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:**

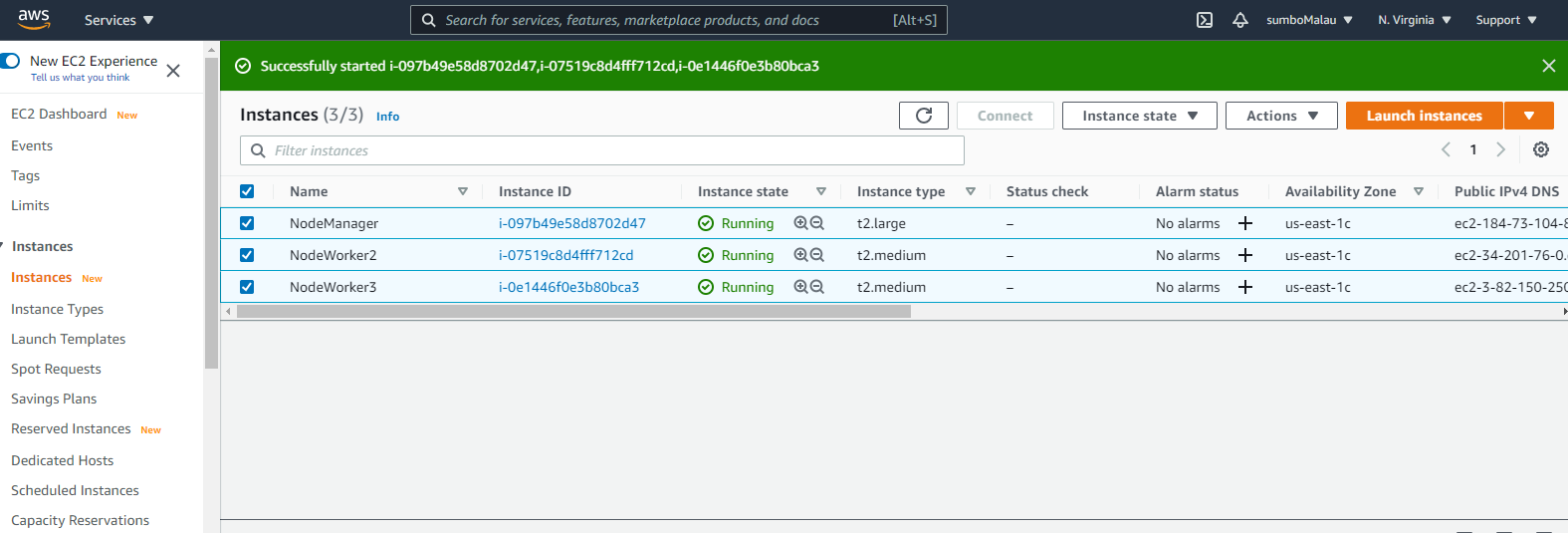
1. Create the cluster (EC2 instances with load balancer and elastic IP in case of AWS)
2. Automate the provisioning of an EC2 instance using Ansible or Chef Puppet
3. Install Docker and Kubernetes on the cluster
4. Implement the network policies at the database pod to allow ingress traffic from the front-end 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

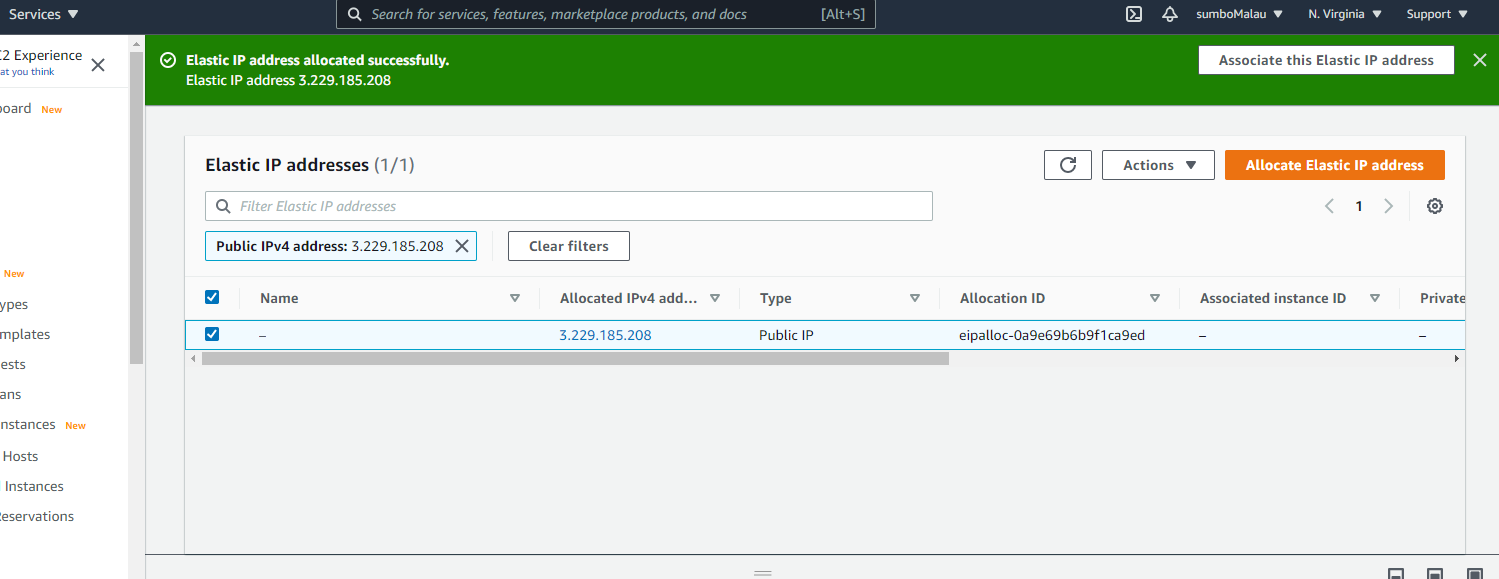
**The following tools must be used:**

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

# 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.

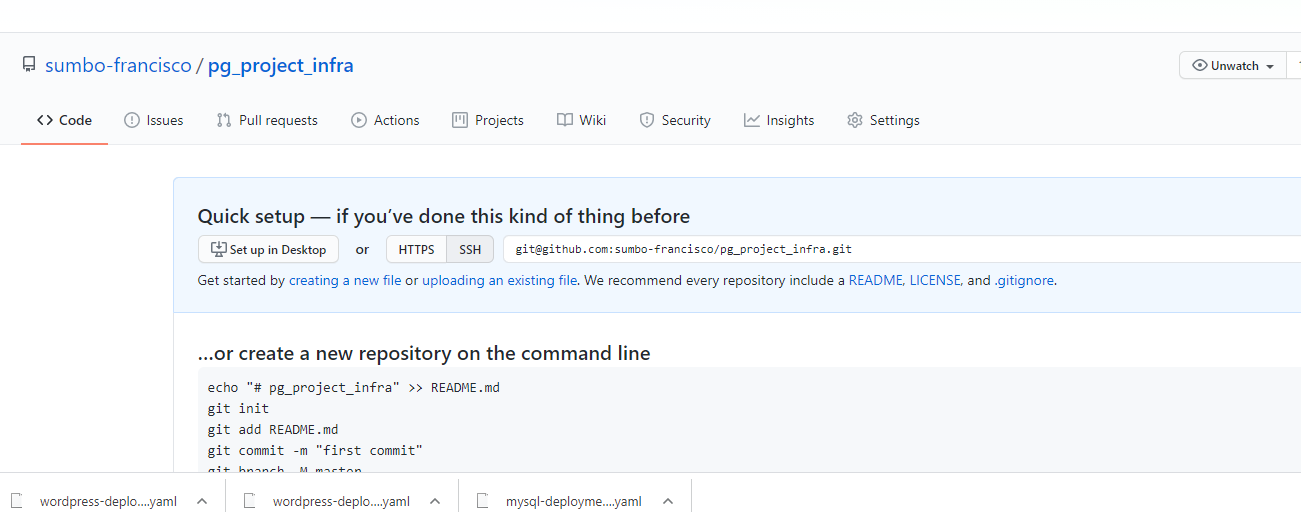


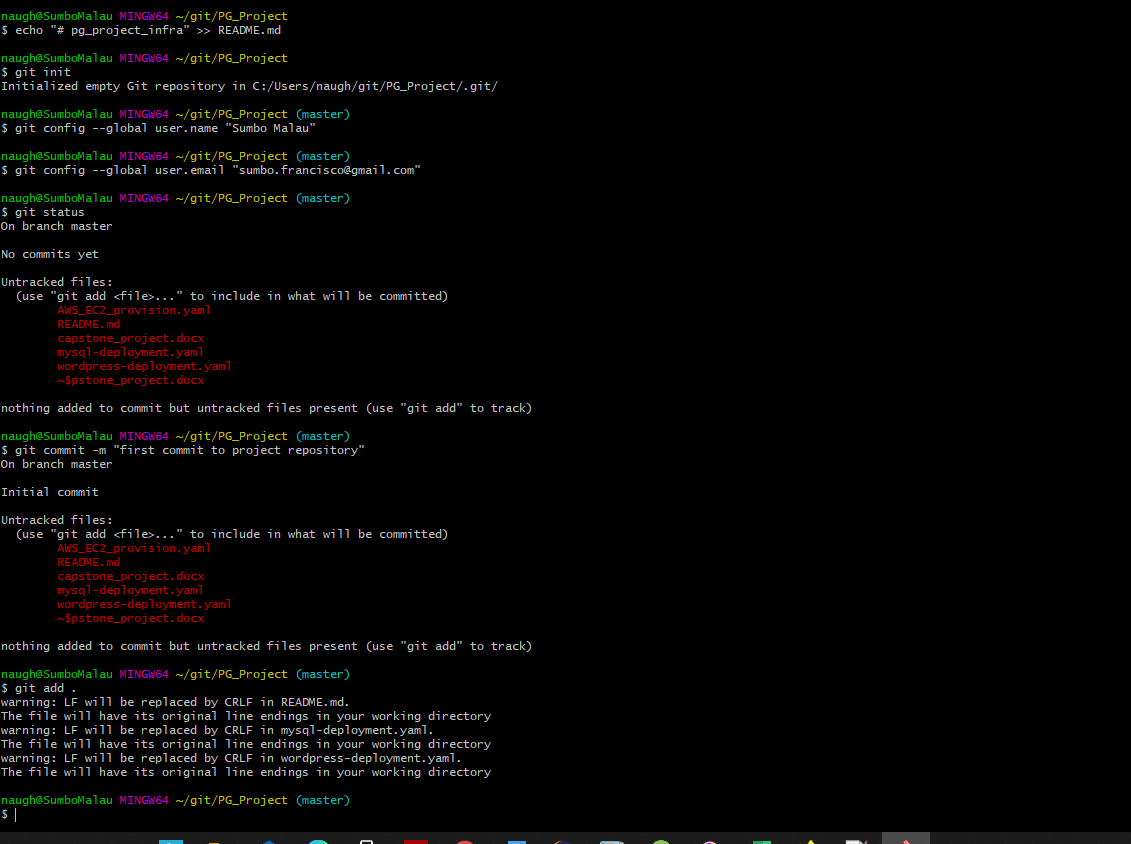


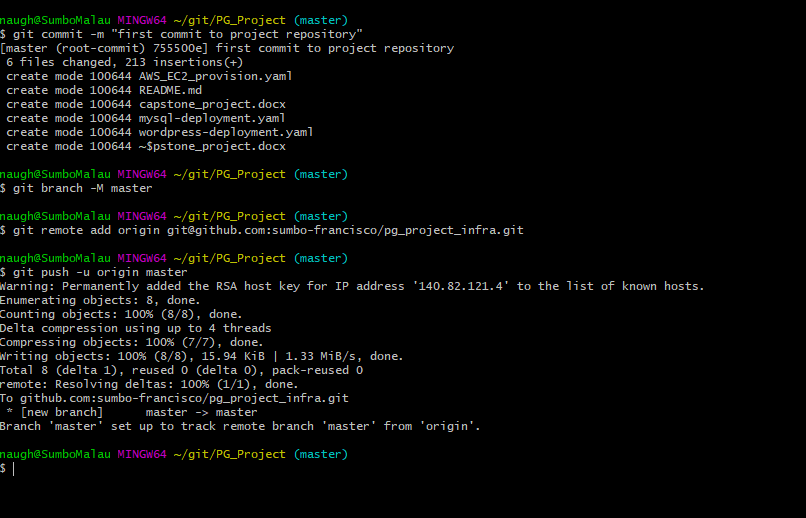
We will deploy a WorldPress web application with MySql database.

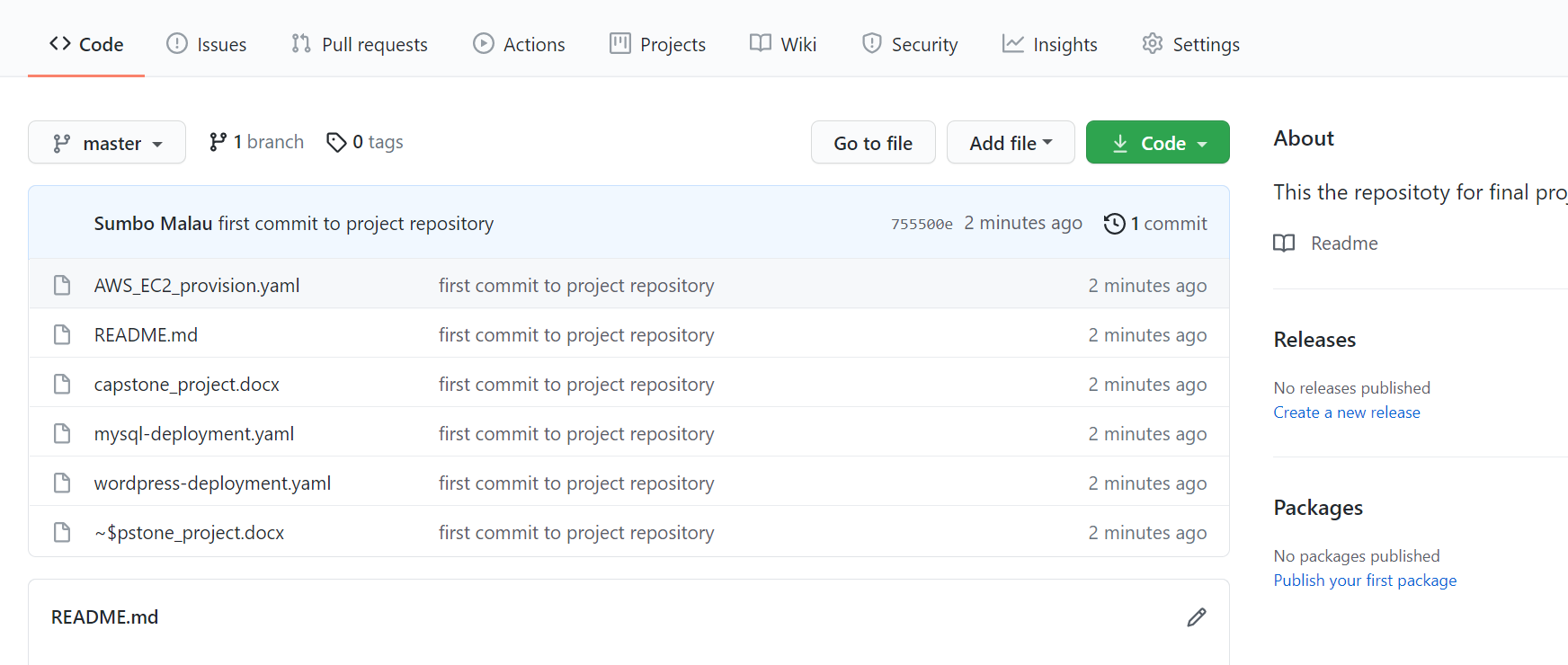
# 2 Project Repository

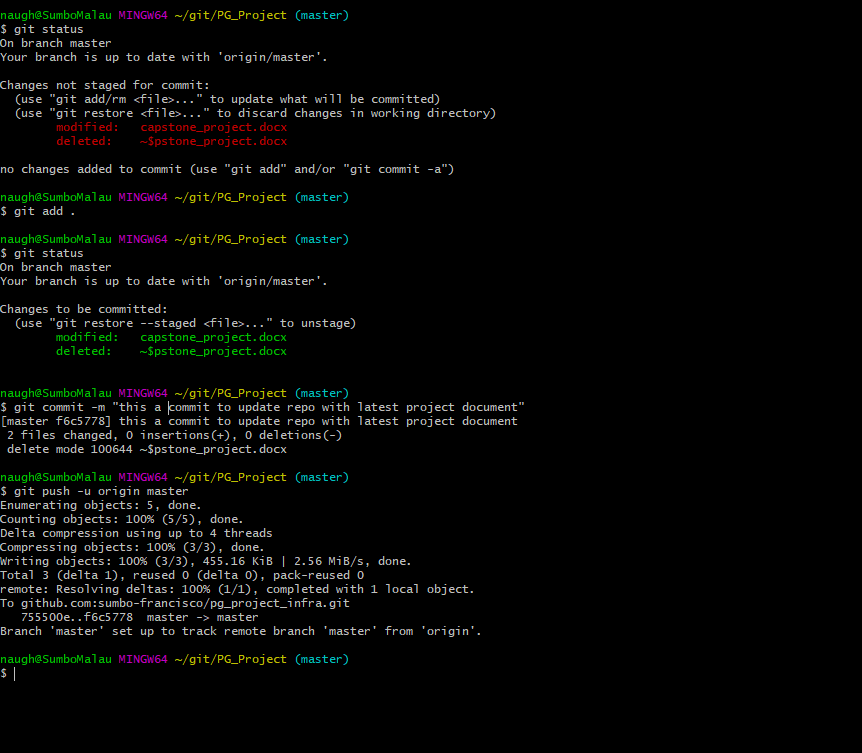
Create a new repo in GitHub:



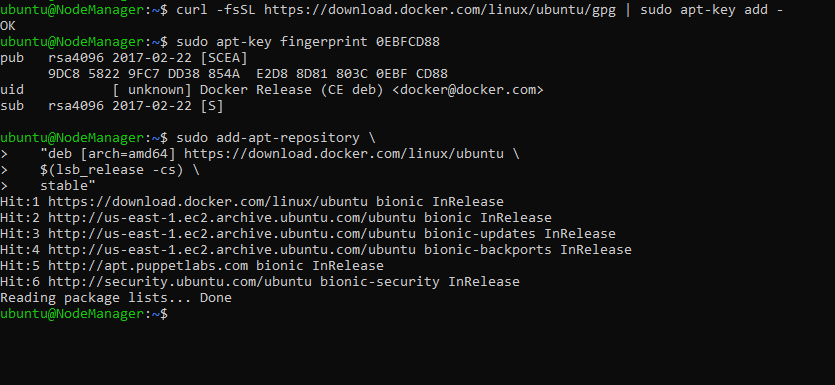


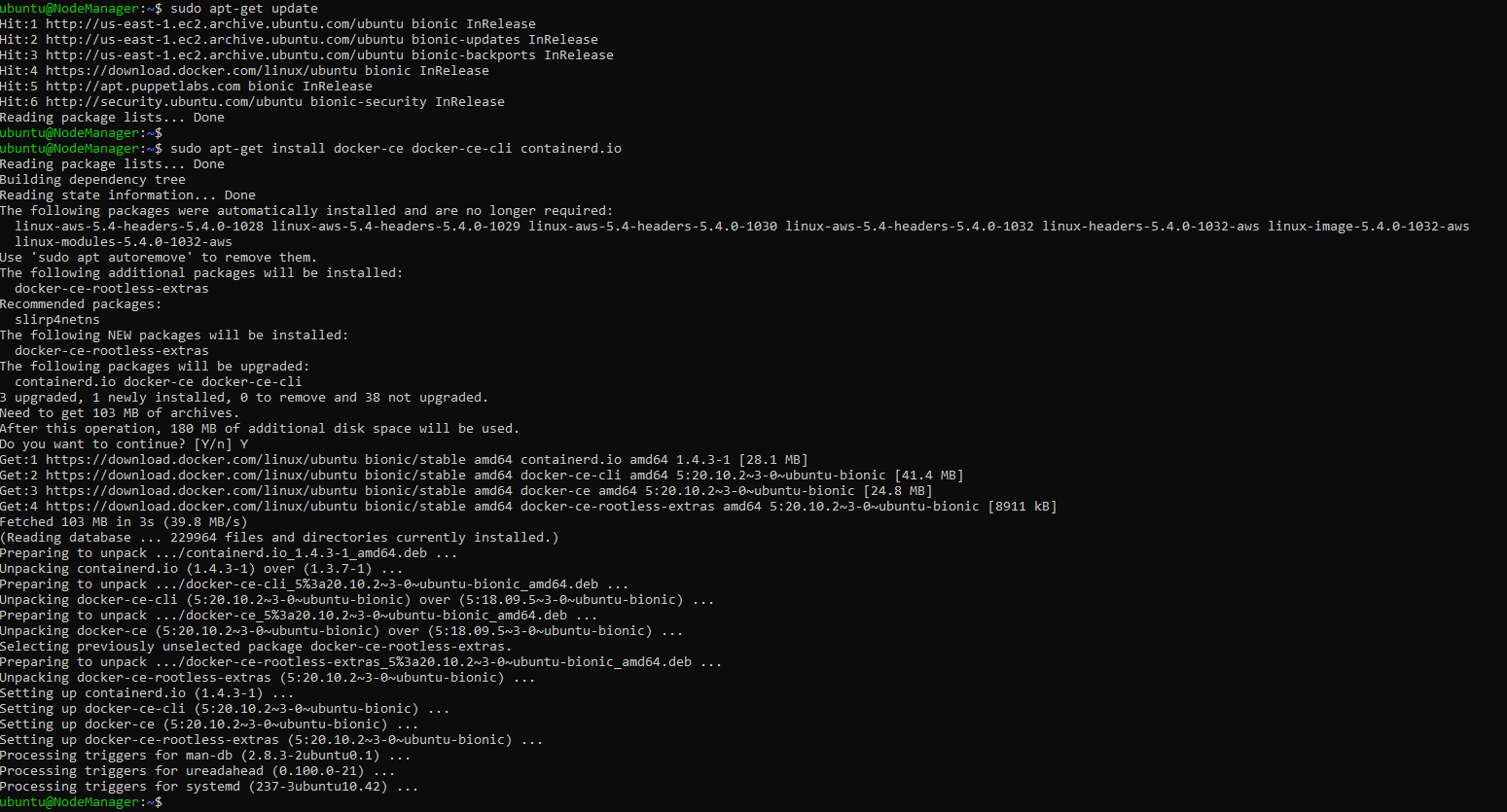




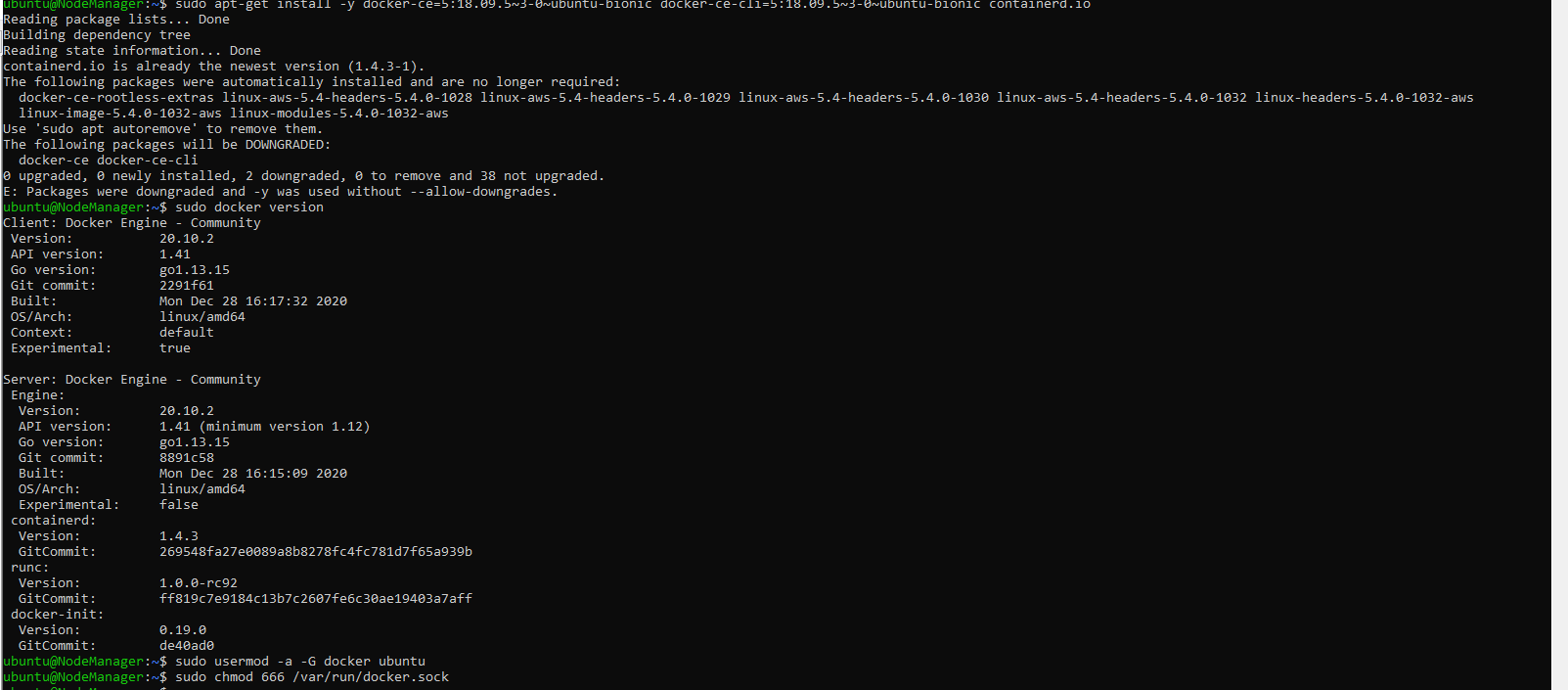


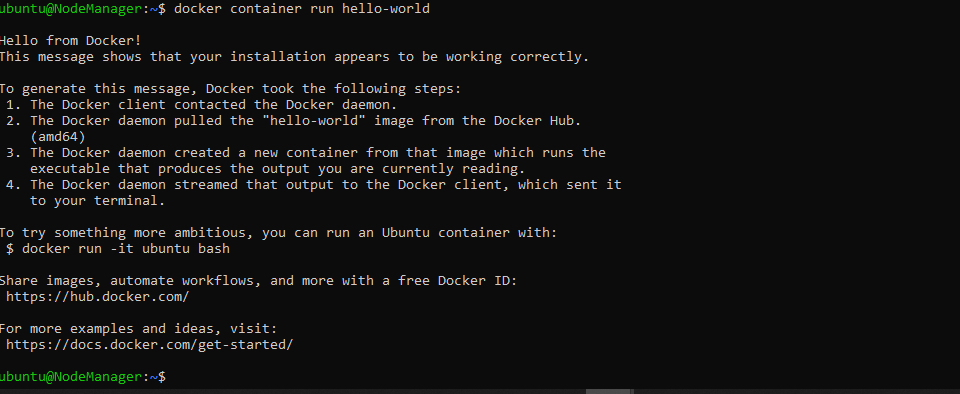
# 3 Docker Server

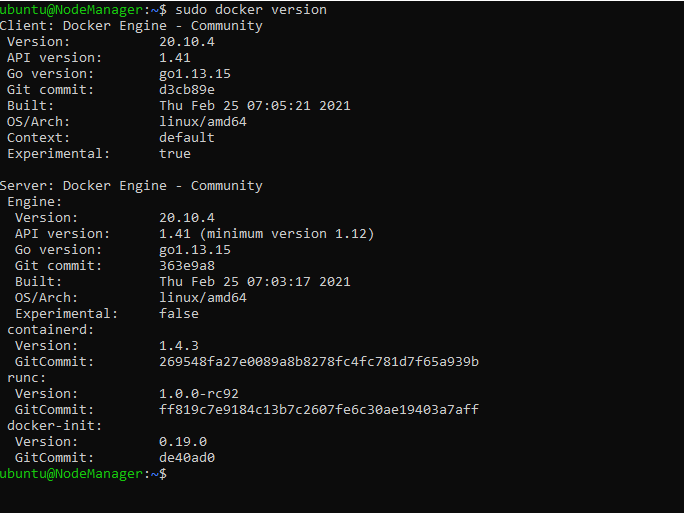




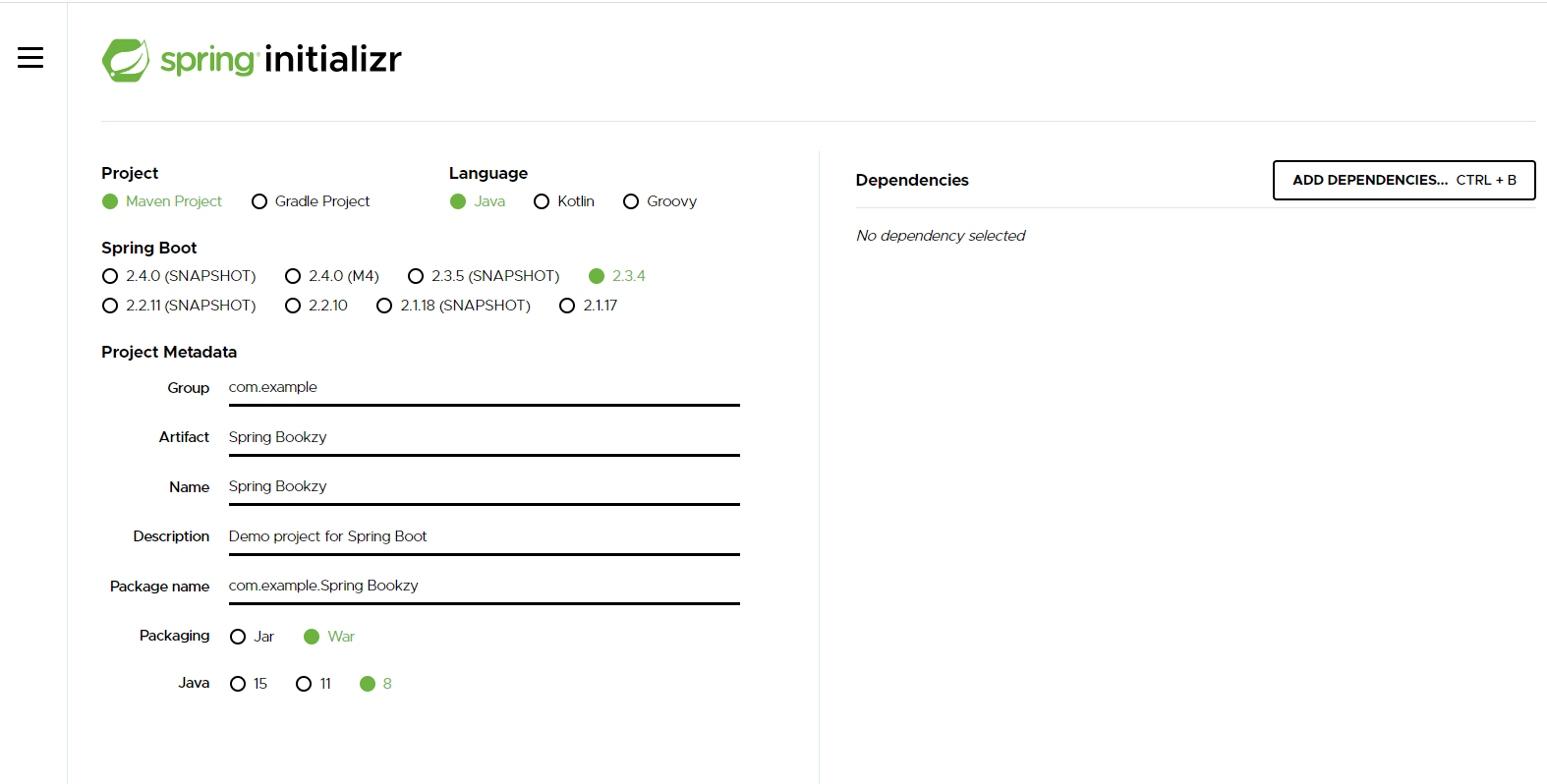
$ sudo apt-get install docker-ce=<VERSION\_STRING> docker-ce-cli=<VERSION\_STRING> containerd.io

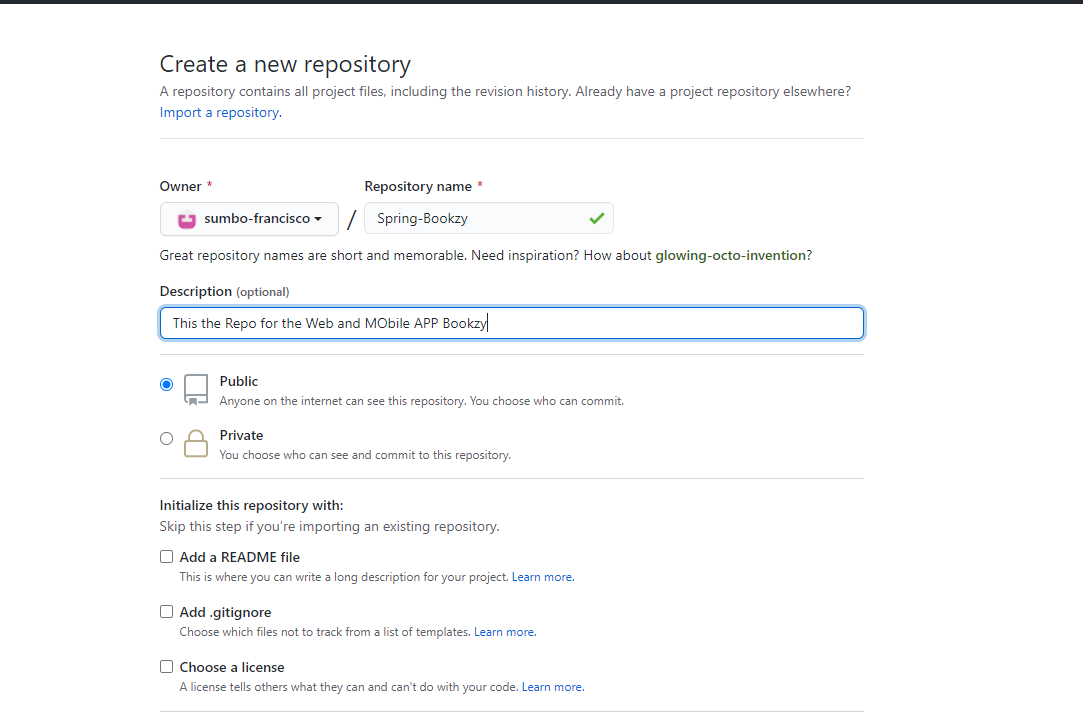


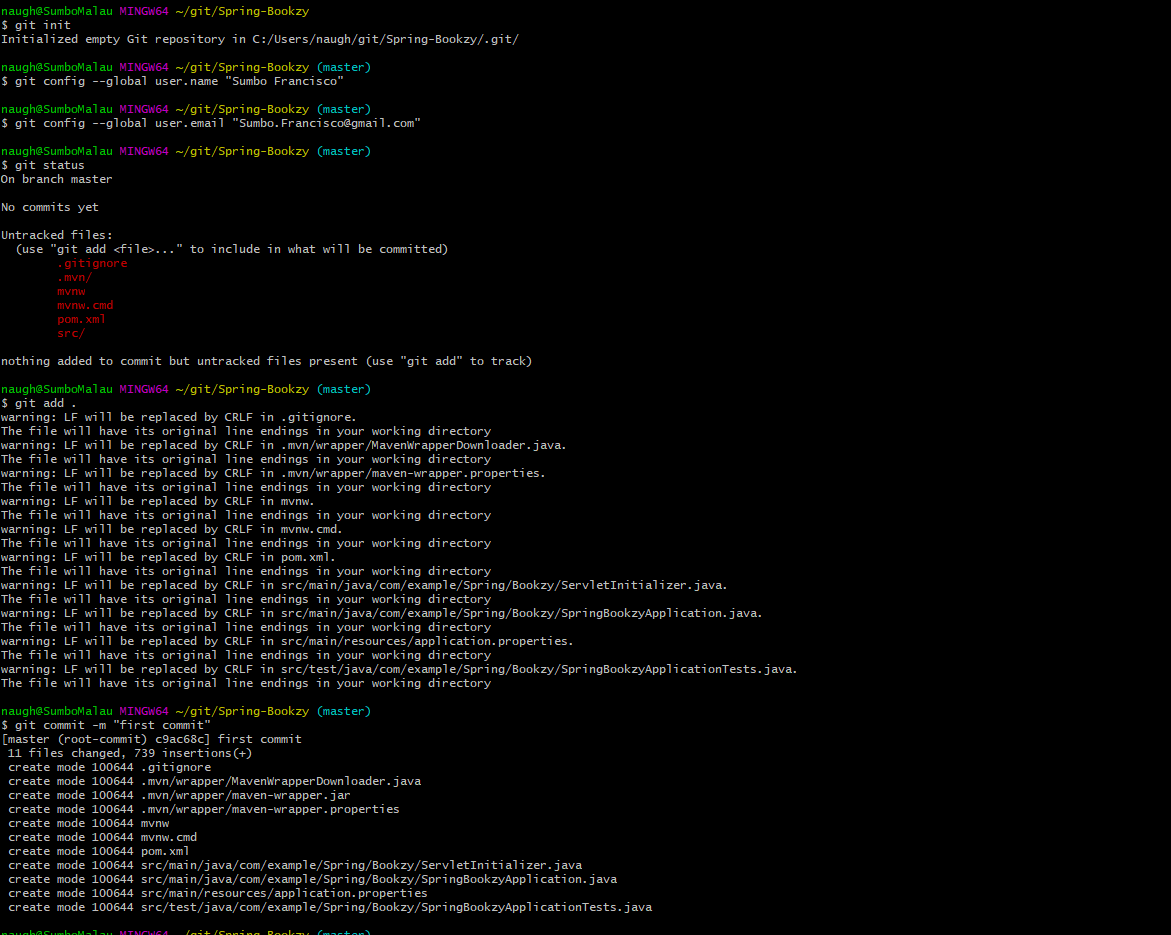


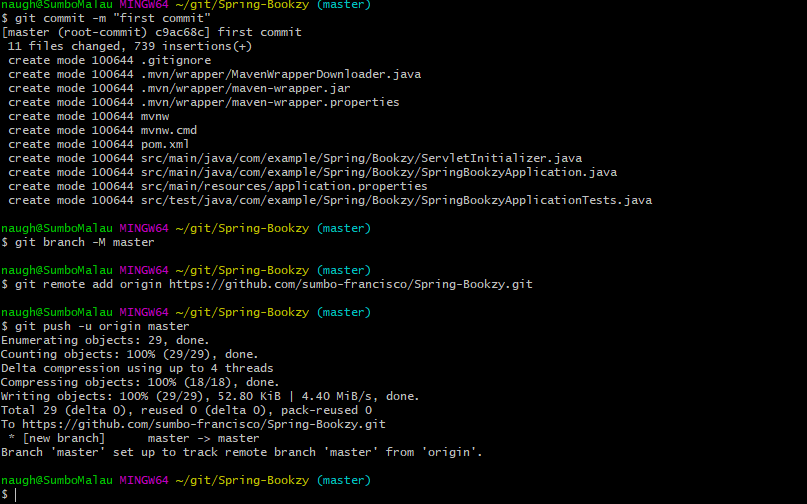


# 4 Build WebAPP (Spring boot Maven project)



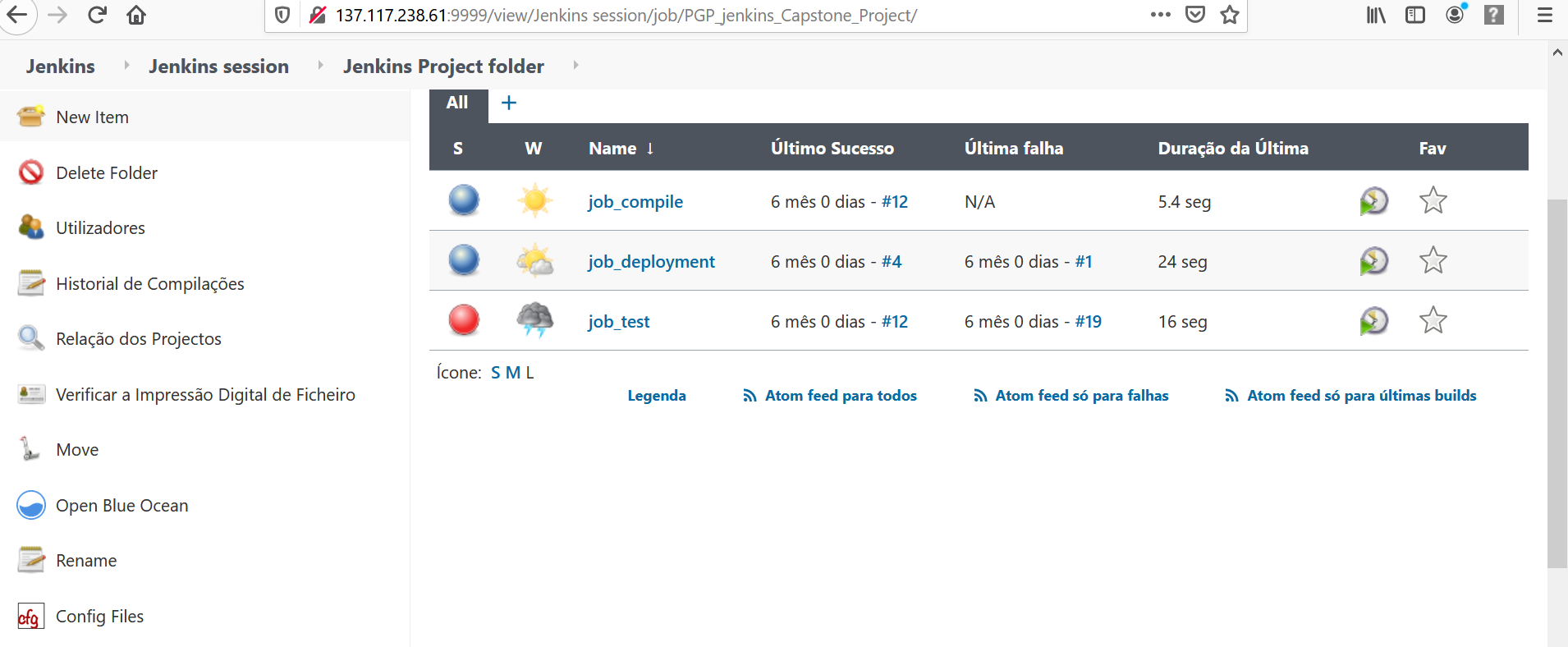


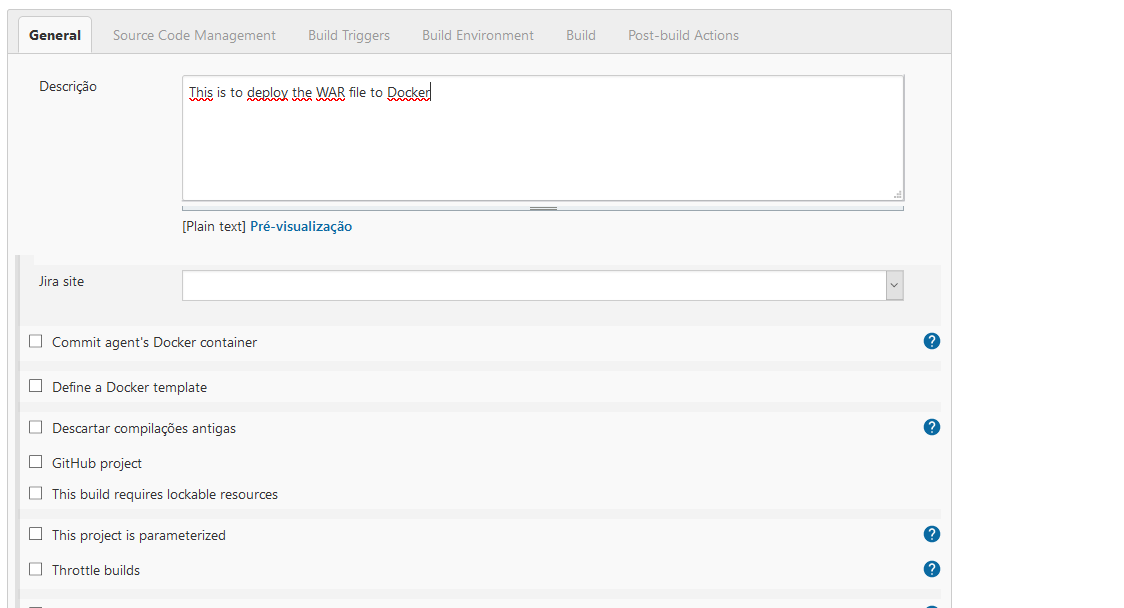


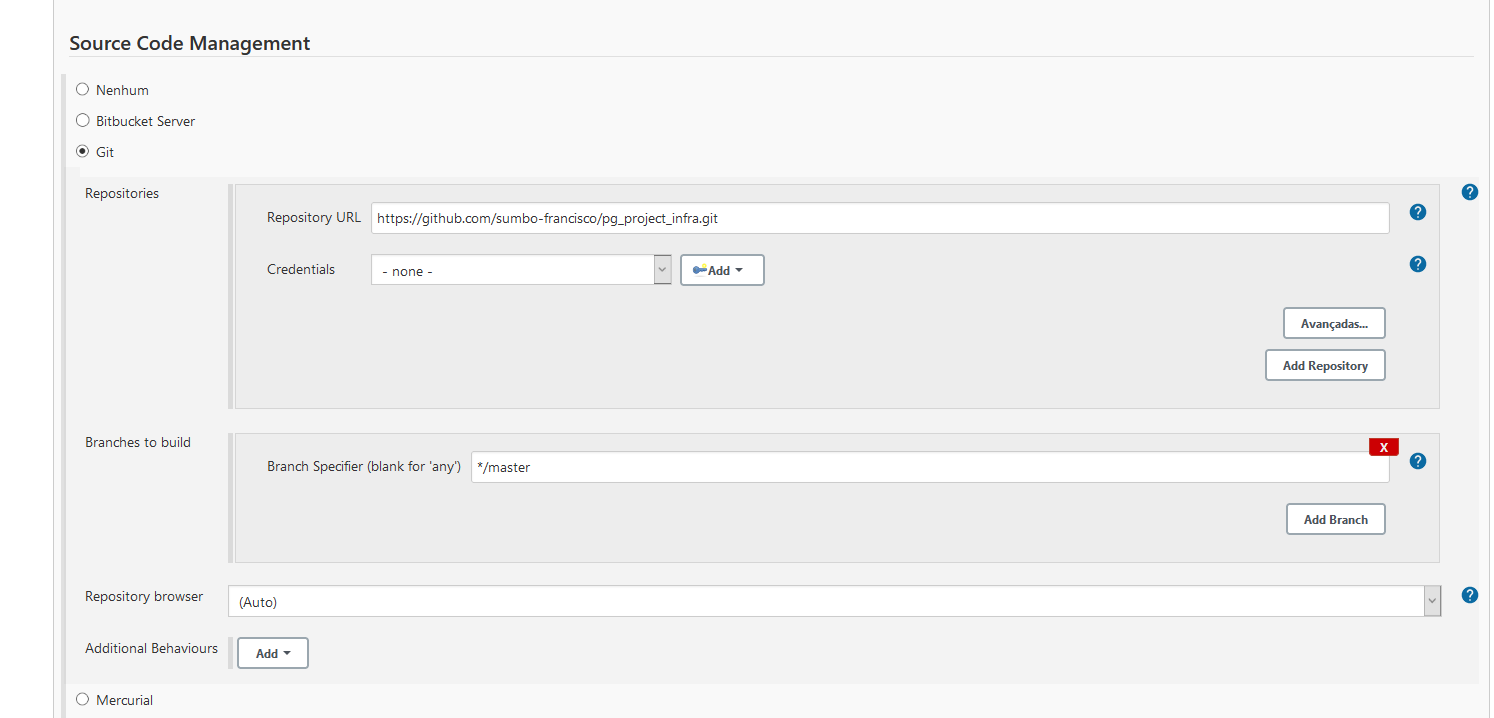


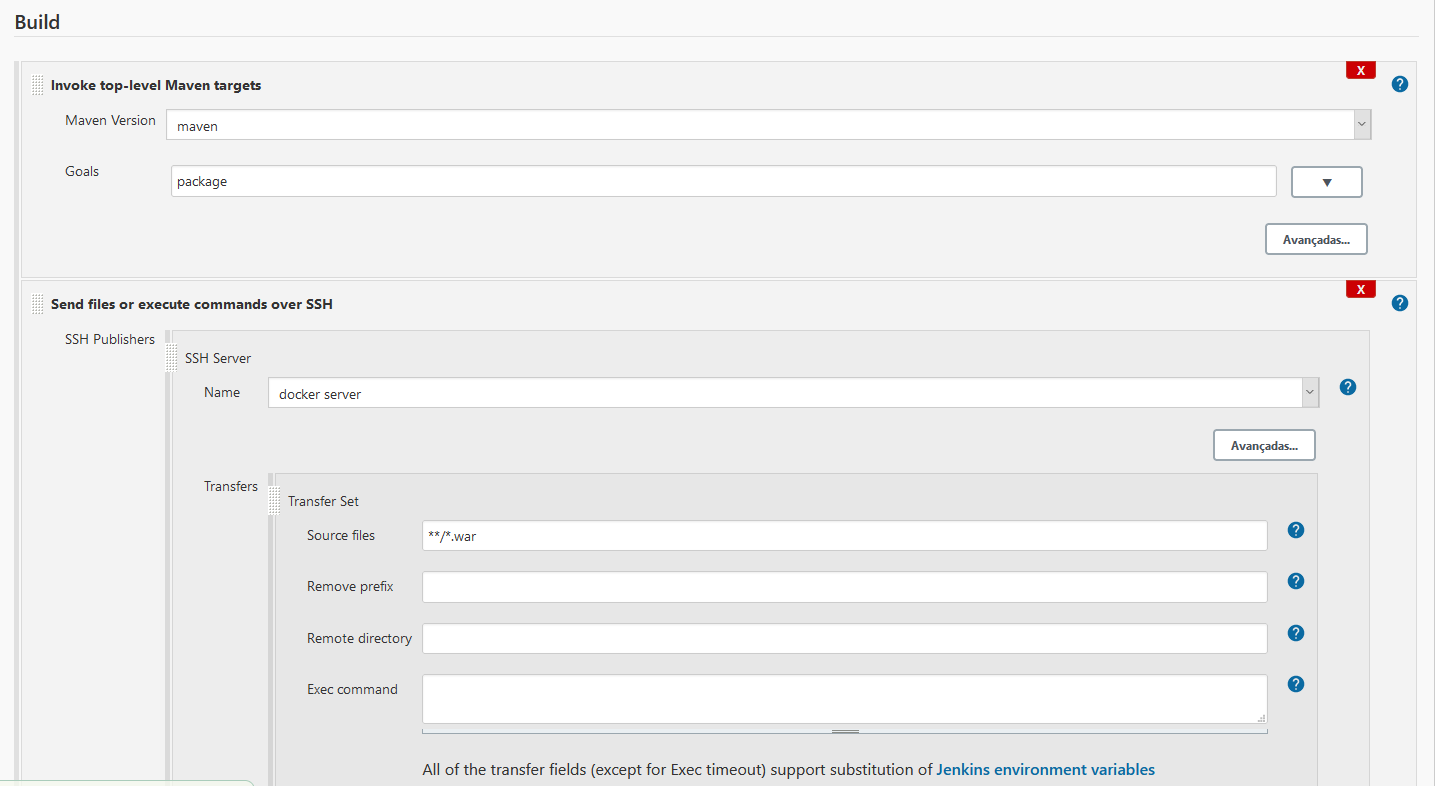
## 4.1 Build and Deploy application in a Tomcat container in Jenkins

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





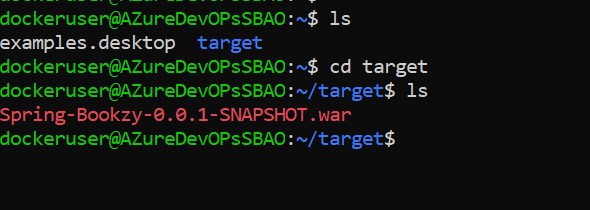


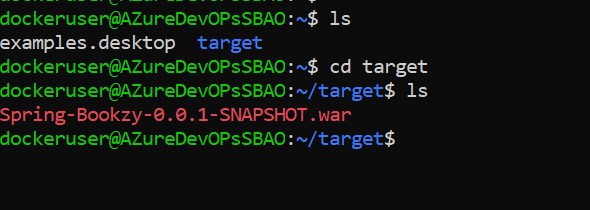




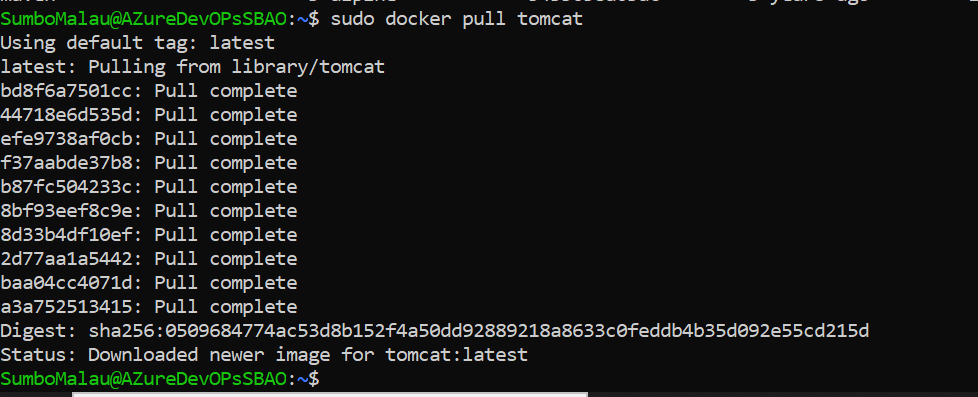


The have successfully built an Web application from the Jenkins server and moved to the Docker server:

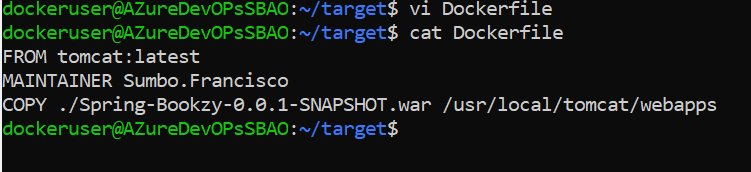


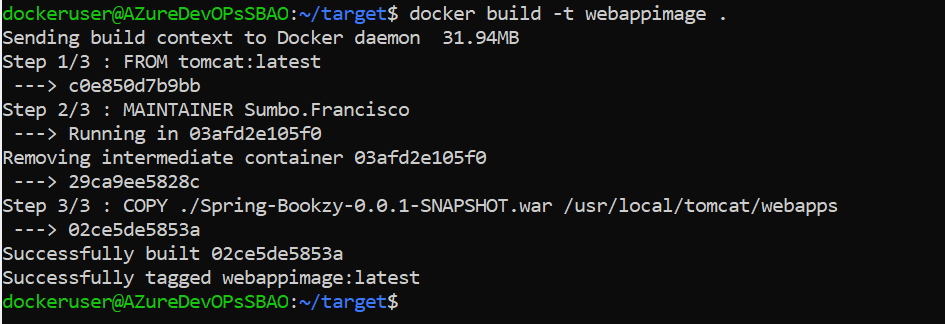


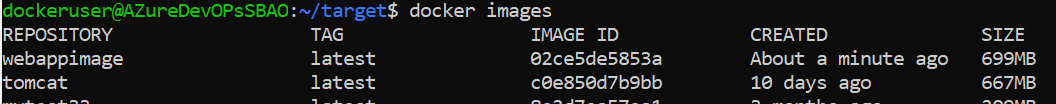
## 4.2 Create the Docker Image (Dockerfile) and Publish on Repository



We will use a Dockerfile to create our image:





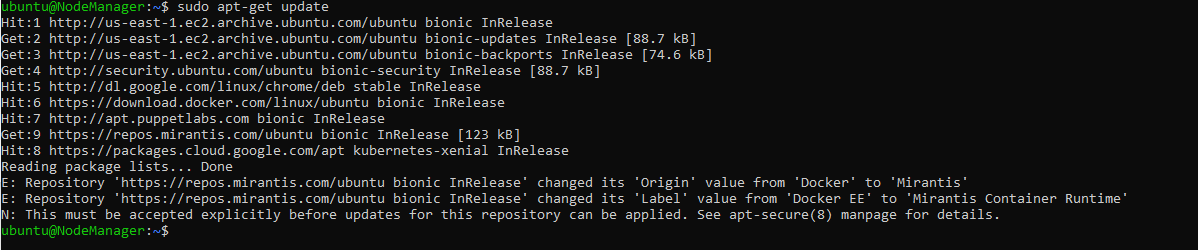


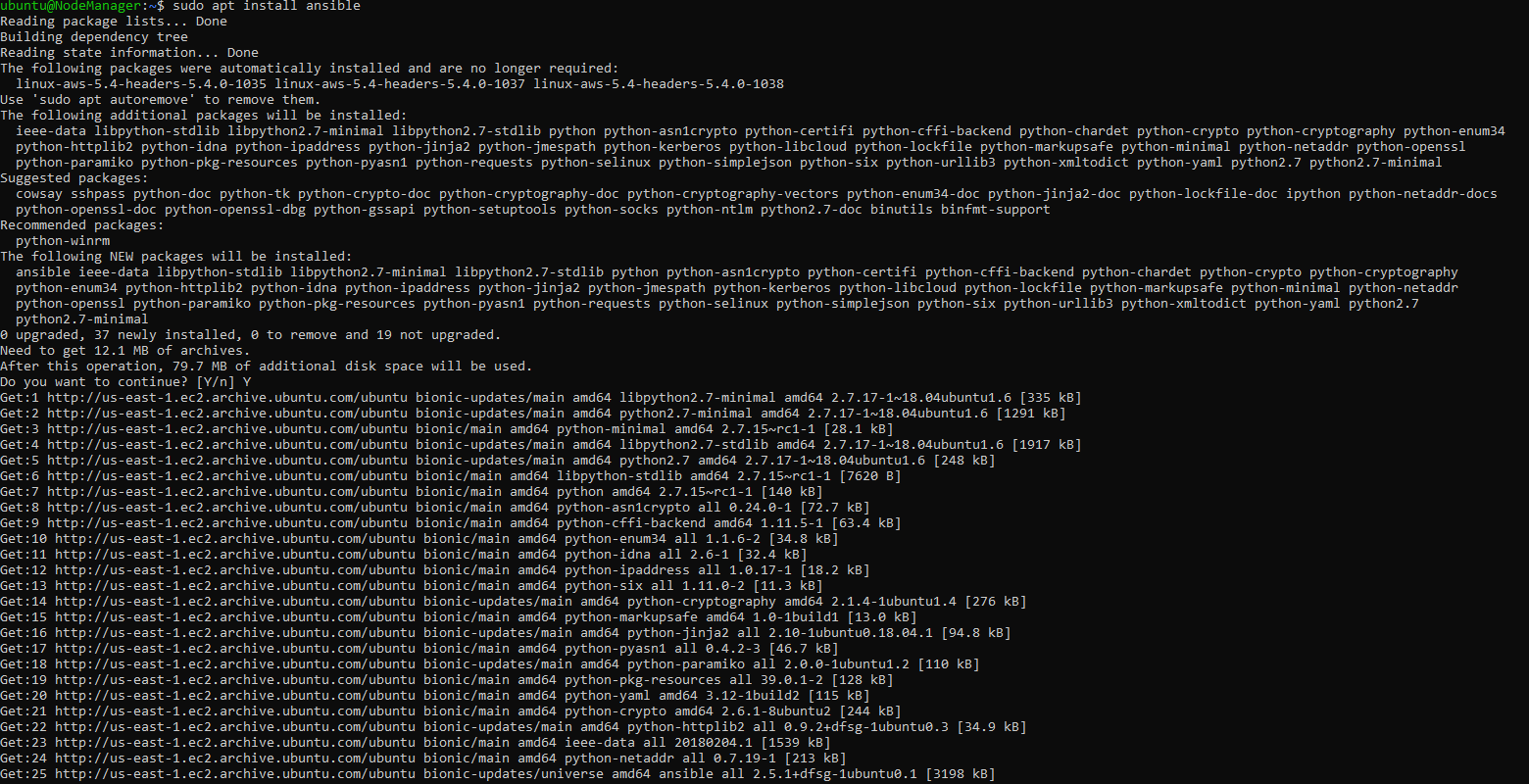
We can test our image application run in a container:

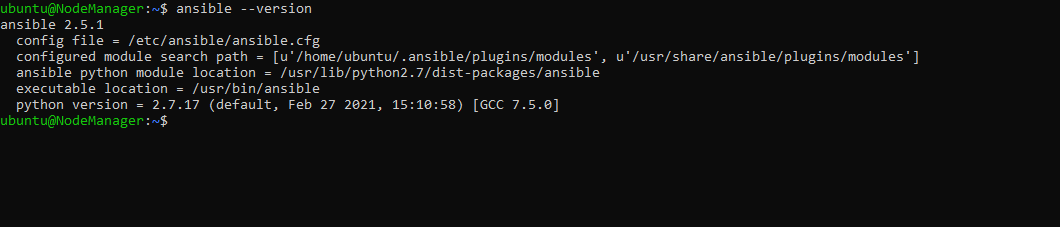
docker run -d --name tomcat-app -p 8080:8082 webappimage

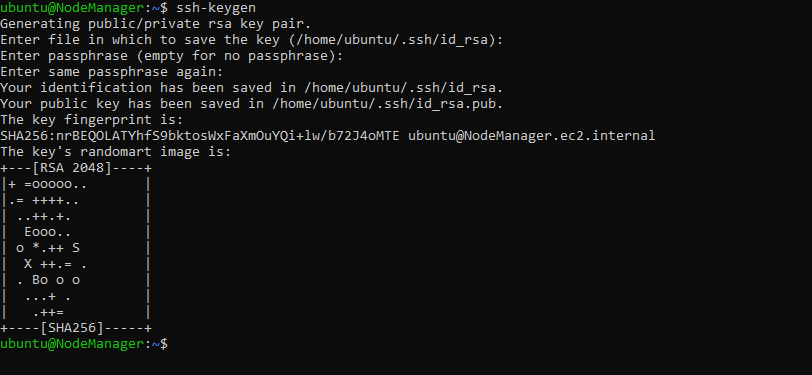
# 5 Automation of AWS EC2 instance provisioning in ANSIBLE

## 5.1 Ansible Installation:

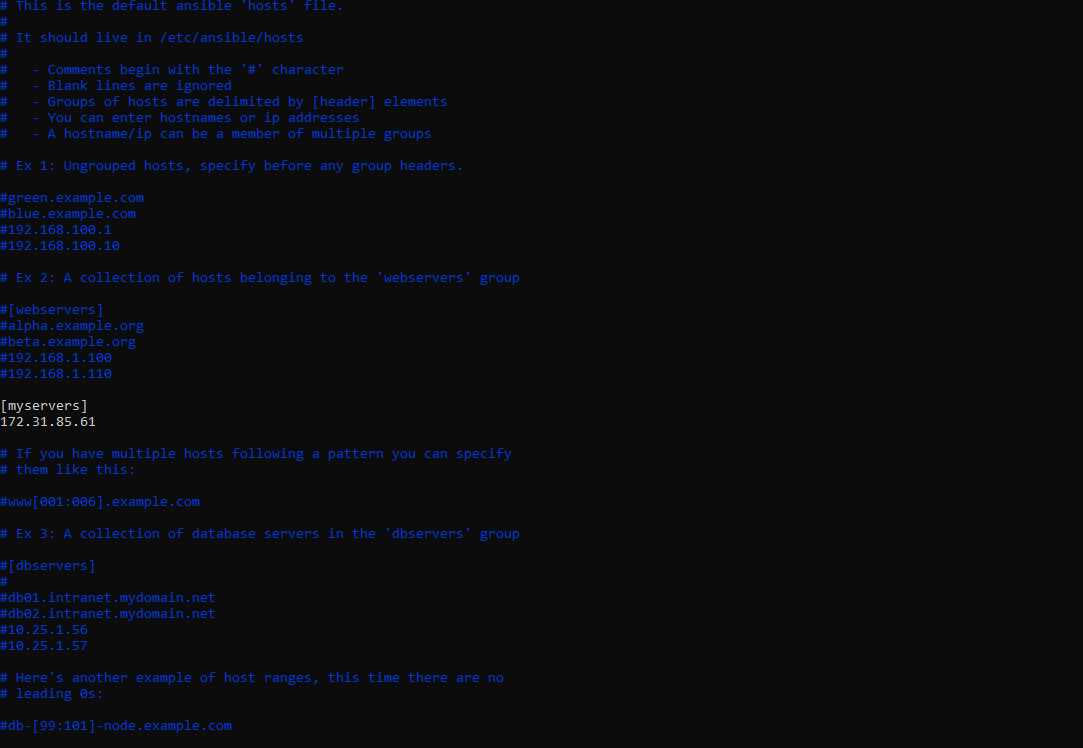




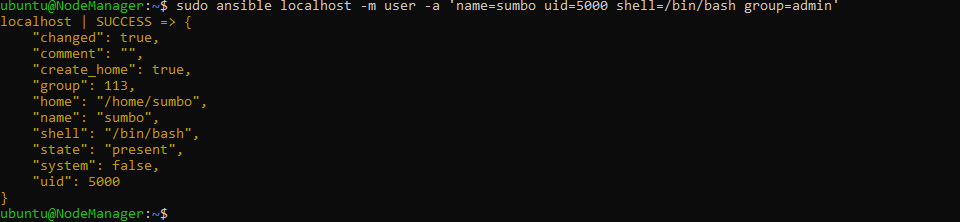




The iventory:



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:

