Name: Lujain Zia
Roll no: 2023-BSE-034
Date: November 7, 2025

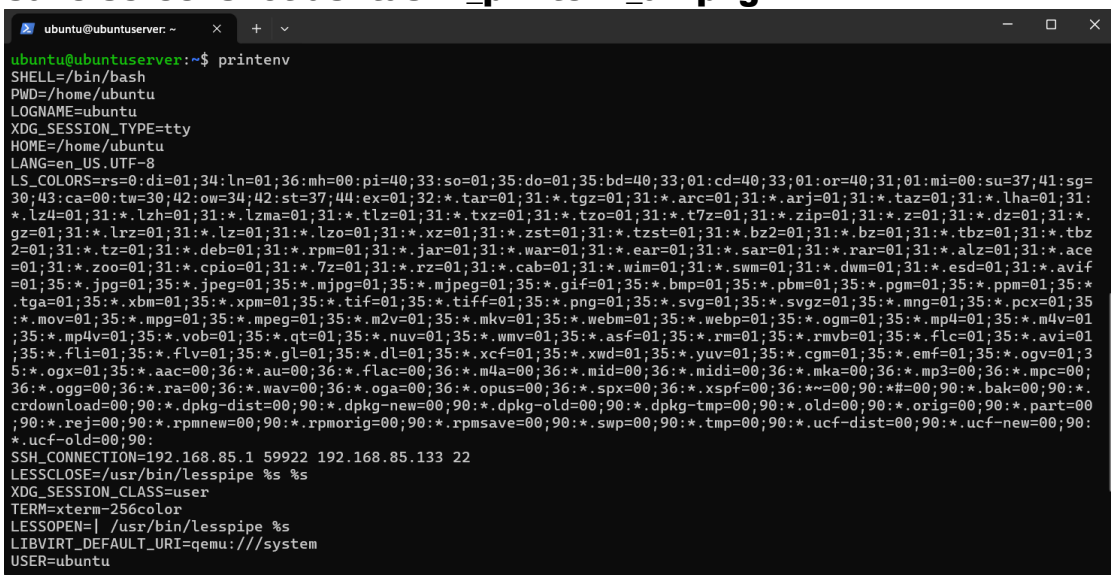
Lab 7

Task 1 — Print & filter environment variables

1. Print all environment variables:

printenv

Save screenshot as: task1_printenv_all.png



2. Filter for SHELL, HOME and USER — run these greps together and capture one combined screenshot:

printenv | grep SHELL
printenv | grep HOME
printenv | grep USER

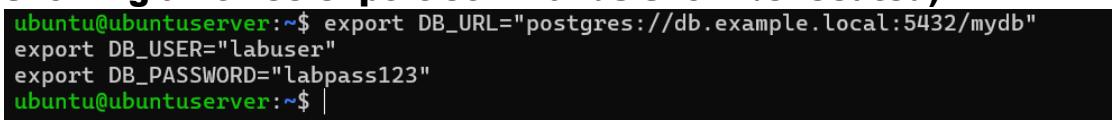
Save screenshot as: task1_grep_shell_home_user.png (single screenshot showing all three grep outputs together)

Task 2 — Export DB_* variables temporarily and observe scope

1. Define all DB_* variables (run the three exports one after another). Capture them in one screenshot showing the three export commands and their execution:

export DB_URL="postgres://db.example.local:5432/mydb"
export DB_USER="labuser"
export DB_PASSWORD="labpass123"

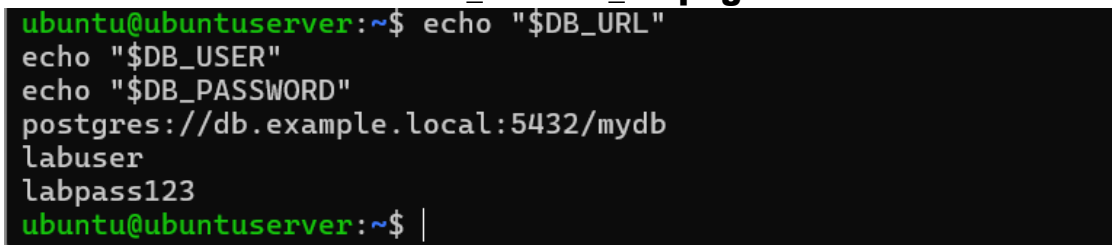
Save screenshot as: task2_exports_all.png (single screenshot showing all three export commands shown/executed)



2. Echo the three variables (run the three echo commands together) and capture one screenshot showing their outputs:

echo "$DB_URL"
echo "$DB_USER"
echo "$DB_PASSWORD"

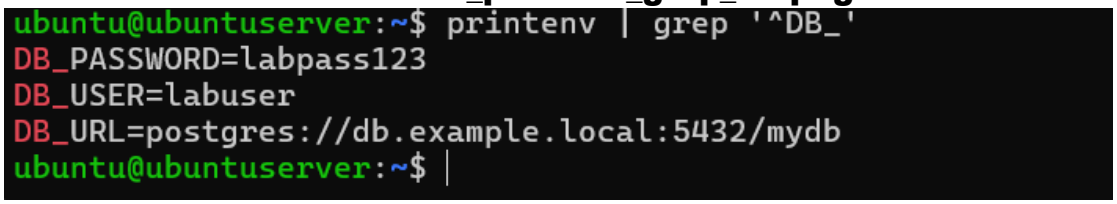
Save screenshot as: task2_echoes_all.png



3. Show all DB_ variables with a single grep command (capture that output):

printenv | grep '^DB_'

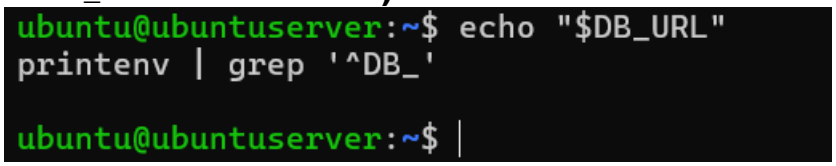
Save screenshot as: task2_printenv_grep_db.png



4. Close the bash session (e.g., exit) and reopen a new terminal. Verify the variables are gone by running the echo(s) and the grep together; capture both checks in one screenshot:

echo "$DB_URL"
printenv | grep '^DB_'

Save screenshot as: task2_after_restart_checks.png (single screenshot showing echo (empty) and printenv | grep '^DB_' with no results)



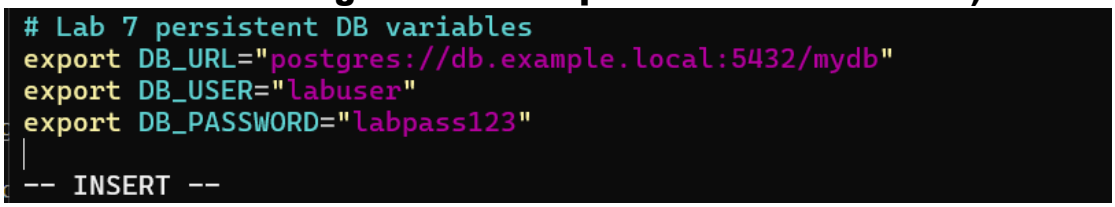
Task 3 — Make DB_* variables persistent in ~/.bashrc

Steps and required screenshots:

1. Open ~/.bashrc in an editor and append the three export lines. Capture the editor showing the three lines added (single screenshot):

vim ~/.bashrc
add at the end
Lab 7 persistent DB variables
export DB_URL="postgres://db.example.local:5432/mydb"
export DB_USER="labuser"
export DB_PASSWORD="labpass123"

Save screenshot as: task3_bashrc_added.png (single screenshot showing the three export lines in the editor)



2. Source ~/.bashrc and capture the source command in one screenshot together with the next verification commands (grouped): run source ~/.bashrc and then immediately run the three echoes and a single grep, capturing all of these in one screenshot:

source ~/.bashrcecho
"$DB_URL"
echo "$DB_USER"
echo "$DB_PASSWORD"
printenv | grep '^DB_'

Save screenshot as: task3_source_and_verification.png (single screenshot showing source, the three echoes, and the grep output)

3. Close and reopen terminal. Verify persistence by running one echo and the grep together — capture both in one screenshot:

echo "$DB_URL"
printenv | grep '^DB_'

Save screenshot as: task3_after_restart_persistent.png (single screenshot showing echo with value and grep output listing DB_ variables)

```
ubuntu@ubuntuserver:~$ echo "$DB_URL"
printenv | grep '^DB_'
postgres://db.example.local:5432/mydb
DB_PASSWORD=labpass123
DB_USER=labuser
DB_URL=postgres://db.example.local:5432/mydb
ubuntu@ubuntuserver:~$
```

Task 4 — System-wide environment variable, welcome script, and PATH

1. View /etc/environment:

sudo cat /etc/environment

Save screenshot as: task4_etc_environment_before.png

```
ubuntu@ubuntuserver:~$ sudo cat /etc/environment
[sudo] password for ubuntu:
Sorry, try again.
[sudo] password for ubuntu:
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin"
ubuntu@ubuntuserver:~$
```

2. Show current PATH:

echo "$PATH"

Save screenshot as: task4_echo_path_before.png

```
ubuntu@ubuntuserver:~$ echo "$PATH"
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
ubuntu@ubuntuserver:~$
```

3. Edit /etc/environment and add Class:

sudo vim /etc/environment
# add line: Class="CC-<your_class_name>"

Save screenshot as: task4_etc_environment_edit_vim.png (editor with edit)

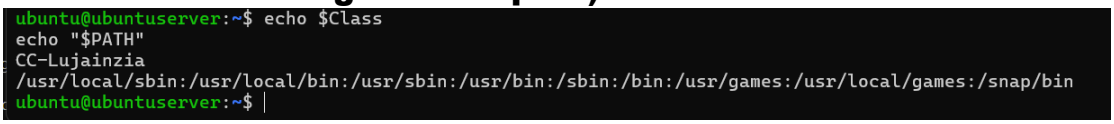
Save screenshot as: task4_etc_environment_after.png (cat or editor view showing the new Class line)

```
ubuntu@ubuntuserver: ~
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin"
Class="CC-Lujainzia"
~
~
~
~
ubuntu@ubuntuserver:~$ cat /etc/environment
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin"
Class="CC-Lujainzia"
ubuntu@ubuntuserver:~$
```

4. Re-login or open a new shell and show Class and PATH together (grouped prints): run echo $Class and echo $PATH together and capture in a single screenshot:

echo $Class
echo "$PATH"

Save screenshot as: task4_echo_class_and_path.png (single screenshot showing both outputs)

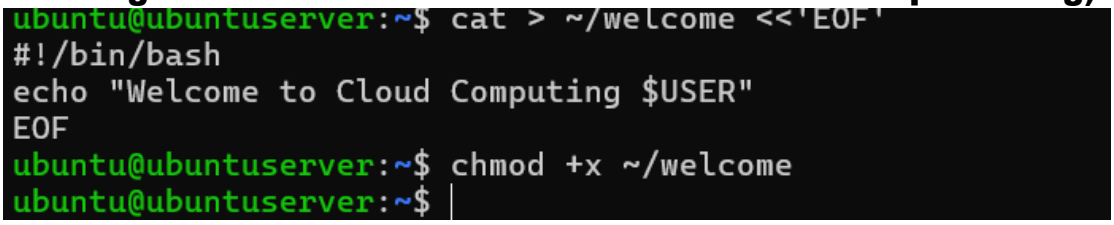


5. Create welcome script at your home directory (~/welcome) and make it executable (capture the heredoc creation and chmod together in one screenshot if possible):

cat > ~/welcome <<'EOF
'#!/bin/bash
echo "Welcome to Cloud Computing $USER"
EOF

chmod +x ~/welcome

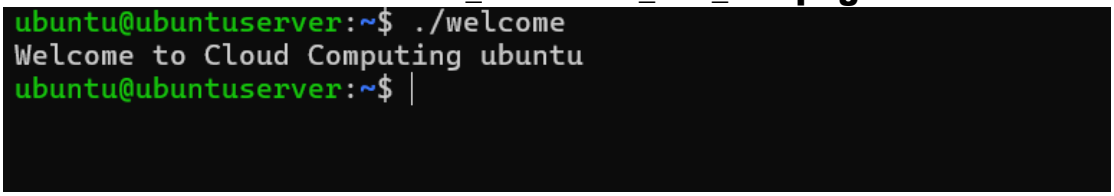
Save screenshot as: task4_welcome_create_and_chmod.png (single screenshot showing heredoc creation command and chmod output/listing)



6. Run the script from your home directory using ./welcome:

cd ~ ./welcome

Save screenshot as: task4_welcome_run_dot.png



7. Add your home directory to PATH in ~/.bashrc. NOTE: per your instruction we do not include an export PATH line here — only add the PATH modification line in the file. Capture the editor showing that PATH line in one screenshot:

vim ~/.bashrc
# add at end:
PATH=$PATH:~

Save screenshot as: task4_bashrc_path_line.png (editor screenshot showing the PATH line only)

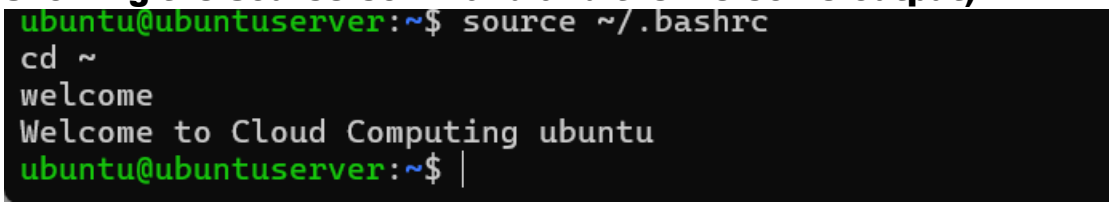
8. Apply the change and run welcome — capture these runtime commands in a separate screenshot (must be taken separately from the editor screenshot):

source ~/.bashrccd ~

welcome

Save screenshot

as: task4_bashrc_source_and_welcome.png (single screenshot showing the source command and the welcome output)

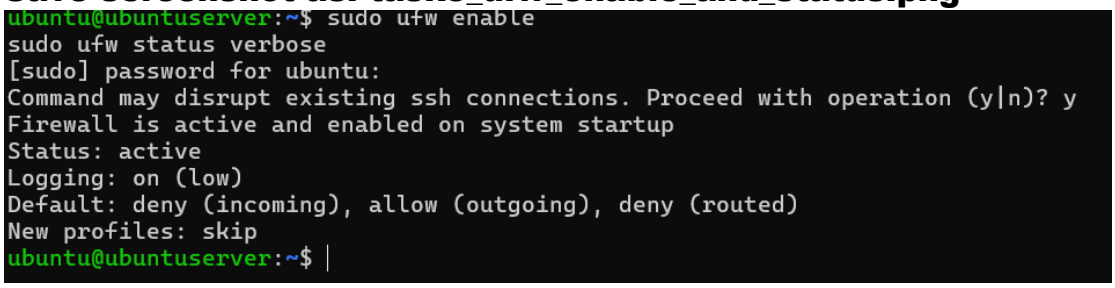


Task 5 — Block and allow SSH using ufw (firewall)

1.Enable ufw and show status (group both commands in one screenshot if you run them together):

sudo ufw enable
sudo ufw status verbose

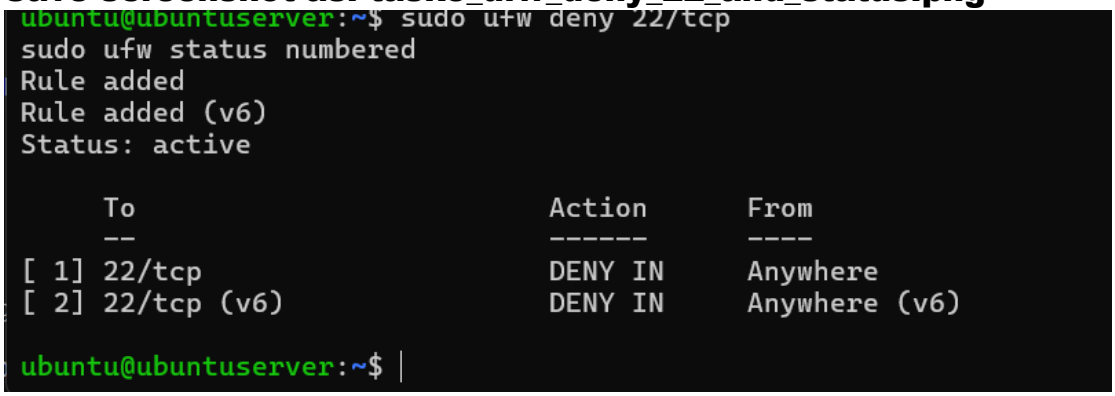
Save screenshot as: task5_ufw_enable_and_status.png



2.Deny TCP port 22 and show status (run deny and status numbered together and capture in one screenshot). Use short form as requested:

sudo ufw deny 22/tcp
sudo ufw status numbered

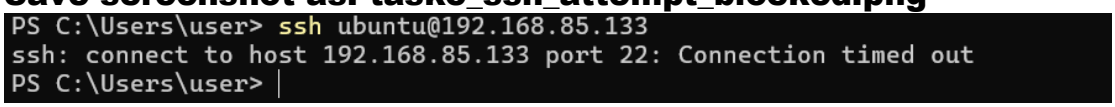
Save screenshot as: task5_ufw_deny_22_and_status.png



3.From Windows host attempt to SSH (expected to fail) — capture the host-side SSH attempt in one screenshot:

ssh username@<server_ip>

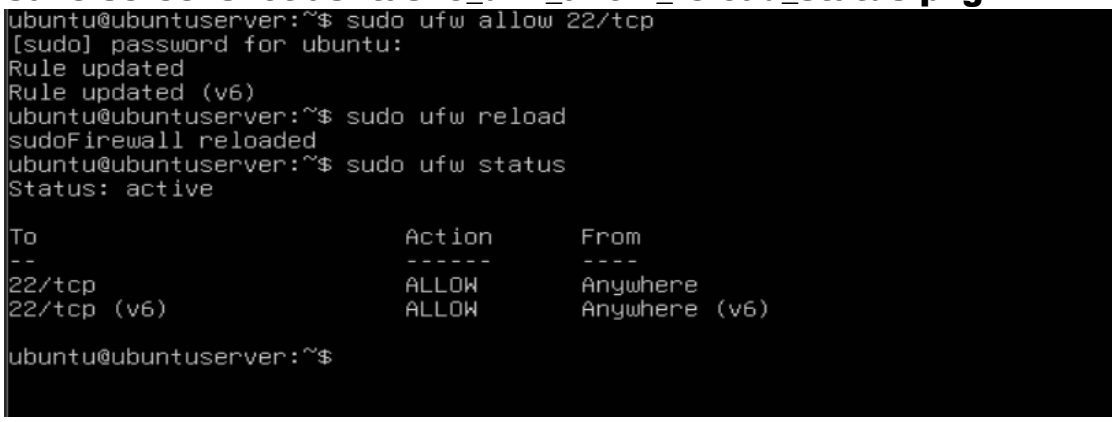
Save screenshot as: task5_ssh_attempt_blocked.png



4.Allow SSH back and reload, then show status (group allow, reload, status in one screenshot if run together). Use short form as requested:

sudo ufw allow 22/tcp
sudo ufw reload
sudo ufw status

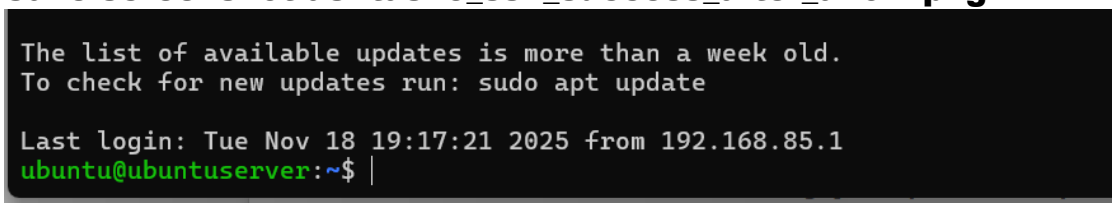
Save screenshot as: task5_ufw_allow_reload_status.png



5.From Windows host attempt SSH again (should succeed) — capture successful login in one screenshot:

ssh username@<server_ip>

Save screenshot as: task5_ssh_success_after_allow.png



Task 6 — Configure SSH key-based login from Windows host

A. On Windows host (client) — group related client actions:

1. Generate ed25519 key pair (if needed) and show the generated files in one screenshot (run ssh-keygen and then list ~/.ssh):

ssh-keygen -t ed25519 -f ~/.ssh/id_lab7 -C "lab_key"
ls -la ~/.ssh

Save screenshot as: task6_windows_sshkey_and_list.png (single screenshot showing keygen result and ls of .ssh folder)

```
PS C:\Users\user> ssh-keygen -t ed25519 -f $env:USERPROFILE\.ssh\id_lab7 -C "lab_key"
Generating public/private ed25519 key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\user\.ssh\id_lab7
Your public key has been saved in C:\Users\user\.ssh\id_lab7.pub
The key fingerprint is:
SHA256:7CXJFuvwBfBespiXIxV2kMt7fDGSlbqivON93Z15Vls lab_key
The key's randomart image is:
+--[ED25519 256]--+
|       . +o.  .   |
|        +.o  o    |
|        .*..+     |
|        BoX+ o    |
|       = Sooo o   |
|        Oo=+ .   E|
|      . .+o o . .*|
|       +.  . . .++|
|     .oo..     ..|
+----[SHA256]-----+
PS C:\Users\user>
```

2. Show the public key content (single screenshot):

type $env:USERPROFILE\.ssh\id_lab7.pub# or
on Git Bash: cat ~/.ssh/id_lab7.pub

Save screenshot as: task6_windows_public_key.png

```
PS C:\Users\user> type $env:USERPROFILE\.ssh\id_lab7.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBBD3cRHs81Zm4OlYL8crIJwRT6gjbHWV9t6lJJtGRNn lab_key
PS C:\Users\user>
```

3. Clear the known_hosts file content and verify it is empty (single screenshot):

Clear contents (PowerShell)Clear-Content
$env:USERPROFILE\.ssh\known_hosts
View the file (should be empty)
type $env:USERPROFILE\.ssh\known_hosts

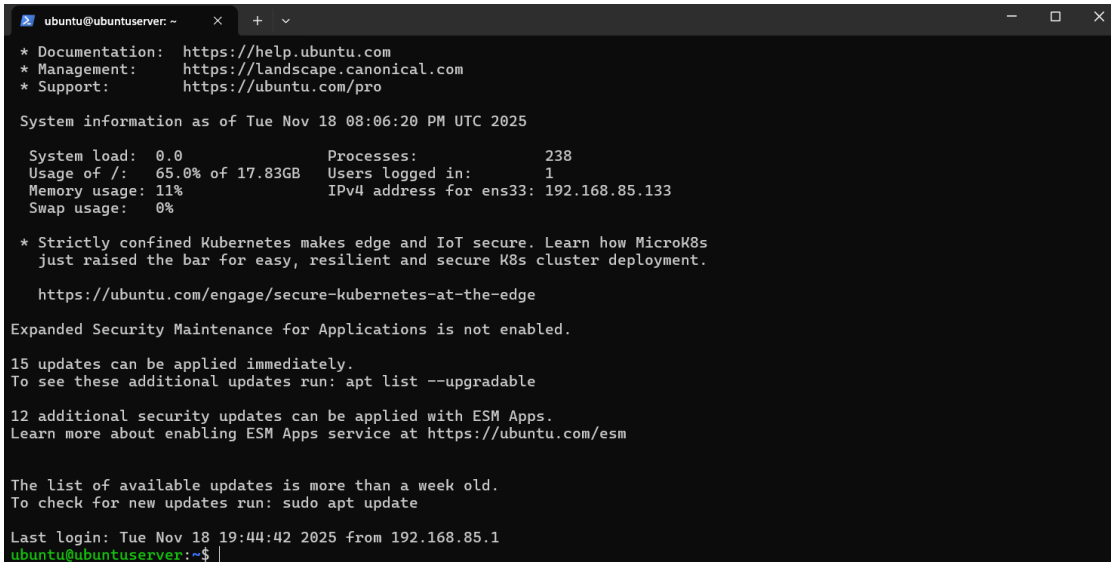
Save screenshot as: task6_windows_known_hosts_cleared_and_empty.png

```
PS C:\Users\user> Clear-Content $env:USERPROFILE\.ssh\known_hosts
PS C:\Users\user> type $env:USERPROFILE\.ssh\known_hosts
PS C:\Users\user>
```

4. Connect to the Ubuntu server using the standard SSH command (this will prompt to accept the server host key because known_hosts is empty). Capture the connection prompt/accept step in one screenshot:

ssh username@<server_ip># Accept the host key prompt (yes) and complete the login (enter password or key passphrase)

Save screenshot as: task6_windows_ssh_accept_hostkey_and_login.png

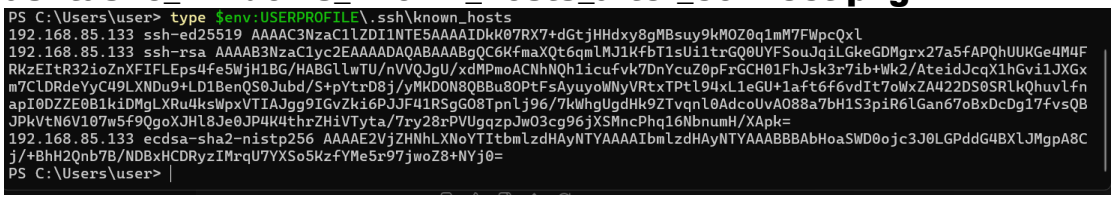


5. After the successful connection, view the known_hosts file to show the server host key was added (single screenshot):

type $env:USERPROFILE\.ssh\known_hosts

Save screenshot

as: task6_windows_known_hosts_after_connect.png



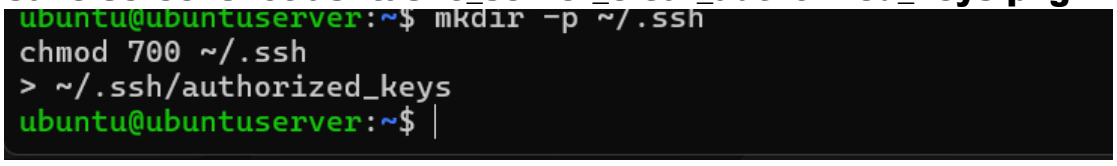
B. On Ubuntu server — group related server-side commands:

1. Prepare the ~/.ssh directory and clear authorized_keys (this will create the directory if missing, set the correct directory permissions, and truncate the authorized_keys file). Capture this command sequence and its output in one screenshot:

mkdir -p ~/.ssh

chmod 700 ~/.ssh> ~/.ssh/authorized_keys

Save screenshot as: task6_server_clear_authorized_keys.png



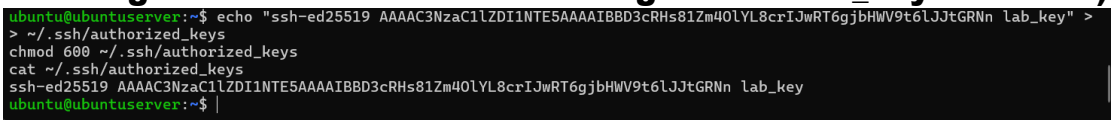
2.Append the public key, set file permissions, and show the resulting authorized_keys (capture commands and resulting file content in one screenshot):

paste public key name id_lab7.pub from Windows client into the echo belowecho "ssh-ed25519 AAAA... yourpublickey ... comment" >> ~/.ssh/authorized_keys

chmod 600 ~/.ssh/authorized_keys

cat ~/.ssh/authorized_keys

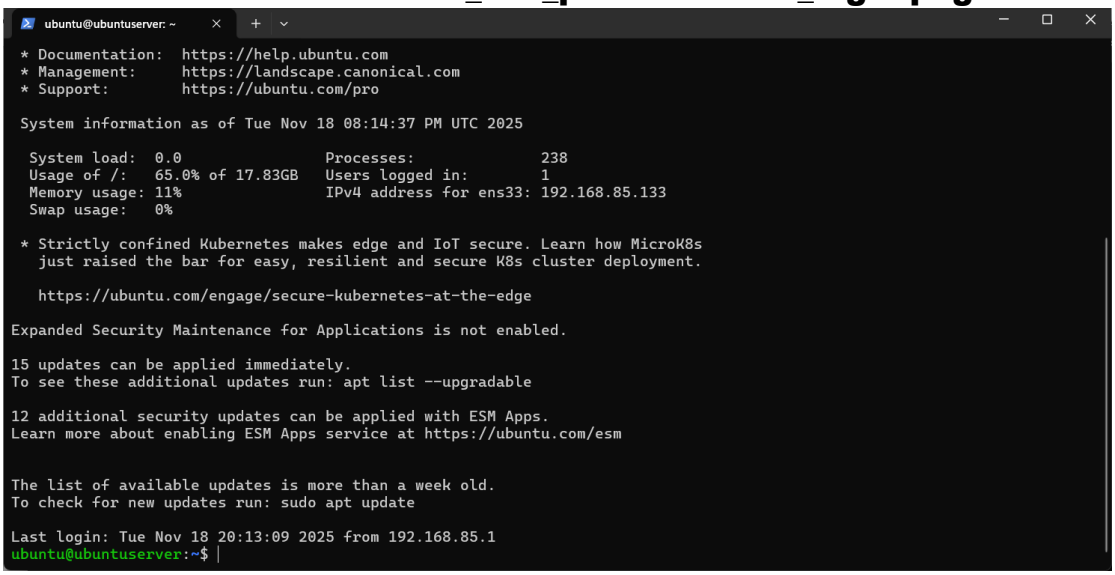
Save screenshot as: task6_server_add_key_and_show.png (single screenshot showing the commands and resulting authorized_keys content)



3. From Windows host test passwordless login (capture successful login in one screenshot):

ssh username@<server_ip>

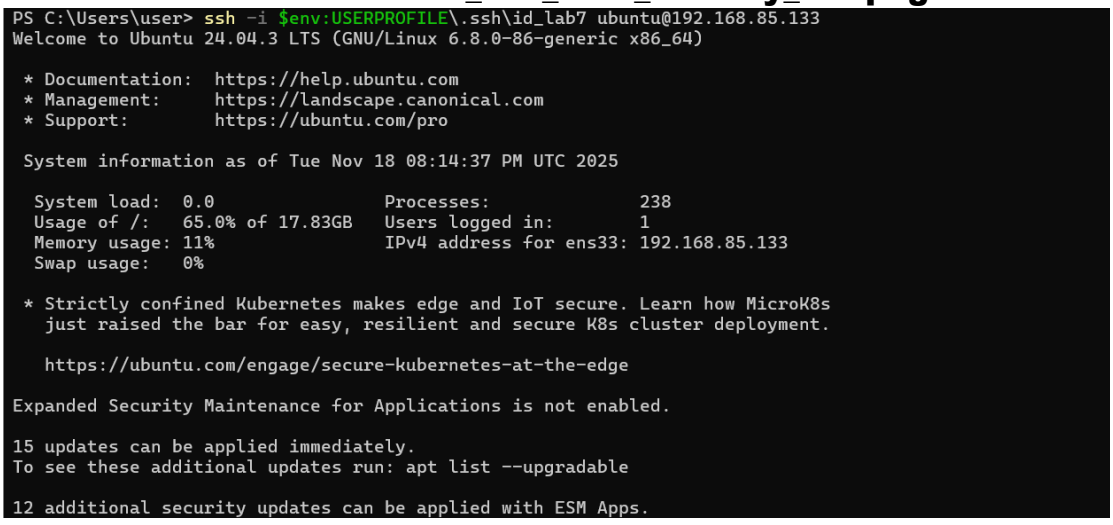
Save screenshot as: task6_ssh_passwordless_login.png



4. Also demonstrate explicit identity usage (single screenshot):

ssh -i ~/.ssh/id_lab7 username@<server_ip>

Save screenshot as: task6_ssh_with_identity_file.png



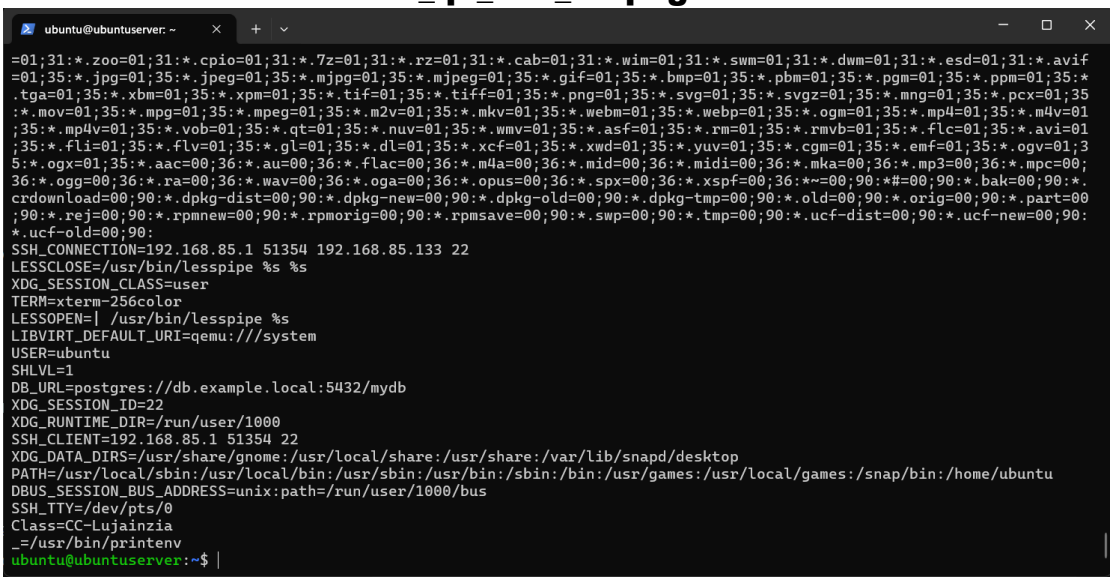
Exam Evaluation Questions

Q1: Quick Environment Audit

Objective: Demonstrate you can inspect the current environment and extract a few key variables.

Actions & evidence:

- **Run a single command to display environment variables and capture its output.**
- **Save screenshot: EE_q1_env_all.png**



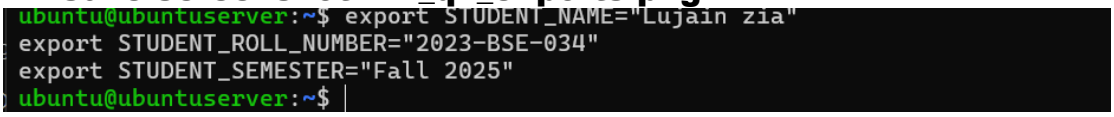
- **In the same terminal session, run three filters (one per line) to show values for PATH, LANG, and PWD, then capture a single screenshot showing the three outputs together.**
- **Save screenshot: EE_q1_env_filters.png**

Q2: Short-lived Student Info

Objective: Show how temporary environment variables behave (session-scoped).

Actions & evidence:

- **In one terminal, set three variables (STUDENT_NAME, STUDENT_ROLL_NUMBER, STUDENT_SEMESTER) using export — execute all three consecutively and capture them in one screenshot (show the commands executed).**
- **Save screenshot: EE_q2_exports.png**



- **Still in the same session, print the three values with echo (grouped) and capture the outputs in one screenshot.**
- **Save screenshot: EE_q2_echoes.png**

```
ubuntu@ubuntuserver:~$ echo $STUDENT_NAME
echo $STUDENT_ROLL_NUMBER
echo $STUDENT_SEMESTER
Lujain zia
2023-BSE-034
Fall 2025
ubuntu@ubuntuserver:~$
```

- **Use a single printenv|grep command to list any STUDENT_ variables and capture the result.**
- **Save screenshot: EE_q2_printenv_grep.png**

```
ubuntu@ubuntuserver:~$ printenv | grep '^STUDENT_'
STUDENT_NAME=Lujain zia
STUDENT_SEMESTER=Fall 2025
STUDENT_ROLL_NUMBER=2023-BSE-034
ubuntu@ubuntuserver:~$
```

- **Exit that shell, open a fresh terminal, and show that the STUDENT_ variables are not set (use echo and printenv|grep together) — capture in one screenshot.**
- **Save screenshot: EE_q2_after_restart.png**

```
ubuntu@ubuntuserver:~$ echo $STUDENT_NAME
printenv | grep '^STUDENT_'

ubuntu@ubuntuserver:~$
```

Q3: Make It Sticky (Persistence Check for Student Info)

Objective: Demonstrate persistence of environment variables across sessions via shell configuration.
Actions & evidence:

- **Edit ~/.bashrc and append the three STUDENT_* exports. Capture a screenshot of the editor showing the new lines.**
- **Save screenshot: EE_q3_bashrc_editor.png**

```
# Persistent student info
export STUDENT_NAME="Lujain zia"
export STUDENT_ROLL_NUMBER="2023-BSE-034"
export STUDENT_SEMESTER="Fall 2025"

                                [ Wrote 127 lines ]
```

- **Reload your shell config with a single command and then verify the three variables and show printenv | grep '^STUDENT_' — capture these verification outputs together in one screenshot.**
- **Save screenshot: EE_q3_after_source.png**

```
ubuntu@ubuntuserver:~$ nano ~/.bashrc
ubuntu@ubuntuserver:~$ source ~/.bashrc
echo $STUDENT_NAME
echo $STUDENT_ROLL_NUMBER
echo $STUDENT_SEMESTER
printenv | grep '^STUDENT_'
Lujain zia
2023-BSE-034
Fall 2025
STUDENT_NAME=Lujain zia
STUDENT_SEMESTER=Fall 2025
STUDENT_ROLL_NUMBER=2023-BSE-034
ubuntu@ubuntuserver:~$
```

- **Close and re-open a terminal and demonstrate the STUDENT_NAME variable is available (echo and printenv grep together) — capture in one screenshot.**
- **Save screenshot: EE_q3_after_restart.png**

```
ubuntu@ubuntuserver:~$ echo $STUDENT_NAME
printenv | grep '^STUDENT_'
Lujain zia
STUDENT_NAME=Lujain zia
STUDENT_SEMESTER=Fall 2025
STUDENT_ROLL_NUMBER=2023-BSE-034
ubuntu@ubuntuserver:~$
```

Q4: Firewall Rules: Block and Restore Ping (ICMP)

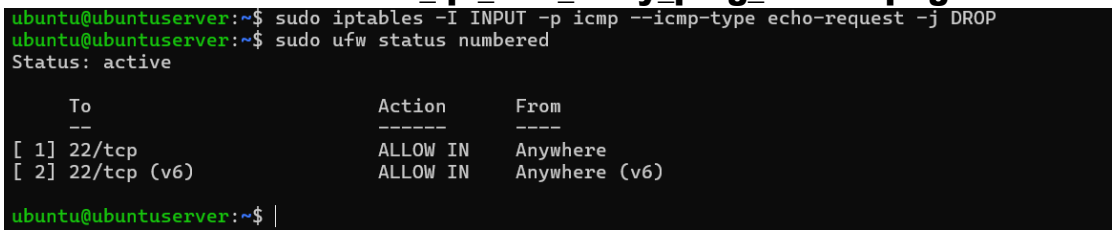
Objective: Demonstrate you can block ping (ICMP echo) traffic using ufw and then re-allow it; show effect from a client.
Actions & evidence:

- **Enable ufw and capture the enable command and status together in one screenshot.**
- **Save screenshot: EE_q5_ufw_enable_status.png**

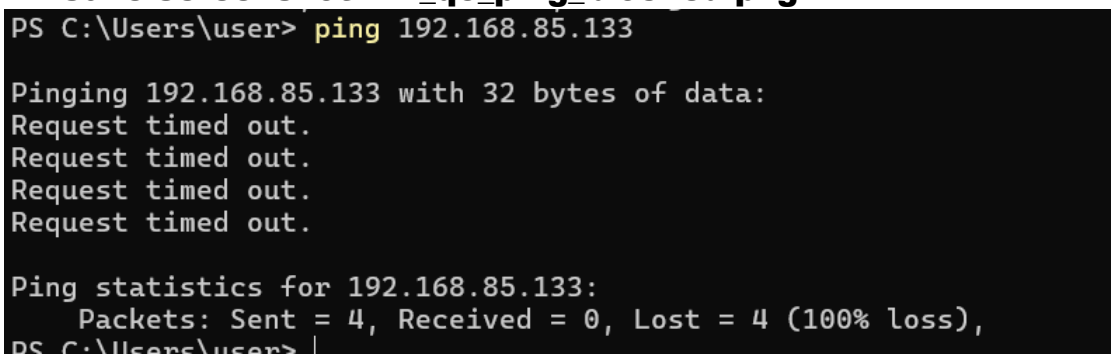
```
ubuntu@ubuntuserver:~$ sudo ufw enable
sudo ufw status verbose
[sudo] password for ubuntu:
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), deny (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW IN    Anywhere
22/tcp (v6)                ALLOW IN    Anywhere (v6)

ubuntu@ubuntuserver:~$
```

- **Add a rule to block ping (ICMP echo) and show ufw status numbered in the same screenshot.**
- **Suggested command example:**
- **sudo ufw deny proto icmp from any to any**
- **sudo ufw status numbered**

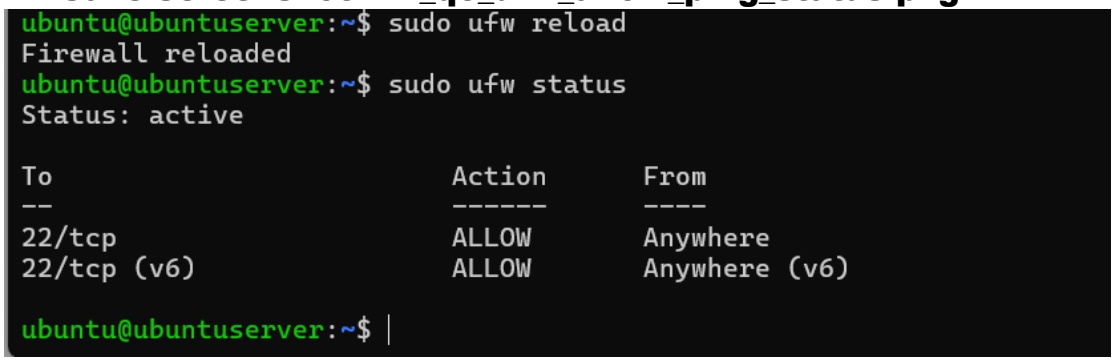
- **Save screenshot: EE_q5_ufw_deny_ping_status.png**



- **From your Windows host (or another client), attempt to ping the server while the rule is active and capture the blocked/failing ping in one screenshot.**
- **Save screenshot: EE_q5_ping_blocked.png**



- **Re-allow ping (ICMP) (or remove the deny rule) and capture the allow/reload/status sequence in one screenshot.**
  Suggested command example:
  sudo ufw allow proto icmp from any to any
  sudo ufw reload
  sudo ufw status
- **Save screenshot: EE_q5_ufw_allow_ping_status.png**



- **From the client, ping the server again and capture successful replies in one screenshot.**
- **Save screenshot: EE_q5_ping_success.png**

```
PS C:\Users\user> ping 192.168.85.133

Pinging 192.168.85.133 with 32 bytes of data:
Reply from 192.168.85.133: bytes=32 time<1ms TTL=64
Reply from 192.168.85.133: bytes=32 time<1ms TTL=64
Reply from 192.168.85.133: bytes=32 time=1ms TTL=64
Reply from 192.168.85.133: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.85.133:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
PS C:\Users\user>
```