Lab: Building an Ansible Inventory

Introduction:

An inventory defines a **collection of hosts** that Ansible will manage. These hosts can also be assigned to **groups**, which can be managed collectively. Groups can contain **child groups**, and hosts can be members of multiple groups. The inventory can also **set variables** that **apply to the hosts** and **groups that it defines**.

Host inventories can be defined in two different ways.

1. A **static host inventory** can be defined by a **text file**.
2. A **dynamic host inventory** can be generated by a script or other program as needed, using external information providers

In this Lab, you will learn below items:

Objective

- Setup **SSH password less** login
- Building an **Ansible Inventory**
- Managing **Ansible Configuration** Files
- Running **Ad Hoc** Commands

1. Login into the Control node **ansi-master** as **root** user with password as **linux.**

1.1 Let us generate SSH key-pair for **root** user.

```
# ssh-keygen -t rsa -N ''
```

Note: Keep pressing enter without entering or changing any value

Output:

```
[root@ansi-master ~]# ssh-keygen -t rsa -N ''
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
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:vfsp+p+nsHEe3IF4Kkpokh07XwkBotdyxvbUzzHzzWM root@ansi-master
The key's randomart image is:
+---[RSA 3072]----+
|  . .            |
| . + . .         |
|. o * o . +      |
| . = o . + * +   |
|    . o S = + E  |
|   o + . . = o o |
|  o * . + = + .  |
|   o + o ..B +.  |
|      o .o++Bo   |
+----[SHA256]-----+
```

1.2 Let us gather ssh public keys, by executing below command

```
# ssh-keyscan ansi-node1 ansi-node2 ansi-node3 >>
~/.ssh/known_hosts
```

Output:

```
[root@ansi-master ~]# ssh-keyscan ansi-node1 ansi-node2 ansi-node3 >> ~/.ssh/known_hosts
# ansi-node1:22 SSH-2.0-OpenSSH_8.0
# ansi-node1:22 SSH-2.0-OpenSSH_8.0
# ansi-node1:22 SSH-2.0-OpenSSH_8.0
# ansi-node2:22 SSH-2.0-OpenSSH_8.0
# ansi-node2:22 SSH-2.0-OpenSSH_8.0
# ansi-node2:22 SSH-2.0-OpenSSH_8.0
# ansi-node3:22 SSH-2.0-OpenSSH_8.0
# ansi-node3:22 SSH-2.0-OpenSSH_8.0
# ansi-node3:22 SSH-2.0-OpenSSH_8.0
```

1.3 Let us copy the keys to all the nodes, by executing below command

Note: password **linux** when prompted.

```
# ssh-copy-id ansi-master
```

Output:

```
[root@ansi-master ~]# ssh-copy-id ansi-master
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
The authenticity of host 'ansi-master (192.168.100.150)' can't be established.
ECDSA key fingerprint is SHA256:ARCmtH/aMBLviMuUgv+2ROx5L7ZRX55ndKmIofVYezM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? 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@ansi-master's password:

Number of key(s) added: 1

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

```
# ssh-copy-id ansi-node1
```

Output:

```
[root@ansi-master ~]# ssh-copy-id ansi-node1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
/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@ansi-node1's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'ansi-node1'"
and check to make sure that only the key(s) you wanted were added.
```

```
# ssh-copy-id ansi-node2
```

Output:

```
[root@ansi-master ~]# ssh-copy-id ansi-node2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
/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@ansi-node2's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'ansi-node2'"
and check to make sure that only the key(s) you wanted were added.
```

```
# ssh-copy-id ansi-node3
```

Output:

```
[root@ansi-master ~]# ssh-copy-id ansi-node3
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
/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@ansi-node3's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'ansi-node3'"
and check to make sure that only the key(s) you wanted were added.
```

1.4 Let us add Hosts and Group to the default inventory file **/etc/ansible/hosts.**

```
# cat >> /etc/ansible/hosts << EOF
ansi-node1
[webserver]
ansi-node2
EOF
```

Note: Insert the above values at the end of the file

1.5 Let us verify the managed hosts in the **/etc/ansible/hosts** inventory file

```
# ansible all --list-hosts
```

Output:

```
[root@ansi-master ~]# ansible all --list-hosts
  hosts (2):
    ansi-node1
    ansi-node2
```

1.6 Let us verify Use the ansible ungrouped --list-hosts command to list only managed hosts that do not belong to a group

```
# ansible ungrouped  --list-hosts
```

Output:

```
[root@ansi-master ~]# ansible ungrouped  --list-hosts
  hosts (1):
    ansi-node1
```

1.7 Let us Use the ansible webservers --list-hosts command to list only managed hosts that belong to webservers group

```
# ansible webserver --list-hosts
```

Output:

```
[root@ansi-master ~]# ansible webserver --list-hosts
  hosts (1):
    ansi-node2
```

1.8 Let us create a custom static inventory file named **inventory** in the **/root** working directory

Information about three managed hosts is listed in the following table. You will assign each host to multiple groups for management purposes based on the purpose of the host, the city where it is located and the deployment environment to which it belongs.

In addition, groups of Indian cities (Bangalore, Delhi and Pune) must be set up as children of the group us so that hosts in India can be managed as a group.

Server Inventory Specifications:

HOSTNAME	PURPOSE	LOCATION	ENVIRONMENT
ansi-node1	**Web server**	**Bangalore**	**Development**
ansi-node2	**Web server**	**Delhi**	**Testing**
ansi-node3	**Web server**	**Pune**	**Production**

```
# cat > ./inventory << EOF
[webservers]
ansi-node[1:3]
[bangalore]
ansi-node1
[delhi]
ansi-node2
[pune]
ansi-node3
[development]
ansi-node1
[testing]
ansi-node2
[production]
ansi-node3
[india:children]
delhi
bangalore
pune
EOF
```

Notice inventory can be grouped in different ways as shown above.

1.9 Use the ansible **all -i inventory --list-hosts** command to list all managed hosts

```
# ansible all -i inventory --list-hosts
```

Output:

```
[root@ansi-master ~]# ansible all -i inventory --list-hosts
  hosts (3):
    ansi-node1
    ansi-node2
    ansi-node3
```

1.10 Use the ansible **ungrouped -i inventory --list-hosts** command to list all managed hosts listed in the inventory file but are not part of a group.

```
# ansible ungrouped -i inventory  --list-hosts
```

Output:

```
[devops@ansi-master ~]$ ansible ungrouped -i inventory  --list-hosts
[WARNING]: No hosts matched, nothing to do
  hosts (0):
```

Note: There are no ungrouped managed hosts in the inventory file

1.11 Use the ansible **development** -i inventory --list-hosts command to list all managed hosts listed in the **development** group.

```
# ansible development -i inventory --list-hosts
```

Output:

```
[root@ansi-master ~]# ansible development -i inventory --list-hosts
  hosts (1):
    ansi-node1
```

1.12 Use the ansible **bangalore** -i inventory --list-hosts command to list all managed hosts listed in the **bangalore** group

```
# ansible bangalore -i inventory --list-hosts
```

Output:

```
[root@ansi-master ~]# ansible bangalore -i inventory --list-hosts
  hosts (1):
    ansi-node1
```

1.13 Use the ansible india -i inventory --list-hosts command to list all managed hosts listed in the **india** group

```
# ansible india -i inventory --list-hosts
```

Output:

```
[root@ansi-master ~]# ansible india -i inventory --list-hosts
  hosts (3):
    ansi-node2
    ansi-node1
    ansi-node3
```

Let us see how to Managing Ansible Configuration Files

1.14 In the home directory create a ansible.cfg file with below entries

```
# cat > ./ansible.cfg <<EOF
[defaults]
inventory = ./inventory
EOF
```

1.15 Let us verify the configuration is working correctly, by executing the below command

```
# ansible all --list-hosts
```

Note: The inventory was picked from home directory as per the cfg file

Output:

```
[root@ansi-master ~]# ansible all --list-hosts
  hosts (3):
    ansi-node1
    ansi-node2
    ansi-node3
```

1.16 Let us edit the **/root /ansible.cfg** file in a text editor

```
# cat >> ./ansible.cfg <<EOF

[privilege_escalation]
become=true
become_method=sudo
become_user=root
become_ask_pass=true
EOF
```

1.17 Verify the contents of the file **/root /ansible.cfg**

```
# cat ./ansible.cfg
```

```
[devops@ansi-master ~]$ cat /home/devops/ansible.cfg
[defaults]
inventory = ./inventory

[privilege_escalation]
become=true
become_method=sudo
become_user=root
become_ask_pass=true
```

1.18 Let us run the --list-hosts command, this should prompt us for the password as per the configuration

```
# ansible all --list-hosts
```

Note: Enter **linux** as password when prompted.

```
[root@ansi-master ~]# ansible all --list-hosts
BECOME password:
  hosts (3):
    ansi-node1
    ansi-node2
    ansi-node3
```

1.19 Let us see how to run adhoc commands

1.20 Using all the hostgroup and the ping module execute an ad hoc command that ensures all managed hosts can run Ansible modules using python

```
# ansible all -m ping
```

Note: Enter **linux** as password when prompted.

```
[root@ansi-master ~]# ansible all -m ping
BECOME password:
ansi-node1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ansi-node2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ansi-node3 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
```

2 Let us modify the configuration file to change the **become_ask_pass** to **false**

```
# sed -i -e "s/become_ask_pass=true/become_ask_pass=false/g"
ansible.cfg
```

Note: You can use any text editor to change the value to false.

3 Using the command module, execute an ad hoc command to identity the user account that ansible uses to perform operations on managed hosts.

```
# ansible all -m ping
```

Note: As we set the ask pass to false, it didn't prompt us for the password

```
[root@ansi-master ~]# ansible all -m ping
ansi-node2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ansi-node3 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ansi-node1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
```