Red Hat

ANSIBLE

RHCE
CAPSTONE PROJECT

AUTOMATED ENTERPRISE LINUX INFRASTRUCTURE DEPLOYMENT USING ANSIBLE

Design, build, and automate a secure, scalable, and enterprise-grade Linux infrastructure using Red Hat Enterprise Linux and Ansible.


AUTOMATION


SECURITY


INFRASTRUCTURE


RELIABILITY


SCALABILITY

PREPARED BY

NANDU SIVADAS

CLOUD & DEVOPS ENTHUSIAST

DATE	:	July 2026
PROJECT TYPE	:	RHCE Capstone Project
TECHNOLOGY	:	Red Hat Enterprise Linux, Ansible
FOCUS AREA	:	Automation, Infrastructure, Security, Monitoring & Scalability

Introduction

The **RHCE Capstone Project** demonstrates the implementation of enterprise Linux infrastructure automation using **Red Hat Enterprise Linux (RHEL)** and **Ansible**. The project automates essential system administration tasks, including inventory management, software installation, service configuration, user management, security, monitoring, and load balancing. By implementing Infrastructure as Code (IaC) principles, the project provides a scalable, consistent, and efficient approach to managing enterprise Linux environments.

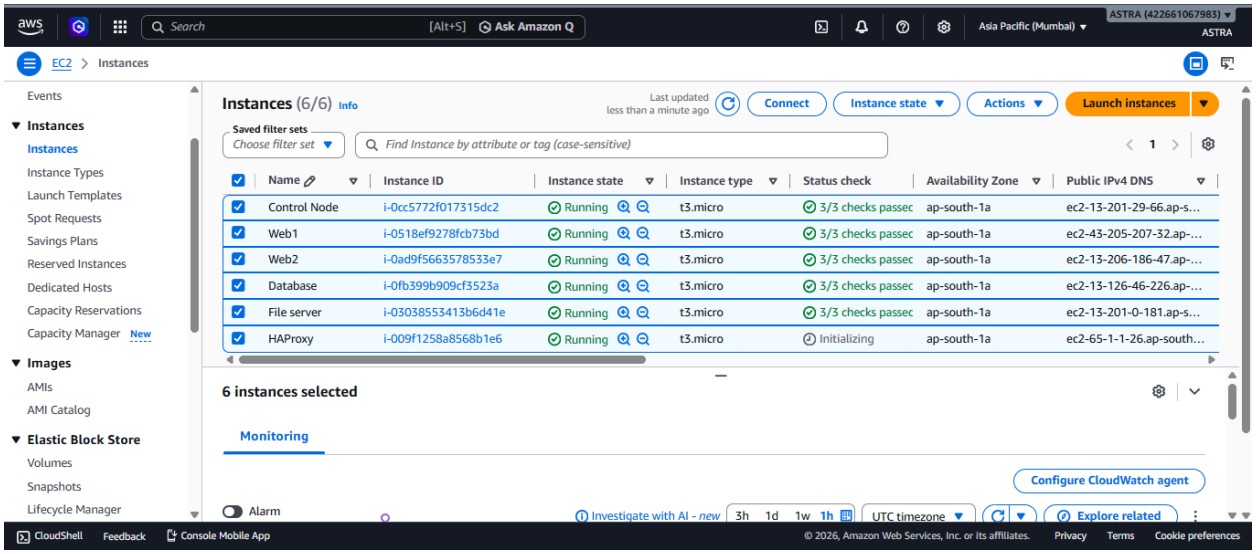
Project Objective

The primary objective of this project is to design and automate an enterprise Linux infrastructure using **Ansible** on **Red Hat Enterprise Linux**. The project demonstrates automation through static and dynamic inventories, playbooks, reusable roles, variables, handlers, and Jinja2 templates, enabling centralized management and consistent configuration across multiple Linux servers.

Infrastructure Overview

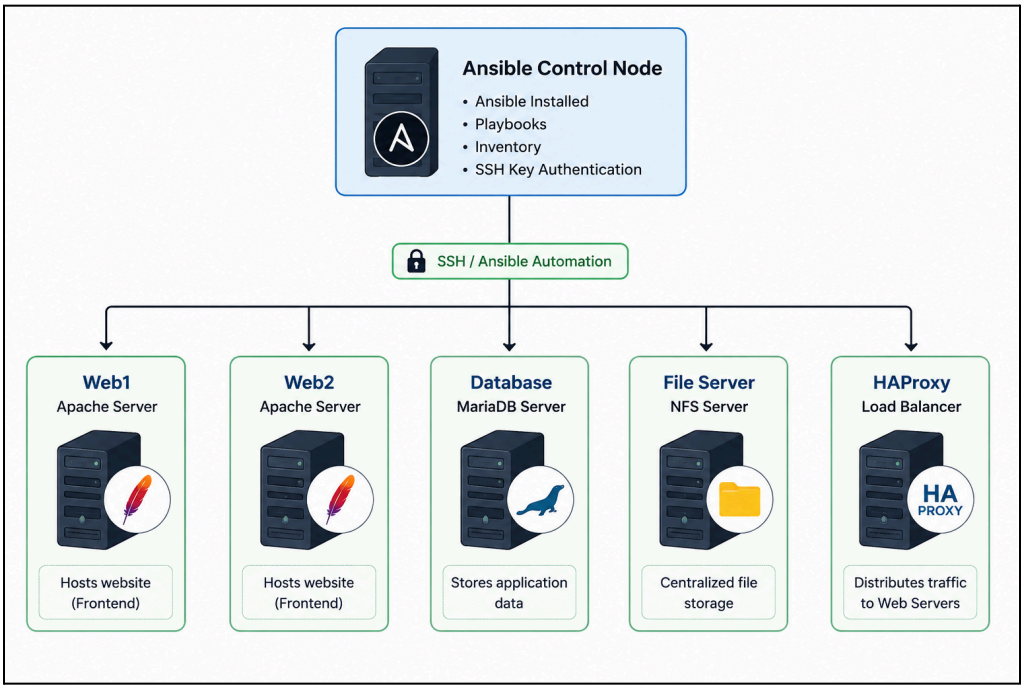
The project environment consists of one **Ansible Control Node** and five managed Linux servers hosted on AWS EC2. The Control Node centrally manages all servers through Ansible over SSH, while each managed node performs a dedicated enterprise role.

Server		Purpose
Control Node	-	Ansible Automation
Web1	-	Apache Web Server
Web2	-	Apache Web Server
Database	-	MariaDB Database Server
File Server	-	NFS File Server
HAProxy	-	Load Balancer

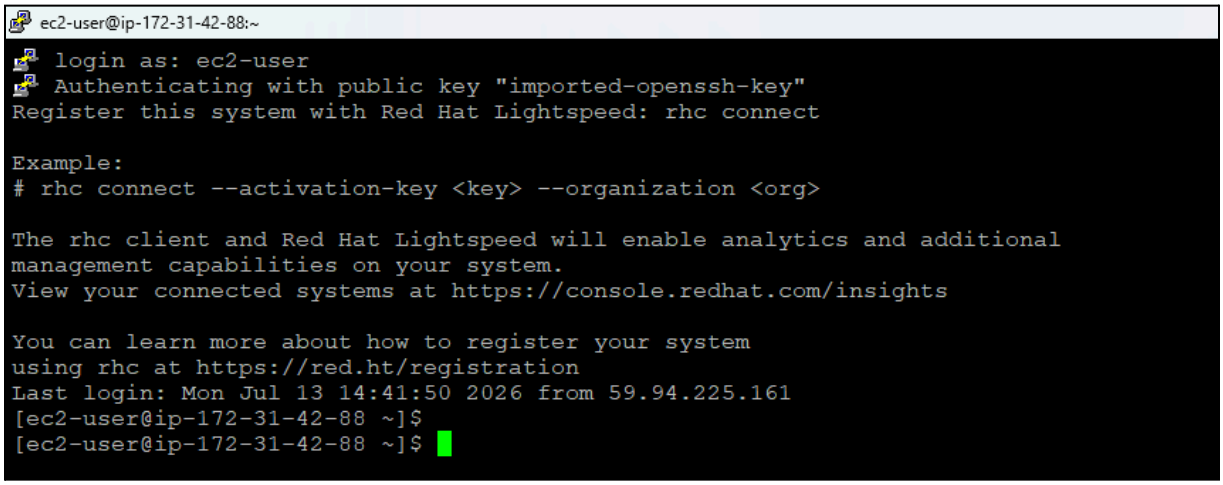


Architecture Diagram

The following architecture illustrates the automated enterprise Linux infrastructure implemented in this project. The **Ansible Control Node** communicates with all managed servers using SSH and executes playbooks to automate configuration and administration tasks. Each managed node provides a specific service within the infrastructure, including web hosting, database management, file sharing, and load balancing.



Connect to all EC2 instances using PuTTY

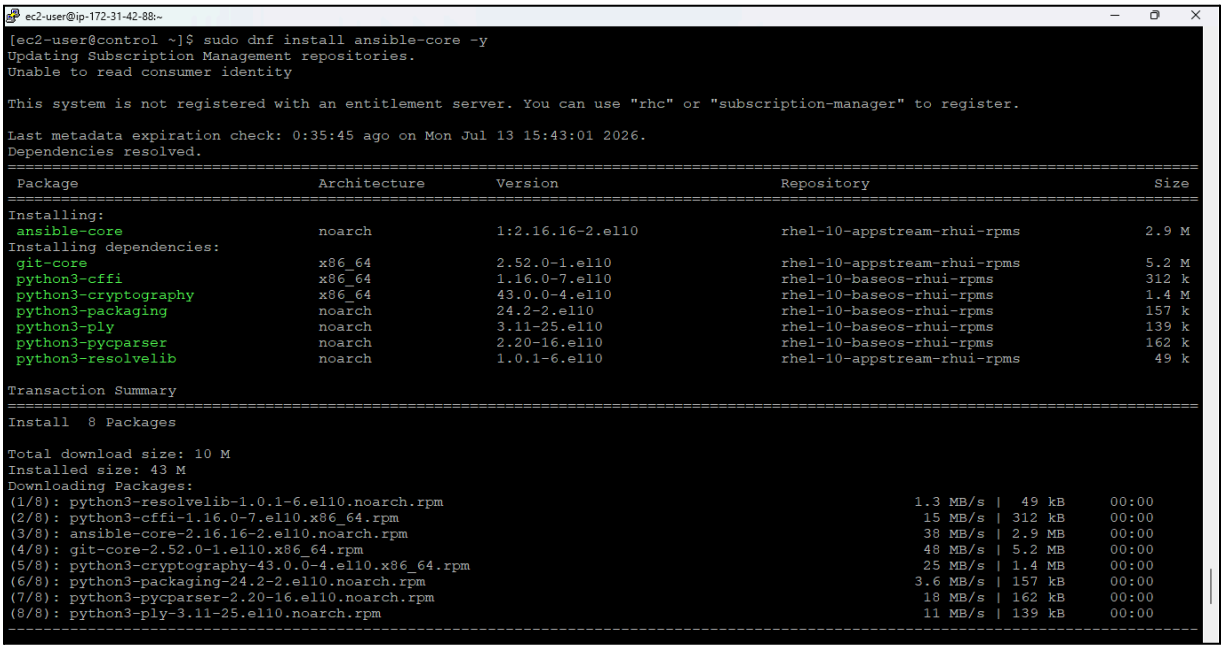


Install and Configure the Ansible Control Node

The Ansible Control Node was configured on a Red Hat Enterprise Linux (RHEL) server to centrally manage and automate multiple Linux servers. The system was updated, Ansible was installed, and the installation was verified using the `ansible --version` command to ensure the environment was ready for automation.

(1) Update the System (# sudo dnf update -y)

(2) Install Ansible (# sudo dnf install ansible-core -y)



(3) Verify the Installation (# ansible --version)

```
[ec2-user@control ~]$ ansible --version
ansible [core 2.16.16]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/ec2-user/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.12/site-packages/ansible
  ansible collection location = /home/ec2-user/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.13 (main, Apr 16 2026, 00:00:00) [GCC 14.3.1 20251022 (Red Hat 14.3.1-4)] (/usr/bin/python3)
  jinja version = 3.1.6
  libyaml = True
[ec2-user@control ~]$
```

Configure Hostnames and Host Resolution

Each server was assigned a unique hostname based on its role to simplify identification and management. The `/etc/hosts` file was updated on all servers to enable hostname resolution between the Ansible Control Node and managed nodes. This allows communication using hostnames instead of IP addresses.

Hostnames Configured

Server		Hostname
Control Node	-	control
Web Server 1	-	web1
Web Server 2	-	web2
Database Server	-	database
File Server	-	fileserver
Load Balancer	-	haproxy

(1) Configure Hostname (Control Node)

```
[ec2-user@ip-172-31-42-88 ~]$ sudo hostnamectl set-hostname control
[ec2-user@ip-172-31-42-88 ~]$ hostname
control
[ec2-user@ip-172-31-42-88 ~]$ exec bash
[ec2-user@control ~]$
```

(2) Configure Host Resolution (/etc/hosts)

```
[ec2-user@control ~]$ sudo vi /etc/hosts
[ec2-user@control ~]$ cat /etc/hosts
# Loopback entries; do not change.
# For historical reasons, localhost precedes localhost.localdomain:
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
# See hosts(5) for proper format and other examples:
# 192.168.1.10 foo.example.org foo
# 192.168.1.13 bar.example.org bar
172.31.42.88 control
172.31.46.87 web1
172.31.37.159 web2
172.31.39.170 database
172.31.43.70 fileserver
[ec2-user@control ~]$
```

Configure Passwordless SSH Authentication

Passwordless SSH authentication was configured between the Ansible Control Node and managed nodes to enable secure and automated remote administration. An SSH key pair was generated on the Control Node, and the public key was copied to the `authorized_keys` file of each managed node. Finally, SSH connectivity was verified by logging into the managed node without a password.

(1) Generate SSH Key Pair

```
[ec2-user@control ~]$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ec2-user/.ssh/id_ed25519):
Enter passphrase for "/home/ec2-user/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ec2-user/.ssh/id_ed25519
Your public key has been saved in /home/ec2-user/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:IRR1Flx/MSXY3Hfv2BsDL+MLDRWgHW3pH+pZO1K/i+0 ec2-user@control
The key's randomart image is:
+--[ED25519 256]--+
|      oo..+=+=o+o|
|     .   o+ o=+.*|
|      . .. .o.. =|
|       . .  .o o.|
|        S  .  =+.|
|           o+.Bo|
|          .o.* *|
|           .=o+.|
|            o+E+|
+----[SHA256]-----+
[ec2-user@control ~]$ cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC11ZDI1NTE5AAAAIADGnVhUIQzk2TB/WgTUDxKFQwGXHTKhfjDcj0he7r8H ec2-user@control
[ec2-user@control ~]$
```

(2) Copy the Public Key to the Managed Node

```
[ec2-user@web1 ~]$ mkdir -p ~/.ssh
[ec2-user@web1 ~]$ chmod 700 ~/.ssh
[ec2-user@web1 ~]$ vi ~/.ssh/authorized_keys
[ec2-user@web1 ~]$ cat ~/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDdATFO54s/5/vRwQ2seqZetjcoRFv4MT534NKGd4IGgJzqpcZKxfOFMNGk2D9+CV8dYujieyY8JS6qhLvZVFEP9gFJxQm0re
JRcjnX56Nw1nEJEPtkheteXFb6znbEJDwWP0w11x9WqoBgQvOOyp5zZ1RCmJSRL2q/LBVCz0TyzWf0ed/B/bzFfmzsUtBA6kYp5+eedgaWOTFU59ybZsuxmmiefsDvfmrZuMwO
3IKcypoAuxNJEKYYOo7rPhiaxVdoXMFlGFOpSJr6WMa8PYwE175MHjtFbXkVQTPVnnUfaAr+4XCkxLEU3Uvq//dirrAzvz5WvoH9oZw8lK3/bSd9 Capstone-Key
ssh-ed25519 AAAAC3NzaC11ZDI1NTE5AAAAIADGnVhUIQzk2TB/WgTUDxKFQwGXHTKhfjDcj0he7r8H ec2-user@control
[ec2-user@web1 ~]$ chmod 600 ~/.ssh/authorized_keys
[ec2-user@web1 ~]$
```

(3) Verify Passwordless SSH Login

```
[ec2-user@control ~]$ ssh web1
Register this system with Red Hat Lightspeed: rhc connect

Example:
# rhc connect --activation-key <key> --organization <org>

The rhc client and Red Hat Lightspeed will enable analytics and additional
management capabilities on your system.
View your connected systems at https://console.redhat.com/insights

You can learn more about how to register your system
using rhc at https://red.ht/registration
Last login: Mon Jul 13 15:39:34 2026 from 59.94.225.161
[ec2-user@web1 ~]$
```

Task 1 – Ansible Automation

Step 01 : Project Initialization

The Ansible project workspace was created by setting up the main project directory and the required folder structure. This layout helps organize inventories, playbooks, roles, templates, variables, and other project files efficiently.

(1) Create the Project Directory

```
[ec2-user@control ~]$ mkdir rhce-project
[ec2-user@control ~]$ cd rhce-project
[ec2-user@control rhce-project]$
```

(2) Create the Project Folder Structure

```
[ec2-user@control ~]$ mkdir rhce-project
[ec2-user@control ~]$ cd rhce-project
[ec2-user@control rhce-project]$ mkdir inventory roles playbooks templates group_vars host_vars files
[ec2-user@control rhce-project]$ ls
files  group_vars  host_vars  inventory  playbooks  roles  templates
[ec2-user@control rhce-project]$
```

(3) Verify the Project Structure (# tree output)

```
[ec2-user@control rhce-project]$ tree
.
├── files
├── group_vars
├── host_vars
├── inventory
├── playbooks
├── roles
└── templates

8 directories, 0 files
[ec2-user@control rhce-project]$
```

Step 02 : Create Static Inventory

A static inventory file was created to define the managed servers and organize them into groups. Connectivity to all servers was verified using the Ansible ping module.

(i) Create the Inventory File

```
[ec2-user@control rhce-project]$ vi inventory/hosts
[ec2-user@control rhce-project]$ cat inventory/hosts
[web]
web1
web2

[database]
database

[file]
fileserver

[haproxy]
haproxy
[ec2-user@control rhce-project]$
```

(ii) Verify Connectivity Using Ansible Ping

```
[ec2-user@control rhce-project]$ ansible all -m ping
haproxy | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
database | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
web1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
fileserver | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
web2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
[ec2-user@control rhce-project]$
```

Step 03 : Create Dynamic Inventory

The AWS EC2 dynamic inventory plugin was configured to automatically discover EC2 instances. AWS credentials and region information were provided to enable automatic inventory generation

Command used:

(i) Install Required Python Packages

```
# python3 -m pip install --user boto3 botocore
```

(ii) Configure AWS Credentials

```
# mkdir ~/.aws
# vi ~/.aws/credentials
# vi ~/.aws/config
```

(iii) Create Dynamic Inventory Configuration

```
# vi inventory/aws_ec2.yml
# ansible-inventory -i inventory/aws_ec2.yml --graph
```

(iv) Verify Dynamic Inventory

```
# ansible all -i inventory/aws_ec2.yml -m ping
```

(i) Dynamic Inventory Configuration

```
[ec2-user@control rhce-project]$ vi inventory/aws_ec2.yml
[ec2-user@control rhce-project]$ cat inventory/aws_ec2.yml
plugin: amazon.aws.aws_ec2

regions:
  - ap-south-1

hostnames:
  - tag:Name
  - private-ip-address

keyed_groups:
  - key: tags.Role
    prefix: role

compose:
  ansible_host: private_ip_address
[ec2-user@control rhce-project]$
```

(ii) Dynamic Inventory Graph

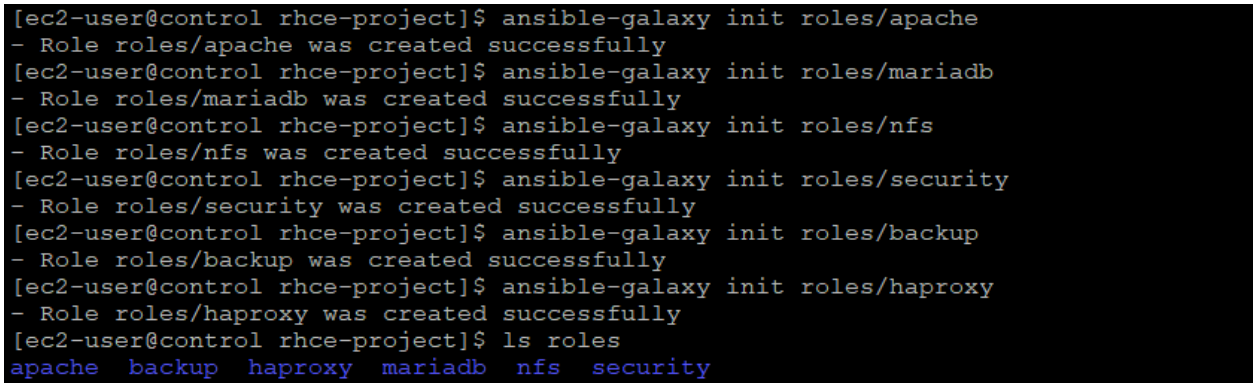
```
[ec2-user@control rhce-project]$ ansible-inventory -i inventory/aws_ec2.yml --graph
[WARNING]: Collection amazon.aws does not support Ansible version 2.16.16
[DEPRECATION WARNING]: The 'tags' host variable is deprecated. Use 'ec2_tags' instead. This feature will be removed from amazon.aws
in a release after 2026-12-01. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
@all:
  |--@ungrouped:
  |--@aws_ec2:
  |  |--HAProxy
  |  |--File server
  |  |--Web1
  |  |--Web2
  |  |--Control Node
  |  |--Database
[ec2-user@control rhce-project]$
```

(iii) Ansible Ping Verification

Step 04 : Create Ansible Roles

Ansible roles were created to organize the automation project into reusable and modular components. Each role is responsible for configuring a specific service, making the project easier to manage, maintain, and scale.

(1) Create Ansible Roles and Verify Created Roles



(2) Verify Project Directory Structure

```
[ec2-user@control rhce-project]$ tree -L 2
.
├── files
├── group_vars
├── host_vars
├── inventory
│   └── hosts
├── playbooks
│   └── site.yml
├── roles
│   ├── apache
│   ├── backup
│   ├── haproxy
│   ├── mariadb
│   ├── nfs
│   └── security
└── templates

14 directories, 2 files
[ec2-user@control rhce-project]$
```

Step 05 : Configure Ansible Inventory

(1) Create the Inventory File (# mkdir -p inventory)

(2) Add Managed Hosts (# vi inventory/hosts)

```
[ec2-user@control rhce-project]$ vi inventory/hosts
[ec2-user@control rhce-project]$ cat inventory/hosts
[web]
web1
web2

[database_servers]
database

[file]
fileserver

[loadbalancer]
haproxy
[ec2-user@control rhce-project]$
```

(3) Verify the Inventory

```
[ec2-user@control rhce-project]$ ansible-inventory -i inventory/hosts --graph
@all:
  |--@ungrouped:
  |--@web:
  |  |--web1
  |  |--web2
  |--@database_servers:
  |  |--database
  |--@file:
  |  |--fileserver
  |--@loadbalancer:
  |  |--haproxy
[ec2-user@control rhce-project]$
```

Step 06: Configure User Management

The User Management role was created to automate the creation and configuration of administrative users across all managed nodes. Using Ansible, the devops user and group were created, sudo privileges were assigned, and SSH key authentication was configured.

Implementation Overview

- Create the devops group.
- Create the devops user.
- Create the user's home directory.
- Add the user to the wheel group.
- Deploy the SSH public key for the devops user.
- Verify the user account and SSH configuration.

(1) Ansible Tasks for User Management (roles/users/tasks/main.yml)

```
[ec2-user@control rhce-project]$ vi roles/users/tasks/main.yml
[ec2-user@control rhce-project]$ cat roles/users/tasks/main.yml
---
# tasks file for roles/users

- name: Create devops group
  group:
    name: devops
    state: present

- name: Create devops user
  user:
    name: devops
    group: devops
    shell: /bin/bash
    create_home: yes
    state: present

- name: Add devops user to wheel group
  user:
    name: devops
    groups: wheel
    append: yes

- name: Copy SSH public key
  authorized_key:
    user: devops
    state: present
    key: "{{ lookup('file', 'id_ed25519.pub') }}"
[ec2-user@control rhce-project]$
```

(2) Main Playbook Including the User Management Role (site.yml)

```
[ec2-user@control rhce-project]$ vi playbooks/site.yml
[ec2-user@control rhce-project]$ cat playbooks/site.yml
---
- hosts: all
  become: yes
  roles:
    - users
```

(3) Executing the User Management playbook

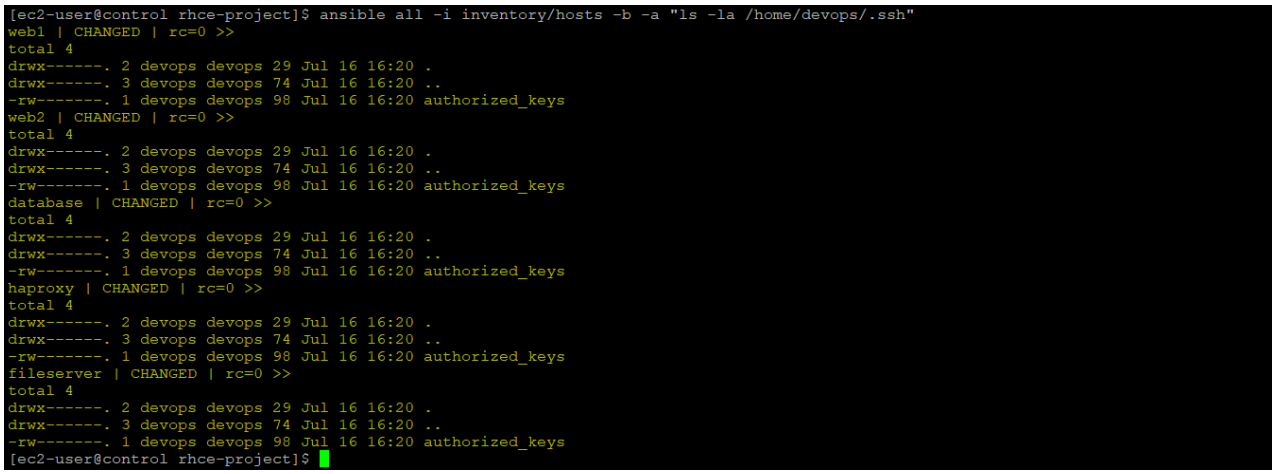
```
PLAY [all] *********************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************
ok: [database]
ok: [haproxy]
ok: [fileserver]
ok: [web1]
ok: [web2]

TASK [users : Create devops group] *********************************************************************************************************
ok: [web1]
ok: [fileserver]
ok: [database]
ok: [haproxy]
ok: [web2]

TASK [users : Create devops user] **********************************************************************************************************
ok: [fileserver]
ok: [haproxy]
ok: [web1]
ok: [database]
ok: [web2]

TASK [users : Add devops user to wheel group] **********************************************************************************************
ok: [web1]
ok: [fileserver]
ok: [web2]
ok: [database]
ok: [haproxy]

TASK [users : Copy SSH public key] *********************************************************************************************************
changed: [web1]
changed: [database]
changed: [fileserver]
changed: [haproxy]
changed: [web2]
```

(4) Verification of the created user (id devops)

```
[ec2-user@control rhce-project]$  ansible all -i inventory/hosts -b -a "id devops"
web1 | CHANGED | rc=0 >>
uid=1002(devops) gid=1002(devops) groups=1002(devops),10(wheel)
web2 | CHANGED | rc=0 >>
uid=1002(devops) gid=1002(devops) groups=1002(devops),10(wheel)
database | CHANGED | rc=0 >>
uid=1002(devops) gid=1002(devops) groups=1002(devops),10(wheel)
fileserver | CHANGED | rc=0 >>
uid=1002(devops) gid=1002(devops) groups=1002(devops),10(wheel)
haproxy | CHANGED | rc=0 >>
uid=1002(devops) gid=1002(devops) groups=1002(devops),10(wheel)
[ec2-user@control rhce-project]$
```

(5) Verification of the SSH Public Key Configuration (authorized_keys)

The verification confirms that the `.ssh` directory and `authorized_keys` file were successfully created for the `devops` user on all managed nodes.



The User Management role was successfully executed on all managed nodes. The **devops** user and group were created, sudo privileges were assigned, and the SSH public key was deployed successfully. Verification confirmed that the user account and SSH configuration were correctly configured.

Task 02 : Web Server Setup

Step 01: Install and Configure Apache HTTP Server

Apache HTTP Server was installed and configured automatically using Ansible. The Apache service was enabled and started, the firewall was configured to allow HTTP access, and a sample website was deployed successfully.

Implementation Overview

- Install Apache (httpd)
- Enable and start Apache service
- Install and enable Firewalld
- Allow HTTP (port 80)
- Deploy the sample website (index.html)
- Restore SELinux context
- Verify website access

Figure 2.1: Create the Sample HTML Webpage (index.html)

```
[ec2-user@control rhce-project]$ vi roles/apache/files/index.html
[ec2-user@control rhce-project]$ cat roles/apache/files/index.html
<!DOCTYPE html>
<html>
<head>
<title>RHCE Capstone</title>
</head>
<body>
<h1>Apache Configured Successfully Using Ansible</h1>
<h2>Welcome to Web Server</h2>
</body>
</html>
[ec2-user@control rhce-project]$
```

Figure 2.2: Ansible Tasks for Apache Installation, Firewalld Configuration, and Website Deployment

```
[ec2-user@control rhce-project]$ vi roles/apache/tasks/main.yml
[ec2-user@control rhce-project]$ cat roles/apache/tasks/main.yml
---
# tasks file for roles/apache

- name: Install Apache
  dnf:
    name: httpd
    state: present

- name: Enable Apache
  service:
    name: httpd
    enabled: yes

- name: Start Apache
  service:
    name: httpd
    state: started

- name: Copy Website
  copy:
    src: index.html
    dest: /var/www/html/index.html

- name: Install firewalld
  dnf:
    name: firewalld
    state: present

- name: Install python3-firewall
  dnf:
    name: python3-firewall
    state: present
```

```
- name: Enable firewalld
  service:
    name: firewalld
    enabled: yes
    state: started
- name: Open HTTP Port
  ansible.posix.firewalld:
    service: http
    permanent: true
    state: enabled
    immediate: true

- name: Restore SELinux Context
  command: restorecon -Rv /var/www/html
```

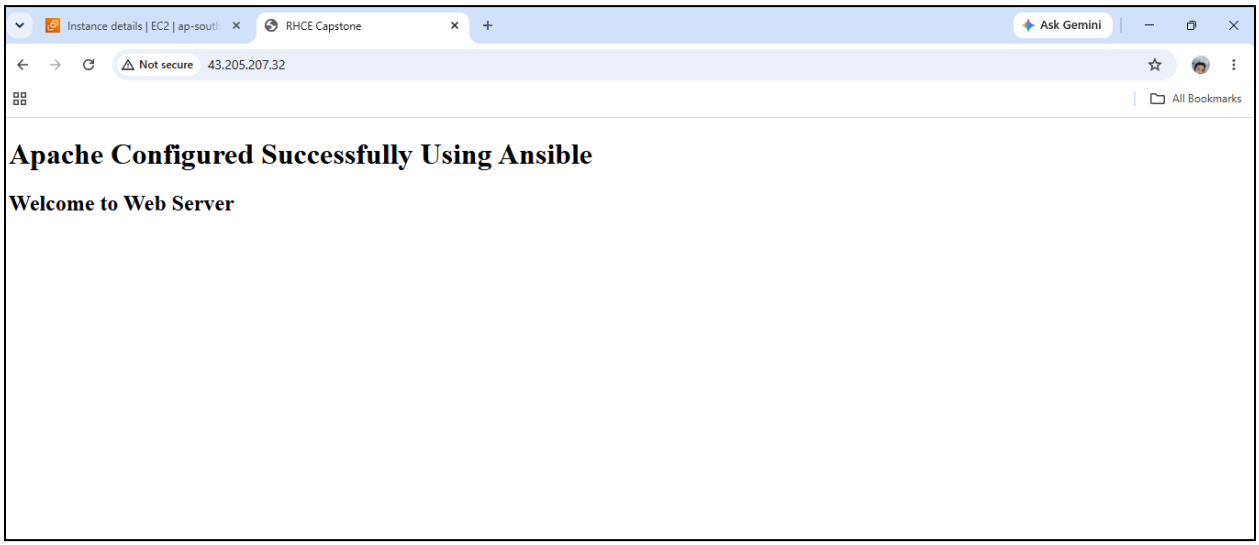
Figure 2.3: Successful execution of the Apache deployment playbook

```
[ec2-user@control rhce-project]$ ansible-playbook playbooks/site.yml
[WARNING]: Found both group and host with same name: database

PLAY [web] ***********************************************************************************************************************************

TASK [Gathering Facts] ***********************************************************************************************************************
ok: [web2]
ok: [web1]

TASK [apache : Install Apache] ***************************************************************************************************************
ok: [web1]
ok: [web2]

TASK [apache : Enable Apache] ****************************************************************************************************************
ok: [web2]
ok: [web1]

TASK [apache : Start Apache] *****************************************************************************************************************
ok: [web1]
ok: [web2]

TASK [apache : Copy Website] *****************************************************************************************************************
ok: [web1]
ok: [web2]

TASK [apache : Install firewalld] ************************************************************************************************************
ok: [web1]
ok: [web2]

TASK [apache : Install python3-firewall] *****************************************************************************************************
ok: [web1]
ok: [web2]

TASK [apache : Enable firewalld] *************************************************************************************************************
ok: [web1]
ok: [web2]

TASK [apache : Open HTTP Port] ***************************************************************************************************************
changed: [web1]
changed: [web2]

TASK [apache : Restore SELinux Context] ******************************************************************************************************
changed: [web2]
changed: [web1]

PLAY RECAP ***********************************************************************************************************************************
web1                       : ok=10   changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
web2                       : ok=10   changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ec2-user@control rhce-project]$
```

Figure 2.4**:** Apache HTTP Server running successfully on the managed node.

```
[ec2-user@control rhce-project]$ ansible web -a "systemctl status httpd"
[WARNING]: Found both group and host with same name: database
web1 | CHANGED | rc=0 >>
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
     Active: active (running) since Tue 2026-07-14 09:47:54 UTC; 23min ago
 Invocation: 162ffb5d45f34b88997927d5ecdd1acd
       Docs: man:httpd.service(8)
   Main PID: 4762 (httpd)
     Status: "Total requests: 3; Idle/Busy workers 100/0;Requests/sec: 0.00214; Bytes served/sec:   1 B/sec"
      Tasks: 230 (limit: 5048)
     Memory: 17.4M (peak: 17.7M)
        CPU: 997ms
     CGroup: /system.slice/httpd.service
             ├─4762 /usr/sbin/httpd -DFOREGROUND
             ├─4763 /usr/sbin/httpd -DFOREGROUND
             ├─4764 /usr/sbin/httpd -DFOREGROUND
             ├─4765 /usr/sbin/httpd -DFOREGROUND
             ├─4766 /usr/sbin/httpd -DFOREGROUND
             └─9114 /usr/sbin/httpd -DFOREGROUND

Jul 14 09:47:54 web1 systemd[1]: Starting httpd.service - The Apache HTTP Server...
Jul 14 09:47:54 web1 (httpd)[4762]: httpd.service: Referenced but unset environment variable evaluates to an empty string: OPTIONS
Jul 14 09:47:54 web1 httpd[4762]: AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.31.
46.87. Set the 'ServerName' directive globally to suppress this message
Jul 14 09:47:54 web1 httpd[4762]: Server configured, listening on: port 80
Jul 14 09:47:54 web1 systemd[1]: Started httpd.service - The Apache HTTP Server.
```

Figure 2.5: Sample website successfully accessed through the Apache web server.



Step 02: Configure Virtual Hosts

Apache Virtual Host was configured using an Ansible template to host the website from a dedicated document root. The custom document root (`/var/www/rhce`) was created, and the Virtual Host configuration was verified before deployment.

Implementation Overview

- Define the custom document root and server name.
- Create the custom website directory (/var/www/rhce).
- Configure the Apache Virtual Host using a Jinja2 template.
- Deploy the Virtual Host configuration.
- Restart Apache using an Ansible handler.
- Verify the configuration.

Figure 2.6: Virtual Host Variables (`group_vars/all.yml`)

```
[ec2-user@control rhce-project]$ mkdir -p group_vars
[ec2-user@control rhce-project]$ vi group_vars/all.yml
[ec2-user@control rhce-project]$ cat group_vars/all.yml
server_name: rhce.local
document_root: /var/www/rhce
[ec2-user@control rhce-project]$
```

Figure 2.7: Ansible Tasks for Virtual Host Configuration

```
[ec2-user@control rhce-project]$ vi roles/apache/tasks/main.yml
[ec2-user@control rhce-project]$ cat roles/apache/tasks/main.yml
---
# tasks file for roles/apache

- name: Install Apache
  dnf:
    name: httpd
    state: present

- name: Enable Apache
  service:
    name: httpd
    enabled: yes

- name: Start Apache
  service:
    name: httpd
    state: started

- name: Create website directory
  file:
    path: "{{ document_root }}"
    state: directory
    mode: '0755'

- name: Copy Website
  copy:
    src: index.html
    dest: "{{ document_root }}/index.html"

- name: Configure VirtualHost
  template:
    src: vhost.conf.j2
    dest: /etc/httpd/conf.d/rhce.conf
  notify:
    - Restart Apache
```

```
- name: Install firewalld
  dnf:
    name: firewalld
    state: present

- name: Install python3-firewall
  dnf:
    name: python3-firewall
    state: present

- name: Enable firewalld
  service:
    name: firewalld
    enabled: yes
    state: started
- name: Open HTTP Port
  ansible.posix.firewalld:
    service: http
    permanent: true
    state: enabled
    immediate: true

- name: Restore SELinux Context
  command: restorecon -Rv /var/www/html
[ec2-user@control rhce-project]$
```

Figure 2.8: Apache Virtual Host Template

```
[ec2-user@control rhce-project]$ vi roles/apache/templates/vhost.conf.j2
[ec2-user@control rhce-project]$ cat roles/apache/templates/vhost.conf.j2
<VirtualHost *:80>

ServerName {{ server_name }}

DocumentRoot {{ document_root }}

<Directory {{ document_root }}>
    AllowOverride All
    Require all granted
</Directory>

ErrorLog logs/{{ server_name }}_error.log
CustomLog logs/{{ server_name }}_access.log combined

</VirtualHost>
[ec2-user@control rhce-project]$
```

Figure 2.9: Generated Virtual Host Configuration

```
[ec2-user@web1 ~]$ sudo cat /etc/httpd/conf.d/rhce.conf
<VirtualHost *:80>

ServerName rhce.local

DocumentRoot /var/www/rhce

<Directory /var/www/rhce>
    AllowOverride All
    Require all granted
</Directory>

ErrorLog logs/rhce.local_error.log
CustomLog logs/rhce.local_access.log combined

</VirtualHost>
[ec2-user@web1 ~]$
```

Figure 2.10: Apache Handler (`roles/apache/handlers/main.yml`)

```
[ec2-user@control rhce-project]$ vi roles/apache/handlers/main.yml
[ec2-user@control rhce-project]$ cat roles/apache/handlers/main.yml
---
# handlers file for roles/apache
- name: Restart Apache
  service:
    name: httpd
    state: restarted
[ec2-user@control rhce-project]$
```

Figure 2.11: Virtual Host Verification

```
[ec2-user@web1 ~]$ cat /var/www/rhce/index.html
<!DOCTYPE html>
<html>
<head>
<title>RHCE Capstone</title>
</head>
<body>
<h1>Apache Configured Successfully Using Ansible</h1>
<h2>Welcome to Web Server</h2>
</body>
</html>
[ec2-user@web1 ~]$
```

Figure 2.12: Website verification

Step 03 : Enable HTTPS (Self-Signed SSL)

HTTPS support was enabled by installing the SSL module, generating a self-signed SSL certificate, configuring an HTTPS Virtual Host, and allowing HTTPS traffic through the firewall.

Implementation Overview

- Install the SSL module (`mod_ssl`).
- Generate a self-signed SSL certificate.
- Configure the HTTPS Virtual Host.
- Allow HTTPS (port 443) through Firewalld.
- Verify secure HTTPS access.

Figure 2.13: Ansible Tasks for HTTPS Configuration

```
[ec2-user@control rhce-project]$ vi roles/apache/tasks/main.yml
[ec2-user@control rhce-project]$ cat roles/apache/tasks/main.yml
---
# tasks file for roles/apache

- name: Install Apache
  dnf:
    name: httpd
    state: present

- name: Install mod_ssl
  dnf:
    name: mod_ssl
    state: present

- name: Generate SSL Certificate
  command: >
    openssl req -newkey rsa:2048 -nodes
    -keyout /etc/pki/tls/private/server.key
    -x509
    -days 365
    -out /etc/pki/tls/certs/server.crt
    -subj "/CN={{ server_name }}"
  args:
    creates: /etc/pki/tls/certs/server.crt

- name: Enable Apache
  service:
    name: httpd
    enabled: yes

- name: Start Apache
  service:
    name: httpd
    state: started
```

```
- name: Create website directory
  file:
    path: "{{ document_root }}"
    state: directory
    mode: '0755'

- name: Copy Website
  copy:
    src: index.html
    dest: "{{ document_root }}/index.html"

- name: Configure VirtualHost
  template:
    src: vhost.conf.j2
    dest: /etc/httpd/conf.d/rhce.conf
  notify:
    - Restart Apache

- name: Configure Global ServerName
  copy:
    dest: /etc/httpd/conf.d/servername.conf
    content: |
      ServerName rhce.local
  notify:
    - Restart Apache

- name: Configure HTTPS VirtualHost
  template:
    src: ssl.conf.j2
    dest: /etc/httpd/conf.d/ssl.conf
  notify:
    - Restart Apache

- name: Install firewalld
  dnf:
    name: firewalld
    state: present
```

```
- name: Install python3-firewall
  dnf:
    name: python3-firewall
    state: present

- name: Enable firewalld
  service:
    name: firewalld
    enabled: yes
    state: started

- name: Open HTTP Port
  ansible.posix.firewalld:
    service: http
    permanent: true
    state: enabled
    immediate: true

- name: Open HTTPS Firewall
  ansible.posix.firewalld:
    service: https
    permanent: yes
    state: enabled
    immediate: yes

- name: Restore SELinux Context
  command: restorecon -Rv /var/www/html
[ec2-user@control rhce-project]$
```

Figure 2.14: HTTPS Virtual Host Template

```
[ec2-user@control rhce-project]$ vi roles/apache/templates/ssl.conf.j2
[ec2-user@control rhce-project]$ cat roles/apache/templates/ssl.conf.j2
<VirtualHost *:443>

ServerName {{ server_name }}

DocumentRoot {{ document_root }}

SSLEngine on

SSLCertificateFile /etc/pki/tls/certs/server.crt
SSLCertificateKeyFile /etc/pki/tls/private/server.key

</VirtualHost>
[ec2-user@control rhce-project]$
```

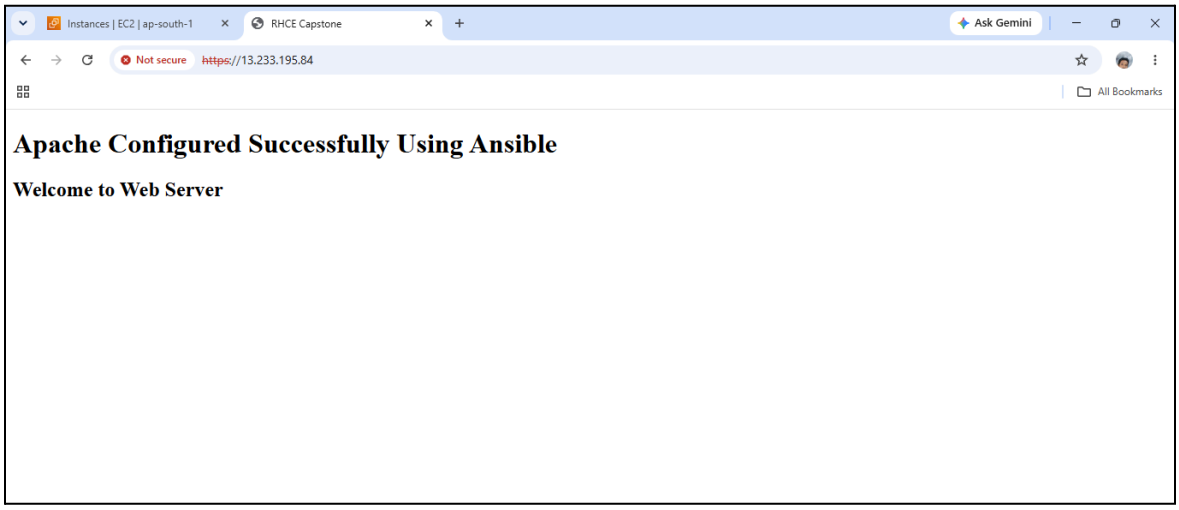
Figure 2.15: Generated HTTPS Virtual Host Configuration

```
[ec2-user@web1 ~]$ cat /etc/httpd/conf.d/ssl.conf
<VirtualHost *:443>

ServerName rhce.local

DocumentRoot /var/www/rhce

SSLEngine on

SSLCertificateFile /etc/pki/tls/certs/server.crt
SSLCertificateKeyFile /etc/pki/tls/private/server.key

</VirtualHost>
[ec2-user@web1 ~]$
```

Figure 2.16: SSL Certificate Verification

```
[ec2-user@web1 ~]$ openssl x509 -in /etc/pki/tls/certs/server.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            41:75:2b:5f:0d:aa:e1:67:65:d6:ab:b3:96:a7:b4:0a:05:f1:c8:12
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=rhce.local
        Validity
            Not Before: Jul 16 07:46:58 2026 GMT
            Not After : Jul 16 07:46:58 2027 GMT
        Subject: CN=rhce.local
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:d7:81:d2:7d:9e:ae:3a:82:5c:53:1c:89:c8:0d:
                    20:61:84:7b:1b:b9:f1:6a:23:99:bb:70:49:a0:70:
                    4d:4f:f2:6a:a8:90:d7:d0:e1:c4:96:bd:1f:19:58:
                    d2:53:b9:f3:9e:6e:66:2c:76:49:d9:3f:3c:86:1c:
                    25:d8:e0:5f:e6:f9:97:92:10:41:03:2e:11:10:b4:
                    78:50:4b:06:24:ce:ca:45:b1:81:f8:93:e5:64:9d:
                    00:38:53:8e:89:d7:43:56:91:86:0a:44:5c:bb:de:
                    ec:20:df:3b:47:69:27:d1:7d:19:36:ea:d1:52:04:
                    ff:8b:20:cb:f8:ed:22:a4:6b:81:b1:1b:44:30:5a:
                    ce:ab:e3:fe:2c:33:0e:cb:f3:3e:a8:69:99:0a:32:
                    e9:8e:82:43:da:02:df:5e:16:16:1a:ed:89:8d:3c:
                    a2:8a:81:0c:17:19:be:16:22:b1:3a:ad:09:8e:bc:
                    49:34:0f:39:a8:26:c5:97:d3:f5:d6:a8:c8:0a:f4:
                    41:b1:56:27:07:2f:f7:87:57:6d:85:5d:35:49:33:
                    36:af:76:f1:3d:11:8e:3d:a0:e5:84:00:89:d3:f2:
                    db:ed:9a:52:73:6c:8c:a6:d5:c3:59:0e:b9:dd:ba:
                    14:89:78:c6:69:2d:7c:23:da:99:68:e6:99:8d:e6:
                    0a:8f
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                4B:21:C7:77:F3:57:7F:36:A3:D4:F0:17:EC:EB:AA:93:52:7C:DC:A7
```

Figure 2.17: HTTPS Website Verification



Task 03 - Database Server

MariaDB was installed and configured automatically using Ansible. A database and database user were created with the required privileges, and the database server was secured by removing the test database and anonymous users.

Step 01: Install and Configure MariaDB

Implementation Overview

- Install MariaDB Server.
- Enable and start the MariaDB service.
- Install the required Python MySQL library.
- Verify the MariaDB service.

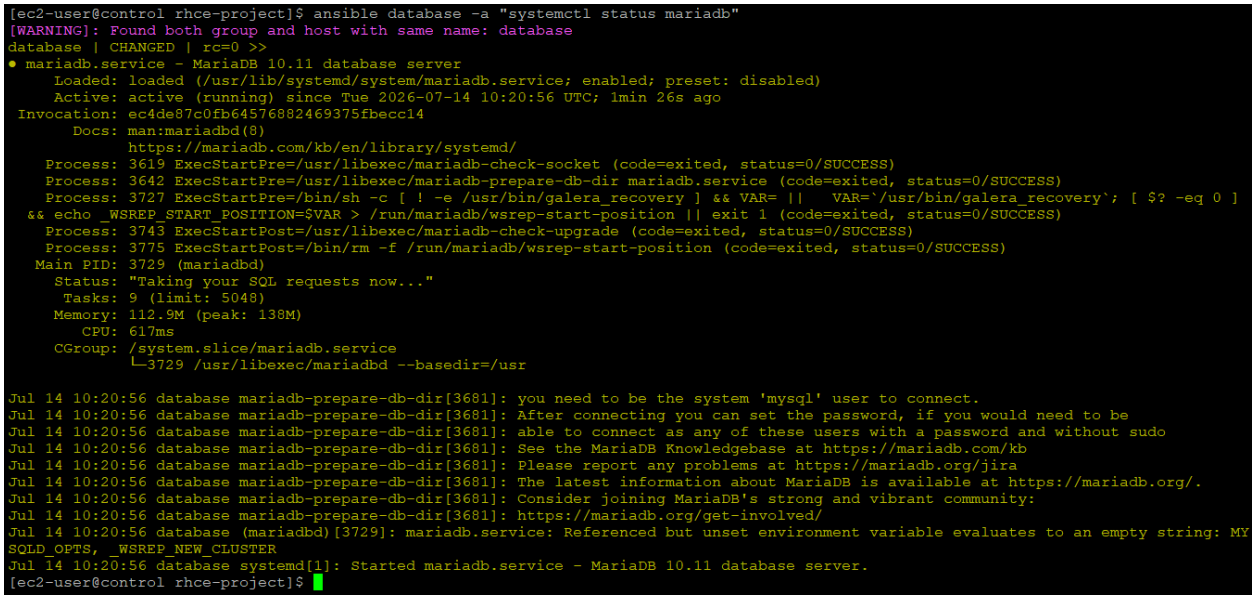
Figure 3.1: MariaDB Installation Tasks (`roles/mariadb/tasks/main.yml`)

```
[ec2-user@control rhce-project]$ vi roles/mariadb/tasks/main.yml
[ec2-user@control rhce-project]$ cat roles/mariadb/tasks/main.yml
---
# tasks file for roles/mariadb

- name: Install MariaDB Server
  dnf:
    name: mariadb-server
    state: present

- name: Enable MariaDB
  service:
    name: mariadb
    enabled: yes

- name: Start MariaDB
  service:
    name: mariadb
    state: started
```

Figure 3.2: Successful MariaDB Playbook Execution

```
PLAY [database] ****************************************************************************************************************************************

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

TASK [mariadb : Install MariaDB Server] ****************************************************************************************************************
changed: [database]

TASK [mariadb : Enable MariaDB] ************************************************************************************************************************
changed: [database]

TASK [mariadb : Start MariaDB] *************************************************************************************************************************
changed: [database]

PLAY RECAP *********************************************************************************************************************************************
database                   : ok=4    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
web1                       : ok=10   changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
web2                       : ok=10   changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ec2-user@control rhce-project]$
```

Figure 3.3: MariaDB Service Status (`systemctl status mariadb`)



Step 02: Create Database and Database User

The MariaDB database server was configured by creating a dedicated database and database user with the required privileges. The server was also secured by removing the default test database and anonymous users using Ansible.

Implementation Overview

- Remove anonymous users.
- Remove the default test database.
- Remove test database privileges.
- Create the application database (`rhcedb`).
- Create the database user (`rhceuser`).
- Grant the required database privileges.
- Verify the database configuration and security.

Figure 3.4: Database Creation and Security Tasks

The following screenshot shows the Ansible tasks used to configure the MariaDB server, including database creation, user creation, privilege assignment, and database security.

```
[ec2-user@control rhce-project]$ vi roles/mariadb/tasks/main.yml
[ec2-user@control rhce-project]$ cat roles/mariadb/tasks/main.yml
---
# tasks file for roles/mariadb

- name: Install MariaDB Server
  dnf:
    name: mariadb-server
    state: present

- name: Install PyMySQL
  dnf:
    name: python3-PyMySQL
    state: present

- name: Enable MariaDB
  service:
    name: mariadb
    enabled: yes

- name: Start MariaDB
  service:
    name: mariadb
    state: started

- name: Remove anonymous users
  community.mysql.mysql_user:
    name: ''
    host_all: true
    state: absent
    login_unix_socket: /var/lib/mysql/mysql.sock
```

```
- name: Remove test database
  community.mysql.mysql_db:
    name: test
    state: absent
    login_unix_socket: /var/lib/mysql/mysql.sock

- name: Remove test privileges
  community.mysql.mysql_user:
    name: ''
    host_all: true
    priv: "test.*:ALL"
    state: absent
    login_unix_socket: /var/lib/mysql/mysql.sock

- name: Create Database
  community.mysql.mysql_db:
    name: "{{ db_name }}"
    state: present
    login_unix_socket: /var/lib/mysql/mysql.sock

- name: Create Database User
  community.mysql.mysql_user:
    name: "{{ db_user }}"
    password: "{{ db_password }}"
    priv: "{{ db_name }}.*:ALL"
    host: "%"
    state: present
    login_unix_socket: /var/lib/mysql/mysql.sock
[ec2-user@control rhce-project]$
```

Figure 3.5: Verification of Database and User Creation

```
[ec2-user@database ~]$ sudo mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 20
Server version: 10.11.18-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| rhcedb             |
| sys                |
+--------------------+
5 rows in set (0.000 sec)

MariaDB [(none)]> SELECT User, Host FROM mysql.user;
+-------------+-----------+
| User        | Host      |
+-------------+-----------+
| rhceuser    | %         |
| mariadb.sys | localhost |
| mysql       | localhost |
| root        | localhost |
+-------------+-----------+
4 rows in set (0.001 sec)
```

Figure 3.6: Verification of Database User Privileges

```
MariaDB [(none)]> SHOW GRANTS FOR 'rhceuser'@'%';
+-----------------------------------------------------------------------------------------------------------+
| Grants for rhceuser@%                                                                                     |
+-----------------------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `rhceuser`@`%` IDENTIFIED BY PASSWORD '*27F63572E2330A3B627EB09B09E92E9BCBF08D64' |
| GRANT ALL PRIVILEGES ON `rhcedb`.* TO `rhceuser`@`%`                                                      |
+-----------------------------------------------------------------------------------------------------------+
2 rows in set (0.000 sec)

MariaDB [(none)]> exit;
Bye
[ec2-user@database ~]$
```

Task 04 - Network File Sharing (NFS)

An NFS server was configured using Ansible to share a common directory. The shared directory was mounted on the web servers, enabling centralized file sharing.

Implementation Overview

- Install NFS utilities.
- Enable and start the NFS service.
- Create the shared directory.
- Configure NFS exports.
- Export the shared directory.
- Mount the NFS share on the web servers.
- Verify the mounted shared directory.

Figure 4.1: Ansible Tasks for NFS Server Configuration

```
[ec2-user@control rhce-project]$ vi roles/nfs/tasks/main.yml
[ec2-user@control rhce-project]$ cat roles/nfs/tasks/main.yml
---
# tasks file for roles/nfs

- name: Install NFS utilities
  dnf:
    name: nfs-utils
    state: present

- name: Enable NFS Server
  service:
    name: nfs-server
    enabled: yes

- name: Start NFS Server
  service:
    name: nfs-server
    state: started

- name: Create NFS Shared Directory
  file:
    path: /shared
    state: directory
    mode: '0777'

- name: Configure Exports
  copy:
    dest: /etc/exports
    content: |
      /shared *(rw,sync,no_root_squash)

- name: Export NFS Share
  command: exportfs -rav
[ec2-user@control rhce-project]$
```

Figure 4.2: Successful Execution of the NFS Playbook

```
PLAY [file] ********************************************************************************************************************************************

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

TASK [nfs : Install NFS utilities] *********************************************************************************************************************
changed: [fileserver]

TASK [nfs : Enable NFS Server] *************************************************************************************************************************
changed: [fileserver]

TASK [nfs : Start NFS Server] **************************************************************************************************************************
changed: [fileserver]

TASK [nfs : Create NFS Shared Directory] ***************************************************************************************************************
changed: [fileserver]

TASK [nfs : Configure Exports] *************************************************************************************************************************
changed: [fileserver]

TASK [nfs : Export NFS Share] **************************************************************************************************************************
changed: [fileserver]

PLAY RECAP *********************************************************************************************************************************************
database                   : ok=4    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
fileserver                 : ok=7    changed=6    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
web1                       : ok=10   changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
web2                       : ok=10   changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ec2-user@control rhce-project]$
```

Figure 4.3: Verification of the NFS Server and Exported Shared Directory.

```
[ec2-user@fileserver ~]$ systemctl status nfs-server
● nfs-server.service - NFS server and services
     Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; enabled; preset: disabled)
    Drop-In: /run/systemd/generator/nfs-server.service.d
             └─order-with-mounts.conf
     Active: active (exited) since Sat 2026-07-18 11:37:04 UTC; 25min ago
 Invocation: eb8d989d2e8d4fdaa7574b6c9fabc51e
       Docs: man:rpc.nfsd(8)
             man:exportfs(8)
   Main PID: 5985 (code=exited, status=0/SUCCESS)
   Mem peak: 1.8M
        CPU: 37ms

Jul 18 11:37:04 fileserver systemd[1]: Starting nfs-server.service - NFS server and services...
Jul 18 11:37:04 fileserver systemd[1]: Finished nfs-server.service - NFS server and services.
[ec2-user@fileserver ~]$
```

```
[ec2-user@fileserver ~]$ sudo exportfs -v
/shared         <world>(sync,wdelay,hide,no_subtree_check,sec=sys,rw,secure,no_root_squash,no_all_squash)
[ec2-user@fileserver ~]$
```

Figure 4.4: Verification of the Mounted NFS Share

mount | grep nfs or # df -h

```
[ec2-user@web1 ~]$ mount | grep nfs
172.31.43.70:/shared on /shared type nfs4 (rw,relatime,vers=4.2,rsize=131072,wsize=131072,namlen=255,hard,fatal_neterrors=none,proto=t
cp,timeo=600,retrans=2,sec=sys,clientaddr=172.31.46.87,local_lock=none,addr=172.31.43.70)
[ec2-user@web1 ~]$ df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/nvme0n1p3        9.8G  2.3G  7.5G  24% /
devtmpfs              417M     0  417M   0% /dev
tmpfs                 454M     0  454M   0% /dev/shm
efivarfs              128K  3.3K  120K   3% /sys/firmware/efi/efivars
tmpfs                 182M  4.4M  178M   3% /run
tmpfs                 1.0M     0  1.0M   0% /run/credentials/systemd-journald.service
/dev/nvme0n1p2        200M  9.0M  191M   5% /boot/efi
tmpfs                 1.0M     0  1.0M   0% /run/credentials/serial-getty@ttyS0.service
tmpfs                 1.0M     0  1.0M   0% /run/credentials/getty@tty1.service
tmpfs                  91M  4.0K   91M   1% /run/user/1000
172.31.43.70:/shared  9.8G  2.3G  7.5G  24% /shared
```

If i created a test file on the file server: # sudo touch /shared/test.txt

```
[ec2-user@fileserver ~]$ sudo touch /shared/test.txt
[ec2-user@fileserver ~]$
```

then on **web1**: # ls /shared

```
[ec2-user@web1 ~]$ ls /shared
test.txt
[ec2-user@web1 ~]$
```

The NFS server was successfully configured, and the shared directory was mounted and verified on the web server✅

Task 05: Security Hardening

System security was enhanced by configuring SELinux, managing firewall rules with Firewalld, implementing SSH key-based authentication, and disabling direct root login.

Implementation Overview

- Configure SELinux.
- Configure Firewalld.
- Configure SSH key-based authentication.
- Disable direct root login.
- Verify the security configuration.

Figure 5.1: Ansible Tasks for Security Hardening

```
[ec2-user@control rhce-project]$ vi roles/security/tasks/main.yml
[ec2-user@control rhce-project]$ cat roles/security/tasks/main.yml
---
# tasks file for roles/security

- name: Install firewalld
  dnf:
    name: firewalld
    state: present

- name: Install python3-firewall
  dnf:
    name: python3-firewall
    state: present

- name: Enable firewalld
  service:
    name: firewalld
    enabled: yes
    state: started

- name: Allow HTTP
  ansible.posix.firewalld:
    service: http
    permanent: true
    state: enabled
    immediate: true

- name: Allow SSH
  ansible.posix.firewalld:
    service: ssh
    permanent: true
    state: enabled
    immediate: true
[ec2-user@control rhce-project]$
```

Figure 5.2: Verification of the SELinux status.

The following screenshot verifies that SELinux is running in **Enforcing** mode on the managed node.

```
[ec2-user@web1 ~]$ getenforce
Enforcing
[ec2-user@web1 ~]$
```

Figure 5.3: Verification of the Firewall Configuration

```
[ec2-user@web1 ~]$ sudo firewall-cmd --list-all
public (default, active)
  target: default
  ingress-priority: 0
  egress-priority: 0
  icmp-block-inversion: no
  interfaces: ens5
  sources:
  services: cockpit dhcpv6-client http https ssh
  ports: 9100/tcp
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[ec2-user@web1 ~]$
```

Figure 5.4: Verification of Disabled Root Login (`grep PermitRootLogin /etc/ssh/sshd_config`)

```
[ec2-user@control ~]$ sudo vi /etc/ssh/sshd_config
[ec2-user@control ~]$
```

```
# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

```
[ec2-user@control ~]$ sudo systemctl restart sshd
[ec2-user@control ~]$ sudo grep "^PermitRootLogin" /etc/ssh/sshd_config
PermitRootLogin no
[ec2-user@control ~]$
```

Task 06 - Load Balancing

HAProxy was configured as a load balancer to distribute incoming HTTP requests across multiple Apache web servers, improving availability and reliability.

Implementation Overview

- Install HAProxy.
- Configure backend web servers.
- Configure frontend listener.
- Enable and start the HAProxy service.
- Verify load balancing.

Figure 6.1: Ansible tasks for HAProxy configuration (`roles/haproxy/tasks/main.yml`)

```
[ec2-user@control rhce-project]$ vi roles/haproxy/tasks/main.yml
[ec2-user@control rhce-project]$ cat roles/haproxy/tasks/main.yml
---
# tasks file for roles/haproxy

- name: Install HAProxy
  dnf:
    name: haproxy
    state: present

- name: Copy HAProxy Configuration
  copy:
    src: haproxy.cfg
    dest: /etc/haproxy/haproxy.cfg
    owner: root
    group: root
    mode: '0644'

- name: Enable HAProxy
  service:
    name: haproxy
    enabled: yes

- name: Start HAProxy
  service:
    name: haproxy
    state: started

- name: Open HTTP Port
  ansible.posix.firewalld:
    service: http
    permanent: yes
    state: enabled
    immediate: yes
[ec2-user@control rhce-project]$
```

Figure 6.2: HAProxy configuration (`haproxy.cfg`)

```
[ec2-user@control rhce-project]$ vi roles/haproxy/files/haproxy.cfg
[ec2-user@control rhce-project]$ cat roles/haproxy/files/haproxy.cfg
global
    daemon
    maxconn 256

defaults
    mode http
    timeout connect 5s
    timeout client 50s
    timeout server 50s

frontend http_front
    bind *:80
    default_backend webservers

backend webservers
    balance roundrobin
    server web1 172.31.46.87:80 check
    server web2 172.31.37.159:80 check
[ec2-user@control rhce-project]$
```

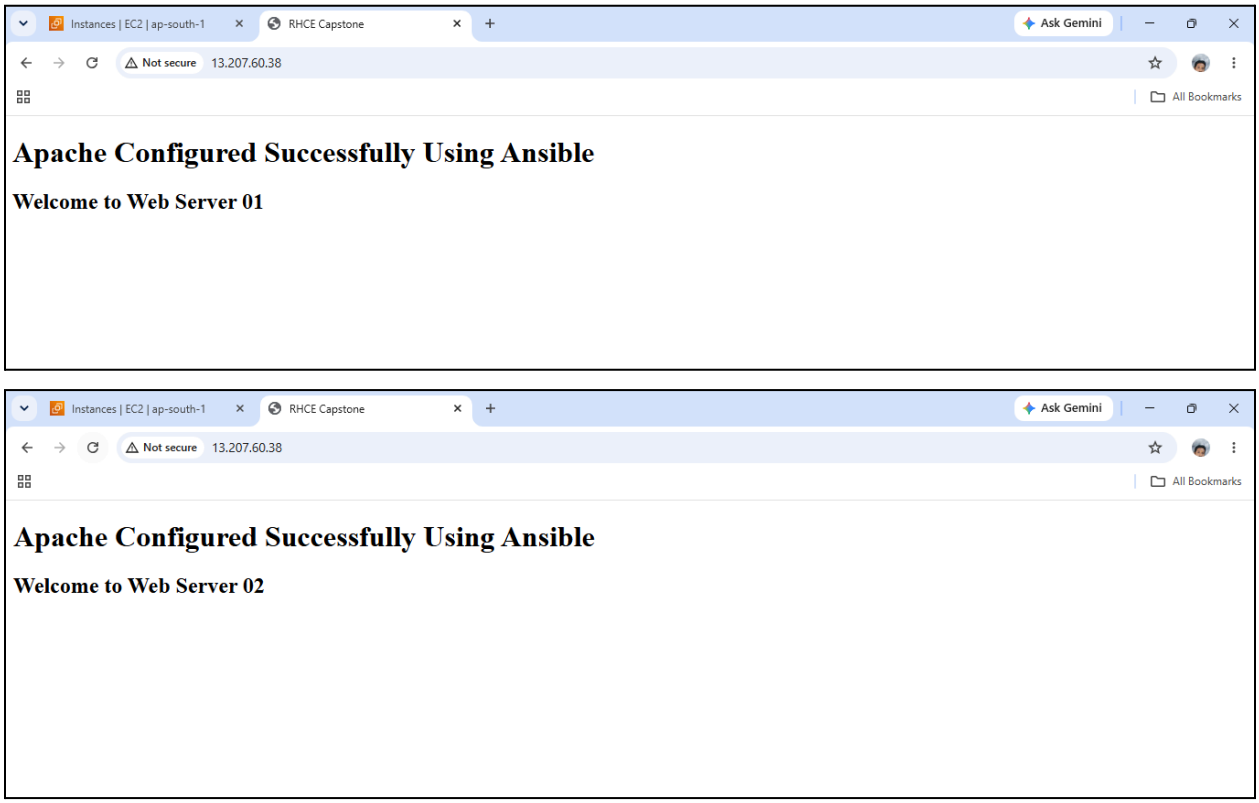
Figure 6.3: Playbook execution

```
PLAY [haproxy] *****************************************************************************************************************************************

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

TASK [haproxy : Install HAProxy] ***********************************************************************************************************************
ok: [haproxy]

TASK [haproxy : Copy HAProxy Configuration] ************************************************************************************************************
ok: [haproxy]

TASK [haproxy : Enable HAProxy] ************************************************************************************************************************
ok: [haproxy]

TASK [haproxy : Start HAProxy] *************************************************************************************************************************
ok: [haproxy]

TASK [haproxy : Open HTTP Port] ************************************************************************************************************************
ok: [haproxy]

PLAY RECAP *********************************************************************************************************************************************
database                   : ok=14   changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
fileserver                 : ok=13   changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
haproxy                    : ok=12   changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
web1                       : ok=20   changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
web2                       : ok=20   changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ec2-user@control rhce-project]$
```

Figure 6.4: Verification of the HAProxy service (`systemctl status haproxy`)

```
[ec2-user@control rhce-project]$ ansible haproxy -b -a "systemctl status haproxy"
haproxy | CHANGED | rc=0 >>
● haproxy.service - HAProxy Load Balancer
     Loaded: loaded (/usr/lib/systemd/system/haproxy.service; enabled; preset: disabled)
     Active: active (running) since Tue 2026-07-14 17:03:46 UTC; 1h 6min ago
 Invocation: 03ddf2f0068045548910f7a7363c59c4
    Process: 18359 ExecStartPre=/usr/sbin/haproxy -f $CONFIG -f $CFGDIR -c -q $OPTIONS (code=exited, status=0/SUCCESS)
   Main PID: 18361 (haproxy)
     Status: "Ready."
      Tasks: 3 (limit: 4155)
     Memory: 6.8M (peak: 7.3M)
        CPU: 812ms
     CGroup: /system.slice/haproxy.service
             ├─18361 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -f /etc/haproxy/conf.d -p /run/haproxy.pid
             └─18364 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -f /etc/haproxy/conf.d -p /run/haproxy.pid

Jul 14 17:03:46 haproxy systemd[1]: Starting haproxy.service - HAProxy Load Balancer...
Jul 14 17:03:46 haproxy haproxy[18361]: [NOTICE]   (18361) : New worker (18364) forked
Jul 14 17:03:46 haproxy haproxy[18361]: [NOTICE]   (18361) : Loading success.
Jul 14 17:03:46 haproxy systemd[1]: Started haproxy.service - HAProxy Load Balancer.
[ec2-user@control rhce-project]$
```

Figure 6.5: Verification of load balancing through the HAProxy IP or DNS in a web browser



Task 07. Monitoring & Logging

Monitoring and logging were implemented to monitor the health and performance of all managed nodes. Prometheus collects system metrics from Node Exporter, Grafana visualizes the collected metrics through dashboards, and rsyslog provides centralized system logging.

Step 01: Configure Node Exporter

Implementation Overview

- Create the Node Exporter Ansible role.
- Install the Node Exporter binary.
- Configure the systemd service.
- Enable and start the Node Exporter service.
- Open port **9100** in the firewall.

Figure 7.1: Ansible tasks for Node Exporter installation (`roles/node_exporter/tasks/main.yml`)

```
[ec2-user@control rhce-project]$ vi roles/node_exporter/tasks/main.yml
[ec2-user@control rhce-project]$ cat roles/node_exporter/tasks/main.yml
---
# tasks file for roles/node_exporter

- name: Create node_exporter user
  user:
    name: node_exporter
    shell: /sbin/nologin

- name: Copy Node Exporter archive
  copy:
    src: node_exporter-1.9.1.linux-amd64/
    dest: /tmp/node_exporter/

- name: Install Node Exporter binary
  copy:
    src: node_exporter-1.9.1.linux-amd64/node_exporter
    dest: /usr/local/bin/node_exporter
    mode: '0755'

- name: Create systemd service
  copy:
    dest: /etc/systemd/system/node_exporter.service
    content: |
      [Unit]
      Description=Prometheus Node Exporter
      After=network.target

      [Service]
      User=node_exporter
      Group=node_exporter
      ExecStart=/usr/local/bin/node_exporter

      [Install]
      WantedBy=multi-user.target
```

Figure 7.2: Successful execution of the Node Exporter playbook.

```
PLAY [all] *********************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************
ok: [web2]
ok: [web1]
ok: [fileserver]
ok: [database]
ok: [haproxy]

TASK [node_exporter : Create node_exporter user] *******************************************************************************************
changed: [web1]
changed: [haproxy]
changed: [web2]
changed: [database]
changed: [fileserver]

TASK [node_exporter : Copy Node Exporter archive] ******************************************************************************************
changed: [fileserver]
changed: [haproxy]
changed: [web1]
changed: [web2]
changed: [database]

TASK [node_exporter : Install Node Exporter binary] ****************************************************************************************
changed: [web2]
changed: [fileserver]
changed: [web1]
changed: [database]
changed: [haproxy]

TASK [node_exporter : Create systemd service] **********************************************************************************************
changed: [web1]
changed: [database]
changed: [web2]
changed: [fileserver]
changed: [haproxy]
```

```
TASK [node_exporter : Reload systemd] ******************************************************************************************
ok: [web2]
ok: [database]
ok: [web1]
ok: [fileserver]
ok: [haproxy]

TASK [node_exporter : Enable Node Exporter] ************************************************************************************
changed: [web2]
changed: [database]
changed: [fileserver]
changed: [haproxy]
changed: [web1]

TASK [node_exporter : Open Node Exporter Port] *********************************************************************************
changed: [web2]
changed: [database]
changed: [fileserver]
changed: [haproxy]
changed: [web1]

PLAY RECAP *********************************************************************************************************************
database                   : ok=22   changed=6    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
fileserver                 : ok=21   changed=7    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
haproxy                    : ok=20   changed=6    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
web1                       : ok=28   changed=9    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
web2                       : ok=28   changed=9    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[ec2-user@control rhce-project]$
```

Figure 7.3: Verification of the Node Exporter service (`systemctl status node_exporter`) and port **9100** listening.

(shows Active: **active (running)** on all servers)

```
[ec2-user@control rhce-project]$ ansible all -b -a "systemctl status node_exporter"
web2 | CHANGED | rc=0 >>
● node_exporter.service - Prometheus Node Exporter
     Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; preset: disabled)
     Active: active (running) since Tue 2026-07-14 18:54:13 UTC; 5min ago
 Invocation: 3f7b5b43303d41ca87a71039bef56de7
   Main PID: 29663 (node_exporter)
      Tasks: 4 (limit: 5048)
     Memory: 5.1M (peak: 5.4M)
        CPU: 18ms
     CGroup: /system.slice/node_exporter.service
             └─29663 /usr/local/bin/node_exporter

Jul 14 18:54:13 web2 node_exporter[29663]: time=2026-07-14T18:54:13.169Z level=INFO source=node_exporter.go:141 msg=time
Jul 14 18:54:13 web2 node_exporter[29663]: time=2026-07-14T18:54:13.169Z level=INFO source=node_exporter.go:141 msg=timex
Jul 14 18:54:13 web2 node_exporter[29663]: time=2026-07-14T18:54:13.169Z level=INFO source=node_exporter.go:141 msg=udp_queues
Jul 14 18:54:13 web2 node_exporter[29663]: time=2026-07-14T18:54:13.169Z level=INFO source=node_exporter.go:141 msg=uname
Jul 14 18:54:13 web2 node_exporter[29663]: time=2026-07-14T18:54:13.169Z level=INFO source=node_exporter.go:141 msg=vmstat
Jul 14 18:54:13 web2 node_exporter[29663]: time=2026-07-14T18:54:13.169Z level=INFO source=node_exporter.go:141 msg=watchdog
Jul 14 18:54:13 web2 node_exporter[29663]: time=2026-07-14T18:54:13.169Z level=INFO source=node_exporter.go:141 msg=xfs
Jul 14 18:54:13 web2 node_exporter[29663]: time=2026-07-14T18:54:13.169Z level=INFO source=node_exporter.go:141 msg=zfs
Jul 14 18:54:13 web2 node_exporter[29663]: time=2026-07-14T18:54:13.171Z level=INFO source=tls_config.go:347 msg="Listening on" addres
s=[::]:9100
Jul 14 18:54:13 web2 node_exporter[29663]: time=2026-07-14T18:54:13.172Z level=INFO source=tls_config.go:350 msg="TLS is disabled." ht
tp2=false address=[::]:9100
```

`# ss -tulpn | grep 9100` or `# firewall-cmd --list-ports`

```
[ec2-user@control rhce-project]$ ansible all -b -m shell -a "ss -tulnp | grep 9100"
fileserver | CHANGED | rc=0 >>
tcp   LISTEN 0      4096               *:9100             *:*    users:(("node_exporter",pid=21261,fd=3))
web1 | CHANGED | rc=0 >>
tcp   LISTEN 0      4096               *:9100            *:*    users:(("node_exporter",pid=29796,fd=3))

haproxy | CHANGED | rc=0 >>
tcp   LISTEN 0      4096               *:9100            *:*    users:(("node_exporter",pid=33144,fd=3))
database | CHANGED | rc=0 >>
tcp   LISTEN 0      4096               *:9100            *:*    users:(("node_exporter",pid=22045,fd=3))
web2 | CHANGED | rc=0 >>
tcp   LISTEN 0      4096               *:9100            *:*    users:(("node_exporter",pid=29663,fd=3))

[ec2-user@control rhce-project]$
```

Step 02: Configure Prometheus

Implementation Overview

- Install Prometheus.
- Configure `prometheus.yml`.
- Configure Prometheus as a systemd service.
- Start the Prometheus service.
- Configure all managed nodes as scrape targets.

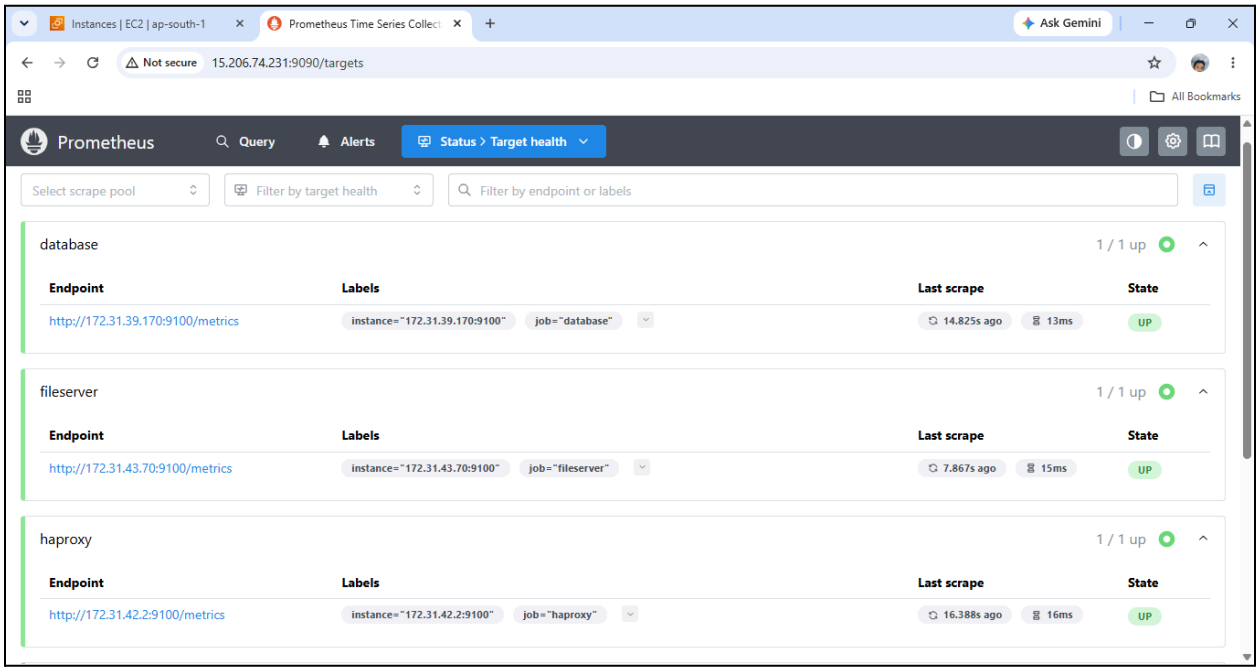
Figure 7.4: Prometheus configuration (`prometheus.yml`).

```
[ec2-user@control prometheus]$ sudo cat /etc/prometheus/prometheus.yml
# my global config
global:
  scrape_interval: 15s

scrape_configs:

  - job_name: "prometheus"

    static_configs:
      - targets:
          - localhost:9090

  - job_name: "webservers"

    static_configs:
      - targets:
          - "172.31.46.87:9100"
          - "172.31.37.159:9100"

  - job_name: "database"

    static_configs:
      - targets:
          - "172.31.39.170:9100"

  - job_name: "fileserver"

    static_configs:
      - targets:
          - "172.31.43.70:9100"

  - job_name: "haproxy"

    static_configs:
      - targets:
          - "172.31.42.2:9100"
[ec2-user@control prometheus]$
```

Figure 7.5: Verification of the Prometheus service (`systemctl status prometheus`).

```
[ec2-user@control prometheus]$ sudo systemctl status prometheus
● prometheus.service - Prometheus
     Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; preset: disabled)
     Active: active (running) since Tue 2026-07-14 19:29:29 UTC; 17s ago
 Invocation: e14ee842cb2a4e608e710979cfa0d4bd
   Main PID: 9049 (prometheus)
      Tasks: 8 (limit: 5048)
     Memory: 130.7M (peak: 132.9M)
        CPU: 108ms
     CGroup: /system.slice/prometheus.service
             └─9049 /usr/local/bin/prometheus --config.file=/etc/prometheus/prometheus.yml --storage.tsdb.path=/var/lib/prometheus

Jul 14 19:29:29 control prometheus[9049]: time=2026-07-14T19:29:29.755Z level=INFO source=head.go:744 msg="On-disk memory mappable ch>
Jul 14 19:29:29 control prometheus[9049]: time=2026-07-14T19:29:29.755Z level=INFO source=head.go:752 msg="Replaying WAL, this may ta>
Jul 14 19:29:29 control prometheus[9049]: time=2026-07-14T19:29:29.755Z level=INFO source=head.go:825 msg="WAL segment loaded" compon>
Jul 14 19:29:29 control prometheus[9049]: time=2026-07-14T19:29:29.755Z level=INFO source=head.go:862 msg="WAL replay completed" comp>
Jul 14 19:29:29 control prometheus[9049]: time=2026-07-14T19:29:29.757Z level=INFO source=main.go:1309 msg="filesystem information" f>
Jul 14 19:29:29 control prometheus[9049]: time=2026-07-14T19:29:29.757Z level=INFO source=main.go:1312 msg="TSDB started"
Jul 14 19:29:29 control prometheus[9049]: time=2026-07-14T19:29:29.757Z level=INFO source=main.go:1497 msg="Loading configuration fil>
Jul 14 19:29:29 control prometheus[9049]: time=2026-07-14T19:29:29.758Z level=INFO source=main.go:1537 msg="Completed loading of conf>
Jul 14 19:29:29 control prometheus[9049]: time=2026-07-14T19:29:29.758Z level=INFO source=main.go:1273 msg="Server is ready to receiv>
Jul 14 19:29:29 control prometheus[9049]: time=2026-07-14T19:29:29.758Z level=INFO source=manager.go:176 msg="Starting rule manager..>
lines 1-21/21 (END)
```

Figure 7.6: Prometheus Targets page showing all monitored nodes in the UP state.



Step 03: Configure Grafana

Implementation Overview

- Install Grafana.
- Start the Grafana service.
- Configure Prometheus as the data source.
- Import the Node Exporter dashboard.
- Visualize server metrics.

Figure 7.7: Successful Prometheus data source connection (Save & Test).

The following screenshot confirms that Grafana was successfully connected to the Prometheus server using the configured data source.

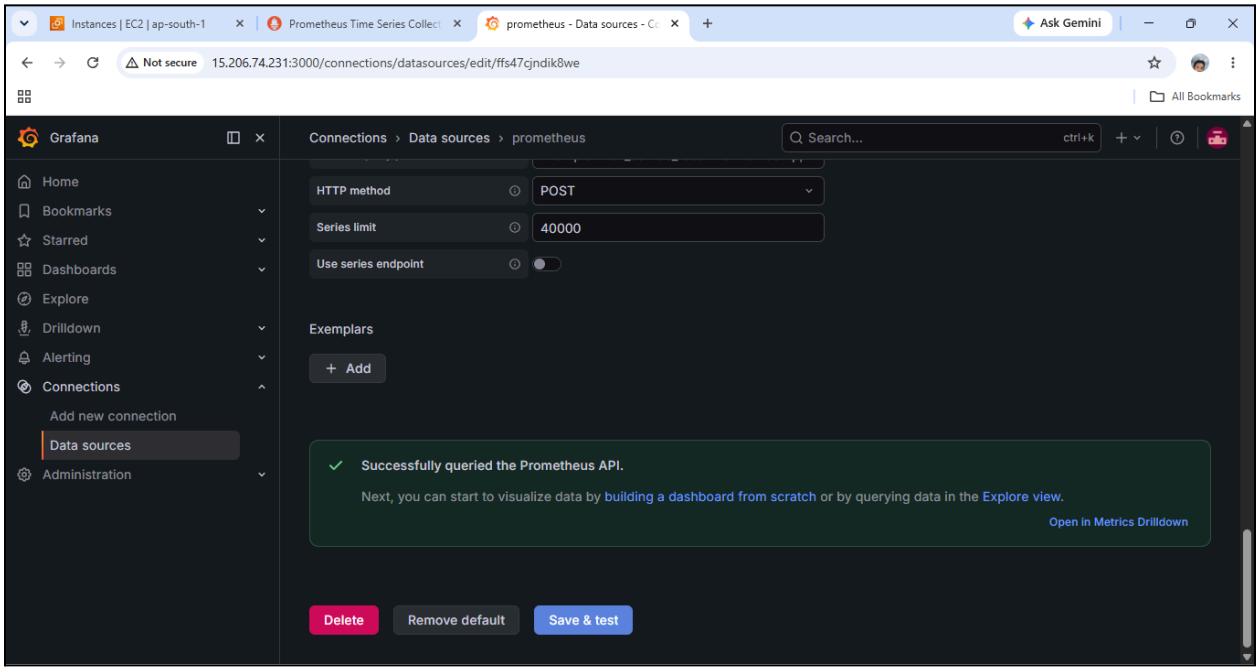
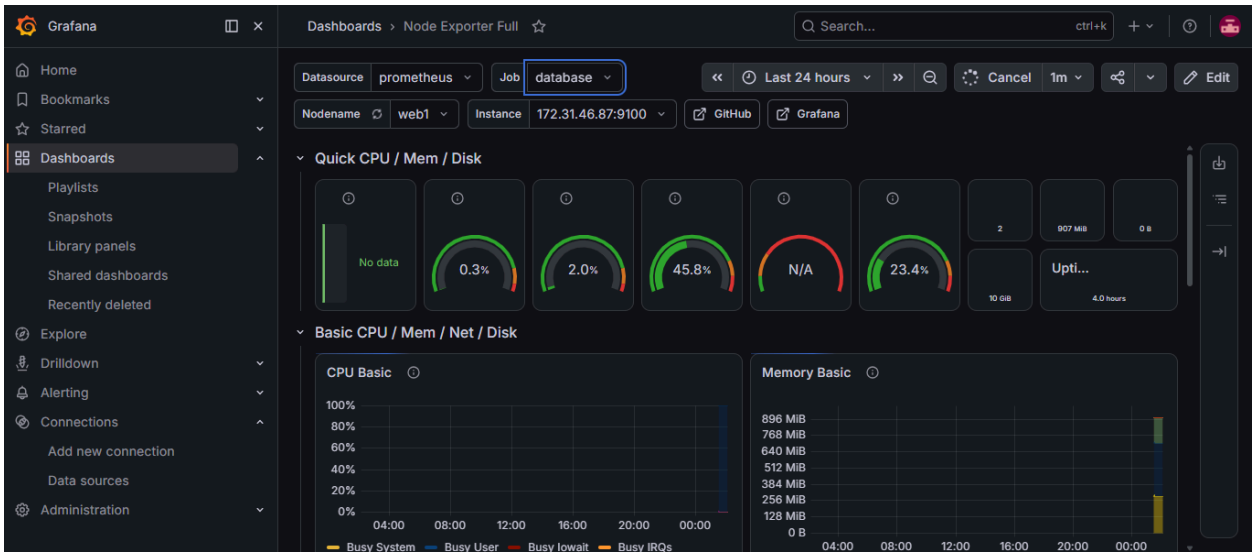


Figure 7.8: Grafana Node Exporter dashboard displaying system metrics.

The following screenshot displays the Grafana Node Exporter dashboard, which visualizes CPU, memory, disk, and other system metrics collected from the monitored nodes.



Step 04: Configure Centralized Logging (rsyslog)

Implementation Overview

- Install and enable rsyslog.
- Configure the rsyslog server.
- Configure remote logging.
- Restart the rsyslog service.
- Verify the logging service.

Figure 7.9: Ansible tasks for rsyslog configuration (`roles/rsyslog/tasks/main.yml`).

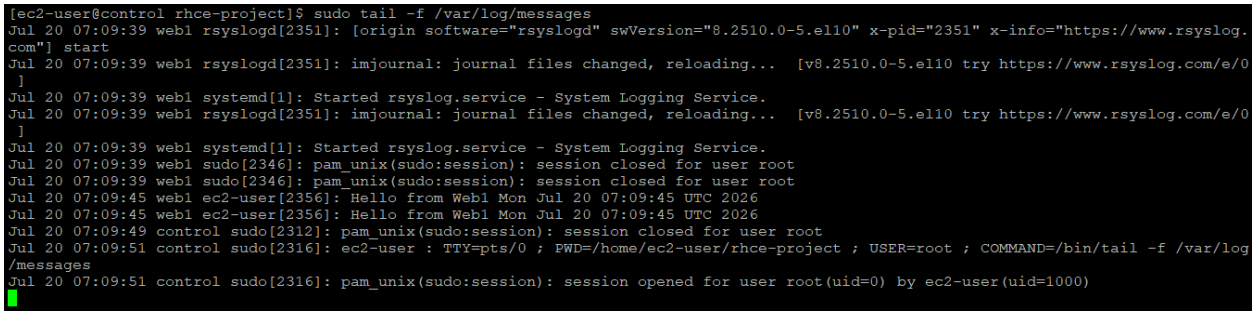
```
[ec2-user@control rhce-project]$ vi roles/rsyslog/tasks/main.yml
[ec2-user@control rhce-project]$ cat roles/rsyslog/tasks/main.yml
---
# tasks file for roles/rsyslog

- name: Install rsyslog
  dnf:
    name: rsyslog
    state: present

- name: Enable rsyslog
  service:
    name: rsyslog
    state: started
    enabled: yes

- name: Configure remote logging
  copy:
    dest: /etc/rsyslog.d/remote.conf
    content: |
      *.* @@172.31.42.88:514

- name: Restart rsyslog
  service:
    name: rsyslog
    state: restarted
```

Figure 7.10: Verification of the rsyslog service (`systemctl status rsyslog`).

```
[ec2-user@control rhce-project]$ sudo systemctl status rsyslog
● rsyslog.service - System Logging Service
     Loaded: loaded (/usr/lib/systemd/system/rsyslog.service; enabled; preset: enabled)
     Active: active (running) since Tue 2026-07-14 20:45:59 UTC; 9min ago
 Invocation: b65f5e448402481a9062404e11646a95
       Docs: man:rsyslogd(8)
             https://www.rsyslog.com/doc/
   Main PID: 10037 (rsyslogd)
      Tasks: 7 (limit: 5048)
     Memory: 3.6M (peak: 4.4M)
        CPU: 57ms
     CGroup: /system.slice/rsyslog.service
             └─10037 /usr/sbin/rsyslogd -n

Jul 14 20:45:59 control systemd[1]: Starting rsyslog.service - System Logging Service...
Jul 14 20:45:59 control systemd[1]: Started rsyslog.service - System Logging Service.
Jul 14 20:45:59 control rsyslogd[10037]: warning during parsing file /etc/rsyslog.conf, on or before line 82: STOP is followed by unr>
Jul 14 20:45:59 control rsyslogd[10037]: [origin software="rsyslogd" swVersion="8.2510.0-5.el10" x-pid="10037" x-info="https://www.rs>
Jul 14 20:45:59 control rsyslogd[10037]: imjournal: journal files changed, reloading...  [v8.2510.0-5.el10 try https://www.rsyslog.co>
lines 1-18/18 (END)
```

Figure 7.10: Verification of Centralized Logging

(sudo tail -f /var/log/messages)



Monitoring and centralized logging were successfully implemented. Node Exporter collected system metrics from all managed nodes, Prometheus monitored the targets successfully, Grafana visualized the collected metrics through dashboards, and rsyslog provided centralized logging for system events.

Task 08 - Cron Jobs & Maintenance

Automated backup tasks were configured using Ansible and scheduled with cron to ensure regular backups of the website files and MariaDB database.

Step 01: Automate Website and Database Backups

Implementation Overview

- Create website and database backup tasks.
- Store backup archives in the `/backup` directory.
- Automate the backup process using Ansible.
- Verify the generated backup archives.

Figure 8.1: Ansible Tasks for Website and Database Backup

(roles/backup/tasks/main.yml)

```
[ec2-user@control rhce-project]$ vi roles/backup/tasks/main.yml
[ec2-user@control rhce-project]$ cat roles/backup/tasks/main.yml
---
# tasks file for roles/backup

- name: Create backup directory
  file:
    path: /backup
    state: directory
    mode: '0755'

- name: Backup website
  archive:
    path: /var/www/html
    dest: /backup/web_backup.tar.gz
    format: gz

- name: Backup MariaDB data directory
  archive:
    path: /var/lib/mysql
    dest: /backup/db_backup.tar.gz
    format: gz
[ec2-user@control rhce-project]$
```

Figure 8.2: Verification of Website and Database Backup Archives

```
[ec2-user@control rhce-project]$ ansible web -b -a "ls -lh /backup"
[WARNING]: Found both group and host with same name: database
web1 | CHANGED | rc=0 >>
total 4.0K
-rw-r--r--. 1 root root 325 Jul 14 10:44 web_backup.tar.gz
web2 | CHANGED | rc=0 >>
total 4.0K
-rw-r--r--. 1 root root 326 Jul 14 10:44 web_backup.tar.gz
[ec2-user@control rhce-project]$
```

```
[ec2-user@control rhce-project]$ ansible database -b -a "ls -lh /backup"
[WARNING]: Found both group and host with same name: database
database | CHANGED | rc=0 >>
total 788K
-rw-r--r--. 1 root root 786K Jul 14 10:44 db_backup.tar.gz
[ec2-user@control rhce-project]$
```

Figure 8.3: Verification of Backup Archive Contents

```
[ec2-user@control rhce-project]$ ansible web -b -m shell -a "tar -tf /backup/web_backup.tar.gz | head"
[WARNING]: Found both group and host with same name: database
web2 | CHANGED | rc=0 >>
html/index.html
web1 | CHANGED | rc=0 >>
html/index.html
[ec2-user@control rhce-project]$ ansible database -b -m shell -a "tar -tf /backup/db_backup.tar.gz | head"
[WARNING]: Found both group and host with same name: database
database | CHANGED | rc=0 >>
mysql/mysql/
mysql/performance_schema/
mysql/sys/
mysql/aria_log_control
mysql/aria_log.00000001
mysql/ibdata1
mysql/ib_logfile0
mysql/ib_buffer_pool
mysql/mysql_upgrade_info
mysql/ddl_recovery.log
[ec2-user@control rhce-project]$
```

Step 02: Schedule Backups Using Cron

The backup playbook was scheduled using cron on the control node to automate periodic backups of the website and database.

Implementation Overview

- Create a dedicated backup playbook.
- Schedule the backup playbook using cron.
- Verify the cron configuration.

Figure 8.4: Backup playbook (`playbooks/backup.yml`).

```
[ec2-user@control rhce-project]$ vi playbooks/backup.yml
[ec2-user@control rhce-project]$ cat playbooks/backup.yml
---
- hosts: web
  become: yes
  roles:
    - backup

- hosts: database
  become: yes
  roles:
    - backup
[ec2-user@control rhce-project]$
```

Figure 8.5: Verification of the scheduled cron job (`crontab -l`).

```
[ec2-user@control rhce-project]$ crontab -e
no crontab for ec2-user - using an empty one
crontab: installing new crontab
[ec2-user@control rhce-project]$
```

```
[ec2-user@control rhce-project]$ crontab -l
0 2 * * * cd /home/ec2-user/rhce-project && /usr/bin/ansible-playbook playbooks/backup.yml >> /var/log/backup.log 2>&1
[ec2-user@control rhce-project]$
```

This means:

- Every day
- At **2:00 AM**
- Run the backup playbook
- Save the output to `/var/log/backup.log`

Cron jobs were configured to automate the execution of backup tasks at scheduled intervals, ensuring regular backups of the website files and database without manual intervention.

Challenges Faced

During the project implementation, several challenges were encountered:

- Configuring Ansible roles correctly.
- Troubleshooting SSH connectivity between the control node and managed nodes.
- Configuring NFS server and client communication.
- Debugging HAProxy backend connectivity.
- Configuring Prometheus monitoring and Node Exporter.
- Automating backup and cron scheduling.

Learning Outcomes

Through this project, the following skills were gained:

- Infrastructure automation using Ansible.
- Linux server administration.
- Apache web server deployment.
- MariaDB administration.
- NFS configuration.
- Security hardening using SELinux and Firewalld.
- Load balancing using HAProxy.
- Monitoring with Prometheus and Grafana.
- Centralized logging using rsyslog.
- Backup automation and cron scheduling.

Project Summary

The RHCE Capstone Project successfully automated the deployment and management of an enterprise Linux infrastructure using Ansible. Multiple services including Apache, MariaDB, NFS, HAProxy, security hardening, user management, monitoring, and backup automation were configured consistently across all managed nodes.

Conclusion

This RHCE Capstone Project successfully demonstrated the automation and management of a complete Linux server infrastructure using Ansible. The project included web server deployment, database configuration, network file sharing, security hardening, load balancing, monitoring, centralized logging, and automated backup management. The implementation reduced manual effort, improved consistency, and demonstrated the practical application of Red Hat Enterprise Linux administration and infrastructure automation concepts.

GitHub Repository

The complete source code, Ansible playbooks, roles, configuration files, and project documentation are available in the GitHub repository.

Repository:

https://github.com/nandusivadas/RHCE-Capstone-Project.git

References

- Red Hat Enterprise Linux Documentation
- Ansible Documentation
- Apache HTTP Server Documentation
- MariaDB Documentation
- HAProxy Documentation
- Prometheus Documentation
- Grafana Documentation