Hands-on AWS CICD Pipeline for Beginners (No Coding Required)

By following this comprehensive guide, you should be able to successfully deploy your Spring Boot application to an AWS EC2 instance using CodePipeline.

This guide walks you through a practical introduction to building a simple CI/CD pipeline on AWS without needing prior knowledge of Git commands or the AWS CLI. We will leverage the user-friendly AWS Management Console for the entire process.

Services Covered:

- **AWS IAM:** Create user and roles
- **AWS S3:** Save build
- **AWS EC2:** Deploy and run app
- **AWS CodeCommit:** A secure Git repository service for version control.
- **AWS CodeBuild:** A build service that compiles your code and runs tests (we'll use pre-built images).
- **AWS CodeDeploy:** A deployment service that automates software deployments.
- **AWS CodePipeline:** A service that orchestrates and manages the entire CI/CD workflow.

Prerequisites:

- An AWS account with appropriate permissions (creation is free tier eligible).

Step 1: Create IAM Roles

1. **EC2 Role:**
    - **Attach policies: `AmazonEC2RoleforAWSCodeDeploy` and `AmazonS3FullAccess`**
2. **CodeDeploy Role:**
    - **Attach policy: `AWSCodeDeployRole`**

Step 2: Launch an EC2 Instance

Create an EC2 instance with the Amazon Linux 2 AMI. Attach the previously created EC2 role. User Data Script for EC2 Instance:

```
#!/bin/bash
sudo yum update -y
sudo yum install -y ruby
cd /home/ec2-user
```

```
wget
https://aws-codedeploy-us-east-1.s3.us-east-1.amazonaws.com/late
st/install
chmod +x ./install
sudo ./install auto
sudo service codedeploy-agent start
```

Step 3: Setup CodeCommit Repository

1. Create IAM User:
    - Ensure the user has `HTTPS Git credentials`.
2. Create a CodeCommit Repository:
    - Name your repository.
3. Clone Repository Locally:

```
git clone <repo-clone-url>
cd <repo-name>
```

4. Create a Spring Boot Project and Add Necessary Files:

`buildspec.yml` for CodeBuild:

```
version: 0.2

phases:
  install:
    runtime-versions:
      java: 20
  pre_build:
    commands:
      - echo Installing dependencies
      - mvn -B dependency:resolve
  build:
    commands:
      - echo Building the project
      - mvn -B package
artifacts:
  files:
    - target/your-app.jar
```

```
    - appspec.yml
    - scripts/*
```

- Replace `your-app.jar` with your actual jar name.

5. Push Changes to CodeCommit:

```
git add .
git commit -m "Initial commit"
git push
```

Step 4: Create an S3 Bucket

Create an S3 bucket with versioning enabled to store build artifacts.

Step 5: Create a CodeBuild Project

1. Source Provider: CodeCommit
2. Artifacts Storage: S3 bucket created in the previous step.

Step 6: Create a CodeDeploy Application

1. Compute Platform: EC2/on-premises
2. Deployment Group:
    - Attach CodeDeploy IAM role.
    - Environment configuration: Select Amazon EC2 instance with a tag "Name" -> "EC2 instance name".

Step 7: Create AppSpec File

`appspec.yml` for CodeDeploy:

```
version: 0.0
os: linux
files:
  - source: /target/my-app.jar
    destination: /home/ec2-user/
hooks:
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
  BeforeInstall:
```

```
    - location: scripts/install_dependencies.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
      timeout: 300
      runas: root
  ValidateService:
    - location: scripts/validate_service.sh
      timeout: 300
      runas: root
```

- Replace `my-app.jar` with your actual jar name.

Step 8: Create Deployment Scripts

scripts/install_dependencies.sh:

```
#!/bin/bash
sudo yum update -y
sudo yum install -y java-17-amazon-corretto-devel
```

scripts/start_server.sh:

```
#!/bin/bash
sudo chmod -R 777 /home/ec2-user
java -jar /home/ec2-user/my-app.jar > /dev/null 2> /dev/null <
/dev/null &
```

scripts/stop_server.sh:

```
#!/bin/bash
echo "Stopping application"
pkill -f 'java -jar'
```

scripts/validate_service.sh:

```
#!/bin/bash
# Check if the application is running
if pgrep -f my-app.jar; then
  exit 0
```

```
else
  exit 1
fi
```

Step 9: Create a CodePipeline

Set up a pipeline to automate the deployment process:

1. Source Stage: CodeCommit repository.
2. Build Stage: CodeBuild project.
3. Deploy Stage: CodeDeploy application.

Summary

1. **IAM Roles:**
   - **EC2 Role: `AmazonEC2RoleforAWSCodeDeploy`, `AmazonS3FullAccess`**
   - **CodeDeploy Role: `AWSCodeDeployRole`**
2. **EC2 Instance Setup:**
   - **Launch with the user data script to install the CodeDeploy agent.**
3. **CodeCommit Repository:**
   - **Clone, create a project, add `buildspec.yml`, push changes.**
4. **S3 Bucket:**
   - **Create and enable versioning.**
5. **CodeBuild Project:**
   - **Set up with CodeCommit and S3.**
6. **CodeDeploy Application:**
   - **Configure with EC2 instances and IAM roles.**
7. **AppSpec and Deployment Scripts:**
   - **Create `appspec.yml` and necessary scripts.**
8. **CodePipeline:**
   - **Automate deployment from CodeCommit to CodeDeploy.**