Exported with AIPRM https://www.aiprm.com by YASH SRIVASTAVA on 1/21/2024, 9:29:04 PM

**User:** I have been assigned a project to automate the organization’s Network Infrastructure where I have to Manage and automate the configuration of on-premises network infrastructure, including routers, switches, and firewalls.

In the organization, we have a solar wind tool to monitor the network of the organization and the SCCM tool I have to use these two tools only to create an idea to automate the infra using these tools with Ansible software.

generate any ideas for me and also give me a working script for those ideas.

Please write in English language.

**ChatGPT:** Certainly! Automating the configuration of on-premises network infrastructure using SolarWinds and SCCM with Ansible can be a powerful solution. Below are a few ideas along with sample Ansible scripts to get you started: Idea 1: Dynamic Inventory from SolarWinds Use Ansible’s dynamic inventory feature to pull information about network devices from SolarWinds. This allows you to keep your Ansible inventory up-to-date with the latest information from SolarWinds. Ansible Script: solarwinds\_inventory.py

#!/usr/bin/env python  
import requests  
import json  
  
# Replace 'YOUR\_SOLARWINDS\_API\_KEY' and 'YOUR\_SOLARWINDS\_API\_ENDPOINT' with your SolarWinds API key and endpoint  
api\_key = 'YOUR\_SOLARWINDS\_API\_KEY'  
api\_endpoint = 'YOUR\_SOLARWINDS\_API\_ENDPOINT'  
  
headers = {  
 'Authorization': 'Bearer ' + api\_key,  
 'Content-Type': 'application/json',  
}  
  
response = requests.get(api\_endpoint, headers=headers)  
devices = response.json()  
  
ansible\_inventory = {"all": {"hosts": [], "vars": {}}}  
  
for device in devices:  
 ansible\_inventory["all"]["hosts"].append(device["hostname"])  
  
print(json.dumps(ansible\_inventory))

Ensure the script is executable (chmod +x solarwinds\_inventory.py) and configure Ansible to use this script as a dynamic inventory. Idea 2: Configuration Management with SCCM Leverage SCCM to store configuration files for network devices and use Ansible to deploy these configurations to the respective devices. Ansible Playbook: deploy\_config.yml

---  
- name: Deploy Network Configurations  
 hosts: all  
 become: yes  
 tasks:  
 - name: Copy configuration files from SCCM  
 win\_copy:  
 src: "\\SCCM\_SERVER\share\network\_configs\{{ inventory\_hostname }}.conf"  
 dest: "C:\Temp\network\_config.conf"  
  
 - name: Apply configuration using template  
 template:  
 src: network\_config.conf.j2  
 dest: "C:\Network\Config\network\_config.conf"  
 notify:  
 - Reload Network Device  
  
 handlers:  
 - name: Reload Network Device  
 win\_shell: "Restart-Service NetworkService"

Ensure you have a Jinja2 template file (network\_config.conf.j2) that includes variables and replaceable values. Idea 3: Automated Monitoring Alerts Integrate Ansible with SolarWinds to automatically respond to certain monitoring alerts. For example, if SolarWinds detects a network outage, use Ansible to attempt an automated recovery. Ansible Playbook: recover\_network.yml

---  
- name: Recover Network after Outage  
 hosts: network\_devices  
 become: yes  
 tasks:  
 - name: Check network connectivity  
 wait\_for:  
 host: "{{ inventory\_hostname }}"  
 port: 22  
 timeout: 300  
  
 - name: Restart network services  
 service:  
 name: network  
 state: restarted

Create a SolarWinds alert that triggers the execution of this playbook when a network outage is detected. These are just starting points, and you’ll need to adapt them based on your specific network devices, configurations, and requirements. Ensure that you have the necessary permissions and security measures in place when integrating these tools with Ansible.

**User:** any idea how to automate the configuration of the router, switch .

Please write in English language.

**ChatGPT:** Certainly! Automating the configuration of routers and switches is a common use case for network automation. Ansible is well-suited for this task. Below is a simplified example to get you started. Note that you may need to adjust the configuration based on the specific make and model of your routers and switches. Idea: Basic Router and Switch Configuration Assuming you have a list of routers and switches in your dynamic inventory from SolarWinds, you can use Ansible to deploy basic configurations like setting up VLANs and configuring interfaces. Ansible Playbook: configure\_network.yml

---  
- name: Configure Routers and Switches  
 hosts: routers\_and\_switches  
 gather\_facts: false  
 become: yes  
 tasks:  
 - name: Configure VLANs on Switches  
 ios\_config:  
 lines:  
 - interface Vlan10  
 - description Management VLAN  
 - ip address 192.168.10.1 255.255.255.0  
 parents: interface range GigabitEthernet1/0/1 - 24  
 when: "'switch' in inventory\_hostname"  
  
 - name: Configure Router Interfaces  
 ios\_config:  
 lines:  
 - interface GigabitEthernet0/0  
 - ip address 192.168.1.1 255.255.255.0  
 parents: interface GigabitEthernet0/0  
 when: "'router' in inventory\_hostname"

In this example:

The playbook uses the ios\_config module, assuming Cisco-like devices. Adjust the module and syntax based on the actual devices you have.

It configures VLAN 10 on switch interfaces GigabitEthernet1/0/1 to 24 with an IP address for the management VLAN.

It configures the router interface GigabitEthernet0/0 with an IP address.

Make sure to customize the playbook according to your specific network topology, device types, and configurations. Additional Considerations:

Device-Specific Modules:

Different network devices might require different Ansible modules. For Cisco devices, you can use ios\_config, but for other vendors, you might need modules like nxos\_config (for Cisco Nexus), arubaoss\_config (for HPE Aruba), etc. Check Ansible documentation for the appropriate modules for your devices.

Credentials and Authentication:

Ensure that you have the necessary credentials to access and configure the network devices. Ansible supports various authentication methods, including username/password, SSH keys, etc.

Error Handling:

Implement error handling and validation in your playbooks. Verify that the commands executed successfully and handle potential failures gracefully.

Testing:

Always test your playbooks in a lab or staging environment before applying them to production devices.

Remember to adapt the playbook based on your network architecture, security policies, and specific device requirements. Additionally, consider version control for your playbooks to track changes and updates effectively.

**User:** didn’t understand explain in simple way

Please write in English language.

**ChatGPT:** Certainly! Let’s break it down: Goal: We want to use Ansible to automatically set up some basic configurations on routers and switches in our network. Steps:

Create a Playbook:

Think of a playbook as a set of instructions for Ansible. In our case, we want to configure routers and switches.

The playbook will tell Ansible what to do on each device.

Understand the Playbook:

The playbook is like a recipe. It says what to do on each device.

It configures VLAN 10 on switch interfaces (GigabitEthernet1/0/1 to 24) and sets an IP address.

It configures a router interface (GigabitEthernet0/0) with an IP address.

Customize for Your Network:

Replace ‘routers\_and\_switches’ with the actual group name from your SolarWinds dynamic inventory.

Adjust device-specific details according to your network setup.

Considerations:

Different devices might need different commands. Check the documentation for your device and adjust the playbook accordingly.

Ensure you have the right credentials (username/password) to access and configure the devices.

Always test in a safe environment before applying changes to your live network.

Run the Playbook:

Save the playbook in a file (e.g., configure\_network.yml).

Run it using the ansible-playbook command in your terminal.

This playbook is just a starting point. You’ll need to adapt it based on your network equipment and requirements. If you encounter any issues, feel free to ask for help!

**User:** only for router

Please write in English language.

**ChatGPT:** Certainly! If you want to create a simple Ansible playbook to configure just a router, you can modify the playbook accordingly. Here’s a playbook that configures the interface of a Cisco-like router:

---  
- name: Configure Router Interface  
 hosts: routers  
 gather\_facts: false  
 become: yes  
 tasks:  
 - name: Configure Router Interface  
 ios\_config:  
 lines:  
 - interface GigabitEthernet0/0  
 - ip address 192.168.1.1 255.255.255.0  
 parents: interface GigabitEthernet0/0

Explanation:

This playbook is named “Configure Router Interface.”

It targets devices in the “routers” group (you should replace it with the actual group name from your inventory).

It uses the ios\_config module to send configuration commands to the router.

In this example, it configures the GigabitEthernet0/0 interface with an IP address (192.168.1.1) and subnet mask (255.255.255.0).

Make sure to adjust the playbook based on the specific interface names, IP addresses, and other details of your router. Save the playbook in a file (e.g., configure\_router.yml) and run it using the ansible-playbook command in your terminal.

**User:** i want to configure the router from Ansible using SolarWinds or SCCM

**ChatGPT:** If you want to configure a router using Ansible and leverage information from SolarWinds or SCCM, you can integrate Ansible with SolarWinds for dynamic inventory and use a playbook to configure the router. Here’s an example: 1. Dynamic Inventory from SolarWinds: Create a script (solarwinds\_inventory.py) to fetch router information from SolarWinds:

#!/usr/bin/env python  
import requests  
import json  
  
# Replace 'YOUR\_SOLARWINDS\_API\_KEY' and 'YOUR\_SOLARWINDS\_API\_ENDPOINT' with your SolarWinds API key and endpoint  
api\_key = 'YOUR\_SOLARWINDS\_API\_KEY'  
api\_endpoint = 'YOUR\_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 (chmod +x solarwinds\_inventory.py) and configure Ansible to use it as a dynamic inventory. 2. Ansible Playbook to Configure Router: Create a playbook (configure\_router.yml) to configure the router using the dynamic inventory:

---  
- name: Configure Router  
 hosts: routers  
 gather\_facts: false  
 become: yes  
 tasks:  
 - name: Configure Router Interface  
 ios\_config:  
 lines:  
 - interface GigabitEthernet0/0  
 - ip address 192.168.1.1 255.255.255.0  
 parents: interface GigabitEthernet0/0

This playbook assumes a Cisco-like router. Adjust the ios\_config section based on your router’s configuration commands. 3. Run the Playbook: Run the playbook in your terminal:

ansible-playbook configure\_router.yml

Important Notes:

Replace placeholders like ‘YOUR\_SOLARWINDS\_API\_KEY’ and ‘YOUR\_SOLARWINDS\_API\_ENDPOINT’ with your actual SolarWinds API key and endpoint.

Ensure that your router supports the configuration commands used in the playbook.

Adapt the playbook to match your router’s specific configuration requirements.

This setup allows you to dynamically fetch router information from SolarWinds and use Ansible to configure the routers based on that information.

**User:** can we merge this in one

**ChatGPT:** Certainly! You can combine the dynamic inventory script and the Ansible playbook into a single file. Below is an example of how you can merge them into one script:

#!/usr/bin/env python  
import requests  
import json  
import subprocess  
  
# Replace 'YOUR\_SOLARWINDS\_API\_KEY' and 'YOUR\_SOLARWINDS\_API\_ENDPOINT' with your SolarWinds API key and endpoint  
api\_key = 'YOUR\_SOLARWINDS\_API\_KEY'  
api\_endpoint = 'YOUR\_SOLARWINDS\_API\_ENDPOINT'  
  
# SolarWinds API request  
headers = {  
 'Authorization': 'Bearer ' + api\_key,  
 'Content-Type': 'application/json',  
}  
  
response = requests.get(api\_endpoint, headers=headers)  
devices = response.json()  
  
# Ansible dynamic inventory structure  
ansible\_inventory = {"routers": {"hosts": [], "vars": {}}}  
  
# Collect router hostnames from SolarWinds  
for device in devices:  
 if device["type"] == "router":  
 ansible\_inventory["routers"]["hosts"].append(device["hostname"])  
  
# Write the Ansible dynamic inventory to a JSON file  
with open("solarwinds\_inventory.json", "w") as json\_file:  
 json.dump(ansible\_inventory, json\_file)  
  
# Run Ansible playbook to configure the router  
subprocess.run(["ansible-playbook", "-i", "solarwinds\_inventory.json", "-e", f"api\_key={api\_key}", "configure\_router.yml"])

Explanation:

The script fetches router information from SolarWinds and creates an Ansible dynamic inventory structure.

It writes this inventory to a JSON file (solarwinds\_inventory.json).

Finally, it runs the Ansible playbook (configure\_router.yml) using the generated dynamic inventory.

Make sure to replace ‘YOUR\_SOLARWINDS\_API\_KEY’ and ‘YOUR\_SOLARWINDS\_API\_ENDPOINT’ with your actual SolarWinds API key and endpoint. You can then run this script to fetch router information from SolarWinds and configure routers using Ansible in a single step.