RH134 Report

Ruojia Zhang

# Chapter 1

Figure 1

Guided exercise for Writing Simple Bash Scripts. Also set roger Zhang as ps1.

Text

Description automatically generated

Note: Write and execute a simple Bash script. Redirect the output of a simple Bash script to a file. You can see the lab finished here.

Figure 2

Lab: Improving Command-line Productivity. In this lab, I create a Bash script that can filter and get relevant information from different hosts. Text

Description automatically generated

Note: Create a Bash script and redirect its output to a file. Use loops to simplify your code. Filter the relevant content using grep and regular expressions.

Chapter Review

In this chapter, I learned: How to create and execute simple Bash scripts. How to use loops to iterate through a list of items from the command-line and in a shell script. How to search for text in log files and configuration files using regular expressions and grep.

# Chapter 2

Figure 3

Guided Exercise: Scheduling a Deferred User Job. You can see in the screenshot that all test are passed

Text

Description automatically generated

Note: I Schedule a job to run at a specified time in the future. Inspect the commands that a scheduled job runs. Delete the scheduled jobs.

Figure 4

Guided Exercise: Managing Temporary Files. You can see in the screenshot that all tests are passed

Text

Description automatically generated

Note: I have done: Configure systemd-tmpfiles to remove unused temporary files from /tmp. Configure systemd-tmpfiles to periodically purge files from another directory.

Chapter Review

In this chapter, I learned: Jobs that are scheduled to run once in the future are called deferred jobs or tasks. Recurring user jobs execute the user's tasks on a repeating schedule. Recurring system jobs accomplish administrative tasks on a repeating schedule that have system-wide impact. The systemd timer units can execute both the deferred or recurring jobs.

# Chapter 3

Figure 5

Guided Exercise: Adjusting Tuning Profiles. You can see in the screenshot that all test are passed

Text

Description automatically generated

Note: In this exercise, I’ve done tune a server’s performance by activating the tuned service and applying a tuning profile.

Figure 6

Lab: Tuning System Performance. You can see in the screenshot that all tests are passed

Text

Description automatically generated

Note: I have done: Activate a specific tuning profile for a computer system. Adjust the CPU scheduling priority of a process.

Chapter Review

In this chapter, I learned: The tuned service automatically modifies device settings to meet specific system needs based on a pre-defined selected tuning profile. To revert all changes made to system settings by a selected profile, either switch to another profile or deactivate the tuned service. The system assigns a relative priority to a process to determine its CPU access. This priority is called the nice value of a process. The nice command assigns a priority to a process when it starts. The renice command modifies the priority of a running process.

# Chapter 4

Figure 7

Guided Exercise: Securing Files with ACLs. You can see in the screenshot that all test are passed

Text

Description automatically generated

Note: In this exercise, I’ve done Use ACL entries to grant access to a group, and deny access to one of its members. Verify that the existing files and directories reflect the new ACL permissions. Set the default ACL on a directory, and confirm that new files and directories inherit its configuration.

Figure 8

Lab: Controlling Access to Files with ACLs. You can see in the screenshot that all tests are passed

Text

Description automatically generated

Note: I have done: Configure set-GID permission on a folder, to inherit group ownership on files and folders inside. Configure ACL entries to allow or deny read/write/execute permissions to users and groups on files and directories. Configure default ACL to get the right ACL and file permissions automatically, on new files and directories.

Chapter Review

In this chapter, I learned: ACLs provide fine-grained access control to files and directories. The getfacl command displays the ACLs on a file or directory. The setfacl command sets, modifies, and removes default and standard ACLs on files and directories. Use default ACLs for controlling new files and directories permissions. Red Hat Enterprise Linux uses systemd and udev to apply predefined ACLs on devices, folders, and files.

# Chapter 5

Figure 9

Guided Exercise: Changing the SELinux Enforcement Mode. You can see in the screenshot that all test are passed

Text

Description automatically generated

Note: In this exercise, I’ve done Use ACL entries to grant access to a group, and deny access to one of its members. Verify that the existing files and directories reflect the new ACL permissions. Set the default ACL on a directory, and confirm that new files and directories inherit its configuration.

Figure 10

Lab: Managing SELinux Security. You can see in the screenshot that all tests are passed

Text

Description automatically generated

Note: I have done: Identify issues in system log files. Adjust the SELinux configuration..

Chapter Review

In this chapter, I learned: The getenforce and setenforce commands are used to manage the SELinux mode of a system. The semanage command is used to manage SELinux policy rules. The restorecon command applies the context defined by the policy. Booleans are switches that change the behavior of the SELinux policy. They can be enabled or disabled and are used to tune the policy. The sealert displays useful information to help with SELinux troubleshooting.

# Chapter 6

Figure 11

Guided Exercise: Adding Partitions, File Systems, and Persistent Mounts. You can see in the screenshot that all test are passed

Text

Description automatically generated

Note: In this exercise, I’ve done use parted, mkfs.xfs, and other commands to create a partition on a new disk, format it, and persistently mount it..

Figure 12

Lab: Managing Basic Storage. You can see in the screenshot that all tests are passed

Text

Description automatically generated

Note: I have done: Display and create partitions using the parted command. Create new file systems on partitions and persistently mount them. Create swap spaces and activate them at boot.

Chapter Review

In this chapter, I learned You can use the parted command to add, modify, and remove partitions on disks with the MBR or the GPT partitioning scheme. You use the mkfs.xfs command to create XFS file systems on disk partitions. You need to add file-system mount commands to /etc/fstab to make those mounts persistent. You use the mkswap command to initialize swap spaces.

# Chapter 7

Figure 13

Guided Exercise: Creating Logical Volumes. You can see in the screenshot that all test are passed

Text

Description automatically generated

Note: In this exercise, I’ve done Create physical volumes, volume groups, and logical volumes with LVM tools. Create new file systems on logical volumes and persistently mount them.

Figure 14

Lab: Managing Logical Volumes. You can see in the screenshot that all tests are passed Graphical user interface, text

Description automatically generated

Note: I have done: Resize the serverb\_01\_lv logical volume to 768 MiB. Create a new 128 MiB logical volume called serverb\_02\_lv with an XFS file system, persistently mounted at /storage/data2.

Chapter Review

In this chapter, I learned LVM allows you to create flexible storage by allocating space on multiple storage devices. Physical volumes, volume groups, and logical volumes are managed by a variety of tools such as pvcreate, vgreduce, and lvextend. Logical volumes can be formatted with a file system or swap space, and they can be mounted persistently. Additional storage can be added to volume groups and logical volumes can be extended dynamically.

# Chapter 8

Figure 15

Guided Exercise: Managing Layered Storage with Stratis. You can see in the screenshot that all test are passed

Text

Description automatically generated

Note: In this exercise, I’ve done Create a thin-provisioned file system using Stratis storage management solution. Verify that the Stratis volumes grow dynamically to support real-time data growth. Access data from the snapshot of a thin-provisioned file system.

Figure 16

Lab: Implementing Advanced Storage Features. You can see in the screenshot that all tests are passed

Text

Description automatically generated

Note: I have done: Create a thinly provisioned file system using Stratis storage management solution. Verify that the Stratis volumes grow dynamically to support real-time data growth. Access data from the snapshot of a thinly provisioned file system. Create a volume using Virtual Data Optimizer and mount it on a file system. Investigate the impact of data deduplication and compression on a Virtual Data Optimizer volume.

Chapter Review

In this chapter, I learned The Stratis storage management solution implements flexible file systems that grow dynamically with data. The Stratis storage management solution supports thin provisioning, snapshotting, and monitoring. The Virtual Data Optimizer (VDO) aims to reduce the cost of data storage. The Virtual Data Optimizer applies zero-block elimination, data deduplication, and data compression to optimize disk space efficiency.

# Chapter 9

Figure 17

Guided Exercise: Managing Network-Attached Storage with NFS. You can see in the screenshot that all test are passed

Text

Description automatically generated

Note: In this exercise, I’ve done Test an NFS Server using the mount command. Configure NFS Shares in the /etc/fstab configuration file to save changes even after a system reboots.

Figure 18

Lab: Accessing Network-Attached Storage. You can see in the screenshot that all tests are passed

Text

Description automatically generated

Note: I have done: Install required packages needed to set up the automounter. Configure an automounter indirect map, getting resources from a preconfigured NFSv4 server.

Chapter Review

In this chapter, I learned Mount and unmount an NFS export from the command line. Configure an NFS export to automatically mount at startup. Configure the automounter with direct and indirect maps, and describe their differences.

# Chapter 10

Figure 19

Guided Exercise: Selecting the Boot Target. You can see in the screenshot that all test are passed

Text

Description automatically generated

Note: In this exercise, I’ve be able to update the system default target and use a temporary target from the boot loader.

Figure 20

Lab: Controlling the Boot Process. You can see in the screenshot that all tests are passed

Text

Description automatically generated

Note: I have done: Reset a lost root password. Diagnose and fix boot issues. Set the default systemd target.

Chapter Review

In this chapter, I learned systemctl reboot and systemctl poweroff reboot and power down a system, respectively. systemctl isolate target-name.target switches to a new target at runtime. systemctl get-default and systemctl set-default can be used to query and set the default target. Use rd.break on the kernel command line to interrupt the boot process before control is handed over from the initramfs. The root file system is mounted read-only under /sysroot. The emergency target can be used to diagnose and fix file-system issues.

# Chapter 11

Figure21

Guided Exercise: Managing Server Firewalls. You can see in the screenshot that all test are passed

Text

Description automatically generated

Note: In this exercise, I’ve be able to configure firewall rules to control access to services.

Figure 22

Lab: Managing Network Security. You can see in the screenshot that all tests are passed

Text

Description automatically generated

Note: I have done: configure firewall and SELinux settings on a web server host.

Chapter Review

In this chapter, I learned The netfilter subsystem allows kernel modules to inspect every packet traversing the system. All incoming, outgoing or forwarded network packets are inspected. The use of firewalld has simplified management by classifying all network traffic into zones. Each zone has its own list of ports and services. The public zone is set as the default zone. The firewalld service ships with a number of pre-defined services. They can be listed using the firewall-cmd --get-services command. Network traffic is tightly controlled by the SELinux policy. Network ports are labeled. For example, port 22/TCP has the label ssh\_port\_t associated with it. When a process wants to listen on a port, SELinux checks to see whether the label associated with it is allowed to bind that port label. The semanage command is used to add, delete, and modify labels.

# Chapter 12

Figure23

Guided Exercise: Automating Installation with Kickstart. You can see in the screenshot that all test are passed

Text

Description automatically generated

Note: In this exercise, I’ve be able to Create a kickstart file. Use ksvalidator to validate the kickstart file's syntax..

Figure 24

Lab: Installing Red Hat Enterprise Linux. You can see in the screenshot One section can’t pass. Although the file is exactly the same as the lab suggests.

Text

Description automatically generated

Note: I have done: Create a kickstart file. Make the kickstart file available to the installer. Perform a kickstart installation.

Chapter Review

In this chapter, I learned The RHEL 8 binary DVD includes Anaconda and all repositories required for installation. The RHEL 8 boot ISO includes the Anaconda installer, accessing repositories over the network during installation. The Kickstart system performs unattended installations. Kickstart files can be created using the Kickstart Generator website or by copying and editing /root/anaconda-ks.cfg. The virt Yum module provides the packages for a RHEL system to become a virtualization host. The cockpit-machines package adds the Virtual Machines menu to Cockpit.

# Chapter 13

Figure25

Guided Exercise: Automating Installation with Kickstart. You can see in the screenshot that all test are passed

Text

Description automatically generated

Note: In this exercise, I’ve be able to install container management tools and use them to run a container.

Figure 26

Lab: Lab: Running Containers You can see in the screenshot that all test are passed

Text

Description automatically generated

Note: I have done: Create detached containers. Configure port redirection and persistent storage. Configure systemd for containers to start when the host machine starts.

Chapter Review

In this chapter, I learned Containers provide a lightweight way to distribute and run an application and its dependencies that may conflict with software installed on the host. Containers run from container images that you can download from a container registry or create yourself. Podman, provided by Red Hat Enterprise Linux, directly runs and manages containers and container images on a single host. Containers can be run as root, or as non-privileged rootless containers for increased security. You can map network ports on the container host to pass traffic to services running in its containers. You can also use environment variables to configure the software in containers.

A picture containing text

Description automatically generated