**Jenkins-POC**

**POC Overview**

1. Setting up Jenkins: Continuous Integration server setup.

2. Installing Docker: Containerization tool installation for environment consistency.

3. Setting up SonarQube: Code quality and security analysis.

4. Setting up EC2 for deployment

5. Pipeline and Deployments: Automating builds, tests, and deployments.

**Github repository link:**

[**https://github.com/rushikeshmj/hello-world**](https://github.com/rushikeshmj/hello-world)

**Launch Virtual Machine using AWS EC2**

Here is a detailed list of the basic requirements and setup for the EC2 instance i

have used for running Jenkins, including the specifics of the instance type, AMI,

and security groups.

EC2 Instance Requirements and Setup:

1. Instance Type

- Instance Type: `t2.large`

- vCPUs: 2

- Memory: 8 GB

- Network Performance: Moderate

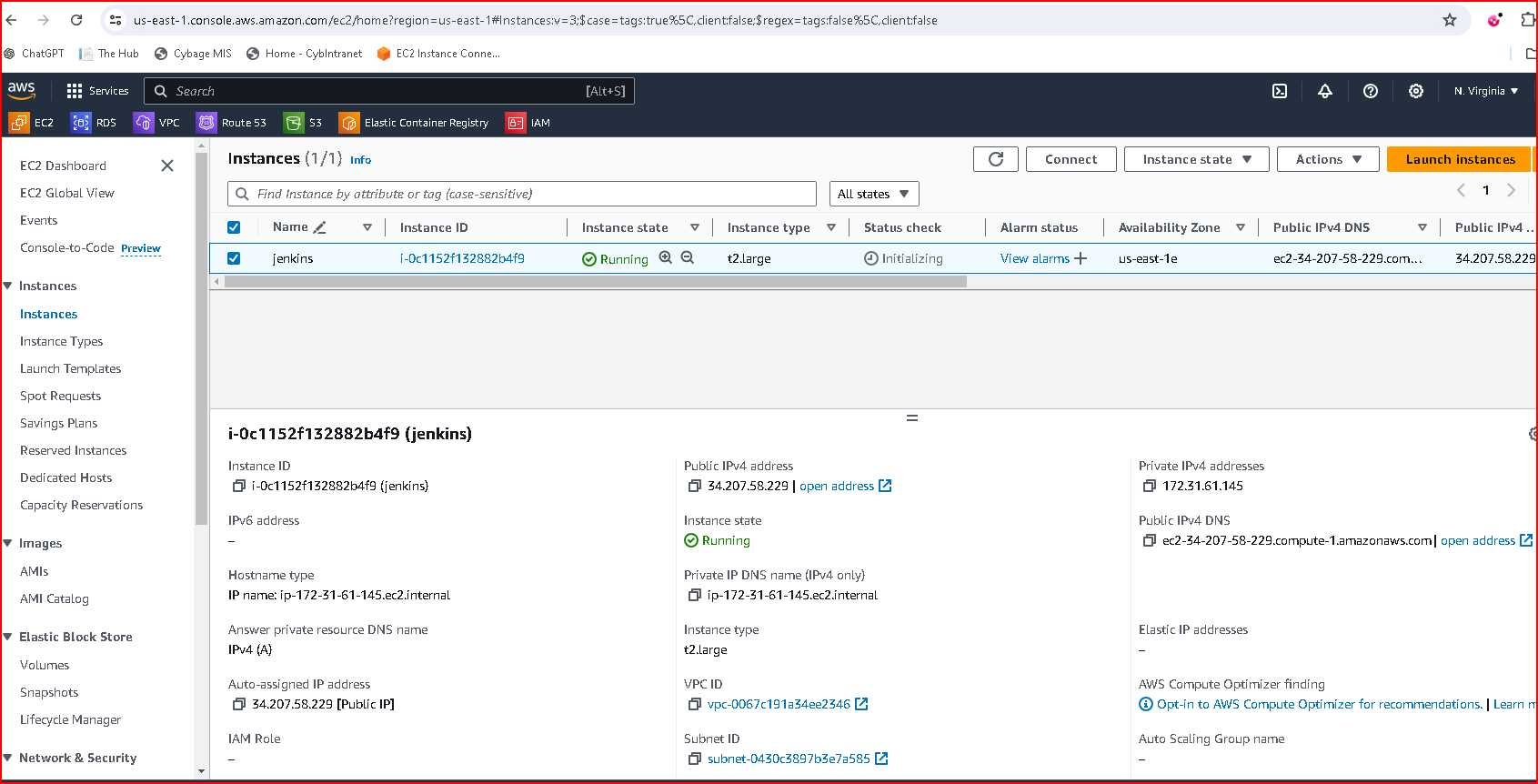
2. Amazon Machine Image (AMI)

- AMI: Ubuntu Server 20.04 LTS (Focal Fossa)

3. Security Groups

Security groups act as a virtual firewall for your instance to control inbound

and outbound traffic.



**After Launching your Virtual machine ,SSH into the Server.**

**Install below tools to EC2**

1. **Jenkins**

I am launching the container of Jenkins because 8080 port is not accessible on my machine

1. **Docker**

Sudo apt-get install docker.io

Run this command to give access to docker

sudo chmod 666 /var/run/docker.sock

1. **Trivy**

sudo apt-get install wget apt-transport-https gnupg lsb-release

wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | sudo

apt-key add -

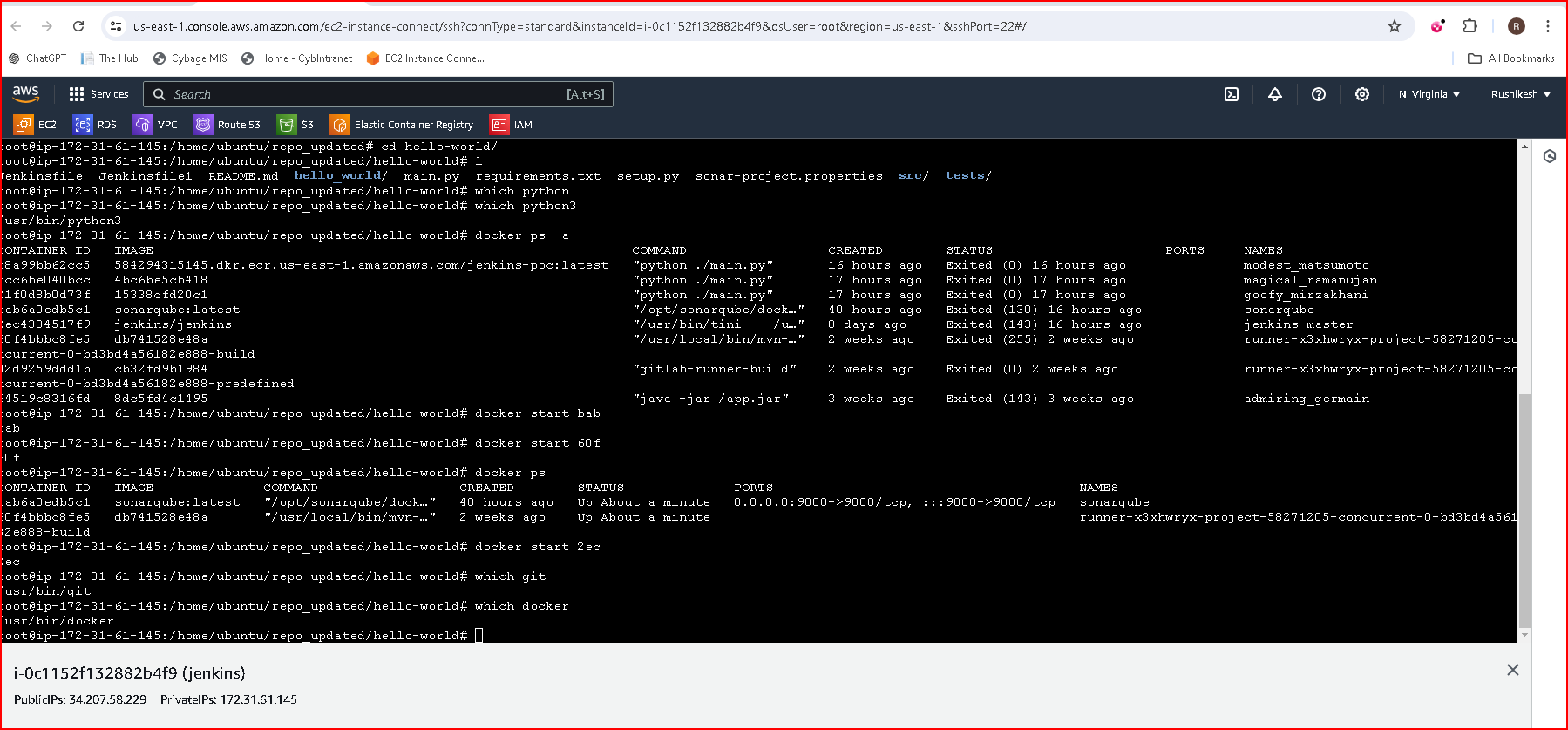
echo deb https://aquasecurity.github.io/trivy-repo/deb $(lsb\_release -sc) main

| sudo tee -a /etc/apt/sources.list.d/trivy.list

sudo apt-get update

sudo apt-get install trivy

1. **Git**
2. **Aws cli**

****

**Configure Jenkins**

Access Jenkins Dashboard:

Open a web browser and navigate to your Jenkins instance

Log in with your Jenkins credentials. (cat address provided on Jenkins)

Install Plugins:

-Go to Manage Jenkins > Manage Plugins.

-Click on the Available tab.

Search for and install the following plugins:

1. Docker: Enables Jenkins to use Docker containers.
2. Sonar Scanner: For Scanning Vulnerabilities.
3. Docker Pipeline: Allows Jenkins to use Docker containers in pipeline

jobs.

1. ECR
2. Cobertura
3. Code scanner plugin
4. Github
5. Pipeline stage view
6. Ssh
7. Suggested plugins

**Configuring SonarQube Scanner Plugin**

1. Manage Jenkins: Go to "Manage Jenkins".

2. Global Tool Configuration: Click on "Global Tool Configuration".

3. SonarQube Scanner:

• Scroll down to the "SonarQube Scanner" section.

• Click "Add SonarQube Scanner".

• Provide a name (e.g., Sonar scanner).

• Optionally, check "Install automatically" to let Jenkins handle the

installation.

• Save the configuration.

4. Manage Jenkins: Go back to "Manage Jenkins".

5. Configure System:

• Scroll down to the "SonarQube servers" section.

• Click "Add SonarQube".

• Provide a name for the server (e.g., SonarQube).

• Set the "Server URL" to the URL of your SonarQube instance.

• Add a "Server Authentication Token".

**Creating a Token on SonarQube**

1. Log in to SonarQube: Open your SonarQube instance in a web browser

and log in.

2. My Account: Click on your user profile at the top-right corner and select

"My Account".

3. Security: Navigate to the "Security" tab.

4. Generate Token: Under "Generate Tokens", provide a name for the

token (e.g., JenkinsToken).

5. Generate: Click on "Generate" and copy the token.

**Adding SonarQube Token to Jenkins**

1. Manage Jenkins: Go to "Manage Jenkins".

2. Configure System: Scroll to the "SonarQube servers" section.

3. Add Token:

• Under the "Server Authentication Token" section, click "Add" next

to "Credentials".

• Select "Jenkins" and then "Secret text".

• Paste the token you copied from SonarQube.

• Provide an ID (e.g., sonarqube-token).

• Save the credentials.

• Select the newly added token from the dropdown list.

**Configuring Docker Plugin**

1. Manage Jenkins: Go to "Manage Jenkins".

2. Global Tool Configuration: Click on "Global Tool Configuration".

3. Docker:

• Scroll down to the "Docker" section.

• Click "Add Docker Tool".

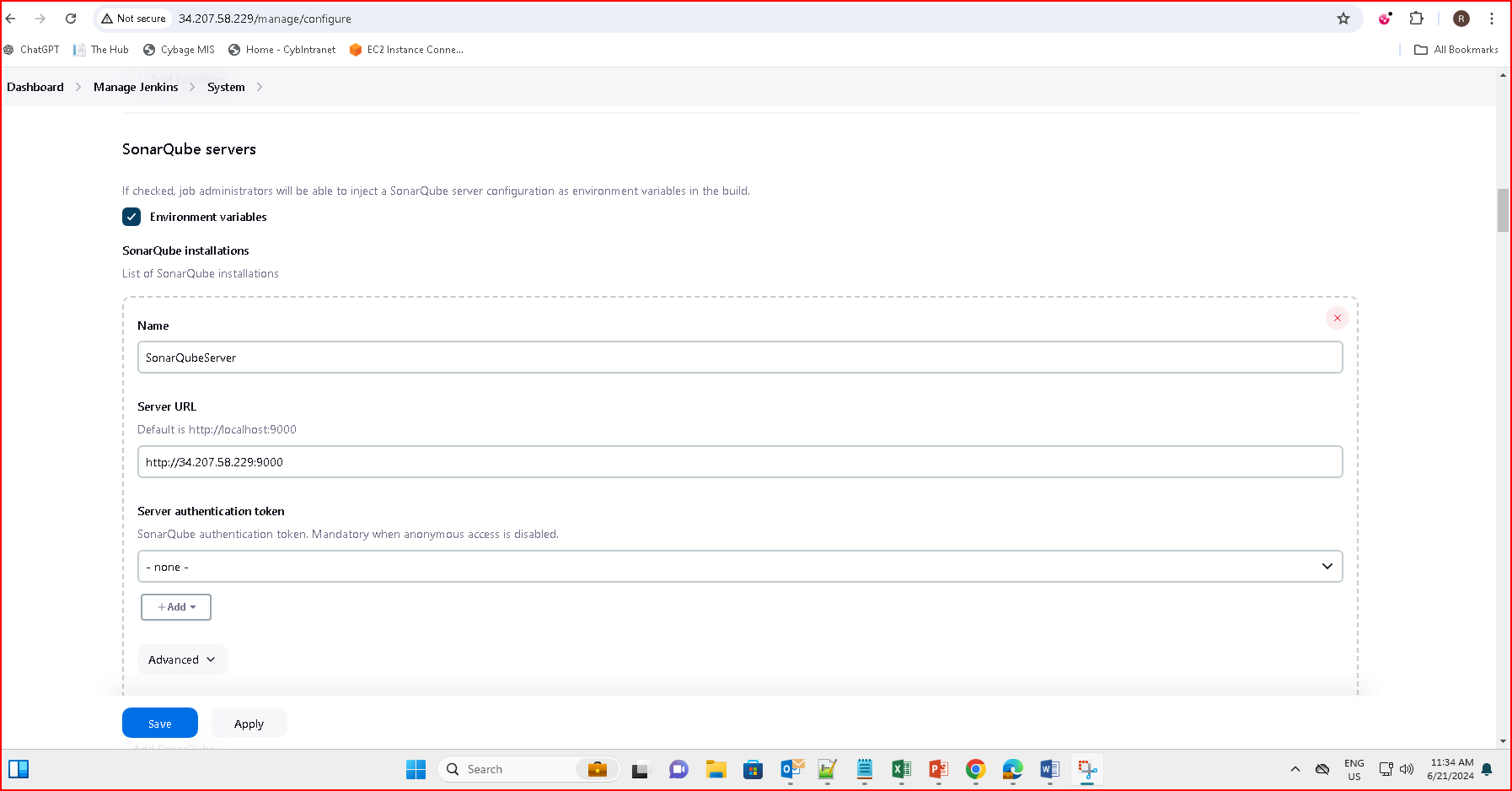
• Provide a name (e.g., docker).

• Optionally, check "Install automatically" to let Jenkins handle the

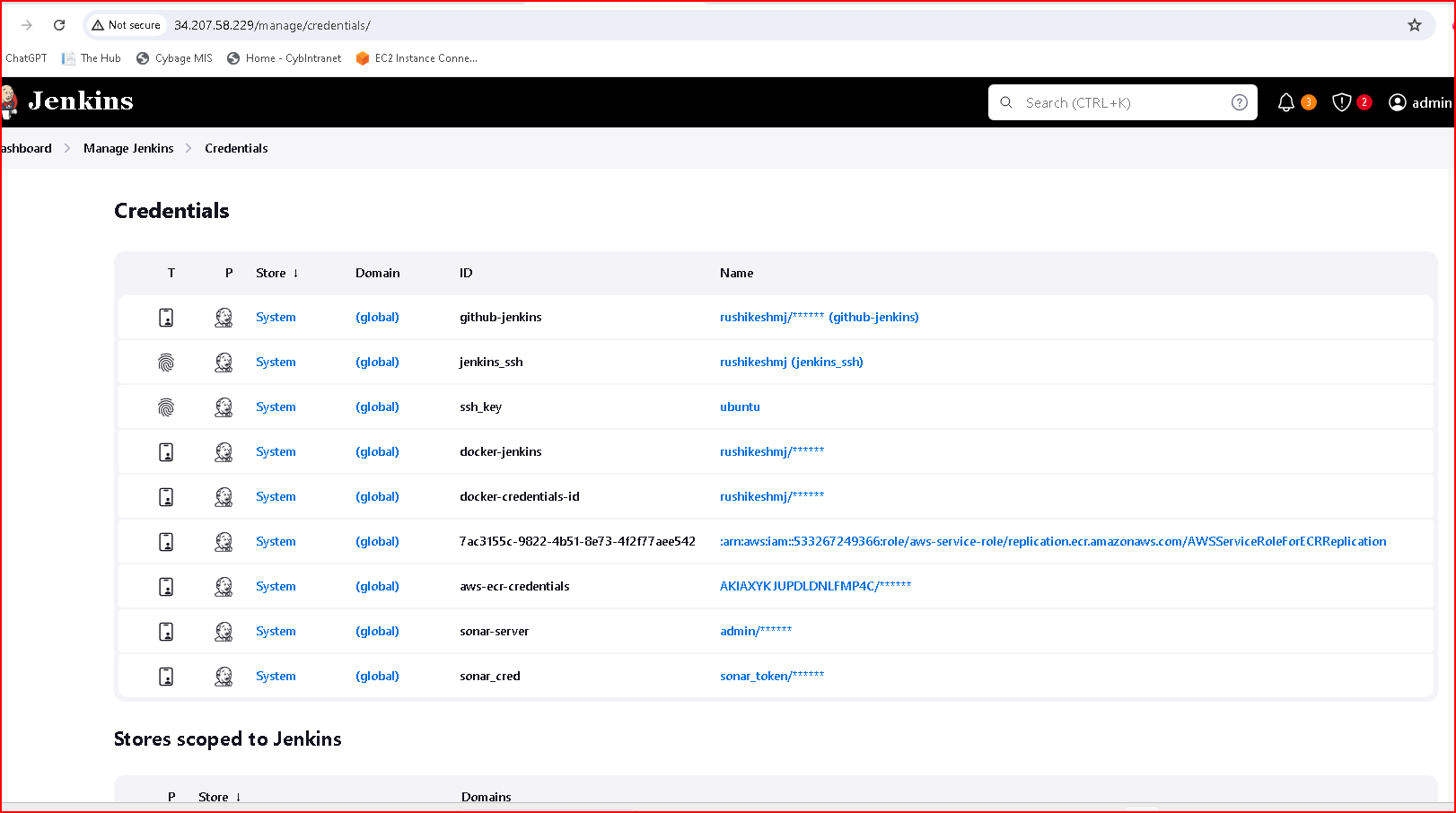
installation.

• Save the configuration.

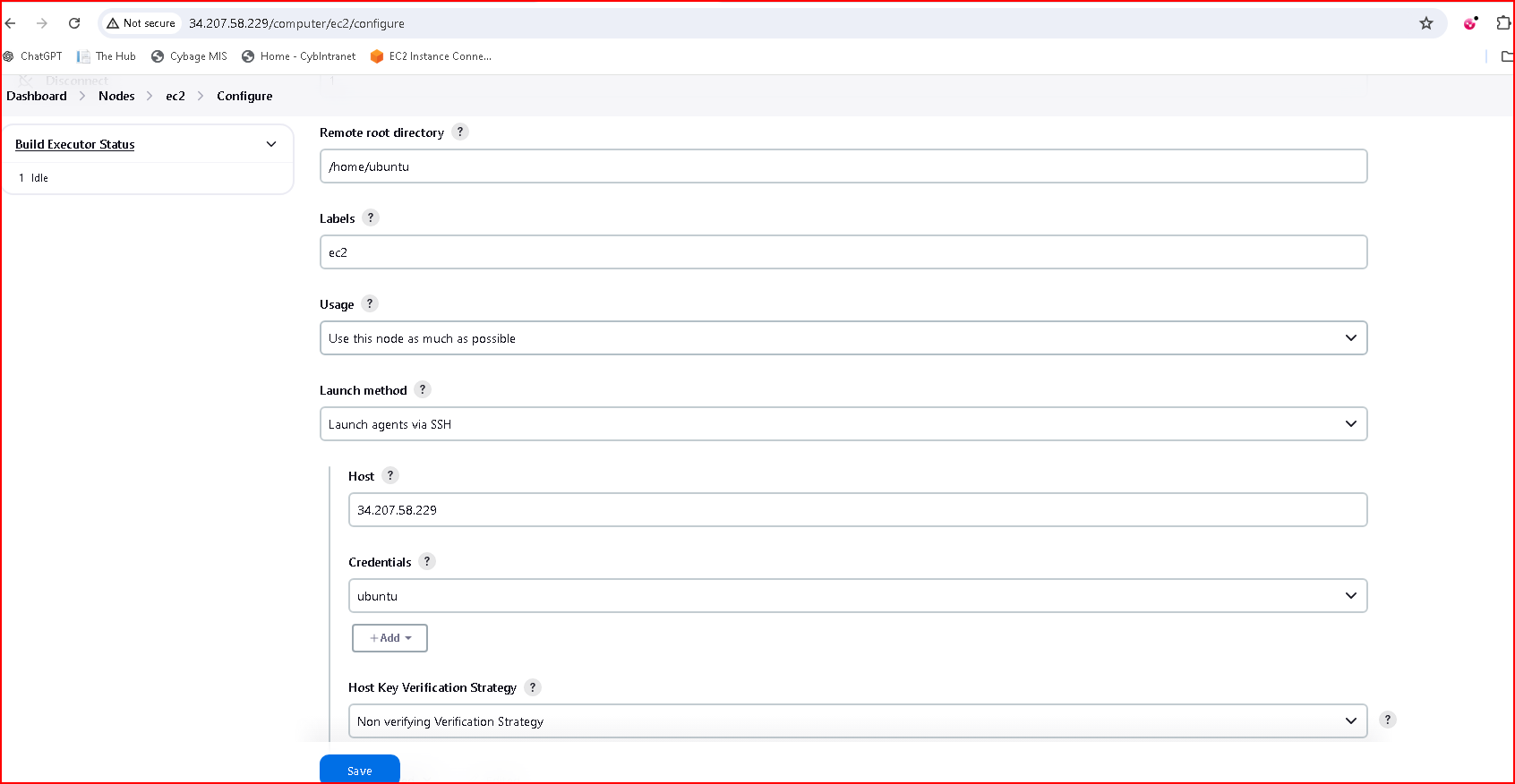
**Configure system with their setting and path**

****

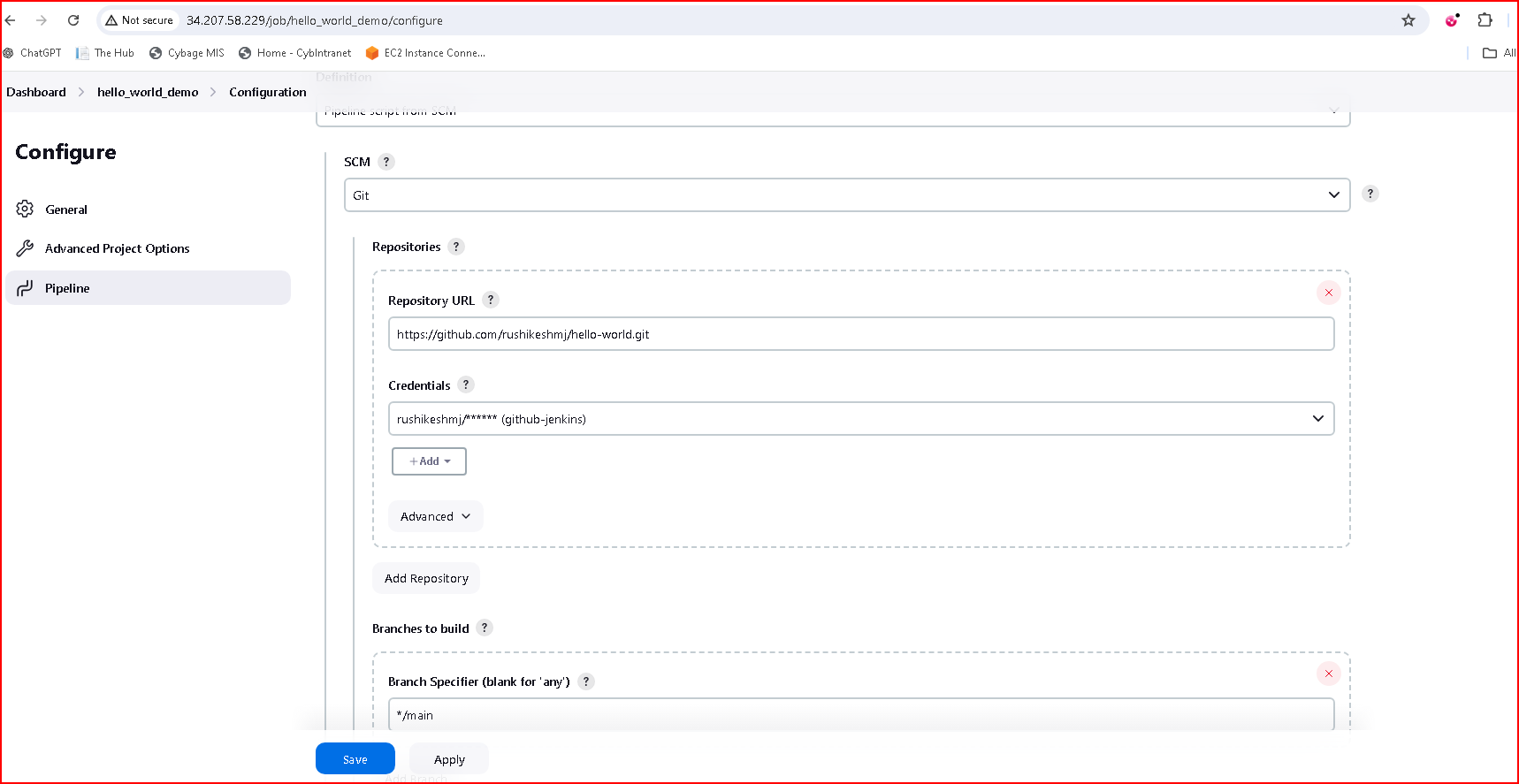
**Save the credentials**

****

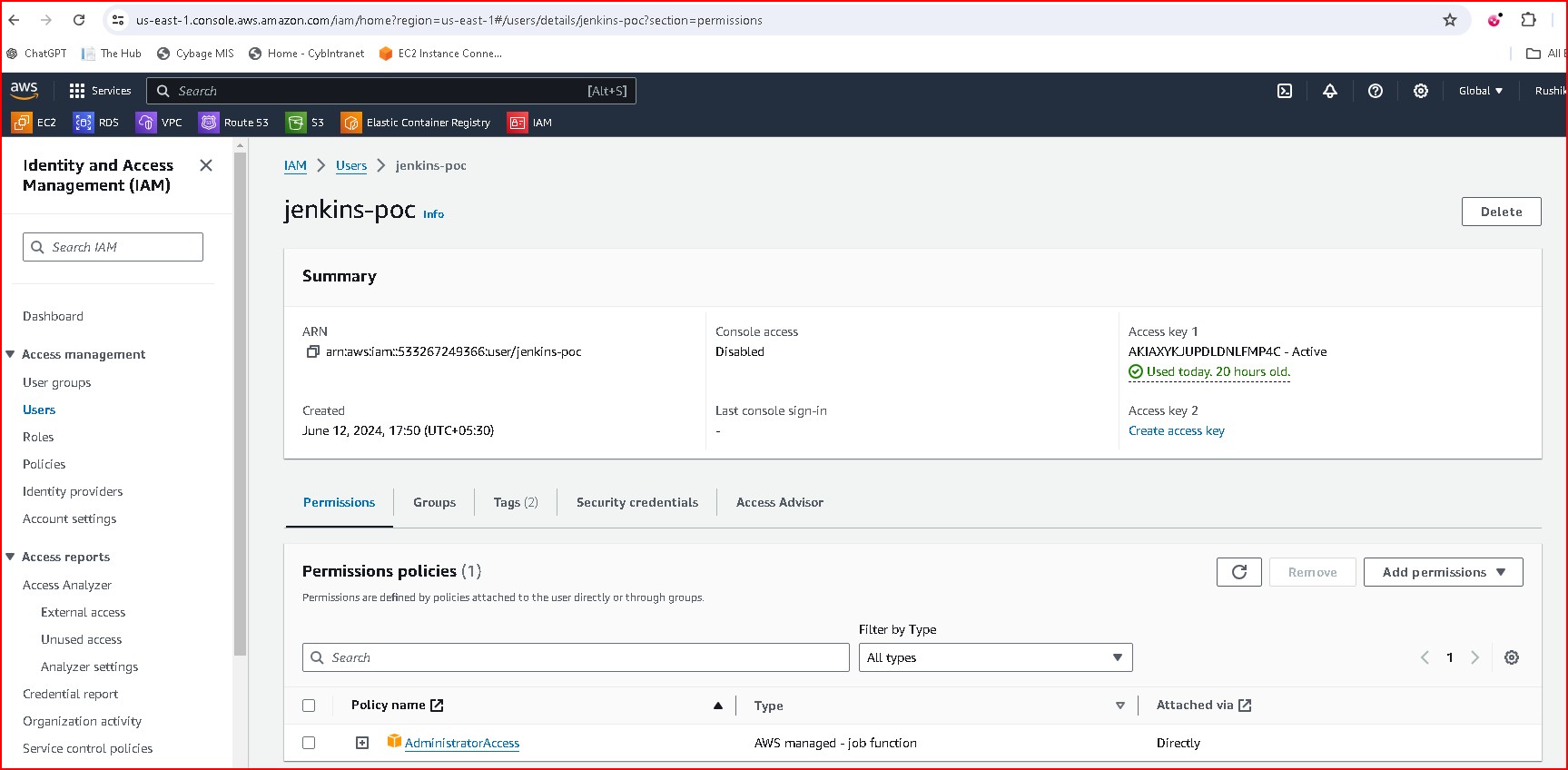
**Create Node and configure**



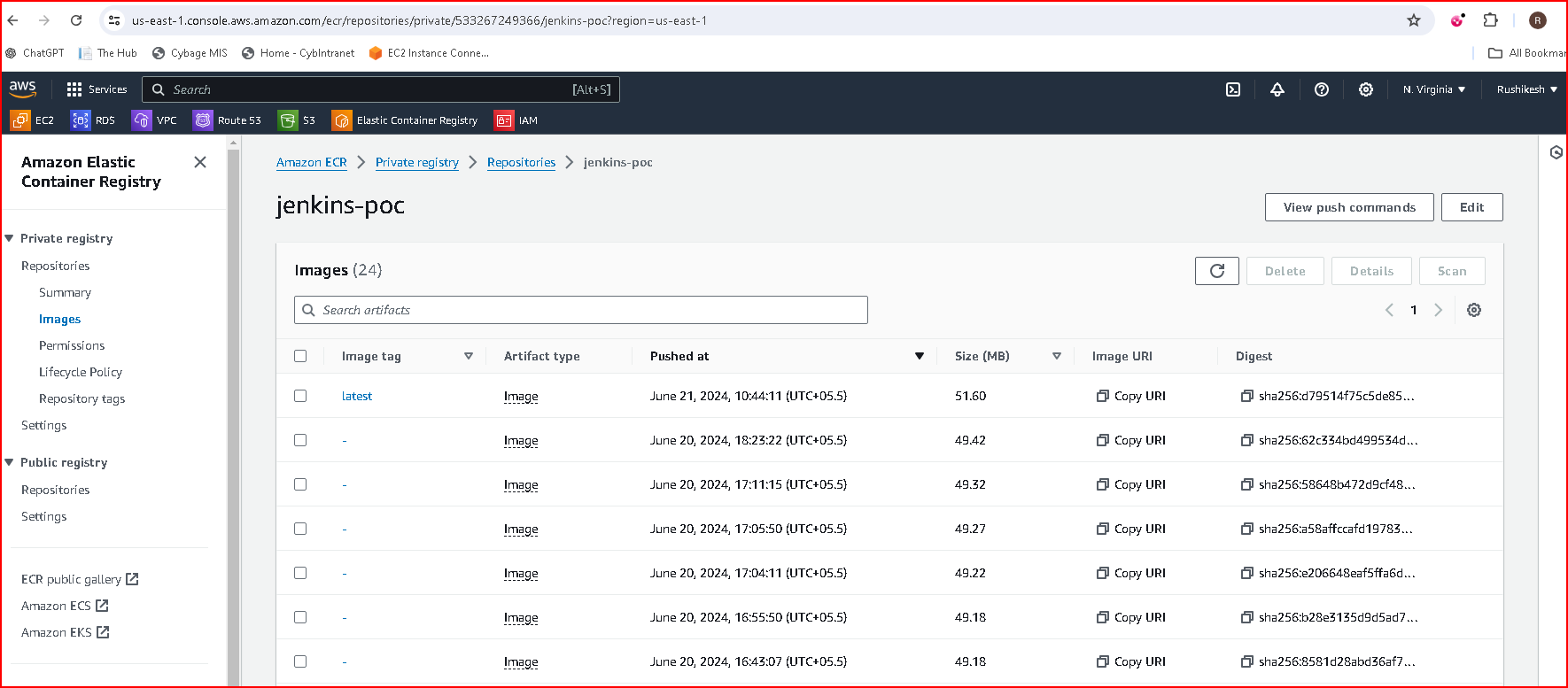
**Create Job and it to github repo and cred**

****

**Create IAM user**

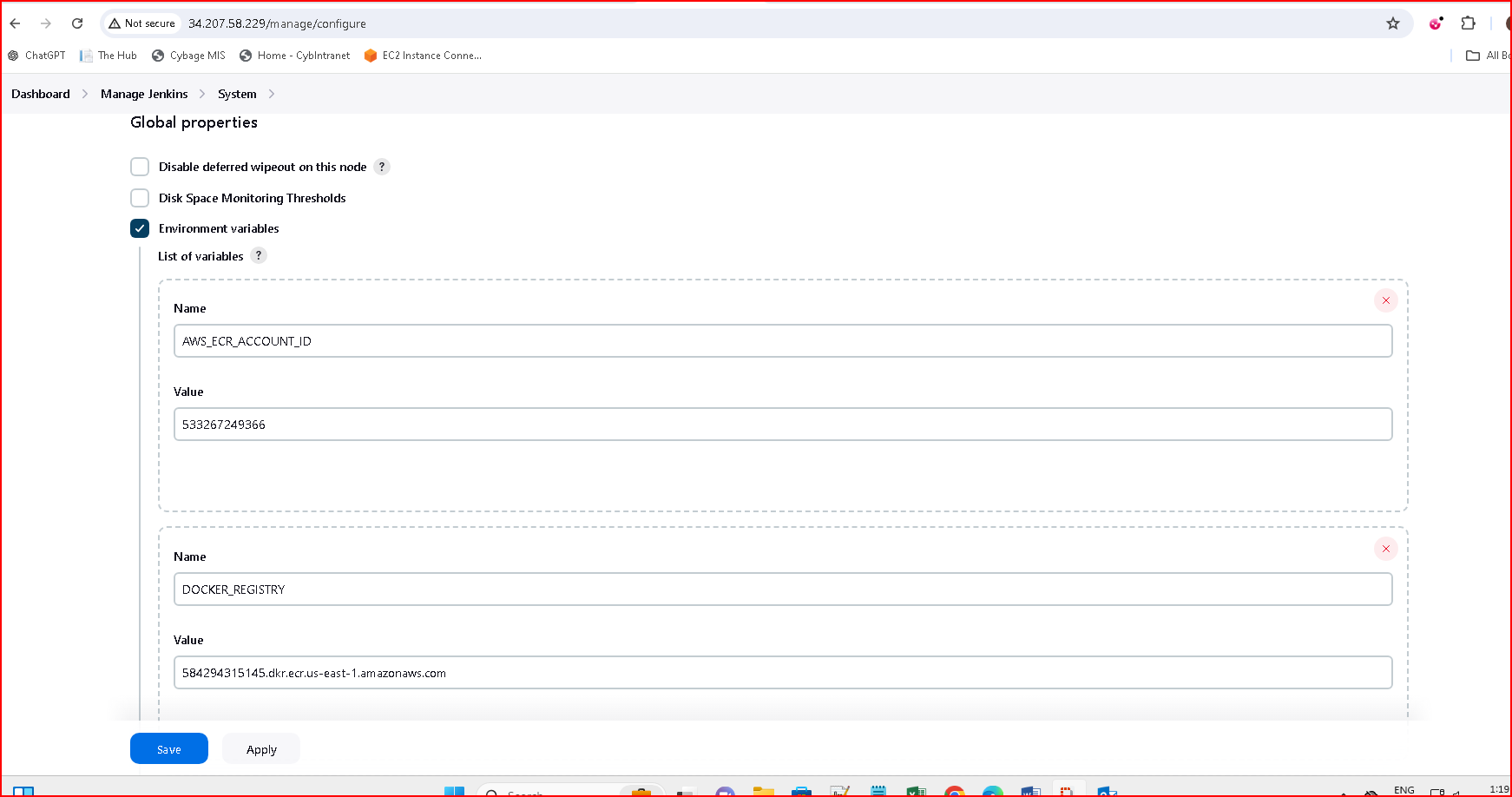
****

**Create ECR repo**

****

**ADD ENV variables**

Manage Jenkins 🡪 system 🡪 Global properties 🡪 ENV varibales



**Create Pipeline**

pipeline {

agent {

label 'ec2'

}

environment {

SONARQUBE\_SERVER = "http://$PUBLIC\_IP:9000/"

AWS\_ECR\_REGION = "us-east-1"

AWS\_ECR\_REPOSITORY = "jenkins-poc"

DOCKER\_IMAGE\_NAME = "hello-world-app"

}

stages {

stage('Checkout') {

steps {

retry(3) {

checkout([$class: 'GitSCM', branches: [[name: '\*/main']], userRemoteConfigs: [[url: 'https://github.com/shekharbo/hello-world.git']]])

}

}

}

stage('Unit Test') {

steps {

sh 'docker --version'

sh 'docker pull python:3.9'

sh 'pip install -r requirements.txt'

sh 'python3 -m venv ~/myenv'

sh """

set +x

. /home/ubuntu/myenv/bin/activate

"""

sh """

/home/ubuntu/.local/bin/pytest /home/ubuntu/workspace/hello\_world\_demo/tests/test\_main.py

"""

}

}

stage('Code Coverage') {

steps {

sh 'pip install coverage'

sh """

/home/ubuntu/.local/bin/coverage run -m pytest /home/ubuntu/workspace/hello\_world\_demo/tests/test\_main.py

"""

sh '/home/ubuntu/.local/bin/coverage report'

sh '/home/ubuntu/.local/bin/coverage xml -o coverage.xml'

cobertura coberturaReportFile: 'coverage.xml'

}

}

stage('SCA and SonarQube') {

steps {

withSonarQubeEnv('SonarQubeServer') {

script {

def scannerHome = tool 'SonarQubeScanner'

if (scannerHome) {

sh "/home/hello-world-demo-python/hello-world/sonar-scanner-5.0.1.3006-linux/bin/sonar-scanner \

-Dsonar.projectKey=hello-world \

-Dsonar.sources=src \

-Dsonar.host.url=${SONARQUBE\_SERVER} \

-Dsonar.login=${SONARQUBE\_LOGIN\_TOKEN}"

} else {

error "SonarQube Scanner not configured."

}

}

}

}

}

stage('Build and tag image using Docker') {

steps {

script {

dir('/home/ubuntu/hello-world-demo-python/hello-world') {

sh 'pwd'

sh 'ls -l Dockerfile'

sh 'docker build -t hello-world-app .'

sh "docker tag hello-world-app ${AWS\_ECR\_ACCOUNT\_ID}.dkr.ecr.${AWS\_ECR\_REGION}.amazonaws.com/${AWS\_ECR\_REPOSITORY}:latest"

withCredentials([usernamePassword(credentialsId: 'aws-ecr-credentials', usernameVariable: 'AWS\_ACCESS\_KEY\_ID', passwordVariable: 'AWS\_SECRET\_ACCESS\_KEY')]) {

sh "aws ecr get-login-password --region ${AWS\_ECR\_REGION} | docker login --username AWS --password-stdin ${AWS\_ECR\_ACCOUNT\_ID}.dkr.ecr.${AWS\_ECR\_REGION}.amazonaws.com"

sh "docker push ${AWS\_ECR\_ACCOUNT\_ID}.dkr.ecr.${AWS\_ECR\_REGION}.amazonaws.com/${AWS\_ECR\_REPOSITORY}:latest"

}

}

}

}

}

stage('Image scan using trivy') {

steps {

sh "trivy image ${AWS\_ECR\_ACCOUNT\_ID}.dkr.ecr.${AWS\_ECR\_REGION}.amazonaws.com/${AWS\_ECR\_REPOSITORY}:latest"

}

}

stage('Deploy to EC2') {

steps {

sshagent(['ssh\_key']) {

sh "ssh -o StrictHostKeyChecking=no ubuntu@$PUBLIC\_IP aws ecr get-login-password --region ${AWS\_ECR\_REGION} | docker login --username AWS --password-stdin ${DOCKER\_REGISTRY}"

sh "ssh -o StrictHostKeyChecking=no ubuntu@$PUBLIC\_IP docker pull ${AWS\_ECR\_ACCOUNT\_ID}.dkr.ecr.${AWS\_ECR\_REGION}.amazonaws.com/${AWS\_ECR\_REPOSITORY}:latest"

sh "ssh -o StrictHostKeyChecking=no ubuntu@$PUBLIC\_IP docker run -d -p 8081:8080 ${AWS\_ECR\_ACCOUNT\_ID}.dkr.ecr.${AWS\_ECR\_REGION}.amazonaws.com/${AWS\_ECR\_REPOSITORY}:latest"

}

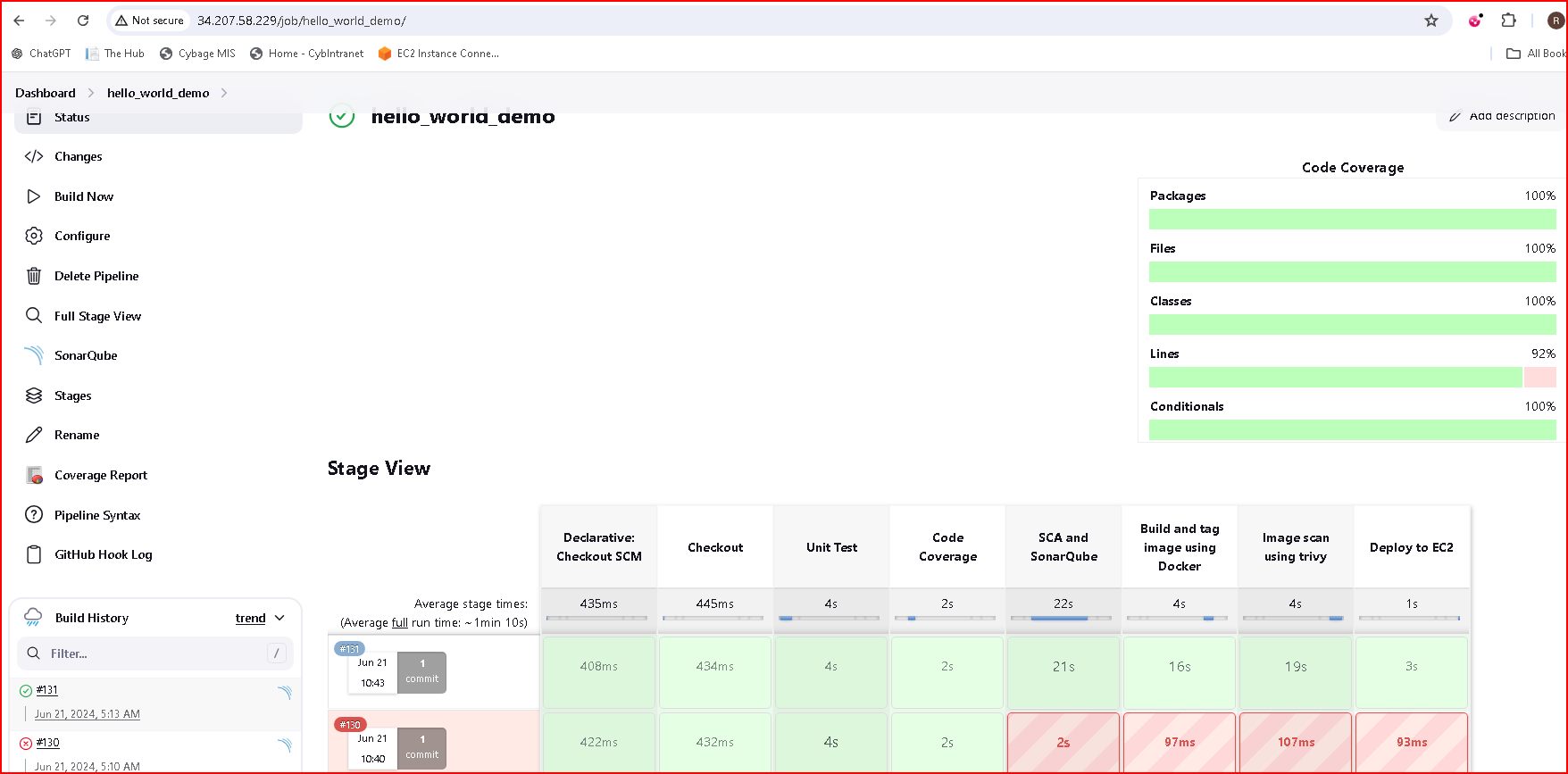
}

}

}

}

**Successful Deployment**

****