

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

## 1. Table of Contents

<i>I. Introduction</i> .....	1
<i>II. Prerequisites</i> .....	2
<i>III. Git Repositories</i> .....	2
1.     Infrastructure as code .....	2
2.     Micro-services application .....	2
<i>IV. Provisioning Infrastructure by Terraform</i> .....	2
1. Provisioning .....	2
<i>V. Deploy AKS</i> .....	3
1. Build Docker Images .....	3
2. Push Docker Images to ACR .....	3
3. Deploy to AKS .....	4
<i>VI. CI/CD On Azure Pipeline</i> .....	5
1. Service Connection .....	5
2. Setup Azure Pipeline .....	6
3. Verify Pipeline .....	7
<i>VII. Prometheus and Grafana</i> .....	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 installed.

## 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**

```
→ sd2793_azure_infrastructure git:(main) ✘ terraform init
Initializing the backend...
Initializing modules...
Initializing provider plugins...
  - Reusing previous version of hashicorp/azurerm from the dependency lock file
  - Reusing previous version of hashicorp/random from the dependency lock file
  - Using previously-installed hashicorp/azurerm v3.116.0
  - Using previously-installed hashicorp/random v3.6.2

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

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

```

[module.acr.azurerm.role_assignment.acr.pull] Destruction complete after 4s
[module.aks.azurerm.kubernetes.cluster.aks.cluster] Modifying... [id=subscriptions/f4c315ac-0439-40db-a501-50e70d15c5bd/resourceGroups/k8sResourceGroup/providers/Microsoft.ContainerService/managedClusters/devops-aks]
[module.aks.azurerm.kubernetes.cluster.aks.cluster] Still modifying... [id=subscriptions/f4c315ac-0439-40db-a501-50e70d15c5bd/resourceGroups/k8sResourceGroup/providers/Microsoft.ContainerService/managedClusters/devops-aks] 10s elapsed
[module.aks.azurerm.kubernetes.cluster.aks.cluster] Modifying... [id=subscriptions/f4c315ac-0439-40db-a501-50e70d15c5bd/resourceGroups/k8sResourceGroup/providers/Microsoft.ContainerService/managedClusters/devops-aks] 30s elapsed
[module.aks.azurerm.kubernetes.cluster.aks.cluster] Modifications complete after 31s [id=subscriptions/f4c315ac-0439-40db-a501-50e70d15c5bd/resourceGroups/k8sResourceGroup/providers/Microsoft.ContainerService/managedClusters/devops-aks]
[module.acr.data.azureurm.kubernetes.cluster.aks] Reading...
[module.acr.data.azureurm.kubernetes.cluster.aks] Read complete after 2s [id=subscriptions/f4c315ac-0439-40db-a501-50e70d15c5bd/resourceGroups/k8sResourceGroup/providers/Microsoft.ContainerService/managedClusters/devops-aks]
[module.acr.azurerm.role_assignment.acr.pull] Creating...
[module.acr.azurerm.role_assignment.acr.pull] Still creating... [id=subscriptions/f4c315ac-0439-40db-a501-50e70d15c5bd/resourceGroups/k8sResourceGroup/providers/Microsoft.ContainerService/managedClusters/devops-aks] 10s elapsed
[module.acr.azurerm.role_assignment.acr.pull] Still creating... [id=subscriptions/f4c315ac-0439-40db-a501-50e70d15c5bd/resourceGroups/k8sResourceGroup/providers/Microsoft.ContainerService/managedClusters/devops-aks] 30s elapsed
[module.acr.azurerm.role_assignment.acr.pull] Creation complete after 25s [id=subscriptions/f4c315ac-0439-40db-a501-50e70d15c5bd/resourceGroups/k8sResourceGroup/providers/Microsoft.ContainerRegistry/registries/sd2793arc/providers/Microsoft.Authorization/roleAssignments/cdd159e-368d-e515-c6272430347f]

Apply complete! Resources: 1 added, 1 changed, 1 destroyed.

Outputs:
ResourceGroup = {
  "k8s_rg_name" = "k8sResourceGroup"
}
acr = [
  {
    "login_server" = "sd2793arc.azurecr.io"
    "name" = "sd2793arc"
  }
]
aks = > sensitive>
security = >subscriptions/f4c315ac-0439-40db-a501-50e70d15c5bd/resourceGroups/k8sResourceGroup/providers/Microsoft.Network/networkSecurityGroups/devops-vnet-app-nsg"
unit_id = >subscriptions/f4c315ac-0439-40db-a501-50e70d15c5bd/resourceGroups/k8sResourceGroup/providers/Microsoft.Network/virtualNetworks/devops-vnet

```

## V. Deploy AKS

### 1. Build Docker Images

#### 1. backend

```

[+] 50/93 msx git:(main) ✘ docker build --platform linux/amd64 src/backend -t backend --load
[+] Building 7.0s (13/13) FINISHED
-> [internal] load build context from Dockerfile
--> transferring dockerfile: 677B
-> [internal] load metadata for docker.io/library/node:ls-buster-slim
--> [internal] resolve docker.io/library/node:ls-buster-slim@sha256:b5c1485662c4308:86a3a08259a34f0071474ad8a879ccbd4c39bbae55b0030
-> [internal] load dockerignore
--> transferring context: 101B
-> [internal] resolve docker.io/library/node:ls-buster-slim@sha256:b5c1485662c4308:86a3a08259a34f0071474ad8a879ccbd4c39bbae55b0030
-> [internal] load build context
--> CACHER [7/6] WORKDIR /src/src/app
--> CACHED [3/6] COPY package.json ./src/src/app/package.json
--> CACHED [3/6] COPY package-lock.json ./src/src/app/package-lock.json
--> CACHER [5/6] RUN npm ci
--> CACHED [4/6] COPY ./src/src/app
--> exporting manifest sha256:7a44ae061598ada5dd2843b04d1086d5202990451a978d7edc35114
--> exporting config sha256:c784a55262157e575e1498bf7acc056e0d101526559e0b543183d75171f
--> sending manifest
--> sending config
--> importing to docker
--> loading layer sha256:0559e0501291916 27.74MB
--> loading layer 97a428807723 4.08KB / 4.08KB
--> loading layer 6fc374b0750 425.98KB / 49.56MB
--> loading layer 320957e07151 453B / 453B
--> loading layer 2337ae080fa3d 114B / 134B
--> loading layer 320957e07151 540B / 540B
--> loading layer 320957e07151 50.76KB / 19.76KB
--> loading layer 732532570675 229.38KB / 19.86MB
--> loading layer dfa58766120a 3.85KB / 3.85KB

What's next:
  View a summary of image vulnerabilities and recommendations -> docker scout quickview

```

#### 2. frontend

```

[+] 50/93 msx git:(main) ✘ docker build --platform linux/amd64 src/frontend -t frontend --load
[+] Building 29.1s (12/12) FINISHED
-> [internal] load build definition from Dockerfile
--> [internal] load build context from Dockerfile
-> [internal] load metadata for docker.io/library/node:ls-buster
--> [internal] load .dockerignore
--> [internal] load dockerignore
--> [internal] resolve docker.io/library/node:ls-buster@sha256:479103df06b40b90f189461b6f824a62986683e26a32c77d7c3e2d855a8e3e0f
--> [internal] load build context
--> CACHER [2/6] WORKDIR /src/src/app
--> CACHED [1/6] COPY package.json ./src/src/app/package.json
--> CACHED [1/6] COPY package-lock.json ./src/src/app/package-lock.json
--> CACHED [4/6] RUN npm ci
--> CACHED [5/6] COPY ./src/src/app
--> exporting layers
--> exporting manifest sha256:dc11854094764d0d77726e93c48a8c3923bdc6c5b396d4fa1d0381fd672
--> exporting config sha256:b5b6e68290d09c056c04de1dc02e54ed00be15ce032428534b14ff7b00c
--> sending manifest
--> sending config
--> importing to docker
--> loading layer 320957e07151 524.29KB / 50.65MB
--> loading layer 55d1b5590c 196.61KB / 17.39MB
--> loading layer e572e72782d3 51.50MB / 51.50MB
--> loading layer 320957e07151 50.76KB / 19.76KB
--> loading layer a7fcbaac839 4.08KB / 4.08KB
--> loading layer b077e5c0d886 491.52KB / 48.02MB
--> loading layer 320957e07151 540B / 540B
--> loading layer 6595e519388fc 452B / 452B
--> loading layer a9c6c099ab0b 132B / 132B
--> loading layer 320957e07151 50.76KB / 19.76KB
--> loading layer 1f4e51b108lab 120.32KB / 121.71MB
--> loading layer 784df580bee2 27.25KB / 27.25KB

What's next:
  View a summary of image vulnerabilities and recommendations -> docker scout quickview

```

### 2. Push Docker Images to ACR

#### 1. Docker Login

```

→ sd2793_msa git:(main) ✘ docker login sd2793arc.azurecr.io
Username: sd2793arc
>Password:
Login Succeeded

```

## 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
ec8ccb2796ae: 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
5952400addef: 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
```

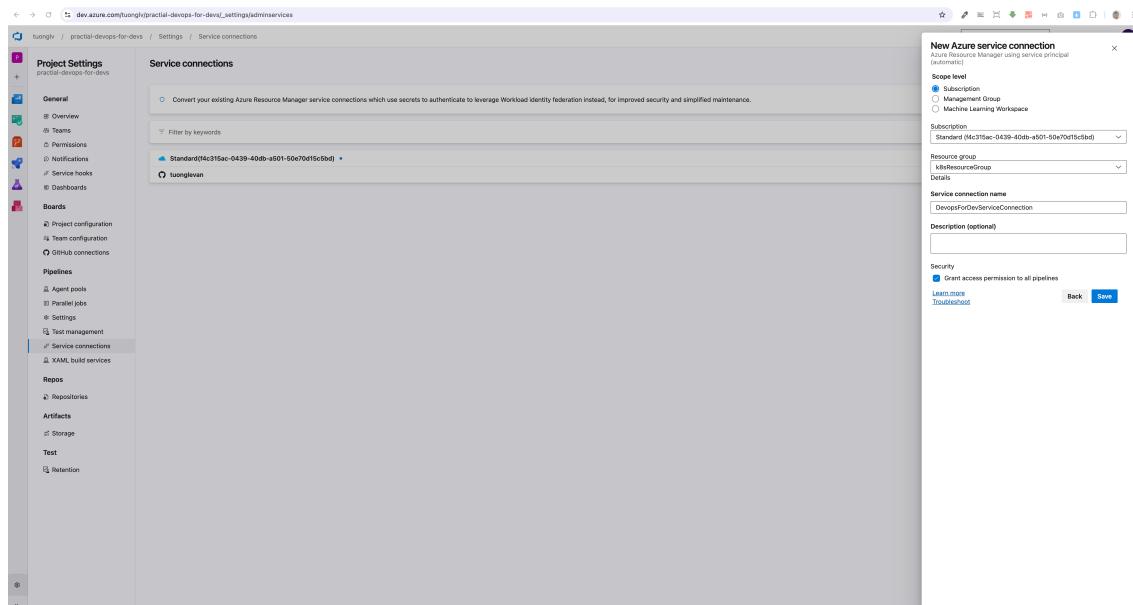
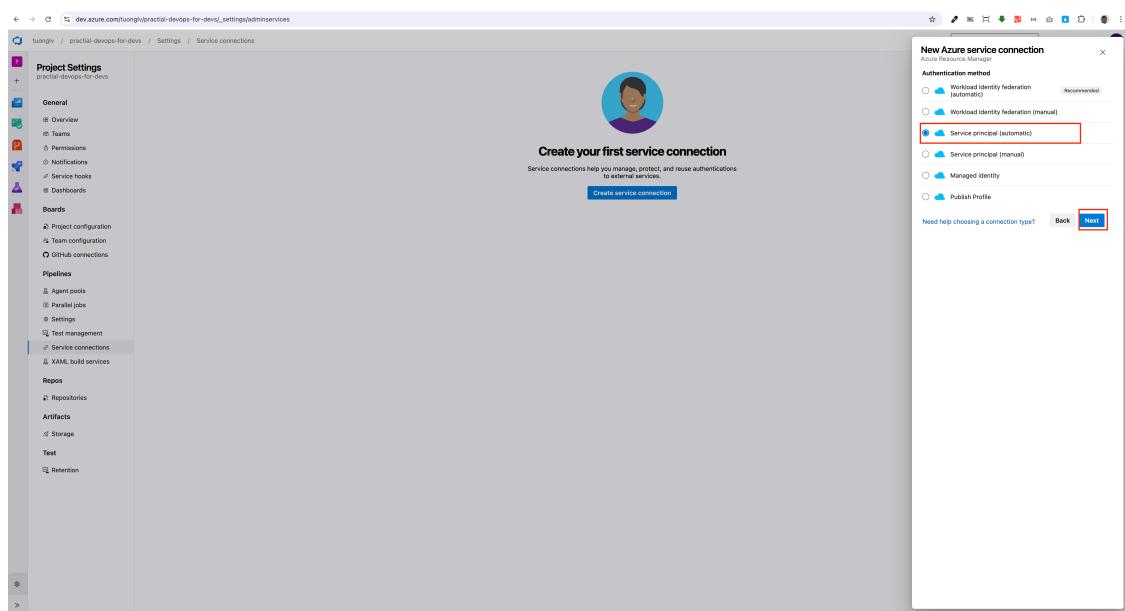
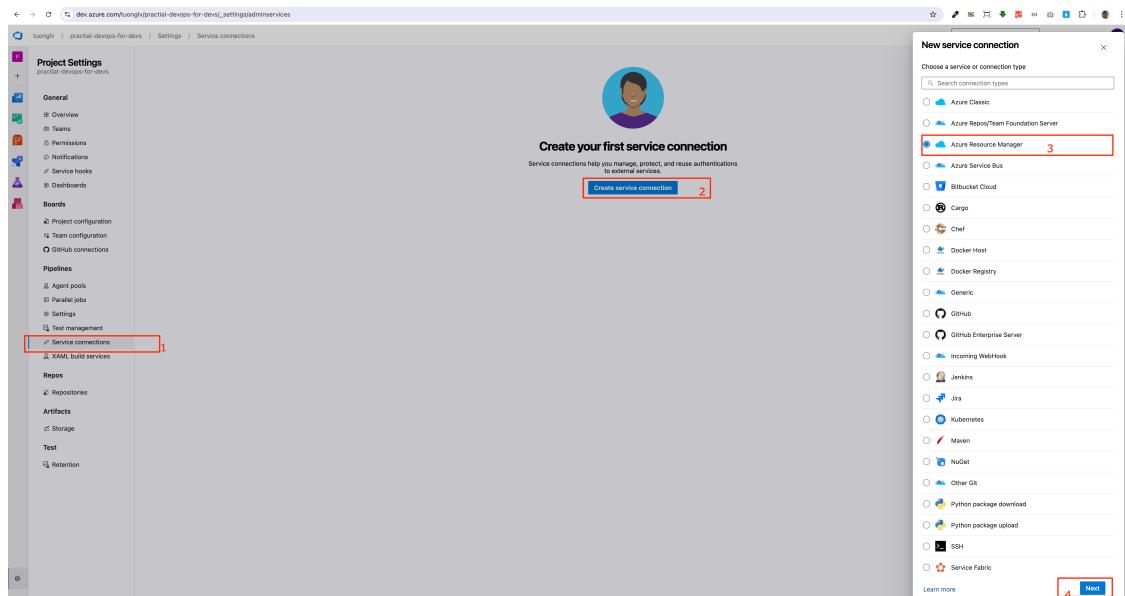
## 5. Verify Web UI

```
→ 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
```

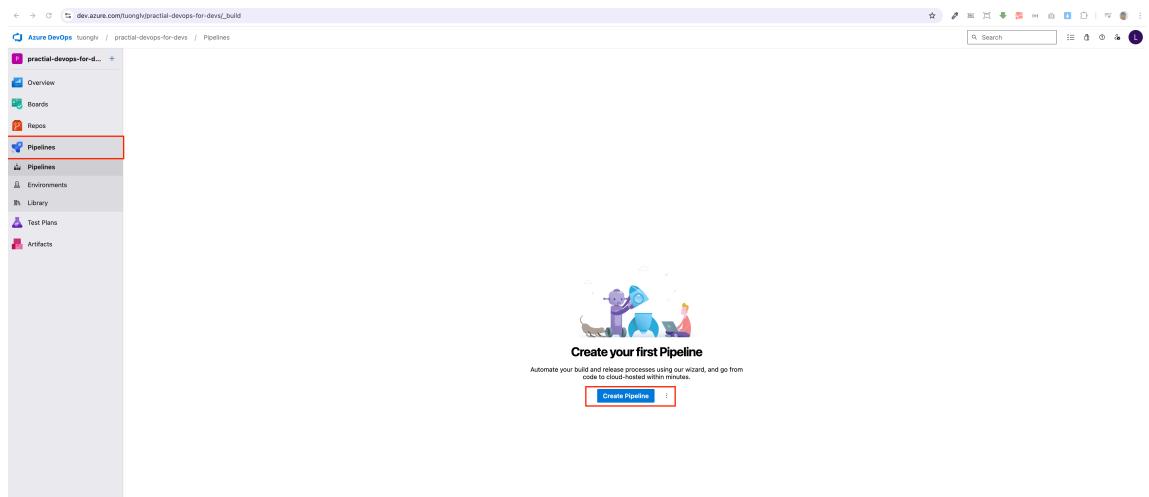


# VI. CI/CD On Azure Pipeline

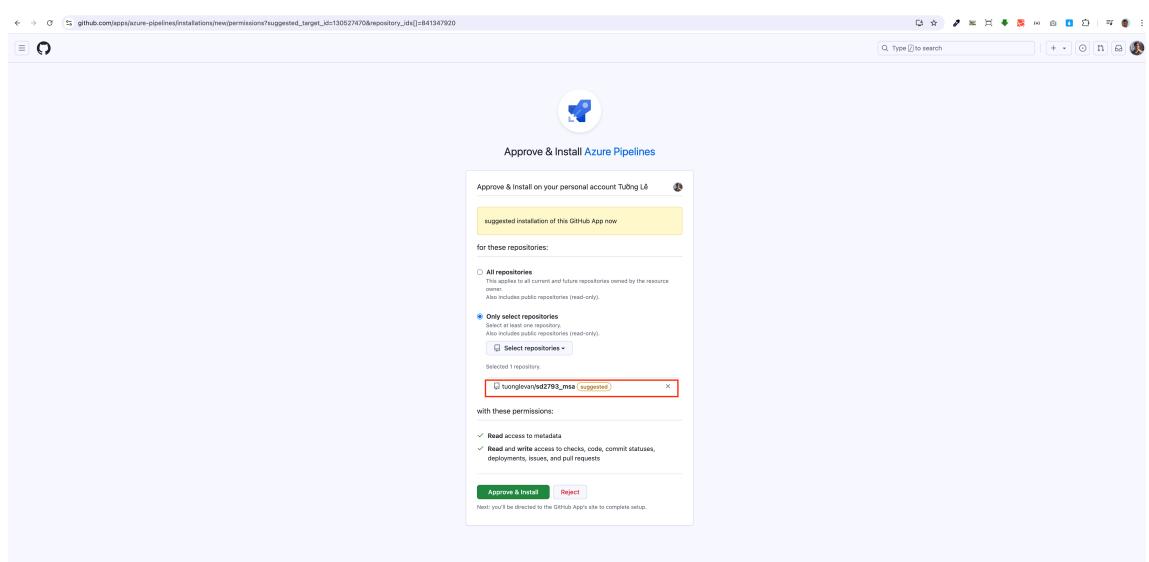
## 1. Service Connection



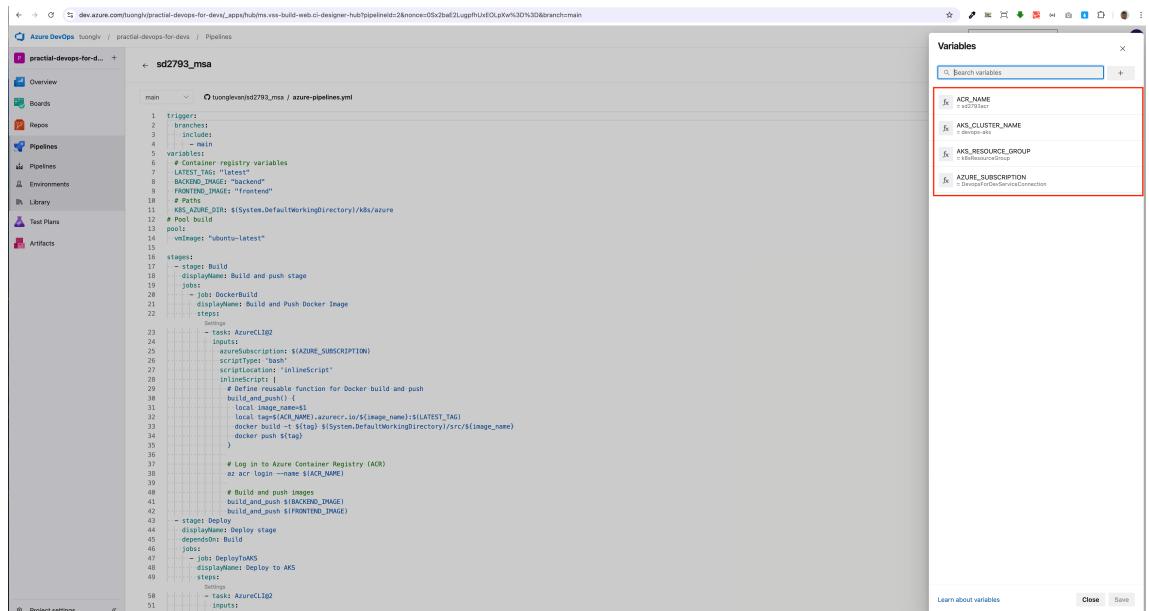
## 2. Setup Azure Pipeline



The screenshot shows the Azure DevOps Pipelines interface. On the left, a sidebar menu has 'Pipelines' selected. A central area displays a 'Create your first Pipeline' wizard with a cartoon illustration of two people working on a computer. Below the illustration, text reads: 'Automate your build and release processes using our wizard, and go from code to visual-hosted within minutes.' A prominent red-bordered button at the bottom right of the wizard says 'Create Pipeline'.

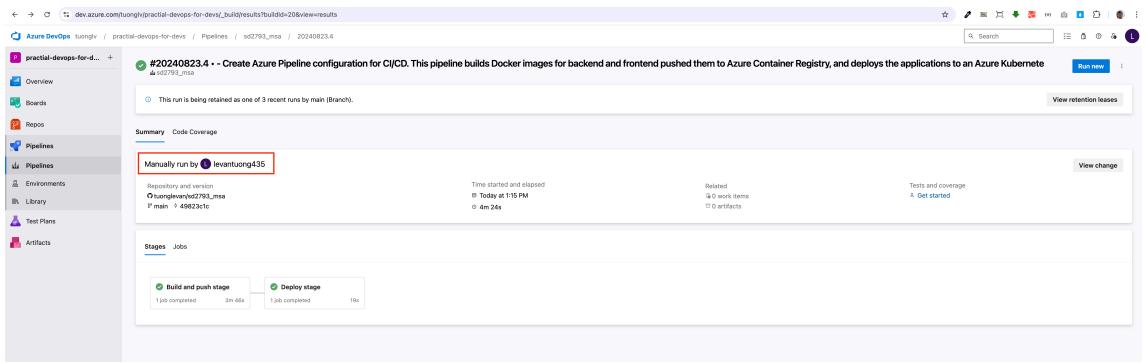
This screenshot shows a GitHub dialog titled 'Approve & Install Azure Pipelines'. It asks for permission to install the app on the user's account. The 'Only select repositories' option is selected, and a single repository 'tudongluong/vnfd2793\_ms' is chosen. The permissions requested are 'Read access to metadata' and 'Read and write access to checks, code, commit statuses, deployments, issues, and pull requests'. At the bottom, there are 'Approve & Install' and 'Reject' buttons.

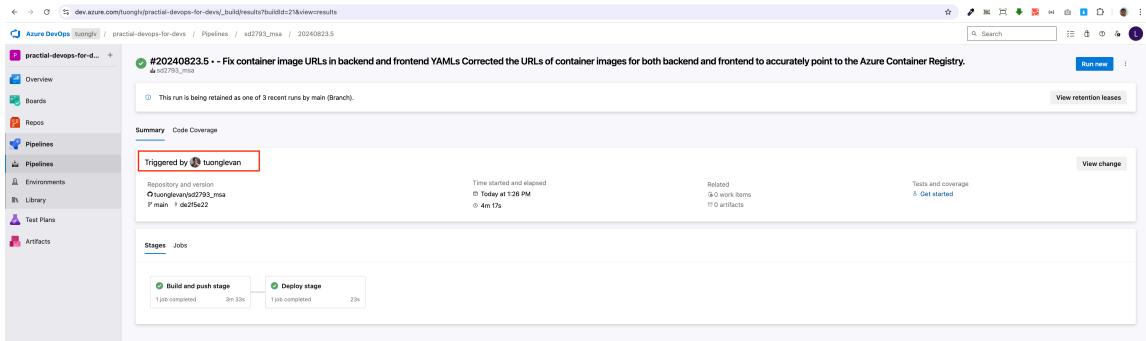
This screenshot shows the Azure DevOps pipeline configuration. The main pane displays a YAML file named 'azure-pipelines.yml' for a project 'sd2793\_ms'. The code defines a pipeline with triggers, branches, and an include section. It includes stages for building Docker images and pushing them to an Azure Container Registry (ACR). The 'Variables' panel on the right lists several environment variables: ACR\_NAME, AKS\_CLUSTER\_NAME, AKS\_RESOURCE\_GROUP, and AZURE\_RESOURCE\_GROUP. The 'ACR\_NAME' variable is highlighted with a red border.

### 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 Helm-ing!*
```

```
+ 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

The screenshot shows the Grafana interface for importing dashboards. On the left, there's a sidebar with navigation links like Home, Started, Dashboards, Explore, Alerting, Connections, and Administration. The main area is titled 'Dashboards' and contains a list of available dashboards. A red box highlights several items under the 'Kubernetes / Compute Resources' category, including 'Cluster', 'Namespace (Pods)', 'Namespace (Workloads)', 'Node (Pods)', 'Pod', and 'Workload'. To the right of the dashboard list, there's a 'Tags' section with a grid of color-coded tags: purple ('alarmmanager-default'), green ('grafana'), blue ('node-exporter'), and orange ('node-min'). At the top right, there are buttons for 'New dashboard', 'New folder', and 'Import'.

Dashboard ID: 1860 [Node Exporter Full](#)

This screenshot shows the 'Import dashboard' dialog box. It includes fields for 'Name' (set to 'Node Exporter Full'), 'Folder' (set to 'Dashboards'), and 'Unique identifier (UID)' (with a note about it being used for identifying dashboards). Under the 'Prometheus' section, it says 'Select a Prometheus data source'. At the bottom, there are 'Import' and 'Cancel' buttons.

### 4. Monitoring in the Dashboard UI.

