Ansible Palo Alto Automation Immersion Day Workshop

Overview

You will use Ansible commands and playbooks to explore and reconfigure a PaloAlto in a virtual environment. Note, that even though we are using a virtual environment, this is not a requirement, everything we do, can be done on physical devices.

The Immersion Day is meant as a follow on to the Network Automation Immersion Day. Some content will be review but in respect to using Ansible with PaloAlto. While you can go through these labs there may be concepts that were covered previously that we will not cover as it is expected that these concepts are already understood. Throughout the labs we hope to share some additional features, elements, good and bad practices, and patterns to using Ansible for Automation.

You will be required to modify some files during this workshop. You will not be required to write your own playbooks as this would require much more time. The playbooks used are open source and thus free to use and modify. Writing playbooks and running them in a test environment is one of the best ways to learn.

Note: Assume the output shown in the examples below will be different to yours.

Part 1: Getting setup

Overview

- Fork the lab github repository to your own repository so you can edit and modify.
- Accessing the jump station
- Downloading your forked repository to the jump station

Fork the Sirius ansible networking GitHub repository

Step 1

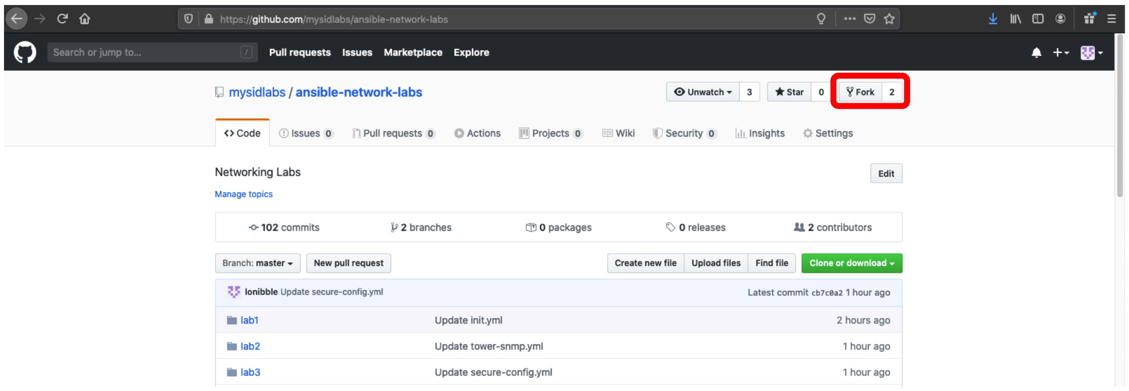
Login in to Github at https://github.com

Step 2

Go to https://github.com/mysidlabs/ansible-pan-labs

Step 3

Click on the Fork button in upper right.



lonibble Update secure-config.yml — Latest commit cb7c0a2 1 hour ago

lab1	Update init.yml	2 hours ago
lab2	Update tower-snmp.yml	1 hour ago
lab3	Update secure-config.yml	1 hour ago

Note: Once forked you can modify all files within GitHub

Connect to the Jump Host

Step 1

SSH to the jump station at jump.mysidlabs.com

For MacOS or Linux users the following is an example using the terminal:

```
$ ssh <<siduserID>>@jump.mysidlabs.com
```

```
Ex.  $ssh siduser101@jump.mysidlabs.com
```

You may get the following message, type **yes** at the prompt:

```
The authenticity of host 'jump.mysidlabs.com (3.132.28.93)' can't be established.
ECDSA key fingerprint is SHA256: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Are you sure you want to continue connecting (yes/no/[fingerprint])? Yes
Warning: Permanently added 'jump.mysidlabs.com,3.132.28.93' (ECDSA) to the list of known hosts.
```

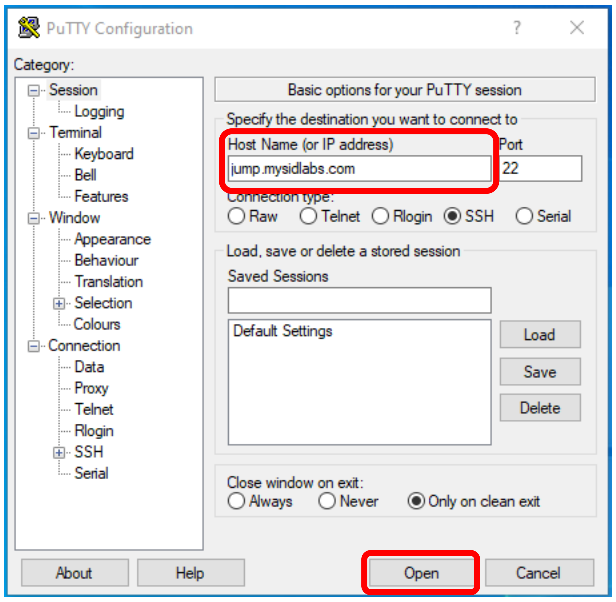
Note: You can remove from known hosts when workshop is completed.

When prompted for your password type in the password the instructor provides

password: *******

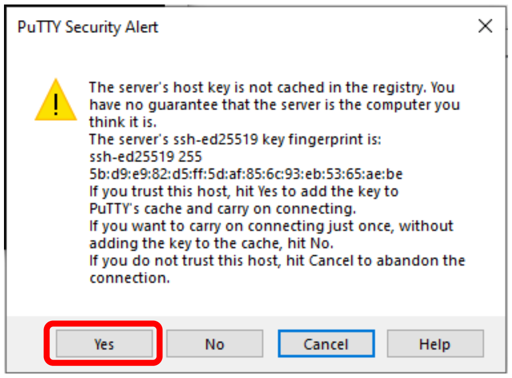
For Windows users the following is an example using Putty:

Type jump.mysidlabs.com in the Host Name box and click the Open button



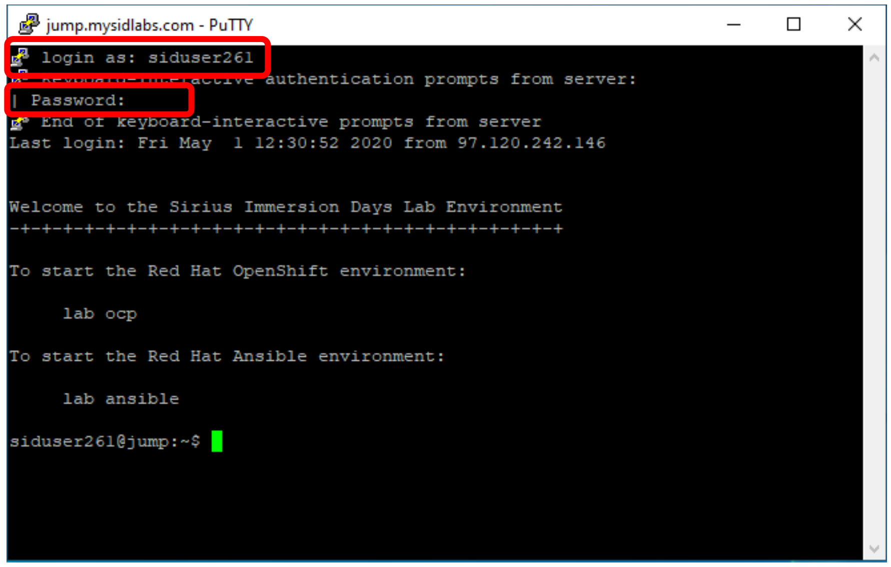
Host Name (or IP address): jump.mysidlabs.com
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Close window on exit: Always Never [✓ Only on clean exit]

Click the Yes button to accept the ssh key


PuTTY Security Alert

The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's ssh-ed25519 key fingerprint is:
ssh-ed25519 255
5b:d9:e9:82:d5:ff:5d:af:85:6c:93:eb:53:65:ae:be
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.

Type in username and password in the terminal screen at the appropriate prompts


jump.mysidlabs.com - PuTTY

```
 login as: siduser261
 Keyboard-interactive authentication prompts from server:
| Password:
 End of keyboard-interactive prompts from server
Last login: Fri May  1 12:30:52 2020 from 97.120.242.146


Welcome to the Sirius Immersion Days Lab Environment
-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

To start the Red Hat OpenShift environment:

     lab ocp

To start the Red Hat Ansible environment:

     lab ansible

siduser261@jump:~$
```

Download repository to jump station

Step 1

Your terminal prompt should change to something like the following:

```
siduser250@jump:~$
```

Step 2

Type in 'TAG=1.3.4 lab ansible' at the promt:

```
siduser101@jump:~$ TAG=1.3.4 lab ansible
```

Step 3

Your terminal prompt should change to something like the following:

```
siduser250@toolkit ~ #
```

Step 4

Clone your repository

```
siduser250@toolkit ~ # git clone https://github.com/<<YOUR_GITHUB_USER>>/ansible-pan-labs
```

Tip	The usage of git becomes very important to "infrastructure as code". Everything resides in github including your changes. If you lose connection from the jump box, the repository will be deleted automatically. All you need to do is clone your repository and you are back to where you were.

Step 5

You should now see the repository in your directory

```
siduser250@toolkit ~ # ls
ansible-pan-labs     dev
```

Step 6

Move into the ansible-network-labs directory

```
siduser250@toolkit ~ # cd ansible-pan-labs
siduser250@toolkit ~/ansible-pan-labs #
```

Additional Information

You can now explore the labs directory

cd = change directory

ls = list contents

pwd = display current working directory

cat = display file

nano or vim = file editor

tree = display file structure from current directory

Connect to Palo Alto via SSH and set username/password

Your student Palo Alto Firewall does not have a username and password configured yet. Access the firewall and set a username and password. Replace <ID> with your student number.

Step 1

Connect to FW via ssh

```
siduser101@toolkit ~/ansible-pan-labs # ssh -i ~/.ssh/network-key.pem admin@siduser<ID>.pan.mysidlabs.com
load pubkey "/home/siduser101/.ssh/network-key.pem": invalid format
The authenticity of host 'siduser101.pan.mysidlabs.com (3.133.142.207)' can't be established.
RSA key fingerprint is SHA256:tfEPSNM5pDbgEOEKv5H059pu1uK8l5T2QjcLIDU03eE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'siduser101.pan.mysidlabs.com,3.133.142.207' (RSA) to the list of known hosts.
Welcome admin.

admin@PA-VM>
```

Step 2

Go into configuration mode

```
admin@PA-VM> configure
```

Step 3

Configure new management user

```
admin@PA-VM# set mgt-config users siduser<ID> password
Enter password   :
Confirm password :
```

Step 4

Give new user admin permissions

```
admin@PA-VM# set mgt-config users siduser<ID> permissions role-based superuser yes
```

Step 5

Save configuration

```
admin@PA-VM# commit

Commit job 2 is in progress. Use Ctrl+C to return to command prompt
```

```
.......55%98%..............100%
Configuration committed successfully
admin@PA-VM#
```

Step 6

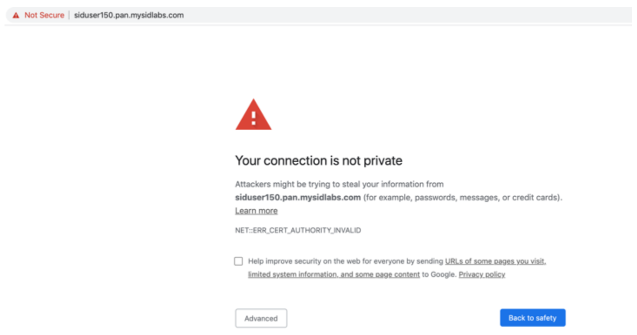
Type **exit** twice to disconnect from your student Palo Alto Firewall

```
admin@PA-VM# exit
Exiting configuration mode
admin@PA-VM> exit
Connection to siduser101.pan.mysidlabs.com closed.
siduser101@toolkit ~/ansible-pan-labs #
```

Connect to Palo Alto via HTTPS with new username and password

Step 1

Open a web browser and connect to https://siduser<ID>.pan.mysidlabs.com



Not Secure | siduser150.pan.mysidlabs.com

Your connection is not private

Attackers might be trying to steal your information from **siduser150.pan.mysidlabs.com** (for example, passwords, messages, or credit cards). Learn more

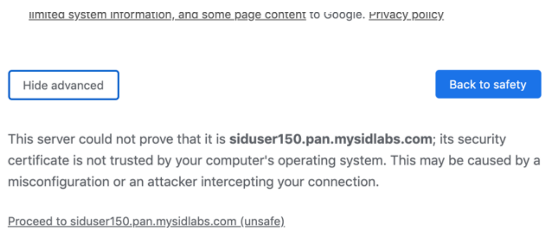
NET::ERR_CERT_AUTHORITY_INVALID

☐ Help improve security on the web for everyone by sending URLs of some pages you visit, limited system information, and some page content to Google. Privacy policy

Advanced | Back to safety

Step 2

Click Advanced



limited system information, and some page content to Google. Privacy policy

Hide advanced | Back to safety

This server could not prove that it is **siduser150.pan.mysidlabs.com**; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

Proceed to siduser150.pan.mysidlabs.com (unsafe)

Step 3

Click "Proceed to siduser<ID>.pan.mysidlabs.com (unsafe)

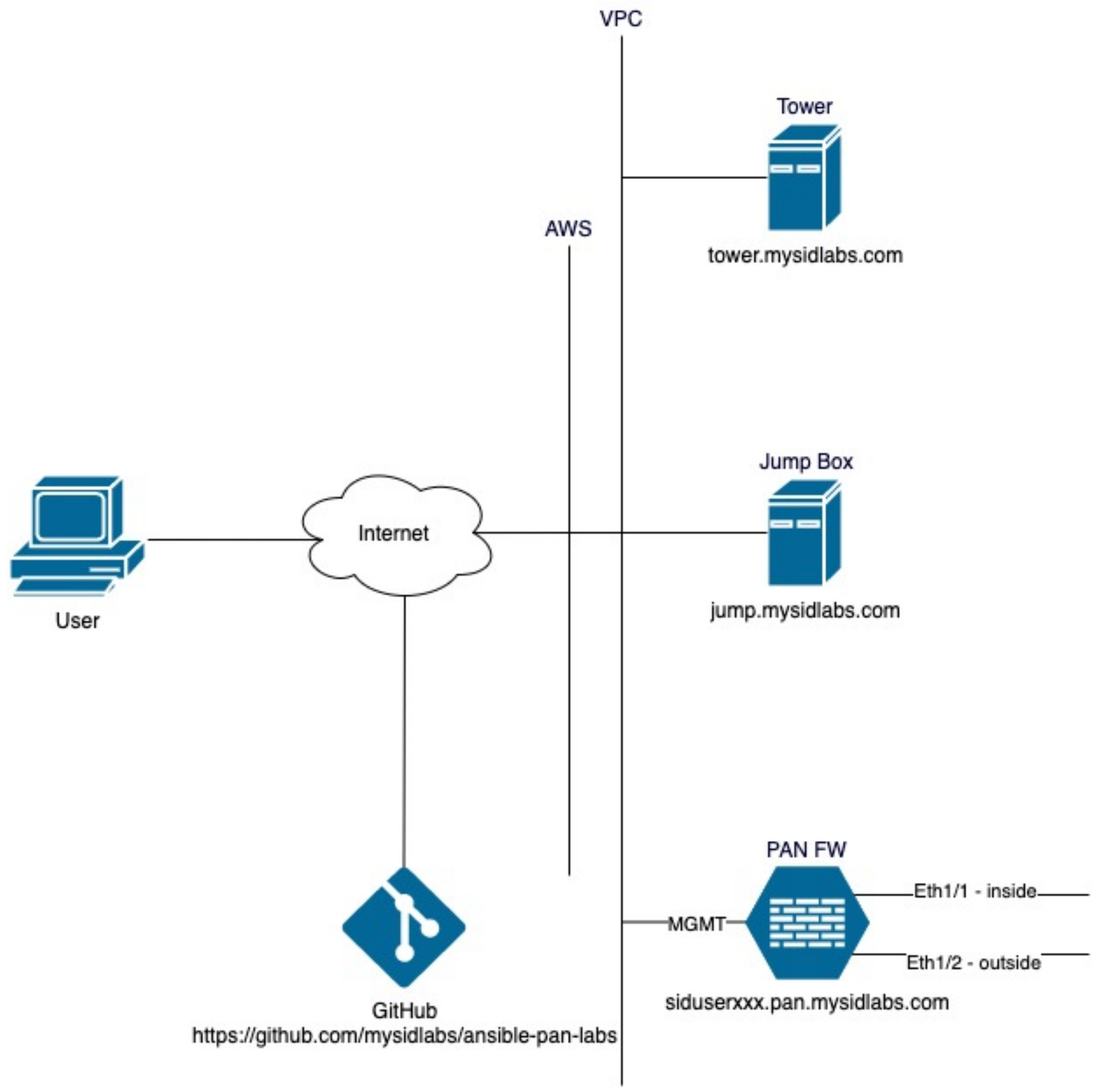
Step 4

Login with your credentials, verify you can see the PAN dashboard

Part 2: Basic PaloAlto labs

Topology

The topology is simple for the sake of learning some ansible basics. The diagram below is an example, the XXX in the hostname is your Student ID. If you are student 199 then the hostname for the PaloAlto would be siduser199.pan.mysidlabs.com.



Lab 1.0: Explore the lab environment

Step 1

Make sure you are in the ansible-pan-labs folder

```
siduser250@toolkit ~/ansible-pan-labs # pwd
/home/siduser250/ansible-pan-labs
```

If you are not, change to the ansible-pan-labs directory

```
siduser250@toolkit ~ # cd ~/ansible-pan-labs/
```

Step 2

Run the ansible command with the --version command to look at what is configured:

```
siduser250@toolkit ~/ansible-pan-labs # ansible --version
ansible 2.9.13
    config file = /home/siduser250/ansible-pan-labs/ansible.cfg
    configured module search path = ['/home/siduser250/.ansible/plugins/modules',
    '/usr/share/ansible/plugins/modules']
    ansible python module location = /usr/lib/python3.8/site-packages/ansible
    executable location = /usr/bin/ansible
    python version = 3.8.5 (default, Aug 12 2020, 10:23:39) [GCC 9.2.1 20190827 (Red Hat 9.2.1-1)]
```

This command gives you information about the version of Ansible, location of the executable, version of Python, search path for the modules and location of the ansible configuration file.

Step 3

Use the cat command to view the contents of the ansible.cfg file.

```
siduser250@toolkit ~/ansible-pan-labs # cat ansible.cfg
[defaults]
deprecation_warnings      = False
gathering                 = explicit
retry_files_enabled       = False
inventory                 = ~/ansible-pan-labs/hosts
connection                = smart
timeout                   = 60
forks                     = 50
host_key_checking         = False
collections_paths         = ~/.ansible/collections
```

```
[ssh_connection]
ssh_args                  = -o ControlMaster=auto -o ControlPersist=30m
scp_if_ssh                = True

[paramiko_connection]
host_key_auto_add         = True

[persistent_connection]
connect_timeout           = 60
command_timeout           = 60
```

Note: the following parameters within the ansible.cfg file:

inventory: shows the location of the ansible inventory being used

collections_paths: shows the location of any installed non default collections

Step 4

Many ansible playbooks contain modules that are not included in Ansible Engine by default. Palo Alto firewalls are one such set of modules. (Starting in version 2.10 all modules will need to be installed as collections. This allows for a slimmed down codebase using only modules you need reducing bloat.)

Go to the docs site for the Palo Alto Ansible Galaxy Collection: https://galaxy.ansible.com/paloaltonetworks/panos

Install the collection:

```
siduser150@toolkit ~/ansible-pan-labs # ansible-galaxy collection install paloaltonetworks.panos
Process install dependency map
Starting collection install process
Installing 'paloaltonetworks.panos:2.2.0' to '/home/siduser150/.ansible/collections/ansible_collections/paloaltonetworks/panos'
siduser150@toolkit ~/ansible-pan-labs #
```

If desired explore the collection, which is a list of module names coded in python.

```
siduser150@toolkit ~/ansible-pan-labs # tree ~/.ansible/collections
```

```
siduser150@toolkit ~/ansible-pan-labs # cat
~/.ansible/collections/ansible_collections/paloaltonetworks/panos/plugins/modules/panos_facts.py
```

Tip	Modules are just python code created by the community. If you cannot locate a module, you could write your own if you had the inclination to do so.

Using Ansible Vault

Step 1

Look at the all.yml group variable file. Notice it is encrypted.

```
siduser150@toolkit ~/ansible-pan-labs # cat group_vars/all.yml
$ANSIBLE_VAULT;1.1;AES256
37646633316539343132663961316431323234383262653935393939343161366466333262626134
3939396462623137643934316165653865393131643763390a393039306337323638323362613830
34313765343462383533363936316231346231303030373830646638353066373632373766623537
6634636330663663640a653730343433636431353364626138653532626131366566636439666466
66653732653764616436306439363165656265306336653063343465313235393237323938653938
33356633633130626663663264643030336261363333961643336313433653863623730353263262
65396433643062326136383633343733383164633939643465623834663932366632396136303066
33626334386436323031306336303135323337626432633863313431396431353639383933383563
39643665653733633035653132633831393539386166613933323736633164643063
```

Step 2

Decrypt the all.yml file and look at it again. Notice it is a variables file containing passwords.

```
siduser150@toolkit ~/ansible-pan-labs # ansible-vault decrypt group_vars/all.yml
Vault password:  password
Decryption successful
siduser150@toolkit ~/ansible-pan-labs # cat group_vars/all.yml
provider:
  username: 'siduser250'
```

```
password: 'Spa2010!'

ip_address: 'siduser250.pan.mysidlabs.com'
```

Step 3

Modify the username, password and address with the appropriate settings for your lab using nano or vi.

```
siduser101@toolkit ~/ansible-pan-labs # nano group_vars/all.yml
```

Tip	Using Nano: arrow keys move the cursor. Make edits normally. When done, "<control>X" will exit, "Y" will save and enter will accept the default file name. Using VI: arrow keys move the cursor. Press "i" to make edits, make edits normally. Press "<esc>" to exit edit mode. Type ":wq!" to save and exit. Pro Tip: ansible-vault edit command will default to VI text editor

Step 4

Encrypt the all.yml file. Enter a new password.

```
siduser150@toolkit ~/ansible-pan-labs # ansible-vault encrypt group_vars/all.yml

New Vault password:

Confirm New Vault password:

Encryption successful
```

Step 5

Check the file is encrypted.

```
siduser150@toolkit ~/ansible-pan-labs # cat group_vars/all.yml

$ANSIBLE_VAULT;1.1;AES256

66653734323765386638666135373238653237333966656638366536356266303533393031316163

6436636631393239316131636231313862313866643635360a653637663734646336353839316362

66353661633732383463363532373462626663323764356365346565383062383836646538373163

3764373364336161320a653536326139616139383232663731393432666236636165643434666263

36333764626462323061613738336536323661336331636339643561353363646538313066396264

30613434303637343136353233623466656233613661373037623032303935633334623365636337

35363632666434333030343137373439363163323232393930653764613762346463393562356639

30613030666264346563646665336639386664346132646136306231633538646239343735613137

36313862653966323936666239333363663432336434333732343663373036613139
```

Step 6

Use view command to unencrypt and view the file.

```
siduser150@toolkit ~/ansible-pan-labs # ansible-vault view group_vars/all.yml
provider:
  username: 'siduser118'
  password: 'P2ssw0r>!'
  ip_address: 'siduser118.pan.mysidlabs.com'
```

Step 7

Use rekey command to change the file encryption key.

```
siduser150@toolkit ~/ansible-pan-labs # ansible-vault rekey group_vars/all.yml
Vault password:
New Vault password:
Confirm New Vault password:
Rekey successful
```

Step 8

Use the "- - help" command to see what other options are available. Also review the ansible-vault documentation.
https://docs.ansible.com/ansible/latest/user_guide/vault.html#vault-ids-and-multiple-vault-passwords

```
siduser150@toolkit ~/ansible-pan-labs # ansible-vault --help
```

Lab 1.1: Gather data from PaloAlto

Step 1:

Look at the file called 1.1_palo_facts.yml.

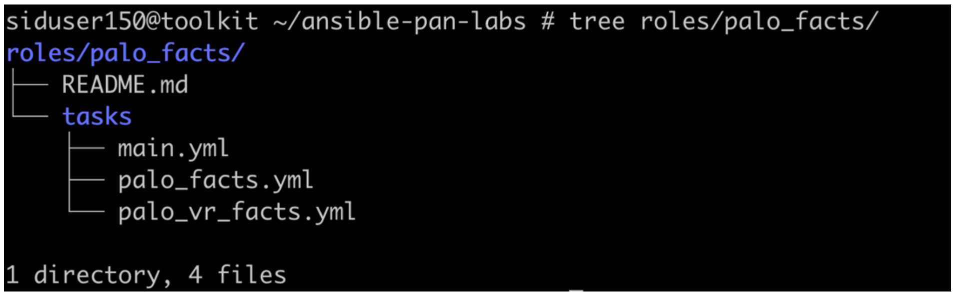
```
siduser250@toolkit ~/ansible-pan-labs # cat 1.1_palo_facts.yml
 ---
- hosts: localhost
  connection: local
  gather_facts: False


  tasks:
    - include_role:
        name: palo_facts
...
```

Step 2:

Ansible playbooks are YAML files. YAML is a structured encoding format that is also extremely human readable. In this case we are using "include_role:" explor the role palo_facts.

siduser150@toolkit ~/ansible-pan-labs # **tree roles/palo_facts/**



```
siduser150@toolkit ~/ansible-pan-labs # tree roles/palo_facts/
roles/palo_facts/
├── README.md
└── tasks
    ├── main.yml
    ├── palo_facts.yml
    └── palo_vr_facts.yml

1 directory, 4 files
```

siduser150@toolkit ~/ansible-pan-labs # **cat roles/palo_facts/tasks/main.yml**

<< output omitted >>

siduser150@toolkit ~/ansible-pan-labs # **cat roles/palo_facts/tasks/palo_facts.yml**

<< output omitted >>

Tip	Notice the FQCN or fully qualified collection name paloaltonetworks.panos.panos_facts. This format is required when using roles with collections that are not installed in engine by default such as Palo Alto collections used in this lab.

```
siduser150@toolkit ~/ansible-pan-labs # cat roles/palo_facts/tasks/palo_vr_facts.yml
<< output omitted >>
```

Step 3:

Run the playbook:

```
siduser250@toolkit ~/ansible-pan-labs # ansible-playbook 1.1_palo_facts.yml --ask-vault-pass
<< output omitted >>
```

Step 4:

Review the output.

Lab 1.2: Adding nodes to PaloAlto

Step 1:

Look at the file called 1.2_palo_initial_setup.yml

```
siduser250@toolkit ~/ansible-pan-labs # cat 1.2_palo_initial_setup.yml
<< output omitted >>
```

explore the role

```
siduser150@toolkit ~/ansible-pan-labs # tree roles/initial_palo_setup/
<< output omitted >>
```

notice that there is a vars directory. This is known as roles vars

```
siduser150@toolkit ~/ansible-pan-labs # cat roles/initial_palo_setup/vars/main.yml
<< output omitted >>
```

Tip	When a variable file exists in the roles directory structure in the vars folder and is called "main.yml" it will be called into the playbook by default, there is no need to reference a specific vars file in the playbook. This is useful because all the vars required for the specific role are located with the role. This can make modular playbook design using roles easier.

NOTE: This role needs some additional work to make it portable for use in production environments but serves our purposes for an AWS lab well.

Step 2:

Run the playbook:

```
siduser250@toolkit ~/ansible-pan-labs # ansible-playbook 1. 2_palo_initial_setup.yml --ask-vault-pass
<< output omitted >>
```

Step 3:

Verifying that the playbook did what you expected. Login to the PaloAlto with your web browser to see what was configured.
https://siduser<<ID>>.pan.mysidlabs.com

Lab 1.3: Adding logging server profile

Step 1:

Look at the file called 1.3_logging_setup.yml

siduser150@toolkit ~/ansible-pan-labs # **cat 1.3_logging_setup.yml**

<< output omitted >>

siduser150@toolkit ~/ansible-pan-labs # **tree roles/palo_log_server/**

<< output omitted >>

siduser150@toolkit ~/ansible-pan-labs # **tree roles/palo_syslog/**

<< output omitted >>

Use github to look at the roles and roles vars

Step 2:

Run the playbook:

siduser250@toolkit ~/ansible-pan-labs # **ansible-playbook 1.3_logging_setup.yml --ask-vault-pass**

<< output omitted >>

Step 3:

Verifying that the playbook did what you expected. Login to the PaloAlto with your web browser to see what was configured.

Lab 1.4: A simple security rule

This step is required for lab 1.5 to work. It adds a deny all rule to the end of the access list which is referenced in the next playbook.

Step 1:

Look at the file called 1.4_simple_security_rule_add.yml

```
siduser250@toolkit ~/ansible-pan-labs # cat 1.4_simple_security_rule_add.yml
```

NOTE: This playbook does not use a role. We did this to illustrate how to use collections in a playbook where roles are not included.

Step 2:

Use cat or github to review the playbook.

Step 3:

Run the playbook:

```
siduser250@toolkit ~/ansible-pan-labs # ansible-playbook 1.4_simple_security_rule_add.yml --ask-vault-pass
<< output omitted >>
```

Step 4:

Verifying that the playbook did what you expected. Login to the PaloAlto with your web browser to see what was configured.

Lab 1.5: Adding rules from a CSV file

Step 1:

Look at the file called 1.5_add_rules_csv.yml

```
siduser250@toolkit ~/ansible-pan-labs # cat 1.5_add_rules_csv.yml
```

Step 2:

Use cat or github to review the roles tasks, roles variables and the roles files.

Note: This playbook pulls variables from a CSV file.

Step 3:

Run the playbook:

```
siduser250@toolkit ~/ansible-pan-labs # ansible-playbook 1.5_add_rules_csv.yml --ask-vault-pass
<< output omitted >>
```

Step 4:

Verifying that the playbook did what you expected. Login to the PaloAlto with your web browser to see what was configured.

Bonus!

Edit the CSV file and add additional rules by running the playbook again.

Lab 1.6: Remove address object

Step 1:

Look at the file called 1.6_remove_address_object.yml

```
siduser250@toolkit ~/ansible-pan-labs # cat 1.6_remove_address_object.yml
```

Step 2:

Use cat or github to review the roles tasks and roles variables

Step 3:

Modify the variable file to a server of your choosing using nano, vi or github

Example

```
remove_addr: 'server_6'
```

Step 4:

Run the playbook:

```
siduser250@toolkit ~/ansible-pan-labs # ansible-playbook 1.6_remove_address_object.yml --ask-vault-pass
<< output omitted >>
```

Step 5:

Verifying that the playbook did what you expected. Login to the PaloAlto with your web browser to see what was configured.

Why is the rule greyed out?

Lab 1.7: Save the configuration

Step 1:

Look at the file called 1.7_palo_backup_configuration.yml

```
siduser250@toolkit ~/ansible-pan-labs # cat 1.7_palo_backup_configuration.yml
```

Step 2:

Review the roles tasks and roles variables with cat or github. The variable file should look like below.

```
---
## Be sure path is writable
backup_path : '~/ansible-pan-labs/'
...
```

Step 3:

Run the playbook:

```
siduser250@toolkit ~/ansible-pan-labs # ansible-playbook 1.7_palo_backup_configuration.yml --ask-vault-pass
<< output omitted >>
```

Step 4:

Use LS note that the backup file exists. Copy the backup file name

Lab 1.8: Restore the configuration

Step 1:

Look at the file called 1.8_palo_restore_configuration.yml

```
siduser250@toolkit ~/ansible-pan-labs # cat 1.8_palo_restore_configuration.yml
```

Step 2:

Review the roles variables with cat or github. The variable file should look like below.

```
---
backup_path : 'home/siduser<ID>/ansible-pan-labs/'
restore_file: 'backup-2021-01-01-00-01.xml'
...
```

Edit the backup_path variable to have your siduser ID.

Edit the restore_file variable to be your saved backup file name.

Step 3:

Run the playbook:

```
siduser250@toolkit ~/ansible-pan-labs # ansible-playbook 1.8_palo_restore_configuration.yml --ask-vault-pass
                        << output omitted >>
```

Success - Congratulations.

Appendix A:

Useful resource links and information

Links:

Ansible Best Practices

https://docs.ansible.com/ansible/latest/user_guide/playbooks_best_practices.html

Ansible Network troubleshooting

https://docs.ansible.com/ansible/latest/network/user_guide/network_debug_troubleshooting.html

Ansible cli_command module information

https://www.ansible.com/blog/deep-dive-on-cli-command-for-network-automation

Variable precedence

https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#variable-precedence-where-should-i-put-a-variable

Additional Notes:

- Remember YAML is very sensitive to correct indentation
- **`Hostvars`** allow us to access meta-data about our inventory hosts.
- The use of an Ansible role is best practice when there is a well-defined scope with a high possibility of re-use.
- If you copy and paste text for a playbook you may get indentation issues. Ansible provides a simple syntax checker, try ansible-playbook --syntax-check backup.yml to verify. A Best Practice is to use a linter, for example ansible-review. Ansible provides excellent online documentation, which is also available from the command line, for example ansible-doc ios_config. For a full list of modules try ansible-doc –l
- There where multiple ways of implementing a playbook where specific tasks or groups of tasks execute against specific hosts. For example, we could have used 1 playbook for configuring every router in the lab utilizing the "when:" statement to ensure specific tasks are only applied to a specific router. Although this is not necessarily following best practices.
- The use of `handlers:` which can be used in any playbook. A handler is a special way of calling a task whenever an action needs to be taken after a previous task. For example, both installing and configuring an application may require a restart. A handler would be notified by both tasks but would only run once when the playbook finishes.