**Task 18: Automated Apache Deployment with Ansible: Simplifying Web Server Setup**

**Ansible** is an open-source automation tool developed by Red Hat. It is used for configuration management, application deployment, task automation, and IT orchestration. Ansible simplifies managing large-scale IT infrastructure by automating repetitive tasks and ensuring consistency across systems.

Apache is one of the most widely used web servers, known for its flexibility and scalability in hosting websites and applications. Automating its deployment with Ansible streamlines the process of setting up web servers in dynamic and large-scale environments.

This task involves leveraging **Ansible**, a powerful automation tool, to efficiently install and start the **Apache HTTP server** (commonly referred to as httpd or apache2) across multiple servers. This automation ensures that the web server is deployed uniformly on all target systems without manual intervention, saving time and reducing the risk of configuration errors.

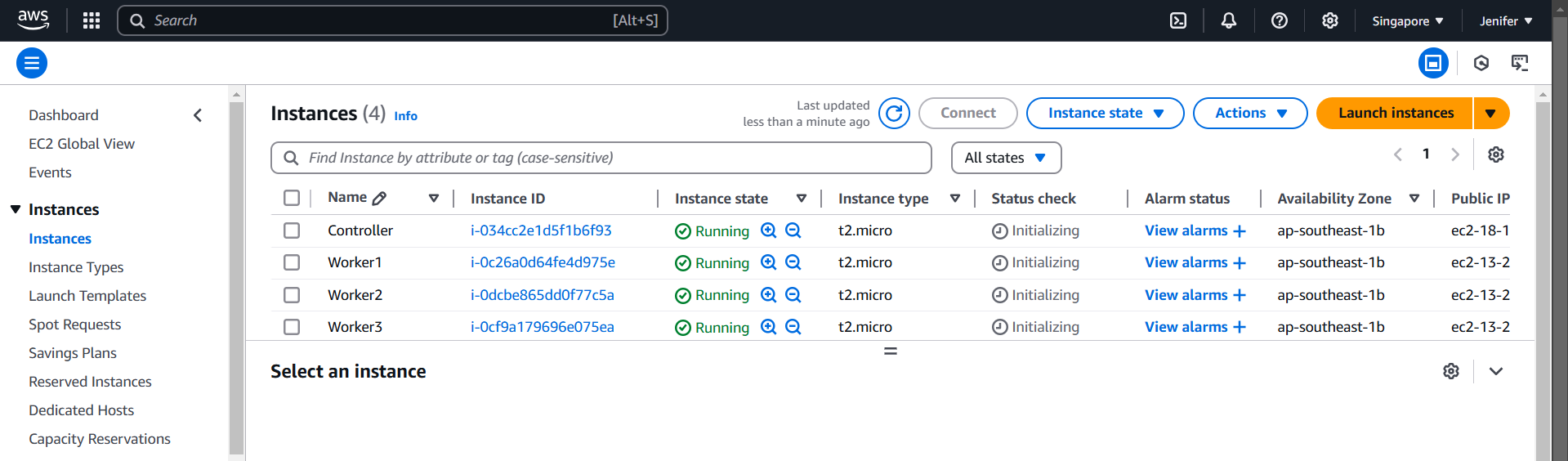
**Objective:**

**The primary goal of this task is to:**

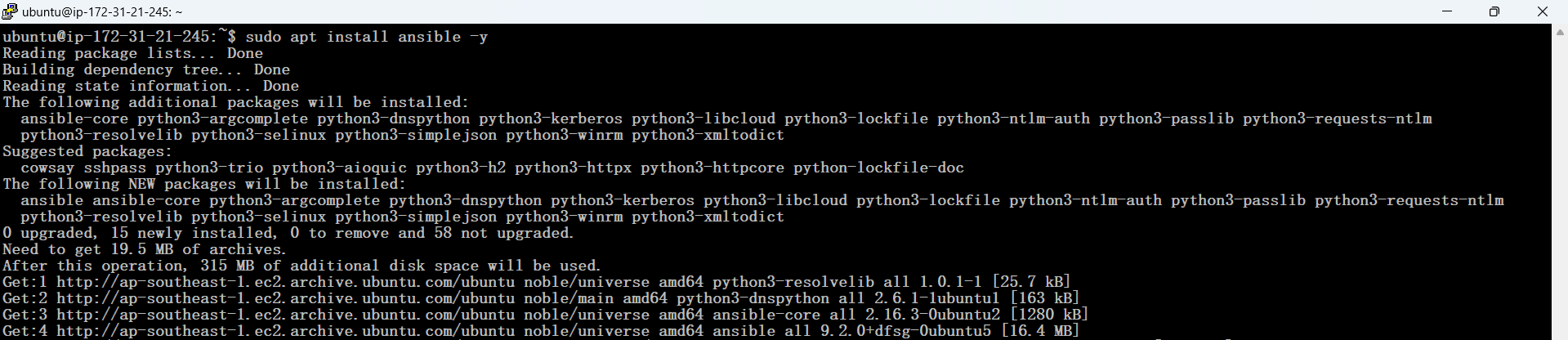
* Install the Apache server on all target nodes.
* Ensure the Apache service is enabled to start at boot.
* Start the Apache server to make it operational immediately.

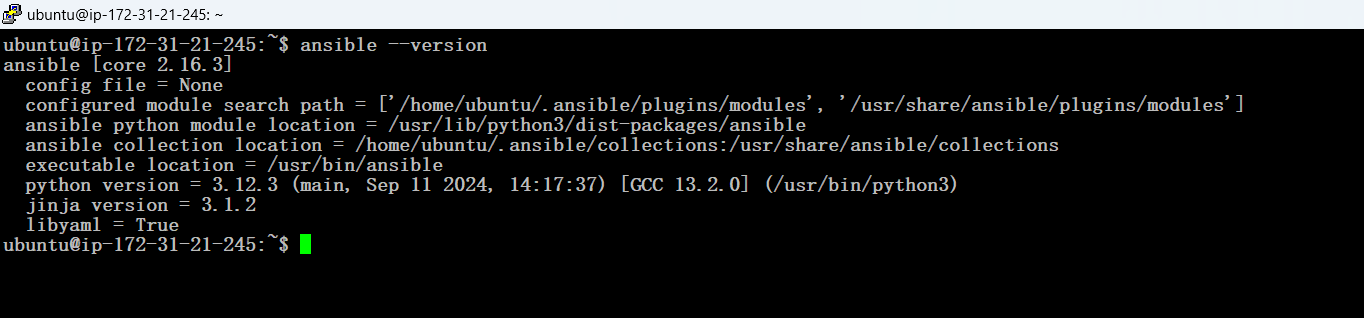
**Practical Implementation: Deploying Apache with Ansible (Step-by-Step with Screenshots)**

1. I have set up one controller node and three worker nodes to execute this task

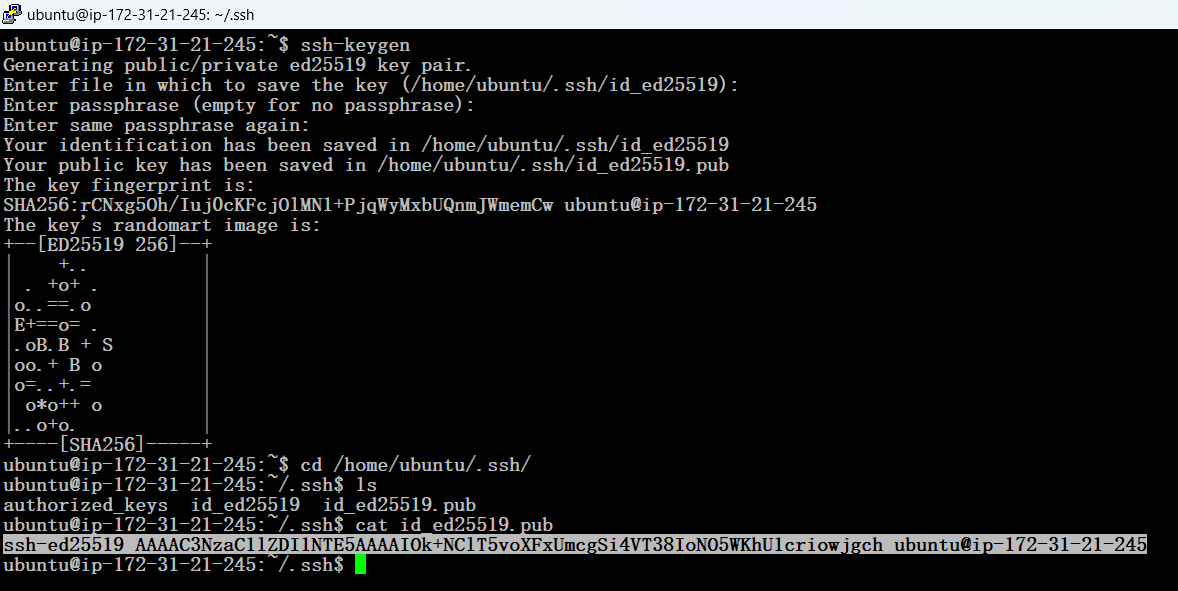


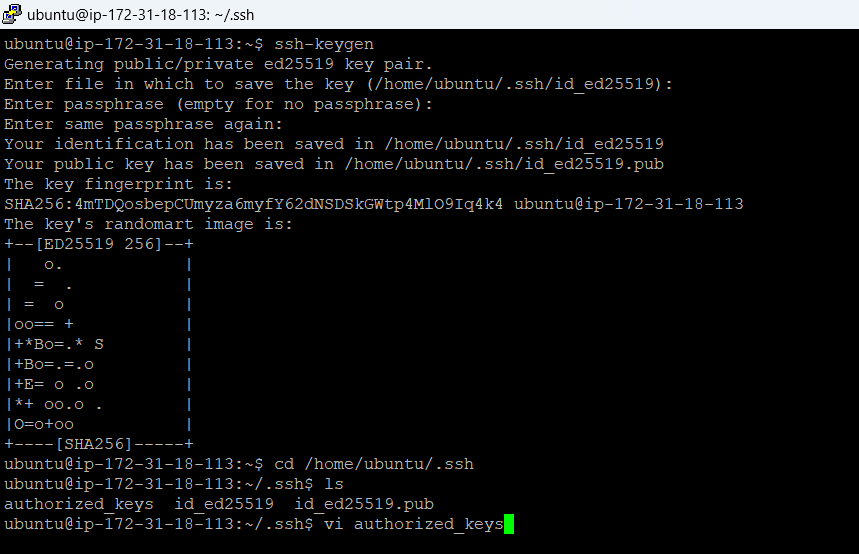
2. I have installed Ansible in the controller node





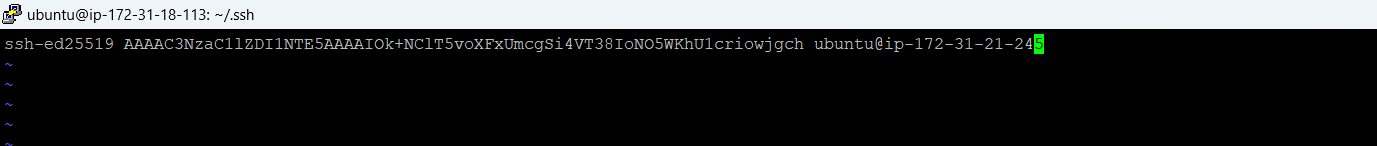
3. Setting up password-less SSH key authentication between the controller node and the worker nodes for seamless and secure communication.

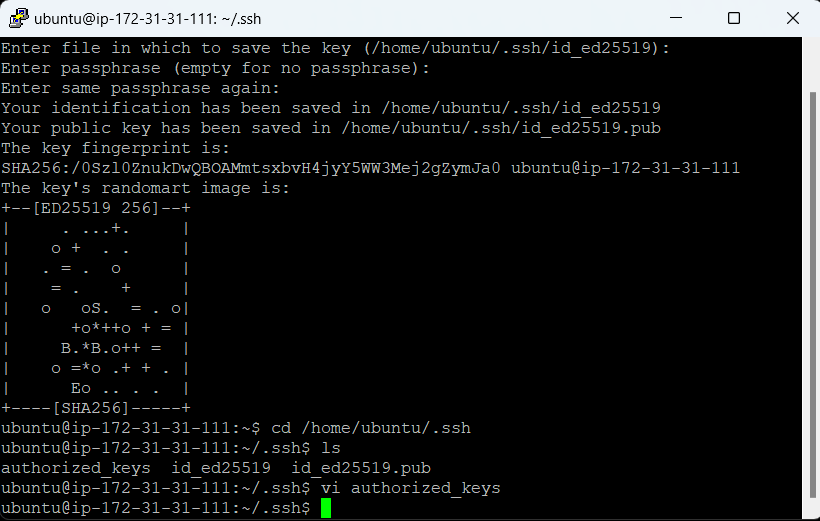


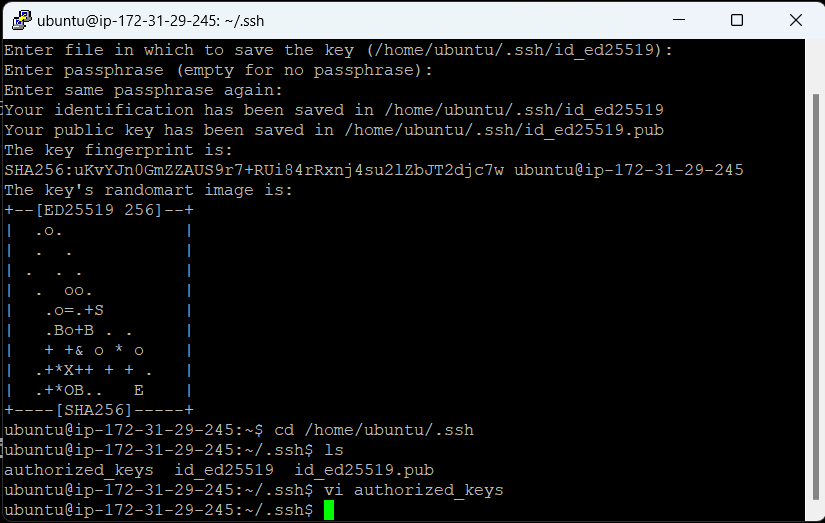


**Authorized keys** are public SSH keys that are stored on a server to grant access to users or systems with the corresponding private SSH keys. They are a crucial part of SSH-based password-less authentication, enabling secure access to a system without requiring a password.

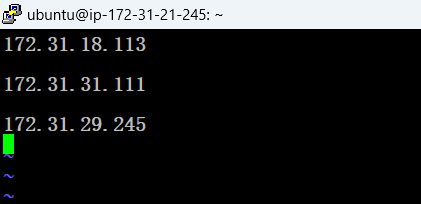
4. Storing Controller’s Public key into the Worker node’s Authorized keys





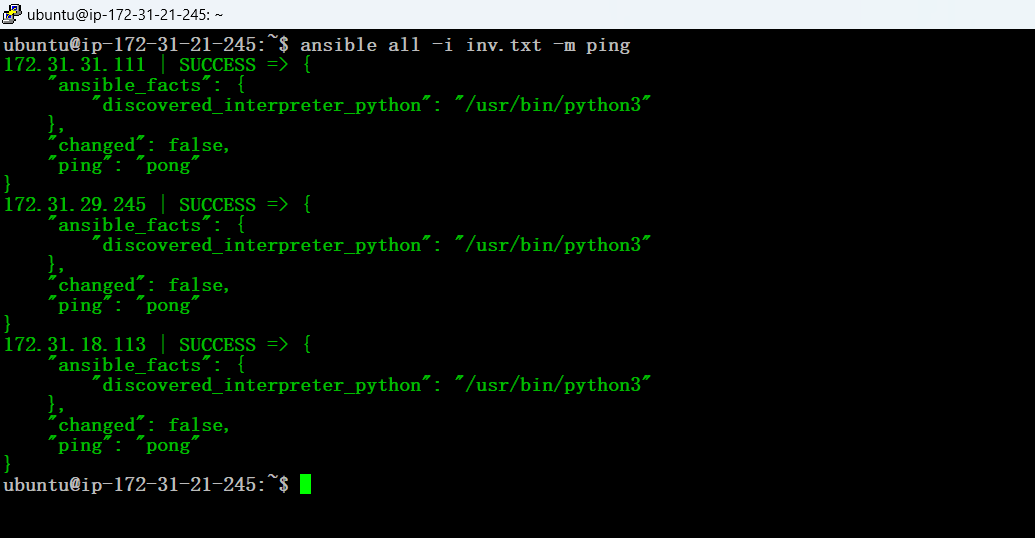


5. Creating the inventory file for defining and managing the target nodes in the Ansible setup.



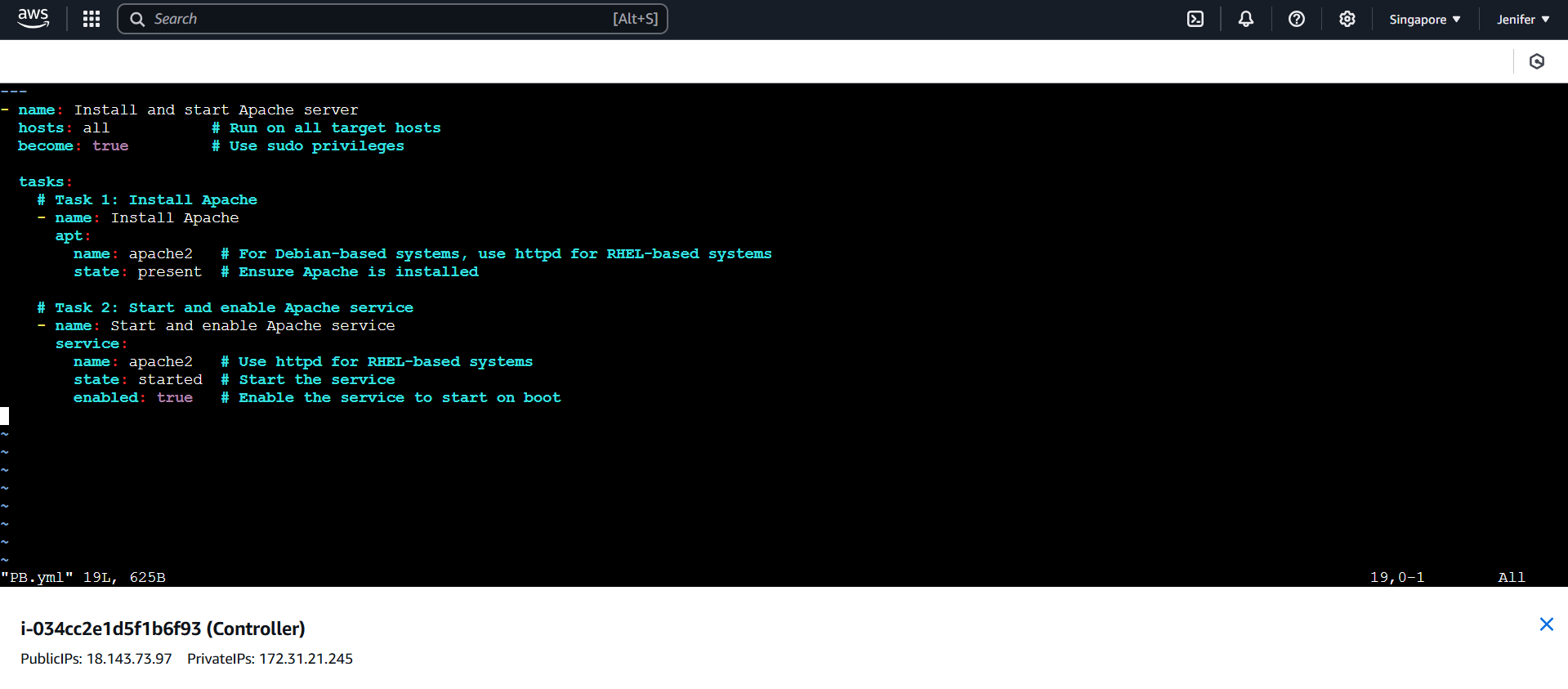
An **inventory file** in Ansible is a configuration file that contains a list of target systems (managed nodes) that Ansible will interact with. It serves as a centralized resource for defining the hosts and groups of hosts, along with their connection details and optional variables.

6. Testing the connection between the Ansible controller node and all the worker nodes Using ADHOK command (ansible all –I inv.txt –m ping).

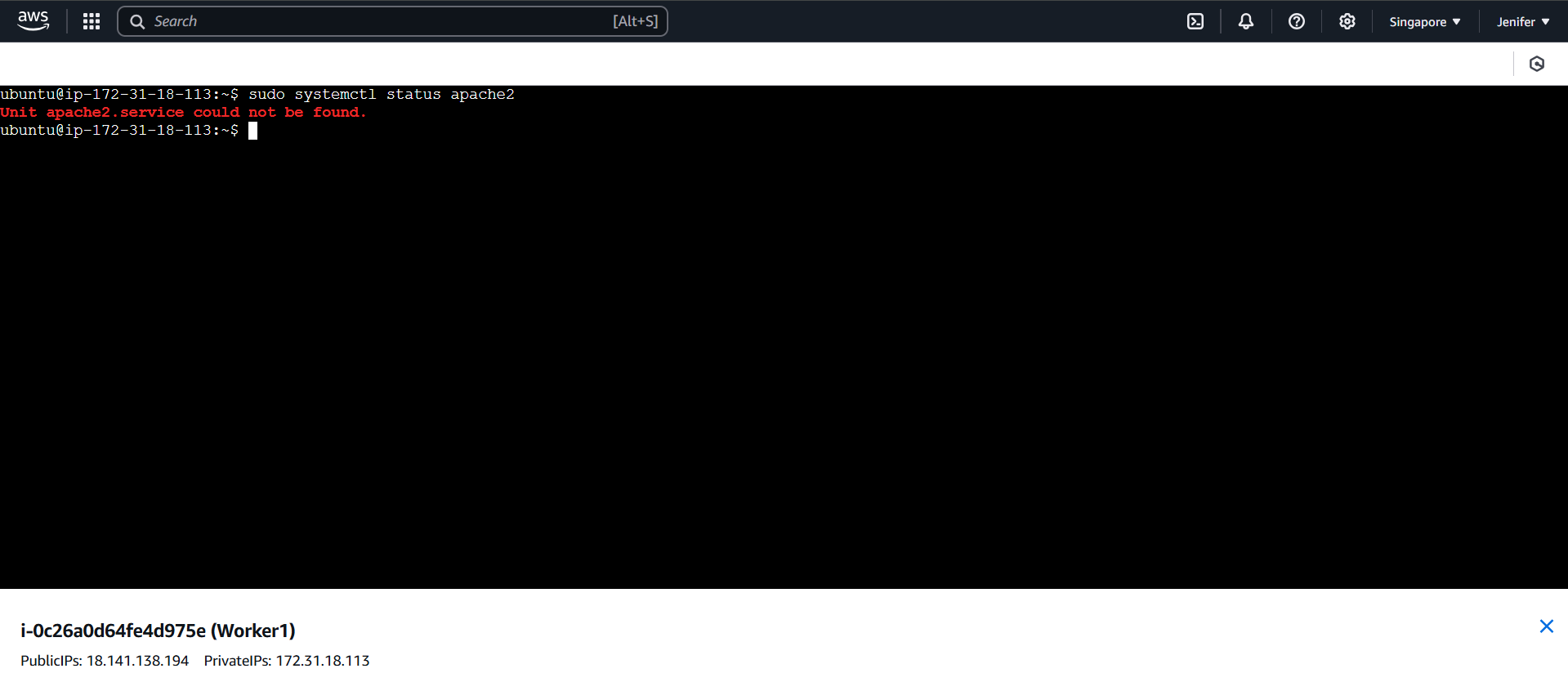


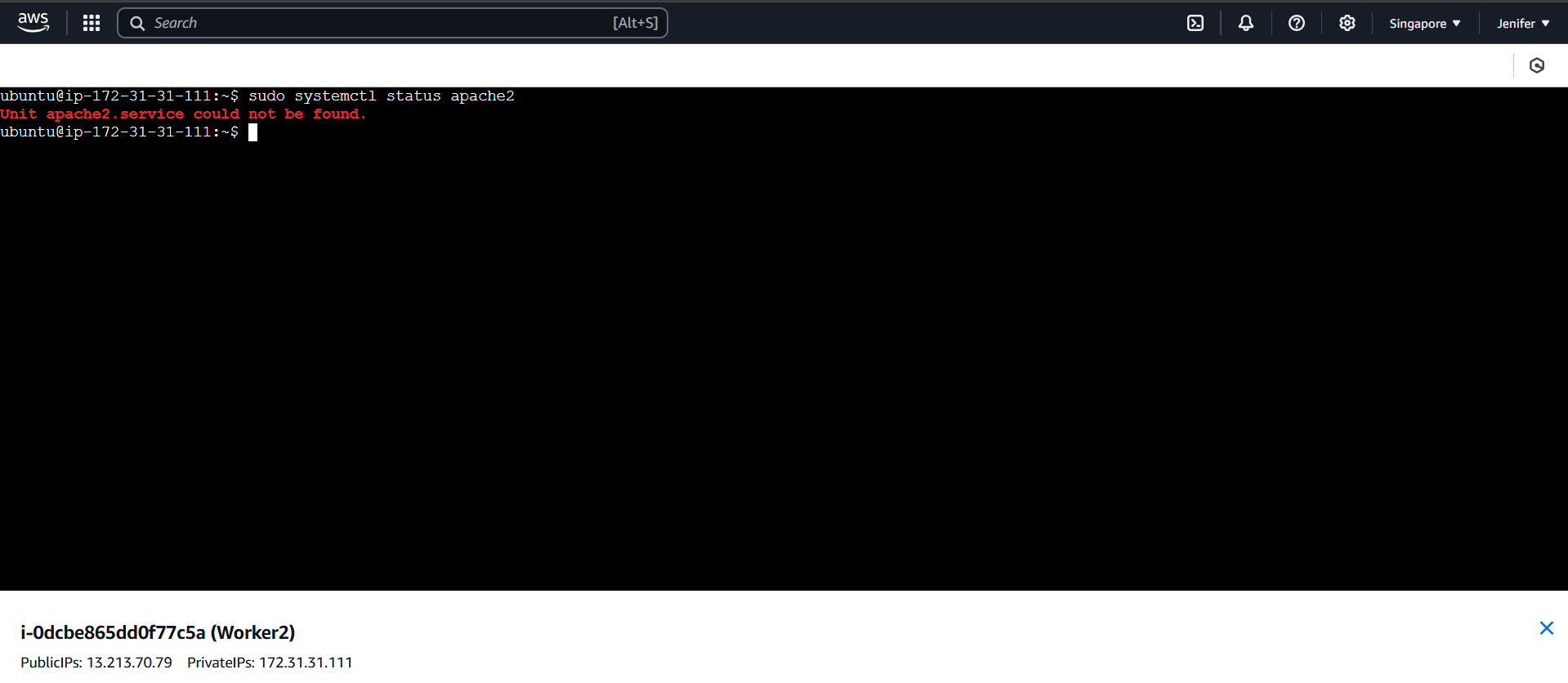
**Ad-hoc commands** are one-off commands used to perform quick tasks without writing a playbook. They are ideal for tasks like testing connectivity, managing files, restarting services, or gathering information from managed nodes.

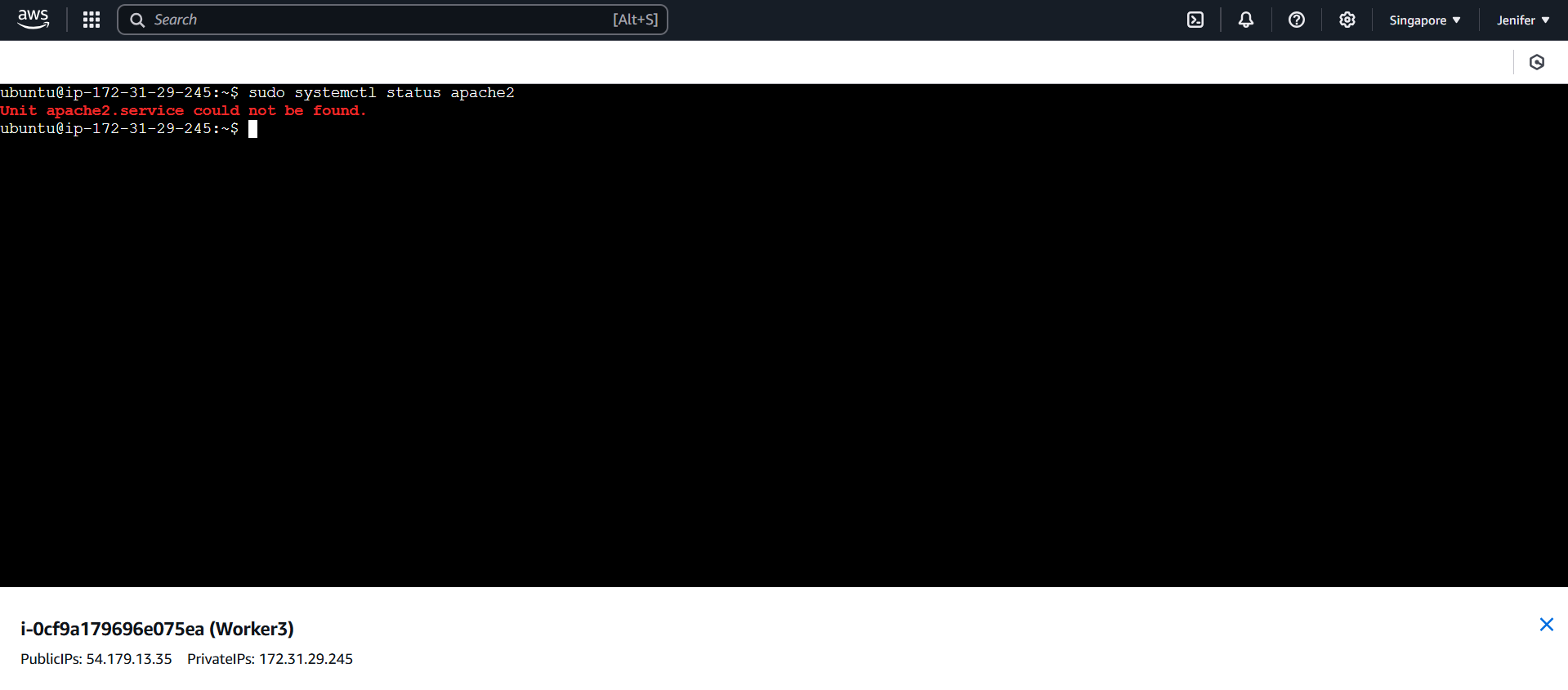
7. Preparing Playbook to install Apcahe2 and start the service for all the worker nodes given in the Inventory file (inv.txt)



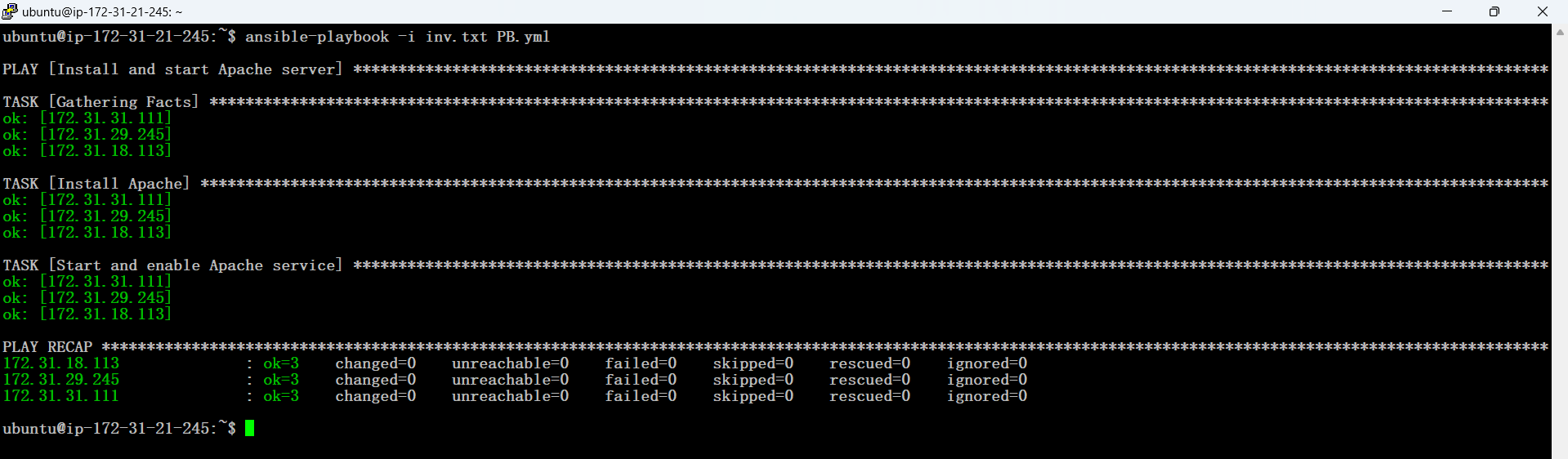
8. Pre-execution status check before running the playbook.





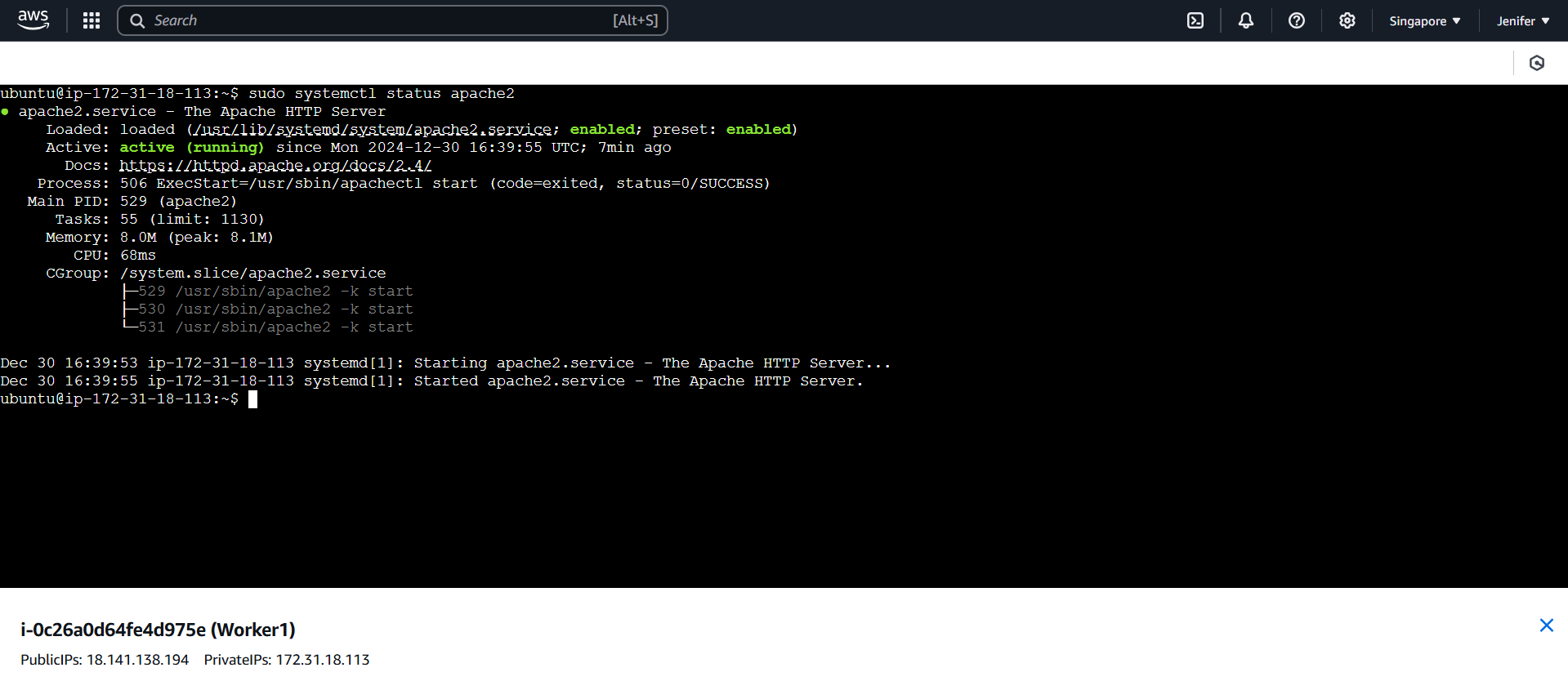


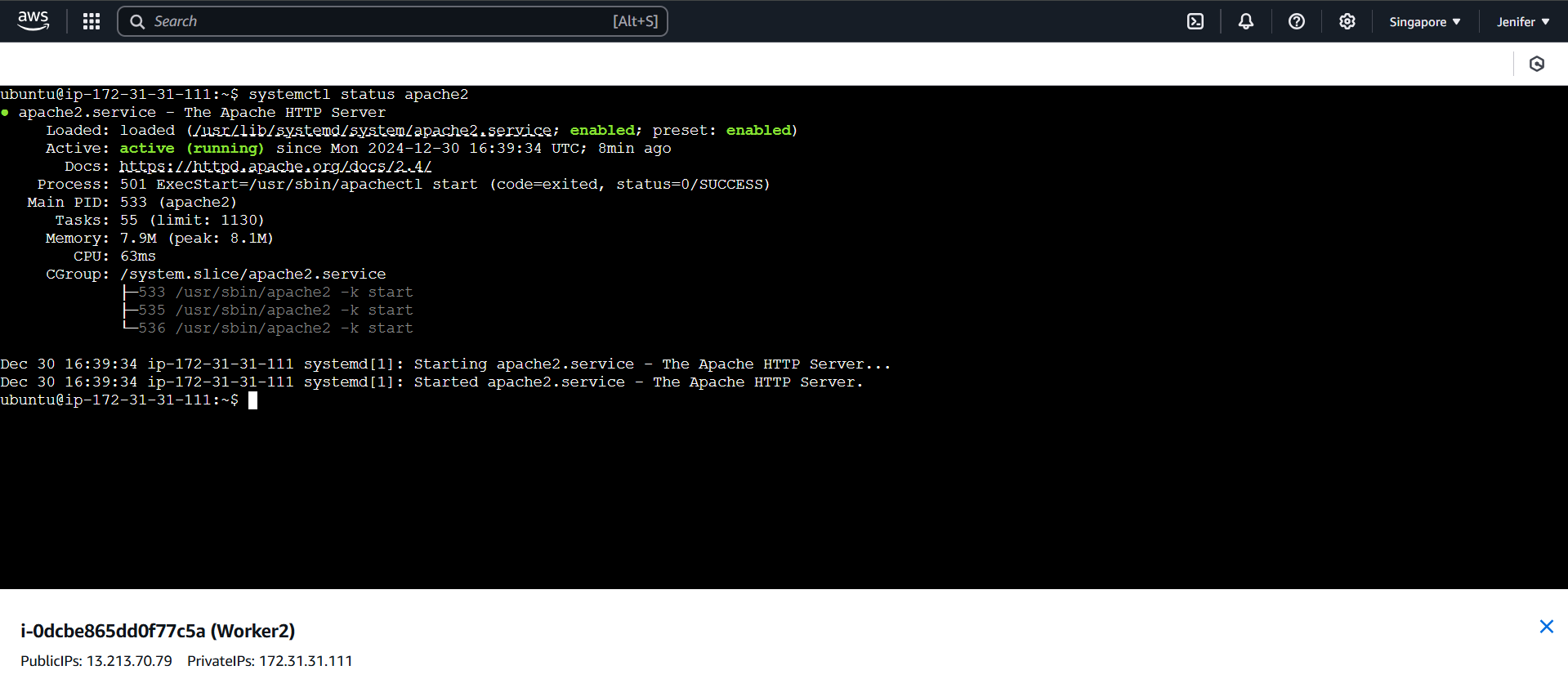
9. Playbook (PB.yml) Execution

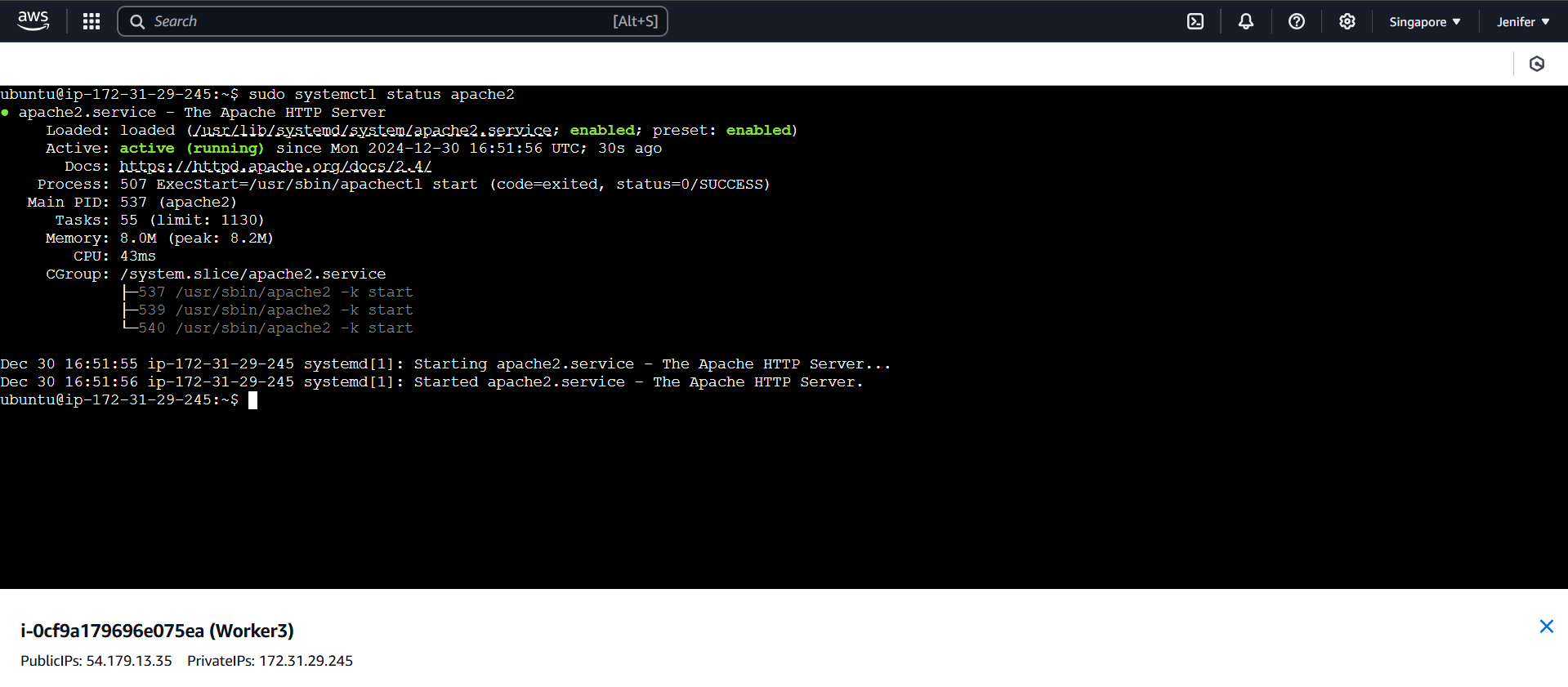


A **playbook** in Ansible is a file written in YAML (Yet Another Markup Language) format that contains a series of tasks, which define the automation process for managing and configuring systems. Playbooks allow you to automate complex workflows and operations in a repeatable and organized way.

10. Apache2 WebServer has successfully installed and the service started in all the three Worker nodes via executing the Playbook (PB.yml)







**Impact:**

By the end of this task, all target servers will have a fully installed and operational Apache HTTP server. This setup will be achieved efficiently through a single Ansible playbook, demonstrating how automation can simplify web server deployment.