

Install JAVA ON MASTER

```
sudo amazon-linux-extras install java-openjdk11 -y
sudo update-alternatives --config java
vim ~/.bash_profile
JAVA_HOME="/usr/lib/jvm/java-11-openjdk-11.0.11.0.9-1.amzn2.0.1.x86_64/bin/java"
source ~/.bash_profile
echo $JAVA_HOME
```

Install Jenkins on MASTER

```
sudo wget -O /etc/yum.repos.d/jenkins.repo \
https://pkg.jenkins.io/redhat-stable/jenkins.repo

sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
sudo yum upgrade -y
sudo yum install jenkins java-11-openjdk-devel -y
sudo systemctl daemon-reload

sudo service jenkins start
sudo service jenkins status
```

```
sudo systemctl status Jenkins
sudo systemctl start jenkins.service
```

OPTIONAL

```
sudo yum install httpd -y ( global config error path error maven http error 403) (ERROR1)
```

Enter the public ip of the master Jenkins server

Public_ip_of_master:8080

Get the initial admin password of Jenkins

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

Install suggested plugins

Create a jenkins user named master from jenkins dashboard

Install git and maven on master & slave

sudo yum install git -y

sudo yum install maven -y

Repo clone and pushed to github

<https://github.com/laxapatiakshaylearning/cicdpipelinejenkins>

Setup master slave

Slave setup on master

Manage jenkins → Manage node & clouds → New Node → give node name linux_node1 → add to permanent

Remote root directory

/home/ec2-user/

Tick on (use websocket) & save

Click on the slave node & right click on agent.jar then copy link address

On slave machine

Download the agent.jar file at /home/ec2-user using wget

wget <http://54.198.69.61:8080/jnlpJars/agent.jar>

Join from slave using below command

```
java -jar agent.jar -jnlpUrl http://54.198.69.61:8080/computer/linux_slave1/jenkins-agent.jnlp  
-secret aa224035b5f55c28c926847bcf9bf60d4cf2dcf75b53be7fb350581a0413b874 -workDir  
"/home/ec2-user/"
```

```
java -jar agent.jar -jnlpUrl http://54.147.245.0:8080/computer/linux_node1/jenkins-agent.jnlp  
-secret a4d1e2ae0a49dc202811fe9b1a34accf06247077e31b64278306ba562ba1624e -workDir  
"/home/ec2-user/"
```

Check if the node/slave is connected

Configure global config

Manage jenkins → Global tool configuration

Jdk

Add jdk

Name: java

Untick install automatically

Java_home =/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.302.b08-0.amzn2.0.1.x86_64

/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.302.b08-0.amzn2.0.1.x86_64

Git

To find the path of git below is the command

Whereis git

/usr/bin/git (ERROR1 optional)

Maven add maven

Name : maven

Untick install automatically

To find the path of maven below is the command

Mvn -v

/usr/share/maven

Name : maven

Path : /usr/share/maven

Click Save

SETUP SONAR ON NEW SERVER

Launch at least instance type t2.medium because the minimum ram required for sonar is 4 gb

Install java

```
sudo amazon-linux-extras install java-openjdk11 -y
```

Check java path

```
sudo update-alternatives --config java
```

```
/usr/lib/jvm/java-11-openjdk-11.0.11.0.9-1.amzn2.0.1.x86_64/bin/java
```

Set environment variable

```
vim ~/.bash_profile
```

```
JAVA_HOME="/usr/lib/jvm/java-11-openjdk-11.0.11.0.9-1.amzn2.0.1.x86_64/bin/java"
```

```
source ~/.bash_profile
```

Install sonarqube

download binary of sonar

```
wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-8.1.0.31237.zip
```

```
unzip sonarqube-8.1.0.31237.zip
```

```
cd sonarqube-8.1.0.31237/bin/linux-x86-64/
```

```
./sonar.sh console
```

Login using admin as username & password

<http://50.17.69.175:9000/> (public ip of sonar server : 9000)

Navigate to admin logo which is at top right corner → click on my account

→ security → enter token name : sonaradmin → click on generate & copy token

d1ced1005c661658d60e9cfcfb8eae0002232cd0

Go to jenkins master server

Install sonar plugin

Managejenkins → manage plugins → available plugins → search plugin

[SonarQube Scanner](#)

& install without restart

Manage Jenkins → Configure system

SonarQube servers

Tick

Environment variables Enable injection of SonarQube server configuration as build environment variables

Click Add sonarqube

Name : sonarqube

http://publicip_of_sonarqube:9000

http://34.203.248.145:9000

Add credentials for sonarqube

Server authentication token

Add → jenkins → secret text in kind → Secret -- paste the token that was copied from sonar server → Description : sonar token →

click on none & select sonar token from drop down menu

Tools Path configuration

Manage jenkins → manage nodes & clouds → click on settings of node → tools location add →

GIT

To find the path of git below is the command

Whereis git (Execute this command on slave node)

/usr/bin/git

JAVA

To find the path of git below is the command

update alternatives command (Execute this command on slave node)

/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.302.b08-0.amzn2.0.1.x86_64

MAVEN

To find the path of maven below is the command

mvn -v (Execute this command on slave node)

/usr/share/maven

Create a freestyle project

Create a job → name for the job : project1 → Freestyle project

Add the git repo

<https://github.com/laxapatiakshaylearning/cicdpipelinejenkins.git>

I have not added credentials as the repo is public

Add webhook in the git portal

Go to git web repo name

Settings webhook add webhook

Payload url :

http://public_ip_of_jenkins:8080/github-webhook/

<http://54.162.56.96:8080/github-webhook/>

Content type

application/json

configure

Let me select individual events

Tick

Branch or tag creation

Commit comments

Forks

Issues

Pull request review comments

Pull request review threads

Pull request reviews

Pull requests

Pushes

Active

Click on add/update webhook

Confirm the dot is replaced by tick, by this we can confirm that the connection is established between github & jenkins master server. The connection is checked by sending a packet to the jenkins server & upon receiving the response from the jenkins server it confirms the connectivity

Enable webhook trigger on jenkins master

GitHub hook trigger for GITScm polling

Build

Add build step

Invoke top level maven targets

Replace default with maven from drop down menu

Goal

clean install

Save → build now

(check all the components are working properly & then add sonar)

Now after checking that the build is successfully build add sonar scanner

Inside project manage

Tick on Prepare SonarQube Scanner environment &

Modify maven goal to

clean install sonar:sonar

Check if the build is successful

And we are able to see the sonar report at sonar dashboard & jenkins dashboard

Check if the webhook is triggered from git

Make some changes into the repo and commit

Now check if jenkins job is getting triggered by webhook below is the text that can be observed in the console output of the build number

Started by GitHub push by laxapatiakshaylearning

Check if we are able to see results on sonarqube server as well as on jenkins

Add jacoco plugin

Manage plugins

[JaCoCo](#)

Install without restart

Configure on job

Post-build Actions

Record jacoco coverage report

Apply and save → click on build & check if report is generated → coverage report

Manually build & check if report is generated

CONFIGURE QUALITY GATES

Add plugin named [Sonar Quality Gates](#)

Configure the plugin

Manage jenkins → Configure system

Quality Gates - Sonarqube → Add Sonar instance

Quality Gates - Sonarqube

Name : sonargate

SonarQube Server URL: <http://54.227.53.61:9000>

SonarQube account token : add token

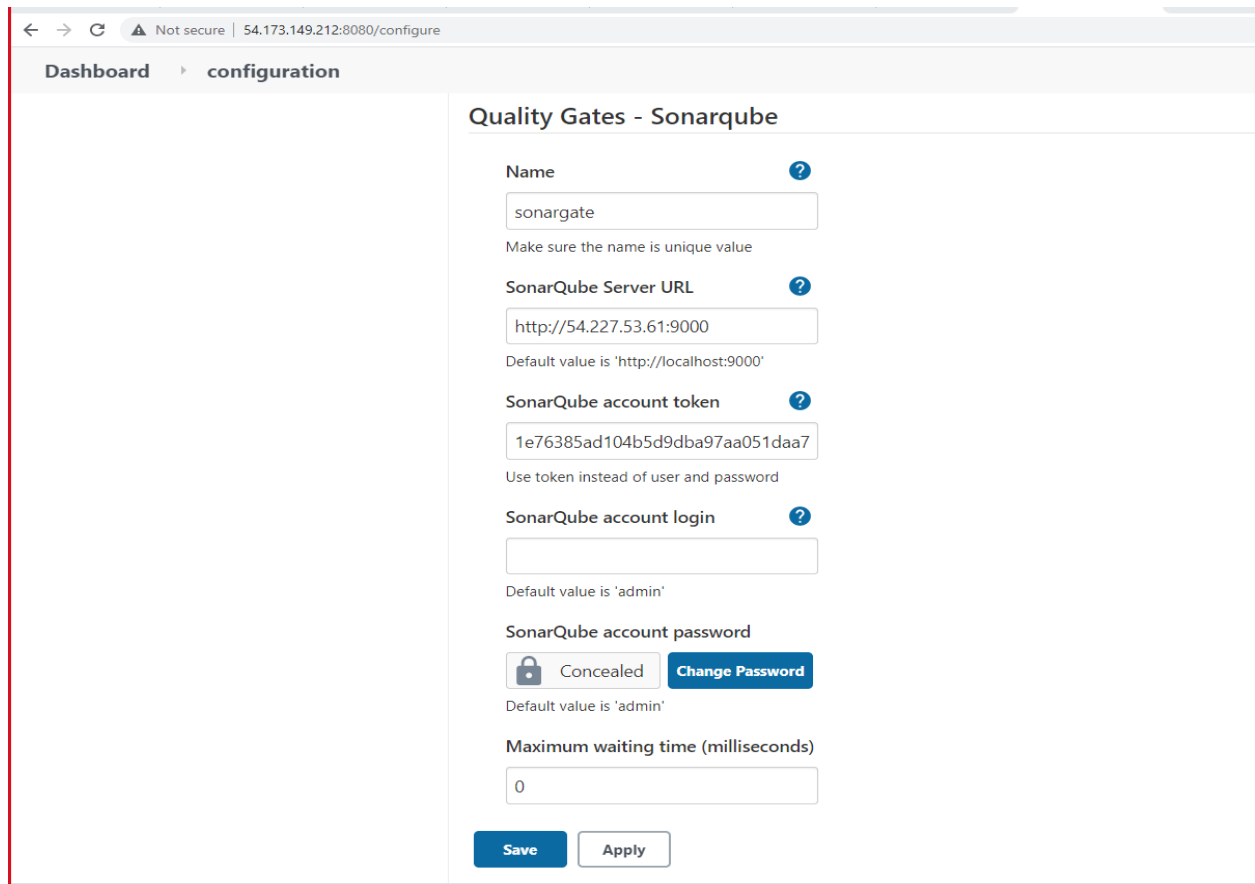
Username : admin

Password : admin

Minimum & maximum waiting time remove 0

Maximum waiting time (milliseconds)

Time to wait next check (milliseconds)



The screenshot shows the SonarQube configuration interface for Quality Gates. The browser address bar indicates the URL is 54.173.149.212:8080/configure. The page has a header with 'Dashboard' and 'configuration' tabs. The main content area is titled 'Quality Gates - Sonarqube'. It contains several configuration fields: 'Name' (sonargate), 'SonarQube Server URL' (http://54.227.53.61:9000), 'SonarQube account token' (1e76385ad104b5d9dba97aa051daa7), 'SonarQube account login' (empty), and 'SonarQube account password' (Concealed). There are also buttons for 'Save' and 'Apply'.

Dashboard > configuration

Quality Gates - Sonarqube

Name ⓘ
sonargate
Make sure the name is unique value

SonarQube Server URL ⓘ
http://54.227.53.61:9000
Default value is 'http://localhost:9000'

SonarQube account token ⓘ
1e76385ad104b5d9dba97aa051daa7
Use token instead of user and password

SonarQube account login ⓘ

Default value is 'admin'

SonarQube account password
Concealed **Change Password**
Default value is 'admin'

Maximum waiting time (milliseconds)
0

Save **Apply**

APPLY → SAVE

BUILD AND CHECK SONAR QUALITY GATE IS PASSED

Now we will create a custom sonar quality gate to check if the build fails

Login to sonar qube portal

CREATE CUSTOM QUALITY GATE

Quality gate → give name (customgateforcid)

Add condition

Select overall code

Code smells is greater than 8

Coverage 96 (if code coverage is less than 96 percent the gate must fail)

Lines to cover 20 (if the lines

Uncovered line

CHECK IF THE STATUS FAILED IS DISPLAYED ON PROJECT WHEN NEW QUALITY GATE IS APPLIED

SonarQube Quality Gate

java-maven-junit-helloworld

FAILED

NOW ADD THE QUALITY GATE BUILD STEP INSIDE THE JOB

MANAGE THE JOB

Add post build action

Quality gates sonarqube plugin

Project key : copy from the project dashboard of sonarqube

Job status when analysis fails : FAILED

CHECK IN THE BUILD CONSOLE THE FOLLOWING LINES & BUILD IS FAILED

PostBuild-Step: Quality Gates plugin build passed: FALSE

Build step 'Quality Gates Sonarqube Plugin' marked build as failure

Finished: FAILURE

MAVEN BUILD PROCESS IS COMPLETED

NOW START THE DOCKER BUILD PROCESS

Reset the quality gate to default to check the next build process from sonarqube dashboard

DOCKER

Add following docker plugins

Docker

docker- build-step

Cloudbees

Install without restart

Install docker on linux slave machine

```
sudo amazon-linux-extras install docker -y
```

```
sudo service docker start
```

```
sudo docker info
```

CREATE JOB 2 for docker build & push to docker hub

Copy from job 1 & edit

Remove sonarqube quality gates plugin

ADD DOCKER BUILD & PUBLISH PLUGIN in add build step

Create a public docker repo name in docker hub & add it in jenkins

Repo name

laxapatiakshaylearning/cicdnewjenkins

Tag

\$BUILD_ID

We build docker image on the jenkins server itself so we do not add server url

Add registry credentials

Username : laxapatiakshaylearning@gmail.com

Password:

Description: docker credentials

Select docker credentials save & build the job

Now restrict docker to run on the slave machine only

Restrict where this project can be run : linuxnode1

IF we get the docker error as below

```
[cicdpipeline] $ docker build -t laxapatiakshaylearning/cicdnewjenkins:7 --pull=true  
/var/lib/jenkins/workspace/cicdpipeline  
Got permission denied while trying to connect to the Docker daemon socket at  
unix:///var/run/docker.sock: Post  
"http://%2Fvar%2Frun%2Fdocker.sock/v1.24/build?buildargs=%7B%7D&cachefrom=%5B%5D&cgroupparent=&cpuperiod=0&cpuquota=0&cpusetcpus=&cpusetmems=&cpushares=0&dockerfile=Dockerfile&labels=%7B%7D&memory=0&memswap=0&networkmode=default&pull=1&rm=1&shmsize=0&t=laxapatiakshaylearning%2Fcicdnewjenkins%3A7&target=&ulimits=null&version=1": dial unix /var/run/docker.sock: connect: permission denied  
Build step 'Docker Build and Publish' marked build as failure  
Finished: FAILURE
```

Do the following

Execute the below command on slave (because we build the job on slave machine where docker is installed)

```
sudo chmod 777 /var/run/docker.sock
```

Check on docker hub if the repo is pushed with the image id tag of build number

Note : that the job id is appended to the tag of docker hub

Now check if job1 is able to trigger job 2 if sonar gates are passed

As the default gates are used the build will be passed configure the below to trigger job 2 when job 1 is successfully completed

Edit project 1

ADD POST BUILD ACTION :

build other projects

Enter : project2

Tick : Trigger only if build is stable

Build job 1 : and confirm the below lines in the console output of job 1

Status => SUCCESS

PostBuild-Step: Quality Gates plugin build passed: TRUE

Triggering a new build of [project2](#)

Finished: SUCCESS

Confirm the job2 is completed

Check in the docker hub if the new image is pushed

Now apply the custom gate from sonar & check if the build fails

Manually build job 1

PostBuild-Step: Quality Gates plugin build passed: FALSE

Build step 'Quality Gates Sonarqube Plugin' marked build as failure

Finished: FAILURE

CREATE JOB 3 Named cd job

Create a third project for cd job

Execute shell

docker pull laxapatiakshaylearning/cicdnewjenkins:latest

docker run -d --name jenkins laxapatiakshaylearning/cicdnewjenkins:latest

We have to run the cd job manually

CONFIRM IF THE JOB IS RUNNING ON SLAVE CONTAINER

```
[ec2-user@ip-172-31-81-126 ~]$ sudo docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

```
[ec2-user@ip-172-31-81-126 ~]$ sudo docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
a38d21686d2e	laxapatiakshaylearning/cicdnewjenkins:latest	"java -jar java-mave..."	19 seconds ago
Exited (1)	18 seconds ago	jenkins	

TILL THIS STEP THE CI CD IS COMPLETED