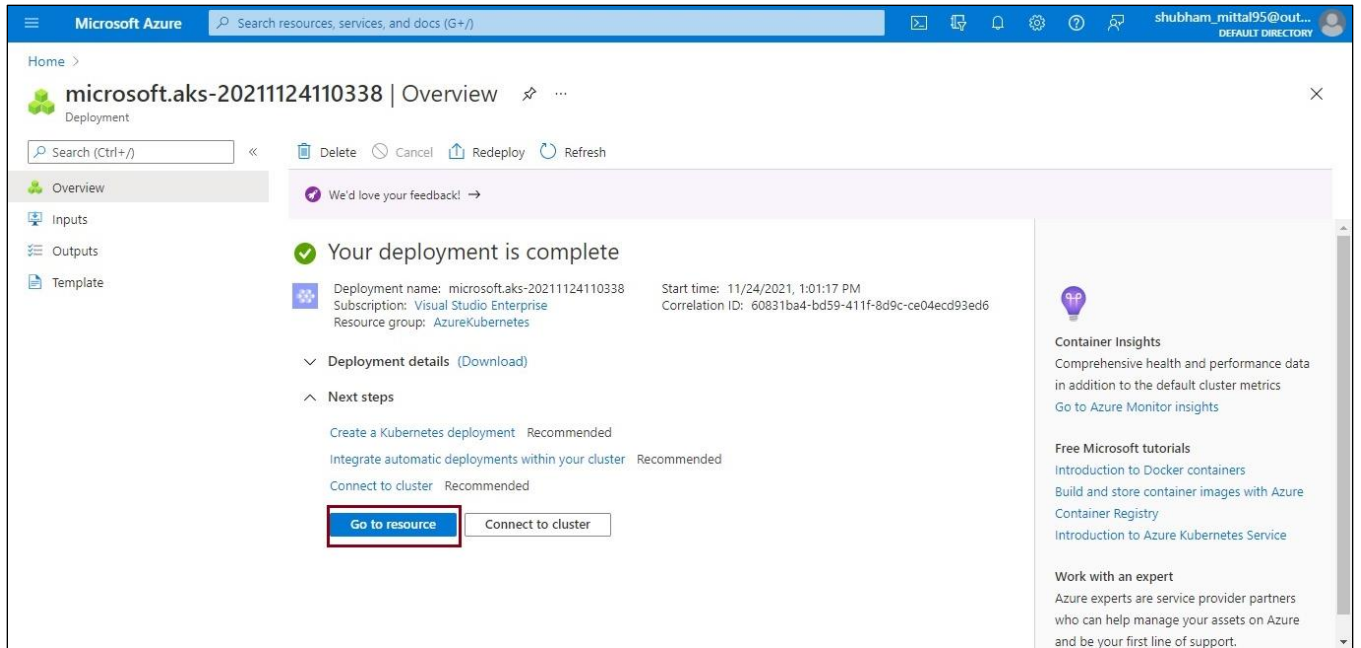


Figure 2.3

The Azure Kubernetes Cluster dashboard appears.

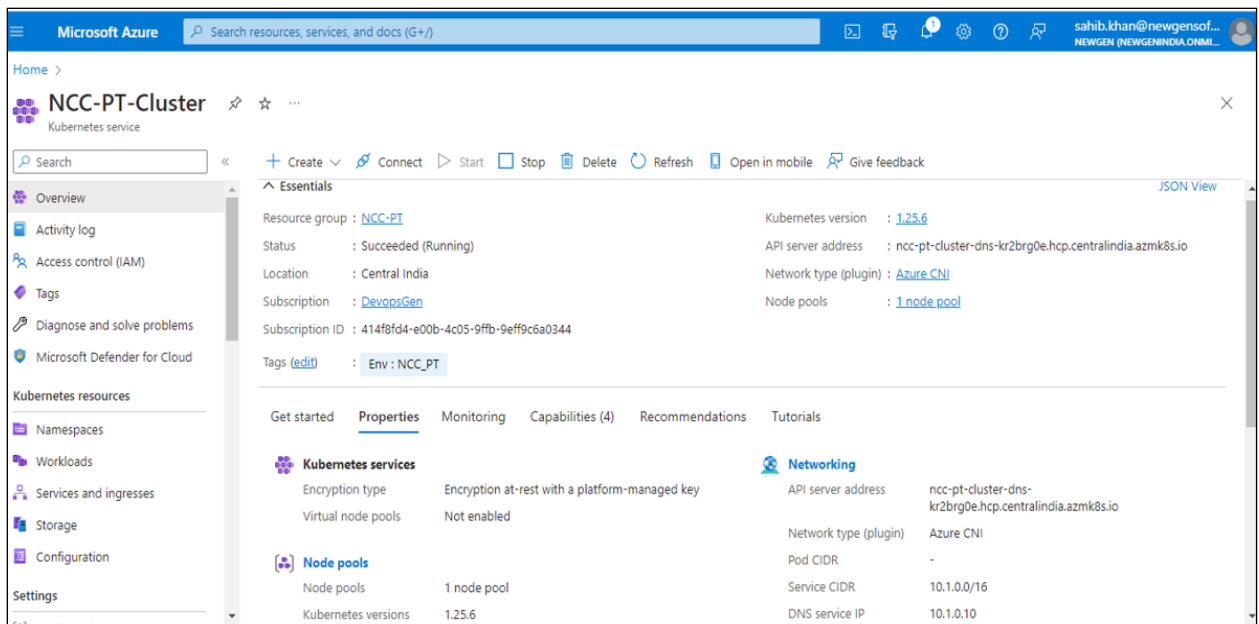


Figure 2.4



## 2.2 Configuring Azure container registry

To configure the Azure Container Registry, perform the below steps:

1. Sign in to the Azure Portal using the below URL:  
<https://portal.azure.com/>
2. After a successful sign in, select 'Create a resource' > Containers > Container Registry.
3. In the Basic tab, specify the following details:
  - Resource group: Select the existing resource group or create a new one, that is, AzureKubernetes.
  - Registry name: Specify the user-defined name that is, newgencontainerregistry.
  - Location: Select the location. For example, UAE North and more.
  - SKU: Choose the SKU based on your usage as Basic, Standard, or Premium. Each SKU carry a different storage size. For example,

SKU	Storage Limit
Basic	10 GiB
Standard	100 GiB
Premium	500 GiB or more

4. Accept default values for the remaining settings. Then select Review + create. After reviewing the settings, select Create.
5. When the Deployment succeeded message appears, select the container registry in the portal.
6. Click Access keys from Settings appears on the left pane.
7. Enable Admin user.

---

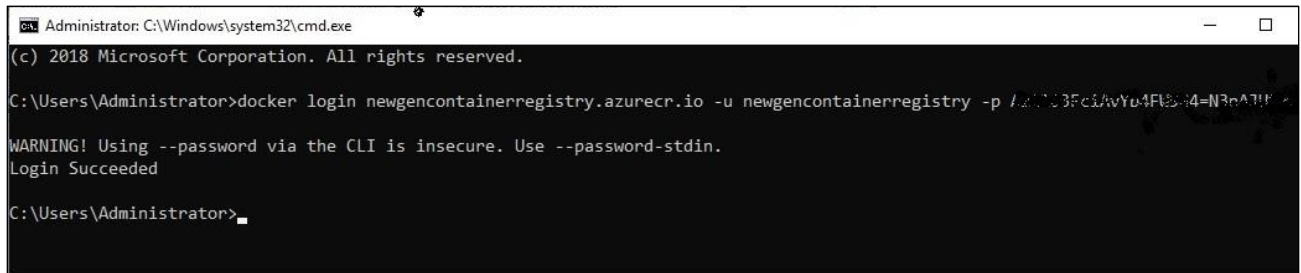
**NOTE:**

Keep the Login server, Username, and password (or password2). These values are required to push or pull Docker images.

---

8. Use the below command to connect to the created container registry from your local machine (Where Docker Engine is already installed):  
`docker login <Container Registry Login server> -u <Container Registry username> -p <Container Registry password>`

For example,



```
Administrator: C:\Windows\system32\cmd.exe
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>docker login newgencontainerregistry.azurecr.io -u newgencontainerregistry -p /Z11C3Fcj/vvYd4FL8j4=N3eA7Ff
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded

C:\Users\Administrator>
```

Figure 2.5

9. After a successful sign in to the Container Registry, use the below command to tag and push the Docker images from your local machine to ACR (Azure Container Registry):

```
docker tag <image name>:<image tag> <container registry server>/<image
name>:<image tag>
docker push <container registry server>/<image name>:<image tag>
```

For Example,

```
docker tag ibps5serviceinstanceweb:sp2
newgencontainerregistry.azurecr.io/ibps5serviceinstanceweb:sp2
docker push newgencontainerregistry.azurecr.io/ibps5serviceinstanceweb:sp2
```

10. Use the below command to pull the Docker images from ACR to your local machine:

```
docker pull <container registry server>/<image name>:<image tag>
```

For Example,

```
docker pull newgencontainerregistry.azurecr.io/ibps5serviceinstanceweb:latest
```

## 2.3 Configuring the ACR image scanning

To configure ACR Image Scanning, perform the below steps:

1. ACR image scanning is done by **Microsoft Defender for Cloud**.
2. Once the image scanning is configured and whenever a Docker image is pushed to the Azure Container Repository, Microsoft Defender for Cloud automatically scans that Docker image. So, to trigger the scan of an image, it is mandatory to push that image in ACR.
3. Make sure the Defender plan is enabled for the **Container registries**.
4. Go to the **Microsoft Defender for Cloud** page > Click **Environment settings** under **Management** > Click the listed subscription.

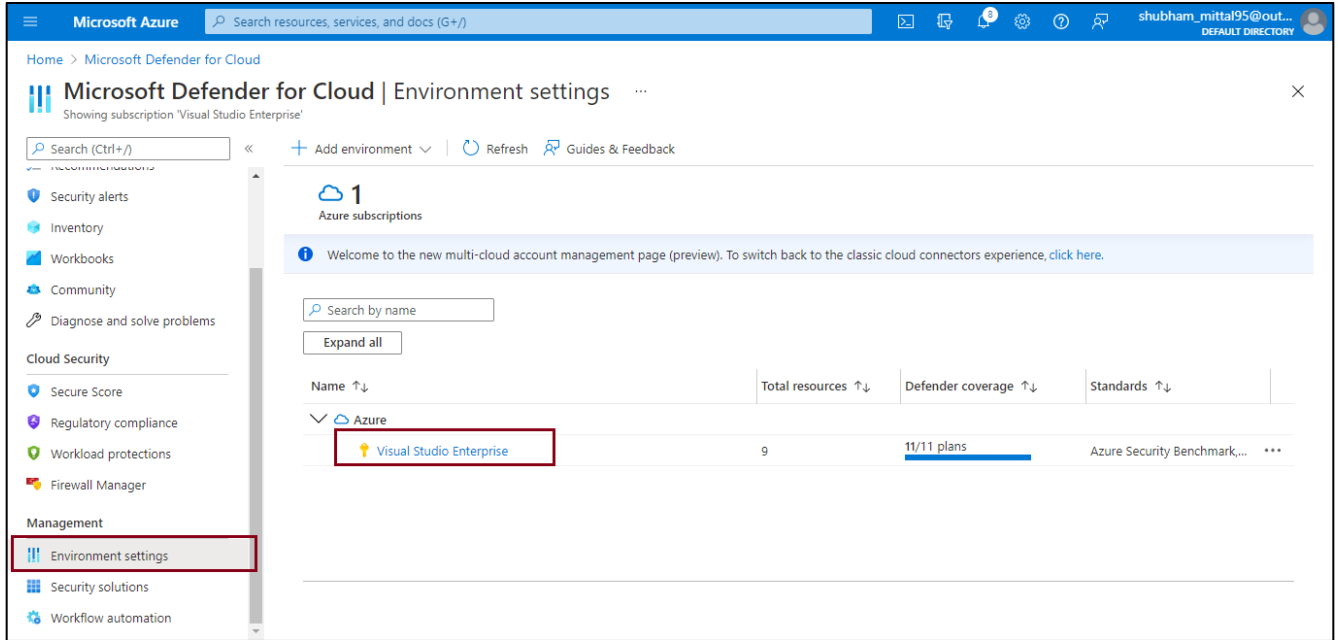


Figure 2.6

5. Go to the bottom of the page and enable the **Container registries** defender plan if it is not already enabled.

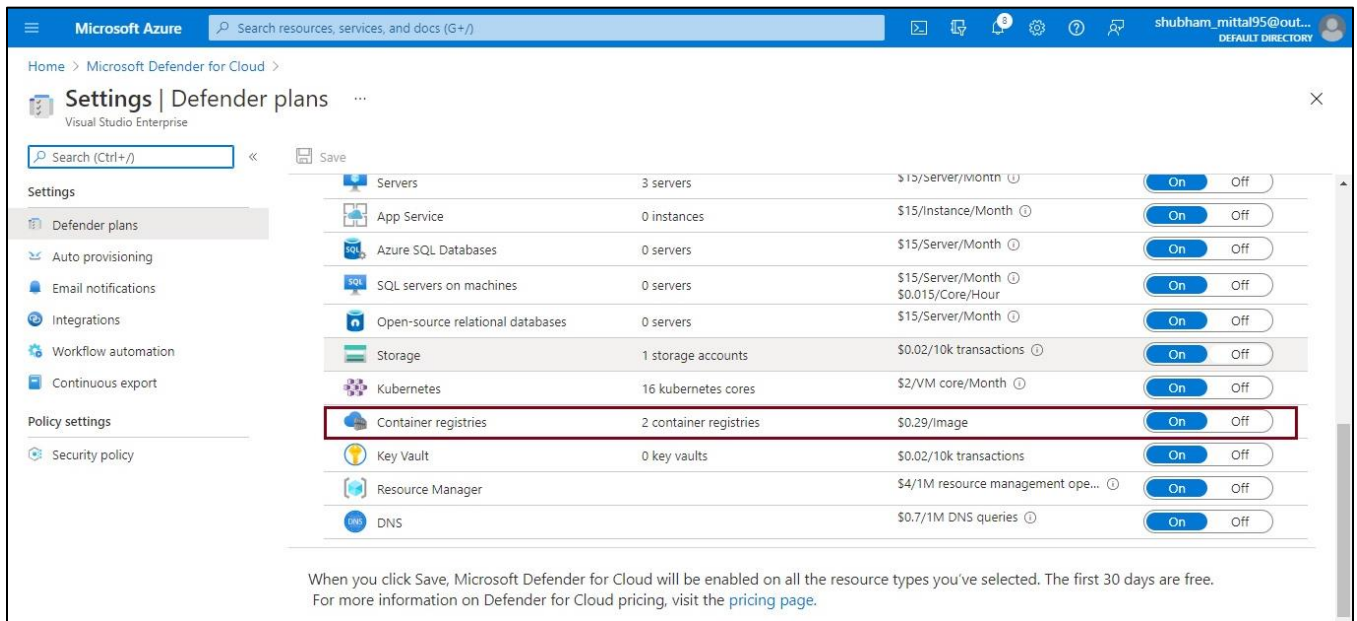


Figure 2.7

6. Microsoft Defender for container registries includes a vulnerability scanner to scan the images in your Azure Resource Manager-based Azure Container Registry registries and provide deeper visibility into your images' vulnerabilities. The integrated scanner is powered by Qualys, the industry-leading vulnerability scanning vendor.
7. When issues are found – by Qualys or Defender for Cloud – you'll get notified in the workload protection dashboard.

## 2.4 Creating a storage account

This section explains how to configure a storage account.

### 2.4.1 Creating a BLOB storage

To create a BLOB storage, perform the below steps:

1. Sign in to the Azure Portal using the below URL:  
<https://portal.azure.com/>
2. On the Azure portal menu, select **All services** > Select **Storage Accounts**.
3. On the Storage Accounts window that appears, click **Create**.
4. On the **Basics** tab, select an active Azure subscription.
5. Under the Resource group field, select your desired resource group, or create a new resource group like **NCCPT**.
6. Next, enter a name for your storage account like **nccblobstoragetest**.
7. Select a **location or region** in which you want to create your storage account that is, UAE North.
8. Select a performance tier. Choose Premium performance tier.
9. Specify the storage account replication process. The default replication option for Premium is Locally-redundant storage (LRS).
10. Keep the other settings as default.
11. Click the **Next: Advanced**.

Home > Storage accounts >

## Create a storage account

Basics | Advanced | Networking | Data protection | Encryption | Tags | Review

### Project details

Select the subscription in which to create the new storage account. Choose a new or existing resource group to organize and manage your storage account together with other resources.

Subscription \*

Resource group \*  [Create new](#)

### Instance details

Storage account name ⓘ \*

Region ⓘ \*

Performance ⓘ \*  Standard: Recommended for most scenarios (general-purpose v2 account)  
 Premium: Recommended for scenarios that require low latency.

Premium account type ⓘ \*

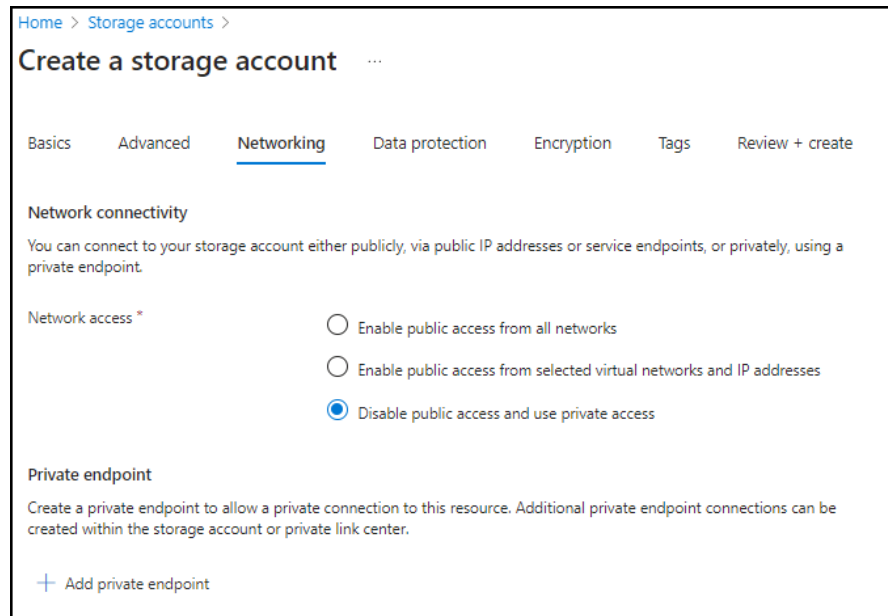
Redundancy ⓘ \*

[Review](#) [< Previous](#) [Next : Advanced >](#)

Figure 2.8

- On the **Advanced** page, keep the default options. At the bottom of the screen, click **Next: Networking**.

13. On the 'Networking' page, choose Network access as 'Disable public access and use private access'.



Home > Storage accounts >

## Create a storage account

Basics   Advanced   **Networking**   Data protection   Encryption   Tags   Review + create

### Network connectivity

You can connect to your storage account either publicly, via public IP addresses or service endpoints, or privately, using a private endpoint.

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

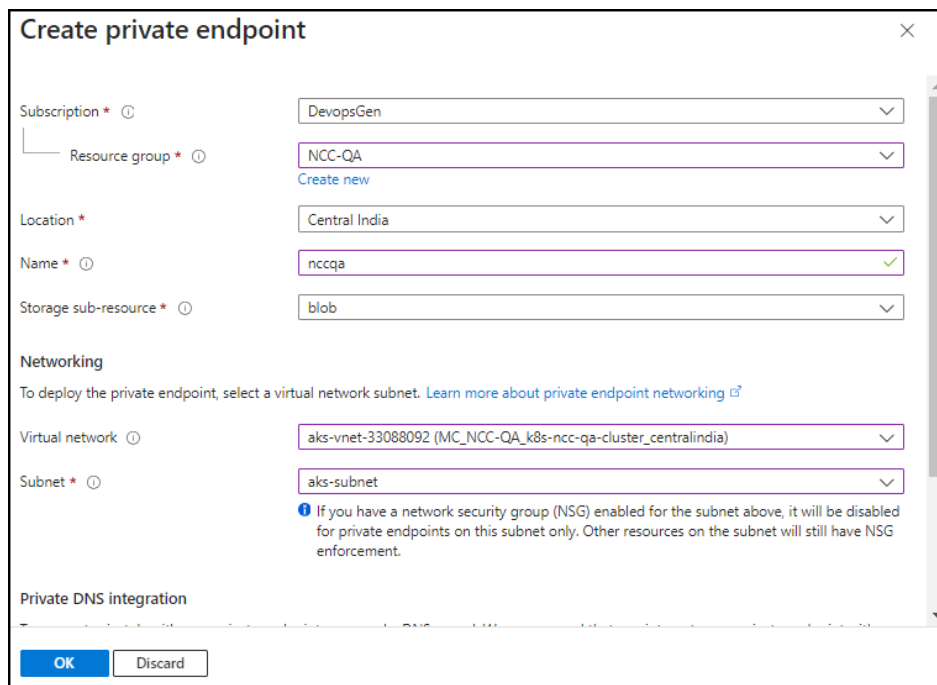
### Private endpoint

Create a private endpoint to allow a private connection to this resource. Additional private endpoint connections can be created within the storage account or private link center.

+ Add private endpoint

Figure 2.9

14. Add a Private Endpoint using the '+' button. Enter relevant details and use the same Virtual Network which was created during Kubernetes Cluster creation. Click **OK**.



### Create private endpoint

Subscription \*

Resource group \*   
[Create new](#)

Location \*

Name \*

Storage sub-resource \*

#### Networking

To deploy the private endpoint, select a virtual network subnet. [Learn more about private endpoint networking](#)

Virtual network

Subnet \*

**i** If you have a network security group (NSG) enabled for the subnet above, it will be disabled for private endpoints on this subnet only. Other resources on the subnet will still have NSG enforcement.

Private DNS integration

Figure 2.10

15. At the bottom of the screen, click the "Next: Data protection>".

16. On the 'Data protection' page, keep the default options. At the bottom of the screen, click the "Next: Tags>".
17. On the 'Tags' page, keep the default options. At the bottom of the screen, click the "Next: Review + create".
18. On the 'Review + create' page, click the **Create** once validation passed.
19. Once deployment is complete, click the 'Go to resource' button.
20. Click the **Access keys** from Settings under Security + Networking appears on the left pane.

**NOTE:**

Keep the Storage account name and key1 (or key2) as these values is required for the Kubernetes volume mounting.

21. Click **Containers** under the Data storage appears on the left pane.
22. On the **Containers** window that appears, Click **+Container**.
23. On the 'New container' panel, specify the following details:
  - **Name:** Specify the unique blob storage name.
  - **Public access level:** Choose default "Private (no anonymous access)".
24. Click **Create**. The Container gets created.
25. After created storage account, click on the **Resource sharing (CORS)** under Settings, showing in the left panel of the screen.
26. Use \* in **Allowed origins**, **Allowed headers**, **Exposed headers** and **Max age** must be 86400.
27. Check all the methods in **Allowed methods** and **Save** the changes.

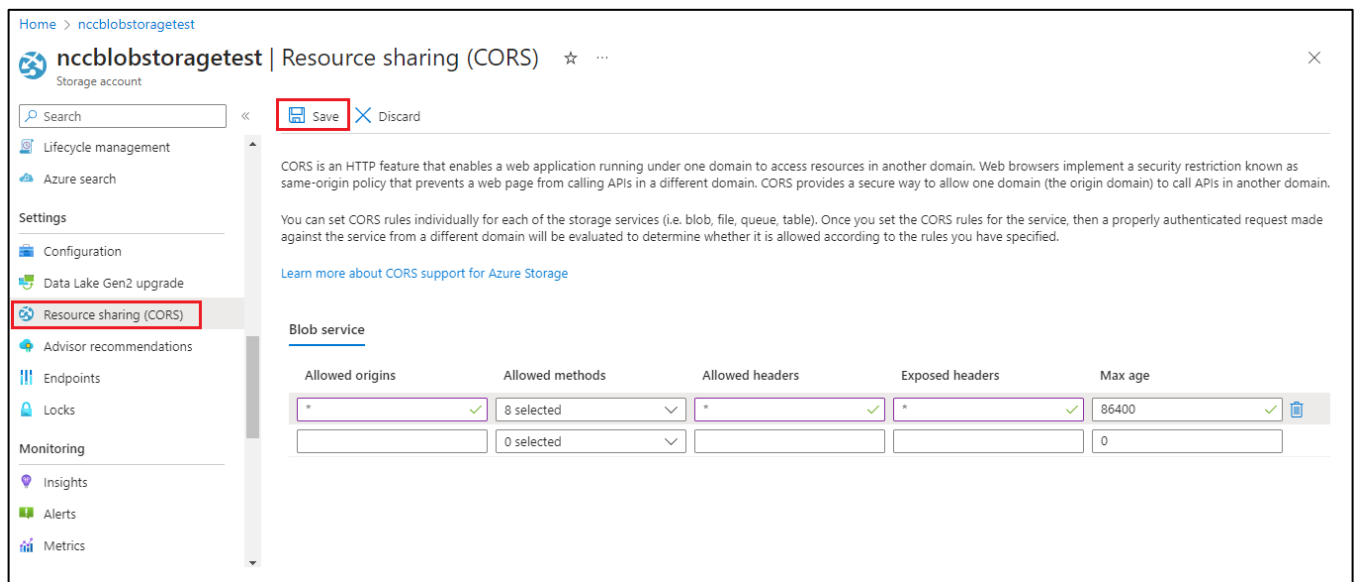
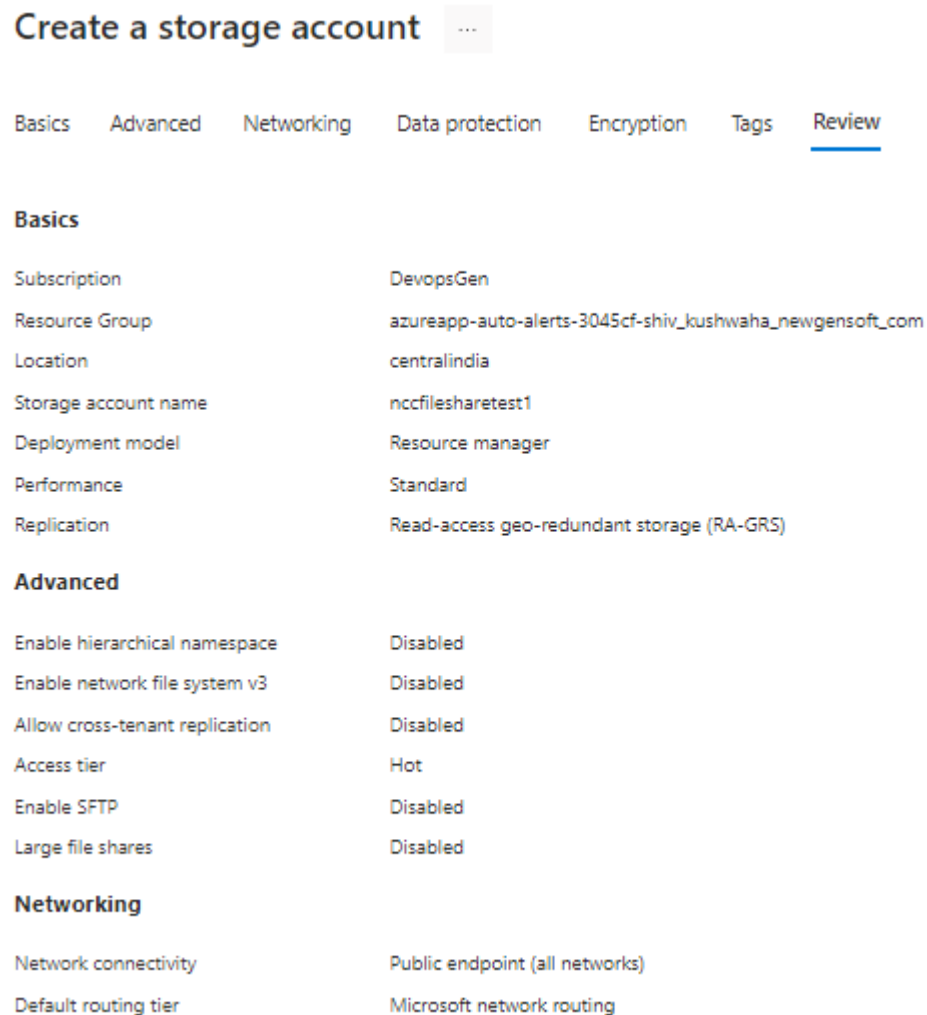


Figure 2.11

## 2.4.2 Creating an Azure file share

To create an Azure file share, perform the below steps:

1. Create a new storage account for File Share.



The screenshot shows the 'Create a storage account' wizard in the Azure portal, with the 'Review' tab selected. The wizard is divided into three sections: Basics, Advanced, and Networking. The Basics section includes fields for Subscription (DevopsGen), Resource Group (azureapp-auto-alerts-3045cf-shiv\_kushwaha\_newgensoft\_com), Location (centralindia), Storage account name (nccfilesharetest1), Deployment model (Resource manager), Performance (Standard), and Replication (Read-access geo-redundant storage (RA-GRS)). The Advanced section includes fields for Enable hierarchical namespace (Disabled), Enable network file system v3 (Disabled), Allow cross-tenant replication (Disabled), Access tier (Hot), Enable SFTP (Disabled), and Large file shares (Disabled). The Networking section includes fields for Network connectivity (Public endpoint (all networks)) and Default routing tier (Microsoft network routing).

Section	Property	Value
Basics	Subscription	DevopsGen
	Resource Group	azureapp-auto-alerts-3045cf-shiv_kushwaha_newgensoft_com
	Location	centralindia
	Storage account name	nccfilesharetest1
	Deployment model	Resource manager
	Performance	Standard
	Replication	Read-access geo-redundant storage (RA-GRS)
Advanced	Enable hierarchical namespace	Disabled
	Enable network file system v3	Disabled
	Allow cross-tenant replication	Disabled
	Access tier	Hot
	Enable SFTP	Disabled
	Large file shares	Disabled
Networking	Network connectivity	Public endpoint (all networks)
	Default routing tier	Microsoft network routing

Figure 2.12

2. Click **Overview** of the created storage account > Click **File shares** under the Data storage appears on the left pane.



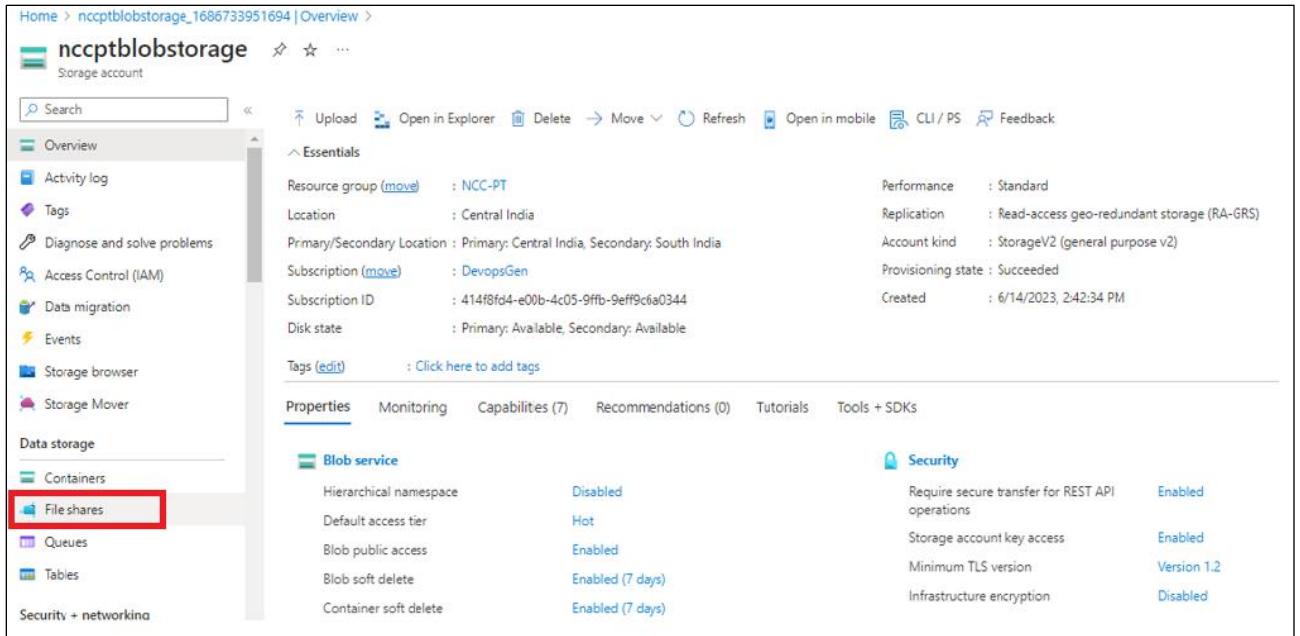
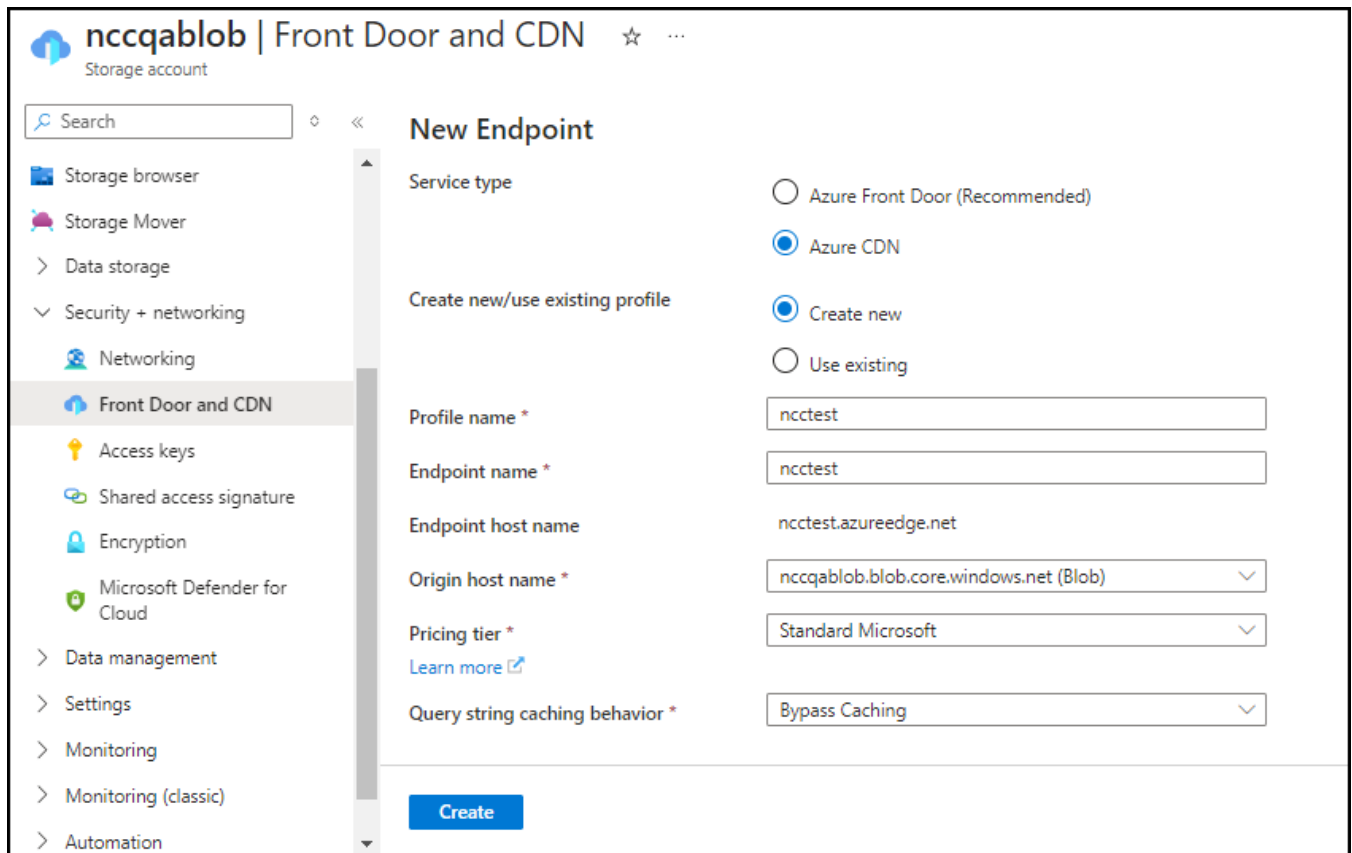


Figure 2.13

3. On the **File shares** window that appears, Click **+File share**.
4. On the **New File share** panel, specify the following details:
  - **Name:** Specify the unique file share name.
  - **Tiers:** Choose the 'Transaction optimized' as tier.
  - Click the **Create**.

## 2.5 Configuring CDN service

For the Storage created account, Click the **Front Door and CDN** under the Security + Networking appears on the left pane. In that, Choose **Service type** as 'Azure CDN', **Pricing Tier** as 'Standard Microsoft' and **Query string caching behavior** as 'Bypass Caching' and click **Create**.



The screenshot shows the 'New Endpoint' configuration page in the Azure Portal. The left navigation pane is open, and 'Front Door and CDN' is selected. The main content area displays the following configuration options:

- Service type:**  Azure Front Door (Recommended),  Azure CDN
- Create new/use existing profile:**  Create new,  Use existing
- Profile name \*:** nctest
- Endpoint name \*:** nctest
- Endpoint host name:** nctest.azureedge.net
- Origin host name \*:** nccqblob.blob.core.windows.net (Blob)
- Pricing tier \*:** Standard Microsoft
- Query string caching behavior \*:** Bypass Caching

A blue 'Create' button is located at the bottom of the form.

Figure 2.14

## 2.6 Configuring Azure cache for redis

Azure Cache for Redis provides fully managed open-source Redis within Azure that can be used as a distributed data or content cache, a session store and, more. It provides an in-memory data store.

To configure the Azure Cache for Redis, perform the below steps:

1. Sign in to the Azure Portal using the below URL:  
<https://portal.azure.com/>
2. On the Azure portal menu or from the Home page, select **Create a resource**.
3. Select the **Databases > Azure Cache for Redis**.
4. Specify the following details under the Basics tab:
  - **Subscription:** Select a valid Azure subscription.
  - **Resource group:** select or create an Azure Resource group, such as AzureKubernetes.

- DNS name: Enter a Redis cache DNS name such as azrediscache.
- Location: Select a region into which you want to create an Azure Cache for Redis.
- Cache type: Select the Redis cache service tier as per your requirement. You can choose from 250 MB to 1455 GB in-memory cache.
- Click the “Next: Networking>”.

5. On the ‘Networking’ page, choose the connectivity method as ‘Private Endpoint’.

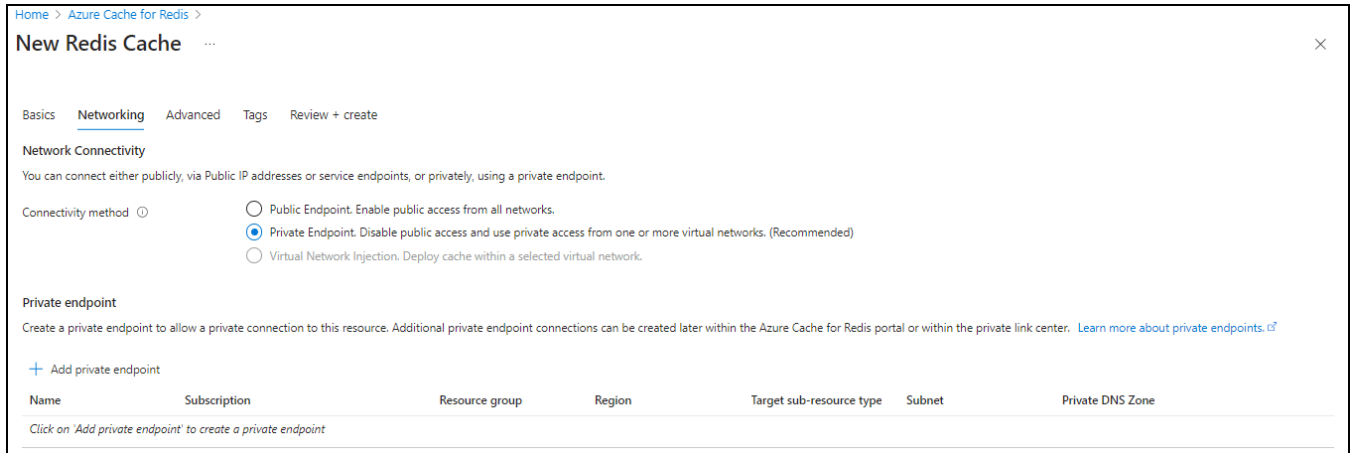


Figure 2.15

6. Create a Private Endpoint using the ‘+’ button. Enter relevant details and use the same Virtual Network which was created during Kubernetes Cluster creation. Click **OK**.

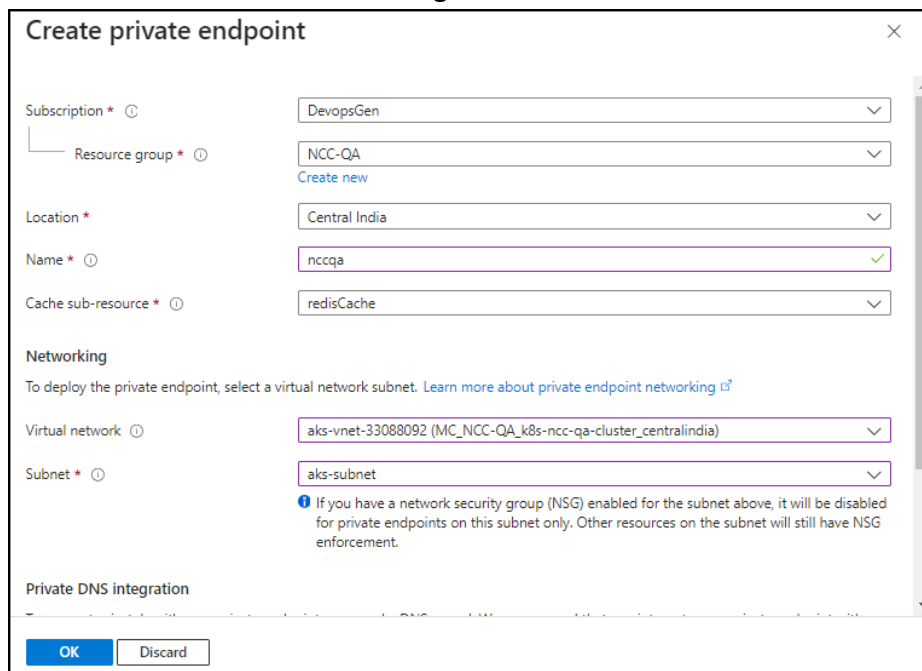


Figure 2.16

7. Then click the “Next: Advanced>” at the bottom of the screen.
6. On the ‘Advanced’ page, enable the Non-TLS port, select the Redis version as 6 and click the “Next: Tags>” at the bottom of the screen.
7. On the ‘Tags’ page, keep the default options. At the bottom of the screen, click the “Next: Review + create>”.
8. On the ‘Review + create’ page, click the Create once validation passed.
9. Once deployment is complete, click the Go to resource button.

## 2.7 Adding the Kubeconfig file

To add or update the kubeconfig file, perform the below steps:

- Open the Azure Cloud Shell <https://portal.azure.com/#cloudshell/> .
- Delete the **.kube/config** file (if already exists) using below command:  
`rm .kube/config`
- Now execute the below command to re-create **.kube/config** file:  
`az aks get-credentials --resource-group <ResourceGroupName> --name <AzureEKSClusterName> --admin`  
**For example,**  
`az aks get-credentials --resource-group NCC-PT --name NCC-PT-Cluster --admin`

## 2.8 Running Kubectl from local machine

**Prerequisites:**

- kubectl: <https://kubernetes.io/docs/tasks/tools/install-kubectl/>
- azure-cli: <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?tabs=azure>

To run kubectl from local machine, perform the below steps:

- Delete the **.kube** folder from C:\Users\<Logged-in Username> folder if exists.
- Execute the below command to worker node:  

```
az aks get-credentials --resource-group <ResourceGroupName> --name <AzureEKSClusterName>
```

**For example,**  

```
az aks get-credentials --resource-group NCC-PT --name NCC-PT-Cluster
```
- Once you have run the above command to connect to the AKS cluster, you can run any kubectl commands. Here are a few examples of useful commands:  
**For example,**  

```
# List all the pods
kubectl get pods

# List all deployments in all namespaces
```

```
kubectl get deployments --all-namespaces=true

# List all deployments in a specific namespace
# Format :kubectl get deployments --namespace <namespace-name>
kubectl get deployments --namespace kube-system
```

## 2.9 Configuration of application gateway ingress controller

This section explains how to configure Application Gateway Ingress Controller.

### 2.9.1 Creation of an application gateway

#### Pre-requisites:

- A subnet must be created in the same virtual network in which the Kubernetes cluster exists.

Perform the below steps to create an Application Gateway:

1. On the Azure portal menu or from the Home page, select **Create a resource**.
2. Select **Networking**.

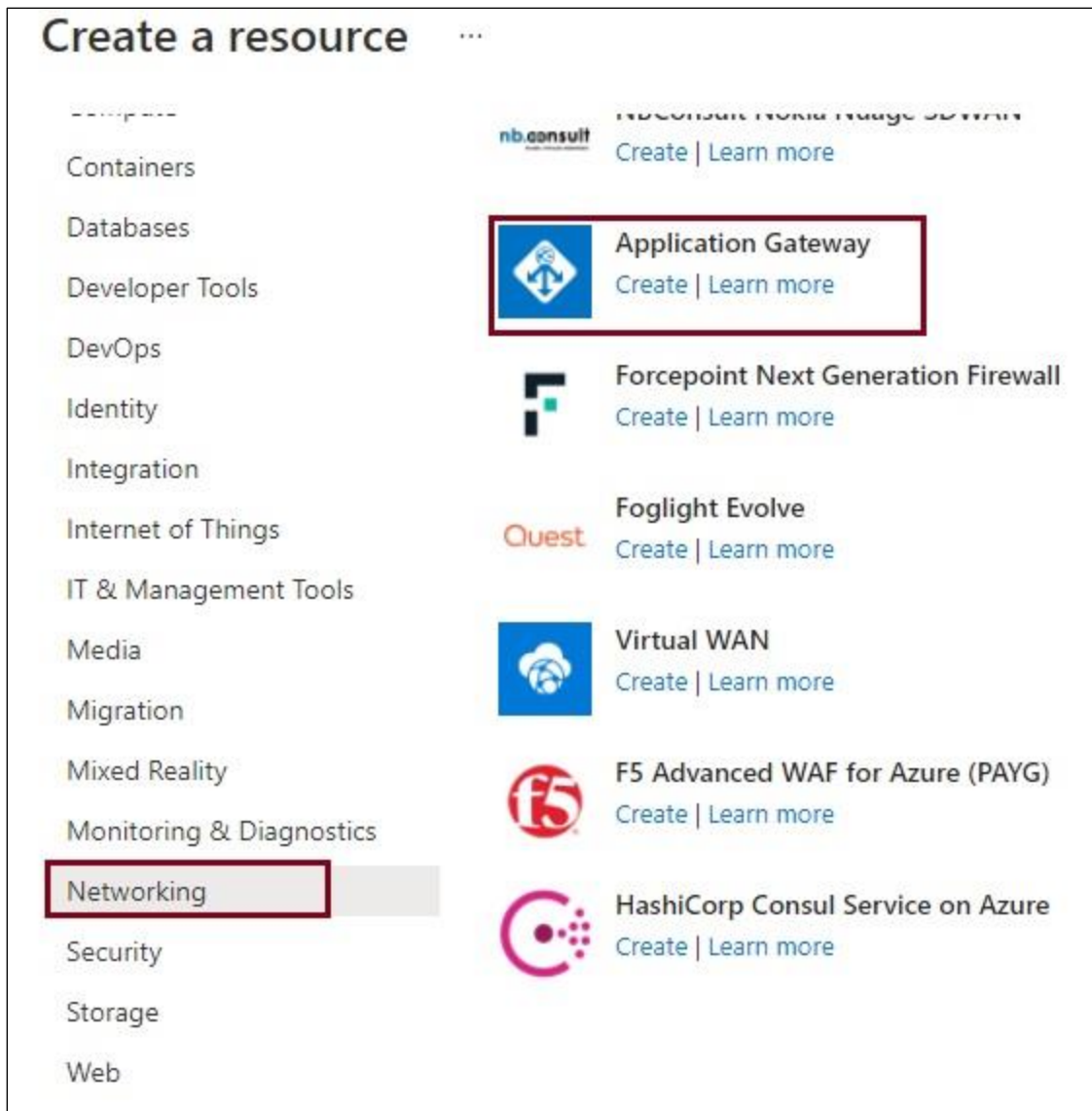


Figure 2.17

3. Select **Application Gateway**. The **Create application gateway** screen appears.
4. Specify the following details under the **Basics** tab:
  - **Subscription**: Select a valid Azure subscription.
  - **Resource group**: Select or create an Azure Resource group, such as **AzureKubernetes**.
  - **Application gateway name**: Enter a Kubernetes cluster name such as **AppGateway-AKSCluster**.
  - **Region**: Select a region into which you want to create an AKS cluster that is, UAE North
  - **Tier**: Select Standard V2.
  - **Virtual network**: Select the same virtual network in which the Kubernetes cluster exists.
  - **Subnet**: Select the created subnet for the application gateway.
  - Keep the other settings as default and then select the **Next: Frontends**.

## Create application gateway ...

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

An application gateway is a web traffic load balancer that enables you to manage traffic to your web application. [Learn more about application gateway](#)

### Project details

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

Subscription \* ⓘ

Resource group \* ⓘ 
  
[Create new](#)

### Instance details

Application gateway name \*  ✓

Region \*

Tier ⓘ

Enable autoscaling  Yes  No

Minimum instance count \* ⓘ

Maximum instance count

Availability zone ⓘ

HTTP2 ⓘ  Disabled  Enabled

### Configure virtual network

Virtual network \* ⓘ 
  
[Create new](#)

Subnet \* ⓘ 
  
[Manage subnet configuration](#)

Figure 2.18

5. Set the Frontend IP address type as **Public**.

6. Select **Add new** for the **Public IP address** and enter a user-defined name that is, *appgwpublicip* and then click **OK**.

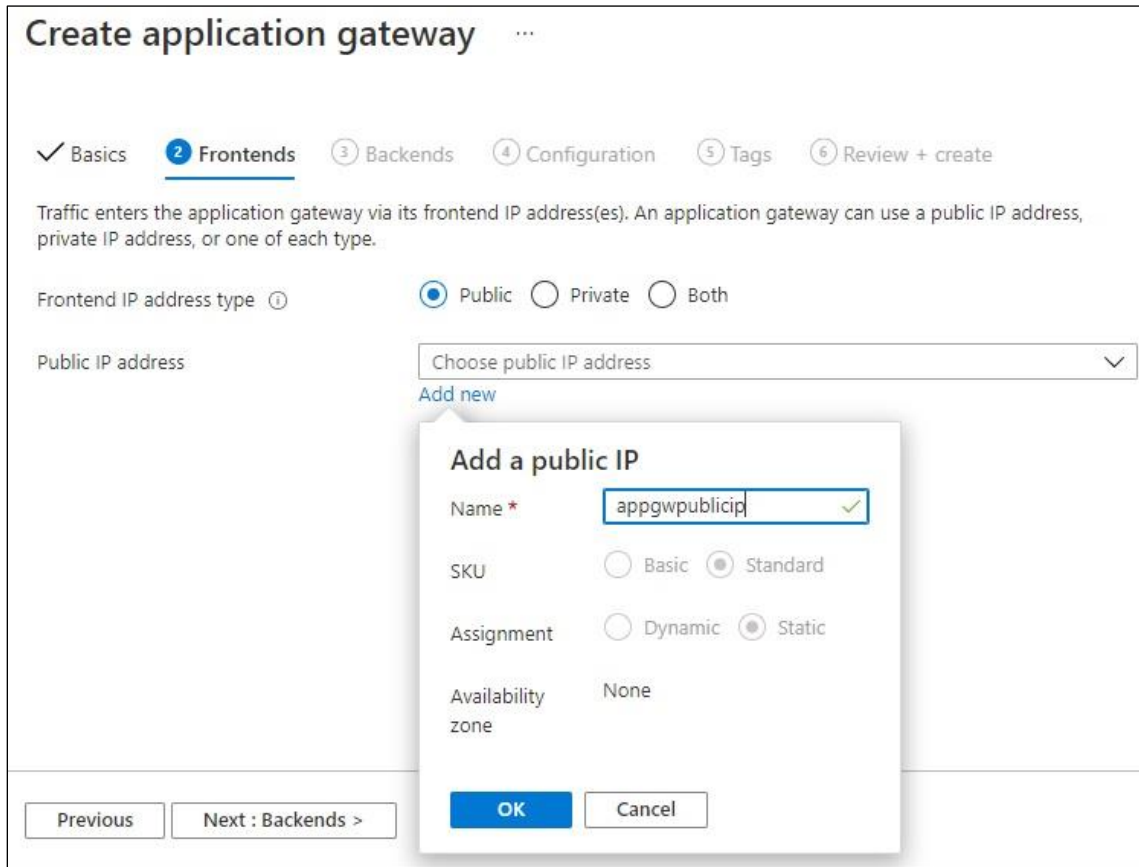


Figure 2.19

7. Select **Next: Backends**. The **Backends** tab appears.
8. Select **add a backend pool**. The Add a backend pool dialog appears.
9. Enter the following details to create an empty backend pool:
  - **Name**: Enter a user-defined name that is, **appgwbackendpool**.
  - **Add backend pool without targets**: Select **Yes** to create a backend pool with no targets.
  - Select **Add** to save the backend pool configuration and return to the **Backends** tab.



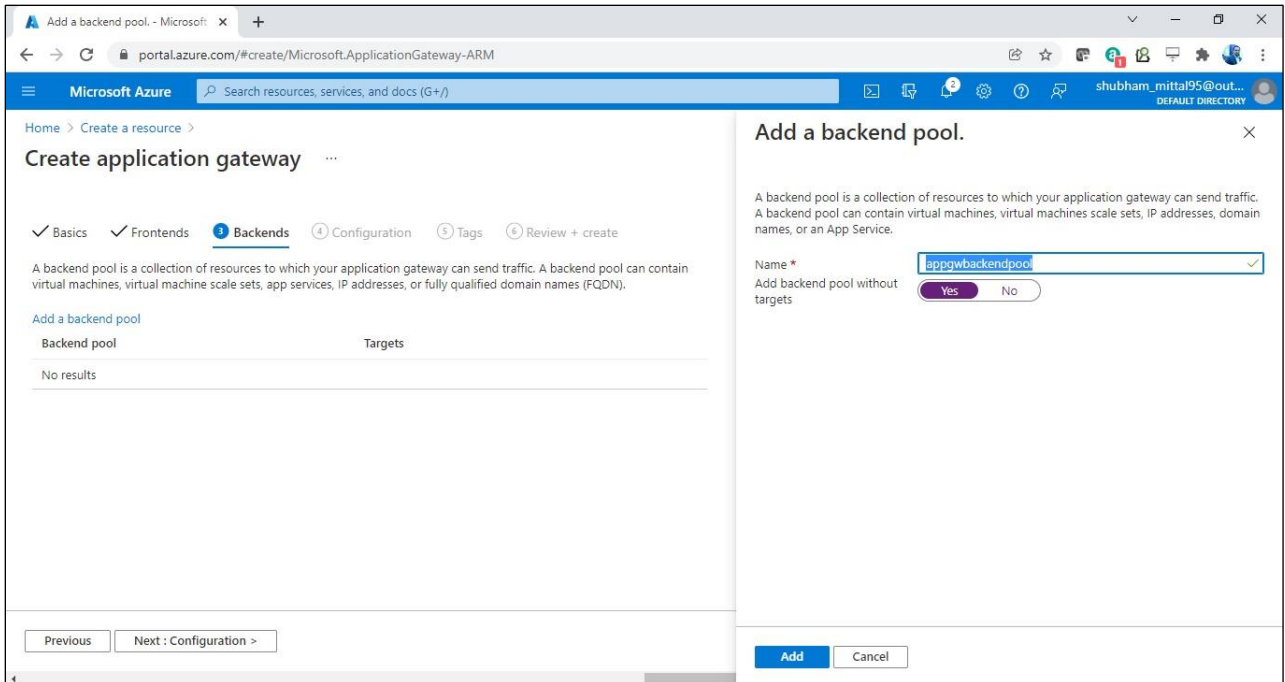


Figure 2.20

10. On the **Backends** tab, select **Next: Configuration**. The Configuration tab appears.
11. Select **Add a routing rule** in the **Routing rules** column. The Add a routing rule dialog appears.
12. Enter the user-defined rule name that is, **routingrule1**.
13. A routing rule requires a listener. On the **Listener** tab, enter the following details:
  - **Listener name**: Enter a user-defined listener name that is, **appgwlistener**.
  - **Frontend IP**: Select Public to select the public IP that you have created in the Frontends tab.
  - Keep the other settings as default and switch to the **Backend targets** tab.

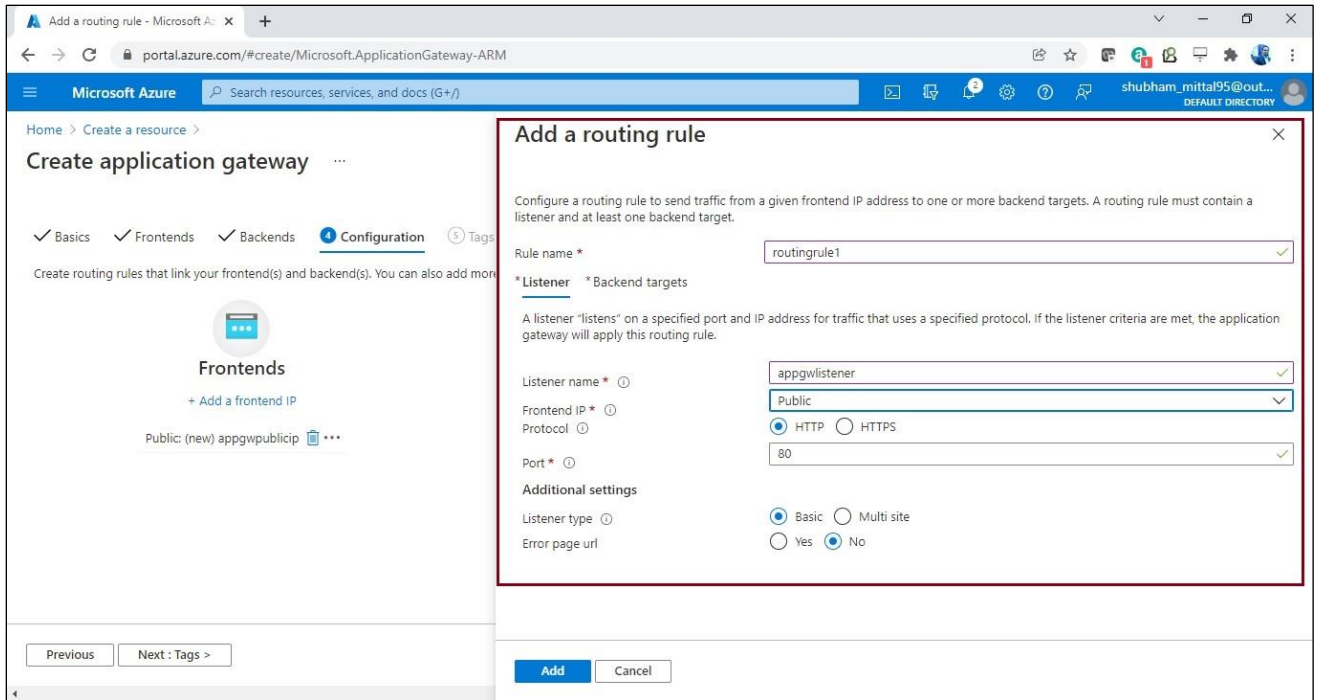


Figure 2.21

14. In the **Backend targets** tab, select the backend pool created in the **Backends** tab for the **Backend target**.
15. For the **HTTP settings**, select **Add new** to add a new HTTP setting.

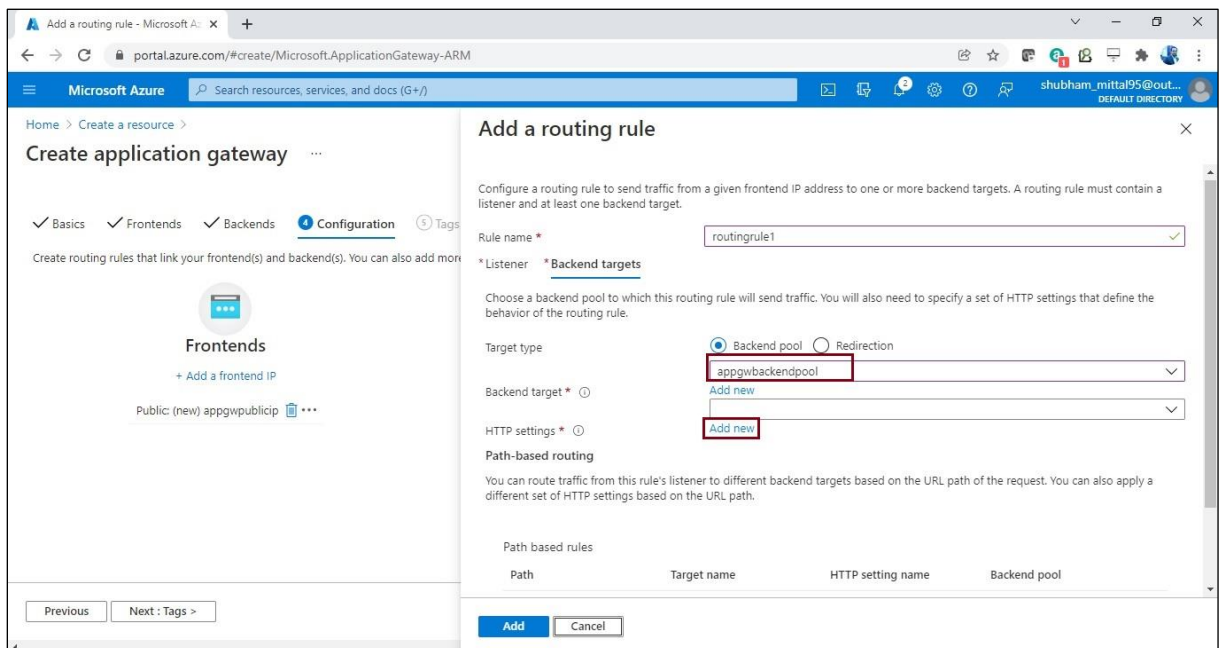


Figure 2.22

16. In the Add an HTTP setting, enter the user-defined HTTP setting name that is, **appgwhttpsetting**.
17. Keep the other settings as default and then click **Add** to return to the Add a routing rule.

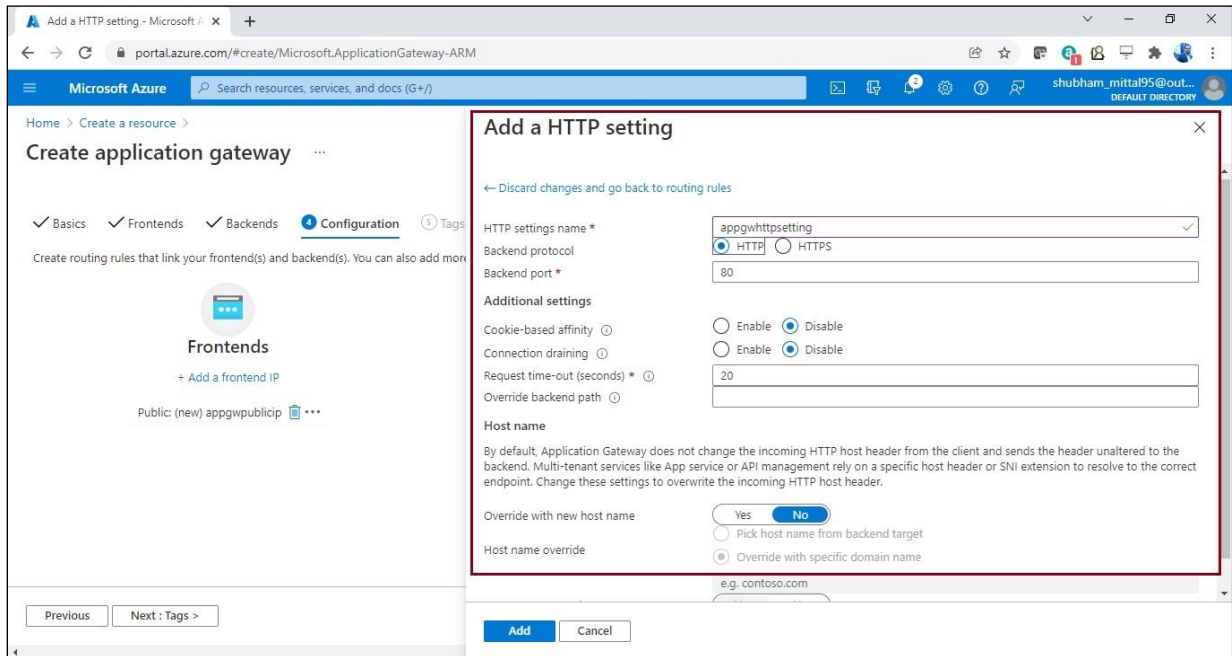


Figure 2.23

18. Select **Add** to save the routing rule in the Add a Routing and return to the **Configuration** tab.
19. Select **Next: Tags** and then click **Next: Review + create**.
20. Once validation is passed, select **Create**.

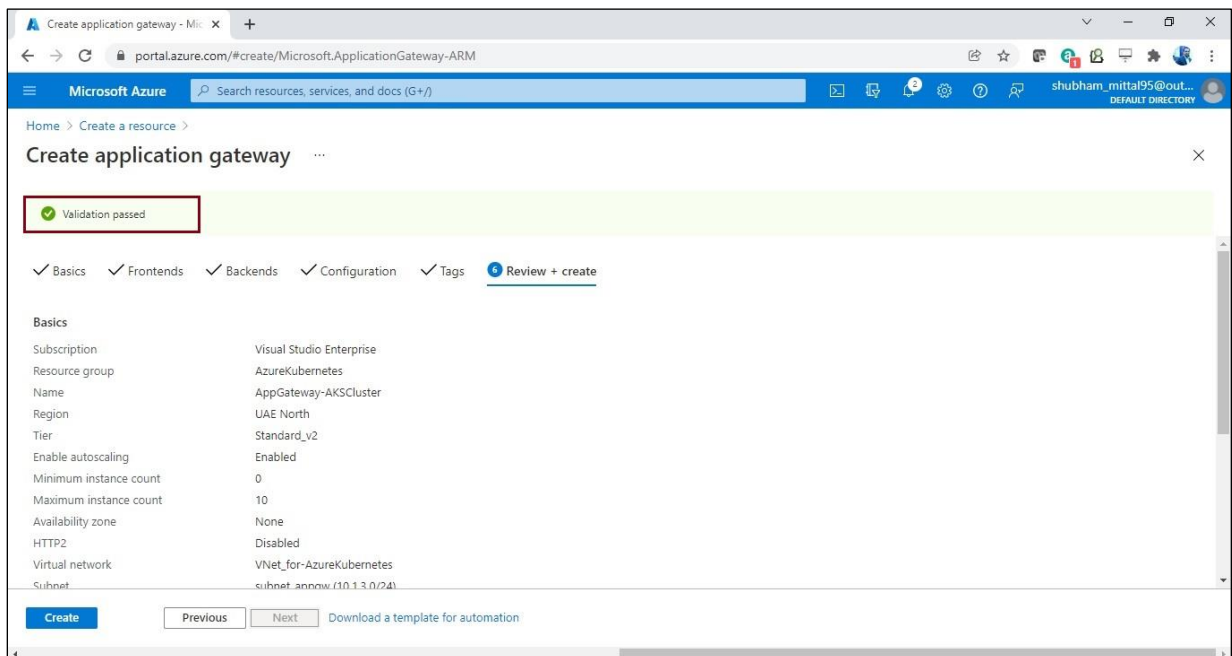


Figure 2.24

21. Once the deployment is complete, click **Go to resource**.

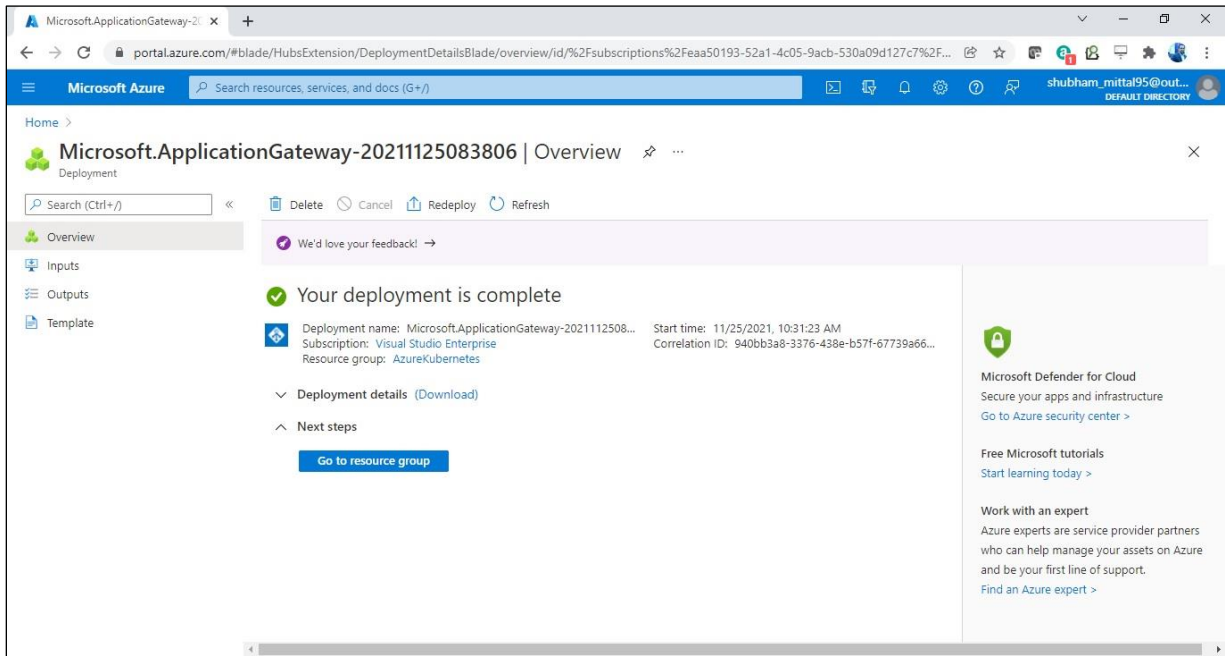


Figure 2.25

## 2.9.2 Installation of an application gateway ingress controller

An ingress controller is a piece of software that provides reverse proxy, configurable traffic routing, and TLS termination for Kubernetes services. Kubernetes ingress resources are used to configure the ingress rules and routes for individual Kubernetes services. Using an ingress controller and ingress rules, a single IP address can be used to route traffic to multiple services in a Kubernetes cluster.

### Pre-requisites:

- Azure Kubernetes Service must be created.
- Application Gateway must be created.

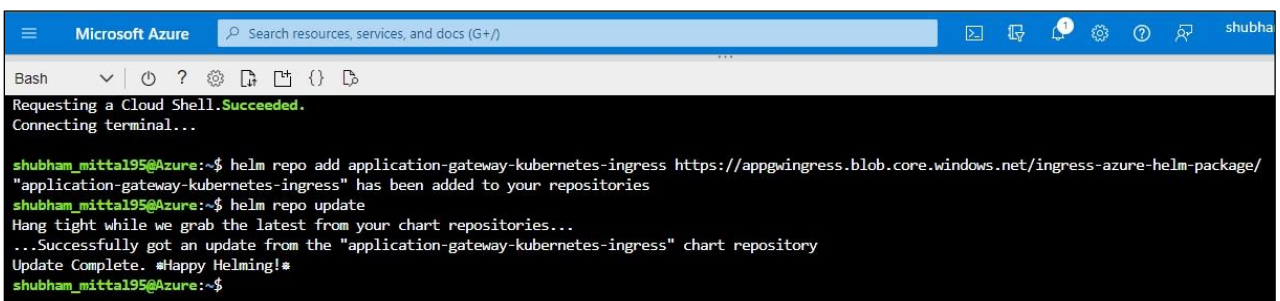
To install an Application Gateway Ingress Controller (AGIC), follow the below steps:

- Install Helm
- Azure Resource Manager Authentication using a Service Principle
- Install Ingress Controller using Helm

## 2.9.3 Install Helm

1. If you use the Azure Cloud Shell <https://portal.azure.com/#cloudshell/> then the Helm CLI is already installed. To install Helm on other platforms, refer to <https://helm.sh/docs/intro/install/>.
2. Open the Azure Cloud Shell and run the following command to add the **application-gateway-kubernetes-ingress** helm package.

```
helm repo add application-gateway-kubernetes-ingress
https://appgwingress.blob.core.windows.net/ingress-azure-helm-package/
helm repo update
```



```
Microsoft Azure Search resources, services, and docs (G+) shubha
Bash
Requesting a Cloud Shell.Succeeded.
Connecting terminal...
shubham_mittal195@Azure:~$ helm repo add application-gateway-kubernetes-ingress https://appgwingress.blob.core.windows.net/ingress-azure-helm-package/
"application-gateway-kubernetes-ingress" has been added to your repositories
shubham_mittal195@Azure:~$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "application-gateway-kubernetes-ingress" chart repository
Update Complete. #Happy Helming!#
shubham_mittal195@Azure:~$
```

Figure 2.26

## 2.9.4 ARM authentication using a service principle

Perform the below steps for ARM Authentication using a service principle:

1. Application Gateway Ingress Controller (AGIC) communicates with the Kubernetes API Server and Azure Resource Manager. It requires authentication to access these APIs
2. Open the Azure Cloud Shell <https://portal.azure.com/#cloudshell/> and run the following command to create a service principle and encode with base64. The base64 encoding is required for the JSON blob to be saved to Kubernetes.

```
az ad sp create-for-rbac --role Contributor --sdk-auth --scopes /subscriptions/<Subscription-id>/resourceGroups/<Resource group> | base64 -w0
```

Where,

**Subscription-id** – Enter your account subscription id

**Resource group** – Enter the name of resource group associated with kubernetes cluster

For Example –

```
az ad sp create-for-rbac --role Contributor --sdk-auth --scopes /subscriptions/323527f6b-535a1-406d-239b-0972646c8500c3/resourceGroups/AzureKubernetes | base64 -w0
```

---

**NOTE:**

Keep the **base64 encoded JSON blob** as these values are required in the following steps for installing AGIC.

---

## 2.9.5 Add or update Kubeconfig file

Perform the below steps to add or update kubeconfig file:

1. Open the Azure Cloud Shell <https://portal.azure.com/#cloudshell/> .
2. Delete the **.kube/config** file (if already exists) using below command:

```
rm .kube/config
```

3. Now execute the below command to re-create **.kube/config** file:

```
az aks get-credentials --resource-group <ResourceGroupName> --name <AzureEKSClusterName>
```

**For example,**

```
az aks get-credentials --resource-group AzureKubernetes --name BPMSuite-AKSCluster
```

## 2.9.6 Install ingress controller using Helm

Perform the below steps to install ingress controller using Helm:

1. Open the Azure Cloud Shell <https://portal.azure.com/#cloudshell/> and run the following command to download the `helm-config.yaml` file which configures the Application Gateway Ingress Controller.

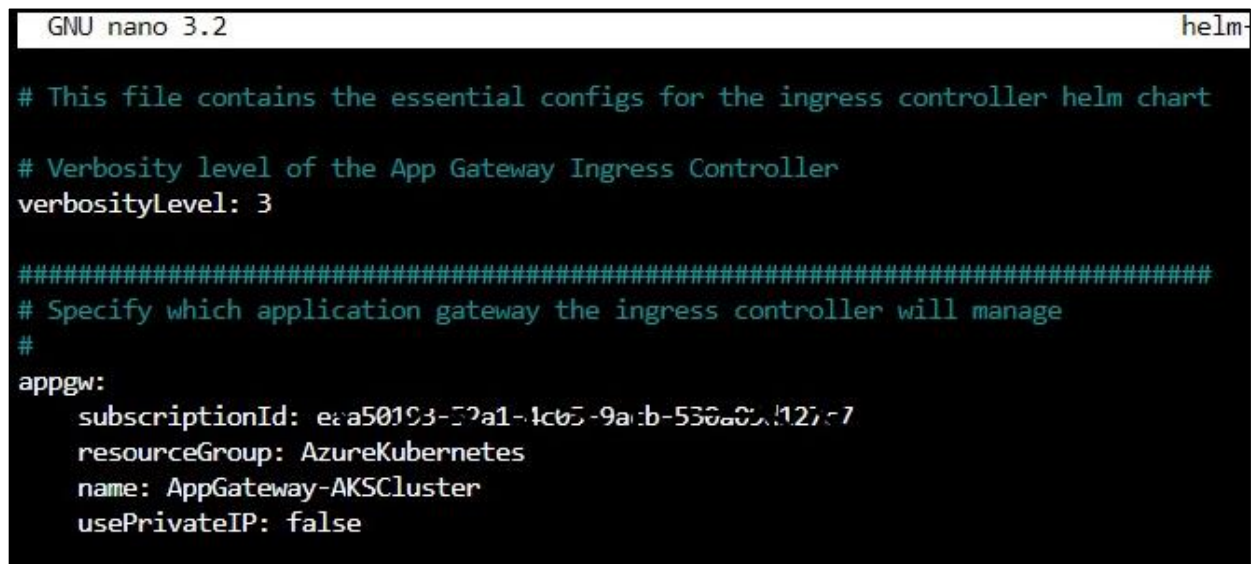
```
wget https://raw.githubusercontent.com/Azure/application-gateway-kubernetes-ingress/master/docs/examples/sample-helm-config.yaml -O helm-config.yaml
```

2. Edit the `helm-config.yaml` file and fill in the values for **appgw** (Application Gateway) and **armAuth** (ARM Authentication using Service Principle).

```
nano helm-config.yaml
```

3. Update the `<subscriptionId>`, `<resourceGroupName>`, and `<applicationGatewayName>` for **appgw**.

For example,



```
GNU nano 3.2                               helm-
# This file contains the essential configs for the ingress controller helm chart
# Verbosity level of the App Gateway Ingress Controller
verbosityLevel: 3
#####
# Specify which application gateway the ingress controller will manage
#
appgw:
  subscriptionId: e8a50193-52a1-4cb5-9a1b-536a05a1127c7
  resourceGroup: AzureKubernetes
  name: AppGateway-AKSCluster
  usePrivateIP: false
```

Figure 2.27

4. Comment the **armAuth** using **AAD-Pod-Identity** and uncomment the **armAuth** using **Service Principle**.

5. Update the **base64 encoded JSON blob** created in the previous step 'ARM Authentication using a Service Principle' for **secretJSON**.

For example,



```

Bash  GNU nano 3.2  helm-config.yaml

# watchNamespace: <namespace>

#####
# Specify the authentication with Azure Resource Manager
#
# Two authentication methods are available:
# - Option 1: AAD-Pod-Identity (https://github.com/Azure/aad-pod-identity)
#armAuth:
#  type: aadPodIdentity
#  identityResourceID: <identityResourceId>
#  identityClientID: <identityClientId>

## Alternatively you can use Service Principal credentials
armAuth:
  type: servicePrincipal
  secretJSON: ewogICJjbGllbnRJZCI6ICJhZGJhYjQ5ZS1iNGI3LTQ0YjAtYjc2ZC0zMjM4N2ZhYWQzODQ1LAogICJjbGllbnRT

#####
# Specify if the cluster is RBAC enabled or not
rbac:
  enabled: false # true/false

```

Figure 2.28

- Specify the rbac enabled as **true** if the cluster is RBAC enabled.

For example,

```

#####
# Specify if the cluster is RBAC enabled or not
rbac:
  enabled: true # true/false

```

Figure 2.29

- Install Helm chart **application-gateway-kubernetes-ingress** with the *helm-config.yaml* configuration from the previous step.

```

helm install ingress-azure \
-f helm-config.yaml \
application-gateway-kubernetes-ingress/ingress-azure \
--version 1.4.0

```

For example,



```
shubham_mittal195@Azure:~$ helm install ingress-azure \
> -f helm-config.yaml \
> application-gateway-kubernetes-ingress/ingress-azure \
> --version 1.4.0
W1125 06:56:40.353928 392 warnings.go:67] apiextensions.k8s.io/v1beta1 CustomResourceDefinition is deprecated in v1.16+, unavailable in v1.22+; use apiextensions.k8s.io/v1 CustomResourceDefinition
NAME: ingress-azure
LAST DEPLOYED: Thu Nov 25 06:56:43 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Thank you for installing ingress-azure:1.4.0.

Your release is named ingress-azure.
The controller is deployed in deployment ingress-azure.

Configuration Details:
-----
* AzureRM Authentication Method:
  - Use Service Principal
* Application Gateway:
  - Subscription ID : eaa50193-52a1-4c05-9acb-530a09d127c7
  - Resource Group : AzureKubernetes
  - Application Gateway Name : AppGateway-AKSCluster
* Kubernetes Ingress Controller:
  - Watching All Namespaces
  - Verbosity level: 3
shubham_mittal195@Azure:~$
```

Figure 2.30

---

**NOTE:**

Use the latest version of ingress-azure. You can get the release information from the below link:

<https://github.com/Azure/application-gateway-kubernetes-ingress/releases>

---

- Application Gateway Ingress Controller runs as a pod in the Kubernetes cluster. You can check the running status of the AGIC pod using the below command:

```
Kubect1 get po | grep ingress
```

For example,

```
shubham_mittal195@Azure:~$ kubect1 get po | grep ingress
ingress-azure-649b85494b-2tthf 1/1 Running 0 10m
```

Figure 2.31

## 2.10 Configuring the DNS zone

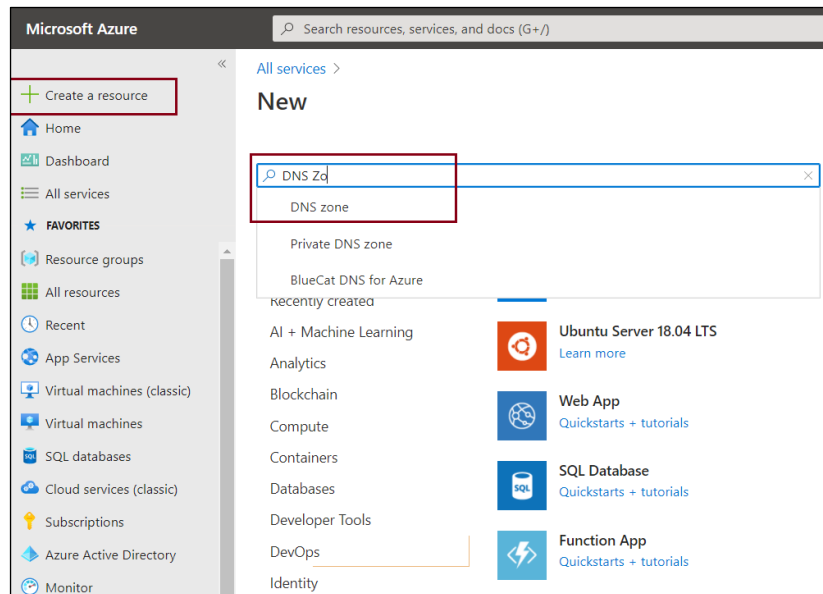
Ingress Controller creates a Load Balancer and routes the incoming requests to the target Kubernetes services according to the host-based routing rules. Host-based routing is a capability of Ingress Controller that redirects the user requests to the right service based on the request-host header. For example, set the rules as below:

- IF URL is 'ibps5serviceinstance.azure.co.in' THEN redirect to iBPS ServiceInstance Web container.
- IF URL is 'ibps5userinstance.azure.co.in' THEN redirect to the iBPS UserInstance Web container.

To support the host-based routing, register a custom domain and create a new RecordSet in DNS Zone for each host-path.

**To create a DNS Zone, follow the below steps:**

1. Sign in to the Azure Portal using <https://portal.azure.com>.
2. After a successful login, click **Create a resource** and search for the **DNS Zone**.



**Figure 2.32**

3. Click **Create**.
4. On the **Create DNS zone** page, specify the following details under the **Basics** tab:
  - **Subscription** – Select a valid Azure subscription.
  - **Resource group** – Select or create an Azure Resource group, such as **AzureKubernetes**.
  - **Name** – Specify a valid DNS Zone name such as **azure.co.in**.
  - Click the **Next: Tags**.
5. On the **Tags** page, keep the default options and click the **Next: Review + create**.
6. On the **Review + create** page, click the **Create** once validation passed.
7. Once deployment is complete, click the **Go to resource**. The Created DNS Zone’s **Overview** page appears.
8. On the top of the **DNS Zone** page, select **+ Record set**.
9. On the **Add record set** page, type or select the following values:
  - **Name** – Enter the user-defined name.
  - **Type** – Select type as “A – IPv4-address”
  - **Alias record set** – Select alias as **Yes**.
  - **Alias type** – Select the alias type as **Azure resource**.
  - **Choose a subscription** – Select a valid Azure subscription.
  - **Azure resource** – Select the Public IP Address created for the Application Gateway, that is, **appgwpublicip**.
  - **TTL (Time To Live)** – Time-to-live of the DNS request specifies how long DNS servers and clients can cache a response. Keep the default values.
  - Click **OK** to save the record set.

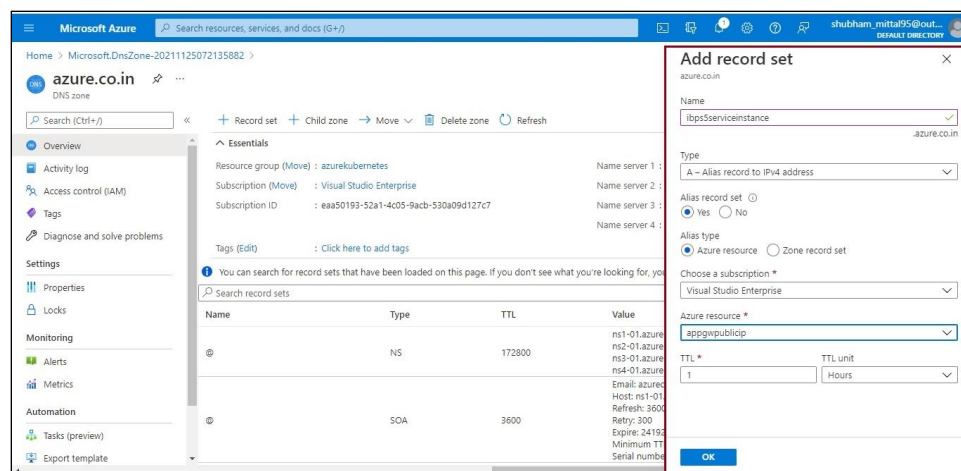


Figure 2.33

## 2.11 Monitoring the Kubernetes dashboard

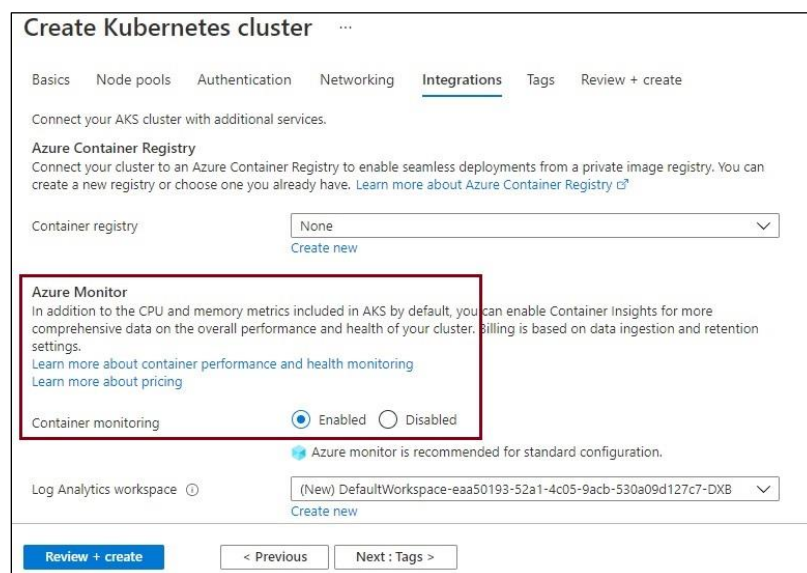
This section explains how to monitor the Kubernetes dashboard.

- The Azure portal includes a Kubernetes resource view for easy access to the Kubernetes resources in your Azure Kubernetes Service (AKS) cluster.
- To see the Kubernetes resources, navigate to your AKS cluster in the Azure portal. The navigation pane on the left is used to access your resources. The resources include:
  - **Namespaces** – Displays the namespaces of your cluster. The filter at the top of the namespace list provides a quick way to filter and display your namespace resources.
  - **Workloads** – Shows the information about deployments, pods, replica sets, stateful sets, daemon sets, jobs, and cron jobs deployed to your cluster. The screenshot below shows the default system pods in an example AKS cluster.
  - **Services and ingresses** – Shows all of your cluster's service and ingress resources.
  - **Storage** – Shows your Azure storage classes and persistent volume information.
  - **Configuration** – Shows your cluster's config maps and secrets.

## 2.12 Azure monitor for container insights

Azure Monitor for containers is a feature designed to monitor the health and performance of container workloads deployed to Azure Kubernetes service. It delivers a comprehensive monitoring experience and gives us performance visibility by collecting memory and processor metrics from controllers, nodes, and containers that are available in Kubernetes through the Metrics API.

By default, Azure Monitor is enabled for container monitoring during Azure Kubernetes service creation (Under Integrations tab).



The screenshot shows the 'Create Kubernetes cluster' page in the Azure portal, specifically the 'Integrations' tab. The page is titled 'Create Kubernetes cluster' and has several tabs: Basics, Node pools, Authentication, Networking, Integrations (selected), Tags, and Review + create. Below the tabs, there is a section for 'Connect your AKS cluster with additional services'. Under this section, there are two main options: 'Azure Container Registry' and 'Azure Monitor'. The 'Azure Monitor' section is highlighted with a red box. It contains the following text: 'In addition to the CPU and memory metrics included in AKS by default, you can enable Container Insights for more comprehensive data on the overall performance and health of your cluster. Billing is based on data ingestion and retention settings. Learn more about container performance and health monitoring. Learn more about pricing.' Below this text, there are two radio buttons: 'Enabled' (which is selected) and 'Disabled'. There is also a note that says 'Azure monitor is recommended for standard configuration.' At the bottom of the page, there is a 'Log Analytics workspace' section with a dropdown menu showing '(New) DefaultWorkspace-aaa50193-52a1-4c05-9acb-530a09d127c7-DXB' and a 'Create new' link. At the very bottom, there are three buttons: 'Review + create', '< Previous', and 'Next : Tags >'.

Figure 2.34

To view the container insights, perform the below steps:

1. Sign in to the Azure portal at <https://portal.azure.com>.
2. On the Azure portal menu or from the Home page, select **All resources > Click on the created Kubernetes service**.
3. Click the **Monitoring >> Insights** appears in the left pane. Here's a series of tabs to monitor your AKS Cluster, Nodes, Containers, Controllers, and more.

## 2.13 Configuring the CosmosDB (MongoDB API)

Azure CosmosDB is a fully managed database service for apps written for MongoDB.

To configure the Dev Stage, follow the below steps:

1. Select the **Azure Cosmos DB API for MongoDB**.

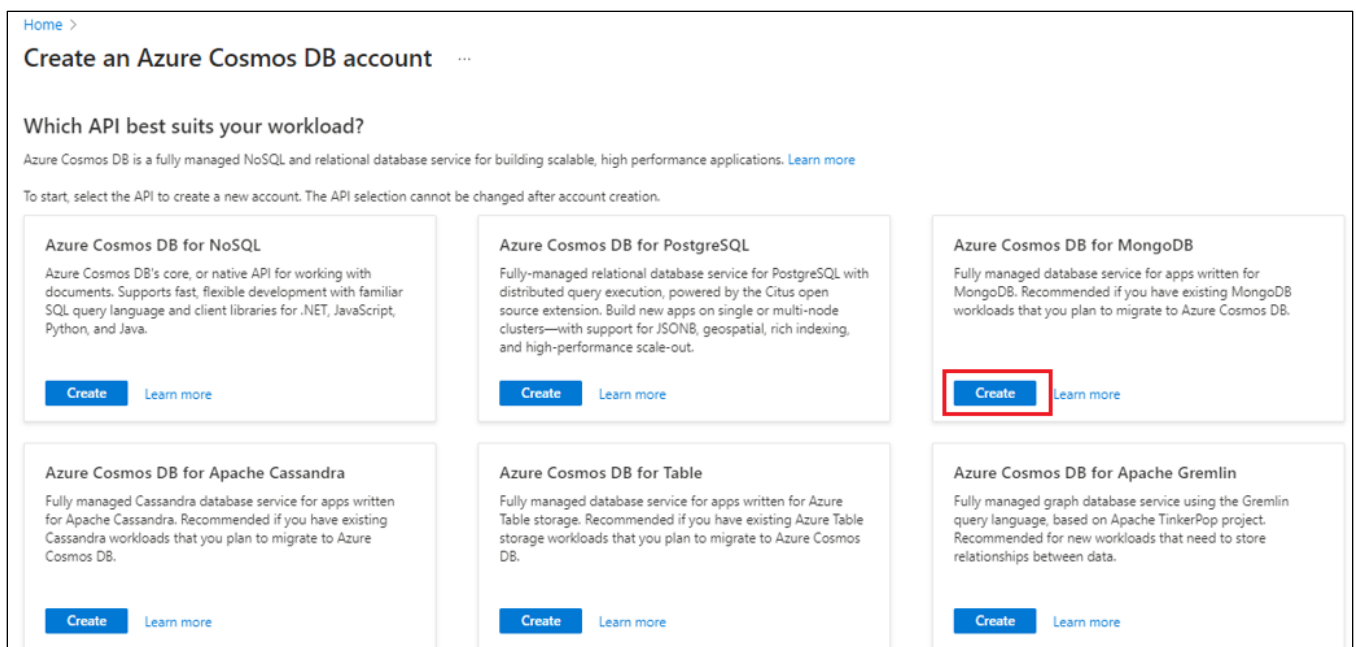


Figure 2.35

2. On the 'Create Azure Cosmos DB Account - Azure Cosmos DB API for MongoDB' page, specify the following details under the **Basics** tab:
  - a. **Subscription:** Select a valid Azure subscription.
  - b. **Resource group:** select or create an Azure Resource group, such as **NCC-PT**.
  - c. **Account Name:** Enter a valid name such as **ncc-pt-mongodb-account**.
  - d. **Location:** Select a region (location) in which you want to create the cosmosDB (MongoDB).
3. Choose the remaining values as default.
4. Click the **Next: Global Distribution**.

5. On the **Global Distribution** page, keep the default options. At the bottom of the screen, click **Next: Networking**.
6. On **Networking** page, choose 'Connectivity method' as **Private endpoint** and choose **Deny** in 'Allow Public Network Access'.

Create Azure Cosmos DB Account - Azure Cosmos DB for MongoDB ...

Basics Global Distribution **Networking** Backup Policy Encryption Tags Review + create

Network connectivity

You can connect to your Azure Cosmos DB account either publically, via public IP addresses or service endpoints, or privately, using a private endpoint.

⚠ The current settings will block all access to the account, review the network configurations below in case you want to allow selected traffic to have access.

Connectivity method \*

All networks

Public endpoint (selected networks)

Private endpoint

Configure Firewall

Allow access from Azure Portal ⓘ  Allow  Deny

Allow access from my IP (14.142.3.154) ⓘ  Allow  Deny

Allow Public Network Access ⓘ  Allow  Deny

Private endpoint

[Review + create](#) [Previous](#) [Next: Backup Policy](#)

Figure 2.36

7. Create a Private Endpoint using the '+' button. Enter relevant details and use the same Virtual Network which was created during Kubernetes Cluster creation. Click **OK**.

**Create private endpoint**

Subscription \* ⓘ DevopsGen

Resource group \* ⓘ NCC-QA  
[Create new](#)

Location \* Central India

Name \* ⓘ nccqa ✓

CosmosDB sub-resource \* ⓘ Azure Cosmos DB for MongoDB

**Networking**  
 To deploy the private endpoint, select a virtual network subnet. [Learn more about private endpoint networking](#)

Virtual network ⓘ aks-vnet-33088092 (MC\_NCC-QA\_k8s-ncc-qa-cluster\_centralindia)

Subnet \* ⓘ aks-subnet

ⓘ If you have a network security group (NSG) enabled for the subnet above, it will be disabled for private endpoints on this subnet only. Other resources on the subnet will still have NSG enforcement.

Private DNS integration

OK Discard

**Figure 2.37**

8. At the bottom of the screen, click the **Next: Backup Policy**.
9. On the **Backup Policy** page, keep the default options (or change them as per requirements). At the bottom of the screen, click the **Next: Encryption**.
10. On the **Encryption** page, keep the default options (or change them as per requirements). At the bottom of the screen, click **Next: Tags**.
11. On the **Tags** page, keep the default options (or change them as per requirements). At the bottom of the screen, click **Next: Review+Create**.
12. On the **Review + create** page, click the **Create** once validation passed.
13. Once db is created, click the **Go to resource** button.
14. Click **Connection String** (under **Settings** on left pane). Keep the **Primary Connection String**. This value requires in yml files for environment variable named 'spring\_data\_mongodb\_uri'.

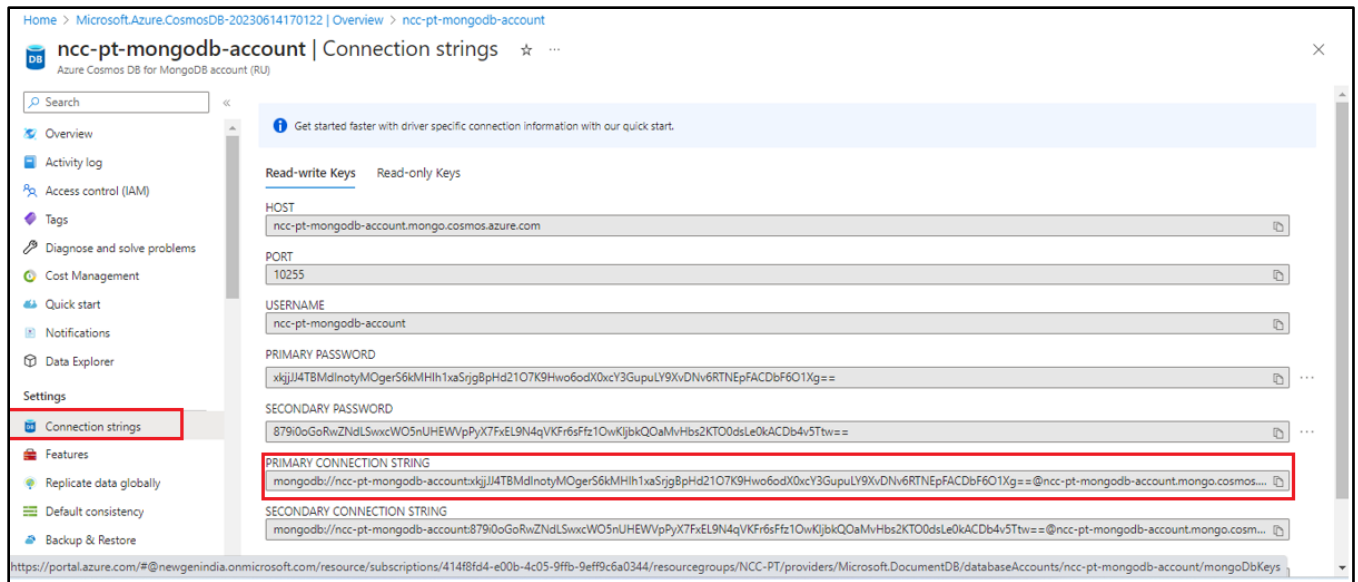


Figure 2.38

15. Click **Data Explorer**.
16. Click **New Database**.
17. Create a new database **ecmdata** with **Database throughput (autoscale)** as **Automatic** and **Database Max RUs 10000**.
18. Create a new database **ecmdataext** with **Database throughput (autoscale)** as **'Automatic'** and **Database Max RUs 5000**.
19. Click **New Collection**.
20. Create collections in Azure Portal, so that the default indexes (that is, "\$\*\*\_1" indexes) are created automatically.
21. Make sure that Auto Indexing is OFF.
22. Refer to the below table to make collections and use Unsharded where there is no Shard Key available (maximum 25 collections in one database).

Following is the segregation of collections into 2 databases:

DataBase name	Collection name	Shard key	Auto indexing
ecmdata	annotation	tenantId	off
ecmdata	auditLogs	tenantId	off
ecmdata	content	tenantId	off
ecmdata	contentLocation	tenantId	off
ecmdata	counter	tenantId	off
ecmdata	dataclass	tenantId	off
ecmdata	folder	tenantId	off
ecmdata	lock	tenantId	off
ecmdata	mail	tenantId	off



DataBase name	Collection name	Shard key	Auto indexing
ecmdata	microApi	tenantId	off
ecmdata	notes	tenantId	off
ecmdata	passwordchangetoken	_id	off
ecmdata	privileges	tenantId	off
ecmdata	prospect	tenantId	off
ecmdata	reportConfigurator	tenantId	off
ecmdata	shareContent	tenantId	off
ecmdata	storageCredentials	tenantId	off
ecmdata	storageLocation	tenantId	off
ecmdata	storageProcess	tenantId	off
ecmdata	subscription		off
ecmdata	syncFolderOperation	tenantId	off
ecmdata	tenant		off
ecmdata	users		off
ecmdataext	app	tenantId	off
ecmdataext	globalTag		off
ecmdataext	roles		off
ecmdataext	secretkeys	tenantId	off
ecmdataext	securityClass		off
ecmdataext	system.indexes	tenantId	off
ecmdataext	userProfile	tenantId	off

Go to the newly created mongodb account and then click on **'Connection String'** (under **'Settings'** on left hand side. Make note of **'Primary Connection String'**.

**Use MongoDB Compass to import ecm.subscription.json file.**

- Enter the connection string of **MongoDB** in URI block as shown in Fig and click **Connect**.

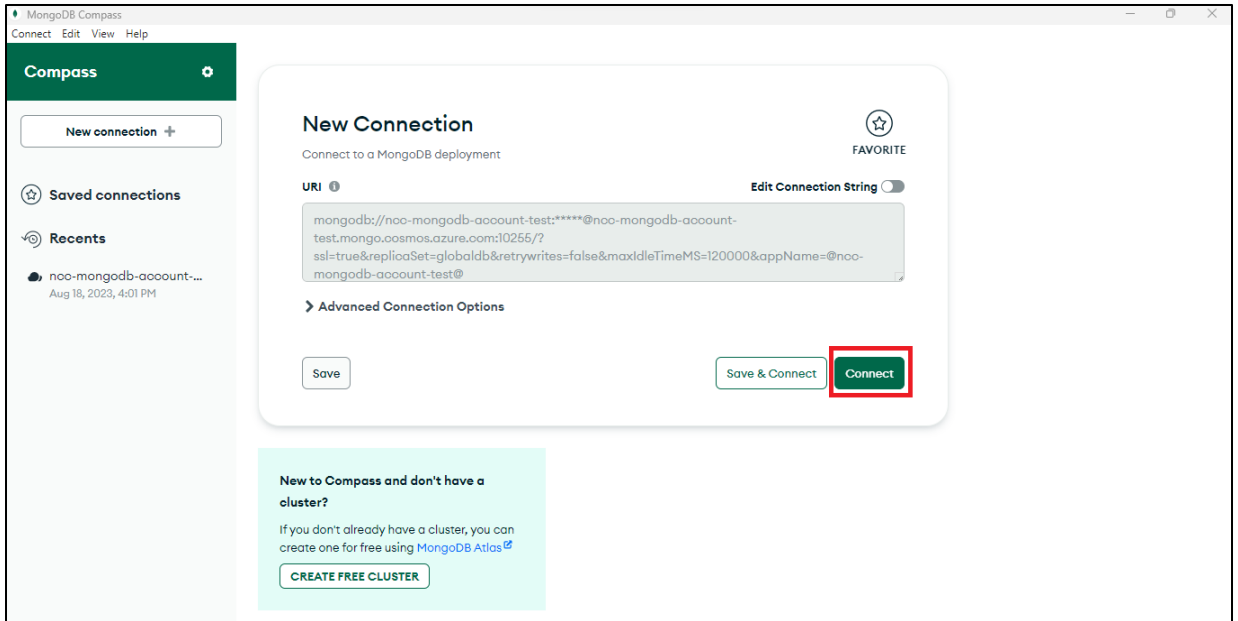


Figure 2.39

- Go to subscription collection in **ecmdata** and click on **Import Data**.

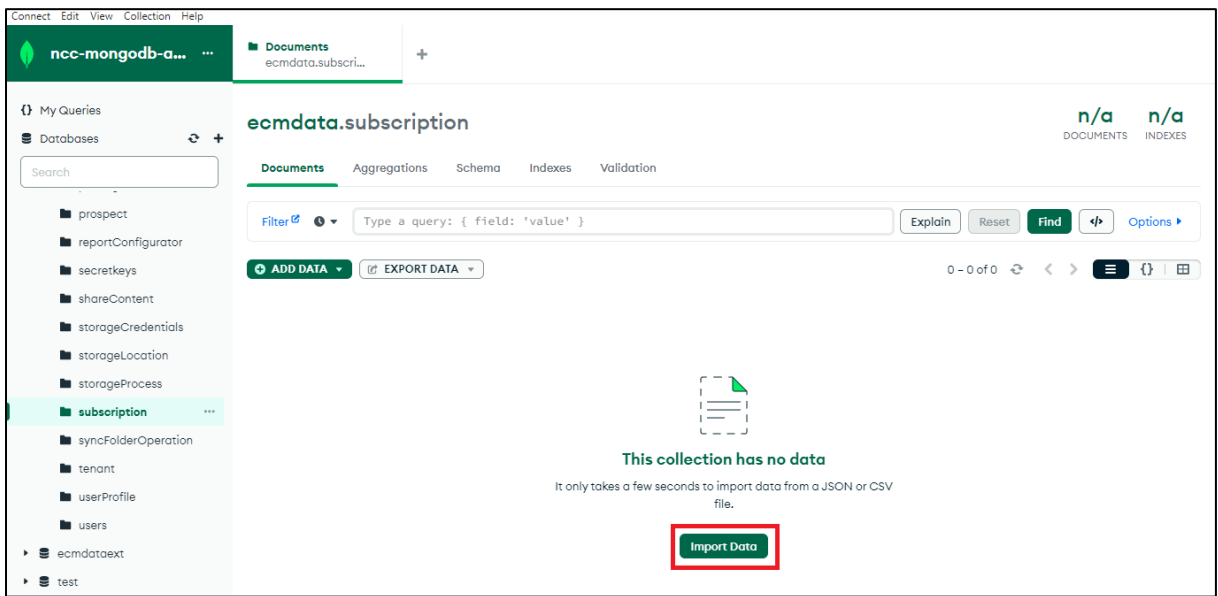


Figure 2.40

- A dialog box will open, choose the **ecmdata.subscription.json** file that is provided to you.

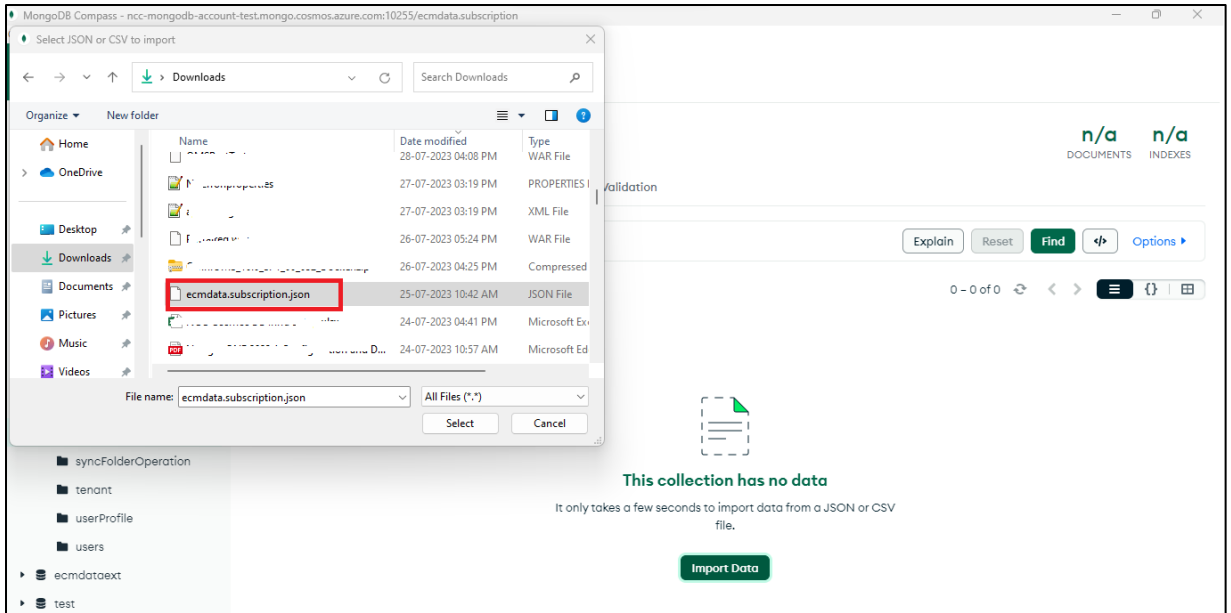


Figure 2.41

- After importing you will get the **Import completed** message.

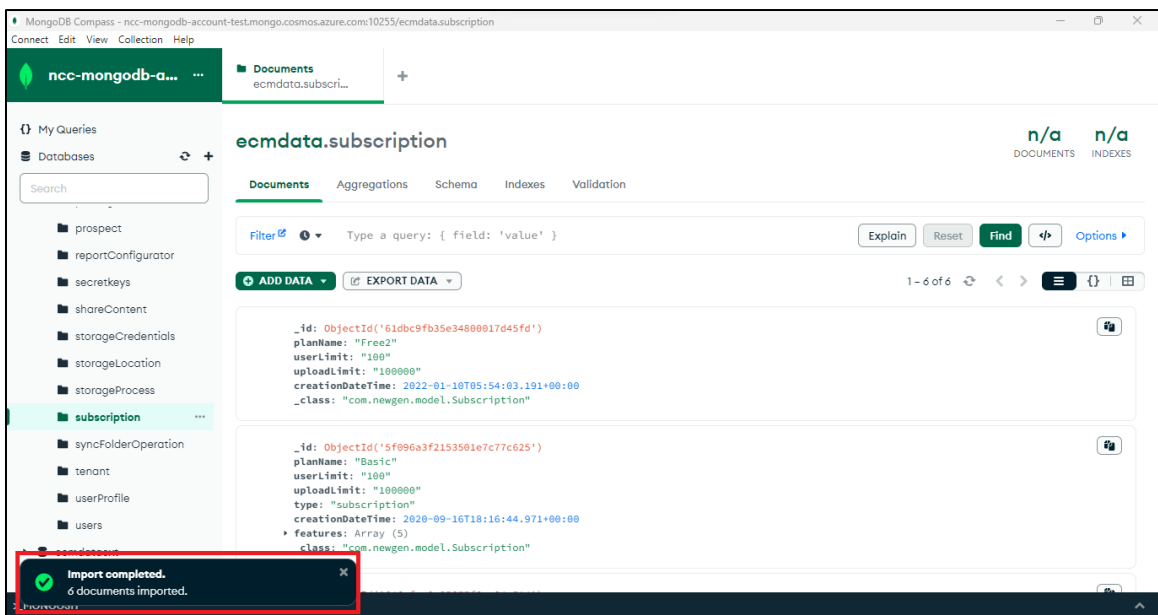


Figure 2.42

## 2.14 Configuring Azure cognitive search

Azure Cognitive Search helps build a full-text search experience and integrate it with custom applications and other Azure services.

To configure the Azure cognitive search, perform the below steps:

1. Go to Azure portal **Home** page and search for **search**. Click **Cognitive search**.

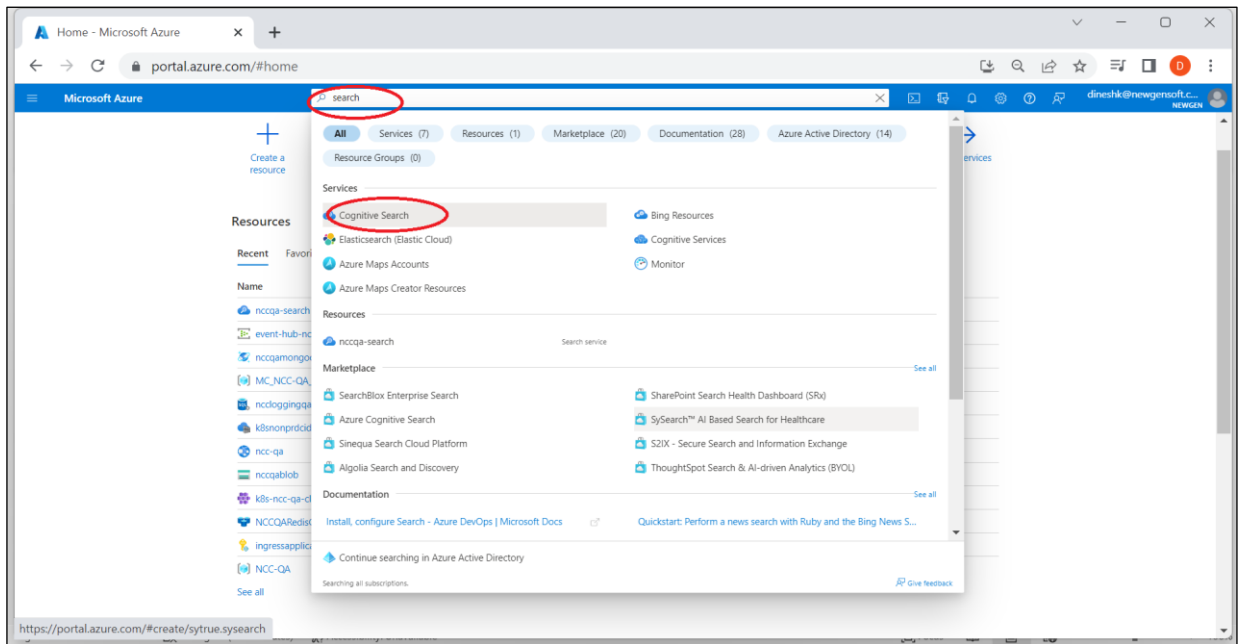


Figure 2.43

2. Click **+Create**.
3. On **Create a search service:Basics** page, enter relevant details. Select **Pricing tier** as **Standard (S-Standard. Supported up to 50 tenants only)**. Click button **Next:Scale**.
4. On **Create a search service:Scale** page, just review the details. Click the button **Next:Networking**.
5. On **Create a search service: Networking** page, choose Endpoint connectivity (data) as **Private**. Create a Private Endpoint using the '+' button.

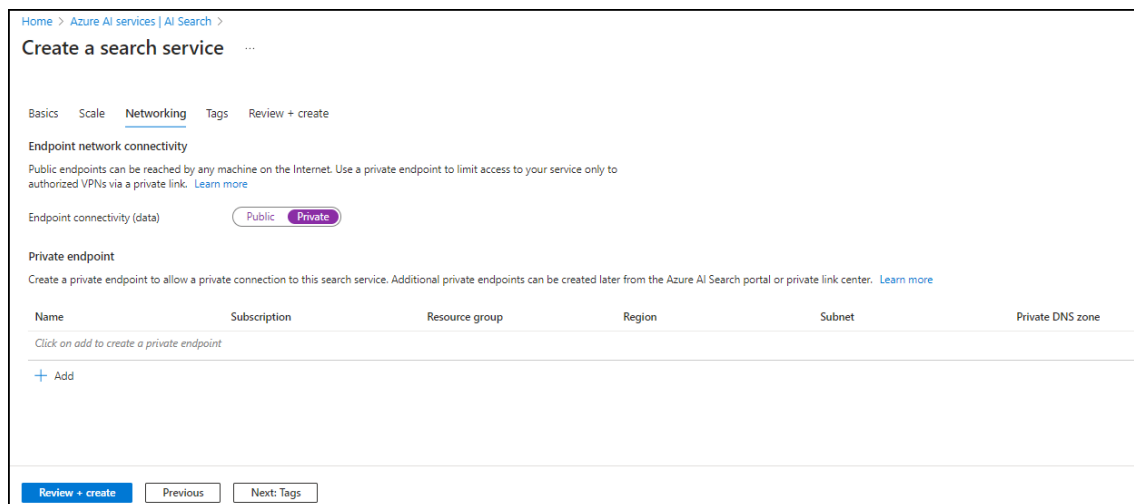


Figure 2.44

- Enter relevant details and use the same Virtual Network which was created during Kubernetes Cluster creation. Click **OK**.

Figure 2.45

- Click the button **Next:Tags**.
- On **Create a search service: Tag** page, choose the default value (blank) or enter the relevant tags.
- Click the button **Next:Review+Create**.
- On **Create a search service: Review+Create** page, review the details. Once the validation is done, click the button **Create**.
- Once created, click the **'Go to resource'** button.
- Create an index with any name. For example, **ncc-pt-search-index1**:
- Add the following fields in **ncc-pt-search-index1**:

Field name	Type	Retrievable	Filterable	Sortable	Facetable	Searchable
doc_id	String	Yes	No	No	No	No
name	String	Yes	No	No	No	Yes
folderName	String	Yes	No	No	No	Yes
contentType	String	Yes	No	No	No	No
folderType	String	Yes	No	No	No	No
comments	String	Yes	No	No	No	Yes
parentFolderId	String	Yes	No	No	No	No
ownerName	String	Yes	Yes	No	Yes	Yes
ownerId	String	Yes	No	No	No	No

Field name	Type	Retrievable	Filtrable	Sortable	Facetable	Searchable
tenantId	String	Yes	Yes	No	No	No
contentLocationId	String	Yes	No	No	No	No
creationDateTime	DateTimeOffset	Yes	No	No	No	NA
revisedDateTime	DateTimeOffset	Yes	No	No	No	NA
accessDateTime	DateTimeOffset	Yes	No	No	No	NA
version	String	Yes	No	No	No	No
flag	String	Yes	No	No	No	No
noOfPages	String	Yes	No	No	No	No
documentType	String	Yes	Yes	No	Yes	Yes
documentSize	String	Yes	No	No	No	No
rid	String	Yes	No	No	No	No
dataclassIndexingText	StringCollection	Yes	Yes	NA	Yes	Yes
metadata	ComplexType	NA	NA	NA	NA	NA
content	String	Yes	No	No	No	No
merged_content	String	Yes	No	No	No	Yes
text	StringCollection	Yes	No	NA	No	No
layoutText	StringCollection	Yes	No	NA	No	No
latest	Boolean	Yes	Yes	No	No	NA
keywords	StringCollection	Yes	Yes	NA	Yes	Yes
primaryContentId	String	Yes	No	No	No	No

---

**NOTE:**

*doc\_id* must be primary key.

---

## 2.15 Configuring the Kafka(Azure Event Hubs)

To configure the Kafka (Azure Event Hubs), perform the below steps:

1. Go to Azure portal **Home** page and search for **Kafka**. Click **Event Hubs**.
2. Click **+ Create**.

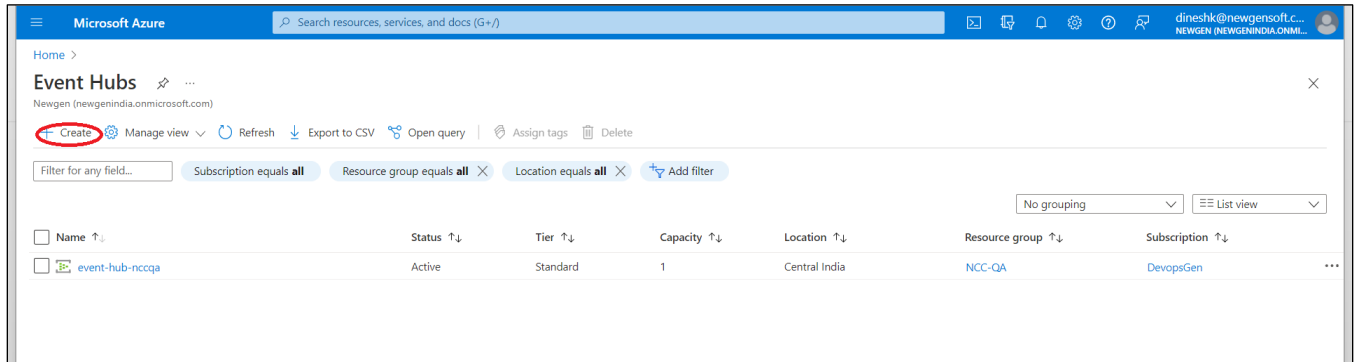


Figure 2.46

3. On **Create Namespace:Basics** page, enter the relevant details. Click **Next:Advanced**.

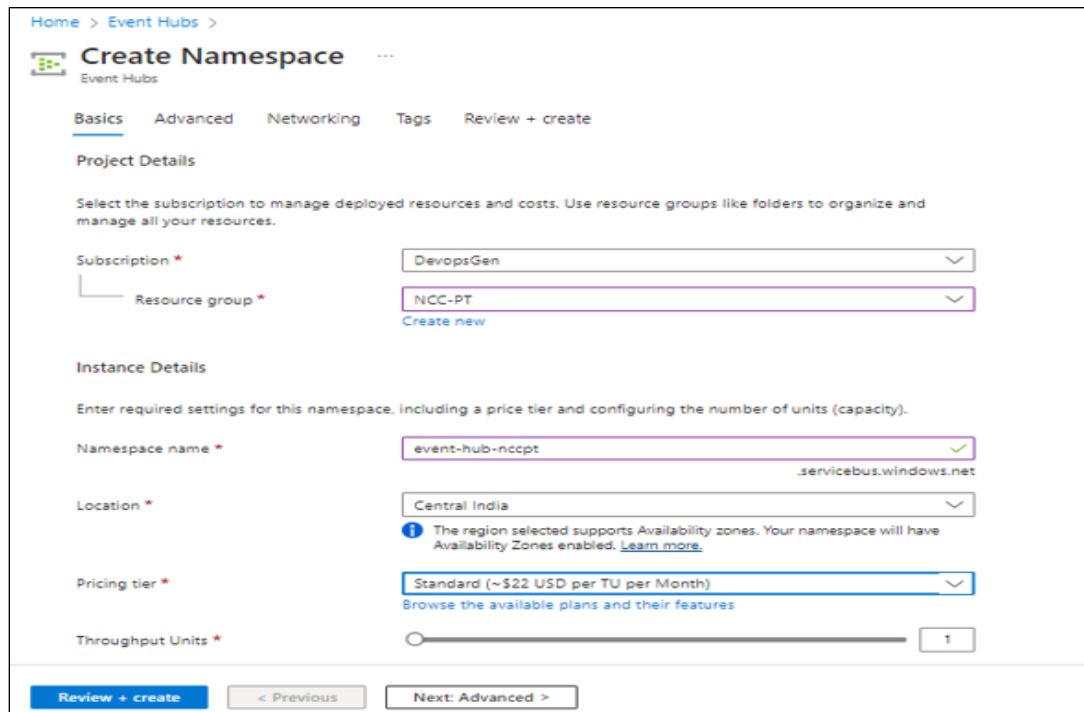


Figure 2.47

4. On **Create Namespace:Advanced** page, select the default values. Click **Next:Networking**.
5. On **'Create Namespace:Networking** page, choose Connectivity method as **Private access**. Create a Private Endpoint using the '+' button

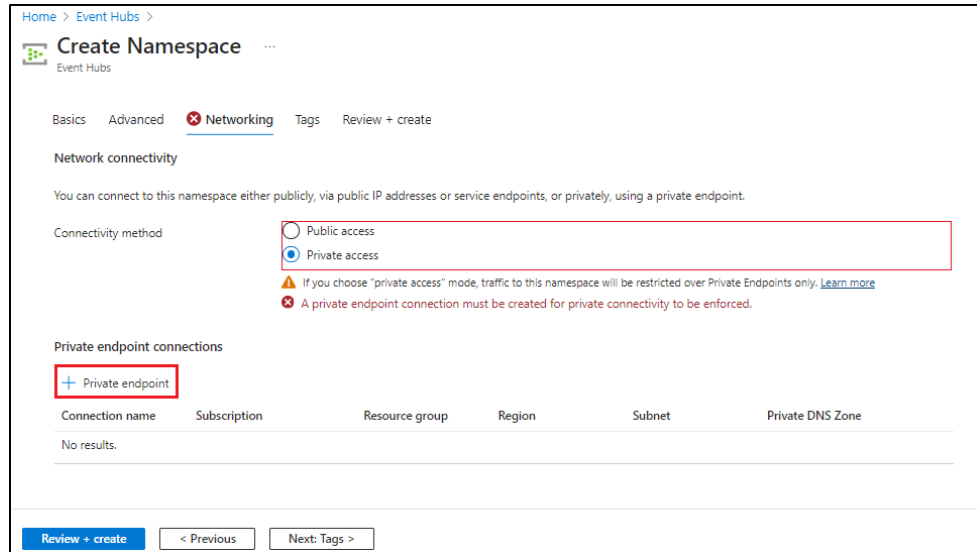


Figure 2.48

6. Enter relevant details and use the same Virtual Network which was created during Kubernetes Cluster creation. Click **OK** and Click **Next:Tags**.

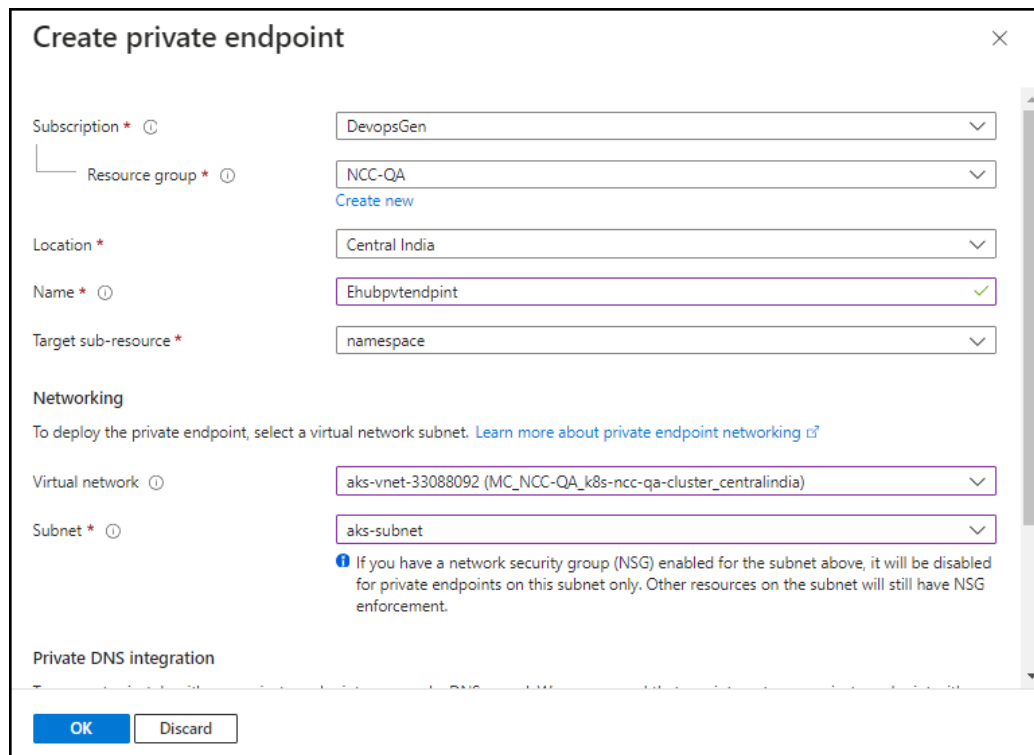


Figure 2.49

7. On **Create Namespace:Tags** page, choose default values or fill relevant details. Click **Next:Review + create**.
8. On **Create Namespace: Review + create** page, click **Create** after validation is done.



9. Once created, click the **Go to resource** button.
10. Click the **Shared access policies** under **Settings** on left. Click **RootManageSharedAccessKey**.
11. Click the **Shared access policies** under **Settings** on left. Click **RootManageSharedAccessKey**.
12. Keep the **Connection string**—primary key (in this case, **Endpoint=sb://event-hub-nccqa.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=vclawO1iSirEUylm8pVm4DP5NBCIROmlptmfMCmbU7I=**). This value is required in yml files for value of environment variable **spring\_kafka\_properties\_sasl\_jaas\_config** (password)

## 2.16 Configuring the MSSQL database server

To configure the MSSQL database server, perform the below steps:

1. Go to Azure portal **Home** page and search for **sql**. Click **SQL servers**.
2. Click **+ Create**.
3. On **Create SQL Server:Basic** page, enter relevant details. For **authentication method**, choose **Use SQL authentication**. Click the button **Next:Networking**.
4. On **Create SQL Server: Networking** page, enter relevant details. Click the button **Next:Additional settings**.
5. On **Create SQL Server: Additional settings** page, enter relevant details. Click the button **Next:Tags**.
6. On **Create SQL Server: Tags** page, enter relevant tags or leave them blank (based on requirements). Click the button **Next:Review+create**.
7. On **Create SQL Server: Review+create** page, review the details. After validation is done, click the button **Create**.
8. After **SQL Server** is created, click the button **Go to resource**. Then Click **Networking** inside **Security** present on the left pane.
9. **Disable** Private network access. Then go to **Private access** tab.
10. **Create a Private Endpoint** using '+' icon.
11. On **Create a private endpoint: Basics** page, enter relevant details. Click the button **Next:Resource**.
12. On **Create a private endpoint: Resources** page, choose the default values. Click the button **Next:Virtual Network**.
13. On **Create a private endpoint: Virtual Network** page, choose the same 'Virtual Network' and 'subnet' that was made during Kubernetes Cluster creation, and choose 'Dynamically allocate IP' address in **Private IP Configuration**. Click the button **Next:DNS**.
14. On **Create a private endpoint: DNS** page, choose the default values. Click on the button **Next:Tags**.
15. On **Create a private endpoint: Tags** page, choose the default values. Click on the button **Create**.
16. Click **+ Create database**.
17. Enter the relevant details. Click **Configure database**.

18. On Configure database, select **Service Tier** as **Standard, 20 DTUs** and **20GB Data Max size**. (This specification is for QA environment and may vary for any other environment.). Click the **'Apply'** button.
19. Click the **Next:Networking** button.
20. On **Networking** page, choose the default values. Click the **Next:Security** button.
21. On **Security** page, choose the default values. Click the **Next:Additional Settings** button.
22. On **Additional Settings** page, choose the default values. Click the **Next:Tags** button.
23. On **Tags** page, choose the default values. Click the **Next:Review+Create** button.
24. On **Review+Create** page, review the details and click the **Create** button.

### Connect to MSSQL Server to run the MSSQL DB scripts.

1. Connect to MSSQL Server using same credentials passed while creating MSSQL database in Azure portal.
2. Click on **Databases**, then **Right-click** on created database and click on **New Query**.
3. Run all the MSSQL queries after clicking on **Execute** you will get **Query executed successfully** message.

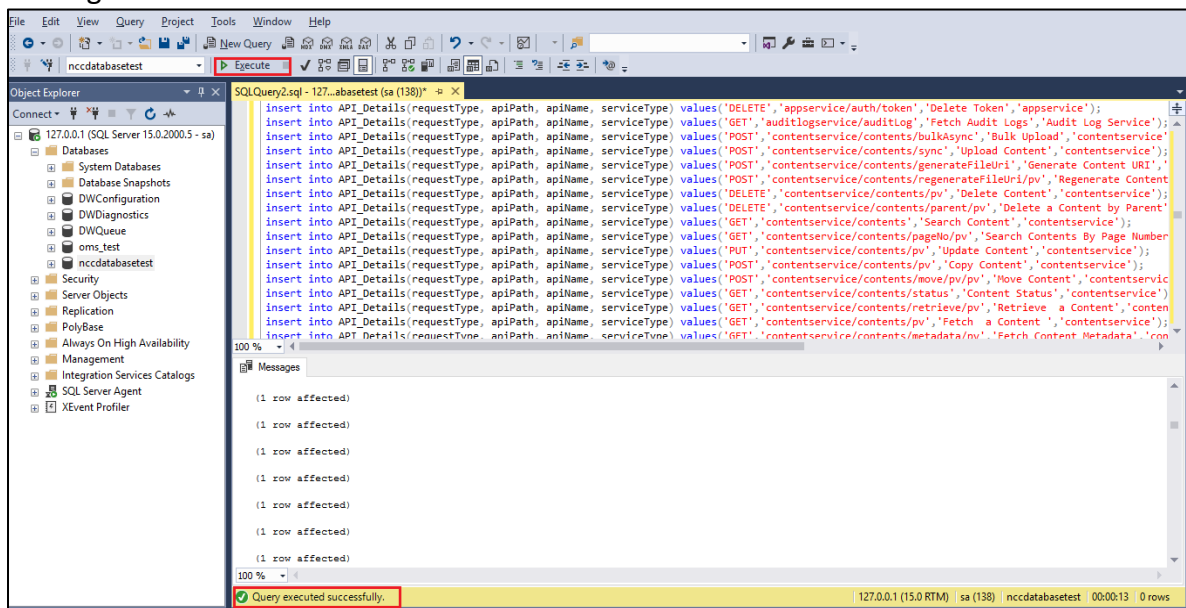


Figure 2.50

## 2.17 Configuring the SendGrid email service

To configure the SendGrid email service, perform the below steps:

1. On Azure portal home page, search for string **sendGrid** and click **Twilio SendGrid**.
2. On **Subscribe To Twilio SendGrid:Basics** page, fill in the relevant details. Change the plan to **50000 email per month**. Click the button **Review + subscribe**.
3. On **Review + subscribe** page, review the details and click the button **Subscribe**.

4. Click on **'Configure account now'** button.
5. Click on **'Create Identity'** button and then enter the details.
6. Go to **'Setting'** and then click on **'API Keys'** button. Click on **'Create API Key'**.
7. Write the API Key Name and choose **'Full Access'** in API Key Permissions. Click on **'Create & View'**.
8. Note down the API Key created and click on **'Done'**.

Follow below steps to create Dynamic Templates:

1. On the left sidebar go to **Email API** and then choose **Dynamic Templates**. Click on **'Create a Dynamic Template'**.
2. Choose the Dynamic Template Name from the Excel sheet provided and click on **'Create'**.
3. Go to created template and note down the Template ID. Click on **'Add Version'**.
4. In Your Email Designs, select **'Blank Template'**.
5. Select **'Code Editor'**.
6. In the terminal paste the html code of the selected template and in **'Settings'** write the **'Version Name'** and **'Subject'** provided in Excel sheet. Click on **'Save'**.

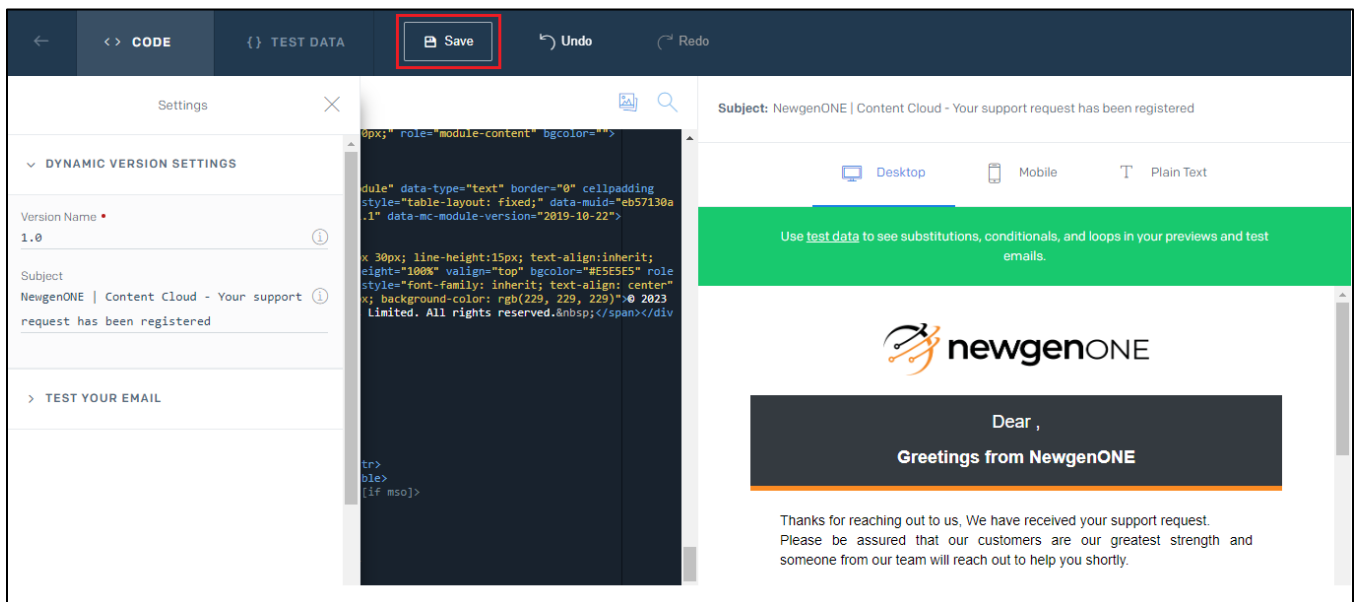


Figure 2.51

## 3 Deploying NCC containers

This section explains how to deploy the NCC containers.

### 3.1 Prerequisites

Ensure that the Azure Kubernetes Service is configured, and its worker nodes are in ready state.

---

**NOTE:**

Refer to the [Configuring Azure Kubernetes cluster](#) section to configure the Azure Kubernetes Service.

---

### 3.2 Overview

NCC (NewgenONE Content Cloud) is an Enterprise Content Management platform for creating, managing, searching, and collaborating large volumes of documents. It provides a highly scalable repository for securely storing and managing documents. It also supports exhaustive document and folder searches based on indexes and general parameters as well as Full Text Search (FTS) on image and electronic documents.

### 3.3 Deliverables

Newgen has isolated the product suite into multiple Docker containers to enable the independent scalability of each Docker container. This separation is done based on the product's functionality and micro-services separation. Newgen is releasing Docker images for all services, YAML files for deployment, and some documentation for end-to-end configurations and deployments.

Newgen product team delivers the following:

- Docker images
- YAML files

### 3.3.1 Docker images

The NCC 2024.1 release includes the below docker images for the initial product deployment:

- annotation-service
- app-service
- auditlog-service
- cabinet-service
- content-service
- dataclass-service
- document-download-service
- folder-service
- logging-service
- mailing-service
- microapi-service
- notes-service
- reporting-service
- roles-service
- search-service
- secretkey-service
- securityclassification-service
- storage-service
- storecredentials-service
- subscription-service
- tenant-service
- user-register-service
- zuul-service

### 3.3.2 YAML files

YAML files stands for “YAML Ain’t Markup Language”. It is a human-readable object configuration file that is used to deploy and manage the objects on the Kubernetes cluster. In other words, it is a manifest file that contains the deployment descriptor of Kubernetes containers. Execute YAML files using “kubectl apply -f <YAMLFile>” or use these files in Azure DevOps Release Pipeline to deploy the containers.

The yml files for NCC 2024.1 Docker images (PT environment) are as follows:

- annotation-service.yaml
- app-service.yaml
- auditlog-service.yaml
- cabinet-service.yaml
- content-service.yaml
- dataclass-service.yaml
- document-download-serv.yaml
- folder-service.yaml
- logging-service.yaml
- mailing-service.yaml
- microapi-service.yaml
- notes-service.yaml
- configmap.yaml
- reporting-service.yaml
- roles-service.yaml
- search-service.yaml
- secretkey-service.yaml
- securityclassification-service.yaml
- storage-service.yaml
- storecredentials-service.yaml
- subscription-service.yaml
- tenant-service.yaml
- user-register-service.yaml
- zuul-service.yaml
- AzureFile\_PV\_PVC.yml
- NGINX-IngressController.yaml

Here's an example of a YAML file:

```
apiVersion: v1
kind: Namespace
metadata:
  name: ncc-pt
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: storage-service
  namespace: ncc-pt
spec:
  replicas: 1
  selector:
    matchLabels:
      name: storage-service
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
  template:
    metadata:
      labels:
        name: storage-service
    spec:
      containers:
      - name: storage-service
        image: ncregistry.azurecr.io/storage_service:#{RELEASE.ARTIFACTS._STORAGE_SERVICE.BUILDID}#
        imagePullPolicy: Always
        securityContext:
          runAsNonRoot: true
        ports:
        - name: http
          containerPort: 8182
        env:
        - name: Release_Pipeline_Number
          value: "#{RELEASE.RELEASENAME}#"
        - name: kafka mailing topic name
```

Figure 3.1

**AzureFile\_PV\_PVC.yml** file is used for Persistent Volume and Persistent Volume Claim.

Persistent Volume (PV) is a storage piece in the cluster that is provisioned using Storage Classes. It contains the Azure FileShare **secretName** and **shareName** that is already created during Azure FileShare creation. It is also used to set the access permission on Azure FileShare using **mountOptions** attribute.

A PersistentVolumeClaim (PVC) is a request for storage by a user. It is similar to a Pod. Pods consume node resources and PVCs consume PV resources. Pods can request specific levels of resources (CPU and Memory). Claims can request specific size and access modes (e.g., they can be mounted ReadWriteOnce, ReadOnlyMany or ReadWriteMany).

```

apiVersion: v1
kind: PersistentVolume
metadata:
  name: azurefile-pv
  namespace: ncc-pt
spec:
  capacity:
    storage: 10Gi

```

Figure 3.2

**NGINX-IngressController.yaml** is used for the ingress controller. An ingress controller is an object running inside the Kubernetes cluster that is used to manage the host-based routing rules.

### **configmap.yaml :**

(ref. <https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/>)

The ConfigMap API resource stores configuration data as key-value pairs. The data can be consumed in pods or provide the configurations for system components such as controllers. ConfigMap is similar to Secrets, but provides a means of working with strings that don't contain sensitive information. Users and system components like can store configuration data in ConfigMap.

The ConfigMap's data field contains the configuration data. Below is the example, this can be simple -- like individual properties defined using `--from-literal` -- or complex -- like configuration files or JSON blobs defined using `--from-file`.

**apiVersion:** v1

**kind:** ConfigMap

**metadata:**

**creationTimestamp:** 2016-02-18T19:14:38Z

**name:** example-config

**namespace:** default

**data:**

*# example of a simple property defined using --from-literal*

**example.property.1:** hello

**example.property.2:** world

*# example of a complex property defined using --from-file*

**example.property.file:** |-

*property.1=value-1*

*property.2=value-2*

*property.3=value-3*

You must create the ConfigMap object before you reference it in a Pod specification. Alternatively, mark the ConfigMap reference as optional in the Pod spec (see [Optional ConfigMaps](#)). If you reference a ConfigMap that doesn't exist and you don't mark the reference as optional, the Pod won't start. Similarly, references to keys that don't exist in the ConfigMap also prevents the Pod from starting, unless you mark the key references as optional.

---

**NOTE:**

You can store the above YAML files in Azure Repo that Azure DevOps Release Pipeline can utilize further.

---

## 3.4 Changing Product's YAML files

The changes in the Product's YAML files are:

- **Namespace:** In the YAML files, default namespace is given as **ncc-pt**. You can change this name as per your choice.

```
apiVersion: v1
kind: Namespace
metadata:
  name: ncc-pt
```

Figure 3.3

- **Name:** In the **storage-service-pt.yaml** file, **storage-service** is given as the default name of Kubernetes objects: **deployment, replica-set, container, and service**. You can change this name as per your choice. But make sure that this name must not be more than 13 letters and this name must be in small letters only. For example,
- **Replica:** In the **storage-service-pt.yaml** file, the default replica is given as **1**. Th one container gets created after the deployment. You can increase this number as our choice/requirements.
- **Image:** In the **storage-service-pt.yaml** file, update the **image** location. By default the below value is given

```
image: nccregistry.azurecr.io/storage_service:#{RELEASE.ARTIFACTS._STORAGE_SERVICE.BUILDID}#
```

Figure 3.4

Where,

**nccregistry.azurecr.io:** is the name of the Azure Container Registry.

**storage\_service:** is the iBPS5 ServiceInstance WEB Docker image name.



- **#{RELEASE.ARTIFACTS.\_IBPS5SERVICEINSTANCEWEB.BUILDID}#**: is a Docker image's tag name in the form of a dynamic variable whose value gets updated at runtime using AzureDevOps Release Pipeline. You can also specify the static tag name like latest, build-number1, and more.
- **SecurityContext**: In the **storage-service-pt.yaml** file, **SecurityContext [runAsNonRoot: true]** is defined that means **storage\_service** container can never be run with root privileges. If any container tries to run with the root user, then Kubernetes stop its deployments.

For example,

```
securityContext:
  runAsNonRoot: true
```

Figure 3.5

- **VolumeMount & Volume**: Volume mounts and volumes are used to persist the data outside the container so that whenever the container terminates due to any reason our data is always persisted. In the **storage-service-pt.yaml** file, there's persisted configuration files or folders and log files.

For example:

```
volumeMounts:
- mountPath: /var/tmp/
  name: contentazurefileshare
```

Figure 3.6

In volumeMounts, **mountPath** is a path inside the container that is being mounted. Here, mountPath cannot be changed as this structure is predefined in a Docker container. **subPath** works as a relative path that is appended to the attached persistent volume's shareName. **subPathExpr** is used to segregate the product logs container wise. And the **name** is a user-defined name that must be matched with the name specified in volumes.

In volumes, **azurefile** is the persistent volume claim name.

- **Ports**: In the **storage-service-pt.yaml** file, containerPort **8182** is specified that means only 8182 port is exposed outside the container and no other port is accessible from outside.
- **ImagePullSecret**: ImagePullSecret is a secret value that is used to pull an image from a private container repository like Azure **Container Registry**.
- Execute the below command to create an ImagePullSecret:

```
kubectl create secret docker-registry registry-secret --docker-server nccregistry.azurecr.io --docker-username=newgencontainerregistry --docker-password kmPF/ytfFu5q6NazqvVYtJ???????
```

ImagePullSecret can also be created from Azure DevOps Release Pipeline.

---

**NOTE:**

- annotation-service.yaml
  - app-service.yaml
  - cabinet-service.yaml
  - dataclass-service.yaml
  - folder-service.yaml
  - zuul-service.yaml
  - mailing-service.yaml
  - notes-service.yaml
  - roles-service.yaml
  - secretkey-service.yaml
  - storecredentials-service.yaml
  - tenant-service.yaml
  - app-service.yaml
  - auditlog-service.yaml
  - content-service.yaml
  - document-download-service.yaml
  - user-register-service.yaml
  - logging-service.yaml
  - microapi-service.yaml
  - reporting-service.yaml
  - search-service.yaml
  - securityclassification-service.yaml
  - subscription-service.yaml
- 

## 3.5 Changes in Application Gateway Ingress YAML Files

Changes in Application Gateway Ingress YAML Files are as follows:

- Along with the product's YAML file, AppGateway Ingress Controller's YAML file '**AppGateway-IngressController.yaml**' are also be shared. Using an ingress controller and ingress rules, a single IP address can be used to route traffic to multiple services in a Kubernetes cluster. The AppGateway Ingress Controller creates a Load Balancer with its external IP and routes the incoming requests to the target Kubernetes services according to the host-based routing rules. Host-based routing is a capability of Ingress Controller that redirects the user requests to the right service based on the request-host header.

For example, set the rules as below:

- IF URL is 'ncc.newgendocker.com' THEN redirect to NCC Zuul Service container.

---

**NOTE:**

To support the host-based routing, register a domain and create a new RecordSet in DNS Zone for each host-path. Refer to the document "Configuration of Azure Kubernetes Cluster" to see the configuration of Application Gateway Ingress Controller and DNS Zone.

---

- Once Application Gateway Ingress is configured and RecordSets are created in DNS Zone, and an SSL certificate is generated, deploy the Ingress controller along with its ruleset using the YAML file.

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: appgw-ingress
  namespace: ncc-qa
  annotations:
    kubernetes.io/ingress.class: azure/application-gateway
    appgw.ingress.kubernetes.io/ssl-redirect: "true"
    appgw.ingress.kubernetes.io/request-timeout: "300"
    appgw.ingress.kubernetes.io/ssl-policy: "AppGwSSLPolicy20170401S"
    appgw.ingress.kubernetes.io/health-probe-status-codes: "200-399,404"
spec:
  tls:
    - hosts:
      - newgendocker.com
      - secretName: ncc-cert
  rules:
    - host: ncc.newgendocker.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: zuul-service
                port:
                  number: 8197

```

Figure 3.7

- In the *AppGateway-IngressController.yml* file, there are multiple host-based rules defined. A few of them are listed below:
  - **ncc.newgendocker.com**

If the host URL is 'ncc.newgendocker.com' then it redirects the user request to the **zuul service** container which is running on port 8197. Here, zuul-service is the name of the Zuul Service container.
- In this YAML file, change the host URL, ServiceName, ServicePort, and the name "**name: appgw-ingress**" as per our choice.
- In this YAML file, it is also defined SSL/TLS configuration via specifying the **tls** spec along with hosts and secretName.

```

spec:
  tls:
    - hosts:
      - newgendocker.com
      - secretName: ncc-cert

```

Figure 3.8

- You can specify the valid DNS against hosts. I.e. **newgendocker.com**

- Before deploying the ingress controller, create a Kubernetes secret to host the certificate and private key. Execute the below command to create a Kubernetes secret:

```
kubectl create secret tls <secret-name> --key <path-to-key> --cert <path-to-crt> -n <Namespace>
```

For example,

```
kubectl create secret tls appgw-cert --key azure.key --cert azure.crt -n ncc
```

- After making the required changes as per our choice, deploy the Ingress controller by executing this YAML file using below command or you can configure it to AzureDevOps Release Pipeline.

```
kubectl apply -f AppGateway-IngressController.yml
```

---

**NOTE:**

To execute the above command, kubectl must be configured on your local server. Refer to the “Configuration of Azure Kubernetes Cluster” document to run kubectl from your local machine.

---

## 3.6 Changes in Configmap.yaml file

This section consists of the configmap.yaml file changes and prerequisites.

### Prerequisites:

- You need to have Kubernetes cluster, and the kubectl command-line tool must be configured to communicate with your cluster.
- A ConfigMap is an API object that store configuration for other objects to use.

The changes are as follows:

- Update the **spring\_data\_mongodb\_uri** and **spring\_data\_mongodb\_database** in configmap.yaml file in the form of key-value pair.

For example:

```
mongo_uri_encrypted: 176F77EB6CDF2209F15BFC6105DBD5E904ABC19C26F3F3F25B62D7696AE
F2AEA05D58FB02C02618D94477422A019EE5F8682913324DF6EFC4EB8193D352C647ECC35D78350B
EC509473648E7F0164D949CD209EF1308E918D5D0DD8B8707E9D509D438B06FC47A790B8630D13A1
71949E726583A6147ABC3A9A1EB9FA611B80F98B88B1C461C2DF45D532BB7DDF187F5E0584224DBA
31688268017C0D4834DF5B7A05F5BA20104F29B00B19C7A833F900E357159FB59A5E688387C1D57B
E5CA90A2563DF32C7CE71CFB51ED1C767CDD68A6B2969AAD075ECA9EFEC711508E3AB15D5B22A1A3
1D93A0AA44D87DBE47A3562C543A11094F494648A7BA9EC517140
mongo_uri_sk: ECBA5F308D616A4A316DA74702905FD6AE56BBC7211C4AD0A46FEF59FA222144A9
26656DD28CE92E45E33696567FB2ED
spring_data_mongodb_database : ecmdata
spring_data_mongodb_database2: ecmdataext
```

- Update the following keys and their values in configmap.yaml:

```

frontend_app_url : https : //ncc-pt-webapp.azurewebsites.net
appgateway_url : https : //nccpt.newgendocker.com/ecmapi
spring_kafka_bootstrap_servers : event-hub-nccpt.servicebus.windows.net:9093
spring_kafka_properties_sasl_jaas_config :
org.apache.kafka.common.security.plain.PlainLoginModule required
  username="$ConnectionString"
password="Endpoint=sb://event-hub-
nccpt.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;Shar
edAccessKey=085phRTXXXAn0n5ymC8MIXXXR9oz0I+AEhIffMbk=";

```

- Update the following keys and their values in configmap.yaml:

```

logging_level_root: "OFF"
logging_level_org_springframework_web: "OFF"
logging_level_org_hibernate: "OFF"
logging_level_com_newgen: "OFF"
logging_level_com_newgen_service: "OFF"
easysearch_enabled: "false"
paperprofile_enabled: "true"
corrus_url: https://ncc-pt-webapp.azurewebsites.net/

```

- Update the following keys and their values in configmap.yaml:

```

spring_redis_password: r8Bqi9cTiTSJZXXXxS5njCI36HDX3bXXXXB8vQ0=
spring_redis_host: nccreXXXXche.redis.cache.windows.net
spring_redis_ssl: "true"
spring_redis_port: "6380"
od_integration: "false"
cron_job1_initial_delay: "14000"
cron_job1_fixed_delay: "3600000"
mailing_sendgrid_apiKey: SG.evBAsDXXXXX3vbAwZdUg.PDTXXXH_kkrF-3VaOXXA
spring_profiles_active: default
spring_data_mongodb_shared: "true"
pagination_batchSize: "5"
version: "1.0"

```

- Update the following keys and their values in configmap.yaml:

```

fts_azure_SearchServiceName: ncc-pt-search
fts_azure_SearchServiceAdminKey: gFttGAAJG19P9YCgE3esWOeu1Hyty99Z513QQpmT1H
fts_azure_SearchServiceQueryKey: EVrIwzIk20eU2aWiKMOB7792LklGYRBjDFAzSeCz8pgo
fts_azure_ApiVersion: "2020-06-30"
fts_azure_IndexName: ncc-pt-search-index1
fts_azure_ConnectionString:
DefaultEndpointsProtocol=https;AccountName=nccptblobstorage;AccountKey=6IGmGB1IS
SKPFXXXXXXXXXXXXXXXXlpXtYIG4LFKuV/CmX+gZt3qXXXXXXXXXXXXXXXXhd1bRcAeIMniBo+AStPkshCA==;En
dpointSuffix=core.windows.net
spring_jackson_serialization_FAIL_ON_EMPTY_BEANS: "false"
base_url: https://ncc-pt-webapp.azurewebsites.net

```

```

auth_whitelist_domains: https://ncc-pt-
webapp.azurewebsites.net/tenantreport/login,https://ncc-pt-
webapp.azurewebsites.net/documentation/login,https://ncc-pt-
webapp.azurewebsites.net/sampleapp/login,https://ncc-pt-
webapp.azurewebsites.net/microUI/login,https://ncc-pt-webapp.azurewebsites.net/
storage_account_key:
"6IGmGBLISSKPFRRXXXXXXXXXXXXtYIG4LFKuV/CmX+gZt3qKxxxxxxhd1xxxxxIxxxxxxx=="
storage_account_name: "nccptblobstorage"
storage_protocol: "https"
mailing_sendgrid_emailFrom: vivek_kumar@newgensoft.com
service_mailing_serviceId: mailing_service
spring_datasource_url:
jdbc:sqlserver://nccpt.database.windows.net:1433;database=nccptsql
spring_datasource_username: u*****
password_encrypted: 5633632D0EB375CBB5B694CAB62E35FC
source_sk: CF216A0D103D848D1B7A5F011A9187BB8AF832FDCCF03CA12D901DA3E5EC841721535
958A8F0120D0AE22B0E62BF59EE
register_user_subject: Welcome To Newgen Content Cloud

```

- Update the following keys and their values in configmap.yaml:

```

ad_auth_userName: ecmadmin@ecmnextad.onmicrosoft.com
ad_auth_password: NCC145%
ad_auth_authority: https://login.microsoftonline.com/common/
ad_auth_client_id: 9b9fXXX4-4229-XXXXXX-86e5-03XXXa29b6
ad_auth_tenant_id: 10c15XXX2-f825-45XXX-a7a7-12faXXXf68a
ad_auth_client_secret: r78y.-K.14q3vYp-92~wJQa5014QBR4mm_
ad_auth_scope: https://graph.microsoft.com/.default
ad_auth_rootuserprinciplename: '@ecmnextad.onmicrosoft.com'
auth_private_key:
/lY+vXQLZbDZhDIh5FKa+UYofIKEFKHVe7uELiPPNniTI0EzvhmYo+xCjJtwnDaRAgMBAAEcggEAHtax
9SHC+odxkDu4StYYK9c/gSjkmrBL9gCs9w1wE7agyQAqbk4x1LU5TaHgvZ/0wD2dnpuNa/ndjSTdq28k
HGC4HeL7/XXXXXXXXXXXXXXXXDZDNZUI0G5HH81zlplJ1qMshP2utfdFGihfSARue0wrxeNt5MBICBv6fKqw
S0Q5xmIkcIBBqhRYorC4TpinYpYJuFxFAELJUSr34SZgfjsWhxVO+XXXXXXXXXXXX/TINOiZaGvHGZm
MXEWDQuOh2HBM618rL9lrXkA0A8NNAtahDsbBOJROND4L2vUcqCbhNYcj4VtFiBJv170rT2mBXbd8w2X
XXXXXXXXXXNapyfVTrfIefSHuHUXLXcLhoRLg+ruQ7VseKKhMG+YitFDLEsU1SdJUeTQuC/DE4e1LuidQ
hfSmNOwD85AVLfbZ0PxmYh5Yp61aVm

```

- Update the following keys and their values in configmap.yaml:

```

corrus_auth_url: https://ncc-pt-
webapp.azurewebsites.net/EasyCMRest/validateSession
corrus_qa_auth_url: https://ncc-pt-webapp.azurewebsites.net/EasyCMRest
kafka_logging_topic_name: "logging-event-hub-dev"
topic_name: "logging-event-hub-dev"
kafka_mailing_topic_name1: "tstoragecredentials"
kafka_mailing_topic_name2: "appservice"
kafka_mailing_topic_name: "event-hub-dev"
topic_name2: "event-hub-dev"
topic_name3: "paper-profile-topic"

```

```

topic_name1: "bulk-file-upload"
topic_paper_profile_name: "paper-profile-topic"
spring_application_name: "tenant-service"
spring_kafka_topic_name: "TenantService"
audit_logs_topic_name: "audit-log-events"
role_service_url: http://roles-service:9900
logging_service_url: http://logging-service:8198
user_service_url: http://user-register-service:9196
folder_service_url: http://folder-service:8185/
upload_folder: /var/tmp/
storage_service_url: http://storage-service:8182
search_service_url: http://search-service:9201
dataclass_service_url: http://dataclass-service:9183
subscription_service_url: http://subscription-service:9195
tenant_service_url: http://tenant-service:9197
content_service_url: http://content-service:8184
roles_service_url: http://roles-service:9900
tenant.service.url: http://tenant-service:9197
alarm_service_url: http://notes-service:9177
cabinet_service_url: http://cabinet-service:8183
storageCredentials_service_url: http://storecredentials-service:8186
secret_service_url: http://secretkey-service:9121
app_service_url: http://app-service:9902
classification.service.url: http://securityclassification-service:9901
usergroup_profile_service_url: http://user-group-service:8189
mail_service_url: http://mailing-service:8202
storage_name: nccptblob

```

- Update the following keys and their values in configmap.yaml:

```

zuul_routes_cabinetservice_url: http://cabinet-service:8183
zuul_routes_storageservice_url: http://storage-service:8182
zuul_routes_folderservice_url: http://folder-service:8185
zuul_routes_contentservice_url: http://content-service:8184
zuul_routes_storagecredentialsservice_url: http://storecredentials-service:8186
zuul_routes_annotationsservice_url: http://annotation-service:8282
zuul_routes_dataclassservice_url: http://dataclass-service:9183
zuul_routes_loggingservice_url: http://logging-service:8198
zuul_routes_dataclassreportservice_url: http://dataclass-report-service:9189
zuul_routes_usergroupservice_url: http://user-group-service:8189
zuul_routes_usergrouplogicservice_url: http://user-group-logic-service:8191
zuul_routes_subscriptionservice_url: http://subscription-service:9195
zuul_routes_tenant-service_url: http://tenant-service:9197
zuul_routes_userregisterservice_url: http://user-register-service:9196
zuul_routes_mailing-service_url: http://mailing-service:8202
zuul_routes_filesharingservice_url: http://file-sharing-service:8888
secretkey_service_url: http://secretkey-service:9121
zuul_routes_documentdownloadservice_url: http://document-download-serv:9199
zuul_routes_searchservice_url: http://search-service:9201
zuul_routes_secretkeyservice_url: http://secretkey-service:9121

```

```
zuul_routes_rollesservice_url: http://roles-service:9900
zuul_routes_appservice_url: http://app-service:9902
zuul_routes_classificationsservice_url: http://securityclassification-
service:9901
zuul_routes_prospectservice_url: http://prospect-service:9122
zuul_routes_microapiservice_url: http://microapi-service:9123
zuul_routes_reportingservice_url: http://reporting-service:8098
zuul_routes_auditlogservice_url: http://auditlog-service:8199
zuul_routes_notesservice_url: http://notes-service:9177
```

- Update the template IDs keys and their values in configmap.yaml:

```
email_admin_moremb: d-552757ec05944e9097c3009fa8d29c84
email_tenant_moremb: d-5baf0145c0dd4dfdacab6b1e90fe2582
email_admin_tenpercent: d-eeb37a36cc67405995f04cc0356c0600
email_tenant_tenpercent: d-78d20eb9c7df447ba1e7564cb92cd3ac
email_admin_emailForTenantTrialExpiry: d-67a7705612574516b587a3e4a194dc04
email_tenant_emailForTenantTrialExpiry: d-25390a8f8b4945b7bfffcc4d810884acf
email_admin_emailForTenantTrialExpiry2DaysBefore: d-
9d3e756dfd944129a15b1706e04a1b88
email_tenant_emailForTenantTrialExpiry2DaysBefore: d-
9d3e756dfd944129a15b1706e04a1b88
email_admin_buisnessPlanExpiry: d-646d5e3df31a4b4d85cb043a47530d9e
email_tenant_buisnessPlanExpiry: d-7d8297f18ee9497ebcea3f5665069c22
email_admin_moreThan10kDocs: d-acca677c07e0436889c715cc262a7a11
email_tenant_moreThan10kDocs: d-5e8654ad4a9a4beb8d796ba2451e85e9
email_admin_tenantPlanExPostGrace: d-44dc3856170044eb835f2929d802ddff
email_tenant_tenantPlanExPostGrace: d-ba9aa6810fa345d48f556b01d2595752
email_tenant_clientFeedback: d-574f33d558584b29a0c1556fa3d01b01
email_tenant_tenantPlanExPostGraceWeek1: d-b794865f1446424b93be842f7272eaaa
email_tenant_tenantPlanExPostGraceWeek2: d-2508ae5e1e1743739c0f4794cacce8a2
email_tenant_tenantPlanExPostGraceWeek3: d-38cb2cc0fd0f418da63a8247d5d11540
email_tenant_tenantPlanExPostGraceWeek4: d-1cba3f66a9a04c52996e6a28c586f84f
email_admin_emailForSupport: d-21b14f6920a24005a11150b343dd42e7
email_tenant_emailForSupport: d-e58c5e43efb4483c9987c326f0a99ee5
email_admin_tenantUpbeforetrialends: d-b5890098cf494d84a42247cd2c2f3d82
email_tenant_tenantUpbeforetrialends: d-7dff0722ee6e44e4ad9cde46f0ff5000
email_admin_tenantExistingPlan: d-a8fc662fc57d4cb888d888f6fe17d84c
email_tenant_tenantExistingPlan: d-68a643470e8f4118a2a0ee1e19e13cf3
email_admin_tenantAddOnEmail: d-93630c9fae0d4ad08a9e2bb43671e148
email_tenant_tenantAddOnEmail: d-717fce7ae8524f4caf9dc8aa0f628e5c
forgot_password_templateId_logging: d-db7374bcf4d54ae8b48b32a88ab5567d
email_admin_contactSales: d-a485ad93c8be4507a7ee5a808a6bd69c
register_user_templateId: d-1db25eac05614fa9a5672322b1c4b102
otp_templateId: d-5bd5df83eb0f484bbfffc5fa0bcea8ac5
invite_user_templateId: d-3de95835fb6746eeaf32a536949b7878
forgot_password_templateId_user: d-934f1bf295f04eec8010a3649bda3840
```



## 3.7 Deploying containers

This section explains how to deploy the containers:

1. Deploy the containers on Azure Kubernetes Service from our local machine by executing the below command or deploy them using Azure DevOps Release Pipeline.
2. Deploy the containers using Azure DevOps for better traceability.

```
kubectl apply -f <YAML_File>
```

For example,

```
kubectl apply -f ServiceInstanceWeb.yml
```

---

### NOTE:

- To execute the above command, kubectl must be configured on your local server. [Refer to the [Configuring Azure Kubernetes Cluster](#) section to run kubectl from your local machine].
  - To deploy the containers using Azure DevOps Release Pipeline, Azure DevOps must be configured. [Refer to the [Configuration of Azure DevOps Release Pipeline for AKS](#) document].
- 

3. In AzureDevOps, a separate Release pipeline is created for each Docker images corresponding to each service deployed.

- annotation-service
- app-service
- auditlog-service
- cabinet-service
- content-service
- dataclass-service
- document-download-serv
- folder-service
- microapi-service
- zuul-service
- logging-service
- mailing-service
- notes-service
- reporting-service.yaml
- roles-service
- search-service
- secretkey-service
- securityclassification-service
- storage-service
- storecredentials-service
- subscription-service
- tenant-service
- user-register-service

4. Trigger the Release Pipeline to deploy the required Docker containers.
5. Once the deployment is done, deployed containers can be visible from the Kubernetes Dashboard. Refer to the [Configuring Azure Kubernetes Cluster](#) section to configure the Kubernetes Dashboard

## 3.8 Configuring the Azure DevOps release pipeline (NCC)

This chapter describes the configuration of Azure DevOps Release Pipeline. Refer to the below sections for procedural details.

### Overview

The Build Pipeline and Release Pipeline are separated into two parts. Build Pipeline is done through the Jenkins server which can be installed on an on-premises machine or a cloud machine. Using the Azure DevOps Release Pipeline cloud service, you can manage the Release pipeline. In this architecture, three stages are created that is, Dev, UAT, and Production and in each stage, deployment is quite different. You can have some more stages depending on the requirements. This document describes the configuration of the Azure DevOps Release Pipeline for container deployment on Azure Kubernetes Service (AKS).

### CICD pipeline architecture

1. The Newgen representative builds the product's base Docker images on the company's on-premises servers using Jenkins.
2. As soon as the Dev team commits the code to the source code repository, the Jenkins pipeline gets triggered. It pulls the code then compiles them and prepares the build artifacts as well as creates Docker images and pushes the newly created Docker images to the Azure Container Registry.
3. As soon as any Docker image is pushed to the Azure Container Registry, Azure DevOps Release Pipeline triggers the deployment to the Dev environment. Here, you can configure the performance testing as well as security testing of the application. In Addition, you can perform manual testing as required.
4. UAT and Production deployments are based on approval and are available on-demand. To deploy to the UAT environment, you need to trigger the UAT deployment. Upon deployment trigger, an approval mail is sent to the project manager or the concerned team. As soon as the project manager approves the go-ahead, UAT deployment gets started automatically.
5. Production deployment is also based on approval, but it is multi-level approval. To deploy a production environment, you require the approval of all stakeholders, and the production environment doesn't get triggered automatically on receiving all the approvals. A manual intervention mail is sent to the engineer who is supposed to deploy to production with a checklist. During deployment, all the checklist points get verified before performing the production deployment. In case any point of the checklist is not covered, then deployment to the production gets rejected.

## 3.9 Configuring the Azure DevOps

Perform the below steps to configure Azure DevOps:

1. Sign in to the Azure DevOps portal at <https://azure.microsoft.com/en-in/services/devops/>
2. After a successful sign in, click **New Project** to create a new project.

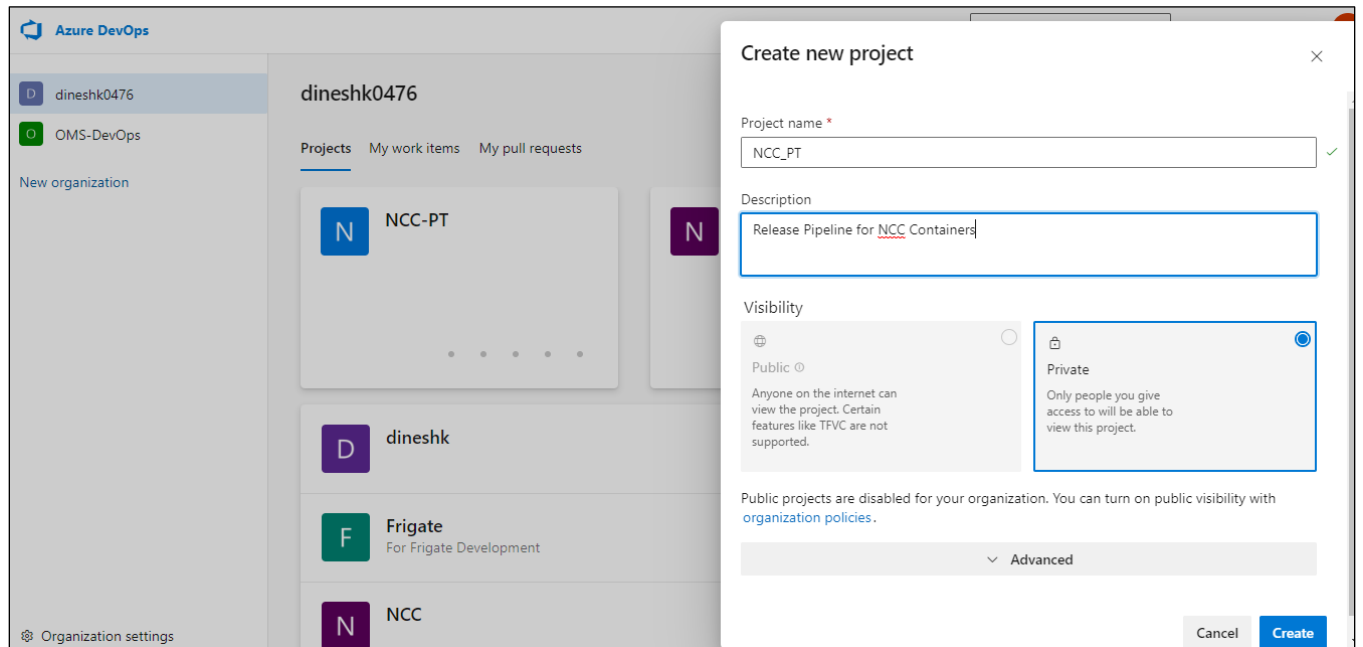


Figure 3.9

3. Specify the **Project name**, and **Description**.
4. Select the **Visibility** as **Private** and create the Azure DevOps Release Pipeline for different Docker Images.

## 3.10 Configuring the release pipeline

This section explains how to create Release Pipeline

### NOTE:

- annotation-service
- app-service
- auditlog-service
- cabinet-service
- content-service
- dataclass-service
- document-download-serv
- folder-service
- microapi-service
- zuul-service
- logging-service
- mailing-service
- notes-service
- reporting-service.yaml
- roles-service
- search-service
- secretkey-service
- securityclassification-service
- storage-service
- storecredentials-service
- subscription-service
- tenant-service
- user-register-service

To create Release Pipeline, perform the below steps:

1. After project creation, the project summary screen appears. Hover over the **Repos** and select **Files**.

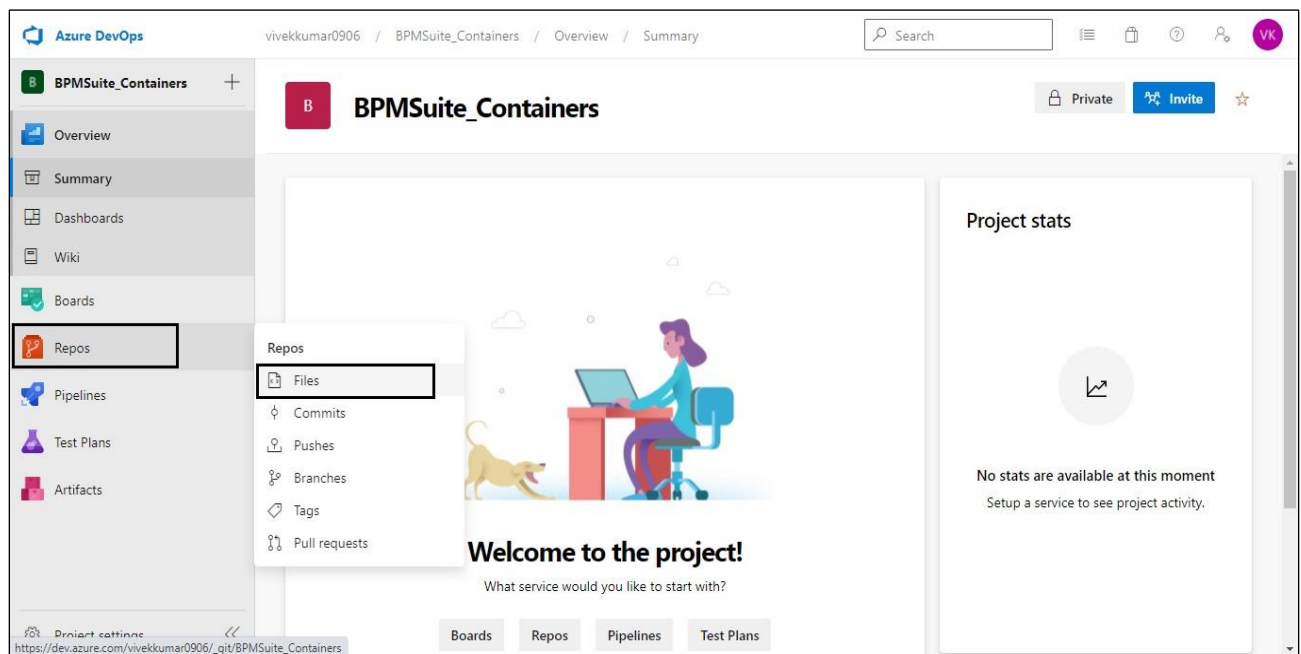


Figure 3.10

2. Click **Initialize**.
3. Click **More actions** and then select the **Upload file(s)**.

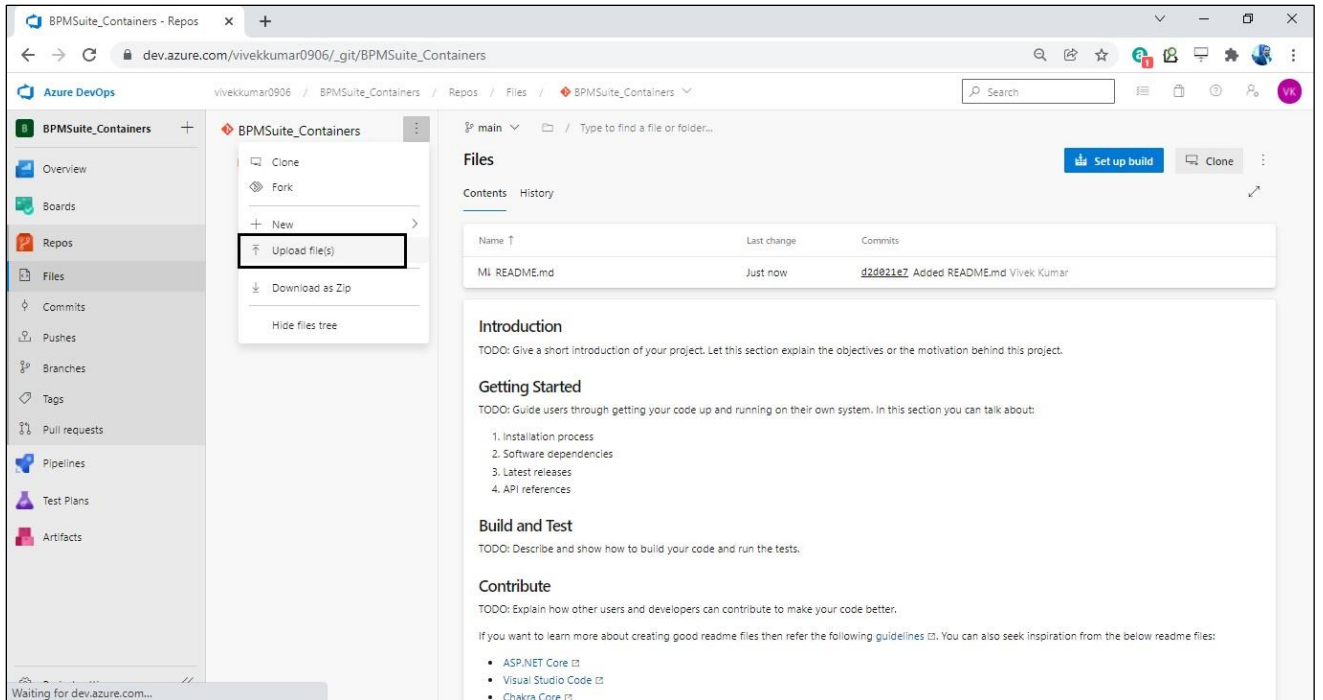


Figure 3.11

4. Browse or drag and drop all the YAML files that have been shared and then select **Commit**.
5. Hover over to the **Pipelines** in the left panel and select **Releases**.

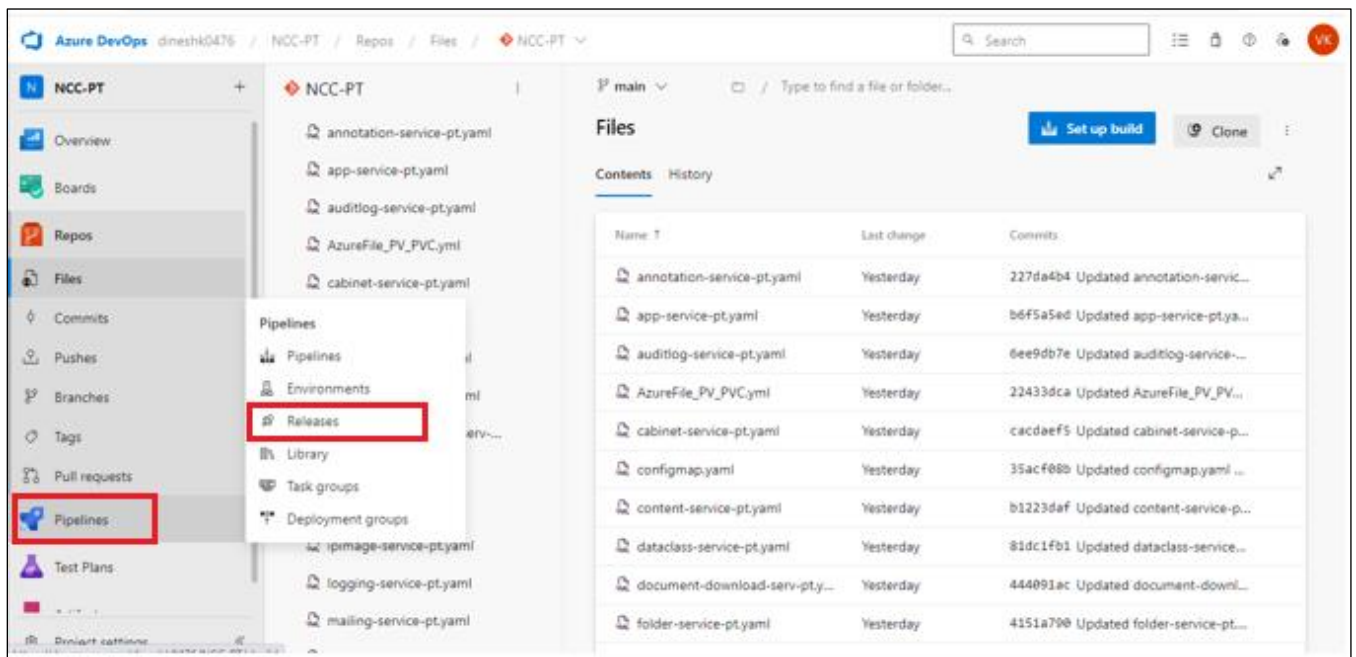


Figure 3.12

6. Click **New Pipeline** button. **Select a template** dialog appears.
7. Select the **Deploy to a Kubernetes cluster** template.

8. Click **Apply**. The **Stage** panel appears.
9. Specify the **Stage name** and click **close** icon to close the dialog.
10. Enter the unique name for your pipeline and click **Save**.
11. Specify **Comment** and click **OK** on the **Save** dialog box.
12. Click **Add an artifact**. The **Add an artifact** dialog appears.
13. Click **Azure Container Registry** under the **Source** type.
14. Select the **Service connection** which authenticates the Azure Container Registry.
15. In case **Service connection** is not created, follow the below steps to create Service connection:  
**Configuration of Service connection for Azure Container Registry:**
  - Click **Manage** link. The **Create service connection** page appears in a new tab.
  - Click **Create service connection**. The New service connection dialog appears.
  - Select **Azure Resource Manager** as the connection type and click **Next**.
  - Select **Service principle (automatic)** as the **Authentication method**.
  - Specify the following parameters:
    - **Subscription** as Scope level.
    - Select an existing **Azure subscription**.
    - Select the **Resource Group** in which Azure Container Registry is created.
    - (Optional) Specify the **Service connection** name and **Description**.
    - Select the checkbox **Grant access permission to all pipelines**.
  - Click **Save**. Once the service connection is created, it appears in the list.
16. If the **Service connection** is already created, then select the created service connection.
17. Select **Resource Group** from the list in which Azure Container Registry is created.
18. Select the created **Azure Container Registry**.
19. Select a Docker image for example, **ibps5serviceinstanceweb** as a **Repository**.
20. Select **Latest** as the **Default version**. Leave the **Source alias** with its default value and click **Add**.

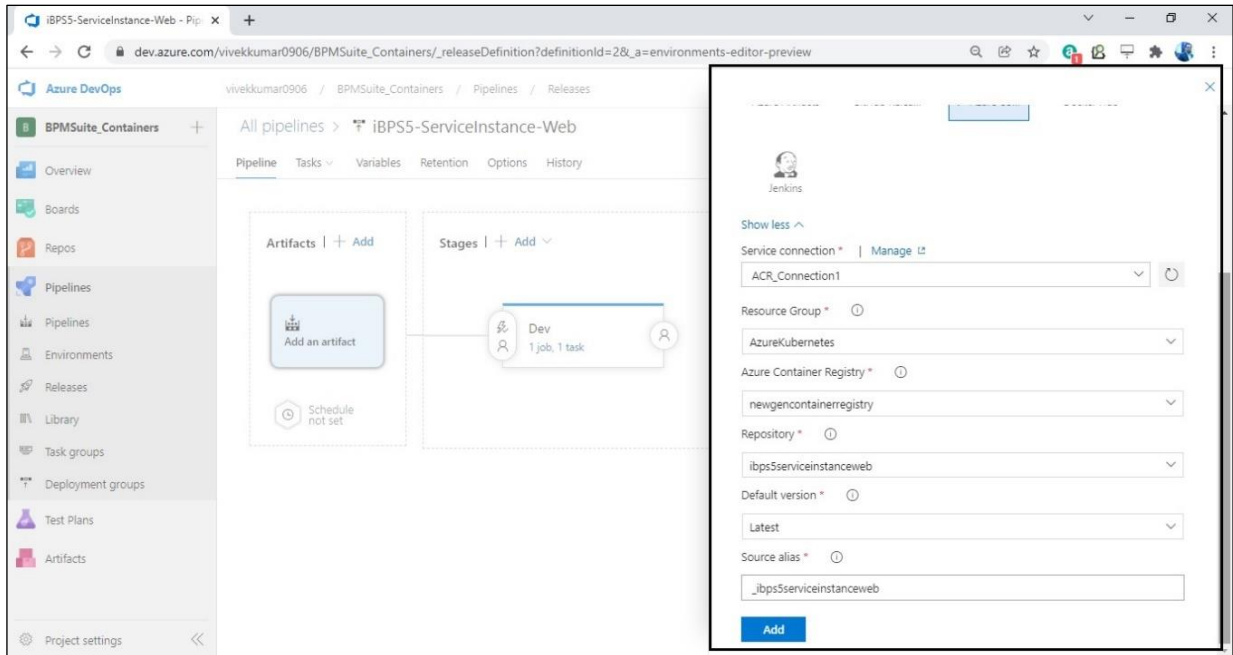


Figure 3.13

21. Once the artifact is added, it appears in the **Artifacts**. Click the **Continuous deployment trigger** icon. The **Continuous deployment trigger** dialog appears.

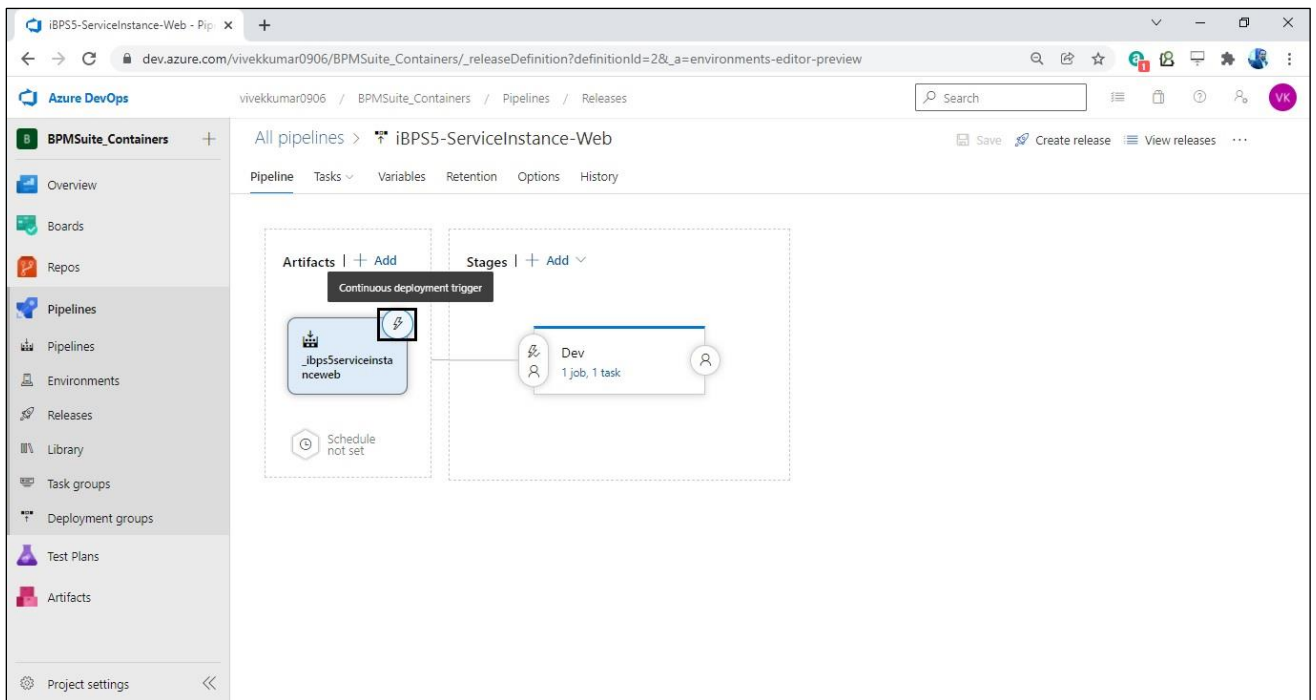


Figure 3.14

22. Enable the **Trigger** and specify the **Tag filter**.

For example,

**^latest\$** - trigger the release only if the tag is "latest"

**v1\.[0-9]** - trigger the release for tags like "v1.23", "beta-v1.3-test"

23. Click Close icon to close the Continuous deployment trigger dialog.

24. Click **Save**.

25. Click **Add an artifact**. The **Add an artifact** dialog appears.

26. Click the **Azure Repos** under the Source type.

27. Select the **project, Source (repository)** and default branch **main**. Also, keep the other settings as default.

28. Click **Add** then **Save**.

29. Configure three stages: Dev, UAT, and Production, and on each stage deployment process is different. You can have some more stages depending on the requirements.

## 3.11 Configuring the Dev stage

To configure the Dev Stage, perform the below steps:

1. Click **View stage tasks**.

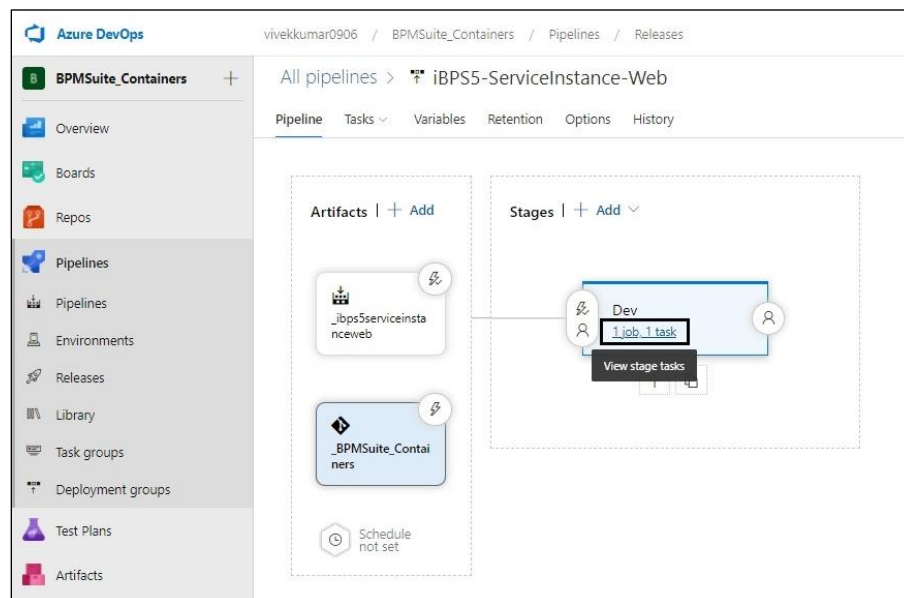


Figure 3.15

2. Click **Agent Job** and then select **ubuntu** as the **Agent Specification**.

3. Click **Add a task to Agent Job** + icon and search for the **Replace Tokens** and add them.

---

### NOTE:

Ensure **Replace Tokens** task must be the 1<sup>st</sup> task under the Agent Job.

---



4. Click **Browse Root Directory** icon under the Replace Token settings.
5. Select appropriate YAML file (for example, *iBPS5.0ServiceInstanceWeb.yml*).
6. Copy the content of the **Root directory** and paste it to the **Target files** textbox.
7. Leave the other settings as default and click **Save**.
8. Click **KubectI** task under the **Agent Job**.
9. Select Task version **1** and specify the **Display name**.
10. Select **Kubernetes Service Connection** as the Service connection type.
11. Select **Kubernetes service connection** which authenticates kubectI to interact with the Kubernetes cluster.
12. If **Kubernetes service connection** is not created, then follow the below step to create Kubernetes service connection.
13. **Configuration of Kubernetes service connection.**
  - Click **Manage** link. The **Service connections** page appears in a new tab.
  - Click **New service connection** or **Create service connection**. The **New service connection** dialog appears.
  - Select **Kubernetes** and click **Next**. The New Kubernetes service connection dialog appears.
  - Select **KubeConfig** as an **Authentication method**.
  - Copy the content of **KubeConfig** file.

---

**NOTE:**

You can get the KubeConfig file by executing below command:

```
az aks get-credentials --resource-group <ResourceGroupName> --name  
<AzureEKSClusterName>
```

For example,

```
az aks get-credentials --resource-group AzureKubernetes --name BPMSuite-AKScluster
```

- Select an existing Azure Kubernetes cluster that is, BPMSuite\_AKScluster
- Specify the **Service connection name** and **Description**.
- Select the checkbox **Grant access permission to all pipelines** and click **Verify** and **Save**. Once the Service connection is created, it appears in the list.

Figure 3.16

14. If **Kubernetes service connection** is already created, then select the created connection.
15. Select the Namespace, that is, **dev**.
16. Select **Apply** command using the **Command** dropdown.
17. Select the checkbox **Use configuration**.
18. Select the radio button **File path**.
19. Browse the **AzureFile\_PV\_PVC.yml** file path from the **Azure Repos**.
20. Expand the **Advanced** tree structure.
21. Select the **Check for latest version** checkbox.
22. Right click the added kubectl task and select **Clone task(s)**.
23. Change the **Display name** of newly cloned task.
24. Browse the *yml* file (for example, **ibPS5.0ServiceInstanceWeb.yml**) path from the **Azure Repos**.
25. Expand the Secrets tree structure.
26. Select **dockerRegistry** as a **Type of secret**.
27. Select **Azure Container Registry (ACR)** as a **Container registry type**.

28. Select the created **Azure service connection** for ACR.
29. Select the created **Azure container registry**.
30. Specify the **secret name** such as **azurepullsecret**.
31. Select the **Force update secret** checkbox.
32. Right click the cloned kubectl task and Select **Clone task(s)**.
33. You can change the **Display name** of newly cloned task.
34. Browse the *AppGateway-IngressController.yml* file path from the **Azure Repos**.
35. Click **Save** button. Now, as soon as any Docker Image is pushed to the Azure container registry with the tag name **sp2**, Azure DevOps trigger the deployment to the **Dev Stage**.

## 3.12 Configuring the UAT stage

To configure the UAT Stage, perform the below steps:

1. UAT deployments are approval based and they are available on-demand. Once you are ready to deploy to the UAT environment, you just need to trigger the UAT deployment. When you trigger that deployment, an approval mail is sent to the project manager or the concerned team. As soon as the approval is provided for the go-ahead, the UAT deployment starts automatically.
2. Go to the **Pipeline** tab of the Release Pipeline for which **Dev stage** is configured (for example, **iBPS5-ServiceInstance-Web**).
3. Select **Dev stage** and click **Clone icon**. A cloned Stage gets created.
4. Specify the name of the cloned stage as **UAT** in the **Stage** panel.
5. Click **Pre-deployment conditions** icon of the UAT stage. The Pre-deployment conditions panel appears.

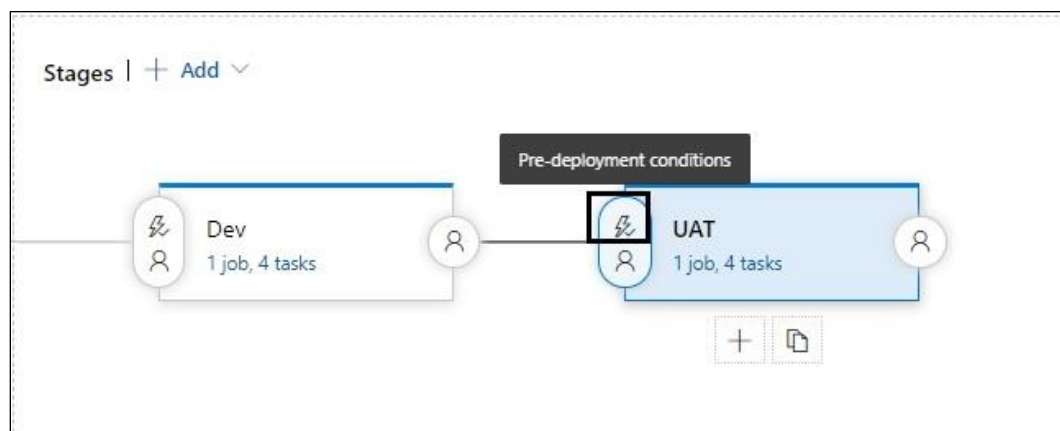


Figure 3.17

6. Select the **Manual Only** under the Triggers section.
7. As soon as the trigger type is changed from After **stage** to **Manual Only**, the UAT stage appears in parallel to Dev Stage instead of a series.

8. In the Pre-deployment conditions panel, enable the **Pre-deployment approvals**.
9. Select the list of users or groups who can approve or reject the deployment to this stage.
10. You can select users or groups by typing their names.
11. Select the **The user requesting a release or deployment should not approve it** checkbox in Approval policies.
12. Click **Close icon** to close the Pre-deployment conditions panel.
13. Click **Save** to save the changes.
14. Click **View stage tasks** link of the UAT stage. Also, make the required changes in the UAT stage's tasks as per your requirements.

For example, you can make the following changes in the below tasks:

- **KubectI Task:** Kubernetes service connection, Kubectl command, changes in YAML files, and so on.

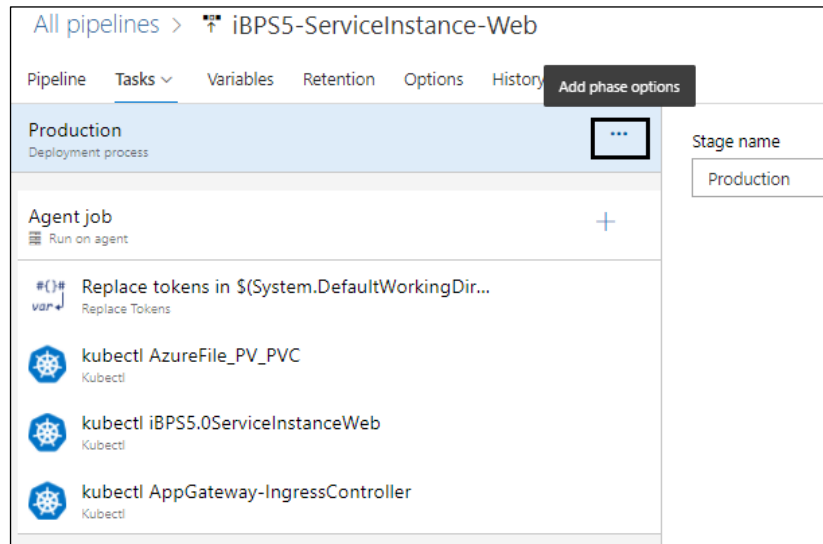
### 3.13 Configuring the production stage

Production deployment is also based on approval, but it is multi-level approval. To deploy a production environment, you require the approval of all stakeholders, and the production environment doesn't get triggered automatically on receiving all the approvals. A manual intervention mail is sent to the engineer who is supposed to deploy to production with a checklist. During deployment, all the checklist points get verified before performing the production deployment. In case any point of the checklist is not covered, then deployment to the production gets rejected.

To configure the Production Stage, perform the below steps:

1. Go to the **Pipeline** tab of the Release Pipeline (for example, **iBPS5-ServiceInstance-Web**) for which **Dev and UAT stages** are just configured.
2. Select the **UAT stage** and click **Clone** icon. A cloned Stage gets created.
3. Specify the name of the cloned stage as **Production** in the **Stage** panel.
4. Click **Pre-deployment conditions** icon of the Production stage. The Pre-deployment conditions dialog appears.
5. Select **Manual Only** option under the Triggers section.
6. As soon the trigger type is changed from **After stage** to **Manual Only**, the Production stage appears in parallel to Dev and UAT stages instead of a series.
7. In the **Pre-deployment conditions** panel, select the list of users or stakeholders whose approval is required for the deployment to the Production stage.
8. Select **Any Order** as an **Approval order**. It indicates that approval of all Stakeholders is required (in any order).
9. Select **The user requesting a release or deployment should not approve it checkbox** in the select policies.

10. Click **Close icon** to close the **Pre-deployment conditions** panel.
11. Click **Save** to save the changes.
12. Click **View stage tasks** link to the **Production** stage.
13. Click **Add phase options** icon in the **Tasks** tab.



**Figure 3.18**

14. Select the **Add an agentless job**.
15. Move **Agentless job** above the **Agent Job** in the **Tasks** tab.
16. Click **Add a task to Agentless job** icon.
17. Add a **Manual intervention** task.
18. Click added task **Manual intervention**.
19. Specify the checklist points that need to execute before deploying to the production stage.

**For example:**

Before deploying to the Production, ensure that the below checklists are completed:

- All Major and Catastrophic bugs must be fixed.
- The latest images must be thoroughly tested on the Dev and UAT stages.
- Approval has taken from all stakeholders.
- Deployment downtime has taken from the client.

20. Select the user or group that are supposed to deploy to the production. A manual intervention mail with the above-mentioned checklist is sent to the engineer who is supposed to deploy to production with a checklist. During deployment, all the checklist points get verified before performing the production deployment. In case any point of the checklist is not covered, then deployment to the production gets rejected.

21. Make the other required changes in the Production stage's tasks as per your requirements.

For example, you can make the following changes in the below tasks:

- **Kubectl Task:** Kubernetes service connection, Kubectl command, changes in YAML files, and so on.

---

**NOTE:**

Refer the above steps to configure the Release Pipeline of other Docker Images.

---

## 4 Creating a new Azure WebApp service

To create a new Azure WebApp service, perform the below steps:

1. Go to Azure portal **Home** page and search for **app**. Click **App Services**.
2. Click **+ Create**.
3. On **Create Web App:Basics** page, fill in the relevant details. For **Publish**, choose option **Code**. For **Runtime Stack**, choose **Node 18 LTS**. For **Operating System**, choose **Windows**. Choose default values for **App Service Plan: Windows Plan** and **Sku and Size**. Click the button **Next:Deployment**.
4. On **Create Web App: Deployment** page, choose default values. Click the button **Next:Networking**.
5. On **Create Web App: Networking** page, choose default values. Click the button **Next:Monitoring**.
6. On **Create Web App: Monitoring** page, choose default values. Click the button **Next:Tags**.
7. On **Create Web App: Tags** page, choose default values or enter relevant tags. Click the button **Next:Review + create**.
8. On **Create Web App: Review + create** page, review the details. Click the button **Create**.
9. Application url can be viewed from the **Overview page**.
10. To add the path mapping information, click the **Configuration tab** inside **Settings** present in Left Pane of created WebApp Service.

### 4.1 Configuring an Azure WebApp service

#### 4.1.1 Application settings

1. To change the version of node server, click on 'Environment Variables' inside **Settings** and add environment variable using '+'. Fill the Name of environment variable in the **Name** input field and node version in the **Value** input field. Click **Apply**.

The screenshot shows a dialog box titled "Add/Edit application setting". It contains the following fields:

- Name \***: WEBSITE\_NODE\_DEFAULT\_VERSIONS
- Value**: ~18.0
- Deployment slot setting**:

At the bottom of the dialog, there are two buttons: "Apply" and "Discard".

Figure 4.1

## 4.1.2 General settings

To change the version of node server, click on 'Configuration' inside **Settings**. Following General settings are to be used for the WebApp Service

- Stack: Node
- Platform: 64 Bit
- Managed Pipeline Version: Integrated
- Basic Auth Publishing Credentials: On
- FTP Basic Auth Publishing Credentials: On
- FTP state: FTPS Only
- HTTP Version: 2.0
- HTTP 2.0 Proxy: Off
- Web sockets: Off
- Always On: On
- Session affinity: On
- HTTPS Only: On
- Minimum Inbound TLS Version: 1.2
- End-to-end TLS encryption: Off
- Remote debugging: Off



### 4.1.3 Path Mappings

To configure the Path Mappings of Azure WebApp service, perform the below steps:

1. Click on **Configuration tab** under the **Settings** section.
2. Now click on **Path mappings** tab to navigate to Virtual applications and directories management section.
3. Click on **New virtual application or directory** button and configure Virtual Path, Physical Path and Type as per the below table.

Virtual Path	Physical Path	Type	Preload Enabled
/	site\wwwroot\webapps\microsite	Application	Yes
/microui	site\wwwroot\webapps\microui	Application	Yes
/newgen-admin	site\wwwroot\webapps\newgen-admin	Application	Yes
/ms-addin	site\wwwroot\webapps\ms-addin	Application	Yes
/content-cloud-admin	site\wwwroot\webapps\content-cloud-admin	Application	Yes
/help	site\wwwroot\help	Directory	No

4. Finally click on the **Save** button to apply the changes.

## 4.2 Creating Build of Web Application

Follow the below mentioned steps to build the web application.

1. Pull the latest code from the Git Repository
2. The directory structure will be as follows:

```
NCC_Web
  Development
    NCC Help Guides
    NCC Micro UI
    NCC Microsite
    NCC Newgen Admin
    NCC Tenant Admin
  Documentation
```

3. Build command is to be run for the application folders present in the Development folder.

## 4.3 Building NCC Microsite

Navigate inside the NCC Microsite folder and perform following steps:

1. Ensure that the following variables are set properly inside the `.env.{environment-name}` file.
  - a. `REACT_APP_WEB_BASE_URL = "{web-app-service-url}"`.  
Example: `"https://ncc-qa.azurewebsites.net "`

- b. `REACT_APP_MICROSERVICE_URL = "{server-url}/ecmapi"`  
Example: `"https://ncc.newgendocker.com/ecmapi"`
2. Run the below mentioned commands one by one:
  - a. `npm install -g dotenv`
  - b. `npm install --legacy-peer-deps`
  - c. `npm run build: {environment name}`  
Value of environment name variable can be:  
development, testing, performance, production as per the environment for which the build is to be created.  
Example: `npm run build:production`
3. Once the build process is successfully completed, a new folder with the name of **build** with deployable files will get created inside **NCC Microsite** folder.

## 4.4 Building NCC Micro UI

Navigate inside the NCC Micro UI folder and perform following steps:

1. Ensure that the following variables are set properly inside the `.env.{environment-name}` file.
  - a. `REACT_APP_WEB_BASE_URL = "{web-app-service-url}/microui"`  
Example: `"https://ncc-qa.azurewebsites.net/microui"`
  - b. `REACT_APP_MICROSERVICE_URL = "{server-url}/ecmapi"`  
Example: `"https://ncc.newgendocker.com/ecmapi"`
  - c. `REACT_APP_UPLOAD_URL = "{zuul-server-url}/zuul/ecmapi"`  
Example: `"https://ncc.newgendocker.com/zuul/ecmapi"`
2. Run the below mentioned commands one by one:
  - a. `npm install -g dotenv`
  - b. `npm install --legacy-peer-deps`
  - c. `npm run build: {environment name}`  
Value of environment name variable can be:  
development, testing, performance, production as per the environment for which the build is to be created.  
Example: `npm run build:production`
3. Once the build process is successfully completed, a new folder with the name of **build** with deployable files will get created inside **NCC Micro UI** folder.

## 4.5 Building NCC Newgen Admin

Navigate inside the NCC Newgen Admin folder and perform following steps:

1. Ensure that the following variables are set properly inside the `.env.{environment-name}` file.
  - a. `REACT_APP_MICROSERVICE_URL = "{server-url}/ecmapi"`  
Example: "https://ncc.newgendocker.com/ecmapi"
2. Run the below mentioned commands one by one:
  - a. `npm install -g dotenv`
  - b. `npm install --legacy-peer-deps`
  - c. `npm run build: {environment name}`  
Value of environment name variable can be:  
development, testing, performance, production as per the environment for which the build is to be created.  
Example: `npm run build:production`
3. Once the build process is successfully completed, a new folder with the name of **build** with deployable files will get created inside **NCC Newgen Admin** folder.

## 4.6 Building NCC Tenant Admin

Navigate inside the NCC Tenant Admin folder and perform following steps:

1. Ensure that the following variables are set properly inside the `.env.{environment-name}` file.
  - a. `NCC_WEB_BASE_URL = "{web-app-service-url}"`.  
Example: "https://ncc-qa.azurewebsites.net "
  - b. `NCC_MICROSERVICE_URL = "{server-url}/ecmapi"`  
Example: "https://ncc.newgendocker.com/ecmapi"
2. Run the below mentioned commands one by one:
  - a. `npm install --legacy-peer-deps`
  - b. `ng build --configuration {environment name}`  
Value of environment name variable can be:  
development, testing, performance, production as per the environment for which the build is to be created.  
Example: `ng build --configuration production`
3. Once the build process is successfully completed, a new folder with the name of **dist** with deployable files will get created inside **NCC Tenant Admin** folder.

## 4.7 Deploying Web Application to App Service

Follow the below mentioned steps to setup FTP site for deployment:

1. To get the FTPS credentials, click on Deployment Center tab under the Deployment section of Microsoft Azure Portal.

2. Now click on FTPS credentials tab to access –
  - a. FTPS endpoint
  - b. FTPS Username
  - c. Password
3. Configure FTP site inside an FTP client and use the above details to establish a connection.
4. Once the connection is established successfully, following directory structure will be visible:  
**/site/wwwroot/**
5. Below mentioned 2 folders must be present inside **/site/wwwroot/** directory.
  - a. webapps
  - b. help
6. In case webapps and help folders are not present, these folders need to be created.
7. Now navigate inside the **webapps** folder and create the following folders inside it if they are not present already.
  - a. newgen-admin
  - b. microui
  - c. microsite
  - d. content-cloud-admin

## 4.8 Deploying NCC Microsite

Perform the following steps to deploy NCC Microsite Web App:

1. Navigate to the path **/NCC\_Web/Development/NCC Microsite/build** and copy all the files present inside the **build** folder.
2. Now open FTP client and establish connection with the NCC FTP site. Once the connection is successful, navigate to **/site/wwwroot/webapps/microsite**
3. Paste all the files copied from the **build** folder to the **microsite** folder and wait for the transfer to complete.
4. Once the file transfer is completed, the web application will be deployed successfully.

## 4.9 Deploying NCC Micro UI

Perform the following steps to deploy NCC Micro UI Web App:

1. Navigate to the path **/NCC\_Web/Development/NCC Micro UI/build** and copy all the files present inside the **build** folder.
2. Now open FTP client and establish connection with the NCC FTP site. Once the connection is successful, navigate to **/site/wwwroot/webapps/microui**
3. Paste all the files copied from the **build** folder to the **microui** folder and wait for the transfer to complete.
4. Once the file transfer is completed, the web application will be deployed successfully.

## 4.10 Deploying NCC Newgen Admin

Perform the following steps to deploy NCC Newgen Admin Web App:

1. Navigate to the path **/NCC\_Web/Development/NCC Newgen Admin/build** and copy all the files present inside the **build** folder.
2. Now open FTP client and establish connection with the NCC FTP site. Once the connection is successful, navigate to **/site/wwwroot/webapps/newgen-admin**
3. Paste all the files copied from the **build** folder to the **newgen-admin** folder and wait for the transfer to complete.
4. Once the file transfer is completed, the web application will be deployed successfully.

## 4.11 Deploying NCC Tenant Admin

Perform the following steps to deploy NCC Tenant Admin Web App:

1. Navigate to the path **/NCC\_Web/Development/NCC Tenant Admin/dist/ncc-tenant-admin** and copy all the files present inside the **ncc-tenant-admin** folder.
2. Now open FTP client and establish connection with the NCC FTP site. Once the connection is successful, navigate to **/site/wwwroot/webapps/content-cloud-admin**
3. Paste all the files copied from the **ncc-tenant-admin** folder to the **content-cloud-admin** folder and wait for the transfer to complete.
4. Once the file transfer is completed, the web application will be deployed successfully.

## 4.12 Deploying NCC Help Guides

Perform the following steps to deploy NCC Help Guides:

1. Navigate to the path **/NCC\_Web/Development/NCC Help Guides** and copy the **help** folder.
2. Now open FTP client and establish connection with the NCC FTP site. Once the connection is successful, navigate to **/site/wwwroot/help**
3. Replace the **help** folder at the location **/site/wwwroot/help** with the help folder which you had copied earlier and wait for the transfer to complete.
4. Once the file transfer is completed, NCC Help Guides will be deployed successfully.

## 4.13 Verifying Deployed Web Applications

To verify, check whether below mentioned URLs are working fine or not:

Assumption: **Web\_App\_Service\_Base\_URL** is the url of the web app service which was created.

Web Application	Web Application URL / Path
NCC Microsite	{Web_App_Service_Base_URL}/
NCC Micro UI	{Web_App_Service_Base_URL}/microui/
NCC Newgen Admin	{Web_App_Service_Base_URL}/newgen-admin/
NCC Tenant Admin	{Web_App_Service_Base_URL}/content-cloud-admin/
NCC Help Guides	{Web_App_Service_Base_URL}/help/content-cloud-admin-guide/index.htm