Setting Up a CI Pipeline

Lesson Plan

PW SKILLS

Setting up a Continuous Integration (CI) pipeline involves automating the process of integrating code changes, running tests, and potentially deploying applications. Below is a general guide to setting up a CI pipeline using common CI tools like Jenkins, Travis CI, or CircleCI.

1. Define Your Project's Requirements

- Version Control System (VCS): Ensure your project is under version control (e.g., Git) and hosted on a platform like GitHub, GitLab, or Bitbucket.
- Dependencies: List all dependencies your project requires to run (e.g., libraries, packages).
- Build and Test Commands: Define the commands needed to build and test your project.

2. Choose a CI Tool

- Jenkins: Ideal for complex, customizable pipelines with various plugins.
- Travis CI: Great for simple setups, especially with GitHub integration.
- CircleCI: Best for fast, efficient pipelines with advanced Docker support.

3. Setup the CI Tool

A. Jenkins

2. Install Jenkins:

- Download and install Jenkins on your local machine or server.
- Access Jenkins via **http://localhost:8080** after installation.

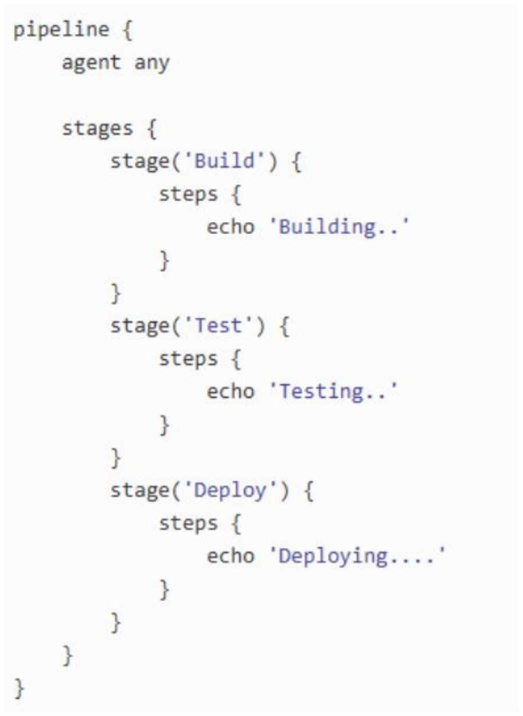
2. Create a New Pipeline:

- From the Jenkins dashboard, click on "**New Item**" and select "**Pipeline**."
- Name your pipeline and configure the source code repository (e.g., GitHub repository).

3. Configure Pipeline:

- Use the "**Pipeline**" section to define the stages of your CI process. This can be done via a **Jenkinsfile** or directly in the UI.

Example Jenkinsfile:



```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                echo 'Building..'
            }
        }
        stage('Test') {
            steps {
                echo 'Testing..'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying....'
            }
        }
    }
}
```

4. Trigger Builds:

- Set up triggers to run the pipeline automatically on code commits or pull requests. Jenkins supports various triggers, including polling the SCM and webhooks from GitHub.

5. Run and Monitor:

- Manually trigger a build to ensure everything works. Monitor the build logs and results directly in Jenkins.

B. Travis CI

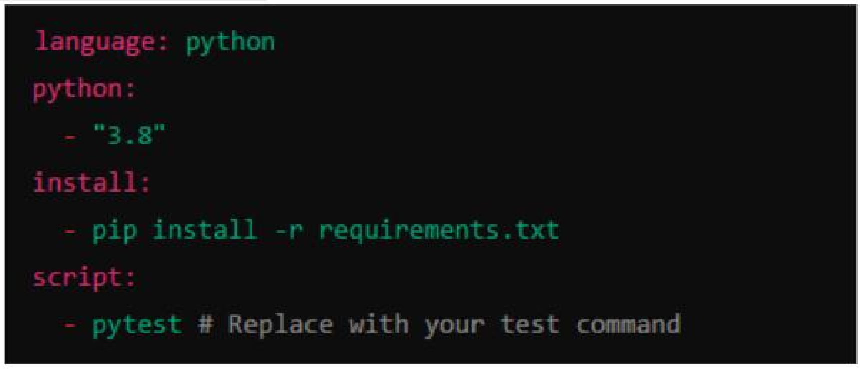
1. Sign Up and Link Repository:

- Sign up on the **Travis CI website** and link your GitHub repository.

2. Create a **.travis.yml** File:

- Add a **.travis.yml** file to the root of your repository to define the pipeline.

Example **.travis.yml** :



```
language: python
python:
  - "3.8"
install:
  - pip install -r requirements.txt
script:
  - pytest # Replace with your test command
```

3. Push Changes:

- Commit and push the .travis.yml file to your repository. Travis CI will automatically start building and testing your project based on this configuration.

4. Monitor Builds:

- View the build results on the Travis CI dashboard, and fix any issues that arise.

C. Circle CI

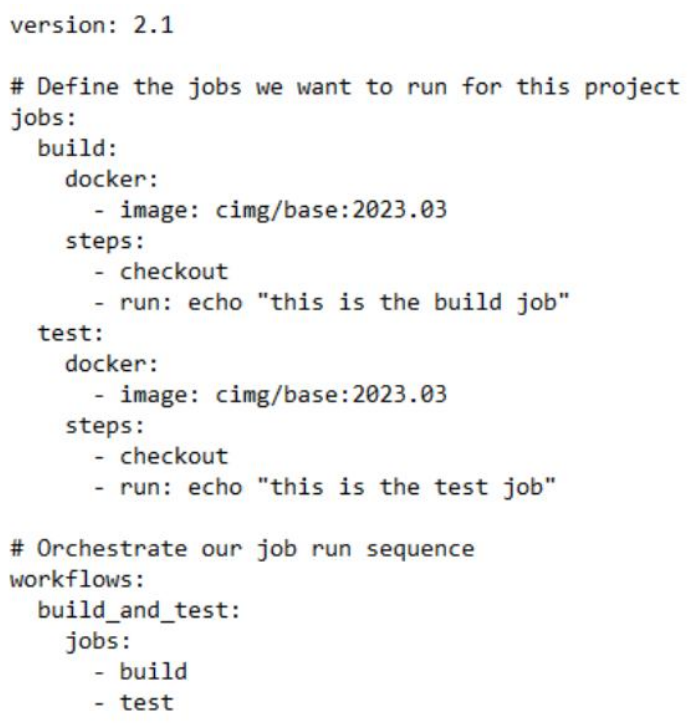
1. Sign Up and Link Repository:

- Sign up on the **CircleCI website** and link your VCS repository.

2. Create a **.circleci/config.yml** File:

- In your repository, create a **.circleci** directory and add a **config.yml** file.

- Example **config.yml**



```
version: 2.1

# Define the jobs we want to run for this project
jobs:
  build:
    docker:
      - image: cimg/base:2023.03
    steps:
      - checkout
      - run: echo "this is the build job"
  test:
    docker:
      - image: cimg/base:2023.03
    steps:
      - checkout
      - run: echo "this is the test job"

# Orchestrate our job run sequence
workflows:
  build_and_test:
    jobs:
      - build
      - test
```

3. Push Changes:

- Commit and push the config.yml file. CircleCI will automatically start the pipeline when changes are detected.

4. Monitor Builds:

- Use the CircleCI dashboard to view the build process and results.