☁️ Cloud Deployment Guide - Complete Beginner's Guide

🎯 Overview

This guide will teach you **step-by-step** how to deploy your Heart Disease Prediction MLOps project to the cloud.

📋 Table of Contents

🔵 Azure Container Apps

Step 1: Create Azure Account

1. Go to: azure.microsoft.com/free/students
2. Click "Start Free" or "Activate Now"
3. Sign in with your school email
4. Complete verification (you'll get $200 credit)
5. Access Azure Portal: portal.azure.com

Step 2: Install Azure CLI

For macOS:

```
brew install azure-cli
```

For Windows:

Download from: aka.ms/installazurecliwindows

For Linux:

```
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
```

Verify Installation:

```
az --version
```

Step 3: Login to Azure

```
# Login
az login

# Set your subscription (if you have multiple)
az account list --output table
az account set --subscription "Your-Subscription-Name"

# Verify
az account show
```

Step 4: Create Resource Group

```
# Create a resource group (like a folder for your resources)
az group create \
  --name heart-disease-rg \
  --location eastus

# Verify
az group list --output table
```

Choose a location near you:

- US: `eastus`, `westus2`
- Europe: `westeurope`, `northeurope`
- Asia: `southeastasia`, `eastasia`

Step 5: Create Container Registry (To Store Your Docker Image)

```
# Create Azure Container Registry
az acr create \
  --resource-group heart-disease-rg \
  --name heartdiseaseacr \
  --sku Basic \
  --admin-enabled true

# Login to registry
az acr login --name heartdiseaseacr
```

```
# Get login server
az acr show --name heartdiseaseacr --query loginServer --output table
```

Note: Registry name must be globally unique. If taken, try: `heartdiseaseacr123`, `hdacr2024`, etc.

Step 6: Build and Push Docker Image to Azure

```
# Navigate to your project
cd heart-disease-mlops

# Build and push in one command (Azure will do it for you!)
az acr build \
  --registry heartdiseaseacr \
  --image heart-disease-api:v1 \
  --file Dockerfile \
  .

# This will take 3-5 minutes
# Azure builds your Docker image in the cloud!
```

Expected Output:

```
Successfully tagged heartdiseaseacr.azurecr.io/heart-disease-api:v1
Successfully pushed heartdiseaseacr.azurecr.io/heart-disease-api:v1
```

Step 7: Create Container App Environment

```
# Install Container Apps extension
az extension add --name containerapp --upgrade

# Register providers
az provider register --namespace Microsoft.App
az provider register --namespace Microsoft.OperationalInsights

# Create environment (like a hosting space)
az containerapp env create \
  --name heart-disease-env \
  --resource-group heart-disease-rg \
  --location eastus
```

Step 8: Deploy Your Container App

```
# Get ACR credentials
ACR_USERNAME=$(az acr credential show --name heartdiseaseacr --query
username --output tsv)
ACR_PASSWORD=$(az acr credential show --name heartdiseaseacr --query
passwords[0].value --output tsv)

# Deploy the container app
az containerapp create \
  --name heart-disease-api \
  --resource-group heart-disease-rg \
  --environment heart-disease-env \
  --image heartdiseaseacr.azurecr.io/heart-disease-api:v1 \
  --registry-server heartdiseaseacr.azurecr.io \
  --registry-username $ACR_USERNAME \
  --registry-password $ACR_PASSWORD \
  --target-port 8000 \
  --ingress external \
  --min-replicas 1 \
  --max-replicas 3 \
  --cpu 0.5 \
  --memory 1Gi

# Get the URL
az containerapp show \
  --name heart-disease-api \
  --resource-group heart-disease-rg \
  --query properties.configuration.ingress.fqdn \
  --output tsv
```

We'll get a URL like: `https://heart-disease-api.azurecontainerapps.io`

Step 9: Test Your Cloud API

```
# Replace with your actual URL
export API_URL="https://heart-disease-api.azurecontainerapps.io"

# Test health endpoint
curl $API_URL/health

# Test prediction
curl -X POST "$API_URL/predict" \
  -H "Content-Type: application/json" \
  -d '{
    "age": 63,
    "sex": 1,
    "cp": 3,
    "trestbps": 145,
    "chol": 233,
    "fbs": 1,
    "restecg": 0,
```

```
    "thalach": 150,
    "exang": 0,
    "oldpeak": 2.3,
    "slope": 0,
    "ca": 0,
    "thal": 1
  }'

# Open in browser
open "$API_URL/docs"
```

✅ **Your API is now live on the internet!**

Step 10: View Logs and Monitor

```
# View logs
az containerapp logs show \
  --name heart-disease-api \
  --resource-group heart-disease-rg \
  --follow

# Check metrics in Azure Portal
# Go to: portal.azure.com → Resource Groups → heart-disease-rg → heart-
disease-api
```

🔄 Setup CI/CD Pipeline with GitHub Actions

Step 1: Create GitHub Repository

```
# Initialize git (if not already done)
cd heart-disease-mlops
git init
git add .
git commit -m "Initial commit"

# Create repo on GitHub.com, then:
git remote add origin https://github.com/YOUR_USERNAME/heart-disease-
mlops.git
git branch -M main
git push -u origin main
```

Step 2: Setup GitHub Secrets

Go to: **GitHub Repository → Settings → Secrets and variables → Actions**

For Azure:

Add these secrets:

- `AZURE_CREDENTIALS` (Get from: `az ad sp create-for-rbac --name "github-actions" --role contributor --scopes /subscriptions/YOUR_SUBSCRIPTION_ID/resourceGroups/heart-disease-rg --sdk-auth`)
- `AZURE_REGISTRY_NAME`: `heartdiseaseacr`
- `AZURE_REGISTRY_USERNAME`: (From Azure portal)
- `AZURE_REGISTRY_PASSWORD`: (From Azure portal)

For AWS:

- `AWS_ACCESS_KEY_ID`
- `AWS_SECRET_ACCESS_KEY`
- `AWS_ACCOUNT_ID`

For Google Cloud:

- `GCP_PROJECT_ID`
- `GCP_SA_KEY` (Service account key JSON)

Step 3: Create GitHub Actions Workflow

Create `.github/workflows/azure-deploy.yml`:

```yaml
name: CI/CD Pipeline - Azure Deployment

on:
  push:
    branches: [ main, develop ]
  pull_request:
    branches: [ main ]

jobs:
  test:
    name: Run Tests
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v3

    - name: Set up Python
      uses: actions/setup-python@v4
      with:
        python-version: '3.9'

    - name: Install dependencies
      run: |
```

```
        python -m pip install --upgrade pip
        pip install -r requirements.txt
        pip install pytest pytest-cov flake8

    - name: Lint with flake8
      run: |
        flake8 src/ --count --select=E9,F63,F7,F82 --show-source --
statistics

    - name: Run tests
      run: |
        pytest tests/ -v --cov=src

  build-and-deploy:
    name: Build and Deploy to Azure
    needs: test
    runs-on: ubuntu-latest
    if: github.ref == 'refs/heads/main' && github.event_name == 'push'

    steps:
    - uses: actions/checkout@v3

    - name: Azure Login
      uses: azure/login@v1
      with:
        creds: ${{ secrets.AZURE_CREDENTIALS }}

    - name: Build and push image to ACR
      run: |
        az acr build \
          --registry ${{ secrets.AZURE_REGISTRY_NAME }} \
          --image heart-disease-api:${{ github.sha }} \
          --image heart-disease-api:latest \
          --file Dockerfile \
          .

    - name: Deploy to Azure Container Apps
      run: |
        az containerapp update \
          --name heart-disease-api \
          --resource-group heart-disease-rg \
          --image ${{ secrets.AZURE_REGISTRY_NAME }}.azurecr.io/heart-
disease-api:${{ github.sha }}

    - name: Verify Deployment
      run: |
        URL=$(az containerapp show \
          --name heart-disease-api \
          --resource-group heart-disease-rg \
          --query properties.configuration.ingress.fqdn \
          --output tsv)

        echo "Testing health endpoint..."
        curl -f https://$URL/health || exit 1
```

```
        echo "✅ Deployment successful!"
        echo "API URL: https://$URL"
```

Step 4: Push and Watch CI/CD in Action

```
# Make a change
echo "# Test CI/CD" >> README.md

# Commit and push
git add .
git commit -m "Test CI/CD pipeline"
git push origin main

# Go to GitHub → Actions tab to watch the magic happen!
```

What happens:

1. ✅ Code is checked out
2. ✅ Tests run automatically
3. ✅ Docker image is built
4. ✅ Image is pushed to container registry
5. ✅ Application is deployed to cloud
6. ✅ Health check verifies deployment

🧪 Testing Your Cloud Deployment

Test Health Endpoint

```
curl https://YOUR_CLOUD_URL/health
```

Test Prediction Endpoint

```
curl -X POST "https://YOUR_CLOUD_URL/predict" \
  -H "Content-Type: application/json" \
  -d '{
    "age": 63,
    "sex": 1,
    "cp": 3,
    "trestbps": 145,
    "chol": 233,
    "fbs": 1,
    "restecg": 0,
    "thalach": 150,
```

```
    "exang": 0,
    "oldpeak": 2.3,
    "slope": 0,
    "ca": 0,
    "thal": 1
  }'
```

Test in Browser

Open: `https://YOUR_CLOUD_URL/docs`

Test with Python

```python
import requests

API_URL = "https://YOUR_CLOUD_URL"

# Health check
response = requests.get(f"{API_URL}/health")
print("Health:", response.json())

# Prediction
data = {
    "age": 63,
    "sex": 1,
    "cp": 3,
    "trestbps": 145,
    "chol": 233,
    "fbs": 1,
    "restecg": 0,
    "thalach": 150,
    "exang": 0,
    "oldpeak": 2.3,
    "slope": 0,
    "ca": 0,
    "thal": 1
}

response = requests.post(f"{API_URL}/predict", json=data)
print("Prediction:", response.json())
```

💰 Cost Management

Azure

```
# Check costs
az consumption usage list --output table
```

```
# Set up budget alert
az consumption budget create \
  --budget-name monthly-budget \
  --amount 10 \
  --time-grain Monthly
```

Stop Services When Not Needed

```
# Azure
az containerapp update --name heart-disease-api \
  --resource-group heart-disease-rg \
  --min-replicas 0
```
