Experiment 7: Configuration Management with Ansible: Basics of Ansible: Inventory, Playbooks, and Modules, Automating Server Configurations with Playbooks, Hands-On: Writing and Running a Basic Playbook

1. Introduction to Ansible

What Is Ansible?

Ansible is an open-source IT automation and configuration management tool. It allows you to manage multiple servers and perform tasks such as:

· **Configuration Management:** Automate the configuration of servers.

· **Application Deployment:** Deploy applications consistently.

· **Orchestration:** Coordinate complex IT workflows and processes.

Key Concepts in Ansible

· **Inventory:**

An inventory is a file (usually in INI or YAML format) that lists the hosts (or groups of hosts) you want to manage. It tells Ansible which machines to target.

· **Playbook:**

A playbook is a YAML file that defines a set of tasks to be executed on your target hosts. It is the heart of Ansible automation. In a playbook, you specify:

o **Hosts:** The target machines (or groups) on which the tasks should run.

o **Tasks:** A list of actions (using modules) that should be executed.

o **Modules:** Reusable, standalone scripts that perform specific actions (e.g., installing packages, copying files, configuring services).

· **Modules:**

Ansible comes with a large collection of built-in modules (such as apt, yum, copy, service, etc.). These modules perform specific tasks on target hosts. You can also write custom modules

· **Agentless:** Ansible uses SSH to communicate with target hosts, so no agent needs to be installed on them.

· **Simplicity:** Playbooks use simple YAML syntax, making them easy to write and understand.

· **Idempotence:** Ansible tasks are idempotent, meaning running the same playbook multiple times yields the same result, ensuring consistency.

· **Scalability:** Ansible can manage a small number of servers to large infrastructures with hundreds or thousands of nodes.

2. Installing Ansible on Ubuntu

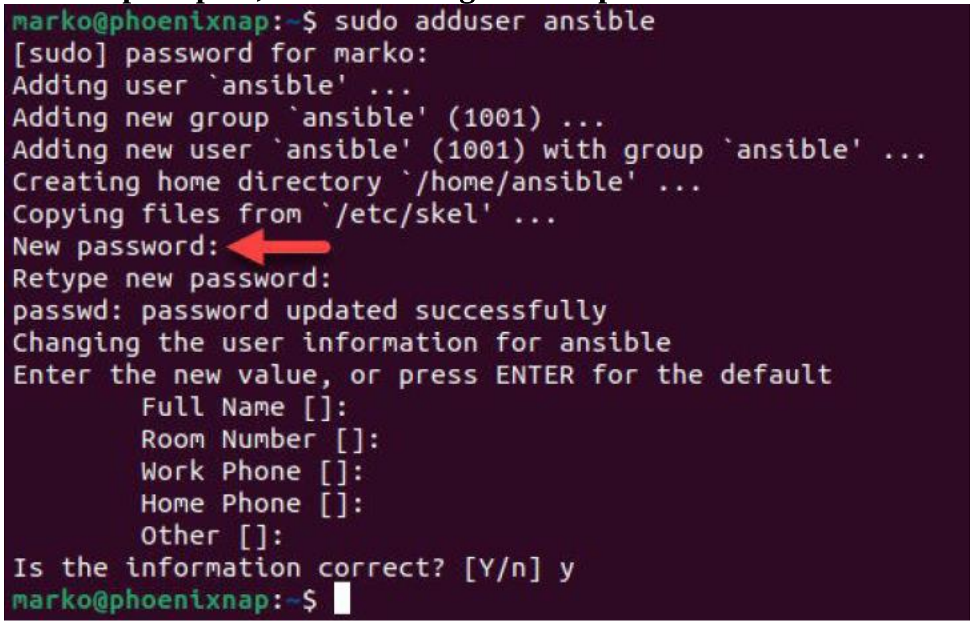
STEP 1: Configure Ansible Control Node

The Ansible control node is a system used to connect to and manage Ansible host servers. Proceed with the steps below to set up the control node on the main server:

1. Create an administrator-level user for the control node. Use the adduser command:

sudo adduser username

2. When prompted, define a strong account password.



```
marko@phoenixnap:~$ sudo adduser ansible
[sudo] password for marko:
Adding user `ansible' ...
Adding new group `ansible' (1001) ...
Adding new user `ansible' (1001) with group `ansible' ...
Creating home directory `/home/ansible' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for ansible
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
marko@phoenixnap:~$
```

Optionally, provide more details about the user by answering questions. Press Enter to skip a question.

3. Use the following usermod command to assign superuser privileges to the account:

sudo usermod -aG sudo username

A membership in the sudo group allows the user to utilize the sudo command to perform administrative tasks.

4. Switch to the newly created user on the control node:

sudo su username

Note: The Ansible control node can be a dedicated server, a local machine, or a virtual machine running Ubuntu.

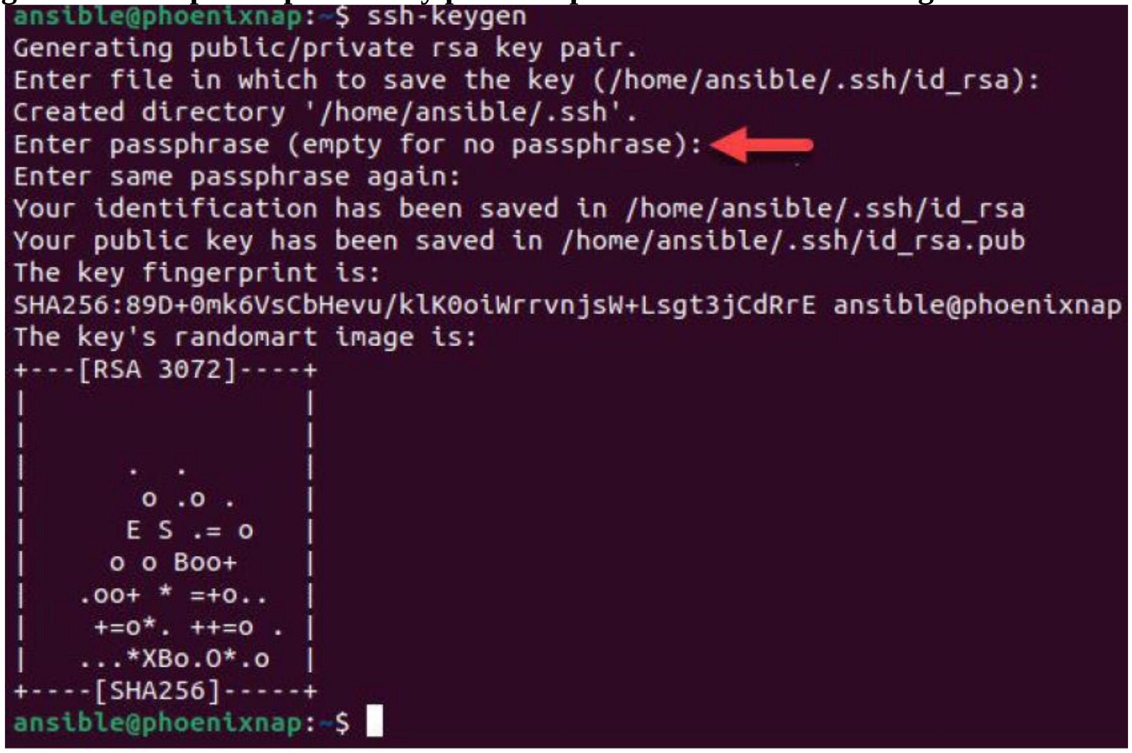
STEP 2: Set up an SSH Key pair

The Ansible control node uses SSH to connect to hosts. Generate an SSH key pair for the Ansible user by executing the following steps:
1. Enter the command below using the Ansible control node command line:

ssh-keygen

Note: If an SSH key pair with the same name already exists, SSH displays a warning asking the user to decide whether to overwrite it. Overwriting makes the previous SSH key pair unusable, so ensure the old keys are no longer needed before confirming.

2. When prompted, provide a passphrase. While adding a strong passphrase is recommended, pressing Enter allows the user to skip the passphrase creation. The system generates the public/private key pair and prints the randomart image.



```
ansible@phoenixnap:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ansible/.ssh/id_rsa):
Created directory '/home/ansible/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ansible/.ssh/id_rsa
Your public key has been saved in /home/ansible/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:89D+0mk6VsCbHevu/klK0oiWrrvnjsW+Lsgt3jCdRrE ansible@phoenixnap
The key's randomart image is:
+---[RSA 3072]----+
|                 |
|                 |
|       .  .      |
|        o .o .   |
|       E S .= o  |
|      o o Boo+   |
|    .oo+ * =+o.. |
|     +=o*. ++=o .|
|    ...*XBo.O*.o |
+----[SHA256]-----+
ansible@phoenixnap:~$
```

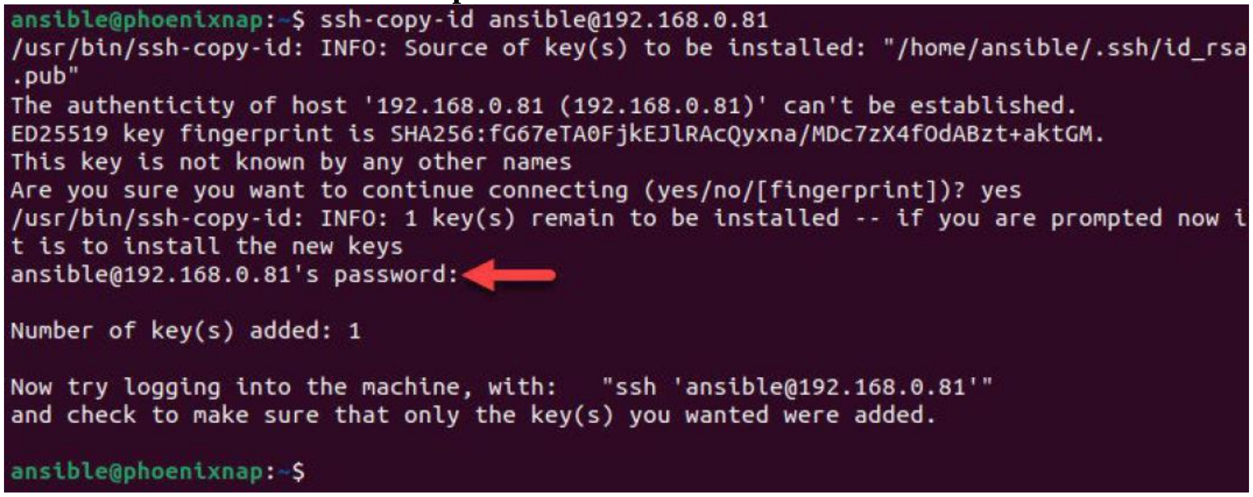
<u>STEP 3: Configure an Ansible Host</u>

Ansible hosts are remote servers managed by the Ansible control node. Each host must have the control node's SSH public key into authorized_keys directory. Apply the steps below for each new Ansible host:
1. Use the following ssh-copy-id command on the control node to copy the public key to a host:

ssh-copy-id username@remote-host

Replace [username] with an existing administrative user on the host system and [remote-host] with the remote host domain or IP address. For example, to copy the key to the user ansible on the host with the local IP address 192.168.0.81, type:

2. Type yes and hit Enter when asked whether to continue connecting to an authenticated host.
3. Enter the remote host account password.



```
ansible@phoenixnap:~$ ssh-copy-id ansible@192.168.0.81
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ansible/.ssh/id_rsa
.pub"
The authenticity of host '192.168.0.81 (192.168.0.81)' can't be established.
ED25519 key fingerprint is SHA256:fG67eTA0FjkEJlRAcQyxna/MDc7zX4fOdABzt+aktGM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now i
t is to install the new keys
ansible@192.168.0.81's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'ansible@192.168.0.81'"
and check to make sure that only the key(s) you wanted were added.

ansible@phoenixnap:~$
```

The utility uploads the public key to the remote host account.

STEP4 : Install Ansible

Use the APT package manager to install the Ansible package on the control node system:
1. Ensure the package index is up to date

sudo apt update

2. Install Ansible on Ubuntu with the following command:

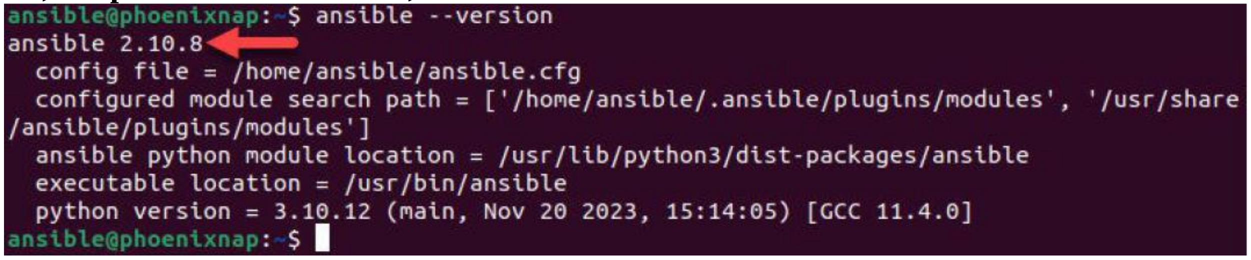
sudo apt install ansible –y

STEP 5: Verify the Installation

Check that Ansible was successfully installed on your Ubuntu system using the ansible command:

ansible --version

The output displays the Ansible version number, the location of the configuration file, the path to the executable, and other information.



```
ansible@phoenixnap:~$ ansible --version
ansible 2.10.8
  config file = /home/ansible/ansible.cfg
  configured module search path = ['/home/ansible/.ansible/plugins/modules', '/usr/share
/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]
ansible@phoenixnap:~$
```

STEP 6: Set up the Inventory File

Once Ansible is installed on the control node, set up an inventory file to allow Ansible to communicate with remote hosts. The inventory file contains all the information about the remote hosts managed through the Ansible control node.

Follow the steps below to create an inventory file on the control node:
1. Create the ansible subdirectory in the etc directory:

```
sudo mkdir -p /etc/ansible
```

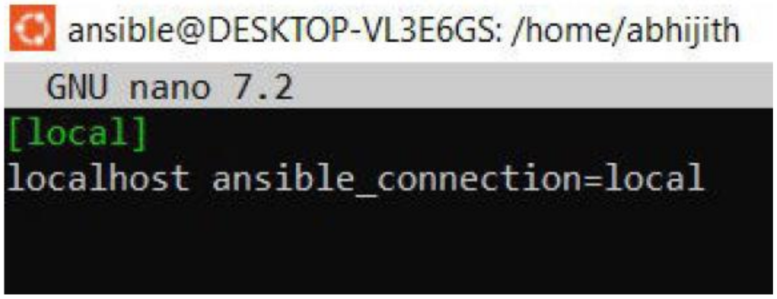
2. Use a text editor such as Nano to create a file named hosts:

```
sudo nano /etc/ansible/hosts
```

3. Add localhost that the control node will manage. Use the following format:

```
[local]
localhost ansible_connection=local
```

The [local] line allows for the creation of categories to organize local hosts. The following example adds a local host using its local IP address 192.168.0.81 and sorts it into the servers category:

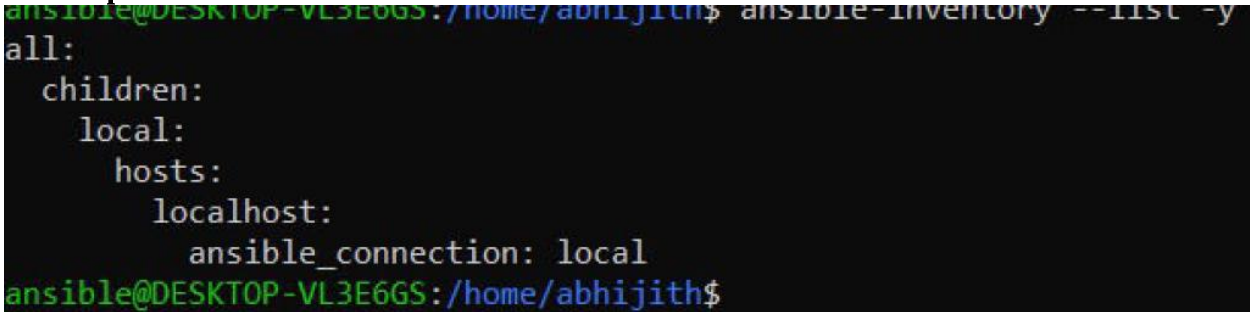


```
ansible@DESKTOP-VL3E6GS: /home/abhijith
 GNU nano 7.2
[local]
localhost ansible_connection=local
```

4. Save the file and exit.
5. Enter the command below to check the items in the inventory:

```
ansible-inventory --list -y
```

The output lists the hosts:



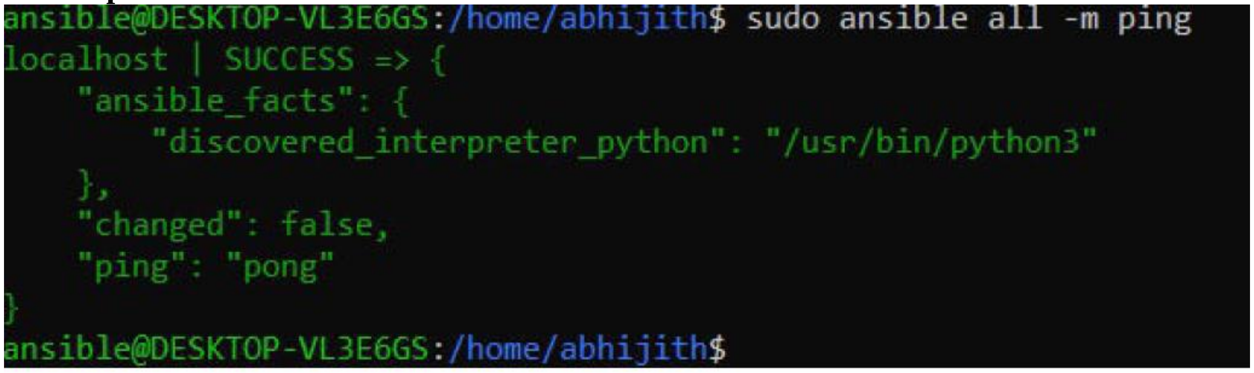
```
ansible@DESKTOP-VL3E6GS:/home/abhijith$ ansible-inventory --list -y
all:
  children:
    local:
      hosts:
        localhost:
          ansible_connection: local
ansible@DESKTOP-VL3E6GS:/home/abhijith$
```

STEP 7: Test the Connection
To ensure the Ansible control node can connect to the local hosts and run commands, use the following ansible command to ping the hosts from the control node:

```
sudo ansible all -m ping
```

Note: When a user connects to the remote hosts for the first time, Ansible asks for confirmation that the hosts are authentic. To confirm the authenticity, enter yes when prompted.

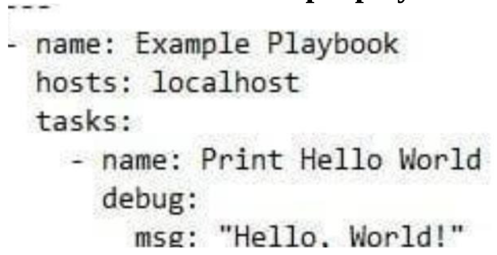
The output confirms the successful connection.



```
ansible@DESKTOP-VL3E6GS:/home/abhijith$ sudo ansible all -m ping
localhost | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
ansible@DESKTOP-VL3E6GS:/home/abhijith$
```

The Ansible control node is now set up to control the connected remote hosts.

STEP 8: Create a sample playbook called HelloWorld.yml and type the following



```
---
- name: Example Playbook
  hosts: localhost
  tasks:
    - name: Print Hello World
      debug:
        msg: "Hello, World!"
```

Save the file and exit
Run the program as ansible-playbook HelloWorld.yml
And note down the output.