Summer 2023

Leveraging Ansible to Locate Network Devices

Elijah Hadden
ekh42@uakron.edu

Leveraging Ansible to Locate Network Devices

Elijah Hadden

The University of Akron

Computer Information Systems: Networking

April 2023

Table of Contents

Part 1: Project Analysis

Project Overview

This report demonstrates the collection of Layer 2 and Layer 3 network device information using Ansible and explains the limitations identified performing such a procedure. Identifying the location of a single device on a Campus Area Network requires access to, and knowledge of, multiple networking devices. To track down a wired device which has broken company policy, a network administrator must first discover the hardware address associated with a given IP, then access the MAC address table on the next hop switch, identify the switchport number, and finally identify the location of the switchport based on its switchport description. The original intention of this project was to simplify the process of locating a device on an enterprise network by placing all relevant information into a singular comprehensive file with the help of network automation software. Maintaining such a document would be beneficial to network security because it could provide low level technicians access to device location information without requiring access to core routing devices, a privilege typically designated to administrators and upper-level network engineers. Creation of this document would also improve incident response time as fewer devices would need to be accessed before the location of a potentially malicious device is identified and tracked down. Catching rouge devices sooner would reduce downtime in the event of a network outage and would increase the chances of catching the perpetrator of a potential cybercrime.

Modifications to Original Proposal

After the procedure was conducted, it was determined that limitations inherent to Ansible prevent the intended information base from being created without the assistance of additional

database software or programming resources and cannot be created using Ansible alone. The primary roadblock that prevents Ansible from effectively producing the intended output was its inability to use reference numerical values stored in a list. Without the ability to use device MAC addresses to generate nested information about devices, Ansible fails at efficiently storing outputs gathered from network switches in a way that can easily be output to a CSV file. This limitation necessitates a discussion regarding what Ansible can and should be used for as it relates to the gathering of Layer 2 and Layer 3 network device information. Through this discussion, alternative avenues will be suggested for accomplishing the original goal described. These topics are the focal point of this report.

Timesheet Analysis

In general, the project had periods of push and pull which ended up averaging out over time. Major discrepancies occurred in physical lab assembly and the creation of the ansible script. Ansible, like many technical skills, was learned more effectively once hands-on practice began. Less time should have been allocated to reading the textbook and more time should have been allocated to practicing with sample playbooks. Additional time was spent assembling the lab environment because forms needed to be created and signed to properly document the signing-out of networking equipment. Overall, the project took 42% longer than the time estimated. Details regarding time estimations can be observed in Figure 1.3.1.

Phases	Tasks within Phase	Estimated Time (in hours)	Estimated task/ phase completion date	Actual Time (in hours)	Actual task/phase completion date	Percent error
Research	Learn Ansible basics	10	2/14/2023	12	2/12/2023	17%
Setup	Physically assemble lab environment	1	2/16/2023	5	2/19/2023	80%
	Install virtual machine and Ansible dependencies	4	2/19/2023	2	2/23/2023	50%
	Create addressing scheme	1	2/21/2023	3	3/2/2023	67%
	Configure Switches	5	2/28/2023	6	3/5/2023	17%
Application	Create Ansible script	10	3/21/2023	35	3/31/2023	71%
Documentation	Compose final report	10	4/2/2023	13	4/11/2023	23%
Presentation	Create PowerPoint	5	4/9/2023	3	4/11/2023	40%
Total		46		**79**		42%

Figure 1.3.1 – Timesheet

Materials

The following materials were used to complete the procedure:

Quantity	Item description	Unit Cost	Total Cost	Where & How Obtained
2	Cisco 3560G WS-C3560G-48PS-S	$0	$0	Signed out from place of employment with written permission from supervisor.
5	7ft. cat 6 cable	$0	$0	Signed out from place of employment with written permission from supervisor.
1	Asus ZenBook 14 UX432FA	$0	$0	Repurposed personal device
1	Toshiba Satellite C655	$0	$0	Repurposed personal devices
2	Raspberry Pi 3	$0	$0	Repurposed personal device
	TOTAL PROJECT COST		$0	

Figure 1.4.1 – Materials

The following form was created and signed to document the process of signing out networking hardware place of employment:

Employee Equipment Checkout Agreement

The University of Akron Network Services

Employee Name: Elijah Hadden Date: 2/14/2023

Email: ekh42@uakron.edu Phone Number: 440-361-9477

ID Number: 4729761 Reason for checkout: Senior project

Equipment Out: 2 / 15 / 2023 Equipment In: 5 / 1 / 2023

Item	Qty.	Mfr.	Model	S/N	Storage Location
Managed Switch	1	Cisco	3560G WS-C3560G-48PS-S	F0C1427W02M	SHS, 53A
Managed Switch	1	Cisco	3560G WS-C3560G-48PS-S	F0C1225W4MV	SHS, 53A
Cat. 6 Cabling	5				Comp. 127

Terms and Conditions:

By checking out the equipment listed above, the student assumes full personal liability for the cost of replacement / repair in the event that any items are lost, stolen or damaged. Equipment that has been lost, stolen or damaged will be immediately reported to the signing supervisor with the understanding that equipment checkout privileges may be revoked. The student agrees to use only official and legal software with any hardware being checked out and agrees to return devices to factory defaults before and after check-out period.

Employee Signature: Elijah Hadden Supervisor Signature: Yolanda Carter

Figure 1.4.2 – Equipment Checkout Agreement

Part 2: Setup

Network Design

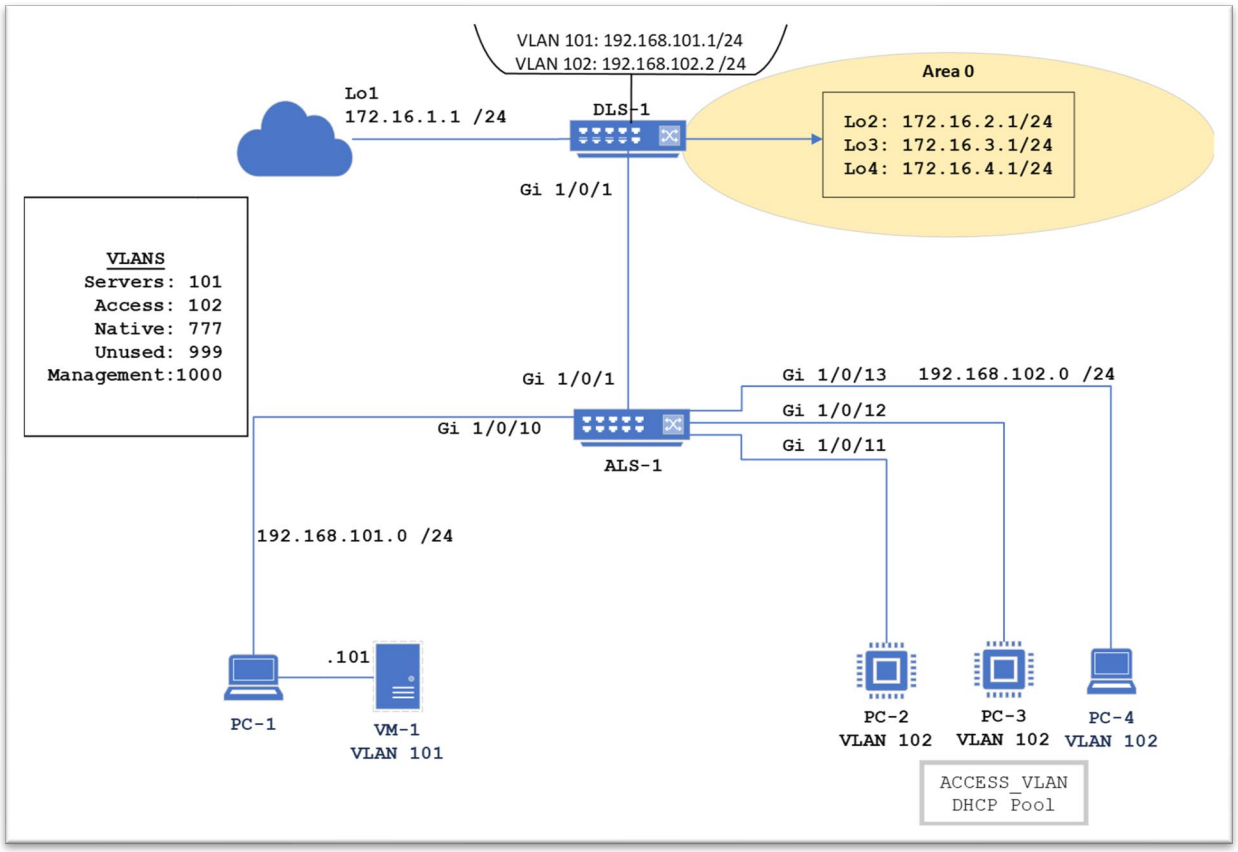


Figure 2.1.1 – Network topology.

Hostname	Interface	IP Address	Subnet Mask
DLS-1	Loopback 1	172.16.1.1	255.255.255.0
	Loopback 2	172.16.2.1	255.255.255.0
	Loopback 3	172.16.3.1	255.255.255.0
	Loopback 4	172.16.4.1	255.255.255.0
	VLAN 101	192.168.101.1	255.255.255.0
	VLAN 102	192.168.102.1	255.255.255.0
	VLAN 1000	192.168.0.1	255.255.255.0
ALS-1	VLAN 1000	192.168.0.100	255.255.255.0
PC-1	VM-1	192.168.101.101	255.255.255.0
PC-2	Ethernet	DHCP Assigned	255.255.255.0
PC-3	Ethernet	DHCP Assigned	255.255.255.0
PC-4	Ethernet	DHCP Assigned	255.255.255.0

Figure 2.1.2 – IP addressing table.

Server Setup

Initial Configuration of Virtual Box and Ansible

A new virtual machine was created using Oracle VM Virtual Box on the Asus ZenBook 14. The virtual machine was set to use 2Gb of RAM and 25 Gb of hard disk storage on the host device.

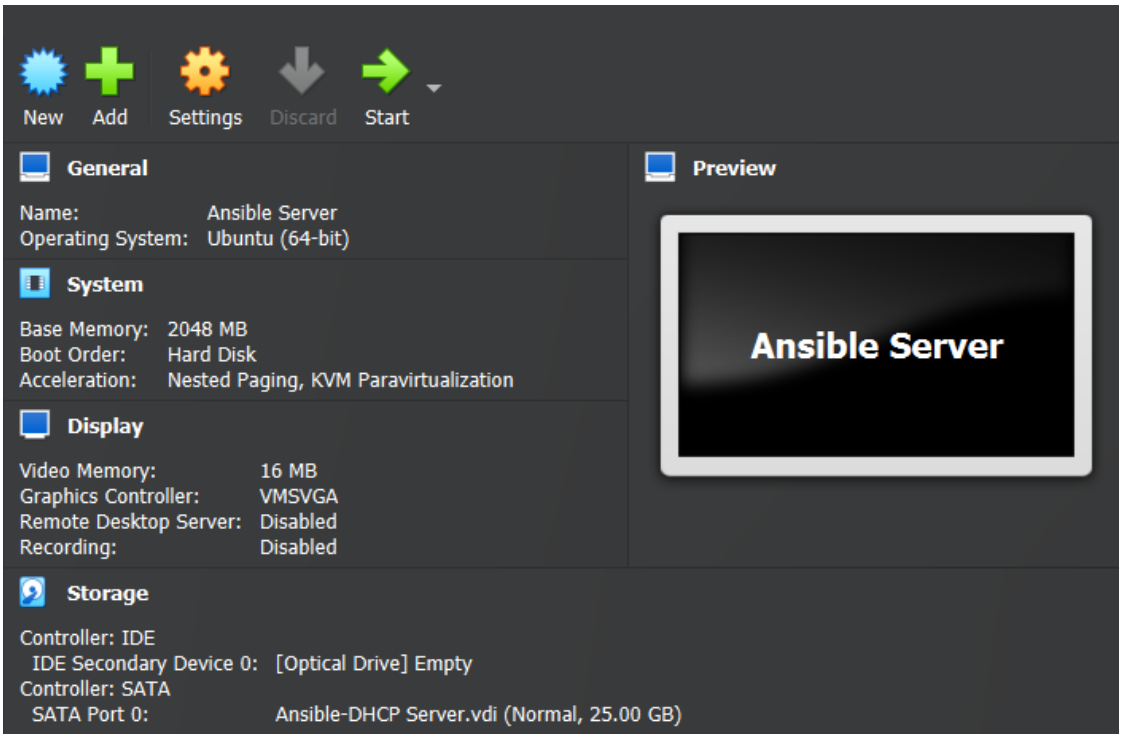


Figure 2.2.1 – Oracle VM Virtual Box configuration.

The most recent release of Ubuntu, version 22.04.2 LTS, was downloaded from https://ubuntu.com/download/desktop and installed on Ansible Server. Once the operating system was installed, Terminal was opened to begin setting up Ansible.

Ansible and its necessary software libraries, including Python and Jinja2 were installed to the server.

```
eli@eli-virtualBox:~$ sudo apt install python3-pip
```

Figure 2.2.2 – Python installation command.

Once the installation has completed, Ansible can be updated.

```
eli@server-1:~$ python3 -m pip install --eli ansible
```

Figure 2.2.3 – Ansible installation command.

The current version of ansible can be confirmed with

```
eli@server-1:~$ ansible --version
```

Figure 2.2.4 – Check Ansible version command.

For this procedure, Ansible 2.14.4 was used.

Figure 2.2.4 – Check Ansible version output.

After Ansible was installed, a static IP address of 192.168.101.101 was manually assigned to Virtual Machine 1 using the Ubuntu graphical user interface (GUI).

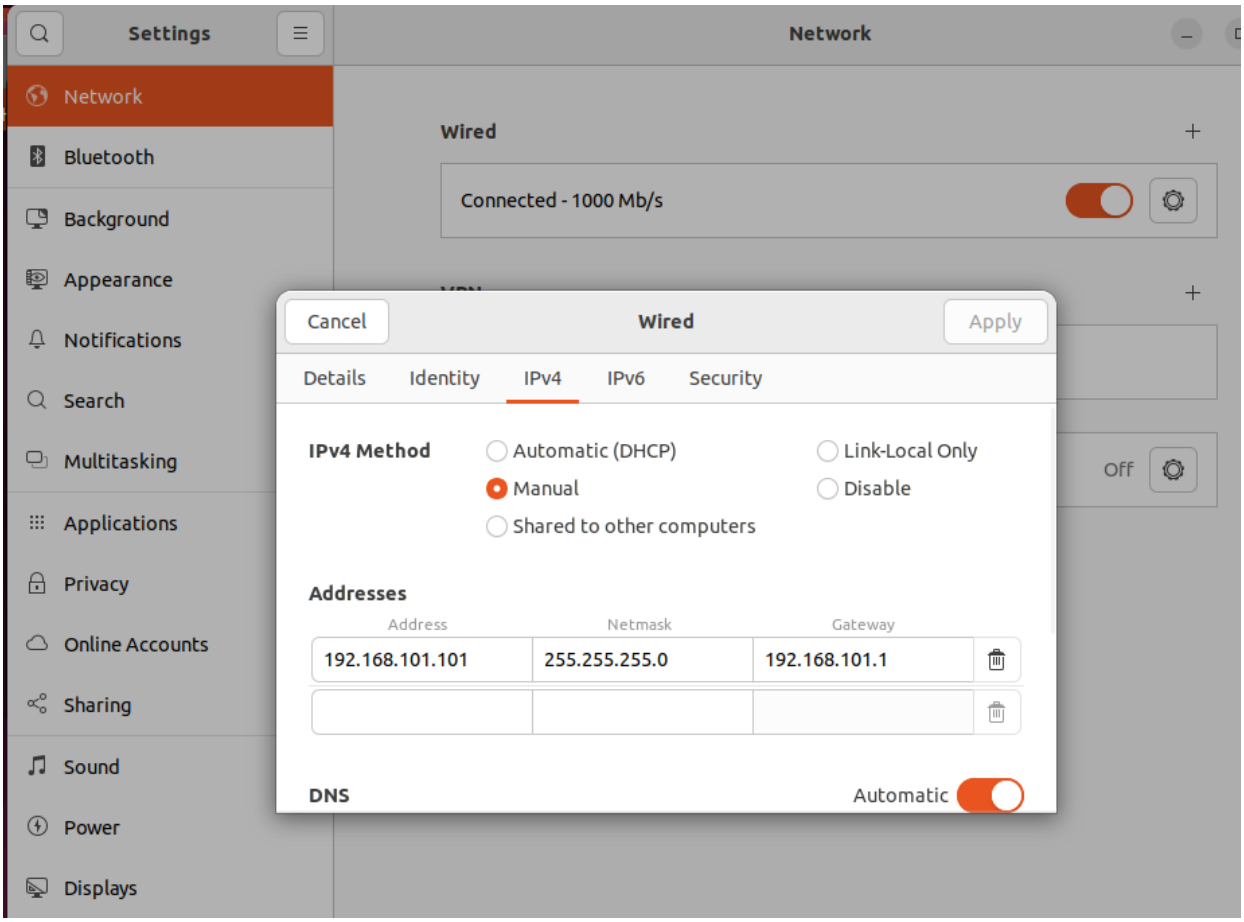


Figure 2.2.5 – Ansible Server IP configuration.

This IP is a member of the 192.168.101.0/24 subnet which is restricted to server VLAN 101. Assigning separate Virtual Local Area Networks, or VLANs, based on the type of traffic being transported increases network security because it divides traffic into separate communication channels such that a device on one VLAN cannot natively access data destined for a device on another VLAN unless routing has been specifically configured as such. VLANs separate a single physical switch into multiple virtual switches, limiting the scope of a networks broadcast domain and restricting what traffic could potentially be accessed by devices connected to the same switch (Global Information Assurance, 2003). In relation to this network environment, end-user devices connected to ALS-1 will be unable to view server traffic originating from the Ansible server using packet sniffing tools such as WireShark. This is

because users will be on a separate VLAN, VLAN 102. This security measure will ensure that the device information being gathered by Ansible cannot inadvertently be captured by end users.

Managed Switches

Full running configurations of DLS-1 and DLS-2 can be observed in Appendix B and Appendix C, respectively. The following configurations were made:

Basic device configuration:

- Hostnames for ALS1 and DLS1 were set.
- Domain command lookup was disabled.
- Automatic sign out was set to 3 minutes 30 seconds.
- All passwords were set to be encrypted.
- A local user account was created for both eli and ansible. Keeping the automation user separate will help with logging events. The password for user eli was set to **el!love$ansibl3** and the password for user ansible was set to **ansibl3love$el!**

Basic interface configurations:

- All unused ports were set to VLAN 999 and shutdown.
- Trunk ports were set to only allow necessary VLAN traffic.
- Access ports were set to access their intended VLAN.
- Spanning Tree was modified with the following:
  - Spanning Tree mode was set to Rapid PVST
  - Access ports were configured for PortFast and BPDU Guard.
- Port descriptions were created.

- VLANS were configured.
- IP addresses were assigned.
- DHCP pool was set up.
    - IP range 192.168.102.1-99 was reserved for potential access devices
    - Pool name was assigned to ACCESS_VLAN
    - Address range 192.168.102.100-1255
    - The default router address was set to be advertised as 192.168.102.1
- OSPF Routing
    - Process ID 1985
    - Loopbacks 2, 3, and 4 propagated
- Initial configurations were copied to the running configuration.

SSH

SSH is an essential component of Ansible's operation and functionality on network devices. A major advantage of using Ansible over other network automation product offerings is that it does not require a client to be running on the devices which it connects to (*Setting up SSH for Ansible*). This feature makes Ansible easy to setup because it can be seamlessly integrated with networks which already use SSH to manage network devices. This feature also makes network automation more affordable because managed nodes do not need to be capable of running special software for it to work, they simply require an SSH connection (Red Hat, 2018). Upon initial configuration of SSH, it was discovered that the switch being configured as DLS-1 did not support SSH with its current software image. The first symptom of this was first identified while attempting to define SSH as the sole protocol for remote management.

```
DLS1(config)#line vty 0 15

DLS1(config-line)#transport input ssh

                          ^

% Invalid input detected at '^' marker.

DLS1(config-line)#transport input ?

  all     All protocols

  none    No protocols

  telnet  TCP/IP Telnet protocol
```

Figure 2.4.1 – DLS-1 lacking SSH support.

Executing the show version command indicates the current system image does not support the K9 encryption functionality necessary for SSH connection. The operating system image will need to be replaced to support SSH.

```
DLS1#show version | include System image

System image file is "flash:c3750-ipbase-mz.122-35.SE5/c3750-ipbase-mz.122-35.SE5.bin"
```

Figure 2.4.2 – Current DLS-1 image lacks K9 license.

Upgrading the Operating System of DLS-1 to Support SSH

Hosting FTP on Server-1

The first step in upgrading the operating system on DLS-1 was to create an FTP server on Server-1. Because the Cisco Catalyst 3750G did not have any USB ports for updating software from removeable memory, downloading a new image file over FTP was the only way to upgrade the operating system. VSFTPD was installed to Server-1 by performing the following terminal command:

```
sudo apt install vsftpd
```

Figure 2.4.3 – Installing ftp service on DLS-1

The next step in upgrading the operating system was to obtain a new image file of Cisco IOS; this was done by opening Mozilla Firefox on Server-1, navigating to https://software.cisco.com/download/home and logging in. Using the Select a Product search bar to search for the switch being used, a Catalyst 3750V2-48PS Switch, and navigating to IOS Software, a serviceable image was found. The IP BASE image supports K9 level encryption, indicating it will support SSH.

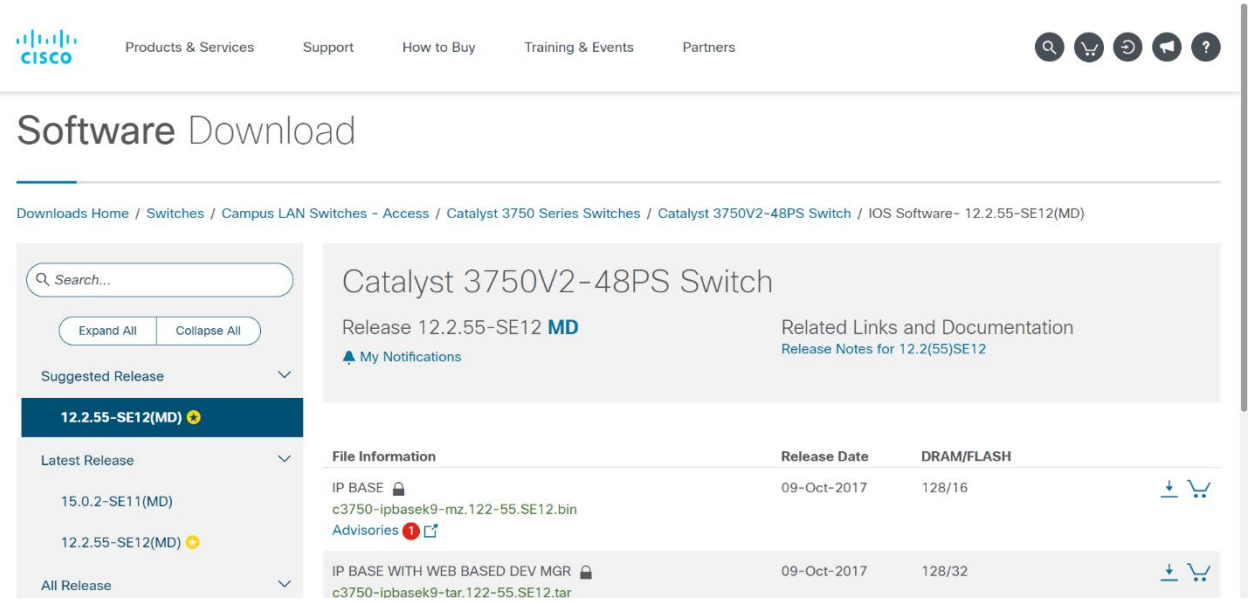


Figure 2.4.4 – Software image for Catalyst 3750

After clicking the Download icon on the right-hand side of the screen, the .bin image file was downloaded to the server and stored under /home/eli/Downloads.

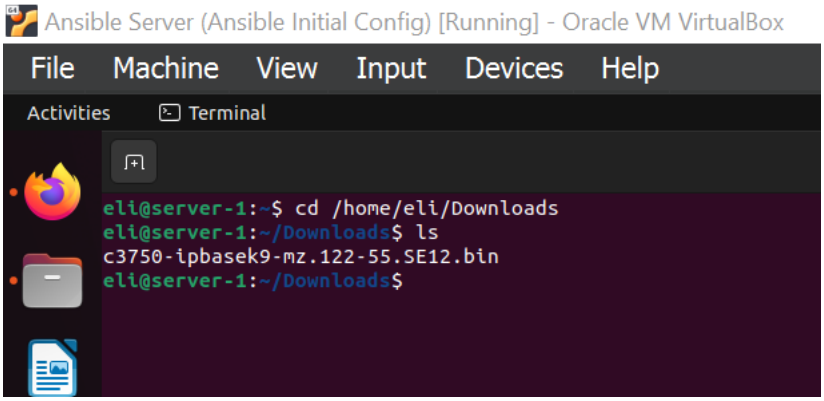


Figure 2.4.5 – DLS-1 image file location.

Installing a New Image File on DLS-1

The image file can now be accessed on DLS-1 through an FTP connection to Server-1.

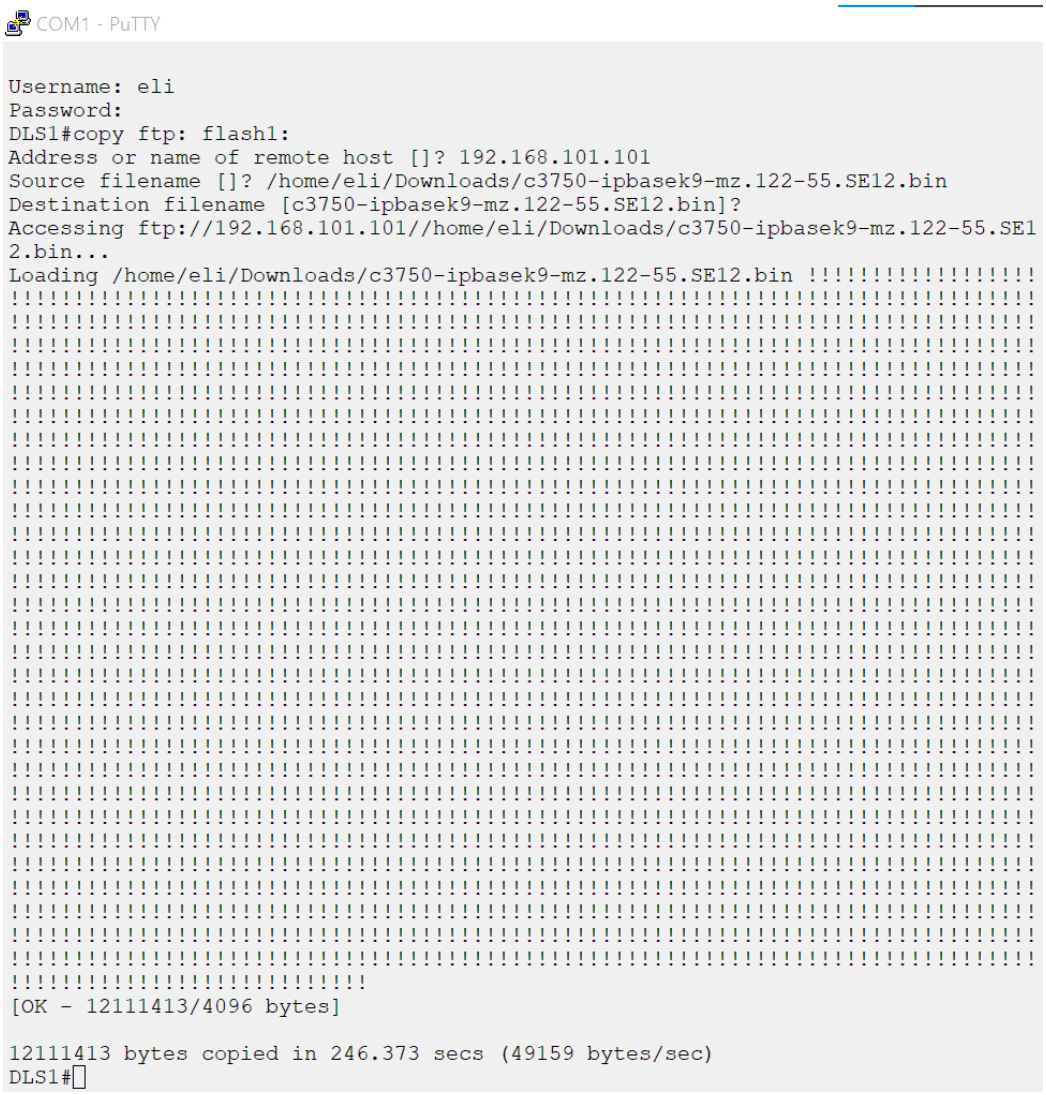


Figure 2.4.6 – Installing image file via FTP.

The new image file, which is currently stored in flash memory on DLS1, can now be set as the boot file.

```
DLS1(config)#boot system switch all flash:c3750-ipbasek9-mz.122-55.SE12.bin
```

Figure 2.4.7 – Changing boot file on DLS-1

After writing the configuration file to memory and reloading the switch, DLS1 boots using the new image file. This can be verified with the show version command.

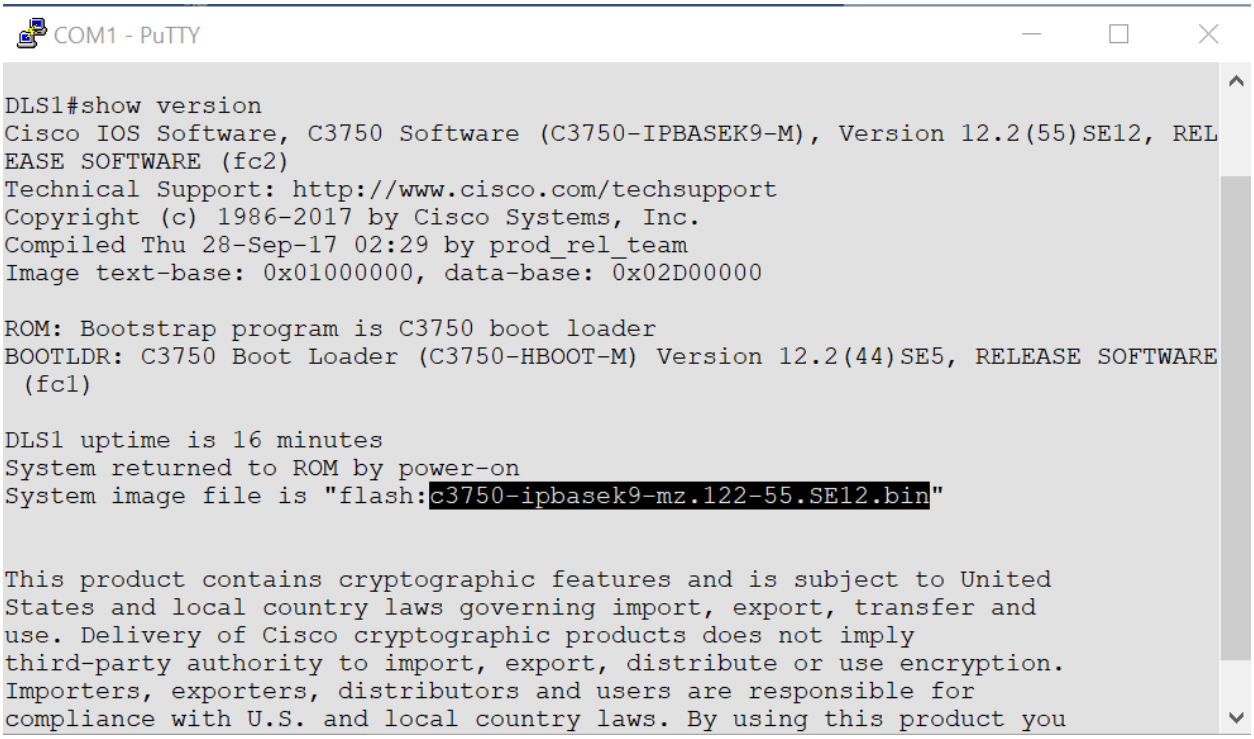


Figure 2.4.8 – DLS1 now supports SSH

Configuring SSH on DLS-1

The domain was set to be named uakron.edu as a placeholder for what would typically be the organization's URL. SSH was set to be the only permitted remote transfer protocol to the device, ensuring legacy Telnet would not create an unnecessary security risk. Remote sessions were set to use local accounts for authentication. SSH version 2 was enabled, and encryption keys were generated with a length of 2048 bits. This bit length will allow for the use of SSHv2 which is required by Ansible and will provide maximum encryption.

```
DLS1(config)#ip domain-name uakron.edu

DLS1(config)#line vty 0 15

DLS1(config-line)#login local

DLS1(config-line)#transport input ssh

DLS1(config-line)#ip ssh version 2

DLS1(config-line)# DLS1(config)#crypto key generate rsa

How many bits in the modulus [512]:2048
```

Figure 2.4.9 – SSH configurations on DLS-1

The most up-to-date version of Cisco IOS available for DLS-1 was only capable of negotiating SSH keys using Diffie-Hellman Group 1 key exchange. The Diffie-Hellman Group 1 key exchange method is considered legacy and is therefore no longer supported by most modern applications (*Legacy options*) nor is it permitted on Ubuntu 22.04 by default. In a production environment, older key exchange methods should not be used because their encryption can potentially be cracked. Because this lab is intended for demonstration purposes only, the older Diffie-Hellman Group 1 key exchange method was manually added to the list of negotiable key exchange methods on VM-1.

```
Host 192.168.0.1 192.168.0.100
    KexAlgorithms +diffie-hellman-group1-sha1
    HostKeyAlgorithms +ssh-rsa
    PubkeyAcceptedKeyTypes +ssh-rsa
    Ciphers aes128-cbc,3des-cbc,aes192-cbc,aes256-cbc
```

Figure 2.4.10 – Modifying KEX on VM-1

Part 3: Procedure

Ansible Inventory

After the necessary software libraries have been installed and network devices have been setup, it is time to begin configuring Ansible. Ansible requires the use of an inventory file. The inventory file contains a list of devices that an Ansible playbook will be run on, otherwise known as managed nodes (Meijer, 2022). Devices are specified by associating a hostname with an IP address and can be organized into groups so that multiple devices can be called in a playbook at once. The inventory file is also capable of storing variables related to the devices listed, such as SSH credentials and Ansible module specification. An inventory file can be specified each time a playbook is run using the -i option or it can be defined within the Ansible config file for a more permanent solution (Meijer, 2022). By default, Ansible searches for a file by the name of "hosts" under /etc/ansible directory (Juniper, 2021). This location would typically be modified to reflect a more structured directory system in a production environment because a single server may host many Ansible playbooks and multiple inventory files, requiring the need for

subdirectories. For this exercise, however, the default location will be used as few files will be required. The inventory file can be created using the touch command in the Ubuntu terminal.

```
eli@server-1:~$ touch /etc/ansible hosts
```

Figure 3.1.1 – Creating the inventory file.

The inventory file was opened using Ubuntu's built-in text editor through the GUI and both DLS1 and ALS1 were added under their respective group names: "routers" and "switches." Network CLI was specified as the network connection to ensure Ansible could properly communicate network commands. Cisco IOS was defined as the operating system so that Cisco specific modules could potentially have been used. In a production environment, when specifying SSH credentials, it is recommended that encrypted keys be generated on the server and referred to in the inventory file or playbook. For testing purposes, SSH credentials were defined in cleartext.

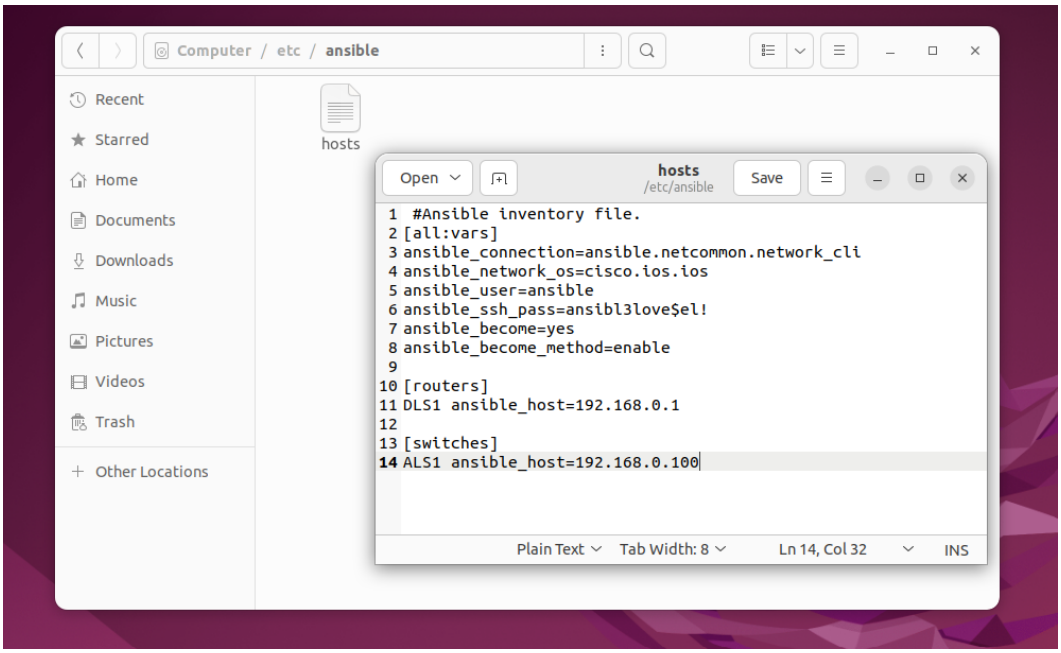


Figure 3.1.2 – Inventory file configuration.

Ansible Playbook

Gather Facts

To create a playbook, the proper directories first needed to be set up. In the Ubuntu Terminal, the Ansible directory was created, and the current user was given full privileges to all subdirectories.

```
eli@server-1:$ mkdir /etc/ansible

eli@server-1:~$ sudo chmod -R 777 /etc/ansible

[sudo] password for eli:
```

Figure 3.2.1.1 – Creating Ansible folder.

Next, three text files were created in the Ansible folder, “information.yml”, “facts.yml”, and “test.yml”.

```
eli@server-1:~$ /etc/ansible$ touch information.yml facts.yml test.yml
```

Figure 3.2.1.2 – Creating playbook files.

Information.yml was used as the final playbook while facts.yml and test.yml were used to test individual modules prior to integrating them with the information playbook.

The first step in configuring the text file was to create a top-level configuration name. This gives the playbook a description so that its function is easily recognizable (Sprygada, 2023). Three dashes were also included in the first line to indicate the beginning of the playbook.

```
---

- name: Gather information from all routers and switches
```

Figure 3.2.1.3 – Creating playbook name.

Next, the gather_facts parameter was set to false. This option is turned on by default in Ansible; it automatically gathers information about network devices and configures them as a variable when a playbook is run. The following playbook was created as a sample to demonstrate what can be collected using Ansible's built-in gather facts feature:

Figure 3.2.1.4 – Sample playbook for gathering generic device facts.

The debug module was used to display the output of the collected variable, ansible_facts. The playbook was run with the following command:

```
eli@server-1:$ ansible-playbook /etc/facts.yml
```

Figure 3.2.1.5 – Command used to run a playbook.

Anytime a playbook is run, this general format is used, exchanging just the playbook name for the playbook being run.

This was the result when the playbook was run in the Ubuntu terminal:

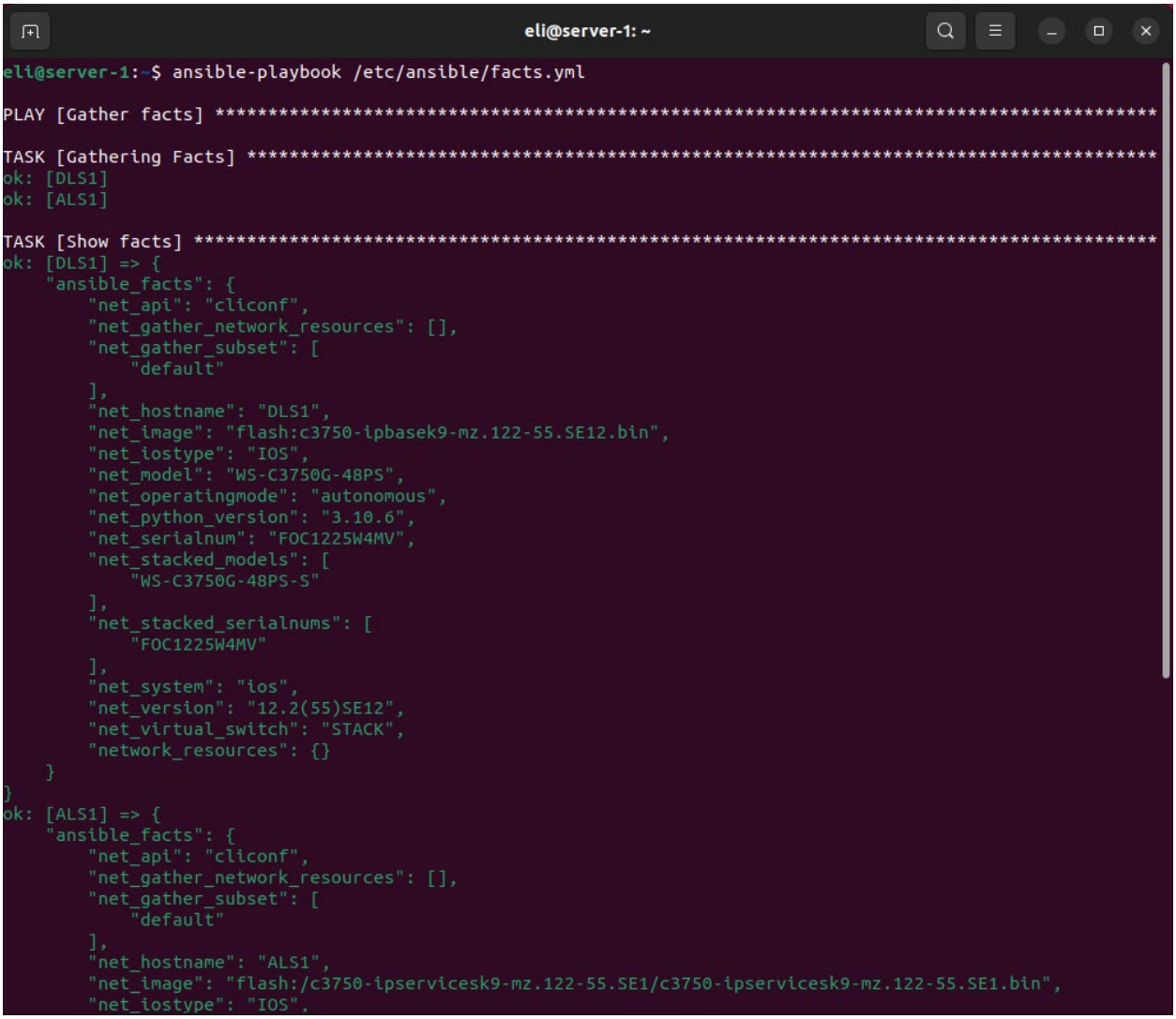


Figure 3.2.1.6 – Output of generic fact gathering module.

The information provided can be used to create dynamic playbooks which rely on device attributes such as the operating system in use, OS version number, or device model. The only piece of information that would be useful from this output as it relates to the goal at hand is the net_hostname as it could be used to specify the switch which each device is connected to. For

demonstration purposes, the built-in Ansible gather_facts module will be disabled. This will allow both the reader and the author the opportunity to see how switch commands can be used to gather information from a device using Ansible programming. Understanding this process will make it easier to understand how to gather more complex data types, such as MAC address or IP address later.

Included with the Cisco IOS collection in Ansible are fact gathering modules, similar to the one built into Ansible. Cisco-specific fact gathering modules can collect networking specific facts from devices running IOS. Like Ansible's built in gather_facts module, the Cisco fact-gathering module will also gather facts by default. Here is a sample playbook that skips the Ansible gather_facts module and intends to gather information about interfaces using Cisco's fact gathering module:

Figure 3.2.1.7 – Sample playbook demonstrating Cisco IOS fact gathering.

Here is the resulting output when running the playbook on Cisco 3750G switches:

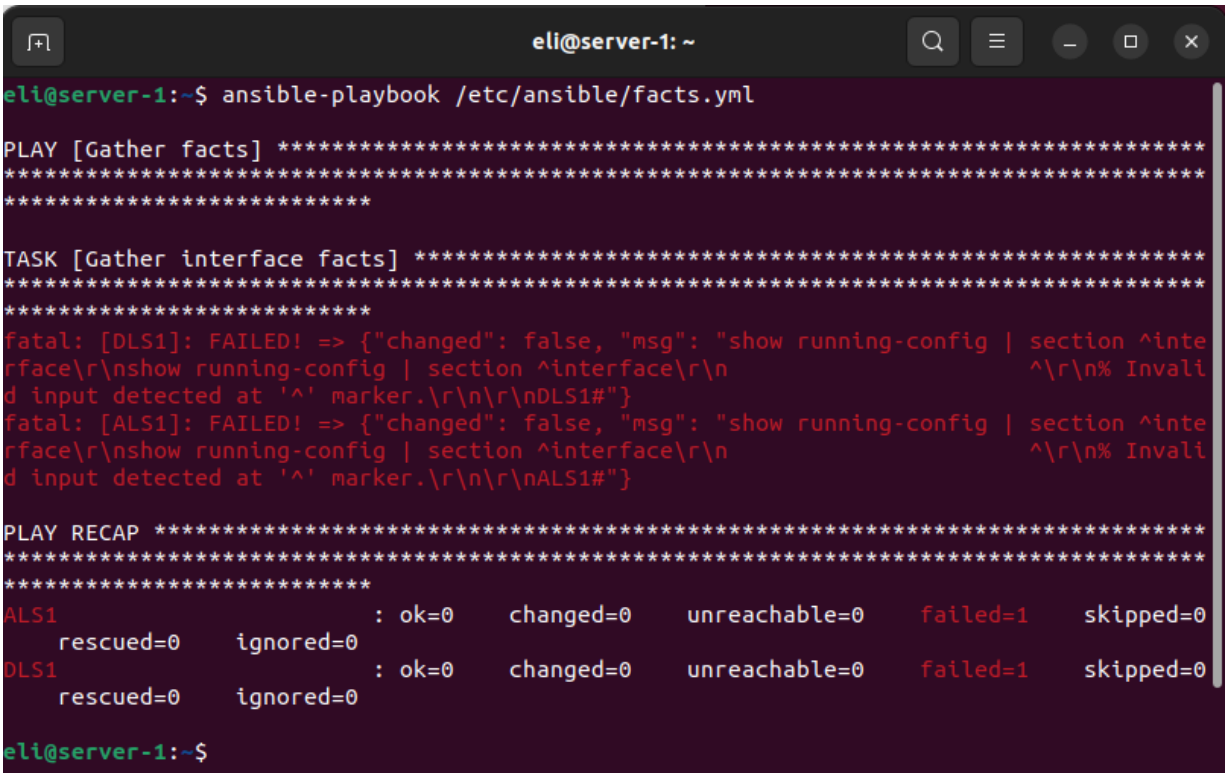


Figure 3.2.1.8 – Output of Cisco IOS fact gathering module.

The Cisco IOS fact gathering module in Ansible does not work on the devices in use because they do not support the command option being used; the “section” output filter is not supported on Cisco IOS 12.2. Rather than using the Cisco IOS fact gathering module, this exercise will be done using manual specification of switch commands to ensure granular control over the data being gathered and insurance that only legacy commands are being used.

One attribute of Ansible that is often cited as a strength when compared to other network automation products is the fact that it is agentless, meaning it requires no additional software to be run on the managed nodes being used (Red Hat, 2018). This means that Ansible can operate on a wide variety of operating systems, including legacy devices. While the Cisco IOS fact gathering module would have been helpful, as it could theoretically gather MAC addresses and

IP addresses automatically, its lack of support on the devices did not entirely prohibit the continuation of the task at hand.

Hostnames

After the gather_facts module was disabled, the tasks section of the play was defined, and the first task was given a name.

```
tasks:

  - name: Gather switch name and assign it to a variable
```

Figure 3.2.2.1 – Creating a task for gathering hostnames.

The task was then specified to utilize Cisco's integrated networking commands, followed by specification of the command to be used, written with Cisco IOS syntax in mind.

```
    cisco.ios.ios_command:

      commands: show running-config | include hostname

    register: switch_name
```

Figure 3.2.2.2 – Defining the actions of the hostname gathering task.

As it is currently written, this playbook will create a new variable for each switch it is run on, however, it will also contain the word "hostname" as it is included in the raw output of the specified command. A sample playbook was created to demonstrate what this output would currently look like.

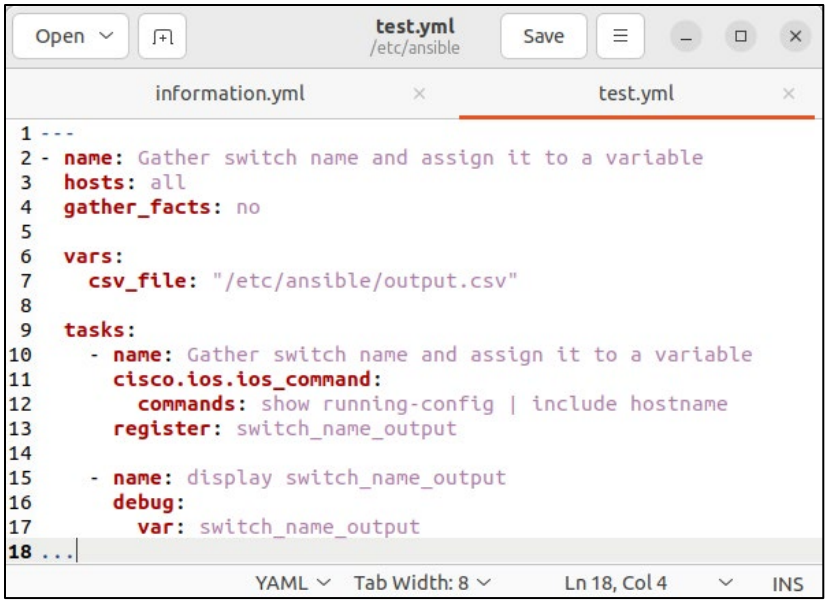


Figure 3.2.2.3 – Sample playbook for gathering unfiltered switch hostname.

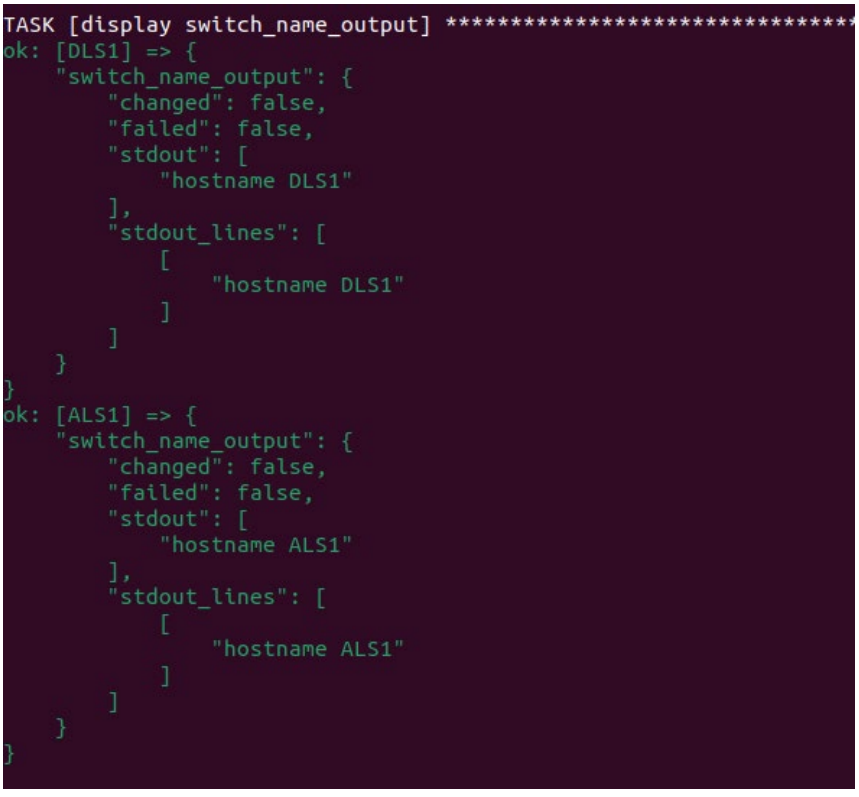


Figure 3.2.2.4 – Output of sample playbook for gathering unfiltered switch hostname.

There are two items to note with this output. As previously stated, the word “hostname” in each variable; this will be attended to in a moment. Another attribute of interest is the fact that the switch_name_output variable returns the same line of text under two separate variable lists: stout and stdout_line. When Ansible uses the CLI connection method, by default variables will be returned in both their raw format as well as their normal output as it is understood by Ansible just in case the two storage methods differ. This allows Ansible to be both flexible and accurate in how variables are defined, however, in our case both are the same. When calling variable in either a debug module or in another task, the normal standard output can be referenced by itself to ensure just one output is being included.

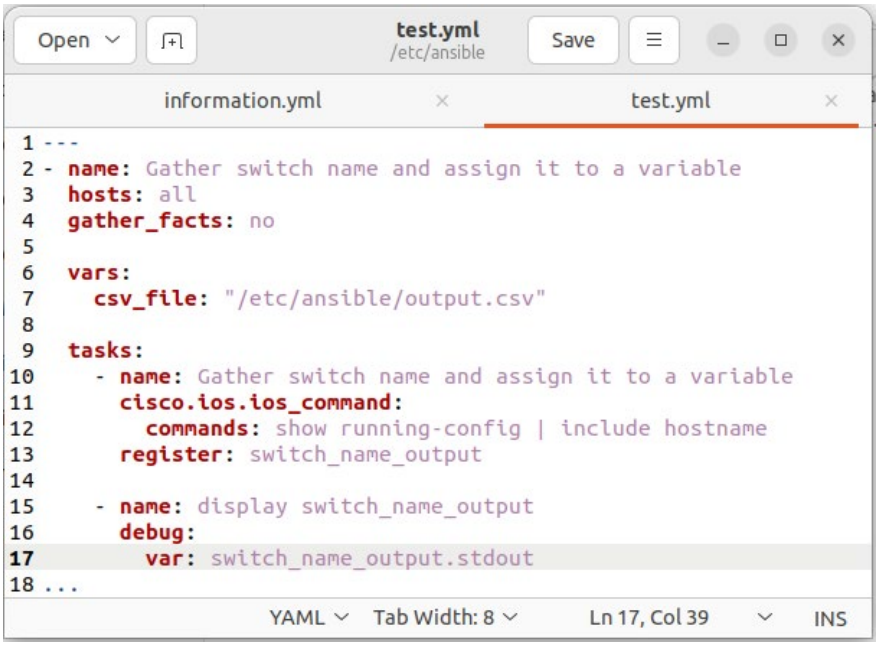


Figure 3.2.2.5 – Sample playbook filtering for standard output.

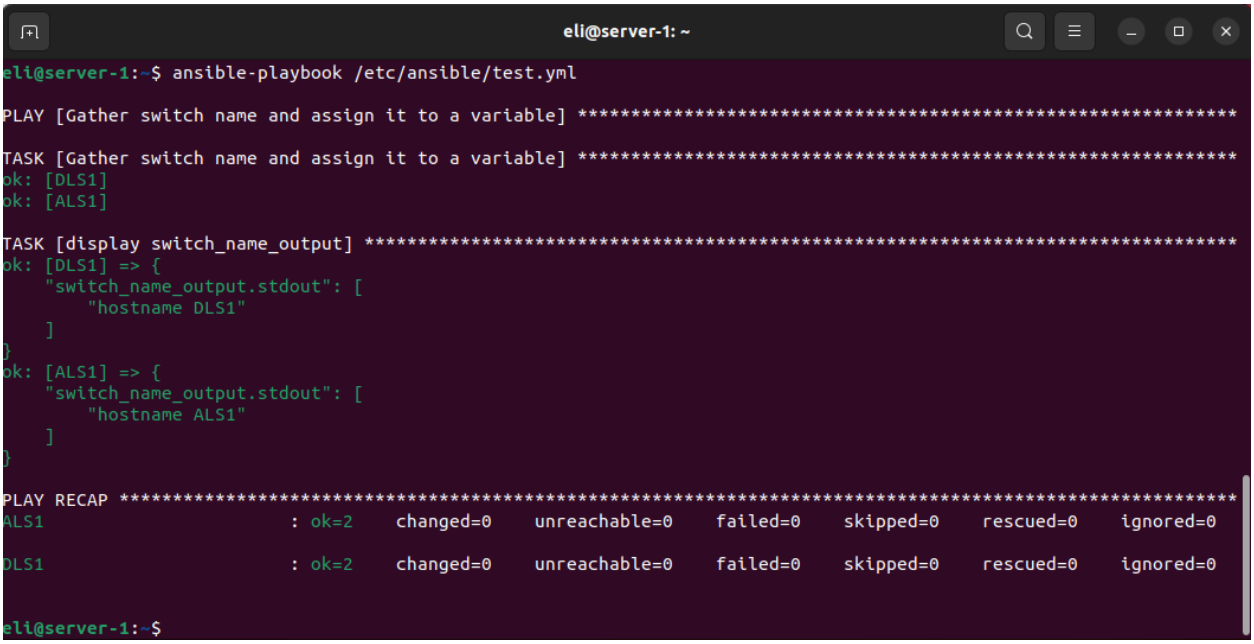


Figure 3.2.2.6 – Output of filtering for standard output.

Now that standard output has been specified, there is only one instance of the switch_name_output variable. We still need to remove that word “hostname,” however, so that it does not show up in our final database. To do this, the regex_search module was used. This module can read a source variable and filter it to only include a singular instance of a regular expression as it is defined by the user. In this case, the switch_name_output variable was defined as the source variable and regex_search was set to stop searching if it found either of the hostnames of the switches being used, ALS1 or DLS1.

```
- name: Format hostname variable by removing the word "hostname"

    set_fact:

      hostnames: "{{ switch_name_output | regex_search ('(ALS1|DLS1)')}}"
```

Figure 3.2.2.7 – Syntax of regular expression search filter.

This task will filter the raw output to only include a potential hostname and assign it to a newly defined variable, hostnames. In a production environment, this filter method would not work well because it would require device names to be specified in more than one location. To adhere to the DRY principle, it would make more sense to specify the inventory document here instead. Because there are very few managed nodes in this topology, manually specifying switch names to include will work fine. Here is a sample playbook created to demonstrate this as well as its output:

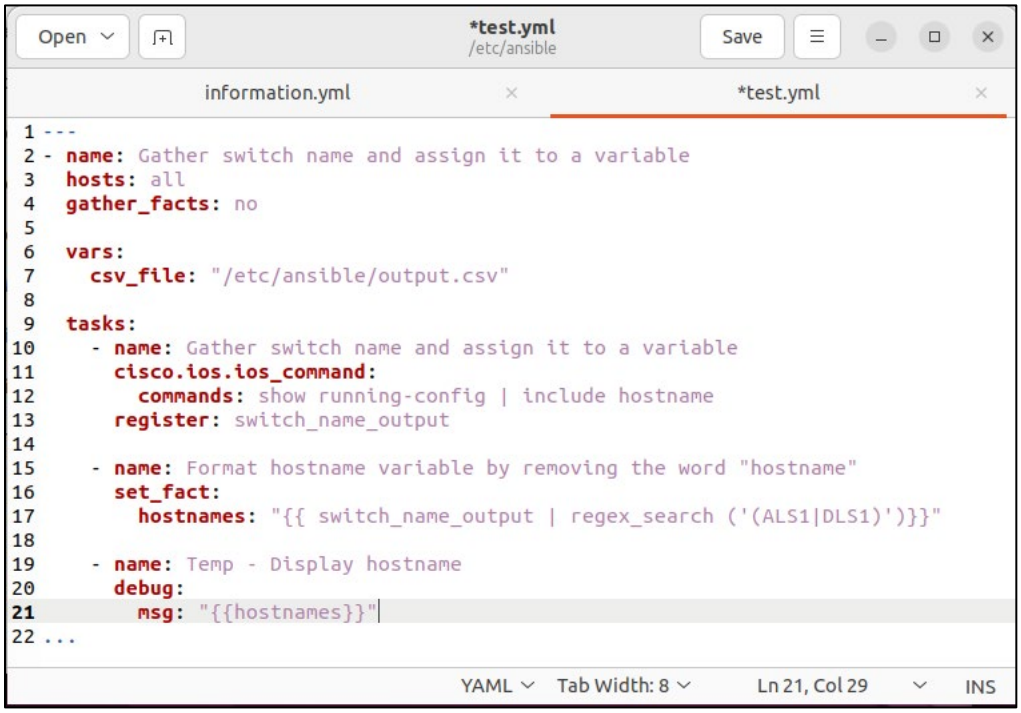


Figure 3.2.2.8 - Sample playbook for filtering hostname with a regular expression.

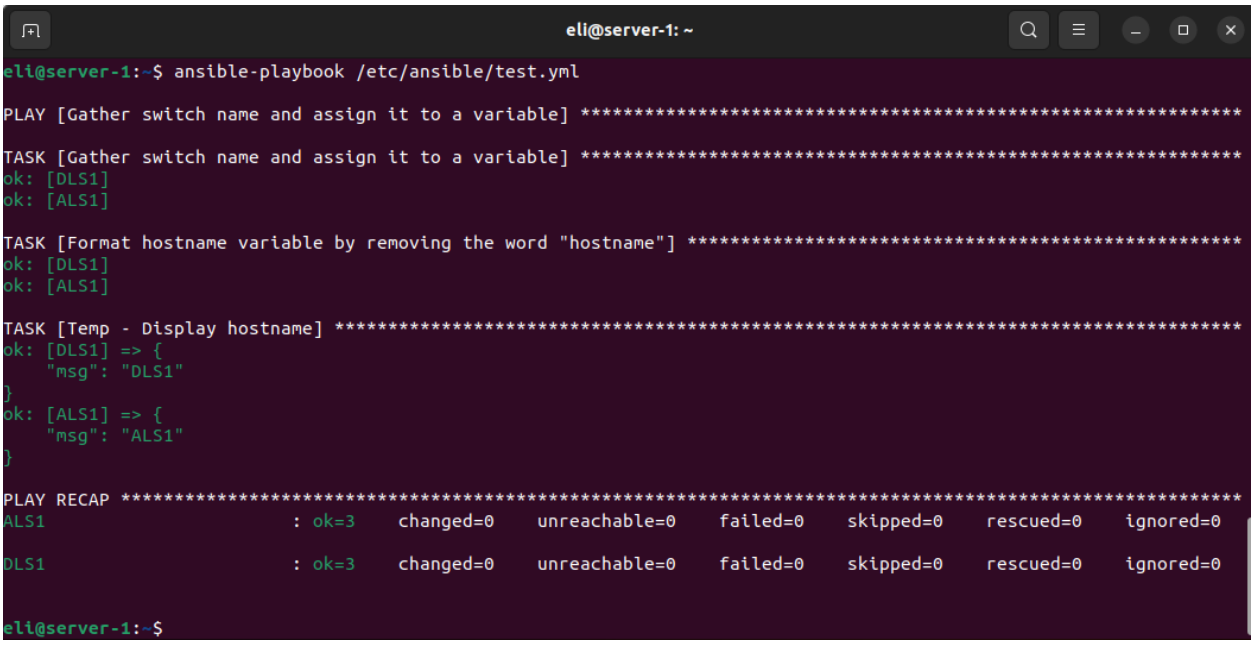


Figure 3.2.2.9 – Output of fully filtered hostname.

After the filter was applied, the variable “hostnames” was returned as just one instance of the hostname for each switch. The concepts used to accomplish this will be applied in a similar fashion to gather information from other switch outputs.

The hostname task was added to information.yml

Interfaces

Interface numbers can be gathered from the output of a show interfaces status output and assigned to a new variable called all_switchports, similar to how switch hostnames were collected from a show run output in the previous section.

```
- name: Gather all interface numbers into one variable
    cisco.ios.ios_command:
      commands: show interfaces status
    register: all_switchports
```

Figure 3.2.3.1 – Creating a task to produce a show interfaces status output.

Because the all_switchports variable will include everything from the command output if left as is, the variable must be run against a filter to only include the switchports themselves. This process becomes a bit more complicated because, rather than searching for two names, this filter must be able to identify the switchport type, stack number, module number, and switchport number of each port. Additionally, this filter will need to gather more than one instance of the expression being searched for. To do this, the regex_findall module was be used to gather every instance of the expression specified. For this task, the previously defined variable all_switchports were parsed for strings that started with either Gi, Fa, or Te to ensure each switchport type could potentially be gathered. The module then searched for 3 instances of numbered expressions, separated by a forward slash. The result was stored in a variable to switchports_tall.

```
- name: Include only interface numbers in a variable
    set_fact:
      switchports_tall: "{{ all_switchports.stdout_lines | regex_findall('(Gi|Fa|Te)([0-9]+)/([0-9]+)/([0-9]+)') }}"
```

Figure 3.2.3.2 – Creating a task to filter the show interfaces status output.

The following test playbook was used to demonstrate why this variable was named switchports_tall.

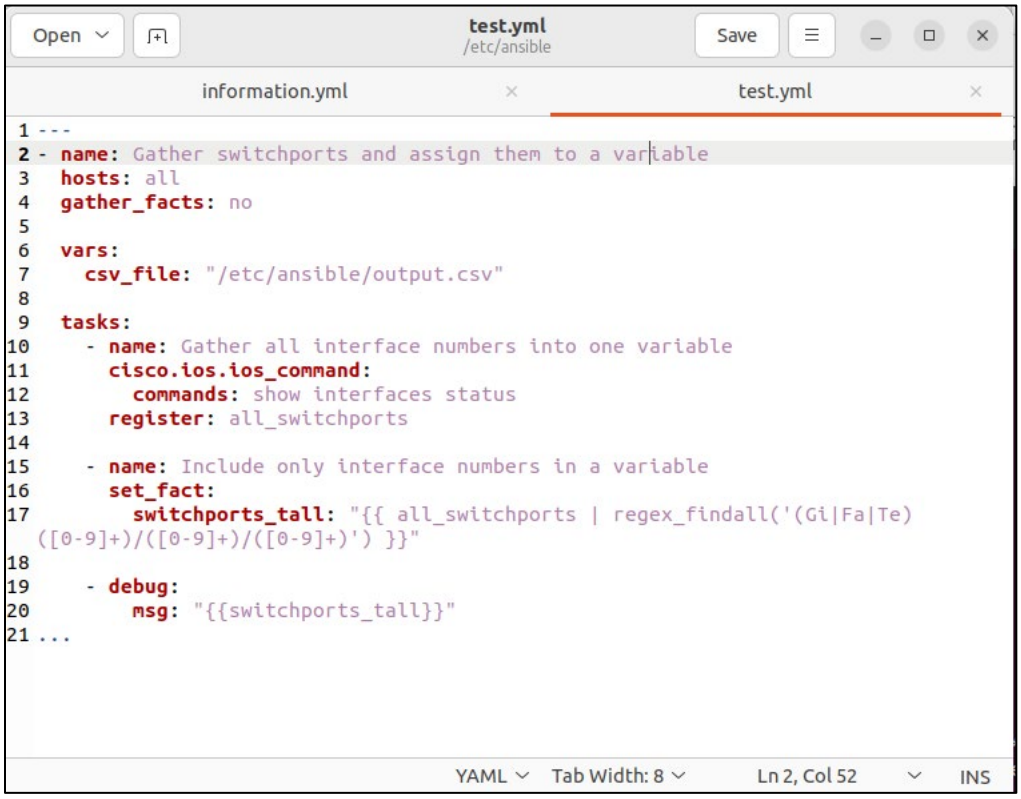


Figure 3.2.3.3 – Sample playbook for gathering partially filtered interface numbers,

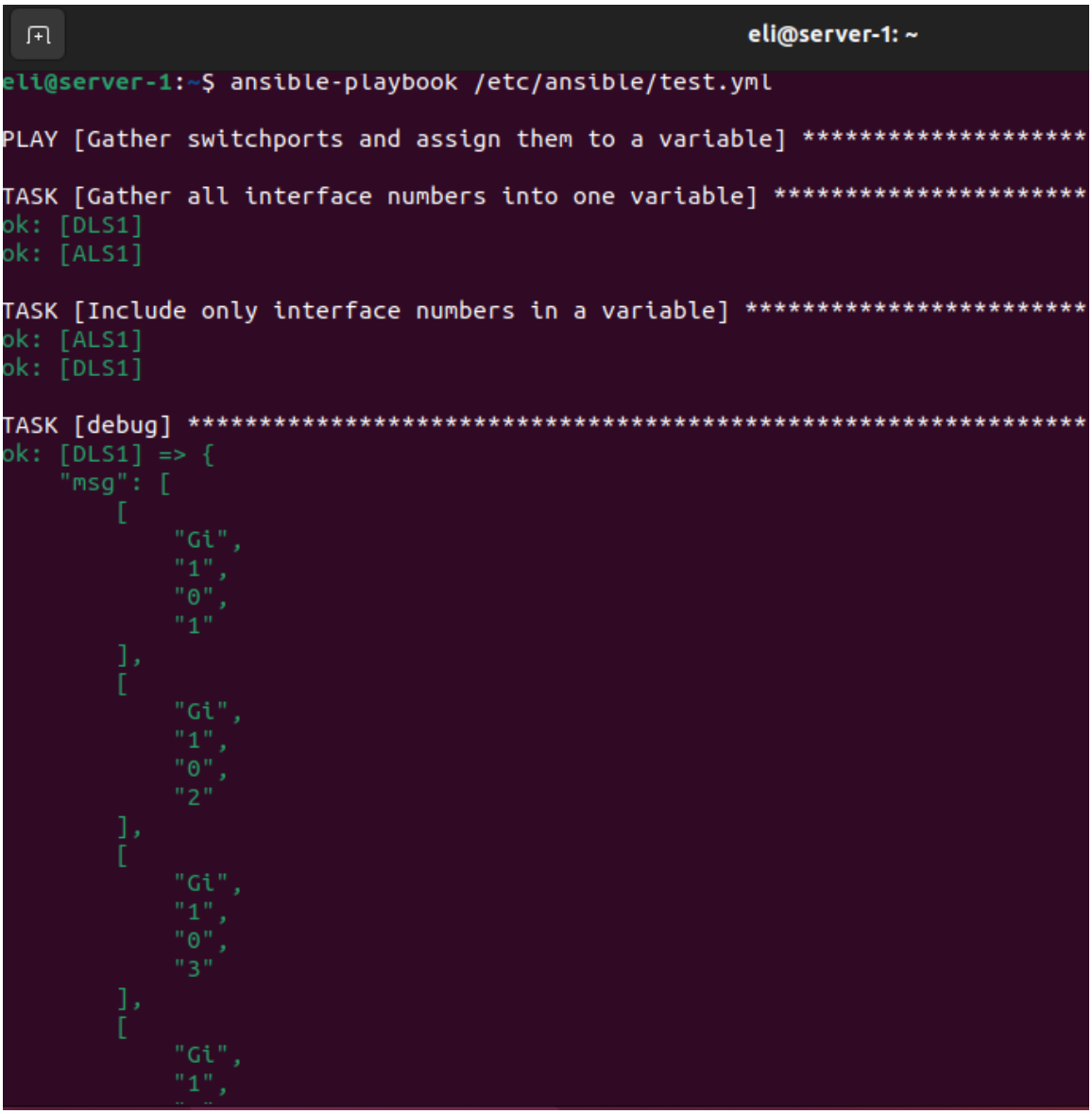


Figure 3.2.3.4 – Output of partially filtered interface task.

While the playbook did gather the specified expressions, it also separated each expression into separate elements. To better display these switchports in the final database, the switchport elements needed to be combined. This was done using a combination of the map module and the split module. The map module was used to replace every instance of a comma with a forward slash. The test playbook was modified to incorporate this.

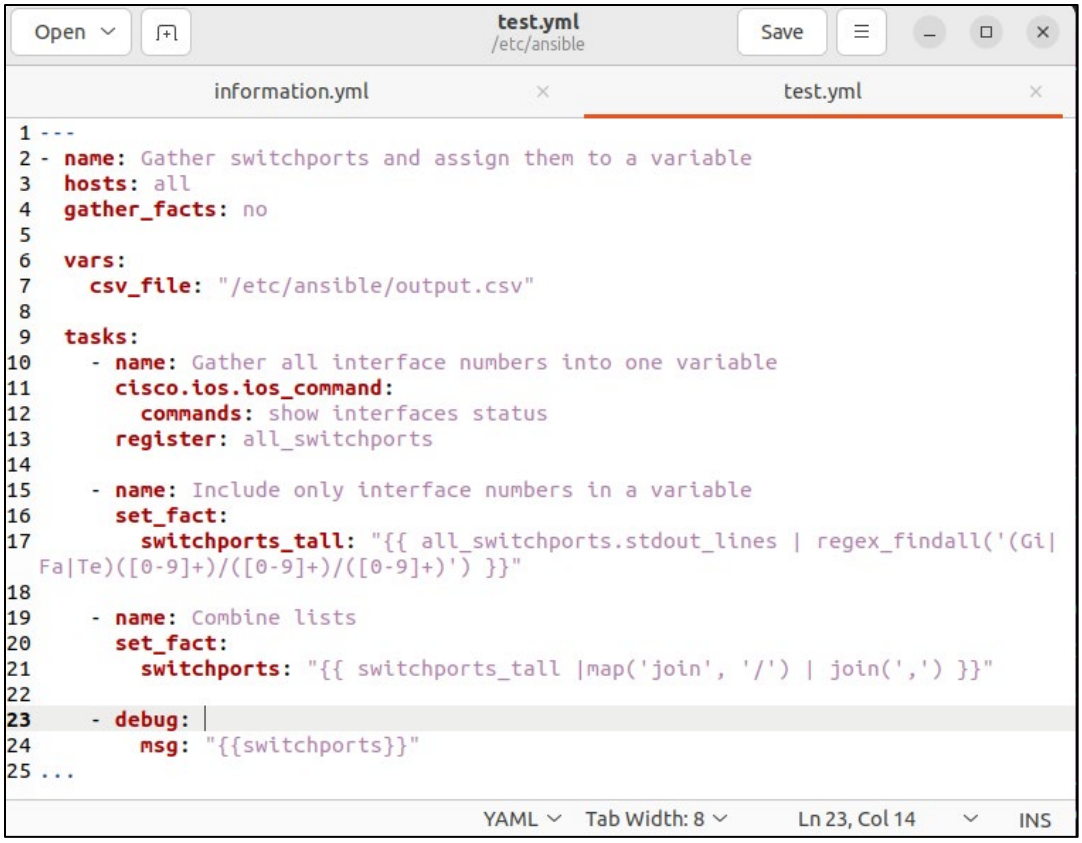


Figure 3.2.3.5 – Sample playbook for further filtered interface numbers.

The result is a singular list that contains every switchport, separated by commas.

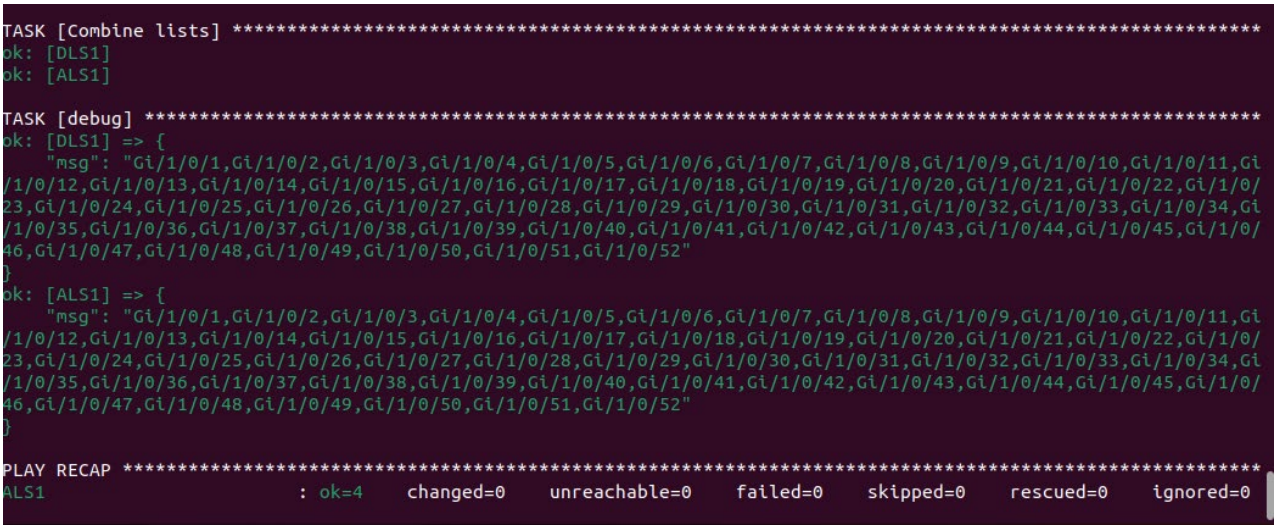


Figure 3.2.3.6 – Output of further filtered interface task.

To separate each switchport into its own element, the split module was used to divide the list at every instance of a comma.

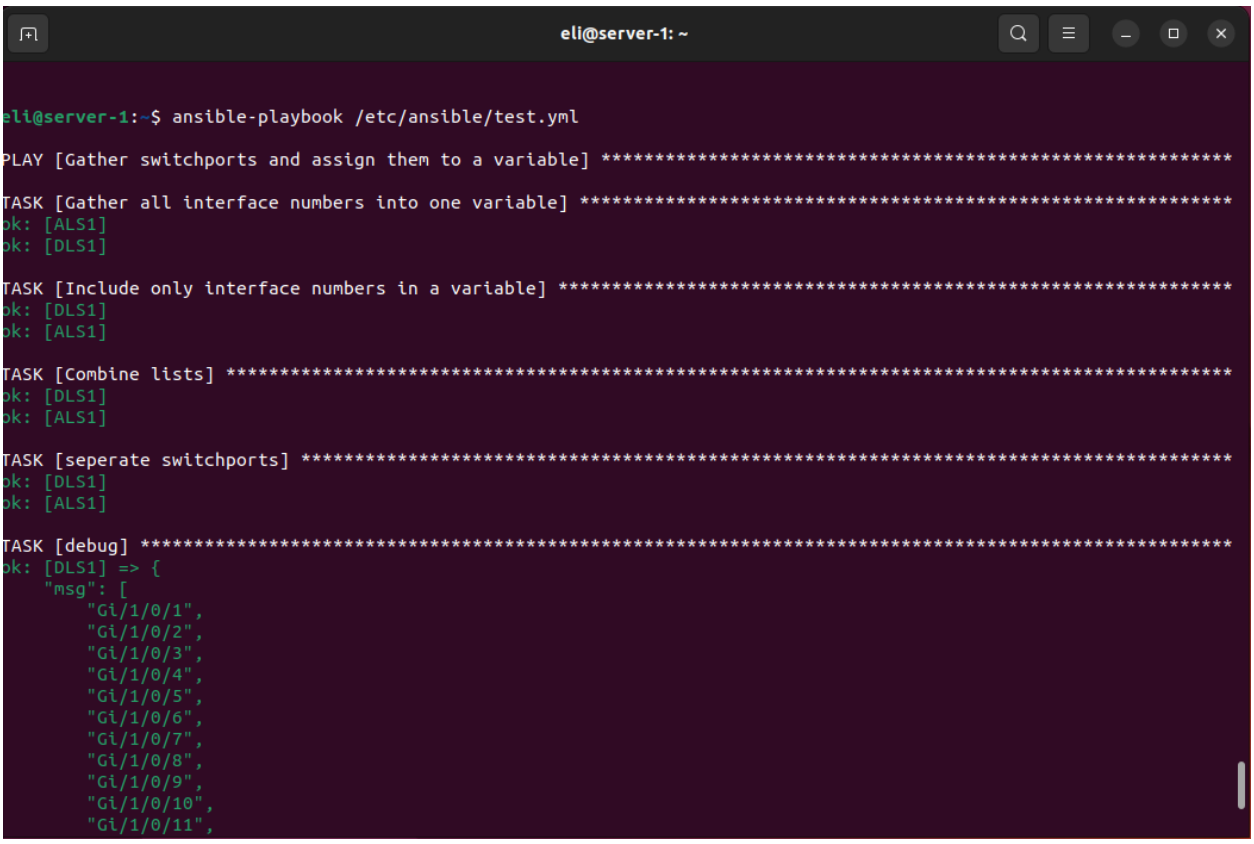


Figure 3.2.3.7 – Output of fully filtered interface task.

This results in the desired output of a list of switchports for each switch that the playbook is run on. The interfaces task was added to information.yml.

MAC Addresses

Using all the previously explained modules, MAC addresses were gathered from a show mac address-table output and run against filters for proper formatting. The biggest difference here was determining which expressions to search for in the regex_findall module. MAC addresses are composed of 12 hexadecimal digits that can be represented differently depending on the operating system being used. In the Cisco IOS, the mac address table represents these

addresses as 3 octets, each containing 4 alphanumeric values, with octets being separated by a decimal. Figure two demonstrates this formatting as seen in ALS1's mac address table output.

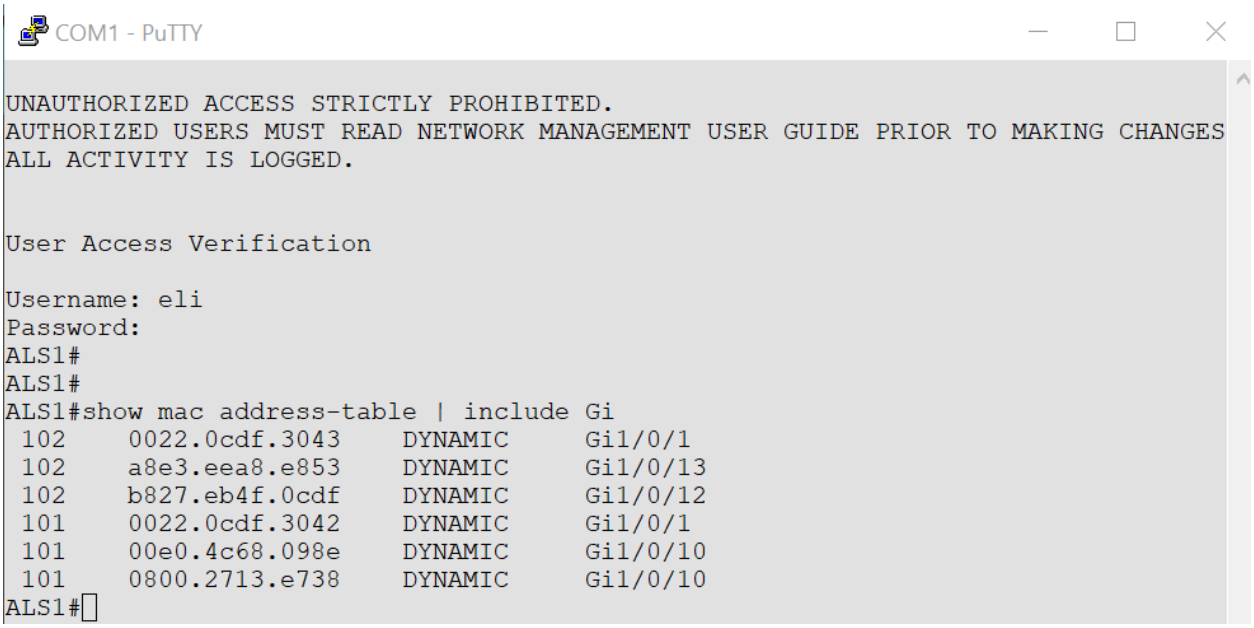


Figure 3.2.4.1 – ALS-1 MAC address table.

To filter this output in ansible, regex_findall was used to parse the output for regular expression characters in the defined format and the map. Split modules were once again used to format the output appropriately.

Open *test.yml /etc/ansible Save

information.yml *test.yml

```
1 ---
2 - name: Gather switchports and assign them to a variable
3   hosts: all
4   gather_facts: no
5
6   vars:
7     csv_file: "/etc/ansible/output.csv"
8
9   tasks:
10    - name: Gather all MAC addresses into one variable
11      cisco.ios.ios_command:
12        commands: show mac address-table | include Gi|Te|Fa
13      register: all_macs
14
15    - name: Include MAC addresses
16      set_fact:
17        macs: "{{ all_macs| regex_findall('([a-f|A-F|0-9]{4}).([a-f|A-F|0-9]{4}).([a-z|A-F|0-9]{4})') }}"
18
19    - name: Combine lists
20      set_fact:
21        macs: "{{ macs |map('join', '.') | join(',') }}"
22
23
24    - name: seperate macs
25      set_fact:
26        macs: "{{macs | split (',')}}"
27
28    - debug:
29        msg: "{{macs}}"
30 ...
```

Figure 3.2.4.2 – Sample playbook for gathering and filtering MAC addreses.

eli@server-1: ~

```
TASK [debug] *******************************************
**************************************************
ok: [DLS1] => {
    "msg": [
        "00e0.4c68.098e",
        "0800.2713.e738",
        "0026.6c7a.e706",
        "b827.eb4f.0cdf",
        "d057.4c95.97c1",
        "00e0.4c68.098e",
        "0800.2713.e738",
        "0026.6c7a.e706",
        "b827.eb4f.0cdf",
        "d057.4c95.97c1"
    ]
}
ok: [ALS1] => {
    "msg": [
        "0022.0cdf.3042",
        "00e0.4c68.098e",
        "0800.2713.e738",
        "0022.0cdf.3043",
        "0026.6c7a.e706",
        "b827.eb4f.0cdf",
        "0022.0cdf.3041",
        "0022.0cdf.3042",
        "00e0.4c68.098e",
        "0800.2713.e738",
        "0022.0cdf.3043",
        "0026.6c7a.e706",
        "b827.eb4f.0cdf",
        "0022.0cdf.3041"
    ]
}
```

Figure 3.2.4.3 – Output of MAC address sample playbook.

The MAC address task was added to information.yml.

IP Addresses

To produce a list of IP addresses, the output of the “show arp” command was registered to a variable. Next, the output was filtered with the regex_finall module, this time slightly modified to parse the output for the format of a standard IPv4 address.

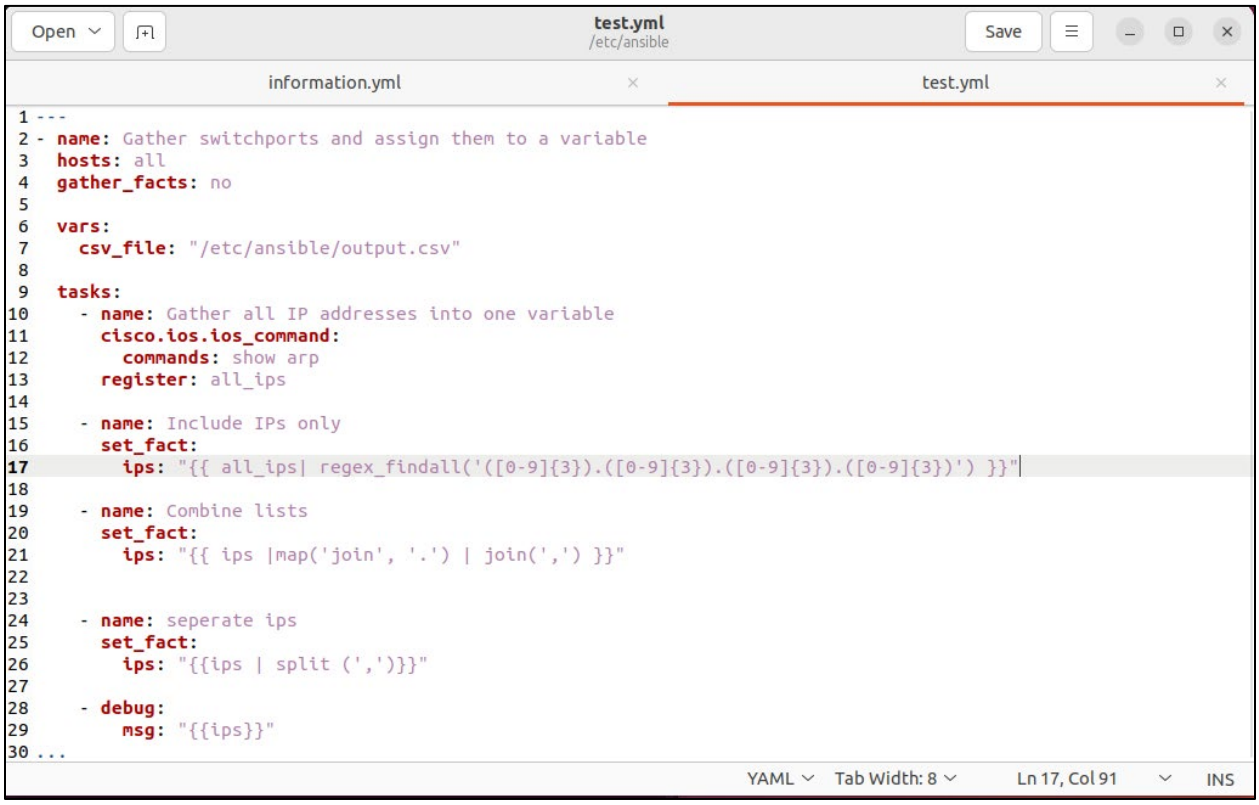


Figure 3.2.5.1 – Sample playbook for gathering and filtering IP addresses.

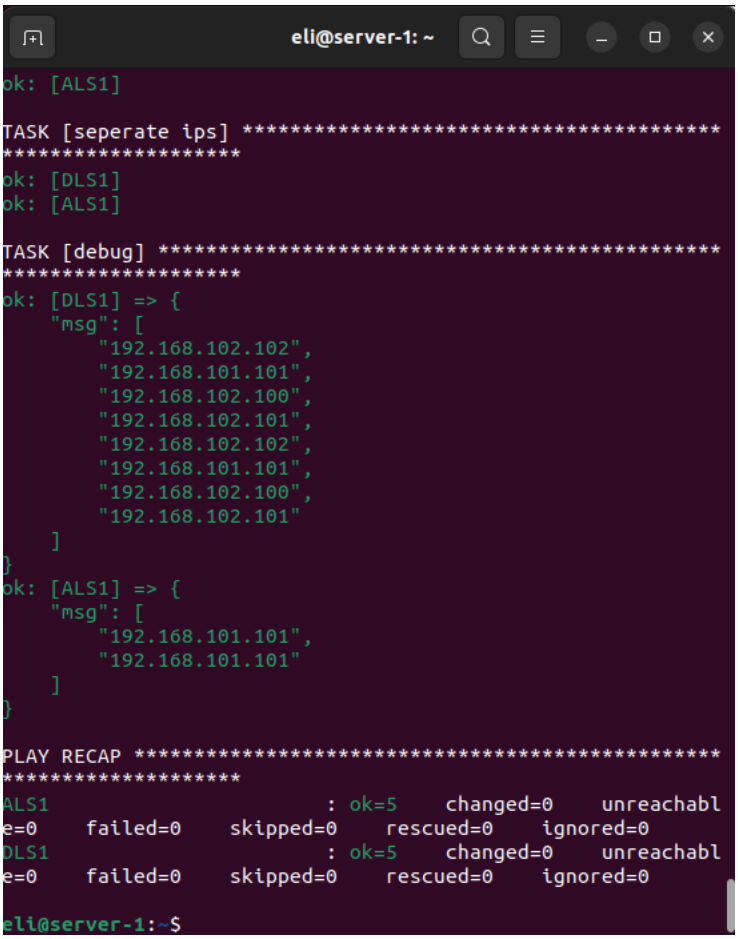


Figure 3.2.5.2 – Output of IP address playbook.

The IP address task was added to information.yml.

Creating the Database

Exporting data to a CSV using Ansible requires the use of a Jinja2 template. Jinja2 is a templating engine that can be used by Ansible to dynamically export variables to external documents (*Templating (Jinja2)*, 2023). A Jinja2 template can be referenced using the template module in Ansible. In the example below, the file information_template.j2 is being referenced as the template file. The resulting output will be sent to a file which uses the hostname variable specified earlier in the playbook to define file names for each switch based on its hostname.

```
- name: Export to CSV

    template:

      src: /etc/ansible/information_template.j2

      dest: /etc/ansible/information"{{hostnames}}".csv
```

Figure 3.2.6.1 – Referencing a Jinja2 template in Ansible playbook.

This template file was then created using the Ubuntu terminal.

```
eli@server-1:~$ touch /etc/ansible/information_template.j2
```

Figure 3.2.6.2 – Creating a new Jinja2 template.

The template file was opened in the default Ubuntu text editor using the GUI. Jinja 2 uses formatting that is like Python and achieves its outputs by passing variables through loops. Jinja2 template variables must be present in the same playbook from which the template has been called for it to work properly. Ansible's Jinja2 template module can only use variables which have been specified within the task itself, therefore, the variables used in the template must be linked to the original variables created when gathering data. In the figure below, two variables have been created, column1 and column2, with values of Switch and Switchport, respectively. These variables are statically set to their string values and can be called in a Jinja2 template. Variables have also been created for hostname and switchport. These variables are dynamic and simply reference the variables created previously from outputs originating from the switches.

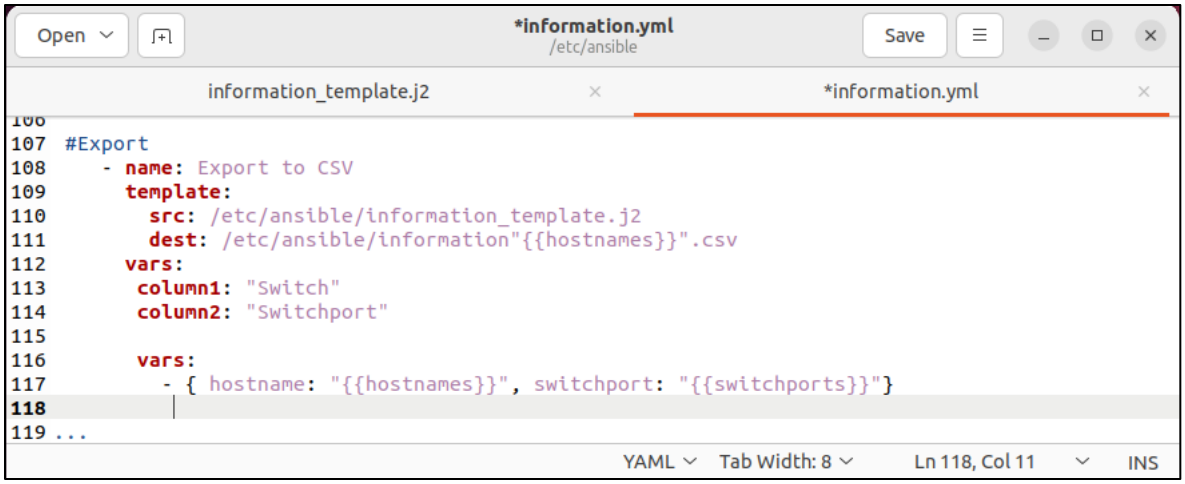


Figure 3.2.6.3 – Ansible task for exporting hostname and switchport.

A Jinja2 template was created to put these variables together and create a simple spreadsheet. The template used CSV formatting to create a loop which printed both the hostname variable and each switchport element for each element in the switchports list.

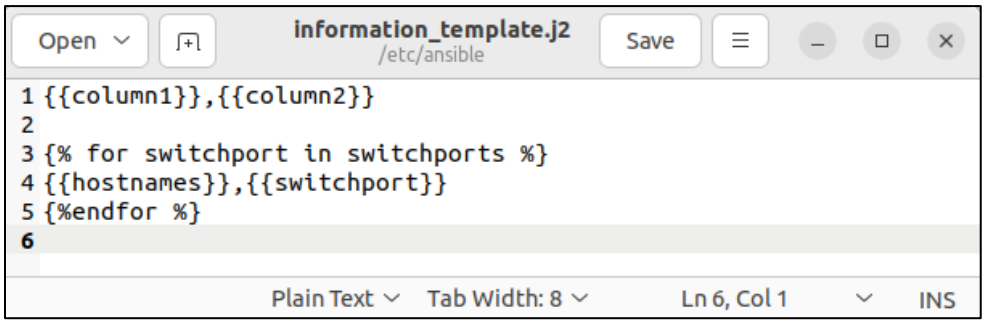


Figure 3.2.6.4 – Jinja2 template for hostnames and switchports.

Two files were generated after running information.yml, one for each switch that the playbook ran on, DLS1 and ALS1.The outputs included columns of switch names and switchports, as well as the column headers defined above.

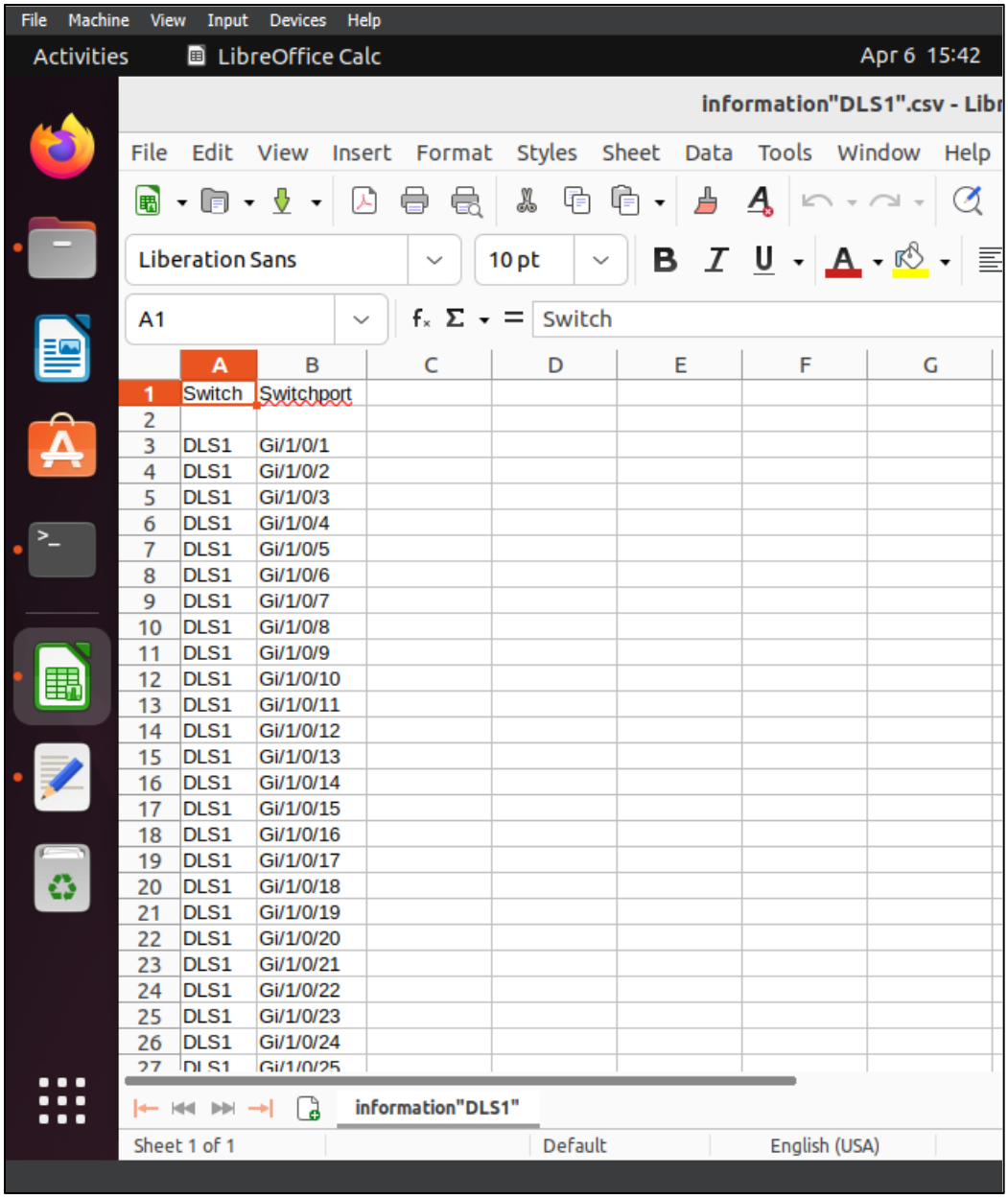


Figure 3.2.6.5 – DLS1 information file created after running information.yml

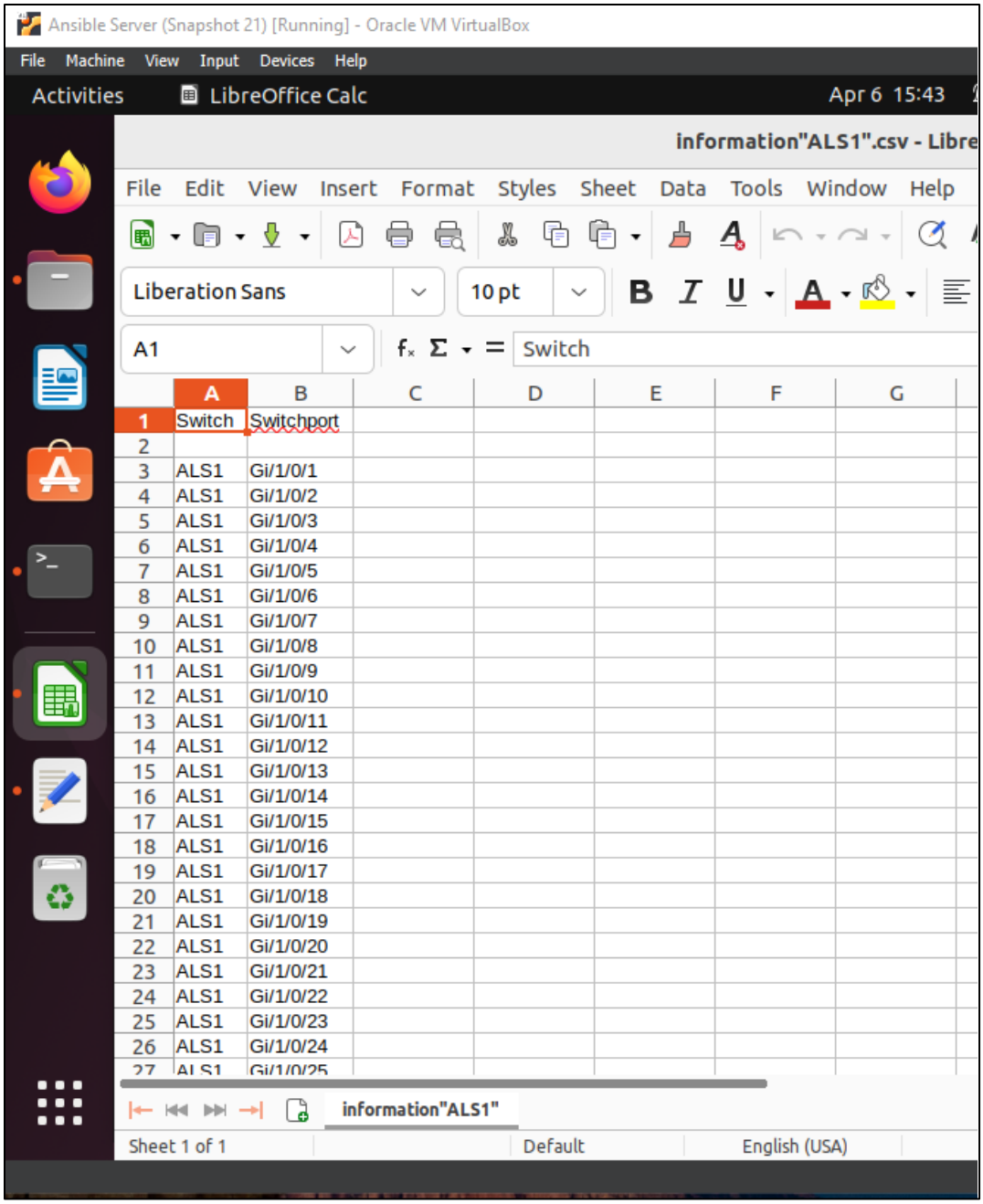
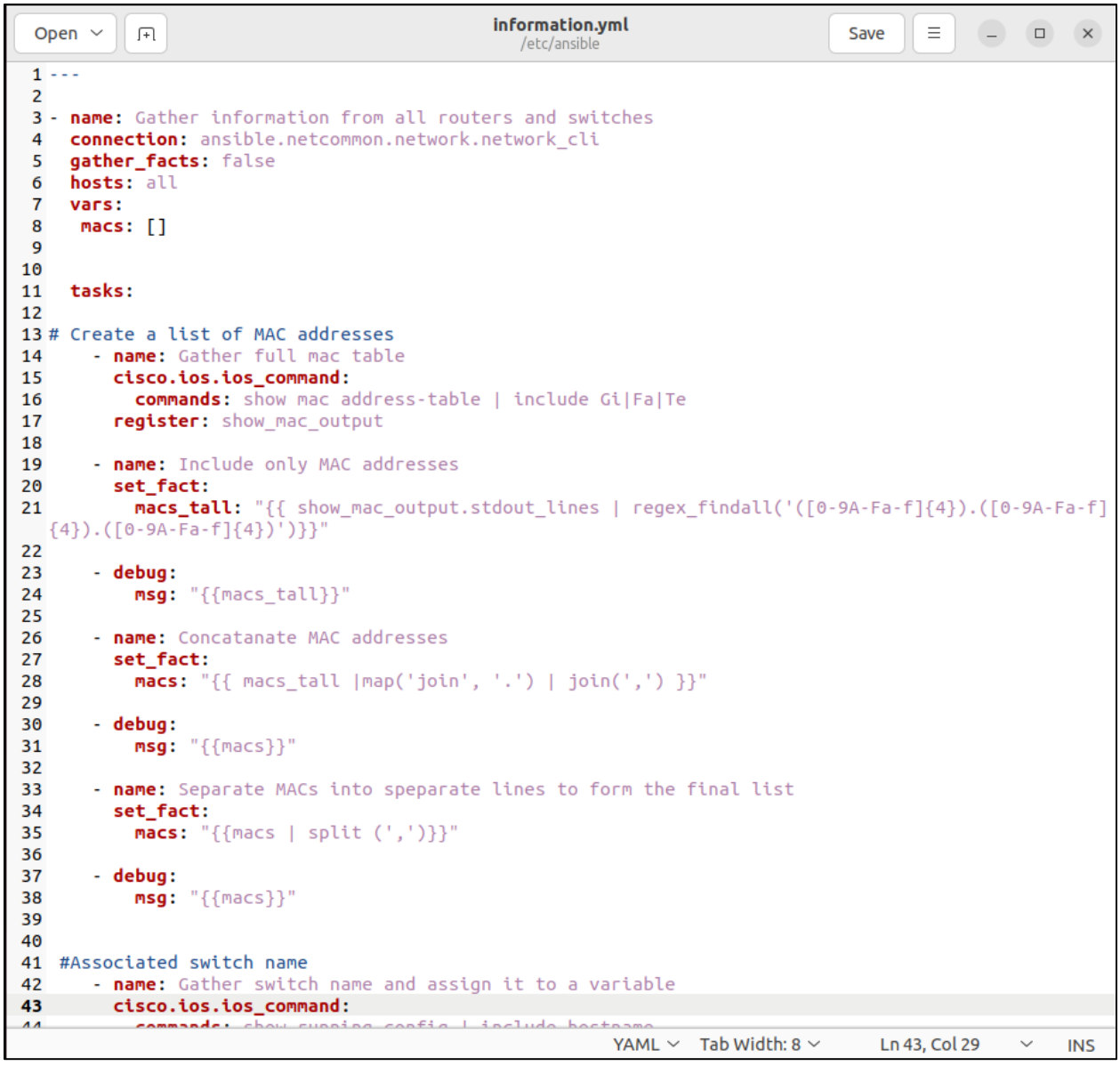


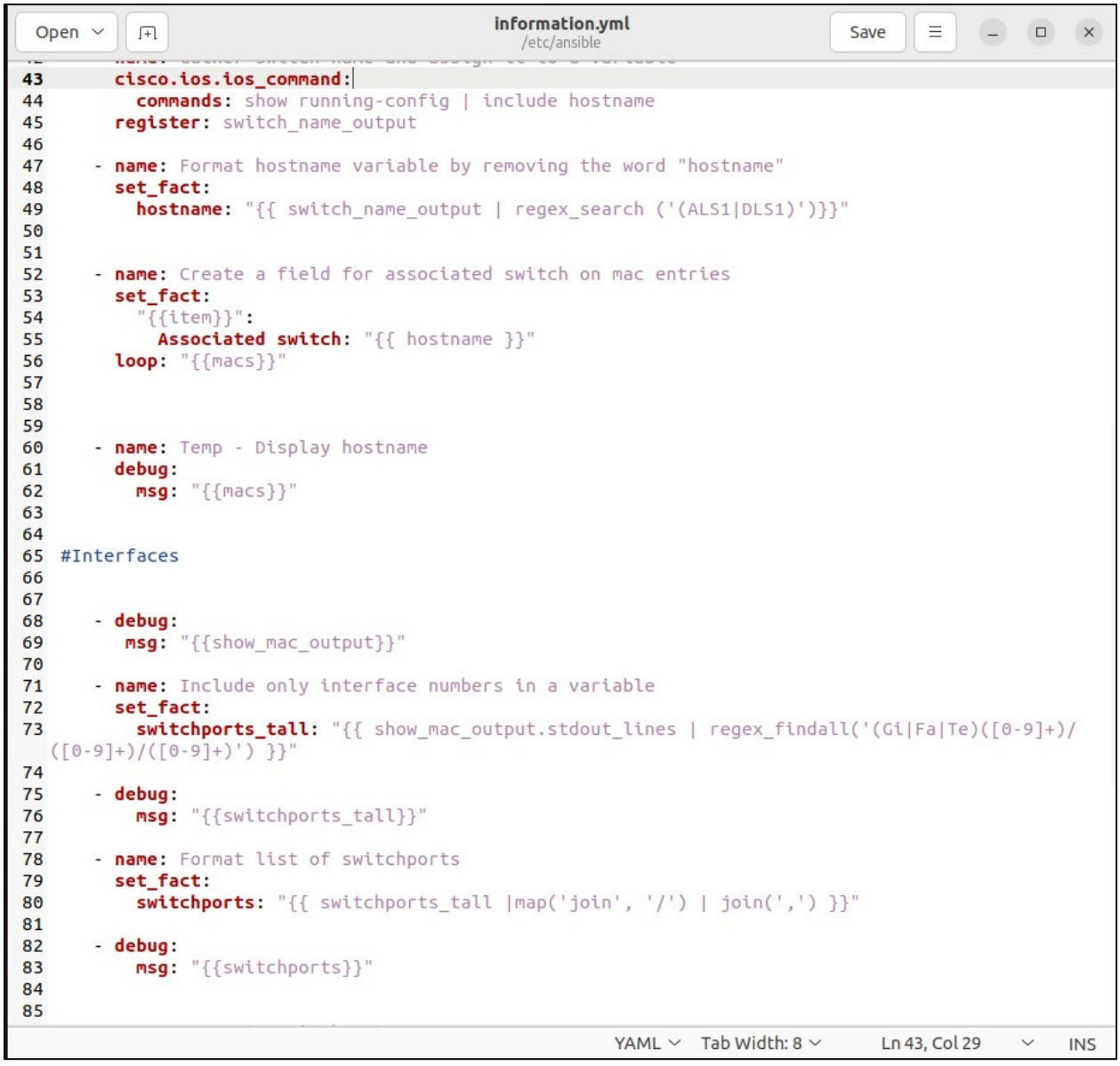
Figure 3.2.6.6 – ALS1 information file created after running information.yml.

To integrate MAC addresses with the spreadsheet, information.yml was modified.




```yaml
---

- name: Gather information from all routers and switches
  connection: ansible.netcommon.network.network_cli
  gather_facts: false
  hosts: all
  vars:
   macs: []


  tasks:

# Create a list of MAC addresses
    - name: Gather full mac table
      cisco.ios.ios_command:
        commands: show mac address-table | include Gi|Fa|Te
      register: show_mac_output

    - name: Include only MAC addresses
      set_fact:
        macs_tall: "{{ show_mac_output.stdout_lines | regex_findall('([0-9A-Fa-f]{4}).([0-9A-Fa-f]{4}).([0-9A-Fa-f]{4})')}}"

    - debug:
        msg: "{{macs_tall}}"

    - name: Concatanate MAC addresses
      set_fact:
        macs: "{{ macs_tall |map('join', '.') | join(',') }}"

    - debug:
        msg: "{{macs}}"

    - name: Separate MACs into speparate lines to form the final list
      set_fact:
        macs: "{{macs | split (',')}}"

    - debug:
        msg: "{{macs}}"


 #Associated switch name
    - name: Gather switch name and assign it to a variable
      cisco.ios.ios_command:
```

Figure 3.2.6.7 – Final information.yml configuration (1/3).




```
43      cisco.ios.ios_command:
44        commands: show running-config | include hostname
45      register: switch_name_output
46
47    - name: Format hostname variable by removing the word "hostname"
48      set_fact:
49        hostname: "{{ switch_name_output | regex_search ('(ALS1|DLS1)')}}"
50
51
52    - name: Create a field for associated switch on mac entries
53      set_fact:
54        "{{item}}":
55          Associated switch: "{{ hostname }}"
56      loop: "{{macs}}"
57
58
59
60    - name: Temp - Display hostname
61      debug:
62        msg: "{{macs}}"
63
64
65  #Interfaces
66
67
68    - debug:
69        msg: "{{show_mac_output}}"
70
71    - name: Include only interface numbers in a variable
72      set_fact:
73        switchports_tall: "{{ show_mac_output.stdout_lines | regex_findall('(Gi|Fa|Te)([0-9]+)/
   ([0-9]+)/([0-9]+)') }}"
74
75    - debug:
76        msg: "{{switchports_tall}}"
77
78    - name: Format list of switchports
79      set_fact:
80        switchports: "{{ switchports_tall |map('join', '/') | join(',') }}"
81
82    - debug:
83        msg: "{{switchports}}"
84
85
```

Figure 3.2.6.8 – Final information.yml configuration (2/3).

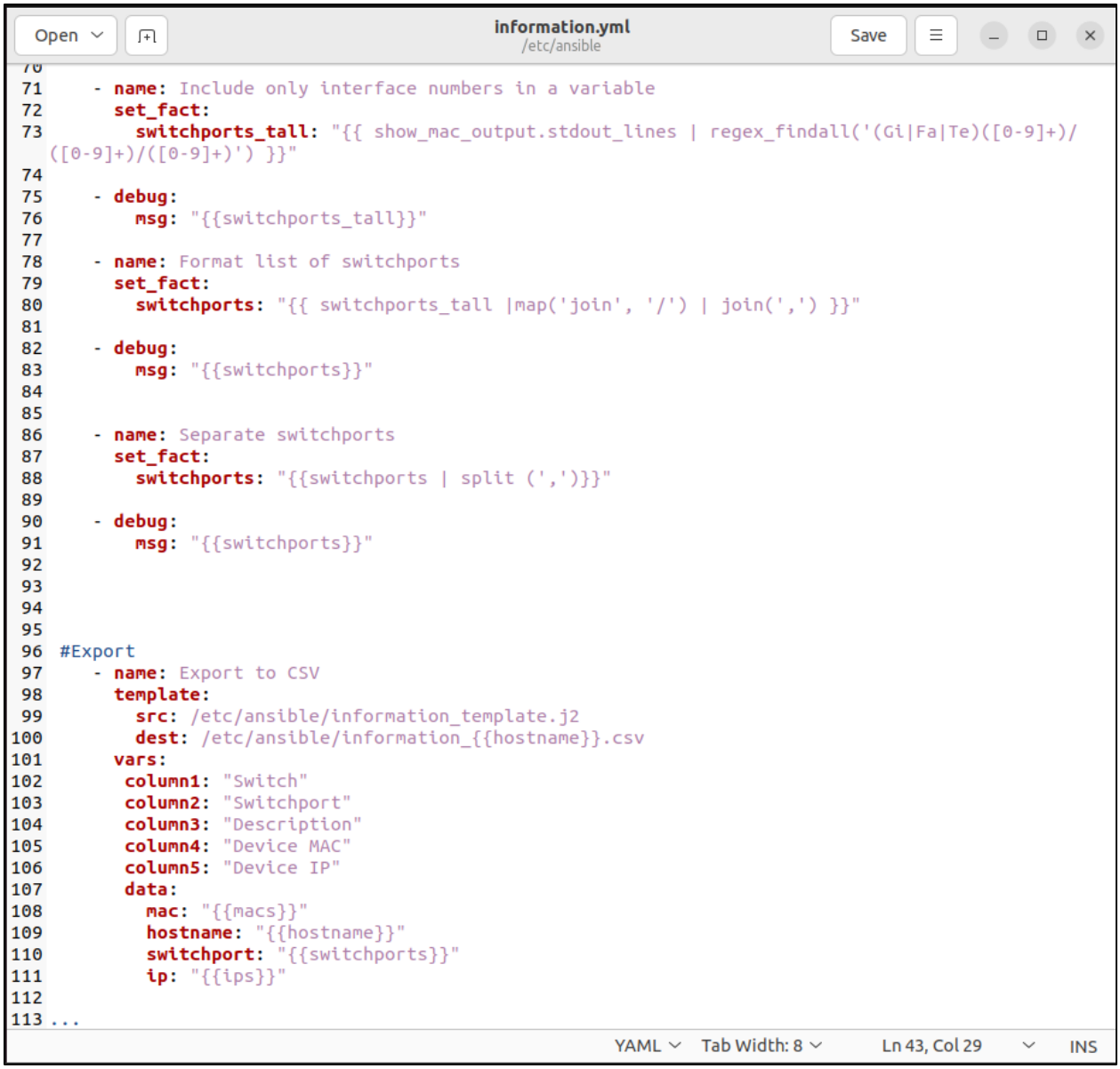


Figure 3.2.6.9 – Final information.yml configuration (3/3).

Incorporating variables such as MAC address and IP address into this spreadsheet significantly increased the complexity of both the Jinja2 template and the playbook itself. As seen in the second task, the process for gathering switch names was altered to contain a loop. For each element in the list of MAC addresses, this loop intended to create a new list inside of it containing the hostname associated with that MAC address. This, in theory would have created a

list out of each MAC address that could then be used to store information about each device. When the playbook was run, however, the following error occurred:

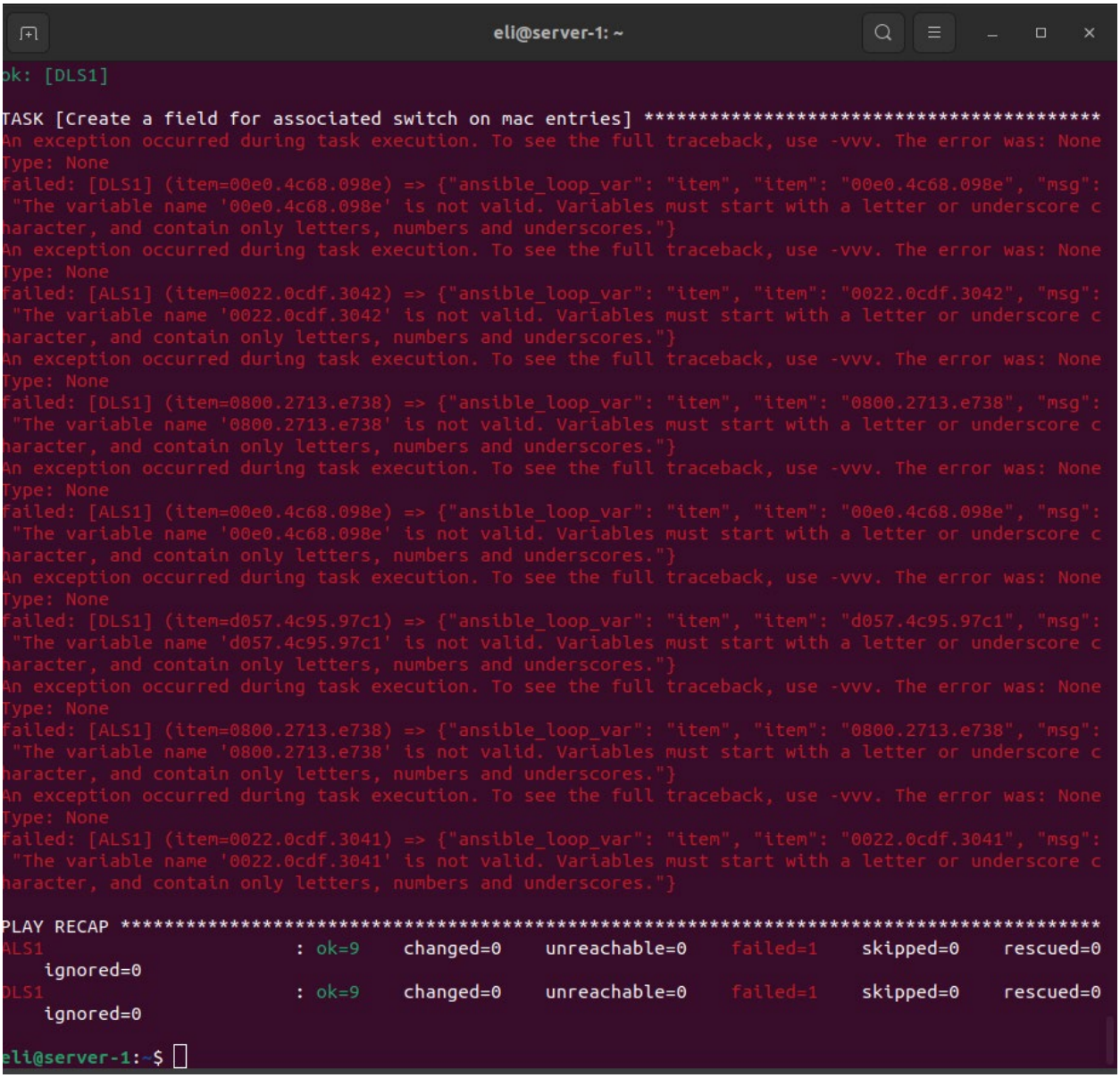


Figure 3.2.6.10 – The variable name is not valid.

Variables which start with a number, rather than a letter or underscore, cannot be used in an Ansible playbook. Without being able to natively call numbered variables, such as MAC

addresses or IP addresses inside an Ansible playbook, running loops over each MAC address becomes impossible.

Part 4: Analysis

Data Structures

Gathering and storing attributes about network devices in an organized manner requires thorough planning as to how the data being collected should be stored and how it will flow throughout the systems being used. For this procedure, pieces of data were gathered into lists containing just one of four types of data: MAC, IP, switchport, or switch hostname. This method of data gathering initially made the most sense because it meant a single show command could be issued on each switch and filtered by the server to produce the intended formatting. Gathering device info into separate lists presents some issues, however. When individual lists contain all MACs or all IPs, they must then be associated together later. This creates an issue in which all the information has been obtained but cannot be displayed in an organized way.

A better solution to gathering and storing attributes about network devices is to store the information collected in separate lists for each device. This allows each device attribute to be associated with a common variable and can be easily called in a loop. This data structure, in theory, would have looked something like this:

```
vars:
        macs: [
                0022.0cdf.3043:[
                        ip:
                        switch:
                        switchport:
                                ]
                b827.eb4f.0cdf:[
                        ip:
                        switch:
                        switchport:
                                ]
                0022.0cdf.3042:[
                        ip:
                        switch:
                        switchport:
                                ]
                00e0.4c68.098e:[
                        ip:
                        switch
                        switchport:
                0800.2713.e738:[
                        ip:
                        switch
                        switchport:
                                 ]
                               ]
```

Figure 4.1.1 – Intended data structure.

The structure above could not be implemented into Ansible because variables cannot begin with numbers, meaning MAC addresses could not be used to further nest device attributes.

Challenges

One challenge that presented itself during the procedure was effective organization of the data being collected. After device attributes such as MAC addresses and IP addresses were collected into lists, it was found that Ansible has no easy way of arranging the data into complex arrays consisting of multiple categories. When gathering information of any kind at enterprise scale, it is crucial that data be stored in a thoroughly organized way. Defining directory structures early on makes it easier to store facts in an organized way and not overpopulate folders when numerous playbooks and inventory files are being used. Not only does this relate to the organization of directories, it also applies to how playbooks are organized. YAML files should be written such that data is not being stored in more location than one unnecessarily and that the least amount of processing is being performed. Although less relevant in the procedures demonstrated, this aspect of data collection and processing is critical at an enterprise level because even the slightest decrease in efficiency can drastically alter the success of automated processes.

Using Ansible to collect information about networking devices presented several other key challenges as well. For this testing environment specifically, the network switches used were older models that are now considered legacy. While this helped shape the perspective of the environment to be inclusive of a variety of devices, it also meant that newer commands used in the Cisco IOS fact gathering module could not be used. Using this module would have made it easier to gather Layer 2 information from switchports and Layer 3 information from the ARP table without having to manually manipulate the outputs of show commands. The old age of the

switches also presented issues when attempting to use SSH to connect to the switch from the Ansible. The key exchange method used by both switches was not enabled by default on the server because they use an outdated encryption method that is disabled by default in Ubuntu 22.04. This issue was resolved by manually adding the encryption method to the list of enabled algorithms on the server's SSH config file. Using outdated encryption methods to pass device data that could potentially leave devices vulnerable is bad security in a production environment and is another reason to refresh backend hardware on a regular basis.

A final roadblock in the procedure prevented Ansible from being able to produce a sizeable portion of the originally intended output. It was discovered during the procedure that Ansible is unable to call numbered variables. This prevented the lists of MAC addresses from being associated with any other attributes and therefore prevented device information from becoming organized. Without properly organizing the data collected, it became unnecessary to print the lists of data to the CSV file. According to Red Hat, Ansible is "not recommended as a data processing/manipulation tool" (Red Hat, 2023). Dictionaries are limited to 1:1 variable association and large arrays are not supported like they are in traditional programming languages. Using Ansible alone to manipulate information about network devices and export it to a CSV file is not a practical solution for locating devices on a network.

Alternative Solutions

While Ansible alone may not be designed to process large amounts of data, it can likely be used in conjunction with other products to produce the intended output. One avenue that should be explored in this topic is the use of a dedicated database. Ansible proved to be successful in gathering device information into variable lists; if each list can be exported to separate files in an organized way, it may be possible to import the data to a dedicated database

system and process it in a more effective fashion. Another potential solution to the problems found in this procedure may be the replacement of Ansible with a fully-fledged programming language. While Ansible is great for its scalability and simplicity, the limitations of the YAML format discovered in this report may suggest that running Python scripts on newer network hardware and returning the results to a common script could make the manipulation of data easier.

Automation Design

The playbook created in this procedure was made with simplicity in mind. Gathering information about devices across an enterprise network is no small undertaking so it was important that the process be simplified where possible. The two main catalysts used to create a simple program were reducing the number of lines it took to achieve goals and reducing the number of files needed to achieve the original goal. For a lab exercise, this methodology worked because only two nodes were being managed and the original design called for Ansible being responsible for both data collection and processing. In a production environment, however, this approach should be altered. The complexity of the automation program being created cannot be forcibly simplified and is entirely dependent upon the task that is presented. As playbooks become more complex, it becomes increasingly helpful to divide tasks and data libraries into smaller files that are called into one another. As previously stated, it is incredibly important that data flows be structured prior to automation programs being created. Developing a sound data flow will be foundational to the organization of automated processes and will create a structure for playbooks that is conducive to large scale network environments. Understanding how core technologies such as Ansible's variable system or an external databases file structure works will be key to developing efficient data storage mechanisms. The simplicity of a playbook does not

necessarily depend on the number of lines or number of files it takes to run, rather, the efficient flow of information from one point to another.

Conclusion

This procedure has demonstrated that Ansible alone is incapable of fully producing a comprehensive information database containing Layer 2 and Layer 3 device information gathered from managed network switches. Ansible does not support the use of numbered values as variables, therefore preventing attributes such as IP address, switchport, and switch hostnames from being properly mapped to a list of MAC addresses. Alternative solutions to this problem have been presented, such as the use of database software or a more robust programming language, both of which should be further explored.

Appendix

Appendix A: Final Ansible Playbook

```
- name: Gather information from all routers and switches

  connection: ansible.netcommon.network.network_cli

  gather_facts: false

  hosts: all



  tasks:



# Create a list of MAC addresses

    - name: Gather full mac table

      cisco.ios.ios_command:

        commands: show mac address-table | include Gi|Fa|Te

      register: show_mac_output



    - name: Include only MAC addresses

      set_fact:
```

```
    macs_tall: "{{ show_mac_output.stdout_lines | regex_findall('([0-9A-Fa-f]{4}).([0-9A-Fa-f]{4}).([0-9A-Fa-f]{4})')}}"


  - debug:
      msg: "{{macs_tall}}"


  - name: Concatenate MAC addresses
    set_fact:
      macs: "{{ macs_tall |map('join', '.') | join(',') }}"


  - debug:
      msg: "{{macs}}"


  - name: Separate MACs into speparate lines to form the final list
    set_fact:
      macs: "{{macs | split (',')}}"


  - debug:
      msg: "{{macs}}"
```

```
#Associated switch name
  - name: Gather switch name and assign it to a variable
    cisco.ios.ios_command:
      commands: show running-config | include hostname
    register: switch_name_output

  - name: Format hostname variable by removing the word "hostname"
    set_fact:
      hostname: "{{ switch_name_output | regex_search ('(ALS1|DLS1)')}}"

  - name: Create a field for associated switch on mac entries
    set_fact:
      "{{item}}":
        Associated switch: "{{ hostname }}"
    loop: "{{macs}}"

  - name: Temp - Display hostname
```

```
    debug:
      msg: "{{macs}}"


#Interfaces
  - debug:
      msg: "{{show_mac_output}}"


  - name: Include only interface numbers in a variable
    set_fact:
      switchports_tall: "{{ show_mac_output.stdout_lines | regex_findall('(Gi|Fa|Te)([0-9]+)/([0-9]+)/([0-9]+)') }}"


  - debug:
      msg: "{{switchports_tall}}"


  - name: Format list of switchports
    set_fact:
      switchports: "{{ switchports_tall |map('join', '/') | join(',') }}"
```

```
  - debug:
      msg: "{{switchports}}"

  - name: Separate switchports
    set_fact:
      switchports: "{{switchports | split (',')}}"

  - debug:
      msg: "{{switchports}}"

#Export
  - name: Export to CSV
    template:
      src: /etc/ansible/information_template.j2
      dest: /etc/ansible/information_{{hostname}}.csv
    vars:
     column1: "Switch"
     column2: "Switchport"
```

```
    column3: "Description"
    column4: "Device MAC"
    column5: "Device IP"
    data:
     mac: "{{macs}}"
     hostname: "{{hostname}}"
     switchport: "{{switchports}}"
     ip: "{{ips}}"
      ...
```

Appendix B: DLS-2 Configuration

```
DLS1#show run
Building configuration...

Current configuration : 12636 bytes
!
version 12.2
no service pad
service timestamps debug uptime
service timestamps log uptime
service password-encryption
!
hostname DLS1
!
```

```
boot-start-marker
boot-end-marker
!
enable secret 5 $1$sSNJ$YcRuKJQa32HWI/OqKop6J/
!
username eli privilege 15 secret 5 $1$WSeH$uPSA5ytWdeQQlBGo03ejO0
username ansible privilege 15 secret 5 $1$xice$8O3VtAcemIeTTdp2kbzL20
!
!
no aaa new-model
clock timezone ETC -5
clock summer-time EST recurring
switch 1 provision ws-c3750g-48ps
system mtu routing 1500
ip routing
no ip domain-lookup
ip domain-name uakron.edu
ip dhcp excluded-address 192.168.102.1 192.168.102.99
!
ip dhcp pool ACCESS_VLAN
   network 192.168.102.0 255.255.255.0
   default-router 192.168.102.1
!
!
!
!
crypto pki trustpoint TP-self-signed-215953408
 enrollment selfsigned
 subject-name cn=IOS-Self-Signed-Certificate-215953408
 revocation-check none
```

```
 rsakeypair TP-self-signed-215953408
!
!
crypto pki certificate chain TP-self-signed-215953408
 certificate self-signed 01
  30820245 308201AE A0030201 02020101 300D0609 2A864886 F70D0101 04050030
  30312E30 2C060355 04031325 494F532D 53656C66 2D536967 6E65642D 43657274
  69666963 6174652D 32313539 35333430 38301E17 0D393330 33303130 30303433
  305A170D 32303031 30313030 30303030 5A303031 2E302C06 03550403 1325494F
  532D5365 6C662D53 69676E65 642D4365 72746966 69636174 652D3231 35393533
  34303830 819F300D 06092A86 4886F70D 01010105 0003818D 00308189 02818100
  AC7947D3 DEE42338 3EE67AB8 6DDC90FF 1ED9AB36 9983DF3D DD956ADA
F4114DAB
  97A90C3E 4FE627C3 36A30354 4C104B9D 8D28A389 1B1D8350 7CA63A48 E34A192A
  95F4E27A AC5C3AD6 579565BB D8602A31 6EF3D3EC E21C76BC 4DD6CF04 3587F834
  E1A85E32 CD179026 EA0C7E48 9205913C BF1E51E9 8E9E582C BD09EC27 E99AB2B9
  02030100 01A36F30 6D300F06 03551D13 0101FF04 05300301 01FF301A 0603551D
  11041330 11820F44 4C53312E 75616B72 6F6E2E65 6475301F 0603551D 23041830
  1680149C 117C659C 1283D4BE EC011DB2 30E44C3D 6548BC30 1D060355 1D0E0416
  04149C11 7C659C12 83D4BEEC 011DB230 E44C3D65 48BC300D 06092A86 4886F70D
  01010405 00038181 007F1A2E 2AEA0652 D6E5A68E C47F0F68 9CD66116 B0D259F4
  08972B1C FB4CBEE3 7313FD36 2F2844D1 54290574 7CBF8389 04DF8BC7 45E9CE77
  EF06A189 4C22D189 E6D4E525 0883CACD 093D7939 FE501BDF 59383674 7D0CF21F
  19D9FD81 290222A1 9002B9A8 A24DE801 D03D3322 CE719097 BB125EDA EC3B836B
  64910EAB 54387928 D8
  quit
!
!
!
!
```

```
spanning-tree mode rapid-pvst
spanning-tree extend system-id
!
vlan internal allocation policy ascending
!
ip ftp source-interface Vlan1000
ip ftp username eli
ip ftp password 7 135445415B5F5C7D
ip ssh version 2
!
!
interface Loopback1
 ip address 172.16.1.1 255.255.255.0
!
interface Loopback2
 ip address 172.16.2.1 255.255.255.0
 ip ospf network point-to-point
!
interface Loopback3
 ip address 172.16.3.1 255.255.255.0
 ip ospf network point-to-point
!
interface Loopback4
 ip address 172.16.4.1 255.255.255.0
 ip ospf network point-to-point
!
interface GigabitEthernet1/0/1
 description Link to ALS-1
 switchport access vlan 999
 switchport trunk encapsulation dot1q
```

```
 switchport trunk allowed vlan 101,102,1000
 switchport mode trunk
 spanning-tree portfast
!
interface GigabitEthernet1/0/2
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/3
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/4
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/5
 description UNUSED
```

```
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/6
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/7
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/8
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
```

```
interface GigabitEthernet1/0/9
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/10
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/11
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/12
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
```

```
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/13
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/14
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/15
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/16
 description UNUSED
 switchport access vlan 999
 switchport mode access
```

```
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/17
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/18
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/19
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/20
 description UNUSED
```

```
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/21
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/22
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/23
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
```

```
interface GigabitEthernet1/0/24
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/25
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/26
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/27
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
```

```
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/28
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/29
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/30
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/31
 description UNUSED
 switchport access vlan 999
 switchport mode access
```

```
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/32
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/33
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/34
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/35
 description UNUSED
```

```
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/36
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/37
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/38
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
```

```
interface GigabitEthernet1/0/39
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/40
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/41
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/42
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
```

```
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/43
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/44
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/45
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/46
 description UNUSED
 switchport access vlan 999
 switchport mode access
```

```
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/47
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/48
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/49
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/50
 description UNUSED
```

```
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/51
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/52
 description UNUSED
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface Vlan1
 no ip address
 shutdown
!
interface Vlan101
 ip address 192.168.101.1 255.255.255.0
!
interface Vlan102
```

```
 ip address 192.168.102.1 255.255.255.0
!
interface Vlan1000
 ip address 192.168.0.1 255.255.255.0
!
router ospf 1985
 log-adjacency-changes
 network 172.16.0.0 0.0.7.255 area 0
 default-information originate
!
ip classless
ip route 0.0.0.0 0.0.0.0 Loopback1
ip http server
ip http secure-server
!
!
vstack
banner motd ^C
UNAUTHORIZED ACCESS STRICTLY PROHIBITED.
AUTHORIZED USERS MUST READ NETWORK MANAGEMENT USER GUIDE PRIOR
TO MAKING CHANGES.
ALL ACTIVITY IS LOGGED.
^C
!
line con 0
 exec-timeout 3 30
 login local
line vty 0 4
 login local
 transport input ssh
```

```
line vty 5 15
 login local
 transport input ssh
!
End


DLS1#show ip int brief
Interface              IP-Address      OK? Method Status                Protocol
Vlan1                  unassigned      YES NVRAM  administratively down down
Vlan101                192.168.101.1   YES NVRAM  up                    up
Vlan102                192.168.102.1   YES NVRAM  up                    up
Vlan1000               192.168.0.1     YES NVRAM  up                    up
GigabitEthernet1/0/1   unassigned      YES unset  up                    up
GigabitEthernet1/0/2   unassigned      YES unset  administratively down down
GigabitEthernet1/0/3   unassigned      YES unset  administratively down down
GigabitEthernet1/0/4   unassigned      YES unset  administratively down down
GigabitEthernet1/0/5   unassigned      YES unset  administratively down down
GigabitEthernet1/0/6   unassigned      YES unset  administratively down down
GigabitEthernet1/0/7   unassigned      YES unset  administratively down down
GigabitEthernet1/0/8   unassigned      YES unset  administratively down down
GigabitEthernet1/0/9   unassigned      YES unset  administratively down down
GigabitEthernet1/0/10  unassigned      YES unset  administratively down down
GigabitEthernet1/0/11  unassigned      YES unset  administratively down down
GigabitEthernet1/0/12  unassigned      YES unset  administratively down down
GigabitEthernet1/0/13  unassigned      YES unset  administratively down down
GigabitEthernet1/0/14  unassigned      YES unset  administratively down down
GigabitEthernet1/0/15  unassigned      YES unset  administratively down down
GigabitEthernet1/0/16  unassigned      YES unset  administratively down down
GigabitEthernet1/0/17  unassigned      YES unset  administratively down down
GigabitEthernet1/0/18  unassigned      YES unset  administratively down down
```

```
GigabitEthernet1/0/19  unassigned      YES unset  administratively down down
GigabitEthernet1/0/20  unassigned      YES unset  administratively down down
GigabitEthernet1/0/21  unassigned      YES unset  administratively down down
GigabitEthernet1/0/22  unassigned      YES unset  administratively down down
GigabitEthernet1/0/23  unassigned      YES unset  administratively down down
GigabitEthernet1/0/24  unassigned      YES unset  administratively down down
GigabitEthernet1/0/25  unassigned      YES unset  administratively down down
GigabitEthernet1/0/26  unassigned      YES unset  administratively down down
GigabitEthernet1/0/27  unassigned      YES unset  administratively down down
GigabitEthernet1/0/28  unassigned      YES unset  administratively down down
GigabitEthernet1/0/29  unassigned      YES unset  administratively down down
GigabitEthernet1/0/30  unassigned      YES unset  administratively down down
GigabitEthernet1/0/31  unassigned      YES unset  administratively down down
GigabitEthernet1/0/32  unassigned      YES unset  administratively down down
GigabitEthernet1/0/33  unassigned      YES unset  administratively down down
GigabitEthernet1/0/34  unassigned      YES unset  administratively down down
GigabitEthernet1/0/35  unassigned      YES unset  administratively down down
GigabitEthernet1/0/36  unassigned      YES unset  administratively down down
GigabitEthernet1/0/37  unassigned      YES unset  administratively down down
GigabitEthernet1/0/38  unassigned      YES unset  administratively down down
GigabitEthernet1/0/39  unassigned      YES unset  administratively down down
GigabitEthernet1/0/40  unassigned      YES unset  administratively down down
GigabitEthernet1/0/41  unassigned      YES unset  administratively down down
GigabitEthernet1/0/42  unassigned      YES unset  administratively down down
GigabitEthernet1/0/43  unassigned      YES unset  administratively down down
GigabitEthernet1/0/44  unassigned      YES unset  administratively down down
GigabitEthernet1/0/45  unassigned      YES unset  administratively down down
GigabitEthernet1/0/46  unassigned      YES unset  administratively down down
GigabitEthernet1/0/47  unassigned      YES unset  administratively down down
GigabitEthernet1/0/48  unassigned      YES unset  administratively down down
```

```
GigabitEthernet1/0/49  unassigned      YES unset  administratively down down
GigabitEthernet1/0/50  unassigned      YES unset  administratively down down
GigabitEthernet1/0/51  unassigned      YES unset  administratively down down
GigabitEthernet1/0/52  unassigned      YES unset  administratively down down
Loopback1              172.16.1.1      YES NVRAM  up                    up
Loopback2              172.16.2.1      YES NVRAM  up                    up
Loopback3              172.16.3.1      YES NVRAM  up                    up
Loopback4              172.16.4.1      YES NVRAM  up                    up


DLS1#show ip route | begin Gateway
Gateway of last resort is 0.0.0.0 to network 0.0.0.0

     172.16.0.0/24 is subnetted, 4 subnets
C       172.16.4.0 is directly connected, Loopback4
C       172.16.1.0 is directly connected, Loopback1
C       172.16.2.0 is directly connected, Loopback2
C       172.16.3.0 is directly connected, Loopback3
C    192.168.0.0/24 is directly connected, Vlan1000
C    192.168.102.0/24 is directly connected, Vlan102
C    192.168.101.0/24 is directly connected, Vlan101
S*   0.0.0.0/0 is directly connected, Loopback1

DLS1#show ip ospf
 Routing Process "ospf 1985" with ID 172.16.4.1
 Start time: 16:48:02.232, Time elapsed: 00:07:56.666
 Supports only single TOS(TOS0) routes
 Supports opaque LSA
 Supports Link-local Signaling (LLS)
 Supports area transit capability
```

It is an autonomous system boundary router

Redistributing External Routes from,

Router is not originating router-LSAs with maximum metric

Initial SPF schedule delay 5000 msecs

Minimum hold time between two consecutive SPFs 10000 msecs

Maximum wait time between two consecutive SPFs 10000 msecs

Incremental-SPF disabled

Minimum LSA interval 5 secs

Minimum LSA arrival 1000 msecs

LSA group pacing timer 240 secs

Interface flood pacing timer 33 msecs

Retransmission pacing timer 66 msecs

Number of external LSA 1. Checksum Sum 0x008B9D

Number of opaque AS LSA 0. Checksum Sum 0x000000

Number of DCbitless external and opaque AS LSA 0

Number of DoNotAge external and opaque AS LSA 0

Number of areas in this router is 1. 1 normal 0 stub 0 nssa

Number of areas transit capable is 0

External flood list length 0

IETF NSF helper support enabled

Cisco NSF helper support enabled

Reference bandwidth unit is 100 mbps

Area BACKBONE(0) (Inactive)

Number of interfaces in this area is 4 (4 loopback)

Area has no authentication

SPF algorithm last executed 00:04:23.260 ago

SPF algorithm executed 3 times

Area ranges are

Number of LSA 1. Checksum Sum 0x001564

Number of opaque link LSA 0. Checksum Sum 0x000000

Number of DCbitless LSA 0

Number of indication LSA 0

Number of DoNotAge LSA 0

Flood list length 0

Appendix C: ALS-2 Configuration

ALS1#show run

Building configuration...

Current configuration : 10489 bytes

!

version 12.2

no service pad

service timestamps debug datetime msec

service timestamps log datetime msec

service password-encryption

!

hostname ALS1

!

boot-start-marker

boot-end-marker

!

!

username eli privilege 15 secret 5 1ddJ/$vajA6KZlVPDBIgv9VcKCM.

username ansible privilege 15 secret 5 1W.pP$vZetSgN1WtBBuVxAqv7Qd1

!

!

no aaa new-model

switch 1 provision ws-c3750g-48ps

```
system mtu routing 1500
no ip domain-lookup
!
!
!
!
crypto pki trustpoint TP-self-signed-1284872064
 enrollment selfsigned
 subject-name cn=IOS-Self-Signed-Certificate-1284872064
 revocation-check none
 rsakeypair TP-self-signed-1284872064
!
!
crypto pki certificate chain TP-self-signed-1284872064
 certificate self-signed 01
  3082023D 308201A6 A0030201 02020101 300D0609 2A864886 F70D0101 04050030
  31312F30 2D060355 04031326 494F532D 53656C66 2D536967 6E65642D 43657274
  69666963 6174652D 31323834 38373230 3634301E 170D3933 30333031 30303034
  33325A17 0D323030 31303130 30303030 305A3031 312F302D 06035504 03132649
  4F532D53 656C662D 5369676E 65642D43 65727469 66696361 74652D31 32383438
  37323036 3430819F 300D0609 2A864886 F70D0101 01050003 818D0030 81890281
  8100D785 8937A9D2 1B44E9FA 264E32C8 FE87B45A DA9608B7 BE6D51B4 2AA87EC7
  8AC21A85 4927024B 1B570E7D EF31504D BBEA78AC 63838E77 EDC51DA0 5297E2BF
  BA3E2C91 6B0D60EB 46C73000 8955FAC3 E01B502F D9DBCD58 5FE8A6EE F9343464
  F45EA8DC EE0C0E92 F0EF8EAB 17304764 3A3D48EA 0AF19593 AB7FB8A6 1C9B6583
  C3F30203 010001A3 65306330 0F060355 1D130101 FF040530 030101FF 30100603
  551D1104 09300782 05414C53 312E301F 0603551D 23041830 1680144E A5FC797C
  93614421 90FEB1C2 81BE0669 FFAF2630 1D060355 1D0E0416 04144EA5 FC797C93
  61442190 FEB1C281 BE0669FF AF26300D 06092A86 4886F70D 01010405 00038181
  0026FBFD 8B2EB071 D7628F24 E6B61769 47001DBE D26C6712 2BA0ABC1 A6BBD23B
```

```
  9244EF21 C3FA0F9F 17AA337A 4CCF29D3 E423DCA5 29A72E5D 5888FCE7 C70C9E10
  A5A9BF1D A093AEEB D298E145 AB728BDC 2CD34B13 5CAF52A0 F3025FAC
052D029B
  79536207 51D49343 140CC105 253DC304 E487C217 38F0AA13 34EFF8CE F40A7826 FA
  quit
!
!
!
!
spanning-tree mode rapid-pvst
spanning-tree extend system-id
!
vlan internal allocation policy ascending
!
!
!
!
interface GigabitEthernet1/0/1
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 101,102,1000
 switchport mode trunk
!
interface GigabitEthernet1/0/2
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/3
```

```
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/4
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/5
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/6
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/7
 switchport access vlan 999
 switchport mode access
```

```
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/8
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/9
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/10
 switchport access vlan 101
 switchport mode access
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/11
 switchport access vlan 102
 switchport mode access
 spanning-tree portfast
 spanning-tree bpduguard enable
!
```

```
interface GigabitEthernet1/0/12
 switchport access vlan 102
 switchport mode access
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/13
 switchport access vlan 102
 switchport mode access
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/14
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/15
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/16
 switchport access vlan 999
 switchport mode access
 shutdown
```

```
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/17
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/18
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/19
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/20
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
```

```
!
interface GigabitEthernet1/0/21
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/22
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/23
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/24
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/25
```

```
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/26
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/27
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/28
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/29
 switchport access vlan 999
 switchport mode access
```

```
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/30
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/31
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/32
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/33
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
```

```
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/34
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/35
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/36
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/37
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
```

```
interface GigabitEthernet1/0/38
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/39
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/40
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/41
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/42
 switchport access vlan 999
```

```
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/43
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/44
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/45
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/46
 switchport access vlan 999
 switchport mode access
 shutdown
```

```
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/47
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/48
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/49
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/50
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
```

```
!
interface GigabitEthernet1/0/51
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface GigabitEthernet1/0/52
 switchport access vlan 999
 switchport mode access
 shutdown
 spanning-tree portfast
 spanning-tree bpduguard enable
!
interface Vlan1
 no ip address
 shutdown
!
interface Vlan1000
 ip address 192.168.0.100 255.255.255.0
!
ip classless
ip http server
ip http secure-server
!
!
ip sla enable reaction-alerts
!
!
```

```
banner motd ^C
UNAUTHORIZED ACCESS STRICTLY PROHIBITED.
AUTHORIZED USERS MUST READ NETWORK MANAGEMENT USER GUIDE PRIOR TO MAKING CHANGES
ALL ACTIVITY IS LOGGED.
^C
!
line con 0
 exec-timeout 3 30
 login local
line vty 0 4
 login local
line vty 5 15
 login local
!
end

ALS1#show ip int brief
Interface              IP-Address      OK? Method Status                Protocol
Vlan1                  unassigned      YES NVRAM  administratively down down
Vlan1000               192.168.0.100   YES NVRAM  up                    up
GigabitEthernet1/0/1   unassigned      YES unset  up                    up
GigabitEthernet1/0/2   unassigned      YES unset  administratively down down
GigabitEthernet1/0/3   unassigned      YES unset  administratively down down
GigabitEthernet1/0/4   unassigned      YES unset  administratively down down
GigabitEthernet1/0/5   unassigned      YES unset  administratively down down
GigabitEthernet1/0/6   unassigned      YES unset  administratively down down
GigabitEthernet1/0/7   unassigned      YES unset  administratively down down
GigabitEthernet1/0/8   unassigned      YES unset  administratively down down
GigabitEthernet1/0/9   unassigned      YES unset  administratively down down
```

```
GigabitEthernet1/0/10  unassigned      YES unset  up                    up
GigabitEthernet1/0/11  unassigned      YES unset  down                  down
GigabitEthernet1/0/12  unassigned      YES unset  down                  down
GigabitEthernet1/0/13  unassigned      YES unset  down                  down
GigabitEthernet1/0/14  unassigned      YES unset  administratively down down
GigabitEthernet1/0/15  unassigned      YES unset  administratively down down
GigabitEthernet1/0/16  unassigned      YES unset  administratively down down
GigabitEthernet1/0/17  unassigned      YES unset  administratively down down
GigabitEthernet1/0/18  unassigned      YES unset  administratively down down
GigabitEthernet1/0/19  unassigned      YES unset  administratively down down
GigabitEthernet1/0/20  unassigned      YES unset  administratively down down
GigabitEthernet1/0/21  unassigned      YES unset  administratively down down
GigabitEthernet1/0/22  unassigned      YES unset  administratively down down
GigabitEthernet1/0/23  unassigned      YES unset  administratively down down
GigabitEthernet1/0/24  unassigned      YES unset  administratively down down
GigabitEthernet1/0/25  unassigned      YES unset  administratively down down
GigabitEthernet1/0/26  unassigned      YES unset  administratively down down
GigabitEthernet1/0/27  unassigned      YES unset  administratively down down
GigabitEthernet1/0/28  unassigned      YES unset  administratively down down
GigabitEthernet1/0/29  unassigned      YES unset  administratively down down
GigabitEthernet1/0/30  unassigned      YES unset  administratively down down
GigabitEthernet1/0/31  unassigned      YES unset  administratively down down
GigabitEthernet1/0/32  unassigned      YES unset  administratively down down
GigabitEthernet1/0/33  unassigned      YES unset  administratively down down
GigabitEthernet1/0/34  unassigned      YES unset  administratively down down
GigabitEthernet1/0/35  unassigned      YES unset  administratively down down
GigabitEthernet1/0/36  unassigned      YES unset  administratively down down
GigabitEthernet1/0/37  unassigned      YES unset  administratively down down
GigabitEthernet1/0/38  unassigned      YES unset  administratively down down
GigabitEthernet1/0/39  unassigned      YES unset  administratively down down
```

```
GigabitEthernet1/0/40  unassigned      YES unset  administratively down down
GigabitEthernet1/0/41  unassigned      YES unset  administratively down down
GigabitEthernet1/0/42  unassigned      YES unset  administratively down down
GigabitEthernet1/0/43  unassigned      YES unset  administratively down down
GigabitEthernet1/0/44  unassigned      YES unset  administratively down down
GigabitEthernet1/0/45  unassigned      YES unset  administratively down down
GigabitEthernet1/0/46  unassigned      YES unset  administratively down down
GigabitEthernet1/0/47  unassigned      YES unset  administratively down down
GigabitEthernet1/0/48  unassigned      YES unset  administratively down down
GigabitEthernet1/0/49  unassigned      YES unset  administratively down down
GigabitEthernet1/0/50  unassigned      YES unset  administratively down down
GigabitEthernet1/0/51  unassigned      YES unset  administratively down down
GigabitEthernet1/0/52  unassigned      YES unset  administratively down down
```

Glossary

Diffie-Hellman Group 1 – Legacy key exchange method. Uses fewer bits to encrypt security keys. Used for establishing SSH connection.

DRY (Don't Repeat Yourself) – A principle used in software development which suggests that variables should only be defined in one location to reduce code redundancy.

Inventory File – A list of devices and associated IP addresses used by Ansible to identify which hosts playbooks should be run on.

K9 – Image set tag used by Cisco to indicate support for longer encryption methods and SSH connections.

Layer 2 – The second layer of the OSI model, also known as the data link layer, refers to the physical addressing of devices.

Layer 3 – The third layer of the OSI model, also knows as the network layer, refers to logical addressing of devices.

Managed node – A device which can be manipulated by Ansible.

Module – Library plugins that can be used in Ansible to execute specific tasks.

Playbook – A list of tasks that are run on a remote device using Ansible. Written in the YAML format.

Spanning-tree – A Layer 2 protocol used to prevent network loops.

Virtual machine host – The physical device on which a virtual machine is running.

VM (Virtual Machine) – A virtualized instance of a computer operating system running on top of a virtual machine host.

Yet Another Markup Language (YAML) – Programming method used by Ansible. Similar in style to Python. Uses .yml extension.

References

Dell. (n.d.). *Setting up SSH for Ansible*. Dell Technologies. Retrieved April 11, 2023, from https://infohub.delltechnologies.com/l/dell-poweredge-getting-started-with-redfish-ansible-modules/setting-up-ssh-for-ansible

Global Information Assurance (2003) Security Implications of Advanced Ethernet Switching Technologies. [White paper]. https://www.giac.org/paper/gsec/3010/security-implications-advanced-ethernet-switching-technologies/105036

Juniper Networks. (2021, May 2). Understanding the Ansible inventory file when managing devices running junos OS. Ansible for Junos OS Developer Guide. Retrieved April 11, 2023, from https://www.juniper.net/documentation/us/en/software/junos-ansible/ansible/topics/concept/junos-ansible-inventory-file-overview.html#:~:text=The%20default%20location%20for%20the,user%2Ddefined%20groups%20of%20hosts.

Legacy options. OpenSSH. (n.d.). Retrieved April 11, 2023, from https://www.openssh.com/legacy.html

Meijer, B. (2022). *Ansible: Up and running* (3rd ed.). O'REILLY MEDIA.

Red Hat (2018) The Benefits of Agentless Architecture. [White paper] https://www.ansible.com/hubfs/pdfs/Benefits-of-Agentless-WhitePaper.pdf

Red Hat. (2023, March 30). *Manipulating data*. Manipulating data - Ansible Documentation. Retrieved April 10, 2023, from https://docs.ansible.com/ansible/latest/playbook_guide/complex_data_manipulation.html

Sprygada, P. (2023, March 30). *Cisco.ios.ios_config module – module to manage configuration sections.* cisco.ios.ios_config module – Module to manage configuration sections. - Ansible Documentation. Retrieved April 5, 2023, from https://docs.ansible.com/ansible/latest/collections/cisco/ios/ios_config_module.html