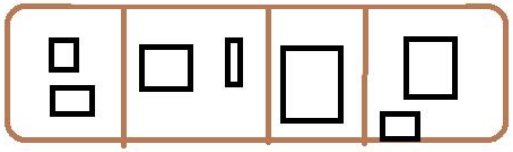
CM TOOLS

ASNIBLE, CHEF, PUPPET etc,

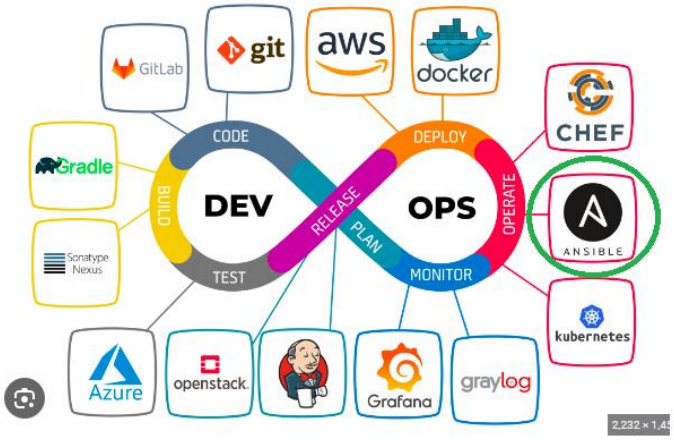
Buliding -- House Computers



APPLIACTION

Configuration of My Application, My Servers

APPliaction
Cloud Data Center
Devops Stages



1. Manual 2. Automate

Cofigration Management

OPERATE

Packages,
Service,
Groups

add
delete
modify etc,

DEVOPS GUY try to manage & monitor Application & It's Behaviour

1. Manual SYSTEM ADMIN : Exam : 200 500 1k 2k ---> Windows Admin, Linux Admin, N/w Admin , DB Admin

2. Automate : will do devops eng : 2k @ Java -- Single min @ yaml Script --> Main Server --> all the nodes

ANSIBLE

It is CM Tool, by using Ansible method we are going to automate the tasks.

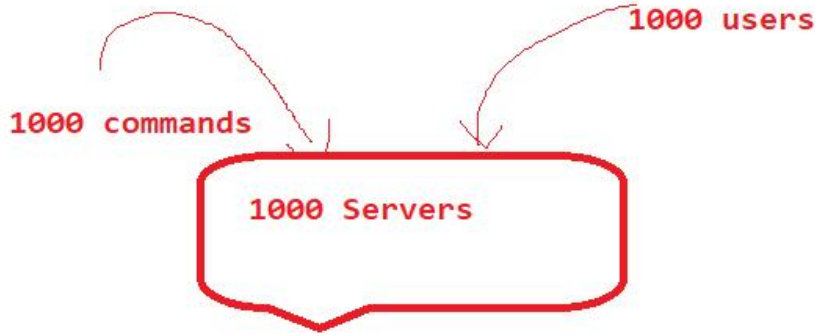
Configuration Managment : IAC (Infra Str as Code)

use:

Testable
Repeatable
Versionable

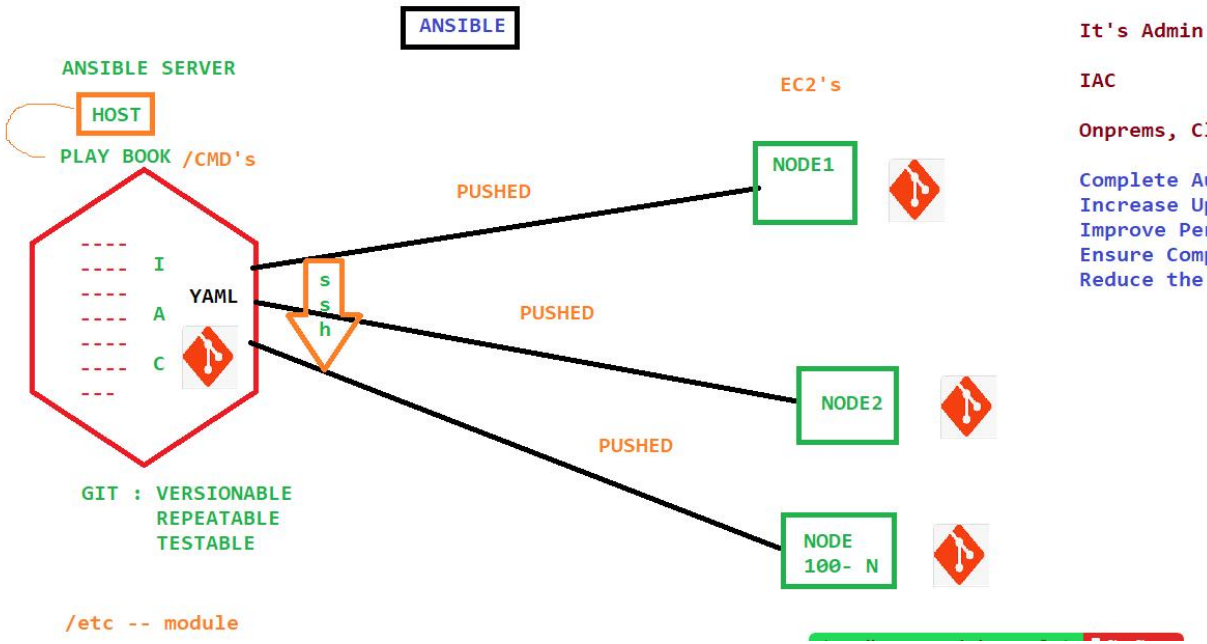
function:

S/w (Install, Update)
n/W (Change IP)
Users (add, delete)
Process (start, stop)



Manually @ Difficult

same sceneraio we can use
Ansible by devops Eng 2-3 min

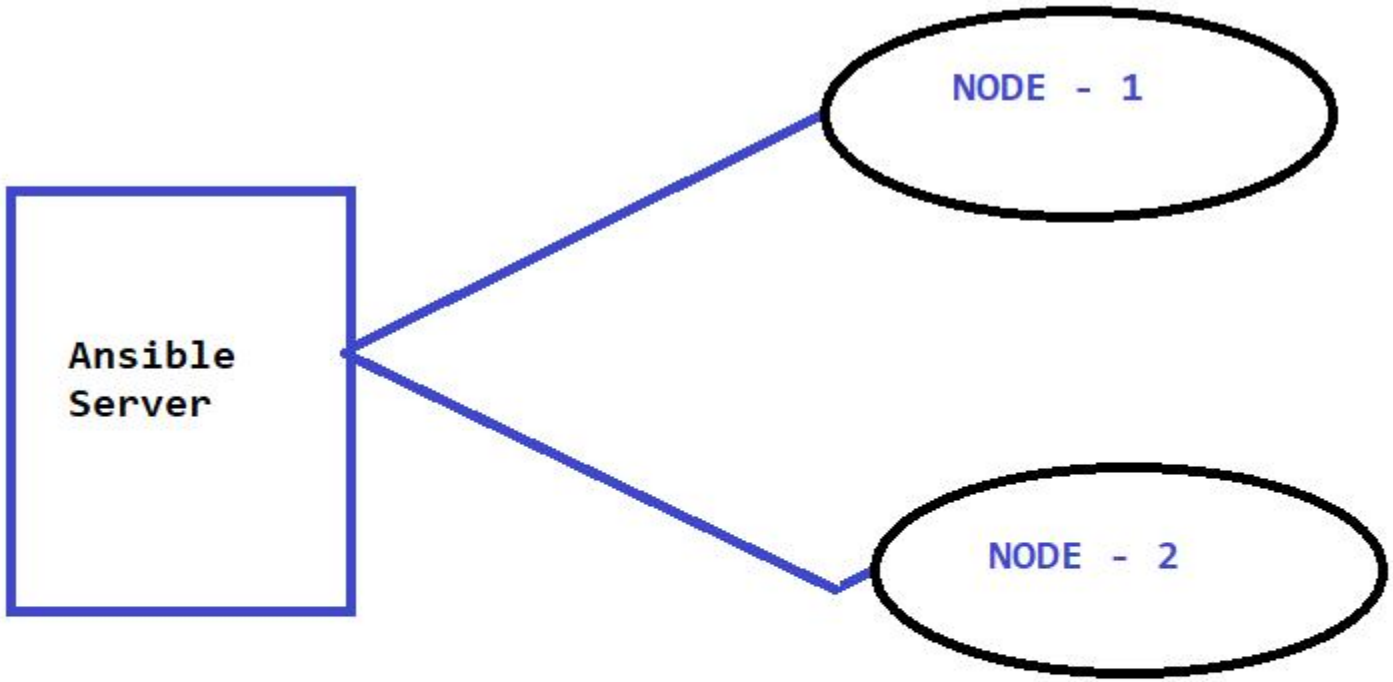


It's Admin Tool

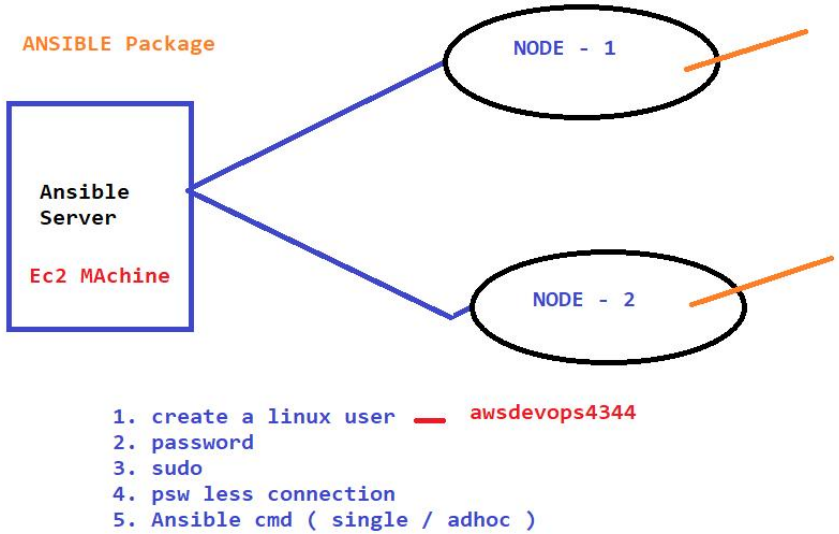
IAC

Onprems, Cloud , Vm

Complete Automation
Increase Uptime
Improve Performance
Ensure Complience
Reduce the Cost



1. create a linux user
2. password
3. sudo
4. psw less connection
5. Ansible cmd (single / adhoc)



Install epel Package

1. wget http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

```
[root@ip-172-31-38-145 ec2-user]# wget http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
--2023-06-26 04:47:39--  http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
Resolving dl.fedoraproject.org (dl.fedoraproject.org)... 38.145.60.24, 38.145.60.22, 38.145.60.23
Connecting to dl.fedoraproject.org (dl.fedoraproject.org)|38.145.60.24|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 15608 (15K) [application/x-rpm]
Saving to: 'epel-release-latest-7.noarch.rpm'

100%[===================================================================================>] 15,608      78.1KB/s   in 0.2s
```

```
[root@ip-172-31-38-145 ec2-user]# ls
epel-release-latest-7.noarch.rpm
[root@ip-172-31-38-145 ec2-user]#
```

2. yum install epel-release-latest-7.noarch.rpm -y

```
[root@ip-172-31-38-145 ec2-user]# yum install epel-release-latest-7.noarch.rpm -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Examining epel-release-latest-7.noarch.rpm: epel-release-7-14.noarch
Marking epel-release-latest-7.noarch.rpm to be installed
Resolving Dependencies
--> Running transaction check
---> Package epel-release.noarch 0:7-14 will be installed
```

3. sudo yum update –y
4. sudo yum install git python python-devel python-pip openssl ansible –y
    a. ansible required python related packages

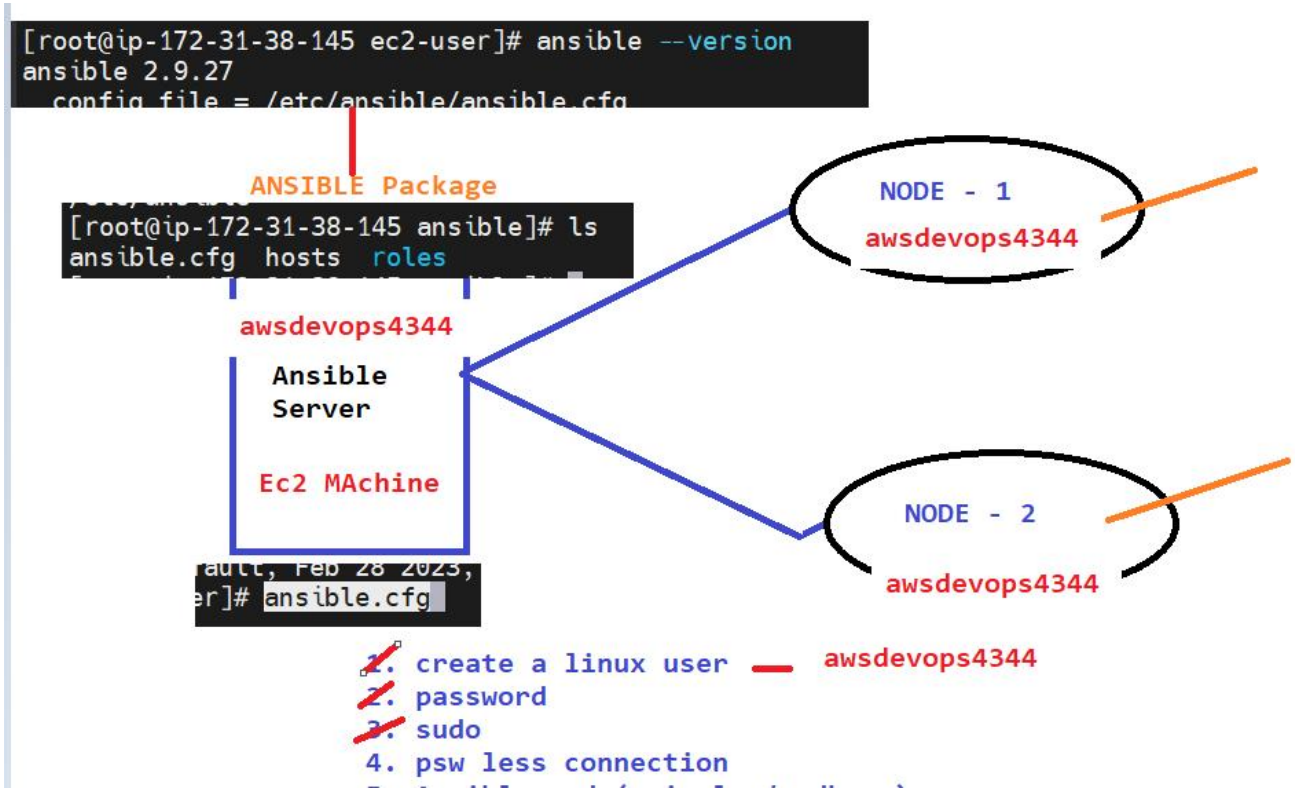
```
[root@ip-172-31-38-145 ec2-user]# sudo yum install git python python-devel python-pip openssl ansible -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
223 packages excluded due to repository priority protections
Package python-2.7.18-1.amzn2.0.6.x86_64 already installed and latest version
Package python-devel-2.7.18-1.amzn2.0.6.x86_64 already installed and latest version
```

5. ansible –version

```
2. ANSIBLE SERVER
[root@ip-172-31-38-145 ec2-user]# ansible --version
ansible 2.9.27
  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, Feb 28 2023, 02:51:06) [GCC 7.3.1 20180712 (Red Hat 7.3.1-15)]
[root@ip-172-31-38-145 ec2-user]#
```

Most Impoartant file is :

```
[root@ip-172-31-38-145 ec2-user]# cd /etc/ansible
[root@ip-172-31-38-145 ansible]#
[root@ip-172-31-38-145 ansible]#
[root@ip-172-31-38-145 ansible]#
[root@ip-172-31-38-145 ansible]# pwd
/etc/ansible
[root@ip-172-31-38-145 ansible]# ls
ansible.cfg  hosts  roles
[root@ip-172-31-38-145 ansible]#
```

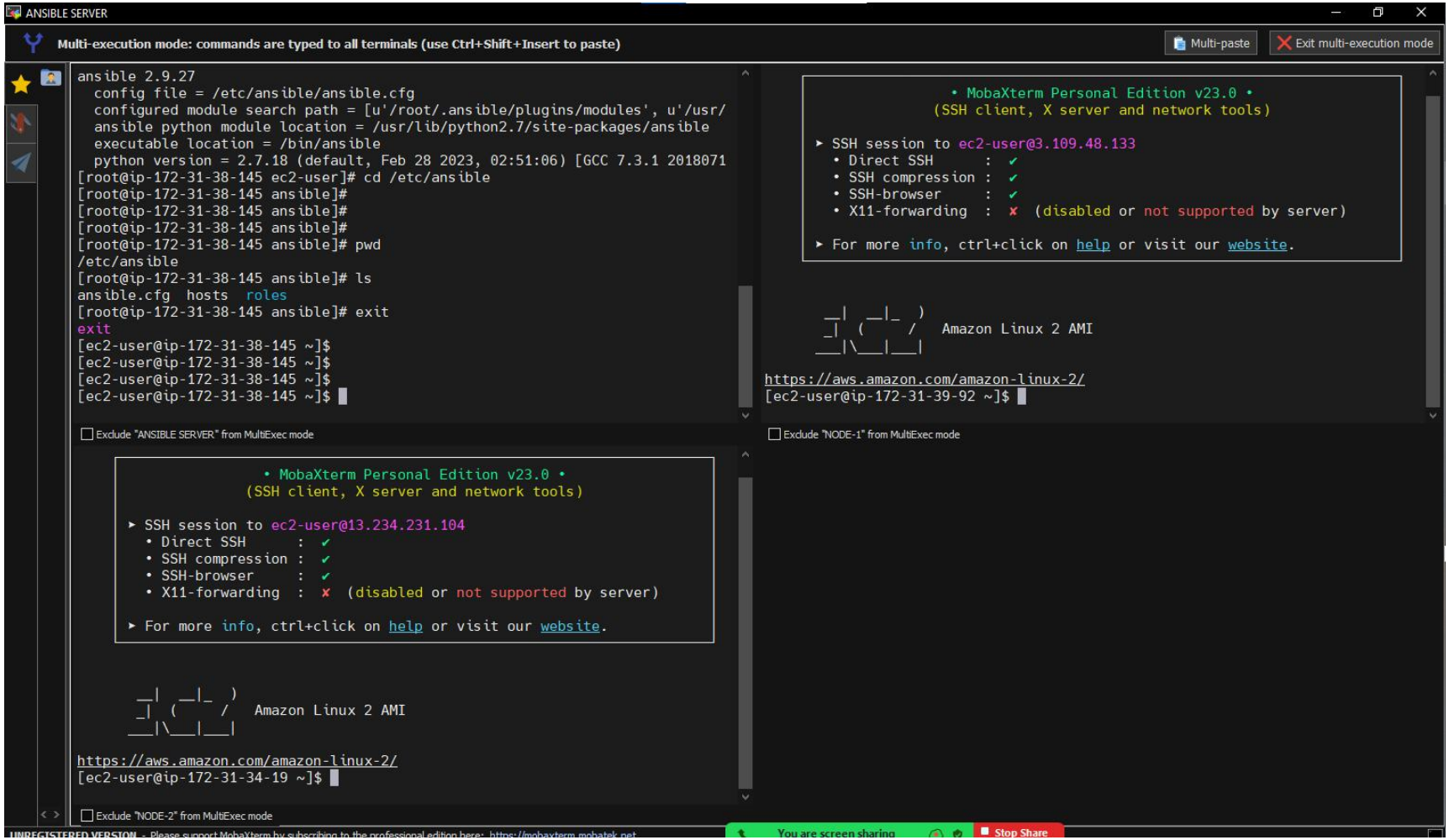


Environment of Ansible Project :

Ansible Server	i-03541504aa2c4f922	Running		t2.micro	2/2 checks passed.	No alarr
Node1	i-0c3bd8d58c3872c2b	Running		t2.micro	2/2 checks passed.	No alarr
Node2	i-0652f118048419a55	Running		t2.micro	2/2 checks passed.	No alarr

Connect Mobaxtream

One – all machines



Added user all the machines

1. Create user
2. Added Password

```
[ec2-user@ip-172-31-38-145 ~]$ sudo -s
[root@ip-172-31-38-145 ec2-user]#
[root@ip-172-31-38-145 ec2-user]# useradd awsdevops4344
[root@ip-172-31-38-145 ec2-user]# passwd welcome@1
passwd: Unknown user name 'welcome@1'.
[root@ip-172-31-38-145 ec2-user]# useradd awsdevops4344
useradd: user 'awsdevops4344' already exists
[root@ip-172-31-38-145 ec2-user]# passwd awsdevops4344
Changing password for user awsdevops4344.
New password:
BAD PASSWORD: The password fails the dictionary check - it is based on a dict
ionary word
Retype new password:
passwd: all authentication tokens updated successfully.
[root@ip-172-31-38-145 ec2-user]#
```

```
[ec2-user@ip-172-31-39-92 ~]$ sudo -s
[root@ip-172-31-39-92 ec2-user]#
[root@ip-172-31-39-92 ec2-user]# useradd awsdevops4344
[root@ip-172-31-39-92 ec2-user]# passwd welcome@1
passwd: Unknown user name 'welcome@1'.
[root@ip-172-31-39-92 ec2-user]# useradd awsdevops4344
useradd: user 'awsdevops4344' already exists
[root@ip-172-31-39-92 ec2-user]# passwd awsdevops4344
Changing password for user awsdevops4344.
New password:
BAD PASSWORD: The password fails the dictionary check - it is based on a dict
ionary word
Retype new password:
passwd: all authentication tokens updated successfully.
[root@ip-172-31-39-92 ec2-user]#
```

Exclude "ANSIBLE SERVER" from MultiExec mode

```
[ec2-user@ip-172-31-34-19 ~]$ sudo -s
[root@ip-172-31-34-19 ec2-user]#
[root@ip-172-31-34-19 ec2-user]# useradd awsdevops4344
[root@ip-172-31-34-19 ec2-user]# passwd welcome@1
passwd: Unknown user name 'welcome@1'.
[root@ip-172-31-34-19 ec2-user]# useradd awsdevops4344
useradd: user 'awsdevops4344' already exists
[root@ip-172-31-34-19 ec2-user]# passwd awsdevops4344
Changing password for user awsdevops4344.
New password:
BAD PASSWORD: The password fails the dictionary check - it is based on a dict
ionary word
Retype new password:
passwd: all authentication tokens updated successfully.
[root@ip-172-31-34-19 ec2-user]#
```

Exclude "NODE-1" from MultiExec mode

3. Provide sudo access

```
[awsdevops4344@ip-172-31-38-145 ec2-user]$ sudo yum install tree -y

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for awsdevops4344:
```

```
[awsdevops4344@ip-172-31-39-92 ec2-user]$ sudo yum install tree -y

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for awsdevops4344:
```

Exclude "ANSIBLE SERVER" from MultiExec mode

```
[awsdevops4344@ip-172-31-34-19 ec2-user]$ sudo yum install tree -y

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for awsdevops4344:
```

Exclude "NODE-1" from MultiExec mode

Added sudo permission (visudo → not allow same user permis)

```
## Allow root to run any commands anywhere
root    ALL=(ALL)       ALL
awsdevops4344 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 --                                          101,37        85%
```

Exclude "ANSIBLE SERVER" from MultiExec mode

```
## Allow root to run any commands anywhere
root    ALL=(ALL)       ALL
awsdevops4344 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, PROCESS
LOCATE, DRIVERS

-- INSERT --                                          101,37
```

Exclude "NODE-1" from MultiExec mode

```
## 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
awsdevops4344 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,
```

Sudo previliages added..

```
[root@ip-172-31-38-145 ec2-user]# su -awsdevops4344
su: invalid option -- 'a'
Try 'su --help' for more information.
[root@ip-172-31-38-145 ec2-user]# su - awsdevops4344
Last login: Mon Jun 26 05:10:55 UTC 2023 on pts/0
[awsdevops4344@ip-172-31-38-145 ~]$
[awsdevops4344@ip-172-31-38-145 ~]$
[awsdevops4344@ip-172-31-38-145 ~]$
[awsdevops4344@ip-172-31-38-145 ~]$
[awsdevops4344@ip-172-31-38-145 ~]$ sudo yum install tree -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                               | 3.7 kB     00:00
223 packages excluded due to repository priority protections
```

```
Try 'su --help' for more information.
[root@ip-172-31-39-92 ec2-user]# su - awsdevops4344
Last login: Mon Jun 26 05:10:55 UTC 2023 on pts/0
[awsdevops4344@ip-172-31-39-92 ~]$
[awsdevops4344@ip-172-31-39-92 ~]$
[awsdevops4344@ip-172-31-39-92 ~]$
[awsdevops4344@ip-172-31-39-92 ~]$
[awsdevops4344@ip-172-31-39-92 ~]$ sudo yum install tree -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                               | 3.7 kB     00:00
Resolving Dependencies
--> Running transaction check
---> Package tree.x86_64 0:1.6.0-10.amzn2.0.1 will be installed
```

Exclude "ANSIBLE SERVER" from MultiExec mode

```
su: invalid option -- 'a'
Try 'su --help' for more information.
[root@ip-172-31-34-19 ec2-user]# su - awsdevops4344
Last login: Mon Jun 26 05:10:55 UTC 2023 on pts/0
[awsdevops4344@ip-172-31-34-19 ~]$
[awsdevops4344@ip-172-31-34-19 ~]$
[awsdevops4344@ip-172-31-34-19 ~]$
[awsdevops4344@ip-172-31-34-19 ~]$
[awsdevops4344@ip-172-31-34-19 ~]$ sudo yum install tree -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                               | 3.7 kB     00:00
Resolving Dependencies
--> Running transaction check
---> Package tree.x86_64 0:1.6.0-10.amzn2.0.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved
```

Exclude "NODE-1" from MultiExec mode

after sudo previliges

We are unable to connect AnsiServer to Nodes.

```
[awsdevops4344@ip-172-31-38-145 ~]$ ssh 172.31.34.19
The authenticity of host '172.31.34.19 (172.31.34.19)' can't be established.
ECDSA key fingerprint is SHA256:bubEeYSEbIf8gPADR/thn3jPM2fhM2f6X72kIVgcTtk.
ECDSA key fingerprint is MD5:b1:77:29:77:1a:5b:b1:bb:14:bf:ac:c2:70:31:69:83.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.31.34.19' (ECDSA) to the list of known hosts.
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[awsdevops4344@ip-172-31-38-145 ~]$ ssh awsdevops4445@172.31.34.19
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[awsdevops4344@ip-172-31-38-145 ~]$
```

Exclude "ANSIBLE SERVER" from MultiExec mode

```
[awsdevops4344@ip-172-31-38-145 ~]$ ssh 172.31.34.19
The authenticity of host '172.31.34.19 (172.31.34.19)' can't be established.
ECDSA key fingerprint is SHA256:bubEeYSEbIf8gPADR/thn3jPM2fhM2f6X72kIVgcTtk.
ECDSA key fingerprint is MD5:b1:77:29:77:1a:5b:b1:bb:14:bf:ac:c2:70:31:69:83.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.31.34.19' (ECDSA) to the list of known hosts.
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[awsdevops4344@ip-172-31-38-145 ~]$ ssh awsdevops4445@172.31.34.19
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[awsdevops4344@ip-172-31-38-145 ~]$ ssh 172.31.39.92
The authenticity of host '172.31.39.92 (172.31.39.92)' can't be established.
ECDSA key fingerprint is SHA256:GPE5gA0Q0xU63rI/kd2zA1DnOCsi5UVkt/EkwWbfXpg.
ECDSA key fingerprint is MD5:5e:b3:72:ec:5d:48:d2:96:53:6b:51:9f:77:01:2e:c7.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.31.39.92' (ECDSA) to the list of known hosts.
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[awsdevops4344@ip-172-31-38-145 ~]$
```

We have Password Oriented Connection .

```
[root@ip-172-31-38-145 ~]# su - awsdevops4344
Last login: Mon Jun 26 05:22:52 UTC 2023 on pts/0
[awsdevops4344@ip-172-31-38-145 ~]$ ssh 172.31.34.19
awsdevops4344@172.31.34.19's password:
Last login: Mon Jun 26 05:22:52 2023

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[awsdevops4344@ip-172-31-34-19 ~]$
```

Exclude "ANSIBLE SERVER" from MultiExec mode

```
[root@ip-172-31-34-19 ssh]# pwd
/etc/ssh
[root@ip-172-31-34-19 ssh]# ls
moduli       ssh_host_ecdsa_key      ssh_host_ed25519_key.pub
ssh_config   ssh_host_ecdsa_key.pub  ssh_host_rsa_key
sshd_config  ssh_host_ed25519_key    ssh_host_rsa_key.pub
[root@ip-172-31-34-19 ssh]# vi
[root@ip-172-31-34-19 ssh]# vi sshd_config
[root@ip-172-31-34-19 ssh]# vi sshd_config
[root@ip-172-31-34-19 ssh]# service sshd restart
Redirecting to /bin/systemctl restart sshd.service
[root@ip-172-31-34-19 ssh]#
```

ANSI SERVER Connected N1 & N2 : with Password

```
ECDSA key fingerprint is MD5:5e:b3:72:ec:5d:48:d2:96:53:6b:51:9f:77:01:2e:c7.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.31.39.92' (ECDSA) to the list of known hosts.
awsdevops4344@172.31.39.92's password:
Last login: Mon Jun 26 05:22:52 2023

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[awsdevops4344@ip-172-31-39-92 ~]$
[awsdevops4344@ip-172-31-39-92 ~]$
[awsdevops4344@ip-172-31-39-92 ~]$
[awsdevops4344@ip-172-31-39-92 ~]$
[awsdevops4344@ip-172-31-39-92 ~]$
[awsdevops4344@ip-172-31-39-92 ~]$ exit
logout
Connection to 172.31.39.92 closed.
[awsdevops4344@ip-172-31-34-19 ~]$
```

ANSI SERVER Connected N1 & N2 : with out Password

```
[awsdevops4344@ip-172-31-34-19 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/awsdevops4344/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/awsdevops4344/.ssh/id_rsa.
Your public key has been saved in /home/awsdevops4344/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:QwZmFfcKwOpC7XefRwC54H8rJ9jf1UpHZrEIJ8bVxTQ awsdevops4344@ip-172-31-34-19.ap-south-1.compute.internal
The key's randomart image is:
+---[RSA 2048]----+
|     •=.oo.  ..E+|
|     oooo....  .o|
|   . o .+o =.. . |
|  . o .o..o.+ . o|
| . o   .S .. . .+|
|  . o . o.. .  = |
|   . . + o +  o o|
|      . + =..o o |
|         =... .  |
+----[SHA256]-----+
[awsdevops4344@ip-172-31-34-19 ~]$
```

2 keys : public & private

```
[awsdevops4344@ip-172-31-34-19 ~]$ ls -a
.  ..  .bash_history  .bash_logout  .bash_profile  .bashrc  .ssh
[awsdevops4344@ip-172-31-34-19 ~]$ cd .ssh/
[awsdevops4344@ip-172-31-34-19 .ssh]$ ls
id_rsa  id_rsa.pub  known_hosts
[awsdevops4344@ip-172-31-34-19 .ssh]$
```

Generate Keys of ANSIBLE SERVER :

```
2. ANSIBLE SERVER    3. NODE-1    4. NODE-2
Connection to 172.31.34.19 closed.
[awsdevops4344@ip-172-31-38-145 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/awsdevops4344/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/awsdevops4344/.ssh/id_rsa.
Your public key has been saved in /home/awsdevops4344/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:/8yzqizcFd4ouo9GwzhlttboYEM6qEIz5hh6wtKwMmc awsdevops4344@ip-172-31-38-145.ap-south-1.compute.internal
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|                 |
|                 |
|     . +   .     |
| . o * +S. +     |
|+=o * B o.+ .    |
|O=oo O + o.      |
|@oE   *o.  +.    |
|+*   .o++ ... =o |
+----[SHA256]-----+
```

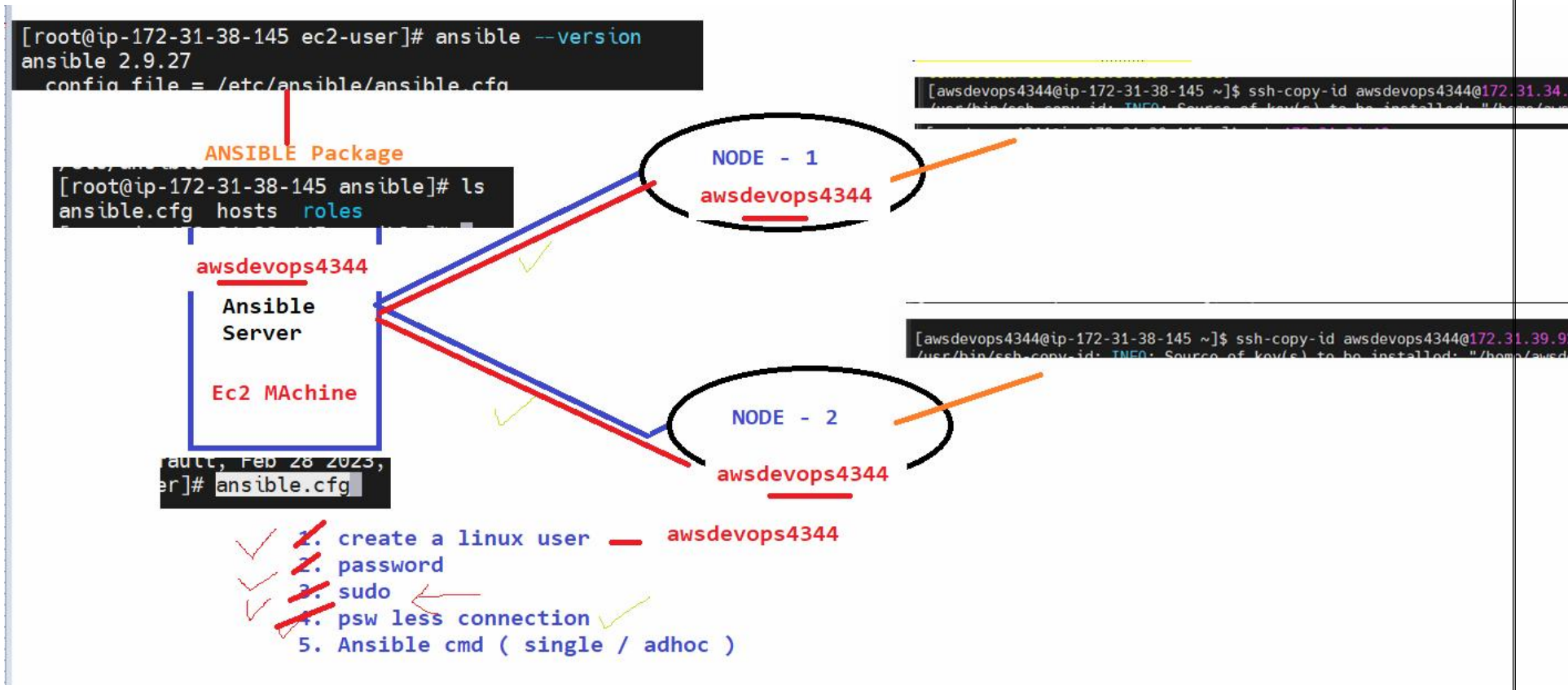
Copy this key for all the nodes (node1 , node2)

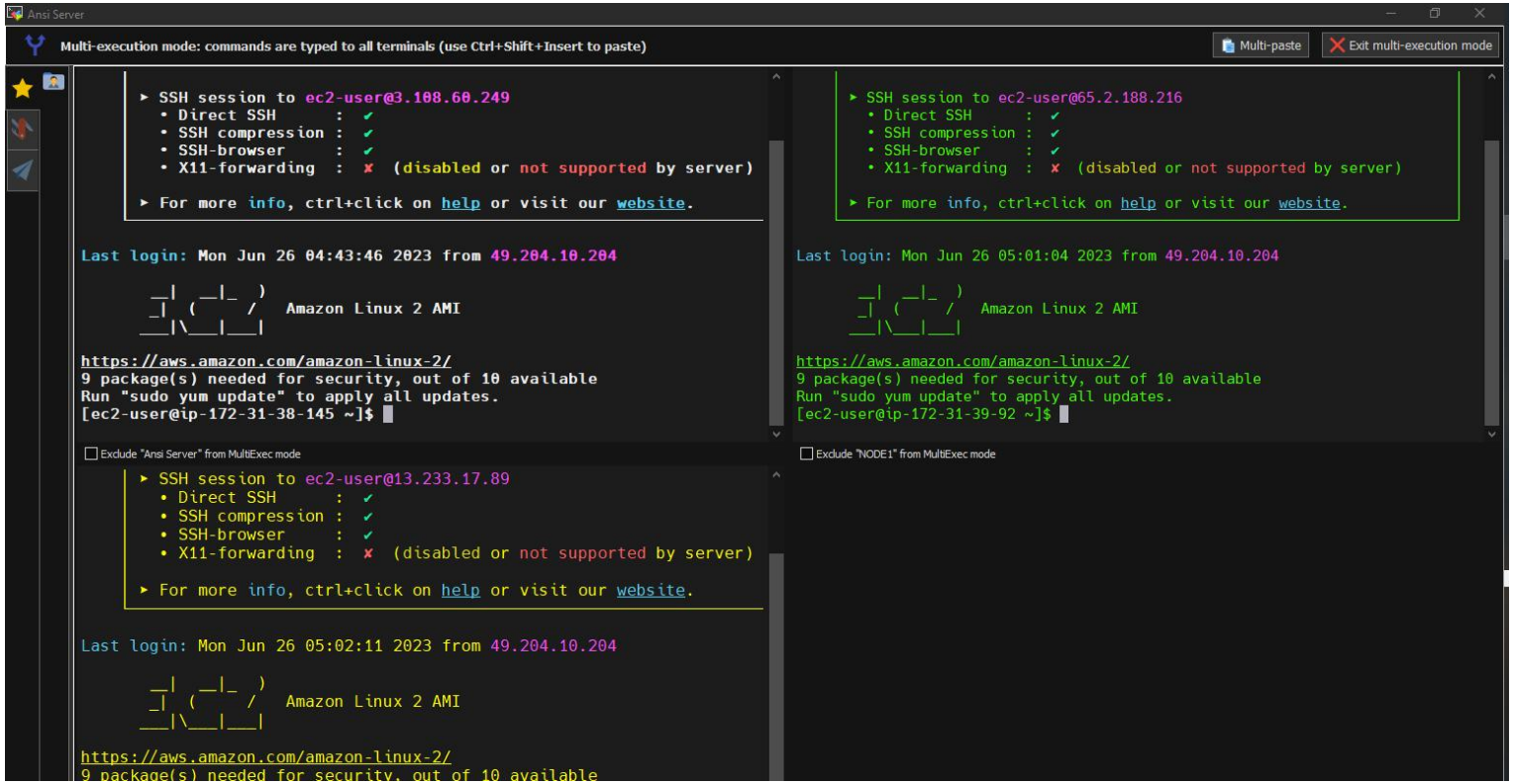
```
[awsdevops4344@ip-172-31-38-145 ~]$ ssh-copy-id awsdevops4344@172.31.34.19
```

```
[awsdevops4344@ip-172-31-38-145 ~]$ ssh 172.31.34.19
Last login: Mon Jun 26 05:42:04 2023 from ip-172-31-38-145.ap-south-1.compute.internal

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
```

```
[awsdevops4344@ip-172-31-38-145 ~]$ ssh-copy-id awsdevops4344@172.31.39.92
```

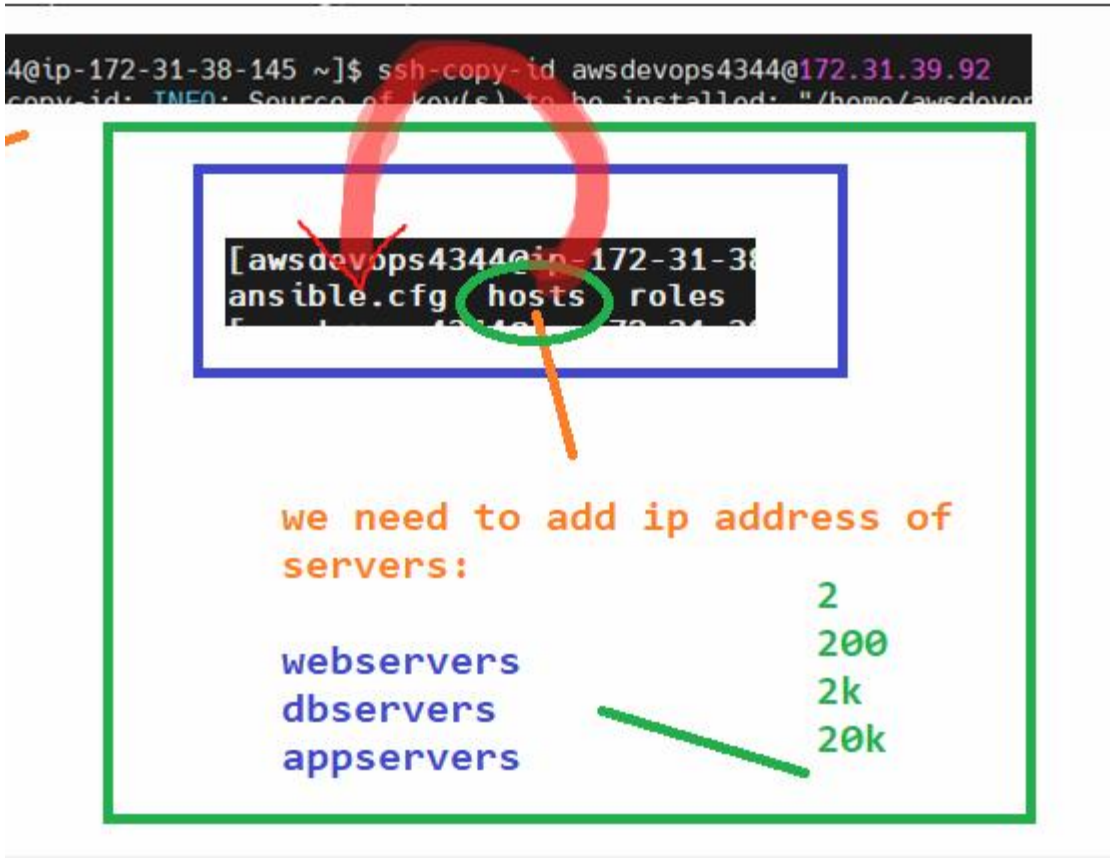
```
[awsdevops4344@ip-172-31-38-145 ~]$ ssh 172.31.39.92
Last login: Mon Jun 26 05:44:36 2023 from ip-172-31-34-19.ap-south-1.compute.internal

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[awsdevops4344@ip-172-31-39-92 ~]$
[awsdevops4344@ip-172-31-39-92 ~]$
[awsdevops4344@ip-172-31-39-92 ~]$
```


[root@ip-172-31-38-145 ec2-user]# ansible --version
ansible 2.9.27
ANSIBLE Package
[root@ip-172-31-38-145 ansible]# ls
ansible.cfg hosts roles
awsdevops4344
Ansible Server
Ec2 MAchine
ansible.cfg
NODE - 1
awsdevops4344
NODE - 2
awsdevops4344
[awsdevops4344@ip-172-31-38-145 ~]$ ssh-copy-id awsdevops4344@172.31.34.
[awsdevops4344@ip-172-31-38-145 ~]$ ssh-copy-id awsdevops4344@172.31.39.9
1. create a linux user
2. password
3. sudo
4. psw less connection
5. Ansible cmd (single / adhoc)
awsdevops4344




```
[awsdevops4344@ip-172-31-38-145 ~]$ sudo ls /etc/ansible
ansible.cfg  hosts  roles
[awsdevops4344@ip-172-31-38-145 ~]$
```

Main point


4@ip-172-31-38-145 ~]$ ssh-copy-id awsdevops4344@172.31.39.92
ansible.cfg hosts roles
we need to add ip address of servers:
webservers
dbservers
appservers
2
200
2k
20k

```
[awsdevops4344@ip-172-31-38-145 ~]$ sudo ls /etc/ansible
ansible.cfg  hosts  roles
[awsdevops4344@ip-172-31-38-145 ~]$ sudo vi /etc/ansible/hosts
```

```
# Ex 1: Ungrouped hosts, specify before any group headers.

[webservers]
172.31.39.92
172.31.34.19

## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10
```

Name of the Host

```
[webservers]
172.31.39.92
172.31.34.19
172.31.35.41
172.31.39.47
[appserver]
172.31.35.41
172.31.39.47
[dbserver]
172.31.39.50
```

add servers based on HOST

Whatever hosts we have that need to impact on ANSIBLE configure

```
[awsdevops4344@ip-172-31-38-145 ~]$ sudo vi /etc/ansible/ansible.cfg
```

Default :

```
# 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 ...

#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
#module_set_locale = False

# plays will gather facts by default, which contain information about
# the remote system.
```

Add permission to host machines

```
# 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
```

List of servers in my ansible

```
[awsdevops4344@ip-172-31-38-145 ~]$ ansible all --list-hosts
  hosts (5):
    172.31.39.92
    172.31.34.19
    172.31.35.41
    172.31.39.47
    172.31.39.50
[awsdevops4344@ip-172-31-38-145 ~]$
```

What is Host Patterns in Ansible ?

```
[webservers]
172.31.39.92
172.31.34.19
172.31.35.41
172.31.39.47
[appserver]
172.31.35.41
172.31.39.47
[dbserver]
172.31.39.50


## 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

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

webservers[0] --1st node
1
webservers[1] -- 2nd node
2

webservers[398] -- 399 node
399

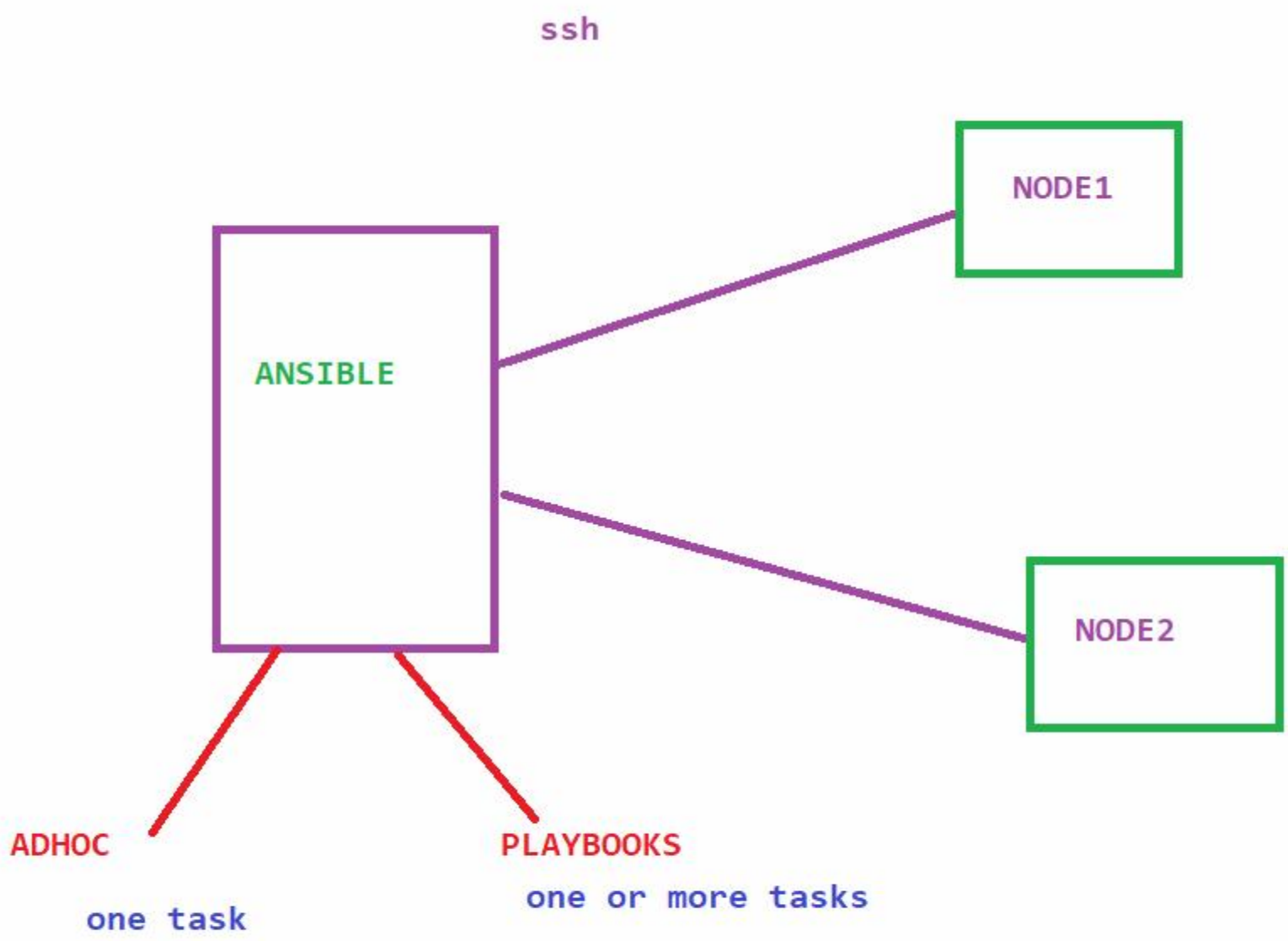
webservers[-1] -- last node
500

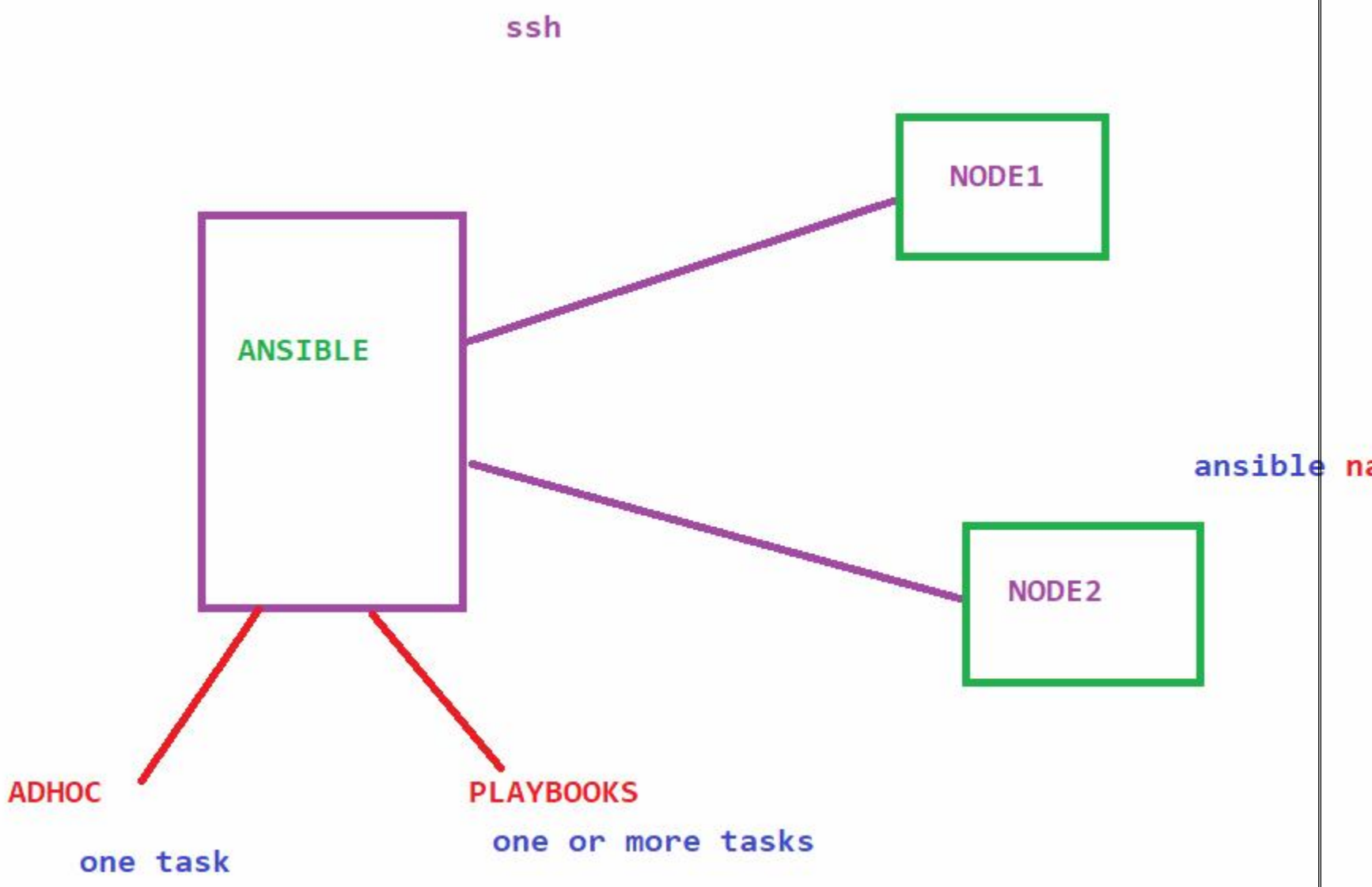
webservers[-2- -- last but one
499

webservers[0:4] -- 1st 5 nodes

webservers:appservers
webservers[2]:appservers[1]

all


ssh
NODE1
ANSIBLE
NODE2
ADHOC
one task
PLAYBOOKS
one or more tasks


ssh
NODE1
ANSIBLE
ansible na
NODE2
ADHOC
one task
PLAYBOOKS
one or more tasks

```
[awsdevops4344@ip-172-31-38-145 ~]$
```

```
[awsdevops4344@ip-172-31-39-92 ~]$ touch t1 t2 t3 t4 t5
[awsdevops4344@ip-172-31-39-92 ~]$ mkdir node1
[awsdevops4344@ip-172-31-39-92 ~]$ ls
node1  t1  t2  t3  t4  t5
[awsdevops4344@ip-172-31-39-92 ~]$
```

Exclude "Ansi Server" from MultiExec mode

```
[awsdevops4344@ip-172-31-34-19 ~]$ touch f1 f2 f3 f4 f5
[awsdevops4344@ip-172-31-34-19 ~]$ mkdir node2
[awsdevops4344@ip-172-31-34-19 ~]$ ls
f1  f2  f3  f4  f5  node2
[awsdevops4344@ip-172-31-34-19 ~]$
```

Exclude "NODE1" from MultiExec mode

```
-bash: cleaa: command not found
[awsdevops4344@ip-172-31-38-145 ~]$ clear
[awsdevops4344@ip-172-31-38-145 ~]$ ansible webservers -m command -a "ls"
[WARNING]: Platform linux on host 172.31.34.19 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.
172.31.34.19 | CHANGED | rc=0 >>
f1
f2
f3
f4
f5
node2
[WARNING]: Platform linux on host 172.31.39.92 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.
172.31.39.92 | CHANGED | rc=0 >>
node1
t1
t2
t3
t4
t5
172.31.35.41 | UNREACHABLE! => {
```

```
[awsdevops4344@ip-172-31-39-92 ~]$ touch t1 t2 t3 t4 t5
[awsdevops4344@ip-172-31-39-92 ~]$ mkdir node1
[awsdevops4344@ip-172-31-39-92 ~]$ ls
node1  t1  t2  t3  t4  t5
[awsdevops4344@ip-172-31-39-92 ~]$
```

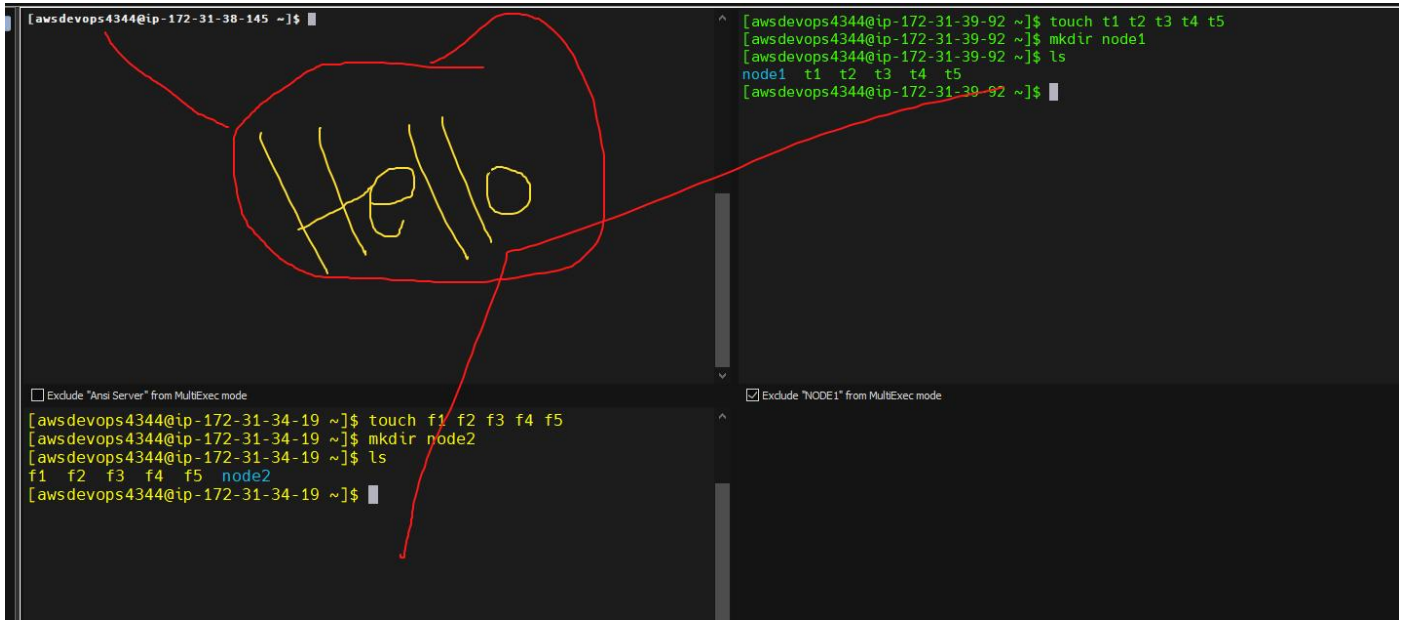
Exclude "Ansi Server" from MultiExec mode

```
[awsdevops4344@ip-172-31-34-19 ~]$ touch f1 f2 f3 f4 f5
[awsdevops4344@ip-172-31-34-19 ~]$ mkdir node2
[awsdevops4344@ip-172-31-34-19 ~]$ ls
f1  f2  f3  f4  f5  node2
[awsdevops4344@ip-172-31-34-19 ~]$
```

Exclude "NODE1" from MultiExec mode

ansible webservers -m command -a "ls"

ansible webservers -m command -a "ls -la"



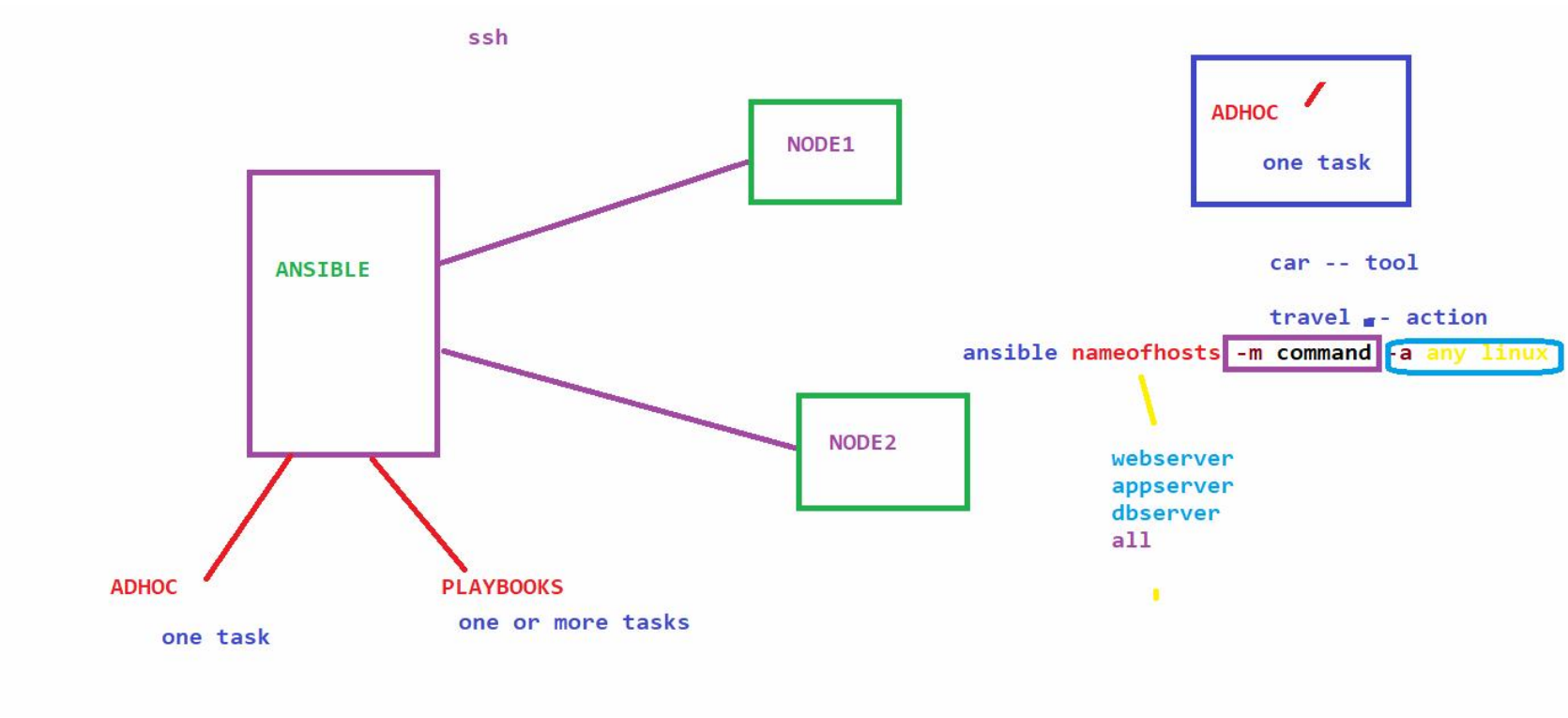
ansible webservers -m command -a "touch hello"

```
[awsdevops4344@ip-172-31-38-145 ~]$ ansible webservers -m command -a "touch hello"
[WARNING]: Consider using the file module with state=touch rather than running
'touch'.  If you need to use command because file 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.
[WARNING]: Platform linux on host 172.31.34.19 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.
172.31.34.19 | CHANGED | rc=0 >>

[WARNING]: Platform linux on host 172.31.39.92 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.
172.31.39.92 | CHANGED | rc=0 >>

172.31.35.41 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: ssh: connect to host 172.31.35.41 po
rt 22: No route to host",
    "unreachable": true
```

Check node end

```
[awsdevops4344@ip-172-31-34-19 ~]$ touch f1 f2 f3 f4 f5
[awsdevops4344@ip-172-31-34-19 ~]$ mkdir node2
[awsdevops4344@ip-172-31-34-19 ~]$ ls
f1  f2  f3  f4  f5  node2
[awsdevops4344@ip-172-31-34-19 ~]$ ls
f1  f2  f3  f4  f5  hello  node2
[awsdevops4344@ip-172-31-34-19 ~]$
```

```
node1  t1  t2  t3  t4  t5
[awsdevops4344@ip-172-31-39-92 ~]$ ls
hello  node1  t1  t2  t3  t4  t5
[awsdevops4344@ip-172-31-39-92 ~]$
```



Remove all unwanted python warnings

```
# some basic default values ...

inventory      = /etc/ansible/hosts
interpreter_python= auto_silent
#library        = /usr/share/my_modules/
#module_utils   = /usr/share/my_module_utils/
#remote_tmp     = ~/.ansible/tmp
#local_tmp      = ~/.ansible/tmp
```

Now no Warnings

```
[awsdevops4344@ip-172-31-38-145 ~]$ ansible webservers -m command -a "ls"
172.31.34.19 | CHANGED | rc=0 >>
f1
f2
f3
f4
f5
hello
node2
172.31.39.92 | CHANGED | rc=0 >>
hello
node1
t1
t2
t3
t4
t5
```

Only Check one Server :

```
[awsdevops4344@ip-172-31-38-145 ~]$ ansible webservers[0]  -m command -a "ls -a"
172.31.39.92 | CHANGED | rc=0 >>
.
..
.ansible
.bash_history
.bash_logout
.bash_profile
.bashrc
hello
node1
.ssh
t1
t2
t3
t4
t5
```

Last server Node

```
[awsdevops4344@ip-172-31-38-145 ~]$ ansible webservers[-1]  -m command -a "ls -a"
172.31.34.19 | CHANGED | rc=0 >>
.
..
.ansible
.bash_history
.bash_logout
.bash_profile
.bashrc
f1
f2
f3
f4
f5
hello
node2
.ssh
[awsdevops4344@ip-172-31-38-145 ~]$
```

How to install package in the all the nodes (tree)

```
[awsdevops4344@ip-172-31-38-145 ~]$ ansible webservers -m command -a "sudo yum install tree -y"
[WARNING]: Consider using 'become', 'become_method', and 'become_user' rather than running sudo
172.31.39.92 | CHANGED | rc=0 >>
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Package tree-1.6.0-10.amzn2.0.1.x86_64 already installed and latest version
Nothing to do
172.31.34.19 | CHANGED | rc=0 >>
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Package tree-1.6.0-10.amzn2.0.1.x86_64 already installed and latest version
Nothing to do
[awsdevops4344@ip-172-31-38-145 ~]$
```

Verifying

```
[awsdevops4344@ip-172-31-39-92 ~]$ tree
.
├── hello
├── node1
├── t1
├── t2
├── t3
├── t4
└── t5
```

Location of all the nodes:

```
[awsdevops4344@ip-172-31-38-145 ~]$ ansible webservers -m command -a "which tree"
172.31.34.19 | CHANGED | rc=0 >>
/usr/bin/tree
172.31.39.92 | CHANGED | rc=0 >>
/usr/bin/tree
[awsdevops4344@ip-172-31-38-145 ~]$
```

I want remove the package for particular node :

```
[awsdevops4344@ip-172-31-38-145 ~]$ ansible webservers[0]  -m command -a "sudo yum remove tree -y"
[WARNING]: Consider using 'become', 'become_method', and 'become_user' rather than running sudo
172.31.39.92 | CHANGED | rc=0 >>
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
---> Package tree.x86_64 0:1.6.0-10.amzn2.0.1 will be erased
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package     Arch          Version                     Repository          Size
================================================================================
Removing:
 tree        x86_64        1.6.0-10.amzn2.0.1          @amzn2-core         83 k

Transaction Summary
================================================================================
Remove  1 Package

Installed size: 83 k
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Erasing    : tree-1.6.0-10.amzn2.0.1.x86_64                                  1/1
  Verifying  : tree-1.6.0-10.amzn2.0.1.x86_64                                  1/1

Removed:
  tree.x86_64 0:1.6.0-10.amzn2.0.1

Complete!
[awsdevops4344@ip-172-31-38-145 ~]$
```

Validate node

```
[awsdevops4344@ip-172-31-39-92 ~]$ tree
-bash: /usr/bin/tree: No such file or directory
[awsdevops4344@ip-172-31-39-92 ~]$
```

Remove Package in the all the nodes.

```
[awsdevops4344@ip-172-31-38-145 ~]$ ansible all -m command -a "yum remove tree -y" -b
[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.
172.31.39.92 | CHANGED | rc=0 >>
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
No Packages marked for removalNo Match for argument: tree
172.31.34.19 | CHANGED | rc=0 >>
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
---> Package tree.x86_64 0:1.6.0-10.amzn2.0.1 will be erased
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package     Arch          Version                     Repository          Size
================================================================================
Removing:
 tree        x86_64        1.6.0-10.amzn2.0.1          @amzn2-core         83 k

Transaction Summary
================================================================================
Remove  1 Package

Installed size: 83 k
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Erasing    : tree-1.6.0-10.amzn2.0.1.x86_64                                 1/1
  Verifying  : tree-1.6.0-10.amzn2.0.1.x86_64                                 1/1

Removed:
  tree.x86_64 0:1.6.0-10.amzn2.0.1

Complete!
[awsdevops4344@ip-172-31-38-145 ~]$
```

- **ANSIBLE USING UBUNTU OS**

Configuration Management

This is process of configuring remote servers from one point of control.

Advantages

----

1) Provisioning of servers

The applications that should be installed on server can be done very quickly from a single centralized location.

2) Idempotent

Configuration management tools are used to bring the server to a particular state, called as desired state. If a server already in the desired state, configuration management tools will not reconfigure that server.

Note: Configuration management tools cannot be used for installing OS from the scratch.

They can be used only for managing the applications on top of the OS.

Configuration management tools - Ansible, chef, puppet, salt etc

++

Ansible -- It is a open source configuration management tool, created using Python.

Main machine in which anisble is installed, is called as controller.

Remote severs that Ansible configures, are called as managed nodes.

Ansible uses agent less policy for configures remote servers ie Ansible is installed only on 1 machine, and we do not require any client side software to be installed on the remote serers.

Ansible performs configuration management through password less ssh.

++++++++++++++++++++++++++++++++++

Create 4 Servers (Ubuntu 18)

1 is controller

3 are managed nodes

Name the instances as

Controller

Server1

Server2

Server3

Ubuntu machines default come with Python3

Ansible supports Python2

We need to downgrade the machines from python3 to Python2

Connect server1

Check the version

$ python3 --version

To Install Python2

$ sudo apt-get update

$ sudo apt-get dist-upgrade (It will point to older apt repository where python2 is available)

$ sudo apt-get install -y python2.7 python-pip

$ sudo apt-get install python3-pip

Now check the version of python

$ python --version

++++++++++++++++++++++++

Establish password less ssh connection

$ sudo passwd ubuntu

(lets give the password as ubuntu only)

$ sudo vim /etc/ssh/sshd_config

change

PasswordAuthentication yes

Save and QUIT

$ sudo service ssh restart

$ exit

++++++++++++++++++

Repeat the same steps in server2 and server3

++++++++++++++++

Now, Connect to controller

Even in controller also python2 version should be available

(So, run the same commands)

$ sudo apt-get update

$ sudo apt-get dist-upgrade

$ sudo apt-get install -y python2.7 python-pip

Now check the version of python

$ python --version

++++++++++++++++++

Now , We need to generate ssh connections

$ ssh-keygen

Now copy the key to managed nodes

$ ssh-copy-id ubuntu@172.31.0.98 (private Ip of server1)

$ ssh-copy-id ubuntu@172.31.1.183 (private Ip of server2)

$ ssh-copy-id ubuntu@172.31.14.179 (private Ip of server3)

+++++++++++++++++++++

Installing ansible now

Connect to controller.

$ sudo apt-get install software-properties-common

(software-properties-common , is a base package which is required to install ansible)

$ sudo apt-add-repository ppa:ansible/ansible

$ sudo apt-get update

$ sudo apt-get install -y ansible

++++++++++++++++

To check ther version of ansible

$ ansible --version

++++++++++++

Write the ip address of nodes in the inventory file

$ cd /etc/ansible

$ ls

$ sudo vim hosts

insert the private ip addresss of 3 servers

save and quit

$ ls -la (to see the list in the current machine)

$ ansible all -a 'ls -la' (you will get the list of the files in all managed nodes)

2 Ways ansible can

1) adhoc commands

2) playbooks

adhoc commands

Important modules in ansible

1) command - This module is used for executing basic linux commands on managed nodes.

2) shell - This module is used to execute commands which involved redirection and piping and to execute shell scripts on managed nodes.

3) ping -- This module is used to check if the remote server is pingable or not.

4) user -- This module is used for user management like create user, setting password, assign home directory etc

5) copy -- This module is used to copy the files and folders from controller to managed nodes

6) fetch -- This module is used to copy files and folder from managed nodes to controller

7) file -- This module is used for creating or deleting files and folders on managed nodes.

8) stat -- Used to capture detailed information about files and folders present in managed nodes.

9) debug -- Used to display output of any module

10) apt -- Used for performing package management on managed nodes ie installing softwares / upgrading repositories etc . It works on ubuntu, debain flavours of linux.

11) yum -- similar to apt module. It works on Red hat linux, centos etc

12) git -- used to perform git version controlling on managed nodes

13) replace -- This is used to replace specific text in configuration file with some other text.

14) service -- used for starting / stoping / restarting services on managed nodes.

15) include -- Used for calling child play books from parent play book

16) uri -- useful in checking if remote url is reachable or not.

17) docker_container -- used to execute docker commands related to container management on managed nodes

18) docker_image -- used to execute commands related to docker images on managed nodes.

19) docker_login -- used to login to docker hub from managed nodes.

20) setup -- used to capturing system information related to the managed nodes.

++

$ ansible all -i /etc/ansible/hosts -m command -a 'free'

$ ansible all -i /etc/ansible/hosts -m command -a 'touch file1'

To check the file which is created

$ ssh 172.31.2.173 (this command will go that machine)

$ ls

$ exit (to come back to controller)

++++++++++++++++

To install docker in all managed nodes

$ ansible all -i /etc/ansible/hosts -m shell -a 'curl -fsSL https://get.docker.com -o get-docker.sh'

$ ansible all -i /etc/ansible/hosts -m shell -a 'sh get-docker.sh'

++++++++++++++

To check docker is installed or not

$ ssh 172.31.2.173

$ docker --version

$ exit (to come back to controller)

++++++++++++++++

Notes:

Ansible performs remote configurations in 2 ways

1) using adhoc commands

2) using play books

Syntx of adhoc commands

$ ansible all/group_name/ipaddress -i path_of_inventory_file -m modulename -a 'arguments'

++++++++++++++++

Ansible command module to check the memory info on all managed nodes

$ ansible all -i /etc/ansible/hosts -m command -a 'free'

++++++++++++++++

To open the default inventory file

$ sudo vim /etc/ansible/hosts

(Observation: 3 ip address are available)

++++++++++++++++

Now, I copy the first two IP address (in a new notepad file)

quit the inventory file

++++++++++++

Create my own inventory file

$ vim myinventory

go to insert mode

paste two ip address

save and quit

++++++++++++

To check the inventory file

$ cat myinventory

++++++++++++

$ ansible all -i myinventory -m command -a 'free'

Observation: free command works on only two machines

++++++++++++

If you do not mention the inventory file, it takes default inventory file.

ex:

$ ansible all -m command -a 'free'

++++++++++++++++++++++++++++++

command module is the default module in ansible

$ ansible all -a 'free'

++++++++++++++++++++++

Note:

The defualt inventory file is /etc/ansible/hosts and when using this inventory file, we need not use -i option.

ex:

$ ansible all -m command -a 'free'

The default module is module. When using command module we need not use -m option

ex:

$ ansible all -a 'free'

Shell Module

----

ansible command to execute ls -la and store the output into file1 on all the managed nodes.

$ ansible all -m shell -a 'ls -la > file2'

To check the file which is created

$ ssh 172.31.12.239

$ ls

$ exit   (to come back to controller)

++++++++++++++

command to install docker on all managed nodes

$ ansible all -m shell -a 'curl -fsSL https://get.docker.com -o get-docker.sh'

$ ansible all -m shell -a 'sh get-docker.sh'

++++++++++++++++++++++++

User Module:

(From controller)

To create new user

$ sudo useradd sai

$ vim /etc/passwd (User will be created in this file)

To set the password

$ sudo passwd sai (sai is the username)

++++++++++

Now, i want to create user in all managed nodes

$ ansible all -m user -a 'name=anu password=Babji'

(we ger error : permission denied)

$ ansible all -m user -a 'name=anu password=Babji' -b (become , for higher privileges on managed nodes)

++++++++++++++

To check if user is create or not

$ ssh 172.31.12.239

$ vim /etc/passwd

$ exit

++++++++++++++++++++

Command to create user and set home directory, user id, default working shell etc

Another example

$ ansible all -m user -a 'name=Ravi password=freefree uid=1234 comment="A regular user" home=/home/ubuntu/Ravi shell=/bin/bash' -b

To check for the new user

$ ssh 172.31.44.218

$ vim /etc/passwd

++++++++++

Install git in all managed nodes

$ ansible all -m apt -a 'name=git state=present' -b

Observation:

We get "changed": false

(That means git is already installed on it. The command has no effect in the nodes)

Now , run the below command

$ ansible all -m apt -a 'name=git state=absent' -b

(absent means - uninstall)

output, we get in yellow color

(scroll up) we get "changed":true

(The command is effected the instance)

Now if we run the below command (with present option)

$ ansible all -m apt -a 'name=git state=present' -b

we get "changed":true

Notes:

apt module -- This is used for package management.

1) ansible all -m apt -a 'name=git state=present' -b

state=present is for installation

state=latest for upgradation

state=absent for uninstallation

++

I want to update apt-repositoty and install tomcat8

ansible all -m apt -a 'name=tomcat8 state=present update_cache=yes' -b

The above command will update apt repository and install tomcat8

To update apt-repository on managed nodes update_cache=yes is used

++++++++++++++++++++++++++

File module

This is used to create files and folder on managed nodes

ansible all -m file -a 'name=/tmp/file5 state=touch'

To check the file which is create

$ ssh 172.31.12.239

$ cd /tmp

$ ls

$ exit

TO create a directory

ansible all -m file -a 'name=/tmp/dir1 state=directory'

To check the directory

$ ssh 172.31.39.33

$ cd /tmp

$ ls

$ exit

To delete the file

ansible all -m file -a 'name=/tmp/file5 state=absent'

++++++++++++++++++++++++

Notes:

Command to create a file on all managed nodes

ansible all -m file -a 'name=/tmp/file1 state=touch'

state=touch is to create files

state=directory is to create directory

state=absent is for deleting file/directory

++++++++++++++++++++

Now,

To know the current user

$ whoami

$ ansible all -m file -a 'name=file1 state=touch'

Now go to managed nodes and check the permission of the file

$ ssh 172.31.12.239

$ ls -l file1

Observe the permissions are rw-rw-r--

Now, I want to change the permissions from controller

$ exit (will come back to controller)

$ ansible all -m file -a 'name=file1 state=touch owner=Anu group=Ravi mode=700' -b

The above command will execute only if Anu user and Ravi group is available in all nodes.

Notes:

File module can be used to change the ownership, group ownership and permissions on the file.

Copy Module

This is used for copying the files from controller into managed nodes.

We know in the file /etc/passwd we have all the information about users

Now I want to copy the file into all nodes

$ ansible all -m copy -a 'src=/etc/passwd dest=/tmp'

To check the file which is copies

$ ssh 172.31.12.239

$ cd /tmp

$ ls

$ exit

++++++++++++++

Scenario:

I want to create tomcat users file in controller and copy the file in all the nodes

$ sudo vim tomcat-users.xml

Go to Insert mode

```
<tomcat-users>
  <user  username="training"  password="freefree"  roles="manager-script"/>
<tomcat-users>
```

:wq

$ ansible all -m copy -a 'src=tomcat-users.xml dest=/etc/tomcat8' -b

To check the file

$ ssh 172.31.12.239

$ cd /etc/tomcat8

$ ls

Open that file to check the contents

$ sudo cat tomcat-users.xml

++++++++++++++++++++++++++++++++

Ansible command to copy /etc/passwd file to all the managed nodes

$ ansible all -m copy -a 'src=/etc/passwd dest=/tmp'

++++++++++++++++++++

Create a tomcat-users.xml file on controller and copy it into all managed nodes into default location of tomcat ie /etc/tomcat8

$ sudo vim tomcat-users.xml

Go to Insert mode

```
<tomcat-users>
  <user  username="training"  password="freefree"  roles="manager-script"/>
<tomcat-users>
:wq
```

$ ansible all -m copy -a 'src=tomcat-users.xml dest=/etc/tomcat8' -b

++++++++++++++++++++++++++++++++++++

Create a file on the controller machine

$ cat > newfile1

aaaa

bbbbb

ccccc

ddddd

Ctrl+d

$ ls -l newfile1

we get the permissions

rw-rw-r--

When we copy the file we have the same permissions

$ ansible all -m copy -a 'src=newfile1 dest=/home/ubuntu'

To got managed node and check the permissions on the file. It remains the same

$ ssh 172.31.39.33

$ ls -l newfile1

$ exit

Command to copy with changes permissions

$ ansible all -m copy -a 'src=newfile1 dest=/home/ubuntu owner=root group=root mode=760' -b

Now, go to node and check the permissions

$ ssh 172.31.35.79

$ ls -l newfile1

$ exit

Notes:

Copy module is used to change the ownership, group ownership and permissions of the files that are copied to managed nodes.

```
$ ansible all -m  copy -a 'src=newfile1  dest=/home/ubuntu owner=root  group=root mode=760' -b
```

++++++++++++++++++++++++++++++++++++++

To copy the file , by replacing the old content with new content

```
$ ansible all -m  copy -a 'content="babji\n"  dest=newfile1'  -b
```

TO to managed node and check the content

```
$ ssh 172.31.11.96
$ sudo cat newfile1
$ exit
```

Notes: Copy module can also send content into the file

```
$ ansible all -m  copy -a 'content="babji\n"  dest=newfile1'  -b
```

++++++++++++++++++++++

Fetch Module (opposite of copy module)

Go to managed node

$ ssh 172-31-35-79

$ cd /etc/tomcat8

$ ls

There is server.xml file

I want to get the file (server.xml) from node to controller

$ exit (come back to controller)

$ ansible all -m fetch -a 'src=/etc/tomcat8/server.xml dest=/tmp' -b

Now to got tmp folder

$ cd /tmp

$ ls

You will find three folders. The names of the folers are IP address of managed nodes

$ cd 172.31.35.102

$ ls

$ cd etc

$ ls

$ cd tomcat8

$ ls

Notes:

Fetch module is used to copy files from managed nodes to controller.

Command to copy tomcat-server.xml file from all managed nodes into /tmp folder on the controller.

$ ansible all -m fetch -a 'src=/etc/tomcat8/server.xml dest=/tmp' -b

Git Modules

This is used to perform git version controlling on the managed nodes.

ansible all -m git -a 'repo=https://github.com/sunildevops77/repo1.git dest=/tmp/mygit' -b

The above command will download the files in all managed nodes.

Go to managed node and check

$ ssh 172.31.35.79

$ cd /tmp

$ ls

$ cd mygit

$ ls

$ exit

Notes:

Ansible command to clone remote git repository into all managed nodes

ansible all -m git -a 'repo=https://github.com/sunildevops77/rep1.git dest=/tmp/mygit' -b

++++++++++++++++++

Service Module

This is used for starting/ stoping / restarting the services.

Ansible command to restart tomcat8 on all managed nodes

$ ansible all   -m service   -a 'name=tomcat8 state=restarted' -b

state=restarted is for restarting a service

state=stopped is for stopping a running service

state=started is for starting a stopped service

Replace module

Go to managed node

$ ssh 172.31.36.52

$ cd /etc/tomcat8/

$ ls

$ sudo vim server.xml

Look for connector port , to see the port number in which it is running. (line 74)

Now, we want to change the port number on all managed nodes, in this scenario

we use replace module.

Quit the server.xml file

$ exit (to come back to controller)

$ ansible all -m replace -a 'regexp=8080 replace=9090 path=/etc/tomcat8/server.xml' -b

Lets check tomcat is respoding on 9090 port in managed node

Get public DNS from aws

ec2-13-251-114-207.ap-southeast-1.compute.amazonaws.com

ec2-13-234-48-168.ap-south-1.compute.amazonaws.com

Open Browser

URL --- ec2-13-251-114-207.ap-southeast-1.compute.amazonaws.com:9090

We will not get the page, becuase we need to restart the service

$ ansible all -m service -a 'name=tomcat8 state=restarted' -b

Now, try the above URL --- it Works!!

replace module

This is used for replacing a specific string with other string.

Ex:

Ansible command to change the port number of tomcat from 8080 to 9090

$ ansible all -m replace -a 'regexp=8080 replace=9090 path=/etc/tomcat8/server.xml' -b

uri module

I want to check facebook is reachable for not in all managed nodes.

$ ansible all -m uri -a 'url=http://facebook.com'

In the output (green color) status - 200

Give a invalid url , we get status as -1

Ex:

$ ansible all -m uri -a 'url=http://hgyi9cb.com'

Now, I want to stop tomcat in all managed nodes (Just repeat)

$ ansible all -m service -a 'name=tomcat8 state=stopped' -b

Notes:

url module is used to check if the url is reachable or not.

Command to check if facebook.com is reachable on all managed nodes.

$ ansible all -m uri -a 'url=http://facebook.com status=200'

++++++++++++++++++++++++++++++++++++

Lets have an example of all modules

Requirement: I want to install tomcat all manages nodes , then i want to copy users.xml in all managed nodes,

I want to change port number of tomcat , then i want to restart the service, finally i want to check

url is reachable or not.

1st we need to unintall tomcat in all managed nodes.

```
$ ansible  all  -m  apt -a 'name=tomcat8 state=absent  purge=yes'  -b
```

```
$  ansible all -m  apt -a 'name=tomcat8 state=present' -b

$  ansible all -m  copy  -a  'src=tomcat-users.xml  dest=/etc/tomcat8'  -b

$ ansible all  -m replace  -a 'regexp=8080 replace=9090 path=/etc/tomcat8/server.xml'  -b

$  ansible all   -m service    -a 'name=tomcat8  state=restarted'  -b
```

To check tomcat is running individually on all servers,

take the private ip of all nodes

172.31.11.96

172.31.6.207

172.31.12.138

$ ansible all -m uri -a 'url=http://172.31.11.96:9090'

It returns status as 200

Similarly check the other two nodes

$ ansible all -m uri -a 'url=http://172.31.6.207:9090'

$ ansible all -m uri -a 'url=http://172.31.12.138:9090'

++++++++++++++++++++

Notes:

Requirement.

I want to install tomcat all modules.Copy tomcat-users.xml in all managed nodes.

Change port number of tomcat from 8080 to 9090. Restart the tomcat8 service.

Finally i want to check url is reachable or not.

```
$  ansible all -m  apt -a 'name=tomcat8 state=present' -b
```

```
$  ansible all -m  copy  -a  'src=tomcat-users.xml  dest=/etc/tomcat8'  -b

$ ansible all  -m replace  -a 'regexp=8080 replace=9090 path=/etc/tomcat8/server.xml'  -b

$  ansible all   -m service    -a 'name=tomcat8  state=restarted'  -b
```

To check tomcat is running individually on all servers,

take the private ip of all nodes

172.31.11.96

172.31.6.207

172.31.12.138

```
$ ansible all -m   uri  -a 'url=http://172.31.11.96:9090  status=200'
```

It returns status as 200

Similarly check the other two nodes

$ ansible all -m uri -a 'url=http://172.31.6.207:9090 status=200'

$ ansible all -m uri -a 'url=http://172.31.12.138:9090 status=200'

Play books

Notes:

Adhoc commands are capable of working only on one module and one set of arguments.

When we want to perform complex configuration management activities,

adhoc commands will be difficult to manage.

In such scenarios, we use play books.

Play book is combination of plays.

Each play is designed to do some activity on the managed nodes.

These plays are created to work on single host or a group of hosts or all the hosts.

The main advantage of play books is reusability.

Play books are created using yaml files.

```
$ mkdir playbooks
$ cd playbooks
$ vim playbook1.yml
INSERT mode

---
- name: Install git and clone a remote repository
  hosts: all
  tasks:
    - name: Install git
      apt:
       name: git
```

```
    state: present
    update_cache: yes
  - name: clone remote git repository
    git:
      repo: https://github.com/babjiawsdevops/git-9am-batch
      dest: /home/ubuntu/newgit
...
```

To check the syntax:

$ ansible-playbook playbook1.yml --syntax-check

(Do not use tab when creating yml file)

To run the playbook

$ ansible-playbook playbook1.yml -b

++++++++++++++++++++++++++++++

Play books

Notes:

Adhoc commands are capable of working only on one module and one set of arguments.

When we want to perform complex configuration management activities,

adhoc commands will be difficult to manage.

In such scenarios, we use play books.

Play book is combination of plays.

Each play is designed to do some activity on the managed nodes.

These plays are created to work on single host or a group of hosts or all the hosts.

The main advantage of play books is reusability.

Play books are created using yaml files.

```
$ mkdir playbooks
$ cd playbooks
$ vim playbook1.yml
INSERT mode

---
- name: Install git and clone a remote repository
  hosts: all
  tasks:
    - name: Install git
      apt:
        name: git
        state: present
        update_cache: yes
```

```
  - name: clone remote git repository
    git:
      repo: https://github.com/babjiawsdevops/git-9am-batch.git
      dest: /home/ubuntu/newgit
...
```

To check the syntax:

```
$ ansible-playbook  playbook1.yml  --syntax-check
```

(Do not use tab when creating yml file)

To run the playbook

```
$ ansible-playbook  playbook1.yml  -b
```

++++++++++++++++++++++++++++++

2nd example on playbook

Create user on all managed nodes and I want to copy passwd file.

$ vim playbook2.yml

```
---
- name: Create user and copy passwd file
  hosts: all
  tasks:
        - name: User creation
          user:
           name: kiran
           password: babjibabji
           uid: 6779
           home: /home/kiran
```

```
    - name: Copy password into users home dir
      copy:
        src: /etc/passwd
        dest: /home/kiran
...
```

Save and quit

$

Check the syntax:

$ ansible-playbook playbook2.yml --syntax-check

To run

$ ansible-playbook playbook2.yml -b

TO check user is created in managed nodes:

$ ssh 172.31.2.173

$ vim /etc/passwd

To check if passwd file is copied to /home/kiran

$ cd /home/kiran

$ ls

$ exit

Ex 3: Playbook to configure tomcat8 (earlier example)

1st uninstall tomcat

$ ansible all -m apt -a 'name=tomcat8 state=absent purge=yes' -b

$ vim playbook3.yml

```
---
- name: Configure  tomcat8
  hosts: all
  tasks:
   - name: Install tomcat8
     apt:
      name: tomcat8
      state: present
   - name: copy tomcat-users.xml file
     copy:
      src:  /home/ubuntu/tomcat-users.xml
      dest: /etc/tomcat8
   - name: change port of tomcat from 8080 to 9090
     replace:
      regexp: 8080
      replace: 9090
```

```
    path: /etc/tomcat8/server.xml
  - name: restart tomcat8
    service:
     name: tomcat8
     state: restarted
  - name: check url response of server 1
    uri:
     url: http://172.31.7.134:9090
  - name:   check url response of server 2
    uri:
     url: http://172.31.3.46:9090
...
```

```
$ ansible-playbook  playbook3.yml  --syntax-check
$ ansible-playbook  playbook3.yml  -b
```

++++++++++++++++++++++++

Requirment:

Install apache2 in all managed nodes, Place our own content in default homepage

```
$ cd playbooks
$ vim playbook4.yml
```

```
---
- name: configuring apache2
  hosts: all
  tasks:
   - name: Install apache2
     apt:
      name: apache2
      state: present
```

Save and quit

$ ansible-playbook playbook4.yml -b

To check apache2 is installed

$ ssh 172.31.12.239

(Homepage of apache2 is present in /var/www/html)

$ cd /var/www/html

$ ls

we get index.html (this html file is default homepage of apache)

Editing the index.html page

This is possible using copy module.

```
$ exit

$ vim playbook4.yml


- name: configuring apache2

  hosts: all

  tasks:

   - name: Install apache2

     apt:

      name: apache2

      state: present

   - name: Edit index.html file

     copy:

      content: "Welcome to Playbooks\n"

      dest: /var/www/html/index.html


save and quit
```

$ ansible-playbook playbook4.yml -b

++++++++++++++++++++

How to open url in terminal?

by using elinks

Ex:

$ elinks http://google.com

We get error (elinks not found)

Let's install elinks

$ sudo apt-get install -y elinks

Now run the command

$ elinks http://google.com

Now we want to look at index.html file in managed nodes

$ elinks http://15.207.99.5

After editing the index.html file, i need to restart the service and check the url response

$ vim playbook4.yml

```
---
- name: configuring apache2
  hosts: all
  tasks:
   - name: Install apache2
     apt:
      name: apache2
      state: present
   - name: Edit index.html file
```

```
  copy:
   content: "Welcome to playbooks\n"
   dest: /var/www/html/index.html
- name: Restart apache2
  service:
   name: apache2
   state: restarted
- name: check url response of server1
  uri:
   url: http://172.31.7.134
   status: 200
- name: check url response of server2
  uri:
   url: http://172.31.3.46
   status: 200
- name: check url response of server3
  uri:
```

url: http://172.31.2.140

status: 200

...

ansible-playbook playbook4.yml -b

Notes:

Ex: Ansible playbook for configure apache2

++

Creating reusable playbooks using variables

3 Types of variables

1) Global scope variables (highest priority) - we pass values from command prompt

2) Host scope variables

3) play scope variables (least priority)

Ex of Global scope variables

$ vim playbook5.yml

```
---
- name: Install software packages
  hosts: all
  tasks:
   - name: Install/uninstall/update etc
     apt:
      name: tree
      state: present
      update_cache: yes
...
```

If we run the above play book 10 times, what happens? tree package will install 10 times.

The above play book is not reusable.

we make small changes to the above code

$ vim playbook5.yml

```
---
- name: Install software packages
  hosts: all
  tasks:
   - name: Install/uninstall/update etc
     apt:
      name: "{{a}}"
      state: "{{b}}"
      update_cache: "{{c}}"
```

...

To run the playbook by passing values to the variables

$ ansible-playbook playbook5.yml --extra-vars "a=git b=absent c=no" -b

(The above command will uninstall git from all nodes)

Run the same playbook with diffrent values

$ ansible-playbook playbook5.yml --extra-vars "a=tree b=present c=no" -b

++++++++++++++++++

Before going to host scope variables,

lets discuss play scope variables

Playscope variables are definined within the playbook and they can effect only in one single play.

Ex:

$ vim playbook7.yml

```
---
- name: Using play scope variable
  hosts: all
  vars:
   - a: tomcat8
```

```
  - b: present
  - c: no
 tasks:
  - name: Install tomcat8
    apt:
     name: "{{a}}"
     state: "{{b}}"
     update_cache: "{{c}}"
...
```

$ ansible-playbook playbook7.yml -b

(It will install tomcat8)

We can run by using extra vars from command line

$ ansible-playbook playbook7.yml --extra-vars "a=tree b=present c=no" -b

The above command will install tree because global scope variables have higher priority

Notes:

Playscope variables

These variables are definied at level of individual plays and they can effect only one play.

Ex:

```
---
- name: Using play scope variable
  hosts: all
  vars:
   - a: tomcat8
   - b: present
```

```
  - c: no
 tasks:
  - name: Install tomcat8
    apt:
     name: "{{a}}"
     state: "{{b}}"
     update_cache: "{{c}}"
...
```

Note: The above playbook works like a template, who's default behaviour is to install tomcat8

But, we can by pass that behaviour and make it work in some other software by passing the variables as extra vars

```
$ ansible-playbook  playbook7.yml  -b  --extra-vars "a=tree b=present c=no"  -b
```

The above command will install tree because global scope variables have higher priority

Notes:

Playscope variables

These variables are definied at level of individual plays and they can effect only one play.

Ex:

```
---
- name: Using play scope variable
  hosts: all
  vars:
   - a: tomcat8
   - b: present
   - c: no
```

```
 tasks:
  - name: Install tomcat8
    apt:
     name: "{{a}}"
     state: "{{b}}"
     update_cache: "{{c}}"
...
```

Note: The above playbook works like a template, who's default behaviour is to install tomcat8

But, we can by pass that behaviour and make it work in some other software by passing the variables as extra vars

++++++++++++++++++++

++++++++++++++++++++++++++++++++++++++

Today we will discuss about host scope variables

Lets create one more managed node.

So, we will have 1 controller 4 nodes.

In step 6 -- Add rule -- All Traffic -- Anywhere

Check the version in the new node

$ python3 --version

We need to downgrade the machines from python3 to Python2

To downgrade

$ sudo apt-get update

$ sudo apt-get dist-upgrade (It will point to older apt repository where python2 is available)

$ sudo apt-get install -y python2.7 python-pip

Now check the version of python

$ python --version

Establish password less ssh connection

$ sudo passwd ubuntu

(lets give the password as ubuntu only)

$ sudo vim /etc/ssh/sshd_config

change

PasswordAuthentication yes

Save and QUIT

$ sudo service ssh restart

$ exit

++++++++++++++++

Now, Connect to controller

Now , We need to generate ssh connections

$ ssh-keygen

Now copy the key to managed nodes

$ ssh-copy-id ubuntu@172.31.6.241 (private Ip of server4)

++++++++++

Now, we need to add the information of managed nodes in the inventory file.

Location of inventory file /etc/ansible

$ cd /etc/ansible

$ ls

$ sudo vim hosts

insert the private ip addresss of 4th server

save and quit

$ ansible all -a 'ls -la' (you will get the list of the files in all managed nodes)

++++++++++++++++++

We can do grouping using [groupname]

Ex:

To do grouping

$ sudo vim hosts

[webserver]

172.31.11.96

172.31.6.207

[appserver]

172.31.12.138

[dbserver]

172.31.31.161

++++++++++++++++++

$ ansible appserver -a 'free' (It runs on one machine 172.31.12.138)

$ ansible webserver -a 'free' (It runs on two machines)

$ ansible all -a 'free'

++++++++++++++++++++++

We can perform grouping on groups

$ sudo vim hosts

[webserver]

172.31.11.96

172.31.6.207

[appserver]

172.31.12.138

[dbserver]

172.31.31.161

[india:children]

webserver

dbserver

$ ansible india -a 'free'

Grouping in inventory file

$ sudo vim /etc/ansible/hosts

[webserver]

172.31.11.96

172.31.6.207

[appserver]

172.31.12.138

[dbserver]

172.31.31.161

[india:children]

webserver

dbserver

Host scope variables

These variables are classified into 2 types

1) Variables to work on group of hosts

2) Variables to work on single hosts

Variables to work on group of hosts

These variables are designed to work on group of hosts.

They are definined in a folder called group_vars

This group_vars folder should be presnent in the same folder where all the playbooks are present.

In this group_vars folder, we should create a file who's name is same as group_name in Inventory file.

In this file we create variables.

Varible which works on group of hosts

$ cd (enter)

$ cd playbooks

$ ls

Varibles which work in group of hosts are divided into two types

1) Variables which work in group of machines

2) Variables which work on one machine

Variables which work in group of machines

----

playbooks$ mkdir group_vars

Note: group_vars folder should be present in the same location of playbook files.

$ cd group_vars

$ vim webserver

a: Prakash

b: logiclabs

c: /home/Prakash

d: 67809

```
e: /bin/bash


Save and Quit


$ cd ..

playbooks$ vim playbook8.yml


---

- name: Using host scope variables

  hosts: webserver

  tasks:

   - name: User creation

     user:

      name: "{{a}}"

      password: "{{b}}"

      home: "{{c}}"

      uid: "{{d}}"
```

```
    shell: "{{e}}"
...
```

save and quit

TO run the playbook

$ ansible-playbook playbook8.yml -b (It runs on two machines)

++++++++++++++++++++++++++++

Lets add few more variables

$ cd group_vars

$ vim webserver

```
a: Prakash

b: durgasoft

c: /home/Prakash

d: 67809

e: /bin/bash

f: tree

g: present

h: no


save and quit


$ cd ..

$ vim playbook9.yml


---

- name: Using host scope variables

  hosts: webserver
```

```
 tasks:
  - name: Install software
    apt:
     name: "{{f}}"
     state: "{{g}}"
     update_cache: "{{h}}"
...
```

$ ansible-playbook playbook9.yml -b

++++++++++++++++++++++++

Variables to work on single hosts

Variables to work on single hosts

These variables are designed on single machine.

Thet are created in folder called host_wars

This host_wars folder should be created in the same location of where the playbooks are present.

playbooks$ mkdir host_vars

$ cd host_vars

$ vim 172.31.6.241 (172.31.6.241 private Ip of server4)

a: firewalld

b: present

c: yes

save and quit

$ cd ..

$ vim playbook10.yml

```
- name: Use host scope variables
  hosts: 172.31.6.241
  tasks:
   - name: Install firewall
     apt:
      name: "{{a}}"
      state: "{{b}}"
      update_cache: "{{c}}"
...
```

save and quit

```
$ ansible-playbook  playbook10.yml  -b
```

++

Implementing loops

Notes: Modules in ansible can be executed multiple times using loops.

$ vim playbook11.yml

```
- name: Install software packages
  hosts: webserver
  tasks:
   - name: Install software
     apt:
      name: "{{item}}"
      state: present
      update_cache: no
     with_items:
      - tree
```

```
    - git
    - default-jdk
    - apache2
...
```

$ ansible-playbook playbook11.yml -b

Ex: Playbook to install diffrent s/w packages

$ vim playbook11.yml

```
- name: Install software packages
  hosts: webserver
  tasks:
   - name: Install software
     apt:
```

```
    name: "{{item}}"
    state: present
    update_cache: no
   with_items:
    - tree
    - git
    - default-jdk
    - apache2
...
```

++++++++++++++++++++++++

Requirement:

Tree needs to be installed

Git needs to be unintalled

jdk needs to be updated

apache needs to be installed and update cache

```
$ cd playbooks
$ vim playbook12.yml
```

```
---
- name: Install software packages
  hosts: webserver
  tasks:
   - name: Install software
     apt:
      name: "{{item.a}}"
      state: "{{item.b}}"
      update_cache: "{{item.c}}"
     with_items:
      - {a: tree,b: present,c: no}
```

```
    - {a: git,b: absent,c: no}
    - {a: default-jdk,b: absent,c: no}
    - {a: apache2,b: present,c: yes}
...
```

save and quit

```
$ ansible-playbook  playbook12.yml  -b
```

++++++++++++++++++++++

Ex: For working on multiple modules with multiple with_items.

Requirement: To create multiple users and files/directories in user's home directories.

```
$ vim playbook13.yml
---
```

```
---
- name: Create users and create files/dir in users home dir
  hosts: all
  tasks:
   - name: Create multiple users
     user:
      name: "{{item.a}}"
      password: "{{item.b}}"
      home: "{{item.c}}"
     with_items:
      - {a: Farhan,b: durgasoft,c: /home/Farhan}
      - {a: Ravi,b: durgasoft,c: /home/ubuntu/Ravi}
   - name: creating files and directories in users home dir
     file:
      name: "{{item.a}}"
      state: "{{item.b}}"
     with_items:
```

```
    - {a: /home/Farhan/file1,b: touch}
    - {a: /home/ubuntu/Ravi/dir1,b: directory}
...
```

save and quit

```
$ ansible-playbook  playbook13.yml  -b
```

To check , user is created or not?

```
$ ssh 172.31.11.96
$ vim /etc/passwd
```

TO check files and dir are created or not

```
$ cd /home/Farhan
$ ls     ( we can see the file)
```

$ cd

$ pwd

$ cd Ravi

$ ls (we can see the dir)

$ exit

++++++++++++++++

Handlers

Handler is a piece of code which is executed, if some other module is executed successfully and it has made some changes.

Handlers are always executed only after all the tasks are executed.

Handlers are executed in the order that are mentioned in the handler section, and not in the order they are called in the tasks section.

Even if handler is called multiple times in the tasks section, it will be executed only once.

Requirement:

$ vim playbook14.yml

```
---
- name: Confugure apache2 using handlers
  hosts: all
  tasks:
   - name: Install apache2
     apt:
      name: apache2
      state: present
   - name: Edit index.html file
     copy:
      content: "Logiclabs\n"
```

```
    dest: /var/www/html/index.html
   notify: Restart apache2
 handlers:
  - name: Restart apache2
   service:
    name: apache2
    state: restarted
...
```

$ ansible-playbook playbook14.yml -b

Note:

As editing the index.html file is successfull, handler is executed.

If you re run the playbook, handler is not executed.

++++++++++++++++++++++

Error Handling

If any module fails in ansible,the execution of the playbook terminates over there.

When we know that certain module might fail, and still we want to continue playbook execution, we can use error handling.

The section of code which might generate an error should be given in block section.

If it generates an error, the control comes to rescue section.

Always section is executed every time, irespective of whether the block is successfull or failure.

$ vim playbook15.yml

```
---
- name: Error handling
  hosts: all
  tasks:
   - block:
      - name: Install apache1
```

```
  apt:
   name: apache1
   state: present
rescue:
 - name: Install apache2
  apt:
   name: apache2
   state: present
always:
 - name: Check url response
  uri:
   url: "{{item}}"
  with_items:
   - http://172.31.7.134
   - http://172.31.3.46
   - http://172.31.2.140
   - http://172.31.6.241
```

…

$ ansible-playbook playbook15.yml -b