

Module Name : Capstone Project

Project name : HealthCare domain

Submitted by : Sachin V Bacha

On : 25-05-2025

Business challenge/requirement

As soon as the developer pushes the updated code on the GIT master branch, the Jenkins pipeline should be triggered and code should be checkout, compiled, tested, packaged and containerized. A new test-cluster should be provisioned and configured automatically with all the required software's and as soon as the cluster is healthy and available, the application must be deployed to the test-server automatically using Kubernetes.

The deployment should then be tested using a test automation tool, and if the build is successful, it should be deployed to the prod server/cluster using Kubernetes.

All this should happen automatically and should be triggered from a push to the GitHub master branch. Kubernetes cluster must contain at least 2 servers and must be monitored continuously using Prometheus and dashboard must be visualized using Grafana.

Tech stack

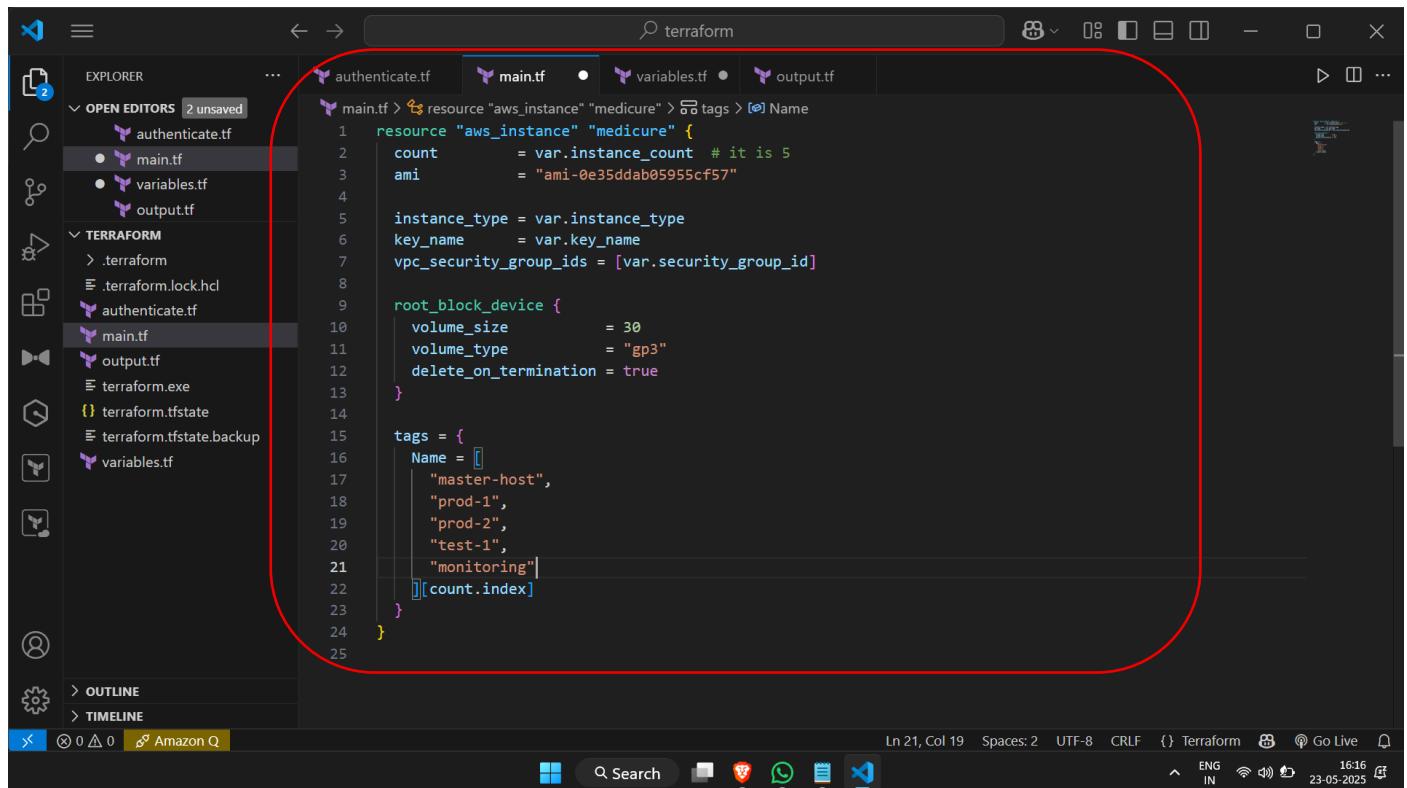
- ✓ Git - For version control for tracking changes in the code files
- ✓ Jenkins - For continuous integration and continuous deployment
- ✓ Docker - For containerizing applications
- ✓ Ansible - Configuration management tools
- ✓ Selenium - For automating tests on the deployed web application
- ✓ Terraform - For creation of infrastructure.
- ✓ Kubernetes – for running containerized application in managed cluster

Launch 5 Instance with terraform

Name	Instance type	OS
Host-master	- t3.medium	ubuntu
Production-1	- t3.medium	ubuntu
Production-2	- t3.medium	ubuntu
Test-1	- t2.medium	ubuntu
Monitoring	-t3.medium	ubuntu

1. Create a terraform directory consist of a terraform.exe file
2. Create Four files authenticate.tf , main.tf , output.tf , variables.tf

Main.tf

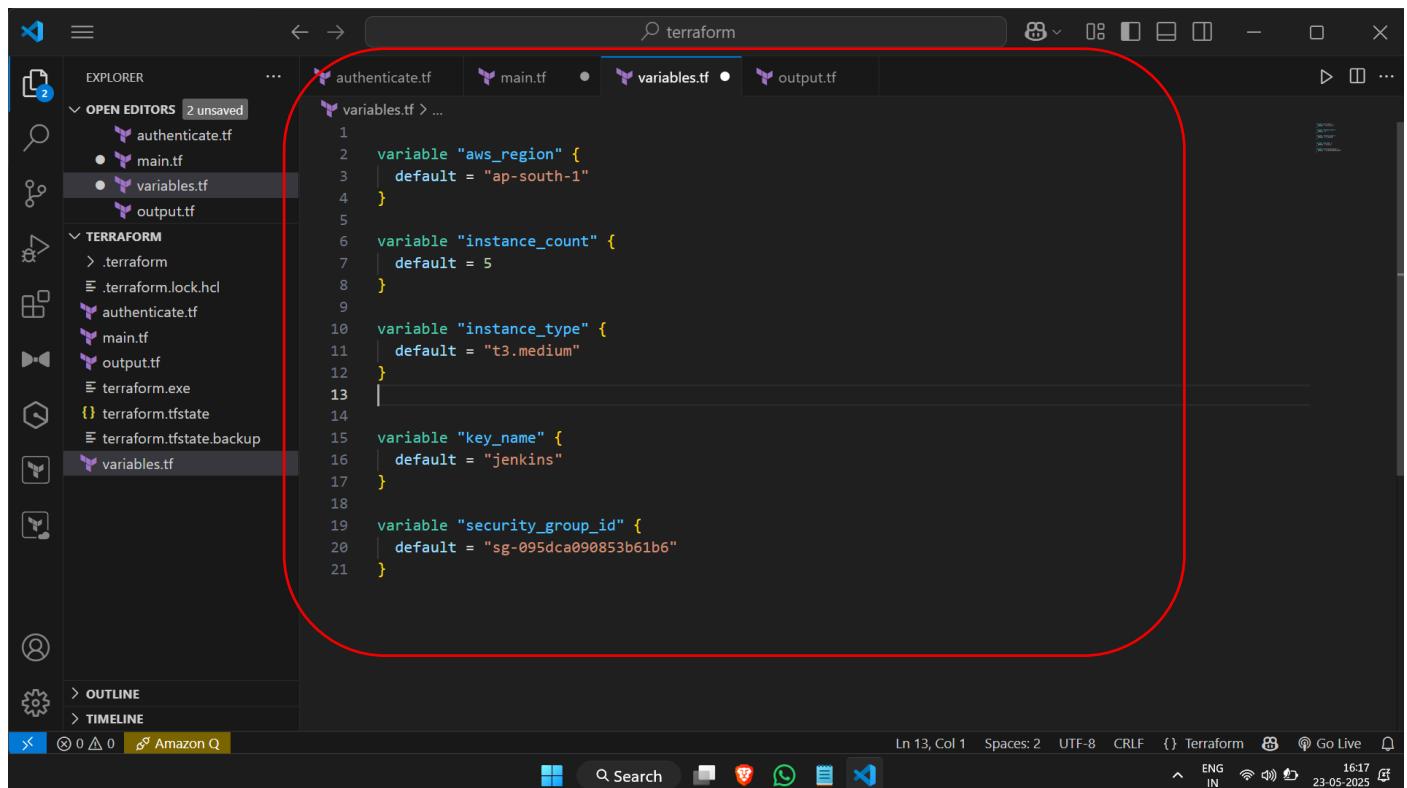


```
resource "aws_instance" "medicure" {
  count      = var.instance_count # it is 5
  ami        = "ami-0e35ddab05955cf57"
  instance_type = var.instance_type
  key_name   = var.key_name
  vpc_security_group_ids = [var.security_group_id]

  root_block_device {
    volume_size      = 30
    volume_type     = "gp3"
    delete_on_termination = true
  }

  tags = [
    Name = [
      "master-host",
      "prod-1",
      "prod-2",
      "test-1",
      "monitoring"
    ][count.index]
  ]
}
```

Variables.tf



```
variable "aws_region" {
  default = "ap-south-1"
}

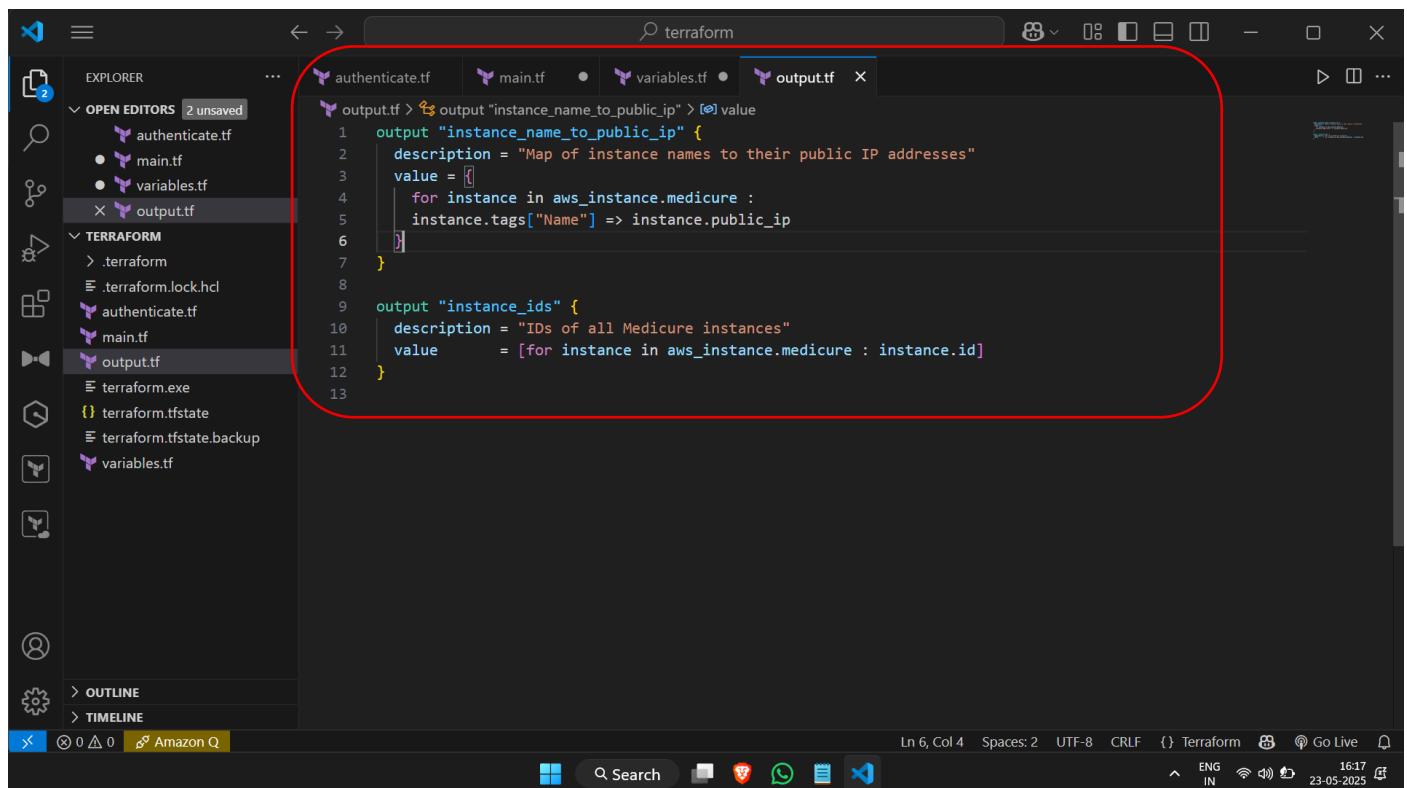
variable "instance_count" {
  default = 5
}

variable "instance_type" {
  default = "t3.medium"
}

variable "key_name" {
  default = "jenkins"
}

variable "security_group_id" {
  default = "sg-095dca090853b61b6"
}
```

Output.tf



```
output "instance_name_to_public_ip" {
  description = "Map of instance names to their public IP addresses"
  value = [
    for instance in aws_instance.medicure :
      instance.tags["Name"] => instance.public_ip
  ]
}

output "instance_ids" {
  description = "IDs of all Medicure instances"
  value      = [for instance in aws_instance.medicure : instance.id]
}
```

Open a new CMD terminal and initialize the terraform using “terraform init” command

The screenshot shows the VS Code interface with the terminal tab active. A red box highlights the terminal output which shows the successful initialization of Terraform.

```
(c) Microsoft Corporation. All rights reserved.  
D:\devops\Terraform>terraform init  
Initializing the backend...  
Initializing provider plugins...  
- Reusing previous version of hashicorp/aws from the dependency lock file  
- Using previously-installed hashicorp/aws v5.97.0  
  
Terraform has been successfully initialized!  
  
You may now begin working with Terraform. Try running "terraform plan" to see  
any changes that are required for your infrastructure. All Terraform commands  
should now work.  
  
If you ever set or change modules or backend configuration for Terraform,  
rerun this command to reinitialize your working directory. If you forget, other  
commands will detect it and remind you to do so if necessary.
```

Use “terraform plan” command to check and verify it the resources are matching as required then use command “terraform apply” to create the resources

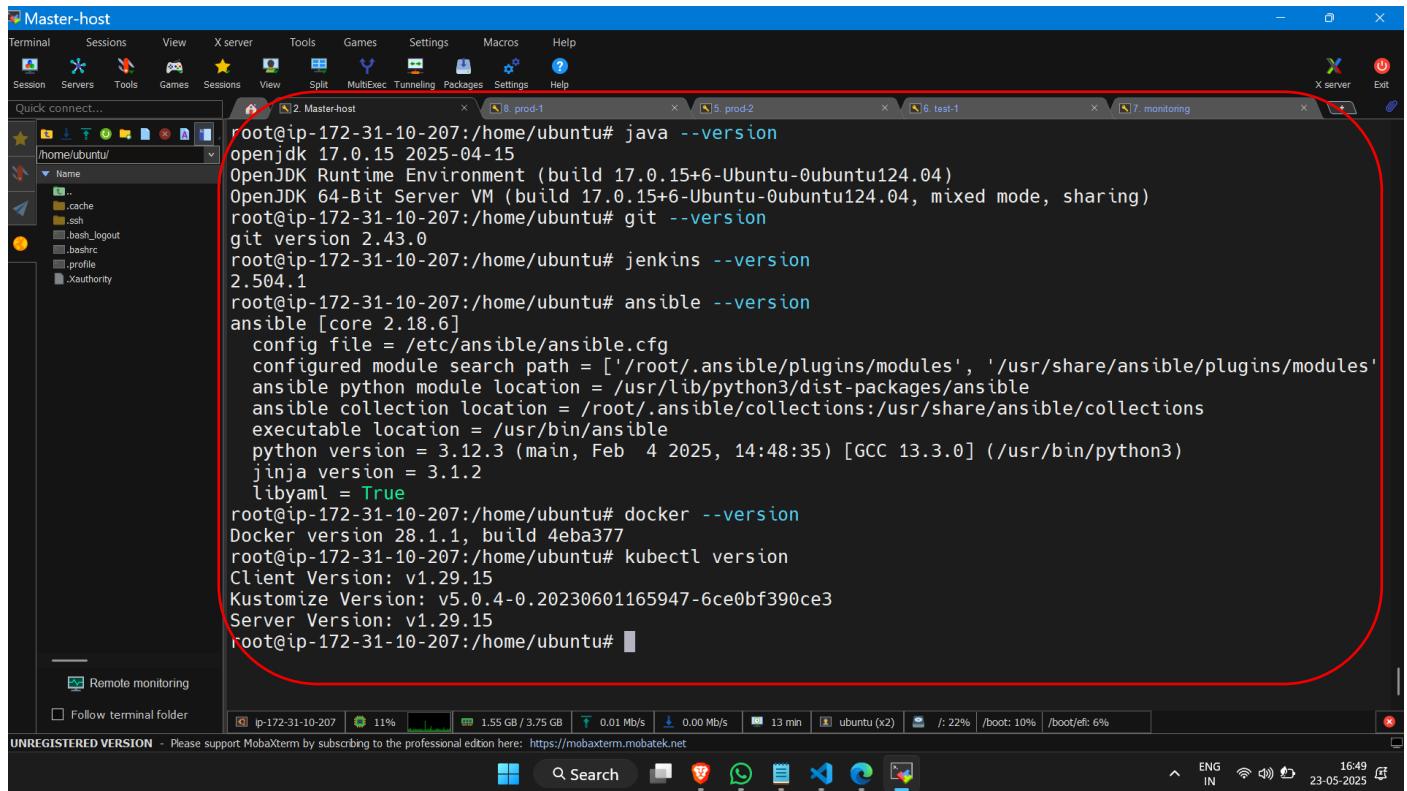
The screenshot shows the VS Code interface with the terminal tab active. A red box highlights the terminal output which shows the results of the terraform plan command, including instance IDs and public IP addresses.

```
Outputs:  
  
instance_ids = [  
    "i-065c300f301271e2c",  
    "i-02630d9959ac61f33",  
    "i-005441acf353580d2",  
    "i-0edd97904b767c184",  
    "i-00d9f293a2ef3c1ee",  
]  
instance_name_to_public_ip = {  
    "master-host" = "3.110.108.195"  
    "monitoring" = "43.204.97.169"  
    "prod-1" = "13.127.120.65"  
    "prod-2" = "43.204.102.113"  
    "test-1" = "13.127.93.81"  
}  
  
D:\devops\Terraform>
```

Connect all the instances through MobaXterm using the key pair attached in terraform script

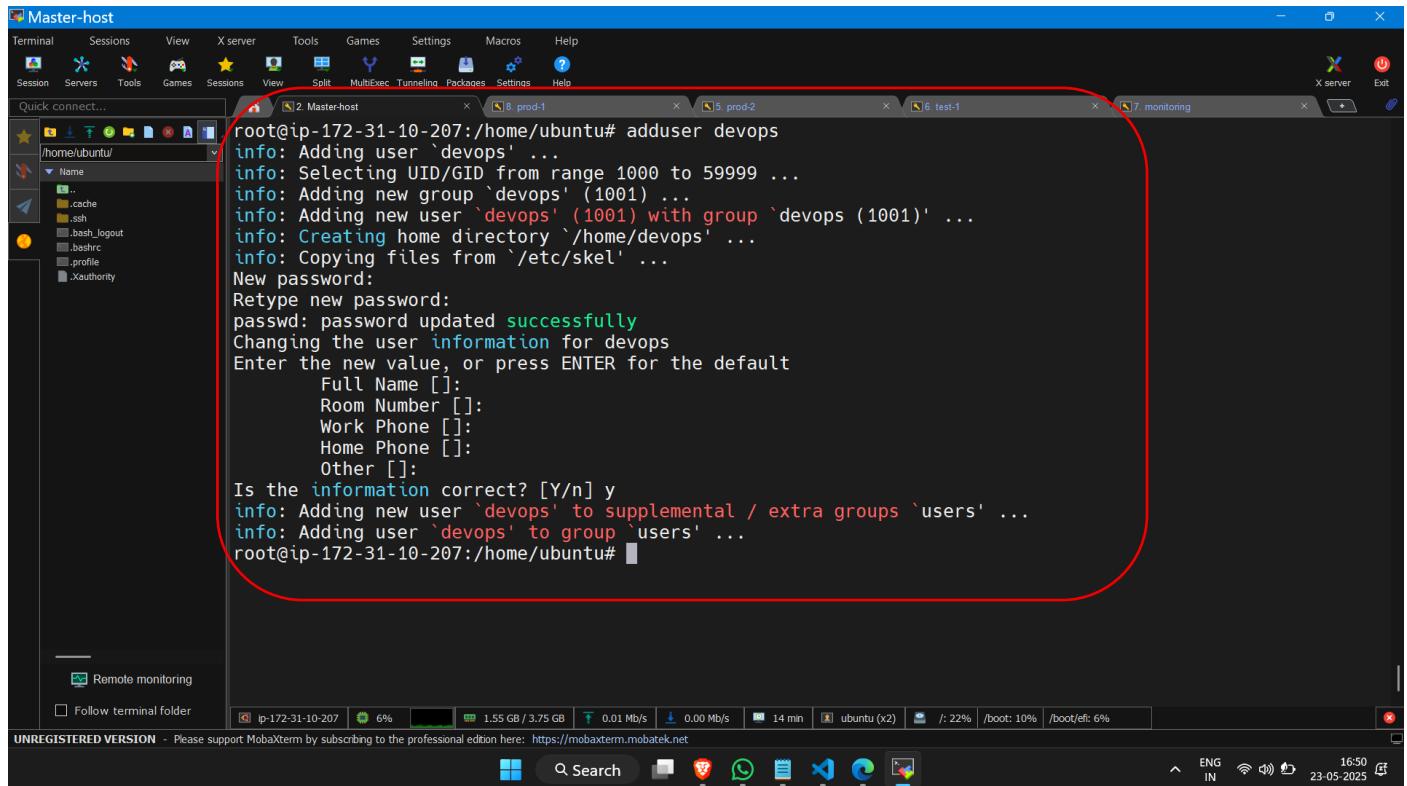
Install all the required packages in the master machine (Jenkins, Java, maven, git, ansible, docker, kubernetes)

which is the Jenkins host and ansible controller and the Kubernetes master node .



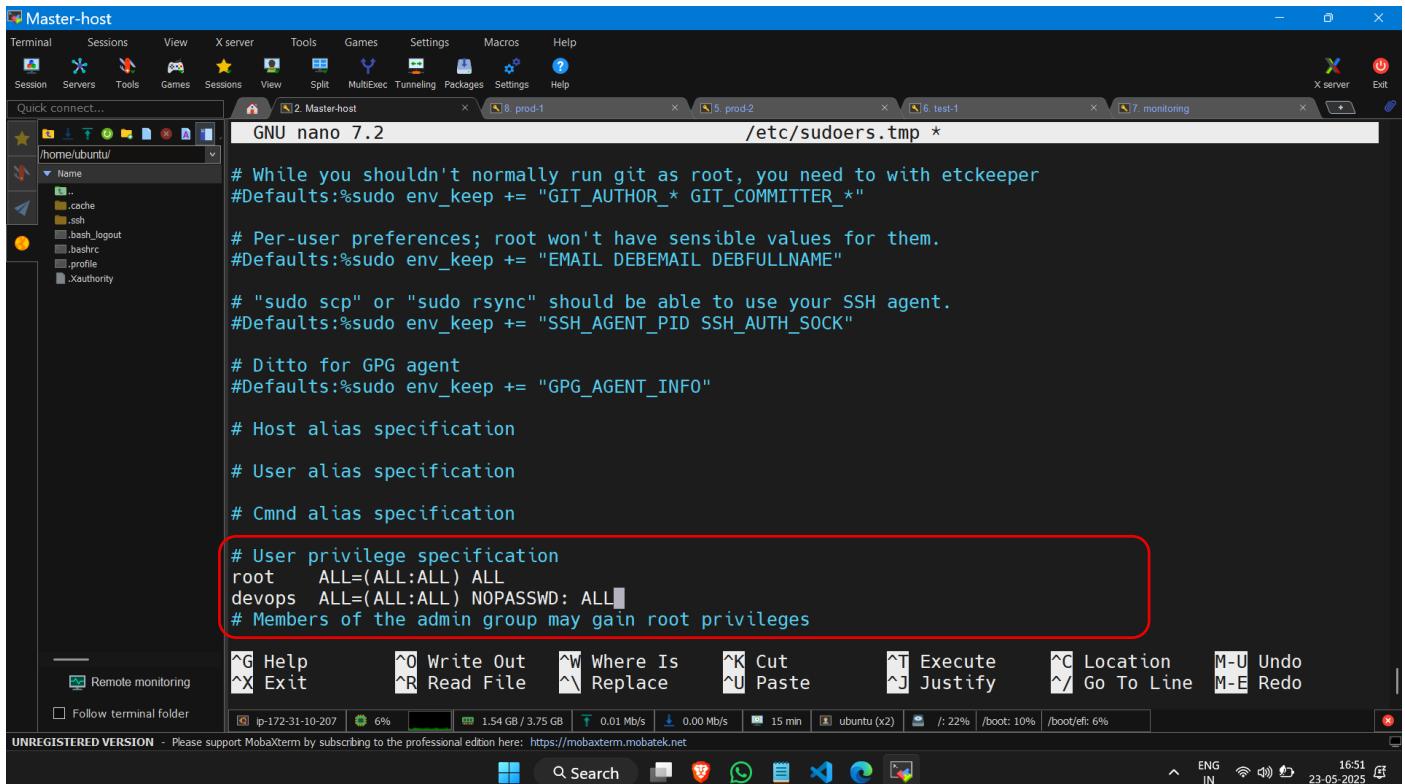
```
root@ip-172-31-10-207:/home/ubuntu# java --version
openjdk 17.0.15 2025-04-15
OpenJDK Runtime Environment (build 17.0.15+6-Ubuntu-0ubuntu124.04)
OpenJDK 64-Bit Server VM (build 17.0.15+6-Ubuntu-0ubuntu124.04, mixed mode, sharing)
root@ip-172-31-10-207:/home/ubuntu# git --version
git version 2.43.0
root@ip-172-31-10-207:/home/ubuntu# jenkins --version
2.504.1
root@ip-172-31-10-207:/home/ubuntu# ansible --version
ansible [core 2.18.6]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Feb 4 2025, 14:48:35) [GCC 13.3.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
root@ip-172-31-10-207:/home/ubuntu# docker --version
Docker version 28.1.1, build 4eba377
root@ip-172-31-10-207:/home/ubuntu# kubectl version
Client Version: v1.29.15
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Server Version: v1.29.15
root@ip-172-31-10-207:/home/ubuntu#
```

Now do Ansible configuration from master node to connect all other nodes as user "devops" with password
In master and all other nodes



```
root@ip-172-31-10-207:/home/ubuntu# adduser devops
info: Adding user `devops' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `devops' (1001) ...
info: Adding new user `devops' (1001) with group `devops (1001)' ...
info: Creating home directory `/home/devops' ...
info: Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for devops
Enter the new value, or press ENTER for the default
  Full Name []:
  Room Number []:
  Work Phone []:
  Home Phone []:
  Other []:
Is the information correct? [Y/n] y
info: Adding new user `devops' to supplemental / extra groups `users' ...
info: Adding user `devops' to group `users' ...
root@ip-172-31-10-207:/home/ubuntu#
```

Now add the user to the sudoers file with the permissions . perform the same in all the other nodes



```
GNU nano 7.2 /etc/sudoers.tmp *
# While you shouldn't normally run git as root, you need to with etckeeper
Defaults: %sudo env_keep += "GIT_AUTHOR_* GIT_COMMITTER_"

# Per-user preferences; root won't have sensible values for them.
Defaults: %sudo env_keep += "EMAIL DEBEMAIL DEBFULLNAME"

# "sudo scp" or "sudo rsync" should be able to use your SSH agent.
Defaults: %sudo env_keep += "SSH_AGENT_PID SSH_AUTH_SOCK"

# Ditto for GPG agent
Defaults: %sudo env_keep += "GPG_AGENT_INFO"

# Host alias specification

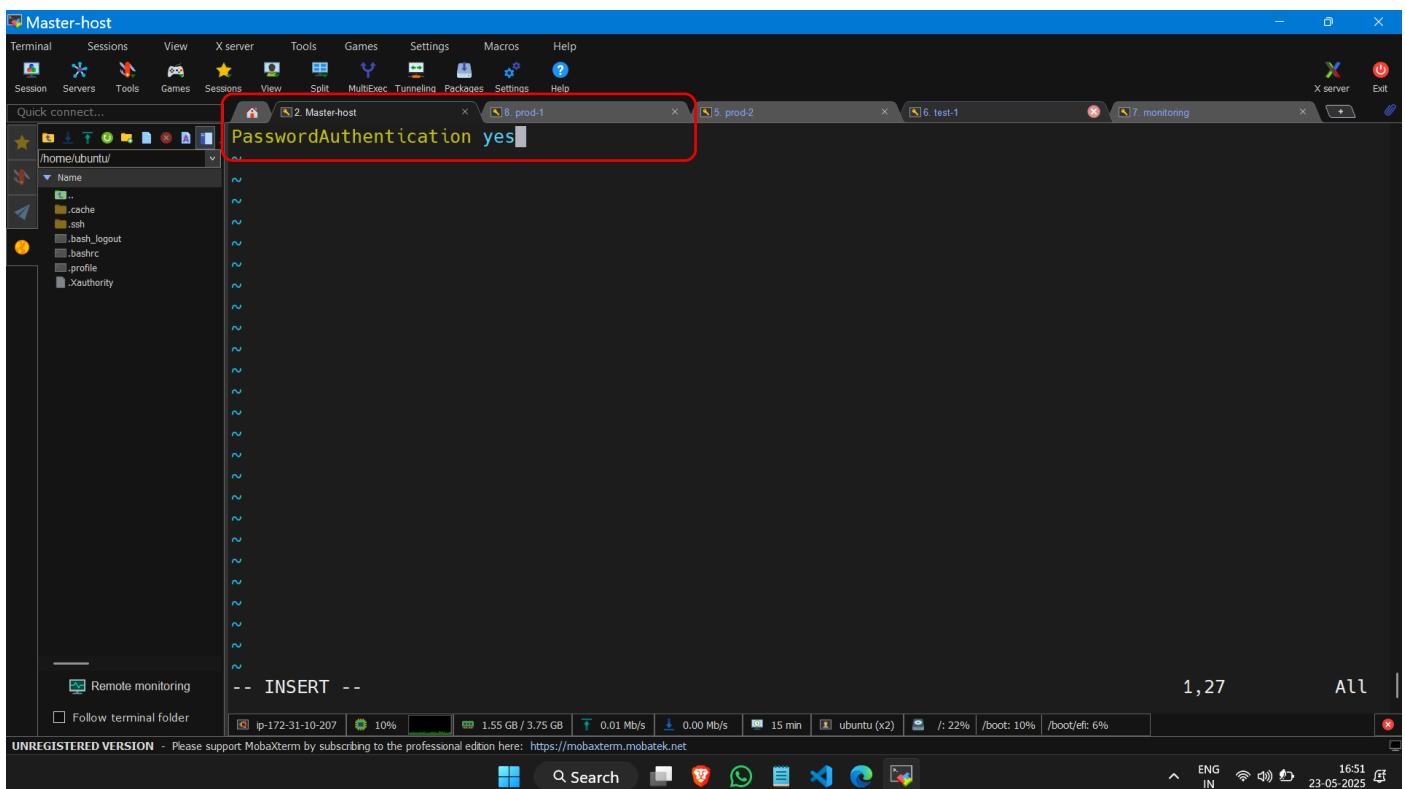
# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
devops  ALL=(ALL:ALL) NOPASSWD: ALL
# Members of the admin group may gain root privileges
```

Now configure sshd configuration in both master and other nodes

Get into the “ vi /etc/ssh/sshd_config.d/60-cloudimg-settings.conf ”and change passwordauthentication to yes



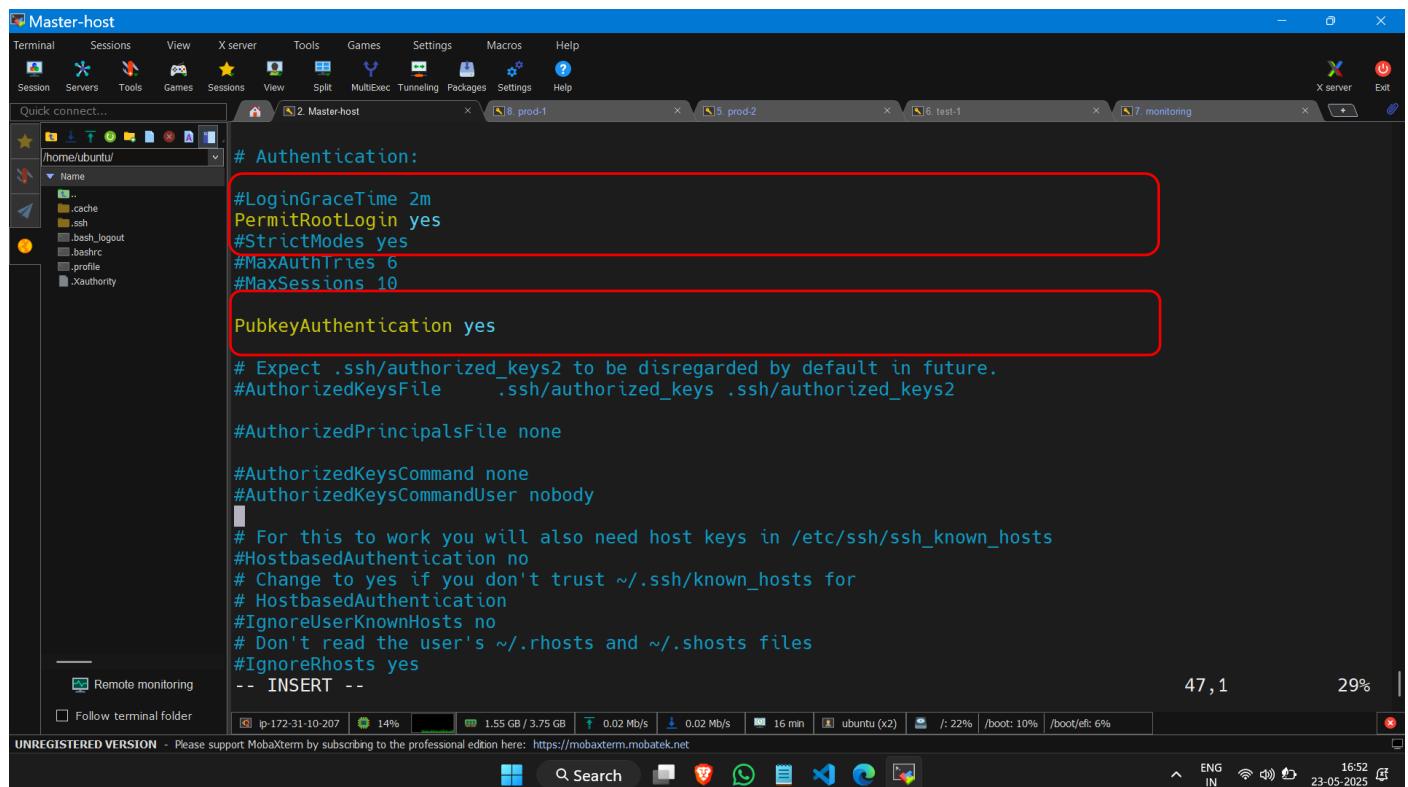
```
Master-host
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect... 2 Master-host x 3 prod-1 x 4 prod-2 x 5 test-1 x 6 monitoring x
GNU nano 7.2 /etc/ssh/sshd_config.d/60-cloudimg-settings.conf
PasswordAuthentication yes
```

Also change the configuration file in using command “vi /etc/ssh/sshd_config” change

Permitlogin to yes

Pubkeyauthentication to yes

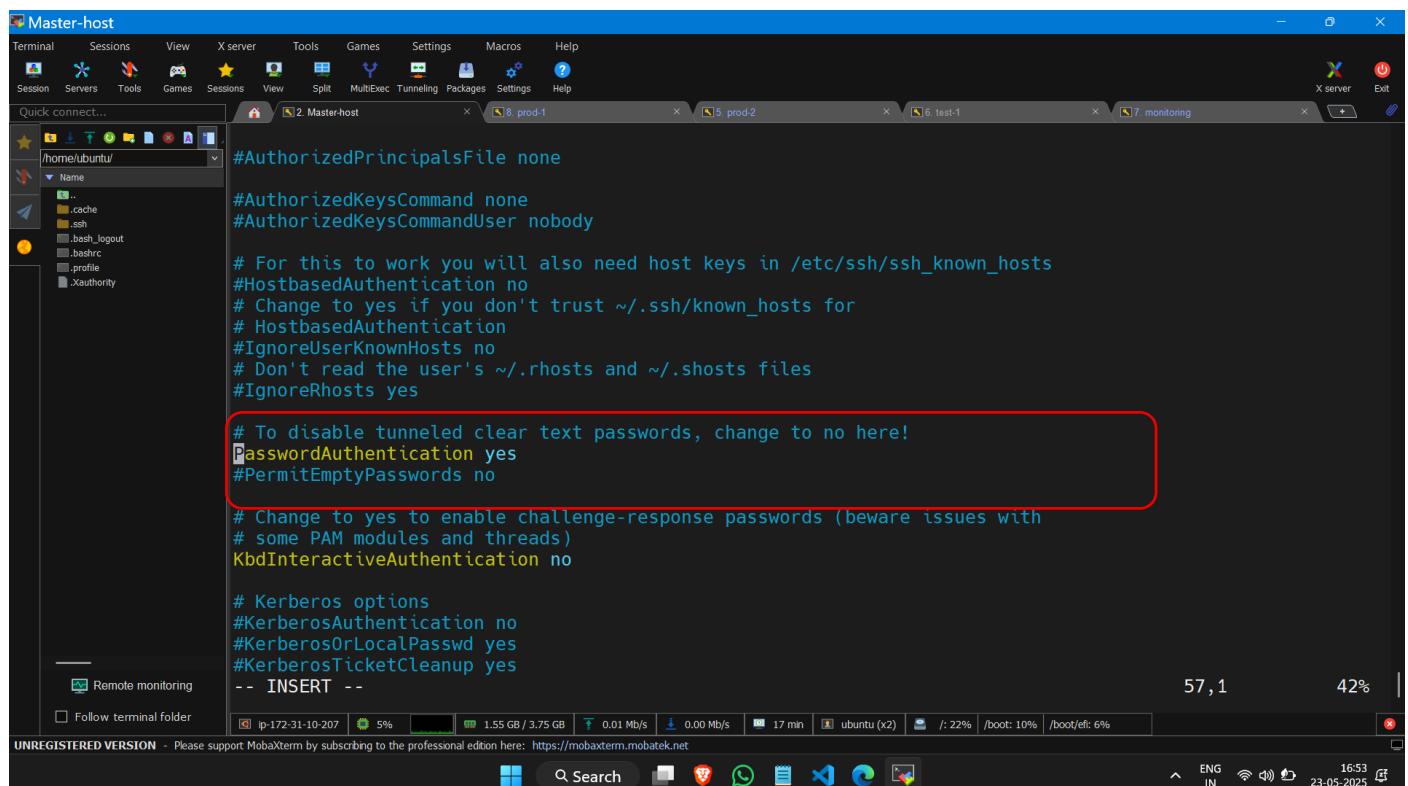
And passwordauthentication to yes -perform the same activity in nodes as well



The screenshot shows the MobaXterm interface with multiple terminal sessions open. The current session is showing the contents of the /etc/ssh/sshd_config file. Two specific sections of the configuration are highlighted with red boxes:

```
# Authentication:  
#LoginGraceTime 2m  
PermitRootLogin yes  
#StrictModes yes  
#MaxAuthTries 6  
#MaxSessions 10  
  
PubkeyAuthentication yes
```

These configurations enable root login via SSH and permit public key authentication.



The screenshot shows the MobaXterm interface with multiple terminal sessions open. The current session is showing the contents of the /etc/ssh/sshd_config file. A larger section of the configuration is highlighted with a red box:

```
#AuthorizedPrincipalsFile none  
  
#AuthorizedKeysCommand none  
#AuthorizedKeysCommandUser nobody  
  
# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts  
#HostbasedAuthentication no  
# Change to yes if you don't trust ~/.ssh/known_hosts for  
# HostbasedAuthentication  
#IgnoreUserKnownHosts no  
# Don't read the user's ~/.rhosts and ~/.shosts files  
#IgnoreRhosts yes  
-- INSERT --  
  
# To disable tunneled clear text passwords, change to no here!  
PasswordAuthentication yes  
#PermitEmptyPasswords no  
  
# Change to yes to enable challenge-response passwords (beware issues with  
# some PAM modules and threads)  
KbdInteractiveAuthentication no  
  
# Kerberos options  
#KerberosAuthentication no  
#KerberosOrLocalPasswd yes  
#KerberosTicketCleanup yes  
-- INSERT --
```

This configuration disables password authentication and enables challenge-response passwords (KBDInteractiveAuthentication).

Now get into the user devops using command “su – devops” and generate a ssh key using “ssh-keygen” command

The screenshot shows a terminal window titled "Master-host" running on a Windows host. The terminal session is for the root user on an Ubuntu 18.04 LTS server (ip-172-31-10-207). The user runs the command "ssh-keygen" to generate an SSH key pair. A red circle highlights the output of the command, which includes the creation of a directory at "/home/devops/.ssh", saving the private key as "id_ed25519", and saving the public key as "id_ed25519.pub". The key fingerprint is also displayed.

```
root@ip-172-31-10-207:/home/ubuntu# su - devops
devops@ip-172-31-10-207:~$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/devops/.ssh/id_ed25519):
Created directory '/home/devops/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/devops/.ssh/id_ed25519
Your public key has been saved in /home/devops/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:/s7NQBhzNQG0Vc+prpVOVteaNmMiqdgtfOf22vrH1Q devops@ip-172-31-10-207
The key's randomart image is:
+--[ED25519 256]--+
|       .o=.
|       + . o.
|       o o .E|
|       = . o|
|       S . . oo|
|       . . . +o.|
|       o ... **.|
|       o .o+o=**+.|
|       ..o==o*=B+.|
+---[SHA256]---+
devops@ip-172-31-10-207:~$
```

Add the other nodes into the hosts file /etc/ansible/hosts file with the private ip address

The screenshot shows a terminal window titled "Master-host" running on a Windows host. The terminal session is for the root user on an Ubuntu 18.04 LTS server (ip-172-31-10-207). The user is editing the "/etc/ansible/hosts" file. A red box highlights the configuration for three hosts: "prod-1" and "prod-2" under the "[production]" group, and "test-1" under the "[test]" group. The "monitor" host is also defined under the "[monitoring]" group. The "alpha" and "beta" hosts are listed as comments.

```
# Ex 1: Ungrouped hosts, specify before any group headers:
## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10

# Ex 2: A collection of hosts belonging to the 'webservers' group:
[production]
prod-1 ansible_host=172.31.11.215
prod-2 ansible_host=172.31.9.12

[test]
test-1 ansible_host=172.31.12.180

[monitoring]
monitor ansible_host=172.31.6.75

## [webservers]
## alpha.example.org
## beta.example.org
## 192.168.1.100
## 192.168.1.110
-- INSERT --
```

And now we have to copy the ssh into other nodes using ssh-copy-id devops@<private_ip> perform this to every node

The screenshot shows a terminal session titled "Master-host" running on a Windows host. The user is executing the command `ssh-copy-id devops@172.31.11.215`. The terminal output indicates that the source key is from `/home/devops/.ssh/id_ed25519.pub`. It shows the authenticity of the host cannot be established due to the lack of known fingerprints. The user is prompted to continue connecting with "yes". The command attempts to log in with the new key(s) and installs the remaining key(s). The password for the target host is requested. The message "Number of key(s) added: 1" is displayed. A note at the bottom instructs the user to try logging into the machine with `ssh 'devops@172.31.11.215'` and check if only the desired key(s) were added.

```
devops@ip-172-31-10-207:~/ssh$ ls id_ed25519.id_ed25519.pub
devops@ip-172-31-10-207:~/ssh$ ssh-copy-id devops@172.31.11.215
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/devops/.ssh/id_ed25519.pub"
The authenticity of host '172.31.11.215 (172.31.11.215)' can't be established.
ED25519 key fingerprint is SHA256:+s2+hXxUIC9KUJWLY04Gq8FC29IxJomFXU4dPZAphM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
devops@172.31.11.215's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'devops@172.31.11.215'"
and check to make sure that only the key(s) you wanted were added.

devops@ip-172-31-10-207:~/ssh$
```

The screenshot shows a terminal session titled "Master-host" running on a Windows host. The user is executing the command `ssh-copy-id devops@172.31.9.12`. The terminal output indicates that the source key is from `/home/devops/.ssh/id_ed25519.pub`. It shows the authenticity of the host cannot be established due to the lack of known fingerprints. The user is prompted to continue connecting with "yes". The command attempts to log in with the new key(s) and installs the remaining key(s). The password for the target host is requested. The message "Number of key(s) added: 1" is displayed. A note at the bottom instructs the user to try logging into the machine with `ssh 'devops@172.31.9.12'` and check if only the desired key(s) were added.

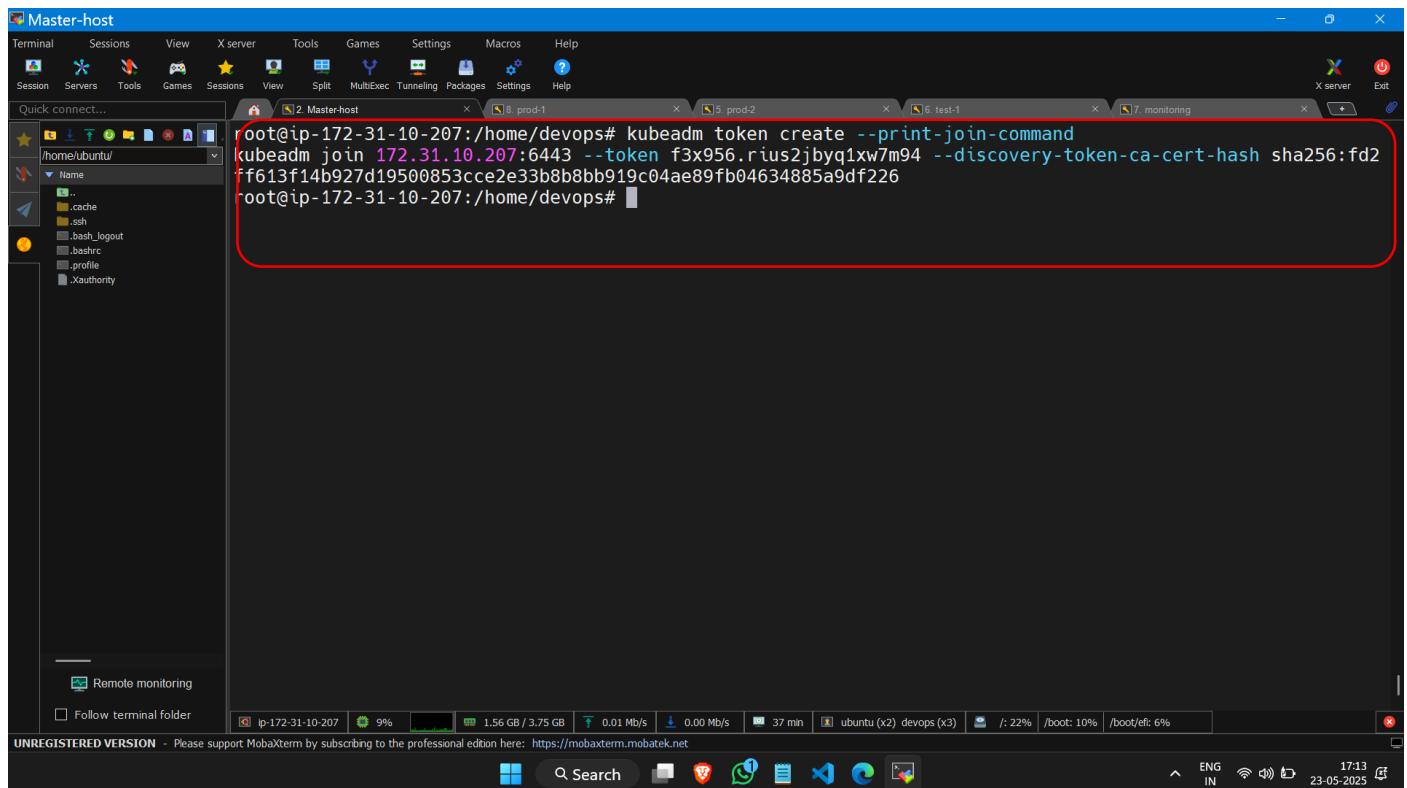
```
devops@ip-172-31-10-207:~/ssh$ ssh-copy-id devops@172.31.9.12
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/devops/.ssh/id_ed25519.pub"
The authenticity of host '172.31.9.12 (172.31.9.12)' can't be established.
ED25519 key fingerprint is SHA256:6ZmciaoA3Vedb41r00Wcu00V3Kjeui037SoCcRhVAxag.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
devops@172.31.9.12's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'devops@172.31.9.12'"
and check to make sure that only the key(s) you wanted were added.

devops@ip-172-31-10-207:~/ssh$
```

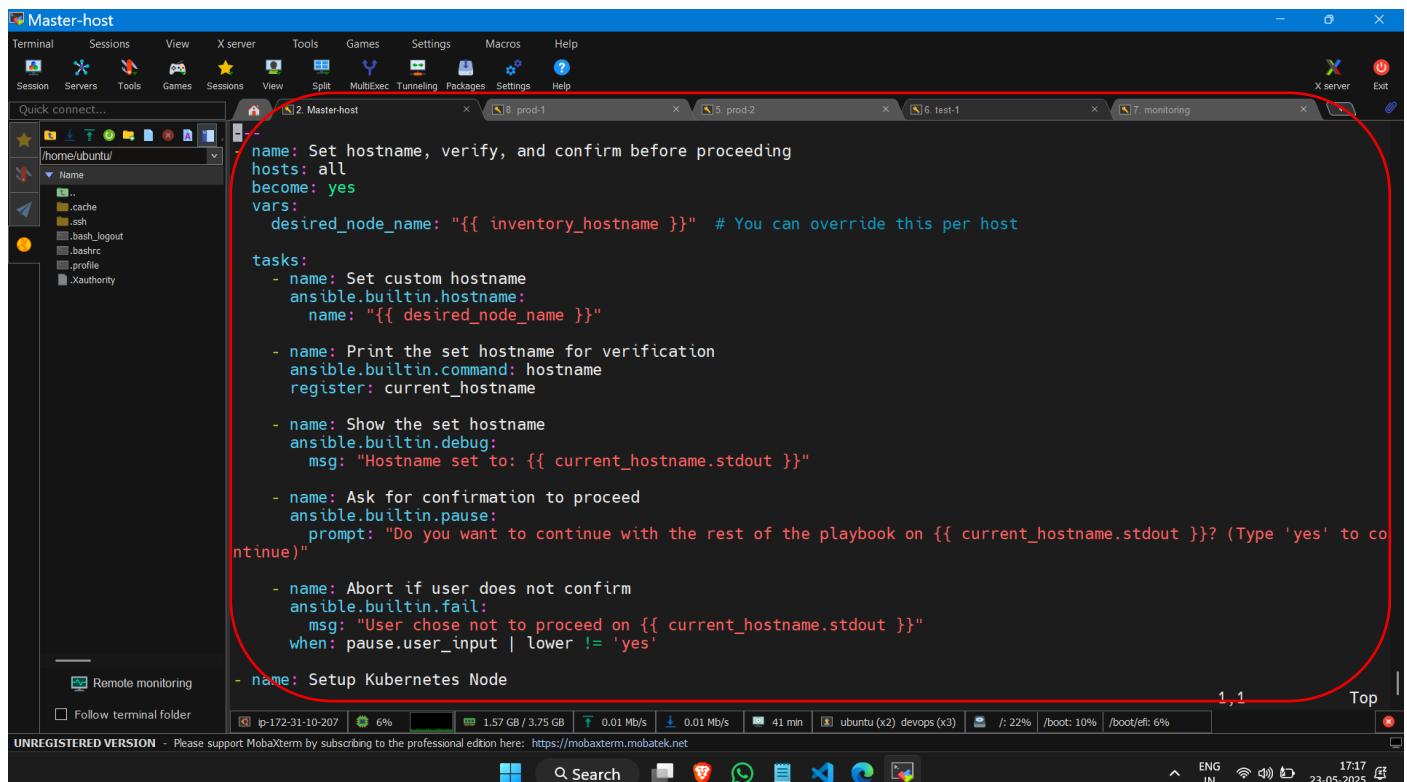
Now generate a Kubernetes join key in the master node using command “ kubeadm create –print-join-command ” and copy the token generated for further use



```
root@ip-172-31-10-207:/home/devops# kubeadm token create --print-join-command
kubeadm join 172.31.10.207:6443 --token f3x956.rius2jbyq1xw7m94 --discovery-token-ca-cert-hash sha256:fd2
f613f14b927d19500853cce2e33b8b8bb919c04ae89fb04634885a9df226
root@ip-172-31-10-207:/home/devops#
```

Now write a ansible playbook to connect the nodes with the kubernetes cluster master ans also set the hostname of each instance accordingly to the hosts file which has been written previously

This book will install the kubernetes slave configuration , network bridging and Kubernetes join token with cri socket



```
name: Set hostname, verify, and confirm before proceeding
hosts: all
become: yes
vars:
  desired_node_name: "{{ inventory_hostname }}" # You can override this per host
tasks:
  - name: Set custom hostname
    ansible.builtin.hostname:
      name: "{{ desired_node_name }}"
  - name: Print the set hostname for verification
    ansible.builtin.command: hostname
    register: current_hostname
  - name: Show the set hostname
    ansible.builtin.debug:
      msg: "Hostname set to: {{ current_hostname.stdout }}"
  - name: Ask for confirmation to proceed
    ansible.builtin.pause:
      prompt: "Do you want to continue with the rest of the playbook on {{ current_hostname.stdout }}? (Type 'yes' to continue)"
  - name: Abort if user does not confirm
    ansible.builtin.fail:
      msg: "User chose not to proceed on {{ current_hostname.stdout }}"
      when: pause.user_input | lower != 'yes'
  - name: Setup Kubernetes Node
```

```
msg: "User chose not to proceed on {{ current_hostname.stdout }}"
when: pause.user_input | lower != 'yes'

- name: Setup Kubernetes Node
  hosts: all
  become: yes
  tasks:
    - name: Download k8s-nodes.sh script
      ansible.builtin.get_url:
        url: https://raw.githubusercontent.com/akshu20791/Deployment-script/main/k8s-nodes.sh
        dest: /tmp/k8s-nodes.sh
        mode: '0777'

    - name: List files in /tmp
      ansible.builtin.command: ls /tmp
      register: ls_output

    - name: Show detailed files in /tmp
      ansible.builtin.command: ls -l /tmp
      register: ls_l_output

    - name: Ensure k8s-nodes.sh is executable
      ansible.builtin.file:
        path: /tmp/k8s-nodes.sh
        mode: '0777'

    - name: Show detailed files in /tmp again
      ansible.builtin.command: ls -l /tmp
      register: ls_l2_output

    - name: Run k8s-nodes.sh script
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

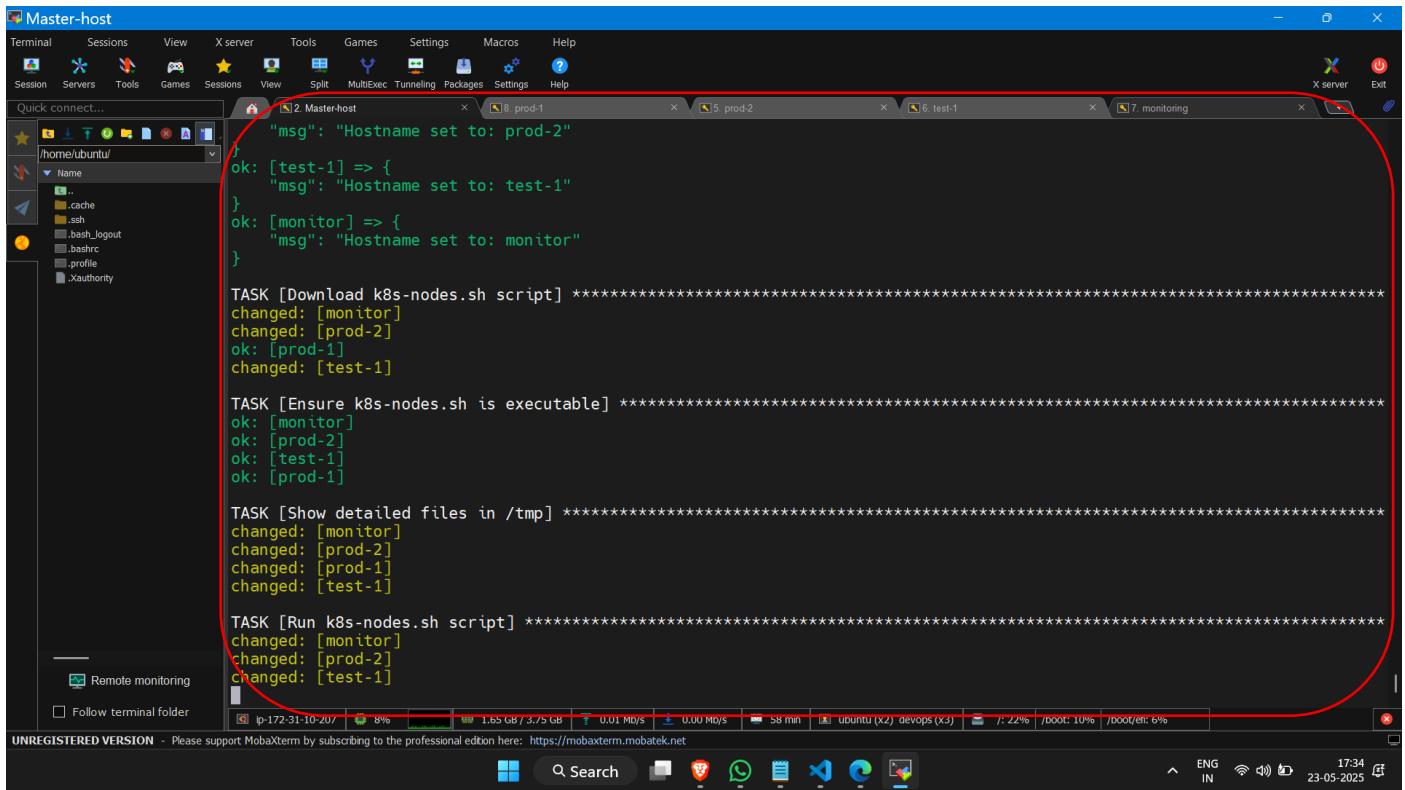
```
ok: [test-1]
TASK [Set custom hostname] *****
changed: [monitor]
changed: [prod-2]
changed: [prod-1]
changed: [test-1]

TASK [Print the set hostname for verification] *****
changed: [monitor]
changed: [prod-2]
changed: [prod-1]
changed: [test-1]

TASK [Show the set hostname] *****
ok: [prod-1] => {
    "msg": "Hostname set to: prod-1"
}
ok: [prod-2] => {
    "msg": "Hostname set to: prod-2"
}
ok: [test-1] => {
    "msg": "Hostname set to: test-1"
}
ok: [monitor] => {
    "msg": "Hostname set to: monitor"
}

TASK [Ask for confirmation to proceed] *****
[Ask for confirmation to proceed]
Do you want to continue with the rest of the playbook on prod-1? (Type 'yes' to continue):
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>



```
"msg": "Hostname set to: prod-2"
ok: [test-1] => {
    "msg": "Hostname set to: test-1"
}
ok: [monitor] => {
    "msg": "Hostname set to: monitor"
}

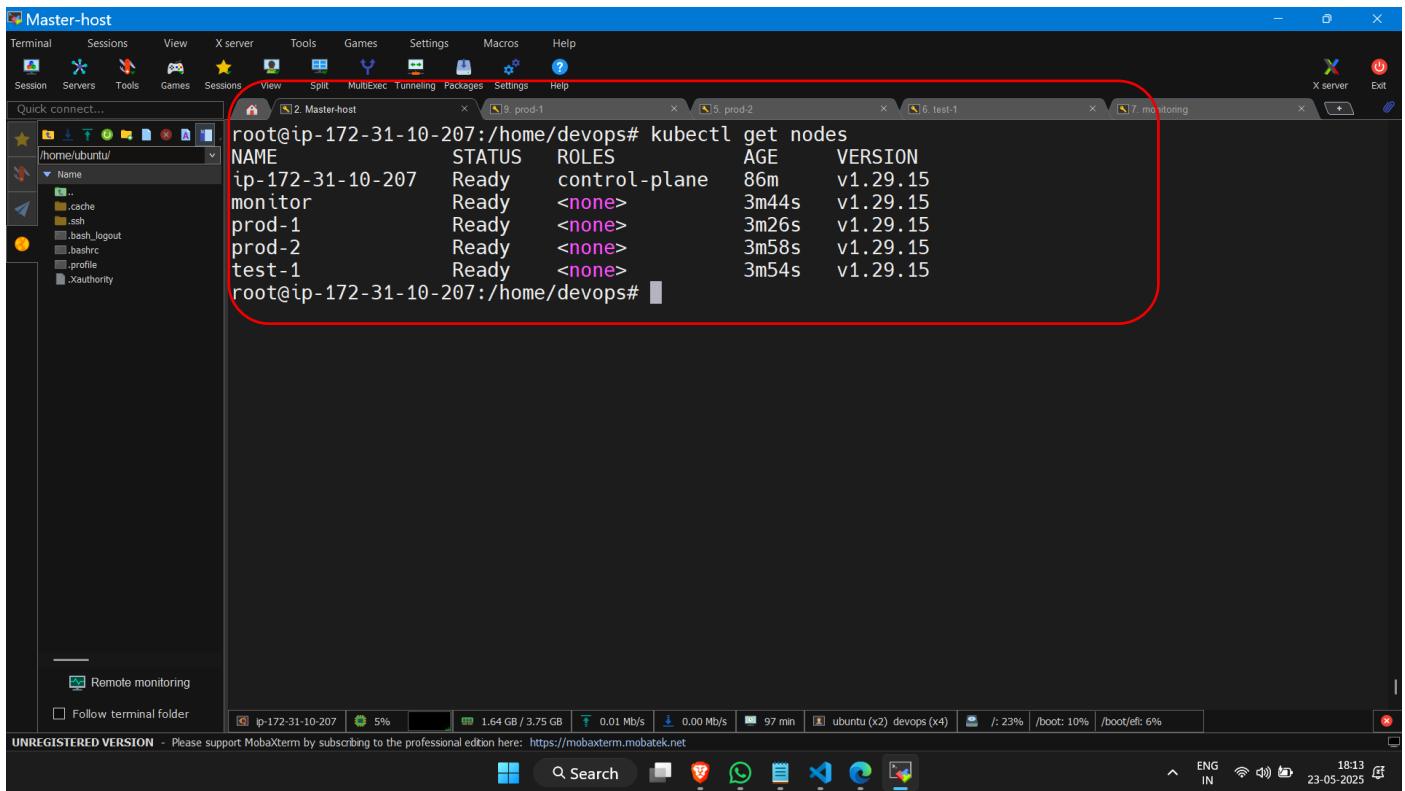
TASK [Download k8s-nodes.sh script] ****
changed: [monitor]
changed: [prod-2]
ok: [prod-1]
changed: [test-1]

TASK [Ensure k8s-nodes.sh is executable] ****
ok: [monitor]
ok: [prod-2]
ok: [test-1]
ok: [prod-1]

TASK [Show detailed files in /tmp] ****
changed: [monitor]
changed: [prod-2]
changed: [prod-1]
changed: [test-1]

TASK [Run k8s-nodes.sh script] ****
changed: [monitor]
changed: [prod-2]
changed: [test-1]
```

After successful run of ansible playbook check if the nodes got connected successfully by running the command “kubectl get nodes”



NAME	STATUS	ROLES	AGE	VERSION
ip-172-31-10-207	Ready	control-plane	86m	v1.29.15
monitor	Ready	<none>	3m44s	v1.29.15
prod-1	Ready	<none>	3m26s	v1.29.15
prod-2	Ready	<none>	3m58s	v1.29.15
test-1	Ready	<none>	3m54s	v1.29.15

Create kubernetes namespace in order to separate the nodes using the namespace using command “`kubectl create namespace <namespace-name>`” here I have created 3 namespace test, monitoring, production and verify them by command “`kubectl get namespace`”

```
root@ip-172-31-10-207:/home/devops# kubectl get namespace
NAME        STATUS   AGE
default     Active   87m
kube-node-lease Active  87m
kube-public  Active   87m
kube-system  Active   87m
monitoring  Active   13s
production  Active   13s
test        Active   13s
root@ip-172-31-10-207:/home/devops#
```

Now set the labels each node using env tag using command “`kubectl label nodes <node-name> env=<name>`” here I have set 3 env i.e prod, test, & monitoring

Check the labels using command `kubectl get nodes --show-labels`”

```
root@ip-172-31-10-207:/home/devops#
root@ip-172-31-10-207:/home/devops# kubectl label nodes prod-1 env=prod
kubectl label nodes prod-2 env=prod
kubectl label nodes test-1 env=test
kubectl label nodes monitor env=monitoring
node/prod-1 labeled
node/prod-2 labeled
node/test-1 labeled
node/monitor labeled
root@ip-172-31-10-207:/home/devops# kubectl get nodes --show-labels
NAME        STATUS   ROLES      AGE    VERSION   LABELS
ip-172-31-10-207 Ready    control-plane 93m    v1.29.15   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=ip-172-31-10-207,kubernetes.io/os=linux,node-role.kubernetes.io/control-plane=,node.kubernetes.io/exclude-from-external-load-balancers=
monitor     Ready    <none>     11m    v1.29.15   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,env=monitoring,kubernetes.io/arch=amd64,kubernetes.io/hostname=monitor,kubernetes.io/os=linux
prod-1      Ready    <none>     10m    v1.29.15   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,env=prod,kubernetes.io/arch=amd64,kubernetes.io/hostname=prod-1,kubernetes.io/os=linux
prod-2      Ready    <none>     11m    v1.29.15   beta.kubernetes.io/arch=amd64,kubernetes.io/hostname=prod-2,kubernetes.io/os=linux
test-1      Ready    <none>     11m    v1.29.15   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux
```

Now verify the nodes status using command “kubectl get nodes -o wide”

The screenshot shows a terminal window in MobaXterm titled "master-host". The command "kubectl get nodes -o wide" has been run, and the output is displayed in a table:

NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE	KERNEL-VERSION	CONTAINER-RUNTIME
ip-172-31-10-207	Ready	control-plane	98m	v1.29.15	172.31.10.207	<none>	Ubuntu 24.04.2 LTS	6.8.0-1024-aws	docker://28.1.1
monitor	Ready	<none>	15m	v1.29.15	172.31.6.75	<none>	Ubuntu 24.04.2 LTS	6.8.0-1024-aws	docker://28.1.1
prod-1	Ready	<none>	15m	v1.29.15	172.31.11.215	<none>	Ubuntu 24.04.2 LTS	6.8.0-1024-aws	docker://28.1.1
prod-2	Ready	<none>	15m	v1.29.15	172.31.9.12	<none>	Ubuntu 24.04.2 LTS	6.8.0-1024-aws	docker://28.1.1
test-1	Ready	<none>	15m	v1.29.15	172.31.12.180	<none>	Ubuntu 24.04.2 LTS	6.8.0-1024-aws	docker://28.1.1

The terminal also shows the prompt "root@ip-172-31-10-207:/home/devops#".

Now access the Jenkins service using the public ip on port 8080

The screenshot shows a web browser window with the address bar displaying "Not secure 3.110.108.195:8080". The main content is the Jenkins "Getting Started" page, which includes a table of available plugins and a note about required dependencies.

Folders	Formatter		
Timestamper	Workspace Cleanup	Ant	Gradle
Pipeline	Github Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline Graph View
Git	SSH Build Agents	Matrix Authorization Strategy	PAM Authentication
LDAP	Email Extension	Mailer	Dark Theme

On the right side of the table, there is a sidebar with the heading "Folders" and a list containing "OWASP Markup Formatter". A note at the bottom of the sidebar states "** - required dependency".

At the bottom of the page, it says "Jenkins 2.504.1".

Login into the Jenkins using initial password which will get at path
/var/jenkins_home/secrets/initialAdminPassword

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job +

Set up a distributed build

Set up an agent

Configure a cloud

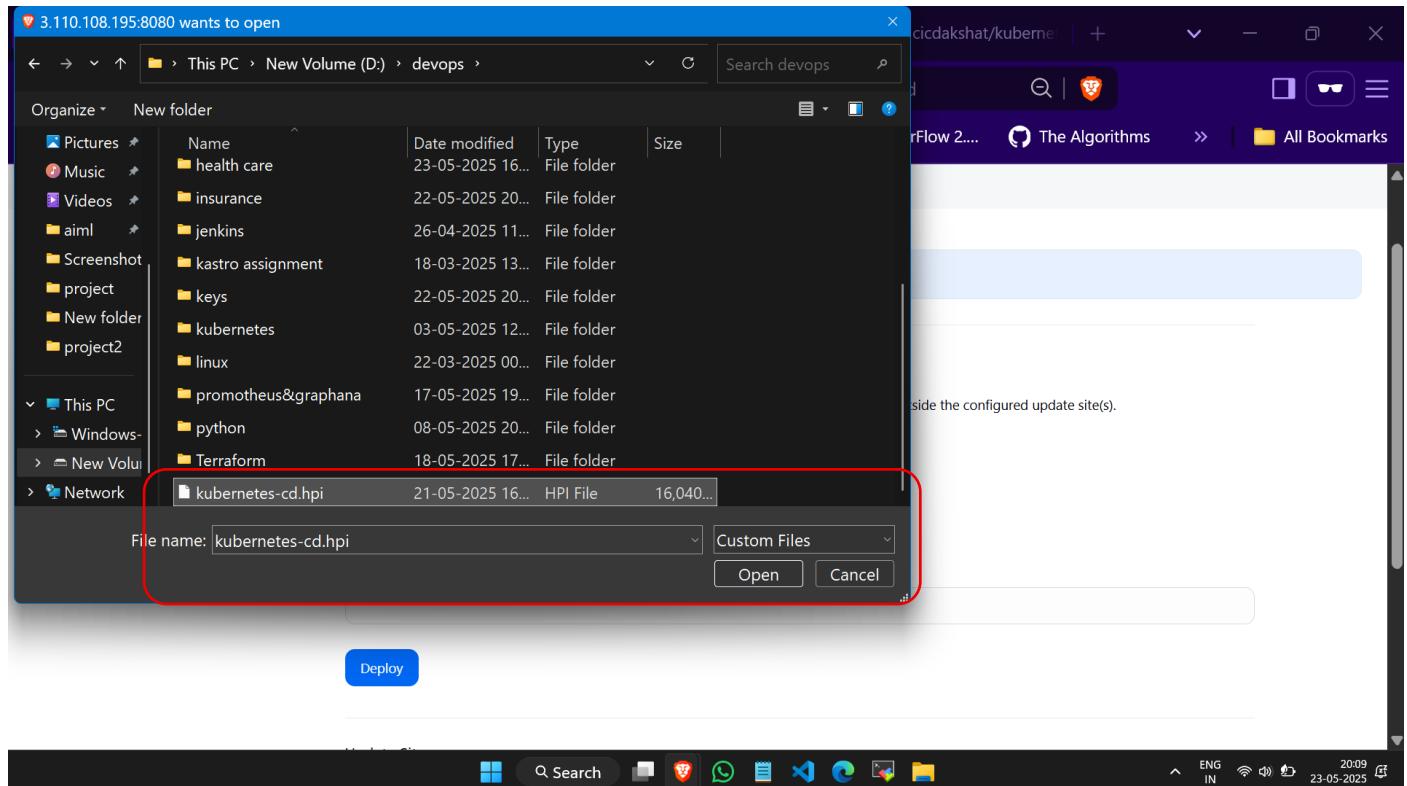
Learn more about distributed builds ?

Navigate to manage Jenkins -> plugins ->available plugins -> docker install docker plugins

Install	Name	Released
<input checked="" type="checkbox"/>	Docker 1274.vcd203fd12e74	Cloud Providers Cluster Management docker 2 mo 16 days ago
<input type="checkbox"/>	Docker Commons 451.vd12c371eeeb_3	Library plugins (for use by other plugins) docker Provides the common shared functionality for various Docker-related plugins. 2 mo 13 days ago
<input checked="" type="checkbox"/>	Docker Pipeline 611.v16e84da_6d3ff	pipeline DevOps Deployment docker Build and use Docker containers from pipelines. 2 mo 19 days ago
<input type="checkbox"/>	Docker API 3.50-108.v211cdd21c383	plugins (for use by other plugins) docker 1 mo 21 days ago

Download Kubernetes plugin the HPI file manually from the repository
["https://github.com/akshu20791/cicdakshat"](https://github.com/akshu20791/cicdakshat) and load it manually into the jenkins

Navigate to manage Jenkins -> plugins ->advanced setting -> then upload the hpi file and install it



Run this commands in the master machine and get the configid

```
cd ~  
ls -a  
cd .kube  
ls  
cat config
```

copy the whole config id which has came up for furhter adding it to jenkins credentials

The screenshot shows a terminal window in MobaXterm titled "master-host". The command "ls -a" has been run, listing various files in the current directory, including a large file named "kubeconfig" which is highlighted with a red box. The "kubeconfig" file contains the following JSON-like content:

```
ls -a
cd .kube
ls
cat config
.ansible .bashrc .launchpadlib .local .ssh .wget-hsts
.bash_history .kube .lessht .profile .viminfo snap
cache config
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0JYnZ6blkKQ3Ur0UV3RFFZSkvWklodmNQQVFFTEJRQXdGVEVUTUJFR0ExVUUQXhNS2EzVmLaWEp1WlhSbGN6QVGdzB5TlRBMU1qRxDPVEV5TVRCYUZ3MhpOVEExTVRrd09URTNNVEjhTUJVeApFekFSQmd0VkJBTVRDbXQxW1WeWJtVjBaWE13Z2dFaU1BMEdDU3FHU01iM0RRRUJBUVVBQTRJQkr3QXdnZ0VLckFvSUJBURJWWYrbEtVNC9YWhnlA2cytnV2MzM0lnaohhVFkzbWxZY28xNkFwdHlrQ0oz0UFqWkZ3adVNvmkKK3BBK3dwNlhpTXRuRFNsMXF0SzQwSWExTm9yekp5N0ExQUJUMUx5SIa0T0tRZXkzaTuya2d0U0uyZXlCdk1tMwp5Uk9sNVJcjlFLSuaxLyteGeU1CVElnQ05wcWM0dUJBT1M2eStld0xsNHNRWllCeFVxdzhUV0Rkd94RnN5Qy85CnRKV21Ld3M1dWozNTVXZXPdWPa3FhckZyQW5hcEJ1c1BEWThNMUpCbFQxMkFPMTE4LyteSeHdyT0dPSzF1MlgKM0grZDZhbjZnYWZBWkFKNyt6M0hQSXFIMLBdcEFKbzRUMzBmcmycTTNnV3YyQWY2cVVYc0l4S3gyS0NEUVk3SwpsZ0w00DJY0TJZZEtvrzl1YjIySEc4S1R1amRIQWdNQkFBR2pXVEJYTUE0R0ExVWREd0VCL3dRRUF3SUNwREFQCKJnTLZIUk1C0WY4RUJUQRBUUgvTU1wR0ExVwREZ1FXQkJRRUpadZ4SFN2Rmwza1RUQVJX0EFMcFLicCtuQVYKQmd0VkhSRUVEakFNZ2dwcmRXSmjbTVsZEdWek1BMEdDu3FU01iM0RRRUJD1VBQTRJQkFRQ3KXYXVIMTvrNworVnEzaXJWek56M01YdjYzTm9QeVdIBhg3T1BhdkQwSjNjcZNRbktTd2VxaVYvSW9mZkd2SktUdnluMkh1eXcxCk4rN3ZYVmtmS09mV1dEVzfZY3E5UEpKVWhSwEhzTTB0dEg5SEtzeWdHNUM0S0Z0ajNhdkZtUhdcWXcYUZVSDUkmfZGL0I3QkjzaU15UWLmMDRMdgZ5VHNLL3RBenzJ0TBQMThYZnhFVGs3Q1lD0WphZ3F1eVQ4djJ0MEh6SkczcwpoQ2hqMzY1VDRQn2J4c0
```

The screenshot shows a web browser window titled "New credentials - Jenkins". The URL is "cicidakshat/kubernetes-cd.hpi at master · Jenkins". The page displays the Jenkins credentials management interface. A specific credential entry is highlighted with a red box, showing the following details:

- Kubernetes configuration (kubeconfig)**: The title of the credential.
- ID**: The identifier for the credential, set to "k8screds".
- Description**: A blank field for a description.
- Kubeconfig**: The configuration type, with "Enter directly" selected.
- Content**: The kubeconfig content, identical to the one shown in the MobaXterm terminal above.

A "Create" button is visible at the bottom left of the credential form.

In the credentials add the docker credentials (dockercreds) and kubernetes credentials (k8screds)

The screenshot shows the Jenkins Global credentials (unrestricted) page. A red box highlights the table where two credentials are listed:

ID	Name	Kind	Description
dockercreds	sachinbacha/*****	Username with password	
k8screds	k8screds	Kubernetes configuration (kubeconfig)	

Below the table are size options: S, M, L.

At the bottom right of the page, it says REST API Jenkins 2.504.1.

Now create a new item with name Healthcare and select the item type as pipeline

The screenshot shows the Jenkins New Item creation page. A red box highlights the 'Enter an item name' field, which contains 'Healthcare'. Another red box highlights the 'Pipeline' item type, which is described as 'Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.' A third red box highlights the 'OK' button at the bottom.

Add the github repository in the general configurations add the forked repository

The screenshot shows the Jenkins configuration page for a job named "Healthcare". The "General" tab is selected. Under the "GitHub project" section, the "Project url" field contains the URL "https://github.com/bachasachin0/star-agile-health-care". This entire section is highlighted with a red box.

Configuration tabs: General, Triggers, Pipeline, Advanced.

General settings checkboxes:

- Discard old builds ?
- Do not allow concurrent builds
- Do not allow the pipeline to resume if the controller restarts

GitHub project settings:

- GitHub project
- Project url: https://github.com/bachasachin0/star-agile-health-care

Advanced settings checkboxes:

- Pipeline speed/durability override ?
- Preserve stashes from completed builds ?
- This project is parameterized ?
- Throttle builds ?

Buttons: Save, Apply.

Add git triggers

The screenshot shows the Jenkins configuration page for the same "Healthcare" job. The "Triggers" tab is selected. Under the "Poll SCM" section, the "Schedule" field contains the cron expression "H * * * *". This entire section is highlighted with a red box.

Configuration tabs: General, Triggers, Pipeline, Advanced.

Triggers settings checkboxes:

- Build after other projects are built ?
- Build periodically ?
- GitHub hook trigger for GITScm polling ?
- Poll SCM ?

Schedule: H * * * *

Text: No schedules so will only run due to SCM changes if triggered by a post-commit hook

Ignore post-commit hooks ?

Trigger builds remotely (e.g., from scripts) ?

Buttons: Save, Apply.

Write the Dockerfile

The screenshot shows a GitHub repository page for 'star-agile-health-care'. A red box highlights the 'Dockerfile' file in the 'Code' tab. The code content is as follows:

```
FROM openjdk:11
ARG JAR_FILE=target/*.jar
COPY ${JAR_FILE} app.jar
EXPOSE 9090
ENTRYPOINT ["java", "-jar", "/app.jar", "--server.port=9090"]
```

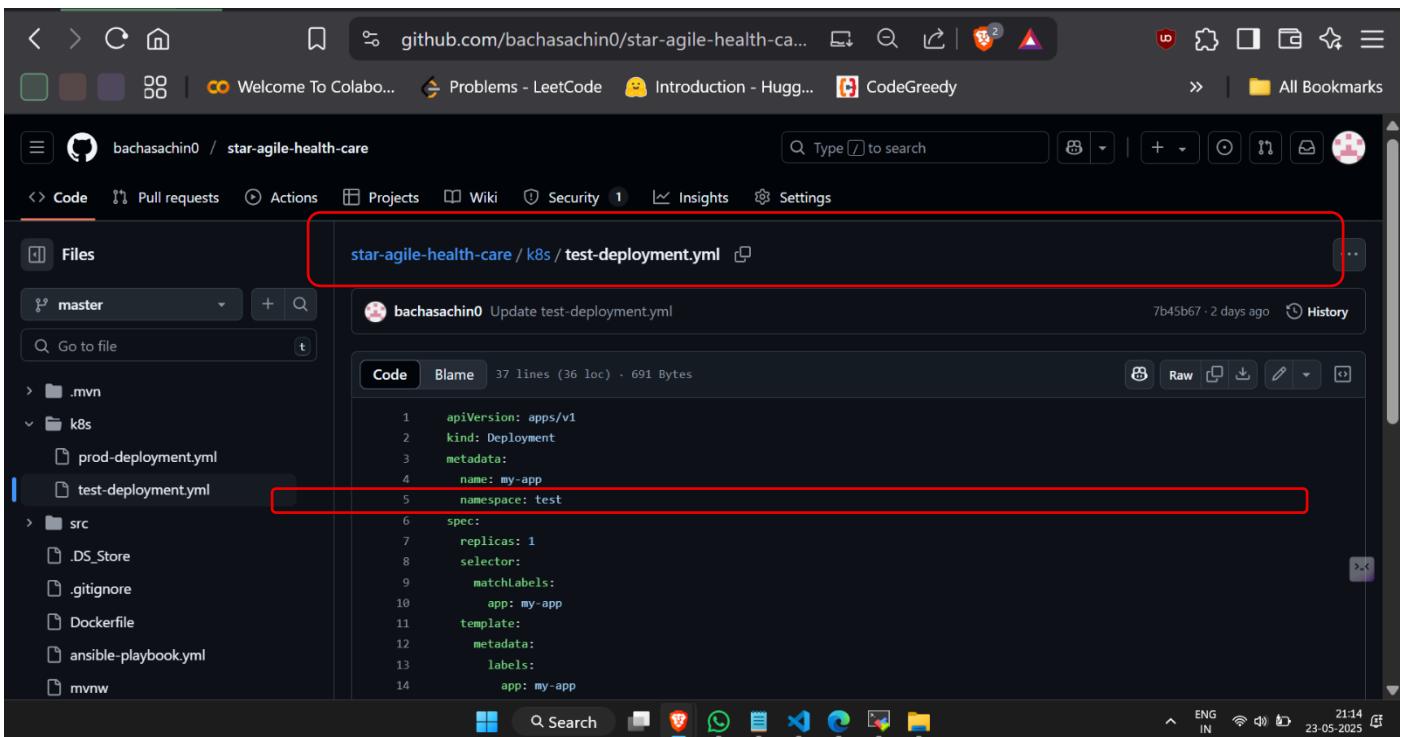
Create a k8s folder and write the deployment scripts of both production and test deployment with namespace production and tag prod

This will ensure that the applications are deployed on the assigned tags and namespace

The screenshot shows a GitHub repository page for 'star-agile-health-care'. A red box highlights the 'prod-deployment.yml' file in the 'k8s' folder. The code content is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mv-app
  namespace: production
spec:
  replicas: 2
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
```

Also write the test deployment script with the namespace test and tag env of test this will ensure that the application is deployed on the specified nodes only



The screenshot shows a GitHub code editor interface. A red box highlights the file path 'star-agile-health-care / k8s / test-deployment.yml'. Another red box highlights the line 'namespace: test' in the YAML code.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
  namespace: test
spec:
  replicas: 1
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
```

Write the Jenkins script to deploy the application

```
pipeline {
  agent any

  stages {
    stage('Checkout Code') {
      steps {
        git url: 'https://github.com/bachasachino/star-agile-health-care.git', branch: 'master'
      }
    }

    stage('Build with Maven') {
      steps {
        sh './mvnw clean package'
      }
    }

    stage('Build Docker Image') {
      steps {
        sh 'docker build -t sachinbacha/medicure:latest .'
      }
    }
  }
}
```

```

stage('Push Docker Image') {
    steps {
        withCredentials([usernamePassword(credentialsId: 'dockercreds', usernameVariable: 'USERNAME',
passwordVariable: 'PASSWORD')]) {
            sh 'echo $PASSWORD | docker login -u $USERNAME --password-stdin'
            sh 'docker push sachinbacha/medicure:latest'
        }
    }
}

stage('Deploy to Test') {
    steps {
        kubernetesDeploy(
            kubeconfigId: 'k8spasswd',
            configs: 'k8s/test-deployment.yml'
        )
    }
}

stage('Wait for Test App to Be Ready') {
    steps {
        sh 'sleep 10'
    }
}

stage('Run Unit Tests') {
    steps {
        sh './mvnw test'
    }
}

stage('Check Build Status') {
    steps {
        script {
            echo "Current build result: ${currentBuild.result}"
        }
    }
}

stage('Publish JUnit Test Results') {
    steps {
        junit 'target/surefire-reports/TEST-*.xml'
    }
}

stage('Deploy to Production') {

```

```
when {
    expression { currentBuild.result == null || currentBuild.result == 'SUCCESS' }
}

steps {
    kubernetesDeploy(
        kubeconfigId: 'k8spasswd',
        configs: 'k8s/prod-deployment.yml'
    )
}

stage('Print App URLs') {
    steps {
        sh """
            echo "👉 Test App: http://3.110.43.250:30090"
            echo "👉 Production App: http://3.110.43.250:30091"
        """
    }
}

post {
    success {
        echo "✅ Pipeline completed successfully!"
    }
    failure {
        echo "✖ Pipeline failed. Check the logs above."
    }
}
```

Run the Jenkins pipeline

The screenshot shows a Jenkins pipeline status page for a "Healthcare" application. The top navigation bar includes tabs for "Healthcare - Jenkins", "Medicure", and "Medicure". The main content area has a summary box for "Healthcare" stating it's a CICD pipeline for a Healthcare application using Kubernetes. Below this is a "Test Result Trend" chart showing a single green bar for "Passed". The "Stage View" table provides a detailed breakdown of build times for various stages: Checkout Code (1s), Build with Maven (17s), Build Docker Image (4s), Push Docker Image (15s), Deploy to Test (574ms), Wait for Test App to Be Ready (3s), Run Unit Tests (7s), Check Build Status (137ms), Publish JUnit Test Results (126ms), Deploy to Production (347ms), Print App URLs (219ms), and Declarative: Post Actions (109ms). The bottom of the page includes a "Builds" section, a "Filter" search bar, and a footer with system status icons and the date 23-May-2025.

Add github webhook to the repository from settings and add the Jenkins ip address link

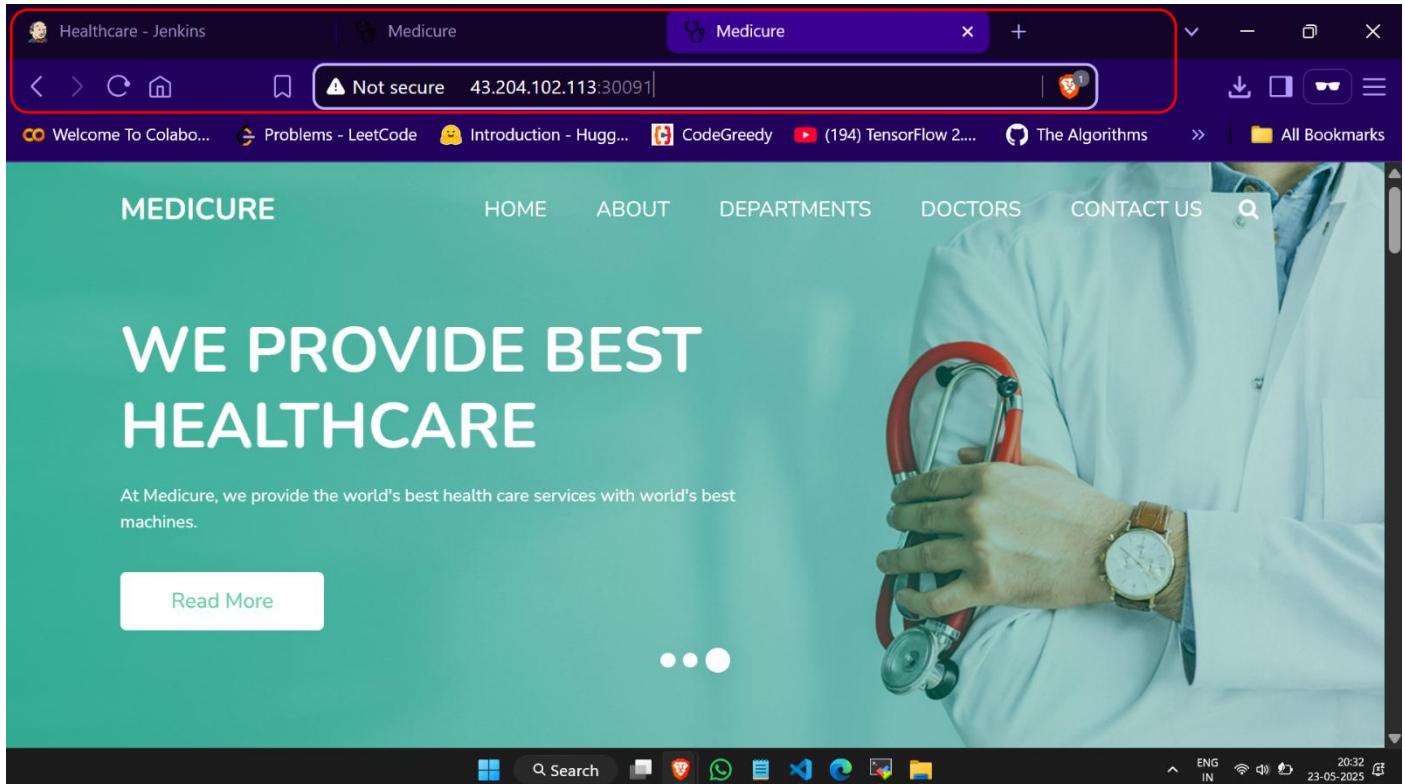
The screenshot shows the GitHub Settings page for managing webhooks. The left sidebar lists various settings categories: General, Access, Collaborators, Moderation options, Code and automation, Branches, Tags, Rules, Actions, Models, Webhooks (which is selected and highlighted in blue), Environments, Codespaces, and Pages. The main content area is titled "Webhooks / Manage webhook". It has two tabs: "Settings" (selected) and "Recent Deliveries". A descriptive text box explains that GitHub will send a POST request to the specified URL with event details. Below it, the "Payload URL *" input field contains the value "http://3.110.198.195:8080/github-webhook|", which is highlighted with a red rectangle. Other fields include "Content type *" set to "application/json", "Secret" (empty), and "SSL verification" (checkbox checked). The bottom navigation bar includes links for Code, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings.

A screenshot of a GitHub repository settings page. The repository is named 'bachasachin0 / star-agile-health-care'. The 'Webhooks' section is highlighted with a red box. It contains a single entry for a webhook pointing to 'http://3.110.108.195:8080/github-w... (push)'. A note below says 'This hook has never been triggered.' There are 'Edit' and 'Delete' buttons next to the entry.

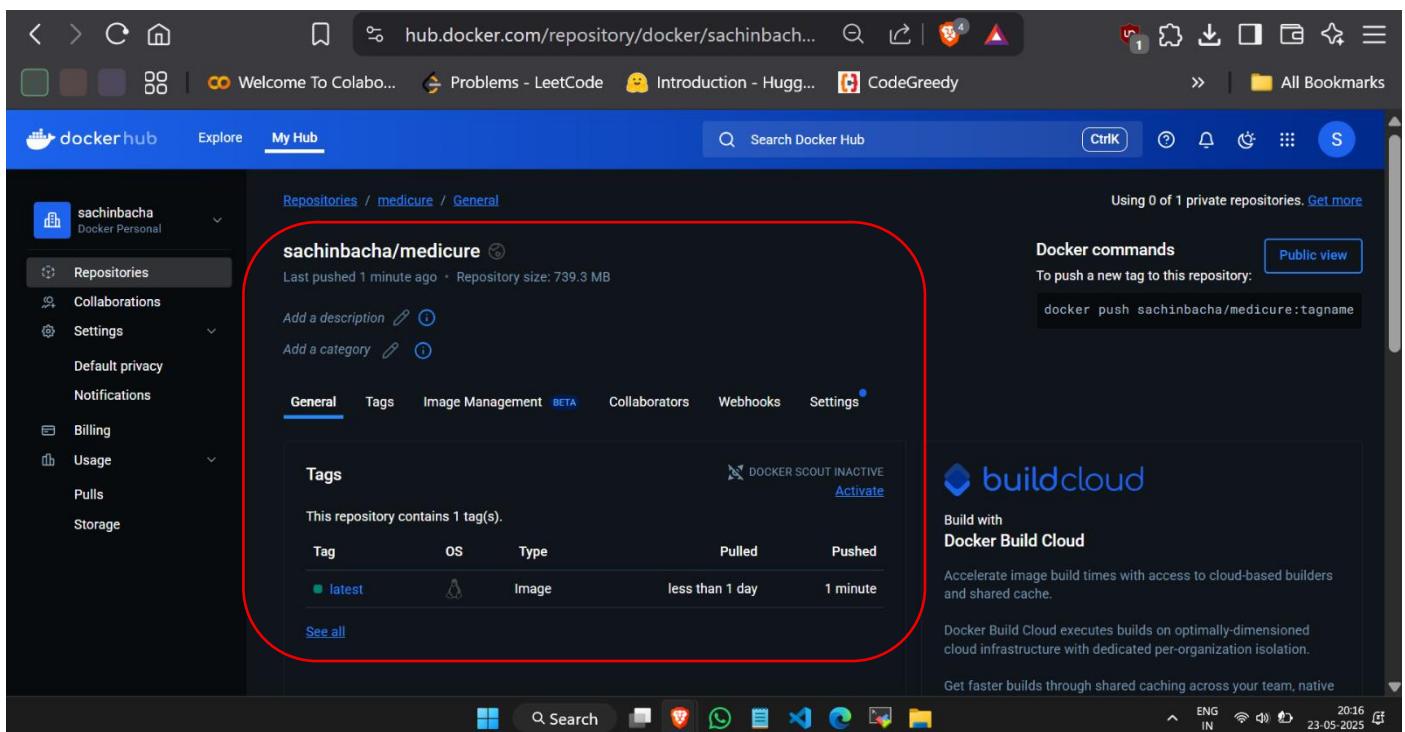
After that access the application on both the production servers using the public ip address on port number 30091

A screenshot of a web browser window showing the 'Medicure' homepage. The URL in the address bar is 'Not secure 13.127.120.65:30091'. The page features a large banner with the text 'WE PROVIDE BEST HEALTHCARE' and a doctor's image holding a stethoscope. A 'Read More' button is visible. The browser interface includes a search bar, a taskbar at the bottom with various icons, and a system tray at the bottom right.

Access the application on the other server also



Check the docker hub if the image got created correctly

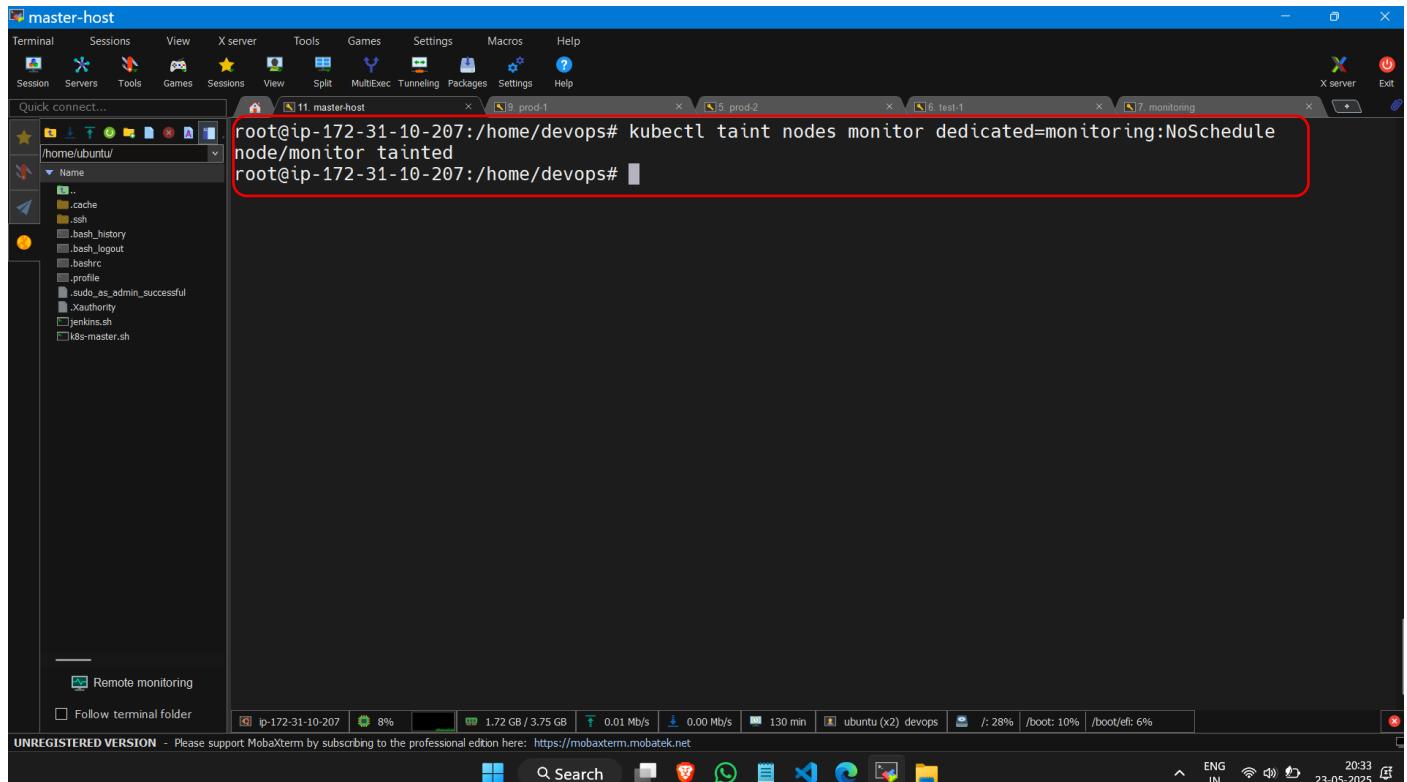


Prometheus and Grafana setup

Now in master machine

Taint the node monitoring which will separate the node from other nodes will be logically separated and no scheduling is done on that node

Using command “kubectl taint nodes <node_name> dedicated=<namespace>:NoSchedule

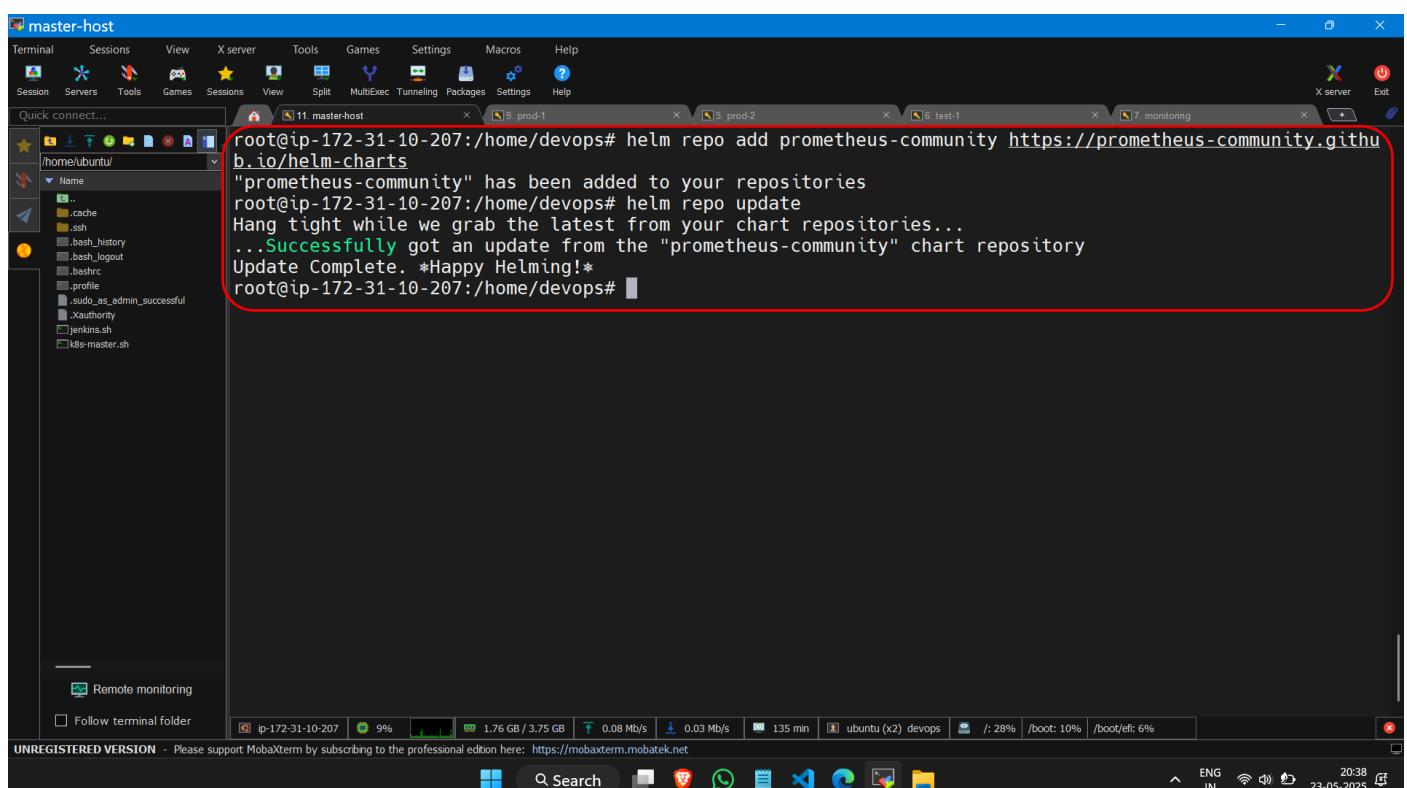


The screenshot shows a terminal window in MobaXterm titled "master-host". The command "kubectl taint nodes monitor dedicated=monitoring:NoSchedule" is being run, resulting in the output "node/monitor tainted". The entire command line and its output are highlighted with a red box.

```
root@ip-172-31-10-207:/home/devops# kubectl taint nodes monitor dedicated=monitoring:NoSchedule
node/monitor tainted
root@ip-172-31-10-207:/home/devops#
```

Install helm using command “curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash”

Install helm community repository and update it



The screenshot shows a terminal window in MobaXterm titled "master-host". The command "helm repo add prometheus-community https://prometheus-community.github.io/helm-charts" is run, followed by "helm repo update". The output indicates the repository was added and updated successfully. The entire command line and its output are highlighted with a red box.

```
root@ip-172-31-10-207:/home/devops# helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
"prometheus-community" has been added to your repositories
root@ip-172-31-10-207:/home/devops# helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. *Happy Helming!*
root@ip-172-31-10-207:/home/devops#
```

Create a monitoring-values.yaml to add tolerations so pods get scheduled only on just **monitoring node**

```
prometheus:
  maximumStartupDurationSeconds: 300    # Must be >= 60
  tolerations:
    - key: "dedicated"
      operator: "Equal"
      value: "monitoring"
      effect: "NoSchedule"

grafana:
  tolerations:
    - key: "dedicated"
      operator: "Equal"
      value: "monitoring"
      effect: "NoSchedule"

nodeExporter:
  tolerations:
    - key: "dedicated"
      operator: "Equal"
      value: "monitoring"
      effect: "NoSchedule"

kubeStateMetrics:
  tolerations:
    - key: "dedicated"
```

Install Prometheus kube stack using helm

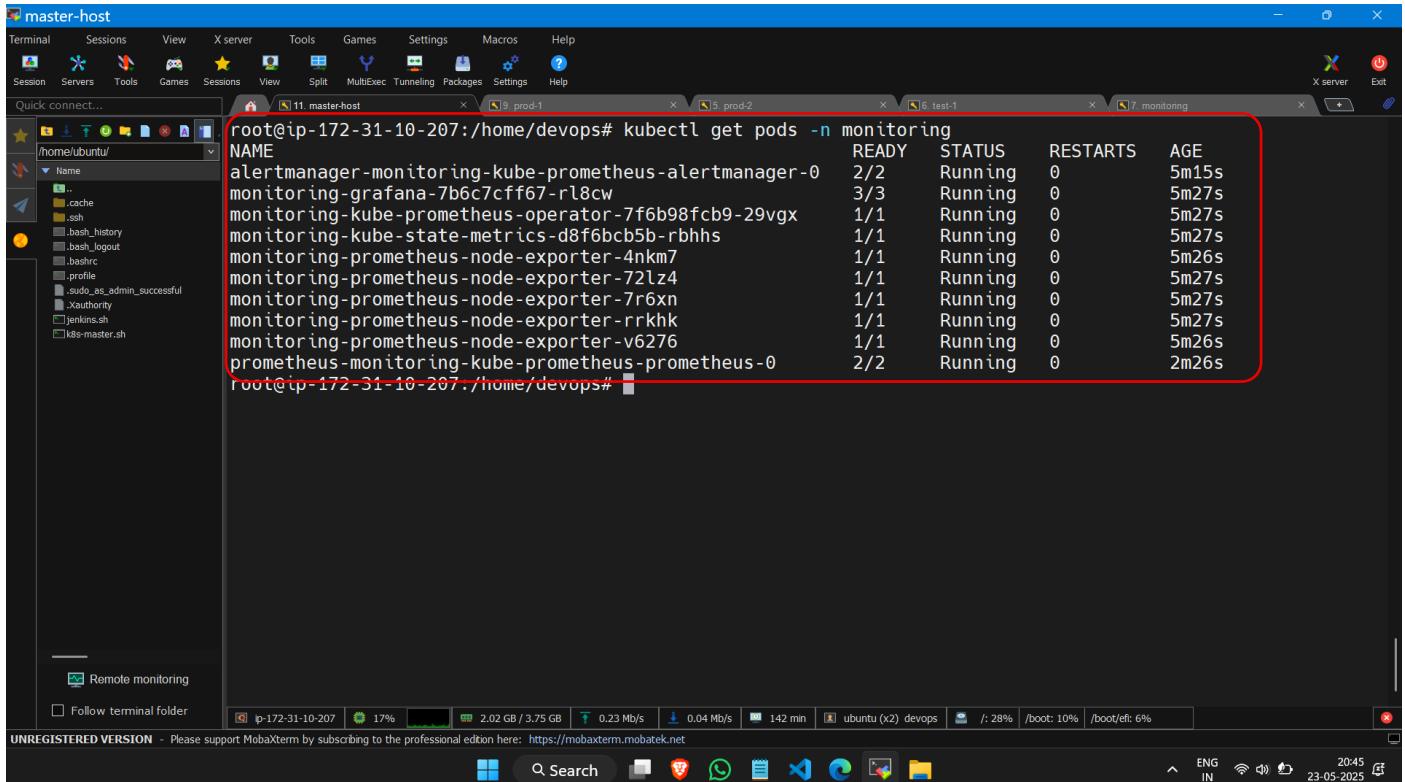
```
root@ip-172-31-10-207:/home/devops# helm upgrade monitoring prometheus-community/kube-prometheus-stack \
-n monitoring -f monitoring-values.yaml
Release "monitoring" has been upgraded. Happy Helming!
NAME: monitoring
LAST DEPLOYED: Fri May 23 15:12:22 2025
NAMESPACE: monitoring
STATUS: deployed
REVISION: 2
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --namespace monitoring get pods -l "release=monitoring"

Get Grafana 'admin' user password by running:
  kubectl --namespace monitoring get secrets monitoring-grafana -o jsonpath="{.data.admin-password}" | base64 -d ; echo

Access Grafana local instance:
  export POD_NAME=$(kubectl --namespace monitoring get pod -l "app.kubernetes.io/name=grafana,app.kubernetes.io/instance=monitoring" -oname)
  kubectl --namespace monitoring port-forward $POD_NAME 3000

Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configure Alertmanager and Prometheus instances using the Operator.
root@ip-172-31-10-207:/home/devops# ^C
root@ip-172-31-10-207:/home/devops#
```

Now get the pods running in the monitoring namespace i.e the pods will now run on just monitoring node and verify if the nodes are running

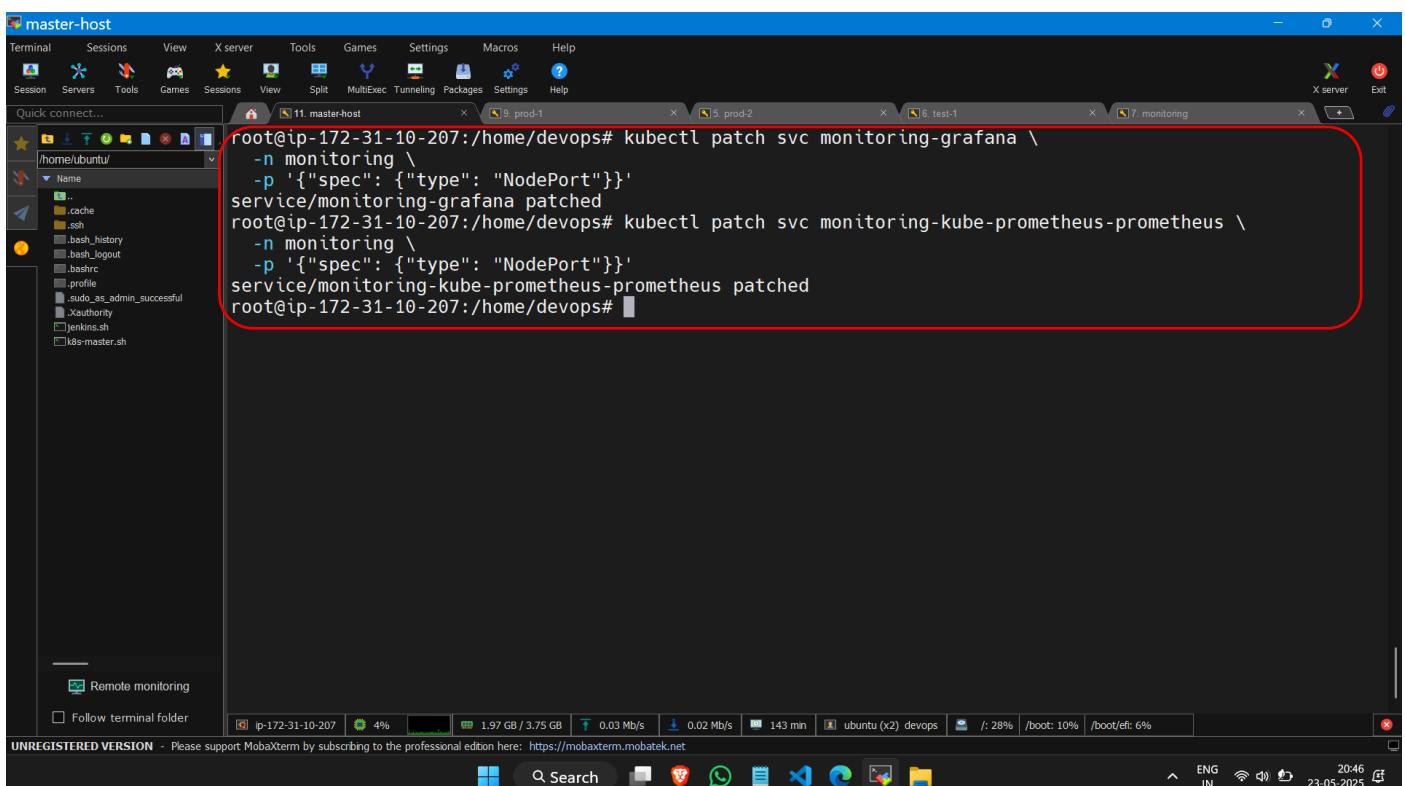


```
root@ip-172-31-10-207:/home/devops# kubectl get pods -n monitoring
NAME                                         READY   STATUS    RESTARTS   AGE
alertmanager-monitoring-kube-prometheus-alertmanager-0   2/2     Running   0          5m15s
monitoring-grafana-7b6c7cff67-r18cw           3/3     Running   0          5m27s
monitoring-kube-prometheus-operator-7f6b98fcb9-29vgx   1/1     Running   0          5m27s
monitoring-kube-state-metrics-d8f6bcb5b-rbhhs      1/1     Running   0          5m27s
monitoring-prometheus-node-exporter-4nk7          1/1     Running   0          5m26s
monitoring-prometheus-node-exporter-72lz4         1/1     Running   0          5m27s
monitoring-prometheus-node-exporter-7r6xn        1/1     Running   0          5m27s
monitoring-prometheus-node-exporter-rrkhk        1/1     Running   0          5m27s
monitoring-prometheus-node-exporter-v6276       1/1     Running   0          5m26s
prometheus-monitoring-kube-prometheus-prometheus-0  2/2     Running   0          2m26s
root@ip-172-31-10-207:/home/devops#
```

Expose Grafana to nodeport to access it through the internet (remote access)

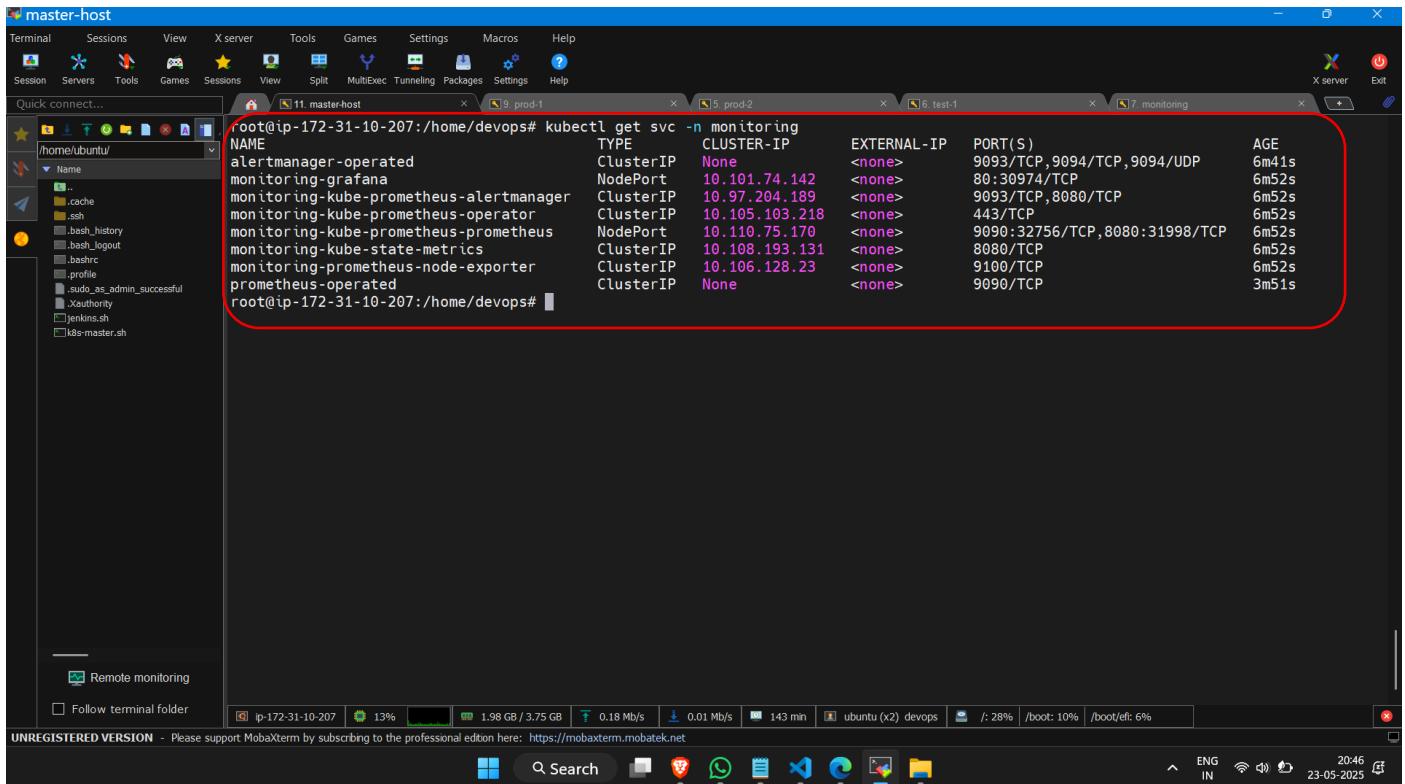
run ["kubectl patch svc monitoring-grafana \ -n monitoring \ -p '{\"spec\": {\"type\": \"NodePort\"}}' service /monitoring-grafana patched "]

&



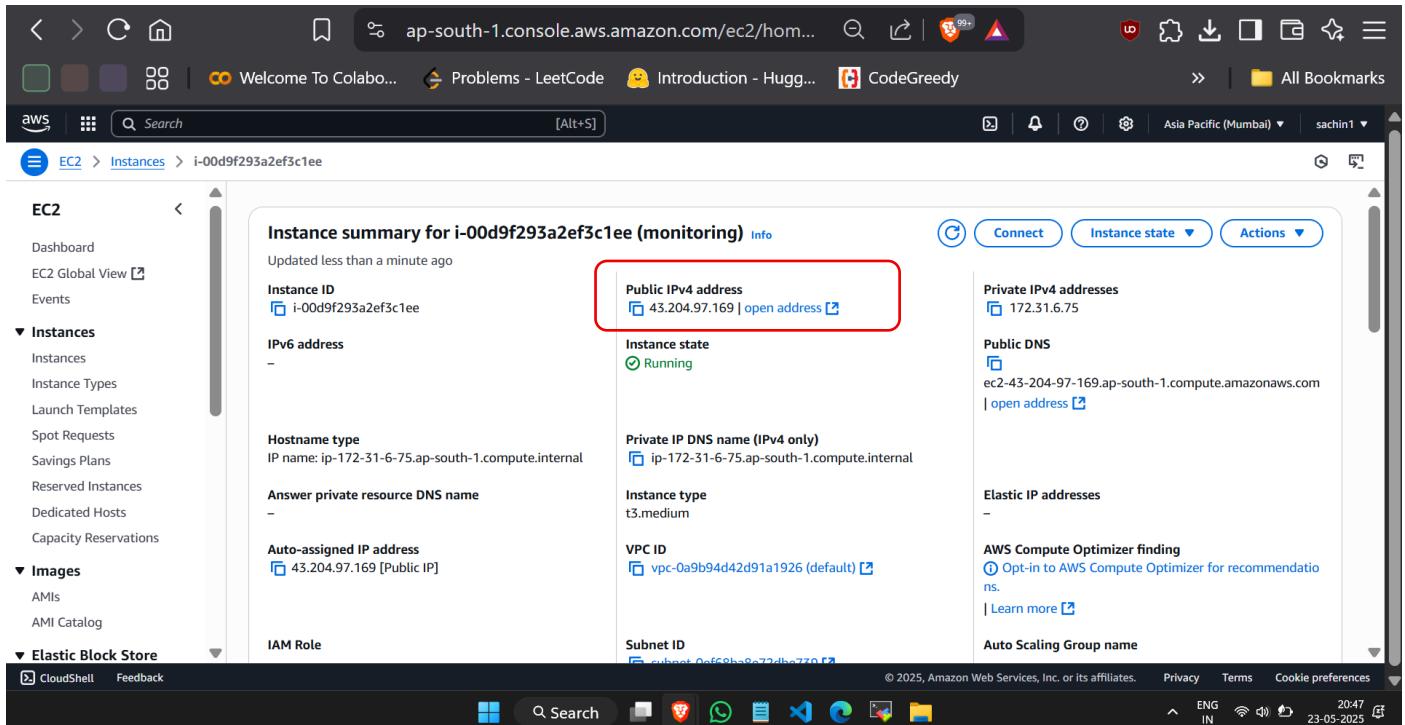
```
root@ip-172-31-10-207:/home/devops# kubectl patch svc monitoring-grafana \
-n monitoring \
-p '{"spec": {"type": "NodePort"}}'
service/monitoring-grafana patched
root@ip-172-31-10-207:/home/devops# kubectl patch svc monitoring-kube-prometheus-prometheus \
-n monitoring \
-p '{"spec": {"type": "NodePort"}}'
service/monitoring-kube-prometheus-prometheus patched
root@ip-172-31-10-207:/home/devops#
```

Now get the service list using “kubectl get svc -n monitoring” to get the cluster ip and the ports on which they are running.



```
root@ip-172-31-10-207:/home/devops# kubectl get svc -n monitoring
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)          AGE
alertmanager-operated  ClusterIP  None        <none>       9093/TCP,9094/TCP,9094/UDP  6m41s
monitoring-grafana   NodePort   10.101.74.142 <none>       80:30974/TCP               6m52s
monitoring-kube-prometheus-alertmanager  ClusterIP  10.97.204.189 <none>       9093/TCP,8080/TCP               6m52s
monitoring-kube-prometheus-operator    ClusterIP  10.105.103.218 <none>       443/TCP                    6m52s
monitoring-kube-prometheus-prometheus ClusterIP  10.110.75.170 <none>       9090:32756/TCP,8080:31998/TCP  6m52s
monitoring-kube-state-metrics       ClusterIP  10.108.193.131 <none>       8080/TCP                  6m52s
monitoring-prometheus-node-exporter  ClusterIP  10.106.128.23  <none>       9100/TCP                  6m52s
prometheus-operated            ClusterIP  None        <none>       9090/TCP                  3m51s
root@ip-172-31-10-207:/home/devops#
```

Get the monitoring public ip address



The screenshot shows the AWS EC2 Instances page for an instance named "i-00d9f293a2ef3c1ee". The instance is labeled as "monitoring". The "Public IPv4 address" field is highlighted with a red box and contains the value "43.204.97.169". Other visible details include:

- Instance ID:** i-00d9f293a2ef3c1ee
- IPv6 address:** -
- Hostname type:** IP name: ip-172-31-6-75.ap-south-1.compute.internal
- Answer private resource DNS name:** -
- Auto-assigned IP address:** 43.204.97.169 [Public IP]
- IAM Role:** -
- Instance state:** Running
- Private IP DNS name (IPv4 only):** ip-172-31-6-75.ap-south-1.compute.internal
- Instance type:** t3.medium
- VPC ID:** vpc-0a9b94d42d91a1926 (default)
- Subnet ID:** subnet-0fc81a972d4b7730
- Private IPv4 addresses:** 172.31.6.75
- Public DNS:** ec2-43-204-97-169.ap-south-1.compute.amazonaws.com
- Elastic IP addresses:** -
- AWS Compute Optimizer finding:** Opt-in to AWS Compute Optimizer for recommendations.
- Auto Scaling Group name:** -

Now access the Prometheus using the monitoring ip address on the exposed port number i.e 32756

Endpoint	Labels	Last scrape	State
http://192.168.224.2:9093/metrics	container="alertmanager", endpoint="http-web", instance="192.168.224.2:9093", job="monitoring-kube-prometheus-alertmanager", namespace="monitoring", pod="alertmanager-monitoring-kube-prometheus-alertmanager-0", service="monitoring-kube-prometheus-alertmanager"	31.789s ago	3ms UP
http://192.168.224.2:8080/metrics	container="config-reloader", endpoint="reloader-web", instance="192.168.224.2:8080", job="monitoring-kube-prometheus-alertmanager", namespace="monitoring", pod="alertmanager-monitoring-kube-prometheus-alertmanager-0", service="monitoring-kube-prometheus-alertmanager"	23.081s ago	2ms UP
https://172.31.10.207:6443/metrics	endpoint="https", instance="172.31.10.207:6443", job="apiserver", namespace="default", service="kubernetes"	13.315s ago	12ms UP
http://192.168.228.196:9153/metrics	container="coredns", endpoint="http-metrics", instance="192.168.228.196:9153", job="coredns", namespace="kube-system", pod="coredns-796658455-6q7t7", service="coredns"	8.163s ago	3ms UP

Get the password of the Grafana using the command "kubectl get secret -n monitoring monitoring-grafana
\
-o jsonpath="{.data.admin-password}" | base64 -d && echo"

```
root@ip-172-31-10-207:/home/devops# kubectl get secret --namespace monitoring monitoring-grafana \
-o jsonpath="{.data.admin-user}" | base64 -d; echo
admin
root@ip-172-31-10-207:/home/devops# kubectl get secret --namespace monitoring monitoring-grafana \
-o jsonpath="{.data.admin-password}" | base64 -d; echo
prom-operator
root@ip-172-31-10-207:/home/devops#
```

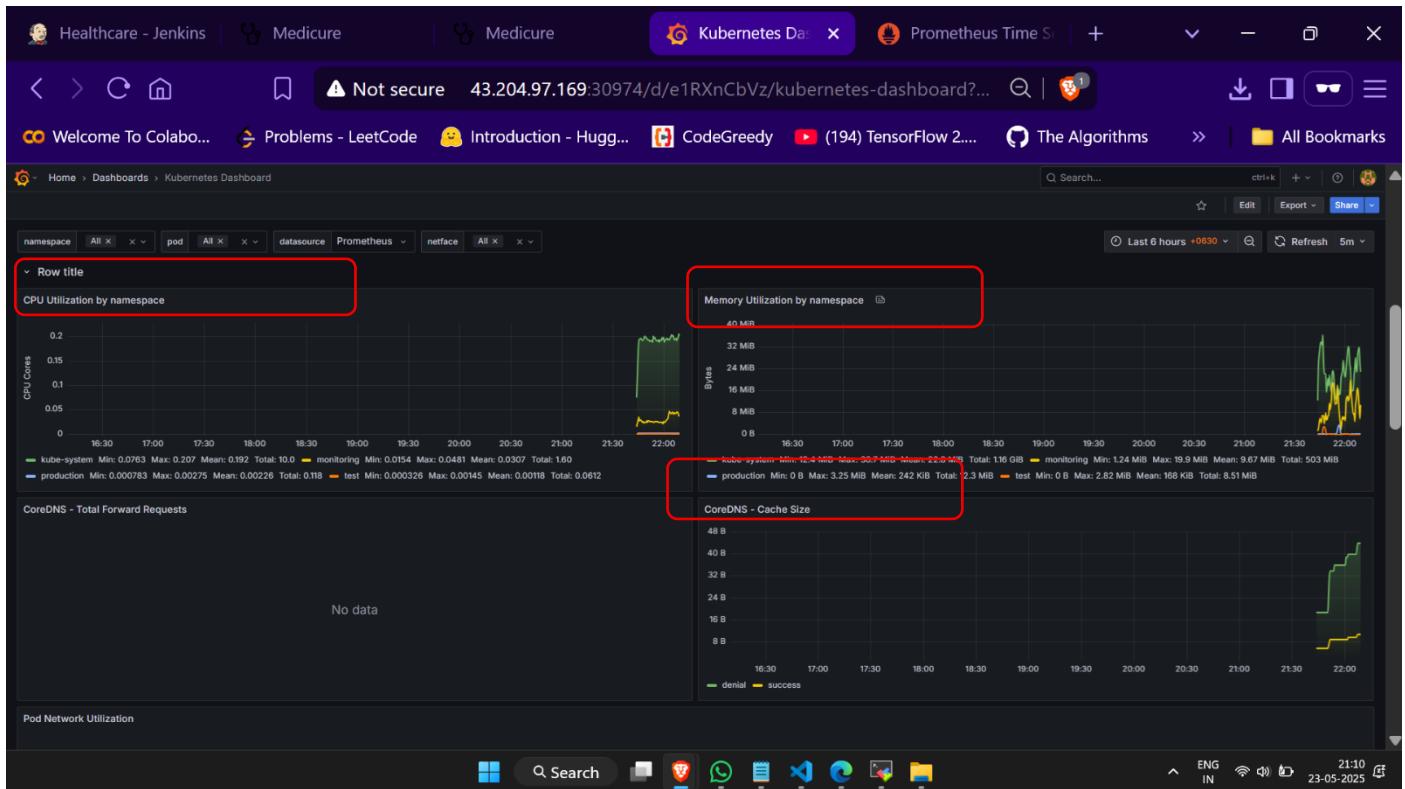
Now using the credentials login into the Grafana on port 30974

The screenshot shows the Grafana home page with a dark theme. At the top, there's a navigation bar with tabs for "Healthcare - Jenkins", "Medicure", "Medicure", "Home - Dashboard", and "Prometheus Time Series". Below the navigation is a toolbar with icons for back, forward, search, and other functions. The main content area features a "Welcome to Grafana" card. To the left is a sidebar titled "Grafana" with sections for Home, Bookmarks, Starred, Dashboards, Explore, Drilldown, Alerting, Connections, and Administration. On the right, there's a "Need help?" section with links to Documentation, Tutorials, Community, and Public Slack. A "Basic" section provides a quick start guide. Below the main content are links to "Dashboards", "Starred dashboards", and "Recently viewed dashboards". A "Latest from the blog" section shows a post about a security release. The bottom of the screen shows system status icons and the date/time.

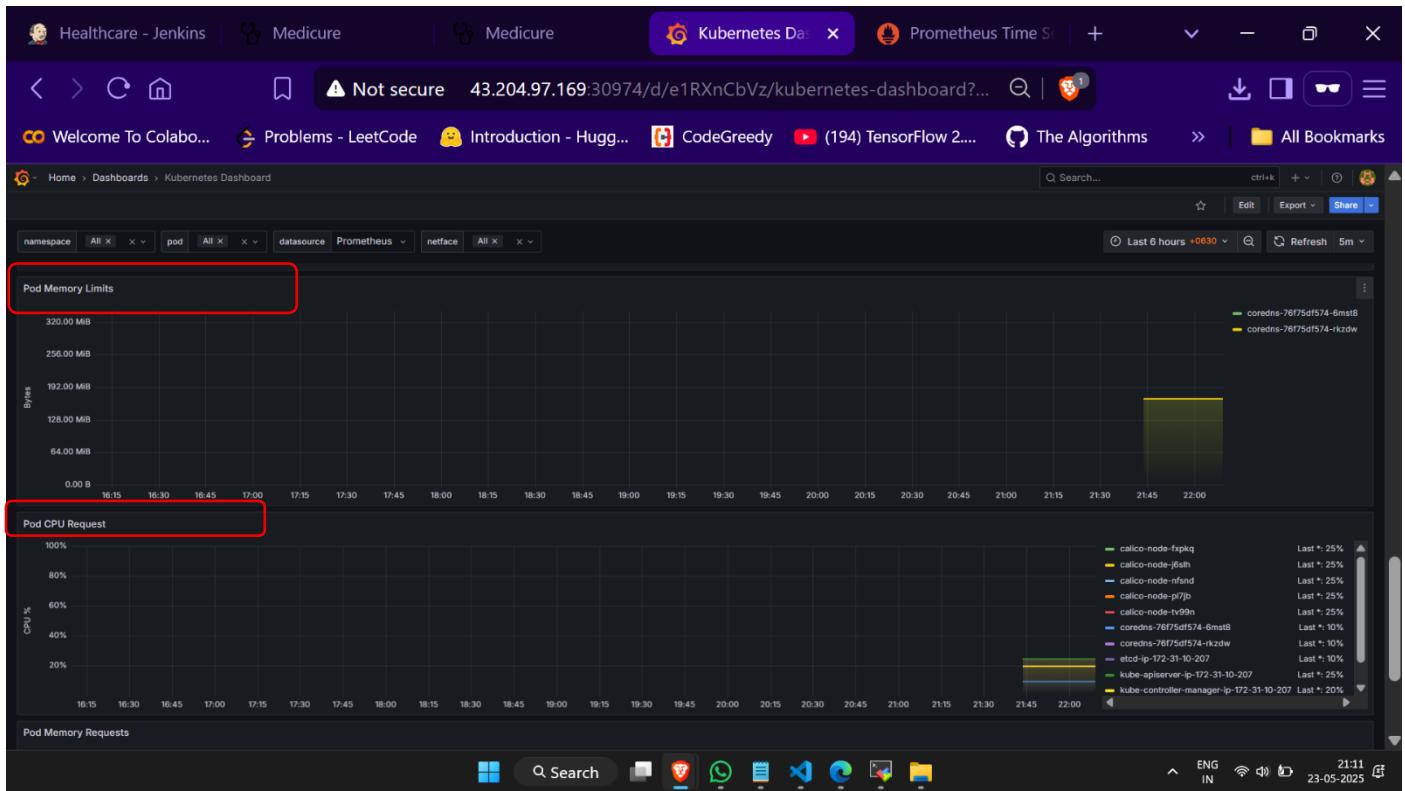
Now create the dashboard using the community supported charts which are pre built but works on our data to monitor all the pods and the production machine nodes using the above filter. we can monitor different metrics by namespace , pods, and networking

The screenshot shows the "Kubernetes Dashboard" within Grafana. The interface includes a sidebar with "Dashboards" selected, and a top navigation bar with tabs for "namespace", "pod", "datasource", "netface", and "All". The main area displays several charts: "Global CPU Usage" (Real Requests: 6.15%, 21.0%), "Global RAM Usage" (Real Requests: 32.62%, 2.29%, 1.77%), "Nodes" (5 Nodes, 7 Namespaces), "Running Pods" (30 Pods), "Cluster CPU Utilization" (6.15%), and "Cluster Memory Utilization" (32.6%). A red circle highlights the top navigation bar and the "Edit", "Export", and "Share" buttons at the top right. The bottom of the screen shows system status icons and the date/time.

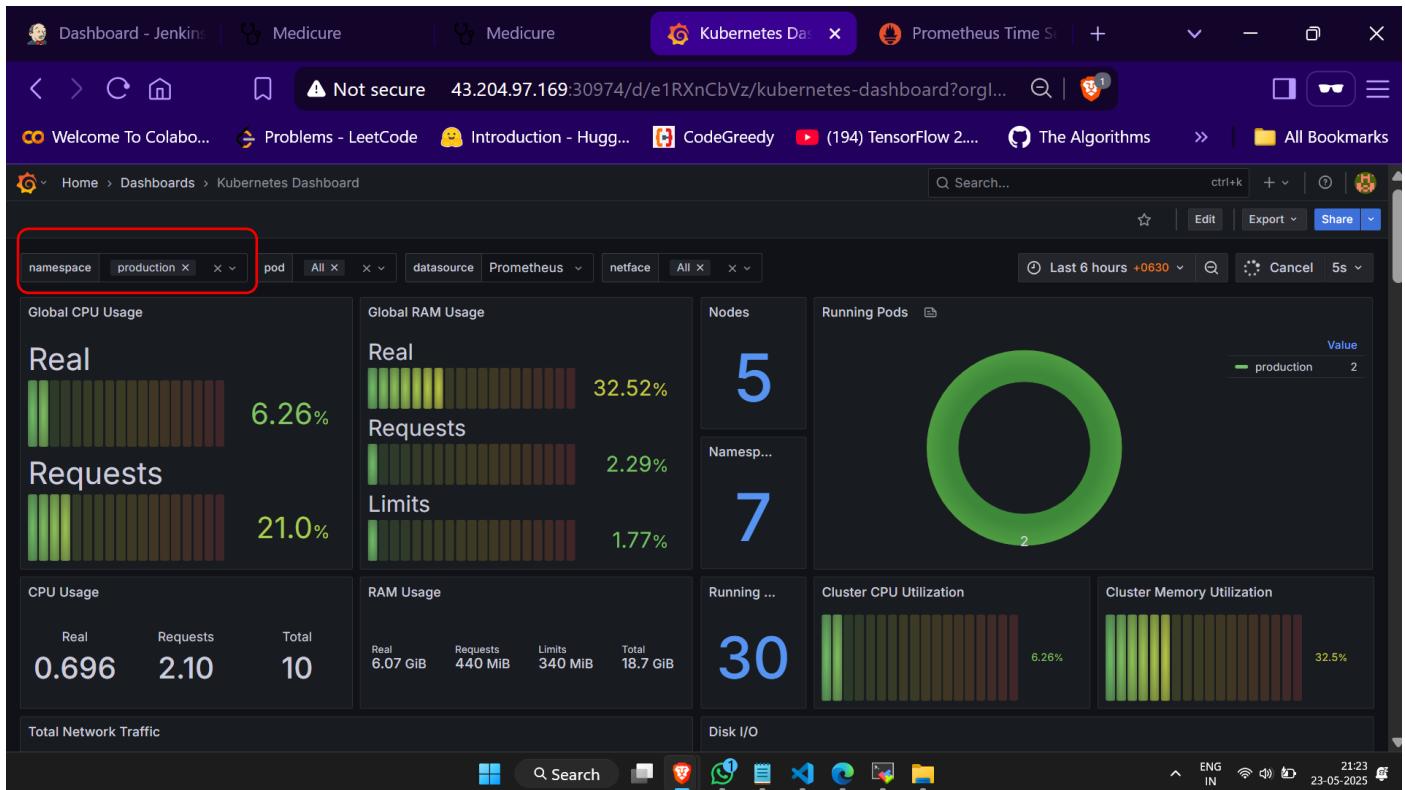
CPU utilization of the nodes i,e by namespace



Memory utilization of nodes



Metrics with namespace filter of production



Now verify the pods running by namespace

The screenshot shows a terminal window in MobaXterm with multiple tabs. The current tab shows the command 'kubectl get pods -n test' with the output:

```
root@ip-172-31-10-207:/home/devops# kubectl get pods -n test
NAME          READY   STATUS    RESTARTS   AGE
my-app-9675ddf65-izrpt  1/1     Running   0          74m
```

The next tab shows 'kubectl get pods -n monitoring' with the output:

```
root@ip-172-31-10-207:/home/devops# kubectl get pods -n monitoring
NAME          READY   STATUS    RESTARTS   AGE
alertmanager-monitoring-kube-prometheus-alertmanager-0  2/2     Running   0          50m
monitoring-grafana-7bb6c7cff67-rl8cw  3/3     Running   0          51m
monitoring-kube-prometheus-operator-7f6b98fcbb9-29vgx  1/1     Running   0          51m
monitoring-kube-state-metrics-d8f6bcbb5b-rbhhs  1/1     Running   0          51m
monitoring-prometheus-node-exporter-4nk7  1/1     Running   0          51m
monitoring-prometheus-node-exporter-72lz4  1/1     Running   0          51m
monitoring-prometheus-node-exporter-7r6xn  1/1     Running   0          51m
monitoring-prometheus-node-exporter-rrkhk  1/1     Running   0          51m
monitoring-prometheus-node-exporter-v6276  1/1     Running   0          51m
prometheus-monitoring-kube-prometheus-prometheus-0  2/2     Running   0          48m
```

The final tab shows 'kubectl get pods -n production' with the output:

```
root@ip-172-31-10-207:/home/devops# kubectl get pods -n production
NAME          READY   STATUS    RESTARTS   AGE
my-app-6ff84684b-2w7kb  1/1     Running   0          66m
my-app-6ff84684b-f6j14  1/1     Running   0          66m
```

The terminal window includes a file explorer sidebar, a status bar at the bottom, and a taskbar at the very bottom.