Device Management with Ansible

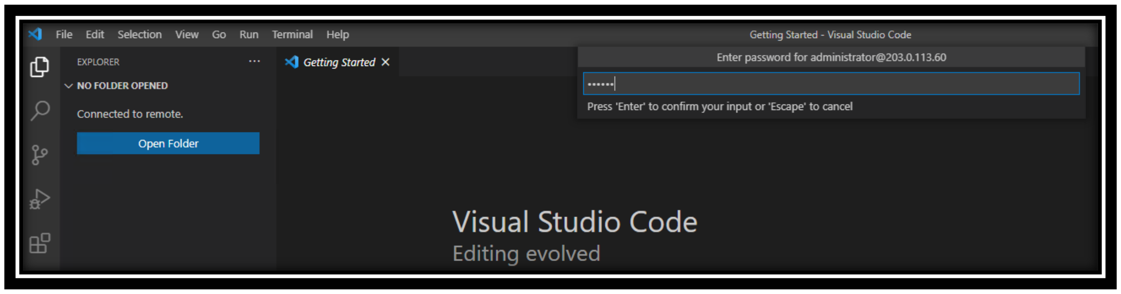
Introduction

Now that we have our inventory ready and we are able to run simple playbooks, we will start using VS code as our editor and write more useful playbooks.

Exercise 1: Using Ansible in VS code

VS code is preinstalled on the windows 10 jump box. Few extensions are also installed such as the remote development extension. Refer to https://code.visualstudio.com/docs/remote/ssh for instruction on how the module was configured.

1. Start VS code and you will see a prompt for the administrator password, Use vpn123.

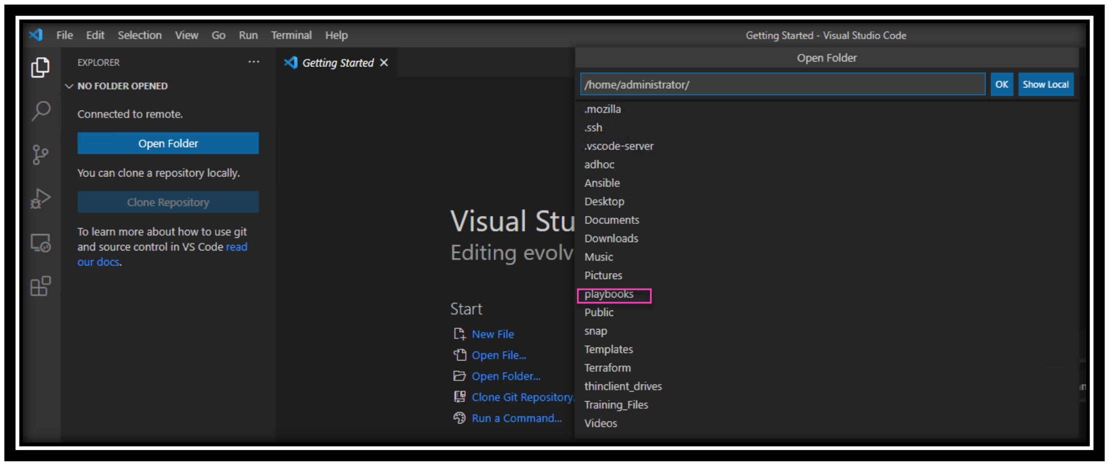


Connected to remote.

Enter password for administrator@203.0.113.60

Press 'Enter' to confirm your input or 'Escape' to cancel

2. Select the playbook folder as our working directory and click OK.



Connected to remote.

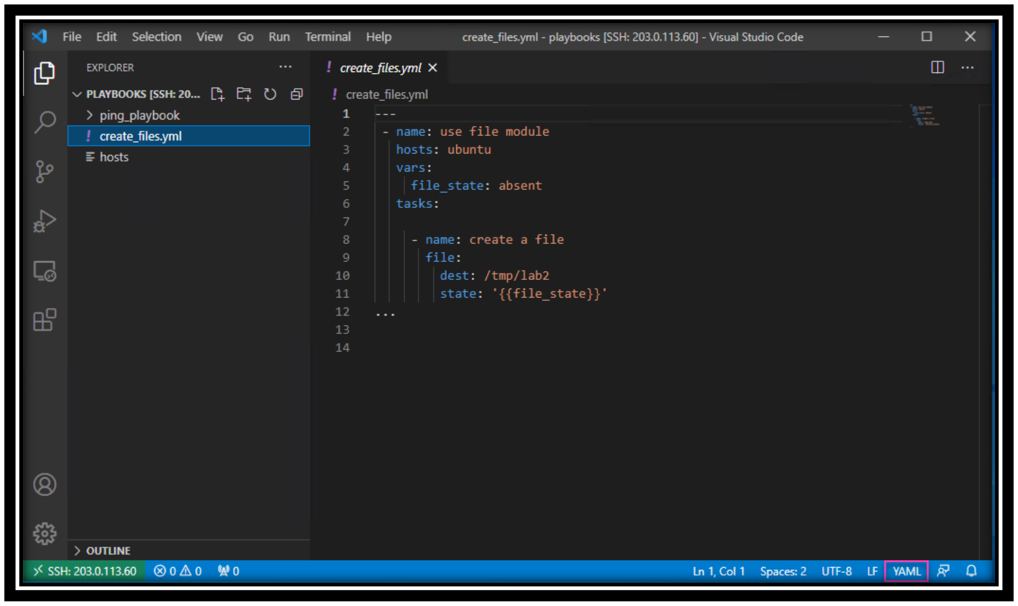
You can clone a repository locally.

To learn more about how to use git and source control in VS Code read our docs.

/home/administrator/

.mozilla
.ssh
.vscode-server
adhoc
Ansible
Desktop
Documents
Downloads
Music
Pictures
playbooks
Public
snap
Templates
Terraform
thinclient_drives
Training_Files
Videos

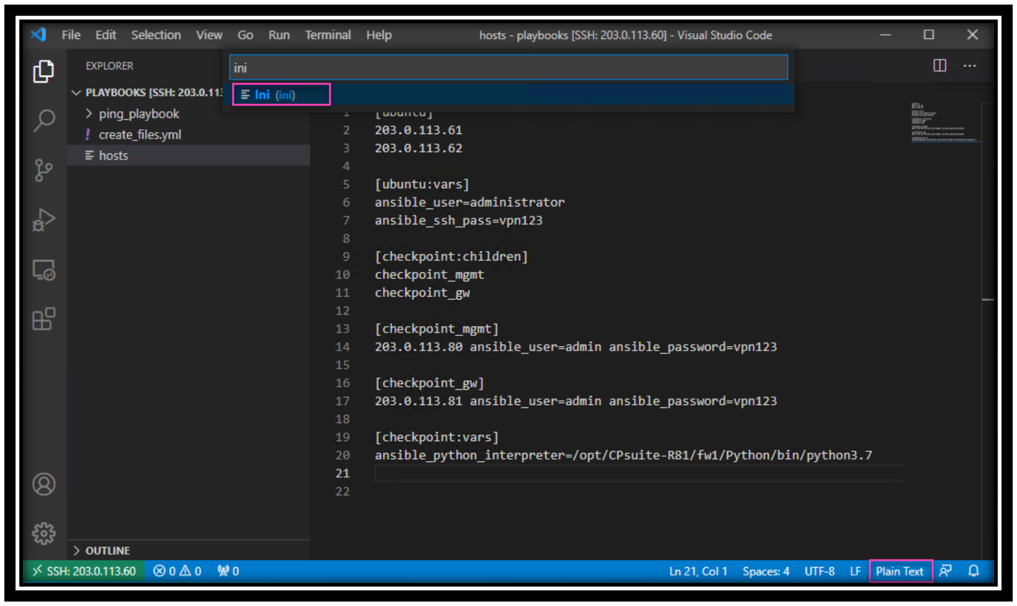
3. Open the existing playbook we created in Lab 2 and notice it is recognized as a YAML file.



File Edit Selection View Go Run Terminal Help
create_files.yml - playbooks [SSH: 203.0.113.60] - Visual Studio Code
EXPLORER
PLAYBOOKS [SSH: 20...
ping_playbook
create_files.yml
hosts
OUTLINE
SSH: 203.0.113.60
Ln 1, Col 1 Spaces: 2 UTF-8 LF YAML

```yaml
---
- name: use file module
  hosts: ubuntu
  vars:
    file_state: absent
  tasks:

    - name: create a file
      file:
        dest: /tmp/lab2
        state: '{{file_state}}'
...
```

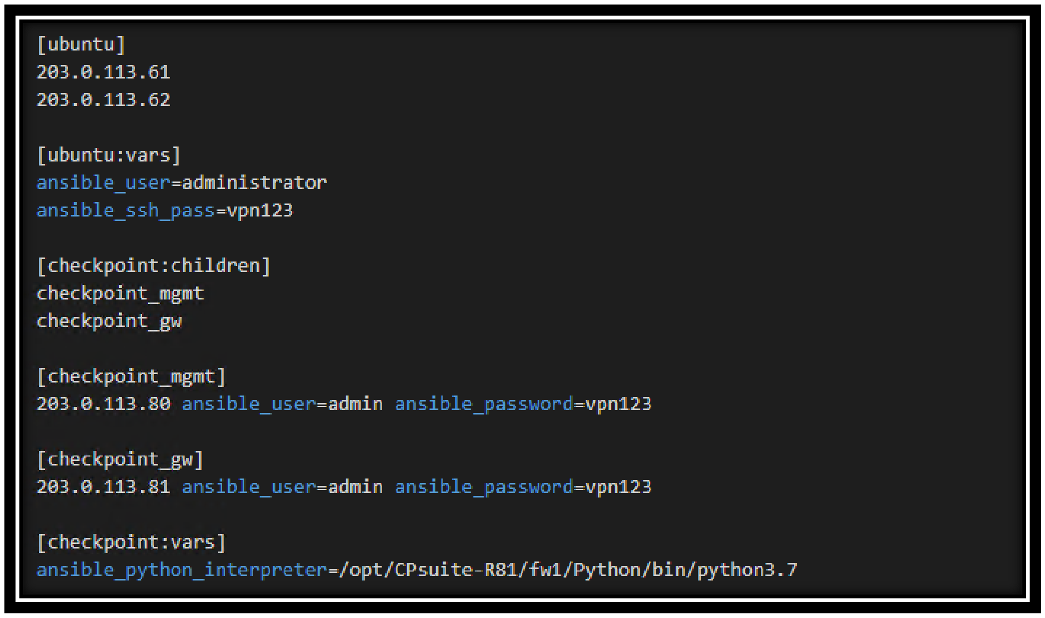
4. Open the inventory file. It is opened as a plain txt file. Click the selection to select the language and use **ini** as the language for the hosts file.



File Edit Selection View Go Run Terminal Help
hosts - playbooks [SSH: 203.0.113.60] - Visual Studio Code
EXPLORER
ini
Ini (ini)
PLAYBOOKS [SSH: 203.0.11:
ping_playbook
create_files.yml
hosts
OUTLINE
SSH: 203.0.113.60
Ln 21, Col 1 Spaces: 4 UTF-8 LF Plain Text

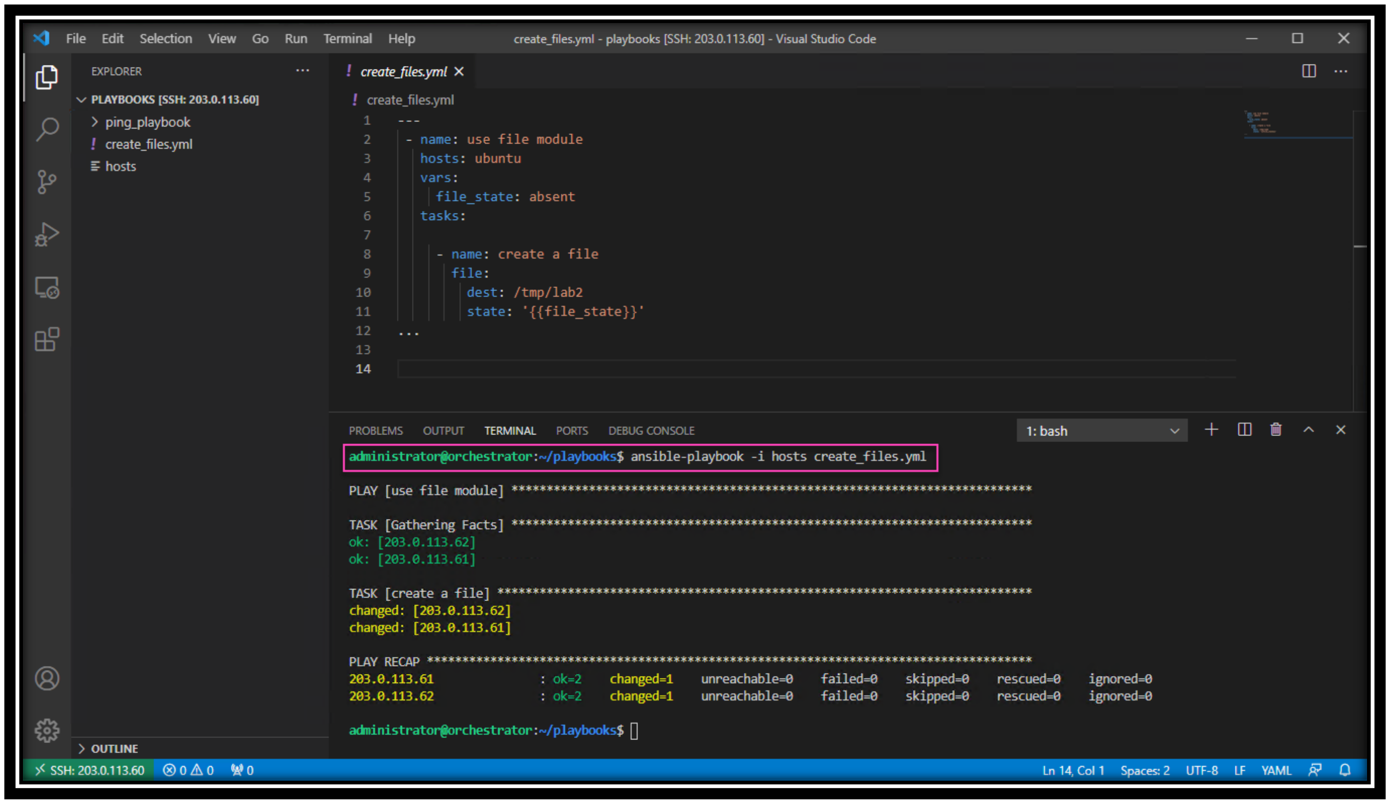
```ini
[ubuntu]
203.0.113.61
203.0.113.62

[ubuntu:vars]
ansible_user=administrator
ansible_ssh_pass=vpn123

[checkpoint:children]
checkpoint_mgmt
checkpoint_gw

[checkpoint_mgmt]
203.0.113.80 ansible_user=admin ansible_password=vpn123

[checkpoint_gw]
203.0.113.81 ansible_user=admin ansible_password=vpn123

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

5. Our inventory is now formatted correctly and the variables should be highlighted. Remember to save the settings with Ctrl+s



```
[ubuntu]
203.0.113.61
203.0.113.62

[ubuntu:vars]
ansible_user=administrator
ansible_ssh_pass=vpn123

[checkpoint:children]
checkpoint_mgmt
checkpoint_gw

[checkpoint_mgmt]
203.0.113.80 ansible_user=admin ansible_password=vpn123

[checkpoint_gw]
203.0.113.81 ansible_user=admin ansible_password=vpn123

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

6. To Open a terminal to out working directory on the ubuntu orchestrator, Use the Ctrl+` keyboard shortcut with the backtick character and run the existing playbook to verify that we can execute playbooks remotely from the VS code.


File Edit Selection View Go Run Terminal Help
create_files.yml - playbooks [SSH: 203.0.113.60] - Visual Studio Code
EXPLORER
PLAYBOOKS [SSH: 203.0.113.60]
ping_playbook
create_files.yml
hosts
OUTLINE
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
1: bash
SSH: 203.0.113.60
Ln 14, Col 1 Spaces: 2 UTF-8 LF YAML

```
---
- name: use file module
  hosts: ubuntu
  vars:
    file_state: absent
  tasks:

    - name: create a file
      file:
        dest: /tmp/lab2
        state: '{{file_state}}'
...
```

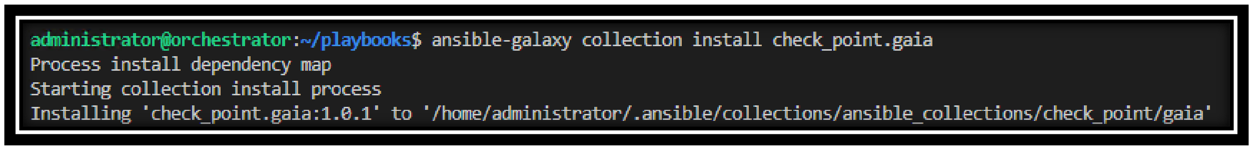
```
administrator@orchestrator:~/playbooks$ ansible-playbook -i hosts create_files.yml

PLAY [use file module] *************************************************************************

TASK [Gathering Facts] *************************************************************************
ok: [203.0.113.62]
ok: [203.0.113.61]

TASK [create a file] ***************************************************************************
changed: [203.0.113.62]
changed: [203.0.113.61]

PLAY RECAP *************************************************************************************
203.0.113.61               : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
203.0.113.62               : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

administrator@orchestrator:~/playbooks$ 
```

Exercise 2: Using the Check Point Gaia collection for Ansible

In this exercise, we will use the Check Point Ansible Gaia Collection to perform simple tasks on a Gaia machine. Only limited operations are available at the time this lab was created.

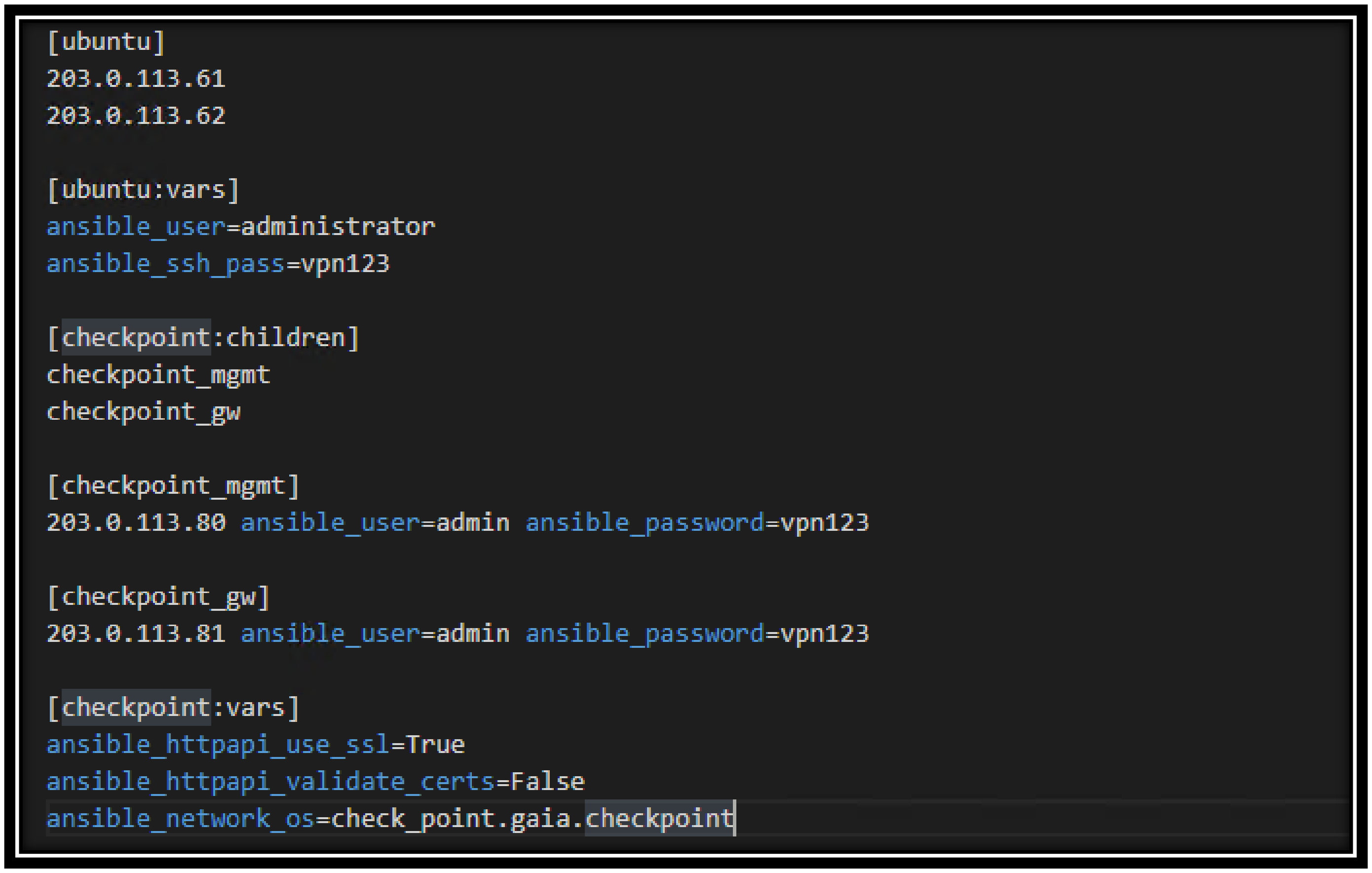
Read the documentation on Ansible galaxy via https://galaxy.ansible.com/check_point/gaia and notice that the Gaia API is required.

1. Before Ansible can use the collection, we need to install it on our orchestrator with the ansible-galaxy module. Use the command:
   ansible-galaxy collection install check_point.gaia



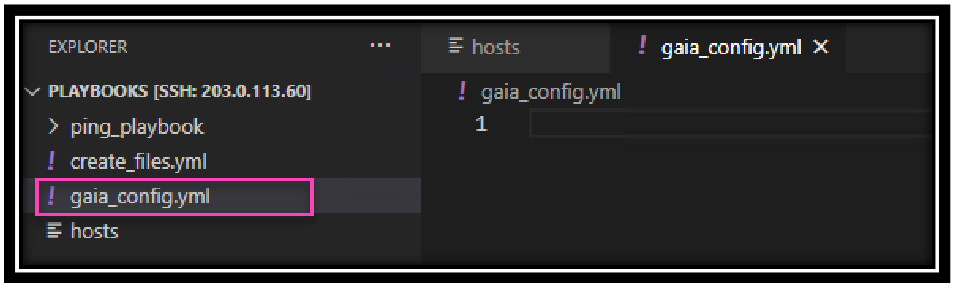
```
administrator@orchestrator:~/playbooks$ ansible-galaxy collection install check_point.gaia
Process install dependency map
Starting collection install process
Installing 'check_point.gaia:1.0.1' to '/home/administrator/.ansible/collections/ansible_collections/check_point/gaia'
```

2. As this module uses the API, we need to edit our inventory to add variable related to the HTTPS connection and certificate validation. Add the following values to the inventory file under the [chechpoint:vars] group.

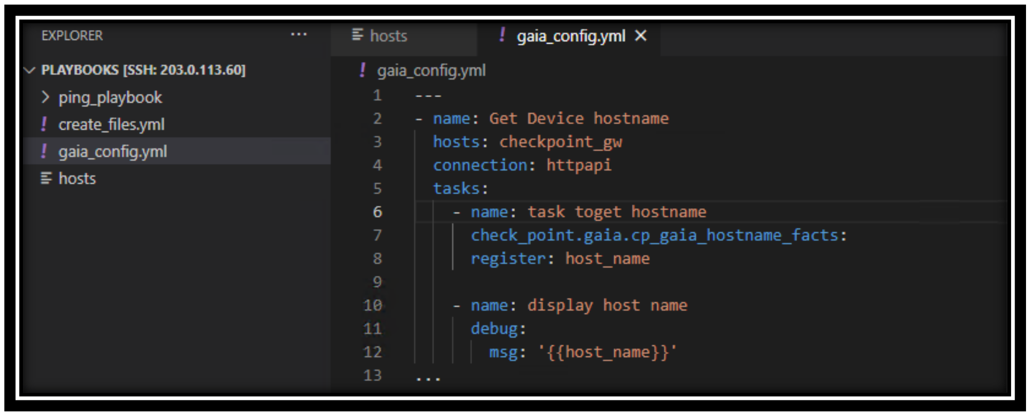


```
[ubuntu]
203.0.113.61
203.0.113.62

[ubuntu:vars]
ansible_user=administrator
ansible_ssh_pass=vpn123

[checkpoint:children]
checkpoint_mgmt
checkpoint_gw

[checkpoint_mgmt]
203.0.113.80 ansible_user=admin ansible_password=vpn123

[checkpoint_gw]
203.0.113.81 ansible_user=admin ansible_password=vpn123

[checkpoint:vars]
ansible_httpapi_use_ssl=True
ansible_httpapi_validate_certs=False
ansible_network_os=check_point.gaia.checkpoint
```

3. From the explorer menu in VS code, add a new file in our working directory and call it gaia_config.yml.

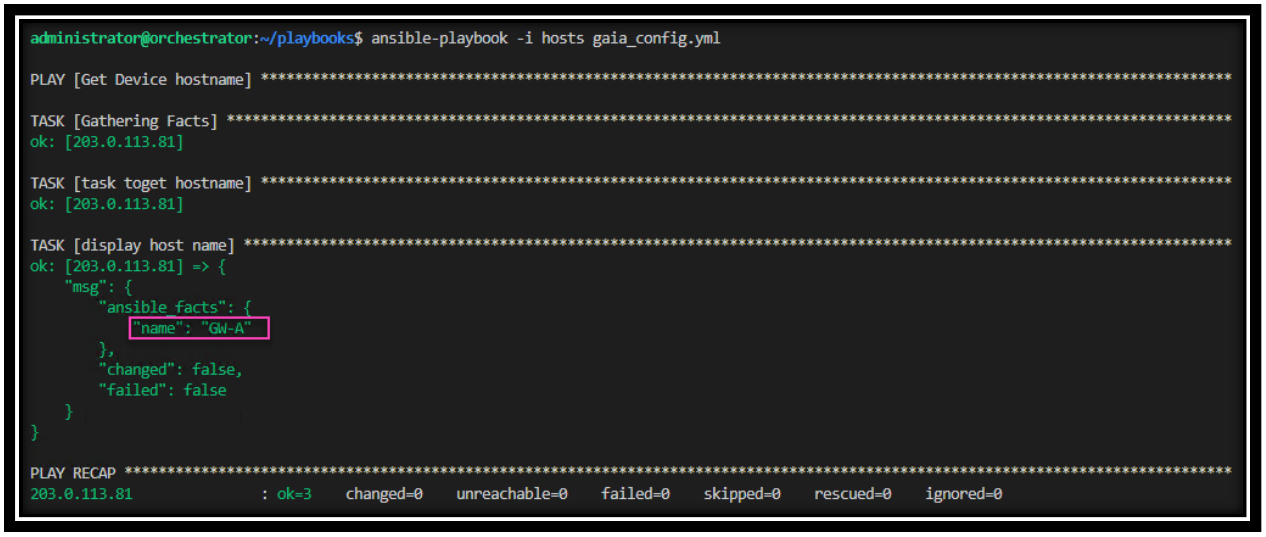


4. Create a simple playbook to get the hostname from our Gaia gateway.



```
---
- name: Get Device hostname
  hosts: checkpoint_gw
  connection: httpapi
  tasks:
    - name: task toget hostname
      check_point.gaia.cp_gaia_hostname_facts:
      register: host_name

    - name: display host name
      debug:
        msg: '{{host_name}}'
...
```

5. Run the playbook and notice that we received the hostname GW-A

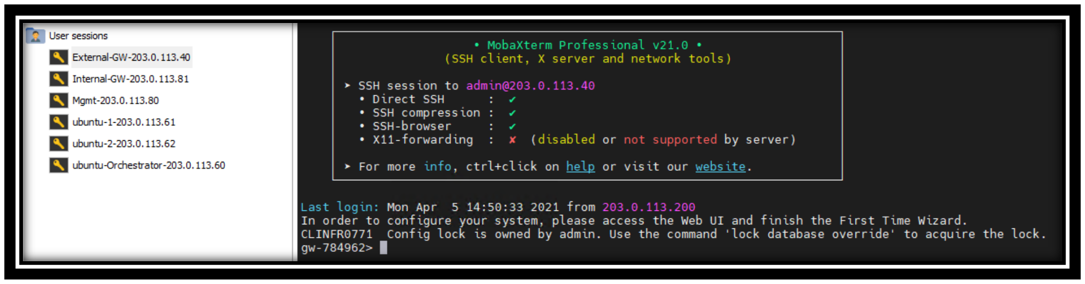


```
administrator@orchestrator:~/playbooks$ ansible-playbook -i hosts gaia_config.yml

PLAY [Get Device hostname] ***********************************************************************************************

TASK [Gathering Facts] ***************************************************************************************************
ok: [203.0.113.81]

TASK [task toget hostname] ***********************************************************************************************
ok: [203.0.113.81]

TASK [display host name] *************************************************************************************************
ok: [203.0.113.81] => {
    "msg": {
        "ansible_facts": {
            "name": "GW-A"
        },
        "changed": false,
        "failed": false
    }
}

PLAY RECAP ***************************************************************************************************************
203.0.113.81               : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Exercise 3: Configuring Gaia with Ansible

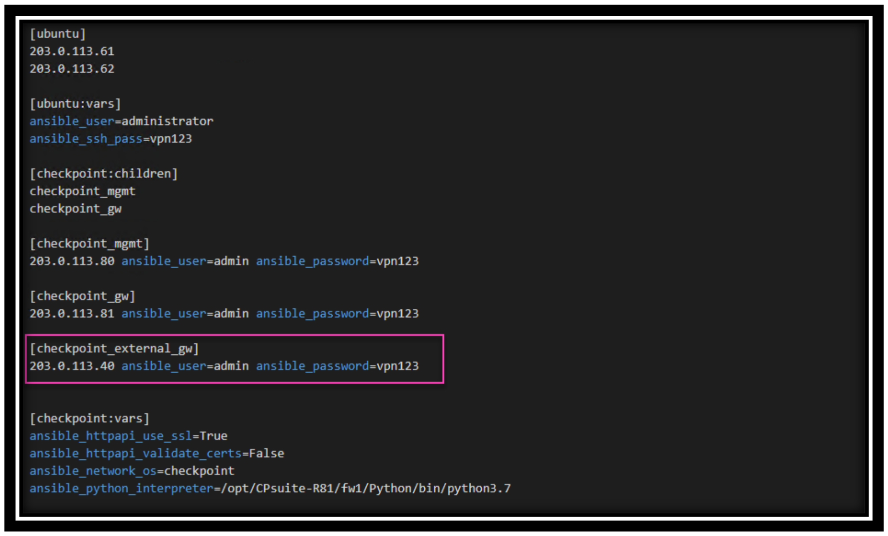
Now that we are familiar with the Gaia collection and the few modules that we can use, we will use other modules to configure a Gaia gateway from scratch after deployment.

1. Use MobaXterm to connect to the External-GW using the bookmarked session. Notice the first tie wizard message.



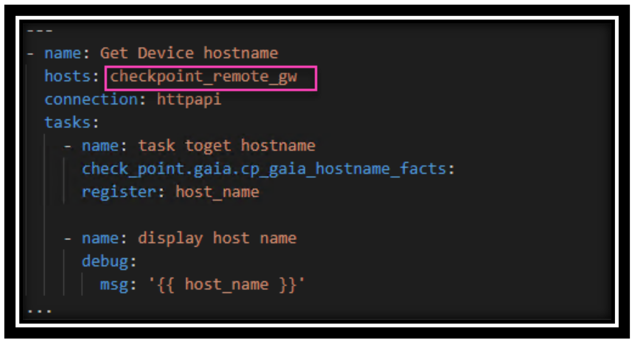
```
              • MobaXterm Professional v21.0 •
           (SSH client, X server and network tools)

 ➤ SSH session to admin@203.0.113.40
   • Direct SSH      :  ✔
   • SSH compression :  ✔
   • SSH-browser     :  ✔
   • X11-forwarding  :  ✘  (disabled or not supported by server)

 ➤ For more info, ctrl+click on help or visit our website.

Last login: Mon Apr  5 14:50:33 2021 from 203.0.113.200
In order to configure your system, please access the Web UI and finish the First Time Wizard.
CLINFR0771  Config lock is owned by admin. Use the command 'lock database override' to acquire the lock.
gw-784962>
```

2. Edit the inventory file and add the external gateway to our inventory



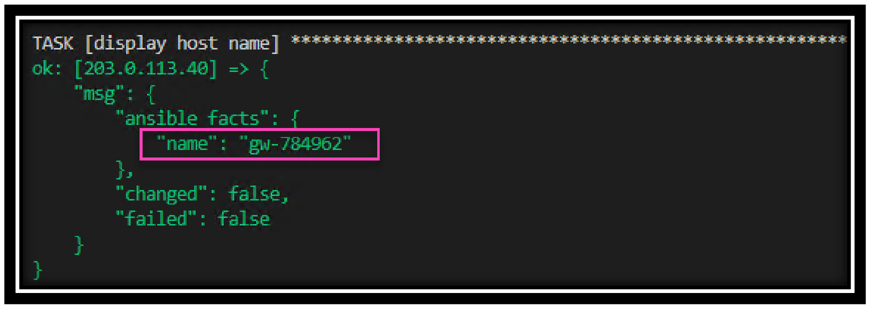
```
[ubuntu]
203.0.113.61
203.0.113.62

[ubuntu:vars]
ansible_user=administrator
ansible_ssh_pass=vpn123

[checkpoint:children]
checkpoint_mgmt
checkpoint_gw

[checkpoint_mgmt]
203.0.113.80 ansible_user=admin ansible_password=vpn123

[checkpoint_gw]
203.0.113.81 ansible_user=admin ansible_password=vpn123

[checkpoint_external_gw]
203.0.113.40 ansible_user=admin ansible_password=vpn123


[checkpoint:vars]
ansible_httpapi_use_ssl=True
ansible_httpapi_validate_certs=False
ansible_network_os=checkpoint
ansible_python_interpreter=/opt/CPsuite-R81/fw1/Python/bin/python3.7
```

3. Edit the playbook and replace the hosts with the external Gateway.



```
---
- name: Get Device hostname
  hosts: checkpoint_remote_gw
  connection: httpapi
  tasks:
    - name: task toget hostname
      check_point.gaia.cp_gaia_hostname_facts:
      register: host_name

    - name: display host name
      debug:
        msg: '{{ host_name }}'
...
```

4. Run the playbook and make sure you are getting the hostname of the remote gateway.

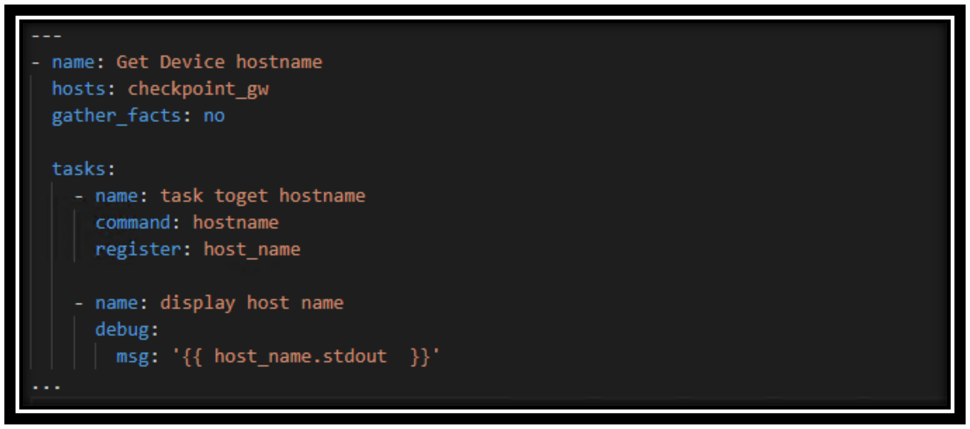


```
TASK [display host name] ********************************************************
ok: [203.0.113.40] => {
    "msg": {
        "ansible_facts": {
            "name": "gw-784962"
        },
        "changed": false,
        "failed": false
    }
}
```

- We can utilize the built-in **command** module to run operations that are not available in the Gaia collection for Ansible. This module runs the commands that we provide without checking the status (non-idempotent). More details can be found on the Ansible documentation portal:

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

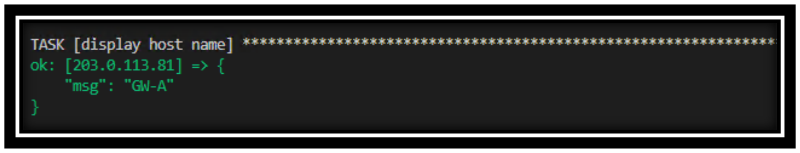
5. Change the playbook and remove the connection plugin line since we will be using the native command module and not the API. Set the host to be the internal gateway checkpoint_gw:



```
---
- name: Get Device hostname
  hosts: checkpoint_gw
  gather_facts: no

  tasks:
    - name: task toget hostname
      command: hostname
      register: host_name

    - name: display host name
      debug:
        msg: '{{ host_name.stdout  }}'
...
```

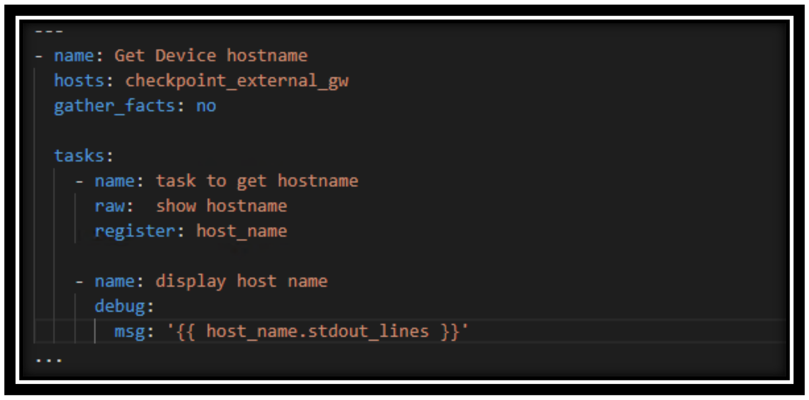
- Note that we are not collecting facts and we are using the built-in commands.
- We are only displaying the stdout from the registered results instead of displaying all the output.

6. Run the playbook and verify that we have the correct results.



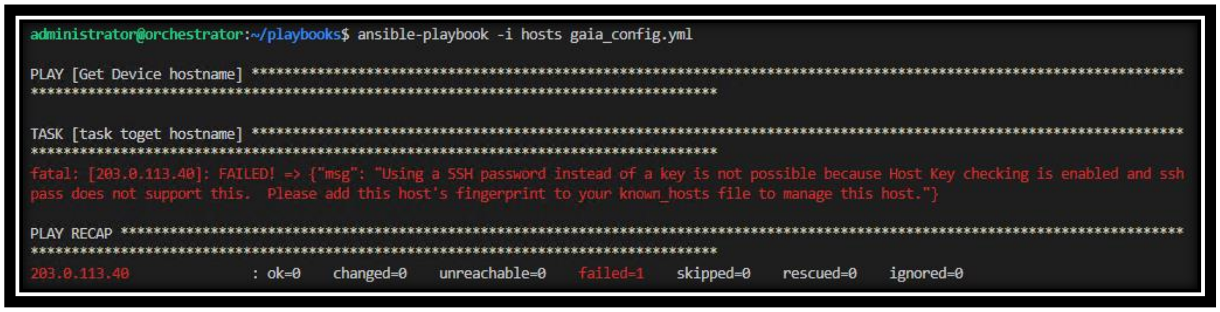
```
TASK [display host name] ********************************************************************
ok: [203.0.113.81] => {
    "msg": "GW-A"
}
```

7. Change the hosts to be the external gateway checkpoint_external_gw



```
---
- name: Get Device hostname
  hosts: checkpoint_external_gw
  gather_facts: no

  tasks:
    - name: task to get hostname
      raw:  show hostname
      register: host_name

    - name: display host name
      debug:
        msg: '{{ host_name.stdout_lines }}'
...
```

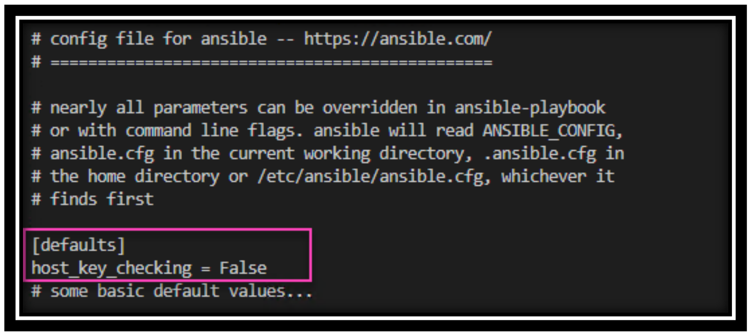
8. Run the playbook and notice that we have an error message.



```
administrator@orchestrator:~/playbooks$ ansible-playbook -i hosts gaia_config.yml

PLAY [Get Device hostname] *****************************************************************************************************
*****************************************************************************************

TASK [task toget hostname] *****************************************************************************************************
*****************************************************************************************
fatal: [203.0.113.40]: FAILED! => {"msg": "Using a SSH password instead of a key is not possible because Host Key checking is enabled and ssh
pass does not support this.  Please add this host's fingerprint to your known_hosts file to manage this host."}

PLAY RECAP *********************************************************************************************************************
*****************************************************************************************
203.0.113.40               : ok=0    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
```

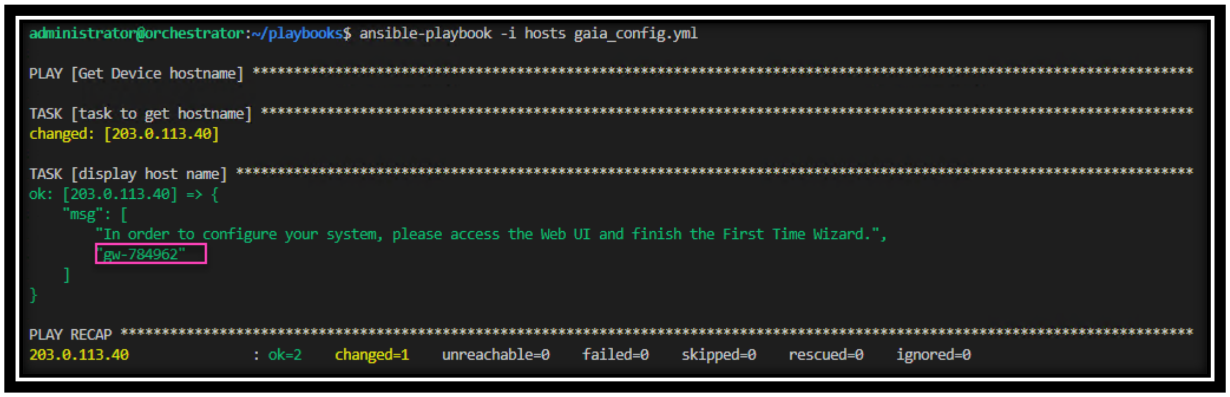
- The external gateway was not configured yet and the default shell is still set to /etc/cli.sh (clish). The command module requires the default shell to be bash.
- To be able to run the first time wizard, we can utilize the raw module which run the command directly on the configured shell on the remote system, in this case clish.
- We also need to change the ansible configurations file to ignore the host key checking using the value host_key_checking = False.
- Notice that we are using stdout_lines instead of stdout.

9. Edit the ansible configurations file on the orchestrator and add the line host_key_checking = False under the [default] section.



```
# config file for ansible -- https://ansible.com/
# ===============================================

# nearly all parameters can be overridden in ansible-playbook
# or with command line flags. ansible will read ANSIBLE_CONFIG,
# ansible.cfg in the current working directory, .ansible.cfg in
# the home directory or /etc/ansible/ansible.cfg, whichever it
# finds first

[defaults]
host_key_checking = False
# some basic default values...
```

10. Run the playbook again and make sure the hostname was returned successfully.

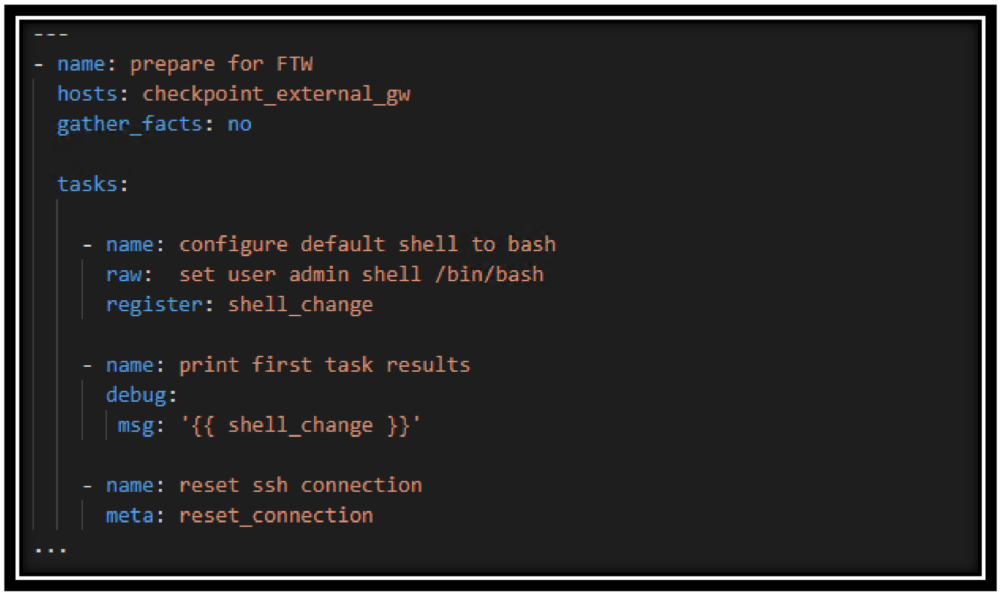


```
administrator@orchestrator:~/playbooks$ ansible-playbook -i hosts gaia_config.yml

PLAY [Get Device hostname] ****************************************************************************************************

TASK [task to get hostname] ***************************************************************************************************
changed: [203.0.113.40]

TASK [display host name] ******************************************************************************************************
ok: [203.0.113.40] => {
    "msg": [
        "In order to configure your system, please access the Web UI and finish the First Time Wizard.",
        "gw-784962"
    ]
}

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

Exercise 4: Running First Time Wizard with Ansible

Now that we are familiar with using the raw module, we can use it to configure our shell and set the expert password. Once that step is completed, we can run the first time wizard using the config_system Gaia command using the built-in command module.

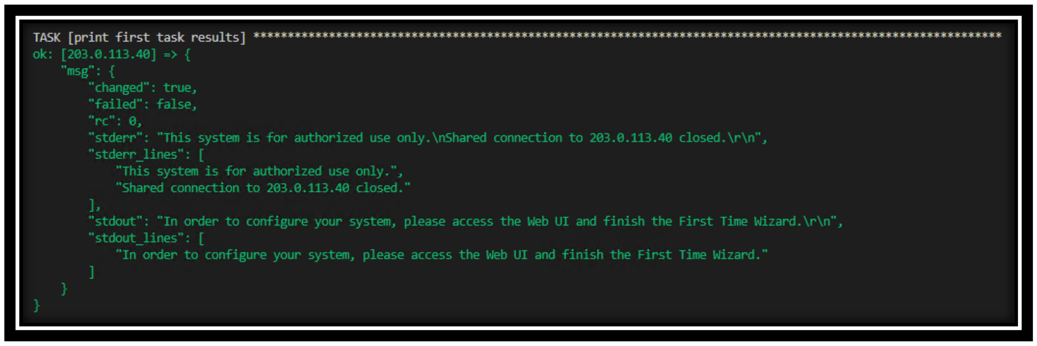
1. Change the playbook to create an expert account using a password and then set the default shell to be /bin/bash.



```
---
- name: prepare for FTW
  hosts: checkpoint_external_gw
  gather_facts: no

  tasks:

    - name: configure default shell to bash
      raw:  set user admin shell /bin/bash
      register: shell_change

    - name: print first task results
      debug:
        msg: '{{ shell_change }}'

    - name: reset ssh connection
      meta: reset_connection
...
```

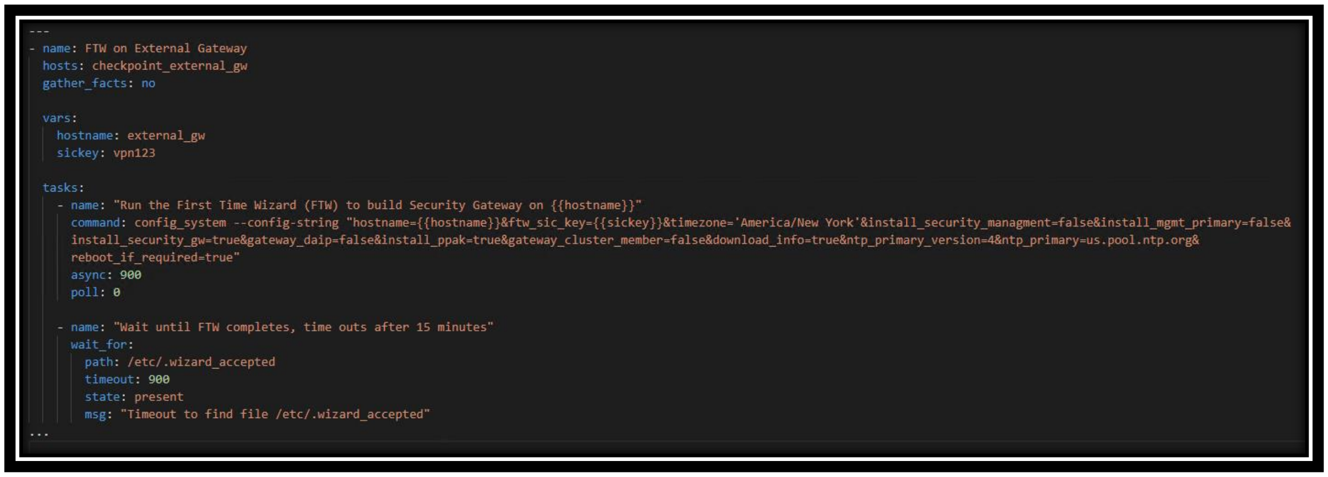
- Note that Ansible will try to reuse the connection for the next playbook if we run within the timeout period. This means that the command will use clish instead of bash. Hence, we are resetting the connection before moving to the next task.

2. Run the playbook and verify the command was executed successfully.



```
TASK [print first task results] ****************************************************************************************************
ok: [203.0.113.40] => {
    "msg": {
        "changed": true,
        "failed": false,
        "rc": 0,
        "stderr": "This system is for authorized use only.\nShared connection to 203.0.113.40 closed.\r\n",
        "stderr_lines": [
            "This system is for authorized use only.",
            "Shared connection to 203.0.113.40 closed."
        ],
        "stdout": "In order to configure your system, please access the Web UI and finish the First Time Wizard.\r\n",
        "stdout_lines": [
            "In order to configure your system, please access the Web UI and finish the First Time Wizard."
        ]
    }
}
```

3. Create a new playbook called ftw.yml. In this playbook, we will use the command module to run the config_system script on Gaia to run the first time wizard. Refer to sk69701 for more details.

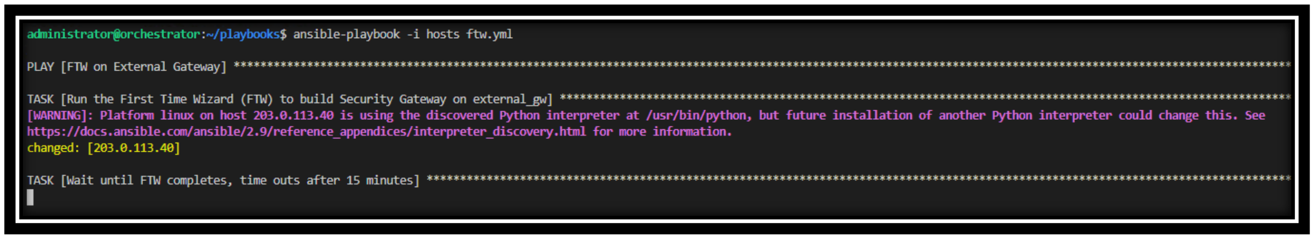


```
---
- name: FTW on External Gateway
  hosts: checkpoint_external_gw
  gather_facts: no

  vars:
    hostname: external_gw
    sickey: vpn123

  tasks:
    - name: "Run the First Time Wizard (FTW) to build Security Gateway on {{hostname}}"
      command: config_system --config-string "hostname={{hostname}}&ftw_sic_key={{sickey}}&timezone='America/New York'&install_security_managment=false&install_mgmt_primary=false&
      install_security_gw=true&gateway_daip=false&install_ppak=true&gateway_cluster_member=false&download_info=true&ntp_primary_version=4&ntp_primary=us.pool.ntp.org&
      reboot_if_required=true"
      async: 900
      poll: 0

    - name: "Wait until FTW completes, time outs after 15 minutes"
      wait_for:
        path: /etc/.wizard_accepted
        timeout: 900
        state: present
        msg: "Timeout to find file /etc/.wizard_accepted"
...
```

- The first time wizard might take few minutes to complete, we are using async asynchronous mode in playbooks to avoid connection timeouts or to avoid blocking subsequent tasks. We are setting the timeout to 900 seconds. However, we want Ansible to move on to do the next task and not wait for the FTW to complete. Hence, we are using poll set to 0.
- The second task is to check if the file /etc/.wizard_accepted. This file is created after the first time wizard is completed otherwise it does not exist.
- For more details refer to the documentations on the ansible portal:

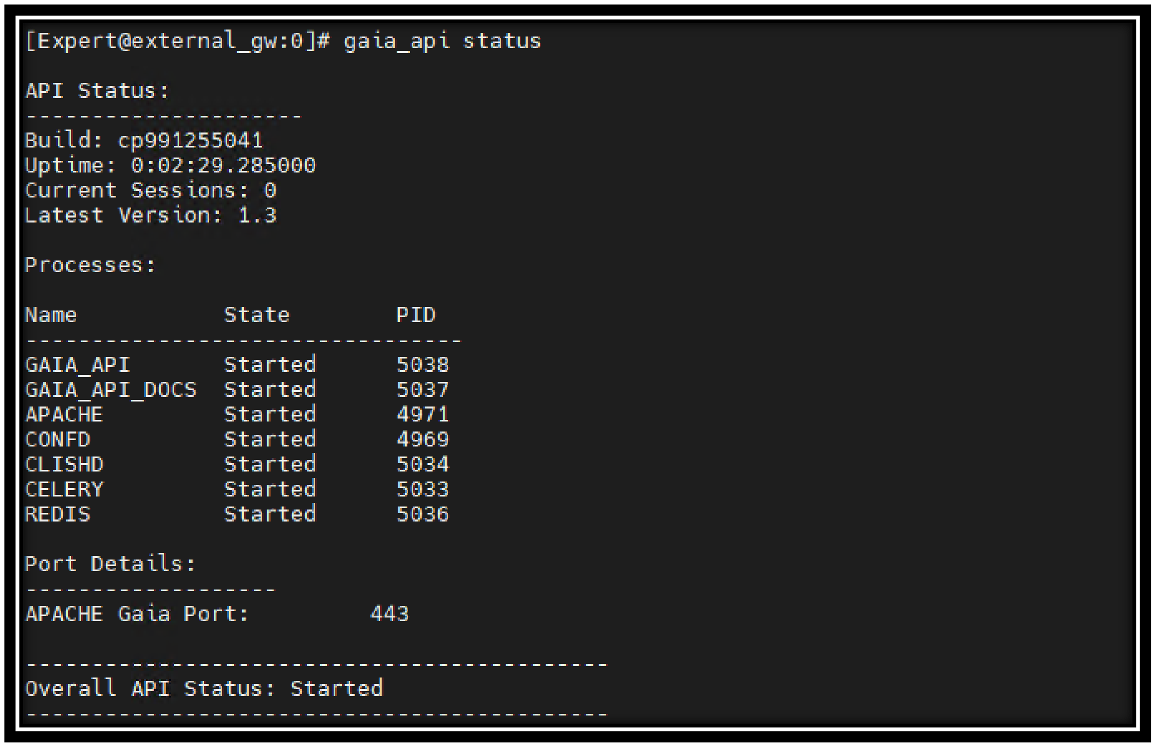
  https://docs.ansible.com/ansible/latest/user_guide/playbooks_async.html

4. Run the playbook for the first time wizard and notice that the second task is waiting for the FTW to complete,



```
administrator@orchestrator:~/playbooks$ ansible-playbook -i hosts ftw.yml

PLAY [FTW on External Gateway] ***************************************************************************************************************

TASK [Run the First Time Wizard (FTW) to build Security Gateway on external_gw] **************************************************************
[WARNING]: Platform linux on host 203.0.113.40 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.
changed: [203.0.113.40]

TASK [Wait until FTW completes, time outs after 15 minutes] **********************************************************************************
```

5. The gateway will reboot after completing the FTW, wait until the gateway is up and connect to the external gateway using MobaXterm. Verify that it was configured correctly.



```
[Expert@external_gw:0]# gaia_api status

API Status:
---------------------
Build: cp991255041
Uptime: 0:02:29.285000
Current Sessions: 0
Latest Version: 1.3

Processes:

Name           State        PID
---------------------------------
GAIA_API       Started      5038
GAIA_API_DOCS  Started      5037
APACHE         Started      4971
CONFD          Started      4969
CLISHD         Started      5034
CELERY         Started      5033
REDIS          Started      5036

Port Details:
-------------------
APACHE Gaia Port:        443

--------------------------------------------
Overall API Status: Started
--------------------------------------------
```

End of Lab 3