Introducing Ansible core components

- Ansible Configuration Files - Defines various default values to be used by Ansible. Almost all default values can be overwritten in ansible ad-hoc commands and playbooks.
- Ansible Inventories - Defines hosts and groups of hosts on which ansible operates.
- Ansible Modules - Module is small python program which is designed to execute specific task on ansible managed nodes or local node.
- Ansible Variables - Variables used in playbook.
- Ansible Facts - Represent data about remote systems gathered by Ansible which can be used in playbooks as variables. Ansible facts are also very important for conditional playbook execution.
- Ansible Plays - Ansible play basically defines target hosts and specific task(s) to be executed on target hosts. Ansible play is written in YAML.
- Ansible Playbooks - List of ansible plays

Ansible Config file

1. ANSIBLE_CONFIG - Environment Variable if set
2. ansible.cfg - In current working Directory
3. ~/.ansible.cfg - User’s home directory
4. /etc/ansible/ansible.cfg - System wide directory

Ansible searches for configuration file in the above order. Config file found first is considered and others are simply ignored.

Example of ansible config file (ansible.cfg)

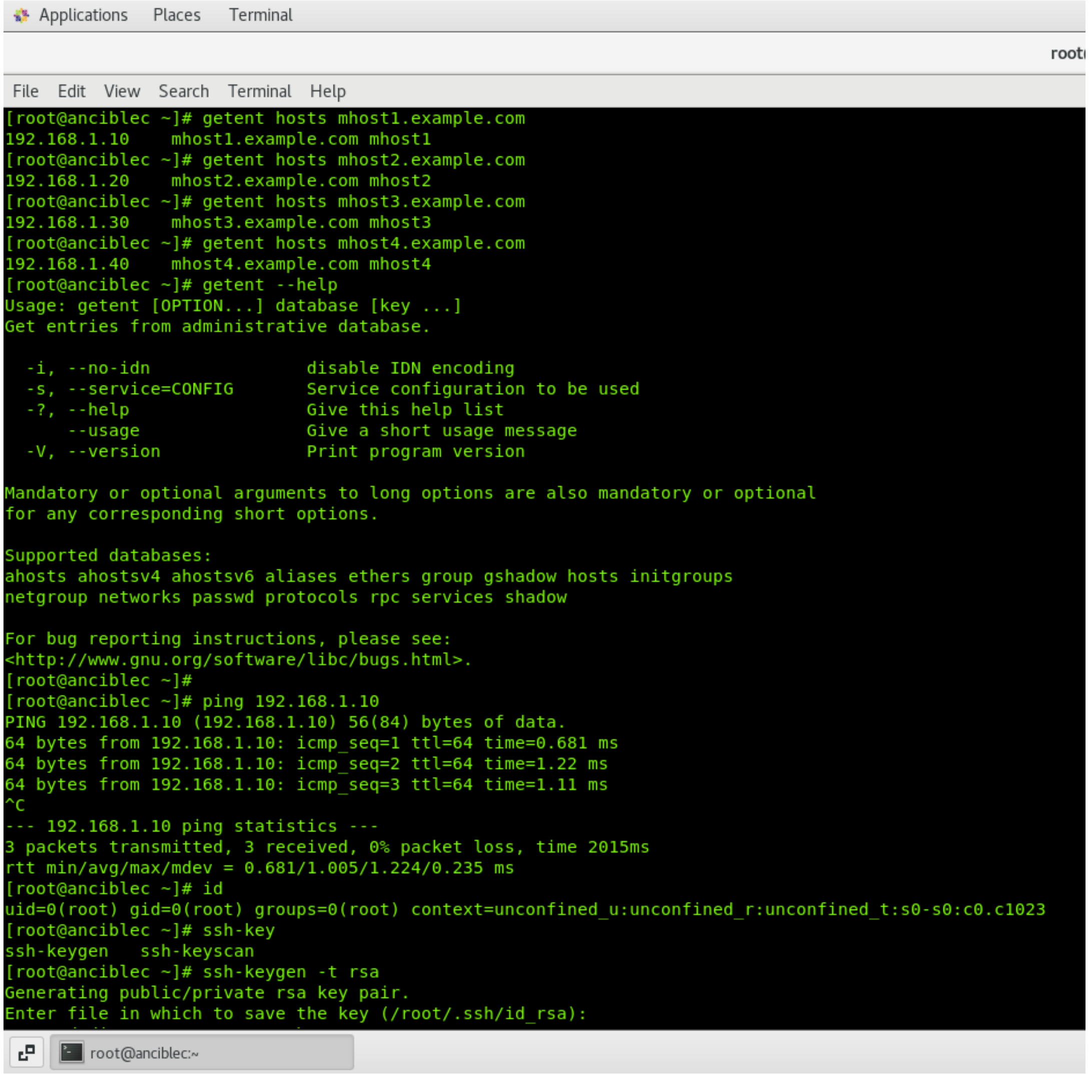
We can define different settings using ansible config file. For example,

- Location of ansible inventory file
- Location of roles directory
- User to be used to connect to remote machines.
- Enabling/Disabling privilege escalation and many more settings.…

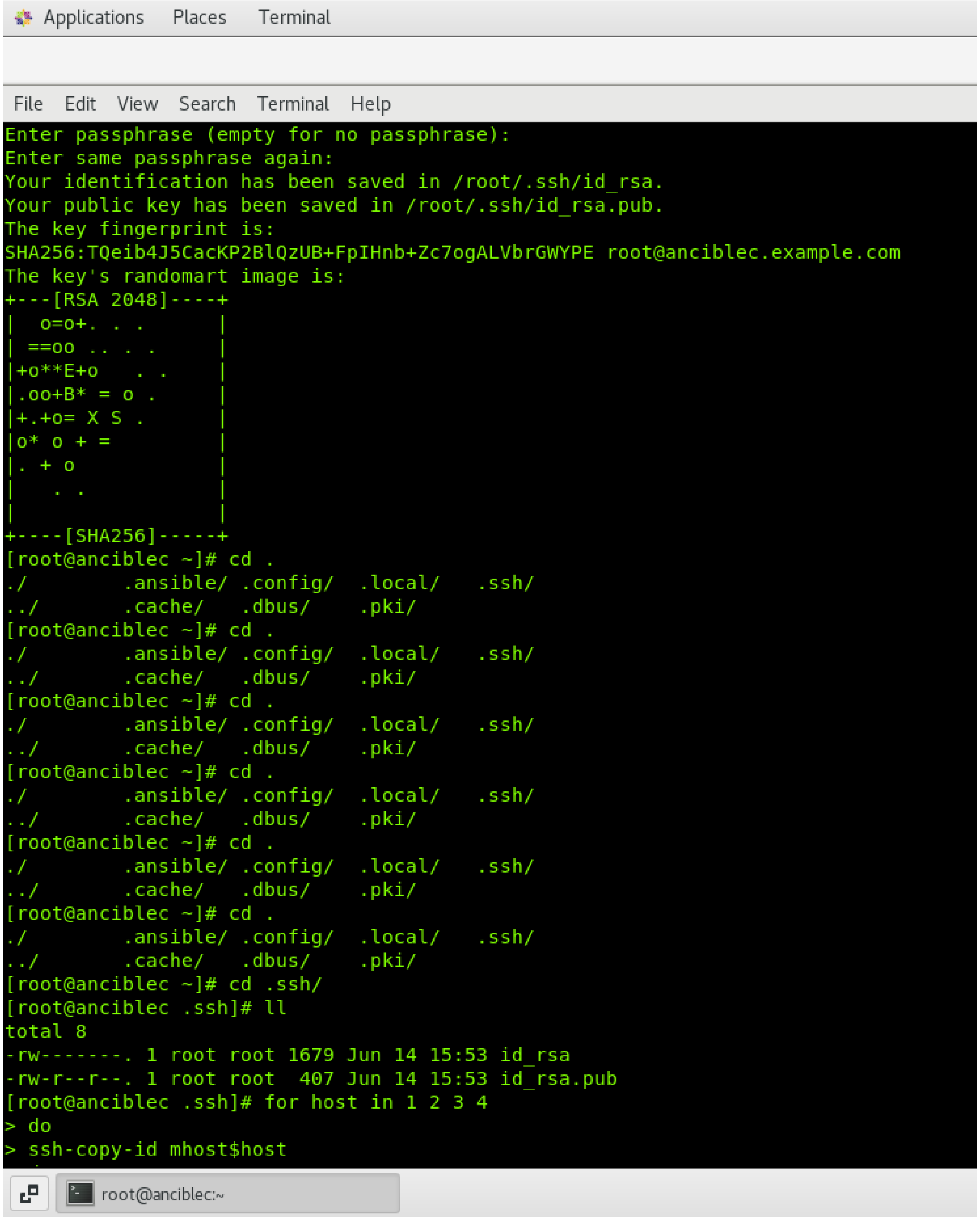
Sample config file:

```
vim /home/ansible/tasks/ansible.cfg

[defaults]

inventory = /home/ansible/tasks/nodes

roles_path = /home/ansible/tasks/roles

remote_user = ansible

[privilege_escalation] become = yes

become_user = root

become_method = sudo
```

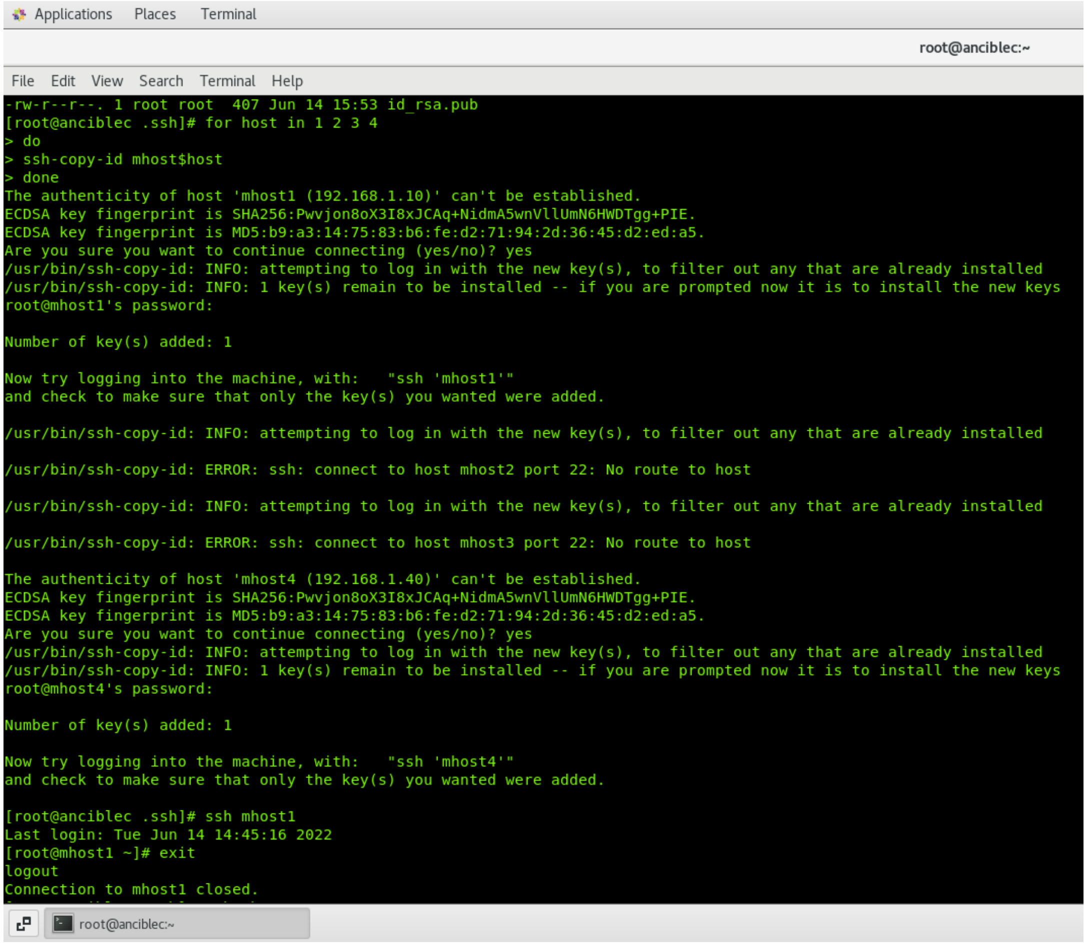
become_ask_pass = False



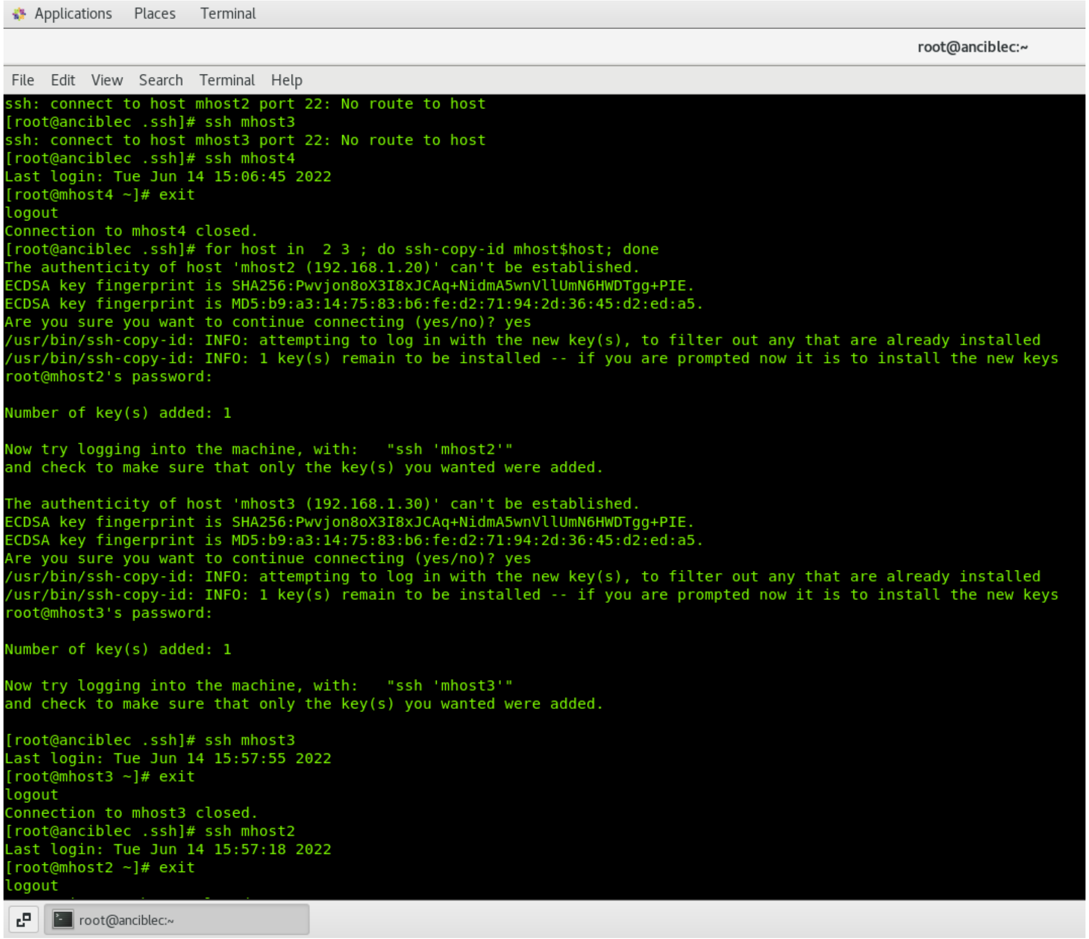
```
[root@anciblec ~]# getent hosts mhost1.example.com
192.168.1.10    mhost1.example.com mhost1
[root@anciblec ~]# getent hosts mhost2.example.com
192.168.1.20    mhost2.example.com mhost2
[root@anciblec ~]# getent hosts mhost3.example.com
192.168.1.30    mhost3.example.com mhost3
[root@anciblec ~]# getent hosts mhost4.example.com
192.168.1.40    mhost4.example.com mhost4
[root@anciblec ~]# getent --help
Usage: getent [OPTION...] database [key ...]
Get entries from administrative database.

  -i, --no-idn               disable IDN encoding
  -s, --service=CONFIG       Service configuration to be used
  -?, --help                 Give this help list
      --usage                Give a short usage message
  -V, --version              Print program version

Mandatory or optional arguments to long options are also mandatory or optional
for any corresponding short options.

Supported databases:
ahosts ahostsv4 ahostsv6 aliases ethers group gshadow hosts initgroups
netgroup networks passwd protocols rpc services shadow

For bug reporting instructions, please see:
<http://www.gnu.org/software/libc/bugs.html>.
[root@anciblec ~]#
[root@anciblec ~]# ping 192.168.1.10
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.
64 bytes from 192.168.1.10: icmp_seq=1 ttl=64 time=0.681 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=64 time=1.22 ms
64 bytes from 192.168.1.10: icmp_seq=3 ttl=64 time=1.11 ms
^C
--- 192.168.1.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2015ms
rtt min/avg/max/mdev = 0.681/1.005/1.224/0.235 ms
[root@anciblec ~]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@anciblec ~]# ssh-key
ssh-keygen   ssh-keyscan
[root@anciblec ~]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
```



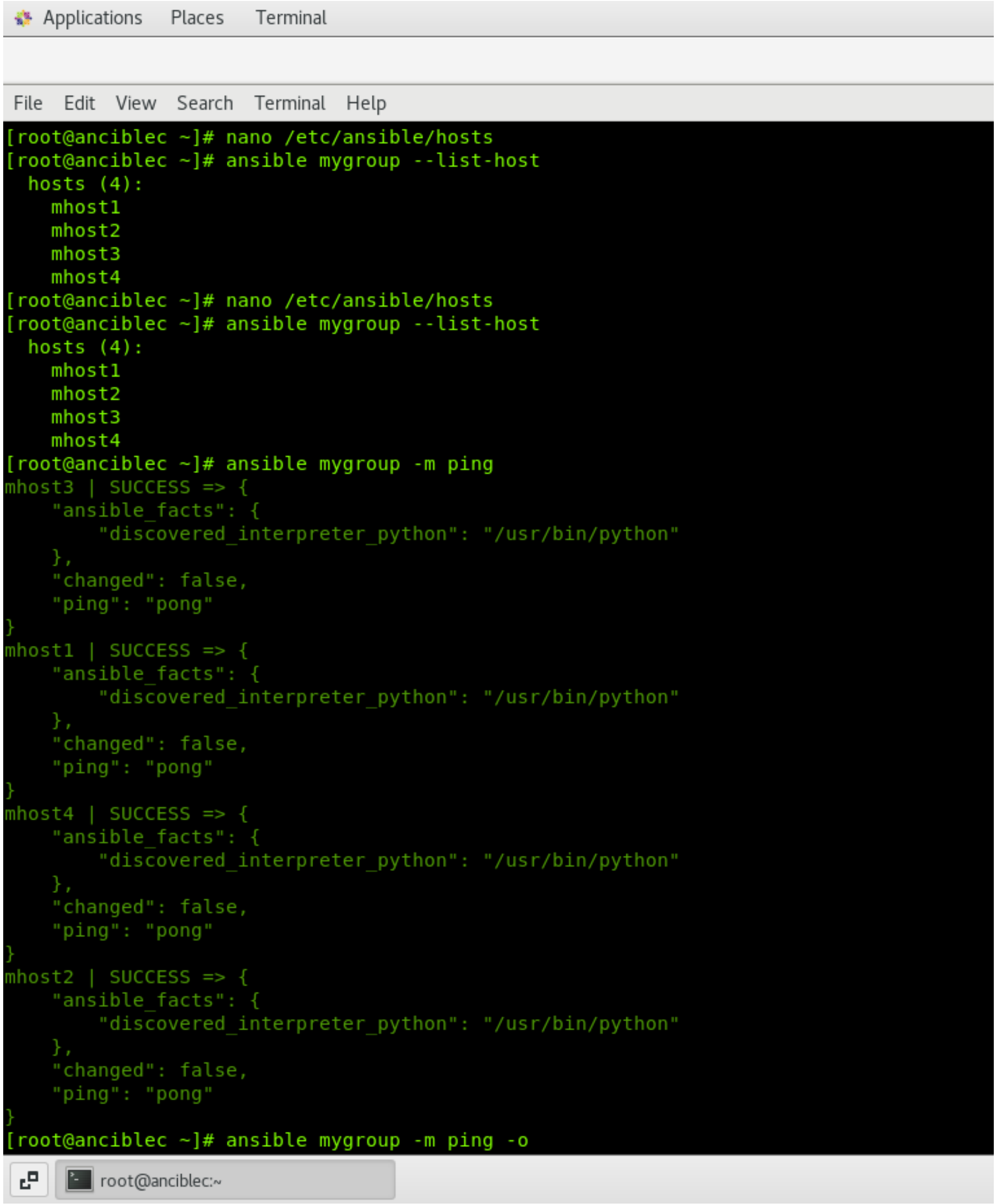
```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:TQeib4J5CacKP2BlQzUB+FpIHnb+Zc7ogALVbrGWYPE root@anciblec.example.com
The key's randomart image is:
+---[RSA 2048]----+
|  o=o+. . .      |
| ==oo .. . .     |
|+o**E+o   . .    |
|.oo+B* = o .     |
|+.+o= X S .      |
|o* o + =         |
|. + o            |
|   . .           |
|                 |
+----[SHA256]-----+
[root@anciblec ~]# cd .
./         .ansible/ .config/  .local/   .ssh/
../        .cache/   .dbus/    .pki/
[root@anciblec ~]# cd .
./         .ansible/ .config/  .local/   .ssh/
../        .cache/   .dbus/    .pki/
[root@anciblec ~]# cd .
./         .ansible/ .config/  .local/   .ssh/
../        .cache/   .dbus/    .pki/
[root@anciblec ~]# cd .
./         .ansible/ .config/  .local/   .ssh/
../        .cache/   .dbus/    .pki/
[root@anciblec ~]# cd .
./         .ansible/ .config/  .local/   .ssh/
../        .cache/   .dbus/    .pki/
[root@anciblec ~]# cd .
./         .ansible/ .config/  .local/   .ssh/
../        .cache/   .dbus/    .pki/
[root@anciblec ~]# cd .ssh/
[root@anciblec .ssh]# ll
total 8
-rw-------. 1 root root 1679 Jun 14 15:53 id_rsa
-rw-r--r--. 1 root root  407 Jun 14 15:53 id_rsa.pub
[root@anciblec .ssh]# for host in 1 2 3 4
> do
> ssh-copy-id mhost$host
```



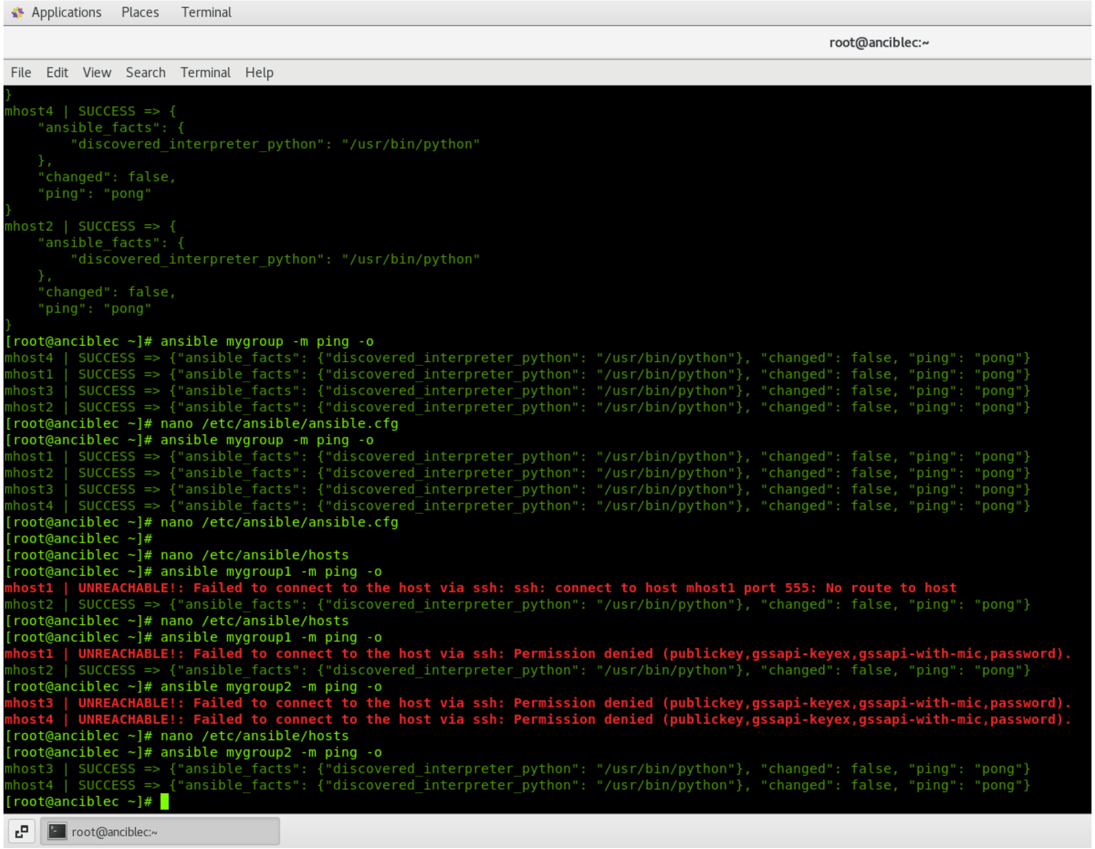
```
-rw-r--r--. 1 root root  407 Jun 14 15:53 id_rsa.pub
[root@anciblec .ssh]# for host in 1 2 3 4
> do
> ssh-copy-id mhost$host
> done
The authenticity of host 'mhost1 (192.168.1.10)' can't be established.
ECDSA key fingerprint is SHA256:Pwvjon8oX3I8xJCAq+NidmA5wnVllUmN6HWDTgg+PIE.
ECDSA key fingerprint is MD5:b9:a3:14:75:83:b6:fe:d2:71:94:2d:36:45:d2:ed:a5.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@mhost1's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'mhost1'"
and check to make sure that only the key(s) you wanted were added.

/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed

/usr/bin/ssh-copy-id: ERROR: ssh: connect to host mhost2 port 22: No route to host

/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed

/usr/bin/ssh-copy-id: ERROR: ssh: connect to host mhost3 port 22: No route to host

The authenticity of host 'mhost4 (192.168.1.40)' can't be established.
ECDSA key fingerprint is SHA256:Pwvjon8oX3I8xJCAq+NidmA5wnVllUmN6HWDTgg+PIE.
ECDSA key fingerprint is MD5:b9:a3:14:75:83:b6:fe:d2:71:94:2d:36:45:d2:ed:a5.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@mhost4's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'mhost4'"
and check to make sure that only the key(s) you wanted were added.

[root@anciblec .ssh]# ssh mhost1
Last login: Tue Jun 14 14:45:16 2022
[root@mhost1 ~]# exit
logout
Connection to mhost1 closed.
```



```
ssh: connect to host mhost2 port 22: No route to host
[root@anciblec .ssh]# ssh mhost3
ssh: connect to host mhost3 port 22: No route to host
[root@anciblec .ssh]# ssh mhost4
Last login: Tue Jun 14 15:06:45 2022
[root@mhost4 ~]# exit
logout
Connection to mhost4 closed.
[root@anciblec .ssh]# for host in  2 3 ; do ssh-copy-id mhost$host; done
The authenticity of host 'mhost2 (192.168.1.20)' can't be established.
ECDSA key fingerprint is SHA256:Pwvjon8oX3I8xJCAq+NidmA5wnVllUmN6HWDTgg+PIE.
ECDSA key fingerprint is MD5:b9:a3:14:75:83:b6:fe:d2:71:94:2d:36:45:d2:ed:a5.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@mhost2's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'mhost2'"
and check to make sure that only the key(s) you wanted were added.

The authenticity of host 'mhost3 (192.168.1.30)' can't be established.
ECDSA key fingerprint is SHA256:Pwvjon8oX3I8xJCAq+NidmA5wnVllUmN6HWDTgg+PIE.
ECDSA key fingerprint is MD5:b9:a3:14:75:83:b6:fe:d2:71:94:2d:36:45:d2:ed:a5.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@mhost3's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'mhost3'"
and check to make sure that only the key(s) you wanted were added.

[root@anciblec .ssh]# ssh mhost3
Last login: Tue Jun 14 15:57:55 2022
[root@mhost3 ~]# exit
logout
Connection to mhost3 closed.
[root@anciblec .ssh]# ssh mhost2
Last login: Tue Jun 14 15:57:18 2022
[root@mhost2 ~]# exit
logout
```



```
[root@anciblec ~]# nano /etc/ansible/hosts
[root@anciblec ~]# ansible mygroup --list-host
  hosts (4):
    mhost1
    mhost2
    mhost3
    mhost4
[root@anciblec ~]# nano /etc/ansible/hosts
[root@anciblec ~]# ansible mygroup --list-host
  hosts (4):
    mhost1
    mhost2
    mhost3
    mhost4
[root@anciblec ~]# ansible mygroup -m ping
mhost3 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
mhost1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
mhost4 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
mhost2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
[root@anciblec ~]# ansible mygroup -m ping -o
```



```
}
mhost4 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
mhost2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
[root@anciblec ~]# ansible mygroup -m ping -o
mhost4 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python"}, "changed": false, "ping": "pong"}
mhost1 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python"}, "changed": false, "ping": "pong"}
mhost3 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python"}, "changed": false, "ping": "pong"}
mhost2 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python"}, "changed": false, "ping": "pong"}
[root@anciblec ~]# nano /etc/ansible/ansible.cfg
[root@anciblec ~]# ansible mygroup -m ping -o
mhost1 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python"}, "changed": false, "ping": "pong"}
mhost2 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python"}, "changed": false, "ping": "pong"}
mhost3 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python"}, "changed": false, "ping": "pong"}
mhost4 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python"}, "changed": false, "ping": "pong"}
[root@anciblec ~]# nano /etc/ansible/ansible.cfg
[root@anciblec ~]#
[root@anciblec ~]# nano /etc/ansible/hosts
[root@anciblec ~]# ansible mygroup1 -m ping -o
mhost1 | UNREACHABLE!: Failed to connect to the host via ssh: ssh: connect to host mhost1 port 555: No route to host
mhost2 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python"}, "changed": false, "ping": "pong"}
[root@anciblec ~]# nano /etc/ansible/hosts
[root@anciblec ~]# ansible mygroup1 -m ping -o
mhost1 | UNREACHABLE!: Failed to connect to the host via ssh: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
mhost2 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python"}, "changed": false, "ping": "pong"}
[root@anciblec ~]# ansible mygroup2 -m ping -o
mhost3 | UNREACHABLE!: Failed to connect to the host via ssh: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
mhost4 | UNREACHABLE!: Failed to connect to the host via ssh: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
[root@anciblec ~]# nano /etc/ansible/hosts
[root@anciblec ~]# ansible mygroup2 -m ping -o
mhost3 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python"}, "changed": false, "ping": "pong"}
mhost4 | SUCCESS => {"ansible_facts": {"discovered_interpreter_python": "/usr/bin/python"}, "changed": false, "ping": "pong"}
[root@anciblec ~]# 
```