DevOps-Project-2-Tier-WebAppDeployment-using-AWSandAzure

By Ritesh Kumar Singh

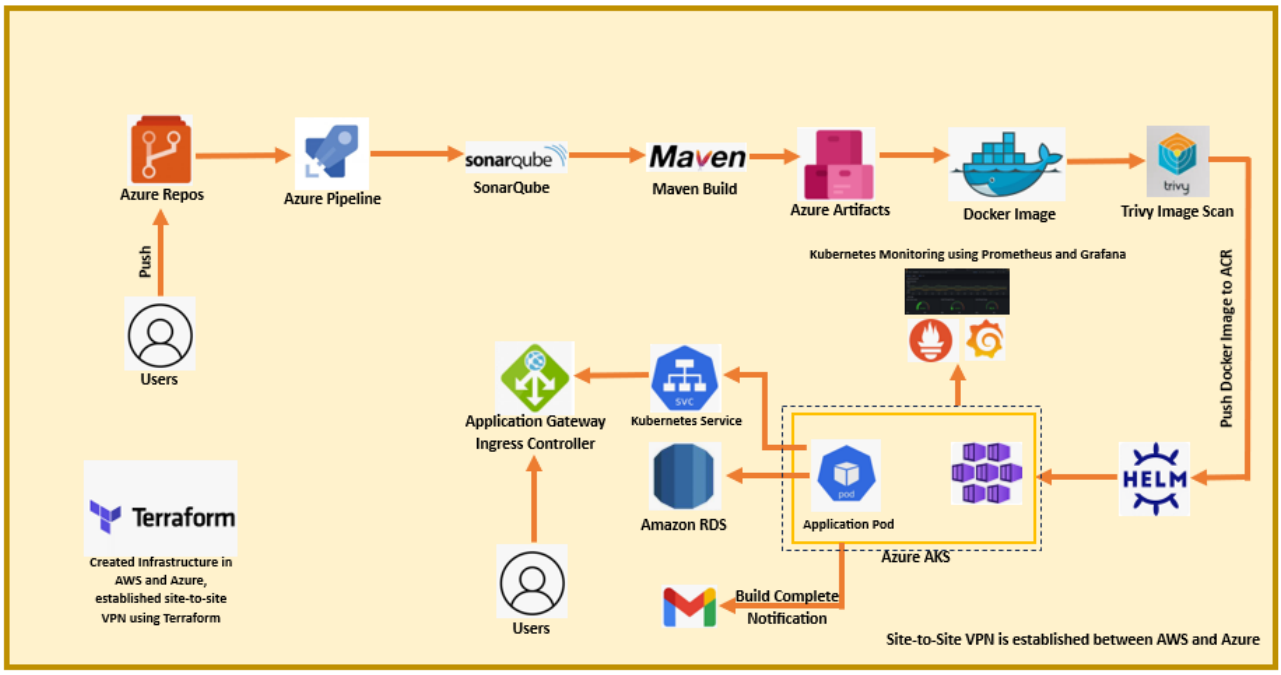
Email Address: - riteshkumarsingh9559@gmail.com

LinkedIn: - https://www.linkedin.com/in/ritesh-kumar-singh-41113128b/

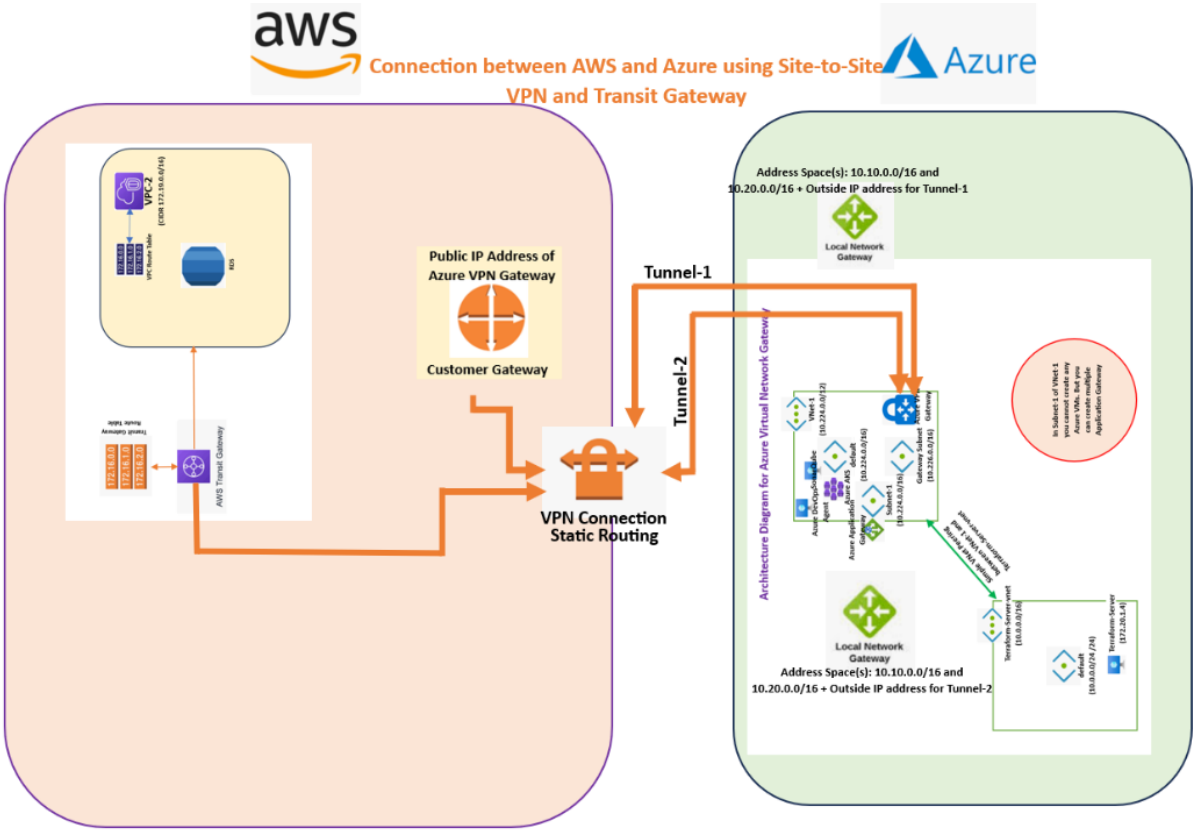
GitHub: - https://github.com/singhritesh85

या कुन्देन्दुतुषारहारधवला या शुभ्रवस्त्रावृता
या वीणावरदण्डमण्डितकरा या श्वेतपद्मासना।
या ब्रह्माच्युत शंकरप्रभृतिभिर्देवैः सदा वन्दिता
सा मां पातु सरस्वती भगवती निःशेषजाड्यापहा ॥

DevOps-Project-2-Tier-WebAppDeployment-using-AWSandAzure



End-to-end Architecture diagram of the project



Connection between AWS and Azure using Site-to-Site VPN

This project aims to create two tier web app deployment (CI/CD) and the two-tier web application was running in the tomcat-based pod which had been created in the AKS Cluster. You can also monitor the AKS Cluster using the prometheus and Grafana installed in the AKS Cluster with the help of helm. I had created Azure file share dynamically using the storage provisioner **file.csi.azure.com** and through the storage class and used this file share to create the PV and PVC dynamically (dynamic provisioning). And hence the prometheus and grafana had persistent storage. I had used Azure DevOps as the CI/CD tool and sonarqube had been used for code analysis. I had used Azure artifacts to store the Artifacts for this project. In this project I used Maven as the build tool and trivy to scan the vulnerabilities present in the docker image. Then pushed the Docker Image to ACR (Azure Container Registries) and finally with the helm of helm it was deployed to the AKS Cluster and pod had been created.

For this Project I used MySQL RDS database and for SonarQube I used PostgreSQL RDS database both were present in the AWS Cloud. To achieve the same my first requirement was to establish the secure connection between Azure cloud and Aws Cloud for this I used the Site-to-Site VPN. I had achieved this using the terraform script present in my GitHub Repo https://github.com/singhritesh85/DevOps-Project-2-Tier-WebAppDeployment-using-AWSandAzure.git. This secure connection had been established between the Azure VNet in which AKS Cluster existed and Aws VPC in which RDS existed. These two RDSs (MySQL and PostgreSQL) were privately accessible.

Before proceeding further, I configured the MySQL RDS and PostgreSQL RDS as shown in the screenshot attached below.

Configuration for MySQL

```
[root@devopsagent-vm ]# yum install -y mysql
```

```
[root@devopsagent-vm ]# mysql -h dbinstance-2.c w.us-east-2.rds.amazonaws.com -u admin --password
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 59
Server version: 5.7.44 Please upgrade to 8.0 or opt-in to the paid RDS Extended Support service before 5.7 reaches end of standard support on 29 February, 202
4: https://a.co/h 0

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database jwt;
Query OK, 1 row affected (0.03 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| innodb             |
| jwt                |
| mydb               |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
7 rows in set (0.03 sec)
```

```
mysql> use jwt;
Database changed
```

```
mysql> CREATE TABLE `USER` (
    -> `id` int(10) unsigned NOT NULL auto_increment,
    -> `first_name` varchar(45) NOT NULL,
    -> `last_name` varchar(45) NOT NULL,
    -> `email` varchar(45) NOT NULL,
    -> `username` varchar(45) NOT NULL,
    -> `password` varchar(45) NOT NULL,
    -> `regdate` date NOT NULL,
    -> PRIMARY KEY  (`id`)
    -> ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
Query OK, 0 rows affected (0.07 sec)

mysql> show tables;
+---------------+
| Tables_in_jwt |
+---------------+
| USER          |
+---------------+
1 row in set (0.02 sec)
```

Then uninstall mysql from devopsagent server as shown in the screenshot attached below.

```
[root@devopsagent-vm ]# yum remove -y mysql
```

Now I **configured the PostgreSQL** for SonarQube as shown in the screenshot attached below.

```
[root@devopsagent-vm ~]# yum install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-8-x86_64/pgdg-redhat-repo-latest.noarch.rpm
```

```
[root@devopsagent-vm ~]# yum -qy module disable postgresql
```

```
[root@devopsagent-vm ~]# yum install -y postgresql14
```

```
[root@devopsagent-vm ~]# psql -h dbinstance-1.c          w.us-east-2.rds.amazonaws.com -U postgres --password
Password:
psql (14.17, server 14.12)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=> create database sonarqubedb;
CREATE DATABASE
postgres=> create user sonarqube with encrypted password 'Cloud#436';
CREATE ROLE
postgres=> grant all privileges on database sonarqubedb to sonarqube;
GRANT
postgres=> \q
```

Then uninstall PostgreSQL as shown in the screenshot attached below.

```
[root@devopsagent-vm ~]# yum remove -y postgresql14
```

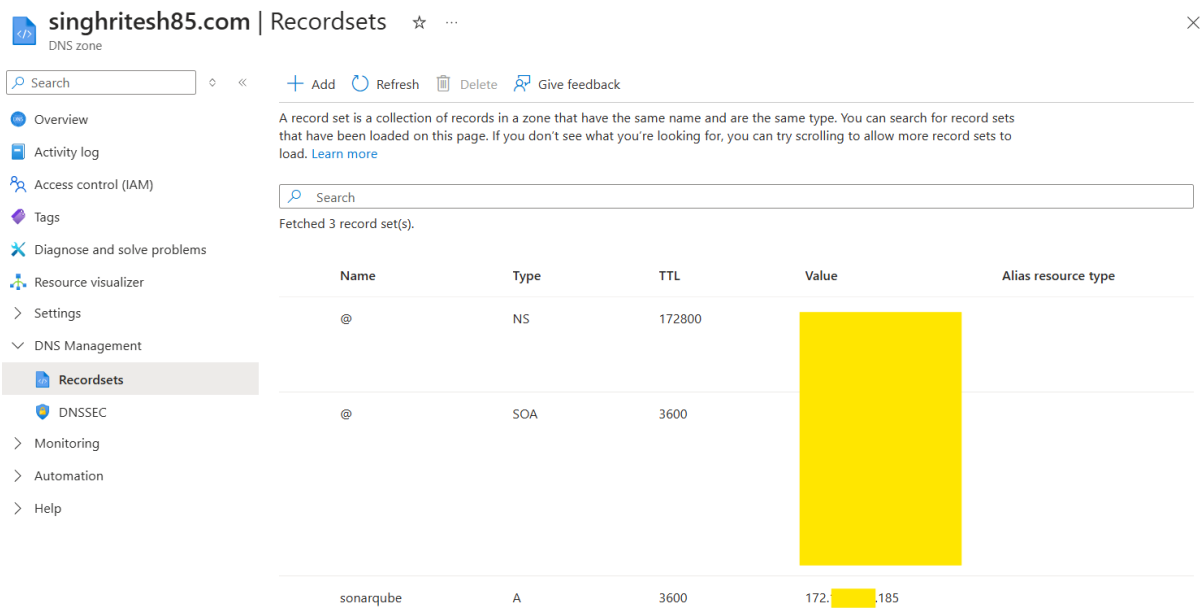
Configured the database information in sonarqube as shown in the screenshot attached below.

```
[root@SonarQube-Server ~]# vim /opt/sonarqube/conf/sonar.properties
```

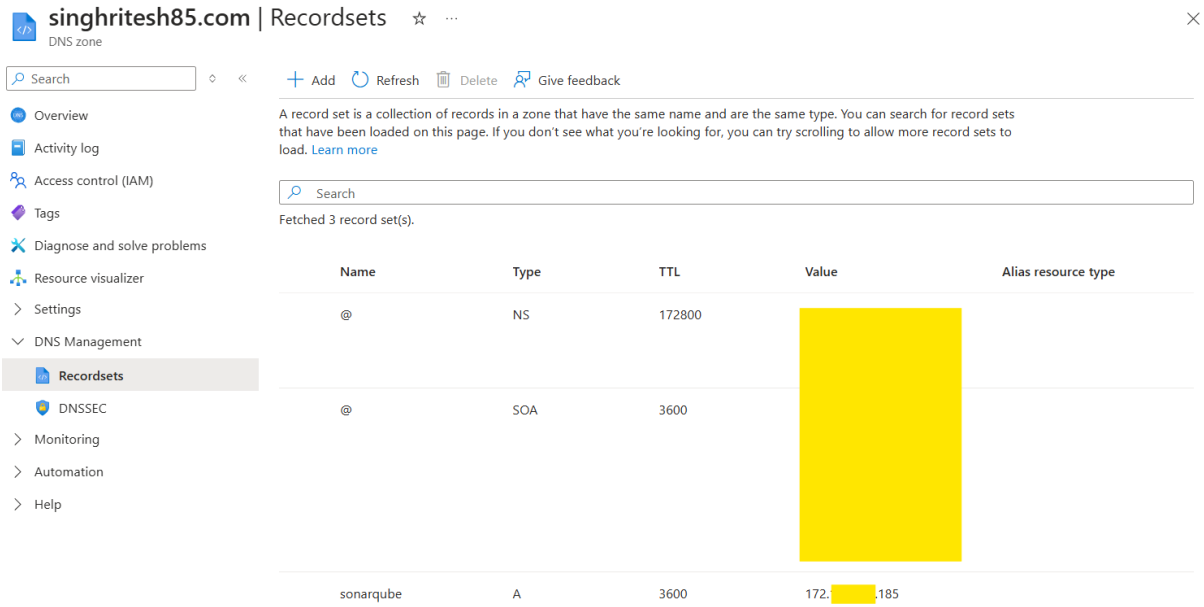
```
# The schema must be created first.
sonar.jdbc.username=s[redacted]e
sonar.jdbc.password=C[redacted]6

#----- Embedded Database (default)
# H2 embedded database server listening port, defaults to 9092
#sonar.embeddedDatabase.port=9092


#----- Oracle 19c/21c
# The Oracle JDBC driver must be copied into the directory extensions/jdbc-driver/oracle/.
# Only the thin client is supported, and we recommend using the latest Oracle JDBC driver. See
# https://jira.sonarsource.com/browse/SONAR-9758 for more details.
# If you need to set the schema, please refer to http://jira.sonarsource.com/browse/SONAR-5000
#sonar.jdbc.url=jdbc:oracle:thin:@localhost:1521/XE


#----- PostgreSQL 11 or greater
# By default the schema named "public" is used. It can be overridden with the parameter "currentSchema".
sonar.jdbc.url=jdbc:postgresql://dbinstance-1.c[redacted]w.us-east-2.rds.amazonaws.com/sonarqubedb
```

```
[root@SonarQube-Server ~]# systemctl start sonarqube.service
[root@SonarQube-Server ~]# systemctl enable sonarqube.service
Created symlink /etc/systemd/system/multi-user.target.wants/sonarqube.service → /etc/systemd/system/sonarqube.service.
[root@SonarQube-Server ~]# systemctl status sonarqube.service
● sonarqube.service - SonarQube service
   Loaded: loaded (/etc/systemd/system/sonarqube.service; enabled; vendor preset: disabled)
   Active: active (running) since Thu 2025-[redacted]UTC; 16s ago
```

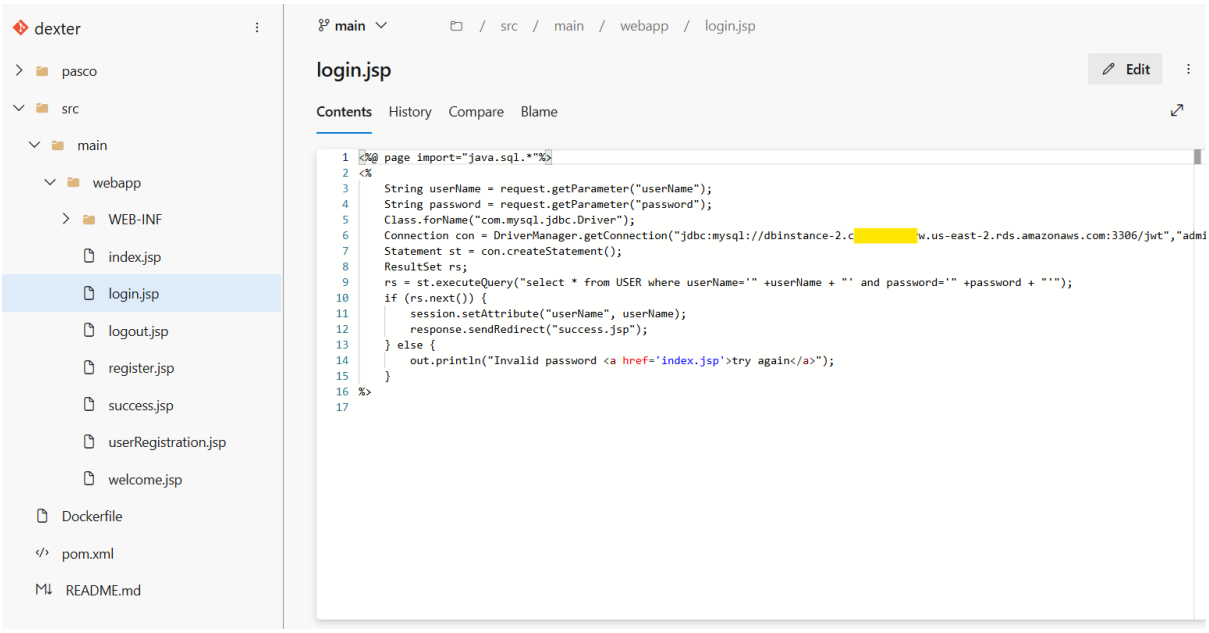
Then get the Public IP address of the Sonarqube Application Gateway and do the entry in the Azure DNS Zone to create the record set as shown in the screenshot attached below.

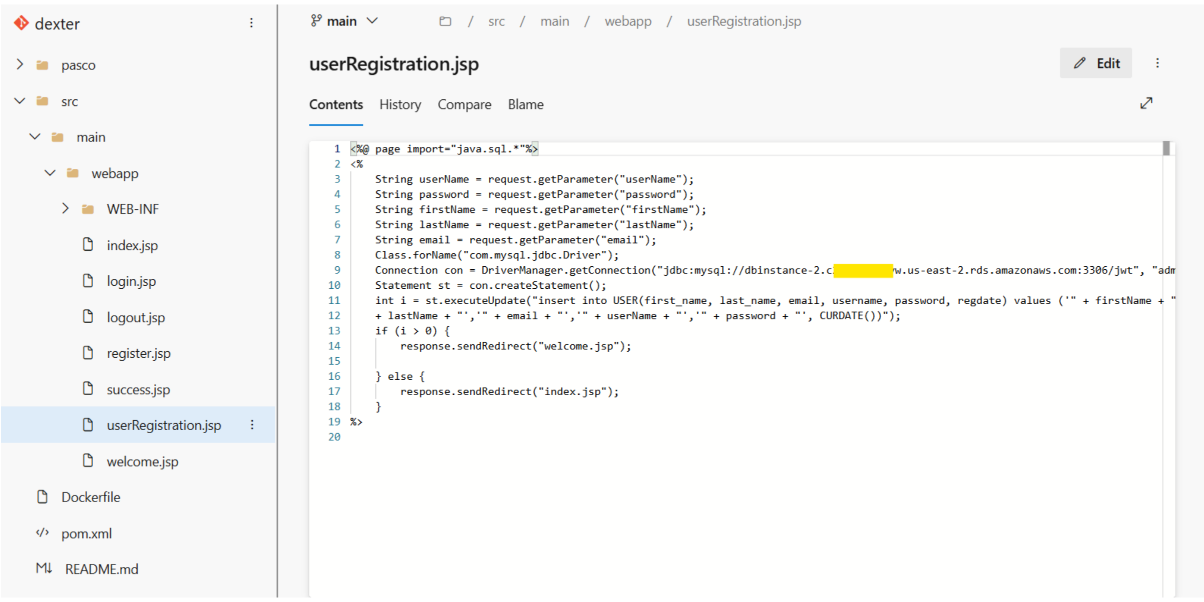


Using the URL I tried to access the SonarQube and found that I was able to access the SonarQube using the URL as shown in the screenshot attached below.



After creation of the two RDS, share the endpoint of MySQL RDS with your development team so that they can update the code according. For the current scenario you need to update the endpoint of MySQL RDS in the file **src/main/webapp/login.jsp** and **src/main/webapp/userRegistration.jsp** and of the source code as shown in the screenshot attached below.





Create Kubernetes Secret to be used for pull image from Azure ACR

kubectl create secret docker-registry devopsmelacr132827a7-auth --docker-server=https://akscontainerregistry2405.azurecr.io --docker-username=akscontainerregistry2405 --docker-password=iXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXp -n demo

In Below command I created the Kubernetes secrets for Grafana and the Application Ingress Rule

kubectl create secret tls ingress-secret --key mykey.key --cert STAR_singhritesh85_com.crt -n demo

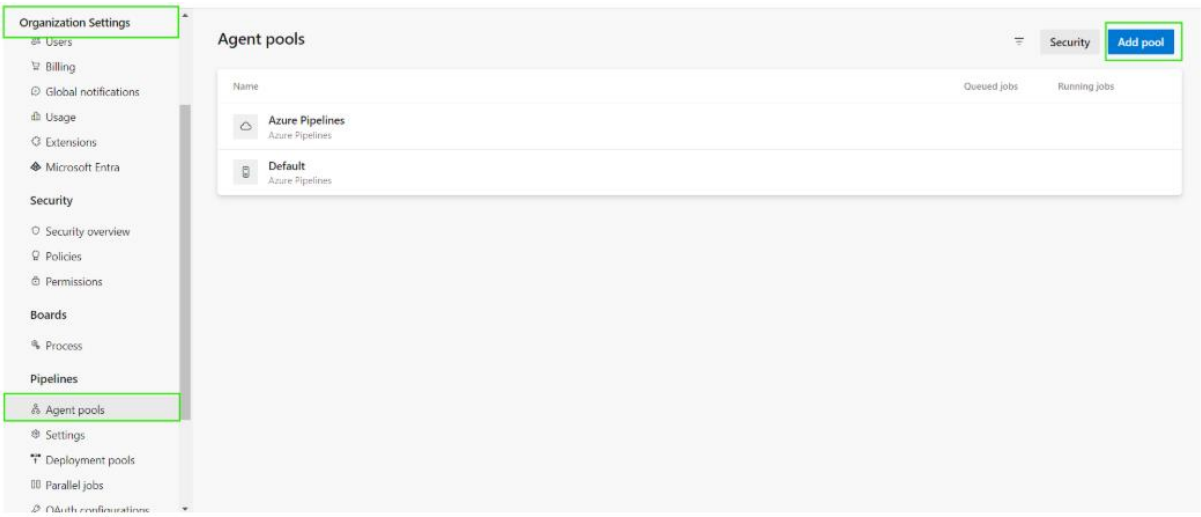
kubectl create secret tls ingress-secret --key mykey.key --cert STAR_singhritesh85_com.crt -n monitoring

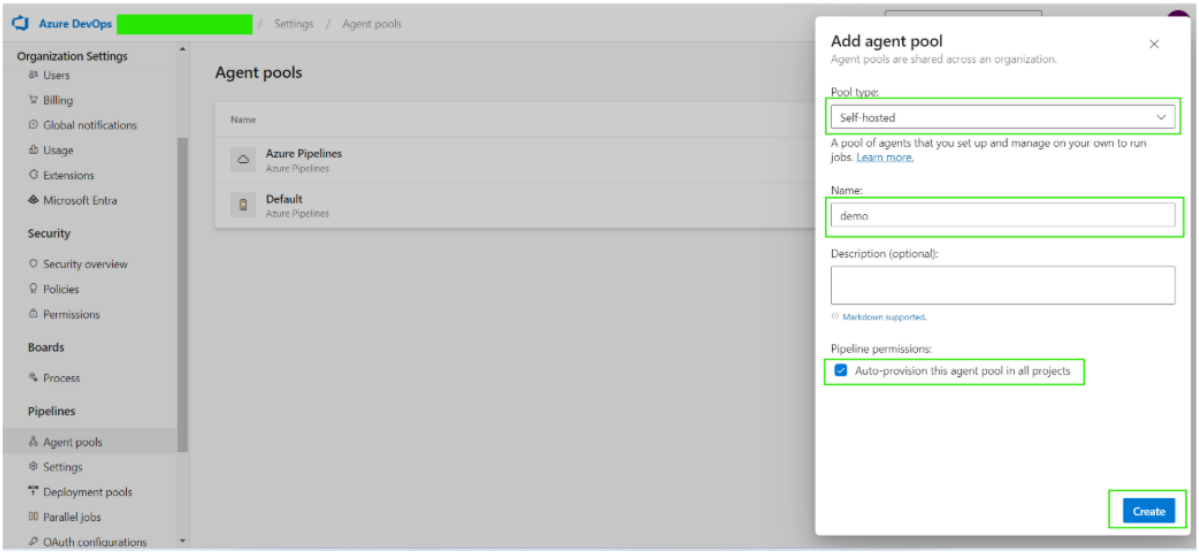
```
[demo@[redacted]~]$ kubectl create secret docker-registry devopsmelacr132827a7-auth --docker-server=https://akscontainerregistry2405.azurecr.io --docker-username=akscontainerregistry2405 --docker-password=N[redacted]N -n demo
```

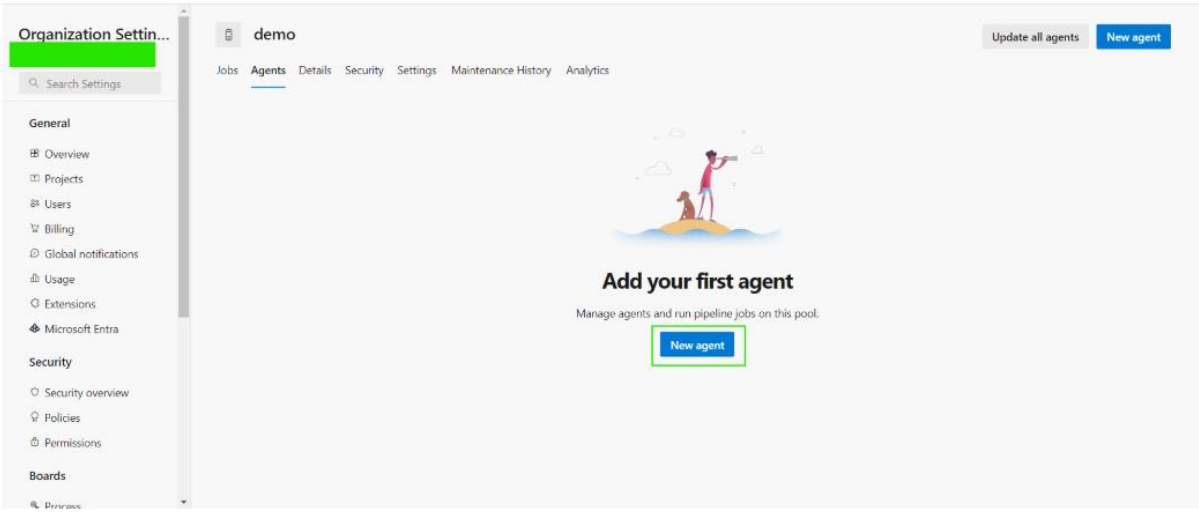
```
[root@[redacted]~]# kubectl create secret tls ingress-secret --key mykey.key --cert STAR_singhritesh85_com.crt -n demo
secret/ingress-secret created
```

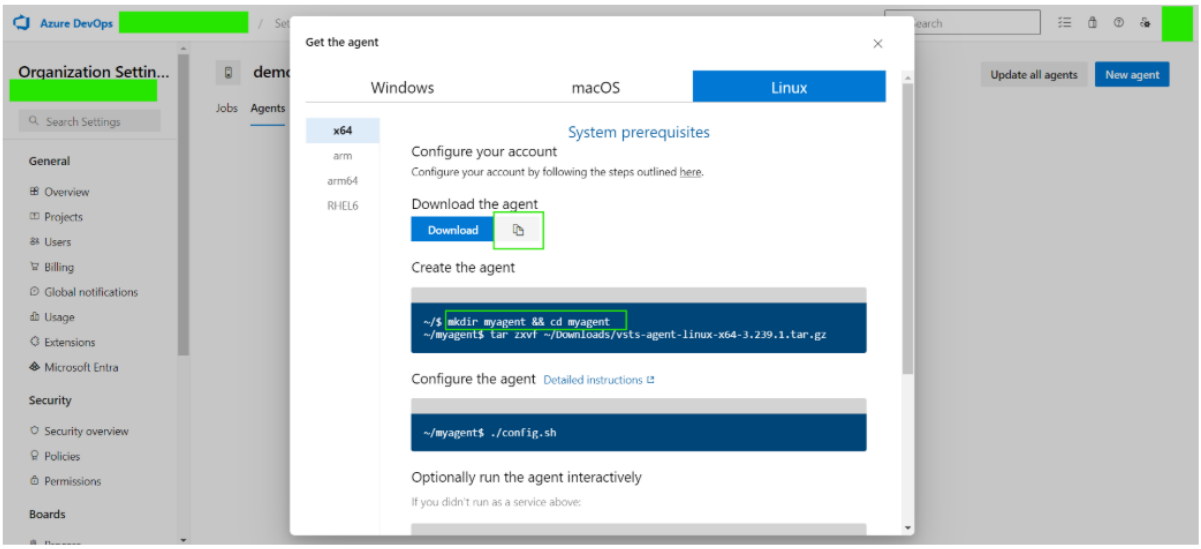
```
[root@[redacted]~]# kubectl create secret tls ingress-secret --key mykey.key --cert STAR_singhritesh85_com.crt -n monitoring
secret/ingress-secret created
```

Install self-hosted agent pool (You can create self-hosted agent pool either at Organisation level or at project level). For this project I created the self-hosted agent pool at Organisation level.


Organization Settings
Users
Billing
Global notifications
Usage
Extensions
Microsoft Entra
Security
Security overview
Policies
Permissions
Boards
Process
Pipelines
Agent pools
Settings
Deployment pools
Parallel jobs
OAuth configurations
Agent pools
Security
Add pool
Name
Queued jobs
Running jobs
Azure Pipelines
Azure Pipelines
Default
Azure Pipelines


Azure DevOps
Settings
Agent pools
Organization Settings
Users
Billing
Global notifications
Usage
Extensions
Microsoft Entra
Security
Security overview
Policies
Permissions
Boards
Process
Pipelines
Agent pools
Settings
Deployment pools
Parallel jobs
OAuth configurations
Agent pools
Name
Azure Pipelines
Azure Pipelines
Default
Azure Pipelines
Add agent pool
Agent pools are shared across an organization.
Pool type:
Self-hosted
A pool of agents that you set up and manage on your own to run jobs. Learn more.
Name:
demo
Description (optional):
Markdown supported.
Pipeline permissions:
Auto-provision this agent pool in all projects
Create


Organization Settin...
Search Settings
General
Overview
Projects
Users
Billing
Global notifications
Usage
Extensions
Microsoft Entra
Security
Security overview
Policies
Permissions
Boards
demo
Update all agents
New agent
Jobs
Agents
Details
Security
Settings
Maintenance History
Analytics
Add your first agent
Manage agents and run pipeline jobs on this pool.
New agent




```
[demo@devopsagent-vm opt]$ sudo rm -rf myagent/
[demo@devopsagent-vm opt]$ mkdir myagent && cd myagent
mkdir: cannot create directory ‘myagent’: Permission denied
[demo@devopsagent-vm opt]$ logout
[root@devopsagent-vm myagent]# chown -R demo:demo /opt/
[root@devopsagent-vm myagent]# su - demo
Last login: Thu Mar 20 10:55:12 UTC 2025 on pts/0
[demo@devopsagent-vm ~]$ mkdir myagent && cd myagent
[demo@devopsagent-vm myagent]$ https://vstsagentpackage.azureedge.net/agent/4.252.0/vsts-agent-linux-x64-4.252.0.tar.gz
-bash: https://vstsagentpackage.azureedge.net/agent/4.252.0/vsts-agent-linux-x64-4.252.0.tar.gz: No such file or directory
[demo@devopsagent-vm myagent]$ wget https://vstsagentpackage.azureedge.net/agent/4.252.0/vsts-agent-linux-x64-4.252.0.tar.gz
--2025-03-20 11:02:49--  https://vstsagentpackage.azureedge.net/agent/4.252.0/vsts-agent-linux-x64-4.252.0.tar.gz
Resolving vstsagentpackage.azureedge.net (vstsagentpackage.azureedge.net)... 23.215.0.17, 23.215.0.9, 2600:141b:f000:4::17c8:c2, ...
Connecting to vstsagentpackage.azureedge.net (vstsagentpackage.azureedge.net)|23.215.0.17|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 147444824 (141M) [application/octet-stream]
Saving to: ‘vsts-agent-linux-x64-4.252.0.tar.gz’

vsts-agent-linux-x64-4.252.0.tar.gz     100%[=========================================================================>] 140.61M   234MB/s    in 0.6s

2025-03-20 11:02:50 (234 MB/s) - ‘vsts-agent-linux-x64-4.252.0.tar.gz’ saved [147444824/147444824]

[demo@devopsagent-vm myagent]$ tar -xvf vsts-agent-linux-x64-4.252.0.tar.gz
```

```
[demo@devopsagent-vm myagent]$ ls
bin  config.sh  env.sh  externals  license.html  reauth.sh  run-docker.sh  run.sh  vsts-agent-linux-x64-4.252.0.tar.gz
[demo@devopsagent-vm myagent]$ rm -f vsts-agent-linux-x64-4.252.0.tar.gz
[demo@devopsagent-vm myagent]$ sudo ./bin/installdependencies.sh
--------OS Information--------
NAME="AlmaLinux"
VERSION="8.7 (Stone Smilodon)"
ID="almalinux"
ID_LIKE="rhel centos fedora"
VERSION_ID="8.7"
PLATFORM_ID="platform:el8"
PRETTY_NAME="AlmaLinux 8.7 (Stone Smilodon)"
ANSI_COLOR="0;34"
LOGO="fedora-logo-icon"
CPE_NAME="cpe:/o:almalinux:almalinux:8::baseos"
HOME_URL="https://almalinux.org/"
DOCUMENTATION_URL="https://wiki.almalinux.org/"
BUG_REPORT_URL="https://bugs.almalinux.org/"

ALMALINUX_MANTISBT_PROJECT="AlmaLinux-8"
ALMALINUX_MANTISBT_PROJECT_VERSION="8.7"
REDHAT_SUPPORT_PRODUCT="AlmaLinux"
REDHAT_SUPPORT_PRODUCT_VERSION="8.7"
--------------------------------
The current OS is Fedora based
--------Redhat Version--------
AlmaLinux release 8.7 (Stone Smilodon)
--------------------------------
```

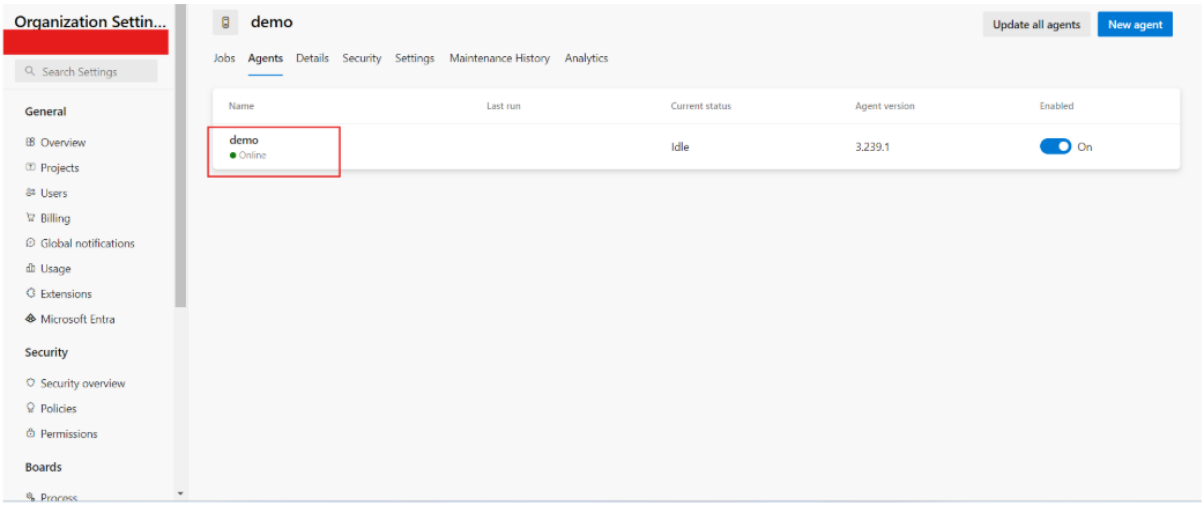
```
[demo@devopsagent-vm myagent]$ ./config.sh

  ___                      ______ _            _ _
 / _ \                     | ___ (_)          | (_)
/ /_\ \_____  _ _ __ ___   | |_/ /_ _ __   ___| |_ _ __   ___
|  _  |_  / | | | '__/ _ \ |  __/| | '_ \ / _ \ | | '_ \ / _ \
| | | |/ /| |_| | | |  __/ | |   | | |_) |  __/ | | | | |  __/
\_| |_/___|\__,_|_|  \___| \_|   |_| .__/ \___|_|_|_| |_|\___||___/
                                    | |
       agent v4.252.0               |_|         (commit [redacted])


>> End User License Agreements:

Building sources from a TFVC repository requires accepting the Team Explorer Everywhere End User License Agreement. This step is not required for building sou
rces from Git repositories.

A copy of the Team Explorer Everywhere license agreement can be found at:
  /home/demo/myagent/license.html

Enter (Y/N) Accept the Team Explorer Everywhere license agreement now? (press enter for N) > Y

>> Connect:

Enter server URL > https://[redacted]
Enter authentication type (press enter for PAT) >
Enter personal access token > [redacted]
Connecting to server ...

>> Register Agent:

Enter agent pool (press enter for default) > demo
Enter agent name (press enter for devopsagent-vm) > demo
Scanning for tool capabilities.
Connecting to the server.
Successfully added the agent
Testing agent connection.
Enter work folder (press enter for _work) >
2025-03-20 11:05:28Z: Settings Saved.
```

```
[demo@devopsagent-vm myagent]$ sudo ./svc.sh install
```

```
[demo@devopsagent-vm myagent]$ sudo ./svc.sh start
```

```
[demo@devopsagent-vm myagent]$ sudo ./svc.sh status
```

Finally Agent came in the online state as shown in the screenshot attached below.



Created the helm chart locally using the command **helm create <chart-name>** as shown in the screenshot attached below.

```
[demo@devopsagent-vm ~]$ helm create folo
```

I edited the helm chart as shown in the screenshot attached below.

```
  containers:
    - name: {{ .Chart.Name }}
      securityContext:
        {{- toYaml .Values.securityContext | nindent 12 }}
      image: "{{ .Values.image.repository }}:{{ .Values.image.tag | default .Chart.AppVersion }}"
      imagePullPolicy: {{ .Values.image.pullPolicy }}
      ports:
        - name: http
          containerPort: 8080
          protocol: TCP
      resources:
        {{- toYaml .Values.resources | nindent 12 }}
  {{- with .Values.nodeSelector }}
  nodeSelector:
    {{- toYaml . | nindent 8 }}
  {{- end }}
  {{- with .Values.affinity }}
  affinity:
    {{- toYaml . | nindent 8 }}
```

I removed the readiness and liveness probe from the helm chart.

```
spec:
  type: {{ .Values.service.type }}
  ports:
    - port: {{ .Values.service.port }}
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    {{- include "folo.selectorLabels" . | nindent 4 }}
```

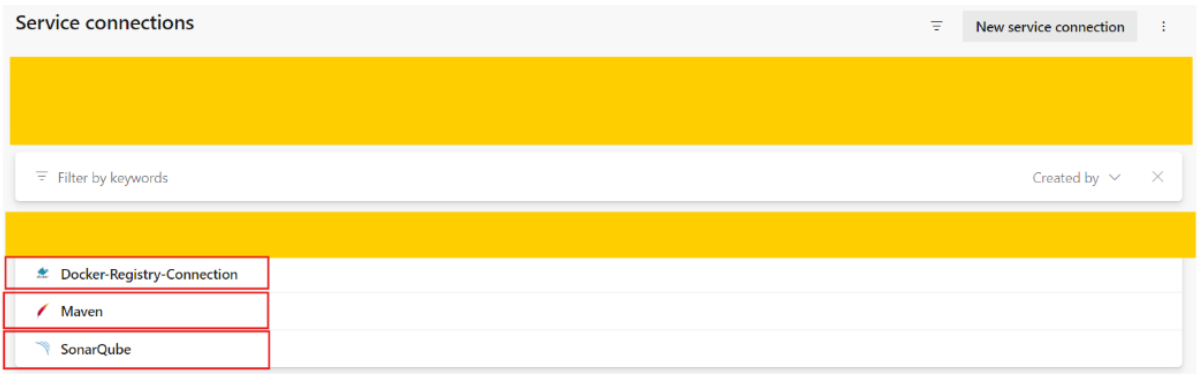
Changed targetPort in service.yaml

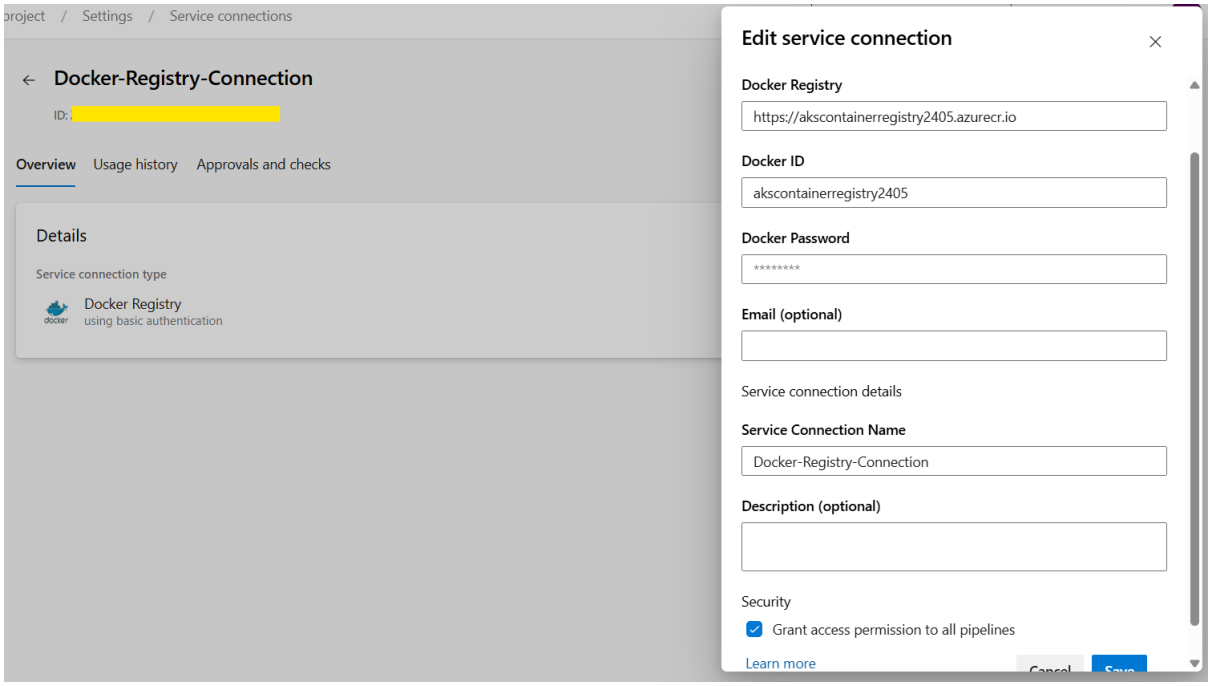
Copied kubeconfig from Terraform-Server to Azure DevOps Self-Hosted Agent.

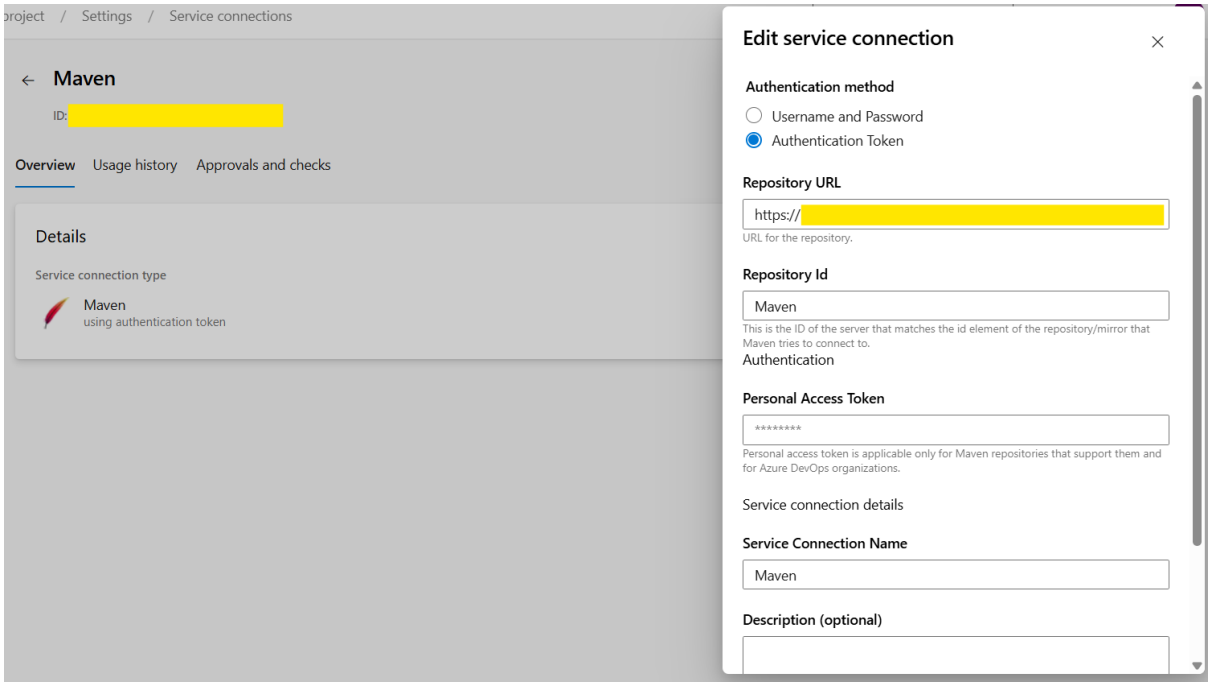
```
[demo@devopsagent-vm ~]$ mkdir ~/.kube
[demo@devopsagent-vm ~]$ vim ~/.kube/config
```

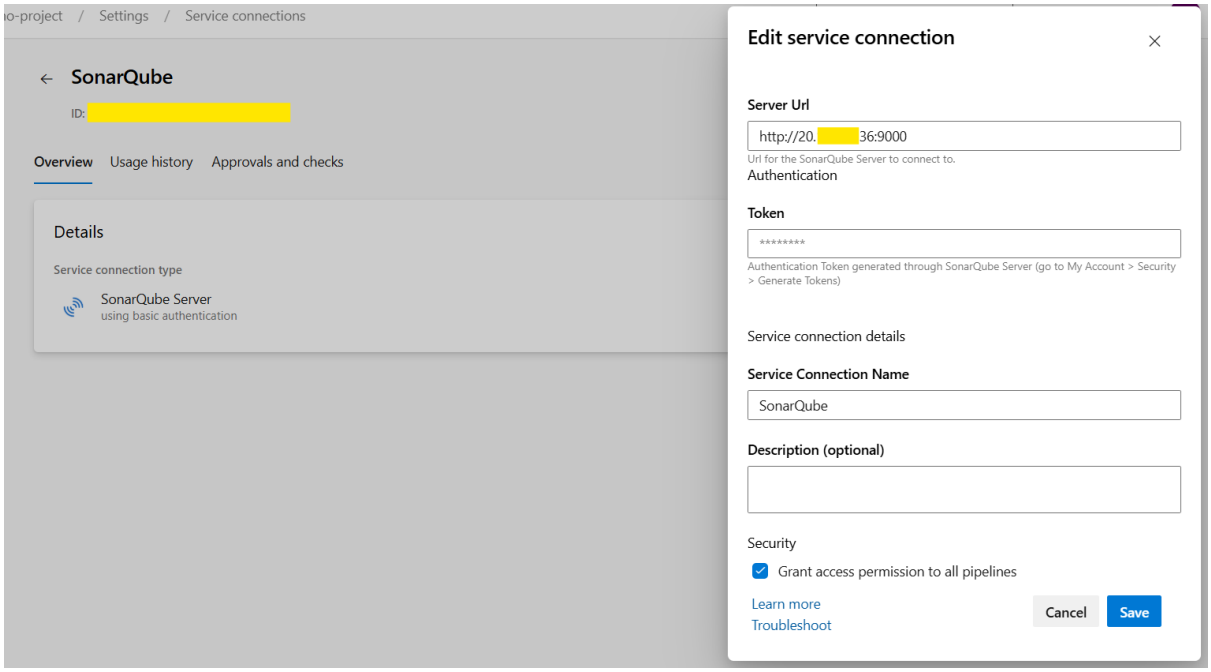
```
[demo@devopsagent-vm ~]$ chmod 600 .kube/config
```

Created three service connections as shown in the screenshot below

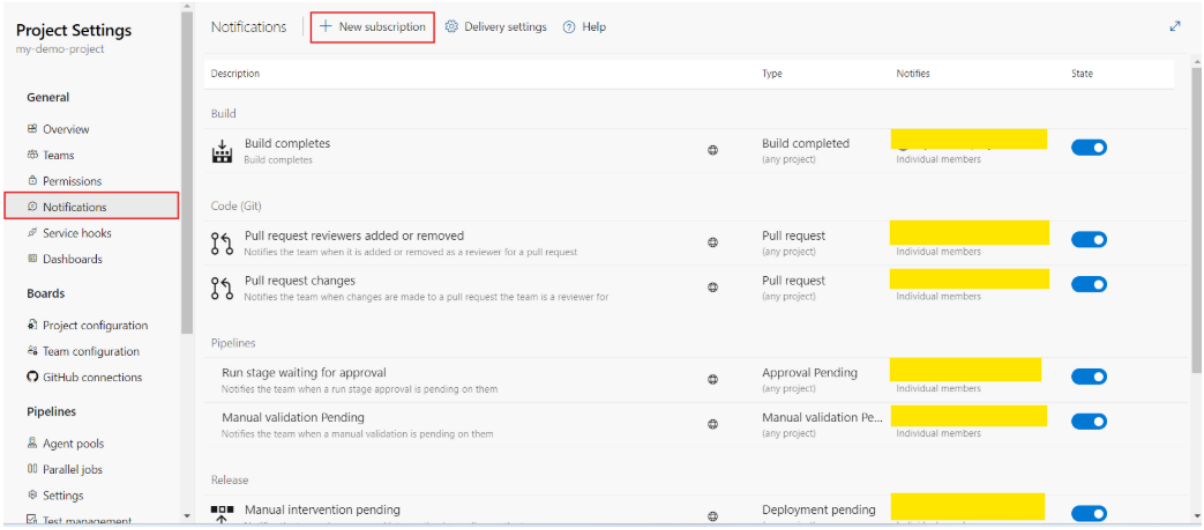


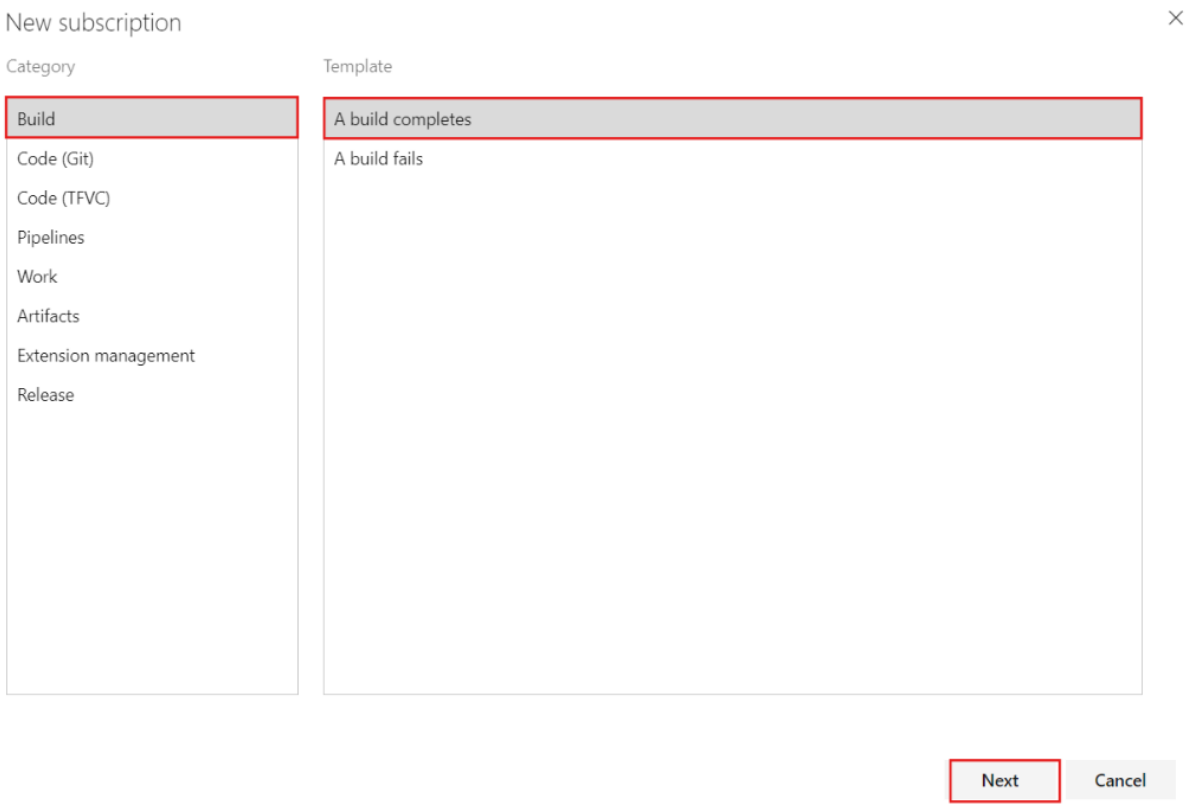

project / Settings / Service connections
Docker-Registry-Connection
ID:
Overview
Usage history
Approvals and checks
Details
Service connection type
Docker Registry
using basic authentication
Edit service connection
Docker Registry
https://akscontainerregistry2405.azurecr.io
Docker ID
akscontainerregistry2405
Docker Password
****
Email (optional)
Service connection details
Service Connection Name
Docker-Registry-Connection
Description (optional)
Security
Grant access permission to all pipelines
Learn more
Cancel
Save

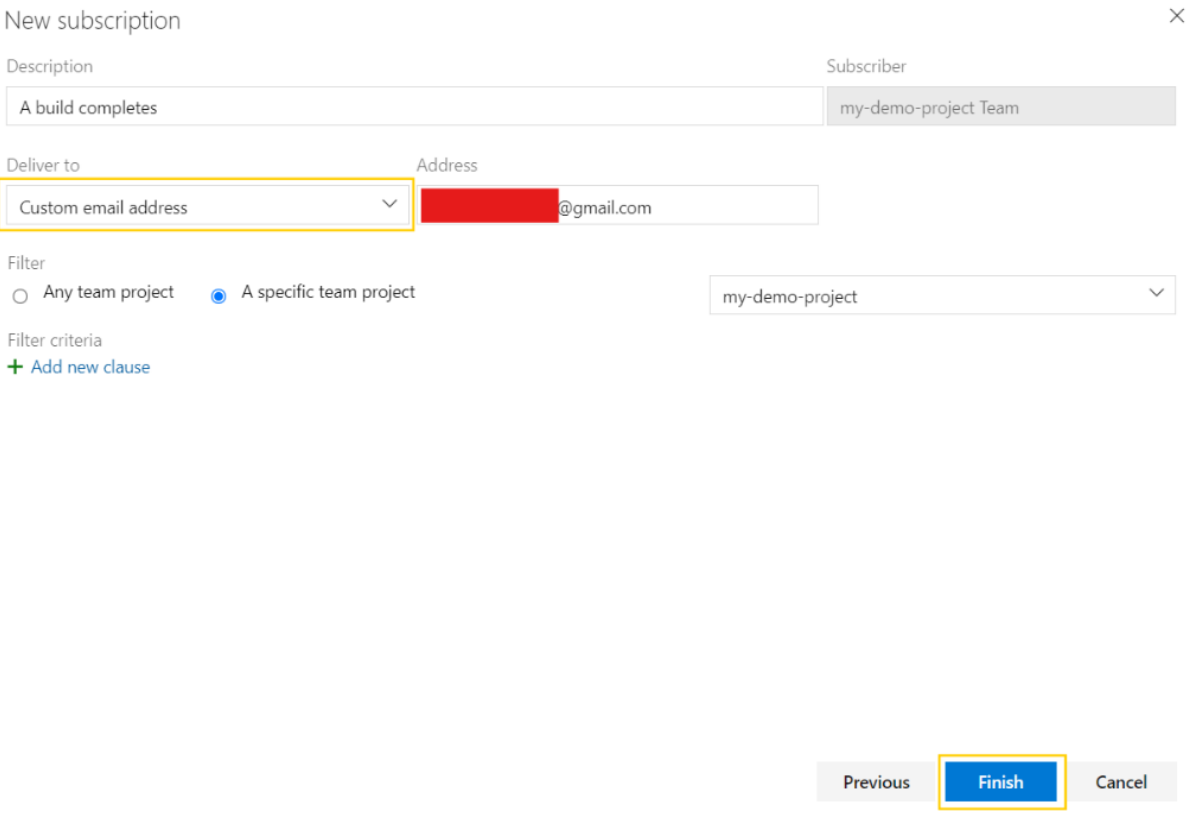

project / Settings / Service connections
Maven
ID:
Overview
Usage history
Approvals and checks
Details
Service connection type
Maven
using authentication token
Edit service connection
Authentication method
Username and Password
Authentication Token
Repository URL
https://
URL for the repository.
Repository Id
Maven
This is the ID of the server that matches the id element of the repository/mirror that Maven tries to connect to.
Authentication
Personal Access Token
****
Personal access token is applicable only for Maven repositories that support them and for Azure DevOps organizations.
Service connection details
Service Connection Name
Maven
Description (optional)

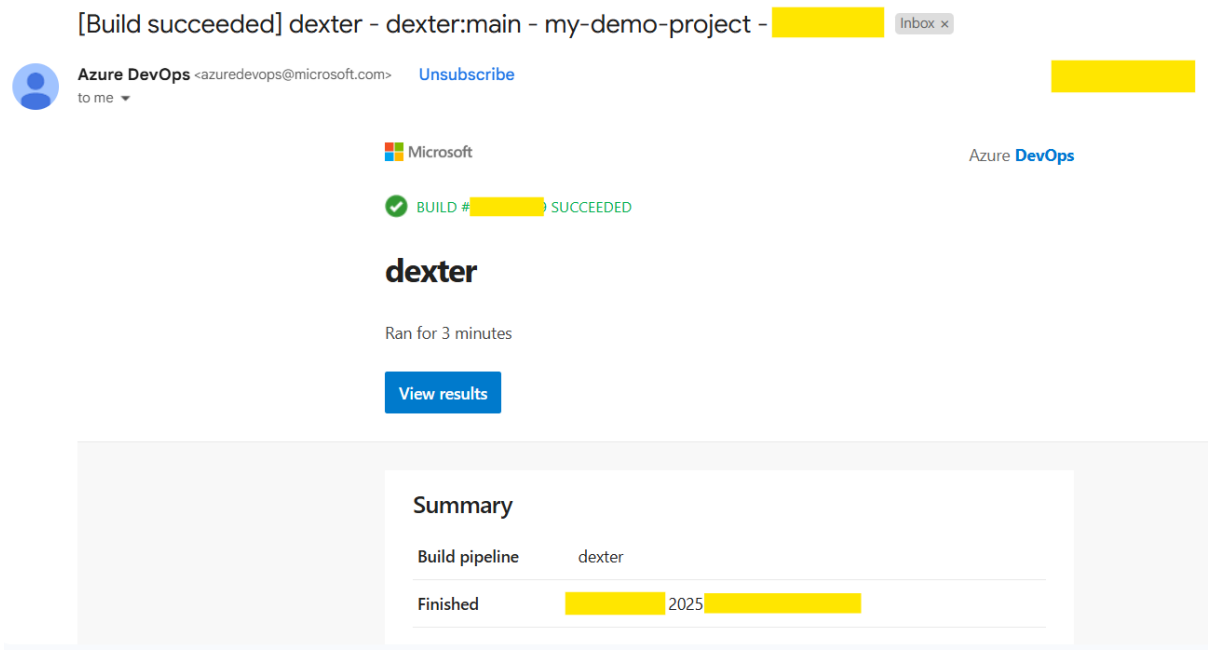


To Send Notification on Group Email ID go project Settings or Organisation Settings (For Global Notification) then got to Notifications and New Subscription and create a new Subscription with custom Email ID.

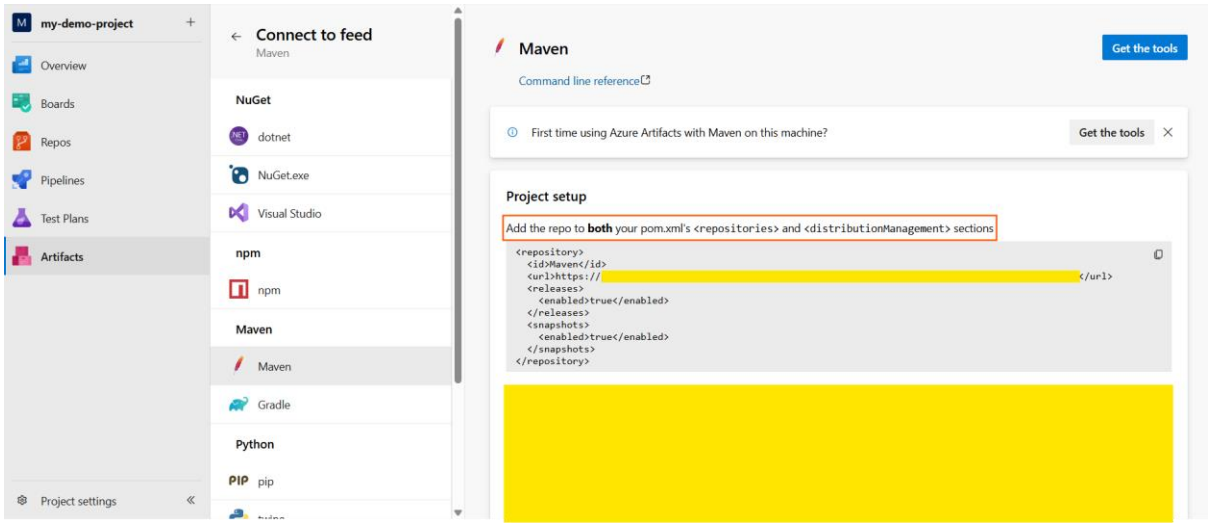


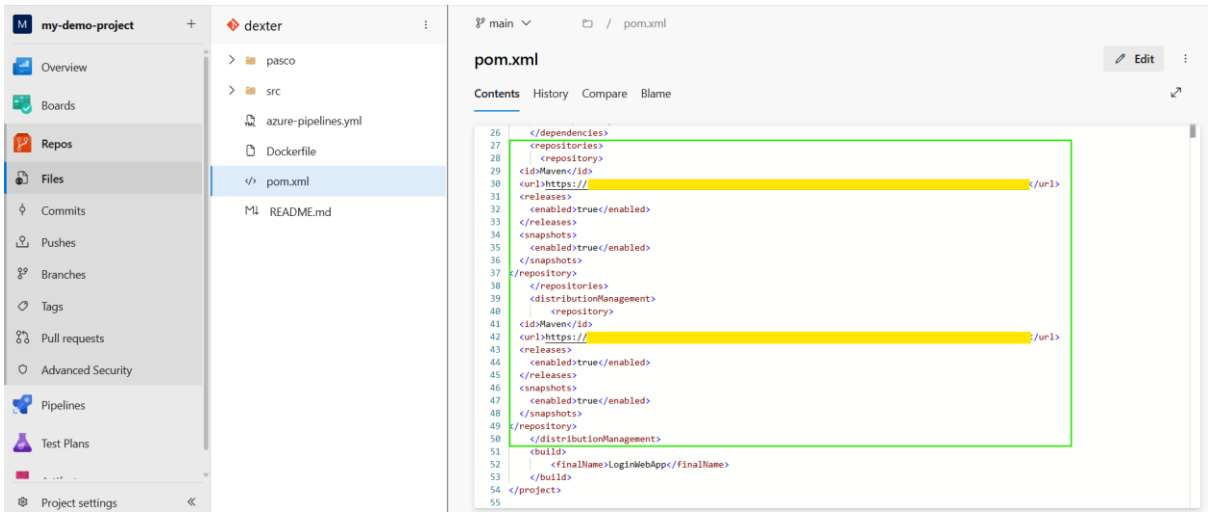

New subscription
Category
Template
Build
Code (Git)
Code (TFVC)
Pipelines
Work
Artifacts
Extension management
Release
A build completes
A build fails
Next
Cancel

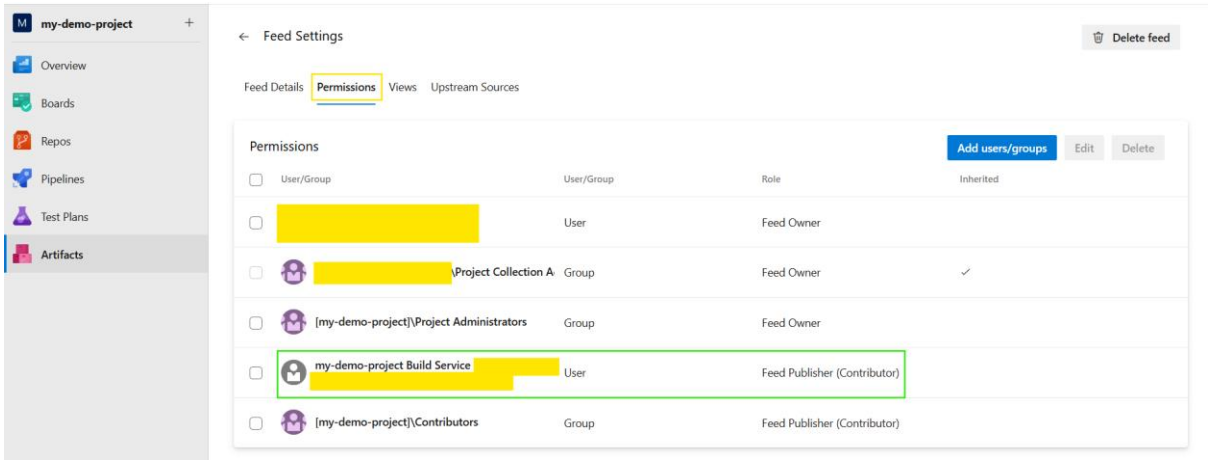

New subscription
Description
A build completes
Subscriber
my-demo-project Team
Deliver to
Custom email address
Address
[redacted]@gmail.com
Filter
Any team project
A specific team project
my-demo-project
Filter criteria
+ Add new clause
Previous
Finish
Cancel



Adding below lines to pom.xml for storing Artifacts to Azure Artifactory







The ingress rule for the final application is as shown below.

```
### kubectl create secret tls ingress-secret --key mykey.key --cert STAR_singhritesh85_com.crt -n demo
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: dexter-ingress
  namespace: demo
  annotations:
    appgw.ingress.kubernetes.io/ssl-redirect: "true"
spec:
  ingressClassName: azure-application-gateway
  tls:
#  - hosts:
#    - dexter.singhritesh85.com
  - secretName: ingress-secret
  rules:
  - host: dexter.singhritesh85.com
    http:
      paths:
      - path: /LoginWebApp
        backend:
          service:
            name: dexter-folo
            port:
              number: 80
        pathType: Prefix
```

The azure-pipelines.yml file is as shown below.

```
trigger:
- main

resources:
- repo: self

variables:
  imagePullSecret: 'devopsmelacr132827a7-auth'

pool:
  name: demo
  demands:
  - agent.name -equals demo

stages:
- stage: Build
  displayName: Build
  jobs:
  - job: Build
    displayName: Build
    steps:
    - task: SonarQubePrepare@5
      inputs:
        SonarQube: 'SonarQube'
        scannerMode: 'Other'
        extraProperties: |
          # Additional properties that will be passed to the scanner,
          # Put one key=value per line, example:
          # sonar.exclusions=**/*.bin
          sonar.projectName=thoro
          sonar.projectKey=thoro
```

```
        sonar.qualitygate.wait=true
    - task: SonarQubePublish@5
      inputs:
        pollingTimeoutSec: '300'
    - task: sonar-buildbreaker@8
      inputs:
        SonarQube: 'SonarQube'
    - task: MavenAuthenticate@0
      inputs:
        artifactsFeeds: 'Maven'
        mavenServiceConnections: 'Maven'
    - task: Maven@4
      inputs:
        mavenPomFile: 'pom.xml'
        goals: 'deploy sonar:sonar'
        publishJUnitResults: false
        javaHomeOption: 'JDKVersion'
        mavenVersionOption: 'Default'
        mavenAuthenticateFeed: false
        effectivePomSkip: false
        sonarQubeRunAnalysis: false
- stage: DockerImage
  displayName: Docker Image
  dependsOn: Build
  jobs:
  - job: DockerImage
    displayName: Docker Image
    steps:
    - checkout: none
    - task: CmdLine@2
      inputs:
```

```
      script: |
        docker system prune -f --all
        docker build -t demoimage:1.01 .
        trivy image --exit-code 0 --severity MEDIUM,HIGH demoimage:1.01
        #trivy image --exit-code 1 --severity CRITICAL demoimage:1.01
    - task: Docker@2
      inputs:
        containerRegistry: 'Docker-Registry-Connection'
        repository: 'samplewebapp'
        command: 'buildAndPush'
        Dockerfile: '**/Dockerfile'
- stage: KubernetesDeployment
  displayName: KubernetesDeployment
  dependsOn: DockerImage
  jobs:
  - deployment: KubernetesDeployment
    displayName: Kubernetes Deployment
    environment: dev
    strategy:
      runOnce:
        deploy:
          steps:
          - checkout: none
          - task: HelmDeploy@0
            inputs:
              connectionType: 'Azure Resource Manager'
              azureSubscription: 'Azure DevOps Service Connection'
              azureResourceGroup: 'aks-rg'
              kubernetesCluster: 'aks-cluster'
              namespace: 'demo'
              command: 'upgrade'
```

```
            chartType: 'FilePath'
            chartPath: '/home/demo/folo'
            releaseName: 'dexter'
            overrideValues: 'imagePullSecrets[0].name=devopsmelacr132827a7-
auth,image.repository=akscontainerregistry2405.azurecr.io/samplewebapp,image.tag=$(Build.BuildI
d),replicaCount=1,service.type=ClusterIP,service.port=80'
```

The ingress rule for grafana is as shown in the screenshot attached below.

```
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: grafana-ingress
  namespace: monitoring
  annotations:
    appgw.ingress.kubernetes.io/ssl-redirect: "true"
spec:
  ingressClassName: azure-application-gateway
  tls:
#  - hosts:
#    - dexter.singhritesh85.com
  - secretName: ingress-secret
  rules:
  - host: grafana.singhritesh85.com
    http:
      paths:
      - path: /
        backend:
          service:
            name: grafana
            port:
              number: 3000
        pathType: Prefix
```

```
[demo@devopsagent-vm ~]$ cat grafana-ingress-rule.yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: grafana-ingress
  namespace: monitoring
  annotations:
    appgw.ingress.kubernetes.io/ssl-redirect: "true"
spec:
  ingressClassName: azure-application-gateway
  tls:
#  - hosts:
#    - dexter.singhritesh85.com
  - secretName: ingress-secret
  rules:
  - host: grafana.singhritesh85.com
    http:
      paths:
      - path: /
        backend:
          service:
            name: grafana
            port:
              number: 3000
        pathType: Prefix
```

```
[demo@devopsagent-vm ~]$ cat ingress-rule.yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: dexter-ingress
  namespace: demo
  annotations:
    appgw.ingress.kubernetes.io/ssl-redirect: "true"
spec:
  ingressClassName: azure-application-gateway
  tls:
#  - hosts:
#    - dexter.singhritesh85.com
  - secretName: ingress-secret
  rules:
  - host: dexter.singhritesh85.com
    http:
      paths:
      - path: /LoginWebApp
        backend:
          service:
            name: dexter-folo
            port:
              number: 80
        pathType: Prefix
```

The finally created ingress is as shown in the screenshot attached below.

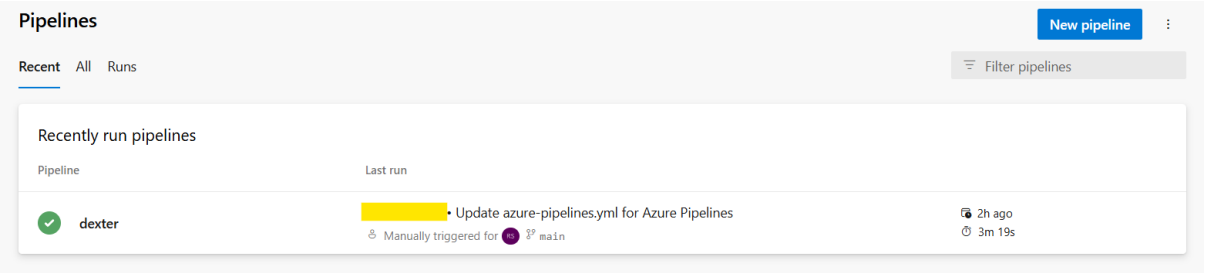
```
[demo@devopsagent-vm ~]$ kubectl apply -f ingress-rule.yaml
ingress.networking.k8s.io/dexter-ingress created
```

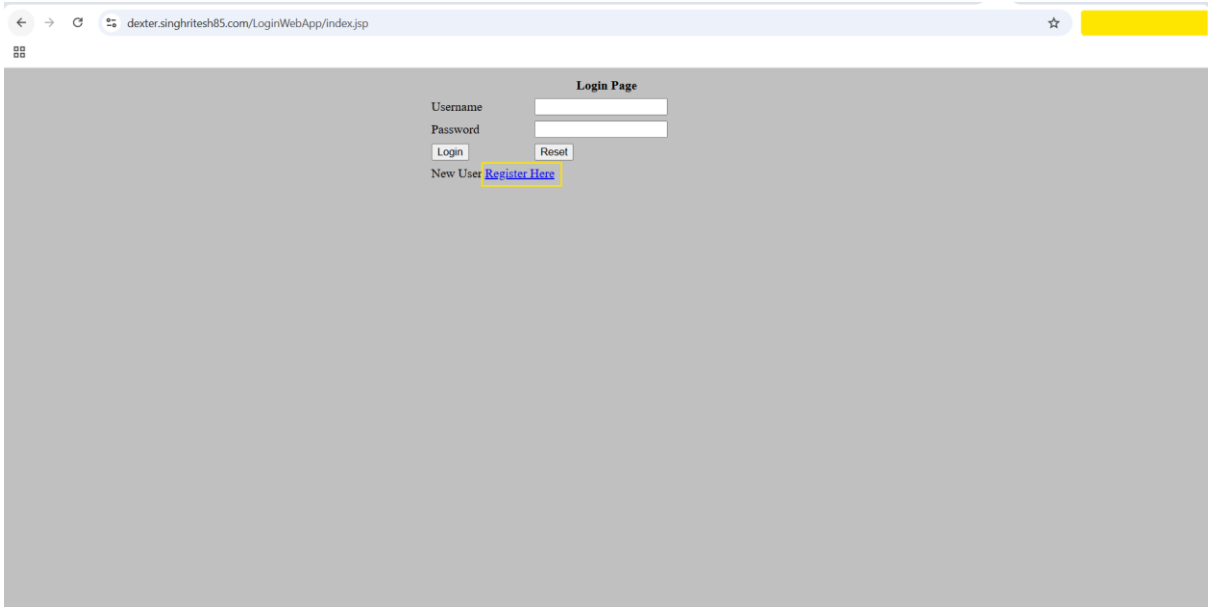
```
[demo@devopsagent-vm ~]$ kubectl get svc -n monitoring
NAME                                      TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
grafana                                   ClusterIP   10.      150    <none>        80/TCP
prometheus-alertmanager                   ClusterIP   10.       34    <none>        9093/TCP
prometheus-alertmanager-headless          ClusterIP   None            <none>        9093/TCP
prometheus-kube-state-metrics             ClusterIP   10.      195    <none>        8080/TCP
prometheus-prometheus-node-exporter       ClusterIP   10.       10    <none>        9100/TCP
prometheus-prometheus-pushgateway         ClusterIP   10.       14    <none>        9091/TCP
prometheus-server                         ClusterIP   10.     .162    <none>        80/TCP
```

```
[demo@devopsagent-vm ~]$ kubectl apply -f grafana-ingress-rule.yaml
ingress.networking.k8s.io/grafana-ingress created
```

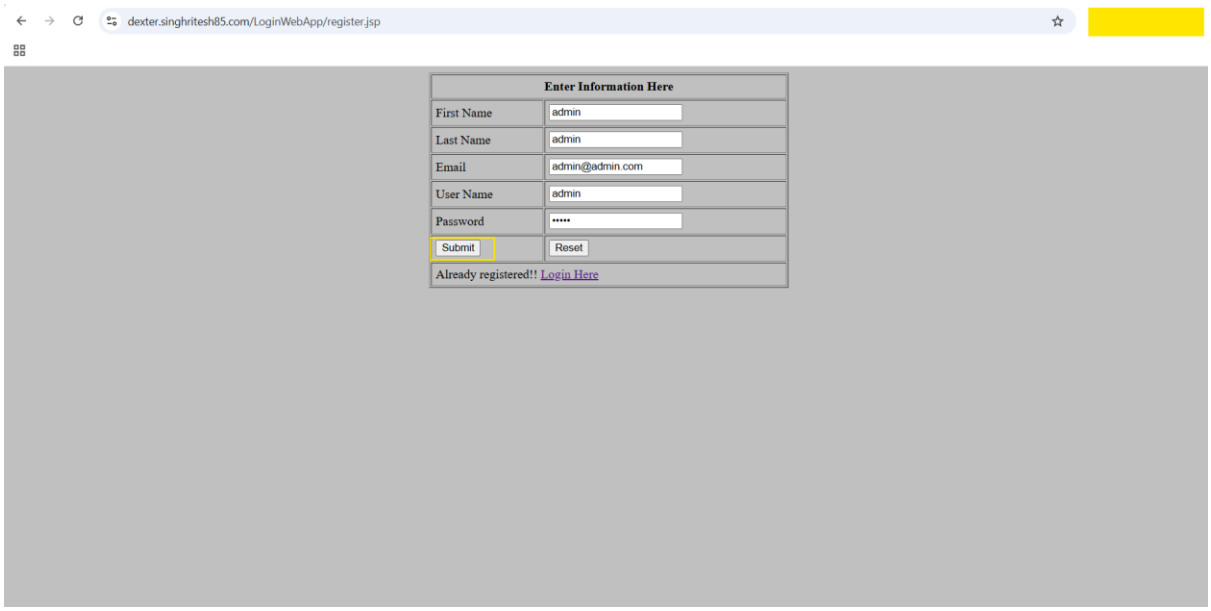
```
[demo@devopsagent-vm ~]$ kubectl get ing -A
NAMESPACE    NAME              CLASS                       HOSTS                       ADDRESS          PORTS     AGE
demo         dexter-ingress    azure-application-gateway   dexter.singhritesh85.com    135.     .211    80, 443   38m
monitoring   grafana-ingress   azure-application-gateway   grafana.singhritesh85.com   135.     .211    80, 443   74s
```

Azure DevOps Pipeline had been executed successfully as shown in the screenshot attached below and I was able to access the application.

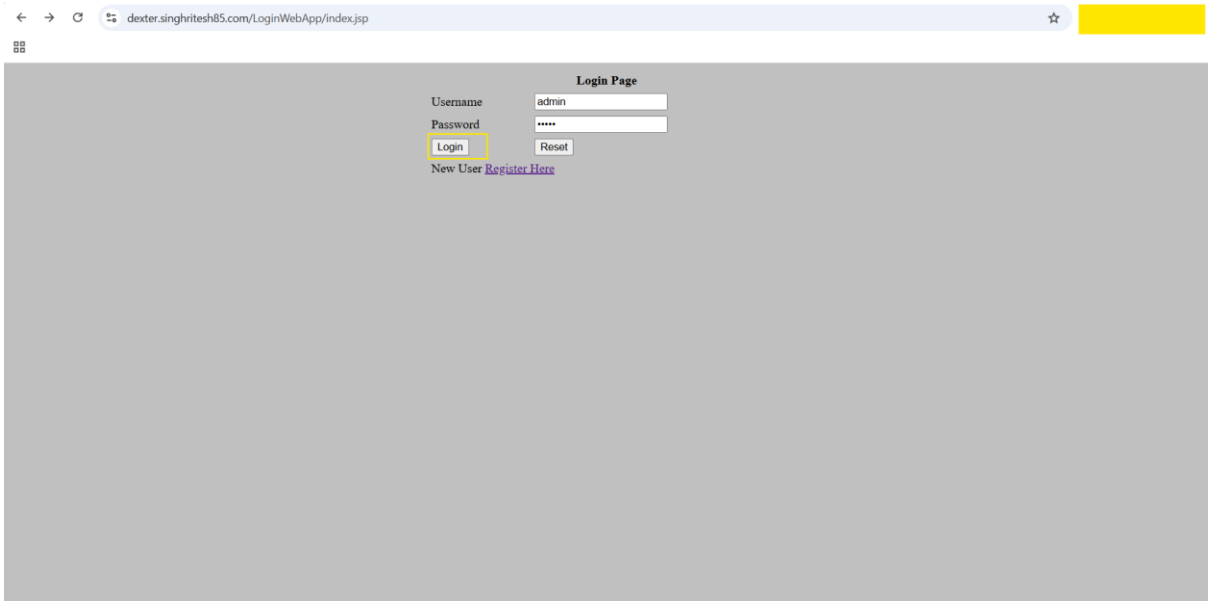


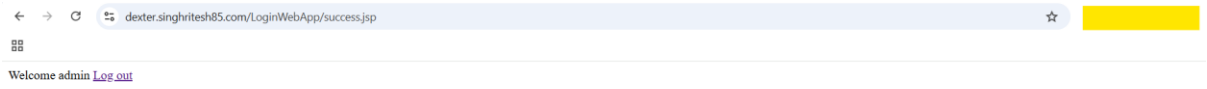


I registered with a user and logged in with the same user as shown in the screenshot attached below.



Then I logged in with the created user as shown in the screenshot attached below.

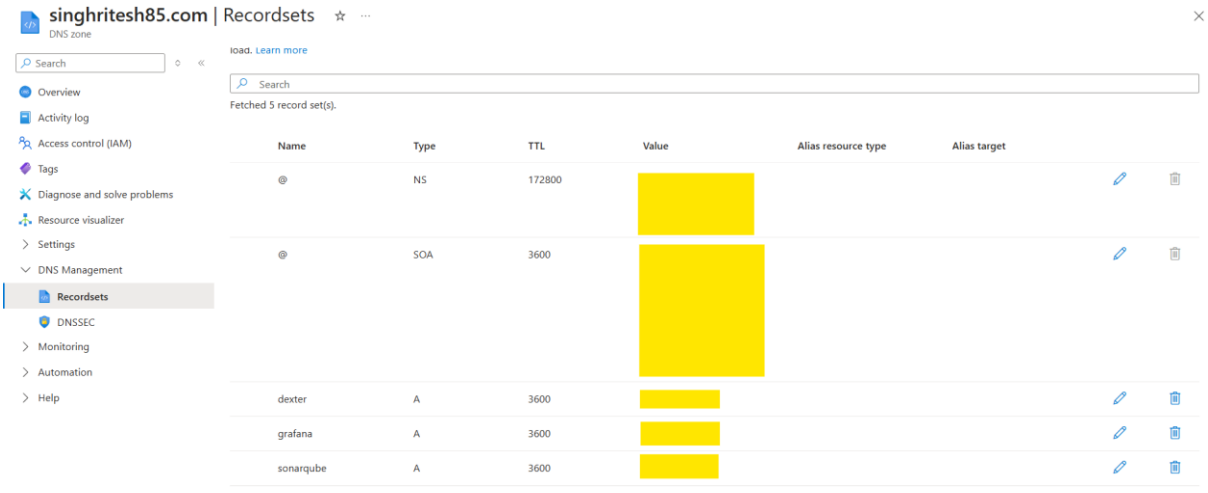




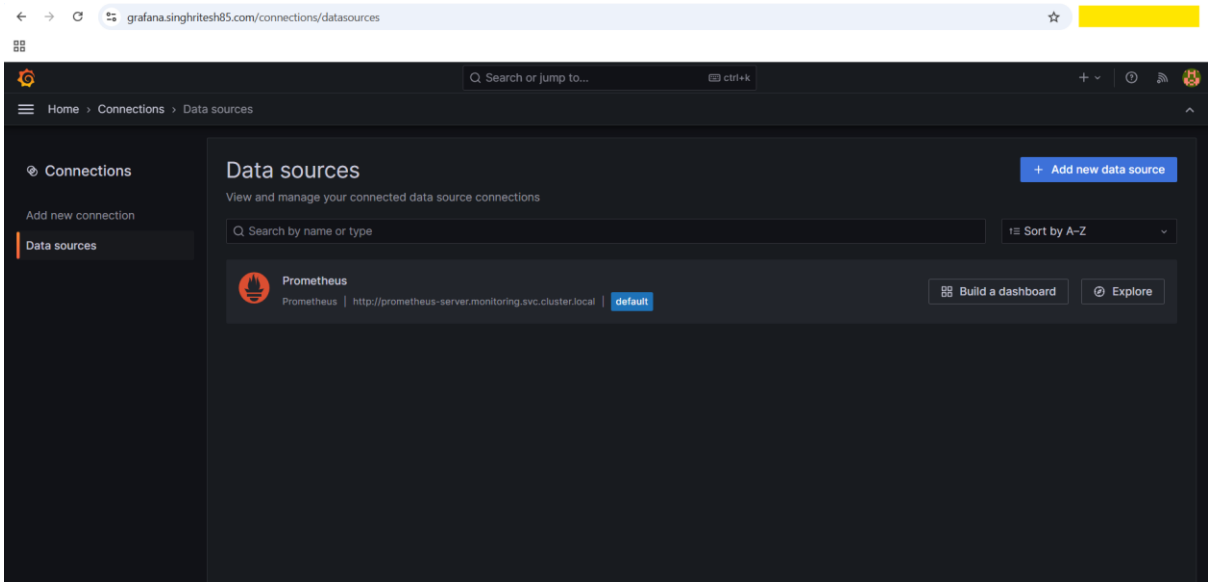
Then I checked the entry in the MySQL database table and found as shown below.

```
[root@devopsagent-vm ~]# mysql -h dbinstance-2.c          .us-east-2.rds.amazonaws.com -u admin --password
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 123
Server version: 5.7.44 Please upgrade to 8.0 or opt-in to the paid RDS Extended Support service before 5.7 reaches end of standard support on 29 February, 202
4: https://a.co/hQqiIn0

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use jwt;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+---------------+
| Tables_in_jwt |
+---------------+
| USER          |
+---------------+
1 row in set (0.02 sec)

mysql> select * from USER;
+----+------------+-----------+-----------------+----------+----------+------------+
| id | first_name | last_name | email           | username | password | regdate    |
+----+------------+-----------+-----------------+----------+----------+------------+
|  1 | admin      | admin     | admin@admin.com | admin    |          | 2025-03-20 |
+----+------------+-----------+-----------------+----------+----------+------------+
1 row in set (0.02 sec)

mysql>
```

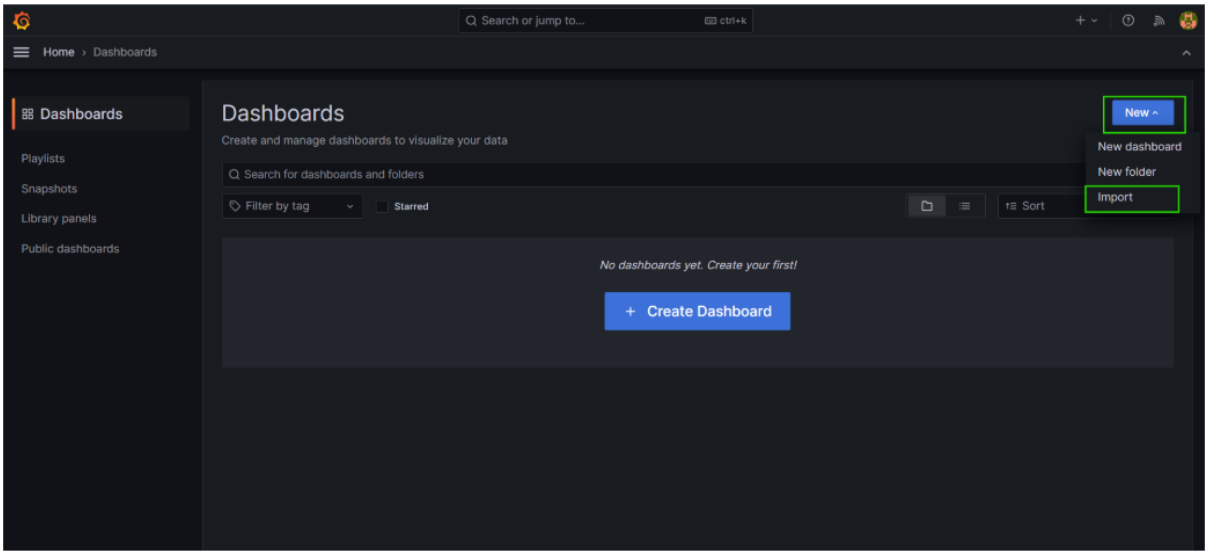
The entry of record sets in Azure DNS Zone is as shown in the screenshot attached below.

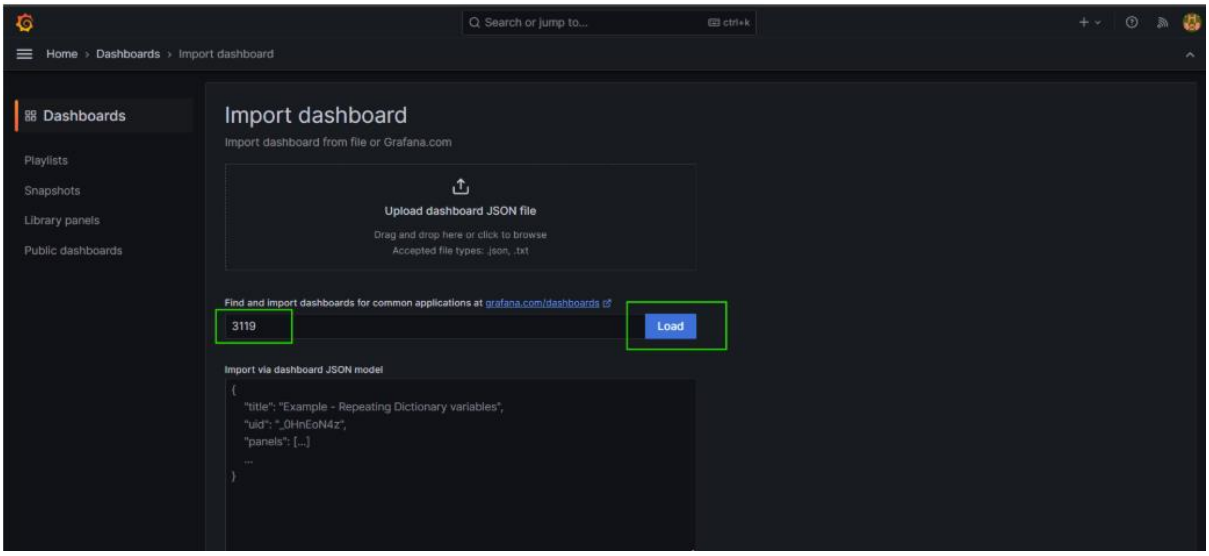


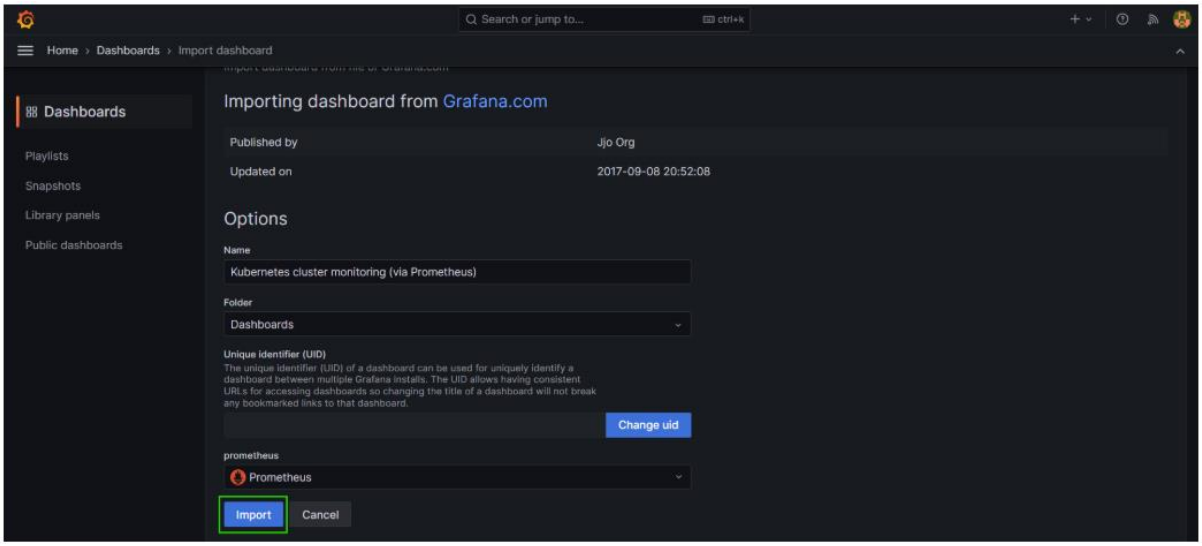
Prometheus and Grafana Configuration

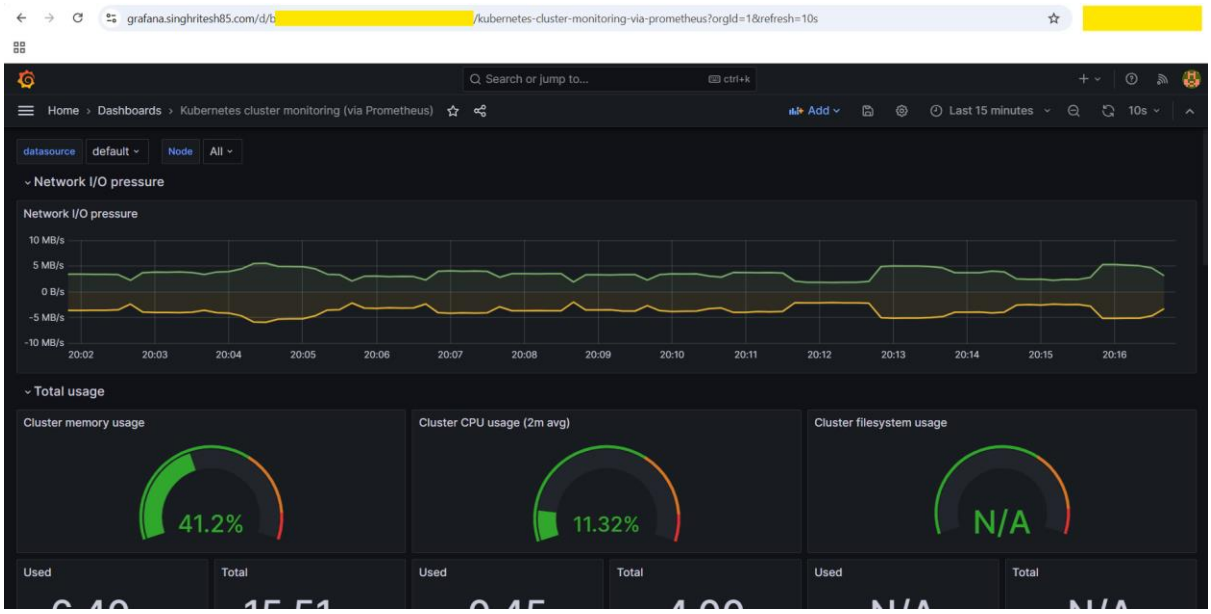


Dashboard has been imported using the ID **3119** as shown in the screenshot below




Search or jump to...
Home › Dashboards › Import dashboard
Dashboards
Playlists
Snapshots
Library panels
Public dashboards
Import dashboard
Import dashboard from file or Grafana.com
Upload dashboard JSON file
Drag and drop here or click to browse
Accepted file types: .json, .txt
Find and import dashboards for common applications at grafana.com/dashboards
3119
Load
Import via dashboard JSON model
{
"title": "Example - Repeating Dictionary variables",
"uid": "_0HnEoN4z",
"panels": [...]
...
}


Search or jump to...
Home › Dashboards › Import dashboard
Dashboards
Playlists
Snapshots
Library panels
Public dashboards
Importing dashboard from Grafana.com
Published by
Jjo Org
Updated on
2017-09-08 20:52:08
Options
Name
Kubernetes cluster monitoring (via Prometheus)
Folder
Dashboards
Unique identifier (UID)
The unique identifier (UID) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The UID allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.
Change uid
prometheus
Prometheus
Import
Cancel


grafana.singhritesh85.com/d/b
/kubernetes-cluster-monitoring-via-prometheus?orgId=1&refresh=10s
Search or jump to...
Home › Dashboards › Kubernetes cluster monitoring (via Prometheus)
Add
Last 15 minutes
10s
datasource
default
Node
All
Network I/O pressure
Network I/O pressure
10 MB/s
5 MB/s
0 B/s
-5 MB/s
-10 MB/s
20:02
20:03
20:04
20:05
20:06
20:07
20:08
20:09
20:10
20:11
20:12
20:13
20:14
20:15
20:16
Total usage
Cluster memory usage
41.2%
Cluster CPU usage (2m avg)
11.32%
Cluster filesystem usage
N/A
Used
Total
Used
Total
Used
Total

GitHub Repo for Source Code: -

https://github.com/singhritesh85/aws-rds-java.git

GitHub Repo: -

https://github.com/singhritesh85/DevOps-Project-2-Tier-WebAppDeployment-using-AWSandAzure.git

References

1. **https://ashok198510.hashnode.dev/cloud-native-two-tier-application-deployment-with-eks-tomcat-and-rds-in-aws**
2. **https://github.com/Ashoksana/aws-rds-java**
3. **https://medium.com/@abiolamajekodunmi2011/implementing-secure-and-observant-3-tier-deployments-on-aws-using-terraform-eks-jenkins-ea2572d239e1**