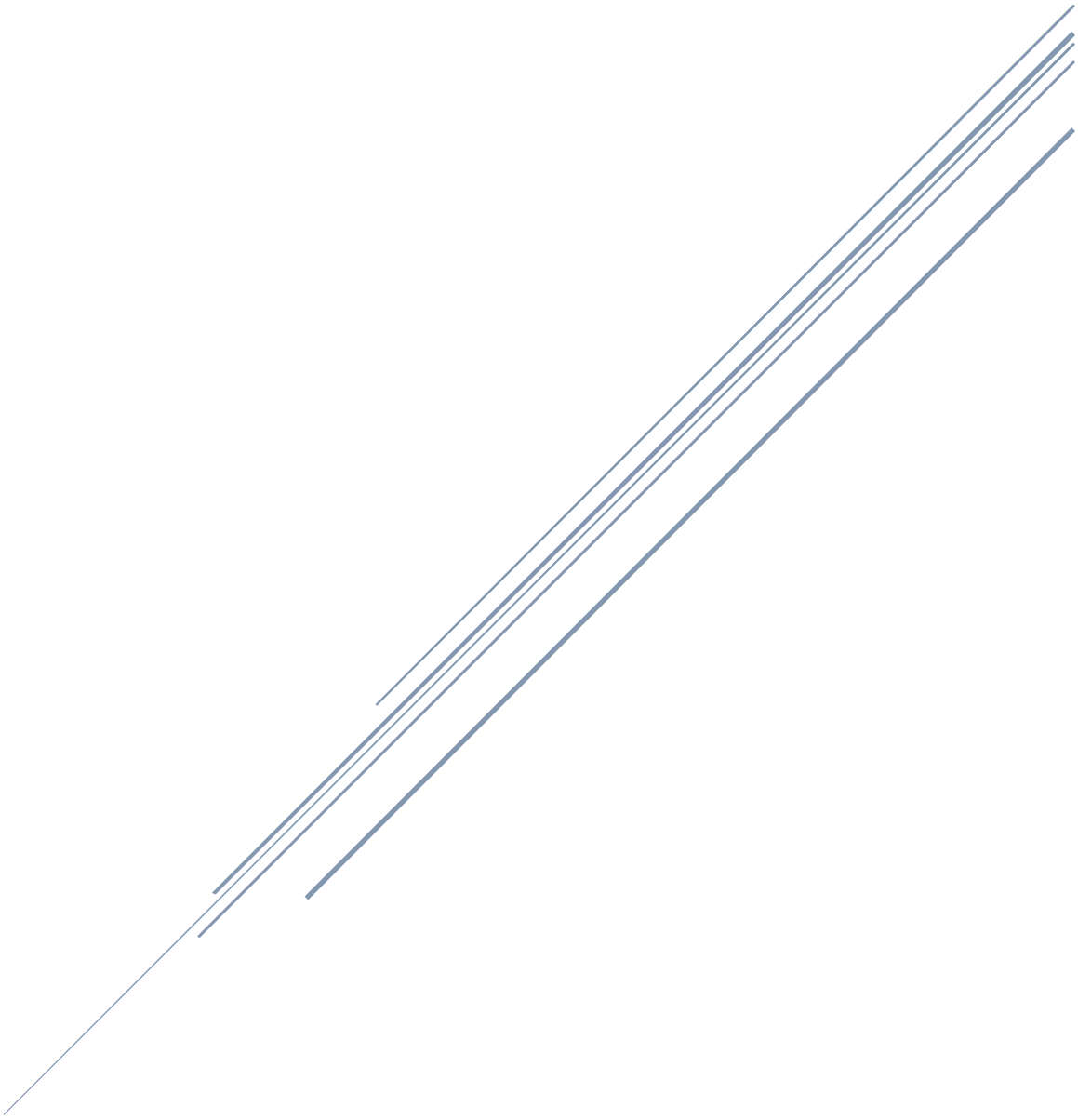
[DOCUMENT TITLE]

[Document subtitle]



Task 1 — Print & filter environment variables

Goal: Show environment variables and filter them using grep.

Commands and required screenshots (grouped as requested):

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)

```
abihanadeem001@abihanadeem:~$ printenv | grep SHELL
SHELL=/bin/bash
abihanadeem001@abihanadeem:~$ printenv | grep HOME
HOME=/home/abihanadeem001
abihanadeem001@abihanadeem:~$ printenv | grep USER
USER=abihanadeem001
abihanadeem001@abihanadeem:~$ _
```

Task 2 — Export DB_* variables temporarily and observe scope

Goal: Create env variables with export in the current shell, verify them, then close shell and show variables are gone.

Per the requested grouping rule: capture all the variable-definition (export) commands in a single screenshot; capture the echo/print checks grouped logically.

Steps and required screenshots:

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)

```
abihanadeem001@abihanadeem:~$ export DB_URL="postgres://db.example.local:5432/mydb"
abihanadeem001@abihanadeem:~$ export DB_USER="labuser"
abihanadeem001@abihanadeem:~$ export DB_PASSWORD="labpass123"
abihanadeem001@abihanadeem:~$
```

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

```
abihanadeem001@abihanadeem:~$ echo "$DB_USER"
labuser
abihanadeem001@abihanadeem:~$ echo "$DB_URL"
postgres://db.example.local:5432/mydb
abihanadeem001@abihanadeem:~$ echo "$DB_PASSWORD"
labpass123
abihanadeem001@abihanadeem:~$
```

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

printenv | grep '^DB_'

- Save screenshot as: task2_printenv_grep_db.png

```
abihanadeem001@abihanadeem:~$ printenv | grep '^DB_'
DB_PASSWORD=labpass123
DB_USER=labuser
DB_URL=postgres://db.example.local:5432/mydb
abihanadeem001@abihanadeem:~$
```

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)

```
abihanadeem001@abihanadeem:~$ echo "$DB_URL"

abihanadeem001@abihanadeem:~$ printenv | grep '^DB_'
abihanadeem001@abihanadeem:~$
```

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

Goal: Add DB_* variables to ~/.bashrc, reload, and verify persistence. Grouped captures: show the three export lines in ~/.bashrc together, and group the post-source checks into one screenshot.

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:

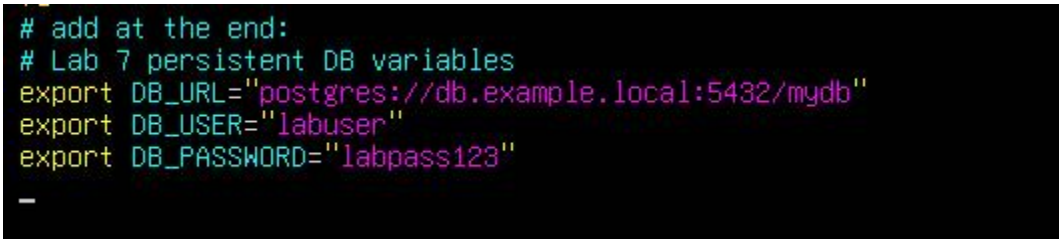
Lab 7 persistent DB variables

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 ~/.bashrc

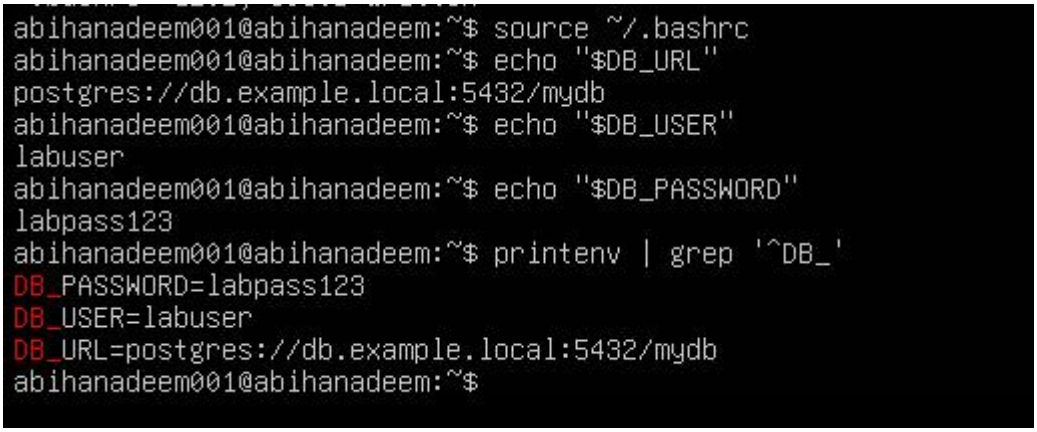
echo "$DB_URL"

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)

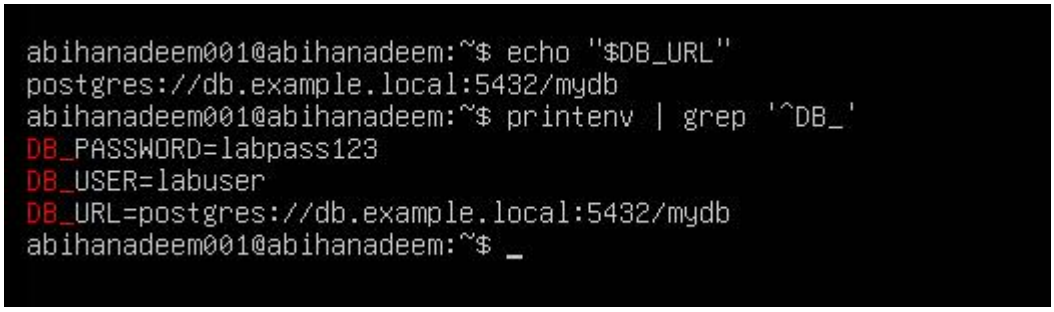


3. Close and reopen terminal. Verify persistence by running one echo and the grep together — capture both in one screenshot:

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)



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

Goal: Add Class variable to /etc/environment, view PATH, create a welcome script at ~/welcome, make it executable, and add PATH entry in ~/.bashrc so welcome can be executed without ./. Capture grouped screenshots as applicable.

Steps and required screenshots (grouping applies to "print with grep" type commands and grouped variable definitions — in this task there is a single system variable definition so a standard per-action capture is used):

1. View /etc/environment:

sudo cat /etc/environment

- Save screenshot as: task4_etc_environment_before.png

2. Show current PATH:

echo "$PATH"

- Save screenshot as: task4_echo_path_before.png

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)

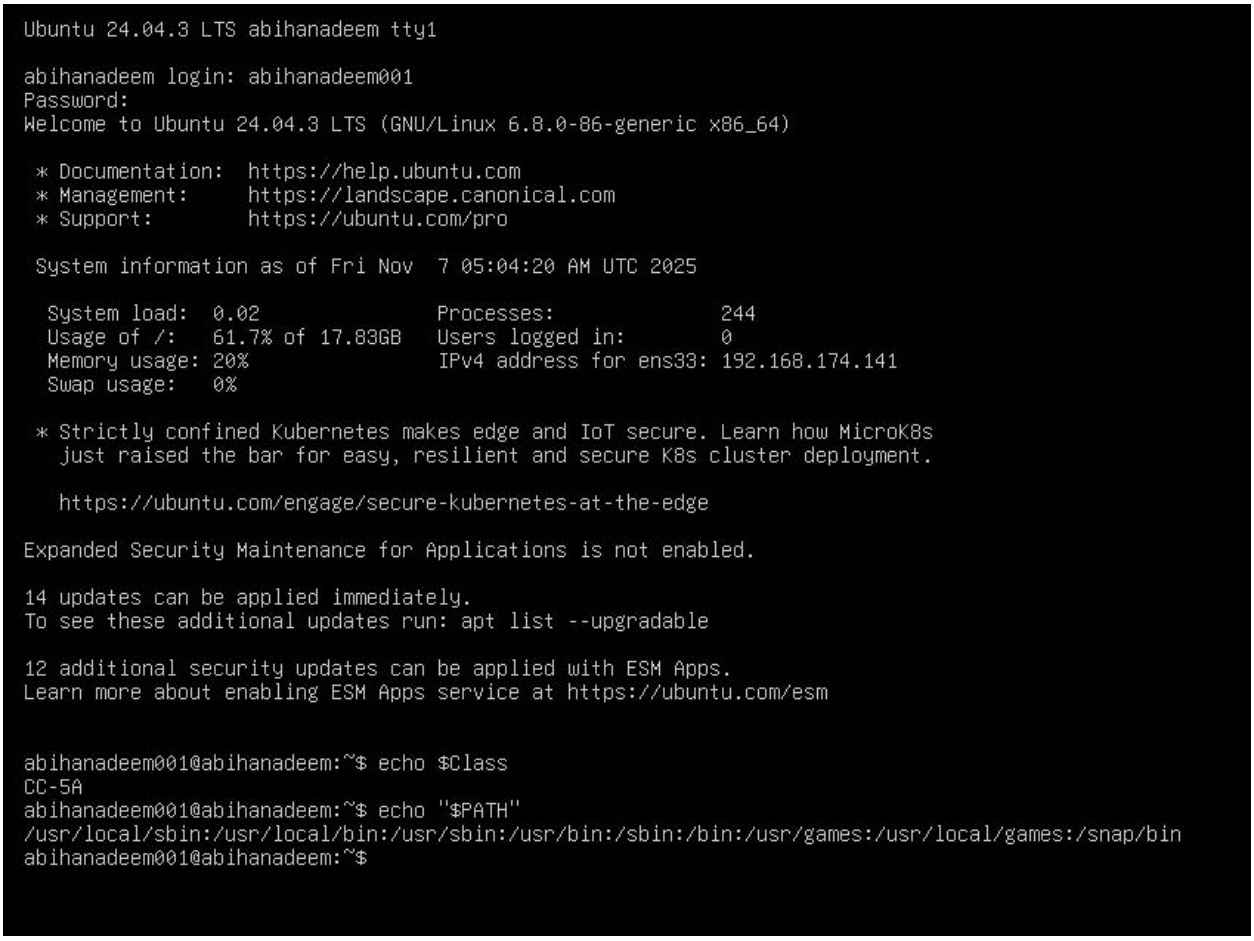
- Save screenshot as: task4_etc_environment_after.png (cat or editor view showing the new Class line)

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)

```
abihanadeem001@abihanadeem:~$ cat > ~/Welcome <<'EOF'
> #!/bin/bash
> echo "Welcome to CLOUD COMPUTING $USER"
> EOF
abihanadeem001@abihanadeem:~$ chmod +x ~/welcome
chmod: cannot access '/home/abihanadeem001/welcome': No such file or directory
abihanadeem001@abihanadeem:~$ chmod +x ~/Welcome
abihanadeem001@abihanadeem:~$ _
```

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

cd ~

./welcome

- Save screenshot as: task4_welcome_run_dot.png

```
abihanadeem001@abihanadeem:~$ cd ~
abihanadeem001@abihanadeem:~$ ./Welcome
Welcome to CLOUD COMPUTING abihanadeem001
abihanadeem001@abihanadeem:~$ _
```

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)

```
PATH=$PATH:~_
-- INSERT --
```

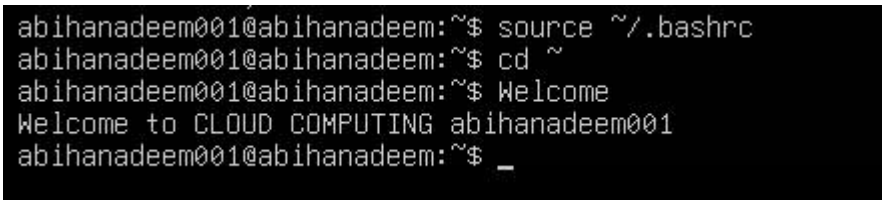
8. Apply the change and run welcome — capture these runtime commands in a separate screenshot (must be taken separately from the editor screenshot):

source ~/.bashrc

cd ~

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)

Goal: Use ufw to deny and allow SSH then verify SSH connectivity changes from host. Save screenshots after each logical command/step; group related print checks when appropriate.

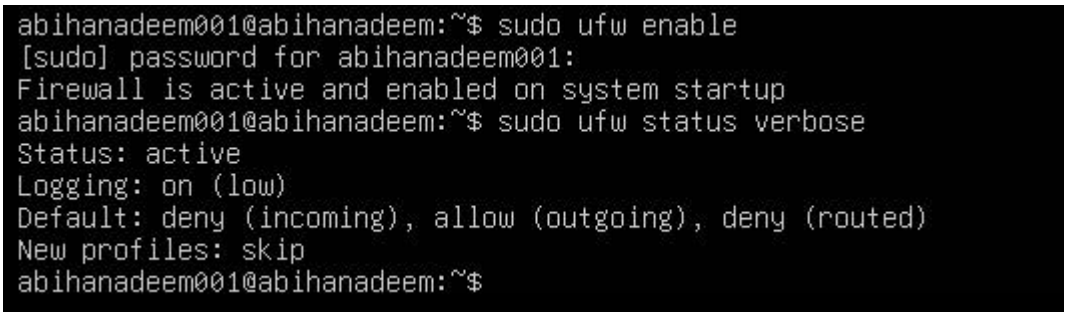
Steps and required screenshots:

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

```
abihanadeem001@abihanadeem:~$ sudo ufw deny 22/tcp
Rule added
Rule added (v6)
abihanadeem001@abihanadeem:~$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22/tcp                     DENY IN     Anywhere
[ 2] 22/tcp (v6)                DENY IN     Anywhere (v6)

abihanadeem001@abihanadeem:~$ _
```

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

```
Command Prompt
Microsoft Windows [Version 10.0.19045.6456]
(c) Microsoft Corporation. All rights reserved.

C:\Users\MOHSIN>ssh abihanadeem001@192.168.174.141
ssh: connect to host 192.168.174.141 port 22: Connection timed out

C:\Users\MOHSIN>_
```

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

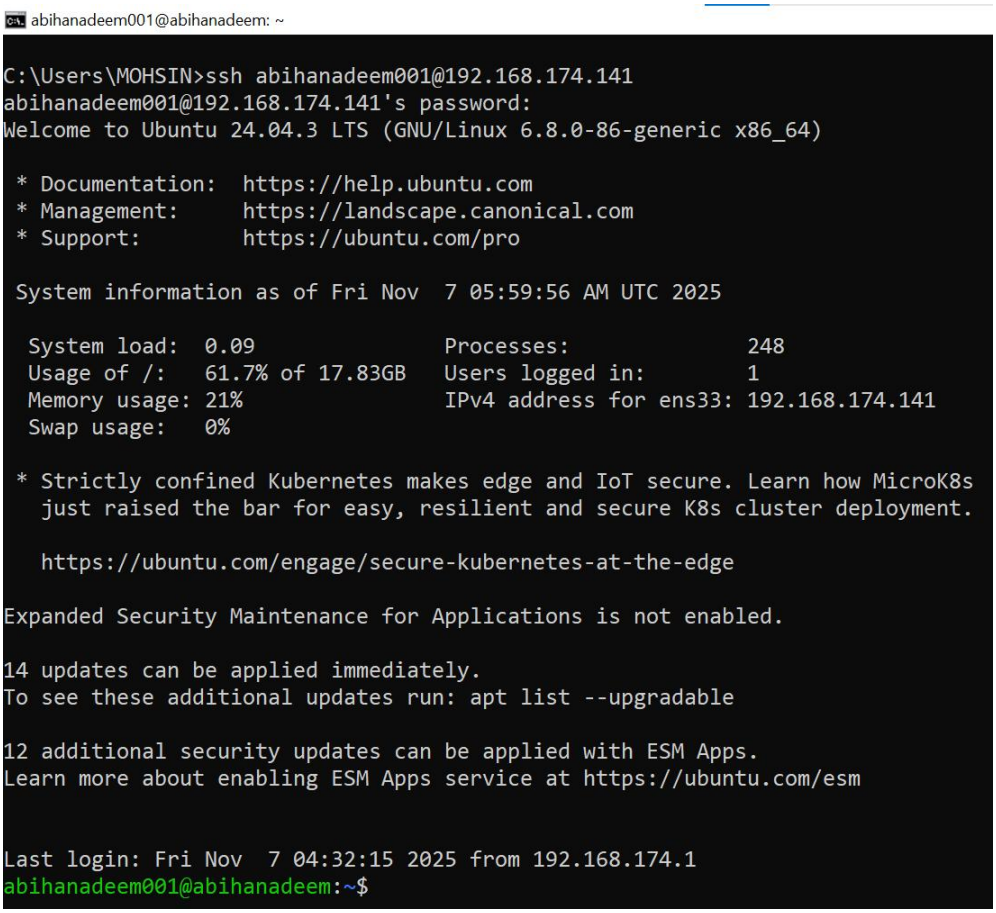
- Save screenshot as: task5_ufw_allow_reload_status.png

```
abihanadeem001@abihanadeem:~$ sudo ufw allow 22/tcp
Rule updated
Rule updated (v6)
abihanadeem001@abihanadeem:~$ sudo ufw reload
Firewall reloaded
abihanadeem001@abihanadeem:~$ sudo ufw status
Status: active

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

abihanadeem001@abihanadeem:~$
```

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

Goal: Copy your public key from the Windows host into the Ubuntu server's ~/.ssh/authorized_keys to allow passwordless SSH. Save grouped screenshots for the client-side actions and the server-side edits/checks.

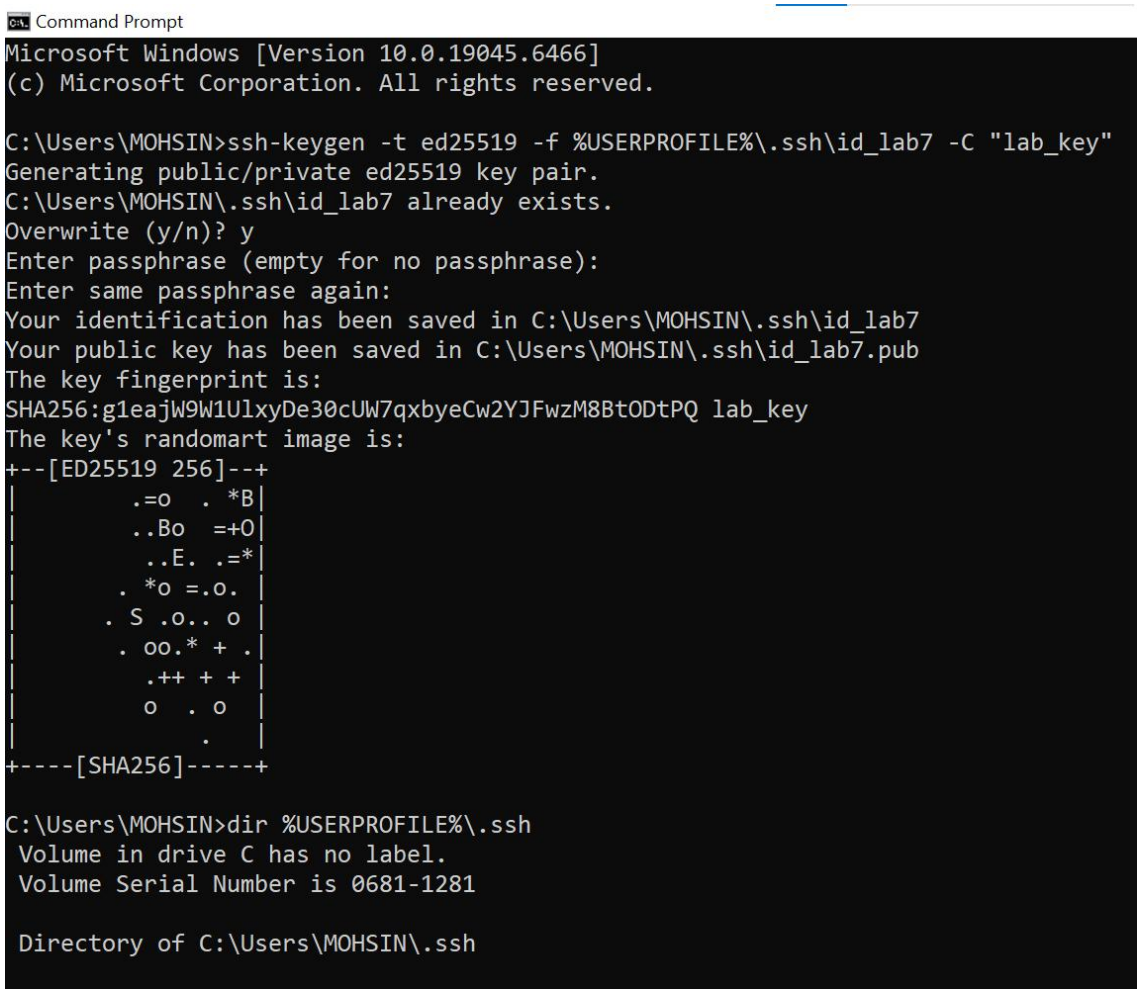
A. On Windows host (client) — group related client actions:

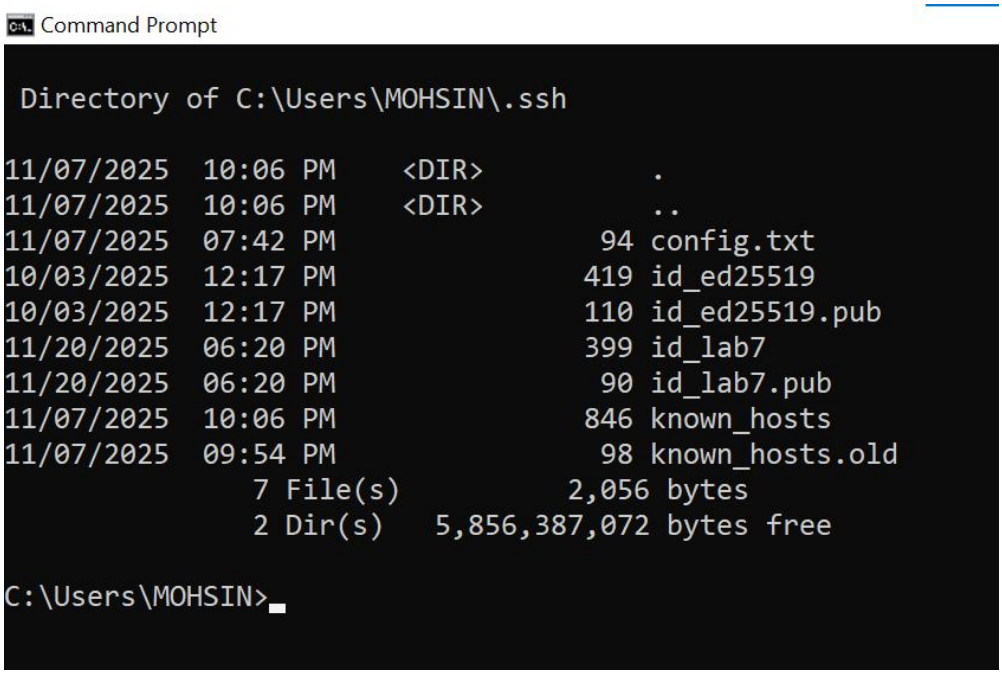
1. Generate ed25519 key pair (if needed) and show the generated files in one screenshot (run ssh-keygen and then list ~/.ssh):

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)



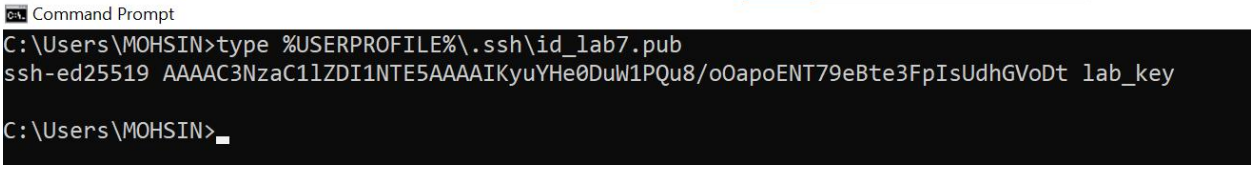


2. Show the public key content (single screenshot):

type $env:USERPROFILE\.ssh\id_lab7.pub

or on Git Bash: cat ~/.ssh/id_lab7.pub

- Save screenshot as: task6_windows_public_key.png



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

Command Prompt

```
C:\Users\MOHSIN>copy NUL %USERPROFILE%\.ssh\known_hosts
Overwrite C:\Users\MOHSIN\.ssh\known_hosts? (Yes/No/All): yes
        1 file(s) copied.

C:\Users\MOHSIN>type %USERPROFILE%\.ssh\known_hosts

C:\Users\MOHSIN>
```

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:

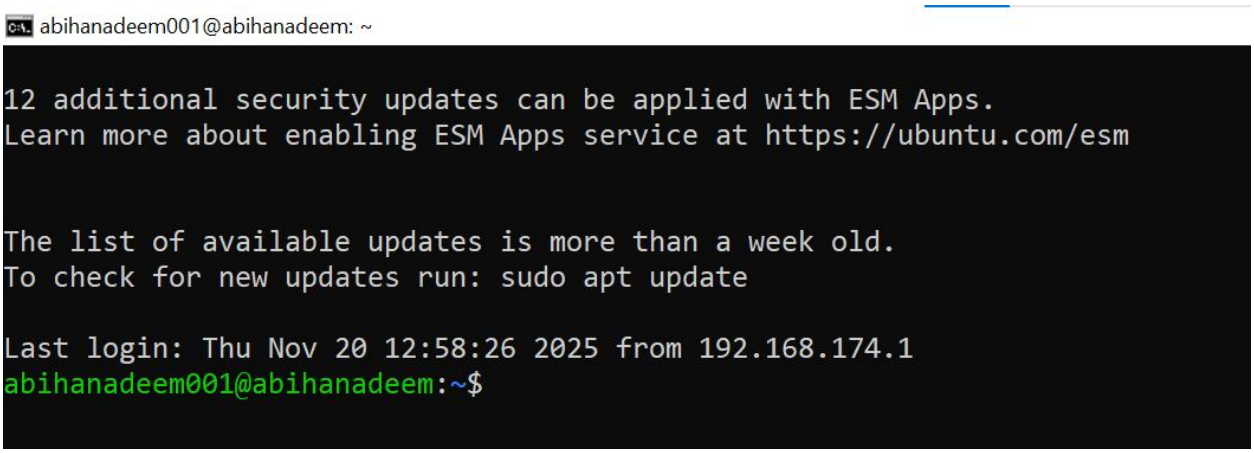
ssh username@<server_ip>

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

abihanadeem001@abihanadeem: ~

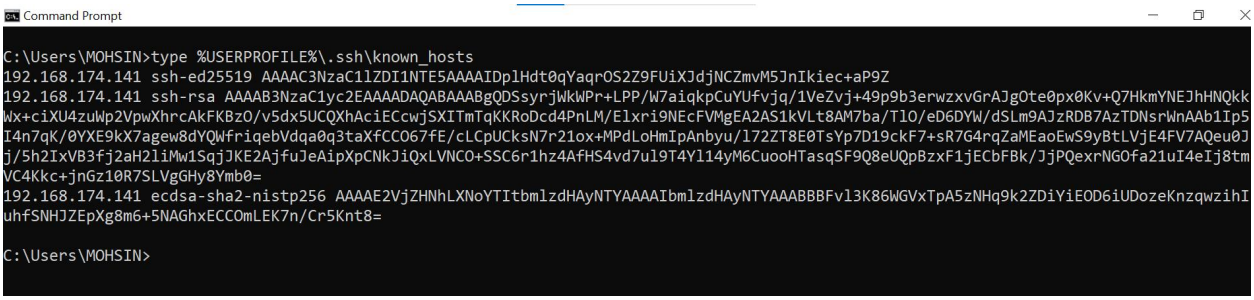
```
C:\Users\MOHSIN>ssh abihanadeem001@192.168.174.141
The authenticity of host '192.168.174.141 (192.168.174.141)' can't be established.
ED25519 key fingerprint is SHA256:TG8kscQEyAaPGTPTgsUeYESbMgZ8x7e13UyKd9Y7Wdw.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? Y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '192.168.174.141' (ED25519) to the list of known hosts.
abihanadeem001@192.168.174.141's password:
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.8.0-86-generic x86_64)

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

 System information as of Thu Nov 20 01:30:55 PM UTC 2025

  System load:  0.0               Processes:             247
  Usage of /:   63.0% of 17.83GB  Users logged in:       1
  Memory usage: 22%               IPv4 address for ens33: 192.168.174.141
  Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

18 updates can be applied immediately.
16 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable
```



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):

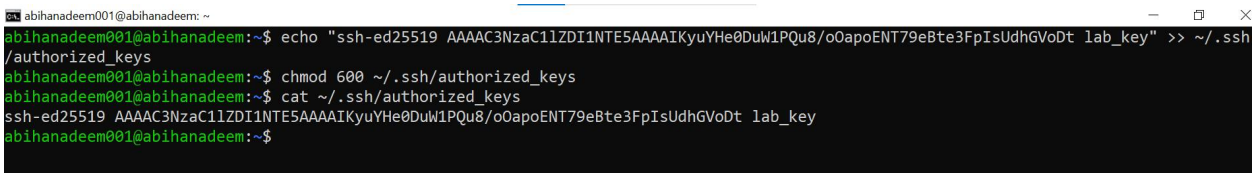
paste public key name id_lab7.pub from Windows client into the echo below

echo "ssh-ed25519 AAAA... yourpublickey ... comment" >> ~/.ssh/authorized_keys

chmod 600 ~/.ssh/authorized_keys

cat ~/.ssh/authorized_keys

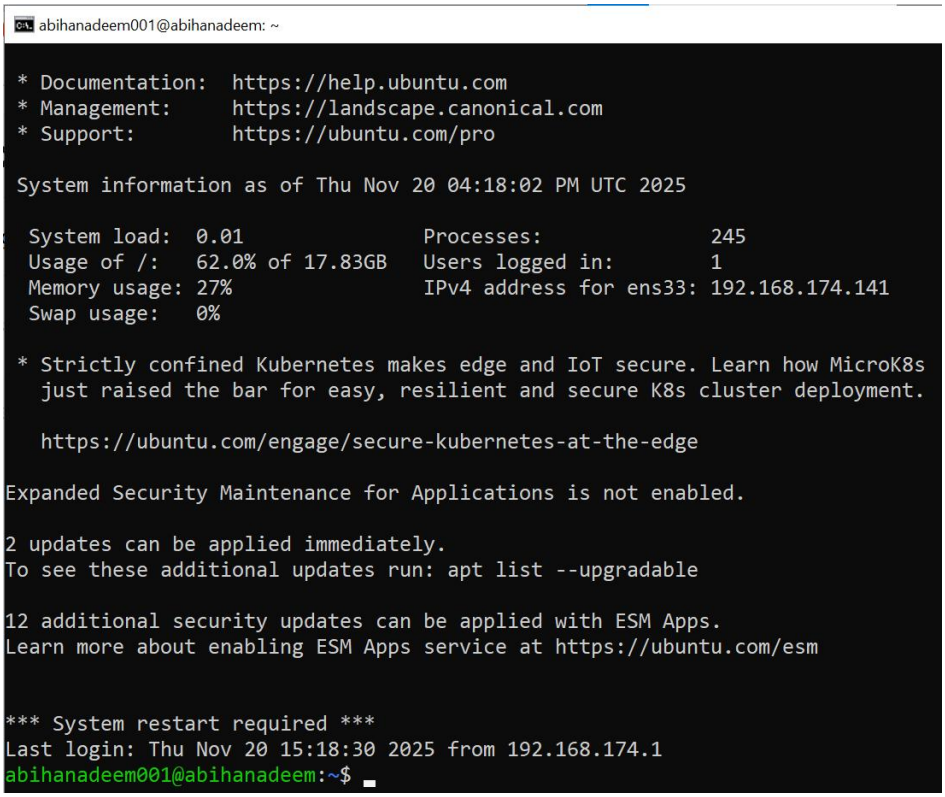
- Save screenshot as: task6_server_add_key_and_show.png (single screenshot showing the commands and resulting authorized_keys content)



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

ssh username@<server_ip>

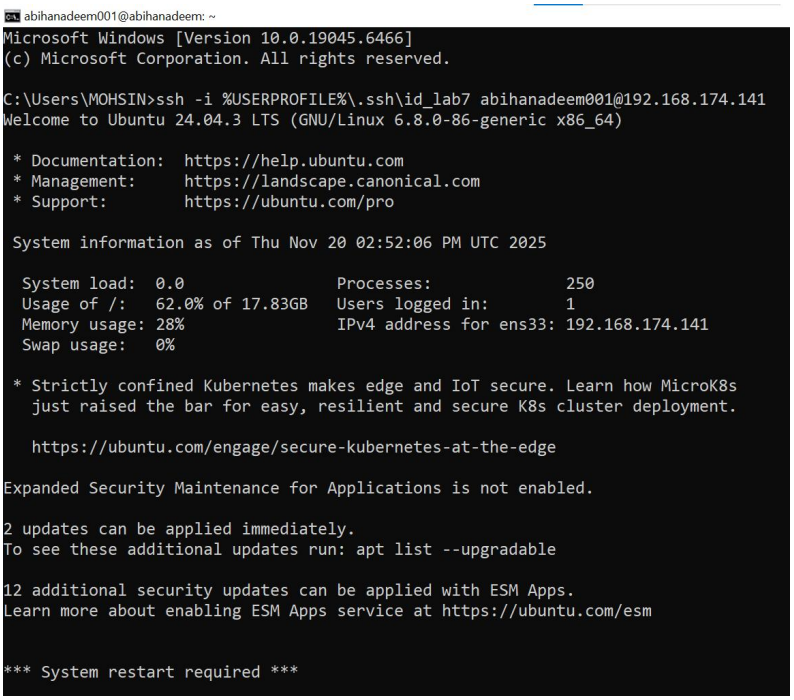
- Save screenshot as: task6_ssh_passwordless_login.png

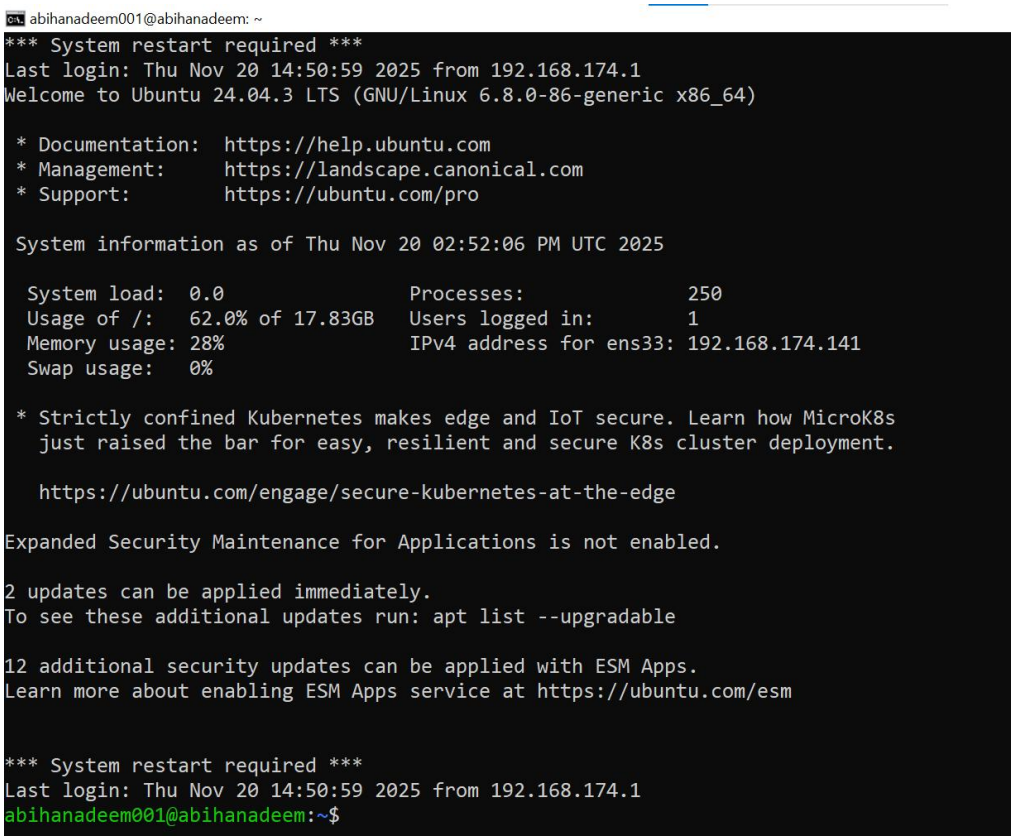


4. Also demonstrate explicit identity usage (single screenshot):

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

- Save screenshot as: task6_ssh_with_identity_file.png





Important notes:

- Do NOT show or upload private key files.
- Ensure server-side permissions are strict: ~/.ssh 700, authorized_keys 600.

Exam Evaluation Questions

Q1: Quick Environment Audit

- Objective: Demonstrate you can inspect the current environment and extract a few key variables.
- Actions & evidence:
    i. Run a single command to display environment variables and capture its output.
        - Save screenshot: EE_q1_env_all.png

ii. 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:

i. 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

```
abihanadeem001@abihanadeem:~$ export STUDENT_NAME="Abiha Nadeem"
abihanadeem001@abihanadeem:~$ export STUDENT_ROLL_NUMBER="2023_BSE_001"
abihanadeem001@abihanadeem:~$ export STUDENT_SEMESTER="5"
abihanadeem001@abihanadeem:~$ _
```

ii. 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

```
abihanadeem001@abihanadeem:~$ echo $STUDENT_NAME
Abiha Nadeem
abihanadeem001@abihanadeem:~$ echo $STUDENT_ROLL_NUMBER
2023_BSE_001
abihanadeem001@abihanadeem:~$ echo $STUDENT_SEMESTER
5
abihanadeem001@abihanadeem:~$
```

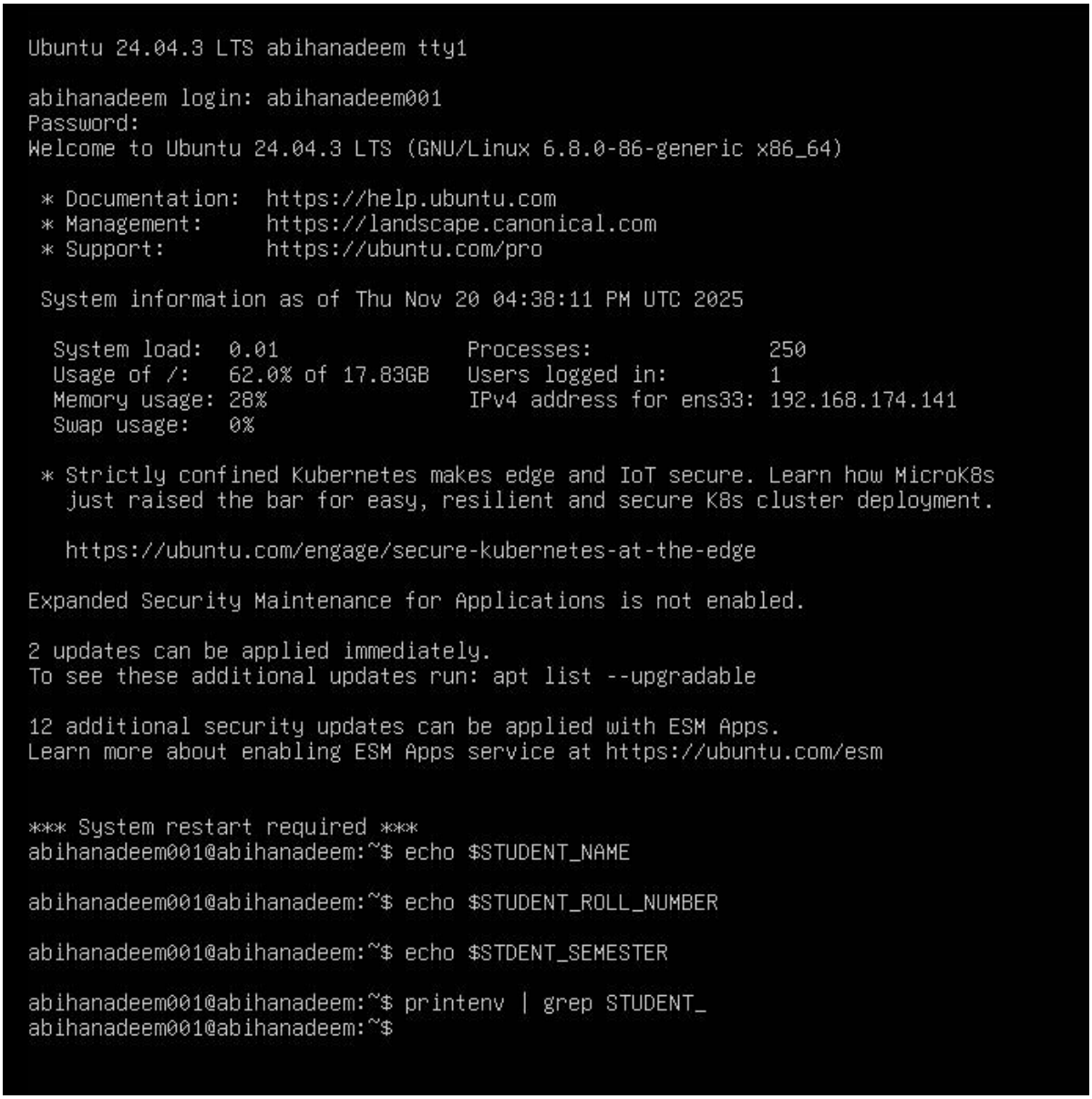
iii. Use a single printenv|grep command to list any STUDENT_ variables and capture the result.

- Save screenshot: EE_q2_printenv_grep.png

```
abihanadeem001@abihanadeem:~$ printenv | grep STUDENT_
STUDENT_NAME=Abiha Nadeem
STUDENT_SEMESTER=5
STUDENT_ROLL_NUMBER=2023_BSE_001
abihanadeem001@abihanadeem:~$ _
```

iv. 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



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

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

```
    . /etc/bash_completion
  fi
fi
# 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"
PATH=$PATH:~
export STUDENT_NAME="Abiha Nadeem"

export STUDENT_ROLL_NUMBER="12345"

export STUDENT_SEMESTER="5"
```

ii. 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

```
abihanadeem001@abihanadeem:~$ source ~/.bashrc
abihanadeem001@abihanadeem:~$ echo $STUDENT_NAME
Abiha Nadeem
abihanadeem001@abihanadeem:~$ echo $STUDENT_ROLL_NUMBER
12345
abihanadeem001@abihanadeem:~$ echo $STUDENT_SEMESTER
5
abihanadeem001@abihanadeem:~$ printenv | grep '^STUDENT_'
STUDENT_NAME=Abiha Nadeem
STUDENT_SEMESTER=5
STUDENT_ROLL_NUMBER=12345
abihanadeem001@abihanadeem:~$ _
```

iii. 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 24.04.3 LTS abihanadeem tty1

abihanadeem login: abihanadeem001
Password:
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.8.0-86-generic x86_64)

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

 System information as of Thu Nov 20 04:44:44 PM UTC 2025

  System load:  0.08              Processes:             253
  Usage of /:   62.0% of 17.83GB  Users logged in:       1
  Memory usage: 28%               IPv4 address for ens33: 192.168.174.141
  Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

2 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

12 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


*** System restart required ***
abihanadeem001@abihanadeem:~$ echo $STUDENT_NAME
Abiha Nadeem
abihanadeem001@abihanadeem:~$ printenv | grep '^STUDENT_'
STUDENT_NAME=Abiha Nadeem
STUDENT_SEMESTER=5
STUDENT_ROLL_NUMBER=12345
abihanadeem001@abihanadeem:~$
```

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

```
abihanadeem001@abihanadeem:~$ sudo ufw enable
[sudo] password for abihanadeem001:
Firewall is active and enabled on system startup
abihanadeem001@abihanadeem:~$ sudo ufw status verbose
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)

abihanadeem001@abihanadeem:~$ _
```

ii. 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

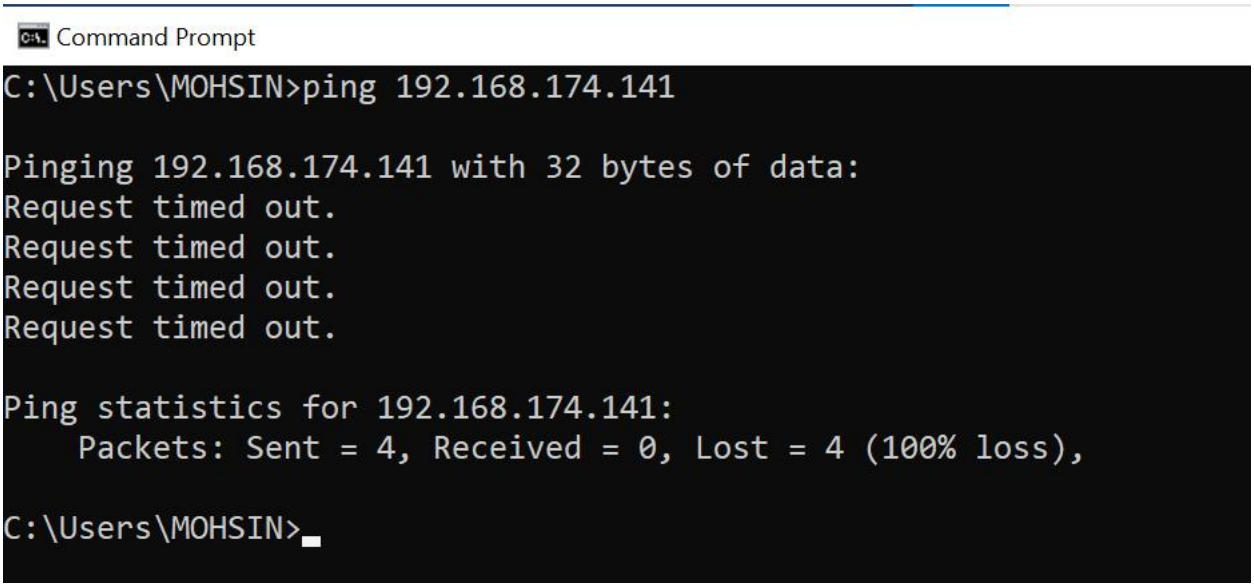
sudo ufw status numbered

- Save screenshot: EE_q5_ufw_deny_ping_status.png

```
abihanadeem001@abihanadeem:~$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22/tcp                     ALLOW IN    Anywhere
[ 2] 22/tcp (v6)                ALLOW IN    Anywhere (v6)

abihanadeem001@abihanadeem:~$
```

iii. 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



iv. Re-allow ping (ICMP) (or remove the deny rule) and capture the allow/reload/status sequence in one screenshot.

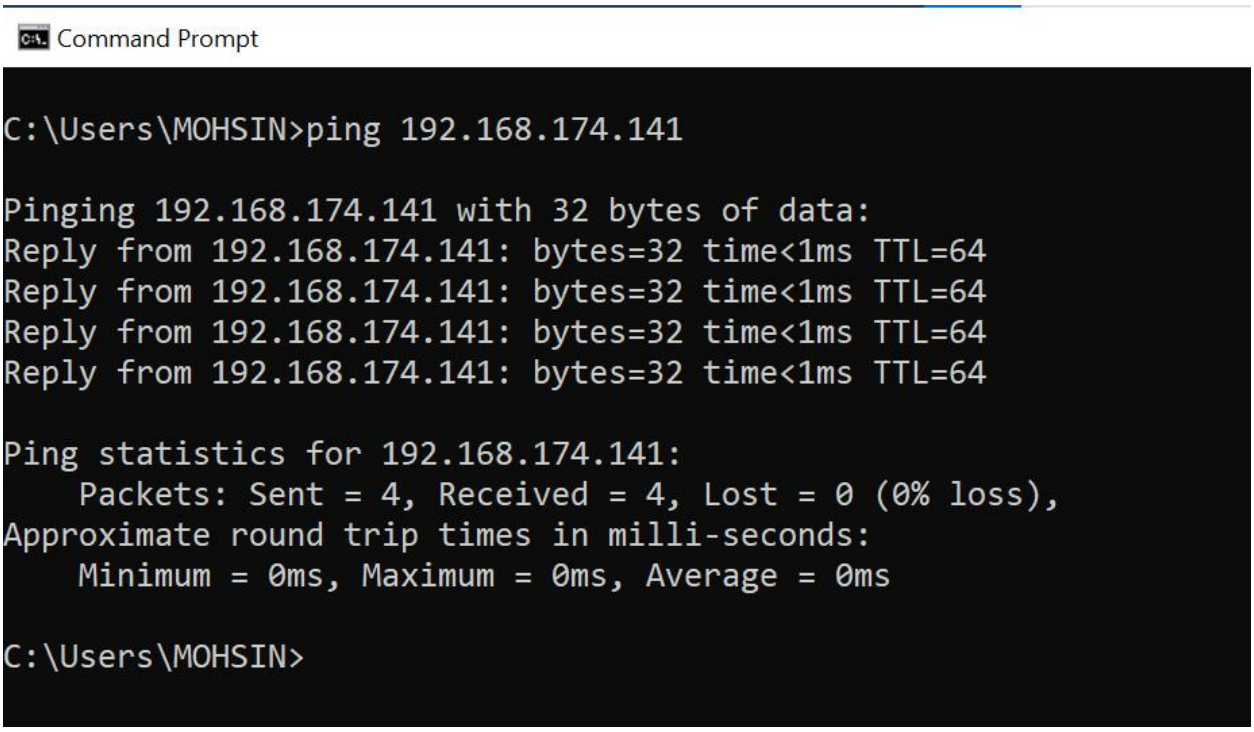
- Suggested command example:
- sudo ufw allow proto icmp from any to any
- sudo ufw reload

  sudo ufw status
- Save screenshot: EE_q5_ufw_allow_ping_status.png

```
-A ufw-before-input -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A ufw-before-output -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A ufw-before-forward -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

# drop INVALID packets (logs these in loglevel medium and higher)
-A ufw-before-input -m conntrack --ctstate INVALID -j ufw-logging-deny
-A ufw-before-input -m conntrack --ctstate INVALID -j DROP

# ok icmp codes for INPUT
-A ufw-before-input -p icmp --icmp-type destination-unreachable -j ACCEPT
-A ufw-before-input -p icmp --icmp-type time-exceeded -j ACCEPT
-A ufw-before-input -p icmp --icmp-type parameter-problem -j ACCEPT
-A ufw-before-input -p icmp --icmp-type echo-request -j ACCEPT

# ok icmp code for FORWARD
-A ufw-before-forward -p icmp --icmp-type destination-unreachable -j ACCEPT
-A ufw-before-forward -p icmp --icmp-type time-exceeded -j ACCEPT
-A ufw-before-forward -p icmp --icmp-type parameter-problem -j ACCEPT
-A ufw-before-forward -p icmp --icmp-type echo-request -j ACCEPT

# allow dhcp client to work
-A ufw-before-input -p udp --sport 67 --dport 68 -j ACCEPT


abihanadeem001@abihanadeem:~$ sudo ufw reload
Firewall not enabled (skipping reload)
abihanadeem001@abihanadeem:~$ sudo ufw status
Status: active

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

abihanadeem001@abihanadeem:~$ sudo ufw reload
Firewall not enabled (skipping reload)
abihanadeem001@abihanadeem:~$ sudo ufw enable

Firewall is active and enabled on system startup
abihanadeem001@abihanadeem:~$ sudo ufw reload
Firewall reloaded
abihanadeem001@abihanadeem:~$ sudo ufw status
Status: active

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

abihanadeem001@abihanadeem:~$
```

v. From the client, ping the server again and capture successful replies in one screenshot.

- Save screenshot: EE_q5_ping_success.png


Command Prompt
C:\Users\MOHSIN>ping 192.168.174.141
Pinging 192.168.174.141 with 32 bytes of data:
Reply from 192.168.174.141: bytes=32 time<1ms TTL=64
Reply from 192.168.174.141: bytes=32 time<1ms TTL=64
Reply from 192.168.174.141: bytes=32 time<1ms TTL=64
Reply from 192.168.174.141: bytes=32 time<1ms TTL=64
Ping statistics for 192.168.174.141:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
C:\Users\MOHSIN>

****