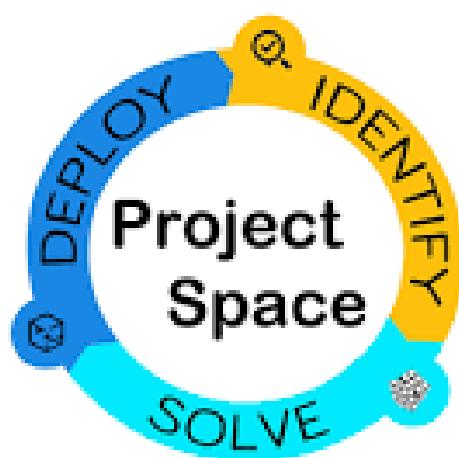


AUTOMATE APPLICATIONS USING ANSIBLE



**BY: Vishal Battula
20A91A0506**

SCOPE:

Ansible is a popular automation tool that allows you to automate IT tasks such as configuration management, application deployment, and infrastructure management. Its scope is vast, and it can be used to automate a wide range of applications and systems.

Some of the popular use cases for Ansible automation include:

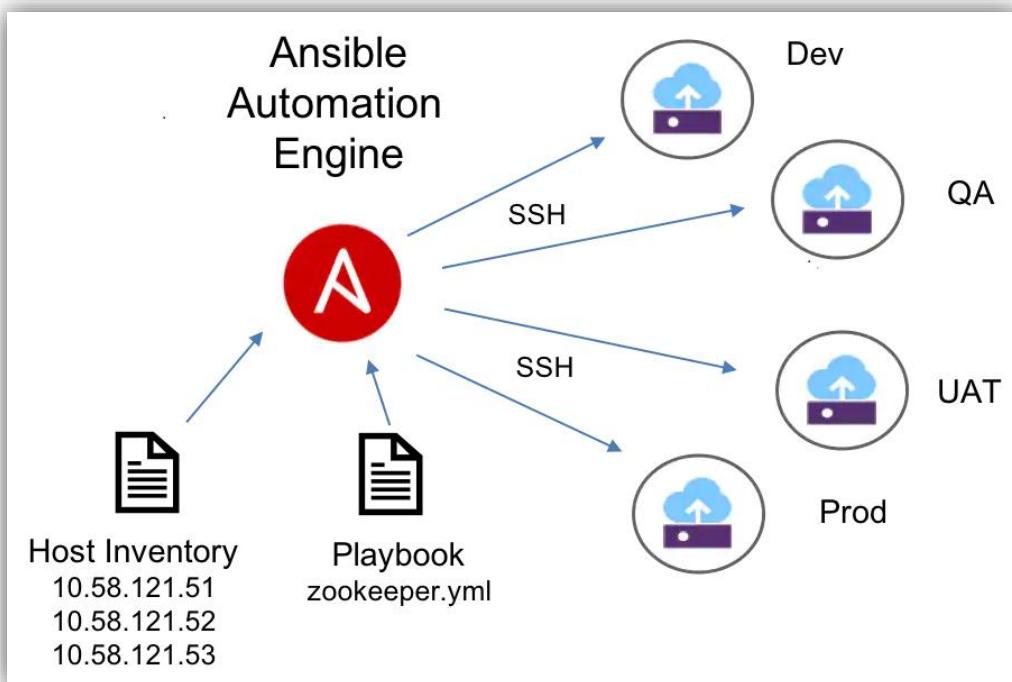
Configuration management: Ansible can be used to automate the configuration of servers, networking devices, and other IT resources. This includes tasks such as setting up users and permissions, installing software and packages, and configuring services.

Application deployment: Ansible can automate the deployment of applications and their dependencies, including web servers, databases, and load balancers. This can be particularly useful in environments where frequent updates and deployments are required.

Infrastructure management: Ansible can automate the provisioning of infrastructure resources such as virtual machines, containers, and cloud instances. This includes tasks such as creating and configuring network interfaces, storage volumes, and security groups.

Security automation: Ansible can be used to automate security-related tasks such as vulnerability scanning, patch management, and compliance auditing.

Overall, Ansible's flexibility and wide range of capabilities make it a valuable tool for automating various IT tasks and applications.



PURPOSE:

The purpose of automating applications using Ansible is to simplify and streamline IT operations. By automating routine tasks, such as software installation, configuration management, and infrastructure provisioning, Ansible can save time and reduce the risk of errors that can occur with manual processes.

Some of the key benefits of automating applications with Ansible include

Improved efficiency: Ansible enables you to automate repetitive tasks, freeing up time for IT teams to focus on more strategic work.

Consistency: Automation with Ansible ensures that IT operations are standardized and consistent, reducing the risk of errors and security vulnerabilities.

Speed: Ansible can execute tasks much faster than manual processes, reducing the time it takes to deploy applications and infrastructure.

Scalability: Ansible is highly scalable and can automate tasks across large numbers of servers and devices.

Collaboration: Ansible provides a platform for collaboration between IT teams, allowing for the sharing of playbooks and roles that can be used across the organization.

Overall, the purpose of automating applications with Ansible is to improve efficiency, consistency, speed, and scalability, while reducing the risk of errors and increasing collaboration between IT teams.

TOOLS & TECHNOLOGIES USED:

Ansible:

- Ansible is an open-source automation tool.
- It uses YAML-based playbooks to define desired system state.
- Ansible is agentless, lightweight, and easy to set up.



Git & Git-Hub:

- Git is a distributed version control system.
- It allows developers to track changes to their code over time.
- Git enables collaboration among developers by allowing them to work on the same codebase simultaneously.



AWS:

- AWS (Amazon Web Services) is a cloud computing platform that provides a wide range of services and solutions for businesses and individuals.
- It offers on-demand computing resources, such as virtual machines, storage, and databases, that can be accessed over the internet.



Slack:

- Slack is a messaging platform that can be used to notify team members about the status of the deployment process.
- Ansible can be configured to send notifications to Slack channels when certain tasks are completed or when there are errors in the deployment process.



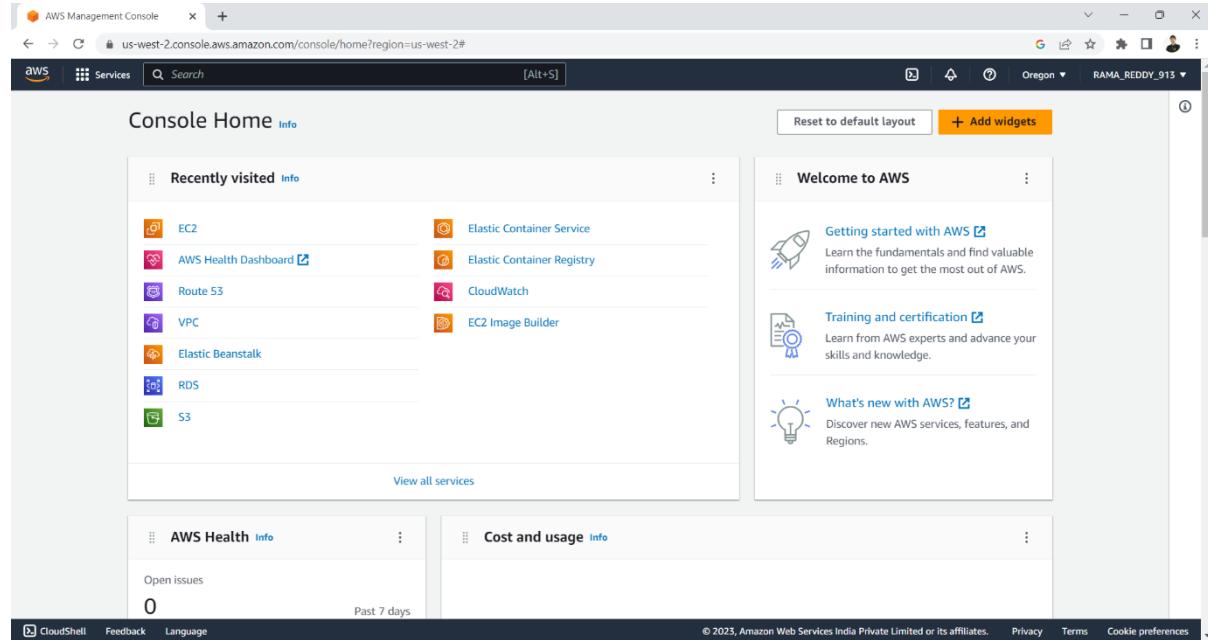
JIRA:

- JIRA is a project management tool that can be used to track issues and bugs during the deployment process.
- Ansible can be integrated with JIRA to create tickets automatically when there are errors in the deployment process.



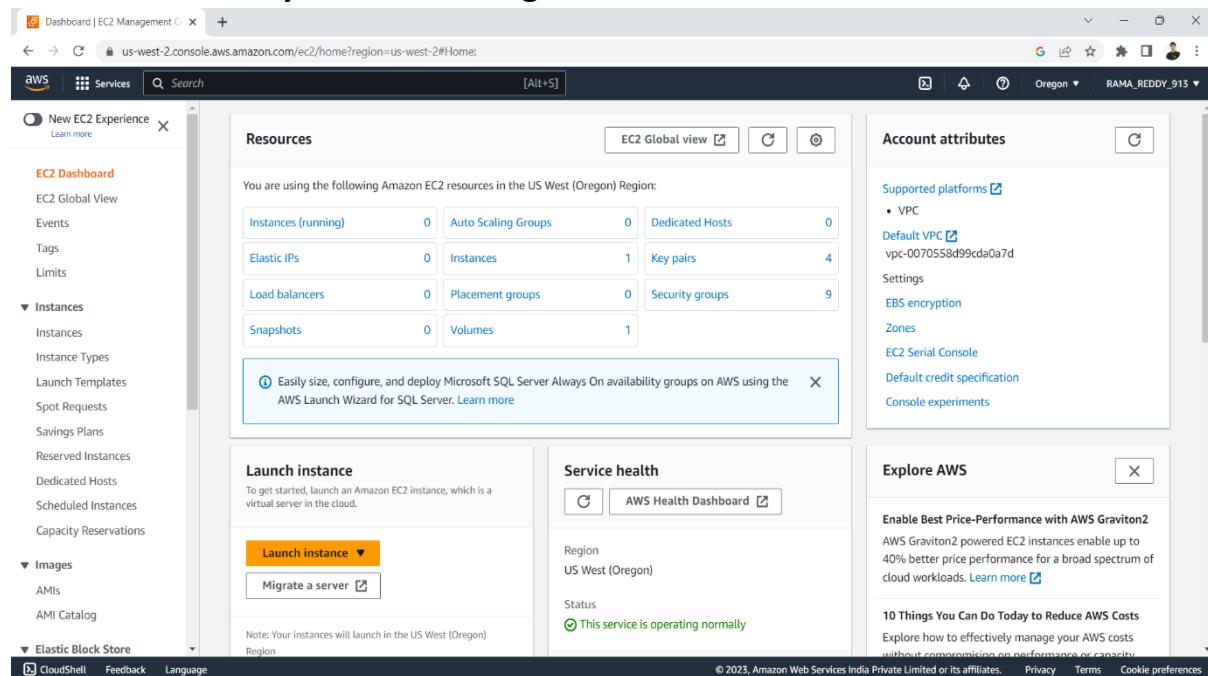
PROOF OF CONCEPT:

Login To aws.amazon.com with your credentials And navigate to console home



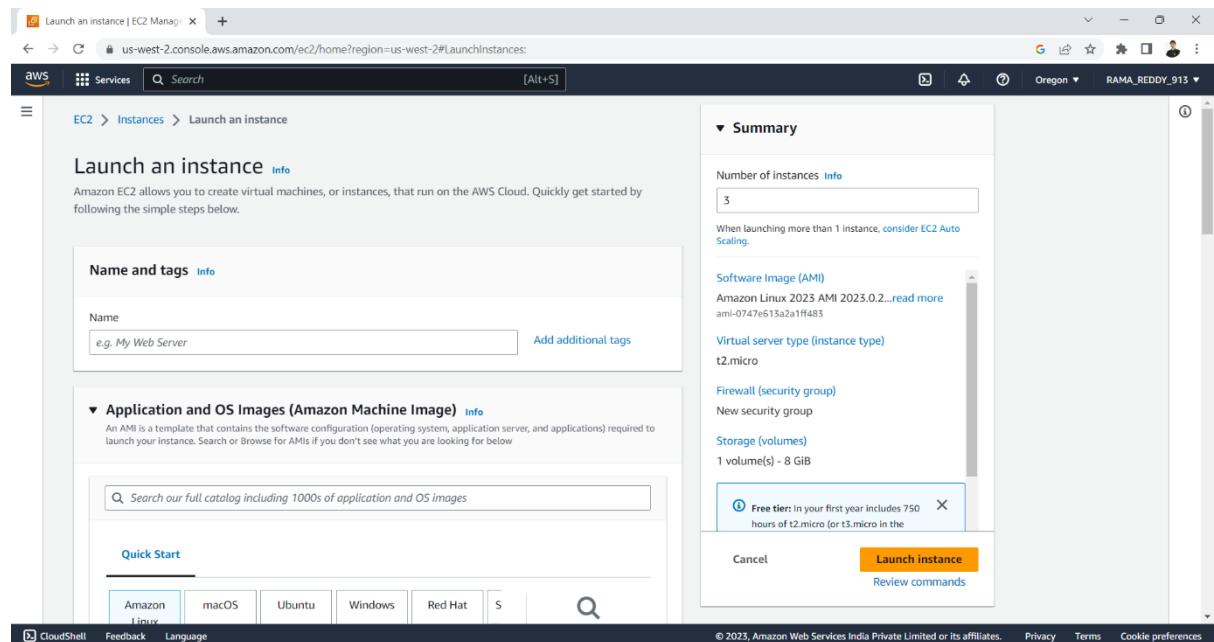
The screenshot shows the AWS Management Console Home page. At the top, there's a navigation bar with tabs for 'AWS' and 'Services', a search bar, and account information for 'Oregon' and 'RAMA_RED俞_915'. Below the navigation is a 'Console Home' section with a 'Recently visited' list containing links to EC2, AWS Health Dashboard, Route 53, VPC, Elastic Beanstalk, RDS, and S3. To the right of this is a 'Welcome to AWS' sidebar with sections for 'Getting started with AWS', 'Training and certification', and 'What's new with AWS?'. Below these are sections for 'AWS Health' (0 open issues) and 'Cost and usage'.

Click on Ec2 then you will be navigated to EC2 Dashboard

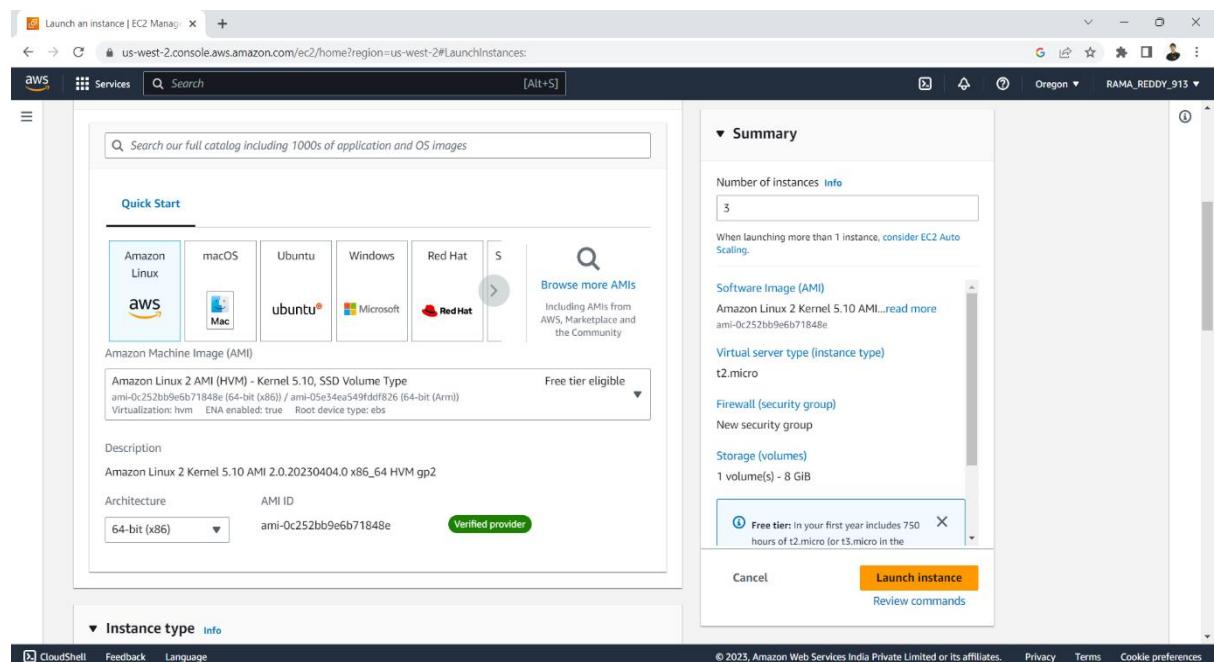


The screenshot shows the EC2 Management Console Home page. The left sidebar includes 'EC2 Dashboard' (New EC2 Experience), 'Instances' (Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Scheduled Instances, Capacity Reservations), 'Images' (AMIs, AMI Catalog), and 'Elastic Block Store'. The main content area has a 'Resources' summary table showing 0 instances (running), 0 auto scaling groups, 0 dedicated hosts, 0 elastic IPs, 1 instance, 4 key pairs, 0 load balancers, 0 placement groups, 9 security groups, 0 snapshots, and 1 volume. It also features a 'Launch instance' button and a 'Service health' status indicator (operating normally). On the right, there's an 'Account attributes' sidebar with options like 'Supported platforms' (VPC), 'Default VPC' (vpc-0070558d99cda0a7d), and 'Explore AWS' sections for price-performance and cost reduction.

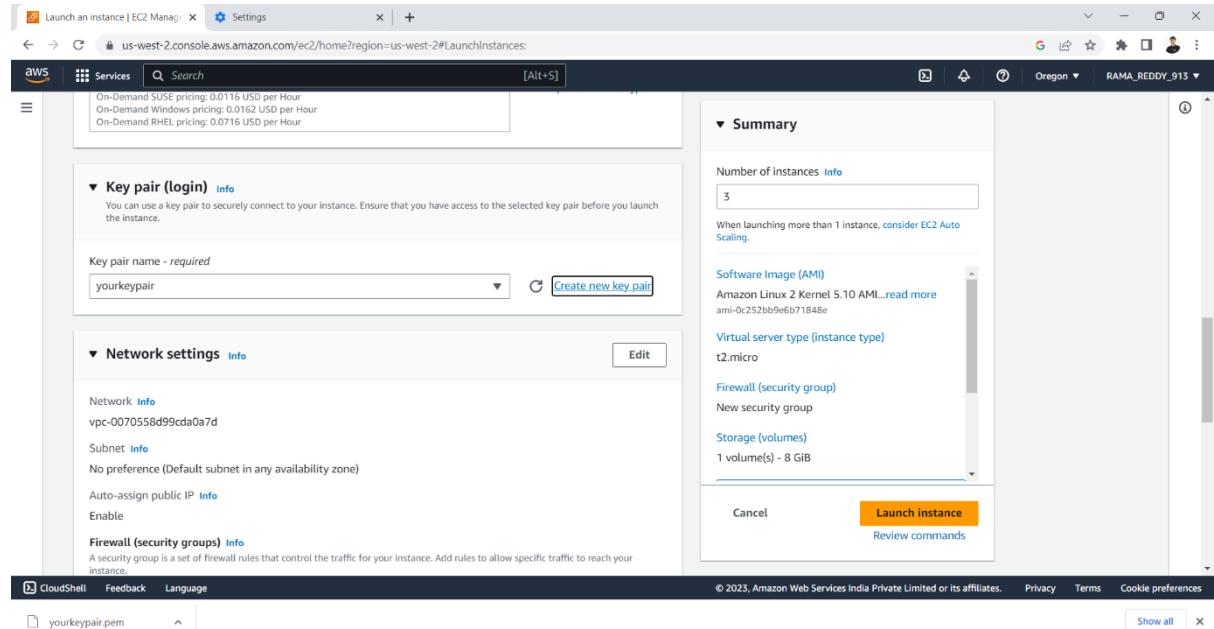
Click On Instances and Launch 3 instances at a time



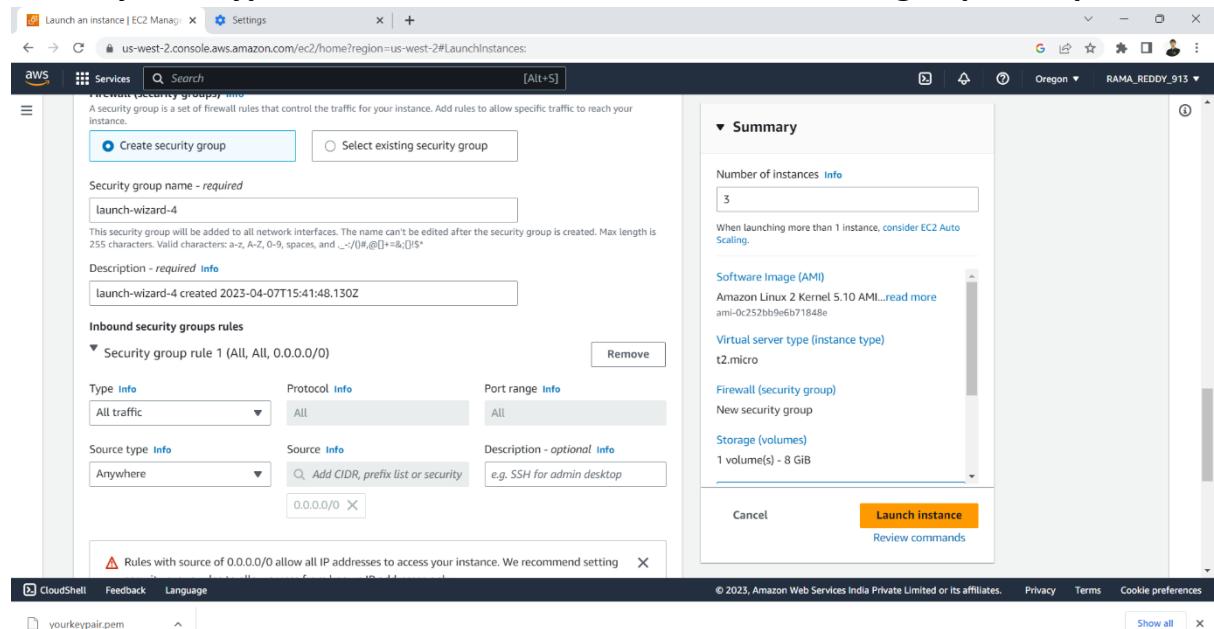
Select the AMI (Amazon Machine Image) in our case select Amazon Linux2
And select the Architecture of the operating System.



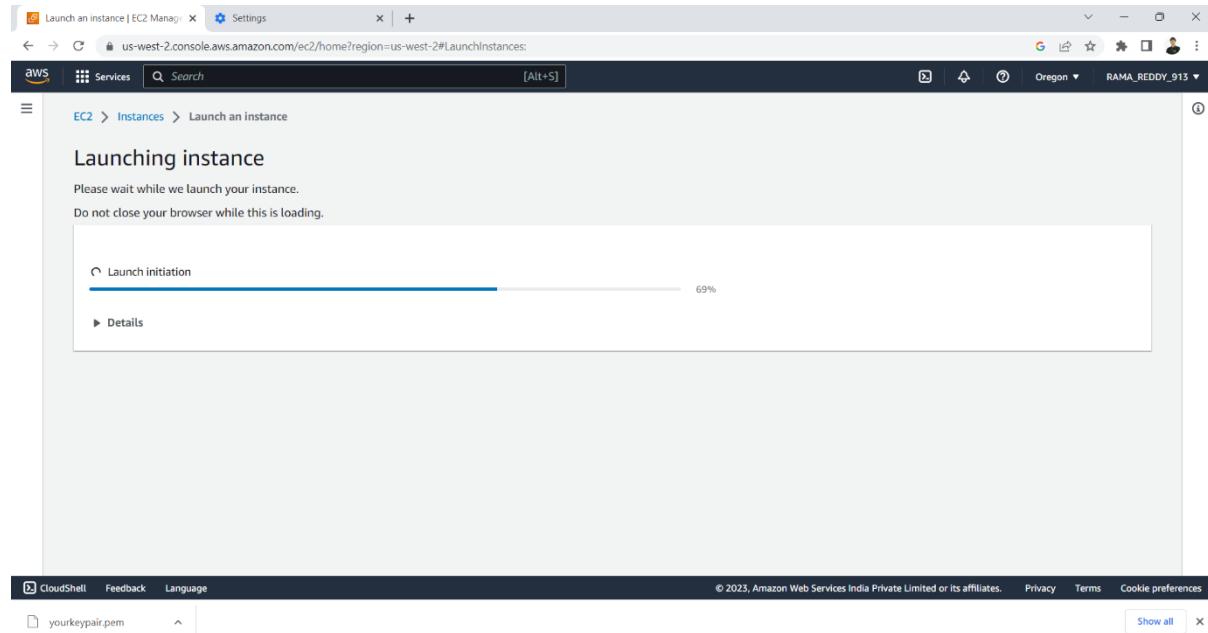
Create a keypair, keypair allows you to connect to your instance .



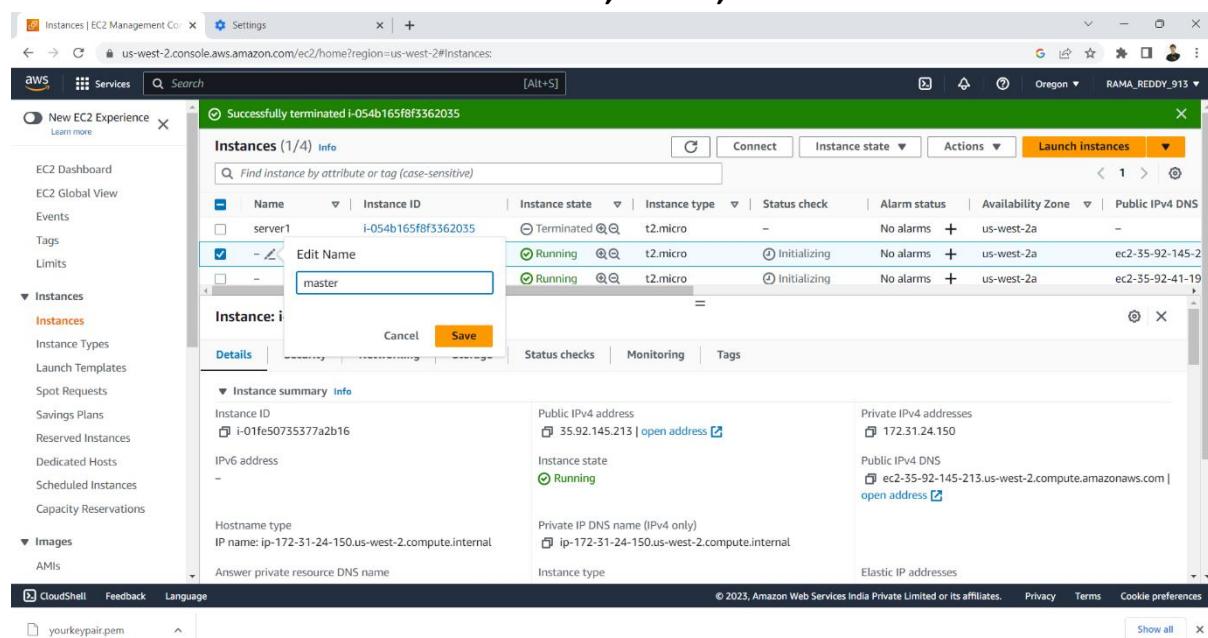
Select your keypair and VPC, subnet and enable auto assign public Ip



Click on create security group and select inbound security group rules. In our case select ALL Traffic, finally click on launch instance.



Instance launched. Rename as Master ,Node1,Node2.

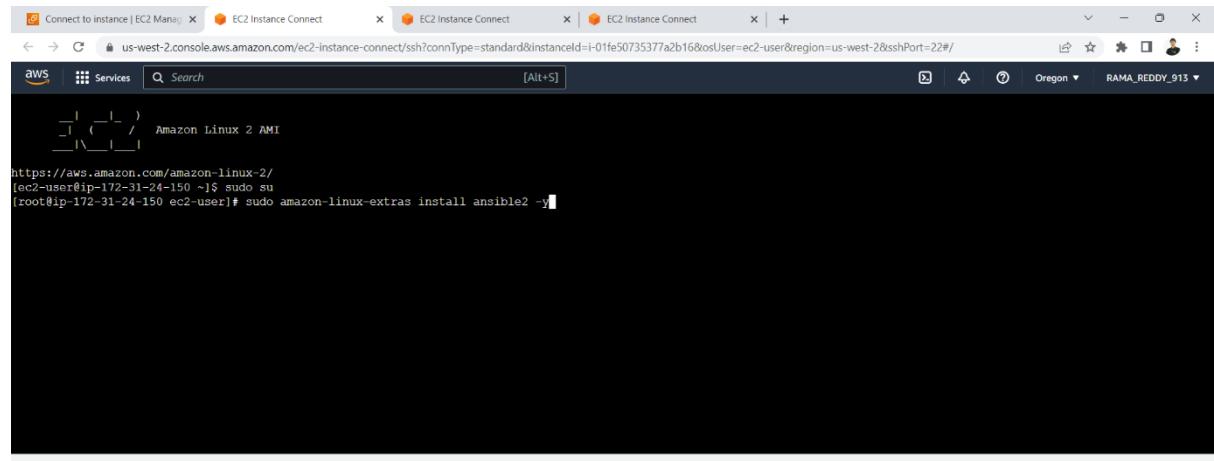


The screenshot shows the AWS EC2 Management Console interface. On the left, a sidebar navigation bar includes links for EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Scheduled Instances, Capacity Reservations, and Images (AMIs). The main content area displays a table of instances. A modal window titled 'Edit Name' is open over the 'node1' instance, which has a blue selection box around it. The modal contains a text input field with 'node1' and a 'Save' button. The table lists four instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
master	i-0fe50755377a2b16	Running	t2.micro	Initializing	No alarms	us-west-2a	ec2-35-92-145-2
node1	i-0e9990104bda6bc8	Running	t2.micro	Initializing	No alarms	us-west-2a	ec2-35-92-41-19
node2	i-038380d7e95ea7010	Running	t2.micro	Initializing	No alarms	us-west-2a	ec2-35-93-146-2

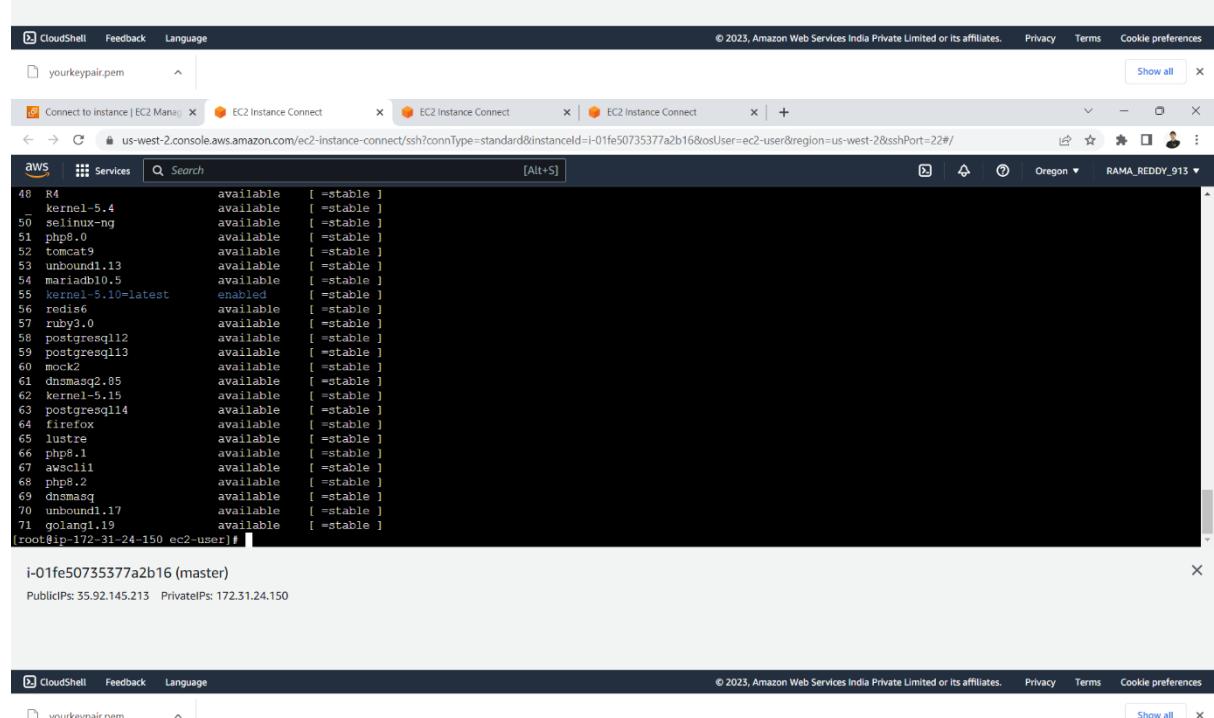
The 'node1' instance's details are shown in the 'Details' tab of the modal. The 'node2' instance's details are shown in the 'Details' tab of the main table. Both instances have identical configurations: Public IPv4 address 35.93.146.2, Private IP DNS name ip-172-31-26-173.us-west-2.compute.internal, and Instance type t2.micro.

Connect to master instance and execute the following commands



```
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-24-150 ~]$ sudo su
[root@ip-172-31-24-150 ec2-user]# sudo amazon-linux-extras install ansible2 -y
```

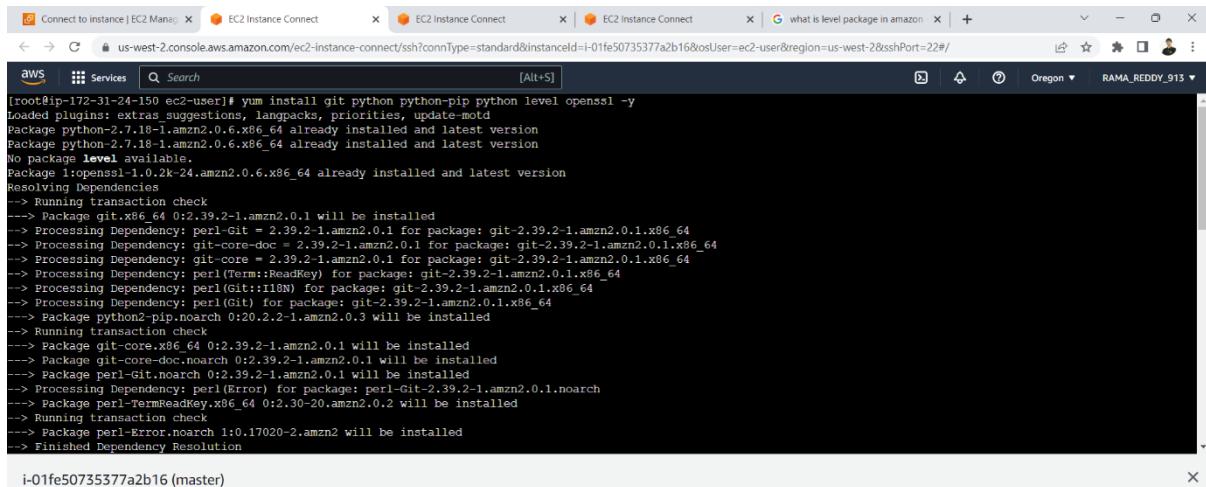
i-01fe50735377a2b16 (master)
PublicIPs: 35.92.145.213 PrivateIPs: 172.31.24.150



```
48 R4 available [ =stable ]
49 kernel-5.4 available [ =stable ]
50 selinux-ng available [ =stable ]
51 php8.0 available [ =stable ]
52 tomcat9 available [ =stable ]
53 unbound1.13 available [ =stable ]
54 mariadb10.5 available [ =stable ]
55 kernel-5.10-latest enabled [ =stable ]
56 redis6 available [ =stable ]
57 ruby3.0 available [ =stable ]
58 postgresql12 available [ =stable ]
59 postgresql13 available [ =stable ]
60 mock2 available [ =stable ]
61 dnsmasq2.05 available [ =stable ]
62 kernel-5.15 available [ =stable ]
63 postgresql14 available [ =stable ]
64 firefox available [ =stable ]
65 lustre available [ =stable ]
66 php8.1 available [ =stable ]
67 awscli1 available [ =stable ]
68 php8.2 available [ =stable ]
69 dnsmasq available [ =stable ]
70 unbound1.17 available [ =stable ]
71 golang1.19 available [ =stable ]
```

[root@ip-172-31-24-150 ec2-user]#

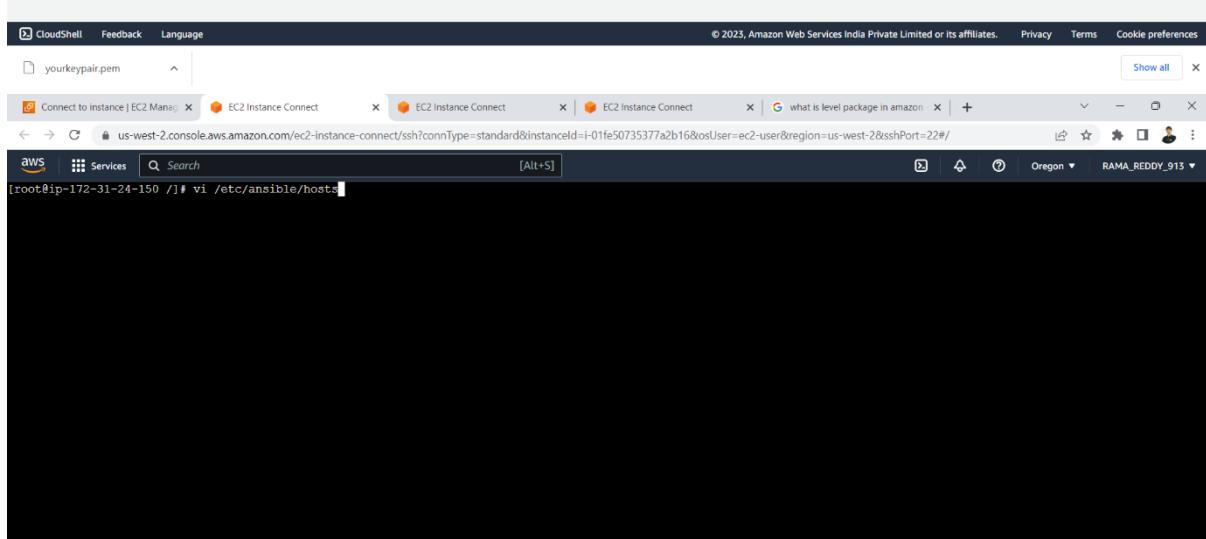
i-01fe50735377a2b16 (master)
PublicIPs: 35.92.145.213 PrivateIPs: 172.31.24.150



```
[root@ip-172-31-24-150 ec2-user]# yum install git python python-pip python level openssl -y
Loaded plugins: extras suggestions, langpacks, priorities, update-motd
Package python-2.7.18-1.amzn2.0.6.x86_64 already installed and latest version
Package python-2.7.18-1.amzn2.0.6.x86_64 already installed and latest version
No package level available.
Package l:openssl-1.0.2k-24.amzn2.0.6.x86_64 already installed and latest version
Resolving Dependencies
--> Running transaction check
--> Package git.x86_64 0:2.39.2-1.amzn2.0.1 will be installed
--> Processing Dependency: perl-Git = 2.39.2-1.amzn2.0.1 for package: git-2.39.2-1.amzn2.0.1.x86_64
--> Processing Dependency: git-core-doc = 2.39.2-1.amzn2.0.1 for package: git-2.39.2-1.amzn2.0.1.x86_64
--> Processing Dependency: git-core = 2.39.2-1.amzn2.0.1 for package: git-2.39.2-1.amzn2.0.1.x86_64
--> Processing Dependency: perl (Term::ReadKey) for package: git-2.39.2-1.amzn2.0.1.x86_64
--> Processing Dependency: perl (Git::i18N) for package: git-2.39.2-1.amzn2.0.1.x86_64
--> Processing Dependency: perl (Git) for package: git-2.39.2-1.amzn2.0.1.x86_64
--> Package python2-pip.noarch 0:20.2.2-1.amzn2.0.3 will be installed
--> Running transaction check
--> Package git-core.x86_64 0:2.39.2-1.amzn2.0.1 will be installed
--> Package git-core-doc.noarch 0:2.39.2-1.amzn2.0.1 will be installed
--> Package perl-git.noarch 0:2.39.2-1.amzn2.0.1 will be installed
--> Processing Dependency: perl-TermReadkey.x86_64 0:2.30-20.amzn2.0.2 will be installed
--> Running transaction check
--> Package perl-TermReadkey.noarch 1:0.17020-2.amzn2 will be installed
--> Finished Dependency Resolution
```

i-01fe50735377a2b16 (master)

PublicIPs: 35.92.145.213 PrivateIPs: 172.31.24.150



```
[root@ip-172-31-24-150 ~]# vi /etc/ansible/hosts
```

i-01fe50735377a2b16 (master)

PublicIPs: 35.92.145.213 PrivateIPs: 172.31.24.150

create a group and enter the public ip of the nodes

The screenshot shows three stacked CloudShell windows. The top window displays an Ansible inventory configuration:

```
# - Comments begin with the '#' character
# - Blank lines are ignored
# - Groups of hosts are delimited by [header] elements
# - You can enter hostnames or ip addresses
# - A hostname/ip can be a member of multiple groups

## 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
[development]
35.92.41.198
35.93.146.2
## [webservers]
## alpha.example.org
## beta.example.org
## 192.168.1.100
## 192.168.1.110

-- INSERT --
```

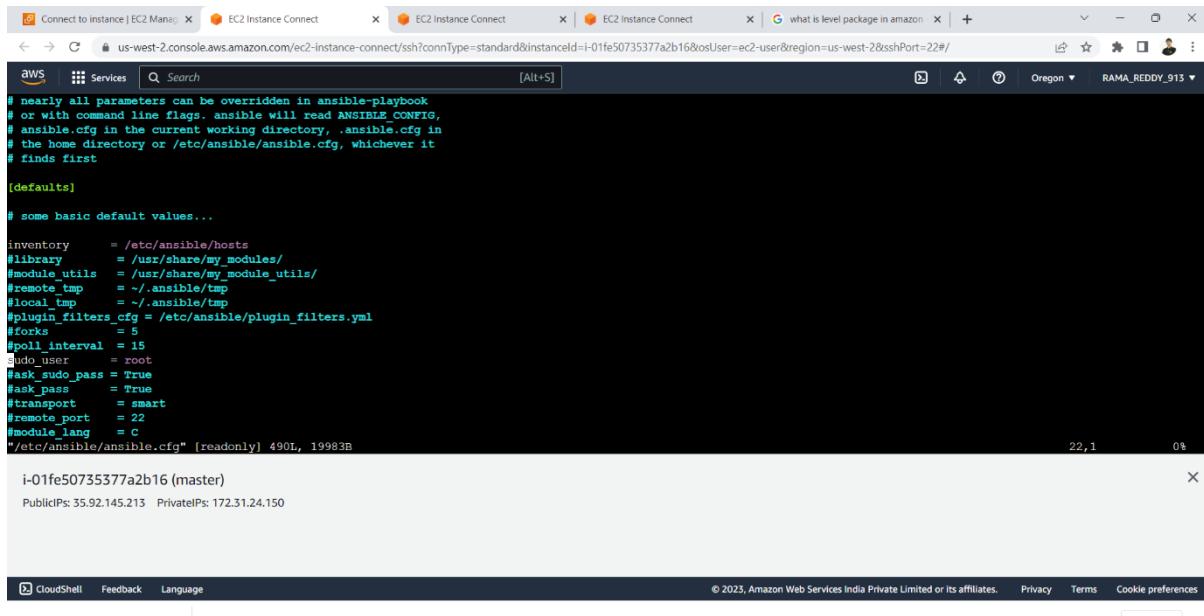
The middle window shows the command to edit the hosts file:

```
[root@ip-172-31-24-150 ~]# vi /etc/ansible/hosts
```

The bottom window shows the command to edit the ansible.cfg file:

```
[root@ip-172-31-24-150 ~]# vi /etc/ansible/ansible.cfg
```

Uncomment inventory and sudo user



The screenshot shows a terminal window in AWS CloudShell. The user is editing the Ansible configuration file (`/etc/ansible/ansible.cfg`). The configuration includes parameters like `inventory`, `library`, `#module_utils`, `#remote_tmp`, `#local_tmp`, `#plugin_filters_cfg`, `#forks`, `#poll_interval`, `sudo_user`, `#ask_sudo_pass`, `#ask_pass`, `#transport`, `#remote_port`, `#module_lang`, and `#remote_user`. The file is 490L long and 19983B in size.

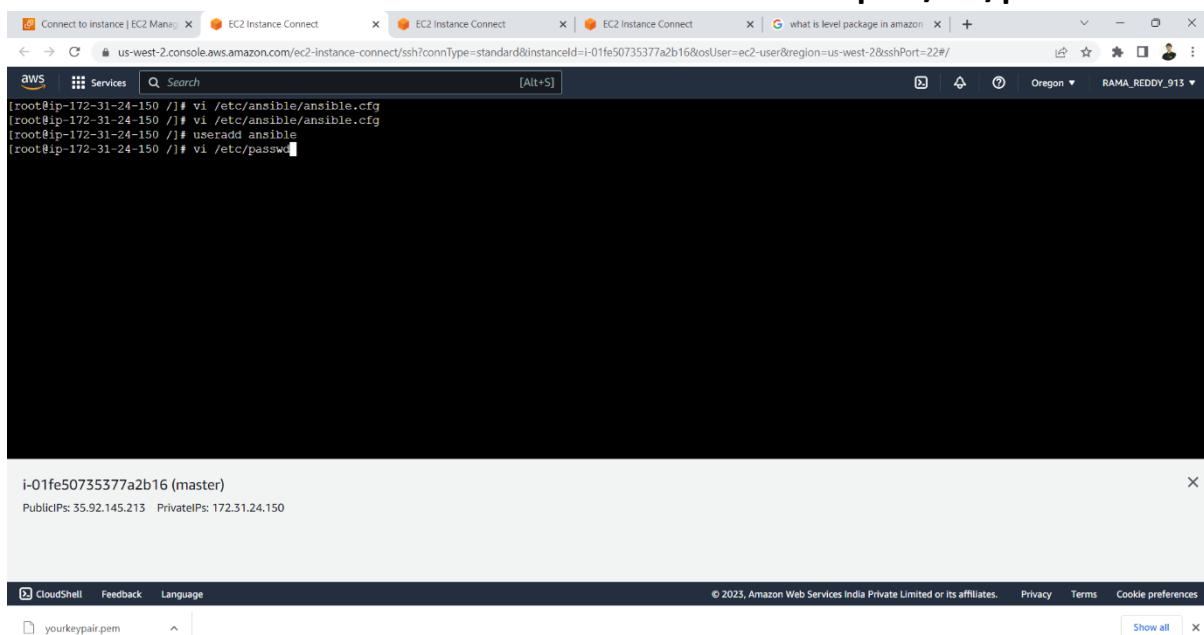
```
# nearly all parameters can be overridden in ansible-playbook
# or with command line flags. ansible will read ANSIBLE_CONFIG,
# ansible.cfg in the current working directory, .ansible.cfg in
# the home directory or /etc/ansible/ansible.cfg, whichever it
# finds first

[defaults]

# some basic default values...

inventory      = /etc/ansible/hosts
library        = /usr/share/my_modules/
#module_utils   = /usr/share/my_module_utils/
#remote_tmp    = ~/ansible/tmp
#local_tmp     = ~/ansible/tmp
#plugin_filters_cfg = /etc/ansible/plugin_filters.yml
#forks          = 5
#poll_interval = 15
sudo_user      = root
#ask_sudo_pass = True
#ask_pass       = True
#transport     = smart
#remote_port   = 22
#module_lang   = C
"/etc/ansible/ansible.cfg" [readonly] 490L, 19983B
```

Create a user named ansible to ensure user is created open /etc/passwd file



The screenshot shows a terminal window in AWS CloudShell. The user is creating a new user account named `ansible` using the `useradd` command. The user then opens the `/etc/passwd` file in vi editor to verify the user entry.

```
[root@ip-172-31-24-150 ~]# vi /etc/ansible/ansible.cfg
[root@ip-172-31-24-150 ~]# vi /etc/ansible/ansible.cfg
[root@ip-172-31-24-150 ~]# useradd ansible
[root@ip-172-31-24-150 ~]# vi /etc/passwd
```

Connect to Instance | EC2 Manager | EC2 Instance Connect | EC2 Instance Connect | what is level package in amazon | + | Oregon | RAMA_RED俞_913

```

aws Services Search [Alt+S]
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/sbin/nologin
systemd-network:x:92:92:Network Management:/sbin/nologin
dbus:x:81:91:System message bus:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
libstoragemgmt:x:999:997:daemon account for libstoragemgmt:/var/run/lsm:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
rngd:x:998:996:Random Number Generator Daemon:/var/lib/rngd:/sbin/nologin
chrony:x:997:995:/var/lib/chrony:/sbin/nologin
ec2-instance-connect:x:996:994:/home/ec2-instance-connect:/sbin/nologin
postfix:x:89:89:/var/spool/postfix:/sbin/nologin
tcpdump:x:72:72::/sbin/nologin
ec2-user:x:1000:1000:EC2 Default User:/home/ec2-user:/bin/bash
ansible:x:1001:1001:/home/ansible:/bin/bash

```

i-01fe50735377a2b16 (master)

PublicIPs: 35.92.145.213 PrivateIPs: 172.31.24.150

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences Show all

give root user preprivileges to the user created

Connect to Instance | EC2 Manager | EC2 Instance Connect | EC2 Instance Connect | what is level package in amazon | + | Oregon | RAMA_RED俞_913

```

root@ip-172-31-24-150 ~]# vi /etc/ansible/ansible.cfg
[root@ip-172-31-24-150 ~]# vi /etc/ansible/ansible.cfg
[root@ip-172-31-24-150 ~]# useradd ansible
[root@ip-172-31-24-150 ~]# vi /etc/passwd
[root@ip-172-31-24-150 ~]# passwd ansible
Changing password for user ansible.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
[root@ip-172-31-24-150 ~]# visudo

```

i-01fe50735377a2b16 (master)

PublicIPs: 35.92.145.213 PrivateIPs: 172.31.24.150

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences Show all

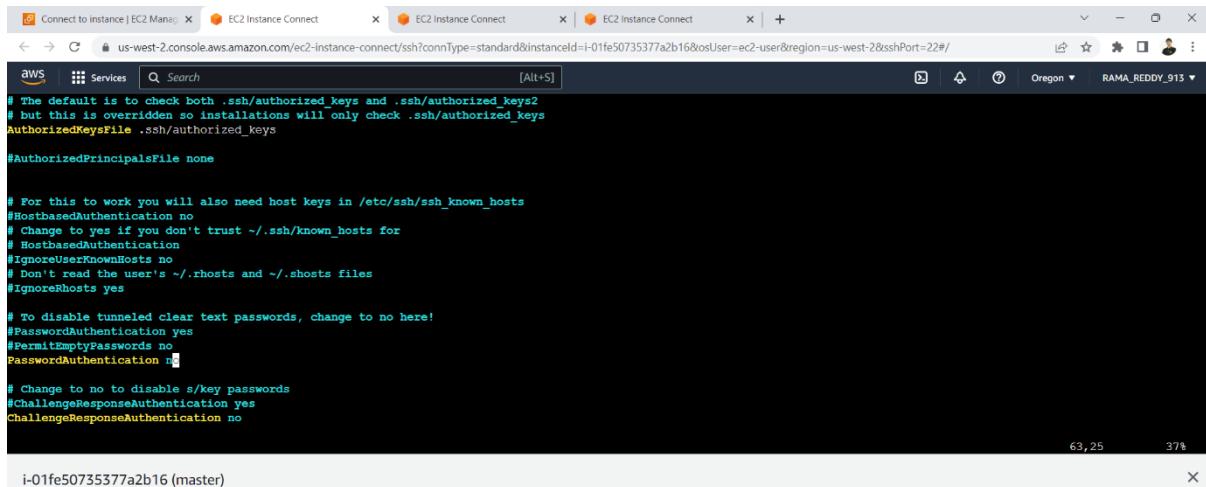
```
# Adding HOME to env_keep may enable a user to run unrestricted
# commands via sudo.
#
# Defaults    env_keep += "HOME"
#
Defaults    secure_path = /sbin:/bin:/usr/sbin:/usr/bin

## Next comes the main part: which users can run what software on
## which machines (the sudoers file can be shared between multiple
## systems).
## Syntax:
##
##     user      MACHINE=COMMANDS
##
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root    ALL=(ALL)        ALL
ansible ALL=(ALL)        NOPASSWD: ALL
## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# tsysd ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES, LOCATE, DRIVERS
## Allows people in group wheel to run all commands

## i-01fe50735377a2b16 (master)
PublicIPs: 35.92.145.213  PrivateIPs: 172.31.24.150
```

open /etc/ssh/sshd_config and change password authentication from no to yes and perform the same process for all the nodes

```
[root@ip-172-31-24-150 ~]# vim /etc/ssh/sshd_config
```



```

# The default is to check both .ssh/authorized_keys and .ssh/authorized_keys2
# but this is overridden so installations will only check .ssh/authorized_keys
AuthorizedKeysFile .ssh/authorized_keys

#AuthorizedPrincipalsFile none

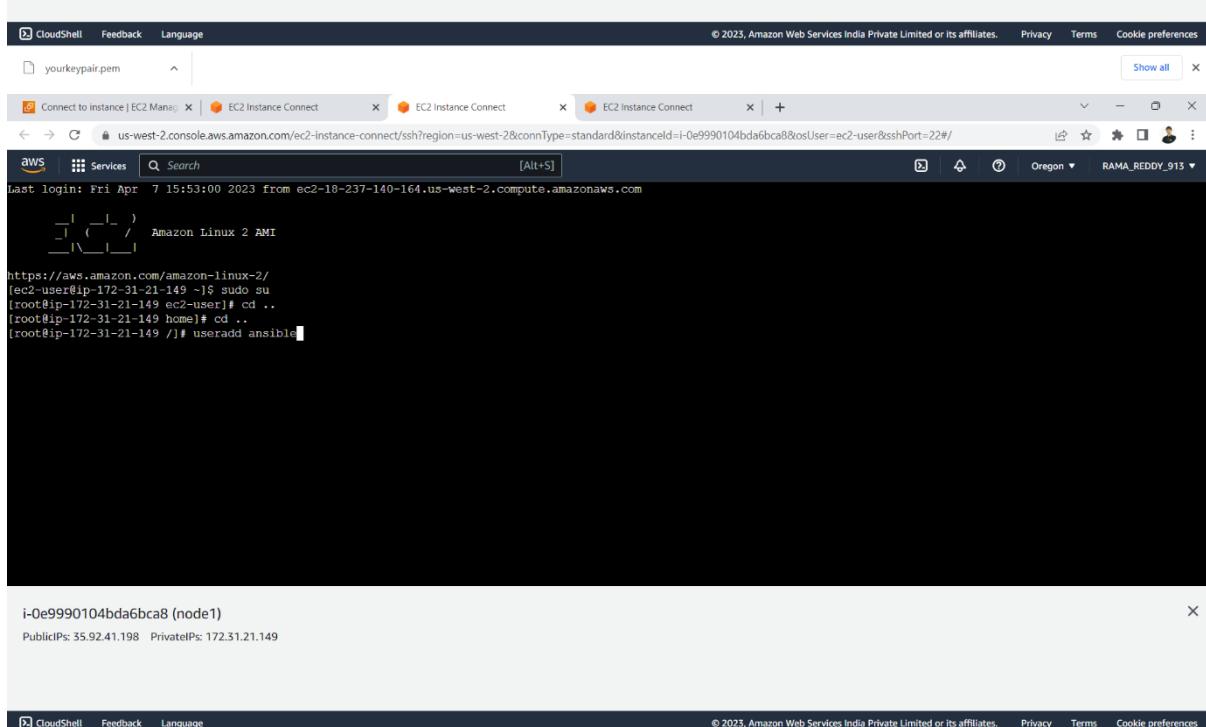
# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# to disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no
PasswordAuthentication no

# Change to no to disable s/Key passwords
#ChallengeResponseAuthentication yes
ChallengeResponseAuthentication no

```

i-01fe50735377a2b16 (master)
PublicIPs: 35.92.145.213 PrivateIPs: 172.31.24.150



```

Last login: Fri Apr 7 15:53:00 2023 from ec2-18-237-140-164.us-west-2.compute.amazonaws.com
[ec2-user@ip-172-31-21-149 ~]$ sudo su
[root@ip-172-31-21-149 ec2-user]# cd ..
[root@ip-172-31-21-149 home]# cd ..
[root@ip-172-31-21-149 /]# useradd ansible

```

i-0e9990104bda6bca8 (node1)
PublicIPs: 35.92.41.198 PrivateIPs: 172.31.21.149

i-0e9990104bda6bca8 (node1)

PublicIPs: 35.92.41.198 PrivateIPs: 172.31.21.149

```
aws CloudShell Feedback Language
Connect to instance | EC2 Manager | EC2 Instance Connect | EC2 Instance Connect | EC2 Instance Connect | EC2 Instance Connect | + | Oregon | RAMA_REDDEY_913 | CloudShell | Feedback | Language | Search | [Alt+S] | © 2023, Amazon Web Services India Private Limited or its affiliates. | Privacy | Terms | Cookie preferences

last login: Fri Apr 7 15:53:00 2023 from ec2-18-237-140-164.us-west-2.compute.amazonaws.com
_ _| _ _ )
_| ( _ / Amazon Linux 2 AMI
__\_\_\_\_

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-21-149 ~]$ sudo su
[root@ip-172-31-21-149 ~]# cd ..
[root@ip-172-31-21-149 home]# useradd ansible
[root@ip-172-31-21-149 ~]# cat /etc/passwd

```

i-0e9990104bda6bca8 (node1)

PublicIPs: 35.92.41.198 PrivateIPs: 172.31.21.149

```
aws CloudShell Feedback Language
Connect to instance | EC2 Manager | EC2 Instance Connect | EC2 Instance Connect | EC2 Instance Connect | EC2 Instance Connect | + | Oregon | RAMA_REDDEY_913 | CloudShell | Feedback | Language | Search | [Alt+S] | © 2023, Amazon Web Services India Private Limited or its affiliates. | Privacy | Terms | Cookie preferences

shutdown:x:6:0:shutdown:/sbin/shutdown
halt:x:7:0:halt:/sbin/halt
mail:x:0:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/sbin/nologin
systemd-network:x:192:192:Network Management:/sbin/nologin
dbus:x:81:81:System message bus:/sbin/nologin
rpcbind:x:32:32:rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
libstoragemgmt:x:999:99:daemon account for libstoragemgmt:/var/run/lsm:/sbin/nologin
sshd:x:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
rngd:x:998:996:Random Number Generator Daemon:/var/lib/rngd:/sbin/nologin
chrony:x:997:995:/var/lib/chrony:/sbin/nologin
ec2-instance-connect:x:996:994:/home/ec2-instance-connect:/sbin/nologin
postfix:x:89:89:/var/spool/postfix:/sbin/nologin
cpdump:x:72:72::/sbin/nologin
ec2-user:x:1000:1000:EC2 Default User:/home/ec2-user:/bin/bash
ansible:x:1001:1001:/home/ansible:/bin/bash
[root@ip-172-31-21-149 ~]# passwd ansible
Changing password for user ansible.
new password:
BAD PASSWORD: The password is shorter than 8 characters
retype new password:
passwd: all authentication tokens updated successfully.
[root@ip-172-31-21-149 ~]#
```

i-0e9990104bda6bca8 (node1)

PublicIPs: 35.92.41.198 PrivateIPs: 172.31.21.149

```

aws Services Search [Alt+S] EC2 Instance Connect EC2 Instance Connect EC2 Instance Connect + Oregon RAMA_RED俞_913
shutdownx:6:0:shutdown:/sbin/shutdown
haltx:7:0:halt:/sbin/halt
mailx:8:12:mail:/var/spool/mail:/sbin/nologin
operatorx:11:0:operator:/root:/sbin/nologin
gamesx:12:100:games:/usr/games:/sbin/nologin
ftpx:14:50:FTP User:/var/ftp:/sbin/nologin
nobodyx:99:99:Nobody:/sbin/nologin
systemd-networkx:192:192:systemd Network Management:/sbin/nologin
ibusx:81:81:System message bus:/sbin/nologin
rpcx:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
libstoragemgmtx:999:997:daemon account for libstoragemgmt:/var/run/lsm:/sbin/nologin
sshdx:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
rpcusobodyx:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfssnobodyx:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
rngd:998:996:Random Number Generator Daemon:/var/lib/rngd:/sbin/nologin
chronyx:997:995:/:/var/lib/chrony:/sbin/nologin
ec2-instance-connectx:996:994:/:/home/ec2-instance-connect:/sbin/nologin
postfixx:89:89:/:/var/spool/postfix:/sbin/nologin
tcpdumpx:72:72:/:/sbin/nologin
ec2-userx:1000:1000:EC2 Default User:/home/ec2-user:/bin/bash
ansiblex:1001:1001:/:/home/ansible:/bin/bash
[root@ip-172-31-21-149 ~]# passwd ansible
Changing password for user ansible.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
[root@ip-172-31-21-149 ~]# visudo
i-0e9990104bda6bca8 (node1)
PublicIPs: 35.92.41.198 PrivateIPs: 172.31.21.149
```



```

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences
aws Services Search [Alt+S] EC2 Instance Connect EC2 Instance Connect EC2 Instance Connect + Oregon RAMA_RED俞_913
Defaults env_keep += "LC_TIME LC_ALL LANGUAGE LINGUAS XKB_CHARSET XAUTHORITY"

## Adding HOME to env_keep may enable a user to run unrestricted
## commands via sudo.
## Defaults env_keep += "HOME"

Defaults secure_path = /sbin:/bin:/usr/sbin:/usr/bin

## Next comes the main part: which users can run what software on
## which machines (the sudoers file can be shared between multiple
## systems).
## Syntax:
##     user    MACHINE=COMMANDS
##     The COMMANDS section may have other options added to it.
##     Allow root to run any commands anywhere
root    ALL=(ALL)      ALL
ansible ALL=(ALL)      NOPASSWD: ALL

## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# sysx ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES, LOCATE, DRIVERS
-- INSERT --
i-0e9990104bda6bca8 (node1)
PublicIPs: 35.92.41.198 PrivateIPs: 172.31.21.149
```

The screenshot shows three stacked terminal windows from the AWS CloudShell interface. The top window displays the command `vi /etc/ssh/sshd_config` being run in a root shell on an EC2 instance. The middle window shows the contents of the `/etc/ssh/sshd_config` file, which includes various SSH configuration options such as host-based authentication, password authentication, and Kerberos support. The bottom window shows the command `vi /etc/ssh/sshd_config` again, indicating the file has been modified.

```
[root@ip-172-31-21-149 ~]# vi /etc/ssh/sshd_config
[root@ip-172-31-21-149 ~]#
```

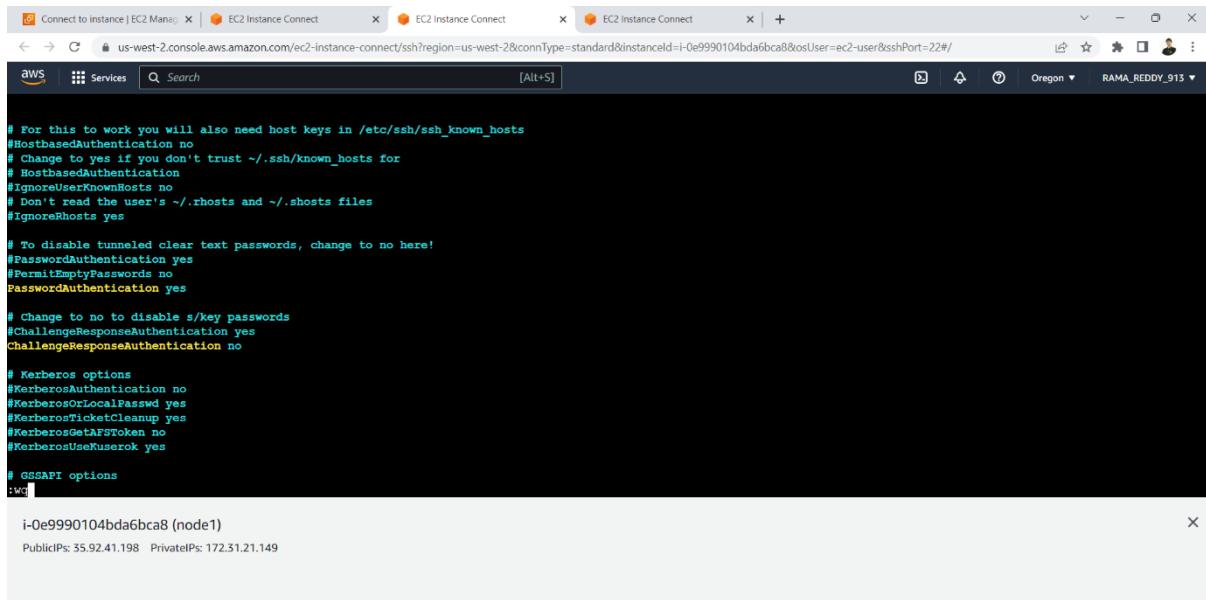
```
# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no
#PasswordAuthentication n

# Change to no to disable s/key passwords
#ChallengeResponseAuthentication yes
ChallengeResponseAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
#KerberosGetFSToken no
#KerberosUseKuserok yes

# GSSAPI options
"/etc/ssh/sshd_config" 140L, 3957B
```



```
# 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

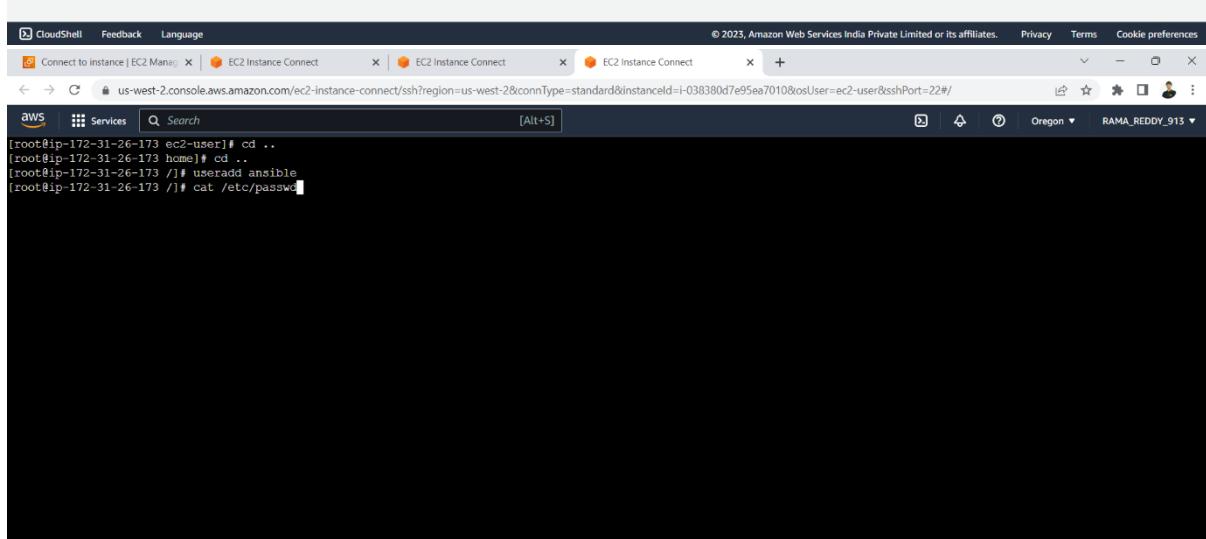
# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no
PasswordAuthentication yes

# Change to no to disable s/key passwords
#ChallengeResponseAuthentication yes
ChallengeResponseAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
#KerberosGetAFSToken no
#KerberosUseKuserok yes

# GSSAPI options
#GSSAPI

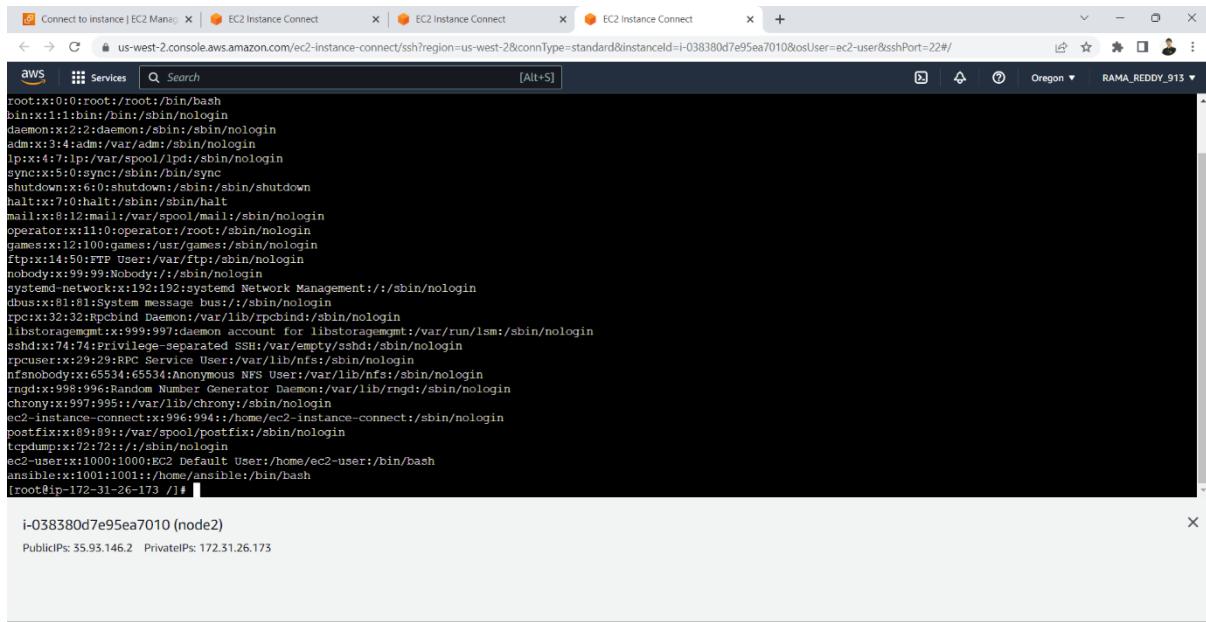
i-0e9990104bda6bca8 (node1)
PublicIPs: 35.92.41.198 PrivateIPs: 172.31.21.149
```



```
[root@ip-172-31-26-173 ec2-user]# cd ..
[root@ip-172-31-26-173 home]# cd ..
[root@ip-172-31-26-173 /]# useradd ansible
[root@ip-172-31-26-173 /]# cat /etc/passwd

i-038380d7e95ea7010 (node2)
PublicIPs: 35.93.146.2 PrivateIPs: 172.31.26.173
```





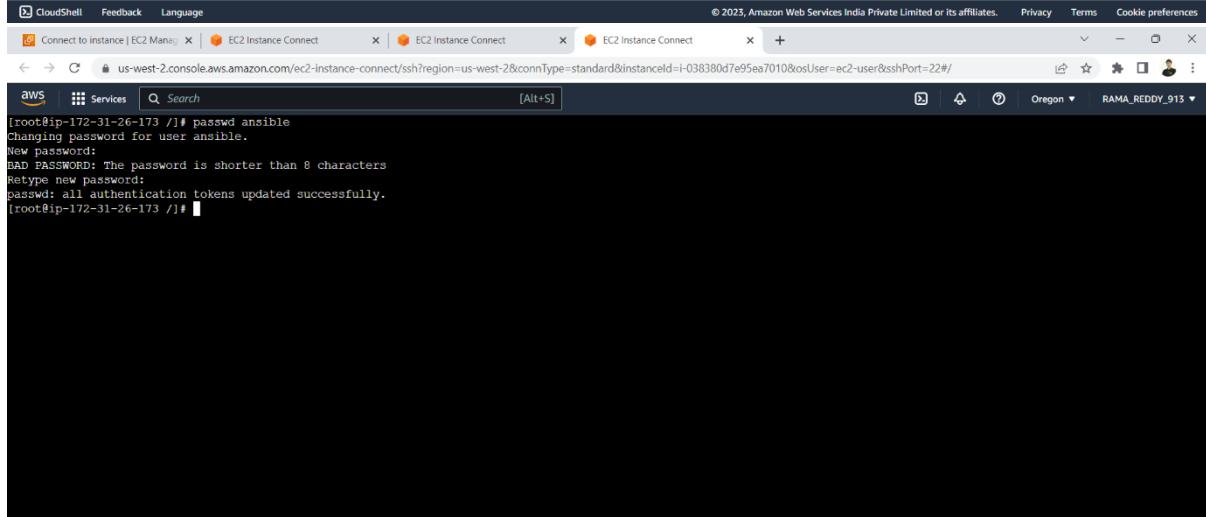
```

root:x:0:root:/root:/bin/bash
bin:x:1:bin:/bin:/sbin/nologin
daemon:x:2:daemon:/sbin:/sbin/nologin
adm:x:3:adm:/var/adm:/sbin/nologin
lp:x:4:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:sync:/sbin:/bin/sync
shutdown:x:6:shutdown:/sbin:/sbin/shutdown
halt:x:7:halt:/sbin:/sbin/halt
mail:x:8:mail:/var/spool/mail:/sbin/nologin
operator:x:11:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/sbin/nologin
systemd-network:x:192:192:Network Management:/sbin/nologin
ibus:x:81:81:System message bus:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
libstoragemgmt:x:999:997:daemon account for libstoragemgmt:/var/run/lsm:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:anonymous NFS User:/var/lib/nfs:/sbin/nologin
rngd:x:998:994:Random Number Generator Daemon:/var/lib/rngd:/sbin/nologin
chrony:x:997:995:/var/lib/chrony:/sbin/nologin
ec2-instance-connect:x:996:994:/home/ec2-instance-connect:/sbin/nologin
postfix:x:69:83:/var/spool/postfix:/sbin/nologin
tcpdump:x:72:74::/sbin/nologin
ec2-user:x:1001:1001:/home/ec2-user:/bin/bash
ansible:x:1001:1001:/home/ansible:/bin/bash
[root@ip-172-31-26-173 ~]#

```

i-038380d7e95ea7010 (node2)

PublicIPs: 35.93.146.2 PrivateIPs: 172.31.26.173



```

[root@ip-172-31-26-173 ~]# passwd ansible
Changing password for user ansible.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: all authentication tokens updated successfully.
[root@ip-172-31-26-173 ~]#

```

i-038380d7e95ea7010 (node2)

PublicIPs: 35.93.146.2 PrivateIPs: 172.31.26.173

i-038380d7e95ea7010 (node2)

PublicIPs: 35.93.146.2 PrivateIPs: 172.31.26.173

CloudShell Feedback Language

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

aws Services Search [Alt+S]

[root@ip-172-31-26-173 ~]# visudo

```
Defaults    env_keep += "LC_TIME LC_ALL LANGUAGE LINGUAS XKB_CHARSET XAUTHORITY"

## Adding HOME to env_keep may enable a user to run unrestricted
## commands via sudo.
## Defaults    env_keep += "HOME"

Defaults    secure_path = /sbin:/bin:/usr/sbin:/usr/bin

## Next comes the main part: which users can run what software on
## which machines (the sudoers file can be shared between multiple
## systems).
## Syntax:
##       user      MACHINE=COMMANDS
##       The COMMANDS section may have other options added to it.
##       Allow root to run any commands anywhere
root      ALL=(ALL)      ALL
ansible ALL=(ALL)  NOPASSWD: ALL

## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES, LOCATE, DRIVERS

-- INSERT --

```

i-038380d7e95ea7010 (node2)

PublicIPs: 35.93.146.2 PrivateIPs: 172.31.26.173

CloudShell Feedback Language

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

The screenshot shows three stacked terminal windows from the AWS CloudShell interface. The top window displays the command `vi /etc/ssh/sshd_config`. The middle window shows the contents of the `/etc/ssh/sshd_config` file, which includes various SSH configuration parameters. The bottom window shows the command prompt again.

```
[root@ip-172-31-26-173 ~]# vi /etc/ssh/sshd_config
```

```
#MaxSessions 10

#PubkeyAuthentication yes

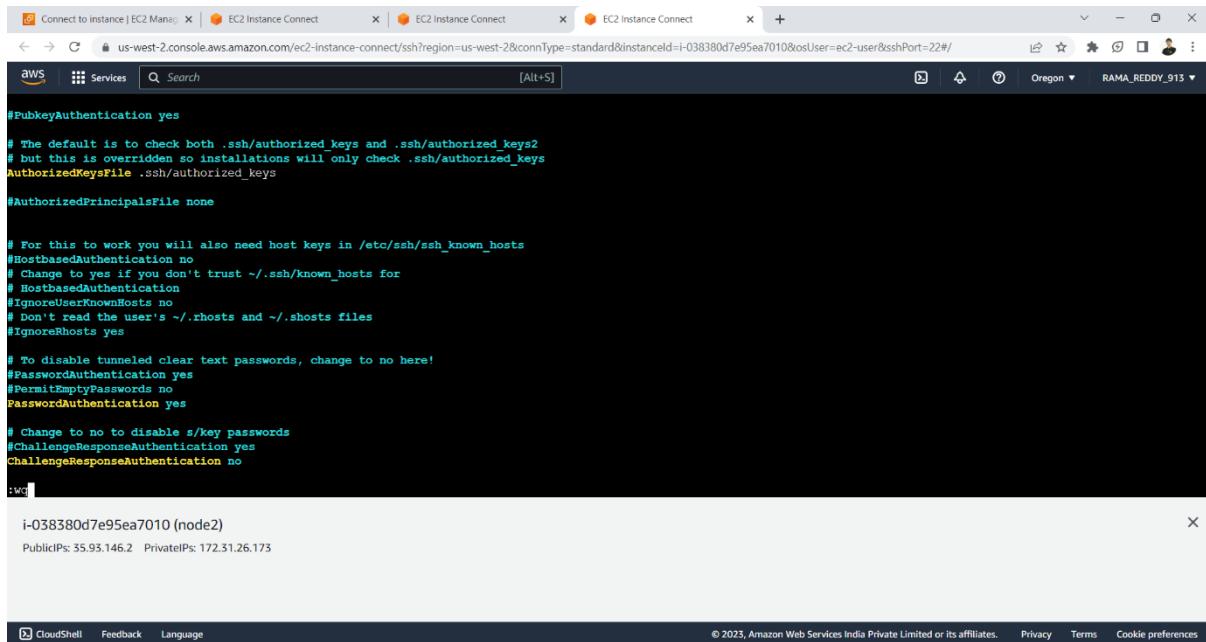
# The default is to check both .ssh/authorized_keys and .ssh/authorized_keys2
# but this is overridden so installations will only check .ssh/authorized_keys
AuthorizedKeysFile .ssh/authorized_keys

#AuthorizedPrincipalsFile none

# 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

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no
PasswordAuthentication no

# Change to no to disable s/key passwords
#ChallengeResponseAuthentication yes
ChallengeResponseAuthentication no
```



```
#PubkeyAuthentication yes
# The default is to check both .ssh/authorized_keys and .ssh/authorized_keys2
# but this is overridden so installations will only check .ssh/authorized_keys
AuthorizedKeysFile .ssh/authorized_keys

#AuthorizedPrincipalsFile none

# 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

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no
#PasswordAuthentication yes

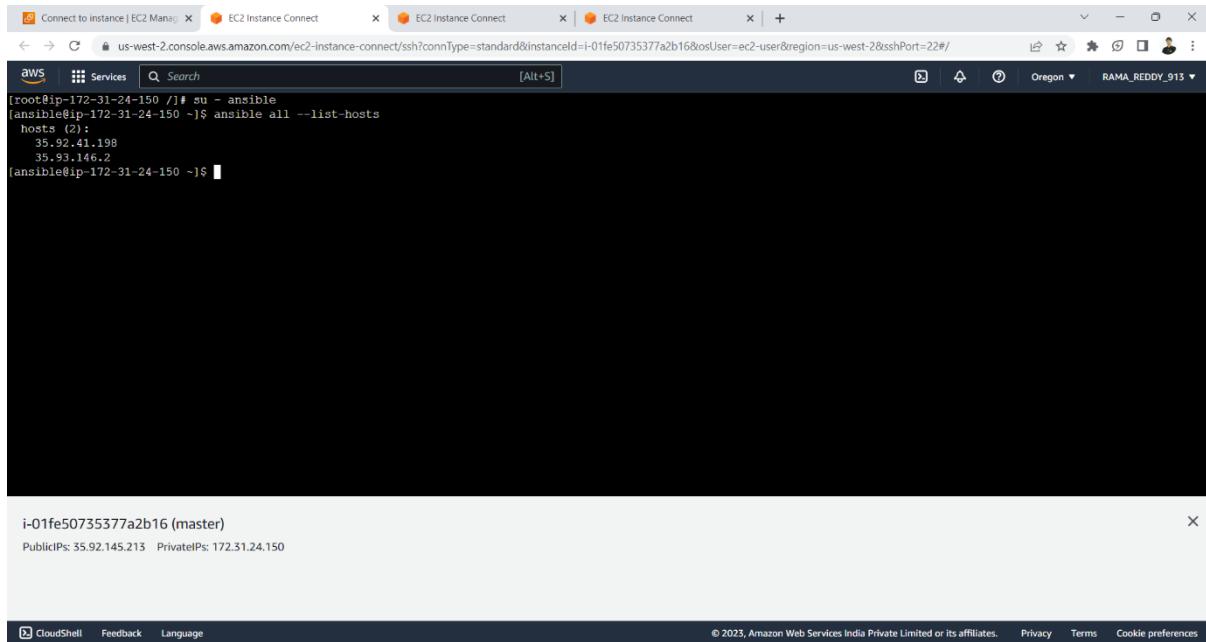
# Change to no to disable s/key passwords
#ChallengeResponseAuthentication yes
#ChallengeResponseAuthentication no
```

rwq []

i-038380d7e95ea7010 (node2)

PublicIPs: 35.93.146.2 PrivateIPs: 172.31.26.173

switch to ansible and check the hosts by executing the following commands

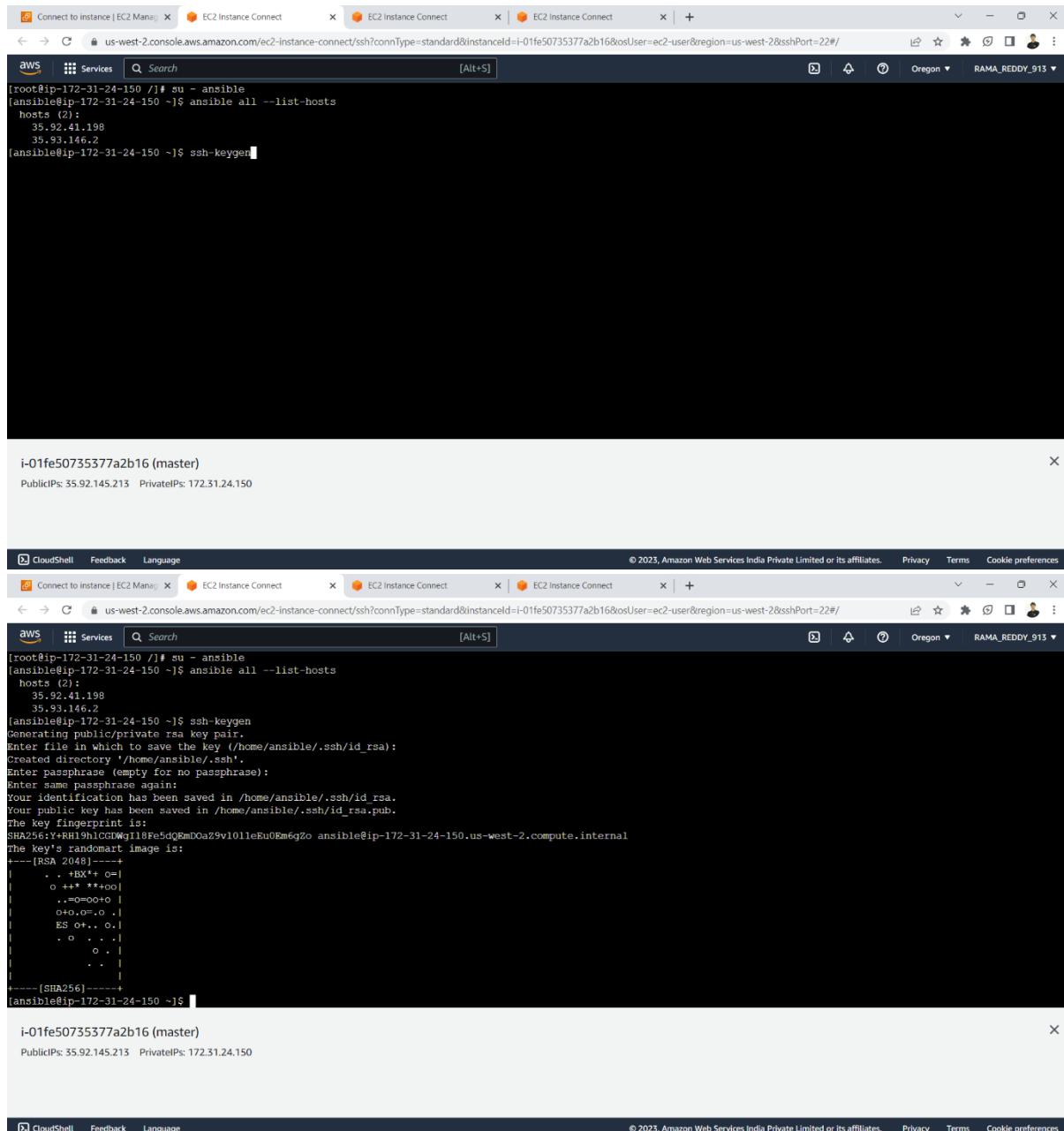


```
[root@ip-172-31-24-150 ~]# su - ansible
(ansible@ip-172-31-24-150 ~)$ ansible all --list-hosts
 hosts (2):
   35.92.41.198
   35.93.146.2
(ansible@ip-172-31-24-150 ~)$
```

i-01fe50735377a2b16 (master)

PublicIPs: 35.92.145.213 PrivateIPs: 172.31.24.150

create a public key by using ssh-keygen command



The screenshot shows three tabs in the AWS CloudShell interface:

- Connect to Instance | EC2 Manager
- EC2 Instance Connect
- EC2 Instance Connect

The main terminal window displays the following command sequence:

```
[root@ip-172-31-24-150 ~]# su - ansible
(ansible@ip-172-31-24-150 ~)$ ansible all --list-hosts
hosts (2):
  35.92.41.198
  35.93.146.2
(ansible@ip-172-31-24-150 ~)$ ssh-keygen
```

Below the terminal, the instance details are shown:

i-01fe50735377a2b16 (master)
PublicIPs: 35.92.145.213 PrivateIPs: 172.31.24.150

At the bottom of the terminal window, the generated RSA key fingerprint is displayed:

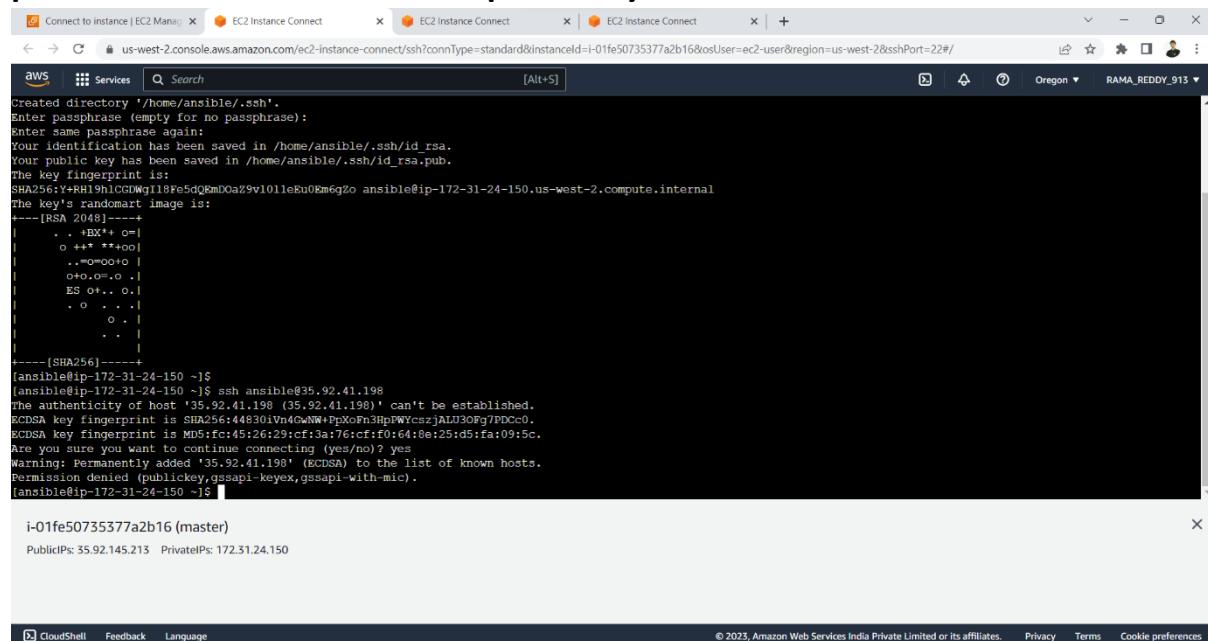
```
SHA256:Y+RH19hIC0Wg1l8Fe5dQRmDOaZ9v101leEu0Erm6g2o ansible@ip-172-31-24-150.us-west-2.compute.internal
```

The terminal concludes with the message:

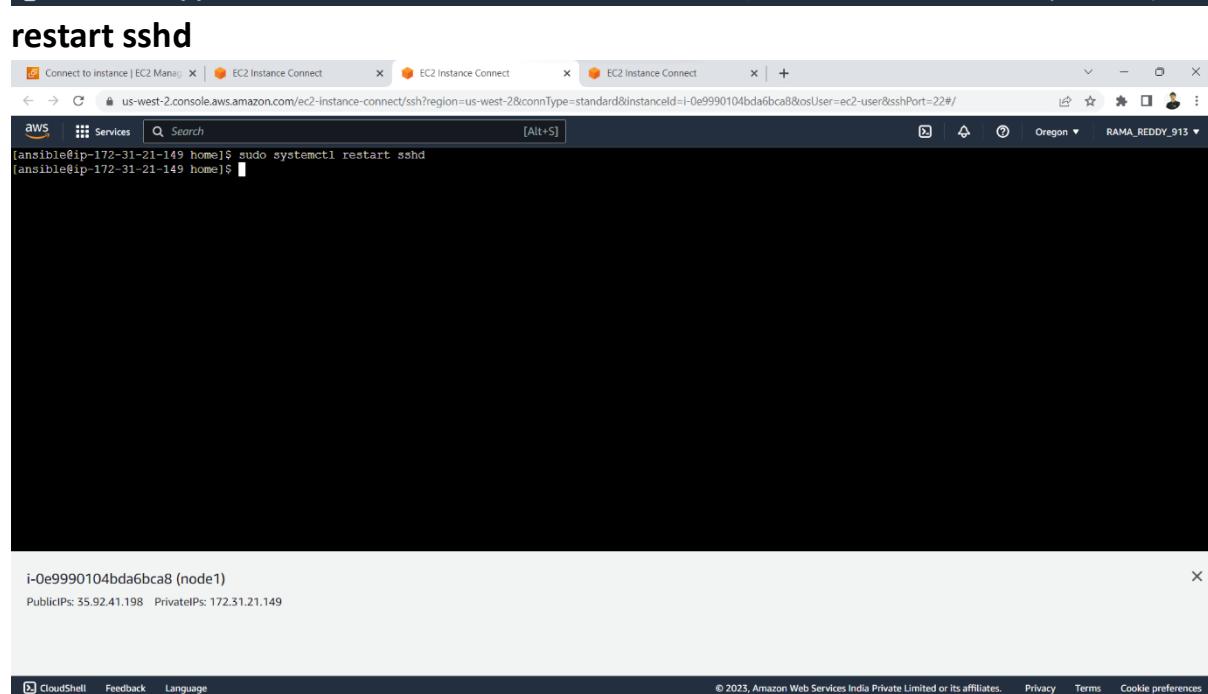
```
the key's randomart image is:
```

-----[RSA 2048]-----
| .. +BX*+o=|
| o ++* **+oo|
| ..+o=o+o+o|
| o+o.o=o .o.|
| ES o+.. o.|
| . o . . .|
| . o . . .|
||
||
-----[SHA256]-----
(ansible@ip-172-31-24-150 ~)\$

connect to the node by using ssh command here we have got an error as permission denied because the public key is not available in the nodes



```
Created directory '/home/ansible/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ansible/.ssh/id_rsa.
Your public key has been saved in /home/ansible/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:Y+RH19hICCDWg18Fe5dQKmD0aZ9v10lleEu0Km6gZo ansible@ip-172-31-24-150.us-west-2.compute.internal
The key's randomart image is:
+---[RSA 2048]----+
| .. +BX** o=|
| o ++* **+oo|
| ..+*o*o+o |
| o+o..o= o .|
| ES o+.. o.|
| . o . . .|
| . o . |
| . . . |
| . . . |
+---[SHA256]----+
(ansible@ip-172-31-24-150 ~)$
(ansible@ip-172-31-24-150 ~)$ ssh ansible@35.92.41.198
The authenticity of host '35.92.41.198 (35.92.41.198)' can't be established.
ECDSA key fingerprint is SHA256:44830iVnQoNW+PpXoFn3HpwYcszjAlU3OFg7PDcc0.
ECDSA key fingerprint is MD5:fc45:26:29:cfc1:a76:cff0:64:8e:25:d5:f1:a0:95:c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '35.92.41.198' (ECDSA) to the list of known hosts.
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
(ansible@ip-172-31-24-150 ~)$
```



```
i-01fe50735377a2b16 (master)
PublicIPs: 35.92.145.213 PrivateIPs: 172.31.24.150

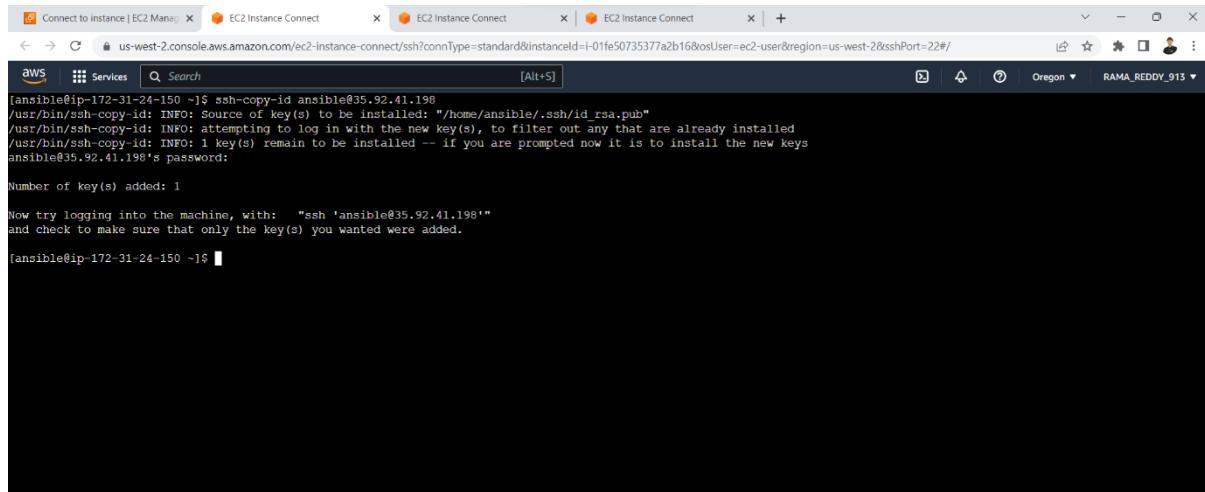
CloudShell Feedback Language
© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences
```

```
restart sshd
[Ansible@ip-172-31-21-149 ~]$ sudo systemctl restart sshd
[Ansible@ip-172-31-21-149 ~]$
```

```
i-0e9990104bda6bca8 (node1)
PublicIPs: 35.92.41.198 PrivateIPs: 172.31.21.149

CloudShell Feedback Language
© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences
```

copy public key generated to the nodes by using following command and now we can connect to the node securely by using ssh command



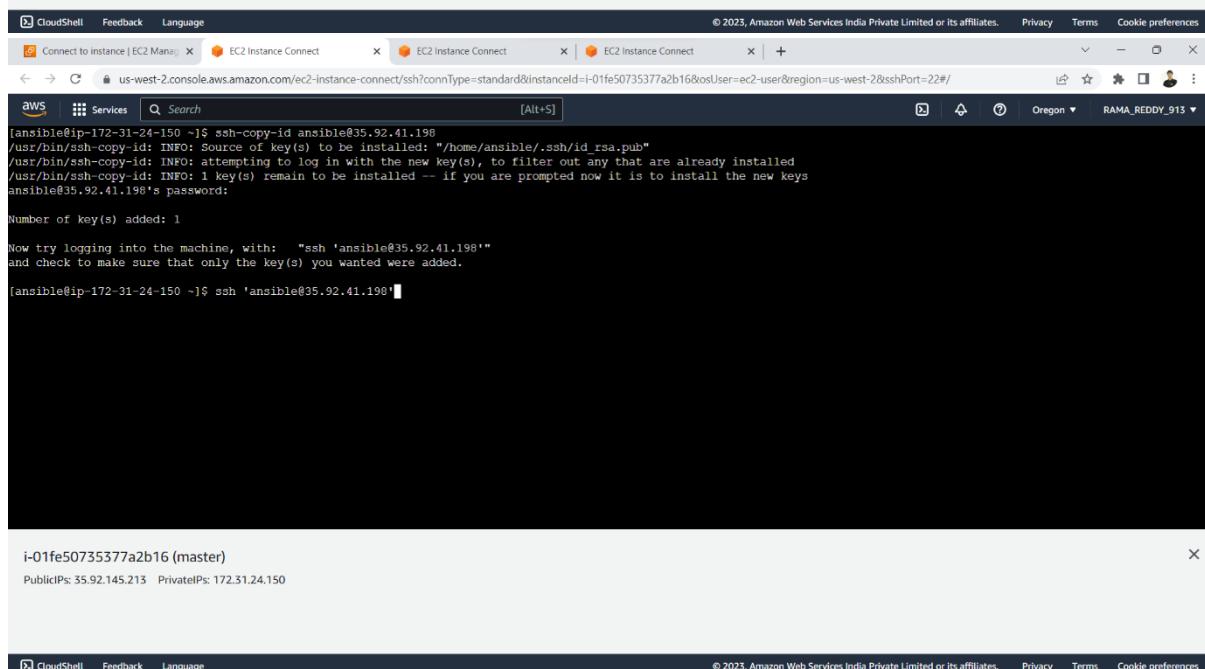
```
[ansible@ip-172-31-24-150 ~]$ ssh-copy-id ansible@35.92.41.198
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansible/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
ansible@35.92.41.198's password:
Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'ansible@35.92.41.198'"
and check to make sure that only the key(s) you wanted were added.

[ansible@ip-172-31-24-150 ~]$
```

i-01fe50735377a2b16 (master)

PublicIPs: 35.92.145.213 PrivateIPs: 172.31.24.150



```
[CloudShell] Feedback Language
[EC2 Instance Connect] EC2 Instance Connect [EC2 Instance Connect]
© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences
[aws] Services Search [Alt+S]
[ansible@ip-172-31-24-150 ~]$ ssh ansible@35.92.41.198
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansible/.ssh/id_rsa.pub"
/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
ansible@35.92.41.198's password:
Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'ansible@35.92.41.198'"
and check to make sure that only the key(s) you wanted were added.

[ansible@ip-172-31-24-150 ~]$ ssh 'ansible@35.92.41.198'
```

i-01fe50735377a2b16 (master)

PublicIPs: 35.92.145.213 PrivateIPs: 172.31.24.150

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences


```

https://aws.amazon.com/amazon-linux-2/
[ansible@ip-172-31-21-149 ~]$ exit
logout
Connection to 35.92.41.198 closed.
[ansible@ip-172-31-24-150 ~]$ ssh-copy-id ansible@35.93.146.2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansible/.ssh/id_rsa.pub"
The authenticity of host '35.93.146.2' (35.93.146.2) can't be established.
ECDSA key fingerprint is SHA256:aj5MrxCXqyKL15TTl5E1xqkX18jeuucGH48W4Q8.
ECDSA key fingerprint is MD5:55::c5:61:0a:9a:ed:4a:75:af:41:cc:29:f9:d9:42:69.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: 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
ansible@35.93.146.2's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'ansible@35.93.146.2'"
and check to make sure that only the key(s) you wanted were added.

[ansible@ip-172-31-24-150 ~]$ ssh 'ansible@35.93.146.2'
Last login: Fri Apr 7 16:54:55 2023
[  ] ( [  ] / Amazon Linux 2 AMI
[  ] \ [  ] [  ] [  ]
https://aws.amazon.com/amazon-linux-2/
[ansible@ip-172-31-26-173 ~]$ [  ]
```

i-01fe50735377a2b16 (master)
PublicIPs: 35.92.145.213 PrivateIPs: 172.31.24.150

the following yaml code will install git ,http on the nodes , starts the httpd server and clones the website application and copies it to /var/www/html

```

https://aws.amazon.com/amazon-linux-2/
[ansible@ip-172-31-24-150 ~]$ cat sam.yml
YAML Syntax Error Fix
[Alt+S]
```

```

---
- name: Deploy website from Git repository to HTTP server
  hosts: developer
  become: true
  vars:
    repo_url: "https://github.com/sunil-1508/cloud_storms.git"
    web_root: "/var/www/html"
  tasks:
    - name: Install Git on HTTP server
      yum:
        name: git
        state: present

    - name: Clone Git repository
      git:
        repo: "{{ repo_url }}"
        dest: "{{ web_root }}"
        clone: true

    - name: Set ownership of web root directory
      file:
        path: "{{ web_root }}"
        owner: apache
        group: apache
        mode: 0755

    - name: Install Apache web server
      sam.yml: 36L, 790B
```

i-01fe50735377a2b16 (master)
PublicIPs: 35.92.145.213 PrivateIPs: 172.31.24.150

run the ansible playbook by using ansible-playbook filename.yml

The screenshot shows a terminal window within the AWS CloudShell interface. The terminal title is 'aws' and the tab bar includes 'Connect to Instance | EC2 Manager', 'EC2 Instance Connect', 'EC2 Instance Connect', 'YAML Syntax Error Fix', and 'Oregon RAMA_RED俞_913'. The terminal content displays the execution of an Ansible playbook named 'sam.yml' on an EC2 instance. The output shows the playbook running through various tasks including gathering facts, installing Git, and cloning a repository. It also includes several warning messages about Python interpreters.

```
[ansible@ip-172-31-24-150 ~]$ vi sam.yml
[ansible@ip-172-31-24-150 ~]$ ansible-playbook sam.yml
[WARNING]: could not match supplied host pattern, ignoring: developer

PLAY [Deploy website from Git repository to HTTP server] ****
  skipping: no hosts matched

PLAY RECAP
[ansible@ip-172-31-24-150 ~]$ vi sam.yml
[ansible@ip-172-31-24-150 ~]$ ansible-playbook sam.yml

PLAY [Deploy website from Git repository to HTTP server] ****

TASK [Gathering Facts] ****
[WARNING]: Platform linux on host 35.92.41.198 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [35.92.41.198]
[WARNING]: Platform linux on host 35.93.146.2 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Python interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [35.93.146.2]

TASK [Install Git on HTTP server] ****
ok: [35.92.41.198]
ok: [35.93.146.2]

TASK [Clone Git repository] ****
i-01fe50735377a2b16 (master)

PublicIPs: 35.92.145.213 PrivateIPs: 172.31.24.150
```

now we can connect ot the website by using public ip address of the hosts

The screenshots show the Facebook login page loaded via its public IP address. The top one is a standard log-in, while the bottom one includes a 'Create a page for a celebrity, brand or business' button.

Facebook - Log in or Sign up

Email or Phone number

Password

Login

Forgot password ?

Create New Account

Create a page for a celebrity, brand or business

Recent Logins

Click your picture or add an account

Mark

Add Account

English (UK) हिन्दी ਪੰਜਾਬ ਮਰਾਠੀ ਤਾਮਿਲ ਬਾਰਾਂ ਸ਼ਹਿਰੀ ਫੇਵਰੋਂ ਮੁਖਹਾਂ ਕਨੂੰਹ ਮਿਆਡੂ

Sign Up Log In Messenger DotNetTec Mobile Find Friends Badges People Pages Places Games

Locations Celebrities Groups Moments About Create Advert Create Page Developers Careers Privacy Cookies

Ads Terms Help

REPOSITORY DETAILS:

The frontend Facebook clone web application and YAML Playbooks we used in this project will be available in the following git hub repository

https://github.com/Sunil-1508/cloud_storms

OUTCOME:

Improved efficiency: Ansible can automate repetitive tasks, such as software installation, configuration management, and infrastructure provisioning. This can save time and reduce the risk of errors that can occur with manual processes. As a result, IT teams can be more efficient and productive, focusing on more strategic work that adds value to the organization.

Increased consistency: Automation with Ansible ensures that IT operations are standardized and consistent, reducing the risk of errors and security vulnerabilities. Ansible allows you to define playbooks and roles that can be used across your organization, ensuring that tasks are executed in a consistent manner. This can help to improve the reliability and stability of your IT environment.

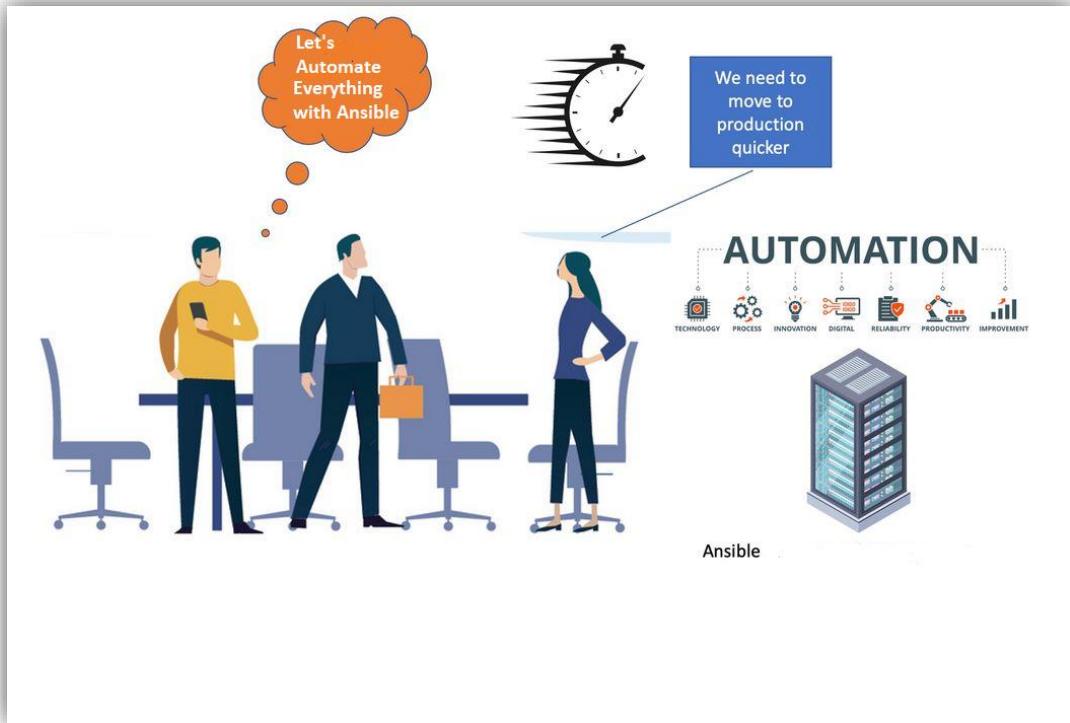
Enhanced scalability: Ansible is highly scalable and can automate tasks across large numbers of servers and devices. This makes it a valuable tool for organizations that need to manage complex IT environments. Ansible allows you to define groups of hosts and execute tasks across those groups, allowing you to easily scale your automation efforts as your environment grows.

Reduced costs: By automating routine tasks, Ansible can help organizations save time and reduce labour costs associated with manual processes. Ansible can also help organizations to avoid costly downtime by automating tasks such as system patching and updates.

Improved collaboration: Ansible provides a platform for collaboration between IT teams, allowing for the sharing of playbooks and roles that can be used across the organization. This can help to improve communication and coordination between teams, leading to a more efficient and effective IT operation.

Enhanced security: Ansible can help organizations maintain a strong security posture by automating tasks such as patch management, vulnerability scanning, and compliance auditing. Ansible can also help to enforce security policies across your environment by automating the configuration of firewalls, security groups, and other security-related tasks.

In summary, automating applications using Ansible can lead to significant improvements in efficiency, consistency, scalability, cost savings, collaboration, and security. These outcomes can help organizations to streamline their IT operations, improve productivity, and reduce the risk of downtime and security breaches.



CONCLUSION:

In conclusion, automating applications using Ansible can provide numerous benefits for organizations looking to improve their IT operations. Ansible is a powerful open-source automation tool that can be used to automate a wide range of tasks, from software installation and configuration management to infrastructure provisioning and deployment.

By automating routine tasks, organizations can improve efficiency, consistency, and scalability, while reducing the costs associated with manual processes. Ansible can also help enhance collaboration between IT teams, allowing for the sharing of playbooks and roles that can be used across the organization.

Additionally, Ansible can help organizations maintain a strong security posture by automating tasks such as patch management, vulnerability scanning, and compliance auditing. By leveraging Ansible's capabilities, organizations can achieve greater efficiency, cost savings, and security, while freeing up IT teams to focus on more strategic work.

The use cases for automating applications using Ansible are diverse and can be applied to a wide range of industries and organizations. From configuration management and continuous deployment to cloud automation and disaster recovery, Ansible can help organizations streamline their IT operations and achieve their business objectives.

Overall, the outcome of automating applications using Ansible can be significant improvements in IT operations, leading to a more streamlined and effective IT environment. By leveraging Ansible's capabilities, organizations can achieve greater efficiency, cost savings, and security, while freeing up IT teams to focus on more strategic work.

REFERENCES:

- Ansible Community Documentation(<https://docs.ansible.com/>)
- Git Tutorial(<https://git-scm.com/docs>)
- Ansible Tutorials (<https://youtu.be/zw8eUSGNLq4>)
- AWS Documentation (<https://docs.aws.amazon.com/>)

CONTACT:

VISHAL BATTULA

Ph. No: 9491308212

Gmail: vishalbattula4194@gmail.com