# Ansible Handbook

V 1.0

Tariq Mehmood
SHERDIL IT ACADEMY

# Contents

Ansible ..... 2
Key terms of Ansible: ..... 2
Setting up Ansible ..... 3
1. Configuring Ansible Controller ..... 3
2. Configuring Ansible hosts ..... 8
3. Setting up SSH ..... 9
Ansible ad-hoc Commands ..... 11
Ansible Modules ..... 12
Ansible Playbook ..... 14
Key Elements of a Playbook: ..... 14
Extra features of a Playbook: ..... 17
1. Condition (when): ..... 17
2. Loops: ..... 17
3. Tags: ..... 18
Ansible Vault: ..... 19
How to Use Ansible Vault: ..... 19

# Ansible

Ansible is an open-source automation tool used to automate tasks in IT systems. It works as a remote controller for managing computers, servers, and other devices. Ansible can be used to setting up software, configuring systems, or running commands on each device.

It works by using simple ad-hoc commands or by using book of instructions called playbooks written in a language called YAML. These playbooks tell Ansible what actions to perform on host devices.

It connects to target systems via SSH or WinRM. It's agentless, meaning no special software needs to be installed on the managed nodes.

It is push-based this means that the control node pushes out configurations or tasks to the host nodes when you execute an ad-hoc command /playbook.

It is known for its scalability, ease of use, and ability to manage diverse environments, including servers, cloud platforms, and network devices, all while ensuring consistency and reliability across systems.

## Key terms of Ansible:

- **Ansible Controller**: The machine from which Ansible is run.
- **Module**: A unit of work Ansible uses to perform tasks.
- **Task**: A single action within a playbook.
- **Role**: A reusable set of tasks and configurations.
- **Fact**: Information about a system gathered by Ansible.
- **Play**: A section of a playbook targeting specific hosts with specific tasks.
- **Host**: A target machine managed by Ansible.
- **Inventory**: A file listing all hosts and groups of hosts for automation

# Setting up Ansible

It can be divided in 3 parts:

1. Configuring Ansible Controller
2. Configuring Ansible hosts
3. Setting up SSH

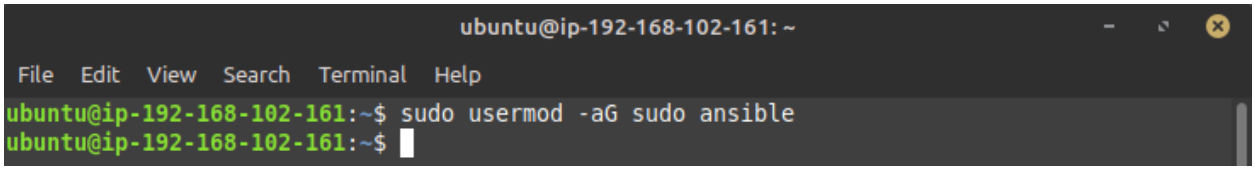
## 1. Configuring Ansible Controller

1st we create ansible user and provide it sudo privileges with following commands (for debian based system):

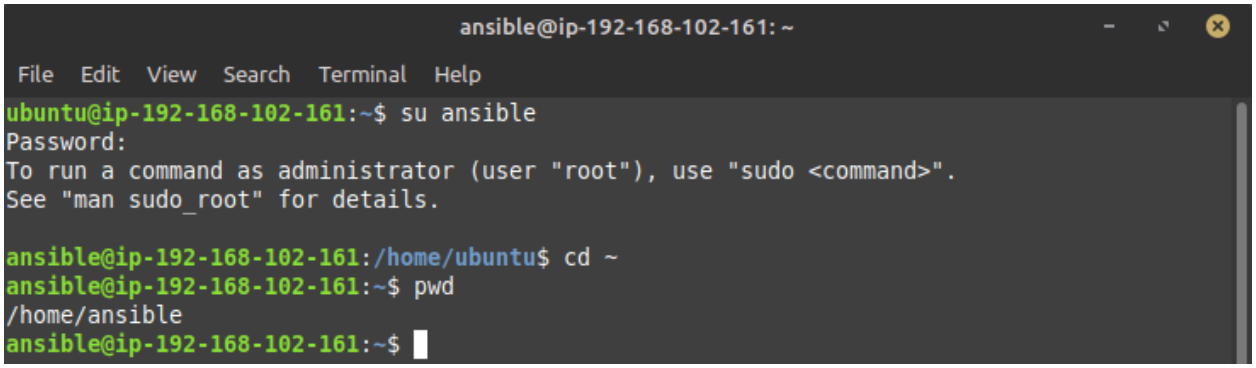
Creating user ansible:
>> sudo adduser ansible

```
ubuntu@ip-192-168-102-161:~$ sudo adduser ansible
Adding user `ansible' ...
Adding new group `ansible' (1006) ...
Adding new user `ansible' (1005) 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 []: ansible
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
ubuntu@ip-192-168-102-161:~$
```

Giving ansible user sudo privileges:
>> sudo usermod -aG sudo ansible




Switching to ansible user:
>> su ansible




Now we have created ansible user on controller now we need to install and configure ansible on controller with following commands:

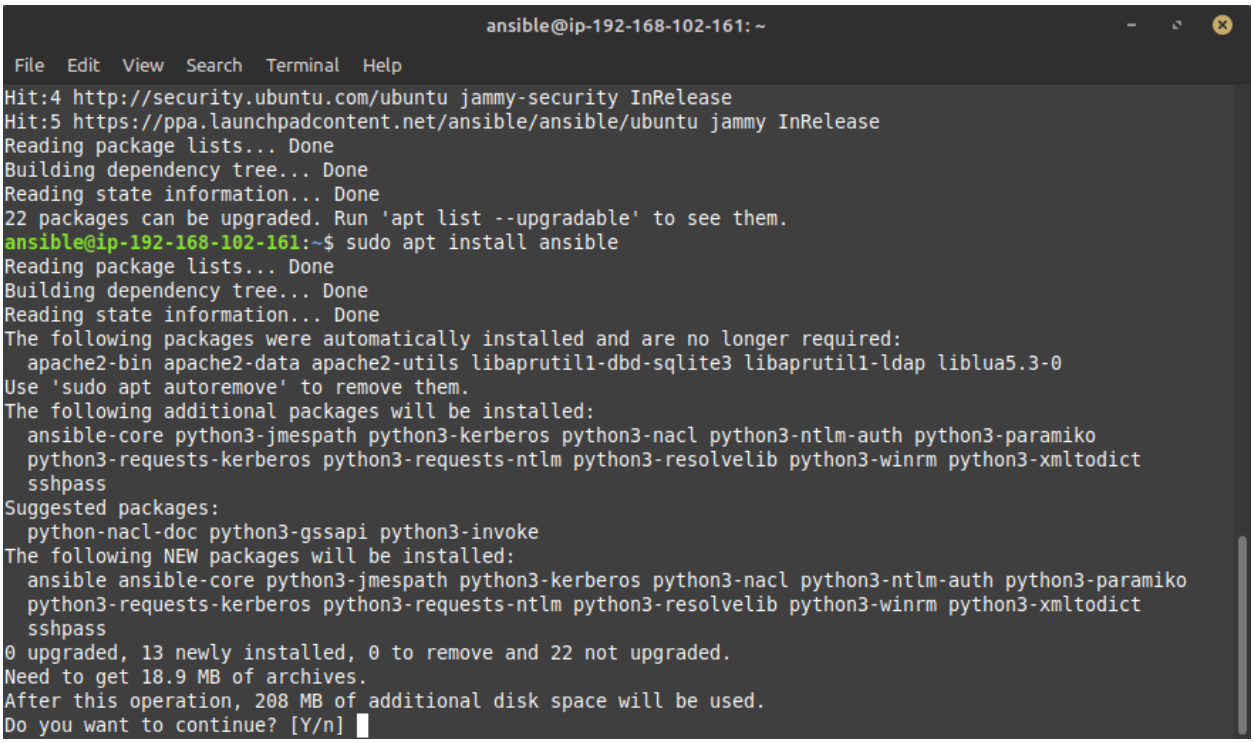
Installing ansible:

>> sudo apt install software-properties-common

>> sudo add-apt-repository --yes --update ppa:ansible/ansible

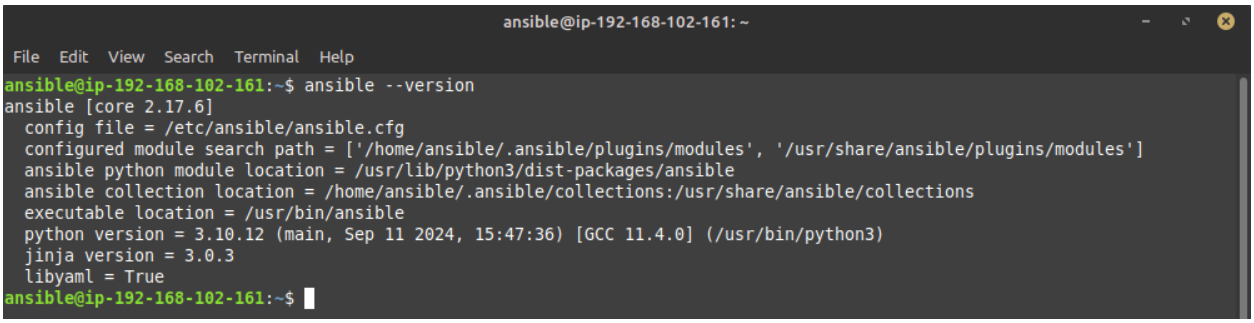
>> sudo apt update

>> sudo apt install ansible




Verifying ansible installation

>> ansible --version




Basic Configuration:

>> cd /etc/ansible  #Create this directory if not existed

>> sudo su

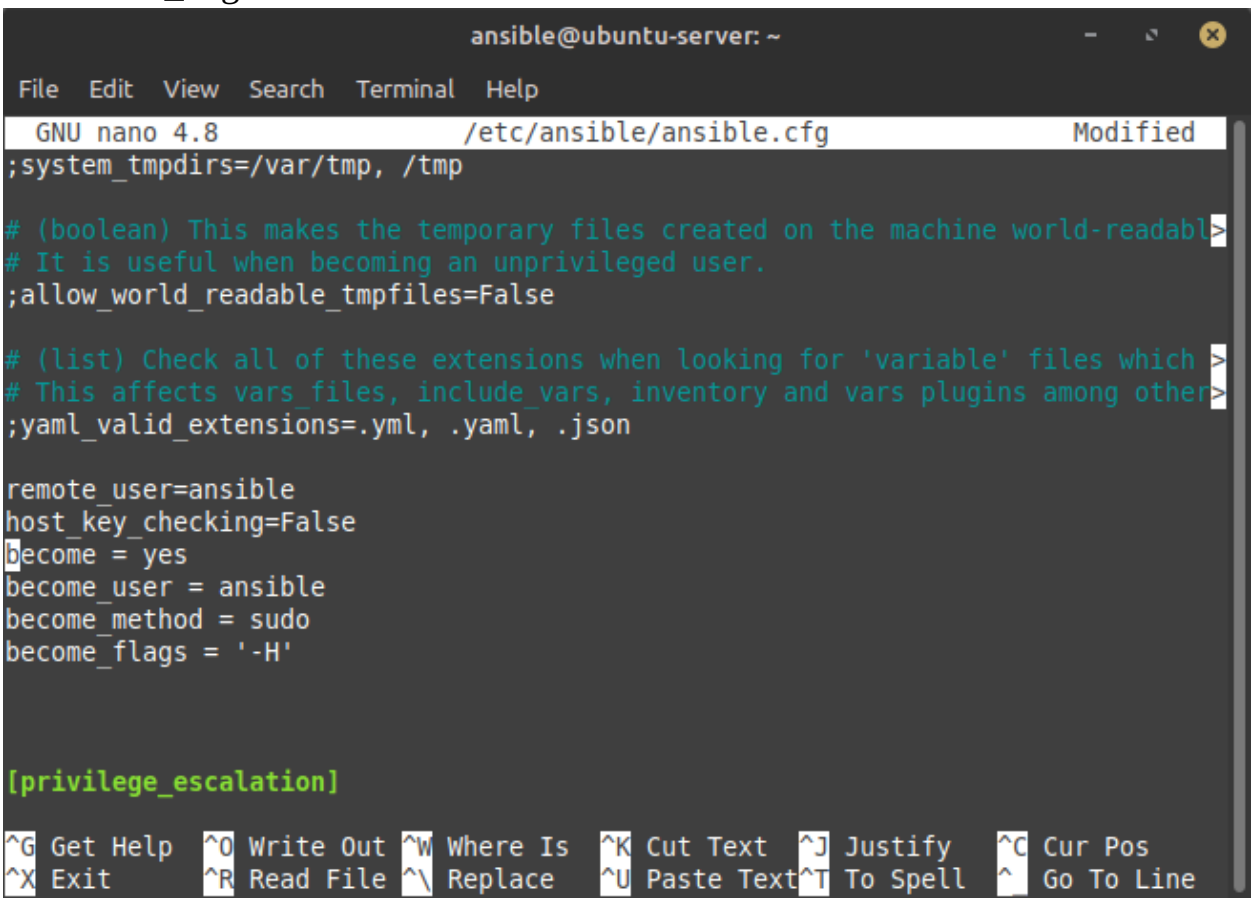
>> ansible-config init  -t all --disabled > ansible.cfg

>> exit

>> sudo nano /etc/ansible/ansible.cfg

#Added the following configuration settings after the [defaults]:

>> remote_user=ansible
>>host_key_checking=False
>>become = yes
>>become_user = ansible
>>become_method = sudo
>>become_flags = '-H'



Adding host group & host IPs in inventory

>> sudo nano /etc/ansible/hosts

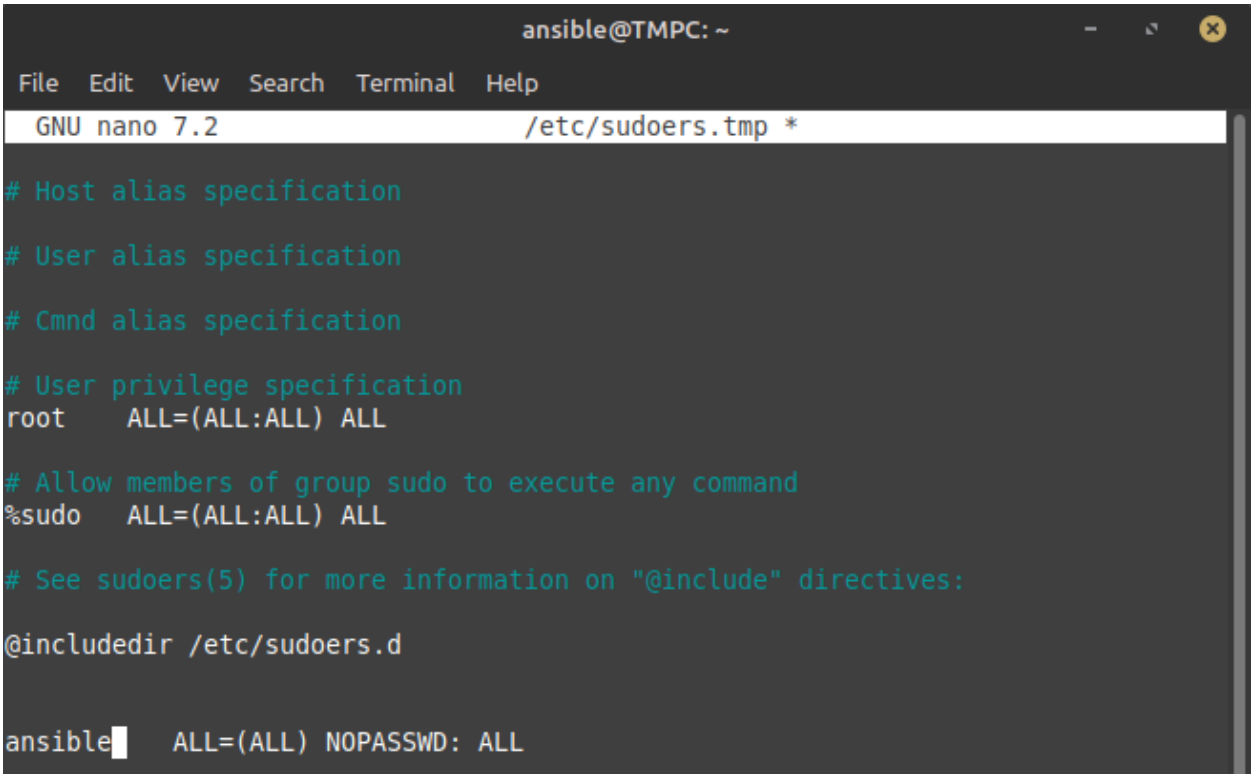
## 2. Configuring Ansible hosts

Creating ansible user with sudo privileges and installing ansible in host systems with following commands (for debian based system):

>> sudo adduser ansible
>> sudo usermod -aG sudo ansible
>> su ansible
>> sudo apt install software-properties-common
>> sudo add-apt-repository --yes --update ppa:ansible/ansible
>> sudo apt update
>> sudo apt install ansible

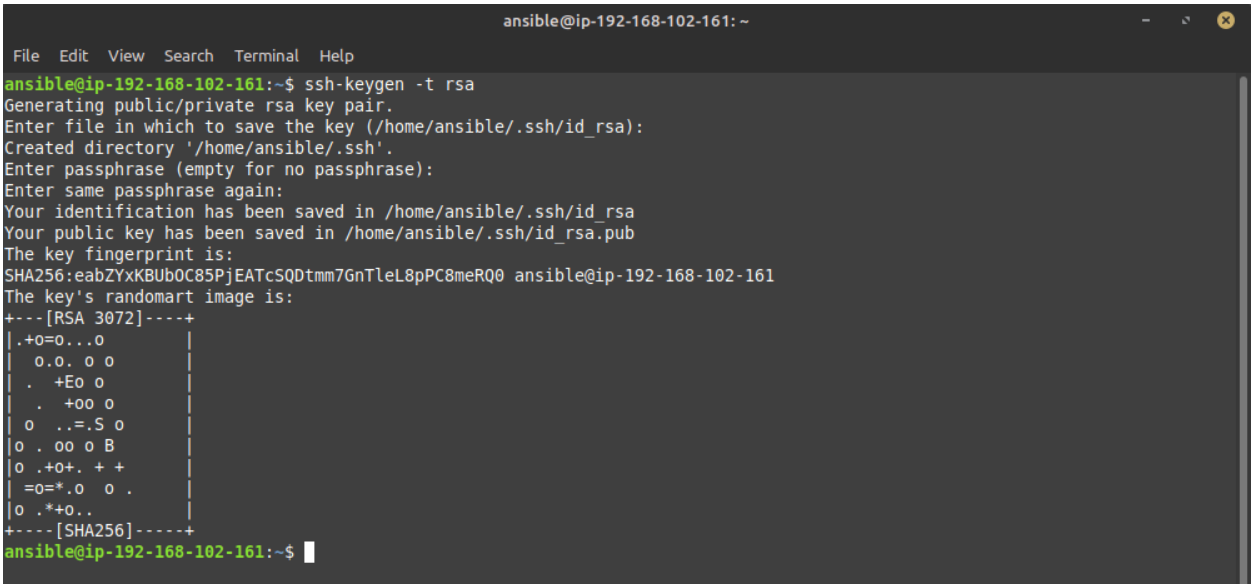
Setting up password-less sudo privileges on host

>> sudo visudo
# Adding following in file
>> ansible ALL=(ALL) NOPASSWD: ALL

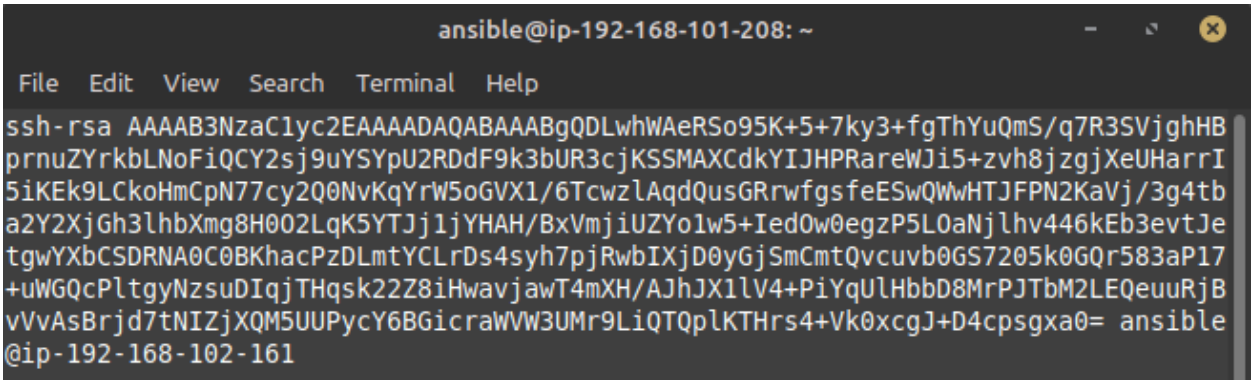


## 3. Setting up SSH

Creating ssh-keygen for ansible user on ansible controller:
>> ssh-keygen -t rsa

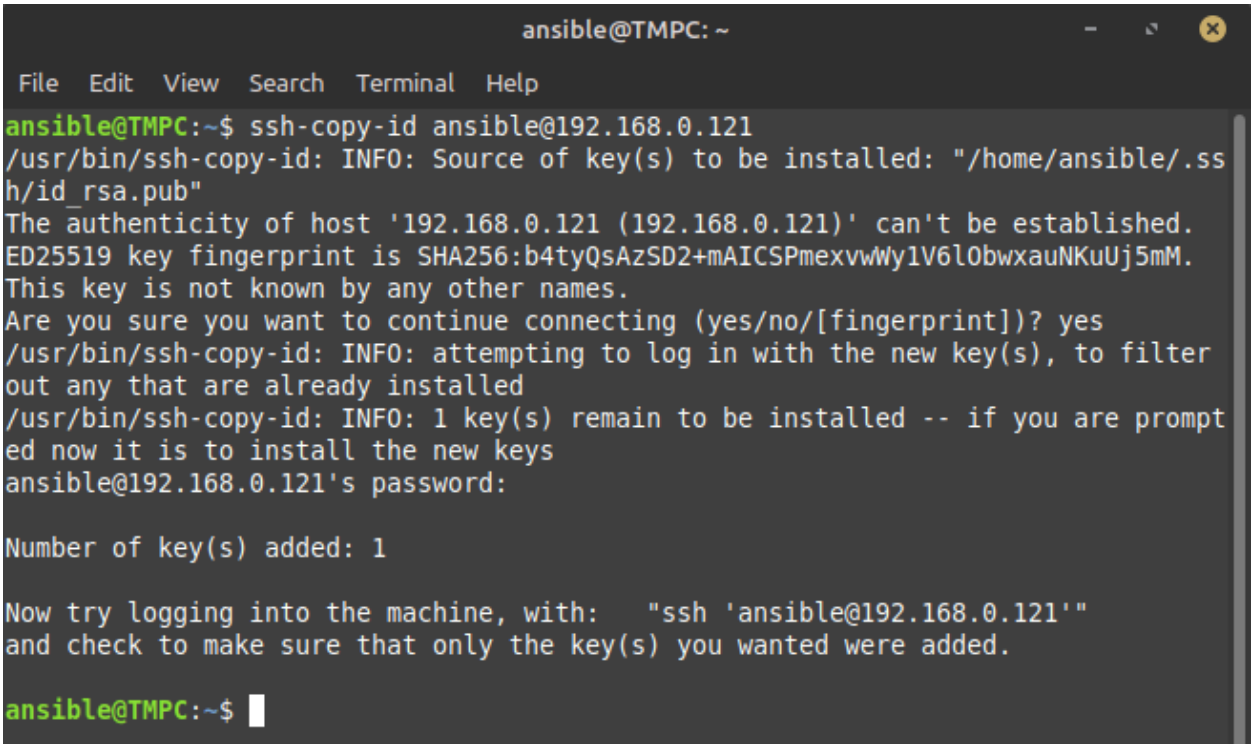



Viewing Ansible controller public ssh key
>>cat .ssh/id_rsa.pub



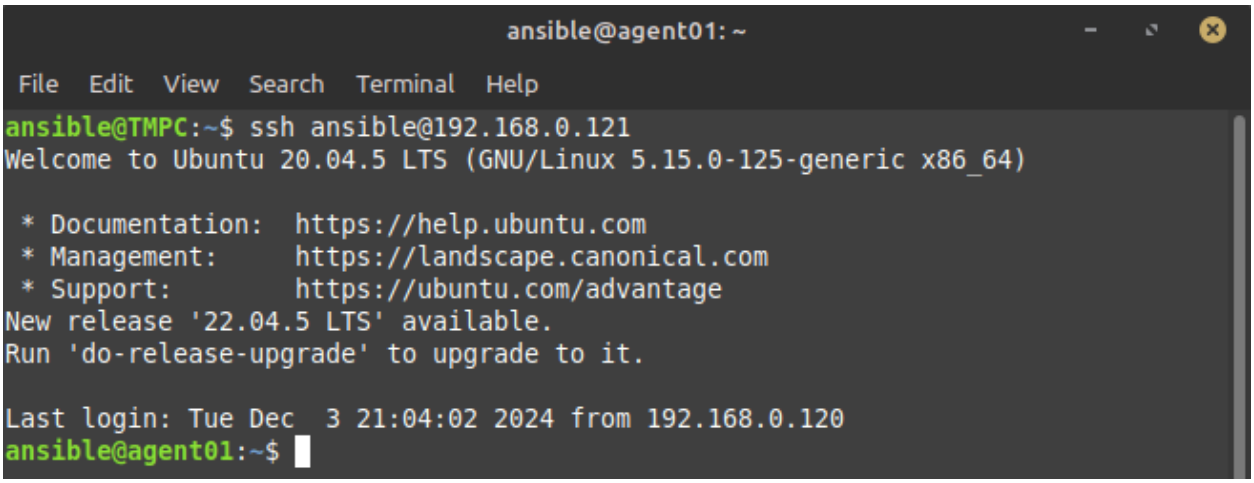
Sharing ssh with following command:

>>ssh-copy-id ansible@host_ip




Verifying ssh sharing

>>ssh ansible@host_ip

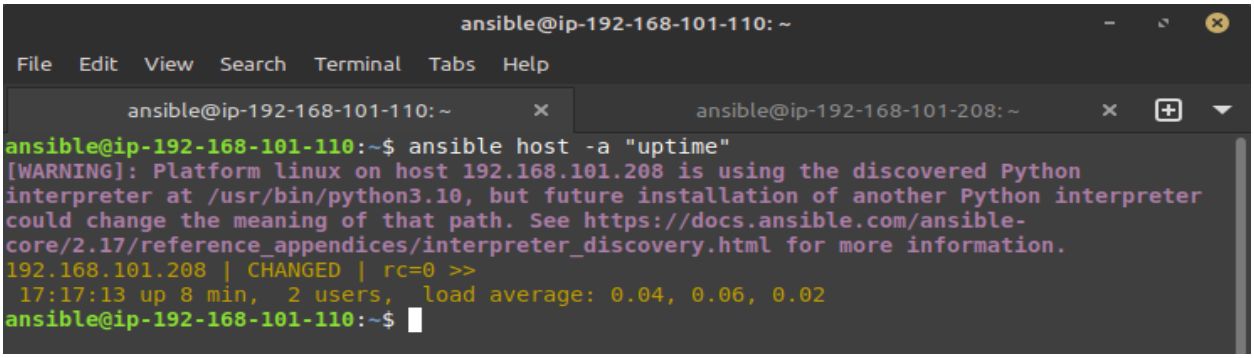


# Ansible ad-hoc Commands

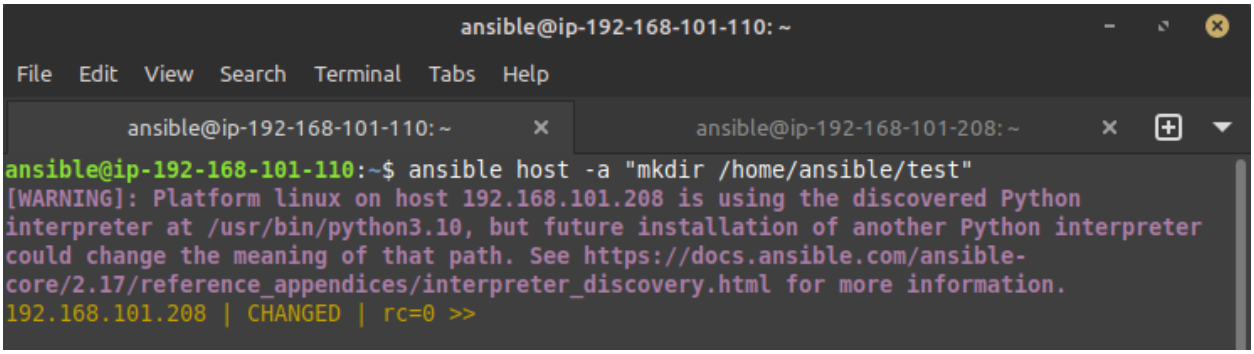
In Ansible, an ad-hoc command is a one-time command that allows you to execute tasks on host systems without creating a full playbook. This is useful for quick, simple tasks. We can execute ad-hoccommands directly (without modules) with -a flag some example for those commands are followings:

>> ansible <hosts> -a "<shell command>"

i. ansible host -a "uptime"




ii. ansible host -a "mkdir /home/ansible/test"

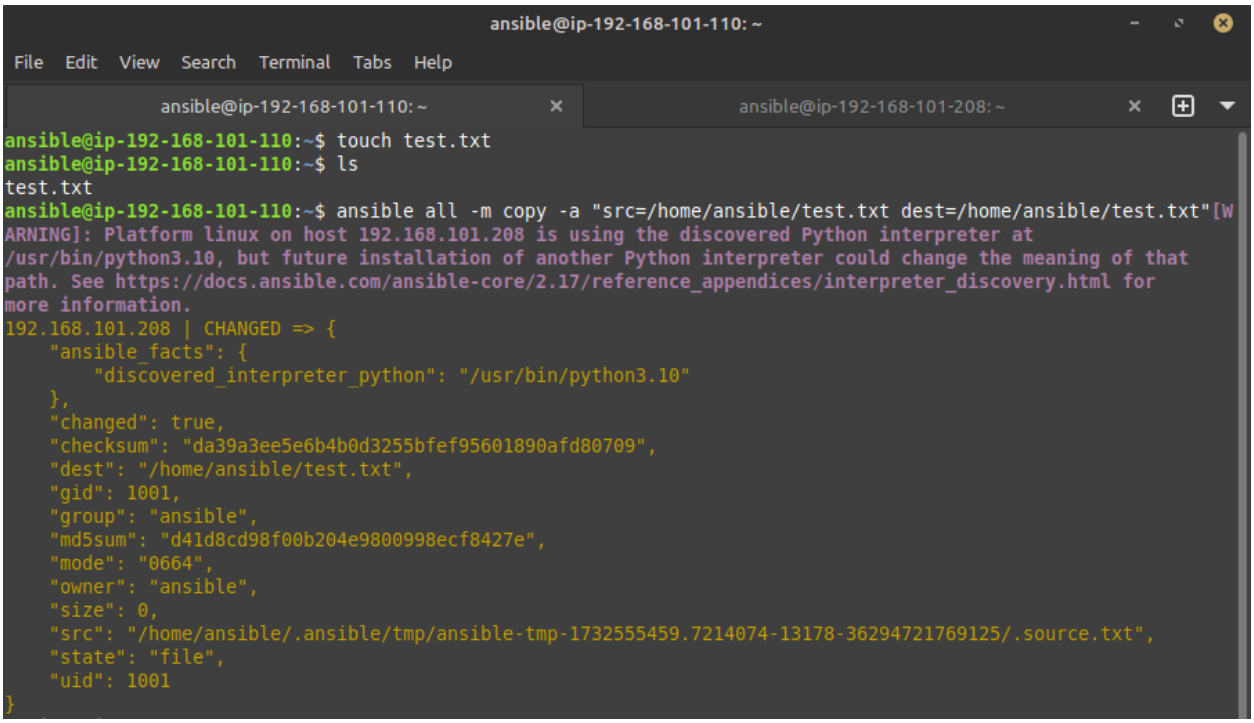


# Ansible Modules

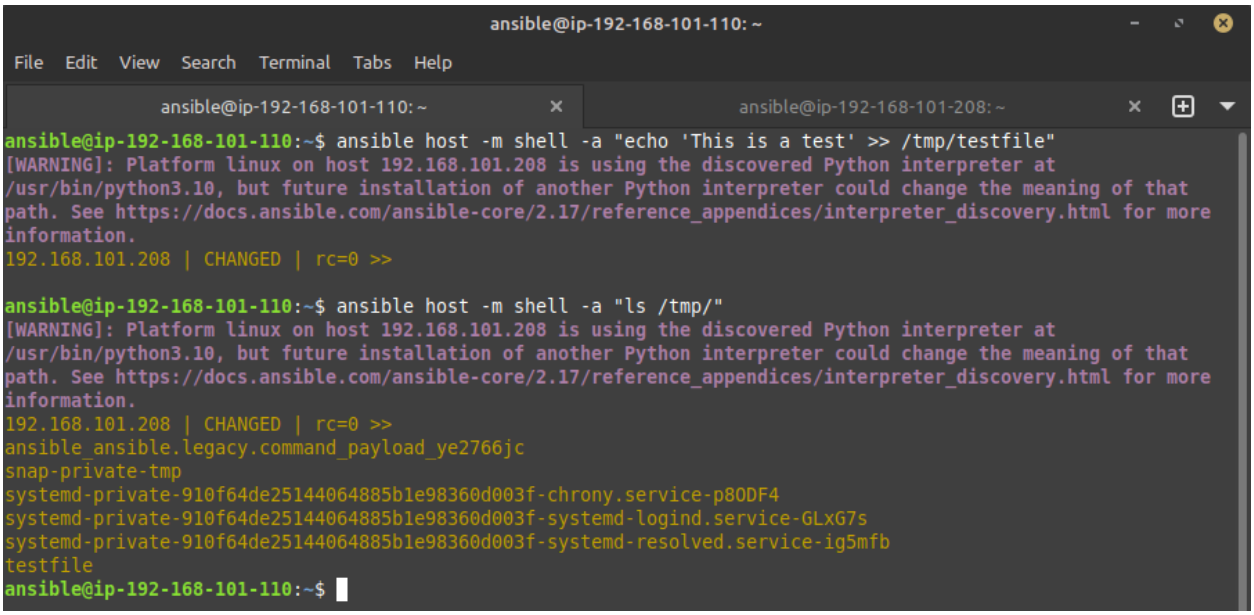
We can also use predefined modules to perform task easily some example are followings:

>> ansible <hosts> -m <module> -a "<arguments>"

i. ansible all -m copy -a "src=/home/ansible/test.txt dest=/home/ansible/test.txt"



ii. ansible host -m shell -a "echo 'This is a test' >> /tmp/testfile"



# Ansible Playbook

An Ansible playbook is a YAML file that defines a series of tasks to be executed on host systems. It allows execution of series of task to multiple host in one go.

## Key Elements of a Playbook:

- **Target**: Part of playbook that defines host & basic configurations.
- **Tasks**: Each task defines a specific action to be performed. Tasks use Ansible modules to perform actions.
- **Variables**: You can define variables to customize the behavior of your playbook.
- **Handlers:** These are special tasks that only run when notified by another task. They are typically used for tasks like restarting services or reloading configurations after changes.

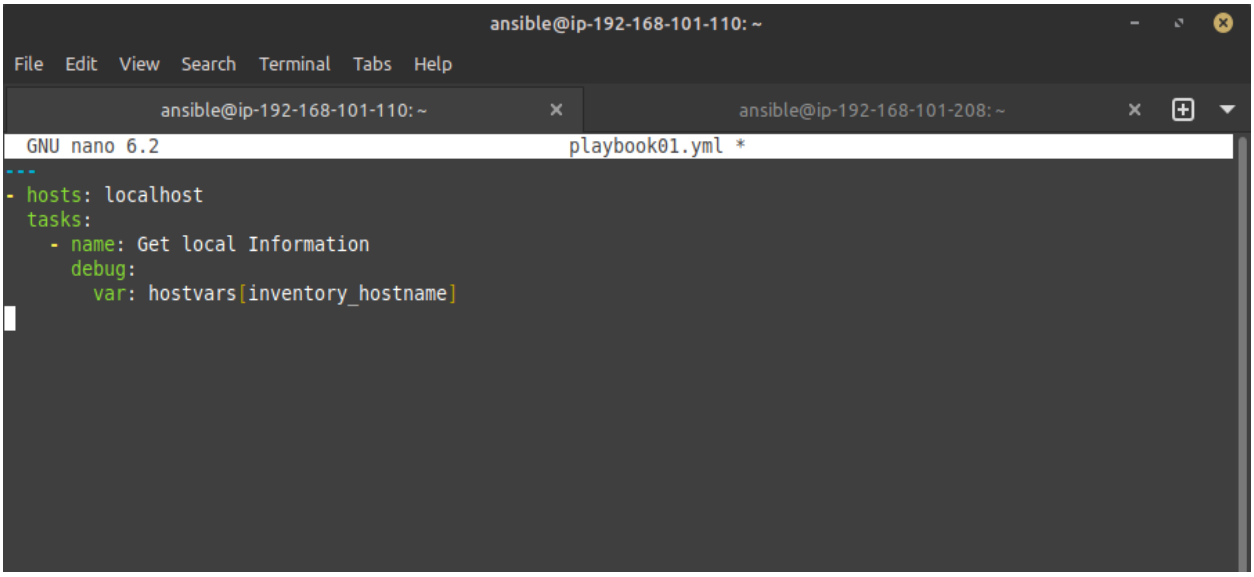
**Following is example of YAML ansible playbook**

```
---
- host: webservers
  become: yes  # Run tasks with sudo privileges
  tasks:
    - name: Install nginx package
      apt:
        name: nginx
        state: present
    - name: Start nginx service
      service:
        name: nginx
        state: started
```

Playbook can be executed with following command:
>> ansible-playbook <playbook.yml>

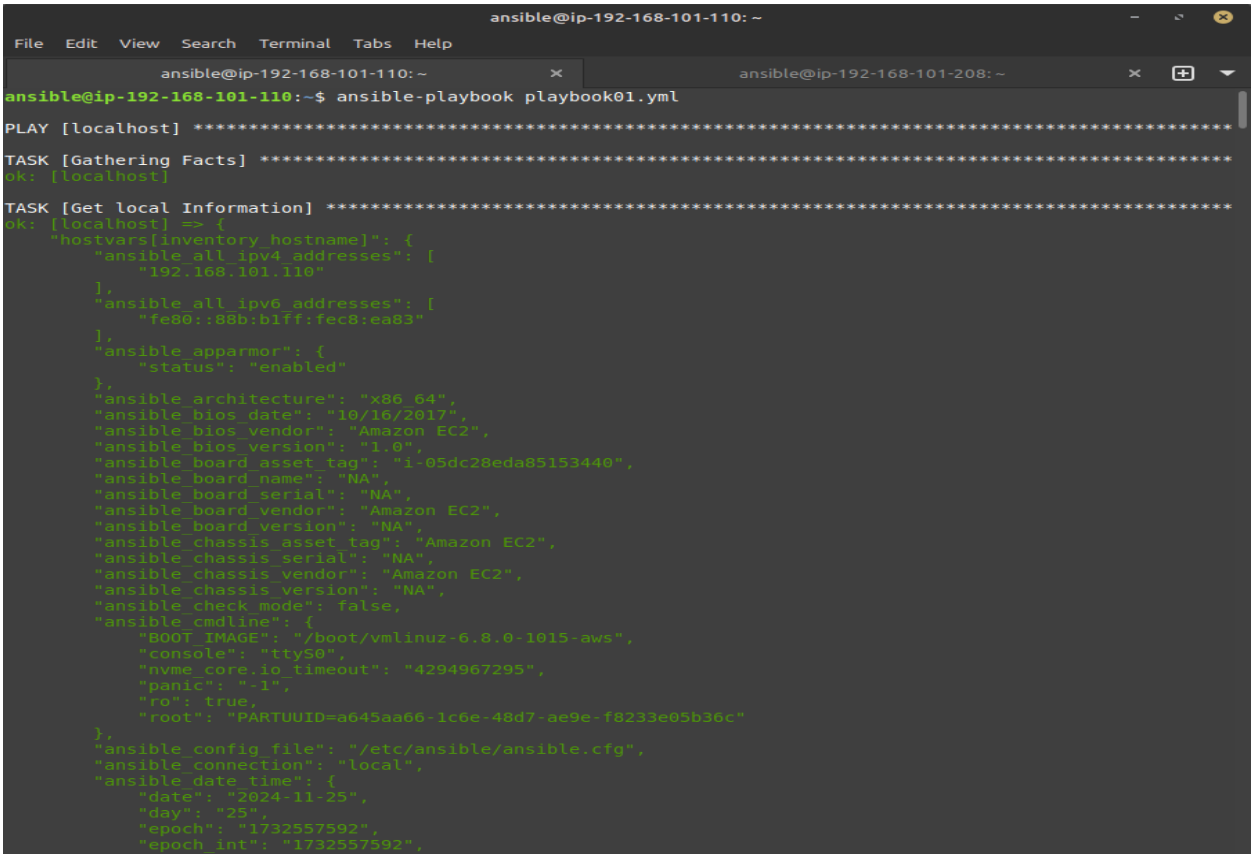
Ansible Playbook Examples:

i. Creating playbook01.yml which shows localhost‘s all the facts and variables:

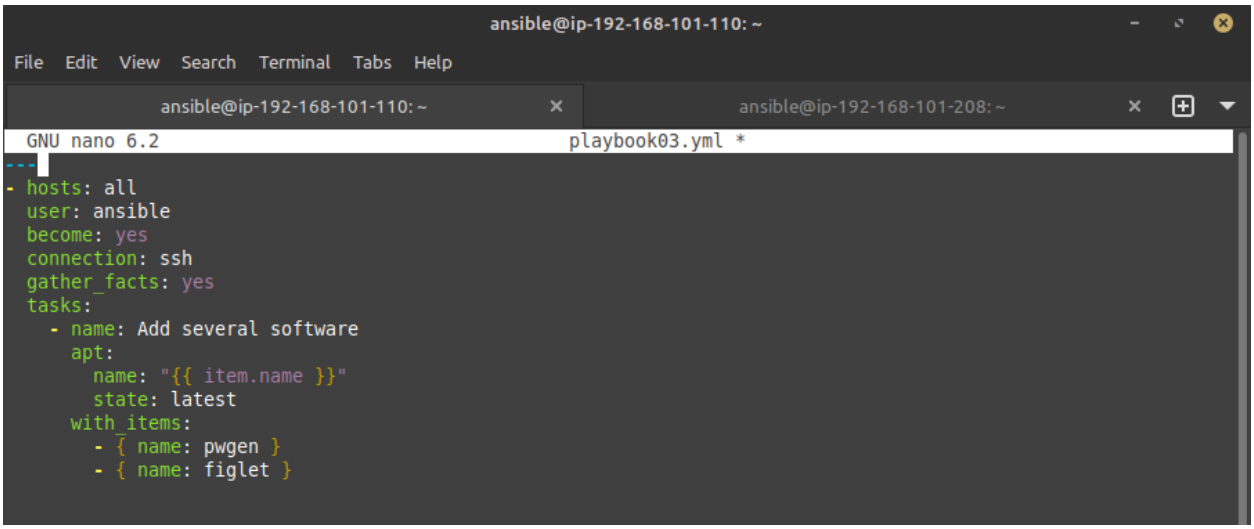



Running playbook:

>> ansible-playbook playbook01.yml

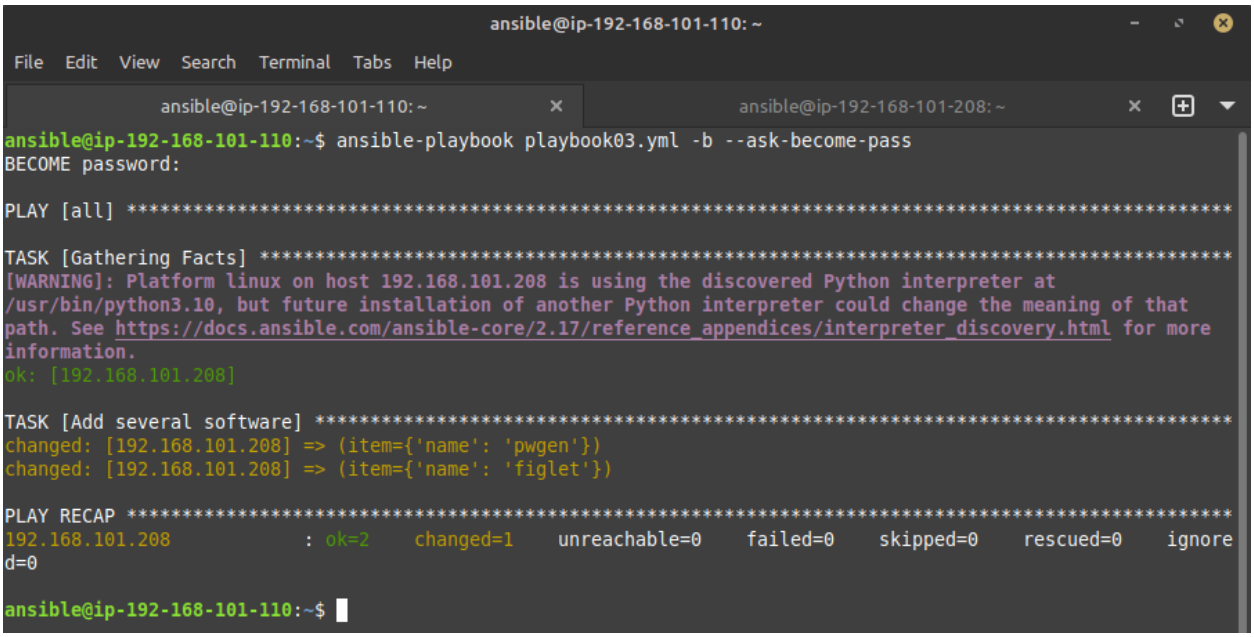


ii. Creating playbook03.yml which install/verify multiple apps on host system:




Running playbook:

>> ansible-playbook playbook03.yml -b –ask-become-pass



## Extra features of a Playbook:

1. **Condition (when):**

The when statement in playbook is used to execute a task only when a certain condition is met. The condition is typically a logical expression or variable value.

**Example :**

```
- name: Install nginx if the condition is true
  ansible.builtin.yum:
    name: nginx
    state: present
  when: ansible_facts['os_family'] == 'RedHat'
```

2. **Loops:**

You can use the loop directive to iterate over a list of items and apply the same task for each item.

**Example :**

```
- name: Install multiple packages
  ansible.builtin.apt:
    name: "{{ item }}"
    state: present
  loop:
    - nginx
    - git
    - vim
```

### 3. Tags:

Tags allow you to run specific parts of a playbook by assigning tags to tasks or plays. This helps you execute only certain parts of the playbook rather than running everything.

**Example :**

```
tasks:
  - name: Install nginx
    ansible.builtin.yum:
      name: nginx
      state: present
    tags:
      - install
```

# Ansible Vault:

Ansible Vault is a feature that allows you to securely store and manage sensitive data, such as passwords, API keys, and private keys, within Ansible playbooks, inventory files, or variables. Vault ensures that this sensitive information is encrypted and can be easily decrypted when needed during playbook execution.

**How to Use Ansible Vault:**

1. **Creating an Encrypted File**

You can create an encrypted file using the ansible-vault create command. For example, to create an encrypted file called secrets.yml:

>> ansible-vault create secrets.yml

This command will prompt you for a password, which you will need to decrypt the file later.

**2. Viewing an Encrypted File**

To view the contents of an encrypted file, use the ansible-vault view command:

>> ansible-vault view secrets.yml

**3. Editing an Encrypted File**

If you need to edit an encrypted file, you can use the ansible-vault edit command:

>> ansible-vault edit secrets.yml

**4. Executing Encrypted File**

To run the playbook that includes a vault-encrypted file, you would provide the vault password using the --ask-vault-pass option.

>> ansible-playbook --ask-vault-pass secrets.yml