

CI CD Pipeline setup by creating Jenkins Pipeline Job:

Step 1.1 Create Jenkins instance and install all the suggested plugins (Refer Jenkins-Intro doc)

Step 1.2 Set Maven and JDK path (Refer Jenkins-Intro PDF)

Step 1.3 Install **ssh-agent, Pipeline Maven Integration, maven integration, Task Scanner, JUnit Attachments** plugins.

Step 2.1 It's time to create CI-CD pipeline, create a Jenkins pipeline job

<https://github.com/prakashk0301/maven-project> (choose branch ci-cd-pipeline)

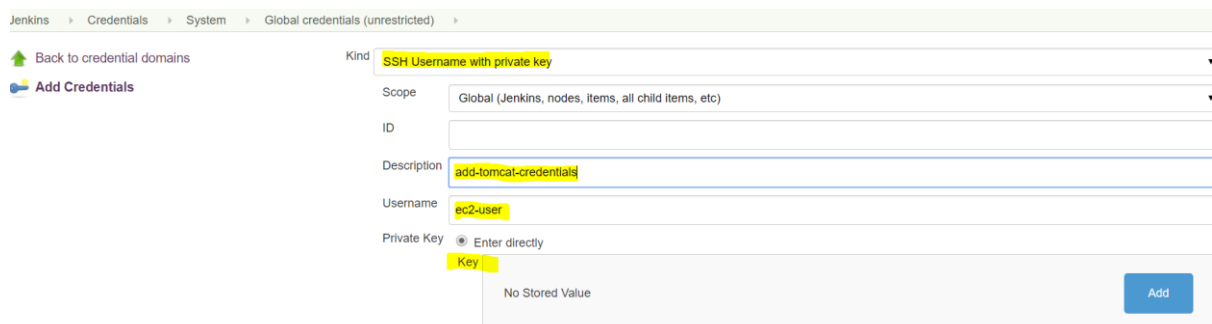
and build your job.

Step 3.1 Create an Instance and install Tomcat (Refer Tomcat installation doc)

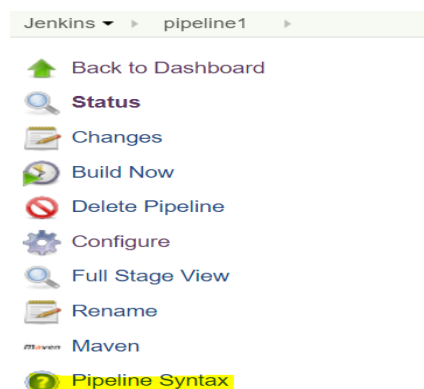
Step 3.2 Open Jenkins home page, in the left corner side look for credentials option.

Click Jenkins->Global Credentials->Add credentials

Choose kind as SSH Username with private key and provide user as ec2-user and add you pem key.

The screenshot shows the Jenkins 'Add Credentials' form. The breadcrumb trail at the top is 'Jenkins > Credentials > System > Global credentials (unrestricted)'. On the left sidebar, there are links for 'Back to credential domains' and 'Add Credentials'. The main form has a 'Kind' dropdown set to 'SSH Username with private key'. The 'Scope' is 'Global (Jenkins, nodes, items, all child items, etc)'. The 'ID' field is empty. The 'Description' field contains 'add-tomcat-credentials'. The 'Username' field contains 'ec2-user'. The 'Private Key' section has a radio button selected for 'Enter directly' and a text input field containing 'Key'. Below the input field is a 'No Stored Value' message and an 'Add' button.

Step 3.3 Go back to Jenkins job and look for **Pipeline Syntax** option.



Step 3.4 Select ssh agent from dropdown menu.

It, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

Sample Step sshagent: SSH Agent

```
node {
  sshagent (credentials: ['deploy-dev']) {
    sh 'ssh -o StrictHostKeyChecking=no -l cloudbees 192.168.1.106 uname -a'
  }
}
```

Multiple credentials could be passed in the array but it is not supported using Snippet Generator.

(from [SSH Agent Plugin](#))

ec2-user (add-tomcat-credentials)

Add

Step 3.5 Generate pipeline Script, It will provide us credential id which can be put in our Jenkinsfile.

// some block: here we can provide scripts so basically it will copy war file from Jenkins server and get it deployed on tomcat server.

Generate Pipeline Script

```
sshagent(['f39cd834-5e32-428e-965f-d6020b95c133']) {
  // some block
}
```

Step 3.6 Copy script and put it in a notepad.

```
sshagent(['f39cd834-5e32-428e-965f-d6020b95c133']) {
  // some block
}
```

Here we can put scripts, which will copy artifact from Jenkins server and through scp it automatically get deployed on tomcat

Let's modify Jenkinsfile now by adding stage called "deploy to tomcat" and "steps"

```
stage('deploy to tomcat') {
  steps {
    sshagent(['f39cd834-5e32-428e-965f-d6020b95c133']) {
      sh 'scp -o StrictHostKeyChecking=no */target/*.war ec2-user@172.31.42.125:/var/lib/tomcat/webapps'
    }
  }
}
```

Ex:

```
stage('deploy to tomcat') {
  steps {
    sshagent(['f39cd834-5e32-428e-965f-d6020b95c133']) {
      sh 'scp -o StrictHostKeyChecking=no */target/*.war ec2-user@172.31.42.125:/var/lib/tomcat/webapps'
    }
  }
}
```

Step 4.1 Build your job, verify console output (change permission if you get any error
/var/lib/tomcat/webapps/)

```
[ssh-agent] Stopped.  
[Pipeline] // sshagent  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] }  
[Pipeline] // withEnv  
[Pipeline] }  
[Pipeline] // node  
[Pipeline] End of Pipeline  
Finished: SUCCESS
```

Enjoy 😊

Jenkins Pipeline execution engine supports two DSL syntaxes: Scripted Pipeline and Declarative Pipeline.

Scripted Pipeline allows users to code their Pipelines using a Groovy DSL. Declarative Pipeline replaces Groovy variable assignments, control structures, loops, and exception handling with a predefined structure and model to allow users of all experience levels to quickly create consistent, concise Pipelines without learning Groovy.

Node Blocks

The first block to be defined is the “node”:

```
node {  
}
```

A “node” is part of the Jenkins distributed mode architecture, where the workload can be delegated to multiple “agent” nodes. A “master” node handles all the tasks in your environment
Stage Blocks

Stage Blocks

The next required section is the “stage”:

```
stage {  
}
```

Your pipeline will consist of several steps that can be grouped in stages. Among these stages you might have:

- Pull code from repository
- Build your project
- Deploy your application
- Perform functional tests
- Perform performance tests

Each of the these can include more than one action. For example, a stage to deploy your application can consist of copying the files to a specific environment for functional tests and to a dedicated server for performance tests; and once files are copied successfully, proceed to deploy it.