Ansible

Ansible is an open-source IT automation engine that automates cloud provisioning, configuration management, application deployment, intraservice orchestration, and other IT needs. Similar to Puppet, and Chef, Ansible enables administrators to manage, automate, and orchestrate various types of server environments. Ansible is agentless, and does not require a software agent to be installed on the target node (server or switch) in order to automate the device. By default, Ansible requires SSH and Python support on the target servers it manages but for MDS switches, Ansible was extended to use both SSH and NX-API and on-switch Python support is not required. Ansible playbooks are written in YAML, that allows you to describe your automation jobs in an easily readable format. Inside each Ansible playbook, we can use various Ansible modules.

The following are MDS specific modules within the cisco.nxos collection:

- nxos_vsan (https://docs.ansible.com/ansible/latest/collections/cisco/nxos/nxos_vsan_module.html#ansible-collections-cisco-nxos-nxos-vsan-module)
- nxos_zone_zoneset (https://docs.ansible.com/ansible/latest/collections/cisco/nxos/nxos_zone_zoneset_module.html#ansible-collections-cisco-nxos-nxos-zone-zoneset-module)
- nxos_devicealias (https://docs.ansible.com/ansible/latest/collections/cisco/nxos/nxos_devicealias_module.html#ansible-collections-cisco-nxos-nxos-devicealias-module)
- nxos_fc_interfaces (https://docs.ansible.com/ansible/devel/collections/cisco/nxos/nxos_fc_interfaces_module.html#ansible-collections-cisco-nxos-nxos-fc-interfaces-module)

To run any arbitrary command, use the nxos_command module (https://docs.ansible.com/ansible/latest/collections/cisco/nxos/nxos_command_module.html#ansible-collections-cisco-nxos-nxos-command-module)

There are also other modules which have limited support for Cisco MDS. See individual module documentation for compatibility with Cisco MDS: https://docs.ansible.com/ansible/latest/collections/cisco/nxos/index.html#modules

Ansible modules make SSH connections or NX-API calls to gather real-time state data and to make configuration changes on the Cisco MDS devices. For more information about Ansible, see Ansible's official documentation.

Note For Cisco MDS Ansible modules, you do not need a Python interpreter on the target node.

For more information on the Cisco MDS modules supported on Ansible, see the Ansible Modules.

- Getting Started, on page 2

Getting Started

For information on Ansible installation, refer to the official Anisble installation guide https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html.

Host File

The host file is where the devices under management are listed. A single device can be in a single group or included in multiple groups. In the below host file, a single group called edge, which has 2 devices, mds1 and mds2 are used. The connection is set to NX-API which uses HTTPS connection to connect to target devices. The username and password is stored in the host file. For further security, this host file can be encrypted using the Ansible Vault, which we are not utilizing here.

Note From Ansibe 2.5, *ansible_connection: local* is deprecated. Use `ansible_connection: ansible.netcommon.network_cli` or `ansible_connection: ansible.netcommon.httpapi` instead. The httpapi connection plugin provides a variety of toggles. All the available options are specified in the Ansible documentation. The network_cli connection plugin provides a variety of toggles. All the available options are specified are specified in the Ansible documentation.

```
$ cat /etc/ansible/hosts
[all:vars]
ansible_connection = ansible.netcommon.httpapi
ansible_httpapi_use_ssl=True
ansible_httpapi_port=8443
ansible_user=username
ansible_password=password

[edge]
mds1
mds2
```

Documentation

Documentation for all Cisco MDS NX-OS modules can be found at https://docs.ansible.com/ansible/latest/collections/cisco/nxos/ or alternatively from the terminal, by utilizing the inbuilt documentation tool.

```
$ansible-doc
```

Example Playbook

In this initial playbook, we will provision a couple of VSANs and also delete a VSAN. We will use the Ansible module called **nxos_vsan** to automate this task.

As you can see below, the playbook is defined in YAML.

```
---
- name: Test that vsan module works
  gather_facts: no
  hosts:
  - mds1
  cisco.nxos.nxos_vsan:
    vsan:
    - id: 922
      interface:
      - fc1/1
      - fc1/2
      - port-channel 1
      name: vsan-SAN-A
      remove: false
      suspend: false
    - id: 923
      interface:
      - fc1/11
      - fc1/21
      - port-channel 2
      name: vsan-SAN-B
      remove: false
      suspend: true
    - id: 1923
      name: vsan-SAN-Old
      remove: true
  register: result
- debug: var=result
```

Assuming the above playbook is called **vsan.yml**, this task can then be run from the terminal as shown below.

```
$ ansible-playbook vsan.yml

PLAY [VSAN TEST (NXOS)] ***************************************************

TASK [Test that vsan module works] ****************************************
changed: [mds1]

TASK [debug] **************************************************************
ok: [mds1] => {
    "result": {
        "changed": true,
        "cmds": [
            "terminal dont-ask",
            "vsan database",
            "vsan 922",
            "vsan 922 name vsan-SAN-A",
            "no vsan 922 suspend",
            "vsan database",
            "vsan 922 interface fc1/1",
            "vsan 922 interface fc1/2",
            "vsan 922 interface port-channel 1",
            "vsan database",
            "vsan 923",
            "vsan 923 name vsan-SAN-B",
            "vsan 923 suspend",
            "vsan database",
            "vsan 923 interface fc1/11",
            "vsan 923 interface fc1/21",
            "vsan 923 interface port-channel 2",
            "vsan database",
            "no vsan 1923",
            "no terminal dont-ask"
```

```
        ],
        "failed": false,
        "messages": [
            "creating vsan 922",
            "setting vsan name to vsan-SAN-A for vsan 922",
            "no suspending the vsan 922",
            "adding interface fc1/1 to vsan 922",
            "adding interface fc1/2 to vsan 922",
            "adding interface port-channel 1 to vsan 922",
            "creating vsan 923",
            "setting vsan name to vsan-SAN-B for vsan 923",
            "suspending the vsan 923",
            "adding interface fc1/11 to vsan 923",
            "adding interface fc1/21 to vsan 923",
            "adding interface port-channel 2 to vsan 923",
            "deleting the vsan 1923"
        ]
    }
}

PLAY RECAP *********************************************************************
mds1                : ok=2    changed=1    unreachable=0    failed=0
```

Conclusion

We have just seen how we can configure/unconfigure VSANs using Ansible, this is a simple example but the modules can be used in a variety of tasks that needs automation.