CI/CD Pipeline with AWS EC2, Jenkins, GitHub, and Node.js

June 3, 2024

Arquitectura Orientada a Servicios
Universidad Politécnica de Chiapas

1 Introduction

This report outlines the steps to set up a Continuous Integration and Continuous Deployment (CI/CD) pipeline using AWS EC2 for Jenkins, GitHub for source control, and Node.js for the application. The Node.js project will include a simple API endpoint.

2 What is CI/CD?

CI/CD stands for Continuous Integration and Continuous Deployment, and it represents a set of practices aimed at improving the development and delivery process of software applications.

2.1 Continuous Integration (CI)

Continuous Integration is a development practice where developers integrate their code into a shared repository several times a day. Each integration is verified by an automated build and tests to detect errors quickly. The main objectives of CI are:

- **Detect and fix errors quickly**: By integrating code frequently, errors are detected and fixed more quickly.
- **Improve software quality**: Frequent automated testing ensures the software maintains its quality throughout the development process.
- **Reduce integration time**: CI helps avoid integration problems that can occur when developers work in isolation for long periods.

2.2 Continuous Delivery (CD)

Continuous Delivery is a practice where code changes are automatically built, tested, and prepared for release to production. Although deployment is not automatic, the system is always ready to deploy to production at any time. The main objectives of Continuous Delivery are:

- **Automate the release process**: Automate the process from code integration to deployment preparation.
- **Ensure deployable software**: Ensure the code is always in a state that can be deployed to production.

2.3 Continuous Deployment (CD)

Continuous Deployment goes a step further by automatically deploying every change that passes all stages of the production pipeline directly to production. The main objectives of Continuous Deployment are:

- **Fully automate the deployment pipeline**: From code integration to production deployment without manual intervention.
- **Reduce delivery time**: Ensure new features and bug fixes reach users as quickly as possible.

2.4 Benefits of CI/CD

- **Faster development speed**: Facilitates rapid and frequent delivery of new features and improvements.
- **Higher software quality**: Automated tests and builds ensure code changes are constantly validated.
- **Reduced risks**: Frequent integrations and deployments allow changes to be made in small iterations, reducing the risk of major issues.

2.5 CI/CD Workflow

1. **Development**: Developers write code and integrate it into the shared repository.
2. **Continuous Integration (CI)**: The code is automatically built and tested to validate changes.
3. **Continuous Delivery (CD)**: Changes that pass the tests are prepared for deployment.
4. **Continuous Deployment (CD)**: Prepared changes are automatically deployed to production.

3 Prerequisites

Before starting, ensure you have the following:

- An AWS account
- A GitHub account
- Basic knowledge of Node.js and Git

4 Setup AWS EC2 Instance

1. Log in to your AWS account and navigate to the EC2 dashboard.
2. Launch a new EC2 instance with the following configurations:
   - **AMI**: Amazon Linux 2 AMI
   - **Instance Type**: t2.micro (sufficient for this tutorial)
   - **Security Group**: Allow SSH (port 22) and HTTP (port 8080)
3. Connect to your instance using SSH.
4. Update the instance and install Java:

```
    sudo yum update -y
    sudo amazon-linux-extras install java-openjdk11 -y

```

5 Install Jenkins

1. Add the Jenkins repository and import the GPG key:

```
    sudo wget -O /etc/yum.repos.d/jenkins.repo \
        https://pkg.jenkins.io/redhat-stable/jenkins.repo
    sudo rpm --import https://pkg.jenkins.io/redhat-stable/
    jenkins.io.key

```

2. Install Jenkins:

```
    sudo yum install jenkins -y

```

3. Start Jenkins:

```
    sudo systemctl start jenkins
    sudo systemctl enable jenkins

```

4. Open port 8080 in the security group to access Jenkins via a web browser.

6 Setup GitHub Repository

1. Create a new repository on GitHub.
2. Initialize the repository with a simple Node.js project:

```
mkdir node-hello-world
cd node-hello-world
npm init -y
npm install express --save
```

3. Create an 'index.js' file with the following content:

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/api/v1/welcome', (req, res) => {
  res.send('Hello, World!');
});

app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

4. Push the code to the GitHub repository:

```
git init
git remote add origin https://github.com/yourusername/node-hello-world.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

7 Configure Jenkins

1. Access Jenkins at `http://your-ec2-instance-ip:8080`.
2. Complete the initial setup and install suggested plugins.
3. Create a new job:
   - Choose **Freestyle project** and name it **Node.js Hello World**.
4. Configure the job:
   - **Source Code Management**: Select **Git** and provide your repository URL.
   - **Build Triggers**: Check **GitHub hook trigger for GITScm polling**.

- **Build Environment**: Add **Provide Node npm bin/ folder to PATH**.
- **Build**: Add build step **Execute shell** and add the following commands:

```
        npm install
        node index.js

```

5. Save the job configuration.

8 Setup GitHub Webhook

1. Navigate to your GitHub repository settings.
2. Under **Webhooks**, click **Add webhook**.
3. Enter the payload URL as `http://your-ec2-instance-ip:8080/github-webhook/`.
4. Set the content type to **application/json**.
5. Select **Just the push event**.
6. Click **Add webhook**.

9 Testing the CI/CD Pipeline

1. Make a change to your Node.js project, e.g., modify the welcome message.
2. Commit and push the change to GitHub.
3. Jenkins should automatically trigger a build and deploy the updated application.
4. Verify that the endpoint `http://your-ec2-instance-ip:3000/api/v1/welcome` displays the updated message.

10 Conclusion

In this report, we have set up a CI/CD pipeline using AWS EC2, Jenkins, GitHub, and Node.js. This pipeline automates the process of integration and deployment, ensuring that new changes are quickly and reliably delivered to the production environment.