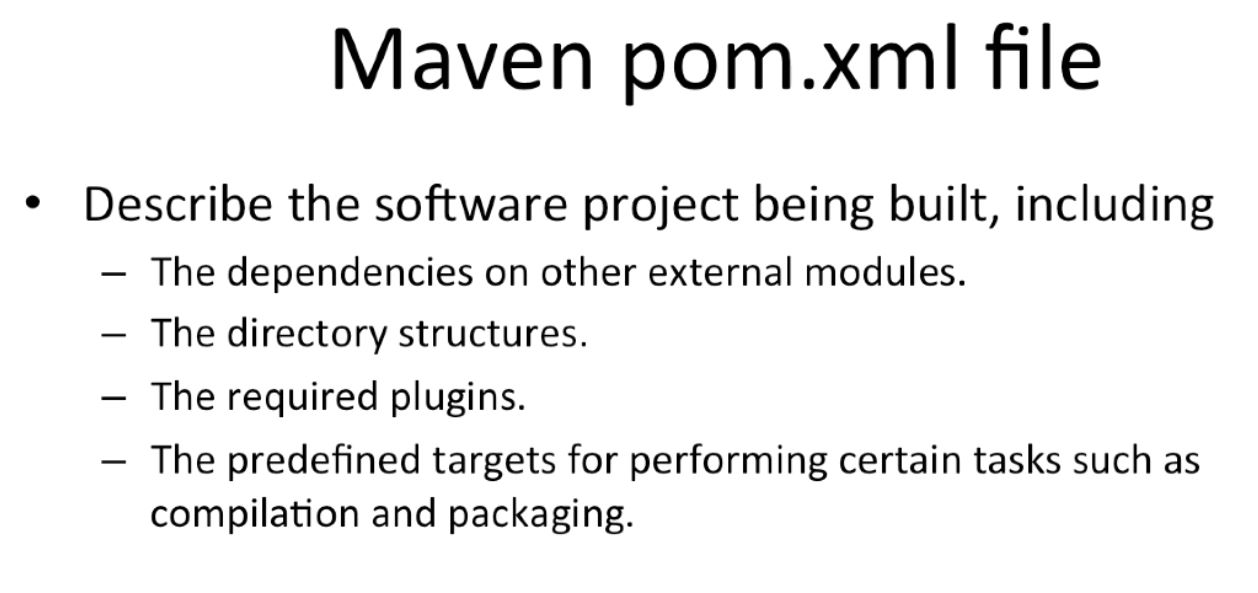
**Jenkins Architecture**

**Master/slave Architecture**

**Sect 2 -lecture 18**

Create maven based project

<https://github.com/jleetutorial/maven-project>



Text Direction: Source Control Polling in Jenkins

Section 2, Lecture 24

**Checking for existing SSH keys:**

https://help.github.com/articles/checking-for-existing-ssh-keys/

**Generating a new SSH key and adding it to the ssh-agent:**

https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/

**Adding a new SSH key to your GitHub account:**

https://help.github.com/articles/adding-a-new-ssh-key-to-your-github-account/

Text Direction: Other Build Triggers

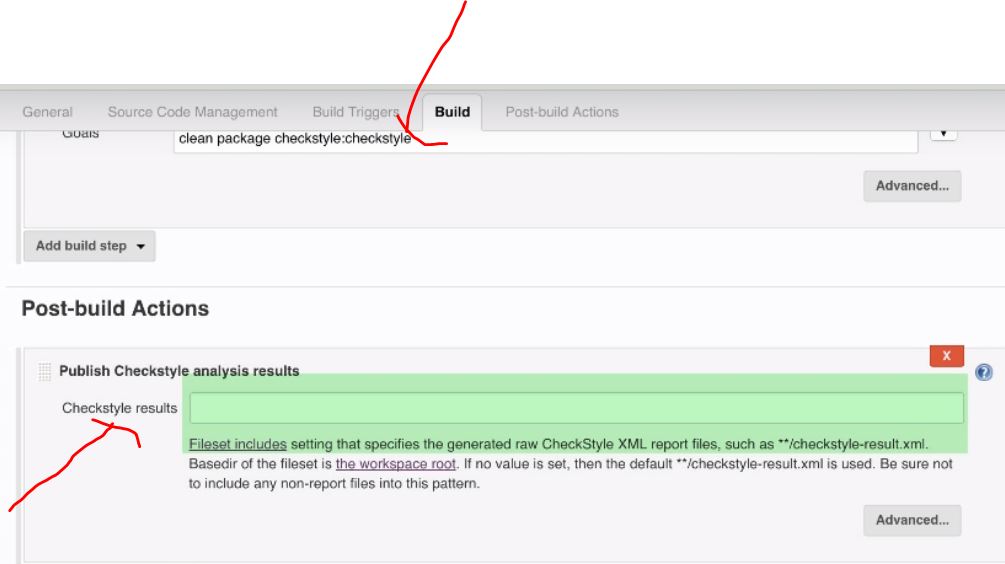
Section 2, Lecture 26

Jenkins GitHub Plugin:

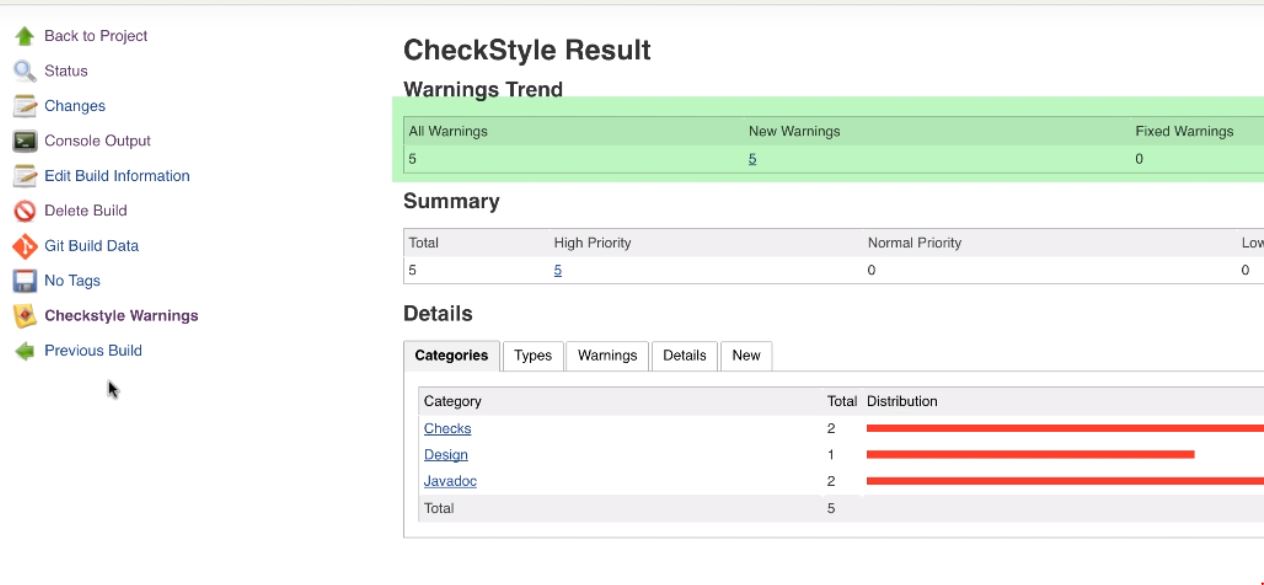
https://wiki.jenkins-ci.org/display/JENKINS/GitHub+Plugin

**Jenkins code quality metrics report**

Manage Jenkins->download checkstyle plugin->configure job



Build now



Other plugins for code check

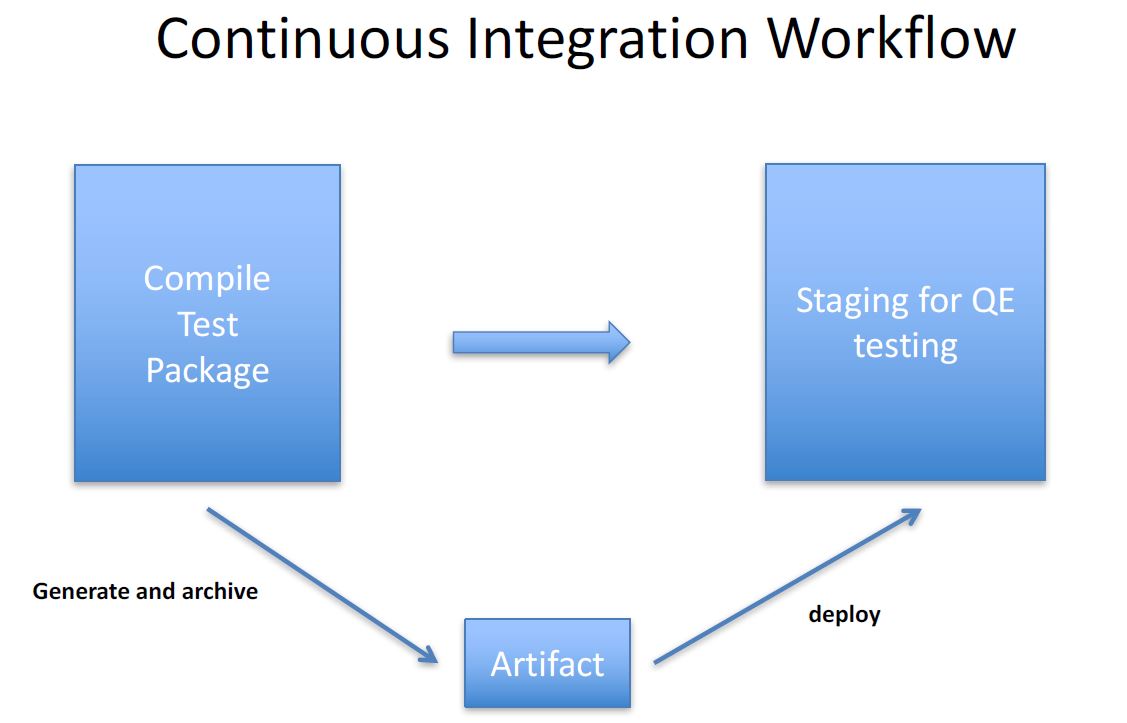
PMD Jenkins plugin:

https://wiki.jenkins-ci.org/display/JENKINS/PMD+Plugin

Findbugs Jenkins Plugin:

<https://wiki.jenkins-ci.org/display/JENKINS/FindBugs+Plugin>

Archive Build Artifacts ( Sec-4, Lec 31)



Deploy using Post-build action in Jenkins ->select archive the artifacts

**Install and configure Tomcat as a staging**

**Environment**

Change default port 8080 in config/server.xml file line 69

**Vi config/tomcat-users.xml**

<user username="rajbir" password="admin123" roles="manager-script"/>

**Deploy to Staging Env ( lect 33 )**

[**https://github.com/rajbirsood/restWebServices**](https://github.com/rajbirsood/restWebServices)

Install

Copy artifact plugin

deploy to container plugins

Create two jobs

* Package (to create artifacts) and in post build action give 2nd job name

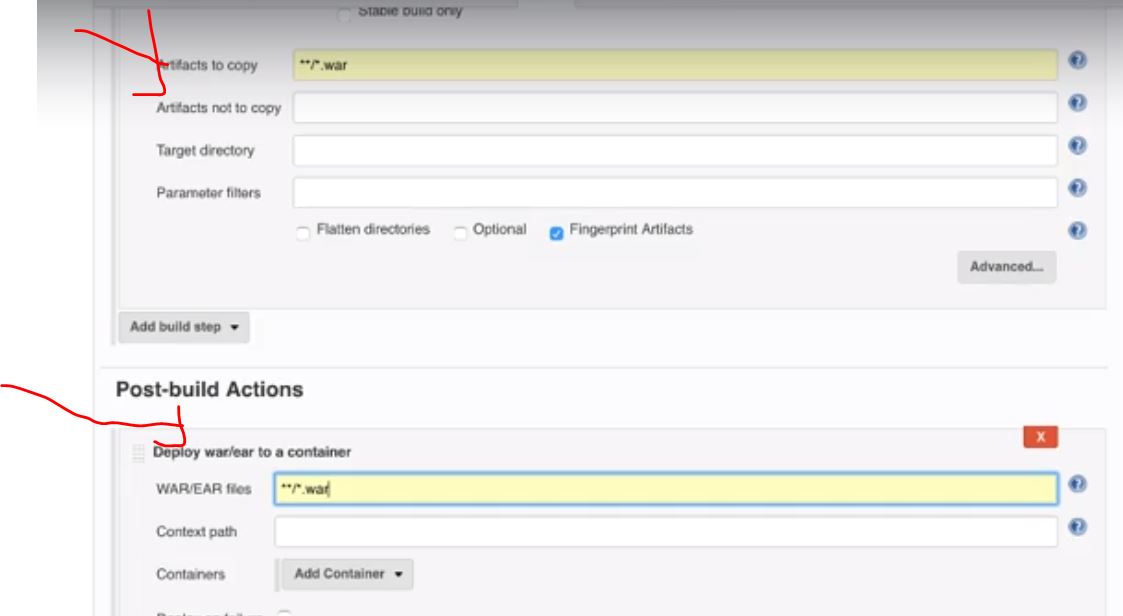
In post build action->archive the artifacts

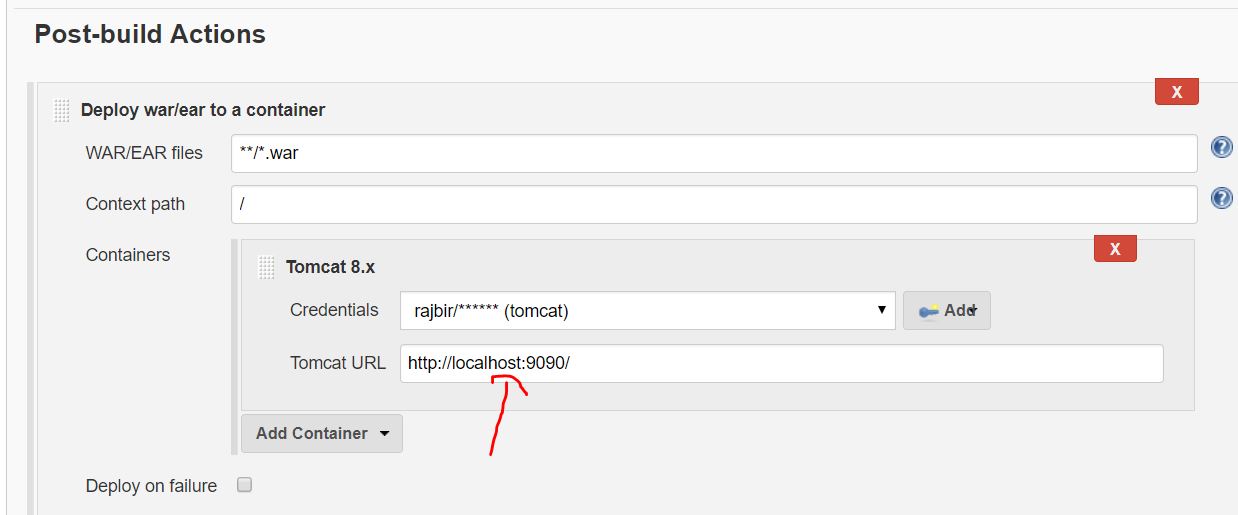


Run job

* Deploy to staging job

In 2nd job, call first job name i:e package

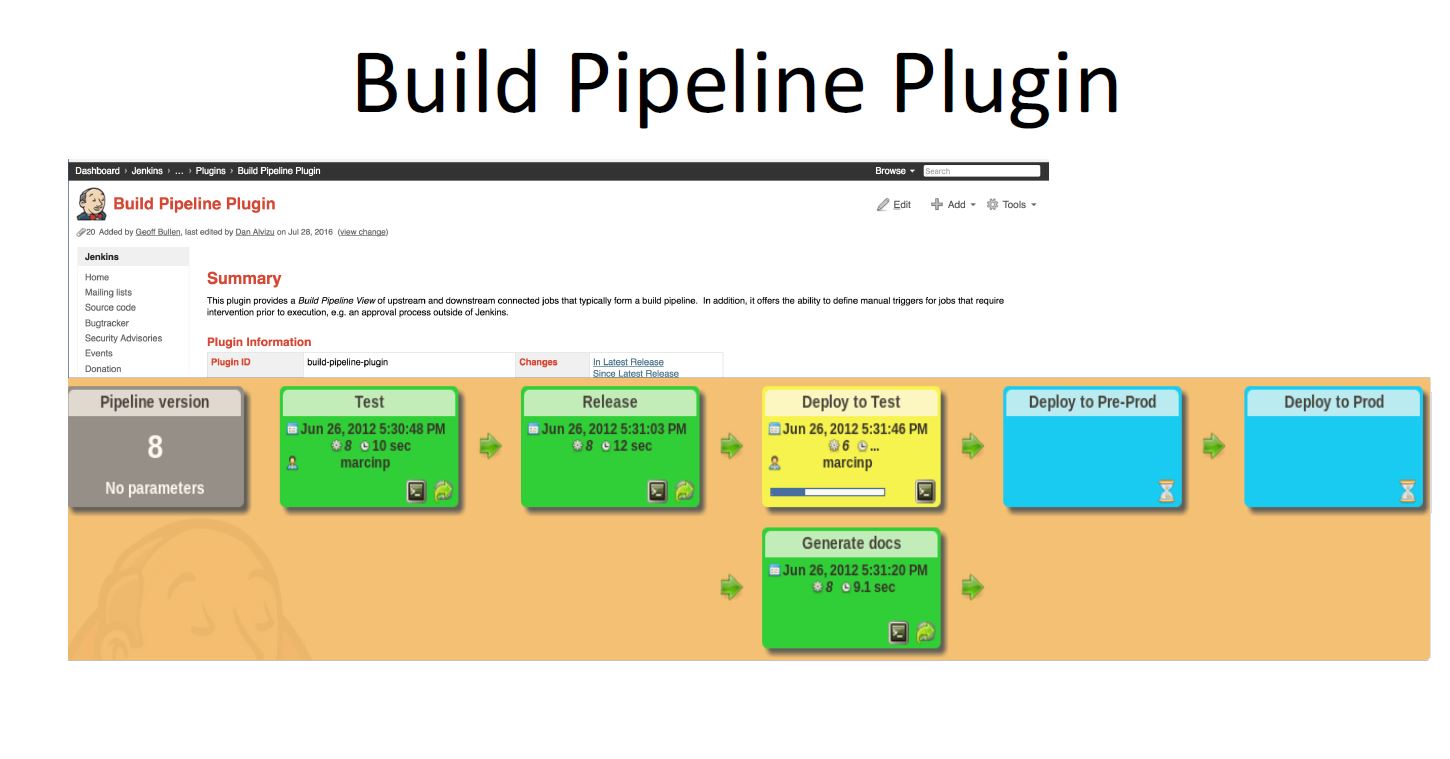




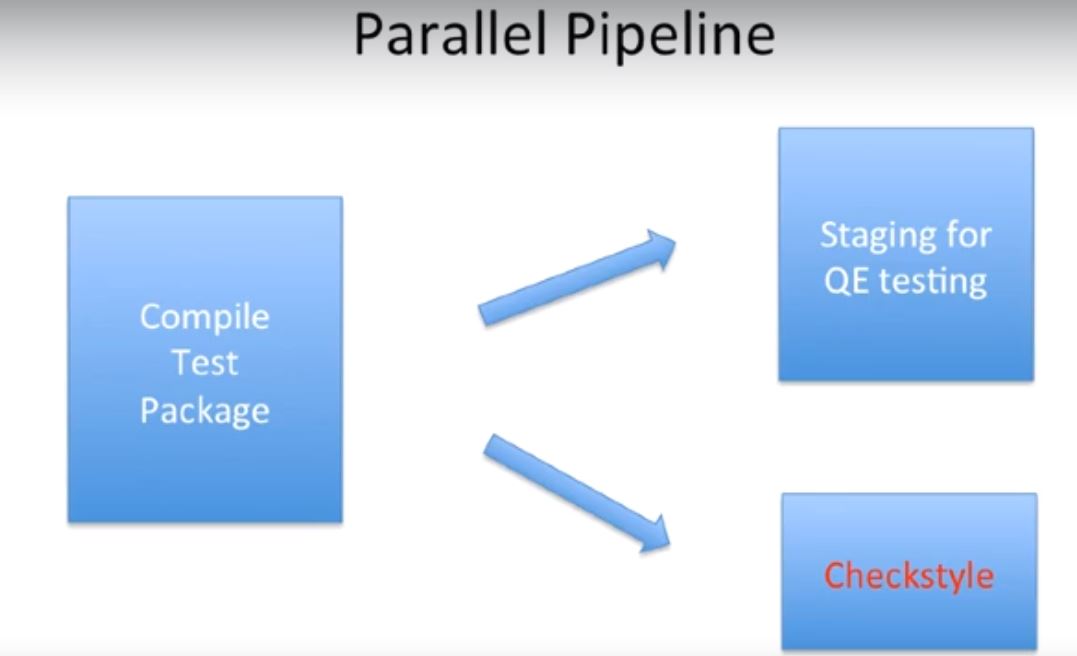
Start tomcat-> check url



**Jenkins build pipeline plugin**

****

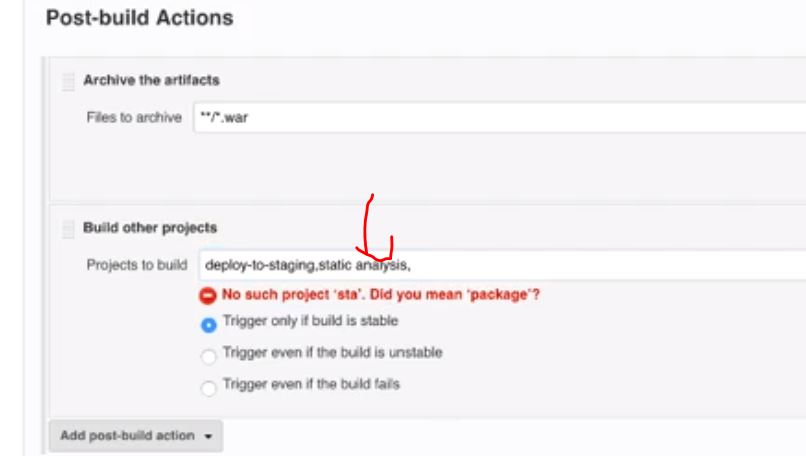
**Parallel Jenkins build ( lect 37)**

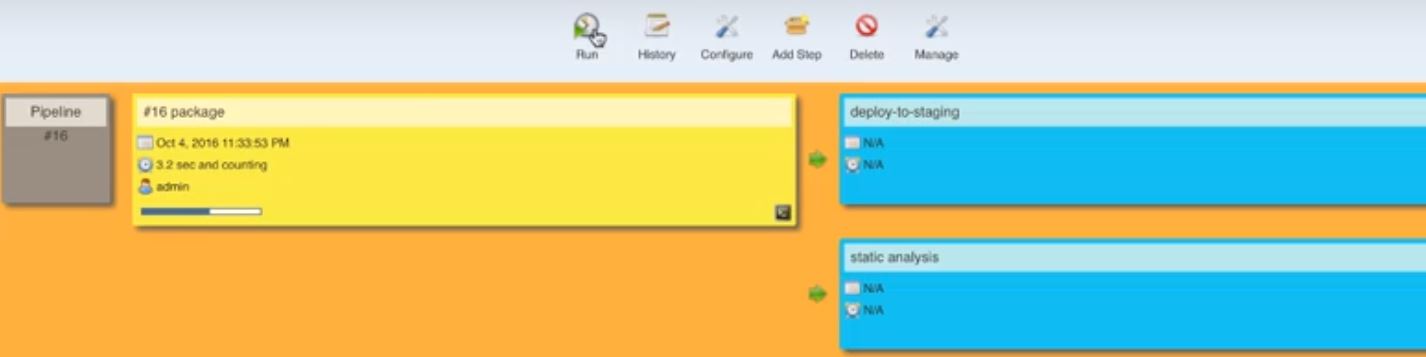
****

We already have package job to compile and package

Now, Create job static-analyisisthat only run checkstyle that will run in parallel

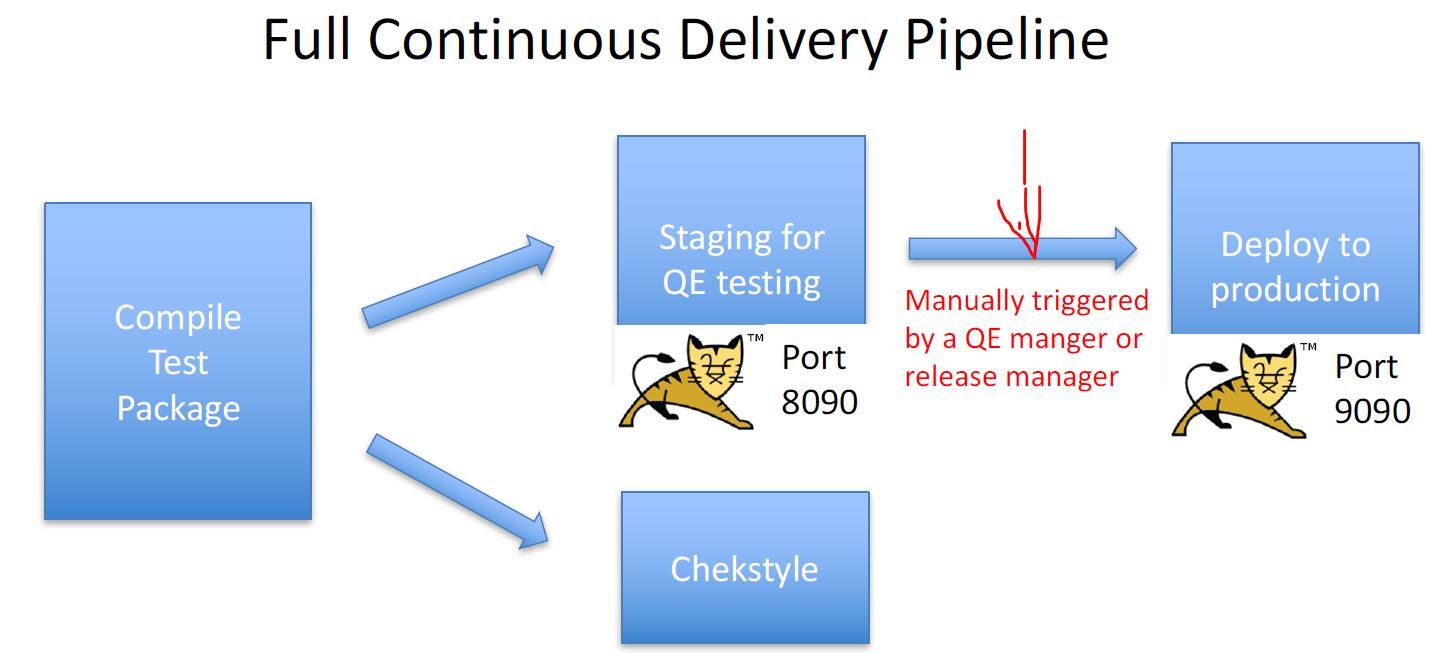
Go to package job->configure->post build actions->





**Deploy to Production**

We don’t want to deploy on production without approval



Create “deploy to Production” job

Upstream job is “deploy to staging”

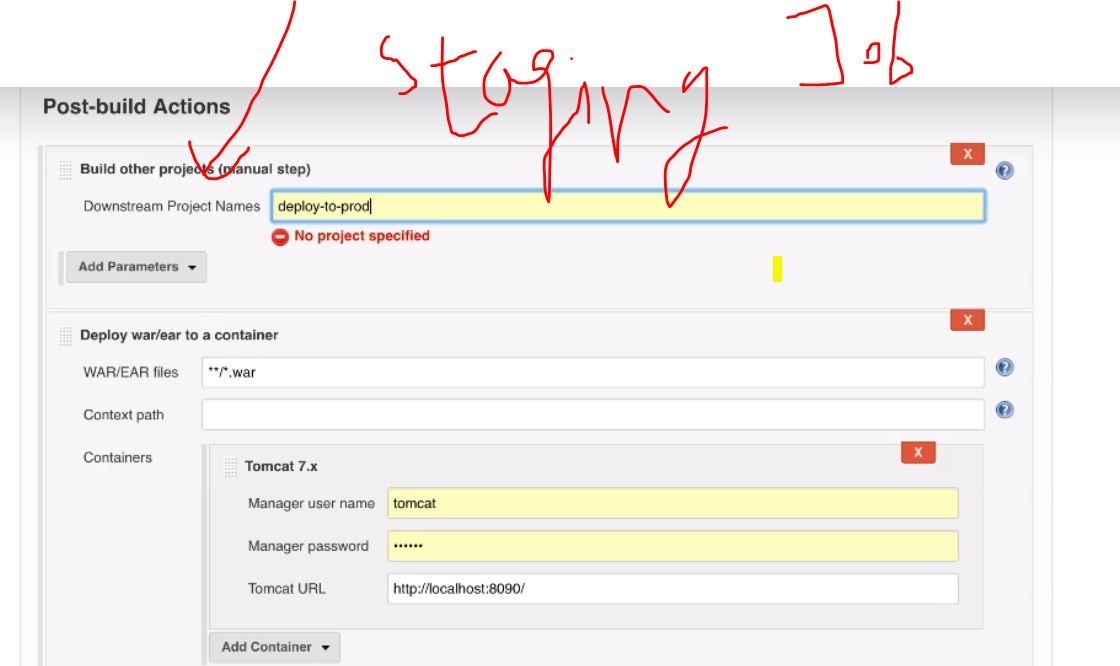
Create or spin new tomcat server for production on different port

Copy Rename to staging and production tomcat ->change port in server.xml file

Create job -> post build action->deploy war to container->save

Now add a trigger for this job (this job can only be run after deploy tp staging job completed successful)

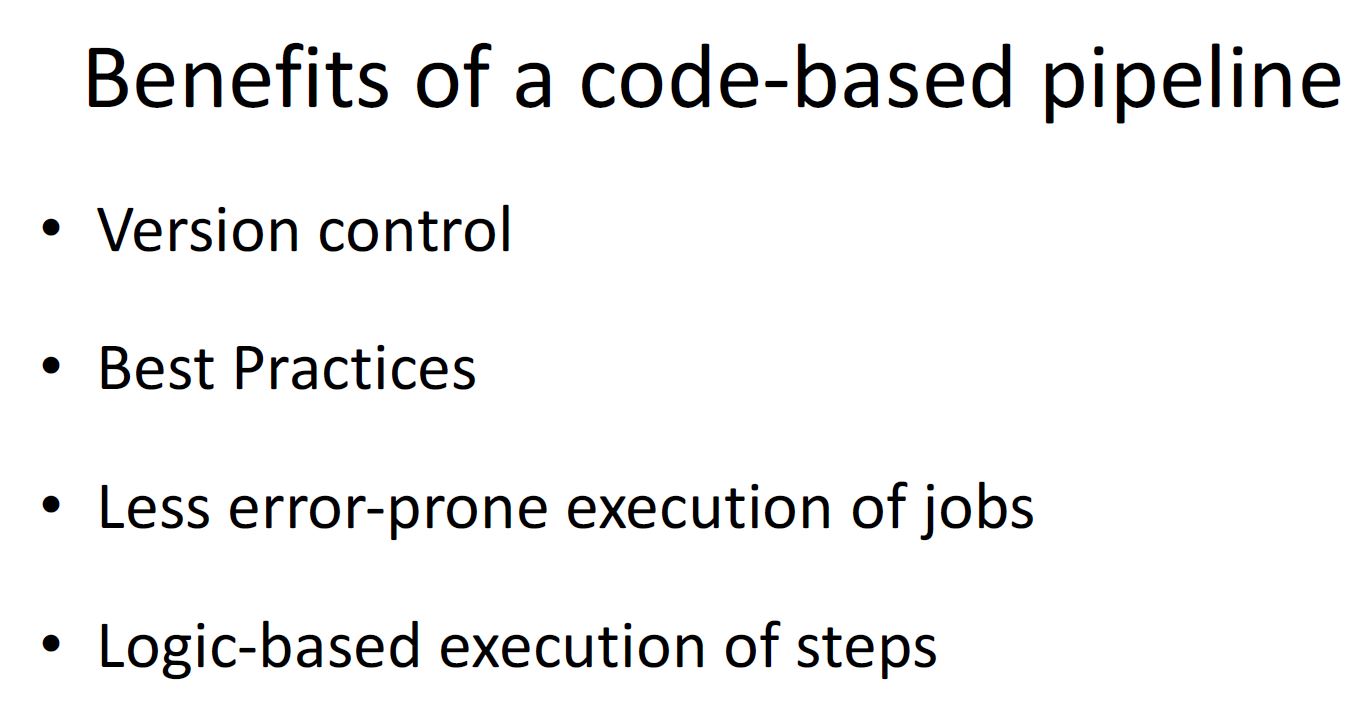
Go deploy to staging job->config->select build other projects (manual step)

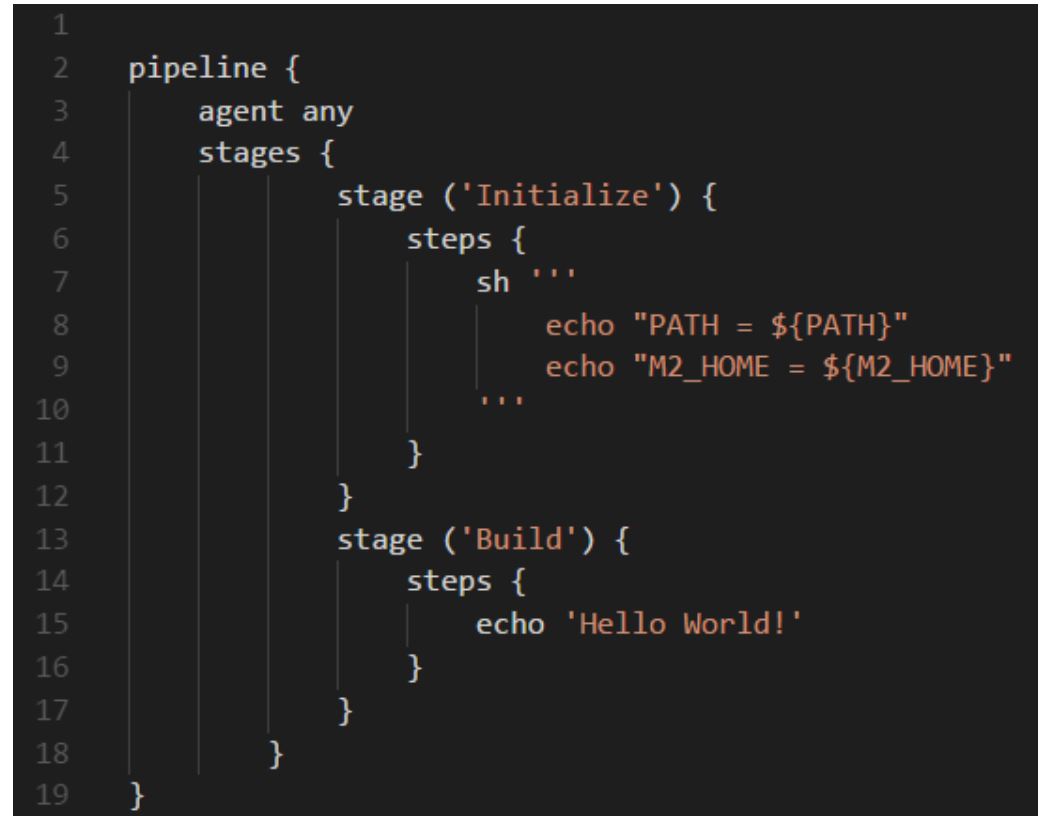


**Jenkins pipeline as a code**

**Overview of pipleline as code (Sec 5, Lect 42)**

It used DSL (domain specific language ) to define tasks which you want Jenkins to perform



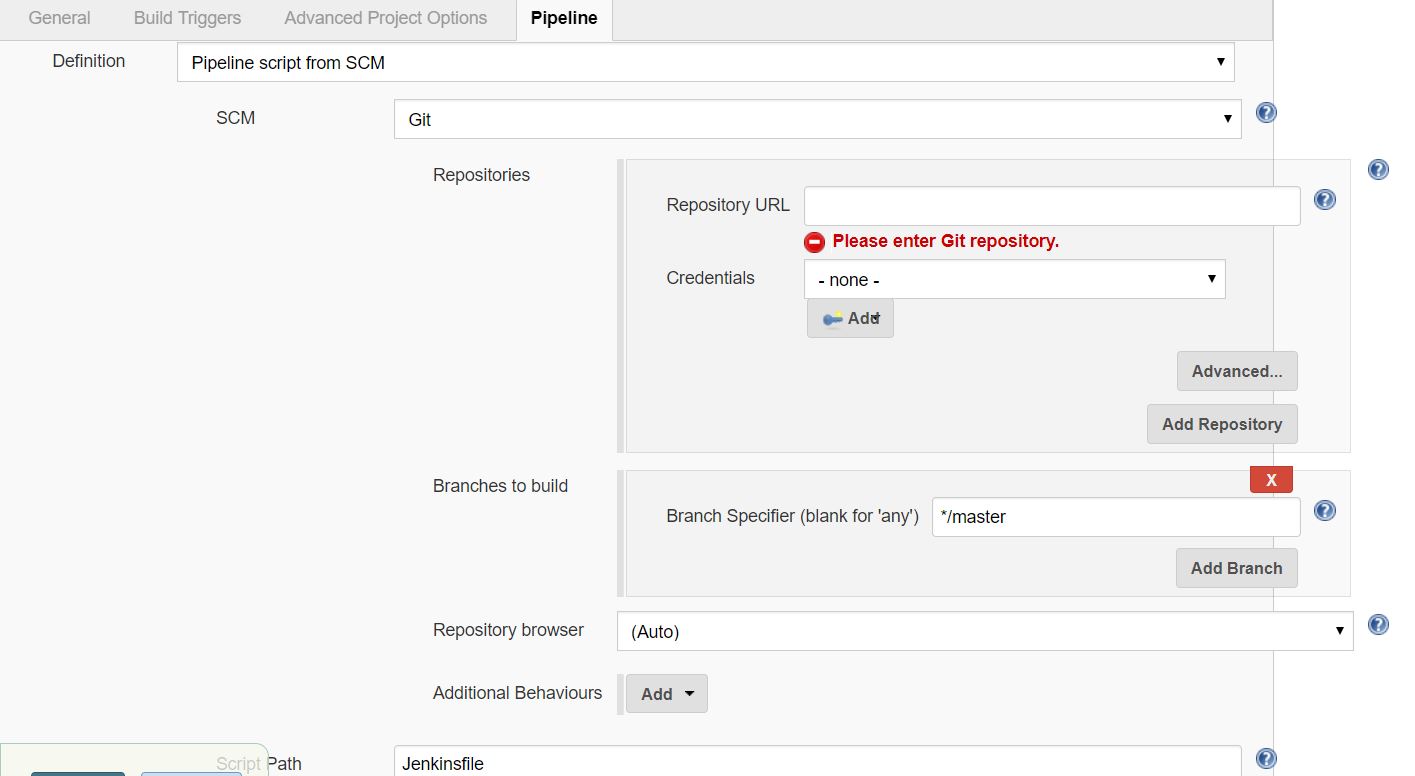


Uses groovy

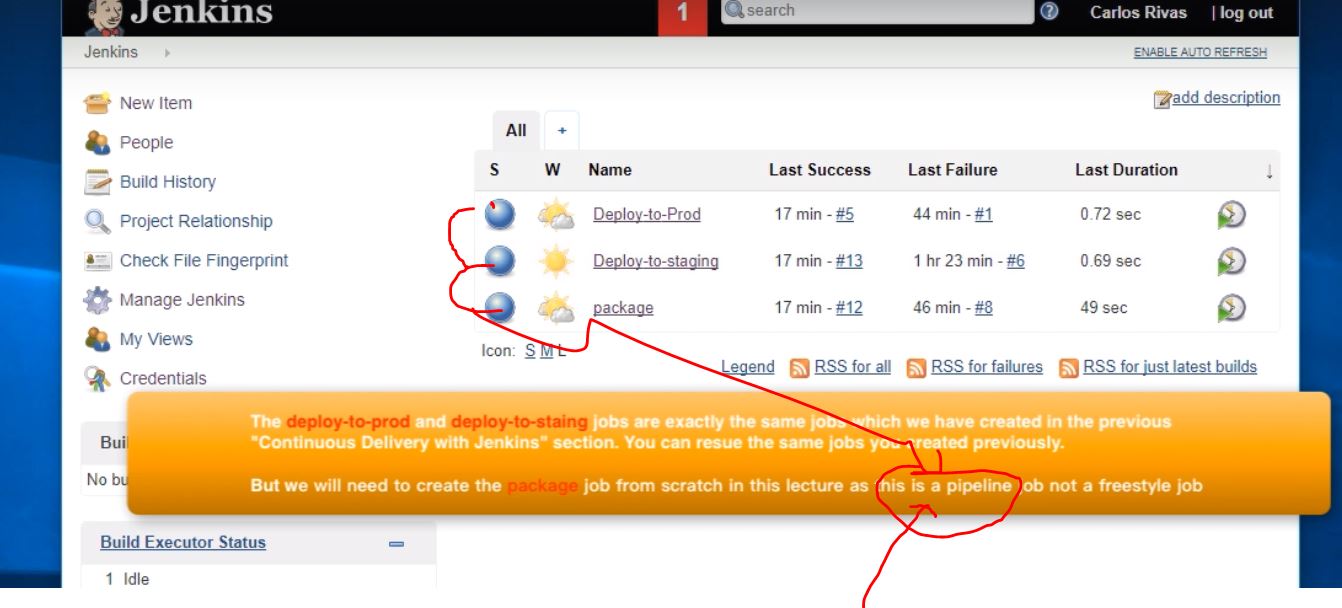
Jenkins dashboard->Manage plugin->download “pipleline” plugin

Create new job ( “pipeplineAscodeExample”)->pipeline->

<https://jenkins.io/doc/book/pipeline/jenkinsfile/>



**Automate Our Existing Jenkins pipeline**

****

rajbi@DESKTOP-P2R5VB3 MINGW64 ~/sandbox/jenkins\_pipelinecode

$ git clone https://github.com/rajbirsood/restWebServices.git

Create Jenkins file

rajbi@DESKTOP-P2R5VB3 MINGW64 ~/sandbox/jenkins\_pipelinecode/restWebServices (master)

$ cat Jenkinsfile

pipeline {

agent any

stages{

stage('Build'){

steps {

echo 'Now building'

bat 'gradlew build war'

}

post {

success {

echo 'Now Archiving...'

archiveArtifacts artifacts: '\*\*/\*.war'

}

}

}

stage ('Deploy to Staging'){

steps {

build job: 'deploy-to-staging'

}

}

}

}

**See deploy-to-staging job**

rajbi@DESKTOP-P2R5VB3 MINGW64 ~/sandbox/jenkins\_pipelinecode/restWebServices (master)

rajbi@DESKTOP-P2R5VB3 MINGW64 ~/sandbox/jenkins\_pipelinecode/restWebServices (master)

$ git add .

rajbi@DESKTOP-P2R5VB3 MINGW64 ~/sandbox/jenkins\_pipelinecode/restWebServices (master)

$ git commit -m "added build job "deploy-to-stage" in Jenkins"

[master 9ddc758] added build job deploy-to-stage in Jenkins

1 file changed, 4 insertions(+), 4 deletions(-)

rajbi@DESKTOP-P2R5VB3 MINGW64 ~/sandbox/jenkins\_pipelinecode/restWebServices (master)

$ git push

Fatal: HttpRequestException encountered.

Username for 'https://github.com': rajbirsood

Counting objects: 3, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (3/3), done.

Writing objects: 100% (3/3), 330 bytes | 0 bytes/s, done.

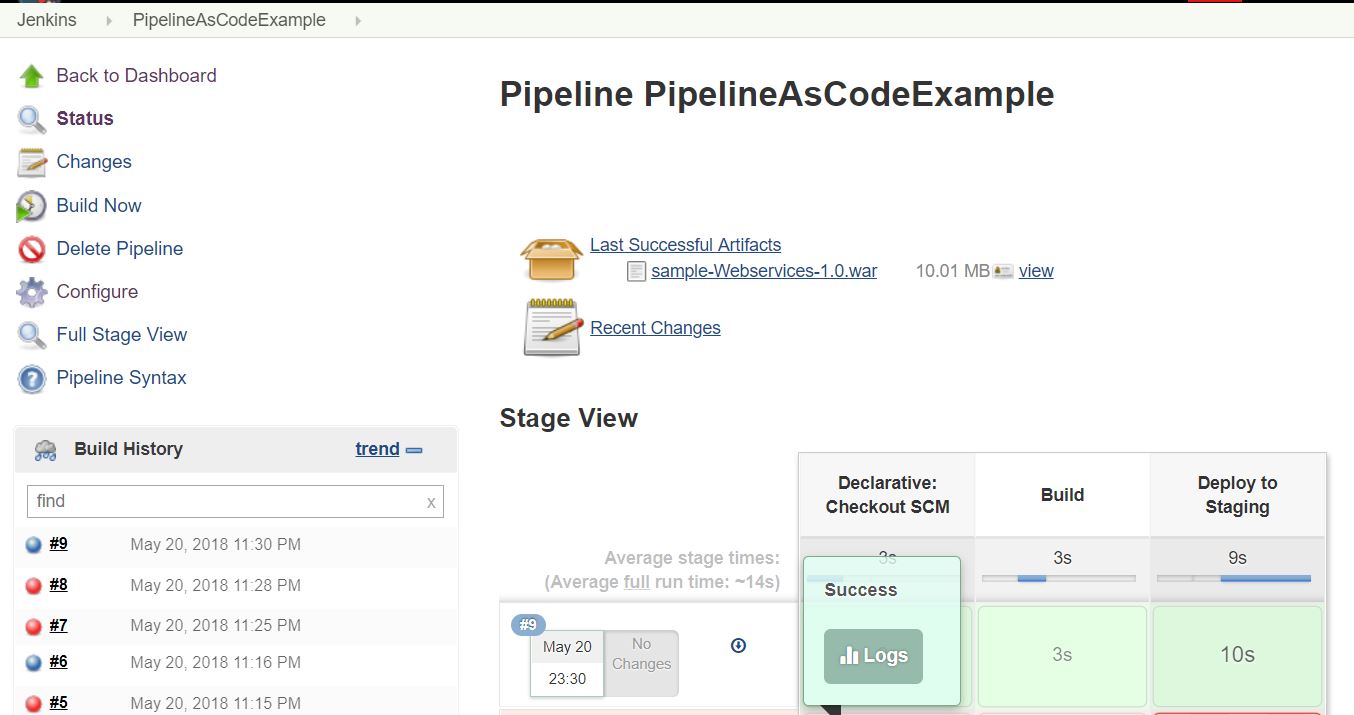
Total 3 (delta 2), reused 0 (delta 0)

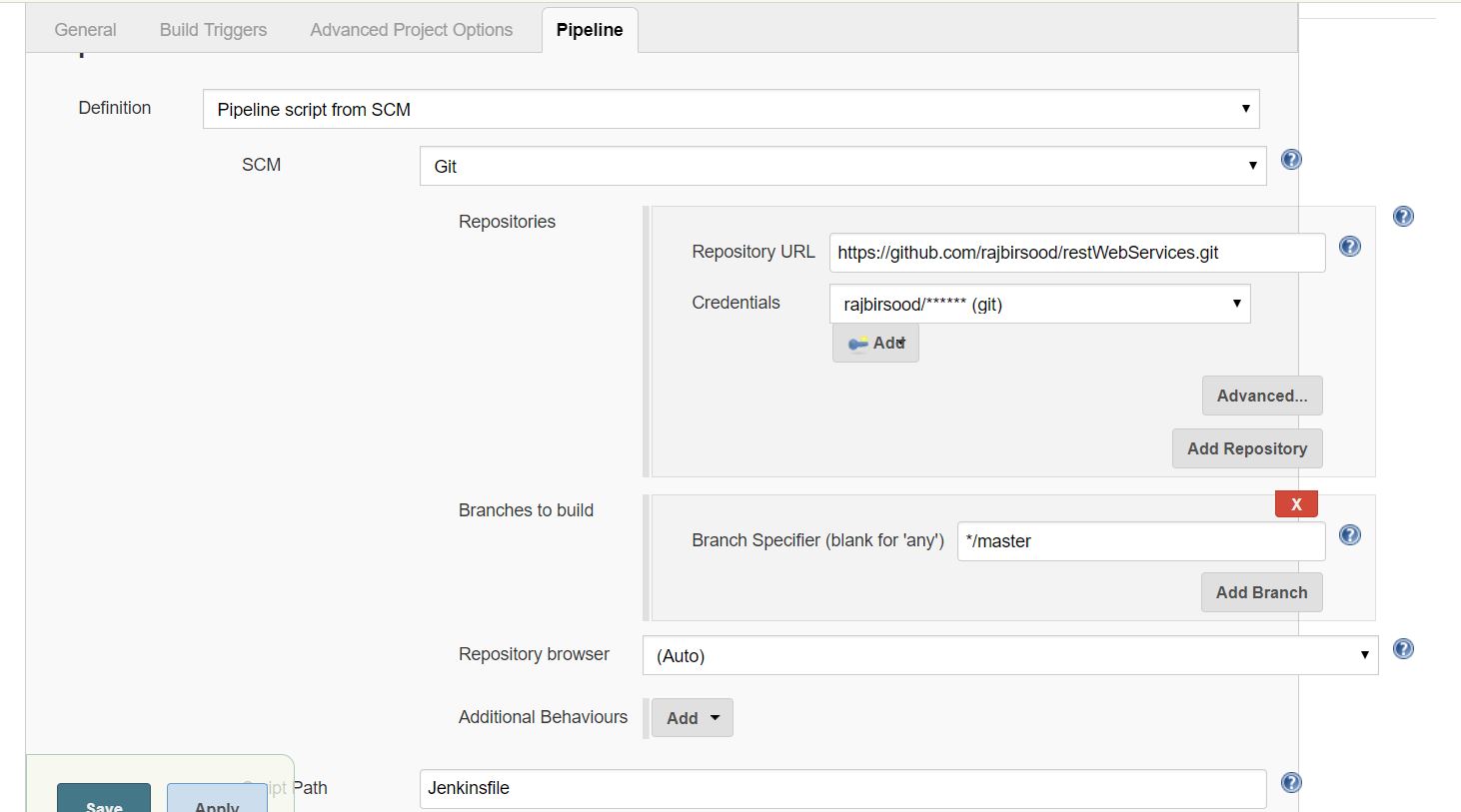
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.

To https://github.com/rajbirsood/restWebServices.git

d5c9832..9ddc758 master -> master

create job

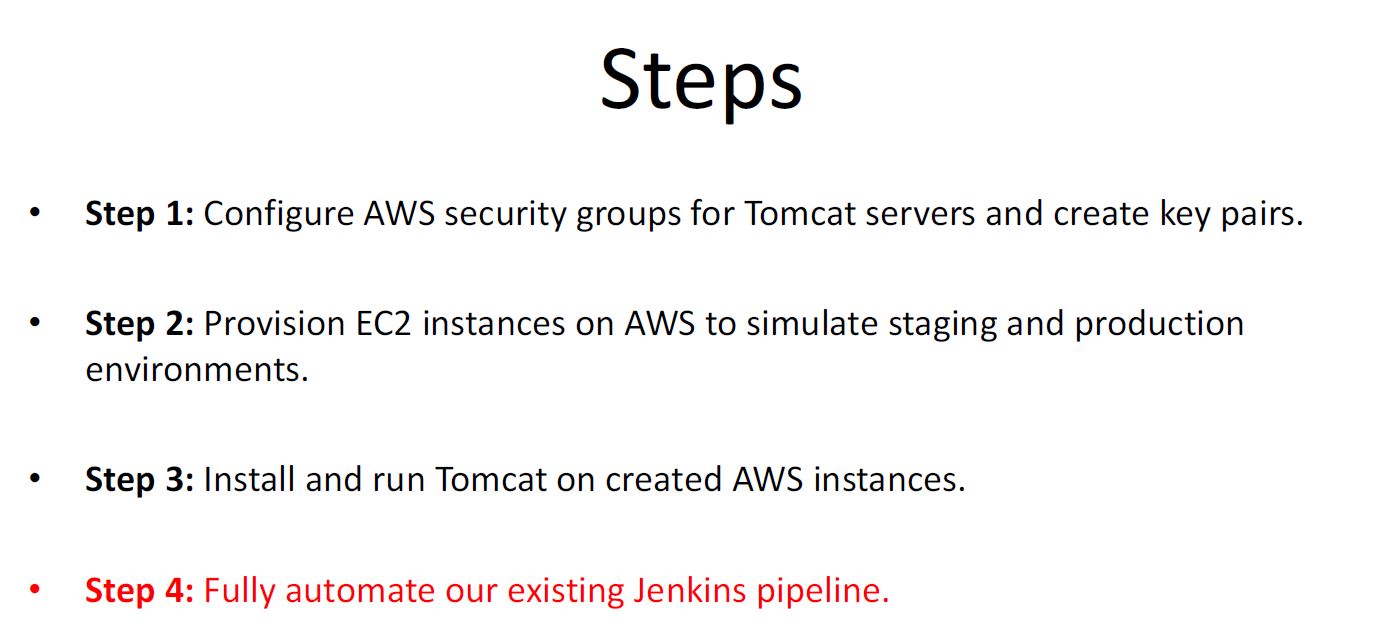


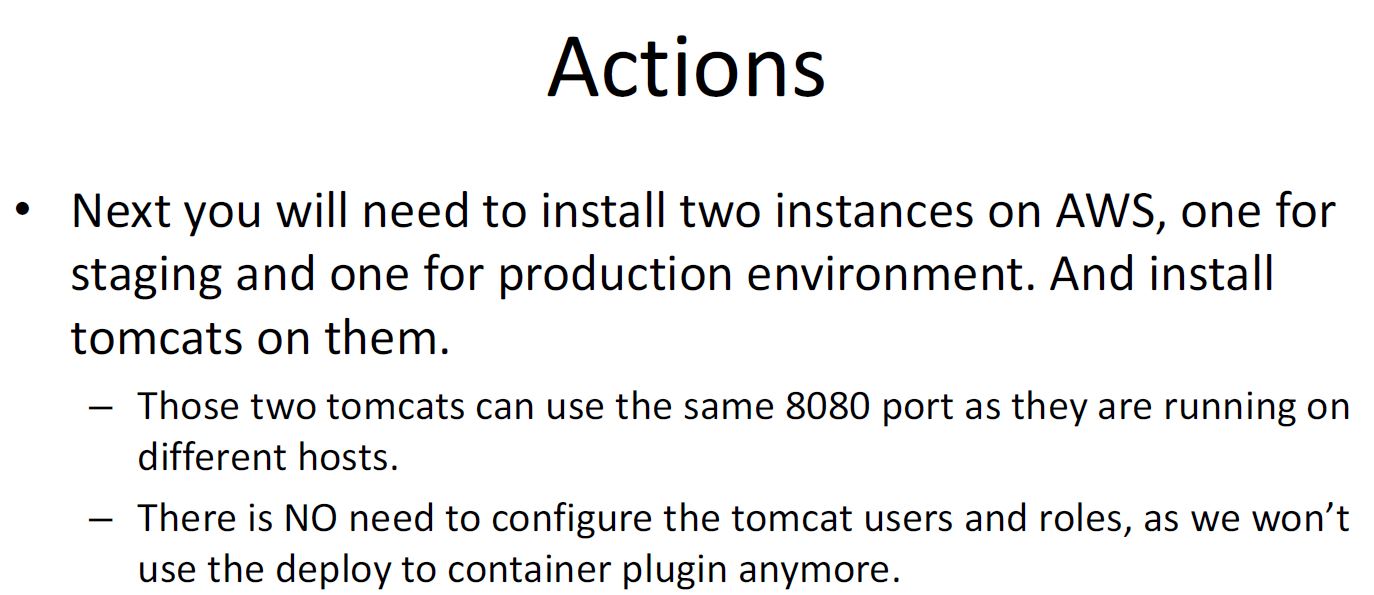




**Fully Automated Jenkins Pipeline ( sec 5, Lecture 47)**

**Download cmder**

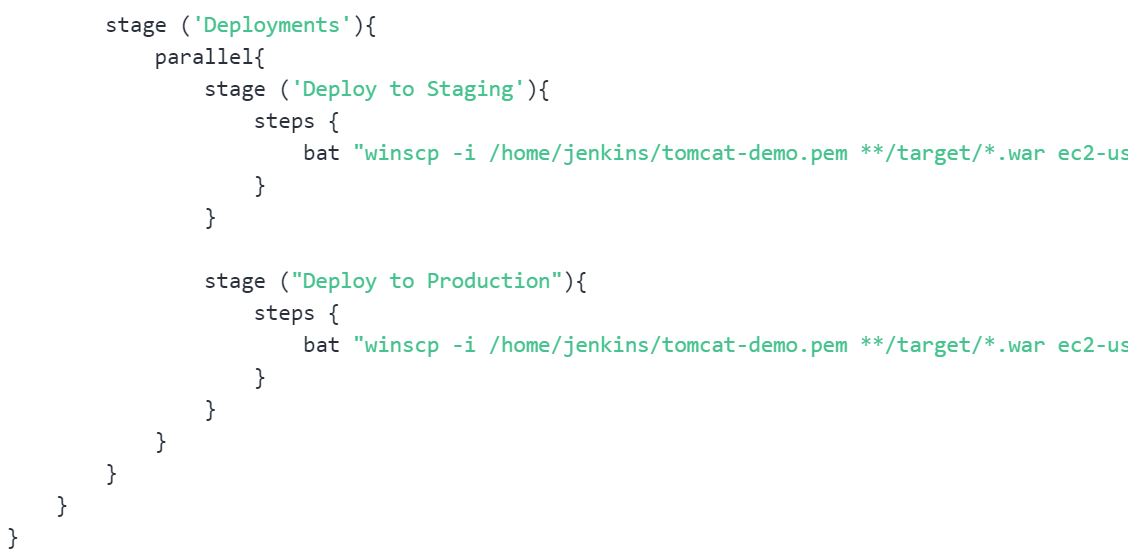
****

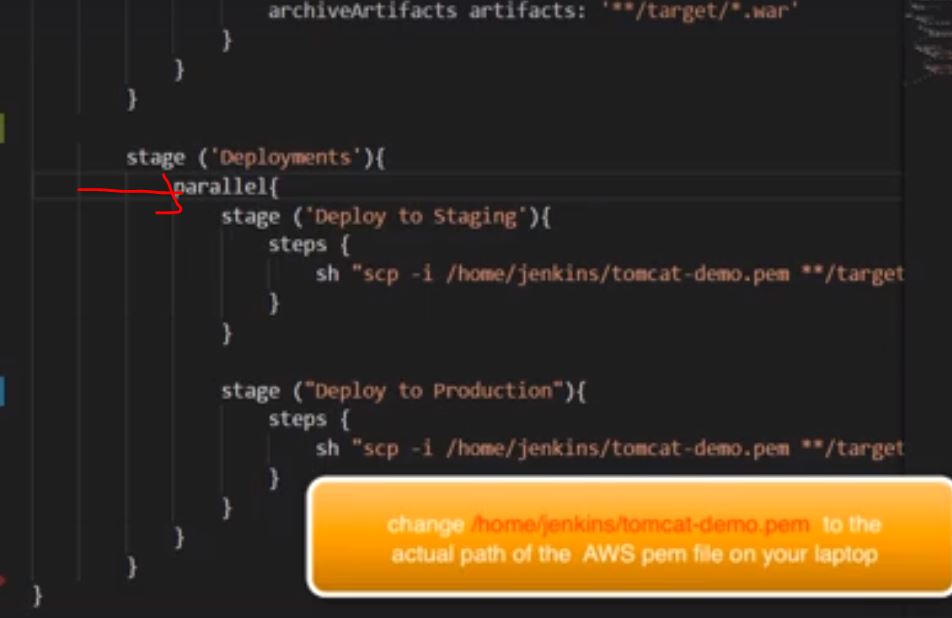
****

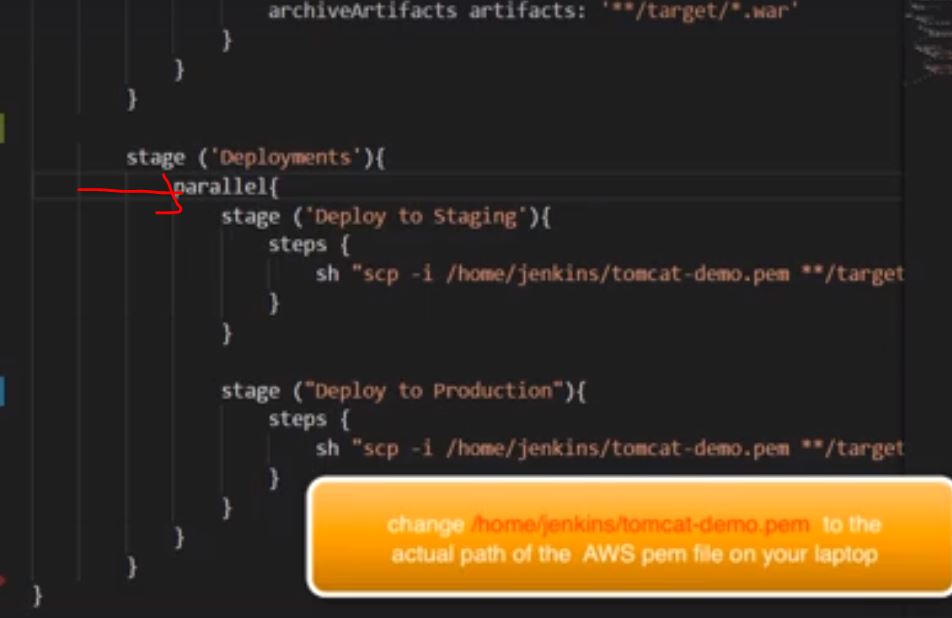
Jenkins file

<https://www.udemy.com/the-complete-jenkins-course-for-developers-and-devops/learn/v4/t/lecture/9385606?start=0>









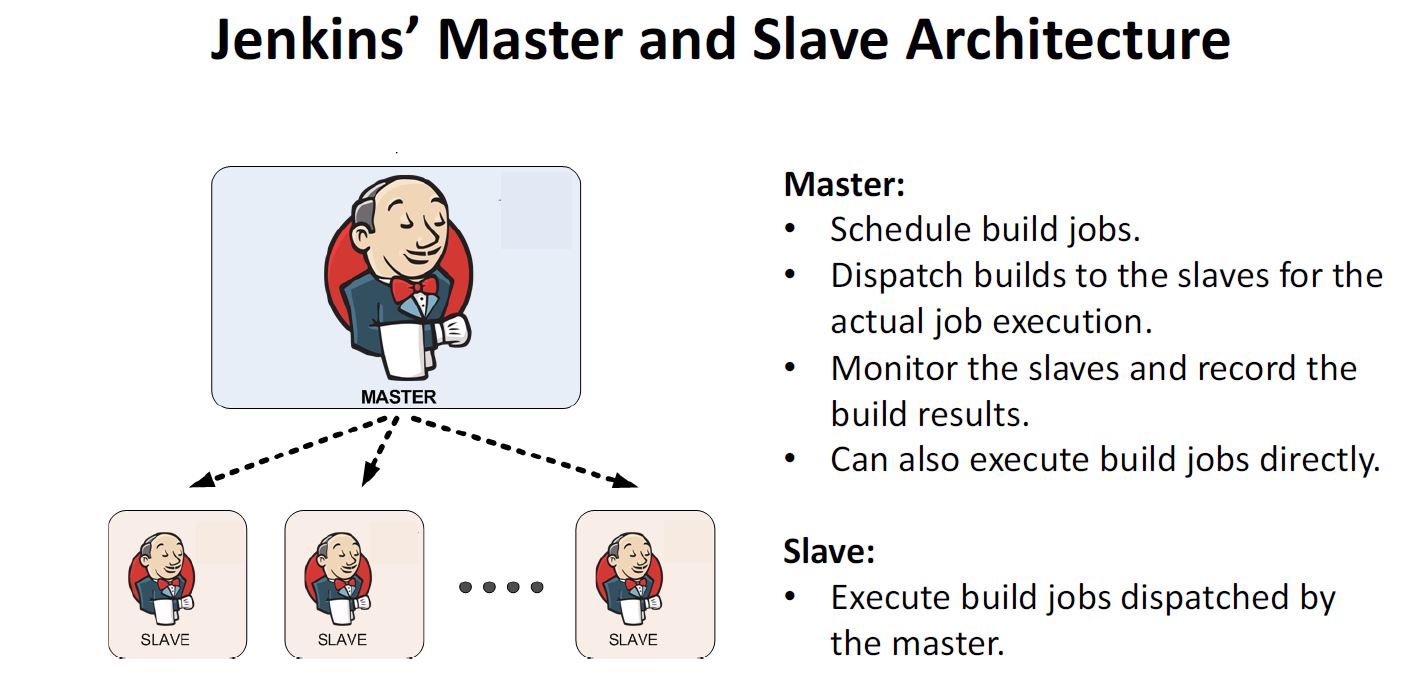
Create pipeline->

Connect to EC2\_machine->

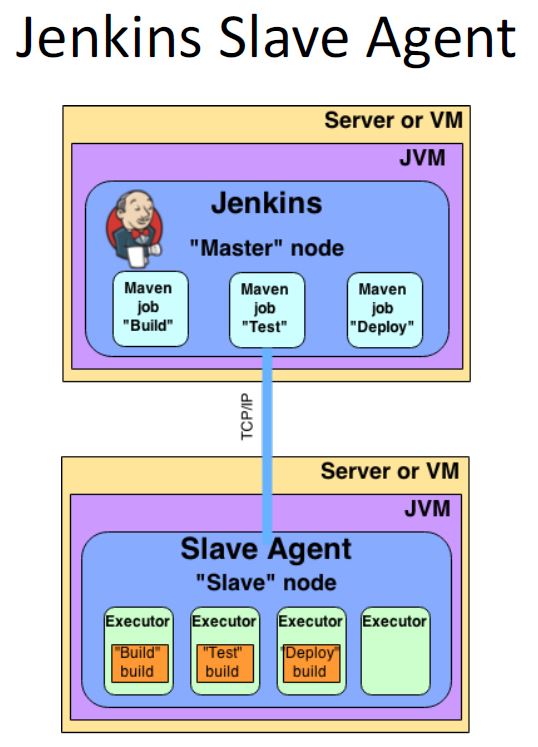
**Distributed Builds** (Section 6 Lec 51)

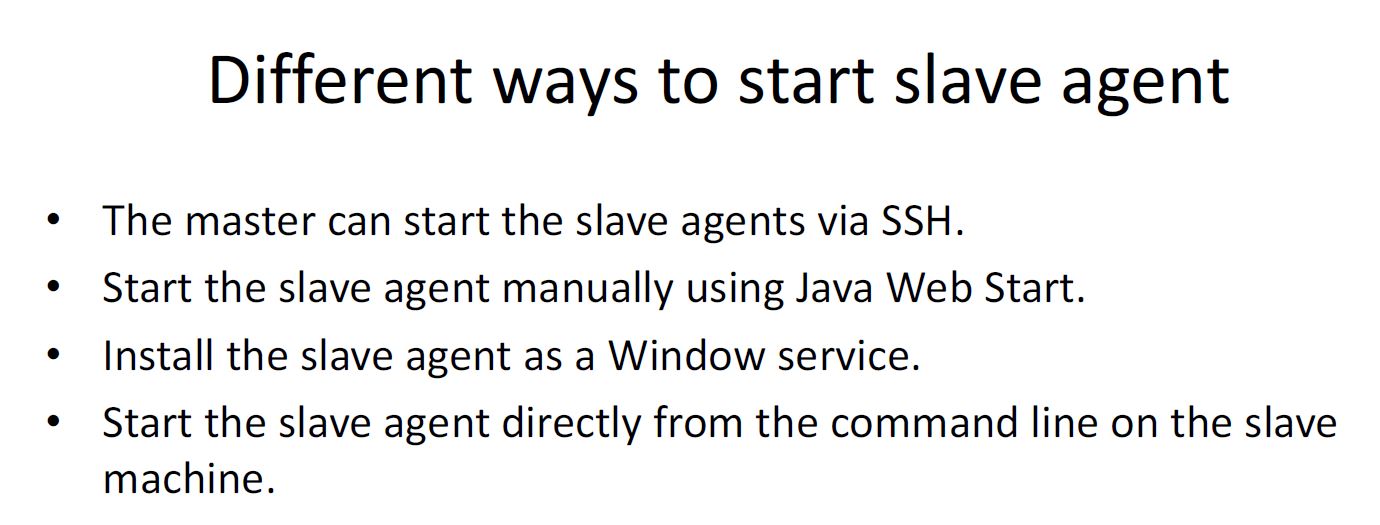
Master/slave Jenkins

**Introduction to Distributed Jenkins Build**

****

**Install Jenkins Master Node in the cloud (lect 52)**

****

****

Best way is to use SSH, Jenkins has its own built in SSH client

**Install Jenkins master in the cloud** (Lect 52)

1. wget -q -O - http://pkg.jenkins-ci.org/debian/jenkins-ci.org.key | apt-key add -
3. echo deb http://pkg.jenkins-ci.org/debian binary/ > /etc/apt/sources.list.d/jenkins.list
5. apt-get update
7. apt-get install jenkins

**Install Jenkins slaves in the cloud and form a**

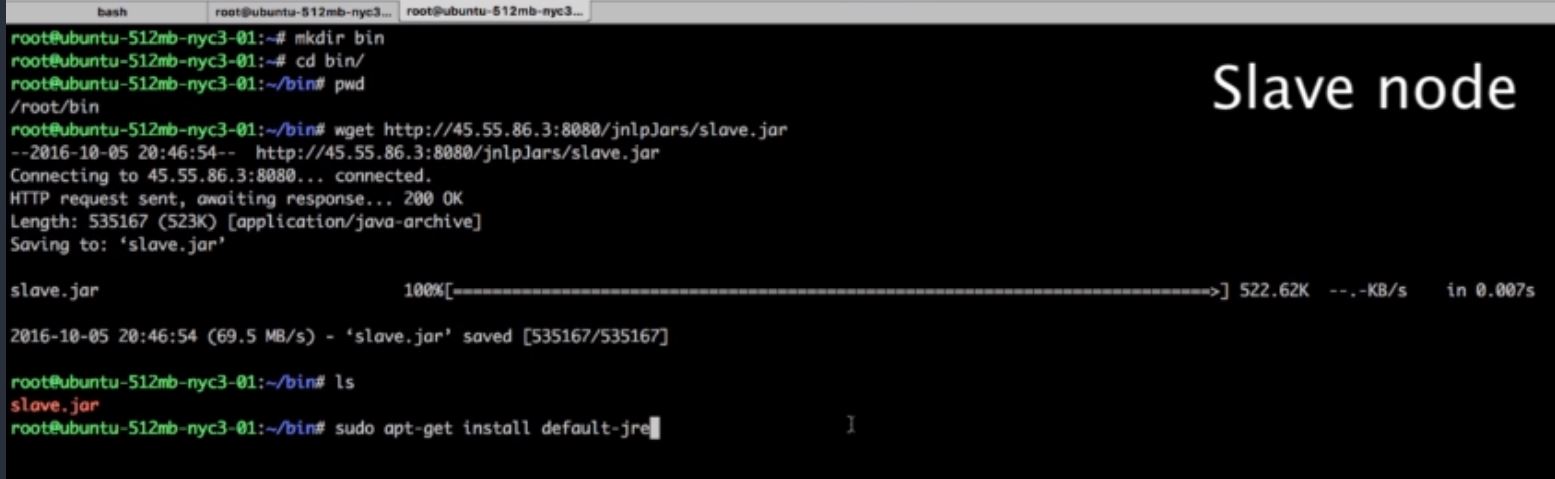
**Jenkins cluster**

Install Jenkins Agent on the Slave Node

**running on master node:**

1. sudo -iu jenkins
3. ssh root@<slave\_ip> mkdir -p .ssh
4. enter password
5. cat .ssh/id\_rsa.pub | ssh root@<slave\_ip> 'cat >> .ssh/authorized\_keys'
6. enter passeord

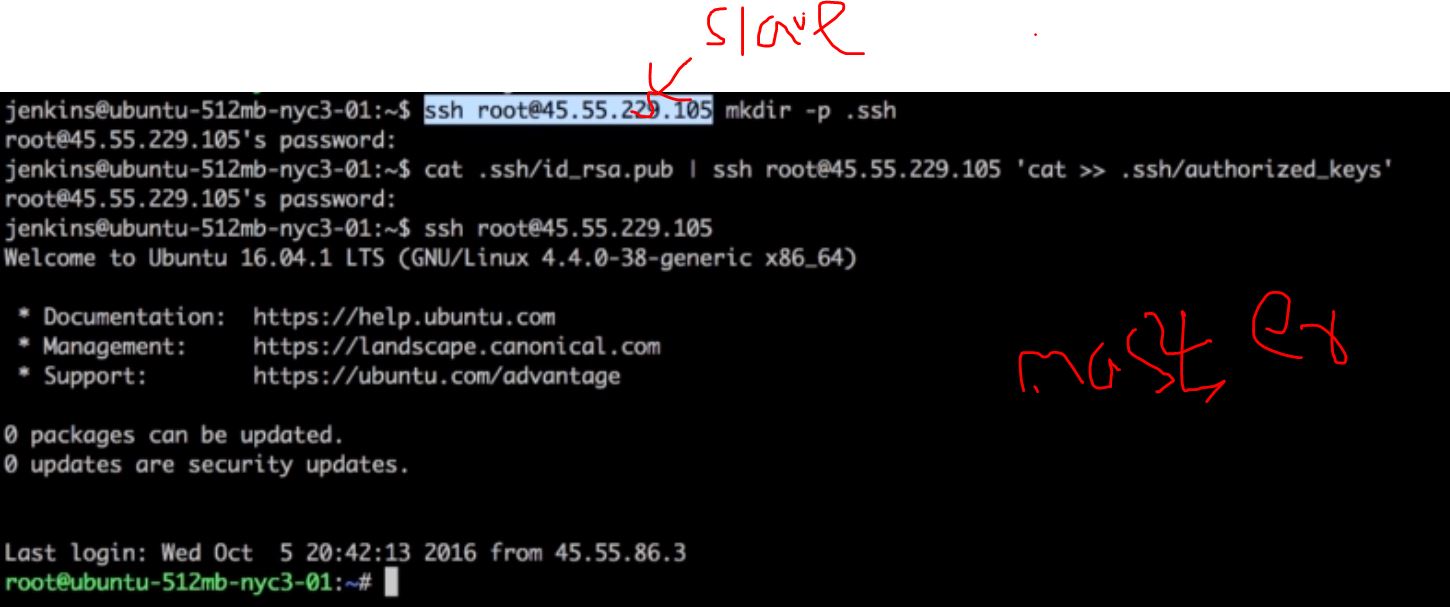
**running on slave node:**

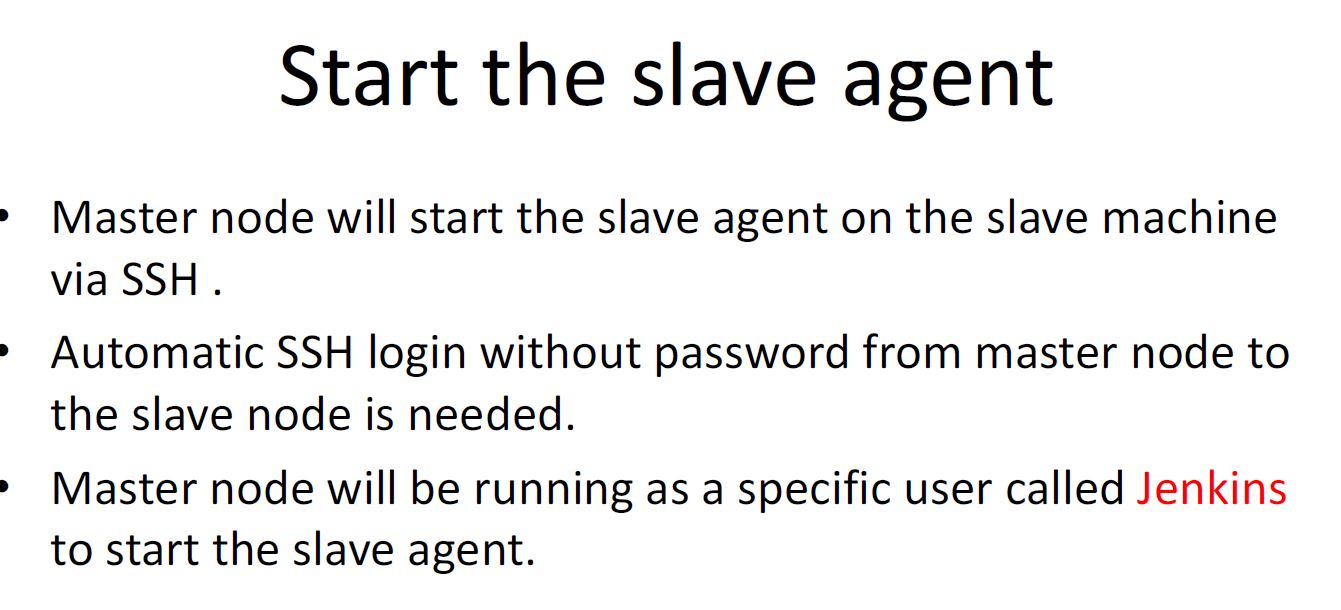


1. mkdir ~/bin
3. cd bin
5. wget http://<master\_ip>:8080/jnlpJars/slave.jar
7. sudo apt-get install default-jre

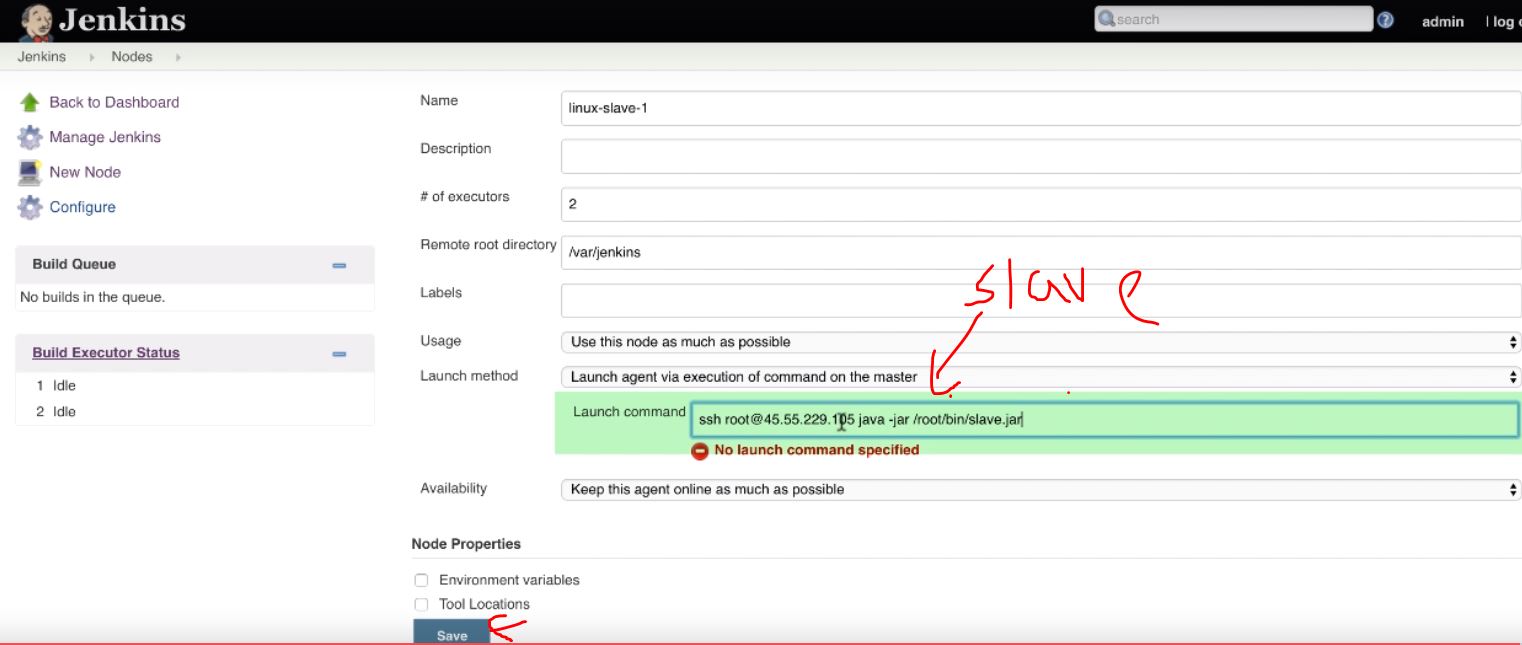
**start slave agent:**

ssh root@<slave\_ip> java -jar /root/bin/slave.jar



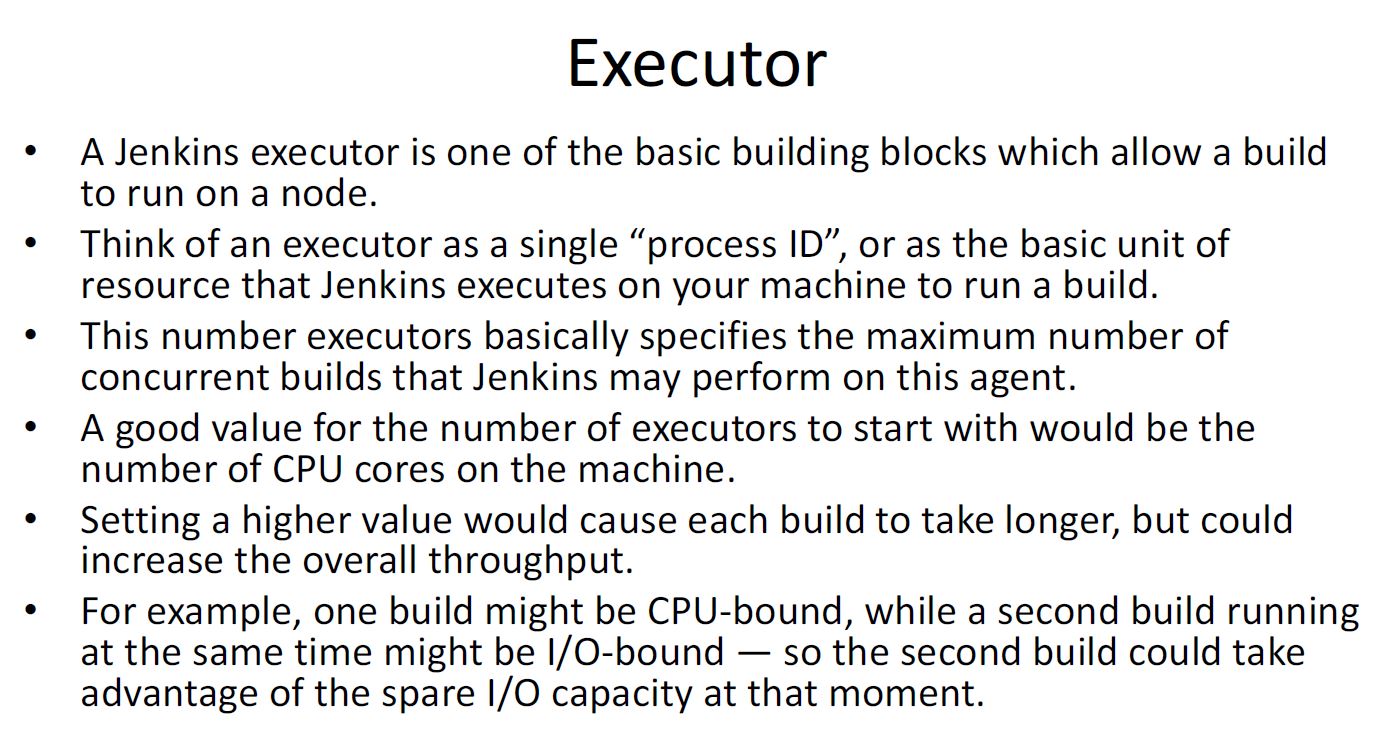
****

Manage Jenkins->Manage nodes->new node->give name->



U will see slave node is not activated yet, then refresh the page



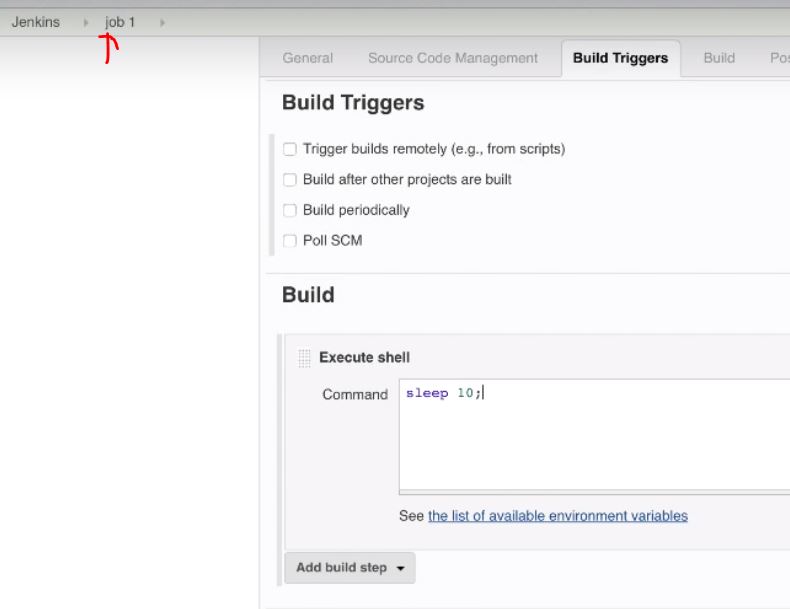


**Concurrent Builds on Jenkins Cluster**

**Label Jenkins Nodes**

Create 3 simple build job

Job 1->Execute shell ( sleep 10)

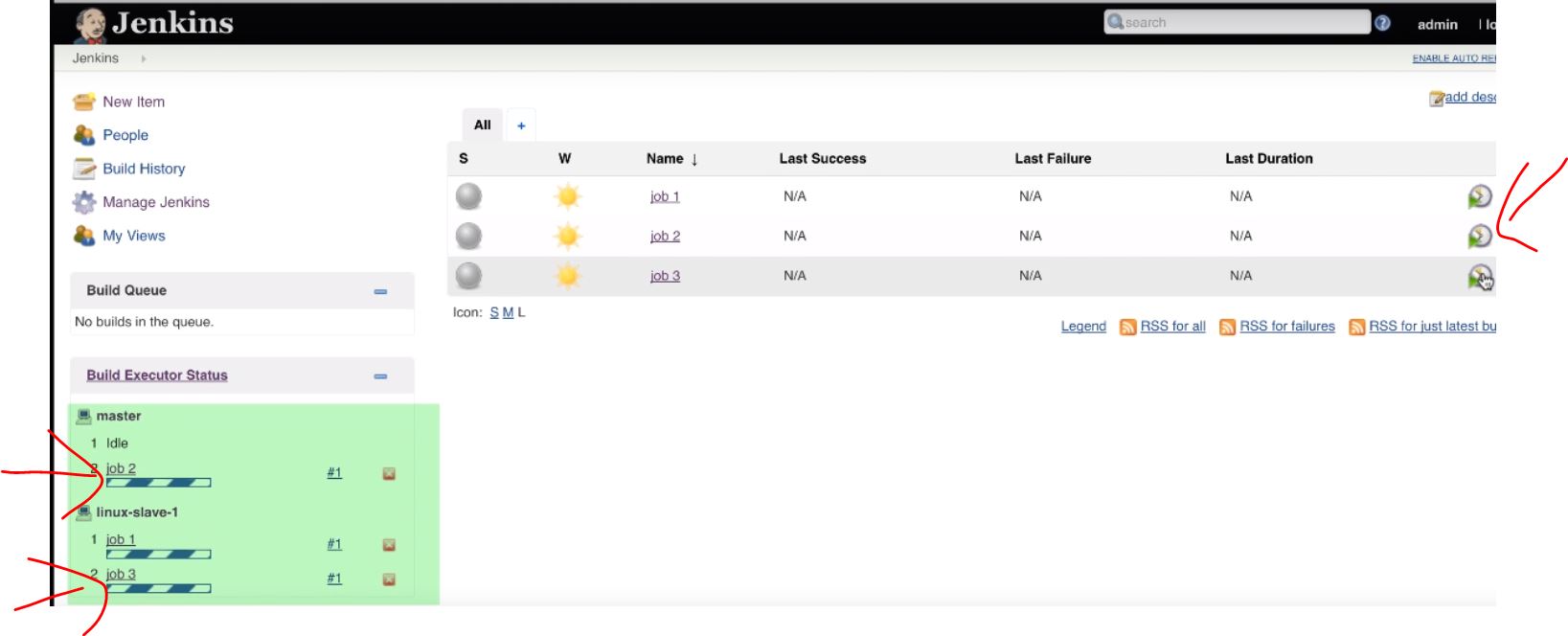


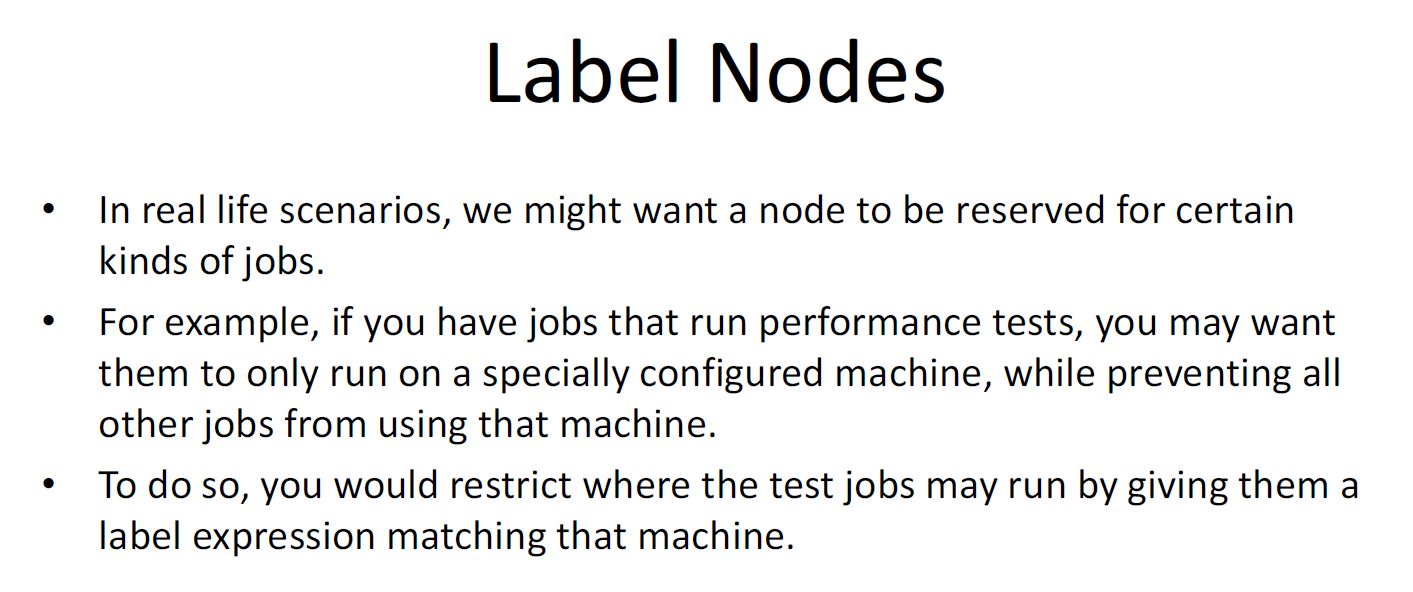
Similarly 2nd job, sleep 10 sec

3rd job, sleep 10 sec

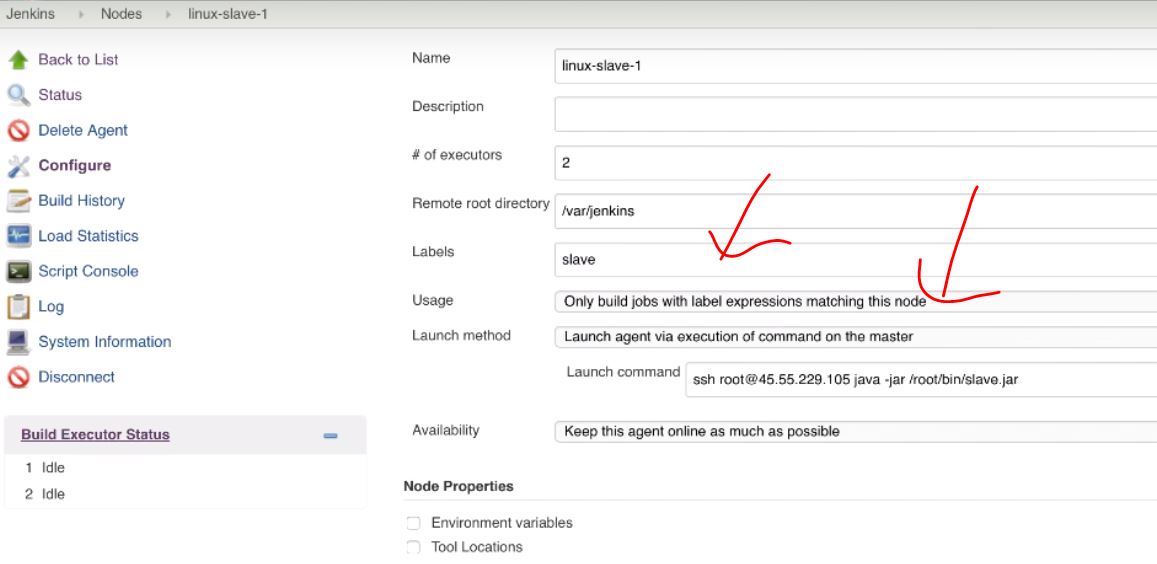
Trigger 3 jobs, two one slave and 1 in master

Master node do scheduling of jobs

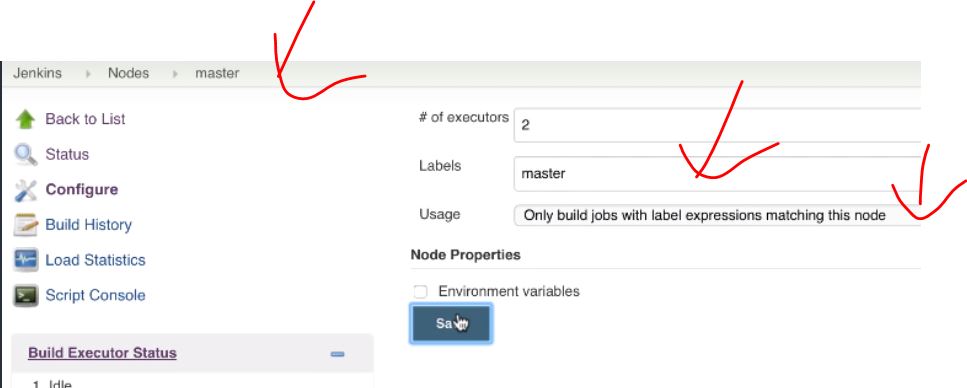




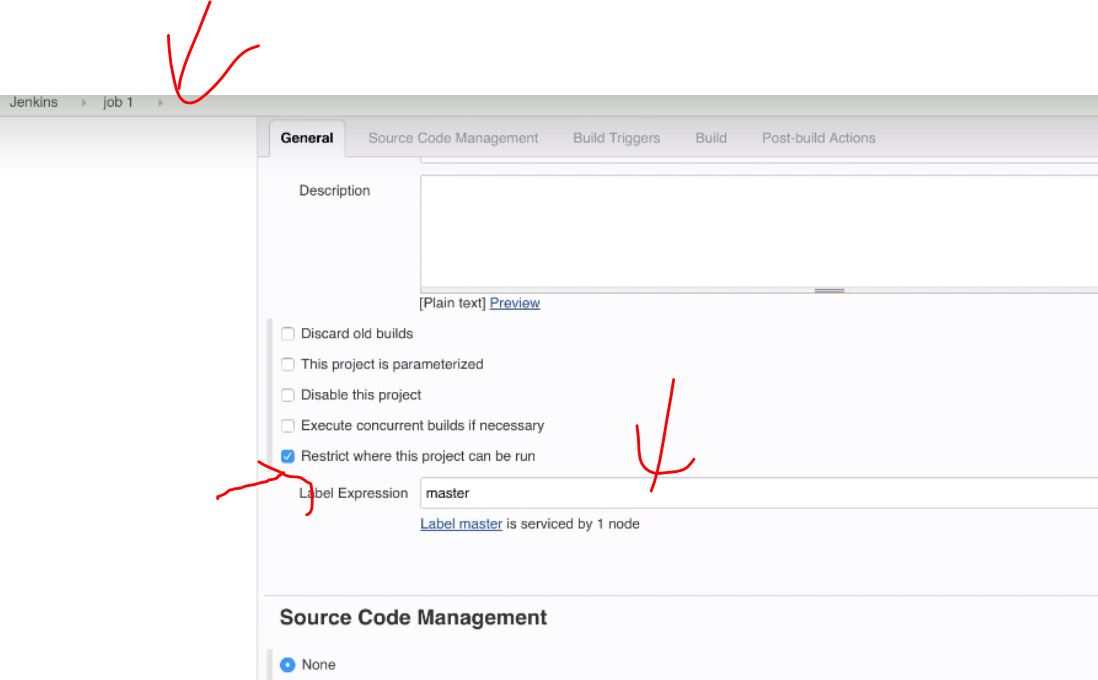
Manage Jenkins->manage nodes->select slave node->configure->

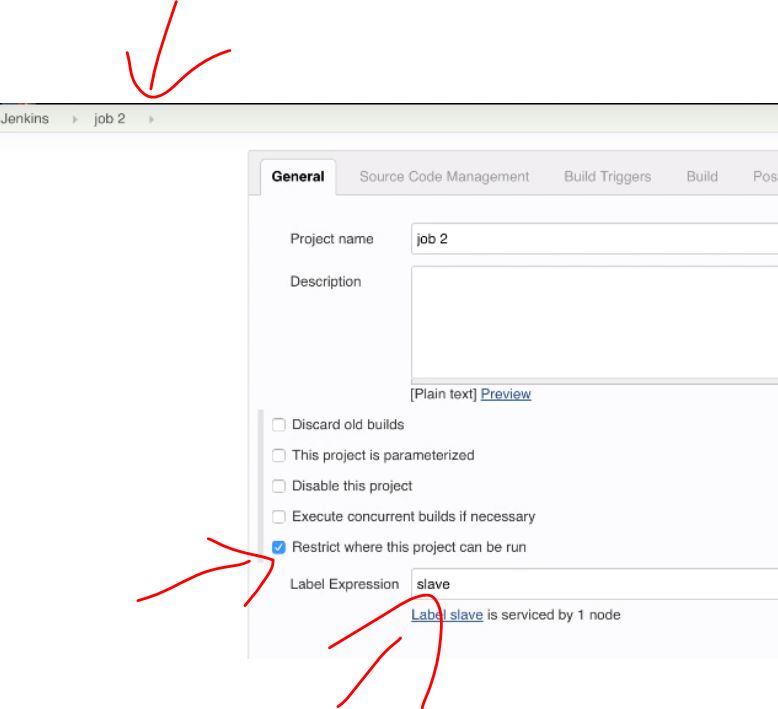


On master node



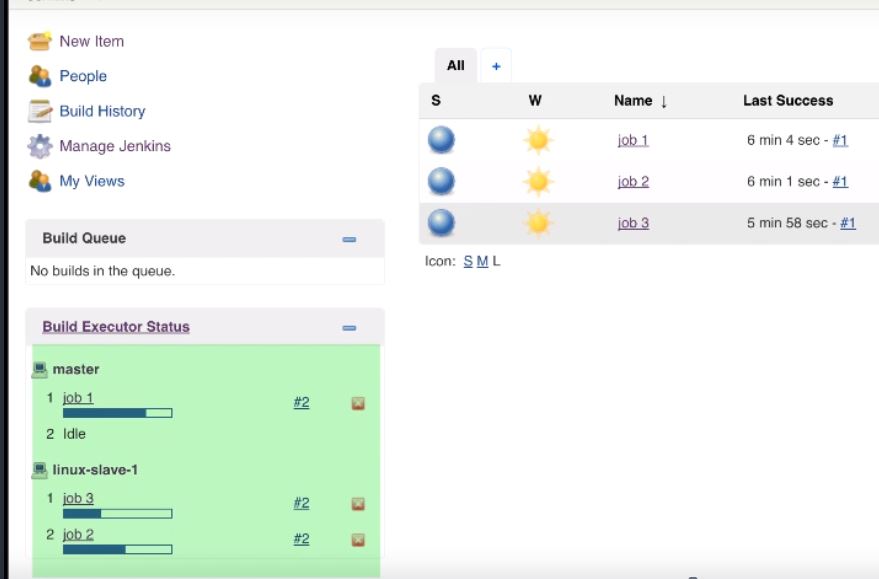
**Now label the job**

****

****

**Same for job 3**

****

****