Ansible

What is Ansible?

Ansible is an open-source automation tool used for application deployment, cloud provisioning, intraservice orchestration, and other IT tasks. It operates by connecting to nodes through SSH, enabling agentless execution and ensuring security without installing additional software on managed hosts. With its declarative language, YAML, Ansible simplifies task and configuration definitions, facilitating efficient automation of complex IT workflows with minimal coding required.

Benefits of Ansible

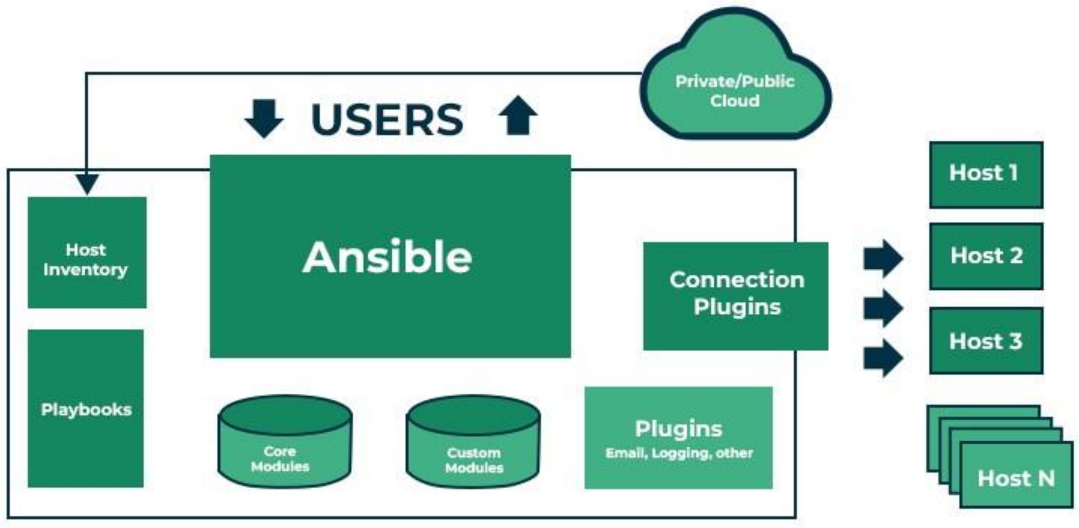
- Free: Ansible is an open-source tool.
- Very simple to set up and use: No special coding skills are necessary to use Ansible's playbooks (more on playbooks later).
- Powerful: Ansible lets you model even highly complex IT workflows.
- Flexible: You can orchestrate the entire application environment no matter where it's deployed. You can also customize it based on your needs.
- Agentless: You don't need to install any other software or firewall ports on the client systems you want to automate. You also don't have to set up a separate management structure.
- Efficient: Because you don't need to install any extra software, there's more room for application resources on your server.

Ansible can be used to deploy the software on different servers at a time without human interaction. Ansible can also be used to configure the servers and create user accounts. Ansible is an agent-less software which means there is no need to install the software in the nodes which means you need to do the SSH to connect the nodes to perform the required operations on the servers.
Ansible playbooks can be written very easily they will be like plain English and Ansible is developed using Python language. There is no need for any knowledge of programming. A single ansible control node can manage thousands of the nodes.

Ansible Architecture

1. **Control node:** Commands and Playbooks can run by invoking /usr/bin/ansible or /usr/bin/ansible-playbook, from any control node. You can use any computer that has Python installed on it as a control node. However, one cannot use a computer with Windows OS as a control node. One can have multiple control nodes.
2. **Managed nodes:** Also, sometimes called "hosts", Managed nodes are the network devices (and/or servers) you manage with Ansible.

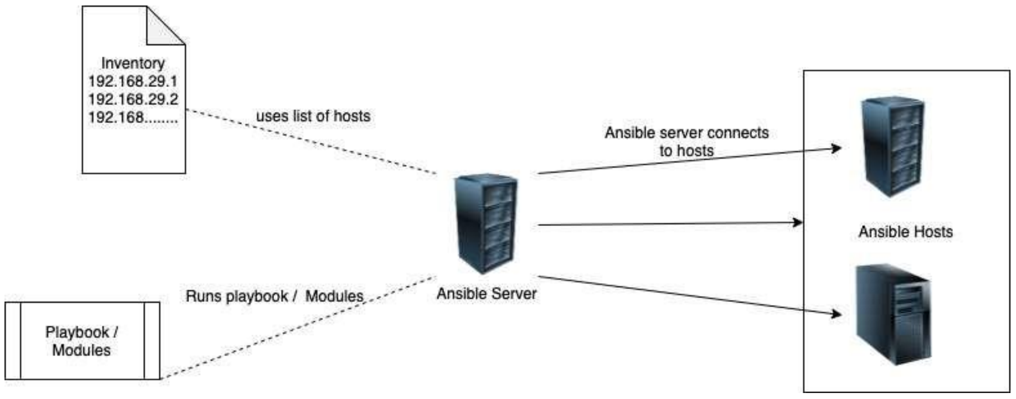
3. **Inventory:** Also sometimes called “host file”, Inventory is the list of Managed nodes use to organize them. It is also used for creating and nesting groups for easier scaling.
4. **Modules:** These are the units of code executed by Ansible. Each module can be used for a specific purpose. One can invoke a single module with a task, or invoke several different modules in a playbook.
5. **Tasks:** The units of action in Ansible. One can execute a single task once with an adhoc command.



KeyTerminologies:

It is important to understand these terminologies before diving into the topic deeply.

- **Ansible Server:** The node where Ansible is installed. This is the node from which orchestration is done.
- **Playbook:** It is a set icon commands written in YAML that describes what should be done and where it should be done.
- **Tasks:** A task is a single procedure that should be done.
- **Role:** This is a way of encapsulating tasks and files related to them so that they can be later used in the playbook.
- **Inventory:** It is a file containing the details about the client nodes.
- **Handler:** A handler is a task that is called when a notifier (similar to a trigger in other technologies) is present in the playbook.
- **Module:** A unit of code or binary that Ansible runs on managed nodes.
- **Ansible nodes/hosts:** The independent systems that are managed by the Ansible server are called nodes or hosts.



Ansible Playbook with example

These are the ordered list of tasks that are saved so you can run those tasks in that order repeatedly. Playbooks are written in YAML and are easy to read, write, share and understand. Ansible playbooks can perform wide variety of tasks as mentioned below

1. Deploying and configuring applications
2. Managing system configurations
3. Orchestrating complex workflows

Example

The language used to write the ansible playbooks was YAML which is human readable. Following sections will consists in ansible playbook.

```
---
-         hosts: all   tasks:
-         name: Install the
Apache web server       apt:
       name: apache2
state: present
```

1. **Hosts:** Host in ansible play book defines no. of remote host that playbook will executed and performs the task specified.
2. **Vars:** Vars in ansible playbook defines the variables which can be used through out the playbook.
3. **Tasks:** This is the section where we are going tot mention the type of task to be executed in the host servers.

 To run your playbook, use the ansible-playbook command.

```
ansible-playbook playbook.yml
```

Ansible - Ad hoc Commands

Ad hoc commands are commands which can be run individually to perform quick functions. These commands need not be performed later.

For example, you have to reboot all your company servers. For this, you will run the Adhoc commands from '**/usr/bin/ansible**'.

These ad-hoc commands are not used for configuration management and deployment, because these commands are of one-time usage.

ansible-playbook is used for configuration management and deployment.

Parallelism and Shell Commands

Reboot your company server in 12 parallel forks at time. For this, we need to set up SSHagent for connection.

```
$ ssh-agent bash
$ ssh-add ~/.ssh/id_rsa
```

To run reboot for all your company servers in a group, 'abc', in 12 parallel forks –

```
$ Ansible abc -a "/sbin/reboot" -f 12
```

By default, Ansible will run the above Ad-hoc commands form current user account. If you want to change this behaviour, you will have to pass the username in Ad-hoc commands as follows –

```
$ Ansible abc -a "/sbin/reboot" -f 12 -u username
```

File Transfer

You can use the Ad-hoc commands for doing **SCP** (Secure Copy Protocol) lots of files in parallel on multiple machines.

Transferring file to many servers/machines

```
$ Ansible abc -m copy -a "src = /etc/yum.conf dest = /tmp/yum.conf"
```

Creating new directory

$ Ansible abc -m file -a "dest = /path/user1/new mode = 777 owner = user1 group = user1 state = directory"

Deleting whole directory and files

```
$ Ansible abc -m file -a "dest = /path/user1/new state = absent"
```

Managing Packages

The Ad-hoc commands are available for yum and apt. Following are some Ad-hoc commands using yum.

The following command checks if yum package is installed or not, but does not update it.

```
$ Ansible abc -m yum -a "name = demo-tomcat-1 state = present"
```

The following command check the package is not installed.

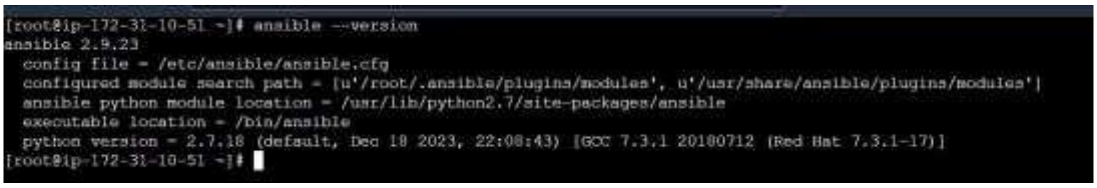
```
$ Ansible abc -m yum -a "name = demo-tomcat-1 state = absent"
```

The following command checks the latest version of package is installed.

```
$ Ansible abc -m yum -a "name = demo-tomcat-1 state = absent"
```

Installing And Configuring Ansible :

Step 1. Create one main ansible instance and 2-3 node instance. Now in main ansible --> Go to root use: sudo -i
Step 2. Now install ansible use the command: amazon-linux-extras install ansible2 y
Step 3. Now check the ansible version: ansible –version



```
[root@ip-172-31-10-51 ~]# ansible --version
ansible 2.9.23
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible
  executable location = /bin/ansible
  python version = 2.7.18 (default, Dec 18 2023, 22:08:43) [GCC 7.3.1 20180712 (Red Hat 7.3.1-17)]
[root@ip-172-31-10-51 ~]# 
```

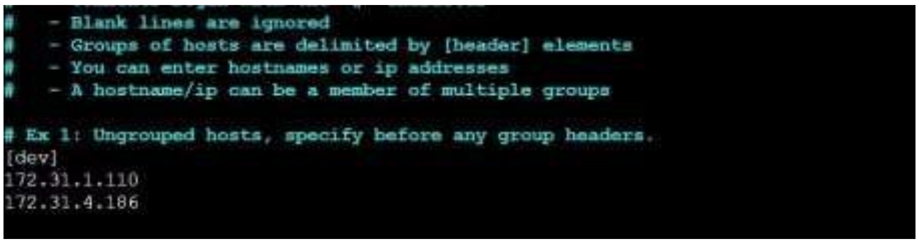
Step 4. Install python use : yum install git python python-pip python-level openssl -y

Step 5. Now check the python version : python --version

Step 6. vi /etc/ansible/hosts : using this command, create a cluster or user(ex:dev) and then, Add private IP address of another instance in this file. This will create nodes.

ex: *user_defined_name+ -- name of the cluster

Private IP address of atleast 2 nodes



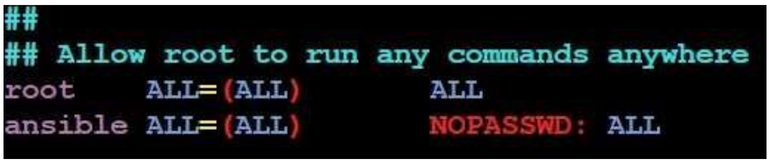
```
#    - Blank lines are ignored
#    - Groups of hosts are delimited by [header] elements
#    - You can enter hostnames or ip addresses
#    - A hostname/ip can be a member of multiple groups

# Ex 1: Ungrouped hosts, specify before any group headers.
[dev]
172.31.1.110
172.31.4.186
```

Step 7. vi /etc/ansible/ansible.cfg: using this command Comment out Inventary and sudo_user.

Step 8. Now add a user, use the command: useradd ansible

Step 9. Set a password and remember it (to use in nodes aswell) use the command: passwd ansible

Step 10. Make it super user and in root add ansible ALL=(ALL) NOPASSWD: ALL
 use the command: visudo

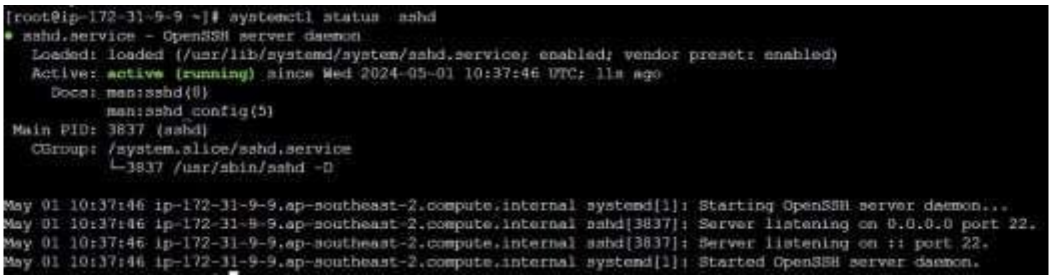


```
##
## Allow root to run any commands anywhere
root     ALL=(ALL)        ALL
ansible  ALL=(ALL)        NOPASSWD: ALL
```

step 11. Now change the password authentication from no to yes, use the command : vim /etc/ssh/sshd_config

step 12. Now restart ssh, use the command: systemctl restart sshd (or) service sshd restart

step 13. Now check the status of ssh, use the command: systemctl status ssh (or) service ssh status



```
[root@ip-172-31-9-9 ~]# systemctl status  sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2024-05-01 10:37:46 UTC; 11s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 3837 (sshd)
   CGroup: /system.slice/sshd.service
           └─3837 /usr/sbin/sshd -D

May 01 10:37:46 ip-172-31-9-9.ap-southeast-2.compute.internal systemd[1]: Starting OpenSSH server daemon...
May 01 10:37:46 ip-172-31-9-9.ap-southeast-2.compute.internal sshd[3837]: Server listening on 0.0.0.0 port 22.
May 01 10:37:46 ip-172-31-9-9.ap-southeast-2.compute.internal sshd[3837]: Server listening on :: port 22.
May 01 10:37:46 ip-172-31-9-9.ap-southeast-2.compute.internal systemd[1]: Started OpenSSH server daemon.
```

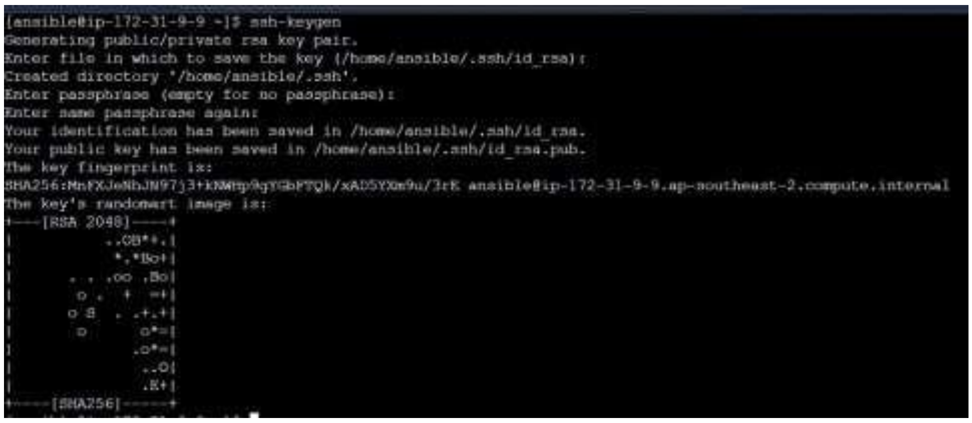
Step 14. Go to ansible user, use the command: su – ansible

Step 15. Make sure you do both nodes configuration before creating key

- sudo -i
- useradd ansible
- passwd ansible

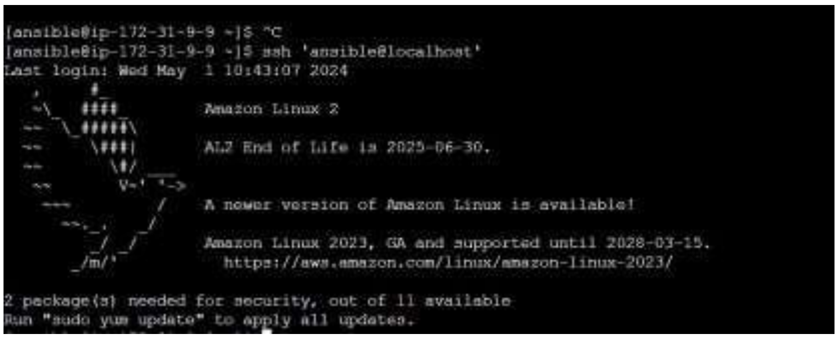
- visudo -- after root add -- ansible ALL=(ALL) NOPASSWD: ALL
- vim /etc/ssh/sshd_config -- goto passwd and make it replace no with yes
- systemctl restart ssh
- systemctl status ssh

Step 16. Create a key use the command: ssh-keygen



```
[ansible@ip-172-31-9-9 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ansible/.ssh/id_rsa):
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:MnfXJeNbJN97j3+kNWHp9gYGbFTQk/xADSYXm9u/3rK ansible@ip-172-31-9-9.ap-southeast-2.compute.internal
The key's randomart image is:
+---[RSA 2048]----+
|         ..OB*+.|
|          *.*Bo+|
|     . . .oo .Bo|
|      o .  +  =+|
|     o S  . .+.+|
|      o      o*=|
|            .o*=|
|             ..O|
|             .E+|
+----[SHA256]-----+
```

Step 17. ssh-copyid ansible@localhost: give same password while authenticating and then copy from above (ssh 'ansible@localhost')



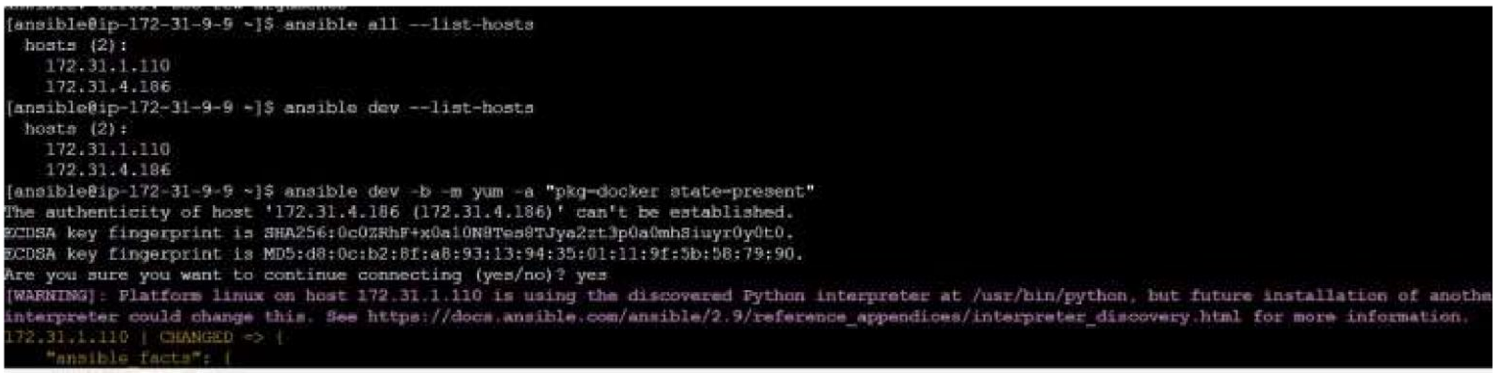
```
[ansible@ip-172-31-9-9 ~]$ ^C
[ansible@ip-172-31-9-9 ~]$ ssh 'ansible@localhost'
Last login: Wed May  1 10:43:07 2024
   ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2025-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

2 package(s) needed for security, out of 11 available
Run "sudo yum update" to apply all updates.
```

Step 18. Now copy the ssh-copy-id ansible@publicipofnode1
Step 19. Now copy from above: ssh 'ansible@publicIPnode1'
Step 20. Do the same for other nodes too,
-->ssh-copy-id ansible@publicIDofnode2
-->ssh 'ansible@publicIPnode2'
Step 21. Make sure you do these nodes configuration before creating key
Step 22. Now come to main ansible to create a playbook
Step 23. use this command to list the hosts: ansible all --list-hosts

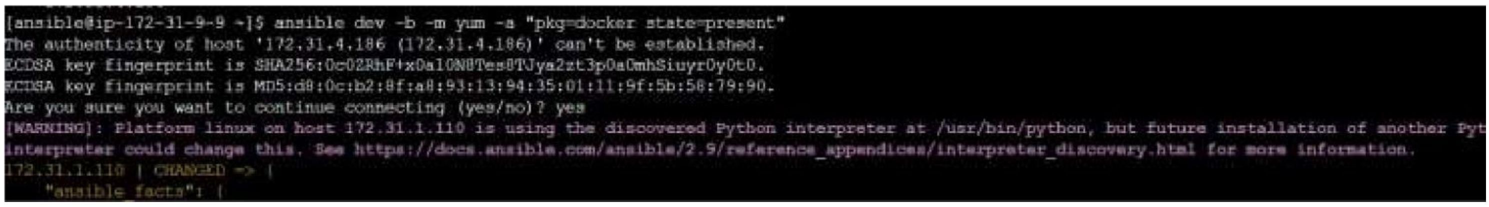
Step 24. Use this command: ansible dev --list-hosts

Step 25. use this command to create module: ansible dev -b -m yum -a "pkg=docker state=present"

```
[illegible]
```

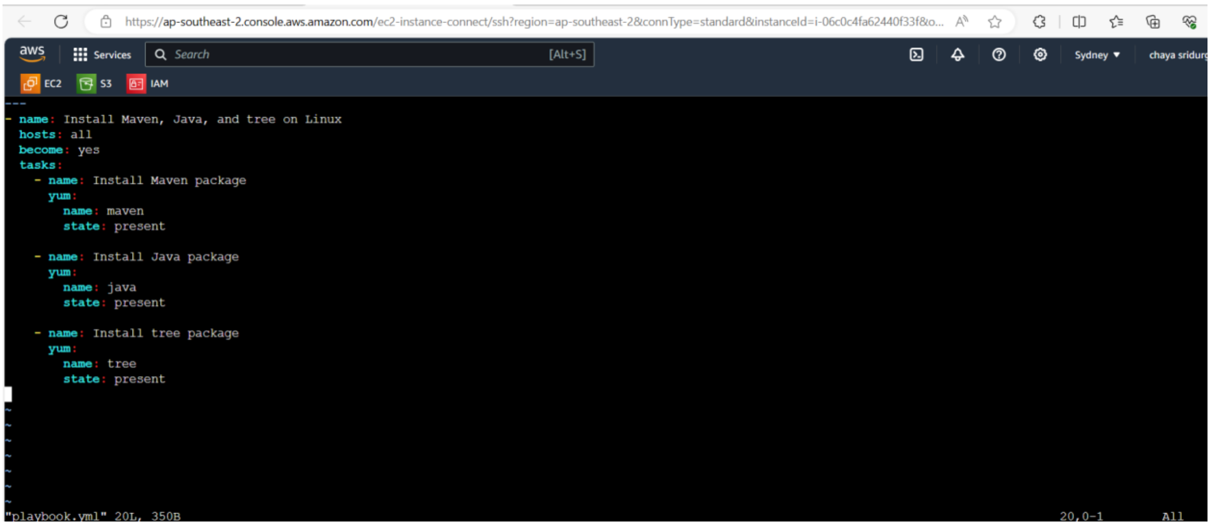


```
[ansible@ip-172-31-9-9 ~]$ ansible all --list-hosts
  hosts (2):
    172.31.1.110
    172.31.4.186
[ansible@ip-172-31-9-9 ~]$ ansible dev --list-hosts
  hosts (2):
    172.31.1.110
    172.31.4.186
[ansible@ip-172-31-9-9 ~]$ ansible dev -b -m yum -a "pkg=docker state=present"
The authenticity of host '172.31.4.186 (172.31.4.186)' can't be established.
ECDSA key fingerprint is SHA256:0c0ZRhF+x0a10N8Tes8TJya2zt3p0a0mhSiuyr0y0t0.
ECDSA key fingerprint is MD5:d8:0c:b2:8f:a8:93:13:94:35:01:11:9f:5b:58:79:90.
Are you sure you want to continue connecting (yes/no)? yes
[WARNING]: Platform linux on host 172.31.1.110 is using the discovered Python interpreter at /usr/bin/python, but future installation of anothe
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
172.31.1.110 | CHANGED => {
    "ansible_facts": {
```



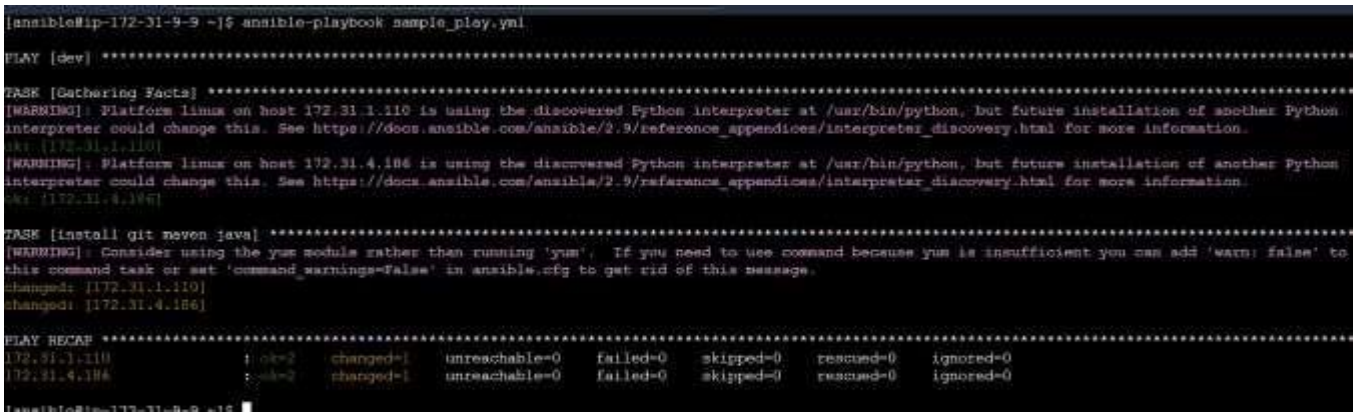
```
[ansible@ip-172-31-9-9 ~]$ ansible dev -b -m yum -a "pkg=docker state=present"
The authenticity of host '172.31.4.186 (172.31.4.186)' can't be established.
ECDSA key fingerprint is SHA256:0c0ZRhF+x0a10N8Tes8TJya2zt3p0a0mhSiuyr0y0t0.
ECDSA key fingerprint is MD5:d8:0c:b2:8f:a8:93:13:94:35:01:11:9f:5b:58:79:90.
Are you sure you want to continue connecting (yes/no)? yes
[WARNING]: Platform linux on host 172.31.1.110 is using the discovered Python interpreter at /usr/bin/python, but future installation of another Pyt
interpreter could change this. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
172.31.1.110 | CHANGED => {
    "ansible_facts": {
```

Step 26. Now create a playbook, use the command: vim sample_play.yml


https://ap-southeast-2.console.aws.amazon.com/ec2-instance-connect/ssh?region=ap-southeast-2&connType=standard&instanceId=i-06c0c4fa62440f33f&o...
aws
Services
Search
[Alt+S]
Sydney ▾
chaya sridur
EC2
S3
IAM

```
---
- name: Install Maven, Java, and tree on Linux
  hosts: all
  become: yes
  tasks:
    - name: Install Maven package
      yum:
        name: maven
        state: present

    - name: Install Java package
      yum:
        name: java
        state: present

    - name: Install tree package
      yum:
        name: tree
        state: present

~
~
~
~
~
~
~
"playbook.yml" 20L, 350B                                                    20,0-1        All
```

Step 27. Now run the playbook use: ansible-playbook sample_play.yml



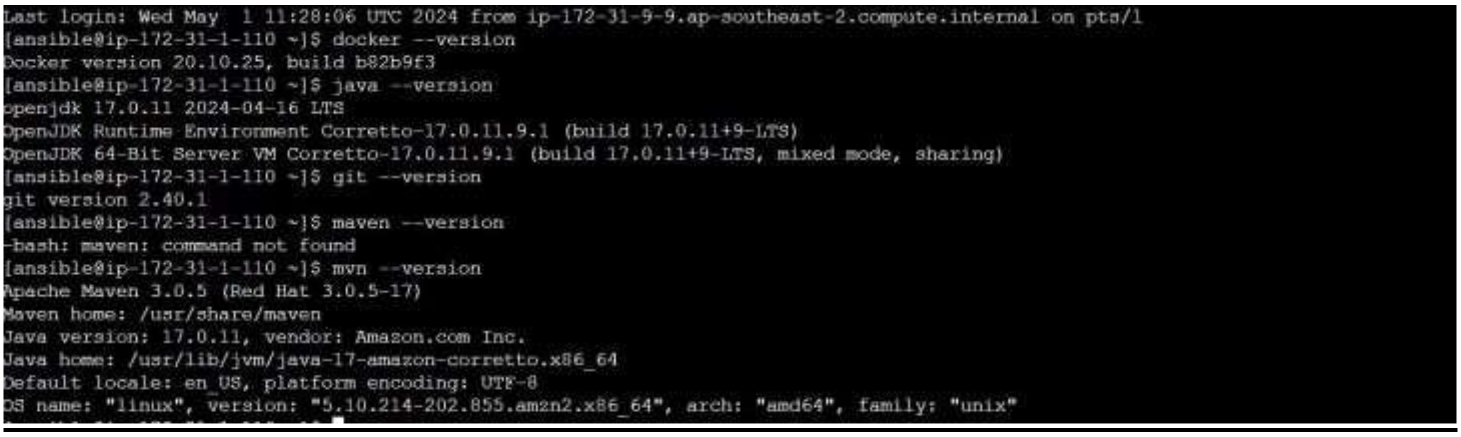
```
[ansible@ip-172-31-9-9 ~]$ ansible-playbook sample_play.yml

PLAY [dev] *********************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************
[WARNING]: Platform linux on host 172.31.1.110 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: [172.31.1.110]
[WARNING]: Platform linux on host 172.31.4.186 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: [172.31.4.186]

TASK [install git maven java] **************************************************************************************************
[WARNING]: Consider using the yum module rather than running 'yum'.  If you need to use command because yum is insufficient you can add 'warn: false' to
this command task or set 'command_warnings=False' in ansible.cfg to get rid of this message.
changed: [172.31.1.110]
changed: [172.31.4.186]

PLAY RECAP *********************************************************************************************************************
172.31.1.110               : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
172.31.4.186               : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ansible@ip-172-31-9-9 ~]$
```

Step 28. Now you can also cross check in other nodes whether whatever you've installed has been done or not by checking it's versions like

--> git --version

--> mvn --version

--> java –version



```
Last login: Wed May  1 11:28:06 UTC 2024 from ip-172-31-9-9.ap-southeast-2.compute.internal on pts/1
[ansible@ip-172-31-1-110 ~]$ docker --version
Docker version 20.10.25, build b82b9f3
[ansible@ip-172-31-1-110 ~]$ java --version
openjdk 17.0.11 2024-04-16 LTS
OpenJDK Runtime Environment Corretto-17.0.11.9.1 (build 17.0.11+9-LTS)
OpenJDK 64-Bit Server VM Corretto-17.0.11.9.1 (build 17.0.11+9-LTS, mixed mode, sharing)
[ansible@ip-172-31-1-110 ~]$ git --version
git version 2.40.1
[ansible@ip-172-31-1-110 ~]$ maven --version
-bash: maven: command not found
[ansible@ip-172-31-1-110 ~]$ mvn --version
Apache Maven 3.0.5 (Red Hat 3.0.5-17)
Maven home: /usr/share/maven
Java version: 17.0.11, vendor: Amazon.com Inc.
Java home: /usr/lib/jvm/java-17-amazon-corretto.x86_64
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "5.10.214-202.855.amzn2.x86_64", arch: "amd64", family: "unix"
```
]

Alternative method to config and install ansible

Ansible installation steps:

Create 3 servers *example ansible-dev, node1, node2+

Dev instance

Then first goto dev instance

Give command sudo -i so you can goto root user

Then give command yum update -y *it will update the packages+

Then create a user and give password

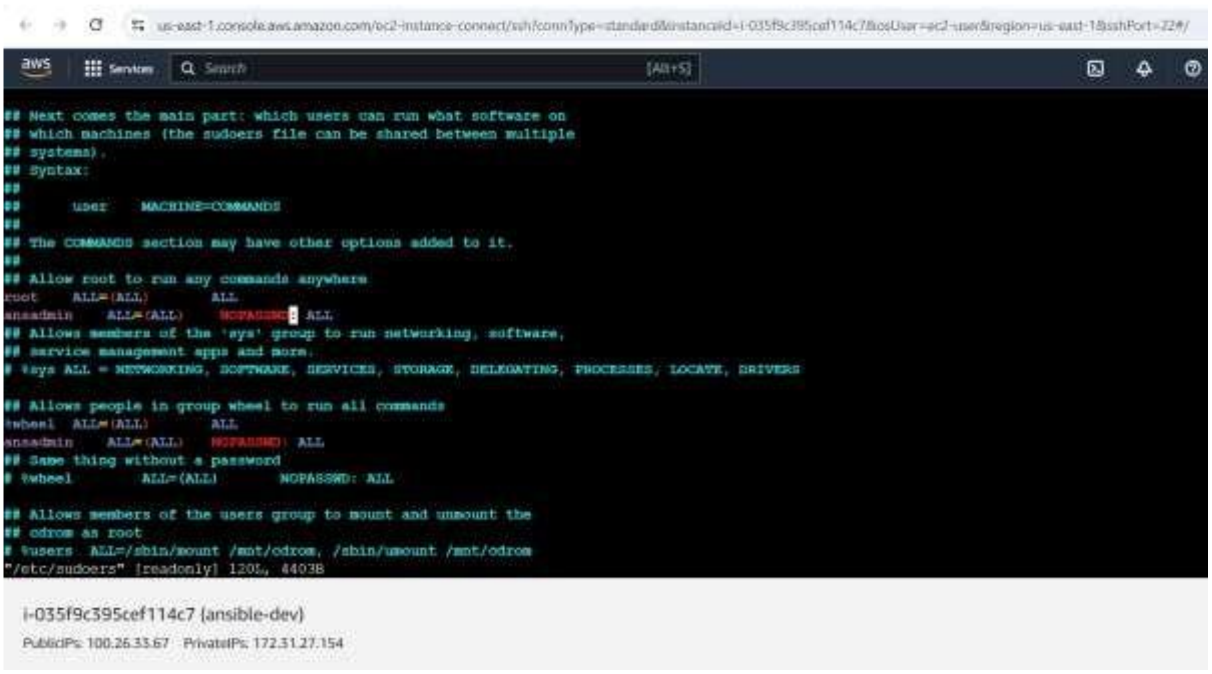
Useradd ansadmin

Passwd ansadmin

Vi /etc/sudoers

As shown in below given picture type below command and save it ansadmin

ALL=(ALL) NOPASSWD: ALL



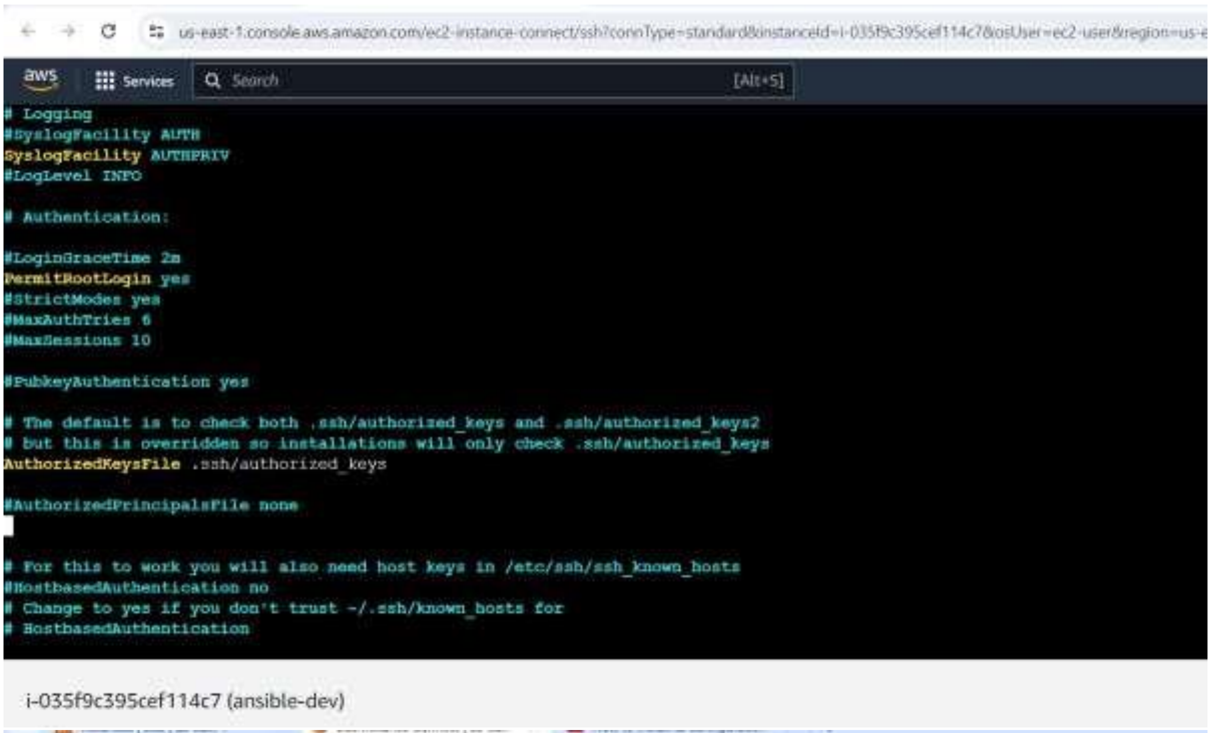
```
## Next comes the main part: which users can run what software on
## which machines (the sudoers file can be shared between multiple
## systems).
## Syntax:
##
##      user    MACHINE=COMMANDS
##
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere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

## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)       ALL
ansadmin    ALL=(ALL)   NOPASSWD: ALL
## Same thing without a password
# %wheel        ALL=(ALL)       NOPASSWD: ALL

## Allows members of the users group to mount and unmount the
## cdrom as root
# %users  ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/cdrom
"/etc/sudoers" [readonly] 120L, 4403B
```

i-035f9c395cef114c7 (ansible-dev)

PublicIPs: 100.26.33.67 PrivateIPs: 172.31.27.154

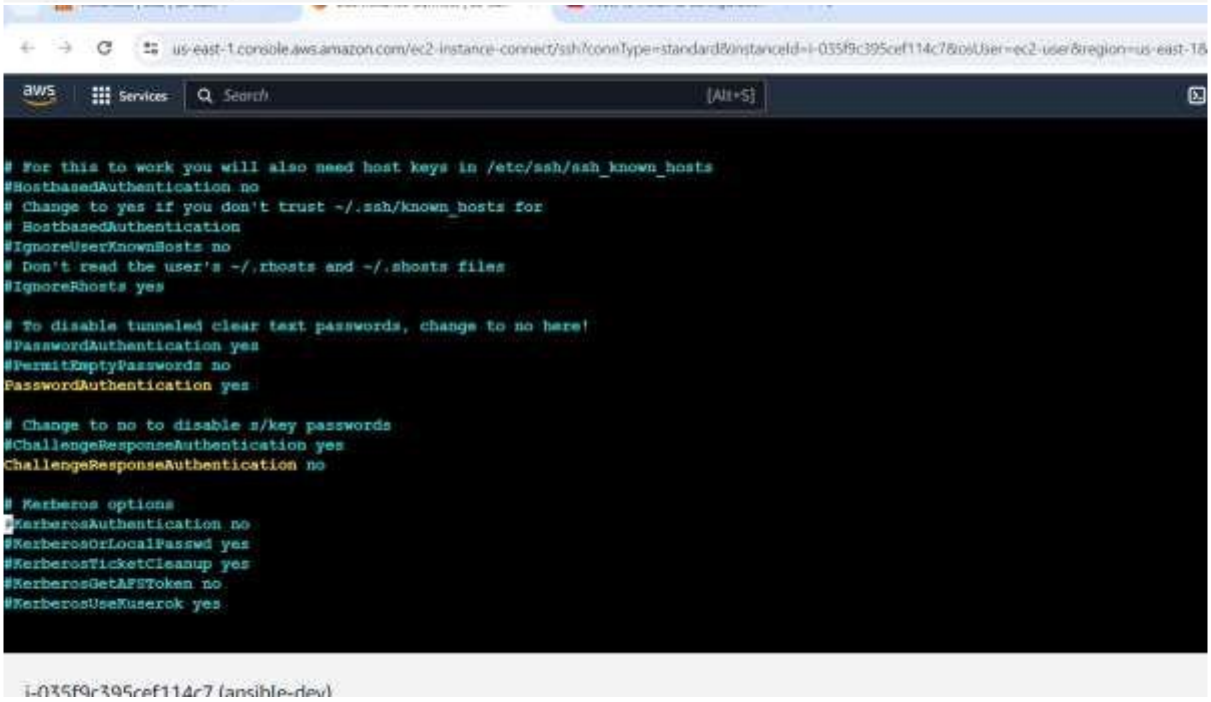
vi /etc/ssh/sshd_config

made the changes as shown in below and save it (uncomment the permitRootLogin and passwordAuthentication yes)



```
# Logging
#SyslogFacility AUTH
SyslogFacility AUTHPRIV
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#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
```

i-035f9c395cef114c7 (ansible-dev)



```
# 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
```

i-035f9c395cef114c7 (ansible-dev)

Systemctl restart sshd

Node1

Note: Perform same steps as mentioned in ansible-dev

sudo -i yum update -y

Useradd ansadmin

Passwd ansadmin

Vi /etc/sudoers

As shown in above given picture type below command and save it

ansadmin ALL=(ALL) NOPASSWD: ALL vi /etc/ssh/sshd_config

uncomment the permitRootLogin and passwordAuthentication yes and save it Systemctl restart sshd

Node2

Perform node1 steps for node2

Dev-instance

Goto dev instance now switch to ansadmin

Su – ansadmin

Ssh-keygen

Cd .ssh then ls -ltr

Ssh-copy-id ansadmin@node1 privateIP

It will ask password give it

Ssh-copy-id ansadmin@node2 privateIP

It will ask password give it

Ssh ansadmin@node2privateIP

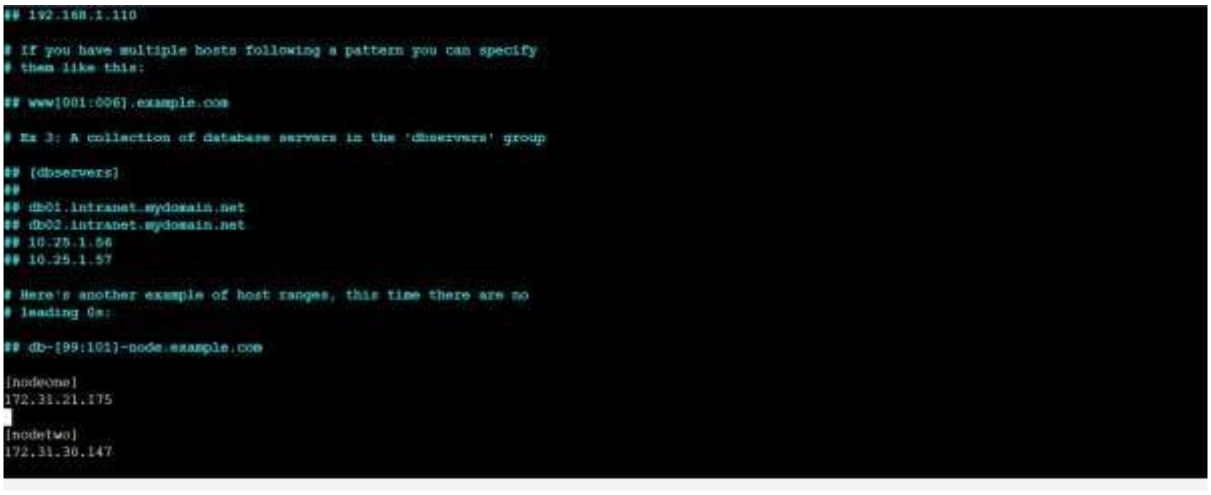
Exit

Sudo amazon-linux-extras install ansible2 -y yum install git python python-pip python-level openssl -y ansible --version cd /etc/ansible

ls -l

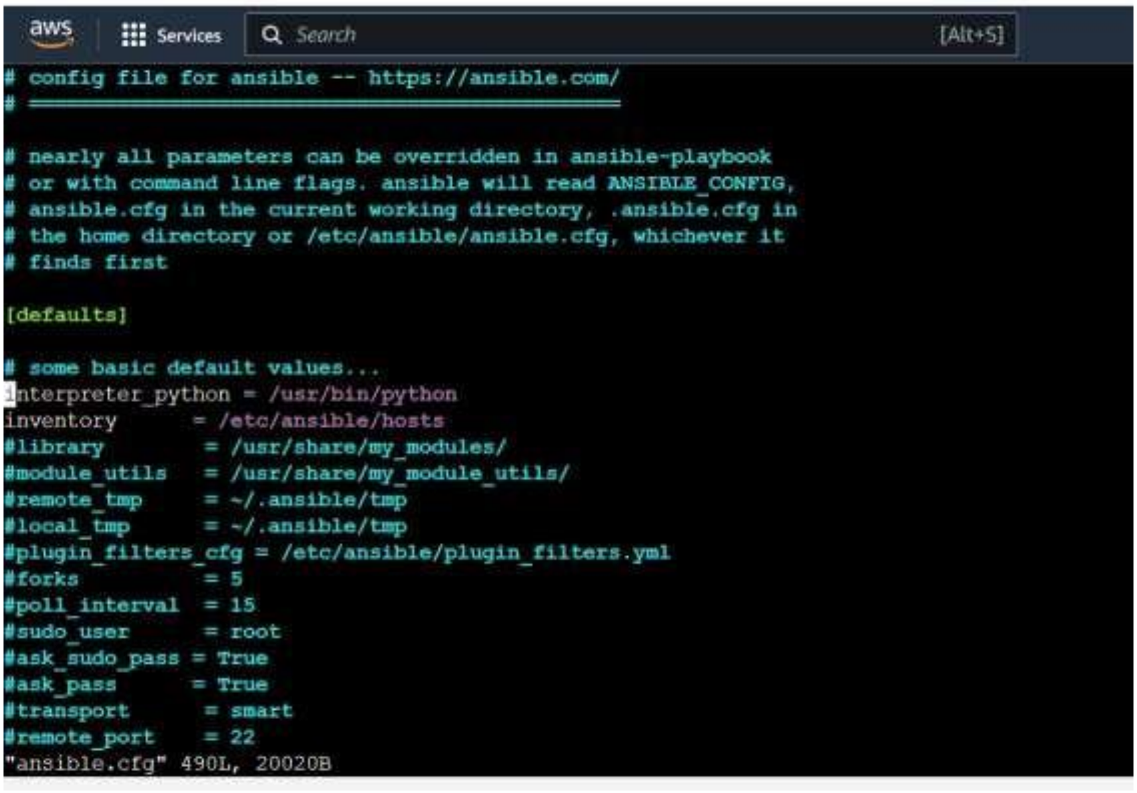
sudo chown -R ansadmin:ansadmin * vi hosts

(at the end of the file give the IP addresses of both nodes)



```
## 192.168.1.110

# If you have multiple hosts following a pattern you can specify
# them like this:

## www[001:006].example.com

# Ex 3: A collection of database servers in the 'dbservers' group

## [dbservers]
##
## db01.intranet.mydomain.net
## db02.intranet.mydomain.net
## 10.25.1.56
## 10.25.1.57

# Here's another example of host ranges, this time there are no
# leading 0s:

## db-[99:101]-node.example.com

[nodeone]
172.31.21.175

[nodetwo]
172.31.30.147
```

Vi ansible.cfg



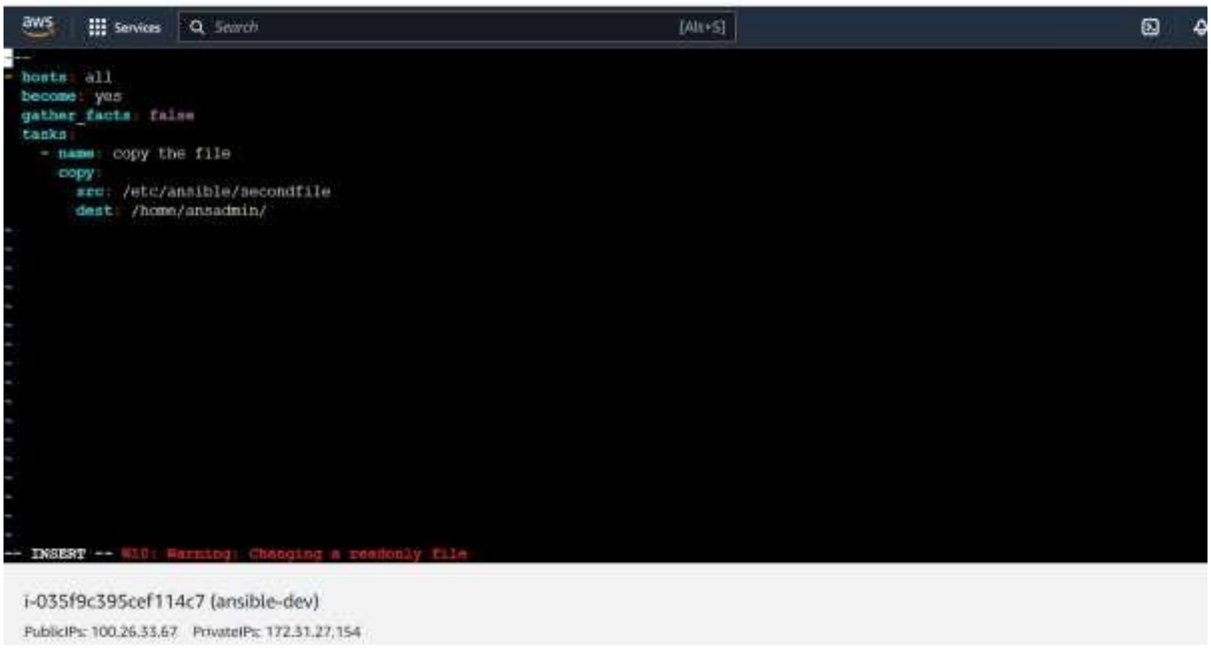
```
# config file for ansible -- https://ansible.com/
# ===============================================

# 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...
interpreter_python = /usr/bin/python
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
"ansible.cfg" 490L, 20020B
```

Ansible all -m ping Sudo

touch fileone

ansible all -i hosts -m copy -a "src=file dest=/home/ansadmin/"

sudo touch secondfile sudo vi first.yml

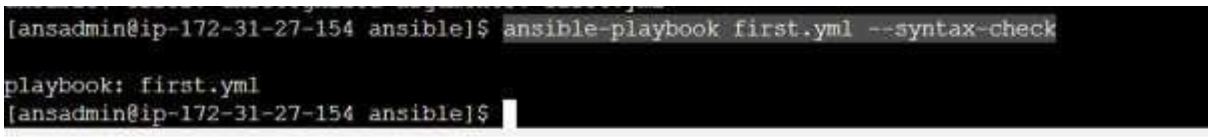


```
---
- hosts: all
  become: yes
  gather_facts: false
  tasks:
    - name: copy the file
      copy:
        src: /etc/ansible/secondfile
        dest: /home/ansadmin/
~
-- INSERT -- W10: Warning: Changing a readonly file
```

i-035f9c395cef114c7 (ansible-dev)

PublicIPs: 100.26.33.67   PrivateIPs: 172.31.27.154

ansible-playbook first.yml --syntax-check



```
[ansadmin@ip-172-31-27-154 ansible]$ ansible-playbook first.yml --syntax-check

playbook: first.yml
[ansadmin@ip-172-31-27-154 ansible]$ 
```

ansible-playbook first.yml –check ansible-playbook first.yml

vi secondfile (type anything and save it) ansible-playbook

first.yml -vvv

Note1

Sudo su – ansadmin

ls -l

cat secondfile

Ansible Interview Questions

1. What Is Ansible?

Ansible is an open-source automation tool developed by Redhat that is meant for simplifying the configuration management, application deployment, and automation of tasks. It allows us in automating repetitive tasks, ensuring consistency and efficiency in managing servers and networks using SSH protocol for make communication with the network Agentlessly.

2. Describe Infrastructure as Code (IaC) And How Ansible Aligns With This Concept.

Infrastructure as Code (IaC) is the way of process used for managing and provisioning the infrastructure using code instead of manual workflow process. Ansible follows this approach allowing the users to describe and manage infrastructure in a code-like manner.

3.Explain Ansible Galaxy, modules, And Plugins in The Context of Ansible Automation.

Ansible Galaxy is a platform providing the features of sharing and downloading Ansible roles. Modules are task executional units, and plugins helps in enhancing Ansible's functionality by extending its capabilities.

4. What Are Ansible Tasks, And How Do They Contribute to The Automation Process?

Ansible tasks are individual work units within a playbook coming with defined actions that to be performed on remote hosts, contributing to the overall automation process.

5. What Makes Ansible Stand Out from Other Configuration Management Tools?

Ansible's agentless architecture and its simplicity in use (YAML syntax), and ease of setup contribute to its standout features among other Configuration Management tools.

6.Briefly Explain Ansible's Architecture Through Outlining Its Different Components.

The control node communicates with managed nodes through SSH protocol executing tasks defined in playbooks. Ansible consists of a control node, managed nodes, inventory, modules, and plugins.

7. What Is The Foundational Programming Language Of Ansible?

Ansible is built on top of python, enhancing its simplicity for clear playbooks and versatility in creating robust modules. Python's wide range of adoption provided good community support, making Ansible an effective and flexible automation tool.

8. What are handlers in Ansible, and how are they used in playbooks?

Handlers in Ansible are tasks that are triggered by other tasks, usually used to respond for the changes that require start, stop, reload or restart the service actions. They are defined in the playbook and executed as per need.

9. Can You Automate Password Input in An Ansible Playbook Using Encrypted Files?

Yes, Ansible helps in allowing you to automate password input in playbooks using encrypted files, specifically using Ansible Vault. Ansible Vault provides a secured way for encrypting sensitive data such as passwords, so that they can be stored safely for usage in your playbooks.

10. What Is the Role of Callback Plugins in Ansible, And How Can They Be Used?

Callback plugins in Ansible play a vital role in customizing output and behavior of ansible. They are mostly used for logging, notifications, or any post-playbook actions, allowing for impressive Ansible functionality.