Installing and Running Ansible to Check System Uptime on a Remote Machine

Understanding Control Machine & Remote Machine in Ansible

In Ansible, we have two main types of machines:

Control Machine (Ansible Controller)
The Control Machine is where Ansible is installed and executed. It is responsible for managing and automating tasks on remote machines using SSH.

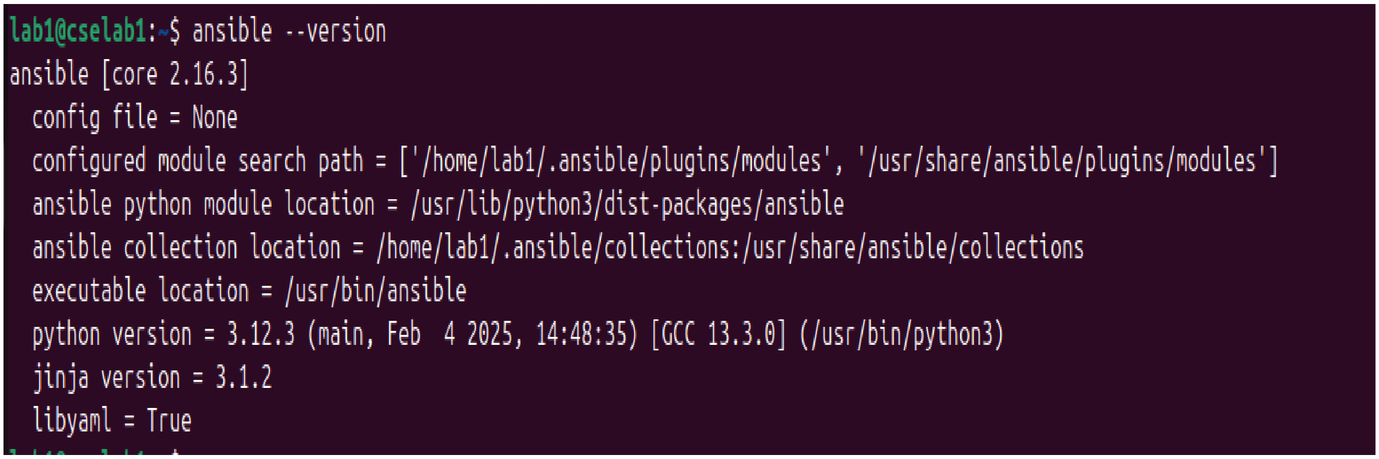
Remote Machine (Managed Node)
The Remote Machine (also called a "Managed Node") is the system that Ansible manages.
This is the machine where Ansible executes tasks, like checking uptime, installing software, or configuring settings.

Step 1: Install Ansible on the Control Machine

```
lab1@cselab1:~$ sudo apt update && sudo apt install ansible -y
```

Verify Installation

Check if Ansible is installed correctly: **ansible --version**



```
lab1@cselab1:~$ ansible --version
ansible [core 2.16.3]
  config file = None
  configured module search path = ['/home/lab1/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/lab1/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Feb  4 2025, 14:48:35) [GCC 13.3.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
```

If you get config file = None, execute below steps.

Create a Global Configuration File

If you want Ansible to use this configuration system-wide, create the file in /etc/ansible/:

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

```
sudo nano /etc/ansible/ansible.cf
```

Add the following content:

```
[defaults]

inventory = inventory.ini

remote_user = lab1

host_key_checking = False

retry_files_enabled = False
```

Save and exit (CTRL + X, then Y, then Enter).

Execute: **ansible --version**

```
ansible [core 2.16.3]

  config file = /etc/ansible/ansible.cfg

  python version = 3.x.x
```

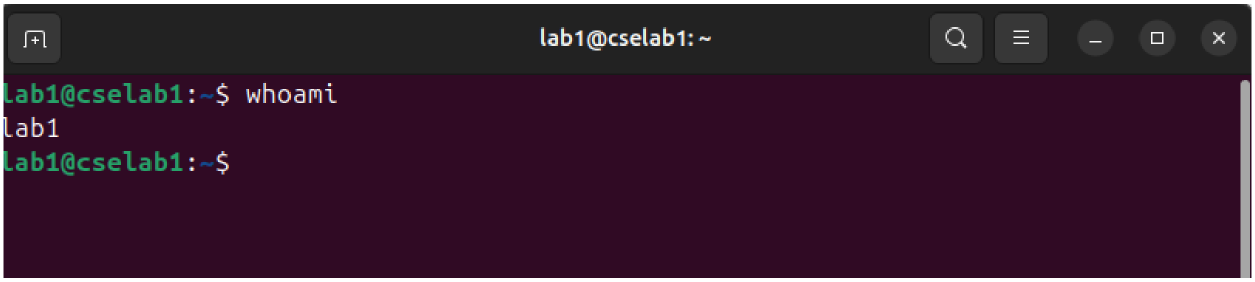
Step 2: Configure SSH Access to the Remote Machine

Before running Ansible,set up SSH access between the control and remote machines.

- Find Your Username and Remote Machine IP:

Execute the below command on your control Machine

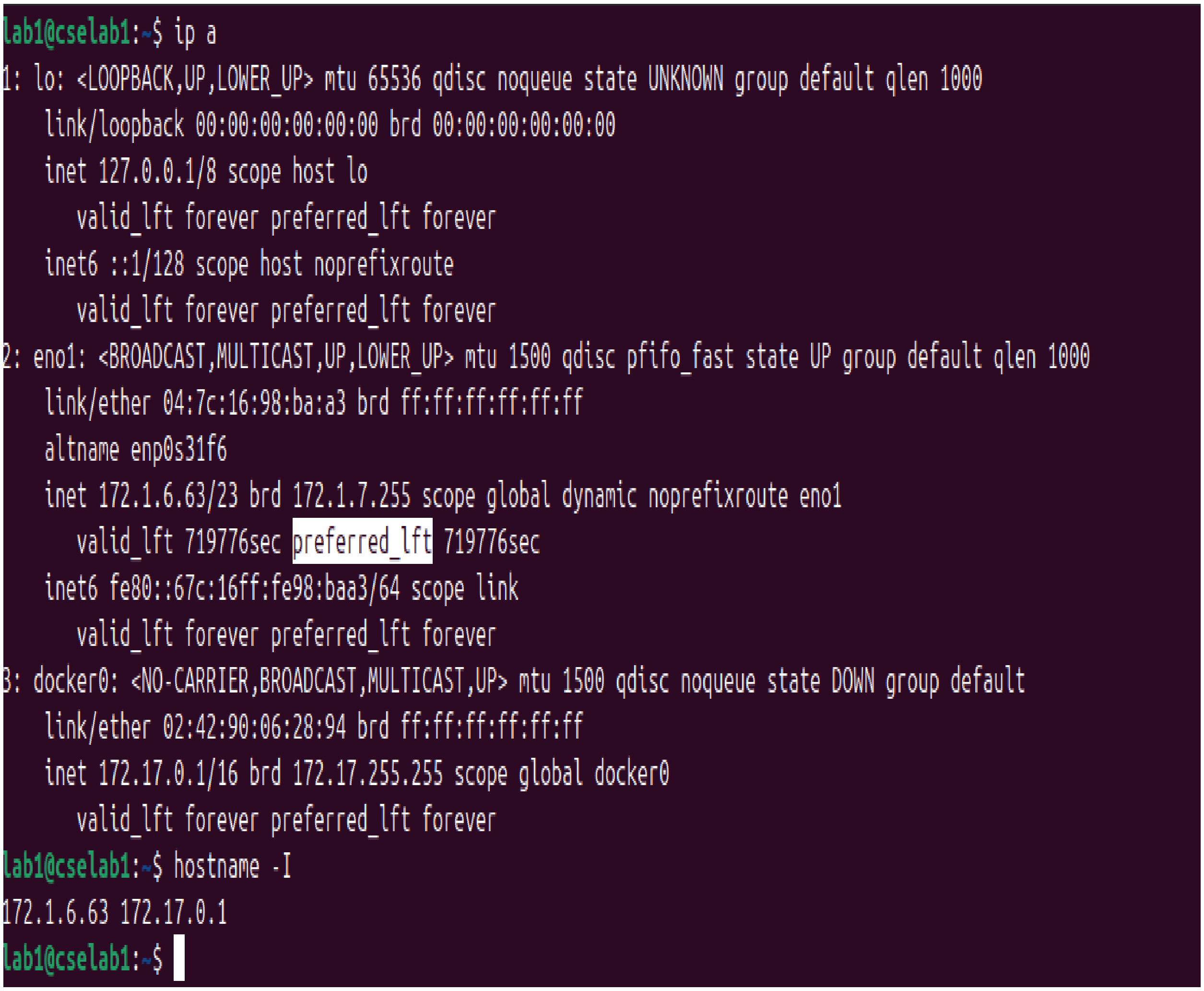
```
whoami
```



```
lab1@cselab1:~$ whoami
lab1
lab1@cselab1:~$
```

This shows your **username** (e.g., lab1).

On the remote machine, find the IP:
ip a



```
lab1@cselab1:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 04:7c:16:98:ba:a3 brd ff:ff:ff:ff:ff:ff
    altname enp0s31f6
    inet 172.1.6.63/23 brd 172.1.7.255 scope global dynamic noprefixroute eno1
       valid_lft 719776sec preferred_lft 719776sec
    inet6 fe80::67c:16ff:fe98:baa3/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:90:06:28:94 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
lab1@cselab1:~$ hostname -I
172.1.6.63 172.17.0.1
lab1@cselab1:~$
```

Here, **172.1.6.63** is the remote machine's IP.

Generate SSH Key (On Control Machine)

ssh-keygen

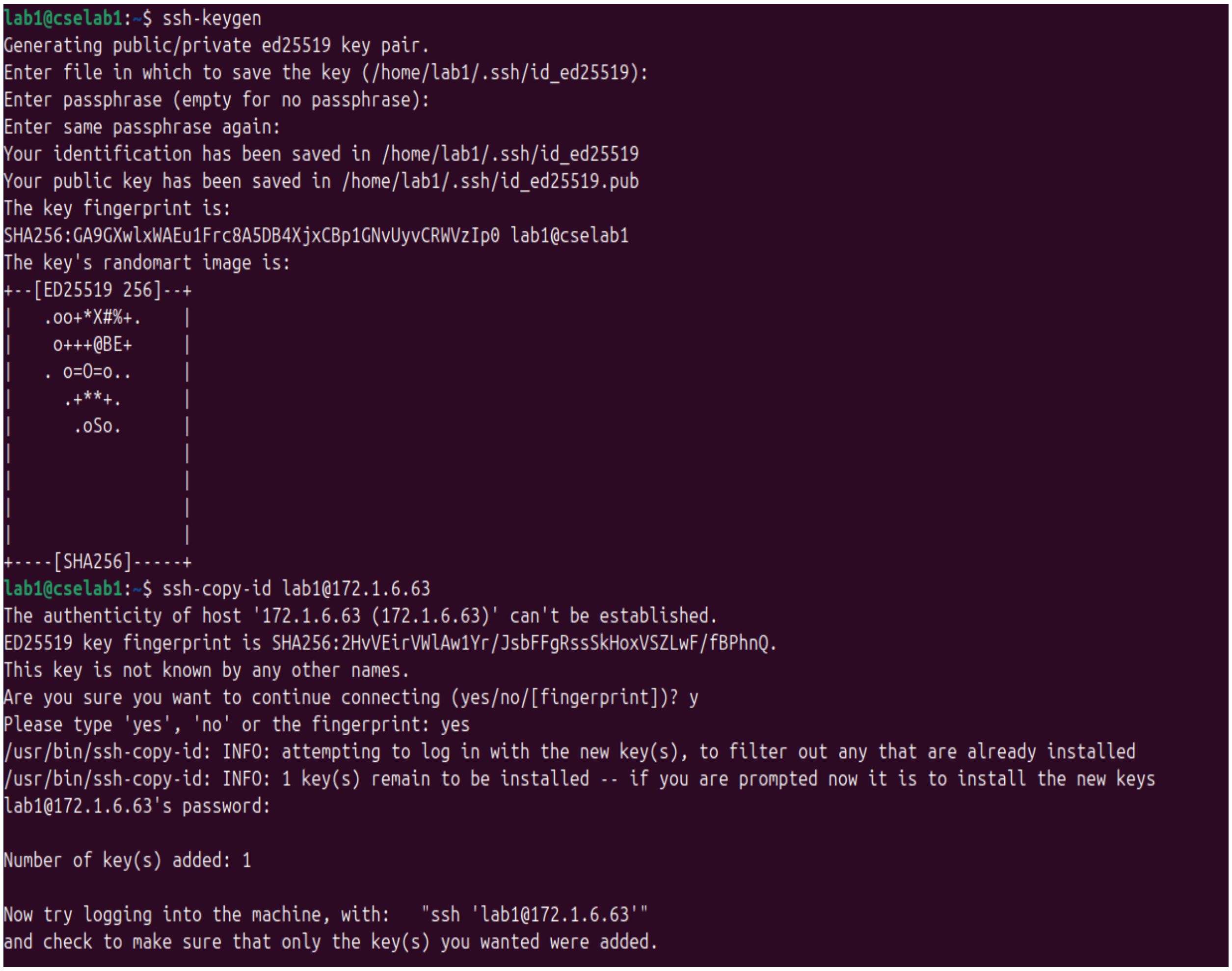
Press **Enter** to accept the default location (~/.ssh/id_ed25519).
Leave the passphrase **empty** (press Enter twice).

Copy the SSH Key to the Remote Machine

Replace lab1 and 172.1.6.63 with your actual username and IP:

ssh-copy-id lab1@172.1.6.63

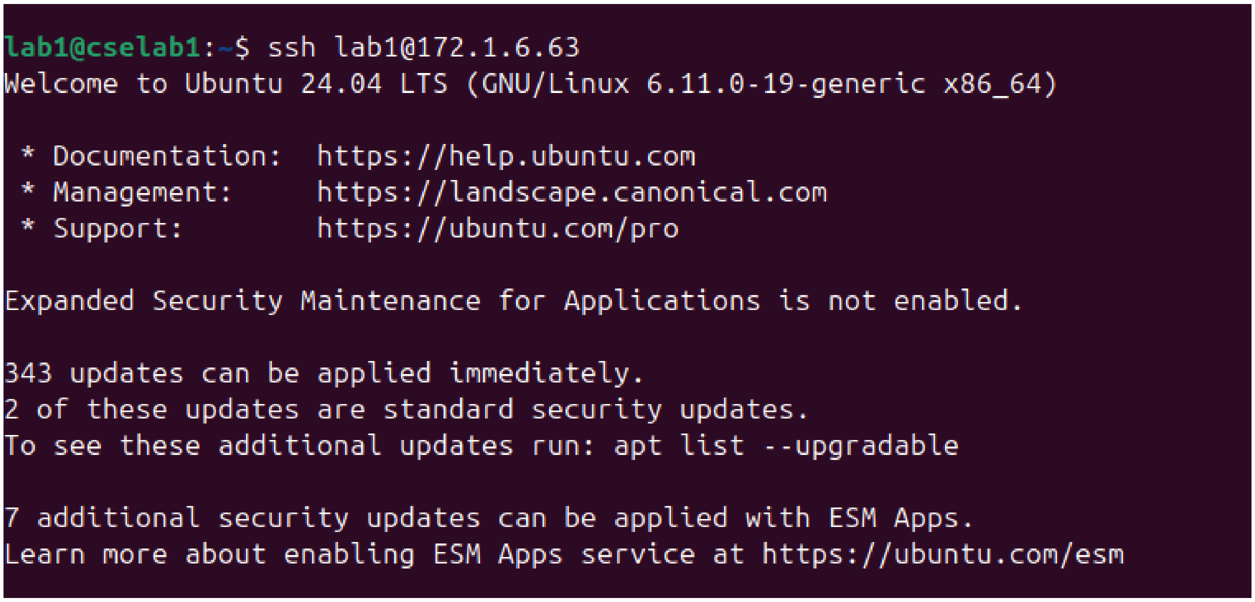
Enter the **remote machine's password** when prompted.
Once completed, SSH authentication will be **passwordless**



```
lab1@cselab1:~$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/lab1/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/lab1/.ssh/id_ed25519
Your public key has been saved in /home/lab1/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:GA9GXwlxWAEu1Frc8A5DB4XjxCBp1GNvUyvCRWVzIp0 lab1@cselab1
The key's randomart image is:
+--[ED25519 256]--+
|   .oo+*X#%+.    |
|    o+++@BE+     |
|   . o=O=o..     |
|     .+**+.      |
|      .oSo.      |
|                 |
|                 |
|                 |
|                 |
+----[SHA256]-----+
lab1@cselab1:~$ ssh-copy-id lab1@172.1.6.63
The authenticity of host '172.1.6.63 (172.1.6.63)' can't be established.
ED25519 key fingerprint is SHA256:2HvVEirVWlAw1Yr/JsbFFgRssSkHoxVSZLwF/fBPhnQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
lab1@172.1.6.63's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'lab1@172.1.6.63'"
and check to make sure that only the key(s) you wanted were added.
```



```
lab1@cselab1:~$ ssh lab1@172.1.6.63
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.11.0-19-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

343 updates can be applied immediately.
2 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

7 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm
```

Test SSH Connection

Now, verify that you can SSH into the remote machine **without entering a password**,

ssh lab1@172.1.6.63

If it logs in without asking for a password, SSH is set up correctly.
If SSH asks for a password, run on the remote machine.

```
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

Step 3: Create an Ansible Inventory File

Create the Inventory File: nano inventory.ini

Add the Following Content:

```
[servers]

remote_host ansible_host=172.1.6.63 ansible_user=lab1
```

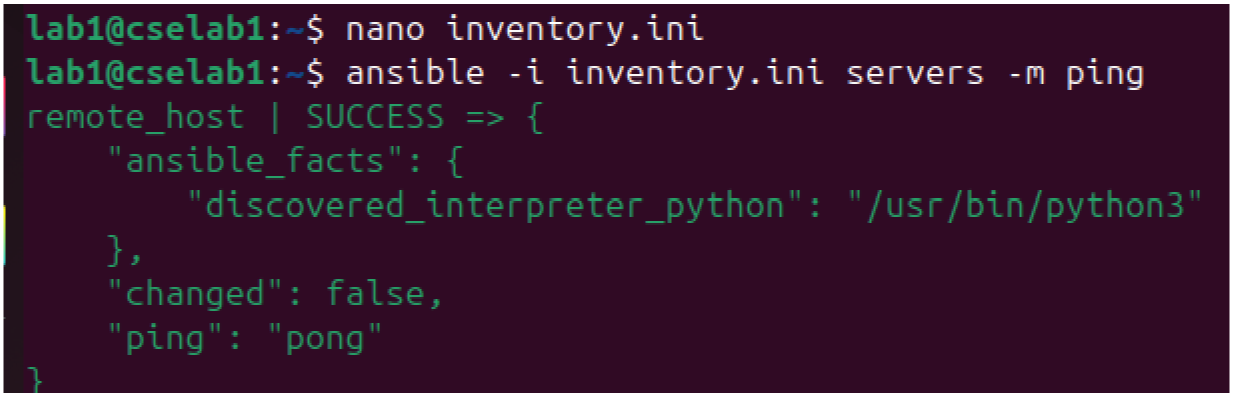
Save and exit (Press CTRL + X, then Y, then Enter).

Step 4: Test Ansible Connectivity

Before running a playbook, test whether Ansible can connect to the remote machine.

Run the Ping Test

ansible -i inventory.ini servers -m ping



```
lab1@cselab1:~$ nano inventory.ini
lab1@cselab1:~$ ansible -i inventory.ini servers -m ping
remote_host | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

This means Ansible successfully connected to the remote machine.

Step 5: Write an Ansible Playbook

Ansible playbooks are YAML files that define tasks.
Create the Playbook File: nano uptime_check.yml

Add the Following YAML Content

```
---
- name: Check System Uptime
  hosts: servers
  gather_facts: no
  tasks:
    - name: Run uptime command
      command: uptime
      register: uptime_output

    - name: Display uptime result
```

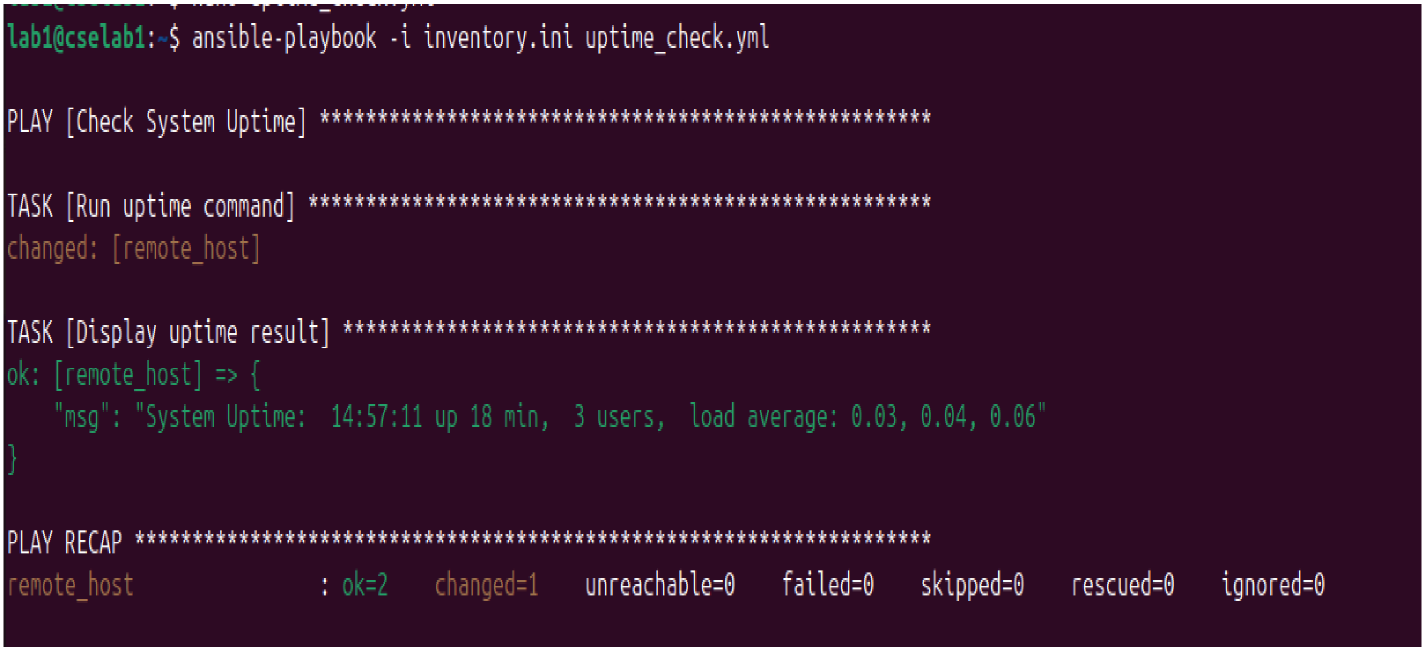
```
        debug:
          msg: "System Uptime: {{ uptime_output.stdout }}"
```

Save and exit (CTRL + X, then Y, then Enter).

Step 6: Run the Ansible Playbook

Execute the playbook with:

```
ansible-playbook -i inventory.ini uptime_check.yml
```



```
lab1@cselab1:~$ ansible-playbook -i inventory.ini uptime_check.yml

PLAY [Check System Uptime] *****************************************************

TASK [Run uptime command] ******************************************************
changed: [remote_host]

TASK [Display uptime result] ***************************************************
ok: [remote_host] => {
    "msg": "System Uptime:  14:57:11 up 18 min,  3 users,  load average: 0.03, 0.04, 0.06"
}

PLAY RECAP *********************************************************************
remote_host                : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

```
TASK [Display uptime result]
***************************************************

ok: [remote_host] => {

    "msg": "System Uptime:  14:57:11 up 18 min,  3 users,  load average: 0.03, 0.04, 0.06"

}
```

TASK [Display uptime result]:

- This task uses the debug module to show the output of the uptime command.

ok: [remote_host]:

- This means the task executed without issues.

"msg": "System Uptime: ...":

- This is the actual result of the uptime command from your remote machine.
- 14:57:11 → The current time on the remote machine.
- **up 18 min** → The machine has been running for 18 minutes since its last reboot.
- 3 users → Three users are currently logged into the system.