# Linux Users, Groups, and sudo

Hands-On Beginner Lab Using JSLinux Fedora 33 RISC-V by Noe Tovar

**Purpose**

This focused beginner lab explains Linux user accounts, groups, privilege checks, sudo concepts, and safe administrative commands. The exercises are designed for a disposable browser-based Linux training VM and include checkpoints so the learner can complete each lab one step at a time.

| | |
|---|---|
| **Audience** | Beginner Linux learner |
| **Estimated time** | 2 to 3 hours with careful practice |
| **Training focus** | Users, groups, identity commands, sudo, and safe privilege elevation |
| **Lab environment** | JSLinux Fedora 33 RISC-V virtual machine in a web browser |
| **Prerequisites** | Completion of basic file/directory labs is helpful but not required |

## Training Roadmap

Complete these labs in order. Some account-management commands require root privileges. Because browser VMs vary, each admin lab includes a quick check and a safe fallback explanation.

| Lab | Topic | Main commands | Outcome |
|---|---|---|---|
| **0** | Set up JSLinux | browser, prompt, whoami, pwd | Working shell |
| **1** | Identity basics | whoami, id, groups, logname | Know your account |
| **2** | User database files | /etc/passwd, /etc/shadow | Understand where accounts are defined |
| **3** | Group database files | /etc/group, getent group | Understand groups |
| **4** | Root and privilege model | id -u, su, sudo concept | Know normal vs admin access |
| **5** | sudo discovery | command -v sudo, sudo -l | Check sudo availability |
| **6** | Create test users | useradd, passwd, userdel | Manage lab accounts |
| **7** | Create test groups | groupadd, groupdel | Manage lab groups |
| **8** | Add users to groups | usermod -aG, gpasswd | Assign membership |
| **9** | Group permissions practice | chgrp, chmod, newgrp | Use groups with files |
| **10** | sudoers concept | visudo, /etc/sudoers.d | Understand sudo rules |
| **11** | Audit and cleanup | last, passwd -S, userdel | Review and remove lab work |
| **12** | Final challenge | all prior commands | Build a mini admin workflow |

## Important Safety Notes Before You Begin

- Practice account and sudo commands only in the disposable JSLinux VM unless you have approval and understand the impact.
- If a command says permission denied, that is part of learning. Do not force commands on a production system.
- The root account can change or delete anything. Always verify what system you are connected to before running administrative commands.
- Some JSLinux images may not include sudo or may already run as root. If sudo is unavailable, complete the read-only labs and use the optional root-only sections only inside the VM.

**Beginner reminder**

sudo means “run this command with elevated privileges according to sudo policy.” It is not a magic unlock. The system must have sudo installed, and your account must be allowed to use it.

# Lab 0: Set Up the Browser-Based Linux Training Environment

### Goal

Open the training VM and confirm you can run commands from the shell.

**Commands you will practice:** whoami, pwd, date

Step 1: Open the JSLinux Fedora 33 RISC-V training VM:

https://bellard.org/jslinux/vm.html?cpu=riscv64&url=fedora33-riscv.cfg&mem=256

1. Wait for the VM to boot completely.
2. Click inside the terminal area so keyboard input goes to Linux.
3. If prompted to log in, use the login information displayed on the JSLinux page or console. If you are already at a prompt, continue.
4. Run the starter commands below.

```
whoami
pwd
date
```

☐ I opened the JSLinux VM.

☐ I can type at the Linux prompt.

☐ I ran whoami, pwd, and date successfully.

# Lab 1: Identify Your Current User and Groups

### Goal

Learn how to identify who you are logged in as and which groups apply to your session.

**Commands you will practice:** whoami, id, groups, logname

```
whoami
id
groups
logname 2>/dev/null || echo "logname may not be available in this environment"
```

### What to observe

whoami shows the effective username. id shows the user ID, primary group ID, and supplementary groups. groups prints group membership in a simple format.

☐ I identified my current username.

☐ I found my UID and GID using id.

☐ I listed my groups.

# Lab 2: Read the Local User Account Database

### Goal

Inspect /etc/passwd safely and understand the fields that define local accounts.

**Commands you will practice:** cat, head, grep, cut, /etc/passwd

```
head /etc/passwd
grep "^root:" /etc/passwd
cut -d: -f1,3,4,6,7 /etc/passwd | head
```

**/etc/passwd fields**

Each line is colon-separated. Common fields are username, password placeholder, UID, GID, comment, home directory, and login shell. Password hashes are not stored here on modern Linux systems.

- ☐ I viewed the first lines of /etc/passwd.
- ☐ I found the root account line.
- ☐ I used cut to display selected fields.

## Lab 3: Read the Local Group Database

**Goal**

Inspect groups and see how users can be associated with each group.

**Commands you will practice:** cat, grep, getent, /etc/group

```
head /etc/group
grep "^root:" /etc/group
getent group root 2>/dev/null || grep "^root:" /etc/group
cut -d: -f1,3,4 /etc/group | head
```

**/etc/group fields**

Each group line is colon-separated. Common fields are group name, password placeholder, GID, and a comma-separated list of members. A user can have one primary group and multiple supplementary groups.

- ☐ I viewed the first lines of /etc/group.
- ☐ I found the root group.
- ☐ I understand that groups help assign access to multiple users.

## Lab 4: Understand Root, Normal Users, and Privilege Checks

**Goal**

Learn the difference between normal access and administrative access.

**Commands you will practice:** id -u, su, sudo concept

```
whoami
id -u
if [ "$(id -u)" -eq 0 ]; then echo "You are root in this VM."; else echo "You are a regular
user."; fi
```

**Root concept**

UID 0 is root. Root can administer the system. In real environments, administrators usually avoid logging in as root directly and use sudo to run specific commands with accountability.

Optional: See whether su is available. Do not switch users unless you know the password.

```
command -v su || echo "su is not available"
```

- ☐ I checked my UID.

☐ I know that UID 0 means root.

☐ I understand why privilege elevation must be handled carefully.

## Lab 5: Check Whether sudo Is Installed and Allowed

### Goal

Discover whether sudo exists in the VM and whether your account can use it.

**Commands you will practice:** command -v sudo, sudo -l, sudo whoami

```
command -v sudo || echo "sudo is not installed in this VM"
sudo -l 2>/dev/null || echo "sudo is unavailable or your user is not allowed to use it"
sudo whoami 2>/dev/null || echo "sudo whoami did not run"
```

### Interpreting results

If sudo whoami prints root, sudo worked. If it asks for a password, that is normal on many systems. If it says command not found or permission denied, use the read-only parts of this guide or continue as root only inside the JSLinux VM.

☐ I checked whether sudo exists.

☐ I attempted sudo -l.

☐ I understand that sudo requires both software and policy permission.

## Lab 6: Create and Remove Test Users in the Lab VM

### Goal

Practice user lifecycle commands in a disposable environment.

**Commands you will practice:** useradd, passwd, id, userdel

### Root-only lab

This lab requires root privileges. If you are not root and sudo does not work, read the commands and continue to the next lab. Do not run these on production servers without approval.

```
# Check if you are root first
id -u

# Create a test user only if you are root or sudo works
sudo useradd -m labuser1 2>/dev/null || useradd -m labuser1
id labuser1
ls -ld /home/labuser1

# Optional password command may be interactive
sudo passwd labuser1 2>/dev/null || passwd labuser1

# Remove the test user when finished with this lab section
sudo userdel -r labuser1 2>/dev/null || userdel -r labuser1
```

☐ I checked whether I had root/admin privileges.

☐ I created a test user, or I understood why permission was denied.

☐ I verified the test user with id.

☐ I removed the test user when finished.

# Lab 7: Create and Remove Test Groups

**Goal**

Practice group lifecycle commands safely.

**Commands you will practice:** groupadd, getent group, groupdel

**Root-only lab**

Creating and deleting groups requires administrative privileges. Use a disposable VM only.

```
sudo groupadd labteam 2>/dev/null || groupadd labteam
getent group labteam 2>/dev/null || grep "^labteam:" /etc/group
sudo groupdel labteam 2>/dev/null || groupdel labteam
getent group labteam 2>/dev/null || echo "labteam removed or not found"
```

☐ I created a test group.

☐ I verified it with getent or /etc/group.

☐ I removed the test group.

# Lab 8: Add a User to a Group

**Goal**

Create a user and group, assign membership, then verify the result.

**Commands you will practice:** useradd, groupadd, usermod -aG, id, groups

**Why -aG matters**

When using usermod with supplementary groups, -aG appends the group. Using -G without -a can replace existing supplementary groups. That can accidentally remove access.

```
sudo groupadd projectgrp 2>/dev/null || groupadd projectgrp
sudo useradd -m projectuser 2>/dev/null || useradd -m projectuser
sudo usermod -aG projectgrp projectuser 2>/dev/null || usermod -aG projectgrp projectuser
id projectuser
groups projectuser

# Cleanup at the end of this lab
sudo userdel -r projectuser 2>/dev/null || userdel -r projectuser
sudo groupdel projectgrp 2>/dev/null || groupdel projectgrp
```

☐ I created a test group and user.

☐ I added the user to the group using usermod -aG.

☐ I verified membership with id or groups.

☐ I cleaned up the test user and group.

# Lab 9: Use Groups with File Permissions

**Goal**

See how group ownership and group permissions work with files and directories.

**Commands you will practice:** chgrp, chmod, ls -l, ls -ld

### Root-only plus file-permission lab

This lab creates a temporary group and directory, changes group ownership, and applies group permissions. Use only in the training VM.

```
cd ~
mkdir -p group-permission-lab
echo "Shared team note" > group-permission-lab/team-note.txt

sudo groupadd sharegrp 2>/dev/null || groupadd sharegrp
sudo chgrp -R sharegrp group-permission-lab 2>/dev/null || chgrp -R sharegrp group-permission-
lab
chmod 770 group-permission-lab
chmod 660 group-permission-lab/team-note.txt
ls -ld group-permission-lab
ls -l group-permission-lab/team-note.txt

# Cleanup
chmod 700 group-permission-lab
rm -f group-permission-lab/team-note.txt
rmdir group-permission-lab
sudo groupdel sharegrp 2>/dev/null || groupdel sharegrp
```

### Permission meaning

770 on a directory means owner and group have read/write/execute. Others have no access. 660 on a file means owner and group can read/write, while others have no access.

- ☐ I created a shared test folder.
- ☐ I changed group ownership with chgrp.
- ☐ I applied group-focused permissions.
- ☐ I cleaned up the lab folder and group.

## Lab 10: Understand sudoers and /etc/sudoers.d Safely

### Goal

Learn how sudo rules are structured without making risky edits.

**Commands you will practice:** sudo -l, visudo, /etc/sudoers, /etc/sudoers.d

```
sudo -l 2>/dev/null || echo "sudo rule listing unavailable"
ls -l /etc/sudoers 2>/dev/null || echo "/etc/sudoers not visible or sudo not installed"
ls -ld /etc/sudoers.d 2>/dev/null || echo "/etc/sudoers.d not available"
```

### Do not casually edit sudoers

A syntax error in sudoers can lock administrators out of sudo. On real Linux systems, use visudo because it checks syntax before saving. In many environments, administrators place custom rules in /etc/sudoers.d with strict file permissions.

Read-only example of a sudoers rule. Do not run this as a command:

```
# Example only, not for blind copy/paste:
# %wheel ALL=(ALL) ALL
# labadmin ALL=(ALL) NOPASSWD: /usr/bin/systemctl status *
```

- ☐ I checked whether sudo rules can be listed.
- ☐ I located /etc/sudoers or /etc/sudoers.d if available.

☐ I understand why visudo is safer than direct editing.

## Lab 11: Audit User and Group Information, Then Clean Up

**Goal**

Practice basic review commands and remove any lab-created leftovers.

**Commands you will practice:** getent passwd, getent group, last, passwd -S, userdel, groupdel

```
getent passwd 2>/dev/null | head || head /etc/passwd
getent group 2>/dev/null | head || head /etc/group
last 2>/dev/null | head || echo "last may not be available"
passwd -S root 2>/dev/null || echo "passwd status may not be available"

# Optional cleanup commands for any leftover lab items
sudo userdel -r labuser1 2>/dev/null || userdel -r labuser1 2>/dev/null || true
sudo userdel -r projectuser 2>/dev/null || userdel -r projectuser 2>/dev/null || true
sudo groupdel labteam 2>/dev/null || groupdel labteam 2>/dev/null || true
sudo groupdel projectgrp 2>/dev/null || groupdel projectgrp 2>/dev/null || true
sudo groupdel sharegrp 2>/dev/null || groupdel sharegrp 2>/dev/null || true
```

☐ I reviewed local users and groups.

☐ I tried basic login/account audit commands.

☐ I cleaned up any leftover lab users and groups.

## Lab 12: Final Challenge: Mini User and Group Administration Workflow

**Goal**

Apply identity, group, permission, and sudo concepts in one guided scenario.

**Commands you will practice:** groupadd, useradd, usermod -aG, mkdir, chgrp, chmod, id, ls -l

Scenario: Create a project team group, create a test project user, assign the user to the group, and secure a shared project directory.

```
# Run only inside the disposable VM and only if you have root/sudo privileges
sudo groupadd appteam 2>/dev/null || groupadd appteam
sudo useradd -m appuser1 2>/dev/null || useradd -m appuser1
sudo usermod -aG appteam appuser1 2>/dev/null || usermod -aG appteam appuser1

mkdir -p ~/appteam-project
echo "Application team project notes" > ~/appteam-project/notes.txt
sudo chgrp -R appteam ~/appteam-project 2>/dev/null || chgrp -R appteam ~/appteam-project
chmod 770 ~/appteam-project
chmod 660 ~/appteam-project/notes.txt

id appuser1
ls -ld ~/appteam-project
ls -l ~/appteam-project/notes.txt

# Cleanup
rm -f ~/appteam-project/notes.txt
rmdir ~/appteam-project
sudo userdel -r appuser1 2>/dev/null || userdel -r appuser1
sudo groupdel appteam 2>/dev/null || groupdel appteam
```

Challenge questions:

- Which command created the group?

- Which command added the user to the group without replacing existing supplementary groups?
- Which command changed the group ownership of the project folder?
- What does chmod 770 mean on the project directory?
- What does chmod 660 mean on the notes file?

☐ I completed the mini workflow or understood which privilege prevented it.

☐ I verified the user membership and permissions.

☐ I cleaned up the final challenge resources.

☐ I answered the challenge questions.

# Users, Groups, and sudo Cheat Sheet

| Command | Purpose | Example |
|---|---|---|
| **whoami** | Show effective username | whoami |
| **id** | Show UID, GID, and groups | id |
| **groups** | Show group membership | groups user1 |
| **getent passwd** | Query user database | getent passwd root |
| **getent group** | Query group database | getent group wheel |
| **useradd -m** | Create user with home directory | useradd -m labuser |
| **passwd** | Set or change password | passwd labuser |
| **userdel -r** | Remove user and home directory | userdel -r labuser |
| **groupadd** | Create group | groupadd appteam |
| **groupdel** | Remove group | groupdel appteam |
| **usermod -aG** | Append user to supplementary group | usermod -aG appteam labuser |
| **chgrp** | Change group ownership | chgrp appteam file.txt |
| **sudo -l** | List allowed sudo commands | sudo -l |
| **sudo command** | Run command with elevated privileges | sudo systemctl status sshd |
| **visudo** | Safely edit sudoers policy | visudo |

## Key Files and Directories

| Path | Purpose | Beginner note |
|---|---|---|
| **/etc/passwd** | Local user account database | Readable by normal users; does not store password hashes |
| **/etc/shadow** | Password hashes and aging info | Restricted; root-only on normal systems |
| **/etc/group** | Local group database | Shows group names, GIDs, and listed members |
| **/etc/sudoers** | Main sudo policy file | Use visudo; do not directly edit casually |
| **/etc/sudoers.d** | Drop-in sudo policy directory | Common for admin-managed sudo rules |
| **/home/username** | User home directory | Usually contains personal files and shell configuration |

**Next steps after this lab**

Repeat the read-only labs until you can explain UID, GID, primary group, supplementary group, root, sudo, and sudoers in your own words. Then practice user/group administration in a local disposable VM such as VirtualBox, VMware Workstation, Hyper-V, or a cloud sandbox before touching production Linux servers.

## Completion Sign-Off

Learner name: _____

Date completed: _____

Notes or questions:

_____

_____

_____

_____