**Certification** **Project I: Problem Statement**

AppleBite Co. is using Cloud for one of their products. The project uses modular components, multiple frameworks and want the components to be developed by different teams or by 3rd-party vendors.

The company’s goal is to deliver the product updates frequently to production with High quality & Reliability. They also want to accelerate software delivery speed, quality and reduce feedback time between developers and testers. As development progressed, they are facing multiple problems, because of various technologies involved in the project. Following are the problems:

•Building Complex builds is difficult

•Incremental builds are difficult to manage, and deploy

To solve these problems, they need to implement Continuous Integration & Continuous Deployment with DevOps using following tools:

**Git**–For version control for tracking changes in the code files

**Jenkins**–For continuous integration and continuous deployment

**Docker**–For deploying containerized applications

**Ansible**-Configuration management tools

This project will be about how to do deploy code to dev/stage/prod etc, just on a click of button. Link for the sample PHP application: <https://github.com/edureka-devops/projCert.git>

**Business challenge/requirement**

As soon as the developer pushes the updated code on the GIT master branch, a new test server should be provisioned with all the required software. Post this, the code should be containerized and deployed on the test server. The deployment should then be built and pushed to the prod server.

All this should happen automatically and should be triggered from a push to the GitHub master branch.

**Steps for executing the solution**:

•Use the Master VM for Jenkins, Ansible, GIT etc.

•Use the fresh instance for Jenkins Slave Node (Test Server)

•Change the IP address of the VMs accordingly

•Add Build Pipeline Plugin and Post-build task plugin to Jenkins on the master VM

•Install python, openssh-server and git on the slave node manually

•Use the image devopsedu/webapp and add your PHP website to it using a Dockerfile

•Push the PHP website, and the Dockerfile to a git repository

**Below tasks should be automated through Jenkins by creating a pipeline:**

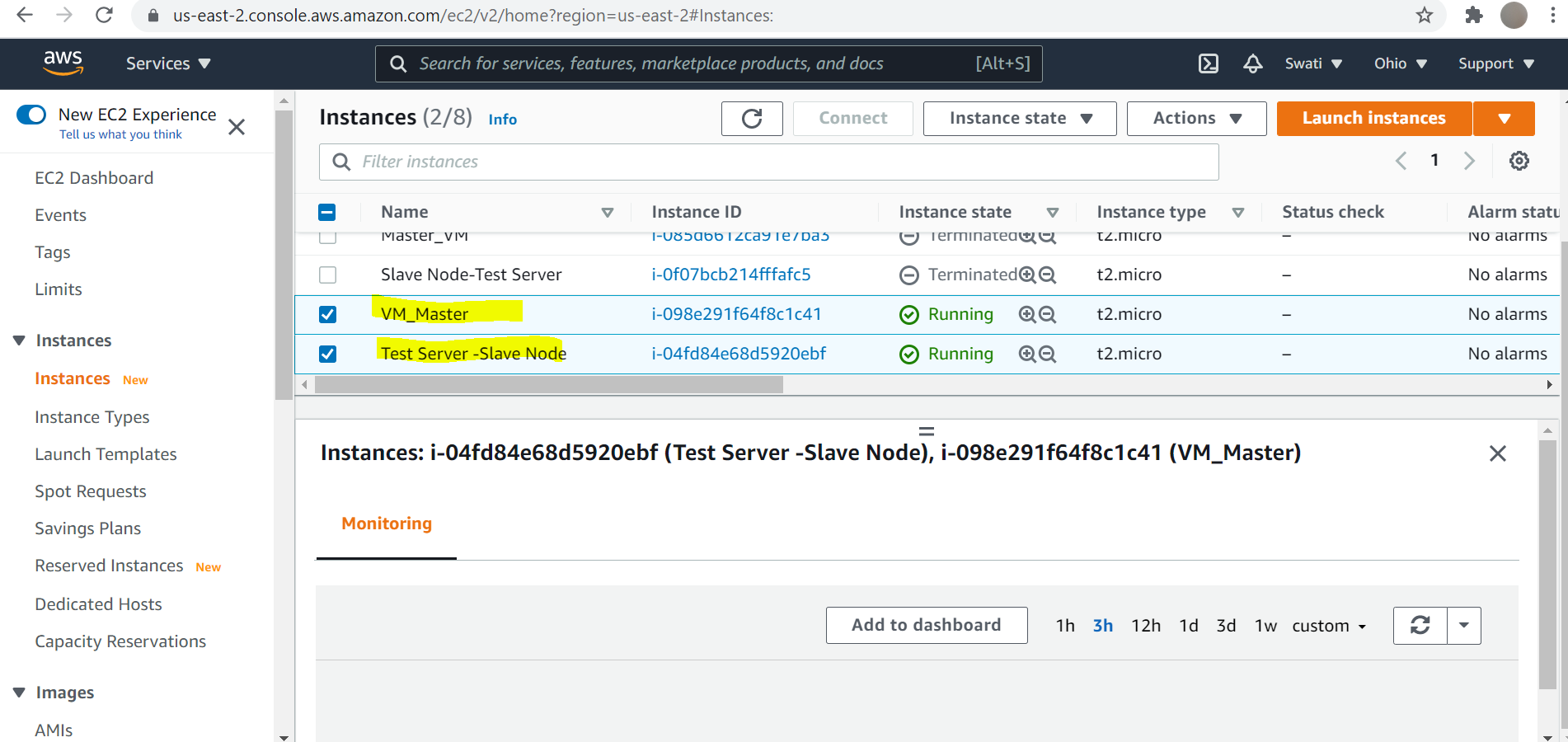
1. Install and configure puppet agent on the slave node (Job 1)

2.Push an Ansible configuration on test server to install docker (Job 2)

3.Pull the PHP website, and the Dockerfile from the git repo and build and deploy your PHP docker container. After. (Job 3)

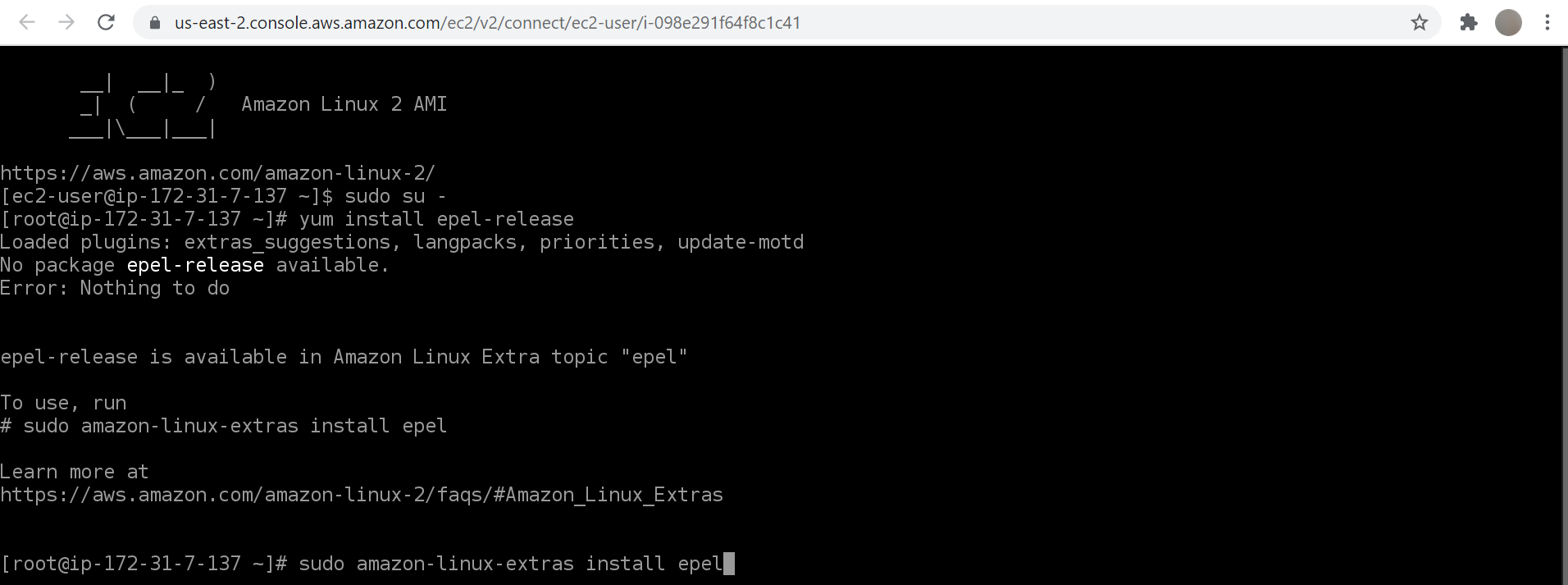
4.If Job 3 fails, delete the running container on Test Server

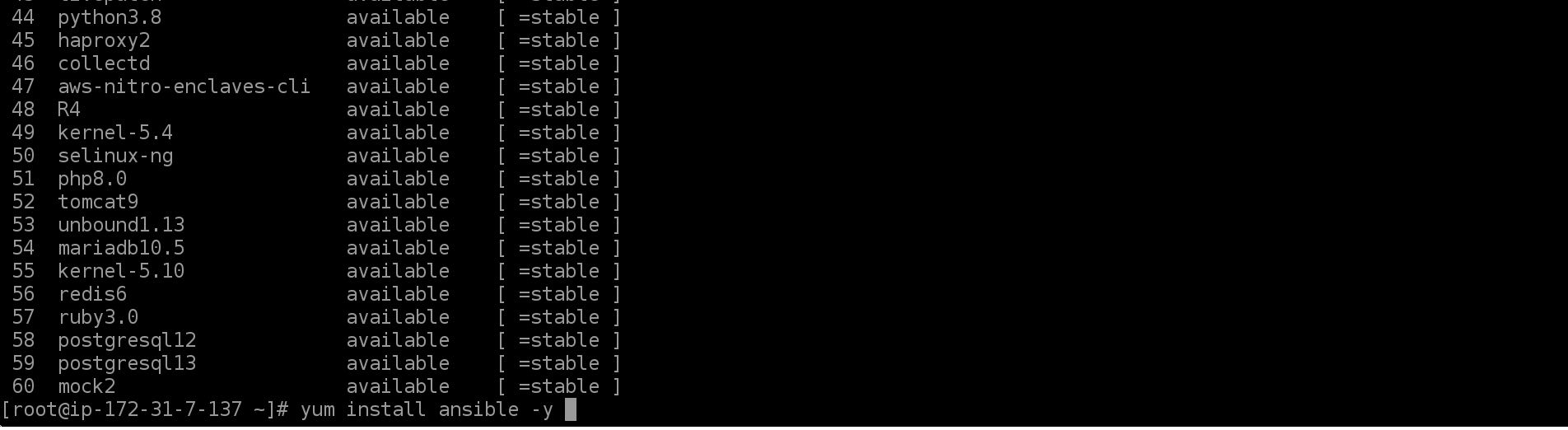
New instances for **Master VM** and **Slave Node (Test Server)** launched.

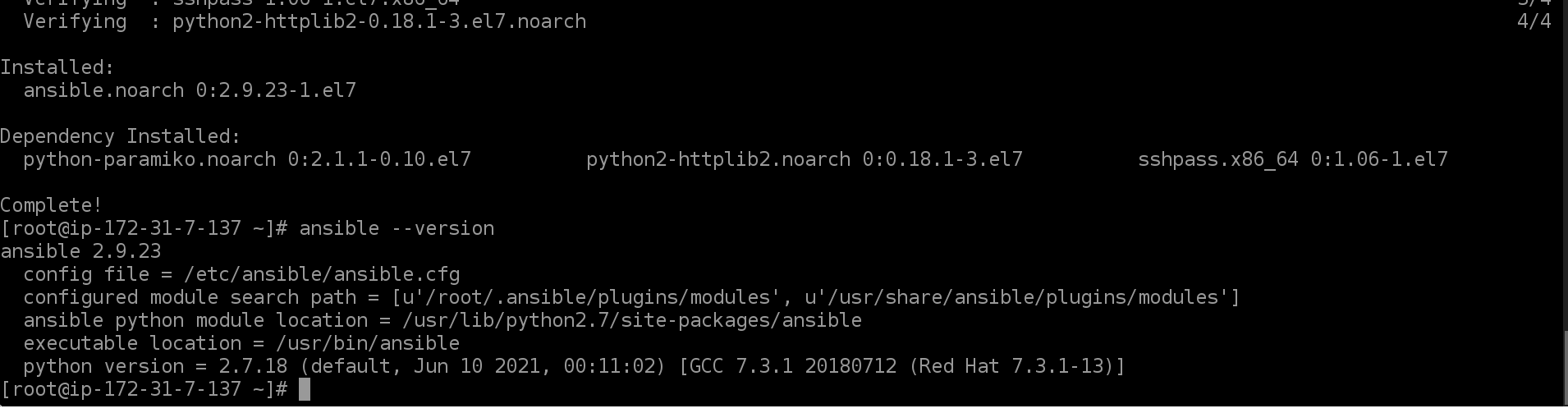


Installed Ansible, Git and Jenkins on master VM manually.

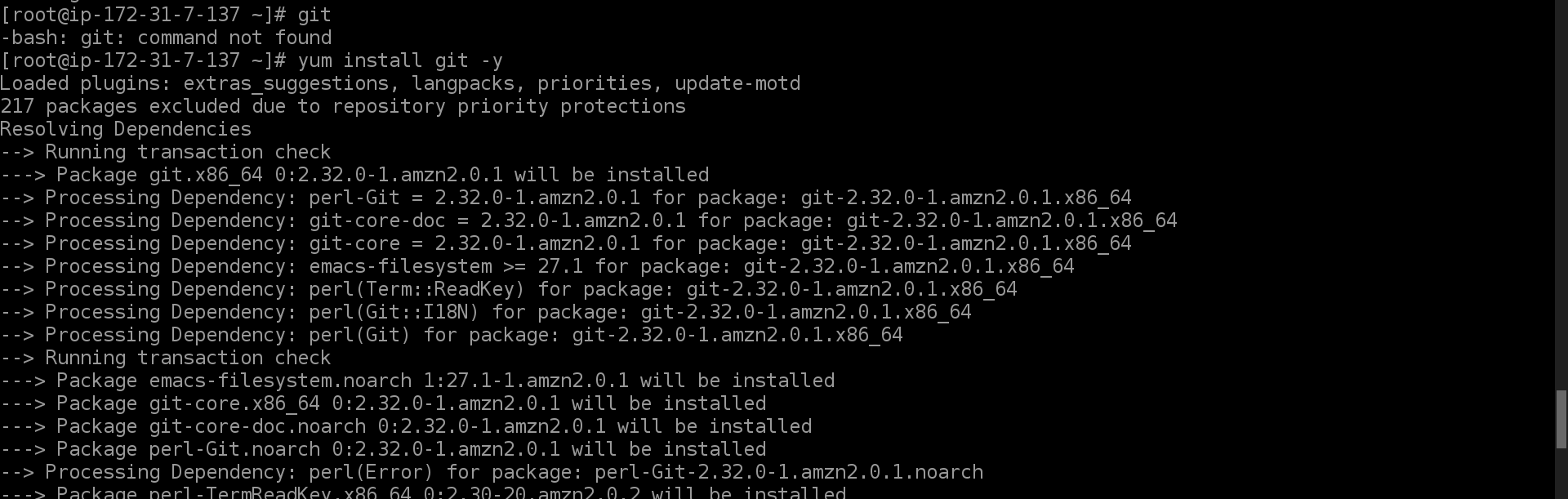
Ansible :

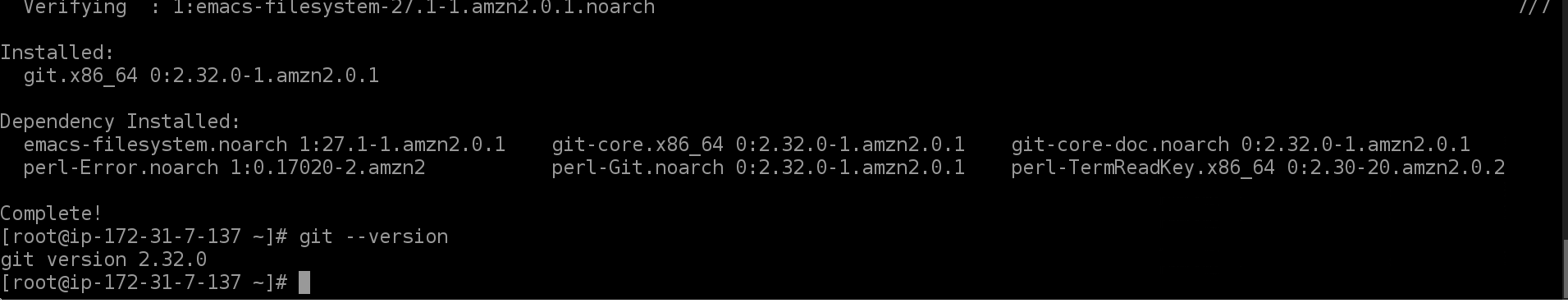






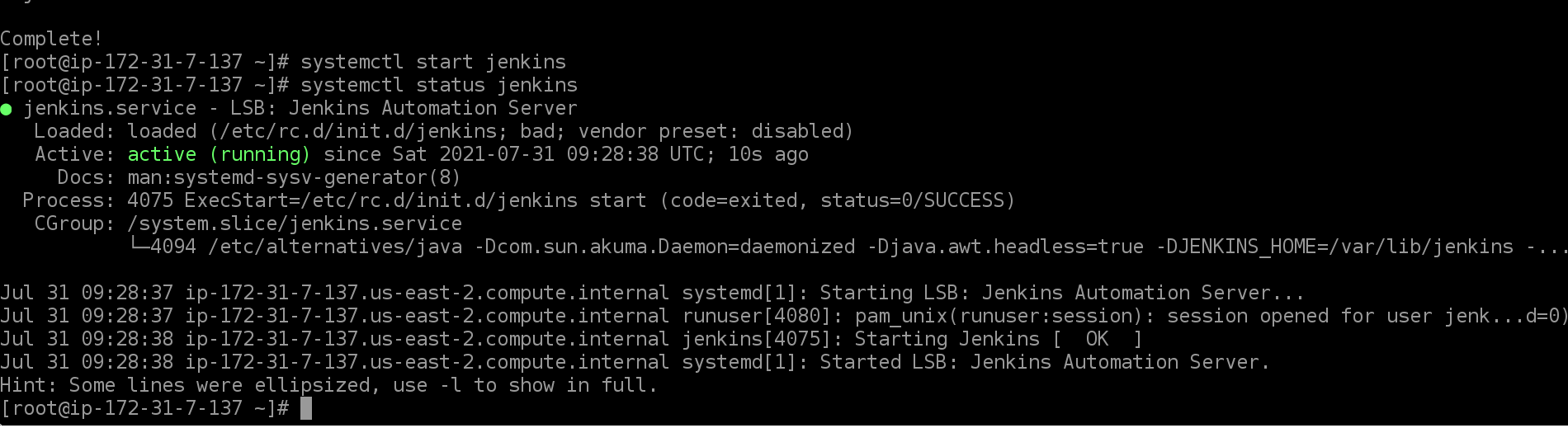
Git :

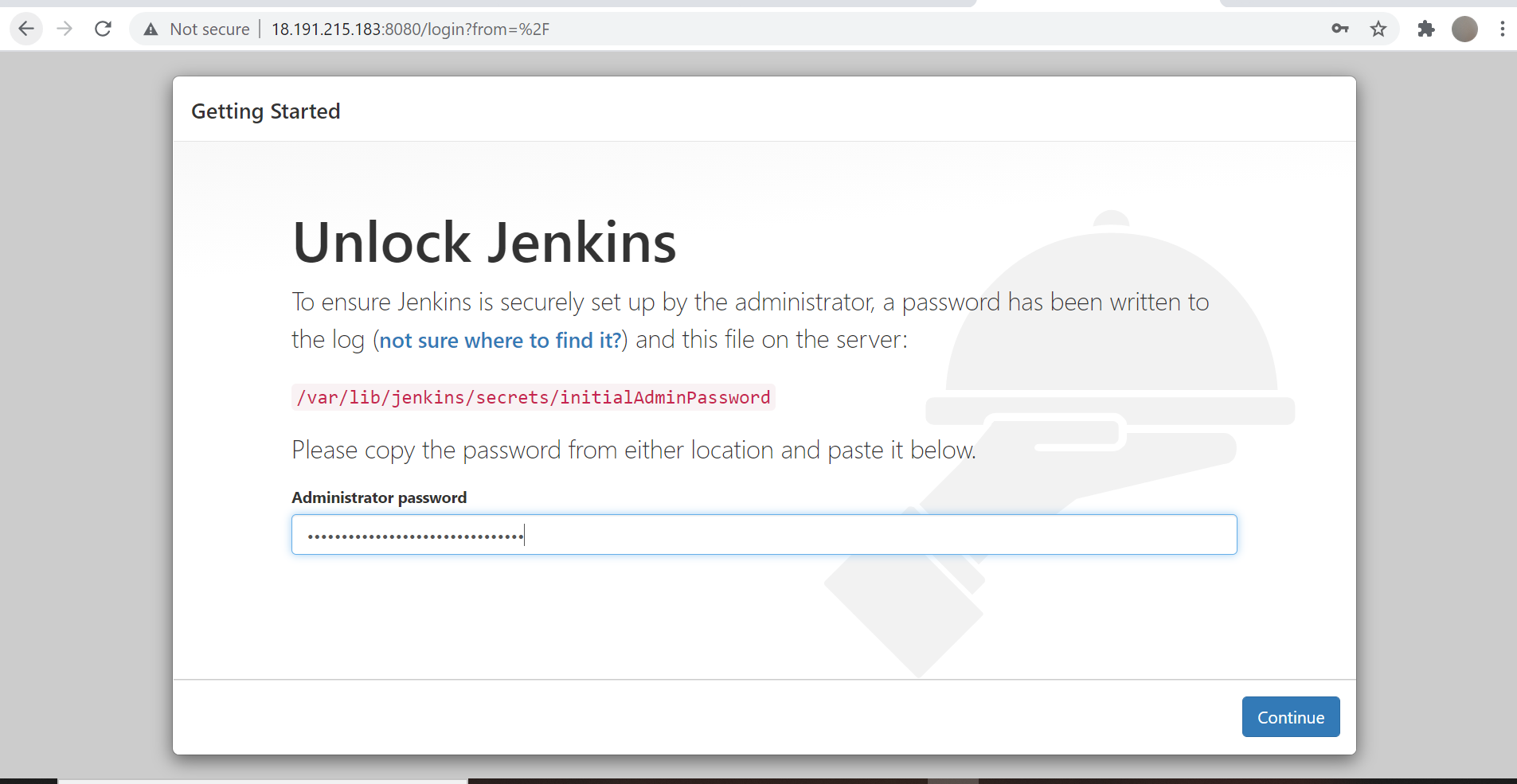


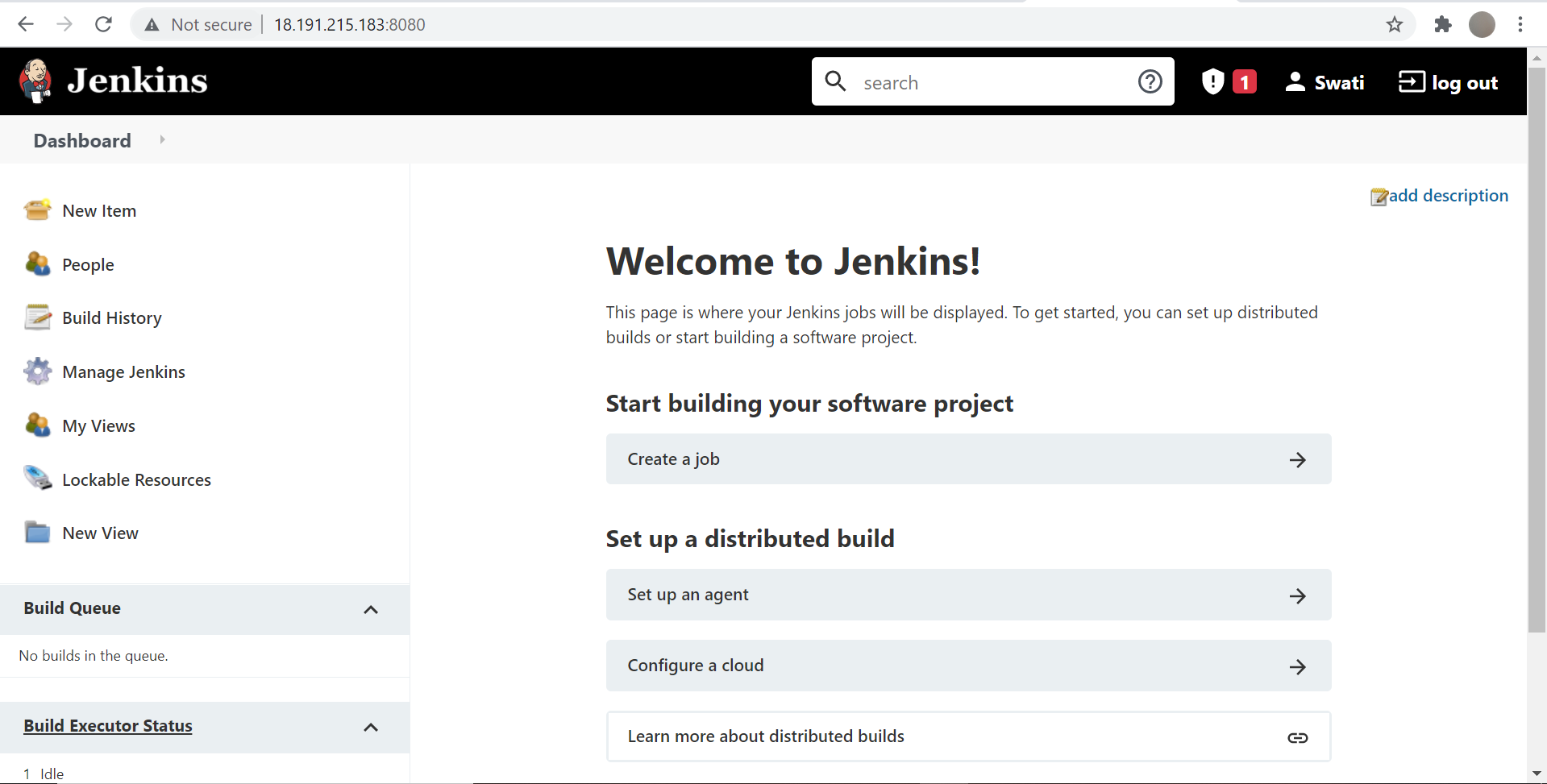


Jenkins : Perquisite- java should be installed to run Jenkins on instance server.





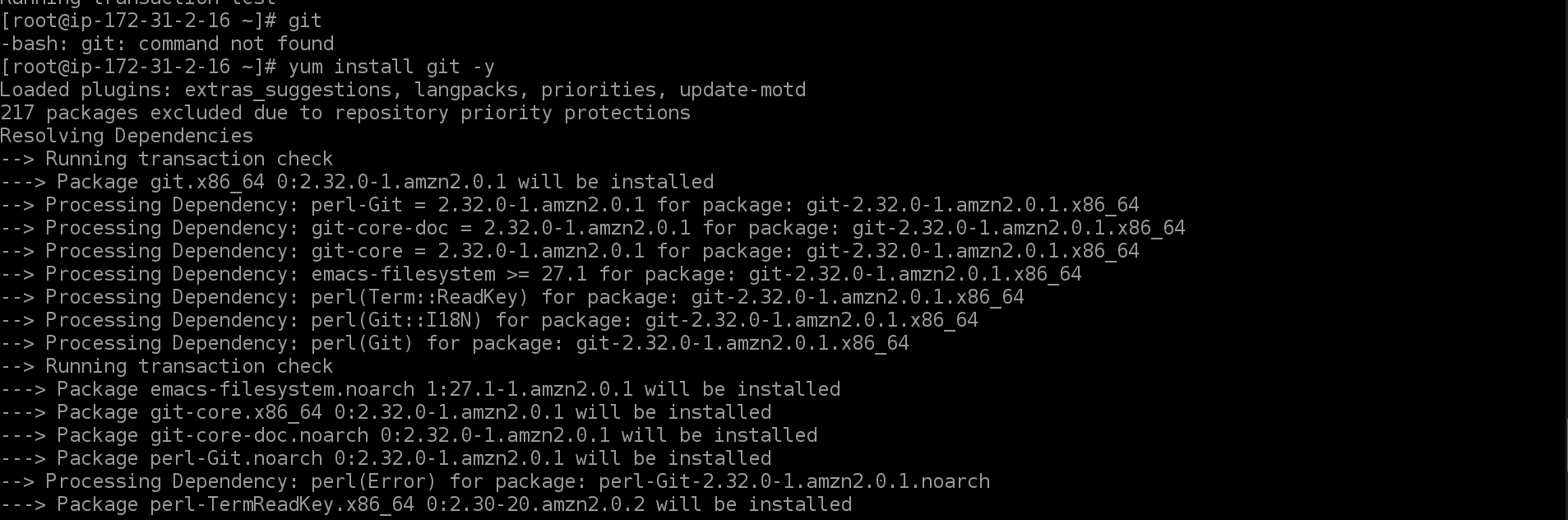


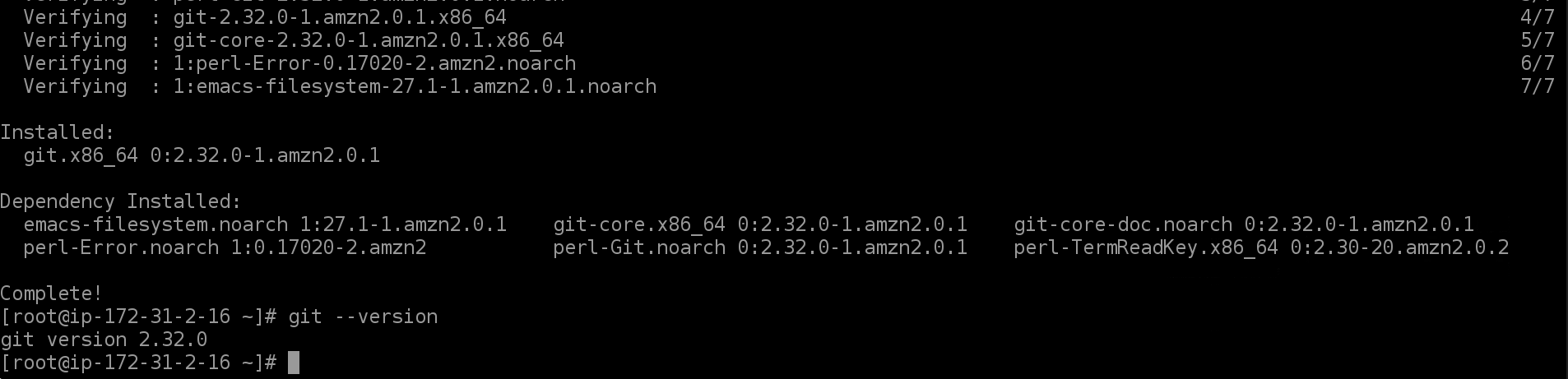


Port number of Jenkins changed from 8080 to 9090.

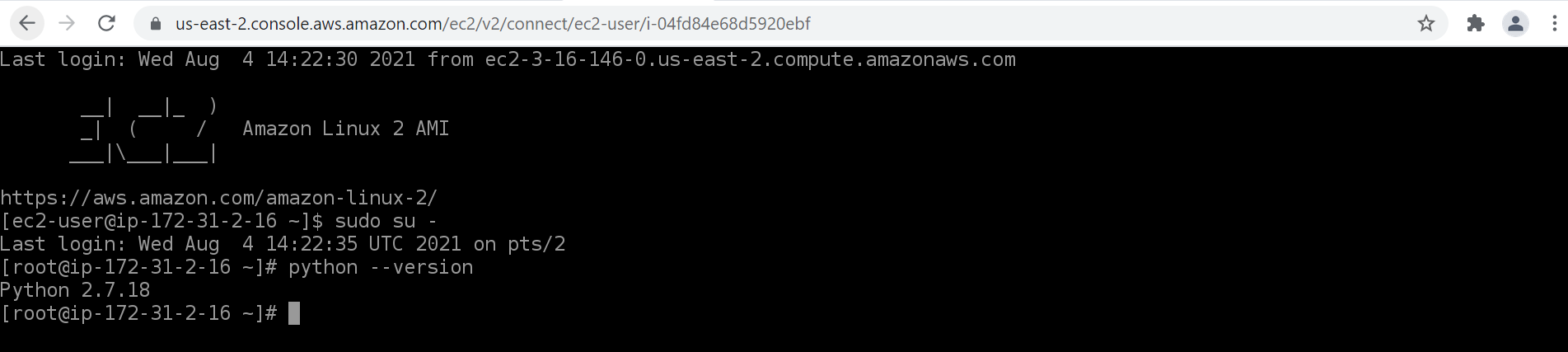
Install git and python on slave node- test server manually.

Git

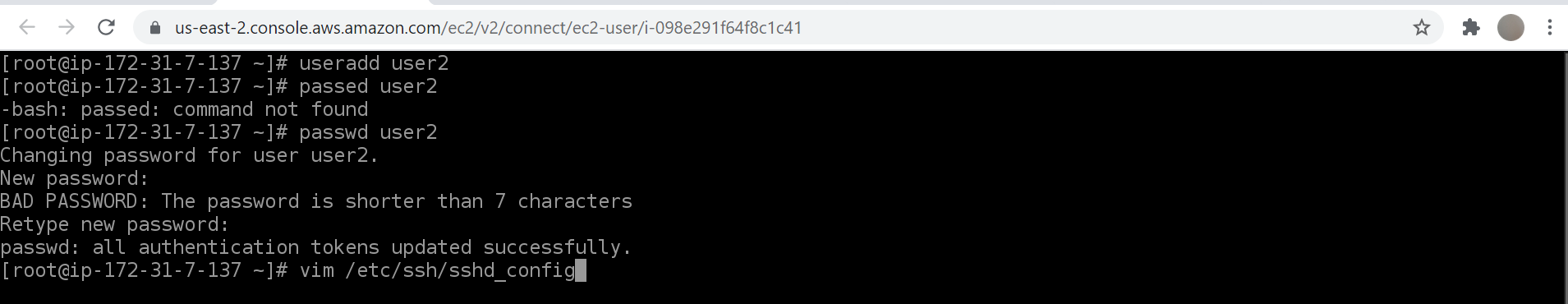


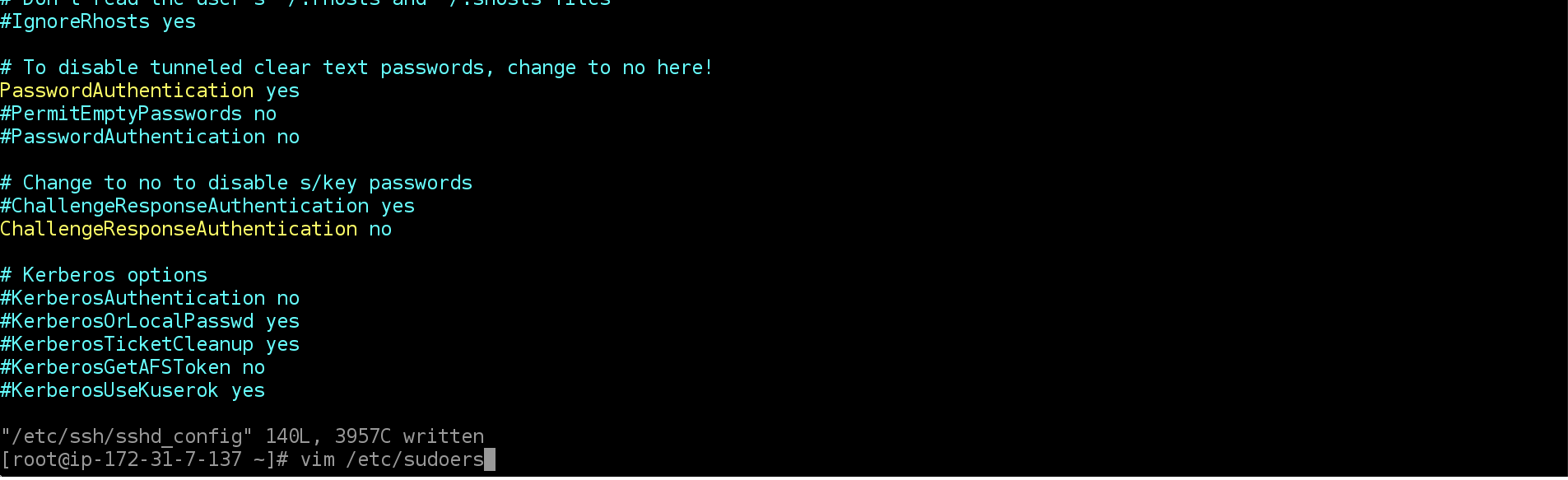


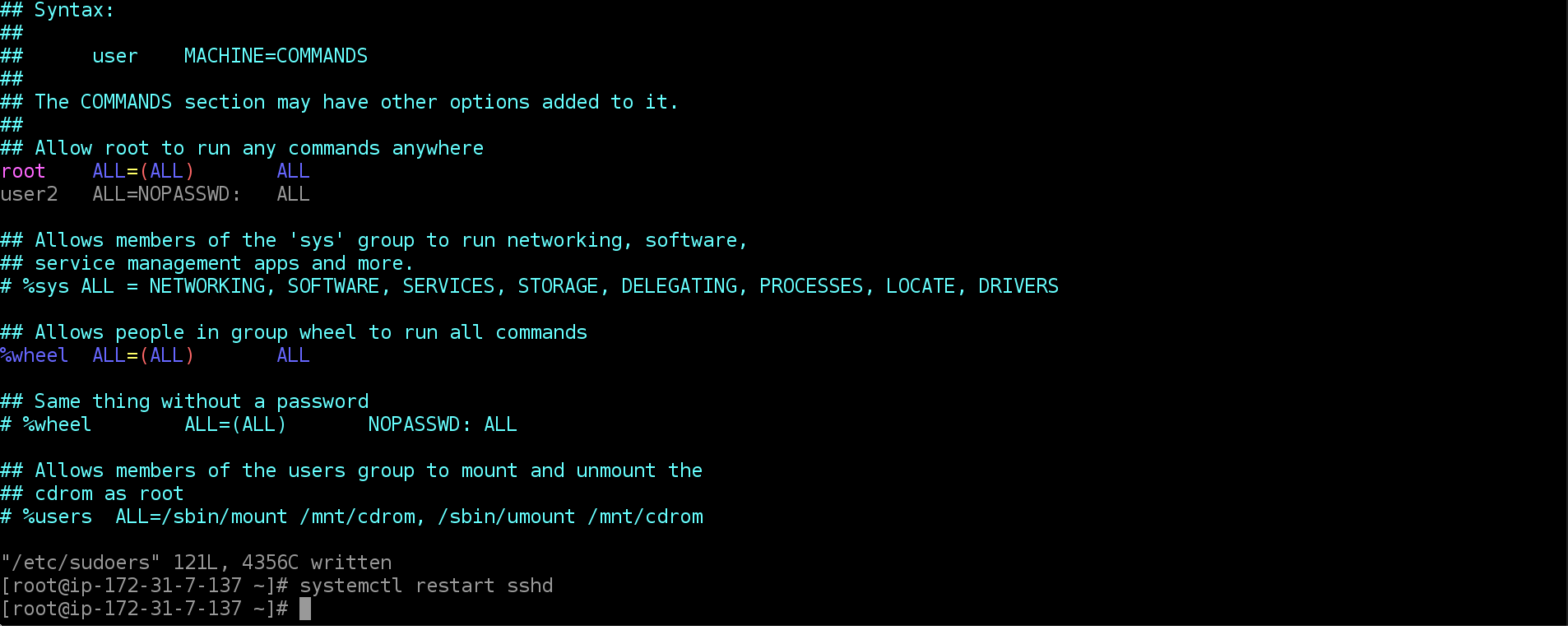
Python is already installed on test server

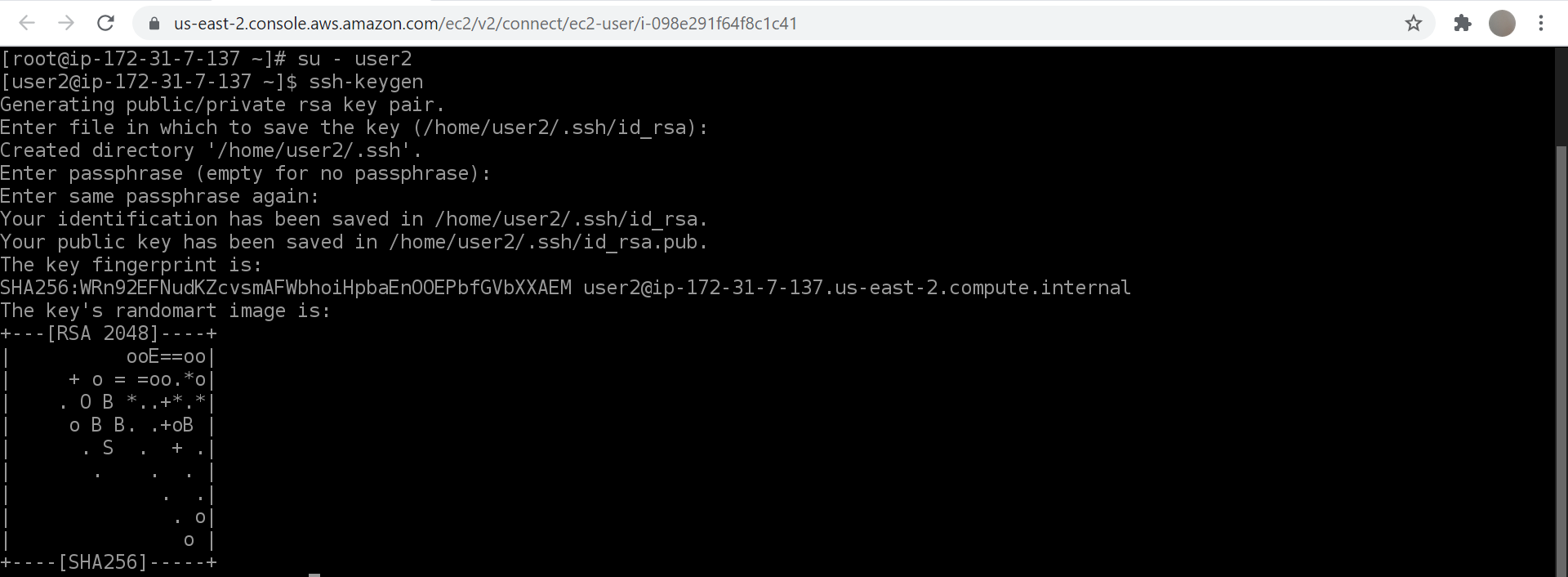


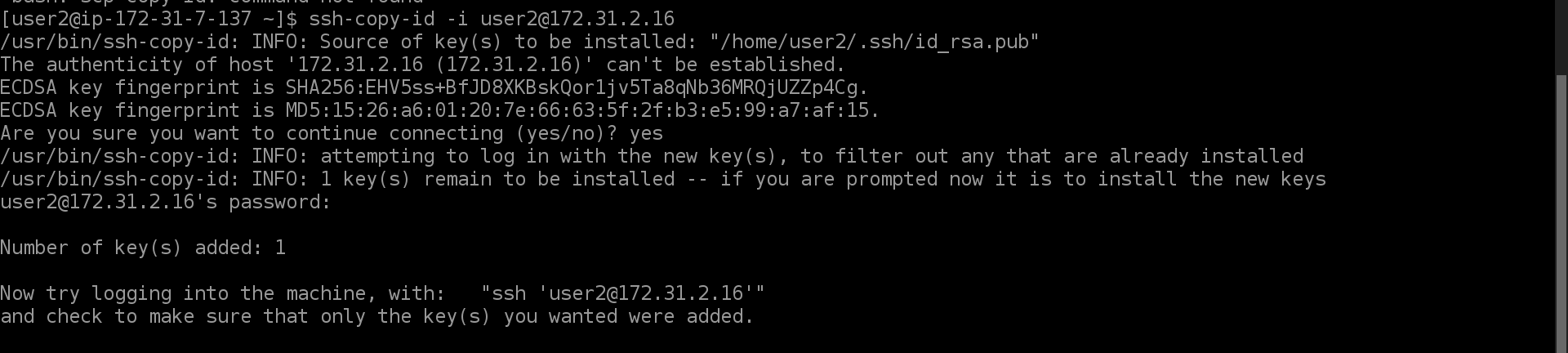
Establish/open ssh connection in between Master VM and Slave node (test server)











Above given steps are followed on both the instances to open ssh connection:

1) user2 created and password generated for this user -user2

2) password authentication set yes

3) user2 added in sudoers file

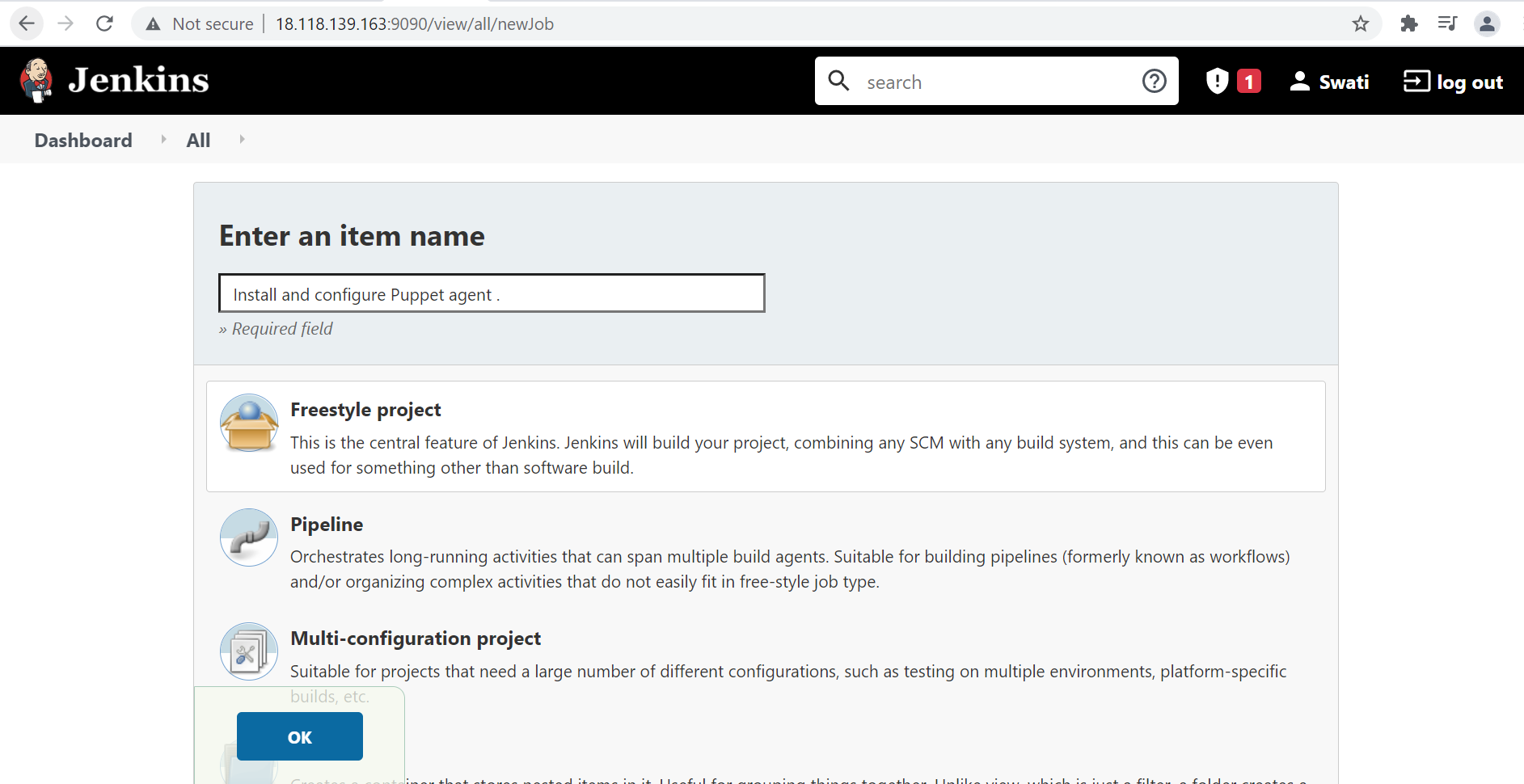
4) restarted sshd

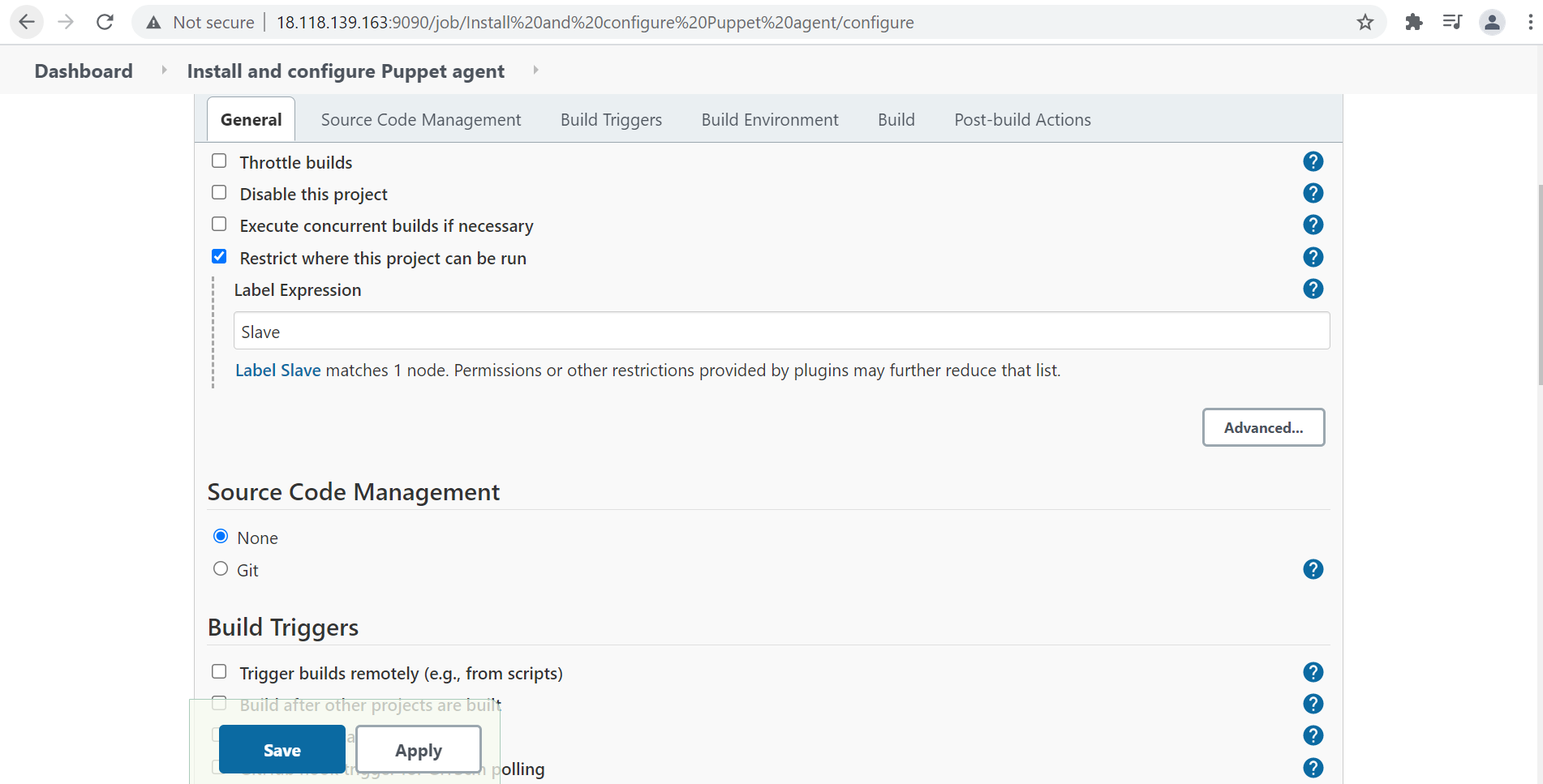
5) setup ssh connection

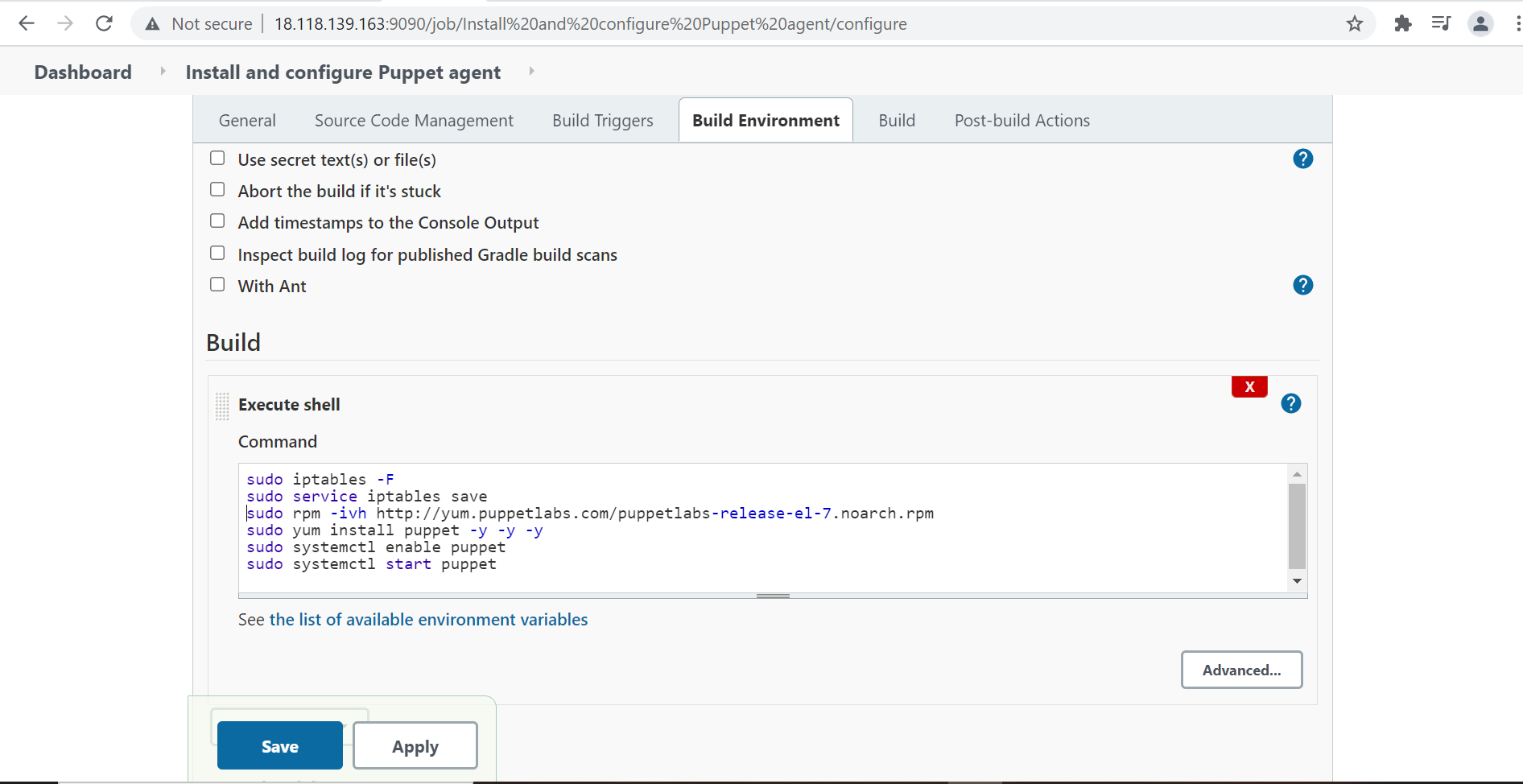
Sample PHP application <https://github.com/edureka-devops/projCert.git> forked to Git repository <https://github.com/deveops-engineer/projCert.git>

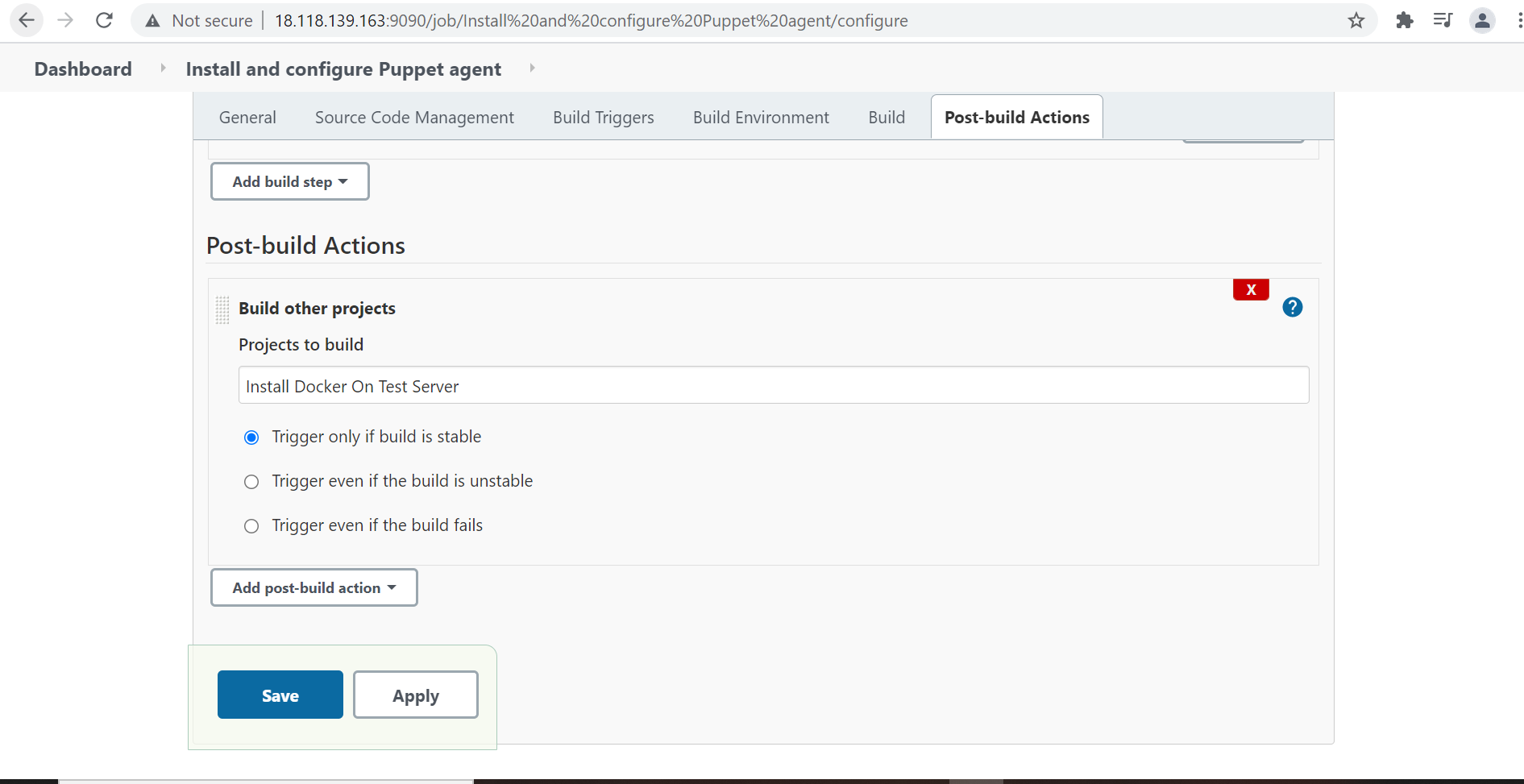
Below given tasks automated through Jenkins jobs given below :

**Job 1** : Install and configure puppet agent on the slave node.

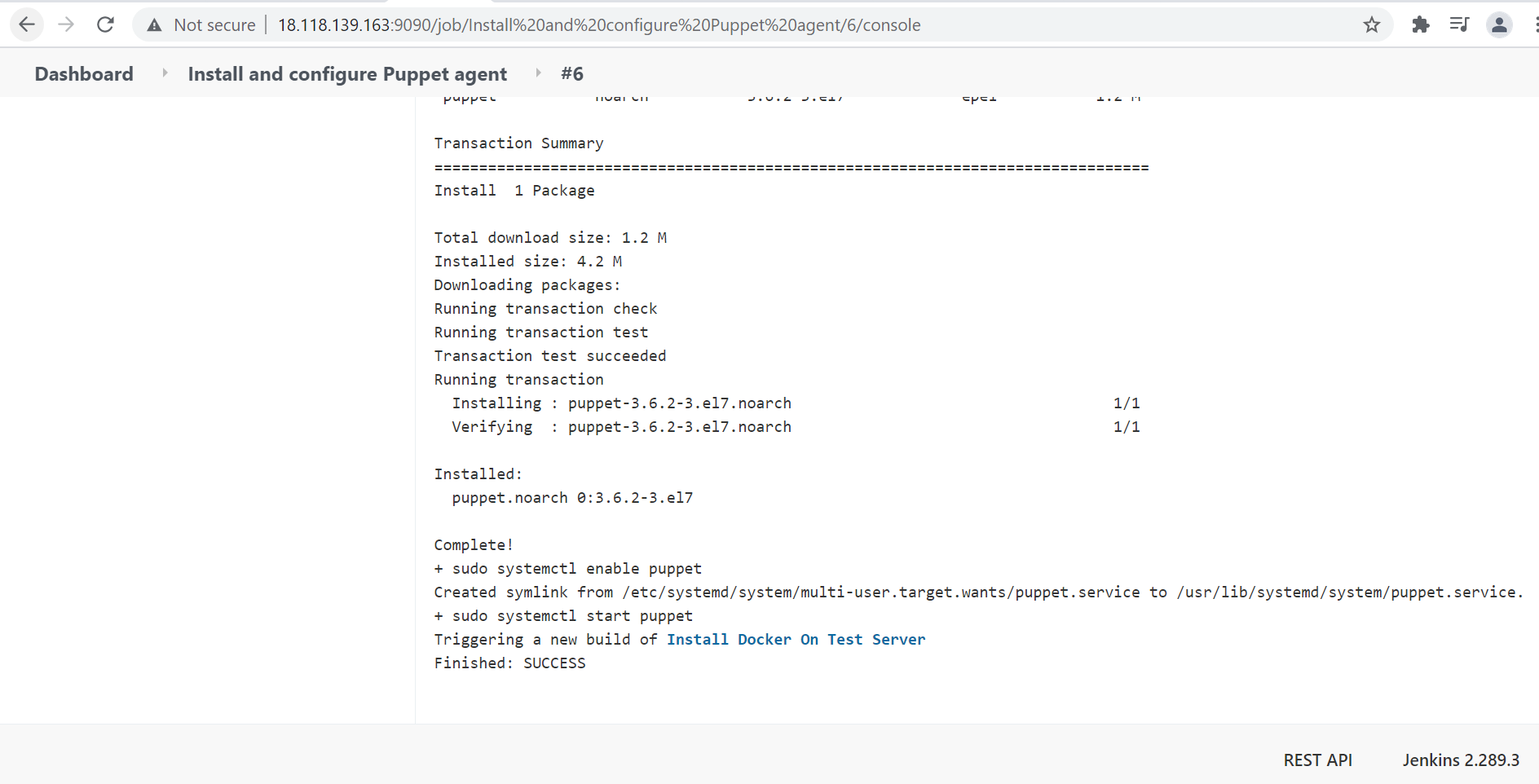




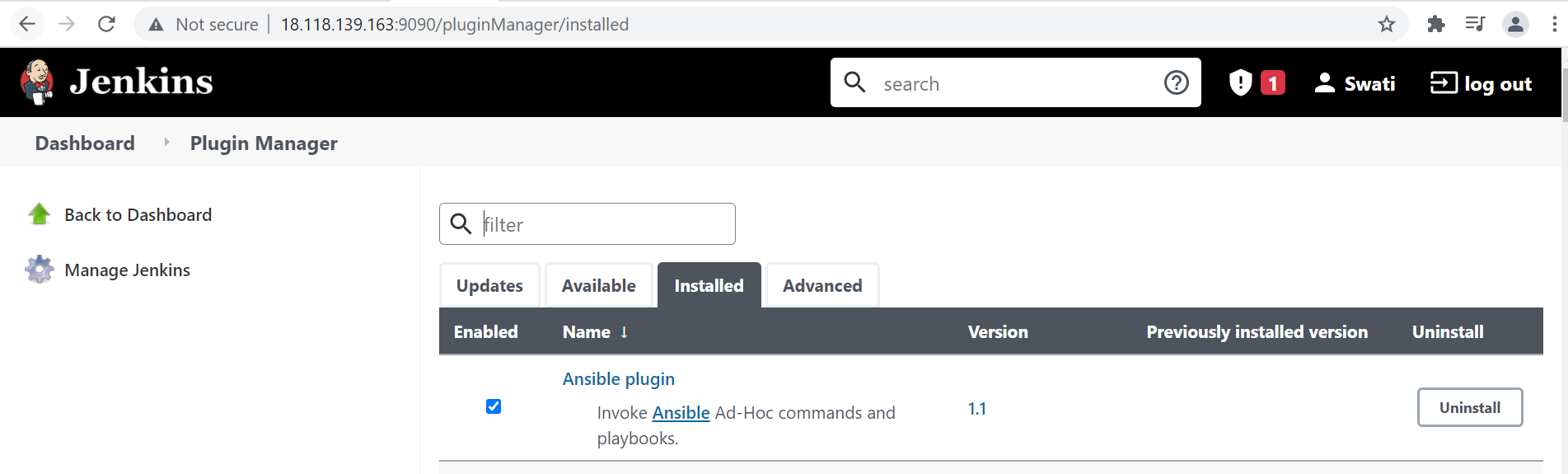


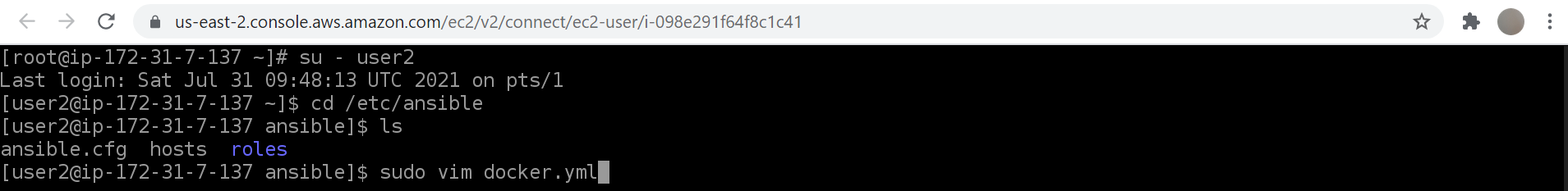


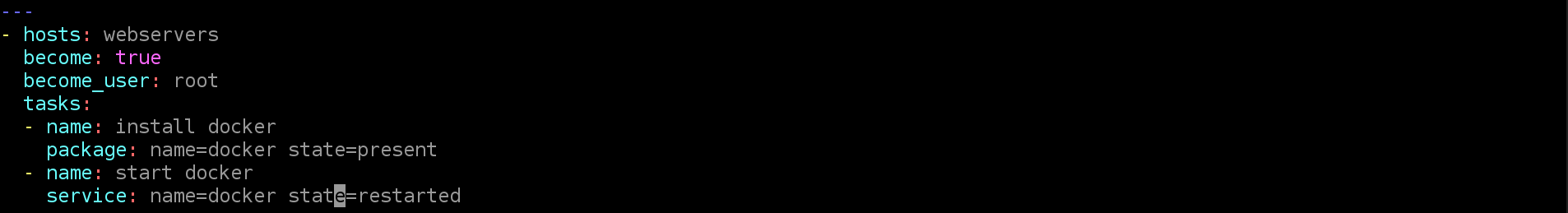
Output :-

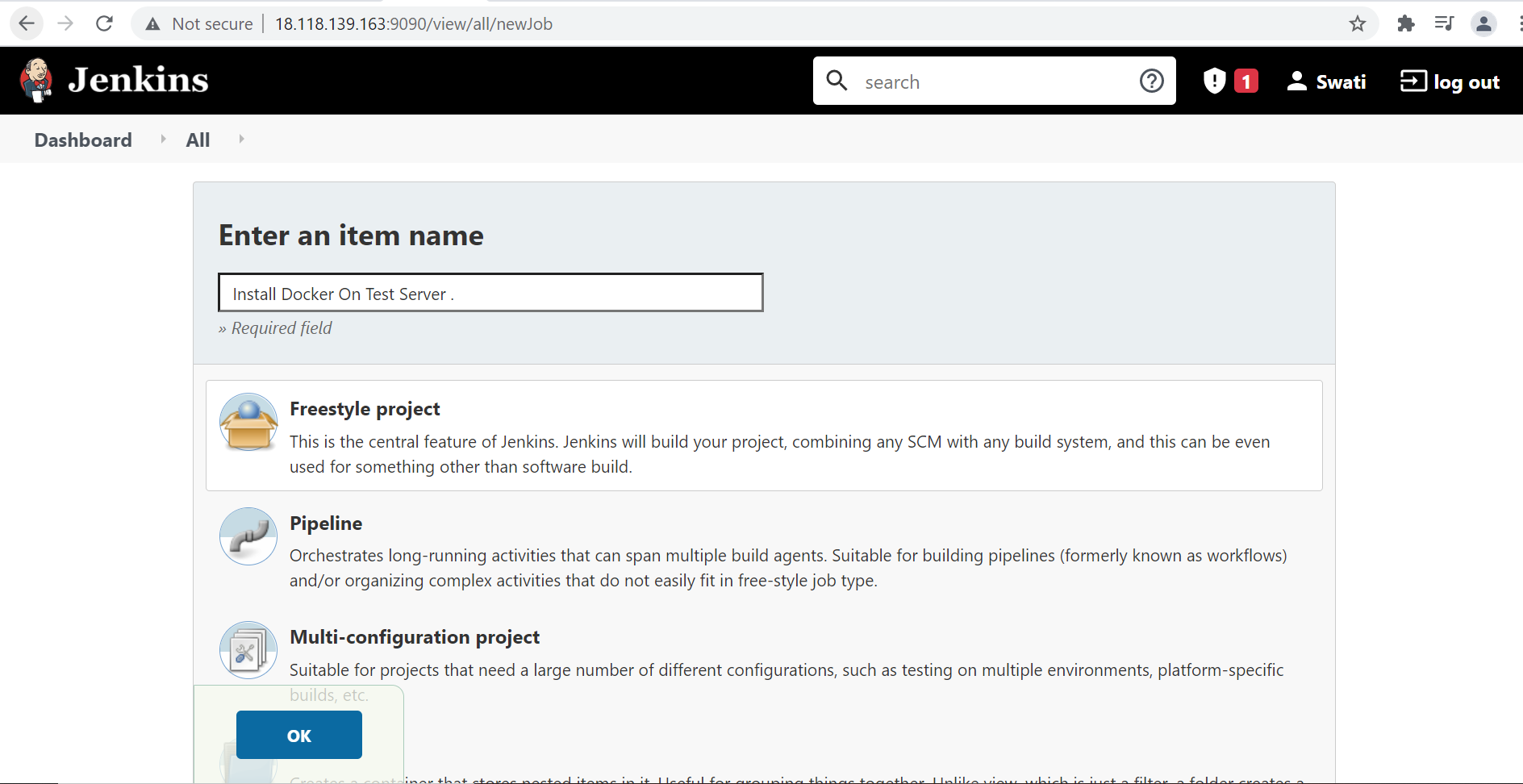


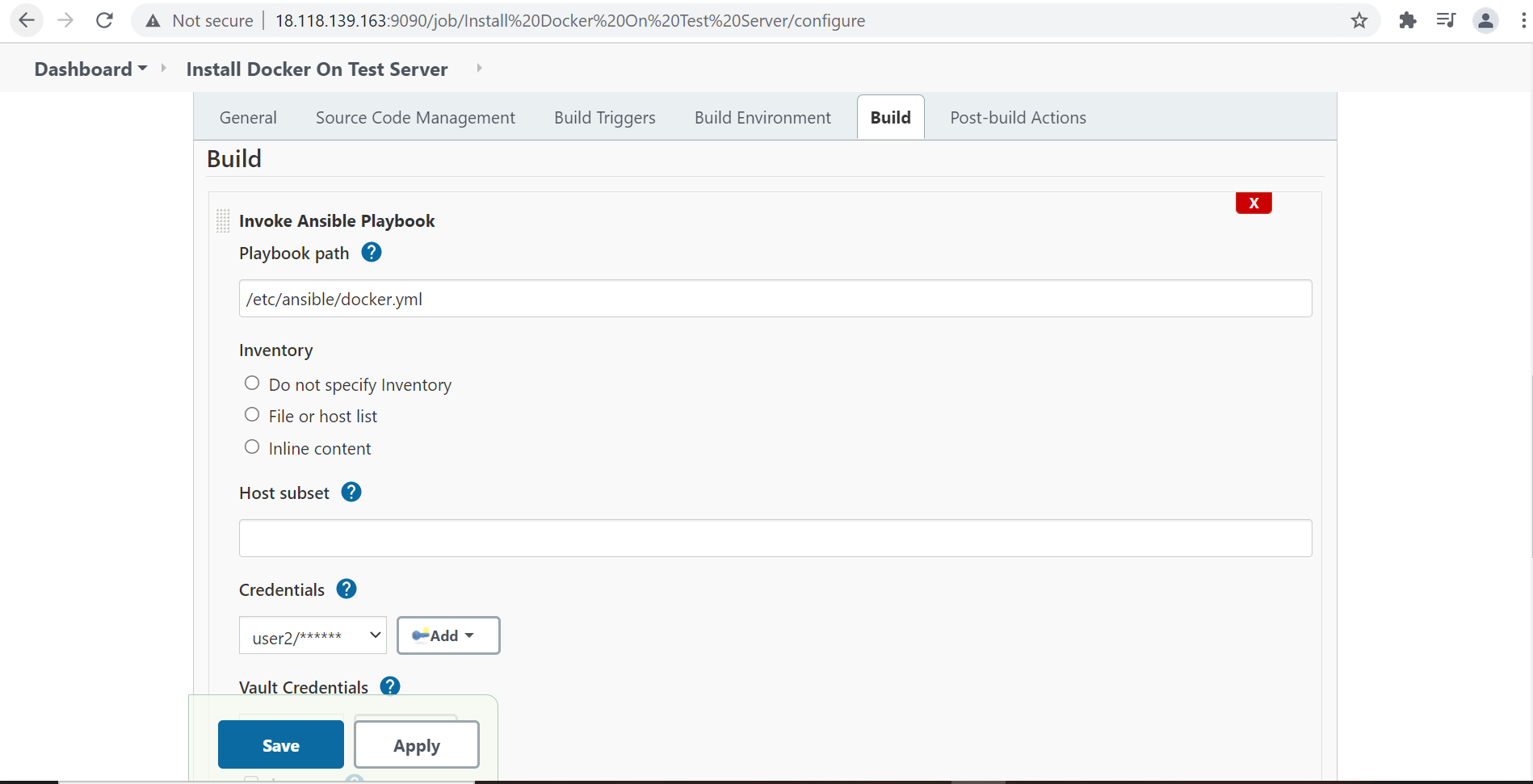
**Job 2**: Push an Ansible Configuration on test server to install docker

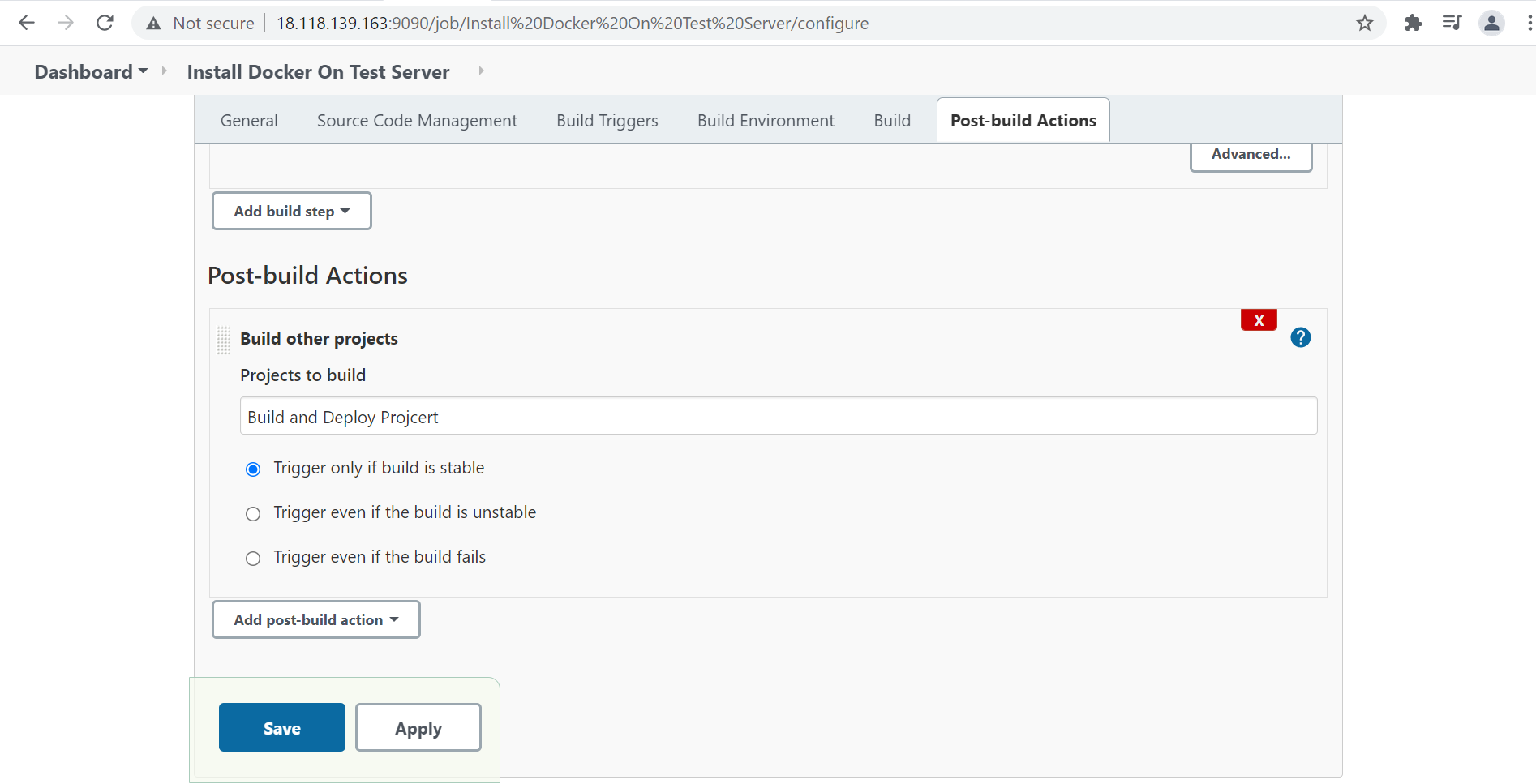




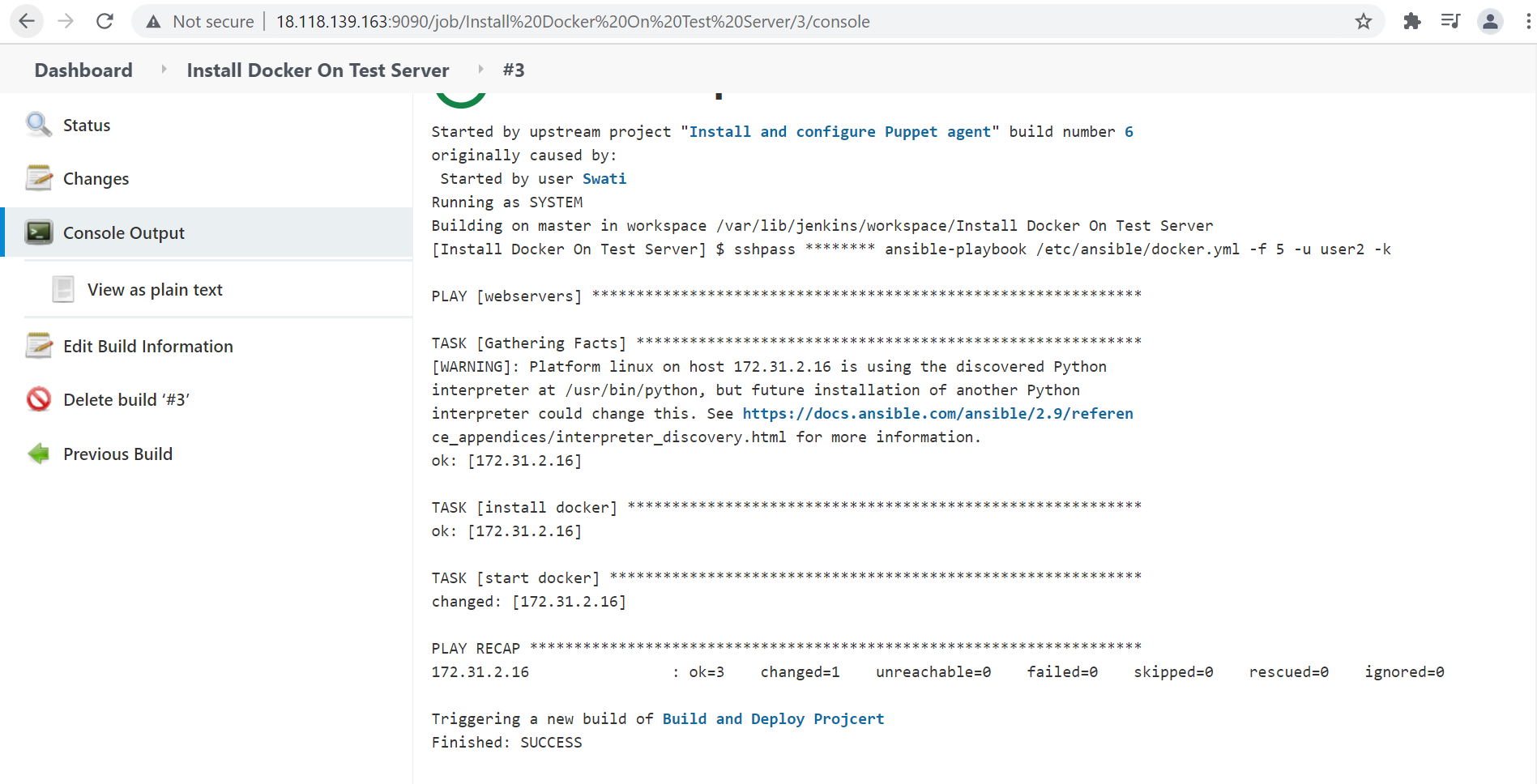


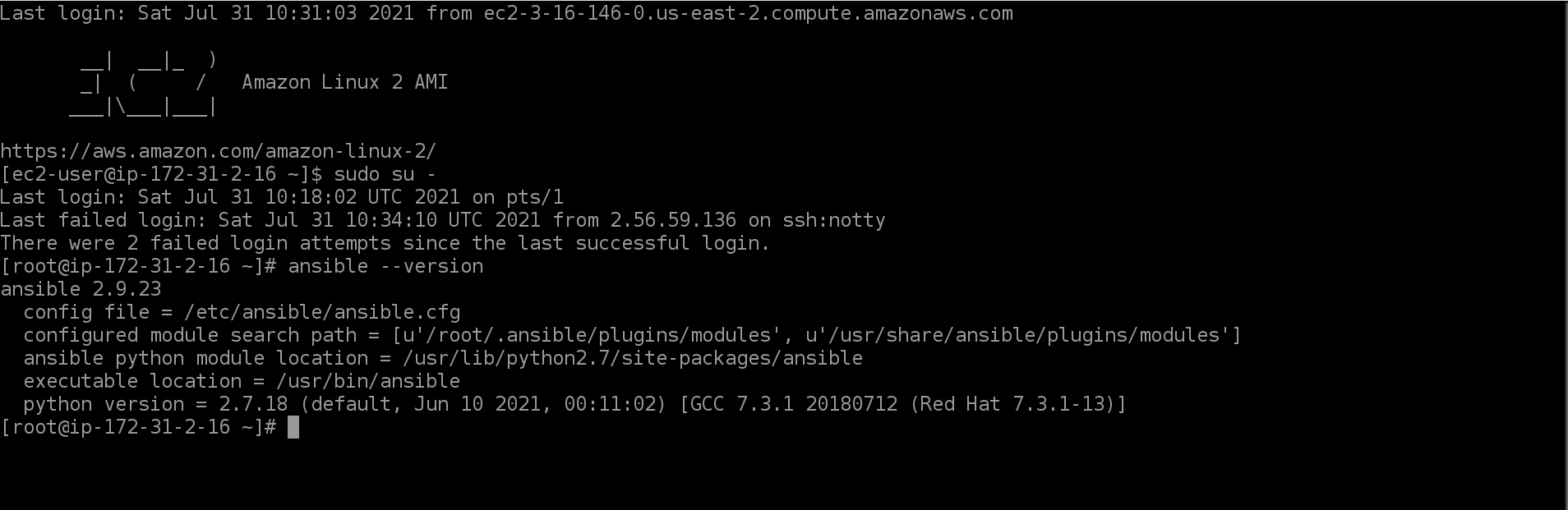




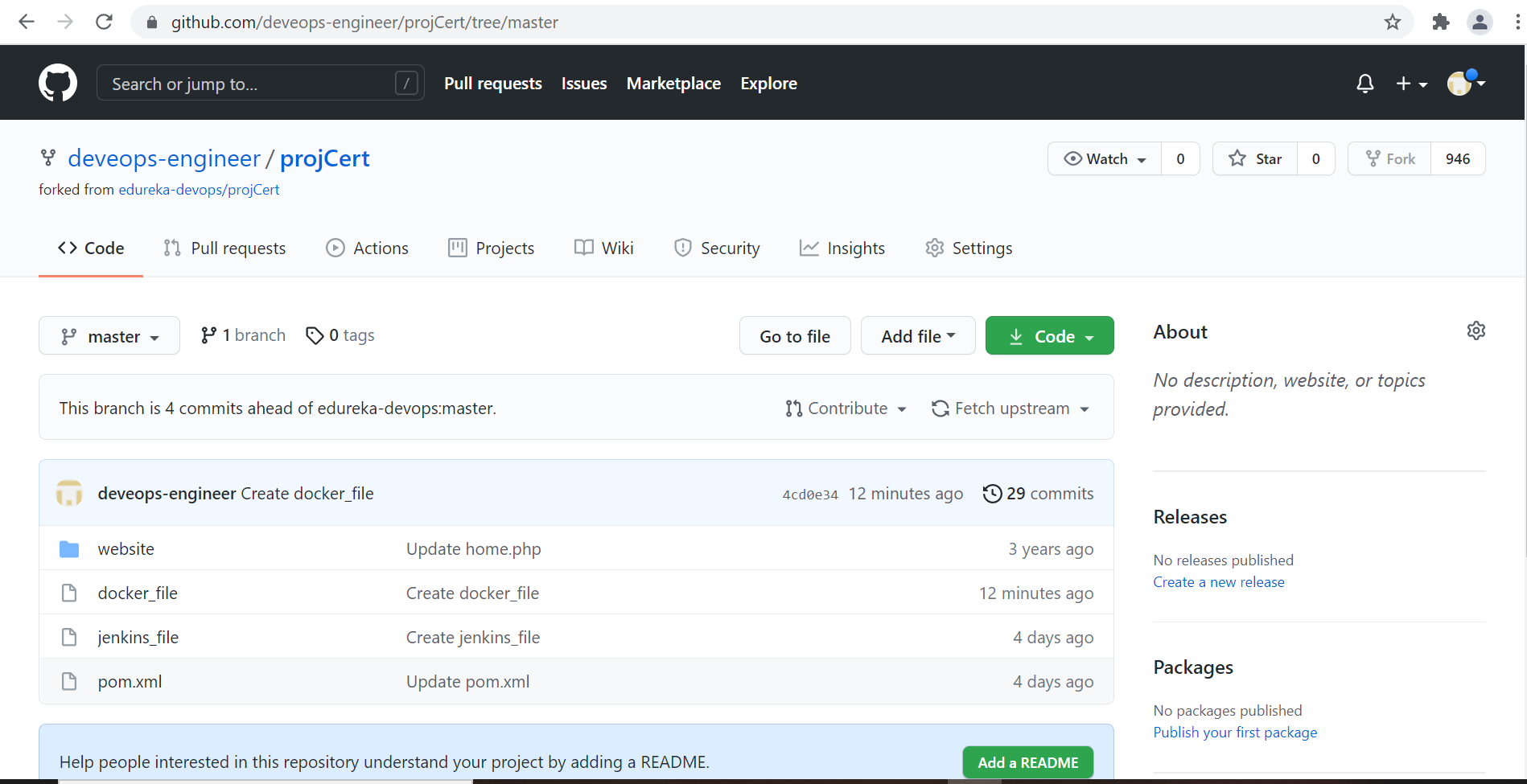


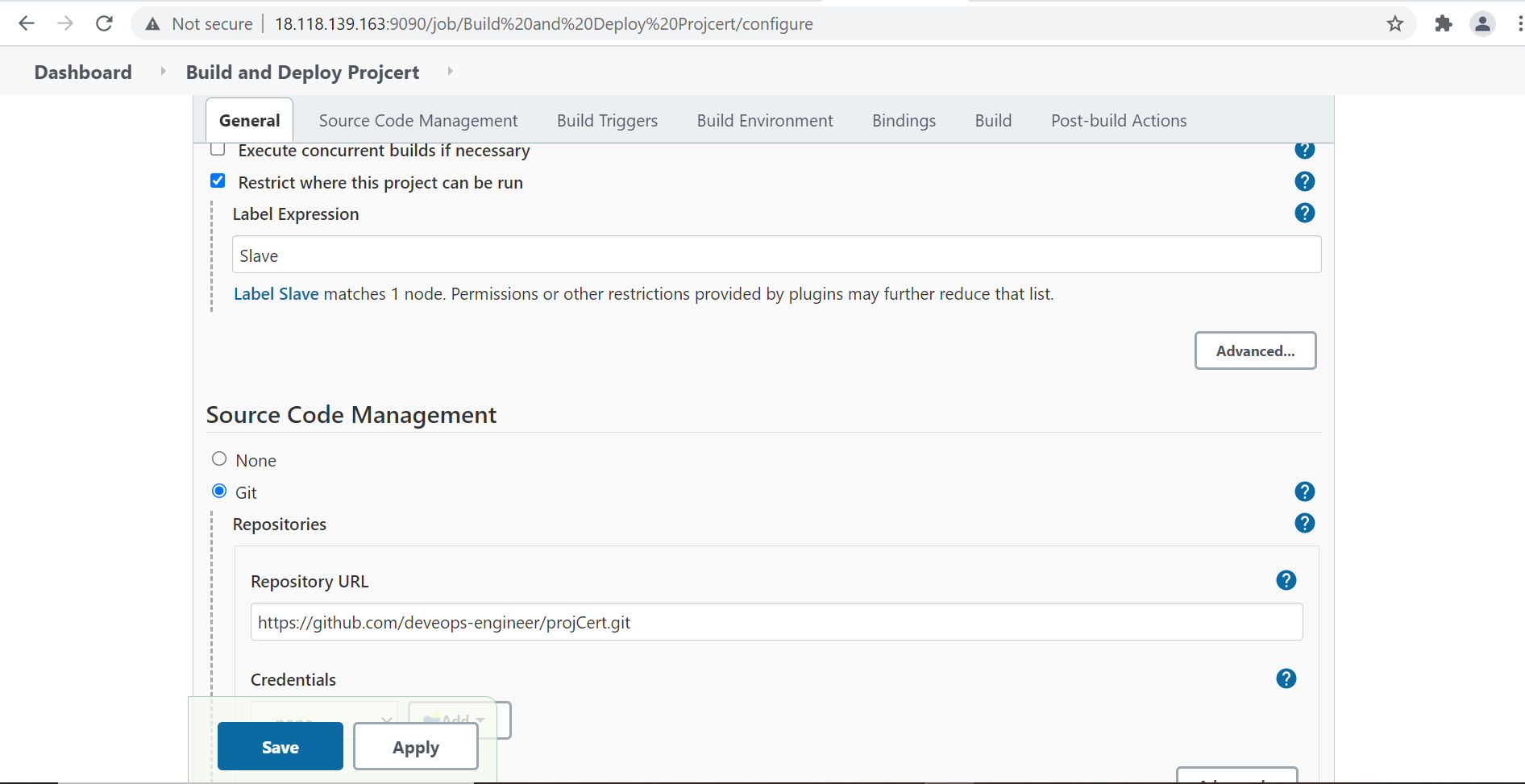
Output :-

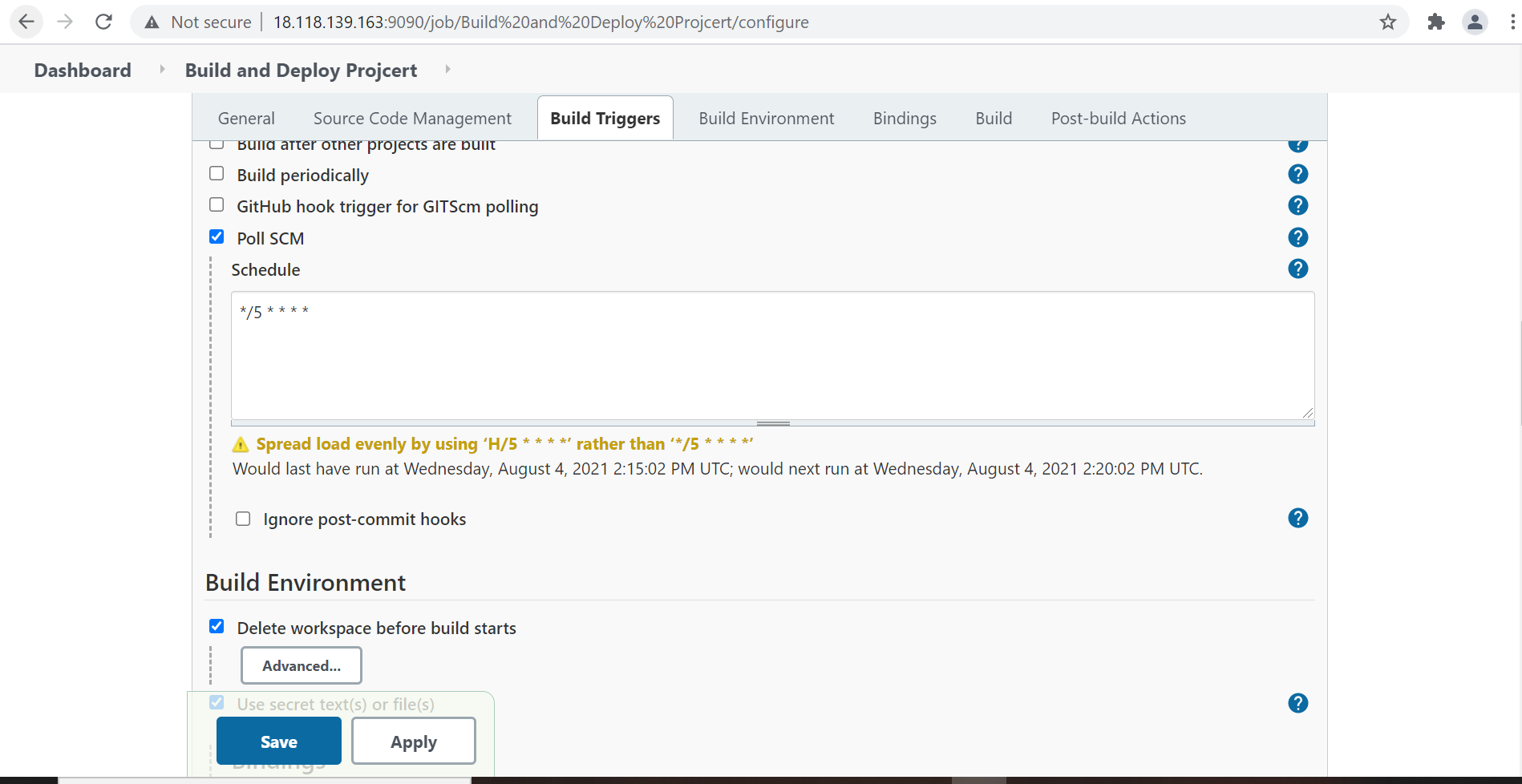


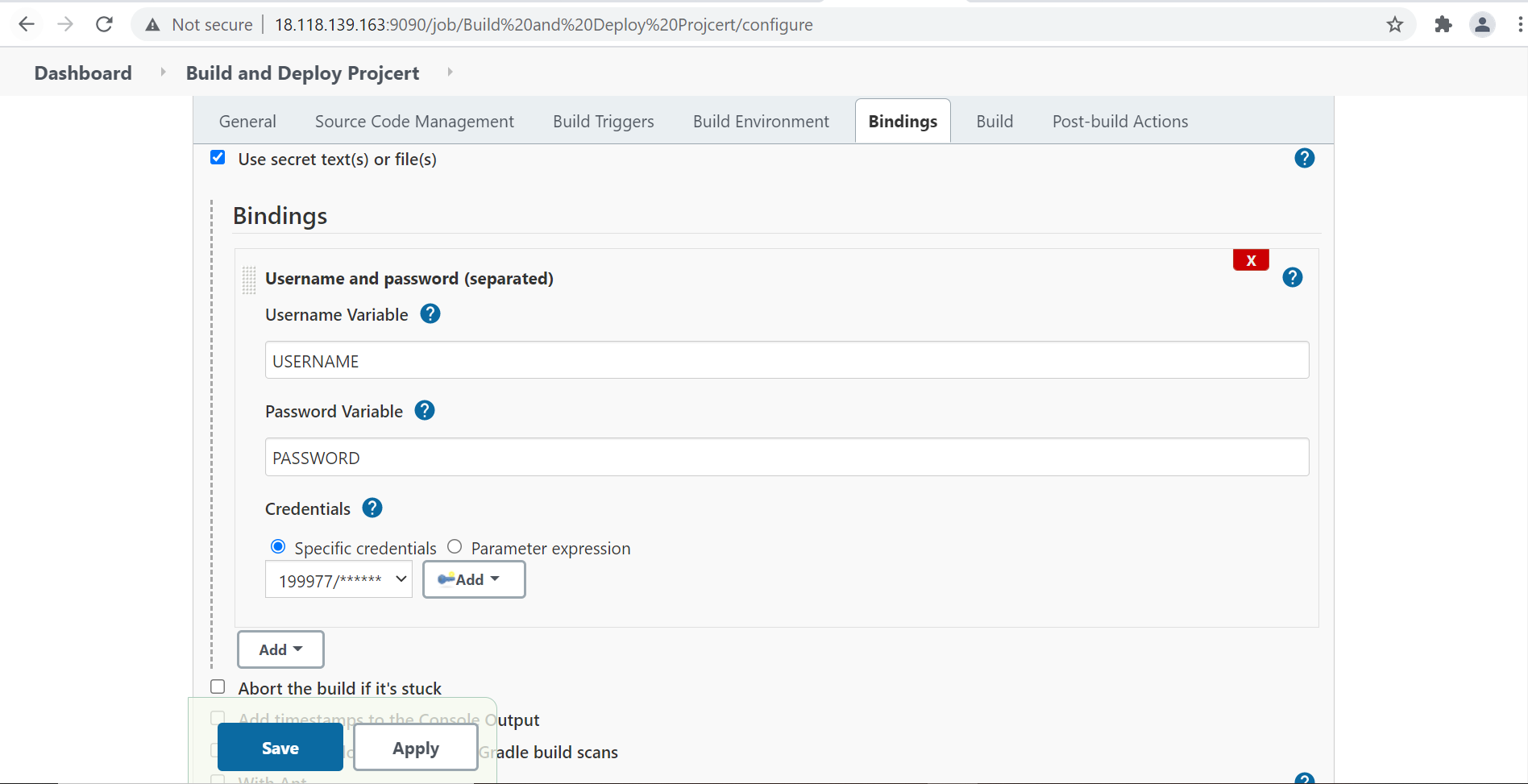


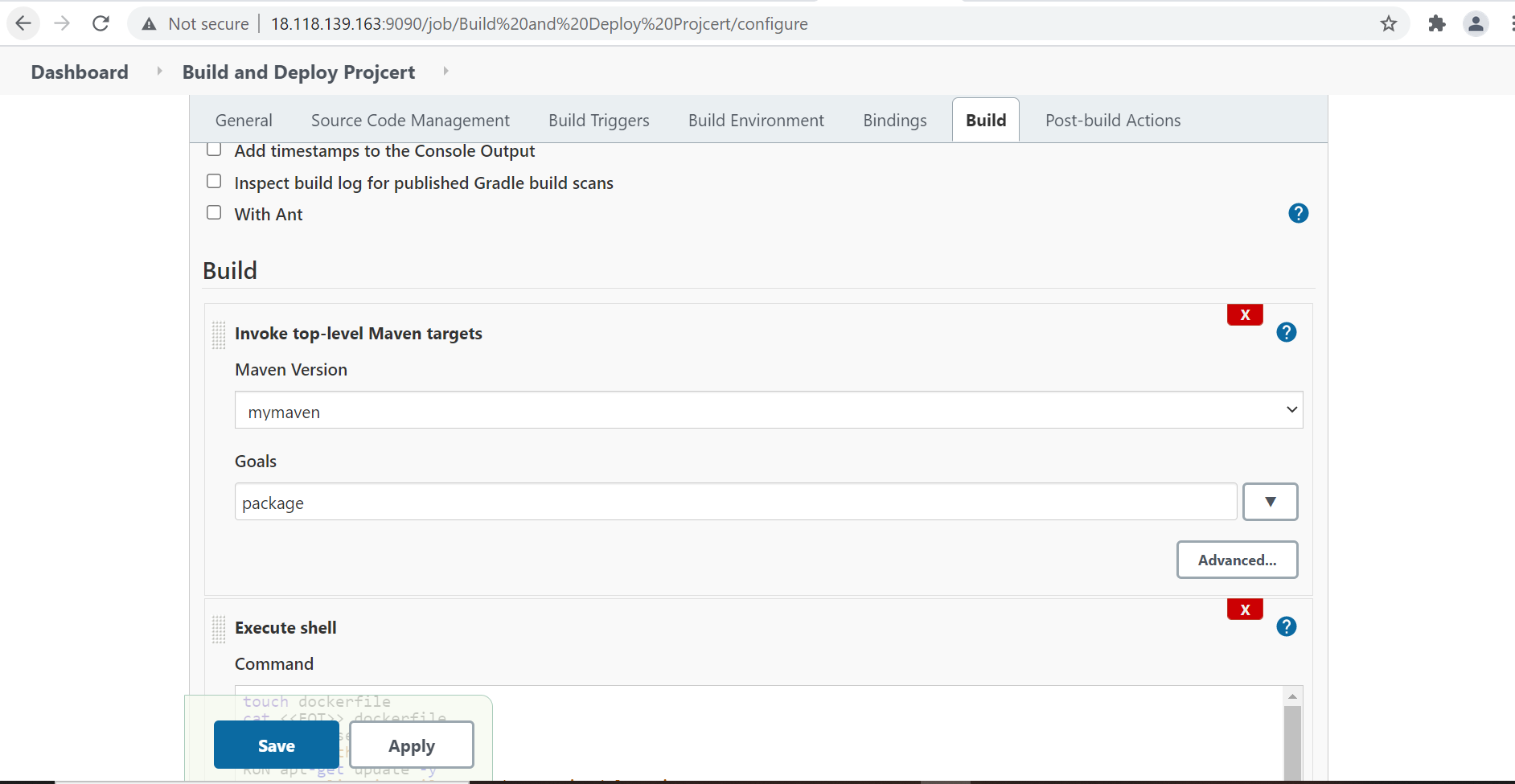
**Job 3:** Pull the PHP website and the Docker file from the git repo. Build and deploy this PHP docker container.

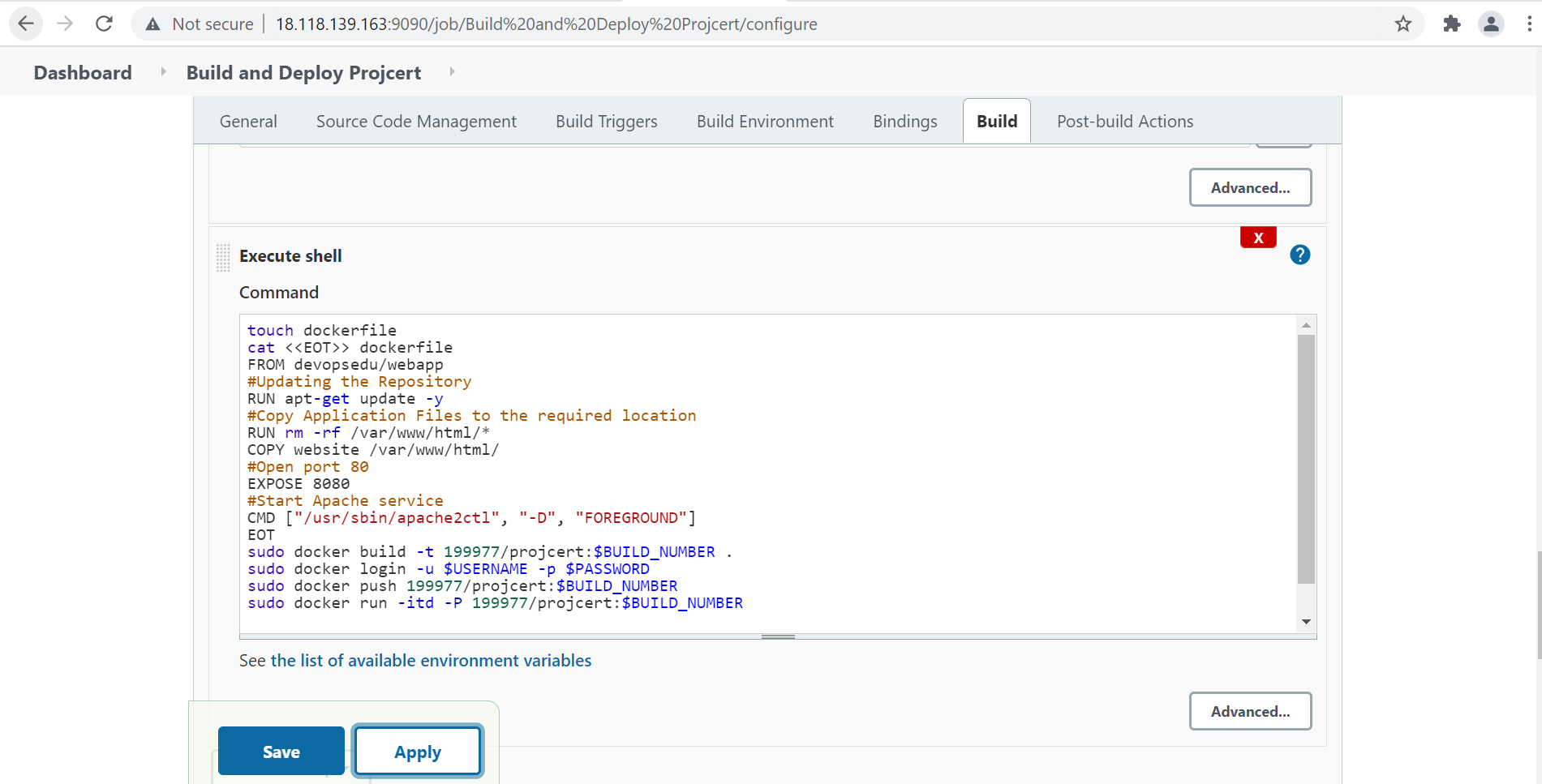


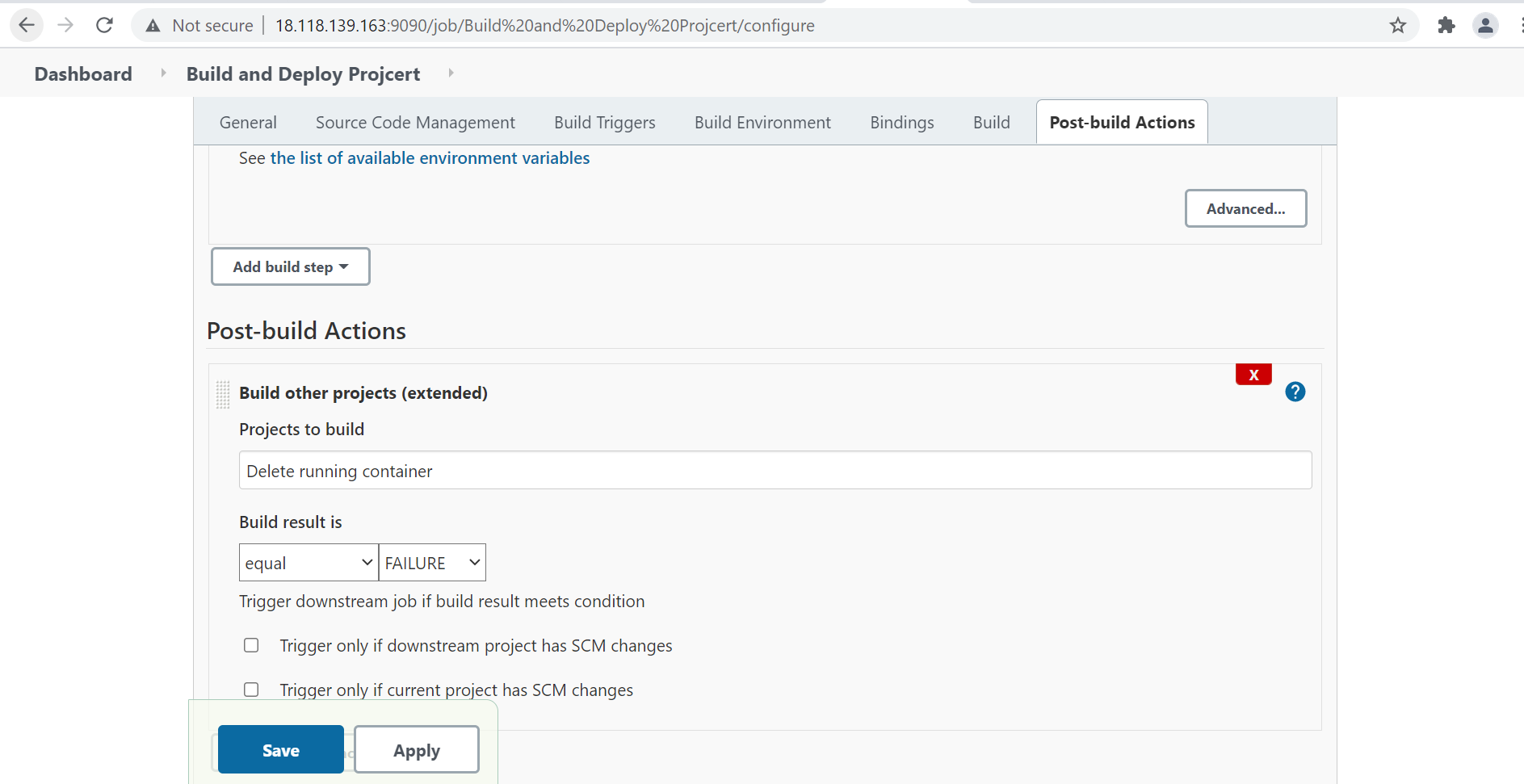




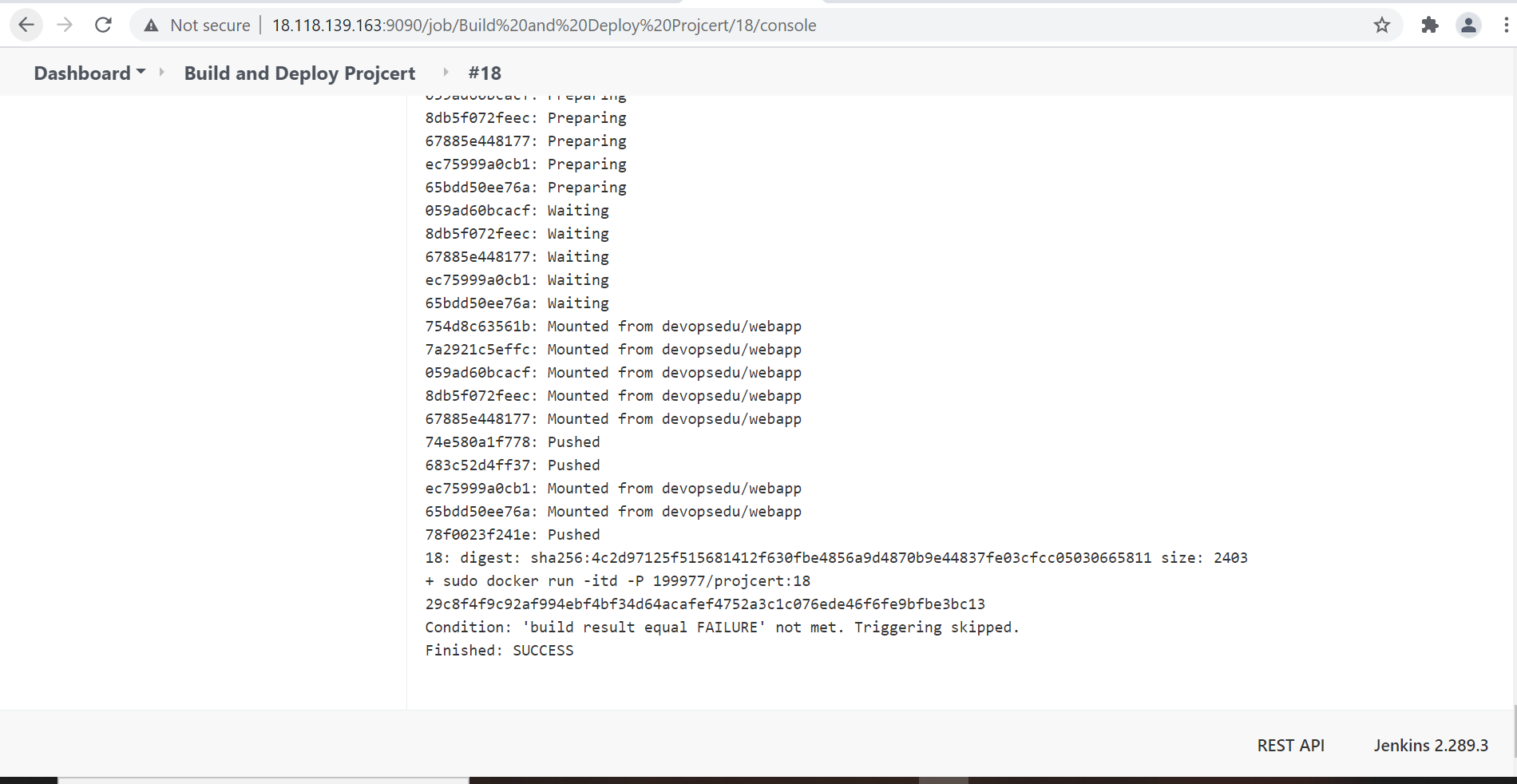


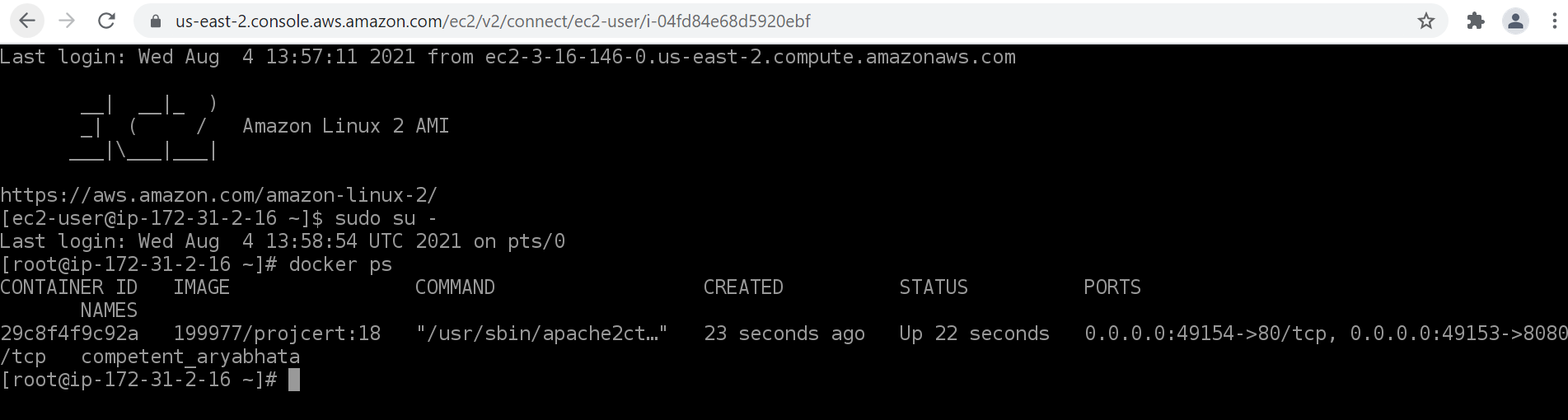




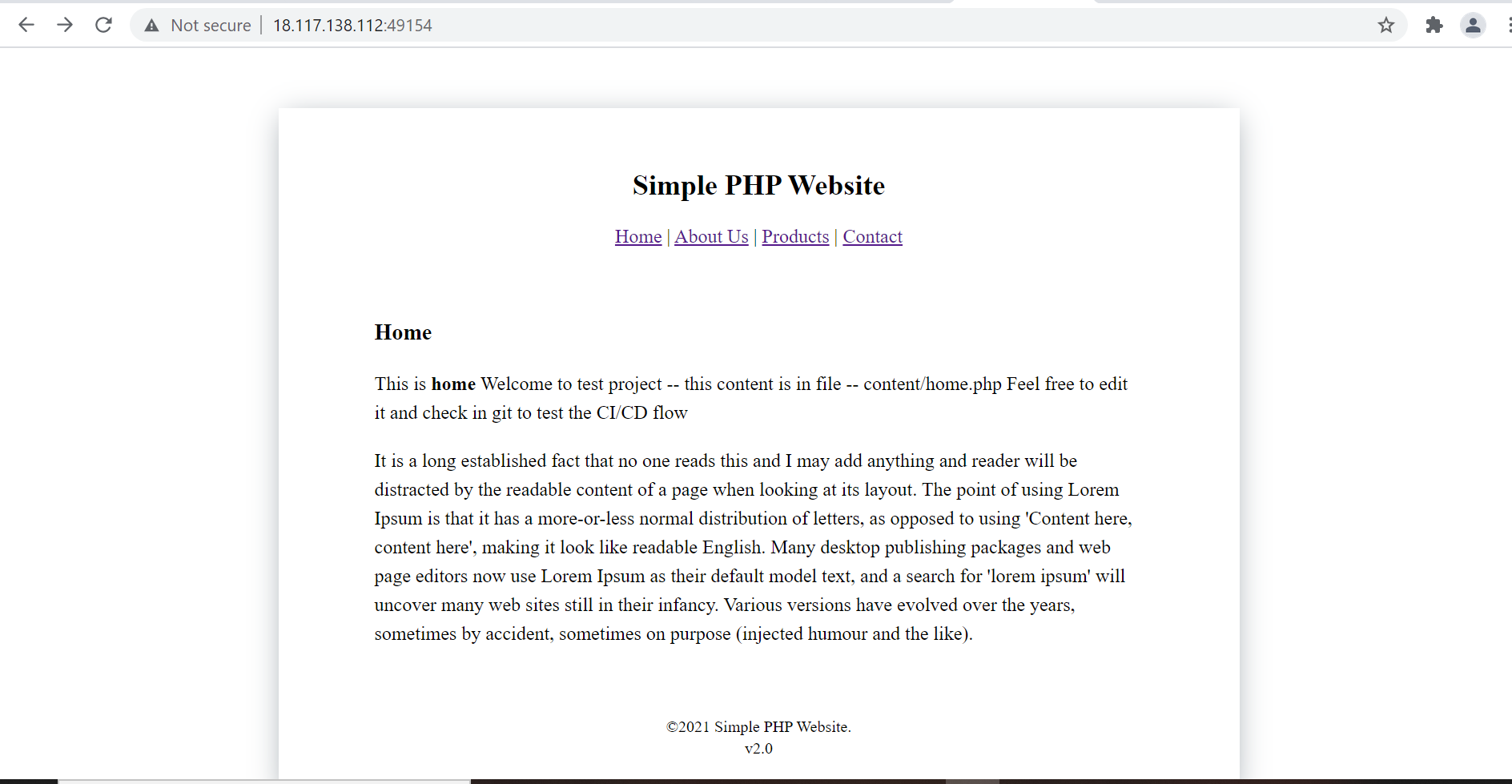


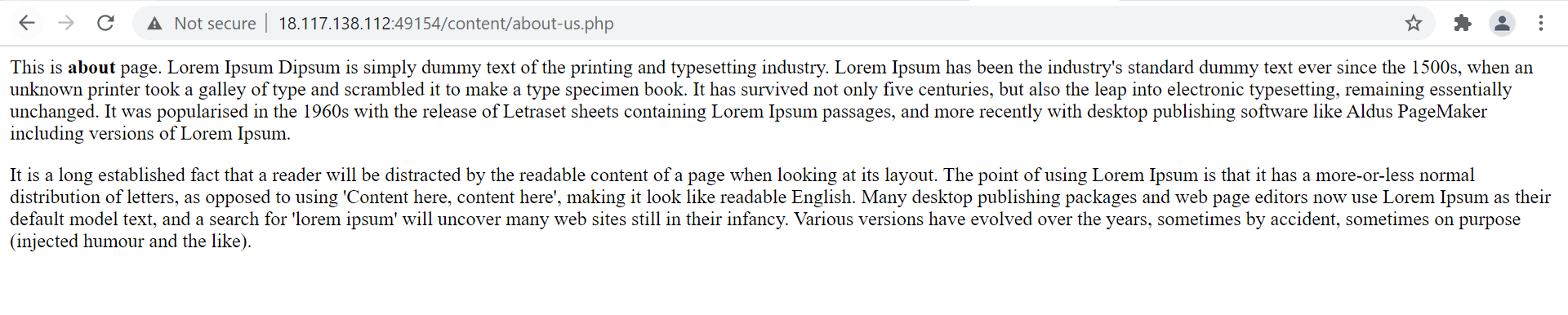
Output :-

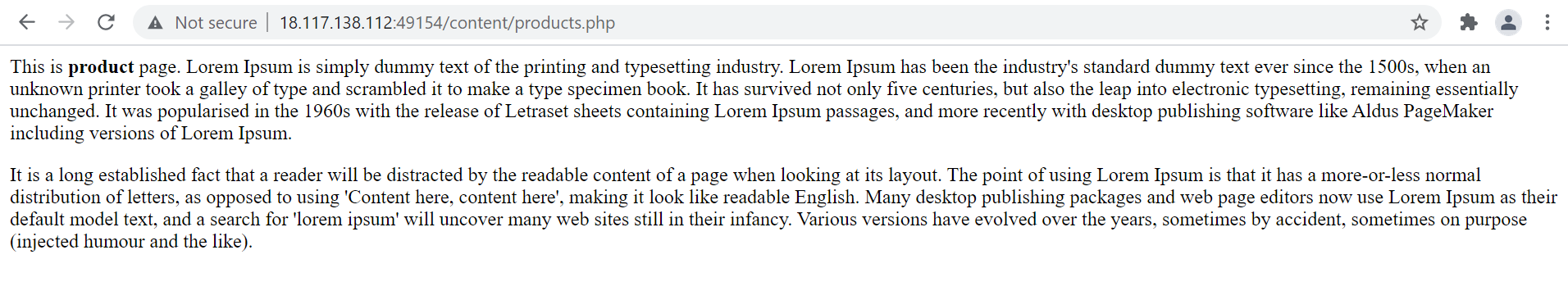


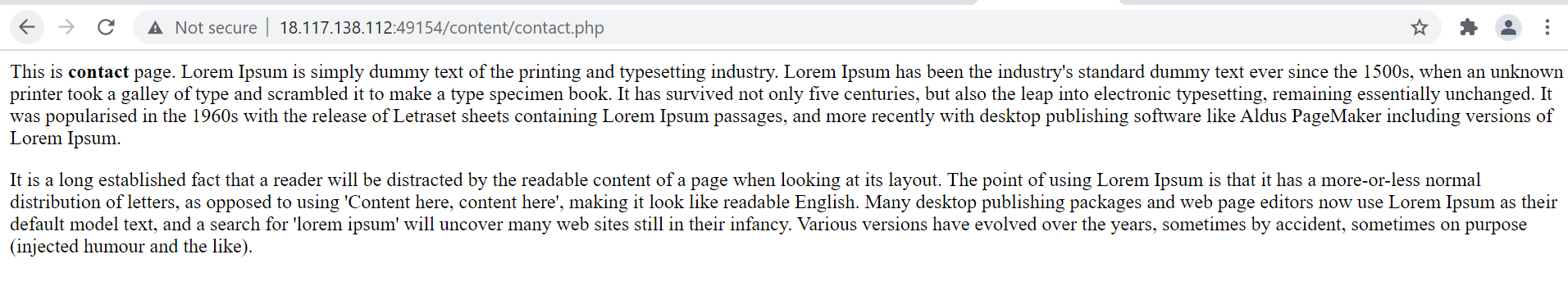




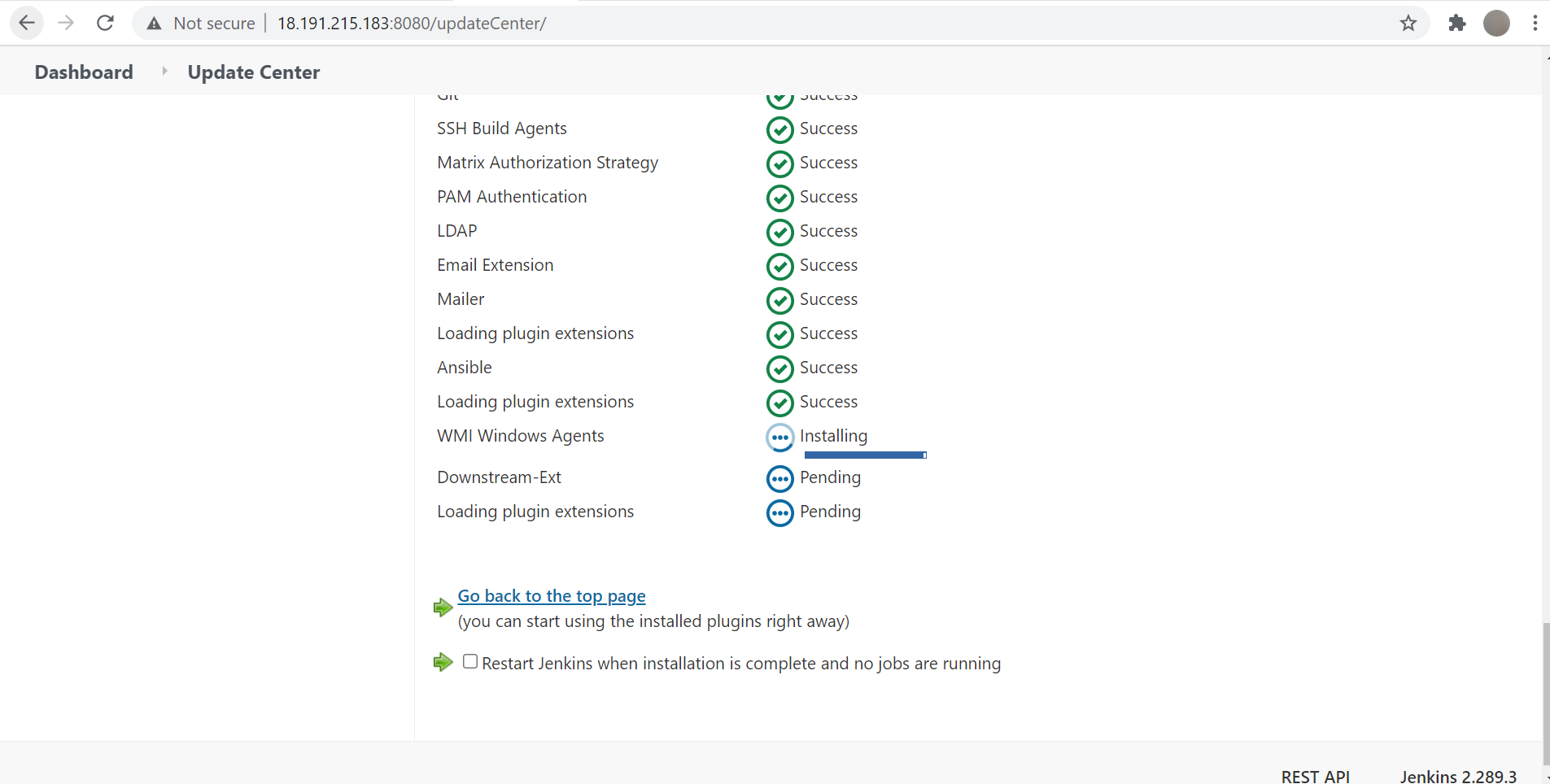


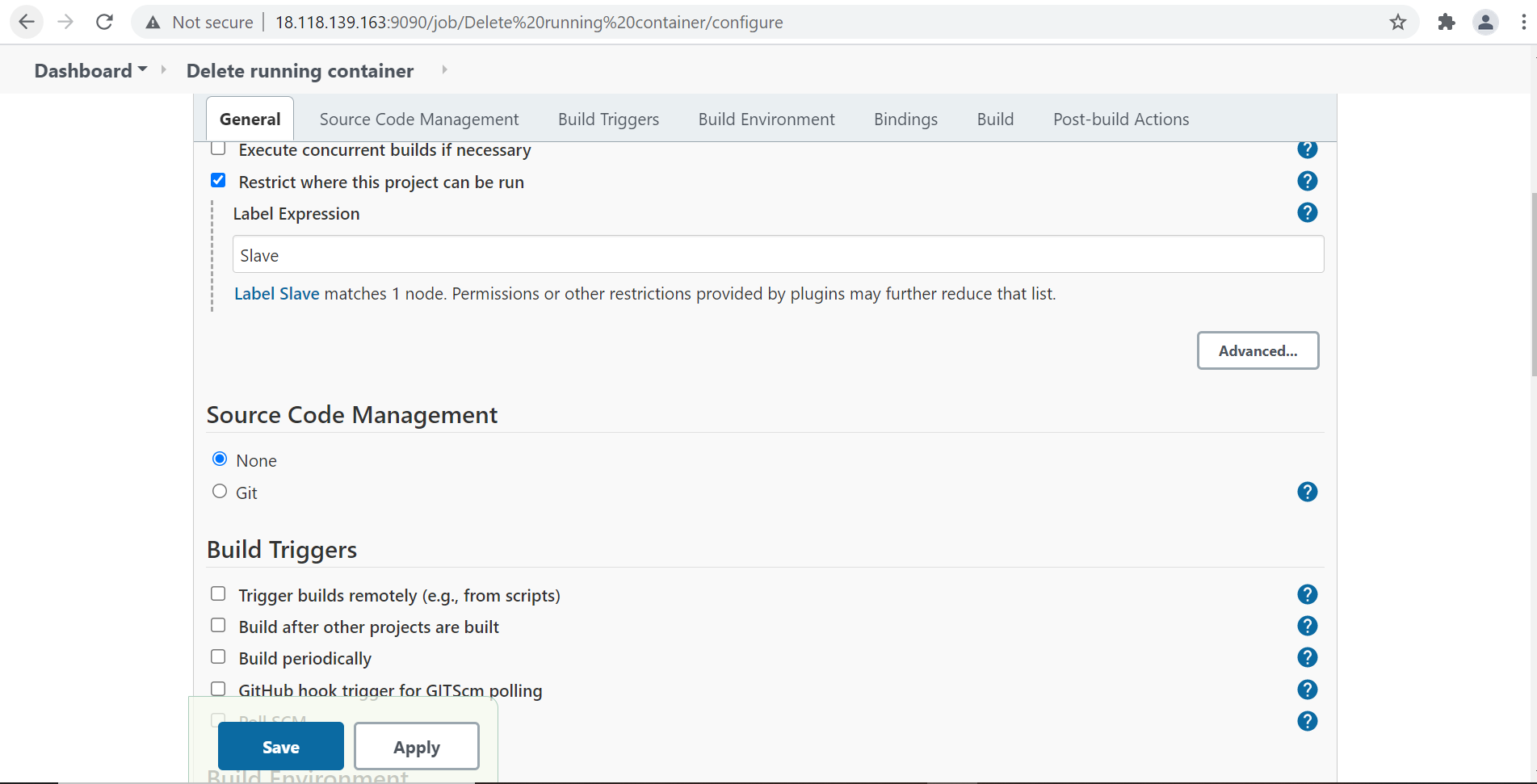




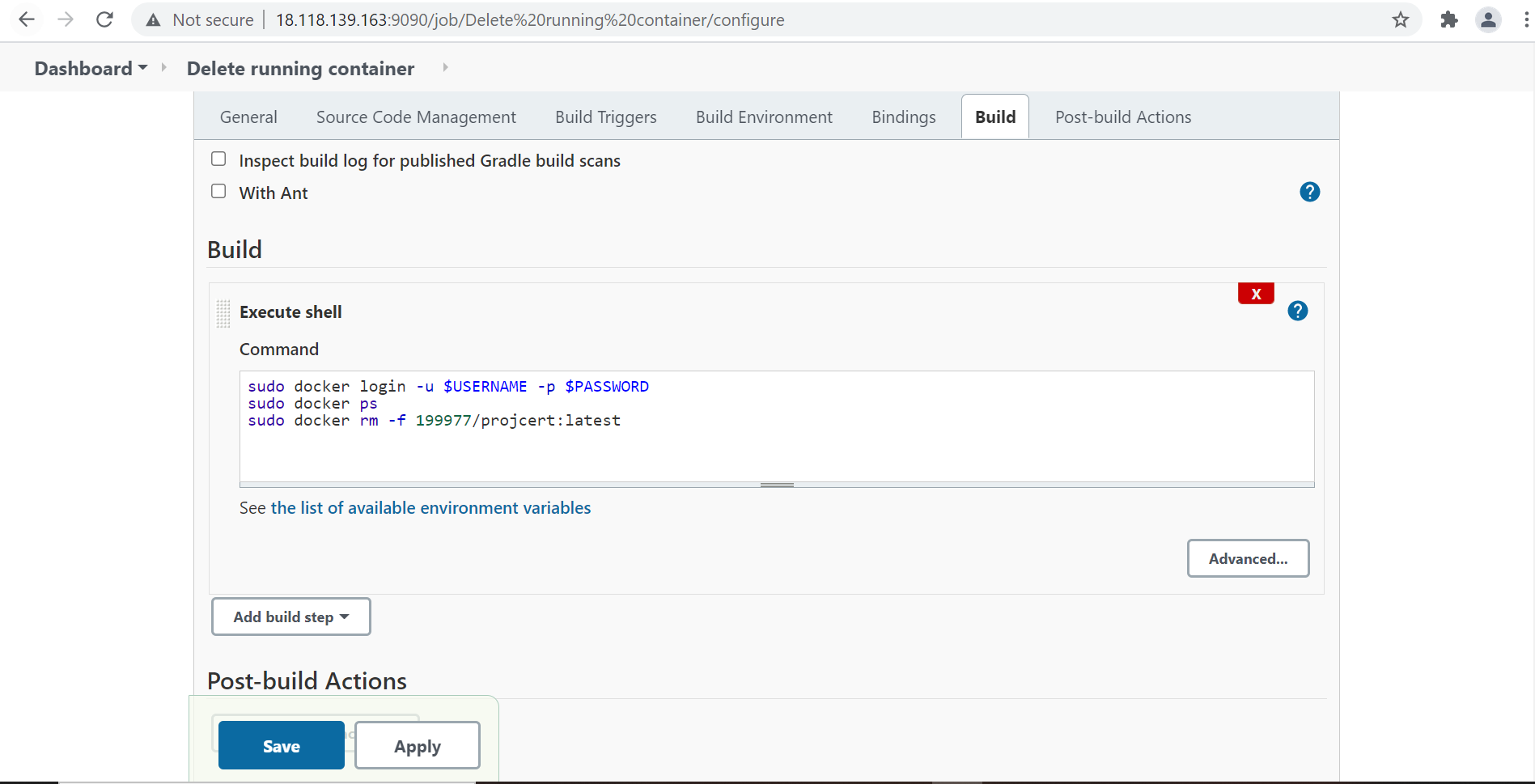


**Job 4**: If Job 3 fails, delete the running container on Test Server

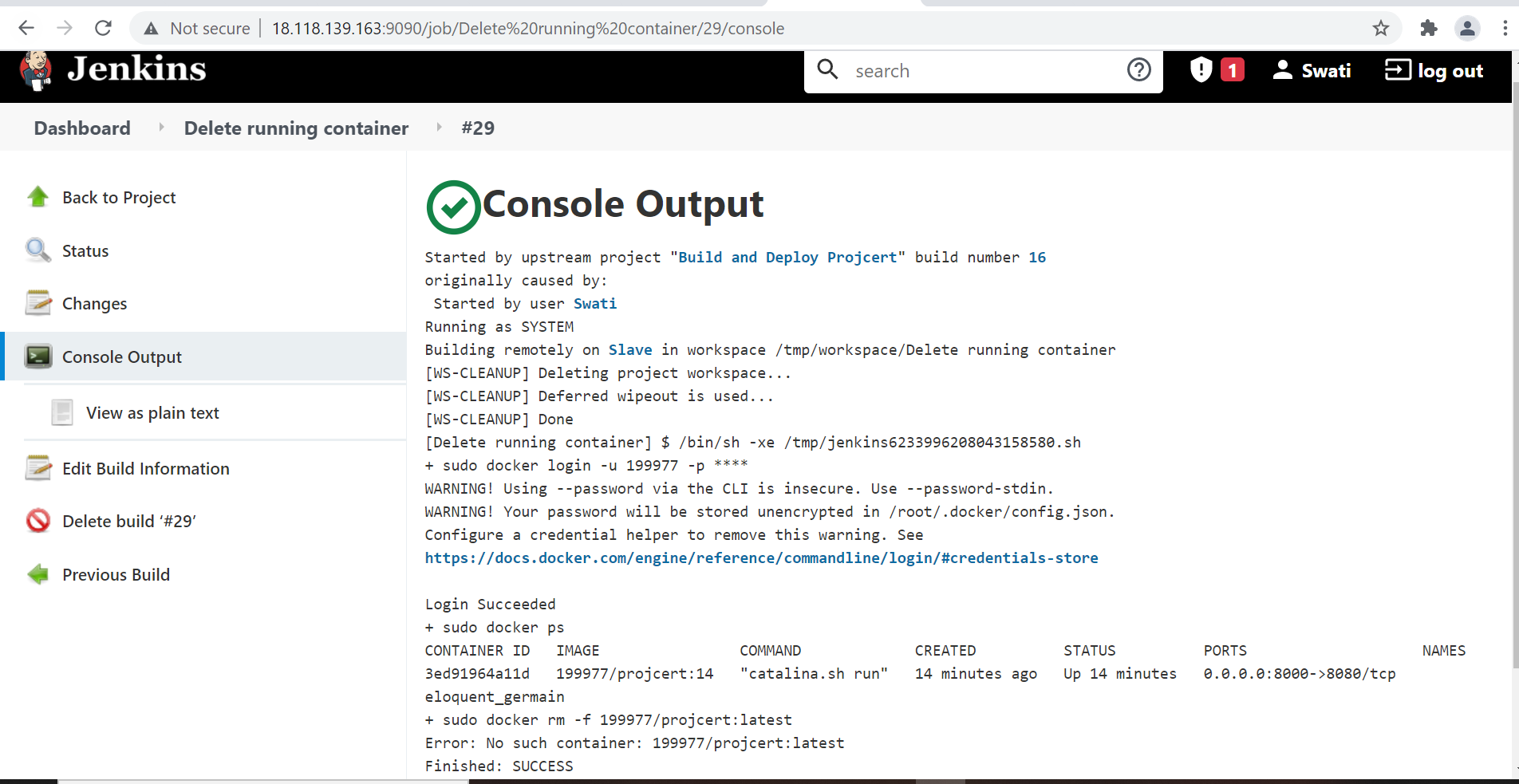




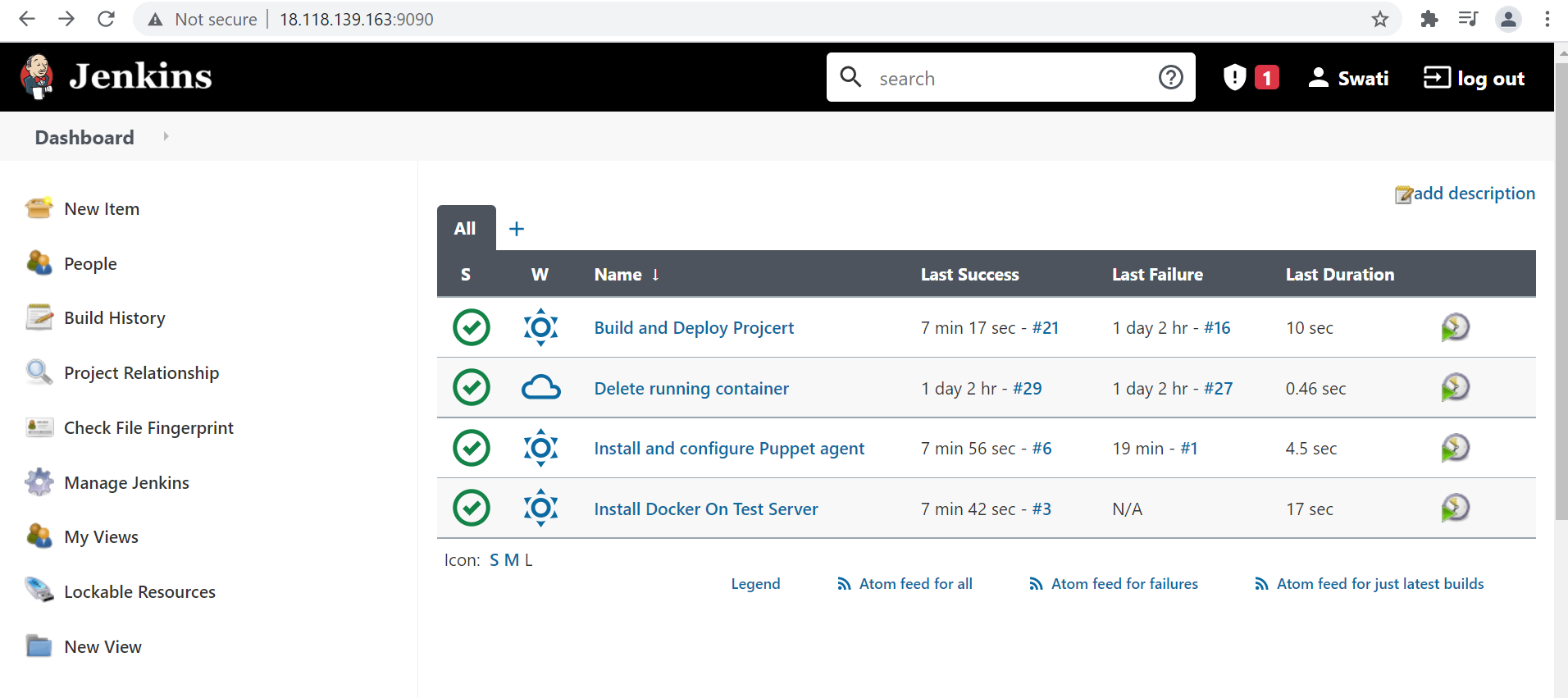




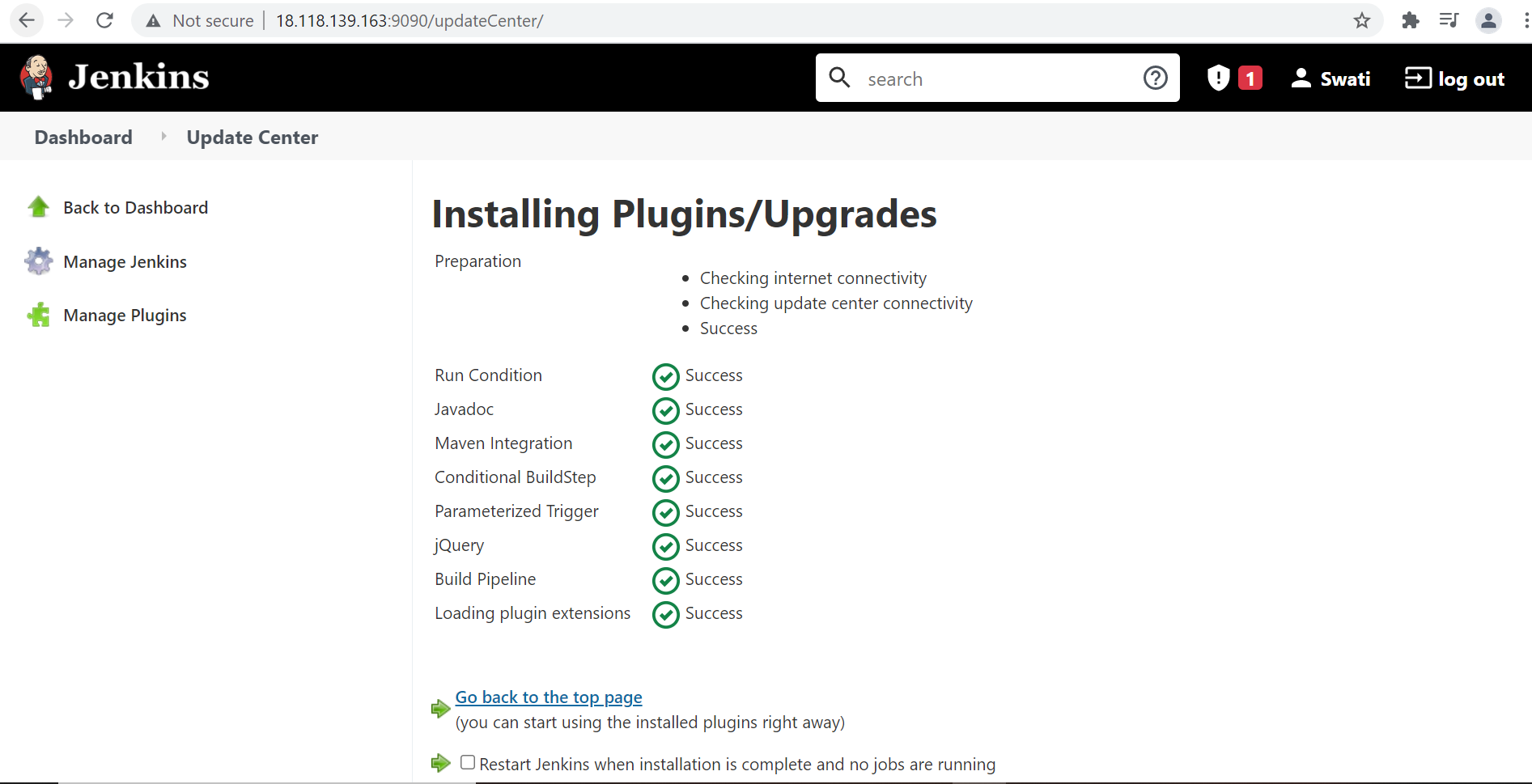
Output :-

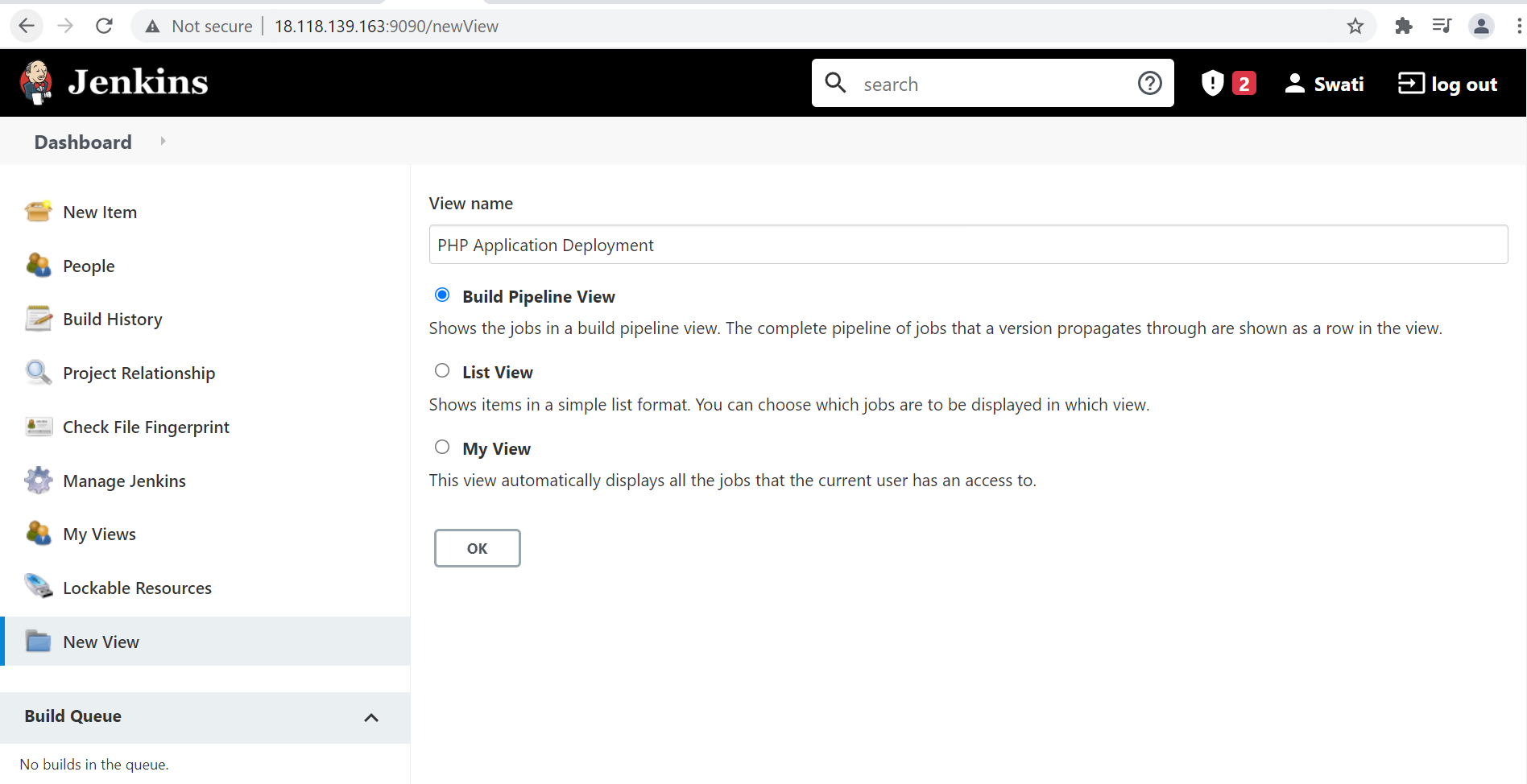


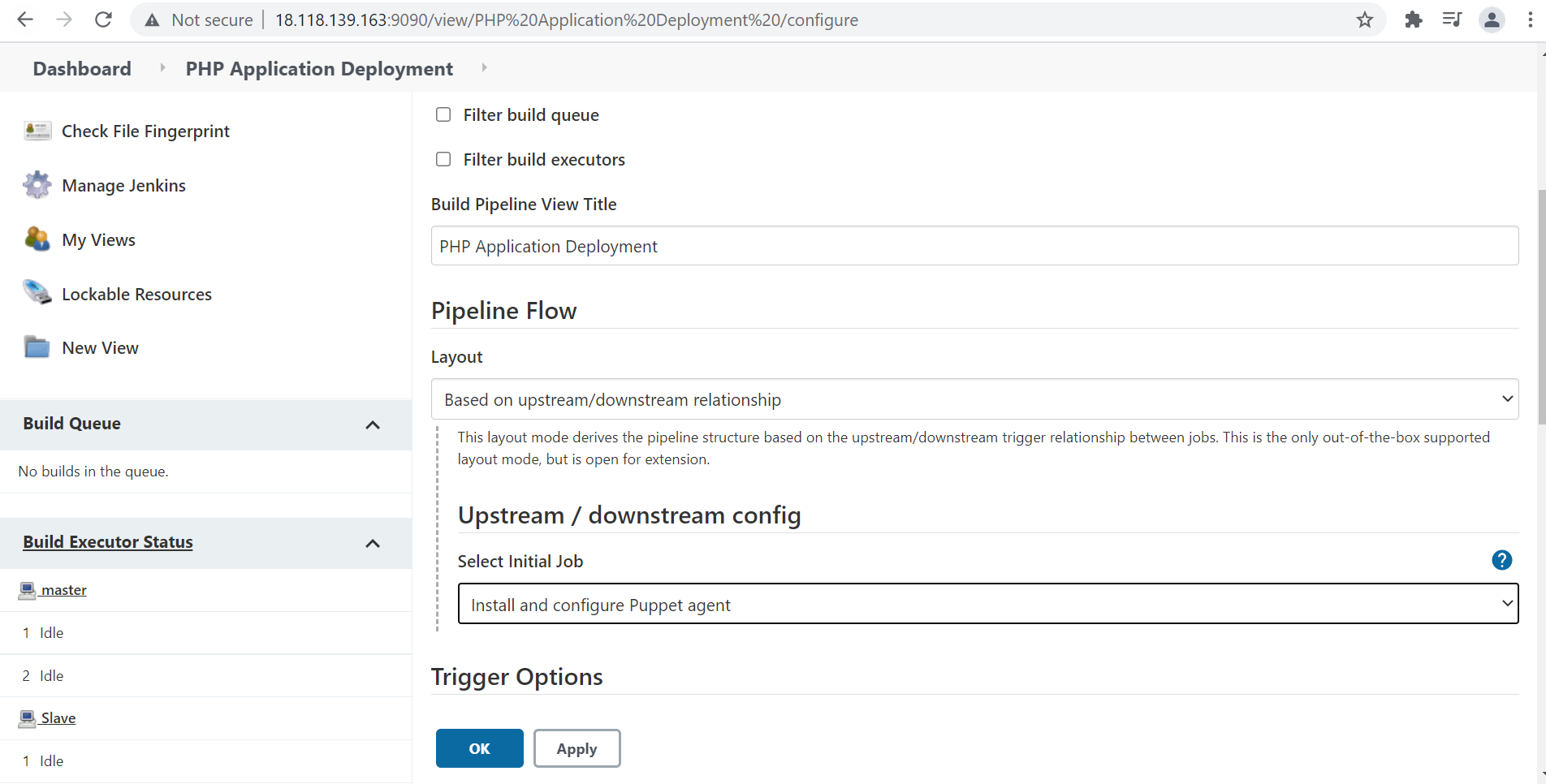
All jobs looks like as in given below screenshot



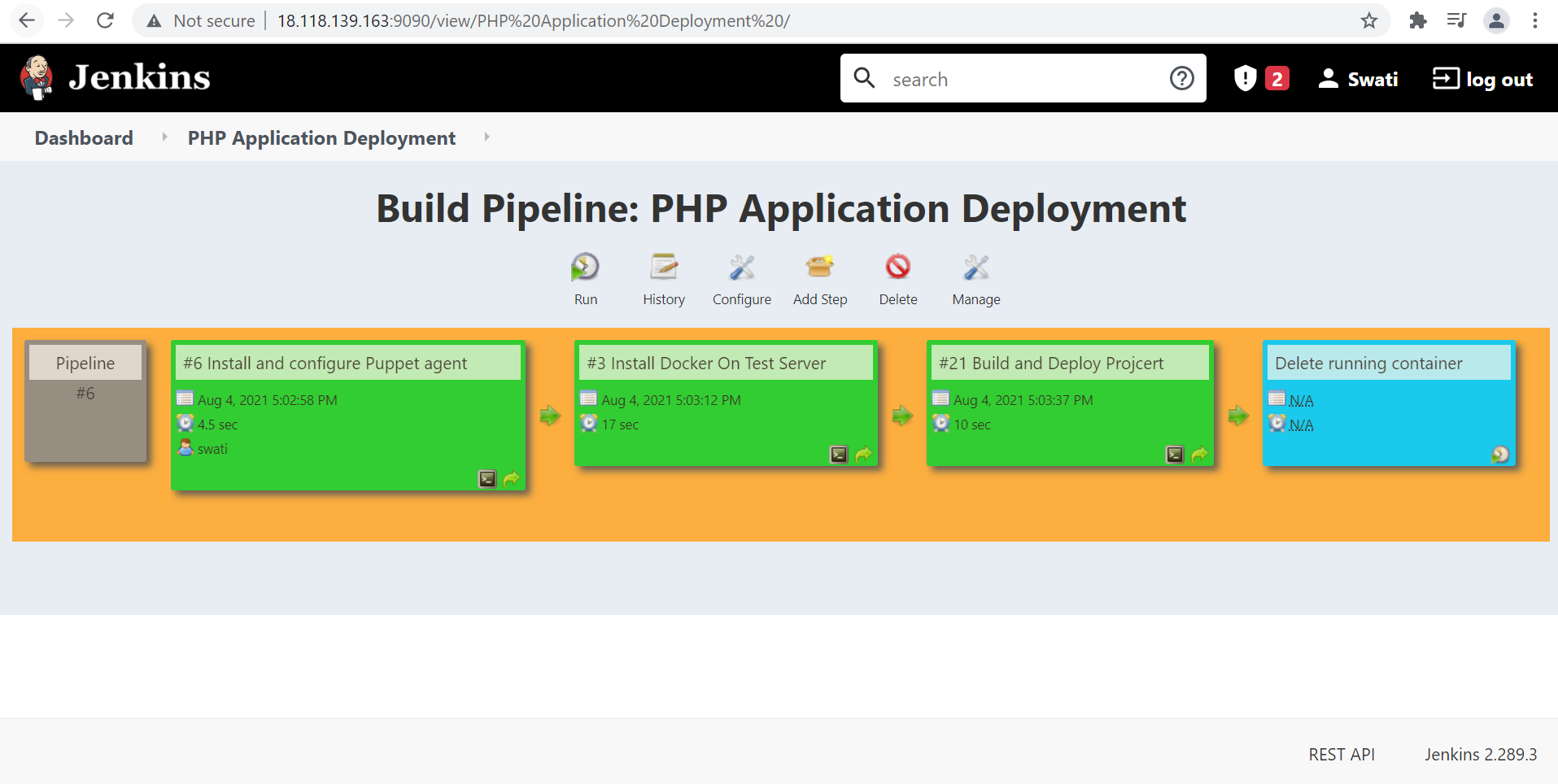
Let’s install build pipeline plugin arrange then in pipeline view :







Given below is the scenario when job3 - build and deploy projcert will be successful, in that case job4 – delete running container will not run.



And if job 3 – build and deploy project got failed then job 4 – delete the running container will run (screenshot is attached below

