Jenkins CI/CD

Jenkins is a platform for creating Continuous Integration and Continuous delivery environment. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration, and continuous delivery. The system offers a wide range of freely available community plugins, Tools and native Jenkins workflows to orchestrate application development and deployment.

This page describes the setup of Jenkins As A Service(JaaS) instance to run the service's CI/CD pipeline. A pipeline is a collection of steps/stages executed in the Continuous Integration and Continuous Delivery (CI/CD) process.

Steps For Basic Jenkins Setup

1.Create A JaaS Instance

It has bunch of plugin already installed in the initial configuration. We can install the necessary plugins From the Jenkins Dashboard, go to Manage Jenkins ->Manage Plugins -> Available Plugins.
Install Maven Integration plugin, NodeJS Plugin and Cloud Foundry Plugin for the build and deployment process of QPA Application.

2.Creating a Jenkins job

To create a Jenkins job, log in to JaaS Instance and go to Jenkins Dashboard, then click on "New Item" to create a new job. On this page, enter the name of our job. It can be anything, but the best practice is to enter the job name the same as the project/repository name for which we are building the job.

3.Cloning Git/GitHub Repository in Jenkins

Once the job configuration is successful, click on "Source Code Management" (SCM) select Git and enter the Repository URL. Under the "Branches to Build" section, specify the default branch for which we want to trigger the pipeline. If our repository is a public repository, then don't need to set any credentials, else we need to mention our GitHub credentials.

4.Build the application& Executing Unit Tests

Now under Jenkins job, click on configure to continue further configuration. Go to Build Steps section, click on Add Build Steps, and select the required build step. For maven build select "Invoke top-level Maven targets" or for build using NodeJS select execute NodeJS script, depending on the build tool for the application and provide the build command.

5.Executing the Job& Verify Build Logs

Once we complete the above configuration, go to the created job in Jenkins job, and click on Build Now. Once we click on Build Now, build will start, which we can check in the "Build History" section. If our build is a *success*, we will see a *green* checkbox, beside our build number, else we will see a *red cross mark* if the build fails.

To check logs of the build, click on the build number, and on "Console Output", will see all the logs generated for this build.

6.Deploy the application& Archiving the Artifact

We use the Cloud Foundry Plugin available in Jenkins to to deploy the application. Under the job ->configuration->Post-build action->choose push to cloud foundry option and provide the cloud space,organization, target and credentials to deploy the application to Cloud Foundry. It will read the configuration from the manifest file of our application

Setup The WebHook in GitHub

It will be required to trigger a new build whenever a new push would be done to the repository and hence, it will be needed to configure Webhook in the GitHub Repository.so that whenever a push has been done the Github shall notify the same Jenkins server, sending a POST request. Go to the GitHub Repository settings and then add Webhooks, enter the Payload URL as Jenkins server IP/github-webhook/

In the above case it is https://nextgenqpa.jaas-gcp.cloud.sap.corp/github-webhook/

CI/CD for Backend Java Spring Boot application with Jenkins

Here Maven is used as the build tool and GitHub as the version control system.
Plugins needed: Maven Integration plugin, Git Plugin and Cloud Foundry Plugin.

Install Maven Plugin

Open Jenkins hit "Manage Jenkins" option on the left side of the Jaas Instance, go to the "Manage Plugins" option to install/remove/view the plugins. Under Manage plugins select "Available Plugins" and search for the "Maven Integration" plugin. Then do "Install without restart" button.

Configuring Maven

Since we are using maven as a build tool, we need to configure maven on Jenkins. To do this, go to manage Jenkins–>configure system. Under configure system, there is an option for maven. click add maven option, give it a name(arbitrary),check "Install automatically" and select any installation method, and click on *Save*.

We can also add multiple versions of maven by specifying a unique name for each one of them.

Install JDK

Our Jaas instance has java version 11,if we need a different version,we can install different JDKs. Go to dashboard–>Manage Jenkins–>JDK Installations–>Add Jdk –> install automatically–>extract *.zip/*.tar.gz and provide the URL for the required JDK.

The tar file of JDK 17 used for the new qpa application is
https://download.oracle.com/java/17/archive/jdk-17.0.7_linux-x64_bin.tar.gz

After installing the JDK provide the installed location under the JAVA_HOME variable with a name, this is used as the JDK tool while writing the pipeline script.
The location in the above case is /var/jenkins_home/tools/hudson.model.JDK/jdk17.7/jdk-17.0.7

Build the maven project

Job Setup

Go to Dashboard -> New Item, give Jenkins job a name, and select Pipeline as the type. go to the Configuration section under General, -> select GitHub project and specify the URL of our git repository. From Build Triggers select GitHub hook trigger for GITScm polling

For pipeline job we can select either pipeline script or pipeline script from SCM option. If we choose pipeline script from SCM, add a Jenkinsfile in the GitHub repo and write all the script there. Go to SCM select git, and specify the git repository URL from where it should fetch the Jenkinsfile. If we choose pipeline script, write the pipeline script in the provided space

For getting the pipeline syntax Go to pipeline job –>pipeline syntax –>Snippet Generator–>select the step(for example if we want to add GitHub Repository choose the option git:GIT ,provide the repository ,branch credentials, and It will generate pipeline scripts.).

Install Cloud Foundry Plugin for Deployment

From the Jenkins Dashboard, go to Manage Jenkins ->Manage Plugins -> Available Plugins And install Cloud Foundry Plugin.
Generate the pipeline script by Providing the cloud space,organization,target and credentials of the Cloud Foundry Environment and generate the pipeline syntax.

Below shows demo pipeline code

```
pipeline {
  agent any
   tools{
      jdk 'jdk17.7'
      maven 'maven'
      }
stages {
  stage('git') {
     steps {
            git 'https://github.com/****/****'
      }
}
   stage('Build') {
        steps {
           sh 'mvn clean install'
        }
      }
    stage('Deployment') {
```

CI/CD Pipeline for Frontend React Application with Jenkins

Install NodeJS plugin &Configure

Open Jenkins: Manage Jenkins > Plugin Manager > Install NodeJS plugin.

For configuring the NodeJs tool in the global tool configuration
Open Jenkins: Manage Jenkins > Global Tool Configuration > NodeJS
Set compatible node version. We can set multiple NodeJS version for multiple application.

Install Cloud Foundry plugin

Open Jenkins: Manage Jenkins > Plugin Manager > Install Cloud Foundry plugin**.**

Build Application with Jenkins Pipeline.

Open Jenkins -> New Item. Enter any job name > Choose Pipeline **>** Click OK.
There 2 options for pipeline job,Pipeline Script or Pipeline Script From SCM

Pipeline Script:
You can write your Pipeline code directly on Jenkins job.

Pipeline Script From SCM:
Pipeline supports fetching the DSL (Domain Specific Language) script from the SCM.
Typically called Jenkinsfile and located in the root of the project.

- Select Pipeline script from SCM from the definition.
- Select Git as SCM
- Provide the GitHub URL and Credentials
- Specify the Branches to build

For both types of the pipeline, you can use this demo pipeline code.

```
pipeline
{
            agent any

            tools {nodejs "nodejs"}

              stages {

                stage('Git') {
                  steps {
                    git 'https://github.com/****/****'
                  }
                }

                stage('Build') {
                  steps {
                    sh 'npm install'
                     sh '<<Build Command>>'
                  }
                }
                }

              }
```