

# **Cloud Automation Community Framework (CACF)**

## **Onboarding Network Endpoints**

**Author:** Dhanabharathi Danassegarane

Satyajit Roy

**Reviewer:** Hernan Pablo Miguel

Aitor Goirigolzarri Ormaza

**Owners:** Network and Edge Automation Team

***Document Version: 1***

***Last Revised: March 20, 2023***

---

## General overview of the steps for onboarding

1. Setup the min requirements and connectivity to the network endpoints from the last jump host and through the CACF tower
2. Re-use the same Jump host Reconfigure jump host if required
3. Prepare the appropriate endpoint credentials to manage the network device and store them in the Tower credentials
4. Add the network required endpoints to the Tower Inventory
5. Set up the CACF automation tools in the tower to prepare the basic Jump host validation playbooks
6. Run the Jump host connectivity “check\_conn” to ensure the step 1 & step 2 are setup good
7. Network devices needs to setup ‘ssh tunnelling’ to execute network module on network devices via Jump hosts
8. Then run the basic network playbook like single command execution to test if the above are setup are fulfilled correctly for network automation.

---

# 1. Minimum Requirements:

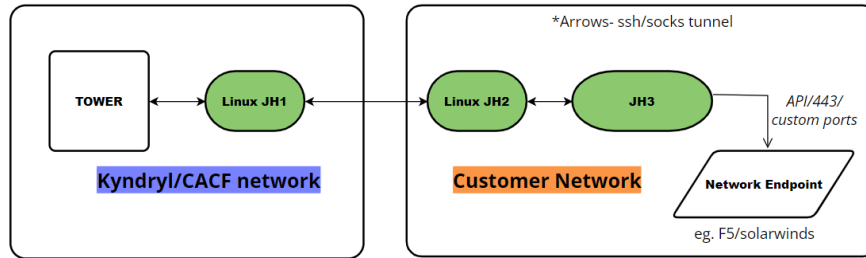
- a. There can either be a common ID for all the network endpoints or individual IDs defined for every type such as Cisco/Checkpoint, which will in turn be defined in the tower credentials to authenticate the endpoints during playbook execution. Furthermore, this ID can either be the locally defined ones or in present in the remote authentication server like CyberArk/TACACS/RADIUS, but this ID should be ensured to be non-interactive i.e., without any kind of Two-Factor Authentication. These ID passwords can be set to rotate according to customer best practices.
- b. This required privileges for this ID (Read-only/Read-write/Restricted) is dependent on the playbook, for e.g., a read-only should suffice on report generation playbook whilst a read-write is necessary for a patching playbook
- c. We don't require python to be installed in the network endpoints unlike server endpoints; Likewise, SUDO privileges are also not applicable here in network endpoints
- d. Port 22/SSH must be open from the last Jump Host towards the network endpoint. We can also re-use the same Jump host setup which is being used in windows or other platform CACF automations.
- e. Although port 22/SSH will be used by most network endpoints, there are some exceptional network endpoints that requires additional ports such as 443 (refer the below section 1.1)
- f. Python interpreter 3 should be enforced for all the network playbooks because, the network playbooks are developed on top of python3 and most of the network ansible galaxy collections supports python3 and above (can be covered in Job template or inventory level)
- g. Network endpoint playbooks should be using the default CACF network venv otherwise the playbook will fail due to missing python packages (/var/lib/aws/venv/ansible\_nw/); *Please note:* The CACF release notes from the official CACF page will regularly list down all the latest python packages which are part of every virtual environment, this can be compared against the playbook's readme if there are any missing packages which needs to be installed. If additional python packages needs to be installed, then a request has to be raised to the global development team of CACF to process it further after approvals.

---

## 1.1 Exceptions:

Some network endpoints such as F5, Palo alto, SolarWinds, will not support the native SSH (port 22) connection because, its deprecated by the respective ansible galaxy modules supporting only API/HTTPS. Below are the additional requirements for such endpoints

- a) The port 443 should be open, in addition to the port 22 from the last jump host (else)
- b) Using "delegate to", within the playbook task or portion of playbook, allows that task to run on the delegated machine and connect to the endpoint seamlessly; whereas this delegated machine is a common jump server in customer's premise which has port 443 open towards the endpoint. For instance, in the below diagram, the JH3 is the delegated common server which has the port 443 open to the network endpoint. These requirements will be clearly documented if required in the playbook's readme; requirements section, for more details please reach out to the playbook owner.



---

## 1.2 Supported Devices and versions

Please refer to the below ansible article for the list of support network endpoints

[https://github.com/ansible/ansible/blob/devel/docs/docsite/rst/network/user\\_guide/platform\\_index.rst](https://github.com/ansible/ansible/blob/devel/docs/docsite/rst/network/user_guide/platform_index.rst)

Furthermore, the supported versions of each category vendor devices are dependent to the version of ansible galaxy module used and its pre-requisites; unless there isn't any version exception specified on the playbook being used as part of CACF.

---

## 2. Setup inventory:

---


### 2.1 Working with inventory

Inventory is the collection of endpoints nodes/hosts, in most cases a set of hosts of same kind can be grouped for common variable assignment to execute the playbooks with ease. Assigning groups also allows for node selection in the Play and bulk variable assignment.

For additional info about inventory management, please refer Tower Inventory Management at: [TWPs/CACF Ansible Tower Inventory Management at master · Continuous-Engineering/TWPs \(kyndryl.net\)](https://kyndryl.net/TWPs/CACF-Ansible-Tower-Inventory-Management-at-master-Continuous-Engineering/TWPs)

- a) A list of managed network nodes/hosts provided by one or more 'inventory sources' can be onboarded dynamically or statically. Your inventory can specify connection information specific to each node, like IP address.
- b) To start off with the static inventory import, you can manually import each device one by one using the official Ansible guide or use a custom bulk import script in cases of huge endpoint list. To add a point here, some of the CACF managed accounts are already using a readymade script prepared in python which can also be leveraged for bulk imports of endpoints and related variables
- c) Moving on with the dynamic inventory import, this is basically achieved by leveraging the source control inventory feature of ansible tower with combination of a script which can be scheduled periodically to check and load new devices from any monitoring tool like SolarWinds or CMDB database. Again, similar script is already running on some of the CACF managed accounts)

#### Samples Inventory groups

\_nw\_inventory

DETAILS

PERMISSIONS

GROUPS

HOSTS

SOURCES

COMPLETED JOBS

SEARCH

Q

KEY

GROUPS ▾


☐ nexus

☐ ios

☐ hp\_procurve

☐ f5

## Sample Host Entry


**BIGIP1\_LAB** 

DETAILS

FACTS


GROUPS

COMPLETED JOBS

\* HOST NAME 

DESCRIPTION

BIGIP1\_LAB

VARIABLES 

YAML

JSON

1

`ansible_host: 8.8.8.8`

---

## 2.2 Defining Inventory variables

The following variables are common for all platforms in the inventory, though they can be overwritten for a particular inventory group or host. Please use this link to define the ansible OS and network connection method according to the network device vendor and type [https://github.com/ansible/ansible/blob/devel/docs/docsite/rst/network/user\\_guide/platform\\_index.rst](https://github.com/ansible/ansible/blob/devel/docs/docsite/rst/network/user_guide/platform_index.rst)

### a. ansible\_connection

Ansible uses the ansible-connection setting to determine how to connect to a remote device. When working with Ansible Networking, set this to an appropriate network connection option, such as `ansible.netcommon.network_cli`, so Ansible treats the remote node as a network device. Without this setting, Ansible would attempt to use ssh to connect to the remote and execute the Python script on the network device, which would fail because Python generally isn't available on network devices. **Please note: This is a mandatory variable to be defined for all the network endpoints in the inventory or host or group level to avoid failure.**

### b. ansible\_network\_os

Informs Ansible which Network platform this hosts corresponds to. This is required when using the `ansible.netcommon.*` connection options.

**Please note: This is a mandatory variable to be defined for all the network endpoints in the inventory or host or group level to avoid failure.**

**Note:** Both “ansible\_user” and “ansible\_password” are usually handed by the credential mapped to the network endpoints variables in the tower, thereby saving time in defining them on individually on every endpoint.

---

## 2.3 Standard set of variables for network endpoint inventory

Below are the commonly used endpoints and their respective standard set of variables, please refer to the respective ansible galaxy's readme for the missing network devices in the table

|                   |                             | ansible_connection: settings available |         |         |       |
|-------------------|-----------------------------|--|---------|---------|-------|
| Network OS        | ansible_network_os:*        | network_cli                            | netconf | httpapi | local |
| Arista EOS        | arista.eos.eos              | ✓                                      |         | ✓       | ✓     |
| Ciena SAOS6       | ciena.saos6.saos6           | ✓                                      |         |         | ✓     |
| Cisco ASA         | cisco.asa.asa               | ✓                                      |         |         | ✓     |
| Cisco IOS         | cisco.ios.ios               | ✓                                      |         |         | ✓     |
| Cisco IOS XR      | cisco.iosxr.iosxr           | ✓                                      |         |         | ✓     |
| Cisco NX-OS       | cisco.nxos.nxos             | ✓                                      |         | ✓       | ✓     |
| Extreme NOS       | community.network.nos       | ✓                                      |         |         |       |
| Extreme SLX-OS    | community.network.slxos     | ✓                                      |         |         |       |
| Extreme VOSS      | community.network.voss      | ✓                                      |         |         |       |
| F5 BIG-IP         |                             |  |         |         | ✓     |
| F5 BIG-IQ         |                             |  |         |         | ✓     |
| Junos OS          | junipernetworks.junos.junos | ✓                                      | ✓       |         | ✓     |
| Lenovo CNOS       | community.network.cnos      | ✓                                      |         |         | ✓     |
| Lenovo ENOS       | community.network.enos      | ✓                                      |         |         | ✓     |
| Meraki            |                             |  |         |         | ✓     |
| MikroTik RouterOS | community.network.routeros  | ✓                                      |         |         |       |
| Ruckus ICX        | community.network.icx       | ✓                                      |         |         |       |
| VyOS              | vyos.vyos.vyos              | ✓                                      |         |         | ✓     |

**Note:** Variable precedence in order is **Job Template (Extra vars) > Inventory > Host level**

---

## 2.4 Credentials

Below is the compiled list of the commonly used credentials in CACF and Event automation for network endpoint automation

| Credential                               | Type           | Remarks   |
|--|----------------|---|
| Jump host Proxy (Mandatory)*             | Custom         | Same JH proxy and credential can be used along platform and network automation.   |
| Network Endpoint (Mandatory) ***         | Machine        | Please refer to the min requirements for additional info  |
| Tower (Mandatory for Event automation) * | Ansible Tower  | This credential is primarily used to authenticate the API calls to and from NEXT and SFS, mainly used in event automation                           |
| NEXT (Mandatory for Event automation) *  | <TBF>          | This credential is primarily used by the global event wrapper role of event automation to send the results to NEXT, mainly used in event automation |
| Git*                                     | Source Control | This credential will be used to access the playbooks present on the github repository.  |

\*\*\* (Optional) In cases where in a single playbook handles two or more network devices which not sharing common user ID and password, another secondary machine type credential should be used for the other endpoint, because ansible restricts using same credential type more than once in same Job template. This secondary credential will be present in the tower, if not it can be requested through Continuous Improvement team.

\* Same credential can be shared across platform / Unix and network automations

Please refer the link for more details on credentials and standard CACF naming format

[Ansible Tower Credtype | cacf docs \(kyndryl.net\)](#)

[Next credentialtype | cacf docs \(kyndryl.net\)](#)

[Source Control Credtype | cacf docs \(kyndryl.net\)](#)

[Credtype remote | cacf docs \(kyndryl.net\)](#)

[Credtype jumphost xxx | cacf docs \(kyndryl.net\)](#)



## Sample Network credential

CREDENTIALS / CREATE CREDENTIAL 11

NEW CREDENTIAL 11

DETAILS

PERMISSIONS

\* NAME ⓘ

sample\_nw\_cred

DESCRIPTION ⓘ

test cred for network endpoint

ORGANIZATION ⓘ

Q

abc

That value was not found. Please enter or select a valid value.

\* CREDENTIAL TYPE ⓘ

Q

Machine

TYPE DETAILS

USERNAME

Q

cisco

PASSWORD

Q

\*\*\*\*\*

👁

☐ Prompt on launch

SSH PRIVATE KEY

HINT: Drag and drop private file on the field below.

Q

---

## 3. Final steps, troubleshooting and validating Ansible tower and jump hosts connectivity

---

### 3.1 Setting up the 'CACF toolsets' to validate jump host connectivity

You can use the connection validator tool from the common CACF automation onboarding scripts to validate if the Tower to jump hosts connectivity is good after onboarding the endpoints. Below is a sample of Project and Job template.

**I. Create Project (Screen 1).**

- Set the project name as xxx\_project\_cacftools (where xxx is the 3-char account code)
- Set the organization, SCM Credential
- Set the SCM URL to [https://github.kyndryl.net/Continuous-Engineering/ansible\\_collection\\_cacf\\_dev\\_commontools](https://github.kyndryl.net/Continuous-Engineering/ansible_collection_cacf_dev_commontools) where the playbook checkconn\_2jh.yml, checkconn\_3jh.yml are available.
- Save the Project.

**II. Create Job Template for testing connection with 2 Jump Host (Screen 2)**

- a. Set the name of the template to xxx\_jobtemplate\_chkcon\_2jh (where xxx is the 3 char account code).
- b. Set the Job Type to Run.
- c. Set the Inventory on which the Playbook needs to be run.
- d. Set the project created above (Screen 1)
- e. Select the Playbook checkconn\_2jh.yml.
- f. Add the **JH2 credentials**
- g. Save the playbook.

**Note:**

-> CACF Delivery (CI) can set up the credentials needed for JH 1 through 5 via an Onboarding Service Request: Credential Set Up - [Account OnBoarding to CACF \(sharepoint.com\)](https://sharepoint.com).

---

### 3.2 Setting up the 'ssh tunnelling' to execute network module on network devices via Jump hosts

#### **3.2.1 Step1: Using the pre-existing CACF role for playbook development by developers.**

The existing CACF role should be leveraged within the network playbook development by developers to setup ssh tunnelling automatically. This role helps in creating the ssh config file required for the connectivity. Below is the link of the repo for the mentioned role

<https://github.kyndryl.net/Continuous-Engineering/ansible-role-network-ssh-tunneling/>

### [Example of the role usage in the playbook](#)

```
---
- name: create file using the ssh tunneling role
hosts: localhost
connection: local
gather_facts: false
tasks:
- include_role:
name: ansible-role-network-ssh-tunneling

- name: ASA Playbook
hosts: all
connection: network_cli
gather_facts: false
tasks:
- name: Command
asa_command:
commands: show version
register: output
- debug:
msg: "{{ output }}"
```

### **3.2.2 Step2: configuring inventory with the required group vars**

Tower admin/account should create a separate group and add below 'ssh common args' in that group; and assign only the network devices to this group. Note: We should not reuse the existing jump host group assigned for linux and windows as they will have a different 'ssh common args'. Please refer to the below example of tunneling role in case of single Jumphost JH1 for reference.

### **Sample extra var configs covering 1 or more JH (for reference)**

#### **Case: single JumpHost(JH)**

```
ansible_ssh_common_args: >-  
  -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -  
  o  
  ProxyCommand="ssh -F {{ playbook_dir }}/ssh_config -W %h:%p  
  {{ jh1_ip }} -o  
  StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null"
```

#### **Case: two JH**

```
ansible_ssh_common_args: >-  
  -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -  
  o  
  ProxyCommand="ssh -F {{ playbook_dir }}/ssh_config -W %h:%p  
  {{ jh2_ip }} -o  
  StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null"
```

#### **Case: Three JH**

```
ansible_ssh_common_args: >-  
  -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -  
  o  
  ProxyCommand="ssh -F {{ playbook_dir }}/ssh_config -W %h:%p  
  {{ jh3_ip }} -o  
  StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null"
```

#### **Case: Four JH**

```
ansible_ssh_common_args: >-  
  -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -  
  o  
  ProxyCommand="ssh -F {{ playbook_dir }}/ssh_config -W %h:%p  
  {{ jh4_ip }} -o  
  StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null"
```