Deploying to Azure Kubernetes Service

In this article

You can deploy your project to Azure Kubernetes Service (AKS) as part of your continuous deployment (CD) workflows.

Introduction

This guide explains how to use GitHub Actions to build and deploy a project to Azure Kubernetes Service.

Note: If your GitHub Actions workflows need to access resources from a cloud provider that supports OpenID Connect (OIDC), you can configure your workflows to authenticate directly to the cloud provider. This will let you stop storing these credentials as long-lived secrets and provide other security benefits. For more information, see "About security hardening with OpenID Connect" and "Configuring OpenID Connect in Azure."

Prerequisites

Before creating your GitHub Actions workflow, you will first need to complete the following setup steps:

1. Create a target AKS cluster and an Azure Container Registry (ACR). For more information, see "Quickstart: Deploy an AKS cluster by using the Azure portal - Azure Kubernetes Service" and "Quickstart - Create registry in portal - Azure Container Registry" in the Azure documentation.
2. Create a secret called `AZURE_CREDENTIALS` to store your Azure credentials. For more information about how to find this information and structure the secret, see the `Azure/login` action documentation.

Creating the workflow

Once you've completed the prerequisites, you can proceed with creating the workflow.

The following example workflow demonstrates how to build and deploy a project to Azure Kubernetes Service when code is pushed to your repository.

Under the workflow `env` key, change the following values:

- `AZURE_CONTAINER_REGISTRY` to the name of your container registry
- `PROJECT_NAME` to the name of your project

- `RESOURCE_GROUP` to the resource group containing your AKS cluster
- `CLUSTER_NAME` to the name of your AKS cluster

This workflow uses the `helm` render engine for the `azure/k8s-bake` action. If you will use the `helm` render engine, change the value of `CHART_PATH` to the path to your helm file. Change `CHART_OVERRIDE_PATH` to an array of override file paths. If you use a different render engine, update the input parameters sent to the `azure/k8s-bake` action.

YAML

```yaml
# This workflow uses actions that are not certified by GitHub.
# They are provided by a third-party and are governed by
# separate terms of service, privacy policy, and support
# documentation.

# GitHub recommends pinning actions to a commit SHA.
# To get a newer version, you will need to update the SHA.
# You can also reference a tag or branch, but the action may change without
warning.

name: Build and deploy to Azure Kubernetes Service

env:
  AZURE_CONTAINER_REGISTRY: MY_REGISTRY_NAME # set this to the name of your
container registry
  PROJECT_NAME: MY_PROJECT_NAME              # set this to your project's name
  RESOURCE_GROUP: MY_RESOURCE_GROUP          # set this to the resource group
containing your AKS cluster
  CLUSTER_NAME: MY_CLUSTER_NAME              # set this to the name of your AKS
cluster
  REGISTRY_URL: MY_REGISTRY_URL              # set this to the URL of your
registry
  # If you bake using helm:
  CHART_PATH: MY_HELM_FILE                   # set this to the path to your helm
file
  CHART_OVERRIDE_PATH: MY_OVERRIDE_FILES     # set this to an array of override
file paths

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v4

    - name: Azure Login
      uses: azure/login@14a755a4e2fd6dff25794233def4f2cf3f866955
      with:
        creds: ${{ secrets.AZURE_CREDENTIALS }}

    - name: Build image on ACR
      uses: azure/CLI@61bb69d64d613b52663984bf12d6bac8fd7b3cc8
      with:
        azcliversion: 2.29.1
        inlineScript: |
          az configure --defaults acr=${{ env.AZURE_CONTAINER_REGISTRY }}
          az acr build -t  -t ${{ env.REGISTRY_URL }}/${{ env.PROJECT_NAME }}:${{
github.sha }}

    - name: Gets K8s context
      uses: azure/aks-set-context@94ccc775c1997a3fcfbfbce3c459fec87e0ab188
      with:
          creds: ${{ secrets.AZURE_CREDENTIALS }}
          resource-group: ${{ env.RESOURCE_GROUP }}
          cluster-name: ${{ env.CLUSTER_NAME }}
      id: login
```

```
    - name: Configure deployment
      uses: azure/k8s-bake@61041e8c2f75c1f01186c8f05fb8b24e1fc507d8
      with:
        renderEngine: 'helm'
        helmChart: ${{ env.CHART_PATH }}
        overrideFiles: ${{ env.CHART_OVERRIDE_PATH }}
        overrides: |
          replicas:2
        helm-version: 'latest'
      id: bake

    - name: Deploys application
    - uses: Azure/k8s-deploy@dd4bbd13a5abd2fc9ca8bdcb8aee152bb718fa78
      with:
        manifests: ${{ steps.bake.outputs.manifestsBundle }}
        images: |
          ${{ env.AZURE_CONTAINER_REGISTRY }}.azurecr.io/${{ env.PROJECT_NAME
}}:${{ github.sha }}
        imagepullsecrets: |
          ${{ env.PROJECT_NAME }}
```

Additional resources

The following resources may also be useful:

- For the original starter workflow, see `azure-kubernetes-service.yml` in the GitHub Actions `starter-workflows` repository.
- The actions used to in this workflow are the official Azure `Azure/login`, `Azure/aks-set-context`, `Azure/CLI`, `Azure/k8s-bake`, and `Azure/k8s-deploy` actions.
- For more examples of GitHub Action workflows that deploy to Azure, see the actions-workflow-samples repository.