**Industry Grade Project 2 : XYZ Technology**

**Aim of this project**

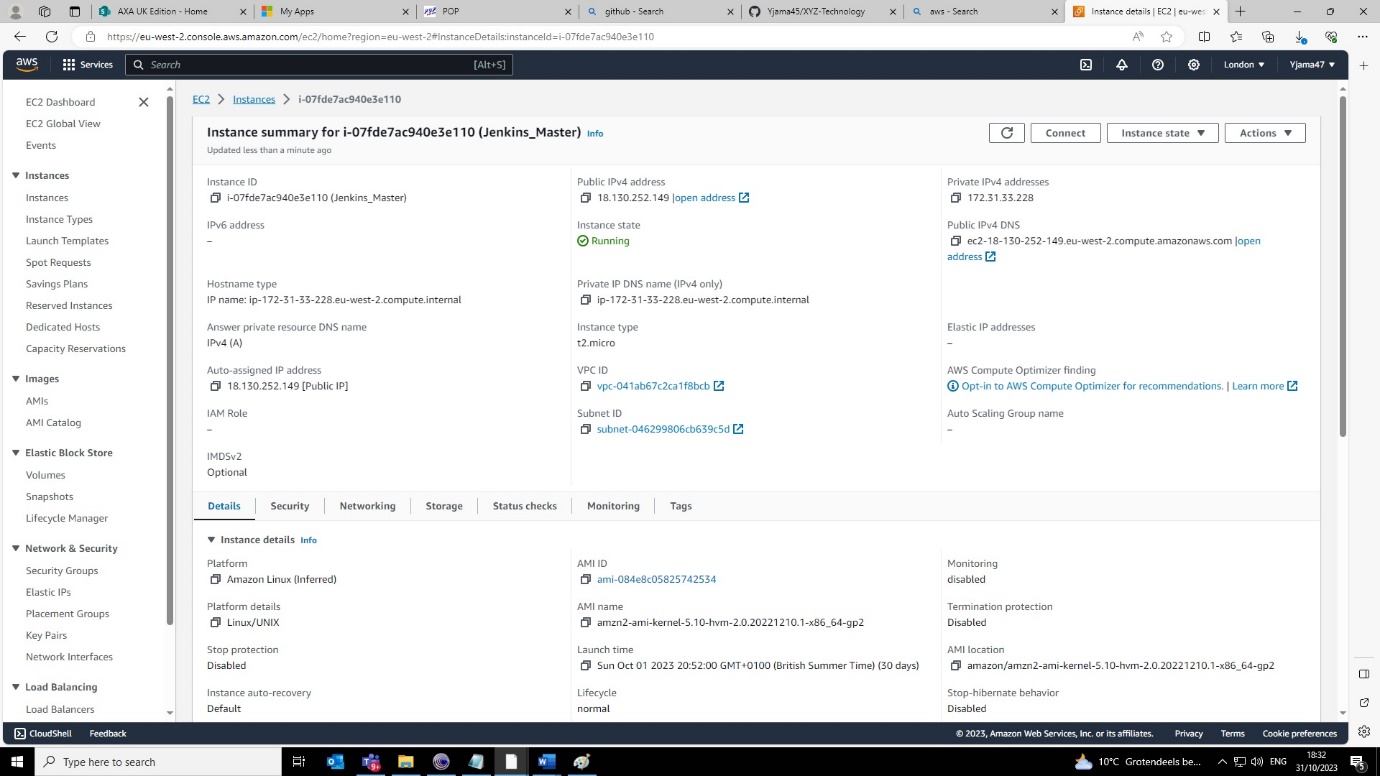
* Develop a continuous integration pipeline in Jenkins to compile, test and package the code
* CICD pipeline to resolve business challenges by XYZtechnologies.
* Real-time understanding and hands-on with Git, Jenkins, Ansible, Docker, Kubernetes, and AWS Devops services so I will demonstrate how to integrate Jenkins with all these tools achieve the above aims.

**Task 1:**

* Create a GitHub repository and push the folder from local repository to remote.



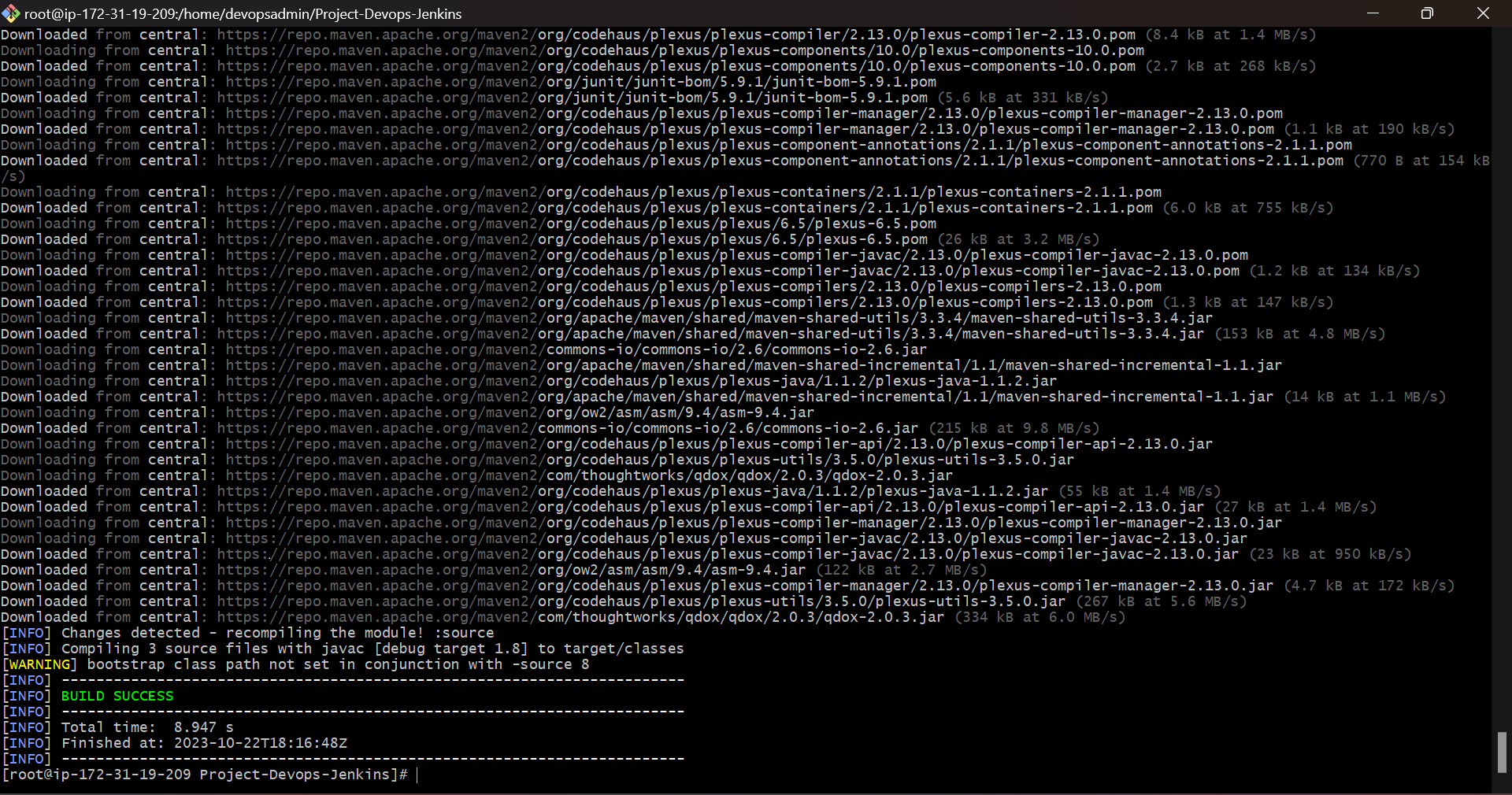
* Create a EC2 instance to create my CI/CD pipeline on.

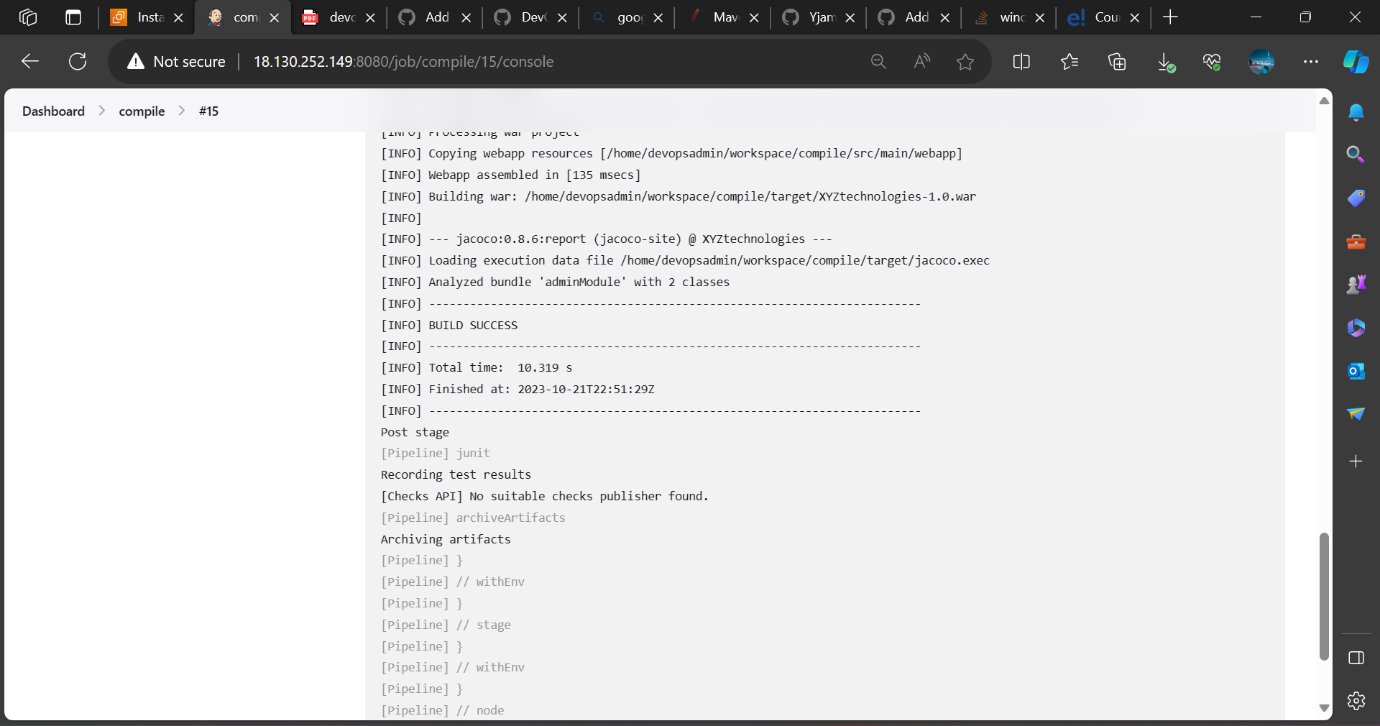


***Task 2:***

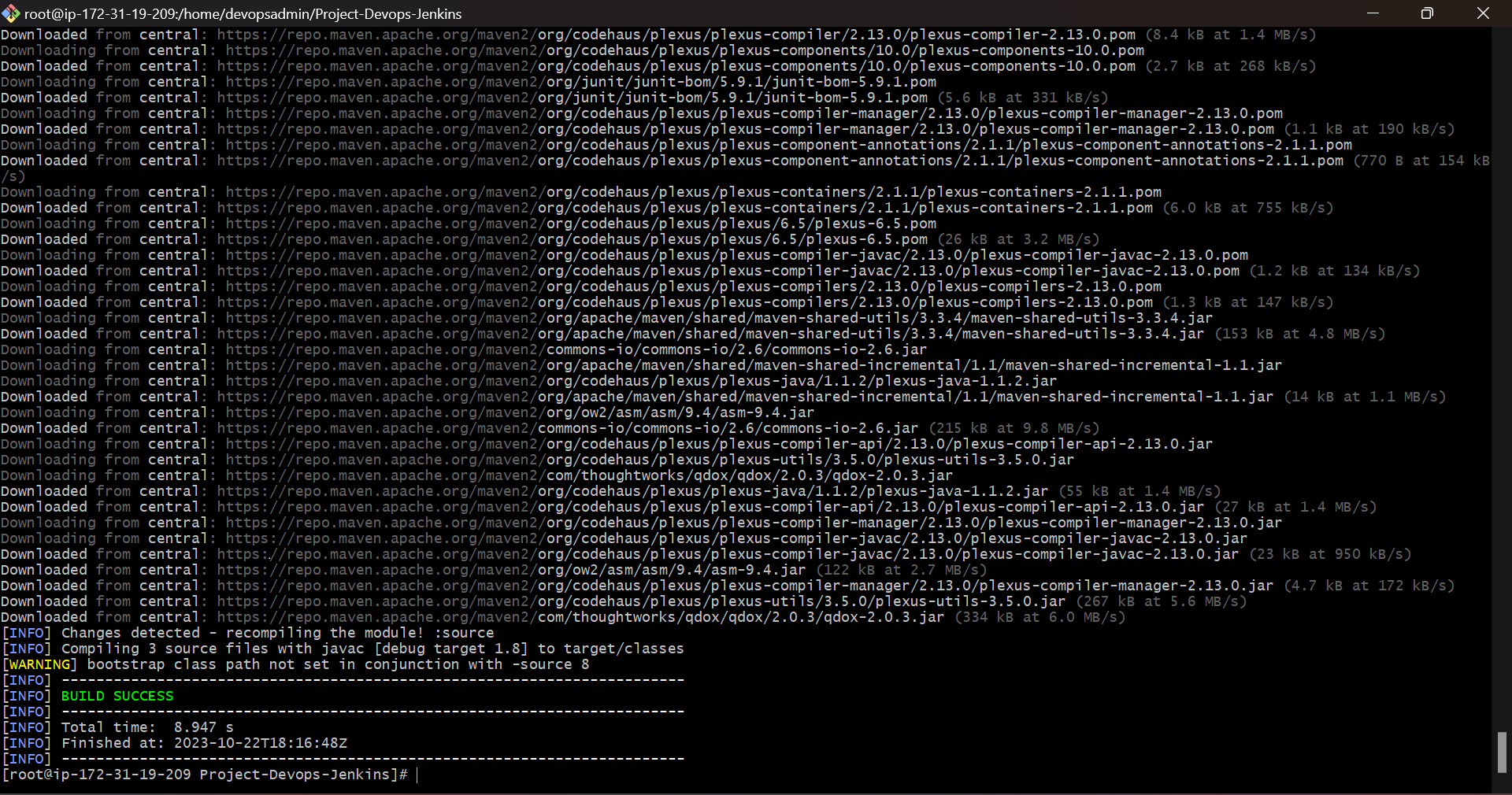
* Now that I have my repository and machine set up with all the relative tools installed such as git, maven, java and JDK set up. It is now time to build our code, so the next step is to compile, test and package our code.
* I am going to do this by using Jenkins and my local machines.

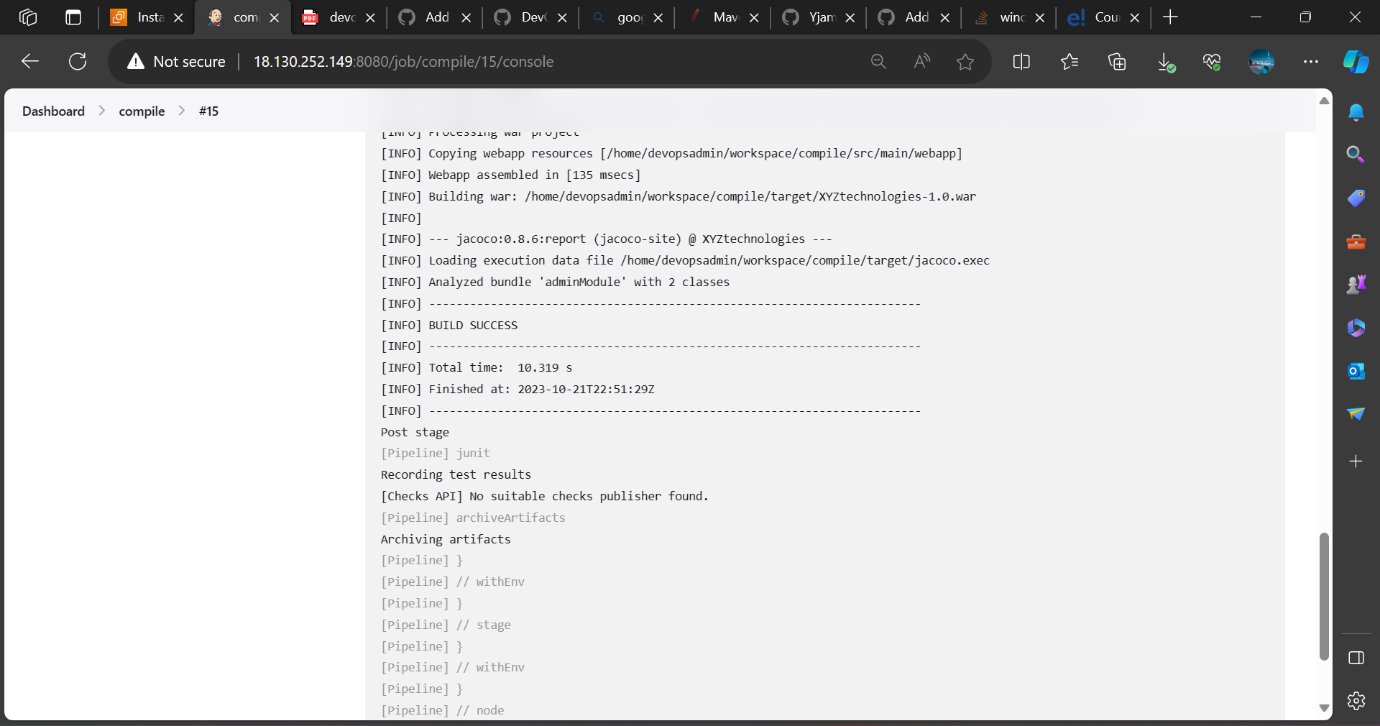
**Step 1: Compile Src code**



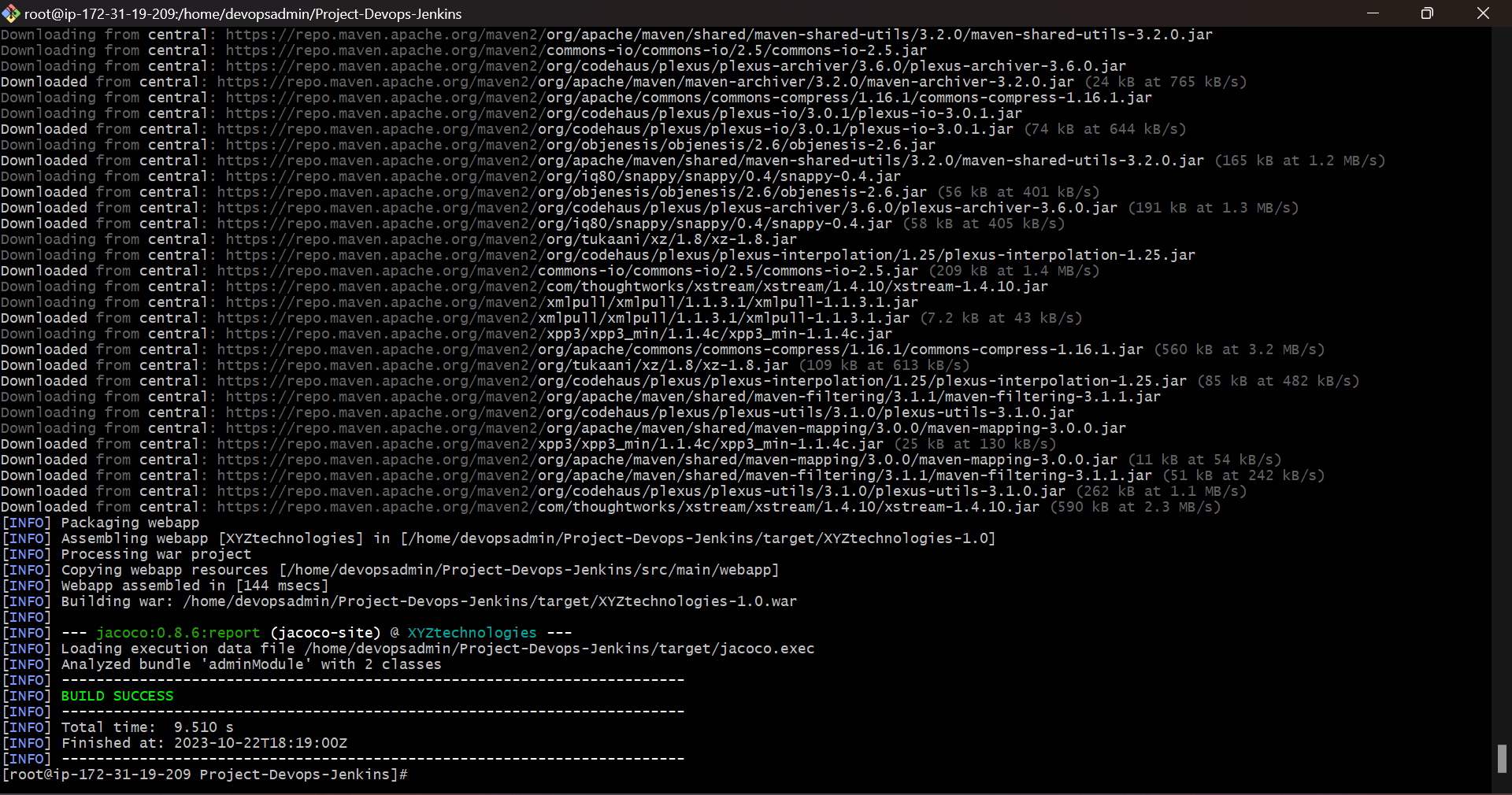


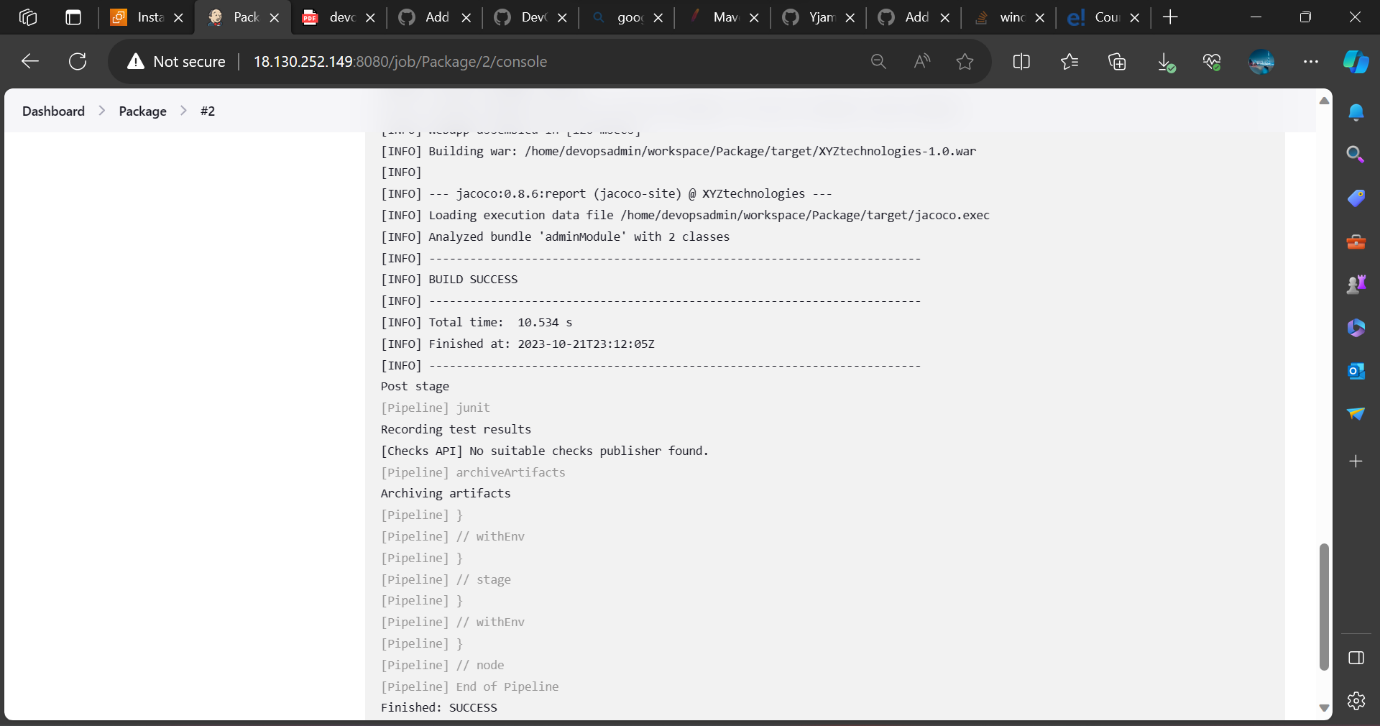
**Step 2: Test Src code**

****

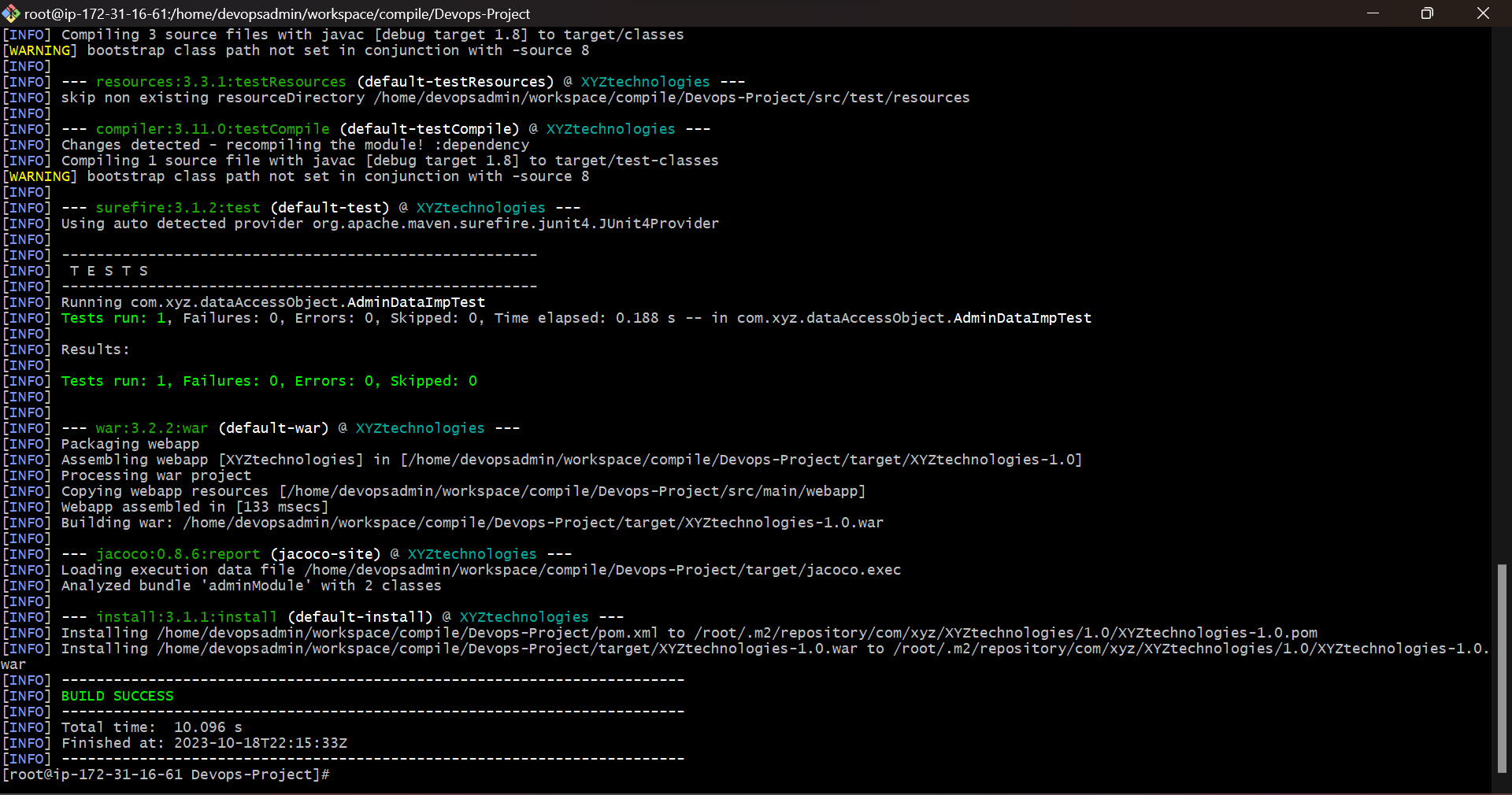
****

**Step 3: Package Src code**

****

****

**Step 4: Clean & Install**

****

I have used maven as a build tool to build the code. Maven and java are installed already in the master machine and configured it on there also. I have run the maven compile, test, and package tasks successfully in the local environment and produced the results for each and executed the maven clean install command to build the code.

I executed the below commands to achieve this.

* 1. /opt/maven/bin/mvn compile
* 2. /opt/maven/bin/mvn test
* 3. /opt/maven/bin/mvn package
* 4. /opt/maven/bin/mvn clean install

- /opt/maven/bin/mvn - Is the path of the maven executable.

1) /opt/maven/bin/mvn compile: Compiles source code of the project

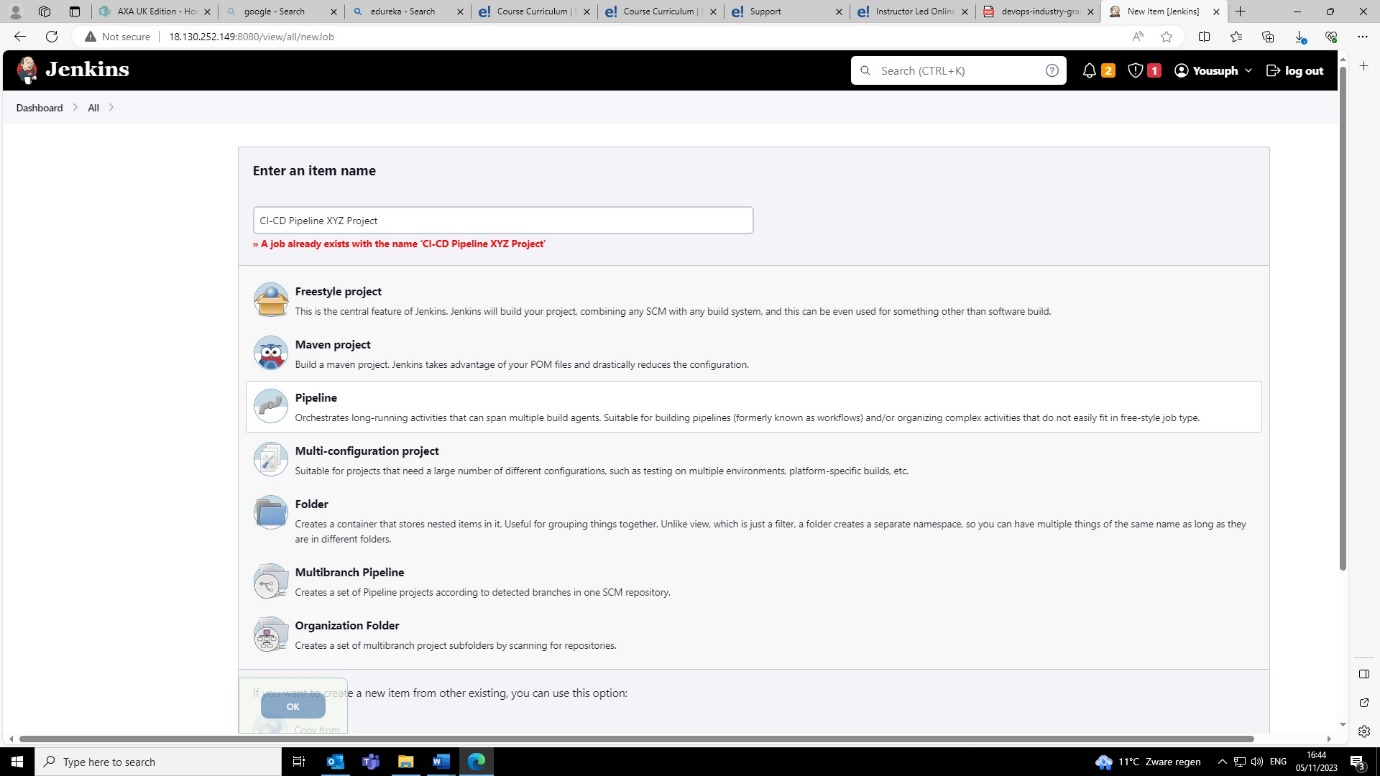
2) /opt/maven/bin/mvn test: Runs tests for the project.

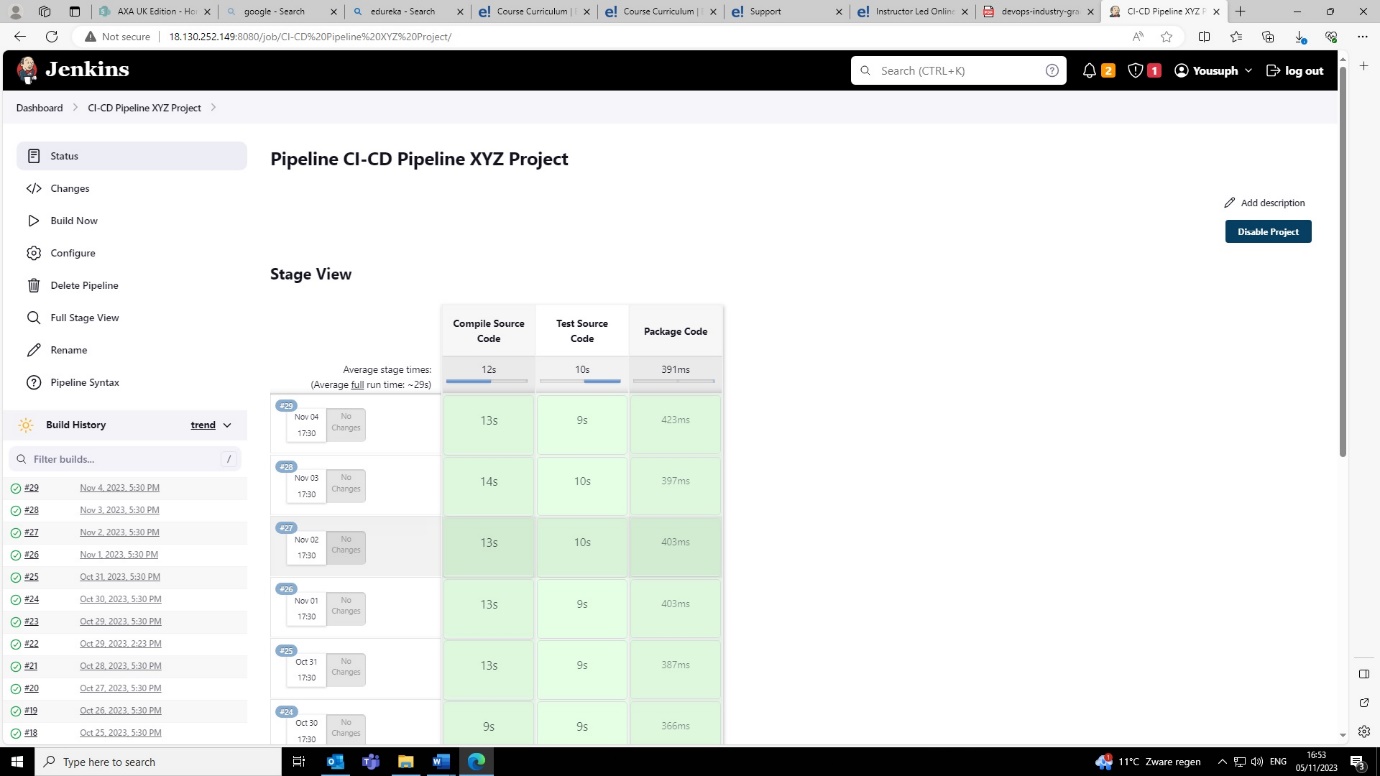
3) /opt/maven/bin/mvn package: Creates WAR file for the project to convert it into a distributable format.

4) /opt/maven/bin/mvn clean install: Using the clean command, which will delete all previously compiled Java .class files and resources in the project. build will start from a clean slate. - Install will then compile, test & package my Java project, and even install/copy my built .war file into my local Maven repository.

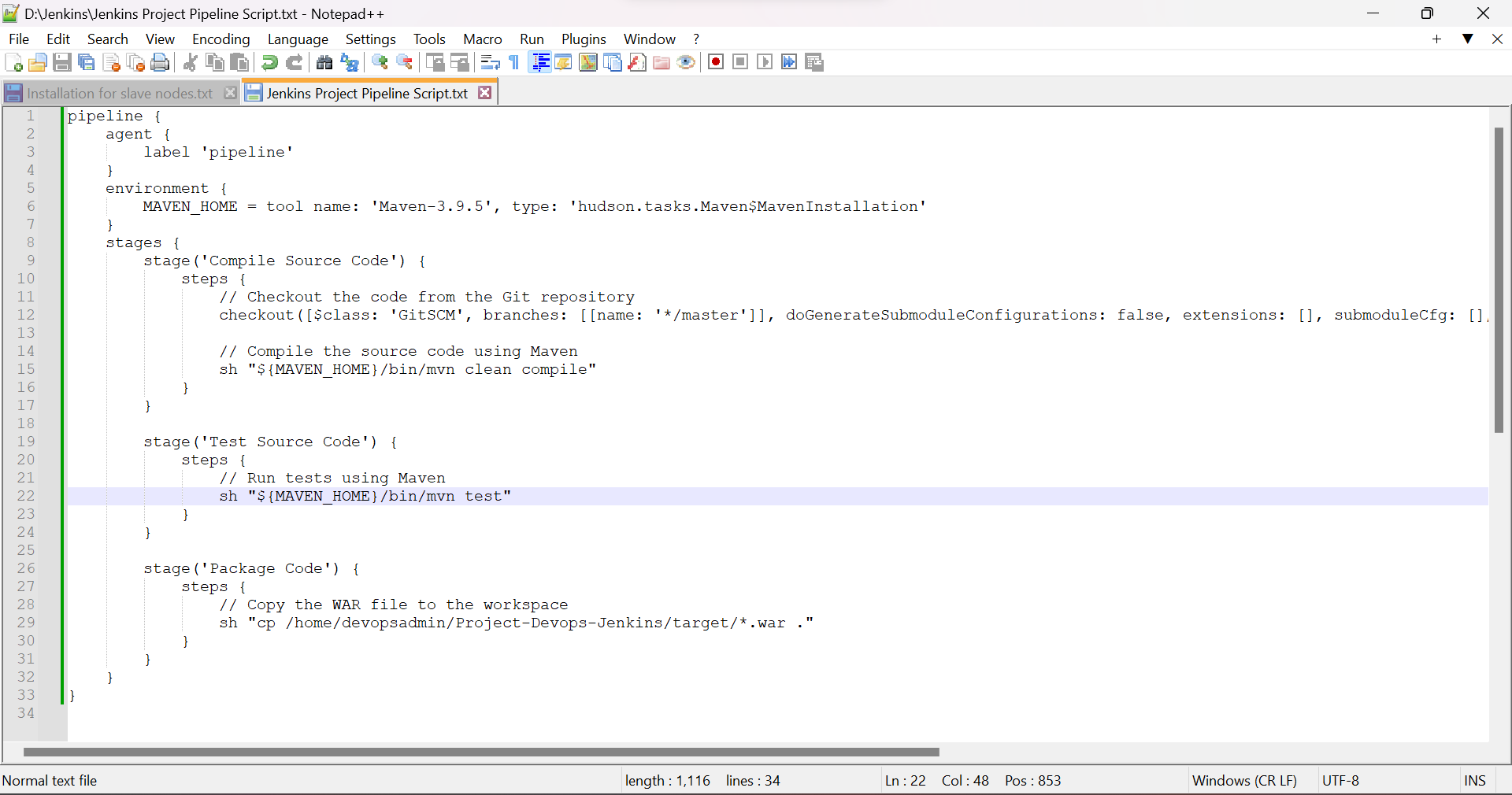
The location in which I check out the source code is from this GitHub repository - <https://github.com/Yjama45/XYZ-Technology.git>

**Step 5: Create a new pipeline project**

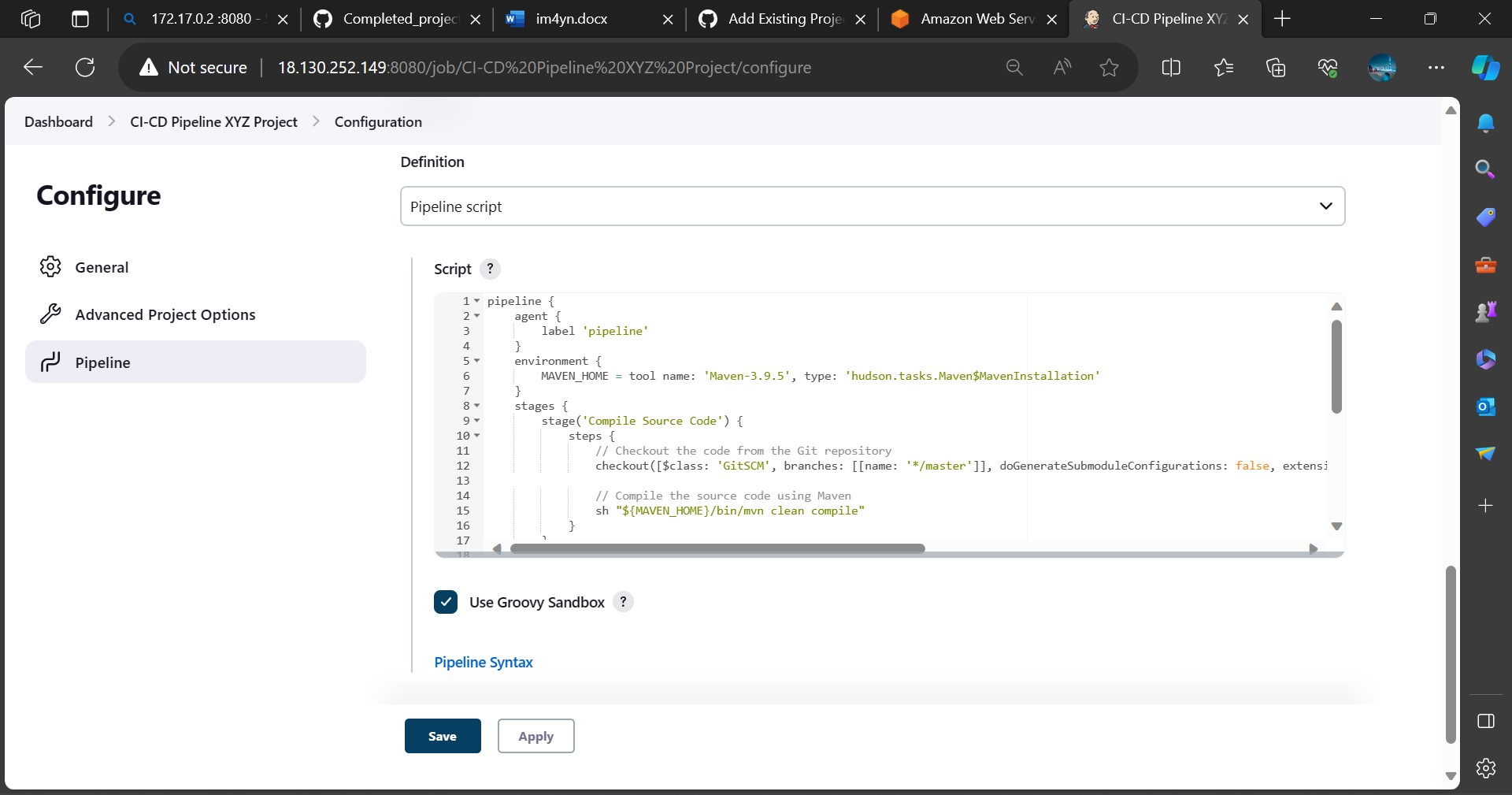
****

****

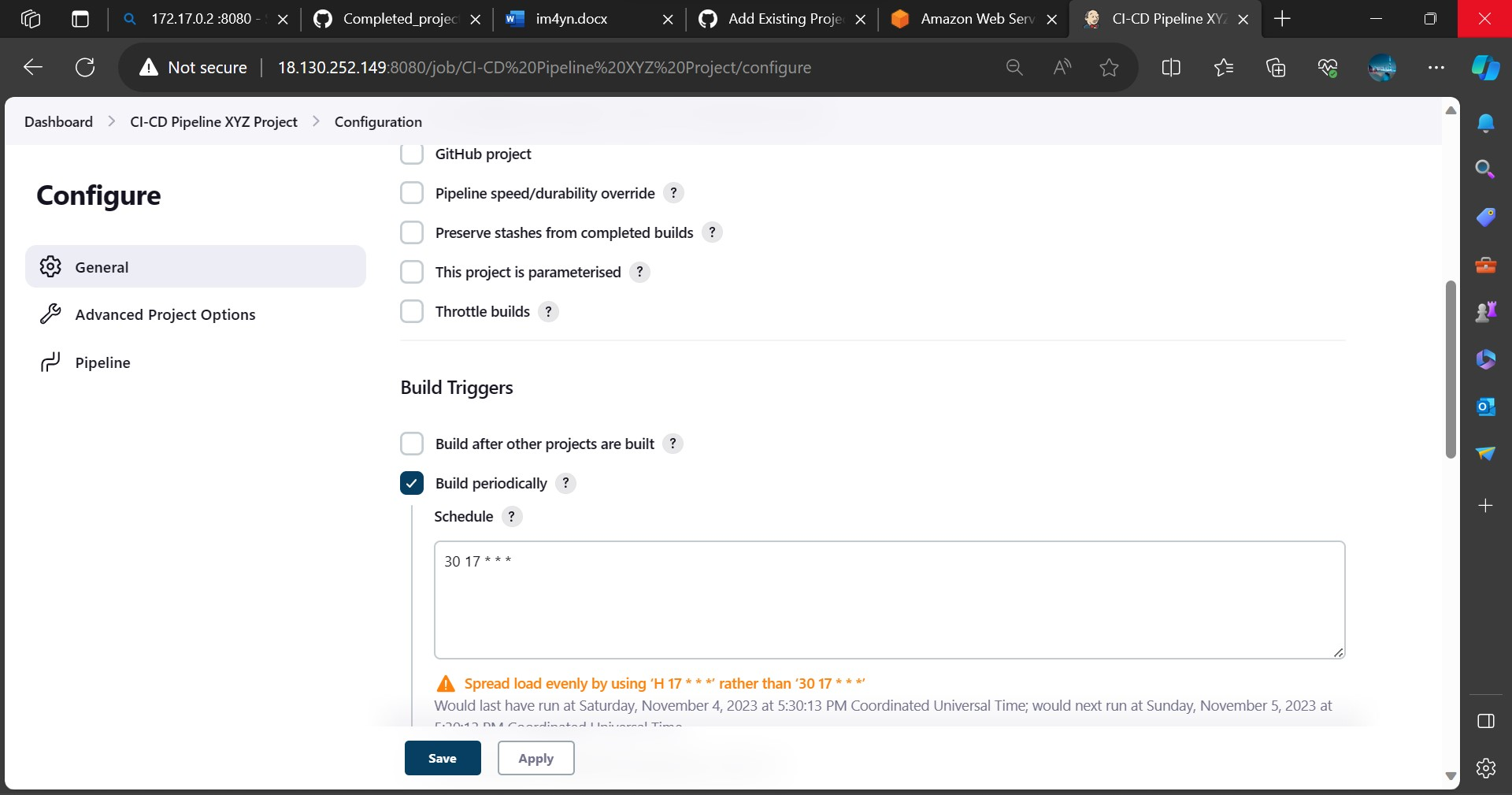
* This is my Jenkins dashboard. I created a pipeline to compile, test, and package the src code.
* The way I created it is as followed.



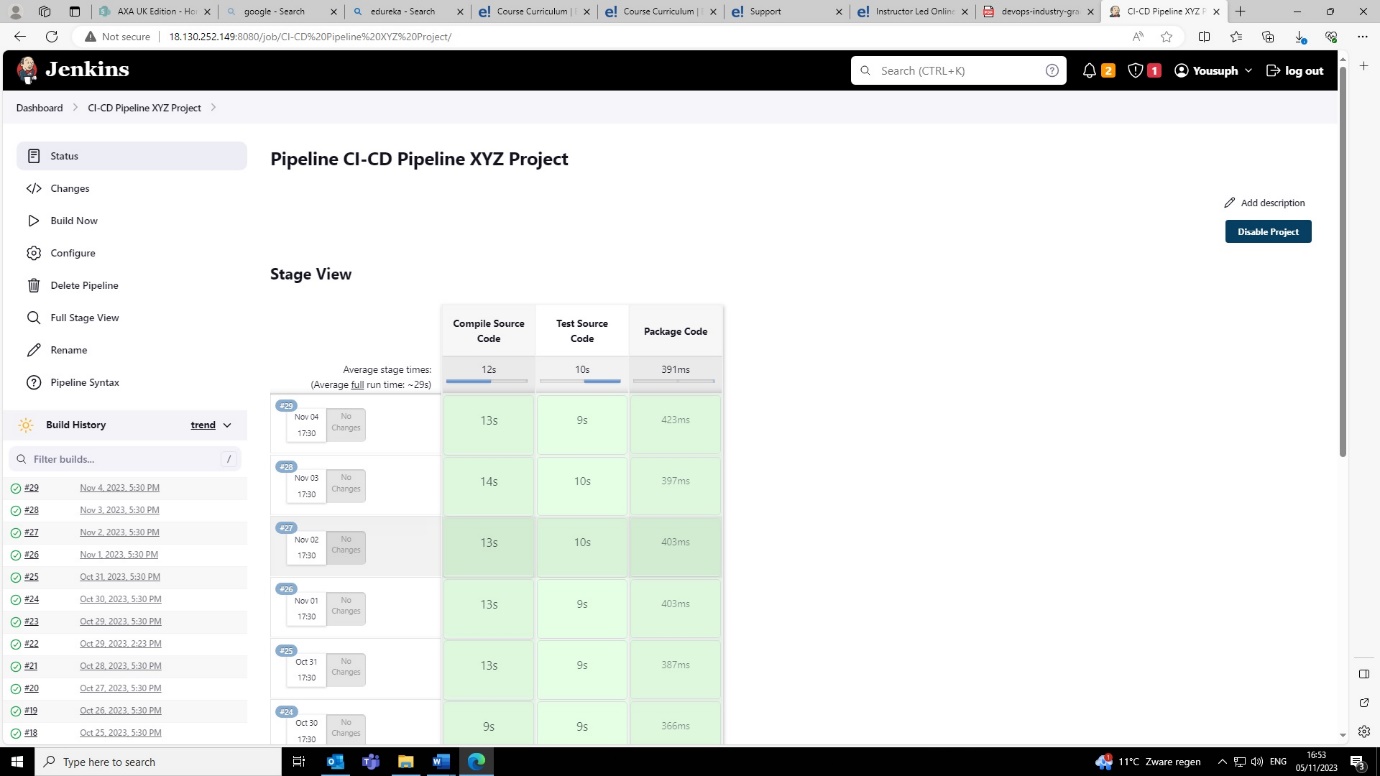
* I created this groovy script to create the stages above to carry out the tasks specified on the script.



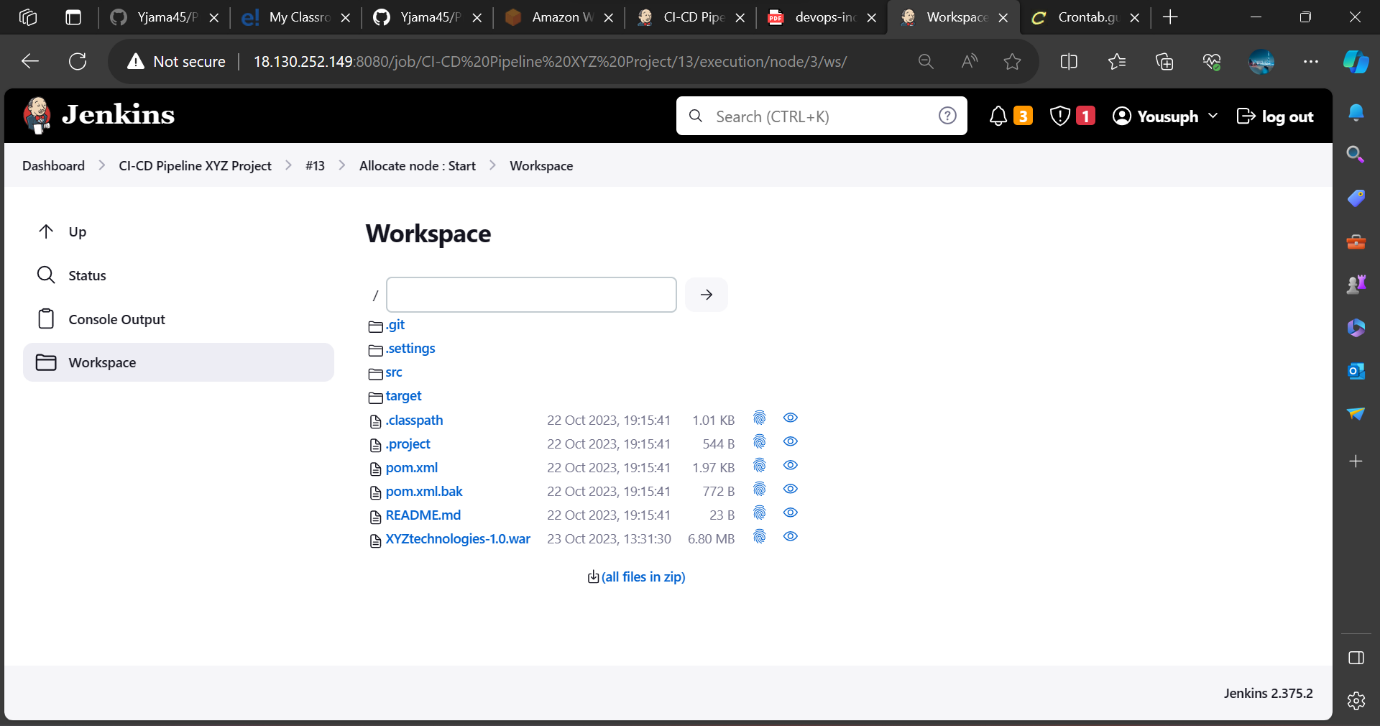
* I then configured it on my Jenkins pipeline project.



* I then created a cron job to perform the build every day at 17:30 pm

****

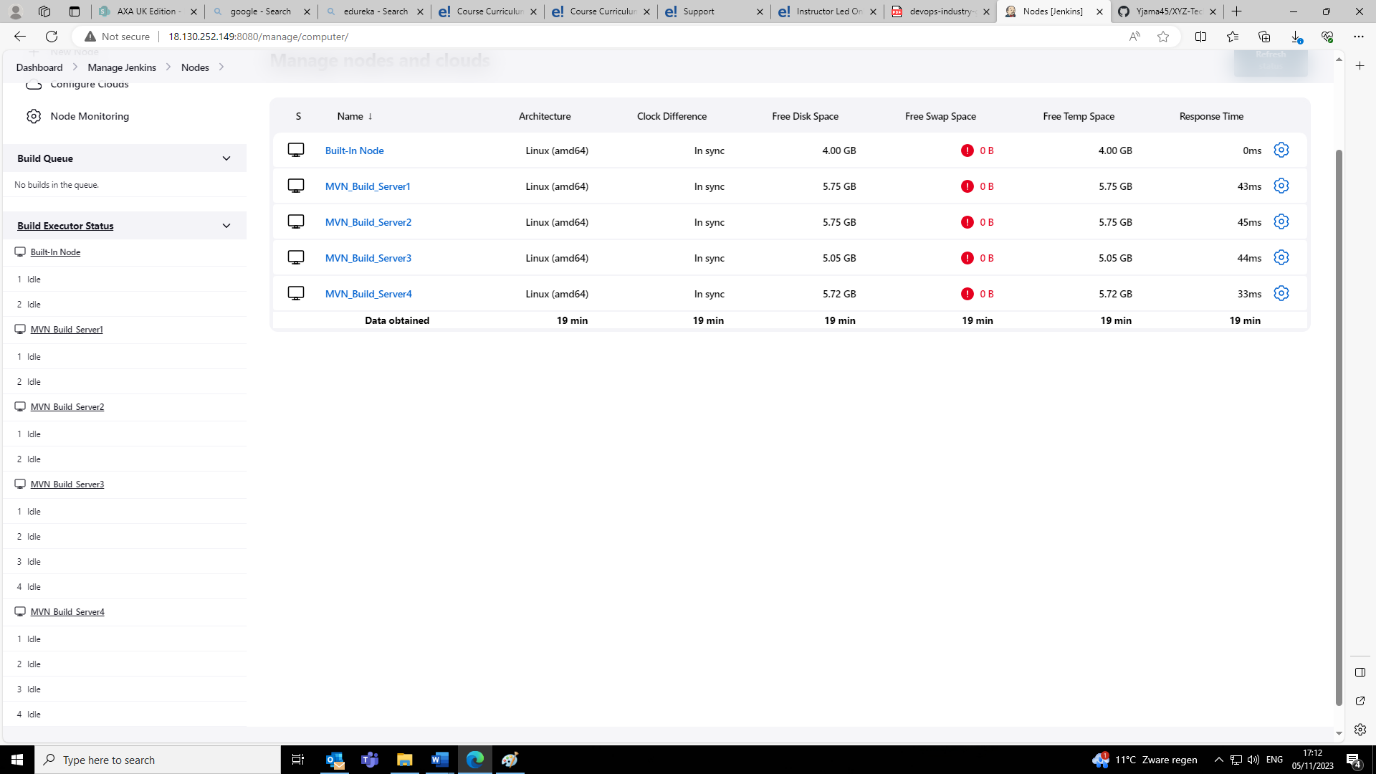
* My dashboard now looks like this, as you can see there has been successful builds done everyday at 17:30pm. These builds successfully compile, test, and package the source code.

****

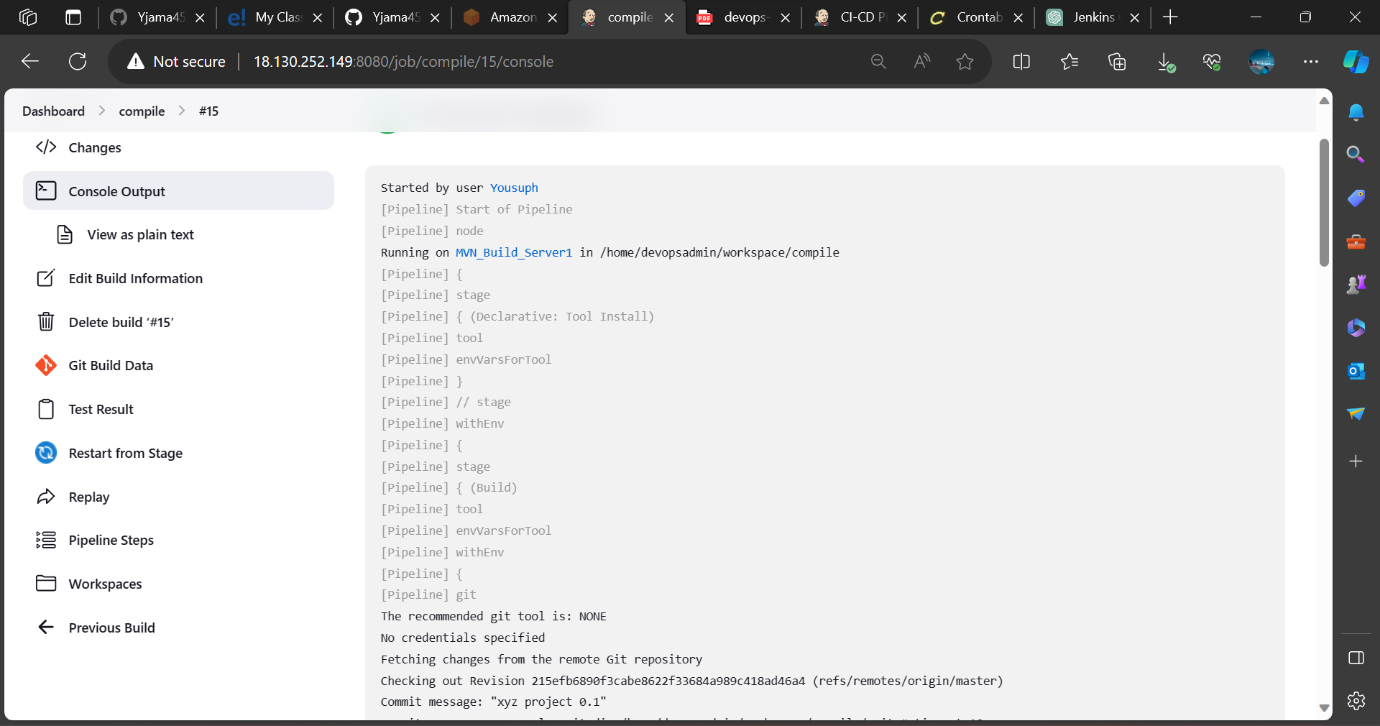
* This is the location of my files containing the src code on my Jenkins master machine.

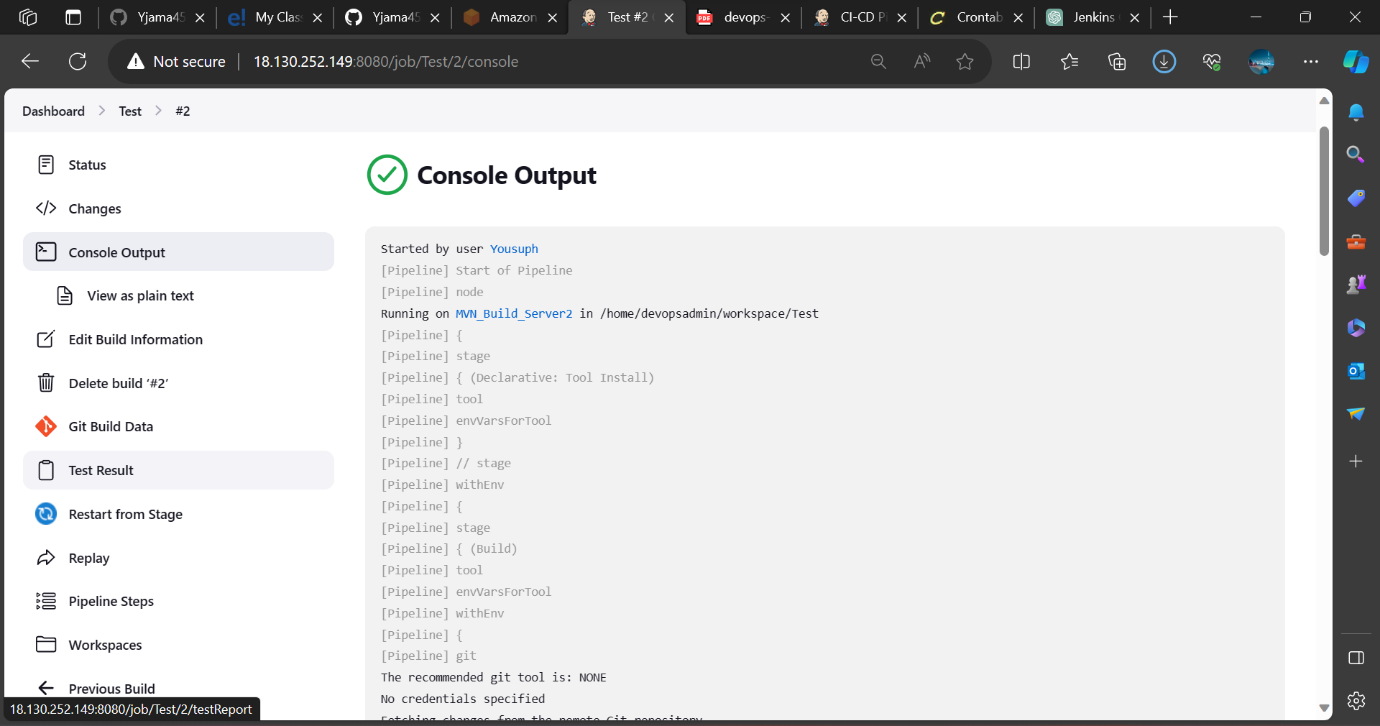
**Step 6: configure master slave architecture and distribute jobs onto different nodes**

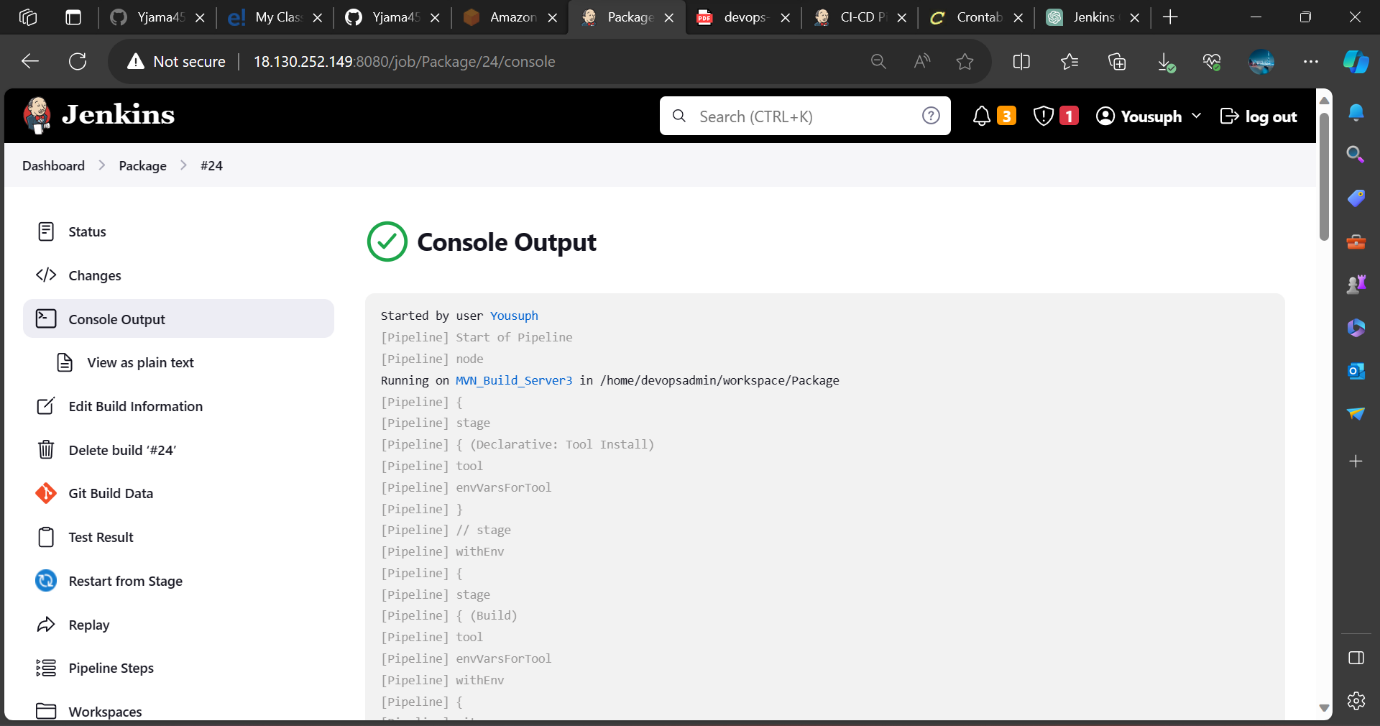
The next step was to configure 3 nodes to carry out the different jobs which was to compile, test and package the source code.



* Each node as mentioned above has different jobs and I differentiated them by attaching a label to each node. Whenever a build is performed it will perform it on one of the nodes I configured on the master machine, the purpose of this is to evenly distribute the load.



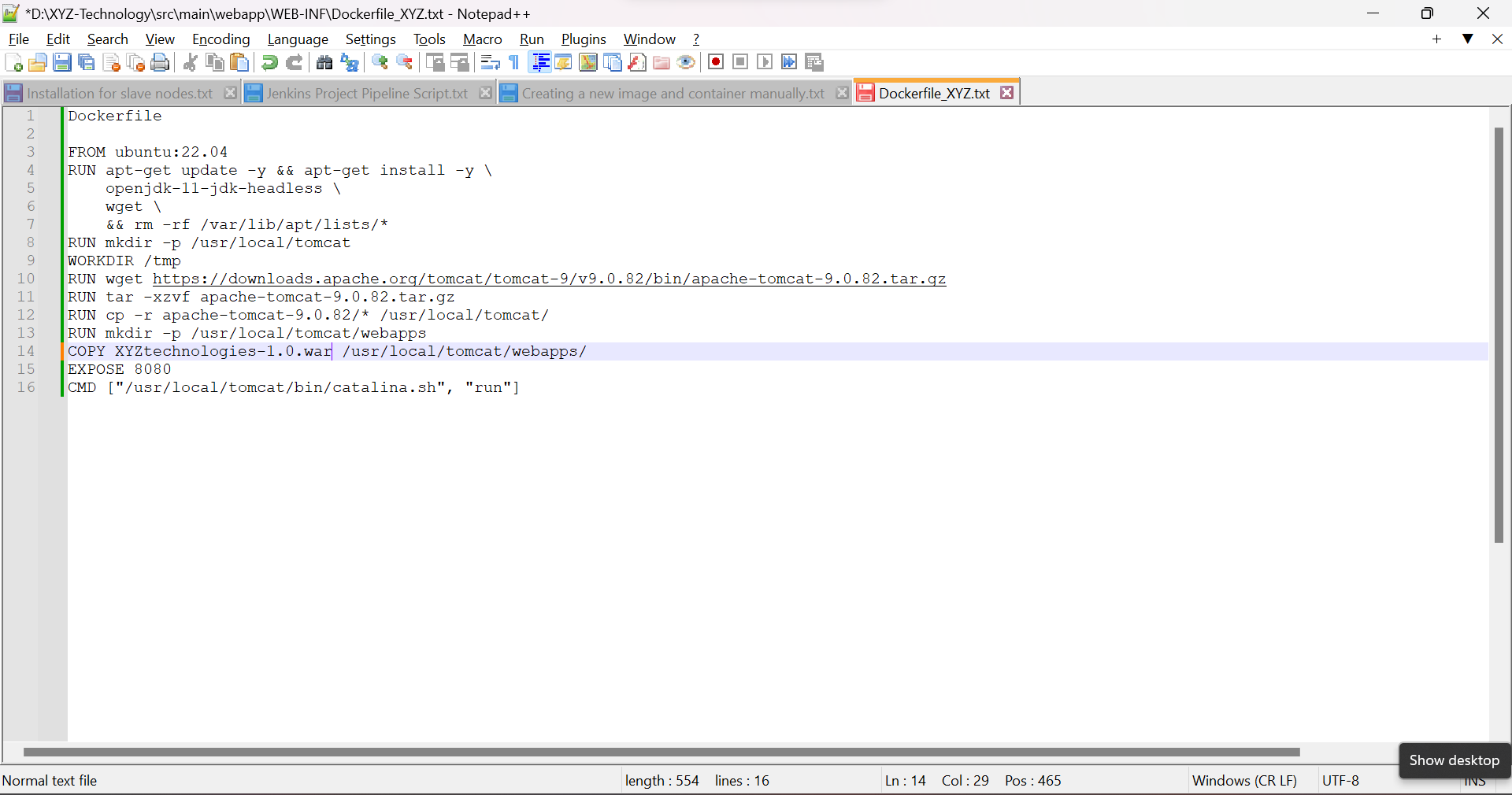




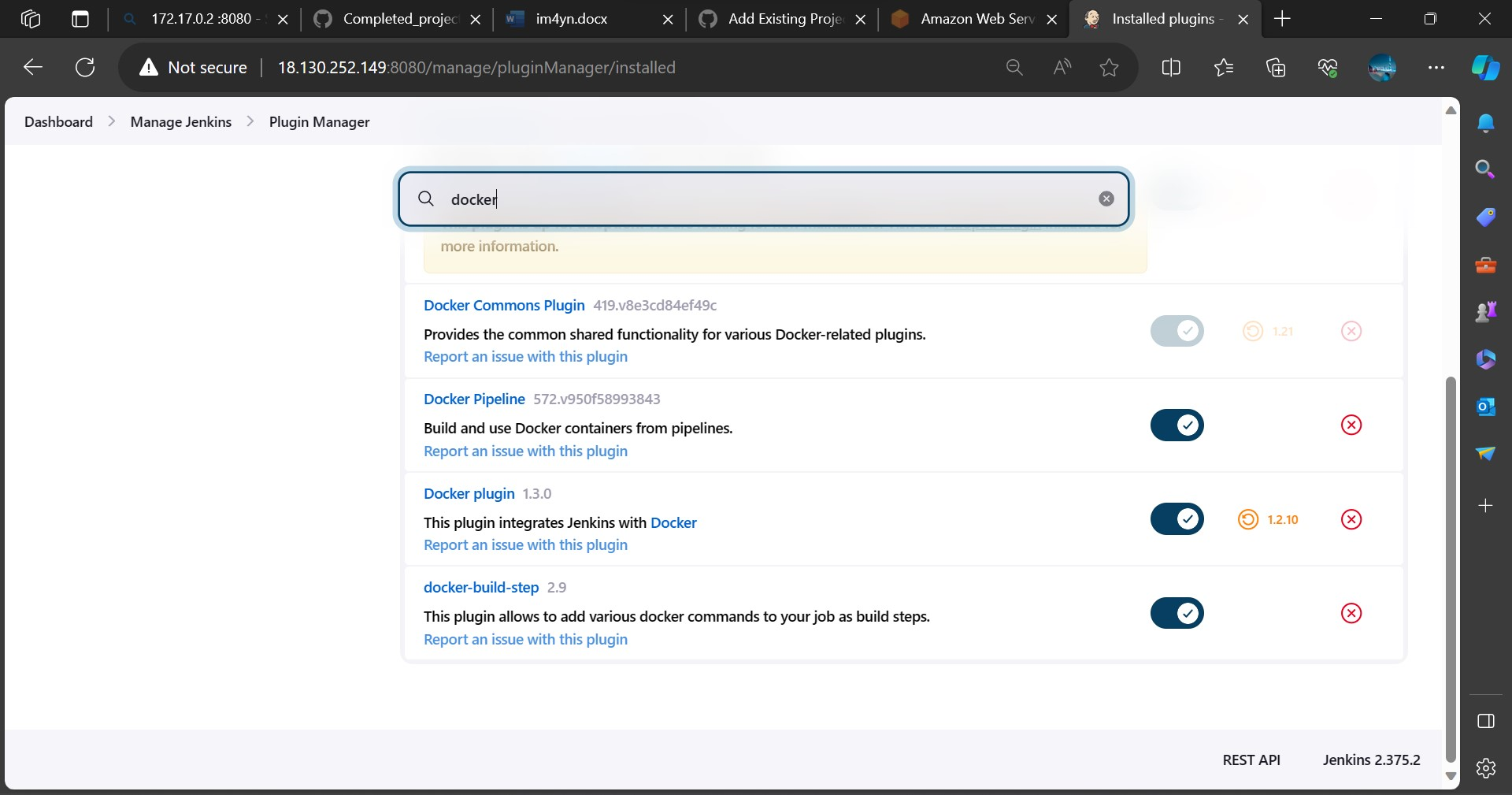
* As you can see the above builds were performed on 3 separate nodes. Each node performs a job which I specified on my script.
* The purpose of this is to load balance the workflow. Instead of having one node perform 3 jobs it is better to have 3 perform a single job for efficiency.

***Task 3***

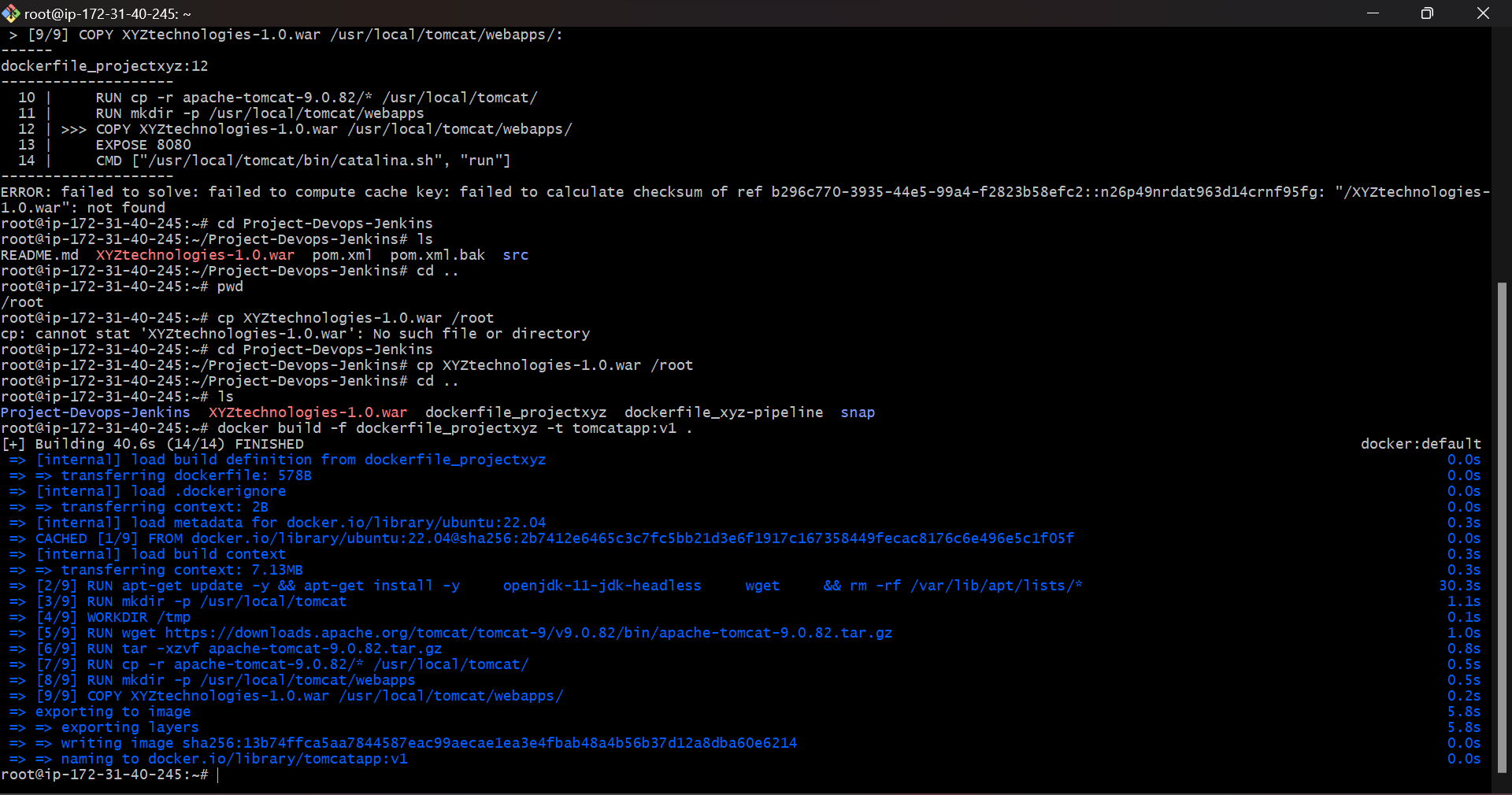
**Step 1: Create a dockerfile**

****

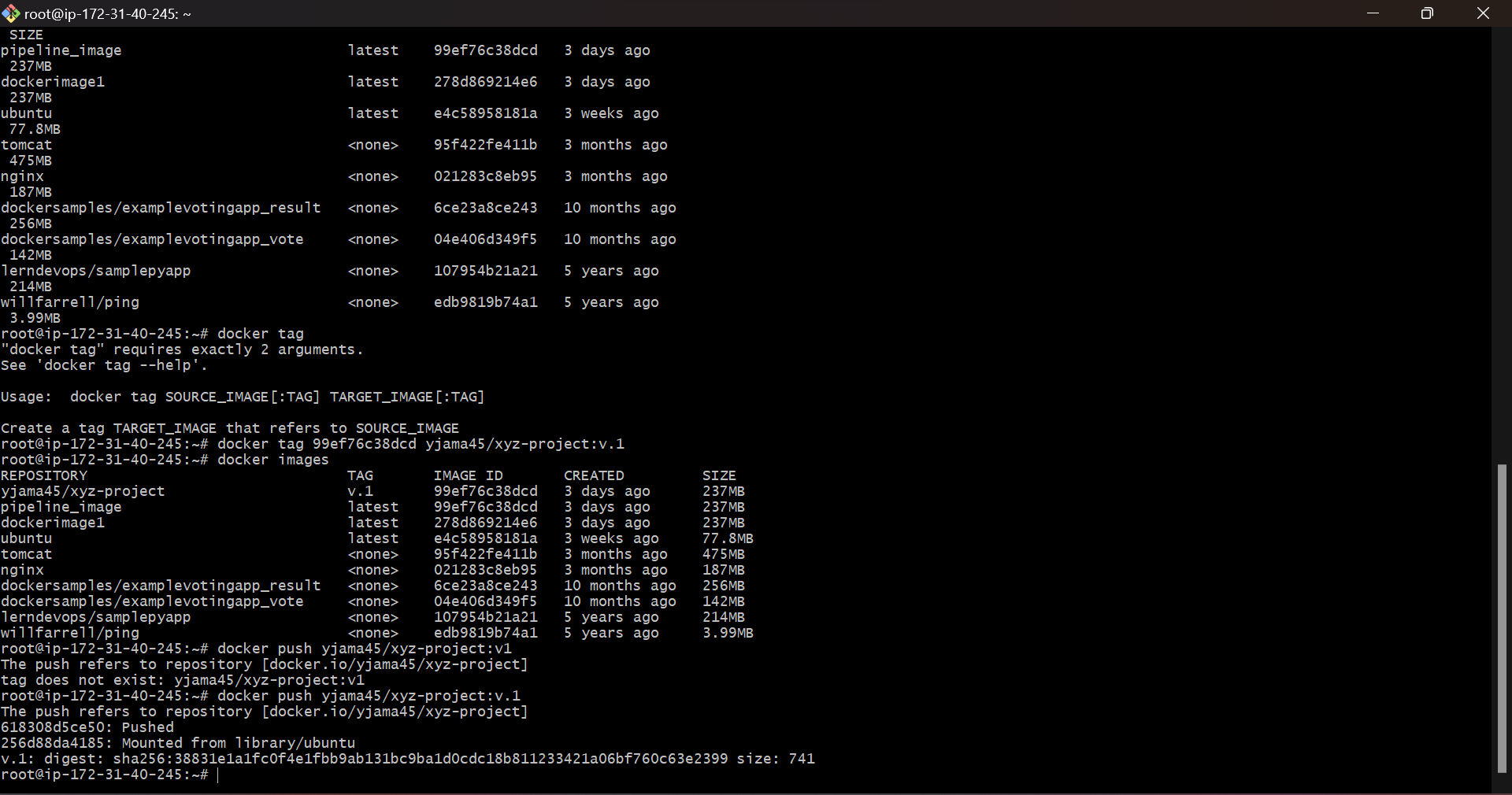
**Step 2: install the docker plugins**



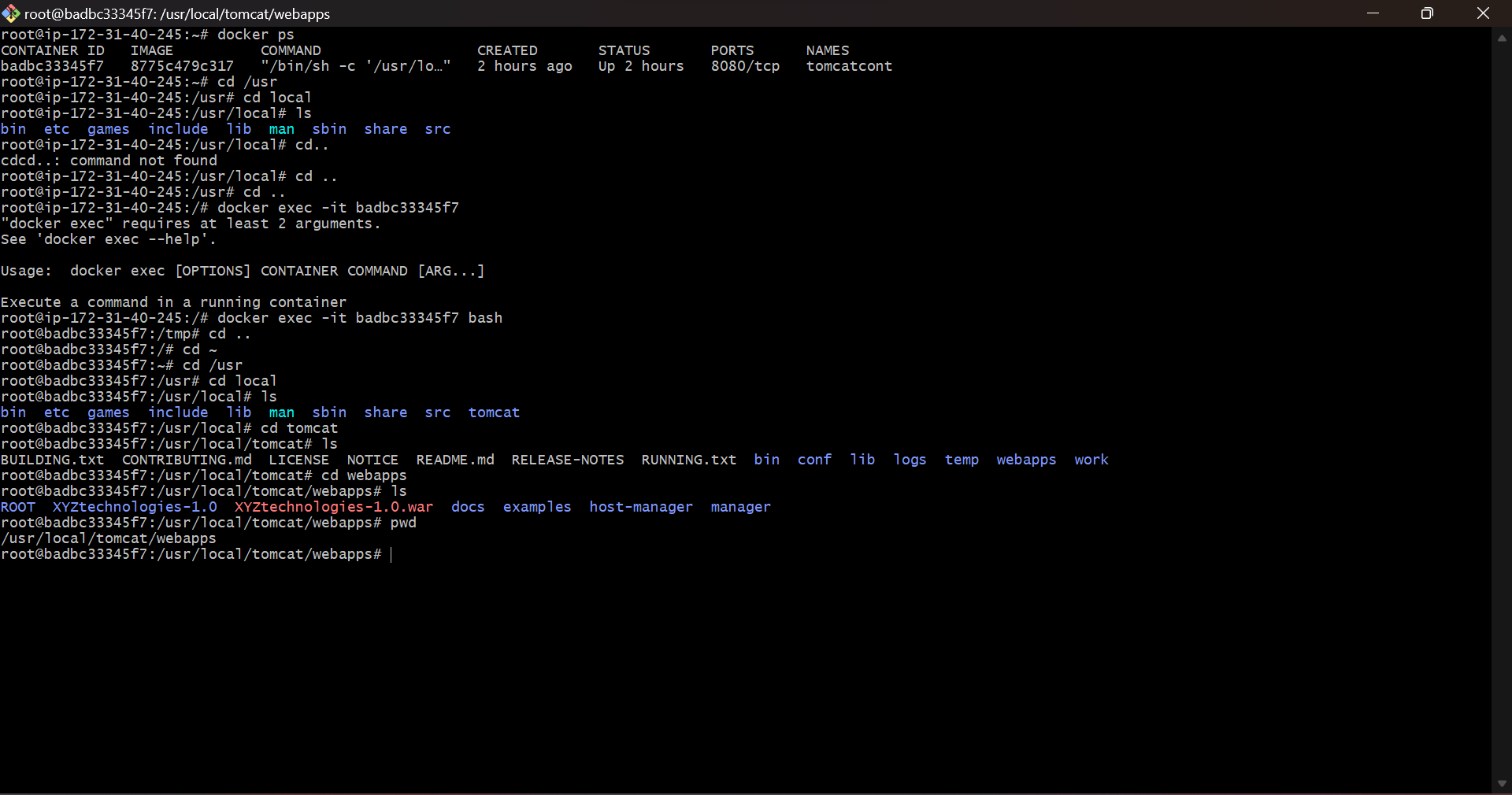
**Step 3: Create the image and container to host the webapp.**



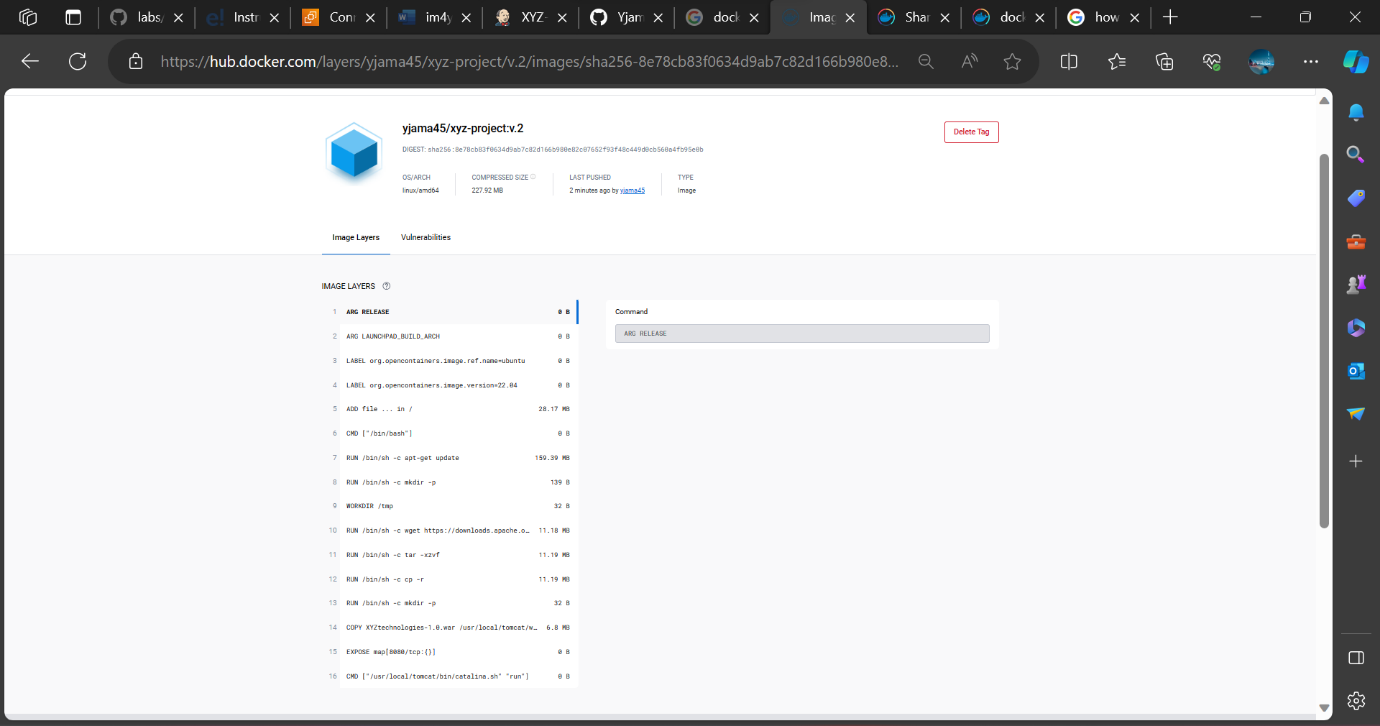
* Performed a build from the dockerfile I created which carried out the above actions stated in my dockerfile.



* Created a container from the image.

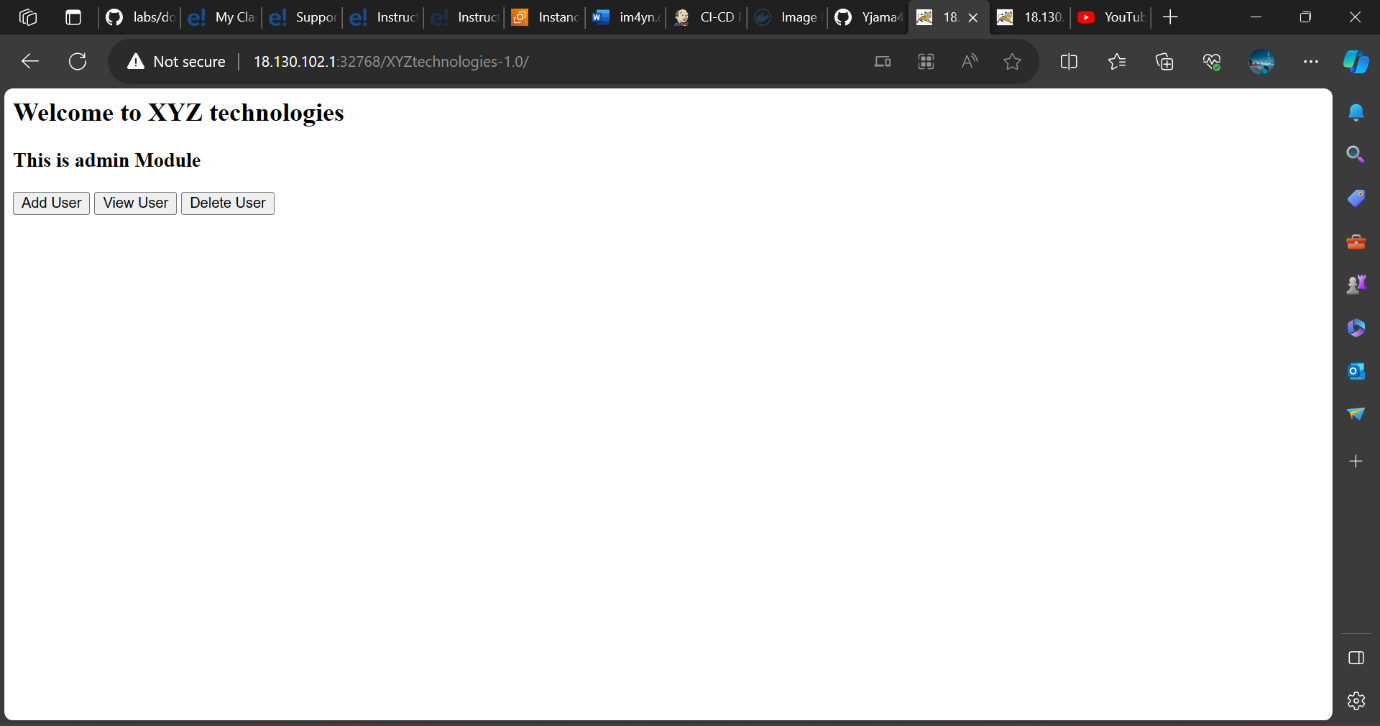


* Entered my container and went to the location where my war file is located.



* My image in my docker hub repository I created with all its necessary layers I stated in my dockerfile.

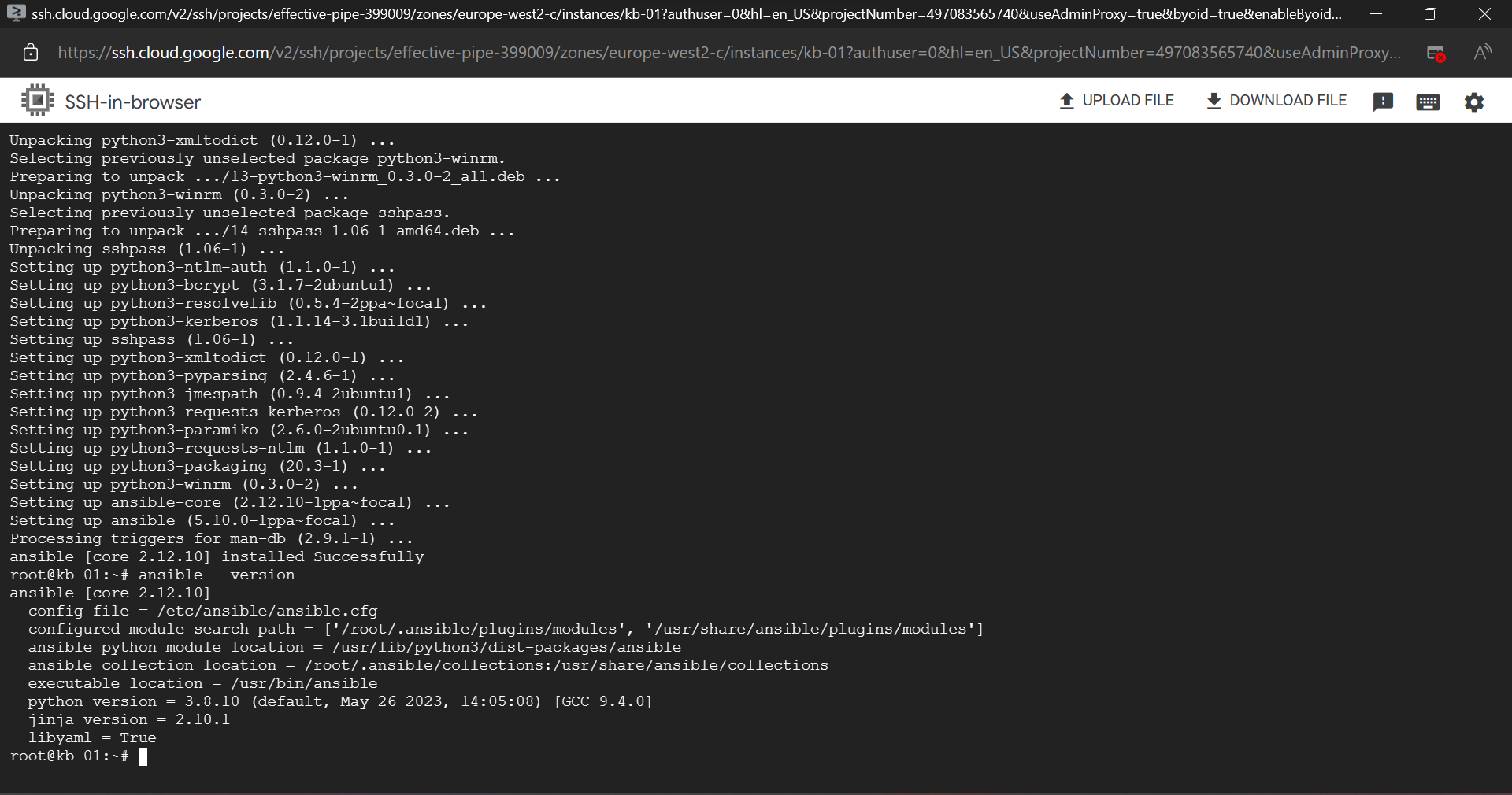
**Step 4: Validate the results**

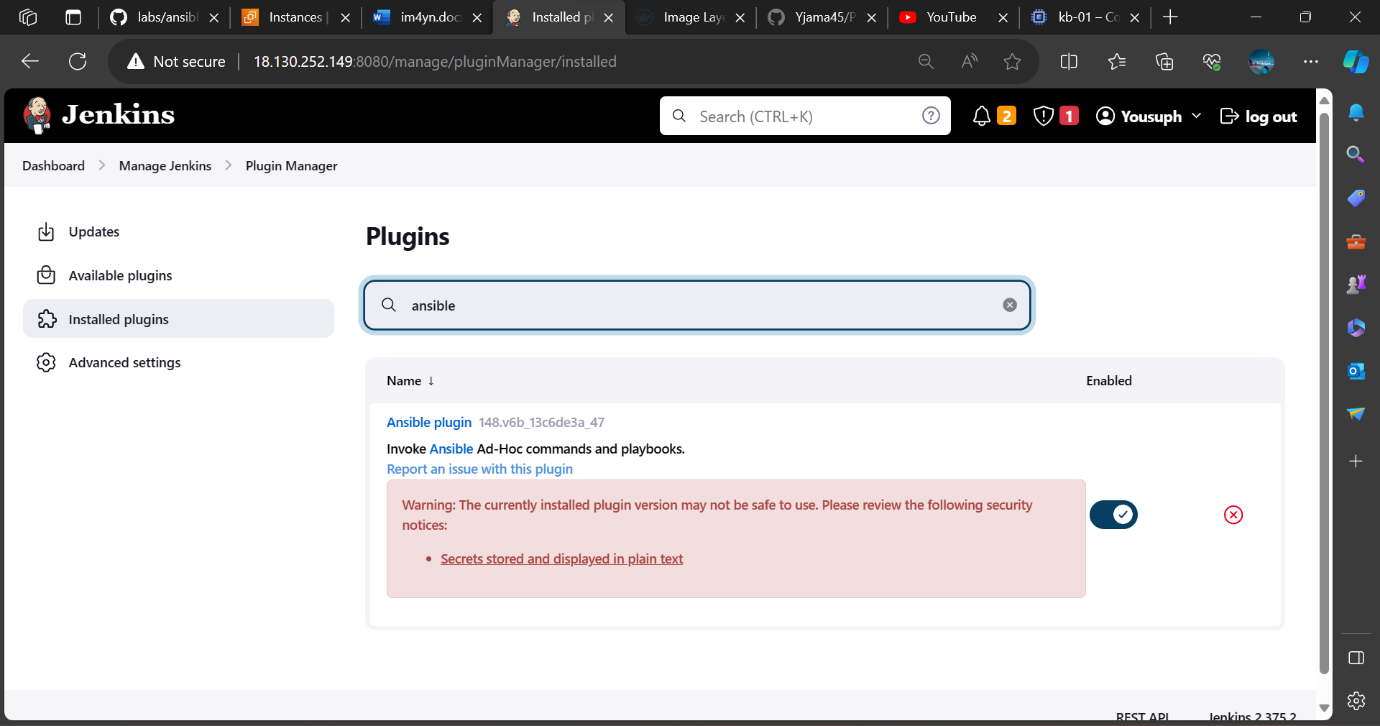


* Using the IP address and port mapping of the container and image I created I was able to validate the results.

***Task 4***

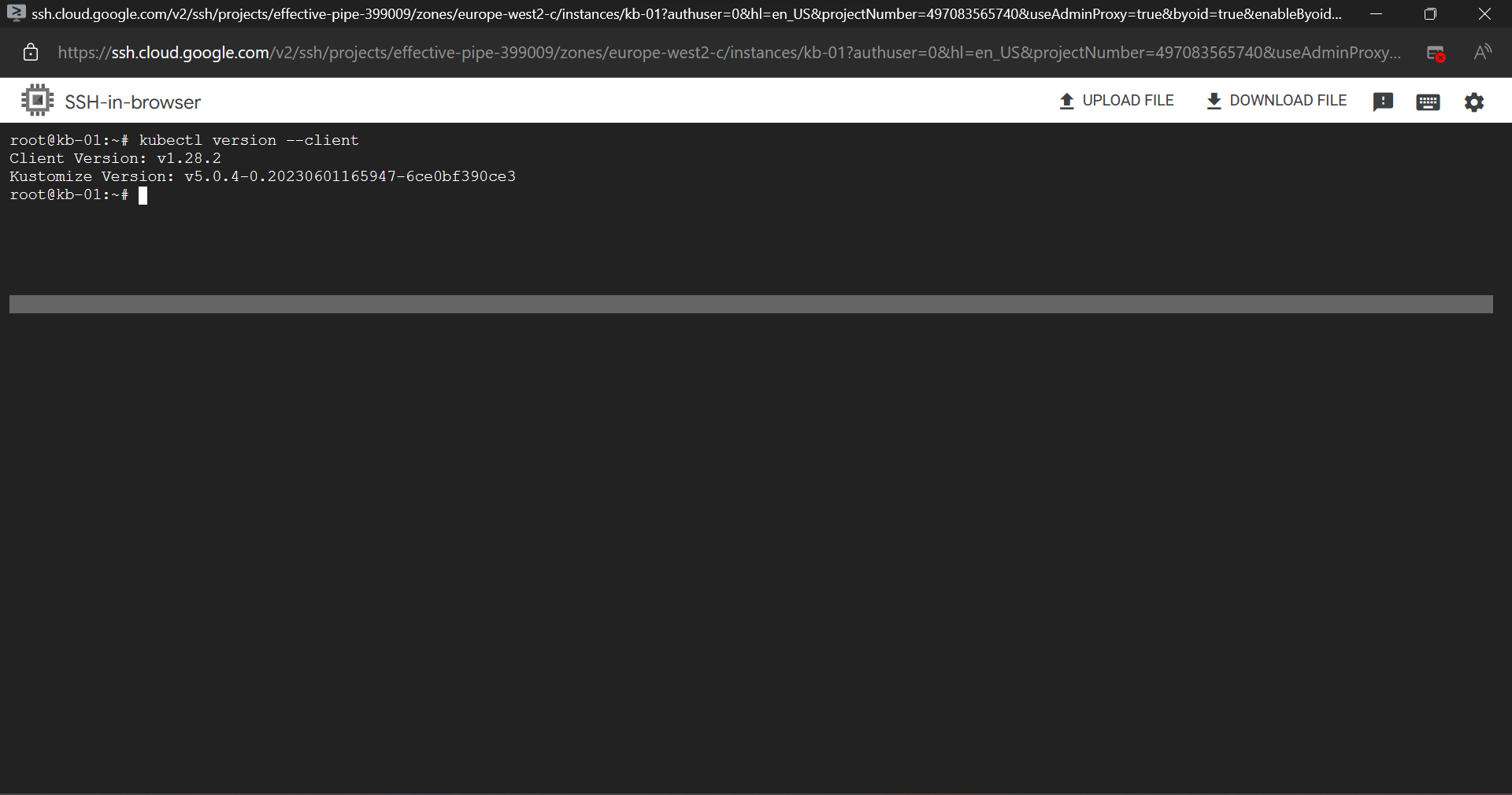
**Step 1: Install Ansbile**

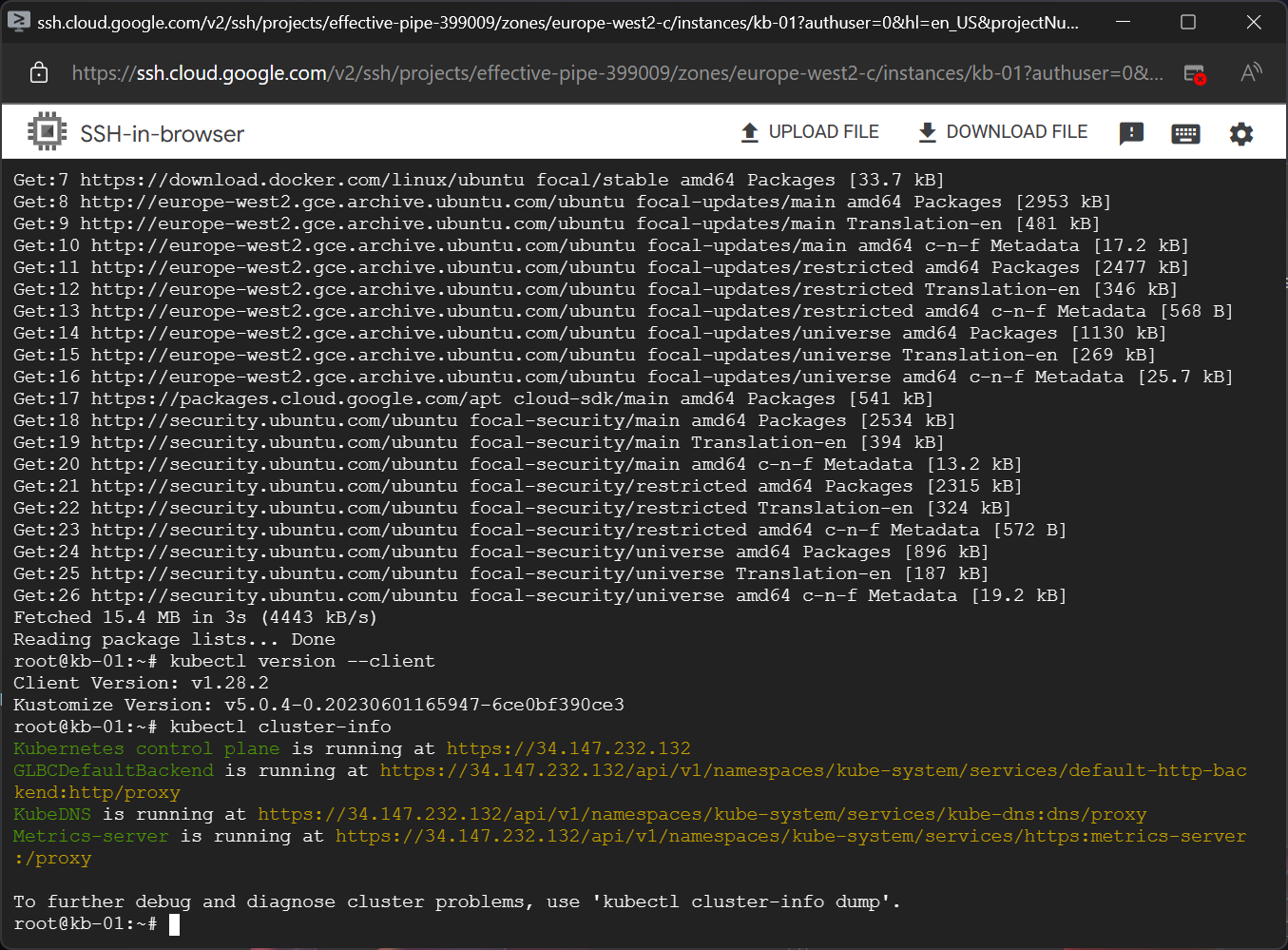




* Install ansible on machine and also install plugins on Jenkins machine

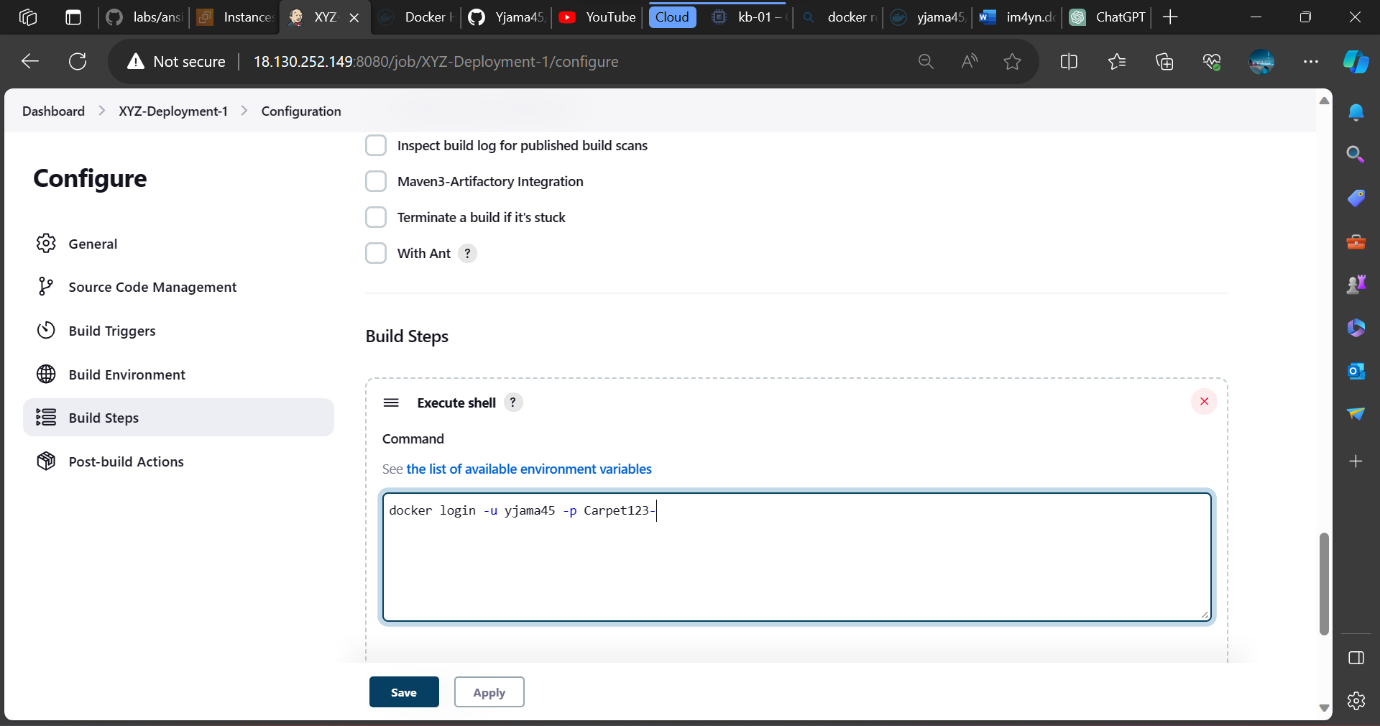
**Step 2: Install Kubernetes**

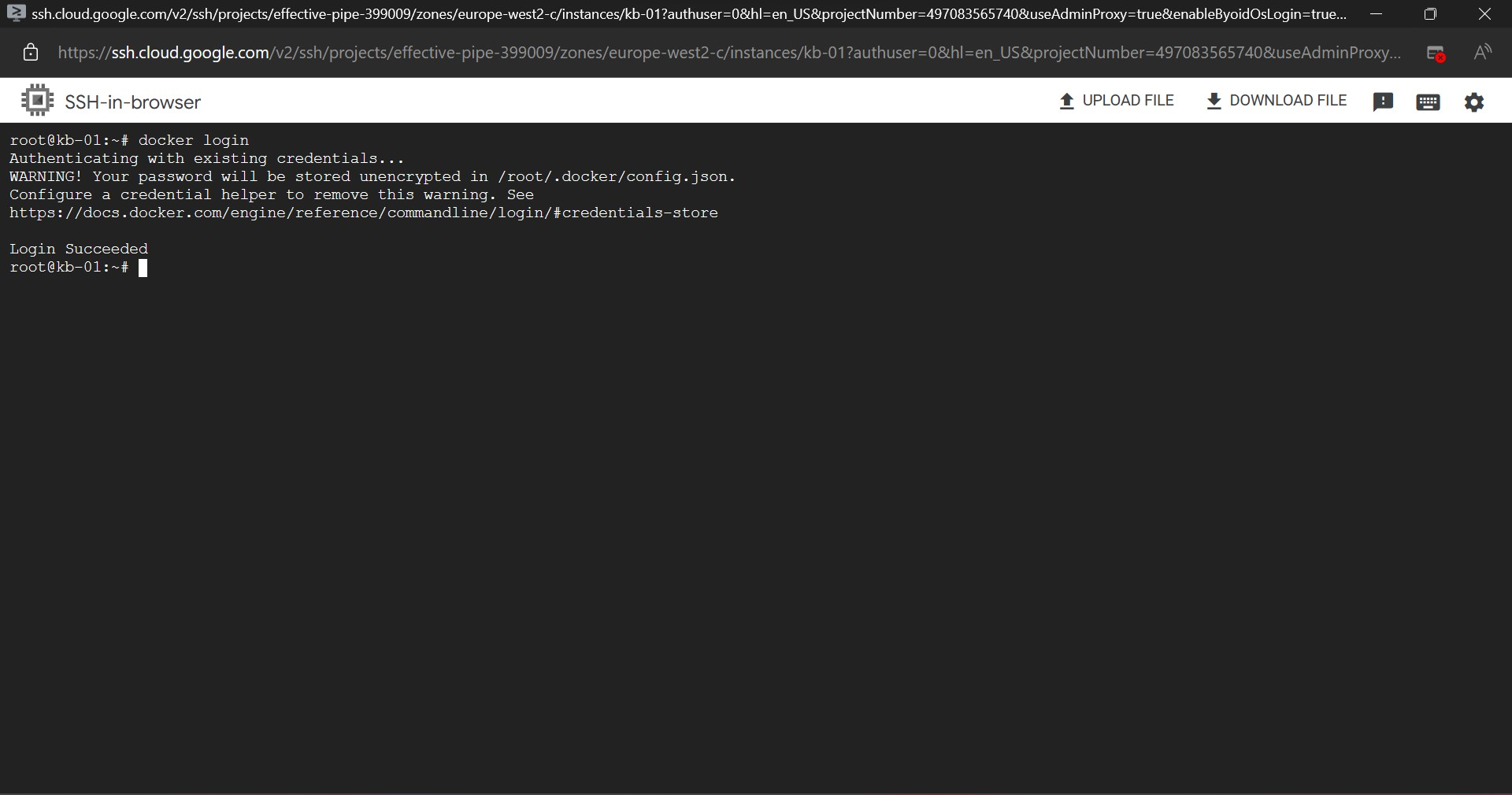
****

****

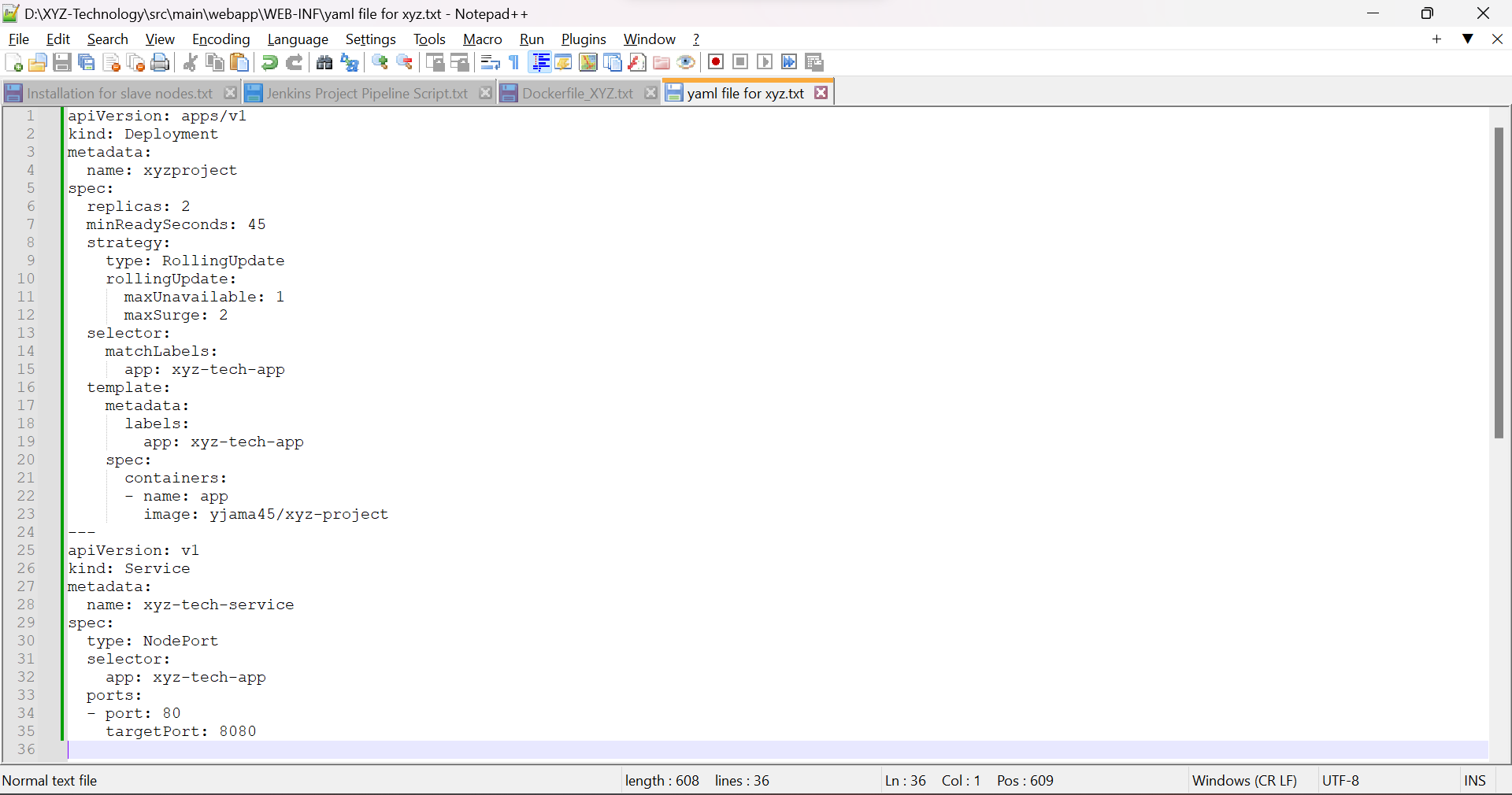
* Installed Kubernetes on my machine and gcloud cli.

**Step 3: Log into Docker**

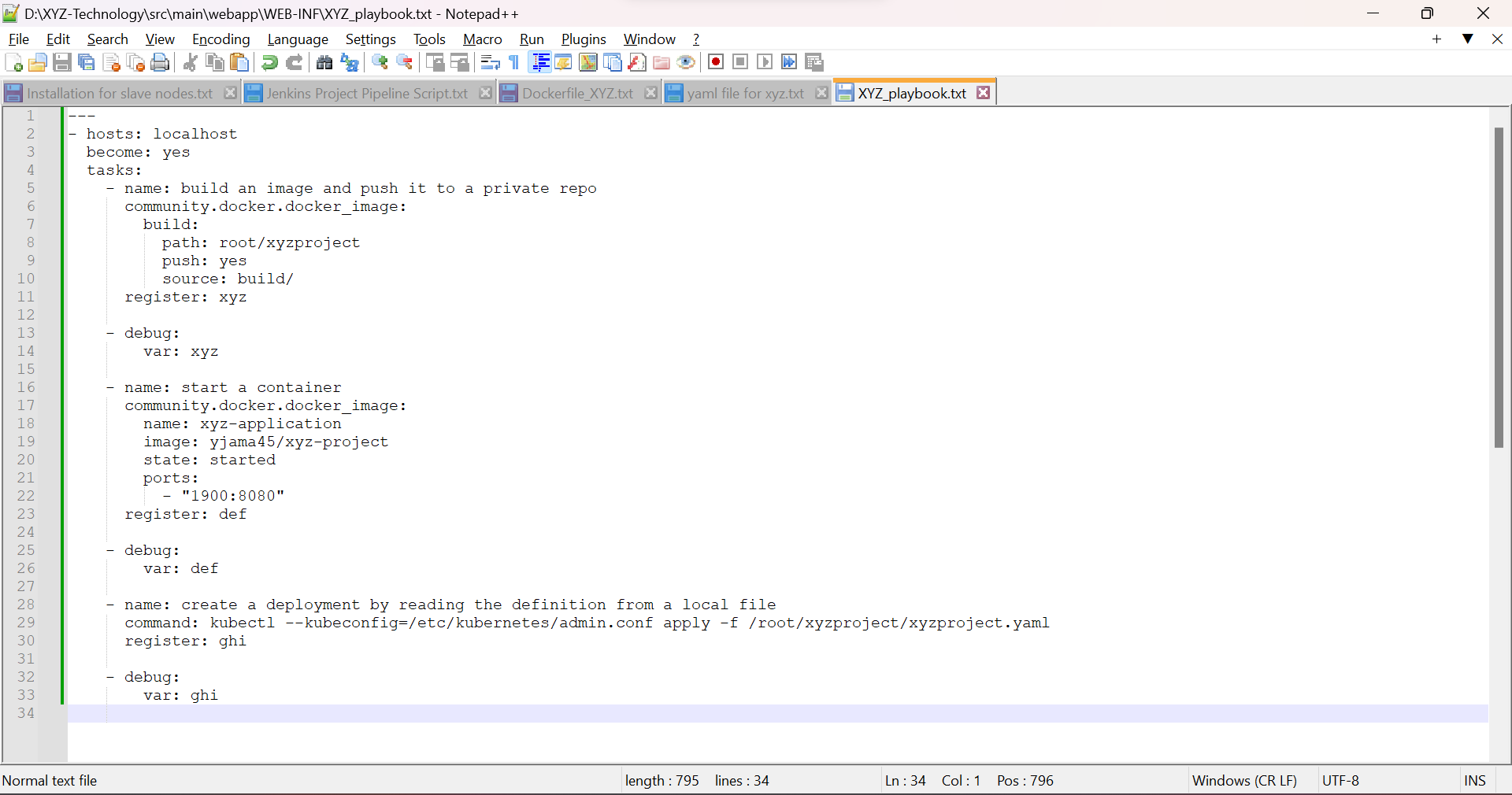
****



**Step 4: create a yaml file and playbook**

****

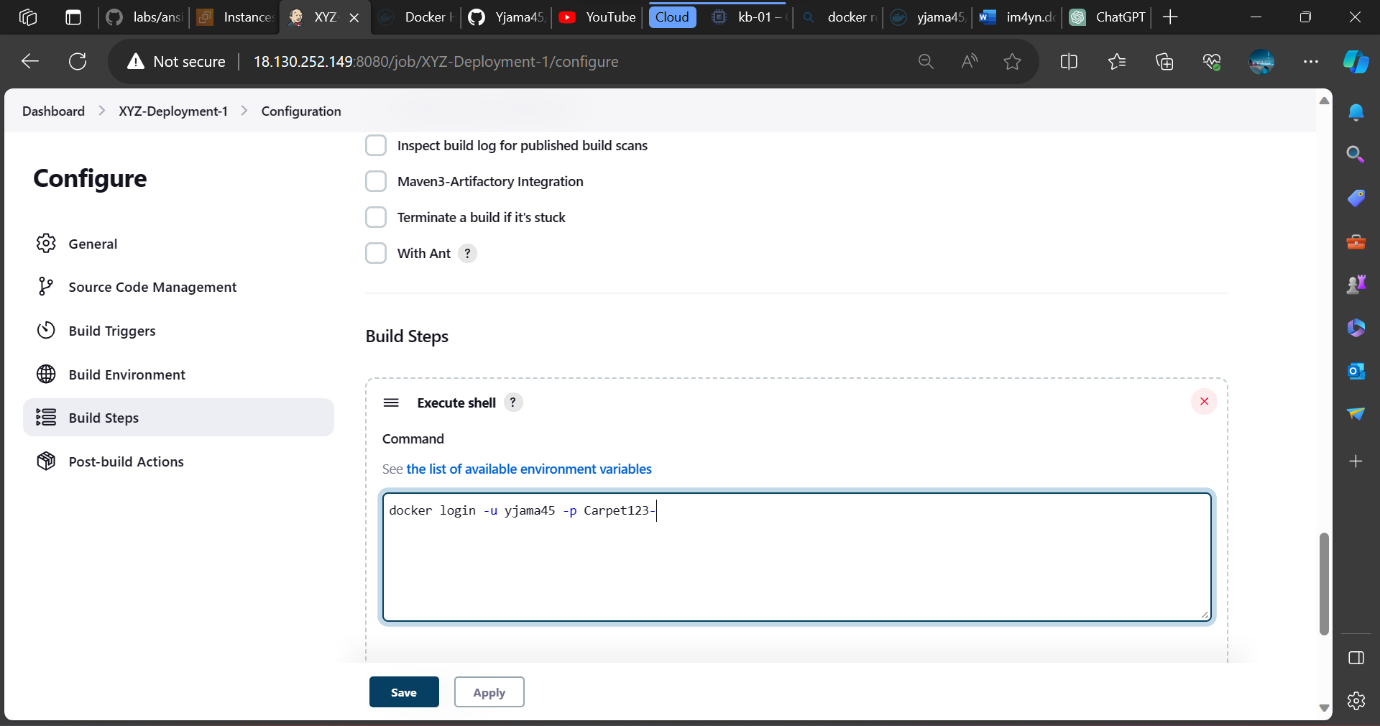
* The deployment for pod, service and deployment.

****

* Playbook created for deployment.

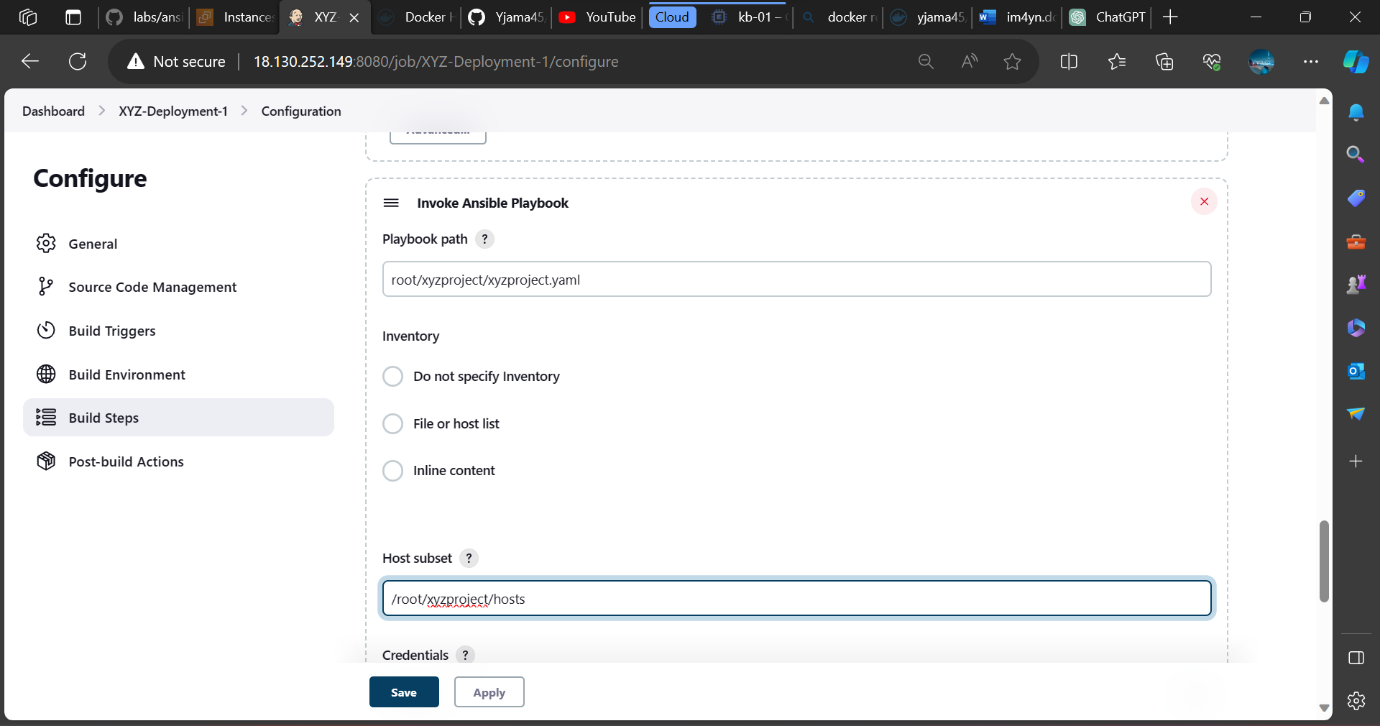
**Step 5: configure Jenkins**

****

****

**Step 6: Invoke ansible playbook**

****

****