Practical DevOps – CI/CD Pipeline and deploying applications on Azure

1. Table of Contents

I. Introduction *1*

II. Prerequisites *2*

III. Git Repositories *2*

1. Infrastructure as code 2
2. Micro-services application 2

IV. Provisioning Infrastructure by Terraform *2*

1. Provisioning 2

V. Deploy AKS *3*

1. Build Docker Images 3
2. Push Docker Images to ACR 3
3. Deploy to AKS 4

VI. CI/CD On Azure Pipeline *5*

1. Service Connection 5
2. Setup Azure Pipeline 6
3. Verify Pipeline 7

VII. Prometheus and Grafana *8*

1. Installation 8
2. Monitoring 9

I. Introduction

1. Provision Azure resources by terraform.
2. Set up Azure DevOps pipeline for CI/CD.
3. Set up monitor AKS Resources with Prometheus and Grafana.

II. Prerequisites

- Have an Azure account.
- Azure CLI installed.
- Azure Subscription.
- Terraform installed.
- Kubectl installed.
- Helm installled.

III. Git Repositories

1. Infrastructure as code

   sd2793_azure_infrastructure

2. Micro-services application

   sd2793_msa

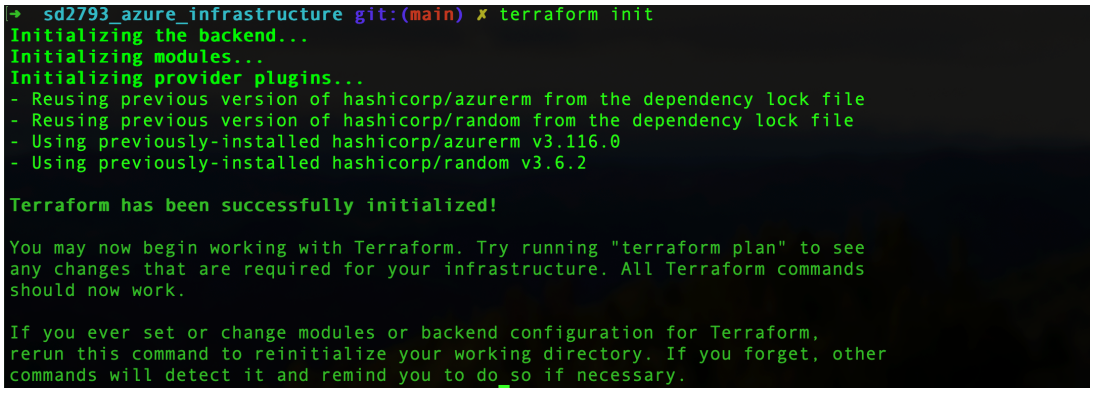
IV. Provisioning Infrastructure by Terraform

1. Provisioning

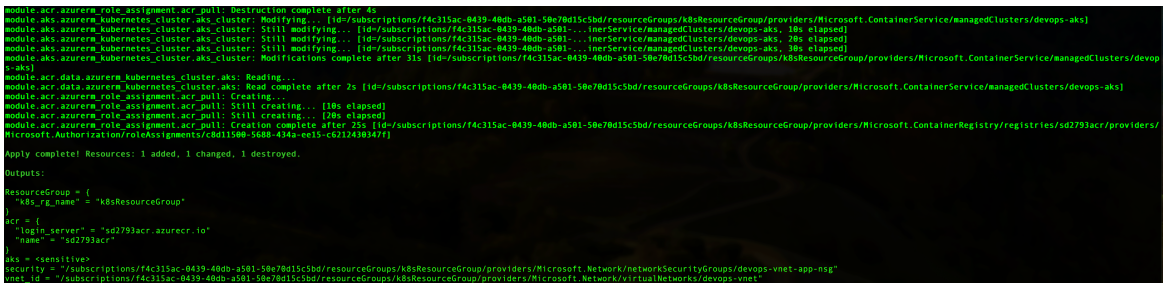
1. Git clone from repository infrastructure as code

2. Go to folder **sd2793_azure_infrastructure**

2.1. Initialize terraform terraform init



2.2. Provisioning infra terraform apply -auto-approve -var-file "terraform.tfvars"

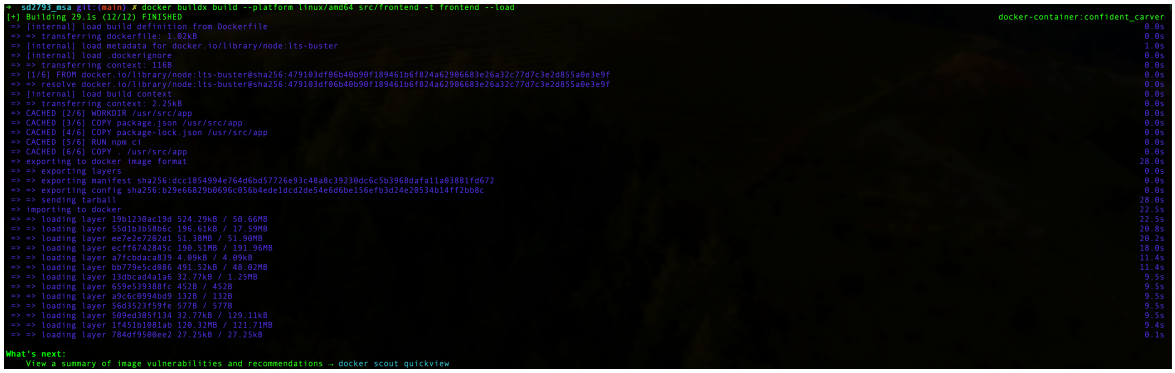


V. Deploy AKS

1. Build Docker Images

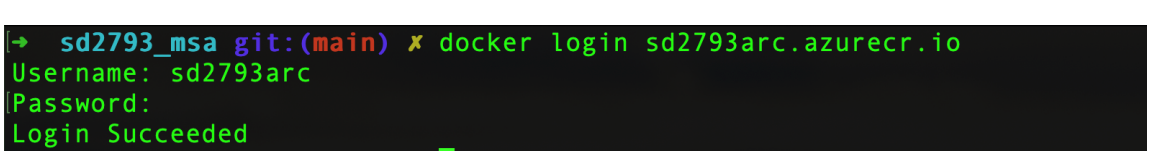
1. backend

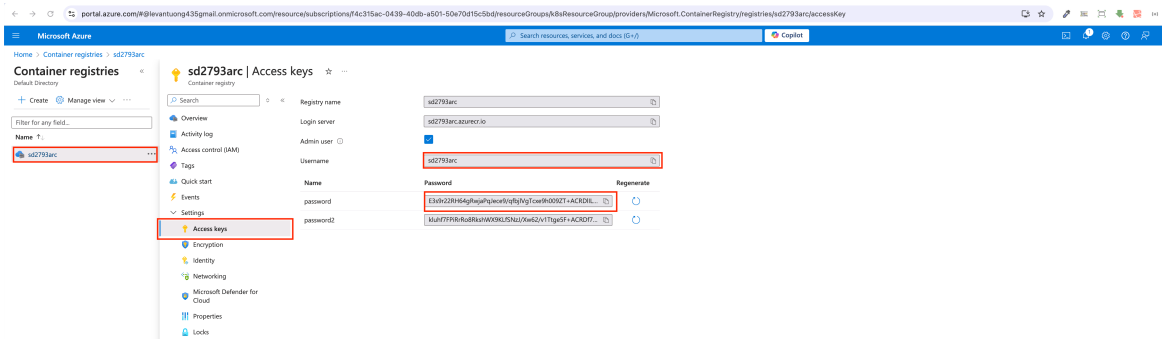
2. frontend



2. Push Docker Images to ACR

1. Docker Login

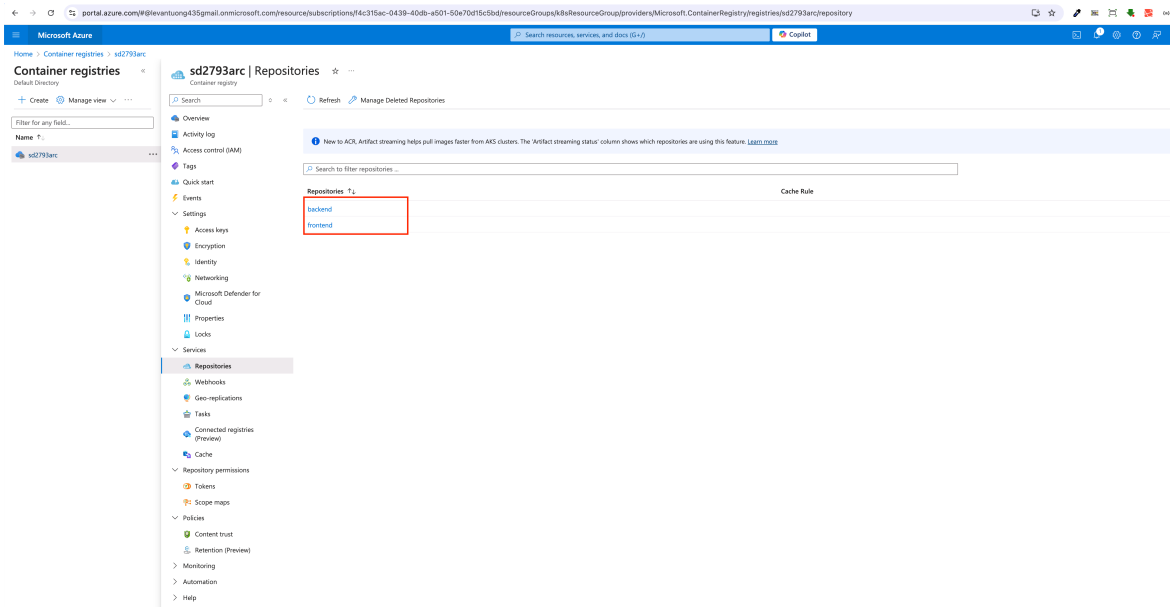




2. Push Docker Images

```
➜  sd2793_msa git:(main) ✗ docker tag backend sd2793arc.azurecr.io/backend:latest
➜  sd2793_msa git:(main) ✗ docker push sd2793arc.azurecr.io/backend:latest
The push refers to repository [sd2793arc.azurecr.io/backend]
83027e85d608: Pushed
2e3071d5ee5f: Pushed
6caec2507d26: Pushed
0ce68de93eb7: Pushed
aaf1354df6e5: Pushed
b1231e04bb57: Pushed
2a7a5ad26516: Pushed
d385e3f24317: Pushed
6d4d4f04857e: Pushed
ec8cbb2796ae: Pushed
latest: digest: sha256:b36aab2d341f900fa254782c7d5c16eae149106ff542fbba9adcf8fa5858129b size: 2410
➜  sd2793_msa git:(main) ✗ docker tag frontend sd2793arc.azurecr.io/frontend:latest
➜  sd2793_msa git:(main) ✗ docker push sd2793arc.azurecr.io/frontend:latest
The push refers to repository [sd2793arc.azurecr.io/frontend]
33011ac9386a: Pushed
234d39c46d3e: Pushed
d94641e89a45: Pushed
200e14039f50: Pushed
5952400adedf: Pushed
ea5a7252901b: Pushed
ab16d908217e: Pushed
9f17557134ba: Pushed
e5af13f8f882: Pushed
5df178db4ff8: Pushed
d026c2b278de: Pushed
33fd23f16700: Pushed
70b9717deb5e: Pushed
latest: digest: sha256:0b241401c62425addf27f34a5ce70436e55ff6229d34d83514f326e35e799db4 size: 3050
```

3. Verify image on ACR



3. Deploy to AKS

1. Config

```
➜  sd2793_msa git:(main) ✗ az aks get-credentials --resource-group k8sResourceGroup --name devops-aks --overwrite-existing
Merged "devops-aks" as current context in /Users/tuonglv/.kube/config
```

2. Deploy AKS

```
➜  sd2793_msa git:(main) ✗ kubectl apply -f k8s/azure/mongodb.yaml
storageclass.storage.k8s.io/mongo-sc created
service/mongo created
configmap/mongo-config created
secret/mongo-secrets created
statefulset.apps/mongo created
➜  sd2793_msa git:(main) ✗ kubectl apply -f k8s/azure/backend.yaml
service/backend created
deployment.apps/backend created
➜  sd2793_msa git:(main) ✗ kubectl apply -f k8s/azure/frontend.yaml
service/frontend created
deployment.apps/frontend created
```

3. Verify services

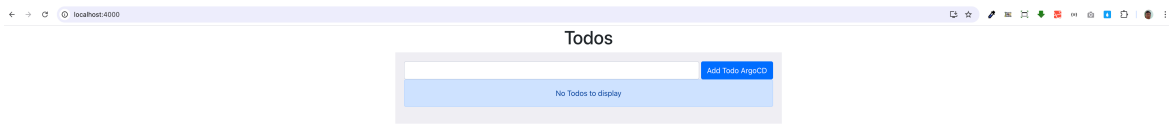
```
➜  sd2793_msa git:(main) ✗ kubectl get services
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)     AGE
backend      ClusterIP   10.10.2.28      <none>        3000/TCP    65s
frontend     ClusterIP   10.10.102.140   <none>        3000/TCP    53s
kubernetes   ClusterIP   10.10.0.1       <none>        443/TCP     114m
mongo        ClusterIP   10.10.232.208   <none>        27017/TCP   88s
```

4. Verify Pods

```
➜  sd2793_msa git:(main) ✗ kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
backend-6d47975dd7-pk2rt    1/1     Running   0          18m
frontend-866f749f4d-ngwt2   1/1     Running   0          34s
mongo-0                     1/1     Running   0          140m
```

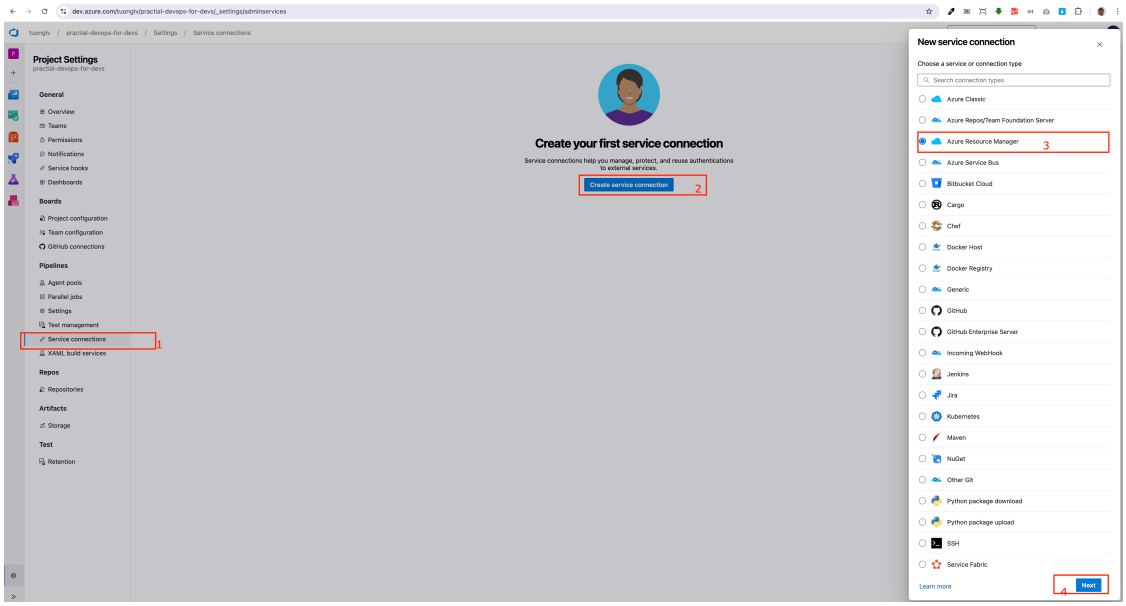
5. Verify Web UI

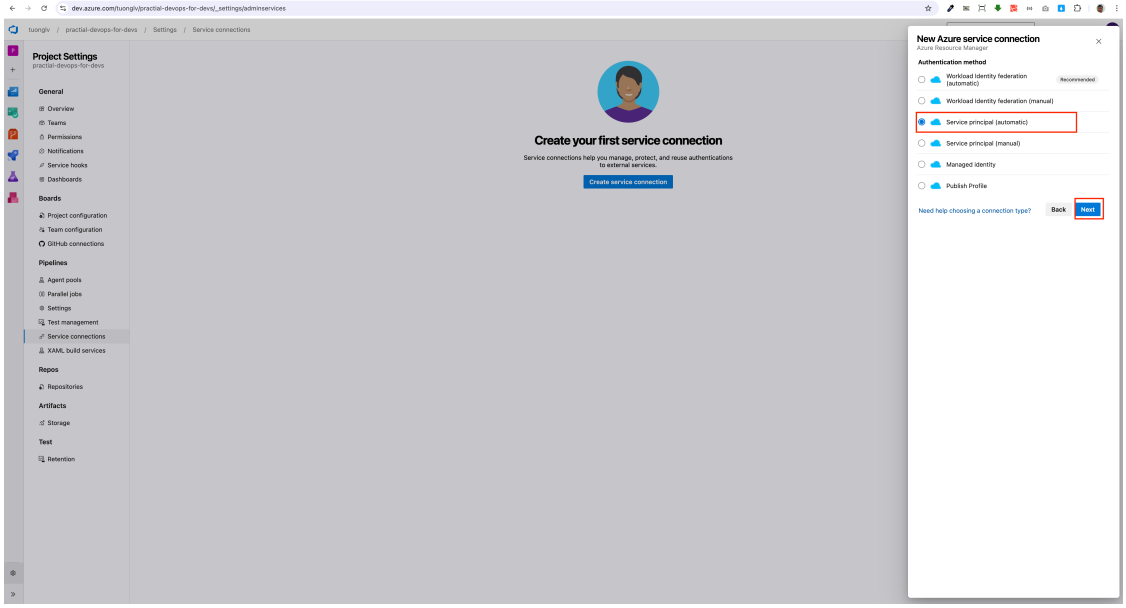
```
➜  sd2793_msa git:(main) ✗ kubectl port-forward svc/frontend 4000:3000
Forwarding from 127.0.0.1:4000 -> 3000
Forwarding from [::1]:4000 -> 3000
Handling connection for 4000
Handling connection for 4000
Handling connection for 4000
Handling connection for 4000
Handling connection for 4000
```

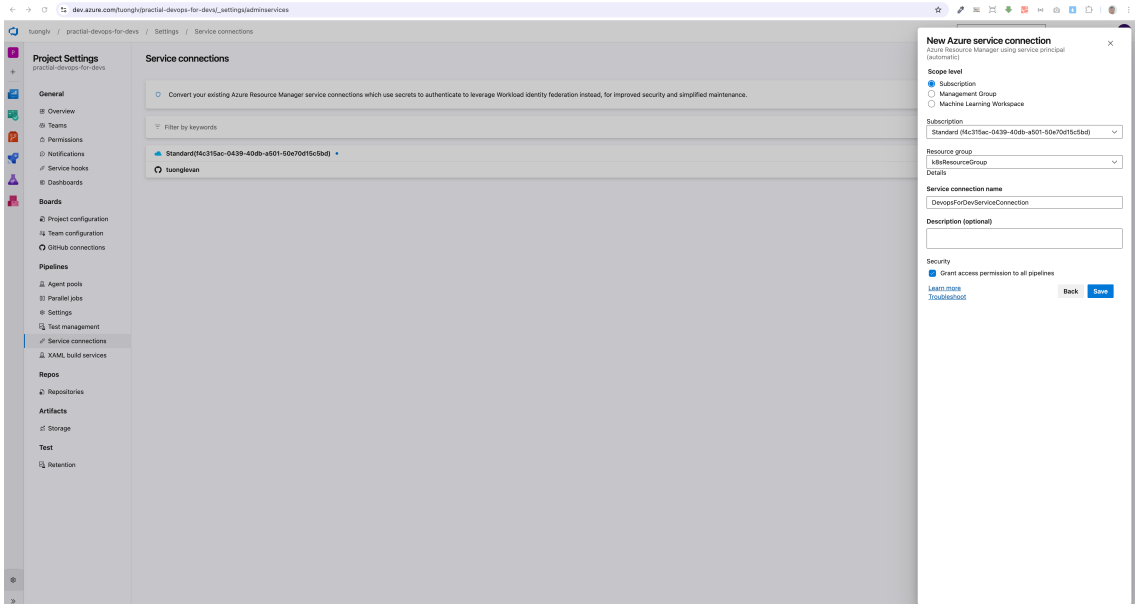


VI. CI/CD On Azure Pipeline

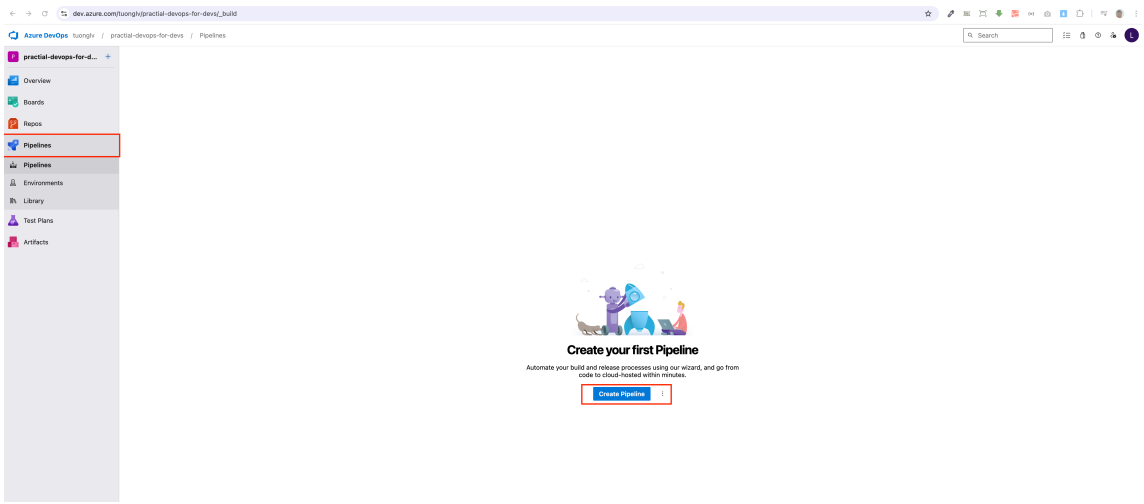
1. Service Connection

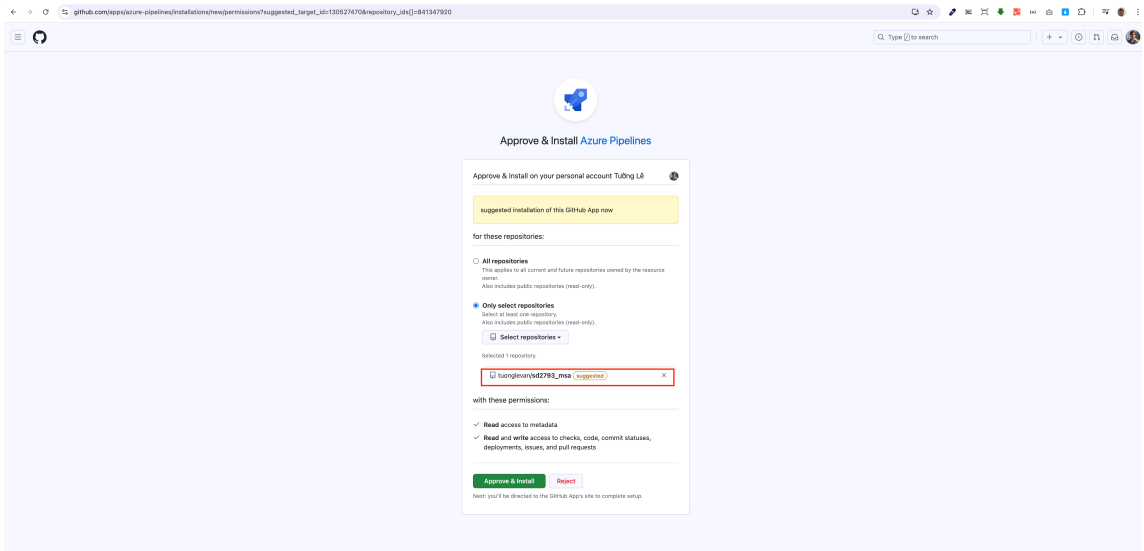


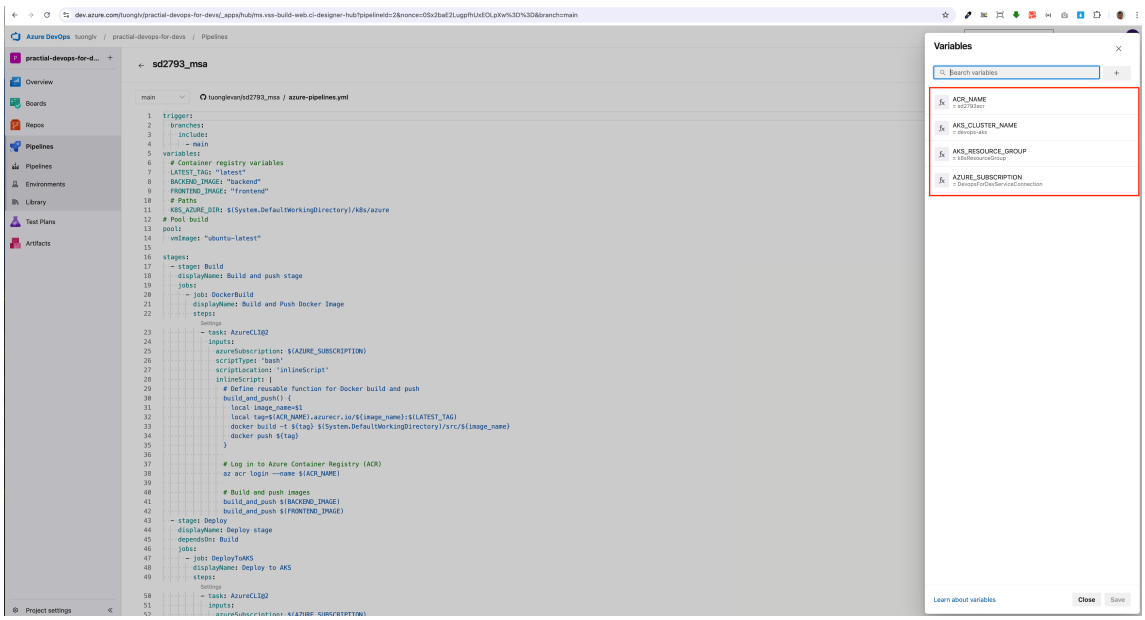




2. Setup Azure Pipeline

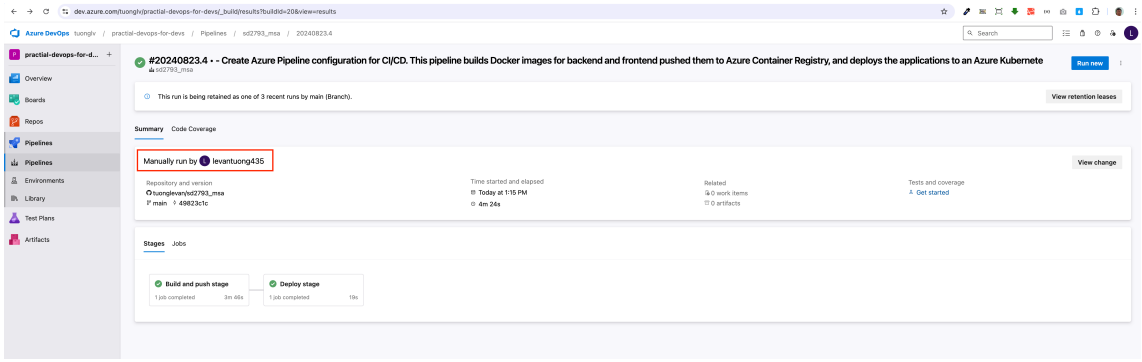




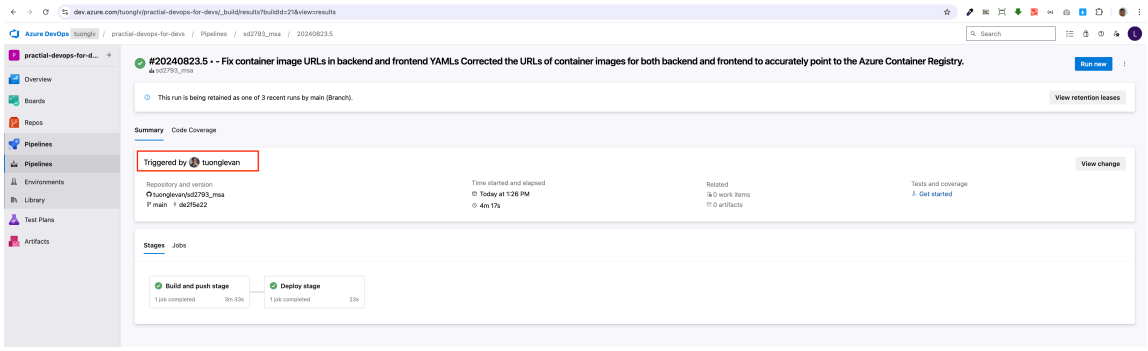


3. Verify Pipeline

1. Manually run Pipeline > Click **Run Pipeline** button



2. Pipeline auto Trigger



3. Verify deploy to AKS

```
➜  ~ kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
backend-697b7c7b5-dt5vc     1/1     Running   0          34s
frontend-647f5cb58b-6gglt   1/1     Running   0          33s
mongo-0                     1/1     Running   0          11m
```

VII. Prometheus and Grafana

1. Installation

Following readme for Setup Prometheus and Grafana in **sd2793_msa** repository

1. Install Prometheus and Grafana

```
➜  sd2793_azure_infrastructure git:(main) helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo add grafana https://grafana.github.io/helm-charts
helm repo update
"prometheus-community" already exists with the same configuration, skipping
"grafana" already exists with the same configuration, skipping
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "grafana" chart repository
...Successfully got an update from the "stable" chart repository
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. ⎈Happy Helming!⎈
```

```
➜  sd2793_azure_infrastructure git:(main) helm install prometheus prometheus-community/kube-prometheus-stack
NAME: prometheus
LAST DEPLOYED: Fri Aug 23 15:49:08 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --namespace default get pods -l "release=prometheus"

Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configure Alertmanager and Prometheus instances using the Operator.
```

2. Monitoring

1. Port Forward and Access UI

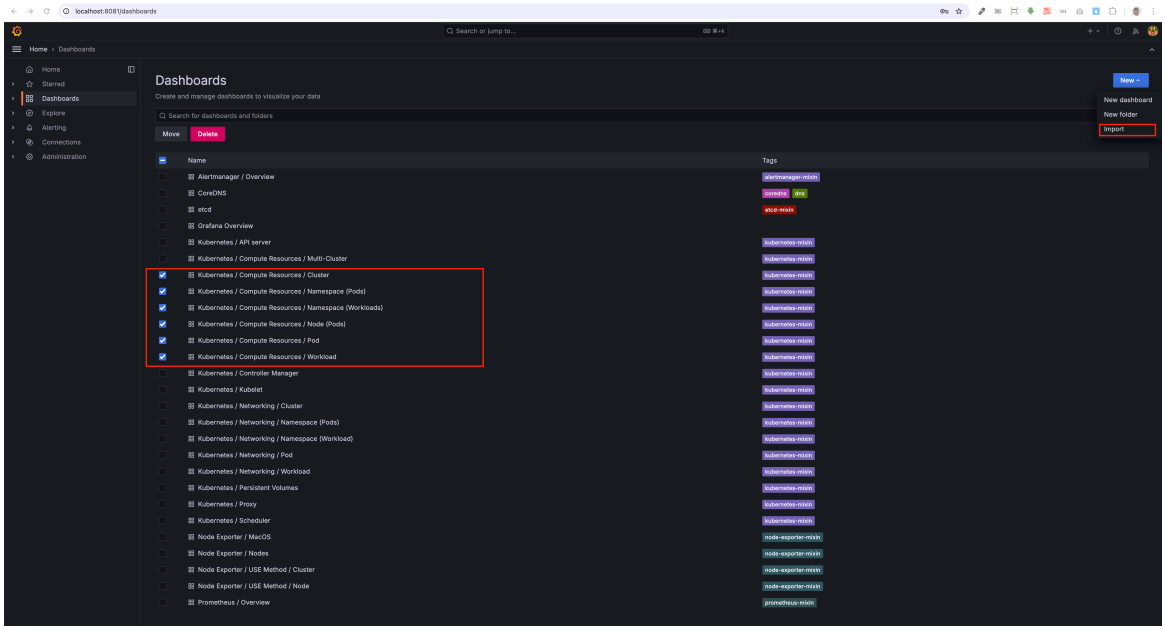
```
➜  sd2793_azure_infrastructure git:(main) kubectl port-forward service/prometheus-grafana 8081:80
Forwarding from 127.0.0.1:8081 -> 3000
Forwarding from [::1]:8081 -> 3000
```

2. Open the Browser and access to http://localhost:8081

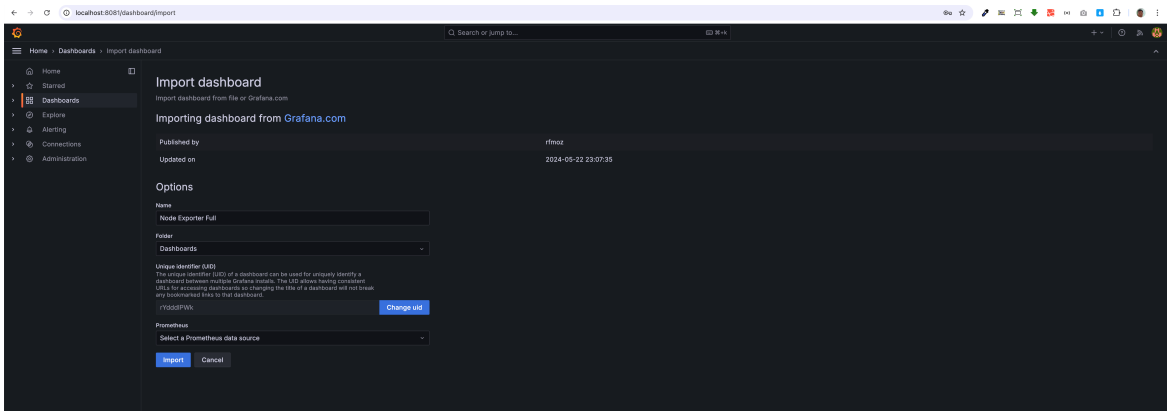
    2.1. Get password admin

```
➜  sd2793_msa git:(main) ✗ kubectl get secret --namespace default prometheus-grafana -o jsonpath="{.data.admin-password}" | base64 --decode
echo
prom-operator
```

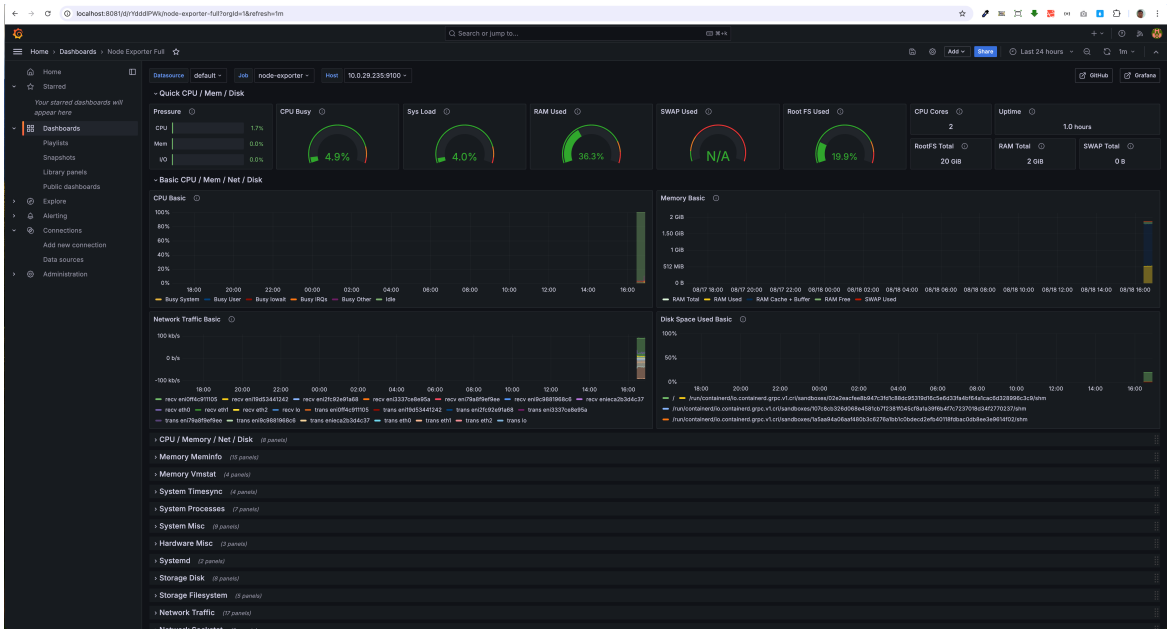
3. Import Metrics to Dashboard



Dashboard ID: 1860 Node Exporter Full



4. Monitoring in the Dashboard UI.


localhost:8081/d/Y6dddPWvk/node-exporter-full?orgId=1&refresh=1m
Home › Dashboards › Node Exporter Full
Datasource default
Job node-exporter
Host 10.0.29.235:9100
Quick CPU / Mem / Disk
Pressure
CPU Busy
4.9%
Sys Load
4.0%
RAM Used
36.3%
SWAP Used
N/A
Root FS Used
19.9%
CPU Cores
2
Uptime
1.0 hours
RootFS Total
20 GiB
RAM Total
2 GiB
SWAP Total
0 B
Basic CPU / Mem / Net / Disk
CPU Basic
Memory Basic
Network Traffic Basic
Disk Space Used Basic
CPU / Memory / Net / Disk (8 panels)
Memory Meminfo (15 panels)
Memory Vmstat (4 panels)
System Timesync (4 panels)
System Processes (7 panels)
System Misc (9 panels)
Hardware Misc (3 panels)
Systemd (2 panels)
Storage Disk (8 panels)
Storage Filesystem (5 panels)
Network Traffic (17 panels)