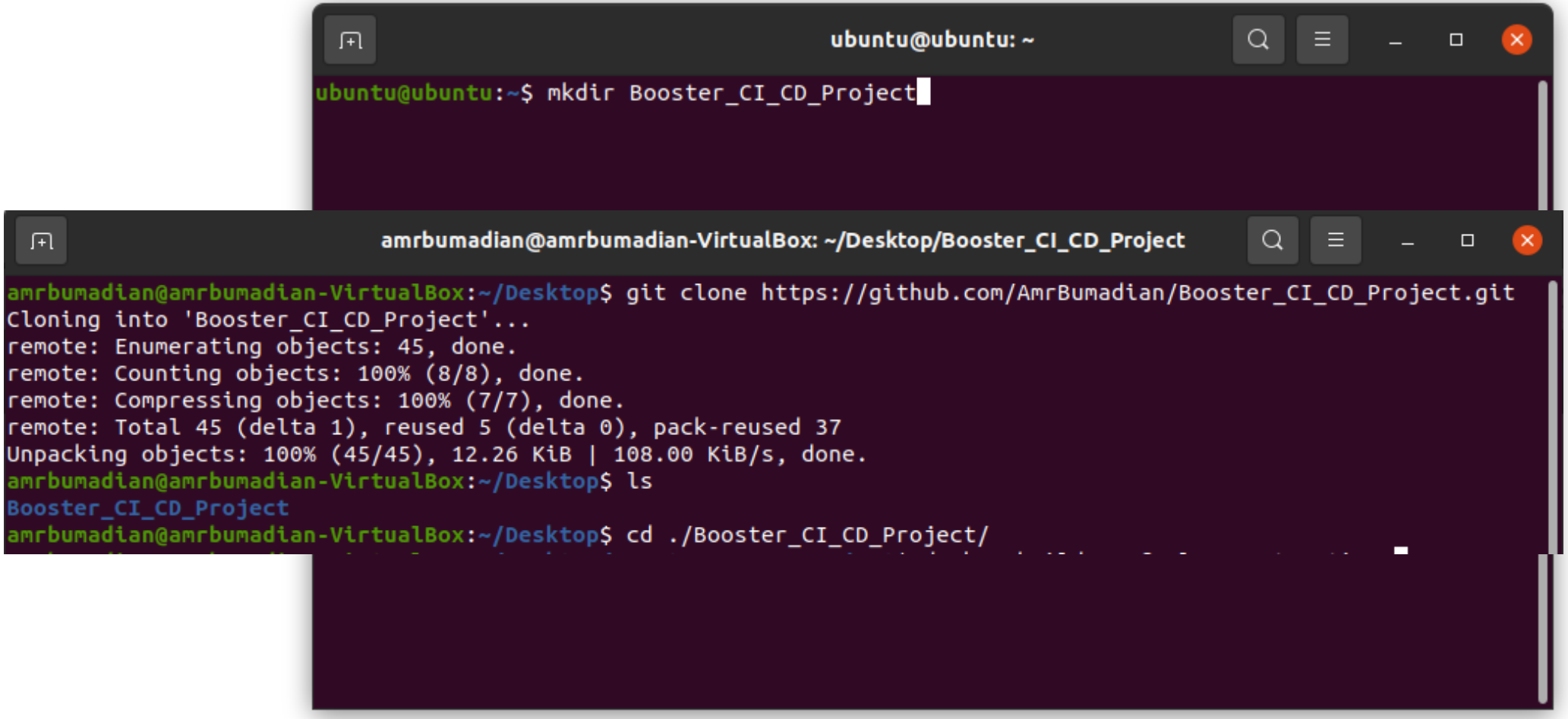
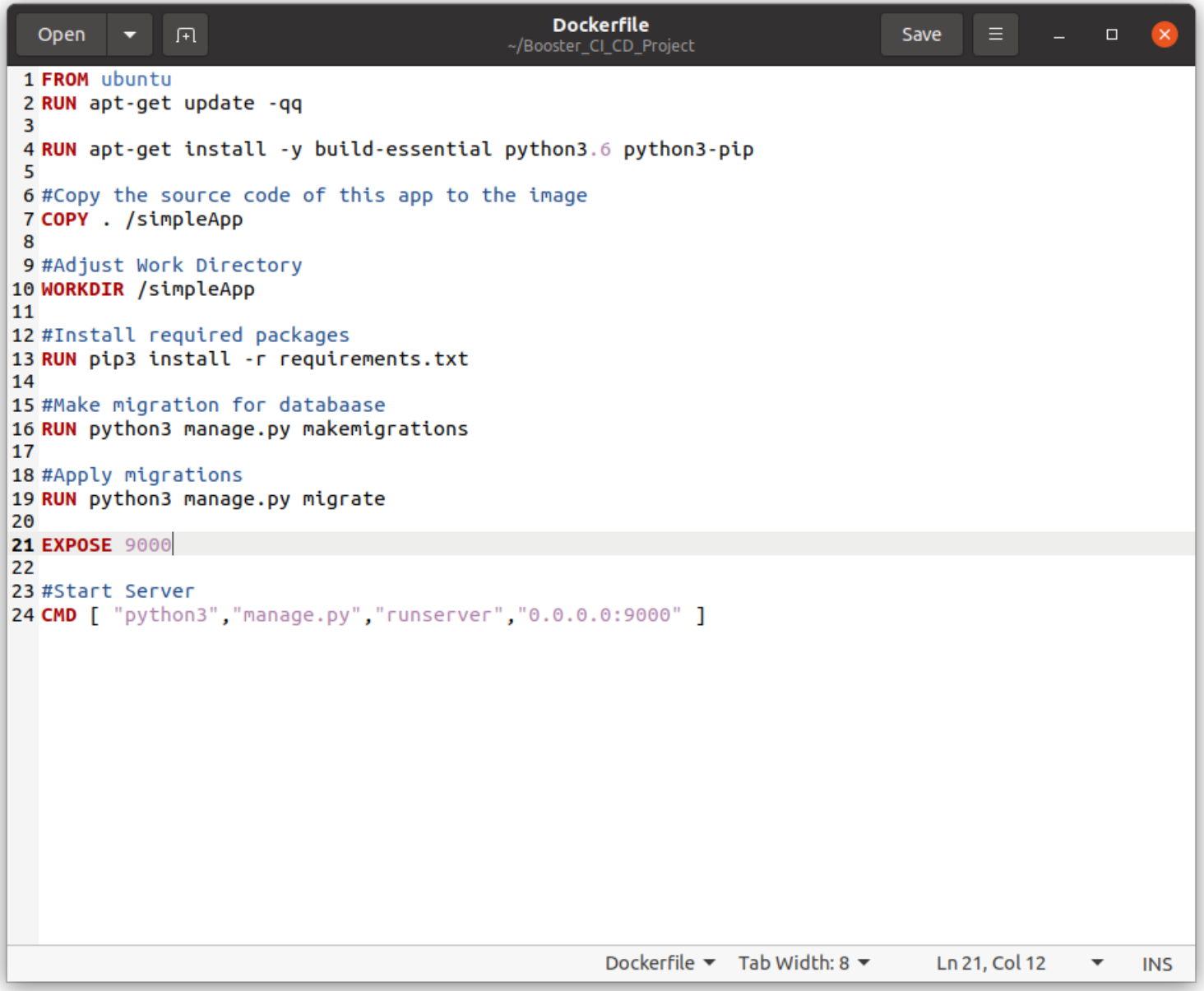
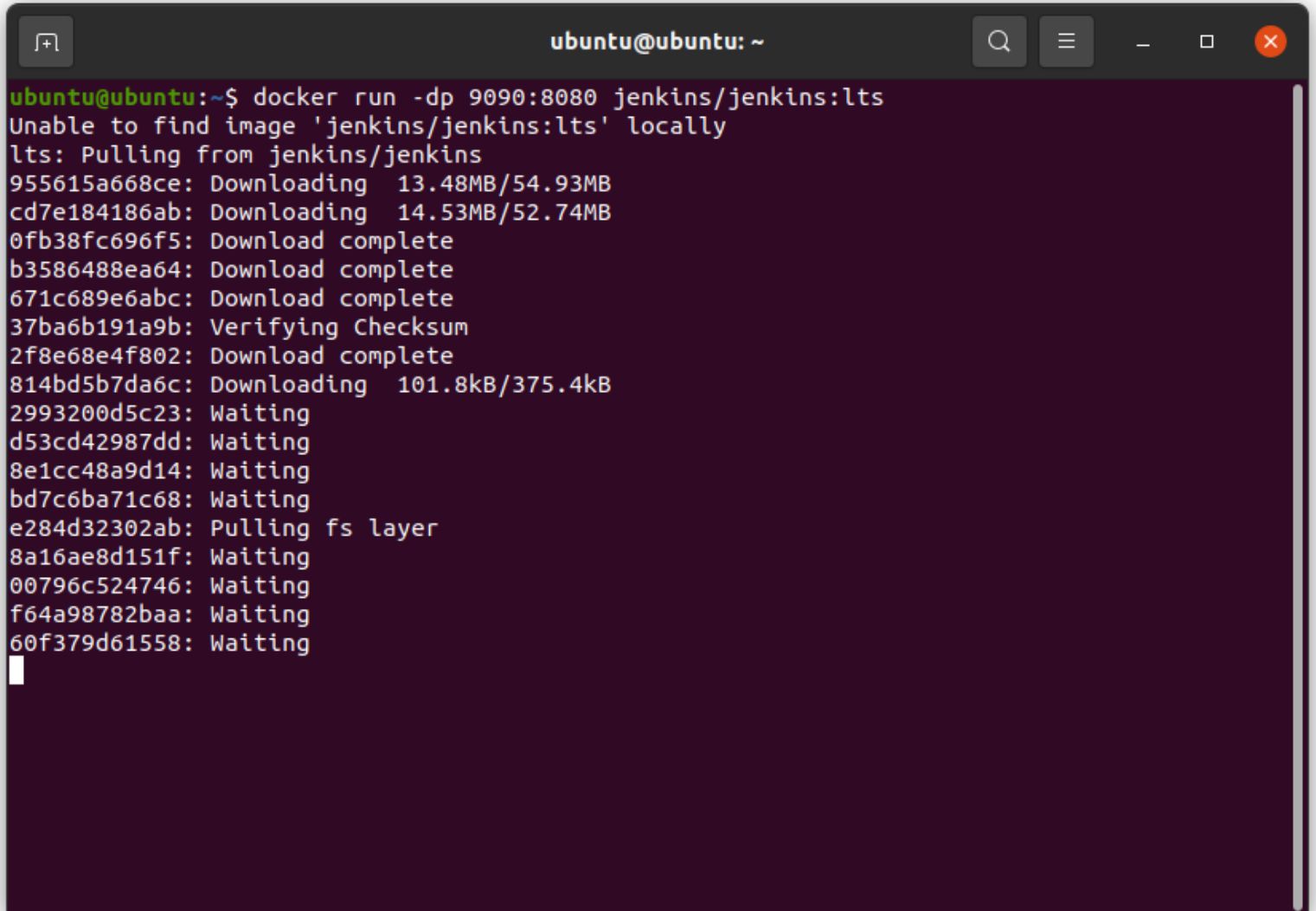
1-Create a new directory for the project, then clone the forked repo form the GitHub repository to the directory.



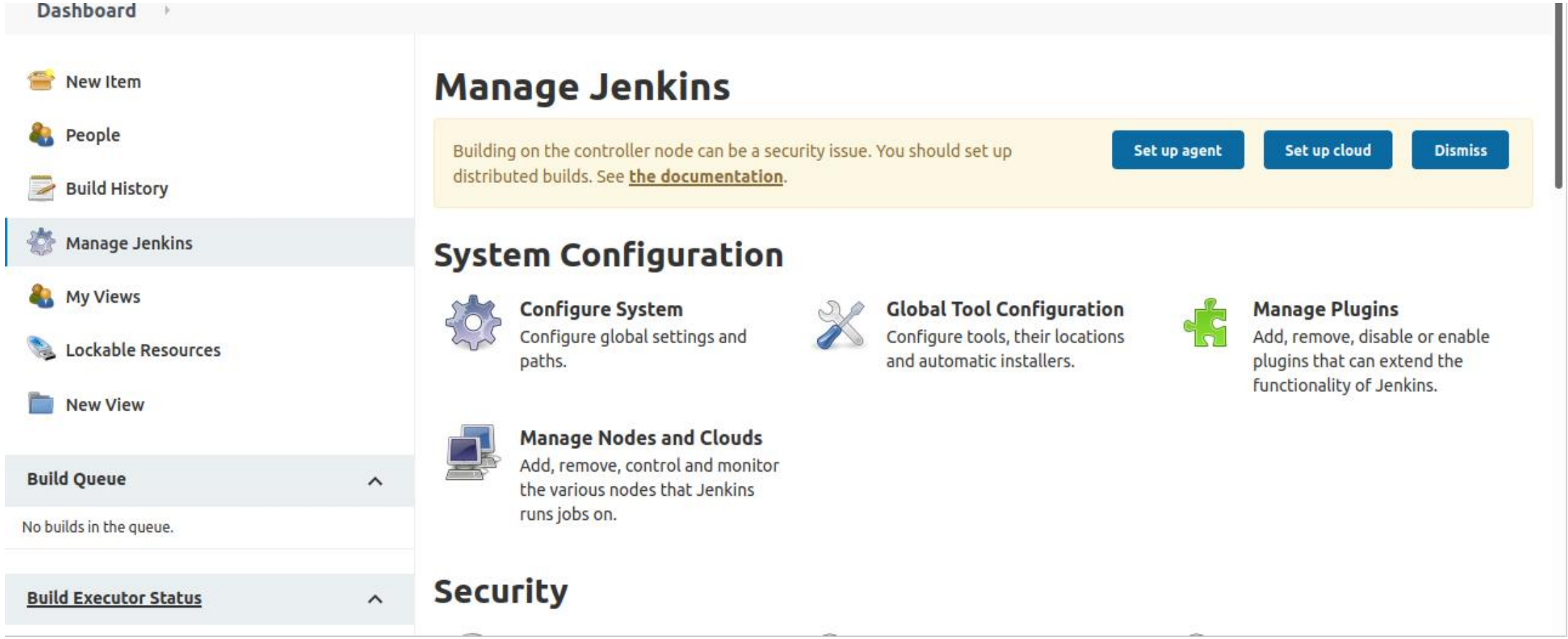
2-Create a new file “DockerFile” in the project directory for building the image.



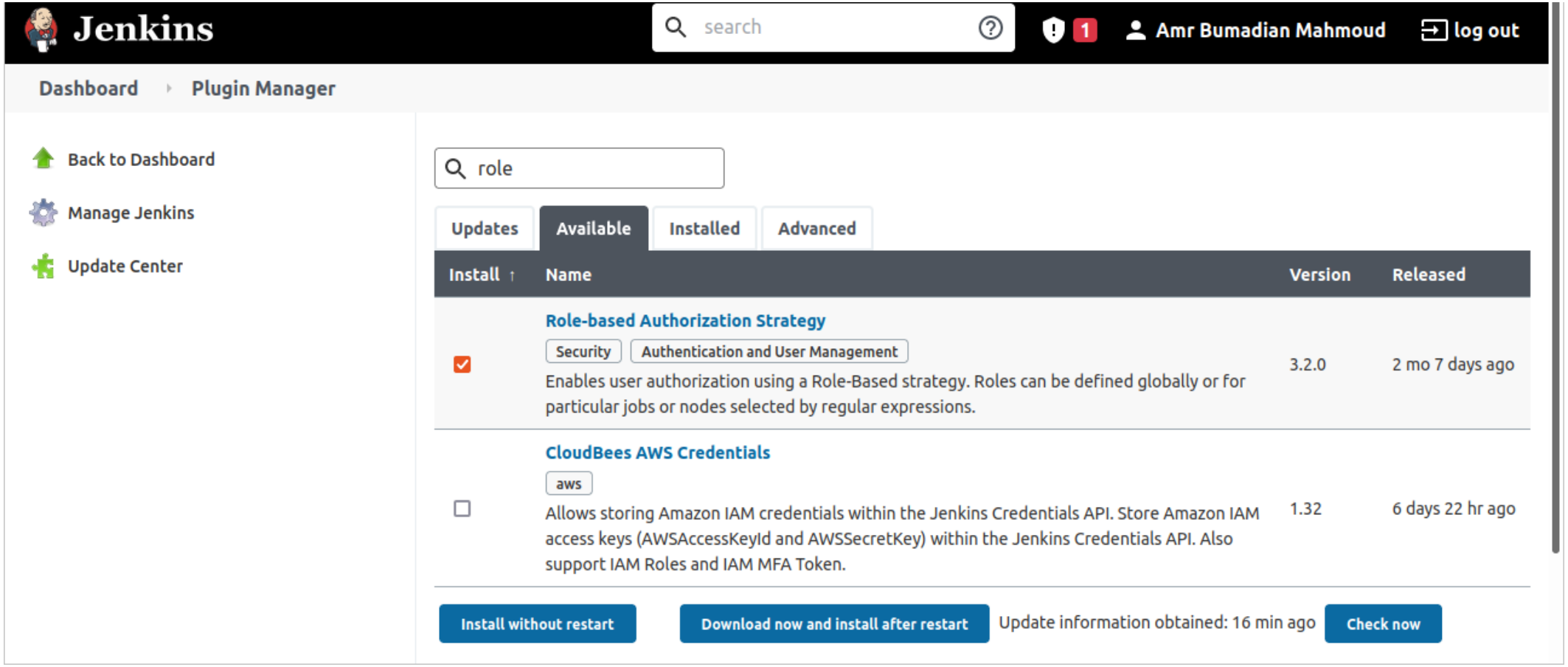
3-Run latest Jenkins image on port 8080 mapped to port 9090, then wait for the image to be pulled, if it is not already downloaded, and run.



4-In the browser, open localhost:9090. That shall start Jenkins, login and create admin account. Then procced to the dashboard.

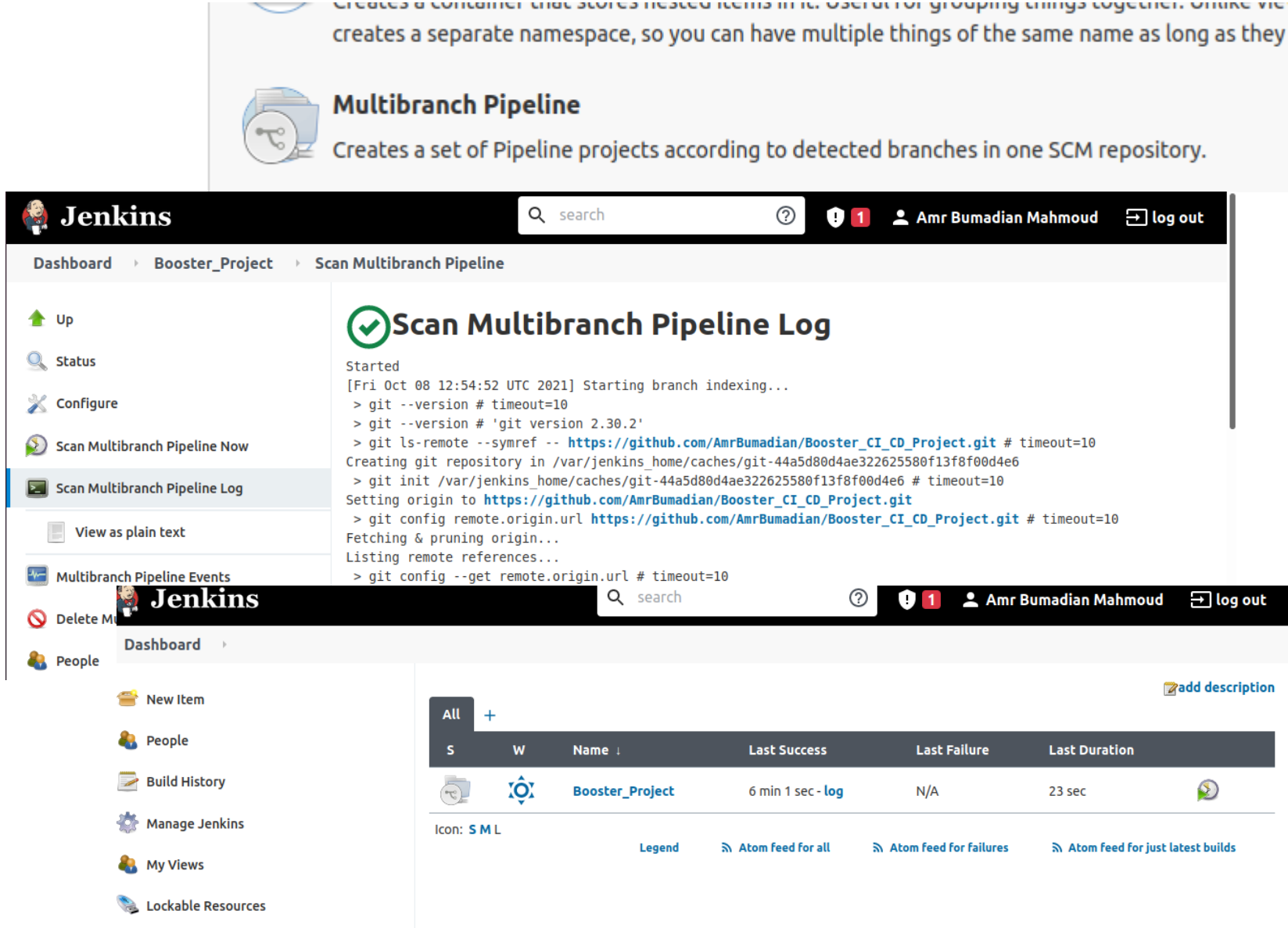


5-In the plugin manager, go to available and search for “Role-Based Authorization Strategy”, check install, download now and install after restart, wait for installing to finish then refresh the page.



6-From the side panel chose new item > multibranch pipeline > choose git then add the link to the GitHub repo.
(Before that add 2 other branches to the repo, development and production)

7-Proceed and wait till the scan finished, the green check mark shall appear. If it took long to finish refresh the page it may be stuck.



8-Create a slave automation file to help automate the process of setting up the Jenkins environment and add the docker client in the slave to be able to use docker, instead of manually setting up the Jenkins environment.

```
FROM ubuntu
USER root

# setup jenkins environment
RUN mkdir -p jenkins_home
RUN chmod 777 jenkins_home
RUN apt-get update -qq
RUN apt-get install -y apt-utils

#install open jdk
RUN apt-get install openjdk-8-jdk -qq

#install ssh server
RUN apt-get install openssh-server -qq
RUN useradd -ms /bin/bash jenkins

# install docker client
RUN apt-get install -qqy \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg \
    software-properties-common
RUN curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -
RUN add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
RUN apt-get update -qq \
    && apt-get install -y docker-ce docker-ce-cli containerd.io
RUN usermod -aG docker jenkins


WORKDIR jenkins_home
CMD [ "/bin/bash" ]

```

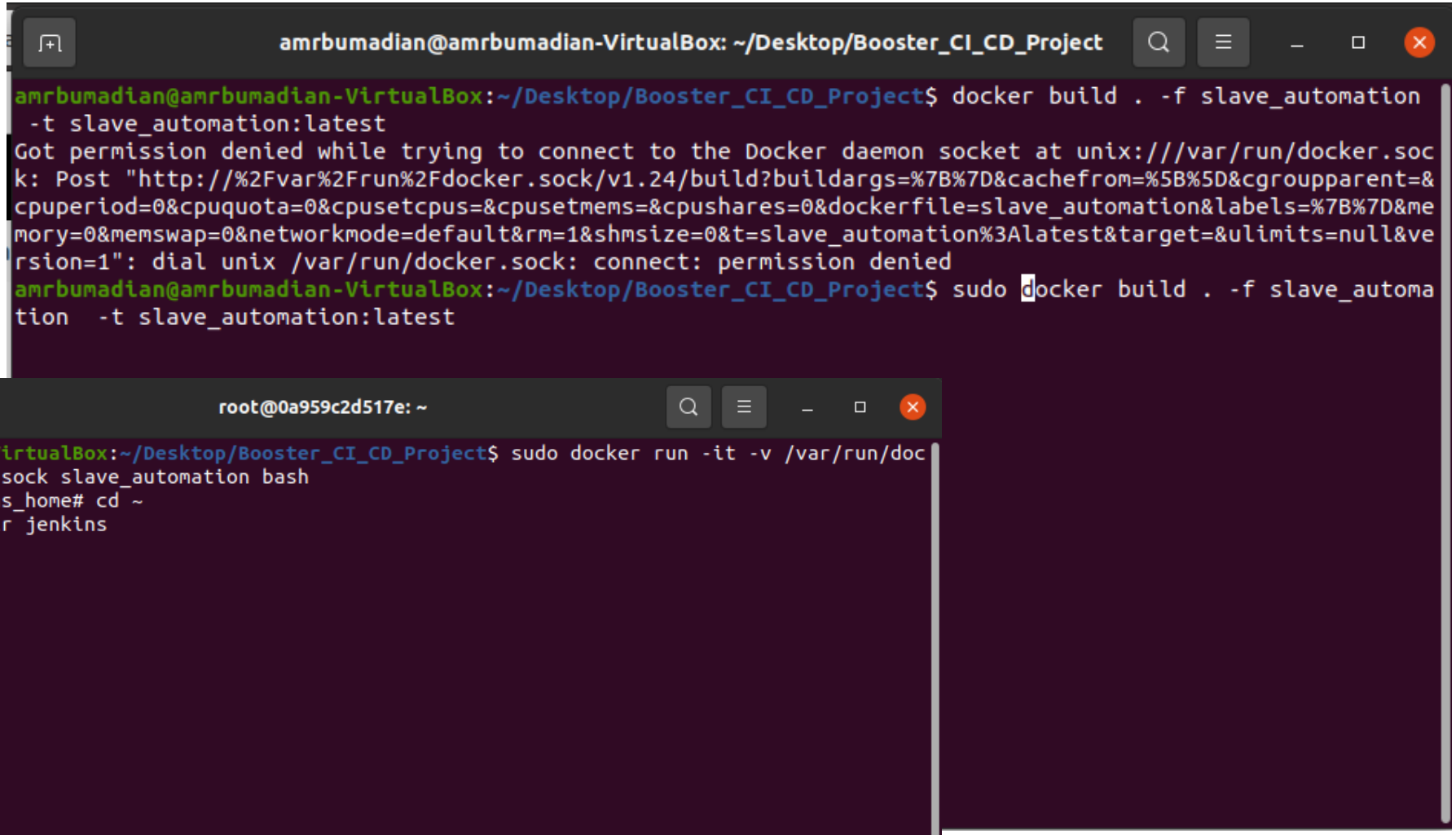
Continue step 8:

-Build the image.

-Run the image to start at bash.

-cd ~

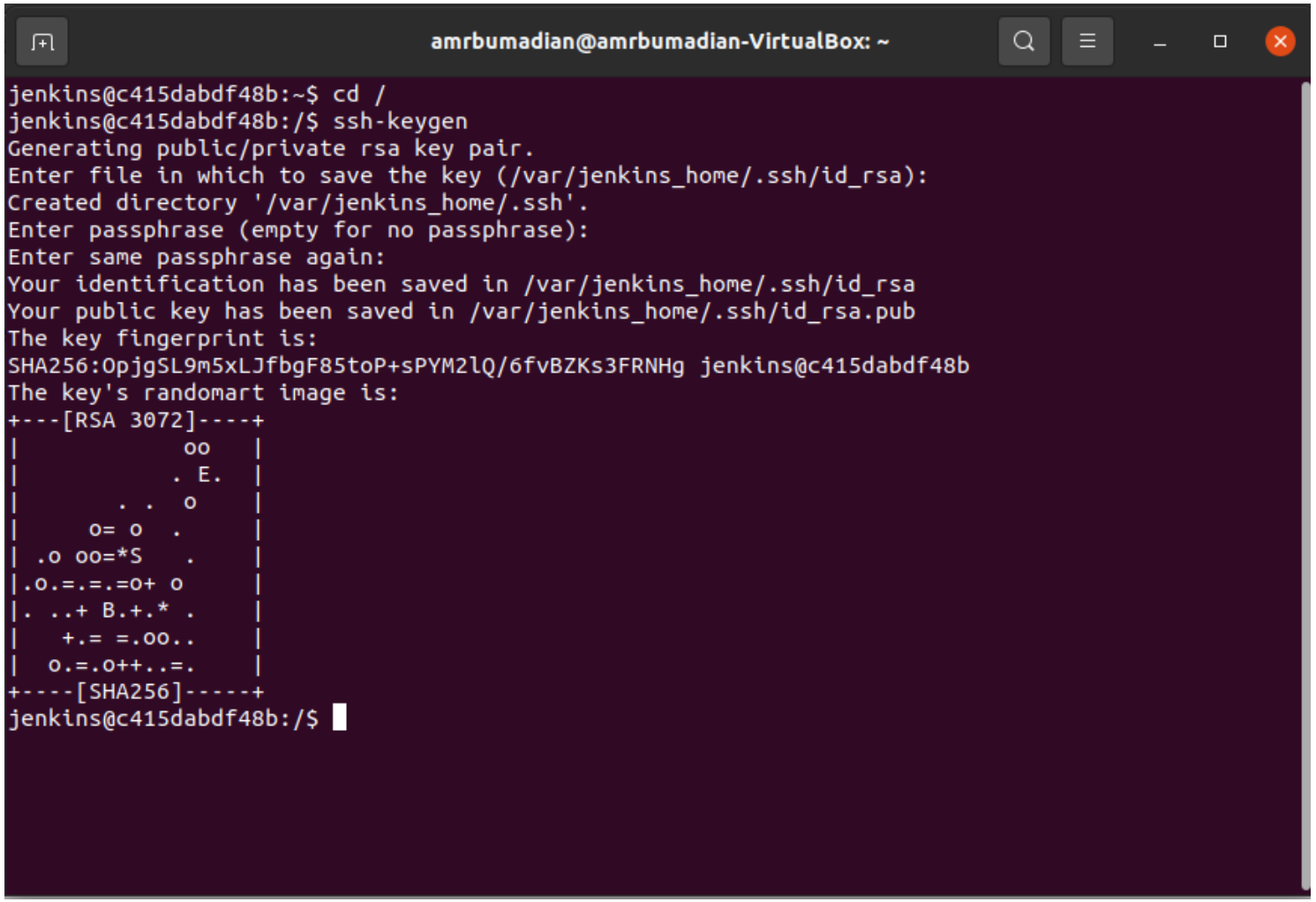
-mkdir jenkins



Continue step 8:

-cd /

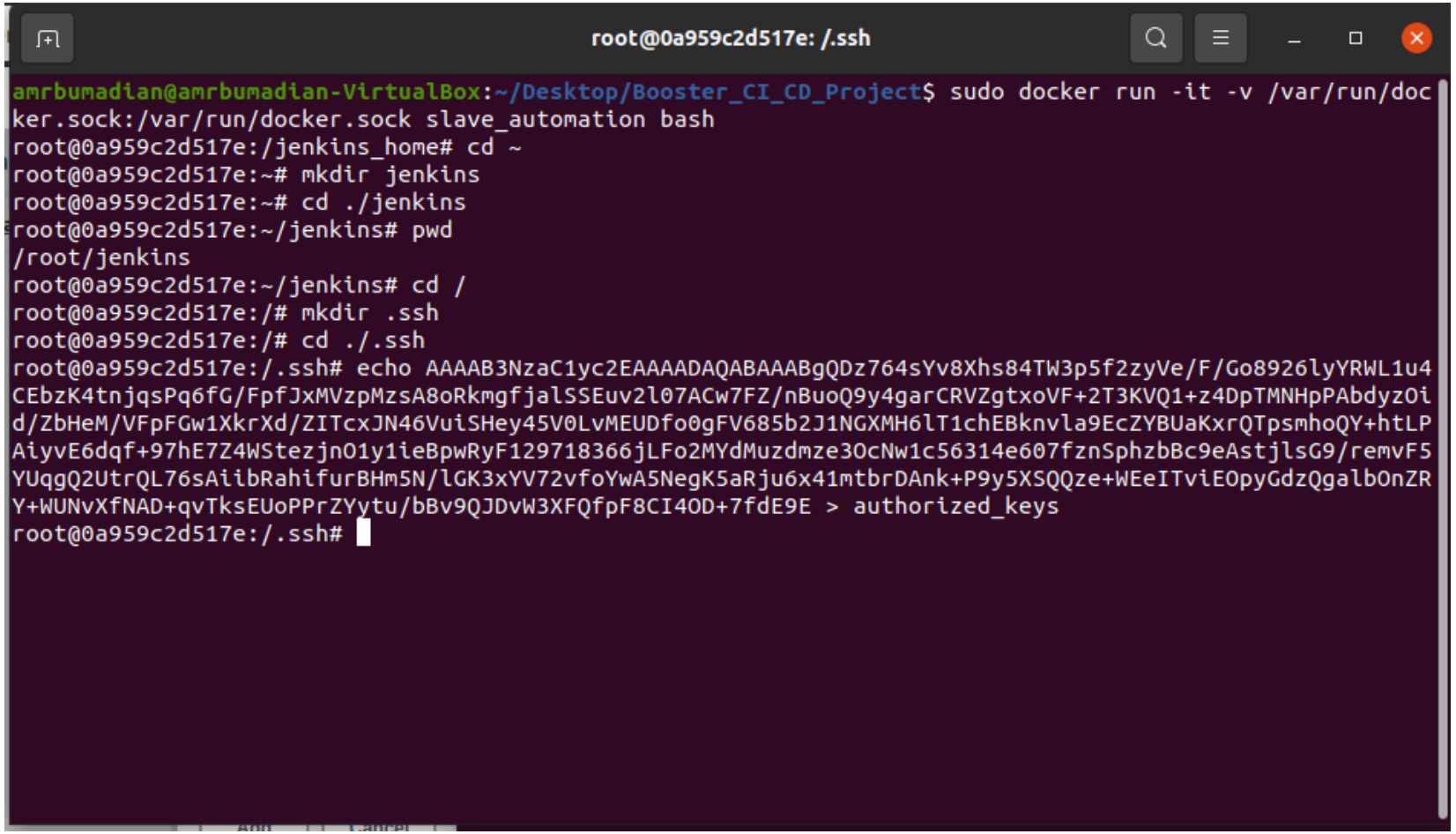
-run ssh-keygen to generate public and private keys, then press “Enter” 3 times.



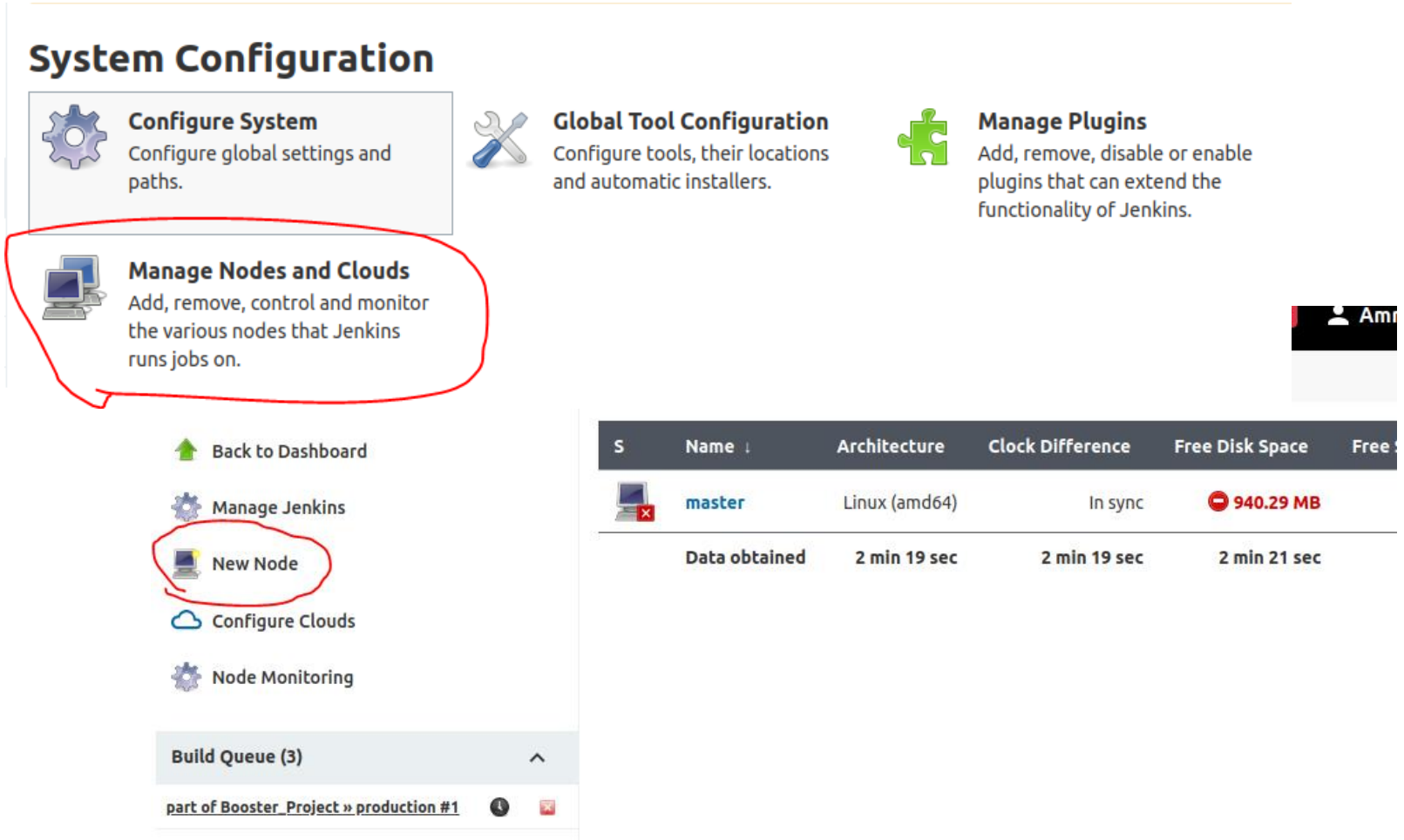
Continue step 8:

Open a new terminal and run bash in the slave automation container already running.

-cd ~
-mkdir Jenkins
- cd ./Jenkins
- Pwd (we will need this path in creating a node)
- cd /
- mkdir .ssh
- cd ./.ssh
- Echo “The public ssh key” > authorized_keys



9-From the dashboard choose manage nodes and clouds, the from the side panel choose add new node.



Continue step 9:

- Docker inspect <id of slave_automation container>

- Copy the IpAddress that will appear below. (will be needed in configuring the node)

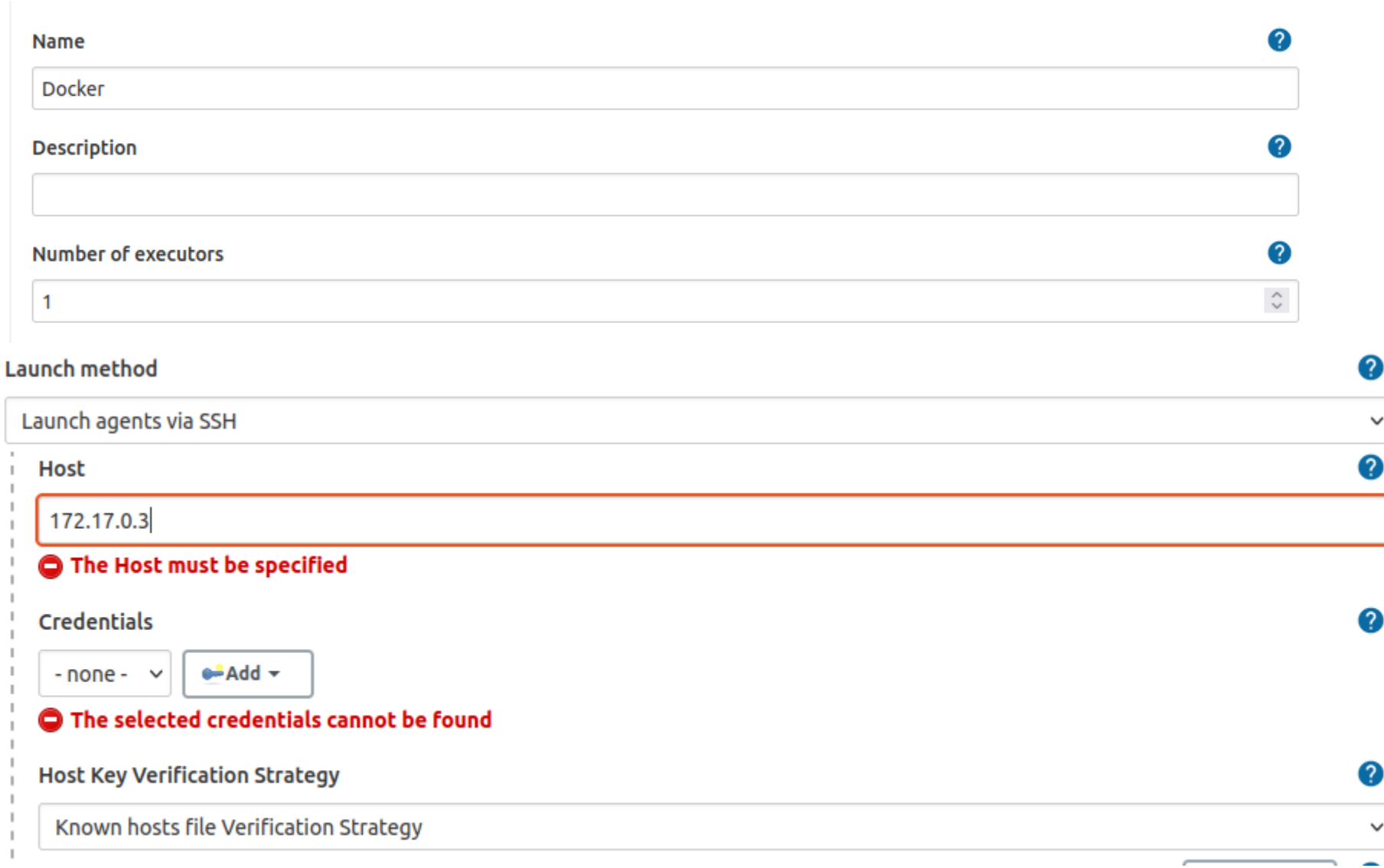
```
amrbumadian@amrbumadian-VirtualBox:~$ ^C
amrbumadian@amrbumadian-VirtualBox:~$ sudo docker inspect 0a959c2d517e
[
    {
        "Id": "0a959c2d517efd8a43018e5447d2e7ea797dff9f14280827fe99ec395c03cb7e",
        "Created": "2021-10-08T13:11:33.473652111Z",
        "Path": "bash",
        "Args": [],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 25435,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2021-10-08T13:11:34.894752464Z",
            "FinishedAt": "0001-01-01T00:00:00Z"
        },
                    "Gateway": "172.17.0.1",
                    "IPAddress": "172.17.0.3",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "MacAddress": "02:42:ac:11:00:03",
                    "DriverOpts": null
                }
            }
        }
    }
]
amrbumadian@amrbumadian-VirtualBox:~$
```

Continue step 9:

- Name the node

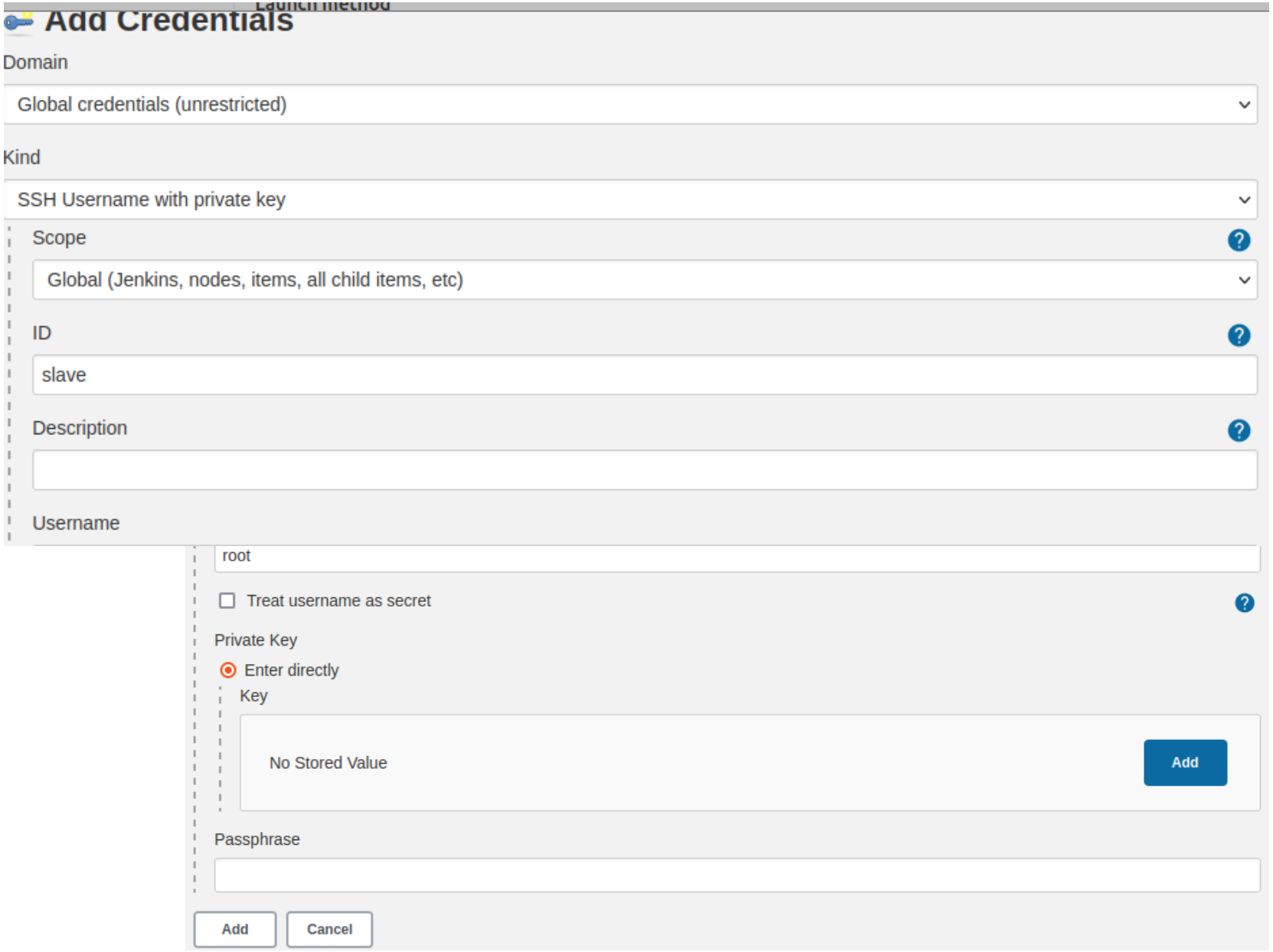
- Add the copied IpAddress in the host then choose add credentials.

- Configure the rest as shown.



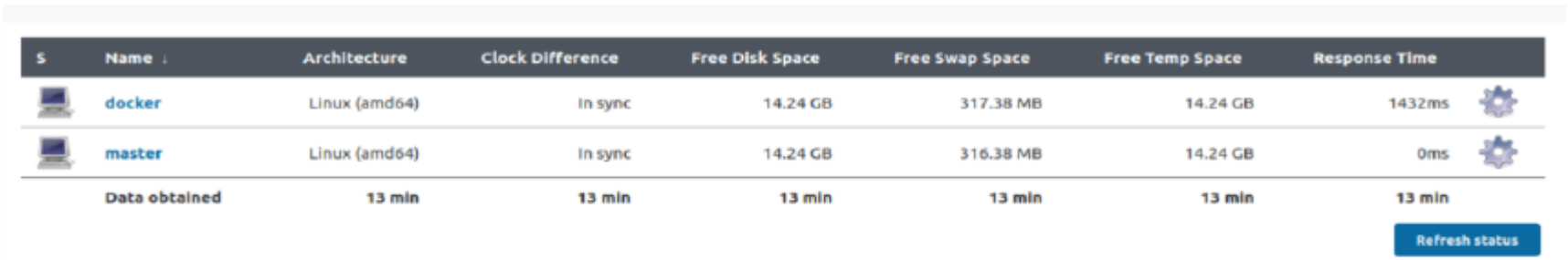
Continue step 9:
- Add the credentials, copy the generated private key and add it in the key input box

- Add the save the node.



Continue step 9:

-Check that the node is successfully authenticated and connected.



S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time	
	docker	Linux (amd64)	In sync	14.24 GB	317.38 MB	14.24 GB	1432ms	
	master	Linux (amd64)	In sync	14.24 GB	316.38 MB	14.24 GB	0ms	
	Data obtained	13 min	13 min	13 min	13 min	13 min	13 min	

Refresh status

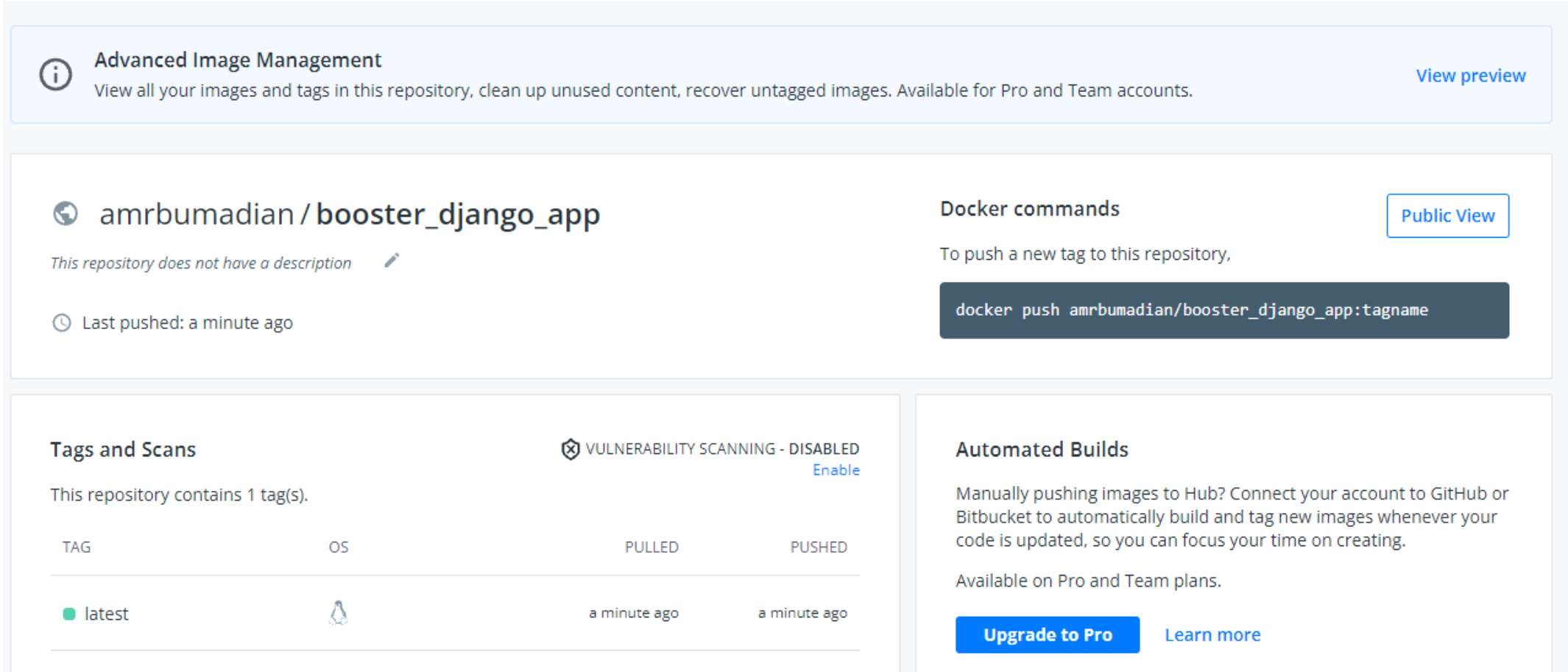
10-Write the Jenkins file, we will stage it as following: preparation, build, push, deploy, and notification for slack.

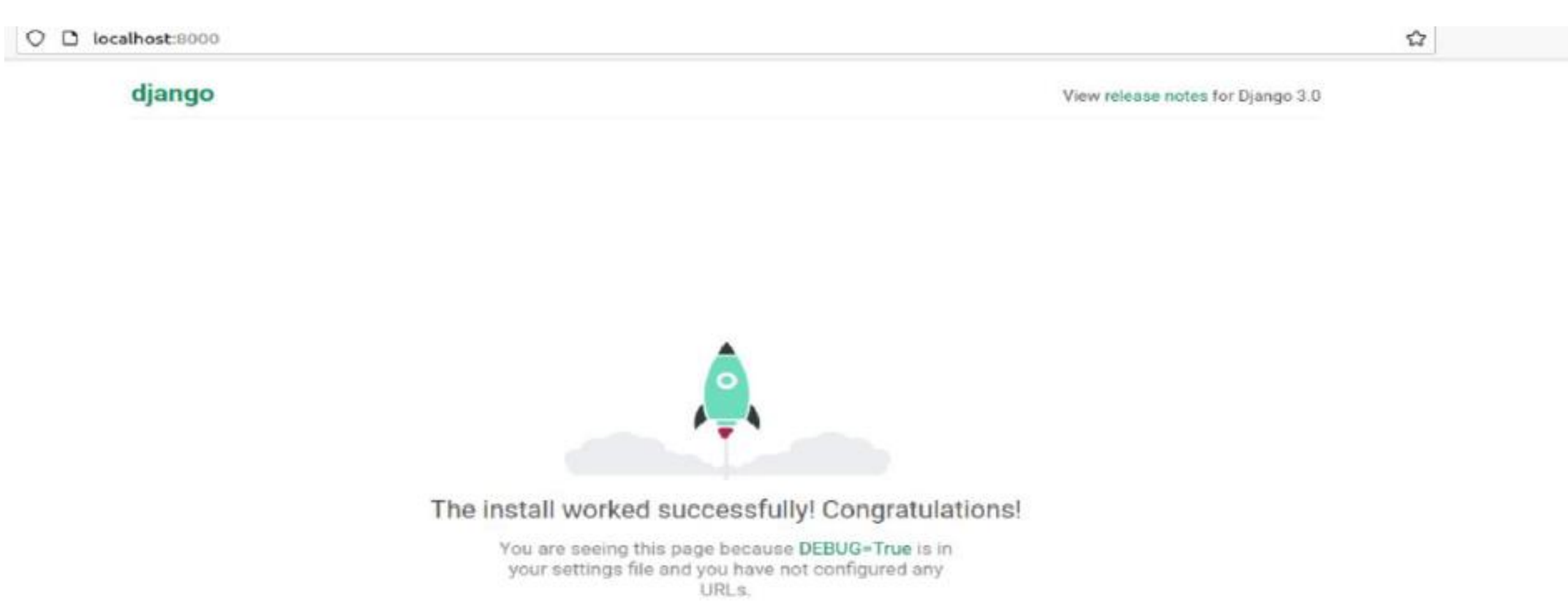
```
pipeline{
  agent {label "docker"}

  stages {

    stage('preparation'){
     steps {
       git 'https://github.com/AmrBumadian/Booster_CI_CD_Project.git'
     }

    }


    stage('build image'){
     steps {
        sh 'docker build . -f Dockerfile -t amrbumadian/booster_django_app:latest'
       }
    }


    stage('push image'){
          steps {
              withCredentials([usernamePassword(credentialsId:"dockerhub",usernameVariable:"USERNAME",passwordVariable:"PASSWORD")]) {

                  sh """
                    docker login -u ${USERNAME} -p ${PASSWORD}
                    docker push amrbumadian/booster_django_app:latest
                  """
              }
          }
       }
    stage('deploy'){
          steps {
              sh 'docker run -d -p 9000:9000 amrbumadian/booster_django_app:latest'
          }
         post {
            success {
              slackSend (color: '#00FF00', message: "SUCCESSFUL")
```

Continue step 10:

-Add login credentials for the DockerHub in the global credentials.

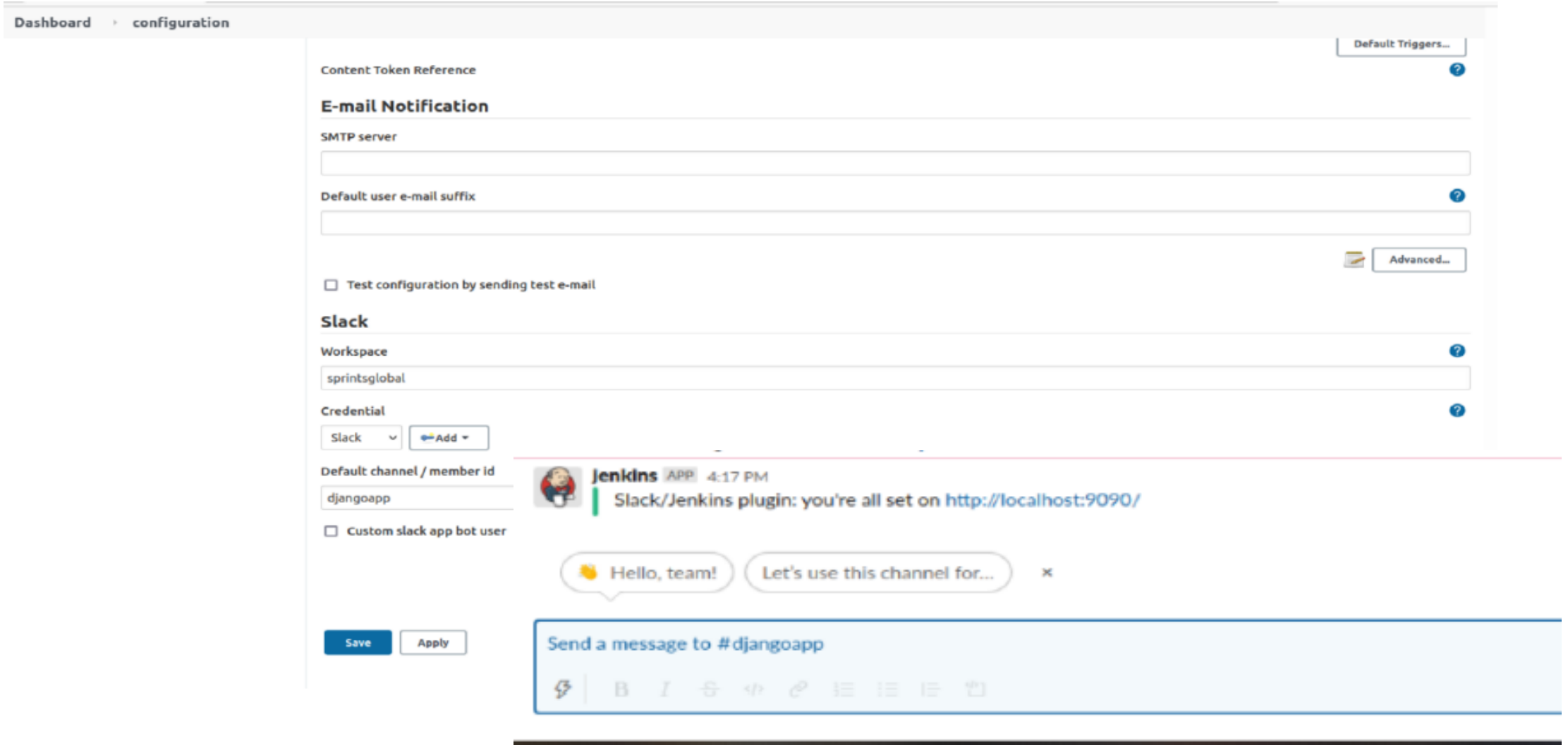
-Build the pipeline.





11-Connect to the website at localhost:8000

12-Add Slack plugin



ALL DONE