Table of Contents

Ansible Setup

Launching Machines with Amazon Linux

- To Setup ansible on EC2 instances with Amazon Linux:
- Launch `3 EC2 instances` and tag one of the instance as control and managed nodes
- Change the hostname for ec2 instances on Amazon Linux AMI only, use below :
- On Control Node:

```
sudo hostnamectl set-hostname control-node.example.com
```

- On Managed Node:

```
sudo hostnamectl set-hostname managed-node-01.example.com
sudo hostnamectl set-hostname managed-node-02.example.com
```

- On Control Node:

```
[ec2-user@control-node ssm-user]$ ping managed-node-01.example.com
ping: managed-node-01.example.com: Name or service not known
```

--

- Edit the **/etc/hosts** file similar with below details:

```
172.31.26.166   control-node.example.com    control-node
172.31.27.151   managed-node-01.example.com managed-node-01
172.31.27.141   managed-node-02.example.com managed-node-02
```

--

Installing Ansible on Control Node

- Check if Python 3 is installed on Linux:
- Installing ansible only on one node i.e only `Control Node`:
- For amazon Linux 2 :

  ```
  sudo yum install python3
  sudo amazon-linux-extras install ansible2 -y
  ```

- For Redhat Family :

  ```
  sudo yum install -y ansible
  ```

- For debian Family

  ```
  sudo apt-get install -y ansible
  ```

--

Managing inventory file

- Create and change directory to the **/home/ec2-user/ansible-demo**

  ```
  mkdir ansible-demo && cd ansible-demo
  ```

- Ansible works against multiple managed nodes or “hosts” in your infrastructure at the same time, using a list or group of lists know as `inventory`.

> The `/etc/ansible/hosts` file is considered the system's default static inventory file.

--

- Create inventory file with name ***inventory*** in the same current working directory with below content `inventory`

```
[myhost]
localhost ansible_connection=local

[dev]
# Host Variable Definition
managed-node-01.example.com hostVariableName=hostVariableValue
ansible_connection=ssh

[qa]
managed-node-02.example.com

# Group Variables Definition
[myhost:vars]
username=myusername
password=mypassword

# [centos]
# managed-node-03.example.com     ansible_user=centos

#[qa:vars]
#ansible_ssh_private_key_file=/home/ec2-user/.ssh/aws.pem
```

- These variables are defined in inventory file.
    - `host variables` : You can easily assign a variable to a single host, then use it later in playbooks.
    - `group variables` : These are variable values that are to be shared for all hosts in a group
- To view all the list of hosts from inventory file

```
ansible -i inventory all --list-hosts
ansible -i inventory dev --list-hosts -v
ansible --version
```

--

Managing ansible.cfg file

- Ansible searches for **ansible.cfg** in these locations in order for preceedence of config file:

```


1. ANSIBLE_CONFIG => `environment variable where path of the ansible.cfg if set`
2. ansible.cfg => `in the current directory`
3. ~/.ansible.cfg => `in the Linux User's Home directory as a hidden file`
4. /etc/ansible/ansible.cfg => Global File
```

```
export ANSIBLE_CONFIG=""
unset ANSIBLE_CONFIG
```

- Using ***./ansible.cfg***
  - If an ***ansible.cfg*** file exists in the directory in which the ***ansible*** command is executed, it is used instead of the global file.
  - This allows administrators to create a directory structure where different environments or projects are stored in separate directories, with each directory containing a configuration file tailored with a unique set of settings.

--

- Using ***~/.ansible.cfg***

  - Ansible looks for a `~/.ansible.cfg` in the user's home directory.
  - If this file exists, this configuration is used instead of the `/etc/ansible/ansible.cfg` if there is no ***ansible.cfg*** file in the current working directory.

- Using ***/etc/ansible/ansible.cfg***

  - The ansible package provides a base configuration file located at ***/etc/ansible/ansible.cfg***. This file is used if no other configuration file is found.

--

- Lets create ***ansible.cfg*** file, it assumes that you can connect to the managed hosts as ***ec2-user*** using SSH key-based authentication, and that `ec2-user` can use sudo to run commands as root without entering a password:

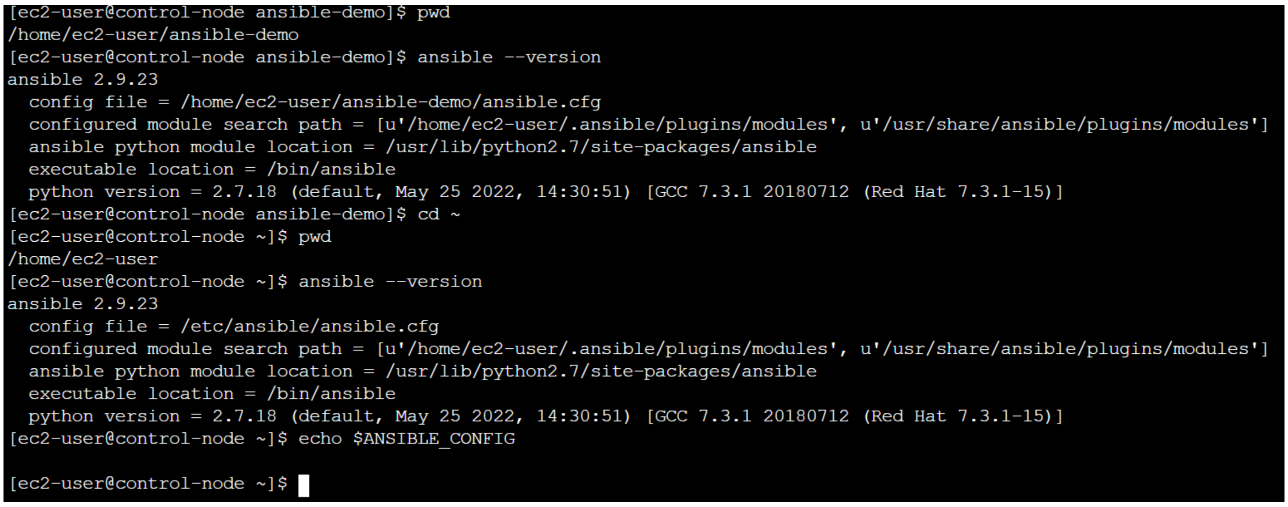
```
# Most of the settings in the configuration file are grouped here
[defaults]
# Python Executable binary path
interpreter_python=/usr/bin/python3
# specifies the path of your inventory file
inventory = ./inventory
# specifies the user who will connect to the managed hosts and run the playbooks
remote_user = ec2-user
ask_pass = false
# specifies the private key identity file to be used when connecting to remote
server.
private_key_file = /path/to/file.pem
host_key_checking = False

# Defines how operations that require escalated privileges are executed on managed
hosts.
[privilege_escalation]
# specify where to allow/disallow privilege escalation; default is False.
become = true
# specify the privilege escalation method; default is `sudo`.
become_method = sudo
```

```
# specify the user you become through privilege escalation; default is root.
become_user = root
# specify whether to ask or not ask for privilege escalation password; default is
False
become_ask_pass = false
```

--

- Check details of the config file using below command from different directories:

  ```
  ansible --version
  ```



```
[ec2-user@control-node ansible-demo]$ pwd
/home/ec2-user/ansible-demo
[ec2-user@control-node ansible-demo]$ ansible --version
ansible 2.9.23
  config file = /home/ec2-user/ansible-demo/ansible.cfg
  configured module search path = [u'/home/ec2-user/.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, May 25 2022, 14:30:51) [GCC 7.3.1 20180712 (Red Hat 7.3.1-15)]
[ec2-user@control-node ansible-demo]$ cd ~
[ec2-user@control-node ~]$ pwd
/home/ec2-user
[ec2-user@control-node ~]$ ansible --version
ansible 2.9.23
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/home/ec2-user/.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, May 25 2022, 14:30:51) [GCC 7.3.1 20180712 (Red Hat 7.3.1-15)]
[ec2-user@control-node ~]$ echo $ANSIBLE_CONFIG

[ec2-user@control-node ~]$
```

- Here, the config file options changes as per path from where ansible command is executed.
- Sections in the ***ansible.cfg*** file.

```
grep "^\[" /etc/ansible/ansible.cfg
```

--

Setup Passwordless ssh login from control node to managed node

- Passwordless ssh can be setup by using one of the below methods:

Using Existing AWS EC2 Key Pair

- As these EC2 instances are creating with same Key Pair, the `key.pem` used to connect to the instance can be copied on the Control Node Path under the same path as inventory file.
- Property `private_key_file=/path/to/file.pem` under ansible.cfg file can be used to specify the path to private key.
- Test the ssh connection from control node to managed nodes.

```
ssh -i /path/to/file.pem ec2-user@managed-node-01
```

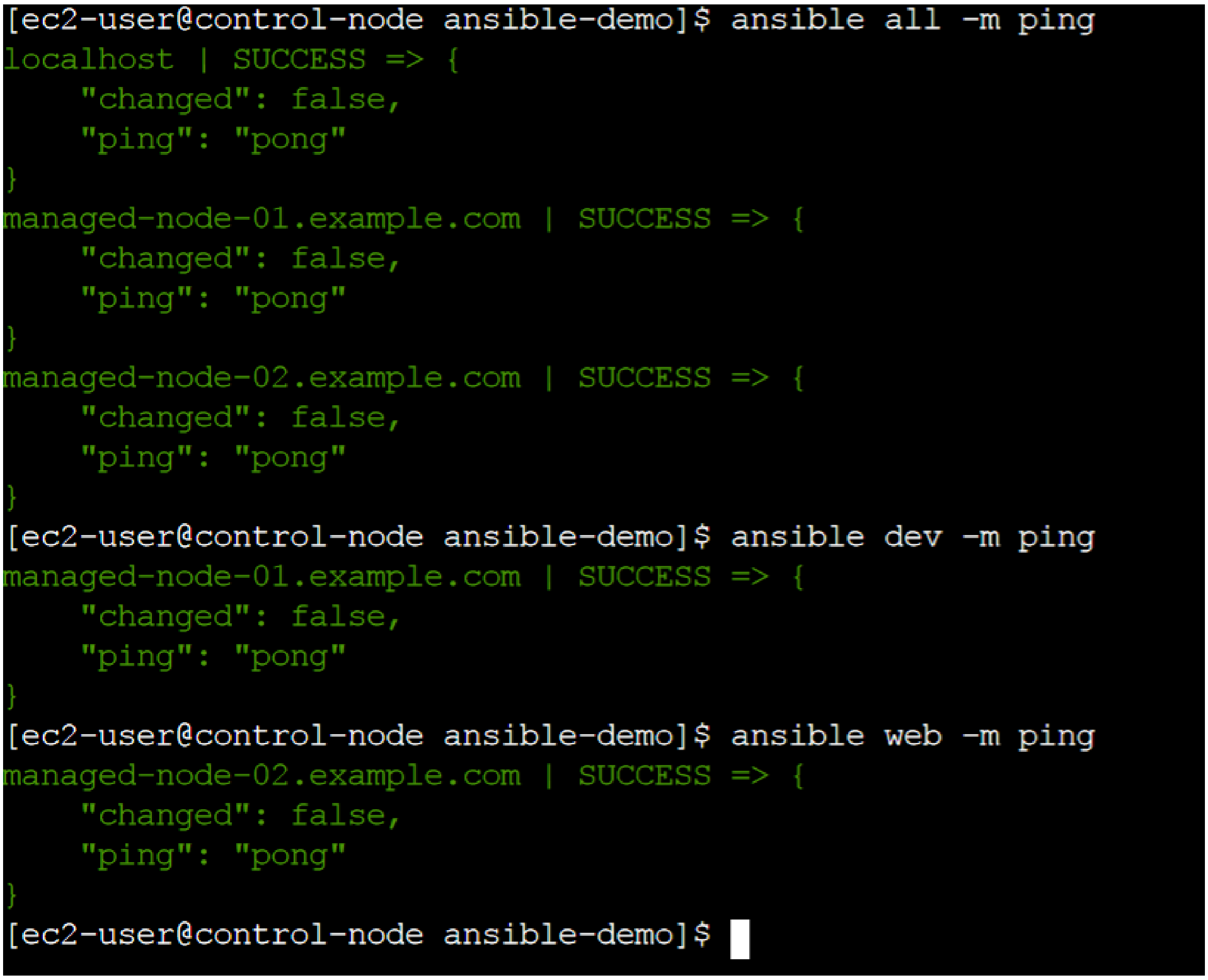
Ansible Modules

Running Ansible Ad Hoc Commands

- General Syntax of Ansible Ad-Hoc Command is:
- ***ansible host-pattern -m module [-a 'module arguments'] [-i inventory]***

```
ansible dev -m ping
ansible all -m ping
ansible qa -m ping
ansible dev123 -m ping
```

--



```
[ec2-user@control-node ansible-demo]$ ansible all -m ping
localhost | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
managed-node-01.example.com | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
managed-node-02.example.com | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[ec2-user@control-node ansible-demo]$ ansible dev -m ping
managed-node-01.example.com | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[ec2-user@control-node ansible-demo]$ ansible web -m ping
managed-node-02.example.com | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[ec2-user@control-node ansible-demo]$
```

- When you get SUCCESS response, it means the module has be successfully executed on remote host.

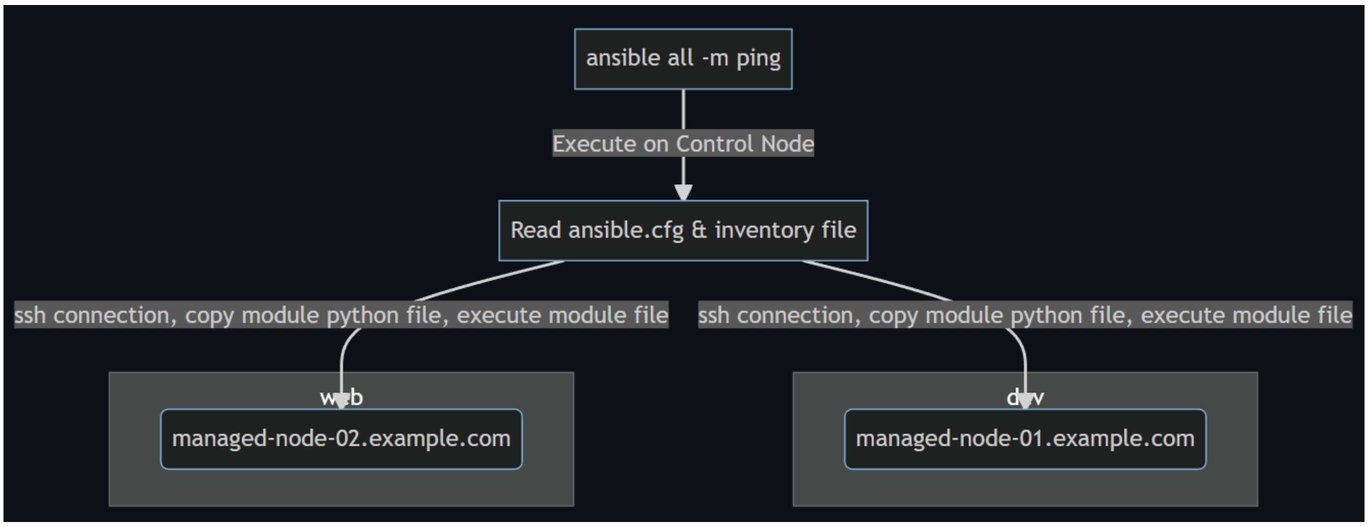
--

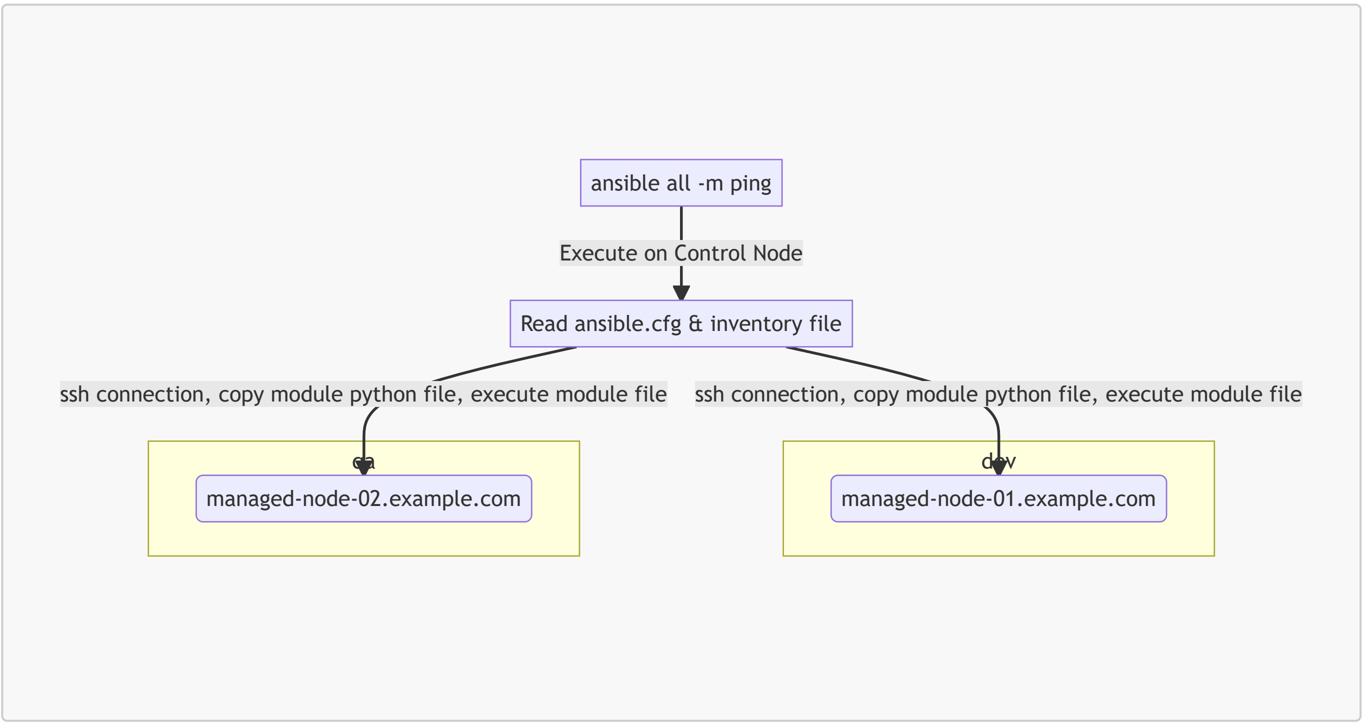
- To get more verbose mode of the execution use:

```
ansible dev -m ping -v
ansible dev -m ping -vv
ansible dev -m ping -vvv
```

- Module file under ansible packages
  - **/usr/lib/python2.7/site-packages/ansible/modules/system/ping.py**
- Execution information of **ansible ping module**

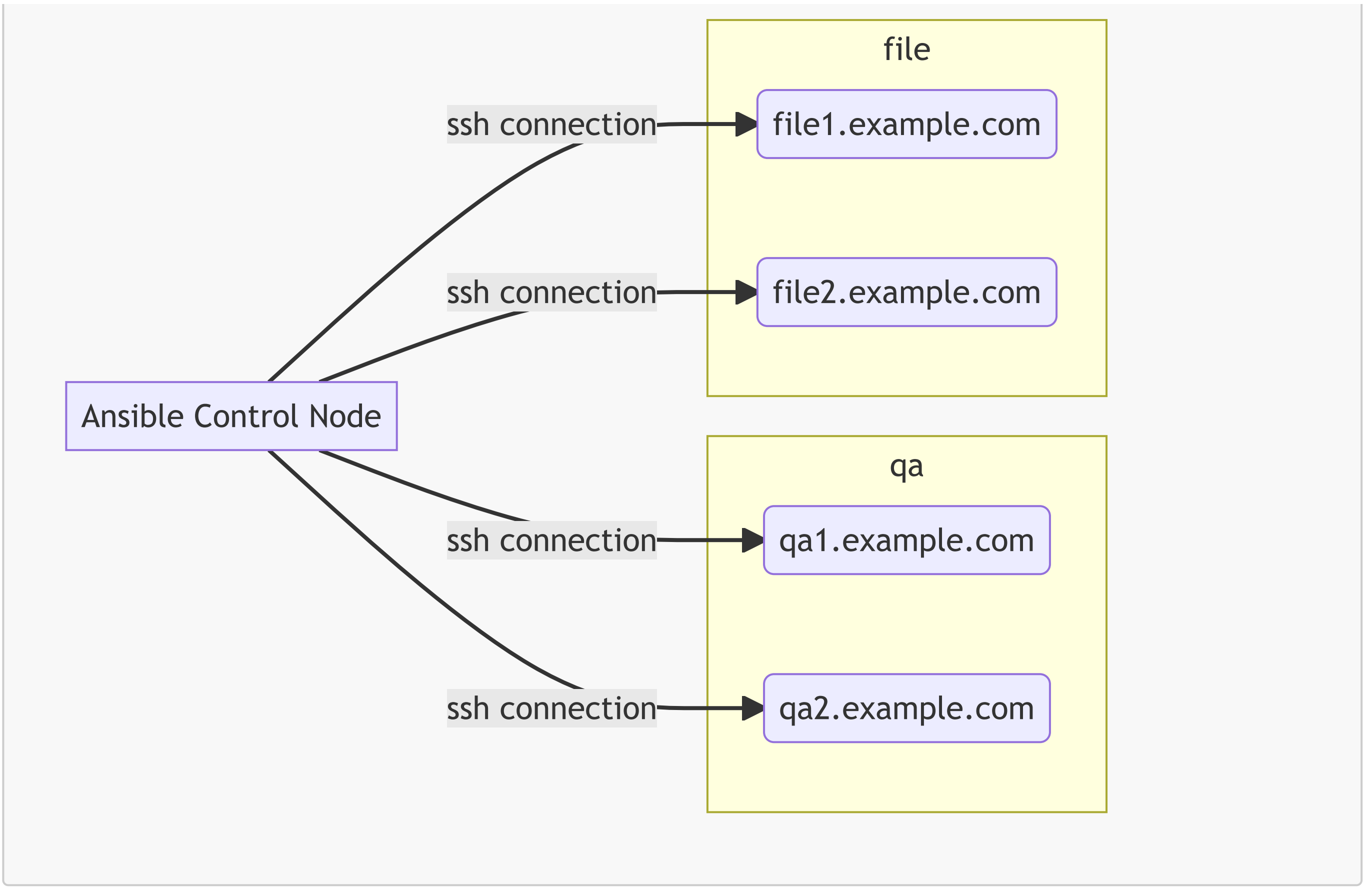
--


ansible all -m ping
Execute on Control Node
Read ansible.cfg & inventory file
ssh connection, copy module python file, execute module file
ssh connection, copy module python file, execute module file
web
managed-node-02.example.com
dev
managed-node-01.example.com


ansible all -m ping
Execute on Control Node
Read ansible.cfg & inventory file
ssh connection, copy module python file, execute module file
ssh connection, copy module python file, execute module file
qa
managed-node-02.example.com
dev
managed-node-01.example.com

--

- Sample scenario of ssh connection into remote managed nodes.



--

- In above verbose mode information, ansible uses SSH authentication in the background to connect to managed nodes, uses the module file (for e.g `ping.py`), copies this module file from control node over managed node in a temporary location and executes the file using python ***/usr/bin/python***.
- On execution of this module file, there is `success/failure` response that is returned.
- If there is ping issue for `localhost`, then try ssh localhost command or append public key to **authorized_keys** file on local server.
- To view all the modules that are present in the module: ***ansible-doc -l***
- To get the number of modules supported by Ansible: ***ansible-doc -l | wc -l***

```
ansible-doc -l
ansible-doc -l | grep -i 'aws'
ansible-doc -l | grep -i 'gcp'
ansible-doc -l | wc -l
```

--

user module

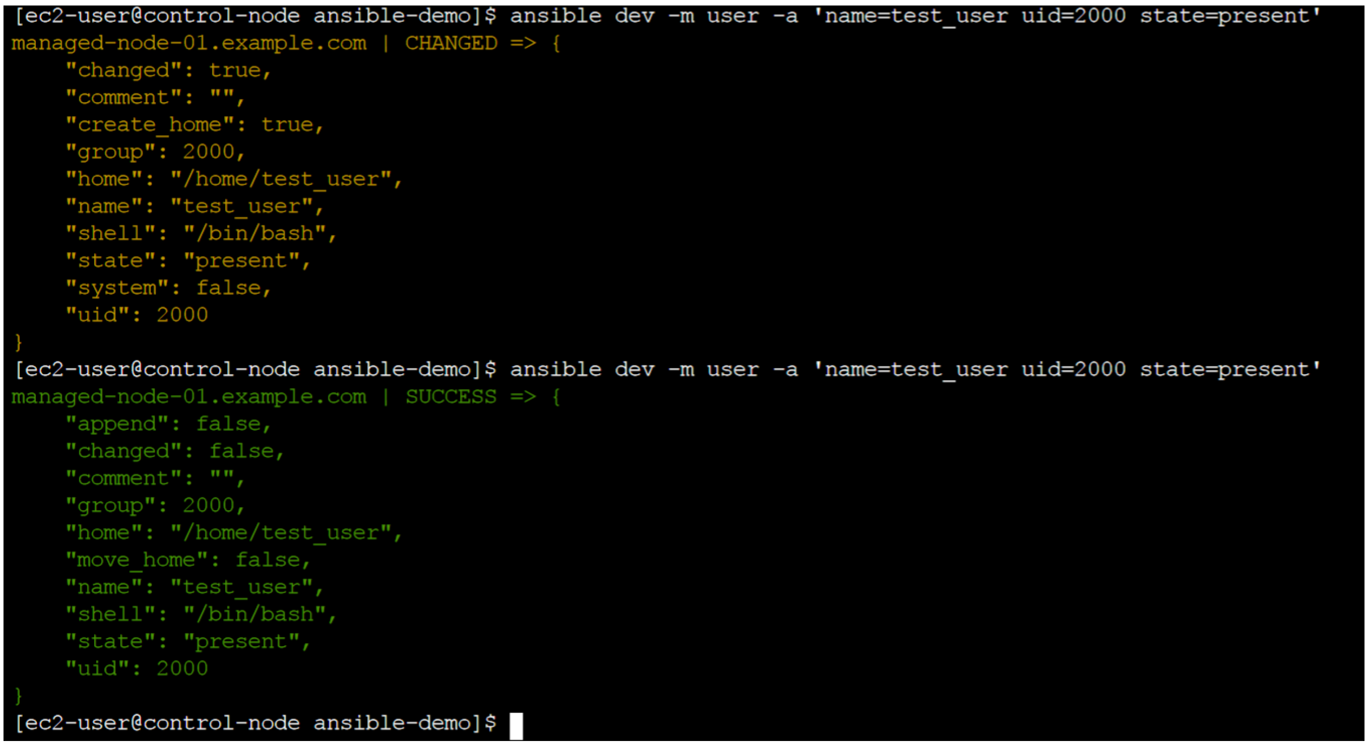
- Add a new linux user

```
ansible dev -m user -a 'name=test_user state=present'
```

- ***Idempotent***
  - Idempotent means modules that can run repeatedly to ensure systems are in a particular state without disrupting those systems if they already are.
  - To check this, we can run the previous ad-hoc again.
  - Here, if ansible has previously made some change, if the same command is executed again, there will be no change i.e **changed : false**

```
ansible dev -m user -a 'name=test_user state=present'
```

--



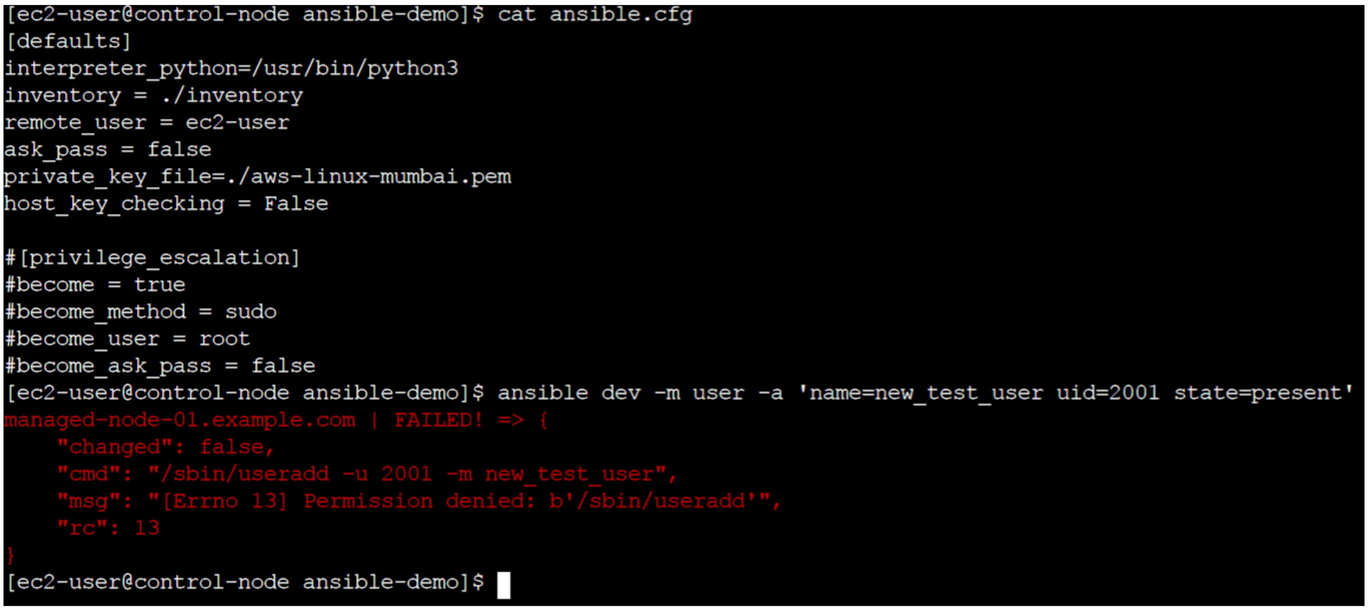
```
[ec2-user@control-node ansible-demo]$ ansible dev -m user -a 'name=test_user uid=2000 state=present'
managed-node-01.example.com | CHANGED => {
    "changed": true,
    "comment": "",
    "create_home": true,
    "group": 2000,
    "home": "/home/test_user",
    "name": "test_user",
    "shell": "/bin/bash",
    "state": "present",
    "system": false,
    "uid": 2000
}
[ec2-user@control-node ansible-demo]$ ansible dev -m user -a 'name=test_user uid=2000 state=present'
managed-node-01.example.com | SUCCESS => {
    "append": false,
    "changed": false,
    "comment": "",
    "group": 2000,
    "home": "/home/test_user",
    "move_home": false,
    "name": "test_user",
    "shell": "/bin/bash",
    "state": "present",
    "uid": 2000
}
[ec2-user@control-node ansible-demo]$
```

--

- Remove the Linux User

```
ansible dev -m user -a 'name=test_user state=absent'
```

- If we change the **./ansible.cfg** file and comment the **privilege_escalation**, the above command will not allow `ec2-user` to run `sudo` operations.



```
[ec2-user@control-node ansible-demo]$ cat ansible.cfg
[defaults]
interpreter_python=/usr/bin/python3
inventory = ./inventory
remote_user = ec2-user
ask_pass = false
private_key_file=./aws-linux-mumbai.pem
host_key_checking = False

#[privilege_escalation]
#become = true
#become_method = sudo
#become_user = root
#become_ask_pass = false
[ec2-user@control-node ansible-demo]$ ansible dev -m user -a 'name=new_test_user uid=2001 state=present'
managed-node-01.example.com | FAILED! => {
    "changed": false,
    "cmd": "/sbin/useradd -u 2001 -m new_test_user",
    "msg": "[Errno 13] Permission denied: b'/sbin/useradd'",
    "rc": 13
}
[ec2-user@control-node ansible-demo]$
```

--

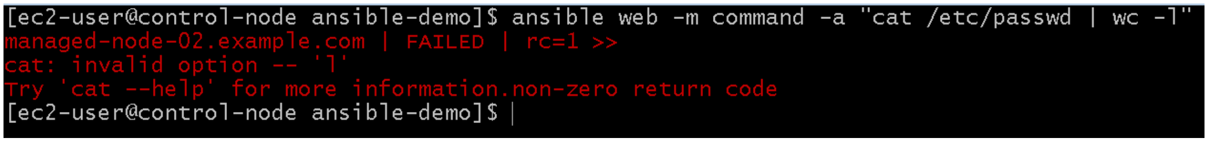
Command vs Shell Modules

- The **command** module allows administrators to execute arbitrary commands on the command line of managed hosts.

```
ansible all -m command -a 'id'
ansible all -m command -a 'df -h'
ansible all -m command -a 'cat /etc/passwd' -vvv
```

There are very significant differences in below modules:

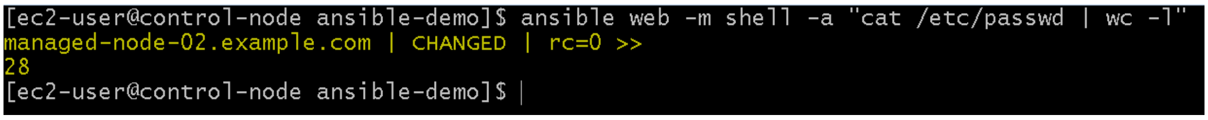
- ***command***
  - `piping or redirection` operations are not supported with the `command` module
  - Below command will result into an error.
  - `ansible qa -m command -a "cat /etc/passwd | wc -l"`



```
[ec2-user@control-node ansible-demo]$ ansible web -m command -a "cat /etc/passwd | wc -l"
managed-node-02.example.com | FAILED | rc=1 >>
cat: invalid option -- 'l'
Try 'cat --help' for more information.non-zero return code
[ec2-user@control-node ansible-demo]$
```

--

- `shell`
  - `piping or redirection` operations are supported with the `shell` module
  - Below command will result into an output
  - `ansible qa -m shell -a "cat /etc/passwd | wc -l"`

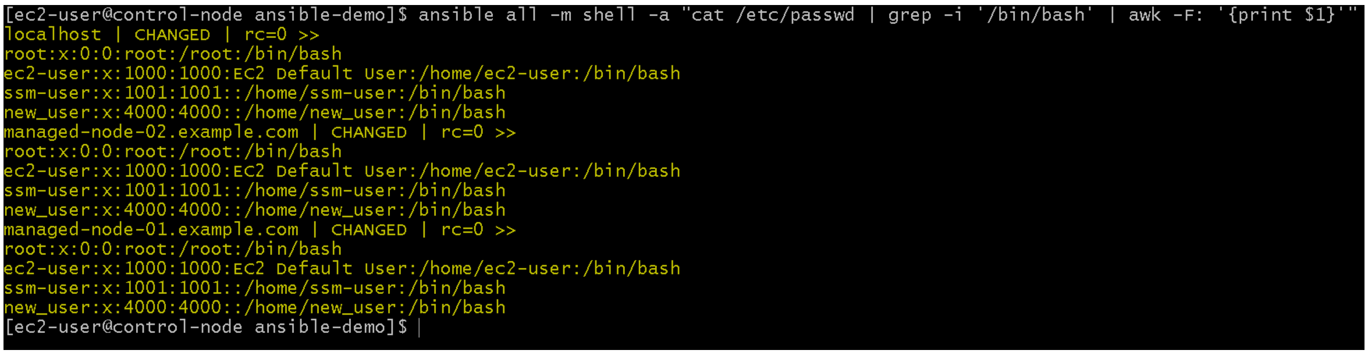


```
[ec2-user@control-node ansible-demo]$ ansible web -m shell -a "cat /etc/passwd | wc -l"
managed-node-02.example.com | CHANGED | rc=0 >>
28
[ec2-user@control-node ansible-demo]$
```

- Examples of linux commands that can be executed on a ad-hoc basis.

```
ansible all -m shell -a "cat /etc/passwd | grep -i '/bin/bash' | awk -F:
'{print $1}'"
ansible all -m shell -a "cat /etc/passwd | grep -i '/bin/bash' | wc -l"
```

--



```
[ec2-user@control-node ansible-demo]$ ansible all -m shell -a "cat /etc/passwd | grep -i '/bin/bash' | awk -F: '{print $1}'"
localhost | CHANGED | rc=0 >>
root:x:0:0:root:/root:/bin/bash
ec2-user:x:1000:1000:EC2 Default User:/home/ec2-user:/bin/bash
ssm-user:x:1001:1001::/home/ssm-user:/bin/bash
new_user:x:4000:4000::/home/new_user:/bin/bash
managed-node-02.example.com | CHANGED | rc=0 >>
root:x:0:0:root:/root:/bin/bash
ec2-user:x:1000:1000:EC2 Default User:/home/ec2-user:/bin/bash
ssm-user:x:1001:1001::/home/ssm-user:/bin/bash
new_user:x:4000:4000::/home/new_user:/bin/bash
managed-node-01.example.com | CHANGED | rc=0 >>
root:x:0:0:root:/root:/bin/bash
ec2-user:x:1000:1000:EC2 Default User:/home/ec2-user:/bin/bash
ssm-user:x:1001:1001::/home/ssm-user:/bin/bash
new_user:x:4000:4000::/home/new_user:/bin/bash
[ec2-user@control-node ansible-demo]$ |
```

--

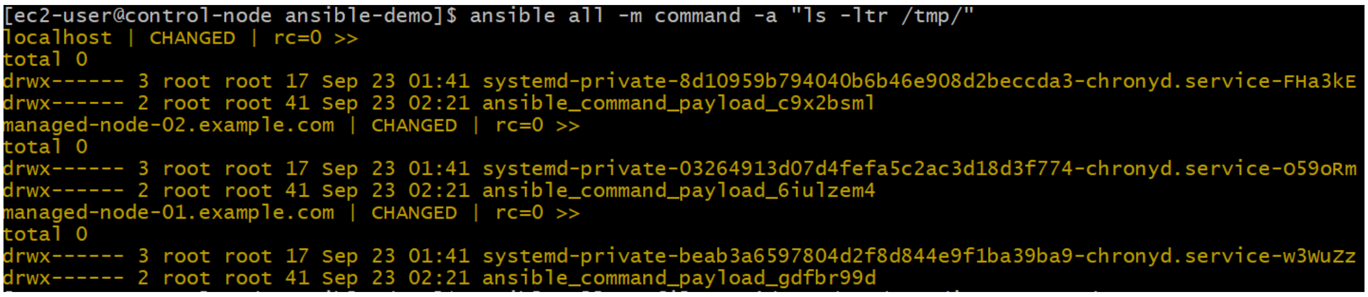
- The following modules are useful:
  - **file** (set permissions and other properties of a file)
  - **copy** (copy a local file to the managed host)
  - `get_url` (download a file to the managed host)
  - `synchronize` (to synchronize content like rsync)
  - `lineinfile` (make sure a certain line is or isn't in a file)
  - Software package management modules, such as `yum`, `apt`, `pip` and so on
  - System administration tools, such as
    - `service` to control daemons
    - `user` module to add, remove and configure users
    - `uri`, which interacts with a web server and can test functionality or issue API requests

--

file module

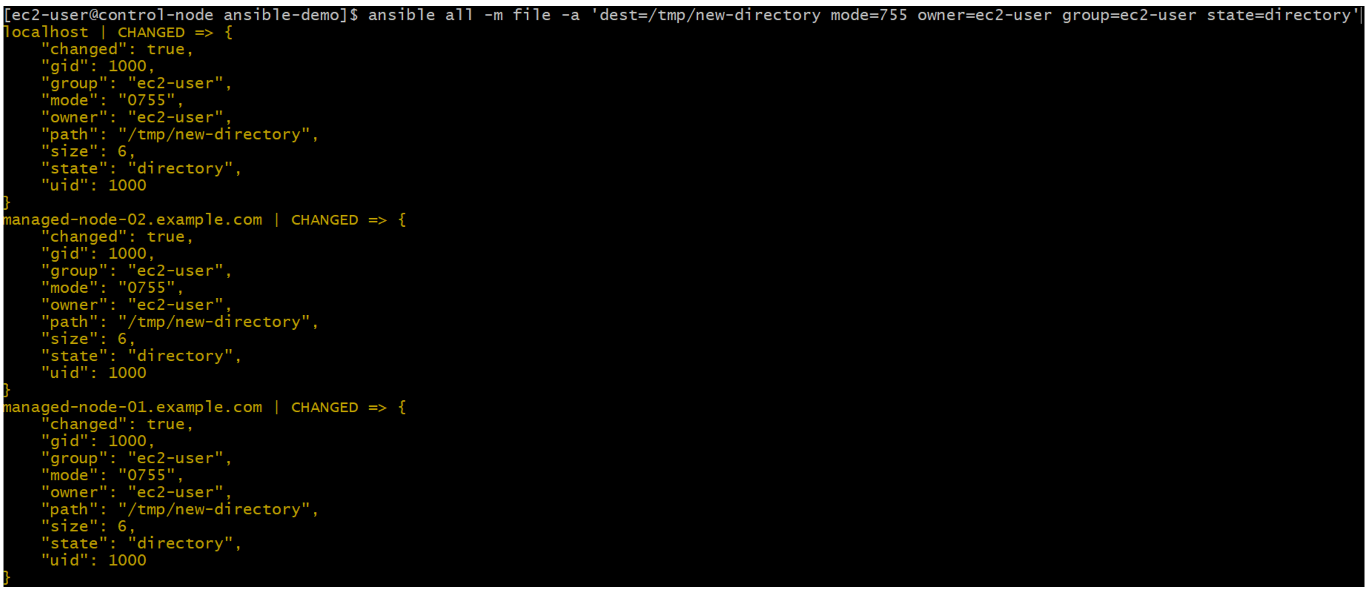
- `file` module commands
  - Create a directory directly to many servers

```
ansible all -m command -a "ls -ltr /tmp/"
ansible all -m file -a 'dest=/tmp/new-directory mode=755 owner=ec2-user group=ec2-
user state=directory'
ansible all -m command -a "ls -ltr /tmp/"
```



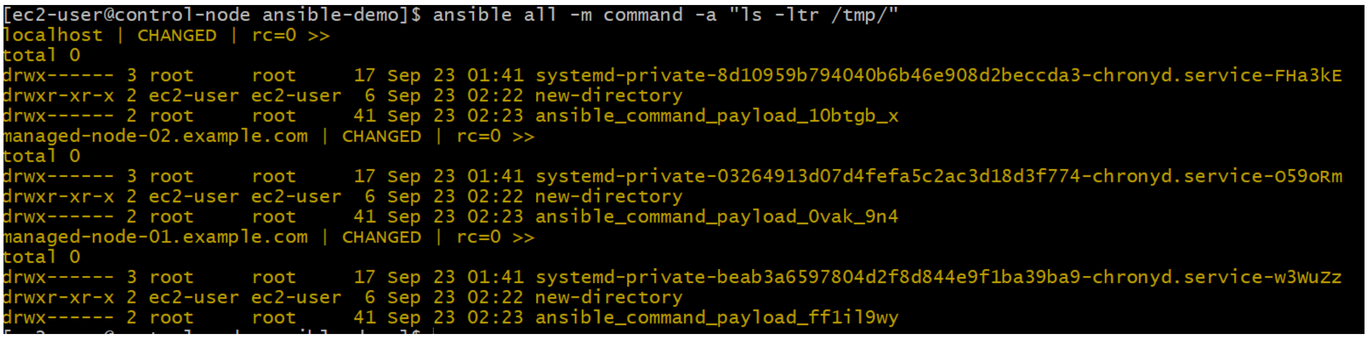
```
[ec2-user@control-node ansible-demo]$ ansible all -m command -a "ls -ltr /tmp/"
localhost | CHANGED | rc=0 >>
total 0
drwx------ 3 root root 17 Sep 23 01:41 systemd-private-8d10959b794040b6b46e908d2beccda3-chronyd.service-FHa3kE
drwx------ 2 root root 41 Sep 23 02:21 ansible_command_payload_c9x2bsml
managed-node-02.example.com | CHANGED | rc=0 >>
total 0
drwx------ 3 root root 17 Sep 23 01:41 systemd-private-03264913d07d4fefa5c2ac3d18d3f774-chronyd.service-059oRm
drwx------ 2 root root 41 Sep 23 02:21 ansible_command_payload_6iulzem4
managed-node-01.example.com | CHANGED | rc=0 >>
total 0
drwx------ 3 root root 17 Sep 23 01:41 systemd-private-beab3a6597804d2f8d844e9f1ba39ba9-chronyd.service-w3WuZz
drwx------ 2 root root 41 Sep 23 02:21 ansible_command_payload_gdfbr99d
```

--



```
[ec2-user@control-node ansible-demo]$ ansible all -m file -a 'dest=/tmp/new-directory mode=755 owner=ec2-user group=ec2-user state=directory'
localhost | CHANGED => {
    "changed": true,
    "gid": 1000,
    "group": "ec2-user",
    "mode": "0755",
    "owner": "ec2-user",
    "path": "/tmp/new-directory",
    "size": 6,
    "state": "directory",
    "uid": 1000
}
managed-node-02.example.com | CHANGED => {
    "changed": true,
    "gid": 1000,
    "group": "ec2-user",
    "mode": "0755",
    "owner": "ec2-user",
    "path": "/tmp/new-directory",
    "size": 6,
    "state": "directory",
    "uid": 1000
}
managed-node-01.example.com | CHANGED => {
    "changed": true,
    "gid": 1000,
    "group": "ec2-user",
    "mode": "0755",
    "owner": "ec2-user",
    "path": "/tmp/new-directory",
    "size": 6,
    "state": "directory",
    "uid": 1000
}
```

--



```
[ec2-user@control-node ansible-demo]$ ansible all -m command -a "ls -ltr /tmp/"
localhost | CHANGED | rc=0 >>
total 0
drwx------ 3 root     root     17 Sep 23 01:41 systemd-private-8d10959b794040b6b46e908d2beccda3-chronyd.service-FHa3kE
drwxr-xr-x 2 ec2-user ec2-user  6 Sep 23 02:22 new-directory
drwx------ 2 root     root     41 Sep 23 02:23 ansible_command_payload_10btgb_x
managed-node-02.example.com | CHANGED | rc=0 >>
total 0
drwx------ 3 root     root     17 Sep 23 01:41 systemd-private-03264913d07d4fefa5c2ac3d18d3f774-chronyd.service-059oRm
drwxr-xr-x 2 ec2-user ec2-user  6 Sep 23 02:22 new-directory
drwx------ 2 root     root     41 Sep 23 02:23 ansible_command_payload_0vak_9n4
managed-node-01.example.com | CHANGED | rc=0 >>
total 0
drwx------ 3 root     root     17 Sep 23 01:41 systemd-private-beab3a6597804d2f8d844e9f1ba39ba9-chronyd.service-w3WuZz
drwxr-xr-x 2 ec2-user ec2-user  6 Sep 23 02:22 new-directory
drwx------ 2 root     root     41 Sep 23 02:23 ansible_command_payload_ff1il9wy
```

- Delete a directory directly from many servers

```
ansible all -m file -a 'dest=/tmp/new-directory mode=755 owner=ec2-user group=ec2-user state=absent'
```

--

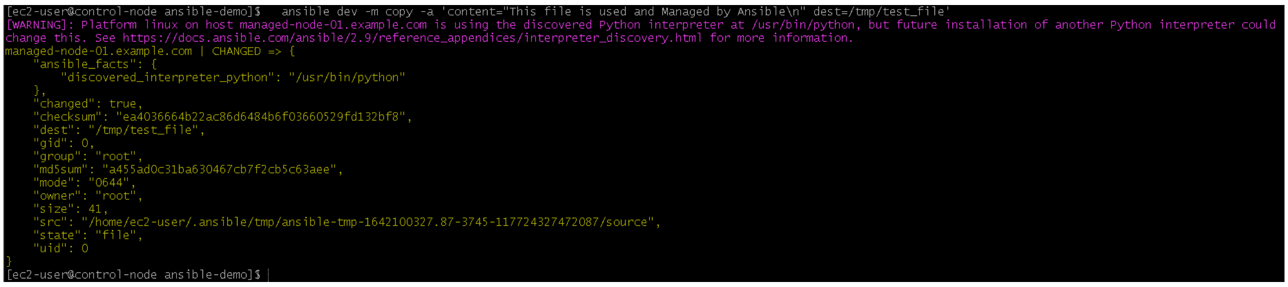
copy module

- Transfer a file from control node to managed servers, Module Name used is **copy**

```
ansible all -m copy -a 'src=/etc/hosts dest=/tmp/hosts'
```

- Write some content to this file

```
ansible dev -m copy -a 'content="This file is used and Managed by Ansible\n"
dest=/tmp/test_file'
```

- 

```
[ec2-user@control-node ansible-demo]$   ansible dev -m copy -a 'content="This file is used and Managed by Ansible\n" dest=/tmp/test_file'
[WARNING]: Platform linux on host managed-node-01.example.com 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.
managed-node-01.example.com | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "checksum": "ea4036664b22ac86d6484b6f03660529fd132bf8",
    "dest": "/tmp/test_file",
    "gid": 0,
    "group": "root",
    "md5sum": "a455ad0c31ba630467cb7f2cb5c63aee",
    "mode": "0644",
    "owner": "root",
    "size": 41,
    "src": "/home/ec2-user/.ansible/tmp/ansible-tmp-1642100327.87-3745-117724327472087/source",
    "state": "file",
    "uid": 0
}
[ec2-user@control-node ansible-demo]$
```

- To view the content of the file

```
ansible all -m command -a 'cat /tmp/test_file'
```

--

setup module

- gather facts will collect all linux level information of a particular and assign gather fact variables.

```
ansible dev -m setup
```

- filter facts

```
ansible dev -m setup -a 'filter=ansible_hostname'
ansible dev -m setup -a 'filter=ansible_fqdn'
ansible dev -m setup -a 'filter=ansible_pkg_mgr'
ansible dev -m setup -a 'filter=ansible_os_family'
ansible dev -m setup -a 'filter=ansible_distribution'
```

Note

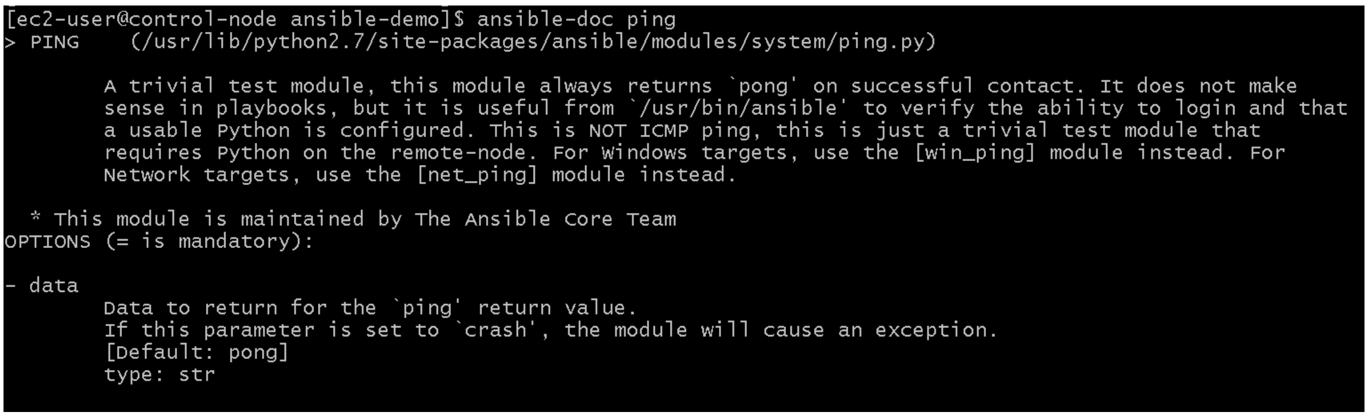
> As running multiple instances running in EC2 might incur costs over and above the Free Tier Limit, make sure EC2 instances are stopped if they are not in use.

--

Reference

- The below command will open up the `ping` module documentation page to get more detail information of all the options supported by this module.

```
ansible-doc ping
```



```
[ec2-user@control-node ansible-demo]$ ansible-doc ping
> PING    (/usr/lib/python2.7/site-packages/ansible/modules/system/ping.py)

        A trivial test module, this module always returns `pong' on successful contact. It does not make
        sense in playbooks, but it is useful from `/usr/bin/ansible' to verify the ability to login and that
        a usable Python is configured. This is NOT ICMP ping, this is just a trivial test module that
        requires Python on the remote-node. For Windows targets, use the [win_ping] module instead. For
        Network targets, use the [net_ping] module instead.

  * This module is maintained by The Ansible Core Team
OPTIONS (= is mandatory):

- data
        Data to return for the `ping' return value.
        If this parameter is set to `crash', the module will cause an exception.
        [Default: pong]
        type: str
```