	Pimpri Chinchwad Education Trust's **Pimpri Chinchwad College of Engineering (PCCoE)** (An Autonomous Institute) Affiliated to Savitribai Phule Pune University(SPPU) ISO 21001:2018 Certified by TUV	

Course: DevOps Laboratory	**Code:** BIT26VS01
Name: Amar Vaijinath Chavan	**PRN:** 124B2F001
Assignment 15: Prepare a Case Study on Ansible and demonstrate its application.	

Aim: To prepare a comprehensive case study on Ansible automation and demonstrate its application by configuring a remote EC2 server using Ansible Playbooks.

Objectives:

- To understand Ansible's agentless architecture and its core components.
- To perform a real-world case study on the impact of Ansible in enterprise DevOps.
- To demonstrate the automation of software installation and configuration management on a remote cloud instance.

1. Theory & Case Study

1.1 Overview of Ansible

Ansible is an open-source IT automation engine that automates cloud provisioning, configuration management, application deployment, and intra-service orchestration. Unlike other tools, it is **agentless**, meaning it does not require any software to be installed on the managed nodes; it connects over standard **SSH**.

1.2 Key Components

- **Control Node:** The machine where Ansible is installed and from which commands are run.
- **Managed Nodes:** The target servers (like AWS EC2 instances) being managed by Ansible.
- **Inventory:** A file that defines the list of managed nodes and their logical groupings.
- **Playbooks:** YAML files that describe the desired state of your systems through a series of "tasks".
- **Modules:** Small programs that Ansible pushes to nodes to perform specific actions (e.g., apt, service).

1.3 Case Study: Amelco Financial Services

Background: Amelco, a provider of betting and financial market solutions, faced challenges managing over 400 VMware nodes across diverse Linux environments. Manual deployments were inconsistent, slow, and prone to downtime. **Implementation:** Amelco migrated to an agentless automation framework using Ansible and Ansible Tower.

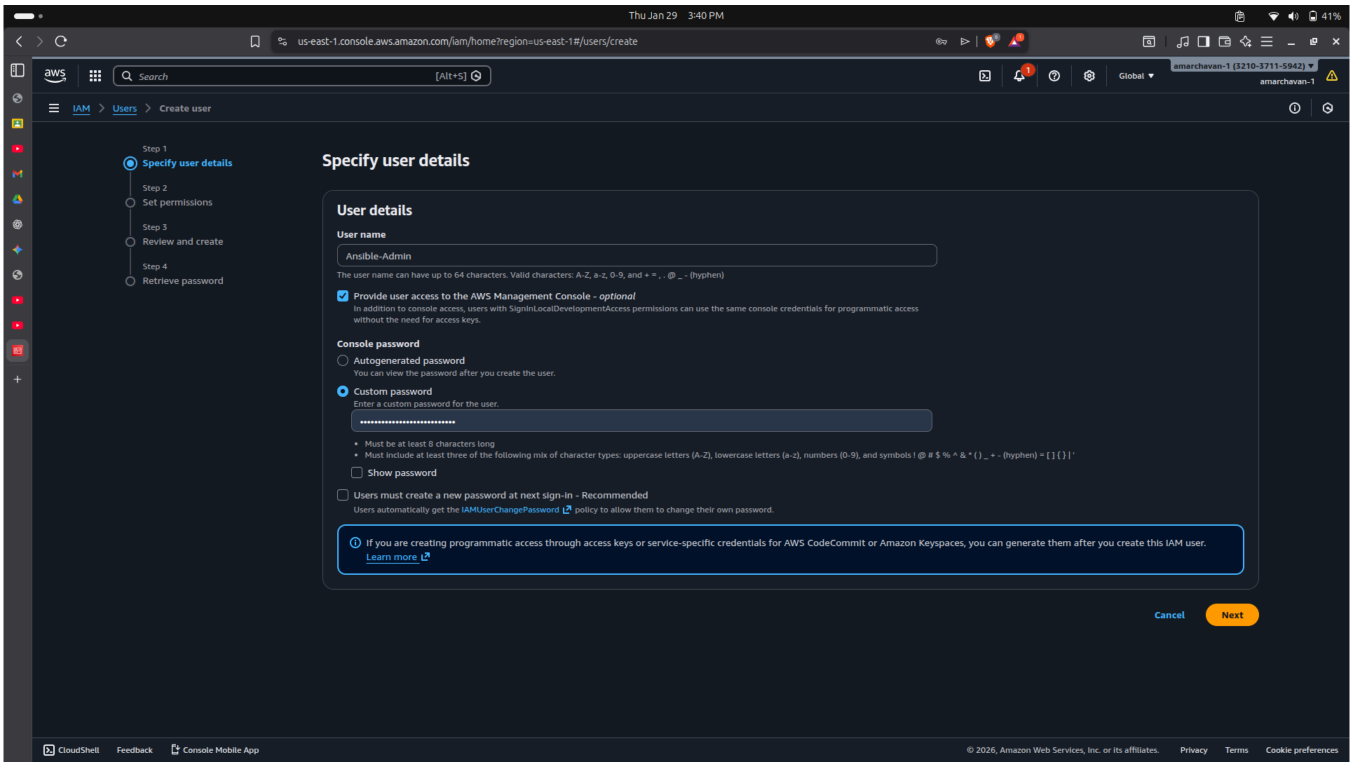
Results:

- **Efficiency:** Deployment times were reduced from weeks to days, and eventually to single-click releases.
- **Consistency:** Automated playbooks ensured a single source of truth across development, test, and production environments.
- **Self-Service IT:** Empowered non-technical teams to trigger automated workflows, reducing reliance on senior sysadmins.

2. Practical Procedure / Steps

Step 1: Environment Preparation (IAM & EC2)

- An IAM user **Ansible-Admin** was created with **AmazonEC2FullAccess** permissions to manage cloud resources programmatically.
- A new EC2 instance named **ansible-lab-server** was launched in the **eu-north-1** (Stockholm) region using an **Ubuntu 24.04 LTS** AMI and a **t3.micro** instance type.



Specify user details

User details

User name: Ansible-Admin

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

[✓] Provide user access to the AWS Management Console - *optional*
In addition to console access, users with SignInLocalDevelopmentAccess permissions can use the same console credentials for programmatic access without the need for access keys.

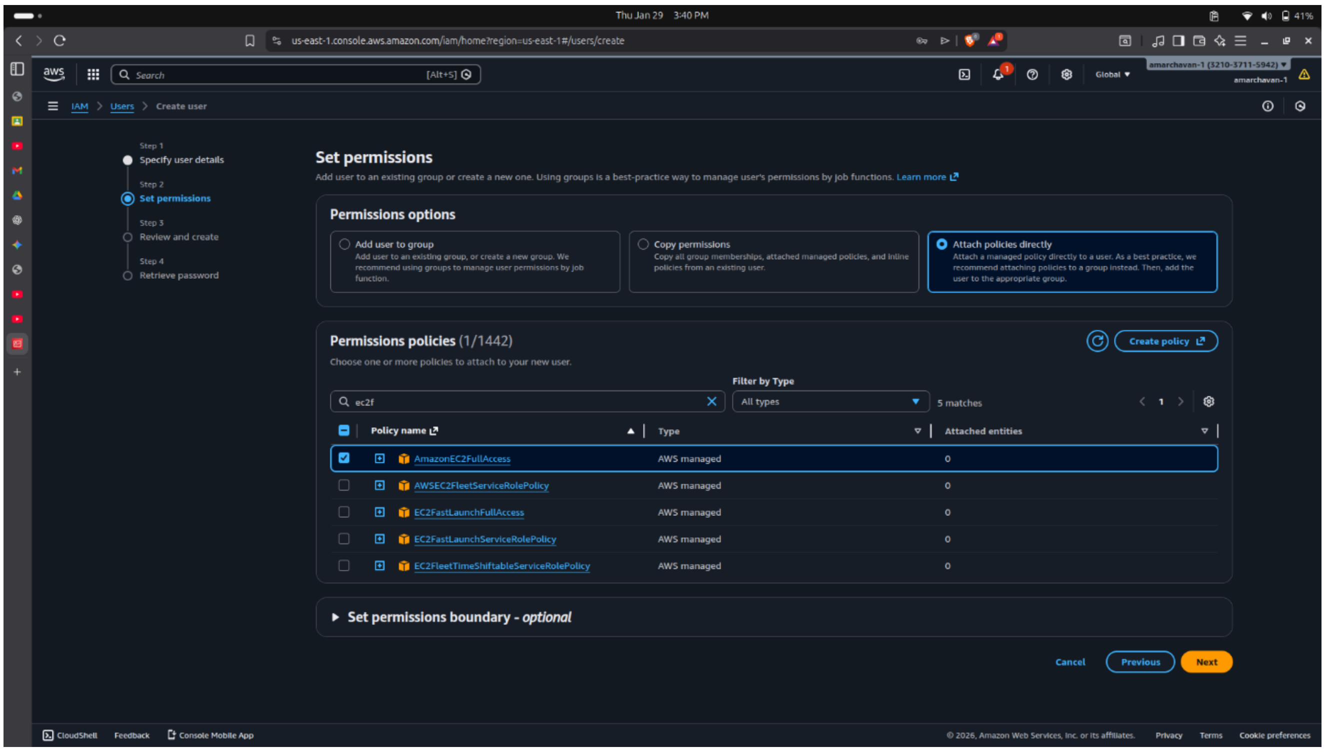
Console password
Autogenerated password [✓ Custom password]
You can view the password after you create the user.
Enter a custom password for the user.
••••••••••••••••••••••••••

- Must be at least 8 characters long
- Must include at least three of the following mix of character types: uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9), and symbols ! @ # $ % ^ & * ( ) _ + - (hyphen) = [ ] { } | '

[ ] Show password

[ ] Users must create a new password at next sign-in - Recommended
Users automatically get the IAMUserChangePassword policy to allow them to change their own password.

If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. Learn more



IAM > Users > Create user

Step 1
Specify user details
Step 2
Set permissions
Step 3
Review and create
Step 4
Retrieve password

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more

Permissions options

- Add user to group
  Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
- Copy permissions
  Copy all group memberships, attached managed policies, and inline policies from an existing user.
- [✓ Attach policies directly]
  Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1/1442)

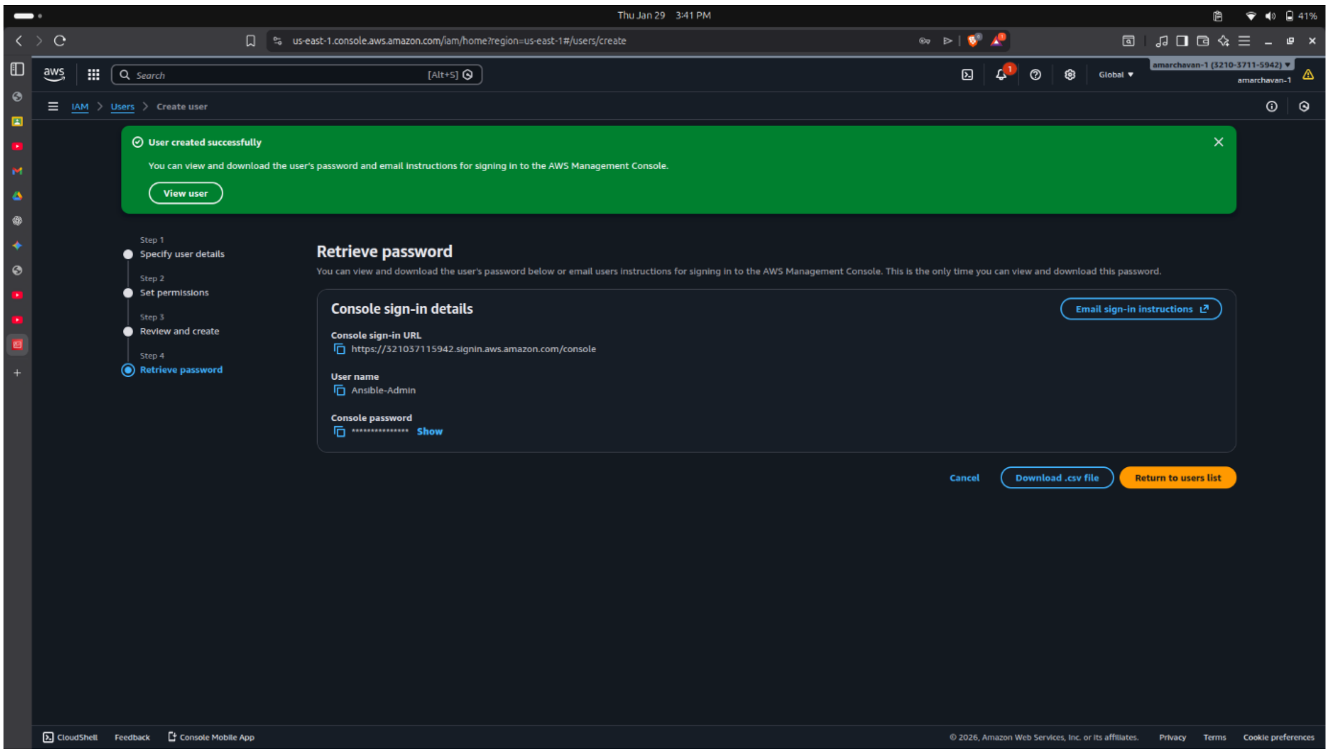
Choose one or more policies to attach to your new user.

Search: ec2f
Filter by Type: All types
5 matches

	Policy name	Type	Attached entities
✓	AmazonEC2FullAccess	AWS managed	0
	AWSEC2FleetServiceRolePolicy	AWS managed	0
	EC2FastLaunchFullAccess	AWS managed	0
	EC2FastLaunchServiceRolePolicy	AWS managed	0
	EC2FleetTimeShiftableServiceRolePolicy	AWS managed	0

▸ Set permissions boundary - *optional*

Cancel | Previous | Next



IAM > Users > Create user

User created successfully

You can view and download the user's password and email instructions for signing in to the AWS Management Console.

View user

Step 1
Specify user details
Step 2
Set permissions
Step 3
Review and create
Step 4
Retrieve password

Retrieve password

You can view and download the user's password below or email users instructions for signing in to the AWS Management Console. This is the only time you can view and download this password.

Console sign-in details

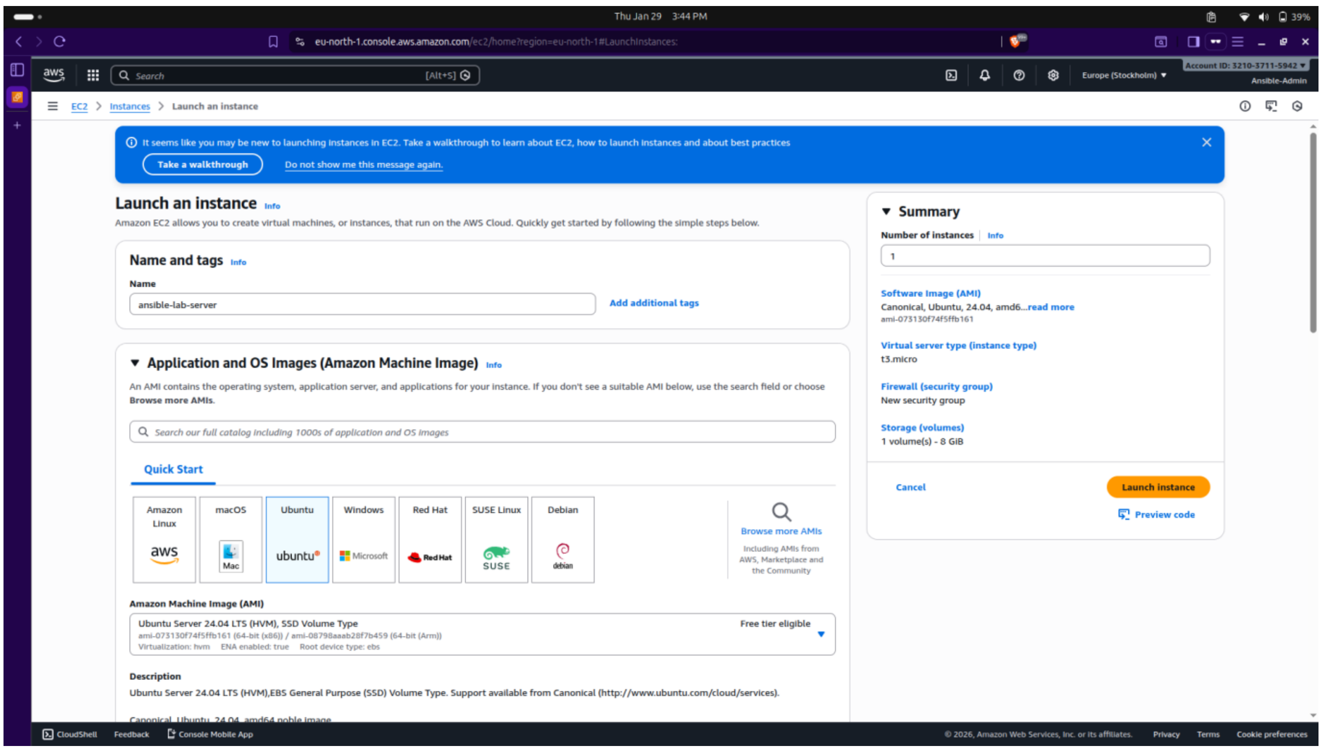
Console sign-in URL
https://321037115942.signin.aws.amazon.com/console

User name
Ansible-Admin

Console password
*************** Show

Email sign-in instructions

Cancel | Download .csv file | Return to users list



EC2 > Instances > Launch an instance

It seems like you may be new to launching instances in EC2. Take a walkthrough to learn about EC2, how to launch instances and about best practices

Launch an instance

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags

Name: ansible-lab-server

Application and OS Images (Amazon Machine Image)

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose **Browse more AMIs**.

Quick Start: Amazon Linux | macOS | [✓ Ubuntu] | Windows | Red Hat | SUSE Linux | Debian

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-073130f74f5ffb161 (64-bit (x86)) / ami-08798aaab28f7b459 (64-bit (Arm))
Virtualization: hvm   ENA enabled: true   Root device type: ebs

Description

Ubuntu Server 24.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

Summary

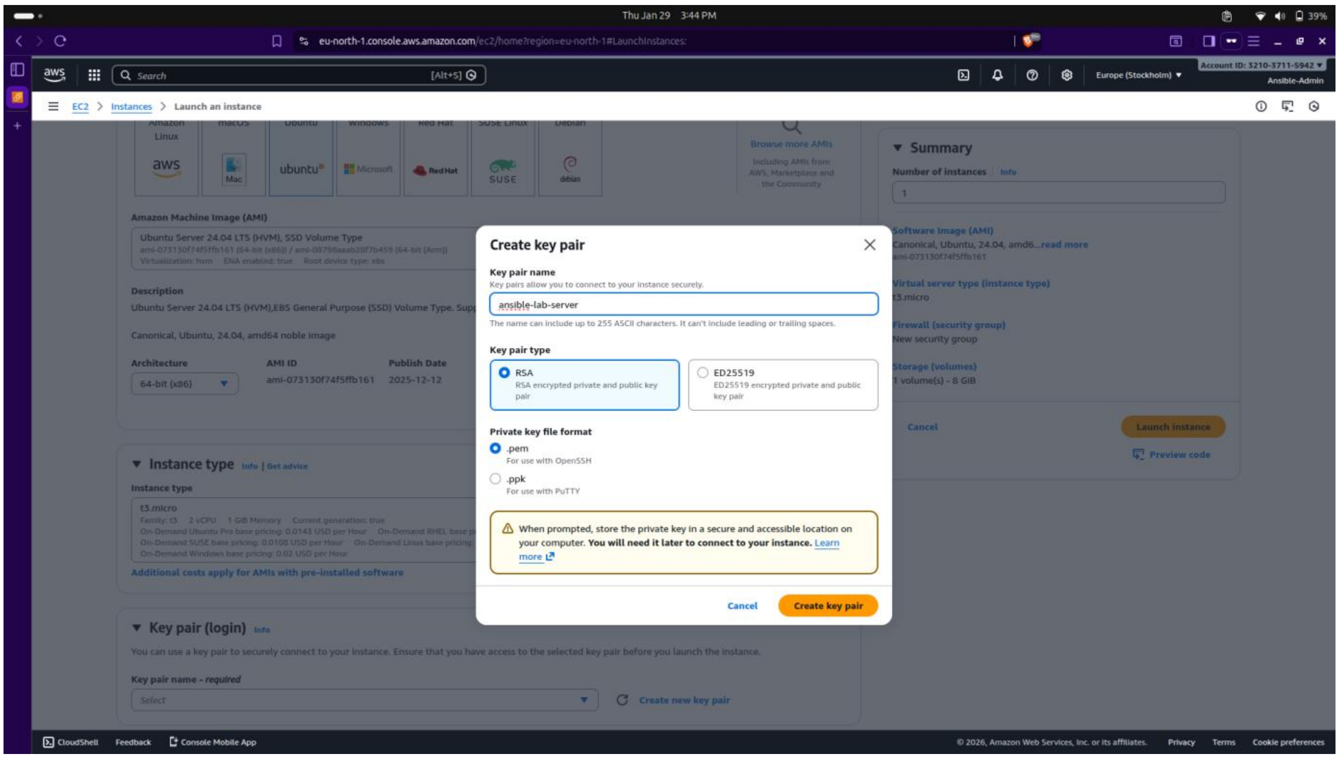
Number of instances: 1

Software Image (AMI): Canonical, Ubuntu, 24.04, amd6...read more
ami-073130f74f5ffb161

Virtual server type (instance type): t3.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB



Create key pair

Key pair name
Key pairs allow you to connect to your instance securely.

ansible-lab-server

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

[✓ RSA] RSA encrypted private and public key pair
ED25519 ED25519 encrypted private and public key pair

Private key file format

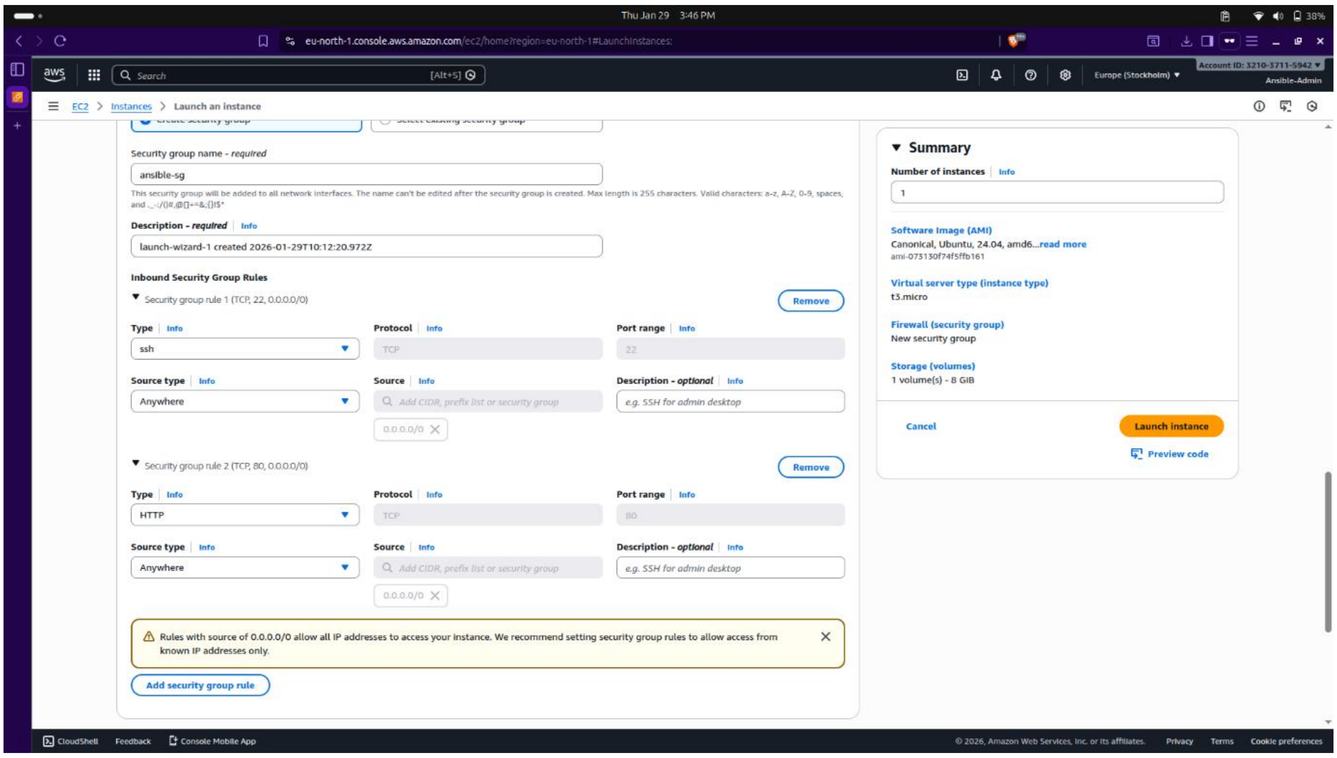
[✓ .pem] For use with OpenSSH
.ppk For use with PuTTY

When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** Learn more

Key pair (login)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*: Select



Thu Jan 29 3:46 PM

eu-north-1.console.aws.amazon.com/ec2/home?region=eu-north-1#LaunchInstances:

Account ID: 3210-3711-5942 ▼ Ansible-Admin — Europe (Stockholm) ▼

EC2 > Instances > Launch an instance

Security group name - *required*

ansible-sg

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&;{}!$*

Description - *required*

launch-wizard-1 created 2026-01-29T10:12:20.972Z

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0)

Type	Protocol	Port range
ssh	TCP	22

Source type	Source	Description - *optional*
Anywhere	0.0.0.0/0	e.g. SSH for admin desktop

▼ Security group rule 2 (TCP, 80, 0.0.0.0/0)

Type	Protocol	Port range
HTTP	TCP	80

Source type	Source	Description - *optional*
Anywhere	0.0.0.0/0	e.g. SSH for admin desktop

⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Add security group rule

Summary

Number of instances: 1

Software Image (AMI): Canonical, Ubuntu, 24.04, amd6...read more
ami-073130f74f5ffb161

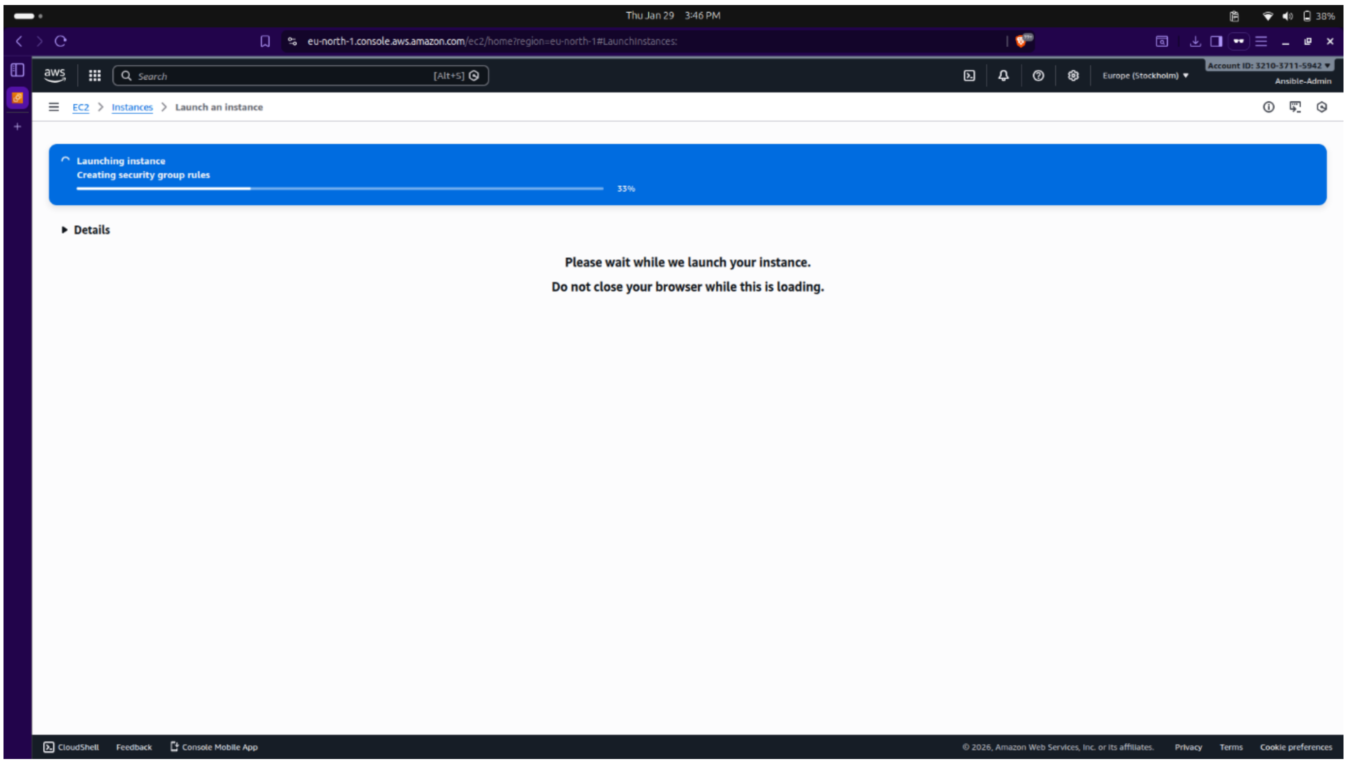
Virtual server type (instance type): t3.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Cancel — Launch instance — Preview code

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



Thu Jan 29 3:46 PM

eu-north-1.console.aws.amazon.com/ec2/home?region=eu-north-1#LaunchInstances:

EC2 > Instances > Launch an instance

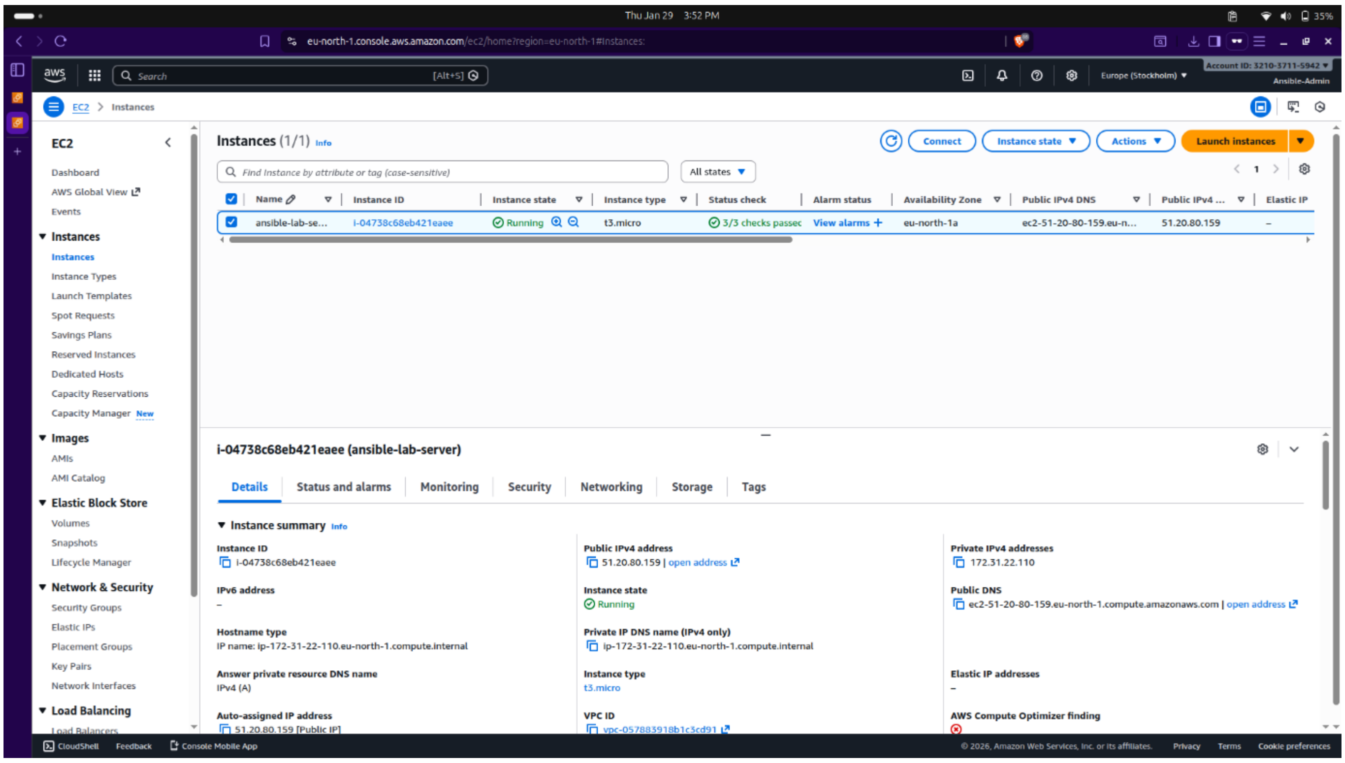
Launching instance
Creating security group rules
33%

▶ Details

Please wait while we launch your instance.

Do not close your browser while this is loading.

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



Instances (1/1)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
ansible-lab-se...	i-04738c68eb421eaee	Running	t3.micro	3/3 checks passec	View alarms +	eu-north-1a	ec2-51-20-80-159.eu-n...	51.20.80.159	–

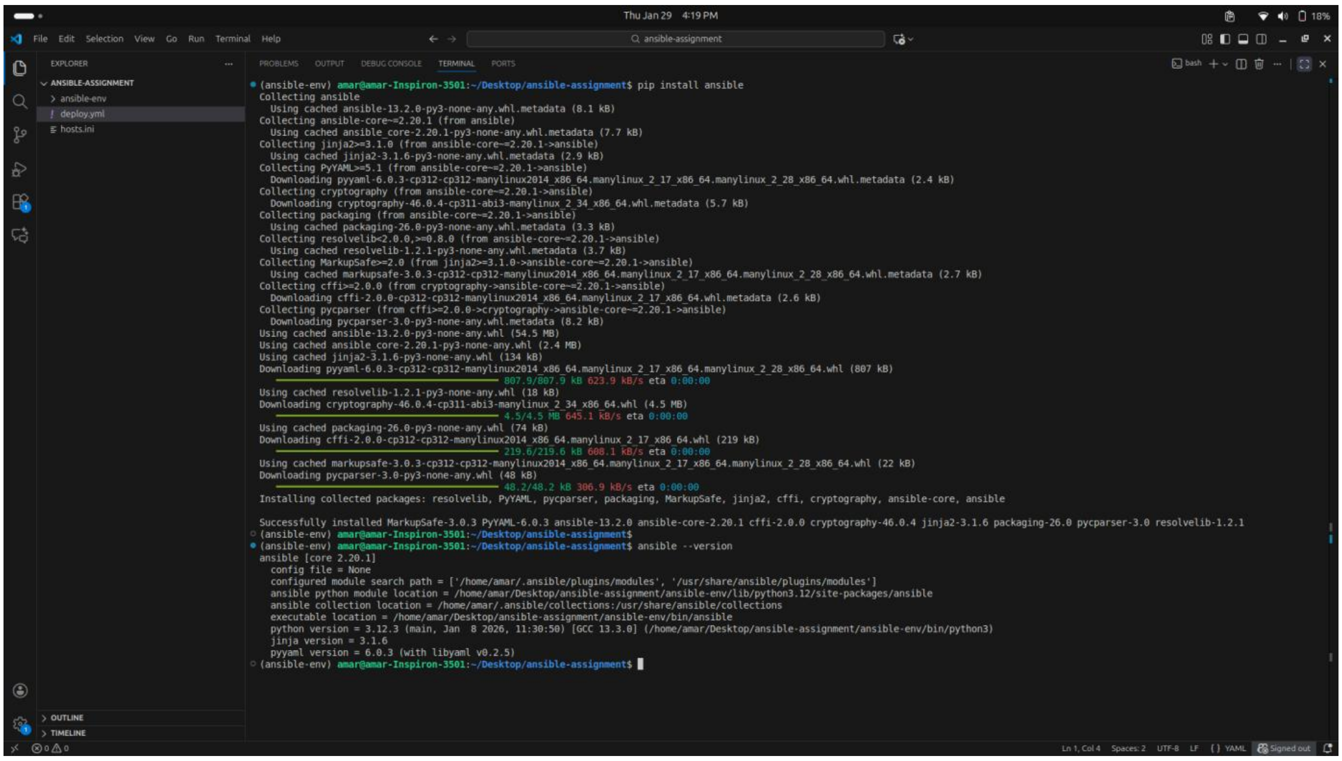
i-04738c68eb421eaee (ansible-lab-server)

Instance summary

Instance ID: i-04738c68eb421eaee	Public IPv4 address: 51.20.80.159 | open address	Private IPv4 addresses: 172.31.22.110
IPv6 address: –	Instance state: Running	Public DNS: ec2-51-20-80-159.eu-north-1.compute.amazonaws.com | open address
Hostname type: IP name: ip-172-31-22-110.eu-north-1.compute.internal	Private IP DNS name (IPv4 only): ip-172-31-22-110.eu-north-1.compute.internal	
Answer private resource DNS name: IPv4 (A)	Instance type: t3.micro	Elastic IP addresses: –
Auto-assigned IP address: 51.20.80.159 [Public IP]	VPC ID: vpc-057883918b1c3cd91	AWS Compute Optimizer finding

Step 2: Ansible Installation on Control Node

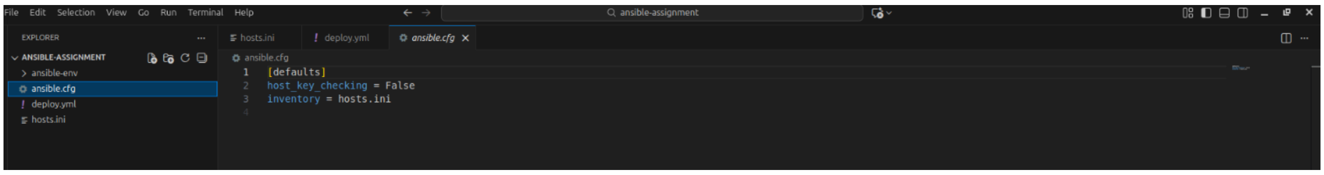
- On the local Ubuntu machine (Control Node), a Python virtual environment was used to install Ansible via pip install ansible.



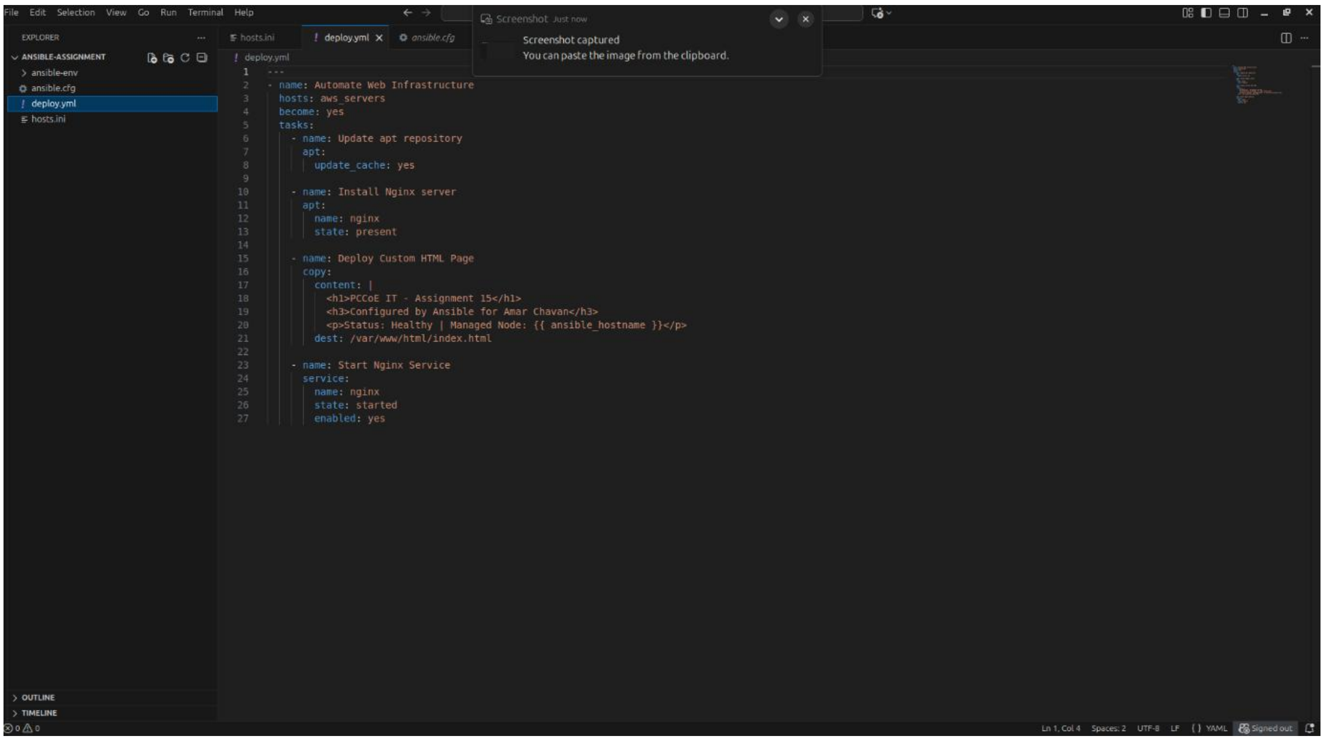
```
(ansible-env) amar@amar-Inspiron-3501:~/Desktop/ansible-assignment$ pip install ansible
Collecting ansible
  Using cached ansible-13.2.0-py3-none-any.whl.metadata (8.1 kB)
Collecting ansible-core~=2.20.1 (from ansible)
  Using cached ansible_core-2.20.1-py3-none-any.whl.metadata (7.7 kB)
Collecting jinja2>=3.1.0 (from ansible-core~=2.20.1->ansible)
  Using cached jinja2-3.1.6-py3-none-any.whl.metadata (2.9 kB)
Collecting PyYAML>=5.1 (from ansible-core~=2.20.1->ansible)
  Downloading pyyaml-6.0.3-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl.metadata (2.4 kB)
Collecting cryptography (from ansible-core~=2.20.1->ansible)
  Downloading cryptography-46.0.4-cp311-abi3-manylinux_2_34_x86_64.whl.metadata (5.7 kB)
Collecting packaging (from ansible-core~=2.20.1->ansible)
  Using cached packaging-26.0-py3-none-any.whl.metadata (3.3 kB)
Collecting resolvelib<2.0.0,>=0.8.0 (from ansible-core~=2.20.1->ansible)
  Using cached resolvelib-1.2.1-py3-none-any.whl.metadata (3.7 kB)
Collecting MarkupSafe>=2.0 (from jinja2>=3.1.0->ansible-core~=2.20.1->ansible)
  Using cached markupsafe-3.0.3-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl.metadata (2.7 kB)
Collecting cffi>=2.0.0 (from cryptography->ansible-core~=2.20.1->ansible)
  Downloading cffi-2.0.0-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata (2.6 kB)
Collecting pycparser (from cffi>=2.0.0->cryptography->ansible-core~=2.20.1->ansible)
  Downloading pycparser-3.0-py3-none-any.whl.metadata (8.2 kB)
Using cached ansible-13.2.0-py3-none-any.whl (54.5 MB)
Using cached ansible_core-2.20.1-py3-none-any.whl (2.4 MB)
Using cached jinja2-3.1.6-py3-none-any.whl (134 kB)
Downloading pyyaml-6.0.3-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl (807 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 807.9/807.9 kB 623.9 kB/s eta 0:00:00
Using cached resolvelib-1.2.1-py3-none-any.whl (18 kB)
Downloading cryptography-46.0.4-cp311-abi3-manylinux_2_34_x86_64.whl (4.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.5/4.5 MB 645.1 kB/s eta 0:00:00
Using cached packaging-26.0-py3-none-any.whl (74 kB)
Downloading cffi-2.0.0-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.whl (219 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 219.6/219.6 kB 608.1 kB/s eta 0:00:00
Using cached markupsafe-3.0.3-cp312-cp312-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl (22 kB)
Downloading pycparser-3.0-py3-none-any.whl (48 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 48.2/48.2 kB 306.9 kB/s eta 0:00:00
Installing collected packages: resolvelib, PyYAML, pycparser, packaging, MarkupSafe, jinja2, cffi, cryptography, ansible-core, ansible

Successfully installed MarkupSafe-3.0.3 PyYAML-6.0.3 ansible-13.2.0 ansible-core-2.20.1 cffi-2.0.0 cryptography-46.0.4 jinja2-3.1.6 packaging-26.0 pycparser-3.0 resolvelib-1.2.1
(ansible-env) amar@amar-Inspiron-3501:~/Desktop/ansible-assignment$
(ansible-env) amar@amar-Inspiron-3501:~/Desktop/ansible-assignment$ ansible --version
ansible [core 2.20.1]
  config file = None
  configured module search path = ['/home/amar/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /home/amar/Desktop/ansible-assignment/ansible-env/lib/python3.12/site-packages/ansible
  ansible collection location = /home/amar/.ansible/collections:/usr/share/ansible/collections
  executable location = /home/amar/Desktop/ansible-assignment/ansible-env/bin/ansible
  python version = 3.12.3 (main, Jan  8 2026, 11:30:50) [GCC 13.3.0] (/home/amar/Desktop/ansible-assignment/ansible-env/bin/python3)
  jinja version = 3.1.6
  pyyaml version = 6.0.3 (with libyaml v0.2.5)
(ansible-env) amar@amar-Inspiron-3501:~/Desktop/ansible-assignment$
```

Step 3: Configuration & Inventory Setup

- An **ansible.cfg** file was created to disable host key checking and specify the inventory file.
- A **hosts.ini** file was configured with the Public IP of the managed node (**51.20.80.159**), the remote user (**ubuntu**), and the path to the private key file.

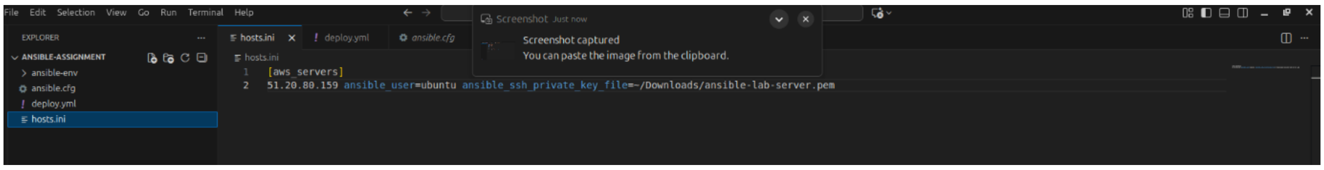


```
[defaults]
host_key_checking = False
inventory = hosts.ini
```


File Edit Selection View Go Run Terminal Help
EXPLORER
ANSIBLE-ASSIGNMENT
ansible-env
ansible.cfg
deploy.yml
hosts.ini
OUTLINE
TIMELINE
Ln 1, Col 4 Spaces: 2 UTF-8 LF { } YAML Signed out

Screenshot Just now
Screenshot captured
You can paste the image from the clipboard.

```
---
- name: Automate Web Infrastructure
  hosts: aws_servers
  become: yes
  tasks:
    - name: Update apt repository
      apt:
        update_cache: yes

    - name: Install Nginx server
      apt:
        name: nginx
        state: present

    - name: Deploy Custom HTML Page
      copy:
        content: |
          <h1>PCCoE IT - Assignment 15</h1>
          <h3>Configured by Ansible for Amar Chavan</h3>
          <p>Status: Healthy | Managed Node: {{ ansible_hostname }}</p>
        dest: /var/www/html/index.html

    - name: Start Nginx Service
      service:
        name: nginx
        state: started
        enabled: yes
```

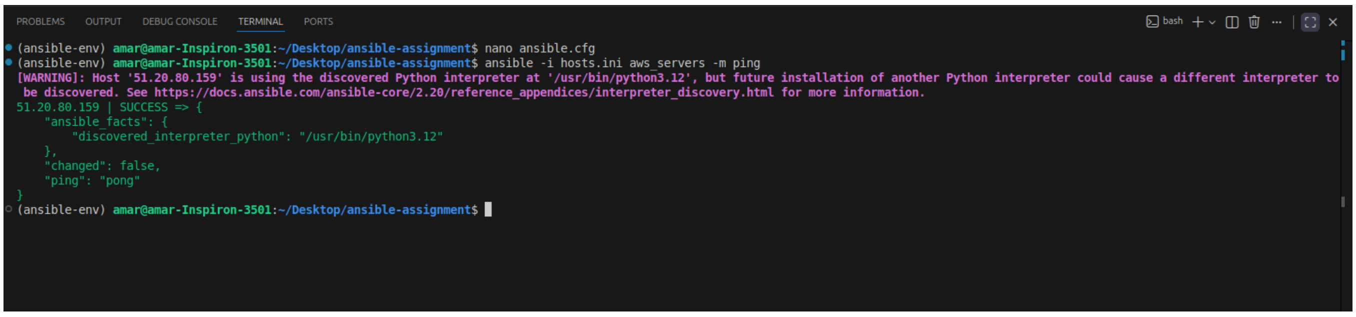


Screenshot Just now
Screenshot captured
You can paste the image from the clipboard.

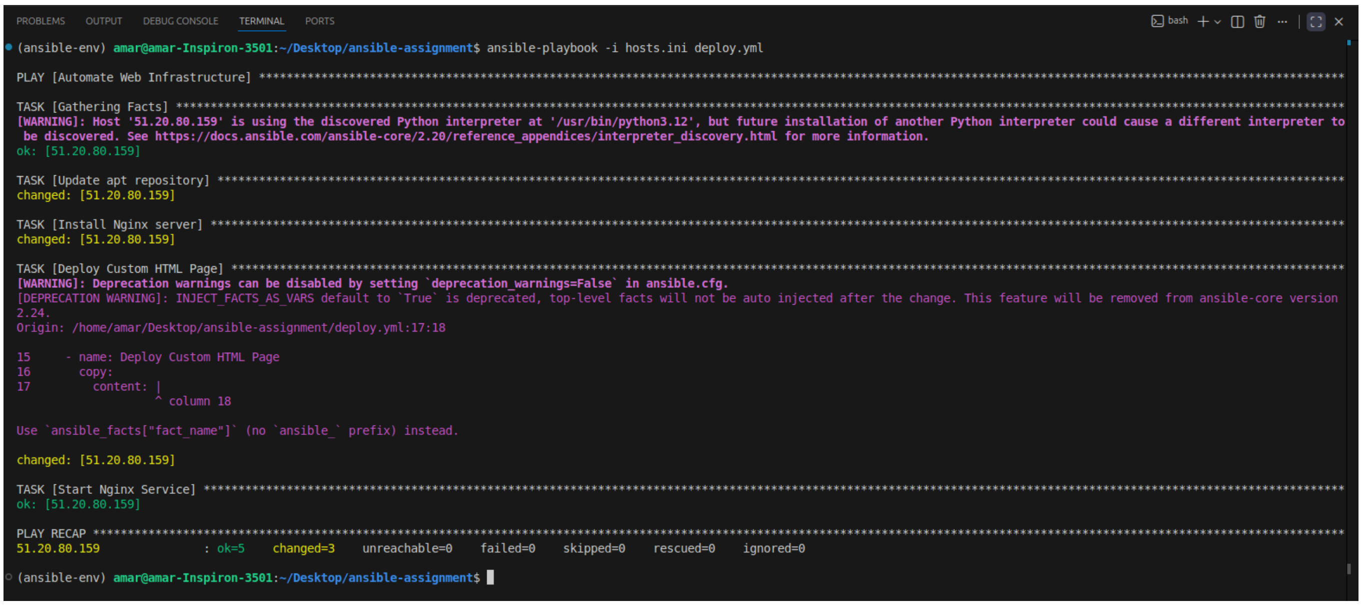
```
[aws_servers]
51.20.80.159 ansible_user=ubuntu ansible_ssh_private_key_file=~/Downloads/ansible-lab-server.pem
```

Step 4: Creating and Running the Playbook

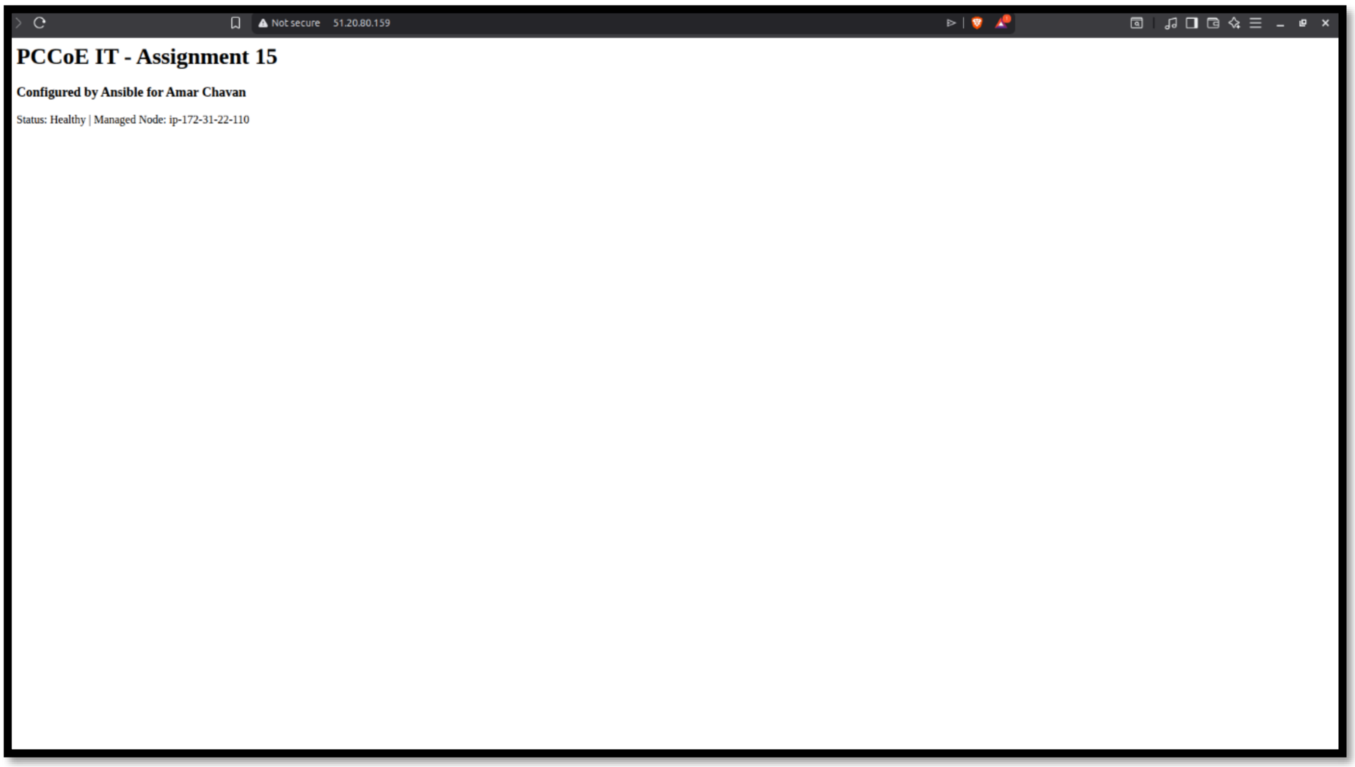
- A playbook named **deploy.yml** was written to automate the following tasks:
    1. Update the apt repository cache.
    2. Install the **Nginx** web server.
    3. Deploy a custom HTML page containing "PCCoE IT - Assignment 15".
    4. Ensure the Nginx service is started and enabled on boot.
- **Execution:** The command ansible-playbook -i hosts.ini deploy.yml was executed.



```
(ansible-env) amar@amar-Inspiron-3501:~/Desktop/ansible-assignment$ nano ansible.cfg
(ansible-env) amar@amar-Inspiron-3501:~/Desktop/ansible-assignment$ ansible -i hosts.ini aws_servers -m ping
[WARNING]: Host '51.20.80.159' is using the discovered Python interpreter at '/usr/bin/python3.12', but future installation of another Python interpreter could cause a different interpreter to
 be discovered. See https://docs.ansible.com/ansible-core/2.20/reference_appendices/interpreter_discovery.html for more information.
51.20.80.159 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3.12"
    },
    "changed": false,
    "ping": "pong"
}
(ansible-env) amar@amar-Inspiron-3501:~/Desktop/ansible-assignment$
```



```
(ansible-env) amar@amar-Inspiron-3501:~/Desktop/ansible-assignment$ ansible-playbook -i hosts.ini deploy.yml

PLAY [Automate Web Infrastructure] ******************************************************************************************************************************************

TASK [Gathering Facts] ******************************************************************************************************************************************************
[WARNING]: Host '51.20.80.159' is using the discovered Python interpreter at '/usr/bin/python3.12', but future installation of another Python interpreter could cause a different interpreter to
 be discovered. See https://docs.ansible.com/ansible-core/2.20/reference_appendices/interpreter_discovery.html for more information.
ok: [51.20.80.159]

TASK [Update apt repository] ************************************************************************************************************************************************
changed: [51.20.80.159]

TASK [Install Nginx server] *************************************************************************************************************************************************
changed: [51.20.80.159]

TASK [Deploy Custom HTML Page] **********************************************************************************************************************************************
[WARNING]: Deprecation warnings can be disabled by setting `deprecation_warnings=False` in ansible.cfg.
[DEPRECATION WARNING]: INJECT_FACTS_AS_VARS default to `True` is deprecated, top-level facts will not be auto injected after the change. This feature will be removed from ansible-core version
2.24.
Origin: /home/amar/Desktop/ansible-assignment/deploy.yml:17:18

15     - name: Deploy Custom HTML Page
16       copy:
17         content: |
                    ^ column 18

Use `ansible_facts["fact_name"]` (no `ansible_` prefix) instead.

changed: [51.20.80.159]

TASK [Start Nginx Service] **************************************************************************************************************************************************
ok: [51.20.80.159]

PLAY RECAP ******************************************************************************************************************************************************************
51.20.80.159               : ok=5    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

(ansible-env) amar@amar-Inspiron-3501:~/Desktop/ansible-assignment$
```



PCCoE IT - Assignment 15

Configured by Ansible for Amar Chavan

Status: Healthy | Managed Node: ip-172-31-22-110

Conclusion

The successful completion of Assignment 15 confirms that Ansible is a superior choice for configuration management due to its simplicity, human-readable YAML syntax, and agentless nature. Through the Amelco case study, the enterprise value of automation—including consistency across environments and significantly reduced deployment times—was clearly identified. The practical demonstration showed how a single Ansible Playbook could orchestrate complex software installations and configurations on a remote AWS EC2 instance securely and efficiently. By implementing Infrastructure as Code (IaC) principles, we achieved a repeatable and reliable deployment process that eliminates manual errors, forming a critical component of modern professional DevOps pipelines.