



CI PIPELINE SETUP WITH VARIOUS TOOLS

****You need to complete this in one go ****

100 rupees max it will cost

[8 GB → 30 GB] and t2.micro -> t2.medium [

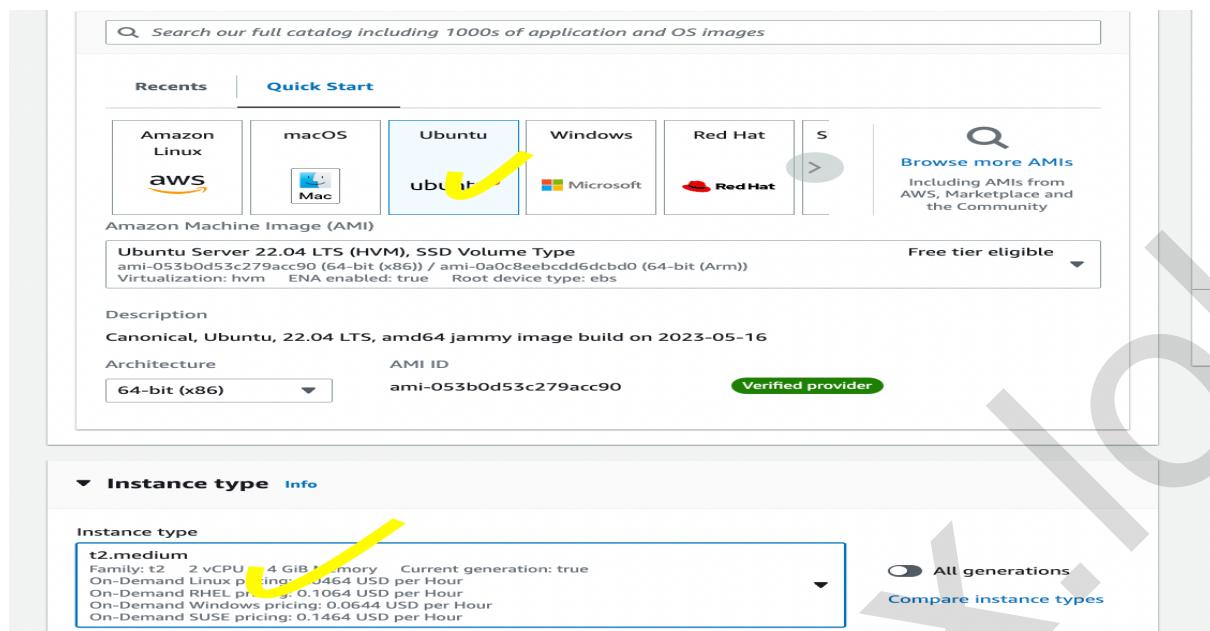
More memory and more cpu]

Step 1 – Create the Ec2 instance in AWS account with these parameters

EC2 type – Ubuntu t2.medium

EBS volume – 30 GB

Region – US-EAST-1



Step 2 – Connect to EC2 and Install all tools in that system as root user

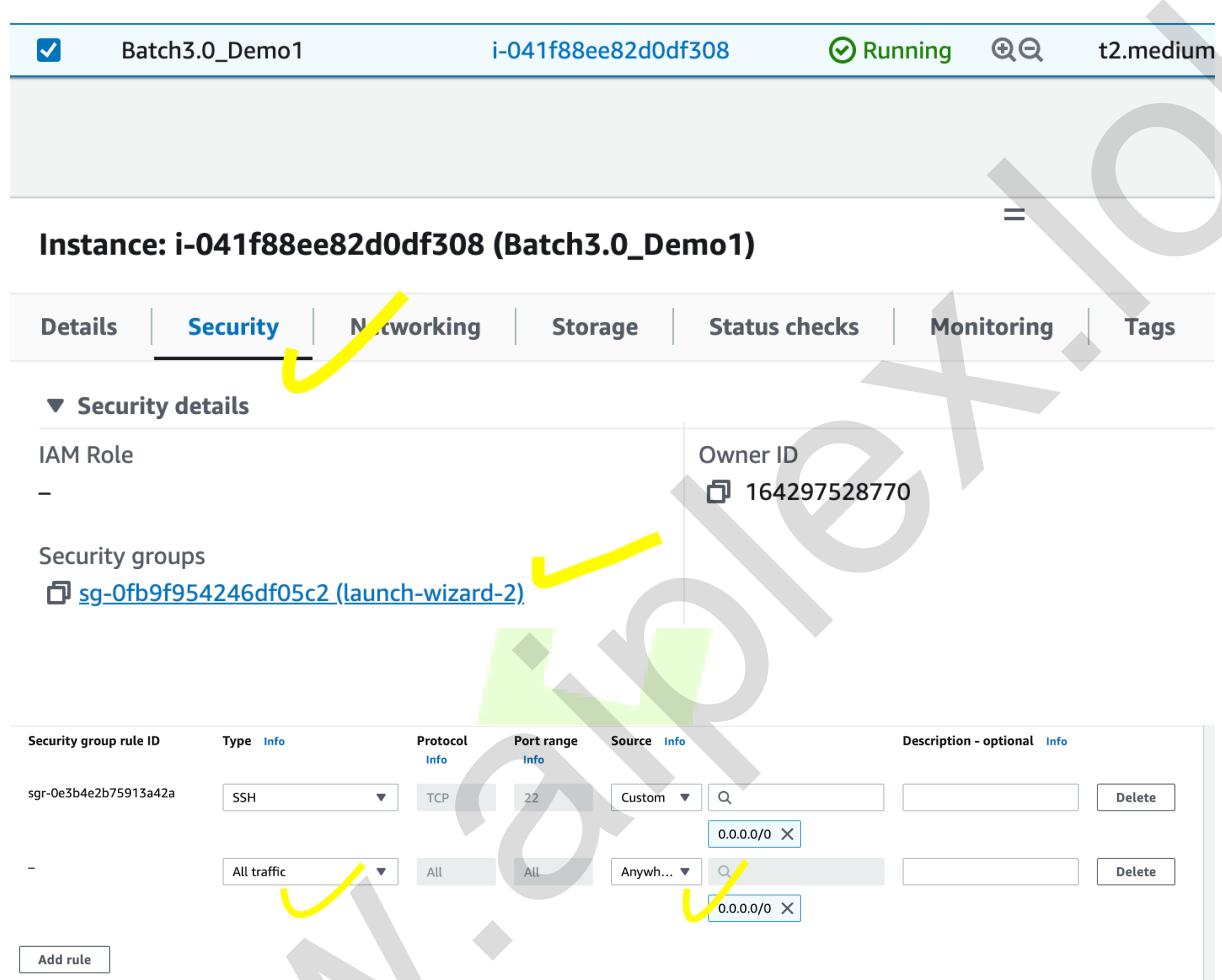
To login as root user - sudo su

Step 3 – Install Jenkins on Ubuntu

Just copy paste the entire commands

https://github.com/praveen1994dec/tools_installation_scripts/blob/main/jenkins.sh

Step 4 – Change the security group of ec2 instance



Step 5 – Sign Into Jenkins console

http://<EC2_PUBLIC_IP>:8080/

Step 6 – Get the Administrator password by hitting the below command in EC2

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

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password is stored in the log ([not sure where to find it?](#)) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

```
.....
```

Step 7 – Install all suggested plugins

Step 8 – Create first user

Getting Started

Create First Admin User

Username

Password

Confirm password

Full name

Jenkins 2.401.1

Skip and continue as admin

Save and Continue

Step 9** – Create a pipeline Job

Enter an item name

Demo_3.0
» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM v even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for build workflows) and/or organizing complex activities that do not easily fit in free-style job type

Step 10 – Add pipeline script as SCM

https://github.com/praveen1994dec/Java_app_3.0.git

Configure Pipeline

General Advanced Project Options Pipeline

Definition

Pipeline script from SCM

SCM Git

Repositories

Repository URL https://github.com/praveen1994dec/Java_app_3.0.git

Credentials none

Add Repository

Advanced

Branches to build

Branch Specifier (blank for 'any') * *main

Add Branch

Repository browser (Auto)

Additional Behaviours

Script Path Jenkinsfile

Lightweight checkout

Step 11 – Add the Plugins

**Dashboard -> Manage Jenkins -> Plugins ->
Available plugins**

Plugins for Sonar/Jfrog –

Sonar Gerrit

SonarQube Scanner

SonarQube Generic Coverage

Sonar Quality Gates

Quality Gates

Artifactory

Jfrog

The screenshot shows the Jenkins 'Manage Jenkins' section under 'Plugins'. A search bar at the top is set to 'Artifactory'. Below it, a table lists several plugins:

- Sonar Gerrit**: Version 377v8f3808963dc5, External Site/Tool Integrations. Description: Allows to submit issues from SonarQube to Gerrit as comments directly. Warning: CSRF vulnerability.
- SonarQube Scanner**: Version 2.15, External Site/Tool Integrations, Build Reports. Description: Allows an easy integration of SonarQube, the open source platform for Continuous inspection of code quality.
- Sonar Quality Gates**: Version 1.3.1, TODO. Description: Fails the build whenever the Quality Gates criteria in the Sonar 5.6+ analysis aren't met (the project Quality Gates status is different than "Passed"). Warning: Credentials transmitted in plain text.
- Quality Gates**: Version 2.5, pipeline. Description: Fails the build whenever the Quality Gates criteria in the Sonar analysis aren't met (the project Quality Gates status is different than "Passed"). Warning: Credentials transmitted in plain text.
- Artifactory**: Version 3.18.3, pipeline. Description: Allows your build jobs to deploy artifacts and resolve dependencies to and from Artifactory, and then have them linked to the build job that created them. The plugin includes a vast collection management for Maven and Gradle builds with Staging and Promotion.
- JFrog**: Version 1.4.0, .NET Development, Maven, npm, Deployment, docker. Description: Allows your build jobs to deploy artifacts and resolve dependencies to and from Artifactory, also allows you to scan your artifacts and builds with JFrog Xray and distribute your software package to remote locations using JFrog Distribution. This is all achieved by the plugin by wrapping JFrog CI Jenkins Pipeline job using the JFrog Plugin.

Step 12 – Setup Docker

https://github.com/praveen1994dec/tools_installation_scripts/blob/main/docker.sh

`docker -v`

Step 13- Install SonarQube

https://github.com/praveen1994dec/tools_installation_scripts/blob/main/sonarqube.sh

Step 13 .1 -> Start docker container if it's not up

docker ps -a [Get the container ID]

```
root@ip-172-31-64-23:/home/ubuntu# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
 NAMES
e10fb52aa1a        sonarqube          "/opt/sonarqube/dock..."   6 seconds ago      Up 3 seconds      0.0.0.0:9000->9000/tcp, :::9000->9000/tcp, 0.0.0.0:9092->9092/tcp, :::9092->9092/tcp
```

docker start <containerID>

Step 13.2 -> Login into sonar dashboard

Username – admin

Password – admin

Step 13.3 -> Create Sonar token for Jenkins

Sonar Dashboard -> Administration -> My Account -> Security -> Create token -> Save the token to some text file

The screenshot shows the SonarQube administration interface under the 'Tokens' section. It includes fields for 'Name' (Jenkins!), 'Type' (Global Analysis Token), 'Expires in' (30 days), and a 'Generate' button. The 'Actions' column contains links for 'Expiration' and 'Delete'.

Step 13.4 -> Integrate Sonar to Jenkins

Sonar Dashboard -> Administration -> Configuration -> webhooks -> Add the below name and url and save

http://<EC2_IP>:8080/sonarqube-webhook/

Create Webhook

All fields marked with * are required

Name *
Jenkins 

URL *
`http://18.205.67.145:8080/sonarqube-webhook/` 

Server endpoint that will receive the webhook payload, for example: "http://my_server/foo". If HTTP Basic authentication is used, HTTPS is recommended to avoid man in the middle attacks. Example: "https://myLogin:myPassword@my_server/foo"

Secret
 If provided, secret will be used as the key to generate the HMAC hex (lowercase) digest value in the 'X-Sonar-Webhook-HMAC-SHA256' header

Create **Cancel**

Step 14 – Install Maven

https://github.com/praveen1994dec/tools_installation_scripts/blob/main/Maven.sh

Step 15 – Install TRIVY for docker image scan

https://github.com/praveen1994dec/tools_installation_scripts/blob/main/trivy.sh

Integrate All tools with Jenkins

Jenkins Dashboard -> Manage Jenkins -> configure system

Step 16 – ADD SONARQUBE

SonarQube installations

List of SonarQube installations

Name	Server URL	Server authentication token
sonar-api	Default is http://localhost:9000 http://18.205.67.145:9000	SonarQube authentication token. Mandatory when anonymous access is disabled.

Step 16.1 -> Click on sonarqube servers -> add url and name -> Click on add token -> Select Secret text -> Add the sonar token from step13.3 -> Give name of token as **sonarqube-api**

The screenshot shows the Jenkins System configuration page under Manage Jenkins > System. A yellow arrow points to the 'SonarQube servers' section. Another yellow arrow points to the 'Name' field containing 'sonar-api'. A third yellow arrow points to the 'Server URL' field containing 'http://18.205.67.145:9000'. A fourth yellow arrow points to the 'Server authentication token' field containing 'sonarqube-api'. A red box highlights the 'Add' button at the bottom left.

Step 17 - Add the docker HUB credentials ID

Jenkins dashboard -> Manage Jenkins -> Credentials -> System -> click on global credentials

The screenshot shows the Jenkins System configuration page under Manage Jenkins > System. A blue button at the top right says '+ Add domain'. Below it is a table with two columns: 'Domain' and 'Description'. The first row shows 'Global credentials (unrestricted)' with a description: 'Credentials that should be available irrespective of domain specification to requirements matching.'

ADD the docker hub credentials with name as **docker**

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind

Username with password

Scope

Global (Jenkins, nodes, items, all child items, etc)

Username

praveensingam1994

Treat username as secret

Password

.....

ID

docker

Description

Step 18 – Add the Jenkins Shared library

Go to Manage Jenkins -> Configure system -> Global pipeline library -> Add below data

Name - my-shared-library

Default version – main

Git -

https://github.com/praveen1994dec/jenkins_shared_lib.git

Global Pipeline Libraries

Sharable libraries available to any Pipeline jobs running on this system. These libraries will be trusted, meaning they run without "sandbox" restrictions and may use \${crab}.

Library
Name ?

Default version ?

- Load implicitly ?
- Allow default version to be overridden ?
- Include @Library changes in job recent changes ?
- Cache fetched versions on controller for quick retrieval ?

Retrieval method

Modern SCM

Loads a library from an SCM plugin using newer interfaces optimized for this purpose. The recommended option when available.

Source Code Management

Git

Project Repository ?

Credentials ?

- none -

Add ▾

Behaviors

Discover branches ?

Add ▾

Fresh clone per build ?

Library Path (optional) ?

Step 19 - Once pipeline is Run Check

- The Jenkins logs
- The Trivy scan vulnerabilities
- The sonarqube dashboard for report

The Jenkins console output shows the build process starting from a GitHub repository and cloning a shared library.

```
Started by user admin
Obtained Jenkinsfile from git https://github.com/praveen1994dec/Java_app_3.0.git
Loading library my-shared-library@main
Attempting to resolve main from remote references...
> git --version # timeout=10
> git --version # 'git version 2.34.1'
> git ls-remote -- https://github.com/praveen1994dec/jenkins_shared_lib.git # timeout=10
Found match: refs/heads/main revision 610e218e6018f9de0d3230c2b706eb02794456
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/BATCH4@libs/86e3083c89e31f8f4c9341a7f237a4faa553b7d750eb8ce2bfc88dc4481eal/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/praveen1994dec/jenkins_shared_lib.git # timeout=10
Fetching without tags
> git fetch --tags
> git rev-parse --verify HEAD # timeout=10
Fetching upstream changes from https://github.com/praveen1994dec/jenkins_shared_lib.git
> git --version # timeout=10
```

The SonarQube dashboard shows the project 'minikube-sample' has passed its quality gate.

Quality Gate Status: Passed

Measures:

- Reliability: 0 Bugs (Grade A)
- Maintainability: 1 Code Smells (Grade A)
- Security: 0 Vulnerabilities
- Security Review: 0 Hotspots
- Coverage: 0%
- Duplications: 0%



Get more free courses at www.aiplex.lol