==============================

### MANUAL: CLONE AND RUN THE SAMPLE FLASK APPLICATION

==============================

1. CLONE THE APP FROM GITLAB

---------------------------------

Command:

git clone https://gitlab.com/swarajpatil\_airo/sampleapp.git

cd sampleapp

2. INSTALL DEPENDENCIES

---------------------------------

Command:

pip install -r requirements.txt

3. RUN THE FLASK APP LOCALLY

---------------------------------

Command:

python app.py

# The app will be accessible at: http://localhost:5000

4. BUILD AND RUN THE DOCKER IMAGE

---------------------------------

Commands:

docker build -t mysampleapp .

docker run -d -p 5000:5000 mysampleapp

# Verify the container is running

docker ps

# Access the app in your browser: http://localhost:5000

5. PUSH DOCKER IMAGE TO DOCKER HUB

---------------------------------

Commands:

docker login

docker tag mysampleapp <your-username>/mysampleapp:latest

docker push <your-username>/mysampleapp:latest

# Verify the image on Docker Hub portal

# This image can now be used for Kubernetes deployment  
  
--------------------------------------------------------------------------------------------------------------  
  
Manual: Create Single-Node AKS Cluster on Azure (via Portal UI)

## 1. Log in

1. Go to https://portal.azure.com/  
2. Sign in with your Azure account.

## 2. Create a Resource Group

1. In the left menu, search \*\*Resource groups\*\*.  
2. Click \*\*+ Create\*\*.  
3. Fill in:  
 - Subscription → your free subscription  
 - Resource group name → e.g. myResourceGroup  
 - Region → nearest to you (East US, Central India, etc.)  
4. Click \*\*Review + Create → Create\*\*.

## 3. Create AKS Cluster

1. In the portal search bar, type \*\*Kubernetes services\*\*.  
2. Click \*\*+ Create → Kubernetes cluster\*\*.  
3. Fill in:  
 - Subscription → same as above  
 - Resource group → myResourceGroup  
 - Kubernetes cluster name → e.g. myAKSCluster  
 - Region → same as above  
 - Kubernetes version → default or latest stable  
 - Availability zones → None (since single node is enough)

## 4. Configure Node Pool

1. In \*\*Node Pools\*\* tab:  
 - Node size → Standard\_B2s (cheap, 2 vCPU, 4GB RAM)  
 - Node count → 1  
 - Enable autoscaling → Disable  
2. Leave Node Pool Mode = System.

## 5. Authentication

1. In \*\*Authentication\*\* tab:  
 - Leave defaults (System-assigned managed identity).  
 - RBAC/AD integration not required for testing simple apps.

## 6. Networking

1. In \*\*Networking\*\* tab:  
 - Networking configuration → leave default (Azure CNI or Kubenet).  
 - Load Balancer → Standard.  
 - Other settings → default.

## 7. Review + Create

1. Click \*\*Review + Create\*\*.  
2. Wait for validation, then click \*\*Create\*\*.  
 (Deployment takes ~5–10 minutes).

## 8. Connect to Cluster

1. Once deployed, go to the AKS resource → \*\*Connect\*\*.  
2. Follow on-screen instructions (you may need Azure CLI).  
 Example command:  
 az aks get-credentials --resource-group myResourceGroup --name myAKSCluster  
3. Verify cluster with:  
 kubectl get nodes  
 (You should see 1 node in Ready state).

## 9. Deploy Your Custom App

### Option A: Portal UI  
1. Go to your AKS cluster → Workloads → Deployments → + Create.  
2. Enter:  
 - Name: myapp  
 - Container image: <your-dockerhub-username>/<your-image>:latest  
 - Port: 80  
3. Create a Service:  
 - Type: LoadBalancer  
 - Port: 80 → TargetPort: 80  
 - Link it to myapp deployment.  
  
### Option B: YAML (CLI)  
Upload a YAML file with Deployment + Service configuration.

## 10. Access Your App

1. Go to \*\*Services and Ingresses\*\* in Portal.  
2. Find your myapp-service.  
3. Copy the External IP.  
4. Open in browser:  
 http://<External-IP>