Step-by-Step AWS CodeDeploy & CodePipeline Setup using Dev EC2 and Test EC2

# Overview

This guide walks through the complete step-by-step process of setting up AWS CodeDeploy and CodePipeline using two separate EC2 instances:  
  
- Dev EC2: Used for code development, zipping, and uploading to S3.  
- Test EC2: Used as the web server where the application is deployed via CodeDeploy.  
  
This approach mirrors real-world Dev/Test architecture and ensures better separation of concerns.

## Phase 1: IAM Configuration

### Step 1: Create IAM Roles

1. Create IAM Role `s3-ec2-full` for EC2 with permission: AmazonS3FullAccess  
2. Create IAM Role `cdrole` for CodeDeploy with permission: AWSCodeDeployRole

## Phase 2: EC2 Instances

### Step 2: Launch Dev EC2 (Developer)

- Launch Amazon Linux 2 instance  
- No need to install CodeDeploy agent  
- Install AWS CLI if not present

### Step 3: Launch Test EC2 (Web Server)

- Launch another Amazon Linux 2 instance  
- Tag it with Key: AppName, Value: SampleApp  
- Attach IAM Role: s3-ec2-full  
- Open Port 80 in security group

## Phase 3: Install CodeDeploy Agent (on Test EC2 only)

### Step 4: SSH into Test EC2 and run:

sudo yum update -y  
sudo yum install ruby -y  
sudo yum install wget -y  
cd /home/ec2-user  
wget https://aws-codedeploy-us-east-1.s3.amazonaws.com/latest/install  
chmod +x ./install  
sudo ./install auto  
sudo service codedeploy-agent status

## Phase 4: Code Development on Dev EC2

### Step 5: Prepare App Code

mkdir -p /root/deploy\_dir/sampleapp/scripts  
cd /root/deploy\_dir/sampleapp  
  
# Create index.html  
echo "<html><h2>Sample App Version 1</h2></html>" > index.html  
  
# Create appspec.yml  
cat > appspec.yml <<EOL  
version: 0.0  
os: linux  
files:  
 - source: /index.html  
 destination: /var/www/html/  
hooks:  
 BeforeInstall:  
 - location: scripts/httpd\_install.sh  
 timeout: 300  
 runas: root  
 - location: scripts/httpd\_start.sh  
 timeout: 300  
 runas: root  
 ApplicationStop:  
 - location: scripts/httpd\_stop.sh  
 timeout: 300  
 runas: root  
EOL  
  
# Create deployment scripts  
echo -e "#!/bin/bash\nyum install -y httpd" > scripts/httpd\_install.sh  
echo -e "#!/bin/bash\nsystemctl start httpd\nsystemctl enable httpd" > scripts/httpd\_start.sh  
echo -e "#!/bin/bash\nsystemctl stop httpd\nsystemctl disable httpd" > scripts/httpd\_stop.sh  
chmod 755 scripts/\*  
  
# Zip the application  
cd /root/deploy\_dir/sampleapp  
zip -r ../sampleapp.zip .

## Phase 5: Upload to S3 and Setup CodeDeploy

### Step 6: Upload Code to S3

aws s3 mb s3://gir-sampleapp  
aws s3 cp /root/deploy\_dir/sampleapp.zip s3://gir-sampleapp

### Step 7: Create CodeDeploy Application

aws deploy create-application --application-name sampleapp

### Step 8: Create Deployment Group (via Console)

- Application name: sampleapp  
- Deployment group name: mygrp  
- Service role: cdrole  
- EC2 tag key: AppName, value: SampleApp  
- Environment: EC2  
- Disable load balancing

## Phase 6: Deployment & Testing

### Step 9: Create Deployment (via Console)

- Application: sampleapp  
- Deployment group: mygrp  
- Revision type: S3  
- S3 Location: gir-sampleapp/sampleapp.zip

### Step 10: Test Deployment

- Open the public IP of Test EC2 in a browser  
- Confirm: 'Sample App Version 1' is visible

## Phase 7: CodePipeline Automation

### Step 11: Prepare Version 2 Code

cp -r /root/deploy\_dir /root/deploy\_dir2  
echo "<html><h2>Sample App Version 2</h2></html>" > /root/deploy\_dir2/sampleapp/index.html  
cd /root/deploy\_dir2/sampleapp  
zip -r ../sampleapp.zip .

### Step 12: Create CodePipeline (via Console)

- Source: S3 (gir-sampleapp)  
- Skip build stage  
- Deploy provider: CodeDeploy (sampleapp, mygrp)

### Step 13: Upload Version 2 to Trigger Pipeline

aws s3 cp /root/deploy\_dir2/sampleapp.zip s3://gir-sampleapp

### Step 14: Verify Auto Deployment

- Refresh browser for Test EC2  
- Confirm: 'Sample App Version 2' is shown