PG DO - DevOps Capstone Project

**Infra Optimization**

Name: Sumbo Francisco

Email: sumbo.francisco@gmail.com

DESCRIPTION

Create a DevOps infrastructure for an e-commerce application to run on high-availability mode.

**Background of the problem statement:**  
A popular payment application, **EasyPay**where users add money to their wallet accounts, faces an issue in its payment success rate. The timeout that occurs with  
the connectivity of the database has been the reason for the issue.  
While troubleshooting, it is found that the database server has several downtime instances at irregular intervals. This situation compels the company to create their own infrastructure that runs in high-availability mode.  
Given that online shopping experiences continue to evolve as per customer expectations, the developers are driven to make their app more reliable, fast, and secure for improving the performance of the current system.

**Implementation requirements:**

1. Create the cluster (EC2 instances with load balancer and elastic IP in case of AWS)
2. Automate the provisioning of an EC2 instance using Ansible or Chef Puppet
3. Install Docker and Kubernetes on the cluster
4. Implement the network policies at the database pod to allow ingress traffic from the front-end application pod
5. Create a new user with permissions to create, list, get, update, and delete pods
6. Configure application on the pod
7. Take snapshot of ETCD database
8. Set criteria such that if the memory of CPU goes beyond 50%, environments automatically get scaled up and configured

**The following tools must be used:**

1. EC2
2. Kubernetes
3. Docker
4. Ansible or Chef or Puppet

Índice

[PG DO - DevOps Capstone Project 1](#_Toc71826098)

[**Infra Optimization** 1](#_Toc71826099)

[1 Introduction 4](#_Toc71826100)

[2 Project Repository 4](#_Toc71826101)

[3 Docker Server 7](#_Toc71826102)

[4 Build WebAPP (Spring boot Maven project) 9](#_Toc71826103)

[4.1 Build and Deploy Web application in a Tomcat Docker container with Jenkins 10](#_Toc71826104)

[4.2 Create the Docker Image (Dockerfile) and Publish on Repository 15](#_Toc71826105)

[Publish and make available image in the Docker Registry 16](#_Toc71826106)

[5 Automation of AWS EC2 instance provisioning in ANSIBLE 18](#_Toc71826107)

[5.1 Ansible Installation: 18](#_Toc71826108)

[5.2 Create the playbook to automate the provision of 3 AWS ec2 instances 20](#_Toc71826109)

[6 Kubernetes Cluster for application Deployment and HA 23](#_Toc71826110)

[6.2 Create the Deployment YAML file to pull the Application image from Repository and configure the application in Pod 26](#_Toc71826111)

[6.3 Configure the Web APP to use 3 replicas 28](#_Toc71826112)

[6.4 Configure LoadBalancer and High Availbility 29](#_Toc71826113)

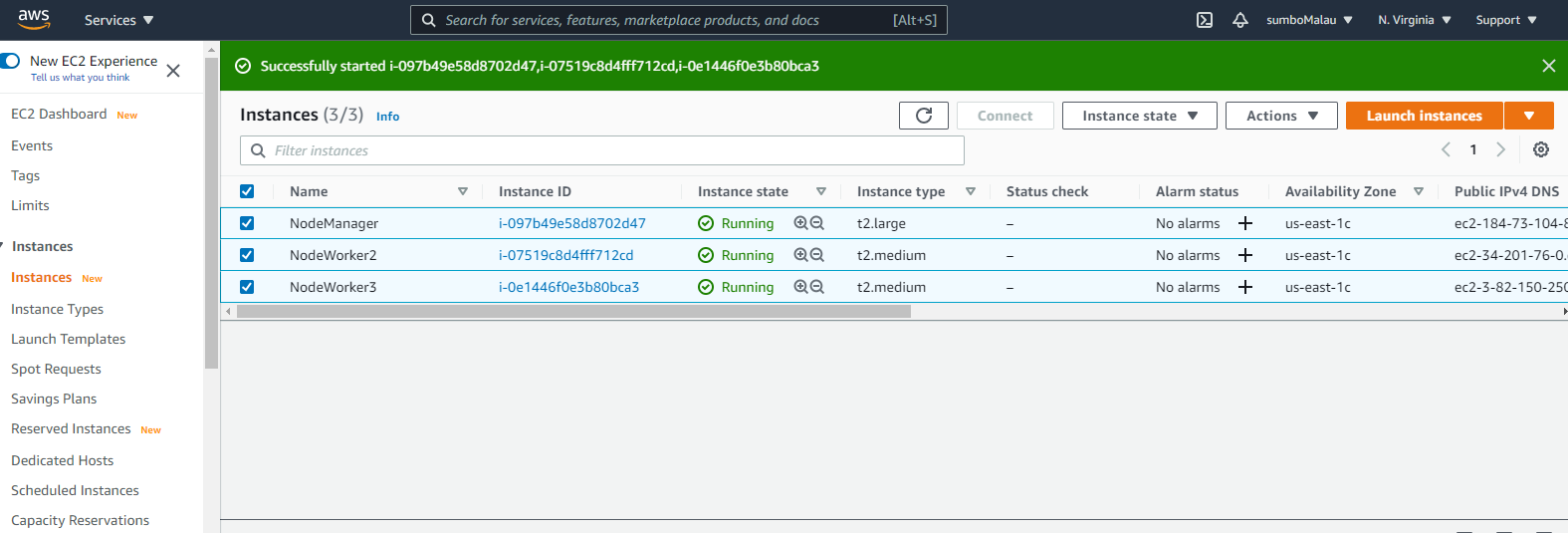
[6.4.1 Testing the loadBalancer 30](#_Toc71826114)

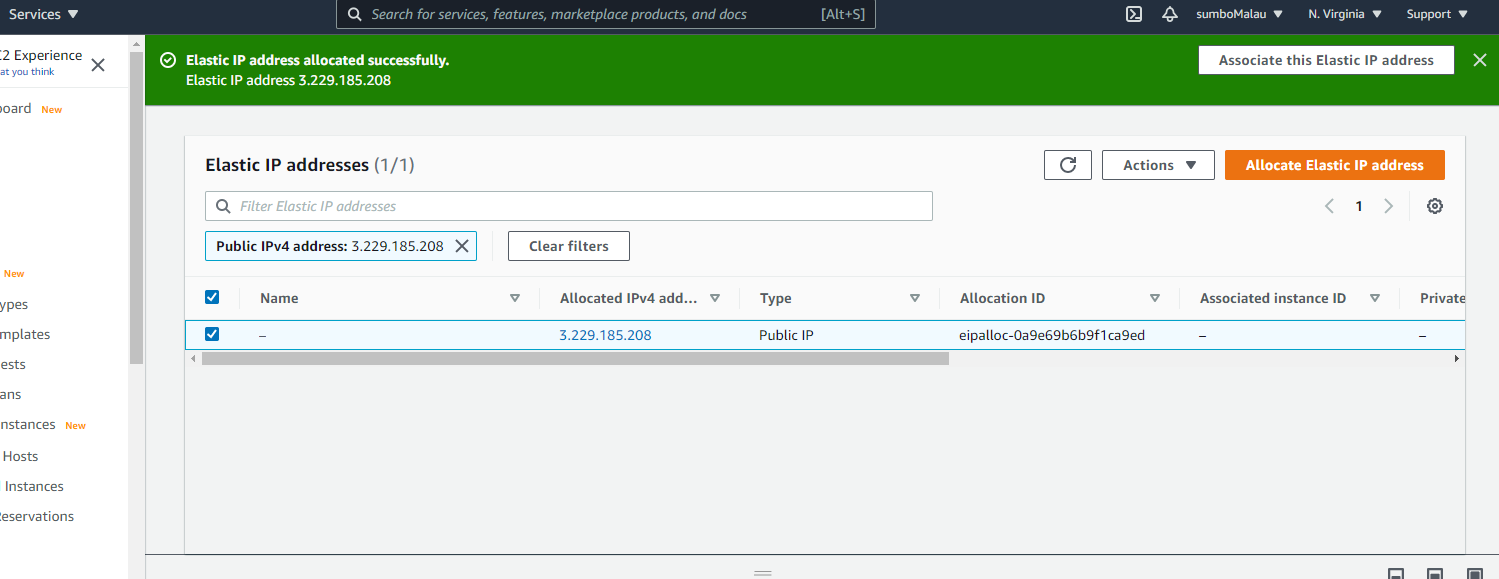
[6.5 Create a new user with permissions to manage pods 31](#_Toc71826115)

[6.6 Take snapshot of ETCD database 34](#_Toc71826116)

# 1 Introduction

For this project, we will implement a cluster in AWS (EC2 instances) with 3 nodes, using Kubernets, to manage the Loadbalance for our application.



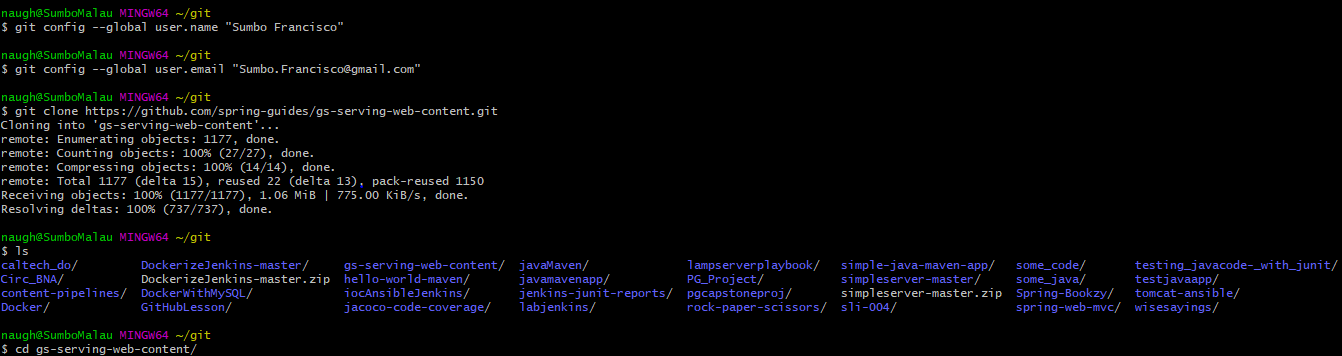


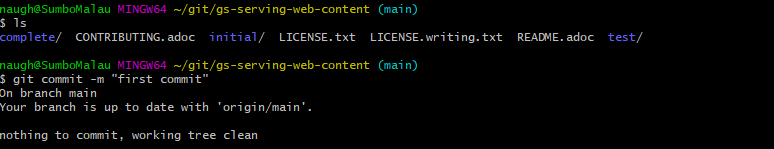
We will create one SpringBoot Web APP, in a dockerImage, and deploy it to our Kubernetes cluster, created in AWS EC2 instances.

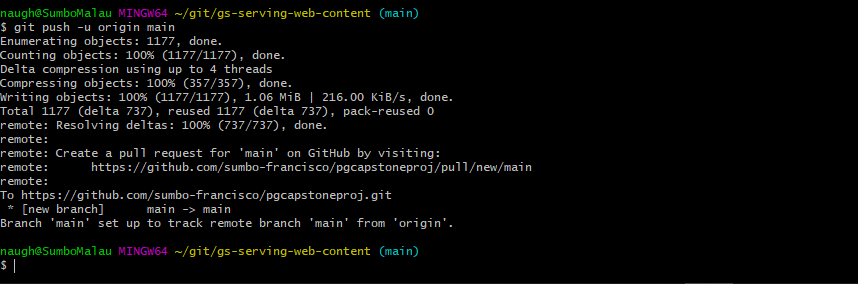
# 2 Project Repository

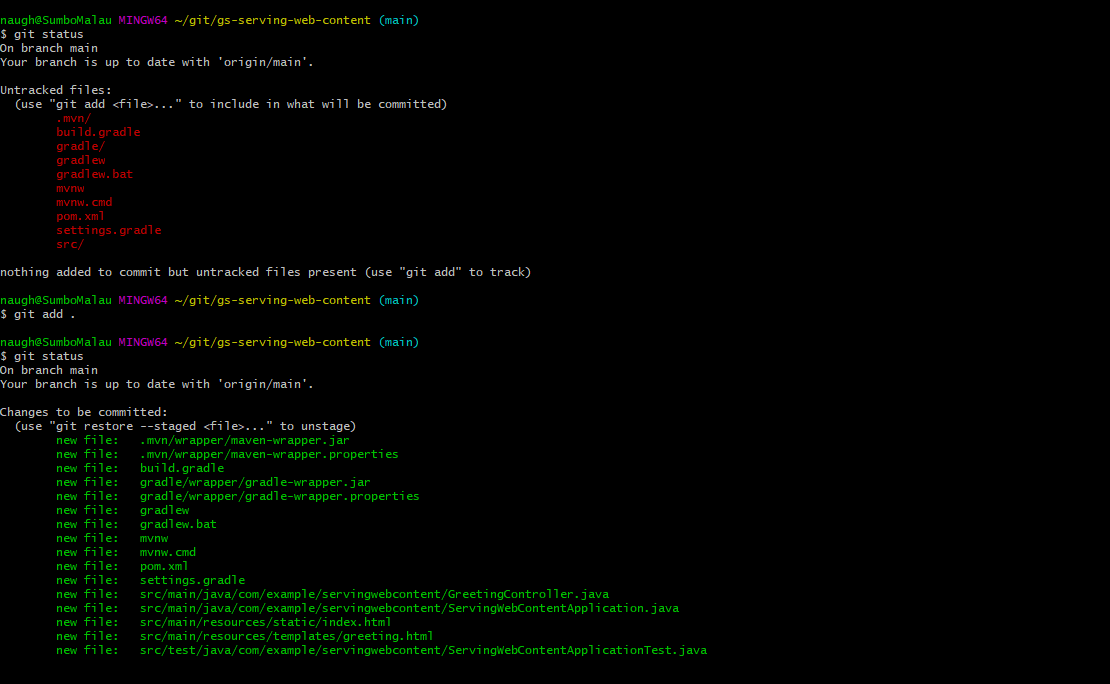
Create a new repo in GitHub (**please select main branch)**:

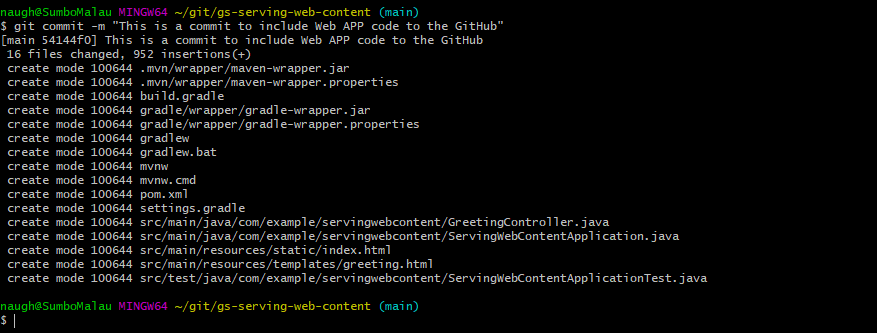
<https://github.com/sumbo-francisco/pgcapstoneproj.git>

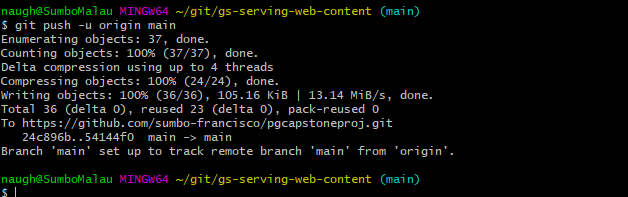


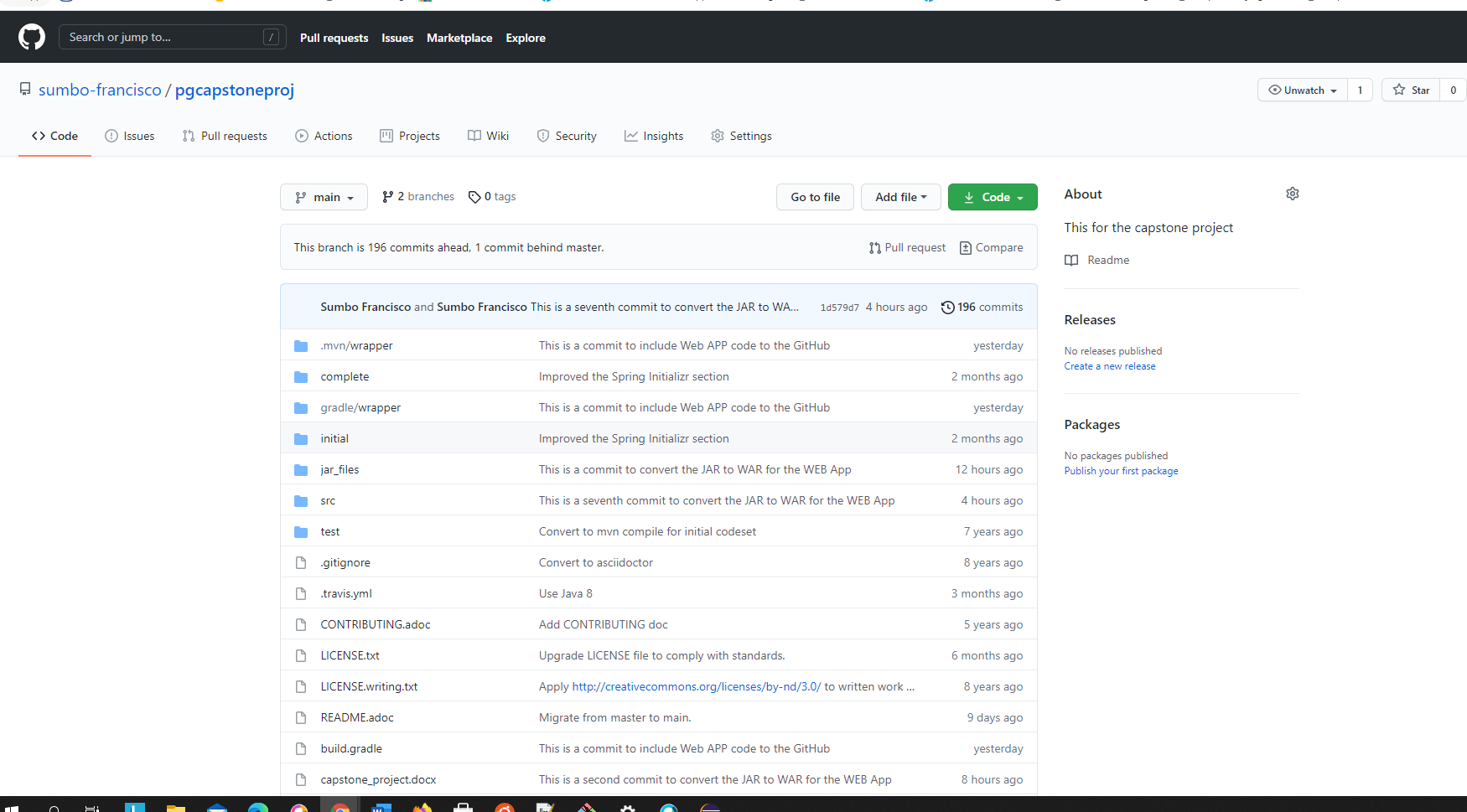




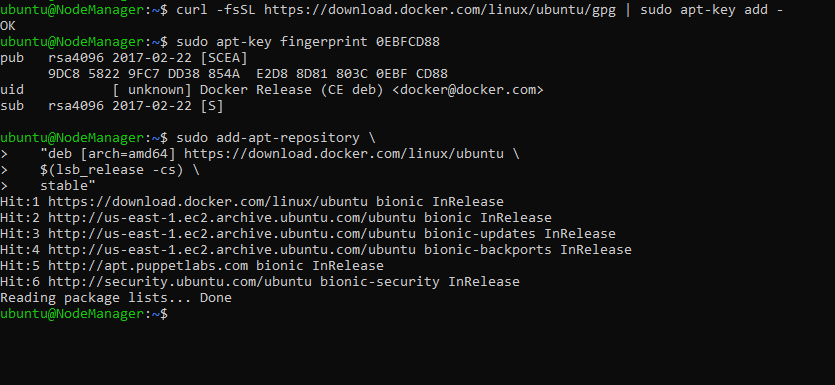


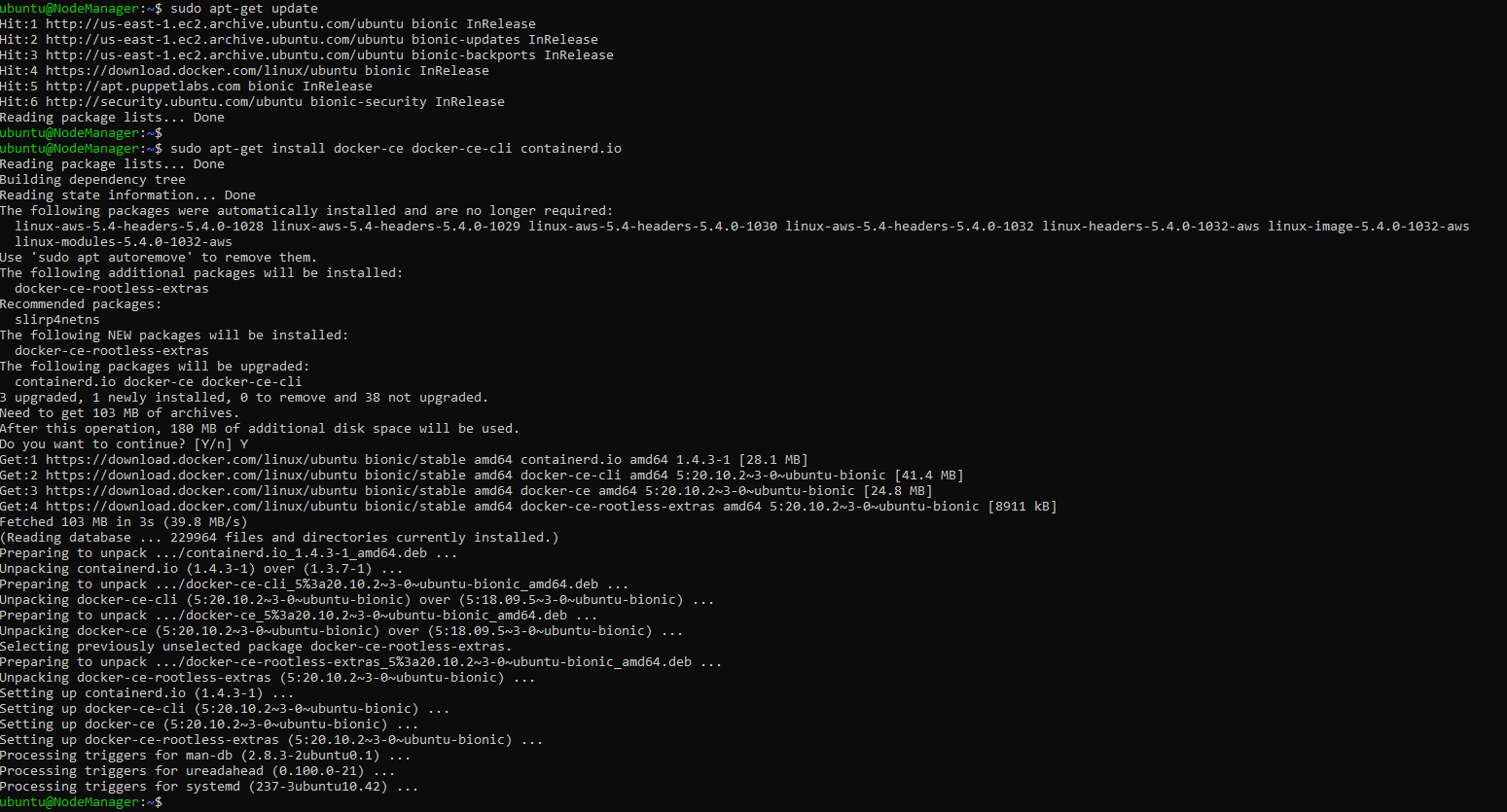




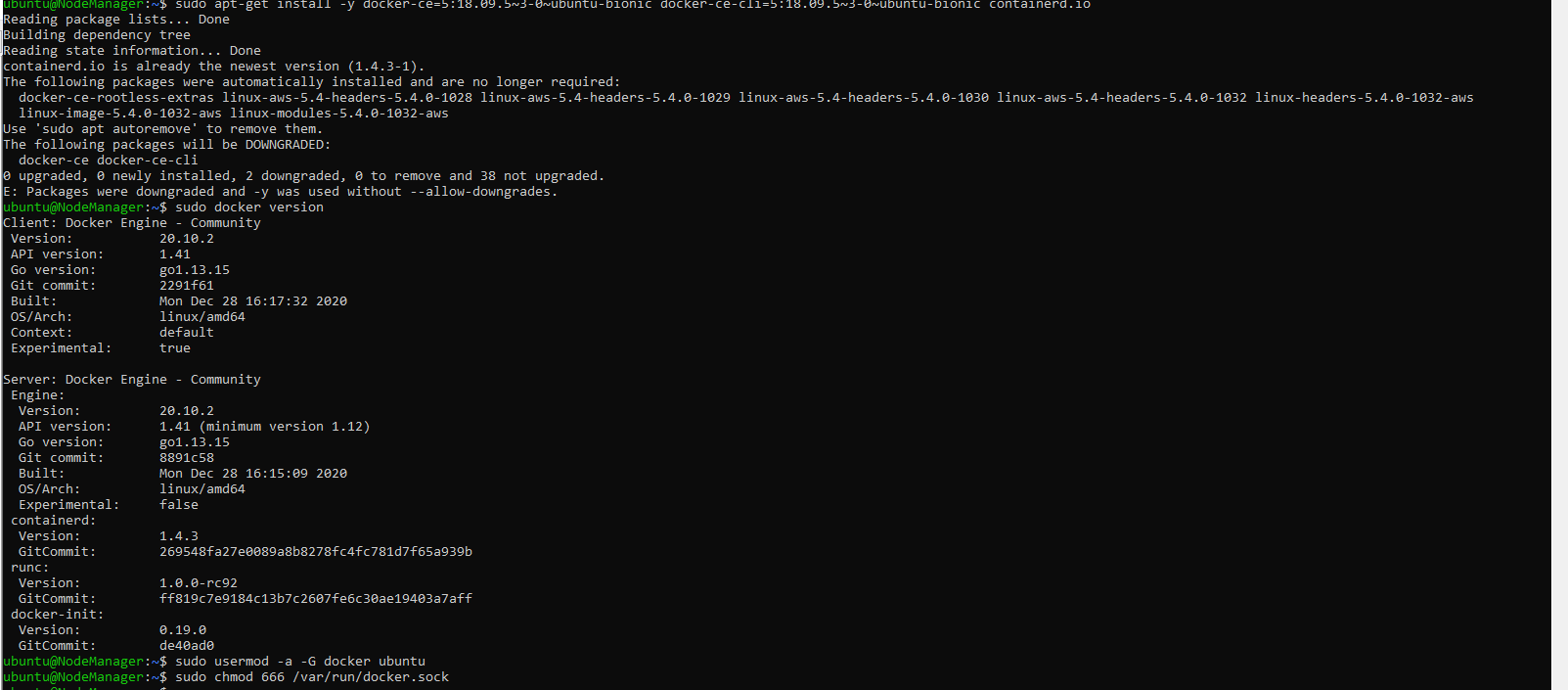


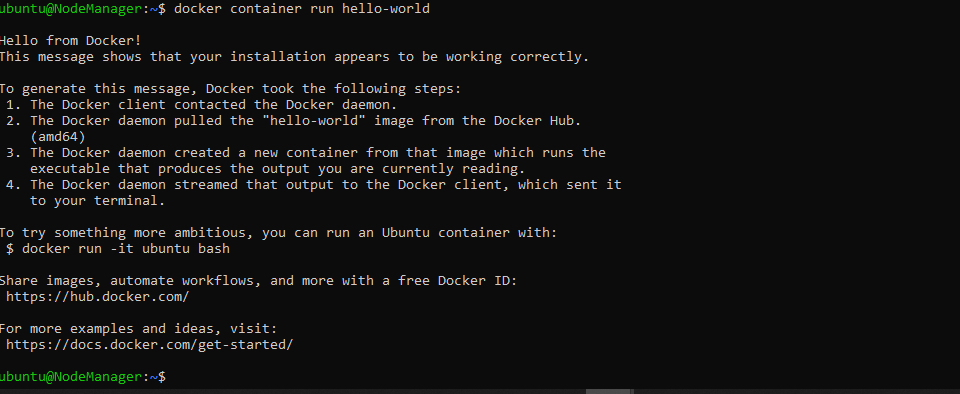
# 3 Docker Server

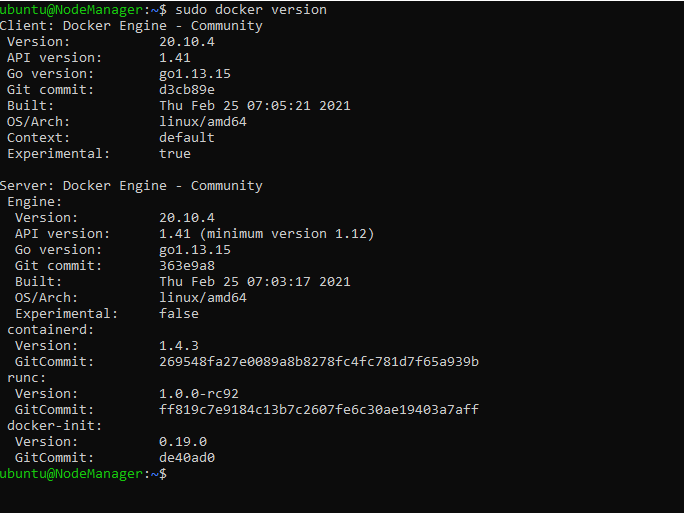




$ sudo apt-get install docker-ce=<VERSION\_STRING> docker-ce-cli=<VERSION\_STRING> containerd.io

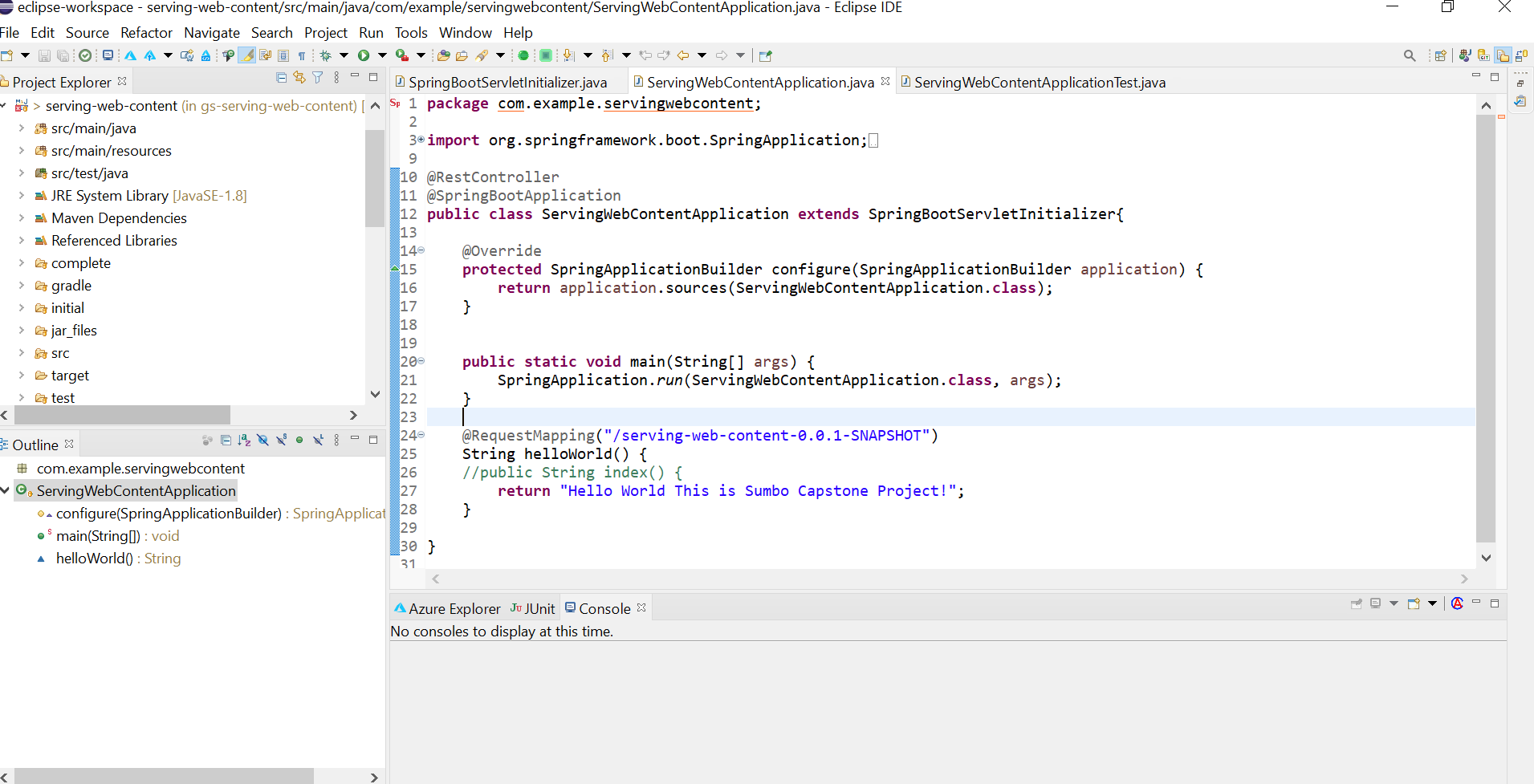


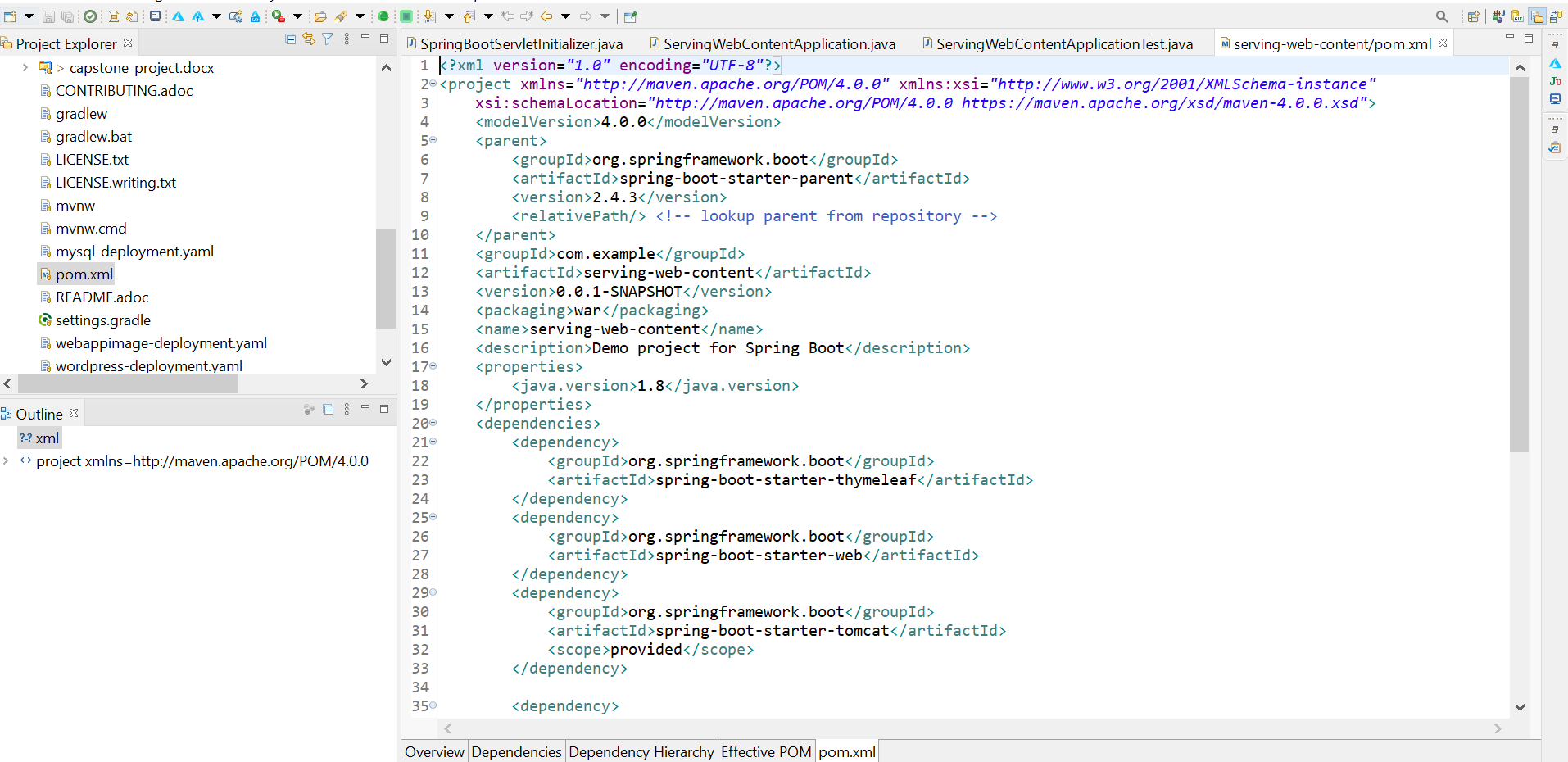




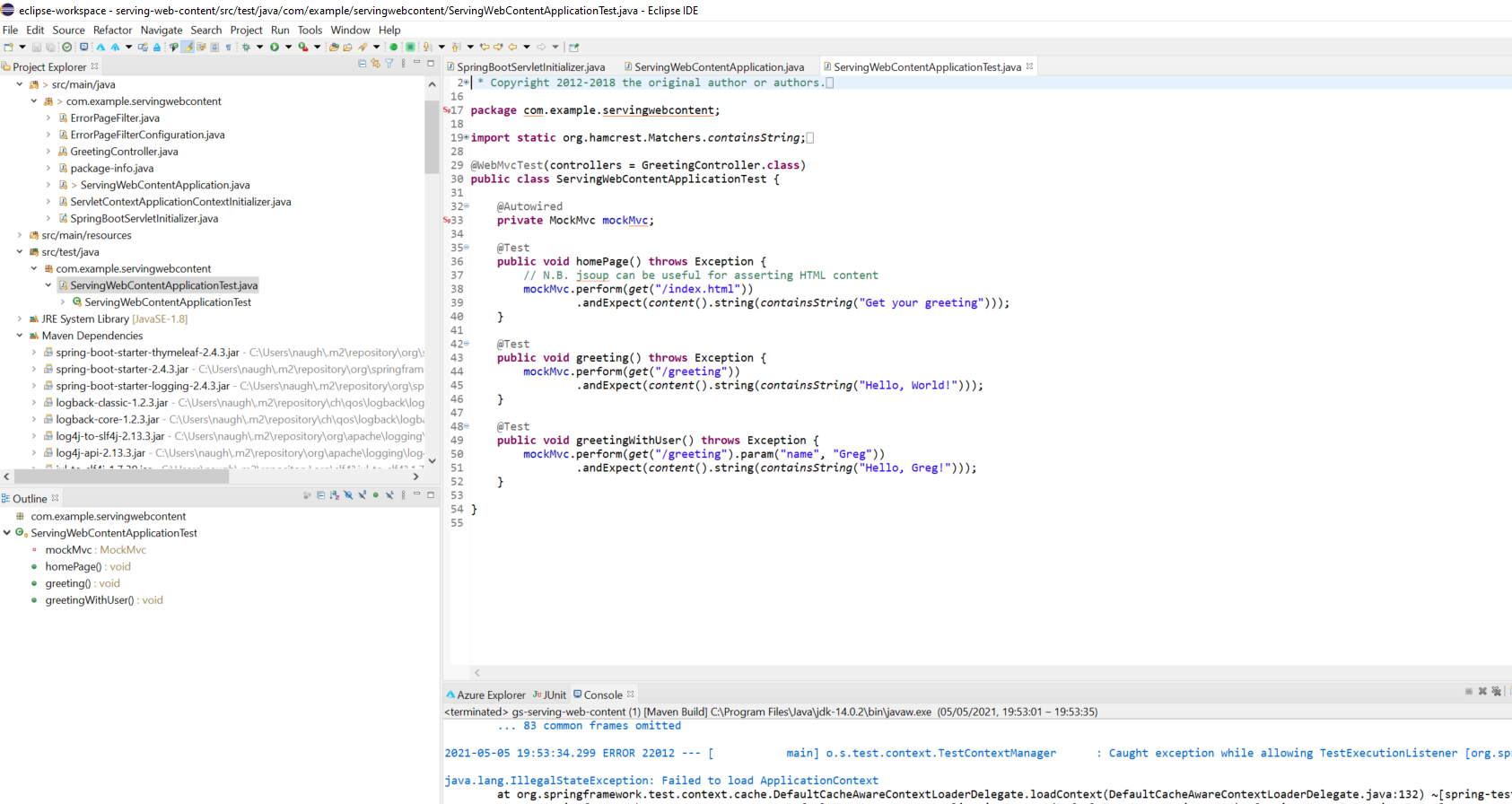
# 4 Build WebAPP (Spring boot Maven project)

I am using one existing Springbook java web application, build in as Maven project. All the project code, are included in my github Repository.



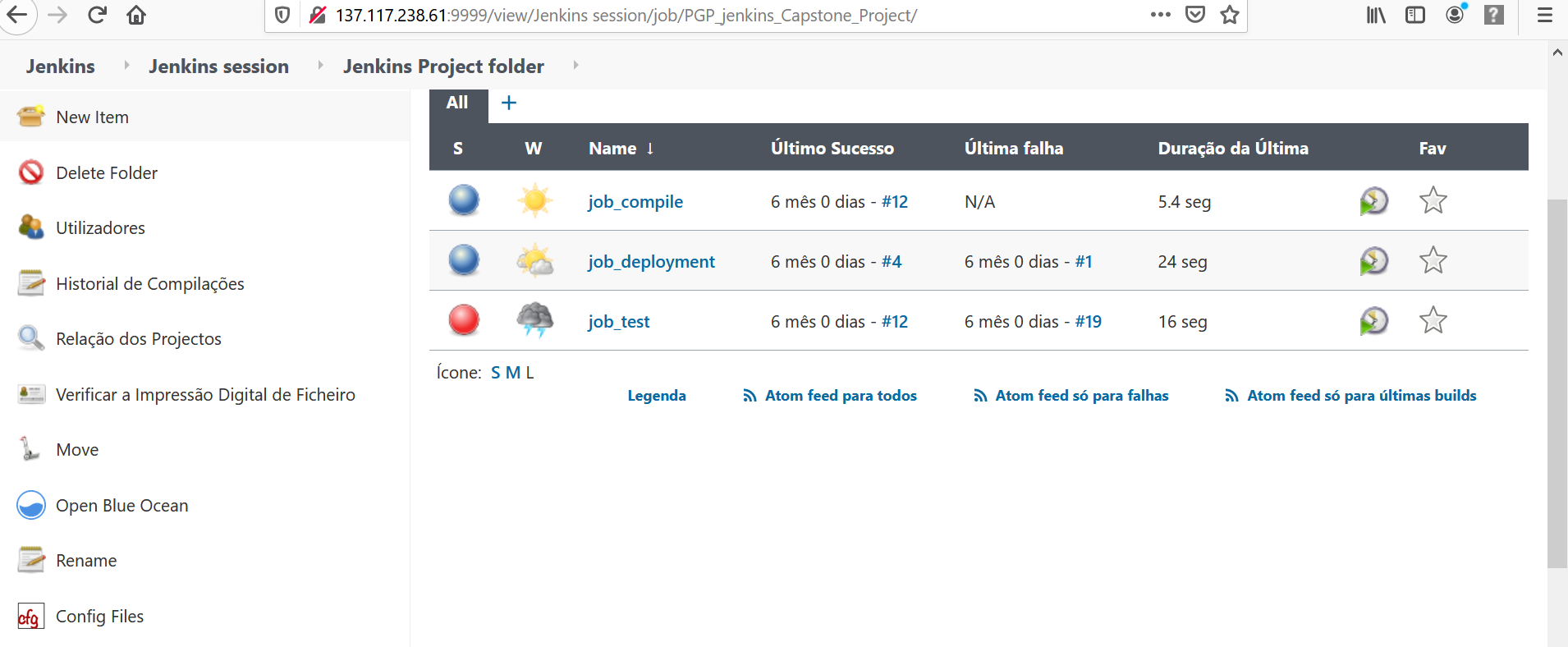


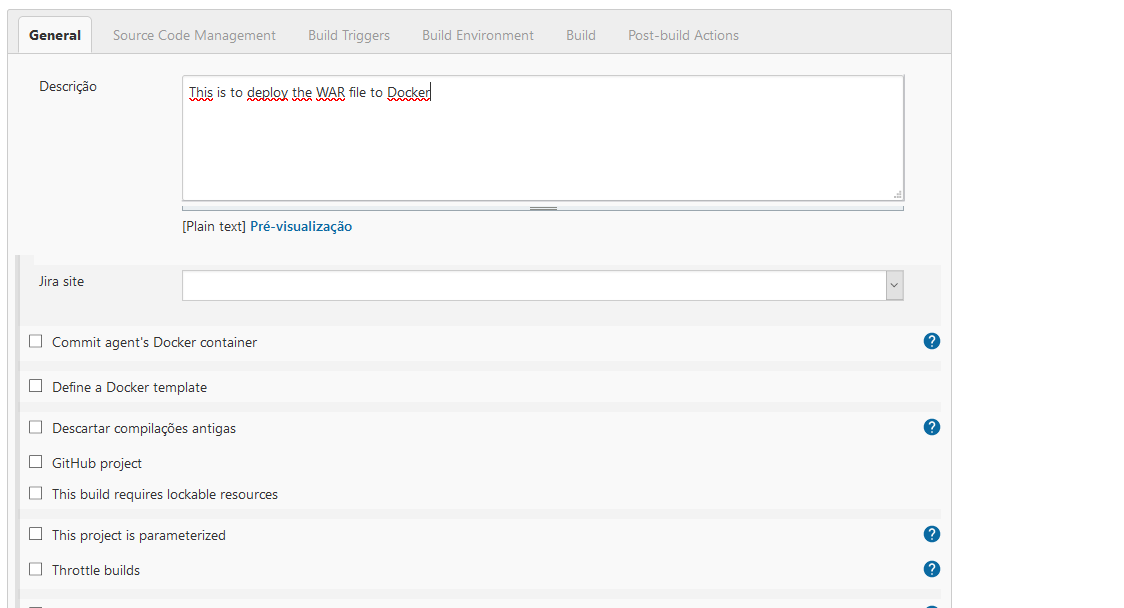
There is already some unit tests created:

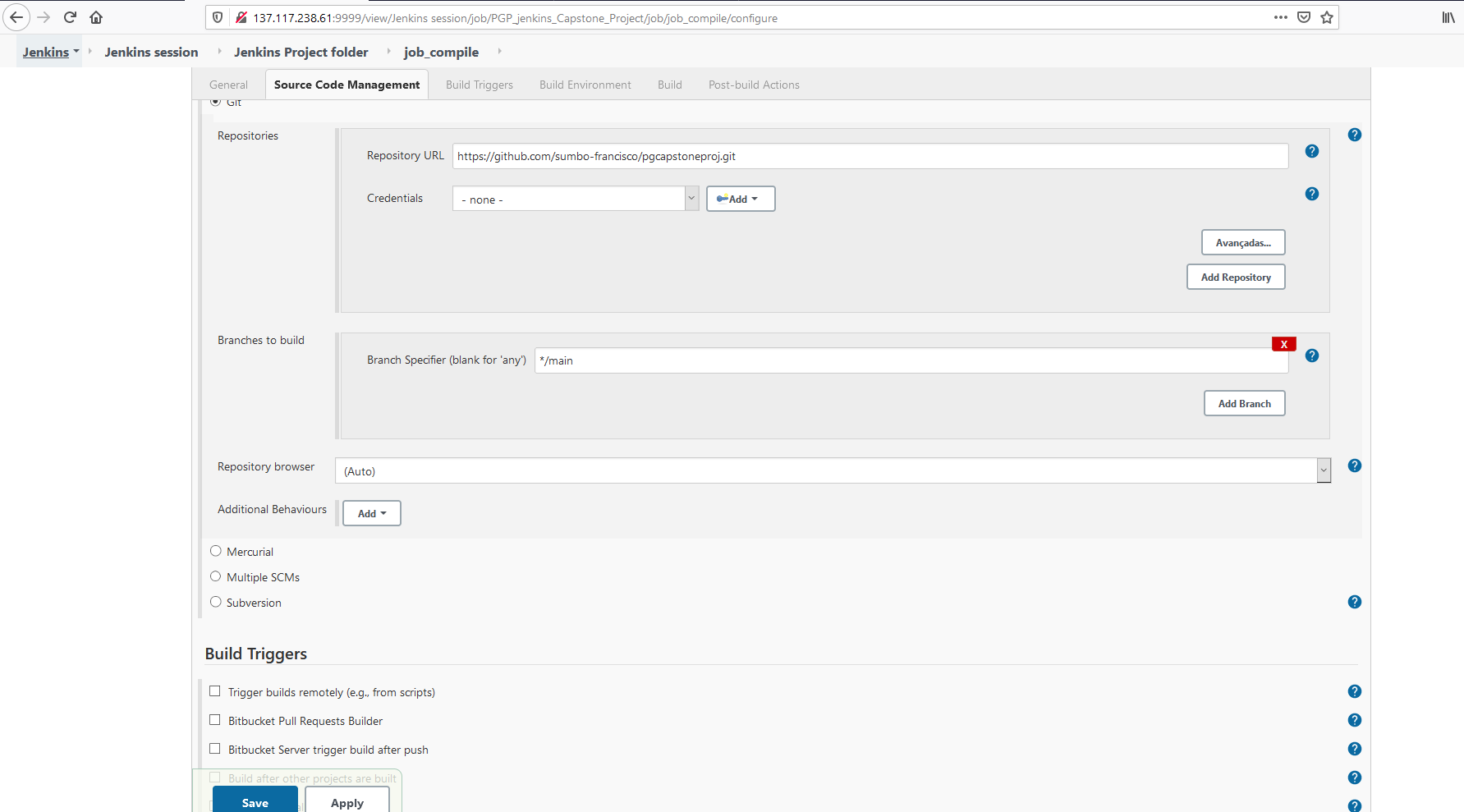


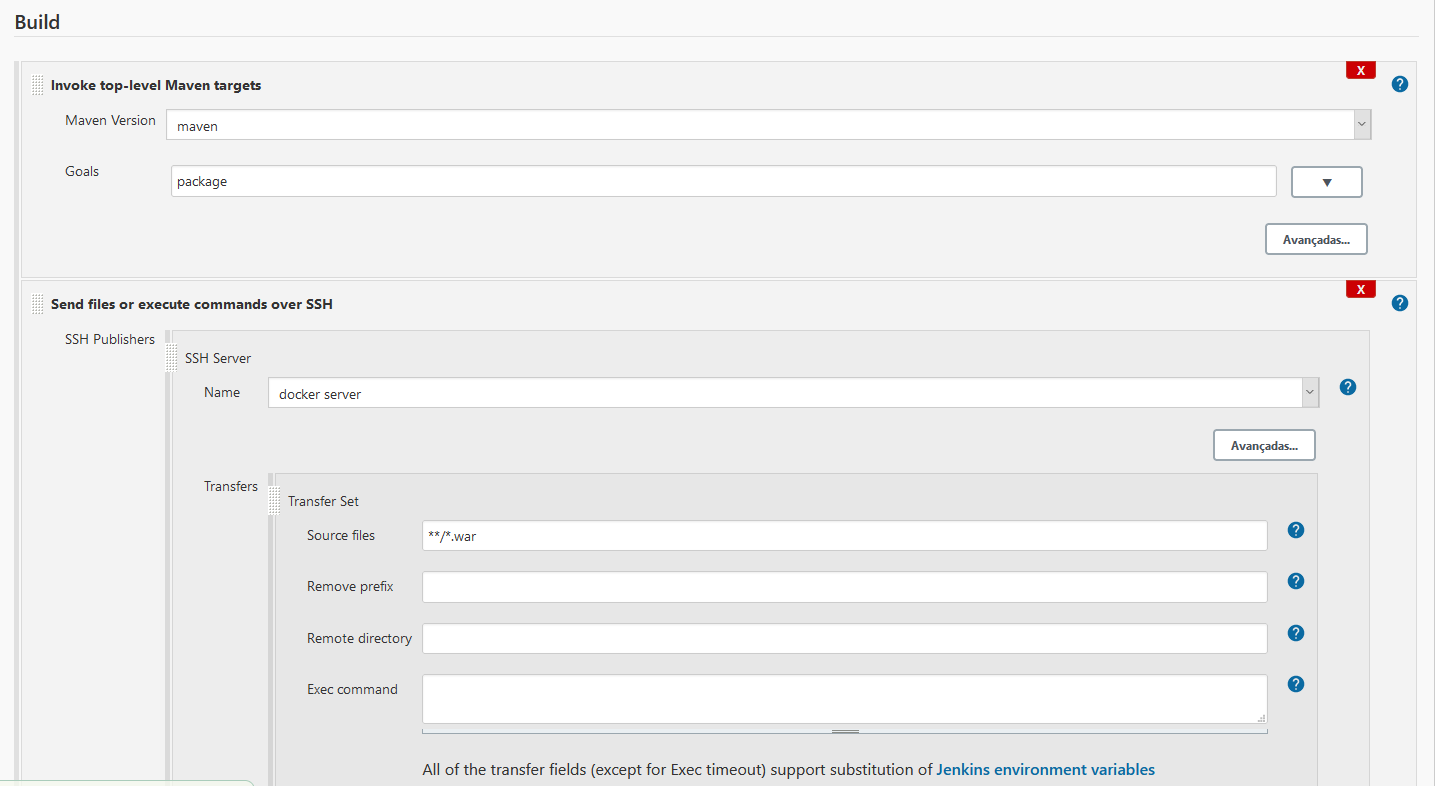
## 4.1 Build and Deploy Web application in a Tomcat Docker container with Jenkins

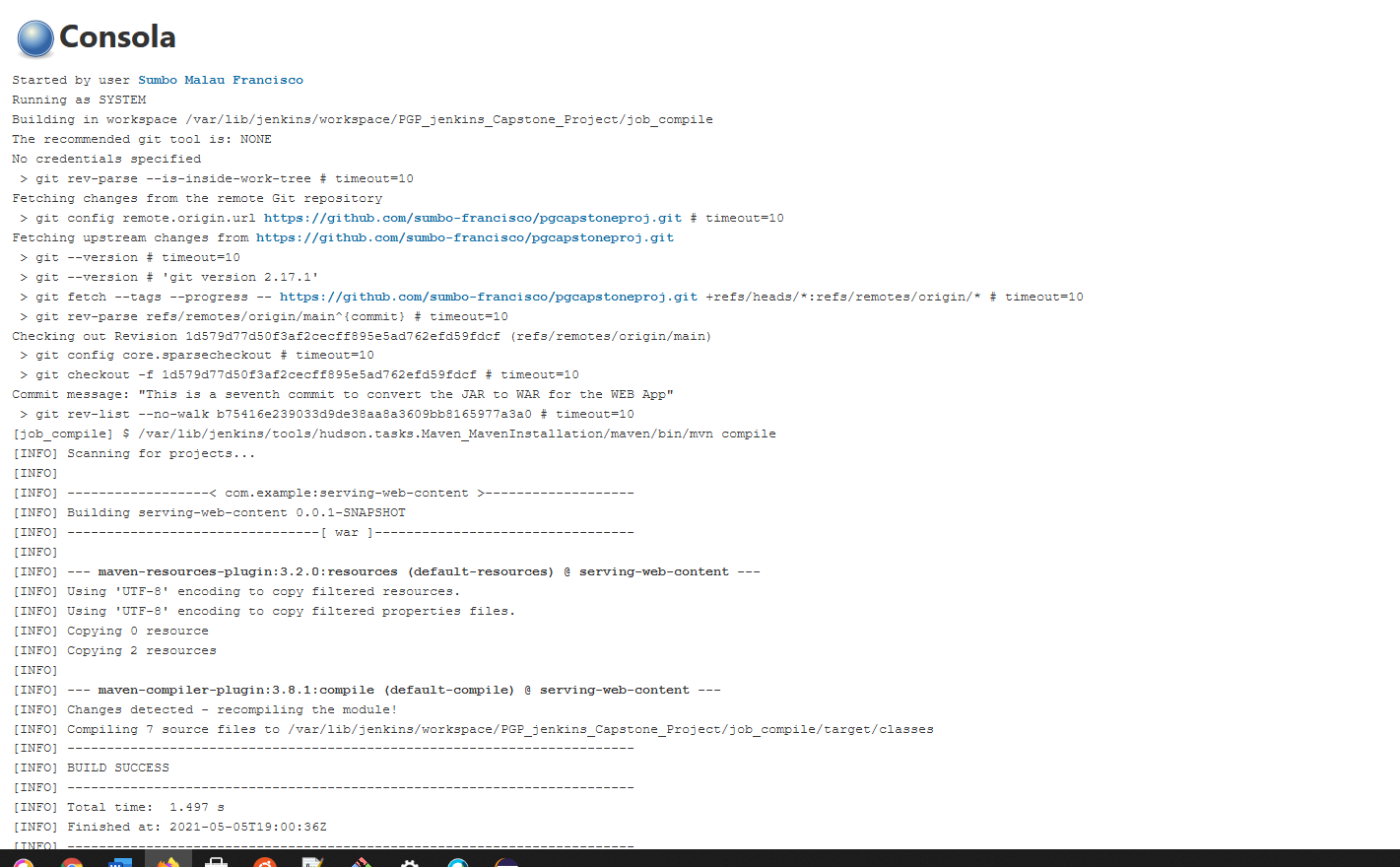
I created 3 jobs in Jenkins to build, compile the APP, and Deploy in a Tomcat Container:

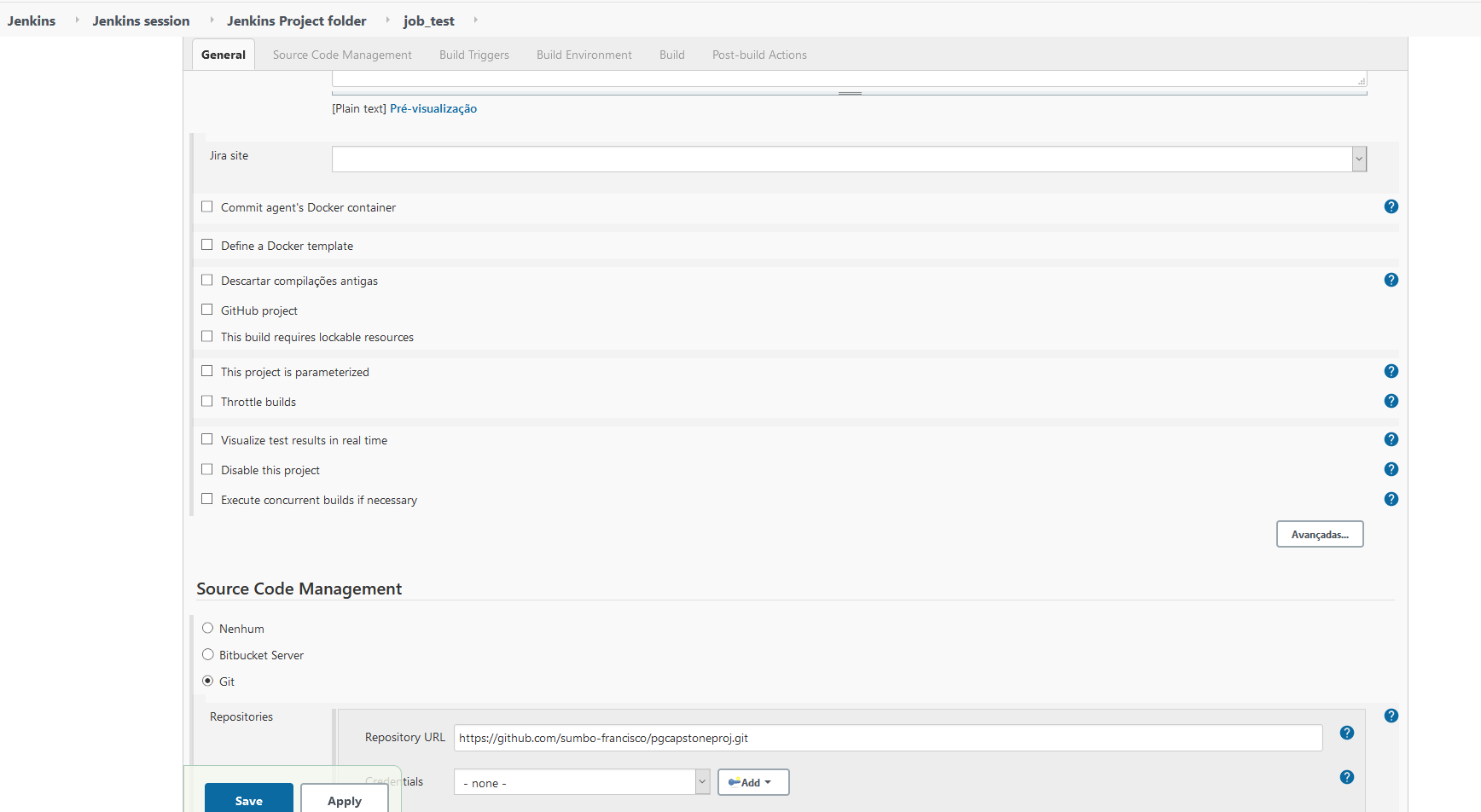


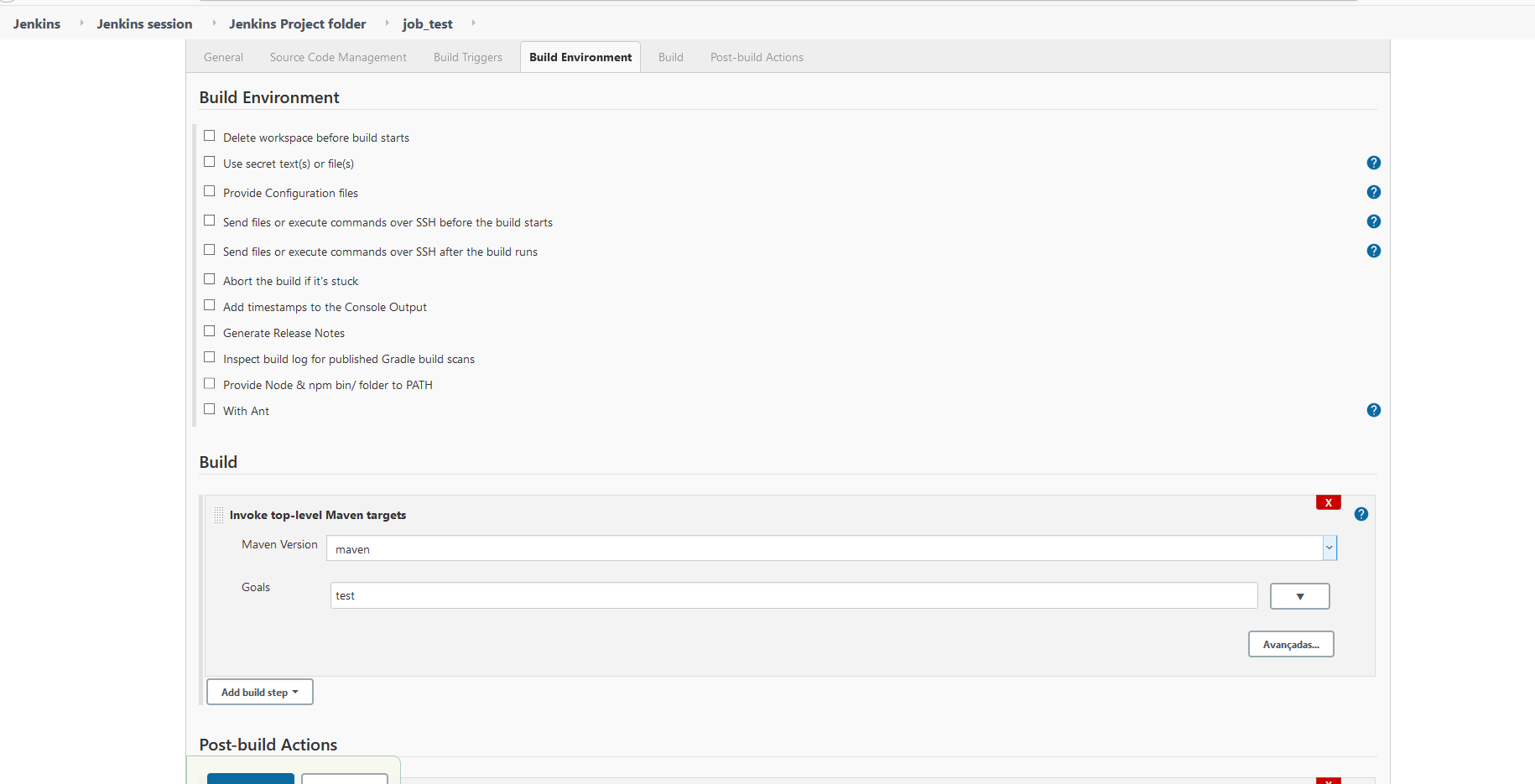


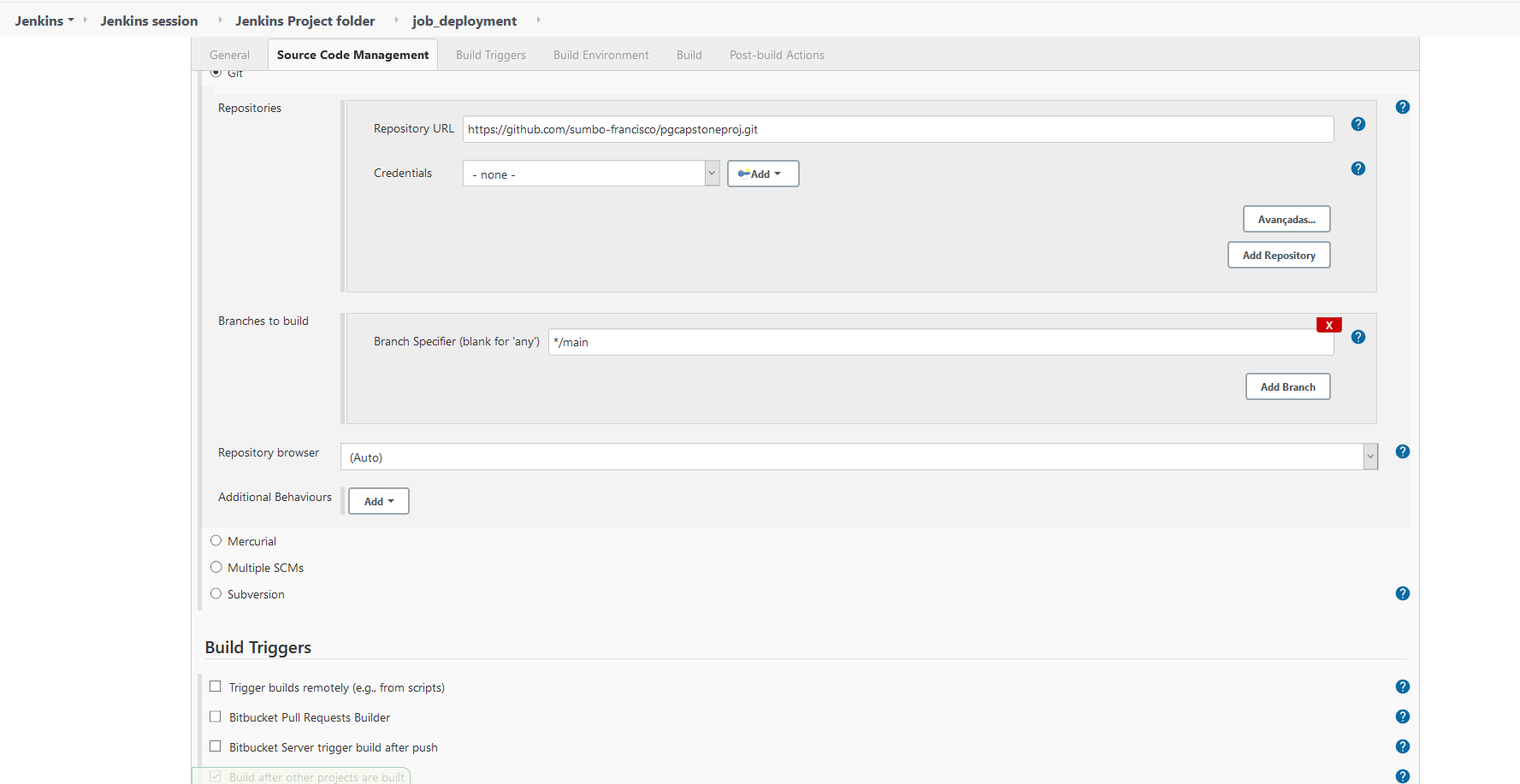


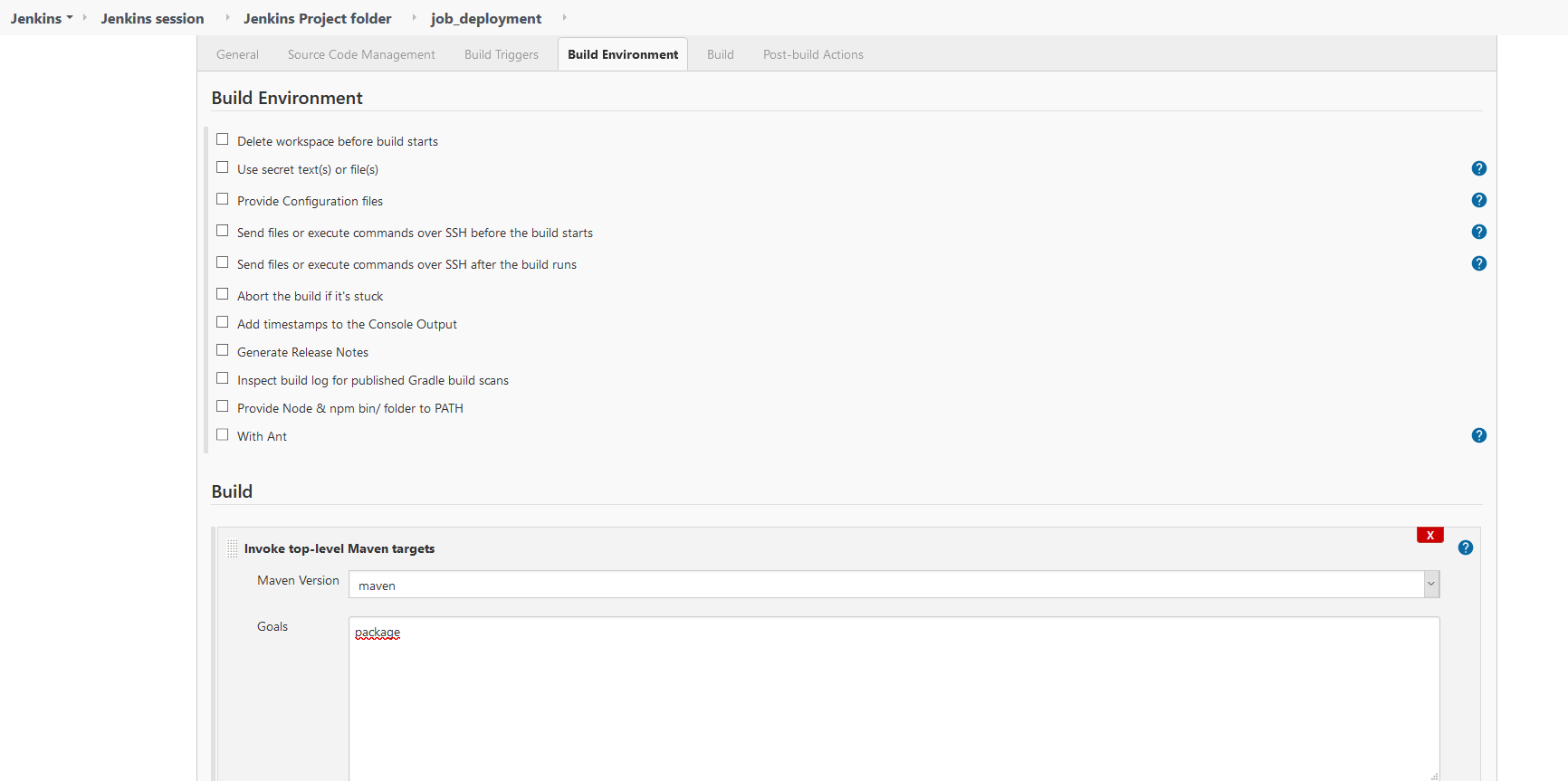


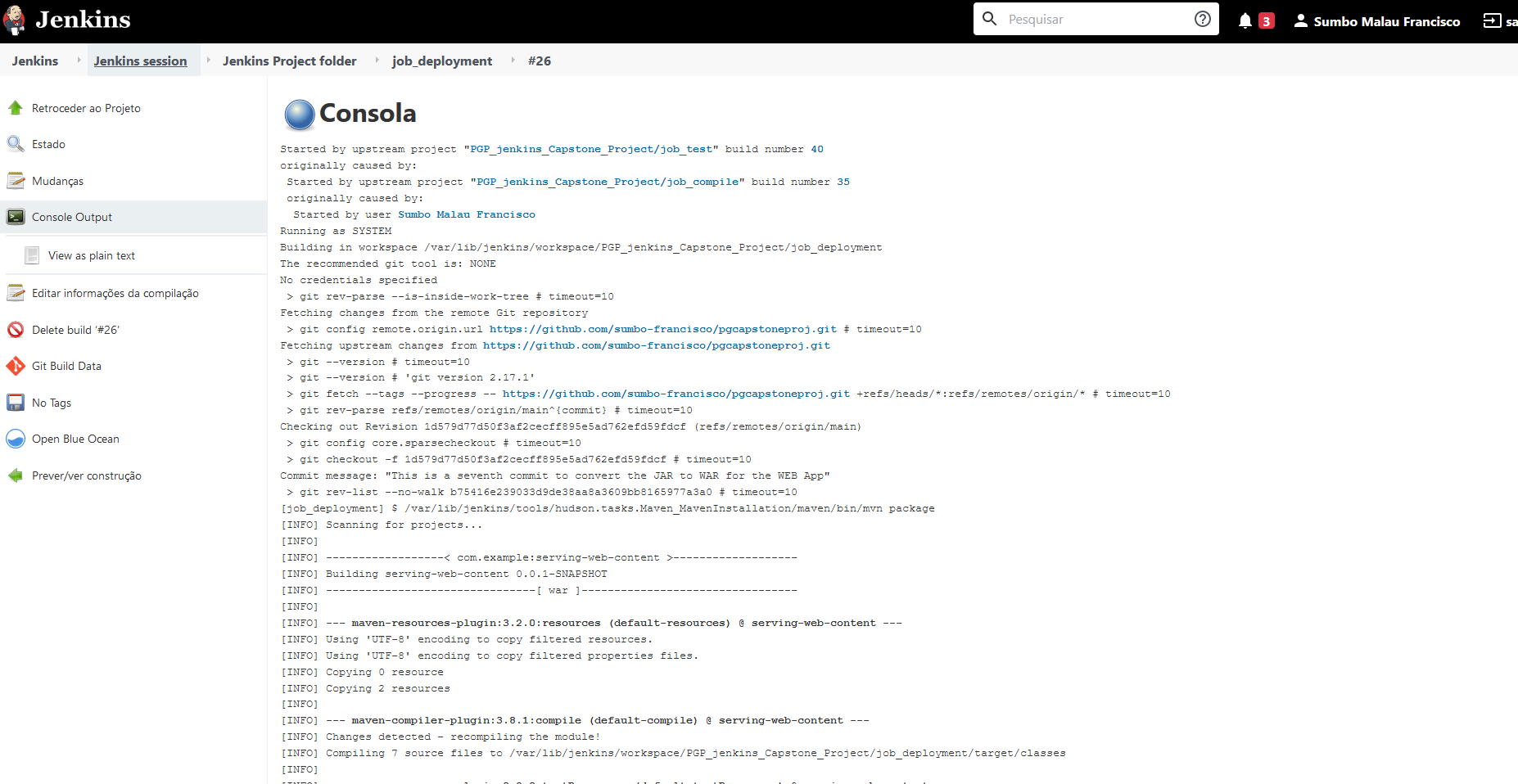


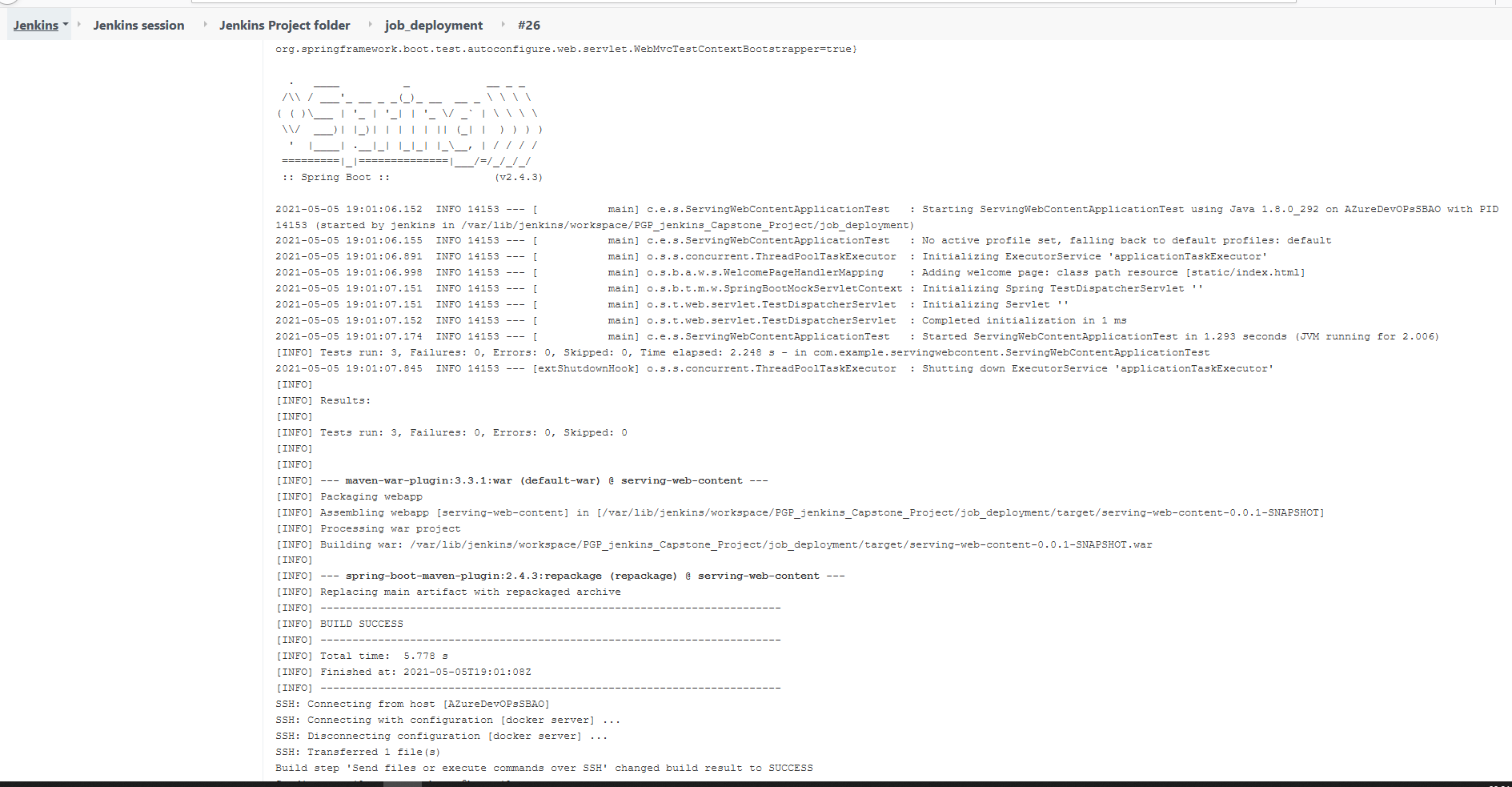




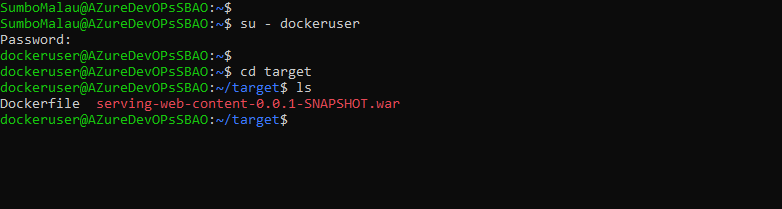




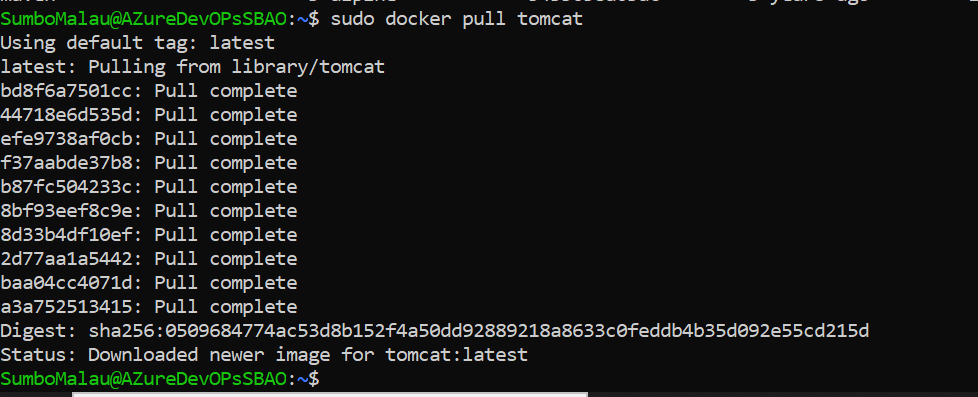




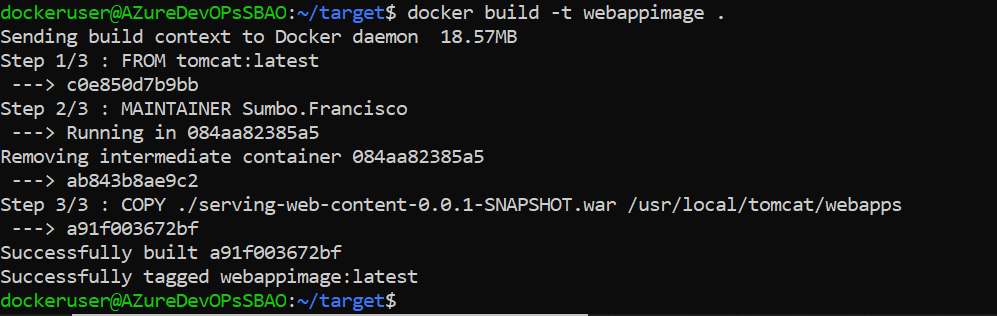
I have successfully built an deploy the Web application from the Jenkins server and moved to the Docker server:

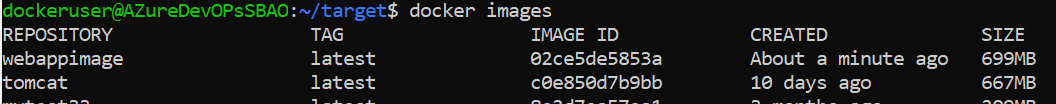


## 4.2 Create the Docker Image (Dockerfile) and Publish on Repository



We will use a Dockerfile to create our image, from a tomcat base image:



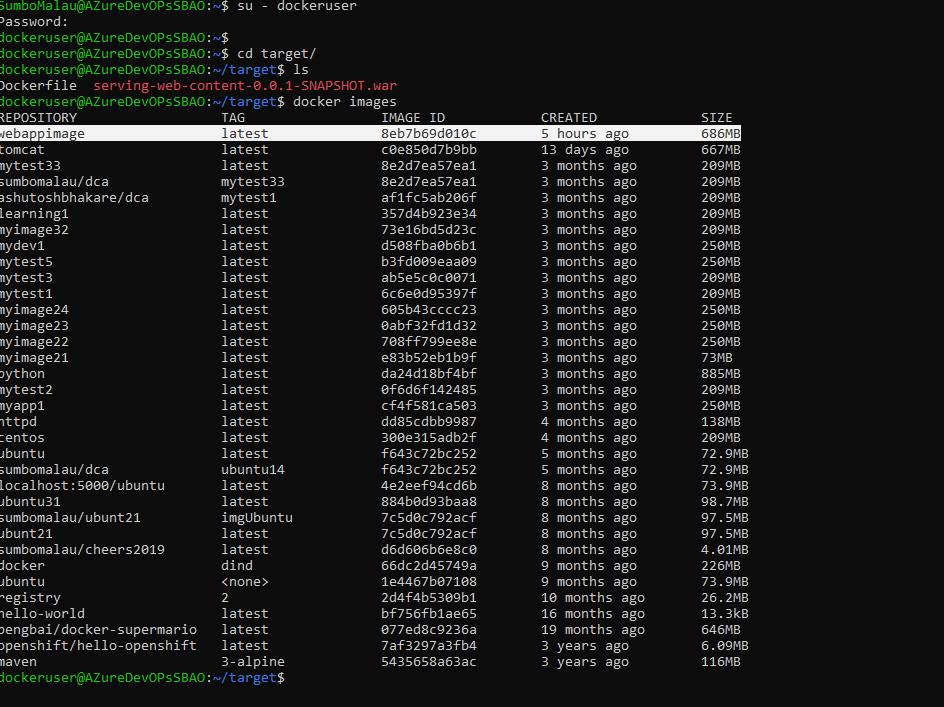


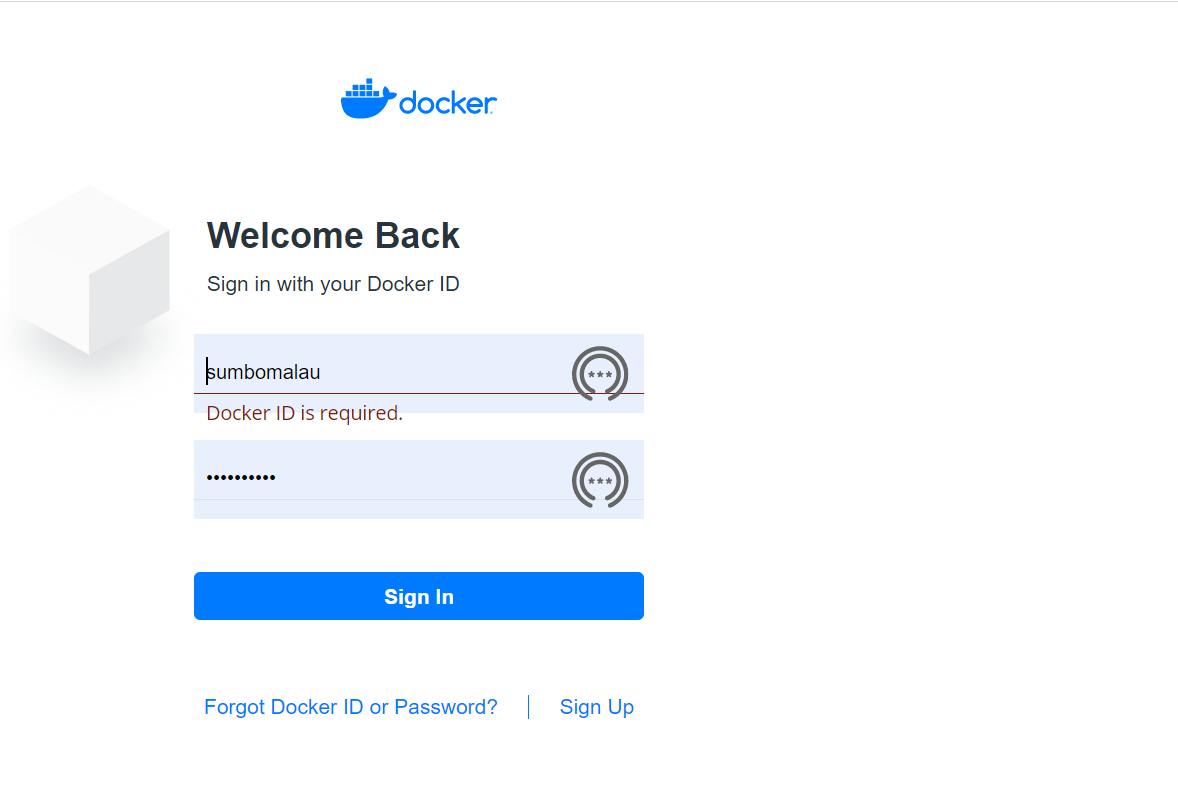
We can test our image application run in a container:

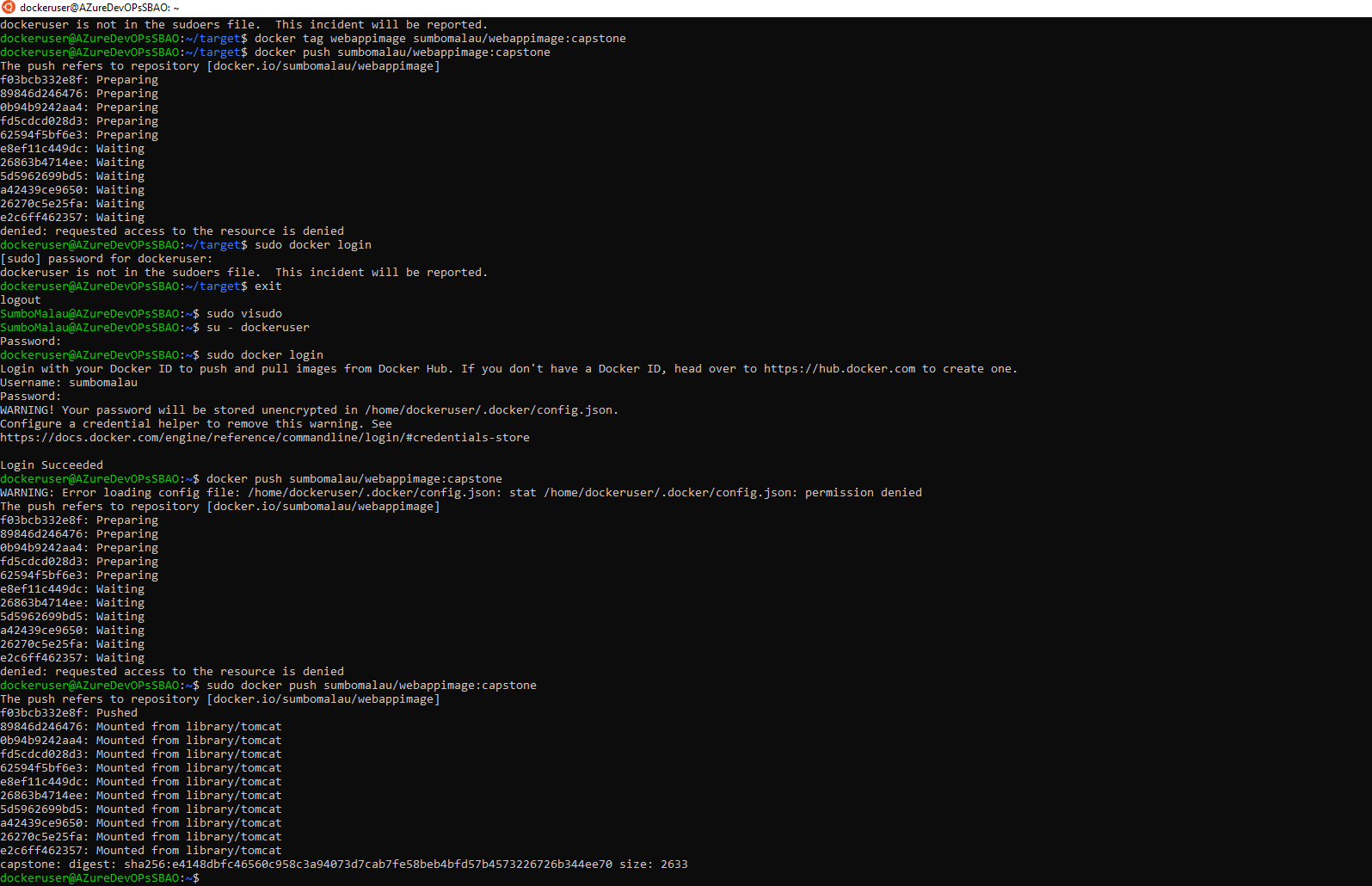
docker run -d --name tomcat-app -p 8082:8080 webappimage

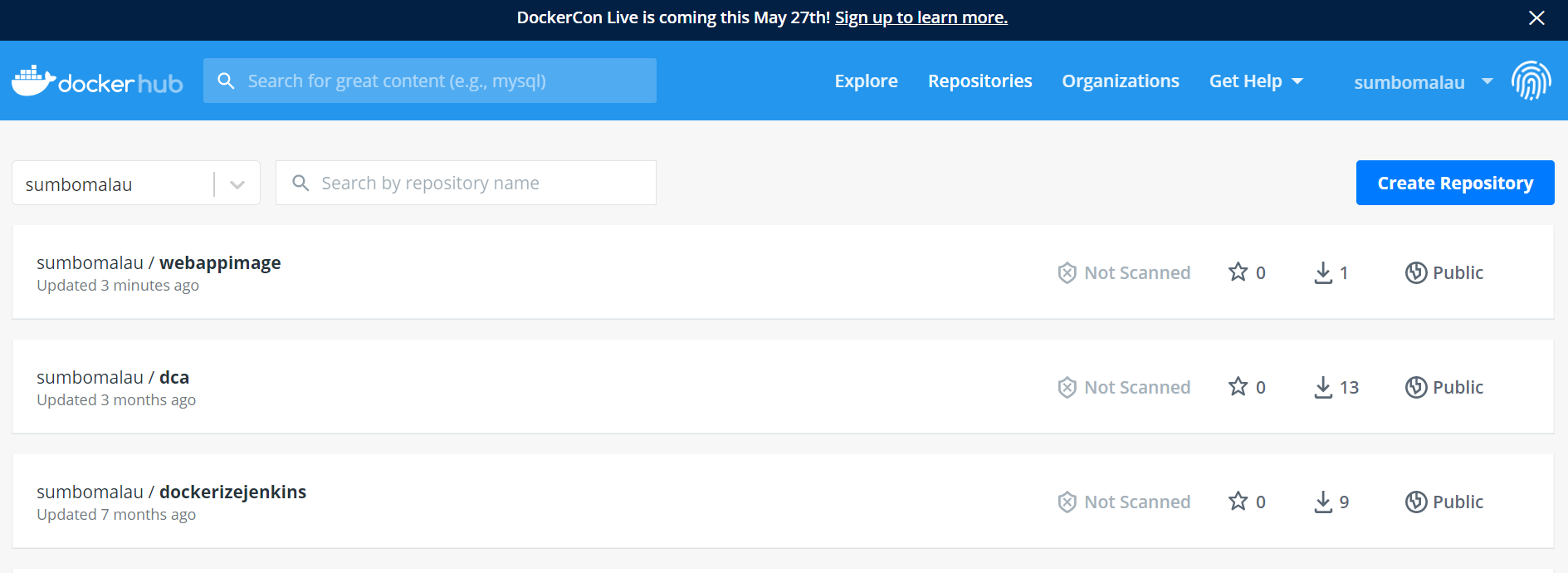
## Publish and make available image in the Docker Registry

Our image name is **webappimage**:



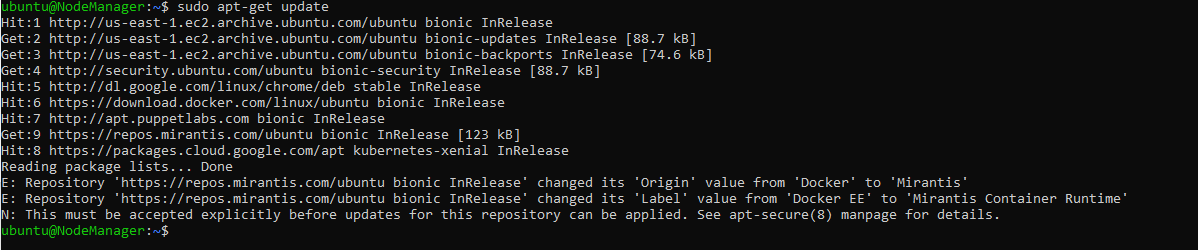


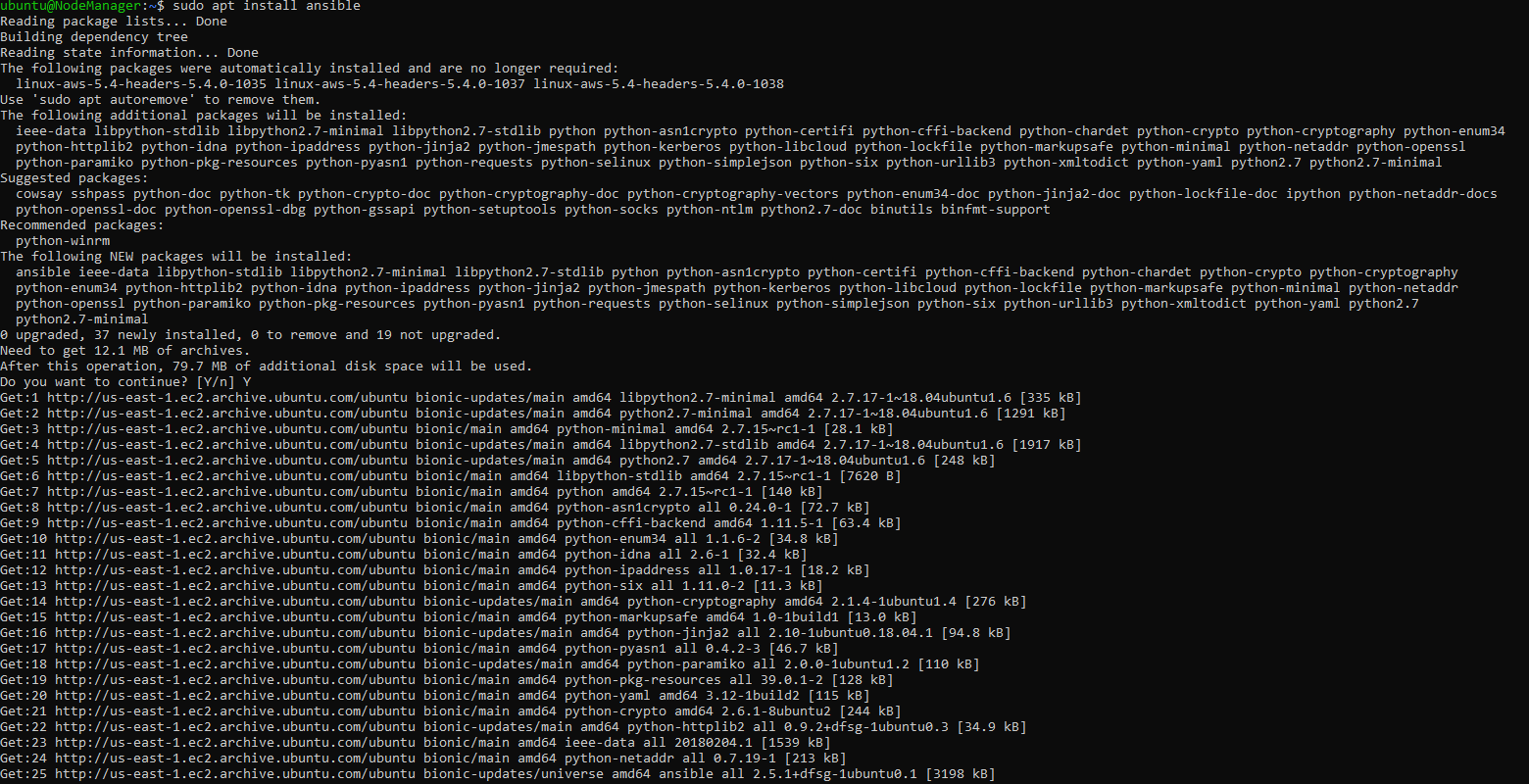


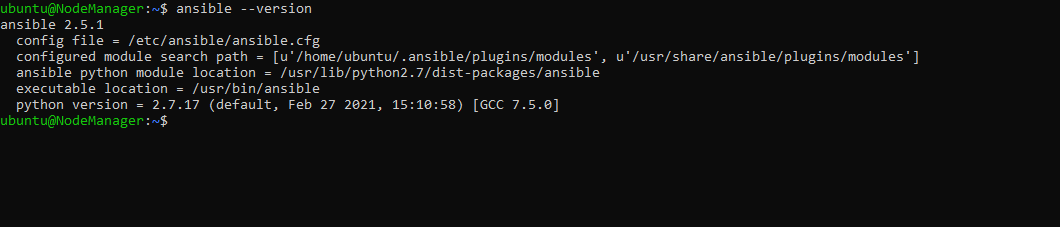


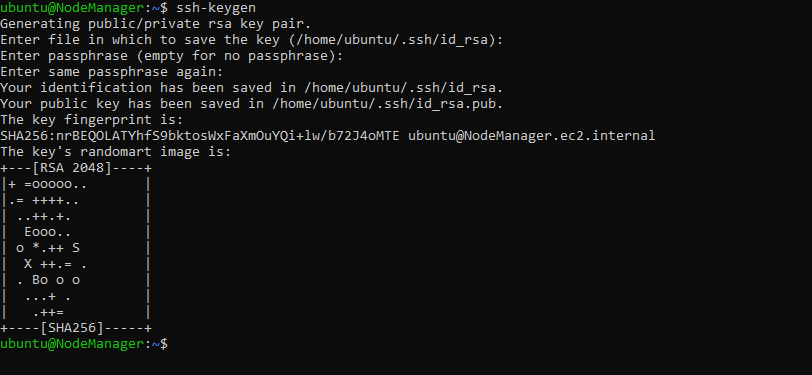
# 5 Automation of AWS EC2 instance provisioning in ANSIBLE

## 5.1 Ansible Installation:

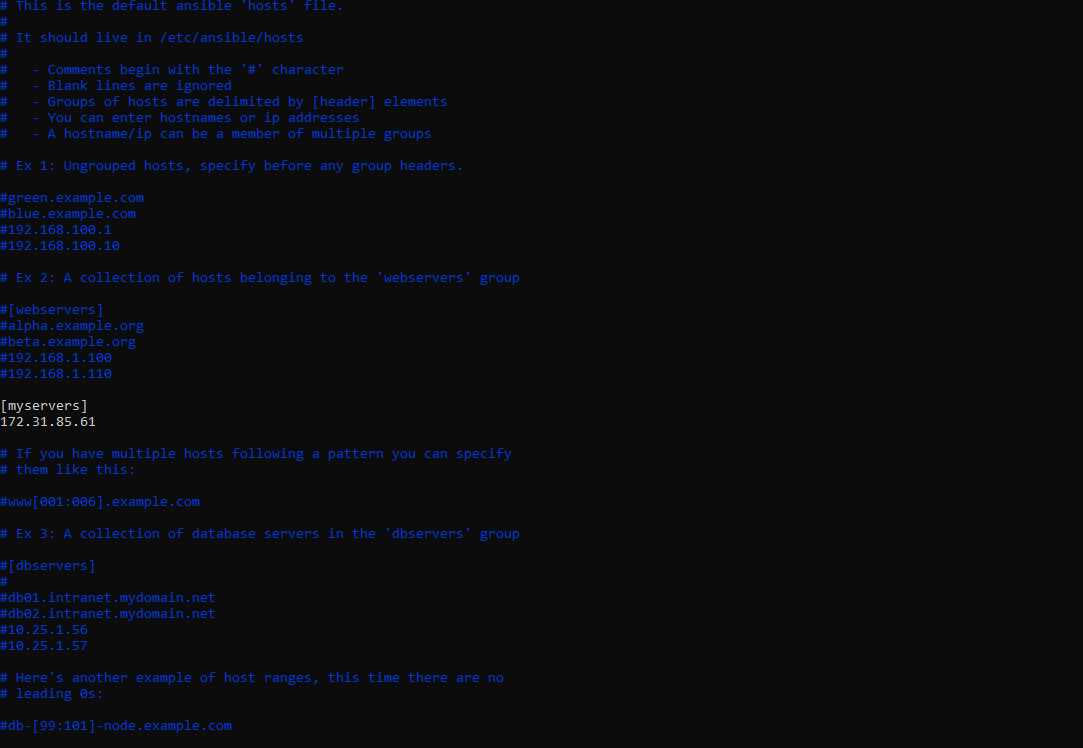




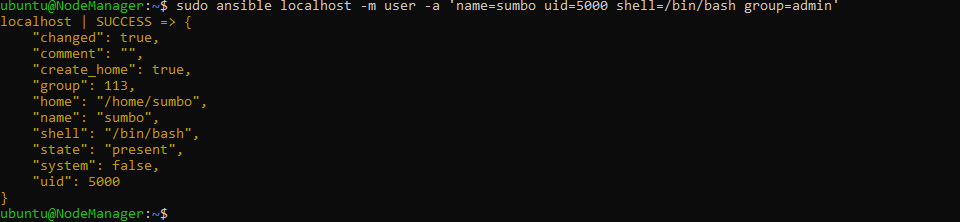




The iventory:

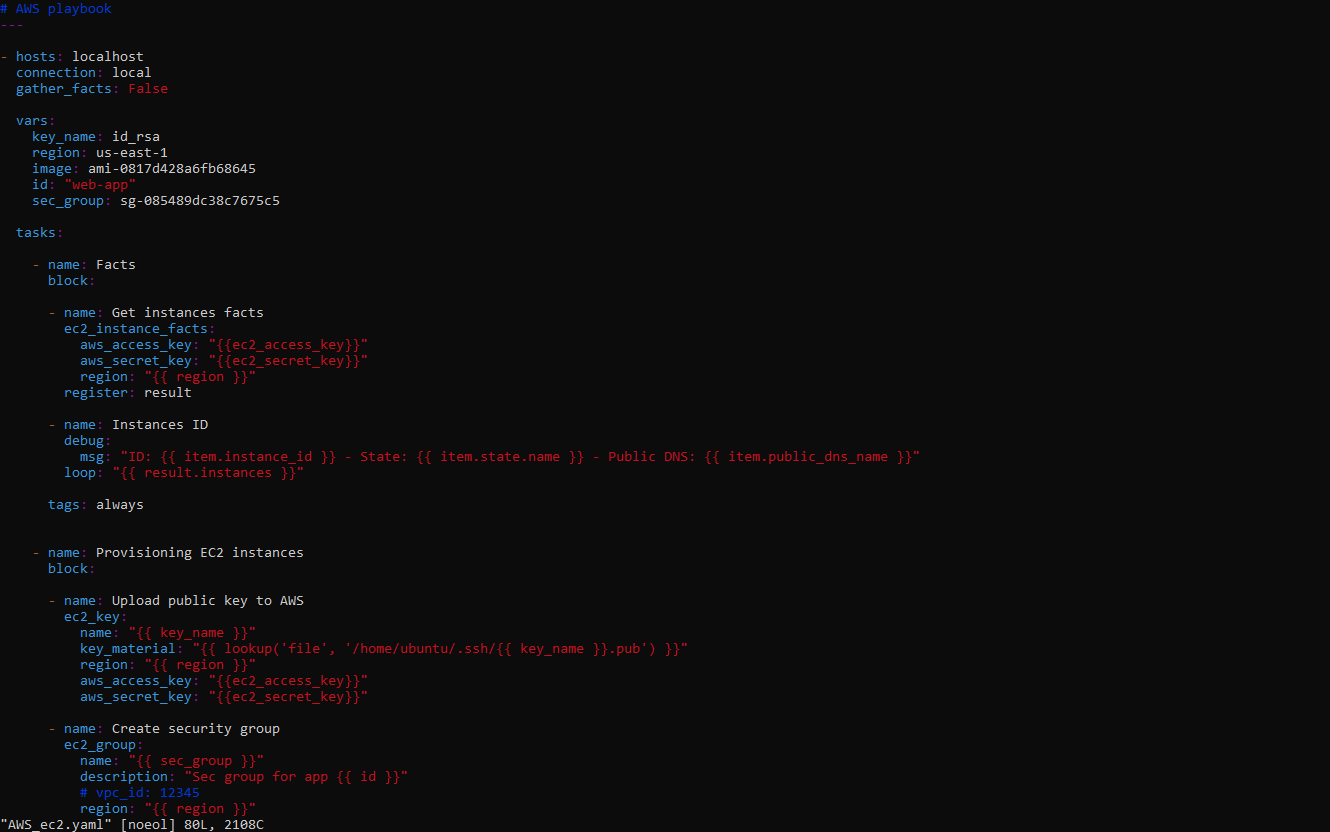


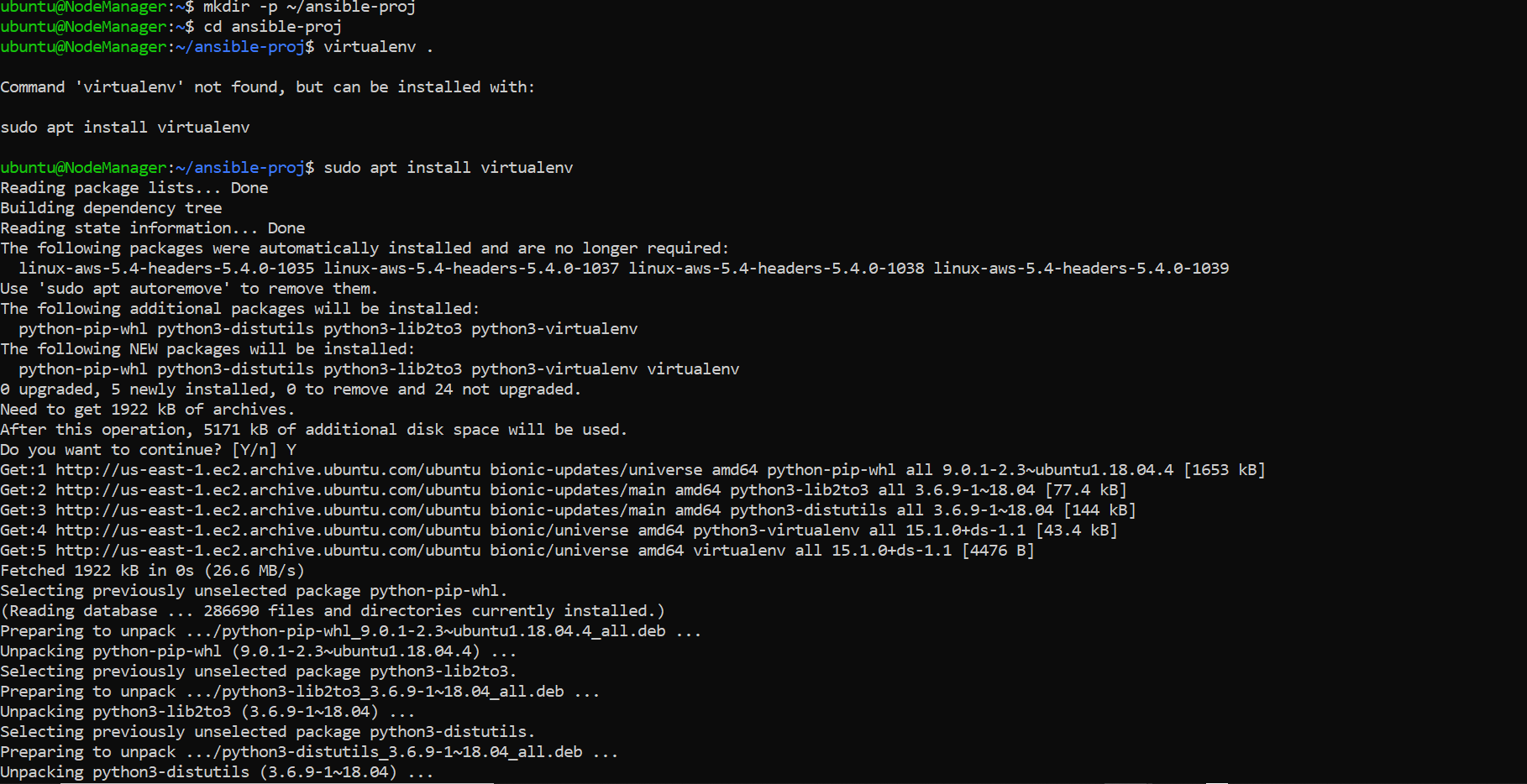
Test Ansible:

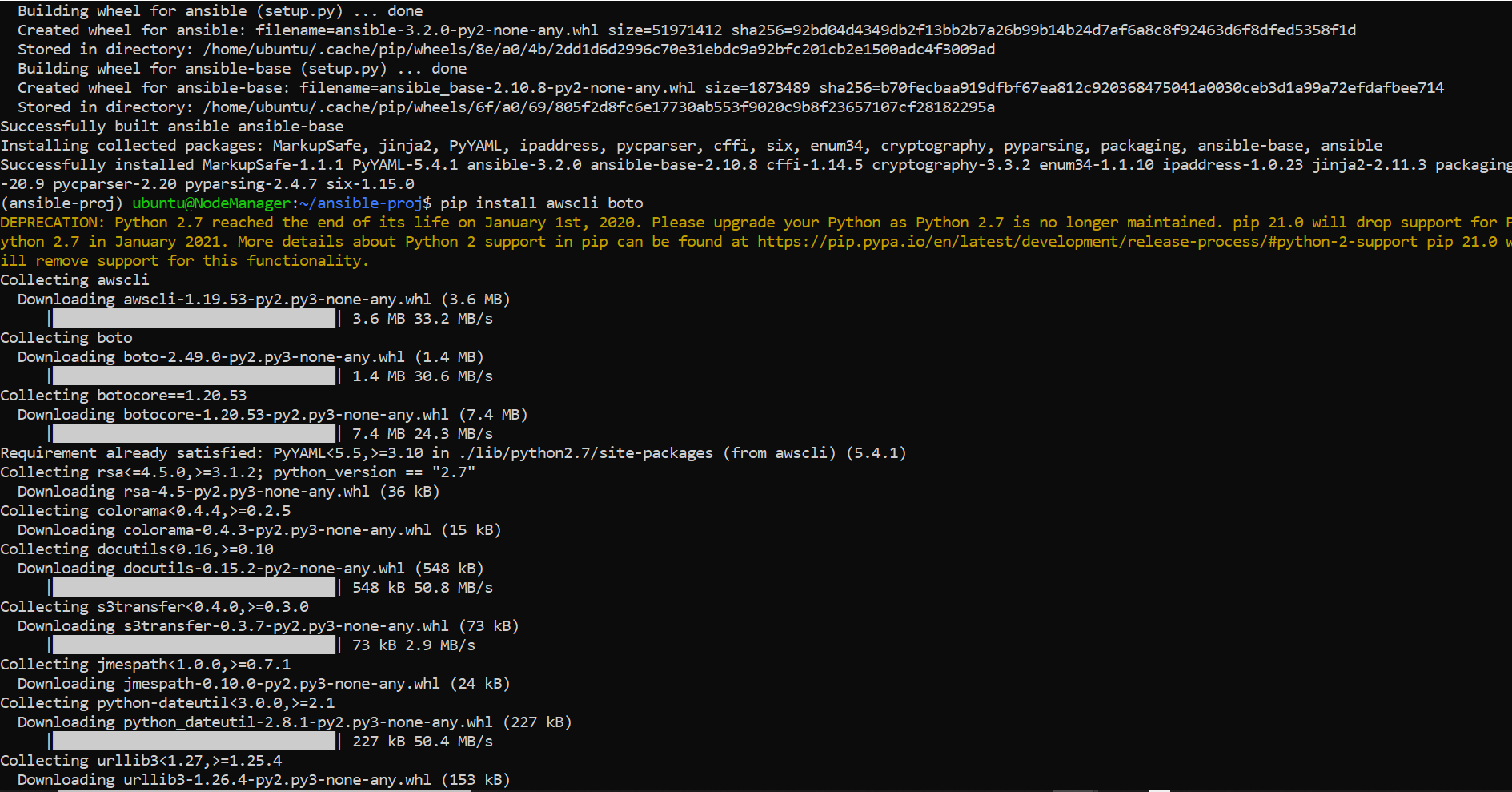


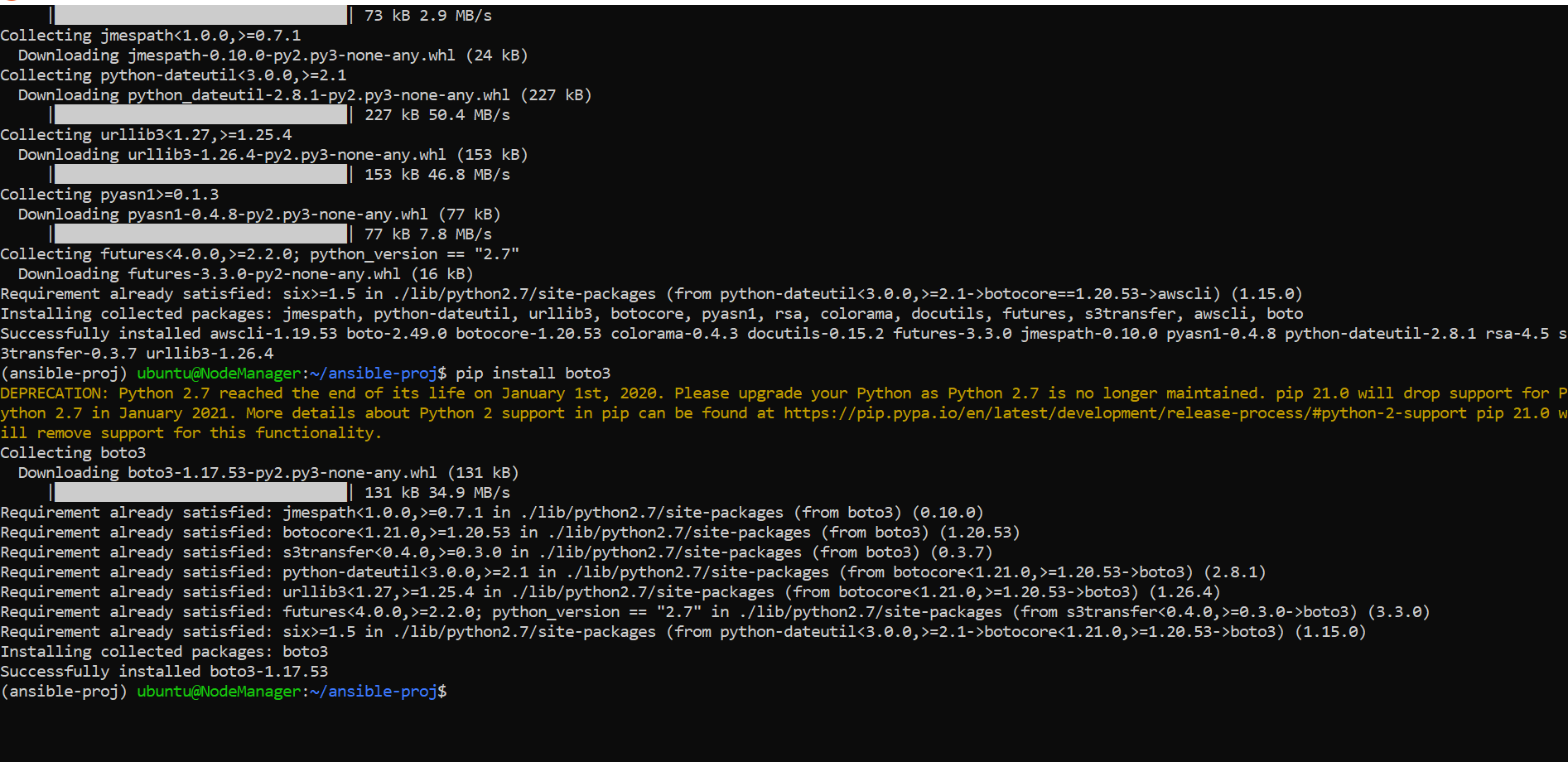
## 5.2 Create the playbook to automate the provision of 3 AWS ec2 instances

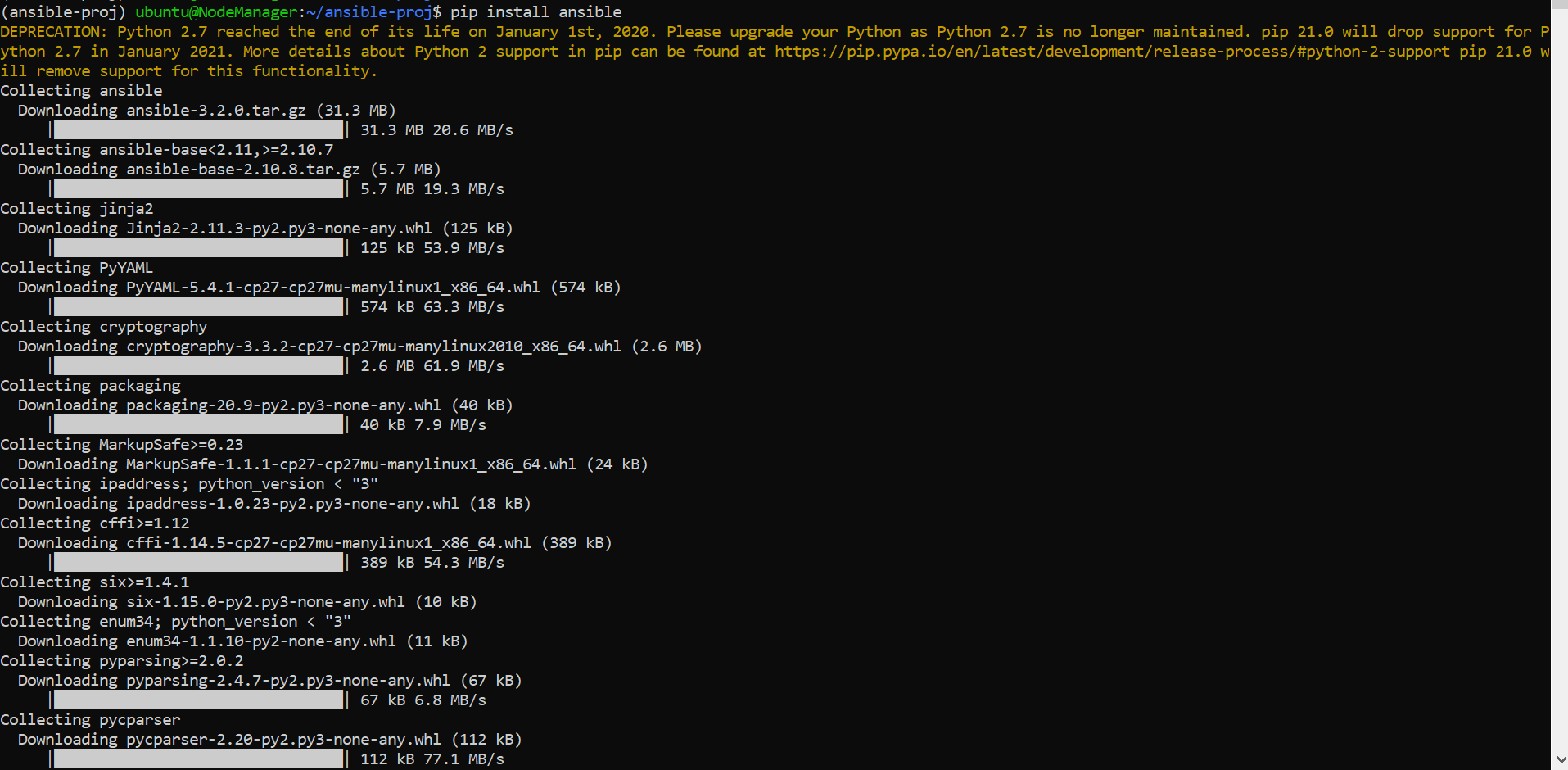
The playbook is created in yaml file named AWS\_ec2.yaml:

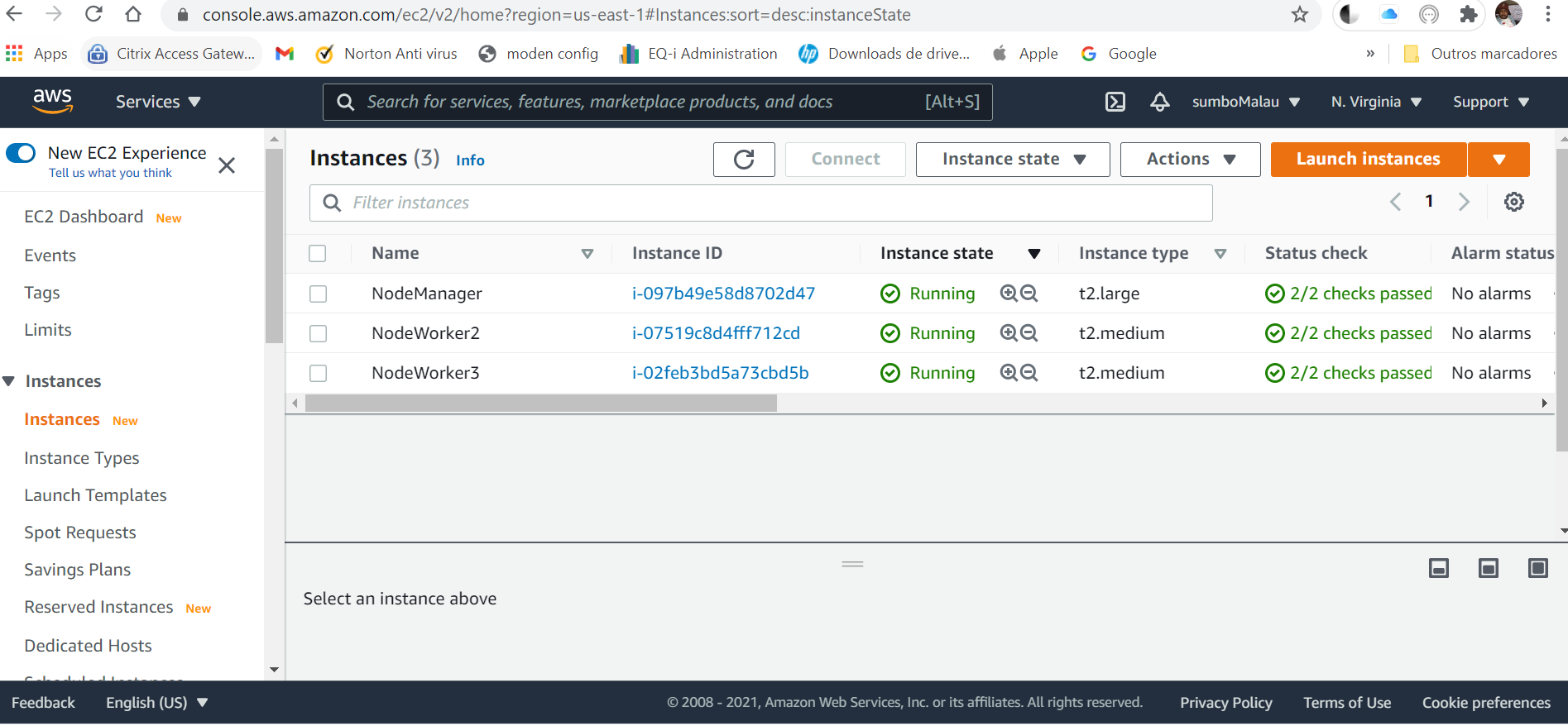






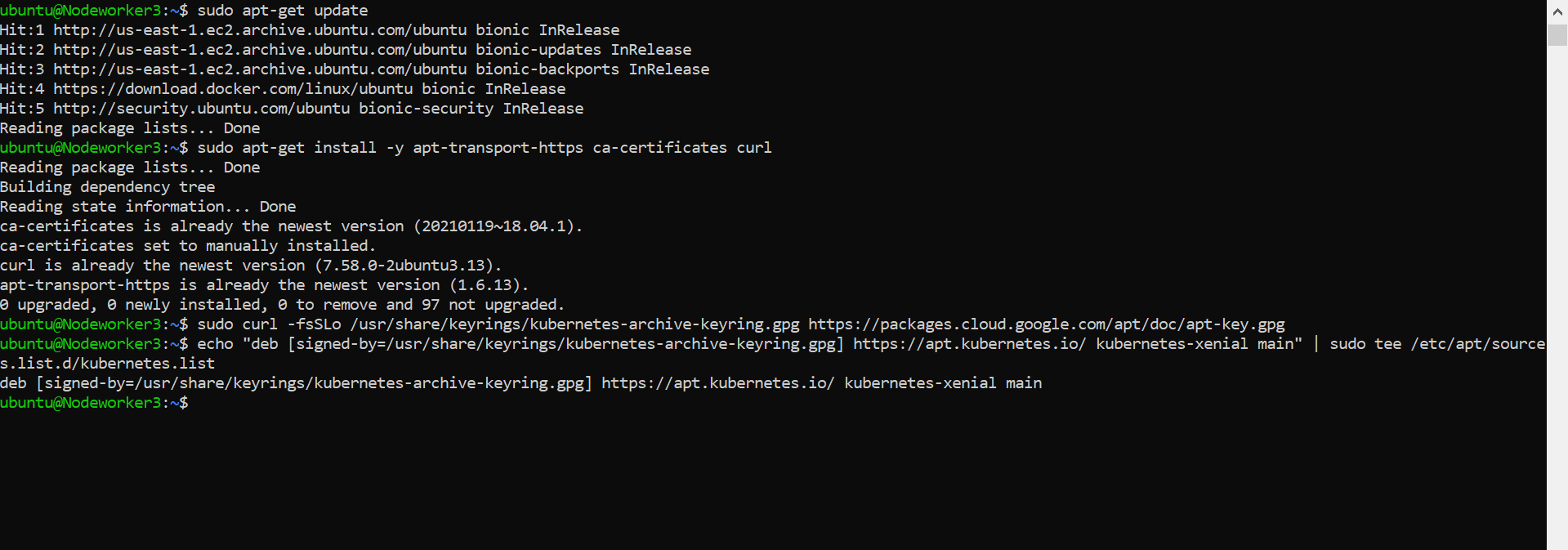


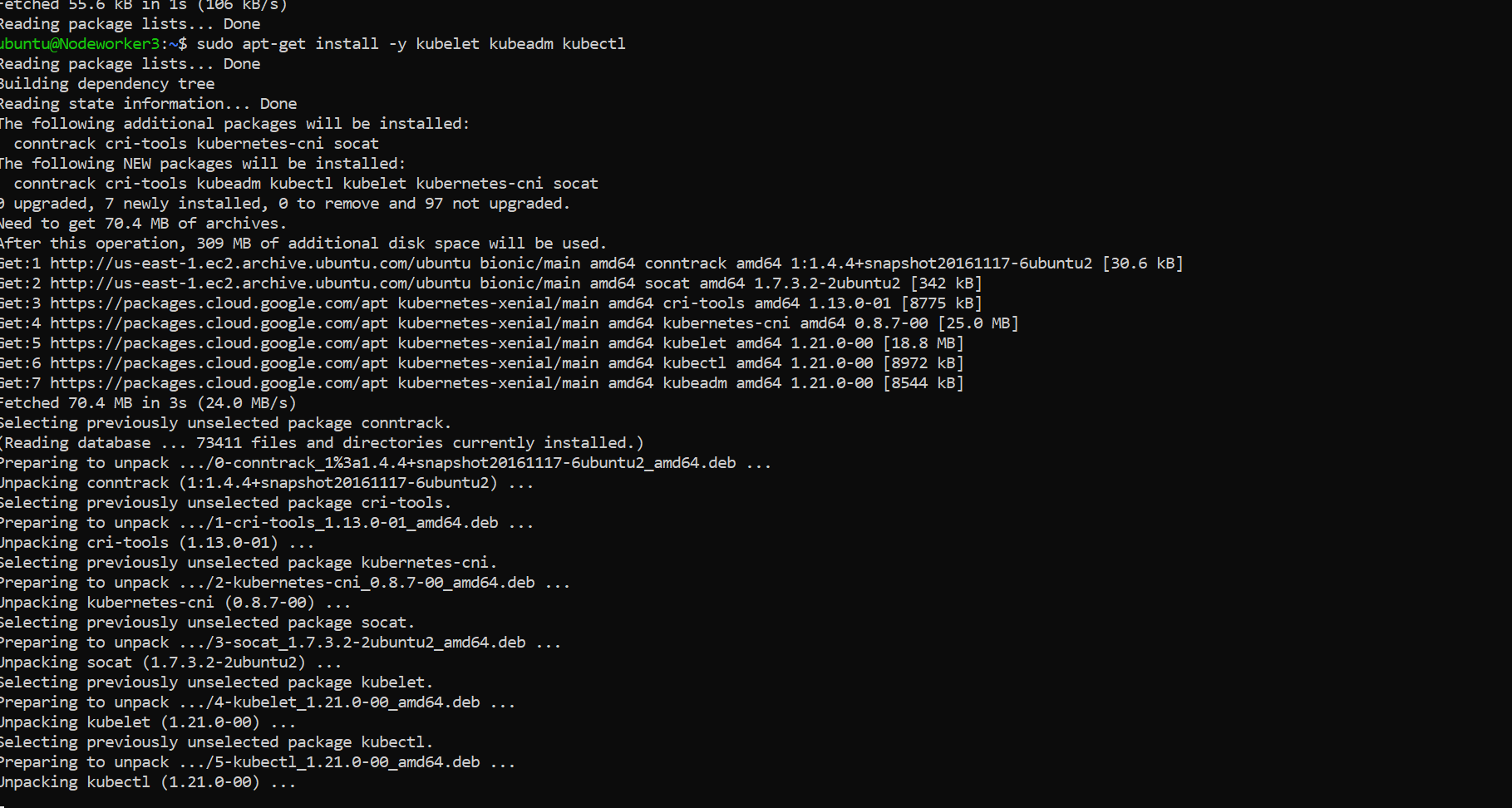


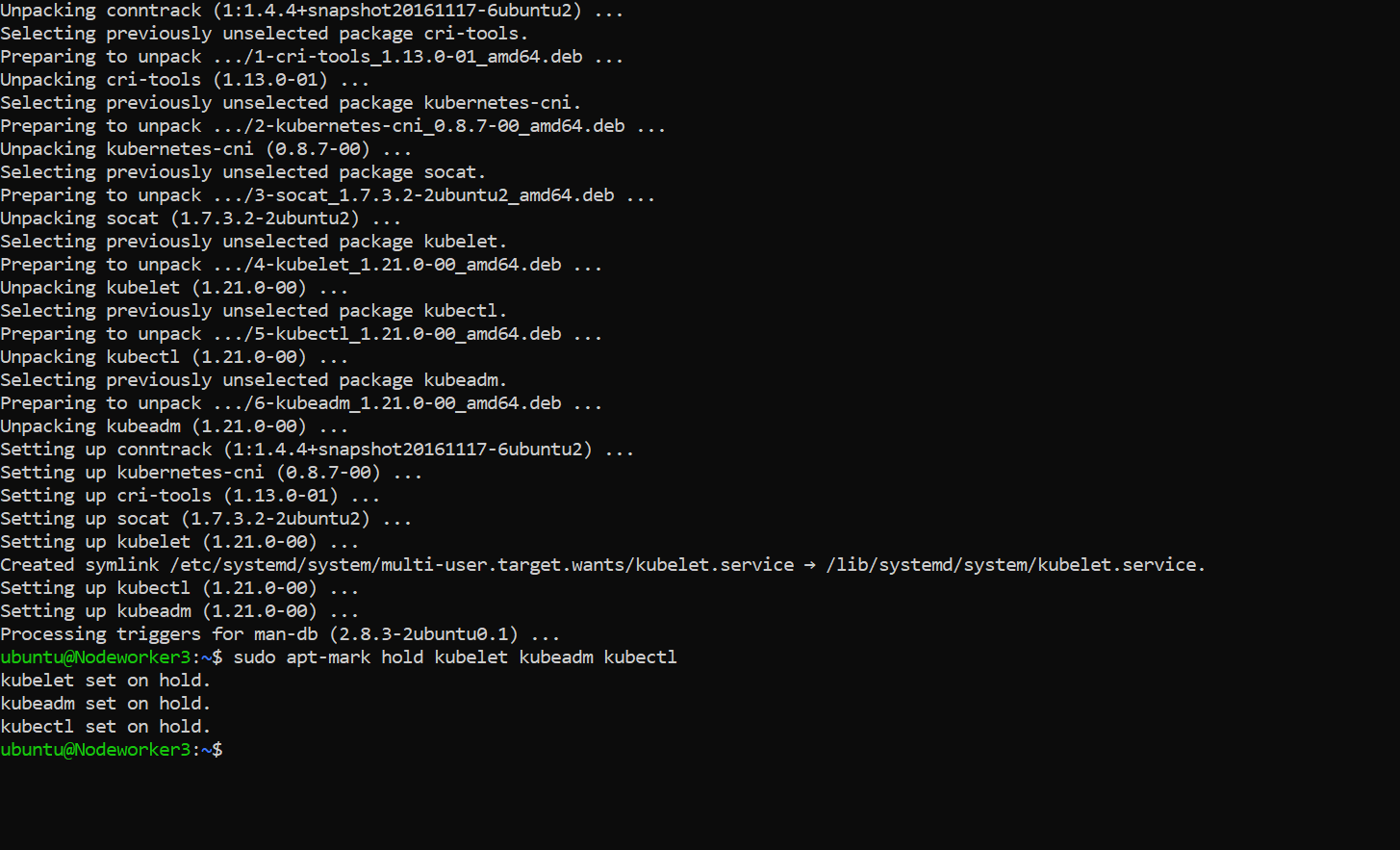


# 6 Kubernetes Cluster for application Deployment and HA

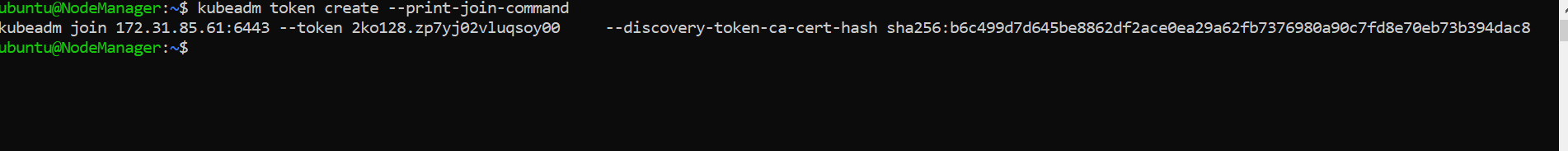
6.1 Kubernet Cluster Creation



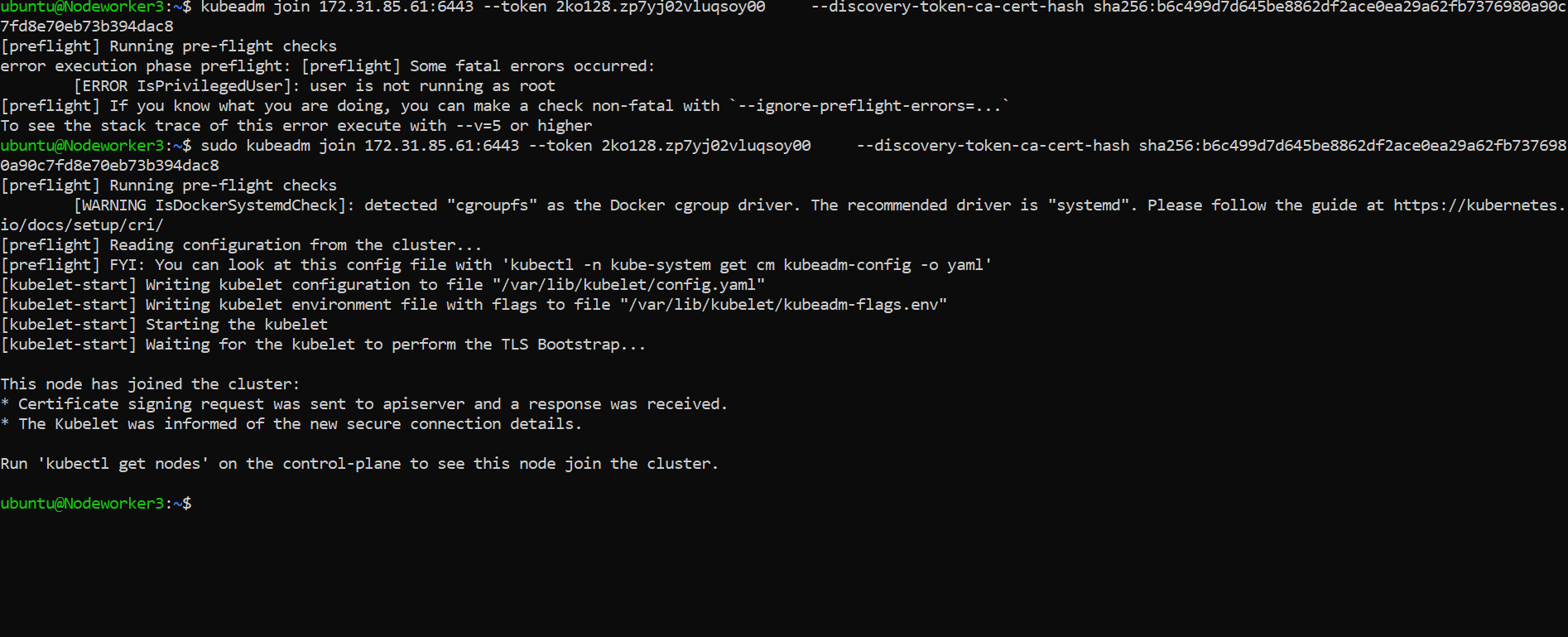


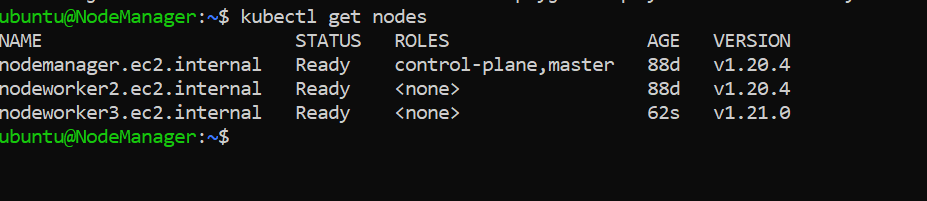


On manager node:



Adding the 2 workers to the Cluster:

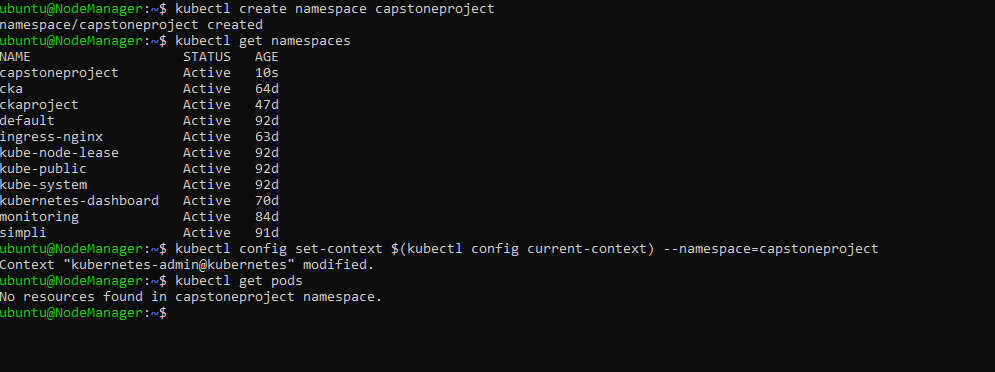




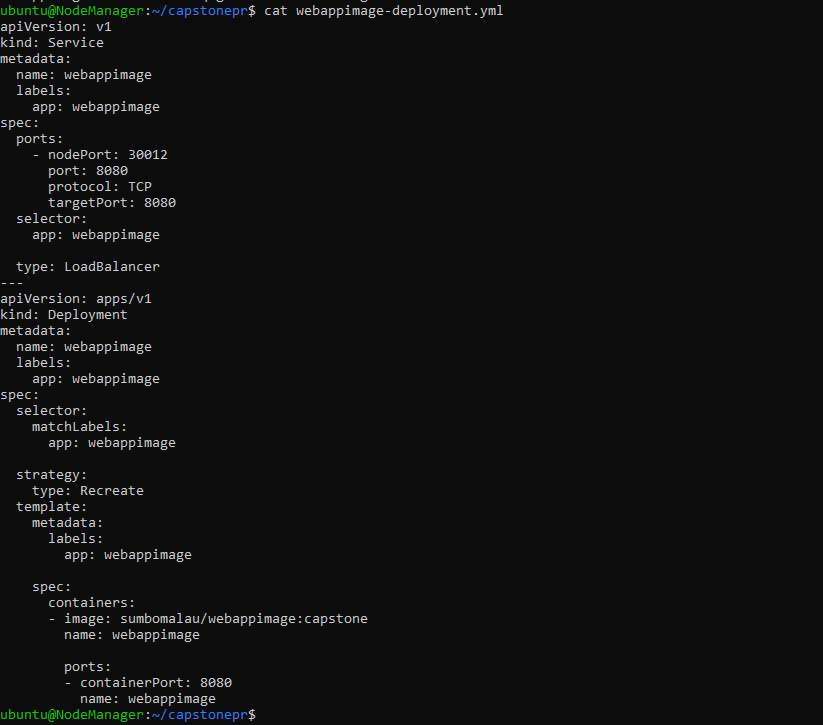


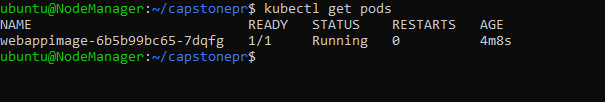
The cluster is ready for us to deploy the pods with our web application running in containers, with load balance and exposed to the users in internet.

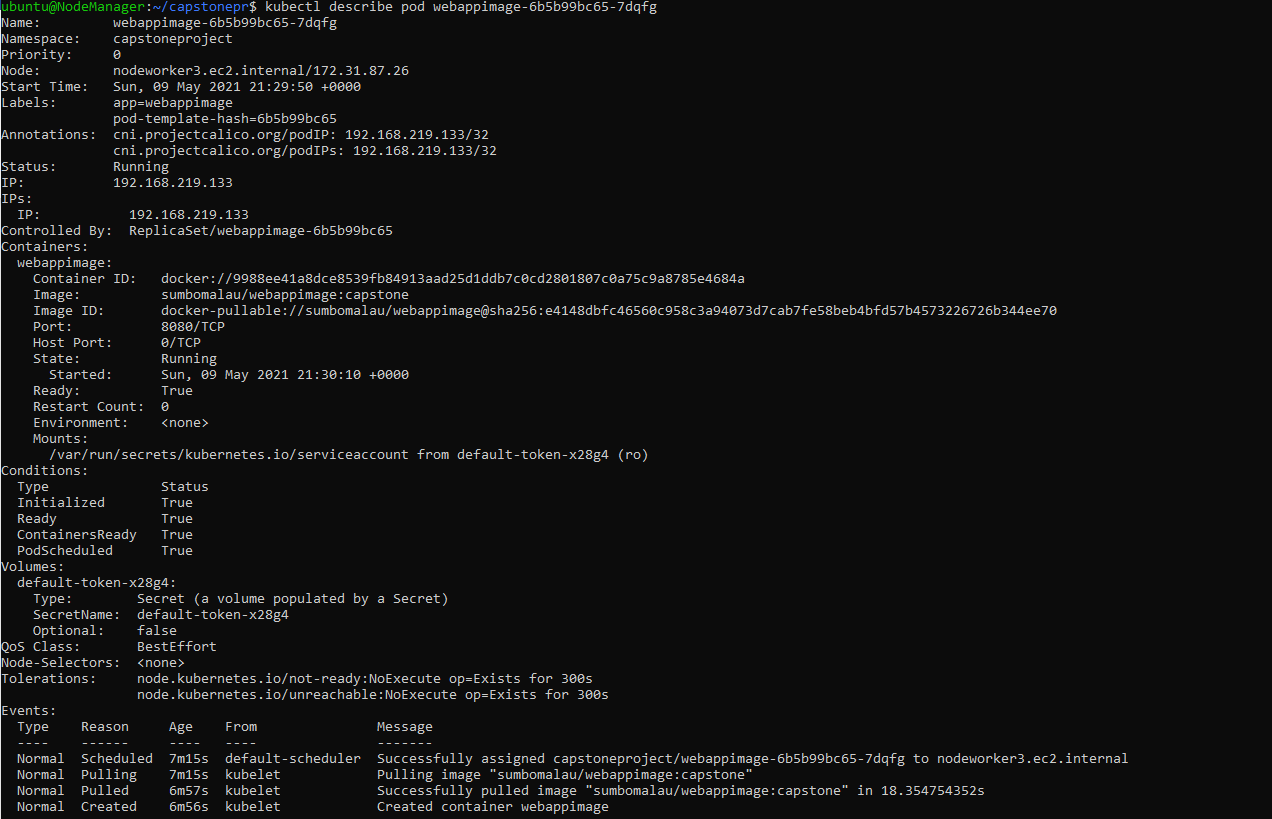
I created new namespace to organize all the resources that will be created on capstone project:

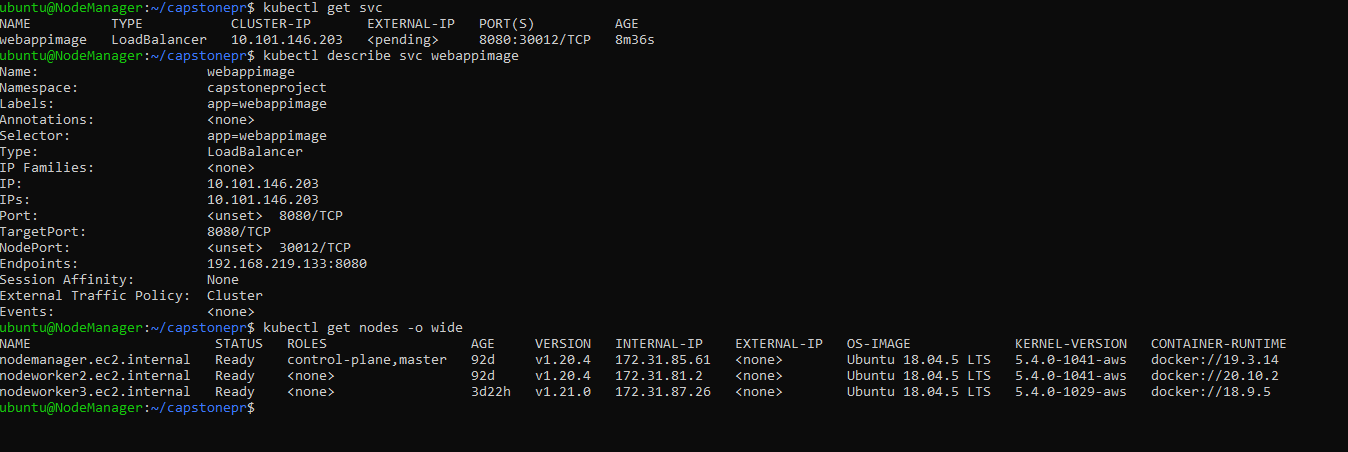


### 6.2 Create the Deployment YAML file to pull the Application image from Repository and configure the application in Pod

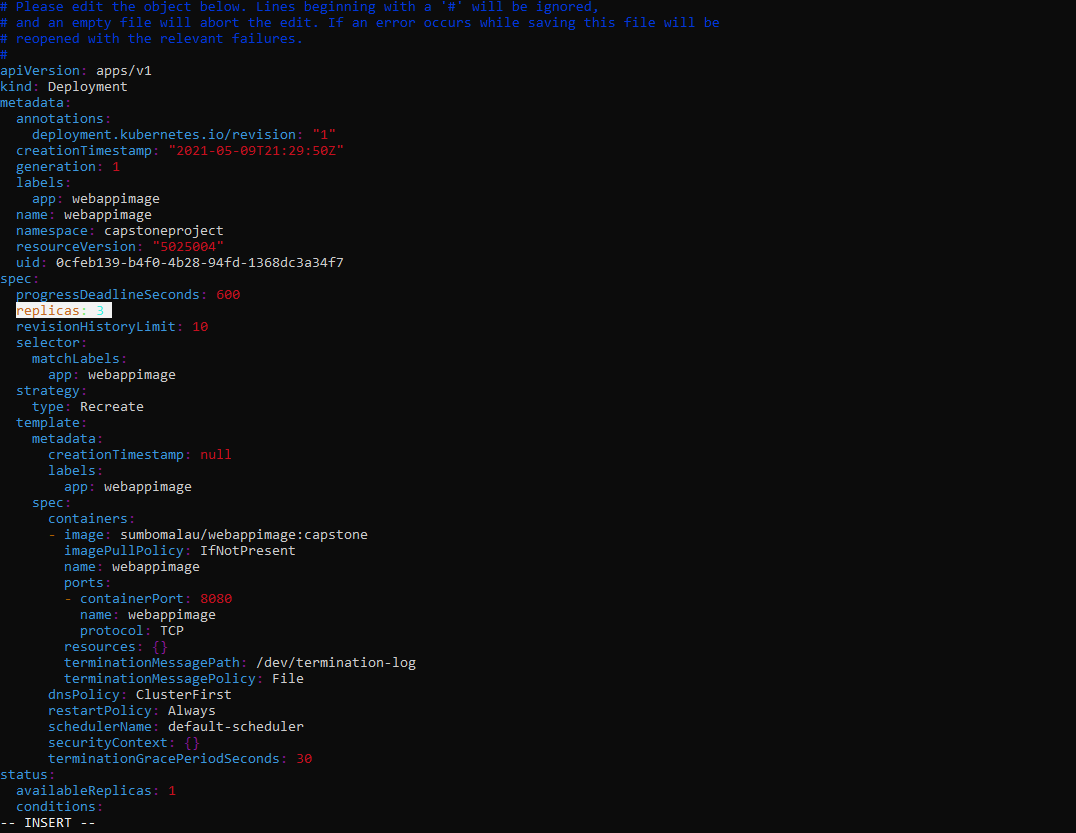


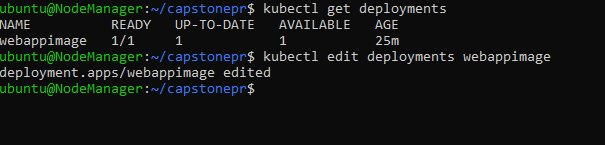


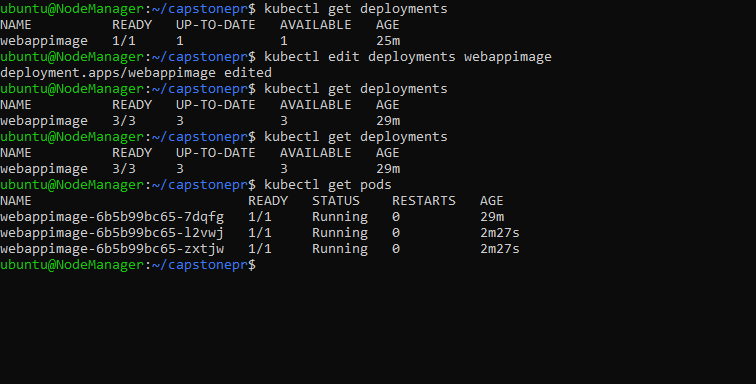




## 6.3 Configure the Web APP to use 3 replicas



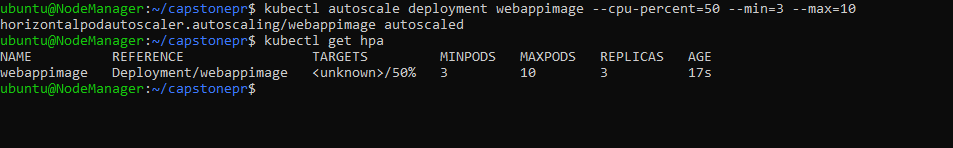


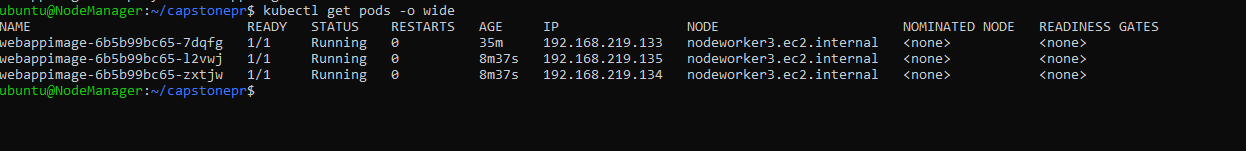


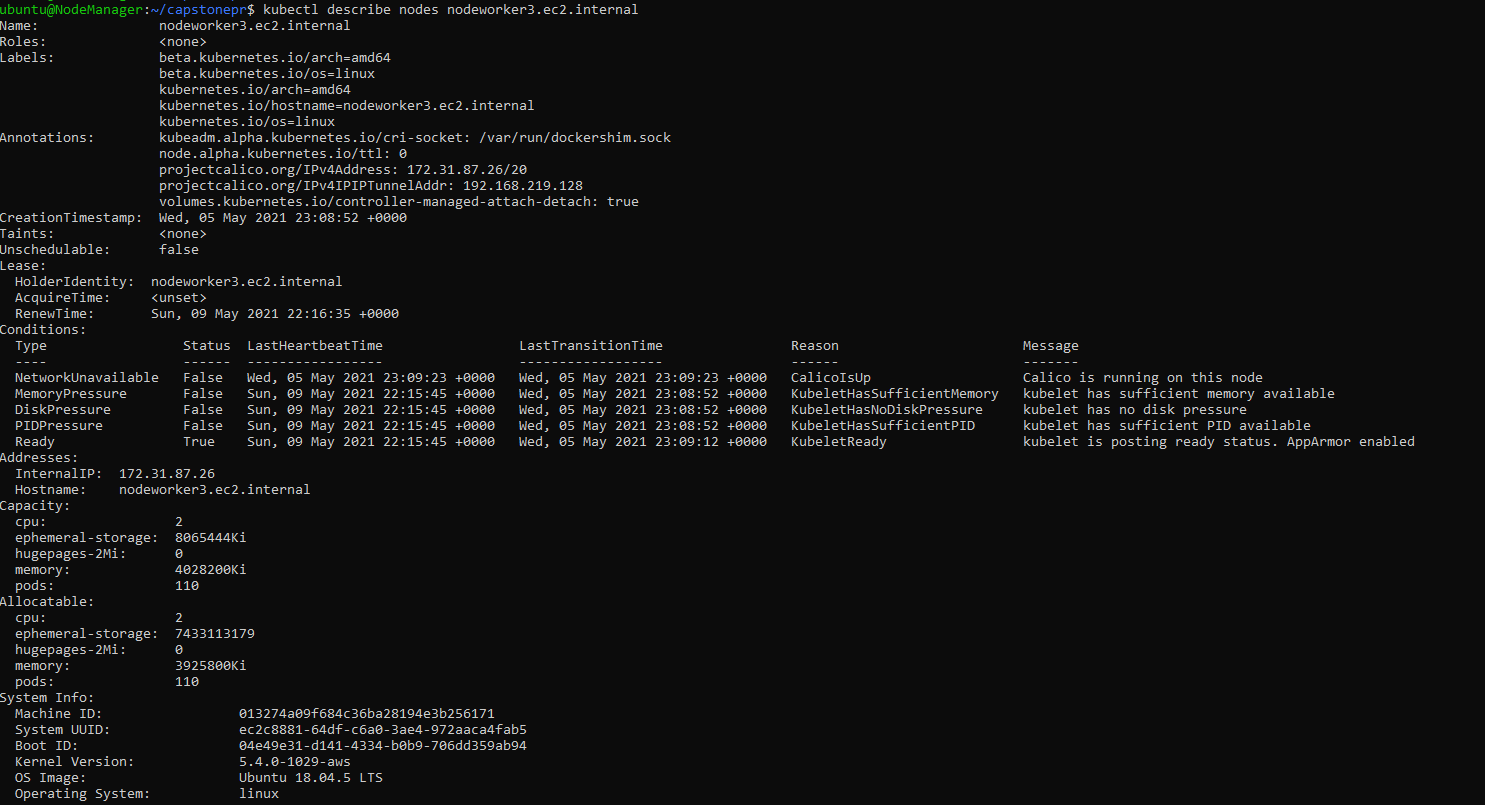
## 6.4 Configure LoadBalancer and High Availbility

I will define a horizontal pod Autoscalter that will mantain between 3 to 10 replicas, across all pods if CPU utlization is 50%.

kubectl autoscale deployment webappimage --cpu-percent=50 --min=3 --max=10





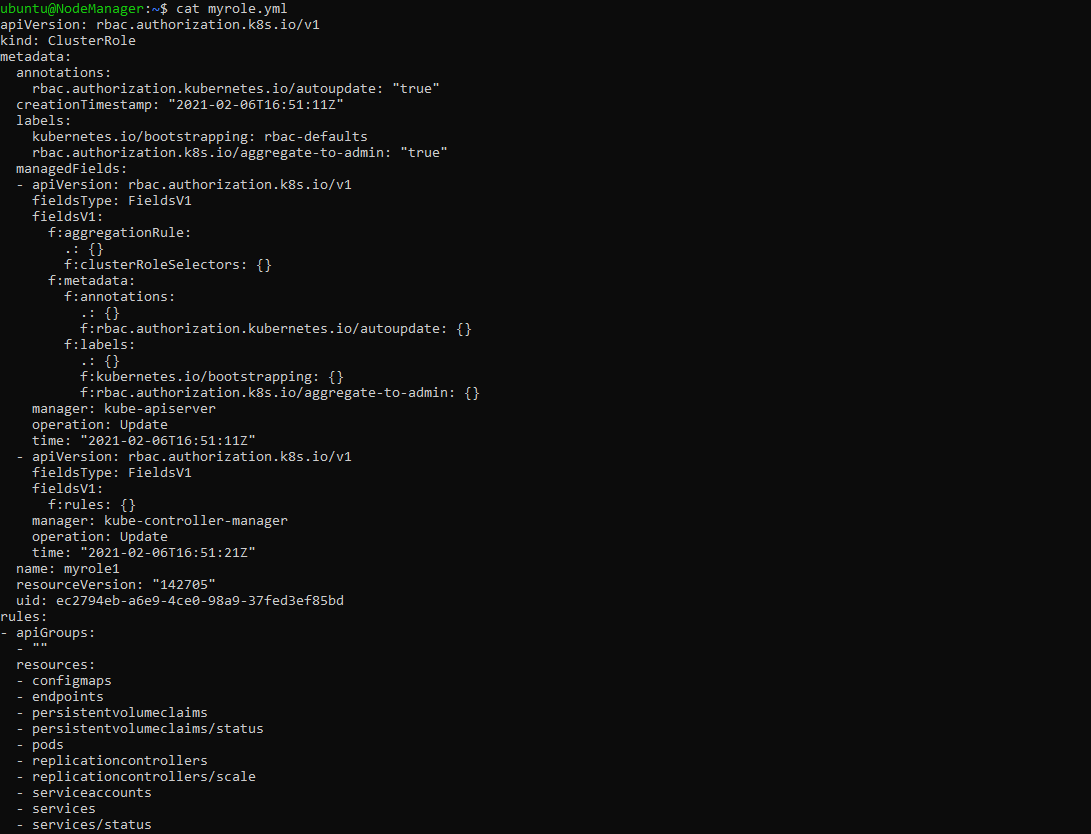


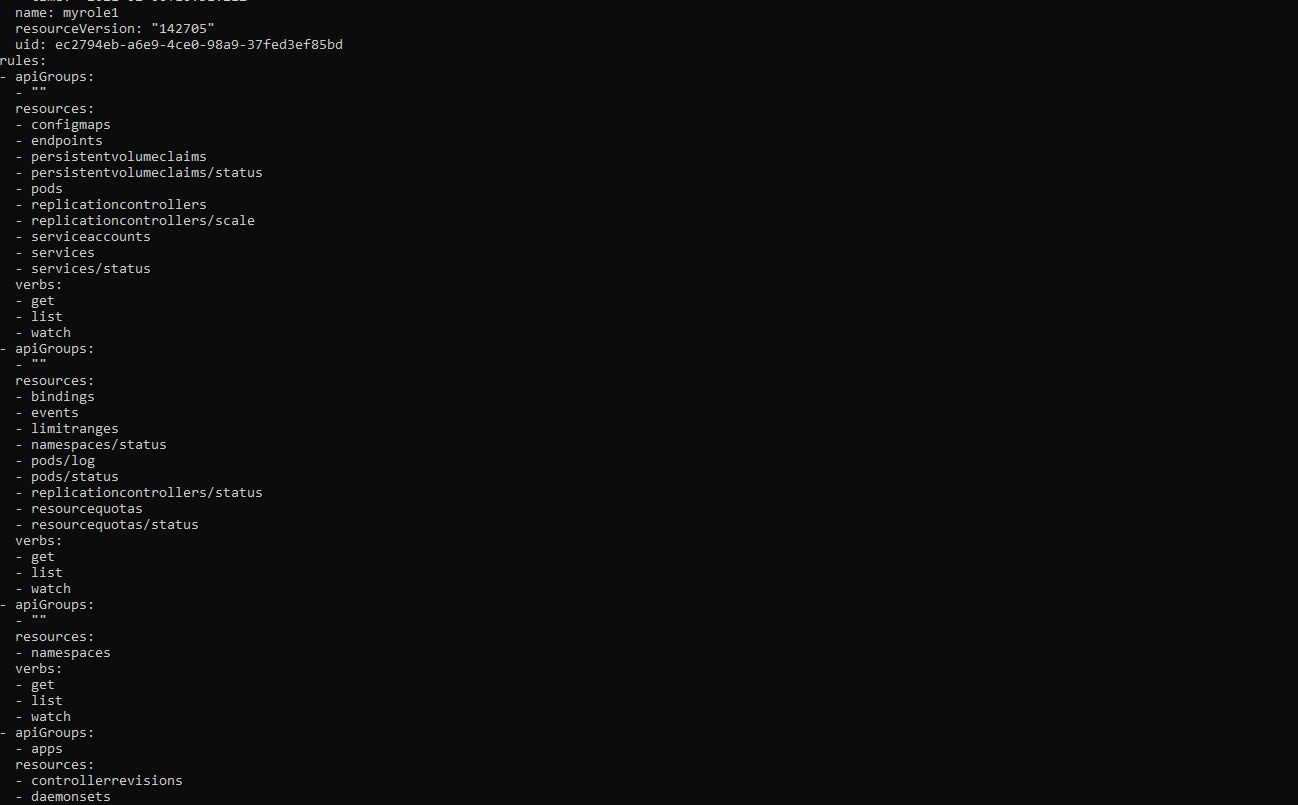


### 6.4.1 Testing the loadBalancer

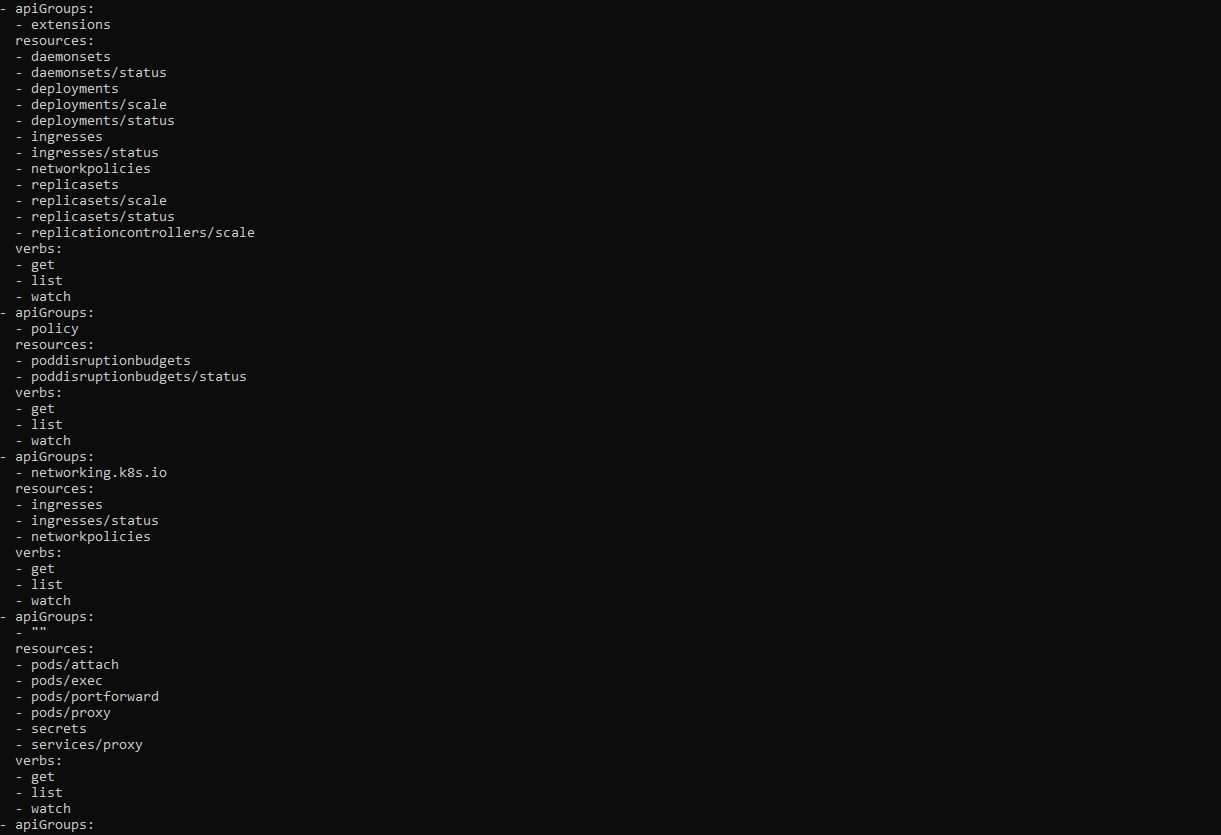
## 6.5 Create a new user with permissions to manage pods

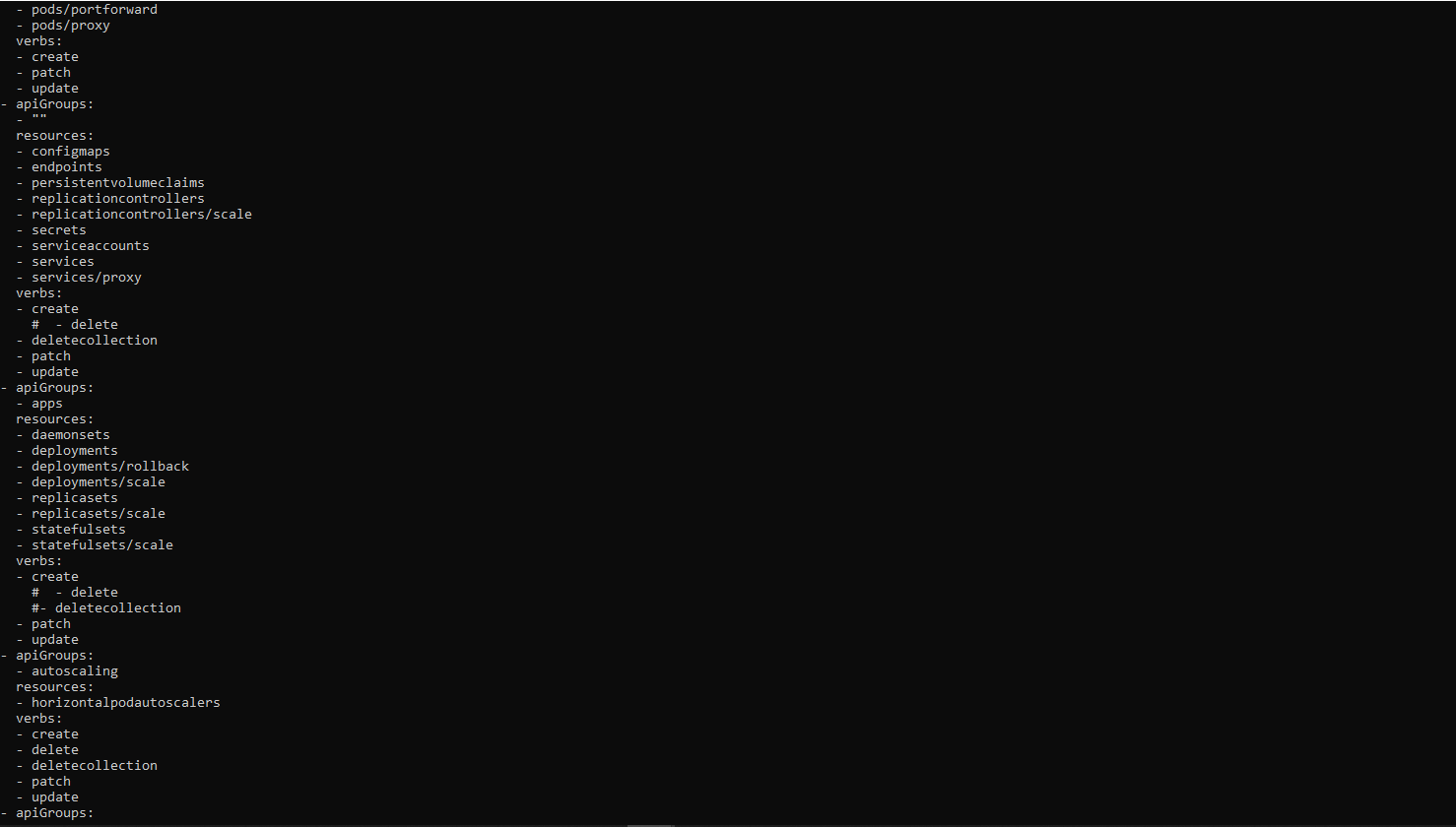
Will create a new YAML file with kind “ClusterRole” to create a new user to manage the pods and resources in our cluster:









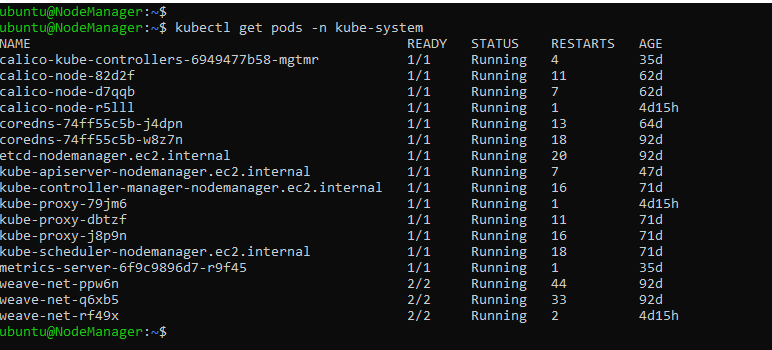




## 6.6 Take snapshot of ETCD database

We will use the built-in snapshot supported by etcd:

ETCDCTL\_API=3 etcdctl --endpoints $ENDPOINT snapshot save snapshotdb



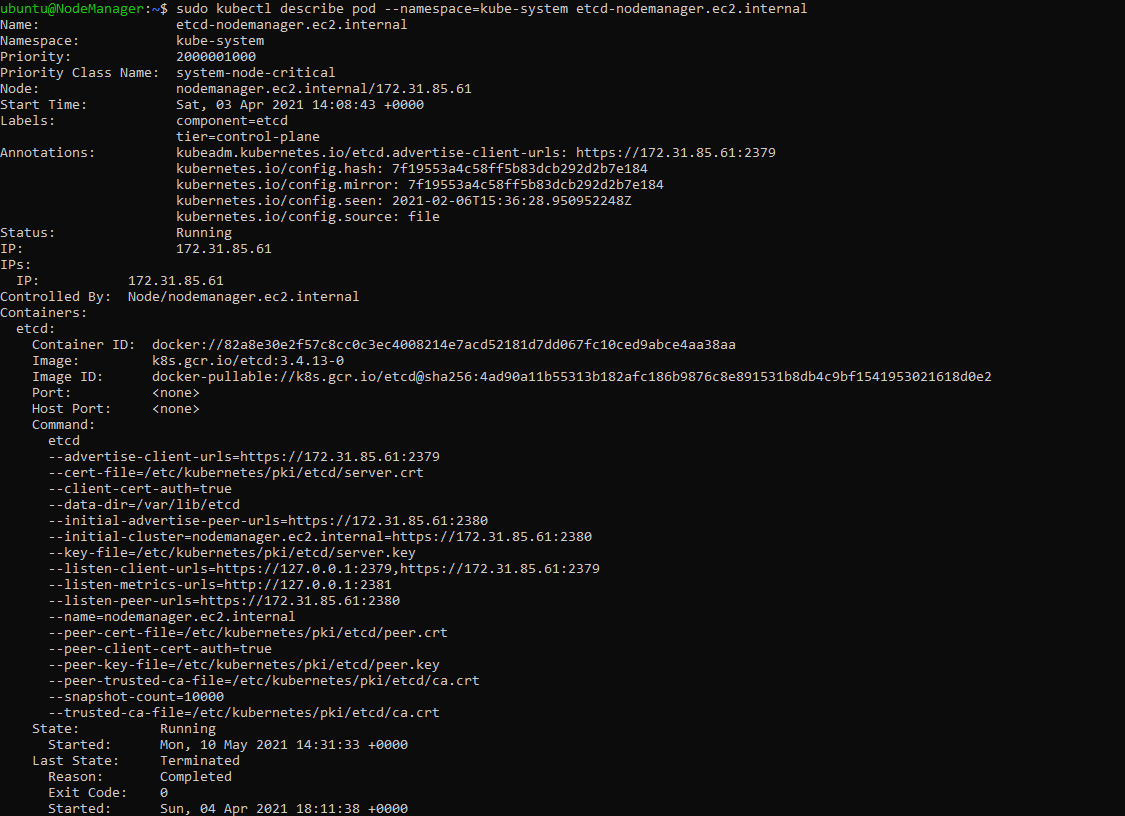
ETCDCTL\_API=3 etcdctl --endpoints=https://127.0.0.1:2379 **\**

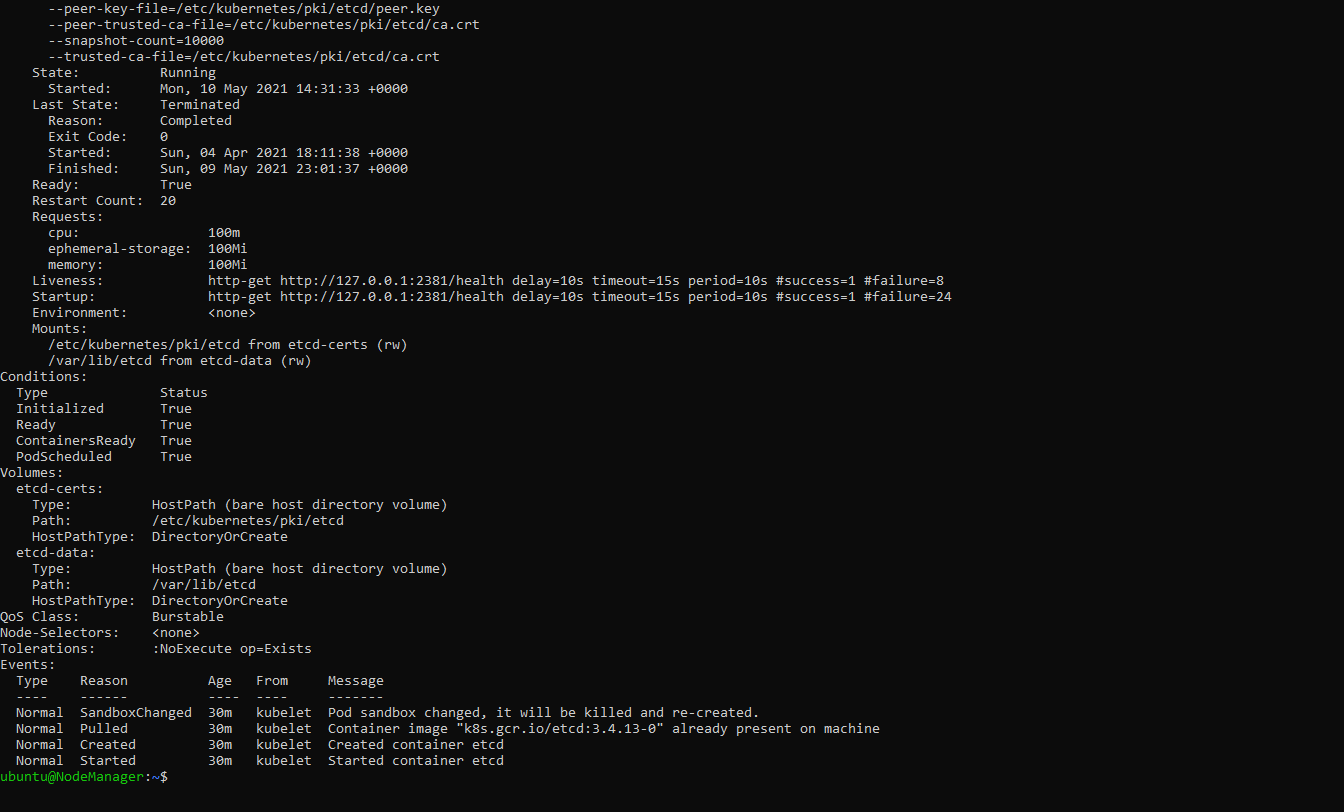
--cacert=<trusted-ca-file> --cert=<cert-file> --key=<key-file> **\**

snapshot save <backup-file-location>

We need to get from etcd pod the values:

* trusted-ca-file
* cert-file
* key-file





ETCDCTL\_API=3 etcdctl --endpoints=https://127.0.0.1:2379 **\**

--cacert= /etc/kubernetes/pki/etcd/ca.crt **\**

--cert=/etc/kubernetes/pki/etcd/server.crt **\**

--key= /etc/kubernetes/pki/etcd/server.key **\**

snapshot save /home/ubuntu/capstonepr/snapshotbk

