

GitHub Repo's Link : [https://github.com/Ahmed-Arafat10/Booster\\_CI\\_CD\\_Project.git](https://github.com/Ahmed-Arafat10/Booster_CI_CD_Project.git)

First of all, I have to create a {dockerfile} to modify image of ubuntu in which slave node in Jenkins Server will run commands of {Jenkinsfile} on it, this image must contains :

- 1) Docker client CLI {to run docker commands}
- 2) OpenJDK
- 3) A created Directory called {Jenkins} to save files in it
- 4) SSH {install it then run service when image is build into a container}
- 5) Authorized\_keys file that contains public key in path {/root/.ssh}
- 5) Git

```
# Task inline 160 in Jenkins#3 file , this is the dockerfile
FROM ubuntu
USER root
RUN apt-get update -qq
RUN apt-get install -y apt-utils
RUN apt install -y apt-transport-https ca-certificates curl 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` test"
RUN apt update
RUN apt-get install docker-ce -
# ---
RUN apt-get update
RUN apt-get install openjdk-8-jdk -
WORKDIR /root
WORKDIR /root/Jenkins
RUN pwd > Path
RUN apt install openssh-server -
# RUN service ssh status
WORKDIR /root/.ssh
CMD cd /root/.ssh && service ssh start && yes | ssh-keygen && mv id_rsa && mv id_rsa.pub && cat /root/.ssh/id_rsa.pub > authorized_keys
WORKDIR /
RUN apt-get install git -
RUN ls /root/.ssh

```

Creating an image from above dockerfile using command:  
\$ sudo docker build . -f dockerfile\_ -t ubuntu\_docker\_cli\_all:v1  
Then Image is created successfully

```
Processing triggers for systemd (245.4-4ubuntu3.13) ...
Removing intermediate container c7594e99f4d
--> 09812e54d6f8
Step 17/21 : WORKDIR /root/.ssh
--> Running in b5a85797c4a9
Removing intermediate container b5a85797c4a9
--> f9bc162eed46
Step 18/21 : CMD cd /root/.ssh && service ssh start && yes | ssh-keygen && mv id_rsa && mv id_rsa.pub && cat /root/.ssh/id_rsa.pub > authorized_keys
--> Running in 9a52b713cd17
Removing intermediate container 9a52b713cd17
--> f5bcbae87d3
Step 19/21 : WORKDIR /
--> Running in 35cdb7f0f0f9
Removing intermediate container 35cdb7f0f0f9
--> 2090f423f5a8
Step 20/21 : RUN apt-get install git -y
--> Running in 3a9d79e70b1f
Reading package lists...
Building dependency tree...
Reading state information...
git is already the newest version (1:2.25.1-1ubuntu3.2).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Removing intermediate container 3a9d79e70b1f
--> 27dbd7dfa0d
Step 21/21 : RUN ls /root/.ssh
--> Running in 0a9334af9b5a
Removing intermediate container 0a9334af9b5a
--> 0a9334af9b5a
Successfully built d4381ede71e8
Successfully tagged ubuntu_docker_cli_all:v1
```

Now I can create a container from image with port [8080] referring to [8080] and a volume mapping of path /var/run/docker.sock to be able to run Docker CLI

Note: as in Line #21 in dockerfile, I used [CMD] that will run a series of commands when docker container is created. These command are: enable SSH service, generating a public & private key and copying public key in Authorized\_keys file

```
root@ef031c81a7d4:~# sudo docker container rm V
Error: No such container: V
root@ef031c81a7d4:~# sudo docker container rm 81725c95b015
81725c95b015
root@ef031c81a7d4:~# sudo docker run -it -p 8080:8080 -v /var/run/docker.sock:/var/run/docker.sock ubuntu_docker_cli_all:v1
 * Starting openshhd Secure Shell server sshd
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in y
Your public key has been saved in y.pub
The key fingerprint is:
SHA256:0EjC0LPqYR1Kg32OkSzvvoFCdPWWKc/xDr5xWH2hTQ root@ef031c81a7d4
The key's randomart image is:
+---[RSA 3072]---+
| ++. .
| .- = o
| +=.o oo . E
| oo.. . . o
| .oo. .S. + . .
| =o . + oo+ o .
| .+= + o+ o .
| . =o+...
| .+ oo.
+---[SHA256]---+
root@ef031c81a7d4:~# sudo docker container ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
32ede207bf9c jenkins_with_docker:vi "/sbin/tini -- /usr/.." 18 hours ago Up 6 hours 50000/tcp, 0.0.0.0:50->8080/tcp, :::50->80
80/tcp affectionate_dijkstra
root@ef031c81a7d4:~# sudo docker container ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
ef031c81a7d4 ubuntu_docker_cli_all:v1 "/bin/sh -c 'cd /root; stoice_kilby'" 18 seconds ago Exited (0) 11 seconds ago
32ede207bf9c jenkins_with_docker:vi "/sbin/tini -- /usr/.." 18 hours ago Up 6 hours 50000/tcp, 0.0.0.0:50->8080/tcp, :::50->8080/tcp, :::50->8080/tcp affectionate_dijkstra
root@ef031c81a7d4:~# sudo docker start ef031c81a7d4
root@ef031c81a7d4:~# sudo docker exec -it ef031c81a7d4 bash
```

As I have exited from container & container has stopped, I have first to start it again then to enter in its bash terminal {Enter inside container} then make sure that SSH is running, docker & git are installed and Jenkins Directory is created.

Everything is working perfectly 😊

```
root@ef031c81a7d4:~# service ssh status
* sshd is running
root@ef031c81a7d4:~# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@ef031c81a7d4:~# docker --version
Docker version 20.10.8, build 3967b7d
root@ef031c81a7d4:~# git --version
git version 2.25.1
root@ef031c81a7d4:~# cd /root/
root@ef031c81a7d4:~# ls
Jenkins
root@ef031c81a7d4:~#
```

I have created a script to automatically copy Private key from container to host & then copy just lines that contains word [IPAddress] from output of \$ docker inspect command { which shows all metadata of a container} and save it in a file called IPAddress. All of this can be done just by passing <Container\_ID> as a parameter while executing script file

```

1 // This script is used to save Private Key of container slave in your host (will be used in credential menu in jenkins server)
2 // then copy file that contains IPAddress of container slave (will be used for host option while creating new node)
3 #! bin/bash
4 docker cp $1:/root/.ssh/id_rsa /home/arafat/Desktop/jenkinsproject/PrivateKey
5 docker inspect $1 | grep -i ipAddress > IPAddress

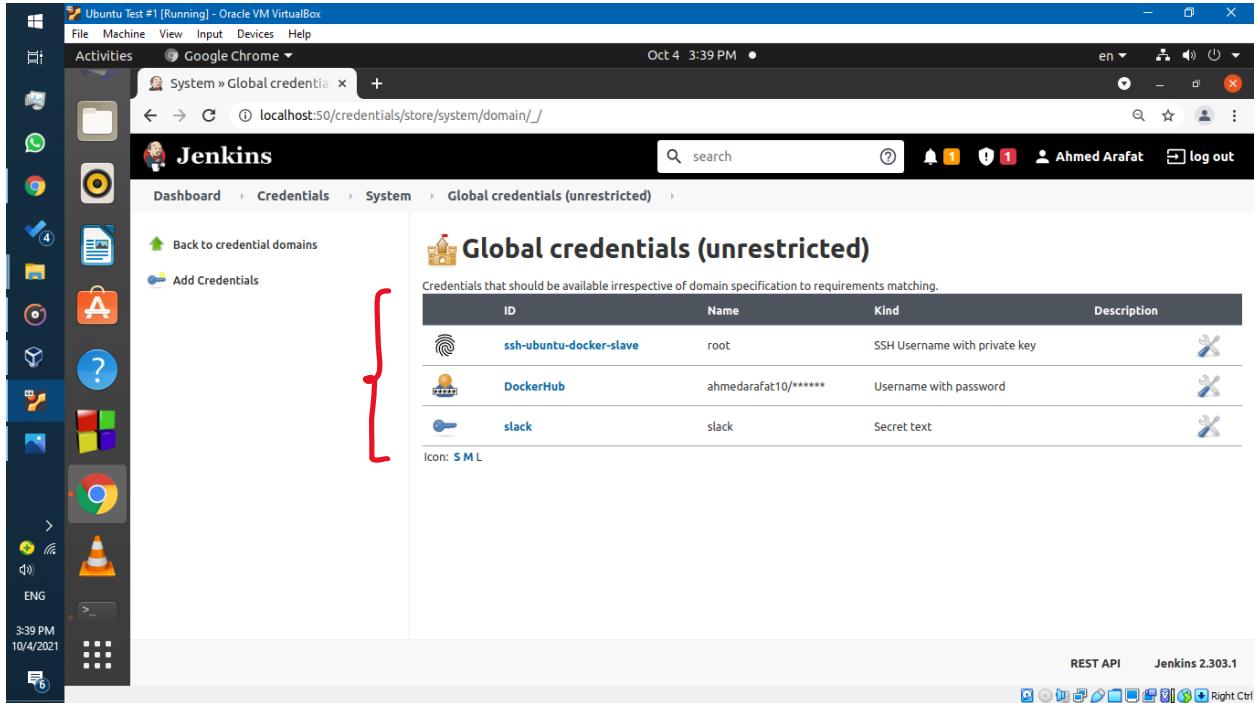
```

This is just implementing previous step on terminal, now I have Private key and IPAddress of Ubuntu Docker Container which will be used to create a slave node inside Jenkins Server

```

root@arafat-VirtualBox:~/Desktop/jenkinsproject$ su
Password:
root@arafat-VirtualBox:/home/arafat/Desktop/jenkinsproject# ls
Booster_CI_CD_Project DOCKER dockerfile dockerfile_ dockerfile1 dockerfile2 Jenkins_Docker script.sh
root@arafat-VirtualBox:/home/arafat/Desktop/jenkinsproject# source script.sh ef03c81a7d4
root@arafat-VirtualBox:/home/arafat/Desktop/jenkinsproject# ls
Booster_CI_CD_Project DOCKER dockerfile dockerfile_ dockerfile1 dockerfile2 IPAddress Jenkins_Docker PrivateKey script.sh
root@arafat-VirtualBox:/home/arafat/Desktop/jenkinsproject# cat IPAddress
"SecondaryIPAddresses": null,
"IPAddress": "172.17.0.3",
"IPAddress": "172.17.0.3",
root@arafat-VirtualBox:/home/arafat/Desktop/jenkinsproject# cat PrivateKey
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzIa1cXktdjEAAAAABG5vbmlUAAAEBm9uZQAAAABAAABlwaAAAAdzc2gtcn
NhAAAawEAAQAAAAYEAtnVFZFLYpMsZKfuRuy3pVovWChzPBCxyQpLjyf-TfQV9qQSS
07Hwx2JAKQS/801Vijpxg0LHiDw0D20v9L0n/rnRxLB8CqxnV3Ns5/qWw+aE5Z2Nu
Y5m0S372txjUCqdLdxz2/pkaE+QtIo0vdh6AQJ/NJJQJUf7tFRqNyhb2VHlqoappBtO2
dZhaCpxSAWKN6oSlw7+sZzRS0fZ0/bgtldsQ@QaN/LfIjhbxBF8TxMHe+YUooTy+mR4S
KYETLajcMmYdyC7/V/Sy/wxnn0Yp5YYEWkMTa6sz/insuT7Aw/79JZ044Yig4osaUNy
gJT2AymfFVQm1y2x6UJIE/IhaoPw7N3xaermTRDx48/6R7kLP1gAoavH1iuMgKcbXa8z
32S72qUrQ9FkvjhA1vNx8BQXz8BnPd87mTj0InNfVvHnyi0+LHAh1SHQ0itvsXgZjsn
/A315rjBY4/2lQ2x/7Mt94eAagPhBSjUm9xFmjAAAFiAEySDIBMkgvAAAB3NzaC1yc2
EAAGBALZ1RwRe2KTBbGn7kbstoval1goc2j/Asckd4o163/k30fakEkt06x8Md1cK
LP/A9Wjl14Kv9Cx9Q1jq2Tr1/Tp/6516yywFqsZ1dzbof6ltsphn0Wdjbm0Zjk+t+bC
1Aqo8YdveGsmhPKHvnKFkPYewgEC/zSSUFLxbexuA1580d1TjaqCraoQUNwNwYwqcUgFl
jY+qE180/+Gc6Ujn2dP24LZXbTAkjfSRStx280X/E1Sh3vgFKKE8vpkeEpGBEyko3Dj
l4ncgu/7/0sv8Mz+tgKe2BMcLk2urM/457Lk+wMP+/SWU00EGio0KEn1DcoCU9gMphVUJ
tctseLCsBPYIVwg071uzd12nq5k0Q18UNP+ke5Cz9AgQlx9Yrj1CnGlwPM99kn06rkaVt
ZFYxwNbzzCQfF8/AZ173f0SkycdJzX1VNIz8tPiixwIzeR0NirVbf4GSbj/WNy0a4wWOP
9puUnsf+zLfeHgGoD4Uo1Dpc6xTIAAAAAMBAAEAAAGAf9uPwKSTR7gF5t5003AJewZLW
Sq892oYw6yGHZGT8nk3lmx1mkWlxmx1L6jTdxByMzPdxxm54SxJx08CEt4M5TuqoWzgE
0kk33FXD6fVPL08g7iTooyWDohL97zZGGVtj3wK1fcYro+b67046wFFxv50d6nd20C+
7nsapb0uLht-E-Td-rc-EEUo-ut-ld-ge-ec-ut-1u-13-E-3-V-a-T-4-f-a-l-t-a-d-u-t-y-v-o-i-n
```

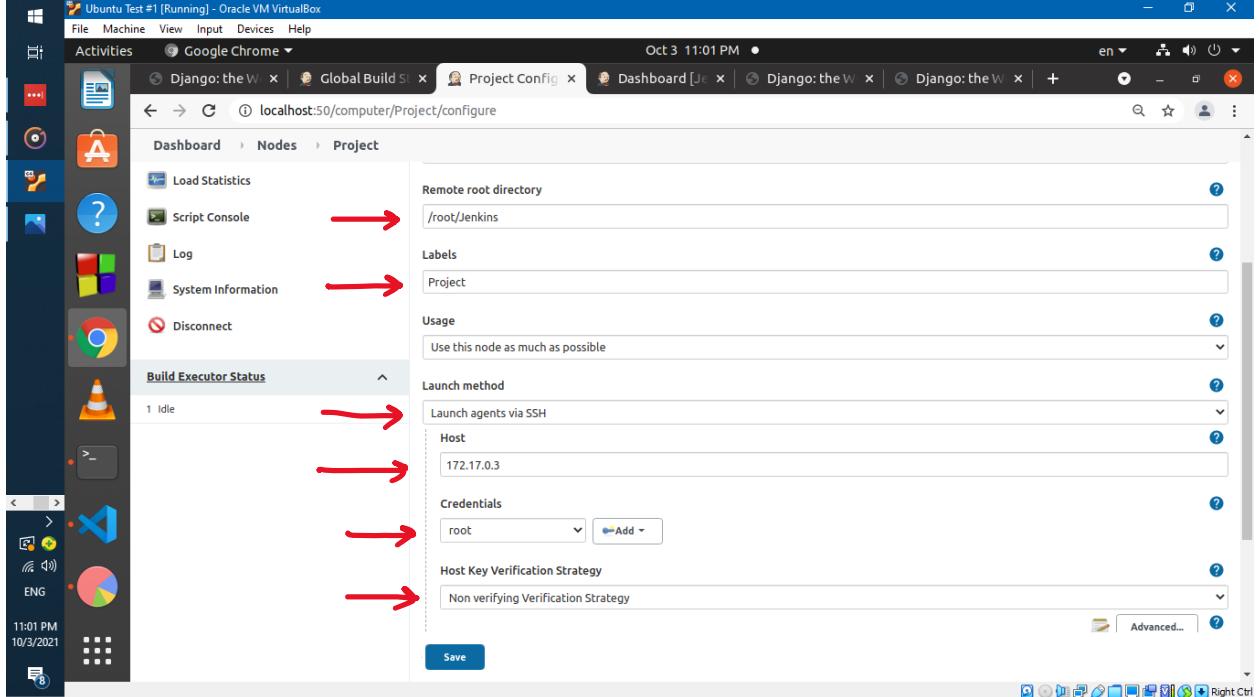
I have added all required credentials from {Credentials Menu} that will be used along project  
Note: Since my GitHub repo of project is public, I don't have to add my credentials of my GitHub Account



A screenshot of a Linux desktop environment showing a Google Chrome window with the Jenkins Global credentials page. The page displays three credentials: ssh-ubuntu-docker-slave (Kind: SSH Username with private key), DockerHub (Kind: Username with password), and slack (Kind: Secret text). A red bracket on the left side of the credentials table groups them together.

After going to {manage node menu} and then creating a new node called “Project”. Now changing remote Dir. To path /root/Jenkins then copying IPAddress to host field  
And then choosing credential named {root}

Note: Host key verification strategy option -> Non verifying verification strategy

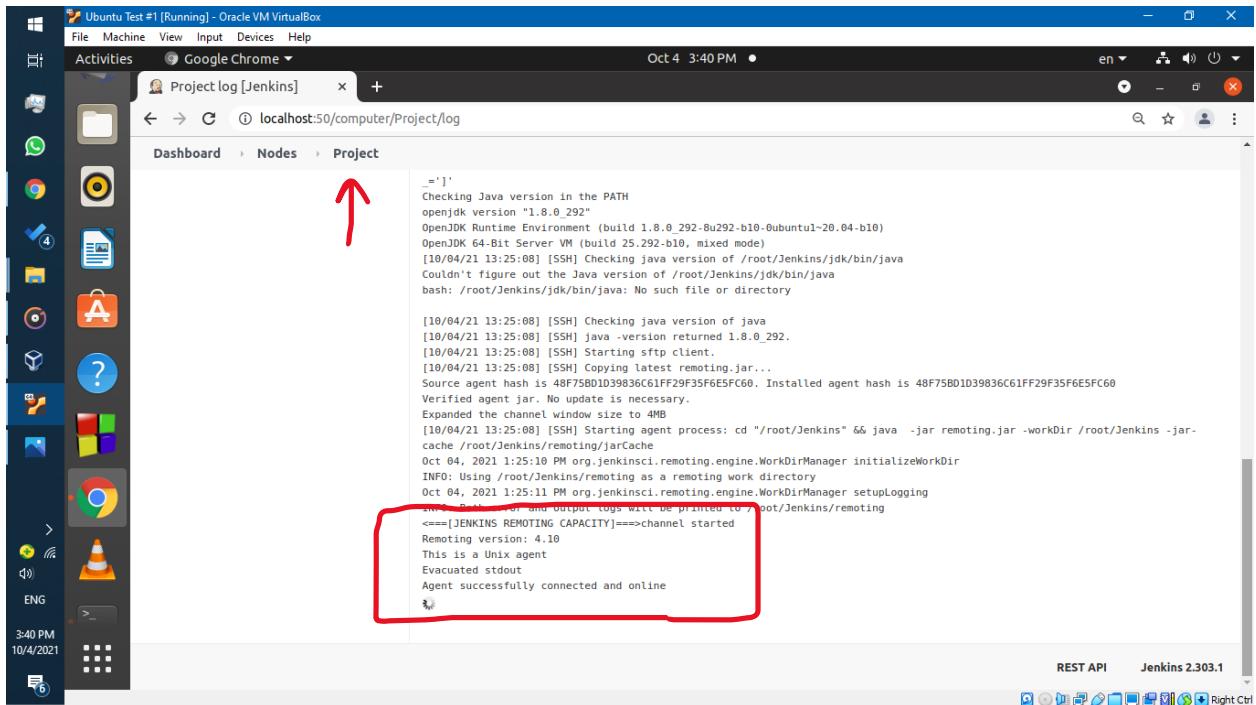


A screenshot of a Linux desktop environment showing a Google Chrome window with the Jenkins Manage Node Project configuration page. The page shows a node named "Project" with the following settings:

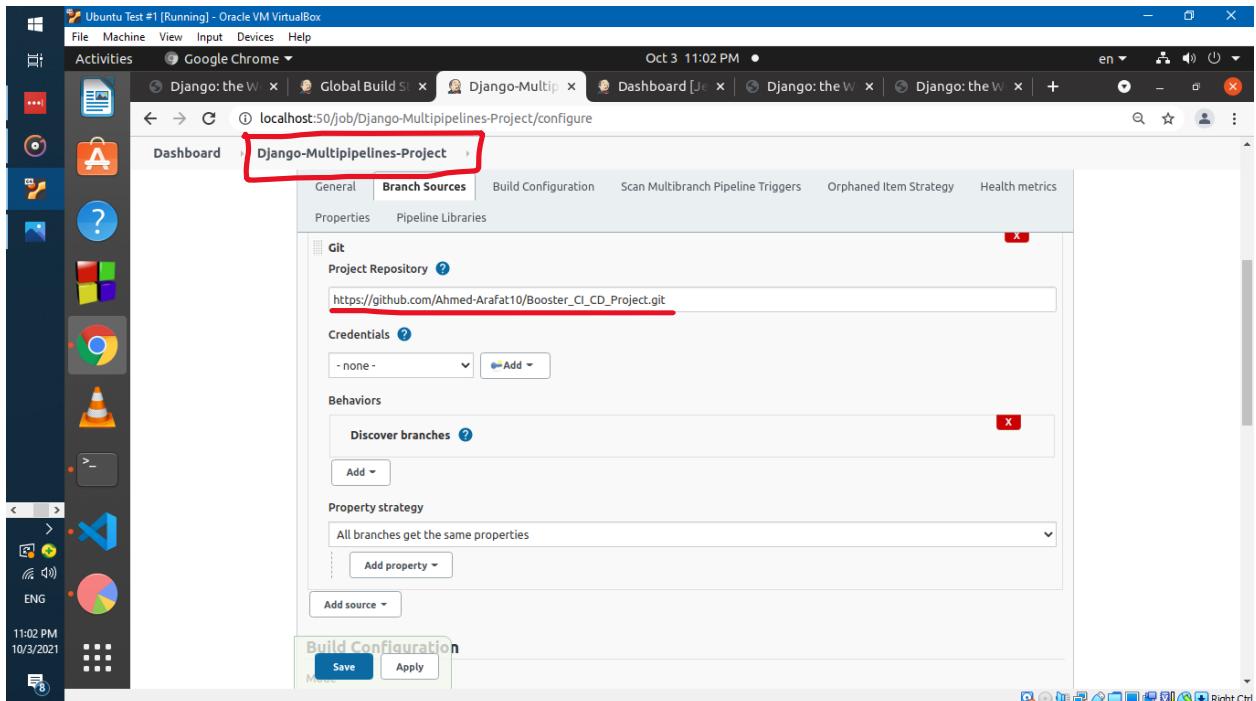
- Remote root directory: /root/Jenkins
- Labels: Project
- Usage: Use this node as much as possible
- Launch method: Launch agents via SSH
- Host: 172.17.0.3
- Credentials: root
- Host Key Verification Strategy: Non verifying Verification Strategy

Red arrows point to each of these configuration fields.

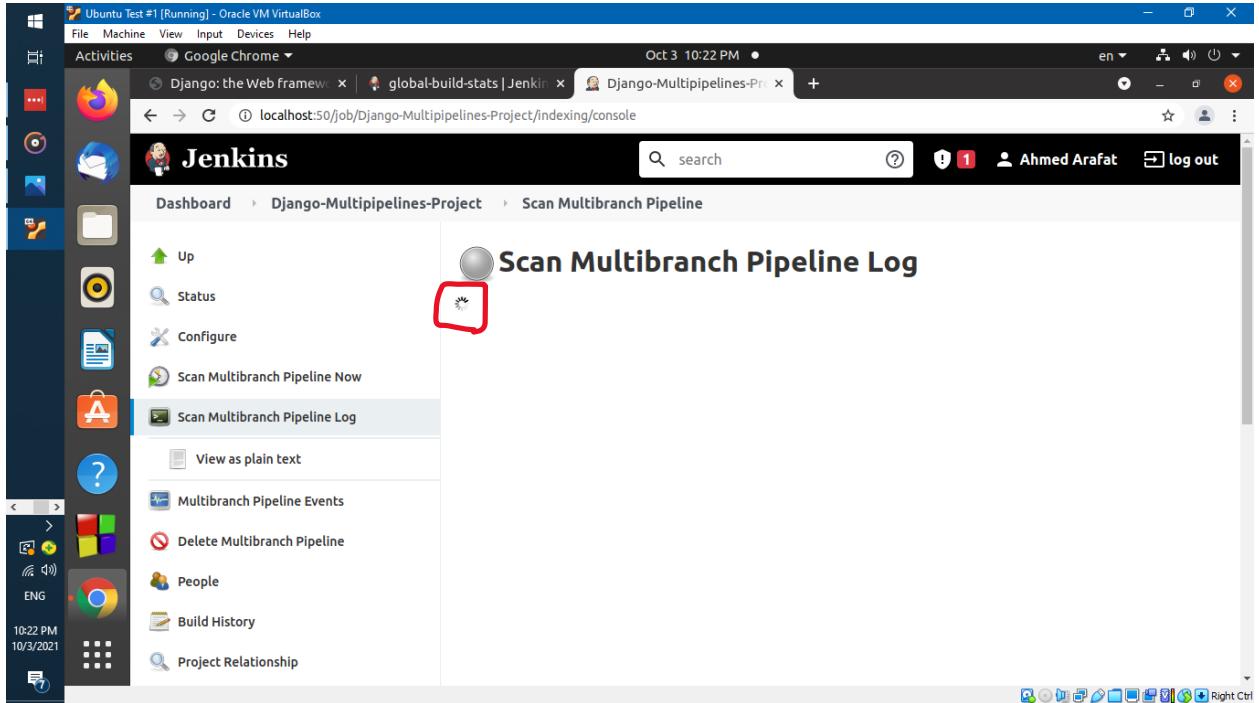
After saving and then creating “Project” slave node  
Node Is created & online successfully 😊



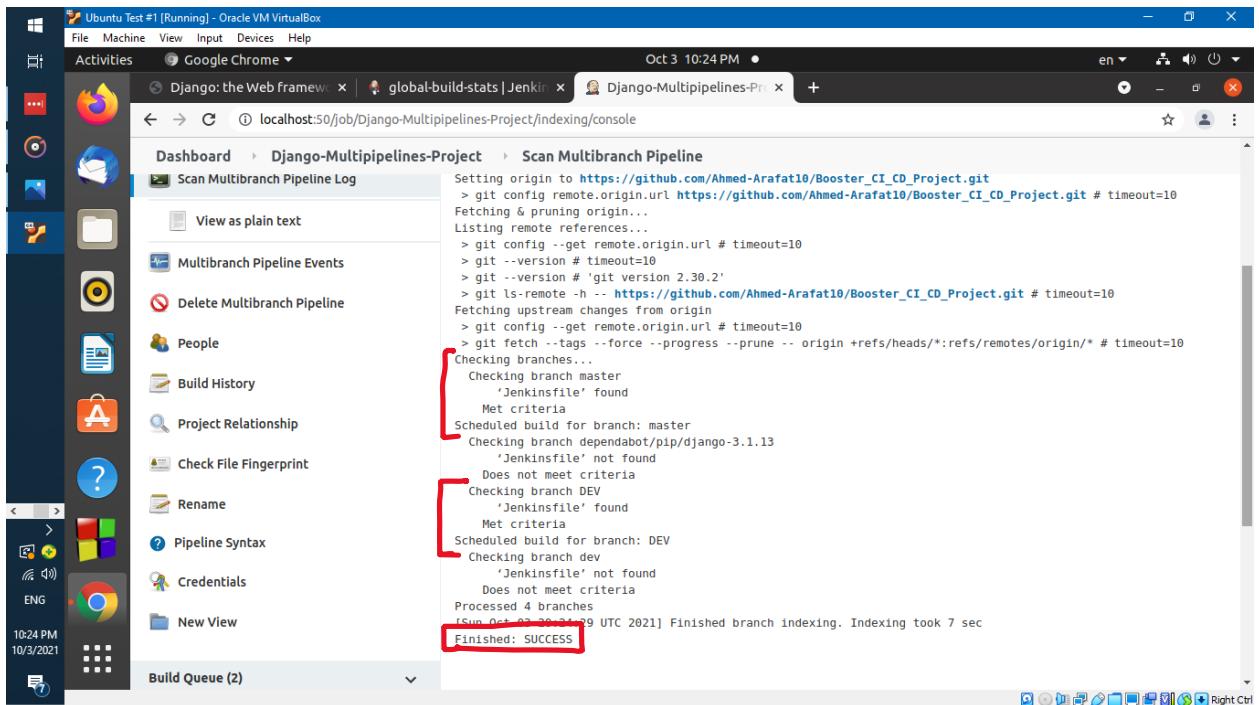
Now go to {New item menu} then create a Multibranch Pipeline called “Django-Multipipelines-Project”  
With git option of Project’s GitHub repo link



Then hit save and wait until it finishes loading



Multibranch pipeline automatically created Two pipelines from branches Master/DEV as both have {Jenkinsfile} in them while neglecting other branches



Here are Branches in Project's repo

The screenshot shows a GitHub repository page for 'Ahmed-Arafat10 / Booster\_CI\_CD\_Project'. The 'Code' tab is selected, displaying the 'master' branch. A modal window titled 'Switch branches/tags' is open, showing a list of branches: master (selected), DEV, dependabot/pip/django-3.1.13, and dev. The master branch has 34 commits. Below the modal, there is a code editor for 'README.md'. On the right side of the page, there are sections for 'About', 'Releases', 'Packages', and 'Languages'.

Master/DEV branches each has a {dockerfile} that will create a modified image from ubuntu image base  
This modified image will install Python3 & Pip3, install required packages for django, make migration for DataBase, apply the migrations then start the server

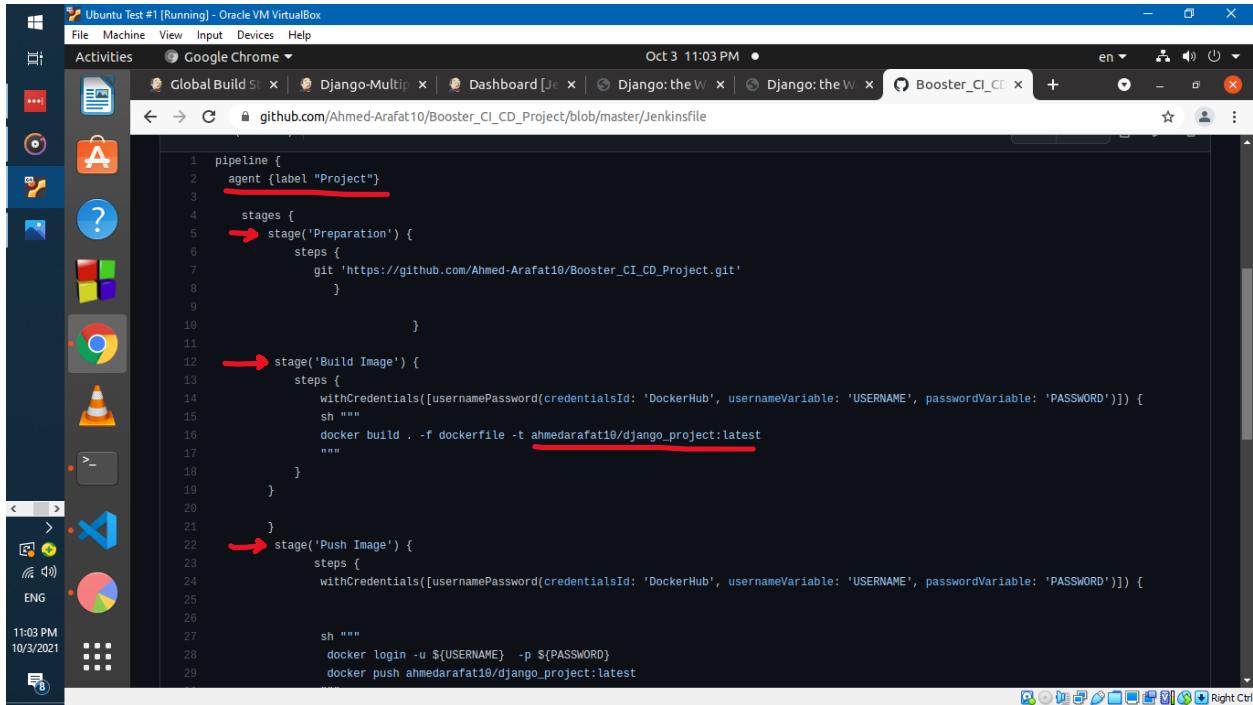
Note: I also can install Docker CLI in image if I want

The screenshot shows a GitHub repository page for 'Ahmed-Arafat10 / Update dockerfile'. The code editor displays a Dockerfile with the following content:

```
FROM ubuntu
COPY simpleApp /root/Jenkins/simpleApp
COPY Meta /root/Jenkins/
WORKDIR /root/Jenkins/
RUN ls
RUN apt-get -y update
RUN apt-get install python3 -y
RUN apt-get -y install python3-pip
WORKDIR /root/Jenkins/
RUN pwd && ls
RUN pip install -r requirements.txt
# RUN ls /root/Jenkins/Django
RUN python3.8 manage.py makemigrations
RUN python3.8 manage.py migrate
CMD python3.8 manage.py runserver 0.0.0.0:8000
```

This is sample of {Jenkinsfile} that is found in both master/DEV branches

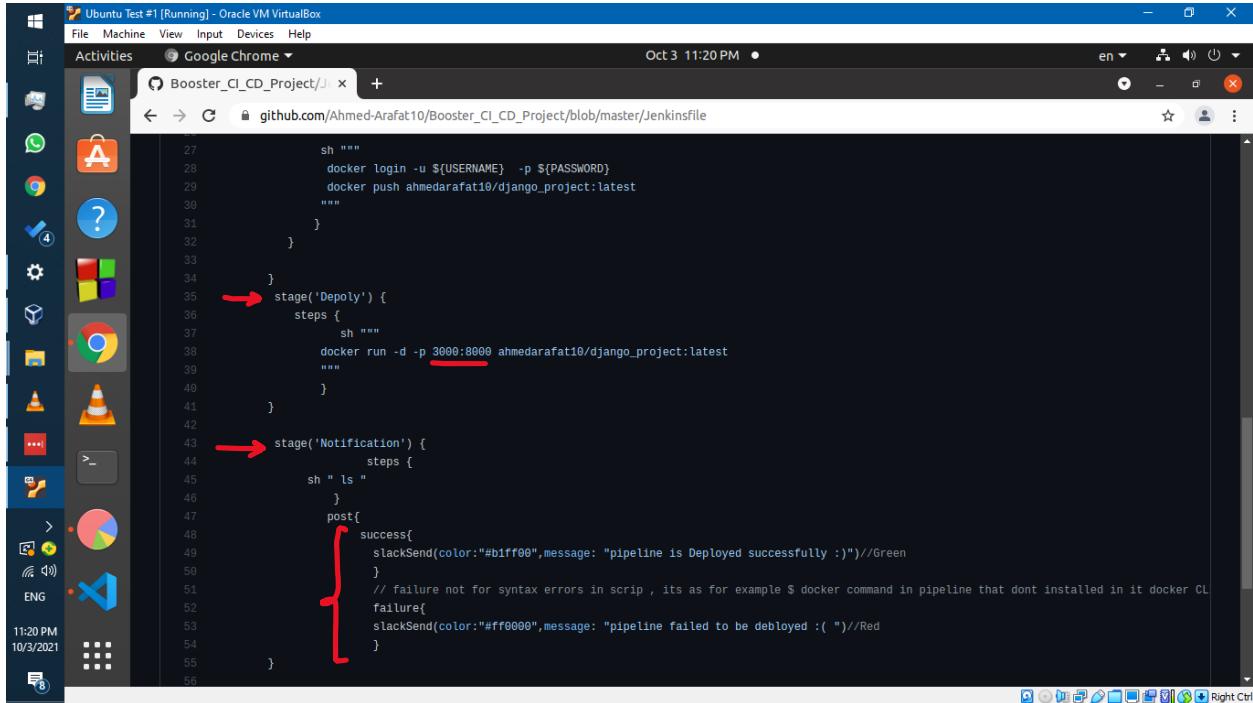
{Just Check Github repo 😊 }



```
1 pipeline {
2     agent {label "Project"}
3
4     stages {
5         stage('Preparation') {
6             steps {
7                 git 'https://github.com/Ahmed-Arafat10/Booster_CI_CD_Project.git'
8             }
9         }
10
11         stage('Build Image') {
12             steps {
13                 withCredentials([usernamePassword(credentialsId: 'DockerHub', usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD')]) {
14                     sh """
15                     docker build . -f dockerfile -t ahmedarafat10/django_project:latest
16                 """
17             }
18         }
19
20         stage('Push Image') {
21             steps {
22                 withCredentials([usernamePassword(credentialsId: 'DockerHub', usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD')]) {
23                     sh """
24                     docker login -u ${USERNAME} -p ${PASSWORD}
25                     docker push ahmedarafat10/django_project:latest
26                 """
27             }
28         }
29     }
30 }
```

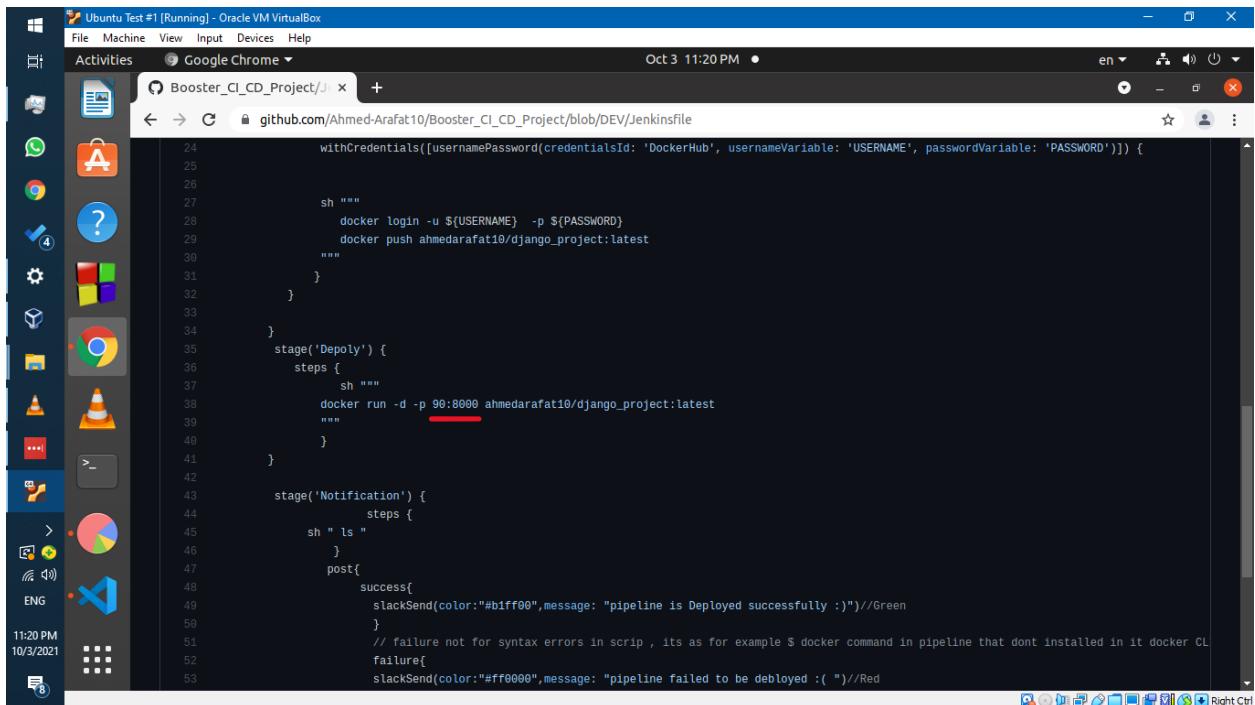
An Important Note: as I'm going to build *TWO* pipelines one for master branch and the other for DEV branch, each pipeline must have a different port referring to default Django port {8000}

So, for master branch's pipeline will have {3000} port



```
27         sh """
28         docker login -u ${USERNAME} -p ${PASSWORD}
29         docker push ahmedarafat10/django_project:latest
30     """
31
32 }
33
34
35 stage('Deploy') {
36     steps {
37         sh """
38         docker run -d -p 3000:8000 ahmedarafat10/django_project:latest
39     """
40 }
41
42
43 stage('Notification') {
44     steps {
45         sh "ls"
46     }
47     post{
48         success{
49             slackSend(color:"#bfff00",message: "pipeline is Deployed successfully :)//Green"
50         }
51         // failure not for syntax errors in script , its as for example $ docker command in pipeline that dont installed in it docker CL
52         failure{
53             slackSend(color:"#ff0000",message: "pipeline failed to be deployed :( ")//Red
54         }
55     }
56 }
```

While DEV branch's pipeline will have {90} port



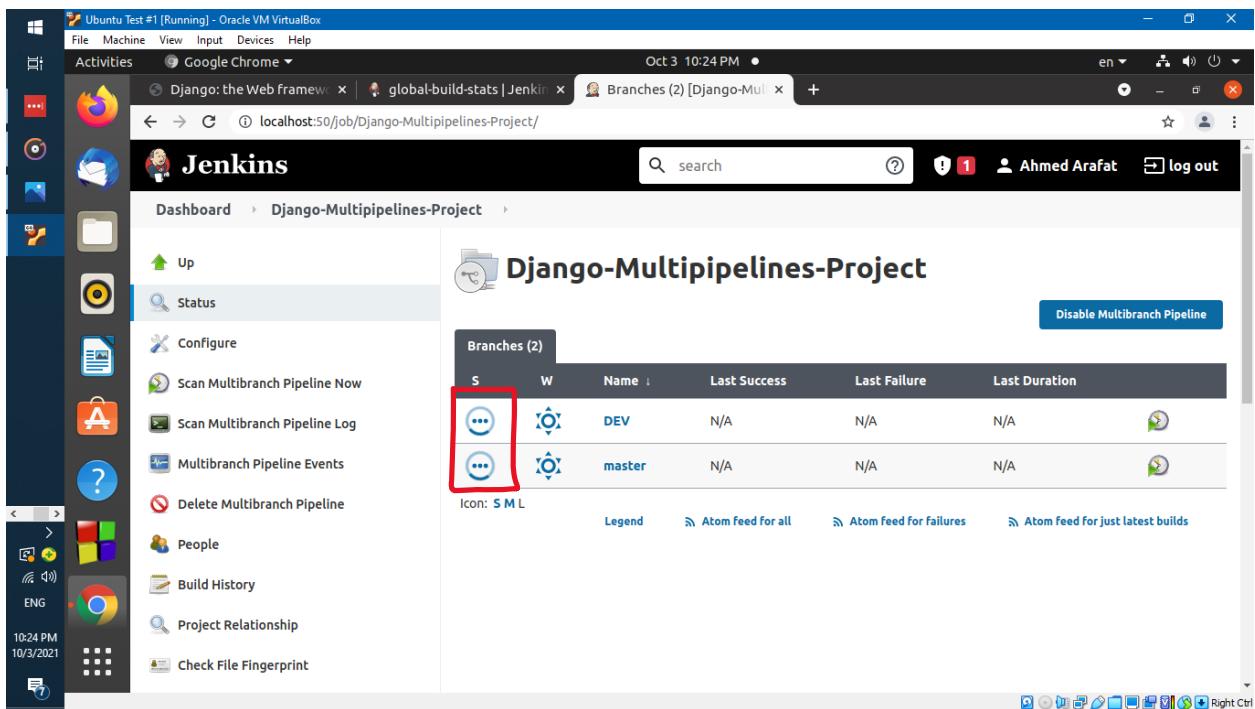
A screenshot of a Linux desktop environment (Ubuntu) running in Oracle VM VirtualBox. The desktop has a dark theme with a vertical dock on the left containing icons for various applications like a terminal, file manager, and system tools. A Google Chrome window is open, displaying a Jenkinsfile for a project named 'Booster\_CI\_CD\_Project'. The Jenkinsfile contains Jenkins Groovy script for a 'Deploy' stage that runs a Docker command with port mapping (90:8999). The code also includes a 'Notification' stage that sends a Slack message upon success or failure. The status bar at the bottom shows the date and time as Oct 3 11:20 PM.

```
withCredentials([usernamePassword(credentialsId: 'DockerHub', usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD')]) {
    sh """
        docker login -u ${USERNAME} -p ${PASSWORD}
        docker push ahmedarafat10/django_project:latest
    """
}

stage('Deploy') {
    steps {
        sh """
            docker run -d -p 90:8999 ahmedarafat10/django_project:latest
        """
    }
}

stage('Notification') {
    steps {
        sh "ls"
    }
    post{
        success{
            slackSend(color:"#bfff00",message: "pipeline is Deployed successfully :)"//Green
        }
        failure{
            slackSend(color:"#ff0000",message: "pipeline failed to be deployed :( ")//Red
        }
    }
}
```

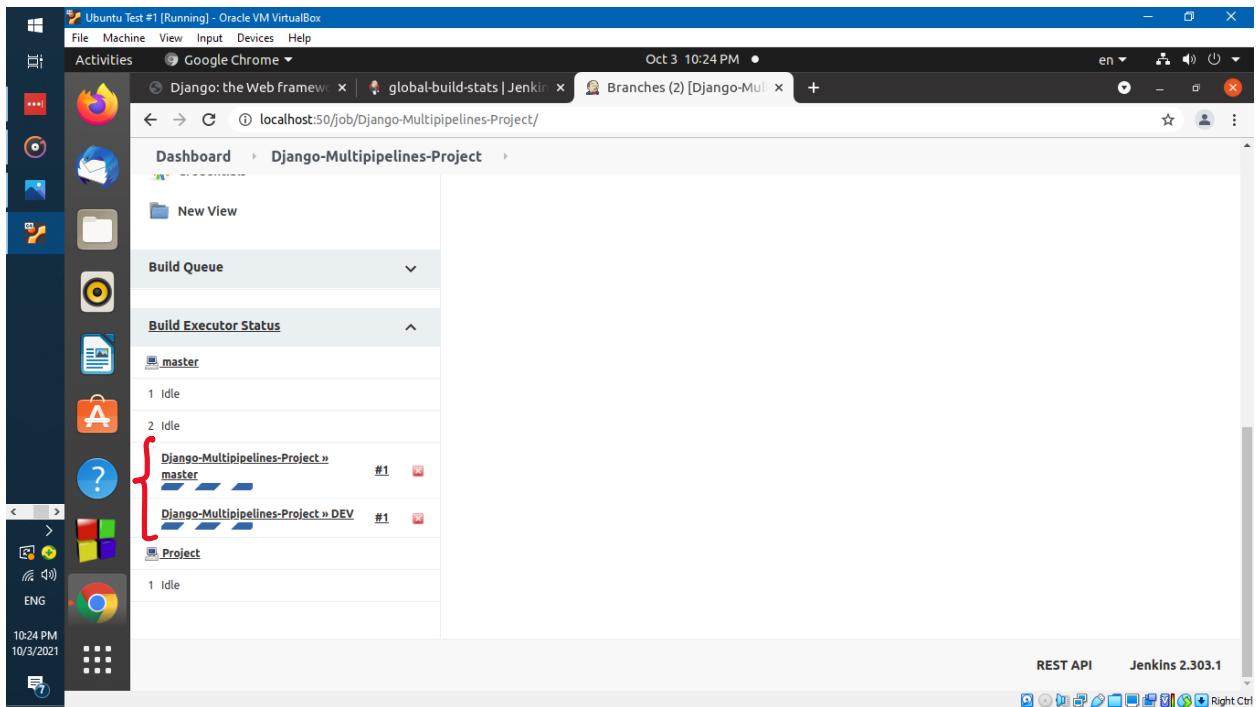
Now both pipelines are being built



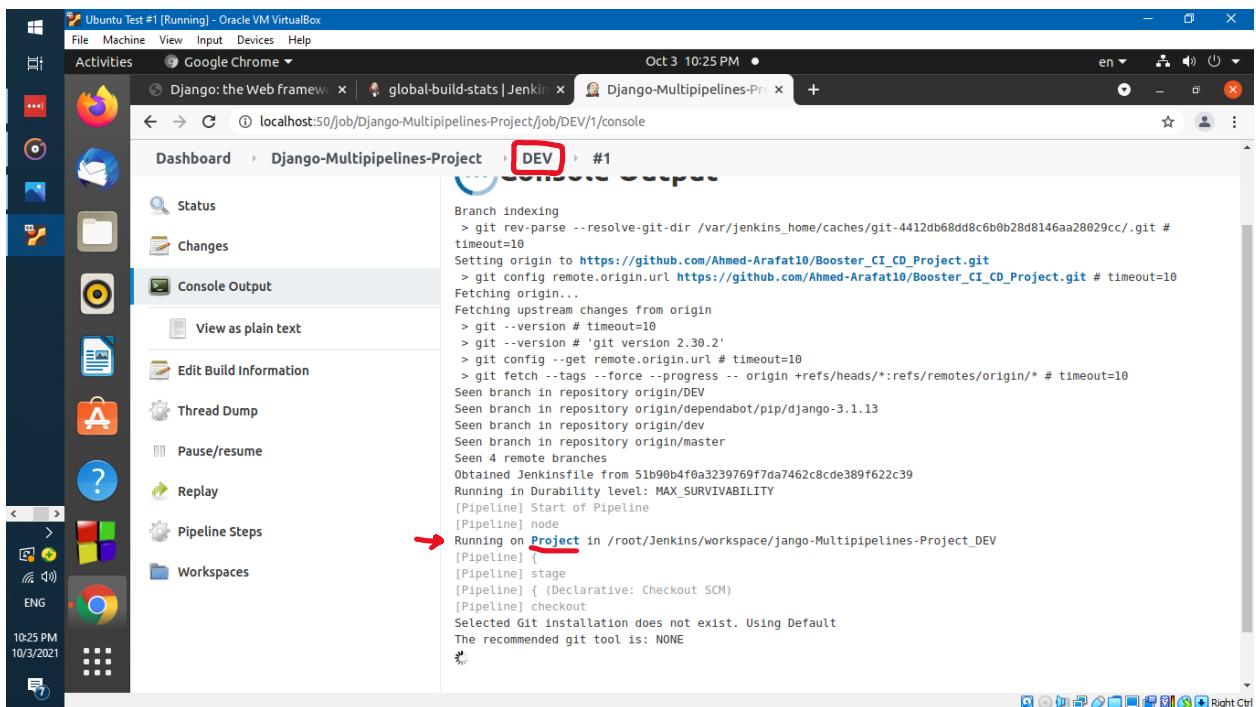
A screenshot of the Jenkins dashboard for the 'Django-Multipipelines-Project'. The left sidebar shows navigation options like Dashboard, Status, Configure, and Multibranch Pipeline Events. The main content area displays a table titled 'Branches (2)' for the 'Django-Multipipelines-Project'. The table lists two branches: 'DEV' and 'master'. Both branches show 'N/A' for Last Success, Last Failure, and Last Duration. The rows for both branches are highlighted with a red box. At the bottom of the table, there are links for Atom feed for all, Atom feed for failures, and Atom feed for just latest builds. The status bar at the bottom shows the date and time as Oct 3 10:24 PM.

S	W	Name	Last Success	Last Failure	Last Duration
		DEV	N/A	N/A	N/A
		master	N/A	N/A	N/A

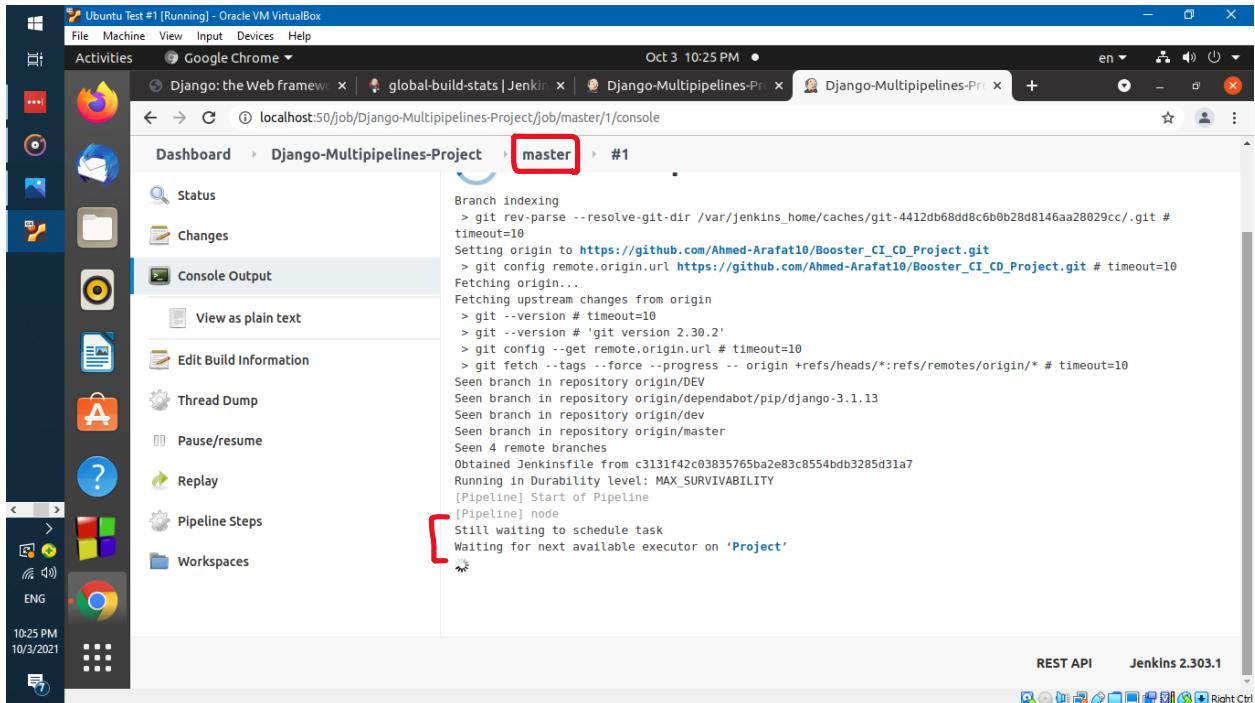
## Still Loading ...



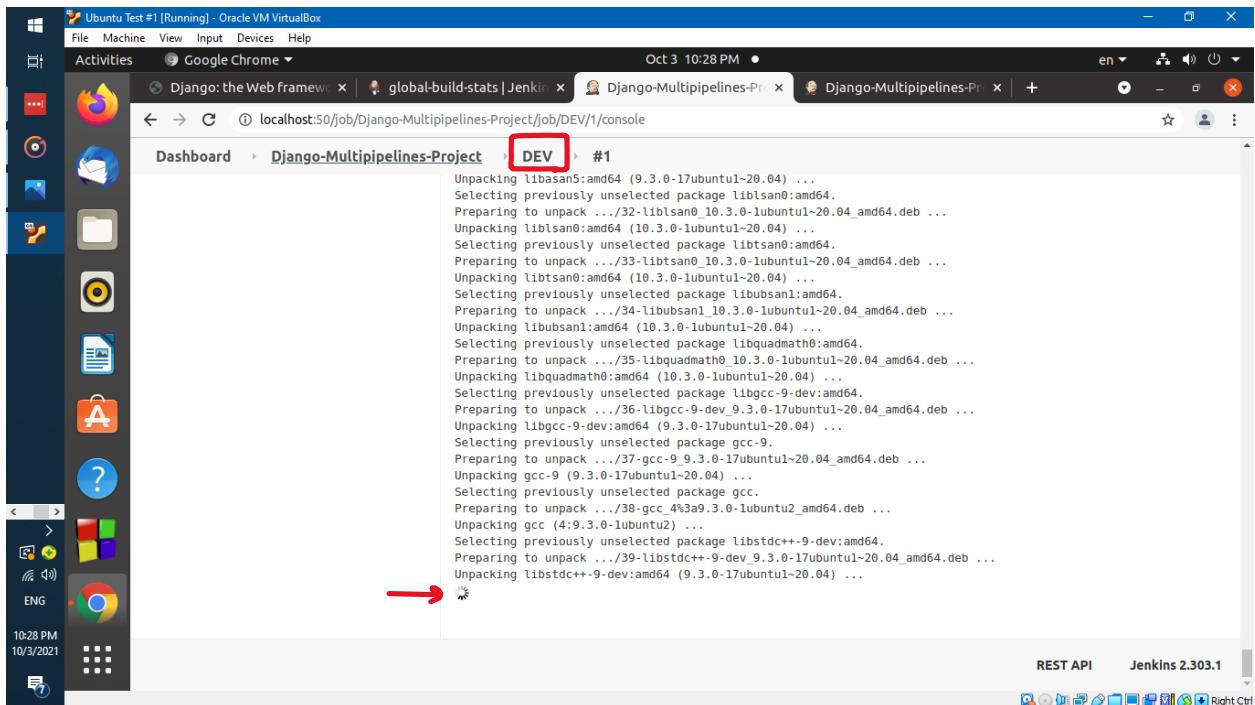
In {Console output Menu}, Jenkins Server is deploying DEV branch's pipeline first on “Project” node



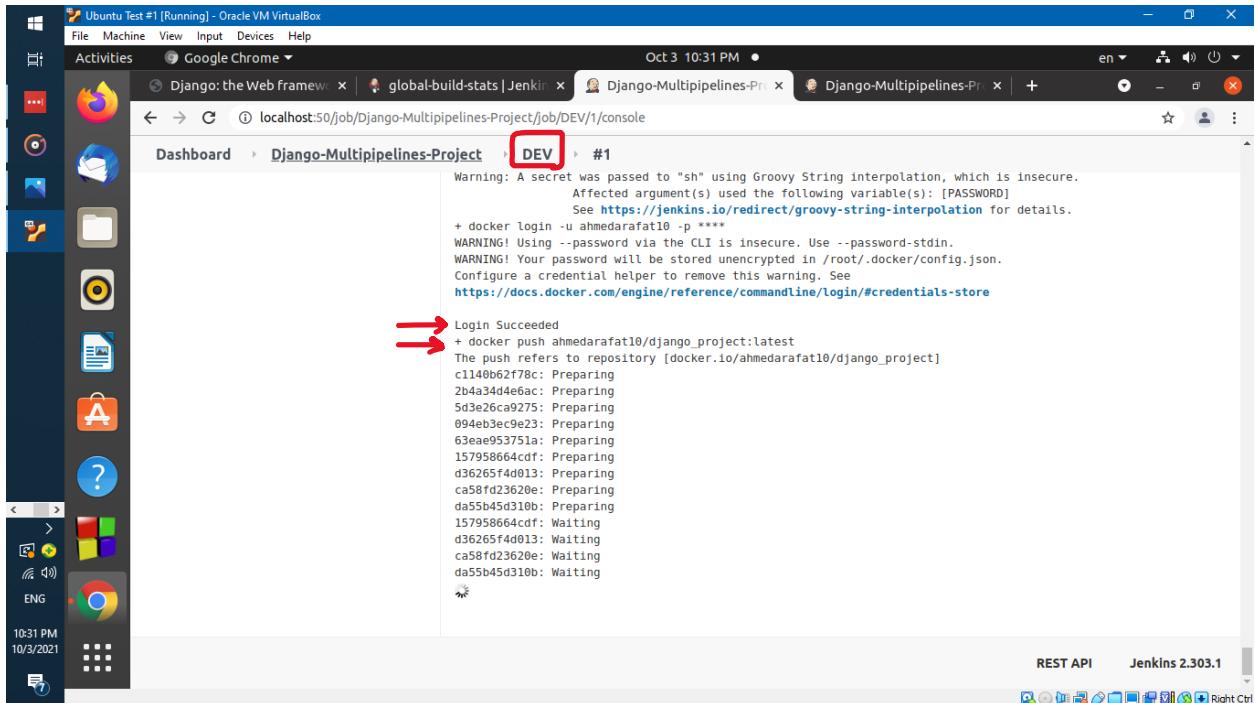
While DEV branch's pipeline is in queue waiting for the other pipeline to finish deploying



DEV branch's pipeline is still deploying {Executing Jenkinsfile's stages > steps}



Now Jenkins Server is pushing created Django Image into my Docker Hub Account

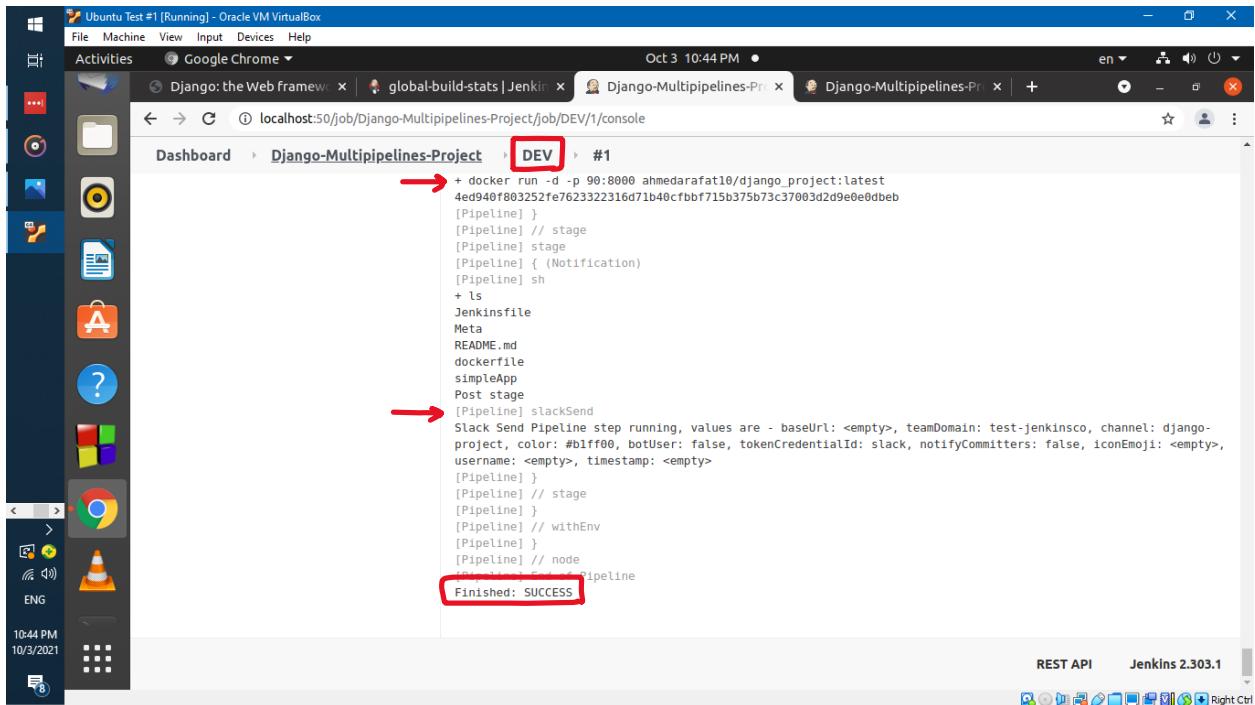


```
Warning: A secret was passed to "sh" using Groovy String interpolation, which is insecure.  
Affected argument(s) used the following variable(s): [PASSWORD]  
See https://jenkins.io/redirect/groovy-string-interpolation for details.  
+ docker login -u ahmedarafat10 -p ****  
WARNING! Using --password via the CLI is insecure. Use --password-stdin.  
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.  
Configure a credential helper to remove this warning. See  
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
```

Login Succeeded

```
+ docker push ahmedarafat10/django_project:latest  
The push refers to repository [docker.io/ahmedarafat10/django_project]  
c1140b62f78c: Preparing  
2b4a34d4e6ac: Preparing  
5d3e26ca9275: Preparing  
094eb3e9e23: Preparing  
63ea953751a: Preparing  
157958664cdf: Preparing  
d36265f4d013: Preparing  
ca58fd23620e: Preparing  
da55b45d310b: Preparing  
157958664cdf: Waiting  
d36265f4d013: Waiting  
ca58fd23620e: Waiting  
da55b45d310b: Waiting
```

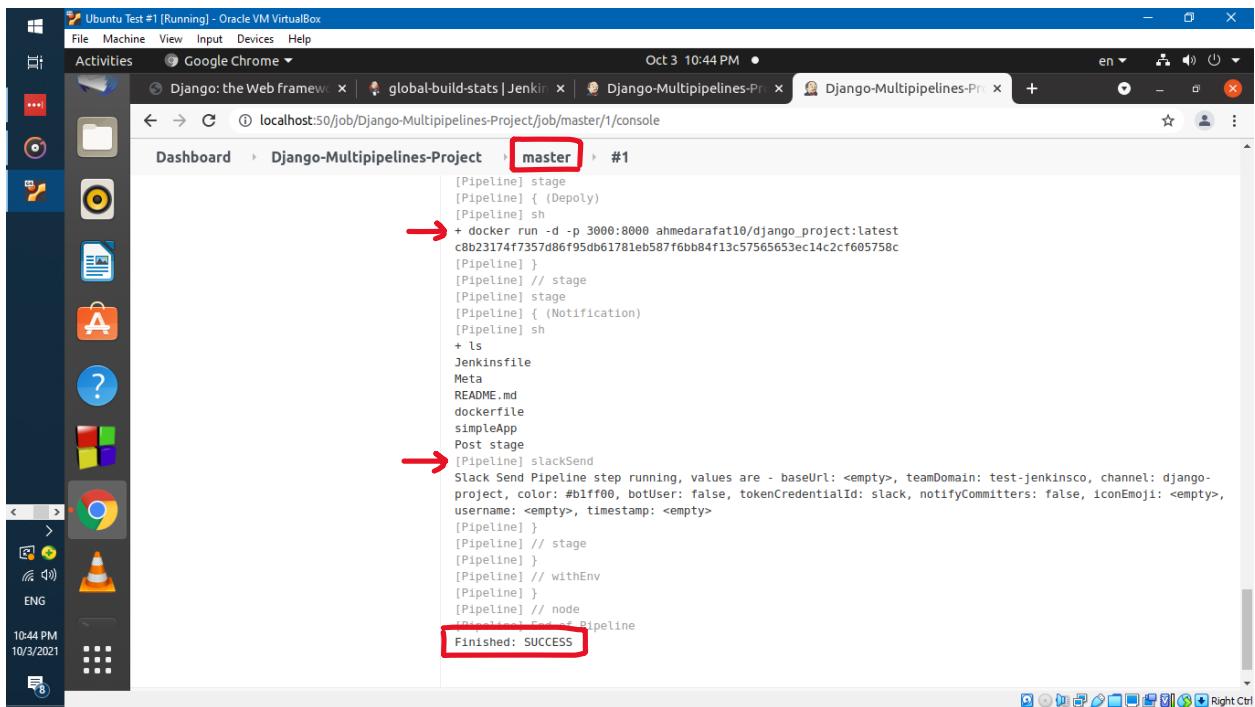
Now DEV branch's pipeline is deployed successfully 😊



```
+ docker run -d -p 90:8000 ahmedarafat10/django_project:latest  
4ed940f803252fe7623322316d71b40cfbbf715b375b73c37003d2d9e0e0db  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] stage  
[Pipeline] { (Notification)  
[Pipeline] sh  
+ ls  
Jenkinsfile  
Meta  
README.md  
dockerfile  
simpleApp  
Post stage  
[Pipeline] slackSend  
Slack Send Pipeline step running, values are - baseUrl: <empty>, teamDomain: test-jenkinsco, channel: django-project, color: #biff00, botUser: false, tokenCredentialId: slack, notifyCommitters: false, iconEmoji: <empty>, username: <empty>, timestamp: <empty>  
[Pipeline] }  
[Pipeline] // stage  
[Pipeline] }  
[Pipeline] // withEnv  
[Pipeline] }  
[Pipeline] // node  
[Pipeline] }  
[Pipeline] // Pipeline
```

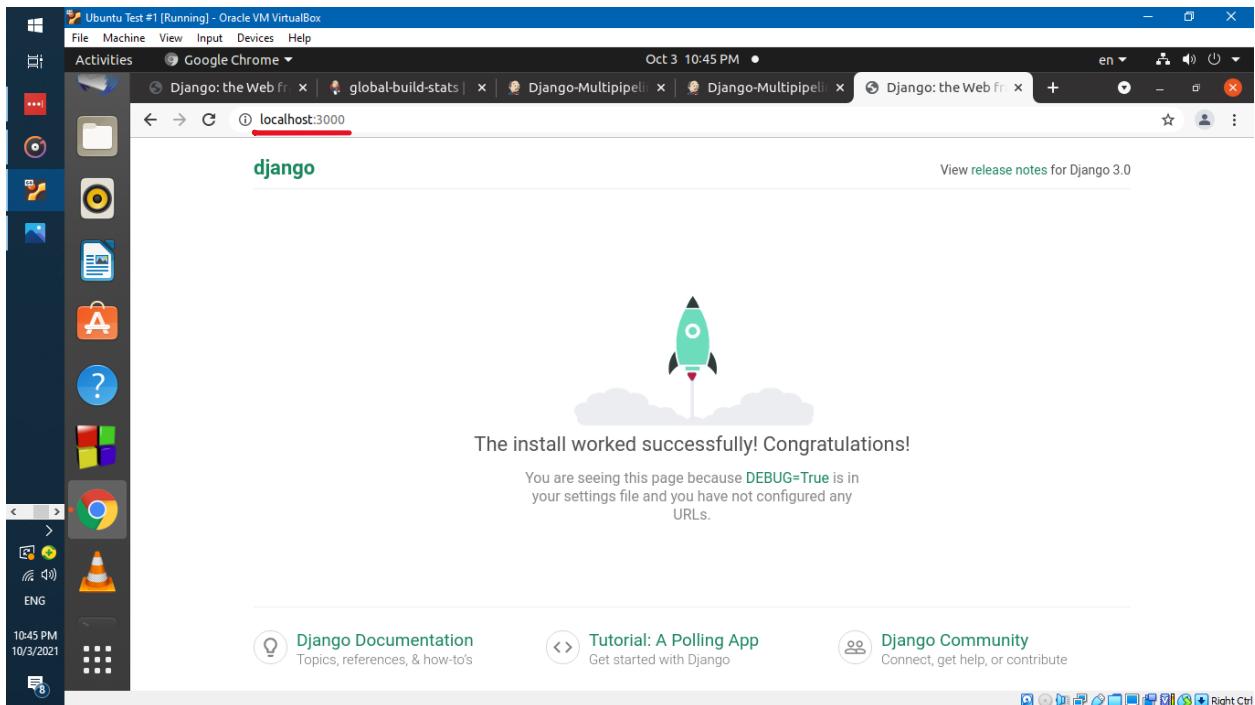
Finished: SUCCESS

After finishing DEV branch's pipeline, Jenkins Server deploys Master branch's pipeline.  
Now Master branch's pipeline is also deployed successfully 😊

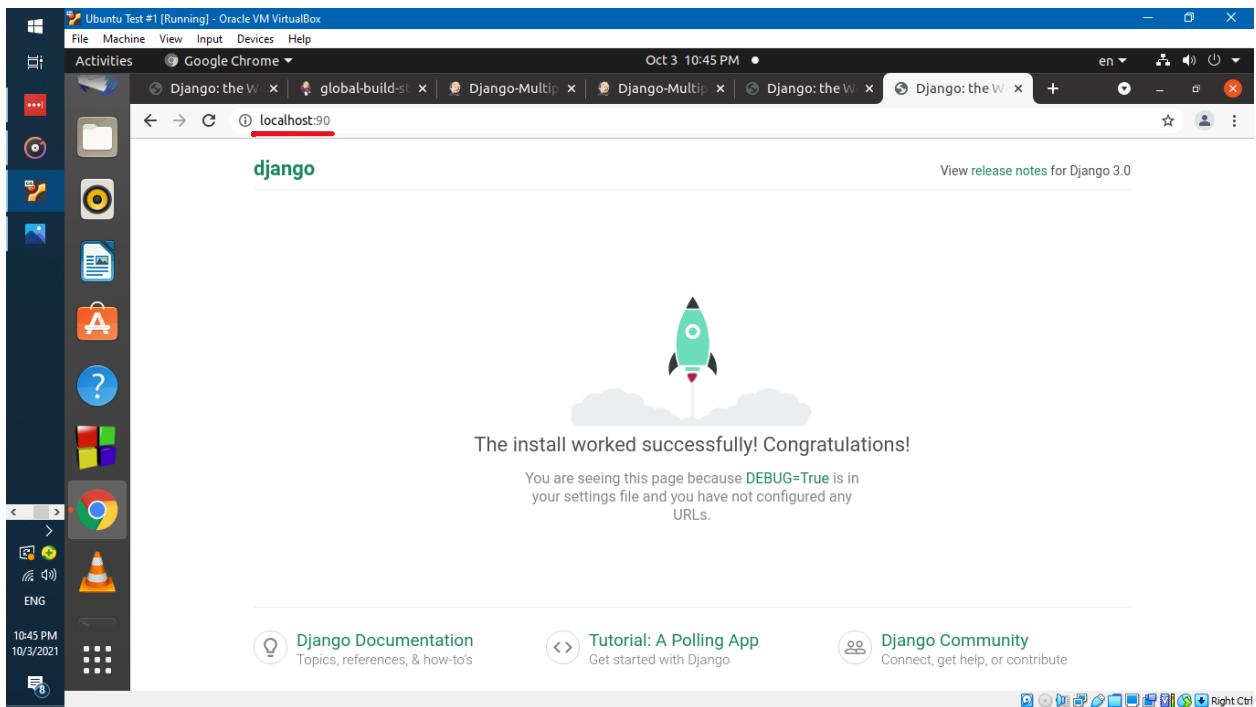


```
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] sh
+ docker run -d -p 3000:8000 ahmedarafat10/django_project:latest
c8b23174f7357d86f95db61781eb587f6bb84f13c5756563ec14c2cf605758c
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Notification)
[Pipeline] sh
+ ls
Jenkinsfile
Meta
README.md
dockerfile
simpleApp
Post stage
[Pipeline] slackSend
Slack Send Pipeline step running, values are - baseUrl: <empty>, teamDomain: test-jenkinsco, channel: django-project, color: #bfff00, botUser: false, tokenCredentialId: slack, notifyCommitters: false, iconEmoji: <empty>, username: <empty>, timestamp: <empty>
[Pipeline]
[Pipeline] // stage
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // node
[Pipeline] // Post Pipeline
[Pipeline] Finished: SUCCESS
```

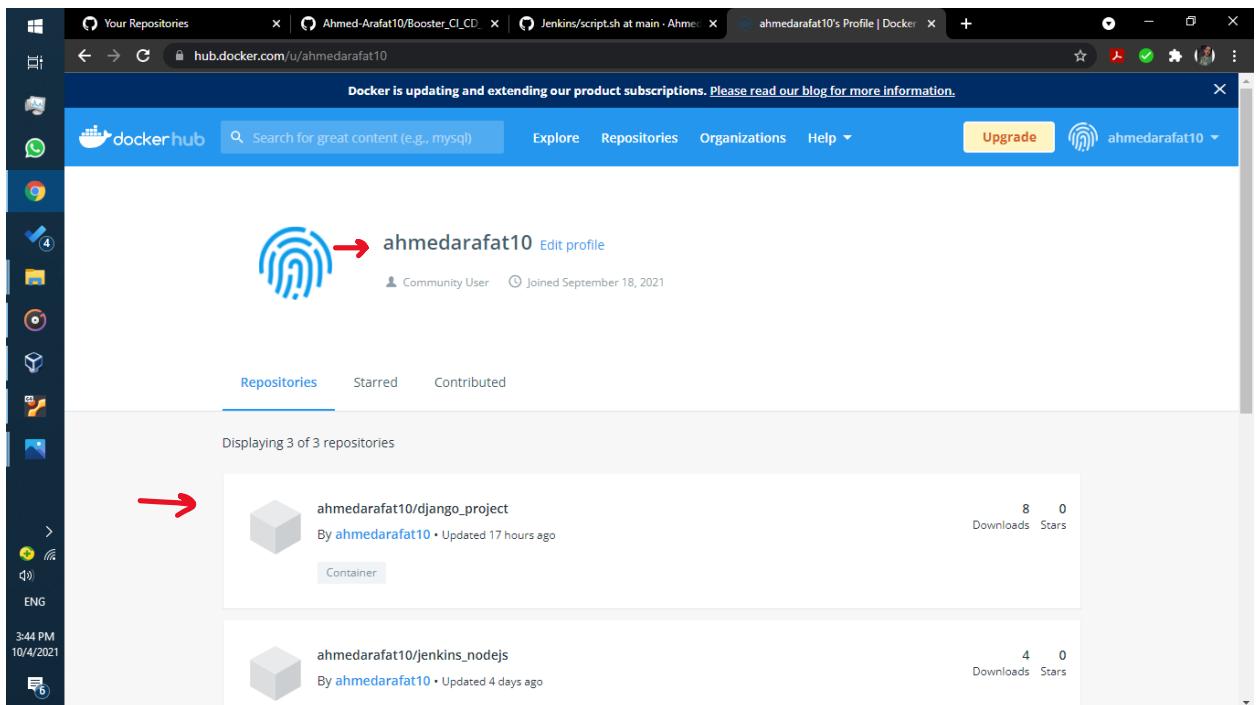
Checking {localhost:3000} -> For Master branch's pipeline  
It Works 😊



Checking {localhost:90} -> For DEV branch's pipeline  
It Works 😊



Created Django Image is pushed into my Docker Hub's Account



Both are working perfectly 😊

The screenshot shows the Jenkins Multibranch Pipeline interface. On the left, there's a sidebar with various Jenkins-related icons and links like 'Scan Multibranch Pipeline Now', 'Configure', and 'Delete Multibranch Pipeline'. The main area is titled 'Branches (2)' and lists two branches: 'DEV' and 'master'. Both branches have green checkmarks next to them, indicating they are successful. The table columns include Status (S), Workstation (W), Name, Last Success, Last Failure, and Last Duration. Below the table, there are links for 'Atom feed for all', 'Atom feed for failures', and 'Atom feed for just latest builds'. The status bar at the bottom shows the date and time as Oct 3 10:47 PM.

A message has been sent to my slack {Django-project} channel when both pipelines are successfully deployed

The screenshot shows the Slack interface for the '# django-project' channel. On the left, there's a sidebar with team members and channels. The channel list shows '# django-project' highlighted with a red arrow. The channel history shows two messages from the 'jenkins' bot: one at 7:30 PM and another at 10:44 PM, both stating 'pipeline is Deployed successfully :)'. These messages are also highlighted with a red box. The status bar at the bottom shows the date and time as Oct 3 10:49 PM.

Now I have to install a Plugin called {global-build-stats} to show statistics about builds

The screenshot shows a Windows desktop environment with a taskbar at the bottom. The taskbar icons include File Explorer, Task View, Start, Task Manager, and others. A browser window titled "Ubuntu Test #1 [Running] - Oracle VM VirtualBox" is open, showing the Jenkins global-build-stats plugin documentation. The URL in the address bar is [plugins.jenkins.io/global-build-stats/#documentation](https://plugins.jenkins.io/global-build-stats/#documentation). The page content includes the plugin's name, version (1.5), release date (4 years ago), required Jenkins version (1.609.1), and install count (11,044). It also features a chart showing the number of installations over time and links to GitHub, Open issues, Report an issue, and Javadoc.

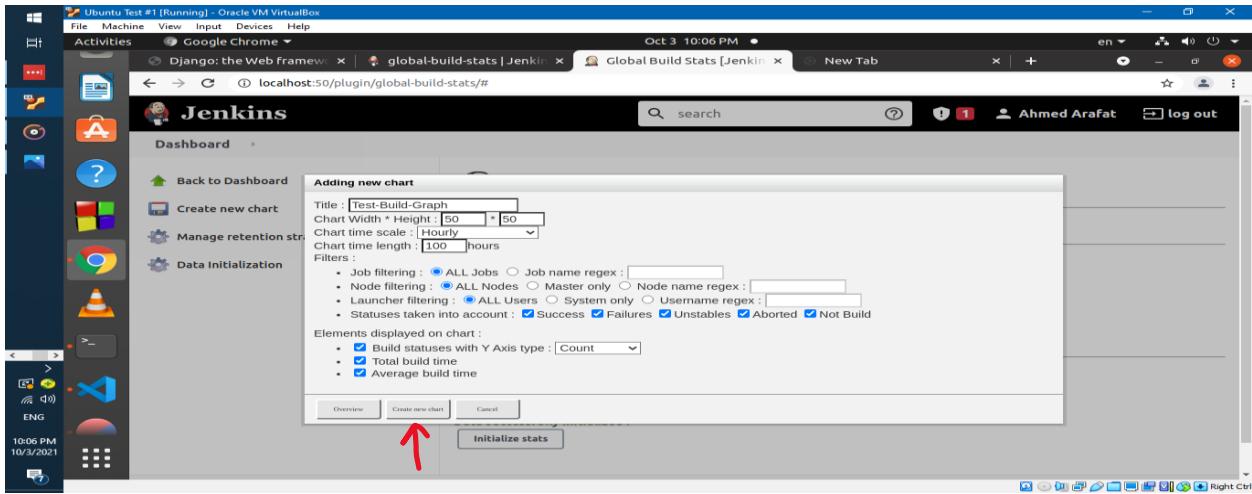
I will install it using terminal, but I will have first to enter Jenkins's docker container by running following command:

```
$ sudo docker exec -it <JenkinsContainer_ID> bash
```

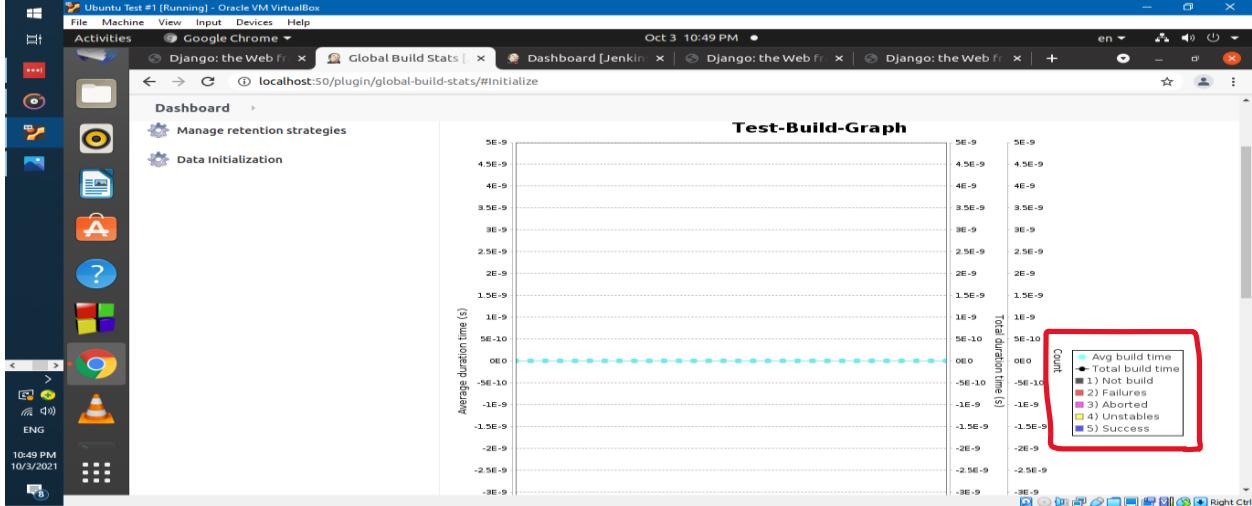
Then running the command shown in the picture inside it

The screenshot shows a Windows desktop environment with a taskbar at the bottom. The taskbar icons include File Explorer, Task View, Start, Task Manager, and others. A browser window titled "Ubuntu Test #1 [Running] - Oracle VM VirtualBox" is open, showing the Jenkins global-build-stats plugin releases page. The URL in the address bar is [plugins.jenkins.io/global-build-stats/#releases](https://plugins.jenkins.io/global-build-stats/#releases). The page lists three releases: 1.5 (4 years ago), 1.4 (5 years ago), and 1.3 (10 years ago). Each release entry includes a command to install via Jenkins' plugin CLI: `jenkins-plugin-cli --plugins global-build-stats:1.5`, `jenkins-plugin-cli --plugins global-build-stats:1.4`, and `jenkins-plugin-cli --plugins global-build-stats:1.3`. SHA-1 and SHA-256 checksums are also provided for each release. The page layout is identical to the documentation page, featuring the same sidebar with Jenkins icons and links to GitHub, Open issues, Report an issue, and Javadoc.

## Configuring new installed plugin in Jenkins Server



Now a graph will show all statistics about builds in Jenkins Server



Now checking all running container in host terminal which are :

- 1) Jenkins server Docker Container
- 2) Ubuntu Docker Container
- 3) Django Docker Container for Master branch
- 4) Django Docker Container for DEV branch

The screenshot shows a terminal window with the command "sudo docker container ps -a" run by user "arafat" at 11:16 PM on Oct 3, 2021. The output lists four containers:

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS	PORTS
c8b23174f735	ahmedarafat0/django_project:latest	"bin/sh -c 'python3..."	14 minutes ago	Up 14 minutes	0.0.0.0:3000->8000/tcp	
4ed940f80325	ahmedarafat0/django_project:latest	"bin/sh -c 'python3..."	16 minutes ago	Up 15 minutes	0.0.0.0:90->8000/tcp	
ef031c81a7d4	ubuntu_docker_cli_all:v1	"bin/sh -c 'cd /root...'"	5 hours ago	Up 5 hours	0.0.0.0:8080->8080/tcp	
32ede207bf9c	jenkins_with_docker:v1	"sbin/tini -- /usr/..."	23 hours ago	Up 57 minutes	50000/tcp, 0.0.0.0:50->8080/tcp, ::1:8080->8080/tcp, affectionate_dijkstra	

That's It 😊