Advanced Linux Shell Scripting for DevOps Engineers with User management

Day 5 : 90Days of DevOps Challenge

SS Shivraj Salunkhe · Mar 28, 2023 · 7 min read

Advanced Linux Shell Scripting for DevOps Engineers covers a wide range of topics, including :

1. Automating System Configuration and Maintenance.
2. Version Control and Continuous Integration.
3. Containerization and Orchestration.
4. Infrastructure as Code(IaC).
5. Security and Compliance.

Advanced Linux Shell Scripting for DevOps Engineers can cover a wide range of topics, but one area that may be particularly relevant to DevOps is user management.

In this context, shell scripts can be used to automate the creation, modification, and deletion of user accounts, as well as other related tasks such as managing groups and permissions.

-> Here are some examples of shell scripts for user management:

1. **Creating a User Account**
   To create a user account, you can use the `useradd` command. Here's an example script that prompts the user for a username and password, and then creates the account:

```
#!/bin/bash

# Prompt the user for a username and password
read -p "Enter username: " username
read -s -p "Enter password: " password

# Create the user account with the given username and password
useradd -m -p "$password" "$username"
```

   This script uses the `useradd` command to create a new user account with the `-m` option to create a home directory for the user, and the `-p` option to set the user's password (which must be provided in encrypted form). Note that you should always use encrypted passwords, and never store them in plain text.

2. **Modifying a User Account**

To modify an existing user account, you can use the `usermod` command. Here's an example script that prompts the user for a username and a new password, and then changes the account's password:

```
#!/bin/bash

# Prompt the user for a username and new password
read -p "Enter username: " username
read -s -p "Enter new password: " new_password

# Change the user's password to the new password
echo "$new_password" | passwd --stdin "$username"
```

This script uses the `passwd` command to change the password for the given user. Note that we use the `--stdin` option to read the new password from standard input, which is passed in using the `echo` command.

3. **Deleting a User Account**
   To delete an existing user account, you can use the `userdel` command. Here's an example script that prompts the user for a username and then deletes the account:

```
#!/bin/bash

# Prompt the user for a username to delete
```

```
read -p "Enter username to delete: " username

# Delete the user account with the given username
userdel "$username"
```

This script uses the `userdel` command to delete the user account with the given username. Note that this will also delete the user's home directory and any files it contains, unless the `-r` option is used to remove the directory recursively.

4. **Managing User Groups and Permissions**
   To manage user groups and permissions, you can use a combination of commands such as `groupadd`, `groupdel`, `usermod`, and `chgrp`. Here's an example script that adds a user to a new group and changes the ownership of a directory:

```
#!/bin/bash

# Prompt the user for a username and group name
read -p "Enter username: " username
read -p "Enter group name: " groupname

# Create the new group if it doesn't exist
if ! getent group "$groupname" > /dev/null; then
    groupadd "$groupname"
fi

# Add the user to the group
```

```
usermod -aG "$groupname" "$username"

# Change the ownership of a directory to the new group
chgrp -R "$groupname" /path/to/directory
```

This script first checks if the specified group exists using the `getent` command, and creates it if it doesn't exist.

Task :

1. **You have to do the same using Shell Script i.e using either Loops or command with start day and end day variables using arguments -**

So Write a bash script createDirectories.sh that when the script is executed with three given arguments (one is directory name and second is start number of directories and third is the end number of directories) it creates specified number of directories with a dynamic directory name.

-> Here's a possible implementation of the `createDirectories.sh` script that creates a specified number of directories with a dynamic name:

```
#!/bin/bash

# Get the directory name, start number, and end number from the command-
dir_name=$1
start_num=$2
```

```
end_num=$3

# Loop through the range of numbers and create a directory for each one
for ((i=$start_num; i<=$end_num; i++)); do
  dir_path="${dir_name}${i}"
  mkdir "$dir_path"
done
```

This script takes three arguments from the command line: the base name for the directories (`dir_name`), the starting number for the directories (`start_num`), and the ending number for the directories (`end_num`). It then uses a loop to iterate over the range of numbers from `start_num` to `end_num`, and for each number, it creates a directory with a name formed by concatenating the `dir_name` and the current number.

For example, if you ran the script as `./ createDirectories.sh day 1 90`, it would create 90 directories named `day1`, `day2`, `day3`, and so on, up to `day90`.

1. **Create a Script to backup all your work done till now.**
   here's a basic shell script that can be used to backup all files in the current directory and its subdirectories:

   ```
   #!/bin/bash

   # set the backup directory
   backup_dir=/path/to/backup/directory
   ```

```
# set the backup file name
backup_file=backup_$(date +%Y%m%d_%H%M%S).tar.gz

# create the backup directory if it doesn't exist
mkdir -p "$backup_dir"

# create the backup file
tar czf "$backup_dir/$backup_file" .

# print a success message
echo "Backup created successfully: $backup_dir/$backup_file"
```

Save this script as `backup.sh` and make it executable using `chmod +x backup.sh`. Then, to run the script, simply execute `./backup.sh`.
This script creates a backup directory, sets a backup file name with the current date and time, creates a compressed tar archive of all files in the current directory and its subdirectories, and saves it to the backup directory. Finally, it prints a message indicating the success of the backup operation.

2. **Read About Cron and Crontab, to automate the backup Script.**
Cron is a time-based job scheduler in Linux and Unix-like operating systems. It allows users to schedule jobs (commands or scripts) to run at specified intervals, such as hourly, daily, weekly, or monthly. Cron jobs can be used for various tasks such as backups, system maintenance, data processing, and monitoring.
Crontab is a configuration file that is used to define the schedule and commands for cron jobs. Each user on the system can have their own crontab file, which is used to schedule jobs to run under their user account.

To automate the backup script we created earlier using cron and crontab, follow these steps:

1. Open the crontab file for your user account using the command:
   `crontab -e`
2. Add the following line to the end of the file:

   ```
   0 0 * * * /path/to/backup.sh
   ```

   This line schedules the backup script to run every day at midnight (0 0 * * *).
   Note: Replace `/path/to/backup.sh` with the actual path to your backup script
3. Save and close the crontab file.
   Cron will now execute the backup script automatically at the specified interval. You can modify the schedule to suit your needs by adjusting the values in the crontab line. For example, to run the backup script every hour, you can use the following crontab line:

   ```
   0 * * * * /path/to/backup.sh
   ```

   This line schedules the backup script to run every hour at minute 0 (0 * * * *).

- > Read about User Management.

1. **Creating new user accounts:** This involves setting up a new account with a unique username and password, assigning the user to a group with specific privileges, and specifying the user's home directory.
2. **Modifying user accounts:** This includes changing the user's password, updating user information such as email address and phone number, modifying user permissions, and changing the user's group membership.
3. **Deleting user accounts:** This involves removing a user account from the system, deleting the user's files and directories, and revoking the user's access to system resources.
4. **Managing user groups:** This includes creating new groups, adding users to existing groups, and removing users from groups.

- > Create 2 users and just display their Usernames.

here are the commands to create two users and display their usernames:

1. **To create the first user, use the following command:**

   ```
   sudo useradd -m user1
   ```

   This creates a new user with the username "user1" and creates a home directory for them (-m option).

2. **To create the second user, use the following command:**

```
sudo useradd -m user2
```

This creates a new user with the username "user2" and creates a home directory for them (-m option).

3. **To display the usernames of the two users, you can use the following command:**

```
cat /etc/passwd | grep user
```

This displays the contents of the /etc/passwd file, which contains user account information, and filters out any lines that don't contain the string "user" using the grep command. The output should look something like this:

```
user1:x:1001:1001::/home/user1:/bin/bash
user2:x:1002:1002::/home/user2:/bin/bash
```

The usernames are the first fields separated by colons, so in this case, the usernames are **"user1"** and **"user2"**.
Thank You for Reading the Blog.. 👍
Stay Connect with me for future updates.

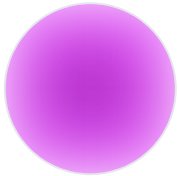
connect with me : linkedin.com/in/shivraj-salunkhe-5881141a4

follow my blog channel : **shivrajofficial.hashnode.dev**

Subscribe to my newsletter

Read articles from directly inside your inbox.
Subscribe to the newsletter, and don't miss out.

Enter your email address SUBSCRIBE



WRITTEN BY

Shivraj Salunkhe

Follow

Dedicated and hardworking undergraduate student pursuing a degree in Information Technology with a passion for learning, leadership experience, and real-world skills through internships and part-time jobs in IT sector. Actively looking for new Opportunities and committed to do personal and professional growth.