

# Jenkins Sonar And Argo CD Doc

Create an ec2 instance t2.xlarge

Connect ec2 with ssh from local

Installations

## **Maven**

```
sudo yum update
sudo yum install -y maven
mvn -version
```

## **Now Install openjdk 11**

```
sudo yum install java-11-amazon-corretto-headless
```

## **Now Install Docker**

```
sudo yum update -y
sudo yum install -y docker
sudo service docker start
sudo chkconfig docker on - This command configures the Docker service to start
automatically on system boot
```

Overall, these commands update the system, install Docker, start the Docker service, and set it to automatically start on system boot, allowing you to work with Docker and run containerized applications on your Linux system.

## **Now install Jenkins**

```
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-2023.key
sudo yum install jenkins -y
sudo systemctl enable jenkins
sudo systemctl start jenkins
```

```
yum update -y yum.noarch
```

## Install Node

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash
```

```
./ ~/.nvm/nvm.sh
```

```
nvm install 16
```

### Verification:

```
node -e "console.log('Running Node.js ' + process.version)"
```

## Install Git

```
sudo yum install -y git
```

## Add following things in security grp

EC2 > Security Groups > sg-057736958663ac14c - launch-wizard-1 > Edit inbound rules

### Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type <a href="#">Info</a>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>	Source <a href="#">Info</a>	Description - optional <a href="#">Info</a>		
sg-01d40ed0a9bf4a0be	Custom TCP ▼	TCP	8080	Custom ▼	Q	<a href="#">0.0.0.0/0</a> ✕	Delete
sg-05796b39626d38970	HTTP ▼	TCP	80	Custom ▼	Q	<a href="#">0.0.0.0/0</a> ✕	Delete
sg-07d94e94ad9fbcee6	SSH ▼	TCP	22	Custom ▼	Q	<a href="#">0.0.0.0/0</a> ✕	Delete
sg-0d09b4dbe5cc12300	HTTPS ▼	TCP	443	Custom ▼	Q	<a href="#">0.0.0.0/0</a> ✕	Delete

### **To check the status of jenkins**

```
sudo systemctl status jenkins
```

### **After this get password from below command in jenkins**

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

### **After Jenkins installation add ssh plugin**

Manage jenkins - plugin

- Available plugin - [SSH Agent](#) - install without restart
- Manage jenkins - tools- Add Maven
- 

### **To integrate webhook in jenkins**

Webhooks / Add webhook

We'll send a `POST` request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, `x-www-form-urlencoded`, etc). More information can be found in [our developer documentation](#).

---

Payload URL \*

`http://18.168.203.73:8080/github-webhook/`

Content type

`application/json`

Secret

---

Which events would you like to trigger this webhook?

☒ Just the push event.

☐ Send me everything.

☐ Let me select individual events.

Create ssh key in jenkins server using command  
`ssh-keygen -t rsa -b 4096`

Command to fetch public key -  
`cat ~/.ssh/id_rsa.pub`

Command to fetch private key -  
`cat ~/.ssh/id_rsa`

**Add this public key in git hub . go to setting go to ssh key and add new key**

Add private key in jenkins now  
Go to jenkins - manage jenkins - credentials - Add ssh username with private key

Add docker creds - username with password  
Codedecode25  
DOCKER\_HUB\_CREDENTIAL - id

Also add this no verification which will be used during git push

Dashboard > Manage Jenkins > Security

☐ Random

☒ Disable

---

Git Host Key Verification Configuration

Host Key Verification Strategy ?

No verification

⚠ This option is generally insecure. Host key will not be verified during SSH connection

Save Apply

Go to jenkins home. Click new item. Click on pipeline. Copy Restaurant Service Http clone url

rl Config [Jenkins]

Not secure | 52.47.128.79:8080/job/rl/configure

Dashboard > rl > Configuration

Configure

General

Advanced Project Options

Pipeline

Discard old builds ?

Do not allow concurrent builds

Do not allow the pipeline to resume if the controller restarts

GitHub project

Pipeline speed/durability override ?

Preserve stashes from completed builds ?

This project is parameterized ?

Throttle builds ?

Build Triggers

Build after other projects are built ?

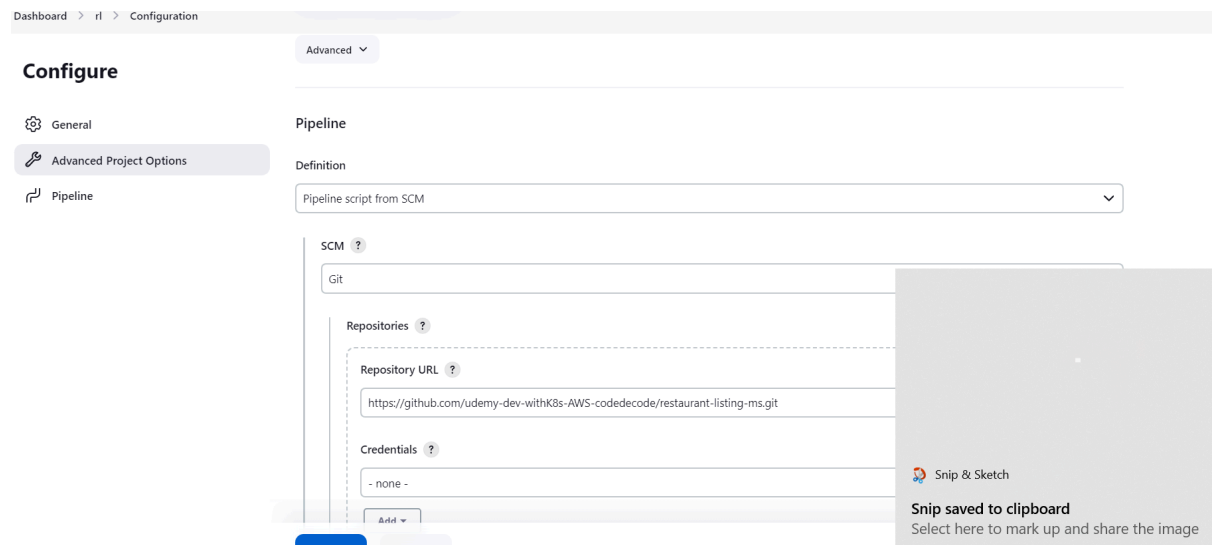
Build periodically ?

☒ GitHub hook trigger for GITScm polling ?

Poll SCM ?

Quiet period ?

Save Apply



save n add dummy commit build automatically triggers  
Nowcheck manifest file

To provide access to docker

```
sudo usermod -aG docker jenkins
```

Restart jenkins after that

```
sudo service jenkins restart
```

## Sonar

**Run Sonar in docker using below command**

```
docker run -d -p 9000:9000 --name sonarqube sonarqube
```

**To check the logs**

```
docker logs -f sonarqube
```

**Hit**

15.188.80.32:9000 for sonar dashboard and username password will be admin

## To generate token go to sonar dashboard

Click on user icon - my account - security and create new token

## Quality Gates

The screenshot shows the SonarQube Quality Gates configuration page. The top navigation bar includes 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', 'Administration', and 'More'. The 'Quality Gates' section is active, showing a list of Sonar rules. The rules are organized into a table with columns for the rule name, operator, and value. The rules include 'Security Hotspots Reviewed' (is less than 100%), 'Maintainability Rating' (is worse than A), 'Coverage' (is less than 80.0%), and 'Duplicated Lines (%)' (is greater than 3.0%). Below this, there is a section for 'Conditions on Overall Code' with a table of metrics like 'Coverage' and 'Line Coverage'. The interface also includes a sidebar with 'Sonar rules' and 'Sonar way' tabs, and a top navigation bar with 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', 'Administration', and 'More'.

## To run sonar in local n push report in sonar dashboard

mvn clean org.jacoco:jacoco-maven-plugin:prepare-agent install sonar:sonar

-Dsonar.host.url=http://15.188.80.32:9000/

-Dsonar.login=squ\_32789bcdadb6e4337e432d6cbc100c2a1a14fde5

## Now put following dependencies in pom

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <configuration>
    <source>1.8</source>
    <target>1.8</target>
    <annotationProcessorPaths>
      <path>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.26</version>
      </path>
      <path>
        <groupId>org.mapstruct</groupId>
        <artifactId>mapstruct-processor</artifactId>
        <version>1.4.2.Final</version>
      </path>
    </annotationProcessorPaths>
  </configuration>
</plugin>
```

```

        </annotationProcessorPaths>
    </configuration>
</plugin>
<plugin>
    <groupId>org.sonarsource.scanner.maven</groupId>
    <artifactId>sonar-maven-plugin</artifactId>
    <version>3.8.0.2131</version>
</plugin>
<plugin>
    <groupId>org.jacoco</groupId>
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>0.8.8</version>
    <executions>
        <execution>
            <id>prepare-agent</id>
            <goals>
                <goal>prepare-agent</goal>
            </goals>
        </execution>
        <execution>
            <id>report</id>
            <goals>
                <goal>report</goal>
            </goals>
        </execution>
    </executions>
</plugin>

```

**Add this command in properties**

```

<sonar.exclusions>**/com/codeddecode/restaurantlisting/dto/** ,
**/**/com/codeddecode/restaurantlisting/entity/**/*</sonar.exclusions>

```

## To generate token go to sonar dashboard

Click on user icon - my account - security and create new token

## Where to get component key

Click on project – project setting – update key

Here you will find component key

```
<groupId>:<artifactId>
```

## Run sonar using below command

```

mvn clean org.jacoco:jacoco-maven-plugin:prepare-agent install sonar:sonar
-Dsonar.host.url=http://localhost:9000/
-Dsonar.login=squ_a44ef243148d9f75cb3248851df2d555d5342ee2

```

After this delryr ptpject



# Argo CD

Command to install argo cd in cluster

```
kubectl create namespace argocd
```

```
kubectl apply -n argocd -f  
https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.y  
aml
```

```
kubectl port-forward svc/argocd-server -n argocd 8080:80
```

To Get the password

Initial username

```
argocd admin initial-password -n argocd
```

```
To install argocd cli  
choco install argocd-cli
```

Create ssh key in jenkins server using command

```
ssh-keygen -t rsa -b 4096
```

Command to fetch public key -

```
cat ~/.ssh/id_rsa.pub
```

Command to fetch private key -

```
cat ~/.ssh/id_rsa
```

```
To delete argocd completely
```

```
kubectl delete -n argocd -f  
https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.y  
aml
```

```
If pods are in terminating state
```

```
kubectl delete pods --all -n argocd --grace-period=0 --force
```

```
Go to setting - repository - connect repo
```

Overall, the provided YAML describes an Argo CD "Application" resource that deploys application resources from a Git repository, with specific configuration options for the Argo CD Image Updater and synchronization behavior.