



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

```
ubuntu@ubuntuserver:~ $ printenv
SHELL=/bin/bash
PWD=/home/ubuntu
LOGNAME=ubuntu
XDG_SESSION_TYPE=tty
HOME=/home/ubuntu
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33:01:cd=40;33:01:or=40;31:01:mi=00:su=37;41:sg=30;43:ca=00:tws=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzha=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.tz=01;31:*.tzs=01;31:*.tzst=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lz=01;31:*.xz=01;31:*.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cipio=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.avif=01;35:*.jpg=01;35:*.jpeg=01;35:*.njpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.webp=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.xcf=01;35:*.xwd=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogg=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:*.rtf=00;36:*.bak=00;90:*.crdownload=00;90:*.dpkg-dist=00;90:*.dpkg-new=00;90:*.dpkg-old=00;90:*.dpkg-tmp=00;90:*.old=00;90:*.orig=00;90:*.part=00;90:*.rej=00;90:*.rpmsave=00;90:*.rpmmorig=00;90:*.swp=00;90:*.tmp=00;90:*.ucf-dist=00;90:*.ucf-new=00;90:*.ucf-old=00;90:*.SSH_CONNECTION=192.168.85.1 59922 192.168.85.133 22
LESSCLOSE=/usr/bin/lesspipe %s %
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %
LIBVIRT_DEFAULT_URI=qemu:///system
USER=ubuntu
```

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)

```
ubuntu@ubuntuserver:~ $ printenv | grep SHELL
printenv | grep HOME
printenv | grep USER
SHELL=/bin/bash
HOME=/home/ubuntu
USER=ubuntu
ubuntu@ubuntuserver:~ $
```

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)

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

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

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

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

```
printenv | grep '^DB_'
```

Save screenshot as: task2_printenv_grep_db.png

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

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)

```
ubuntu@ubuntuserver:~$ echo "$DB_URL"
printenv | grep '^DB_'

ubuntu@ubuntuserver:~$ |
```

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)

```
# Lab 7 persistent DB variables
export DB_URL="postgres://db.example.local:5432/mydb"
export DB_USER="labuser"
export DB_PASSWORD="labpass123"
|
-- INSERT --
```

- 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)

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

- 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:~$ vim /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:~$ 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)

```
ubuntu@ubuntuserver:~$ echo $CLASS
echo "$PATH"
CC-Lujainzia
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
ubuntu@ubuntuserver:~$ |
```

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)

```
ubuntu@ubuntuserver:~$ cat > ~/welcome <<'EOF'
#!/bin/bash
echo "Welcome to Cloud Computing $USER"
EOF
ubuntu@ubuntuserver:~$ chmod +x ~/welcome
ubuntu@ubuntuserver:~$ |
```

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

```
cd ~ ./welcome
```

Save screenshot as: task4_welcome_run_dot.png

```
ubuntu@ubuntuserver:~$ ./welcome
Welcome to Cloud Computing ubuntu
ubuntu@ubuntuserver:~$ |
```

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:~|
```

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)

```
ubuntu@ubuntuserver:~$ source ~/.bashrc
cd ~
welcome
Welcome to Cloud Computing ubuntu
ubuntu@ubuntuserver:~$ |
```

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

```
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
ubuntu@ubuntuserver:~$ |
```

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

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

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

ubuntu@ubuntuserver:~$ |
```

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

```
PS C:\Users\user> ssh ubuntu@192.168.85.133
ssh: connect to host 192.168.85.133 port 22: Connection timed out
PS C:\Users\user> |
```

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

```
ubuntu@ubuntuserver:~$ sudo ufw allow 22/tcp
[sudo] password for ubuntu:
Rule updated
Rule updated (v6)
ubuntu@ubuntuserver:~$ sudo ufw reload
sudoFirewall reloaded
ubuntu@ubuntuserver:~$ sudo ufw status
Status: active

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

ubuntu@ubuntuserver:~$
```

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

```
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
```

```
Last login: Tue Nov 18 19:17:21 2025 from 192.168.85.1
ubuntu@ubuntuserver:~$ |
```

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:
SHA256:7CXJFuwBfBespiXIxV2kMt7fDGSlbqiv0N93Z15Vls 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 AAAAC3NzaC1lZDI1NTE5AAAAIBBD3cRHs81Zm40LYL8crIJwRT6gjbHWV9t6lJJtGRNn 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

```

ubuntu@ubuntuserver:~ + 
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro

System information as of Tue Nov 18 08:06:20 PM UTC 2025

System load: 0.0          Processes:           238
Usage of /: 65.0% of 17.83GB   Users logged in:      1
Memory usage: 11%          IPv4 address for ens33: 192.168.85.133
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.

15 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

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Tue Nov 18 19:44:42 2025 from 192.168.85.1
ubuntu@ubuntuserver:$ |

```

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

```

PS C:\Users\user> type $env:USERPROFILE\.ssh\known_hosts
192.168.85.133 ssh-ed25519 AAAAC3NzaC1lZDI1NTk07RX7+dGtjHHdxy8gMBsuy9kM0Z0q1mM7FWpcQxl
192.168.85.133 ssh-rsa AAAAB3NzaC1yc2EAAADQABAAABgQC6KfmaXQt6qmLMJ1KfbT1sU11trGQ0UYFSouJqiLGkeGDmgrx27a5fAPQhUUKGe4M4F
RKzEItR32ioznXFILEps4fe5wjhIBG/HABGllwTU/nVQJgul/xMPmoACNhQh1icufvk7dnYcuZ0pfrGCh01FhJsk3rJibWk2/AteidJcqXhGv1lXGx
m7CLDRdeYcU9LXNDu9+LD1benQS6Jubd/S+PytrD8j+yMKDNQBBu80PtFsAyuoWNyVRtxTptL94xL1eGU+1aft6f6vdIt7owxZA422DS0SRlkQhuvlfn
apiODZZE0B1kiDMgLXRu4ksWpxVTIAJgg9IGVzk16PJJF41RSgG08Tpnlj96/7kwlgUgdHk9ZTvqn10AdcoUvA088a7bH1S3piRGlGan67oBxDGd17fvsQB
JPlkVtN6V107w5f9QgoXJh18Je0JPjk4thrZHiVTypa/7ry28rPVUggzpJw03cg96jXSMcPhq16NbnumH/XApk=
192.168.85.133 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYT1tbmlzdHAYNTYAAAIBmlzdHAYNTYAAABBAbHoaSD0ojc3J0LGPddG4BX1JMgpA8C
j+BhH2Qnb7B/NDBxHCDryzIMrqU7YXSo5kzfYMe5r97jwoZB+Nyjo=
PS C:\Users\user> |

```

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

```

ubuntu@ubuntuserver:~$ mkdir -p ~/.ssh
ubuntu@ubuntuserver:~$ chmod 700 ~/.ssh
> ~/.ssh/authorized_keys
ubuntu@ubuntuserver:~$ |

```

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)

```
ubuntu@ubuntuserver:~$ echo "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBBD3cRHs81Zm40LYL8crIJwRT6gjbHWV9t6lJJtGRNn lab_key" > ~/.ssh/authorized_keys
> ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
cat ~/.ssh/authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBBD3cRHs81Zm40LYL8crIJwRT6gjbHWV9t6lJJtGRNn lab_key
ubuntu@ubuntuserver:~|
```

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

```
ubuntu@ubuntuserver:~ x + v
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro

System information as of Tue Nov 18 08:14:37 PM UTC 2025

System load: 0.0 Processes: 238
Usage of /: 65.0% of 17.83GB Users logged in: 1
Memory usage: 11% IPv4 address for ens33: 192.168.85.133
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.

15 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

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Tue Nov 18 20:13:09 2025 from 192.168.85.1
ubuntu@ubuntuserver:~|
```

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

```
PS C:\Users\user> ssh -i $env:USERPROFILE\.ssh\id_lab7 ubuntu@192.168.85.133
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 Tue Nov 18 08:14:37 PM UTC 2025

System load: 0.0 Processes: 238
Usage of /: 65.0% of 17.83GB Users logged in: 1
Memory usage: 11% IPv4 address for ens33: 192.168.85.133
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.

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

12 additional security updates can be applied with ESM Apps.
```

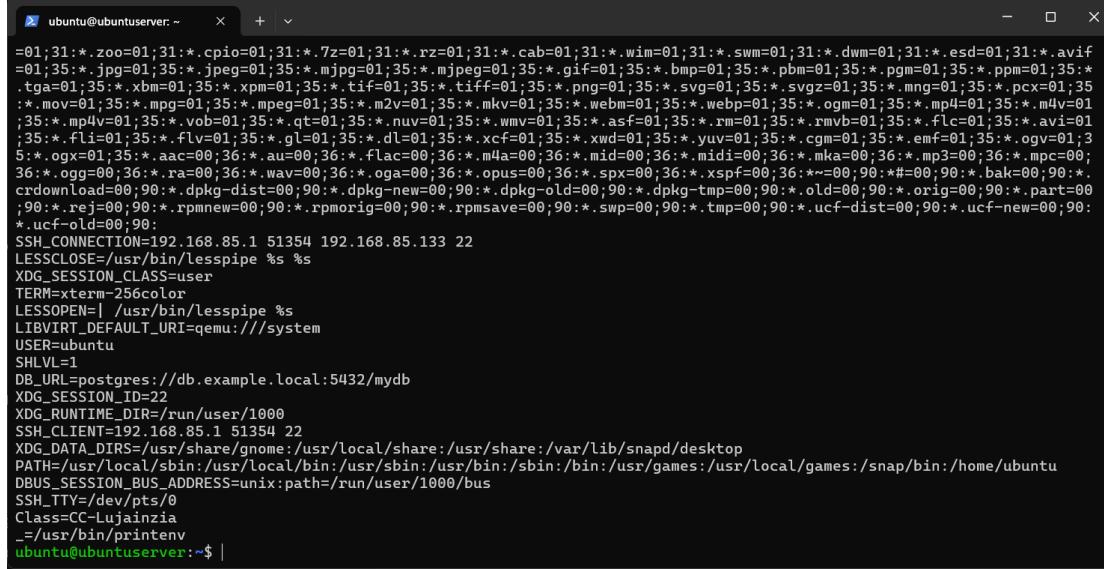
Exam Evaluation Questions

Q1: Quick Environment Audit

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

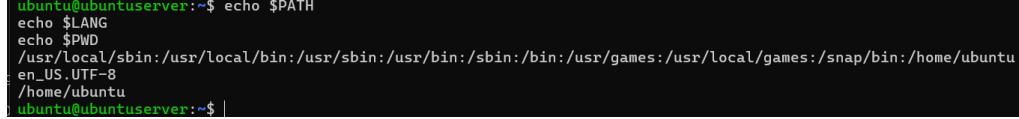
Actions & evidence:

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



```
ubuntu@ubuntuserver:~ $ printenv
=01;31::* .zoo=01;31::* .cpio=01;31::* .7z=01;31::* .rz=01;31::* .cab=01;31::* .wim=01;31::* .swm=01;31::* .dwm=01;31::* .esd=01;31::* .avif
=01;35::* .jpg=01;35::* .jpeg=01;35::* .mjpg=01;35::* .mpeg=01;35::* .tif=01;35::* .png=01;35::* .svg=01;35::* .svgz=01;35::* .mng=01;35::* .pcx=01;35
.* .tga=01;35::* .xbm=01;35::* .xpm=01;35::* .tiff=01;35::* .tiff=01;35::* .m2v=01;35::* .mkv=01;35::* .webm=01;35::* .webp=01;35::* .ogg=01;35
.* .mp4=01;35::* .vob=01;35::* .qt=01;35::* .nuv=01;35::* .wmv=01;35::* .asf=01;35::* .rm=01;35::* .rmvb=01;35::* .flc=01;35::* .avi=01
;35.* .mp4v=01;35::* .vob=01;35::* .qt=01;35::* .nuv=01;35::* .wmv=01;35::* .asf=01;35::* .rm=01;35::* .rmvb=01;35::* .flc=01;35::* .avi=01
;35.* .ogg=01;35::* .aac=00;36::* .au=00;36::* .flac=00;36::* .m4a=00;36::* .mid=00;36::* .midi=00;36::* .mka=00;36::* .mp3=00;36::* .mpc=00;
36::* .ogg=00;36::* .ra=00;36::* .wav=00;36::* .oga=00;36::* .opus=00;36::* .spx=00;36::* .xspf=00;36::* ~=00;90::* #=00;90::* .bak=00;90::*;
crdownload=00;90::* .dpkg-dist=00;90::* .dpkg-new=00;90::* .dpkg-old=00;90.* .dpkg-tmp=00;90.* .old=00;90.* .orig=00;90.* .part=00
;90.* .rej=00;90.* .rpminew=00;90::* .rpmmorig=00;90.* .rpmsave=00;90.* .swp=00;90.* .tmp=00;90.* .ucf-dist=00;90.* .ucf-new=00;90*
.* .ucf-old=00;90:
SSH_CONNECTION=192.168.85.1 51354 192.168.85.133 22
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
LIBVIRT_DEFAULT_URI=qemu://system
USER=ubuntu
SHLVL=1
DB_URL=postgres://db.example.local:5432/mydb
XDG_SESSION_ID=22
XDG_RUNTIME_DIR=/run/user/1000
SSH_CLIENT=192.168.85.1 51354 22
XDG_DATA_DIRS=/usr/share/gnome:/usr/local/share:/usr/share:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/home/ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
SSH_TTY=/dev/pts/0
Class=CC-Lujainzia
_=~/usr/bin/printenv
ubuntu@ubuntuserver:~$ |
```

- **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**



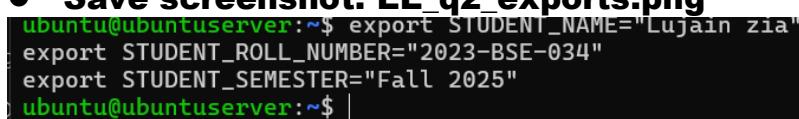
```
ubuntu@ubuntuserver:~$ echo $PATH
echo $LANG
echo $PWD
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/home/ubuntu
en_US.UTF-8
/home/ubuntu
ubuntu@ubuntuserver:~$ |
```

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:~$ export STUDENT_NAME="Lujain zia"
export STUDENT_ROLL_NUMBER="2023-BSE-034"
export STUDENT_SEMESTER="Fall 2025"
ubuntu@ubuntuserver:~$ |
```

```
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**

```
ubuntu@ubuntuserver:~$ sudo iptables -I INPUT -p icmp --icmp-type echo-request -j DROP
ubuntu@ubuntuserver:~$ sudo ufw status numbered
Status: active

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

ubuntu@ubuntuserver:~$ |
```

- **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**

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

Pinging 192.168.85.133 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.85.133:
  Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PS C:\Users\user> |
```

- **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**

```
ubuntu@ubuntuserver:~$ sudo ufw reload
Firewall reloaded
ubuntu@ubuntuserver:~$ sudo ufw status
Status: active

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

ubuntu@ubuntuserver:~$ |
```

- **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> |
```