If we want to configure a router using Ansible and leverage information from SolarWinds or SCCM, we can integrate Ansible with SolarWinds for dynamic inventory and use a playbook to configure the router.

Here is an example:

1. Dynamic Inventory from SolarWinds:

Here I create a script (slw\_inv.py) to fetch router information from SolarWinds:

#!/usr/bin/env python

import requests

import json

api\_key = 'SOLARWINDS\_API\_KEY'

api\_endpoint = 'SOLARWINDS\_API\_ENDPOINT'

headers = {

    'Authorization': 'Bearer ' + api\_key,

    'Content-Type': 'application/json',

}

response = requests.get(api\_endpoint, headers=headers)

devices = response.json()

ansible\_inventory = {"routers": {"hosts": [], "vars": {}}}

for device in devices:

    if device["type"] == "router":

        ansible\_inventory["routers"]["hosts"].append(device["hostname"])

print(json.dumps(ansible\_inventory))

Make this script executable (slw\_inv.py) and configure Ansible to use it as a dynamic inventory.

1. Ansible Playbook to Configure Router:   
   Now I Create a playbook (config\_router.yml) to configure the router using the dynamic inventory:

This playbook assumes a Cisco-like router. Adjust the ios\_config section based on our router’s configuration commands.

1. Run the Playbook:

ansible-playbook configure\_router.yml

**CODE Break Down for slw\_inv.py :**

#!/usr/bin/env python

import requests

import json

api\_key = 'SOLARWINDS\_API\_KEY'

api\_endpoint = 'SOLARWINDS\_API\_ENDPOINT'

The shebang line #!/usr/bin/env python specifies the interpreter that should be used to execute the script. In this case, it’s telling the system to use the Python interpreter.

import requests: Imports the requests library, which is used for making HTTP requests.

import json: Imports the json module, which provides methods for working with JSON data.

import subprocess: Imports the subprocess module, which allows you to spawn new processes, connect to their input/output/error pipes, and obtain their return codes.

headers = {

    'Authorization': 'Bearer ' + api\_key,

    'Content-Type': 'application/json',

}

This block creates a dictionary named headers containing HTTP headers required for the API request. It includes the Authorization header with the API key and specifies that the content type is JSON.

response = requests.get(api\_endpoint, headers=headers)

devices = response.json()

The script then sends a GET request to the SolarWinds API using the provided endpoint (api\_endpoint) and headers (headers).

The response is stored in the response variable, and response.json() parses the JSON content of the response into a Python dictionary, which is stored in the devices variable.

ansible\_inventory = {"routers": {"hosts": [], "vars": {}}}

This line initializes an Ansible dynamic inventory structure. It starts with an empty list of hosts under the "routers" group.

for device in devices:

    if device["type"] == "router":

        ansible\_inventory["routers"]["hosts"].append(device["hostname"])

This loop iterates through the devices obtained from the SolarWinds API response.

If a device is identified as a router (based on its "type" attribute), its hostname is added to the list of hosts under the "routers" group in the Ansible inventory.

print(json.dumps(ansible\_inventory))

**CODE Break Down:**

1. Playbook Header:

---: Indicates the start of a YAML document.   
- name: Configure Router: Defines a playbook with the name "Configure Router".

1. Playbook Configuration:   
     
   hosts: routers: Specifies that this playbook will run on hosts belonging to the "routers" group. The group is typically defined in the Ansible inventory file.   
     
   gather\_facts: false: Disables the gathering of facts about the target hosts. Gathering facts is a feature that collects information about the hosts (e.g., IP addresses, OS details) before executing tasks.   
     
   become: yes: Enables privilege escalation, allowing the playbook to run tasks as a superuser (e.g., using sudo).
2. Tasks:   
   - name: Configure Router Interface: Defines a task with the name "Configure Router Interface". This is a human-readable description of the task.   
     
   ios\_config: Uses the ios\_config Ansible module, which is designed for configuring Cisco IOS devices (routers and switches).   
     
   lines: Specifies a list of configuration lines to be applied. In this case, it configures the router's GigabitEthernet0/0 interface.   
     
   parents: Identifies the context where the lines should be applied. In this example, it specifies that the lines should be under the "interface GigabitEthernet0/0" context.

This playbook is essentially instructing Ansible to connect to routers (from the "routers" group), disable fact gathering, and execute a task to configure the GigabitEthernet0/0 interface on the router using specific configuration lines.

