|  |  |
| --- | --- |
|  | **Qatar University**  **College of Engineering**  **Department of Computer Science and Engineering** |

Operating Systems-(CMPS 450)

**Task: Project Phase 1**

*Fall 2024*

**Project Group Members:**

Marwan Hashish ID:201701546

Abdulla Jamali ID: 202104080

Nasser Aljufairi ID: 202005009

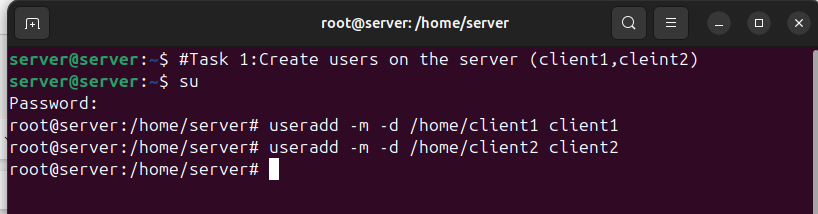
Murshed Al-Muhannadi ID: 201706102

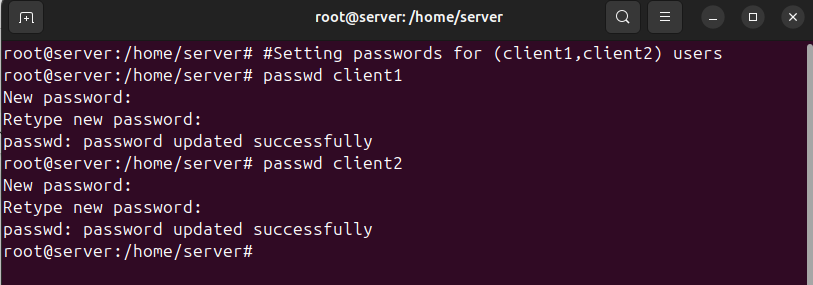
**Instructor**: ***Eng.Heba Dawoud***

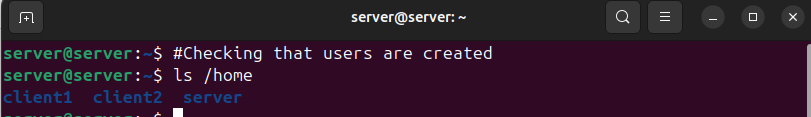
|  |  |  |
| --- | --- | --- |
| **student name** | **Student part in the Project** | **participation percentage** |
| Marwan Hashish |  |  |
| Abdulla Jamali |  |  |
| Nasser Aljufairi |  |  |
| Murshed Al-Muhannadi |  |  |

**Task 1: Setup Server side (VM1)**

**-Create users on the server (client 1, client2)**

-Setting passwords for client1 & client2

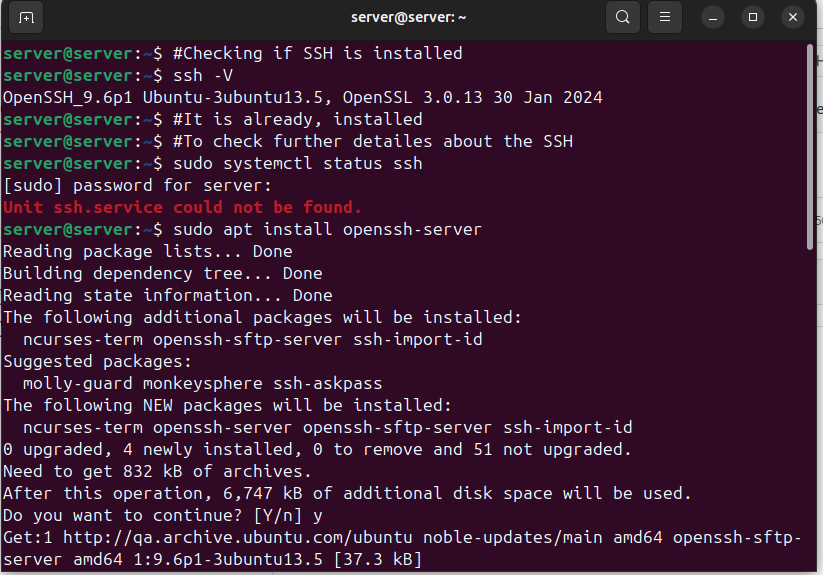
-Checking that users are created (client1, client2):

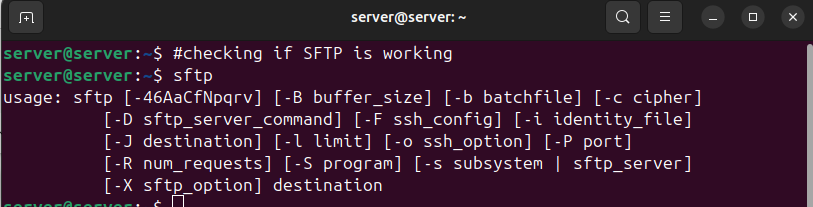


**-Install and enable SHHD**

-Checking if SSH is installed  
-The command to install SSH if it’s not installed: $ sudo apt install openssh-server

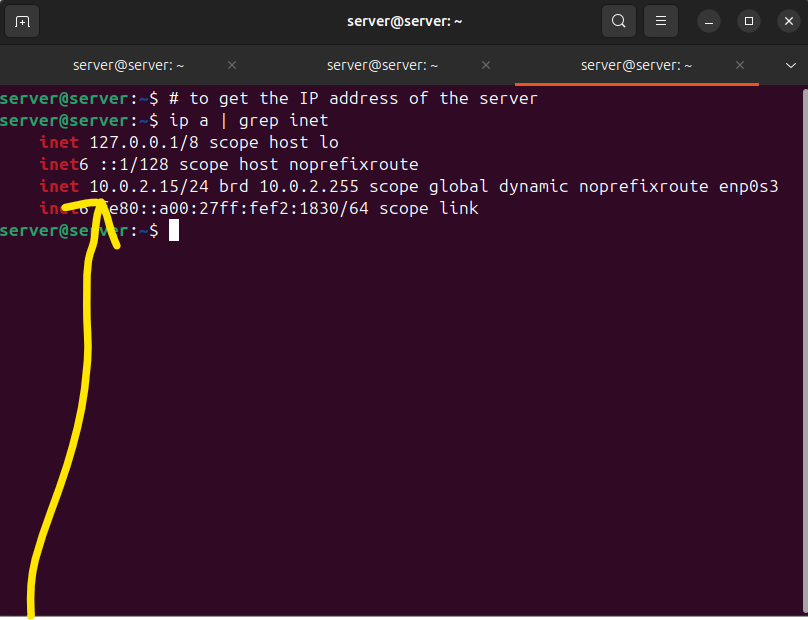
But it was already installed so no need for this step.

-Check if SFTP is working

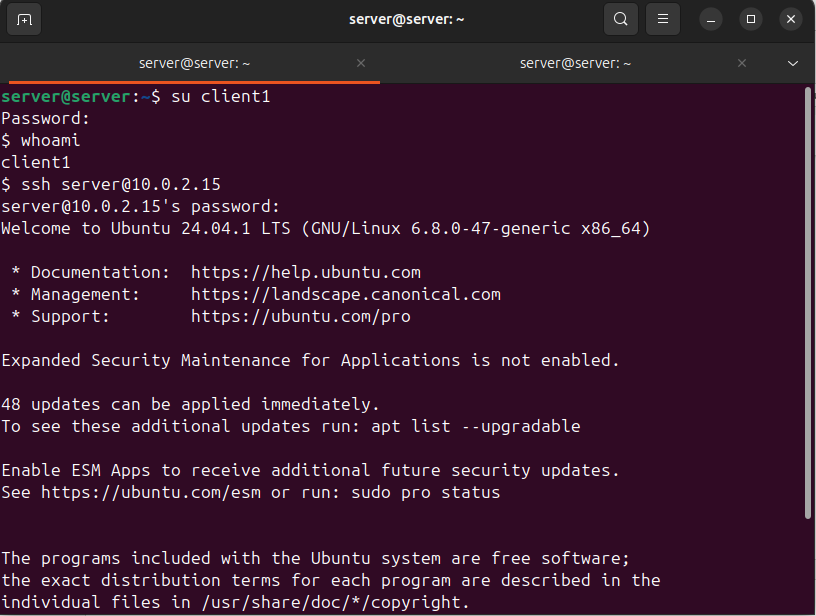


**Task 2 : Configuration**

To Get the IP of the server we did:



Accessing the server from client 1 using SSH and the ip address of the server we just retrieved:

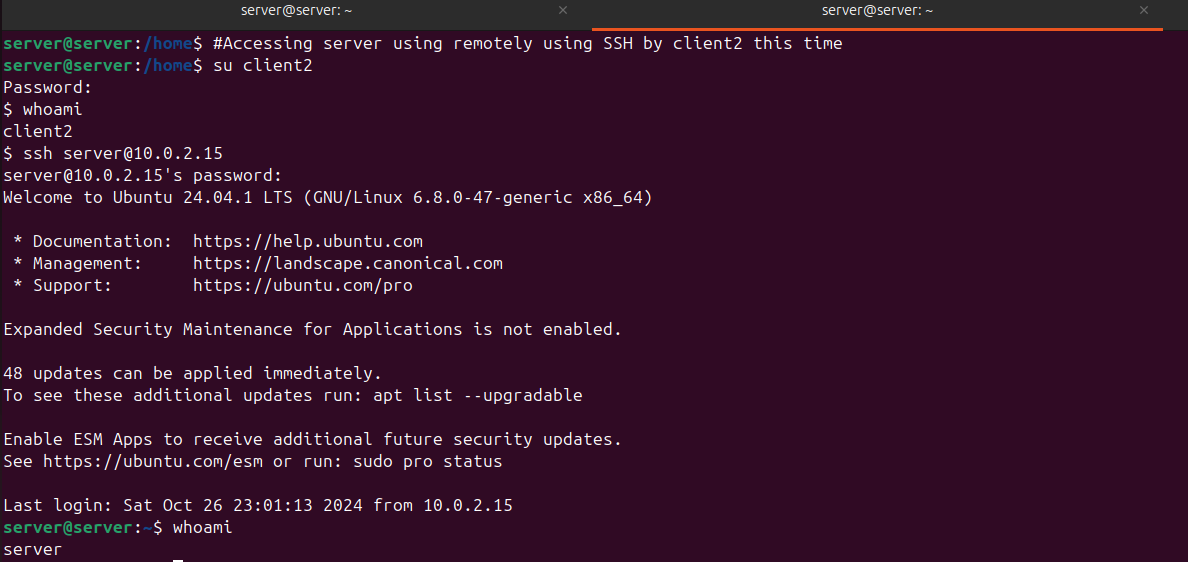


We managed to access the server using SSH (remotely) by client1 user:

A black screen with white text

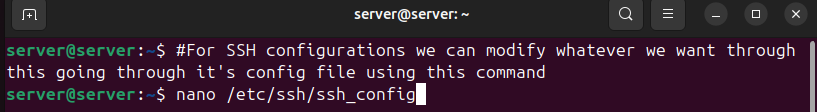
Description automatically generated

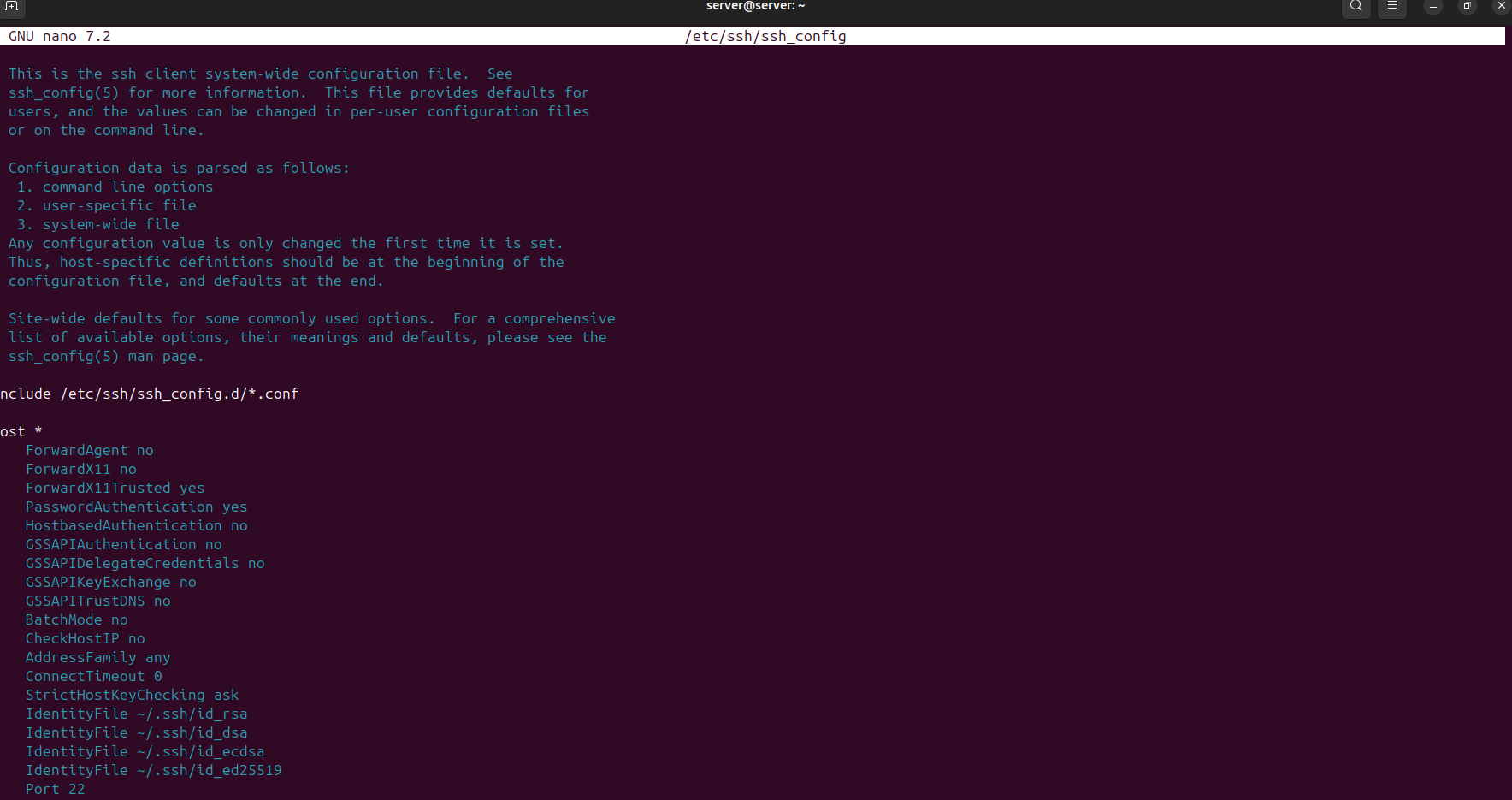
-We deduce that SSH is enabled successfully for client 1, we now will do the same for client2:



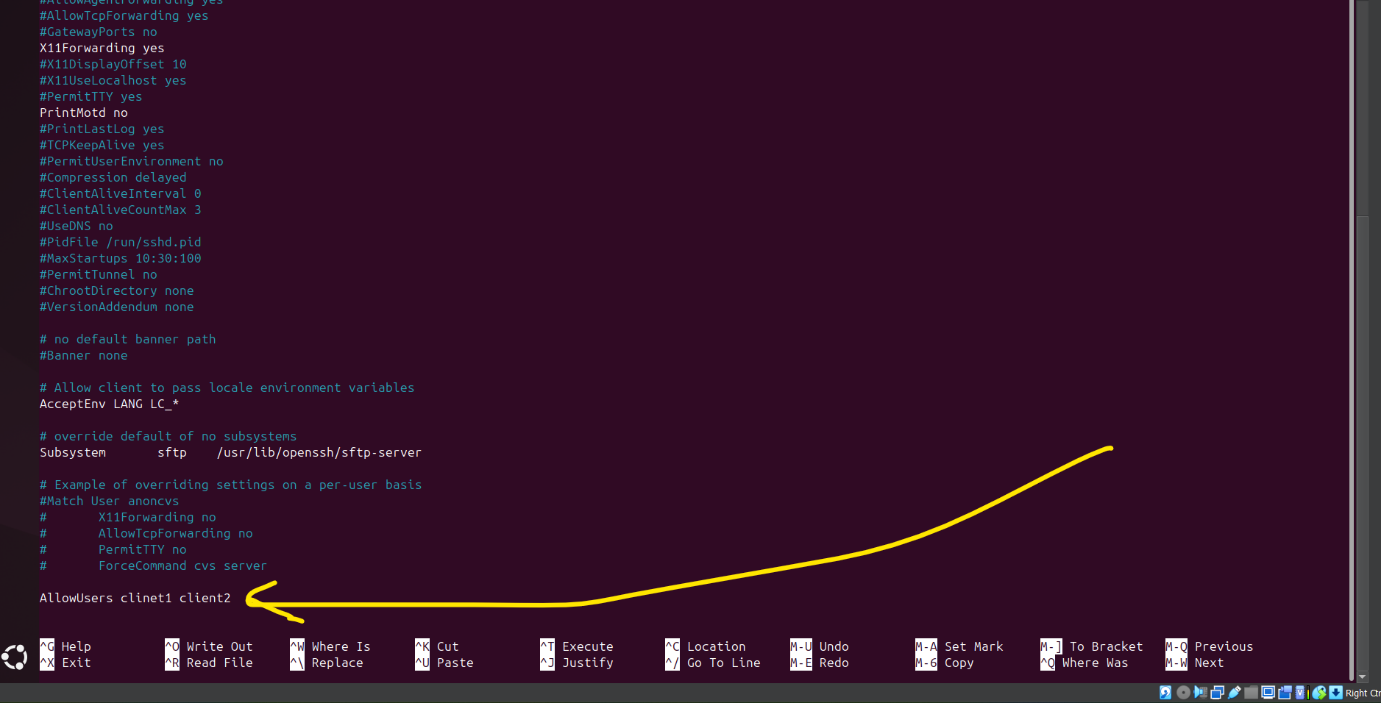
-We deduce that SSH is enabled successfully for client 2

-Mostly the configuration of SSH is done however:



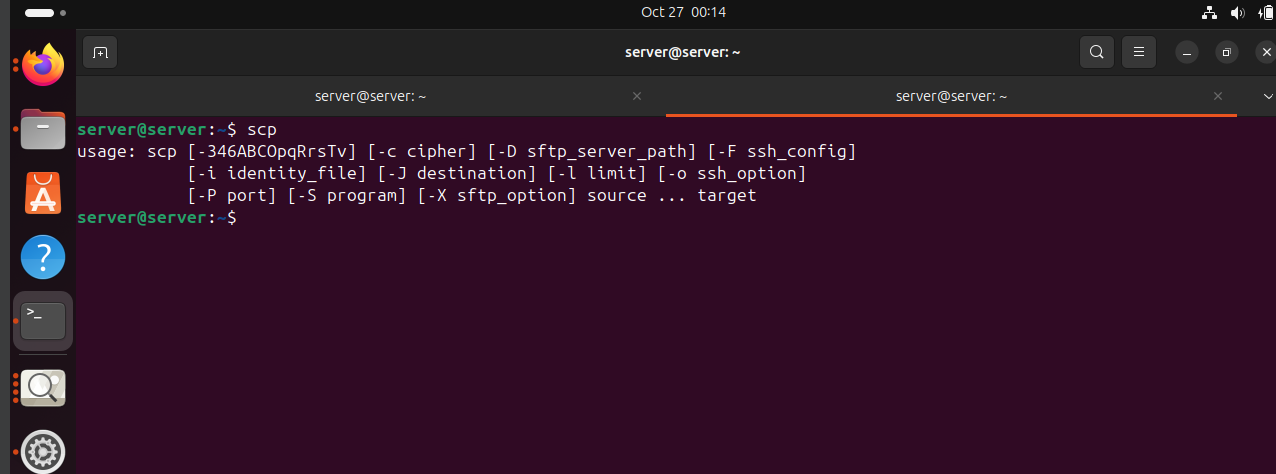


-We can enable or disable based on our needs using: sudo nano /etc/ssh/ssh\_config.



-We add this option for it to only allow the users: client1, client2

-Checking the SCP is working



**Task 3: Shell Scripting**

Network.sh: purpose is testing connectivity from server to clients. (using pin)

#!/bin/bash

# Function to install network tools if they are not already installed

install\_network\_tools() {

# Checking if ping and traceroute commands are available

if ! command -v ping &> /dev/null || ! command -v traceroute &> /dev/null; then

echo "🚀 Installing necessary network tools... Please wait!"

# Installing network tools (ping, traceroute)

sudo apt update && sudo apt install -y iputils-ping traceroute

echo "🎉 Network tools installed successfully! 🚀 Ready to check connectivity."

else

echo "✅ Network tools already installed and ready!"

fi

}

# Function to log message with date and time format

log\_message() {

local message=$1

# Getting the current date and time with nanoseconds for uniqueness

local timestamp=$(date "+%Y-%m-%d %H:%M:%S:%N")

# Adding a unique identifier with hostname and process ID

local unique\_id=$(hostname)\_$$

echo "$timestamp [$unique\_id] - $message" | tee -a network.log

}

# Function to check connectivity by pinging the target IP with a retry mechanism

check\_connectivity() {

local target\_ip=$1

local retries=3

local delay=2

for ((i=1; i<=retries; i++)); do

log\_message "Attempting to ping $target\_ip... Attempt $i"

if ping -c 1 -W 5 "$target\_ip" &> /dev/null; then

log\_message "Connectivity with $target\_ip is ok."

return 0

else

log\_message "Attempt $i failed for $target\_ip. Retrying in $((delay \* i)) seconds..."

sleep $((delay \* i))

fi

done

log\_message "Failed to reach $target\_ip after $retries attempts. Running traceroute..."

run\_traceroute "$target\_ip"

}

# Function to run traceroute if ping fails and log output

run\_traceroute() {

local target\_ip=$1

log\_message "Running traceroute for $target\_ip..."

{

echo "Routing Table:"

route -n

echo "Hostname: $(hostname)"

echo "Testing DNS resolution:"

nslookup google.com

echo "Tracing route to Google:"

traceroute google.com

echo "Pinging Google for verification:"

ping -c 3 google.com

} | tee -a network.log

# Reboot the machine if connectivity check fails

log\_message "Rebooting machine due to connectivity failure."

sudo reboot

}

# Function to generate a custom summary report at the end

summarize\_results() {

echo "📊 Summary Report 📊" | tee -a network.log

echo "Total Checks Performed: $total\_checks" | tee -a network.log

echo "Successful Connections: $successful\_checks" | tee -a network.log

echo "Failed Connections: $failed\_checks" | tee -a network.log

}

# Function to detect OS type for compatibility message

detect\_os\_type() {

if [[ "$OSTYPE" == "linux-gnu"\* ]]; then

log\_message "Detected Linux OS."

elif [[ "$OSTYPE" == "darwin"\* ]]; then

log\_message "Detected macOS. Traceroute may differ."

else

log\_message "Unknown OS type. Please verify compatibility."

fi

}

# Main script execution

main() {

detect\_os\_type

install\_network\_tools

# Initialize counters for a unique summary report

total\_checks=0

successful\_checks=0

failed\_checks=0

# Check if IPs are provided as arguments

if [ $# -lt 1 ]; then

exit\_with\_code 99 "Error: No IPs provided. Exiting with custom code 99."

fi

# Loop through each IP argument and test connectivity

for target\_ip in "$@"; do

total\_checks=$((total\_checks + 1))

if check\_connectivity "$target\_ip"; then

successful\_checks=$((successful\_checks + 1))

else

failed\_checks=$((failed\_checks + 1))

fi

done

# Call the summarize\_results function at the end

summarize\_results

}

# Execute the main function with all command-line arguments passed to the script

main "$@"

**Traceroute.sh script**

run\_traceroute() {

local target\_ip=$1

log\_message "Running traceroute for $target\_ip..."

{

echo "Routing Table:"

route -n

echo "Hostname: $(hostname)"

echo "Testing DNS resolution:"

nslookup google.com

echo "Tracing route to Google:"

traceroute google.com

echo "Pinging Google for verification:"

ping -c 3 google.com

} | tee -a network.log

log\_message "Rebooting machine due to connectivity failure."

sudo reboot

}