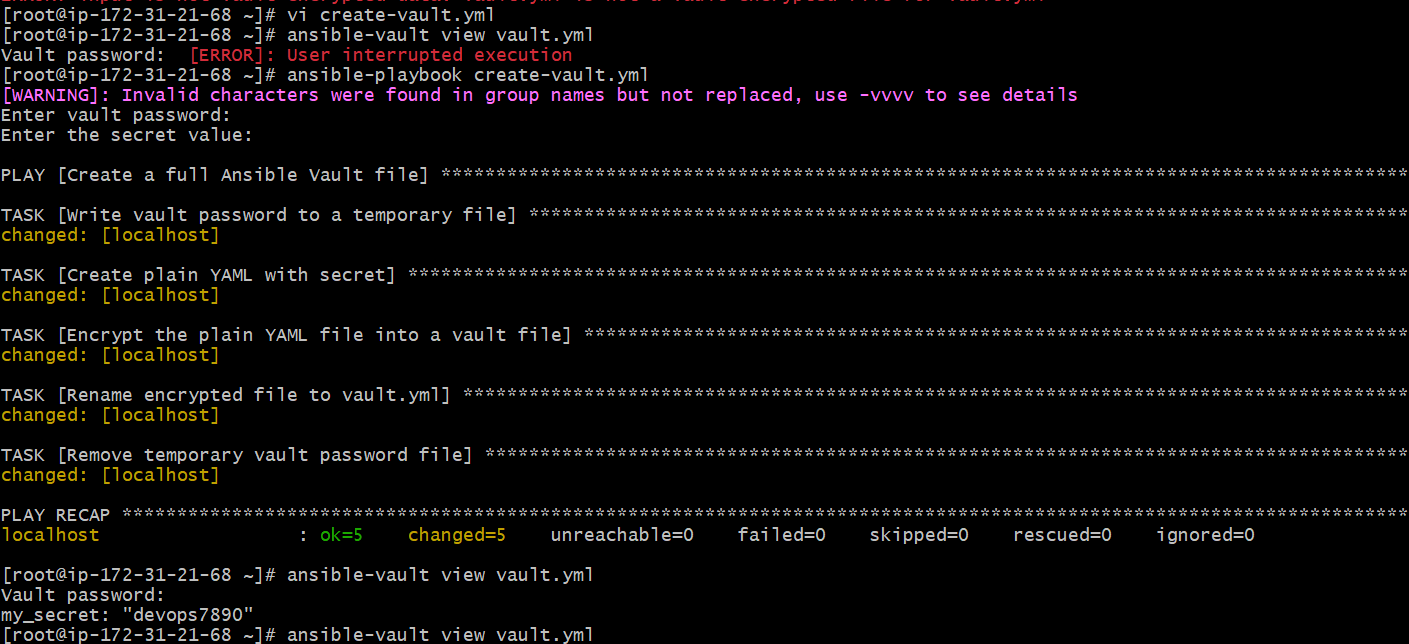
**Ansible-04**

**1.Create ansible playbook to create ansible vault.**

Create a create-vault.yml playbook

and run ansible-playbook create-vault.yml

****

We are creating an **Ansible playbook** whose purpose is to **create an Ansible Vault file**.

The playbook **asks us** for:

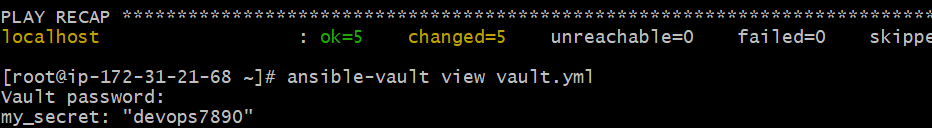
* A **vault password** (used to lock the file).
* The **secret value(s)** we want to protect.

The playbook then:

1. Encrypts that file using ansible-vault encrypt with the given password.
2. Produces a **vault.yml** file that is now encrypted.
3. Final outcome: we get a **password-protected vault file** (vault.yml) that safely stores our secret values

Now the result is a **proper vault file** that you can open with:

ansible-vault view vault.yml

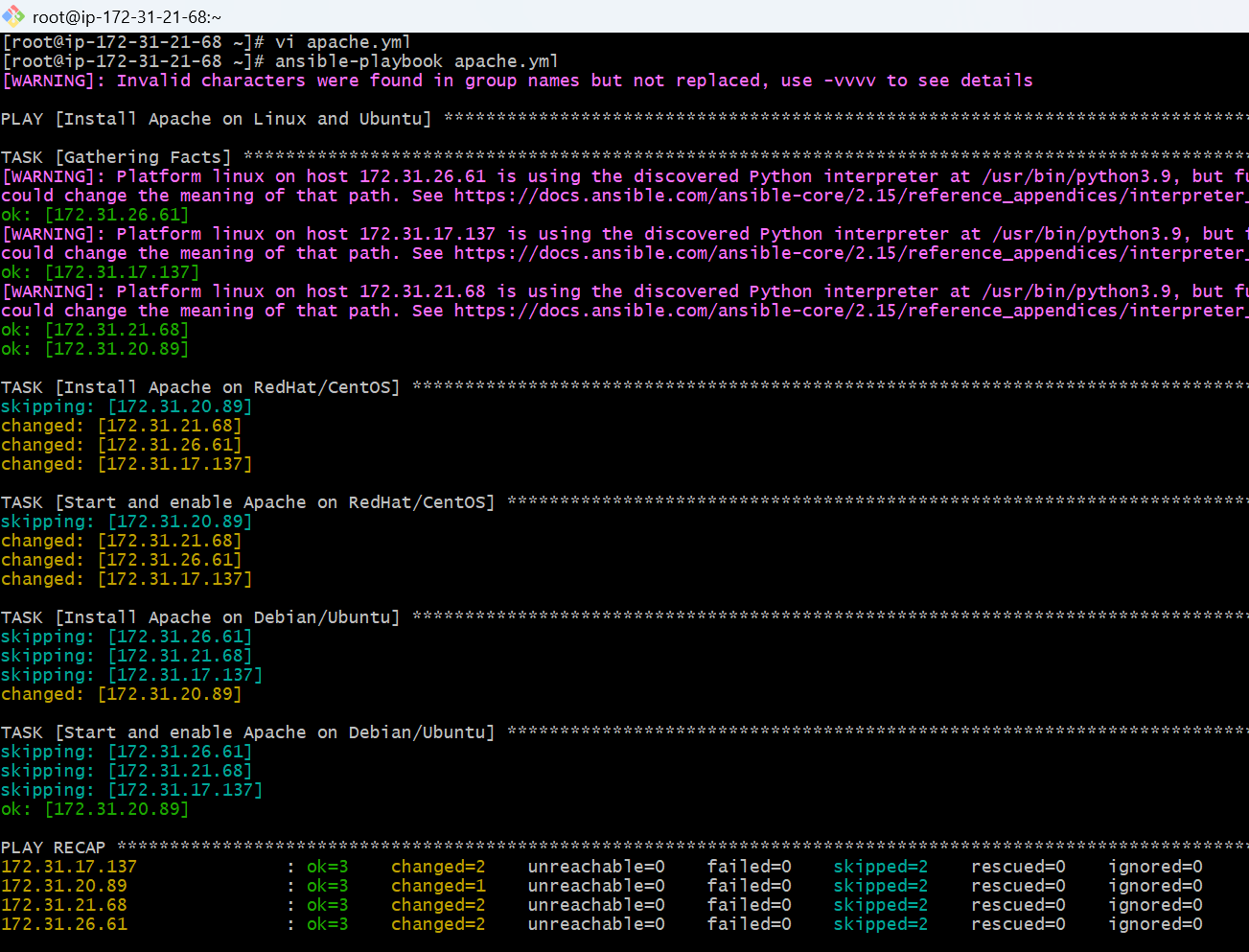


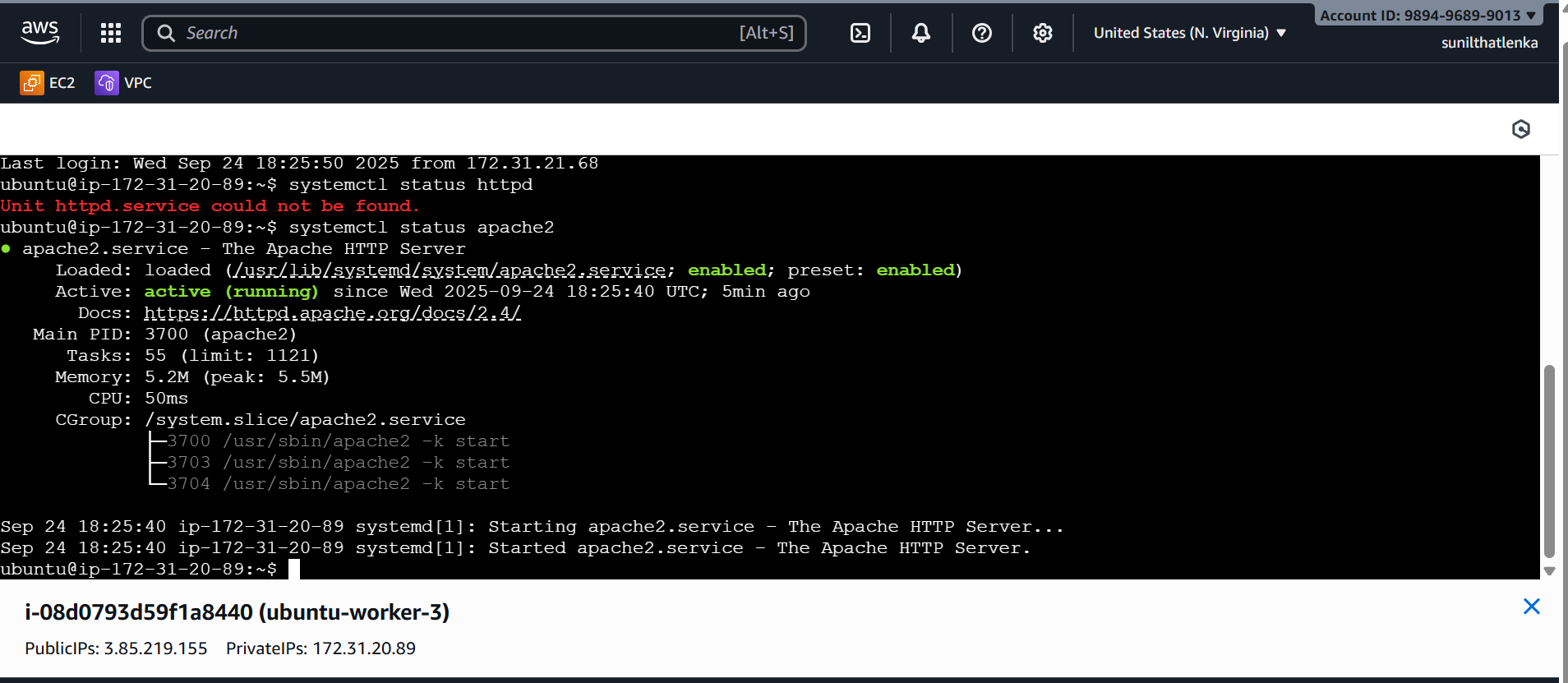
**2.Write a ansible playbook to install apache in linux and ubuntu machine by using when condition.**

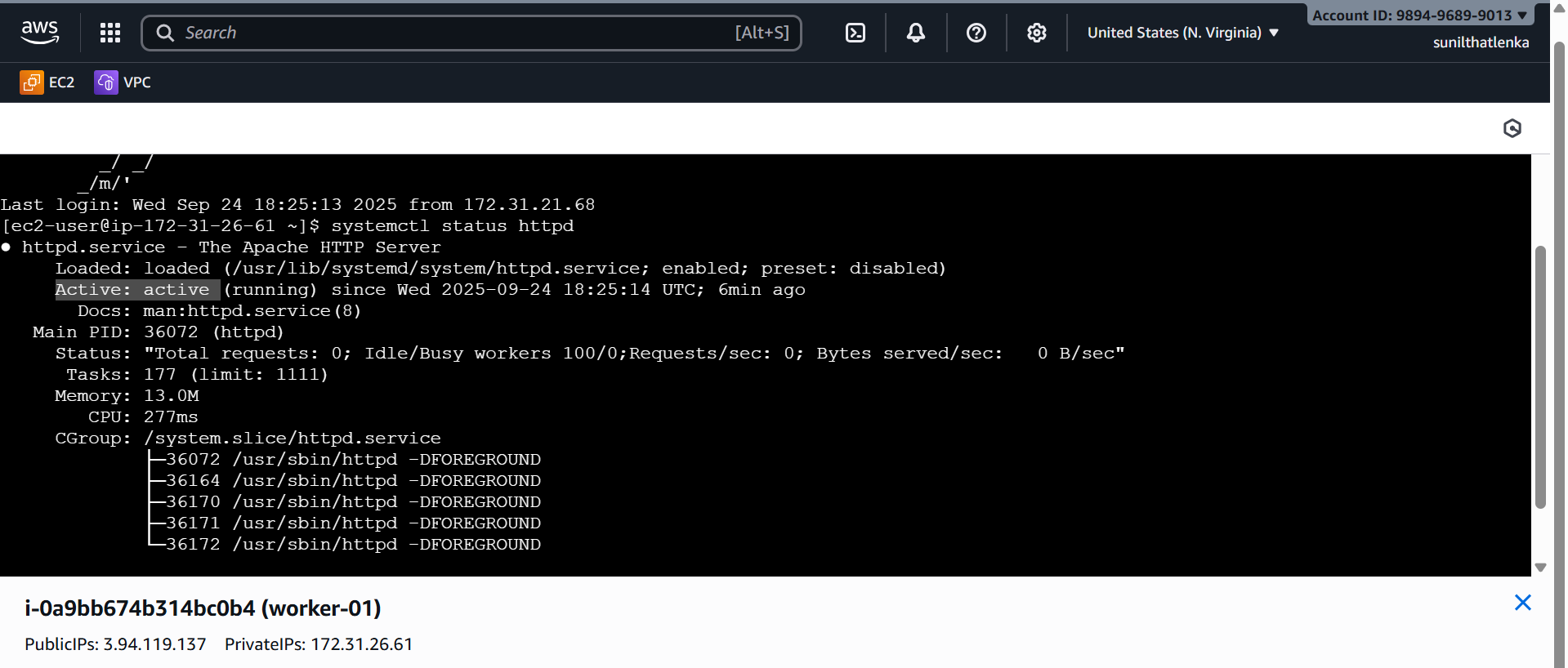
Here I have 3 ec2s in that 1 master, 2-worker nodes and 1 ubuntu worker-node configured with the controller Master node

I have created a playbook apache.yml and run 🡪 ansible-playbook apache.yml

It will install in all the worker nodes in ec2 and Ubuntu nodes parallely

****

****

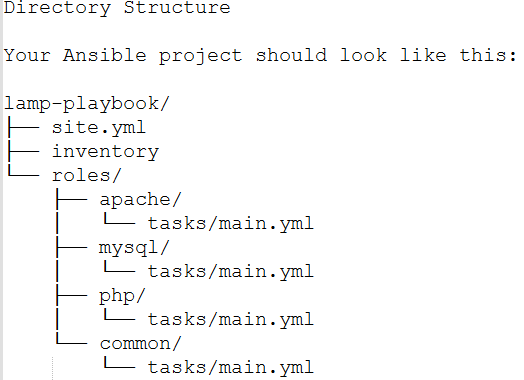
****

---  
- name: Install Apache on Linux and Ubuntu  
  hosts: all  
  become: yes  
  gather\_facts: yes   # required to detect ansible\_os\_family  tasks:  
    - name: Install Apache on RedHat/CentOS  
      yum:  
        name: httpd  
        state: present  
      when: ansible\_os\_family == "RedHat"    - name: Start and enable Apache on RedHat/CentOS  
      service:  
        name: httpd  
        state: started  
        enabled: yes  
      when: ansible\_os\_family == "RedHat"    - name: Install Apache on Debian/Ubuntu  
      apt:  
        name: apache2  
        state: present  
        update\_cache: yes  
      when: ansible\_os\_family == "Debian"    - name: Start and enable Apache on Debian/Ubuntu  
      service:  
        name: apache2  
        state: started  
        enabled: yes  
      when: ansible\_os\_family == "Debian"

**3.Create ansible playbook using roles to configure LAMP stack**

The **LAMP stack** is a popular set of open-source software used to build and run dynamic websites and web applications

1. **LAMP stack** = Linux + Apache + MySQL/MariaDB + PHP.  
   → Basically, it’s a web server setup.
2. **Ansible playbook** = A YAML file that describes the steps to install and configure software on servers.  
   → Instead of manually typing commands (yum install httpd, systemctl start httpd, etc.), you automate it with Ansible.
3. **Using roles** = Ansible best practice for organizing tasks.
   * Normally, a simple playbook could just have tasks listed.
   * But when you use **roles**, you split the playbook into logical parts (like one role for Apache, one for MySQL, one for PHP).
   * Each role has its own tasks, files, templates, variables. This makes the playbook modular and reusable.
4. **Write an Ansible playbook (site.yml)** that includes multiple roles.
5. **Create roles** for:
   * common → system updates or basic setup.
   * apache → install & configure Apache web server.
   * mysql → install & configure MySQL database.
   * php → install & configure PHP and extensions.
6. **Run the playbook** on a target server (like an EC2 instance or VM).
   * After it runs, the server should be ready with a working **LAMP stack**.
   * You should be able to open a browser and check Apache (index.html) and PHP (info.php).



# Steps to Configure LAMP Stack with Ansible Roles

### Step-1 Create project

mkdir lamp-playbook

cd lamp-playbook

### Step-2 Create roles structure

mkdir roles

cd roles

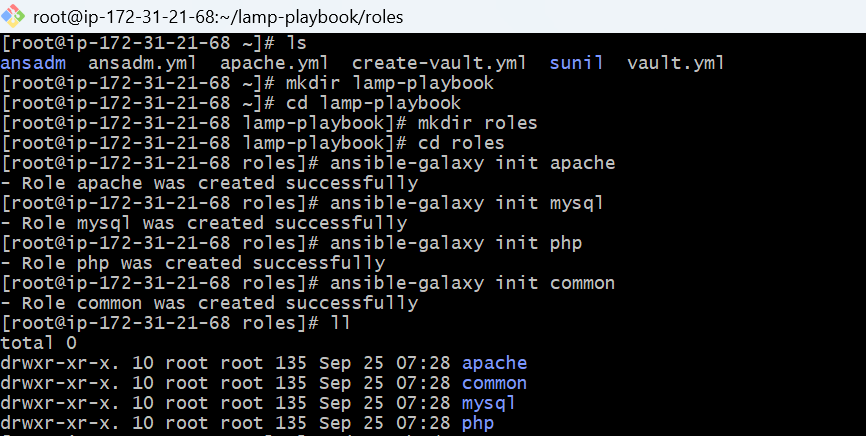
ansible-galaxy init common

ansible-galaxy init apache

ansible-galaxy init mysql

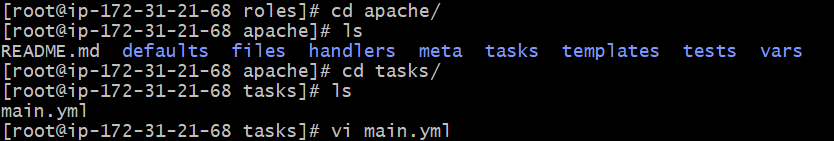
ansible-galaxy init php

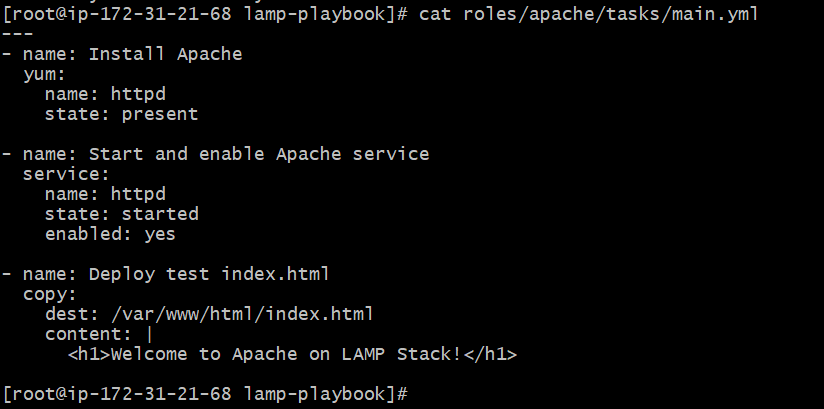
This generates the folder tree for each role.

****

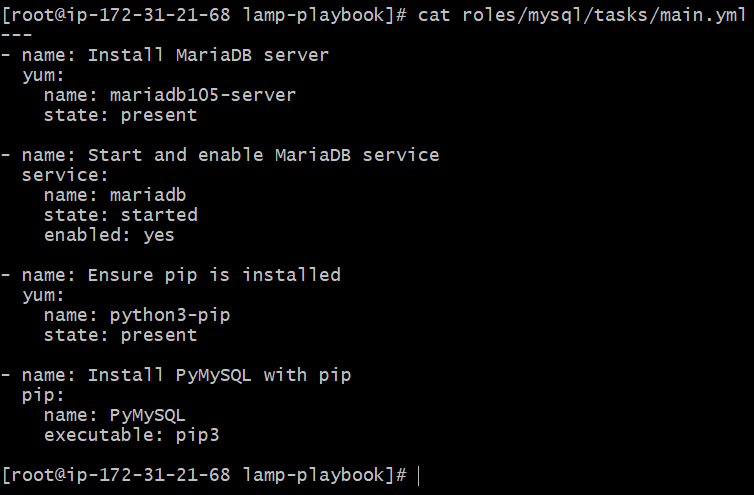
**Step -3 Configure each role**

**Apche(**roles/apache/tasks/main.yml)

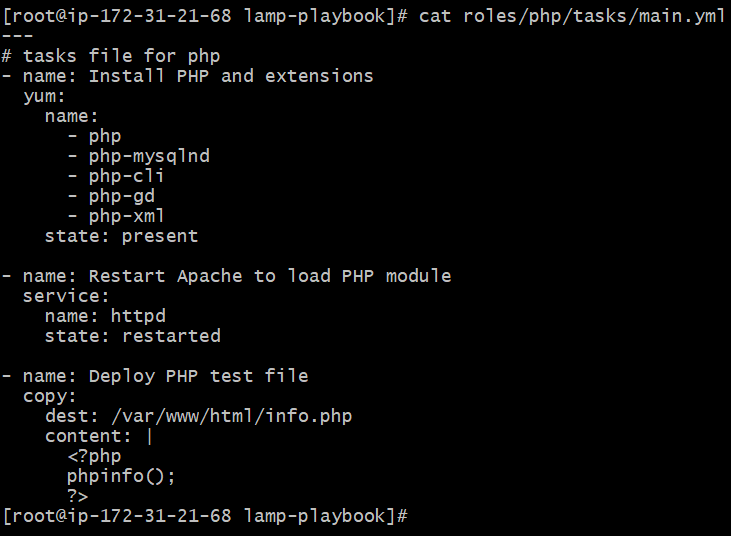
****

****

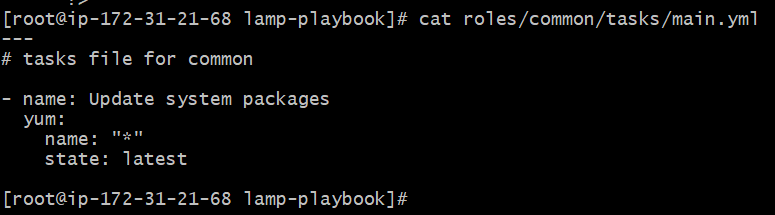
**My sql** (roles/mysql/tasks/main.yml)

****

php(roles/php/tasks/main.yml)

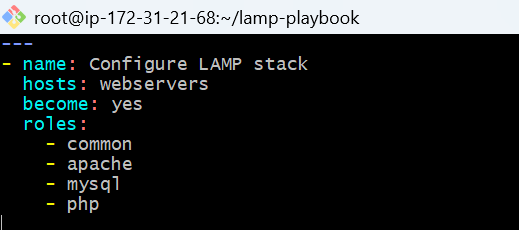


common(roles/common/tasks/main.yml)

****

**Step-4: Create main playbook file**

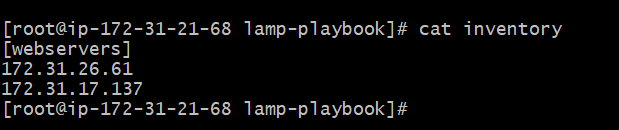
vi site.yml

****

### Step-5 Create inventory file

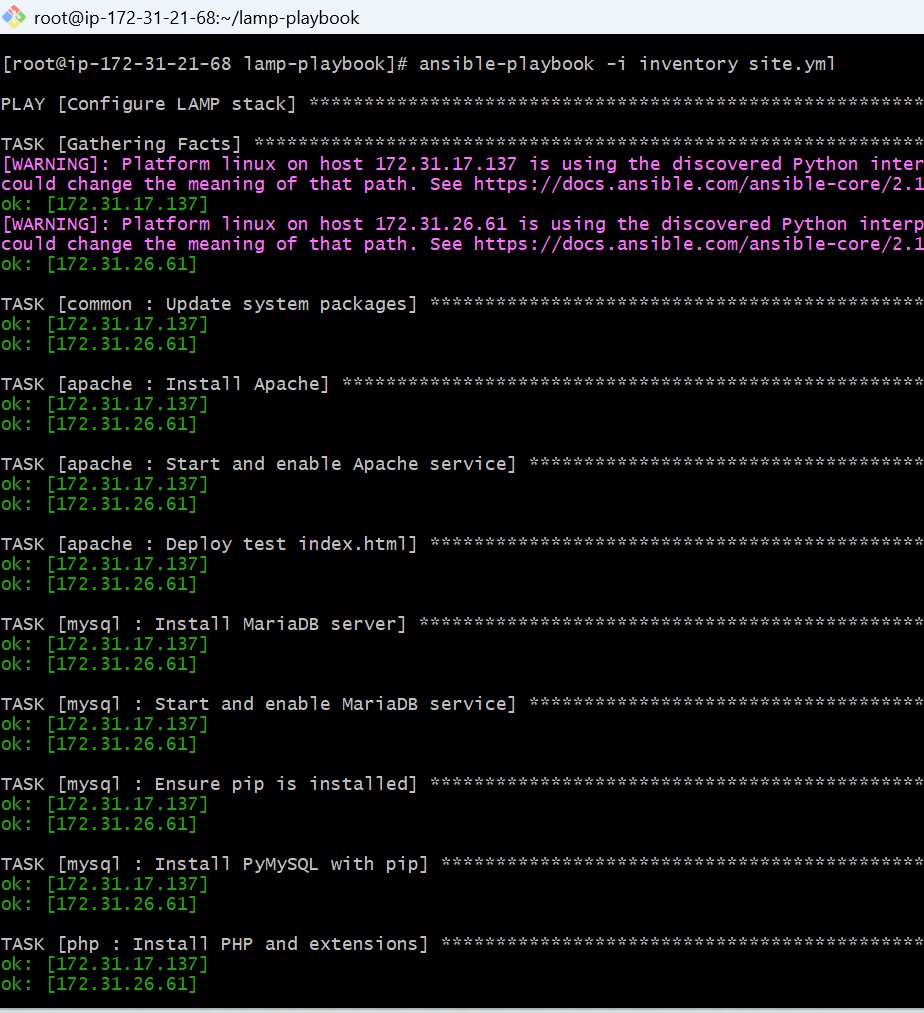
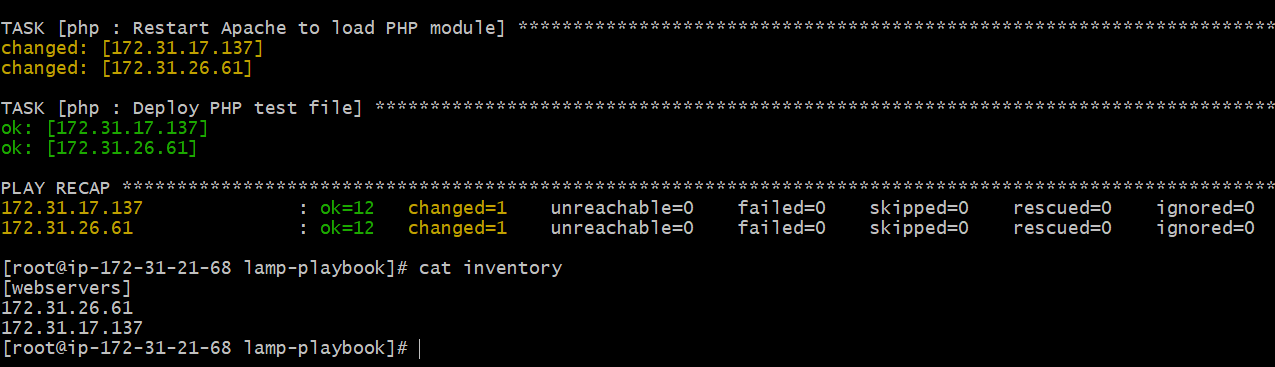
vi inventory

Add your worker- servers



**Step-6: run**

**ansible-playbook -i inventory site.yml**

**** ****

### ✅ In simple words:

You have to **automate the full setup of a LAMP server** using Ansible in a structured way (with roles). Instead of writing all tasks in one big file, you divide them into roles:

* **Playbook (site.yml)** → the “orchestra conductor”
* **Roles (apache, mysql, php, common)** → the “musicians” doing specific tasks

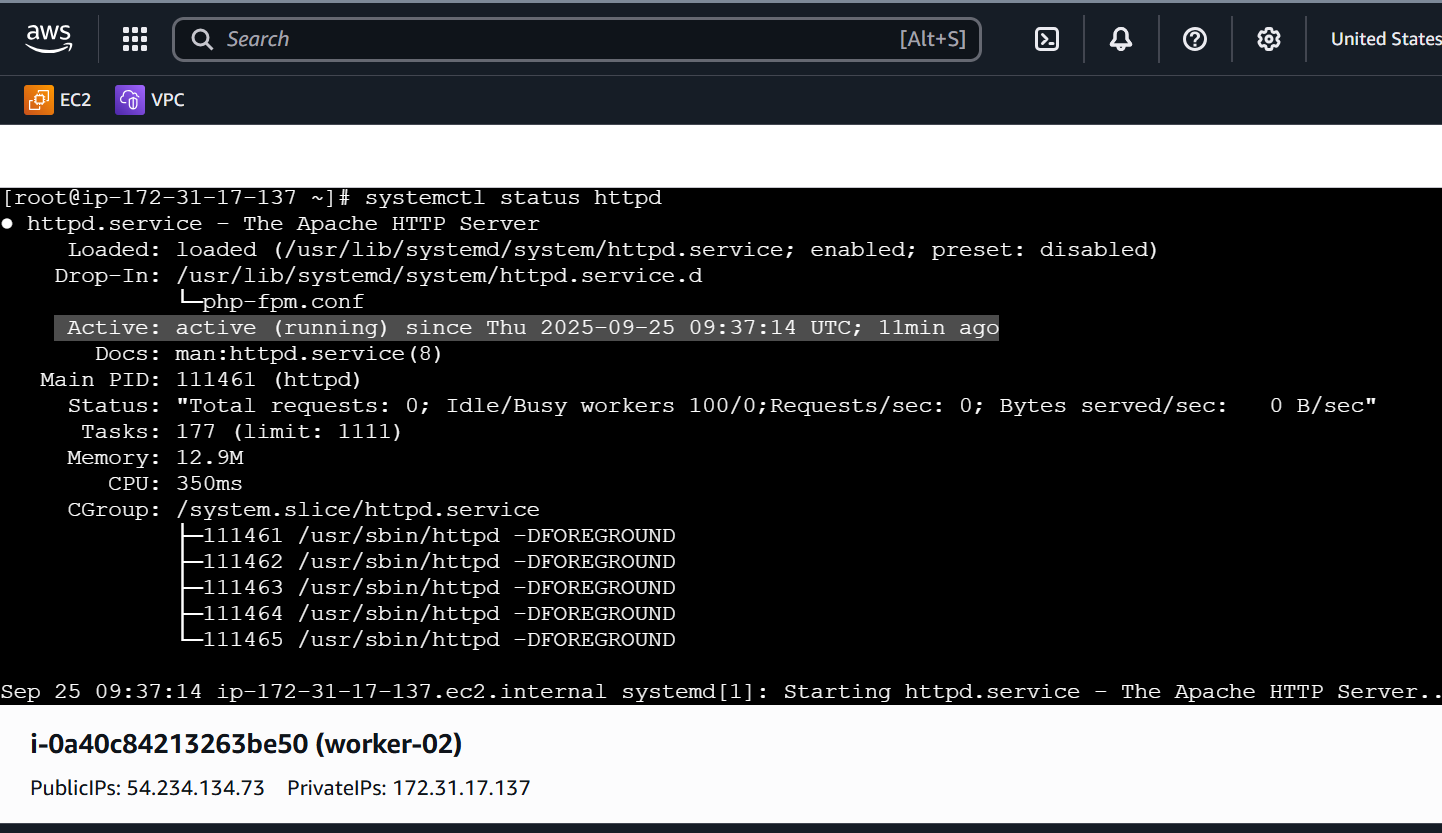
**Step-7 : Verify**  
now you need to **verify the LAMP stack** is really working.

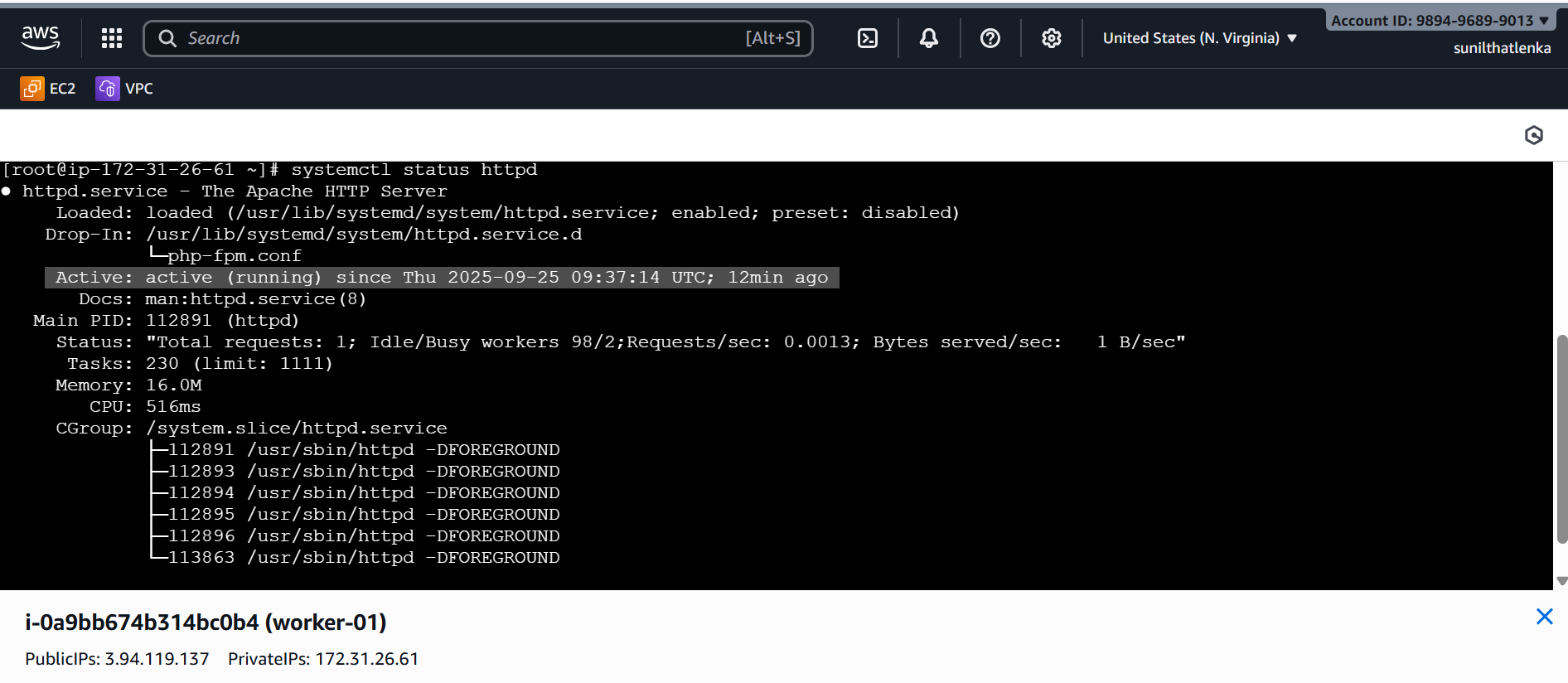
## 1. Check Apache (Web Server)

### On your server:

systemctl status httpd You should see **active (running)**.

### From your browser: Open: http://<your\_server\_ip>/

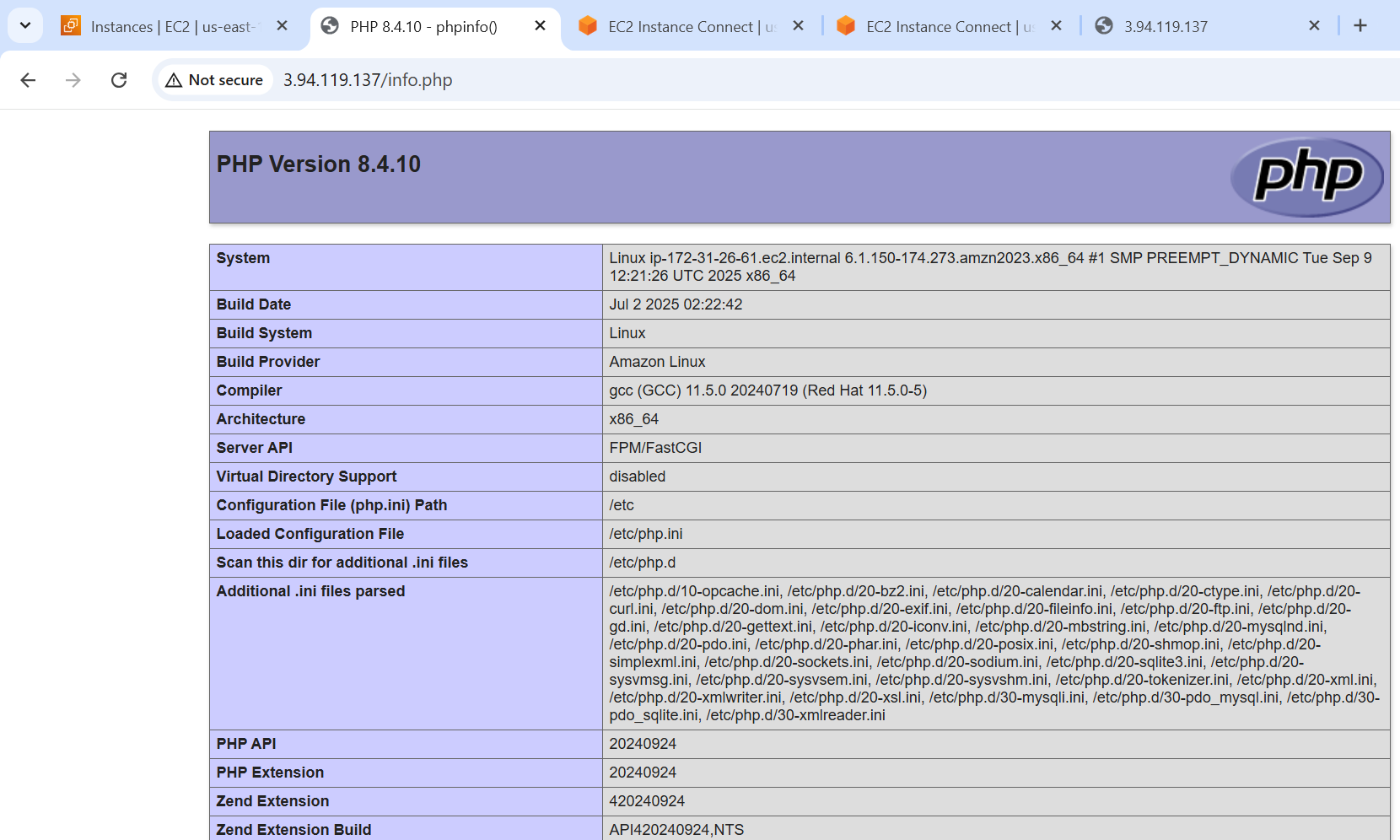


## 2 Check PHP

### In your browser:

* Open: http://<your\_server\_ip>/info.php
* You should see the **PHP Info page**

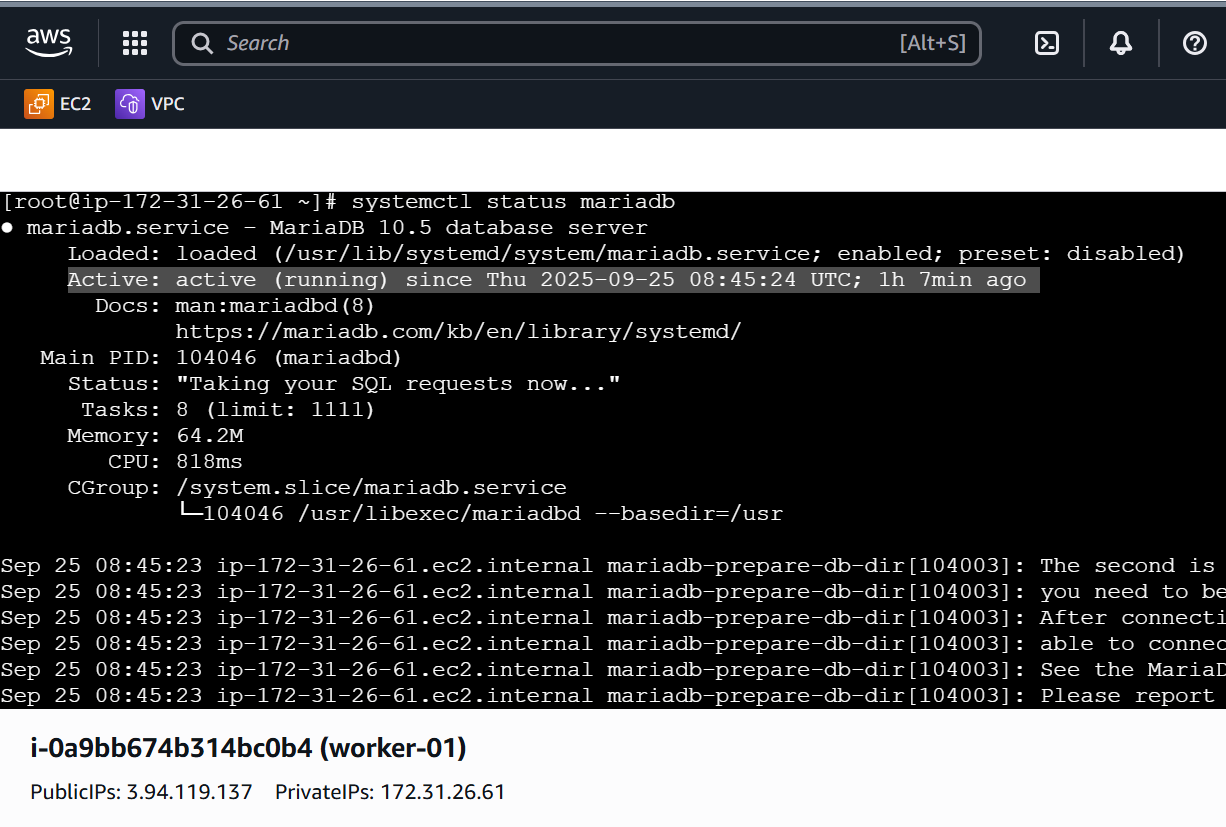


## 3 Check MariaDB/MySQL

### On your server:

systemctl status mariadb

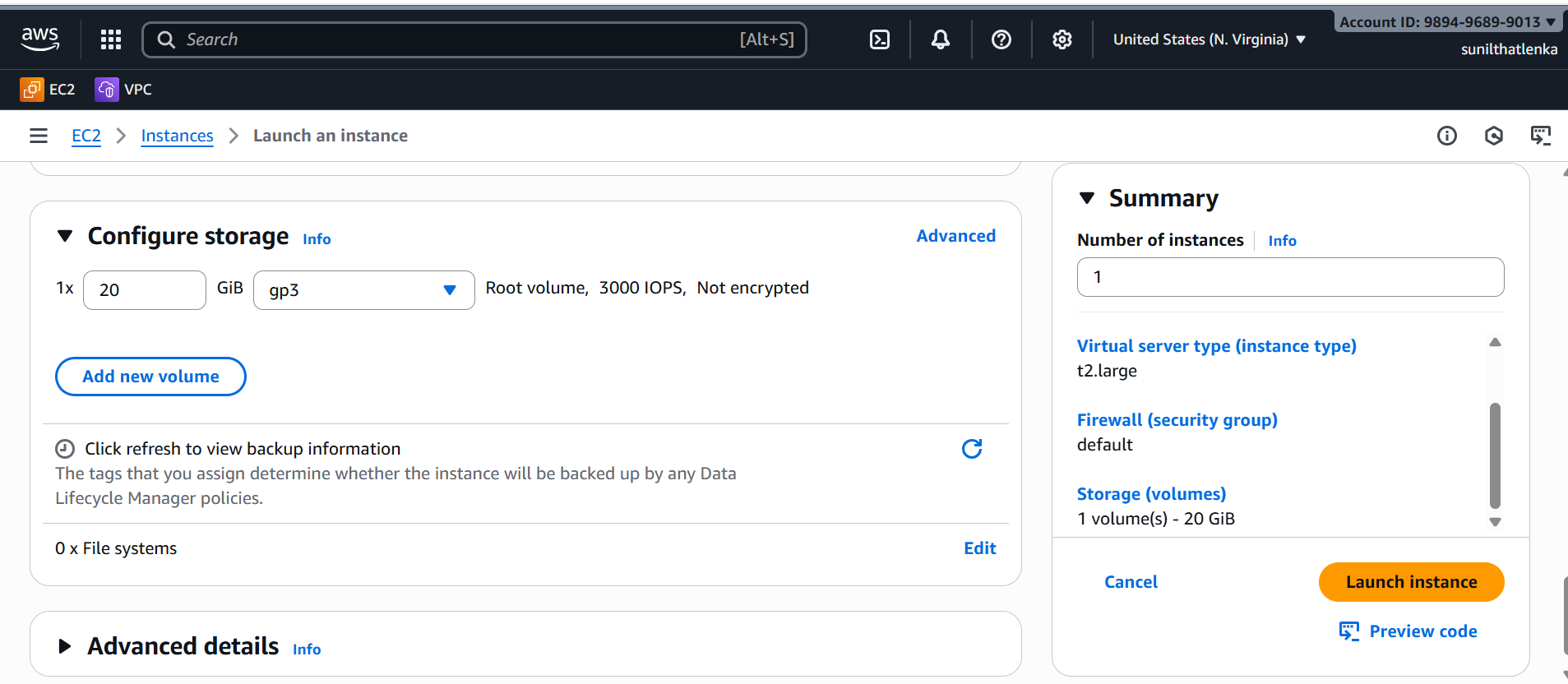
Should say **active (running)**.



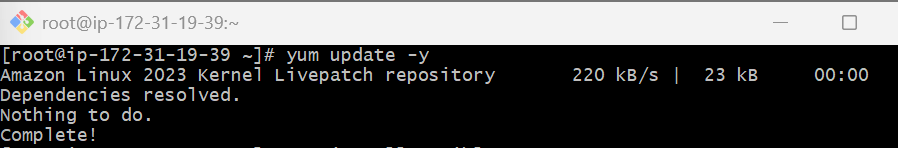
**4.Setup ansible AWX and explore the options.**

Step 1: Launch EC2 Instance

with t2 large and 20gb storage

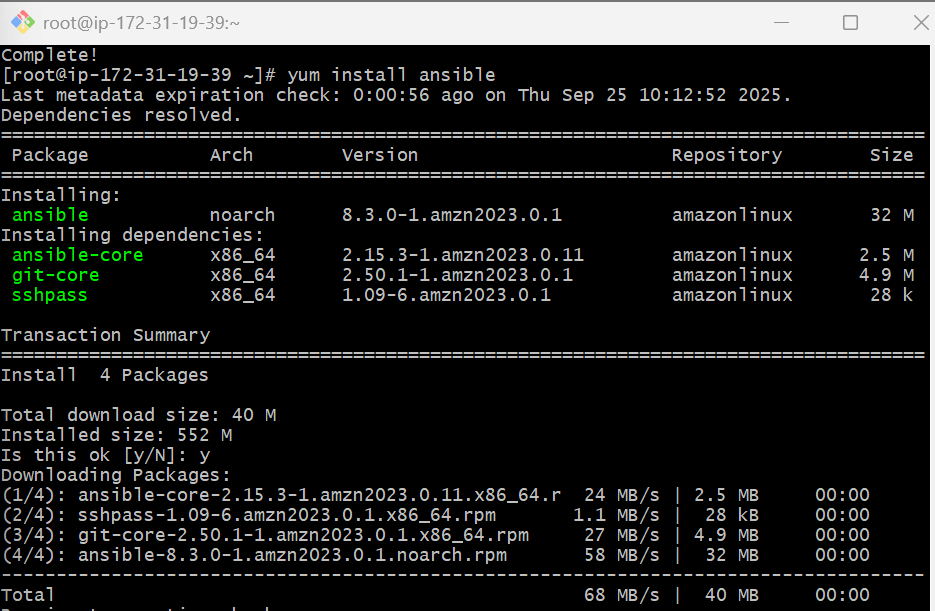
****

Step 2: Update System

****

**Step-3 install ansible**

**Yum intstall ansible**

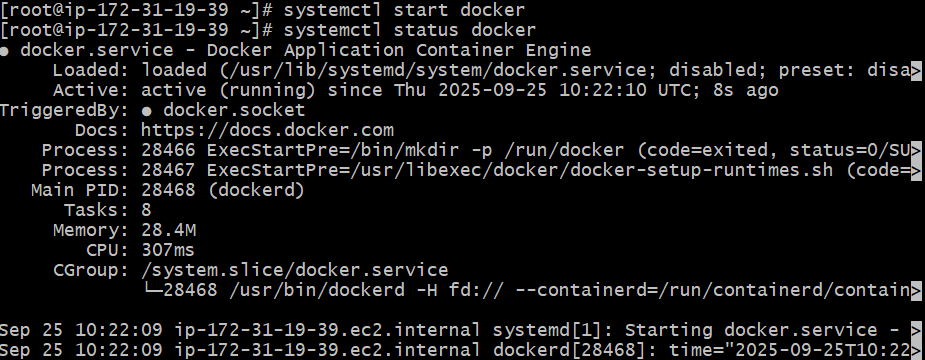
****

Step 4: Install Required Packages

yum install docker

systemctl start docker

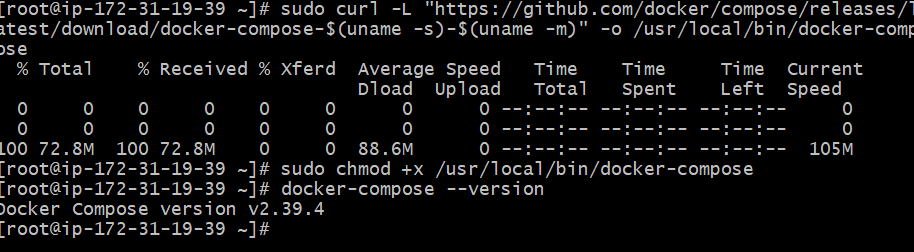
systemctl status docker

****

## ****Step 4: Install Docker Compose****

Amazon Linux 2 doesn’t come with Docker Compose by default:

docker-compose --version

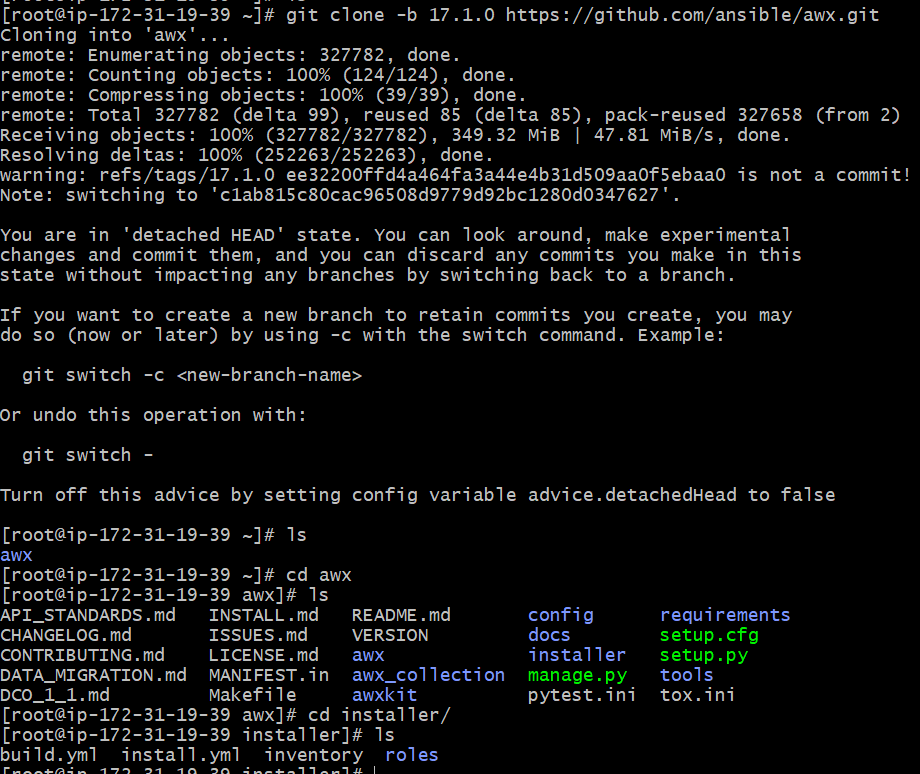
****

## ****Step 5: Clone AWX Repo****

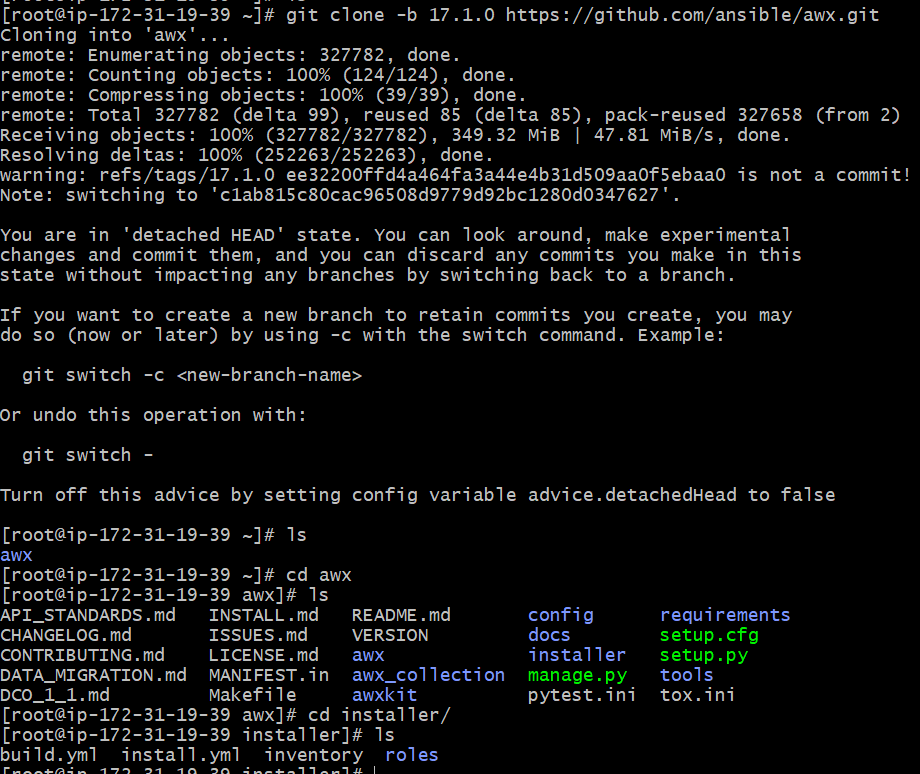
### Let’s now clone the AWX source code from GitHub

cd ~  
git clone -b 17.1.0 https://github.com/ansible/awx.git  
cd awx

cd

****

**Cd awx/installer**

****

## ****Step 6: Configure AWX Inventory****

Open the inventory file to set credentials and ports:

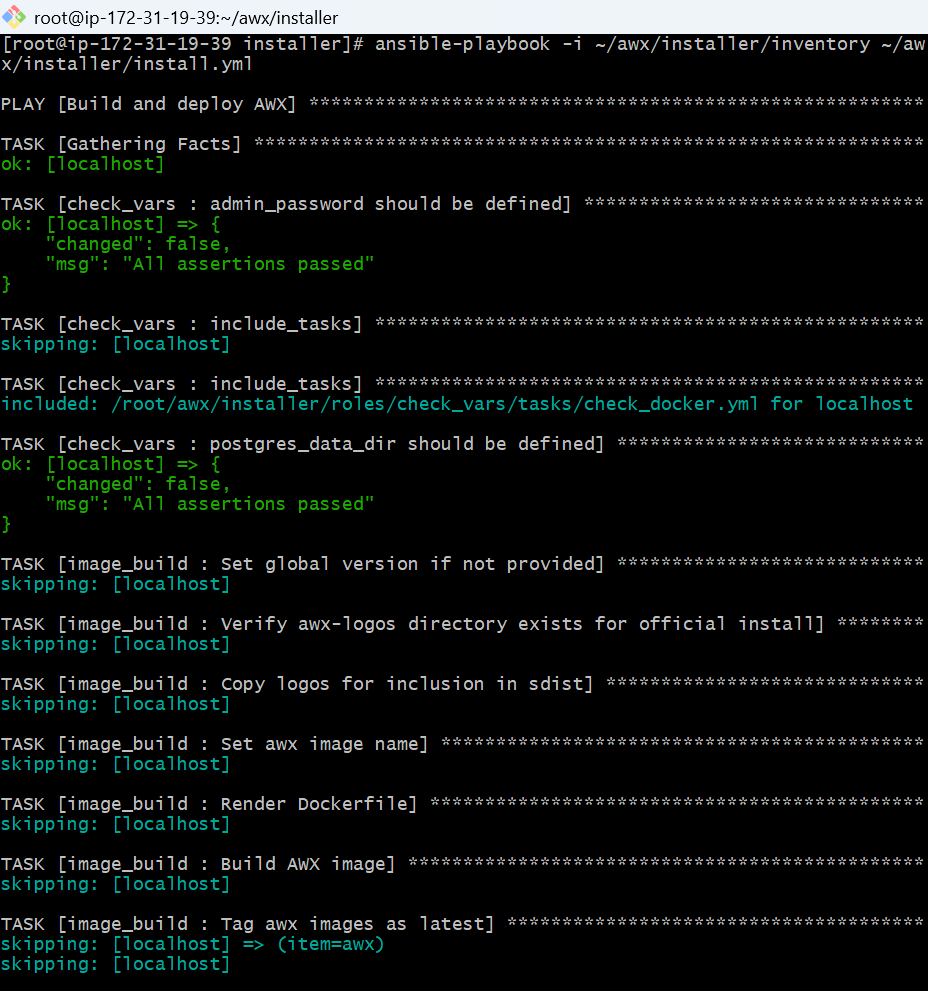
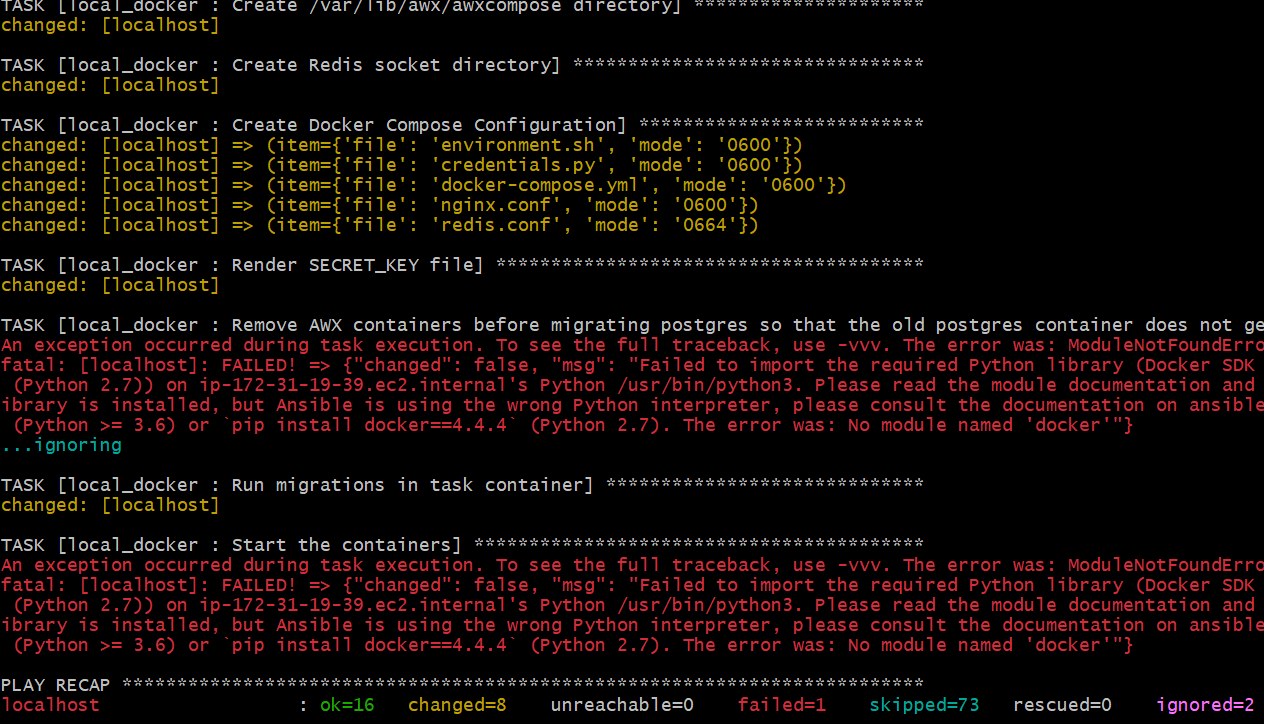
vi inventory

add these data in inventory

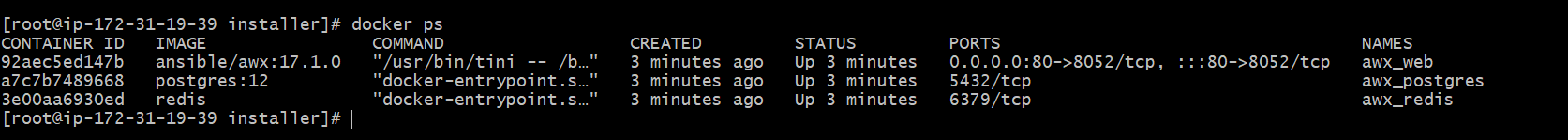
admin\_password=Admin@123  
secret\_key=Admin@123  
pg\_database=awx  
pg\_password=Admin@123  
awx\_alternate\_dns\_servers="8.8.8.8,8.8.4.4"  
postgres\_data\_dir="/var/lib/awx/pgdocker"  
docker\_compose\_dir="/var/lib/awx/awxcompose"  
project\_data\_dir="/var/lib/awx/projects"

### Install AWX by running the Ansible Playbook

ansible-playbook -i ~/awx/installer/inventory ~/awx/installer/install.yml

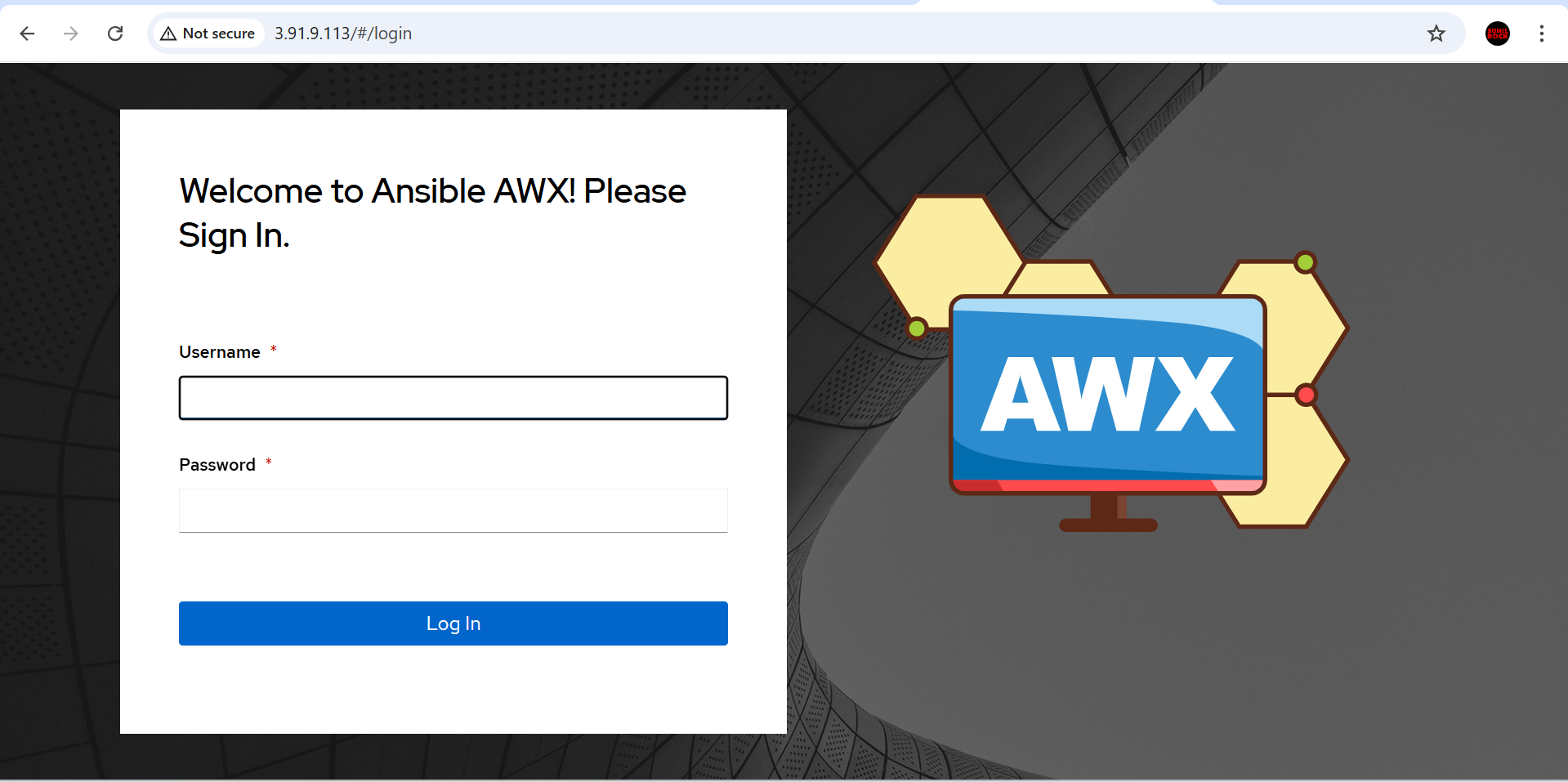
 

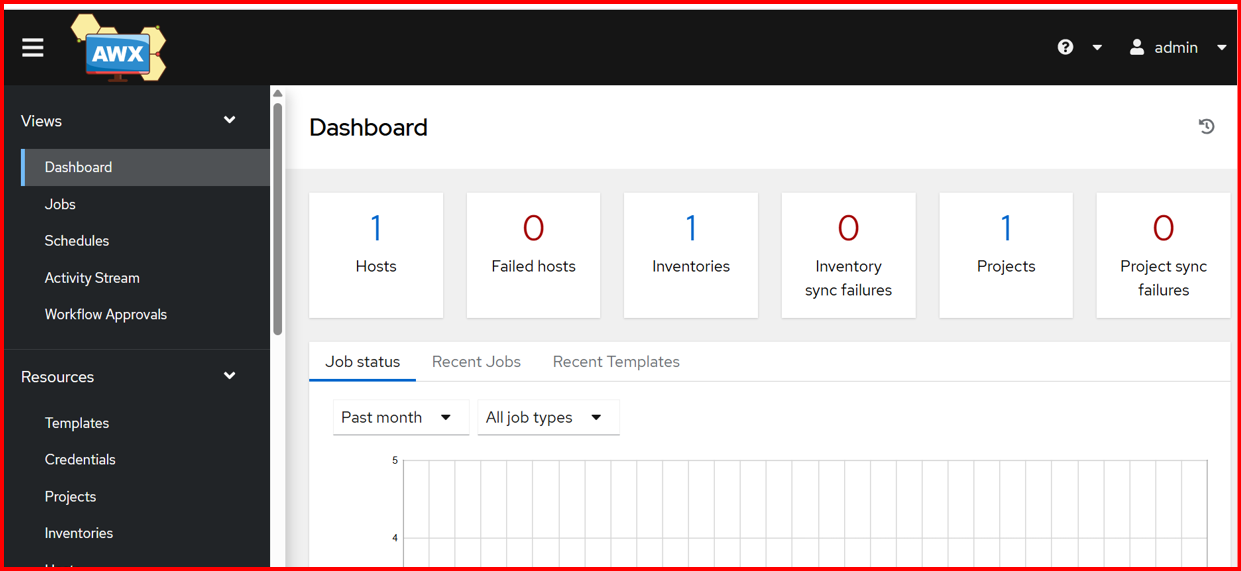
Check docker ps



## ****Step 8: Access AWX****

* Open in browser: http://<EC2\_PUBLIC\_IP>
* Login:
  + **Username:** admin
  + **Password:** the one you set in inventory (Admin@123)

****

****