cd /c/Users/rchan/Downloads/

Nexus-sg 🡺

ssh – port 22 -- myip(source type)

custom tcp – port 8081 – myip(source type)

custom tcp – port 8081 –myip(source type) – source – Jenkins sg

Jenkins-sg 🡪

ssh – 22 –myip

custom ip -- 8080 –myip

**My IP:** (Recommended for testing) - This allows access only from your current public IP address.

**Custom IP:** Enter your public IP address or a range of IPs (CIDR notation) if you need to allow access from multiple locations.

**0.0.0.0/0 (Not Recommended for Production):** This allows access from anywhere on the internet. **Only use this for temporary testing and immediately restrict it afterwards.** It's a major security risk.

Jenkins dashboard error

Add ICMP to VPC>>NACL>>inbound rules >> add ICMP

🡪cd c/Users/rchan/Downloads

Installation:

sudo apt-get update

sudo apt-get install openjdk-17-jdk -y

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \

https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \

https://pkg.jenkins.io/debian-stable binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update

sudo apt-get install jenkins -y

Freestyle vs pipeline as a code

Freestyle jobs

* Graphical jobs
* Learning, understanding & exploring Jenkins
* Not recommended in real time now

Pipeline as a code – groovy

By default Jenkins comes with maven plugin but we need to install tools

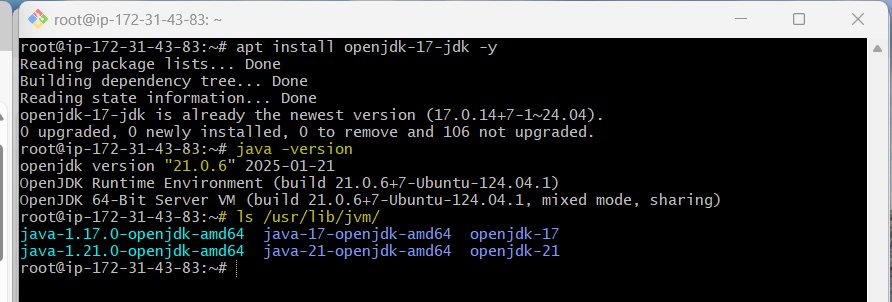
Jenkins url

<http://18.205.239.113:8080/>

manage Jenkins>>tools>>add JDK17 and MAVEN3.9

but we have only JDK21 installed, if you see java version we only see java21, if we go to the path we see jdk17 as we installed now jdk17 as below:

apt install openjdk-17-jdk -y



steps for CI pipeline:

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

Jenkins setup

Nexus setup – amazon linux 2023

Sonarqube setup ubuntu 24

Security grp

Plugins

Integrate – nexus, sonarqube

Pipeline script

Set notification

Plugins:

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

Nexus

Sonarqube

Git

Pipeline maven integration plugin

Build timestamp

Pipeline:

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

Automate the pipeline setup with jenkin sfile

Jenkins file defines stages in cicd pipeline

Jenkins file is a text file with pipeline DSL syntax.

If we are writing the entire Jenkins pipeline in text format, it contains the steps that are required for running the Jenkins pipeline.

Similar to groovy

2 syntaxes::

Scripted – traditional way of writing a jenkins pipeline as a code, it starts with word “node”

Declarative -- recent feature of the Jenkins pipeline which helps us to write the pipeline in a easier way, it strts with word “pipeline”

Pipeline is the main block and everything inside pipeline will be executed by Jenkins nodes/agent.

Agent: it defines that in which server pipeline executes

Stage: contains of unique tasks such as build, test, deploy etc

Step: tells what exactly needs to be done

Pipeline {

Agent any

stages {

stage(‘Hello) {

steps {

echo ‘Hello World’

}

}

}

}

Code analysis:

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

Detects vulnerabilities and functional errors.

OWASp vulnerabilities – open web application security vulns, an international org focused on improving security of software.

Term refers to security vulnerabilities identified and documented by OWASp as part of their top ten list.

Sonar token

squ\_b6b13fa1a3d719aa52ec0fd9b5e66d2a629f09af

DOCKER AGENT:

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

**Docker Slave Configuration**

Run the below command to Install Docker

sudo apt update

sudo apt install docker.io

**Grant Jenkins user and Ubuntu user permission to docker deamon.**

sudo su -

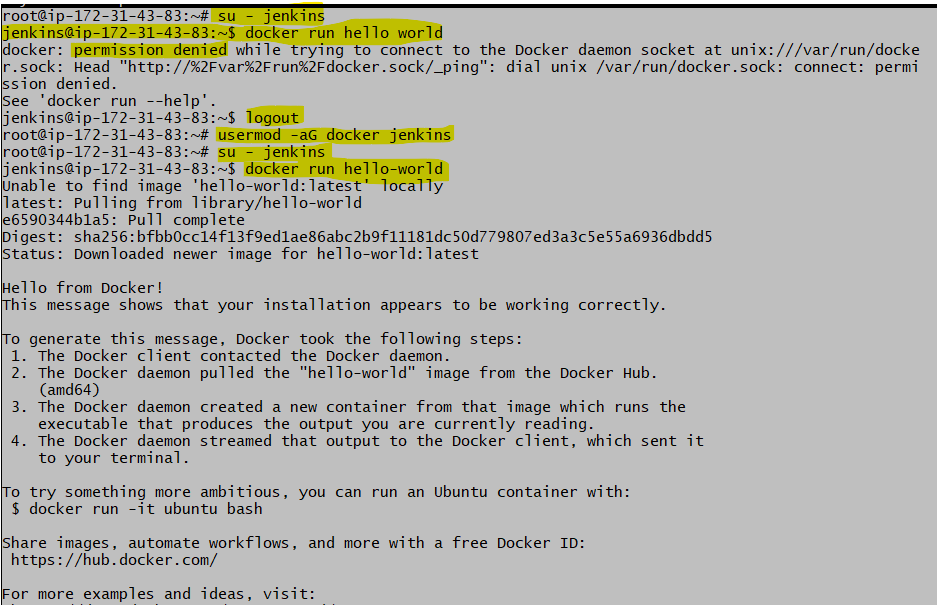
usermod -aG docker jenkins

usermod -aG docker ubuntu

systemctl restart docker

* To enter into docker use, su - jenkins

If you get permission error while running docker, grant access properl (usermod)

****

Once you are done with the above steps, it is better to restart Jenkins.

http://<ec2-instance-public-ip>:8080/restart

The docker agent configuration is now successful.

Example

Dockerfile for deploying java based web appln using Apache tomcat with multistage build

Build the Java appln

FROM maven:3.9.9-eclipse-temurin-21-jammy AS BUILD\_IMAGE

RUN git clone https://github.com/hkhcoder/vprofile-project.git

RUN cd vprofile-project && git checkout docker && mvn install

Deployment using Tomcat

FROM tomcat:10-jdk21

RUN rm -rf /usr/local/tomcat/webapps/\*

COPY --from=BUILD\_IMAGE vprofile-project/target/vprofile-v2.war /usr/local/tomcat/webapps/ROOT.war

Expose and start the appln

EXPOSE 8080

CMD ["catalina.sh", "run"]

Build the Docker img based on the dockerfile just created

docker build -t vprofile-app .

Start the container using using the following cmd

docker run -d -p 8080:8080 vprofile-app

**Access the Application:**

Open a web browser and go to http://localhost:8080 (or replace localhost with the server's IP address if running on a remote machine).

**Check Logs (Optional):**

To ensure the container is running smoothly or debug any issues, check the logs:

docker logs <container-id>

Replace <container-id> with the container's ID, which you can find using:

docker ps

**Stop the Container:**

If you need to stop the container:

docker stop <container-id>

IAM user -- For creating registry credentials, create IAM user with access keys

ECR -- ECR registry is docker registry from AWS and ACR registry is docker reg from Azure where we need to store docker images --

Plugin

Docker, docker pipeline,ecr

Jenkins

Store aws access keys (credentials)

Install docker engine in jenkins

Triggers:

Git webhook

Poll SCM

Scheduled jobs

Remote triggers

Build after other projects are built

Steps:

Create git repository on github

Ssh auth

Create jenkinsfile in git repo & commit

Create Jenkins job to access jenkinsfile from git repo

Test triggers

Error: key verification failed when accessing repository from Jenkins

Jenkins>>manage Jenkins>>security>>git host key verification config>>Accept fist connection and apply

Public key in github

Private key in jenkins

Cron jobs are scheduled tasks in Unix-like operating systems that run automatically at specified times or intervals. They are managed by the **cron** service and are useful for automating repetitive tasks like system backups, cleaning logs, or running scripts.

To set up a cron job, you use a **crontab** file (short for "cron table"), where you define the timing and the command to execute. Each line in the crontab follows this format:

\* \* \* \* \* command\_to\_be\_executed

- - - - -

| | | | |

| | | | +---- Day of the week (0 - 7, where 0 and 7 both mean Sunday)

| | | +------ Month (1 - 12)

| | +-------- Day of the month (1 - 31)

| +---------- Hour (0 - 23)

+------------ Minute (0 - 59)

For example, if you want a script to run at 3:30 PM every day, your entry would look like this:

30 15 \* \* \* /path/to/your/script.sh

Jenkins Master

For

Scheduling jobs

Assigning jobs to slaves for execution

Monitoring status of slaves

Displaying build restarts n logs

Jenkins slave

Worker nodes that execute build tasks assigned by master

Authentication and Authorization

Authentication is login – who can login

Authorization is privilege/permissions – what can user do once logged in