Network Automation for Routing and Switching: Leveraging Tools Like Ansible and Netconf

Nikhil Bhagat
Principal Network Engineer
Independent Scholar, Network Engineering
nikhil.bhagat90@gmail.com

***Abstract*—** With growing network size and complexity, service providers and enterprises face ever-increasing difficulty to keep the networks in their desired configurations with minimum latency and high availability. Manual configuration and implementation processes are error-prone, inefficient, and costly. Network automation using tools like Ansible or Netconf can definitely aid in reducing implementation errors, reduce engineering time and eventually reduce operational costs. Automation can help enterprises and service providers increase effectiveness, and optimize. This paper describes current difficulties with manual network configurations, talks about automation as a tool for reducing outages, and introduces Ansible and Netconf. This paper also provides the setup and configuration of both tools, configuration examples, and important design considerations when applying network automation tools.

Keywords— Manual Network Configuration, Ansible, Netconf, Network Automation, standardization, increase efficiency.

I. INTRODUCTION

As networks scaled up to accommodate an increasing array of network devices, applications, and services, the complexity of their deployment has only increased. It is not just the routers and switches, but policies, security setups and protocols that control the behavior of the network that service providers and enterprises are juggling in the evolving network environment. In most environments, network management has meant manually configuring the hardware — often via command-line interfaces (CLI) [1]. This worked well when networking was relatively new, but in today's large distributed networks, it's poses a great challenge.

Manual network configuration can become incredibly error-prone if there are multiple network engineers changing things in different places in the network [2]. A single misconfiguration can cause a crash or other security vulnerability which can impact thousands of users or devices. Further, manual processes are cumbersome and slower to adapt to network upgrades or provisioning, which inhibits the organization's scaling capabilities.

Network automation solutions solve these challenges and allows organizations to automate network infrastructure management and setup [3]. Ansible and Netconf automate tasks, allowing network engineers to create configurations and run them on the network in a consistent, reproducible fashion [4]. This reduces human error, and makes network operations consistent and predictable.

This paper dives deep into the current challenges with manual configuration and provides a solution to overcome these challenges with the help of network automation. The paper describes ways to automate routing and switching with the help of Ansible and Netconf.

II. CURRENT CHALLENGES WITH MANUAL NETWORK CONFIGURATION CHANGES IN PRODUCTION ENVIRONMENTS

Manual network settings are still common across most organizations, particularly those where legacy systems and processes prevail. However, this form of network control also brings several challenges that can significantly affect the performance, security, and capacity of the network:

A. Human Error

Human error is perhaps the most important challenge with manual network configuration. Misconfigurations (IP

addresses, routing policies, ACLs, etc.) could cause network disruption, breach, or service outages [3]. In large complex networks with multiple network devices, it is hard to consistently implement configuration changes across the entire infrastructure. It is likely to suffer from configuration drift and inconsistencies.

B. Time-Consuming Processes

Manual configurations involve network engineers accessing the devices and configuring them through CLI [5]. This isn't just slow, it's time-consuming too if the engineers are rolling out new services or handling reproducible configuration [6]. As a result, teams cannot scale their work effectively or adapt to real-time change.

C. No Standardization

In organizations where multiple network engineers monitor and handle changes, configuration processes aren't usually consistent [7]. Engineers might use a different method to configure components, and configurations may differ and overlap. Without templates and workflows standardized, network policies can be difficult to apply uniformly to the entire infrastructure.

D. Difficult to Scale

As networks grow larger in size and become complex, manually maintaining these infrastructures become time-consuming [8]. Adding more devices or segments to a network can require hours or days of manual configuration. This is especially harmful for service providers, who have to offer new services as fast as possible to satisfy customers.

E. Difficult to troubleshoot the network

Troubleshooting and diagnosing problems can be difficult with a manual managed network infrastructure because config implementation may need to be distributed across several devices with no single point of control [9]. Monitoring a change or determining the origin of a configuration error is not a simple task, especially in complex networks where many parts depend on one another.

With these limitations, network automation has become a promising option for enterprises that wish to automate network operations without human intervention.

III. WHY AUTOMATION CAN HELP SERVICE PROVIDERS MINIMIZE SELF-INFLICTED OUTAGES

Service providers suffer from self-inflicted outages, whether it is due to a configuration mistake or a variation introduced by inconsistent configuration [10]. Network automation solves these issues as it offers a way to automate monotonous work, eliminate manual labor, and maintain network consistency. These are the most important ways that automation can reduce outages:

A. Minimize Human Error

Service providers can significantly minimize the human error by automating network configurations. Automating solutions enable engineers to programmatically define configuration templates and policies that can be rolled out to multiple devices, in a standard way [11]. This decreases outage potential caused by misconfigurations.

B. Consistency and Standardization

Network automation tools enforce consistency by using the same template in all devices [12]. This will make sure all the network devices run with same policies and will reduce the possibility of configuration drift or inconsistent configurations. A standard configuration will not only help engineers troubleshoot issues efficiently but also predict network performance.

C. Faster Service Provisioning

The Network automation allows providers to start up new services or provision new devices efficiently and quickly. Rather than manually configuring each device, automation helps engineers administer default configuration to the network with minimum delay and impact on the service [13].

D. Automated Rollbacks and Recovery

Automating tools can automatically rollback to a known stable configured state that is safe in case of a misconfiguration [14]. The rollback helps avoid long disruptions and downtime. This is a capability that's extremely useful in highly distributed networks where it can be time-consuming to determine the root cause of a problem.

E. Improved Network Visibility

Automation solutions give you greater visibility into the network through a centralized management and

monitoring of configurations. It enables engineers to track their upgrades, see configuration history, and monitor how the updates effect network performance. This transparency is also useful in detecting problems early before they cause potential outages.

IV. INTRODUCTION TO ANSIBLE

Ansible is an open-source automation tool, which makes it easy to configure and maintain network equipment, servers and applications [15]. It defines tasks and configurations using a simple, human-readable language (YAML), which makes it easy for network engineers who do not have a lot of programming experience. Since Ansible is agentless, it does not require software to be installed on the target systems. Ansible runs over standard network protocols such as SSH [16].

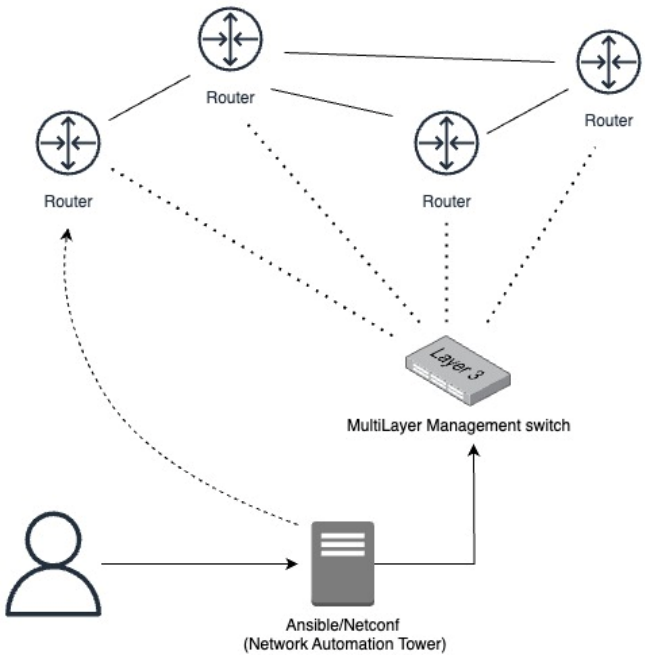

Fig 1. Network Automation with Ansible/Netconf

Ansible is widely used network automation tool to automate repetitive actions, like setting up default configuration on routers and switches, releasing software updates, and administering network policies. It is also very scalable, allowing businesses to easily automate configurations on large networks. Ansible is cross-vendor compatible with the majority of networking vendors, such as Cisco, Juniper and Arista, which makes it ideal for multi-vendor systems [4].

V. HOW TO CONFIGURE AND SETUP ANSIBLE

A. *Install Ansible*

Use package managers to install Ansible on the majority of Linux distributions or macOS. In Ubuntu, for instance, installation of Ansible can be performed with these commands [17]:

```
sudo apt update
sudo apt install ansible
```

For CentOS:

```
sudo yum install epel-release
sudo yum install ansible
```

For macOS, installation can be performed using Homebrew:

```
brew install ansible
```

B. *Set Up Inventory File*

In inventory file, Ansible defines the devices that will be controlled by ansible. The inventory file may be as small as a list of IP addresses or hostnames [18]. Here's an illustration of a simple inventory sheet:

```
[routers]
router1 ansible_host=192.168.1.21
ansible_user=admin  ansible_password=example123
router2 ansible_host=192.168.1.22
ansible_user=admin ansible_password= example123

[switches]
switch1 ansible_host=192.168.2.21
ansible_user=admin ansible_password= example123
switch2 ansible_host=192.168.2.22
ansible_user=admin ansible_password= example123
```

C. *Write a Playbook*

Ansible playbooks describe the actions one wants to execute on the devices in the inventory file [19]. The playbooks are YAML and each playbook has task. Below is a simple playbook for configuring an IP address on a Cisco router:

```
---
- name: Configure Cisco Router
  hosts: routers
  tasks:
    - name: Configure interface IP address
      ios_config:
        lines:
          - interface GigabitEthernet0/1
          - ip address 192.168.3.1 255.255.255.0
```

D. *Execute the Playbook*

When the playbook is created you can run it with the ansible-playbook command:

```
ansible-playbook -i inventory_file playbook.yml
```

Ansible will connect to the devices configured in the inventory file and execute the playbook configurations.

VI. CONFIGURATION OF ANSIBLE

The following is an executable version of Ansible playbook for configuring multiple VLANs on a Cisco switch::

```
---
- name: Configure VLANs on Cisco Switch
  hosts: switches
  tasks:
    - name: Create VLAN 10
      ios_config:
        lines:
          - vlan 10
          - name Dept1_VLAN
    - name: Create VLAN 20
      ios_config:
        lines:
          - vlan 20
          - name Dept2_VLAN
    - name: Assign VLANs to interfaces
      ios_config:
        lines:
          - interface GigabitEthernet0/0/1
          - switchport mode access
          - switchport access vlan 50
          - interface GigabitEthernet0/0/2
          - switchport mode access
          - switchport access vlan 60
```

This playbook will create:

Creates two VLANs:

VLAN 50 called Dept1_VLAN.

VLAN 60 called Dept2_VLAN.

Configures two interfaces on the switch:

GigabitEthernet0/0/1 is configured as an access port for VLAN 50.

GigabitEthernet0/0/2 is configured as an access port for VLAN 60.

This automated operation allows the vlans and interface configurations to be uniform across Cisco switches without having to login to every switch.

VII. INTRODUCTION TO NETCONF

Netconf (Network Configuration Protocol) is an XML-based protocol for configuring network devices [20]. Netconf was introduced to overcome the shortcomings of traditional manual CLI-based administration by offering an automation standard, vendor-independent way to programmatically configure network devices.

Netconf runs on secure transport mechanisms like SSH and uses XML to represent device configurations [21]. It supports access to get device configuration, modify existing device configuration, and apply device-wide consistency across devices. Netconf also facilitates transactions-based operations, so that changes to configuration happen atomically.

VIII. HOW TO CONFIGURE AND SETUP NETCONF

A. *Enable Netconf on the Network Device*

Netconf needs to be installed on the target network device before one can use it. Using Cisco IOS routers, for instance, one can enable Netconf using the following commands [22]:

```
configure terminal
netconf-yang
```

This command enables Netconf and YANG data model in the router so it can be programmatically controlled by Netconf.

B. *Install a Netconf Client*

A Netconf client is required to connect to the Netconf enabled device. Python has an existing Netconf client library called ncclient that can be installed with pip:

```
pip install ncclient
```

C. *Write a Python Script to connect with Netconf*

With ncclient, one can develop Python programs that connect to the Netconf server on the device [23]. For example, below is a Python script that retrieves the Cisco router configuration:

```
from ncclient import manager

with manager.connect(host="192.168.4.1", port=830, username="admin", password=" example123", hostkey_verify=False) as m:
    config = m.get_config(source="running")
    print(config)
```

This script connects to a Cisco router through netconf and extracts the current configuration.

IX. SAMPLE CONFIGURATION OF NETCONF

Below script showcases a Netconf request to configure an IP on an interface of a Cisco router:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <interface>
          <GigabitEthernet>
            <name>2</name>
            <ip>
              <address>
                <primary>
                  <address>192.168.20.1</address>
                  <mask>255.255.255.0</mask>
                </primary>
              </address>
            </ip>
          </GigabitEthernet>
        </interface>
      </native>
    </config>
  </edit-config>
</rpc>
```

This XML payload configures an IP address on GigabitEthernet2 of the router [24].

X. KEY DIFFERENCES BETWEEN ANSIBLE AND NETCONF

Even though both Ansible and Netconf are network automation tools there are some differences between them [25]. Choosing the right tool for automation as per the requirements is essential for success.

TABLE 1. DIFFERENCES BETWEEN ANSIBLE AND NETCONF

Feature	Ansible	Netconf
Automation Type	Configuration management and automation tool	Network configuration protocol
User friendly	User-friendly and intuitive	Needs understanding of YANG models
Configuration Scope	Use to orchestrate complex workflows	Not designed to orchestrate workflows by default
Protocol used for connection	SSH	XML/JSON over SSH
Granularity	Tasks are abstracted providing high-level granularity	Provides Fine-grained, protocol-level control over network devices
Multi-threading	Supports multi-threading	Does not support multi-threading by default

XI. CONSIDERATIONS WHILE USING NETWORK AUTOMATION TOOLS

Even though network automation provides a lot of advantages, there some considerations that organizations should look into before adopting automation tools such as Ansible and Netconf:

A. Compatibility

Make sure that the automation solution you are running is supported by your network devices [26]. Vendors might provide less or more Netconf/Ansible modules, depending on device model and operating system.

B. Change Control

Automating network configuration can be risky, if not managed carefully. Change management and version control must be in place to log configuration changes and ensure that changes are tested before rolling them out to production [3].

C. Maintain Network Security

Network automation tools need to be able to access devices (most likely using SSH or another encrypted protocol). It is very important to have proper authentication and authorization to prevent access to the network devices.

D. Stress Testing and Verification

Organizations must verify the automation scripts thoroughly in a lab before rolling out network configuration [27]. This helps troubleshoot unidentified issues, and will ensure that the finalized scripts will be error-free during deployment.

XII. CONCLUSION

Network automation is transforming the routing and switching landscape for service providers and enterprises. Using tools such as Ansible and Netconf organizations can automate manual network setting, prevent human error, and enable network performance optimization. Ansible, with its basic YAML playbooks, can be used to build powerful, dynamic network automation, while Netconf is a common, vendor-neutral protocol for managing device configurations in the background. All of these tools help organizations construct more reliable, scalable and performant networks that are ready to adapt to the needs of modern networks.

REFERENCES

[1] R. Caruso, "Network management: a tutorial overview," in Proc. of IEEE Global Telecommunications Conference, pp. 1-8, 2010.

[2] D. A. F. B. C. A. C. M., "Computer network operations—making it all go at once," in Proc. of ACM SIGCOMM, Seattle, WA, 2009.

[3] S. Lee, T. M. Wong, and H. Kim, "To automate or not to automate: On the complexity of network configuration," in Proc. of ACM SIGCOMM, pp. 1-8, 2009.

[4] "Network automation with Ansible," Online, Available: https://ansible.com/network-automation, 2017.

[5] W. Zheng, R. Bianchini, and T. D. Nguyen, "Automatic configuration of internet services,"

IEEE/ACM Transactions on Networking, vol. 17, no. 1, pp. 118–130, 2009.

[6] H. Feamster, "The evolution of network configuration: a tale of two campuses," in Proc. of ACM Internet Measurement Conference, pp. 14–20, 2007.

[7] T. Benson, A. Akella, and D. A. Maltz, "Unraveling the complexity of network management," in Proc. of ACM Internet Measurement Conference, pp. 93–104, 2010.

[8] R. Chadha, G. Lapiotis, and S. A. Wright, "Guest editorial - policy-based networking," IEEE Communications Magazine, vol. 45, no. 10, pp. 18-22, Oct. 2007.

[9] U. H. Rao, "Challenges of implementing network management solution," in Proc. of IEEE GLOBECOM, pp. 2007–2011, Dec. 2010.

[10] D. Oppenheimer, A. Ganapathi, and D. A. Patterson, "Why do internet services fail, and what can be done about it?," in Proc. of USENIX Symposium on Internet Technologies and Systems, pp. 1–10, 2003.

[11] "Network automation," Online, Available: https://networkautomation.com, 2017.

[12] S. M. Bellovin and R. Bush, "Configuration management and security," IEEE Security & Privacy, vol. 4, no. 6, pp. 85-88, 2006.

[13] R. Cannistra, et al., "Enabling autonomic provisioning in SDN cloud networks with NFV service chaining," in Proc. of IEEE Conference on Network Function Virtualization and Software Defined Networks, pp. 1–5, Nov. 2015.

[14] Y. Ravi and M. Ravi, "NetRevert: rollback recovery in SDN," in Proc. of IEEE Conference on Network Softwarization (NetSoft), pp. 22–25, June 2017.

[15] "Ansible Documentation," Online, Available: https://docs.ansible.com.

[16] L. Hochstein, Ansible: Up and Running, 2nd ed., O'Reilly Media, 2017.

[17] "Installing Ansible," Online, Available: https://docs.ansible.com/ansible/latest/installation_guide, 2017.

[18] "Ansible Documentation," Online, Available: https://docs.ansible.com, 2017.

[19] "Ansible Network Examples," Online, Available: https://docs.ansible.com/ansible/network, 2017.

[20] "Network Configuration Protocol (NETCONF)," Online, Available: https://tools.ietf.org/html/rfc6241.

[21] J. Huang, B. Zhang, G. Li, X. Gao, and L. Yan, "Challenges to the new network management protocol: NETCONF," IEEE Communications Surveys & Tutorials, vol. 16, no. 3, pp. 1686–1705, 2014.

[22] S. Lee, T. M. Wong, and H. S. Kim, "Improving manageability through reorganization of routing-policy configurations," in Proc. of IEEE INFOCOM, pp. 1-10, 2013.

[23] S. Bhushan, "A Python module for NETCONF clients," in Proc. of ACM SIGCOMM, pp. 11–14, 2017.

[24] Cisco, "Cisco IOS XE native YANG model," GitHub repository, YangModels, 2017. [Online]. Available: https://github.com/YangModels/yang/blob/main/vendor/cisco/xe/1671/Cisco-IOS-XE-native.yang.

[25] "Agentless Architecture," Online, Available: https://docs.ansible.com/ansible/latest/architecture, 2018.

[26] W. Brockelsby and S. Dilda, "Tactical network automation with NetZTP and One Shot," in Proc. of IEEE MILCOM, pp. 1924-1929, Nov. 2018.

[27] A. M. Mazin, R. A. Rahman, M. Kassim, and A. Mahmud, "Performance analysis on network automation interaction with network devices using Python," in Proc. of IEEE International Conference on Telecommunications and Signal Processing (TSP), pp. 1-5, July 2019.