

Building the Inventory

Duration: 25 Minutes

Introduction

In the first lab, we installed Ansible and learned how to use Ad-hoc commands and write a simple playbook to utilize built-in module to test connectivity to target hosts. In this lab, we will build playbook to perform useful tasks.

Exercise 1: Building your Inventory

The automation lab we are using consists of multiple Check Point devices, windows VMS and Ubuntu orchestrator and target. We will built a static inventory file that contains all the managed nodes and we will use it for the rest of the tasks.

1. Under the Playbooks directory, create a new file called hosts. This will be our inventory.
2. Add our targets and their variables using the following format.

```
[ubuntu]
10.0.3.61
10.0.3.62

[ubuntu:vars]
ansible_user=admin
ansible_ssh_pass=

[checkpoint:children]
checkpoint_mgmt
checkpoint_gw

[checkpoint_mgmt]
10.0.1.100 ansible_user=admin ansible_password=Cpwins!1

[checkpoint_gw]
10.0.1.10 ansible_user=admin ansible_password=Cpwins!1
```

- Notice that we have a group called `ubuntu` and it has variable assigned using the method we learned in lab 1 `[ubuntu:vars]`.
- We also created a group to represent the Check Point devices. In this case, we assigned the variables directly inside the group.
- In case we would like to apply changes to the management or gateway only, we created separate groups, one for the gateway and one for the management.

3. Disable the SSH Key Checking for Ansible. Create a file `ansible.cfg` in the current directory.

```
admin@orchestrator:~/adhoc$ vi ansible.cfg
admin@orchestrator:~/adhoc$ cat ansible.cfg
[defaults]
host_key_checking = False
admin@orchestrator:~/adhoc$
```

4. Run the ping module as ad-hoc command to test connectivity to the gateway:

```
ansible checkpoint_gw -i hosts -m ping
```

```
admin@orchestrator:~$ ansible checkpoint_gw -i hosts -m ping
[WARNING]: Platform linux on host 10.0.1.10 is using the discovered Python interpreter at /usr/bin/python3, but future installation of
another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.15/reference_appendices/interpreter_discovery.html for more information.
10.0.1.10 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
```

- Note that the connectivity check worked. However, there is a warning regarding the python path.
- As this is R81.20, the python location should be `/opt/CPsuite-R81.20/fw1/Python/bin/python3.7.cat`
- Older Gateways use Python 2.7 paths such as `/opt/CPsuite-R80.20/fw1/Python/bin/python2.7`.
- To specify the python interpreter in Python, use the variable `ansible_python_interpreter`

5. Edit the inventory file and add the python interpreter as a variable for all Check Point devices.

```
[ubuntu]
10.0.3.61
10.0.3.62


[ubuntu:vars]
ansible_user=admin
ansible_ssh_pass=

[checkpoint:children]
checkpoint_mgmt
checkpoint_gw

[checkpoint_mgmt]
10.0.1.100 ansible_user=admin ansible_password=Cpwins!1

[checkpoint_gw]
10.0.1.10 ansible_user=admin ansible_password=Cpwins!1

[checkpoint:vars]
ansible_python_interpreter=/opt/CPsuite-R81.20/fw1/Python/bin/python3.7
```



6. Run the test again but this time run it against **all Check Point hosts** (use the group *checkpoint*).

```
admin@orchestrator:~$ ansible checkpoint -i hosts -m ping
10.0.1.10 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
10.0.1.100 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
```

7. Now that we have many groups and hosts in the inventory, try to run a connectivity check against all hosts. (Note that when the interpreter is not configured, it is displayed as discovered.)

```
admin@orchestrator:~$ ansible all -i hosts -m ping
10.0.3.61 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
10.0.3.62 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
10.0.1.10 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
10.0.1.100 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
```

- Notice that we typed the addresses for the Check Point devices twice, once in the group **checkpoint** and later in the separate groups. The better approach is to create subgroups.
- To achieve that, we can use the `children` keyword.
- The format is `[group_name:children]`.

Exercise 2: Using Custom Inventory in Playbooks

We will now use ansible playbooks to run operations on the target hosts and apply the changes to specific hosts in our inventory.

1. Create a simple playbook to create a simple file on the Ubuntu machines. Chose a name for this Playbook, `create_file.yml`.

```

---
- name: use file module
  hosts: ubuntu
  tasks:

    - name: create a file
      file:
        dest: /tmp/lab2
        state: touch
...

```

2. Run the playbook using the command:

```
ansible-playbook -i hosts create_file.yml
```

```

admin@orchestrator:~$ ansible-playbook -i hosts create_file.yml

PLAY [use file module] *****

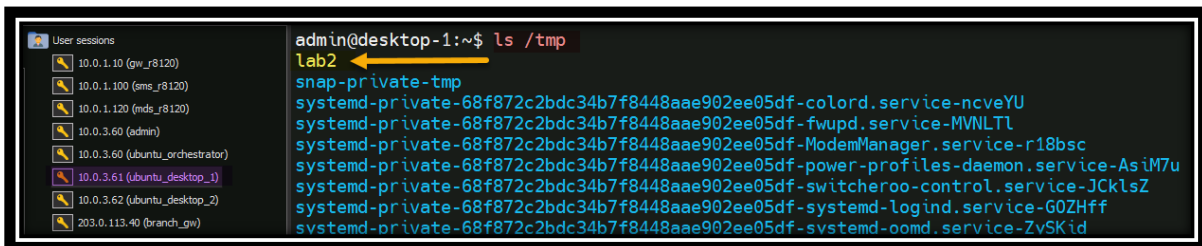
TASK [Gathering Facts] *****
ok: [10.0.3.61]
ok: [10.0.3.62]

TASK [create a file] *****
changed: [10.0.3.62]
changed: [10.0.3.61]

PLAY RECAP *****
10.0.3.61      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
10.0.3.62      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

```

3. Connect to the Ubuntu nodes and verify that the file was created.



```

admin@desktop-1:~$ ls /tmp
lab2
snap-private-tmp
systemd-private-68f872c2bdc34b7f8448aae902ee05df-colord.service-ncveYU
systemd-private-68f872c2bdc34b7f8448aae902ee05df-fwupd.service-MVNLTl
systemd-private-68f872c2bdc34b7f8448aae902ee05df-ModemManager.service-r18bsc
systemd-private-68f872c2bdc34b7f8448aae902ee05df-power-profiles-daemon.service-AsiM7u
systemd-private-68f872c2bdc34b7f8448aae902ee05df-switcheroo-control.service-JCklsZ
systemd-private-68f872c2bdc34b7f8448aae902ee05df-systemd-logind.service-G0ZHzf
systemd-private-68f872c2bdc34b7f8448aae902ee05df-systemd-oomd.service-ZvSKid

```

4. To delete the file, we need to change the state from `touch` to `absent`. Refer to the module documentations for more details on how to control the module options.

https://docs.ansible.com/ansible/latest/collections/ansible/builtin/file_module.html

```

---
- name: use file module
  hosts: ubuntu
  tasks:

    - name: create a file
      file:
        dest: /tmp/lab2
        state: absent
...

```

5. Run the playbook again and make sure the files were deleted from both Ubuntu nodes.
6. It is recommended to make the file state as a variable as this entry will change. Change the playbook to have the state entered as a variable.

```

---
- name: use file module
  hosts: ubuntu
  tasks:
    - name: create a file
      file:
        dest: /tmp/lab2
        state: '{{file_state}}'
...

```

- Now that the state can be provided as a variable, we can provide it in the hosts file or directly when we run the playbook from CLI. Use the following command to create the files again:

```
ansible-playbook -i hosts create_files.yml -e file_state=touch
```

```

admin@orchestrator:~$ ansible-playbook -i hosts create_file.yml -e file_state=touch

PLAY [use file module] *****

TASK [Gathering Facts] *****
ok: [10.0.3.61]
ok: [10.0.3.62]

TASK [create a file] *****
changed: [10.0.3.61]
changed: [10.0.3.62]

PLAY RECAP *****
10.0.3.61      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
10.0.3.62      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

```

- Notice that we provided the variable value with `-e` flag.

- Try the other method and provide the variable value in the inventory file.

```

admin@orchestrator:~$ cat hosts
[ubuntu]
10.0.3.61
10.0.3.62

[ubuntu:vars]
ansible_user=admin
ansible_ssh_pass=0p1g67323k
file_state=absent

[checkpoint:children]
checkpoint_mgmt
checkpoint_gw

[checkpoint_mgmt]
10.0.1.100 ansible_user=admin ansible_password=Cpwins!1

[checkpoint_gw]
10.0.1.10 ansible_user=admin ansible_password=Cpwins!1

[checkpoint:vars]
ansible_python_interpreter=/opt/CPsuite-R81.20/fw1/Python/bin/python3.7

```

- Run the playbook and make sure the files were deleted.

- Notice that we did not provide any variables in the command itself however, ansible discovered the value from the inventory file.

10. Remove the previous variable from the inventory file. Now create the variable inside the playbook using the format below (before the tasks execution start!).

```
---
- name: use file module
  hosts: ubuntu
  vars:
    file_state: touch
  tasks:
    - name: create a file
      file:
        dest: /tmp/lab2
        state: '{{file_state}}'
...
```

11. Run the playbook and confirm the file was created successfully.

```
admin@orchestrator:~$ ansible-playbook -i hosts create_file.yml

PLAY [use file module] *****

TASK [Gathering Facts] *****
ok: [10.0.3.62]
ok: [10.0.3.61]

TASK [create a file] *****
changed: [10.0.3.61]
ok: [10.0.3.62]

PLAY RECAP *****
10.0.3.61      : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
10.0.3.62      : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

12. Try to add the variable to the command “`ansible-playbook -i hosts create_file.yml -e file_state=absent`” while leaving the variable defined in the playbook. Was the file deleted or created? Which variables had the higher priority?
13. Change the state to absent and run the playbook to clean up your environment.

End of Lab 2