

PROJECT REPORT
ON
“RHEL Apache Web Server Deployment”

SUBMITTED BY:

Ayush Kumar Thakur

UID- 24MCA20322(5 B)

UNDER THE GUIDANCE OF:

Mrs. GEETANJALI SHARMA

ON ‘OCT, 2024’



University Institute of Computing
Chandigarh University

Mohali, Punjab, 140413

CERTIFICATE

This is to certify that **Ayush Kumar Thakur** (UID: 24MCA20322) has successfully completed the project titled **“RHEL Apache Web Server Deployment”** at **University Institute of Computing** under my supervision and guidance in fulfillment of the requirements of the first semester, **Master of Computer Applications**, at Chandigarh University, Mohali, Punjab.

The project demonstrates the practical implementation of server configuration and deployment using Red Hat Linux.

It highlights the student's proficiency in web hosting and Linux server management acquired during the course.

**DR
ABDULLAH
HOD,(UIC)**

**MS
GEETANJALI
SHRMA
(Project Guide)**

ACKNOWLEDGEMENT

We deem it a pleasure to acknowledge our sense of gratitude to our project guide **MS GEETANJALI SHARMA**, under whose supervision we have carried out the project work. His incisive and objective guidance, along with his timely advice, encouraged us with a constant flow of energy to continue the work.

We wish to reciprocate in full measure the kindness shown by **DR ABDULLAH** (H.O.D, University Institute of Computing), who inspired us with his valuable suggestions in successfully completing the project work.

We shall remain grateful to **DR MANISHA MALHOTRA**, Additional Director, University Institute of Computing, for providing us a strong academic atmosphere by enforcing strict discipline, enabling us to carry out the project work with utmost concentration and dedication.

Finally, we must say that no achievement is ever made without sacrifices at some end, and it is here where we owe our special debt to our parents and friends for showing their generous love and care throughout the entire period of time.

ABSTRACT

The project titled RHEL Apache Web Server Deployment focuses on deploying an Apache web server on Red Hat Enterprise Linux (RHEL). The primary objective is to create a fully functional web server that serves static web content. The process includes the installation and configuration of Apache, followed by the creation of an index.html file, which serves as the default webpage. After configuring the server, the IP address of the machine is retrieved using the `ip addr show` command. By entering this IP into a browser, the server responds by displaying the contents of the index.html file, confirming successful deployment.

This project not only demonstrates core server setup skills but also highlights important concepts such as IP address configuration, web server management, and basic Linux administration. The deployment illustrates a practical use case of hosting a web page on a Linux server, offering valuable insights into the process of managing and securing web servers. This project serves as a foundation for more complex web hosting and server management tasks, emphasizing the reliability and flexibility of Apache on the RHEL platform.

S.NO	TABLE OF CONTENT	PAGE
1	Chapter 1 : Introduction 1.1 Background 1.2 Objectives 1.3 Scope of the Project	6-7
2	Chapter 2 : Implementation 2.1. Environment Setup 2.2. Installing Apache Web Server 2.3. Configure Firewall 2.4. Configuring Apache 2.5. File Permissions and Security 2.6. Creating and Hosting Static Content 2.7. Testing and Verification 2.8. Final Documentation	8-16
3	Chapter 3: Conclusion 3.1 Achievements 3.2 Future Enhancements	17-18
4	Chapter 4: References	19

Chapter 1: Introduction

This project is titled Apache Web Server Deployment on Red Hat Linux. This project deals with installing, configuring, and deploying an Apache web server-one of the most widely used web server platforms available in the world-today. Apache, for example, has long enjoyed popularity as a favorite for web applications and static content hosting due to its robustness, flexibility, and ease of use. This project will deploy a web server on Red Hat Enterprise Linux, hosting a static webpage, index.html, under its IP address. This project basically demonstrates how to deploy a web server but also goes ahead to explore basic practices in server management such as file permissions, security, and checking the functionality of the server. The project demonstrates the most important skills required when dealing with managing a web server in a real world environment by achieving these objectives.

1.1. Background

Web servers are now one of the most important components in the modern internet infrastructure, facilitating web content and applications delivery to people all over the world. Within the numerous options for web server solutions, it is perhaps one of the most leading web servers due to its broad acceptance and flexibility: the Apache HTTP Server. Apache was itself an open- source system that supported static and dynamic forms of content toward different web hosting requirements.

With its enterprise-level support and security features, Red Hat Enterprise Linux is a great thing for stable environments. In this project, I will deploy an Apache web server on RHEL and show how one could set up a secure and efficient system for serving static web pages. Apache and RHEL combine reliability and scalability with a solid foundation upon which I can build for future web hosting tasks.

1.2 Objectives

- Install and configure the Apache HTTP Server on Red Hat Enterprise Linux (RHEL).
- To deploy a static webpage (index.html) and ensure it is served correctly by the Apache server.
- To retrieve and test the server's IP address for accessing hosted content from a web browser.
- To configure appropriate file permissions and security settings to protect the server and its content.
- To provide a practical demonstration of web server deployment, ensuring smooth functionality and reliability.
- To lay the groundwork for future enhancements, including SSL encryption, virtual hosting, and dynamic content hosting.

1.3 Scope of the Project

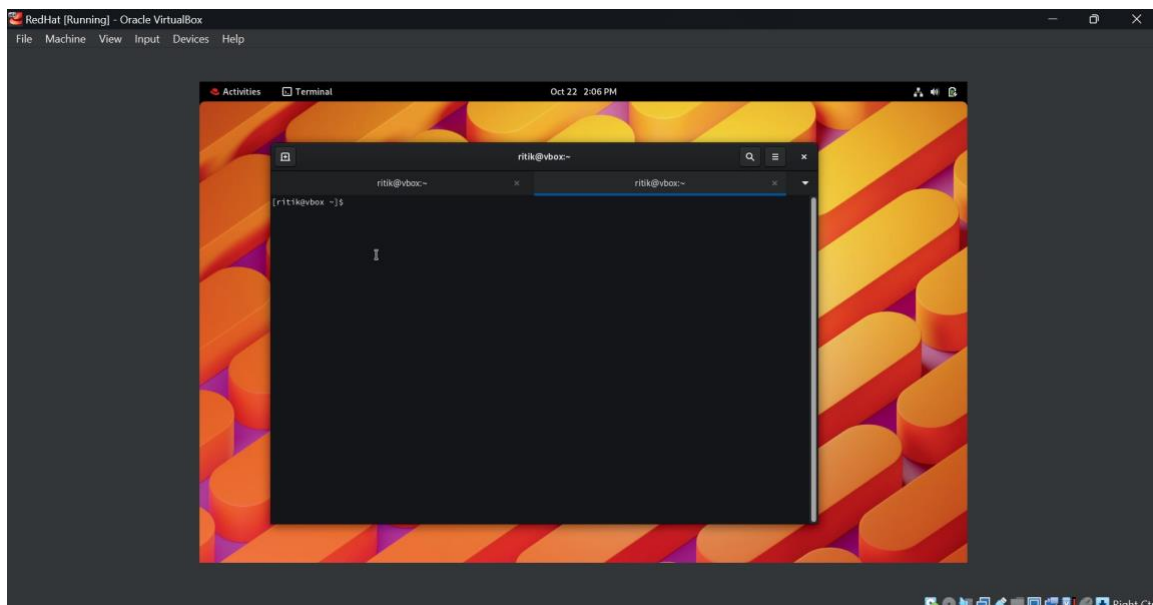
- Install Apache Web Server on a Red Hat Enterprise Linux (RHEL) system.
- Configure Apache to host and serve a static webpage (index.html).
- Ensure File Permissions and Security, preventing unauthorized access.
- Retrieve and Use the Server's IP Address for web page access across different devices.
- Test and Verify Server Functionality by accessing the hosted content through a web browser.
- Foundation for Advanced Configurations, such as SSL encryption and dynamic content hosting in future work.

Chapter 2: Implementation

The implementation phase of this project includes the installation of the Apache web server on Red Hat Enterprise Linux (RHEL) with proper configuration to deliver static web content. In this step-by-step process, it covers the installation of Apache; adjustments needed for some configurations; and ensuring proper permissions on the files, which assure its security. After installing a server, the project also creates a simple static webpage, index.html, hosted by Apache. Implement the final part and test whether the server is configured properly, by accessing the web content with the server's IP in a browser. Demonstrating practical implementation has shown an understanding of server management, from its simplest form, in leaving it as a base for longer tasks in the future.

2.1. Environment Setup

To begin the project, ensure that Red Hat Enterprise Linux (RHEL) is properly installed and updated. Configure the environment by installing any necessary dependencies and tools for managing the Apache web server. This step ensures your system is ready for server deployment.



2.2. Installing Apache Web Server

The next step is to install the Apache HTTP Server. Use the package manager yum or dnf to install Apache on RHEL. After installation, start and enable the Apache service so that it runs on system boot.

```
ritik@vbox:~$ sudo dnf install httpd
[sudo] password for ritik:
Updating Subscription Management repositories.
Last metadata expiration check: 0:02:30 ago on Tuesday 22 October 2024 01:42:51 PM.
Dependencies resolved.
=====
Package                Architecture Version                      Repository                    Size
=====
Installing:
httpd                  x86_64      2.4.57-11.el9_4.1          rhel-9-for-x86_64-appstream-rpms 51 k
Installing dependencies:
apr                    x86_64      1.7.0-12.el9_3             rhel-9-for-x86_64-appstream-rpms 126 k
apr-util              x86_64      1.6.1-23.el9               rhel-9-for-x86_64-appstream-rpms 97 k
apr-util-bdb          x86_64      1.6.1-23.el9               rhel-9-for-x86_64-appstream-rpms 14 k
httpd-core            x86_64      2.4.57-11.el9_4.1          rhel-9-for-x86_64-appstream-rpms 1.5 M
httpd-filesystem      noarch      2.4.57-11.el9_4.1          rhel-9-for-x86_64-appstream-rpms 14 k
httpd-tools           x86_64      2.4.57-11.el9_4.1          rhel-9-for-x86_64-appstream-rpms 86 k
redhat-logos-httpd    noarch      90.4-2.el9                 rhel-9-for-x86_64-appstream-rpms 18 k
Installing weak dependencies:
apr-util-openssl      x86_64      1.6.1-23.el9               rhel-9-for-x86_64-appstream-rpms 17 k
mod_http2             x86_64      2.0.26-2.el9_4             rhel-9-for-x86_64-appstream-rpms 167 k
mod_lua               x86_64      2.4.57-11.el9_4.1          rhel-9-for-x86_64-appstream-rpms 60 k
Transaction Summary
=====
Install 11 Packages

Total download size: 2.2 M
Installed size: 6.0 M
Is this ok [y/N]: y
Downloading Packages:
(1/11): apr-util-openssl-1.6.1-23.el9.x86_64.rpm 7.0 kB/s | 17 kB 00:02
(2/11): apr-util-bdb-1.6.1-23.el9.x86_64.rpm 5.7 kB/s | 14 kB 00:02
(3/11): apr-util-1.6.1-23.el9.x86_64.rpm 39 kB/s | 97 kB 00:02
(4/11): redhat-logos-httpd-90.4-2.el9.noarch.rpm 51 kB/s | 18 kB 00:00
(5/11): mod_http2-2.0.26-2.el9_4.x86_64.rpm 310 kB/s | 167 kB 00:00
(6/11): apr-1.7.0-12.el9_3.x86_64.rpm 104 kB/s | 126 kB 00:00
(7/11): httpd-2.4.57-11.el9_4.1.x86_64.rpm 122 kB/s | 51 kB 00:00
(8/11): httpd-filesystem-2.4.57-11.el9_4.1.noarch.rpm 42 kB/s | 14 kB 00:00
(9/11): httpd-tools-2.4.57-11.el9_4.1.x86_64.rpm 157 kB/s | 86 kB 00:00
(10/11): mod_lua-2.4.57-11.el9_4.1.x86_64.rpm 121 kB/s | 60 kB 00:00
```

```
Complete!
[ritik@vbox ~]$ sudo systemctl start httpd
[ritik@vbox ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
   Active: active (running) since Tue 2024-10-22 13:47:49 IST; 18s ago
     Docs: man:httpd.service(8)
  Main PID: 35289 (httpd)
    Status: "Total requests: 0; Idle/Busy workers 100/0; Requests/sec: 0; Bytes served/sec: 0 B/sec"
    Tasks: 177 (limit: 10966)
   Memory: 22.3M
      CPU: 95ms
    CGroup: /system.slice/httpd.service
            └─35289 /usr/sbin/httpd -DFOREGROUND
              └─35290 /usr/sbin/httpd -DFOREGROUND
                └─35291 /usr/sbin/httpd -DFOREGROUND
                  └─35292 /usr/sbin/httpd -DFOREGROUND
                    └─35293 /usr/sbin/httpd -DFOREGROUND
```

2.3. Configure Firewall

The firewall serves as a critical security feature, controlling network traffic that enters and leaves the server. In this project, it protects the Apache Web Server by selectively allowing specific types of traffic, such as HTTP requests, while blocking unauthorized access. By managing these rules, the firewall ensures that only trusted connections reach the server, reducing the risk of potential attacks.

A key feature of the firewall is the ability to add permanent rules, meaning configuration changes, such as allowing HTTP traffic for Apache, are retained after reboots. This persistence is essential in a web server environment where reliability is a priority. With these rules in place, the Apache server can serve web pages without needing manual adjustments each time the system restarts.

The firewall's flexibility allows for specific customization, such as adding or removing services based on project needs. In this project, only HTTP traffic is enabled to minimize potential vulnerabilities. This control makes the firewall a valuable asset in maintaining security and accessibility for the Apache Web Server, allowing safe connections from other devices on the network.

```
[ritik@vbox ~]$ sudo firewall-cmd --permanent --add--service=httpd
[sudo] password for ritik:
usage: 'firewall-cmd --help' for usage information or see firewall-cmd(1) man page
firewall-cmd: error: unrecognized arguments: --add--service=httpd
[ritik@vbox ~]$ sudo firewall-cmd --reload
success
[ritik@vbox ~]$
```

2.4. Configuring Apache

Once Apache is installed, the configuration files need to be updated. Navigate to the Apache configuration directory (/etc/httpd/conf/httpd.conf) to make necessary changes, such as setting the document root or enabling virtual hosts if required. Adjust the server settings to suit your project needs.

```
ritik@vbox: ~]$ cat /etc/httpd/conf/httpd.conf
#
# This is the main Apache HTTP server configuration file. It contains the
# configuration directives that give the server its instructions.
# See <URL:http://httpd.apache.org/docs/2.4/> for detailed information.
# In particular, see
# <URL:http://httpd.apache.org/docs/2.4/mod/directives.html>
# for a discussion of each configuration directive.
#
# See the httpd.conf(5) man page for more information on this configuration,
# and httpd.service(8) on using and configuring the httpd service.
#
# Do NOT simply read the instructions in here without understanding
# what they do. They're here only as hints or reminders. If you are unsure
# consult the online docs. You have been warned.
#
# Configuration and logfile names: If the filenames you specify for many
# of the server's control files begin with "/" (or "drive:/" for Win32), the
# server will use that explicit path. If the filenames do *not* begin
# with "/", the value of ServerRoot is prepended -- so 'log/access_log'
# with ServerRoot set to '/www' will be interpreted by the
# server as '/www/log/access_log', where as '/log/access_log' will be
# interpreted as '/log/access_log'.
#
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
# Do not add a slash at the end of the directory path. If you point
# ServerRoot at a non-local disk, be sure to specify a local disk on the
# Mutex directive, if file-based mutexes are used. If you wish to share the
# same ServerRoot for multiple httpd daemons, you will need to change at
# least PidFile.
#
ServerRoot "/etc/httpd"
#
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, instead of the default. See also the <VirtualHost>
# directive.
#
```

```
ritik@vbox: ~$ sudo yum install httpd
Installing      : httpd-tools-2.4.57-11.el9_4.1.x86_64      5/11
Running scriptlet: httpd-filesystem-2.4.57-11.el9_4.1.noarch 6/11
Installing      : httpd-filesystem-2.4.57-11.el9_4.1.noarch 6/11
Installing      : httpd-core-2.4.57-11.el9_4.1.x86_64      7/11
Installing      : mod_lua-2.4.57-11.el9_4.1.x86_64         8/11
Installing      : redhat-logos-httpd-90.4-2.el9.noarch     9/11
Installing      : mod_http2-2.0.26-2.el9_4.x86_64         10/11
Installing      : httpd-2.4.57-11.el9_4.1.x86_64          11/11
Running scriptlet: httpd-2.4.57-11.el9_4.1.x86_64          11/11
Verifying      : apr-util-1.6.1-23.el9.x86_64             1/11
Verifying      : apr-util-bdb-1.6.1-23.el9.x86_64         2/11
Verifying      : apr-util-openssl-1.6.1-23.el9.x86_64     3/11
Verifying      : redhat-logos-httpd-90.4-2.el9.noarch     4/11
Verifying      : apr-1.7.0-12.el9_3.x86_64               5/11
Verifying      : mod_http2-2.0.26-2.el9_4.x86_64         6/11
Verifying      : httpd-2.4.57-11.el9_4.1.x86_64          7/11
Verifying      : httpd-core-2.4.57-11.el9_4.1.x86_64      8/11
Verifying      : httpd-filesystem-2.4.57-11.el9_4.1.noarch 9/11
Verifying      : httpd-tools-2.4.57-11.el9_4.1.x86_64    10/11
Verifying      : mod_lua-2.4.57-11.el9_4.1.x86_64        11/11
Installed products updated.

Installed:
apr-1.7.0-12.el9_3.x86_64          apr-util-1.6.1-23.el9.x86_64
apr-util-bdb-1.6.1-23.el9.x86_64  apr-util-openssl-1.6.1-23.el9.x86_64
httpd-2.4.57-11.el9_4.1.x86_64    httpd-core-2.4.57-11.el9_4.1.x86_64
httpd-filesystem-2.4.57-11.el9_4.1.noarch httpd-tools-2.4.57-11.el9_4.1.x86_64
mod_http2-2.0.26-2.el9_4.x86_64  mod_lua-2.4.57-11.el9_4.1.x86_64
redhat-logos-httpd-90.4-2.el9.noarch

Complete!
[ritik@vbox ~]$
```

2.5. File Permissions and Security

File permissions are essential for securing the server. Ensure that only authorized users have the right access to the web server files and directories. Use commands like `chmod` and `chown` to set proper permissions.

```
[ritik@vbox ~]$ sudo chmod -R 755 /var/www/html
[sudo] password for ritik:
[ritik@vbox ~]$
```

2.6. Creating and Hosting Static Content

Create a simple HTML file (`index.html`) that will be served as the default webpage. Write the HTML code in the `/var/www/html` directory, which is Apache's default document root.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Apache Server Info</title>
  <style>
body {
  font-family: Arial, sans-serif;
  background-color: #4f4f4;
  display: flex;
  flex-direction: column;
  min-height: 100vh;
}
```

```
nav {
    background-color: #5424a;
    color: white;
    padding: 15px;
    text-align: center;
}
nav a {
    color: white;
    margin: 0 15px;
    text-decoration: none;
}
.container {
    flex: 1;
    display: flex;
    align-items: center;
    justify-content: center;
    padding: 20px;
}
.content {
    background-color: #ffffff;
    padding: 20px;
    max-width: 800px;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
    text-align: left;
}
footer {
    background-color: #5424a;
    color: white;
    text-align: center;
    padding: 10px;
}
</style>
```

```

</head>
<body>

  <nav>
    <h1>Ritik Rana</h1>
    <p>UID: 24MCA20279</p>
  </nav>

  <div class="container">
    <div class="content">
      <h2>About Apache Server</h2>

      <p>Apache HTTP Server, commonly referred to as Apache, is a
      free and open-source cross-platform web server software. It is one of
      the oldest and most reliable web servers in existence and played a
      pivotal role in the growth of the World Wide Web.</p>

      <p>Apache provides a secure, efficient, and extensible server that
      delivers HTTP services in sync with current HTTP standards. The
      software can be configured to suit a variety of use cases, from serving
      static content to complex web applications.</p>

      <p>Some key features of Apache include:</p>
      <ul>
        <li>Modular architecture</li>
        <li>Support for multiple programming languages</li>
        <li>Comprehensive documentation</li>
        <li>Robust security features</li>
      </ul>
    </div>
  </div>

  <footer>
    <p>© Ritik Rana</p>
  </footer>

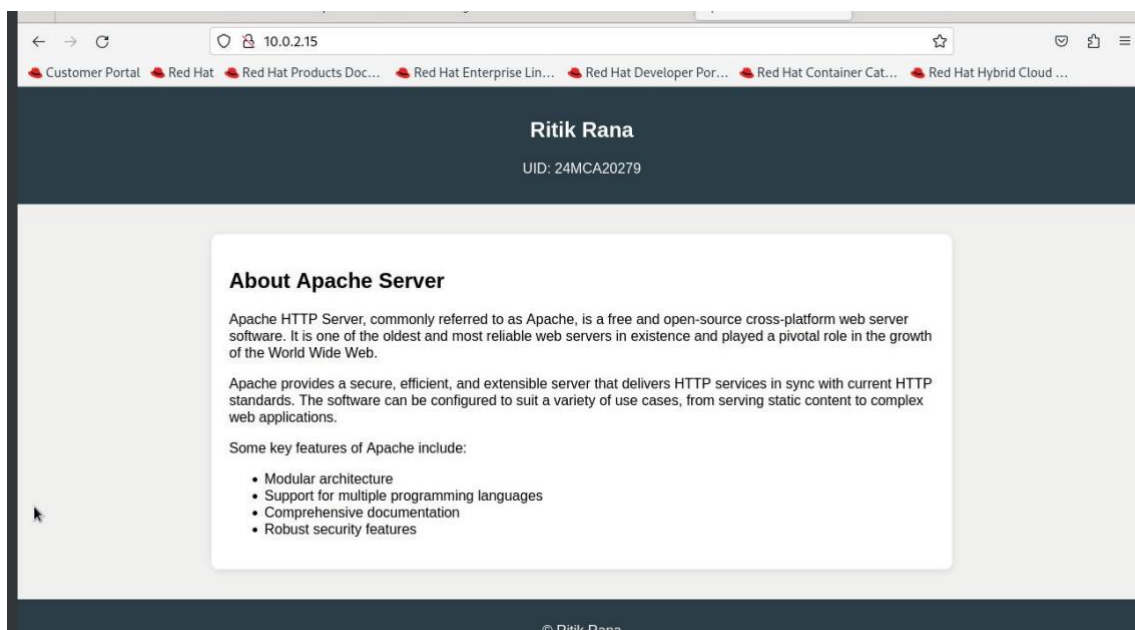
</body>
</html>

```

2.7. Testing and Verification

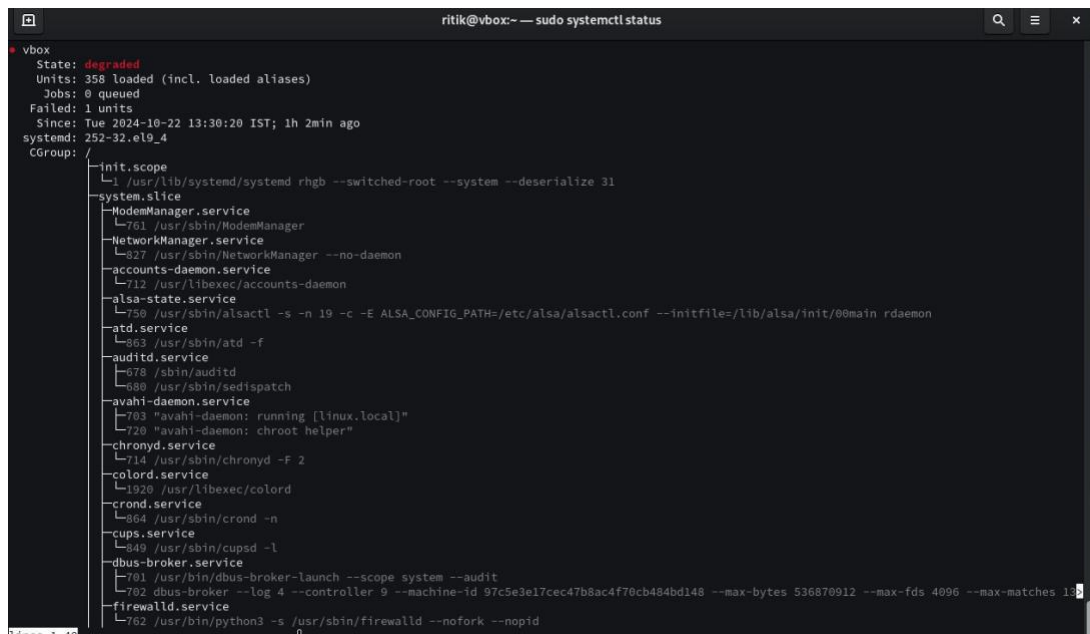
After creating the web content, retrieve the server's IP address using the `ip addr show` command. Open a browser, enter the IP address, and verify that the Apache server is serving the `index.html` page correctly.

```
ritik@vbox:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:40:f8:96 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
        valid_lft 84751sec preferred_lft 84751sec
    inet6 fd00::a00:27ff:fe40:f896/64 scope global dynamic noprefixroute
        valid_lft 85946sec preferred_lft 13946sec
    inet6 fe80::a00:27ff:fe40:f896/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```



2.8. Final Documentation

This project covers setting up an Apache Web Server on Red Hat Linux, allowing HTML files to be hosted and accessed over a network. By installing Apache, configuring the firewall, and adding a basic HTML file, the server is fully prepared for web access. Testing can be done by entering the server's IP address in a browser, verifying that the HTML page loads successfully. This guide provides all necessary commands, configurations, and instructions to set up the Apache Web Server and test its functionality.



```
ritik@vbox:~ — sudo systemctl status vbox
State: degraded
Units: 358 loaded (incl. loaded aliases)
Jobs: 0 queued
Failed: 1 units
Since: Tue 2024-10-22 13:30:20 IST; 1h 2min ago
systemd: 252-32.e19_4
CGroup: /
├─init.scope
│   └─ /usr/lib/systemd/systemd rhgb --switched-root --system --deserialize 31
├─system.slice
│   ├── ModemManager.service
│   │   └─ 761 /usr/sbin/ModemManager
│   ├── NetworkManager.service
│   │   └─ 827 /usr/sbin/NetworkManager --no-daemon
│   ├── accounts-daemon.service
│   │   └─ 712 /usr/libexec/accounts-daemon
│   ├── alsa-state.service
│   │   └─ 750 /usr/sbin/alsactl -s -n 19 -c -E ALSA_CONFIG_PATH=/etc/alsa/alsactl.conf --initfile=/lib/alsa/init/00main rdaemon
│   ├── atd.service
│   │   └─ 993 /usr/sbin/atd -f
│   ├── auditd.service
│   │   ├── 678 /sbin/auditd
│   │   └─ 680 /usr/sbin/auditd
│   ├── avahi-daemon.service
│   │   ├── 703 "avahi-daemon: running [linux.local]"
│   │   └─ 720 "avahi-daemon: chroot helper"
│   ├── chronyd.service
│   │   └─ 714 /usr/sbin/chronyd -F 2
│   ├── colord.service
│   │   └─ 1920 /usr/libexec/colord
│   ├── crond.service
│   │   └─ 864 /usr/sbin/crond -n
│   ├── cups.service
│   │   └─ 849 /usr/sbin/cupsd -l
│   ├── dbus-broker.service
│   │   ├── 701 /usr/bin/dbus-broker-launch --scope system --audit
│   │   ├── 702 dbus-broker --log 4 --controller 9 --machine-id 97c5e3e17cec47b8ac4f70cb484bd148 --max-bytes 536870912 --max-fds 4096 --max-matches 13
│   └─ firewalld.service
│       └─ 762 /usr/bin/python3 -s /usr/sbin/firewalld --nofork --nopid
```


Chapter 3: Conclusion

The implementation phase of this project includes the installation of the Apache web server on Red Hat Enterprise Linux (RHEL) with proper configuration to deliver static web content. In this step-by-step process, it covers the installation of Apache; adjustments needed for some configurations; and ensuring proper permissions on the files, which assure its security. After installing a server, the project also creates a simple static webpage, `index.html`, hosted by Apache. Implement the final part and test whether the server is configured properly, by accessing the web content with the server's IP in a browser. Demonstrating practical implementation has shown an understanding of server management, from its simplest form, in leaving it as a base for longer tasks in the future.

3.1. Achievements

- Successfully installed and configured the Apache web server on Red Hat Enterprise Linux (RHEL) to host a static webpage.
- Created and served a custom `index.html` page, demonstrating proficiency in basic web development and content management.
- Implemented proper file permissions and security measures to protect server files and ensure only authorized access.
- Successfully retrieved the server's IP address and verified its functionality through testing in a web browser, confirming the web server's operational status.
- Identified opportunities for future enhancements, such as implementing SSL/TLS for secure connections and exploring dynamic content capabilities for a more interactive user experience.

3.2 Future Enhancements

- **Implement SSL/TLS:** Setting up HTTPS with SSL/TLS certificates (e.g., via Let's Encrypt) will enhance security by encrypting data in transit, protecting sensitive information exchanged between the server and clients.
- **Dynamic Content Support:** In future iterations, Apache can be configured to serve dynamic content, integrating technologies like PHP, Python, or Node.js to build more complex web applications, improving user interactivity and functionality.
- **Performance Testing:** Conducting stress tests and analyzing the server's response under load will help optimize the configuration for real-world use cases, ensuring that the server can handle varying traffic levels efficiently.

REFERENCES

- ▶ Apache HTTP Server Documentation. Retrieved from <https://httpd.apache.org/docs/>
- ▶ Red Hat Enterprise Linux Documentation. Retrieved from https://access.redhat.com/documentation/enus/red_hat_enterprise_linux/
- ▶ Installing Apache HTTP Server on RHEL. Retrieved from <https://www.tecmint.com/install-apache-on-rhel/>
- ▶ SSL/TLS with Let's Encrypt on Apache. Retrieved from <https://www.digitalocean.com/community/tutorials/how-to-secure-apache-with-letsencrypt-on-red-hat-enterprise-linux-7> Understanding
- ▶ File Permissions in Linux. Retrieved from <https://www.lifewire.com/linux-file-permissions-2201190>