perplexity

https://www.youtube.com/watch?v=361bfIvXMBI

refer the above vedio and make step by step detailed guide on that to save it in pdf

Now I'll create a comprehensive, beginner-friendly step-by-step guide based on the video tutorial.

Complete Jenkins CI/CD Pipeline Setup Guide with SonarQube, Docker, and GitHub Webhooks on AWS

Overview

This comprehensive guide will walk you through setting up a complete CI/CD pipeline from scratch. The tutorial is designed for beginners and covers every step in detail. You'll learn to create an automated system where pushing code to GitHub automatically triggers Jenkins to build, test with SonarQube, and deploy using Docker.

What You'll Build

Pipeline: GitHub Repository → Jenkins (via webhook) → SonarQube Analysis → Docker Deployment → Browser Access

Infrastructure: Three AWS EC2 instances working together:

- **Jenkins Server**: Orchestrates the entire pipeline
- **SonarQube Server**: Analyzes code quality and security
- **Docker Server**: Hosts the deployed application

Estimated Time: 2-3 hours for complete setup

Prerequisites

Before starting, ensure you have:

- AWS Account with appropriate permissions[1] [2]
- GitHub account
- Basic understanding of Linux commands
- SSH client (Terminal, Git Bash, or PuTTY)
- A simple HTML/CSS website project (we'll use a basic template)

Phase 1: AWS Infrastructure Setup

Step 1.1: Create SSH Key Pair

1. **Login to AWS Console**
   - Navigate to AWS Console
   - Sign in to your account
   - Search for "EC2" and click on the service
2. **Create Key Pair**[3]
   - In the EC2 dashboard, click on "Key Pairs" in the left navigation
   - Click "Create key pair"
   - Name: `jenkins-ssh-key`
   - Key pair type: RSA
   - Private key file format: `.pem`
   - Click "Create key pair"
   - **Important**: Save the downloaded `.pem` file securely - you'll need it for SSH access

Step 1.2: Launch Jenkins Server

1. **Launch EC2 Instance**[1] [2]
   - Click "Launch Instance"
   - **Name**: `Jenkins-Server`
   - **AMI**: Ubuntu Server 22.04 LTS (Free tier eligible)
   - **Instance type**: t2.medium (2 vCPU, 4GB RAM)
   - **Key pair**: Select `jenkins-ssh-key` (created above)
2. **Configure Security Group**
   - Click "Edit" next to Network settings
   - Security group name: `Jenkins-SG`
   - **Inbound rules**:
     - SSH (22): Source - My IP
     - Custom TCP (8080): Source - Anywhere (0.0.0.0/0)
     - HTTP (80): Source - Anywhere (0.0.0.0/0)
   - Click "Launch instance"

Step 1.3: Launch SonarQube Server

1. **Create Second Instance**
   - Click "Launch Instance"
   - **Name**: `SonarQube-Server`
   - **AMI**: Ubuntu Server 22.04 LTS
   - **Instance type**: t2.medium (SonarQube requires at least 2GB RAM)[4] [5]
   - **Key pair**: Select `jenkins-ssh-key`
2. **Configure Security Group**
   - Security group name: `SonarQube-SG`
   - **Inbound rules**:
     - SSH (22): Source - My IP
     - Custom TCP (9000): Source - Anywhere (0.0.0.0/0)
   - Click "Launch instance"

Step 1.4: Launch Docker Server

1. **Create Third Instance**
   - Click "Launch Instance"
   - **Name**: `Docker-Server`
   - **AMI**: Ubuntu Server 22.04 LTS
   - **Instance type**: t2.medium
   - **Key pair**: Select `jenkins-ssh-key`
2. **Configure Security Group**
   - Security group name: `Docker-SG`
   - **Inbound rules**:
     - SSH (22): Source - My IP
     - Custom TCP (8085): Source - Anywhere (0.0.0.0/0)
   - Click "Launch instance"

Step 1.5: Verify All Instances

1. **Check Instance Status**
   - Go to EC2 → Instances
   - Ensure all three instances show "Running" status
   - Note down the public IP addresses of all instances

Phase 2: Jenkins Server Configuration

Step 2.1: Connect to Jenkins Server

1. **Open Terminal/Command Prompt**

```
# Navigate to downloads folder where your key is stored
cd Downloads

# Set appropriate permissions for the key file
chmod 400 jenkins-ssh-key.pem

# Connect to Jenkins server (replace with your actual IP)
ssh -i jenkins-ssh-key.pem ubuntu@YOUR_JENKINS_SERVER_IP
```

2. **Verify Connection**
   - You should see a welcome message and Ubuntu prompt
   - If you get permission errors, double-check the key file permissions

Step 2.2: Install Java (Required for Jenkins)

```
# Update package repositories
sudo apt update

# Install Java 11 (required for Jenkins)
sudo apt install openjdk-11-jre -y

# Verify Java installation
java -version
```

Expected Output: You should see Java version information[1] [2]

Step 2.3: Install Jenkins

1. **Add Jenkins Repository**[1] [2]

```
# Download Jenkins key
curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee /usr/share/ke

# Add Jenkins repository
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/d

# Update package list
sudo apt update

# Install Jenkins
sudo apt install jenkins -y
```

2. **Start Jenkins Service**

```
# Enable Jenkins to start on boot
sudo systemctl enable jenkins

# Start Jenkins service
sudo systemctl start jenkins

# Check Jenkins status
sudo systemctl status jenkins
```

Step 2.4: Initial Jenkins Setup

1. **Get Initial Admin Password**

   ```
   # Retrieve the initial admin password
   sudo cat /var/lib/jenkins/secrets/initialAdminPassword
   ```

   Important: Copy this password - you'll need it shortly

2. **Access Jenkins Web Interface**
   - Open browser and go to: `http://YOUR_JENKINS_SERVER_IP:8080`
   - Paste the admin password from above
   - Click "Continue"
3. **Install Suggested Plugins**
   - Click "Install suggested plugins"
   - Wait for installation to complete (5-10 minutes)
4. **Create Admin User**[6]
   - **Username**: admin
   - **Password**: Choose a secure password
   - **Full name**: Jenkins Admin
   - **Email**: your-email@domain.com
   - Click "Save and Continue"
5. **Instance Configuration**
   - Jenkins URL should be auto-populated
   - Click "Save and Finish"
   - Click "Start using Jenkins"

Step 2.5: Change Server Hostname (Optional but Recommended)

```
# Set hostname for easy identification
sudo hostnamectl set-hostname Jenkins
# Reload bash
/bin/bash
```

Phase 3: SonarQube Server Setup

Step 3.1: Connect to SonarQube Server

1. **Open New Terminal Tab**

```
# Connect to SonarQube server
ssh -i jenkins-ssh-key.pem ubuntu@YOUR_SONARQUBE_SERVER_IP
```

2. **Set Hostname**

```
sudo hostnamectl set-hostname SonarQube
/bin/bash
```

Step 3.2: Install Java 17 (Required for SonarQube)

```
# Update repositories
sudo apt update

# Install Java 17 (SonarQube requires Java 17+)
sudo apt install openjdk-17-jdk -y

# Verify installation
java -version
```

Step 3.3: Download and Install SonarQube

1. **Download SonarQube**[4] [7]

```
# Download SonarQube Community Edition
wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-10.4.1.88267.z

# Install unzip utility
sudo apt install unzip -y

# Extract SonarQube
unzip sonarqube-10.4.1.88267.zip

# Move to /opt directory
sudo mv sonarqube-10.4.1.88267 /opt/sonarqube
```

2. **Create SonarQube User**[4]

```
# Create sonar user and group
sudo useradd -r -s /bin/false sonar

# Change ownership
sudo chown -R sonar:sonar /opt/sonarqube
```

Step 3.4: Start SonarQube

1. **Start SonarQube Service**

```
# Navigate to SonarQube bin directory
cd /opt/sonarqube/bin/linux-x86-64/

# Start SonarQube (use console to see logs)
sudo ./sonar.sh console
```

2. **Verify SonarQube is Running**
   - Wait for startup to complete (look for "SonarQube is operational" message)
   - Open browser: `http://YOUR_SONARQUBE_SERVER_IP:9000`
   - You should see SonarQube login page
3. **Initial SonarQube Login**[5]
   - **Username**: admin
   - **Password**: admin
   - You'll be prompted to change the password
   - Set new password: Choose a secure password and remember it

Step 3.5: Create SonarQube Project and Token

1. **Create New Project**
   - Click "Create Project" → "Manually"
   - **Project display name**: `Website-Scanner`
   - **Project key**: `website-scanner`
   - Click "Set Up"
2. **Configure Analysis Method**
   - Select "With Jenkins"
   - Choose "GitHub" as DevOps platform
   - Click "Configure Analysis"
   - Select "Other" (for HTML/CSS projects)
3. **Generate Authentication Token**[8] [9]
   - Click on admin user (top right) → "My Account"
   - Go to "Security" tab
   - **Generate Token**:
     - Name: `jenkins-token`
     - Type: "Global Analysis Token"
     - Expires in: 30 days

- Click "Generate"
- **Important**: Copy and save this token securely

Phase 4: Docker Server Installation

Step 4.1: Connect to Docker Server

```
# Open new terminal tab and connect
ssh -i jenkins-ssh-key.pem ubuntu@YOUR_DOCKER_SERVER_IP

# Set hostname
sudo hostnamectl set-hostname Docker
/bin/bash
```

Step 4.2: Install Docker

1. **Update System and Install Prerequisites** [10] [11]

```
# Update package index
sudo apt update

# Install required packages
sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

2. **Add Docker Repository** [11] [12]

```
# Add Docker's official GPG key
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/

# Add Docker repository
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https

# Update package index
sudo apt update
```

3. **Install Docker**

```
# Install Docker CE
sudo apt install docker-ce docker-ce-cli containerd.io -y

# Start and enable Docker
sudo systemctl start docker
sudo systemctl enable docker

# Verify Docker installation
sudo docker --version
```

Step 4.3: Configure Docker for Non-Root Access

```
# Add current user to docker group
sudo usermod -aG docker ubuntu

# Apply group changes
newgrp docker

# Test Docker without sudo
docker ps
```

Phase 5: GitHub Repository and Webhook Configuration

Step 5.1: Create GitHub Repository

1. **Create New Repository**
   - Go to GitHub
   - Click "New repository"
   - **Repository name**: `jenkins-cicd-project`
   - **Description**: "Jenkins CI/CD Pipeline Project"
   - **Visibility**: Public
   - **Initialize**: Check "Add a README file"
   - Click "Create repository"

Step 5.2: Add Project Files

1. **Create Website Files**
   - Click "Add file" → "Create new file"
   - **Filename**: `index.html`

```
<!DOCTYPE html>
<html>
<head>
    <title>Jenkins CI/CD Demo</title>
    <style>
        body { font-family: Arial, sans-serif; text-align: center; padding: 50px; }
        h1 { color: #2c3e50; }
        .container { max-width: 600px; margin: 0 auto; }
    </style>
</head>
<body>
    <div class="container">
        <h1>Welcome to Jenkins CI/CD Pipeline</h1>
        <p>This website was automatically deployed using Jenkins, SonarQube, and Dock
        <p>Pipeline Status: Successfully Deployed</p>
    </div>
```

```
</body>
</html>
```

- Click "Commit new file"

2. **Create Dockerfile**
   - Click "Add file" → "Create new file"
   - **Filename**: `Dockerfile`

```
FROM nginx:alpine
COPY . /usr/share/nginx/html
EXPOSE 80
```

   - Click "Commit new file"

Step 5.3: Configure GitHub Webhook

1. **Access Repository Settings**[13] [14]
   - Go to your repository
   - Click "Settings" tab
   - Click "Webhooks" in left sidebar
   - Click "Add webhook"
2. **Configure Webhook**[13] [14]
   - **Payload URL**: `http://YOUR_JENKINS_SERVER_IP:8080/github-webhook/`
   - **Content type**: application/json
   - **Which events**: Select "Just the push event"
   - **Active**: Checked
   - Click "Add webhook"

Phase 6: Jenkins Pipeline Creation and Integration

Step 6.1: Install Required Jenkins Plugins

1. **Access Jenkins Plugin Manager**
   - Go to Jenkins dashboard
   - Click "Manage Jenkins" → "Manage Plugins"
   - Click "Available" tab
2. **Install Plugins**[8] [9]
   - Search and install:
     - "SonarQube Scanner"
     - "SSH Pipeline Steps"

        - "GitHub Integration"
    - Check "Restart Jenkins when installation is complete"
    - Wait for Jenkins to restart

Step 6.2: Configure SonarQube Integration

1. **Add SonarQube Server**[8] [9]
    - Go to "Manage Jenkins" → "Configure System"
    - Scroll to "SonarQube servers" section
    - Click "Add SonarQube"
    - **Name**: `SonarQube-Server`
    - **Server URL**: `http://YOUR_SONARQUBE_SERVER_IP:9000`
2. **Add SonarQube Token**[8]
    - Click "Add" next to Server authentication token
    - **Kind**: Secret text
    - **Secret**: Paste your SonarQube token
    - **ID**: `sonarqube-token`
    - Click "Add"
    - Select the token from dropdown
    - Click "Save"
3. **Configure SonarQube Scanner**[9]
    - Go to "Manage Jenkins" → "Global Tool Configuration"
    - Scroll to "SonarQube Scanner"
    - Click "Add SonarQube Scanner"
    - **Name**: `SonarScanner`
    - Check "Install automatically"
    - Click "Save"

Step 6.3: Configure SSH for Remote Deployment

1. **Generate SSH Key on Jenkins Server**

```
# Switch to Jenkins user
sudo su - jenkins

# Generate SSH key pair
ssh-keygen -t rsa -b 4096
# Press Enter for all prompts (no passphrase)

# Copy public key
cat ~/.ssh/id_rsa.pub
```

2. **Configure Docker Server for SSH Access**

```
# On Docker server, edit SSH config
sudo nano /etc/ssh/sshd_config

# Find and modify these lines:
PasswordAuthentication yes
PubkeyAuthentication yes

# Save and restart SSH
sudo systemctl restart sshd

# Set password for ubuntu user
sudo passwd ubuntu
# Enter a secure password
```

3. **Test SSH Connection**

```
# From Jenkins server (as jenkins user)
ssh ubuntu@YOUR_DOCKER_SERVER_IP
# Enter the password you set
# Type 'exit' to return to Jenkins server

# Copy SSH key to Docker server
ssh-copy-id ubuntu@YOUR_DOCKER_SERVER_IP
# Enter password when prompted
```

Step 6.4: Configure Jenkins for Remote Deployment

1. **Install Publish Over SSH Plugin**
   - Go to "Manage Jenkins" → "Manage Plugins"
   - Install "Publish Over SSH" plugin
2. **Configure SSH Servers** [15] [16]
   - Go to "Manage Jenkins" → "Configure System"
   - Scroll to "Publish over SSH"
   - Click "Add" SSH Server
   - **Name**: `Docker-Server`
   - **Hostname**: `YOUR_DOCKER_SERVER_IP`
   - **Username**: `ubuntu`
   - **Remote Directory**: `/home/ubuntu`
   - Click "Test Configuration" to verify
   - Click "Save"

Step 6.5: Create Jenkins Pipeline Job

1. **Create New Job**
   - Click "New Item"
   - **Item name**: `Website-CICD-Pipeline`
   - Select "Freestyle project"
   - Click "OK"
2. **Configure Source Code Management**
   - Select "Git"
   - **Repository URL**: Your GitHub repository HTTPS URL
   - **Branch**: `*/main`
3. **Configure Build Triggers**[13]
   - Check "GitHub hook trigger for GITScm polling"
4. **Add Build Steps**
   Step 1: SonarQube Analysis[9] [17]
   - Click "Add build step" → "Execute SonarQube Scanner"
   - **Analysis properties**:

```
sonar.projectKey=website-scanner
sonar.projectName=Website Scanner
sonar.projectVersion=1.0
sonar.sources=.
sonar.host.url=http://YOUR_SONARQUBE_SERVER_IP:9000
sonar.login=YOUR_SONARQUBE_TOKEN
```

   Step 2: Deploy to Docker[15]
   - Click "Add build step" → "Send files or execute commands over SSH"
   - **SSH Server**: Select "Docker-Server"
   - **Source files**: `**/*`
   - **Remote directory**: `website`
   - **Exec command**:

```
cd /home/ubuntu/website
docker build -t my-website .
docker stop website-container || true
docker rm website-container || true
docker run -d -p 8085:80 --name website-container my-website
```

5. **Save Configuration**
   - Click "Save"

Phase 7: Testing and Verification

Step 7.1: Test Manual Build

1. **Run First Build**
   - Go to your pipeline job
   - Click "Build Now"
   - Monitor build progress in "Console Output"
2. **Verify SonarQube Analysis**
   - Go to SonarQube dashboard
   - Check if your project appears with analysis results
3. **Verify Docker Deployment**
   - Open browser: `http://YOUR_DOCKER_SERVER_IP:8085`
   - You should see your website

Step 7.2: Test Webhook Automation

1. **Make Code Change**
   - Go to your GitHub repository
   - Edit `index.html`
   - Change the title or add new content
   - Commit changes
2. **Verify Automatic Build**
   - Check Jenkins dashboard
   - Build should trigger automatically
   - Monitor build progress
3. **Verify Updated Deployment**
   - Refresh your website
   - Changes should be reflected

Step 7.3: Monitor Pipeline Health

1. **Check Jenkins Build History**
   - Green builds = Success
   - Red builds = Failure (check console output)
2. **Monitor SonarQube Quality Gates**
   - Check code quality metrics
   - Review any detected issues

3. **Verify Docker Container Status**

```
# On Docker server
docker ps
# Should show running website-container
```

Phase 8: Troubleshooting Guide

Common Issues and Solutions

Issue 1: Jenkins Can't Connect to GitHub

- **Solution**: Verify webhook URL includes `/github-webhook/`
- Check Jenkins URL is accessible from internet
- Verify GitHub webhook shows green checkmarks

Issue 2: SonarQube Analysis Fails[9]

- **Solution**: Check SonarQube server is running
- Verify authentication token is correct
- Ensure SonarQube plugin is properly configured

Issue 3: Docker Deployment Fails[15]

- **Solution**: Check SSH connectivity between Jenkins and Docker server
- Verify Docker is running and accessible
- Check port 8085 is open in security group

Issue 4: Website Not Accessible

- **Solution**: Verify security group allows traffic on port 8085
- Check if Docker container is running: `docker ps`
- Review Docker container logs: `docker logs website-container`

Issue 5: Build Fails After Git Push

- **Solution**: Check webhook configuration
- Verify Jenkins has proper Git repository access
- Review Jenkins build console output for errors

Performance Optimization Tips

1. **Resource Management**
    - Monitor EC2 instance CPU and memory usage
    - Consider upgrading instance types if needed
    - Use CloudWatch for monitoring

2. **Security Best Practices** [18]
    - Regularly update Jenkins plugins
    - Use strong passwords and tokens
    - Implement proper access controls
    - Keep SonarQube and Docker updated
3. **Pipeline Improvements**
    - Add email notifications for build failures
    - Implement parallel builds for larger projects
    - Use Jenkins Pipeline as Code (Jenkinsfile)
    - Add automated testing stages

Maintenance Tasks

Weekly:

- Check Jenkins plugin updates
- Review SonarQube quality gate results
- Monitor system resource usage

Monthly:

- Update server packages: `sudo apt update && sudo apt upgrade`
- Review and rotate authentication tokens
- Check Docker image sizes and cleanup

Quarterly:

- Backup Jenkins configuration
- Review and update security groups
- Plan for infrastructure scaling if needed

Conclusion

Congratulations! You've successfully built a complete CI/CD pipeline that:

1. **Automatically triggers** when code is pushed to GitHub
2. **Analyzes code quality** using SonarQube
3. **Deploys applications** using Docker containers
4. **Provides continuous monitoring** and feedback

This setup forms the foundation for modern DevOps practices and can be extended with additional features like:

- Automated testing frameworks

- Multiple environment deployments (dev, staging, production)
- Advanced monitoring and alerting
- Infrastructure as Code using Terraform

The skills you've learned here are directly applicable to enterprise-level DevOps environments and will serve as an excellent foundation for your DevOps journey.

Next Steps:

- Explore Jenkins Pipeline as Code (Jenkinsfile)
- Learn about Kubernetes for container orchestration
- Implement Infrastructure as Code with Terraform
- Add automated testing to your pipeline

Remember to shut down your AWS instances when not in use to avoid unnecessary charges, and always follow security best practices in production environments.

⁂

1. https://trainwithshubham.blog/how-to-easily-install-jenkins-on-aws-ec2-ubuntu/
2. https://www.geeksforgeeks.org/devops/install-and-configure-jenkins-in-aws-ec2/
3. https://www.jenkins.io/doc/tutorials/tutorial-for-installing-jenkins-on-AWS/
4. https://github.com/jae1choi/sonaqueue-installation-guide
5. https://docs.vultr.com/how-to-use-sonarqube-on-ubuntu-22-04-lts
6. https://www.cloudthat.com/resources/blog/how-to-configure-jenkins-on-ec2-instance
7. https://docs.sonarsource.com/sonarqube-server/10.8/setup-and-upgrade/install-the-server/introduction/
8. https://docs.sonarsource.com/sonarqube-server/10.8/analyzing-source-code/ci-integration/jenkins-integration/global-setup/
9. https://www.geeksforgeeks.org/devops/how-to-integrate-sonarqube-with-jenkins/
10. https://www.linkedin.com/pulse/docker-installation-aws-ec2-instance-nandhini-saravanan
11. https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/install-docker.html
12. https://tutorialsdojo.com/how-to-install-docker-on-ubuntu-using-amazon-ec2/
13. https://www.hatica.io/blog/github-jenkins-webhooks-integration/
14. https://dzone.com/articles/adding-a-github-webhook-in-your-jenkins-pipeline
15. https://stackoverflow.com/questions/55260365/execute-commands-on-remote-host-in-a-jenkinsfile
16. https://www.reddit.com/r/devops/comments/htpcr0/open_a_remote_ssh_connection_from_jenkins_server/
17. https://stackoverflow.com/questions/62876686/how-to-setup-sonarqube-scanner-in-jenkins-for-a-js-ts-react-project
18. https://docs.cloudbees.com/docs/cloudbees-ci-kb/latest/client-and-managed-controllers/github-webhook-configuration