**1. AWS CodeCommit: Set up the Repository**

**Step 1: Create a Repository**

1. Open the **AWS Management Console** and go to **CodeCommit**.
2. Click **Create Repository** and name it (e.g., MyPythonAppRepo).
3. Note the **Clone URL** of the repository.

**Step 2: Clone the Repository Locally**

Run the following in your terminal:

bash

Copy code

git clone <CodeCommit-HTTPS-URL>

cd MyPythonAppRepo

**Step 3: Add Your Python Application Code**

1. Create a simple Python application, e.g., app.py:

python

Copy code

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def home():

return "Hello, World! This is a CI/CD demo!"

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host="0.0.0.0", port=80)

1. Add a requirements.txt file:

makefile

Copy code

flask==2.3.2

1. Add a shell script for deployment, deploy.sh:

bash

Copy code

#!/bin/bash

sudo apt update

sudo apt install -y python3-pip

pip3 install -r requirements.txt

python3 app.py &

**Step 4: Push the Code to CodeCommit**

bash

Copy code

git add .

git commit -m "Initial commit"

git push origin main

**2. AWS CodeBuild: Set up Build Process**

**Step 1: Create a Buildspec File**

Add a buildspec.yml file to the root of your project:

yaml

Copy code

version: 0.2

phases:

install:

runtime-versions:

python: 3.9

build:

commands:

- echo "Build step"

- pip install -r requirements.txt

artifacts:

files:

- '\*\*/\*'

discard-paths: yes

**Step 2: Create a CodeBuild Project**

1. Navigate to **CodeBuild** in the AWS Console and click **Create build project**.
2. Name the project (e.g., MyPythonAppBuild).
3. Select **CodeCommit** as the source provider and choose your repository.
4. Under **Environment**, select:
   * Managed image: Ubuntu.
   * Runtime: Standard.
   * Image: aws/codebuild/standard:6.0.
5. Ensure the **buildspec.yml** is auto-detected or provide its path.
6. Save the project.

**3. AWS CodeDeploy: Set up Deployment Process**

**Step 1: Create an IAM Role for EC2**

1. Create a role with **AmazonEC2RoleforAWSCodeDeploy** policy.
2. Attach this role to your EC2 instance.

**Step 2: Install CodeDeploy Agent on EC2**

SSH into your EC2 instance and run:

bash

Copy code

sudo yum update -y

sudo yum install ruby -y

sudo yum install wget -y

cd /home/ec2-user

wget https://aws-codedeploy-${AWS\_REGION}.s3.${AWS\_REGION}.amazonaws.com/latest/install

chmod +x ./install

sudo ./install auto

sudo service codedeploy-agent start

**Step 3: Create an appspec.yml File**

Add the following to your project:

yaml

Copy code

version: 0.0

os: linux

files:

- source: .

destination: /home/ec2-user/MyPythonApp

hooks:

AfterInstall:

- location: deploy.sh

timeout: 300

runas: ec2-user

**Step 4: Create a CodeDeploy Application and Deployment Group**

1. Go to **CodeDeploy** in the AWS Console.
2. Create an application (e.g., MyPythonAppDeployment).
3. Create a deployment group:
   * Environment configuration: EC2 instances.
   * Select your instance via tag or Auto Scaling group.
   * Attach the service role created earlier.

**4. AWS CodePipeline: Set up the CI/CD Pipeline**

**Step 1: Create a CodePipeline**

1. Navigate to **CodePipeline** and click **Create pipeline**.
2. Name the pipeline (e.g., MyPythonAppPipeline).
3. Choose **New Service Role**.

**Step 2: Add Stages**

1. **Source Stage**:
   * Source provider: CodeCommit.
   * Repository: Select your repository.
   * Branch: main.
2. **Build Stage**:
   * Build provider: CodeBuild.
   * Project: Select your build project.
3. **Deploy Stage**:
   * Deploy provider: CodeDeploy.
   * Application name: Select your CodeDeploy application.
   * Deployment group: Select your deployment group.

**Step 3: Review and Create**

Click **Create pipeline**. It will automatically trigger the pipeline.

**5. Test the Setup**

1. Push a new change to the repository:

bash

Copy code

echo "print('Updated')" >> app.py

git add .

git commit -m "Testing CI/CD pipeline"

git push origin main

1. Watch the pipeline execute in **CodePipeline**.
2. Access your application via the public IP of your EC2 instance.