**Cisco switch config through Ansible.**

Ansible is a highly useful network automation and configuration tool. It uses python scripts to deploy configurations on a wide variety of networking devices and is completely agentless. More and more companies opt for an automated way to manage network devices due to the time and effort that can be saved.

**Pre-requisites** -

It is helpful if you have some experience with the Cisco CLI. If not SSH is simple to set-up so some quick research should give you all you need.

**SSH connection** -

Ansible is agentless so no software needs to be installed on the desired hardware. An SSH connection is however required for Ansible to pass configurations to the device. The following steps are a reminder on how to configure SSH on a switch.

1. Wire into a switch.
   1. Find your console cable number (light blue cable behind your pc)
      1. Wire from that number in the cabinet into the console port of an unused switch.
   2. Find your Ethernet number (darker blue cable behind your pc)
      1. Run a wire from that number into one of the front interfaces on the switch.
2. Open a PUTTY session on your host lab PC.
   1. Under the session tab choose the serial option and COM1 should automatically be entered into the text box.
   2. Click open.
3. Configure SSH on the switch.
   1. Enter privileged exec mode.
   2. Enter config mode.
   3. Enable a secret password of “cisco” (must be configured to SSH in as a client).
   4. Enter interface vlan1 (default management vlan).
   5. Enter an ip address and subnet mask.
   6. Perform the no shutdown command.
   7. Exit back to privileged exec mode.
   8. Configure an ip domain name.
   9. Specify SSH version 2.
   10. Add an SSH username//password of admin//cisco.
   11. Enter line vty 0 4.
   12. Set login local.
   13. Set transport input as SSH.
   14. Perform a show run to check the configurations.
4. Test connection.
   1. On a client VM that operates on the same network as the switch open up a terminal/console.
   2. Attempt to ping the ip you assigned to the switch to confirm connectivity.
   3. SSH into the switch with the ($ ssh username@ipaddress).
      1. Enter any usernames and passwords as prompted.

**Inventory file** -

Now that we know the SSH connection is there we can start to create/edit the relevant Ansible files. The inventory file is the /etc/ansible/hosts file by default. In practice you would create a new one and point to it so that each playbook has its own set of inventory rules. For the purposes of this lab we will use the default file for simplicity.

The inventory file contains host information and can contain configurations.

1. Open the hosts file with a text editor (nano, vi...etc).
2. Edit the File to contain your switched information as followed.
   1. [cisco]` # Group name that can be called later to affect all devices in the group.
   2. 192.168.xx.xx # The ip address of your switch.
   3. [cisco:vars] # Define variables for the cisco group.
   4. ansible\_python\_intepreter=/usr/bin/python # optional line as the VM’s in the lab don't seem to contain the correct path for python. Without this you can perform some functions like the ansible ‘ping’ module (ansible -m ping all).
   5. Save this file.

**Config file** -

By default, the config file is /etc/ansible/ansible.cfg. It contains configurations and therefore can also be replicated for different tasks and pointed to in the script. We will once again use the default for simplicity.

1. Open the ansible.cfg file with a text editor.
2. Edit the file with some simple configurations (Note; All of the commented out values are defaults).
   1. [defaults] # This line is located near the top of the document.
   2. host\_key\_checking = false # Saves logging into the SSH connection every time (bad practice in the real world due to man in the middle attacks).
   3. Timeout = 20 # Extends the default timeout slightly.

**Playbook** -

Playbooks are the scripts that Ansible uses and are YAML/.yml/Yet another markup language files.

Ansible is python based. Below is an example of a simple script to change global configurations on a Cisco switch. (Note: Ansible does not use tabs, only spaces with work. Also the structure is important to take note of where lines are indented to.

1. In the /etc/ansible directory create another folder called playbooks.
2. Within the playbooks folder create your first .yml file (Example: Cisco\_test1.yml)
3. Add the following text to the .yml file.

--- # All YAML files start with three dashes

- name: Cisco switch config # Name of the play, can be whatever you want.

hosts: cisco # The group from the hosts file. Can be a group, all or single hosts.

gather\_facts: false # Declares that we don’t need information from the switch.

Connection: local # Sets it as a local connection.

vars: # Variables are declared in this section.

cli: # Declares the name of the variable.

username: admin # SSH username.

password: cisco # SSH password.

Tasks: # One playbook can have multiple tasks.

- name: Global config settings # Name of the task, can be anything.

Ios\_config: # A module prebuilt into ansible (you can make your own modules).

provider: “{{ cli }}” # Calls the variable that was declared earlier.

authorize: yes # Enters into privileged exec mode on the switch.

auth\_pass: cisco # Enable level password to access privileged exec mode.

Lines: # The config lines that will be sent to the switch.

- no ip http server # Turns the http server service off.

register: print\_output # Saves the value as print output.

- debug var=print\_output # Prints the outcome of the above task to the terminal.

1. Save the file.

**Test** -

Now we need to test the script. In order to do this, open a terminal/console on the client VM and run the following command.

1. $ ansible-playbook cisco\_test1.yml # If you saved the yam file under a different name use that instead. (Note: run from inside the same directory or specify the file path before the name.)

You can watch as the playbook runs through. Hopefully at the end you will have no errors and the status will show as ok and changed.

If you wish to the go further extra lines can be added to playbook for almost all configurations including physical and virtual port configuration.

Note: if you run a playbook twice in a row it won’t show as changed as it actively tests whether a change can be made and does nothing if it is unnecessary.

**Summary** -

As I’m sure you can see setting up a playbook may seem long compared to connecting to a switch and manually inputting the data when it comes to changing one setting, but imagine if you had 100 switches to fully configure. Ansible would allow you to add all 100 of those switches to the host file and run the script throughout your entire network in one go. You could even add a second hosts group for routers and add a second task in the playbook that only affects the routers group and fully configured them as well. As you can imagine for a network administrator/engineer this could potentially save hundreds of hours of configuration.