Ansible Tutorial: Installation and Usage with Ansible Modules

softwaretestinghelp.com/ansible-tutorial-1

Hands-on Ansible Tutorial with Ansible installation, usage, and configuration with Ansible Modules:

We discussed about AWS Elastic Beanstalk in our previous tutorial.

LIST of Tutorials in this Ansible Series:

Tutorial #1: Ansible installation and modules
Tutorial #2: Ansible Playbooks and Vaults
Tutorial #3: Ansible Roles and Integration with Jenkins

Ansible is an open source tool that helps in task automation, application deployment, cloud provisioning and configuration management.

Also Read =>DevOps Training Tutorial list

So we are talking about IT orchestration where tasks are run in sequence in several different machines or servers.

Ansible does this by connecting to multiple machines through SSH and runs the tasks which have been configured into playbooks and uses a simple language called YAML (Yet Another Markup Language).

Overview of Ansible

Most importantly Ansible does not use an agent to automate tasks on different machines.

Ansible ensures maintaining exact versions and up to date information to the software packages.

For Example, if you want to install JDK 8 or Tomcat or any other software package in 10 or 20 different machines it is not actually feasible to go to all the machines and install them rather use Ansible to automate the installation or even software deployments using Playbooks and Inventory written in a very simple language.

So Ansible is:

- Free and Open Source
- Maintained by Redhat
- Essentially a server configuration
- Configuration Management

In this 3 part Ansible Tutorial series, we will discuss hands-on approach on the following topics:

- Installation and configuration process
- Inventory
- Ansible Modules
- Ad-hoc commands,
- Task automation using playbooks
- Ansible roles
- Ansible vault
- Ansible and AWS

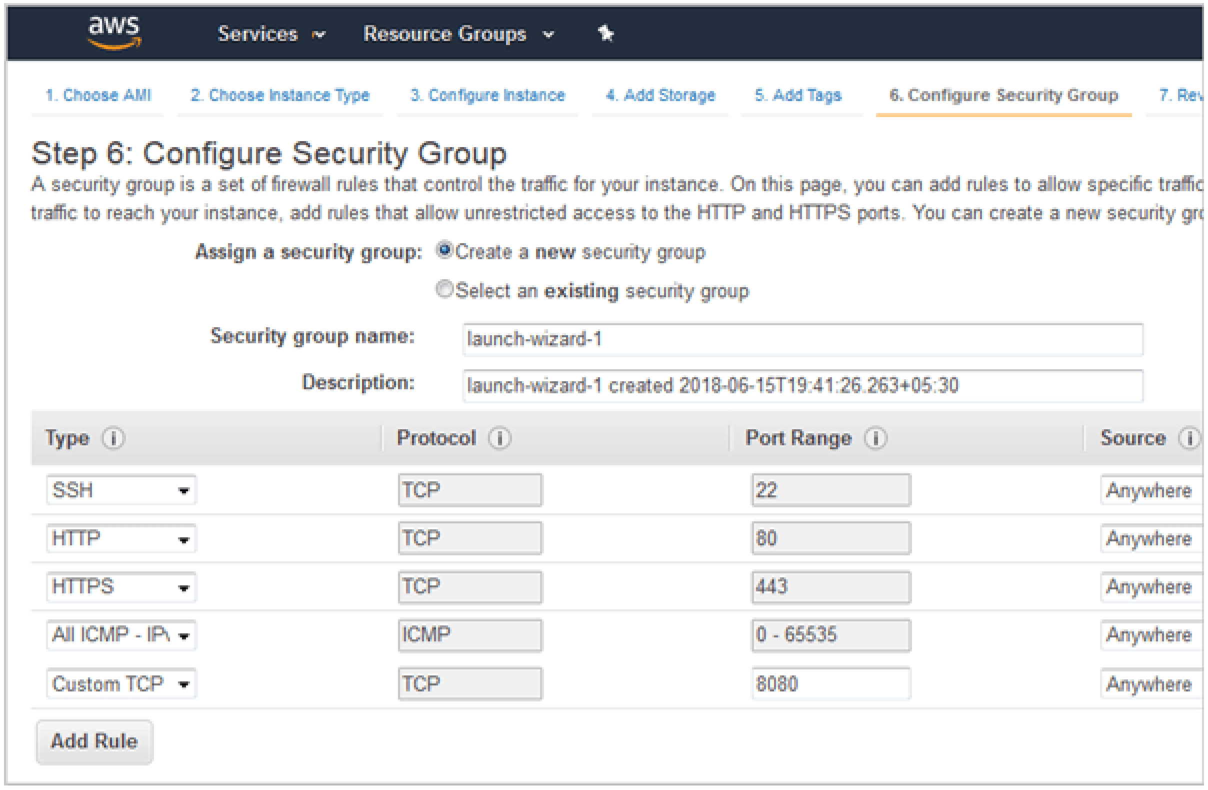
Ansible Installation Process

Ansible can be installed and run from any machine.

Typically you will need a **Control machine** for installation which should be Linux. Windows machine does not support being a control machine. The control machine will manage the other remote machines. As mentioned earlier Ansible uses SSH to manage remote machines.

Throughout this tutorial, I will be using AWS EC2 instances to showcase the examples. I have used 2 instances (one control machine and other as a target for automating tasks) and Redhat Linux 7.5.

Whether on-premise or cloud instances you will need to open ports appropriately based on the tasks being automated. I have the following ports open as a part of the security group for the EC2 instances to demonstrate the examples mentioned in the tutorial.



Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic
traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security gr

Assign a security group: [✓ Create a **new** security group] Select an **existing** security group

Security group name: launch-wizard-1

Description: launch-wizard-1 created 2018-06-15T19:41:26.263+05:30

Type	Protocol	Port Range	Source
SSH	TCP	22	Anywhere
HTTP	TCP	80	Anywhere
HTTPS	TCP	443	Anywhere
All ICMP - IP\	ICMP	0 - 65535	Anywhere
Custom TCP	TCP	8080	Anywhere

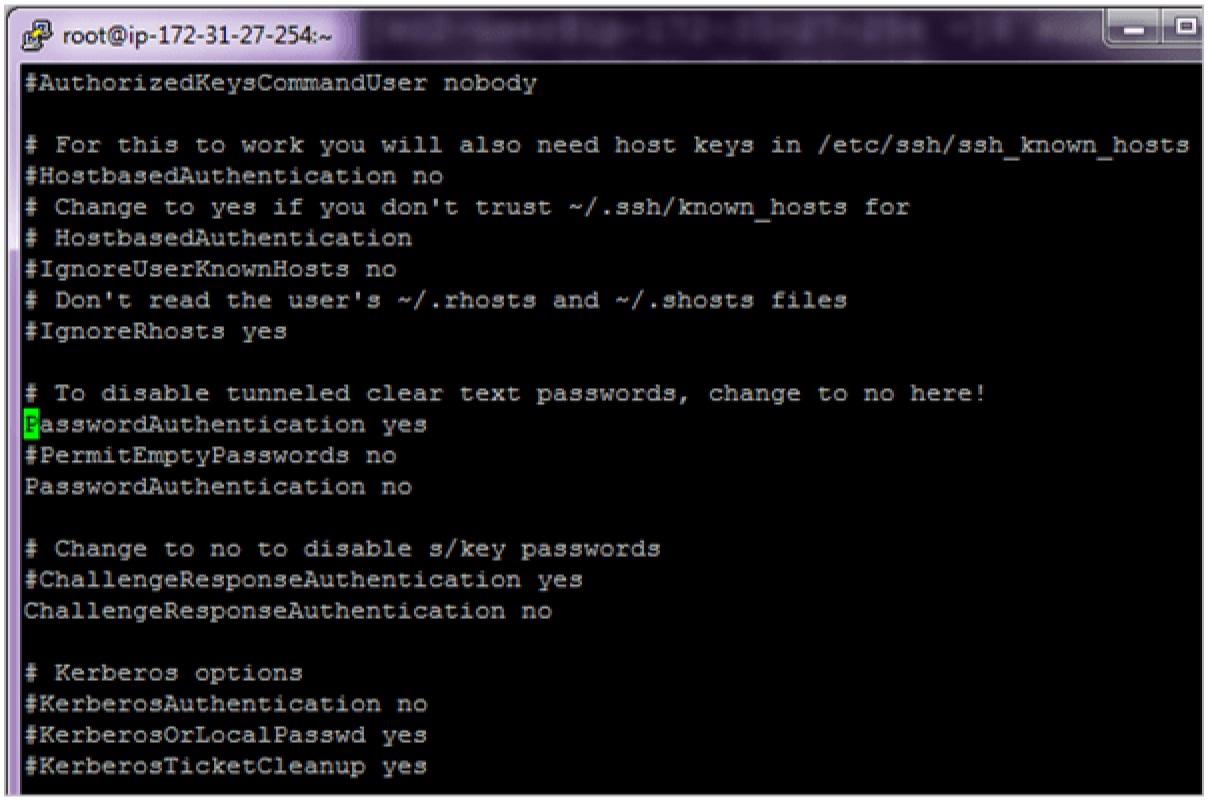
In the above screen, I have mentioned opening port 8080 as I will be showing about automating software deployment automation using Tomcat which will be useful from a DevOps point of view, especially during the continuous delivery process.

As mentioned before I will be using one control machine and a target machine. To start with installation, perform the steps as shown below in both the machines.

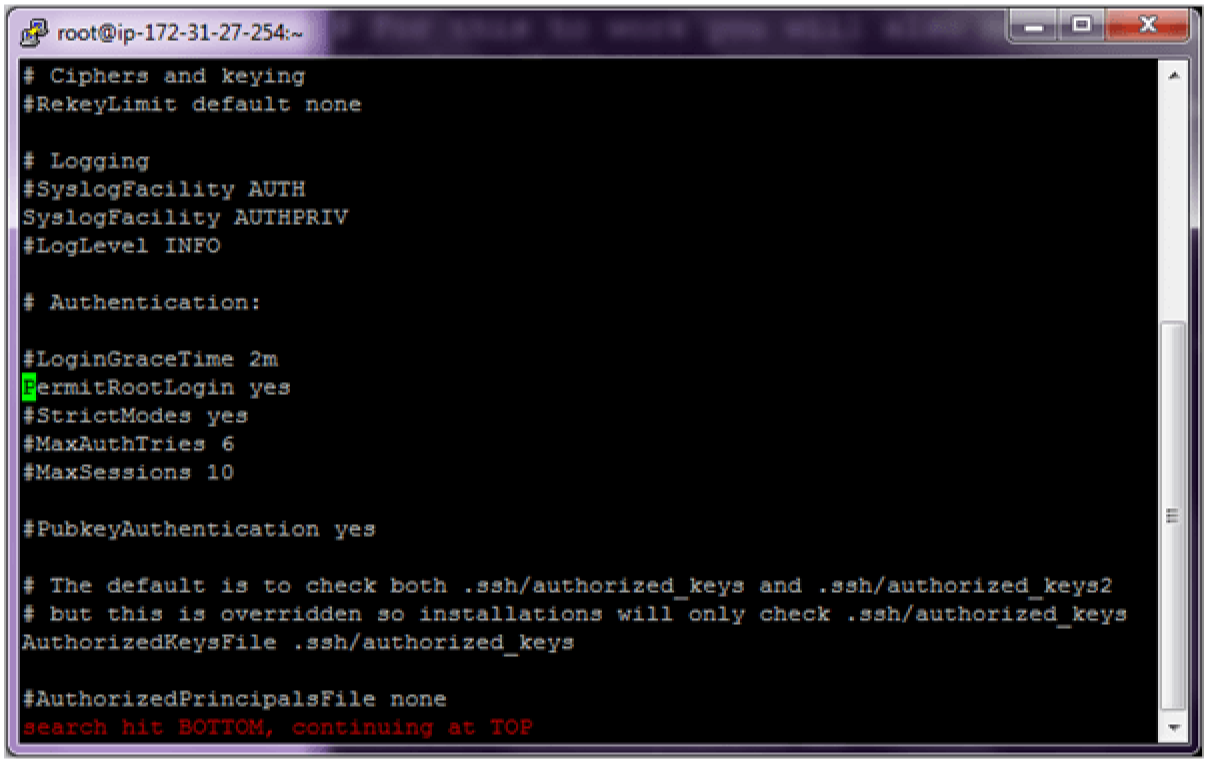
a) Create a common id on both the machines, for **Example**, **ansible** with SUDO privileges. This id will be used for communicating across all the machines involved for automation of tasks.

```
# useradd ansible
# passwd ansible
```

b) Edit the **/etc/ssh/sshd_config** file on the **control machine** and uncomment out the lines for **PasswordAuthentication and PermitRootLogin**



```
#AuthorizedKeysCommandUser nobody

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

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

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

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
```



```
# Ciphers and keying
#RekeyLimit default none

# 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
search hit BOTTOM, continuing at TOP
```

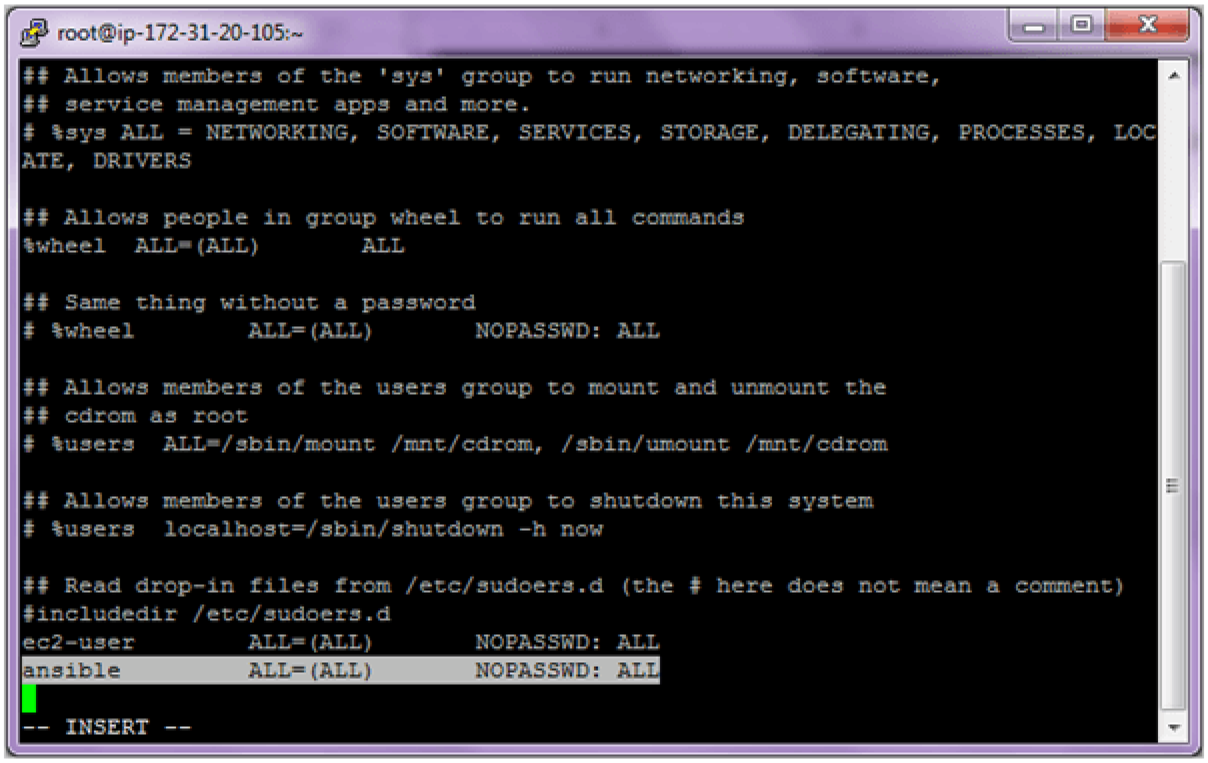
Perform the above steps on both the machines. Once completed, restart the **sshd** service on both the machines.

```
# systemctl restart sshd
```

c) For complete automation of tasks, we will need passwordless SSH authentication else the whole process will not be used if you have to key in the password every time.

So post the changes done above if we run the command ssh <target machine> and ssh <control machine> we will need to key in the password every time which is not the right procedure to execute Ansible tasks.

d) To enable passwordless authentication to perform the steps shown below. Firstly add the user **ansible** to the **/etc/sudoers** file on both the machines which will enable the user **ansible** to run any command which requires root privileges.

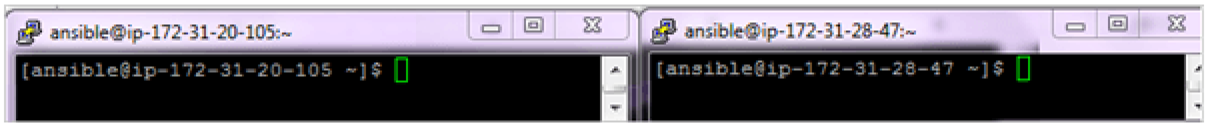


```
root@ip-172-31-20-105:~
## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES, LOC
ATE, DRIVERS

## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)       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

## Allows members of the users group to shutdown this system
# %users  localhost=/sbin/shutdown -h now

## Read drop-in files from /etc/sudoers.d (the # here does not mean a comment)
#includedir /etc/sudoers.d
ec2-user        ALL=(ALL)       NOPASSWD: ALL
ansible         ALL=(ALL)       NOPASSWD: ALL

-- INSERT --
```

Save and exit the file after adding the user.

e) Going forward we will use the user **ansible** to perform all the steps. So switch to the user **ansible.**

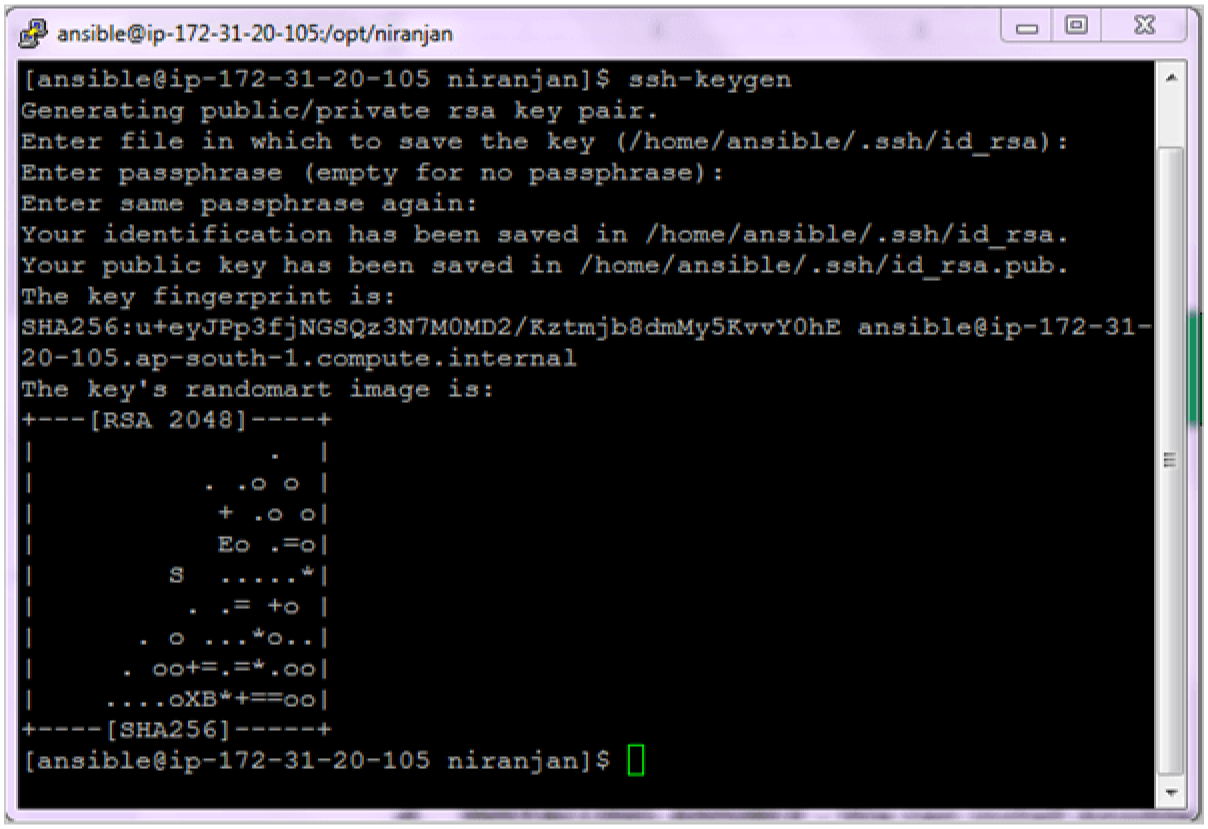
Control Machine su – ansible AND Target Machine su – ansible



```
ansible@ip-172-31-20-105:~
[ansible@ip-172-31-20-105 ~]$
```

```
ansible@ip-172-31-28-47:~
[ansible@ip-172-31-28-47 ~]$
```

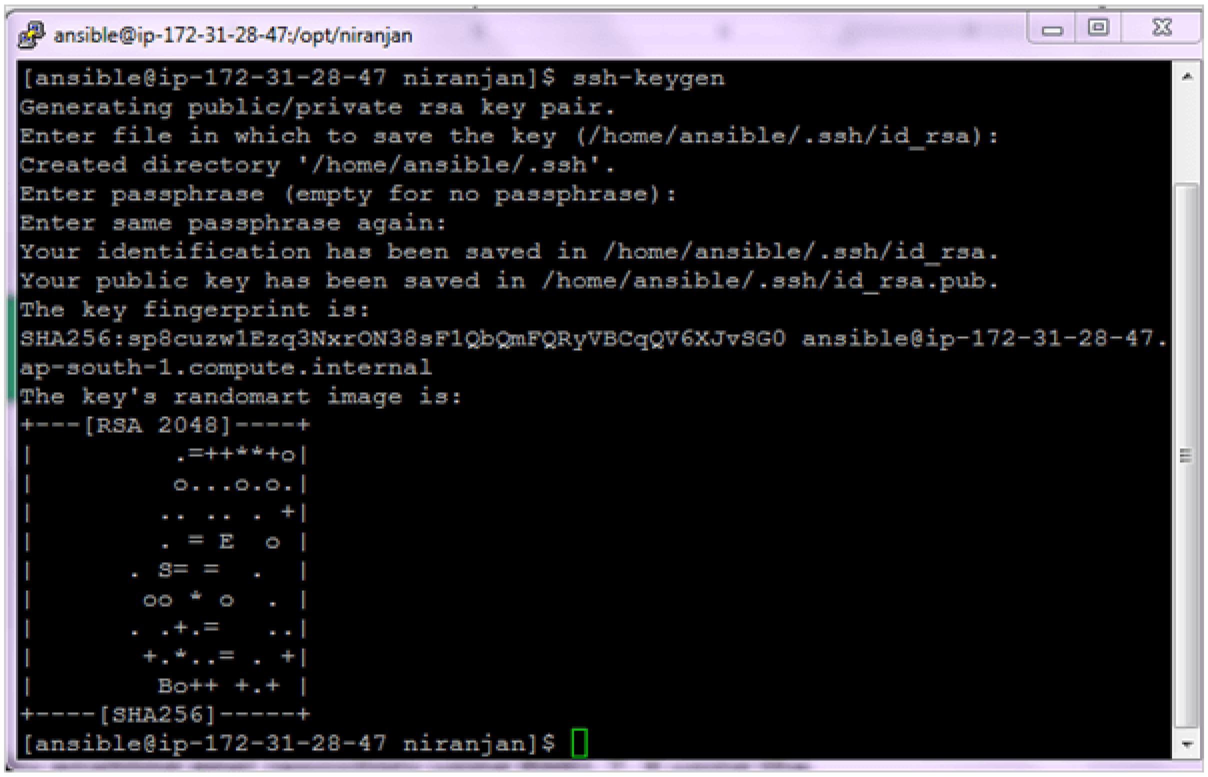
Control Machine ssh-keygen



ansible@ip-172-31-20-105:/opt/niranjan

```
[ansible@ip-172-31-20-105 niranjan]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ansible/.ssh/id_rsa):
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:u+eyJPp3fjNGSQz3N7M0MD2/Kztmjb8dmMy5KvvY0hE ansible@ip-172-31-
20-105.ap-south-1.compute.internal
The key's randomart image is:
+---[RSA 2048]----+
|              .  |
|          . .o o |
|           + .o o|
|           Eo .=o|
|        S  .....*|
|         . .= +o |
|      . o ...*o..|
|     . oo+=.=*.oo|
|    ....oXB*+==oo|
+----[SHA256]-----+
[ansible@ip-172-31-20-105 niranjan]$
```

Target Machine ssh-keygen

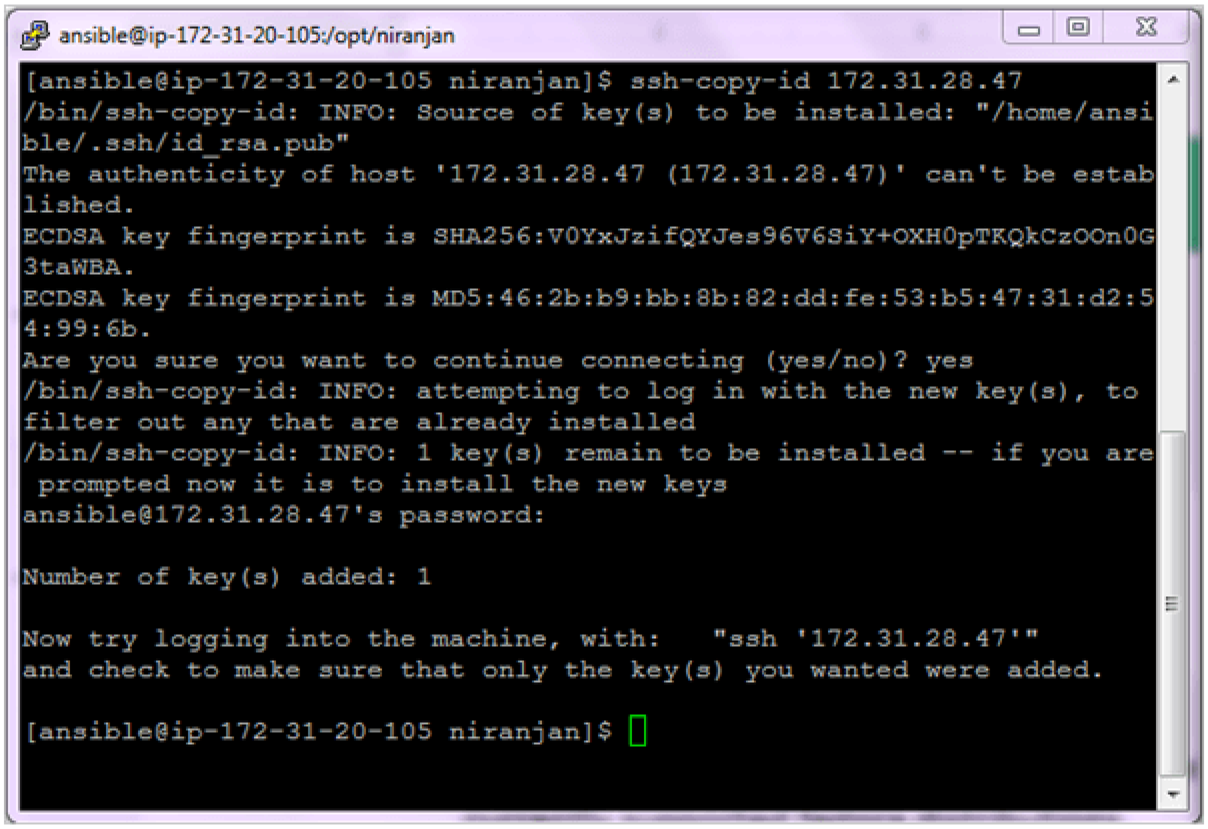


ansible@ip-172-31-28-47:/opt/niranjan

```
[ansible@ip-172-31-28-47 niranjan]$ 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:sp8cuzw1Ezq3NxrON38sF1QbQmFQRyVBCqQV6XJvSG0 ansible@ip-172-31-28-47.
ap-south-1.compute.internal
The key's randomart image is:
+---[RSA 2048]----+
|         .=++**+o|
|         o...o.o.|
|        .. .. . +|
|        . = E  o |
|      . S= =  .  |
|       oo * o  . |
|      . .+.=   ..|
|       +.*..= . +|
|        Bo++ +.+ |
+----[SHA256]-----+
[ansible@ip-172-31-28-47 niranjan]$
```

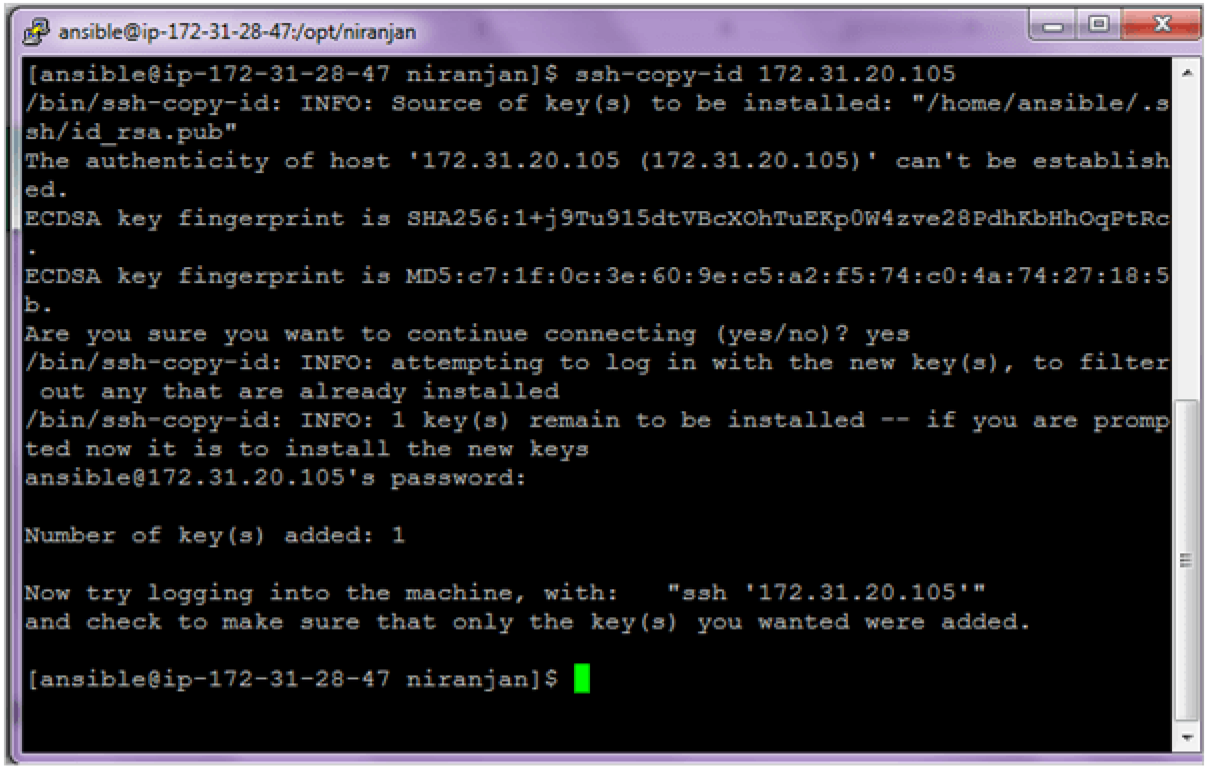
Copy the ssh key to the target machine and vice versa.

Control Machine ssh-copy-id <IP-Address-Host-Machine>



```
ansible@ip-172-31-20-105:/opt/niranjan
[ansible@ip-172-31-20-105 niranjan]$ ssh-copy-id 172.31.28.47
/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansi
ble/.ssh/id_rsa.pub"
The authenticity of host '172.31.28.47 (172.31.28.47)' can't be estab
lished.
ECDSA key fingerprint is SHA256:V0YxJzifQYJes96V6SiY+OXH0pTKQkCzOOn0G
3taWBA.
ECDSA key fingerprint is MD5:46:2b:b9:bb:8b:82:dd:fe:53:b5:47:31:d2:5
4:99:6b.
Are you sure you want to continue connecting (yes/no)? yes
/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to
filter out any that are already installed
/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
 prompted now it is to install the new keys
ansible@172.31.28.47's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh '172.31.28.47'"
and check to make sure that only the key(s) you wanted were added.

[ansible@ip-172-31-20-105 niranjan]$
```

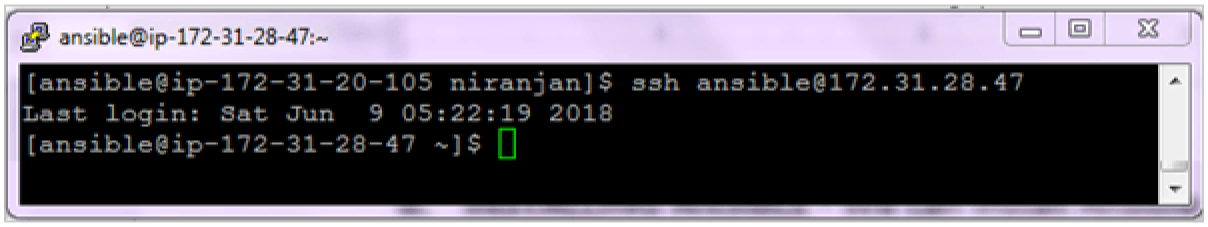
Target Machine ssh-copy-id <IP-Address-Control-Machine>



```
ansible@ip-172-31-28-47:/opt/niranjan
[ansible@ip-172-31-28-47 niranjan]$ ssh-copy-id 172.31.20.105
/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansible/.s
sh/id_rsa.pub"
The authenticity of host '172.31.20.105 (172.31.20.105)' can't be establish
ed.
ECDSA key fingerprint is SHA256:1+j9Tu915dtVBcXOhTuEKp0W4zve28PdhKbHhOqPtRc
.
ECDSA key fingerprint is MD5:c7:1f:0c:3e:60:9e:c5:a2:f5:74:c0:4a:74:27:18:5
b.
Are you sure you want to continue connecting (yes/no)? yes
/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
 out any that are already installed
/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are promp
ted now it is to install the new keys
ansible@172.31.20.105's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh '172.31.20.105'"
and check to make sure that only the key(s) you wanted were added.

[ansible@ip-172-31-28-47 niranjan]$
```

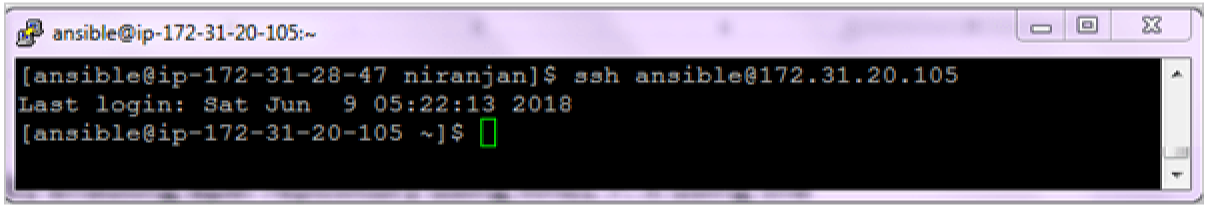
We are now able to log in without entering the password. After the check out of the ssh connectivity on both the machines and be logged in as ansible user.

Control Machine: ssh ansible@<IP-Address-Host-Machine



```
[ansible@ip-172-31-20-105 niranjan]$ ssh ansible@172.31.28.47
Last login: Sat Jun  9 05:22:19 2018
[ansible@ip-172-31-28-47 ~]$
```

Target Machine: ssh ansible@<IP-Address-Control-Machine>



```
[ansible@ip-172-31-28-47 niranjan]$ ssh ansible@172.31.20.105
Last login: Sat Jun  9 05:22:13 2018
[ansible@ip-172-31-20-105 ~]$
```

f) Install wget if not installed on both the machines.

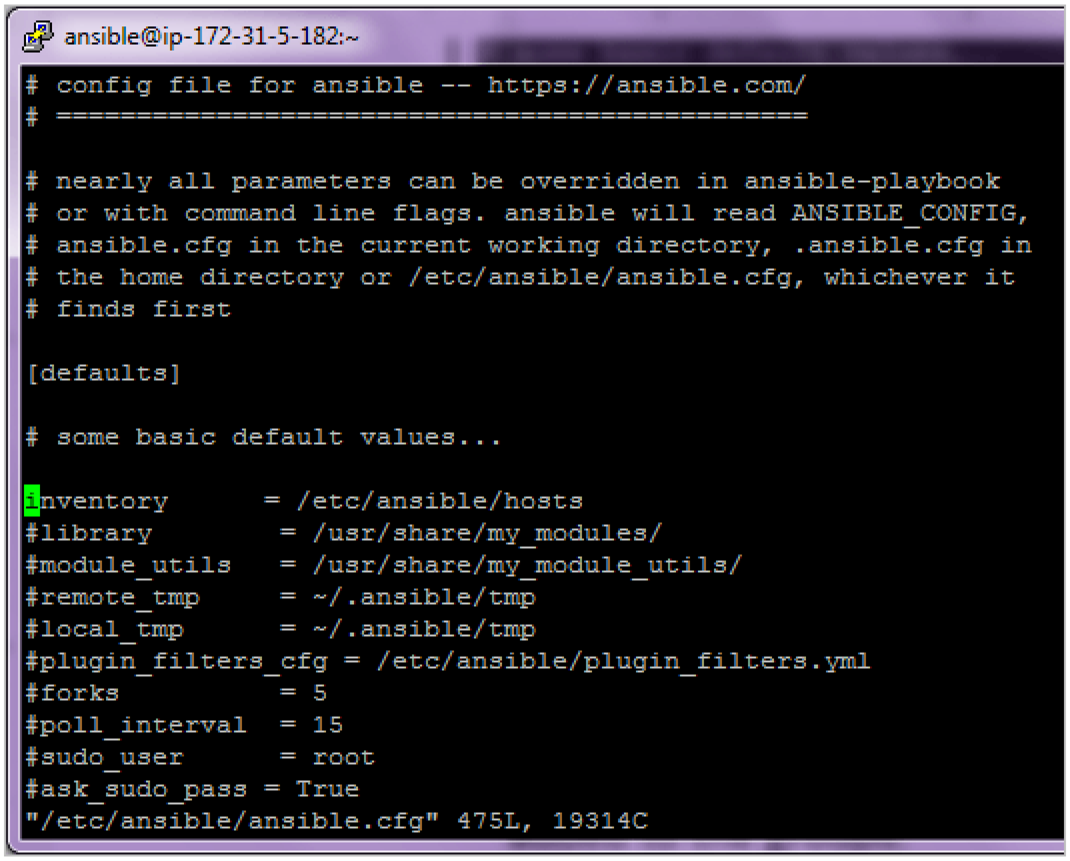
g) We can now install **ansible on the Control machine only** by enabling the EPEL repo from fedora which provides add-on software packages. Perform the following steps to install **ANSIBLE.**

```
$ wget http://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
$ sudo rpm -ivh epel-release-latest-7.noarch.rpm

$ ansible --version
```

The ansible version used is 2.5.3

h) Edit the **ansible.cfg** file and enable the inventory file parameter on the Control machine.

```
$ sudo vi /etc/ansible/ansible.cfg
```



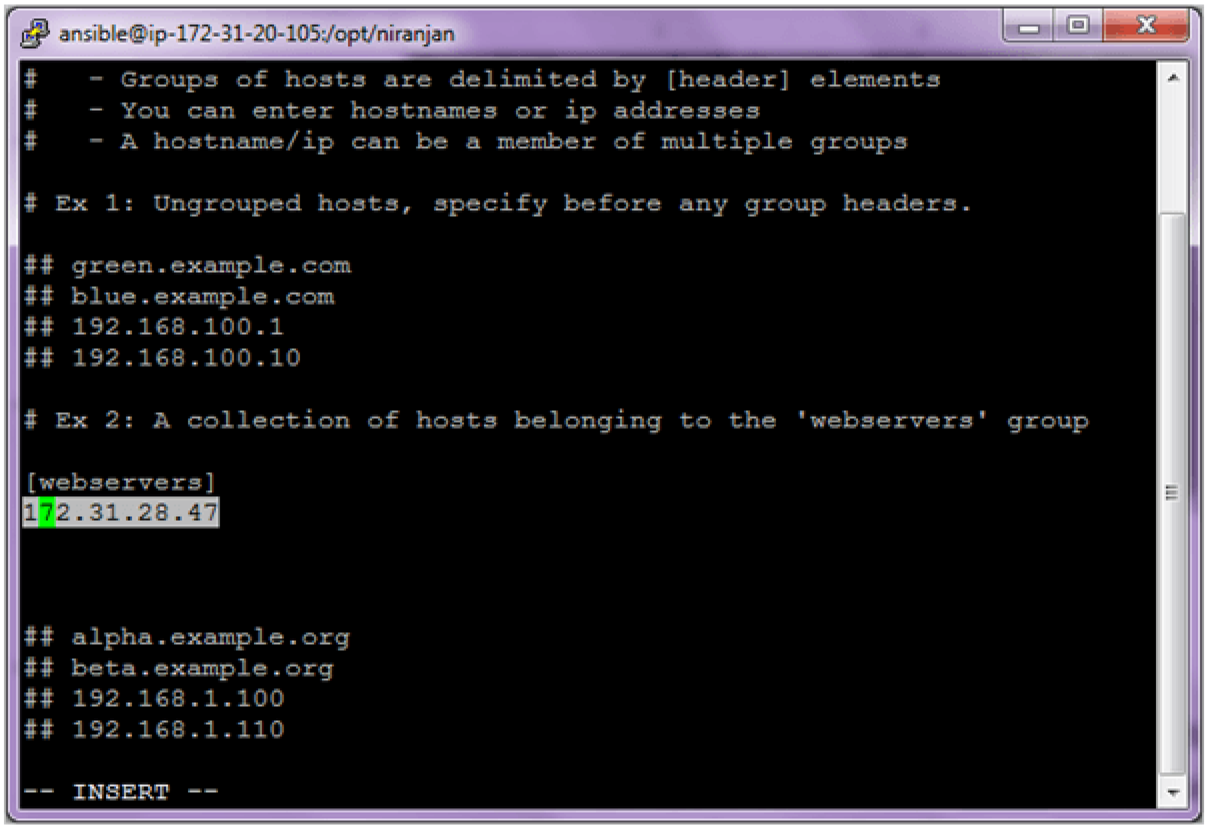
ansible@ip-172-31-5-182:~

```
# 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
"/etc/ansible/ansible.cfg" 475L, 19314C
```

i) Ansible uses the concept of Inventory to manage and track the target machines. By default, this file is located in **/etc/ansible/hosts** and can be changed as well. A host file consists of groups for better classification and multiple machines under the group. All the required machines can be added to those groups.

Every group is denoted by a square bracket and a group name within. A server can actually exist in multiple groups.

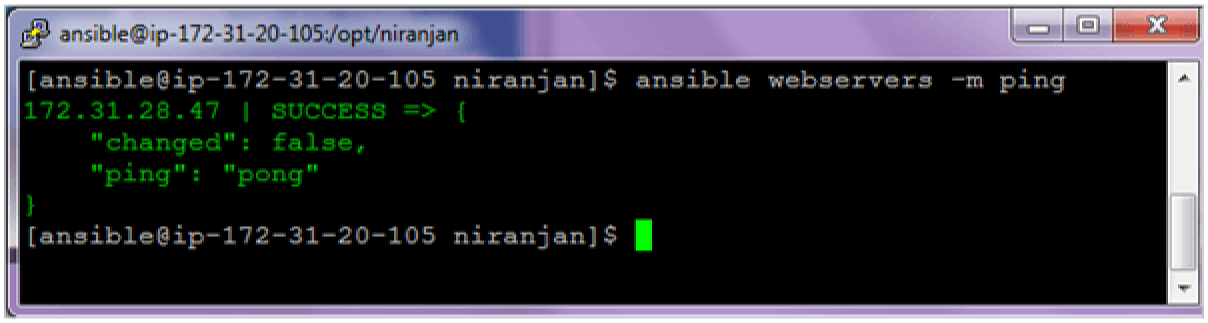
Edit the inventory file **/etc/ansible/hosts** and add all the servers which need to be managed.


ansible@ip-172-31-20-105:/opt/niranjan

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

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

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

# Ex 2: A collection of hosts belonging to the 'webservers' group

[webservers]
172.31.28.47



## alpha.example.org
## beta.example.org
## 192.168.1.100
## 192.168.1.110

-- INSERT --
```

j) To test the connectivity of the servers under the webserver's group run the **ansible ping** command as shown. Here **ping** is a module which performs a particular function to test whether the hosts can be connected as defined in the inventory file or not. We will see more about various modules and its examples in the next section.

```
$ ansible webservers -m ping
```

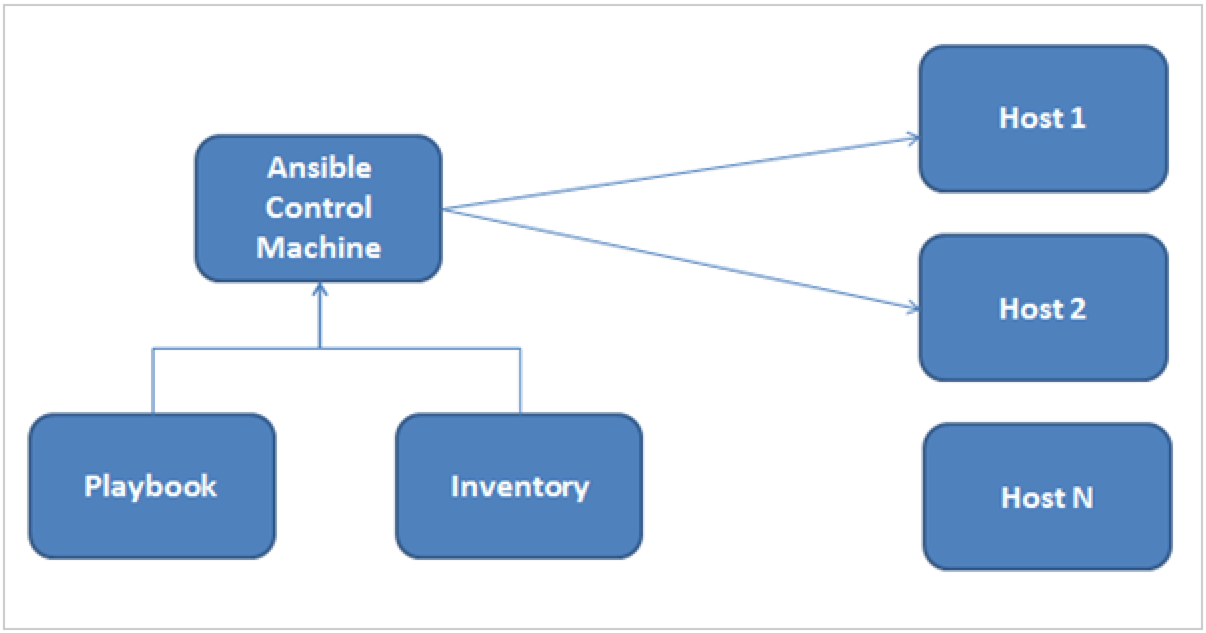

ansible@ip-172-31-20-105:/opt/niranjan

```
[ansible@ip-172-31-20-105 niranjan]$ ansible webservers -m ping
172.31.28.47 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[ansible@ip-172-31-20-105 niranjan]$
```

To list the hosts in the inventory file, you can run the below command

```
$ ansible webservers --list-hosts
```

Ansible Usage



Ansible consist of 3 main components

- Control Machine
- Inventory
- Playbook

The control machine manages the execution of the Playbook. It can be installed on your laptop or on any machine on the internet.

The Inventory file provides a complete list of all the target machines on which various modules are run by doing an ssh connection and install the necessary software's.

The playbook consists of steps that the control mechanism will perform on the servers defined in the inventory file.

Very important to understand here is that Ansible interacts with all the servers defined in the inventory through the SSH protocol which is a secure method of remote login. Every operation is done and file transfer is encrypted.

So as you would have seen in the previous section Ansible does not use any kind of database for installation and is very easy to install, we will now proceed with the actual usage of Ansible starting with Modules which is the main building block.

Ansible Modules

Modules are the main building blocks of Ansible and are basically reusable scripts that are used by Ansible playbooks. Ansible comes with a number of reusable modules. These include functionality for controlling services, software package installation, working with files and directories etc.

The syntax is as follows while running the ad-hoc commands which help in running single or simple tasks just once and which need not be run later. For **E.g.** just installing Tomcat on all servers.

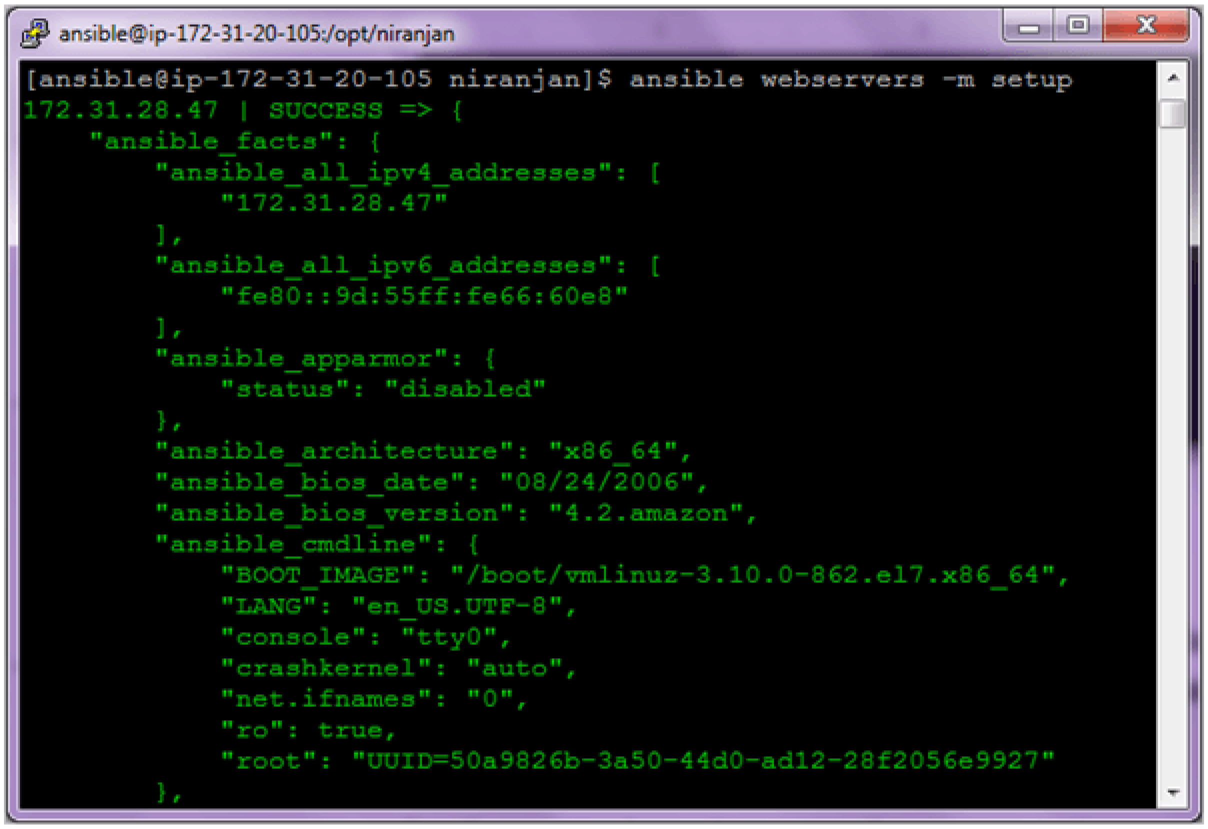
```
ansible hostORgroup -m module_name -a "arguments" -u username --become
```

Let's have a look at some of the most popular Ansible modules and their usage through the ad-hoc commands and later on in the playbook.

#1) Setup Module

To get information about the network or hardware or OS version or memory related information the setup module will help to gather the same about the target machines. On the control, the machine runs the below command.

```
$ ansible webservers -m setup
```



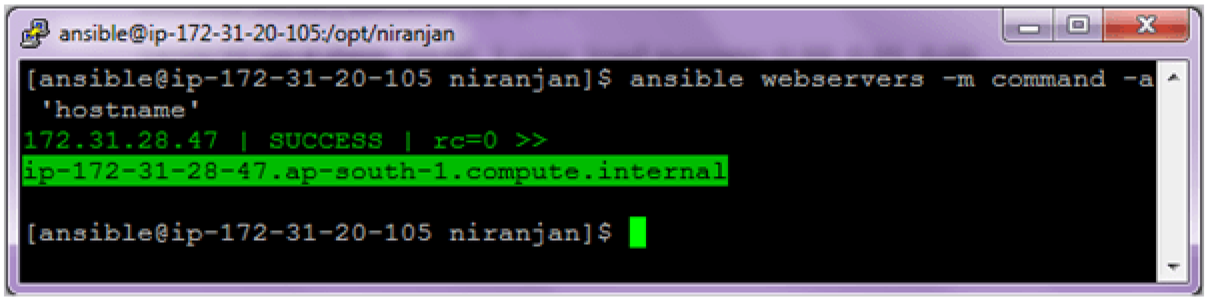
```
[ansible@ip-172-31-20-105 niranjan]$ ansible webservers -m setup
172.31.28.47 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "172.31.28.47"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::9d:55ff:fe66:60e8"
        ],
        "ansible_apparmor": {
            "status": "disabled"
        },
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "08/24/2006",
        "ansible_bios_version": "4.2.amazon",
        "ansible_cmdline": {
            "BOOT_IMAGE": "/boot/vmlinuz-3.10.0-862.el7.x86_64",
            "LANG": "en_US.UTF-8",
            "console": "tty0",
            "crashkernel": "auto",
            "net.ifnames": "0",
            "ro": true,
            "root": "UUID=50a9826b-3a50-44d0-ad12-28f2056e9927"
        },
```

#2) Command Module

The command module simply executes a specific command on the target machine and gives the output.

Some of the Examples are given below

```
$ ansible webservers –m command - an ‘uptime’
$ ansible webservers –m command –a ‘hostname’
```



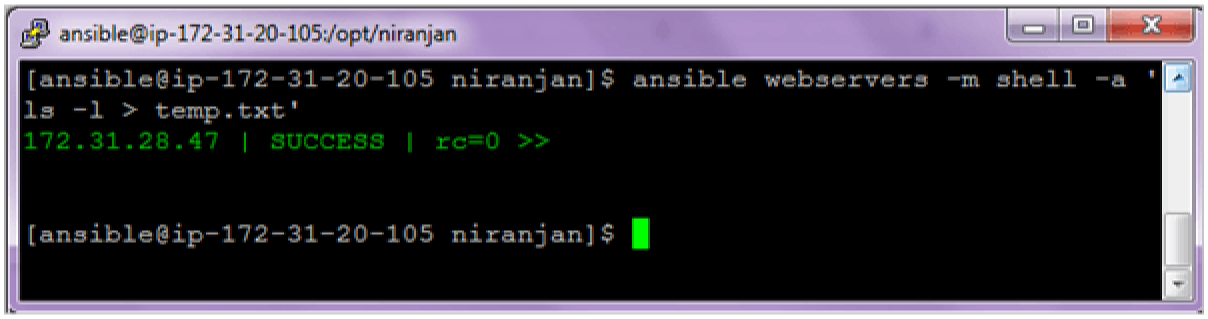
```
[ansible@ip-172-31-20-105 niranjan]$ ansible webservers -m command -a
 'hostname'
172.31.28.47 | SUCCESS | rc=0 >>
ip-172-31-28-47.ap-south-1.compute.internal

[ansible@ip-172-31-20-105 niranjan]$
```

#3) Shell Module

To execute any command in the shell of your choice you can use the Shell module. The shell module commands are run in /bin/sh shell and you can make use of the operators like '>' or '|' (pipe symbol or even environment variables.

So primarily the difference between the Shell and Command module is that if you actually do not need to use the operators like the ones mentioned then you could use the command module.

```
$ ansible webservers -m shell -a 'ls -l > temp.txt'
```



```
[ansible@ip-172-31-20-105 niranjan]$ ansible webservers -m shell -a '
ls -l > temp.txt'
172.31.28.47 | SUCCESS | rc=0 >>


[ansible@ip-172-31-20-105 niranjan]$
```

On the machines under webservers group check for the file created and run the command to view the text file.

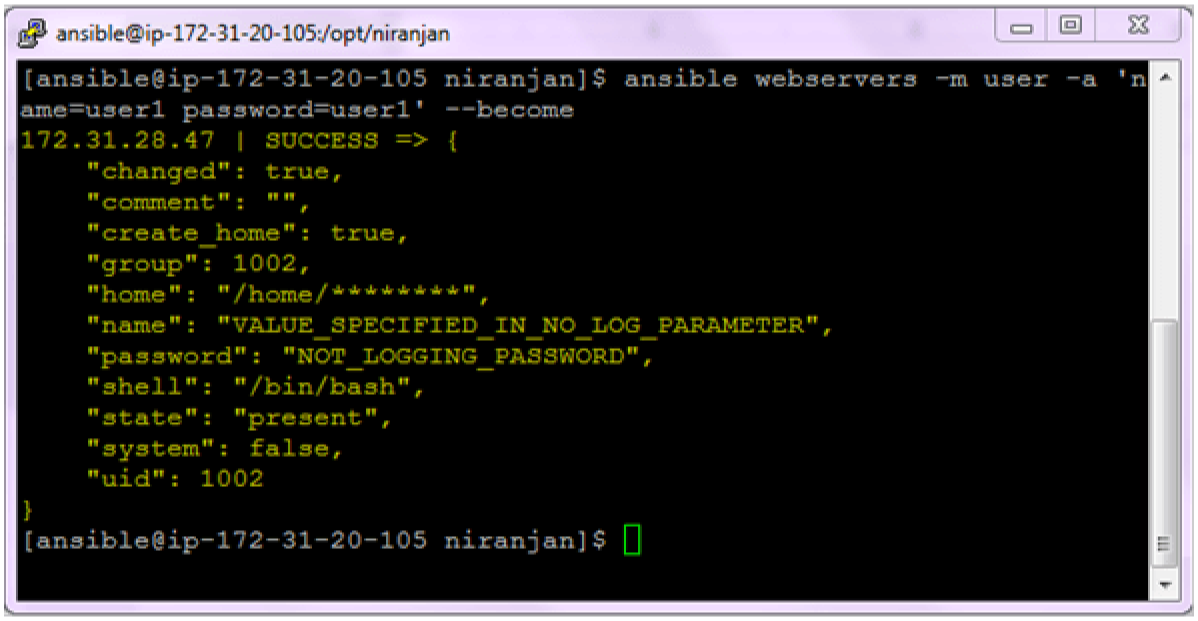
```
$ ansible webservers –m command -a ‘cat temp.txt’
```

#4) User Module

Using this module one can create or delete users.

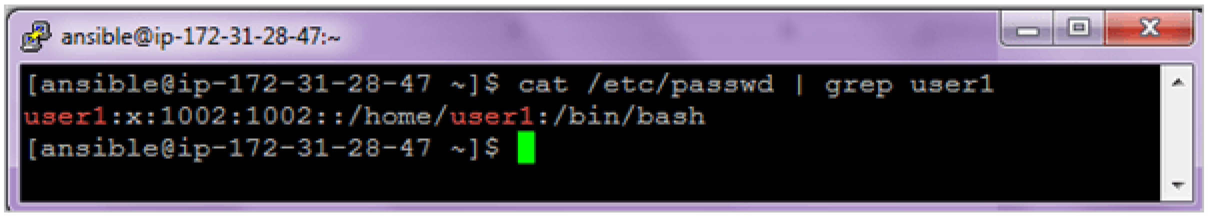
To add user

```
$ ansible webservers -m user -a 'name=user1 password=user1' --become
```



ansible@ip-172-31-20-105:/opt/niranjan

```
[ansible@ip-172-31-20-105 niranjan]$ ansible webservers -m user -a 'n
ame=user1 password=user1' --become
172.31.28.47 | SUCCESS => {
    "changed": true,
    "comment": "",
    "create_home": true,
    "group": 1002,
    "home": "/home/********",
    "name": "VALUE_SPECIFIED_IN_NO_LOG_PARAMETER",
    "password": "NOT_LOGGING_PASSWORD",
    "shell": "/bin/bash",
    "state": "present",
    "system": false,
    "uid": 1002
}
[ansible@ip-172-31-20-105 niranjan]$
```

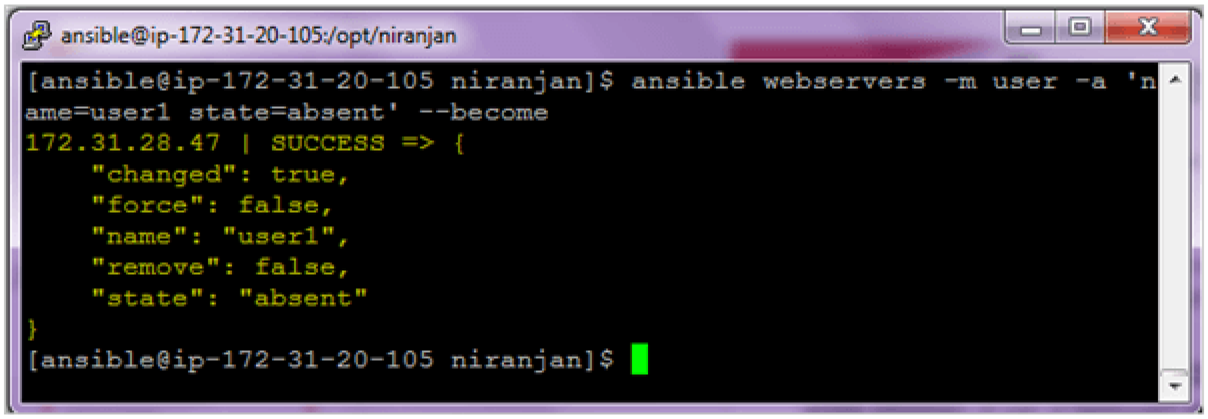


ansible@ip-172-31-28-47:~

```
[ansible@ip-172-31-28-47 ~]$ cat /etc/passwd | grep user1
user1:x:1002:1002::/home/user1:/bin/bash
[ansible@ip-172-31-28-47 ~]$
```

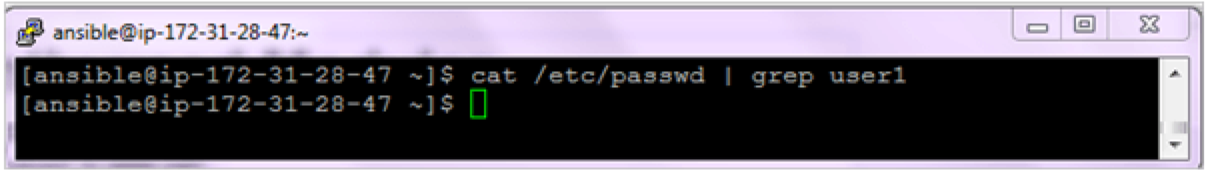
To delete user

```
$ ansible webservers -m user -a 'name=user1 state=absent' --become
```



ansible@ip-172-31-20-105:/opt/niranjan

```
[ansible@ip-172-31-20-105 niranjan]$ ansible webservers -m user -a 'n
ame=user1 state=absent' --become
172.31.28.47 | SUCCESS => {
    "changed": true,
    "force": false,
    "name": "user1",
    "remove": false,
    "state": "absent"
}
[ansible@ip-172-31-20-105 niranjan]$
```



ansible@ip-172-31-28-47:~

```
[ansible@ip-172-31-28-47 ~]$ cat /etc/passwd | grep user1
[ansible@ip-172-31-28-47 ~]$
```

Options:

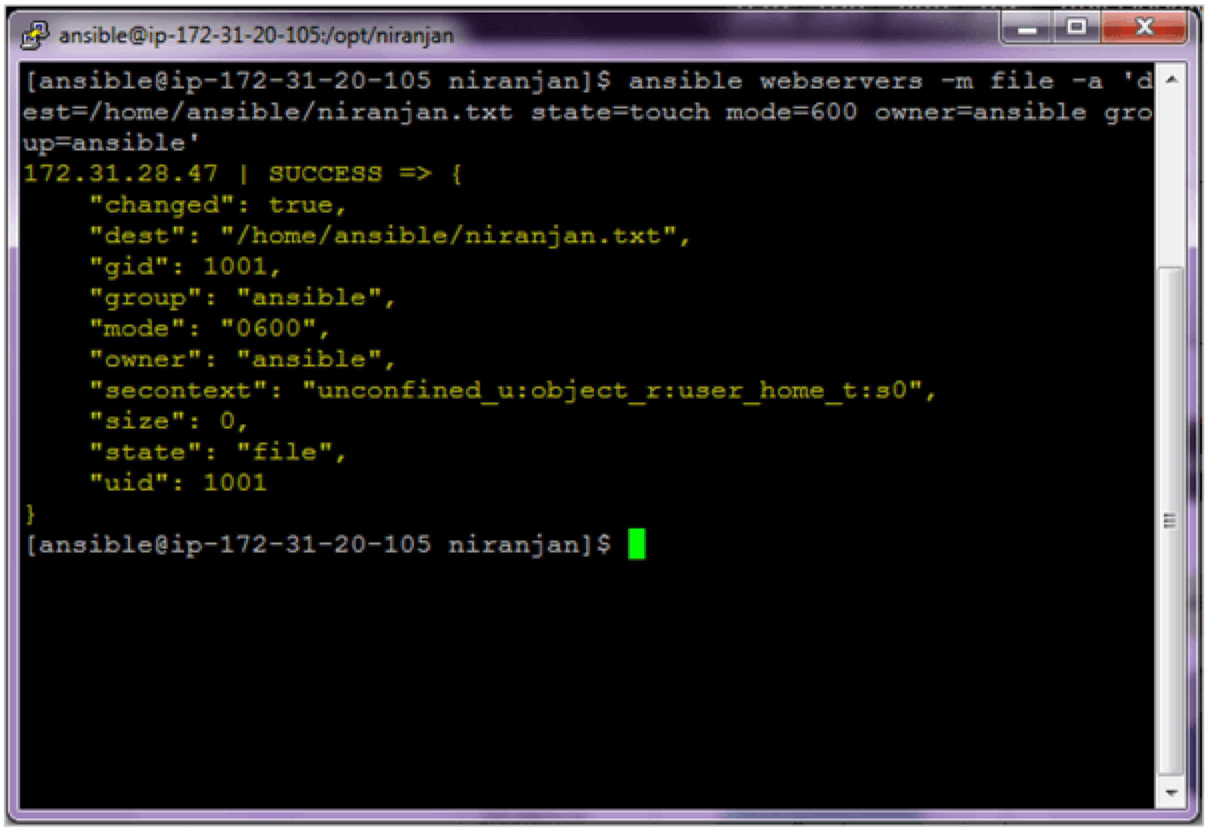
- **become** – Privilege to the superuser to run the command
- **state=absent** to delete the user

#5) File Module

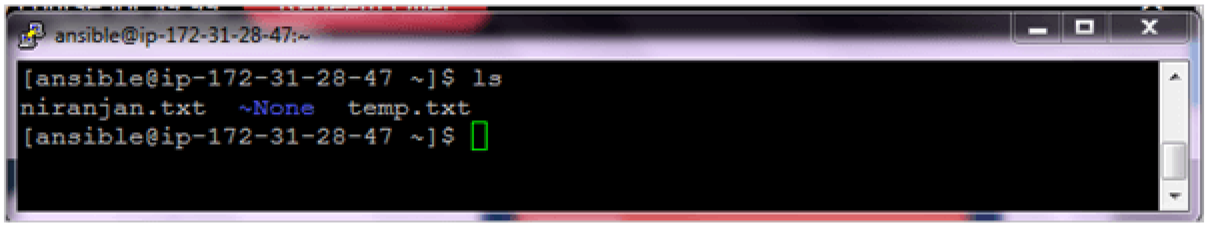
This module is used to create files, directories, set, or change file permissions and ownership etc

Example 1: Create a file

`$ ansible webservers -m file -a 'dest=/home/ansible/niranjan.txt state=touch mode=600 owner=ansible group=ansible'`



```
[ansible@ip-172-31-20-105 niranjan]$ ansible webservers -m file -a 'd
est=/home/ansible/niranjan.txt state=touch mode=600 owner=ansible gro
up=ansible'
172.31.28.47 | SUCCESS => {
    "changed": true,
    "dest": "/home/ansible/niranjan.txt",
    "gid": 1001,
    "group": "ansible",
    "mode": "0600",
    "owner": "ansible",
    "secontext": "unconfined_u:object_r:user_home_t:s0",
    "size": 0,
    "state": "file",
    "uid": 1001
}
[ansible@ip-172-31-20-105 niranjan]$
```



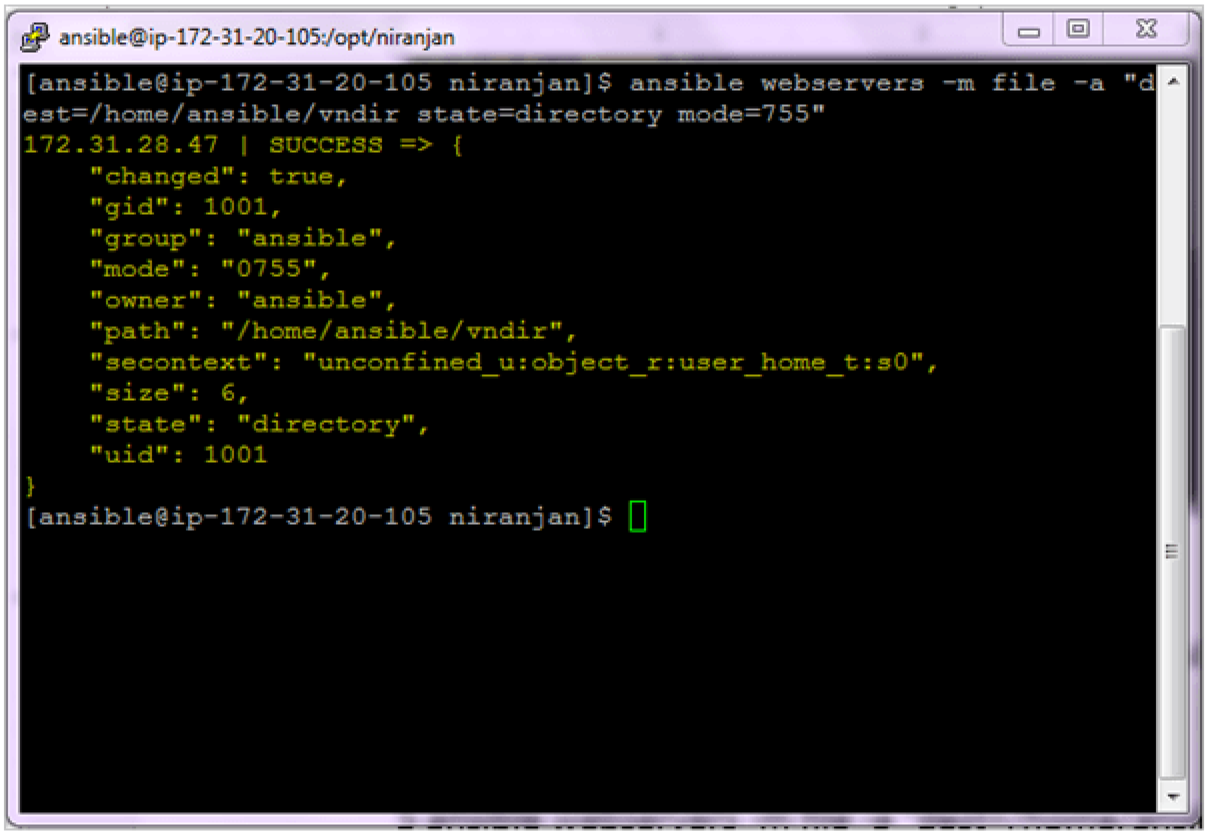
```
[ansible@ip-172-31-28-47 ~]$ ls
niranjan.txt  ~None  temp.txt
[ansible@ip-172-31-28-47 ~]$
```

Example 2: Create a directory

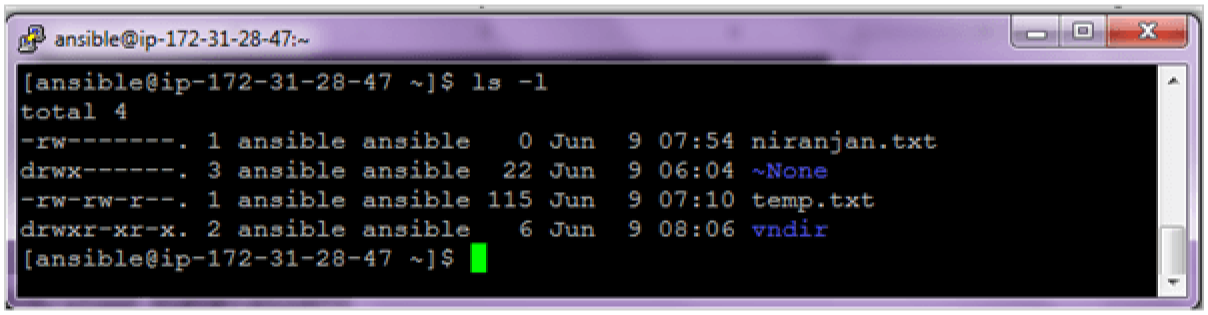
To create a directory using the file module, you need to set two parameters.

- Path(alias – name, dest) – This is the absolute path of the directory to be created.
- State – You should enter the value as 'directory.' By default, the value is 'file'.

`$ ansible webservers -m file -a "dest=/home/ansible/vndir state=directory mode=755"`



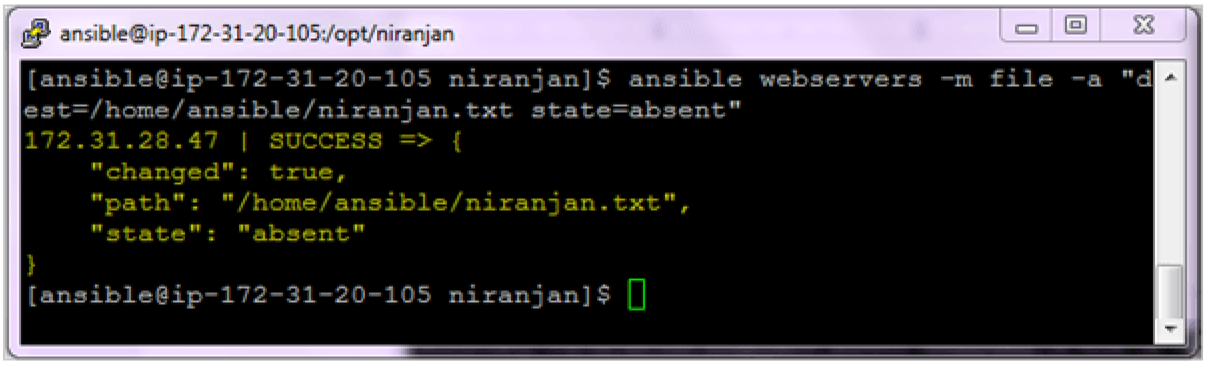
```
[ansible@ip-172-31-20-105 niranjan]$ ansible webservers -m file -a "d
est=/home/ansible/vndir state=directory mode=755"
172.31.28.47 | SUCCESS => {
    "changed": true,
    "gid": 1001,
    "group": "ansible",
    "mode": "0755",
    "owner": "ansible",
    "path": "/home/ansible/vndir",
    "secontext": "unconfined_u:object_r:user_home_t:s0",
    "size": 6,
    "state": "directory",
    "uid": 1001
}
[ansible@ip-172-31-20-105 niranjan]$
```



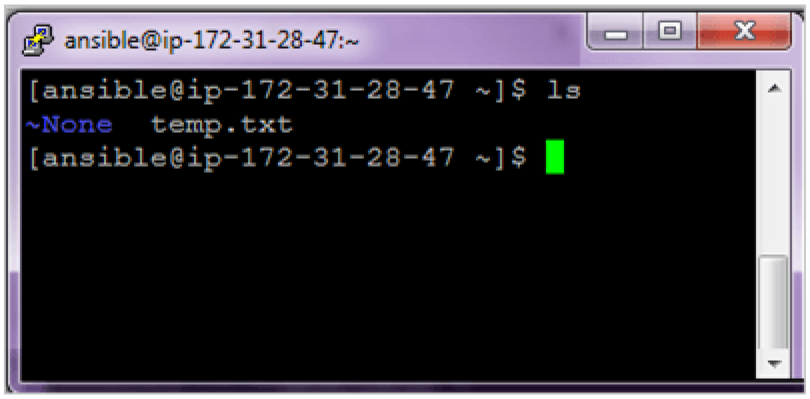
```
[ansible@ip-172-31-28-47 ~]$ ls -l
total 4
-rw-------. 1 ansible ansible   0 Jun  9 07:54 niranjan.txt
drwx------. 3 ansible ansible  22 Jun  9 06:04 ~None
-rw-rw-r--. 1 ansible ansible 115 Jun  9 07:10 temp.txt
drwxr-xr-x. 2 ansible ansible   6 Jun  9 08:06 vndir
[ansible@ip-172-31-28-47 ~]$
```

Example 3: Delete a file

```
$ ansible webservers -m file -a "dest=/home/ansible/niranjan.txt state=absent"
```



```
[ansible@ip-172-31-20-105 niranjan]$ ansible webservers -m file -a "d
est=/home/ansible/niranjan.txt state=absent"
172.31.28.47 | SUCCESS => {
    "changed": true,
    "path": "/home/ansible/niranjan.txt",
    "state": "absent"
}
[ansible@ip-172-31-20-105 niranjan]$
```

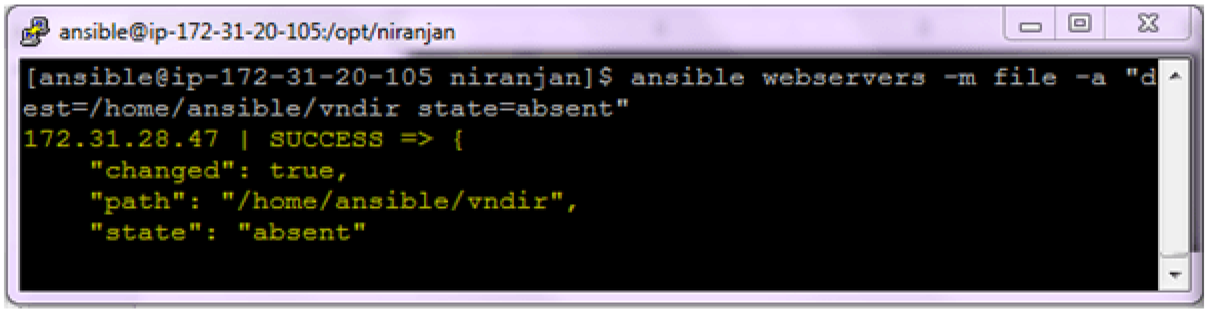


```
[ansible@ip-172-31-28-47 ~]$ ls
~None   temp.txt
[ansible@ip-172-31-28-47 ~]$
```

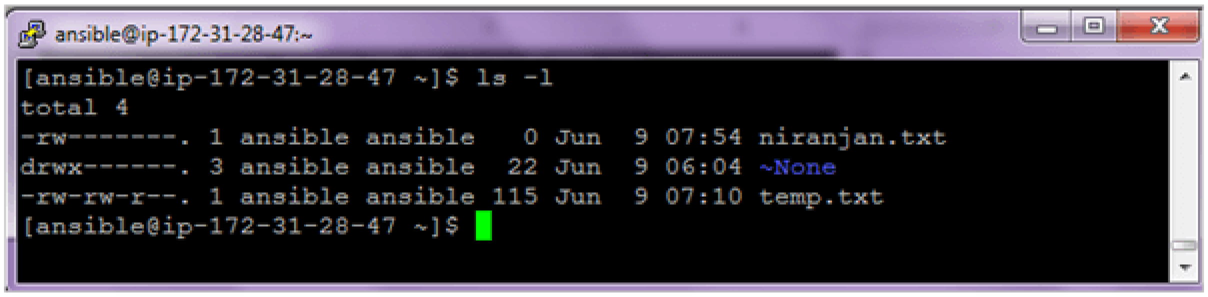
Example 4: Delete a directory

You can delete a directory by setting the state parameter value to **absent**. The directory and all its contents will be deleted.

```
$ ansible webservers -m file -a "dest=/home/ansible/vndir state=absent"
```



```
[ansible@ip-172-31-20-105 niranjan]$ ansible webservers -m file -a "d
est=/home/ansible/vndir state=absent"
172.31.28.47 | SUCCESS => {
    "changed": true,
    "path": "/home/ansible/vndir",
    "state": "absent"
```

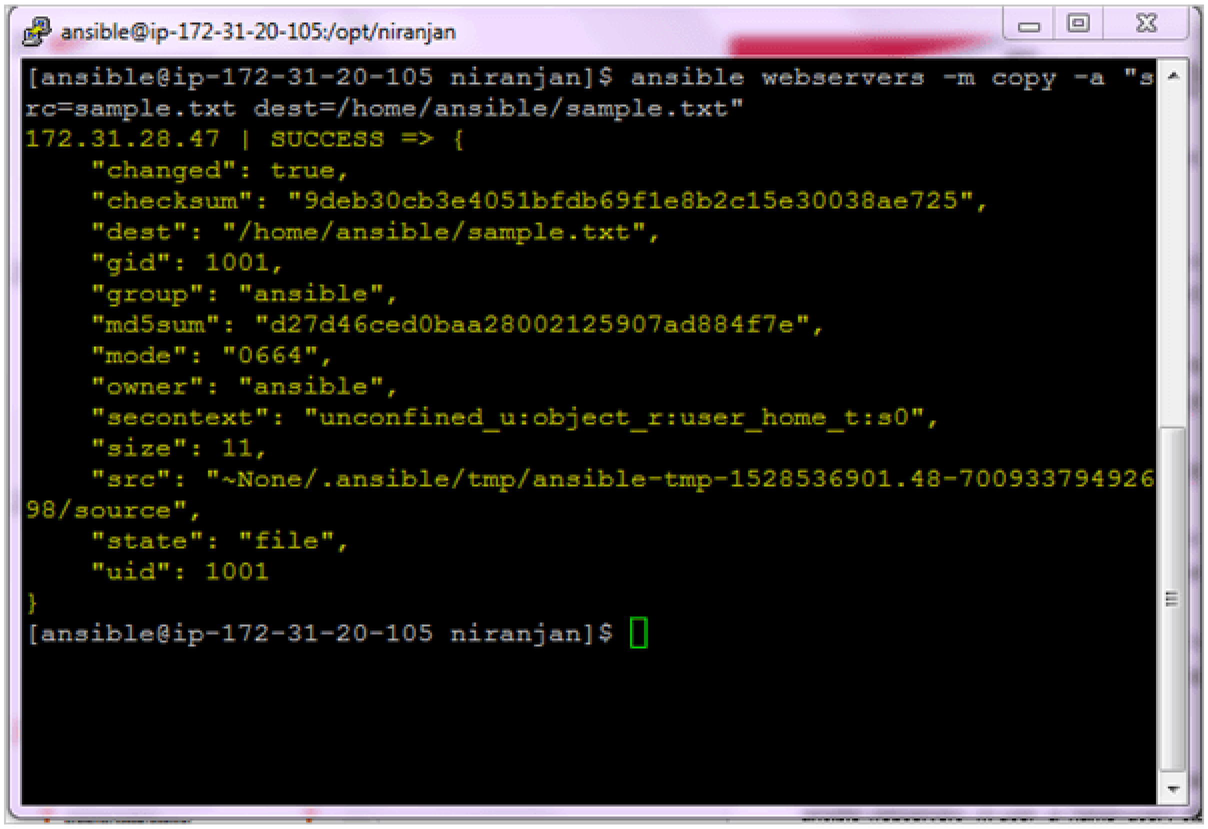


```
[ansible@ip-172-31-28-47 ~]$ ls -l
total 4
-rw-------. 1 ansible ansible   0 Jun  9 07:54 niranjan.txt
drwx------. 3 ansible ansible  22 Jun  9 06:04 ~None
-rw-rw-r--. 1 ansible ansible 115 Jun  9 07:10 temp.txt
[ansible@ip-172-31-28-47 ~]$
```

#6) Copy Module

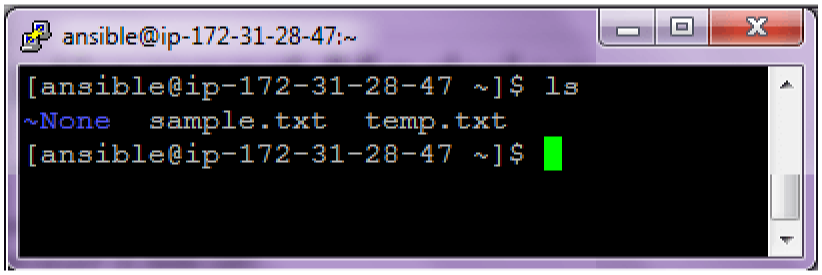
It is used for copying files to multiple target machines.

```
$ ansible webservers -m copy -a "src=sample.txt dest=/home/ansible/sample.txt"
```



ansible@ip-172-31-20-105:/opt/niranjan

```
[ansible@ip-172-31-20-105 niranjan]$ ansible webservers -m copy -a "s
rc=sample.txt dest=/home/ansible/sample.txt"
172.31.28.47 | SUCCESS => {
    "changed": true,
    "checksum": "9deb30cb3e4051bfdb69f1e8b2c15e30038ae725",
    "dest": "/home/ansible/sample.txt",
    "gid": 1001,
    "group": "ansible",
    "md5sum": "d27d46ced0baa28002125907ad884f7e",
    "mode": "0664",
    "owner": "ansible",
    "secontext": "unconfined_u:object_r:user_home_t:s0",
    "size": 11,
    "src": "~None/.ansible/tmp/ansible-tmp-1528536901.48-700933794926
98/source",
    "state": "file",
    "uid": 1001
}
[ansible@ip-172-31-20-105 niranjan]$
```



ansible@ip-172-31-28-47:~

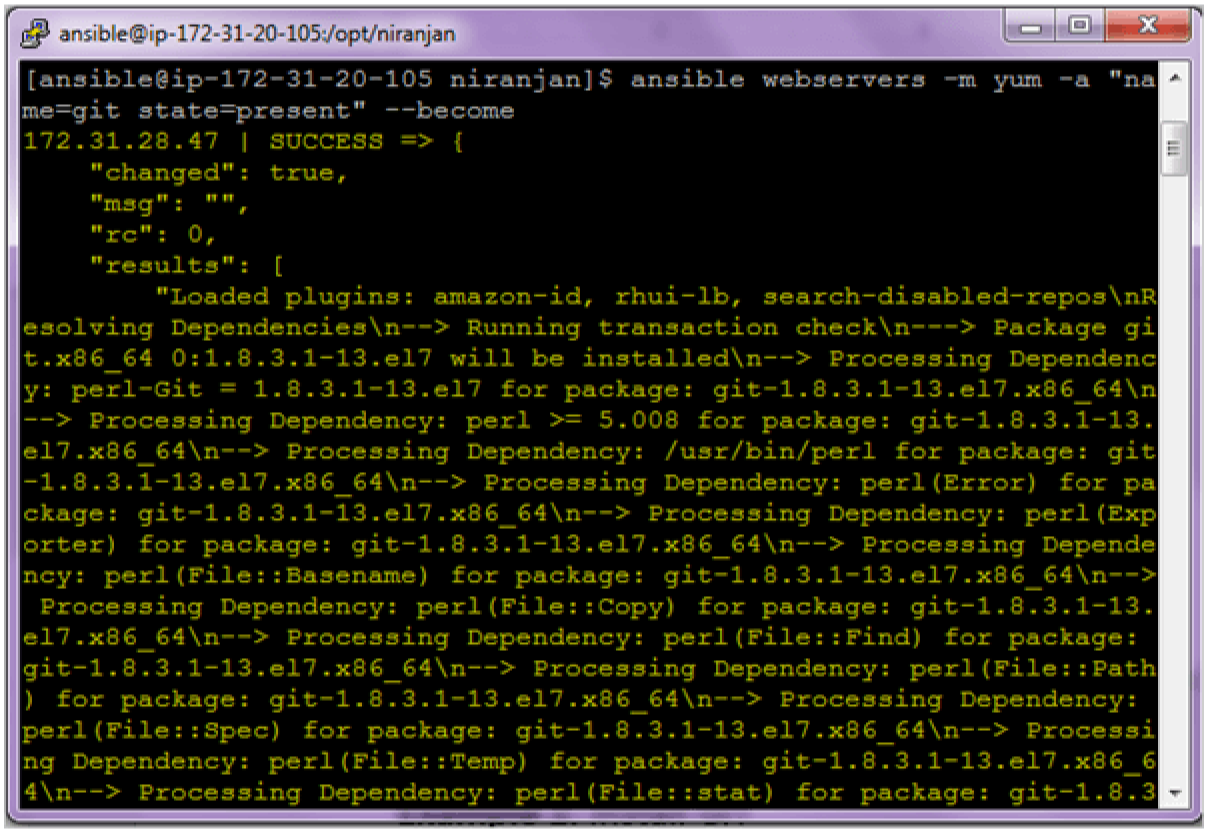
```
[ansible@ip-172-31-28-47 ~]$ ls
~None  sample.txt  temp.txt
[ansible@ip-172-31-28-47 ~]$
```

#7) Managing Software Packages

If you need to install software packages through ‘yum’ or ‘apt’ you can use the below commands.

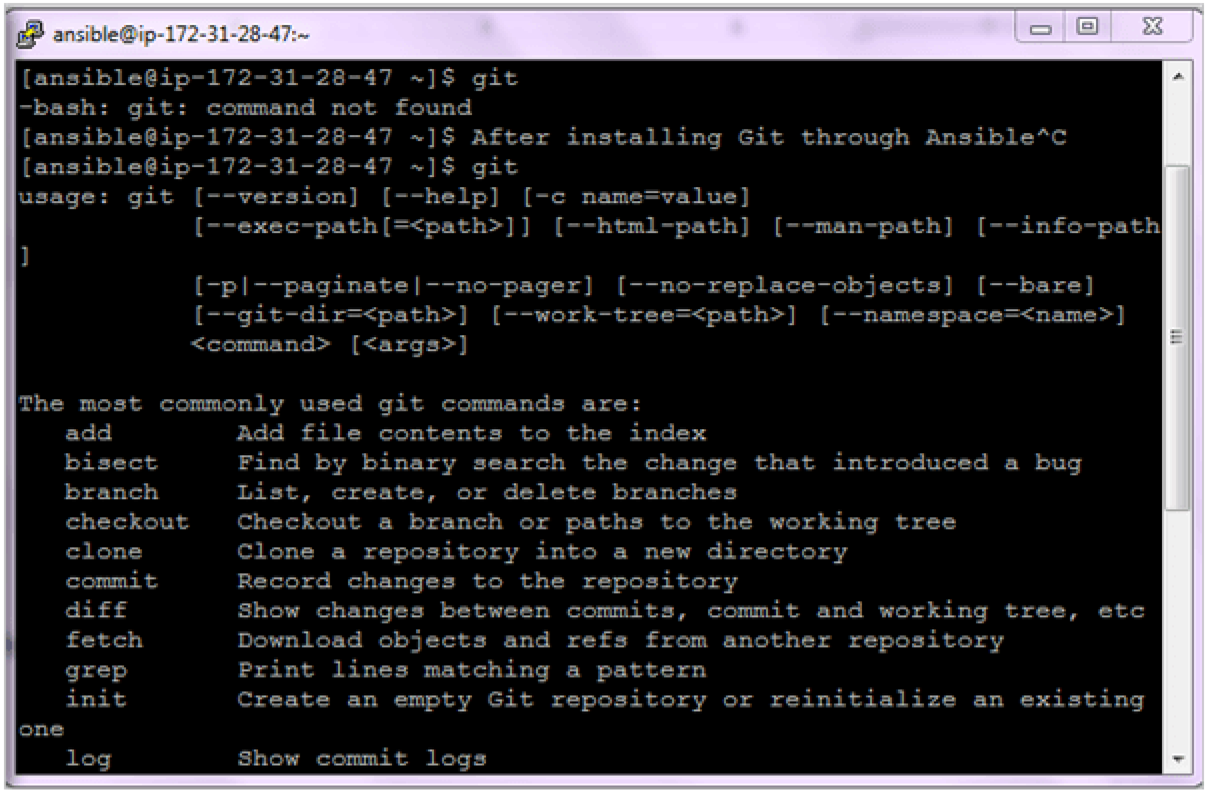
Example 1: Install GIT

```
$ ansible webservers –m yum -a “name=git state=present” --become
```



ansible@ip-172-31-20-105:/opt/niranjan

```
[ansible@ip-172-31-20-105 niranjan]$ ansible webservers -m yum -a "na
me=git state=present" --become
172.31.28.47 | SUCCESS => {
    "changed": true,
    "msg": "",
    "rc": 0,
    "results": [
        "Loaded plugins: amazon-id, rhui-lb, search-disabled-repos\nR
esolving Dependencies\n--> Running transaction check\n---> Package gi
t.x86_64 0:1.8.3.1-13.el7 will be installed\n--> Processing Dependenc
y: perl-Git = 1.8.3.1-13.el7 for package: git-1.8.3.1-13.el7.x86_64\n
--> Processing Dependency: perl >= 5.008 for package: git-1.8.3.1-13.
el7.x86_64\n--> Processing Dependency: /usr/bin/perl for package: git
-1.8.3.1-13.el7.x86_64\n--> Processing Dependency: perl(Error) for pa
ckage: git-1.8.3.1-13.el7.x86_64\n--> Processing Dependency: perl(Exp
orter) for package: git-1.8.3.1-13.el7.x86_64\n--> Processing Depende
ncy: perl(File::Basename) for package: git-1.8.3.1-13.el7.x86_64\n-->
 Processing Dependency: perl(File::Copy) for package: git-1.8.3.1-13.
el7.x86_64\n--> Processing Dependency: perl(File::Find) for package:
git-1.8.3.1-13.el7.x86_64\n--> Processing Dependency: perl(File::Path
) for package: git-1.8.3.1-13.el7.x86_64\n--> Processing Dependency:
perl(File::Spec) for package: git-1.8.3.1-13.el7.x86_64\n--> Processi
ng Dependency: perl(File::Temp) for package: git-1.8.3.1-13.el7.x86_6
4\n--> Processing Dependency: perl(File::stat) for package: git-1.8.3
```



ansible@ip-172-31-28-47:~

```
[ansible@ip-172-31-28-47 ~]$ git
-bash: git: command not found
[ansible@ip-172-31-28-47 ~]$ After installing Git through Ansible^C
[ansible@ip-172-31-28-47 ~]$ git
usage: git [--version] [--help] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path
]
           [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

The most commonly used git commands are:
   add        Add file contents to the index
   bisect     Find by binary search the change that introduced a bug
   branch     List, create, or delete branches
   checkout   Checkout a branch or paths to the working tree
   clone      Clone a repository into a new directory
   commit     Record changes to the repository
   diff       Show changes between commits, commit and working tree, etc
   fetch      Download objects and refs from another repository
   grep       Print lines matching a pattern
   init       Create an empty Git repository or reinitialize an existing
one
   log        Show commit logs
```

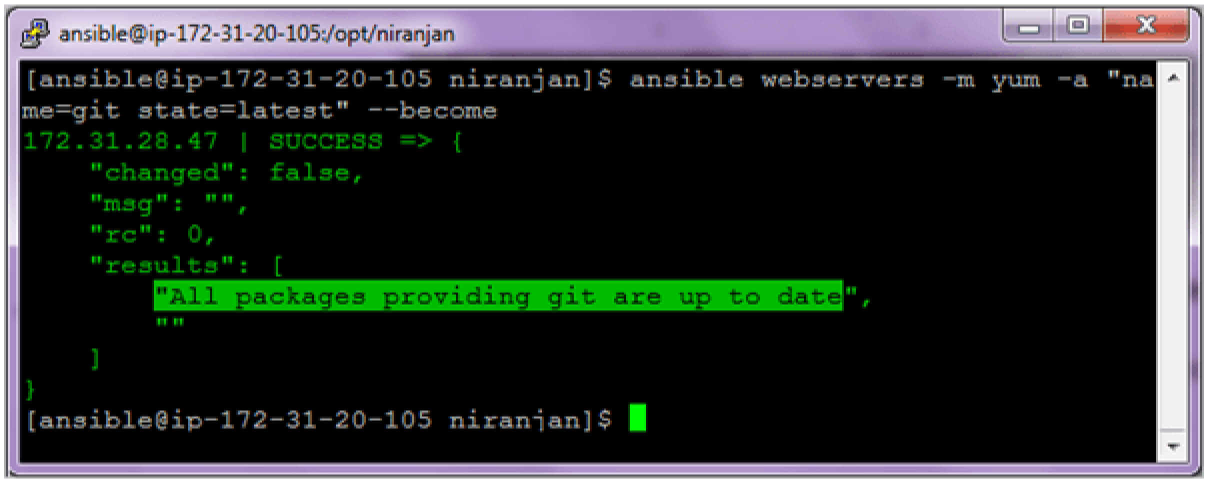
On the right-hand window, you can see if git is not installed it will give command not found and once installed it will show up the output.

In this command, **state=present** will check if the package is installed or not and if not installed it will install the latest version.

Example 2: Check if the package is installed & update it to the latest version.

```
$ ansible webservers -m yum -a “name=git state=latest”
```

In the above command, **state=latest** will update the package to the latest version only.



ansible@ip-172-31-20-105:/opt/niranjan

```
[ansible@ip-172-31-20-105 niranjan]$ ansible webservers -m yum -a "na
me=git state=latest" --become
172.31.28.47 | SUCCESS => {
    "changed": false,
    "msg": "",
    "rc": 0,
    "results": [
        "All packages providing git are up to date",
        ""
    ]
}
[ansible@ip-172-31-20-105 niranjan]$
```

Example 3: Install Apache Webserver

```
$ ansible webservers -m yum -a "name=httpd state=present" –become
```

Example 4: Check if Maven is installed or not.

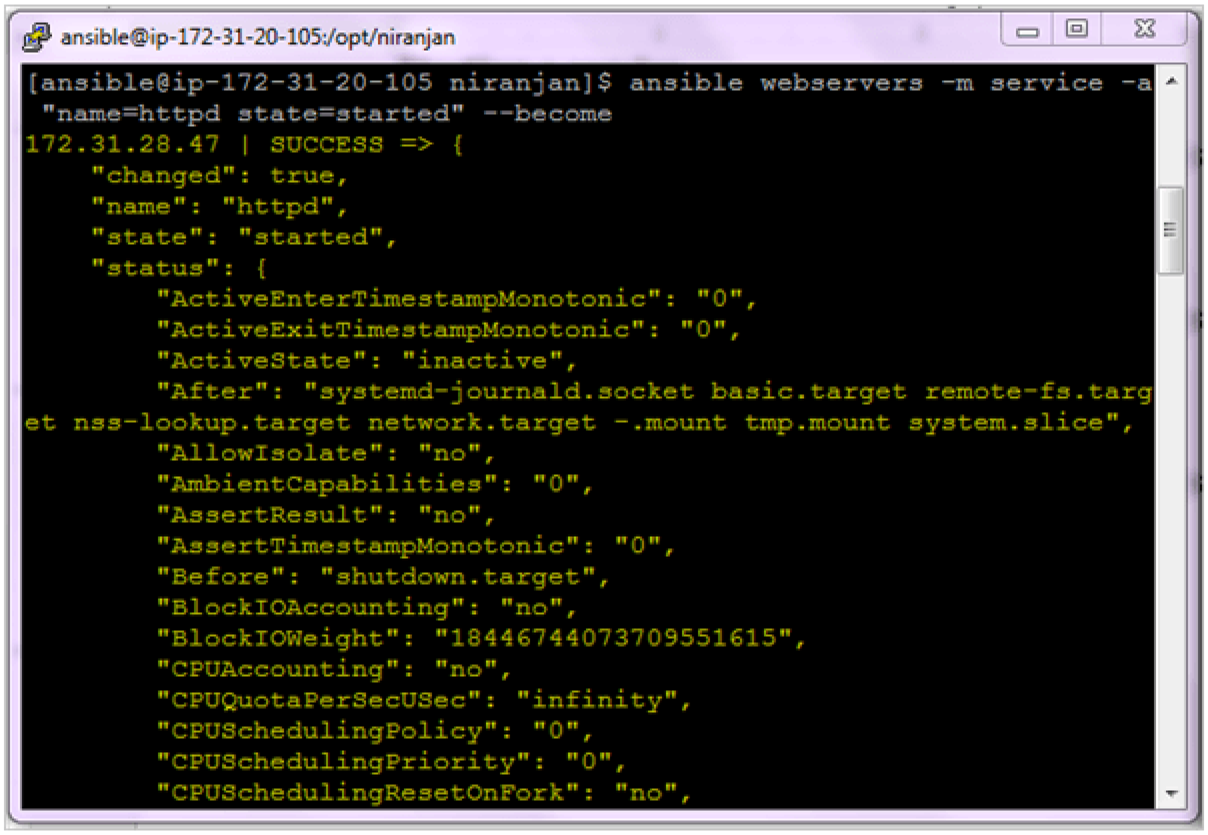
```
$ ansible webservers -m yum -a "name=maven state=absent" –become
```

#8) Managing Services Module

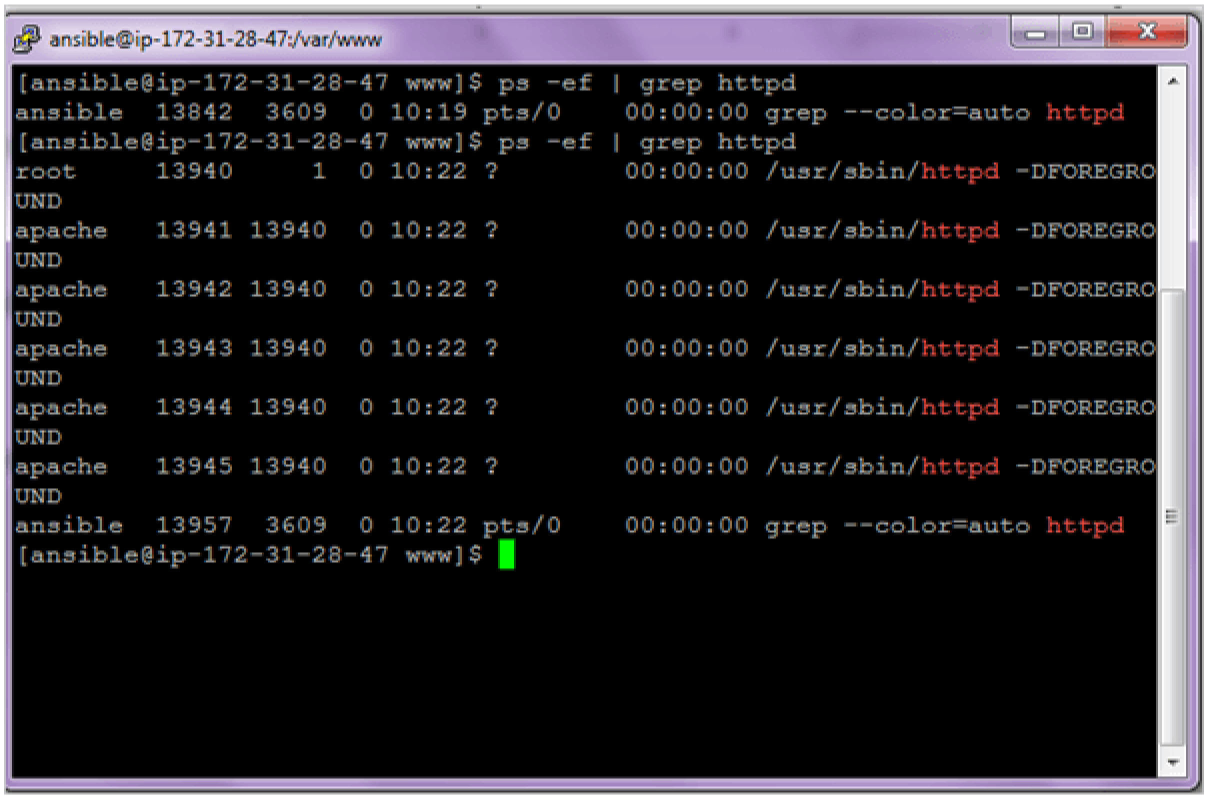
To manage services with ansible, we use a module **‘service’.**

Starting a service

```
$ ansible webservers -m service -a “name=httpd state=started” --become
```



```
[ansible@ip-172-31-20-105 niranjan]$ ansible webservers -m service -a
 "name=httpd state=started" --become
172.31.28.47 | SUCCESS => {
    "changed": true,
    "name": "httpd",
    "state": "started",
    "status": {
        "ActiveEnterTimestampMonotonic": "0",
        "ActiveExitTimestampMonotonic": "0",
        "ActiveState": "inactive",
        "After": "systemd-journald.socket basic.target remote-fs.targ
et nss-lookup.target network.target -.mount tmp.mount system.slice",
        "AllowIsolate": "no",
        "AmbientCapabilities": "0",
        "AssertResult": "no",
        "AssertTimestampMonotonic": "0",
        "Before": "shutdown.target",
        "BlockIOAccounting": "no",
        "BlockIOWeight": "18446744073709551615",
        "CPUAccounting": "no",
        "CPUQuotaPerSecUSec": "infinity",
        "CPUSchedulingPolicy": "0",
        "CPUSchedulingPriority": "0",
        "CPUSchedulingResetOnFork": "no",
```



```
[ansible@ip-172-31-28-47 www]$ ps -ef | grep httpd
ansible  13842  3609  0 10:19 pts/0    00:00:00 grep --color=auto httpd
[ansible@ip-172-31-28-47 www]$ ps -ef | grep httpd
root     13940     1  0 10:22 ?        00:00:00 /usr/sbin/httpd -DFOREGRO
UND
apache   13941 13940  0 10:22 ?        00:00:00 /usr/sbin/httpd -DFOREGRO
UND
apache   13942 13940  0 10:22 ?        00:00:00 /usr/sbin/httpd -DFOREGRO
UND
apache   13943 13940  0 10:22 ?        00:00:00 /usr/sbin/httpd -DFOREGRO
UND
apache   13944 13940  0 10:22 ?        00:00:00 /usr/sbin/httpd -DFOREGRO
UND
apache   13945 13940  0 10:22 ?        00:00:00 /usr/sbin/httpd -DFOREGRO
UND
ansible  13957  3609  0 10:22 pts/0    00:00:00 grep --color=auto httpd
[ansible@ip-172-31-28-47 www]$
```

Stopping a service

```
$ ansible webservers -m service -a “name=httpd state=stopped” --become
```

Restarting a service

```
$ ansible webservers -m service -a “name=httpd state=restarted --become
```

Click here to get the Complete Module list.

Summary

In this tutorial, I introduced you to the basic concepts and components of Ansible and we also have seen more about installation, configuration, and usage of Ansible with the help of Ansible modules which is the main component in Ansible execution.

In our upcoming tutorial, we will focus on the very important aspect of writing playbooks for task automation and Ansible vaults to keep sensitive data in encrypted files.

PREV Tutorial | NEXT Tutorial