**Lab Guide: Basic Ansible Operations**

**1. Prerequisites**

* Ensure Ansible is installed on your control node (your local machine).
* You should have SSH access to the remote machines (referred to as "hosts").
* The inventory file should be set up with the remote hosts grouped as demo, devices, s1, and r1.

**2. Verify Ansible Installation**

* Run the following command to list all available Ansible modules:

bash

Copy code

ansible-doc -l | grep ping

**3. Managing Users**

**Create a New User**

* Create a user named "raman" on the remote hosts in the demo group:

bash

Copy code

ansible demo -m user -a "name=raman state=present comment='created via ansible cli' group=root shell=/bin/bash"

**Modify a User**

* Modify the comment for the user "raman":

bash

Copy code

ansible demo -m user -a "name=raman state=present comment='created by Raman' group=root shell=/bin/bash"

**Create a User on Network Devices**

* Create a user named "raman" with privilege level 15 on network devices in the devices group:

bash

Copy code

ansible devices -m ios\_user -a "name=raman state=present privilege=15 configured\_password=admin1 password\_type=password"

**Delete a User**

* Remove the user "raman" from the remote hosts in the demo group:

bash

Copy code

ansible demo -m user -a "name=raman state=absent comment='created by Raman' group=root shell=/bin/bash"

**4. Gathering System Information**

**Gather All System Facts**

* Gather all system facts for the hosts in the demo group:

bash

Copy code

ansible demo -m setup

**Filter System Information**

* Gather only specific system information, such as the distribution name or default IPv4 address:

bash

Copy code

ansible demo -m setup -a filter="ansible\_distribution"

ansible demo -m setup -a filter="ansible\_default\_ipv4"

**5. Managing Packages**

**Install a Package**

* Install the telnet package on the remote hosts in the demo group:

bash

Copy code

ansible demo -m apt -a "name=telnet state=installed"

**Ensure a Package is Present**

* Ensure that the telnet package is present:

bash

Copy code

ansible demo -m apt -a "name=telnet state=present"

**Remove a Package**

* Remove the telnet package from the remote hosts:

bash

Copy code

ansible demo -m apt -a "name=telnet state=absent"

**6. Managing Files**

**Create or Modify a File**

* Create or modify a file named /var/tmp/gagandeep with specific permissions and ownership:

bash

Copy code

ansible demo -m file -a "path=/var/tmp/gagandeep mode=777 group=root state=touch"

**Delete a File**

* Delete the file /var/tmp/gagandeep:

bash

Copy code

ansible demo -m file -a "path=/var/tmp/gagandeep mode=777 group=root state=absent"

**7. Executing Commands on Remote Hosts**

**Check System Uptime**

* Run the uptime command on the remote hosts in the demo group:

bash

Copy code

ansible demo -m command -a uptime

**Conclusion**

This guide provided a series of commands to perform basic operations using Ansible, such as user management, package management, file handling, and gathering system information. You can further extend these tasks by creating playbooks and roles for more complex automation workflows.

**Lab Guide: Ansible for Network Automation with Cisco IOS**

**1. Prerequisites**

* Ansible is installed and configured on your control node.
* You have network devices (e.g., Cisco IOS) accessible via SSH.
* The inventory file (/etc/ansible/hosts) is set up with the devices group, listing all your network devices.
* Ensure that the necessary Python packages (e.g., netmiko, paramiko, pynxos) are installed for Ansible's network modules.

**2. Basic Command Execution on Network Devices**

**Run a Show Command**

* Use the ios\_command module to run a command on devices in the devices group:

bash

Copy code

ansible devices -m ios\_command -a "commands='sh run | i user '"

* This command filters the running configuration to display lines containing the word "user."

**Check IP Interface Brief**

* Use the ios\_command module to display a brief summary of all IP interfaces:

bash

Copy code

ansible devices -m ios\_command -a "commands='sh ip int brief'"

**3. Gathering Facts from Network Devices**

**Collect All Device Facts**

* Use the ios\_facts module to gather all available facts from the devices:

bash

Copy code

ansible devices -m ios\_facts

**Gather Specific Facts**

* Gather hardware-related information:

bash

Copy code

ansible devices -m ios\_facts -a "gather\_subset=hardware"

* Gather interface-related information:

bash

Copy code

ansible devices -m ios\_facts -a "gather\_subset=interfaces"

**4. Working with Ansible Playbooks**

**Create a Basic Playbook**

* Create a file named first.yml with the following content:

yaml

Copy code

---

- hosts: demo

tasks:

- name: Install apache2 on server

action: apt name=apache2 state=present

- name: Add a directory

action: file path=/opt state=directory

- name: Install nginx on server

action: apt name=nginx state=present

* This playbook installs Apache and Nginx and creates a directory on the demo group hosts.

**Create a Network Device Playbook**

* Create a file named sec.yml with the following content:

yaml

Copy code

---

- name: Gather and display IOS device facts

hosts: devices

gather\_facts: false

tasks:

- name: Gather IOS device interfaces facts

ios\_facts:

gather\_subset: interfaces

- name: Display the interface IPv4 addresses

debug:

msg: "All IPv4 addresses: {{ ansible\_net\_all\_ipv4\_addresses }}"

* This playbook gathers interface facts and displays all IPv4 addresses for devices in the devices group.

**Running the Playbooks**

* Run the first playbook:

bash

Copy code

ansible-playbook first.yml

* Run the second playbook for network devices:

bash

Copy code

ansible-playbook sec.yml

**5. Advanced Playbook Configuration**

**Modifying the Playbook for Full Configuration**

* Update sec.yml to gather all configuration details and display them:

yaml

Copy code

---

- name: Gather and display full IOS configuration

hosts: devices

gather\_facts: false

tasks:

- name: Gather full IOS facts

ios\_facts:

gather\_subset: all

- name: Display the hostname and OS version

debug:

msg: "The hostname is {{ ansible\_net\_hostname }} and the OS version is {{ ansible\_net\_version }}"

* This updated playbook now gathers all facts and displays the hostname and OS version of the devices.

**Conclusion**

This lab guide provides hands-on experience with Ansible for network automation, focusing on basic command execution, fact gathering, and creating Ansible playbooks for managing Cisco IOS devices. By following these steps, you will develop a strong foundation in using Ansible to automate network device management.

**Lab Guide: Ansible Privilege Escalation**

**1. Prerequisites**

* Ansible is installed and configured on your control node.
* You have SSH access to the remote hosts with multiple users (raman and khanna).
* The remote hosts should have sudo privileges configured for the user khanna.
* An inventory file (/etc/ansible/hosts) is set up with the demo group containing the target hosts.

**2. Understanding the Playbook Structure**

The playbook first.yml provided escalates privileges from the user raman to khanna and performs several tasks on the remote hosts.

yaml

Copy code

---

- hosts: demo

user: raman

become\_user: khanna

become\_method: su

tasks:

- name: Remove apache2 from the server

action: apt name=apache2 state=absent

- name: Create a file in the root directory

action: file path=/root/test state=present

- name: Remove nginx from the server

action: apt name=nginx state=absent

**3. Step-by-Step Guide**

**3.1. Setting Up the Inventory File**

Ensure your inventory file contains the demo group with the appropriate hosts:

ini

Copy code

# /etc/ansible/hosts

[demo]

192.168.1.10

192.168.1.11

**3.2. Writing the Playbook**

Create a new playbook file named first.yml:

bash

Copy code

vi first.yml

Paste the following content into the file:

yaml

Copy code

---

- hosts: demo

user: raman

become\_user: khanna

become\_method: su

tasks:

- name: Remove apache2 from the server

action: apt name=apache2 state=absent

- name: Create a file in the root directory

action: file path=/root/test state=present

- name: Remove nginx from the server

action: apt name=nginx state=absent

**3.3. Key Components of the Playbook**

* **user: raman**: The tasks will be executed on the remote hosts as the user raman.
* **become\_user: khanna**: The playbook will escalate privileges to the user khanna before running the tasks.
* **become\_method: su**: This specifies that su is used to switch to the khanna user.

**3.4. Running the Playbook**

To execute the playbook with privilege escalation, run the following command:

bash

Copy code

ansible-playbook first.yml

Ansible will log in as the user raman, escalate privileges to khanna, and then perform the tasks defined in the playbook.

**4. Expected Outcomes**

When the playbook is executed:

* The apache2 package will be removed from the remote hosts.
* A file named test will be created in the /root directory on the remote hosts.
* The nginx package will be removed from the remote hosts.

**5. Verifying the Results**

After running the playbook, you can manually verify the changes by logging into one of the remote hosts:

bash

Copy code

ssh raman@192.168.1.10

sudo su khanna

dpkg -l | grep apache2 # Verify apache2 is removed

dpkg -l | grep nginx # Verify nginx is removed

ls /root/test # Verify the file exists

**Conclusion**

This lab guide demonstrated how to use Ansible's privilege escalation features to perform tasks as a different user on remote hosts. By configuring and executing the first.yml playbook, you learned how to remove packages and create files with elevated privileges. This approach is essential when managing systems requiring administrative tasks across multiple hosts securely.

**Lab Guide: Ansible Playbook with Variables and Dynamic Execution**

**1. Prerequisites**

* Ansible is installed and configured on your control node.
* You have SSH access to the remote hosts grouped under demo in the inventory file.
* Your inventory file (/etc/ansible/hosts) is set up and accessible.

**2. Understanding the Playbook Structure**

The playbook third.yml provided allows for the installation of a package, the creation of users, and the use of variables that can be overridden at runtime.

yaml

Copy code

---

- hosts: demo

vars:

pkgname: tree

user1: gagan

user2: ""

uid1: 9876

uid2: 8766

tasks:

- name: Install tree

snap:

name: '{{pkgname}}'

state: present

- name: Creating user1

user:

name: '{{user1}}'

uid: '{{uid1}}'

state: present

- name: Creating user2

user:

name: '{{user2}}'

uid: '{{uid2}}'

state: present

**3. Step-by-Step Guide**

**3.1. Setting Up the Inventory File**

Ensure your inventory file contains the demo group with the appropriate hosts:

ini

Copy code

# /etc/ansible/hosts

[demo]

192.168.1.10

192.168.1.11

**3.2. Writing the Playbook**

Create a new playbook file named third.yml:

bash

Copy code

vi third.yml

Paste the following content into the file:

yaml

Copy code

---

- hosts: demo

vars:

pkgname: tree

user1: gagan

user2: ""

uid1: 9876

uid2: 8766

tasks:

- name: Install tree

snap:

name: '{{pkgname}}'

state: present

- name: Creating user1

user:

name: '{{user1}}'

uid: '{{uid1}}'

state: present

- name: Creating user2

user:

name: '{{user2}}'

uid: '{{uid2}}'

state: present

**4. Performing Syntax Checks and Dry Runs**

Before executing the playbook, it's essential to check for any syntax errors and understand the potential impact of running the playbook.

**4.1. Syntax Check**

Run the following command to check the syntax of your playbook:

bash

Copy code

ansible-playbook --syntax-check third.yml

**4.2. Dry Run (Check Mode)**

Use the --check flag to simulate the execution of the playbook without making any changes:

bash

Copy code

ansible-playbook third.yml --check

**4.3. Step-by-Step Execution**

To run the playbook interactively and confirm each step, use the --step flag:

bash

Copy code

ansible-playbook third.yml --step

**5. Executing the Playbook with Variable Overrides**

You can override the default variables defined in the playbook by passing them at runtime using the -e flag.

**5.1. Override a Single Variable**

Override the user2 variable to create a user named "gagan":

bash

Copy code

ansible-playbook third.yml -e user2="gagan"

**5.2. Override Multiple Variables**

Override both user1 and user2 to create users "raman" and "gagan":

bash

Copy code

ansible-playbook third.yml -e user1="raman" -e user2="gagan"

**6. Verifying the Results**

After executing the playbook, log into one of the remote hosts to verify the changes:

bash

Copy code

ssh raman@192.168.1.10

id gagan # Check if user gagan is created

id raman # Check if user raman is created

snap list | grep tree # Verify if the 'tree' package is installed

**7. Advanced Usage**

You can further customize this playbook by adding more tasks, using different package managers, or incorporating conditional logic to handle different scenarios.

**Conclusion**

This lab guide demonstrated how to use Ansible playbooks with variables for dynamic execution. You learned how to install packages, create users with dynamic names and UIDs, perform syntax checks, dry runs, and execute playbooks with variable overrides. This approach is versatile and can be adapted to many automation scenarios.

**Lab Guide: Ansible Network Automation with Cisco IOS Devices**

**1. Prerequisites**

* **Ansible Installation:** Ensure Ansible is installed on your control node.
* **SSH Access:** You have SSH access to your Cisco IOS devices.
* **Inventory Setup:** An inventory file (inv) is configured with your Cisco IOS devices.

**2. Setting Up the Inventory File**

The inventory file inv specifies the devices you will manage using Ansible, including connection details and necessary variables.

ini

Copy code

# inv file content

[dev01]

d1 ansible\_host=devnetsandboxiosxe.cisco.com

[dev02]

d2 ansible\_host=18.230.113.45

[dev\_group:children]

dev01

dev02

[dev01:vars]

ansible\_user="admin"

ansible\_connection=network\_cli

ansible\_password="C1sco12345"

ansible\_network\_os="ios"

[dev02:vars]

ansible\_user="ec2-user"

ansible\_ssh\_private\_key\_file="/root/cisco.pem"

ansible\_connection=network\_cli

ansible\_network\_os="ios"

* **ansible\_host:** The IP address or hostname of the device.
* **ansible\_user:** The username to connect to the device.
* **ansible\_connection:** Specifies the connection type (network\_cli for network devices).
* **ansible\_password or ansible\_ssh\_private\_key\_file:** Authentication details.
* **ansible\_network\_os:** Specifies the network OS (e.g., ios for Cisco IOS).

**3. Understanding the Playbook Structure**

The playbook fifth.yml performs several tasks on Cisco IOS devices, including gathering device facts, changing the hostname, creating a user, and running commands.

yaml

Copy code

# fifth.yml content

- name: Network Getting Started First Playbook Extended

connection: network\_cli

gather\_facts: false

hosts: dev\_group

vars:

username: ansible

privilege: 15

password: ansible

tasks:

- name: Get config for IOS devices

cisco.ios.ios\_facts:

gather\_subset: all

- name: Display the config

debug:

msg: "The hostname is {{ ansible\_net\_hostname }} and the OS is {{ ansible\_net\_version }}"

- name: Change hostname

ios\_config:

lines:

- hostname ANSIBLE\_LUMEN

- name: Create Username

cisco.ios.ios\_config:

backup: yes

lines:

- username {{ username }} priv {{ privilege }} secret {{ password }}

- name: Get changed config for IOS devices

cisco.ios.ios\_facts:

gather\_subset: all

register: res

- name: Display changed config

debug:

var: res.ansible\_facts.ansible\_net\_config

- name: Run show users

ios\_command:

commands: sh run | i user

register: rk

- name: Display user information

debug:

var: rk

**4. Step-by-Step Execution**

**4.1. Running the Playbook for All Devices**

To execute the playbook for all devices in the dev\_group, run the following command:

bash

Copy code

ansible-playbook -i inv fifth.yml

This will perform the tasks on both dev01 and dev02.

**4.2. Running the Playbook for a Specific Device**

To limit the playbook execution to a specific device, use the -l option:

* **For dev01:**

bash

Copy code

ansible-playbook -i inv fifth.yml -l dev01

* **For dev02:**

bash

Copy code

ansible-playbook -i inv fifth.yml -l dev02

**5. Understanding Key Tasks in the Playbook**

* **Gathering Facts:**

The playbook starts by gathering all configuration details from the IOS devices.

yaml

Copy code

- name: Get config for IOS devices

cisco.ios.ios\_facts:

gather\_subset: all

* **Changing the Hostname:**

The hostname of the device is changed to ANSIBLE\_LUMEN.

yaml

Copy code

- name: Change hostname

ios\_config:

lines:

- hostname ANSIBLE\_LUMEN

* **Creating a User:**

A new user (ansible) is created with privilege level 15 and a secret password.

yaml

Copy code

- name: Create Username

cisco.ios.ios\_config:

backup: yes

lines:

- username {{ username }} priv {{ privilege }} secret {{ password }}

* **Running Commands:**

The playbook runs a command to display users on the device.

yaml

Copy code

- name: Run show users

ios\_command:

commands: sh run | i user

**6. Verifying the Changes**

After executing the playbook, you can manually verify the changes by logging into one of the devices:

bash

Copy code

ssh admin@devnetsandboxiosxe.cisco.com

show running-config | include hostname # Verify the hostname change

show running-config | include username # Verify the user creation

**Conclusion**

This lab guide demonstrated how to automate network device management using Ansible. You learned how to configure an inventory file, create a playbook to manage Cisco IOS devices, and execute the playbook to perform tasks like gathering device facts, changing hostnames, and creating users. This approach is essential for managing large-scale networks efficiently.