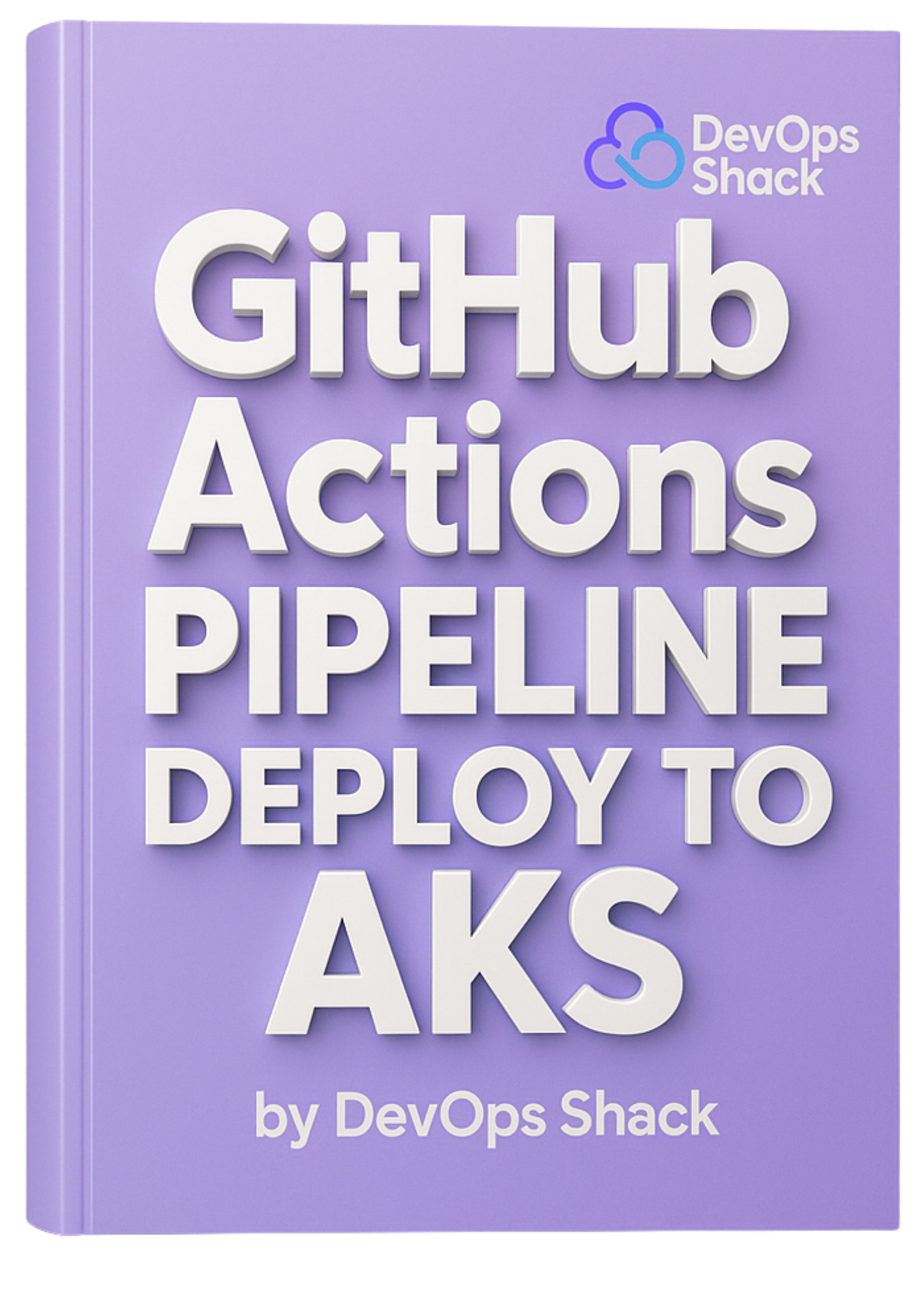
****

**GitHub Actions Piepline Deploy To AKS**

**Repo is added below the video in module**

**PHASE 1: Prerequisites**

Before starting, ensure you have:

| **Tool** | **Command to verify** |
| --- | --- |
| Azure CLI | az version (install from MS site) |
| kubectl CLI | kubectl version --client |
| Logged in to Azure | az login |
| Active Subscription | az account show |

**PHASE-1.2: Connect to Azure Account**

az login –use-device-code

**PHASE 2: Create the Resource Group**

az group create `

--name devops-rg `

--location centralus

**PHASE 3: Create AKS Cluster with Workload Identity + OIDC**

This is the full, PowerShell-friendly multi-line version:

az aks create `

--resource-group devops-rg `

--name devops-aks `

--node-count 2 `

--enable-managed-identity `

--generate-ssh-keys `

--enable-workload-identity `

--enable-oidc-issuer

⚠️ This will:

* Create AKS with default system node pool
* Enable Workload Identity (for GitHub OIDC-based login)
* Generate SSH keys in ~/.ssh

**PHASE 4: The "get-credentials" Issue Appears**

Even after successful creation, the command:

az aks get-credentials --resource-group devops-rg --name devops-aks

**Summary of Final State**

| **Step** | **Status** |
| --- | --- |
| Resource group created | ✅ devops-rg |
| AKS cluster created | ✅ devops-aks |
| Node pool up | ✅ 2 nodes |
| kubectl connected | ✅ Working |

**Optional: Confirm OIDC Issuer Setup**

Check that OIDC URL is working:

az aks show `

--resource-group devops-rg `

--name devops-aks `

--query "oidcIssuerProfile.issuerUrl" `

-o tsv

This URL is required when integrating with GitHub Actions via OIDC token exchange.

**Full PowerShell Script for GitHub OIDC to Azure AKS**

**🔹 Step 1: Create Azure AD App for GitHub Identity**

$App = az ad app create --display-name "github-aks-identity" | ConvertFrom-Json

$AppId = $App.appId

$AppObjectId = $App.id

**🔹 Step 2: Get Tenant ID and Subscription ID**

$TenantId = az account show --query tenantId -o tsv

$SubscriptionId = az account show --query id -o tsv

**🔹 Step 3: Create Federated Credential JSON**

**# Make sure to get the value of AppObjectId using write-host $AppObjectId**

**# in $Subject Add Your github Username & reponame**

$AppObjectId = "5a1ddc91-d654-4b3c-a597-1f1992792361"

$Subject = "repo:jaiswaladi246/MultiTier-DotNET-MongoDB-K8:ref:refs/heads/main"

$TempFile = "federated-credential.json"

# Delete old if exists

$OldId = az ad app federated-credential list --id $AppObjectId --query "[?name=='github-oidc-cred'].id" -o tsv

if ($OldId) { az ad app federated-credential delete --id $AppObjectId --federated-credential-id $OldId }

# Create new credential

@{

name = "github-oidc-cred"

issuer = "https://token.actions.githubusercontent.com"

subject = $Subject

description = "OIDC federated credential from GitHub Actions"

audiences = @("api://AzureADTokenExchange")

} | ConvertTo-Json -Depth 3 | Out-File $TempFile -Encoding utf8

az ad app federated-credential create --id $AppObjectId --parameters $TempFile

Remove-Item $TempFile

**🔹 Step 5: Create Service Principal (required for RBAC)**

az ad sp create --id $AppId

Wait ~15 seconds after this command due to Azure replication delays.

**🔹 Step 6: Get AKS Resource ID**

$AksName = "devops-aks"

$ResourceGroup = "devops-rg"

$AksId = az aks show `

--name $AksName `

--resource-group $ResourceGroup `

--query id -o tsv

**🔹 Step 7: Assign RBAC Role to Service Principal to Access AKS Credentials**

$SpObjectId = az ad sp show --id $AppId --query id -o tsv

az role assignment create `

--assignee $SpObjectId `

--role "Azure Kubernetes Service RBAC Cluster Admin" `

--scope $AksId

**✅ Step 8: Add These GitHub Repo Secrets**

In your GitHub repo:  
**Settings → Secrets and variables → Actions → New repository secret**

| **Secret Name** | **Value** |
| --- | --- |
| AZURE\_CLIENT\_ID | az ad app list --display-name "github-aks-identity" --query "[0].appId" -o tsv |
| AZURE\_TENANT\_ID | az account show --query tenantId -o tsv |
| AZURE\_SUBSCRIPTION\_ID | az account show --query id -o tsv |
| AZURE\_RESOURCE\_GROUP | "devops-rg" |
| AKS\_CLUSTER\_NAME | "devops-aks" |
| DOCKERHUB\_USERNAME | "adijaiswal" |
| DOCKERHUB\_TOKEN | DockerHub Access Token from your DockerHub account |

**🔹 Step 11: Add Correct GitHub Actions Workflow (.github/workflows/deploy.yml)**

name: CI/CD to AKS using GitHub OIDC

# correction 12

on:

push:

branches:

- main

permissions:

id-token: write

contents: read

env:

IMAGE\_NAME: adijaiswal/dotnet-mongo-crud

TAG: latest

jobs:

build-and-deploy:

runs-on: ubuntu-latest

steps:

- name: Checkout Code

uses: actions/checkout@v3

- name: Login to DockerHub

uses: docker/login-action@v2

with:

username: ${{ secrets.DOCKERHUB\_USERNAME }}

password: ${{ secrets.DOCKERHUB\_TOKEN }}

- name: Build Docker Image

run: docker build -t $IMAGE\_NAME:$TAG .

- name: Push Docker Image

run: docker push $IMAGE\_NAME:$TAG

- name: Azure Login with OIDC

uses: azure/login@v1

with:

client-id: ${{ secrets.AZURE\_CLIENT\_ID }}

tenant-id: ${{ secrets.AZURE\_TENANT\_ID }}

subscription-id: ${{ secrets.AZURE\_SUBSCRIPTION\_ID }}

- name: Set AKS context

uses: azure/aks-set-context@v3

with:

resource-group: ${{ secrets.AZURE\_RESOURCE\_GROUP }}

cluster-name: ${{ secrets.AKS\_CLUSTER\_NAME }}

- name: Deploy to AKS

run: |

kubectl get ns webapps || kubectl create ns webapps

kubectl apply -f dotnet-mongo-app.yaml -n webapps

**🔹 Step 12: Trigger the Workflow**

git commit --allow-empty -m "🚀 Final: Trigger OIDC AKS deploy"

git push

Go to **GitHub → Actions**, and you should now see:

* ✅ Azure login success via OIDC
* ✅ Docker image pushed to DockerHub
* ✅ Kubernetes context set via aks-set-context
* ✅ App deployed to AKS