8. Ansible and Cisco IOS

8. Ansible and Cisco IOS

To give a better explanation of IOS configuration with Ansible, we will now setup a simple environment. With the following setup you will be able to retrieve data from the IOS devices and diagnose/troubleshoot your Ansible environment.

8.1 Configuration

In this section we will describe how to use ansible for the configuration of cisco device.

8.1.1 The environment

The file location of the environment is in the /etc/ansible directory. Within this directory we will add the following files: Hosts, Ansible.cfg and show.yml.

The hosts file will be the inventory file (described in chapter 2) containing the address information about the IOS devices. It will also contain the connection information and password for the IOS devices for Ansible to push/receive data from these devices.

The Ansible.cfg file is a configuration file for Ansible that is needed to disable SSH key checks when connecting to a device with SSH. Because this is a test environment and the IOS devices are not setup with SSH keys this must be disabled or Ansible will refuse to communicate with the devices.

The show.yml is the playbook that we will create to retrieve data from the IOS devices.

8.1.2 Disable SSH key checks

On order for the test environment to work we first will create the Ansible.cfg file in the directory /etc/ansible. Once you have done that you can add the following into the file:

```
[defaults]
host_key_checking = no

[paramiko_connection]
host_key_checking = no
```

8.1.3 Setting up the inventory file

In order to push configurations to the IOS devices we need to specify the addresses of the IOS devices. To do this we will create the file hosts in the directory /etc/ansible Within this file we will create a group called IOS and there specify the ip-addresses of our IOS devices. Our test environment will consist out of 2 routers. After we have specified the addresses of the IOS devices the connection method and password of these devices also must be configured for Ansible in order to use these devices.
The first part of the hosts file we use for the test environment contains the following entries:

```
[IOS]
Router1 ansible_host=10.0.1.14
Router2 ansible_host=10.0.1.15
```

The layout of the file (as mention in chapter 3) is as follows:

```
(hostname) ansible_host=(ip-address)
```

Then we will add the following entries to the hosts file in order to setup the connection type and password that Ansible will use to communicate with the IOS devices:

```
[all:vars]
ansible_become=yes
ansible_become_method=enable
ansible_network_os=ios
ansible_connection=network_cli
ansible_user=cisco
ansible_password=cisco
```

Ansible uses existing privileges to execute tasks with root privileges or with another's user permissions. Gaining these permissions can be achieved using the become statement . In our example these are the lines: *ansible_become=yes* and *ansible_become_method=enable*. The third and fourth line are to determine how to connect to a remote device and which network platform the host needs to correspond to. The last two lines are to set a user with password to remote log in.

With all these entries added the hosts file will contain the following:

```
[IOS]
Router1 ansible_host=10.0.1.14
Router2 ansible_host=10.0.1.15

[all:vars]
ansible_become=yes
ansible_become_method=enable
ansible_network_os=ios
ansible_user=cisco
ansible_password=cisco
ansible_connection=network_cli
```

8.1.4 Setting up a Playbook

In the last file that we will create for the test environment we will configure a playbook that sends a command to the IOS devices and then retrieves the information that the IOS device gives. For this we will create the file show.yml in the /etc/ansible directory with the following entries:

```
- name: Get data
  hosts: IOS
  tasks:

    - name: Gather interface data.
      ios_command:
        commands:
          - show ip int brief
      register: if_data

    - name: Interface output
      debug:
        var: if_data['stdout_lines'][0]
```

In the top of the Playbook we specify the name and what devices are used with the “hosts:” line. This is a reference to the hosts file where we specified the devices under the group [IOS].
Once we have done that, we create a task that gathers the information of the show ip interface brief command. This is done by using ios_command module, within this module we will use “commands:” to list the commands Ansible will run on the IOS devices. In the last part we take the information that the IOS devices gives us and we save it with the name if_data.

```
- name: Gather interface data.
      ios_command:
        commands:
          - show ip int brief
      register: if_data
```

> **Note**
>
> The commands will be executed from top to bottom. So, if we would add the 2 lines:
>
> - show ip int brief
> - show run
>
> The show ip int brief command will be executed first. This is important because some commands for IOS devices are used in different modes.

In order to display the data we requested from the IOS devices we must use the following lines:

```
- name: Interface output
      debug:
        var: if_data['stdout_lines'][0]
```

With this we request the data in the variable if_data.

You can now run the playbook with the following line:

```
ansible-playbook show.yml
```

The result should be as follows:

```
PLAY [Get data]
TASK [Gathering Facts] ***** ******** ****** **** ***** ***** ***** *****
ok: [Router2]
ok: [Router1]
TASK [Gather interface data.] ***** ******** ****** **** ***** ***** *****
ok: [Router1]
ok: [Router2]
TASK [Interface output) ***** ******** ****** **** ***** ***** ***** *****
ok: (Router1) -> {
    "if_data[ istdout_lines' ][0]": [
        "Interface          IP-Address      OK?         Method          Status                  Protocol",
        "GigabitEthernet1   10.0.1.14       YES         DHCP            up                      up      ",
        "GigabitEthernet2   unassigned      YES         NVRM            down                    down    ",
        "GigabitEthernet3   unassigned      YES         NVRM            administratively down   down    ",
        "GigabitEthernet4   unassigned      YES         NVRM            administratively down   down    ",
        "Loopback1          192.168.0.3     YES         NVRM            up                      up      ",
        "Loopback2          192.168.1.3     YES         NVRM            up                      up      "
    ]
}
ok: (Router2) -> {
    "if_data[ istdout_lines' ][0]": [
        "Interface          IP-Address      OK?         Method          Status                  Protocol",
        "GigabitEthernet1   10.0.1.15       YES         DHCP            up                      up      ",
        "GigabitEthernet2   unassigned      YES         NVRM            administratively down   down    ",
        "GigabitEthernet3   unassigned      YES         NVRM            administratively down   down    ",
        "GigabitEthernet4   unassigned      YES         NVRM            administratively down   down    ",
        "Loopback1          192.168.0.4     YES         NVRM            up                      up      ",
        "Loopback2          192.168.1.4     YES         NVRM            up                      up      "
    ]
}
```

8.2 Cisco IOS modules

In this section we will describe how to use ansible for the configuration of cisco device.

8.2.1 ios_logging

This module provides the logging of the Cisco IOS devices.

```
- name: configure host logging
  cisco.ios.ios_logging:
    dest: host
    name: 172.28.10.1
    state: present
```

The example above configures a host logging. The destination of the logs in set on host, the name is set with an IP-address of the destination. It is required to set the name when *dest* is set on *host.*

8.2.2 ios_facts

This module collects a base set of device facts from a device. The module will always collect a base set of facts from a device and can enable or disable the collection of additional facts.

```
- name: Gather all legacy facts
  cisco.ios.ios_facts:
    gather_subset: all

- name: Gather only the configuration and default facts
  cisco.ios.ios_facts:
    gather_subset:
    - config
```

In the first part of the example above we want to gather all facts. In the second part we gather only the configuration facts. With the line gather_subset you can gather the different facts that you want to collect.

8.2.3 ios_system

This module allows the management to configure host system parameters or remove parameters from the Cisco IOS devices.

```
- name: configure hostname and domain name
  cisco.ios.ios_system:
    hostname: Ansible_Cisco_IOS
    domain_name: test.example.com
```

In this example we configured a hostname and a domain name. The host and domain name can also be deleted with the following:

```
- name: remove complete configuration
  cisco.ios.ios_system:
    state: absent
```

8.2.4 ios_user

This module provides management of the local usernames configured on the network devices. This module also allows playbooks to manage each username individual or the total of the usernames in the current running configuration. It also supports deleting the usernames from the configuration that are not explicitly defined.

You can create a new user as follows:

```
- name: create a new user
  cisco.ios.ios_user:
    name: ansible
    nopassword: true
    sshkey: "{{ lookup('file', '~/.ssh/id_rsa.pub') }}"
    state: present
```

In this example a new user is created without a password. This allows the user to login to the system without being authenticated by a password. In this example SSH key are used. This key will be found in a file that is given as a parameter. The last line sets the state of the username, when the state is set on present, the username should be configured in the active running configuration. When the state is set to absent the username should not be in the active configuration.

8.2.5 ios_vlan

This module provides management of the VLANs on Cisco IOS devices.

```
- name: Create vlan
  cisco.ios.ios_vlan:
    vlan_id: 10
    name: testvlan
    state: present

- name: Add interfaces to VLAN
  cisco.ios.ios_vlan:
    vlan_id: 10
    interfaces:
    - GigabitEthernet0/0
    - GigabitEthernet0/1

- name: Check if interfaces is assigned to VLAN
  cisco.ios.ios_vlan:
    vlan_id: 10
    associated_interfaces:
    - GigabitEthernet0/0
    - GigabitEthernet0/1

- name: Delete vlan
  cisco.ios.ios_vlan:
    vlan_id: 10
    state: absent
```

In the example above we create a VLAN and added interface to the VLAN. After the interface are added we checked if the interfaces were added to the VLAN. In the last part of our example we deleted the VLAN. The only crucial part of adding, deleting or assigning interface to a VLAN is the vlan_id, without this your playbook won't work or the wrong VLANs are deleted.

8.2.6 ios_ping

This module tests the reachability using a ping to a remote destination. For Windows targets, you have to use the win_ping module instead.

```
- name: Ping 10.0.0.4 from the source using prod vrf and setting a count
  cisco.ios.ios_ping:
    dest: 10.0.0.4
    source: loopback1
    vrf: prod
    count: 20
```

In the example above we want to ping 10.0.0.4 from the destination loopback1. This is defined in the third and fourth line. In the last line we define the number of packets to send to the destination host.

8.2.7 ios_banner

This module configures the banner for both login and Message Of The Day (MOTD) on the remote devices that are running Cisco IOS. It also allows a playbook to add or remove banner text from a running configuration.

```
- name: configure the login banner
  cisco.ios.ios_banner:
    banner: login
    text: |
      Welcome to the Ansible guide.
      This guide will help you to
      install and understand Ansible
    state: present

- name: remove the MOTD banner
  cisco.ios.ios_banner:
    banner: MOTD
    state: absent
```

In the first part of our example we configure a login banner with the text: "Welcome to the Ansible guide. This guide will help you to install and understand Ansible". In the second part of the example we delete the MOTD from the running configuration.