

Capstone Projects 1

Problem statement :

You have been Hired Sr. DevOps Engineer in Abode Software. They want to implement DevOps Lifecycle in their company. You have been asked to implement this lifecycle as fast as possible. Abode Software is a product-based company, their product is available on this GitHub link.

<https://github.com/hshar/website.git>

Following are the specifications of the lifecycle:

1. Install the necessary software on the machines using a configuration management tool.
2. Git Workflow has to be implemented
- 3.Code Build should automatically be triggered once commit is made to master branch or develop branch.

If commit is made to master branch, test and push to prod

If commit is made to develop branch, just test the product, do not push to prod

- 4.The Code should be containerized with the help of a Dockerfile. The Dockerfile should be built every time there is a push to Git-Hub. Use the following pre-built container for your application:

hshar/webapp

The code should reside in '/var/www/html'

- 5.The above tasks should be defined in a Jenkins Pipeline, with the following jobs:

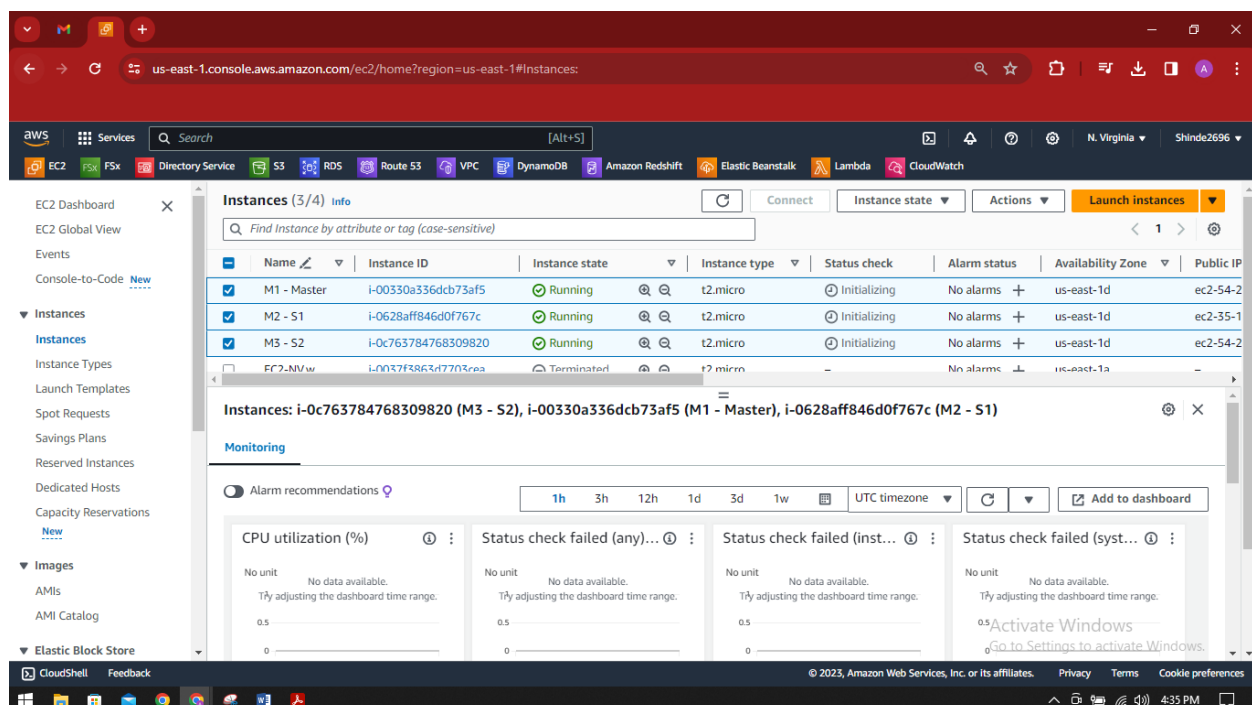
Job1 : build

Job2: test

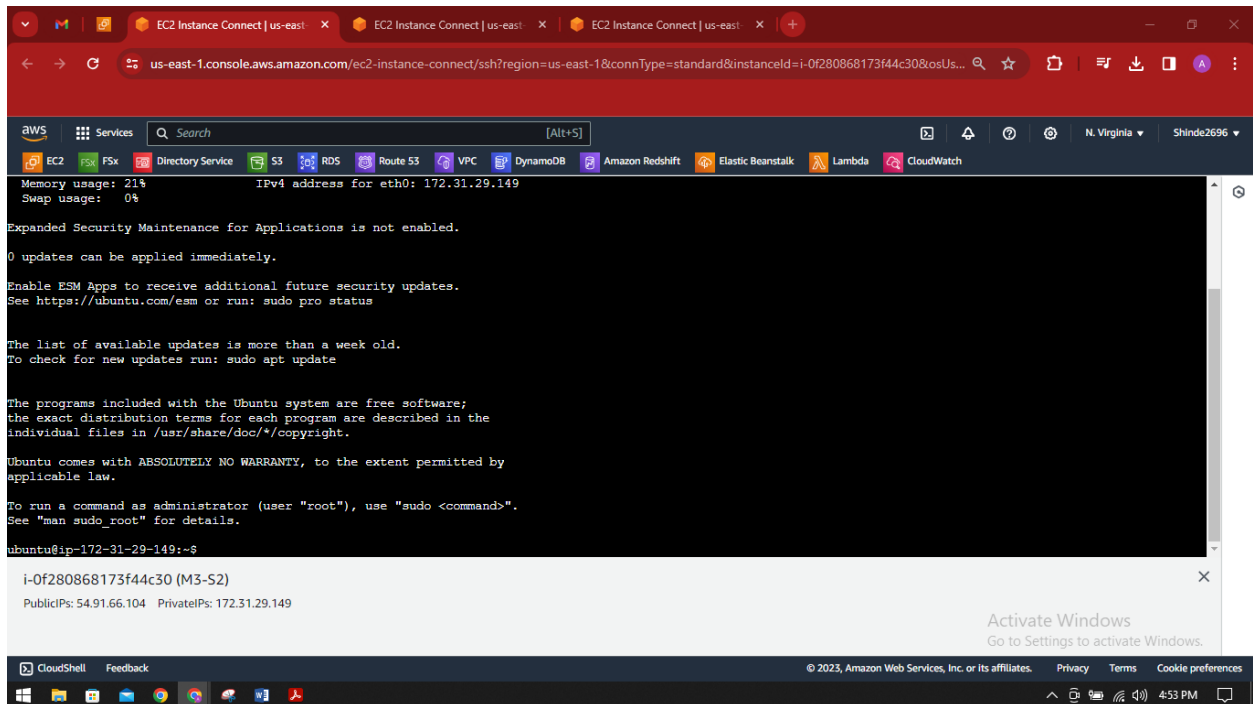
Job3 : prod

Solution :

Creating 3 instances Machine 1 will be Master and the other 2 will be sleeves named M1 – Master and M2 – S1 and M3 – S2 using Ubuntu OS



Connected to All 3 instances



```
us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?region=us-east-1&connType=standard&instanceId=i-0f280868173f44c30&osUs...

AWS Services Search [Alt+S]
EC2 FSx Directory Service S3 RDS Route 53 VPC DynamoDB Amazon Redshift Elastic Beanstalk Lambda CloudWatch

Memory usage: 21% IPv4 address for eth0: 172.31.29.149
Swap usage: 0%

Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

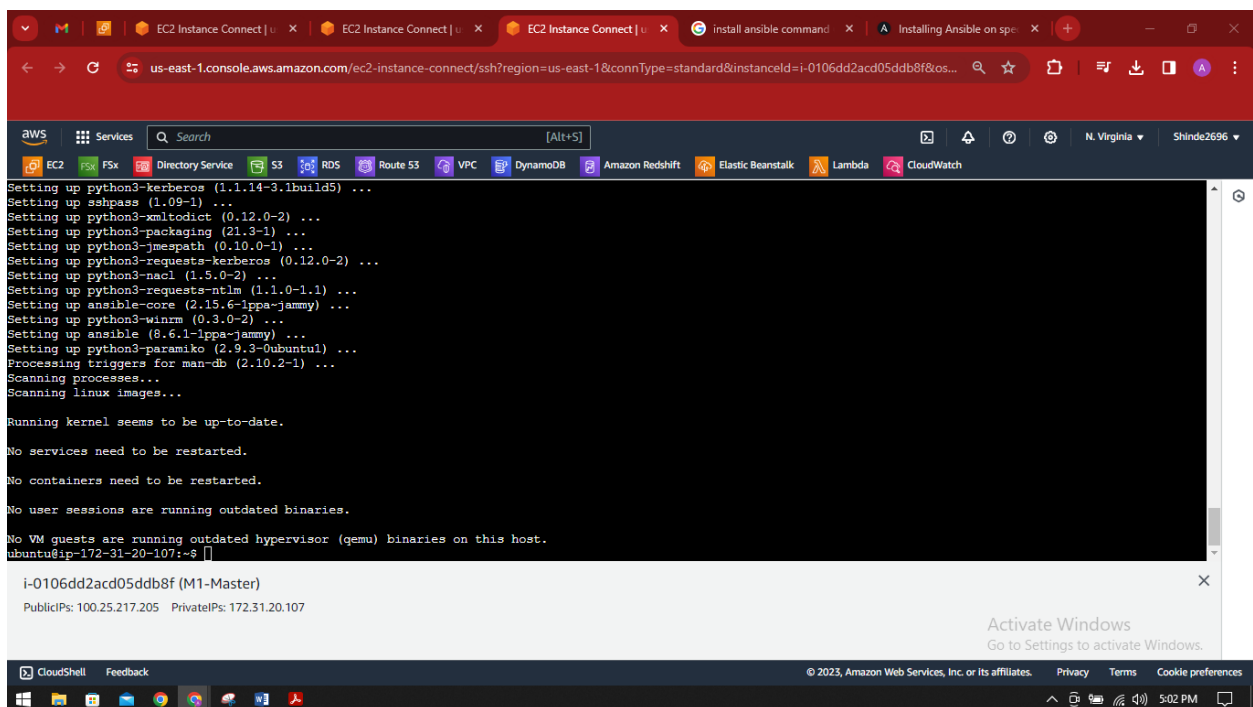
ubuntu@ip-172-31-29-149:~$

i-0f280868173f44c30 (M3-S2)
PublicIPs: 54.91.66.104 PrivateIPs: 172.31.29.149

Activate Windows
Go to Settings to activate Windows.
```

Installed Ansible on Master machine by running below commands

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository --yes --update ppa:ansible/ansible
sudo apt install ansible
```



```
us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?region=us-east-1&connType=standard&instanceId=i-0106dd2acd05ddb8f&os...

AWS Services Search [Alt+S]
EC2 FSx Directory Service S3 RDS Route 53 VPC DynamoDB Amazon Redshift Elastic Beanstalk Lambda CloudWatch

Setting up python3-kerberos (1.1.14-3.1build5) ...
Setting up sshpass (1.09-1) ...
Setting up python3-xlrd (2.0.1-2) ...
Setting up python3-packaging (21.3-1) ...
Setting up python3-jmespath (0.10.0-1) ...
Setting up python3-requests-kerberos (0.12.0-2) ...
Setting up python3-nacl (1.5.0-2) ...
Setting up python3-requests-ntlm (1.1.0-1.1) ...
Setting up ansible-core (2.15.6-lppa-jammy) ...
Setting up python3-winscp (0.3.0-2) ...
Setting up ansible (8.6.1-lppa-jammy) ...
Setting up python3-paramiko (2.9.3-0ubuntu1) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

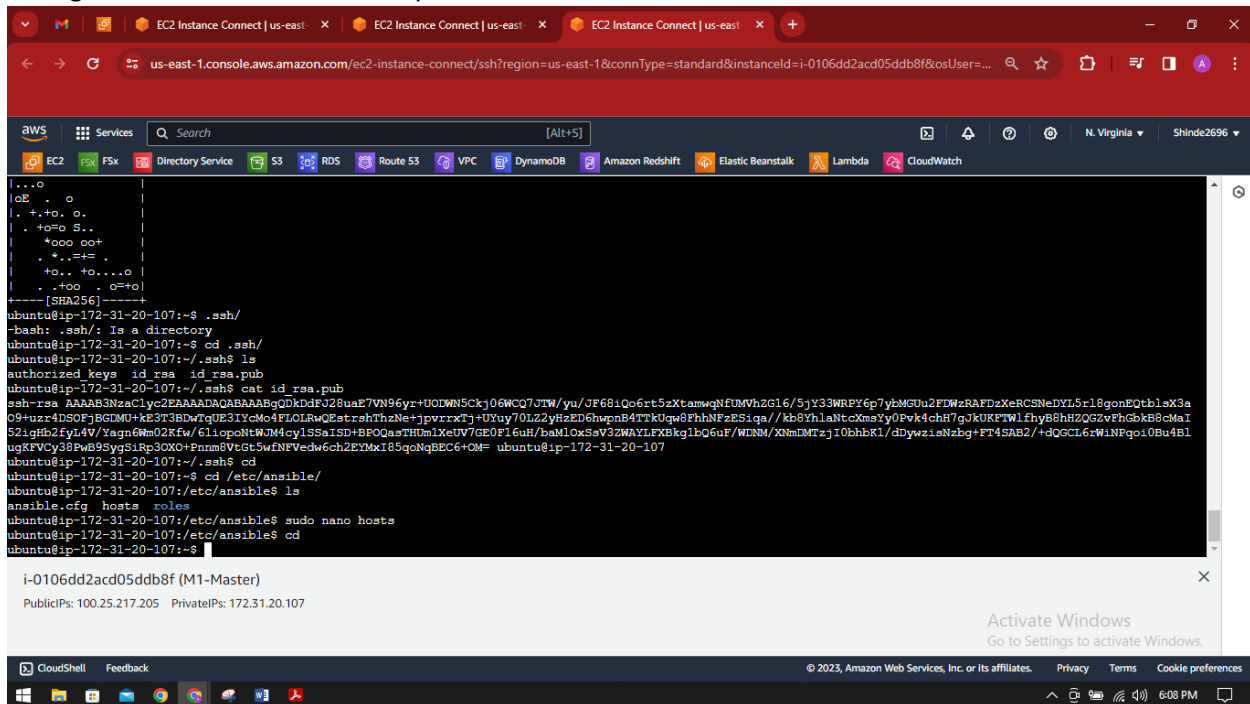
No VM guests are running outdated hypervisor (qemu) binaries on this host.

ubuntu@ip-172-31-20-107:~$

i-0106dd2acd05ddb8f (M1-Master)
PublicIPs: 100.25.217.205 PrivateIPs: 172.31.20.107

Activate Windows
Go to Settings to activate Windows.
```

Configured the Ansible successfully and the Master machine is linked to both the Sleeves machine



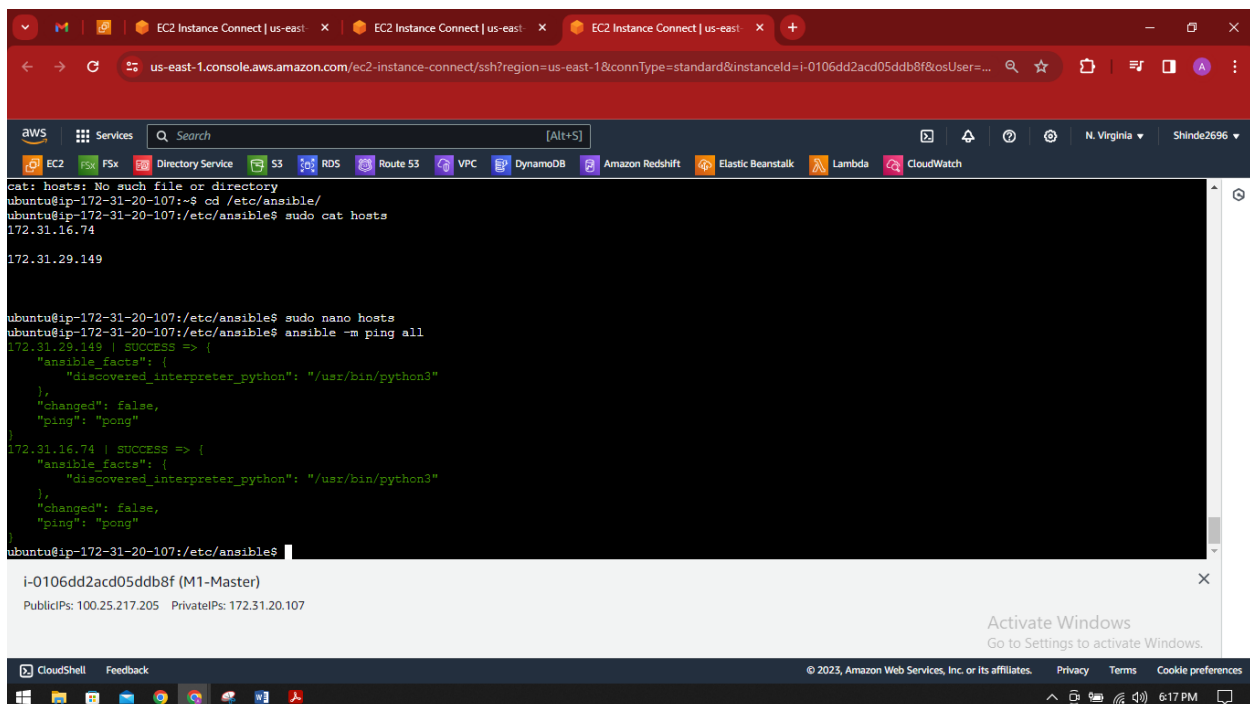
The screenshot shows the AWS CloudShell interface with a terminal window. The terminal output shows the user navigating to the .ssh directory, listing files, and displaying the contents of the authorized_keys file. The user then navigates to the /etc/ansible directory and lists files. Finally, the user runs 'sudo nano hosts' to edit the hosts file.

```
i-0106dd2acd05ddb8f (M1-Master)
PublicIPs: 100.25.217.205 PrivateIPs: 172.31.20.107

Activate Windows
Go to Settings to activate Windows.

CloudShell Feedback
© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences
6:08 PM
```

Add the Private ip address of the 2 sleeves machine to the directory of the Anisble on the Master machine and then ping all



The screenshot shows the AWS CloudShell interface with a terminal window. The terminal output shows the user running 'cat /etc/ansible/hosts' which returns an error. The user then runs 'sudo cat /etc/ansible/hosts' and pastes the private IP addresses of the two sleeve machines. Finally, the user runs 'ansible -m ping all' to verify connectivity to both machines.

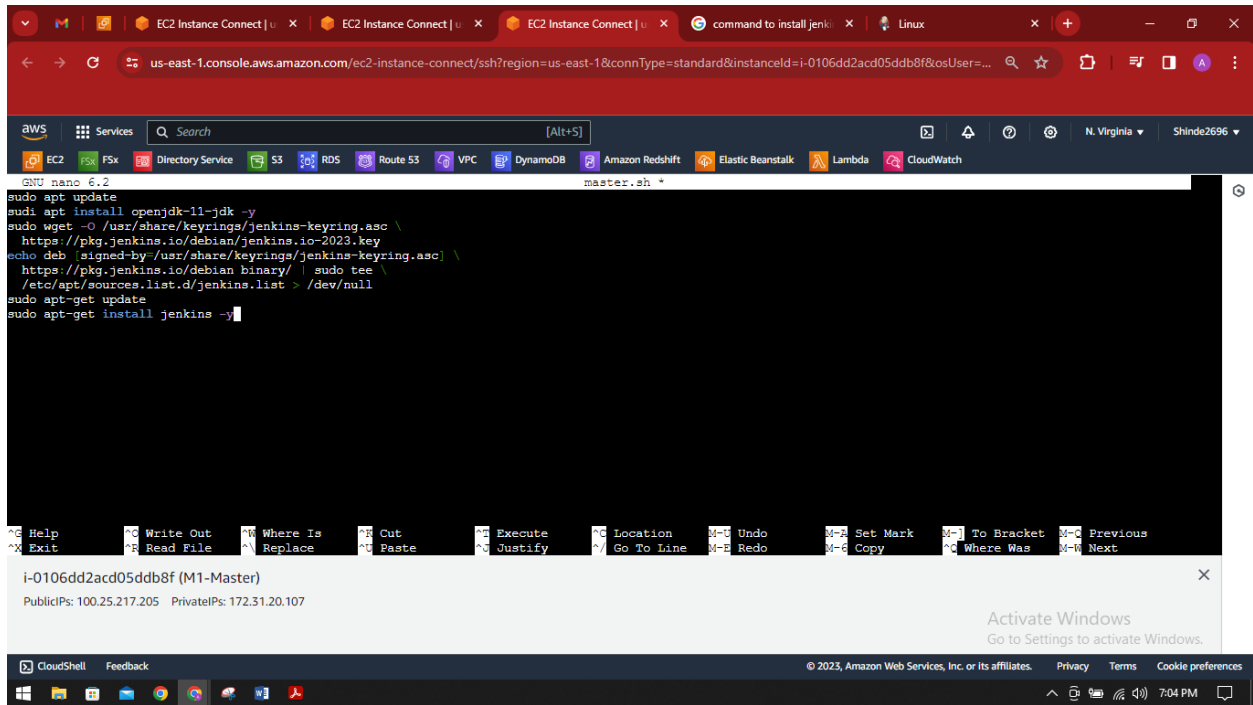
```
i-0106dd2acd05ddb8f (M1-Master)
PublicIPs: 100.25.217.205 PrivateIPs: 172.31.20.107

Activate Windows
Go to Settings to activate Windows.

CloudShell Feedback
© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences
6:17 PM
```

Installing Jenkins and Java in the Master machine using commands

```
sudo apt update
sudo apt install openjdk-11-jdk -y
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian/jenkins.io-2023.key
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins -y
```



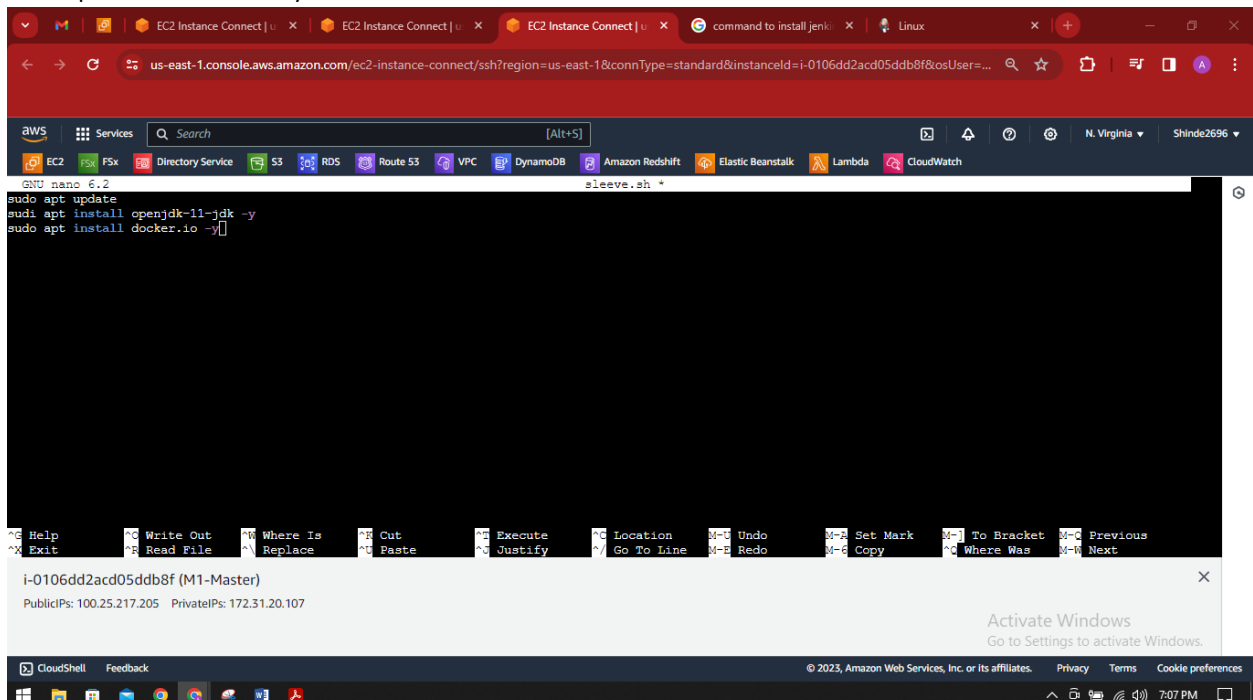
The screenshot shows the AWS CloudShell interface with a terminal window titled 'master.sh'. The terminal displays the following commands and their output:

```
GNU nano 6.2
sudo apt update
sudo apt install openjdk-11-jdk -y
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian/jenkins.io-2023.key
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins -y
```

The terminal output shows the package lists being updated and the installation of openjdk-11-jdk. The bottom of the screenshot shows the AWS CloudShell interface with the instance ID 'i-0106dd2acd05ddb8f (M1-Master)' and public/private IP addresses.

And then Installing Java and Docker in the Sleeves machine using below commands

```
sudo apt update
sudo apt install openjdk-11-jdk -y
sudo apt install docker.io -y
```

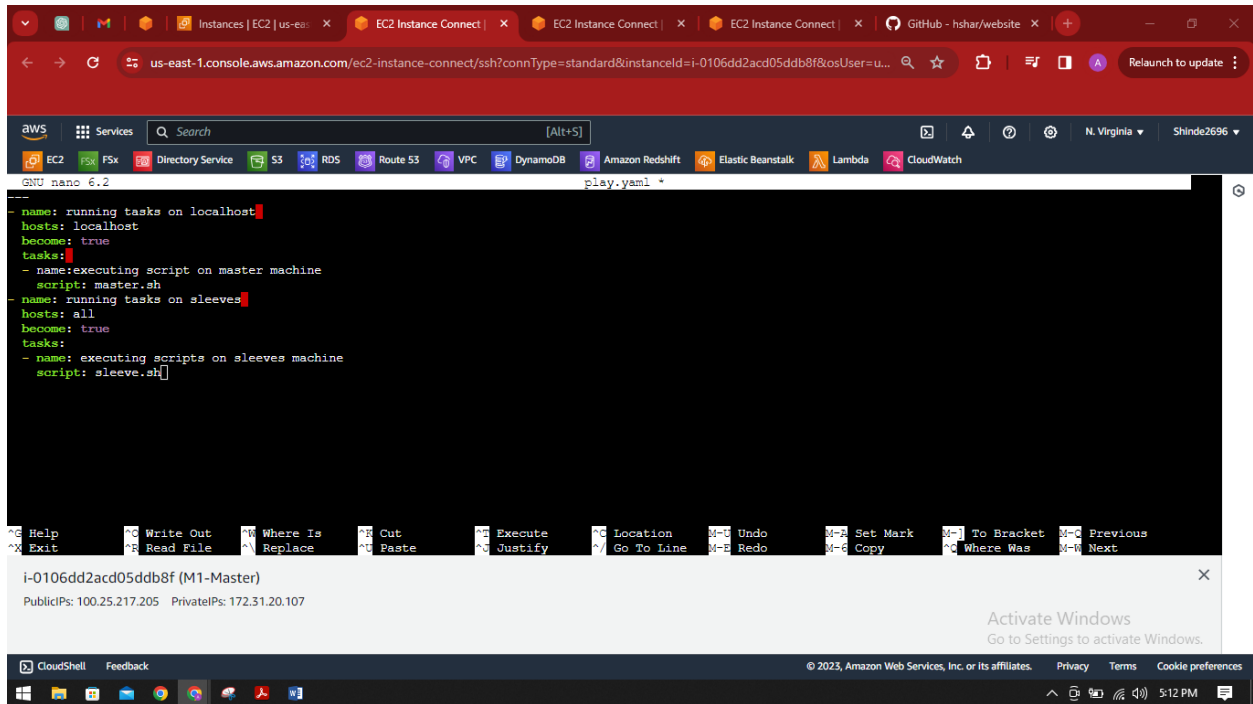


The screenshot shows the AWS CloudShell interface with a terminal window titled 'sleeve.sh'. The terminal displays the following commands and their output:

```
GNU nano 6.2
sudo apt update
sudo apt install openjdk-11-jdk -y
sudo apt install docker.io -y
```

The terminal output shows the package lists being updated and the installation of openjdk-11-jdk and docker.io. The bottom of the screenshot shows the AWS CloudShell interface with the instance ID 'i-0106dd2acd05ddb8f (M1-Master)' and public/private IP addresses.

Create a playbook to execute for the Master as well as Sleeves machines named play.yaml

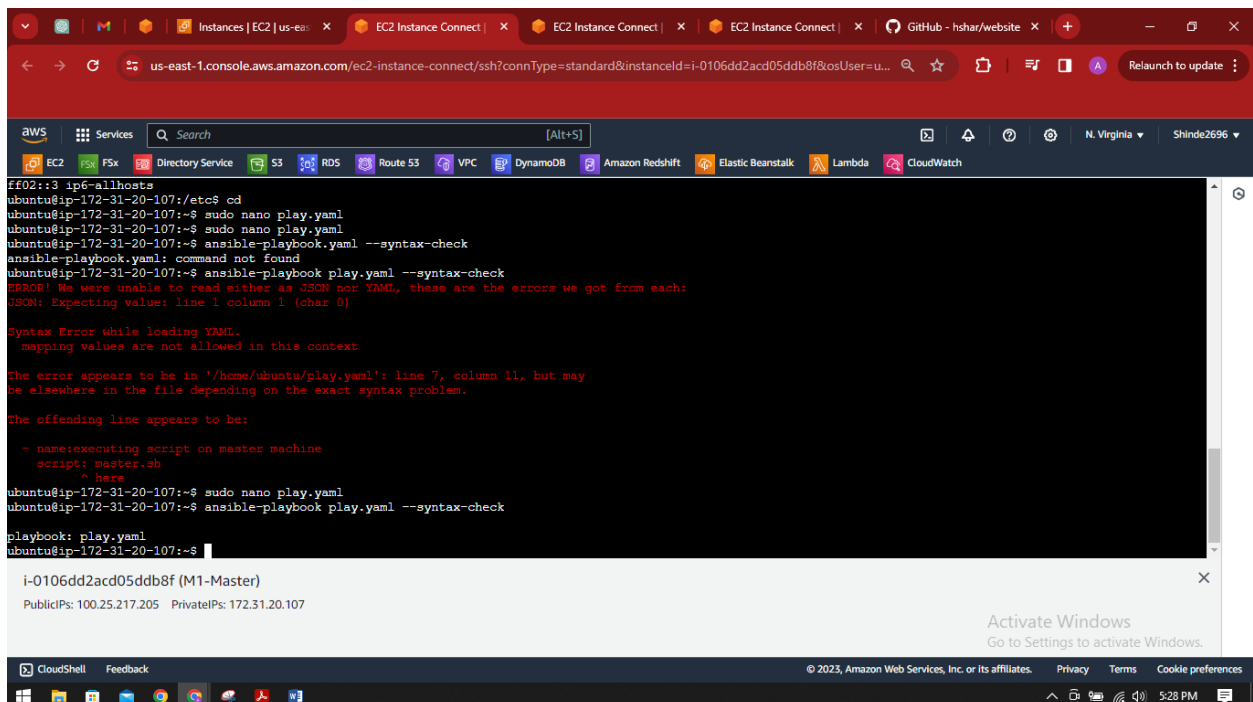


The screenshot shows the AWS CloudShell interface. At the top, there's a browser window with the URL `us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0106dd2acd05ddb8f&osUser=u...`. Below the browser, the AWS Services menu is visible. The main terminal window shows the `nano` editor editing `play.yaml`. The content of the file is as follows:

```
---
- name: running tasks on localhost
  hosts: localhost
  become: true
  tasks:
    - name: executing script on master machine
      script: master.sh
- name: running tasks on sleeves
  hosts: all
  become: true
  tasks:
    - name: executing scripts on sleeves machine
      script: sleeve.sh
```

At the bottom of the terminal, the instance information is displayed: `i-0106dd2acd05ddb8f (M1-Master)`, `PublicIPs: 100.25.217.205`, and `PrivateIPs: 172.31.20.107`. An `Activate Windows` watermark is visible in the bottom right corner.

And then ran did a syntax check and then dry run to check if there are any error,
And found an error and then checked in the play.yaml and then corrected it and then checked again.



The screenshot shows the AWS CloudShell interface with the terminal window displaying the following commands and output:

```
ff02::3 ip6-allhosts
ubuntu@ip-172-31-20-107:/etc$ cd
ubuntu@ip-172-31-20-107:~$ sudo nano play.yaml
ubuntu@ip-172-31-20-107:~$ sudo nano play.yaml
ubuntu@ip-172-31-20-107:~$ ansible-playbook play.yaml --syntax-check
ansible-playbook.yaml: command not found
ubuntu@ip-172-31-20-107:~$ ansible-playbook play.yaml --syntax-check
ERROR! We were unable to read either as JSON nor YAML, these are the errors we got from each:
JSON: Expecting value: line 1 column 1 (char 0)

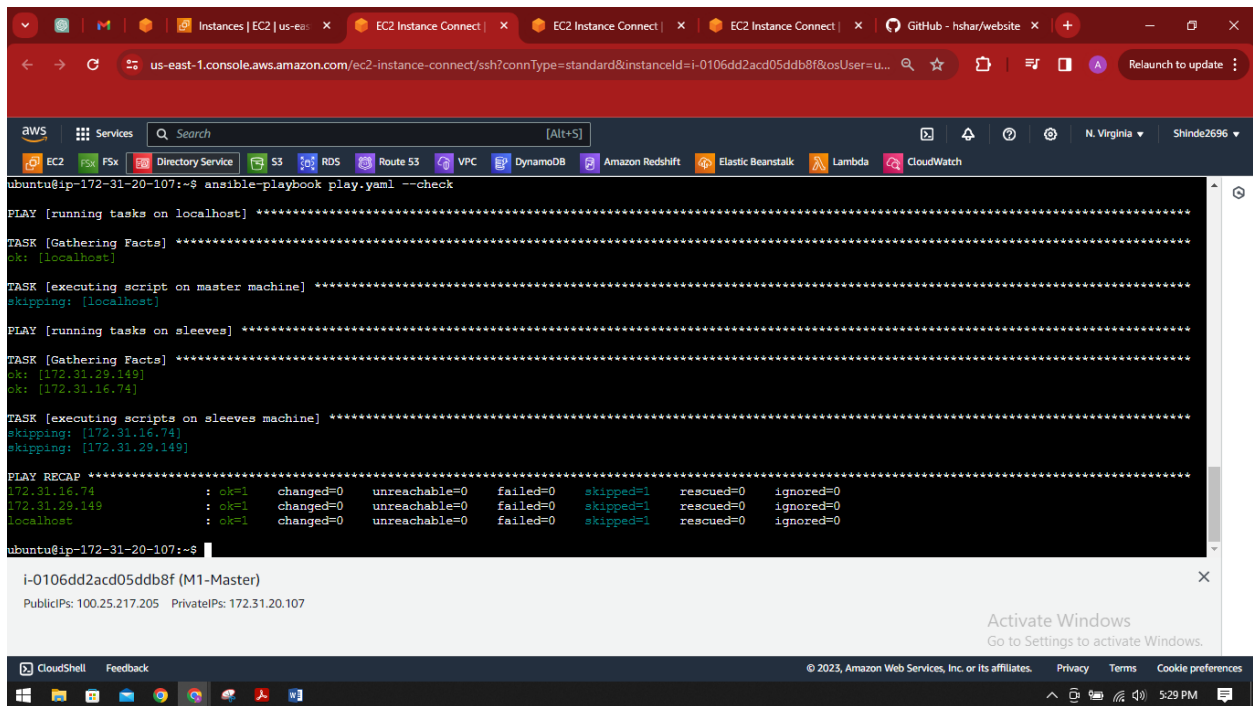
Syntax Error while loading YAML.
  mapping values are not allowed in this context

The error appears to be in '/home/ubuntu/play.yaml': line 7, column 11, but may
be elsewhere in the file depending on the exact syntax problem.

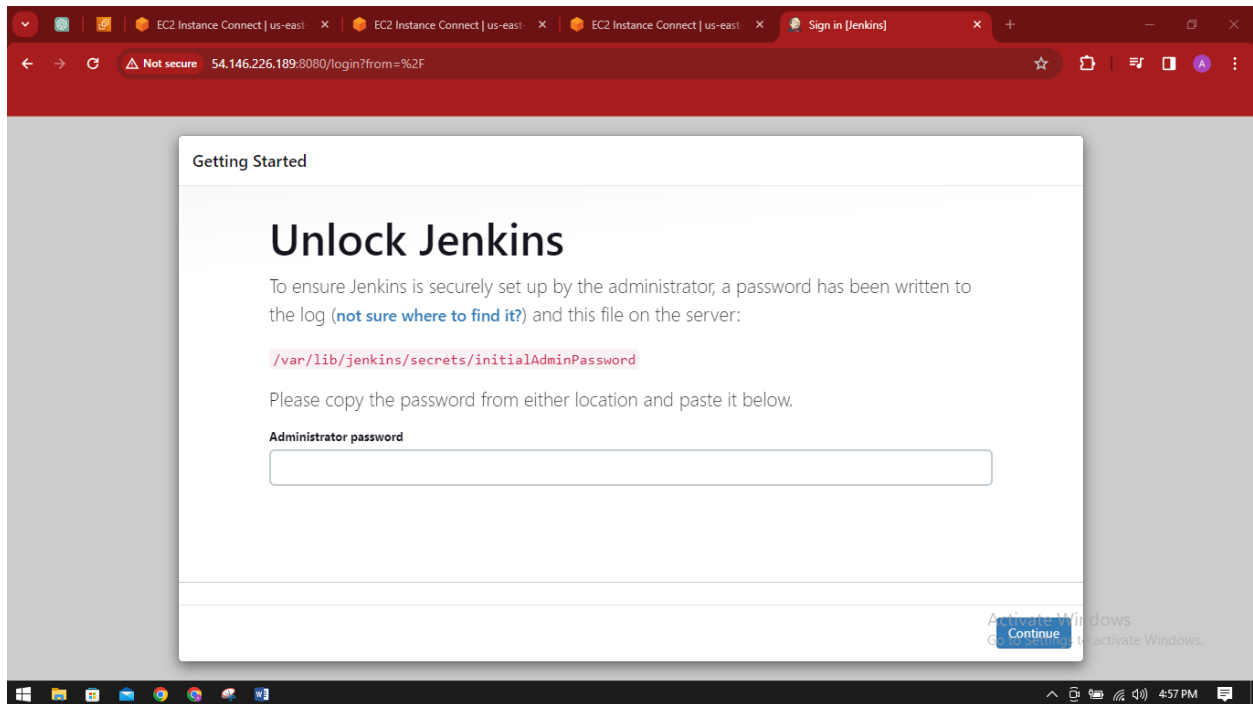
The offending line appears to be:

    - name: executing script on master machine
      script: master.sh
        ^ here
ubuntu@ip-172-31-20-107:~$ sudo nano play.yaml
ubuntu@ip-172-31-20-107:~$ ansible-playbook play.yaml --syntax-check
playbook: play.yaml
ubuntu@ip-172-31-20-107:~$
```

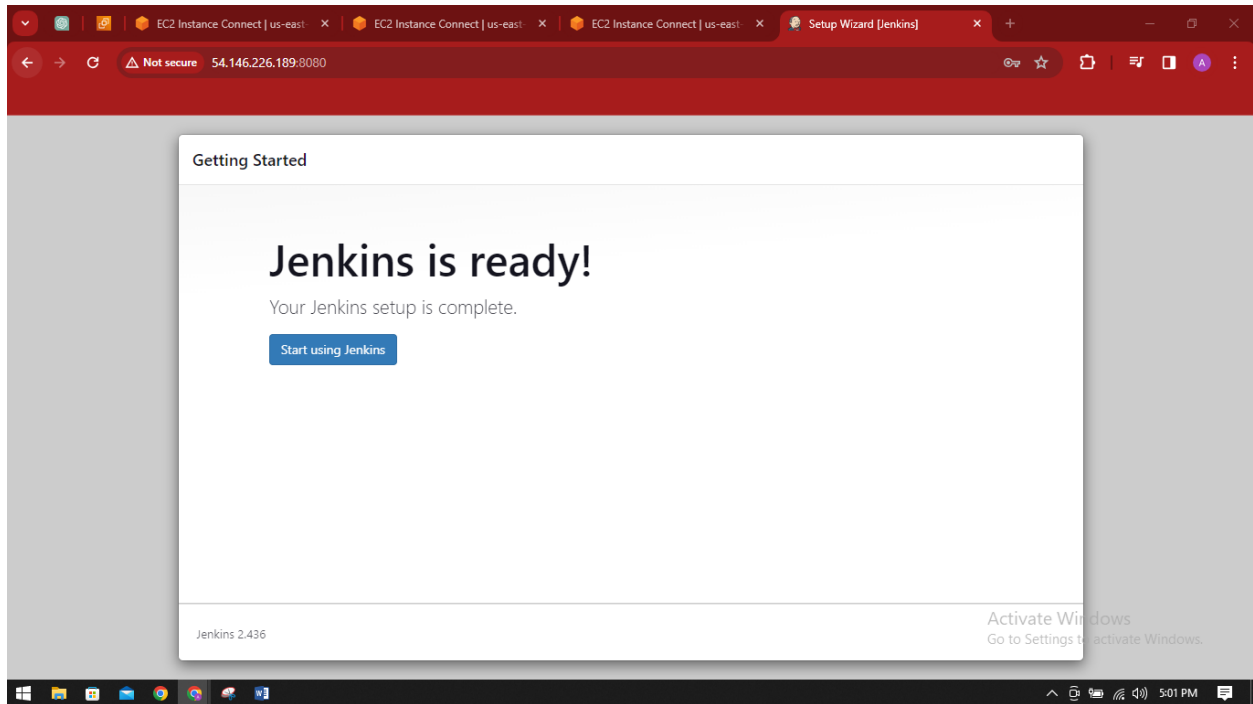
The instance information at the bottom remains the same: `i-0106dd2acd05ddb8f (M1-Master)`, `PublicIPs: 100.25.217.205`, and `PrivateIPs: 172.31.20.107`. The `Activate Windows` watermark is also present.



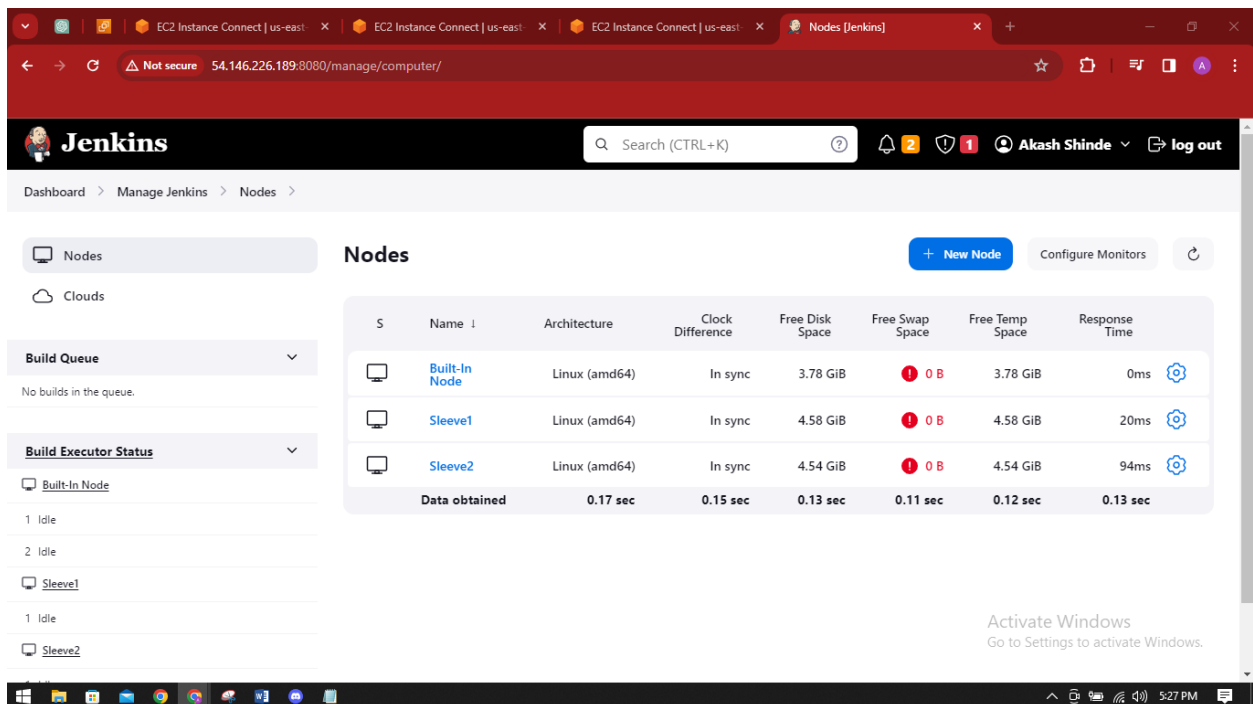
Hence everything was run successfully executing the file to install the required tools,
And we can see the Jenkins page when we go to the public IP on port 8080



Create Admin User and set the password and then hence Completed the Jenkins setup

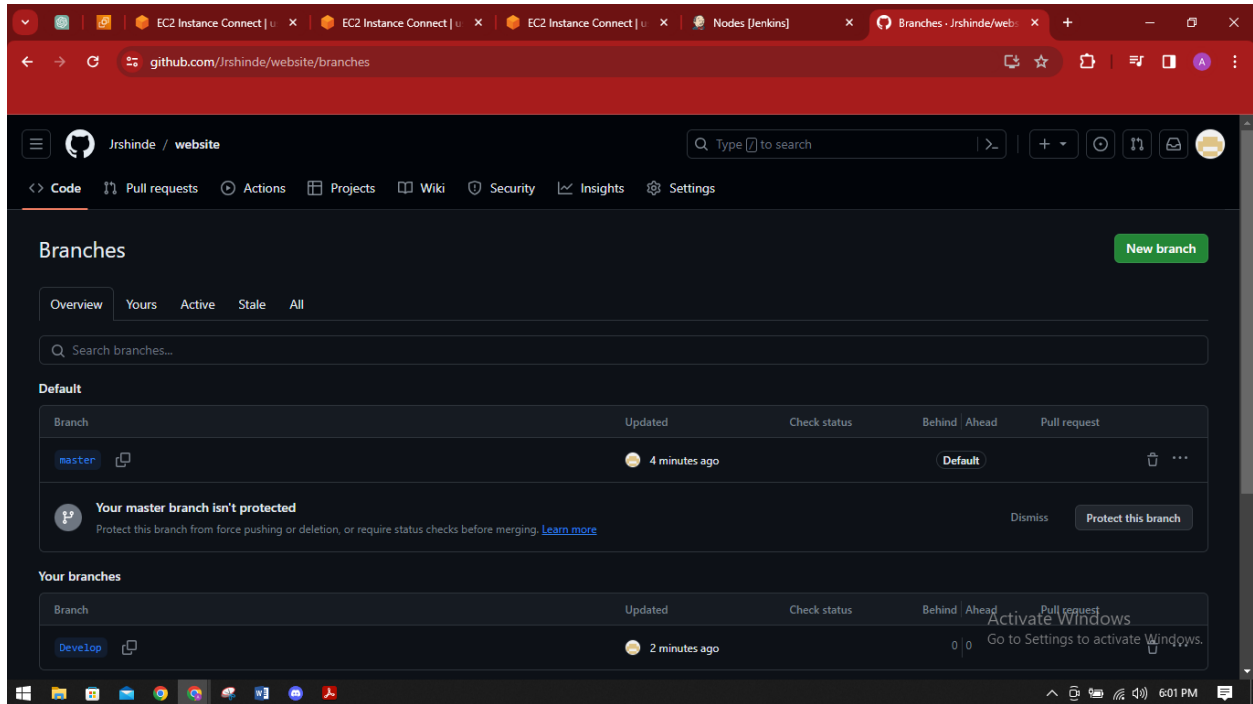


Created nodes for the Sleeves machine in the Jenkins named Sleeve1 and Sleeve2

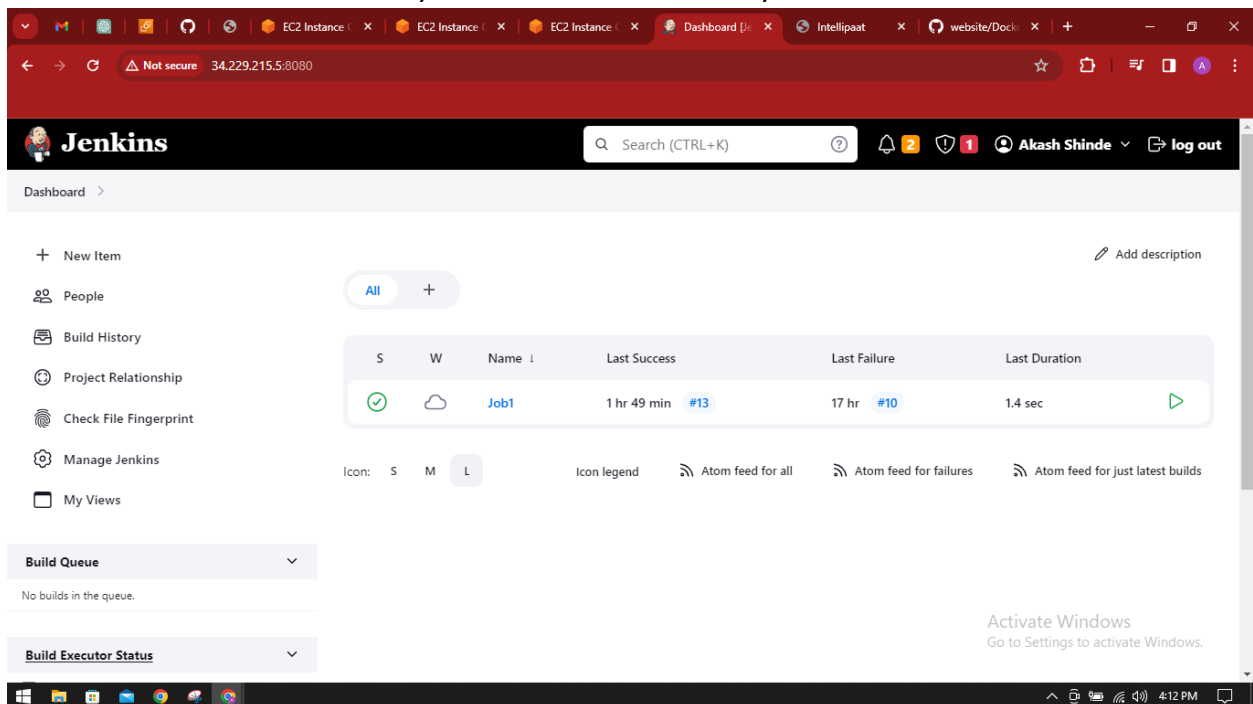


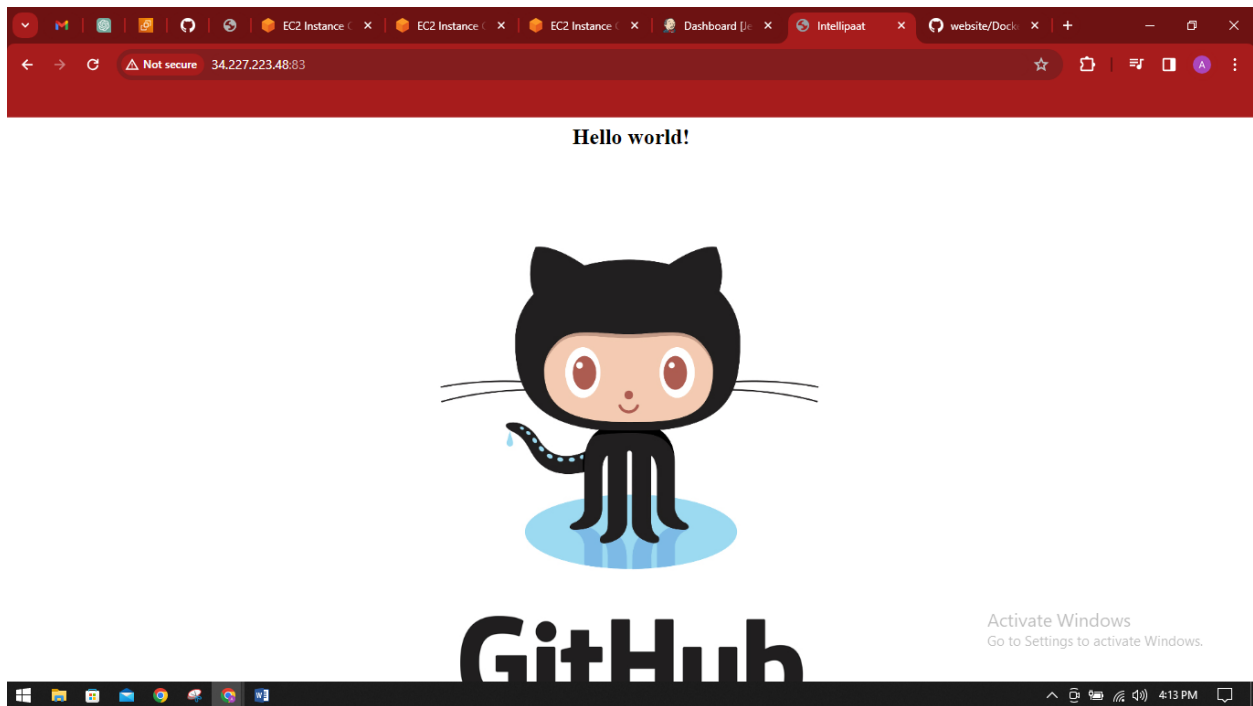
Created a Docker file in the Git and then created 2 branches one Master and then it's sub branch Develops

And saved the code in '/var/www/html'



Hence Job 1 created successfully and executed successfully





So All the 3 Jobs are build successfully

Dashboard >

+ New Item Add description

People All +

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☁	Job1	16 min #17	17 min #16	1.2 sec ▶
✓	☀	Job2	3 min 11 sec #3	N/A	1.2 sec ▶
✓	☀	Job3	1 min 2 sec #1	N/A	30 sec ▶

Build Queue ▼ Icon: S M L Icon legend Atom feed for all Atom feed for failures Atom feed for just latest builds

No builds in the queue.

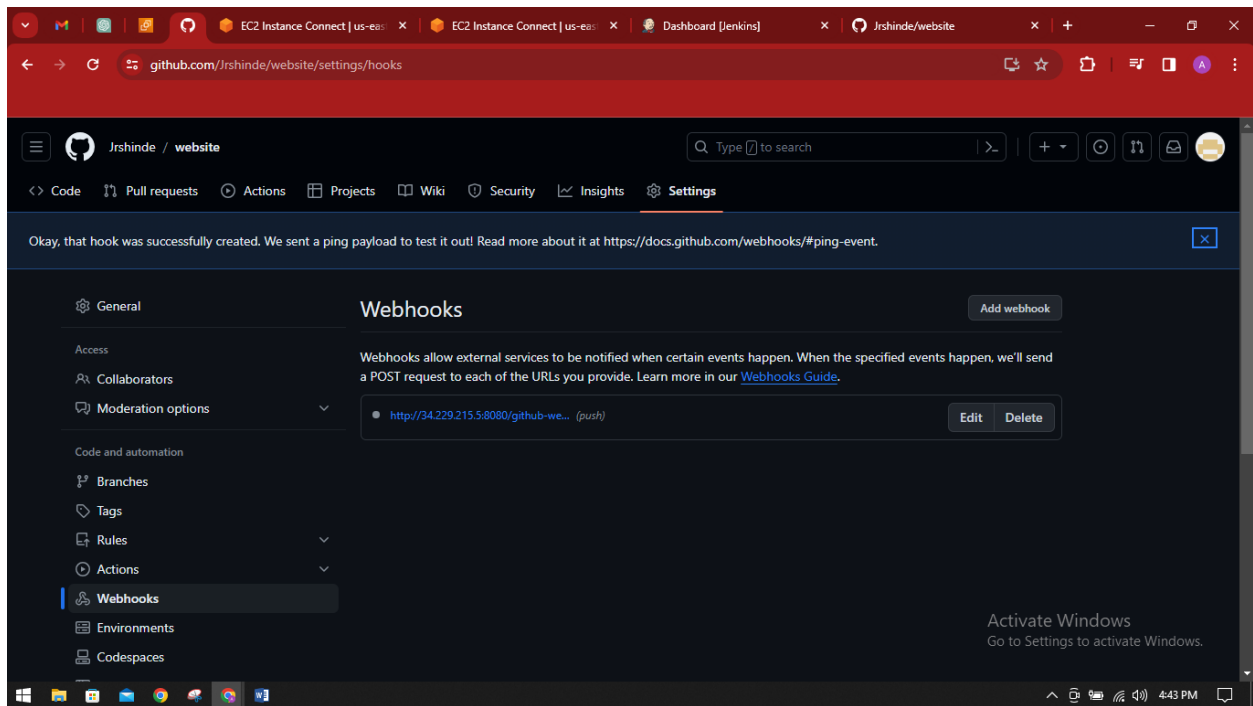
Build Executor Status ▼

Connected the Job2 and Job3, so that once the Job2 is ran successfully it will trigger the Job3 and run automatically which means Job3 is queued after the Job2.

The top screenshot shows the Jenkins 'Configure' page for Job2. The left sidebar lists configuration options: General, Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions. The 'Post-build Actions' section is expanded, showing a 'Build other projects' action. The 'Projects to build' field contains 'Job3'. The trigger options are: 'Trigger only if build is stable' (selected), 'Trigger even if the build is unstable', and 'Trigger even if the build fails'. There are 'Save' and 'Apply' buttons at the bottom.

The bottom screenshot shows the Jenkins 'Job2' overview page. The left sidebar lists actions: Status, Changes, Workspace, Build Now, Configure, Delete Project, GitHub Hook Log, and Rename. The main content area shows 'Downstream Projects' with 'Job3' listed. Below this, the 'Permalinks' section lists build history links: 'Last build (#5), 2 min 38 sec ago', 'Last stable build (#5), 2 min 38 sec ago', 'Last successful build (#5), 2 min 38 sec ago', and 'Last completed build (#5), 2 min 38 sec ago'. There are also buttons for 'Add description' and 'Disable Project'.

And a Webhook is also added in the Github too:



So in this project the successful implementation of the DevOps lifecycle at Adobe Software has significantly improved the efficiency and reliability of our product development processes. Through the integration of Ansible for software provisioning, Gitflow for version control, and Jenkins for continuous integration and deployment, we have established a streamlined and automated workflow. The three-stage pipeline, encompassing Build, Test, and Prod stages, ensures that code changes undergo thorough testing before reaching production.