Linux — Day 2: Users, Groups & Permissions

> Friendly, compact, and practical — focused on what DevOps engineers actually do. Each concept includes a short explanation, the exact commands you'll use, and a real DevOps scenario showing why it matters.

Quick TL;DR

- Users: human accounts, root (super), and service accounts.
- Groups: primary + supplementary — used to share permissions.
- Passwords: hashed in `/etc/shadow`, user info in `/etc/passwd`.
- Ownership & permissions control who can read/write/execute files.
- `sudo` gives controlled privilege elevation — manage via `visudo`.

---

1) Core files to know (one-line each)

- `/etc/passwd` — user records (login name, uid, gid, home, shell).
- `/etc/shadow` — secure password hashes and aging metadata.
- `/etc/group` — group definitions and group members.
- `/etc/gshadow` — secure group admin info.

Quick look:

```
getent passwd username    # show passwd entry (works with NSS)
getent shadow username    # needs root
getent group groupname
```

DevOps scenario — audit a CI runner user quickly: `getent passwd cicd` to verify UID/home/shell before adding keys.

---

2) User management (create / modify / delete)

Create a user

```
# minimal, with home and default shell
sudo useradd -m -s /bin/bash alice
sudo passwd alice
```

```
# service account (no login shell, system UID, no home)
sudo useradd -r -s /usr/sbin/nologin -M --system myservice
```

Useful options

- `-m` → create home from `/etc/skel`
- `-s` → set login shell
- `-G` → supplementary groups
- `-g` → primary group
- `-u` → set UID
- `-e` → expiry date
- `-f` → inactive days before lock

Modify user

```
sudo usermod -aG docker alice      # append to group
sudo usermod -l newname oldname     # change login name
sudo usermod -s /bin/false svcuser  # disable login interactively
```

Delete user

```
sudo userdel username               # remove account
sudo userdel -r username            # remove account + home + mail spool
```

DevOps scenario — create a `deploy` user with limited shell and place public key for CI: this avoids using root in deployment steps.

---

3) Password aging & policies

- Use `chage` or `passwd` options to manage expiration & warnings.

```
# show policy
sudo chage -l alice

# set policies
sudo chage -m 7 -M 90 -W 7 alice   # min 7 days, max 90 days, warn 7 days
# set expiration date
sudo chage -E 2026-12-31 bob

# passwd equivalents
sudo passwd -n 7 -x 90 -w 7 alice
```

DevOps scenario — enforce password rotation for on-call accounts that can access production tools; integrate with IAM where possible for automation.

---

4) Groups — share access safely

- Primary group is set in `/etc/passwd`. Supplementary groups are in `/etc/group`.

```
# create group with explicit GID
sudo groupadd -g 2000 deployers

# view groups of user
groups alice
getent group deployers

# add/remove
sudo usermod -aG deployers alice   # append to supplementary groups
sudo gpasswd -d alice deployers     # remove
```

DevOps scenario — make a `deployers` group, put CI/service accounts there, chown artifacts to `:deployers` and set group write for safe collaborative deploys.

---

5) Switching accounts (su, newgrp, sudo)

- `su user` → switch to user, keeps current env.
- `su - user` → login shell (loads user env).
- `newgrp group` → switch current shell's group ID.
- `sudo -u user command` → run single command as another user.

```
sudo -u postgres psql -c '\l'
su - deploy    # get full deploy user's environment
newgrp deployers  # temporarily change group for file creation
```

DevOps scenario — run `sudo -u prometheus` to query metrics without logging in as that account.

---

6) Ownership (chown / chgrp)

```
# set owner only
sudo chown alice /srv/app/config.yml
# set owner and group
sudo chown alice:deployers /srv/app
```

```
# recursive
sudo chown -R www-data:www-data /var/www/html
# change group only (alias)
sudo chgrp -R deployers /srv/app
```

DevOps scenario — after a deploy, ensure web files are owned by `www-data` so the web server can read them and a `deployers` group can update them.

---

7) Permissions (chmod) — symbolic & octal

Quick mapping

```
r = 4, w = 2, x = 1
rwx  = 7, rw- = 6, r-x = 5, r-- = 4
owner-group-other (u-g-o)
```

Examples

```
# symbolic
chmod u=rw,g=r,o=r file        # 644
chmod u=rwx,g=rx,o= file       # 750
# octal
chmod 644 file                 # owner rw, group r, other r
chmod 750 dir                  # owner rwx, group rx, other none
```

Special bits

- **setuid (s on owner x)** → executable runs with owner privileges
- **setgid (s on group x)** → executable/dir runs with group privileges / new files inherit group
- **sticky bit (t on others execute)** → on dirs, only owners can delete their files (e.g., `/tmp`)

```
# setgid on directory so new files inherit group
sudo chmod g+s /srv/app
# sticky bit on shared upload dir
sudo chmod +t /srv/uploads
```

DevOps scenario — use setgid for team folders so all files remain group-writable by `deployers`; sticky bit on shared temp upload directories prevents other users from deleting each other's files.

---

8) umask — default permissions for newly created files

- `umask` shows the subtraction mask. Common default `umask 0022` (files 644, dirs 755) or `0002` for shared groups (files 664, dirs 775).

```
umask                          # show
umask 0002                     # set for group writable workflow
# persist by adding to /etc/profile or user shell RC (~/.profile or ~/.config/
fish/config.fish)
```

DevOps scenario — set `umask 0002` on build servers so artifacts are group editable by CI/CD team members.

---

9) Sudo — safe privilege elevation

- Edit sudoers with `visudo` or drop a file in `/etc/sudoers.d/`.

```
# allow user to run specific commands without password
echo "alice ALL=(ALL) NOPASSWD: /usr/bin/systemctl restart myapp" | sudo tee /
etc/sudoers.d/alice_myapp
sudo visudo -f /etc/sudoers.d/myrules   # safer edit mode
```

Best practice: prefer least-privilege (only required commands), use `NOPASSWD` sparingly, and keep rules in `/etc/sudoers.d/` for auditability.

DevOps scenario — grant Jenkins or a deploy user only the exact `systemctl` + `rsync` commands needed for deployments.

---

10) Auditing & troubleshooting commands

```
id alice                # uid/gid and groups
getent passwd alice     # view entry via NSS
getent group deployers
sudo passwd -S alice    # show passwd lock status
sudo chage -l alice     # show password aging
find / -uid 1001        # find files owned by UID (useful for orphaned files)
ls -l /path/to/file     # check owner & perms
```

DevOps scenario — after migrating a user between servers, `find / -uid OLD_UID` helps locate leftover files and reassign ownership.

---

11) Quick playbooks (copy-paste ready)

A) Create a locked service account (no login)

```
sudo useradd -r -s /usr/sbin/nologin --no-create-home --system deploybot
```

B) Create deploy user + group and prepare directory

```
sudo groupadd deployers
sudo useradd -m -s /bin/bash -G deployers deploy
sudo mkdir -p /srv/myapp
sudo chown -R deploy:deployers /srv/myapp
sudo chmod 2775 /srv/myapp  # setgid so files inherit group
```

C) Allow deploy to restart service (least-privilege sudo)

```
echo "deploy ALL=(root) NOPASSWD: /bin/systemctl restart myapp.service" | sudo
tee /etc/sudoers.d/deploy_myapp
sudo chmod 440 /etc/sudoers.d/deploy_myapp
```

D) Audit and fix orphan files after UID change

```
# find files owned by old UID and chown to new user
sudo find / -xdev -uid 1500 -exec chown deploy:deployers {} +
```

---

12) Security dos & don'ts

- DO use key-based SSH access and disable password login for important accounts.
- DO keep service accounts unprivileged (`/usr/sbin/nologin`).
- DO manage sudoers in `/etc/sudoers.d/` and use `visudo` to validate.
- DON'T use shared human accounts for auditing (each person should have own login).
- DON'T give NOPASSWD on broad command sets.

---

13) Rapid cheatsheet (commands you must memorize)

```
useradd/usermod/userdel
passwd, chage
groupadd/groupmod/groupdel, gpasswd
```

```
chown, chgrp
chmod (symbolic & octal), chmod g+s, chmod +t
umask
visudo /etc/sudoers.d/
getent passwd | getent group
id, groups, whoami
```

---

14) 60-second tasks to practice

1. Create a `readonly` service user with no shell.
2. Create `deployers` group and add your test user.
3. Create `/srv/demoapp`, set `g+s`, and show new files inherit group.
4. Add a sudoers file allowing only `systemctl restart demoapp.service` for your deploy user.