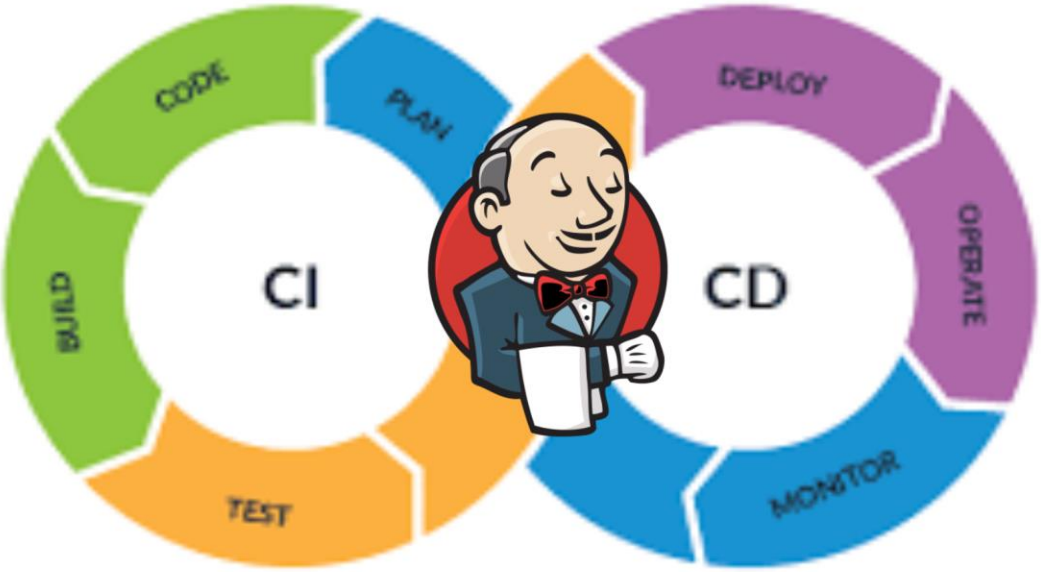

Day 25 Task: Complete Jenkins CI/CD Project - Continued with Documentation

Today, we're focusing on completing the Jenkins CI/CD project from Day 24 by adding thorough documentation. This step is crucial for understanding your project better and helping others who might use or contribute to it.

Table of contents

Docker Setup
Jenkins Setup
GitHub Webhook Integration
Conclusion

Introduction

Completing your Jenkins CI/CD project with proper documentation is crucial for ensuring the project's usability and maintainability. This guide will help you document your Jenkins CI/CD project step-by-step, making it easier for others to understand and for you to reuse in the future. Additionally, we will set smaller, manageable goals to make the process more efficient and rewarding.

Steps for Jenkins CI/CD Project

Let's dive into the detailed documentation.

GitHub Setup

1. **Create a GitHub Account:**
    - Follow the steps in Creating an account on GitHub.
2. **Create a GitHub Repository:**
    - Refer to Create a GitHub Repository.
3. **Set Up Git Configuration:**

```
git config --global user.email "your email id"
git config --global user.name "Your name"
git config --global --list  # List all the git config properties
```

4. **Clone the Repository:**

```
git clone <Your repo URL>
```

5. **Initialize Git Repository:**

```
git init
```

6. **Add Files:**

```
git add <filename>
```

7. **Commit Changes:**

```
git commit -m "message"
```

8. **Push Changes:**

```
git push origin <branchname>
```

AWS Setup

9. **Create an AWS Account:**
    - Follow the steps given in Create an AWS Account.
10. **Launch an EC2 Instance:**
    - Refer to How to create EC2 Instance and connect an instance to a client server.

Software Installation

11. **Install Docker:**

```
sudo apt-get update && sudo apt-get install docker.io
```

12. **Install Docker Compose:**

```
sudo apt-get install docker-compose
```

13. **Install Java:**

```
sudo apt install openjdk-11-jre
```

14. **Add Jenkins Repository Key:**

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

15. **Add Jenkins Repository:**

```
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
```

16. **Install Jenkins:**

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

17. **Start Jenkins Service:**

```
sudo systemctl start jenkins
systemctl status jenkins  # Check Jenkins status
```

Docker Setup

18. **Add User to Docker Group:**

```
sudo usermod -aG docker $USER
```

19. **Verify User Group Addition:**

```
grep docker /etc/group
```

20. **Fix Docker Permission Errors:**

```
sudo chown $USER /var/run/docker.sock
sudo chmod 777 /var/run/docker.sock
```

21. **Create a Dockerfile:**
    - Refer to Creating a Dockerfile.

Example:

```
FROM python:3.9-slim-buster
WORKDIR /app
COPY . .
RUN pip install flask
EXPOSE 5000
CMD [ "python", "main.py"]
```

22. **Build Docker Image:**

```
docker build -t <appname> .
```

23. **List Docker Images:**

```
docker images
```

24. **Run Docker Container:**

```
docker run -d -p hostport:containerport <imagename>
```

25. **List Running Containers:**

```
docker ps
docker ps -a  # For all containers whether it is running or exited or stopped.
```

26. **Set Up Inbound Rules:**
    - Add specific ports to your instance for accessing your application via browser (refer to Setting up inbound rules).

Jenkins Setup

27. **Access Jenkins:**
    - Visit <Your_Public_IP:8080> in your browser.
28. **Unlock Jenkins:**

- Use the admin password found in the location specified in the Jenkins setup page.

29. **Set Up Jenkins:**
    - Follow instructions in Setting up Jenkins.
30. **Create a Freestyle Job:**
    - Refer to First Freestyle Job.

GitHub Webhook Integration

31. **Create a Webhook in GitHub:**
    - Refer to How to create a Webhook.
32. **Configure Webhook:**
    - Payload URL: `<Jenkins_URL:port/github-webhook/>`
    - Select Events: That should trigger the webhook.
    - Add Webhook: Click on the "Add webhook" button and refresh the page to confirm activation.
33. **Integrate Webhook with Jenkins Job:**
    - Follow the steps in Jenkins Freestyle Job with Webhook.
34. **Test Integration:**
    - Make changes to any file in your GitHub repo, commit the changes, and verify that Jenkins triggers the job automatically.
35. **Access Application:**
    - Use `<Your_Public_IP>:Port` specified in your application to verify deployment.

Conclusion

This detailed documentation covers the entire process of setting up a complete Jenkins CI/CD project using GitHub and Docker. By creating a Jenkins job for your CI/CD pipeline, you automate the process from source code management to deployment, ensuring consistent and reliable software delivery. Jenkins orchestrates these steps based on defined triggers, providing a streamlined and efficient workflow for development and deployment.

Key Takeaways:

- Documenting your project ensures long-term usability and aids collaboration.

- Breaking down tasks into smaller goals can make the process more manageable and rewarding.

By following these steps, you'll have a well-documented Jenkins CI/CD project that you can confidently add to your resume and showcase to potential employers.

Connect and Follow Me on Socials

LINKDIN | **GITHUB** |**TWITTER**