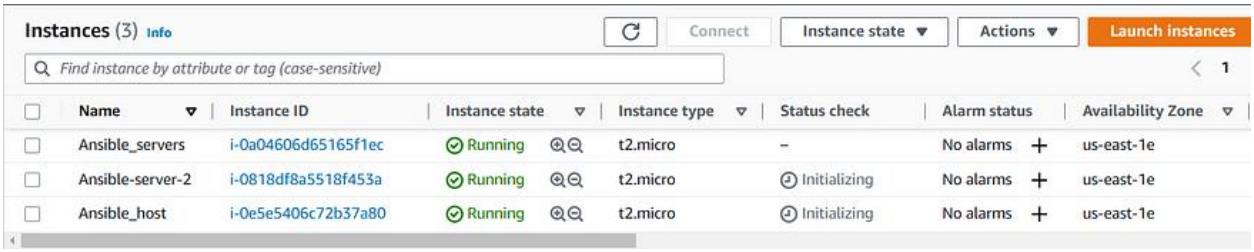
Things to be covered in Hands-On.

1. Creating Host server.
2. Install Ansible to Host server.
3. Creating 2 more EC2 instances (servers)
4. Add these EC2s to inventory file.
5. Configure the servers using command method.
6. Uses of playbooks.
7. Deployment of a simple webpage using Ansible.

So, fasten your seat belt and just login to your AWS account. and follow me for next 10–15 minutes.

Step_1

First of all we will create 3 EC2 instances . make sure all three are created with same key pair. our one server would be host server where we perform all the tasks and rest are other servers.



SERVERS

STEP2

— Now we will install Ansible in host server (Ansible_host). you can follow the followings commands to install Ansible.

sudo apt-add-repository ppa:ansible/ansible

sudo apt update

sudo apt install ansible

STEP3

— copy the private key from local to Host server (Ansible_host) at (/home/ubuntu/.ssh). you can follow the following command to copy the key from local to Ansible_host.

scp -i "<<key pair name>>" <<key pair name>> <<public DNS of EC2>>:/home/ubuntu/.ssh

```
C:\Users\Admin\Downloads>scp -i "ansible-key-pair.pem" ansible-key-pair.pem ubuntu@ec2-100-25-167-3.compute-1.amazonaws.
com:/home/ubuntu/.ssh
The authenticity of host 'ec2-100-25-167-3.compute-1.amazonaws.com (100.25.167.3)' can't be established.
ECDSA key fingerprint is SHA256:vyDf2bF5DQSxRA9uQbroGZaz4aqZWXRutC6j255YtoU.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
Warning: Permanently added 'ec2-100-25-167-3.compute-1.amazonaws.com,100.25.167.3' (ECDSA) to the list of known hosts.
ansible-key-pair.pem                                                          100% 1678     4.5KB/s   00:00
```

adding key pair to ec2

STEP4

— Now we will access the inventory file using *sudo vim /etc/ansible/hosts*

Ansible ad hoc commands- using ad hoc commands is a quick way to run a single task on one or more managed nodes.

some examples of valid use cases are rebooting servers, copying files, checking connection status, managing packages, gathering facts etc.

The pattern for ad hoc commands looks like this :

$ansible [host-pattern] -m [module] -a "[module options]"

now we will define some servers in our inventory file and ping them using *ansible all -m ping (you can refer the Screenshot)*

```
# If you have multiple hosts following a pattern, you can specify
# them like this:

## www[001:006].example.com

# Ex 3: A collection of database servers in the 'dbservers' group:

## [dbservers]
##
## db01.intranet.mydomain.net
## db02.intranet.mydomain.net
## 10.25.1.56
## 10.25.1.57

# Here's another example of host ranges, this time there are no
# leading 0s:

## db-[99:101]-node.example.com

[servers]

server1 ansible_host=100.26.164.161
server2 ansible_host=54.237.210.164

[servers:vars]
ansible_python_interpreter=/usr/bin/python3
ansible_ssh_private_key_file=/home/ubuntu/.ssh/ansible-key-pair.pem
"/etc/ansible/hosts" 53L, 1230B
```

Defining the servers in inventory file

Now we will ping the servers and do some server configurations (refer the screenshot for better understanding)

```
aws  Services  Search  [Alt+S]
ubuntu@ip-172-31-60-17:~$ ansible all -m ping
server2 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
server1 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
ubuntu@ip-172-31-60-17:~$ ansible all -a "df -h" -u ubuntu
server1 | CHANGED | rc=0 >>
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       7.6G  1.6G  6.0G  21% /
tmpfs           484M     0  484M   0% /dev/shm
tmpfs           194M  836K  193M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
/dev/xvda15     105M  6.1M   99M   6% /boot/efi
tmpfs            97M  4.0K   97M   1% /run/user/1000
server2 | CHANGED | rc=0 >>
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       7.6G  1.6G  6.0G  21% /
tmpfs           484M     0  484M   0% /dev/shm
tmpfs           194M  836K  193M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
/dev/xvda15     105M  6.1M   99M   6% /boot/efi
tmpfs            97M  4.0K   97M   1% /run/user/1000
ubuntu@ip-172-31-60-17:~$ ansible servers -a "sudo apt update"
server1 | CHANGED | rc=0 >>
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
```

SERVER CONFIGURATION

Booom we have successfully ping the 2 hosts that we have defined in our host file and checked the disk space of both server at the same time just by a single command , that's the power of Ansible. you can also perform some other tasks if you want.

STEP5

— in this step we will learn about role of playbooks in ansible.

Playbooks are the simplest way in Ansible to automate repeating tasks in the form of resuable and consistent configuration files. Playbooks are defined in YAML files and any orderd set of steps to be executed in our managed nodes.

As mentioned , task in a playbook are executed from top to bottom. At a minimum, a playbook should define the managed nodes to target and some tasks to run against them.

here we will see some example of playbooks and execute them and see how can we configure the server.

```yaml
-
 name: This playbook will install nginx
 hosts: servers
 become: yes
 tasks:
   - name: install nginx
     apt:
       name: nginx
       state: latest
   - name: start nginx
     service:
       name: nginx
       state: started
       enabled: yes
```

A sample playbook to install nginx

Now we will run this playbook by using (ansible-playbook <<name of play book>>) .

```
ubuntu@ip-172-31-60-17:~/playbooks$ sudo vim nginx.yml
ubuntu@ip-172-31-60-17:~/playbooks$ ansible-playbook nginx.yml

PLAY [This playbook will install nginx] ***********************************************************************************************************************

TASK [Gathering Facts] ****************************************************************************************************************************************
ok: [server2]
ok: [server1]

TASK [install nginx] ******************************************************************************************************************************************
changed: [server2]
changed: [server1]

TASK [start nginx] ********************************************************************************************************************************************
ok: [server2]
ok: [server1]

PLAY RECAP ****************************************************************************************************************************************************
server1                    : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
server2                    : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

so, as we can see in above screenshot that nginx is installed in both the server. so, it's time to ssh one of the server and check whether nginx is installed or not.

```
ubuntu@ip-172-31-52-16:~$ service nginx status
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2023-02-12 04:23:28 UTC; 1min 12s ago
       Docs: man:nginx(8)
    Process: 2813 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
    Process: 2814 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Main PID: 2910 (nginx)
      Tasks: 2 (limit: 1143)
     Memory: 4.4M
        CPU: 23ms
     CGroup: /system.slice/nginx.service
             ├─2910 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─2913 "nginx: worker process" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""

Feb 12 04:23:28 ip-172-31-52-16 systemd[1]: Starting A high performance web server and a reverse proxy server...
lines 1-15
```

nginx is installed in the servers

boom, as we can see we have successfully executed the playbook. so again, we saw how powerful and effective this tool is.

STEP6

— Now we will deploy a sample webpage using the ansible playbook. this sounds interesting, is it?

so first we will create an index.html file in which source code is present. at the end of the blog i will provide Github repository link where all the codes and playbooks are present.

```html
<!DOCTYPE html>
<html>
  <head>
    <style>
      body {
        font-family: Arial, sans-serif;
        background-color: #f2f2f2;
        color: #333;
        text-align: center;
      }

      h1 {
        font-size: 36px;
        margin-top: 50px;
        color: #6130e8;
      }

      p {
        font-size: 18px;
        margin: 20px 0;
      }
    </style>
  </head>
  <body>
    <h1>Thank you people for reading my blog</h1>
    <p>kindly leave a feedback</p>
  </body>
</html>
```

index.html

now we will create a playbook to perform this particular task.

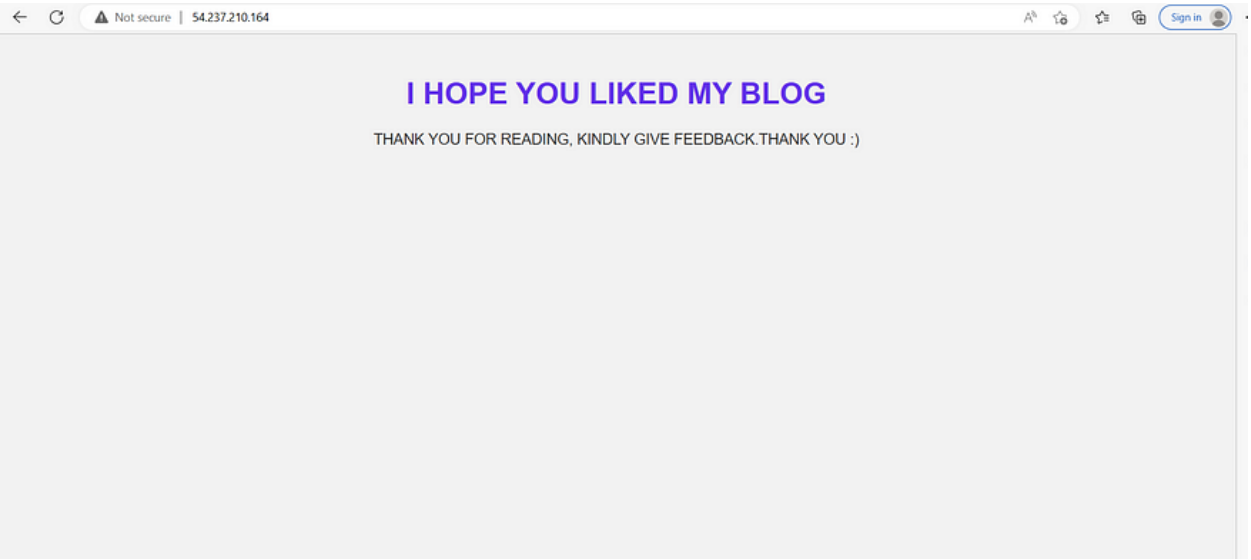
```yaml
-
 name: this is a  simple html project
 hosts: servers
 become: yes
 tasks:
   - name: Install nginx
     apt:
       name: nginx
       state: latest

   - name: Start nginx
     service:
       name: nginx
       state: started

   - name: Deploy webpage
     copy:
       src: index.html
       dest: /var/www/html
```

Playbook to deploy a webpage

So, now we will run this playbook and see that webpage has deployed to all the dedicated servers.



Deployment of a sample webpage using Ansible playbook

So, we successfully deployed a sample webpage to all the dedicated servers. and we can access this web application using theirs Public IP address.