**ANSIBLE**

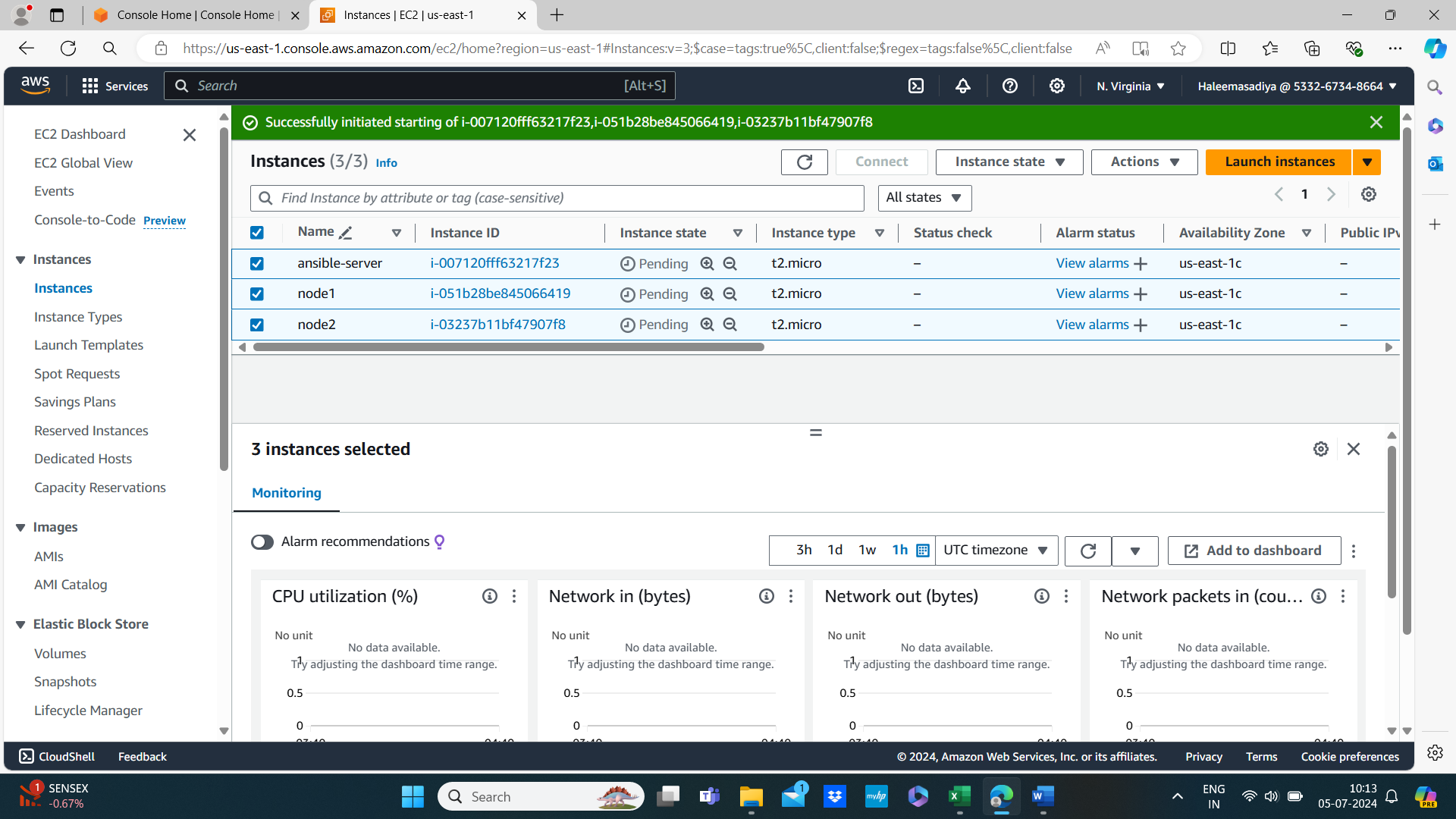
Ansible is an opensource IT Automation tool that automates application deployment, infrastructure service orchestration, cloud provisioning and many more. It uses YAML files or playbooks at run time. In this guide, you will learn how to install and configure Ansible on Amazon Linux 2 on AWS.

Ansible uses push mechanism that’s why it doesn’t require any agent on nodes. Whereas puppet and chef configuration management tools uses pull mechanism and requires agents to be installed on nodes.

Although there are various advantages of using Ansible in your infra but below are few and important advantages.

* Agentless
* No need to install nodes on remote servers
* Totally rely on SSH
* Various big organisations uses Ansible such as Apple, NASA, Juniper etc.

Let’s create 3 instances in aws one for ansible server and remaining two for nodes i. e node1 and node2



Connect ansible server using ssh and nodes with ec2 instances connect

Then become sudo user by using the command sudo su

Now download and install ansible server

$ sudo yum update -y

$ wget <https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm>

$ sudo yum install epel-release-latest-7.noarch.rpm

$ sudo yum update -y

$ sudo yum install python python-devel python-pip openssl ansible -y

$ ansible –version

**To create group:**

* Vi /etc/ansible/hosts

[demo]

Node1 pvtip

Node2 pvtip

We can create any no. of groups here

To come out from the editor esc + :wq

**Configure file changes:**

* Vi /etc/ansible/ansible.cfg

Uncomment -inventory and sudo\_user

Create user:

* Adduser <user\_name>
* Passwd <user\_name> 🡪 press enter 🡪then give passwd

To become ansible user

* su - <user\_name>

NODE1

* sudo su
* adduser <user\_name>
* passwd <user\_name>
* su - <user\_name>

NODE2

* sudo su
* adduser <user\_name>
* passwd <user\_name>
* su - <user\_name>

Now give **sudo privileges to ansible user:**

* visudo
* allow root to run any file
* root ALL=(ALL) ALL

<user\_name> ALL=(ALL) NOPASSWD: ALL

* repeat the above steps in node1 and node2

**to establish ssh connection:**

* vi /etc/ssh/sshd\_config
* uncomment

# permit root login yes

# password authentication yes

* repeat the above steps in node1 and node2
* the restart the service run the below command in three nodes
* # service sshd restart
* Now go to ansible server then become ansible user using the below command

Su - <user\_name>

Ssh <node1 pvtip>

Finally able to access node1 in ansible server then

Ssh <node2 pvtip>

It ask for the password

* If I want again node1 access it ask password again and again to avoid this issue we are going ti use trust relationship
* Now go back to ansible server then become ansible-user then run the following commads

# su - <user\_name>

# ssh-keygen

# ls-a

.ssh will appear

# cd .ssh/ 🡪 two keys will appear one is public key and other is private key

* Copy this public key in two using by the following commands

# ssh-copy-id ansible@pvtip1

# ssh-copy-id ansible@pvtip2

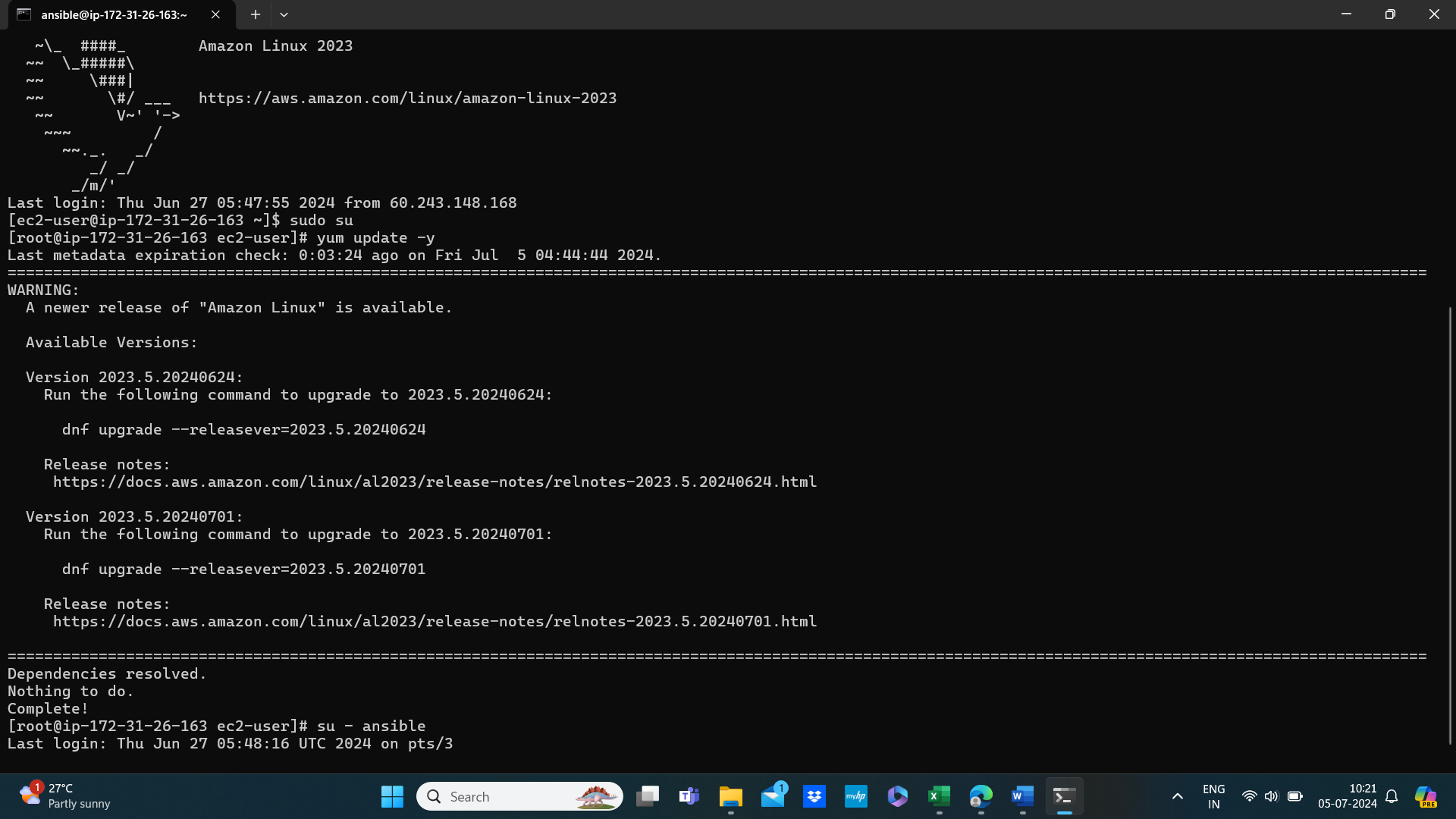
# cd ..

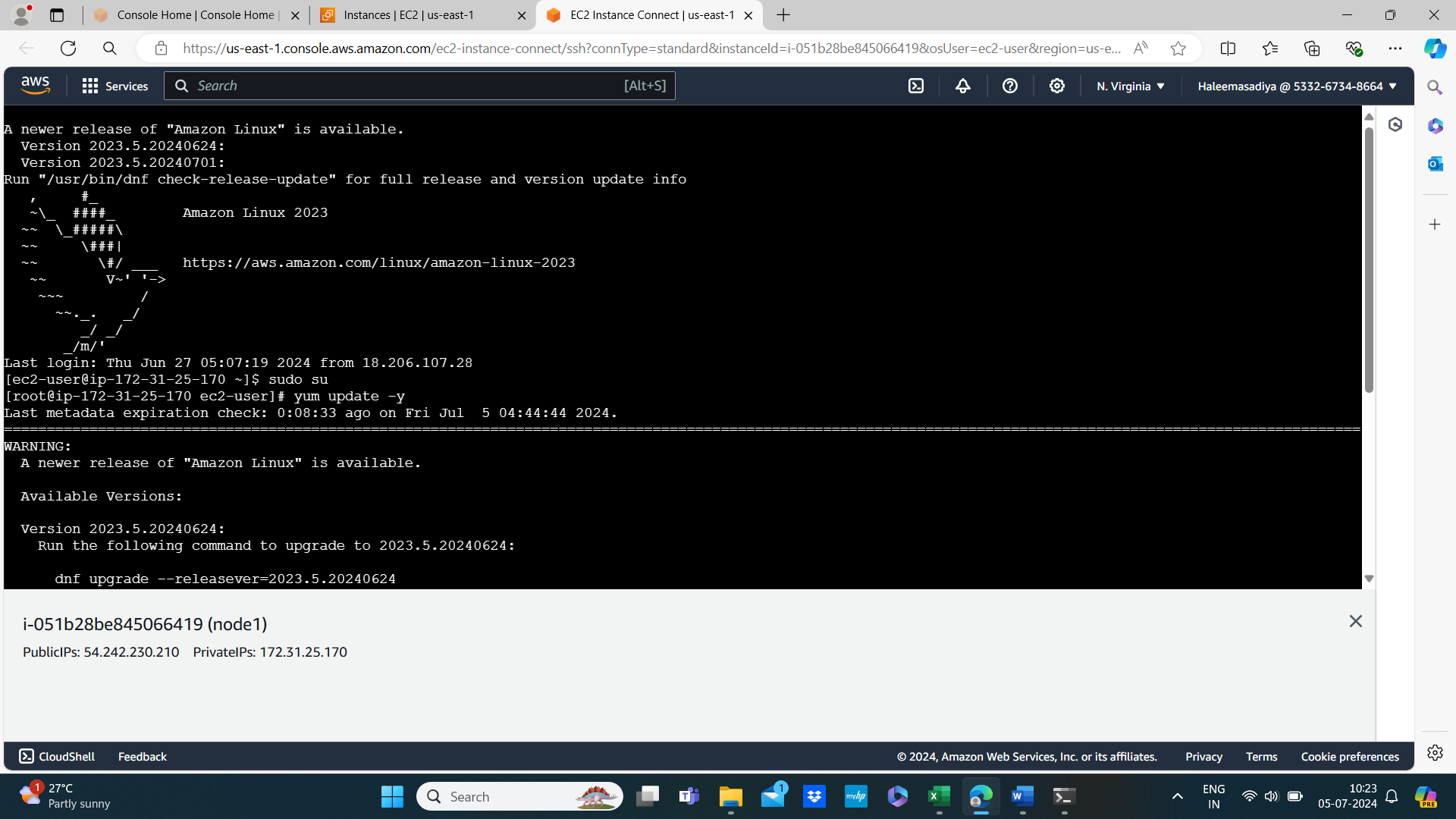
# ssh <private\_ip> 🡪in this time no password required we can easily connect to nodes with out enter password again and again

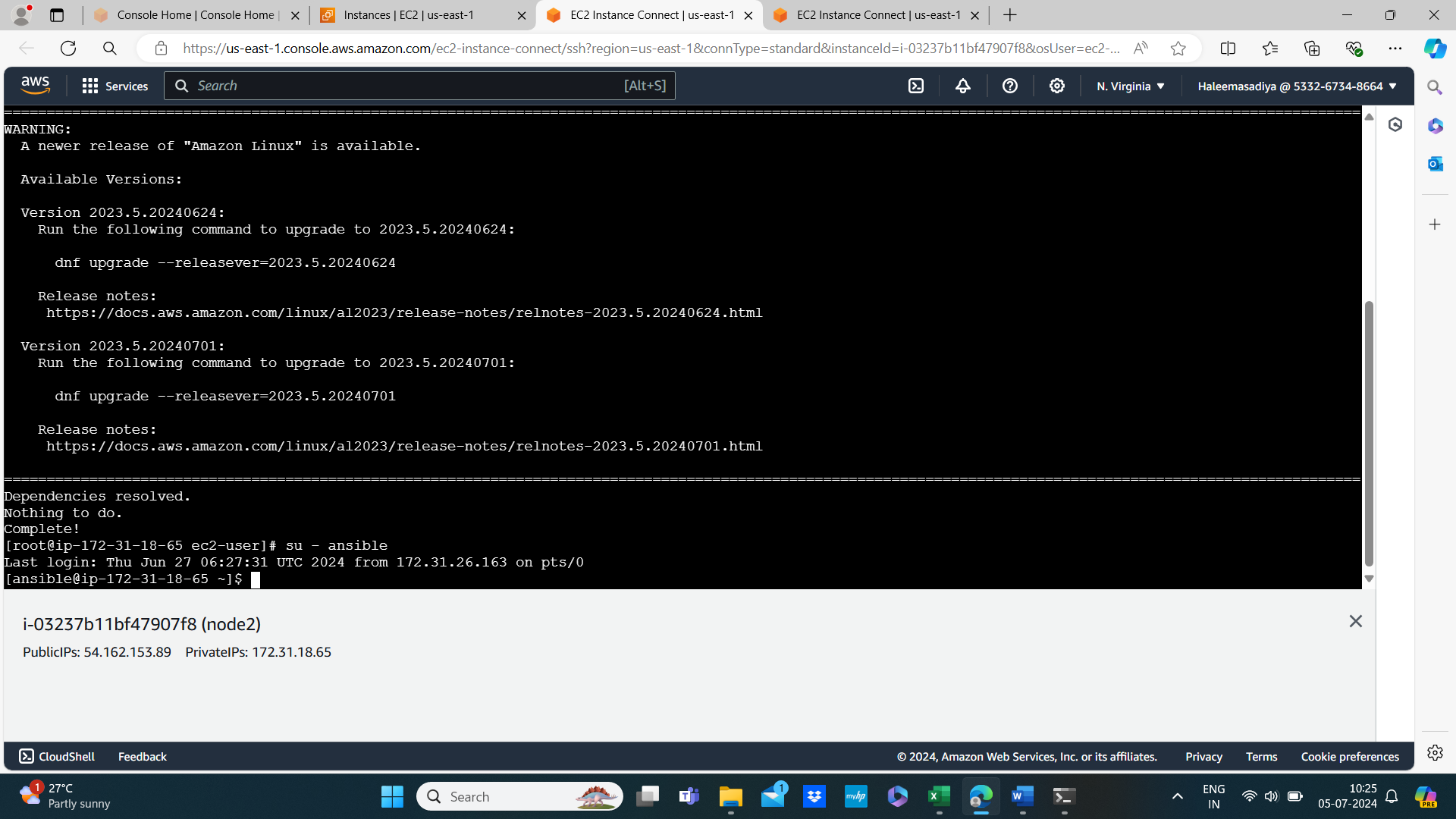
Then update the instances yum update -y

Then become ansible user su - <user\_name>

Perform above three actions in three instances







Host pattern:

A host pattern is a way to match one or more hosts from your inventory file (a list of all the machines you manage with Ansible). It allows you to control where your Ansible tasks are executed.

**Why Use Host Patterns?**

* **Flexibility:** Easily target specific hosts or groups without changing your playbooks.
* **Scalability:** Manage large environments efficiently by grouping hosts and using patterns.
* **Control:** Exclude certain hosts or run tasks on multiple hosts/groups simultaneously.

# ansible -I /etc/ansible/hosts all –list-host

# ansible -I /etc/ansible/hosts <[group\_name]> --list-host

# ansible -I /etc/ansible/hosts group\_name[-1] --list-host 🡪last machine

# ansible -I /etc/ansible/hosts group\_name[0] --list-host 🡪 first machine

There are 3 ways of pushing the codes into nodes

1. Ad-hoc commands 2. Modules 3. Playbook

Ad-hoc commands:

Ad hoc commands in Ansible are powerful tools for executing one-time tasks, troubleshooting, and making quick changes across your infrastructure. They provide the flexibility to perform operations on multiple hosts directly from the command line without needing to write a playbook.

**Why Use Ad Hoc Commands?**

* **Immediate Actions:** Perfect for immediate actions that don’t justify a full playbook.
* **Troubleshooting:** Quickly diagnose and fix issues.
* **Simplicity:** Easy to run and don’t require setup or configuration.

Syntax: ansible <host-pattern> -m <module> -a "<arguments>"

Modules:

Modules in Ansible are the building blocks that perform specific tasks on your managed hosts. They are small programs that Ansible runs on your hosts to perform actions like installing packages, copying files, managing services, and more.

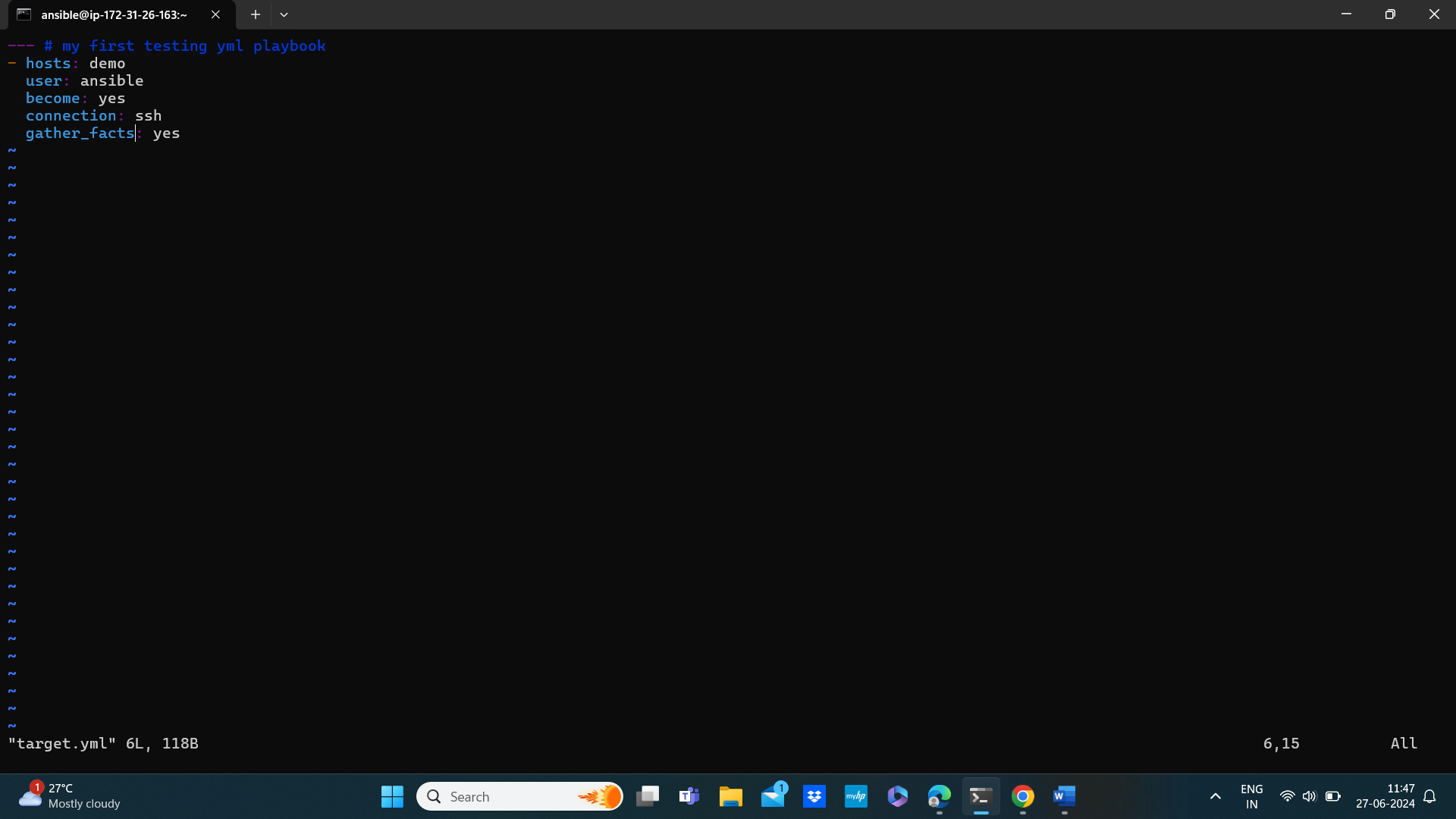
**What are Modules?**

* **Small Programs:** Each module is a small, task-specific program written to do one thing well.
* **Reusable:** Modules can be used in multiple playbooks and ad hoc commands.
* **Broad Functionality:** Ansible comes with many built-in modules that cover a wide range of tasks.

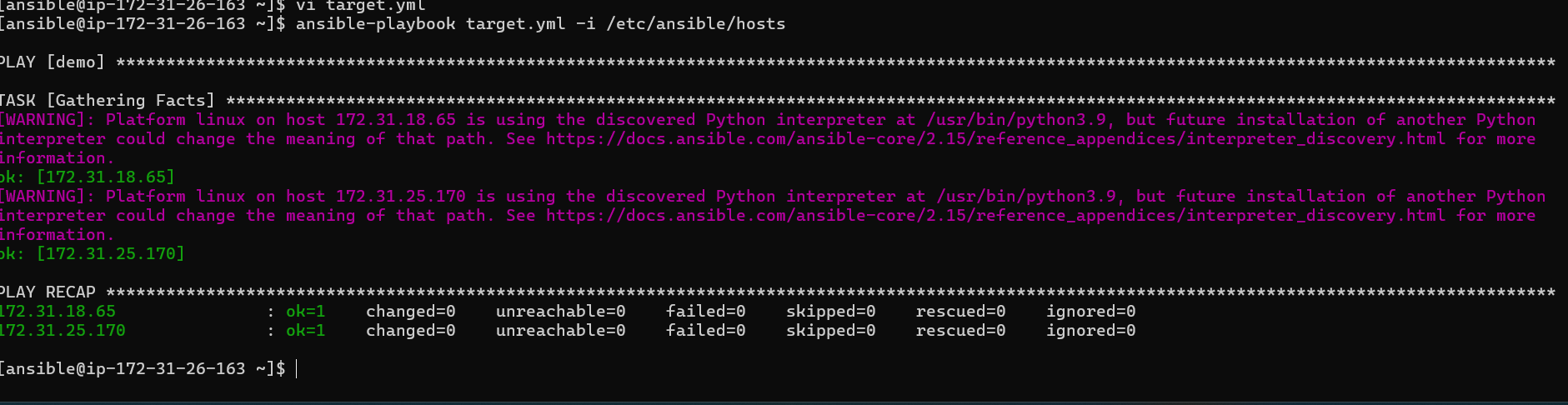
### What is a Playbook?

* **YAML File:** Playbooks are written in YAML, which is a human-readable data serialization format.
* **Series of Tasks:** A playbook contains one or more "plays", and each play contains a series of tasks.
* **Configuration Management:** Playbooks define the desired state of your systems and automate their configuration.

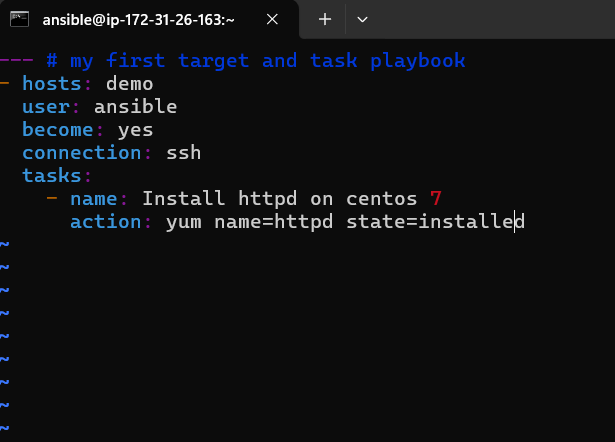
Vi target.yml



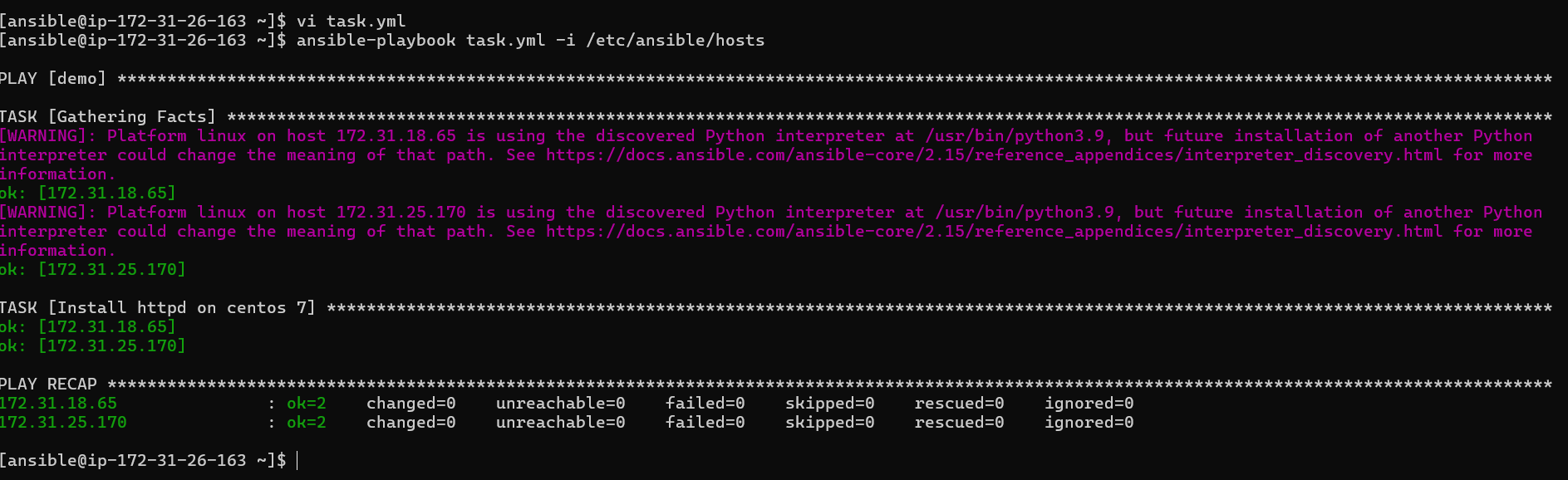
Ansible-playbook target.yml -I /etc/ansible/hosts



Vi task.yml

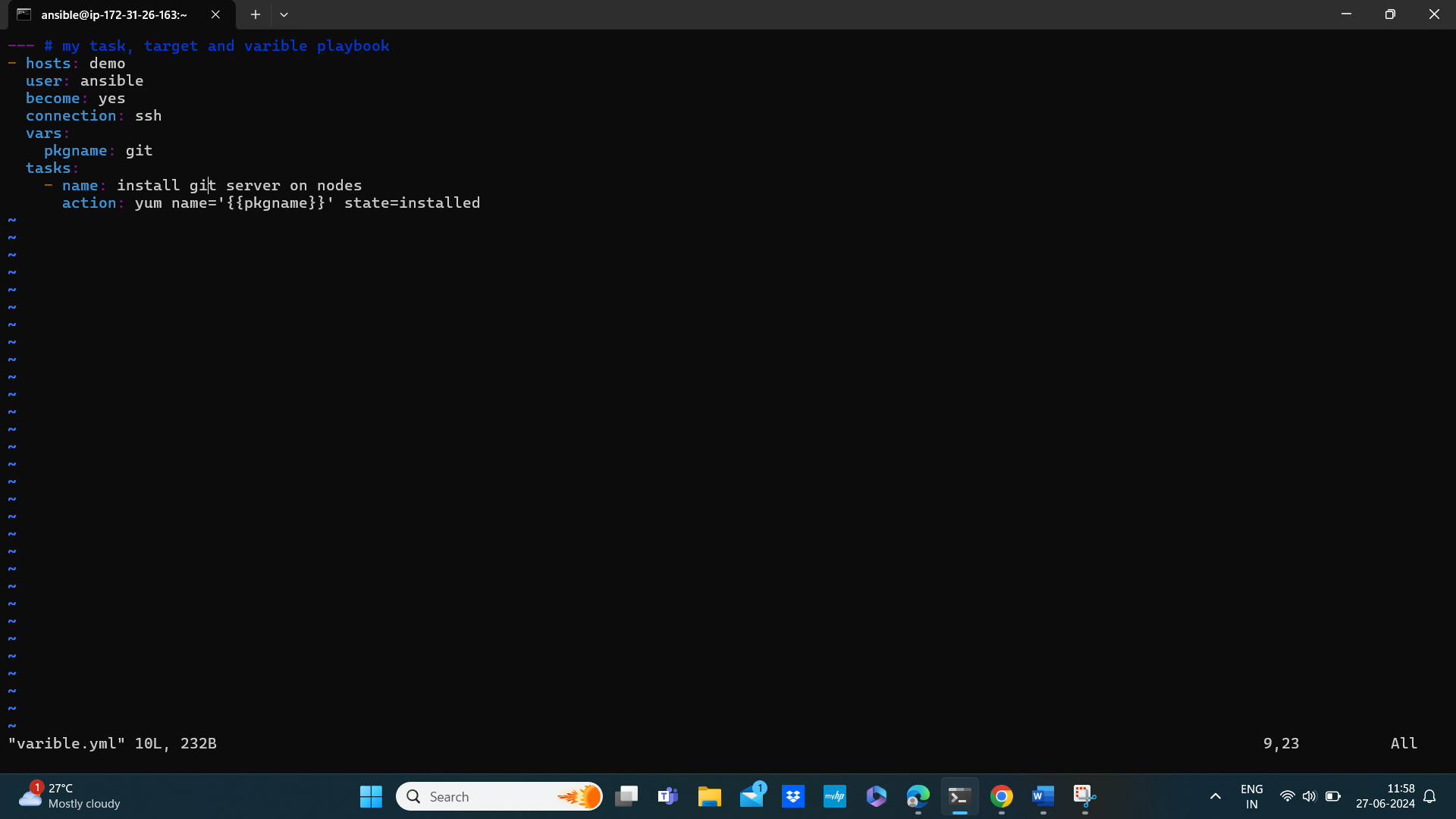


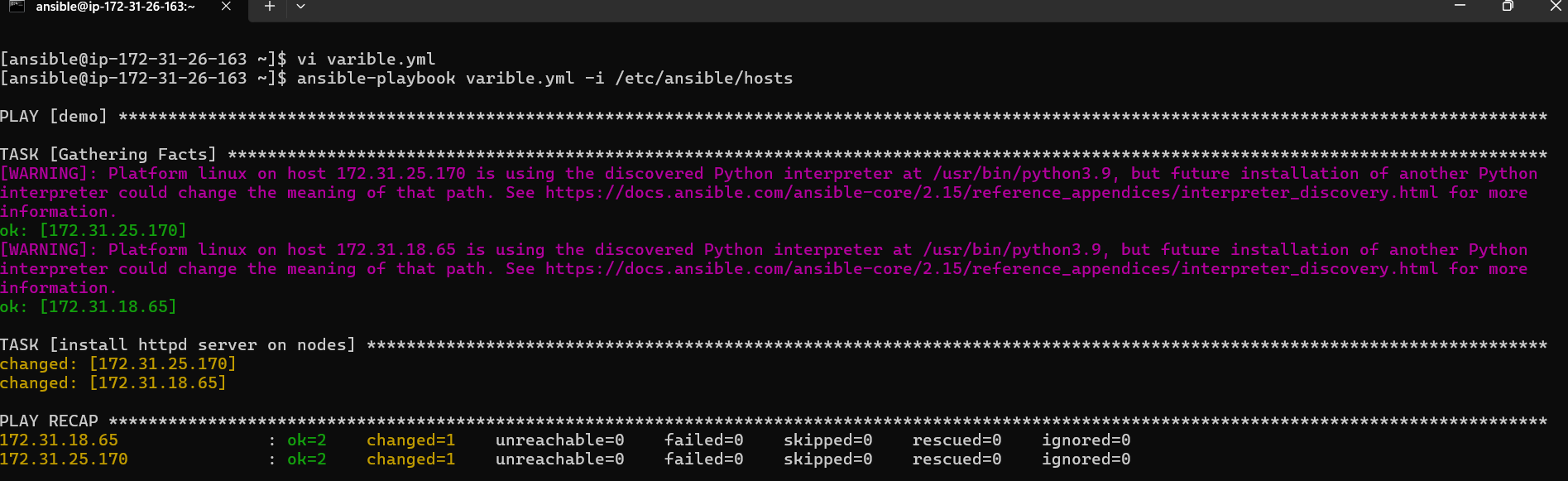
Ansible-playbook vi task.yml -i /etc/ansible/hosts



Variable: Variables in Ansible are used to store values that can be reused and dynamically substituted within playbooks, tasks, and templates to customize automation workflows.

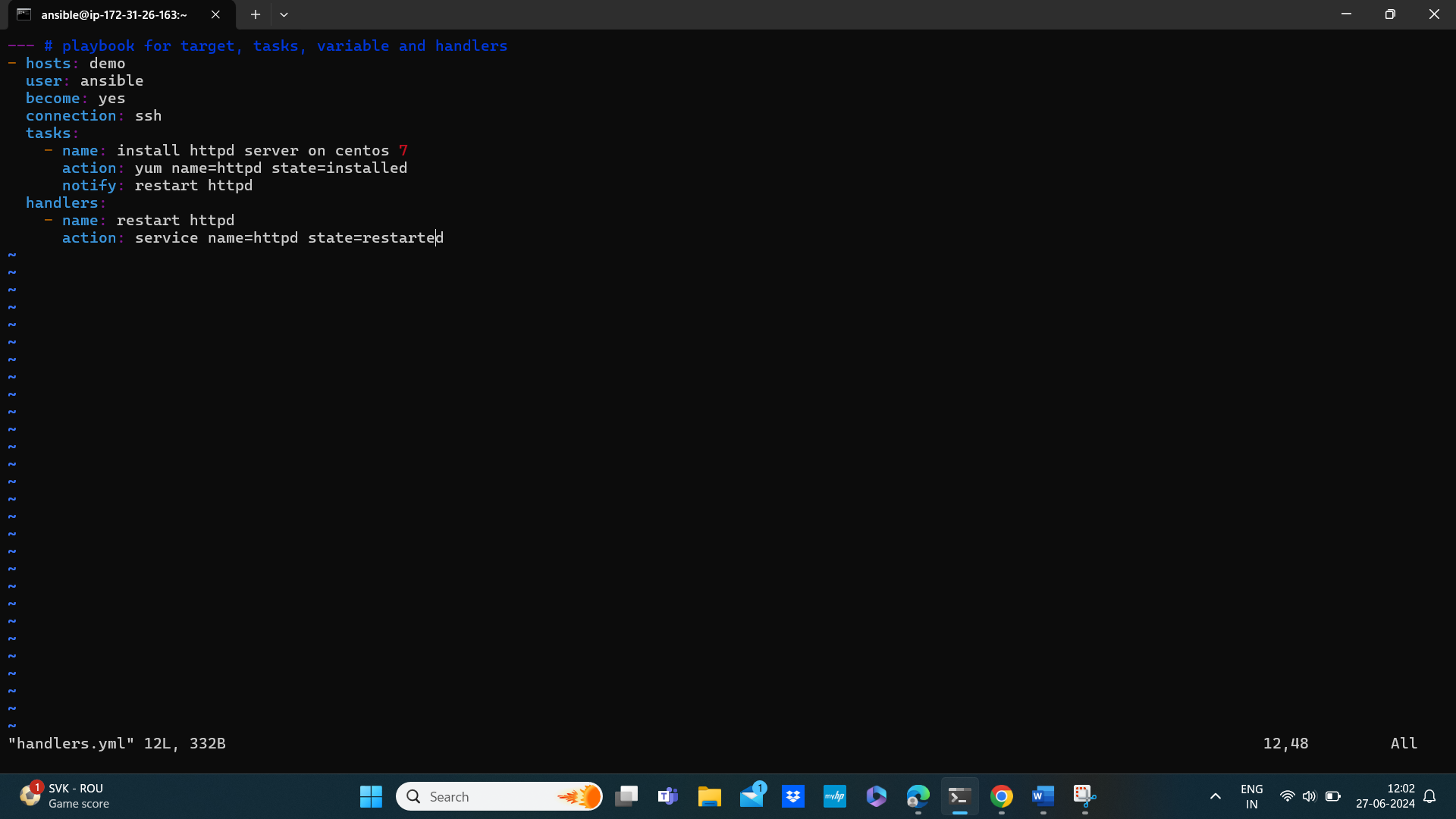
Vi varible.yml

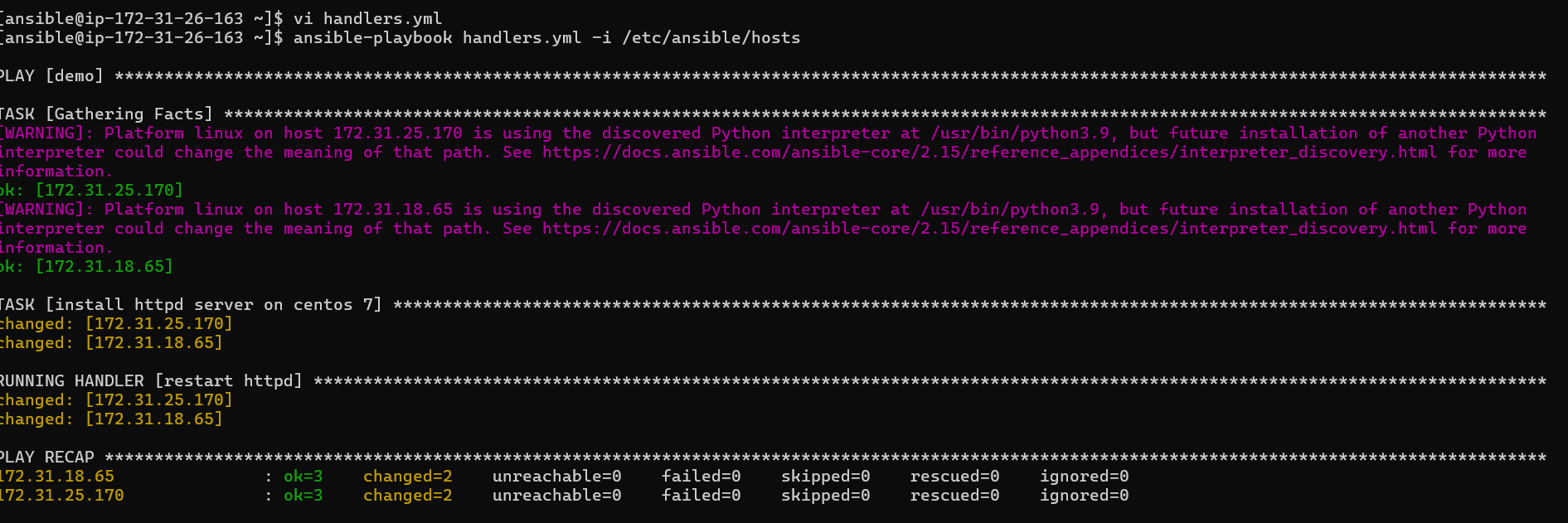




Handlers: Handlers in Ansible are special tasks that are triggered by other tasks when a change occurs, typically used to restart services or reload configurations only when necessary.

Vi handlers.yml





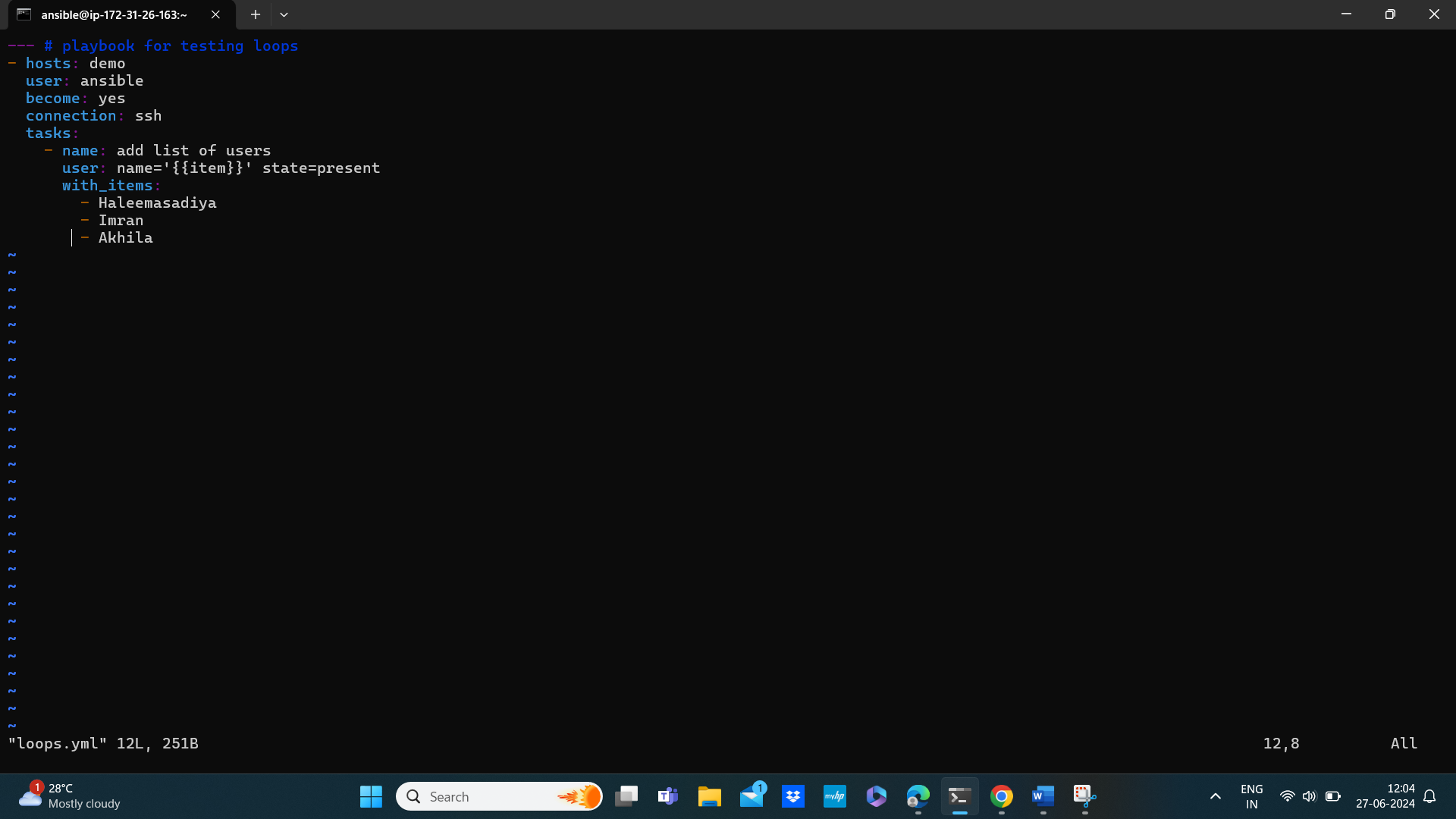
Dry run: check whether the playbook is formatted correctly

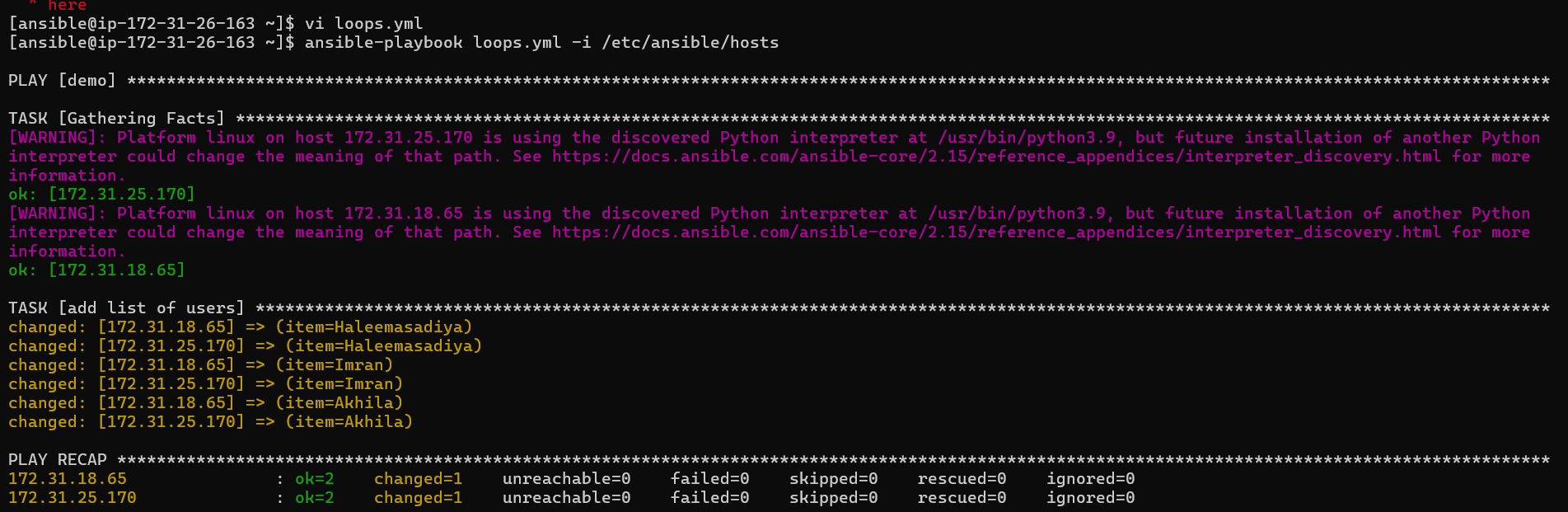
# ansible-playbook handler.yml -I /etc/ansible/hosts –check

Loops:

Loops in Ansible allow you to iterate over a list of items to perform the same task multiple times with different inputs.

Vi loops.yml

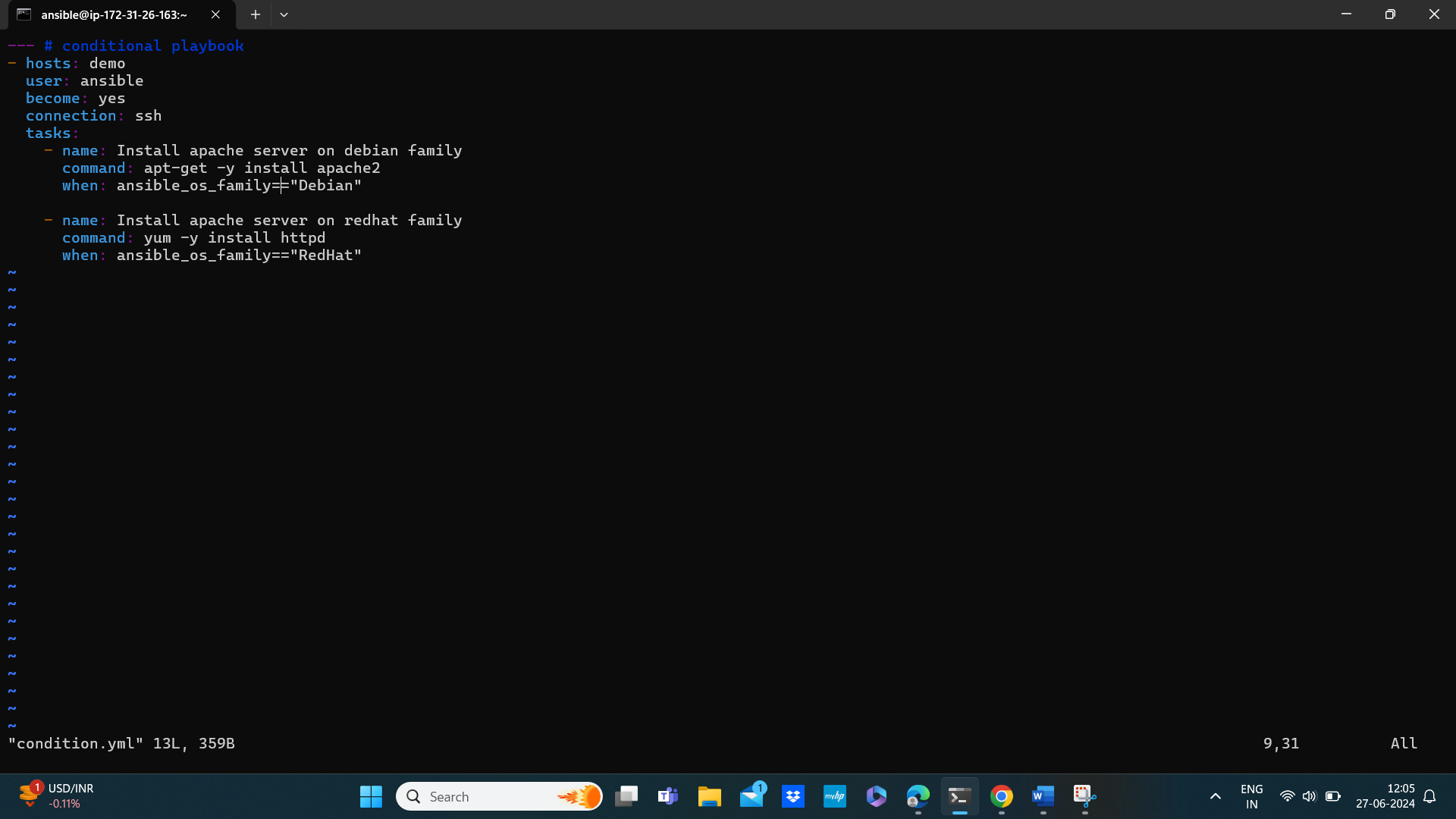


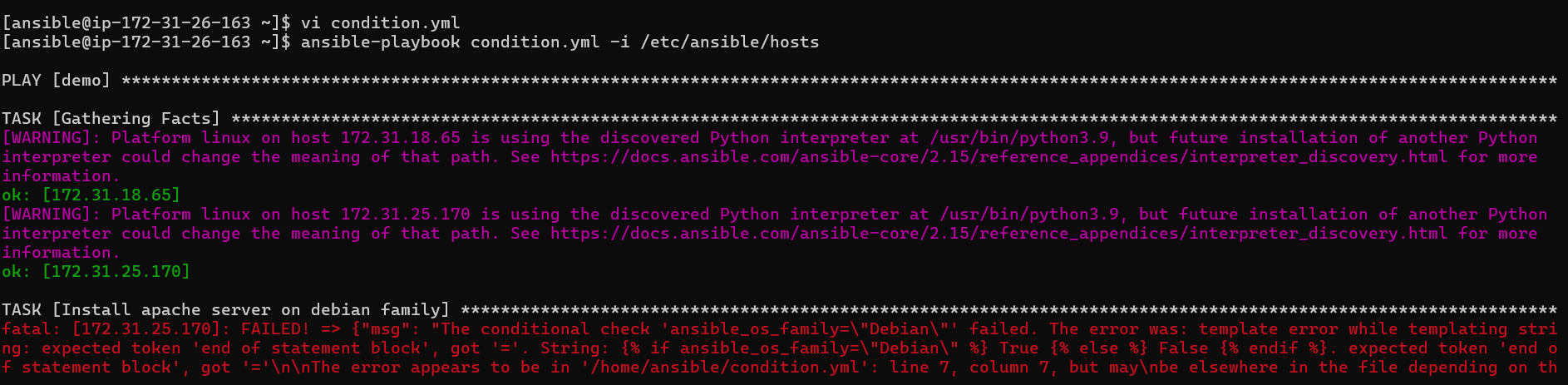


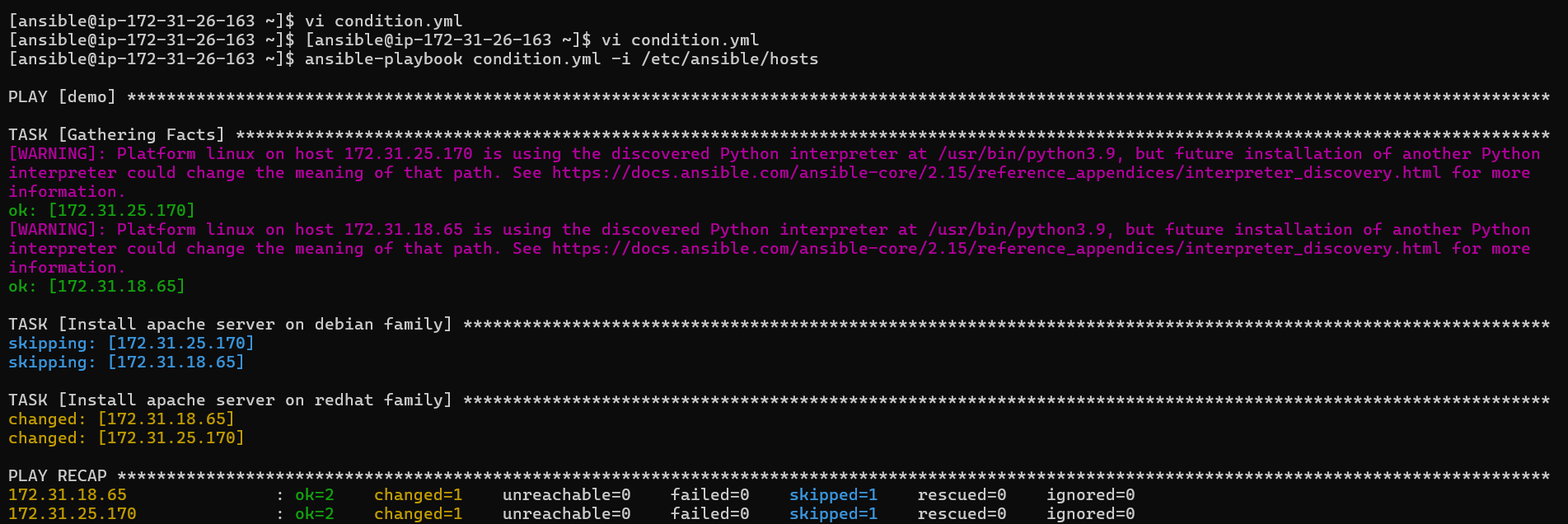
Conditions:

Conditions in Ansible allow you to control the execution of tasks based on certain criteria. This is primarily achieved using the when keyword, which lets you specify a condition that must be true for the task to run.

Vi conditions.yml

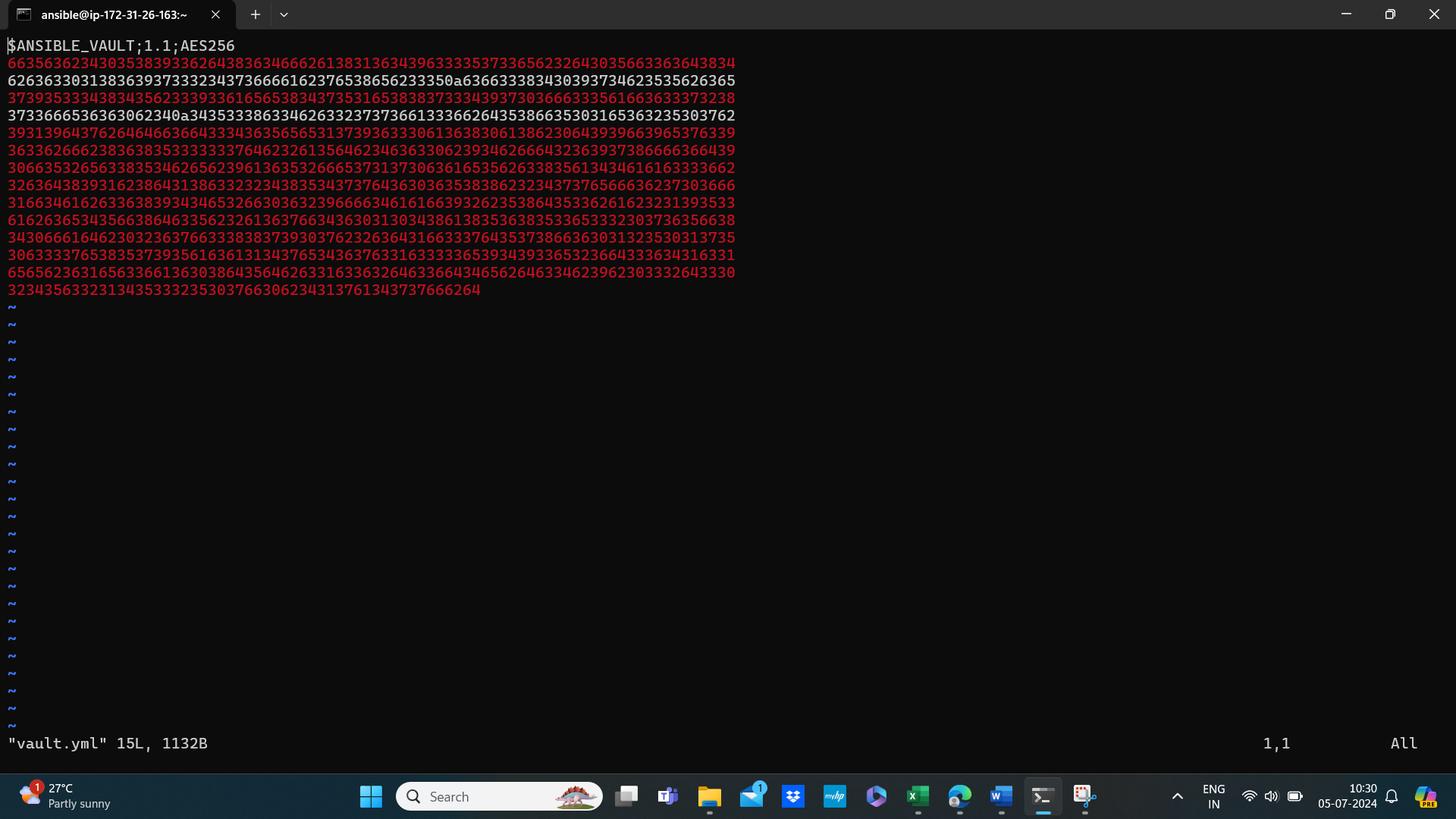




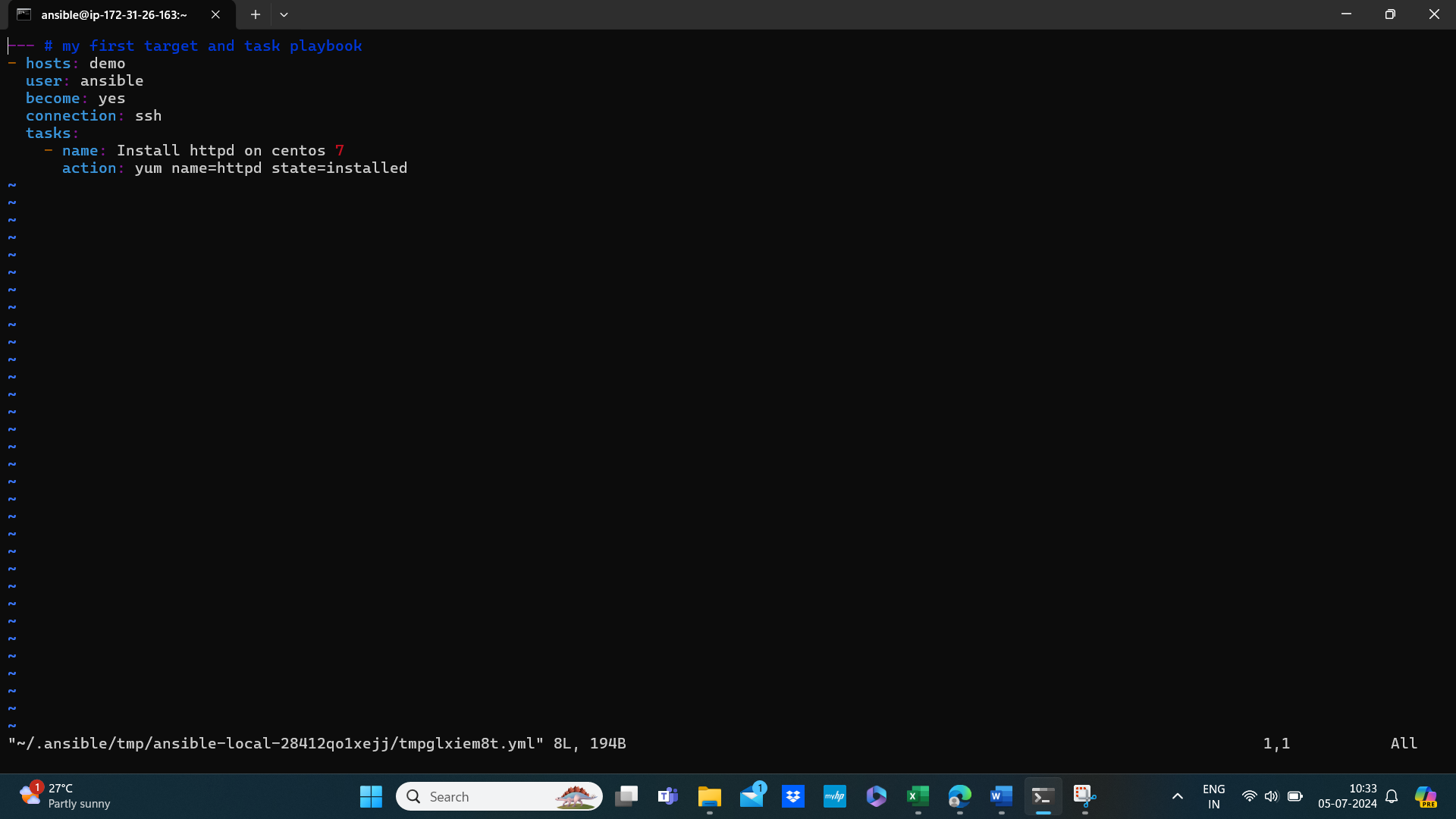


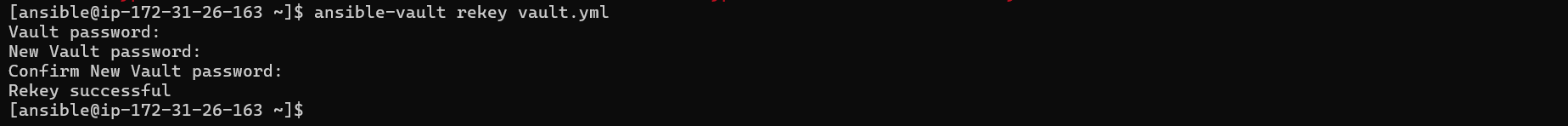
Vault: Ansible Vault is a feature that allows you to encrypt and securely store sensitive data, such as passwords and keys, within your playbooks and files.











* To encrypt an existing playbook

# ansible-vault encrypt target.yml

* To decrypt an encrypted playbook

# ansible-vault decrypt target.yml