Configure FMC with Ansible to Update FTD Interface IP

Contents

Introduction

Prerequisites

- Requirements
- Components Used

Background Information

Configure

- Network Diagram
- Configurations

Verify

Troubleshoot

Related Information

Introduction

This document describes the steps to automate Firepower Management Center(FMC) to configure Firepower Threat Defense(FTD) interface IP with Ansible.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Ansible
- Ubuntu Server
- Cisco Firepower Management Center (FMC) Virtual
- Cisco Firepower Threat Defense (FTD) Virtual

In the context of this laboratory situation, Ansible is deployed on Ubuntu.

It is essential to ensure that Ansible is successfully installed on any platform supported by Ansible for running the Ansible commands referenced in this article.

Components Used

The information in this document is based on these software and hardware versions:

- Ubuntu Server 22.04
- Ansible 2.10.8
- Python 3.10
- Cisco Firepower Threat Defense Virtual 7.4.1
- Cisco Firepower Management Center Virtual 7.4.1

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

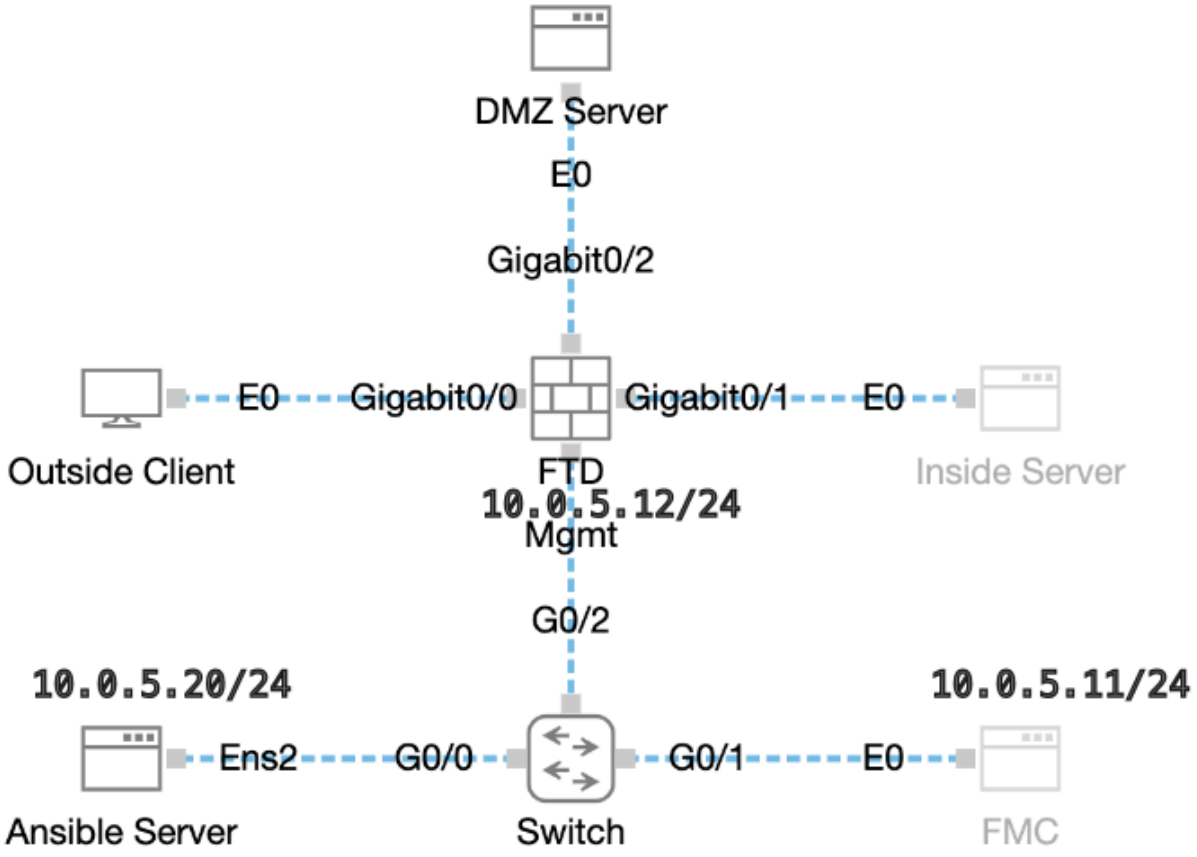
Background Information

Ansible is a highly versatile tool, demonstrating significant efficacy in managing network devices. Numerous methodologies can be employed to run automated tasks with Ansible. The method employed in this article serves as a reference for test purposes.

In this example, the interface ip address, mask and interface name are updated to FTD after running the playbook example successfully.

Configure

Network Diagram



Topology

Configurations

Because Cisco does not support example scripts or customer-written scripts, we have some examples you can test depending on your needs.

It is essential to ensure that preliminary verificationhas been duly completed.

- Ansible server possesses internet connectivity.

- Ansible server is capable of successfully communicating with the FMC GUI Port (the default port for FMC GUI is 443).
- The FTD is registered successfully to FMC.

Step 1. Connect to the CLI of the Ansible server via SSH or console.

Step 2. Run command ansible-galaxy collection install cisco.fmcansible in order to install Ansible collection of FMC on your Ansible server.

<#root>

cisco@inserthostname-here:~$

ansible-galaxy collection install cisco.fmcansible

Step 3. Run command mkdir /home/cisco/fmc_ansible in order to create a new folder to store the related files. In this example, the home directory is /home/cisco/, the new folder name is fmc_ansible.

<#root>

cisco@inserthostname-here:~$

mkdir /home/cisco/fmc_ansible

Step 4. Navigate to the folder **/home/cisco/fmc_ansible**, create inventory file. In this example, the inventory file name is inventory.ini.

<#root>

cisco@inserthostname-here:~$

cd /home/cisco/fmc_ansible/

ccisco@inserthostname-here:~/fmc_ansible$

ls

inventory.ini

You can duplicate this content and paste it for utilization, altering the **highlighted** sections with the accurate parameters.

<#root>

[fmc]

10.0.5.11

```
[fmc:vars]
ansible_user=
```

cisco

```
ansible_password=
```

cisco

```
ansible_httpapi_port=443
ansible_httpapi_use_ssl=True
ansible_httpapi_validate_certs=False
network_type=HOST
ansible_network_os=cisco.fmcansible.fmc
```

Step 5. Navigate to the folder **/home/cisco/fmc_ansible**, create variable file. In this example, the variable file name is fmc-configure-interface-vars.yml.

<#root>

```
cisco@inserthostname-here:~$
```

cd /home/cisco/fmc_ansible/

```
ccisco@inserthostname-here:~/fmc_ansible$
```

ls

fmc-configure-interface-vars.yml

```
inventory.ini
```

You can duplicate this content and paste it for utilization, altering the **highlighted** sections with the accurate parameters.

<#root>

```
    user:
      domain: 'Global'
    onboard:
      acp_name: 'TEMPACP'
    device_name:
      ftd1: 'FTDA'
    ftd_data:
      outside_name: '
```

Outside

```
'
  inside_name: '
```

Inside

```
'
```

```
  dmz_name: '
DMZ
'
  outside_ip: '
10.1.1.1
'
  inside_ip: '
10.1.2.1
'
  dmz_ip: '
10.1.3.1
'
  mask24: '
255.255.255.0
'
```

Step 6.Navigate to the folder **/home/cisco/fmc_ansible**, create playbook file. In this example, the playbook file name is fmc-configure-interface-playbook.yaml.

<#root>

```
cisco@inserthostname-here:~$
 cd /home/cisco/fmc_ansible/

ccisco@inserthostname-here:~/fmc_ansible$
ls

fmc-configure-interface-playbook.yaml
fmc-configure-interface-vars.yml inventory.ini
```

You can duplicate this content and paste it for utilization, altering the **highlighted** sections with the accurate parameters.

<#root>

```
---

- name: Update FTD Interface IP Address
  hosts: fmc
  connection: httpapi

  tasks:
```

```
        - name: Task01 - Get User Domain
          cisco.fmcansible.fmc_configuration:
            operation: getAllDomain
            filters:
              name: "{{
```

user.domain

```
 }}"
        register_as: domain

    - name: Task02 - Get Devices
      cisco.fmcansible.fmc_configuration:
        operation: getAllDevice
        path_params:
          domainUUID: '{{ domain[0].uuid }}'
        query_params:
          expanded: true
        filters:
          name: "{{
```

device_name.ftd1

```
 }}"
        register_as: device_list

    - name: Task03 - Get Physical Interfaces
      cisco.fmcansible.fmc_configuration:
        operation: getAllFTDPhysicalInterface
        path_params:
          containerUUID: '{{ device_list[0].id }}'
          domainUUID: '{{ domain[0].uuid }}'
        register_as: physical_interfaces

    - name: Task04 - Setup Outside Interface with static IP
      cisco.fmcansible.fmc_configuration:
        operation: updateFTDPhysicalInterface
        data:
          ifname: "{{
```

ftd_data.outside_name

```
 }}"
          ipv4:
            static:
              address: "{{ Outside_ip | default(
```

ftd_data.outside_ip

```
) }}"
              netmask: "{{ Outside_netmask | default(
```

ftd_data.mask24

```
) }}"
          MTU: 1500
          enabled: True
          mode: NONE
          type: physicalinterface
          name:
```

GigabitEthernet0/0

```
        path_params:
          domainUUID: '{{ domain[0].uuid }}'
```

```
          containerUUID: '{{ device_list[0].id }}'
          objectId: '{{ physical_interfaces[0].id }}'

    - name: Task05 - Setup Inside Interface with static IP
      cisco.fmcansible.fmc_configuration:
        operation: updateFTDPhysicalInterface
        data:
          ifname: "{{
```

ftd_data.inside_name

```
 }}"
          ipv4:
            static:
              address: "{{ Inside_ip | default(
```

ftd_data.inside_ip)

```
 }}"
              netmask: "{{ Inside_netmask | default(
```

ftd_data.mask24

```
) }}"
          MTU: 1500
          enabled: True
          mode: NONE
          type: physicalinterface
          name:
```

GigabitEthernet0/1

```
        path_params:
          domainUUID: '{{ domain[0].uuid }}'
          containerUUID: '{{ device_list[0].id }}'
          objectId: '{{ physical_interfaces[1].id }}'

    - name: Task06 - Setup DMZ Interface with static
      cisco.fmcansible.fmc_configuration:
        operation: updateFTDPhysicalInterface
        data:
          ifname: "{{
```

ftd_data.dmz_name

```
 }}"
          ipv4:
            static:
              address: "{{ DMZ_ip | default(
```

ftd_data.dmz_ip

```
) }}"
              netmask: "{{ DMZ_netmask | default(
```

ftd_data.mask24

```
) }}"
          MTU: 1500
          enabled: True
          mode: NONE
          type: physicalinterface
          name:
```

GigabitEthernet0/2

```
    path_params:
      domainUUID: '{{ domain[0].uuid }}'
      containerUUID: '{{ device_list[0].id }}'
      objectId: '{{ physical_interfaces[2].id }}'

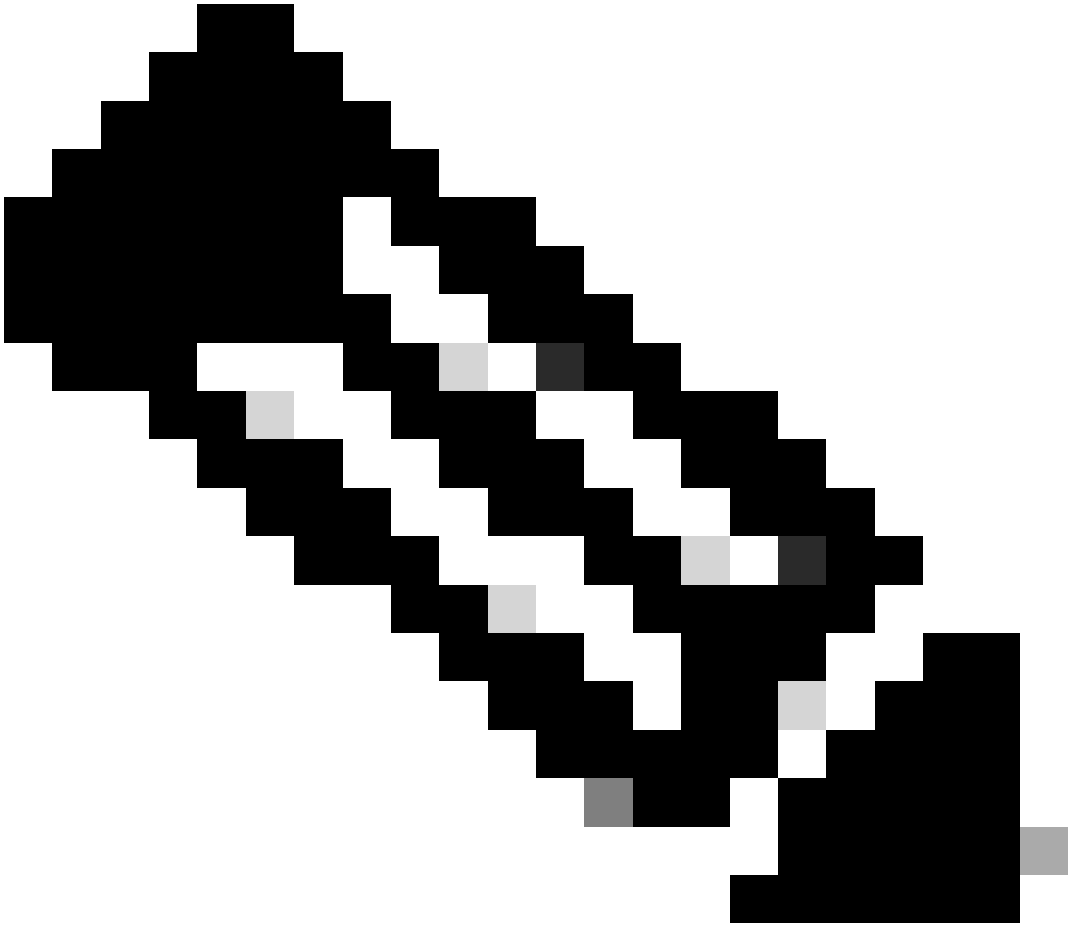
- name: Task07 - Get Deployable Devices
  cisco.fmcansible.fmc_configuration:
    operation: getDeployableDevice
    path_params:
      domainUUID: '{{ domain[0].uuid }}'
    query_params:
      expanded: true
    register_as: deploy_devices

- name: Task08 - Start Deployment
  cisco.fmcansible.fmc_configuration:
    operation: createDeploymentRequest
    data:
      version: '{{ deploy_devices[0].version }}'
      deviceList:
        - '{{ deploy_devices[0].device.id }}'
      forceDeploy: True
    path_params:
      domainUUID: '{{ domain[0].uuid }}'
    register_as: deployment_job

- name: Wait for Deployment Complete
  ansible.builtin.wait_for:
    timeout: 120
  delegate_to: localhost

- name: Task09 - Poll Deployment Status Until Deployment Successful
  cisco.fmcansible.fmc_configuration:
    operation: getDeploymentDetail
    path_params:
      containerUUID: '{{ deploy_devices[0].device.id }}'
      domainUUID: '{{ domain[0].uuid }}'
    register_as: deployment_status
  until: deployment_status[0].status is match("SUCCEEDED")
  retries: 200
  delay: 3

- name: Task10 - Stop The Playbook If The Deployment Failed
  fail:
    msg: 'Deployment failed. Status: {{ deployment_status[0].status }}'
  when: deployment_status[0].status is not match("SUCCEEDED")
```



Note: The names highlighted in this example playbook serve as variables. The corresponding values for these variables are preserved within the variable file.

Step 7. Navigate to the folder **/home/cisco/fmc_ansible**, run command ansible-playbook -i <inventory_name>.ini <playbook_name>.yaml -e@"<playbook_vars>.yml"in order to play the ansible task.

In this example, the command is ansible-playbook -i inventory.ini fmc-configure-interface-playbook.yaml -e@"fmc-configure-interface-vars.yml" .

<#root>

```
cisco@inserthostname-here:~$
```

cd /home/cisco/fmc_ansible/

```
cisco@inserthostname-here:~/fmc_ansible$
```

ls

```
fmc-configure-interface-playbook.yaml fmc-configure-interface-vars.yml  inventory.ini

cisco@inserthostname-here:~/fmc_ansible$
```

ansible-playbook -i inventory.ini fmc-configure-interface-playbook.yaml -e@"fmc-configure-interface-vars

```
PLAY [Update FTD Interface IP Address] ***************************************************************

TASK [Gathering Facts] *******************************************************************************
ok: [10.0.5.11]

TASK [Task01 - Get User Domain] **********************************************************************
ok: [10.0.5.11]

TASK [Task02 - Get Devices] **************************************************************************
ok: [10.0.5.11]

TASK [Task03 - Get Physical Interfaces] **************************************************************
ok: [10.0.5.11]

TASK [Task04 - Setup Outside Interface with static IP] ***********************************************
changed: [10.0.5.11]

TASK [Task05 - Setup Inside Interface with static IP] ************************************************
changed: [10.0.5.11]

TASK [Task06 - Setup DMZ Interface with static] ******************************************************
changed: [10.0.5.11]

TASK [Task07 - Get Deployable Devices] ***************************************************************
ok: [10.0.5.11]

TASK [Task08 - Start Deployment] *********************************************************************
changed: [10.0.5.11]

TASK [Wait for Deployment Complete] ******************************************************************
ok: [10.0.5.11]

TASK [Task09 - Poll Deployment Status Until Deployment Successful] ***********************************
ok: [10.0.5.11]

TASK [Task10 - Stop The Playbook If The Deployment Failed] *******************************************
skipping: [10.0.5.11]

PLAY RECAP *******************************************************************************************
10.0.5.11 : ok=11 changed=4 unreachable=0 failed=0 skipped=1 rescued=0 ignored=0
```

Verify

Use this section to confirm that your configuration works properly.

Connect to the CLI of the FTD via SSH or console and run the commands `show interface ip brief` and `show running-config interface GigabitEthernet 0/X` .

The interface name, ip address and mask are configured successfully.

<#root>

```
> show interface ip brief
Interface            IP-Address   OK? Method Status   Protocol
GigabitEthernet0/0   10.1.1.1
     YES manual
up       up


GigabitEthernet0/1   10.1.2.1
     YES manual
up       up


GigabitEthernet0/2   10.1.3.1
     YES manual
up       up



>
show running-config interface GigabitEthernet 0/0


!
interface GigabitEthernet0/0
nameif
Outside


cts manual
propagate sgt preserve-untag
policy static sgt disabled trusted
security-level 0
ip address 10.1.1.1 255.255.255.0




>
show running-config interface GigabitEthernet 0/1


!
interface GigabitEthernet0/1
nameif
Inside


cts manual
propagate sgt preserve-untag
policy static sgt disabled trusted
security-level 0
ip address 10.1.2.1 255.255.255.0
```

```
>
show running-config interface GigabitEthernet 0/2

!
interface GigabitEthernet0/2
nameif
DMZ

cts manual
propagate sgt preserve-untag
policy static sgt disabled trusted
security-level 0
ip address 10.1.3.1 255.255.255.0
```

Troubleshoot

This section provides information you can use to troubleshoot your configuration.

In order to see more logs of ansible playbook, you can run ansible playbook with –vvv

```
cisco@inserthostname-here:~/fmc_ansible$ ansible-playbook -i inventory.ini fmc-configure-interface-play
```

Related Information

[Cisco Devnet FMC Ansible]