Comprehensive Guide to Setting Up Jenkins CI/CD with Docker, SonarQube, and ArgoCD on Ubuntu

This document provides a step-by-step guide for setting up a Jenkins CI/CD pipeline with integrated tools such as Docker, SonarQube, and ArgoCD on an Ubuntu server. The guide includes setting up an EC2 instance, installing necessary tools, configuring Jenkins, integrating GitHub, and more.

1. Create an EC2 Instance

- Launch an EC2 instance using Ubuntu OS and the `t2.large` instance type.
- Open all inbound traffic for the instance (for demo purposes) through security groups. In production, configure the security rules based on your requirements.

2. Connect to the EC2 Instance

SSH into the EC2 instance using your terminal:

```
ssh -i <your-key.pem> ubuntu@<your-public-ip>
```

3. Install Java

Jenkins requires Java, so install OpenJDK 17:

```
sudo apt update
sudo apt install openjdk-17-jre
```

Verify the Java installation:

```
java -version
```

4. Install Jenkins

4.1 Add Jenkins Repository and Install

1. Import the Jenkins key:

```
curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo
tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

2. Add the Jenkins repository to the sources list:

```
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]
https://pkg.jenkins.io/debian binary/ | sudo tee
/etc/apt/sources.list.d/jenkins.list > /dev/null
```

3. Update the package list:

```
sudo apt-get update
```

4. Install Jenkins:

```
sudo apt-get install jenkins
```

4.2 Check Jenkins Process

After installation, check if Jenkins is running:

```
ps -ef | grep jenkins
```

4.3 Open Jenkins Web Interface

Access Jenkins via the public IP address and port 8080 (e.g., `http://<your-public-ip>:8080`).

- Get the Jenkins unlock password:

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

- Use the password to unlock Jenkins, then install the recommended plugins.
- Set up an admin username and password.

4.4 Create and Configure a Jenkins Pipeline Job

1. **Create a New Job**:
    - Select **Pipeline** as the job type.
    - Set up the GitHub repository URL and configure the Jenkinsfile path.
2. **Configure GitHub Integration**:
    - Instead of writing the Jenkinsfile directly in Jenkins, integrate with GitHub.
    - Set up GitHub credentials in Jenkins, allowing it to access your repository.

4.5 Install Docker Plugin in Jenkins

- Install the Docker plugin through the **Jenkins Plugin Manager**.

5. Install SonarQube for Static Code Analysis

5.1 Install SonarQube on Ubuntu Server

1. Switch to root user:

```
sudo su -
```

2. Add a user for SonarQube:

```
adduser sonarqube
```

3. Download and install SonarQube:

```
wget
https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.4.
0.54424.zip
```

```
unzip sonarqube-9.4.0.54424.zip
chmod -R 755 sonarqube-9.4.0.54424
chown -R sonarqube:sonarqube sonarqube-9.4.0.54424
```

4. Start SonarQube:

```
cd sonarqube-9.4.0.54424/bin/linux-x86-64/
./sonar.sh start
```

5. Access SonarQube via `http://<your-public-ip>:9000`. The default login credentials are:
    - Username: **admin**
    - Password: **admin**

5.2 Configure SonarQube in Jenkins

- Generate a SonarQube token from the SonarQube dashboard under **My Account > Security**.
- Go to Jenkins and add the token as a **Secret Text** in **Manage Jenkins > Manage Credentials**.

6. Install Docker on Ubuntu Server

1. Update package list:

```
sudo apt update
```

2. Install Docker:

```
sudo apt install docker.io
```

3. Add Jenkins and Ubuntu users to the Docker group:

```
usermod -aG docker jenkins
usermod -aG docker ubuntu
```

4. Restart Docker:

```
systemctl restart docker
```

7. Set Up Maven and Docker in Jenkins Pipeline

1. Install Maven and Docker tools on the Jenkins server.
2. Configure the pipeline to use Maven and Docker for building the project.

8. Set Up ArgoCD for Kubernetes Deployment

8.1 Install ArgoCD Using Operators

1. Install ArgoCD Operator using the OperatorHub:

```
    apiVersion: argoproj.io/v1alpha1
kind: ArgoCD
metadata:
  name: example-argocd
spec: {}
```

2. Apply the configuration:

```
kubectl apply -f argo-cd.yaml
```

3. Check running pods:

```
kubectl get pods -n operators
```

8.2 Configure ArgoCD with GitHub

1. Sync the GitHub repository with ArgoCD.
2. Set up the GitHub repository URL, cluster name, and other necessary configurations.

8.3 Access ArgoCD Web Interface

- Access ArgoCD via `http://<your-public-ip>:8080` and log in using the default credentials.
- Change the password upon first login.

9. Complete CI/CD Pipeline in Jenkins

With everything set up, you can now integrate Docker, SonarQube, ArgoCD, and GitHub into your Jenkins pipeline. Here's an example of a Jenkinsfile for the pipeline:

groovy
Copy code

```
pipeline {
  agent {
    docker {
      image 'ramagiri/maven-docker-agent:v1'
      args '--user root -v /var/run/docker.sock:/var/run/docker.sock'
// mount Docker socket
    }
  }
  stages {
    stage('Checkout') {
      steps {
        sh 'echo passed'
      }
    }
    stage('Build and Test') {
      steps {
        sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app && mvn
clean package'
      }
    }
    stage('Static Code Analysis') {
      environment {
        SONAR_URL = "http://<your-sonarqube-ip>:9000"
      }
      steps {
        withCredentials([string(credentialsId: 'sonarqube', variable:
'SONAR_AUTH_TOKEN')]) {
```

```
          sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app &&
mvn sonar:sonar -Dsonar.login=$SONAR_AUTH_TOKEN
-Dsonar.host.url=${SONAR_URL}'
        }
      }
    }
    stage('Build and Push Docker Image') {
      environment {
        DOCKER_IMAGE =
"jithendarramagiri1998/ultimate-cicd:${env.BUILD_NUMBER}"
        REGISTRY_CREDENTIALS = credentials('docker-cred')
      }
      steps {
        script {
          sh 'cd java-maven-sonar-argocd-helm-k8s/spring-boot-app &&
docker build -t ${DOCKER_IMAGE} .'
          def dockerImage = docker.image("${DOCKER_IMAGE}")
          docker.withRegistry('https://index.docker.io/v1/',
"docker-cred") {
            dockerImage.push()
          }
        }
      }
    }
    stage('Update Deployment File') {
      environment {
        GIT_REPO_NAME = "Jenkins-Zero-To-Hero"
        GIT_USER_NAME = "************"
      }
      steps {
        withCredentials([string(credentialsId: 'github', variable:
'GITHUB_TOKEN')]) {
          sh '''
            git config user.email "jithendar*******gmail.com"
            git config user.name "***************"
            BUILD_NUMBER=${BUILD_NUMBER}
```

```
            sed -i "s/replaceImageTag/${BUILD_NUMBER}/g"
java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests/deployment.
yml
            git add
java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests/deployment.
yml
            git commit -m "Update deployment image to version
${BUILD_NUMBER}"
            git push
https://${GITHUB_TOKEN}@github.com/${GIT_USER_NAME}/${GIT_REPO_NAME}
HEAD:main
          '''
        }
      }
    }
  }
}
```

9.1 Conclusion

Once the pipeline runs successfully, you will have a fully automated CI/CD pipeline integrating Jenkins, SonarQube, Docker, and ArgoCD. This setup allows for seamless code analysis, containerized builds, and Kubernetes-based deployments.