PART 2 — FULL CREDIT EXPECTATIONS (ONE ACTIVITY ONLY)

Server = **Rocky Linux**
Client = **Rocky Linux** (unless stated otherwise)

You must be able to show:

- service installed
- config file edited
- service enabled + running
- firewall configured
- verification command
- (often) client access proof

1️⃣ SAMBA (SMB FILE SHARING) — BEST PICK

SERVER (Rocky)

Install

```
sudo dnf install -y samba samba-client
```

Create share directory

```
sudo mkdir -p /srv/samba/share
sudo chown -R nobody:nobody /srv/samba/share
sudo chmod 777 /srv/samba/share
```

Configure Samba

```
sudo nano /etc/samba/smb.conf
```

Add (lab-style, simple guest share):

```
[share]
path = /srv/samba/share
browseable = yes
writable = yes
guest ok = yes
read only = no
```

Check config

```
testparm
```

Enable service

```
sudo systemctl enable --now smb
sudo systemctl status smb
```

Firewall

```
sudo firewall-cmd --add-service=samba --permanent
sudo firewall-cmd --reload
```

CLIENT (Rocky)

List shares

```
smbclient -L //<SERVER_IP> -N
```

Mount share

```
sudo mkdir /mnt/samba
sudo mount -t cifs //<SERVER_IP>/share /mnt/samba -o guest
ls /mnt/samba
```

✅ **Grader checks**

- `smb.conf`
- `systemctl status smb`
- `testparm`
- successful client access

2️⃣ FTP (vsftpd)

SERVER (Rocky)

Install

```
sudo dnf install -y vsftpd
```

Configure

```
sudo nano /etc/vsftpd/vsftpd.conf
```

Ensure:

```
anonymous_enable=YES
write_enable=YES
local_enable=YES
```

Create FTP directory

```
sudo mkdir -p /var/ftp/pub
sudo chmod 777 /var/ftp/pub
```

Enable service

```
sudo systemctl enable --now vsftpd
```

```
sudo systemctl status vsftpd
```

Firewall

```
sudo firewall-cmd --add-service=ftp --permanent
sudo firewall-cmd --reload
```

CLIENT (Rocky)

```
ftp <SERVER_IP>
```

Login:

```
Name: anonymous
Password: [Enter]
```

Test:

```
ls
put test.txt
quit
```

✅ **Grader checks**

- vsftpd running
- config edited
- firewall open
- upload/download works

3️⃣ VIRTUAL WEB SERVER (HTTP + DNS)

SERVER (Rocky – Apache)

Install

```
sudo dnf install -y httpd
```

Create web content

```
echo "Virtual Host 1" | sudo tee /var/www/html/index.html
```

Enable service

```
sudo systemctl enable --now httpd
sudo systemctl status httpd
```

Firewall

```
sudo firewall-cmd --add-service=http --permanent
sudo firewall-cmd --reload
```

SERVER (Windows Server – DNS)

- DNS Manager
- Create **A record** or **CNAME**
- Point hostname → Rocky web server IP

CLIENT (Rocky)

```
curl http://hostname.yourdomain.com
```

✅ **Grader checks**

- Apache running
- DNS resolves

- HTTP reachable from client

4️⃣ SECURE WEB SERVER (HTTPS)

SERVER (Rocky)

Install

```
sudo dnf install -y httpd mod_ssl
```

Create self-signed cert

```
sudo openssl req -x509 -nodes -days 365 \
-newkey rsa:2048 \
-keyout /etc/pki/tls/private/localhost.key \
-out /etc/pki/tls/certs/localhost.crt
```

Enable service

```
sudo systemctl enable --now httpd
```

Firewall

```
sudo firewall-cmd --add-service=https --permanent
sudo firewall-cmd --reload
```

CLIENT (Rocky)

```
curl -k https://hostname.yourdomain.com
```

✅ Grader checks

- HTTPS listening on 443

- cert exists
- client access works

5️⃣ NFS (Network File System)

SERVER (Rocky)

Install

```
sudo dnf install -y nfs-utils
```

Create export

```
sudo mkdir -p /srv/nfs/share
sudo chmod 777 /srv/nfs/share
```

Configure

```
sudo nano /etc/exports
```

Add:

```
/srv/nfs/share *(rw,sync,no_root_squash)
```

Apply exports

```
sudo exportfs -rav
```

Enable service

```
sudo systemctl enable --now nfs-server
sudo systemctl status nfs-server
```

Firewall

```
sudo firewall-cmd --add-service=nfs --permanent
sudo firewall-cmd --reload
```

CLIENT (Rocky)

```
sudo mkdir /mnt/nfs
sudo mount <SERVER_IP>:/srv/nfs/share /mnt/nfs
ls /mnt/nfs
```

✅ **Grader checks**

- `/etc/exports`
- `exportfs -v`
- mount works

6️⃣ GPOs (Windows-only option)

This one is **procedural**, not command-heavy.

SERVER (Windows Server)

- Group Policy Management
- Create GPO
- Link to OU or domain
- Configure **one visible policy**
    - Disable Control Panel (classic lab example)

CLIENT (Windows 11)

```
gpupdate /force
gpresult /r
```

✅ **Grader checks**

- GPO linked
- client shows it applied

KEY TAKEAWAY (THIS IS IMPORTANT)

For **full credit**, Part 2 always expects:

1. Install
2. Configure (file edited)
3. Enable/start service
4. Firewall adjusted
5. Verification
6. Client access (when applicable)

If you skip **any one**, you lose points.

Part 1 — Build the infrastructure (pfSense + Windows Server 2025 + Win11 + Rocky) and get DHCP/DNS/ADDS working

A) Lab 1: Topology + static IPs (before AD/DHCP)

VMware host actions (bench machine)

- Make linked clones and store them on the approved drive (Lab 1 uses D:\VMs; your practical slide says Thawspace (T:), so follow exam rule).

pfSense (SERVER: gateway)

1. Assign interfaces (console):

- `1) Assign Interfaces` → **No VLANs** → set **WAN = NAT NIC**, **LAN = LAN-segment NIC**
2. Set LAN IP (console):

- `2) Set interface(s) IP address` → LAN → static like `192.168.1.254/24`

- **Do NOT enable DHCP on pfSense LAN** (DHCP is on Windows Server later)

Windows Server 2025 (SERVER for AD/DNS/DHCP later)

- Static IPv4 example:
    - IP `192.168.1.2`
    - /24
    - GW = pfSense LAN (`192.168.1.254`)
    - DNS: primary = itself/loopback, secondary = `8.8.8.8` (or RIT resolver)

Rocky Linux (CLIENT for practical, also can be server in Part 2)

- Set hostname:

```
sudo su -
hostnamectl set-hostname <FQDN>
hostnamectl
```

- Static IP via nmcli:

```
nmcli connection edit ens192
set ipv4.addresses 192.168.X.X/24
set ipv4.gateway 192.168.X.X
set ipv4.dns 8.8.8.8
save
quit
nmcli
```

Windows 11 (CLIENT)

- Rename:

```
Rename-Computer -NewName <hostname>
```

- Static IP (PowerShell):

```
Get-NetAdapter
New-NetIPAddress -InterfaceIndex 7 `
 -IPAddress 192.168.1.3 `
 -PrefixLength 24 `
 -DefaultGateway 192.168.1.X
Set-DnsClientServerAddress `
 -InterfaceIndex 7 `
 -ServerAddresses ("8.8.8.8","8.8.4.4")
```

B) Lab 2: AD DS + DNS + DHCP + create user + join Win11 to domain

Windows Server 2025 (SERVER)
You must end Part 1 with: AD DS installed, DNS installed, DHCP installed + scope active, and Win11 joined.

DHCP scope creation (SERVER)

- DHCP console: IPv4 → **New Scope**

Exclusions:

- Exclude **pfSense LAN IP**
- Exclude a small range for servers (example shown excludes `192.168.10.1–10` and `.254`)

Configure options:

- Gateway = pfSense LAN IP

DNS = server IP already listed; add secondary `8.8.8.8`

Activate the scope

Create a test user + make Domain Admin (SERVER)

- When setting password: uncheck “must change”, check “cannot change” + “password never expires”

Add user to **Domain Admins** group via “Add to a group” → select Domain Admins

Clients switch to DHCP (CLIENTS)

- Win11 and Rocky: change NIC config to DHCP; verify with `ipconfig /all` on Windows

Join Windows 11 to domain (CLIENT: Win11)
Run as admin:

```
Add-Computer –DomainName yourdomain.com –Credential (Get-Credential)
```

Reboot, then log in as “Other user” into the domain.

Part 2 — The 6 easiest activities (your list) + exact Server/Client commands

From your bank, the easiest 6 are:
GPOs, Virtual Web Server, Secure Web Server, Samba, FTP, NFS
(skip **RAID** + **Email** unless you’re forced)

1) GPOs (Windows Server only)

SERVER (Windows Server 2025)

- Typical flow: Group Policy Management → make GPO → link to OU → update clients.
  CLIENT (Win11)

```
gpupdate /force
```

That's the "make it apply now" button.

---

2) Virtual Web Server (Rocky = web server; Windows Server = DNS)

This comes straight out of Lab 6: you create vhosts + DNS aliases.

SERVER (Rocky web server)
Install + start Apache:

```
sudo dnf -y install httpd
sudo systemctl enable --now httpd
```

Lab 6 explicitly says install httpd and start/enable the service.

Open firewall for HTTP (conceptually required by Lab 6):

- Lab 6 says create a firewall rule to allow incoming HTTP then reload firewalld.

SERVER (Windows Server DNS)

- Add DNS aliases (CNAMEs) for `www`, and your 2 vhosts (example uses "starlord" + "gamora"), mapped to the web server FQDN.

CLIENT (Rocky client)
Test resolution + HTTP:

```
ping <site.yourdomain>
curl -I http://<site.yourdomain>
```

---

3) Secure Web Server (HTTPS / self-signed)

Same idea as above, but you add TLS (Lab 6 activities 3–4).

SERVER (Rocky web server)

- You're expected to create a self-signed cert and configure Apache for TLS (Lab 6 summary).

CLIENT (Rocky client)
Test:

```
curl -k https://<site.yourdomain>
```

(`-k` ignores the self-signed warning)

---

4) FTP (Rocky server + Rocky client)

Lab 5 is explicit: install/configure FTP + create a "drop box".

SERVER (Rocky)
Core tasks:

- install vsftpd
- enable/start service
- firewall allow FTP
- set up dropbox directory/permissions

CLIENT (Rocky)
Test with:

```
ftp <server>
```

(or `curl ftp://...`)

---

5) Samba (Rocky server + Rocky client)

Lab 5 explicitly includes “Installing Samba and Creating Share” and mounting/accessing shares.

SERVER (Rocky)
Core tasks:

- install samba packages
- configure `/etc/samba/smb.conf`
- enable/start smb services
- firewall allow samba
- create share directory + perms

CLIENT (Rocky)
Mount test (common approach):

```
sudo mount -t cifs //<server>/<share> /mnt/<mountpoint> -o
username=<user>
```

---

6) NFS (Rocky server + Rocky client)

Lab 5 explicitly includes “Installing NFS and Creating a Simple NFS Share.”

SERVER (Rocky)
Core tasks:

- install `nfs-utils`
- configure `/etc/exports`
- `exportfs -rav`
- enable/start `nfs-server`
- firewall allow nfs-related services

CLIENT (Rocky)
Mount test:

```
sudo mount -t nfs <server>:/export/path /mnt/<mountpoint>
```

Part 3 — rsync script + cron (the exact “cron command” you asked for)

A) Script expectation (review slide)

You’re writing a script that does backup/restore using rsync, then scheduling it via cron.

B) The simplest Python style (subprocess.run)

Use this pattern:

```
#!/usr/bin/env python3
import subprocess
from datetime import datetime

SRC = "/home/student/data/"
DST = "/backup/data/"

cmd = ["rsync", "-avh", "--delete", SRC, DST]
r = subprocess.run(cmd, capture_output=True, text=True)

with open("/var/log/backup_report.txt", "a") as f:
    f.write(f"\n[{datetime.now()}] rc={r.returncode}\n")
    f.write(r.stdout)
    f.write(r.stderr)
```

C) “What explicit command runs a cron?”

You don’t “run cron” manually. You:

1. **start/enable the cron daemon**

```
sudo systemctl enable --now crond
```

2. **install a job with**

```
crontab -e
```

That's the command you'll actually type during the exam to schedule your script.

Example cron entry (run every 5 minutes):

```
*/5 * * * * /usr/bin/python3 /home/student/backup.py
```

How to bring instructions into the exam legally (no browser, but terminal + GitHub)

Do this **before** the exam:

1. Put a **plain-text README.md** in a GitHub repo (don't rely on PDFs).
2. During exam, in terminal:

```
git clone <your_repo_url>
cd <repo>
less README.md
```

If you want, paste your **Part 2 service choice** when you walk in (which one you'll actually do), and I'll tell you the fastest execution order and the 5 commands that usually catch people.

pfSense NICs:

- Adapter 1 = **NAT** (WAN via DHCP)
- Adapter 2 = **LAN Segment** (your private Net01/whatever)