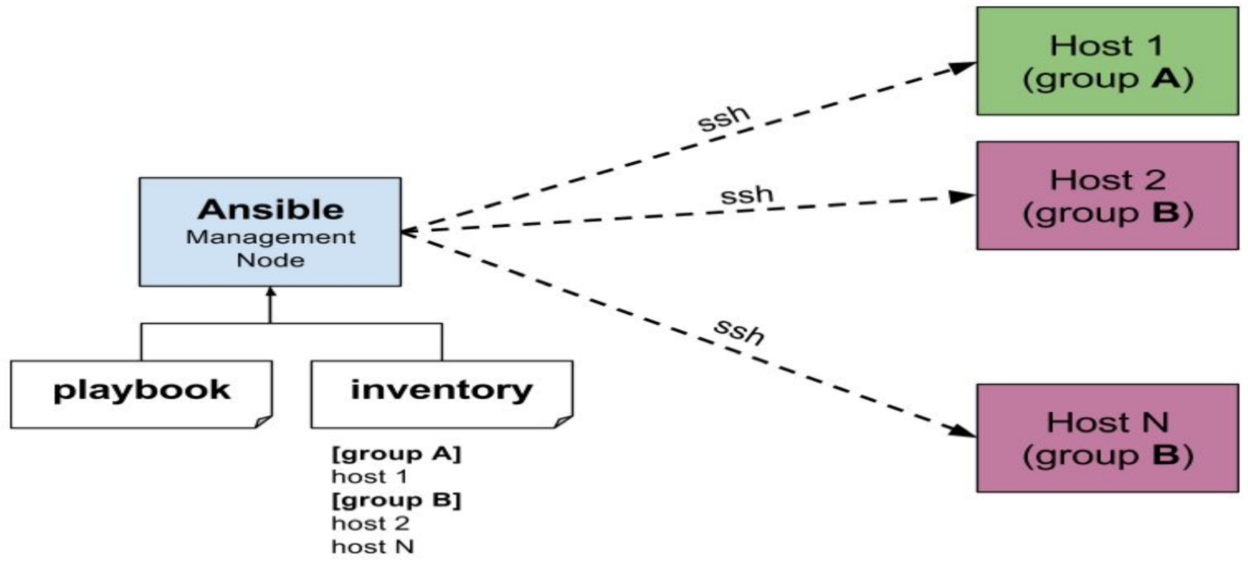
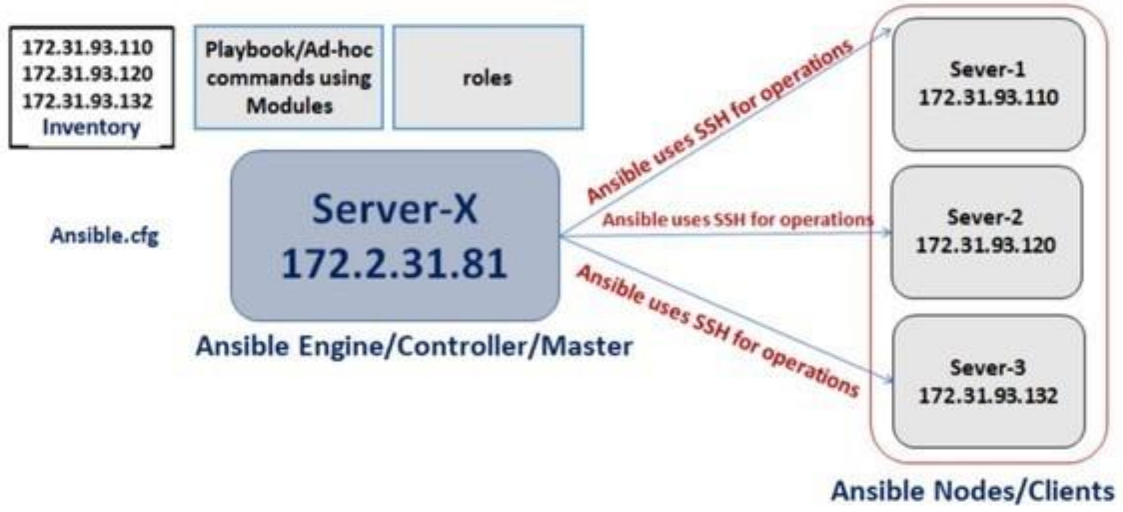
Setting Up Ansible Friendly Env




172.31.93.110
172.31.93.120
172.31.93.132
Inventory
Playbook/Ad-hoc commands using Modules
roles
Ansible.cfg
Server-X
172.2.31.81
Ansible Engine/Controller/Master
Ansible uses SSH for operations
Ansible uses SSH for operations
Ansible uses SSH for operations
Sever-1
172.31.93.110
Sever-2
172.31.93.120
Sever-3
172.31.93.132
Ansible Nodes/Clients

Spin up 3 servers with amazon linux 2023, use a t3.medium instance

pass this userdata

```
#!/bin/bash
sudo useradd ansible
sudo echo ansible:ansible | chpasswd
sudo echo "ansible ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers
sudo sed -i "s/PasswordAuthentication no/PasswordAuthentication yes/g" /etc/ssh/sshd_config
sudo sed -i "s/.*#PermitRootLogin yes/PermitRootLogin yes/g" /etc/ssh/sshd_config
sudo service sshd restart
```

ansible-master, ansible-node1, ansible-node2

Create User ansible in all 3 servers

#sudo useradd ansible

#sudo passwd ansible

===

You can use this steps if you mised pass the userdata above.

step 1 : edit the sshd_config file of all your 3 servers to allow the password based login on the

```
sudo sed -i 's/.*PasswordAuthentication no/Passwordauthentication yes/g' /etc/ssh/sshd_config
sudo sed -i 's/.*#i/PermitRootLogin yes/g' /etc/ssh/sshd_config
sudo echo "ansible  ALL=(ALL)      NOPASSWD: ALL" >> /etc/sudoers
```

Step 2 : Not required as the above commands are the short cuts

#sudo vi /etc/ssh/sshd_config

:/PasswordAuthentication :

:/PermitRootLogin

PermitRootLogin no to PermitRootLogin yes

PasswordAuthentication no to PasswordAuthentication yes

—-- Not required as the above commands are the short cuts

#sudo service sshd restart

===

To begin exploring Ansible as a means of managing our various servers, we need to install the Ansible software on ansible master. What ever we are doing below is only in the ansible-master

To get Ansible for amazon linux,

```
# sudo dnf update -y
# sudo dnf install ansible-core -y
```

Check Ansible version

```
# ansible –version
```

With Amazon linux, ansible does not come with a default inventory file. so let's create one

```
# mkdir -p ~/ansible/inventory
# cd ~/ansible
# touch inventory/hosts
# vi inventory/hosts
```

Create a configuration file

# Create default configuration file

sudo touch /etc/ansible/ansible.cfg

upodate the configuration file

sudo vi ansible.cfg

paste below in the configuration file

```
[defaults]
inventory = /etc/ansible/inventory/hosts
host_key_checking = False
```

# Ensure ansible user can access these directories

sudo chown -R ansible:ansible /etc/ansible

```
[defaults]
inventory = /etc/ansible/inventory/hosts
host_key_checking = False
```

Ansible keeps track of all of the servers that it knows about through a "hosts" file. We need to set up this file first before we can begin to communicate with our other computers.

Open the file with root privileges like this:

```
#sudo vi /etc/ansible/inventory/hosts
[webserver]
Ip address of ansible-node1
Ip address of ansible-node2
```

In your ansible-master, login as user ansible and generate ssh key on ansible control server

```
#ssh ansible@ip address of the master server
```

Create an ssh key in master server and copy it to node servers
This will create .ssh folder (/home/ansadm/.ssh). Hit enter all the way through

#ssh-keygen -t rsa

This will create .ssh folder (/home/ansible/.ssh). Hit enter all the way through

#chmod 700 /home/ansible/.ssh

#ssh-copy-id ansible@ip address of you ansible-node1

#ssh-copy-id ansible@ip address of you ansible-node2

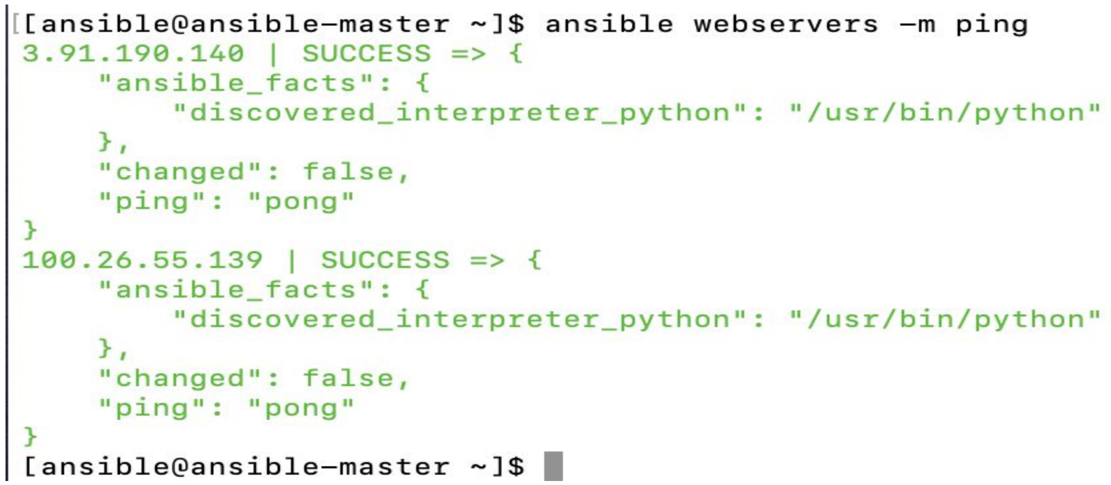
#**ssh ansible@**ip address of your ansible-node1

#**ssh ansible@**ip address of your ansible-node2

Now all three servers are configured, ansible control server can do ssh on both the servers

check connectivity of hosts is

#ansible <group> -m ping



```
[ansible@ansible-master ~]$ ansible webservers -m ping
3.91.190.140 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
100.26.55.139 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
[ansible@ansible-master ~]$
```

- **Check if package is installed & update it**
- $ ansible <group> -m yum -a “name=ntp state=latest”
- **Check if package is installed & don’t update it**
- $ ansible <group> -m yum -a “name=ntp state=present”
- **Check if package is at a specific version**
- $ ansible -i <group> -b -m yum -a “name= ntp-1.8 state=present”
- $ ansible -i <group> -b -m yum -a “name= ntp-1.8 state=latest”
- **Now check the RAM Memory usage on all servers using the 'free -m' command.**
- ansible hakase-testing -m shell -a 'free -m' --become
- **Check the disk available on the root partition using the fdisk command.**
- ansible hakase-testing -m shell -a 'df -h /dev/sda2' --become
- I**nstall a single package using the ad-hoc command with the apt module as below.**
- ansible hakase-testing -m yum -a 'name=nginx state=latest' --become
- **Now check the uptime of each server.**
- ansible hakase-testing -m shell -a 'uptime' --become
- **Check if package is not installed**
- $ ansible <group> -b -m yum -a “name=ntp state=absent”
- **Starting a service**
- $ ansible <group> -b -m service -a “name=httpd state=started”
- **Stopping a service**
- $ ansible <group> -b -m service -a “name=httpd state=stopped”
- **Restarting a service**
- $ ansible <group> -b -m service -a “name=httpd state=restarted”

- **Working with file module**
- $ ansible -i <group> -b -m file -a “name=/root/test state=touch”
- $ ansible -i <group> -b -m file -a “name=/root/test state=absent”
- **Creating system user with user and modules**
- $ ansible -i <group> -b -m user-a “name=user1 state=present”
- $ ansible -i <group> -b -m user-a “name=jjtech state=absent”
- $ ansible -i <group> -b -m group -a “name=group1”
- $ ansible -i <group> -b -m user -a “name=user1 group=group1”
- Ansible Glossary
- The following Ansible-specific terms are largely used throughout this guide:
- Control Machine / Node: a system where Ansible is installed and configured to connect and execute commands on nodes.
- Node: a server controlled by Ansible.
- Inventory File: a file that contains information about the servers Ansible controls, typically located at /etc/ansible/hosts.
- Playbook: a file containing a series of tasks to be executed on a remote server.
- Role: a collection of playbooks and other files that are relevant to a goal such as installing a web server.
- Play: a full Ansible run. A *play* can have several playbooks and roles, included from a single playbook that acts as entry point.

- **Running Playbooks**
- To run a playbook and execute all the tasks defined within isshschat, use the ansible-playbookcommand:
-
- #ansible-playbook myplaybook.yml
- To overwrite the default hosts option in the playbook and limit execution to a certain group or host, include the option -l in your command:
-
- #ansible-playbook -l server1 myplaybook.yml
-
- **Getting Information about a Play**
- The option --list-tasks is used to list all tasks that would be executed by a play without making any changes to the remote servers:
-
- #ansible-playbook myplaybook.yml --list-tasks
- Similarly, it is possible to list all hosts that would be affected by a play, without running any tasks on the remote servers:
-
- #ansible-playbook myplaybook.yml --list-hosts
- You can use *tags* to limit the execution of a play. To list all tags available in a play, use the option --list-tags:
-
- #ansible-playbook myplaybook.yml --list-tags
-
- #ansible-playbook packages.yml
-
- #ansible --list-hosts all