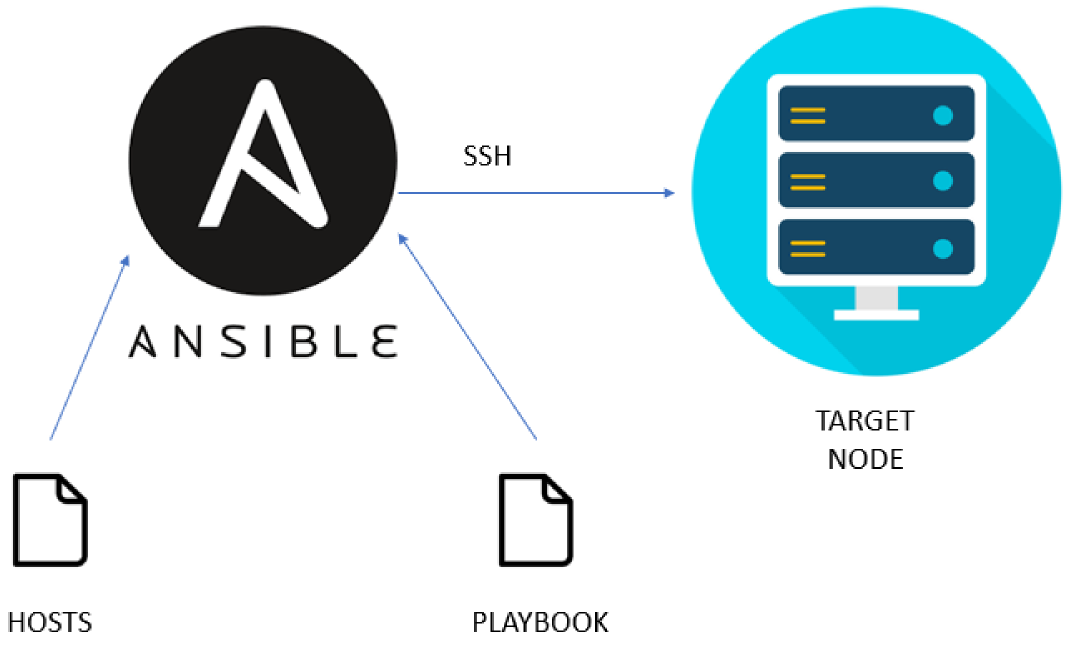
Introduction to Ansible

Ansible is an agentless automation that automates deployment, configuration management (maintain infrastructure consistency) and orchestration (execution of multiple applications in order). Ansible gains it's popularity due to it's simplicity for being agentless, efficient, requires no additional software installed on target machine, use the simple YAML and complete with reporting.



Ansible architecture is very simple. It requires Ansible Server basically a node (laptop, PC or server) where Ansible is installed with the module of configuration files called **playbook** and inventory of target servers called **hosts**. Playbook consists of Roles, and Roles consists of Tasks. Task is an individual command in Ansible. By using inventory we group the nodes by using labels.

Ansible Server and the node talks by using passwordless SSH.

Flow of working with Ansible:

1. Create playbook and inventory in local machine
2. Create SSH to the target nodes
3. Ansible Server gathers the facts of the target nodes to get the indication of the target nodes
4. Playbook are sent to nodes
5. Playbook are executed in the nodes

Important Terms

Ansible server: The machine where Ansible is installed and from which all tasks and playbooks will be ran
Module: Basically, a module is a command or set of similar commands meant to be executed on the client-side
Task: A task is a section that consists of a single procedure to be completed
Role: A way of organizing tasks and related files to be later called in a playbook
Fact: Information fetched from the client system from the global variables with the gather-facts operation
Inventory: File containing data about the ansible client servers. Defined in later examples as hosts file
Play: Execution of a playbook
Handler: Task which is called only if a notifier is present
Notifier: Section attributed to a task which calls a handler if the output is changed
Tag: Name set to a task which can be used later on to issue just that specific task or group of tasks.

Setup Ansible Server

STEP 1 — Setup Ansible Server

Install Ansible and dependencies

First is Python 3.8

```
sudo apt install python3.8-venv
python3 –version
```

If your Python version returned from above is less than 3.8 then:

```
sudo update-alternatives --install /usr/bin/python3 \
python3 /usr/bin/python3.8 1
python3 –version
```

Install other prerequisite package of Python:

```
apt install --no-install-recommends python3-netaddr python3-
ipaddr
```

Next we will install the Ansible

```
$ sudo apt-get update
$ sudo apt-get -y install software-properties-common
$ sudo apt-add-repository ppa:ansible/ansible
$ sudo apt-get update
$ sudo apt-get install -y ansible
```

STEP 2— Setup SSH Connection to Target Server

Config Router to access via SSH:

```
R3(config)#username lab privilege 15 secret apnic
R3(config)#ip domain name summitiig.net
R3(config)#crypto key generate rsa
The name for the keys will be: R3.summitiig.net
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]: 1048
% Generating 512 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 0 seconds)

R3(config)#
*Apr 16 14:45:22.991: %SSH-5-ENABLED: SSH 1.99 has been enabled
R3(config)#ip ssh version 2
R3(config)#line vty 0 4
R3(config-line)# login local
```

Try to access Router via SSH from Server.
If you are having those type of error:

```
# ssh lab@100.68.3.5
Unable to negotiate with 100.68.3.5 port 22: no matching key
exchange method found. Their offer: diffie-hellman-group-
exchange-sha1,diffie-hellman-group14-sha1,diffie-hellman-group1-
sha1
```

You can solve this by configuring non standard client options by creating a file in /etc/ssh/ssh_config.d/:

```
#echo "KexAlgorithms diffie-hellman-group-exchange-sha1,diffie-
hellman-group14-sha1" >>/etc/ssh/ssh_config.d/weak.conf
#echo "Ciphers aes128-cbc" >>/etc/ssh/ssh_config.d/weak.conf
```

Test SSH connection:

```
# ssh lab@100.68.3.5
The authenticity of host '100.68.3.5 (100.68.3.5)' can't be
established.
RSA key fingerprint is
SHA256:Fh6of2DmmPp9dF21n+ztSPguKvWvKkSj50PTbmCk5rA.
Are you sure you want to continue connecting
(yes/no/[fingerprint])? yes
Warning: Permanently added '100.68.3.5' (RSA) to the list of
known hosts.
Password:
```

STEP 3-Edit hosts file

Edit hosts file on /etc/ansible/hosts and add your target server

```
# mv /etc/ansible/hosts /etc/ansible/hosts.old

# vim /etc/ansible/hosts
[ios_router]
R3 ansible_host=100.68.3.5
[ios_router:vars]
ansible_ssh_user=lab
ansible_ssh_pass=apnic
ansible_connection=network_cli
ansible_network_os=ios
```

STEP 4-Create your first Ansible Playbook

Playbooks are text files written in YAML format and therefore need:

- to start with three dashes (---)
- proper indentation using spaces and **not** tabs!
- to start with three dots (…)

In this example we are going to automate a interface configuration:

```
vim interface.yml
---
- name: "SET IP ADDRESS ON ACCESS ROUTER"
  hosts: R3
  become: yes
  become_method: enable
  tasks:
    - name: "SET IP ADDRESS ON ACCESS ROUTER"
```

```
cisco.ios.ios_config:
  parents: "interface FastEthernet0/0"
  lines:
    - description TO-CUST-01
    - ip address 10.0.0.0 255.255.255.254
    - ipv6 address 2001:db8:1::/127
  after: "no shutdown"
```

STEP 5-Testing and Running Playbook

```
$ ansible-playbook --syntax-check interface.yml
$ ansible-playbook interface.yml
```

Some points to avoid common errors in Ansible YAML:

- Use consistent indentation: YAML relies on indentation to define the structure of the file, so make sure to use consistent indentation throughout your playbook. Typically, two spaces are used for indentation in Ansible YAML.
- Be mindful of colons and hyphens: Colons (`:`) are used to denote key-value pairs in YAML, while hyphens (`-`) are used to denote list items. Make sure to use them correctly and consistently.
- Check for proper syntax: YAML is a strict markup language, so ensure that your playbook adheres to the correct YAML syntax. Use tools like `ansible-lint` or online YAML validators to check for syntax errors.
- Use quotes for strings with special characters: If your string contains special characters like spaces, colons, or square brackets, make sure to enclose it in single or double quotes to avoid parsing errors.
- Validate module parameters: Each Ansible module has specific parameters and syntax requirements. Make sure to refer to the module documentation and use the correct parameters and values in your playbook.
- Use appropriate data types: YAML supports various data types such as strings, numbers, lists, and dictionaries. Use the appropriate data type for each parameter or value in your playbook to avoid type mismatch errors.
- Avoid mixing tabs and spaces: YAML can be sensitive to mixing tabs and spaces for indentation. Stick to using spaces for indentation to avoid indentation-related errors.
- Double-check host and variable names: Make sure to use the correct host and variable names in your playbook. Typos or mismatched names can lead to errors or unexpected results.
- Test thoroughly: Always test your playbook on a test environment before running it in production. This helps identify and fix any potential errors or issues before affecting your production environment.

Following these best practices can help you avoid common errors and ensure smooth execution of your Ansible playbooks.

Route-map in Ansible:

Create 1st Route-map by using `cisco.ios.ios_route_maps` module which will generate Cisco config like:

```
route-map test_1 permit 10
match ip address prefix-list default

route-map test_1 permit 20
match ip address prefix-list BOGONS

route-map test_1 permit 30
match ip address prefix-list test_1_new
match rpki valid
set local-preference 100

route-map test_1 permit 40
match ip address prefix-list test_1_new
match rpki not-found
set local-preference 100

route-map test_1 deny 100
```

Create a playbook routemap.yml:

```
---
- name: "PLAY 1: Setup route map"
  connection: network_cli
  hosts: R3
  become: yes
  become_method: enable
  tasks:
    - name: Merge provided Route maps configuration
      cisco.ios.ios_route_maps:
        config:
          - route_map: test_1
            entries:
              - sequence: 10
                action: deny
                match:
                  ip:
                    address:
                      prefix_lists:
                        - default
              - sequence: 20
```

```
                action: deny
                match:
                  ip:
                    address:
                      prefix_lists:
                        - BOGONS
              - sequence: 30
                action: permit
                match:
                  ip:
                    address:
                      prefix_lists:
                        - test_1_new
                  rpki:
                    valid: true
                set:
                  local_preference: 100
              - sequence: 40
                action: permit
                match:
                  ip:
                    address:
                      prefix_lists:
                        - test_1_new
                  rpki:
                    not_found : true
                set:
                  local_preference: 100
              - sequence: 100
                action: deny

...
```

Check and run:

```
$ ansible-playbook --syntax-check routemap.yml
$ ansible-playbook routemap.yml
```

Lets make it to take "test_1" and "test_1_new" values from the command line in your Ansible playbook, you can use the vars_prompt section to prompt the user for input, and then use those input values in your playbook:

```
Vim routemap-r3.yml

---
- name: "PLAY 1: Setup route map"
  connection: network_cli
  hosts: R3
  vars_prompt:
    - name: route_map_name
      prompt: "Enter the route map name: "
      private: false
    - name: prefix_list_name
      prompt: "Enter the prefix list name: "
      private: false
  vars:
    route_map: "{{ route_map_name }}"
    prefix_list: "{{ prefix_list_name }}"
  tasks:
    - name: Merge provided Route maps configuration
      cisco.ios.ios_route_maps:
        config:
          - route_map: "{{ route_map }}"
            entries:
              - sequence: 10
                action: deny
                match:
                  ip:
                    address:
                      prefix_lists:
                        - default
              - sequence: 20
                action: deny
                match:
                  ip:
                    address:
                      prefix_lists:
                        - BOGONS
              - sequence: 30
                action: permit
                match:
                  ip:
                    address:
                      prefix_lists:
                        - "{{ prefix_list }}"
                  rpki:
                    valid: true
                set:
                  local_preference: 1000
              - sequence: 40
                action: permit
                match:
                  ip:
                    address:
                      prefix_lists:
                        - "{{ prefix_list }}"
                  rpki:
                    not_found : true
                set:
                  local_preference: 1000
              - sequence: 100
                action: deny
```

Check and run:

```
$ ansible-playbook --syntax-check routemap-r3.yml
$ ansible-playbook routemap-r3.yml
```

Now create a new playbook routemap-r3-v6.yml for IPv6 route-map. Just use “ipv6” instead of “ip”

BGP in Ansible:

Create bgp config by using `ios_config` module which will generate Cisco config like:

```
router bgp 65002
 neighbor 100.68.3.2 remote-as 132884
 neighbor 100.68.3.2 description AS132884
 address-family ipv4
  neighbor 100.68.3.2 activate
  neighbor 100.68.3.2 soft-reconfiguration inbound
  neighbor 100.68.3.2 maximum-prefix 10 warning-only
  neighbor 100.68.3.2 route-map CLIENT-IN in
  neighbor 100.68.3.2 route-map DEFAULT out
```

Where Peer IP (100.68.3.2), remote-as (132884), description, in and out route-map name will take from the command line:

```
nano ebgp-r3.yml
---
- name: "PLAY 1: Setup iBGP Peer to R4"
  hosts: R3
  connection: network_cli
  become: yes
  become_method: enable
  vars_prompt:
    - name: peer_ip
      prompt: "Enter Neighbor IP"
      private: false
    - name: peer_asn
      prompt: "Enter the remote-as"
      private: false
    - name: peer_des
      prompt: "Enter the Neighbor description"
      private: false
    - name: prefix_limit
      prompt: "Enter Prefix limit for this Neighbor"
      private: false
    - name: peer_map_in
      prompt: "Enter the route-map in name"
      private: false
    - name: peer_map_out
      prompt: "Enter the route-map out name"
      private: false
  tasks:
    - name: "TASK 1: Configure BGP Peer"
      ios_config:
        commands:
          - "router bgp 65002"
          - " neighbor {{ peer_ip }} remote-as {{ peer_asn }}"
          - " neighbor {{ peer_ip }} description {{ peer_des }}"
          - " address-family ipv4"
          - "  neighbor {{ peer_ip }} activate"
          - "  neighbor {{ peer_ip }} soft-reconfiguration inbound"
          - "  neighbor {{ peer_ip }} maximum-prefix {{ prefix_limit }}
warning-only"
          - "  neighbor {{ peer_ip }} route-map {{ peer_map_in }} in"
          - "  neighbor {{ peer_ip }} route-map {{ peer_map_out }} out"
      register: bgp_setup

    - name: "SUMMARY TASK: Debug output"
      debug:
        var: bgp_setup
```

Check and run:

```
$ ansible-playbook --syntax-check ebgp-r3.yml
$ ansible-playbook ebgp-r3.yml
```

Now create a new playbook `ebgp-r3-v6.yml` for IPv6 route-map. Just use “ipv6” instead of “ipv4” in address-family