Project : LAMP Stack Setup with User & Permission Management, Monitoring, and Backup on RHEL 9

Project Summary:

In this project, you will build a complete **web server environment** using the LAMP stack (Linux, Apache, MariaDB, PHP). You'll also add key **system administration components**:

1. User & group-based access control
2. Service & resource monitoring
3. Automated backup of critical web content

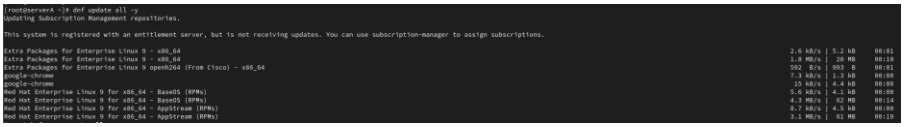
This project simulates a **real-world server management task** for a small company or development team

Tools & Skills Coverd	Tools
OS	RHEL 9
Services	Apache, MariaDB, PHP
Commands:	dnf, chmod, chown, setfacl, tar, crontab, firewall-cmd, systemctl
Monitoring	Cockpit Web Console
Security	SELinux, FirewallD

Step 1: System Preparation

Note: Ensures system is up-to-date with latest security and stability fixes

Commands: dnf update all -y



Step2: User & Group Management

Note: Create group and user ,commands : sudo groupadd dev ,sudo groupadd qa

```
[root@serverA ~]# sudo groupadd dev
groupadd: group 'dev' already exists
[root@serverA ~]# sudo groupadd qa
[root@serverA ~]#
```

Creates two groups: dev for developers and **qa** for testers.

Used later for folder-level access control.

```
[root@serverA ~]# sudo useradd alice -G dev
useradd: user 'alice' already exists
[root@serverA ~]# sudo useradd dhivan -G qa
useradd: user 'dhivan' already exists
[root@serverA ~]# sudo useradd bob -G qa
[root@serverA ~]#
```

Note: Adds users alice and bob and assigns them to dev and qa groups.

-G specifies secondary group.

- Sets a predefined password (**Redhat@123**) for each user.

- --stdin reads the password from standard input (non-interactively).

```
[root@serverA ~]# sudo passwd --stdin alice
Changing password for user alice.
Redhad@123
passwd: all authentication tokens updated successfully.
```

```
[root@serverA ~]# sudo passwd --stdin bob
Changing password for user bob.
Redhat@123
passwd: all authentication tokens updated successfully.
```

STEP 3: Shared Directory with Group Permissions

Note :Creates a shared directory for dev/qa teams - /webshare

```
[root@serverA ~]# mkdir /webshare
[root@serverA ~]#
```

Note: Changes group ownership of /webshare to dev

Commands : " sudo chown root:dev /webshare "

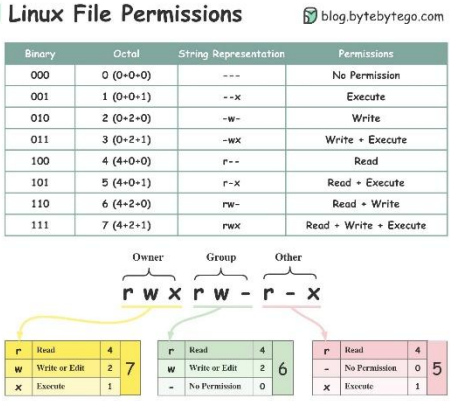
- Changes group ownership of /webshare to **dev**.
- User: **root**, Group: **dev**.

```
[root@serverA ~]# chown root:dev /dev
[root@serverA ~]#
```

- 2 = SetGID bit: files created in this directory inherit the group (dev)
- 775 = rwxrwxr-x — full permissions for owner and group, read/execute for others.

📘 ***Interview Note:*** SetGID is important for collaborative group folders.

```
[root@serverA ~]# chmod 2775 /webshare
[root@serverA ~]#
```



StickyBit :Adds a sticky bit: users can't delete other users' files +t

Commands: chmod +t /webshare

```
[root@serverA ~]# chmod +t /webshare
[root@serverA ~]#
```

STEP 4: Apache Installation and Setup

Note: 1.Installs the Apache HTTP Server (httpd is the package name) Commands: "**dnf install httpd -y**"

2. start and enable these services "httpd" – start –enable httpd

```
[root@serverA ~]# systemctl start httpd
[root@serverA ~]# systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
```

➔ **Firewall Configuration**: **Note**: Opens HTTP port (80) permanently in FirewallD**.**

What is SElinux : SELinux (Security-Enhanced Linux) is a mandatory access control (MAC) system that enforces strict rules on what processes can access which files and resources — even if they're running as root.

It helps prevent privilege escalation and zero-day attacks by confining services**.**

	Commands	Mode	Current Mode
Check SELinux Status	getenforce	Enforcing	Policies are applied
		Permissive	Only logs violations.
		Disabled	SELinux is off

```
[root@serverA ~]# firewall-cmd --permanent --add-service=http
success
[root@serverA ~]# firewall-cmd --reload
success
[root@serverA ~]#
```

Note : Opens HTTP port (80) permanently in FirewallD and –reload for applied changes

STEP 5: MariaDB Installation and Securing

Note : Install a mariya DB Server by using DNF package format : **Commands** : "**dnf install mariadb-server -y**" and start,enable

```
Installed:
  mariadb-3:10.5.27-1.el9_5.x86_64          mariadb-backup-3:10.5.27-1.el9_5.x86_64          mariadb-common-3:10.5.27-1.el9_5.x86_64    mariadb-connector-c-3.2.6-1.el9_0.x86_64          mariadb-connector-c-config-3.2.6-1.el9_0.noarch
  mariadb-errmsg-3:10.5.27-1.el9_5.x86_64   mariadb-gssapi-server-3:10.5.27-1.el9_5.x86_64   mariadb-server-3:10.5.27-1.el9_5.x86_64    mariadb-server-utils-3:10.5.27-1.el9_5.x86_64     mysql-selinux-1.0.13-1.el9_5.noarch
  perl-DBD-MariaDB-1.21-16.el9_0.x86_64     perl-File-Copy-2.34-481.el9.noarch               perl-Sys-Hostname-1.23-481.el9.x86_64
```

1.Securing " mysql_secure_installation"

Purpose : This command is used to secure your MariaDB/MySQL server after installation.When you install MariaDB/MySQL, it's not secure by default — this tool helps you configure basic security settings interactively.

Its will promoted by several steps below here :

1.Setting root password **2.**Removing anonymous users **3**. Disabling remote root login **4.** Removing test DB **5**. Reloading privileges

```
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!


By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
 ... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
[root@serverA ~]#
```

Sample Output below here :

Securing the MySQL server deployment.

1.Enter current password for root (enter for none):
Set root password? [Y/n] Y
New password:
Re-enter new password:
2.Remove anonymous users? [Y/n] Y
3.Disallow root login remotely? [Y/n] Y
4.Remove test database and access to it? [Y/n] Y
5.Reload privilege tables now? [Y/n] Y

STEP 6: PHP Installation and Test Page

Note:" **dnf install php php-mysqlnd php-cli php-common -y** "

Package	Purpose	Interview Point of view
php	Installs the core PHP interpreter	This command installs PHP and essential PHP modules that allow it to work with Apache and MariaDB (MySQL). It prepares your system to run dynamic websites like WordPress, or your own PHP apps.
php-mysqlnd	Enables PHP to connect to MySQL/MariaDB databases using **MySQL Native Driver (nd)**	
php-cli	Allows you to run PHP scripts from the **command line**	

php-common	Installs shared files and core PHP libraries (required by all other modules)	

Sample Output :

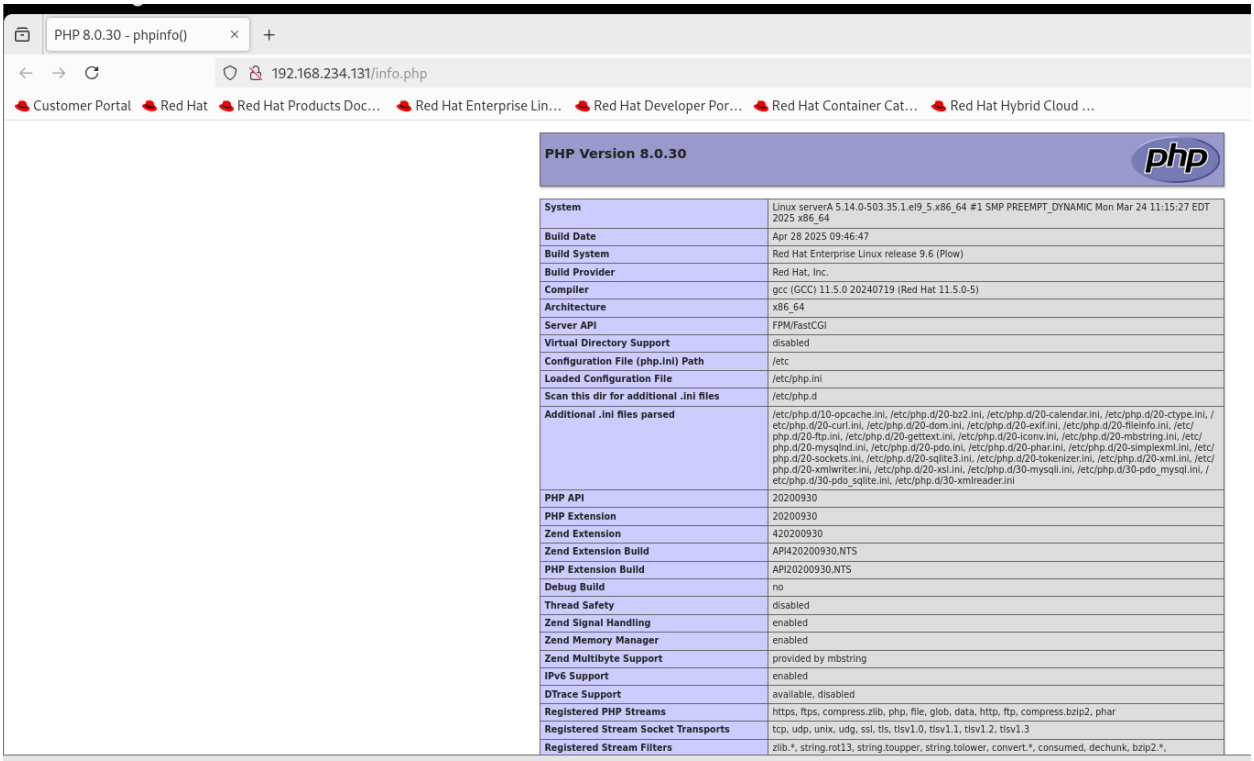
```
Installed:
  nginx-filesystem-2:1.20.1-22.el9_6.2.noarch      php-8.0.30-3.el9_6.x86_64          php-cli-8.0.30-3.el9_6.x86_64      php-common-8.0.30-3.el9_6.x86_64      php-fpm-8.0.30-3.el9_6.x86_64      php-mbstring-8.0.30-3.el9_6.x86_64
  php-mysqlnd-8.0.30-3.el9_6.x86_64                php-opcache-8.0.30-3.el9_6.x86_64  php-pdo-8.0.30-3.el9_6.x86_64      php-xml-8.0.30-3.el9_6.x86_64
```

After Installation - Test PHP :

1. **Create a PHP test file**
   "nano /var/www/html/info.php" > instead a some php code >

```
[root@serverA ~]# cat /var/www/html/info.php
<?php
phpinfo();
?>
[root@serverA ~]#
```

Save and open| **http://<your-server-ip>/info.php** | in your browser
Note : Good to test LAMP stack is integrated correctly.



System	Linux serverA 5.14.0-503.35.1.el9_5.x86_64 #1 SMP PREEMPT_DYNAMIC Mon Mar 24 11:15:27 EDT 2025 x86_64
Build Date	Apr 28 2025 09:46:47
Build System	Red Hat Enterprise Linux release 9.6 (Plow)
Build Provider	Red Hat, Inc.
Compiler	gcc (GCC) 11.5.0 20240719 (Red Hat 11.5.0-5)
Architecture	x86_64
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/10-opcache.ini, /etc/php.d/20-bz2.ini, /etc/php.d/20-calendar.ini, /etc/php.d/20-ctype.ini, /etc/php.d/20-curl.ini, /etc/php.d/20-dom.ini, /etc/php.d/20-exif.ini, /etc/php.d/20-fileinfo.ini, /etc/php.d/20-ftp.ini, /etc/php.d/20-gettext.ini, /etc/php.d/20-iconv.ini, /etc/php.d/20-mbstring.ini, /etc/php.d/20-mysqlnd.ini, /etc/php.d/20-pdo.ini, /etc/php.d/20-phar.ini, /etc/php.d/20-simplexml.ini, /etc/php.d/20-sockets.ini, /etc/php.d/20-sqlite3.ini, /etc/php.d/20-tokenizer.ini, /etc/php.d/20-xml.ini, /etc/php.d/20-xmlwriter.ini, /etc/php.d/20-xsl.ini, /etc/php.d/30-mysqli.ini, /etc/php.d/30-pdo_mysql.ini, /etc/php.d/30-pdo_sqlite.ini, /etc/php.d/30-xmlreader.ini
PHP API	20200930
PHP Extension	20200930
Zend Extension	420200930
Zend Extension Build	API420200930,NTS
PHP Extension Build	API20200930,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, compress.bzip2, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, convert.*, consumed, dechunk, bzip2.*,

STEP 7: Cockpit for Monitoring

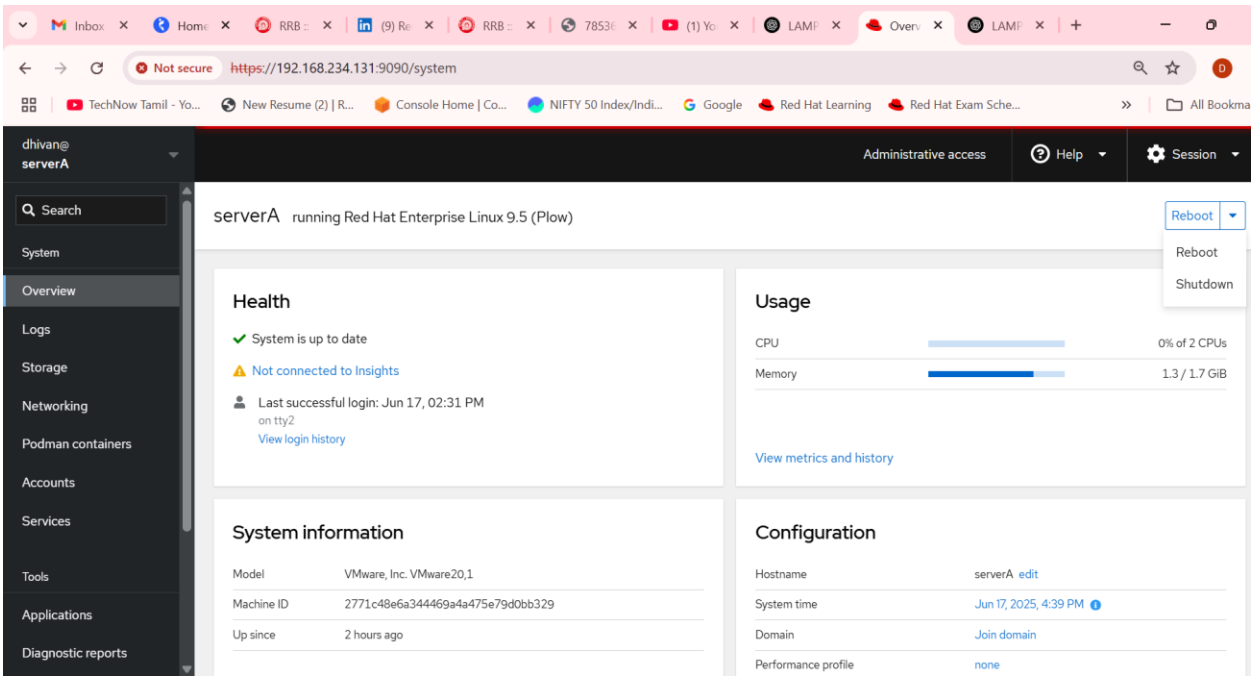
Note : install cockpit package using dnf " dnf install cockpit -y "

```
Upgraded:
  cockpit-334.1-1.el9_6.x86_64
```

1.start and enable these services after add a firewall

```
[root@serverA ~]# firewall-cmd --permanent --add-service=cockpit
Warning: ALREADY_ENABLED: cockpit
success
[root@serverA ~]# firewall-cmd --reload
success
[root@serverA ~]#
```

Redirect to your local browser " https://192.166.234.131"9090"



Why Use Cockpit in Your Project? 1. Real-time system monitoring (CPU, memory, disk, network) 2. Easier troubleshooting without command-line 3. Secure web interface (uses HTTPS) 4. Great for beginners transitioning from GUI to CLI

STEP 8: Backup Script and Automation

In this step, you'll create a script to automatically back up your Apache website content (/var/www/html/) every day. Backups are stored as .tar.gz files with the current date.

1.create dir for backup : mkdir /backup

```
[root@serverA ~]# mkdir /LAMPBackup
[root@serverA ~]#
```

2.Create script name on "backup_web.sh " wigh time and date

```
[root@serverA ~]# echo '#!/bin/bash
> tar -czf /backup/web_$(date +%F).tar.gz /var/www/html/' | sudo tee /usr/local/bin/backup_web.sh
#!/bin/bash
tar -czf /backup/web_$(date +%F).tar.gz /var/www/html/
[root@serverA ~]# ^C
```

Explanation:

- #!/bin/bash = tells Linux this is a shell script
- tar -czf = command to compress files into .tar.gz
- /backup/web_$(date +%F).tar.gz = backup file name with today's date
- /var/www/html/ = your website folder to back up
- tee = saves this script as a file at /usr/local/bin/backup_web.sh

3.Make the Script Executable : you need to given permission to run script "chmod +x /usr/local/bin/backup_web.sh "

```
[root@serverA ~]# chmod +x /usr/local/bin/backup_web.sh
[root@serverA ~]#
```

4. Create the Backup Script

```
                                                    root@serverA:~
  GNU nano 5.6.1
#!/bin/bash
tar -czf /backup/web_$(date +%F).tar.gz /var/www/html/
```

Above commands are Explanation Below here

Line	Command / Code	Purpose
#!/bin/bash	Shebang line	Tells the OS to run this script using Bash shell.
BACKUP_DIR="/backup"	Variable	Path to backup folder.
SOURCE_DIR="/var/www/html"	Variable	Path of your website data.
DATE=$(date +%F)	Variable	Gets current date (e.g., 2025-06-10) in YYYY-MM-DD format.
BACKUP_FILE="$BACKUP_DIR/web_$DATE.tar.gz"	Variable	Backup filename with date.
tar -czf "$BACKUP_FILE" "$SOURCE_DIR"	tar command	Compresses the website folder into a .tar.gz file.

5.Make the Script Executable:

"chmod +x /usr/local/bin/backup_web.sh "

```
[root@serverA bin]# chmod +x /usr/local/bin/backup_web.sh
[root@serverA bin]#
```

Now you can run it directly like to " /usr/local/bin/backup_web.sh " if you correctly following the above commands see like that BELOW output : Date and file format

```
[root@serverA bin]# /usr/local/bin/backup_web.sh
tar: Removing leading `/' from member names
[root@serverA bin]# ls /backup
web_2025-06-17.tar.gz
[root@serverA bin]#
```

Step9: Schedule Daily Cron Job

What is Crontab: Crontab is a time-based job scheduler in Unix-like operating systems

Crontab Syntax:

```
┌──────────── minute (0 - 59)
│ ┌────────── hour (0 - 23)
│ │ ┌──────── day of month (1 - 31)
│ │ │ ┌────── month (1 - 12)
│ │ │ │ ┌──── day of week (0 - 7)
│ │ │ │ │       (0 or 7 is Sunday)
│ │ │ │ │
│ │ │ │ │
0  2 * * * root /usr/local/bin/backup_web.sh
```

Note : Adds a **cron job** to /etc/crontab, telling the system to run the backup script every day at **2:00 AM**.

```
0 2 * * * /usr/bin/local/backup_web.sh
```

Step 9 : Final Check List

Step	Description	Status
🔹 1	**Update & Upgrade RHEL 9 (sudo dnf update -y)**	✅
🔹 2	**Install Apache Web Server (sudo dnf install httpd -y)**	✅
🔹 3	**Start and Enable Apache (sudo systemctl enable --now httpd)**	✅
🔹 4	**Install MariaDB Server (sudo dnf install mariadb-server -y)**	✅
🔹 5	**Start and Secure MariaDB (sudo systemctl start mariadb, sudo mysql_secure_installation)**	✅

Step	Description	Status
🔹 6	Install PHP + MySQL Modules (sudo dnf install php php-mysqlnd php-cli php-common -y)	✅
🔹 7	Create PHP Test Page (`echo ""	sudo tee /var/www/html/info.php`)
🔹 8	Verify PHP Page in Browser (http://server-ip/info.php)	✅
🔹 9	Configure Firewall (sudo firewall-cmd --permanent --add-service=http && sudo firewall-cmd --reload)	✅
🔹 10	Create Backup Directory (sudo mkdir -p /backup)	✅
🔹 11	Write Backup Script (/usr/local/bin/backup_web.sh)	✅
🔹 12	Make Script Executable (sudo chmod +x /usr/local/bin/backup_web.sh)	✅
🔹 13	Test Script Output (sudo /usr/local/bin/backup_web.sh, then ls /backup)	✅
🔹 14	Add to Crontab for Automation (sudo crontab -e)	✅