Application Deployment with Azure Kubernetes Service (AKS) and Azure Pipelines

Introduction

This document provides a step-by-step guide to deploying an application on Azure Kubernetes Service (AKS) using Azure Pipelines. It covers setting up the AKS cluster, configuring Azure Pipelines, and automating the deployment process.

Prerequisites

Before starting, ensure you have the following:

- *An **Azure account** with access to create resources.*
- ***Azure CLI** is installed on your local machine.*
- *A **GitHub repository** containing your application code.*
- *An **Azure DevOps account** with a project set up.*
- ***kubectl** installed for Kubernetes cluster management.*

Step 1: Set Up an AKS Cluster

1. ***Login to Azure***
   az login
2. ***Create a Resource Group***
   az group create --name MyResourceGroup --location eastus
3. ***Create an AKS Cluster***
   az aks create --resource-group MyResourceGroup --name MyAKSCluster --node-count 2 --enable-addons monitoring --generate-ssh-keys

4. ***Get AKS Credentials***
   ```
   az aks get-credentials --resource-group MyResourceGroup
   --name MyAKSCluster
   ```
5. ***Verify the Cluster***
   ```
   kubectl get nodes
   ```

Step 2: Configure Azure Container Registry (ACR)

1. ***Create an ACR***
   ```
   az acr create --resource-group MyResourceGroup --name
   MyACR --sku Basic
   ```
2. ***Login to ACR***
   ```
   az acr login --name MyACR
   ```
3. ***Enable AKS to Pull Images from ACR***
   ```
   az aks update --resource-group MyResourceGroup --name
   MyAKSCluster --attach-acr MyACR
   ```

Step 3: Build and Push Docker Image

1. ***Build the Docker Image***
   ```
   docker build -t myacr.azurecr.io/myapp:v1 .
   ```
2. ***Push the Image to ACR***
   ```
   docker push myacr.azurecr.io/myapp:v1
   ```

Step 4: Deploy Application to AKS

Create a Deployment YAML file (deployment.yaml)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
spec:
  replicas: 2
  selector:
    matchLabels:
```

```
    app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: myapp
        image: myacr.azurecr.io/myapp:v1
        ports:
```

1.
   ```
           - containerPort: 80
   ```
2. ***Apply the Deployment***
   ```
   kubectl apply -f deployment.yaml
   ```
3. ***Verify the Deployment***
   ```
   kubectl get pods
   ```

Step 5: Set Up Azure Pipelines for CI/CD

1. ***Go to Azure DevOps*** *and navigate to your project.*
2. ***Create a new pipeline*** *and select your GitHub repository.*

Choose "Starter Pipeline" *and replace the content with the following YAML:*

```
trigger:
- main

pool:
  vmImage: 'ubuntu-latest'

steps:
- task: Docker@2
  inputs:
    containerRegistry: 'MyACR'
    repository: 'myapp'
```

```
    command: 'buildAndPush'
    Dockerfile: '**/Dockerfile'
    tags: '$(Build.BuildId)'

- task: Kubernetes@1
  inputs:
    connectionType: 'Azure Resource Manager'
    azureSubscription: 'MyAzureSubscription'
    azureResourceGroup: 'MyResourceGroup'
    kubernetesCluster: 'MyAKSCluster'
    command: 'apply'
```

3. *arguments: '-f deployment.yaml'*
4. ***Save and Run the Pipeline.***
5. ***Verify Deployment on AKS***
   kubectl get pods