guide me to create a CD pipeline from azure devops portal. i want to deploy docker image from ACR to kubernetes. i have one kubernetes.yml file which is to be used to deploy the image from acr to kubernetes. guide me with the steps

**ChatGPT said:**

ChatGPT

Memory updated

Here’s a step-by-step guide to creating a Continuous Deployment (CD) pipeline in Azure DevOps for deploying a Docker image from Azure Container Registry (ACR) to Kubernetes using a Kubernetes YAML file.

**Prerequisites**

1. **Azure DevOps Project** - You need an Azure DevOps project set up.
2. **Azure Kubernetes Service (AKS)** - A running AKS cluster.
3. **Azure Container Registry (ACR)** - Your Docker image should be pushed to ACR.
4. **Service Connection** - A service connection to AKS in Azure DevOps.
5. **Kubernetes YAML File** - Ensure you have a valid Kubernetes YAML file for deployment.

**Step 1: Create a Service Connection for AKS**

1. Go to **Azure DevOps Portal**.
2. Navigate to **Project Settings** > **Service Connections**.
3. Click on **New Service Connection** > **Kubernetes**.
4. Select **Azure subscription** and then choose your **AKS cluster**.
5. Click **Next**, provide a name for the connection, and then **Save**.

**Step 2: Create a Release Pipeline**

1. Navigate to **Pipelines** > **Releases**.
2. Click on **New pipeline** > **Empty job**.
3. Name your release pipeline and select the **Artifact** source:
   * Select **Azure Container Registry** as the artifact source.
   * Choose the correct container registry and repository.
   * Enable **Continuous Deployment Trigger**.

**Step 3: Configure the Release Stage**

1. Click on **Stage 1** (rename it if needed).
2. Click on **Agent Job** and **Add a task**:
   * Select **Deploy to Kubernetes**.
3. Configure the **Kubernetes deployment task**:
   * **Kubernetes service connection**: Select the service connection created earlier.
   * **Namespace**: Use the desired namespace (e.g., default).
   * **Manifest file path**: Select your kubernetes.yml file.
   * **Image name**: Provide the image reference from ACR.
   * **Tag**: Use $(Build.BuildId) or latest depending on your versioning strategy.

**Step 4: Set Up Environment Variables (Optional)**

If your YAML file contains placeholders, you can replace them dynamically by using **Pipeline Variables**.

**Step 5: Save and Create a Release**

1. Click **Save**.
2. Click **Create Release** to test the deployment.
3. Monitor the deployment logs to ensure the image is pulled from ACR and deployed to AKS.

**Final Notes**

* If the image isn't updating in the cluster, make sure the Kubernetes deployment YAML uses the correct imagePullPolicy: Always.
* Ensure your Kubernetes cluster has permissions to pull from ACR.
* Use Helm if you want a more structured and versioned deployment process.

Would you like help with automating rollbacks or setting up monitoring?

4o

**You said:**

one example for kubernetes.yml file

**ChatGPT said:**

ChatGPT

Here’s a basic example of a kubernetes.yml file to deploy your Docker image from **Azure Container Registry (ACR)** to **Azure Kubernetes Service (AKS)**.

**Example kubernetes.yml**

yaml

CopyEdit

apiVersion: apps/v1

kind: Deployment

metadata:

name: my-app

labels:

app: my-app

spec:

replicas: 2 # Number of pods

selector:

matchLabels:

app: my-app

template:

metadata:

labels:

app: my-app

spec:

containers:

- name: my-app-container

image: myacr.azurecr.io/my-app:latest # Replace with your ACR and image name

ports:

- containerPort: 80

env:

- name: ENVIRONMENT

value: "production"

imagePullSecrets:

- name: acr-secret # Ensure you have created this Kubernetes secret

---

apiVersion: v1

kind: Service

metadata:

name: my-app-service

spec:

selector:

app: my-app

ports:

- protocol: TCP

port: 80

targetPort: 80

type: LoadBalancer # Change to ClusterIP if using an ingress controller

**Key Points**

* **Replace myacr.azurecr.io/my-app:latest** with your actual **ACR URL and image name**.
* **replicas: 2** - Specifies the number of pods.
* **imagePullSecrets: acr-secret** - Ensures Kubernetes can pull images from ACR (if authentication is required).
* **Service Type:**
  + LoadBalancer - Exposes the app externally.
  + ClusterIP - Used for internal communication (e.g., with an ingress controller).

**Steps to Apply This Deployment in AKS**

1. Make sure your ACR authentication is set up in Kubernetes:

sh

CopyEdit

kubectl create secret docker-registry acr-secret \

--docker-server=<your-acr-name>.azurecr.io \

--docker-username=<your-acr-username> \

--docker-password=<your-acr-password> \

--namespace default

1. Apply the deployment using:

sh

CopyEdit

kubectl apply -f kubernetes.yml

This YAML file can be used in your **Azure DevOps Release Pipeline** to deploy automatically. Let me know if you need modifications or further explanations!

4o

**You said:**

Manifest file path: Select your kubernetes.yml file. for this one, where should we keep the yml file in the project folder? because when i try to browse folders, im not able to find the file. because the browse folder only opens the build file from the CI pipeline

**ChatGPT said:**

ChatGPT

Since the **release pipeline** in Azure DevOps only has access to artifacts from the **CI pipeline**, you need to ensure that your kubernetes.yml file is included as part of the **CI build pipeline**.

**Steps to Include kubernetes.yml in the Build Artifact**

1. **Place the kubernetes.yml file** in your source repository, ideally in a folder like:

bash

CopyEdit

/manifests/kubernetes.yml

or

bash

CopyEdit

/infra/kubernetes.yml

1. **Modify your Azure DevOps CI Pipeline (azure-pipelines.yml) to Publish the YAML file**

Add the following task at the end of your **CI pipeline (azure-pipelines.yml)**:

yaml

CopyEdit

- task: PublishBuildArtifacts@1

displayName: "Publish Kubernetes Manifest"

inputs:

pathToPublish: 'manifests' # Change to the actual folder path where kubernetes.yml is stored

artifactName: 'drop' # This makes it available in the release pipeline

1. **Verify the Artifacts in the CI Pipeline**
   * After running the CI pipeline, go to **Pipelines** > **Runs**.
   * Click on the latest run.
   * Under **Artifacts**, you should see a drop folder containing the kubernetes.yml file.

**Using the YAML File in the Release Pipeline**

1. In your **Release Pipeline**, go to **Artifact** and select the drop artifact from the **CI pipeline**.
2. In the **Deploy to Kubernetes task**, for **Manifest file path**, click **Browse** and navigate to:

shell

CopyEdit

$(System.DefaultWorkingDirectory)/drop/kubernetes.yml

* + If you published it inside a subfolder like /manifests, the path would be:

shell

CopyEdit

$(System.DefaultWorkingDirectory)/drop/manifests/kubernetes.yml

**Final Notes**

* The key reason you **couldn't find the YAML file** was that it wasn’t included in the **Build artifacts** from the CI pipeline.
* Adding the PublishBuildArtifacts task ensures the YAML is available in the **Release pipeline**.

Now, your pipeline should be able to deploy the YAML file correctly! Let me know if you need more help.

4o

**You said:**