10-minute Quick start:

GitHub in Actions for Kubernetes on Azure

GitHub Actions gives you the flexibility to build an automated software development lifecycle workflow. With GitHub Actions for Azure you can create workflows that you can set up in your repository to build, test, package, release and **deploy** to Azure.

With the following steps you will be able to deploy a Nodejs app on an Azure Kubernetes Service cluster using GitHub Actions for Azure: aka.ms/GHActions4AZ.

1. Navigate to the following repo <https://github.com/azooinmyluggage/AKS>
2. Create a new branch by using your name and date combination

A screenshot of a cell phone

Description automatically generated

1. Navigate to the Actions tab in the GitHub repository

A screenshot of a cell phone

Description automatically generated

1. Click on “I understand my workflow, go ahead and run them”
2. In the Actions tab click on “New workflow” and proceed to click on “Set up a workflow yourself”

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

1. Now copy paste the below Sample workflow which uses GitHub Actions to build container image and deploy a NodeJS app to an **Azure Kubernetes Service (AKS)** cluster

on: [push]

jobs:

build:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@master

# Connect to Azure Container registry (ACR)

- uses: azure/docker-login@v1

with:

login-server: demo.azurecr.io

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

password: ${{ secrets.REGISTRY\_PASSWORD }}

# Docker build and push to a Azure Container registry (ACR)

- run: |

docker build . -t demo.azurecr.io/k8sdemo:${{ github.sha }}

docker push demo.azurecr.io/k8sdemo:${{ github.sha }}

# Set the target AKS cluster.

- uses: azure/aks-set-context@v1

with:

creds: '${{ secrets.AZURE\_CREDENTIALS }}'

cluster-name: desattir

resource-group: desattir

# Create imagepullsecret for Azure Container registry (ACR)

- uses: azure/k8s-create-secret@v1

with:

container-registry-url: demo.azurecr.io

container-registry-username: ${{ secrets.REGISTRY\_USERNAME }}

container-registry-password: ${{ secrets.REGISTRY\_PASSWORD }}

secret-name: demo-k8s-secret

namespace: kubecondemo

# Deploy app to AKS

- uses: azure/k8s-deploy@v1

with:

manifests: |

manifests/deployment.yml

manifests/service.yml

images: |

demo.azurecr.io/k8sdemo:${{ github.sha }}

imagepullsecrets: |

demo-k8s-secret

namespace: kubecondemo

The workflow uses a Dockerfile in the repo to build the container image and push it to an Azure Container Registry.

Once Container CI is done the next set of actions create an [imagePullSecret](https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-private-registry/) and then use Kubernetes Manifest files to deploy the container image to an AKS cluster.

1. In the last line uncomment the namespace field and provide a name for the new namespace to which your app will be deployed. Now commit the workflow file
2. Navigate to the Repository Settings  Secrets tabA screenshot of a cell phone

   Description automatically generated

Review the [secrets](https://help.github.com/en/github/automating-your-workflow-with-github-actions/virtual-environments-for-github-actions) and their usage in the workflow file. All the secrets are pre-populated for the lab but you can create them (not required for the lab). The secrets are:

1. Azure Credential for connecting and deploying to an AKS cluster
2. Container registry credentials for connecting to ACR

For Azure credentials, you can create an Azure SPN by running the following command.

$ az ad sp create-for-rbac --name build-demo-k8s --scopes /subscriptions/avb1291-\*\*\*\*-46be-\*\*\*\*-70349146ddrr/resourceGroups/build-demo --role contributor --sdk-auth

In the sample workflow of this repo we are using Azure SPN but other options are also supported. You can also use Kubeconfig [az aks get-credentials](https://docs.microsoft.com/en-us/cli/azure/aks?view=azure-cli-latest#az-aks-get-credentials) or for any other cluster refer to [Kubernetes](https://kubernetes.io/docs/concepts/configuration/organize-cluster-access-kubeconfig/) documentation. The [azure/k8s-set-context](https://github.com/Azure/k8s-actions/tree/master/k8s-set-context) action supports Kubernetes service account as well.

* + Define a new secret in the [repository](https://github.com/bbq-beets/k8s/settings/secrets) >> “Add a new secret”
  + Paste the output of the 'az ad sp...' command into the secret value field and add your secret.

{

"clientId": "7\*\*\*\*\*\*\*\*\*8",

"clientSecret": "\*\*\*\*\*\*",

"subscriptionId": "qvb11291-\*\*\*\*-\*\*\*\*-\*\*\*\*-70349\*\*46dre8",

"tenantId": "72\*\*\*\*bf-\*\*\*\*-\*\*\*\*-\*\*\*\*-\*\*\*\*\*011db4\*",

"activeDirectoryEndpointUrl": "https://login.microsoftonline.com",

"resourceManagerEndpointUrl": "https://management.azure.com/",

"activeDirectoryGraphResourceId": "https://graph.windows.net/",

"sqlManagementEndpointUrl": "https://management.core.windows.net:8443/",

"galleryEndpointUrl": "https://gallery.azure.com/",

"managementEndpointUrl": "https://management.core.windows.net/"

}

Similarly get the username and password of your container registry and create secrets for them. For Azure Container registry refer to admin [account document](https://docs.microsoft.com/en-us/azure/container-registry/container-registry-authentication#admin-account) for username and password.

1. Navigate to the Actions tab and wait for the workflow run to complete. Once the run is complete, navigate to the last step and copy the service IP from the log and use that to launch your app in the browser

A screenshot of a cell phone

Description automatically generated