COURSEWARE

Pipeline

Contents

Overview

Jenkins Pipeline (or simply "Pipeline" with a capital "P") is a **suite of plugins** which supports implementing and integrating *continuous delivery pipelines* into Jenkins.

Jenkinsfile

The definition of a Jenkins Pipeline is written into a text file (called a **Jenkinsfile**) which in turn can be committed to a project's **source control repository**.

Creating a Jenkinsfile and committing it to source control provides a *number of immediate benefits*:

- Automatically **creates a Pipeline build process** for all branches and pull requests.
- **Code review**/iteration on the Pipeline.
- **Audit trail** for the Pipeline.
- **Single source of truth** for the Pipeline, which can be viewed and edited by multiple members of the project.

While the syntax for defining a Pipeline, either in the web UI or with a Jenkinsfile is the same, it is generally considered best practice to define the Pipeline in a Jenkinsfile and check that in to source control.

Pipeline Concepts

Pipeline

A Pipeline is a **user-defined** model of a CD pipeline.

A Pipeline's code **defines your entire build process**, which typically includes *stages* for building an application, testing it and then delivering it.

Agent

The Agent section **specifies where the entire Pipeline, or a specific stage, will execute** in the Jenkins environment, depending on where the agent section is placed.

The section **must be defined at the top-level** inside the pipeline block, but stage-level usage is optional.

For example, you may wish to run one stage of the job inside a specific Docker container only. To do this, you would define the agent as `docker` in that particular stage.

Stage

A stage block **defines a conceptually distinct subset of tasks performed through the entire Pipeline** (e.g. "Build", "Test" and "Deploy" stages), which is used by many plugins to visualise or present Jenkins Pipeline status/progress.

Step

A **single task**. Fundamentally, a step tells Jenkins what to do at a particular point in time (or "step" in the process).

Example

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                //
            }
        }
        stage('Test') {
            steps {
                //
            }
        }
        stage('Deploy') {
            steps {
                //
            }
        }
    }
}
```

Tutorial

In this tutorial, we will run a simple Pipeline job in Jenkins:

1. Install and set up Jenkins, using this script:

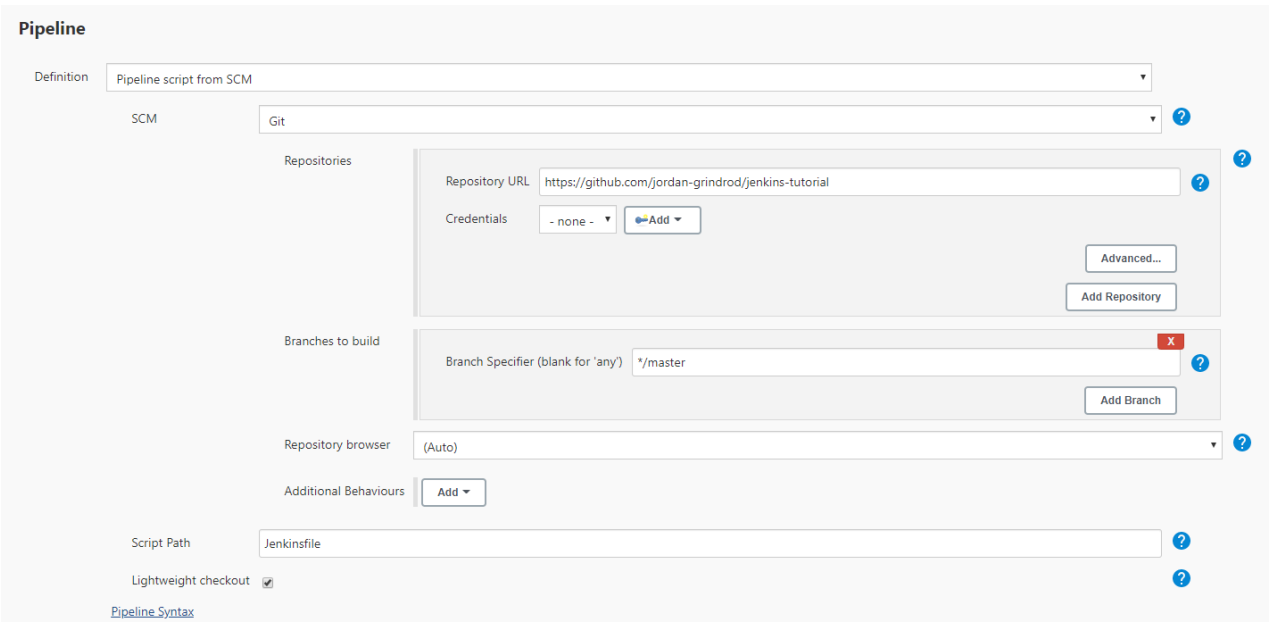
```
#!/bin/bash
if type apt > /dev/null; then
    pkg_mgr=apt
    java="openjdk-8-jre"
elif type yum /dev/null; then
    pkg_mgr=yum
    java="java"
fi
echo "updating and installing dependencies"
sudo ${pkg_mgr} update
sudo ${pkg_mgr} install -y ${java} wget git > /dev/null
echo "configuring jenkins user"
sudo useradd -m -s /bin/bash jenkins
echo "downloading latest jenkins WAR"
sudo su - jenkins -c "curl -L https://updates.jenkins-ci.org/latest/jenkins.war
--output jenkins.war"
echo "setting up jenkins service"
sudo tee /etc/systemd/system/jenkins.service << EOF > /dev/null
[Unit]
Description=Jenkins Server

[Service]
User=jenkins
WorkingDirectory=/home/jenkins
ExecStart=/usr/bin/java -jar /home/jenkins/jenkins.war

[Install]
WantedBy=multi-user.target
EOF
sudo systemctl daemon-reload
sudo systemctl enable jenkins
sudo systemctl restart jenkins
sudo su - jenkins << EOF
until [ -f .jenkins/secrets/initialAdminPassword ]; do
    sleep 1
    echo "waiting for initial admin password"
done
until [[ -n "\$(cat  .jenkins/secrets/initialAdminPassword)" ]]; do
    sleep 1
    echo "waiting for initial admin password"
done
echo "initial admin password: \$(cat .jenkins/secrets/initialAdminPassword)"
EOF
```

2. Go to Jenkins and click on `New Item`, on the left hand side, and then choose `Pipeline`. Give the job a name, such as "jenkins-tutorial".
3. Create a new Repository on your Version Control Provider - in this tutorial, we will be using GitHub - and call it "jenkins-tutorial".
4. Create a file in this repository, called `Jenkinsfile`.
5. In the Jenkinsfile, enter the following:

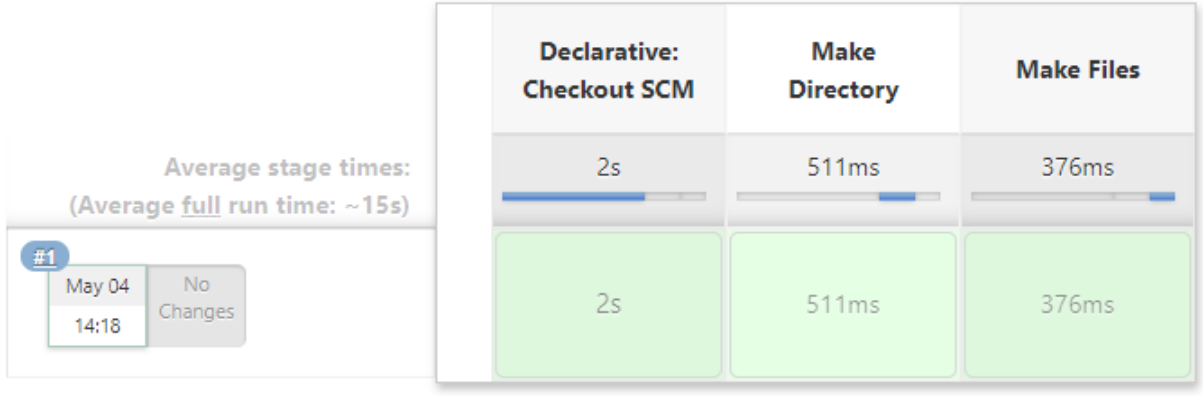
```
pipeline{
        agent any
        stages{
            stage('Make Directory'){
                steps{
                    sh "mkdir ~/jenkins-tutorial-test"
                }
            }
            stage('Make Files'){
                steps{
                    sh "touch ~/jenkins-tutorial-test/file1 ~/jenkins-tutorial-
test/file2"
                }
            }
        }
}
```

6. Go to your Jenkins instance, and change the `Pipeline` section to the below, making sure to use the URL for the repository you made in step `2` for

`Repository URL`:



7. Click Save
8. Click `Build Now`, which can be found on the left hand side
9. The job should run, showing the status at each stage:



10. Go to your Jenkins machine and run the following command:

```
sudo su - jenkins
ls -al jenkins-tutorial-test
```

You should see `file1` and `file2`, which have been created by the Jenkins Pipeline job!

Exercises

1. Create a Jenkinsfile
2. Declare 3 stages in the Jenkinsfile:
   - Clone https://gitlab.com/qacdevops/chaperootodo_client, making sure it doesn't already exist.
   - Install Docker and Docker-Compose.
   - Deploy the application, using `sudo docker-compose pull && sudo -E DB_PASSWORD=${DB_PASSWORD} docker-compose up -d`.
3. Build the Job
4. Navigate to the port 80 on the Jenkins machine - you should see the app up and running.

▶ Hint