Table of Contents

Jenkins Pipeline Project

Jenkins Build with Jenkinsfile

- Navigate to **New Item**, Provide a name for your new item and select **Pipeline** type.
- Enter below Pipeline code into the **Script text** area.

`Jenkinsfile`

```
pipeline {
    agent any
    parameters {
        choice(name: 'EnvironmentName', choices: ['dev', 'qa', 'test', 'prod'],
description: 'Enter the Environment to be used.')
    }
    stages {
        stage('Build') {
            steps {
                echo 'Building..'
                echo "Running ${env.BUILD_ID} on ${env.JENKINS_URL}"
            }
        }
        stage('Test') {
            steps {
                echo 'Testing..'
                echo "${params.EnvironmentName} is value retrieved!"
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying....'
            }
```

```
        }
    }
}
```

- Execute the Jenkins Pipeline and validate the stage and steps execution in the Build Execution Console Output.
- Add the Jenkinsfile Program Code into Github Repository, and use **Pipeline Script from SCM** and configure the Github Authentication to point the Jenkins Job to the Jenkinsfile code, stored in the Github Repository.

--

Jenkins pipeline-syntax

- Jenkins has a built-in **Snippet Generator** utility that is helpful for creating bits of code for individual steps, discovering new steps provided by plugins, or experimenting with different parameters for a particular step.
- The Snippet Generator is dynamically populated with a list of the steps available to the Jenkins instance. The number of steps available is dependent on the plugins installed which explicitly expose steps for use in Pipeline.
- To generate a **step snippet** with the **Snippet Generator**:
    - Navigate to the Pipeline Syntax link from a configured Pipeline, or at `${YOUR_JENKINS_URL}/pipeline-syntax`
    - Select the desired step in the Sample Step dropdown menu
    - Use the dynamically populated area below the Sample Step dropdown to configure the selected step.
    - Click **Generate Pipeline Script** to create a snippet of Pipeline which can be copied and pasted into a **Jenkinsfile** in the Pipeline.
- The code for a Jenkinsfile should be available in Github Repo
- Click the Add Source button, select git choose the type of repository you want to use and fill in the details.
- Click the Save button and watch your first Pipeline run!

--

Jenkinsfile Code Execution

- Execute all the Jenkinsfiles code from Github Repo.

Pipeline Syntax Reference

- pipeline-syntax

--

Configuring Credentials in Jenkinsfile

- Navigate to `Jenkins Home page > Credentials > System > Add Credentials`.
- Select Scope as `Global` **Global** - When credentials are to be added for a Pipeline project/item. **System** - When credentials are to be added for a Jenkins itself to interact with system administration functions., such as email authentication, agent connection, etc. This option applies the scope of the credential to a single object only.
- Types of credentials:
    - **Secret text** - a token such as an API token (e.g. a GitHub personal access token)
    - **Username and password** - which could be a colon separated string in the format username:password

--

Jenkins Docker

- Install a **Docker Pipeline** Plugin to have agent support in the Jenkinsfile
- Install Docker Daemon on your host.

```
sudo amazon-linux-extras install docker -y
sudo systemctl start docker
sudo systemctl status docker
sudo systemctl enable docker

sudo usermod -a -G docker jenkins
cat /etc/group | grep docker

# Add permissions to below file
sudo chmod 666 /var/run/docker.sock
```

- Add the below code for Jenkinsfile

```
pipeline {
    agent {
        docker { image 'python:3.8-slim' }
    }
    stages {
        stage('build') {
            steps {
                sh 'python --version'
                sh 'echo Hello Jenkins'
            }
        }
    }
}
```

- When the Pipeline executes, Jenkins will automatically start the specified container and execute the defined steps within it:

--

Jenkins Pipeline Image Creation-Assignment

- CI Executions like Docker image creation shell script, Infrastucture.
- Steps for Docker image creation Pipeline to store images in Docker Hub/ECR.
- Create Jenkins Credential for storing the Docker Hub User Login details.
- Keep all application Code with Dockerfile in the Git Repository.

```
app.py
requirements.txt
Dockerfile
scripts/build_push.sh
```

- Write Jenkinsfile to execute the above shell script in the Build Stage of the Pipeline Code.

--

Reference

- https://issues.jenkins.io/browse/JENKINS-66361
- https://www.jenkins.io/doc/book/pipeline/jenkinsfile/

Jenkins Github Webhook

- Integrate jenkins with github so CICD Pipeline is started/executed automatically when any commit is made to the Github repo.
- **Github Server Configuration in Jenkins**
  - Go to **Jenkins > Manage Jenkins > Under System Configuration select System > Add Github Server > Add a Name : jenkins-github-webhook-server and Enter API URL : http://public-ip:8080/github-webhook/**
    - Replace the public-ip with actual IP of Jenkins
- **Webhook Configuration in Github**
  - Lets add a webhook in Github to point to Jenkins URL
  - In **Github Repository > Go to Repository > Settings > Under Webhooks and Add webhook** > Specify **http://public-ip:8080/github-webhook/** as Payload URL.

--

- **Project Configuration in Jenkins**
  - Go to Jenkins Project, Under Configure, Select the **GitHub hook trigger for GITScm polling** checkbox under Build Triggers tab > `Save`.
- For Webhook to work, open port `8080` in security group for IP address of Github.
  - Visit githubs-ip-addresses

    - You can retrieve a list of GitHub's IP addresses from the meta API endpoint
    - IP address under the **hooks** can be used for whitelisting.
- Now if we make some changes/add commits to some file in Github Repository that is configured in webhook, this Jenkins Project will trigger automatically.
    - The Started by details for Jenkins Job will be Github Push Event.
- Jenkins receives a Github Payload similar to this Github Push Webhook Event Payload

--

> When Jenkins receives a GitHub push hook, GitHub Plugin checks to see whether the hook came from a GitHub repository which matches the Git repository defined in SCM/Git section of this job. If they match and this option is enabled, GitHub Plugin triggers a one-time polling on GITScm. When GITScm polls GitHub, it finds that there is a change and initiates a build.

Reference

- Reset Jenkins Admin Password from CLI