Deploy a Python App to an AKS Cluster Using Azure Pipelines

Introduction

You are responsible for deploying a Python app to AKS. You have the code and a pipeline template, and you must create a CI/CD pipeline in Azure DevOps.

1. Create an Azure DevOps Organization

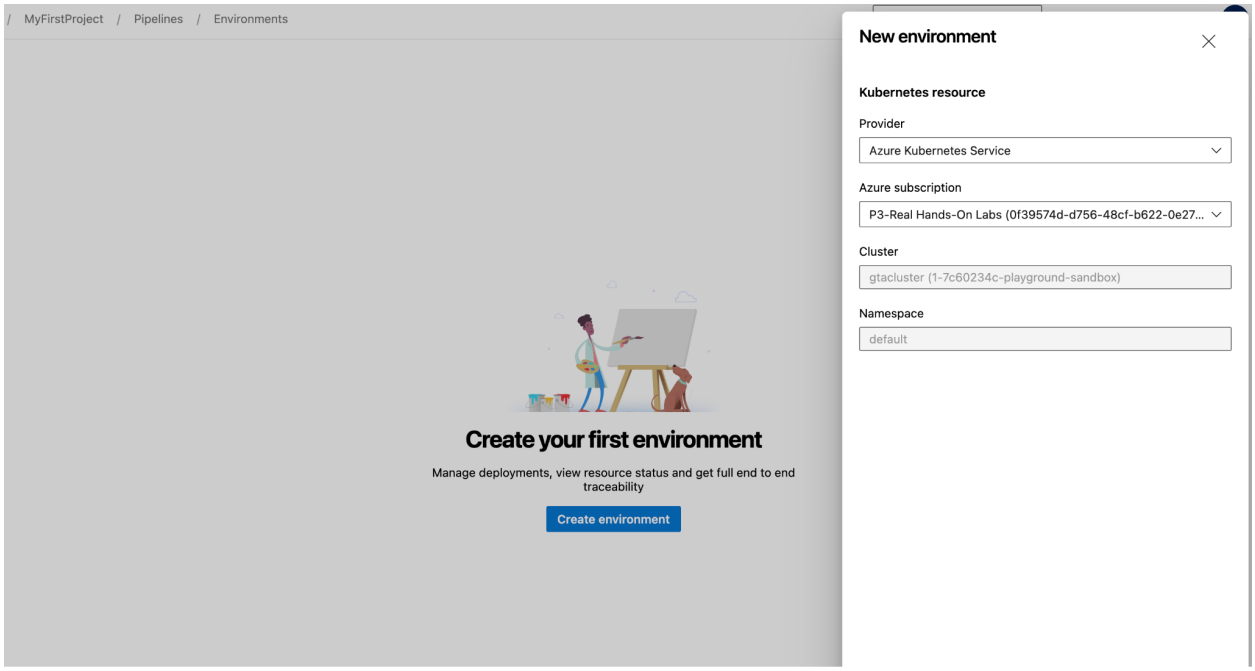
1. Navigate to Azure DevOps organizations using the search bar along the top menu bar.
2. Select My Azure DevOps Organizations.
3. Leave the pre-populated details and click Continue.
4. On the Get started with Azure DevOps page, click Create new organization.
5. Click Continue to accept the Terms of Service.
6. Verify the organization details auto-populate, then fill in the CAPTCHA and click Continue.
7. Your DevOps organization is created.
8. On the Create a project to get started page, fill in the project details:
    a. Project name: MyFirstProject
    b. Visibility: Private
9. Click Create project.

2. Import Code and Set Up the Environment

2.1. Create Your Environment

1. In the sidebar menu, select Repos.
2. In the Import a repository section, click Import.
3. In the Clone URL field, enter the following repository: https://github.com/hosniah/content-az400-lab-resources
4. Click Import.

5. After the import completes successfully, the Files page should open automatically.
6. At the top of the page, use the dropdown to switch the branch from DSC to aks.
7. You should now see the files you need for the lab.
8. In the sidebar menu, select Pipelines, then select Environments.
9. Click Create environment.



10. Fill in the environment details:
    a. Name: dev
    b. Resource: Kubernetes
11. Click Next.
12. Ensure the resource details auto-populate, then click Validate and create.

2.2. Create a Service Connection and an ACR

1. In the bottom left corner, click Project Settings.
2. This opens the settings in a new tab.
3. In the sidebar menu, select Service connections.
4. In the top right, click New service connection.
5. In the pane on the right, select Docker Registry, then click Next.

6. You'll need to gather some information from an Azure Container Registry (ACR) before you finish creating your service connection.
7. Navigate back to your Azure DevOps tab.
8. Verify which region your resources are in:
    a. Navigate to All resources using the hamburger menu in the top left corner.
    b. Review your resources and note which location they are in.
    c. You'll use this location to configure your service connection.
    d. Copy the name of the provided Kubernetes service to a separate file for later use.
9. Initialize Cloud Shell:
    a. Along the top menu bar, click the Cloud Shell icon (>_) to open the Cloud Shell terminal.
    b. In the Welcome to Azure Cloud Shell window, select Bash.
    c. In the storage window, select Show advanced settings.
    d. Ensure the Cloud Shell region field is set to the same location as your resources.
    e. In the Storage account field, enter a unique storage account name.
    f. In the File share field, enter fileshare.
    g. Click Create storage.
10. Your terminal may take a few minutes to initialize.
11. Get your resource group name:
    a. Navigate to Resource groups using the hamburger menu in the top left corner.
    b. Copy the provided resource group's name to a separate file along with your Kubernetes service name.
    c. You'll use it to configure your ACR and to verify your AKS cluster.
12. Note: You may need to minimize the Cloud Shell terminal for this.
13. Create an Azure Container Registry using the resource group name you copied:

    az acr create -g <RESOURCE_GROUP_NAME> --name <UNIQUE_ACR_NAME> --sku Premium --admin-enabled true

14. After the ACR is created, minimize the Cloud Shell terminal and navigate to All resources using the hamburger menu in the top left corner of the Azure portal.
15. Select your container registry name from your resource list.
    Note: You may need to refresh the resources if you don't see your ACR listed.
16. In the ACR's sidebar menu, select Access keys. You'll use these details for your service connection.
17. Add the New Docker Registry service connection details:
    a. Fill in the Docker Registry:
        i. From the Access keys page, copy the Login server.
        ii. Navigate to the service connections tab and paste the login server into the Docker Registry field using the format https://<LOGIN_SERVER>.
    b. Fill in the Docker ID:
        i. Navigate to the ACR tab and copy the Username.
        ii. Navigate to the service connections tab and paste the username into the Docker ID field.
    c. Fill in the Docker Password:
        i. Navigate to the ACR tab and copy either password.
        ii. Navigate to the service connections tab and paste the password into the Docker Password field.
    d. In the Service connection name field, enter ACR.
    e. Below Security, check the Grant access permission to all pipelines checkbox.
18. After the service connection details are complete, click Save.

3. Create the CI/CD Pipeline

3.1. Update the Deployment Image

1. From the service connections tab, use the sidebar menu to select Repos.
2. Select the manifests folder and open the vote.yml file.

3. In the top right, click Edit.
4. Update the deployment image:
5. Navigate to the ACR tab and copy the Login server again.
6. Navigate to the vote.yaml tab and replace the container image on line 59 with the Azure ACR DNS registry name, following the format <LOGIN_SERVER>/azure-vote-front:v1.
   Note: The ACR DNS name ends with azurecr.io.
7. In the top right, click Commit.
8. In the Commit pane, ensure the Branch name is set to aks, then click Commit.

3.2.Create the Pipeline

1. In the sidebar menu, select Pipelines.
2. Click Create Pipeline.
3. For the code location, select Azure Repos Git.
4. For the repository, select the MyFirstProject repo.
5. Select Existing Azure Pipelines YAML file.
6. Fill in the YAML file details:
   a. Branch: aks
   b. Path: /azure-pipelines.yml
7. Click Continue.
8. Update the azure-pipelines.yml code:
   a. Below trigger, replace - master with - aks.
   b. Navigate to the ACR tab and copy the Login server again.
   c. Navigate to the pipeline tab and replace the containerRegistry with your copied Azure ACR DNS registry name.
9. In the top right corner, click Save and run.
10. In the pane on the right, leave the default settings and click Save and run.
11. After the pipeline is triggered, select Build stage to monitor the build progress.
12. After the build has completed, select Deploy to dev.

13. Note the banner in the terminal indicating that the pipeline needs additional permissions to continue the deployment.
14. Click View to the right of the banner.
15. Click Permit to the right of the two requested permissions and confirm each selection. The deployment takes a few minutes to complete.

4. Access the AKS Cluster

1. Navigate to the ACR tab.
2. Clear the Cloud Shell terminal (reopen the terminal if you previously closed it):
3. clear
4. Access the AKS cluster using the resource group name and Kubernetes service name you saved earlier:
   az aks get-credentials -g <RESOURCE_GROUP_NAME> -n <KUBERNETES_SERVICE_CLUSTER_NAME>
5. List the current services in the namespace:
   kubectl get svc
6. Copy the external IP from the azure-vote-front details.
7. Open the IP address in a new browser tab to check connectivity to the app.