

Linux & PowerShell Professional Mastery

Practical Lab Workbook

Master the Command Line. Understand Your Systems. Secure Your Network. Automate Your Work.

30 Days: From Beginner to Advanced IT Professional
Ubuntu 24.04 LTS · Windows 11 · Windows PowerShell 5.1 · PowerShell 7.x

Edition 2026.11.08 · Days 01–02 of 30 · Built 2026-11-08

Copyright © 2026 Linux & PowerShell Professional Mastery contributors. Released under the MIT License.

Contents

1. Lab Environment & Safety Rules 3
2. Lab 01: First Contact 5
3. Lab 02: Project Workspace 10

Lab Environment & Safety Rules

Every lab in this course follows these rules. They reflect how professional IT and security teams work.

1. Authorised systems only

- Run labs on **localhost**, **WSL2**, **your own virtual machines**, or systems you have **explicit written permission** to test.
- **Never** scan, probe, brute-force or capture traffic on networks or hosts you do not own or administer. That includes employer, school, public Wi-Fi and internet hosts. Unauthorised scanning is illegal in many countries (for example under the IT Act 2000 in India and the Computer Misuse Act in the UK).
- Network discovery labs (Nmap, Days 20 and 27) target `127.0.0.1`, your own VMs, or `scanme.nmap.org`, which the Nmap project explicitly permits for light testing.

2. Snapshots and backups before changes

Before any lab that changes **permissions, firewall rules, services, users or system configuration**:

- **VM:** take a snapshot.
- **WSL:** `wsl --export Ubuntu-24.04 $HOME\wsl-backup.tar`
- **Windows:** create a restore point (`Checkpoint-Computer -Description "Before Day 18 lab"` in an elevated Windows PowerShell 5.1), and export what you will change (for example `netsh advfirewall export C:\Backup\fw.wfw`).
- **Files:** copy the original with a timestamp (`cp -a file file.bak.$(date +%F)`).

3. No real credentials

- Use throwaway lab accounts (`student`, `labuser1`) and passwords that you use nowhere else.
- **Never** paste real passwords, API keys, tokens or private keys into scripts, screenshots, tickets or this course's repository.

4. Destructive commands are always explained first

Commands that delete, overwrite or disrupt (`rm -r`, `Remove-Item -Recurse`, `kill -9`, `Stop-Service`, firewall changes) are marked with a warning, explained, and preceded by a **preview** (`ls`, `-WhatIf`) step.

5. Never disable security controls as a workaround

If a lab fails because of the firewall, antivirus, SELinux/AppArmor or execution policy, **diagnose why**. Do not turn the control off. Where a control must be adjusted, the lesson shows the narrowest, reversible change, and how to undo it.

6. Least privilege

Work as a normal user. Use `sudo` or an elevated PowerShell only for the specific command that needs it, then return to the normal session.

Recommended folder layout

```
~/lpm-lab/            # Linux practice root
$HOME\lpm-lab\        # Windows practice root
    day01/ day02/ ...  # one folder per day
    reports/           # output of course scripts
```

Lab 01: First Contact

Field	Value
Lab ID	LAB-01
Related lesson	Day 01: Introduction to Linux and PowerShell
Difficulty	Beginner
Time	45–60 minutes
Risk level	None. Read-only commands only.

Objective

Become comfortable opening both shells, identifying exactly **which user, machine, OS and shell** you are working on, and using built-in help to discover command options. You will finish by writing a one-line identity report you could paste into a change ticket.

Required Environment

- **Linux:** Ubuntu 24.04 LTS (VM, WSL2 or physical) with a normal (non-root) user account.
- **Windows:** Windows 11 with Windows PowerShell 5.1. PowerShell 7 is optional for Task 1.3.
- The course repository downloaded or cloned (for Task 5), or just the two script files from the Script Library.

Initial Conditions

- You are logged in as a normal user, not root or Administrator.
- No files need to exist. You will create one output file in your home directory: `lab01-report.txt`.

Step-by-Step Instructions

Task 1 — Open the shells and identify them

Linux

1. Press Ctrl+Alt+T (or open the Ubuntu tab in Windows Terminal).
2. Run:

```
echo "$SHELL"
echo "$BASH_VERSION"
```

Windows

1. Open **Windows PowerShell** (not PowerShell 7) from the Start menu.

2. Run:

```
$PSVersionTable.PSVersion
$PSVersionTable.PSEdition
```

3. If PowerShell 7 is installed, run `pwsh` in the same window, repeat both commands, then type `exit` to return.

Task 2 — Record identity and system facts

Linux

```
whoami
hostname
pwd
uname -r
uname -m
grep PRETTY_NAME /etc/os-release
date -u -Iseconds
```

Windows

```
whoami
$env:COMPUTERNAME
Get-Location
Get-ComputerInfo -Property OsName, OsVersion, OsArchitecture
(Get-Date).ToUniversalTime().ToString('o')
```

Task 3 — Use help to discover options (no web search)

Linux

1. Use `man ls` (search inside with `/`) to find the options for: long format, show hidden files, human-readable sizes, sort by modification time.
2. Use `ls --help | grep -i -- '--reverse'` to confirm the reverse-sort option.
3. Run the combined command and list your home directory sorted newest-first, including hidden files.

Windows

1. Run `Get-Help Get-ChildItem -Parameter Force` and `Get-Help Get-ChildItem -Parameter Depth`.
2. Run `Get-Command -Noun ChildItem` and note every command returned.
3. List your home folder, including hidden items, sorted newest-first: `Get-ChildItem $HOME -Force | Sort-Object LastWriteTime -Descending | Select-Object -First 10`

Task 4 — One-line identity report

Build a single command that prints `user@host | OS | kernel/build | UTC time`, then save it to `lab01-report.txt` in your home directory.

Task 5 — Compare with the course script

Run the course's system-information script (read it first: never run scripts you have not read):

Linux

```
cd ~/linux-powershell-professional-mastery/scripts/linux
less system-info.sh          # read it, press q to quit
bash system-info.sh
```

Windows

```
Set-Location ~\linux-powershell-professional-mastery\scripts\powershell
Get-Content .\System-Info.ps1 | more
powershell.exe -NoProfile -ExecutionPolicy Bypass -File .\System-Info.ps1
```

`-ExecutionPolicy Bypass` applies **only to this one process**. It does not change the machine policy. Execution policy is explained on Day 19.

Expected Results

- Task 1: Linux shows `/bin/bash` and a version like `5.2.21(1)-release`. Windows PowerShell shows `5.1.x` / `Desktop`, and PowerShell 7 shows `7.x` / `Core`.
- Task 2: seven (Linux) or five (Windows) facts, all matching your machine.
- Task 3: Linux combined command `ls -laht ~`. Windows lists ten items, newest first.
- Task 4: a file containing one line such as `student@ubuntu-lab | Ubuntu 24.04.1 LTS | 6.8.0-48-generic | 2026-11-08T10:15:00+00:00`.
- Task 5: the script prints a report whose user, host, OS and time match your manual results.

Verification Steps

Linux

```
cat ~/lab01-report.txt
wc -l ~/lab01-report.txt          # should print 1
```

Windows

```
Get-Content $HOME\lab01-report.txt
(Get-Content $HOME\lab01-report.txt).Count    # should print 1
```

Troubleshooting

Problem	Fix
`grep: /etc/os-release: No such file or directory`	You are not on Linux (perhaps Git Bash on Windows). Use a real Ubuntu terminal or WSL.
`Get-ComputerInfo` is slow	Normal: it queries WMI. Keep `-Property` short.
`Get-Help` shows only syntax	Run `Update-Help` in an elevated Windows PowerShell, or use `Get-Help ... -Online`.
`.\System-Info.ps1 cannot be loaded because running scripts is disabled`	Use the `powershell.exe -ExecutionPolicy Bypass -File` form shown above (process scope only).
`bash: system-info.sh: No such file or directory`	You are in the wrong directory. Use `pwd`, then `cd` to `scripts/linux`.

Questions

1. Which PowerShell edition did you open first, and how did you prove it?
2. Is your kernel release the same as your Ubuntu version? Explain the difference.
3. Which `ls` options did you combine in Task 3, and what does each do?
4. Why is the report time in UTC?
5. What did the system-info script report that you did not collect manually?

Solution

Show full solution

Task 1

```
echo "$SHELL"          # /bin/bash
echo "$BASH_VERSION"   # e.g. 5.2.21(1)-release
```

```
$PSVersionTable.PSVersion   # Major 5 Minor 1 ...
$PSVersionTable.PSEdition   # Desktop  (Core in PowerShell 7)
```

Task 3

```
ls -laht ~     # -l long, -a all (hidden), -h human sizes, -t newest first
ls -lahtr ~    # -r reverses: oldest first
```

Task 4 — Linux

```
echo "$(whoami)@$(hostname) | $(. /etc/os-release && echo "$PRETTY_NAME") | $(uname -r) | $
(date -u -Iseconds)" > ~/lab01-report.txt
cat ~/lab01-report.txt
```

`$( ... )` is **command substitution**: Bash runs the inner command and inserts its output. `. /etc/os-release` loads the file's variables (such as `PRETTY_NAME`) into the subshell.

Task 4 — Windows

```
$os = Get-ComputerInfo -Property OsName, OsVersion
"$env:USERNAME@$env:COMPUTERNAME | $($os.OsName) | $($os.OsVersion) | $((Get-Date).ToUniversalTi
me().ToString('o'))" |
    Out-File -FilePath $HOME\lab01-report.txt -Encoding utf8
Get-Content $HOME\lab01-report.txt
```

`$( ... )` inside a double-quoted string is a **subexpression**: PowerShell evaluates it and inserts the result.

Answers to questions

1. Usually Windows PowerShell 5.1, proven by `$PSVersionTable.PSEdition` = `Desktop`.
2. No. The kernel release (e.g. `6.8.0-48-generic`) is the Linux kernel version. Ubuntu 24.04 is the distribution version: the whole collection of kernel, tools and packages.
3. `-l` long listing, `-a` include hidden, `-h` human-readable sizes, `-t` sort by time.
4. UTC is unambiguous across time zones and daylight-saving changes, and it sorts correctly.
5. Typically CPU model, core count, memory, disk usage, uptime and IP addresses. These are covered in later days.

Lab 02: Project Workspace

Field	Value
Lab ID	LAB-02
Related lesson	Day 02: File and Directory Management
Difficulty	Beginner
Time	60–75 minutes
Risk level	Low. All changes stay inside `~/lab02`. Do **not** run any command outside that folder.

Objective

Create a standard project workspace, populate it with sample files, take a correct backup, reorganise files with wildcards, and perform a safe, previewed cleanup in **both** Bash and PowerShell.

Required Environment

- Ubuntu 24.04 LTS terminal (normal user) and/or Windows 11 PowerShell (5.1 or 7, **not** elevated).
- About 1 MB free disk space.

Initial Conditions

- `~/lab02` (Linux) / `$HOME\lab02` (Windows) **must not exist**. Check first:
  - Bash: `ls -d ~/lab02` should report `No such file or directory`.
  - PowerShell: `Test-Path $HOME\lab02` should return `False`.
- If it exists from an earlier attempt, rename it (`mv ~/lab02 ~/lab02.old`) rather than deleting it.

Step-by-Step Instructions

1. **Create the tree** `lab02/webapp/{config,logs,src,docs}` plus `lab02/backups`, using **one** command per shell.
2. **Create files**:
   - `config/app.conf`, `config/db.conf`
   - `logs/access-2026-11-01.log`, `logs/access-2026-11-02.log`, `logs/error-2026-11-02.log`, `logs/debug.tmp`, `logs/cache.tmp`
   - `src/main.sh` (Linux) / `src/Main.ps1` (Windows)
   - `docs/README.txt` containing the text `Webapp documentation`
3. **Back up the config folder** to `backups/config-YYYY-MM-DD` (today's date) so that the backup folder **directly** contains `app.conf` and `db.conf`, not a nested `config` folder.
4. **Reorganise logs**: create `logs/archive` and move only the `access-*` logs from **2026-11-01** into it.
5. **Rename** `docs/README.txt` to `docs/README.md`.
6. **Preview and then delete** all `*.tmp` files in `logs`.
7. **Attempt to remove** `webapp/src` with the "only if empty" command and observe that it refuses.

8. **Display the final tree.**

Commands to Execute

Try each step yourself first. Commands are in the Solution.

Expected Results

Final structure (both platforms, apart from the script file name):

```
lab02
├── backups
│   └── config-2026-11-08
│       ├── app.conf
│       └── db.conf
└── webapp
    ├── config
    │   ├── app.conf
    │   └── db.conf
    ├── docs
    │   └── README.md
    ├── logs
    │   ├── access-2026-11-02.log
    │   ├── archive
    │   │   └── access-2026-11-01.log
    │   └── error-2026-11-02.log
    └── src
        └── main.sh
```

Verification Steps

Bash

```
cd ~/lab02
find . | sort
ls webapp/logs/*.tmp 2>/dev/null | wc -l          # expect 0
ls backups/config-*/                               # expect app.conf db.conf
cat webapp/docs/README.md                          # expect "Webapp documentation"
```

PowerShell

```
Set-Location $HOME\lab02
Get-ChildItem -Recurse | Select-Object -ExpandProperty FullName | Sort-Object
(Get-ChildItem webapp\logs -Filter *.tmp).Count    # expect 0
Get-ChildItem backups\config-* | Get-ChildItem -Name   # expect app.conf db.conf
Get-Content webapp\docs\README.md                  # expect "Webapp documentation"
```

Troubleshooting

Symptom	Cause	Fix
Backup contains `config-.../config/app.conf`	Destination folder already existed when you copied the directory	Delete that backup folder (preview first) and copy the **contents**: `cp -r webapp/config/. backups/config-$(date +%F)/`
`mv: cannot stat 'webapp/logs/access-2026-11-01*'`	Wrong directory or typo	`pwd`; use tab completion
PowerShell `Rename-Item` error about the path	`-NewName` given a path	Use only `README.md` as the new name
`rmdir` succeeded on `src`	`src` was empty because step 2 was skipped	Recreate the file and repeat step 7
Files show up in `C:\Windows\System32`	You opened an elevated PowerShell, which starts there	Close it; use a normal window; `Set-Location ~`

Questions

1. Why did step 3 require copying the folder's **contents**?
2. What would have happened in step 6 if you had typed `rm logs/*.tmp *` by accident? How does previewing protect you?
3. Why does `rmdir` (and `Remove-Item` without `-Recurse`) refuse to delete `src`, and why is that useful?
4. Which PowerShell parameter showed you what would be deleted without deleting it?
5. How would you make step 1 safe to run twice?

Solution

Bash solution

```bash
# 1. tree (idempotent)
mkdir -p ~/lab02/webapp/{config,logs,src,docs} ~/lab02/backups
cd ~/lab02/webapp

# 2. files
touch config/{app,db}.conf
touch logs/access-2026-11-0{1,2}.log logs/error-2026-11-02.log logs/{debug,cache}.tmp
touch src/main.sh
echo "Webapp documentation" > docs/README.txt

# 3. backup the CONTENTS of config into a dated folder
backup_dir=~/lab02/backups/config-$(date +%F)
mkdir -p "$backup_dir"
cp -av config/. "$backup_dir"/

# 4. reorganise logs
mkdir -p logs/archive
mv -v logs/access-2026-11-01*.log logs/archive/

# 5. rename
mv -v docs/README.txt docs/README.md

# 6. preview, then delete
ls logs/*.tmp
rm -iv logs/*.tmp

# 7. refuses because src is not empty
```

```
rmdir src          # rmdir: failed to remove 'src': Directory not empty

# 8. final tree
cd ~/lab02 && (tree 2>/dev/null || find . | sort)
```

PowerShell solution

```
# 1. tree (idempotent with -Force for directories)
$base = Join-Path $HOME 'lab02'
'webapp\config','webapp\logs','webapp\src','webapp\docs','backups' |
    ForEach-Object { New-Item -Path (Join-Path $base $_) -ItemType Directory -Force | Out-Null }
Set-Location (Join-Path $base 'webapp')

# 2. files
'config\app.conf','config\db.conf',
'logs\access-2026-11-01.log','logs\access-2026-11-02.log','logs\error-2026-11-02.log',
'logs\debug.tmp','logs\cache.tmp','src\Main.ps1' |
    ForEach-Object { New-Item -Path $_ -ItemType File | Out-Null }
New-Item -Path docs\README.txt -ItemType File -Value 'Webapp documentation' | Out-Null

# 3. backup the CONTENTS of config
$backupDir = Join-Path $base ("backups\config-{0}" -f (Get-Date -Format 'yyyy-MM-dd'))
New-Item -Path $backupDir -ItemType Directory -Force | Out-Null
Copy-Item -Path config\* -Destination $backupDir -PassThru | Select-Object Name

# 4. reorganise logs
New-Item -Path logs\archive -ItemType Directory -Force | Out-Null
Move-Item -Path logs\access-2026-11-01*.log -Destination logs\archive -PassThru

# 5. rename (name only, not a path)
Rename-Item -Path docs\README.txt -NewName README.md

# 6. preview, simulate, delete
Get-ChildItem logs -Filter *.tmp
Remove-Item -Path logs\*.tmp -WhatIf
Remove-Item -Path logs\*.tmp

# 7. Remove-Item on a non-empty folder without -Recurse asks for confirmation. Answer N.
Remove-Item -Path src

# 8. final tree
Set-Location $base
Get-ChildItem -Recurse | Select-Object -ExpandProperty FullName
```

Answers to questions

1. The backup folder already existed (we created it first). Copying a *directory* into an existing directory nests it. Copying `config/.` (Bash) or `config\*` (PowerShell) copies only the contents.
2. The extra `*` would expand to **every** file in the current directory, and `rm` would delete them all. Running `ls` with the exact same pattern first shows the full list before anything is deleted.
3. They only remove empty directories by default. That makes cleanup safe: if something unexpected is inside, nothing is lost.
4. `-WhatIf`.
5. Use `mkdir -p` / `New-Item -ItemType Directory -Force`. Both succeed silently if the directories already exist.

Clean-up (optional)

```
ls -R ~/lab02 && rm -rI ~/lab02
```

```
Remove-Item $HOME\lab02 -Recurse -WhatIf
Remove-Item $HOME\lab02 -Recurse
```