After adding a *normal_user* and an *admin_user* to the system, I added a command named *health* to the server's command line. This command runs a Python file named *system_health.py*, which displays the CPU, RAM, and disk usage of the system. Here's how I did it:

```
erfan@erfan-virtual-machine:/usr/local/bin$ nano system_health.py
```

```
erfan@erfan-virtual-machine: /usr/local/bin
erfan@erfan-virtual-machine: /usr/local/bin   admin_user@erfan-virtual-machine: ~   normal_user@erfan-virtual-machine: ~
  GNU nano 4.8                                  system_health.py
#!/usr/bin/env python3
import psutil

def get_health():
    cpu = f"CPU: {psutil.cpu_percent()}%"
    mem = f"RAM: {psutil.virtual_memory().percent}%"
    disk = f"Disk: {psutil.disk_usage('/').percent}%"
    return "\n".join([cpu, mem, disk])

if __name__ == "__main__":
    print("=== System Health ===")
    print(get_health())

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text
```

```
erfan@erfan-virtual-machine:/usr/local/bin$ sudo chmod +x /usr/local/bin/system_health.py
```

```
erfan@erfan-virtual-machine:/usr/local/bin$ sudo mv /usr/local/bin/system_health.py /usr/local/bin/health
erfan@erfan-virtual-machine:/usr/local/bin$ health
=== System Health ===
CPU: 0.0%
RAM: 74.3%
Disk: 100.0%
erfan@erfan-virtual-machine:/usr/local/bin$
```

Then, I created an *ssh_connection.py* file for both the *admin_user* and *normal_user* to establish an SSH connection using the *Paramiko* library:

```
#!/usr/bin/env python3
import paramiko
import getpass
import sys
import socket
import tty
import termios
import select

def setup_terminal():
    """Set up terminal for raw input"""
    old_attrs = termios.tcgetattr(sys.stdin)
    tty.setraw(sys.stdin.fileno())
    return old_attrs

def restore_terminal(old_attrs):
    """Restore terminal settings"""
    termios.tcsetattr(sys.stdin, termios.TCSADRAIN, old_attrs)

def interactive_shell(channel):
    """Handle the interactive shell session"""
    old_attrs = setup_terminal()

    try:
        while True:
            r, w, e = select.select([channel, sys.stdin], [], [])

            if channel in r:
                try:
                    data = channel.recv(1024)
                    if not data:
                        break
                    sys.stdout.write(data.decode())
                    sys.stdout.flush()
                except socket.timeout:
                    continue

            if sys.stdin in r:
                char = sys.stdin.read(1)
                if char == '\x1d':  # Ctrl+] to exit
                    break
                channel.send(char)

    finally:
        restore_terminal(old_attrs)
        print("\nConnection closed.")
```

```python
def main():
    print("=== Python SSH Client ===")
    host = input("Server IP/Hostname: ").strip()
    user = input("Username: ").strip()
    pwd = getpass.getpass("Password (leave empty for SSH key auth): ") or None

    client = paramiko.SSHClient()
    client.load_system_host_keys()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    try:
        client.connect(
            hostname=host,
            username=user,
            password=pwd,
            look_for_keys=True,
            allow_agent=True,
            timeout=10
        )

        channel = client.invoke_shell(term='xterm-256color')
        channel.settimeout(1)

        print(f"\nConnected to {host}. Press Ctrl+] to exit.\n")
        interactive_shell(channel)

    except Exception as e:
        print(f"\nError: {str(e)}")
    finally:
        client.close()

if __name__ == "__main__":
    main()
```

After that, I created a *command_wrapper.sh* file to restrict the shell environment, allowing only the execution of the *health* command:

```bash
#!/bin/bash

ALLOWED_COMMAND="health"
PS1="> "

while true; do
    read -p "$PS1" CMD

    if [[ "$CMD" == "exit" ]]; then
        echo "Exiting..."
        break
    elif [[ "$CMD" == "$ALLOWED_COMMAND" ]]; then
        $CMD
    else
        echo "Error: Command not allowed."
    fi
done
```

Then, I opened the server's *authorized_keys* file—which contains the public keys for both *normal_user* and *admin_user*—and modified *normal_user*'s key to enforce the restrictions and finally restarted the ssh server:

```
  GNU nano 4.8                                                    /home/erfan/.ssh/authorized_keys
command="/home/erfan/Desktop/command_wrapper.sh" ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIAfGV5FFgHzligheQ8bFWkT3O0OghrLoN6XqyDyQ9bLd erfahmadi03@gmail.com
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIJ27xjtXlGmSV5H3M/NJjijpJS+sKlE/SHbokEKqIR0E erfahmadi03@gmail.com
```

Here is the execution of this part for *normal_user*:

```
normal_user@erfan-virtual-machine:~$ python3 ssh_connection.py
=== Python SSH Client ===
Server IP/Hostname: 192.168.38.130
Username: erfan
Password (leave empty for SSH key auth):

Connected to 192.168.38.130. Press Ctrl+] to exit.

> ls
Error: Command not allowed.
> pwd
Error: Command not allowed.
> health
=== System Health ===
CPU: 0.0%
RAM: 79.3%
Disk: 100.0%
> exit
Exiting...

Connection closed.
normal_user@erfan-virtual-machine:~$
```

And here is the execution of this part for *admin_user*:

```
admin_user@erfan-virtual-machine:~$ python3 ssh_connection.py
=== Python SSH Client ===
Server IP/Hostname: 192.168.38.130
Username: erfan
Password (leave empty for SSH key auth):

Connected to 192.168.38.130. Press Ctrl+] to exit.

Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-136-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Introducing Expanded Security Maintenance for Applications.
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

     https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
New release '22.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Wed Apr  9 15:39:04 2025 from 192.168.38.130
erfan@erfan-virtual-machine:~$ ls
Desktop    Downloads  Music     Public  Templates
Documents  go         Pictures  snap    Videos
erfan@erfan-virtual-machine:~$ pwd
/home/erfan
erfan@erfan-virtual-machine:~$ health
=== System Health ===
CPU: 0.0%
RAM: 78.8%
Disk: 100.0%
erfan@erfan-virtual-machine:~$ exit
logout

Connection closed.
admin_user@erfan-virtual-machine:~$
```

I wrote a Bash script that automatically backs up a specified directory to a target location at regular time intervals. The script runs periodically every *time_interval* minutes (depending on the configuration):

```
  GNU nano 4.8
#!/bin/bash

if [ "$#" -ne 3 ]; then
    echo "Usage: $0 input_directory output_directory time_interval_minutes"
    echo "Example: $0 /home/user/documents /backups 60"
    exit 1
fi

input_dir="$1"
output_dir="$2"
interval_minutes="$3"

interval_seconds=$((interval_minutes * 60))

if [ ! -d "$input_dir" ]; then
    echo "Error: Input directory $input_dir does not exist"
    exit 1
fi

mkdir -p "$output_dir"

echo "Starting backup process:"
echo "  Source:      $input_dir"
echo "  Destination: $output_dir"
echo "  Interval:    every $interval_minutes minutes"
echo "Press Ctrl+C to stop"

while true; do
    timestamp=$(date +"%Y%m%d_%H%M%S")
    backup_name="backup_${timestamp}.tar.gz"
    backup_path="${output_dir}/${backup_name}"

    echo -n "$(date '+%Y-%m-%d %H:%M:%S') - Creating backup..."
    tar -czf "$backup_path" -C "$(dirname "$input_dir")" "$(basename "$input_dir")"

    if [ $? -eq 0 ] && [ -f "$backup_path" ]; then
        backup_size=$(du -h "$backup_path" | cut -f1)
        echo " done! (size: $backup_size)"
    else
        echo " failed!"
        exit 1
    fi

    sleep "$interval_seconds"
done
```

```
erfan@erfan-virtual-machine:~/Desktop$ nano backup_script.sh
erfan@erfan-virtual-machine:~/Desktop$ chmod +x backup_script.sh
erfan@erfan-virtual-machine:~/Desktop$ ./backup_script.sh /home/erfan/Desktop/ECS/ /home/erfan/Desktop/backup 1
Starting backup process:
  Source:      /home/erfan/Desktop/ECS/
  Destination: /home/erfan/Desktop/backup
  Interval:    every 1 minutes
Press Ctrl+C to stop
2025-04-10 21:30:46 - Creating backup... done! (size: 12M)
2025-04-10 21:31:48 - Creating backup... done! (size: 12M)
^C
```

In the next step, I modified the *ssh_connection.py* file for the *admin_user* to include file transfer capabilities, enabling both download and upload operations to/from the server. The implementation uses Paramiko's SFTP functionalities:

```python
def download_file(ftp_client, local_path, remote_path):
    """Download file from remote server to local machine"""
    try:
        ftp_client.get(remote_path, local_path)
        print(f"File downloaded to {local_path}.")
    except Exception as e:
        print(f"Error downloading file: {str(e)}")

def upload_file(ftp_client, local_path, remote_path):
    """Upload file from local machine to remote server"""
    try:
        ftp_client.put(local_path, remote_path)
        print(f"File uploaded to {remote_path}.")
    except Exception as e:
        print(f"Error uploading file: {str(e)}")
```

```
admin_user@erfan-virtual-machine:~$ cat A
hello
admin_user@erfan-virtual-machine:~$ python3 ssh_connection.py
=== Python SSH Client ===
Server IP/Hostname: 192.168.38.130
Username: erfan
Password (leave empty for SSH key auth):

Connected to 192.168.38.130.


Enter command (Download/Upload)_file local_path remote_path, or 'shell' to enter interactive shell, or 'exit' to quit: Upload_file
Invalid command. Type 'Download_file', 'Upload_file', 'shell', or 'exit'.

Enter command (Download/Upload)_file local_path remote_path, or 'shell' to enter interactive shell, or 'exit' to quit: Upload_file /home/admin_user/A /home/erfan/Desktop/A
File uploaded to /home/erfan/Desktop/A.

Enter command (Download/Upload)_file local_path remote_path, or 'shell' to enter interactive shell, or 'exit' to quit: shell
Entering interactive shell. Type 'exit' to leave.
Last login: Wed Apr  9 19:15:56 2025 from 192.168.38.130
erfan@erfan-virtual-machine:~$ cd Desktop/
erfan@erfan-virtual-machine:~/Desktop$ ls
 A                                    console_export
 command_wrapper.sh                  'distributed systems'
'Console - Dev Tools - Elastic_files'  ECS
'Console - Dev Tools - Elastic.html'   system_health.py
erfan@erfan-virtual-machine:~/Desktop$ cat A
hello
erfan@erfan-virtual-machine:~/Desktop$ exit
logout

Connection closed.

Enter command (Download/Upload)_file local_path remote_path, or 'shell' to enter interactive shell, or 'exit' to quit: Download_file /home/admin_user/B /home/erfan/Desktop/A
File downloaded to /home/admin_user/B.

Enter command (Download/Upload)_file local_path remote_path, or 'shell' to enter interactive shell, or 'exit' to quit: exit
admin_user@erfan-virtual-machine:~$ ls
A  B  Downloads  e1  ee  eeeede  eeeedes  eeeee  eww  myenv  ssh_connection.py
admin_user@erfan-virtual-machine:~$ cat B
hello
admin_user@erfan-virtual-machine:~$
```

Bonus:

Finally, I modified the *ssh_connection.py* file to include a *commands_log* list that records every command entered by the client. Upon connection termination, the script displays the complete command history.

Admin user ssh connection:

```python
def main():
    print("=== Python SSH Client ===")
    host = input("Server IP/Hostname: ").strip()
    user = input("Username: ").strip()
    pwd = getpass.getpass("Password (leave empty for SSH key auth): ") or None
    client = paramiko.SSHClient()
    client.load_system_host_keys()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    try:
        client.connect(
            hostname=host,
            username=user,
            password=pwd,
            look_for_keys=True,
            allow_agent=True,
            timeout=10
        )
        print(f"\nConnected to {host}.\n")
        ftp_client = client.open_sftp()
        commands_log = list()
        while True:
            command = input("\nEnter command (Download/Upload)_file local_path remote_path, or 'shell' to enter interactive shell, or 'exit' to quit: ").strip()
            if command.lower().startswith("download_file") or command.lower().startswith("upload_file"):
                parts = command.split()
                if len(parts) == 3:
                    operation, local_path, remote_path = parts
                    if operation.lower() == "download_file":
                        download_file(ftp_client, local_path, remote_path, commands_log)
                    elif operation.lower() == "upload_file":
                        upload_file(ftp_client, local_path, remote_path, commands_log)
                else:
                    print("Invalid command format. Please use: <Download/Upload>_file local_path remote_path")
            elif command.lower() == "shell":
                channel = client.invoke_shell(term='xterm-256color')
                channel.settimeout(1)
                print("Entering interactive shell. Type 'exit' to leave.")
                interactive_shell(channel, commands_log)
            elif command.lower() == "exit":
                print("commands log:\n")
                for command in commands_log:
                    print(command)
                break
            else:
                print("Invalid command. Type 'Download_file', 'Upload_file', 'shell', or 'exit'.")
        ftp_client.close()

    except Exception as e:
        print(f"\nError: {str(e)}")
    finally:
        client.close()
```

```
Server IP/Hostname: 192.168.38.130
Username: erfan
Password (leave empty for SSH key auth):

Connected to 192.168.38.130.


Enter command (Download/Upload)_file local_path remote_path, or 'shell' to enter interactive shell, or 'exit' to quit: shell
Entering interactive shell. Type 'exit' to leave.
Last login: Thu Apr 10 21:05:25 2025 from 192.168.38.130
erfan@erfan-virtual-machine:~$ ls
Desktop    Downloads  Music     Public  Templates
Documents  go         Pictures  snap    Videos
erfan@erfan-virtual-machine:~$ cd Desktop/
erfan@erfan-virtual-machine:~/Desktop$ ls
 A                                  'Console - Dev Tools - Elastic.html'
 admin_connection.py                 console_export
 B                                  'distributed systems'
 backups                             ECS
 command_wrapper.sh                  system_health.py
'Console - Dev Tools - Elastic_files'   user_connection.py
erfan@erfan-virtual-machine:~/Desktop$ nano A
Enter command (Download/Upload)_file local_path remote_path, or 'shell' to enter interactive shell, or 'exit' to quit: Download_f
Invalid command. Type 'Download_file', 'Upload_file', 'shell', or 'exit'.

Enter command (Download/Upload)_file local_path remote_path, or 'shell' to enter interactive shell, or 'exit' to quit: Download_file /home/admin_user/A1 /home/erfan/Desktop/A
File downloaded to /home/admin_user/A1.

Enter command (Download/Upload)_file local_path remote_path, or 'shell' to enter interactive shell, or 'exit' to quit: shell
Entering interactive shell. Type 'exit' to leave.
Last login: Thu Apr 10 21:16:09 2025 from 192.168.38.130
erfan@erfan-virtual-machine:~$ ls
Desktop    Downloads  Music     Public  Templates
Documents  go         Pictures  snap    Videos
erfan@erfan-virtual-machine:~$ exit
logout

Connection closed.

Enter command (Download/Upload)_file local_path remote_path, or 'shell' to enter interactive shell, or 'exit' to quit: Upload_file /home/admin_user/A1 /home/erfan/Desktop/B1
File uploaded to /home/erfan/Desktop/B1.

Enter command (Download/Upload)_file local_path remote_path, or 'shell' to enter interactive shell, or 'exit' to quit: exit
commands log:

ls
cd Desktio/op
ls
nano A
exit
downloaded /home/erfan/Desktop/A to /home/admin_user/A1
downloaded /home/erfan/Desktop/A to /home/admin_user/A1
ls
exit
uploaded /home/admin_user/A1 to /home/erfan/Desktop/B1
admin_user@erfan-virtual-machine:~$
```

Normal user ssh connection:

```python
def main():
    print("=== Python SSH Client ===")
    host = input("Server IP/Hostname: ").strip()
    user = input("Username: ").strip()
    pwd = getpass.getpass("Password (leave empty for SSH key auth): ") or None
    client = paramiko.SSHClient()
    client.load_system_host_keys()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    try:
        client.connect(
            hostname=host,
            username=user,
            password=pwd,
            look_for_keys=True,
            allow_agent=True,
            timeout=10
        )
        channel = client.invoke_shell(term='xterm-256color')
        channel.settimeout(1)
        commands_log = list()
        print(f"\nConnected to {host}. Press Ctrl+] to exit.\n")
        interactive_shell(channel, commands_log)
    except Exception as e:
        print(f"\nError: {str(e)}")
    finally:
        print("commands log:\n")
        for command in commands_log:
            print(command)
        client.close()
```

```
normal_user@erfan-virtual-machine:~$ python3 ssh_connection.py
=== Python SSH Client ===
Server IP/Hostname: 192.168.38.130
Username: erfan
Password (leave empty for SSH key auth):

Connected to 192.168.38.130. Press Ctrl+] to exit.

> ls
Error: Command not allowed.
> health
=== System Health ===
CPU: 0.0%
RAM: 84.0%
Disk: 90.0%
> exit
Exiting...

Connection closed.
commands log:

ls
health
exit
normal_user@erfan-virtual-machine:~$
```