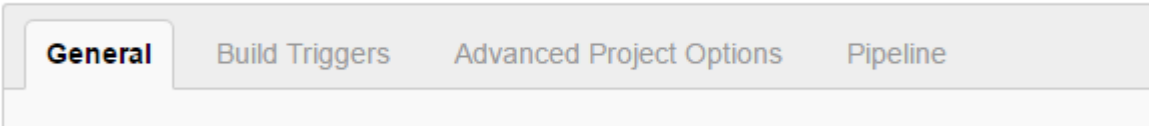
Jenkins Pipeline Job

Select the Pipeline job after giving a name to the job.



The menu is different from the freestyle project, where we configured build, post build action etc. In case of Pipeline job we do not have those option, instead we have a menu item to create a Pipeline itself. Let's look at the option.

The main section of the Pipeline job is the Pipeline section where we write a groovy script to align the steps in a pipeline.

Groovy is based on Java, so one who understands Java can very easily grasp with the groovy syntax.

Jenkins provides a help to create the Groovy script by using the 'Pipeline syntax' option.

Now to start creating a Pipeline job to build using Maven as the build stage, click on the 'Pipeline Syntax ' option, this will open a page to create groovy syntax for the action (steps) in the build job.

Here select the step as 'bat: Windows Batch script'. Here enter the commands that we will use for running the maven build, as 'clean package'

Click on the 'Generate Pipeline script' and this will provide the groovy syntax for build step.

The obvious step would be to archive the artefacts. For this we can use the snippet generator again, and in that select the step as 'step: General build step', in here select the 'Archive the artifact' build step.

Likewise select appropriate sample steps to create the entire build script in Groovy for the Pipeline job.

One very important thing to note here is that the pipeline job as to be assigned to a node to work with the pipeline job. So select the groovy snippet generator and select, 'node:Allocate node'

This functionality in the pipeline job actually allows selecting which node to be used for running the job and thus manage the way we want the Jenkins to run parallel tasks.

This functionality in Jenkins is on the principle that Jenkins works in the Master – slave mode, while most of the jobs are run on the master, if the master remains

busy, then the agent is pulled to work on the request. The jobs are distributed to the node (agents) as per their availability.

```
node {
   stage 'checkout'
   git 'https://github.com/ganeshhp/helloworldweb.git'

   def project_path='TestProject/simpleWebApp'

      dir(project_path) {
         stage 'build'
         bat 'mvn clean package'

         stage 'archive'
         archiveArtifacts 'target/*.war'

         stage 'issue log'
         step([$class: 'JiraVersionCreatorBuilder'])
      }

}
```

The Pipeline syntax being a Groovy script, we can modify, manipulate it to a great extent.

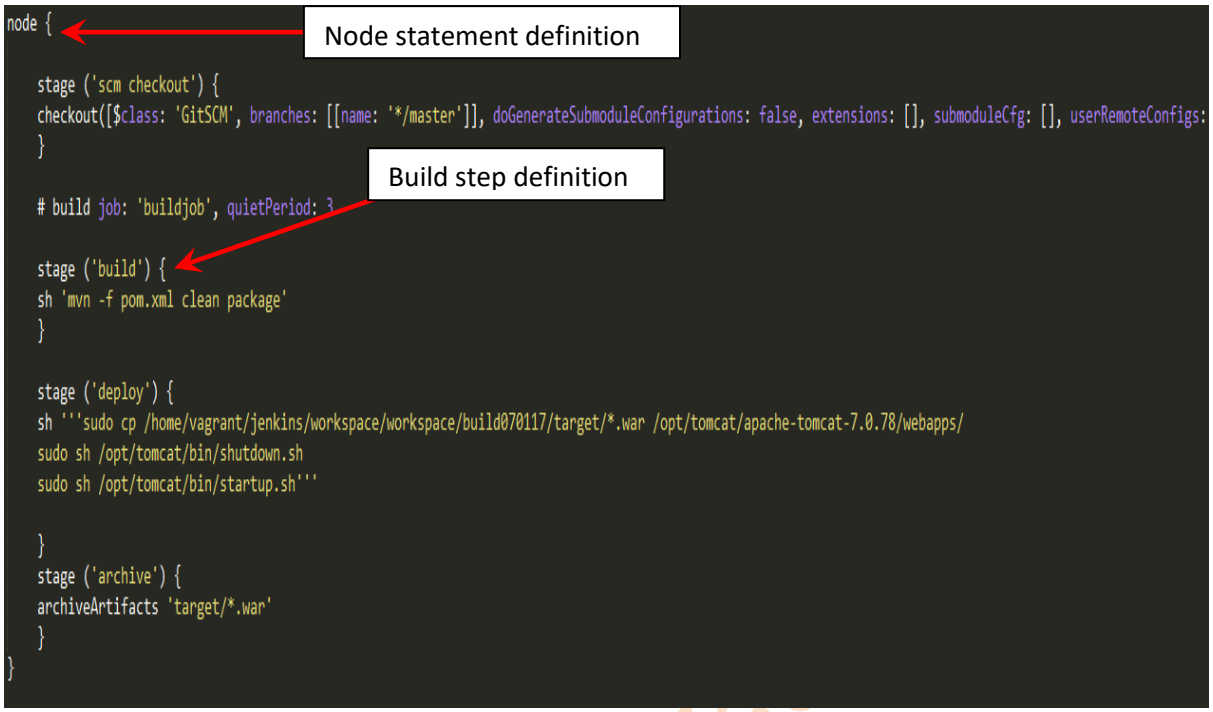
Try adding some stages as shown below.,

```
stage('Build') {

      build 'sample'

  }
```

Here the ‘sample’ is a job which is executed as a step in the pipeline job.



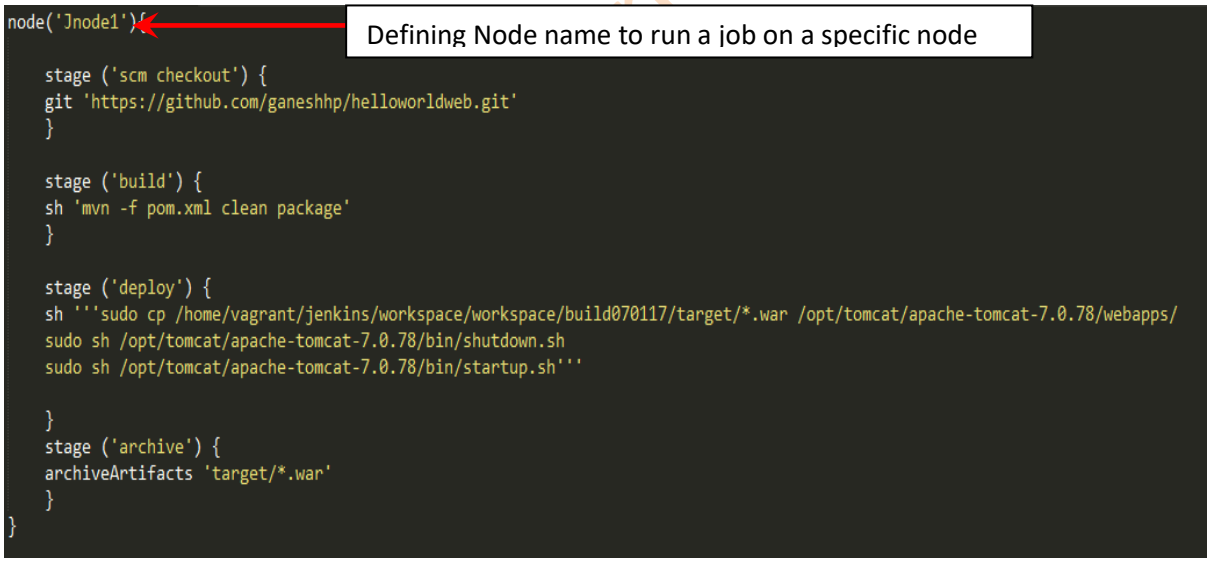

```
node {
   stage ('scm checkout') {
   checkout([$class: 'GitSCM', branches: [[name: '*/master']], doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [], userRemoteConfigs:
   }

   # build job: 'buildjob', quietPeriod: 3

   stage ('build') {
   sh 'mvn -f pom.xml clean package'
   }

   stage ('deploy') {
   sh '''sudo cp /home/vagrant/jenkins/workspace/workspace/build070117/target/*.war /opt/tomcat/apache-tomcat-7.0.78/webapps/
   sudo sh /opt/tomcat/bin/shutdown.sh
   sudo sh /opt/tomcat/bin/startup.sh'''

   }
   stage ('archive') {
   archiveArtifacts 'target/*.war'
   }
}
```




```
node('Jnode1'){
    stage ('scm checkout') {
    git 'https://github.com/ganeshhp/helloworldweb.git'
    }

    stage ('build') {
    sh 'mvn -f pom.xml clean package'
    }

    stage ('deploy') {
    sh '''sudo cp /home/vagrant/jenkins/workspace/workspace/build070117/target/*.war /opt/tomcat/apache-tomcat-7.0.78/webapps/
    sudo sh /opt/tomcat/apache-tomcat-7.0.78/bin/shutdown.sh
    sudo sh /opt/tomcat/apache-tomcat-7.0.78/bin/startup.sh'''

    }
    stage ('archive') {
    archiveArtifacts 'target/*.war'
    }
}
```

Declarative Pipeline Syntax:

Jenkins Pipeline script can also be written in form of Declarative syntax. This is going to be used widely in practise.

We start writing the script by defining ‘Pipeline’ as key work. What follows thereafter are pipeline keyword.

For defining node we write it as ‘agent’.

Agent declarations:

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.

Parameter options are… ‘any’, ‘none’, ‘label’, ‘node’, ‘docker’, ‘dockerfile’.

Stage declaration:

Each stage is defined as ‘stage’, statement.

Sample script:

```
pipeline {
    agent { docker 'maven:3-alpine' }
    stages {
        stage('Example Build') {
            steps {
                sh 'mvn -B clean verify'
            }
        }
    }
}
```

Also,

```
pipeline {
    agent none
    stages {
        stage('Example Build') {
            agent { docker 'maven:3-alpine' }
            steps {
                echo 'Hello, Maven'
                sh 'mvn --version'
            }
        }
        stage('Example Test') {
            agent { docker 'openjdk:8-jre' }
            steps {
                echo 'Hello, JDK'
                sh 'java -version'
            }
        }
    }
}
```

POST (post build action):

The post section defines one or more additional steps that are run upon the completion of a Pipeline’s or stage’s run (depending on the location of the post section within the Pipeline).

This has following post conditional blocks: post-condition blocks: always, changed, fixed, regression, aborted, failure, success, unstable, unsuccessful, and cleanup.

```
pipeline {
    agent any
    stages {
        stage('Example') {
            steps {
                echo 'Hello World'
            }
        }
    }
    post {
        always {
            echo 'I will always say Hello again!'
        }
    }
}
```

Stages statement:

Containing a sequence of one or more stage directives, the stages section is where the bulk of the "work" described by a Pipeline will be located. At a minimum it is recommended that stages contain at least one stage directive for each discrete part of the continuous delivery process, such as Build, Test, and Deploy.

Sample Jenkinsfile:

```
/* This script is designed and maintained by Ganesh Palnitkar
*/
def notify(status) {
    mail (
        body:"""${status}: Job '${env.JOB_NAME}
[${env.BUILD_NUMBER}]': Check console output at,
                href='${env.BUILD_URL}'>${env.JOB_NAME}
[${env.BUILD_NUMBER}]""",
        cc: '<valid-email-id>',
        subject: """JenkinsNotification: ${status}:""",
        to: '<alid-email-id>'
       )
 }
def server = Artifactory.server 'artifactory'
def uploadSpec = """{
      "files": [
      {
        "pattern": "target/petclinic.war",
```

```
        "target": "petclinic/"
       }
                ]
}"""

/*def downloadSpec = """{
      "files": [
       {
           "pattern": "petclinic/*.war",
           "target": "./"
        }
      ]
     }"""
*/
pipeline {
 agent none
    stages {
      stage('SCM_Chekout') {
          agent { label "node1" }
                  steps {
                      script {
                              notify('build-started')
                          }
                checkout([$class: 'GitSCM',
                    branches: [[name: '*/master']],
                    doGenerateSubmoduleConfigurations: false,
                    extensions: [],
                    submoduleCfg: [],
                    userRemoteConfigs: [[url:
'https://github.com/ganeshhp/helloworldweb.git']]])
             }
        }
      stage('Build'){
          agent { label "node1" }
                  steps {
                sh 'mvn -f pom.xml clean package'
             }
        }

        stage('push-to-artifactory') {
          agent { label "node1" }
                  steps {
                script {
                            server.upload(uploadSpec)
             }
        }
       }
        stage('Deploy') {
          agent { label "node2" }
             steps {
               script {
```

```
                    input('Deploy Package to Production?')
                    notify('Deployment-to-Production')
                       }
                    sh 'wget
http://35.228.115.235:8081/artifactory/petclinic/Helloworldweb
app.war'
                    sh 'cp ./Helloworldwebapp.war
/opt/tomcat/webapps/'
                    sh '/opt/tomcat/bin/catalina.sh run'

             }
          }
    }
}
```