Mastering Shell Scripting for DevOps and Cloud

Shell scripting is a powerful tool in DevOps for automating repetitive tasks, managing system configurations, and enhancing CI/CD pipelines. Below is a complete guide on shell scripting for DevOps, along with practical examples.

1. What is Shell Scripting?

Shell scripting is writing a series of commands in a file (script) that the shell can execute in sequence. It's often used to automate tasks, manage servers, and enhance productivity in DevOps.

In Shell scripting every script starts with #!/bin/bash and every script file will save .sh extension.

The Shebang Line: #!/bin/bash

- **Purpose:** This line, often referred to as the “shebang,” tells the operating system which interpreter to use for executing the script. In this case, /bin/bash specifies the Bash shell.
- **Importance:** Without it, the system might try to execute the script as a regular binary, leading to errors.

The .sh Extension: A Convention

- **Purpose:** The .sh extension is a common convention to indicate that a file is a shell script. It helps in file identification and organization.
- **Not Mandatory:** While it’s a widely used practice, it’s not strictly required. Scripts can run without the .sh extension as long as the shebang line is present.

2. Why Shell Scripting in DevOps?

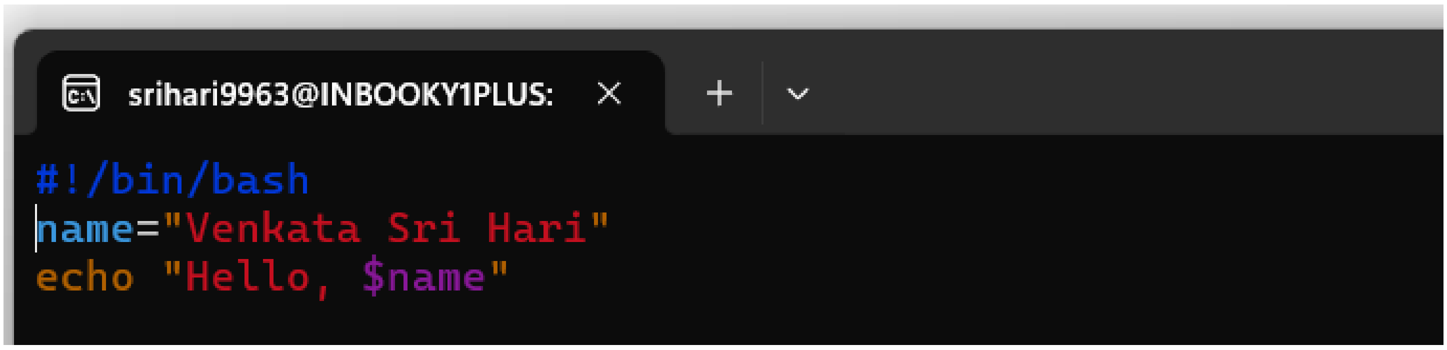
- **Automation:** Automate repetitive tasks like backups, deployments, monitoring, etc.
- **Integration:** Integrate different DevOps tools and systems.
- **Customization:** Tailor scripts to meet specific organizational needs.
- **Efficiency:** Execute multiple commands in a sequence to save time.

3. Basic Shell Scripting Concepts

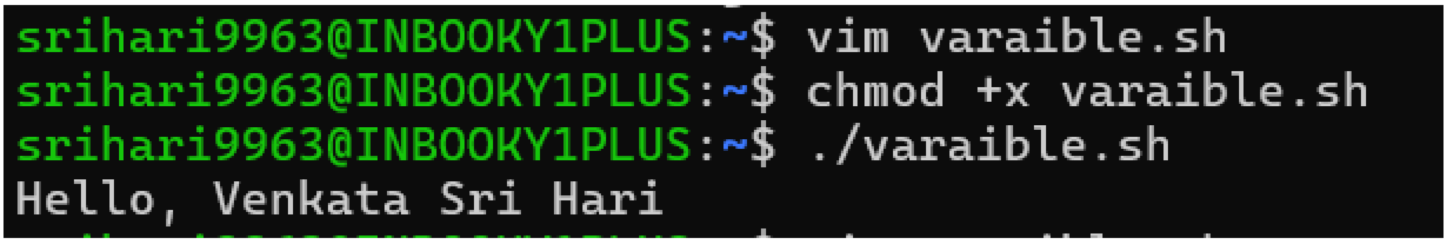
3.1. Variables

Variables store data, here i am using VIM editor to write the scripts in ubuntu.

```bash
#!/bin/bash
name="Venkata Sri Hari"
echo "Hello, $name"
```



```
#!/bin/bash
name="Venkata Sri Hari"
echo "Hello, $name"
```

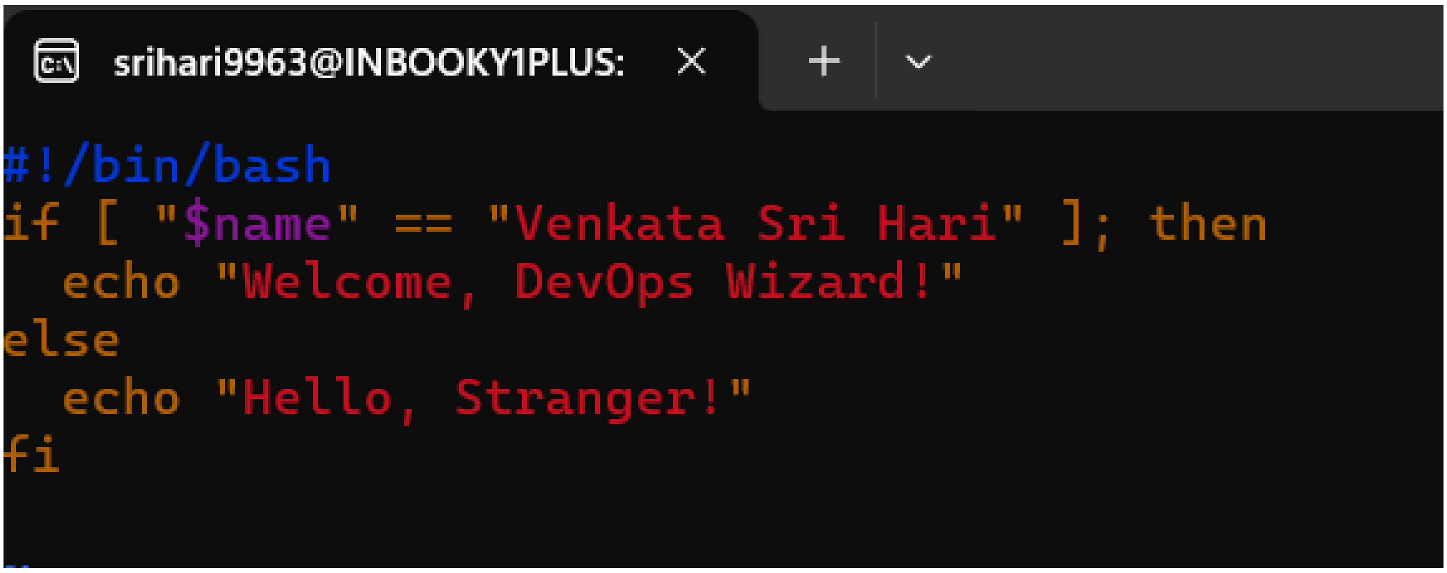


```
srihari9963@INBOOKY1PLUS:~$ vim varaible.sh
srihari9963@INBOOKY1PLUS:~$ chmod +x varaible.sh
srihari9963@INBOOKY1PLUS:~$ ./varaible.sh
Hello, Venkata Sri Hari
```

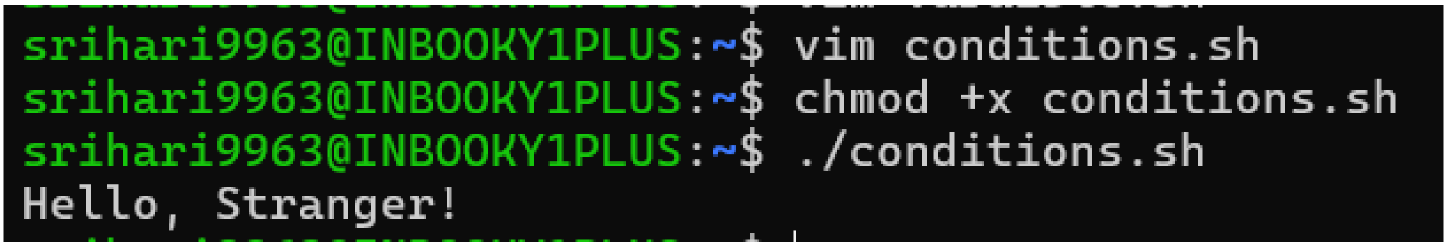
3.2. Conditional Statements

Conditional statements help in decision-making within scripts.

```
#!/bin/bash
if [ "$name" == "Venkata Sri Hari" ]; then
  echo "Welcome, DevOps Wizard!"
else
  echo "Hello, Stranger!"
fi
```



```
#!/bin/bash
if [ "$name" == "Venkata Sri Hari" ]; then
  echo "Welcome, DevOps Wizard!"
else
  echo "Hello, Stranger!"
fi
```



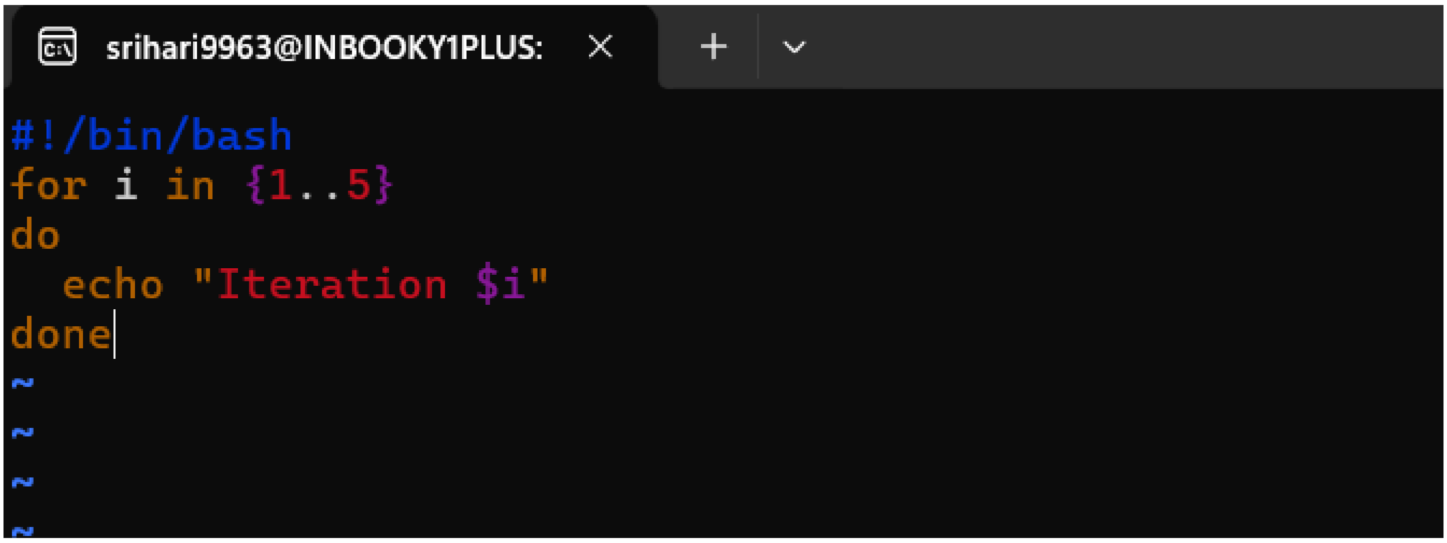
```
sriharі9963@INBOOKY1PLUS:~$ vim conditions.sh
srihari9963@INBOOKY1PLUS:~$ chmod +x conditions.sh
srihari9963@INBOOKY1PLUS:~$ ./conditions.sh
Hello, Stranger!
```

3.3. Loops

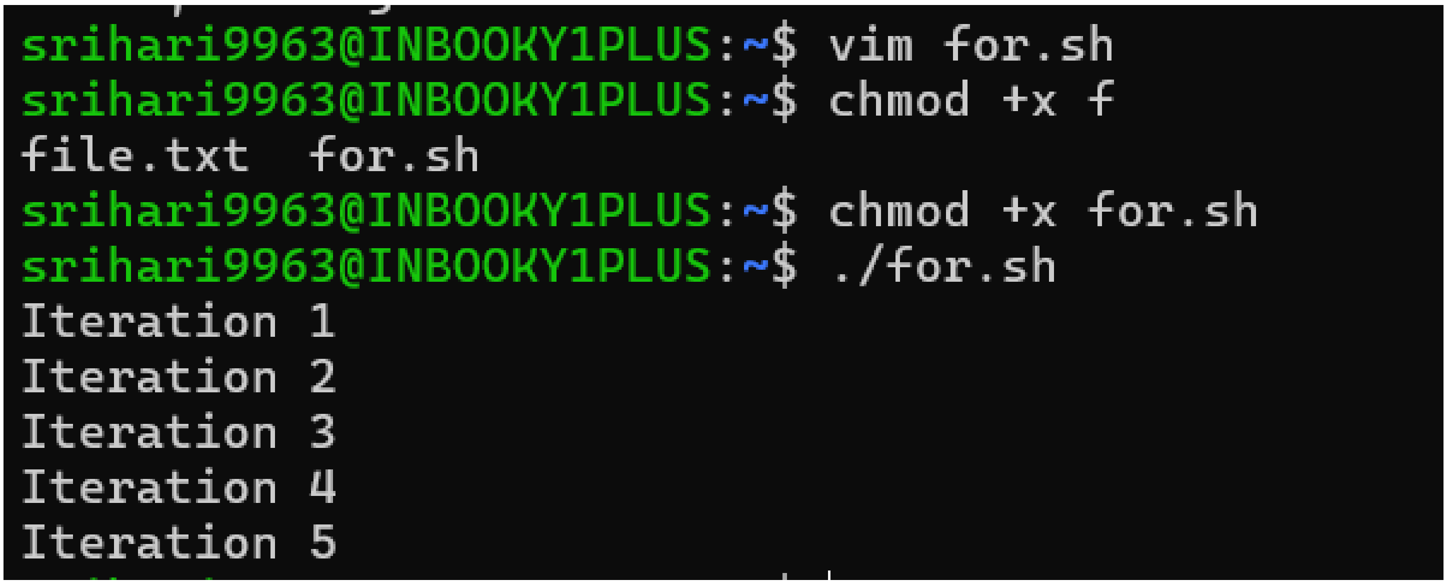
Loops automate repetitive tasks.

- **For Loop:** A for loop in shell scripting is used to iterate over a list of items and execute a block of code for each item. There are primarily two ways to use a for loop in Bash:

```
#!/bin/bash
for i in {1..5}
do
 echo "Iteration $i"
done
```



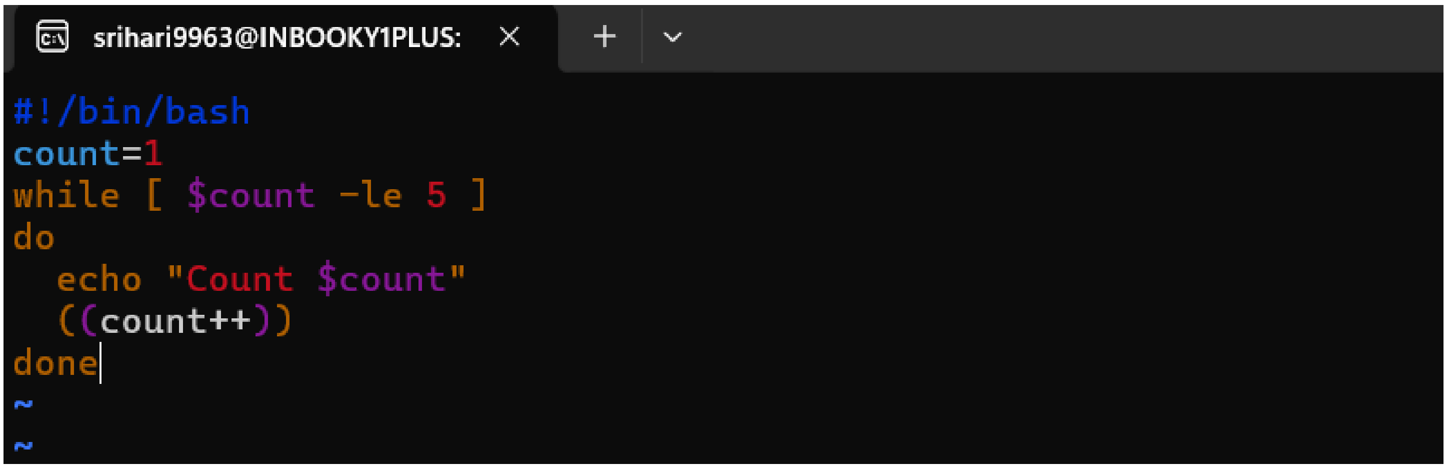
```
#!/bin/bash
for i in {1..5}
do
  echo "Iteration $i"
done
~
~
~
~
```



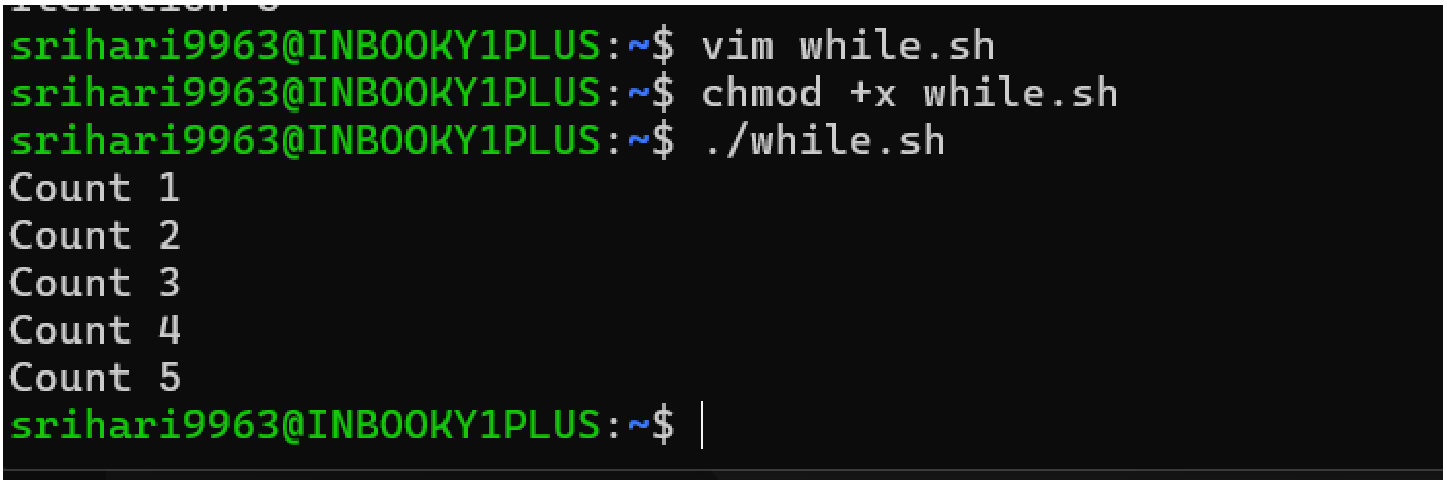
```
sriharí9963@INBOOKY1PLUS:~$ vim for.sh
srihari9963@INBOOKY1PLUS:~$ chmod +x f
file.txt  for.sh
srihari9963@INBOOKY1PLUS:~$ chmod +x for.sh
srihari9963@INBOOKY1PLUS:~$ ./for.sh
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
```

- **While Loop:** A while loop in shell scripting repeatedly executes a block of code as long as a given condition is true.

```
#!/bin/bash
count=1
while [ $count -le 5 ]
do
 echo "Count $count"
 ((count++))
done
```



```
#!/bin/bash
count=1
while [ $count -le 5 ]
do
  echo "Count $count"
  ((count++))
done
~
~
```

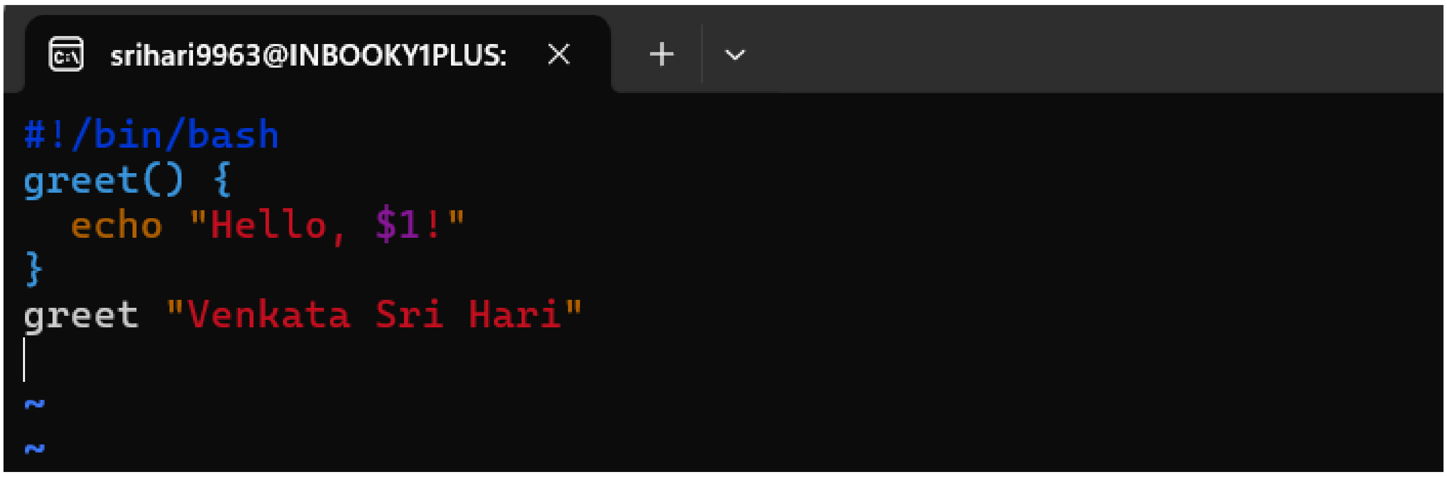


```
sriharі9963@INBOOKY1PLUS:~$ vim while.sh
srihari9963@INBOOKY1PLUS:~$ chmod +x while.sh
srihari9963@INBOOKY1PLUS:~$ ./while.sh
Count 1
Count 2
Count 3
Count 4
Count 5
srihari9963@INBOOKY1PLUS:~$
```

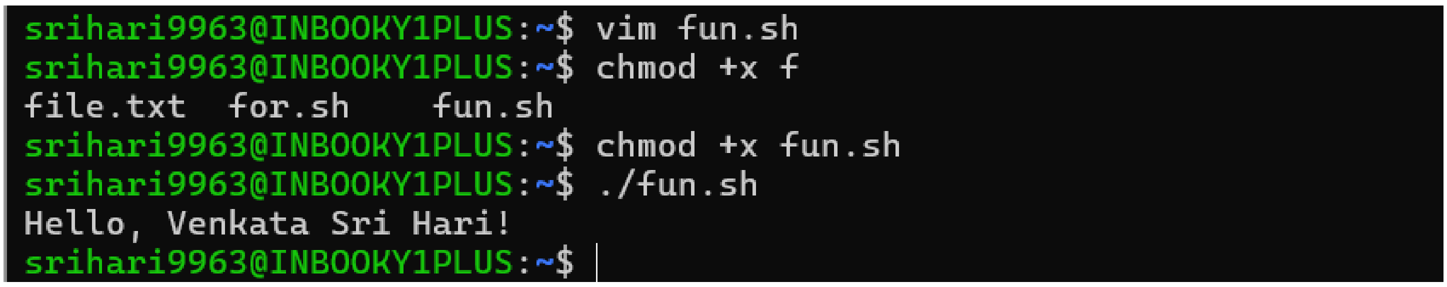
3.4. Functions

Functions encapsulate code blocks that can be reused.

```
#!/bin/bash
greet() {
 echo "Hello, $1!"
}
greet "Venkata Sri Hari"
```



```
#!/bin/bash
greet() {
  echo "Hello, $1!"
}
greet "Venkata Sri Hari"

~
~
```

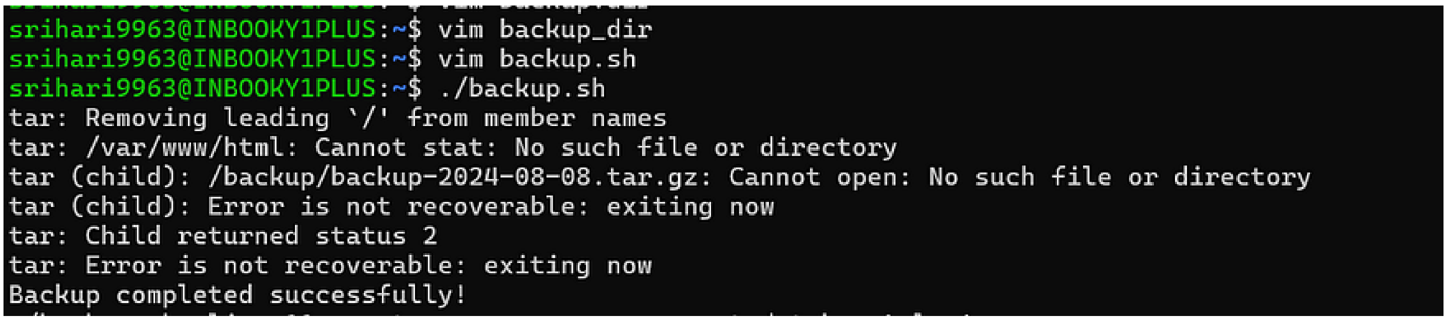


```
srihari9963@INBOOKY1PLUS:~$ vim fun.sh
srihari9963@INBOOKY1PLUS:~$ chmod +x f
file.txt  for.sh    fun.sh
srihari9963@INBOOKY1PLUS:~$ chmod +x fun.sh
srihari9963@INBOOKY1PLUS:~$ ./fun.sh
Hello, Venkata Sri Hari!
srihari9963@INBOOKY1PLUS:~$
```

4. Shell Scripting in DevOps: Practical Examples

4.1. Automating Backups

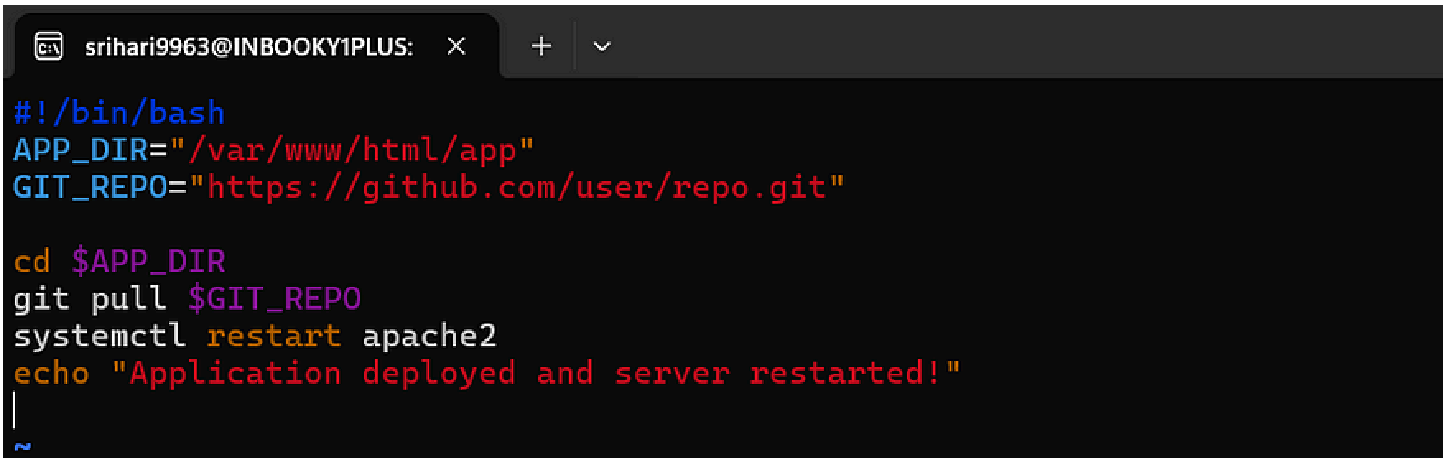
```
#!/bin/bash
BACKUP_DIR="/backup"
SOURCE_DIR="/var/www/html"
DATE=$(date +%F)

tar -czf $BACKUP_DIR/backup-$DATE.tar.gz $SOURCE_DIR
echo "Backup completed successfully!"
```



```
srihari9963@INBOOKY1PLUS:~$ vim backup_dir
srihari9963@INBOOKY1PLUS:~$ vim backup.sh
srihari9963@INBOOKY1PLUS:~$ ./backup.sh
tar: Removing leading `/' from member names
tar: /var/www/html: Cannot stat: No such file or directory
tar (child): /backup/backup-2024-08-08.tar.gz: Cannot open: No such file or directory
tar (child): Error is not recoverable: exiting now
tar: Child returned status 2
tar: Error is not recoverable: exiting now
Backup completed successfully!
```

4.2. Deploying an Application

```
#!/bin/bash
APP_DIR="/var/www/html/app"
GIT_REPO="https://github.com/user/repo.git"

cd $APP_DIR
git pull $GIT_REPO
systemctl restart apache2
echo "Application deployed and server restarted!"
```

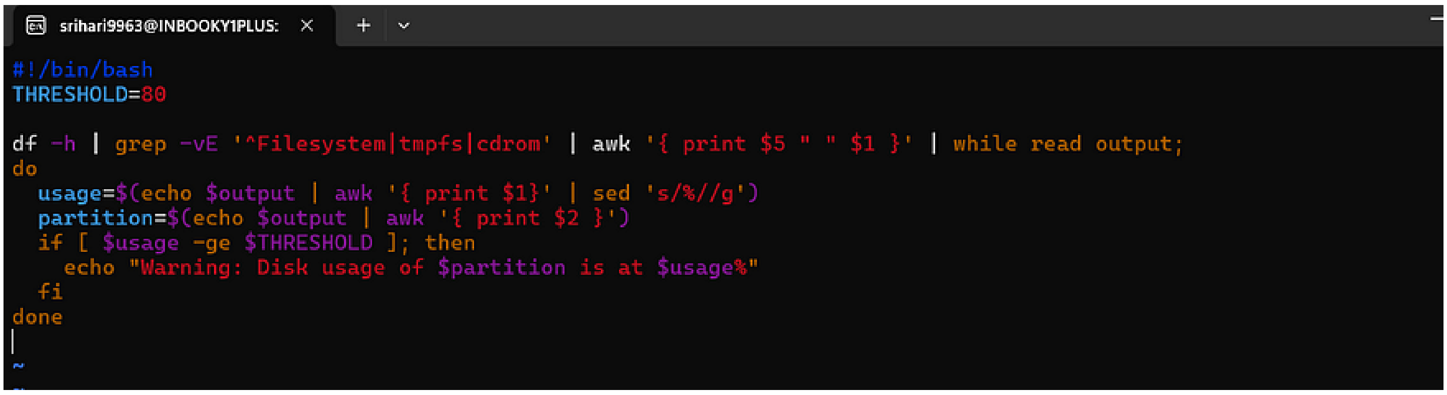


```
#!/bin/bash
APP_DIR="/var/www/html/app"
GIT_REPO="https://github.com/user/repo.git"

cd $APP_DIR
git pull $GIT_REPO
systemctl restart apache2
echo "Application deployed and server restarted!"

~
```

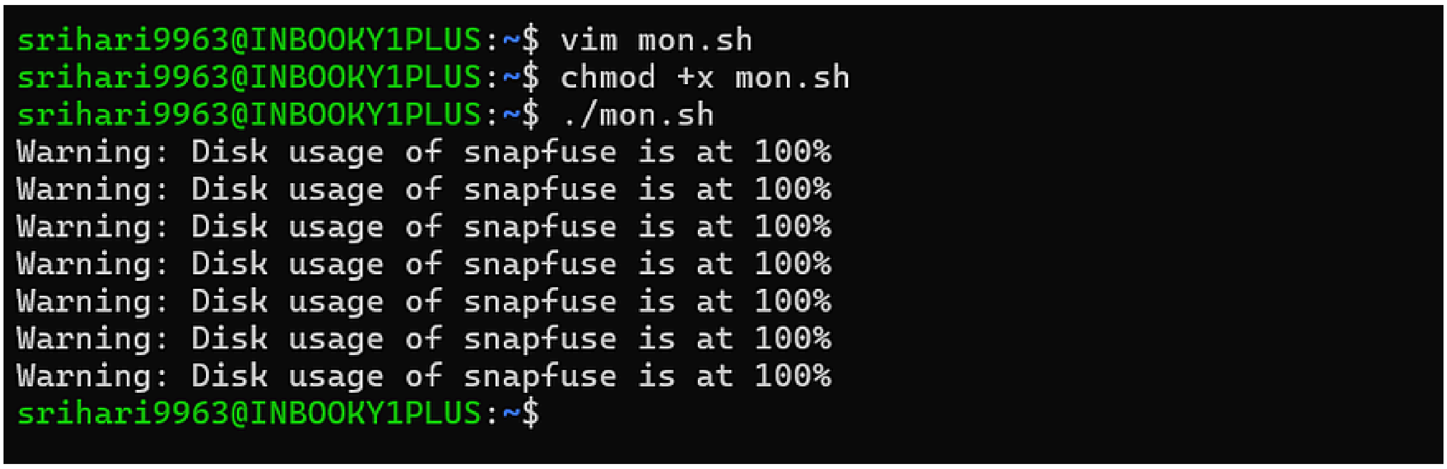
4.3. Monitoring Disk Usage

```
#!/bin/bash
THRESHOLD=80
```

```
df -h | grep -vE '^Filesystem|tmpfs|cdrom' | awk '{ print $5 " " $1 }' | while read output;
do
 usage=$(echo $output | awk '{ print $1}' | sed 's/%//g')
 partition=$(echo $output | awk '{ print $2 }')
 if [ $usage -ge $THRESHOLD ]; then
  echo "Warning: Disk usage of $partition is at $usage%"
 fi
done
```



```
srihari9963@INBOOKY1PLUS:
#!/bin/bash
THRESHOLD=80

df -h | grep -vE '^Filesystem|tmpfs|cdrom' | awk '{ print $5 " " $1 }' | while read output;
do
  usage=$(echo $output | awk '{ print $1}' | sed 's/%//g')
  partition=$(echo $output | awk '{ print $2 }')
  if [ $usage -ge $THRESHOLD ]; then
    echo "Warning: Disk usage of $partition is at $usage%"
  fi
done
~
```



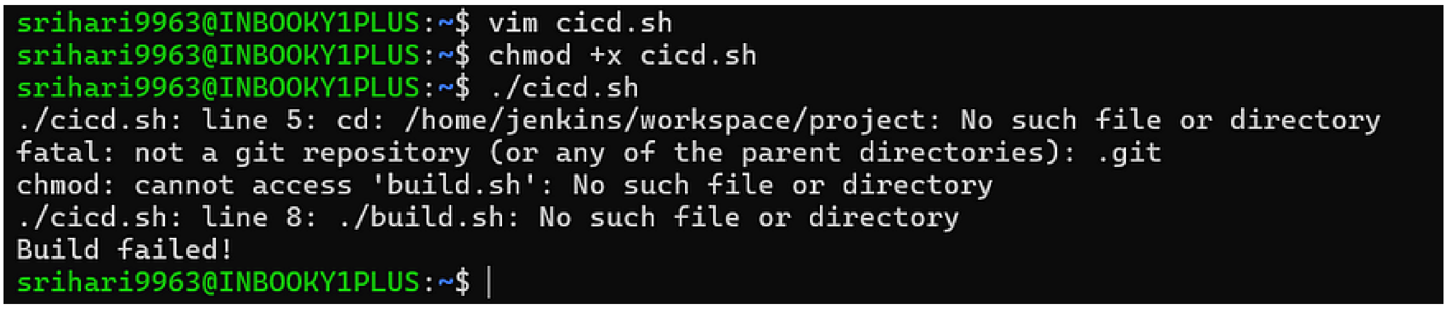
```
srihari9963@INBOOKY1PLUS:~$ vim mon.sh
srihari9963@INBOOKY1PLUS:~$ chmod +x mon.sh
srihari9963@INBOOKY1PLUS:~$ ./mon.sh
Warning: Disk usage of snapfuse is at 100%
Warning: Disk usage of snapfuse is at 100%
Warning: Disk usage of snapfuse is at 100%
Warning: Disk usage of snapfuse is at 100%
Warning: Disk usage of snapfuse is at 100%
Warning: Disk usage of snapfuse is at 100%
Warning: Disk usage of snapfuse is at 100%
srihari9963@INBOOKY1PLUS:~$
```

4.4. Automating CI/CD Pipeline

```
#!/bin/bash
REPO_DIR="/home/jenkins/workspace/project"
BUILD_SCRIPT="build.sh"

cd $REPO_DIR
git pull origin master
chmod +x $BUILD_SCRIPT
./$BUILD_SCRIPT
if [ $? -eq 0 ]; then
 echo "Build successful!"
 ./deploy.sh
```

```
else
 echo "Build failed!"
fi
```

In this case we see the output **“Build Failed!”** because currently I don’t have available the REPO_DIR= “/home/jenkins/workspace/project”, so its failed.



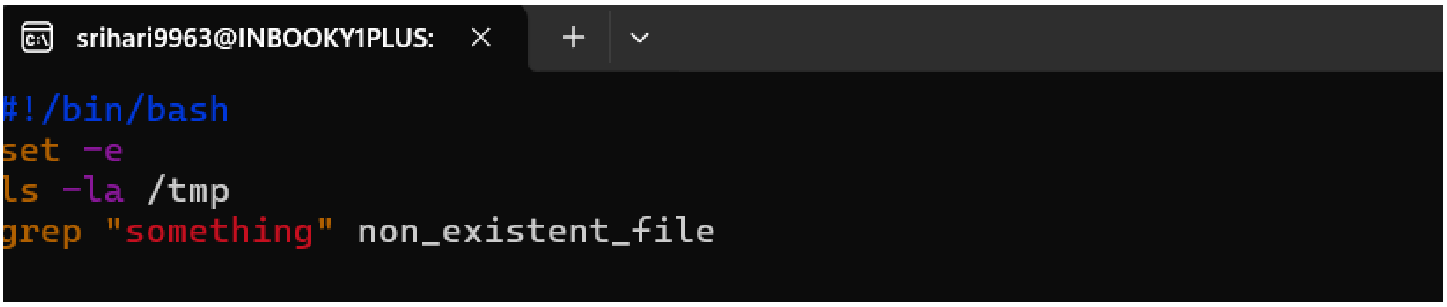
```
srihari9963@INBOOKY1PLUS:~$ vim cicd.sh
srihari9963@INBOOKY1PLUS:~$ chmod +x cicd.sh
srihari9963@INBOOKY1PLUS:~$ ./cicd.sh
./cicd.sh: line 5: cd: /home/jenkins/workspace/project: No such file or directory
fatal: not a git repository (or any of the parent directories): .git
chmod: cannot access 'build.sh': No such file or directory
./cicd.sh: line 8: ./build.sh: No such file or directory
Build failed!
srihari9963@INBOOKY1PLUS:~$ |
```

5. Advanced Shell Scripting Techniques

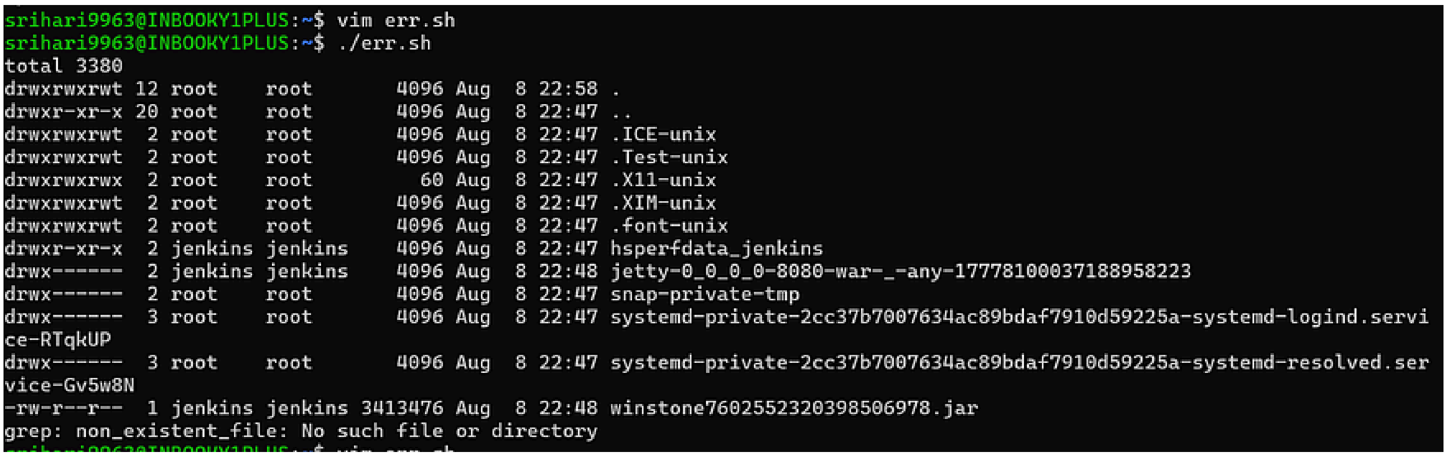
5.1. Error Handling

Use set -e to stop the script if any command fails.

```
#!/bin/bash
set -e
ls -la /tmp
grep "something" non_existent_file
```



```
#!/bin/bash
set -e
ls -la /tmp
grep "something" non_existent_file
```

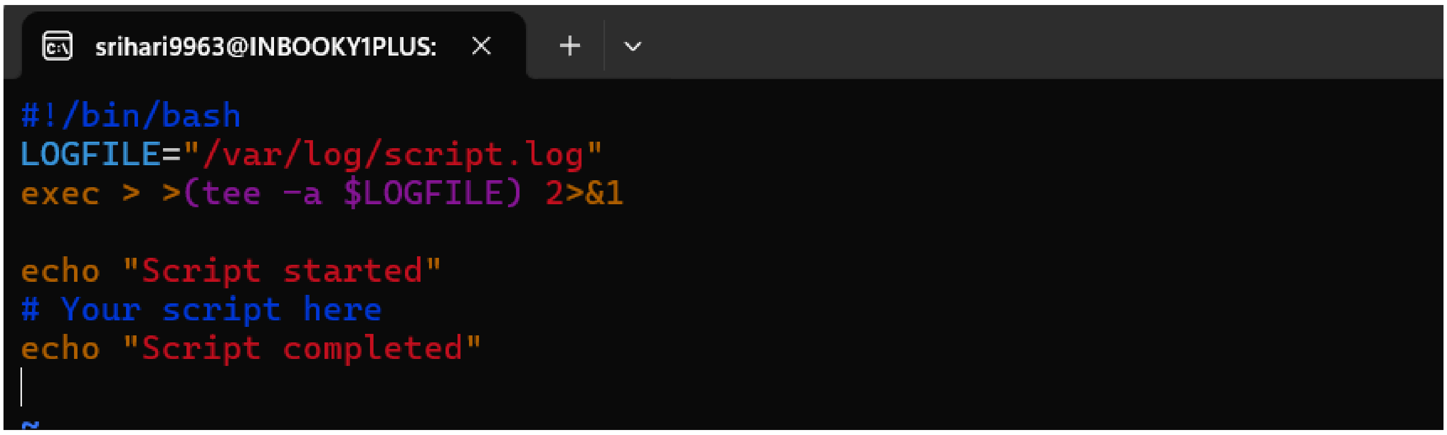


```
sriharі9963@INBOOKY1PLUS:~$ vim err.sh
srihari9963@INBOOKY1PLUS:~$ ./err.sh
total 3380
drwxrwxrwt 12 root    root       4096 Aug  8 22:58 .
drwxr-xr-x 20 root    root       4096 Aug  8 22:47 ..
drwxrwxrwt  2 root    root       4096 Aug  8 22:47 .ICE-unix
drwxrwxrwt  2 root    root       4096 Aug  8 22:47 .Test-unix
drwxrwxrwx  2 root    root         60 Aug  8 22:47 .X11-unix
drwxrwxrwt  2 root    root       4096 Aug  8 22:47 .XIM-unix
drwxrwxrwt  2 root    root       4096 Aug  8 22:47 .font-unix
drwxr-xr-x  2 jenkins jenkins    4096 Aug  8 22:47 hsperfdata_jenkins
drwx------  2 jenkins jenkins    4096 Aug  8 22:48 jetty-0_0_0_0-8080-war-_-any-17778100037188958223
drwx------  2 root    root       4096 Aug  8 22:47 snap-private-tmp
drwx------  3 root    root       4096 Aug  8 22:47 systemd-private-2cc37b7007634ac89bdaf7910d59225a-systemd-logind.servi
ce-RTqkUP
drwx------  3 root    root       4096 Aug  8 22:47 systemd-private-2cc37b7007634ac89bdaf7910d59225a-systemd-resolved.ser
vice-Gv5w8N
-rw-r--r--  1 jenkins jenkins 3413476 Aug  8 22:48 winstone7602552320398506978.jar
grep: non_existent_file: No such file or directory
```

5.2. Logging

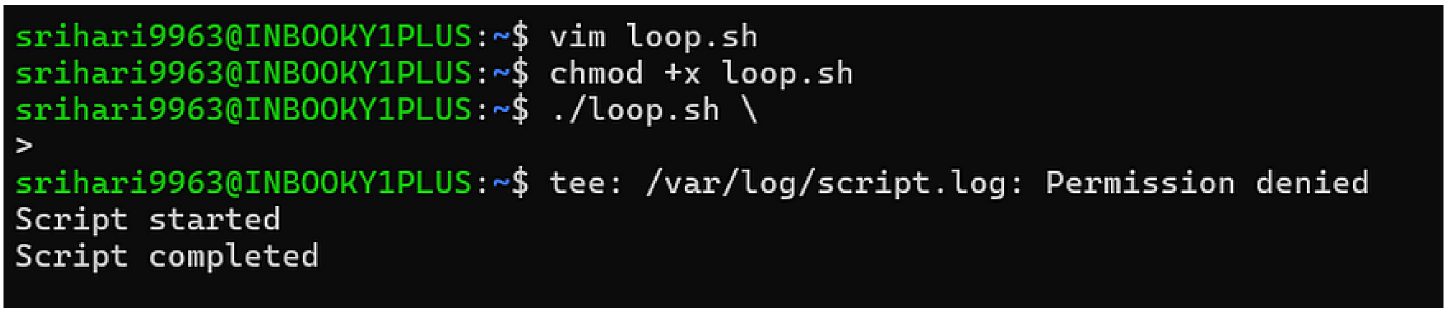
Log script output for auditing and debugging.

```
#!/bin/bash
LOGFILE="/var/log/script.log"
exec > >(tee -a $LOGFILE) 2>&1

echo "Script started"
# Your script here
echo "Script completed"
```



srihari9963@INBOOKY1PLUS:

```
#!/bin/bash
LOGFILE="/var/log/script.log"
exec > >(tee -a $LOGFILE) 2>&1

echo "Script started"
# Your script here
echo "Script completed"
```



```
srihari9963@INBOOKY1PLUS:~$ vim loop.sh
srihari9963@INBOOKY1PLUS:~$ chmod +x loop.sh
srihari9963@INBOOKY1PLUS:~$ ./loop.sh \
>
srihari9963@INBOOKY1PLUS:~$ tee: /var/log/script.log: Permission denied
Script started
Script completed
```

5.3. Scheduling with Cron

Automate script execution with cron jobs.

```
#!/bin/bash

# Example: Run a backup script daily at 2 AM
0 2 * * * /path/to/backup.sh
```

6. File and Directory Management

- Create, delete, copy, and move files and directories:

```
#!/bin/bash

mkdir /tmp/test_directory
touch /tmp/test_directory/test_file.txt
cp /tmp/test_directory/test_file.txt /tmp/test_file_copy.txt
mv /tmp/test_file_copy.txt /tmp/test_directory/
rm -r /tmp/test_directory
```

- Find and search for files:

```
#!/bin/bash
find /var/log -name "*.log"
grep "ERROR" /var/log/syslog
```

- Manage file permissions and ownership:

```
#!/bin/bash

chmod 755 /path/to/file
chown user:group /path/to/file
```

7. User Account Management

```
#!/bin/bash
user_name="devops_user"
useradd $user_name
passwd $user_name
echo "User $user_name created and password set."
```

8. System Health Check Script

```
#!/bin/bash
echo "Checking system health..."
df -h
free -m
```

```
uptime
echo "System health check completed."
```

These examples provide a foundation for creating shell scripts that automate common DevOps tasks, improve efficiency, and maintain system stability.

You’re welcome! Have a great time ahead! Enjoy your day!

Please Connect with me any doubts.

Mail: sriharimalapati6@gmail.com

LinkedIn: www.linkedin.com/in/

GitHub: https://github.com/Consultantsrihari

Medium: Sriharimalapati — Medium

Thanks for watching ##### %%%% Sri Hari %%%%

Shellscripting

Shell Script

DevOps

Automation

Linux