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.

**Before proceeding,check whether the SSH is installed your system:**

sudo apt update

sudo apt install -y openssh-server

### Enable and Start SSH Service

Once installed, enable and start the SSH service:

sudo systemctl enable ssh

sudo systemctl start ssh

**3. Verify SSH Service is Running**

sudo systemctl status ssh

# Step 1: Install Ansible on the Control Machine

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

E: The repository 'https://apt.kubernetes.io kubernetes-xenial Release' does not have a Release file. N: Updating from such a repository can't be done securely, and is therefore disabled by default. N: See apt-secure(8) manpage for repository creation and user configuration details. W: https://repo.mongodb.org/apt/ubuntu/dists/focal/mongodb-org/4.4/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details. W: GPG error: https://repo.mongodb.org/apt/ubuntu noble/mongodb-org/8.0 InRelease: The following signatures couldn't be verified because the public key is not available: NO\_PUBKEY 41DE058A4E7DCA05 E: The repository 'https://repo.mongodb.org/apt/ubuntu noble/mongodb-org/8.0 InRelease' is not signed. what is this error

If you get this error please execute the below following commands:

1. Remove Kubernetes Repository and Packages

# Remove the repository

sudo rm -f /etc/apt/sources.list.d/kubernetes.list

# Uninstall Kubernetes packages

sudo apt remove --purge -y kubelet kubeadm kubectl

sudo apt autoremove -y

2. Remove MongoDB Repository and Packages

# Remove MongoDB repository

sudo rm -f /etc/apt/sources.list.d/mongodb\*.list

# Remove old GPG keys (if present)

sudo rm -f /etc/apt/trusted.gpg.d/mongodb\*.gpg

sudo rm -f /usr/share/keyrings/mongodb-server-keyring.gpg

# Uninstall MongoDB packages

sudo apt remove --purge -y mongodb-org

sudo apt autoremove -y

3. Clean Up and Update System

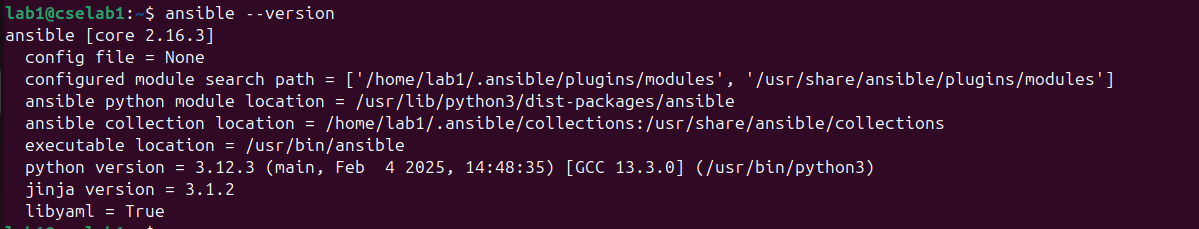
sudo apt update && sudo apt autoremove -y && sudo apt autoclean

Once done with the removing the packages, once again execute the command below:

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

## Verify Installation

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



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.cfg 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]

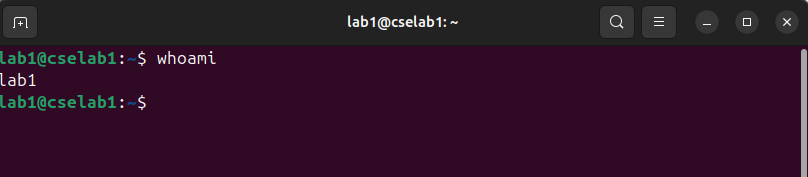
config file = /etc/ansible/ansible.cfg python version = 3.x.x

# Step 2: Configure SSH Access to the Remote Machine

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

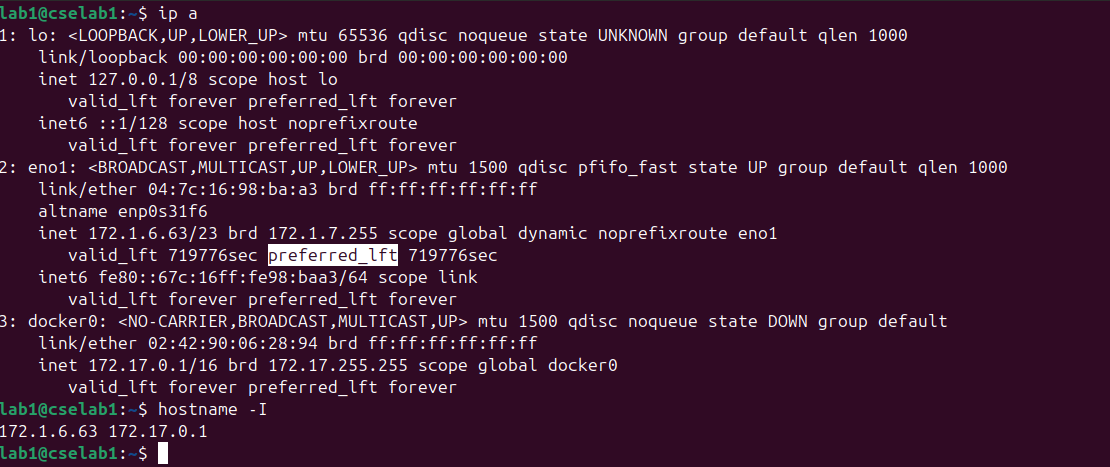
* Find Your Username and Remote Machine IP:

Execute the below command on your control Machine whoami



This shows your **username** (e.g., lab1). On the remote machine, find the IP:

ip a



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).

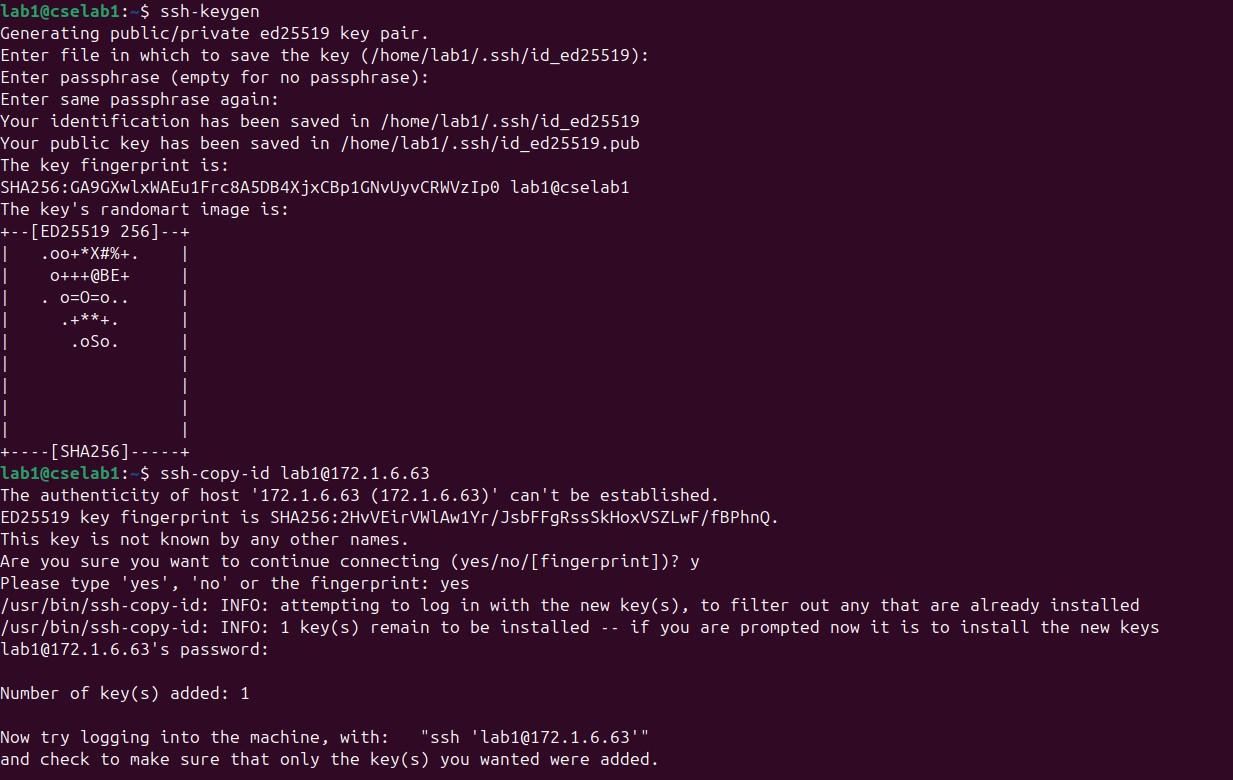
## Copy the SSH Key to the Remote Machine

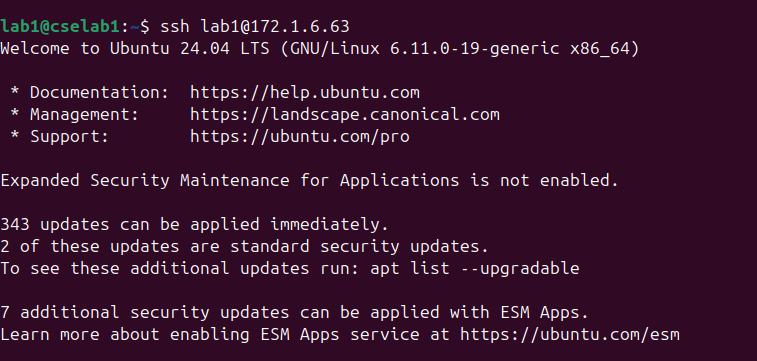
Replace lab1 and 172.1.6.63 with your actual username and IP:

**ssh-copy-id** [**lab1@172.1.6.63**](mailto:lab1@172.1.6.63)

Enter the **remote machine’s password** when prompted.

Once completed, SSH authentication will be **passwordless**





## Test SSH Connection

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

**ssh** [**lab1@172.1.6.63**](mailto: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]

172.1.58.114 ansible\_user=cselab3 ansible\_ssh\_pass=yourpassword ansible\_ssh\_common\_args='-o StrictHostKeyChecking=no'

**The ansible\_ssh\_pass=yourpassword refers to the password of the remote machine (172.1.58.114), not the control node.**

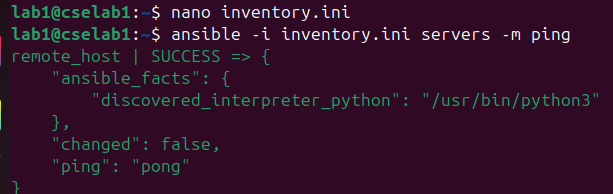
Save and exit (Press CTRL + X, then Y, then Enter).

# 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**



**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 }}"**

**debug:**

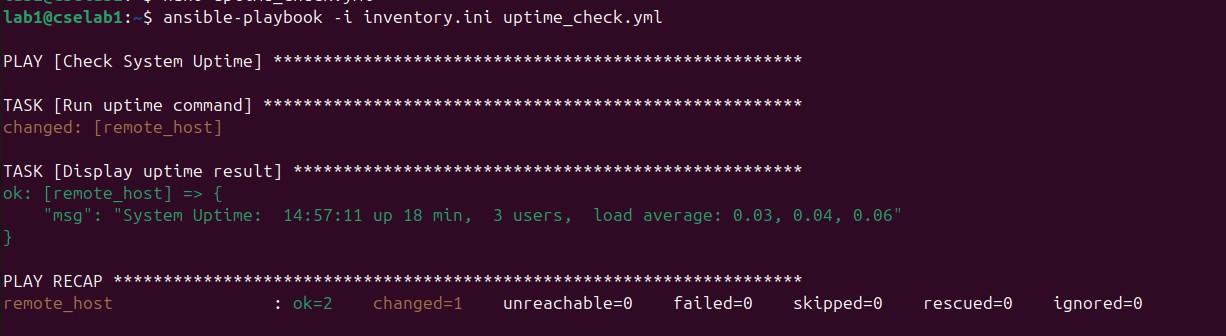
**msg: "System Uptime: {{ uptime\_output.stdout }}"**

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**



**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.