Author:-Dinesh.C.Patel

(RHCSA,RHCE,AWS,DO188 & HashiCorp Terraform Certified)[8766924468]

Email:-dineshcpatel2727@gmail.com

==========

RHEL 10 Version – Updated Topics

- RHEL 10 installation
- *Red Hat Lightspeed (AI-assisted automation)*
- Flatpak & AppStream updates
- Image Builder
- Stratis & LVM improvements

➢What is Red Hat Lightspeed – AI-assisted automation?

Red Hat Lightspeed is an **AI-powered assistant** built into the Red Hat ecosystem (RHEL + Ansible) that helps system administrators, DevOps engineers, and developers **automate faster and smarter.**

- I**t is essentially AI + automation integration:**

- Uses **Generative AI models** trained on **Red Hat knowledge base, Ansible playbooks, and best practices.**
- Helps **write, optimize, and explain Ansible playbooks.**
- Suggests **automation workflows** for common system admin tasks (user management, security hardening, networking, containers, etc.).

- Reduces the complexity of automation for **beginners** while improving **productivity** for experts.

➢Key Benefits of Red Hat Lightspeed in RHEL 10

- **AI-assisted Playbook Creation**

  1) Suggests Ansible playbooks in real time.

  2) Example: If you want to configure Apache, Lightspeed generates a working Ansible role/playbook snippet instantly.

- **Error Detection & Best Practices**

  1) Detects mistakes in playbooks.

  2) Suggests corrections aligned with Red Hat best practices**.**

- **Explain ability (Learning)**

  1) Explains each line of a playbook in simple language → great for students & new admins.

- **Speed & Efficiency**

  1) Cuts down manual writing.

  2) Faster automation deployment for large infrastructures.

  3) Works across RHEL, OpenShift, Ansible Automation Platform.

➢Enabling the Command-Line AI Assistant on RHEL 10

 # How to install lightspeed?

dnf install command-line-assistant -y

Examples:

c "your question or prompt"

 # Attach a File

- c -a <filename> "your question"

Pipe Output into the Assistant

• **command | c "question"** You can pipe the output of any command directly into the assistant.

• Example: **cat <logfile> | c** or
cat <logfile> | c "how do I solve this?"

Interactive Mode

• **c -i**
Starts an interactive session, allowing you to ask multiple follow-up questions in a conversational manner.

Conversation History

- c history --all — Show all past interactions.
- c history --first — Display the very first conversation.
- c history --last — Show the most recent conversation.
- c history --filter "term" — Filter history by a specific keyword (e.g. "podman").
- c history --clear — Clear your conversation history entirely**.**

Terminal Capture of Previous Commands

• **Enable capture**: This tells Lightspeed to start monitoring your terminal session. c shell --enable-capture — Start capturing your terminal session.

- **Reference previous commands**:
  c -w 1 "your question" — Refers to output from the most recent command. [-w =window] c -w 2 "question" — Refers to the second most recent, and so on.

• **Stop capture**: Press Ctrl + D in your terminal.

c "What is SHELL?"

c "Help me figure out why this system is slow to boot"

c "how do I find all the files in the /etc that have been modified in the last hour"

c "how to troubleshoot sshd failing to start"

c "how to configure dhcp server?"

#c "How do I find the largest files in the root filesystem which are not owned by a package"

c "How do I open port 8080 in the firewall?"

c "How can I check which process is using a particular port?"

c "Create a 500MB swap partition and active it"

REDIRECTING A COMMAND OUTPUT TO THE COMMAND-LINE ASSISTANT

Use the log file that contains information that you want to understand by redirecting that log file output to command-line assistant powered by RHEL Lightspeed

 # cat error.log | c "how do I solve this?"

ENABLING THE COMMAND-LINE ASSISTANT TO CAPTURE YOUR TERMINAL ACTIVITY

Enable the terminal capture for your current terminal session

c shell --enable-capture (enable capturing)

Run some commands

ls -l /etc

df -h

ps -aux | grep sshd

c -w 1 "what does this process list mean?" [-w = window]

c -w 2 "is any filesystem almost full?"

c -w 3 "what are the permissions here?"

Stop capture: Press Ctrl + D in your terminal.

- **USING THE COMMAND-LINE ASSISTANT TO DEBUG OR TROUBLESHOOT SYSTEM ISSUES**

c “how to troubleshoot network errors”

c “I cannot access my server with SSH. Can you give me a list of things to troubleshoot?”

$ c “I need to boot into a different kernel”

$ c “how to troubleshoot SSHD failing to start”

$ c "I am failing to start sshd process"

c -a /root/error.log "Help me analyze this log"

- **USING THE COMMAND-LINE ASSISTANT TO TROUBLESHOOT SSHD SERVICE FAILING TO START**

 # systemctl restart sshd [give the error]

 # systemctl status sshd

[service fail]

1. Run that journalctl command. Add the tail command to get the last 30 lines, pipe that output into the command-line assistant, and add a query to understand the error.

journalctl -xeu sshd.service | tail -n 30 | c “here are the logs, please help me understand this”

***Give the error**: there's an error on line 21, where it mentions "Bad configuration option: Porrt".*

2. Ask the command-line assistant to generate a command on how to fix this typing error

c “what is the command that I can use to change “Porrt ”to “Port” in the /etc/ssh/sshd_config file?

3. Use the command suggested by the command-line assistant.

sed -i s/Porrt/Port/g /etc/ssh/sshd_config

systemctl restart sshd

systemctl status sshd

Bingooo……

DONE

- **20 TGPT Prompts for Linux Tasks**

1. List and explain all the files and directories under /etc that are related to system configuration.

2. Write a command to find and delete all `.log` files older than 7 days in `/var/log`.

3. Show how to check which services are enabled and disabled at boot in systemd.

4. Give me a step-by-step guide to create a new user with sudo privileges in Linux.

5. Explain the difference between hard links and soft links with examples.

6. Write a one-liner to find all running processes that contain the word 'nginx'.

7. How do I set a static IP address in RHEL-based and Debian-based distributions?

8. Explain step-by-step how to create, mount, and make persistent a new partition in Linux.

9. Write commands to compress a directory with tar and extract it again.

10. How to schedule a cron job that backs up `/home` every night at 2 AM?

11. What is SELinux? Show commands to check its current status and temporarily disable it.

12. Explain the difference between `/bin`, `/sbin`, `/usr/bin`, and `/usr/sbin`.

13. Write a shell script that monitors CPU usage and sends an alert if usage is above 80%.

14. Show me how to configure password expiration policy for all users.

15. Write commands to list all open network ports and the processes using them.

16. How to permanently set environment variables in Linux for all users?

17. Explain step-by-step how to create and manage LVM volumes in Linux.

18. Write a command to recursively change ownership of `/opt/app` to user `deploy`.

19. What are runlevels in Linux? Show how to change the default target in systemd.

20. Explain the boot process in Linux step by step.

➢ Creative TGPT Linux Tasks

1. Shell Script Creation

Write a shell script that checks disk usage of `/home` and emails admin if usage is above 80%.

2. Automation with Cron

Create a cron job entry that runs a backup script every day at 11 PM and logs output to

/var/log/backup.log.

3. Systemd Service File

Write a systemd service file to run a Python script located at /opt/app/main.py on startup.

4. Firewall Rules

Generate firewall-cmd commands to allow only SSH, HTTP, and HTTPS, and block everything else in RHEL9.

5. Dockerfile Generation

Write a Dockerfile to run a simple Nginx web server that serves files from /usr/share/nginx/html.

6. Ansible Playbook

Write an Ansible playbook to install Apache, start the service, and deploy a custom index.html page.

7. Log Analyzer Script

Write a bash script that parses /var/log/secure for failed SSH login attempts and lists IPs with more than 5 failures.

8. Monitoring Script

Create a shell script that monitors CPU and memory usage every 5 seconds and saves it in

/tmp/sysreport.txt.

9. File Organizer

Write a bash script that moves all .jpg and .png files from ~/Downloads into ~/Pictures/ and organizes them by date.

10. Custom MOTD

Write a script that updates the Linux MOTD to display hostname, IP address, uptime, and number of logged-in users.

➢ Flatpak Setup, Example & Testing in RHEL 10

What Flatpak RHEL 10?

Flatpak is a **Package System** for Linux that delivers applications in a **sandbox.**

Each app comes with all the libraries it needs → no "dependency hell".

Works across many Linux distributions (RHEL, Fedora, Ubuntu, Debian, etc.).

Mainly used for **desktop applications** like browsers (Firefox, Chrome), editors (VS Code, GIMP), IDEs, etc.

- **DNF = System-level package manager (OS + services).**

- Flatpak = User-level app manager (sandboxed desktop apps).

- Flatpak is NOT designed for server-side services like httpd, mariadb, dhcpd, nfs, etc.

- Flatpak is focused on desktop GUI applications (e.g., Firefox, GIMP, LibreOffice, VS Code)

Feature	YUM	DNF (New YUM)	Flatpak
Scope	**RPM package manager**	**RPM package manager**	**Universal app runtime**
Repos	**RHEL/CentOS repos**	**RHEL/CentOS repos**	**Flathub / Flatpak repos**
Dependency mgmt	**Slower, older**	**Faster, better**	**Bundled inside app (sandboxed)**
Sandboxing	**No**	**No**	**Yes (isolated apps)**
Use case	**System packages**	**System packages**	**Desktop apps, universal installs**

- **Summary**

✓ Installed Flatpak on RHEL 10

✓ Added Flathub repository

✓ Installed and tested Firefox, GIMP and VScode ✓ Verified apps using flatpak list and flatpak run

1. Install Flatpak

First, install Flatpak package:

dnf install flatpak -y

Check version:

flatpak --version

2. Add Remote Repositories

(a) Default RHEL AppStream Remote - Check existing remotes:
flatpak remotes

(b) Add Flathub Repository
flatpak remote-add --if-not-exists flathub https://flathub.org/repo/flathub.flatpakrepo

Verify:
flatpak remotes

3. Search for an Application

Example: Search Firefox application
flatpak search firefox

4. Install an Application

Install Firefox from Flathub:
flatpak install flathub org.mozilla.firefox -y

5. Run the Application

Run Firefox:
flatpak run org.mozilla.firefox

6. Verify Installation

List installed Flatpak apps:
flatpak list --app

Example output:

Name	Application ID	Version	Branch	Installation
Firefox	org.mozilla.firefox	129.0	stable	system

7. Update & Maintenance

Update all apps:

flatpak update -y

Remove an app:

flatpak uninstall org.mozilla.firefox -y [-y = yes]

Remove unused runtimes:

flatpak uninstall --unused -y

8. Testing with Another App (GIMP Example)

flatpak search gimp [to search app]

flatpak install flathub org.gimp.GIMP -y [to install app] (-y = yes)

flatpak run org.gimp.GIMP [to run app]

flatpak info org.gimp.GIMP [to show info about app]

flatpak info --show-location org.gimp.GIMP [to show install location]

flatpak update org.gimp.GIMP [to update app]

flatpak uninstall org.gimp.GIMP -y [to uninstall app]

Testing Another Example: Install VS Code

flatpak search code

sudo flatpak install flathub com.visualstudio.code -y

flatpak run com.visualstudio.code

flatpak info com.visualstudio.code

Extra Command:

sudo flatpak update <appID> [Update a specific app]

flatpak run <appID> --version [Run app with arguments]

flatpak info --files <appID> [List all installed files for an app]

flatpak remote-ls [source list]

flatpak uninstall --all [to remove all apps]

flatpak uninstall --unused [Remove unused runtimes]

[NOTE: to store all app in /var/lib/flatpack/app directory]

==========

➢ What is Image Builder in RHEL 10?

Image Builder is a tool in RHEL that allows you to create **customized operating system images** for different platforms (cloud, virtualization, bare metal, edge devices).

- It uses Blueprints (TOML files) where you define which packages, configurations, and users you want.
- It can build images in many formats:

1) Cloud images → AWS AMI, Azure VHD, Google GCE image
2) VM images → qcow2, VMDK, VDI
3) Edge images → OS Tree-based images for IoT/edge devices

4) ISO & installer images → Bootable media

➢Key Improvements in RHEL 10 Image Builder

1. Cloud Profiles
   - Ready-made templates for AWS, Azure, GCP, OpenStack.
   - Automatically configures cloud-init, drivers, and agents.
   - Supports hybrid/multi-cloud deployment faster.
2. Edge Profiles
   - Creates OSTree-based minimal images for IoT & edge devices.
   - Supports rpm-ostree updates (atomic, rollback-capable updates).
   - Improved security hardening for disconnected environments.
   - Useful for industries like telecom, retail, manufacturing where devices are remote.
3. Web Console Integration (Cockpit)
   - Image Builder is integrated into Cockpit → GUI-based workflow.
   - Easier for admins to build and download images directly.

How to install and Create custom Image

dnf install osbuild-composer composer-cli cockpit-composer bash-completion -y

systemctl enable --now osbuild-composer.socket

systemctl enable --now cockpit.socket

vim myserver.toml

```
name = "myserver"
description = "RHEL 10 custom server image with extra packages"
version = "0.0.1"

# Add base packages
[[packages]]
name = "httpd"
version = "*"

[[packages]]
name = "vim"
version = "*"

[[packages]]
name = "wget"
version = "*"

[[packages]]
name = "curl"
version = "*"

[[packages]]
name = "git"
version = "*"

[[packages]]
name = "podman"
```

```
version = "*"

[[packages]]
name = "bash-completion"
version = "*"

# Example: add development tools
[[packages]]
name = "gcc"
version = "*"

[[packages]]
name = "make"
version = "*"

# Customize users
[[customizations.user]]
name = "test-vm"
description = "super user"
password = "$6$z3Pr7hMAbc$Z1aIVZ9..."
groups = ["wheel"]

# Example: set timezone
[customizations.timezone]
timezone = "Asia/Kolkata"

# Example: set hostname
```

[customizations]
hostname = "myserver.example.com"

:wq!

composer-cli blueprints push myserver.toml [Push it to composer]
composer-cli blueprints list [List available blueprints]
composer-cli compose start myserver qcow2 [Build an Image]
composer-cli compose status [Check status]
composer-cli compose image [Once complete, download]
#**composer-cli compose types** [Check all available]
composer-cli compose start myserver ami [Create Cloud Image for AWS]
composer-cli blueprints show myserver [to show blueprint info]
composer-cli compose delete [Remove Old Builds]
composer-cli compose image d02e46b9-4e38-4053-b53c-822686bf9199 --filename /root/node1-rhcsa-rhel9.qcow2

Supported Image Types

Some formats available in RHEL 10:

- qcow2 → KVM/VirtualBox
- ami → Amazon EC2
- vhd → Azure
- vmdk → VMware

- live-installer-iso → Custom bootable installer
- edge-commit / edge-container → Edge devices

➢ **Create encrypt password**

```
# openssl passwd -6
```

More Information: https://osbuild.org/docs/user-guide/blueprint-reference/

➢What is Stratis in RHEL 10?

- Stratis is a **local storage management tool** introduced by **Red Hat** to simplify advanced storage tasks.
- It provides a **simple, modern interface** for managing storage pools, snapshots, and file systems.
- The goal is to **make storage management as easy as possible** without replacing LVM entirely.

➢ Improvements in RHEL 10 Stratis

- **Storage pools** (like volume groups in LVM)
- **File systems** (like logical volumes, automatically formatted with XFS)
- **Thin provisioning** (only uses space when data is actually written)
- **Snapshots** (instant copy of a filesystem state)
- **Integration with systemd** (pools/filesystems are persistent across reboot)

➢ Stratis vs LVM in RHEL 10

Feature	LVM + XFS (RHEL 10)	Stratis (RHEL 10)
Complexity	Multiple commands (pvcreate, vgcreate, lvcreate, mkfs.xfs)	Single stratis command
Thin Provisioning	Requires manual setup	Built-in
Snapshots	Manual + slower	Instant and easy
Monitoring	Use lvs, vgs, xfs_info separately	Use stratis report
Ease of Use	Steep learning curve	Beginner-friendly

IMP NOTE: The **combined capacity** of all block devices (/dev/sda, /dev/sdb, etc.) in the pool adds up to 1 TB.

Stratis pools are **thin-provisioned**, meaning you can create filesystems that appear bigger than the physical size, but the actual data usage grows as you write.

➢ Installing Stratis in RHEL 10

dnf install stratisd stratis-cli -y
systemctl enable --now stratisd

This installs and starts the **Stratis daemon** (stratisd) which manages storage.

➢ Basic Workflow (Step-by-Step)

1. Create a Storage Pool

stratis pool create mypool /dev/sda

- mypool → name of your storage pool
- /dev/sda → disk or partition to use

You can add more disks later:

stratis pool add-data mypool /dev/sdb

---

2. Create a Filesystem

stratis fs create mypool myfs

- Creates a thin-provisioned XFS filesystem
- Appears automatically under /stratis/mypool/myfs

---

3. Mount the Filesystem

mkdir /mnt/mydata
mount /dev/stratis/mypool/myfs /mnt/mydata

To make it permanent:

echo "/dev/stratis/mypool/myfs /mnt/mydata xfs defaults 0 0" | sudo tee -a /etc/fstab

---

4. Create a Snapshot

stratis fs snapshot mypool myfs myfs-snap1

- Creates a snapshot filesystem
- Appears under /stratis/mypool/myfs-snap1

You can roll back or mount it for recovery.

---

5. Monitor & Report

stratis pool list

stratis fs list

stratis report

➢ Example Lab Scenario for Practice

You can use this in your RHEL 10 lab setup:

1. Attach two extra virtual disks (/dev/sda, /dev/sdb).
2. Create a Stratis pool skynetpool using both disks.
3. Create a filesystem projectfs and mount it under /data.
4. Copy files into /data (fill up some space).
5. Create a snapshot projectfs-snap1.
6. Delete some files, then mount snapshot to verify recovery.

✓Stratis Lab – Step-by-Step Practical

1. Attach Two Extra Virtual Disks

Environment: RHEL 10 VM (KVM, VirtualBox, VMware, Proxmox — any works)

• Add two disks (for example 2 GB each).

• Verify they are visible:

lsblk

You should see something like:

```
sda 40G disk
 ├─sda1   1G part   /boot
 └─sda2   39G part   /
sdb         2G  disk
sdc         2G  disk
```

---

2. Install & Start Stratis

dnf install -y stratisd stratis-cli

systemctl enable --now stratisd

Check service status:

systemctl status stratisd

---

3 Create Stratis Pool

Create a pool named **demopoo**l with both disks:

stratis pool create **demopoo**l /dev/sda /dev/sdb

Verify pool:

stratis pool list

---

4 Create Filesystem in Pool

stratis fs create demopool projectfs

List filesystems:

stratis fs list

You'll see:

Pool	Name	Name Used	Created
demopool	projectfs	546 MiB	2025-09-24 10:32:45

---

5 Mount Filesystem Under /data

Create mount point:

mkdir /data

mount /dev/stratis/demopool/projectfs /data

Verify:

df -hT /data

Make persistent (add to /etc/fstab):

echo "/dev/stratis/demopool/projectfs /data xfs defaults 0 0" | tee -a /etc/fstab

---

6 Copy Files to Fill Space

Create some dummy data:

dd if=/dev/zero of=/data/file1.img bs=100M count=5

dd if=/dev/zero of=/data/file2.img bs=50M count=10

#ls -lh /data

---

7 Create Snapshot

stratis fs snapshot demopool projectfs projectfs-snap1

Verify snapshot:

stratis fs list

You should now see projectfs-snap1.

---

8 Delete Files from Main FS

rm -f /data/file1.img

rm -f /data/file2.img

ls -lh /data

---

9 Mount Snapshot to Verify Recovery

Create a temporary mount point:

mkdir /mnt/snap

mount /dev/stratis/demopool/projectfs-snap1 /mnt/snap

Check snapshot contents:

ls -lh /mnt/snap

You should see file1.img and file2.img still present — meaning data can be restored.

- Cleanup (Optional)

  Unmount snapshot & main FS:

  # umount /mnt/snap

  # umount /data

  Delete snapshot/filesystem (if not needed):

  # stratis fs destroy demopool projectfs-snap1

  # stratis fs destroy demopool projectfs [to delete filesystem]

  # stratis fs list

```
# sed  -i  '\|/data|d' /etc/fstab   [delete stratis filesystem entry]
# stratis pool destroy demopool  [to delete pool]
# stratis pool list
```

*****Thank You!*****