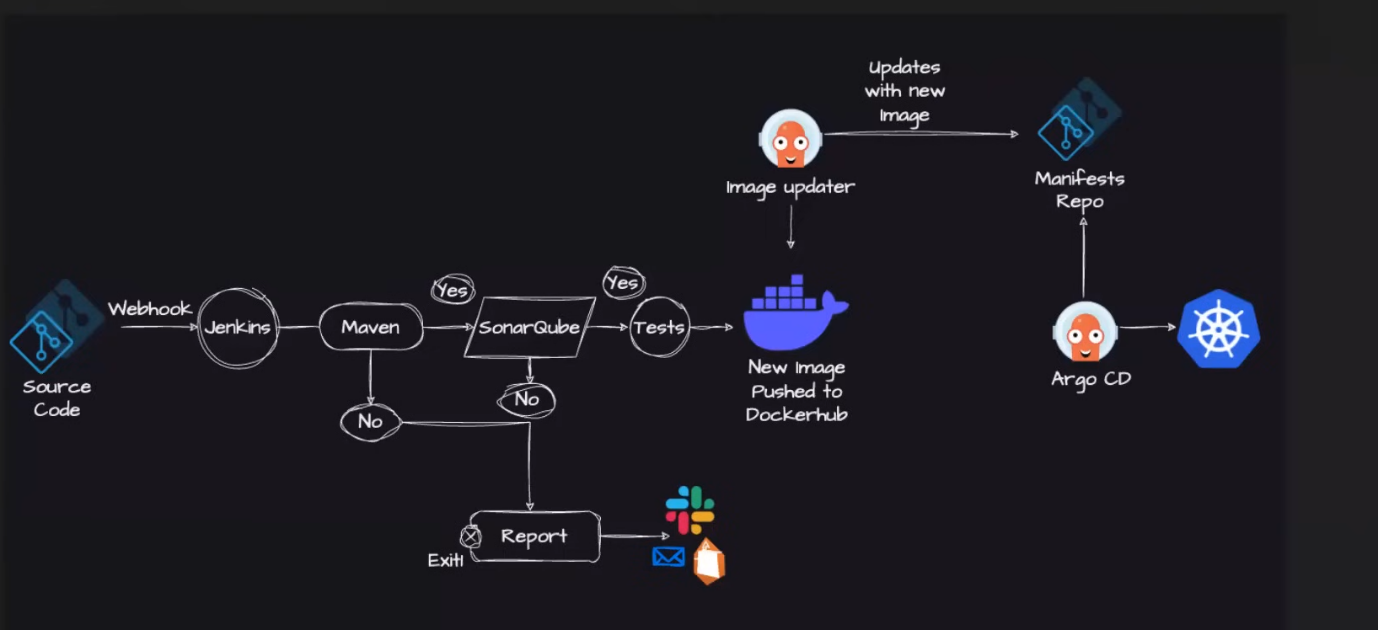
**Ultimate CICD project Using Maven, SonarQube, Argo CD, Helm and K8s**



**Architecture diagram for the project**

**Steps: -**

Jenkins Pipeline for Java based application using Maven, SonarQube, Argo CD, Helm and Kubernetes

Here are the step-by-step details to set up an end-to-end Jenkins pipeline for a Java application using SonarQube, Argo CD, Helm, and Kubernetes:

Prerequisites:

- Java application code hosted on a Git repository

- Jenkins server

- Kubernetes cluster

- Helm package manager

- Argo CD

Steps:

1. Install the necessary Jenkins plugins:

1.1 Git plugin

1.2 Maven Integration plugin

1.3 Pipeline plugin

1.4 Kubernetes Continuous Deploy plugin

2. Create a new Jenkins pipeline:

2.1 In Jenkins, create a new pipeline job and configure it with the Git repository URL for the Java application.

2.2 Add a Jenkinsfile to the Git repository to define the pipeline stages.

3. Define the pipeline stages:

Stage 1: Checkout the source code from Git.

Stage 2: Build the Java application using Maven.

Stage 3: Run unit tests using JUnit and Mockito.

Stage 4: Run SonarQube analysis to check the code quality.

Stage 5: Package the application into a JAR file.

Stage 6: Deploy the application to a test environment using Helm.

Stage 7: Run user acceptance tests on the deployed application.

Stage 8: Promote the application to a production environment using Argo CD.

4. Configure Jenkins pipeline stages:

Stage 1: Use the Git plugin to check out the source code from the Git repository.

Stage 2: Use the Maven Integration plugin to build the Java application.

Stage 3: Use the JUnit and Mockito plugins to run unit tests.

Stage 4: Use the SonarQube plugin to analyze the code quality of the Java application.

Stage 5: Use the Maven Integration plugin to package the application into a JAR file.

Stage 6: Use the Kubernetes Continuous Deploy plugin to deploy the application to a test

environment using Helm.

Stage 7: Use a testing framework like Selenium to run user acceptance tests on the deployed application.

Stage 8: Use Argo CD to promote the application to a production environment.

5. Set up Argo CD:

Install Argo CD on the Kubernetes cluster.

Set up a Git repository for Argo CD to track the changes in the Helm charts and Kubernetes manifests.

Create a Helm chart for the Java application that includes the Kubernetes manifests and Helm values.

Add the Helm chart to the Git repository that Argo CD is tracking.

6. Configure Jenkins pipeline to integrate with Argo CD:

6.1 Add the Argo CD API token to Jenkins credentials.

6.2 Update the Jenkins pipeline to include the Argo CD deployment stage.

7. Run the Jenkins pipeline:

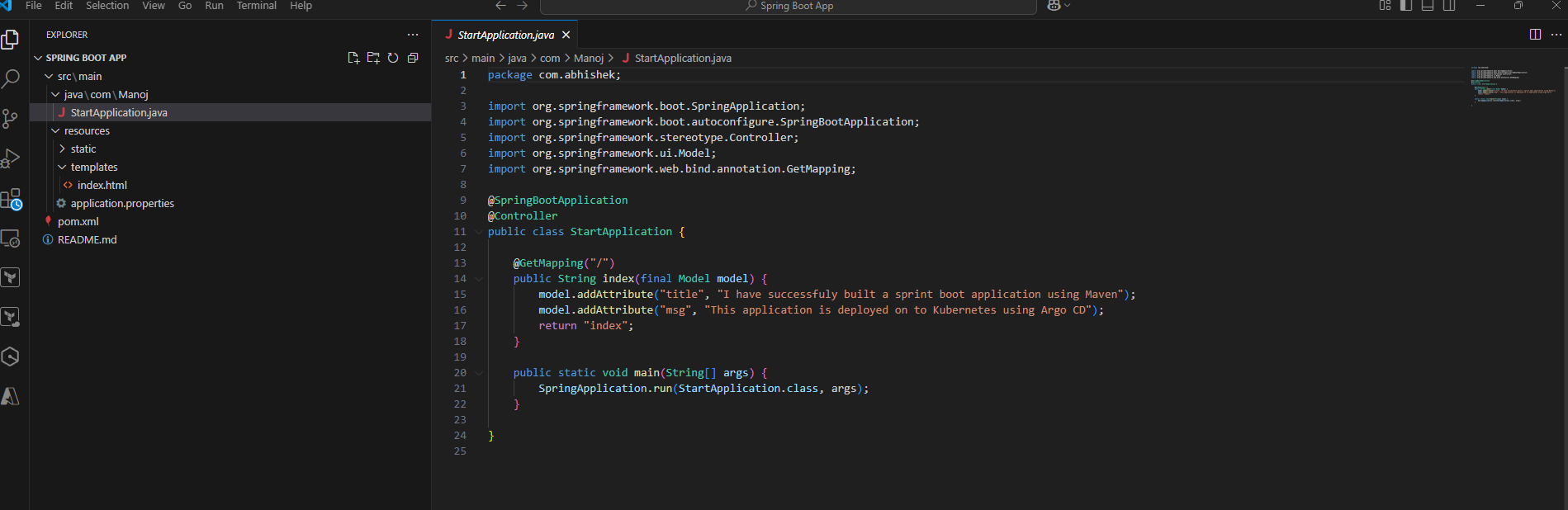
7.1 Trigger the Jenkins pipeline to start the CI/CD process for the Java application.

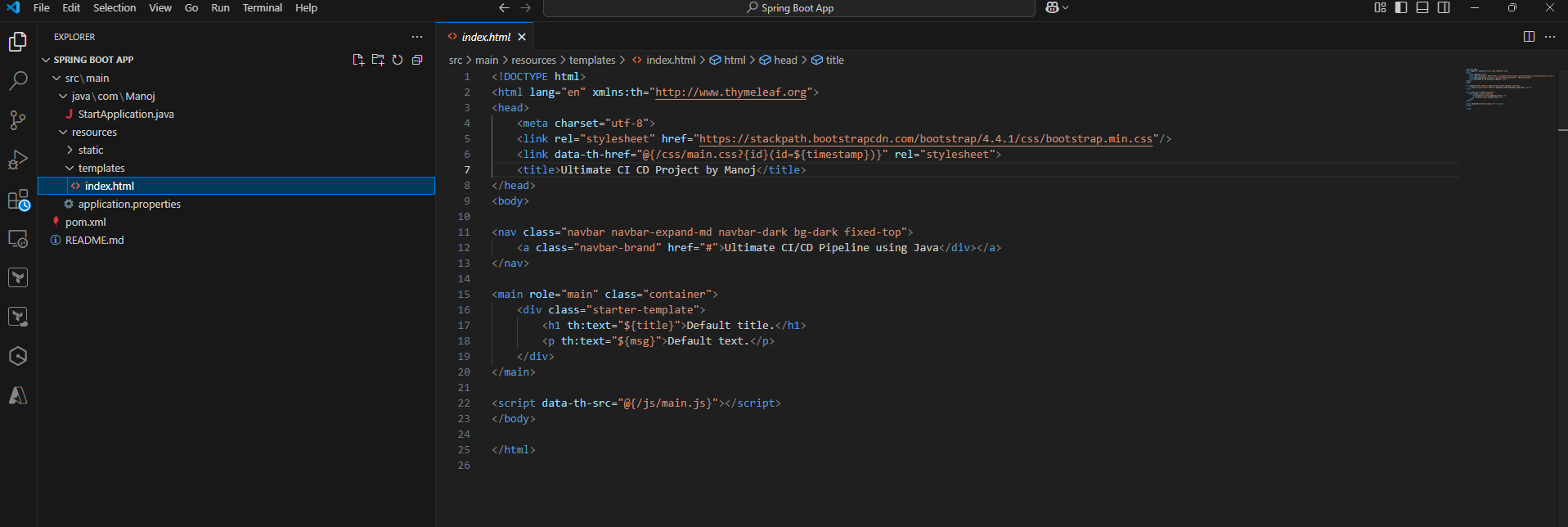
7.2 Monitor the pipeline stages and fix any issues that arise.

This end-to-end Jenkins pipeline will automate the entire CI/CD process for a Java application, from code checkout to production deployment, using popular tools like SonarQube, Argo CD, Helm, and Kubernetes.

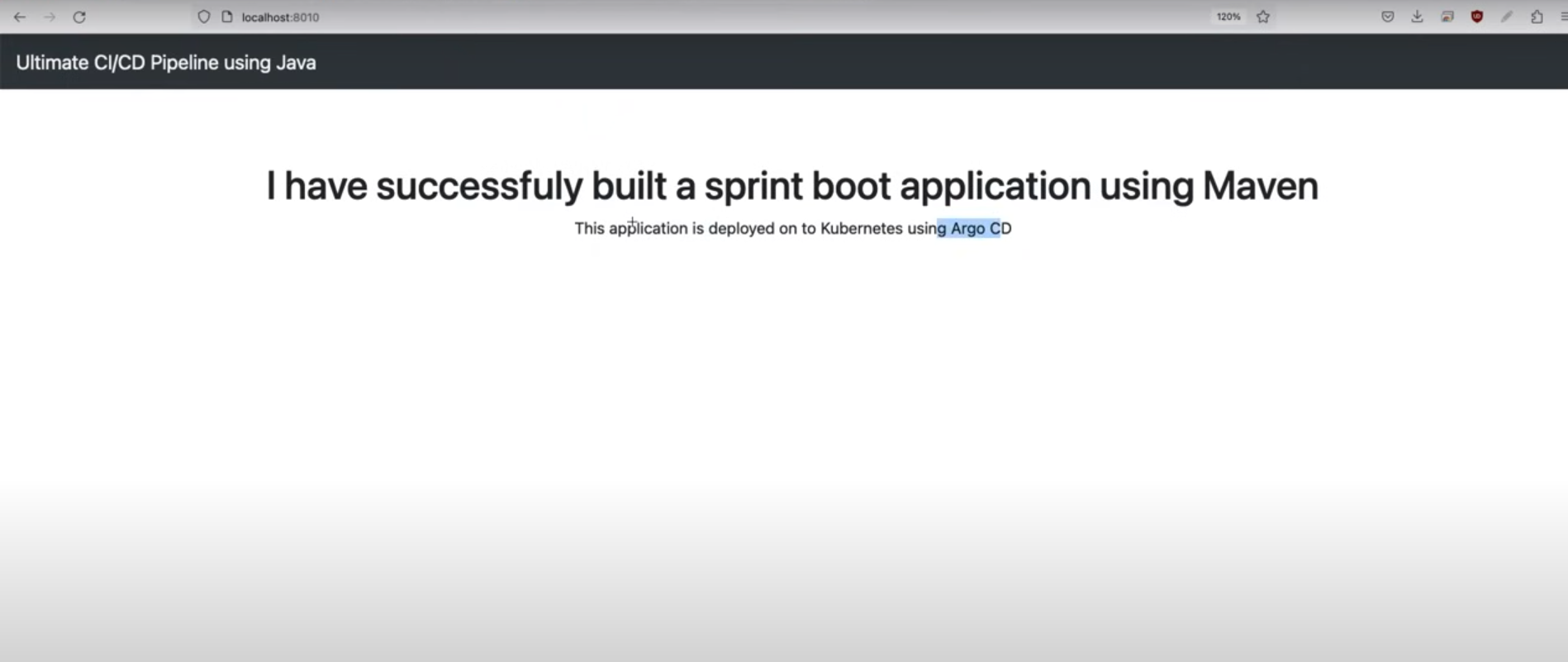
**In detail steps: -**

1. Create a sample Java Spring boot application.



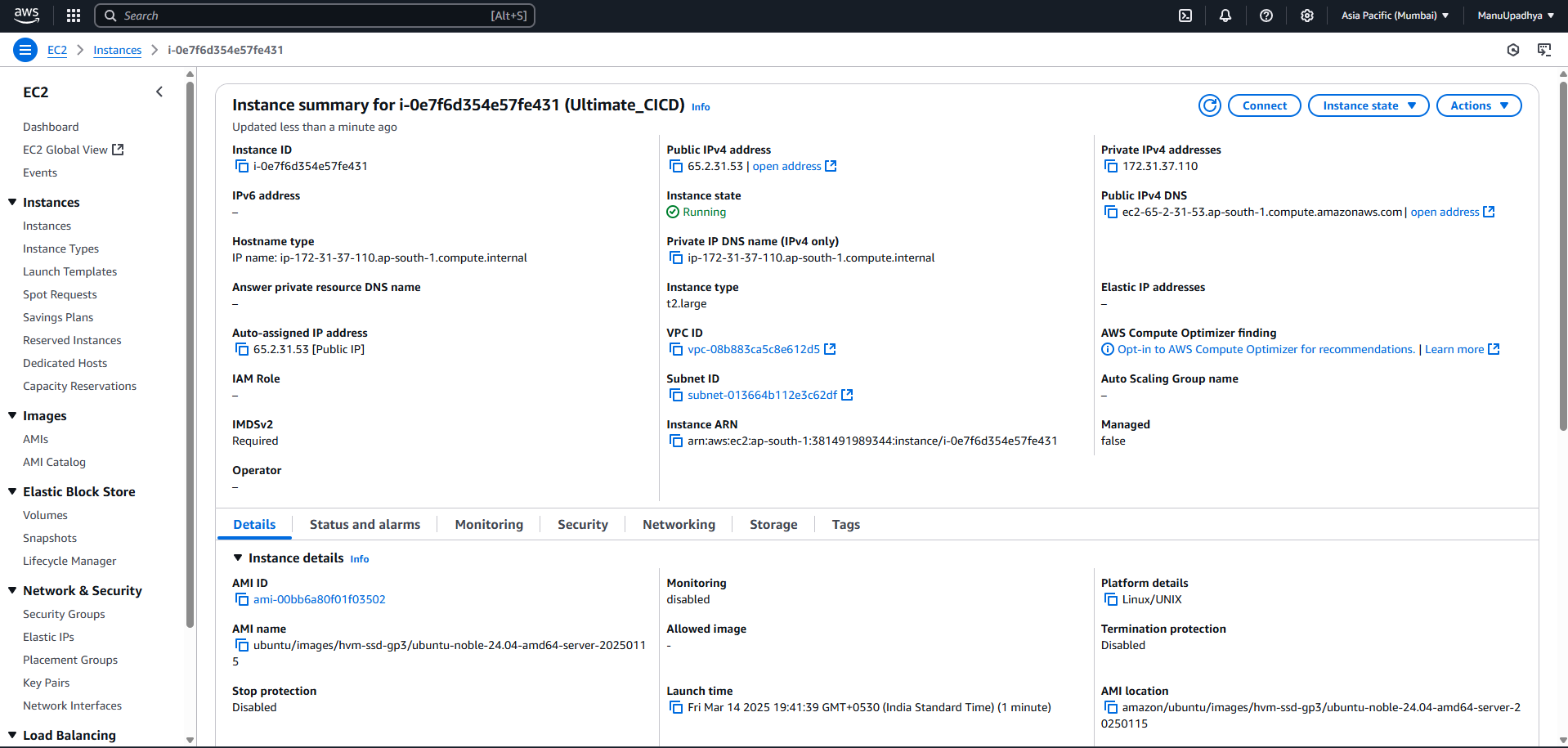


Push the code to github.



Static Application will look like above.

1. Create an EC2 t2.Large 8GB SSD instance.



1. Install Jenkins on the created ec2 instance.

First install Java JDK 11 using below command: -

sudo apt update

sudo apt install openjdk-17-jre

Check whether the java is installed or not using the command : -

java -version

Install Jenkins using below command: -

**curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \**

**/usr/share/keyrings/jenkins-keyring.asc > /dev/null**

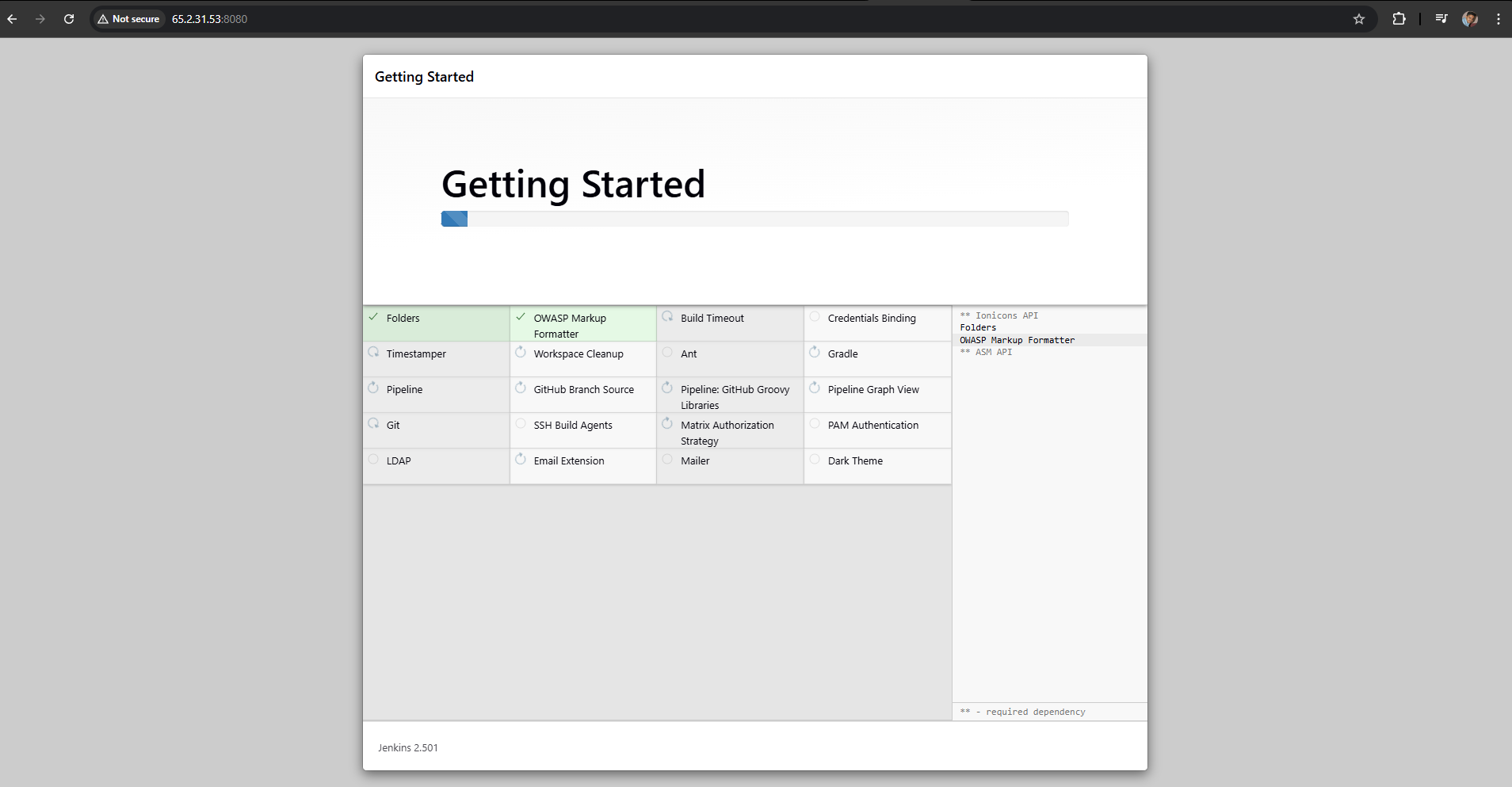
**echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \**

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

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

**sudo apt-get update**

**sudo apt-get install jenkins**



Log into Jenkins and and install plugins.

Note: - if you’re unable to access the Jenkins in the chrome check the security group of ex2 instance, whether port 8080 is allowed or not.

