In all the previous chapters in this book, we have been describing our network infrastructure using ansible variables stored in YAML files. This approach is acceptable however it is not the optimum solution when we try to adopt automation across the organization. We need to have our network inventory, IP addresses , VLANs in a central system which will act as the authoritative source of truth for our network. This system should have a robust and powerful API which can be queried by other automation and OSS/BSS systems to retrieve and update the network inventory.

NetBox is an open source Inventory system for the network infrastructure which was initially developed by the network engineering team at Digital Ocean to document their data center infrastructure. It is a simple yet powerful and highly extensible inventory system which can provide the capability to be the network source of truth for different information that we need to document regarding our network. It provides the capability to document and describe the below features on any Network infrastructure

* **IP address management (IPAM)** - IP networks and addresses, VRFs, and VLANs
* **Equipment racks** - Organized by group and site
* **Devices** - Types of devices and where they are installed
* **Connections** - Network, console, and power connections among devices
* **Virtualization** - Virtual machines and clusters
* **Data circuits** - Long-haul communications circuits and providers
* **Secrets** - Encrypted storage of sensitive credentials

NetBox is a Django based python application which uses PostgresSQL as a backend data storage and it uses NGINX as the front end web server along with other optional components that work together in order to deliver the NetBox system. It has a powerful REST API which can be used to retrieve/update the data stored in the Netbox database.  
In this chapter we will outline the following three main uses cases for the integration between Ansible and NetBox

* Using Ansible to populate the data in NetBox for the different network information that is modeled by NetBox like sites, Devices, IP addresses.
* Using NetBox as the dynamic Inventory source for NetBox to retrieve and build ansible inventory
* Using NetBox as the source of information for data needed by ansible in order to provision and configure network devices.

We are going to use a sample network composed of two data center sites with a Spine/leaf Fabric in each site. We will model all the information and populate it into netbox. The below table captures this sample network infrastructure

|  |  |  |
| --- | --- | --- |
| **Site** | **Device** | **Role** |
| DC1 | dc1-spine01 | Spine Switch |
| DC1 | dc1-spine02 | Spine Switch |
| DC1 | dc1-leaf01 | Leaf Switch |
| DC1 | dc1-leaf02 | Leaf Switch |
| DC2 | dc2-spine01 | Spine Switch |
| DC2 | dc2-spine02 | Spine Switch |
| DC2 | dc2-leaf01 | Leaf Switch |
| DC2 | dc2-leaf02 | Leaf Switch |

The main recipes covered in this chapter is shown below

* Installing NetBox
* Integrating NetBox with Ansible.
* Populating Sites on NetBox.
* Populating Devices on NetBox.
* Populating Interfaces on NetBox.
* Populating IP addresses on NetBox.
* Populating IP Prefixes on NetBox.
* Using NetBox as Dynamic Inventory Source for Ansible
* Generating Configuration using NetBox Data.

# Technical Requirements

All the code presented in this chapter can be found in the below Git Repo

<https://github.com/PacktPublishing/Network-Automation-Cookbook/tree/master/ch11_netbox>

Below are the Software releases that this chapter is based on

· Ansible Machine Running CentOS 7.7

· Ansible 2.8.5

. Python 3.6.8

· Arista vEOS running EOS 4.20.1F

. NetBox v2.6.5

# **Installing NetBox**

In this recipe we will outline how to install NetBox using docker containers and how to start all the required containers to have a functional NetBox server. Using docker containers to install NetBox is the simplest approach to get started with NetBox.

## **Getting Ready**

In order to start installing the NetBox on a Linux machine, this machine needs to have internet connectivity to pull the required docker image required for NetBox operation from docker Hub.

## **How to do it..**

1. Install Docker on the CentOS Linux machine as per the below URL.

[https://docs.docker.com/install/linux/docker-ce/Debian/](https://docs.docker.com/install/linux/docker-ce/ubuntu/)

1. Install Docker compose following the instructions in the below URL

<https://docs.docker.com/compose/install/>

1. Clone the netbox repo as shown below into a new directory (netbox\_src)

$ git clone <https://github.com/netbox-community/netbox-docker.git> netbox\_src

1. Change to the directory netbox\_src and pull all the required docker images using docker-compose as shown below

$ cd netbox\_src

$ /usr/local/bin/docker-compose pull

1. Start all the docker containers as shown below

$ /usr/local/bin/docker-compose up -d

## **How it works..**

As outlined in this chapter introduction, NetBox is composed of multiple applications that integrate together in order to deliver the required netbox application. The most simple installation method for netbox is using docker containers and to use a single docker-compose definition file to describe the interaction between the different docker services in order to deliver the netbox application.

In this recipe we described the steps required to install netbox using docker and docker-compose which greatly simplifies the installation steps to have a functional netbox server. The developers behind the netbox created the required docker images needed to run the netbox using docker and have created the required docker compose file that describe the overall interaction between the different netbox components in order to setup a netbox server. All the netbox setup instructions along with the Dockerfiles and docker-compose file to build and deploy netbox using docker containers is maintained in the below URL.

<https://github.com/netbox-community/netbox-docker>

## **There's More ...**

In order to simplify the installation of netbox I have created an ansible role within this chapter code to deploy netbox. In order to utilize this role we need to perform the following

1. On the ansible control machine clone this chapter code
2. Update the hosts file with the correct IP address for your netbox server

$ cat hosts

**< --- Output omitted for bevitry --- >**

[netbox]

netbox ansible\_host=172.20.100.32

1. Run the ansible playbook pb\_deploy\_netbox

$ ansible-playbook pb\_deploy\_netbox.yml

## **See Also ...**

For more information regarding How to install NetBox using Docker containers please check the below URL

[**https://github.com/netbox-community/netbox-docker**](https://github.com/netbox-community/netbox-docker)

# **Integrating NetBox with Ansible**

In this recipe, we will outline how to integrate between Ansible and NetBox via NetBox API. This integration is mandatory as it will allow us to populate NetBox Database from using Ansible playbooks as well as use NetBox as our dynamic inventory source to Build Ansible inventory in later recipes.

## **Getting Ready**

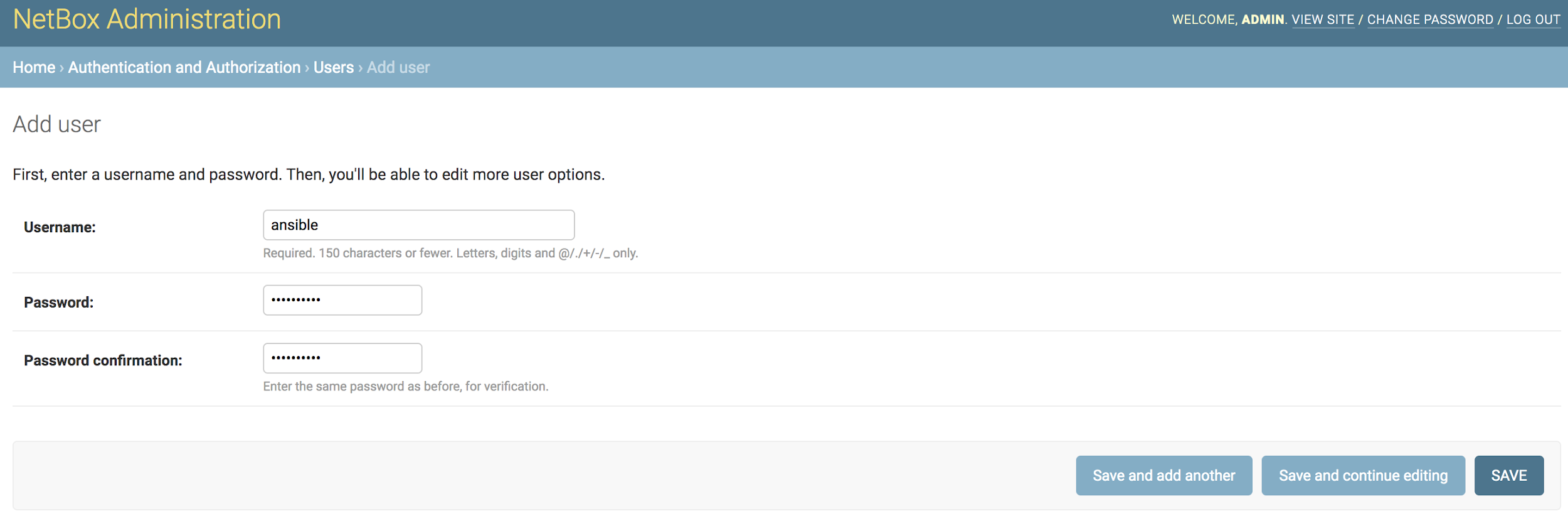
NetBox should be installed as outlined in the previous recipe and IP reachability need to present between the ansible control machine and the NetBox server. Ansible will communicate for NetBox over port 80 so port 80 needs to be open on the NetBox server

## **How to do it..**

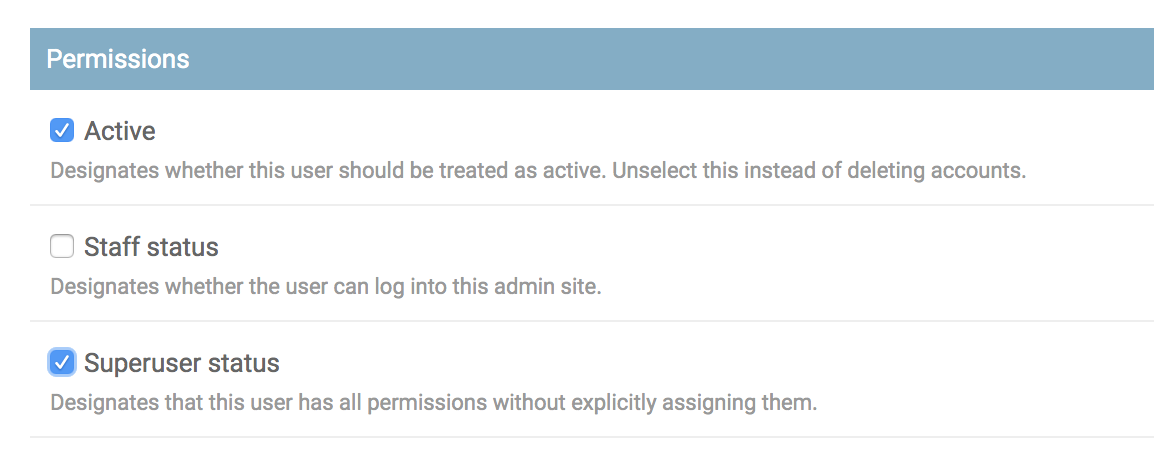
1. On the Ansible control machine install the pynetbox python package

$ sudo pip install pynetbox

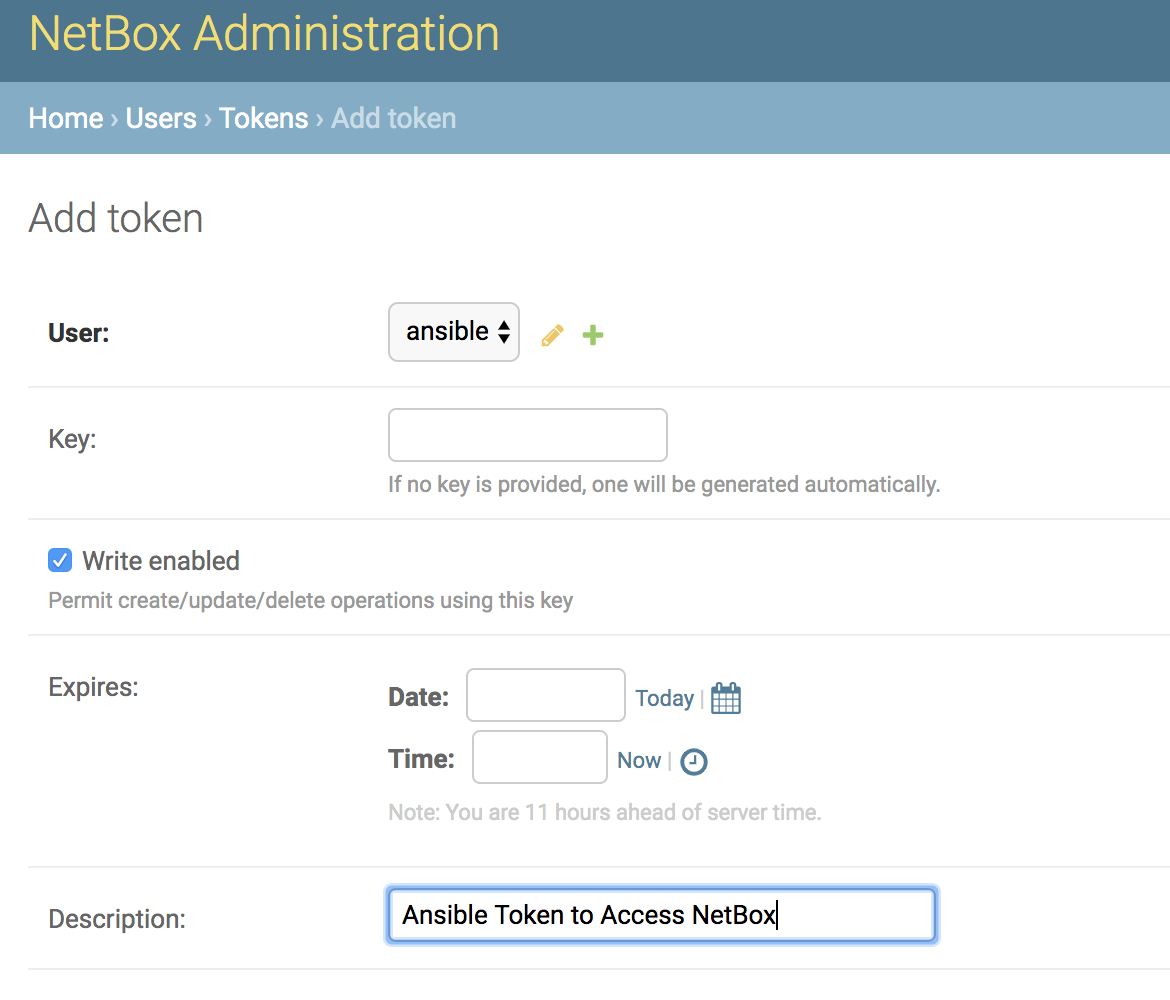
1. Login to the NetBox server using the Admin User and change to the admin tab to create a new User as shown below



1. Assign this new User superuser privileges to be able to write to NetBox DB



1. Create a New Token for this new User



1. In our project directory ch11\_netbox create our ansible inventory file hosts as shown below

$ cat hosts

[dc1]

dc1-spine01 ansible\_host=172.20.1.41

dc1-spine02 ansible\_host=172.20.1.42

dc1-leaf01 ansible\_host=172.20.1.35

dc1-leaf02 ansible\_host=172.20.1.36

[dc2]

dc2-spine01 ansible\_host=172.20.2.41

dc2-spine02 ansible\_host=172.20.2.42

dc2-leaf01 ansible\_host=172.20.2.35

dc2-leaf02 ansible\_host=172.20.2.36

[leaf]

dc[1:2]-leaf0[1:2]

[spine]

dc[1:2]-spine0[1:2]

1. Create the ***group\_vars*** folder and create the ***all.yml*** file and populate it as shown below

---

netbox\_url: http://172.20.100.32:8080

netbox\_token: e3afd6708f2a6cb45413c31c713ce6a146d69075

## **How it works..**

In this recipe we are setting up the integration between Ansible and NetBox. In order to start using the Ansible Modules to populate the database of Netbox we need to install the python module ***pynetbox***, This module is mandatory for all the netbox ansible modues that we are going to use in this chapter.

On the Netbox side we start by creating a a new user with complete admin rights on the netbox. This will grant this user full prvilage to create/edit/delete any object within netbox database. Next step is to create a token which will be used to authenticate all the API requrests from ansible towards NetBox.

Finally we create our ansible inventory and declate two parameters in our ansible varaibles which are the netbox\_url and netbox\_token to hold the API endpoint as well as the token for the ansible user on netbox.

## **See Also ...**

For more information regarding pybatfish and the Ansible Roles developed by Batfish to be used with ansible please check the below URLs.

<https://github.com/batfish/batfish/blob/master/README.md>

# **Populating Sites in NetBox**

In this recipe we will outline how to Create Sites into NetBox. Sites is a logical construct within netbox which allow us to group our infrastructure based on physical location. We need to define our sites in order to start to declare our devices and place them in these sites.

## **Getting Ready**

Integration between Ansible and NetBox is in place as outlined in the previous recipe.

## **How to do it..**

1. Update the ***group\_vars/all.yml*** file with the below sites data

sites:

- name: DC1

description: "Main Data Center in Sydney"

location: Sydney

- name: DC2

description: "Main Data Center in KSA"

location: Riyadh

1. Create a new ***roles*** directory under the ch11\_netbox.
2. Create a new ansible role called ***build\_netbox\_db*** and populate the ***tasks/main.yml*** file as shown below.

$ cat roles/build\_netbox\_db/tasks/main.yml

---

- name: Create NetBox Sites

netbox\_site:

netbox\_token: "{{ netbox\_token }}"

netbox\_url: "{{ netbox\_url }}"

data:

name: "{{ item.name | lower }}"

description: "{{ item.description | default(omit) }}"

physical\_address: "{{ item.location | default(omit) }}"

state: "{{ netbox\_state }}"

loop: "{{ sites }}"

run\_once: yes

tags: netbox\_sites

1. Update the ***defaults/main.yml*** file with the below data

$ cat roles/build\_netbox\_db/defaults/main.yml

---

netbox\_state: present

1. Create a new playbook called pb\_build\_netbox\_db.yml with the below contents

$ cat pb\_build\_netbox\_db.yml

---

- name: Populate NetBox DataBase

hosts: all

gather\_facts: no

vars:

ansible\_connection: local

tasks:

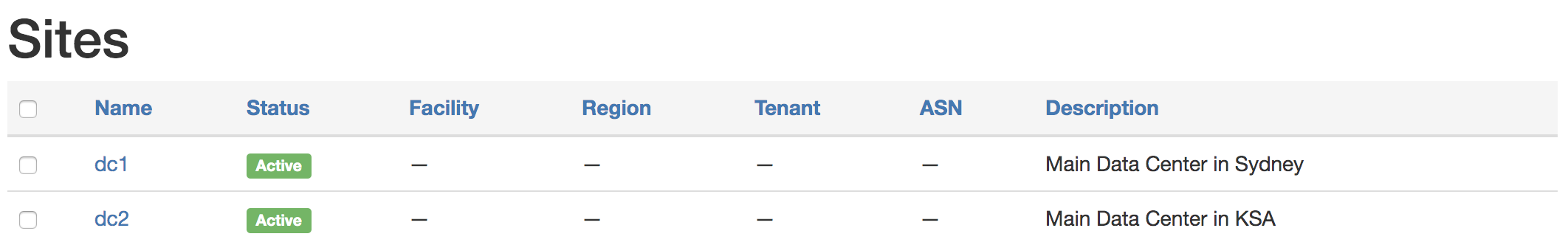
- import\_role:

name: build\_netbox\_db

## **How it works..**

In this recipe we start by populating the sites in our sample network, we define a new data structure for the sites in all.yml file under the group\_vars. We create an ansible role in order to build the netbox database and the first task within this role we use the netbox\_site module to create all sites within our network. We loop across all the sites defined in the sites data structure and push the data to netbox using the netbox\_site module.

We create a new playbook which will be our master playbook to populate all our network inventory into netbox and we reference the role that we have created in order to start executing all the tasks within this role. Once we run this playbook the sites are populated into netbox as shown below



## **See Also ...**

For more information regarding netbox\_site module please check the below URL.

<https://docs.ansible.com/ansible/latest/modules/netbox_site_module.html>

# **Populating Devices in NetBox**

In this recipe we outline how to create an populate network devices in NetBox, this will include declaring the device model and manufacturer along with their role within our network. This will help us build an accurate inventory of our network infrastructure which we can use later in the chapter in the last recipe to build a dynamic inventory for Ansible from NetBox.

## **Getting Ready**

Ansible and Netbox integration should be in place and sites should be defined and populated in NetBox as outlined in the previous recipe. This is critical since when we start to populate devices in NetBox we need to tie them to an existing site.

## **How to do it..**

1. Update the ***group\_vars/all.yml*** file with the devices information as shown below

$ cat group\_vars/all.yml  
  
**< --- Output Omitted for brevity --- >**

devices:

- role: Leaf\_Switch

type: 7020SR

vendor: Arista

color: 'f44336' # red

- role: Spine\_Switch

type: 7050CX3

ru: 2

vendor: Arista

color: '2196f3' # blue

1. Create the ***group\_vars/leaf.yml*** and ***group\_vars/spine.yml*** files and update them with the below information

$ cat group\_vars/leaf.yml

---

device\_model: 7020SR

device\_role: Leaf\_Switch

vendor: Arista

$ cat group\_vars/spine.yml

---

device\_model: 7050CX3

device\_role: Spine\_Switch

vendor: Arista

1. Create a new task to create the manufacturer for all the devices in our inventory under the ***tasks/create\_device\_vendors.yml*** file as shown below

$ cat roles/build\_netbox\_db/tasks/create\_device\_vendors.yml

- name: NetBox Device // Get Existing Vendors

uri:

url: "{{ netbox\_url }}/api/dcim/manufacturers/?name={{ device }}"

method: GET

headers:

Authorization: "Token {{ netbox\_token }}"

Accept: 'application/json'

return\_content: yes

body\_format: json

status\_code: [200, 201]

register: netbox\_vendors

run\_once: yes

tags: device\_vendors

- name: NetBox Device // Create Device Vendors

uri:

url: "{{ netbox\_url }}/api/dcim/manufacturers/"

method: POST

headers:

Authorization: "Token {{ netbox\_token }}"

Accept: 'application/json'

return\_content: yes

body\_format: json

body:

name: "{{ device }}"

slug: "{{ device | lower }}"

status\_code: [200, 201]

when:

- netbox\_vendors.json.count == 0

- netbox\_state == 'present'

run\_once: yes

tags: device\_vendors

1. Update the ***tasks/main.yml*** file to include the create\_device\_vendors.yml as shown below

$ cat roles/build\_netbox\_db/tasks/main.yml

**< --- Output Omitted for brevity --- >**

- name: Create NetBox Device Vendors

include\_tasks: create\_device\_vendors.yml

loop: "{{ devices | map(attribute='vendor') | list | unique}}"

loop\_control:

loop\_var: device

run\_once: yes

tags: device\_vendors

1. Create a new task to create all the device models for all our network devices in our inventory under the ***tasks/create\_device\_types.yml*** file as shown below

$ cat roles/build\_netbox\_db/tasks/create\_device\_types.yml

- name: NetBox Device // Get Existing Device Types

uri:

url: "{{ netbox\_url }}/api/dcim/device-types/?model={{ device.type }}"

method: GET

headers:

Authorization: "Token {{ netbox\_token }}"

Accept: 'application/json'

return\_content: yes

body\_format: json

status\_code: [200, 201]

register: netbox\_device\_types

run\_once: yes

tags: device\_types

- name: NetBox Device // Create New Device Types

uri:

url: "{{ netbox\_url }}/api/dcim/device-types/"

method: POST

headers:

Authorization: "Token {{ netbox\_token }}"

Accept: 'application/json'

return\_content: yes

body\_format: json

body:

model: "{{ device.type }}"

manufacturer: { name: "{{ device.vendor }}"}

slug: "{{ device.type | regex\_replace('-','\_') | lower }}"

u\_height: "{{ device.ru | default(1) }}"

status\_code: [200, 201]

when:

- netbox\_device\_types.json.count == 0

- netbox\_state != 'absent'

register: netbox\_device\_types

run\_once: yes

tags: device\_types

1. Update the ***tasks/main.yml*** file to include the create\_device\_types.yml as shown below

$ cat roles/build\_netbox\_db/tasks/main.yml

**< --- Output Omitted for brevity --- >**

- name: Create NetBox Device Types

include\_tasks: create\_device\_types.yml

loop: "{{ devices }}"

loop\_control:

loop\_var: device

run\_once: yes

tags: device\_types

1. Create a new task to create all the device roles for all our network devices in our inventory under the ***tasks/create\_device\_roles.yml*** file as shown below

$ cat roles/build\_netbox\_db/tasks/create\_device\_roles.yml

- name: NetBox Device // Get Existing Device Roles

uri:

url: "{{ netbox\_url }}/api/dcim/device-roles/?name={{ device.role}}"

method: GET

headers:

Authorization: "Token {{ netbox\_token }}"

Accept: 'application/json'

return\_content: yes

body\_format: json

status\_code: [200, 201]

register: netbox\_device\_role

tags: device\_roles

- name: NetBox Device // Create New Device Roles

uri:

url: "{{ netbox\_url }}/api/dcim/device-roles/"

method: POST

headers:

Authorization: "Token {{ netbox\_token }}"

Accept: 'application/json'

return\_content: yes

body\_format: json

body:

name: "{{ device.role }}"

slug: "{{ device.role | lower }}"

color: "{{ device.color }}"

status\_code: [200, 201]

when:

- netbox\_device\_role.json.count == 0

- netbox\_state != 'absent'

register: netbox\_device\_role

tags: device\_roles

1. Update the ***tasks/main.yml*** file to include the create\_device\_roles.yml as shown below

$ cat roles/build\_netbox\_db/tasks/main.yml

**< --- Output Omitted for brevity --- >**

- name: Create NetBox Device Roles

include\_tasks: create\_device\_roles.yml

loop: "{{ devices }}"

loop\_control:

loop\_var: device

run\_once: yes

tags: device\_roles

1. Create a new task to populate all the device in our inventory under the ***tasks/create\_device.yml*** file as shown below

---

- name: Provision NetBox Devices

netbox\_device:

data:

name: "{{ inventory\_hostname }}"

device\_role: "{{ device\_role }}"

device\_type: "{{ device\_model }}"

status: Active

site: "{{ inventory\_hostname.split('-')[0] }}"

netbox\_token: "{{ netbox\_token }}"

netbox\_url: "{{ netbox\_url }}"

state: "{{ netbox\_state }}"

register: netbox\_device

tags: netbox\_devices

1. Update the ***tasks/main.yml*** file to include the create\_device\_roles.yml as shown below

$ cat roles/build\_netbox\_db/tasks/main.yml

**< --- Output Omitted for brevity --- >**

- name: Create NetBox Device

include\_tasks: create\_device.yml

tags: netbox\_devices

## **How it works..**

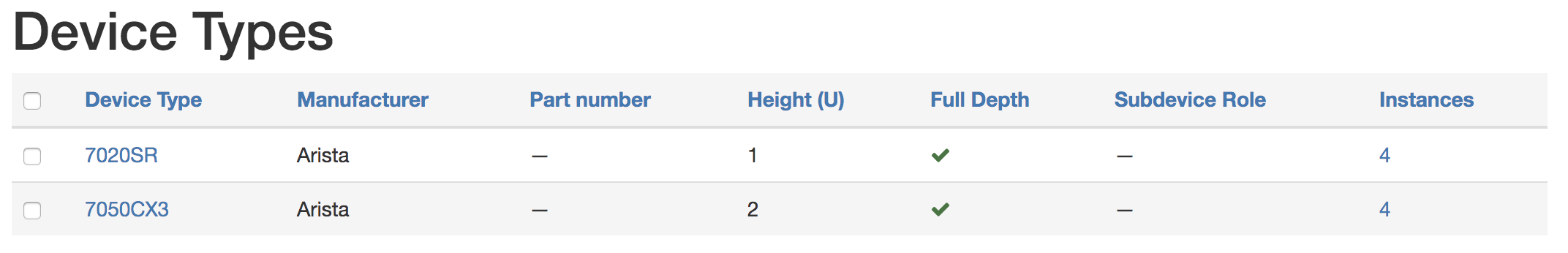
In Order to populate our network devices into netbox we need to populate the below parameters related to the devices into netbox first before adding the devices

* All the manufacturers for all our network devices.
* All the device models for our network equipment.
* All the device roles that will be assigned to each network device.

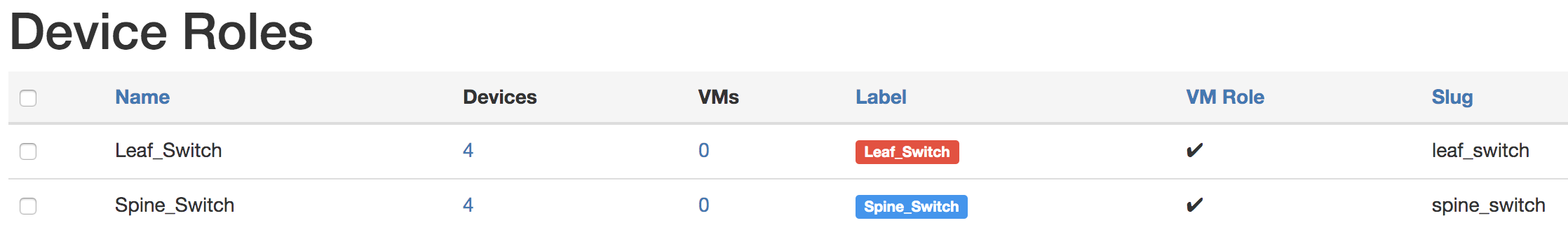
In order to populate all these information there is no pre-built module in ansible to accomplish these tasks and build these objects into netbox. So in order to populate this information into netbox we use the **URI** module which allow us to trigger REST API calls to the correct API endpoint responsible for each of these objects. In all these tasks we use the same approach as outlined below

* We first query the API endpoint using the GET method to get any matching object in the netbox database.
* If the element is not present we create it via a POST REST call to create this object and we supply the needed data to create this object.
* If the element is already present we skip the create part.

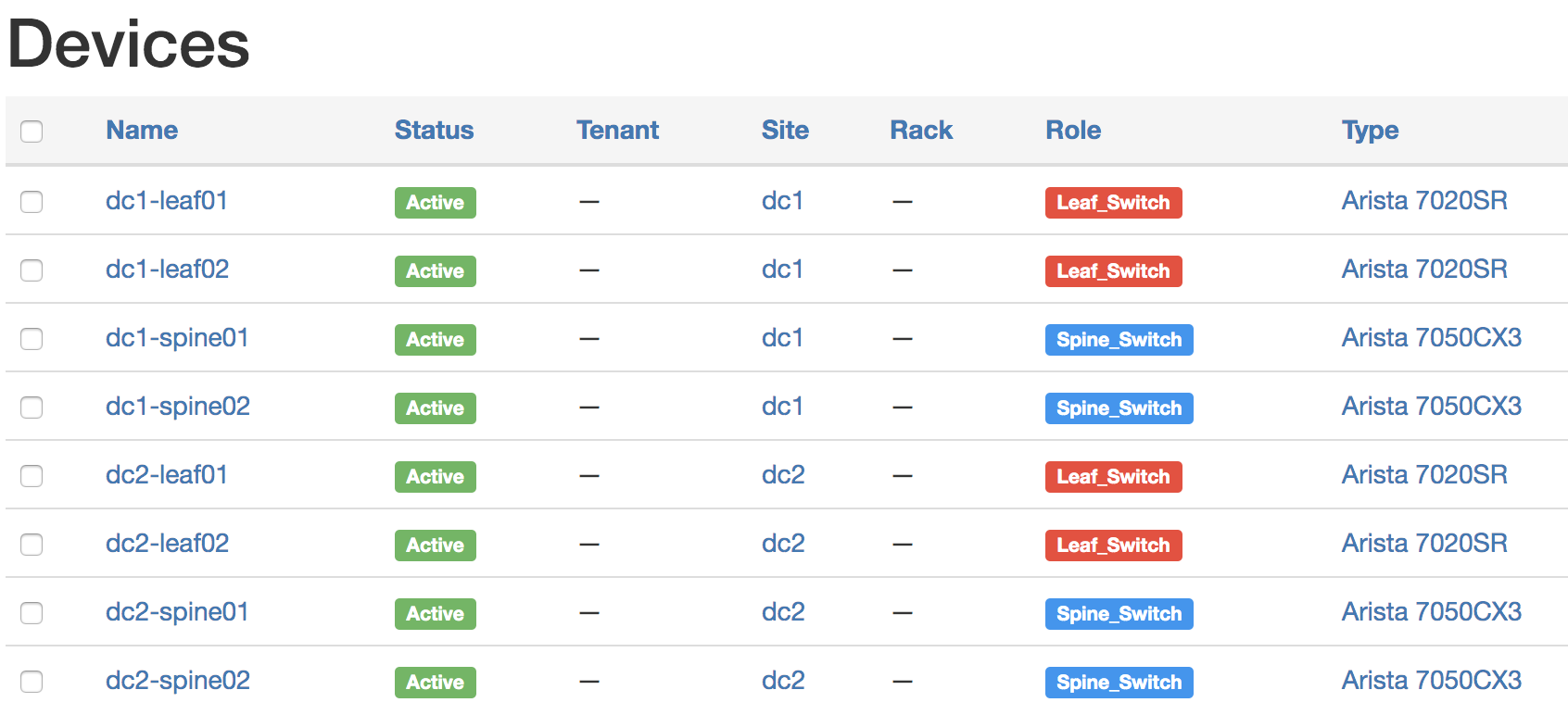
Once we run our playbook we can see that all the device types have been populated into netbox



Also all the device roles for our equipment is populated as shown below



Once we build all the required objects needed to define a device in NetBox (like device role and device types) we use the ansible built-in module ***netbox\_device*** to create all the devices in our ansible inventory. The below snippet outlines all the devices have been correctly populated into NetBox Database.



In this Recipe we used the URI module to trigger API calls to NetBox API in order to create objects within its Database. In order to understand more what are the available APIs and which parameters needs to be passed in each API call we need to check the API documentation for NetBox. The documentation for the API is contained within the NetBox installation and can be accessed in the below URL.

[http://<netbox\_server\_ip>:8080/api/docs/](http://172.20.100.32:8080/api/docs/)

## **See Also ...**

* For More information regarding the NetBox API please consult the below URL

<https://netbox.readthedocs.io/en/stable/api/overview/>

* For More information regarding the ansible module to create devices on netbox please consult the below URL <https://docs.ansible.com/ansible/latest/modules/netbox_device_module.html>

# **Populating Interfaces in NetBox**

In this recipe we will outline how to populate interfaces on network devices in netbox. This provides us with a complete inventory for our devices and will allow us to assign IP addresses to each network device as well as to model the network links within our network.

## **Getting Ready**

In order to create the network interfaces, the devices need to be already created as outlined in the previous recipe.

## **How to do it..**

1. Update the ***group\_vars/all.yml*** with the point to point links within our network fabric in each data center as shown below

p2p\_ip:

dc1-leaf01:

- {port: Ethernet8, ip: 172.10.1.1/31 , peer: dc1-spine01, pport: Ethernet1, peer\_ip: 172.10.1.0/31}

- {port: Ethernet9, ip: 172.10.1.5/31 , peer: dc1-spine02, pport: Ethernet1, peer\_ip: 172.10.1.4/31}

**< --- Output Omitted for brevity --- >**

dc2-leaf01:

- {port: Ethernet8, ip: 172.11.1.1/31 , peer: dc2-spine01, pport: Ethernet1, peer\_ip: 172.11.1.0/31}

- {port: Ethernet9, ip: 172.11.1.5/31 , peer: dc2-spine02, pport: Ethernet1, peer\_ip: 172.11.1.4/31}

1. Create a new task to create all the interfaces for all our network devices in our inventory under the ***tasks/create\_device\_intf.yml*** file as shown below

$ cat roles/build\_netbox\_db/tasks/create\_device\_intf.yml

---

- name: Create Fabric Interfaces on Devices

netbox\_interface:

netbox\_token: "{{ netbox\_token }}"

netbox\_url: "{{ netbox\_url }}"

data:

device: "{{ inventory\_hostname }}"

name: "{{ item.port }}"

description: "{{ item.type | default('CORE') }} | {{ item.peer }}| {{ item.pport }}"

enabled: true

form\_factor: 1000Base-t (1GE)

mgmt\_only: false

mode: Access

state: "{{ netbox\_state }}"

loop: "{{ p2p\_ip[inventory\_hostname] }}"

when: p2p\_ip is defined

tags: netbox\_intfs

1. Update the ***tasks/main.yml*** file to include the create\_device\_intfs.yml as shown below

$ cat roles/build\_netbox\_db/tasks/main.yml

**< --- Output Omitted for brevity --- >**

- name: Create NetBox Device Interfaces

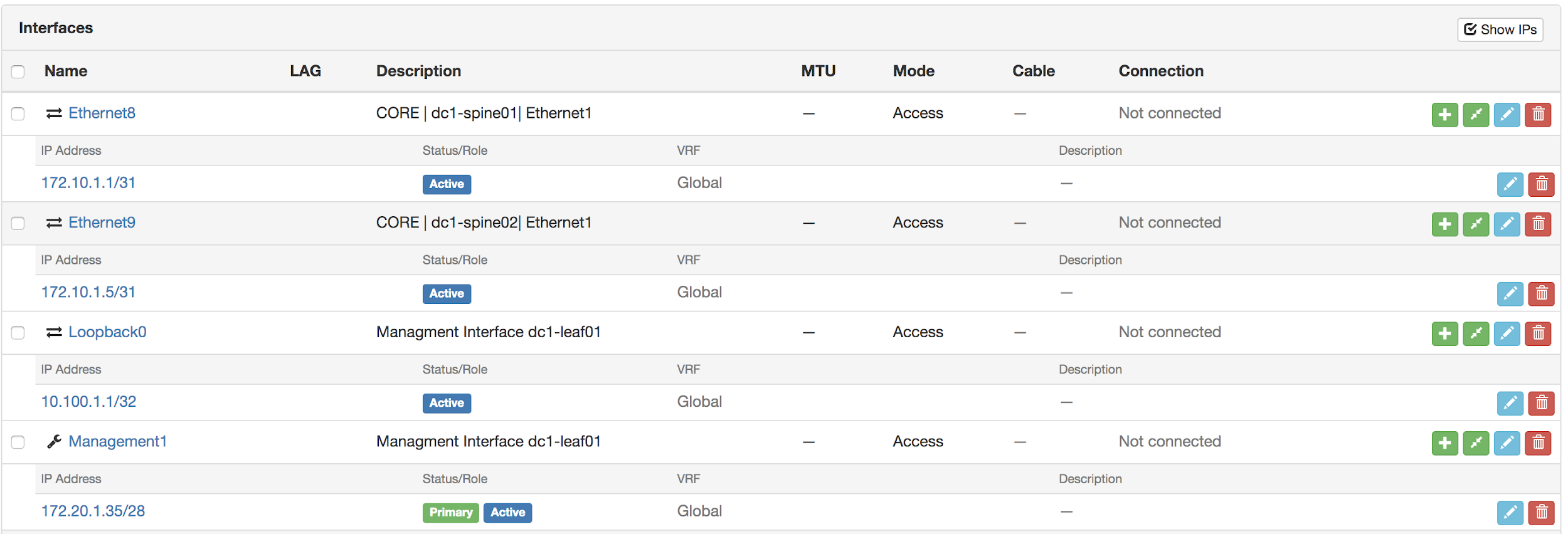
include\_tasks: create\_device\_intf.yml

tags: netbox\_intfs

## **How it works..**

In order to populate all the point to point interfaces in our each data center fabric, we first create the p2p\_ip data structure which holds all the parameters needed to model these point to point links. We then use the netbox\_interface module to create all these links in netbox. Using the same module and following the exact same procedures we can model the management (Out of band management) and loopback interface on our network devices.

The below snippet outline the devices one one of our devices in netbox and how the interfaces are populated



## **See Also ...**

For More information regarding the ansible module to create interfaces on netbox please consult the below URL

<https://docs.ansible.com/ansible/latest/modules/netbox_interface_module.html#netbox-interface-module>

# 

# **Populating IP Address in NetBox**

In this recipe we will outline create IP addresses in NetBox and how to bind these addresses into the interfaces on each of our network devices.

## **Getting Ready**

The network interfaces on each device within our inventory need to be defined and populated into netbox as outlined in the previous recipe.

## **How to do it..**

1. Create a new task to create all the IP addresses attached to the network interfaces for all our network devices in our inventory under the ***tasks/create\_device\_intf\_ip.yml*** file as shown below

$ cat roles/build\_netbox\_db/tasks/create\_device\_intf.yml

---

- name: Create Fabric IPs

netbox\_ip\_address:

netbox\_token: "{{ netbox\_token }}"

netbox\_url: "{{ netbox\_url }}"

data:

address: "{{ item.ip }}"

interface:

name: "{{ item.port }}"

device: "{{ inventory\_hostname }}"

state: "{{ netbox\_state }}"

loop: "{{ p2p\_ip[inventory\_hostname] }}"

tags: netbox\_ip

1. Update the ***tasks/main.yml*** file to include the create\_device\_intfs.yml as shown below

$ cat roles/build\_netbox\_db/tasks/main.yml

**< --- Output Omitted for brevity --- >**

- name: Create NetBox Device Interfaces IP Address

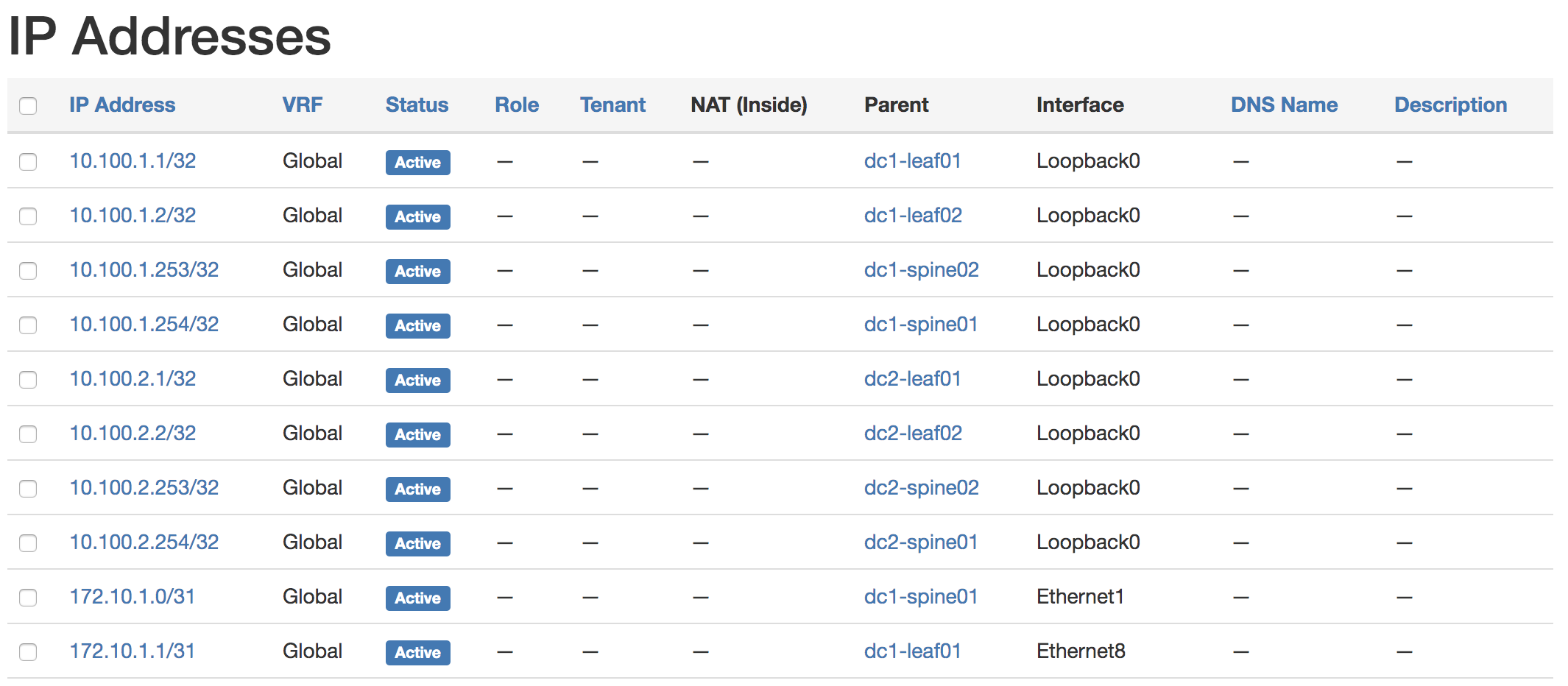
include\_tasks: create\_device\_intf\_ip.yml

tags: netbox\_ip

## **How it works..**

In order to populate all the point to point IP addresses used on each data center fabric, we capture this information in the p2p\_ip data structure which holds all the ip addresses assigned on each interface within our data center fabric. We use the netbox\_ip\_address module to loop across this data structure and populate all the IP addresses assigned to each interface on each device within our data center fabric. The same logic we implement for the management and loopback interfaces.

The below snippet outline some of the IP addresses populated on netbox



## **See Also ...**

For More information regarding the ansible module to create IP addresses on netbox please consult the below URL

<https://docs.ansible.com/ansible/latest/modules/netbox_ip_address_module.html#netbox-ip-address-module>

# **Populating IP Prefixes in NetBox**

In this recipe we will outline create IP Prefixes in Netbox. This allows us to utilize netbox as our IPAM solution to manage IP address assignment within our network.

## **Getting Ready**

No specific requirements is needed in order to populate IP subnets/prefixes into netbox as long as we don’t bound these prefixes to a specific site. In case we bound some subnets into a specific site, then these respective sites need to be defined in netbox prior to this assignment..

## **How to do it..**

1. Update the ***group\_vars/all.yml*** file with the IP prefix information as shown below

$ cat group\_vars/all.yml  
  
**< --- Output Omitted for brevity --- >**

subnets:

- prefix: 172.10.1.0/24

role: p2p\_subnet

site: dc1

- prefix: 172.11.1.0/24

role: p2p\_subnet

site: dc2

- prefix: 10.100.1.0/24

role: loopback\_subnet

site: dc1

- prefix: 10.100.2.0/24

role: loopback\_subnet

site: dc2

- prefix: 172.20.1.0/24

role: oob\_mgmt\_subnet

site: dc1

- prefix: 172.20.2.0/24

role: oob\_mgmt\_subnet

site: dc2

1. Update the ***tasks/main.yml*** file in our role definition to include the below task as shown below

$ cat roles/build\_netbox\_db/tasks/main.yml

**< --- Output Omitted for brevity --- >**

- name: Create IP Prefixes

netbox\_prefix:

netbox\_token: "{{ netbox\_token }}"

netbox\_url: "{{ netbox\_url }}"

data:

prefix: "{{ item.prefix }}"

site: "{{ item.site | default(omit) }}"

status: Active

state: "{{ netbox\_state }}"

loop: "{{ subnets }}"

loop\_control:

label: "{{ item.prefix }}"

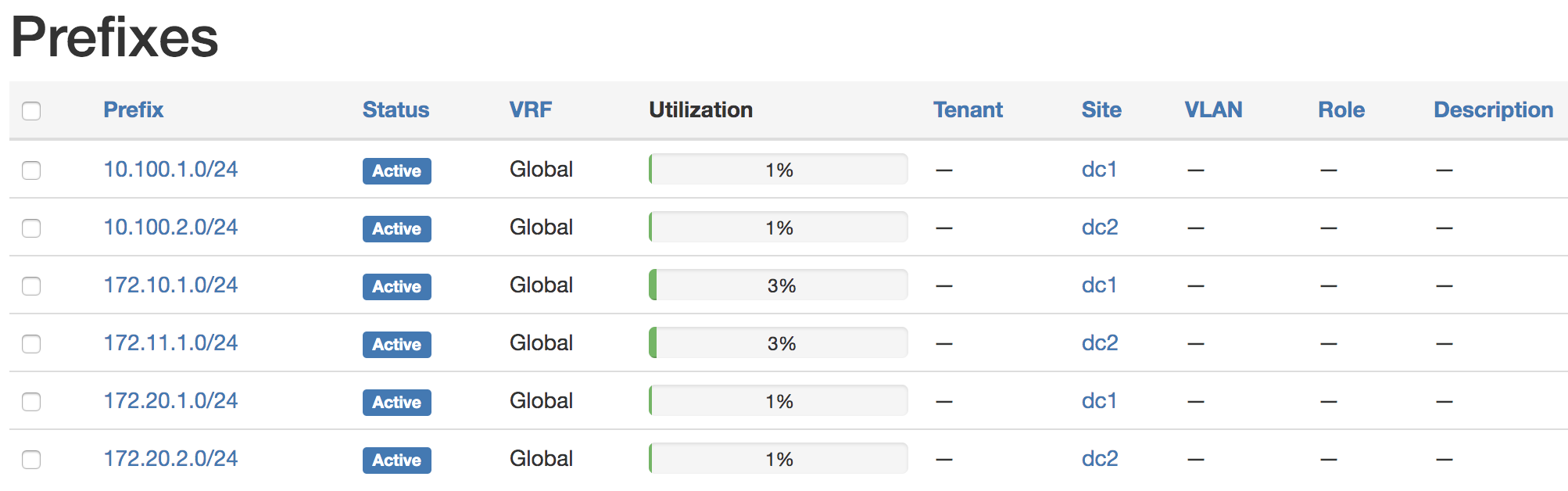
run\_once: yes

tags: netbox\_prefix

## **How it works..**

We define our subnets in the group\_vars/all.yml file under the subnets data structure, we then use the netbox\_prefix module to loop over this data structure and populate the prefixes in netbox.

The below snippet outline the populated prefixes within netbox and their respective utilization



## **See Also ...**

For More information regarding the ansible module to create IP prefixes on netbox please consult the below URL

<https://docs.ansible.com/ansible/latest/modules/netbox_prefix_module.html#netbox-prefix-module>

# **Using Netbox as dynamic Inventory source for Ansible**

In this recipe we will outline how to use netbox as a dynamic inventory source for netbox. With this approach netbox will have the inventory for our network infrastructure and we will use the different grouping available in netbox (like sites, device-roles, etc…) to build dynamic inventory for ansible and to group them according to netbox.

## **Getting Ready**

Integration between Netbox and Ansible need to be in place as outlined in the previous recipes.

## **How to do it..**

1. In our main directory create a new folder called netbox\_dynamic\_inventory
2. In this new directory create a new yaml file called ***netbox\_inventory\_source.yml*** with the below content

$ cat netbox\_dynamic\_inventory/netbox\_inventory\_source.yml

---

plugin: netbox

api\_endpoint: http://172.20.100.32:8080

token: e3afd6708f2a6cb45413c31c713ce6a146d69075

group\_by:

- device\_roles

- sites

1. Create a new playbook called ***pb\_generate\_report.yml*** with the below content

$ cat netbox\_dynamic\_inventory/pb\_create\_report.yml

---

- name: Create Report from Netbox Data

hosts: all

gather\_facts: no

connection: local

tasks:

- name: Build Report

blockinfile:

block: |

netbox\_data:

{% for node in play\_hosts %}

- { node: {{ node }} , type: {{ hostvars[node].device\_types[0] }} , mgmt\_ip: {{ hostvars[node].ansible\_host }} }

{% endfor %}

path: ./netbox\_report.yaml

create: yes

delegate\_to: localhost

run\_once: yes

## **How it works..**

In all the examples and recipes that we have outlined in this book so far we have been using a static inventory file ( in most cases called hosts) where we define our inventory in this file and ansible will parse this inventory file before executing our playbooks. In this reseipce we will use a differnet inventory source which is called a dynamic inventory . In this situation we don’t have a static file that holds our inventory, but we build our inventory dynamically when we run our playbooks during execution time. All our inventory in this example is maintained in netbox and we will use netbox as out inventory source.

For ansible to use a dynamic inventory source a plugin must be in place to talk to this inventory source in order to retrieve our inventory and any variables associated with it. Ansible as of version 2.8 introduced netbox as a plugin that can be used as inventory source and in order to use this plugin we need to define a YAML file which outline the different parameters needed by ansible to communicate with netbox. The mandatory parameters are

* **Plugin**, name in our case it is netbox
* **Api\_endpoint**, which is the API endpoint for our netbox server
* **Token**, which is the authentication token that we have created to establish communication between ansible and our netbox server.

In the YAML declaration file we can specify how we will group our inventory which is coming from netbox. We use the group\_by attribute to outline the paraterms which we will use to group our infrastructure and in our case we are using the device\_roles and sites to group our infrastrcutre.

## There's more...

We can test our dynamic inventory by executing the below command to see how ansible will generate the inventory

$ ansible-inventory --list -i netbox\_inventory\_source.yml

Below is a snippet of the output for the above command that outline the host variables that was retrieved from netbox for single device

{

"\_meta": {

"hostvars": {

"dc1-leaf01": {

"ansible\_host": "172.20.1.35",

"device\_roles": [

"Leaf\_Switch"

],

"device\_types": [

"7020SR"

],

"manufacturers": [

"Arista"

],

"primary\_ip4": "172.20.1.35",

"sites": [

"dc1"

]

},

The below snippet outline the groups that ansible built based on the grouping coming from netbox

"all": {

"children": [

"device\_roles\_Leaf\_Switch",

"device\_roles\_Spine\_Switch",

"sites\_dc1",

"sites\_dc2",

"ungrouped"

]

},

"device\_roles\_Leaf\_Switch": {

"hosts": [

"dc1-leaf01",

"dc1-leaf02",

"dc2-leaf01",

"dc2-leaf02"

]

},

We create a new playbook to test the integration between ansible and netbox and to make sure we can use the data retrieved from netbox as a dynamic inventory source. We care a simple report for each device in the netbox dynamic inventory along with some of the parameters coming from netbox.

Running the playbook we get the below report

$ ansible-playbook pb\_create\_report.yml -i netbox\_inventory\_source.yml

$ cat netbox\_report.yml

# BEGIN ANSIBLE MANAGED BLOCK

netbox\_data:

- { node: dc1-leaf01 , type: 7020SR , mgmt\_ip: 172.20.1.35 }

- { node: dc1-leaf02 , type: 7020SR , mgmt\_ip: 172.20.1.36 }

- { node: dc2-leaf01 , type: 7020SR , mgmt\_ip: 172.20.2.35 }

- { node: dc2-leaf02 , type: 7020SR , mgmt\_ip: 172.20.2.36 }

- { node: dc1-spine01 , type: 7050CX3 , mgmt\_ip: 172.20.1.41 }

- { node: dc1-spine02 , type: 7050CX3 , mgmt\_ip: 172.20.1.42 }

- { node: dc2-spine01 , type: 7050CX3 , mgmt\_ip: 172.20.2.41 }

- { node: dc2-spine02 , type: 7050CX3 , mgmt\_ip: 172.20.2.42 }

# END ANSIBLE MANAGED BLOCK

## **See Also ...**

For More information regarding the netbox plugin please consult the below URL

<https://docs.ansible.com/ansible/latest/plugins/inventory/netbox.html>

To understand more about Ansible dynamic inventory please consult the below URL

<https://docs.ansible.com/ansible/latest/user_guide/intro_dynamic_inventory.html>

# **Building Configuration using NetBox**

In this recipe we will outline how to generate configuration and push configuration to network devices using the data retrieved from netbox.

## **Getting Ready**

In this recipe we will continue to use netbox as our dynamic inventory source so all the configuration outlined in the previous recipe need to be implemented.

## **How to do it..**

1. Under the netbox\_dynamic\_inventory directory create the netbox\_data.yml file with the below content to read the netbox\_data.yml file

$ cat pb\_build\_config.yml

---

- name: Create Report from Netbox Data

hosts: all

gather\_facts: no

connection: local

tasks:

- name: Read netbox Data

include\_vars: netbox\_data.yml

run\_once: yes

1. Update the pb\_build\_config.yml to include a task to query netbox for all interfaces in the datbase

- name: Get Data from Netbox

uri:

url: "{{ netbox\_url }}/api/dcim/interfaces/?device={{.inventory\_hostname }}"

method: GET

headers:

Authorization: "Token {{ netbox\_token }}"

Accept: 'application/json'

return\_content: yes

body\_format: json

status\_code: [200, 201]

register: netbox\_interfaces

delegate\_to: localhost

run\_once: yes

1. Update the playbook with the below task to push configuration to the devices

- name: Push Config

eos\_config:

lines:

- description {{ port.description }}

parent: interface {{ port.name }}

loop: "{{ netbox\_interfaces.json.results }}"

loop\_control:

loop\_var: port

vars:

ansible\_connection: network\_cli

ansible\_network\_os: eos

## **How it works..**

In this recipe we use the NetBox as our SOT and we retrieve the interfaces on a given device using a GET API call to the interfaces endpoints on NetBox and filter this API call by specifying only the interfaces for this specific device. The API call to achieve this is ***api/dcim/interfaces/?device=<deivce-name>/.***

The below snippet outlines the response we get from NetBox

ok: [dc1-spine01] => {

"netbox\_interfaces": {

"api\_version": "2.6",

"changed": false,

"connection": "close",

"json": {

"results": [

{

"description": "CORE | dc1-leaf01| Ethernet8",

"device": {

"display\_name": "dc1-spine01",

"id": 44,

"name": "dc1-spine01",

"url": "http://172.20.100.35:8080/api/dcim/devices/44/"

},

"enabled": true,

**<-- Output Omitted for Brevity -->**

"name": "Ethernet1",

**<-- Output Omitted for Brevity -->**

},

We use the data retrieved from the API to configure the description on all the ports on the devices as per the Data in NetBox database. In this case we use the ***eos\_config*** to push this data to our Arista EOS boxes.