# Chapter 2

**Chapter Introduction**

In this chapter, we will outline how to automate Cisco IOS-based devices using Ansible. We will explore the different modules available in ansible to automate configuration and collecting network information from Cisco IOS devices.  This chapter will be based on below sample network diagram and we will walk through how we can implement this network design using ansible.

Below is the Software releases that this chapter is based on

* Cisco IOS 15.1
* Cisco IOS-XE 16.6.1
* Ansible 2.7

The main recipes covered in this chapter is shown below

* Building Ansible Network Inventory.
* Connecting and Authentication to IOS Devices.
* Configuring System options on IOS Devices using Ansible.
* Interface configuration on IOS Devices.
* Configuring trunk and Access ports on IOS Devices.
* Configuring VLANs on IOS Devices.
* Configuring Interface IP addresses on IOS Devices.
* Configuring OSPF on IOS Devices.
* Configuring BGP on IOS Devices.
* Validating Network reachability on IOS devices.
* Retrieving IOS Device facts.
* Retreiving OPerational Data from IOS Devices

**Building Network Inventory**

In this chapter, we will outline how to build and structure the Ansible Inventory to describe the network setup outlined above.

**Getting Ready**

Ansible is installed on the Control machine

**How to do it..**

* Create a new directory with the following name ***ios\_netops***
* Inside this new folder create hosts file with the below content

$ cat hosts  
[access]  
access01 ansible\_host=172.20.1.18  
access02 ansible\_host=172.20.1.19  
  
[core]  
core01 ansible\_host=172.20.1.20  
core02 ansible\_host=172.20.1.21  
  
[wan]  
wan01 ansible\_host=172.20.1.22  
wan02 ansible\_host=172.20.1.23  
  
[lan:children]  
access  
core  
  
[network:children]  
lan  
wan

* create ansible.cfg file as shown below

$ cat ansible.cfg  
[defaults]  
inventory=./hosts  
retry\_files\_enabled=False  
gathering=explicit  
host\_key\_checking=False

**How it works..**

We built the ansible inventory using the ***hosts*** file and we defined multiple groups in order to group the different devices in our topology into these groups as follows

* We created the ***access*** group which has both access switches (access01 and access02) in our topology.
* We created the ***core*** group which group all core switches which will act as the L3 termination for all the Vlans on the access switches.
* We created the ***wan*** group which group all our Cisco IOS-XE routes which will act as our wan routers
* We created another group called lan which group both access and core groups.
* We created the ***network*** group which groups both ***lan*** and ***wan*** groups.

Finally, we create the ***ansible.cfg*** file and configure it to point to our ***hosts*** file to be used as ansible inventory file and we disable the setup module which is not needed when running ansible against network nodes.

**Connecting to Cisco IOS Devices**

In this recipe, we will outline how to connect to Cisco IOS Devices from Ansible via SSH in order to start managing the devices from Ansible.

**Getting Ready**

In order to follow along with this recipe, an ansible inventory file should be constructed as per the previous recipe, also IP reachability between the Ansible Control machine and all the devices in the network must be configured.

**How to do it..**

* Inside the directory ios\_netops create the folder ***groups\_vars*** .
* inside the group\_vars folder create the ***network.yml*** file with the below contents.

$cat network.yml  
ansible\_network\_os: ios  
ansible\_connection: network\_cli  
ansible\_user: lab  
ansible\_ssh\_pass: lab123  
ansible\_become: yes  
ansible\_become\_password: admin123  
ansible\_become\_method: enable

* On all the IOS devices ensure the following is configured to setup SSH access.

!  
hostname <device\_hostname>  
!  
ip domain name <domain\_name>  
!  
username lab secret 5 <password\_for\_lab\_user>.  
!  
enable secret 5 <enable\_password>.  
!  
line vty 0 4  
 login local  
 transport input ssh  
!

* Generate SSH keys on the Cisco IOS Devices from the config mode as shown below

(config)#crypto key generate rsa  
Choose the size of the key modulus in the range of 360 to 4096 for your  
 General Purpose Keys. Choosing a key modulus greater than 512 may take  
 a few minutes.  
  
How many bits in the modulus [512]: 2048  
% Generating 2048 bit RSA keys, keys will be non-exportable...  
[OK] (elapsed time was 0 seconds)

* on the ansible.cfg file add the following

$ cat ansible.cfg  
[defaults]  
inventory=./hosts  
retry\_files\_enabled=False  
gathering=explicit  
**host\_key\_checking=False**

**How it works..**

On the Cisco devices, we must setup SSH keys as well as have a username and password configured on the devices so as ansible can open ssh connection to the managed Cisco IOS devices. We also configured an enable password to be able to enter privilege mode and to do configuration changes. Once we apply all these configurations on the devices, we are ready to setup Ansible.

On the Ansible machine, we include all the variables required to establish the SSH sessions in the network.yml file, as per our inventory file the network group includes all the devices within our topology and thus all the attributes that we will configure in this file will apply to all the devices in our inventory. Below is a breakdown of the attributes that we included in the file:

* **ansible\_connection**, this sets how ansible connectes to the device, in this scenario we set it to ***network\_cli*** to indicate we will use ssh to connect to a network device.
* **ansible\_network\_os**, when using network\_cli as the connection plugin to connect to the network device we must indicate which network OS ansible will be connecting to so as to use the correct ssh parameters with the devices. In this scenario we will set it to ***ios*** since all the devices in our topology is IOS based devices.
* **ansible\_user**, this parameter specifies the username that ansible will use to establish the ssh session with the network device
* **ansible\_ssh\_pass**, this parameter specifies the password that ansible will use to establish the ssh session with the device.
* **ansible\_become**, this instruct ansible whether to use enable command to enter privileged mode when configuring or executing show commands on the managed device. we set this to ***yes*** in our context since we will require privilege mode to configure the devices.
* **ansible\_become\_password**, this specifies the enable password to use to enter the privileged mode on the managed IOS device.
* **ansible\_become\_method**,  this option specifies the method to use to enter the privilege mode, in our scenario this is the ***enable*** command on IOS devices.

In this receipe, i have defined the ssh password and the enable passwords as plain text just for simplicity however this is highly discouraged. We should use ***ansible-vault*** to secure the passwords and outline in the ansible-vault recipe in the previous chapter.

By default, the ssh client when connecting to any ssh device will try to verify the identity of the remote device to which it will establish an ssh session with. if the device is unknown to the ssh client it will ask  the use to verify the identify as shown below

$ ssh lab@172.20.1.23  
The authenticity of host '172.20.1.23 (172.20.1.23)' can't be established.  
RSA key fingerprint is SHA256:qcDgix+tdH+0IE9exyJ5LAxPOcJQwY5uuqZEb41H6qk.  
Are you sure you want to continue connecting (yes/no)?

When using ansible to automate the IOS devices and we use SSH to connect to the devices, one of the options overcome this host key checking that the ssh client will prefer is to disable host key checking as we did in the ***ansible.cfg*** file, this option is not recommended since it opens the door to security risks for connecting to devices which are not trusted. However, in this secnario we will continue to use, however for production environments this is not advised. The optimum scenario would be to use ssh key-based authentication

**There is More..**

In case we need to verify the identiy of the ssh hosts that we will connect to and thus enable host\_key\_checking, we can automate the addition of the ssh fingrpint of the remote ssh hosts to the ~/.ssh/known\_hosts file using ansible. We need to create a new playbook that will run on the ansible control mahcine to connect to the remote devices using ssk-keyscan command, then we collect the fingerprint of the remote machines and add them to the ~/.ssh/known\_hosts file. The playbook is shown below

---  
- name: Gather SSH keys  
 hosts: all  
 tasks:  
 - name: scan ssh keys  
 command: ssh-keyscan {{ ansible\_host }}  
 delegate\_to: localhost  
 register: ssh\_keys  
  
- name: Record Keys in in ssh known files  
 hosts: localhost  
 vars:  
 - hosts\_file: "~/.ssh/known\_hosts"  
 tasks:  
 - name: create know hosts file  
 file:  
 path: "{{ hosts\_file }}"  
 state: file  
 changed\_when: false  
  
 - name: Populate the known\_hosts file  
 blockinfile:  
 block: |  
 {% for host in groups['all'] if hostvars[host].ssh\_keys.stdout != '' %}  
 {{ hostvars[host].ssh\_keys.stdout}}  
 {% endfor %}  
 path: "{{ hosts\_file }}"  
 create: yes

We run this playbook on the ansible control machine to store the ssh keys from the remotely managed nodes before we run any of our playbooks.

**Configuring Basic System information**

In this chapter, we will outline how we can configure basic system parameters on the Cisco IOS devices like setting the hostname, DNS server and NTP servers.

Following the network setup that we outlined at the start of this chapter, we will configure the following information on all the Cisco IOS devices

* DNS Servers 8.8.8.8 and 8.8.4.4
* NTP Server 172.20.1.17

**Getting Ready**

An ansible inventory file must be present as well as the configuration for the ansible to connect to the Cisco IOS devices via SSH must be in place as outlined in the previous recipe

**How to do it..**

* Add the below information on the network.yml file under group\_vars directory

$ cat group\_vars/network.yml  
<---- Snippet ---->  
name\_servers:  
 - 8.8.8.8  
 - 8.8.4.4  
  
ntp\_server: 172.20.1.17

* Create a new playbook called ***pb\_build\_network.yml*** with the below information

$ cat pb\_build\_network.yml  
---  
- name: "PLAY 1: Configure All Lan Switches"  
 hosts: lan  
 tags: lan  
 tasks:  
 - name: "P1T1: Configure Hostname and Domain Name"  
 ios\_system:  
 hostname: "{{ inventory\_hostname }}"  
 domain\_name: "{{ domain\_name }}"  
 lookup\_enabled: no  
 name\_servers: "{{ name\_servers }}"  
  
 - name: "P1T2: Configure NTP"  
 ios\_ntp:  
 server: "{{ ntp\_server }}"  
 logging: true  
 state: present

**How it works..**

In the ***network.yml*** file we define the name\_servers variable as a list of DNS servers and we also define ntp\_servers which define the NTP servers that we want to configure on the IOS devices.

We create a playbook and the first play target all the hosts in the ***lan*** group (this includes both access and core devices) and within this play, we reference two tasks:

* **ios\_system**, which set the hostname and the DNS servers on the devices.
* **ios\_ntp**, which configured the NTP on the IOS devices and enable logging for NTP events.

Both these modules are declarative ansible modules in which we just identify the state about our infrastructure and Ansible covert this declaration into the needed IOS commands. The modules retrieve the configuration of the devices and compare its current state with our intended state (to have DNS and NTP configured on them) and then if the current state is not aligned with it it will apply the needed configuration on the devices.

When we run these tasks on all the ***lan*** devices the following configuration is pushed to the devices as shown below

!  
ip name-server 8.8.8.8 8.8.4.4  
no ip domain lookup  
ip domain name lab.net  
!  
ntp logging  
ntp server 172.20.1.17  
!

**Configuring Interfaces on IOS Devices**

In this recipe, we will outline how to configure the basic interface properties on Cisco IOS-based devices like setting the interface description, Interface MTU and enabling the interfaces. We will configure all the links within our topology as having Link MTU of 1500 and to be full duplex.

**Getting Ready**

To follow along with this recipe, an ansible inventory is assumed to be already setup and we will continue to build on the previous recipe to configure the interfaces as per our intended setup outlined in the start of this chapter.

**How to do it..**

* In the ***network.yml*** file (under group\_vars folder)  add the following content

$ cat group\_vars/network.yml  
<-----Snippet ------->  
intf\_duplex: full  
intf\_mtu: 1500

* Create a new file called ***lan.yml*** under the group\_vars folder with the following data

$ cat group\_vars/lan.yaml  
  
interfaces:  
 core01:  
 - name: Ethernet0/1  
 description: access01\_e0/1  
 mode: trunk  
 - name: Ethernet0/2  
 description: access02\_e0/1  
 mode: trunk  
 - name: Ethernet0/3  
 description: core01\_e0/3  
 mode: trunk  
<-------- Snippet ------------>  
 access01:  
 - name: Ethernet0/1  
 description: core01\_e0/1  
 mode: trunk  
 - name: Ethernet0/2  
 description: core02\_e0/1  
 mode: trunk  
 - name: Ethernet0/3  
 description: Data\_vlan  
 mode: access  
 vlan: 10  
<----------- Snippet ------------->

* Update the ***pb\_build\_network.yml*** file with the following tasks

[223] → cat pb\_build\_network.yml  
---  
- name: "PLAY 1: Configure All Lan Switches"  
 hosts: lan  
 tags: lan  
 tasks:  
<------------Snippet --------------------->  
 - name: "P1T3: Configure Interfaces"  
 ios\_interface:  
 name: "{{ item.name }}"  
 description: "{{ item.description }}"  
 duplex: "{{ intf\_duplex }}"  
 mtu: "{{ intf\_mtu }}"  
 state: up  
 loop: "{{ interfaces[inventory\_hostname] }}"  
 register: ios\_intf

**How it works..**

**There is More..**

**Configuring L2 VLANs on IOS Devices**

In this recipe, we will outline how to configure VLANs on Cisco IOS-based devices

**Getting Ready**

**How to do it..**

**How it works..**

**There is More..**

**Configuring Trunk and Access Interfaces**

In this recipe, we will outline how to configure how to setup Interfaces on Cisco IOS Devices as either Trunk or access ports and how to assign the Correct VLANs on them

**Getting Ready**

**How to do it..**

**How it works..**

**There is More..**

**Configuring Interface IP addresses**

**Getting Ready**

**How to do it..**

**How it works..**

**There is More..**

**Configuring OSPF on IOS Devices**

**Getting Ready**

**How to do it..**

**How it works..**

**There is More..**

**Configuring BGP on IOS Devices**

**Getting Ready**

**How to do it..**

**How it works..**

**There is More..**

**Testing Network Reachability on IOS Devices**

**Getting Ready**

**How to do it..**

**How it works..**

**There is More..**

**Collecting IOS Device facts**

**Getting Ready**

**How to do it..**

**How it works..**

**There is More..**

**Retrieving Operational data from IOS Devices**

**Getting Ready**

**How to do it..**

**How it works..**

**There is More..**

**Connecting to Cisco IOS Devices**

**Getting Ready**

**How to do it..**

**How it works..**

**There is More..**

## Configuring L2 VLANs on Cisco IOS

In this recipe we will outline how to configure L2 VLANs on Cisco IOS devices as per the network topology discussed in the intro in this chapter. We will configure the

### Getting Ready

We will be building on the pervious receipes discussed in this chapter to contunre to configrie the L2 VLANs on all the Lan devices within our sample topololgy.

### How to do it

* On ***the lan.yml*** file (under group\_vars folder) add the following

$ cat group\_vars/lan.yaml

vlans:

- name: Data

vlan\_id: 10

- name: Voice

vlan\_id: 20

- name: Web

vlan\_id: 100

* Update the ***pb\_build.yml*** playbook with the following task

---

- name: "PLAY 1: Configure All Lan Devices"

hosts: lan

tags: lan

connection: network\_cli

tasks:

🡨------- Snippet ------🡪

- name: "P1T4: Create L2 VLANs"

ios\_vlan:

aggregate: "{{ vlans }}"

register: ios\_vlans

tags: vlan

### How it is done

On the lan.yml file we define a vlans list data structure which holds all the VLANs we will need to configure on all our core and access switches. This variable will be available for all the core and access switches and ansible will use this variable in order to provision the required VLANs on the remote devices.

We use another declarative module called ***ios\_vlan*** which takes the vlan definition (its name and the vlan-id) and configure these VLANs on the remote managed device. Its pulls the existing configuration from the device and compare it with the list of devices that need to be present and only push the delta.

We use the loop construct to go through all the items in the vlans and configure all the respective vlans on all the devices.

After running this task on the devices below is the output from one of the access switches

## Configuring Access and Trunk Interfaces

In this recipe, we will show how to configure access and trunk interfaces on Cisco IOS-based devices and how to map interfaces to access vlan as well as how to allow specific vlans on the trunks. Following our sample toplolgy we will configure the interfaces on the devices as shown in thi table

|  |  |  |  |
| --- | --- | --- | --- |
| Device | Interface | Mode | Vlans |
| Core01 | Ethernet0/1 | Trunk | 10,20,100 |
| Core01 | Ethernet0/2 | Trunk | 10,20,100 |
| Core01 | Ethernet0/3 | Trunk | 10,20,100,200 |
| Access01 | Etherent0/1 | Trunk | 10,20,100 |
| Access01 | Ethenet0/2 | Trunk | 10,20,100 |
| Access01 | Ethernet0/3 | Access | 10 |

### Getting Ready

This recipe is a continuation for all the previous recipes in this chapter.

### How to do it

* On the lan.yml file under group\_vars folder add the following information

interfaces:

core01:

- name: Ethernet0/1

description: access01\_e0/1

mode: trunk

- name: Ethernet0/2

description: access02\_e0/1

mode: trunk

- name: Ethernet0/3

description: core01\_e0/3

mode: trunk

access01:

- name: Ethernet0/1

description: core01\_e0/1

mode: trunk

- name: Ethernet0/2

description: core02\_e0/1

mode: trunk

- name: Ethernet0/3

description: Data\_vlan

mode: access

vlan: 10

* Create a new ***core.yml*** file under group\_vars and include the following in it

core\_vlans:

- name: l3\_core\_vlan

vlan\_id: 200

interface: Ethernet0/3

* Update the pb\_build\_network.yml playbook with the following

---

- name: "PLAY 1: Configure All Lan Devices"

hosts: lan

tags: lan

tasks:

< ------- Snippet ----- >

- name: "P1T5: Configure L2 Trunks"

ios\_l2\_interface:

name: "{{ item.name }}"

mode: "{{ item.mode }}"

trunk\_allowed\_vlans: "{{ vlans | map(attribute='vlan\_id') | join(',') }}"

state: present

loop: "{{ interfaces[inventory\_hostname] | selectattr('mode','equalto','trunk') | list }}"

- name: "P1T6: Enable dot1q Trunks"

ios\_config:

lines:

- switchport trunk encapsulation dot1q

parents: interface {{item.name}}

loop: "{{ interfaces[inventory\_hostname] | selectattr('mode','equalto','trunk') | list }}"

tags: dot1q

- name: "P1T7: Configure Access Ports"

ios\_l2\_interface:

name: "{{ item.name }}"

mode: "{{ item.mode}}"

access\_vlan: "{{ item.vlan }}"

state: present

loop: "{{ interfaces[inventory\_hostname] | selectattr('mode','equalto','access') | list }}"

### How it is done

### There is More

## Configuring IP addresses on the Interfaces

### Getting Ready

### How to do it

### How it is done

### There is More

## Configuring OSPF

### Getting Ready

### How to do it

### How it is done

### There is More

## Configuring BGP

### Getting Ready

### How to do it

### How it is done

### There is More

## Collecting IOS facts

### Getting Ready

### How to do it

### How it is done

### There is More

## Running Operational Commands on IOS Devices

### Getting Ready

### How to do it

### How it is done

### There is More