**Advith**

**Azure Pre-Prod environment configuration Guide**

**Table of Contents Page**

**PART – 1: CREATING RESOURCES IN AZURE**

1. **Architecture diagram of Pre-Prod environment 4**
2. **Create Resource group 4**
3. [**Create Azure MySQL server and migrate databases 4**](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.3dy6vkm)
   1. [Create MySQL server for pre-prod environment 4](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.30j0zll)
   2. [Migrate Databases from staging to pre-prod MySQL server](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.3znysh7) 5
   3. [Errors while importing DB’s 7](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.z337ya)
4. [**Create Container in the Blob storage**](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.4d34og8) **8**
5. [**Create AKS cluster and Azure container registry**](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.2s8eyo1) **8**
   1. [Steps to create AKS cluster and ACR](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.44sinio) 8
   2. [Accessing AKS cluster from CentOS 9](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.z337ya)
   3. [Accessing ACR](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.44sinio) 10

**PART – 2: CONTAINERIZATION**

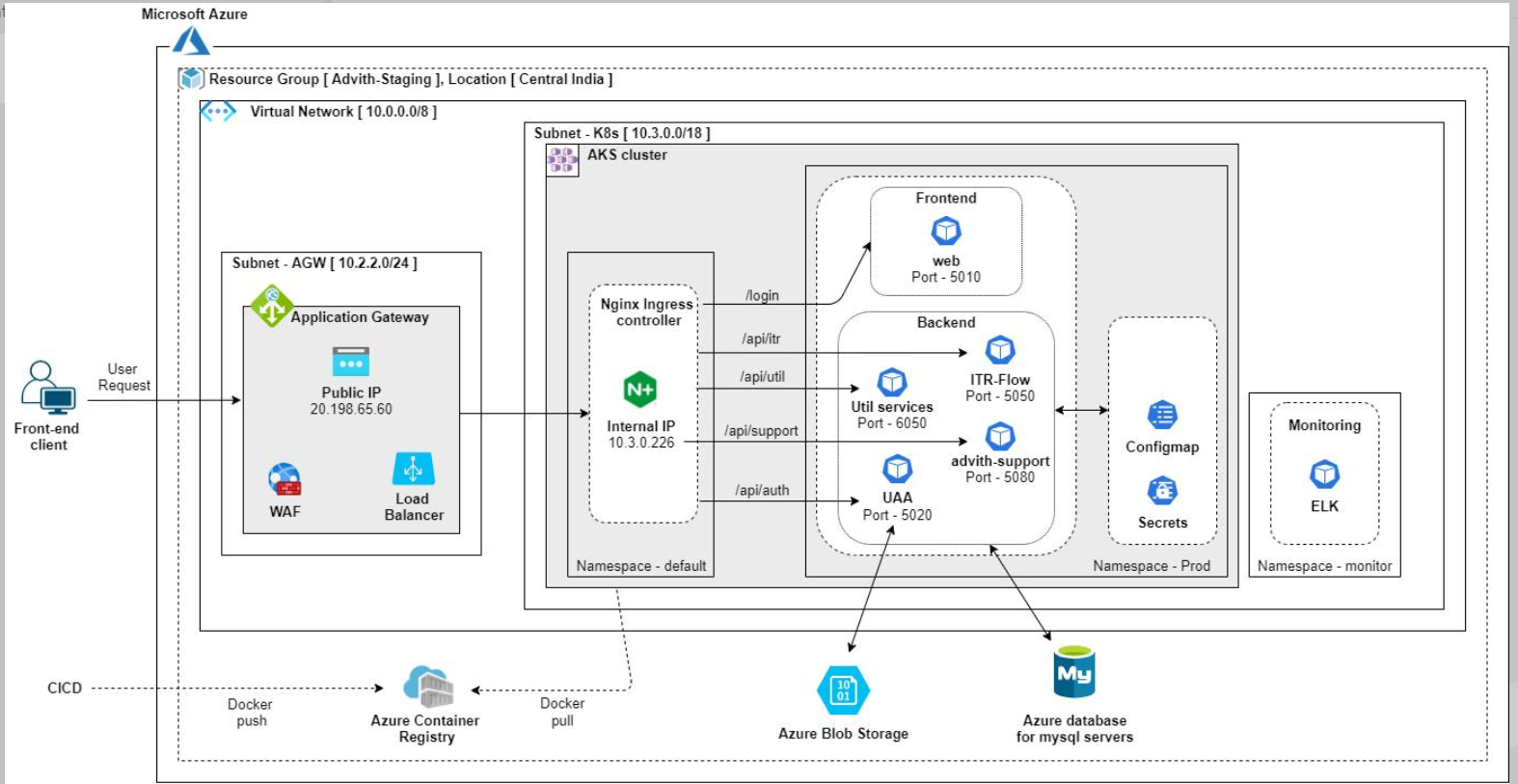
1. [**Dockerfiles for Backend services 1**](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.3dy6vkm)**1**
2. [**Kubernetes YAML’s for backend services 1**](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.4d34og8)**2**
   1. [Configmaps 1](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.30j0zll)2
   2. [Secrets](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.3znysh7) 15
   3. [Service account, roles and role binding 1](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.z337ya)5
   4. [Persistent Volumes (PV)](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.3znysh7) 16
   5. [Persistent Volume Claim (PVC)](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.3znysh7) 17
   6. [Deployments](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.z337ya) 18
   7. [Services](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.3znysh7) 23
3. [**Dockerfile for frontend 2**](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.2s8eyo1)**4**
4. [**Kubernetes YAML’s for frontend 2**](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.2s8eyo1)**5**
   1. [Deployment 2](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.44sinio)5
   2. [Service 2](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.z337ya)5
5. [**Configure ingress controller**](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.17dp8vu) **26**
   1. [Install nginx ingress controller with its external IP disabled 2](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.44sinio)6
   2. [Configure ingress-rules.yaml](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.z337ya) 28
6. [**Azure application gateway**](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.17dp8vu) **29**
7. [**CI/CD Architecture**](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.17dp8vu) **34**
8. [**Jenkins Job configuration**](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.17dp8vu) **34**
   1. Jenkins job for frontend service [3](https://docs.google.com/document/d/1AqpQMDijKX5vzDwP9hA-iQ1DbTNS2RO3/edit#heading=h.44sinio)5
   2. Jenkins job for backend service 36

13.2.1 UAA 36

* 1. Error while pushing/pulling docker image from harbor repo 36

PART -1: Creating resources in Azure

1. ARCHITECTURE DIAGRAM OF PRE-PROD ENVIRONMENT



1. CREATE RESOURCE GROUP

For Pre-Prod we are using the existing resource group **“Advith-Staging”** in Azure that holds related resources for pre-prod environment.

Region chosen for creating all resources: Central India

1. CREATE AZURE MYSQL SERVER AND MIGRATE DATABASES

**3.1. Create MySQL server for pre-prod environment with the name advith-preprod-mysql-server with below specifications:**

Deployment - Single Server Deployment

Data Source - Null

MySQL version - 8.0

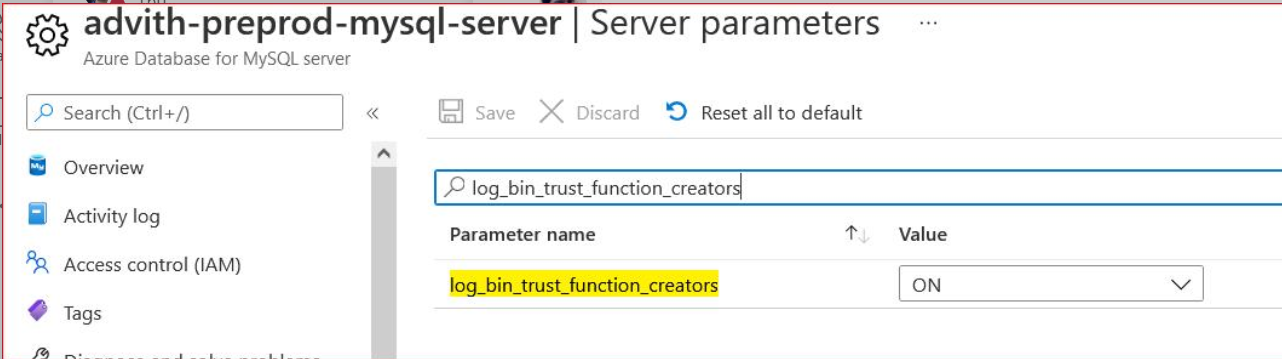
Performance configuration - Basic, 1 vCore(s), 5 GB

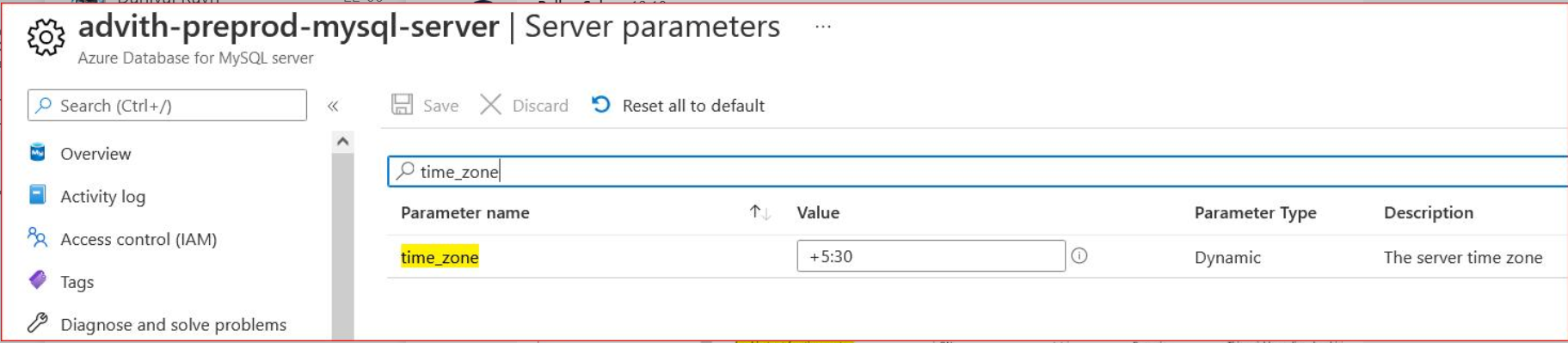
Backup Redundancy - LRS redundancy (Locally redundant)

SSL - Disabled

Once MySQL server is created set the following server parameters

Navigate to MySQL server in Azure portal -> server parameters -> make the “log\_bin\_trust\_function\_creators” parameters “ON” and “time\_zone” parameter to “+5:30” ie., IST as shown below



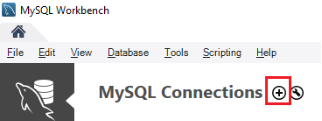


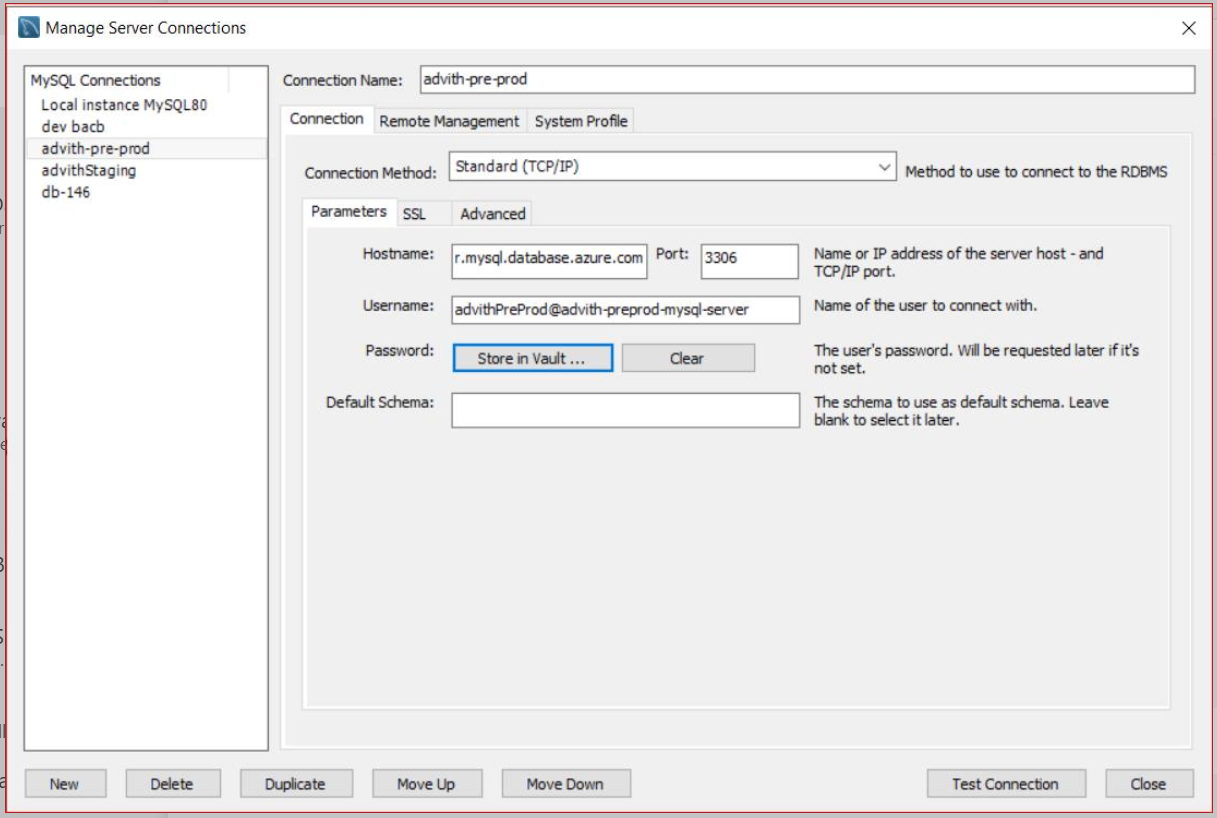
**3.2 Migrate Databases from staging to pre-prod MySQL server:**

Prerequisites: MySQL workbench

Step-1: Connect to Azure MySQL server from workbench.

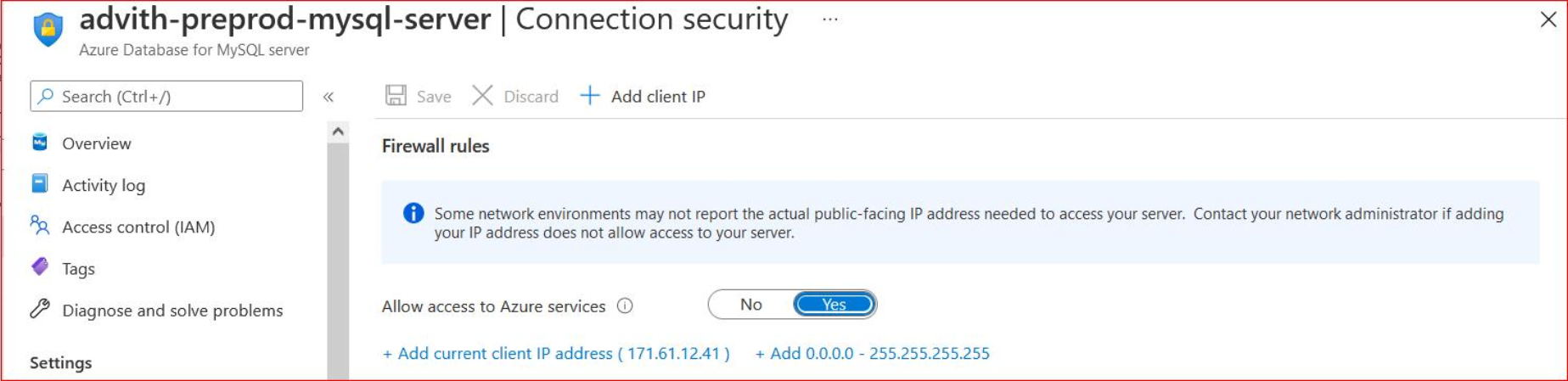
Open MySQL workbench -> Add a new MySQL connection -> Fill in Azure MySQL server connection details





Click on test connection -> If everything seems fine then you will get a successful connection pop up.

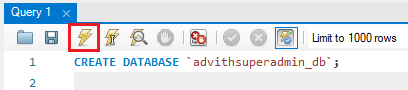
If not, in Azure portal, verify if your system has appropriate permissions in order to connect to MySQL -> Navigate to Azure MySQL server -> Connection security -> Add a firewall rule -> Specify your system public IP or IP range



Step -2:

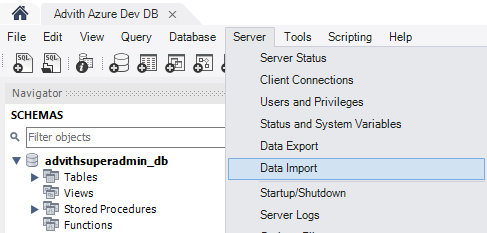
Get sqldump file from developers for required Databases that are in Azure staging MySQL “server advith-staging-mysql-server.mysql.database.azure.com”

Step-3: Create a DB’s in Azure with same name as in Advith pre-prod Mysql server

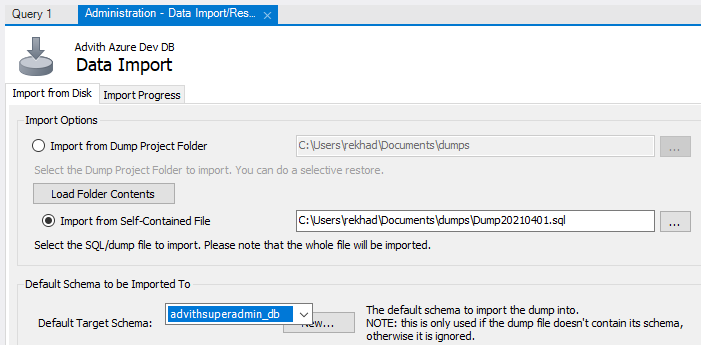


Step-4: Import advith-staging-mysql-server DB’s to pre-prod MySQL server:

Once connected to Azure MySQL server from MySQL workbench -> Navigate to server option -> Click on Data import option



Click on “import from self-contained file” -> Choose the location of sql dump file -> In “default target schema”, select Database that you will be importing via .sql file and click on start import



Likewise, import all DB’s from Azure staging to Azure pre-prod MySQL server

* 1. **Errors while importing DB’s:**

**Error-1:**

ERROR 1227 (42000) at line 2007: Access denied; you need (at least one of) the SUPER or SET\_USER\_ID privilege(s) for this operation

mysqldump: Got errno 22 on write

The above error may occur while importing a dump file or creating procedure that contains definers

Resolution-1:

To resolve this error, the admin user can grant privileges to create or execute procedures by running GRANT command as shown below:

GRANT CREATE ROUTINE ON mydb.\* TO 'someuser'@'somehost';

GRANT EXECUTE ON PROCEDURE mydb.myproc TO 'someuser'@'somehost';

Resolution-2:

Alternatively, you can replace the definers with the name of the admin user that is running the import process as shown below

DELIMITER;; /\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`127.0.0.1`\*/ /\*!50003

DELIMITER;;

/\* Modified to \*/

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`AdminUserName`@`ServerName`\*/ /\*!50003

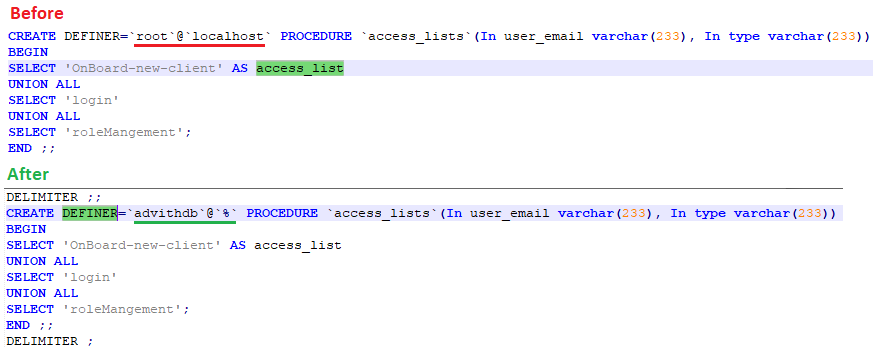
DELIMITER ;

For this, Open the sqldump file and search for “definer”

Previously it is DEFINER=`root`@`localhost`

Replace it like DEFINER=`azure\_sqlserver\_username`@`%`

Modify the username and hostname for all definer lines as shown below:



1. CREATE CONTAINER IN THE BLOB STORAGE

**In the existing storage account “advithstagingstorage” Create container**

**“advith-preprod-container”**

1. CREATE AKS CLUSTER AND AZURE CONTAINER REGISTRY
   1. **Steps to create AKS cluster and ACR:**

Cluster Name - Advith-Production-AKS

Kubernetes Version - 1.20.5

Node size - Standard\_B2ms (2 cpu’s, 8 GB memory)

Node count - 2

Enable VM scale sets - yes

Network configuration - Azure CNI (Create a VNet, subnet for nodes and provide address range for K8s services, K8s DNS service IP, Docker bridge address and finally DNS name prefix)

Network policy - Azure

Container Registry - Create a container registry with admin user enabled and SKU - standard

* 1. **Accessing AKS cluster from CentOS:**

1. Install azure-cli
2. Install kubectl
3. Install docker
4. Login to Azure, get AKS credential file and connect

* **Install azure-cli**

sudo rpm --import <https://packages.microsoft.com/keys/microsoft.asc>

sudo sh -c 'echo -e "[azure-cli]

name=Azure CLI

baseurl=https://packages.microsoft.com/yumrepos/azure-cli

enabled=1

gpgcheck=1

gpgkey=https://packages.microsoft.com/keys/microsoft.asc" > /etc/yum.repos.d/azure-cli.repo'

yum install azure-cli

* **Install kubectl**

cat <<EOF > /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86\_64

enabled=1

gpgcheck=1

repo\_gpgcheck=1

gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

EOF

yum install -y kubectl

* **Install Docker**

yum install -y yum-utils device-mapper-persistent-data lvm2

yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

yum install -y docker-ce-19.03.12

sudo systemctl start docker

sudo systemctl enable docker

sudo systemctl status docker

* **Login to Azure, get AKS credential file and connect**
* Login to Azure account using below command

**az login**

You will get a message like

“To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code FEDSJEHBR to authenticate”

Now choose your Microsoft Azure account and you will be logged in successfully.

* If you have multiple subscriptions in Azure account then choose appropriate subscription.

**az account set --subscription YOUR\_SUBSCRIPTION**

* Connect to AKS cluster

**az aks get-credentials --resource-group AKS\_RG\_NAME --name AKS\_CLUSTER\_NAME**

Once you have connected to AKS cluster, Kubernetes will store configuration in ~/.kube/config file.

* Verify that you can access AKS by running kubectl commands

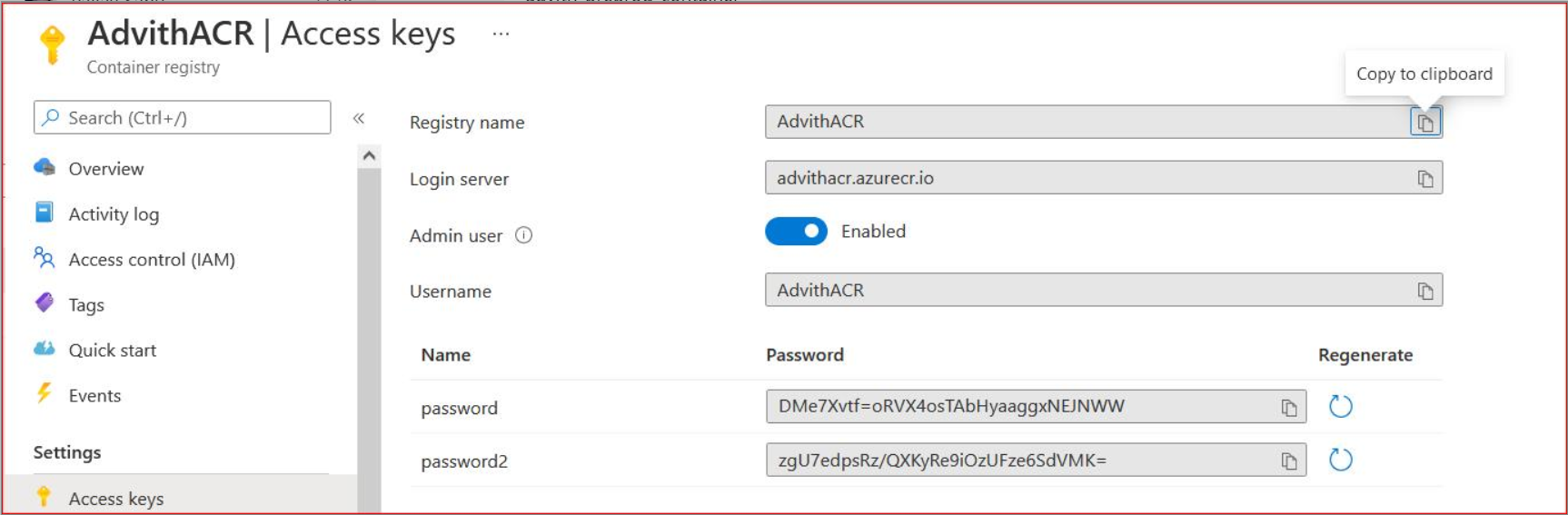
**kubectl cluster-info**

* 1. **Accessing ACR:**
* Command to login to ACR

**docker login <login\_server\_name>**

It will ask for credentials

* Navigate to your container registry -> Under “access keys” you can find the ACR credentials



PART -2: Containerization

1. DOCKERFILES FOR BACKEND SERVICES

To run the micro-services in containers with Docker, first clone the repository from Gitlab and build it (mvn clean install -DskipTests). Now write a Dockerfile to copy the resultant war file into container and run it using java –jar command. Place the Dockerfile under root directory of your project.

**UAA:**

FROM openjdk:8-jdk-alpine

RUN apk update

RUN apk add tzdata

ENV TZ Asia/Kolkata

ADD target/advithmultitenant-0.0.1-SNAPSHOT.war /home/uaa.war

EXPOSE 5020

ENTRYPOINT [ "sh", "-c", "java -jar /home/uaa.war" ]

**Util-services:**

FROM openjdk:8-jdk-alpine

RUN apk update

RUN apk add tzdata

ENV TZ Asia/Kolkata

ADD target/commonServices.war /home/util-services.war

EXPOSE 6050

ENTRYPOINT [ "sh", "-c", "java -jar /home/util-services.war" ]

**ITR-Flow:**

FROM openjdk:8-jdk-alpine

RUN apk update

RUN apk add tzdata

ENV TZ Asia/Kolkata

ADD target/advith-itr-service-0.0.1-SNAPSHOT.war /home/itr-flow.war

EXPOSE 5050

ENTRYPOINT [ "sh", "-c", "java -jar /home/itr-flow.war" ]

**Advith-support project:**

FROM openjdk:8-jdk-alpine

RUN apk update

RUN apk add tzdata

ENV TZ Asia/Kolkata

ADD target/advith-support-0.0.1-SNAPSHOT.war /home/advith-support.war

EXPOSE 5080

ENTRYPOINT [ "sh", "-c", "java -jar /home/advith-support.war" ]

Note:

In the above Dockerfile timezone is set to “Asia/Kolkata”. This is because the timezone of container in which the application running is UTC where as the timezone of MySQL database is IST. Due to this time difference, after the application starts running, it is failing to store data in the database.

1. KUBERNETES YAMLS FOR BACKEND SERVICES

Below are the K8s resources being used for Advith:

1. Configmap
2. Secrets
3. Service account, roles and role binding
4. Persistent Volume (PV)
5. Persistent Volume Claim (PVC)
6. Deployment
7. Service
   1. **Configmap:**

Create a configmap to externalize the properties present in bootstrap.properties and application yaml file.

How to add key values in configmap for spring boot application?

For example, to store DB URL in configmap, instead of simply specifying url, you should specify fully qualified path to key like “multitenancy.mtapp.master.datasource.url” and provide its value separated by a colon. Likewise specify all key-value pairs under data section inside configmap.

**Util-configmap.yaml**

apiVersion: v1

kind: ConfigMap

metadata:

name: util-configmap

data:

dev-config.json:

'{

"spring.profiles.active": "preprod",

"multitenancy.mtapp.master.datasource.url": "jdbc:mysql:// advith-preprod-mysql-server.mysql.database.azure.com:3306/advithsuperadmin\_db?allowPublicKeyRetrieval=true&useLegacyDatetimeCode=false&serverTimezone=IST",

"multitenancy.mtapp.master.datasource.username": "advithPreProd@advith-preprod-mysql-server",

"twilio.account\_sid": "AC0b250f5af69bdb8d9ce89b5ed9a8861c",

"twilio.auth\_token": "9413fa2409ab216344ad6658dd147e65",

"twilio.trial\_number": "+19096554679",

"server.port": "6050",

"sendgrid.api-key": "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",

"sendgrid.from": "support@advithitec.in",

"sendgrid.name": "Advith Support Team",

"sendgrid.username": "username",

"sendgrid.password": "password"

}'

**Uaa-configmap.yaml**

apiVersion: v1

kind: ConfigMap

metadata:

name: uaa-configmap

data:

dev-config.json:

'{

"spring.profiles.active": "preprod",

"multitenancy.mtapp.master.datasource.url": "jdbc:mysql://advith-preprod-mysql server.mysql.database.azure.com:3306/advithsuperadmin\_db?allowPublicKeyRetrieval=true&useLegacyDatetimeCode=false&serverTimezone=IST",

"multitenancy.mtapp.master.datasource.username": "advithPreProd@advith-preprod-mysql-server",

"twilio.account\_sid": "AC0b250f5af69bdb8d9ce89b5ed9a8861c",

"twilio.auth\_token": "9413fa2409ab216344ad6658dd147e65",

"twilio.trial\_number": "+19096554679",

"server.port": "5020",

"blob.connection-string": "DefaultEndpointsProtocol=https;AccountName=advithstagingstorage;AccountKey=LcfxFRl0BRZRnFHk5196m95js4ljWkN9k8rRtGCrolp3xO624F48jPqzqledWI9074b/yAgcpO1JXZ9NQwaToQ==;EndpointSuffix=core.windows.net",

"blob.container-name": "advith-staging-container",

"sendgrid.api-key": "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",

"sendgrid.from": "support@advithitec.in",

"sendgrid.name": "Advith Support Team",

"sendgrid.username": "username",

"sendgrid.password": "password",

"send-save-notificaiton-url": "http://advithutilservices:6050/api/util",

"send-save-notificaiton-url-CRC": "http://advithutilservices:6050/api/util/saveCRCBellNotification",

"send-save-notificaiton-url-WA": "http://advithutilservices:6050/api/util/sendwhatsAppNotification",

"create-sub-servie-url": "http://advithutilservices:6050/api/audit/create-all-sub-services"

}'

**Itr-configmap.yaml**

apiVersion: v1

kind: ConfigMap

metadata:

name: itr-configmap

data:

dev-config.json:

'{

"spring.profiles.active": "preprod",

"multitenancy.mtapp.master.datasource.url": "jdbc:mysql://advith-preprod-mysqlserver.mysql.database.azure.com:3306/advithsuperadmin\_db?allowPublicKeyRetrieval=true&useLegacyDatetimeCode=false&serverTimezone=IST",

"multitenancy.mtapp.master.datasource.username": "advithPreProd@advith-preprod-mysql-server",

"send-save-notificaiton-url": "http://advithutilservices:6050/api/util",

"send-save-notificaiton-url-CRC": "http://advithutilservices:6050/api/util/saveCRCBellNotification",

"send-save-notificaiton-url-WA": "http://advithutilservices:6050/api/util/sendwhatsAppNotification"

}'

**Advith-support-configmap.yaml**

apiVersion: v1

kind: ConfigMap

metadata:

name: advith-support-configmap

data:

dev-config.json:

'{

"spring.profiles.active": "preprod",

"multitenancy.mtapp.master.datasource.url":

"jdbc:mysql://advith-preprod-mysqlserver.mysql.database.azure.com:3306/advithsuperadmin\_db?allowPublicKeyRetrieval=true&useLegacyDatetimeCode=false&serverTimezone=IST",

"multitenancy.mtapp.master.datasource.username":

"advithPreProd@advith-preprod-mysql-server",

"server.port": "5080",

"send-save-notificaiton-url": "https://api.advithitec.in/api/util/commonServices/api"

}'

* 1. **Secrets:**

In secrets, we are storing DB password in encoded format. As DB password remains same for all backend services, we create a common secret

Encode the Mysql server password using “ echo -n "<>" | base64 ”, we will get an encoded string which will be used inside secrets.yaml.

**Secrets.yaml**

apiVersion: v1

kind: Secret

metadata:

name: db-secret

type: Opaque

data:

multitenancy.mtapp.master.datasource.password: TWF2M3JpY0Ax

* 1. **Service account, roles and role binding:**

In Kubernetes, service accounts are used to provide an identity for pods. These are used to provide appropriate permissions for pods to communicate with other K8s resources like configmaps, services, endpoints, etc.

**Service-account.yaml**

# create the service account

apiVersion: v1

kind: ServiceAccount

metadata:

name: api-reader

---

# create the role to grant access to configmaps

apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

name: role-api-reader

rules:

- apiGroups: [""] # "" indicates the core API group

resources: ["pods","configmaps","endpoints","services"]

verbs: ["get", "watch", "list"]

---

# bind the role and the service account

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

name: rolebinding-api-reader

subjects: #specify the user details

- kind: ServiceAccount

name: api-reader # Name is case sensitive

roleRef: #specify the details of role which we created

kind: Role #this must be Role or ClusterRole

name: role-api-reader # this must match the name of the Role or ClusterRole you wish to bind to

apiGroup: rbac.authorization.k8s.io

* 1. **Persistent Volume (PV):**

Here we are using PV and PVC’s to persist pod logs.

**Uaa-pv.yaml**

apiVersion: v1

kind: PersistentVolume

metadata:

name: uaa-pre-prod-pv

labels:

type: uaa-pv

spec:

storageClassName: manual

capacity:

storage: 1Gi

accessModes:

- ReadWriteMany

hostPath:

path: "/mnt/data/uaa"

**Util-services-pv.yaml**

apiVersion: v1

kind: PersistentVolume

metadata:

name: util-services-pre-prod-pv

labels:

type: util-services-pv

spec:

storageClassName: manual

capacity:

storage: 1Gi

accessModes:

- ReadWriteMany

hostPath:

path: "/mnt/data/util-services"

**Itr-flow-pv.yaml**

apiVersion: v1

kind: PersistentVolume

metadata:

name: itr-flow-pre-prod-pv

labels:

type: itr-flow-pv

spec:

storageClassName: manual

capacity:

storage: 1Gi

accessModes:

- ReadWriteMany

hostPath:

path: "/mnt/data/itr-flow"

**Advith-support-pv.yaml**

apiVersion: v1

kind: PersistentVolume

metadata:

name: advith-support-pre-prod-pv

labels:

type: advith-support-pv

spec:

storageClassName: manual

capacity:

storage: 1Gi

accessModes:

- ReadWriteMany

hostPath:

path: "/mnt/data/advith-support"

* 1. **Persistent Volume Claim (PVC):**

**Uaa-pvc.yaml**

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: uaa-pv-claim

spec:

selector:

matchLabels:

type: uaa-pv

storageClassName: manual

accessModes:

- ReadWriteMany

resources:

requests:

storage: 1Gi

**Util-services-pvc.yaml**

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: util-services-pv-claim

spec:

selector:

matchLabels:

type: util-services-pv

storageClassName: manual

accessModes:

- ReadWriteMany

resources:

requests:

storage: 1Gi

**Itr-flow-pvc.yaml**

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: itr-flow-pv-claim

spec:

selector:

matchLabels:

type: itr-flow-pv

storageClassName: manual

accessModes:

- ReadWriteMany

resources:

requests:

storage: 1Gi

**Advith-support-pvc.yaml**

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: advith-support-pv-claim

spec:

selector:

matchLabels:

type: advith-support-pv

storageClassName: manual

accessModes:

- ReadWriteMany

resources:

requests:

storage: 1Gi

* 1. **Deployments:**

As we will push images to Azure container registry (ACR), properly tag the image according to the login server name of your ACR instance.

How to tag an image to push to ACR?

Below is the image tagging format to publish an image to ACR

<acr\_login\_name>.azurecr.io/<image\_name>:<tag>

Configure imagePullSecrets:

We cannot pull an image from ACR without authenticating against ACR. So create a secret that holds ACR credentials and mention that secret name under “imagePullSecrets” in the deployment yaml file.

Ex of creating secret for ACR credentials:

kubectl create secret -n <namespace> docker-registry <secret-name> \

--docker-server=<acr-login-server-name> \

--docker-username=<acr-username> \

--docker-password=<acr-password>

**Uaa-deployment.yaml**

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: uaa

spec:

selector:

matchLabels:

app: uaa

replicas: 2

revisionHistoryLimit: 10

template:

metadata:

labels:

app: uaa

spec:

serviceAccountName: api-reader

imagePullSecrets:

- name: acr-secret

containers:

- name: uaa

image: advithacr.azurecr.io/<reponame>/<image>:<tag>

imagePullPolicy: Always

ports:

- containerPort: 5020

env:

- name: SPRING\_APPLICATION\_JSON

valueFrom:

configMapKeyRef:

name: uaa-configmap

key: dev-config.json

- name: DB\_PASSWORD

valueFrom:

secretKeyRef:

name: db-secret

key: multitenancy.mtapp.master.datasource.password

volumeMounts:

- mountPath: /home/advith/logs

name: uaa-volume

volumes:

- name: uaa-volume

persistentVolumeClaim:

claimName: uaa-pv-claim

restartPolicy: Always

**Util-deployment.yaml**

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: util-services

spec:

selector:

matchLabels:

app: util-services

replicas: 2

revisionHistoryLimit: 10

template:

metadata:

labels:

app: util-services

spec:

serviceAccountName: api-reader

imagePullSecrets:

- name: acr-secret

containers:

- name: util-services

image: advithacr.azurecr.io/<reponame>/<image>:<tag>

imagePullPolicy: Always

ports:

- containerPort: 6050

env:

- name: SPRING\_APPLICATION\_JSON

valueFrom:

configMapKeyRef:

name: util-configmap

key: dev-config.json

- name: DB\_PASSWORD

valueFrom:

secretKeyRef:

name: db-secret

key: multitenancy.mtapp.master.datasource.password

volumeMounts:

- mountPath: /home/advith/logs

name: util-volume

volumes:

- name: util-volume

persistentVolumeClaim:

claimName: util-services-pv-claim

restartPolicy: Always

**Itr-deployment.yaml**

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: itr-flow

spec:

selector:

matchLabels:

app: itr-flow

replicas: 2

revisionHistoryLimit: 10

template:

metadata:

labels:

app: itr-flow

spec:

serviceAccountName: api-reader

imagePullSecrets:

- name: acr-secret

containers:

- name: itr-flow

image: advithacr.azurecr.io/<reponame>/<image>:<tag>

imagePullPolicy: Always

ports:

- containerPort: 5050

env:

- name: SPRING\_APPLICATION\_JSON

valueFrom:

configMapKeyRef:

name: itr-configmap

key: dev-config.json

- name: DB\_PASSWORD

valueFrom:

secretKeyRef:

name: db-secret

key: multitenancy.mtapp.master.datasource.password

volumeMounts:

- mountPath: /home/advith/logs

name: itr-volume

volumes:

- name: itr-volume

persistentVolumeClaim:

claimName: itr-flow-pv-claim

restartPolicy: Always

**Advith-support-deployment.yaml**

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: advith-support

spec:

selector:

matchLabels:

app: advith-support

replicas: 1

revisionHistoryLimit: 10

template:

metadata:

labels:

app: advith-support

spec:

serviceAccountName: api-reader

imagePullSecrets:

- name: acr-secret

containers:

- name: advith-support

image: advithacr.azurecr.io/<reponame>/<image>:<tag>

imagePullPolicy: Always

ports:

- containerPort: 5080

env:

- name: SPRING\_APPLICATION\_JSON

valueFrom:

configMapKeyRef:

name: advith-support-configmap

key: dev-config.json

- name: DB\_PASSWORD

valueFrom:

secretKeyRef:

name: db-secret

key: multitenancy.mtapp.master.datasource.password

volumeMounts:

- mountPath: /home/advith/logs

name: advith-support-volume

volumes:

- name: advith-support-volume

persistentVolumeClaim:

claimName: advith-support-pv-claim

restartPolicy: Always

* 1. **Services:**

**Uaa-service.yaml**

kind: Service

apiVersion: v1

metadata:

name: advithuaaservice

spec:

selector:

app: uaa

type: ClusterIP

ports:

- protocol: TCP

port: 5020

targetPort: 5020

**Util-service.yaml**

---

kind: Service

apiVersion: v1

metadata:

name: advithutilservices

spec:

selector:

app: util-services

type: ClusterIP

ports:

- protocol: TCP

port: 6050

targetPort: 6050

**Itr-service.yaml**

---

kind: Service

apiVersion: v1

metadata:

name: advithitrservice

spec:

selector:

app: itr-flow

type: ClusterIP

ports:

- protocol: TCP

port: 5050

targetPort: 5050

**Advith-support-service.yaml**

---

kind: Service

apiVersion: v1

metadata:

name: advithsupportservice

spec:

selector:

app: advith-support

type: ClusterIP

ports:

- protocol: TCP

port: 5080

targetPort: 5080

1. DOCKERFILE FOR FRONTEND

As the frontend is ReactJS application, it requires nginx in order to start the application. Here we are using nginx as base image, build the npm package on the on-prem server and copy the resultant build to docker image. Also, we will copy .conf file to /etc/nginx/conf.d folder.

**Dockerfile**

FROM nginx:1.18.0-alpine

COPY ./build /usr/share/nginx/web-advithfin/

COPY ./advith.conf /etc/nginx/conf.d/advith.conf

EXPOSE 5010

**Advith.conf**

server {

listen 5010;

root /usr/share/nginx/web-advithfin;

index index.html;

server\_name localhost;

location /advith-uat/login {

proxy\_pass http://localhost:5010/;

}

location / {

try\_files $uri /index.html =404;

}

}

1. KUBERNETES YAML FILES FOR FRONTEND
   1. **Deployment:**

**Web-advithfin-deployment.yaml**

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: web-advithfin

spec:

selector:

matchLabels:

app: web

replicas: 1

revisionHistoryLimit: 10

template:

metadata:

labels:

app: web

spec:

serviceAccountName: api-reader

imagePullSecrets:

- name: acr-secret

containers:

- name: web

image: advithacr.azurecr.io/<reponame>/<image>:<tag>

imagePullPolicy: Always

ports:

- containerPort: 5010

restartPolicy: Always

* 1. **Service:**

**Web-advithfin-service.yaml**

---

kind: Service

apiVersion: v1

metadata:

name: web-service

spec:

selector:

app: web

type: ClusterIP

ports:

- protocol: web

port: 5010

targetPort: 5010

- protocol: http

port: 80

targetPort: 80

1. CONFIGURE INGRESS CONTROLLER

Install NGINX ingress controller with its external IP disabled.

* 1. **Install nginx ingress controller with its external IP disabled:**

**Prerequisites:**

* Install helm

wget https://get.helm.sh/helm-v3.2.4-linux-amd64.tar.gz

tar -xvf helm-v3.2.4-linux-amd64.tar.gz

sudo mv linux-amd64/helm /usr/bin/

sudo chmod +x /usr/bin/helm

helm version

**Installation process:**

* Choose a private ip from your AKS subnet and ensure it is not being used using the below command

**az network vnet check-ip-address --name <AKS-VNet-name> -g <resource-group-name> --ip-address <ip-address>**

Ex:

az network vnet check-ip-address --name Advith-Staging-AKS-VNet -g Advith-Staging --ip-address 10.3.0.226

* Delegate appropriate roles (Reader and Contributor) for the AKS cluster to provision the private ip from subnet

**az role assignment create --assignee <appId> --scope <resourceScope> --role Contributor**

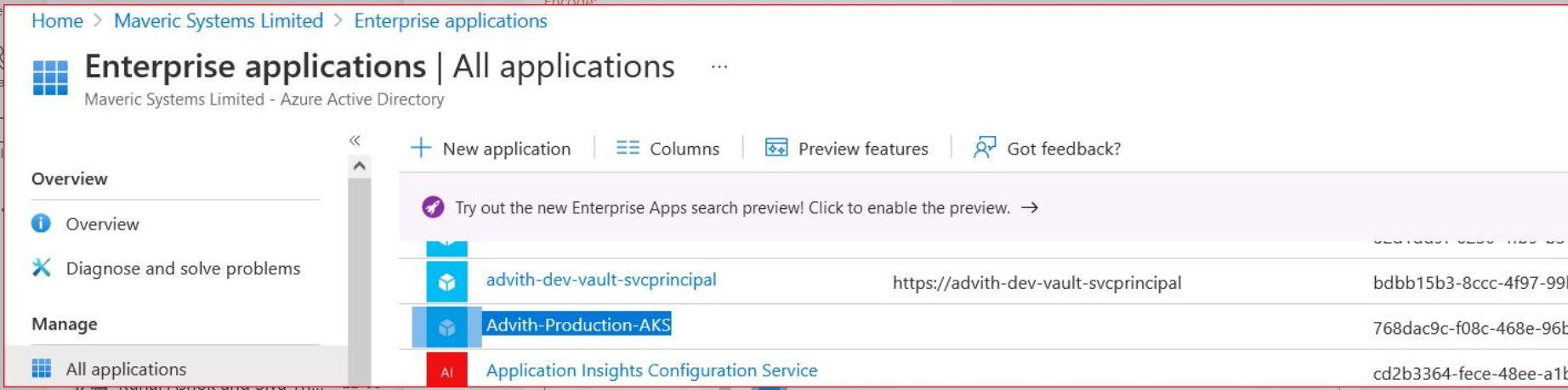
**az role assignment create --assignee <appId> --scope <resourceScope> --role Reader**

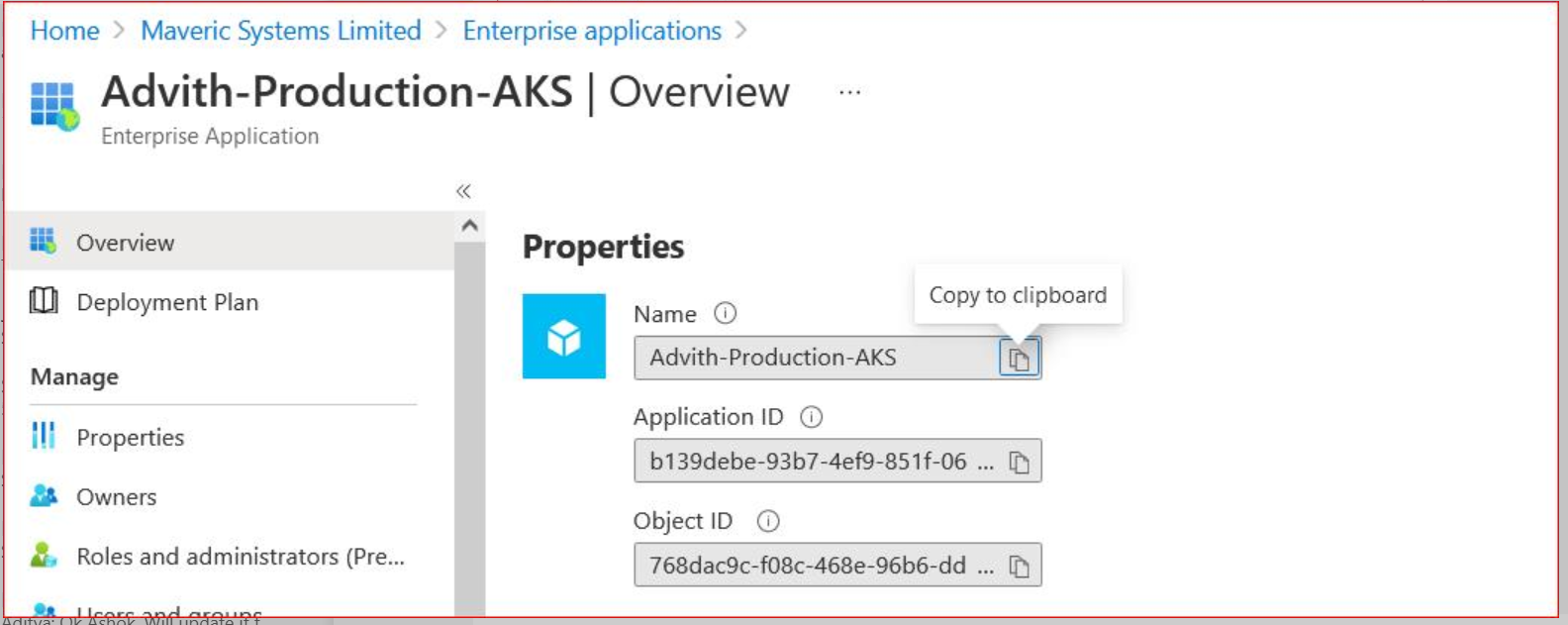
Ex:

• az role assignment create --assignee b139debe-93b7-4ef9-851f- 06a4d0ac98bc --scope /subscriptions/80c62d6f-637d-4687-89c3-f0f4dd0fb33f/resourceGroups/Advith-Staging/providers/Microsoft.Network/virtualNetworks/Advith-Staging-AKS-VNet/subnets/Advith-Production-AKS-Subnet --role Contributor

• az role assignment create --assignee b139debe-93b7-4ef9-851f- 06a4d0ac98bc --scope /subscriptions/80c62d6f-637d-4687-89c3-f0f4dd0fb33f/resourceGroups/Advith-Staging/providers/Microsoft.Network/virtualNetworks/Advith-Staging-AKS-VNet/subnets/Advith-Production-AKS-Subnet --role Reader

In the above command appid refers to your AKS cluster application id. For this navigate Azure Active directory -> Enterprise applications -> choose application type as "all applications" -> search for the name of your AKS cluster.





* Create an internal ingress yaml file with the chosen private IP address in the first step.

controller:

service:

loadBalancerIP: 10.3.0.226

annotations:

service.beta.kubernetes.io/azure-load-balancer-internal: "true"

* Add and update helm repo for nginx ingress controller

**helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx**

**helm repo update**

* Install nginx ingress controller

**helm install nginx-ingress ingress-nginx/ingress-nginx \**

**--namespace pre-prod \**

**-f internal-ingress.yaml \**

**--set controller.replicaCount=1 \**

**--set controller.nodeSelector."beta\.kubernetes\.io/os"=linux \**

**--set defaultBackend.nodeSelector."beta\.kubernetes\.io/os"=linux \**

**--set controller.admissionWebhooks.patch.nodeSelector."beta\.kubernetes\.io/os"=linux**

* 1. **Configure ingress-rules.yaml:**

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

annotations:

kubernetes.io/ingress.class: nginx

name: ingress-rule

spec:

rules:

- http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: web-service

port:

number: 80

- path: /advith-uat/login

pathType: Prefix

backend:

service:

name: web-service

port:

number: 80

- path: /api/auth

pathType: Prefix

backend:

service:

name: advithuaaservice

port:

number: 80

- path: /api/util

pathType: Prefix

backend:

service:

name: advithutilservices

port:

number: 80

- path: /api/itr

pathType: Prefix

backend:

service:

name: advithitrservice

port:

number: 80

- path: /api/support

pathType: Prefix

backend:

service:

name: advithsupportservice

port:

number: 80

1. AZURE APPLICATION GATEWAY

**Create Azure application gateway**

Tier - STANDARD

Firewall mode - Prevention

Enable autoscaling - No

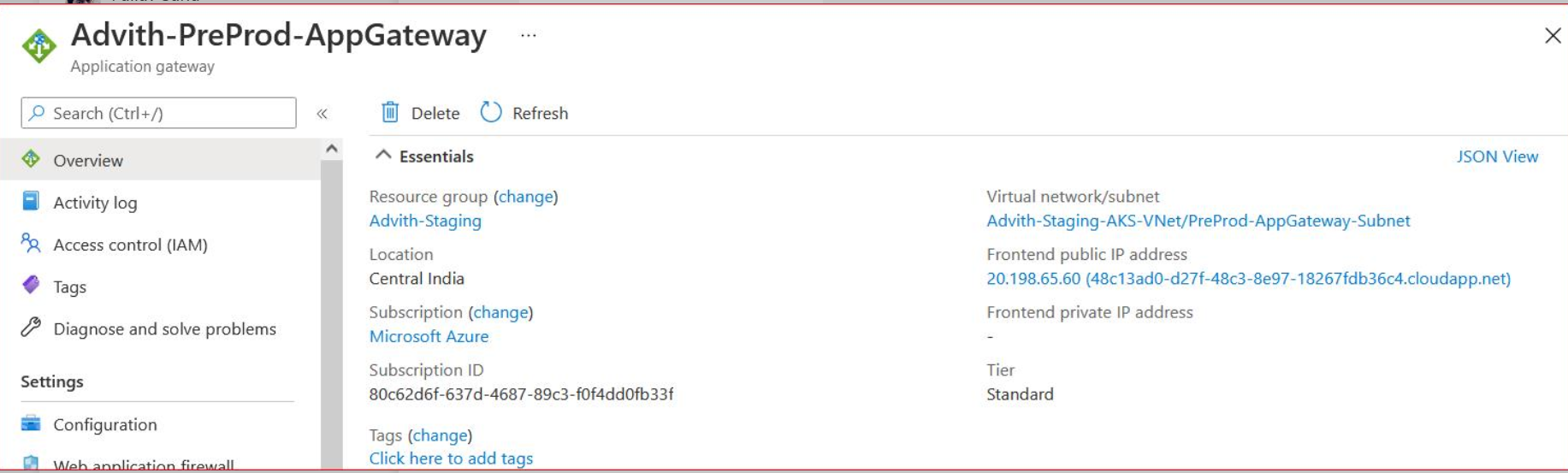
Instance count - 1

SKU size - Small

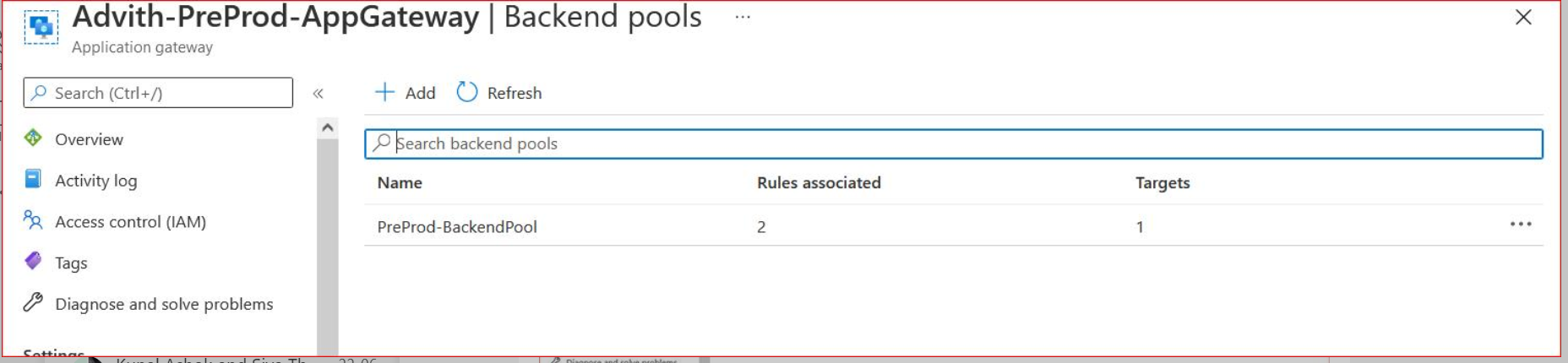
Virtual network - Choose same VNet in which AKS cluster exists

Subnet - Create a new one for app gateway

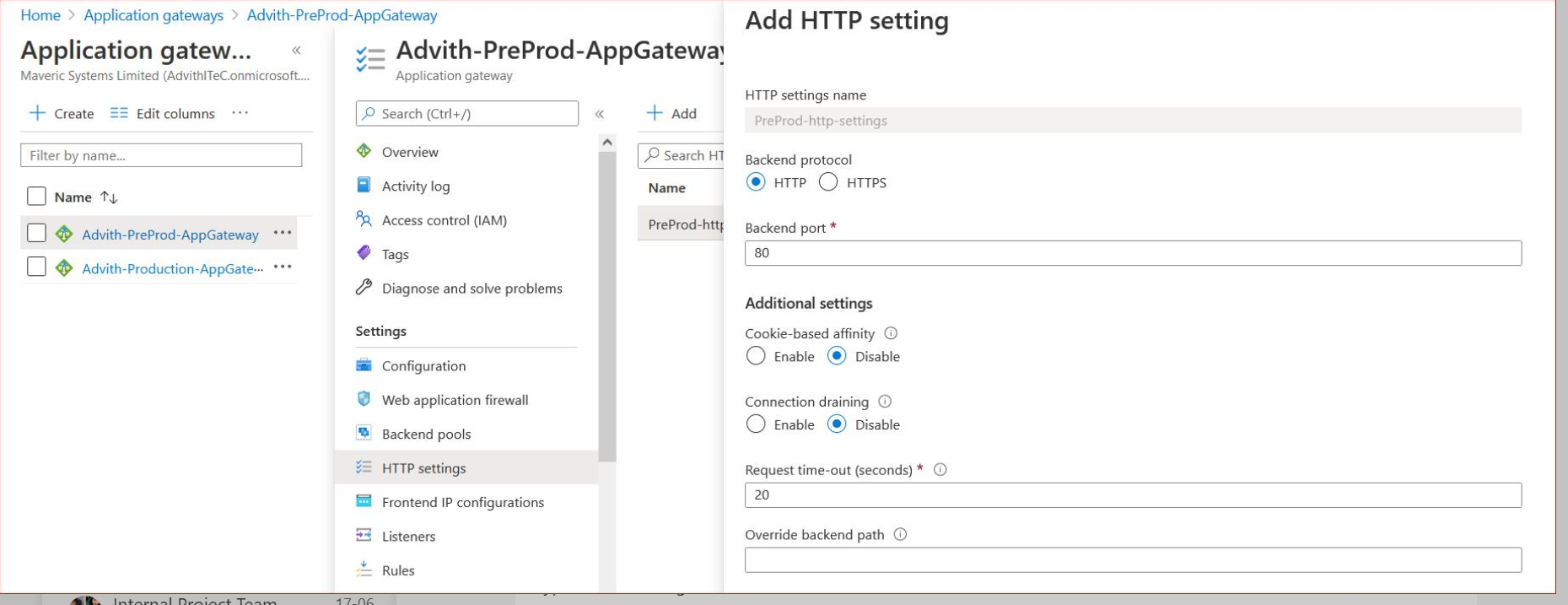
Public IP address - Create one static public IP

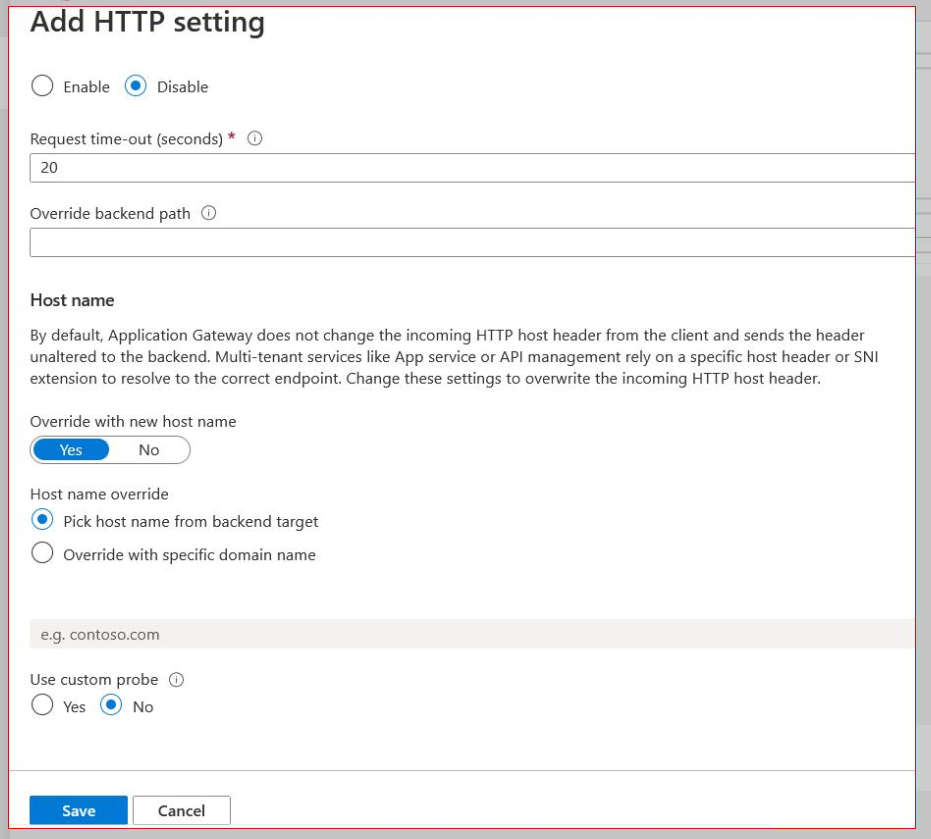


**Backend pool:**

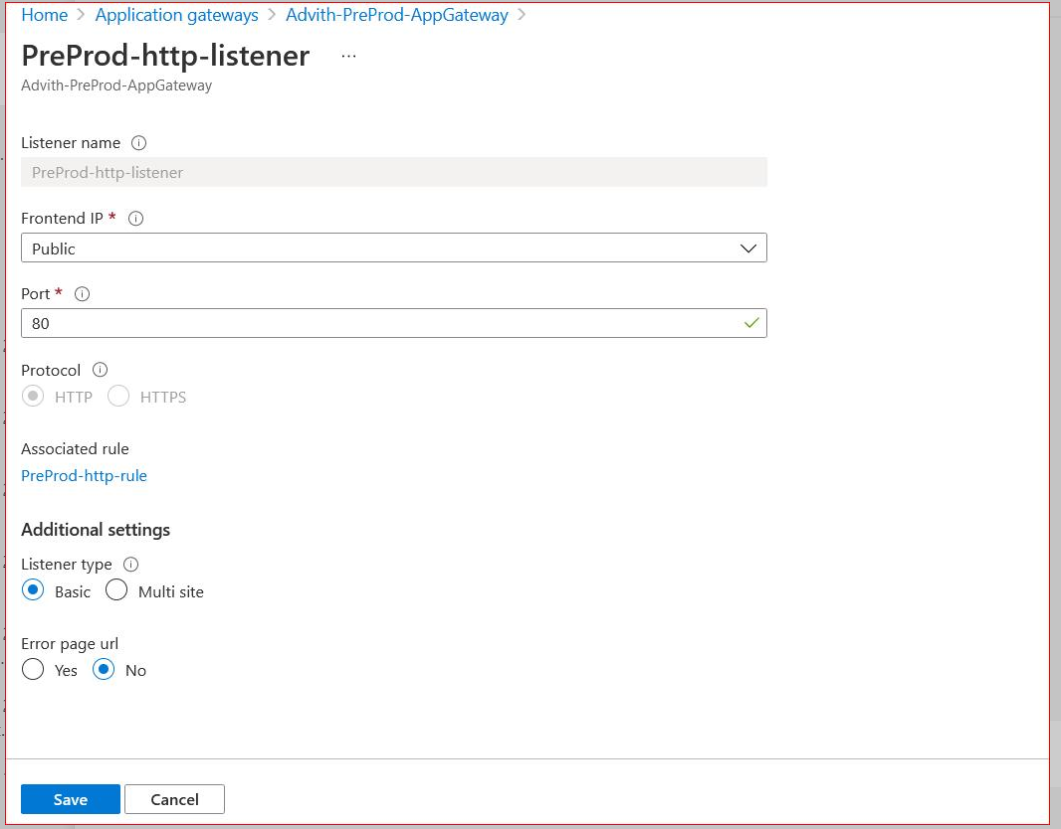


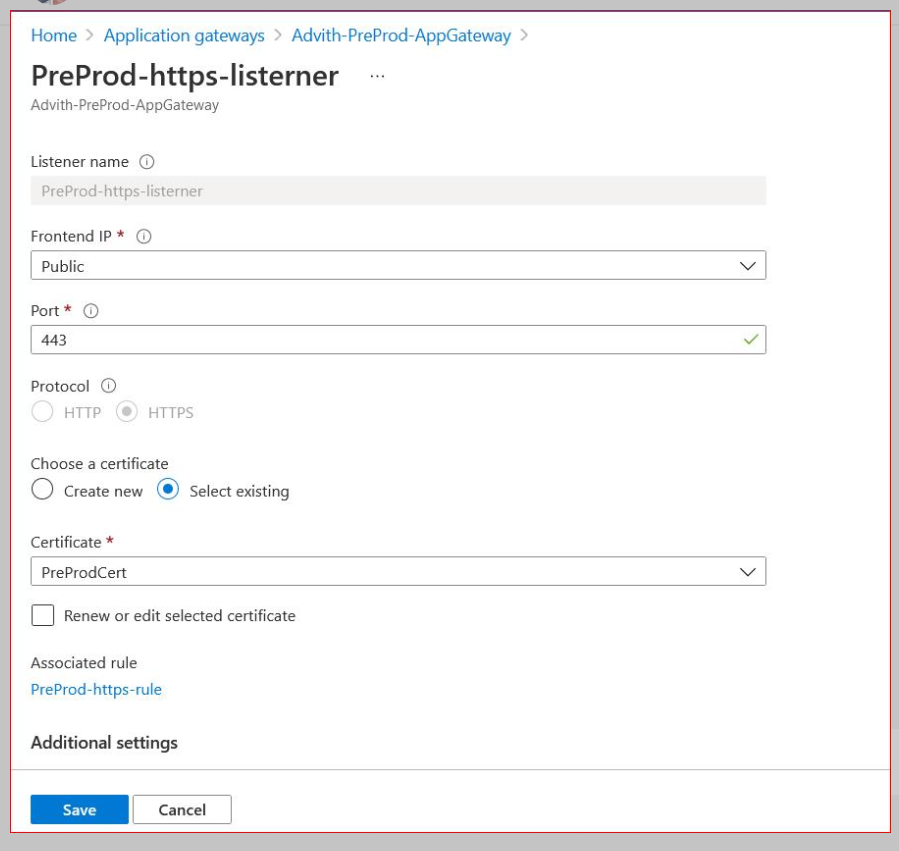
**HTTP settings:**



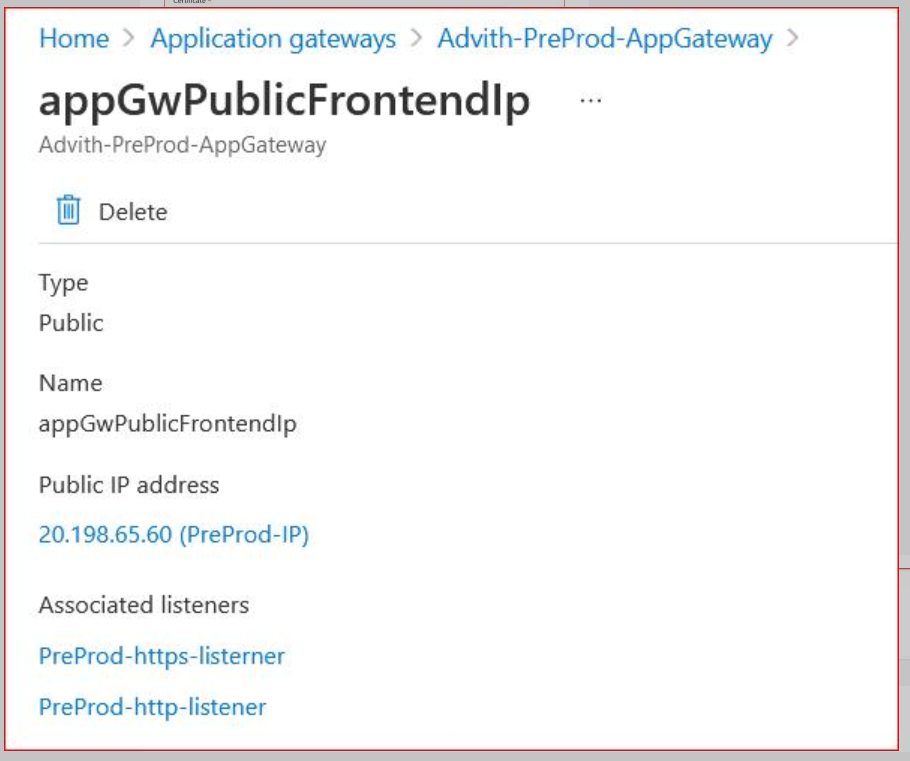


**Listeners:**

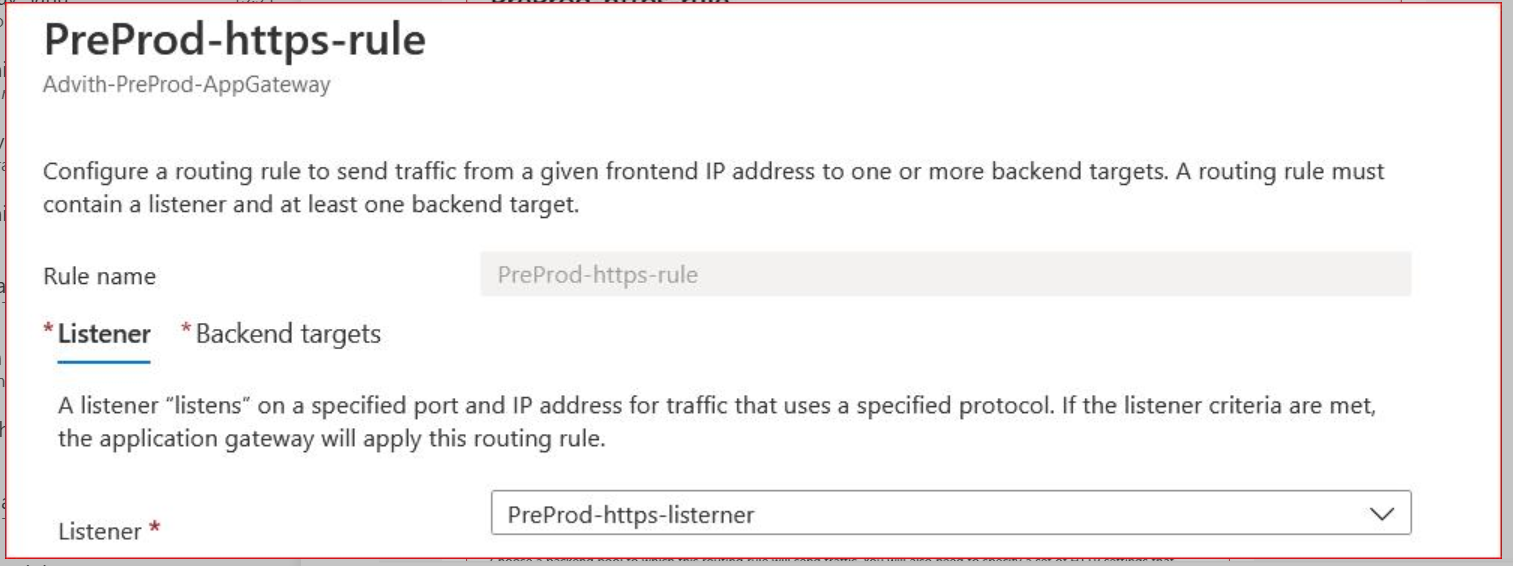


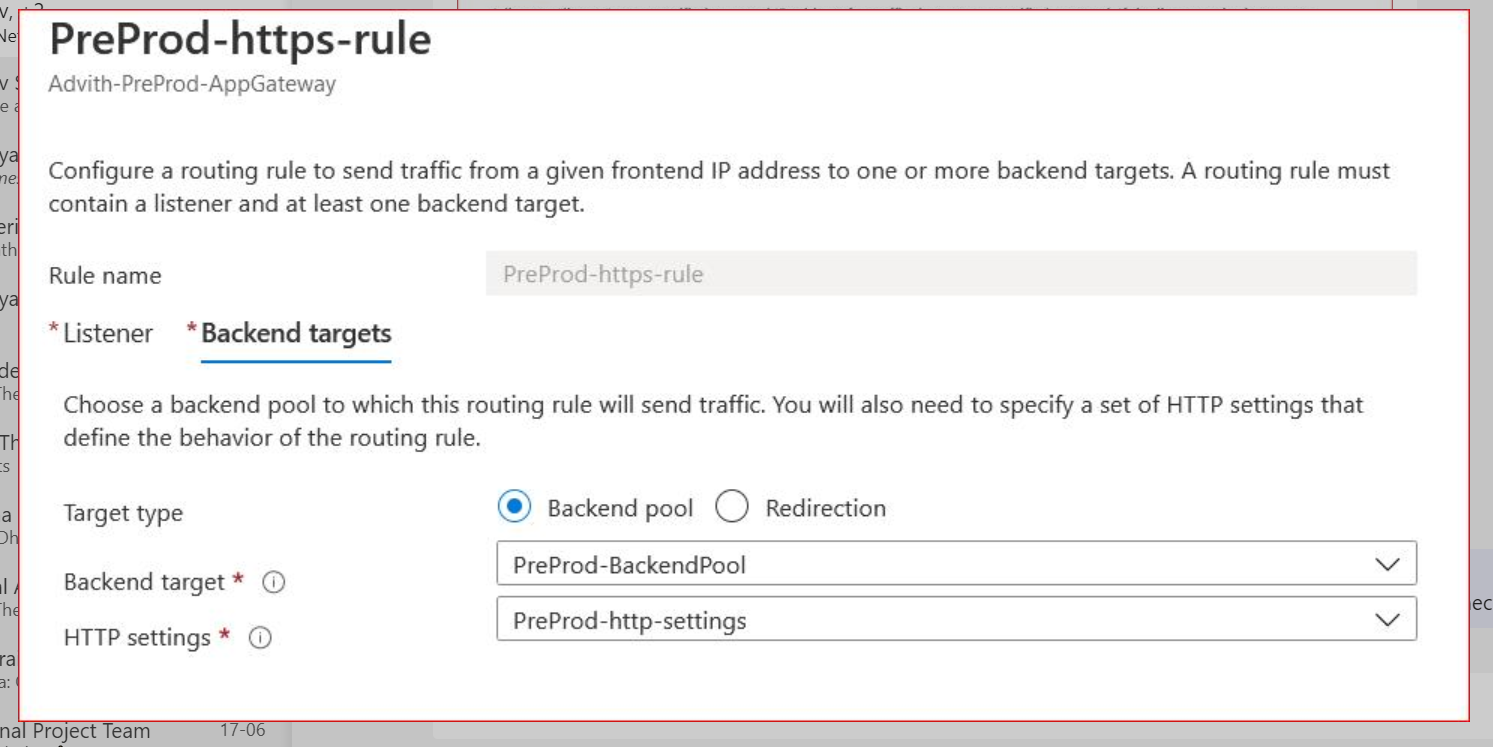


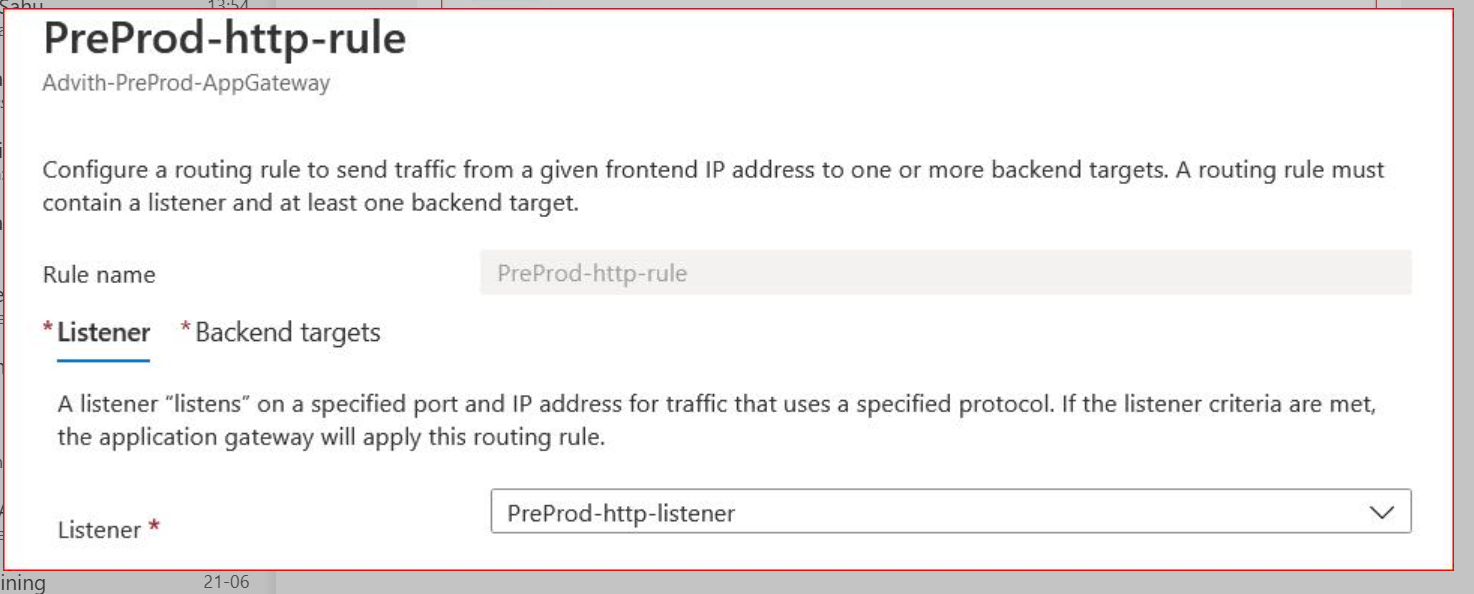
**Frontend IP configurations:**

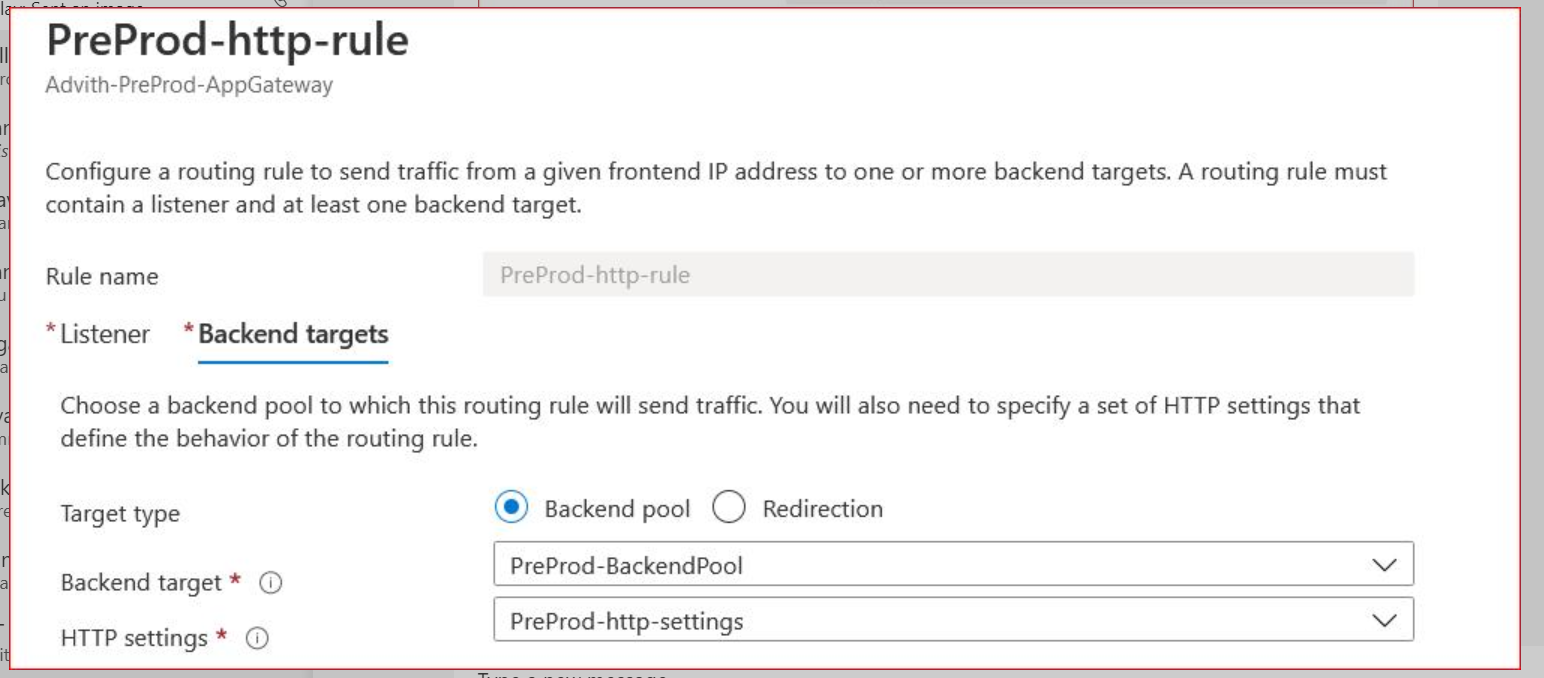


**Rules:**

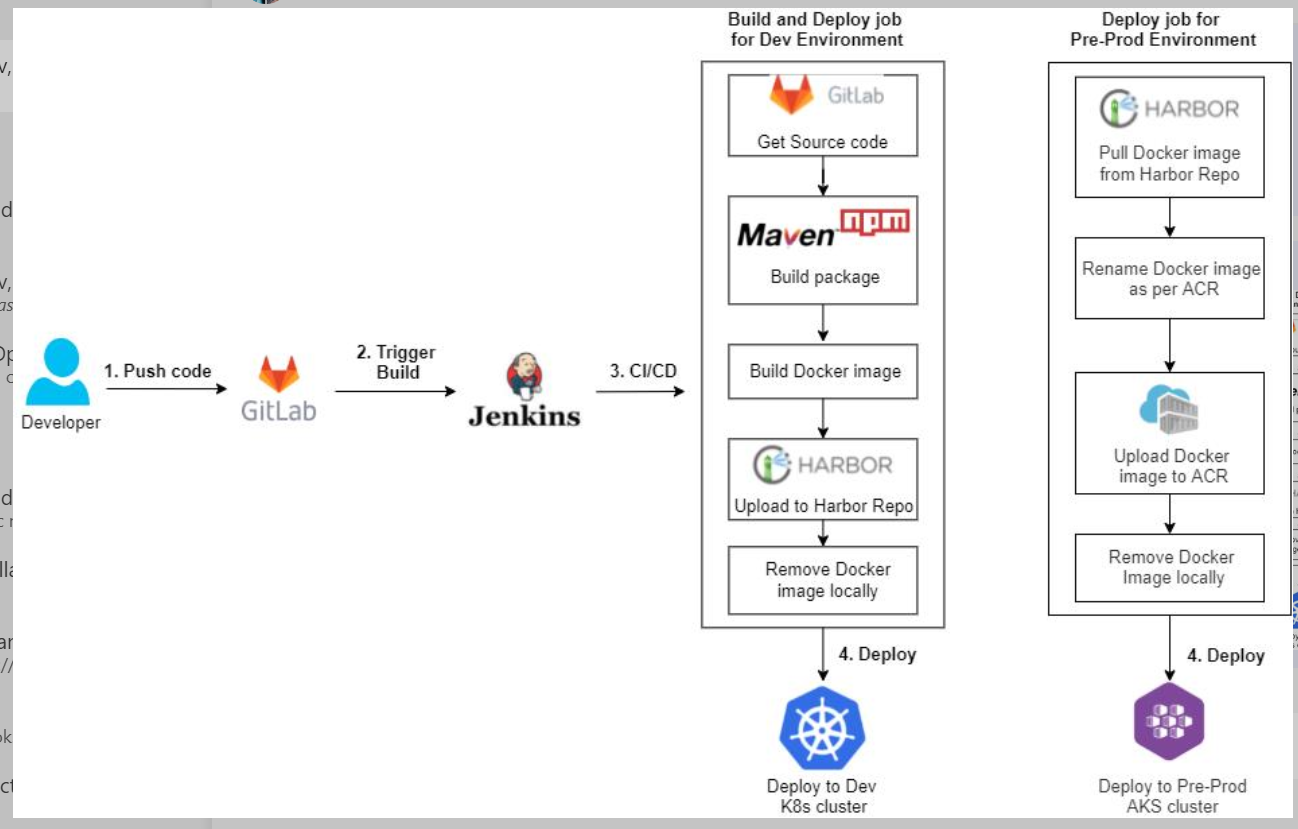








1. CI/CD ARCHITECTURE



The CI/CD for Dev and pre-prod environments is configured in such a way that

**In Dev environment,** we will build the code, build docker image, push the resultant docker image to Harbor Repo and deploy it to Dev Kubernetes cluster.

**In pre-prod environment**, we are not going to build the code and docker image again. Instead we will directly deploy specific version of docker image which is already been deployed in dev environment. So, here, we will pull specific version of docker image from Harbor repo, tag it accordingly to push to Azure container registry and deploy it to Pre-Prod AKS cluster.

Note:

The above mentioned CI/CD scenario is not applicable for frontend, as developers are maintaining two different branches for Dev and Pre-Prod environments. So for frontend, we will build the code, build docker image, push the resultant docker image to ACR and deploy it to pre-prod AKS cluster.

GitLab URL: <http://172.16.238.105:9090/>

Jenkins URL: <http://172.16.238.113:8080/>

Harbor URL: <http://172.16.238.123/>

1. JENKINS JOB CONFIGURATION

* Here we are using 172.16.236.148 server (with label name: advith-dev) as jenkins slave for Pre-Prod environment.
* First, configure Azure container registry (ACR) credentials and Harbor repo credentials in jenkins global credentials with kind Username and password

Create a jenkins job of type “pipeline”. Choose “This job is parameterized” and create two string parameters, one being “**BRANCH**” as **PreprodBranch** and other being “**NAMESPACE”** as **pre-prod.**

**13.1 Jenkins job for frontend service:**

**Parameter name - Default value**

pipeline {

agent {

label 'advith-dev'

}

parameters {

string name: 'BRANCH', defaultValue: 'PreprodBranch'

string name: 'NAMESPACE', defaultValue: 'pre-prod'

}

stages {

stage ('Code checkout') {

steps {

git branch: "${params.BRANCH}", credentialsId: 'gitlab\_cred', url: 'http://172.16.238.105:9090/advith-finance-system/web-advithfin.git'

}

}

stage ('npm Build') {

steps {

sh """

cd /var/lib/jenkins/workspace/Advith-PreProd-Web

npm install

npm run-script build:preprod

"""

}

}

stage ('Build Docker image') {

steps {

script {

TAG = sh(script: "git describe --tags --abbrev=0", returnStdout: true).trim()

sh "docker build -t advithstagingacr.azurecr.io/advithpreprod/web-advithfin:$TAG ."

}

}

}

stage('Upload Docker image to ACR') {

steps{

withCredentials([usernamePassword(credentialsId: 'Advith-Staging-ACR-Credentials', passwordVariable: 'ACRPassword', usernameVariable: 'ACRUser')]) {

sh "docker login http://advithstagingacr.azurecr.io -u ${env.ACRUser} -p ${env.ACRPassword}"

sh "docker push advithstagingacr.azurecr.io/advithpreprod/web-advithfin:$TAG"

}

}

}

TAG - Specify respective docker tag name here

NAMESPACE - pre-prod

* 1. **Jenkins job for Frontend service:**

|  |
| --- |
| stage('Remove Docker Image locally') {  steps {  sh "docker rmi advithstagingacr.azurecr.io/advithpreprod/web-advithfin:$TAG"  }  }  stage('Deploy to AKS') {  steps{  sh "kubectl config use-context Advith-Production-AKS"  sh "kubectl set image deployment/web-advithfin -n $NAMESPACE web=advithstagingacr.azurecr.io/advithpreprod/web-advithfin:$TAG"  }  }  }  } |

**13.2 Jenkins job for backend service:**

**13.2.1 UAA**

<Pipeline script yet to be added>

**13.3 Error while pushing/pulling docker image from harbor repo:**

Get <https://172.16.238.123/v2/>: dial tcp 172.16.238.123:443: connect: no route to host

**Solution:**

In Jenkins slave machine ie., 172.16.238.148, create daemon.json file at location /etc/docker and add Harbor repo ip as shown below.

vi /etc/docker/daemon.json

Add ip of harbor

{

"insecure-registries" : ["172.16.238.123"]

}

Once added do docker deamon reload and restart

sudo systemctl daemon-reload

sudo systemctl restart docker