Continuous Integration (CI) and Continuous Delivery (CD) Jenkins Pipeline job. We also implement manual steps for the deployment to a staging environment.

###### ****Prerequisites:****

****✔  Maven:****Install Maven on your Jenkins server.

****✔  SSH Publisher Plugin:**** Install ‘SSH Publisher’ or ‘Publish over SSH’ plugin in Jenkins.

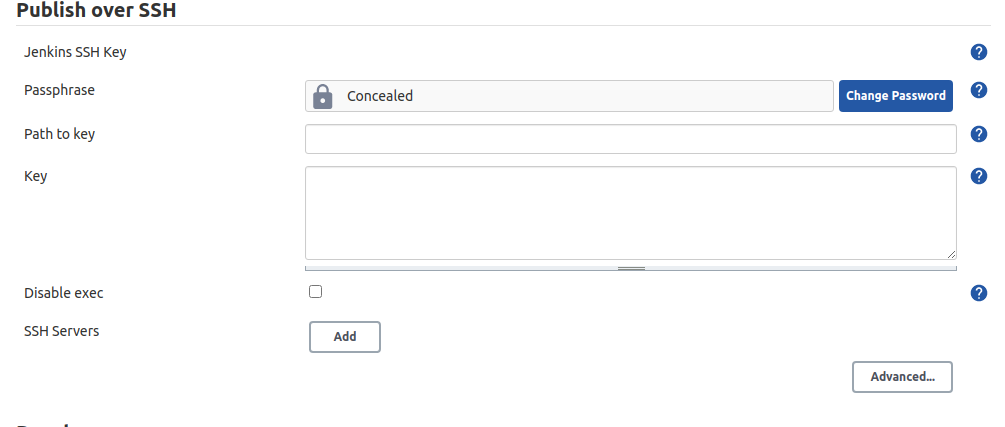
****✔  Staging server:****a remote server on which we will deploy the artifact. In this, the AWS EC2 instance is being used.

****✔  Java Web Application:**** Get the demo project from here**. **https://github.com/dwops-git/springDemo****

#### ****Step 0. Install and Configure the SSH Publisher Plugin****

****✔****Install the plugin by going to****Manage Jenkins > Manage**** Plugin > Available Tab > Search SSH publisher > Install the plugin.

****✔****Configure the plugin by going to Manage Jenkins > Configure System > Find Publish over SSH.



****✔****Fill the required by fields as follows:

****✔****Passphrase: If has any. (In this case, we don’t have any)

****✔****Key: Paste the .pem file or private key of the staging server. 

****✔  SSH Servers:****Click ‘Add’ 

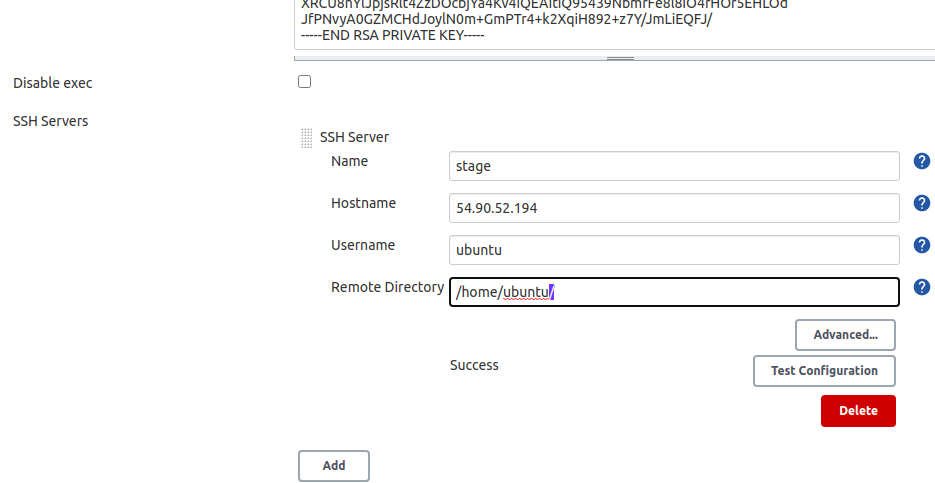
****✔  Name:****Give an appropriate name 

****✔  Hostname:****IP address of the staging server. 

****✔  Username:**** username used to log in to the staging server. 

****✔  Remote Directory:****Directory path where you want to deploy the artifact.

****✔****After that test the configuration by clicking the Test button.



#### ****Step 1. Create a Pipeline Job.****

****✔****Click on ****New Items****.

****✔****Give the project a suitable name and choose the Pipeline project type.

#### ****Step 2. Configure the CI Part.****

****✔****Go to the Build Triggers section.

****✔****Select Poll SCM and schedule to check for any changes for every minute using the following cron expression: \* \* \* \* \*

****✔****Under Pipeline write the basic syntax of the pipeline such as:

Pipeline {

agent any

stages{

stage(‘’){

}

}

}

****✔****Open the ****Pipeline Syntax**** page in a new tab to generate pipeline syntax.

###### ****Stage 1: ‘Git Clone’****

****✔****Write your first stage to clone the code from GitHub by generating the syntax.

###### ****Stage 2: ‘Clean Package’****

****✔****We will be using ****Maven****installed on the Jenkins server for the next 2 stages.

****✔****In the ****steps**** section, write the maven command:

stage('Clean Package'){

steps{

sh ‘mvn clean package’

}}

****✔  Clean**** will clean the target directory and the ****package****will build and packages the project.

###### ****Stage 3: ‘Test the code’****

****✔****Next, we will use maven to perform unit tests on the project.

****✔****In the ****steps**** section, write the below command: sh ‘mvn test’

stage('Test the Code'){

steps{

sh ‘mvn test’

}}

#### ****Step 3. Adding the CD Part in Pipeline****

Now, we will add the last stages of the pipeline.

###### ****Stage 4. ‘Manual Approval’****

****✔****This stage will be used to get the manual approval from another Jenkins User. This will prevent the automatic deployment of artifacts (.war file) on the staging server.

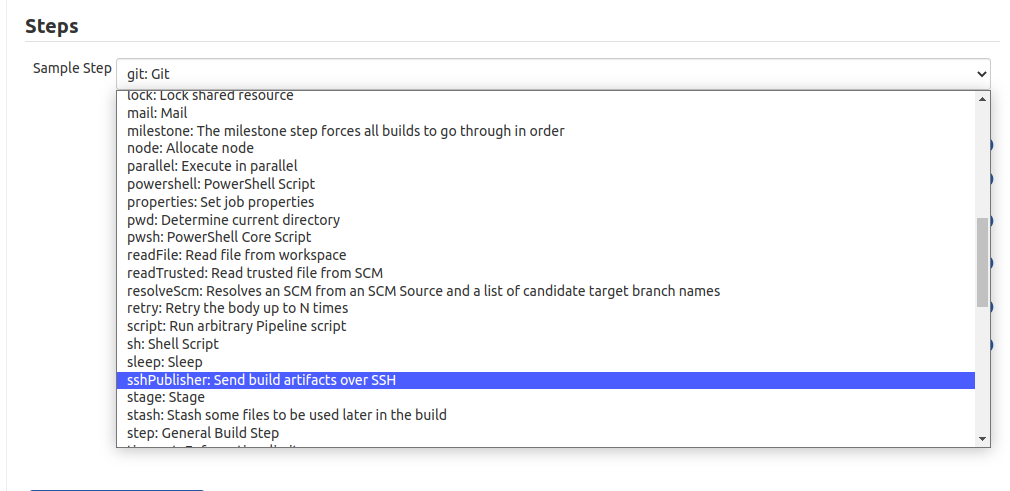
****✔****For simplicity, we will add a basic ‘Proceed or Abort’ option here.

****✔****You can get the code from the Pipeline Syntax page by choosing the ‘input’ option.

input ‘Deploying to stage server’

###### ****Stage 5. ‘Deploy’****

****✔****In this stage, we will deploy the artifact (.war file) to the staging server using the SSH Publisher plugin.



****✔****Just select ‘sshPublisher’ in the dropdown of Sample Step and fill the required fields:

****✔****Name: Select the name of the server configured in Step 0.

****✔****Source files: give the path of the artifact using a pattern. For eg. \*\*/\*.war

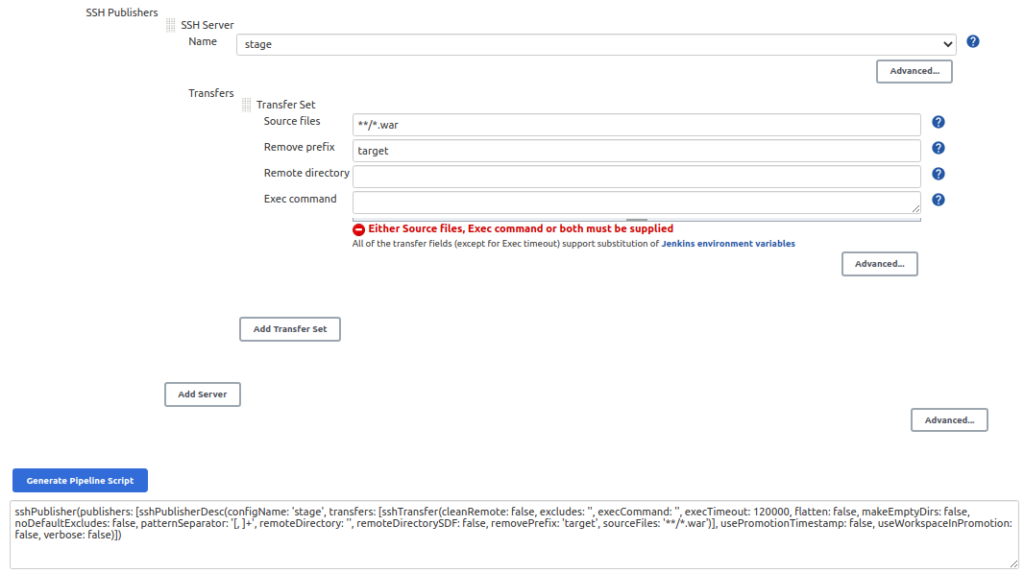
****✔****Remove Prefix: If need to remove some folder from the path.

****✔****Remote directory: This is the same field configured in

****Step 0****. So, we can leave this blank. 

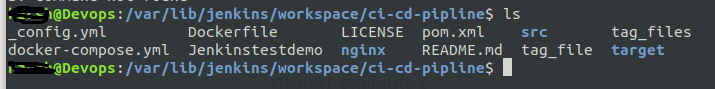
****✔  Exec Command:**** To execute any command on the server.

****✔****Next, Generate the Script and paste it into the ****steps**** section of this ****stage****.



****✔****Save the configuration.

****Note:****You can see the Code and the build of the code on the Jenkins server at /var/lib/jenkins/<name\_of\_job>. This will help you to know the directory structure of the job and how to provide the path of the .war file in the SSH Publisher plugin configuration.



#### ****Final Script:****

pipeline {

    agent any

    stages {

        stage('git clone') {

            steps {

                git 'https://github.com/dwops-git/springDemo.git'

            }

        }

        stage('Clean package'){

            steps{

                sh 'mvn clean package'

            }

        }

        stage('Test the code'){

            steps{

                sh 'mvn test'

            }

        }

        stage('Manual Approval'){

            steps{

                input 'Deploying to Stage server'

            }

        }

        stage('deploy'){

            steps{

                sshPublisher(publishers: [sshPublisherDesc(configName:   
'stage', transfers: [sshTransfer(cleanRemote: false,   
excludes: '', execCommand: '', execTimeout: 120000, flatten: false,   
makeEmptyDirs: false, noDefaultExcludes: false,   
patternSeparator: '[, ]+', remoteDirectory: '',   
remoteDirectorySDF: false, removePrefix: 'target',   
sourceFiles: '\*\*/\*.war')], usePromotionTimestamp: false,   
useWorkspaceInPromotion: false, verbose: false)])

            }

        }

    }

}

#### ****Step 5. Build the Project.****

****✔****Build the Job by clicking on ****Build Now.****

****Or****

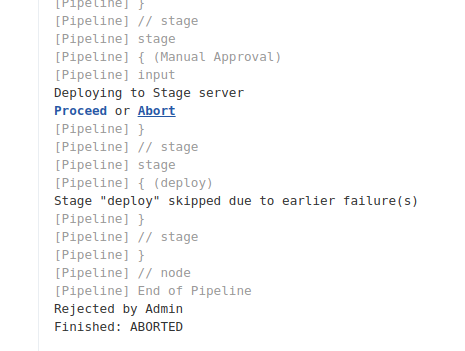
****✔****As we have implemented ****Continuous Integration**** in the job. So, we can just push a commit in GitHub code and it will automatically get triggered and start the Build.

#### ****Step 6. Implement Manual Approval**** ****Step****

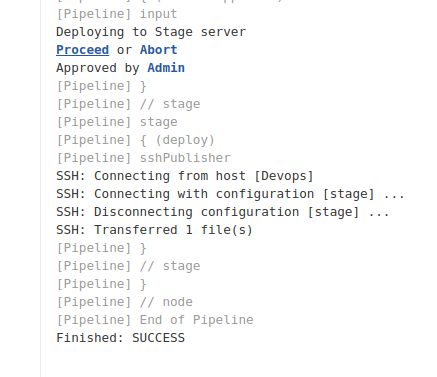
****✔****Go to the****Console Output.****

****✔****After implemented all three stages there will be a prompt for manual approval i.e. stage 4.

****✔****If we click on ****Abort****, it will skip the next stage and abort the deployment on the staging server.

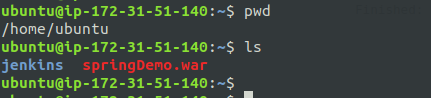


****✔****And, if we click on ****Proceed,**** then it will move to the next stage and deploy the artifact on the stage server.



#### ****Step 7. Check the Successful Deployment of Artifact.****

****✔****Login to the Staging server and check the .war file at the path specified in the ****Remote Directory****in ****Step 0.****



You can Install a Tomcat server on the staging server and use this war file.