|  |  |  |  |
| --- | --- | --- | --- |
|  | Bansilal Ramnath Agarwal Charitable Trust's  Vishwakarma Institute of Information Technology  **Department of**  **Artificial Intelligence and Data Science** | | |
| Name: Siddhesh Dilip Khairnar | | | |
| Class: TY | Division: B | | Roll No: 372028 |
| Semester: 5th | | Academic Year: 2023-2024 | |
| Subject Name & Code: Cloud Computing and Analytics (ADUA31203) | | | |
| Title of Assignment: Write ansible playbook to install nginx on target servers | | | |

**Assignment 6**

**Title:** Write an ansible playbook to deploy NGINX Web server.

**Theory:**

**1) What is YAML**

YAML is a human-readable data serialization language that is often used for writing configuration files. Depending on whom you ask, YAML stands for yet another markup language or YAML isn’t markup language (a recursive acronym), which emphasizes that YAML is for data, not documents.

YAML is a popular programming language because it is designed to be easy to read and understand. It can also be used in conjunction with other programming languages. Because of its flexibility, and accessibility, YAML is used by Ansible® to create automation processes, in the form of Ansible Playbooks.

**YAML syntax**

YAML files use a .yml or .yaml extension, and follow specific syntax rules.

YAML has features that come from Perl, C, XML, HTML, and other programming languages. YAML is also a superset of JSON, so JSON files are valid in YAML.

There are no usual format symbols, such as braces, square brackets, closing tags, or quotation marks. And YAML files are simpler to read as they use Python-style indentation to determine the structure and indicate nesting. Tab characters are not allowed by design, to maintain portability across systems, so whitespaces—literal space characters—are used instead.

Comments can be identified with a pound or hash symbol (#). It’s always a best practice to use comments, as they describe the intention of the code. YAML does not support multi-line comment, each line needs to be suffixed with the pound character.

A common question for YAML beginners is “What do the 3 dashes mean?” 3 dashes (---) are used to signal the start of a document, while each document ends with three dots (...).

**This is a very basic example of a YAML file:**

**#Comment: This is a supermarket list using YAML**

**#Note that - character represents the list**

**---**

**food:**

**- vegetables: tomatoes #first list item**

**- fruits: #second list item**

**citrics: oranges**

**tropical: bananas**

**nuts: peanuts**

**sweets: raisins**

Note that the structure of a YAML file is a map or a list, and it follows a hierarchy depending on the indentation, and how you define your key values. Maps allow you to associate key-value pairs. Each key must be unique, and the order doesn't matter. Think of a Python dictionary or a variable assignment in a Bash script.

A map in YAML needs to be resolved before it can be closed, and a new map is created. A new map can be created by either increasing the indentation level or by resolving the previous map and starting an adjacent map.

A list includes values listed in a specific order and may contain any number of items needed. A list sequence starts with a dash (-) and a space, while indentation separates it from the parent. You can think of a sequence as a Python list or an array in Bash or Perl. A list can be embedded into a map.

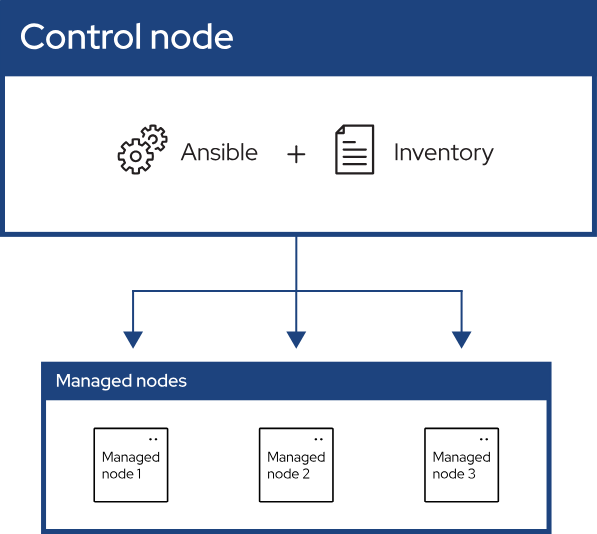
In the example provided above “vegetables” and “fruits” represent items that are part of the list named “food”.

YAML also contains scalars, which are arbitrary data (encoded in Unicode) that can be used as values such as strings, integers, dates, numbers, or booleans.

When creating a YAML file, you’ll need to ensure that you follow these syntax rules and that your file is valid. To achieve it, you can use a linter—an application that verifies the syntax of a file. The yamllint command can help to ensure you’ve created a valid YAML file before you hand it over to an application.

**2) Introduction to Ansible**

Ansible automates the management of remote systems and controls their desired state.



As shown in the preceding figure, most Ansible environments have three main components:

**Control node**

A system on which Ansible is installed. You run Ansible commands such as ansible or ansible-inventory on a control node.

**Inventory**

A list of managed nodes that are logically organized. You create an inventory on the control node to describe host deployments to Ansible.

**Managed node**

A remote system, or host, that Ansible controls.

**Introduction to Ansible**

Ansible provides open-source automation that reduces complexity and runs everywhere. Using Ansible lets you automate virtually any task. Here are some common use cases for Ansible:

* Eliminate repetition and simplify workflows
* Manage and maintain system configuration
* Continuously deploy complex software
* Perform zero-downtime rolling updates

Ansible uses simple, human-readable scripts called playbooks to automate your tasks. You declare the desired state of a local or remote system in your playbook. Ansible ensures that the system remains in that state.

As automation technology, Ansible is designed around the following principles:

**Agentless architecture**

Low maintenance overhead by avoiding the installation of additional software across IT infrastructure.

**Simplicity**

Automation playbooks use straightforward YAML syntax for code that reads like documentation. Ansible is also decentralized, using SSH existing OS credentials to access to remote machines.

**Scalability and flexibility**

Easily and quickly scale the systems you automate through a modular design that supports a large range of operating systems, cloud platforms, and network devices.

**Idempotence and predictability**

When the system is in the state your playbook describes, Ansible does not change anything, even if the playbook runs multiple times.

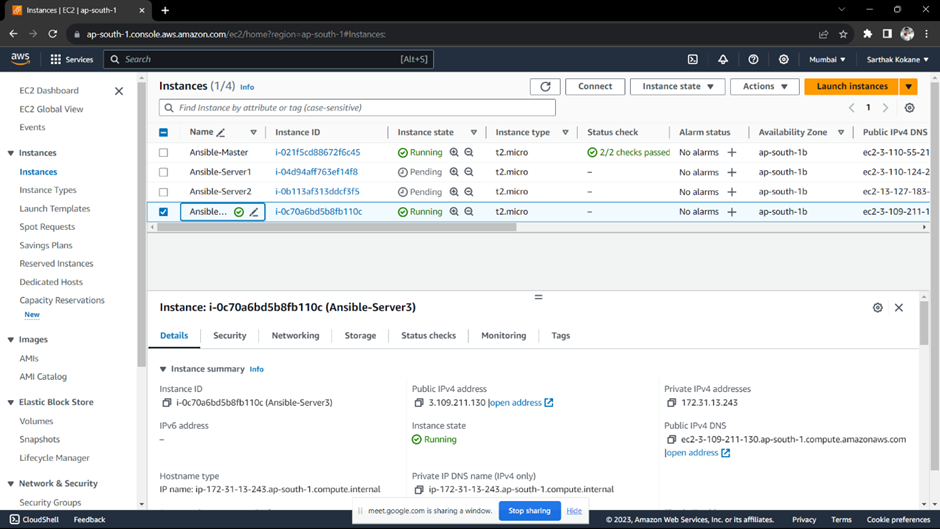
**Implementation:**

1. **Architecture:**



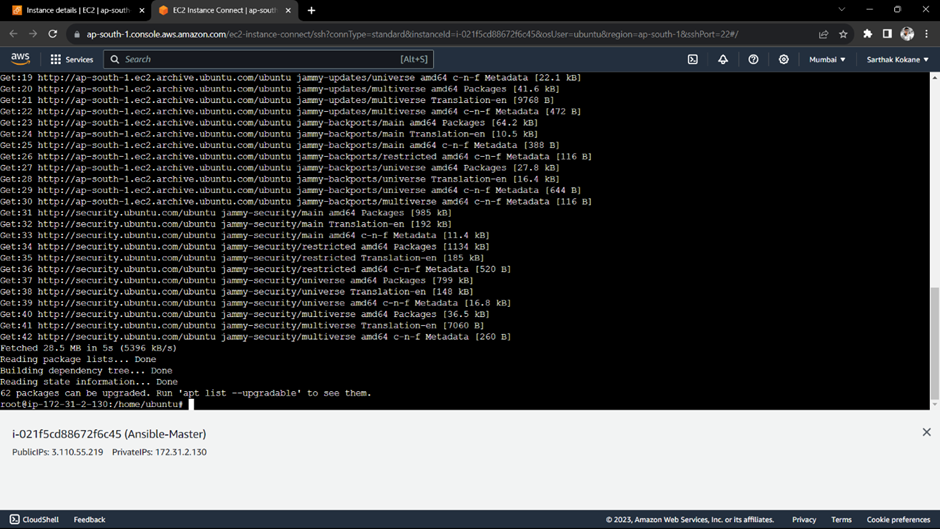
*Figure1: Architecture Diagram*

1. **Steps**
2. Create 4 ec2 instances of **Ubuntu machine.**



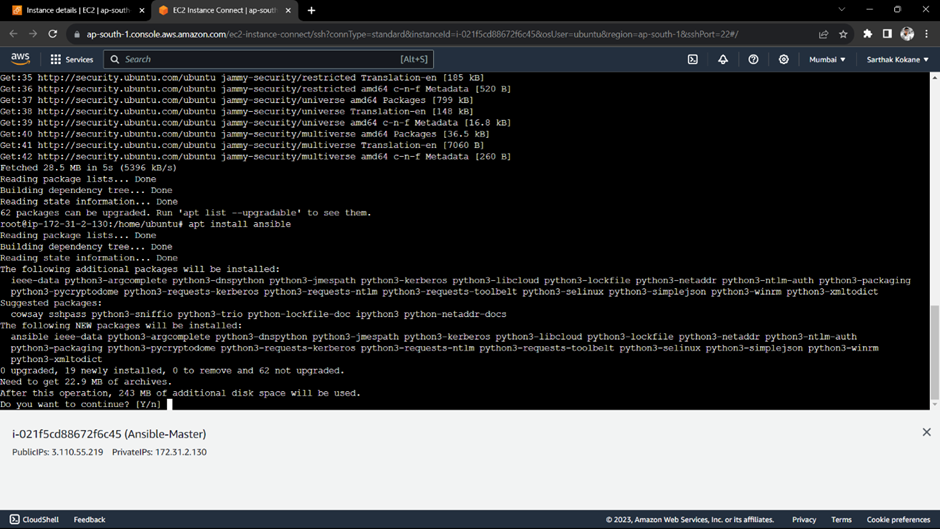
1. Connect to **“Ansible-Master”** server
2. Run following commands.

* **sudo -i**
* **apt update**



* Install ansible using command.

**apt install ansible**



* Check the version of ansible using command

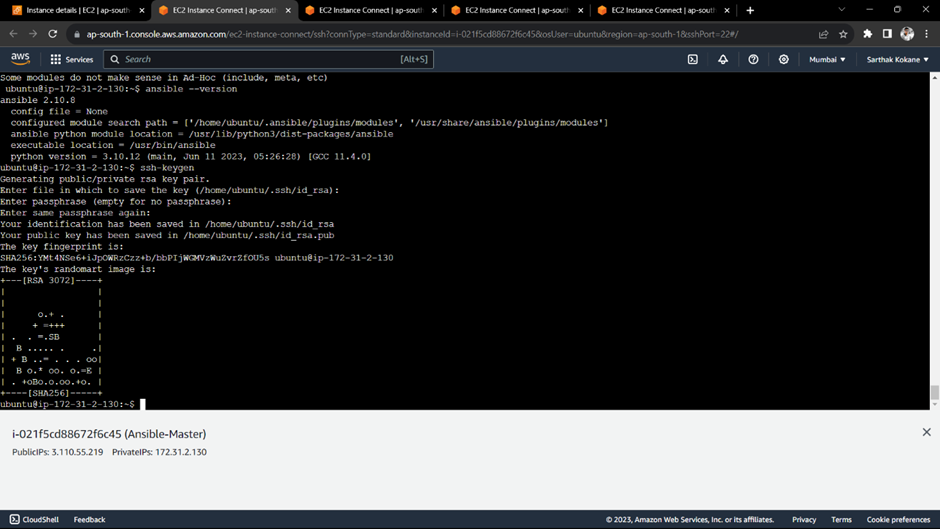
**ansible --version**

* Update remaining all hosts i.e. Ansible-Server1, Ansible-Server2, Ansible-Server3 using command

**sudo apt-get update**

1. Generate a ssh key on Ansible-master using command

**ssh-keygen**

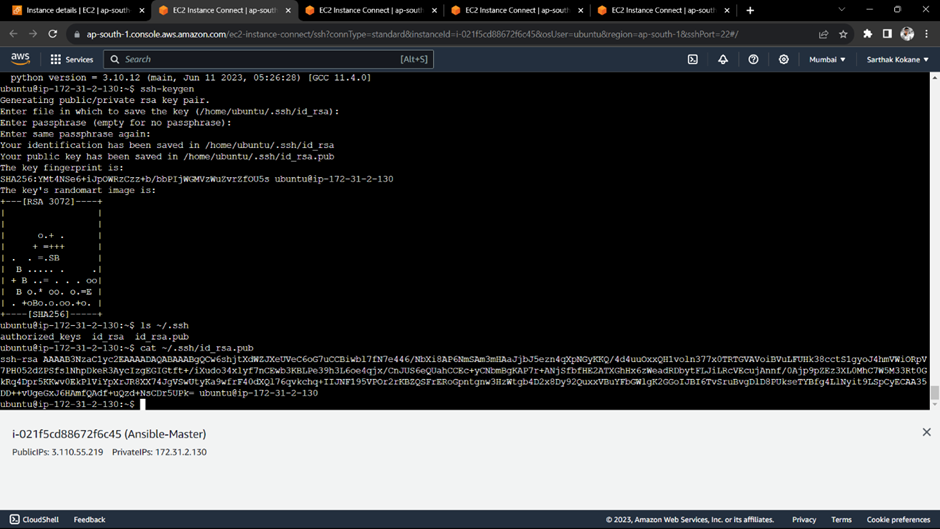
****

1. copy the public key which is in .ssh folder into “authorized keys” on ansible-server1

commands:

**ls ~/.ssh**

**cat ~/.ssh/id\_rsa.pub**



1. Connect to ansible-server1 and again give command

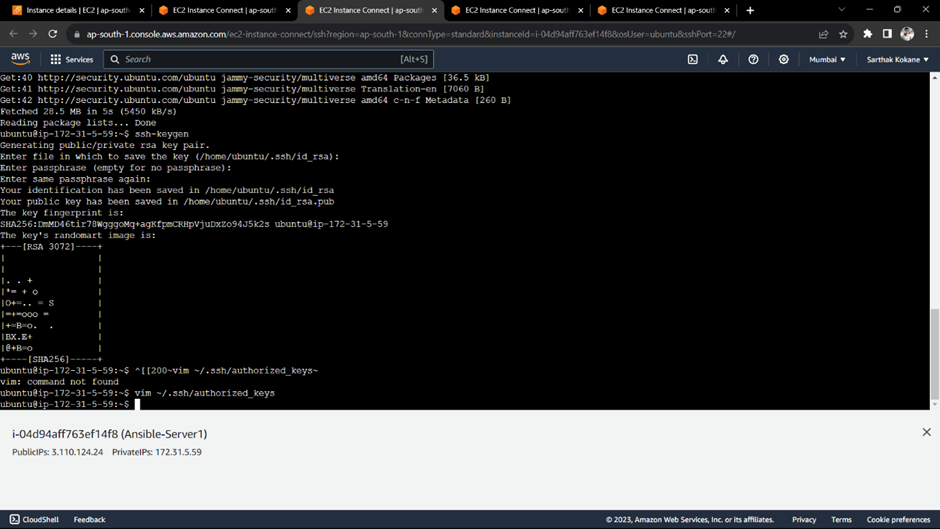
**ssh-keygen**

**It will create the same files on ansible-server1**

Now,

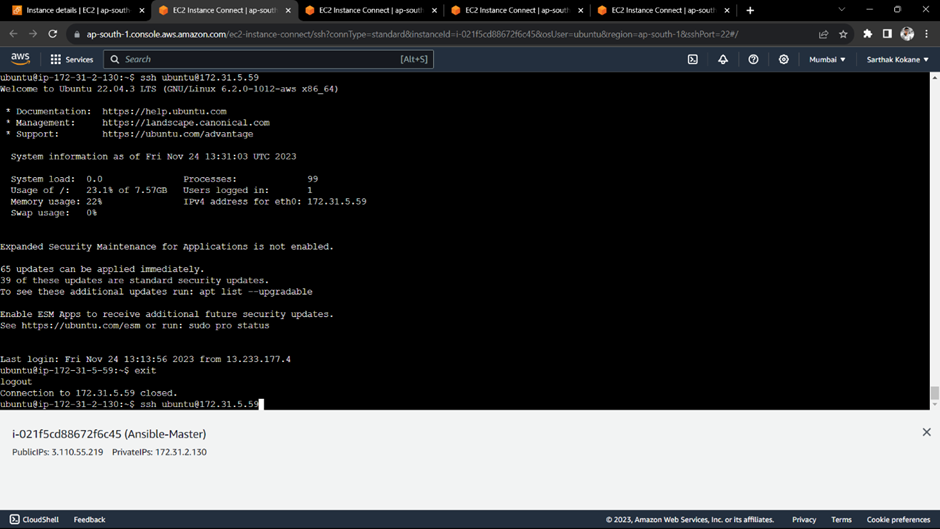
**vim ~/.ssh/authorized\_keys**

**copy the public key**



1. Now login to Ansible-master and try to connect to ansible server using command

**ssh ubuntu@private-ip**



**Create a playbook on Ansible-master**

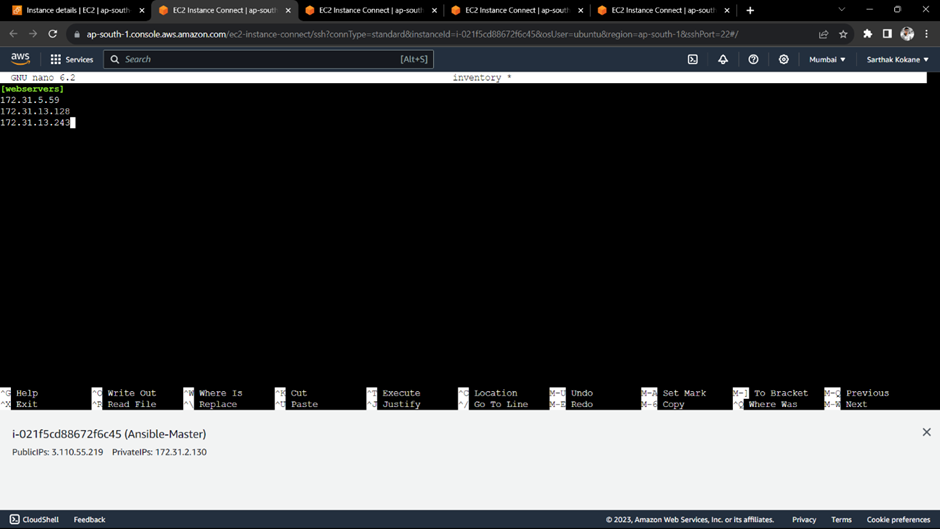
**Step 1:-** Connect to “Ansible-Master”

**Step 2:-** Create a new folder “ansible-project” using command

**Step 3:**

* cd ansible-project
* nano inventory

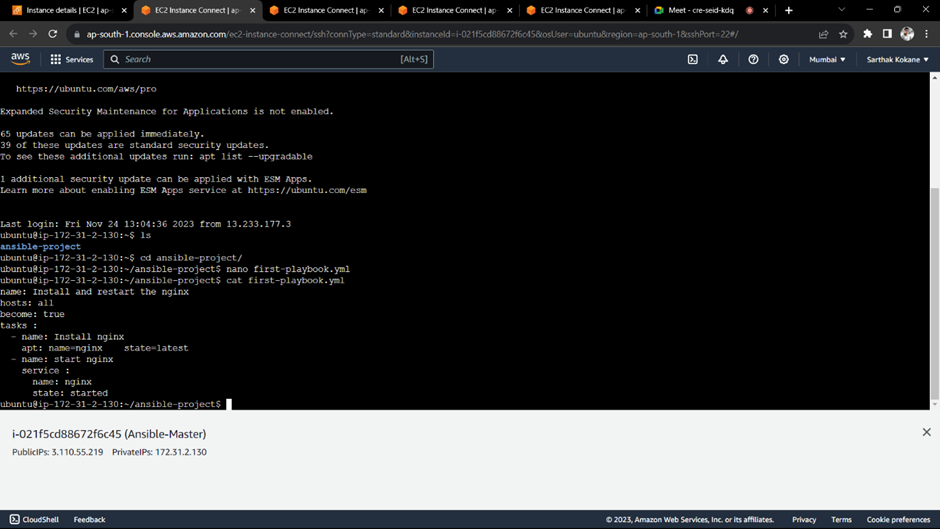
1. Write the private IP of “Ansible-server1” into inventory.
2. Write the private IP of “Ansible-server2” into inventory.
3. Write the private IP of “Ansible-server3” into inventory.



1. Save and exit.

**Task: Install Nginx and Start Nginx**

**Step 1:** Create a new file called “first-playbook.yml”



**Code:**

**- name: Install and restart the nginx**

**hosts: all**

**become: true**

**tasks:**

**- name: install nginx**

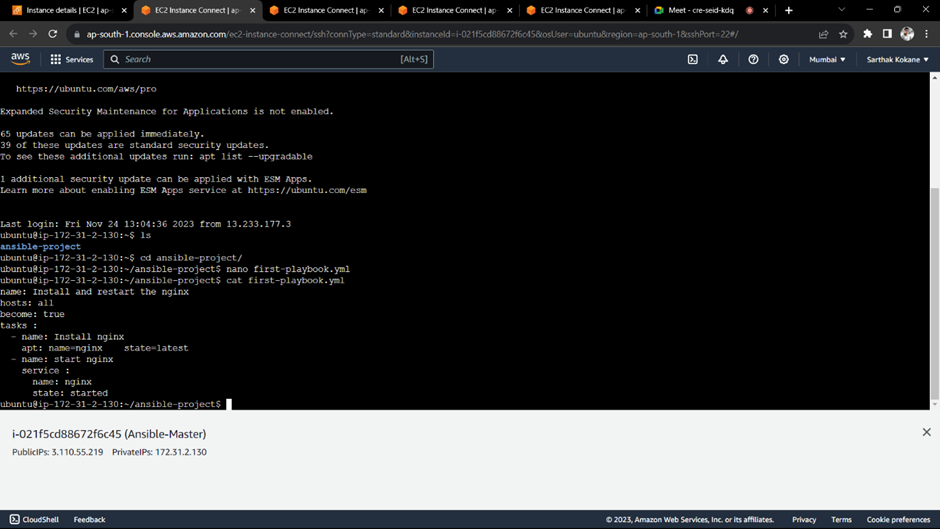
**apt: name=nginx state=latest**

**- name: start nginx**

**service:**

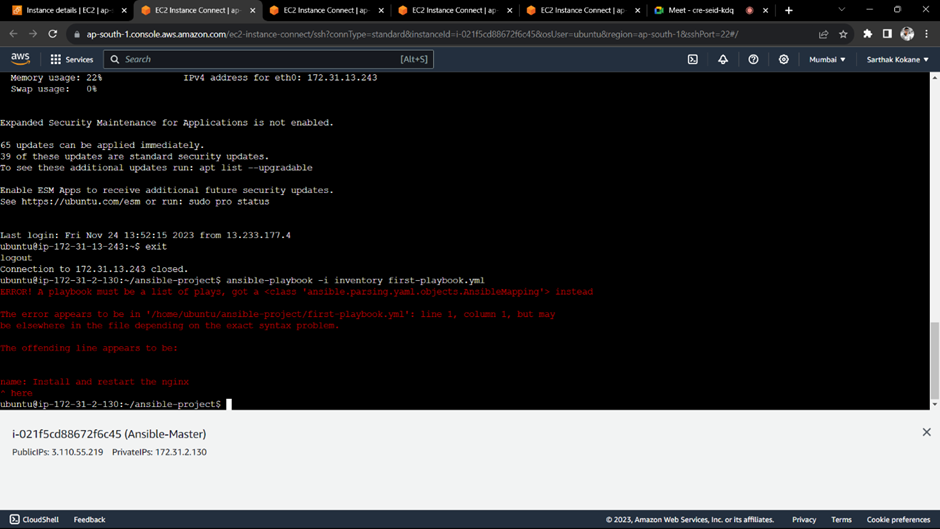
**name: nginx**

**state: started**

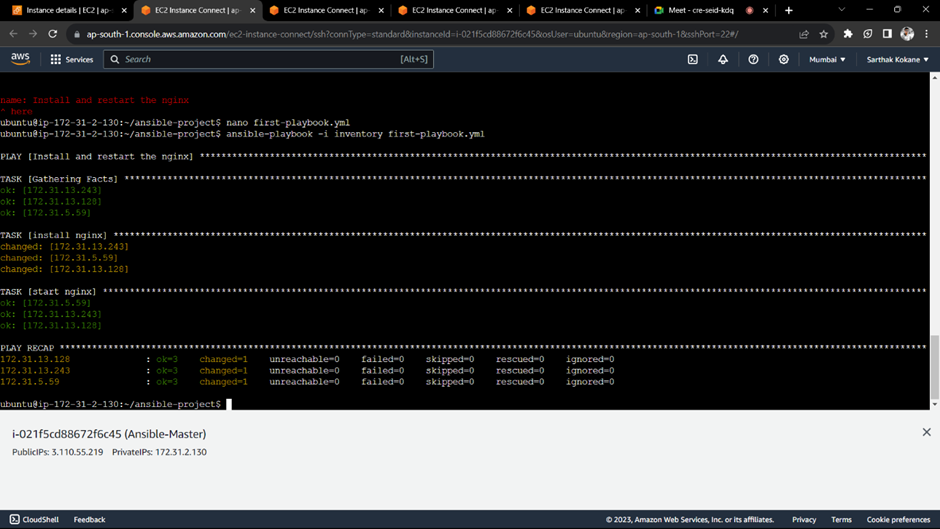
****

**Execute the playbook by using command:**

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



**Output:**

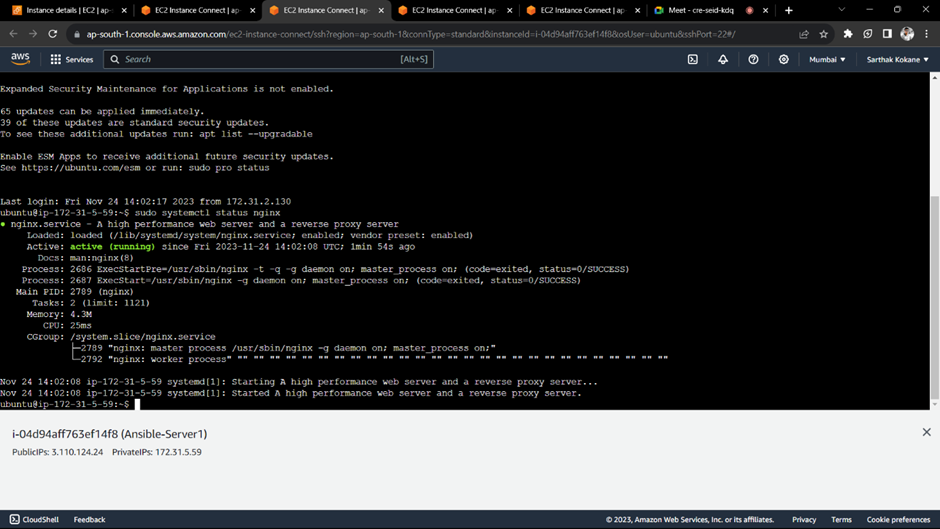
****

**Verify the output:**

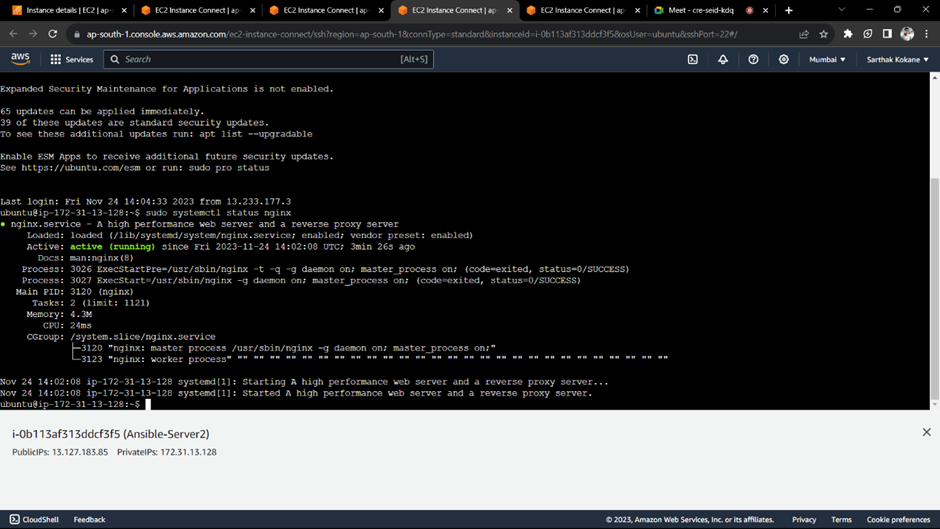
**Step 1:** Connect to any Ansible-Server

**Step 2:** Run the command: **sudo systemctl status nginx**

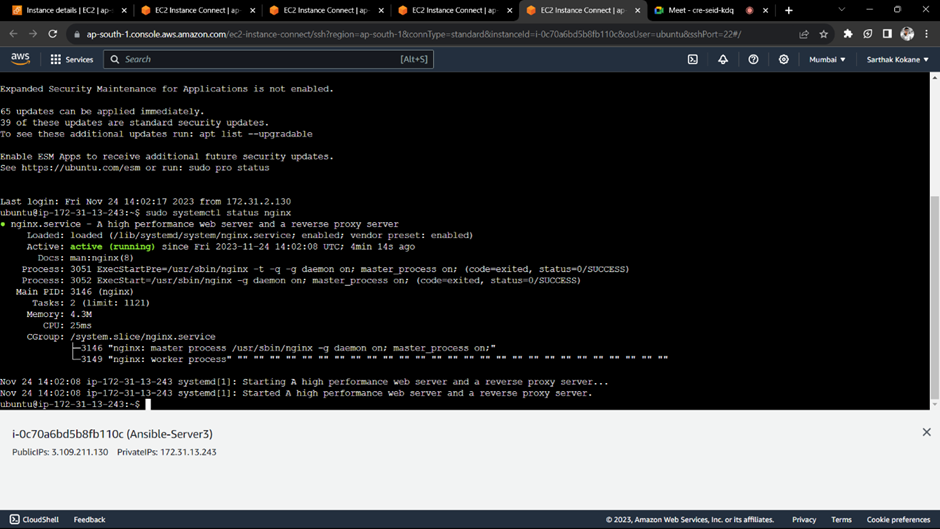
* *Ansible-Server1*

****

* *Ansible-Server2*

**

* *Ansible-Server3*

**

**Conclusion:** Thus, we have successfully used Ansible playbooks to install nginx on 3 target servers.