Ansible Project

AWS Infrastructure Setup using Ansible.

Overview

This project automates the creation and configuration of AWS EC2 instances using Ansible. The setup includes three EC2 instances (two Ubuntu and one Amazon Linux) and establishes password less SSH authentication. Additionally, NGINX is installed and configured on the instances.

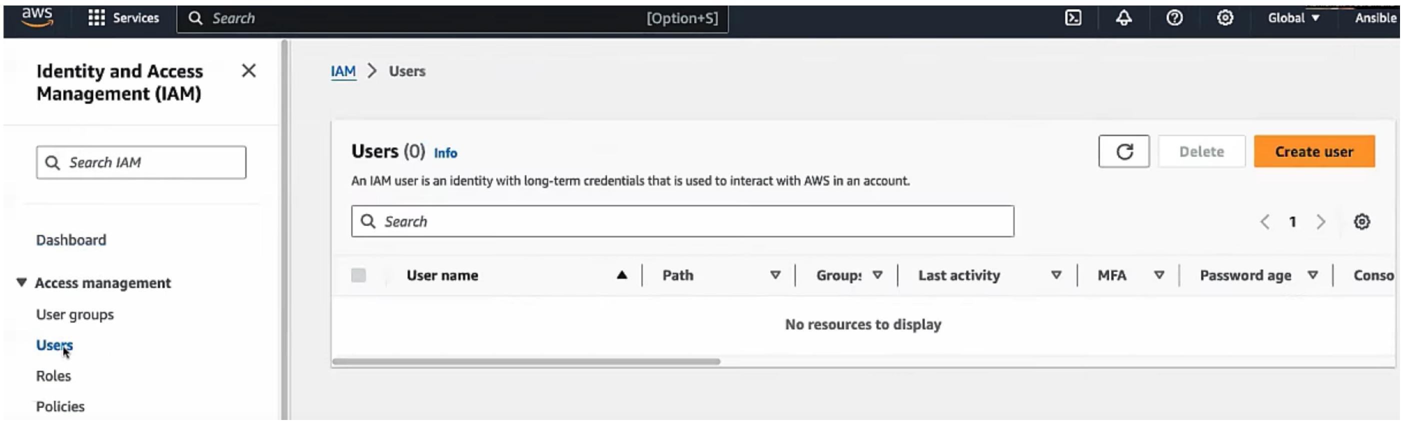
Prerequisites

1. Ansible installed on your local machine/control node.
2. Install boto3 and the Ansible Galaxy collection for Amazon AWS:

   pip install boto3

   ansible-galaxy collection install amazon.aws

3. Create an AWS IAM user with full EC2 access permissions.
4. Generate access key and secret access key for the IAM user.

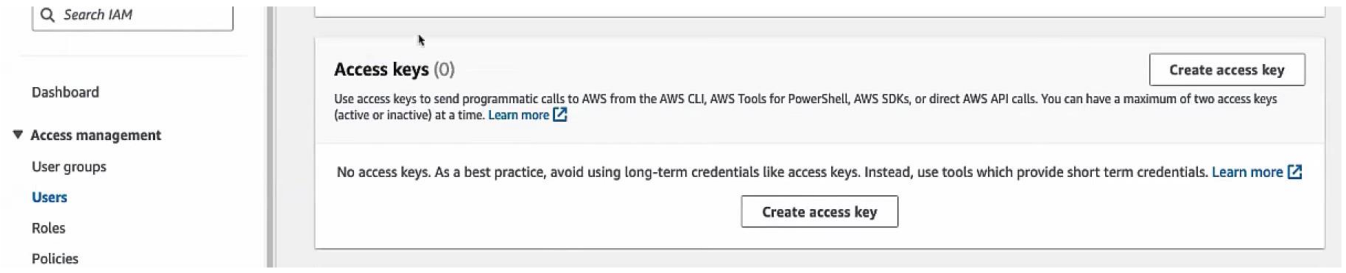


Users (0) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

User name	Path	Groups	Last activity	MFA	Password age	Conso

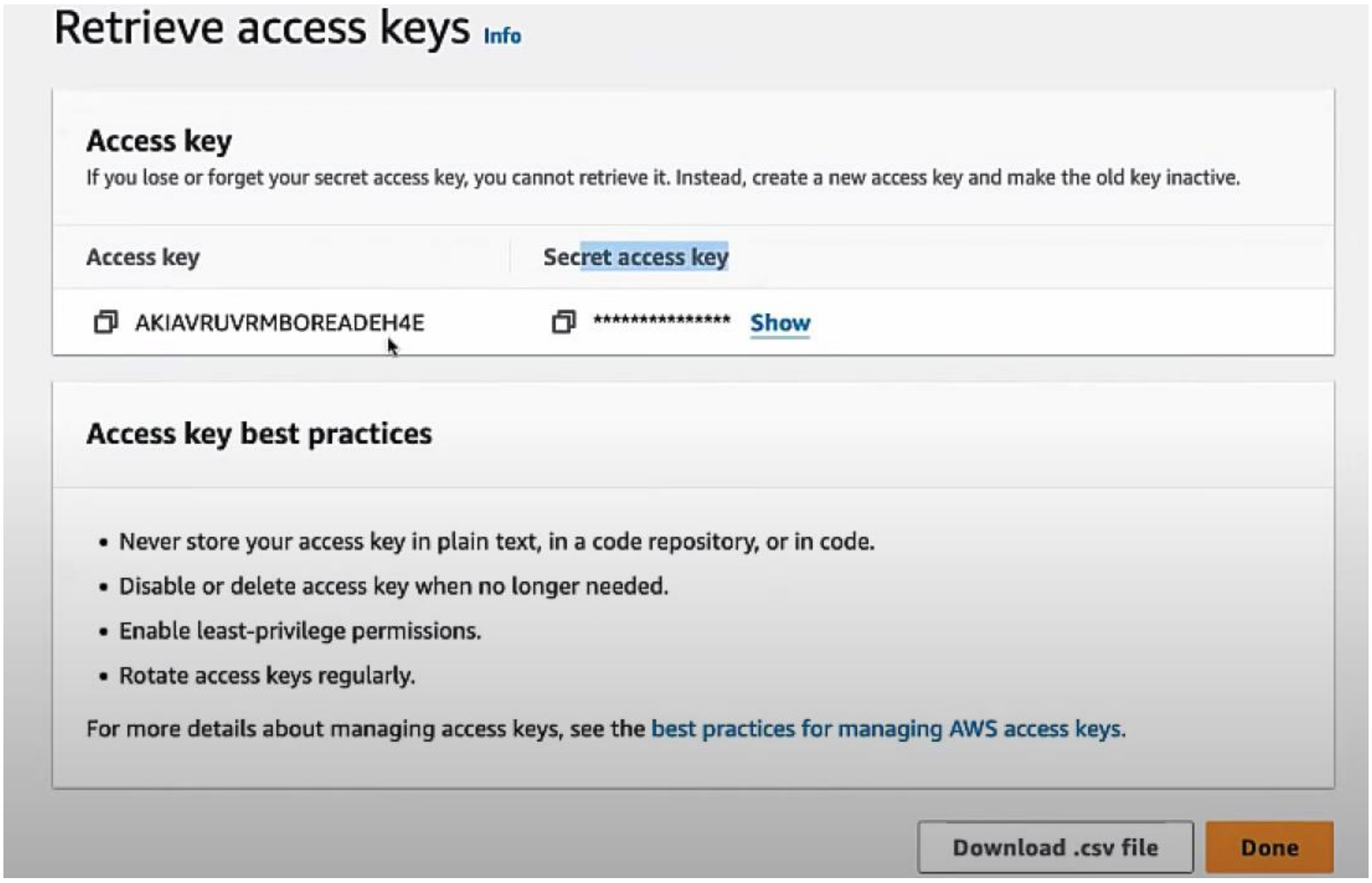
No resources to display



Access keys (0)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. Learn more

No access keys. As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. Learn more


Retrieve access keys Info

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIAVRUVRMBOREADEH4E	*************** Show

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the best practices for managing AWS access keys.

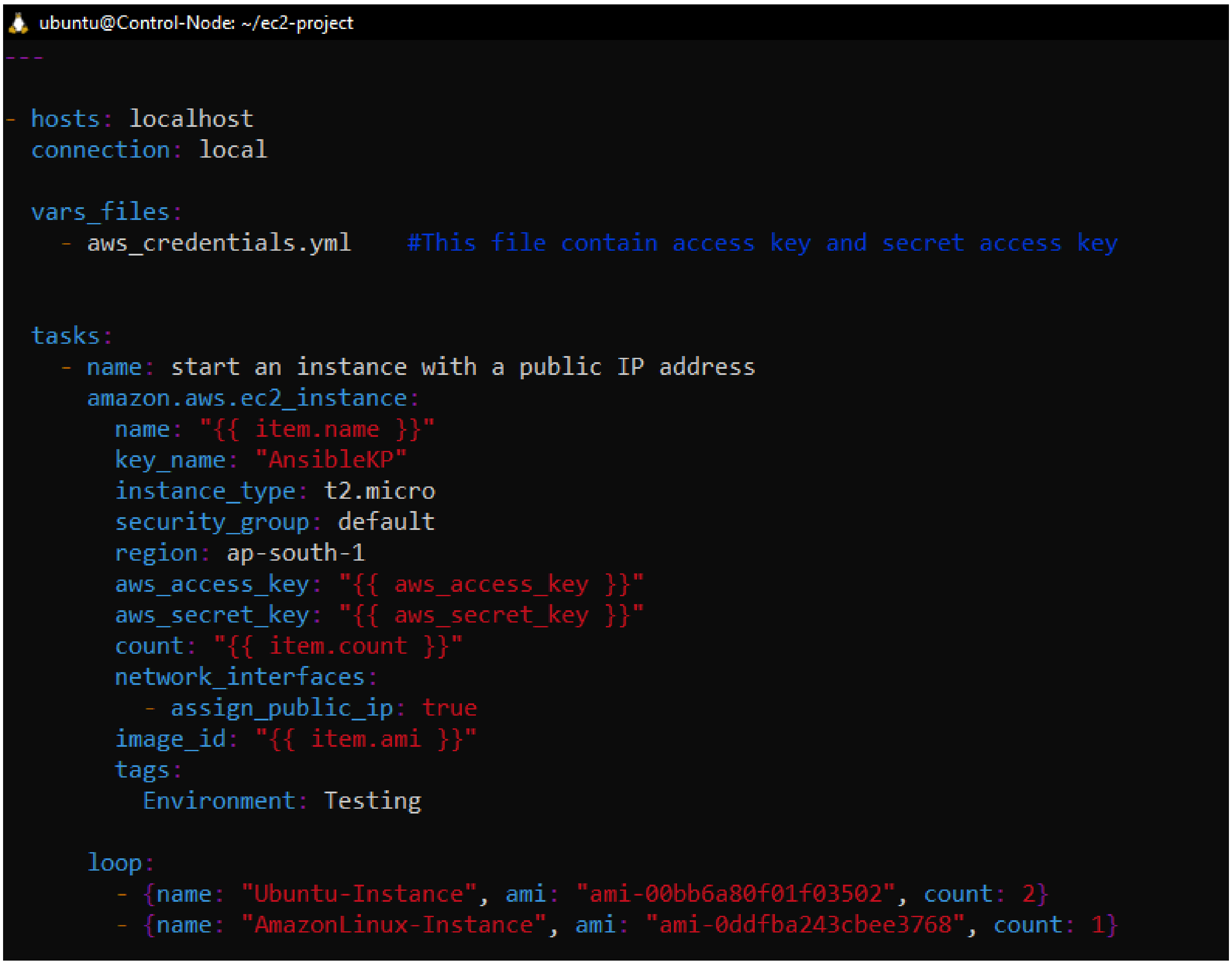
5. Store the access key and secret access key in Ansible Vault:

```
openssl rand -base64 32 > vault.pass

ansible-vault create ./aws_credentials.yml --vault-password-file vault.pass
```

6. In the aws_credentials.yml file, add the access key and secret access key:

```
aws_access_key: {your_access_key}
aws_secret_key: {your_secret_key}
```

7. This playbook creates the specified EC2 instances using the Ansible Galaxy collection for Amazon AWS.



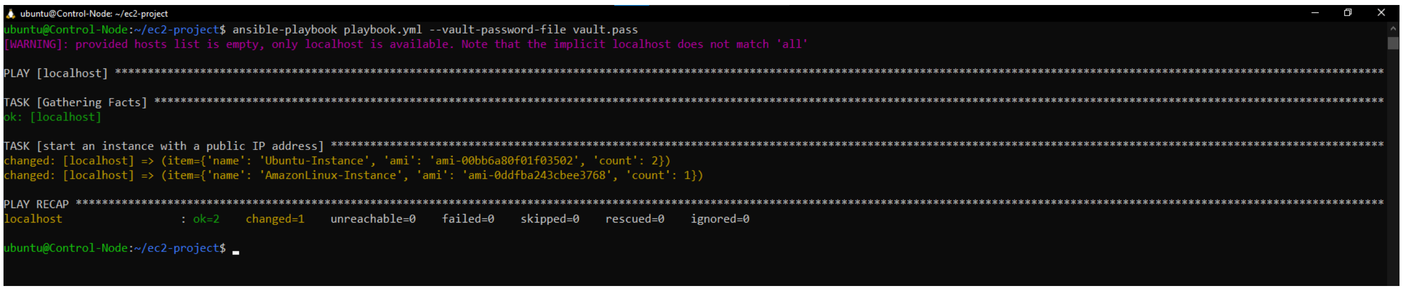
```
ubuntu@Control-Node: ~/ec2-project
---

- hosts: localhost
  connection: local

  vars_files:
    - aws_credentials.yml    #This file contain access key and secret access key


  tasks:
    - name: start an instance with a public IP address
      amazon.aws.ec2_instance:
        name: "{{ item.name }}"
        key_name: "AnsibleKP"
        instance_type: t2.micro
        security_group: default
        region: ap-south-1
        aws_access_key: "{{ aws_access_key }}"
        aws_secret_key: "{{ aws_secret_key }}"
        count: "{{ item.count }}"
        network_interfaces:
          - assign_public_ip: true
        image_id: "{{ item.ami }}"
        tags:
          Environment: Testing

      loop:
        - {name: "Ubuntu-Instance", ami: "ami-00bb6a80f01f03502", count: 2}
        - {name: "AmazonLinux-Instance", ami: "ami-0ddfba243cbee3768", count: 1}
```

8. Run the playbook to create EC2 instances:

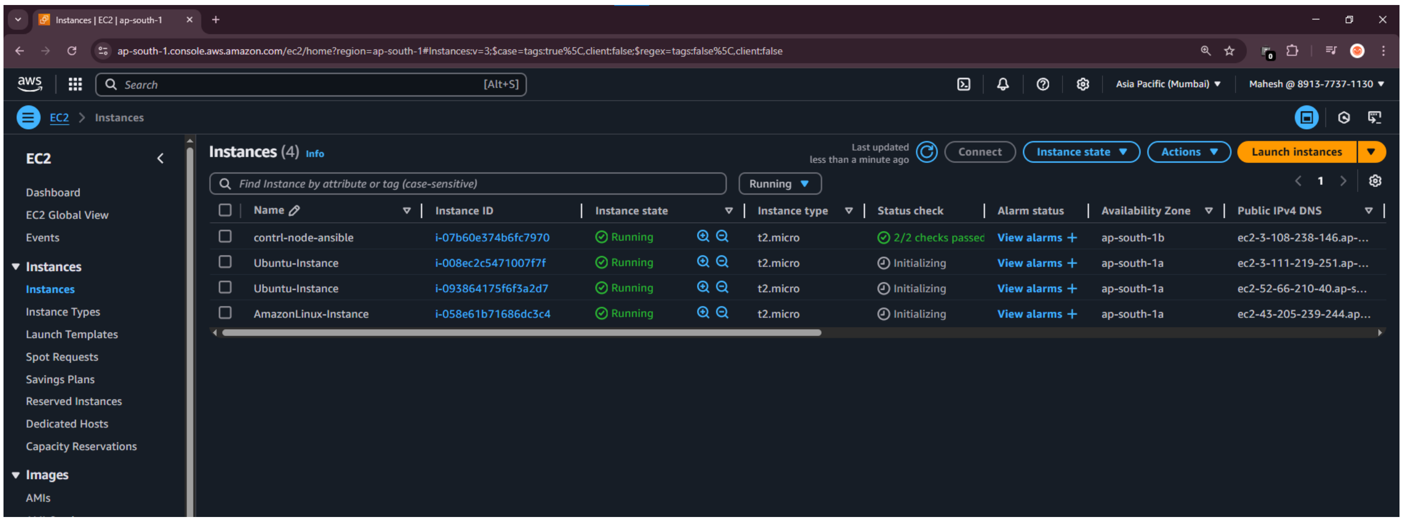
   ansible-playbook playbook.yml --vault-password-file vault.pass

   After running this command, Ansible will create three EC2 instances:

- Two Ubuntu instances
- One Amazon Linux instance



```
ubuntu@Control-Node: ~/ec2-project
ubuntu@Control-Node:~/ec2-project$ ansible-playbook playbook.yml --vault-password-file vault.pass
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [localhost] ***********************************************************************************************************************************************************

TASK [Gathering Facts] *****************************************************************************************************************************************************
ok: [localhost]

TASK [start an instance with a public IP address] **************************************************************************************************************************
changed: [localhost] => (item={'name': 'Ubuntu-Instance', 'ami': 'ami-00bb6a80f01f03502', 'count': 2})
changed: [localhost] => (item={'name': 'AmazonLinux-Instance', 'ami': 'ami-0ddfba243cbee3768', 'count': 1})

PLAY RECAP *****************************************************************************************************************************************************************
localhost                  : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

ubuntu@Control-Node:~/ec2-project$ _
```



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
contrl-node-ansible	i-07b60e374b6fc7970	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1b	ec2-3-108-238-146.ap-...
Ubuntu-Instance	i-008ec2c5471007f7f	Running	t2.micro	Initializing	View alarms +	ap-south-1a	ec2-3-111-219-251.ap-...
Ubuntu-Instance	i-093864175f6f3a2d7	Running	t2.micro	Initializing	View alarms +	ap-south-1a	ec2-52-66-210-40.ap-s...
AmazonLinux-Instance	i-058e61b71686dc3c4	Running	t2.micro	Initializing	View alarms +	ap-south-1a	ec2-43-205-239-244.ap...

Task-2 Password less SSH Authentication

To set up password less SSH authentication, follow these steps:

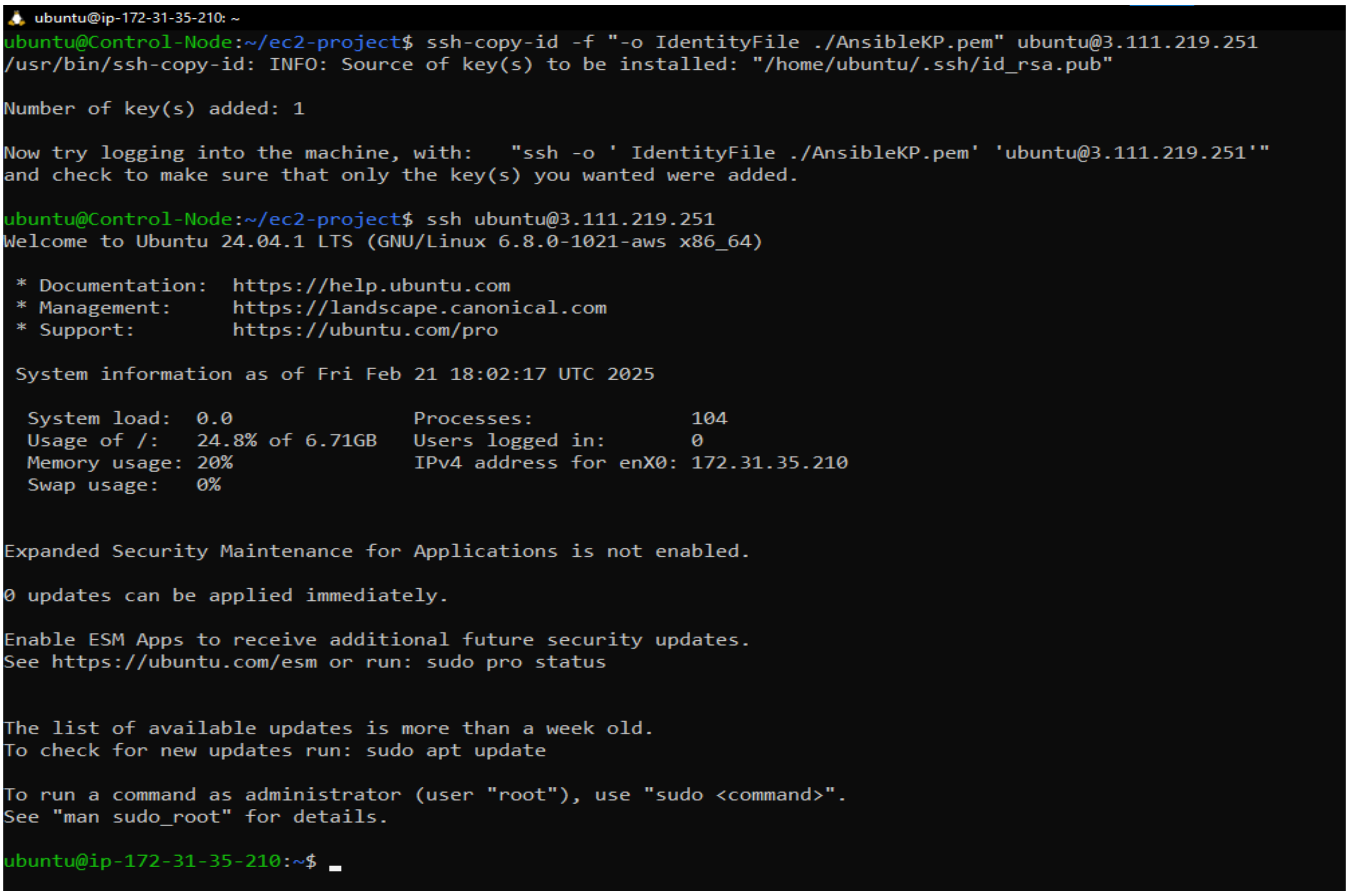
1. Ensure you have the AnsibleKP.pem file on your control node.
2. Use the following command to set up password-less SSH authentication for Ubuntu instances:

ssh-copy-id -f "-o IdentityFile <path_to_pem_file>" ubuntu@<instance_public_ip>

3. Repeat the same process for the Amazon Linux instance, but use the ec2-user:

ssh-copy-id -f "-o IdentityFile <path_to_pem_file>" ec2-user@<instance ip>

check if you are able to connect to all the instance: -

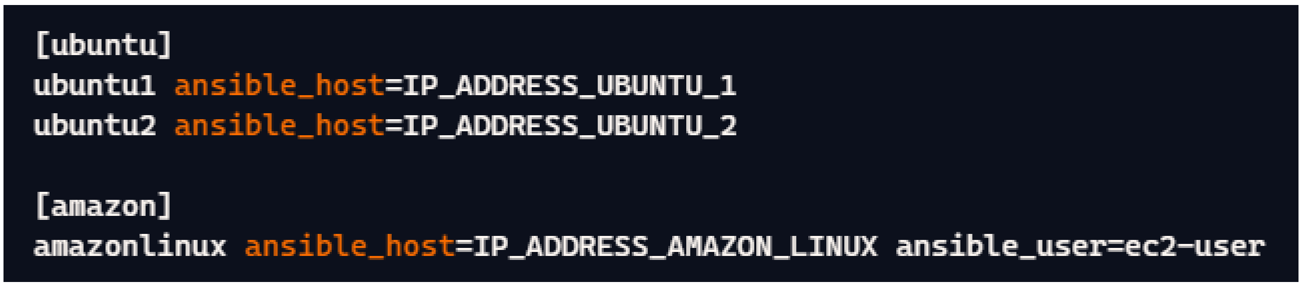


```
ubuntu@ip-172-31-35-210: ~
ubuntu@Control-Node:~/ec2-project$ ssh-copy-id -f "-o IdentityFile ./AnsibleKP.pem" ubuntu@3.111.219.251
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ubuntu/.ssh/id_rsa.pub"

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh -o ' IdentityFile ./AnsibleKP.pem' 'ubuntu@3.111.219.251'"
and check to make sure that only the key(s) you wanted were added.

ubuntu@Control-Node:~/ec2-project$ ssh ubuntu@3.111.219.251
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1021-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Fri Feb 21 18:02:17 UTC 2025

  System load:  0.0               Processes:             104
  Usage of /:   24.8% of 6.71GB   Users logged in:       0
  Memory usage: 20%               IPv4 address for enX0: 172.31.35.210
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-35-210:~$ _
```

Task-3 Installing and Configuring NGINX

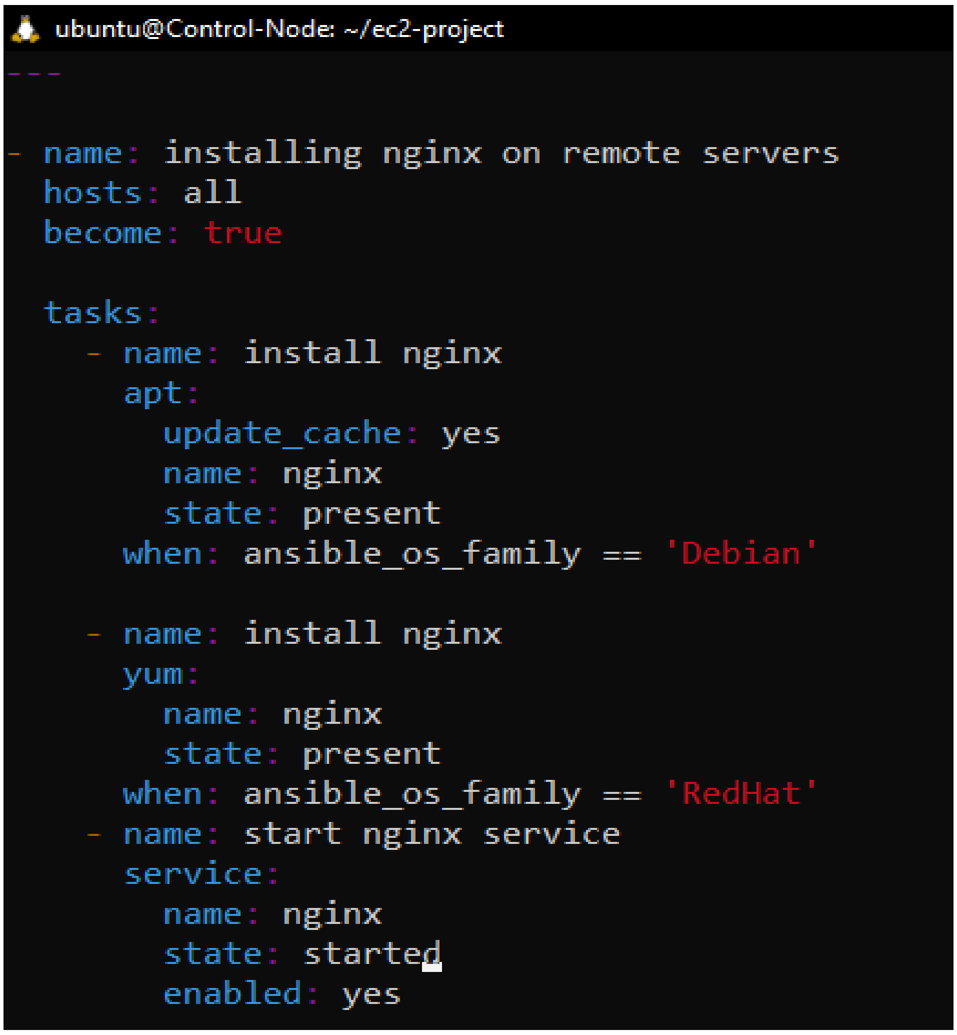
1. **Ansible Inventory File (inventory.ini)**

First, create an Ansible inventory file that contains the public IP addresses of all the instances:



```
[ubuntu]
ubuntu1 ansible_host=IP_ADDRESS_UBUNTU_1
ubuntu2 ansible_host=IP_ADDRESS_UBUNTU_2

[amazon]
amazonlinux ansible_host=IP_ADDRESS_AMAZON_LINUX ansible_user=ec2-user
```

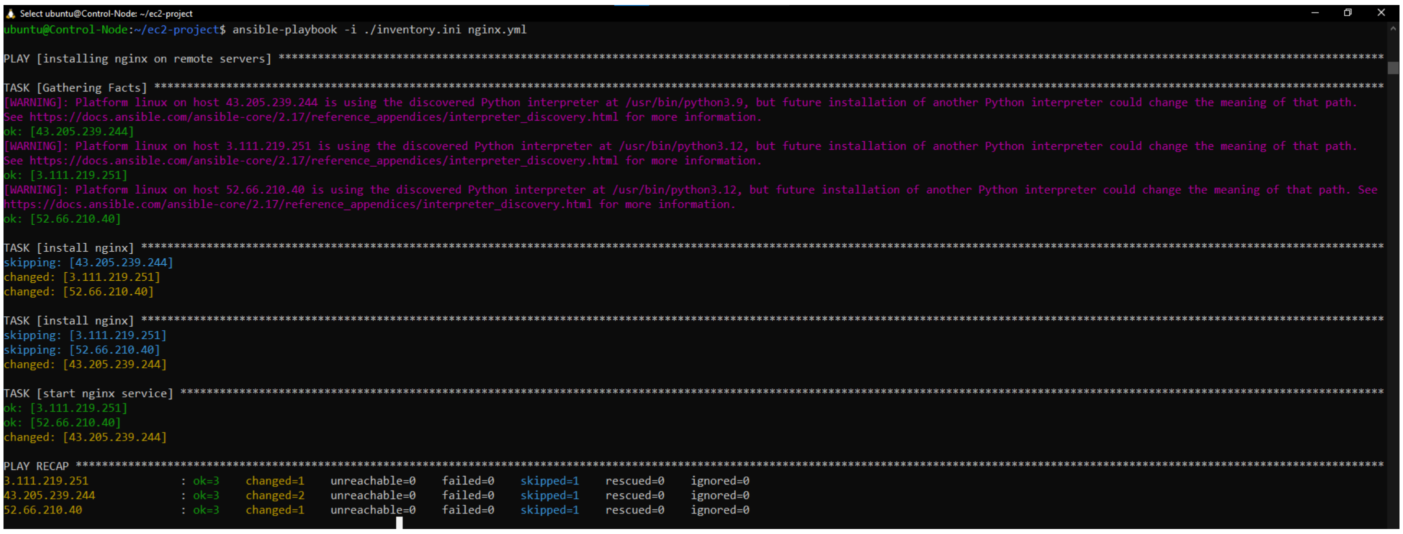
2. playbook: Installing and Configuring NGINX (nginx.yml)



```
ubuntu@Control-Node: ~/ec2-project
---

- name: installing nginx on remote servers
  hosts: all
  become: true

  tasks:
    - name: install nginx
      apt:
        update_cache: yes
        name: nginx
        state: present
      when: ansible_os_family == 'Debian'

    - name: install nginx
      yum:
        name: nginx
        state: present
      when: ansible_os_family == 'RedHat'
    - name: start nginx service
      service:
        name: nginx
        state: started
        enabled: yes
```

Run the nginx.yml playbook to install and configure NGINX on ***all instances:***

ansible-playbook -i ./inventory.ini nginx.yml



```
ubuntu@Control-Node:~/ec2-project$ ansible-playbook -i ./inventory.ini nginx.yml

PLAY [installing nginx on remote servers] **********************************************************************************************

TASK [Gathering Facts] *****************************************************************************************************************
[WARNING]: Platform linux on host 43.205.239.244 is using the discovered Python interpreter at /usr/bin/python3.9, but future installation of another Python interpreter could change the meaning of that path.
See https://docs.ansible.com/ansible-core/2.17/reference_appendices/interpreter_discovery.html for more information.
ok: [43.205.239.244]
[WARNING]: Platform linux on host 3.111.219.251 is using the discovered Python interpreter at /usr/bin/python3.12, but future installation of another Python interpreter could change the meaning of that path.
See https://docs.ansible.com/ansible-core/2.17/reference_appendices/interpreter_discovery.html for more information.
ok: [3.111.219.251]
[WARNING]: Platform linux on host 52.66.210.40 is using the discovered Python interpreter at /usr/bin/python3.12, but future installation of another Python interpreter could change the meaning of that path. See
https://docs.ansible.com/ansible-core/2.17/reference_appendices/interpreter_discovery.html for more information.
ok: [52.66.210.40]

TASK [install nginx] *******************************************************************************************************************
skipping: [43.205.239.244]
changed: [3.111.219.251]
changed: [52.66.210.40]

TASK [install nginx] *******************************************************************************************************************
skipping: [3.111.219.251]
skipping: [52.66.210.40]
changed: [43.205.239.244]

TASK [start nginx service] *************************************************************************************************************
ok: [3.111.219.251]
ok: [52.66.210.40]
changed: [43.205.239.244]

PLAY RECAP *****************************************************************************************************************************
3.111.219.251              : ok=3    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
43.205.239.244             : ok=3    changed=2    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
52.66.210.40               : ok=3    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

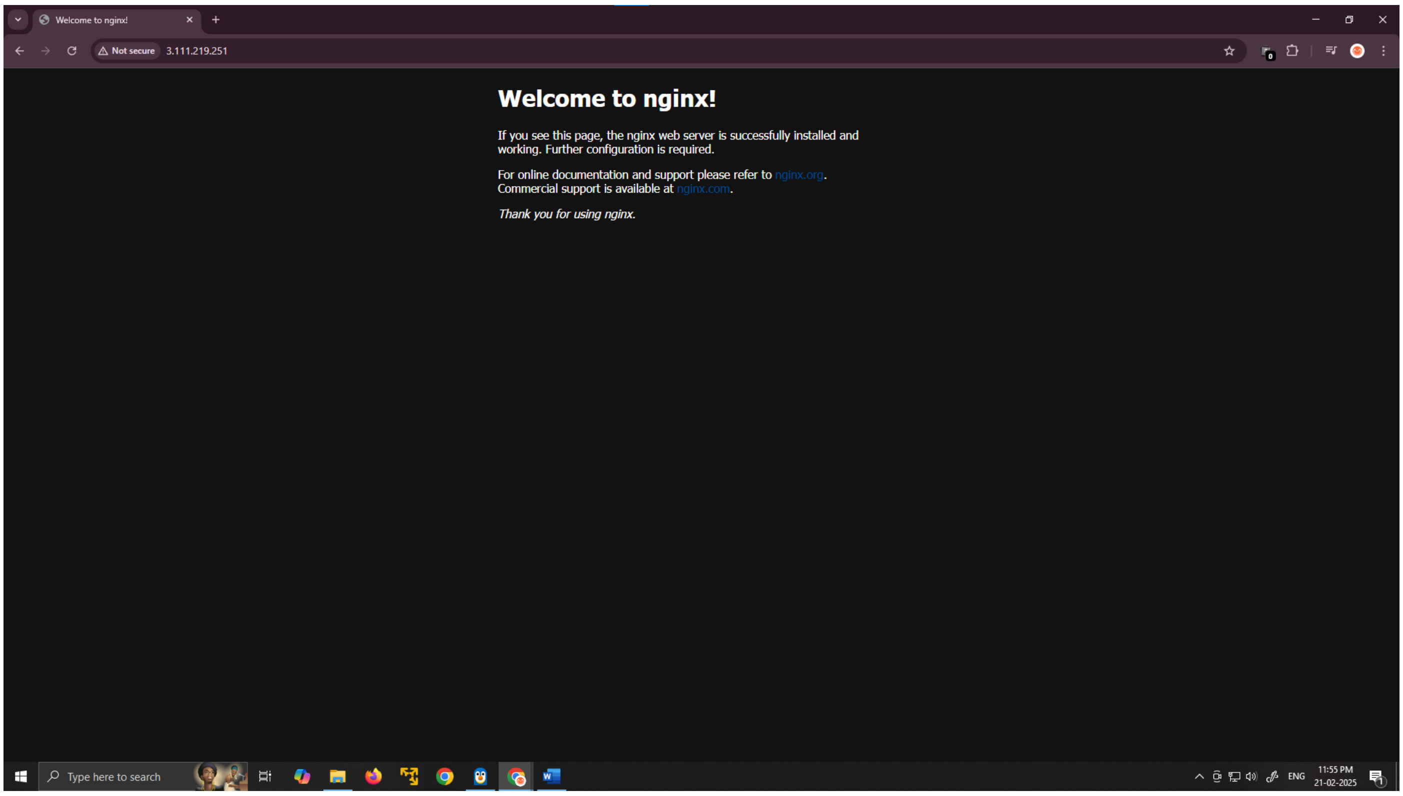
Verify Installation
Open your web browser and check if you can see the NGINX welcome page using the public IP address of each instance.
For example:

- Ubuntu Instance 1: http://IP_ADDRESS_UBUNTU_1
- Ubuntu Instance 2: http://IP_ADDRESS_UBUNTU_2
- Amazon Linux Instance: http://IP_ADDRESS_AMAZON_LINUX

You should see a page similar to the following:

Final Result:

Nginx configured on all the instances: -



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.