Check Point and Ansible Setup

Version 1.0
Sep 23, 2020

Setup the Ubuntu Machine to use Ansible and the Check Point R80 Management Module

Objective: Setup Ansible on a newly installed Ubuntu 18.04LTS Server. We will install the latest version of the Ansible module from galaxy.ansible.com along with Ansible 2.9.
Detailed instructions can be found on the Ansible Installation site for the Ubuntu install - https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#installing-ansible-on-ubuntu

1. Connect to your newly installed Ubuntu machine via SSH

2. Update the system by running the following commands
   a. `sudo apt-get update`
   b. `sudo apt-get upgrade`

3. Install Ansible by running the following commands
   a. `sudo apt install software-properties-common`
   b. `sudo apt-add-repository --yes --update ppa:ansible/ansible`
   c. `sudo apt install ansible`

4. Validate the version of Ansible that is installed. The Check Point module requires Ansible V2.9 or later
   a. Run – `ansible --version`

```
ubuntu@ubuntu:~$ ansible --version
ansible 2.9.13
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/home/ubuntu/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.17 (default, Jul 20 2020, 15:37:01) [GCC 7.5.0]
ubuntu@ubuntu:~$
```

5. Now that Ansible 2.9 is installed we can install the Check Point Ansible module for R80. Detailed instructions - https://galaxy.ansible.com/check_point/mgmt

   Run the following command –
   `ansible-galaxy collection install check_point.mgmt`

```
ubuntu@ubuntu:~$ ansible-galaxy collection install check_point.mgmt
Process install dependency map
Starting collection install process
Installing 'check_point.mgmt:2.0.0' to '/home/ubuntu/.ansible/collections/ansible_collections/check_point/mgmt'
ubuntu@ubuntu:~$
```

©2020 Check Point Software Technologies Ltd. All rights reserved

Setup Ansible to use the Check Point R80 Module

Objective: Setup Ansible to use the newly install Check Point Module. In this example the Check Point Smart Center will have the ip address of 10.5.0.82

1. Make sure that you have access to the R80 Manager and the API server is open for communication with the Ansible Server. Open SmartConsole and check “Manage & Settings > Blades > Management API > Advanced settings”.

2. Via SSH run “api status” on the R80 Management server and verify that it is open and the API server is running and ready.

```
[Expert@R81EAMgmt:0]# api status

API Settings:
---------------------
Accessibility:                  Require all granted
Automatic Start:                Enabled

Processes:

Name       State      PID       More Information
-----------------------------------------------
API        Started    8816
CPM        Started    8816      Check Point Security Management Server is running and ready
FWM        Started    8493
APACHE     Started    7980
```

3. On the Ansible host create a hosts file that will contain your initial configuration. Here is a simple example.

```
[check_point]
10.5.0.82
[check_point:vars]
ansible_httpapi_use_ssl=True
ansible_httpapi_validate_certs=False
ansible_user=api_user
ansible_password=vpn123
ansible_network_os=check_point.mgmt.checkpoint
```

The entries below [check_point] are the ip address listing of the management server(s).
Variables used with the Check Point R80 Ansible module are listed under the [check_point:vars] section.
The ansible_user/ansible_password is the R80 user that is configured on the management server.
The ansible_network_os property informs Ansible what module we would like to use. As shown above this will use the latest module that is install from galaxy for the Check Point module. If you would like to use the original module as delivered from Ansible 2.9 replace this setting with `ansible_network_os=checkpoint`

©2020 Check Point Software Technologies Ltd. All rights reserved

4. Next create a simple Ansible playbook to create a simple network.

```yaml
---
- name: playbook name
  hosts: check_point
  connection: httpapi
  gather_facts: false
  tasks:
    - name: Task to manage a network
      check_point.mgmt.cp_mgmt_network:
        name: "My Ansible Network 1"
        subnet: "172.31.254.0"
        mask_length: 24
        auto_publish_session: true
        state: present
```

5. Now run the ansible-playbook command and reference the hosts file for the playbook. Use the following command
   `ansible-playbook -i hosts ansible_network.yml`

```
ubuntu@ubuntu:~$ ansible-playbook -i hosts ansible_network.yml

PLAY [playbook name] ***********************************************************************************************

TASK [Task to manage a network] ************************************************************************************
changed: [10.5.0.82]

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

ubuntu@ubuntu:~$
```

6. In the example playbook, change the state to absent to remove the network.

```yaml
---
- name: playbook name
  hosts: check_point
  connection: httpapi
  gather_facts: false
  tasks:
    - name: Task to manage a network
      check_point.mgmt.cp_mgmt_network:
        name: "My Ansible Network 1"
        subnet: "172.31.254.0"
        mask_length: 24
        auto_publish_session: true
        state: absent
```

7. Now run the following command to remove our example network.
   `ansible-playbook -i hosts ansible_network.yml`

©2020 Check Point Software Technologies Ltd. All rights reserved

```
ubuntu@ubuntu:~$ ansible-playbook -i hosts ansible_network.yml

PLAY [playbook name] ***********************************************************************************************

TASK [Task to manage a network] ************************************************************************************
changed: [10.5.0.82]

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

ubuntu@ubuntu:~$
```

Setup Ansible to use the Check Point Gaia API Module

Objective: Setup Ansible to use Check Point Gaia API Module. In this example we will use a new gateway running at 10.5.0.83 using the Gaia API V1.4 that is already installed.

1. First, we need to install the Ansible Gaia API module from galaxy.ansible.com. Run the following command to install the module
   `ansible-galaxy collection install check_point.gaia`

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

2. Next verify that the user you would like to use in the Gaia is enabled. By default only the admin user is enabled. In our example I will use gaia_user as my user on the Gaia API. The command that was used in Gaia to enable this user is the following.

   ```
   gaia_api access --user gaia_user --enable true
   ```

   This process is documented as part of the Gaia API – API access here - https://sc1.checkpoint.com/documents/latest/GaiaAPIs/#api_access~v1.4%20

3. After the user is enabled, restart the Gaia API server using the following command

   ```
   gaia_api restart
   ```

4. Create a host file that will be used for the Ansible Gaia API module

©2020 Check Point Software Technologies Ltd. All rights reserved

```
[check_point_gaia]
10.5.0.83
[check_point_gaia:vars]
ansible_httpapi_use_ssl=True
ansible_httpapi_validate_certs=False
ansible_user=gaia_user
ansible_password=vpn1234
ansible_network_os=check_point.gaia.checkpoint
```

The entries below [check_point_gaia] are the ip address listing of the gaia device(s).

Variables used with the Check Point Gaia API Ansible module are listed under the [check_point_gaia:vars] section.
The ansible_user/ansible_password is the user that is configured in Gaia and enabled for use in the Gaia API.

5. Create a Gaia API playbook. For the first example we will change the hostname of the gateway.

```
---
- name: Gaia API Examples
  hosts: check_point_gaia
  connection: httpapi
  gather_facts: no
  tasks:
    - name: Manage Hostname on Gaia
      check_point.gaia.cp_gaia_hostname:
        name: "r80dot40_ansible_gw"
```

6. Next, execute the playbook and reference the hostfile we created with the gaia server.

```
ubuntu@ubuntu:~$ ansible-playbook -i gaia_api_hosts gaia_api_playbook.yml

PLAY [Gaia API Examples] ********************************************************************************************

TASK [Manage Hostname on Gaia] **************************************************************************************
changed: [10.5.0.83]

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

ubuntu@ubuntu:~$
```

7. Update the playbook to include a change to physical interface eth1

©2020 Check Point Software Technologies Ltd. All rights reserved

```yaml
---
- name: Gaia API Examples
  hosts: check_point_gaia
  connection: httpapi
  gather_facts: no
  tasks:
    - name: Manage Hostname on Gaia
      check_point.gaia.cp_gaia_hostname:
        name: "r80dot40_ansible_gw"
    - name: Manage eth1 on Gaia
      check_point.gaia.cp_gaia_physical_interface:
        name: "eth1"
        ipv4_address: "172.16.254.254"
        ipv4_mask_length: 24
        mtu: "1500"
        enabled: True
        comments: "Ansible eth1 example"
```

8. Run the play again to update the eth1 interface of the gateway. Changes should only be made to the interface, the hostname is unchanged.

```
ubuntu@ubuntu:~$ ansible-playbook -i gaia_api_hosts gaia_api_playbook.yml

PLAY [Gaia API Examples] ********************************************************************************************

TASK [Manage Hostname on Gaia] **************************************************************************************
ok: [10.5.0.83]

TASK [Manage eth1 on Gaia] ******************************************************************************************
changed: [10.5.0.83]

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

ubuntu@ubuntu:~$
```

Use the Ansible command module to make changes to Gaia directly

Objective: Use the built-in command module in Ansible to make changes to the Gaia operating system. Note that any changes made in this manner is not idempotent! Please be aware of what is being updated in your configuration. This is an un-official way to make changes to Gaia!!

1. Requirements for use
   - Gateways of version R80.10 or later. Previous versions had a whitelist that is required for Python. This restriction is not applicable to R80.10 or later.
   - Versions previous to R80.40 did not have a symbolic link to Python in a typical location that Python can be found. In these cases you will need to specify the location of the Python Interrupter. See setup instructions below for an example.

©2020 Check Point Software Technologies Ltd. All rights reserved

- **The commands that are executed are not idempotent and will be run as listed.** These commands are run just like a user is running them from the CLI every time the playbook is called.
- A check of the configuration lock should be done to verify another user is not making changes at the same time Ansible is attempting to make changes.
- An admin user on Gaia configured for /bin/bash as the shell

2. Make sure that you have an admin user configured on the Gaia machine that is setup with /bin/bash as the configured shell.

3. Add the SSH host fingerprint to the Ansible Server. This way when the connection is established it is trusted. Ansible by default will fail the connection if it has not connected to the host before. A simple way to do this is to ssh from the Ansible host to the gateway and accept the fingerprint.

4. Setup a hosts file to represent your inventory. Also note the specific Python version listed and the specific line to point to the Python Interrupter.

```
[check_point_gaia]
10.5.0.83
[check_point_gaia:vars]
ansible_httpapi_use_ssl=True
ansible_httpapi_validate_certs=False
ansible_user=gaia_user
ansible_password=vpn1234
ansible_network_os=check_point.gaia.checkpoint
ansible_python_interpreter=/opt/CPsuite-R80.40/fw1/Python/bin/python3.7

[check_point_gaia_r80_20]
10.5.0.85
[check_point_gaia_r80_20:vars]
ansible_httpapi_use_ssl=True
ansible_httpapi_validate_certs=False
ansible_user=gaia_user
ansible_password=vpn1234
ansible_network_os=check_point.gaia.checkpoint
ansible_python_interpreter=/opt/CPsuite-R80.20/fw1/Python/bin/python2.7
```

5. In our first example setup a simple playbook to pull the static-route configuration from the gateway. In this example we are checking for the database lock status and will fail the playbook if a lock exists when there is another admin in read/write to the database.

©2020 Check Point Software Technologies Ltd. All rights reserved

```yaml
---
- name: Command Module Examples
  hosts: check_point_gaia
  gather_facts: no
  tasks:
    - name: "Check for a Config Lock"
      command: "clish -c 'show config-lock'"
      register: lock_status

    - name: "Fail if Configuration is found to be locked"
      fail:
        msg: "The Configuration is locked.  Failing playbook"
      when: "'CLINFR0771' in lock_status.stdout"

    - name: "Show static route configuration"
      command: "clish -c 'show configuration static-route'"
      register: routes

    - debug:
        msg: "{{ routes.stdout_lines }}"
```

6. Now run the play and see the results. The debug statement will output the commands output.

```
ubuntu@ubuntu:~$ ansible-playbook -i gaia_api_hosts gaia_ansible_command.yml

PLAY [Command Module Examples] ***********************************************************************

TASK [Check for a Config Lock] ***********************************************************************
changed: [10.5.0.83]

TASK [Fail if Configuration is found to be locked] ***************************************************
skipping: [10.5.0.83]

TASK [Show static route configuration] ***************************************************************
changed: [10.5.0.83]

TASK [debug] *****************************************************************************************
ok: [10.5.0.83] => {
    "msg": [
        "set static-route default nexthop gateway address 10.5.0.1 on",
        "set static-route 192.168.1.1/32 nexthop gateway address 10.5.0.254 on"
    ]
}

PLAY RECAP *******************************************************************************************
10.5.0.83                  : ok=3    changed=2    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

7. Next we want to make a change to the clish configuration in Gaia. In this example we will update DNS and NTP in one playbook. We will also check again for the lock status on the database before we make a change. Note that the “-s” at the end of each clish statement. This tells Gaia/clish to save the configuration after each command is executed.

©2020 Check Point Software Technologies Ltd. All rights reserved

```yaml
---
- name: Command Module Examples
  hosts: check_point_gaia
  gather_facts: no
  tasks:
    - name: "Check for a Config Lock"
      command: "clish -c 'show config-lock'"
      register: lock_status

    - name: "Fail if Configuration is found to be locked"
      fail:
        msg: "The Configuration is locked.  Failing playbook"
      when: "'CLINFR0771' in lock_status.stdout"

    - name: "Configure DNS and NTP Settings on Gateway"
      command: "{{ item }}"
      with_items:
        - clish -c 'set dns primary 8.8.4.4' -s
        - clish -c 'set dns secondary 8.8.8.8' -s
        - clish -c 'set dns tertiary 1.1.1.1' -s
        - clish -c 'set ntp server primary 1.pool.ntp.org version 4' -s
        - clish -c 'set ntp server secondary 0.pool.ntp.org version 4' -s
        - clish -c 'set ntp active on' -s
        - clish -c 'unlock database'
```

8. Now run the play and see the update.

```
ubuntu@ubuntu:~$ ansible-playbook -i gaia_api_hosts gaia_ansible_dnsntp.yml

PLAY [Command Module Examples] ***********************************************************************

TASK [Check for a Config Lock] ***********************************************************************
changed: [10.5.0.83]

TASK [Fail if Configuration is found to be locked] ***************************************************
skipping: [10.5.0.83]

TASK [Configure DNS and NTP Settings on Gateway] *****************************************************
changed: [10.5.0.83] => (item=clish -c 'set dns primary 8.8.4.4' -s)
changed: [10.5.0.83] => (item=clish -c 'set dns secondary 8.8.8.8' -s)
changed: [10.5.0.83] => (item=clish -c 'set dns tertiary 1.1.1.1' -s)
changed: [10.5.0.83] => (item=clish -c 'set ntp server primary 1.pool.ntp.org version 4' -s)
changed: [10.5.0.83] => (item=clish -c 'set ntp server secondary 0.pool.ntp.org version 4' -s)
changed: [10.5.0.83] => (item=clish -c 'set ntp active on' -s)
changed: [10.5.0.83] => (item=clish -c 'unlock database')

PLAY RECAP *******************************************************************************************
10.5.0.83                  : ok=2    changed=2    unreachable=0    failed=0    skipped=1    rescued=0    ignore

ubuntu@ubuntu:~$
```

©2020 Check Point Software Technologies Ltd. All rights reserved