www.linkedin.com/in/arun-kumar-akula

ANSIBLE – Configuration Management Tool

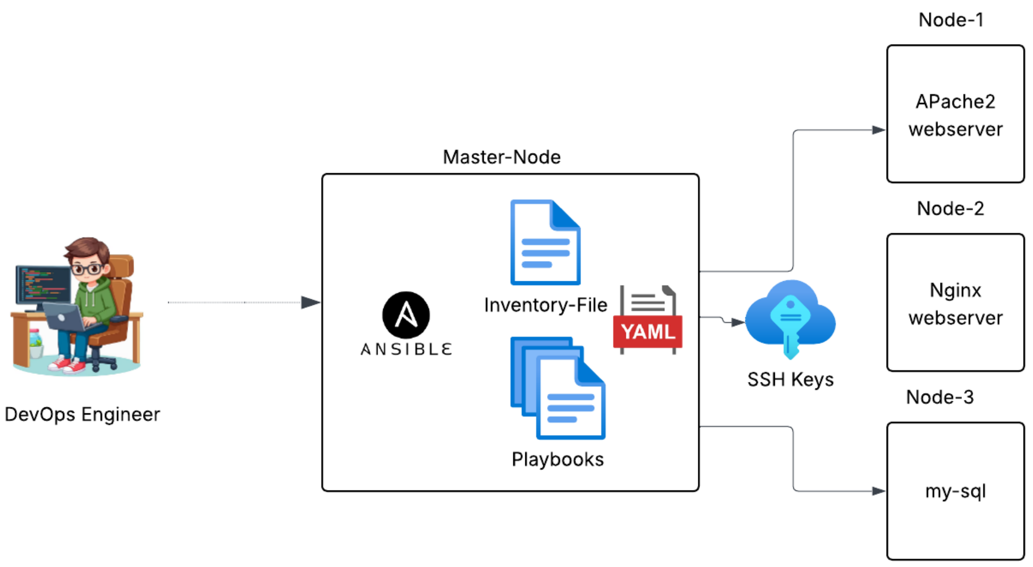
Ansible is an open-source automation tool used for configuration management, application deployment, task automation, and orchestration**.**

- It is an **agentless tool**. Ansible does not require any agent or additional software to be installed on the target nodes. It communicates with them using SSH.
- It is **Idempotent** Ensures the same result every time you run the automation.
- It allows us to automate repetitive tasks and manage servers efficiently.
- Ansible uses simple language like YAML for playbooks, which is easy to read and write.

Commonly we use Ansible for:

- Installing and configuring software like Apache, MySQL, etc.
- Deploying apps across multiple servers.
- Automate daily tasks like updates and backups.
- Manage multiple services together like start database before web server.

Ansible Architecture Overview :



- **Master Node:** The machine where Ansible is installed and from where tasks are executed.
- **Nodes:** The target machines (servers, VMs) that Ansible manages.
- **Inventory File:** A file that lists the managed nodes (IPs/hostnames), grouped logically.
- **Playbooks:** YAML files containing a set of tasks that define the automation steps.
- **SSH:** The secure connection method used to communicate with managed nodes.

WorkFlow:

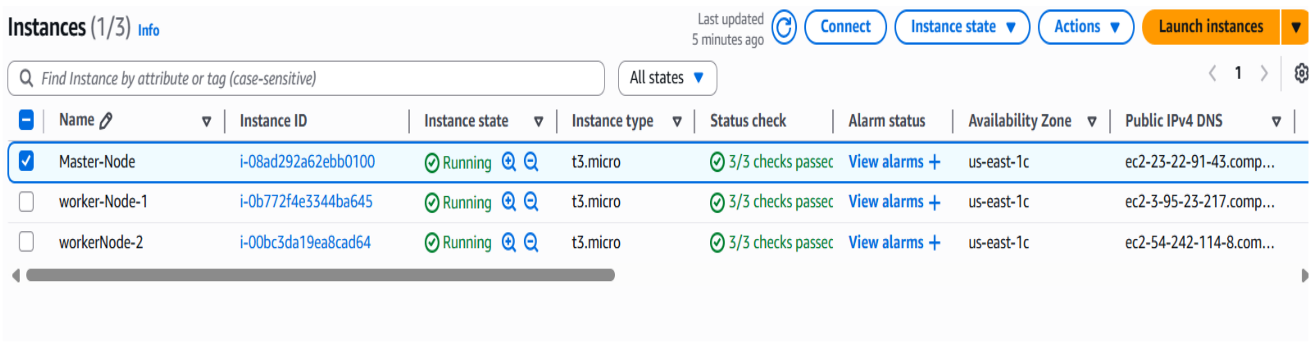
- User writes a Playbook on the Control Node.
- The Inventory File defines the target hosts.
- The Playbook is executed using the ansible-playbook command.
- Ansible connects to Managed Nodes over SSH.
- Tasks are executed using Ansible modules.
- Results are displayed on the Control Node terminal.

Installation and Setup of Ansible :

To use Ansible, you only need to install it on the Control Node. The Managed Nodes do not require any software installations just SSH access.

Step1: Installing Ansible on Ubuntu (Control Node)

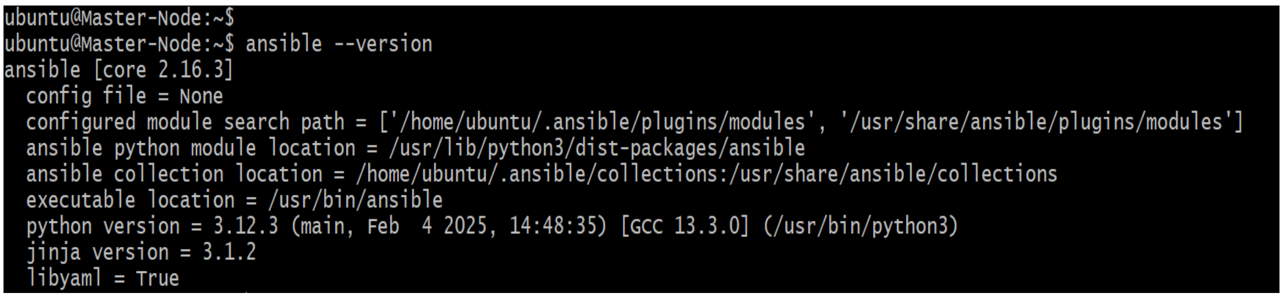
- Launch 3 EC2 Instances with security ports (ssh & http) (Master Node + Managed Nodes)


Instances (1/3) Info
Last updated 5 minutes ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
1

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
Master-Node	i-08ad292a62ebb0100	Running	t3.micro	3/3 checks passec	View alarms +	us-east-1c	ec2-23-22-91-43.comp...
worker-Node-1	i-0b772f4e3344ba645	Running	t3.micro	3/3 checks passec	View alarms +	us-east-1c	ec2-3-95-23-217.comp...
workerNode-2	i-00bc3da19ea8cad64	Running	t3.micro	3/3 checks passec	View alarms +	us-east-1c	ec2-54-242-114-8.com...

- Connect to EC2- Master Node Instance via SSH
- Change the host-name using command (**sudo vi /etc/hostname**) to avoid confusion.
- Update the ubuntu server using command: **sudo apt update**

- Install Ansible server using command: **sudo apt install ansible -y**
- Check the ansible version using command: **ansible –version**

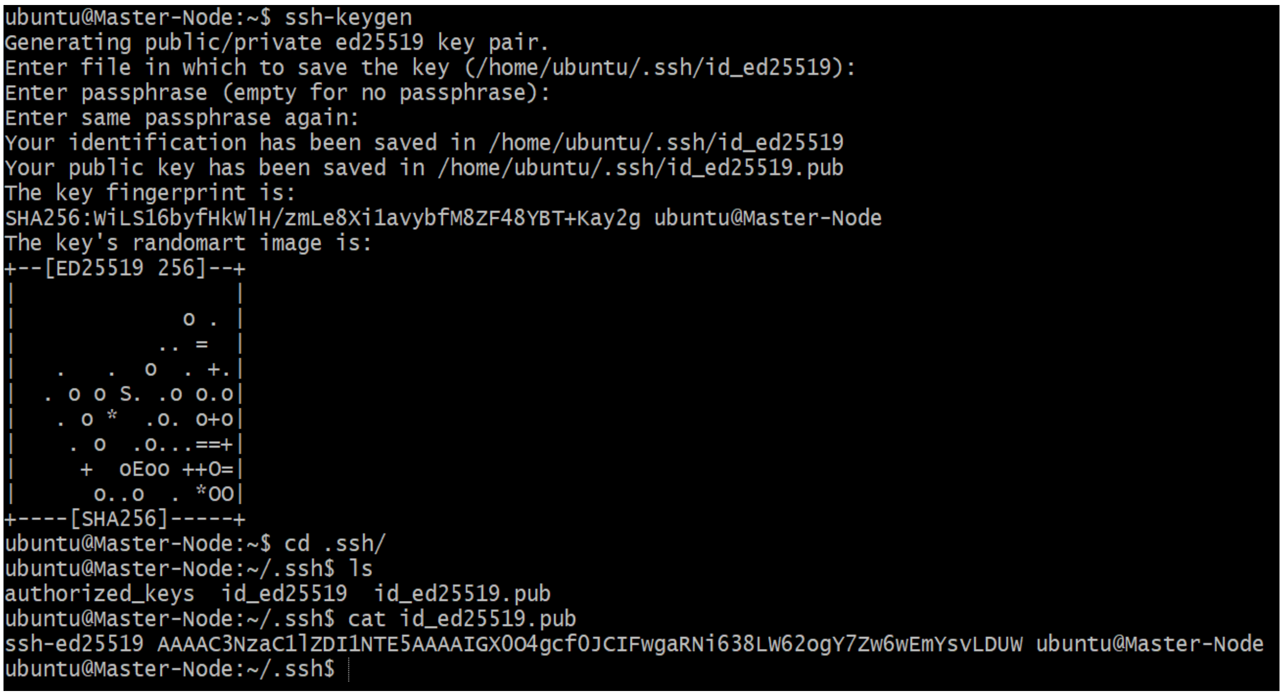


```
ubuntu@Master-Node:~$
ubuntu@Master-Node:~$ ansible --version
ansible [core 2.16.3]
  config file = None
  configured module search path = ['/home/ubuntu/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/ubuntu/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Feb  4 2025, 14:48:35) [GCC 13.3.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
```

Step 2: Setup SSH to Managed Nodes from Control Node

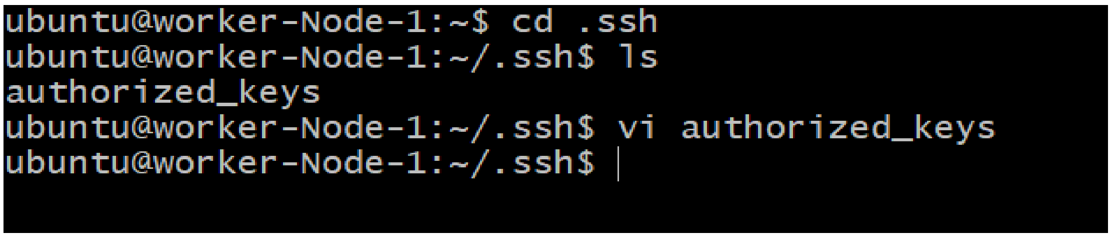
- We have two ways to manage the nodes from the Ansible control node.
  - Copy the SSH key from the local machine to the EC2 instance using ssh-copy-id.
  - Manually generate an SSH key and add the public key to the EC2 instances ~/.ssh/authorized_keys file.

- Generate ssh-key using command: **ssh-keygen (** This command generates a public and private key pair. Copy the public key and paste it into the ~/.ssh/authorized_keys file on the worker nodes)



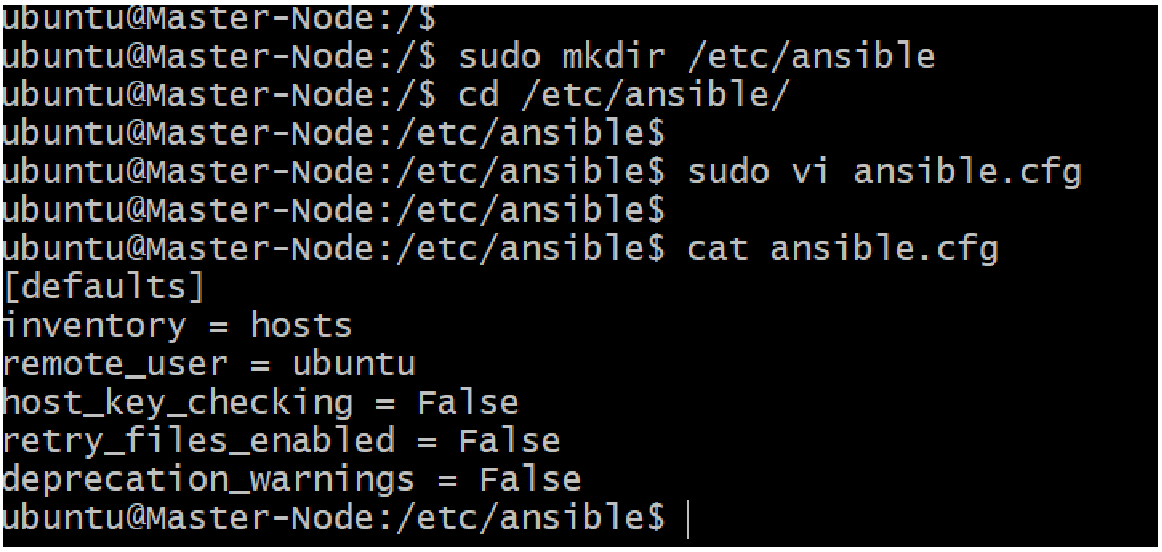
```
ubuntu@Master-Node:~$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_ed25519
Your public key has been saved in /home/ubuntu/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:WiLS16byfHkWlH/zmLe8Xi1avybfM8ZF48YBT+Kay2g ubuntu@Master-Node
The key's randomart image is:
+--[ED25519 256]--+
|                 |
|            o .  |
|          .. =   |
|   .   .  o  . +.|
|  . o o S.  .o o.o|
|   . o *   .o. o+o|
|    . o  .o...==+|
|     +  oEoo ++O=|
|      o..o  . *OO|
+----[SHA256]-----+
ubuntu@Master-Node:~$ cd .ssh/
ubuntu@Master-Node:~/.ssh$ ls
authorized_keys  id_ed25519  id_ed25519.pub
ubuntu@Master-Node:~/.ssh$ cat id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGXOO4gcfOJCIFwgaRNi638LW62ogY7Zw6wEmYsvLDUW ubuntu@Master-Node
ubuntu@Master-Node:~/.ssh$
```

- Check the public key using command: cat file-name and copy the key paste it on the node servers



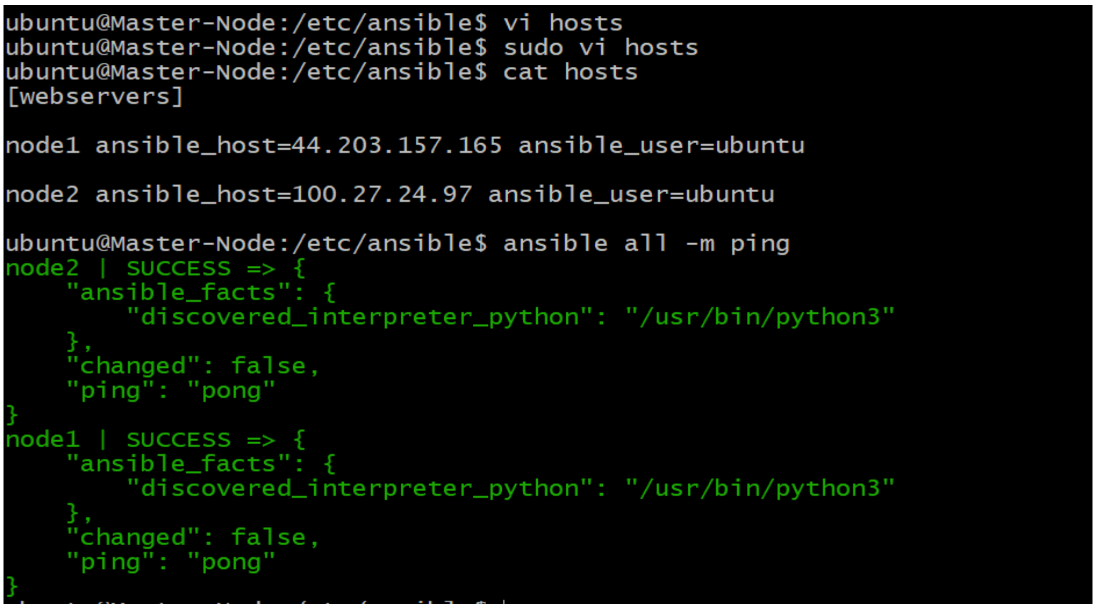
```
ubuntu@worker-Node-1:~$ cd .ssh
ubuntu@worker-Node-1:~/.ssh$ ls
authorized_keys
ubuntu@worker-Node-1:~/.ssh$ vi authorized_keys
ubuntu@worker-Node-1:~/.ssh$
```

- Create a **ansible.cfg** file, file is used to customize and control Ansible’s behavior. While Ansible works with default settings.
  - Create a config file in the path: **/etc/ansible**
  - In the config.cfg file try to give the below settings.



```
ubuntu@Master-Node:/$
ubuntu@Master-Node:/$ sudo mkdir /etc/ansible
ubuntu@Master-Node:/$ cd /etc/ansible/
ubuntu@Master-Node:/etc/ansible$
ubuntu@Master-Node:/etc/ansible$ sudo vi ansible.cfg
ubuntu@Master-Node:/etc/ansible$
ubuntu@Master-Node:/etc/ansible$ cat ansible.cfg
[defaults]
inventory = hosts
remote_user = ubuntu
host_key_checking = False
retry_files_enabled = False
deprecation_warnings = False
ubuntu@Master-Node:/etc/ansible$
```

- **Create a hosts file and configure node details**, The hosts file also called the inventory file tells Ansible which servers to manage.
- The hosts file path should be in **/etc/ansible/hosts**

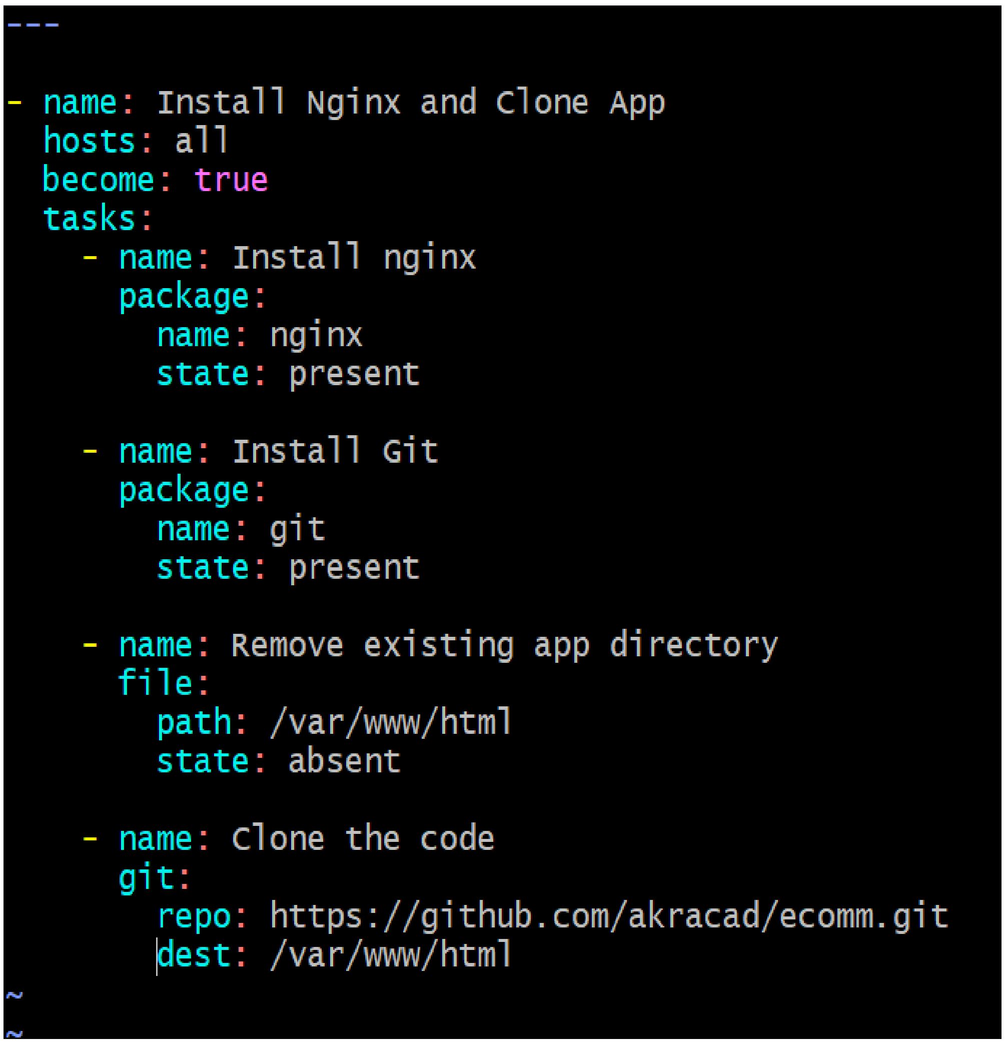


```
ubuntu@Master-Node:/etc/ansible$ vi hosts
ubuntu@Master-Node:/etc/ansible$ sudo vi hosts
ubuntu@Master-Node:/etc/ansible$ cat hosts
[webservers]

node1 ansible_host=44.203.157.165 ansible_user=ubuntu

node2 ansible_host=100.27.24.97 ansible_user=ubuntu

ubuntu@Master-Node:/etc/ansible$ ansible all -m ping
node2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
node1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

- Let’s try to ping, Ansible will connect to both servers in the webservers group.
- Command to ping: ansible all -m ping
- As we can see, our servers have successfully connected.

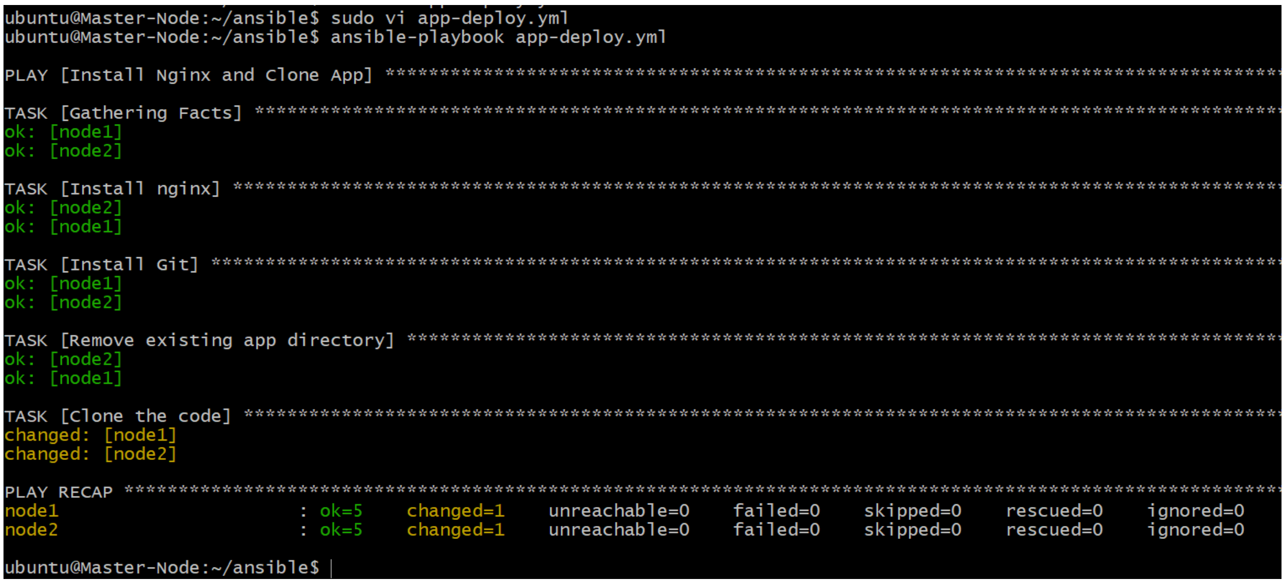
Step 3: let’s try write a Ansible playbook to install Nginx, Git, and clone a GitHub app to /var/www/html:

- Let’s create a ansible playbook using command**: sudo vi app-deploy.yml**



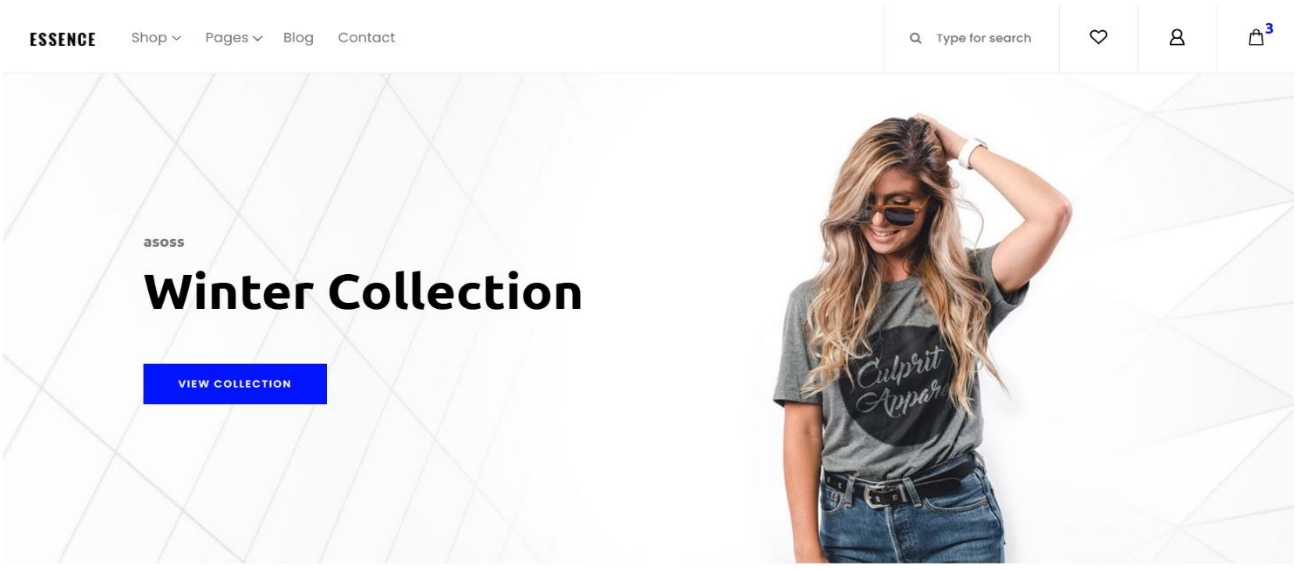
```
---

- name: Install Nginx and Clone App
  hosts: all
  become: true
  tasks:
    - name: Install nginx
      package:
        name: nginx
        state: present

    - name: Install Git
      package:
        name: git
        state: present

    - name: Remove existing app directory
      file:
        path: /var/www/html
        state: absent

    - name: Clone the code
      git:
        repo: https://github.com/akracad/ecomm.git
        dest: /var/www/html
~
~
```

- Now the apply the playbook using command: **ansible-playbook file-name**
- We can observer here the playbook successfully executed.



```
ubuntu@Master-Node:~/ansible$ sudo vi app-deploy.yml
ubuntu@Master-Node:~/ansible$ ansible-playbook app-deploy.yml

PLAY [Install Nginx and Clone App] ***********************************************************************

TASK [Gathering Facts] ***********************************************************************************
ok: [node1]
ok: [node2]

TASK [Install nginx] *************************************************************************************
ok: [node2]
ok: [node1]

TASK [Install Git] ***************************************************************************************
ok: [node1]
ok: [node2]

TASK [Remove existing app directory] *********************************************************************
ok: [node2]
ok: [node1]

TASK [Clone the code] ************************************************************************************
changed: [node1]
changed: [node2]

PLAY RECAP ***********************************************************************************************
node1                      : ok=5    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
node2                      : ok=5    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ubuntu@Master-Node:~/ansible$ |
```

- Now check the nginx version in the Nodes using **Nginx –version**
- Now try to Access the application whether it is running or not.!

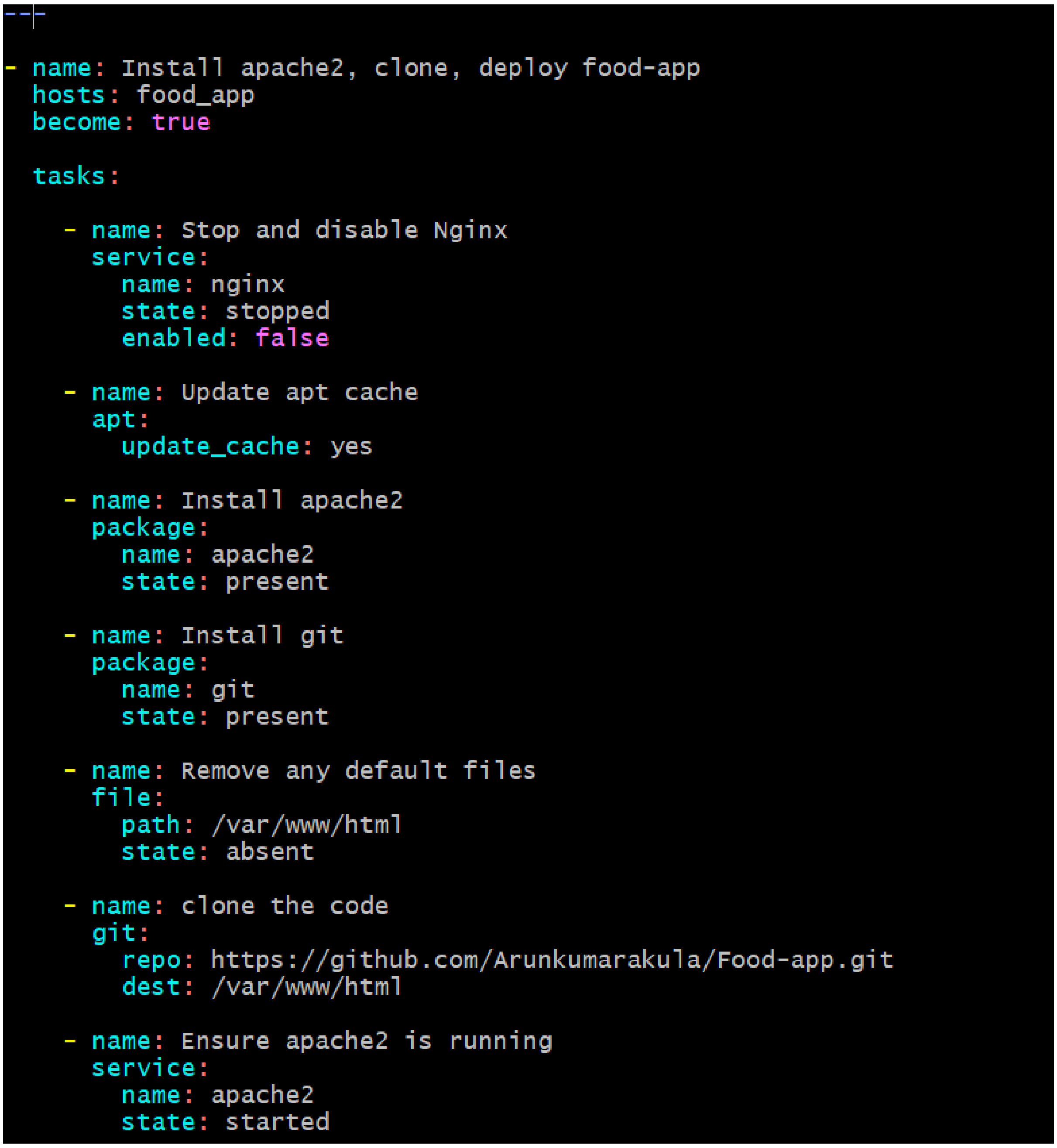


- Successfully running the ecomm application.!

www.linkedin.com/in/arun-kumar-akula

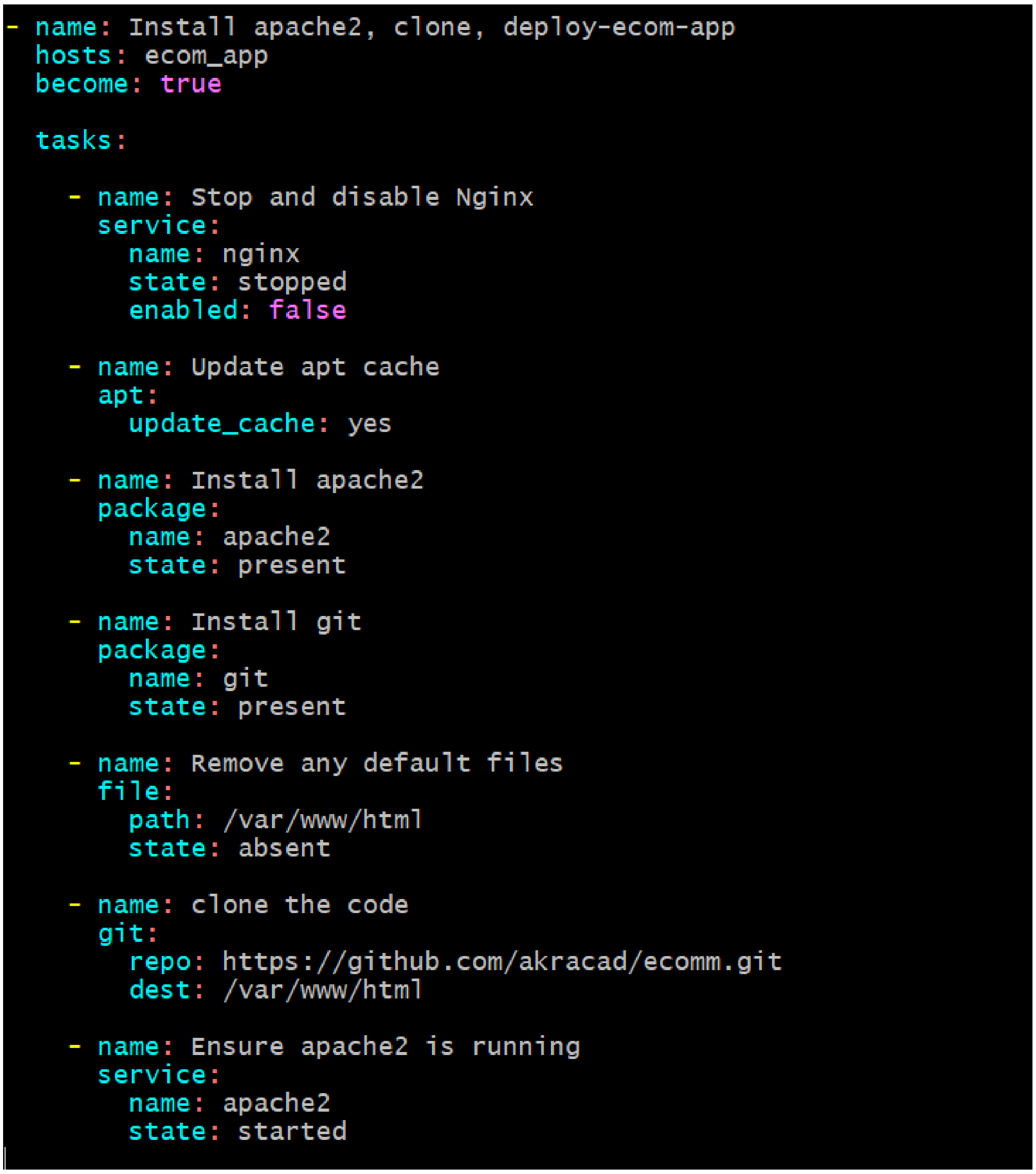
let's try installing Apache2 and deploying a different application on each server.

Step1 : create a playbook eg: food-ecom-app.yml



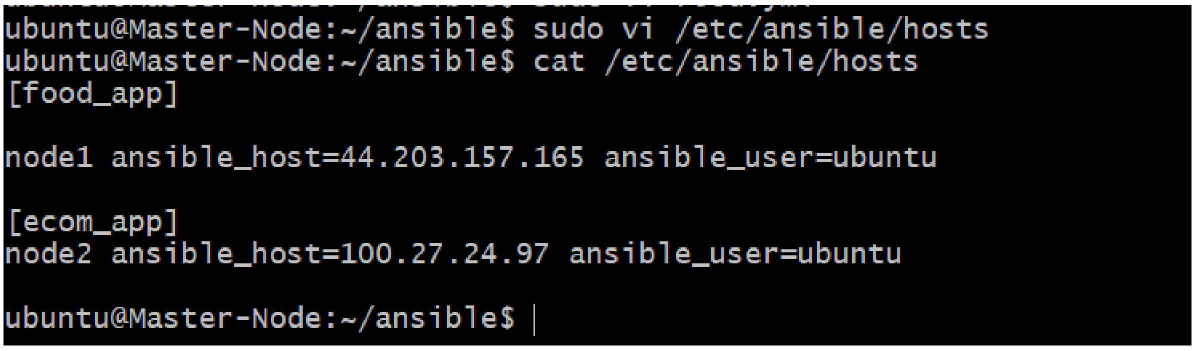
```
---

- name: Install apache2, clone, deploy food-app
  hosts: food_app
  become: true

  tasks:

    - name: Stop and disable Nginx
      service:
        name: nginx
        state: stopped
        enabled: false

    - name: Update apt cache
      apt:
        update_cache: yes

    - name: Install apache2
      package:
        name: apache2
        state: present

    - name: Install git
      package:
        name: git
        state: present

    - name: Remove any default files
      file:
        path: /var/www/html
        state: absent

    - name: clone the code
      git:
        repo: https://github.com/Arunkumarakula/Food-app.git
        dest: /var/www/html

    - name: Ensure apache2 is running
      service:
        name: apache2
        state: started
```

- Continued yml file



```
- name: Install apache2, clone, deploy-ecom-app
  hosts: ecom_app
  become: true

  tasks:

    - name: Stop and disable Nginx
      service:
        name: nginx
        state: stopped
        enabled: false

    - name: Update apt cache
      apt:
        update_cache: yes

    - name: Install apache2
      package:
        name: apache2
        state: present

    - name: Install git
      package:
        name: git
        state: present

    - name: Remove any default files
      file:
        path: /var/www/html
        state: absent

    - name: clone the code
      git:
        repo: https://github.com/akracad/ecomm.git
        dest: /var/www/html

    - name: Ensure apache2 is running
      service:
        name: apache2
        state: started
```

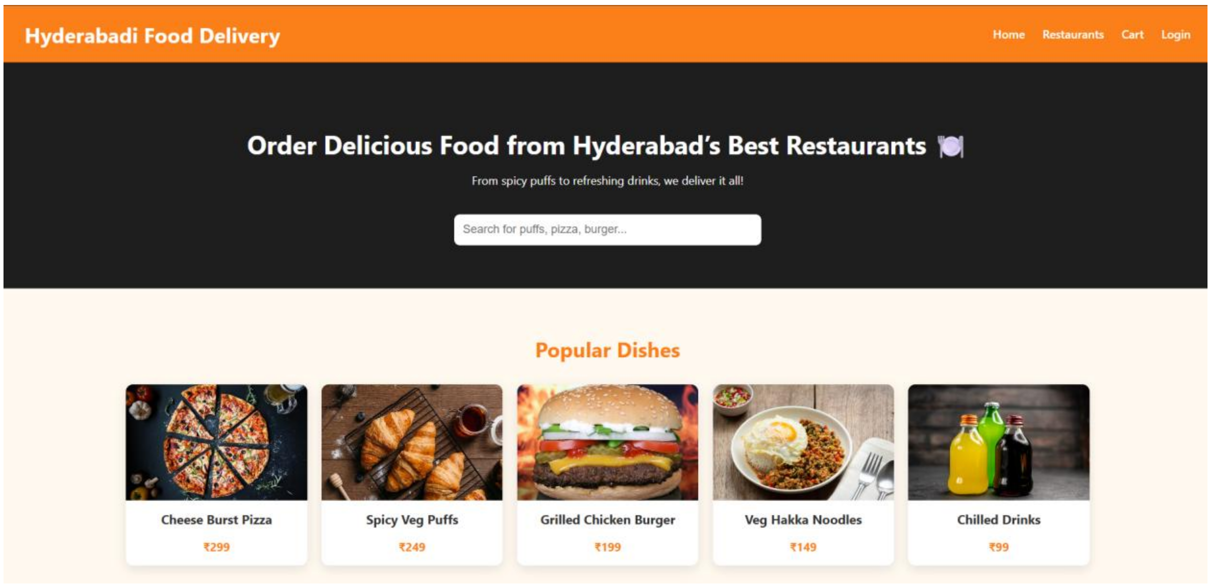
- Now change the hosts file .



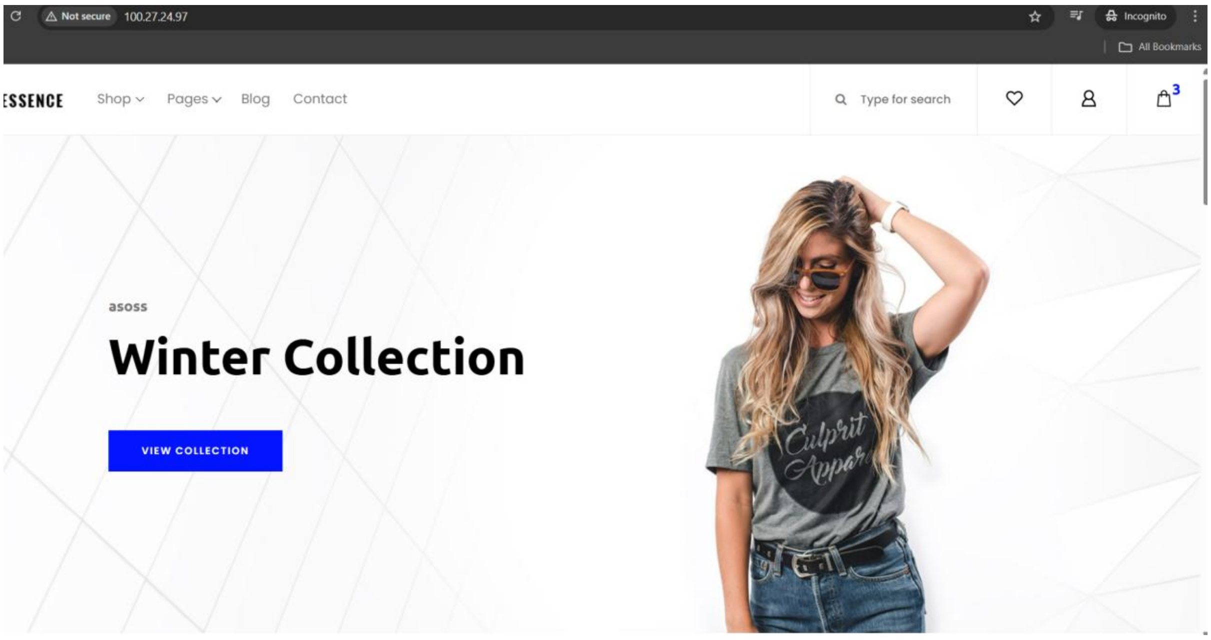
```
ubuntu@Master-Node:~/ansible$ sudo vi /etc/ansible/hosts
ubuntu@Master-Node:~/ansible$ cat /etc/ansible/hosts
[food_app]

node1 ansible_host=44.203.157.165 ansible_user=ubuntu

[ecom_app]
node2 ansible_host=100.27.24.97 ansible_user=ubuntu

ubuntu@Master-Node:~/ansible$
```

- Let’s try to execute the playbook using **ansible-playbook food-app.yml**

- After successfull execution try to access the application we can see that different apps in each server
- Here we can see thar in the server 1 food application is running.!



- In the server-2 ecom application is running.!



www.linkedin.com/in/arun-kumar-akula