# **ASSIGNMENT 1 — APPLICATION DEPLOYMENT TO AKS**

Your goal here is:

Move the **same Kubernetes workload** from Docker Desktop → **Azure Kubernetes Service (AKS)**

## **1️⃣ Prerequisites for AKS**

Install & verify:

az version

kubectl version --client

Login to Azure:

az login

Set subscription (if multiple):

az account set --subscription "<SUBSCRIPTION\_ID>"

## **2️⃣ Create Azure Container Registry (ACR)**

AKS **cannot pull from localhost:5000**, so we replace your local registry with **ACR**.

az group create \

--name aks-rg \

--location eastus

az acr create \

--resource-group aks-rg \

--name manvithacr \

--sku Basic

Login to ACR:

az acr login --name manvithacr

ACR login server:

az acr show --name manvithacr --query loginServer -o tsv

Example:

manvithacr.azurecr.io

## **3️⃣ Build & Push Images to ACR (NO CODE CHANGES)**

### **Frontend**

docker build -t manvithacr.azurecr.io/frontend:latest frontend/

docker push manvithacr.azurecr.io/frontend:latest

### **Backend-A**

docker build -t manvithacr.azurecr.io/backend-a:latest backend-a/

docker push manvithacr.azurecr.io/backend-a:latest

### **Backend-B**

docker build -t manvithacr.azurecr.io/backend-b:latest backend-b/

docker push manvithacr.azurecr.io/backend-b:latest

✅ Only change needed in YAML:

image: manvithacr.azurecr.io/backend-a:latest

## **4️⃣ Create AKS Cluster**

az aks create \

--resource-group aks-rg \

--name image-app-aks \

--node-count 2 \

--enable-managed-identity \

--attach-acr manvithacr \

--generate-ssh-keys

Attach kubeconfig:

az aks get-credentials \

--resource-group aks-rg \

--name image-app-aks

Verify:

kubectl get nodes

## **5️⃣ Deploy Your Existing Kubernetes Manifests**

Your deployment order **DOES NOT CHANGE**:

kubectl apply -f k8s/00-namespace.yaml

kubectl apply -f k8s/01-resourcequota.yaml

kubectl apply -f k8s/02-secrets.yaml

kubectl apply -f k8s/03-configmap.yaml

kubectl apply -f k8s/04-pvc.yaml

kubectl apply -f k8s/05-postgres-statefulset.yaml

kubectl apply -f k8s/06-services.yaml

kubectl apply -f k8s/07-backend-a-deployment.yaml

kubectl apply -f k8s/08-backend-b-deployment.yaml

kubectl apply -f k8s/09-frontend-deployment.yaml

kubectl apply -f k8s/10-networkpolicy.yaml

kubectl apply -f k8s/11-hpa.yaml

## **6️⃣ Access Application (AKS)**

If frontend Service is NodePort:

kubectl get svc frontend -n image-app

For real AKS exposure (recommended):

type: LoadBalancer

Then:

kubectl get svc frontend -n image-app

Open:

http://<EXTERNAL-IP>

✅ **Assignment 1 DONE**

# **✅ ASSIGNMENT 2 — SONARQUBE INTEGRATION (VULNERABILITY CHECKS)**

SonarQube is used **before deployment** to validate **code quality + security**.

## **1️⃣ Run SonarQube (Docker – simplest for assignment)**

docker run -d \

--name sonarqube \

-p 9000:9000 \

sonarqube:lts

Open:

http://localhost:9000

Default login:

admin / admin

## **2️⃣ Create SonarQube Project**

1. Create Project → **Manual**
2. Project Key: image-app
3. Generate Token → copy it

## **3️⃣ Run Sonar Scan (Frontend Example)**

docker run --rm \

-e SONAR\_HOST\_URL=http://host.docker.internal:9000 \

-e SONAR\_LOGIN=<TOKEN> \

-v $(pwd)/frontend:/usr/src \

sonarsource/sonar-scanner-cli

Repeat for:

* backend-a
* backend-b

## **4️⃣ What SonarQube Checks (Mention in Report)**

✔ Code smells  
✔ Vulnerabilities  
✔ Security hotspots  
✔ Duplications  
✔ Maintainability

📸 Screenshot required:

* Project dashboard
* Passed Quality Gate

✅ **Assignment 2 DONE**

# **✅ ASSIGNMENT 3 — MONITORING WITH PROMETHEUS & GRAFANA**

We’ll use **Kubernetes-native monitoring**.

## **1️⃣ Install Metrics Server (Required for HPA)**

kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

Verify:

kubectl top pods -n image-app

## **2️⃣ Install Prometheus & Grafana (Helm)**

Install Helm:

helm version

Add repo:

helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

helm repo update

Install stack:

helm install monitoring prometheus-community/kube-prometheus-stack \

--namespace monitoring \

--create-namespace

## **3️⃣ Access Grafana**

Get password:

kubectl get secret monitoring-grafana \

-n monitoring \

-o jsonpath="{.data.admin-password}" | base64 --decode

Port-forward:

kubectl port-forward svc/monitoring-grafana 3000:80 -n monitoring

Open:

http://localhost:3000

Login:

admin / <password>

## **4️⃣ What You Must Show in Screenshots**

✔ Pod CPU usage  
✔ Memory usage  
✔ HPA scaling  
✔ Namespace-level metrics  
✔ Deployment replicas increasing

Prometheus automatically scrapes:

* kubelets
* cAdvisor
* API server
* Pods

✅ **Assignment 3 DONE**

# **🧠 WHAT TO WRITE IN YOUR REPORT (COPY SAFE)**

The application was deployed to Azure Kubernetes Service using container images stored in Azure Container Registry. Kubernetes manifests were reused without modification, demonstrating platform portability. SonarQube was integrated into the CI validation phase to perform static code analysis, vulnerability detection, and quality gate enforcement. Cluster monitoring was implemented using Prometheus and Grafana to visualize resource usage, pod health, and autoscaling behavior. Horizontal Pod Autoscaler dynamically scaled services based on CPU and memory metrics, ensuring high availability and efficient resource utilization.

# **✅ FINAL CHECKLIST FOR SUBMISSION**

✔ AKS nodes running  
✔ ACR images pushed  
✔ Pods in Running  
✔ Frontend accessible via LoadBalancer  
✔ SonarQube dashboard screenshot  
✔ Grafana metrics screenshot  
✔ HPA scaling screenshot

## **🔥 NEXT STEP (I CAN DO THIS FAST FOR YOU)**

Tell me **one**:

1️⃣ “Give me AKS-specific YAML changes only”  
2️⃣ “Generate full README.md for all 3 assignments”  
3️⃣ “Give me screenshot checklist + commands”  
4️⃣ “Create architecture diagram (AKS + Sonar + Monitoring)”

I’ll generate it **ready to submit**.