Jenkins:

Imp Links:

https://github.com/prakashk0301/angular-realworld-example-app

https://github.com/prakashk0301/maven-spring-boot-java-app

https://github.com/prakashk0301/spring-boot-mongo-docker

Python sample project: <https://github.com/edonosotti/ci-cd-tutorial-sample-app>

1: what is CI/CD (Continuous Integration and Continuous Delivery)

2: Approach to achieve CI/CD,

3: Tools

- CI tools

- Build tools -> install tools on Jenkins(build agent) (JAva:- maven/ant/gradle , DotNet:- >MSBuild/nuget , NodeJS/JAvaScript/AngularJS :- > NPM , Python:-> python)

- Deployments tools

- static code analysis tool

- artifactory managers tool

- testing tool

4. pipeline method

5. CI/CD java

6. CI/CD NodeJS

7. CI/CD dotnet

[CI-CD basic flow.jpg](https://drive.google.com/open?id=1QFlCkqJ-yFwyNKEUinDwpbpNpQmzuWtk&usp=drive_copy)

----

Docker:

Jenkins Docker integration

Docker based CI/CD python

Docker CI/CD Java

----

Kubernetes

Jenkins Kubernetes, Container Registry integration

K8s based CI/CD

k8s CI/CD Java

----

Ansible

Jenkins Ansible integration

CI/CD pipeline

----Terraform

Jenkins Terraform integration

========================Install Jenkins on Amazon t2 micro , Jenkins works on port number 8080 (open ports in SG 8080, 443, 80, 22 & set source to Anywhere)

Ref link: https://www.jenkins.io/doc/book/installing/

Jenkins Job/Jenkins Pipeline/Item

what is plugin: plugin is nothing but a piece of code/small software that allows us to use that feature.

ex: if you want to integration jenkins with docker then install docker plugin

Freestyle: you can configure the stage manually(you have to tell jenkins how to clone the code, how to build the code,how to run unit test cases, how to compile the code and so on ....)

Freestyle is compatible with any language :

we prefer freestyle for automation: execute shell script/powershell/command, infrastructure provision(terraform),

general ---> description

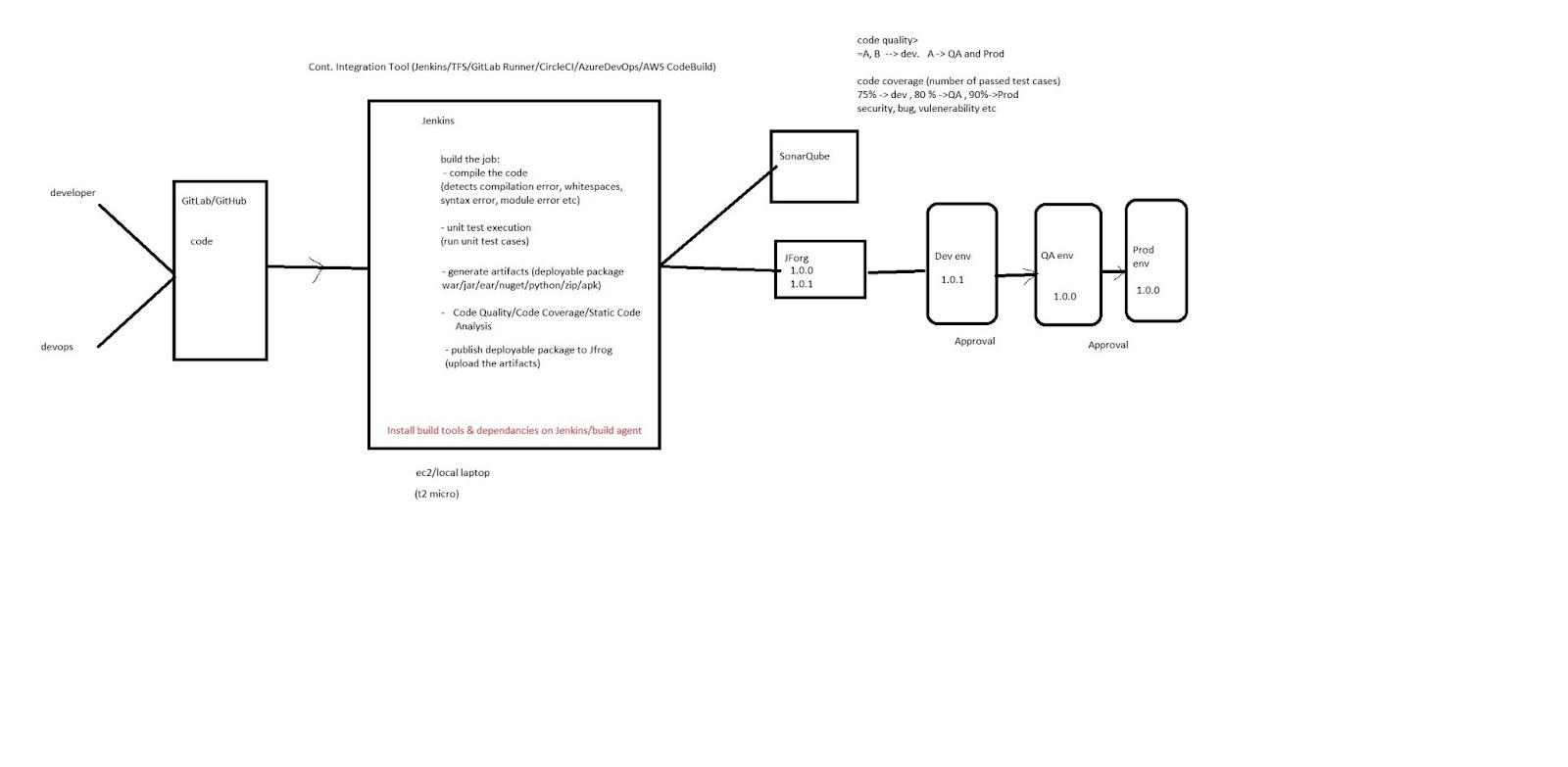
source code section ---> GitHub/Gitlab URL and branch

build trigger ---> how to trigger the job, you can trigger manually, specific time period

build env section ---> parameter or parameter can be passed. run this script on any specific location in pune

build section ---> how to build, you can choose maven,ant, shell, powershell, linux command

post build section ---> operation after build activity ex: you can trigger an email to stakeholders



Groovy DSL: Pipeline job (70%)

—---------------------------- 26-11-2022

**Maven build tool**:- maven is one of the build tool, we prefer maven to **build** **java** related projects

(compile the code, execute unit test cases, generate artifact or deplorable package, run integrate test)

1. Install maven package of build server (build agent: where you can run commands)

-> in our case build agent is jenkins itself (build job on jenkins server)

sudo wget http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo

sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo

sudo yum install -y apache-maven

mvn --version

1. Install maven plugin

Manage jenkins -> plugin -> installed/available/update

[Maven Integration](https://plugins.jenkins.io/maven-plugin)

[Maven Release Plug-in](https://plugins.jenkins.io/m2release)

[Maven Invoker](https://plugins.jenkins.io/maven-invoker-plugin)

(what is jenkins plugin: plugin is piece of software, that allows us to integrate the tool/use the feature)

Then install without restart

1. Configure tool or declare executable path.

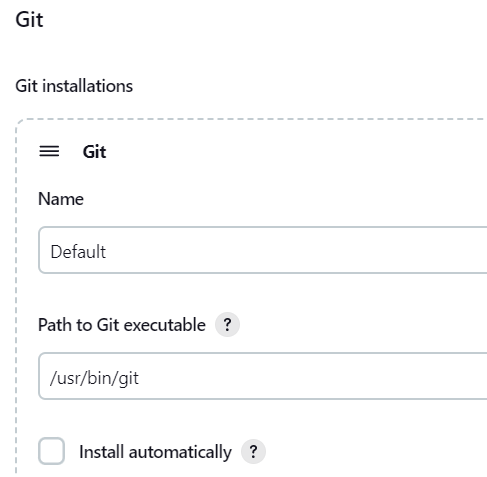
Run **mvn --version** on linux terminal: > to get Maven and JDK path

**which git** -> to get git installed directory

Copy maven path

Jenkins dashboard -> manage jenkins -> configure tool -> declare maven environment path , JDK path, Git





Q. what is maven **lifecycle rule/ maven phases/maven goals**

Ans: the default Maven lifecycle consists of 7 phases.

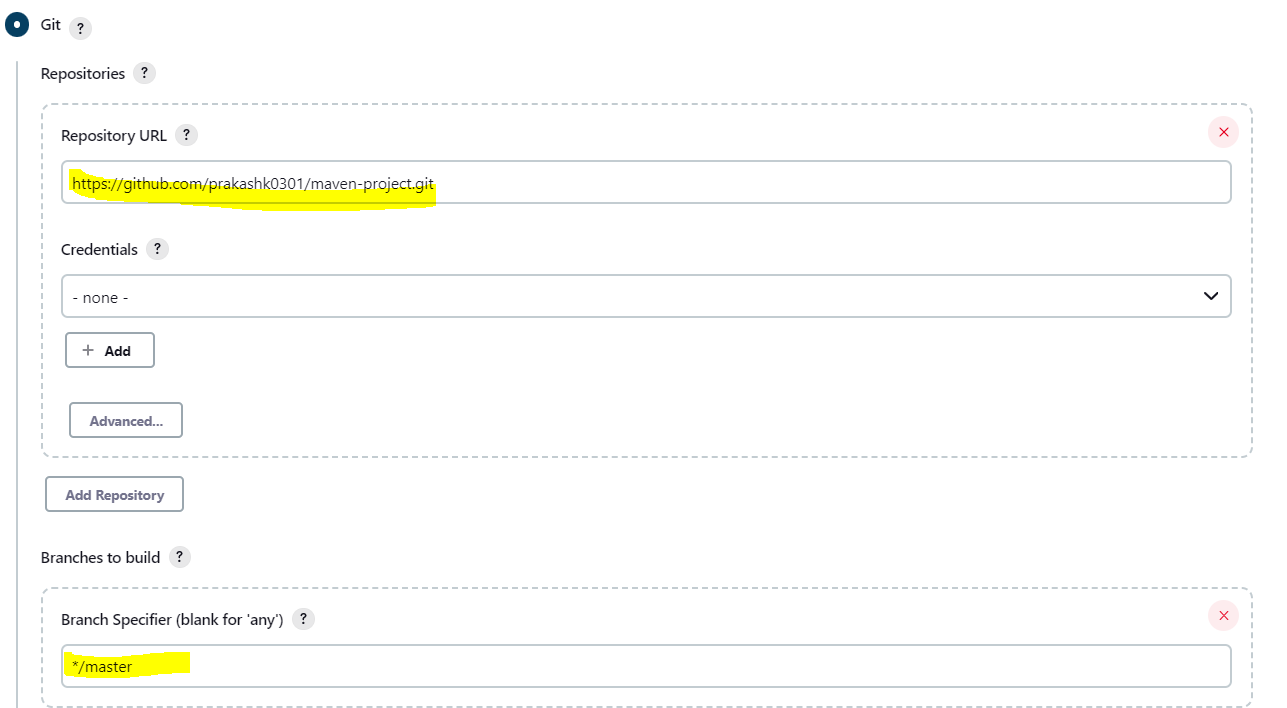
1. **validate**: this step validates if the project structure is correct or not.

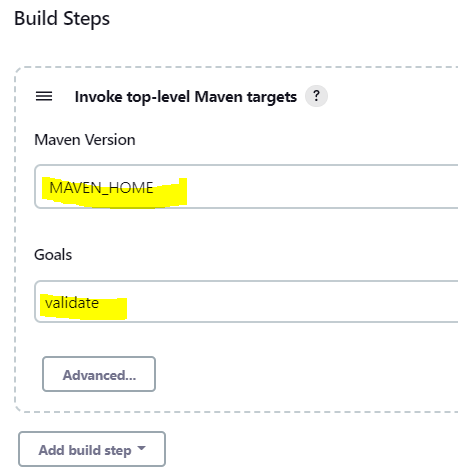
(it check for pom.xml is present or not, it checks if all the dependencies have been downloaded or not)

Pom.xml: project object management -> it contains all the project dependencies(how to build the code, tool name, tool version, download tool from repository or URL to the build agent)

How to create the jenkins job:-

Github link: <https://github.com/prakashk0301/maven-project>





1. **compile**: it compiles the code, it checks code error, syntax error, whitespace

It converts the **.java file to .class classes** (it converts from high level to low level so that system can understand the code)

(by default maven executes all the previous life cycle or it executes life cycle in sequence )

1. **test**: it executes all the **unit test framework** or unit test cases written by the dev team. This stage can be performed before you can create the artifact.

{code coverage : it checks for unit test case report, lets says 75% test cases are passed then code coverage would be 75% , and based on this condition we can do deployment

70%-> dev

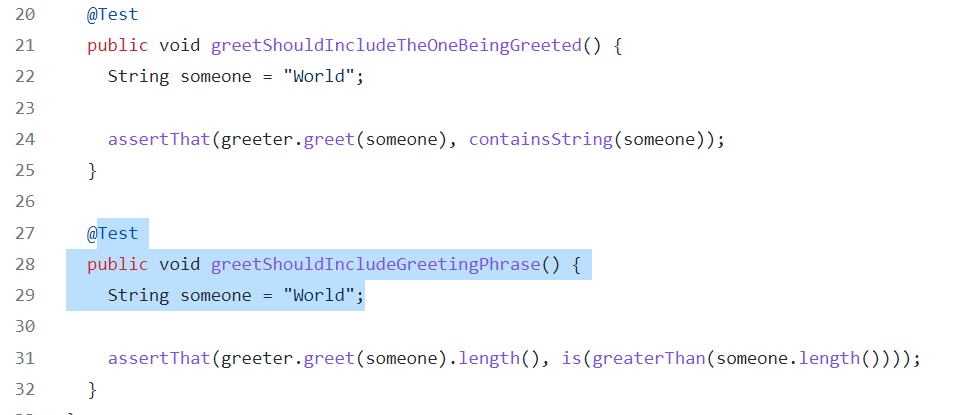
80%-> QA

85 or 90%-> Prod}

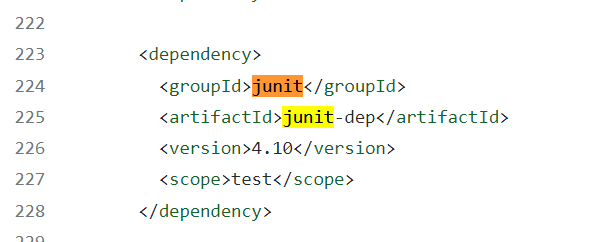
We can mention unit test execution tool name in pom.xml (junit/selenium/Nunit etc)

(by default maven executes all the previous life cycle or it executes life cycle in sequence )

There are 2 test cases: <https://github.com/prakashk0301/maven-project/blob/master/server/src/test/java/com/example/TestGreeter.java>

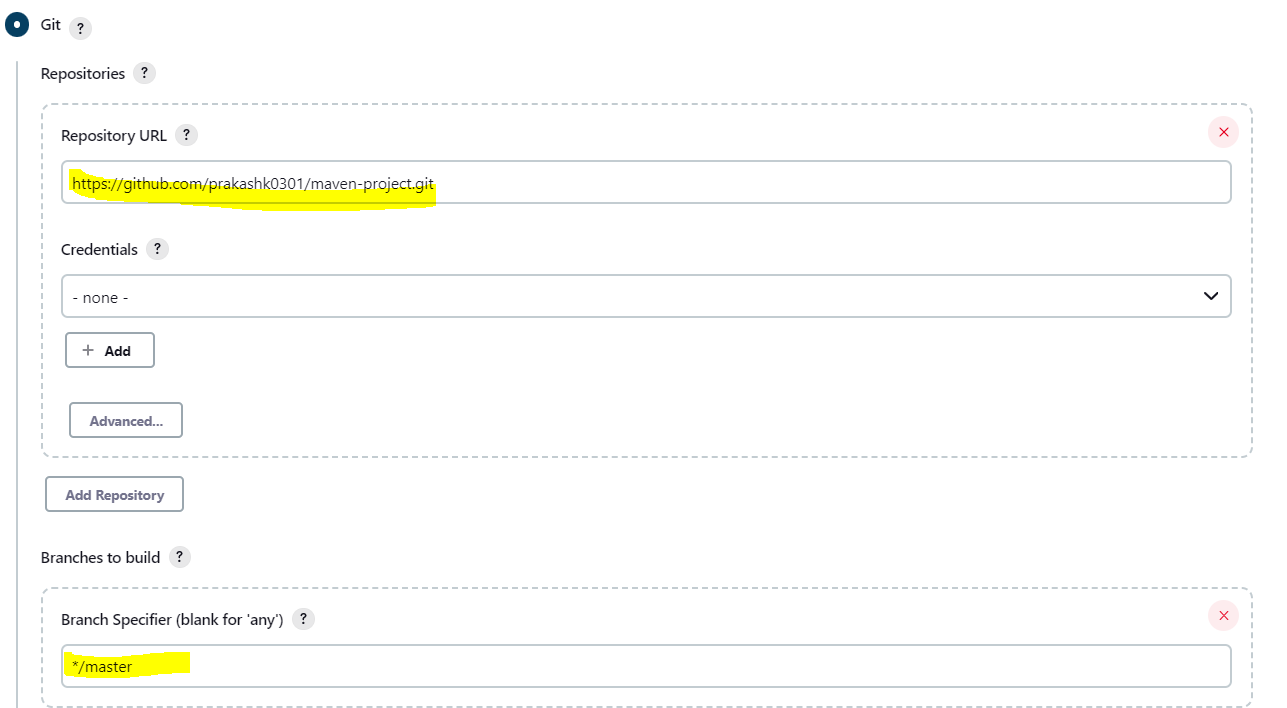


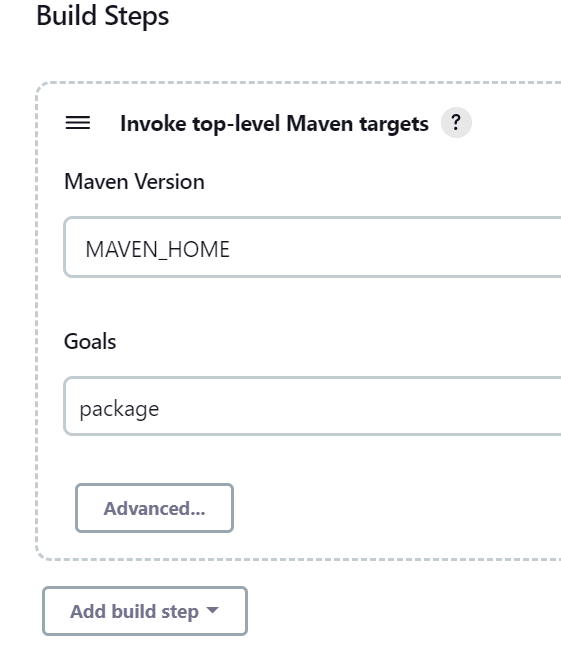
To run unit test cases we are using Junit. <https://github.com/prakashk0301/maven-project/blob/master/pom.xml>



1. **package**: once your code is compiled and test framework execution done you can generate artfact or deployable packages.

take the compiled code and package it in its distributable format, such as a JAR or WAR or EAR.





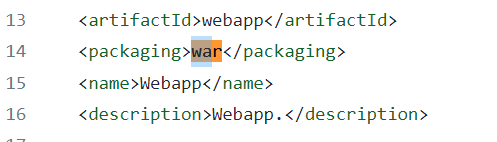
WAR (artifact) file can be deployed to dev/QA/prod

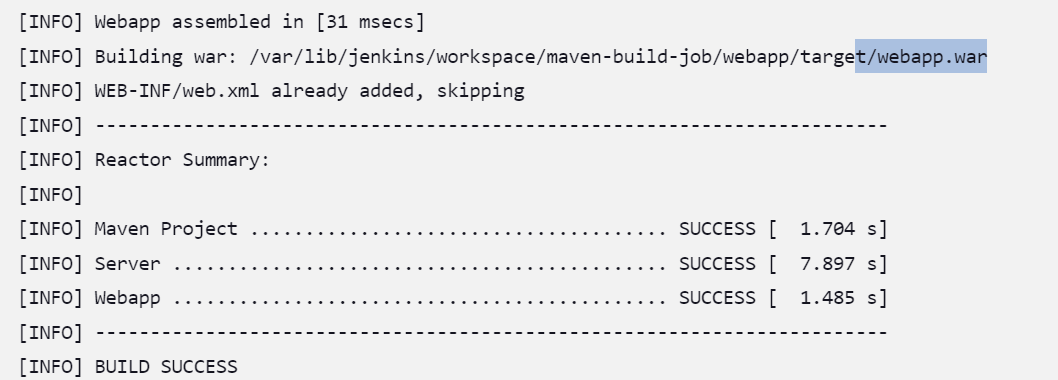
Files contains required libraries to run the code/App

WAR: web archive (when you want to deploy websites/UI/dashboard )

Artifact format is mentioned in pom.xml

<https://github.com/prakashk0301/maven-project/blob/master/webapp/pom.xml>





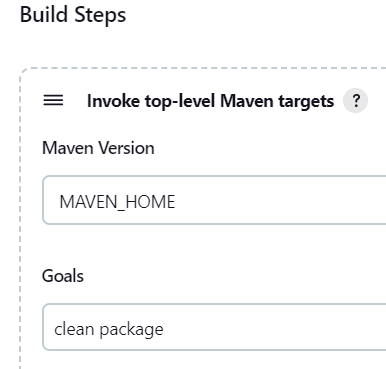
1. **verify:** it verifies the generated artifacts, the artifac has all the required files (libraries) or not.

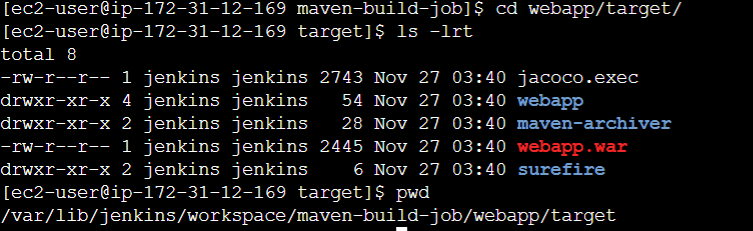
**Problem statement:**

1. Jenkins is not cleaning workspace directory on every new build (it uses old catches to build )

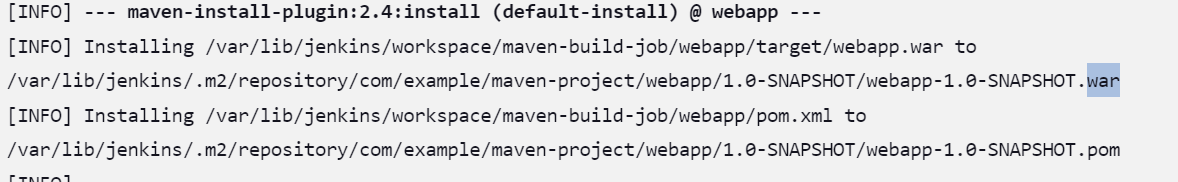
How to fix: simply clean the“source directory”

clean package





1. it is very hard to maintain the artifact version in the workspace directory, because on every new build old artifact will be replaced.
2. **install**: by default jenkins generates artifatc in workspace area/directory, and it is hard maintain versioning in the workspace directory, maven allows us to copy artifact in the .m2 maven directory (local maven repository). Install goal copy artifact from workspace to .m2 folder.



**Problem with install** goal: unnecessary it increases the size of jenkins server.

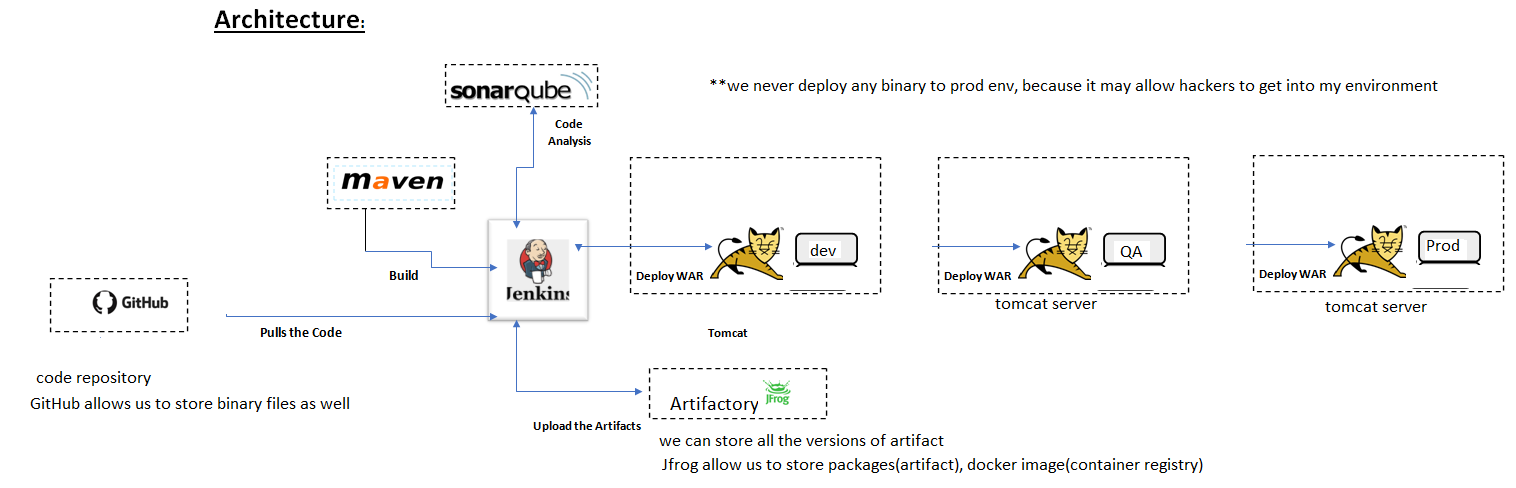
If something goes with the jenkins server then we won't be able to deploy any OLD artifacts.

How to fix: instead of storing artifact on jenkins server we can use Jfrog or Nexus (artifactory manager server)

Q. Can we store artifacts on the GitHub repository???

Ans: No, we can not store artifact on GitHub because of code binary files.

Github allows us to store binary files, and we do not prefer to deploy any binary files to prod env.



1. deploy: it allows us to publish(upload) packages to remote repository(jforg)

\*\* then how to upload packages to jfrog???

Instead of ‘deploy” goal we can consider jfrog plugin

Simply use “package” goal to generate artifact then upload to jfrog using plugin

—-27-11-2022

1. Install build tool/package along with dependencies on build server
2. Install required plugin on jenkins dashboard
3. Global tool configuration and provide executable path

**CI/CD (Continuous integration & Continuous delivery):**

<https://www.redhat.com/en/topics/devops/what-is-ci-cd>

**Continuous integration:**

CD (cont delivery) vs CD (cont deployment)

If prod deployment is a manual process (you can have some approval stages, initial stage) we can call this a continuous delivery approach.

If prod deployment is an automated process (code is stable) we can call this a continuous deployment approach.

***Continuous Integration (CI)****:*

The technical goal of CI is to establish a consistent and **automated way to build, package, and test applications**. With consistency in the integration process in place, teams are more likely to commit code changes more frequently, which leads to better collaboration and software quality. Errors can be detected in very early stage

***Continuous delivery(CD)****:*

*Continuous delivery* picks up where continuous integration ends. **CD automates the delivery of applications/artifacts to selected infrastructure environments (Dev/QA/Prod)**. Most teams work with multiple environments other than the production, such as development and testing environments, and CD ensures there is an automated way to push code changes/artifacts to them. Manual **deployment on prod env**. (early stage)

***Continuous deployment(CD)****: Similar to Continuous delivery , but* Automated **deployment on prod env**. (once your product is mature/pipeline is streamline)

WhastApp Link: <https://chat.whatsapp.com/Dk0cGpE4iFa1IDXqQuoa21>

===================================

3/12/2022

Java web application: tomcat/jboss/weblogic

Python web application: apache/nginx

Nodejs web application: apache/nginx

dotnet/dotnet core: windows iis

CI-CD with tomcat :-

1: create ec2 instance , open port 8080 (tomcat works on this port)

Q. what is the default port of tomcat..do you run tomcat on port 8080. How to change the default port?

(default port \*2)

8080\*2=

Or

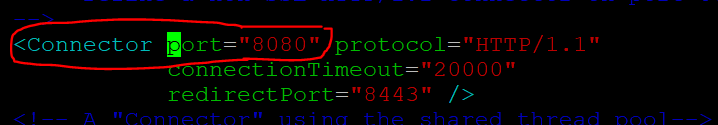
Default port +1

8080+1=8081

cd /usr/share/tomcat/conf

vi server.xml and update port value from 8080 to 8081

Also update your SG -> inbound-> allow->8081



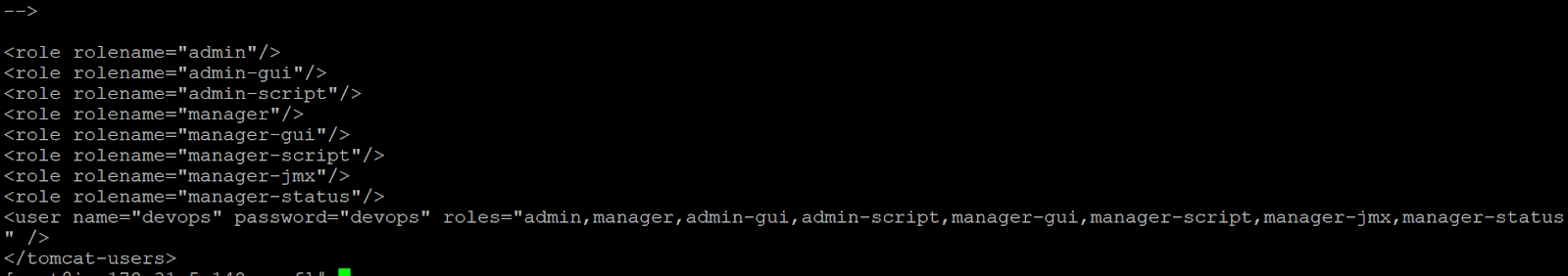
Restart tomcat services

Sudo systemctl restart tomcat

Access tomcat URL: http://<tomcat IP>:8081

To update user id & password:

vi tomcat-users.xml



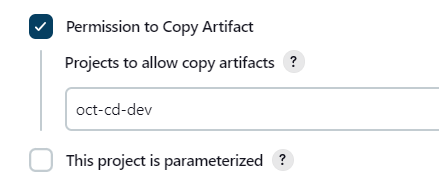
Restart service then access:

Install plugin

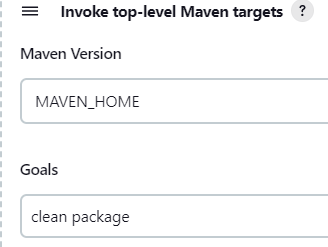
Manage jenkins -> manage plugin ->

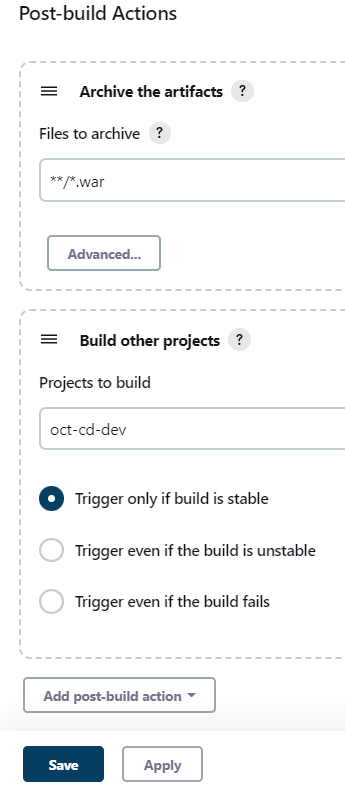
1. Deploy to container
2. Copy Artifact

Create CI job:



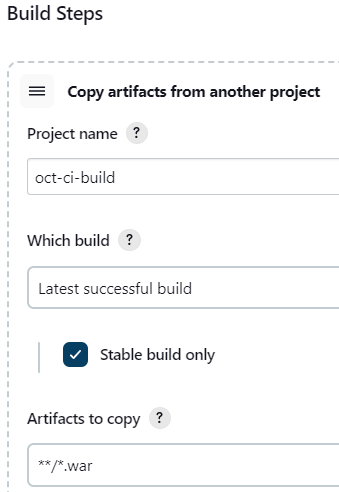


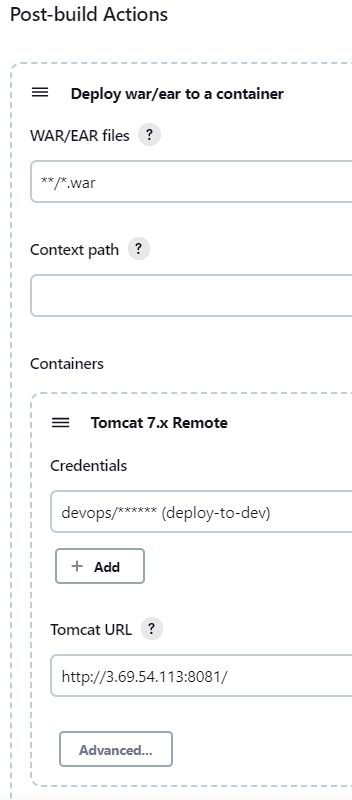




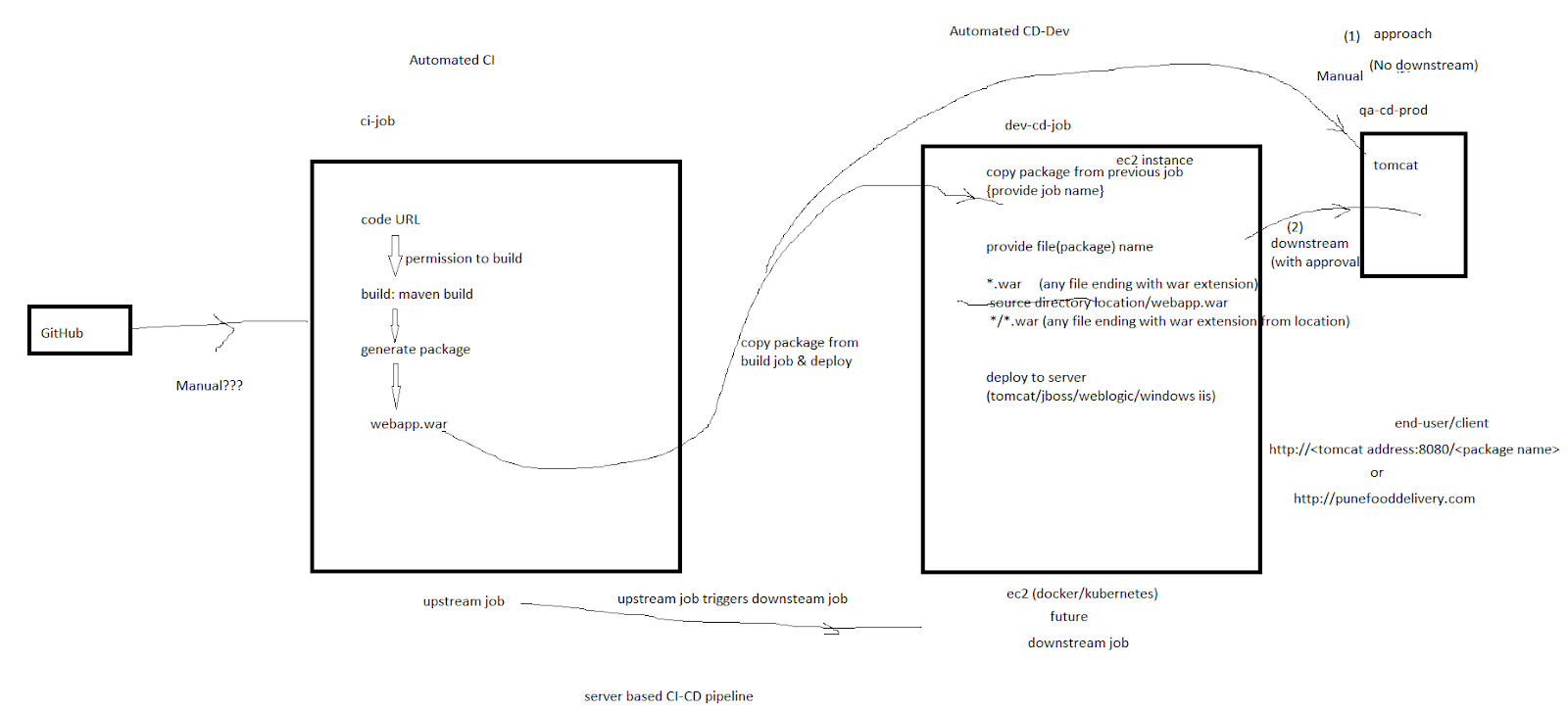
(:build other project:- :setting up downstream job)

**Create CD job:-**





Q. What do you understand about **upstream** and **downstream** jobs????

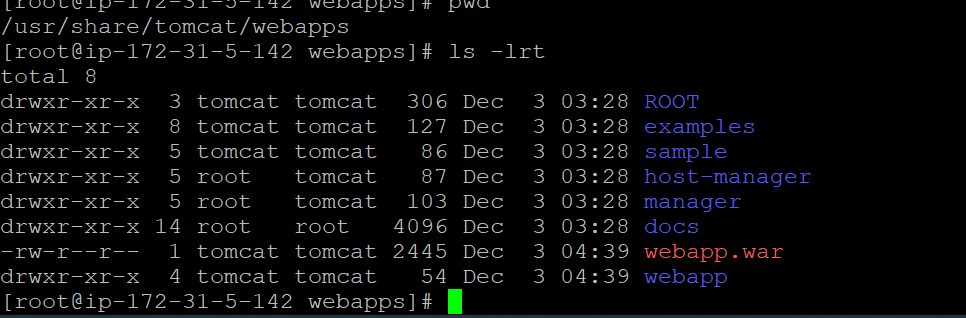


How to verify deployment status:

Access tomcat’s terminal

**cd /usr/share/tomcat/webapps {this is path where we can deploy artifact}**

ls -lrt



Tomcat is very advanced tool, automatically it extract all files (war/jar/ear)

http://<tomcat URL>:8081/webapp

\*\*\* you will be able to see the content of this file: <https://github.com/prakashk0301/maven-project/blob/master/webapp/src/main/webapp/index.jsp>

**Problem statement**: whenever there is any change in the code, sometime need to trigger /run/build the jenkins job

Q. How do you trigger jenkins jobs?

Q. Difference between poll scm and cron??

Q. Difference between poll and webhook(hook) ?

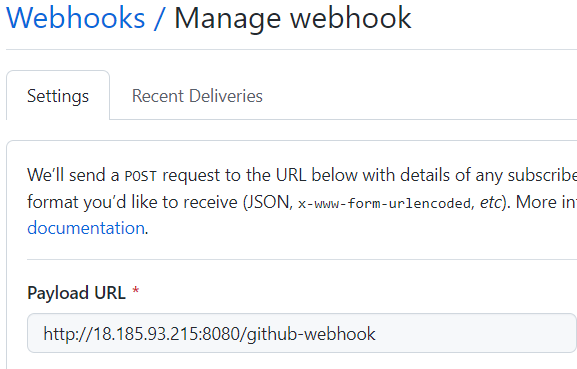
**Webhook** :- webhook is a service, webhook notifies jenkins on every code change/event

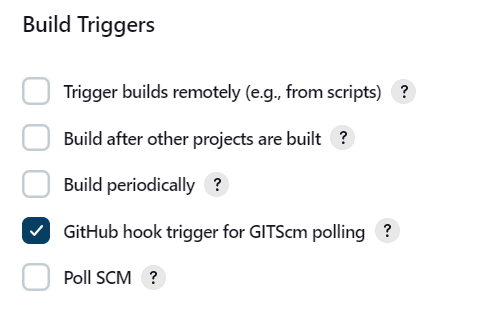
Push event, pull event, branch or tag creation/deletion

Open your github repo: <https://github.com/prakashk0301/maven-project>

Setting > webhook -> provide your current jenkins URL

http://18.185.93.215:8080/github-webhook





Assignment:

1. CI-CD with Jboss????

Hint: create new ec2 instance, install jboss (wildfly) , create new jenkins CD job, add jboss credentials, add jboss and deploy

1. CI-CD dev, QA (add new jenkins job and deploy to QA with approval)

Or

1. CI-CD dev, QA (add new jenkins job and deploy to QA by setting up downstream job)

—-----------4-12-2022

Q. Can we schedule jenkins job?

{create a jenkins job to clean tmp directory on every sunday at 2:00 AM}

{Can we run Jenkins job after every 30 min}

Cron syntax:-

cron syntax:

\* \* \* \* \*

(min 0-59) hour(0-23) (1-31 day of month) (month 1-12/jan-dec) (day of week 0-7/1-7/mon-sun where 0=7=sunday)

\*=every

1. Schedule a jenkins job at every sunday 2:00 AM

0 2 \* \* sun OR 0 2 \* \* 0 OR 0 2 \* \* 7

2. Schedule jenkins job after every 20 min???

\*/20 \* \* \* \* ----> correct ->8:50 (create jenkins job) 9:10 9:30 9:50

20 \* \* \* \* ----> incorrect -> 8:00 8:20, 8:40 9:00 9:20 9:40 10:00

\*/3 \* \* \* \* ----> incorrect -> every 3 min -> 8:50 8:53 8:57 9:00 9:03

3: trigger jenkins job first sunday of every quarter at 12:00 AM??

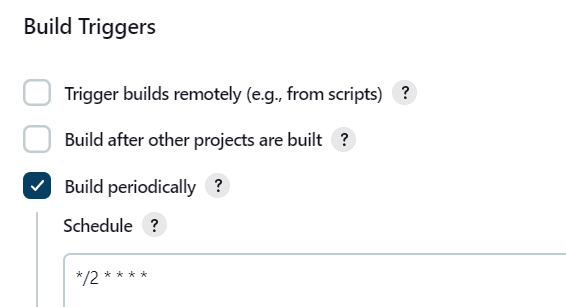
0 0 \* \*/4 0

0 0 \* \*/3 0 ---> correct \*/3

OR

0 0 1-7 \*/3 0 ---> correct , we also can define a range

\* \* \* \* \* --> every min



\*\*\*we can use Build Periodically option to trigger repetitive job (clean up activity, patch installation, monitoring, file transfer, backup creation)

<https://crontab.guru/>

—------------- You can schedule job for cleanup activity

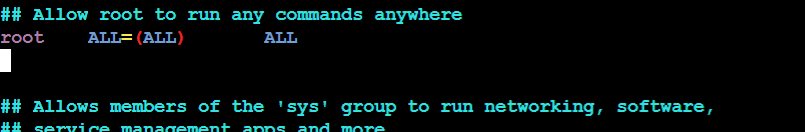
Once you are connected to server:

sudo su - {login as super user}

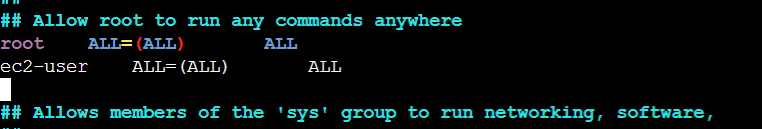
cd /etc/

vi sudoers

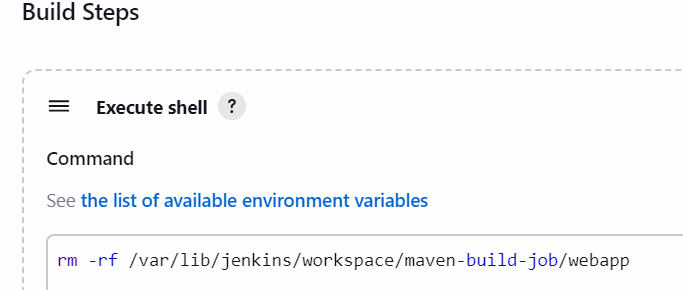
Before change



Add your automation user



You can create a job to clean some folder



\*\*\*provide absolute location

—----------------------**Master Slave**

Problem statement:

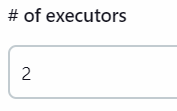
1. What if the build agent is not available

(how many job build agent can run concurrently/parallely)

(build agent is down)

1. What if you have many jenkins job to run (complex environment/many jenkins job)

Manage jenkins -> configure system



Number of Executor: build agent can run this job parallelly

We can have 1 master and many slave (build agent) instances

How to decide number of slave agents:

1. Build time (if build time is more than 1 hour)
2. How many job you have/how many dev/Qa

\*\*\*Install only build tools & dependencies on build agent (slave VM)

\*\*\* do not install jenkins package on slave VM

Install java git maven on slave vm

Step 1. Launch an instance (Amazon Linux) , Access Instance terminal and install and setup java environment

sudo yum install -y git aws-cli

sudo amazon-linux-extras install java-openjdk11

sudo alternatives --config java

(In this command point to the latest/or required java version by selection)

Step 2. Install Apache Maven: Add the apache maven repository to the yum repos, and install apache from there

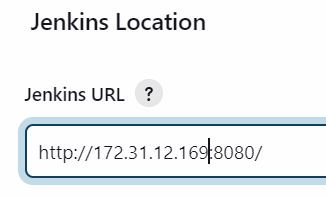
sudo wget http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo

sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo

sudo yum install -y apache-maven

mvn --version

Open Jenkins dashboard -> Manage Jenkins-> configure system -> update current Public IP of jenkins (or private ip of jenkins)



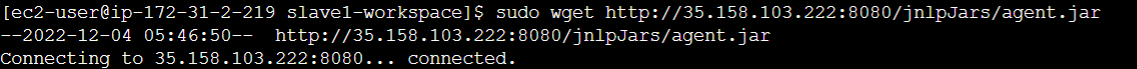
Open Jenkins dashboard -> Manage Jenkins -> Manage node and Cloud

Once you add slave. Right on agent.jar

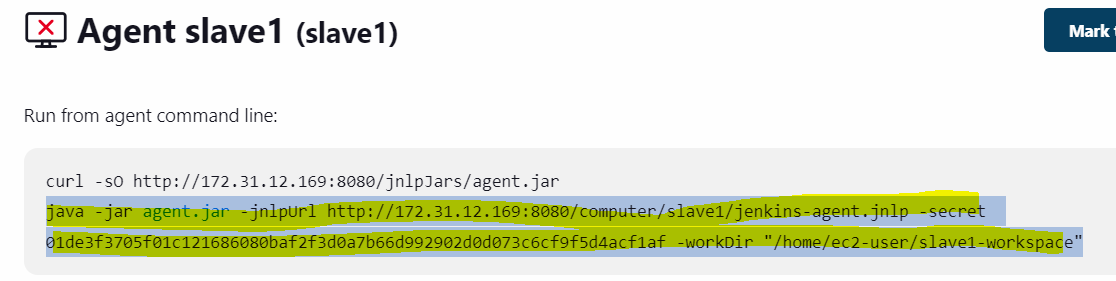


Go to the slave agent and run

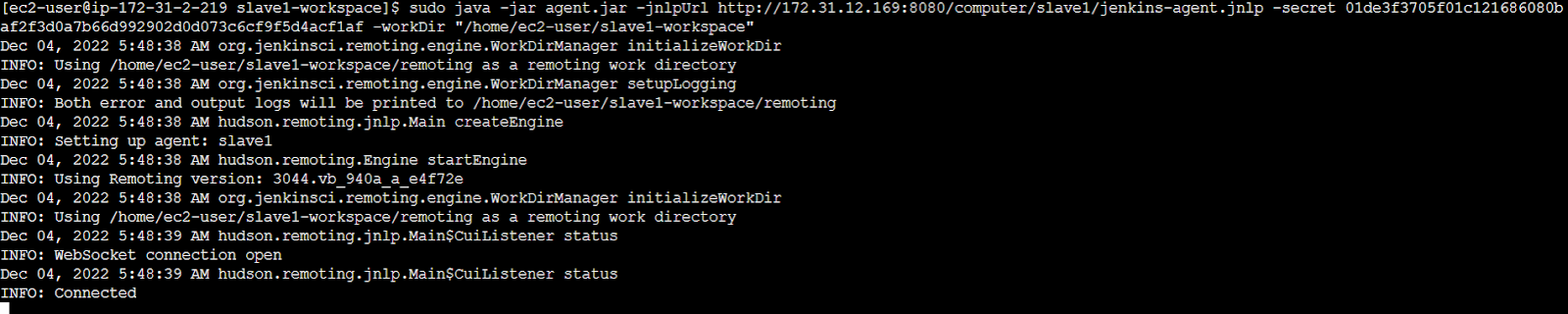
sudo wget http://<your current jenkins public ip>:8080/jnlpJars/agent.jar



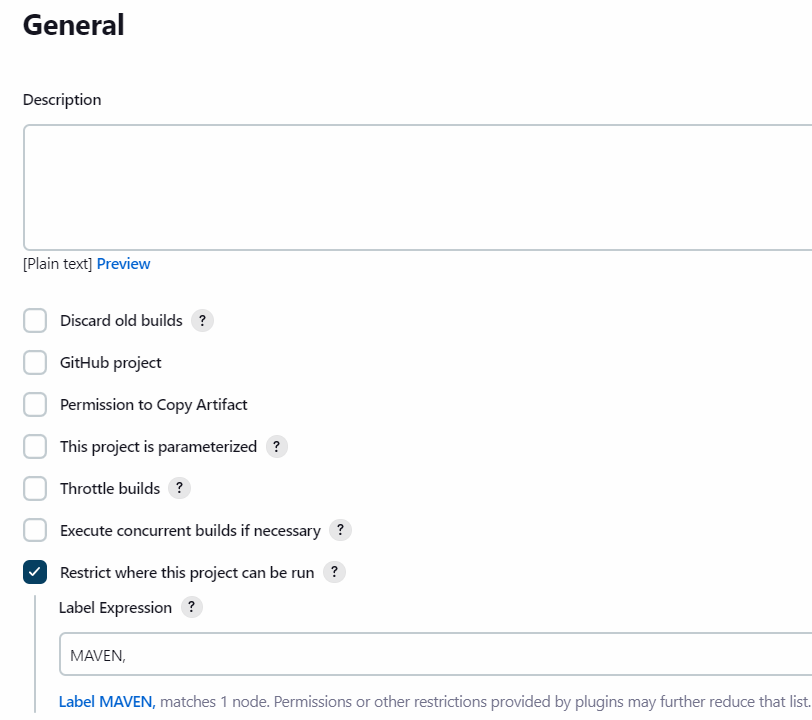
Once you download agent.jar on slave -> create connectivity between jenkins & slave



Copy highligted command and run on slave vm

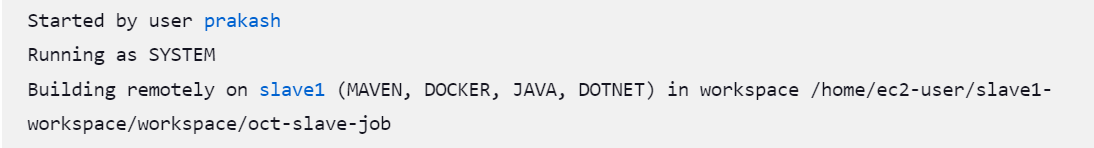


Create Jenkins job:



Provide correct LABEL value as per master slave architecture.

Check the output.



\*\*\*\*\*how to open this connection for all time

cd /ec2-user/home/slave1-workspace

sudo vi autostart.sh

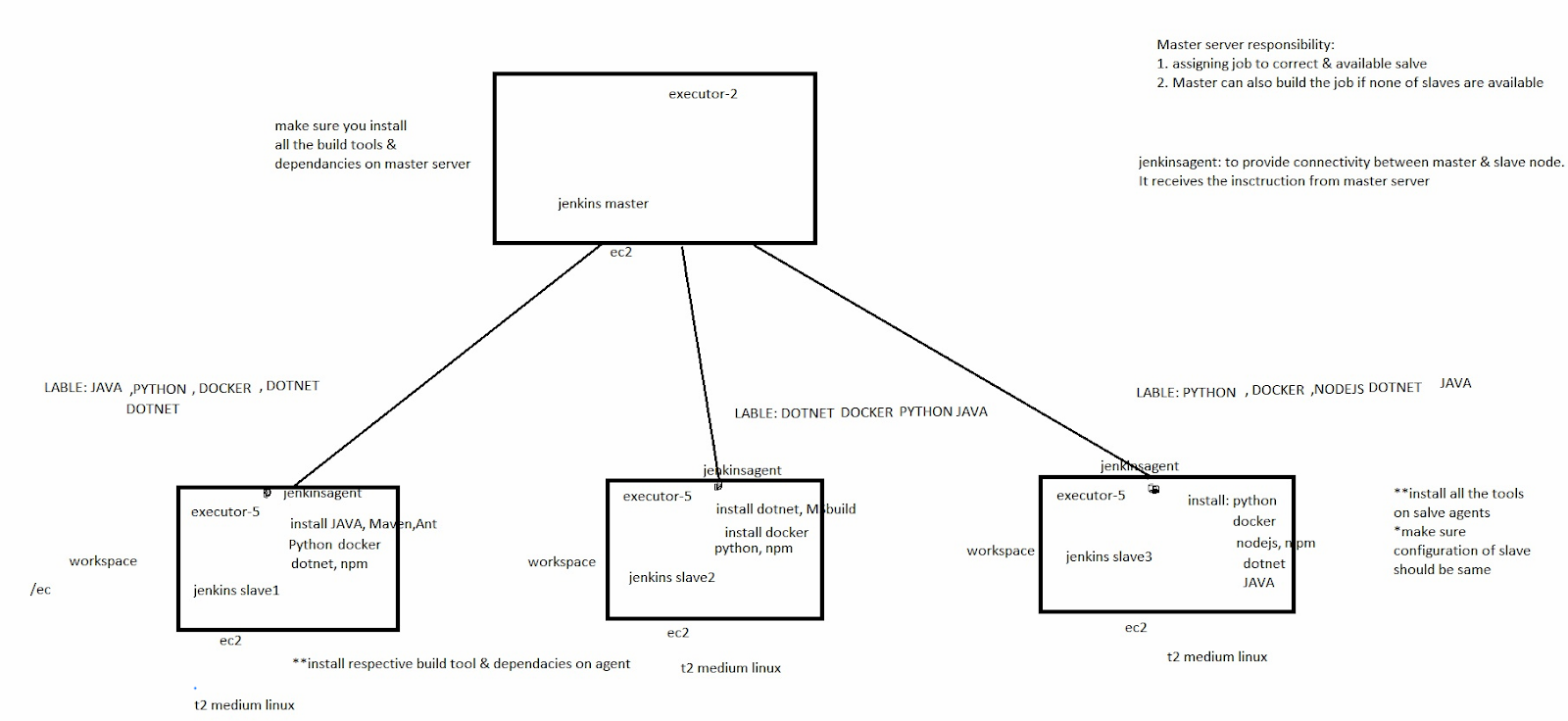
sudo java -jar agent.jar -jnlpUrl http://172.31.12.169:8080/computer/slave1/jenkins-agent.jnlp -secret 01de3f3705f01c121686080baf2f3d0a7b66d992902d0d073c6cf9f5d4acf1af -workDir "/home/ec2-user/slave1-workspace" **&**

**{add & at last, & (ampersand) runs the process in background for all time}**

**sudo chmod 744 autostart.sh**

**./autostart.sh**

**Now if you press ctrl c still you will be connected to the master**

****

**===========10-21-2022**

**sonarQube: static code analysis- >** we can integrate sonarqube during jenkins build process. The purpose of Sonarqube is to read the output of build job and generate report based on output, in that report we can generate below findings

**Vulnerability: weakness in system (**backend path/port/module/etc**)**

**Bugs:**

**CodeQuality: A, B, C, D, E**

**CodeCoverage:** passed test cases/failed , jacoco (java code coverage) plugin generated the code coverage report

For dev deployment: at least 70%

For QA deployment: at least 80%

For Prod deployment: at least 90%

Flow

* Clone the code from source code repo (github)
* Validate the code, compile the code
* **Prepare sonarqube analysis** (we are asking sonar to check the below steps: sonar scanner plugin)
* Execute unit test cases (junit/nunit/selenium)
* Build the code (maven)
* **Prepare sonarqube report based on the previous logs/output and publish to sonarqube dashboard**
* SonanQube also depends on some plugins

(findbug plugins -> find knows bug in the code

Jacoco -> generate code coverage report

Checkstyle and Violation plugin->related to code quality, code smell, syntax error )

Install SonarQube:-

SonarQube can’t be installed on t2 micro

( SonarQube server requires at least 2GB of RAM to run efficiently and 1GB of free RAM for the OS.)

1. Create t2 medium aws linux instance (make sure you terminate the instance after yout lab)
2. Open port 9000 in your SG

(<https://devopscube.com/setup-and-configure-sonarqube-on-linux/> : traditional method to install sonarqube)

1. SonarQube can be installed on Docker

sudo yum update -y

sudo yum install docker -y

sudo service docker start

sudo systemctl enable docker

sudo service docker status

{docker is platform where you can deploy Application on container

Container is similar to an instance

You can create many containers on a single docker instance}

Create sonarqube container:-

**sudo docker run -d --name sonar -p 9000:9000 sonarqube**

**Where:**

run : create & start docker container

-d : detached mode (run the container all time in background )

-p : port (to expose container port)-> port forwarding

9000:9000 -> container port : VM port {application default port: VM port}

\*\*\*VM port should be unique

sonarqube: docker image name

How to get the correct docker image name: <https://hub.docker.com/search?q=sonar>

\*\*\*<https://hub.docker.com/>

To check running container

**sudo docker ps** —-> list all the running containers

You can access from dashboard:

**http://<public ip of sonar instance>:9000**

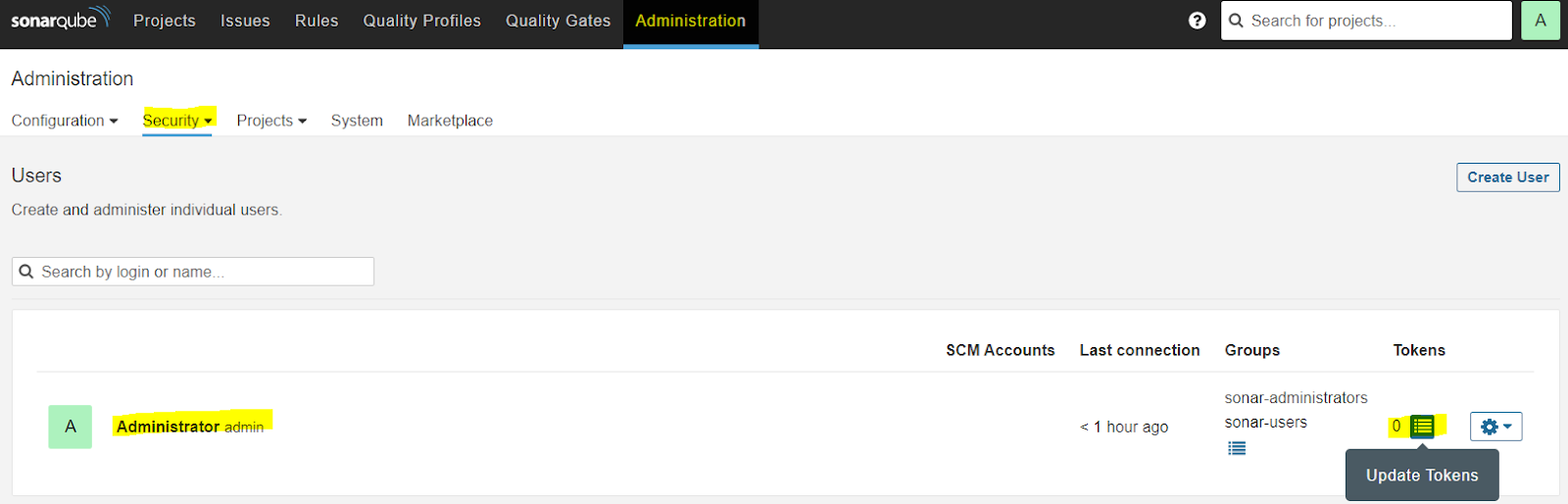
Default user id & password

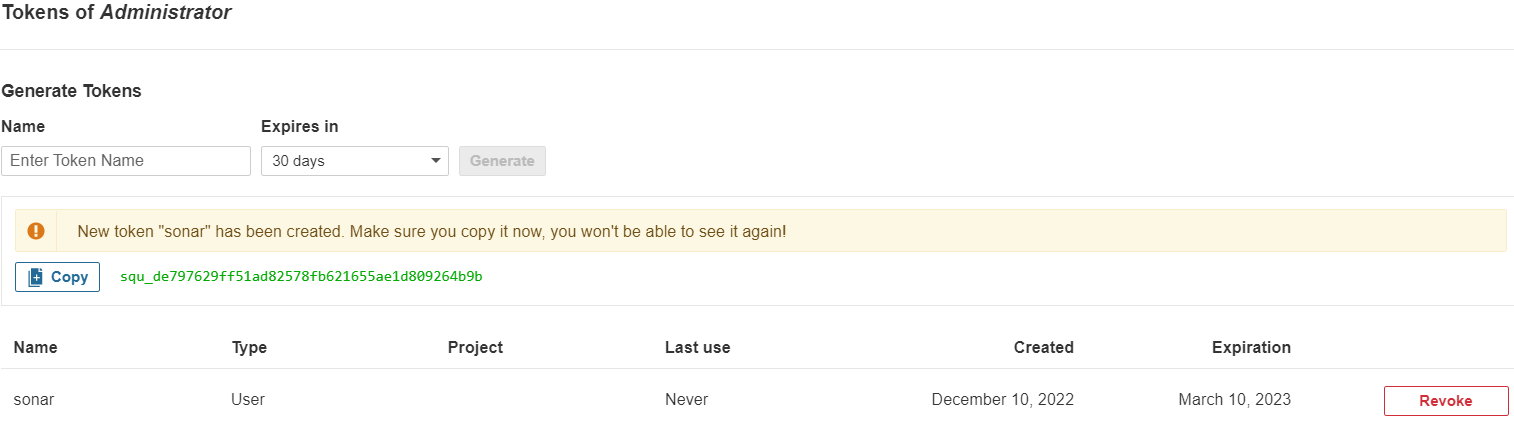
admin

admin

1. Create sonarqube token (for jenkins authentication)

Sonarqube dashboard-> Administration->security-> users->generate token

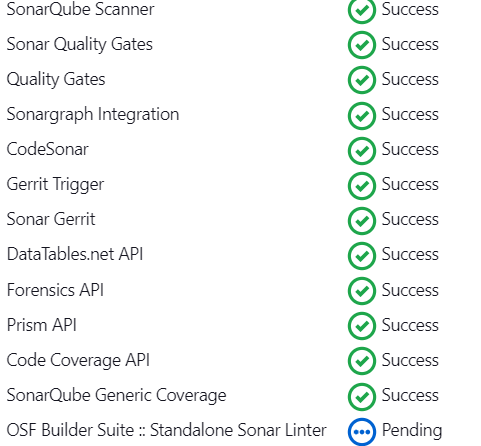




Mytoken: squ\_de797629ff51ad82578fb621655ae1d809264b9b

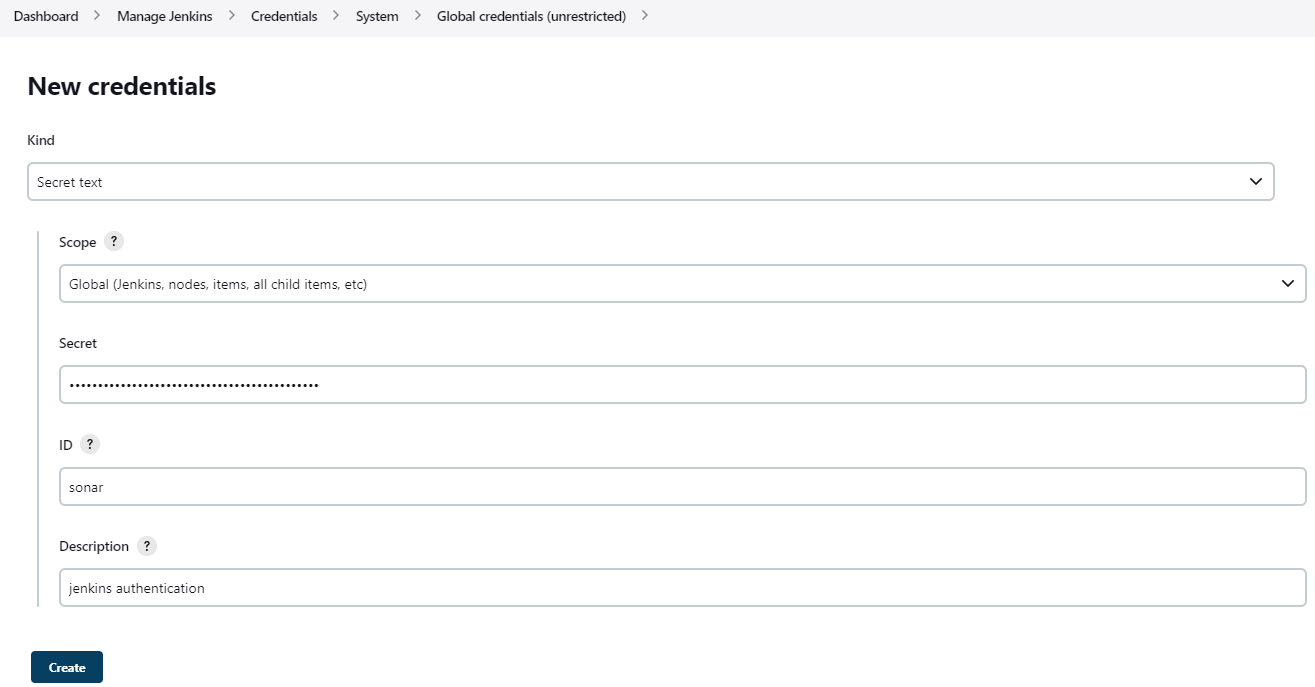
1. Install plugins on jenkins

Search by this name “sonar”



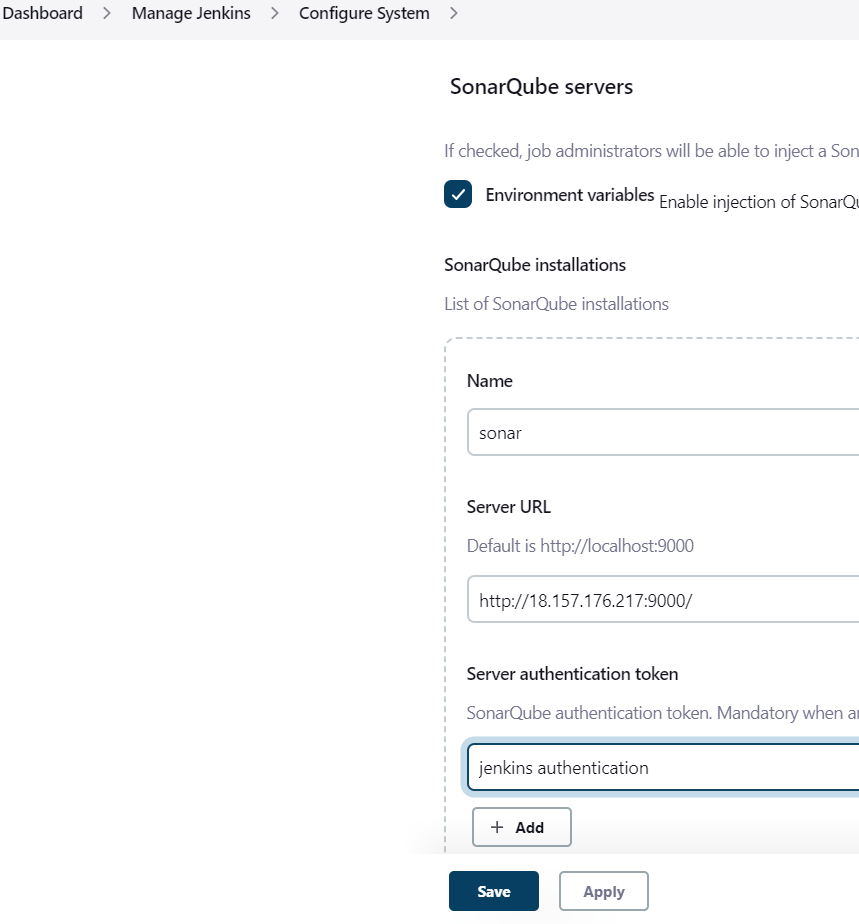
1. Add sonar token in jenkins (so that jenkins can publish report to sonar dashboard)

Manage jenkins-> manage credentials -> system-> global-> add credentials-> secret text ->

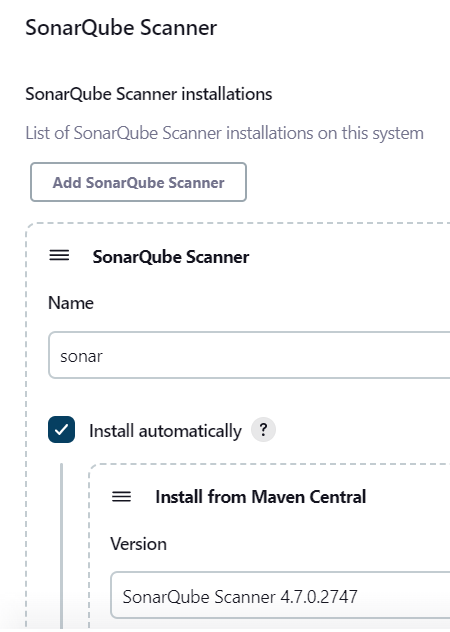


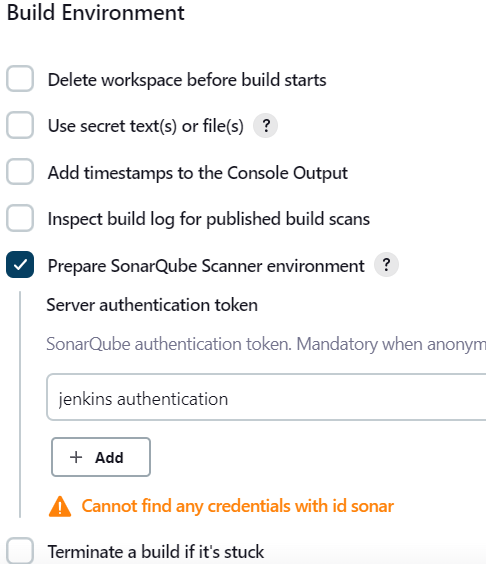
1. Add sonar URL (jenkins can publish the report)

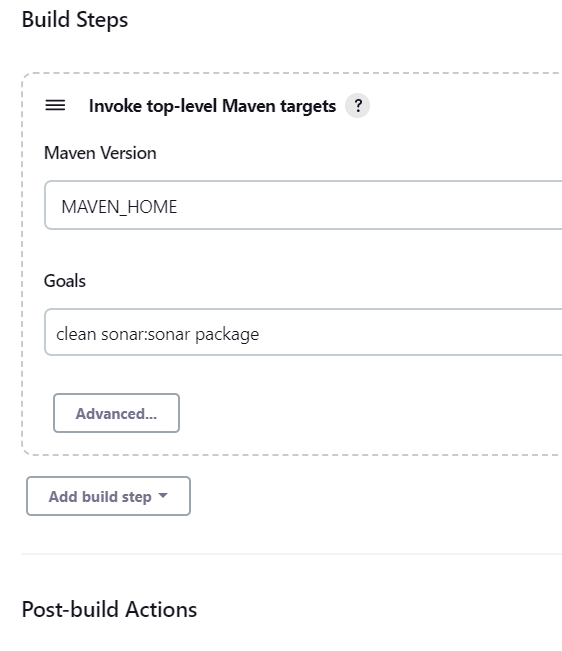
Manage jenkins-> configure system -> sonarqube server-> add sonar url and authentication token



1. Global tool configure: manage jenkins-> global tool configuration -> sonarqube scanner -> install automatically



JEnkins job creation:



Goal: clean sonar:sonar package

Where: first sonar is plugin, and 2nd sonar is goal.

Assignment:- Add sonar stage in CI-CD , if sonar coverage report is more than 70 % then deploy automatically to dev env

**\*\*\*\* start sonarqube container (after VM reboot)**

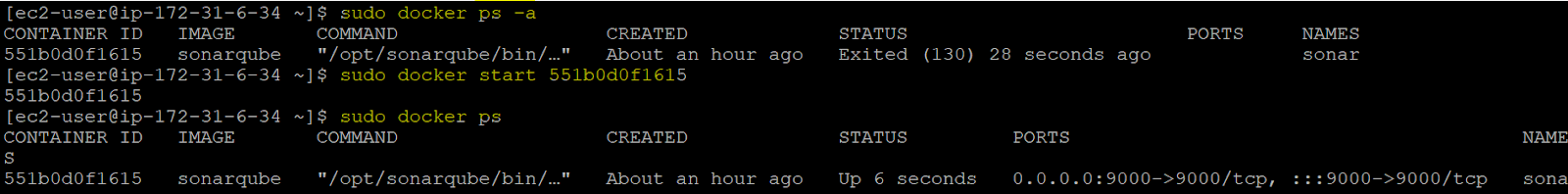
**sudo docker ps -a —-> it display running+stopped both the container**

**sudo docker start <container ID>**

**Or**

**docker start sonar**

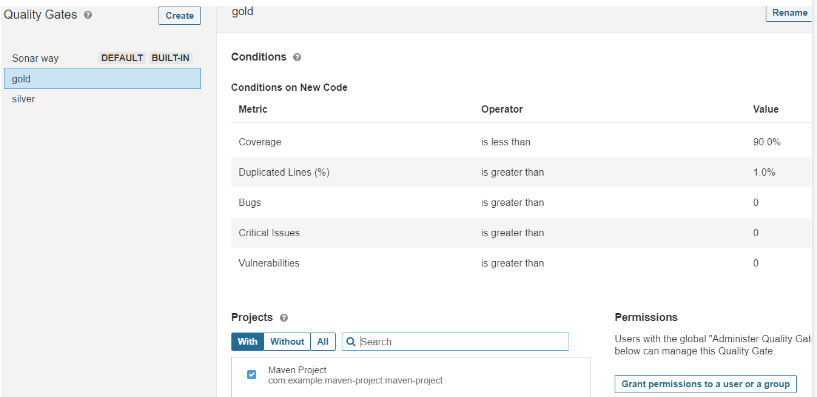
**sudo docker ps —-> you should be able to see the container**

****

Quality Gate: is nothing but a filter, where we can have some criteria (condition)

In my project we have created 3 types of quality gate

1. Gold Quality gate: Production environment

* Code coverage value : 90%

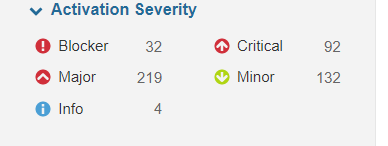
1. Silver Quality gate: QA Env

* Code coverage value : 80%

1. Bronze Quality gate: Dev env

* Code coverage value : 70%

Quality Profile: we can add/remove rules, we also can add custom rule

* [Regular expressions should not contain multiple spaces](http://3.74.229.213:9000/coding_rules?open=java%3AS6326&rule_key=java%3AS6326)
* Zero whitespace: [Whitespace for text block indent should be consistent](http://3.74.229.213:9000/coding_rules?open=java%3AS5664&rule_key=java%3AS5664)
* ["clone" should not be overridden](http://3.74.229.213:9000/coding_rules?open=java%3AS2975&rule_key=java%3AS2975)
* [Regular expressions should not contain empty groups](http://3.74.229.213:9000/coding_rules?open=java%3AS6331&rule_key=java%3AS6331)
* 

===============Pipeline

There are 2 types of pipeline

1. **Scripted pipeline** (old one-> complex to write,each activity needs to define in script)

It always starts with “node” keyword

You need to put your script the jenkins job only

1. **Declarative pipeline** (new one-> very easy, and supports env variables)

Now you can put your script to the code repository

It starts with “pipeline” keyword

Vi jenkinsfile -> upload to github, along with code

Declarative Pipeline type of jenkins job:-

vi Jenkinsfile //we can upload this file to github repo along with the code

**pipeline {** //declarative pipeline always starts with "pipeline" keyword

**agent any** // all below stages can run on any available executor

**stages {** //it contains all the stages

**stage('stage-1-print-hi')** //define stage number 1, you can give any name

**{**

**steps { sh 'echo hi' }** //steps: it tells jenkins how to perform the stage, it contains actual command, where "sh-> represent execute command in shell "

**}**

**stage('stage-2-test')**

**{**

**steps { sh 'echo executing-test-cases' }**

**}**

**stage('stage-3-build')**

**{**

**steps { sh 'echo code-is-building' }**

**}**

**}**

==========18-12-2022

Create a declarative pipeline to print hello

vi Jenkinsfile

**pipeline {**

**agent any**

**stages {**

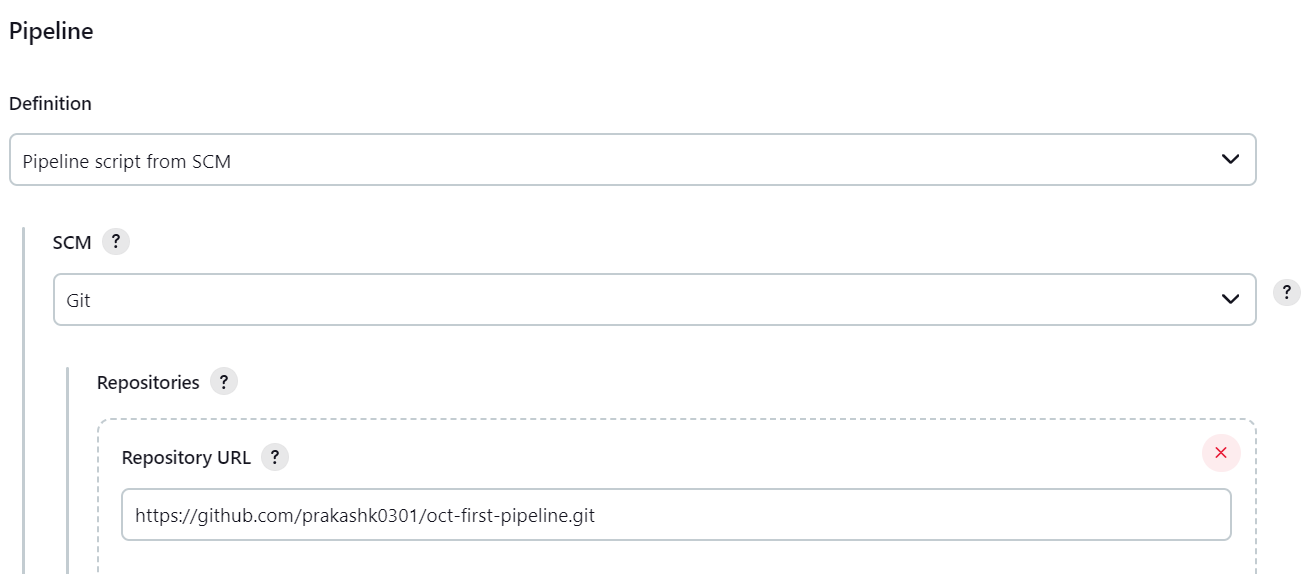
**stage ('print a message')**

**{ steps { sh 'echo this\_is\_pipeline' } }**

**}**

**}**

--> once you are done with the coding->create a github repo-> create a file called "Jenkinsfile" -> done -> Create a Jenkins job -> select "pipeline"



---------------------------------Create a declarative pipeline to print hello

vi Jenkinsfile

pipeline {

agent any

stages {

stage ('print a message')

{steps { sh 'echo this\_is\_pipeline' }}

}

}

--> once you are done with the coding->create a github repo-> create a file called "Jenkinsfile" -> done -> Create a Jenkins job -> select "pipeline"

======================Create a Jenkinsfile and print "job-is-building" , "package-is-deploying" , "deploy-to-prod"

vi Jenkinsfile

pipeline { //always start with pipeline

agent any //this job will run on any available executor

stages { //contains all the stages

stage ('stage-print-first-message')

{ steps { sh 'echo job-is-building' }} //what to do, you can provide your command

stage ('stage-print-second-message')

{steps {sh 'echo packing-is-deploying'}}

stage('stage-final-stage')

{steps{ sh 'echo deploy-to-prod' }}

}

}

OR vi Jenkins2

pipeline {

agent any

stages {

stage('print messages')

{ steps { (sh 'echo job-is-building ')

(sh 'echo packing-is-deploying ')

(sh 'echo deploy-to-prod) } }

}}

=============Syntax:

pipeline {

agent any

stages {

}}

-----ref link https://www.jenkins.io/doc/book/pipeline/

24-12-2022

**Pipeline node and executable shells:** <https://www.jenkins.io/doc/pipeline/steps/workflow-durable-task-step/>

Jenkins install

Search by “pipeline” keyword and install related pipeline plugins

| Common API for Blue Ocean | Success |
| --- | --- |
| REST API for Blue Ocean | Success |
| Pub-Sub "light" Bus | Success |
| Pipeline SCM API for Blue Ocean | Success |
| HTML Publisher | Success |
| JWT for Blue Ocean | Success |
| Design Language | Success |
| Blue Ocean Core JS | Success |
| Web for Blue Ocean | Success |
| Favorite | Success |
| REST Implementation for Blue Ocean | Success |
| Pipeline implementation for Blue Ocean | Pending |
| Pipeline: Declarative Agent API | Pending |
| Git Pipeline for Blue Ocean | Pending |
| GitHub Pipeline for Blue Ocean | Pending |
| Pipeline Utility Steps | Pending |
| Parameterized Trigger | Pending |
| jQuery | Pending |
| Build Pipeline | Pending |
| H2 API | Pending |
| Config File Provider | Pending |
| JUnit | Pending |
| **Pipeline Maven Integration** | Pending |
| **SSH Pipeline Steps** | Pending |
| **Delivery Pipeline** | Pending |
| Loading plugin extensions |  |

Q. How to restart jenkins from the browser??

Ans: <jenkins url>**/restart**

**Jenkins will restart , it is not going to wait for job to finish**

Q. How to safely restart jenkins from the browser??

Ans: <jenkins url>**/safeRestart**

**Jenkins will restart once all running jobs are finished.**

Q. How to restart Jenkins forcefully?

Ans: <jenkins url>**/restart**

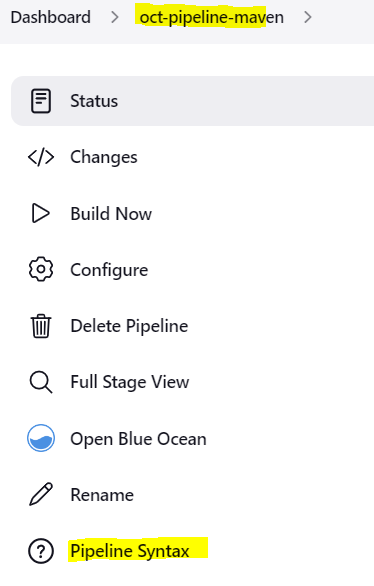
Q. Difference between safe-restart and force-restart?

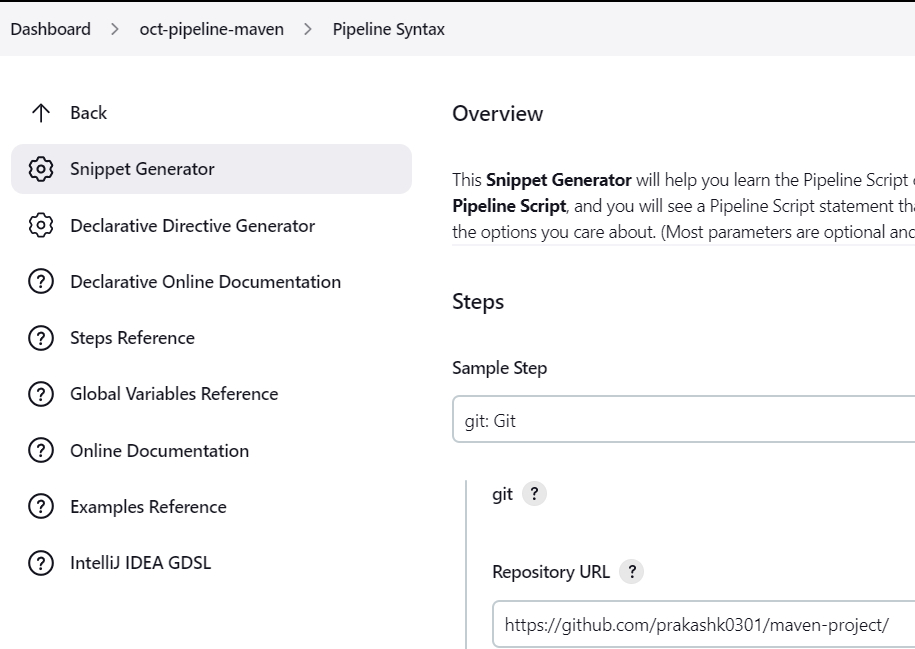
forceRestart: **Jenkins will restart , it is not going to wait for job to finish**

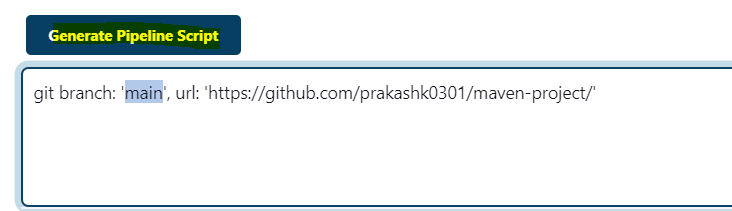
safeRestart: **Jenkins will restart once all running jobs are finished.**

**Ref jenkinsfile:** [**https://www.jenkins.io/doc/book/pipeline/jenkinsfile/**](https://www.jenkins.io/doc/book/pipeline/jenkinsfile/)

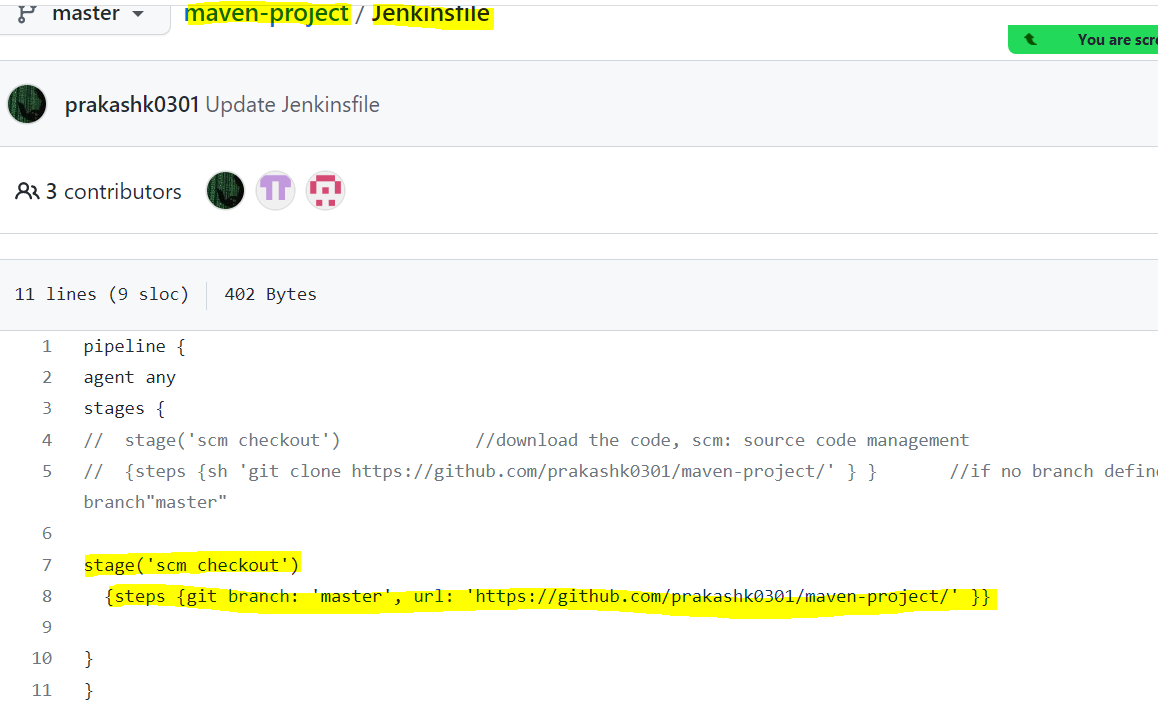
* Create jenkins pipeline job
* Select syntax generator and generate first step for git

****

****

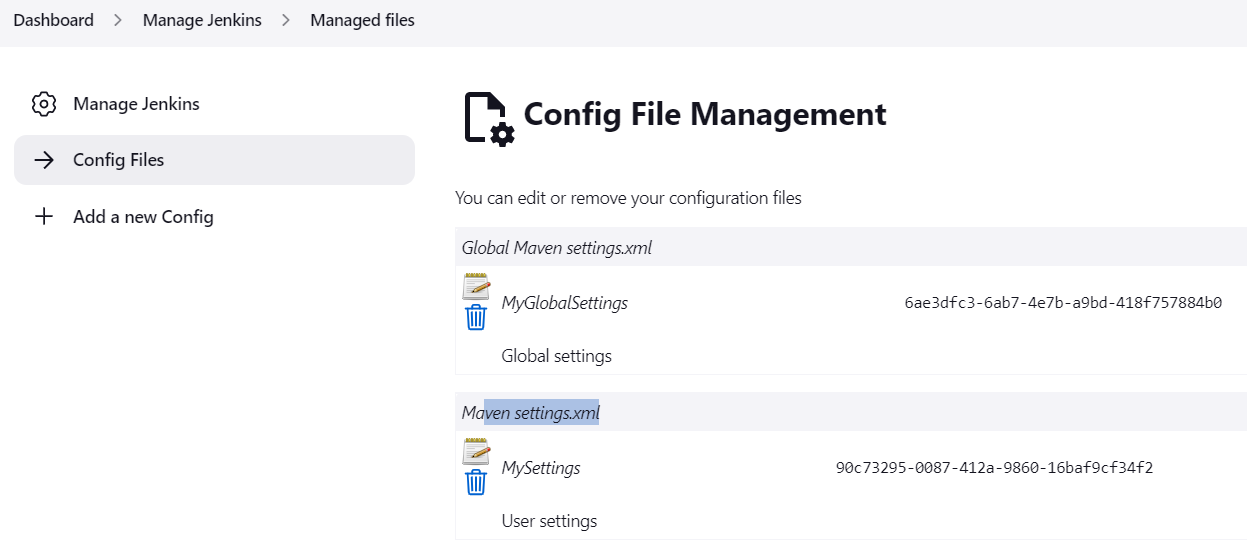
****

Create Jenkinsfile in the code repository:



—-----------add new stage

Before you add new stage just defined **global maven setting.xml** and **maven setting.xml**



| pipeline { |  |
| --- | --- |
|  | agent any |
|  | stages { |
|  | // stage('scm checkout') //download the code, scm: source code management |
|  | // {steps {sh 'git clone https://github.com/prakashk0301/maven-project/' } } //if no branch defined then it will download the code from default branch"master" |
|  |  |
|  | stage('scm checkout') |
|  | {steps {git branch: 'master', url: 'https://github.com/prakashk0301/maven-project/' }} |
|  |  |
|  | stage('execute unit test framework') |
|  | {steps {withMaven(globalMavenSettingsConfig: '6ae3dfc3-6ab7-4e7b-a9bd-418f757884b0', jdk: 'JAVA\_HOME', maven: 'MAVEN\_HOME', mavenSettingsConfig: '90c73295-0087-412a-9860-16baf9cf34f2') |
|  | { |
|  | sh 'mvn test' |
|  | }}} |
|  |  |
|  |  |
|  | stage('create deployable package') |
|  | {steps {withMaven(globalMavenSettingsConfig: '6ae3dfc3-6ab7-4e7b-a9bd-418f757884b0', jdk: 'JAVA\_HOME', maven: 'MAVEN\_HOME', mavenSettingsConfig: '90c73295-0087-412a-9860-16baf9cf34f2') |
|  | { |
|  | sh 'mvn clean package' |
|  | }}} |
|  |  |
|  | } |
|  | } |

Ref jenkinsfile : <https://github.com/prakashk0301/maven-project/blob/master/Jenkinsfile>

{genereate exe file <https://techcommunity.microsoft.com/t5/modern-work-app-consult-blog/part-3-create-a-jenkins-pipeline-to-deploy-msix-desktop-apps/ba-p/3160430>}

All jenkins plugin path: <https://updates.jenkins.io/download/plugins/>

Assignment:

1. Create freestyle job using gradle and ant

25-12-2022

Create a new ec2 instance : (Amazon Linux 2 AMI (HVM), SSD Volume Type)

yum install -y java-1.8.0

yum install -y tomcat

systemctl start tomcat

systemctl status tomcat

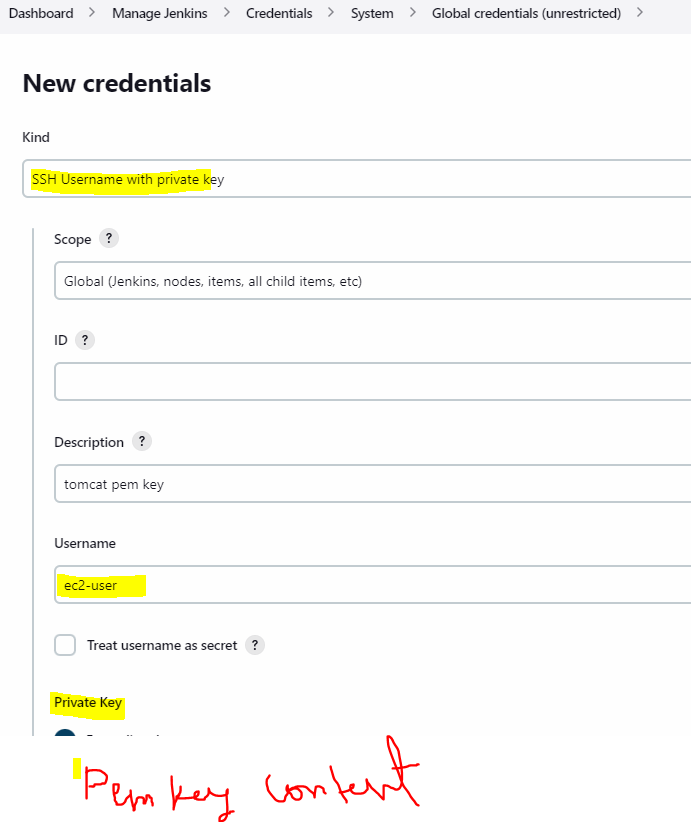
systemctl enable tomcat ----> to start the service for all time (Automatic)

sudo yum install -y tomcat-webapps tomcat-admin-webapps tomcat-docs-webapp tomcat-javadoc

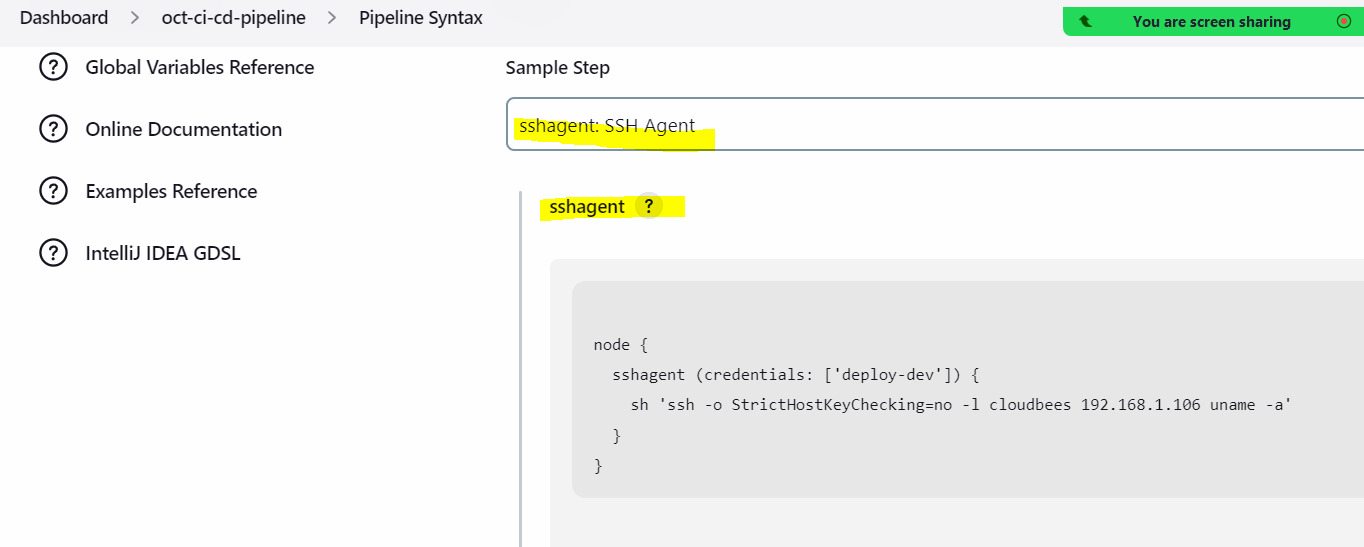
Install plugins:-

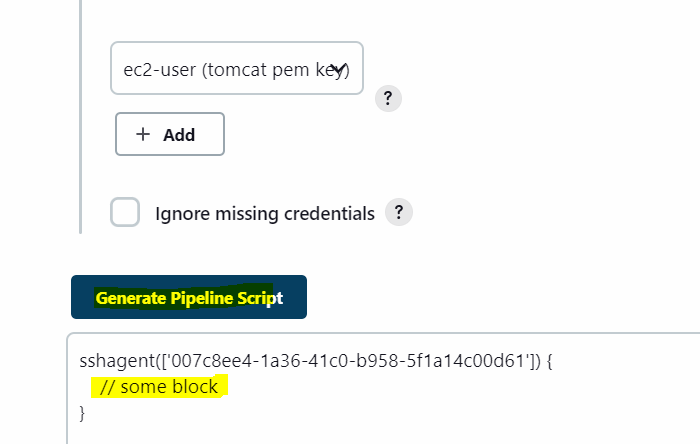
| SSH | Pending |
| --- | --- |
| SSH Agent | Pending |
| SCP publisher | Pending |

Add credentials -> pem key of tomcat



Add new stage:





----------------

origin syntax:

sh 'ssh -o StrictHostKeyChecking=no -l cloudbees 192.168.1.106 uname -a'

->change ssh to scp

-> -o : option -> it means we can add some option with scp

->StrictHostKeyChecking=no -> don't check any host key, if key is correct then simply perform the command "scp" else prompt error message

sh 'scp -o StrictHostKeyChecking=no -l cloudbees 192.168.1.106 uname -a'

\*/target/\*.war --> source file (\*/target/webapp.war or webapp/target/webapp.war ---> check in the jenkins directory)

ec2-user@<private or public ip of tomcat>:/var/lib/tomcat/webapps → /var/lib/tomcat/webapps this is a destination directory

sh 'scp -o StrictHostKeyChecking=no \*/target/\*.war ec2-user@18.156.175.215:/var/lib/tomcat/webapps'

---

original syntax:

sshagent(['007c8ee4-1a36-41c0-b958-5f1a14c00d61']) {

// some block

}

---> update commands

sshagent(['007c8ee4-1a36-41c0-b958-5f1a14c00d61'])

{ sh 'scp -o StrictHostKeyChecking=no \*/target/\*.war ec2-user@18.156.175.215:/var/lib/tomcat/webapps'}

--

Final stage:

**stage('deploy-to-dev')**

**{steps { sshagent(['007c8ee4-1a36-41c0-b958-5f1a14c00d61'])**

**{ sh 'scp -o StrictHostKeyChecking=no \*/target/\*.war ec2-user@18.156.175.215:/var/lib/tomcat/webapps'} }}**

run jenkins job if you get any permission error then go to tomcat

cd /var/lib/tomcat/webapps

sudo chmod 660 .

then re-run your job

Access your app from browser:

http://<tomcat public ip>:8080/webapp/

{ref Jenkinsfile: <https://github.com/prakashk0301/maven-project/blob/master/Jenkinsfile1>}

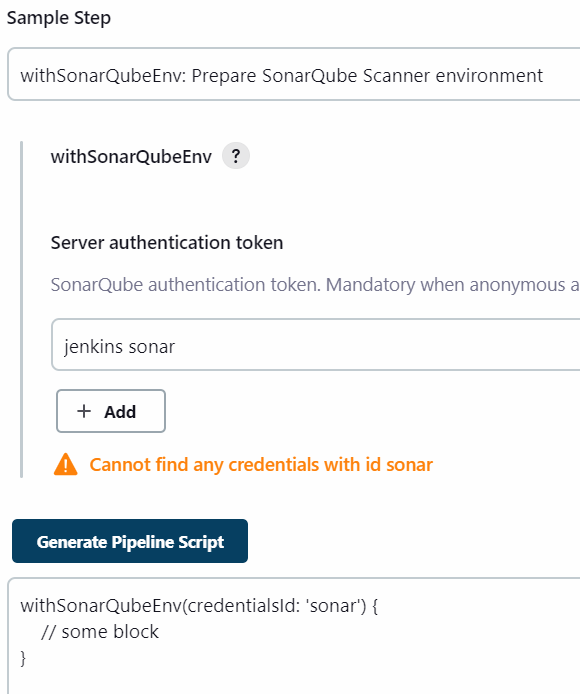
—-------------------------------------------------- Sonar integration

Create sonar server -> generate sonar token

Update sonar token -> manage jenkins-> manage credentials -> update sonar token

Update public ip of sonar : -> manage jenkins -> configure system -> update sonar public IP

Syntax generate -> search for “withSonar”



generate sonar script:

withSonarQubeEnv(credentialsId: 'sonar')

{

// some block

}

Generate script for maven:-

withMaven(globalMavenSettingsConfig: '6ae3dfc3-6ab7-4e7b-a9bd-418f757884b0', jdk: 'JAVA\_HOME', maven: 'MAVEN\_HOME', mavenSettingsConfig: '90c73295-0087-412a-9860-16baf9cf34f2') {

// some block

}

Create final script for sonar analysis and package building

withSonarQubeEnv(credentialsId: 'sonar')

{

withMaven(globalMavenSettingsConfig: '6ae3dfc3-6ab7-4e7b-a9bd-418f757884b0', jdk: 'JAVA\_HOME', maven: 'MAVEN\_HOME', mavenSettingsConfig: '90c73295-0087-412a-9860-16baf9cf34f2')

{ sh 'mvn clean package **sonar:sonar**' }

}

Create final stage:-

stage ('sonar analysis & code build')

{steps

{ withSonarQubeEnv(installationName: 'sonar', credentialsId: 'sonar') //installationName name is coming from configure-system

{

withMaven(globalMavenSettingsConfig: '6ae3dfc3-6ab7-4e7b-a9bd-418f757884b0', jdk: 'JAVA\_HOME', maven: 'MAVEN\_HOME', mavenSettingsConfig: '90c73295-0087-412a-9860-16baf9cf34f2')

{ sh 'mvn clean package sonar:sonar' }

} }}

Ref Jenkinsfile: <https://github.com/prakashk0301/maven-project/blob/master/Jenkinsfile1>

Jenkins Sonar integration: <https://igorski.co/sonarqube-scans-using-jenkins-declarative-pipelines/>

Assignment:

CI-CD pipeline using ssh (remote login)

Create CI for nodeJS project

Create CI job for python

—-------------------------

Q. How to execute stages parallelly?

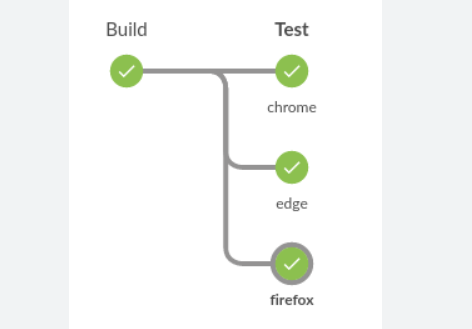
Ans: stage can be executed parallel if they are **independent**

Test-chrome

Test-edge

Test-firefox

Can be performed parallelly

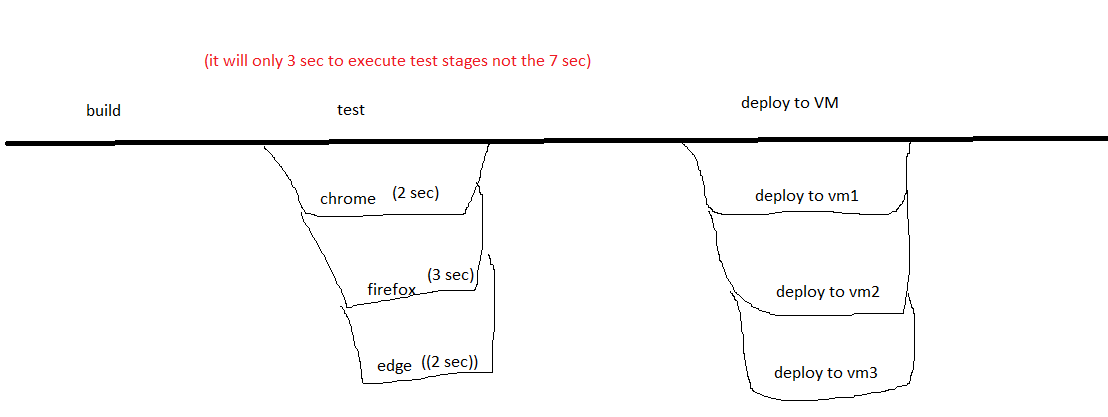


–

deploy to vm1

Deploy to vm2

Deploy to vm3



Build and test can Not execute parallely (because test is depend on build stage, thats we need to perform build and test stages sequentially )

We can execute any stages parallelly using “parallel” option in pipeline

Ref Jenkinsfile: <https://github.com/prakashk0301/maven-project/blob/master/Jenkinsfileparallel>

<https://github.com/prakashk0301/maven-project/blob/master/Jenkinsfileparallel1>

=====================How to execute multiple steps in same stage

Q. Diff between **stage parallel and steps parallel**

**=====================How to execute multiple steps in same stage:**

**when to consider multiple steps:-> when you want to perform diff diff activity/command .**

**stage('deploy-to-dev VM1')**

**{steps { sshagent(['007c8ee4-1a36-41c0-b958-5f1a14c00d61'])**

**{ sh 'scp -o StrictHostKeyChecking=no \*/target/abc.war ec2-user@VM1:/var/lib/tomcat/webapps' //command1**

**sh 'scp -o StrictHostKeyChecking=no \*/target/image.jpeg ec2-user@VM1:/var/lib/tomcat/webapps'} }} //command2 ,**

**example:**

**stage('execute unit test framework')**

**{steps {withMaven(globalMavenSettingsConfig: '6ae3dfc3-6ab7-4e7b-a9bd-418f757884b0', jdk: 'JAVA\_HOME', maven: 'MAVEN\_HOME', mavenSettingsConfig: '90c73295-0087-412a-9860-16baf9cf34f2')**

**{**

**sh 'mvn test'**

**sh 'mvn package'**

**}}}**

**example2:**

**stage('execute unit test framework')**

**{steps { sh 'echo googlechrome-test' //jenkins runs steps sequentially . 9:57 AM**

**sh 'echo firefox-test' }} // 9:59 AM ,, if command**

**example3:**

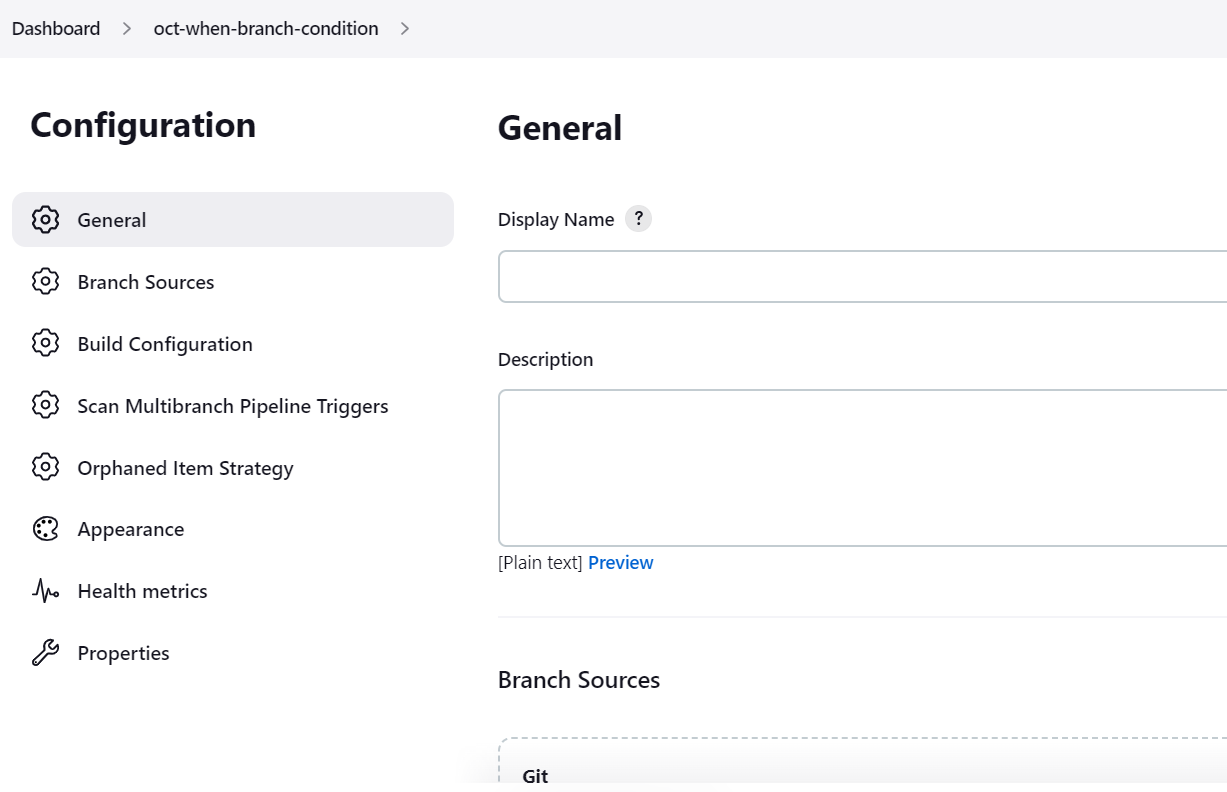
**stage('install tomcat')**

**{steps { sh 'yum install tomcat -y' //jenkins runs steps sequentially . 9:57 AM**

**sh 'service tomcat start' }} // 9:59 AM ,, if command**

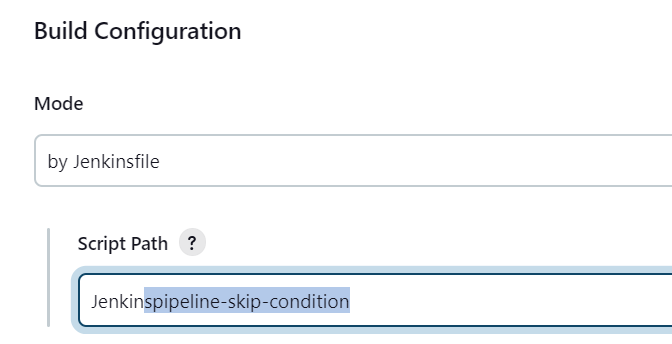
======================How to skip any stage

Create multi branch pipeline job not pipeline





Ref pipeline: <https://github.com/prakashk0301/maven-project/blob/master/Jenkinspipeline-skip-condition>



2. when branch is "prod" then deploy to prod env if not then skip the stage

pipeline {

agent any

stages {

stage('Example Build') {

steps {

echo 'Hello World'

}

}

stage('Example Deploy') {

**when** {

branch 'master' //jenkins will check branch if branch not found then skip the stage

}

steps {

echo 'Deploying'

}

}

}

}

========

Assignment based on condition:-

1. continue on error:-> if stage 1 is failed then continue with stage-2

ex: stage-1 -> scan the code (if scanning failed/or code coverage is not greater than 90%),

stage-2-> build the code (continue on error and build the package)

pipeline

{ agent any

stages

{

stage('stage-1')

stage('stage-2')

}

}

-------conditional

1. continue on error:-> if stage 1 is failed then continue with stage-2

ex: stage-1 -> scan the code (if scanning failed/or code coverage is not greater than 90%),

stage-2-> build the code (continue on error and build the package)

2. when branch is "prod" then deploy to prod env if not then skip the stage

pipeline {

agent any

stages {

stage('Example Build') {

steps {

echo 'Hello World'

}

}

stage('Example Deploy') {

when {

branch 'production' //jenkins will check branch if branch not found then skip the stage

}

steps {

echo 'Deploying'

}

}

}

}