Write the ultimate user and group management guide in Linux systems in extreme details covering multiple topics and scenarios (service users/accounts, guests, temporary delegates, management users with sub UIDs, daily-drivers + the corresponding group concepts, and way more).

Here's a structured foundation for your **ultimate Linux user and group management guide**. This is long and exhaustive by design, covering every major concept, edge case, and scenario.

Ultimate Guide to User and Group Management in Linux Systems

1. Core Concepts

1.1 Users

- **Root user (`UID=0`)**: Omnipotent. All permissions bypassed. Always protect and restrict access.
- **System users**: Accounts with `UID < 1000` (on most distros) created automatically for daemons (e.g., `www-data`, `sshd`). Typically locked (no login shell, no password).
- **Normal users**: Human accounts for daily use (`UID ≥ 1000` by default).
- **Virtual/Service-specific users**: Created for applications, temporary roles, automation. May have restricted shells (`/usr/sbin/nologin`, `/bin/false`).

1.2 Groups

- **Primary group**: Assigned at user creation. Same `GID` as `UID` for private groups (common on Debian/Ubuntu).
- **Supplementary groups**: Grant additional privileges (e.g., `sudo`, `docker`).
- **System groups**: Used for permission delegation (e.g., `wheel`, `adm`, `lp`).
- **Special groups**: Linked to kernel features (e.g., `nogroup`, `nobody`).

1.3 Identification

- Users and groups are stored in:
  - **`/etc/passwd`** → user info (UID, shell, home).
  - **`/etc/shadow`** → encrypted passwords, expiry.
  - **`/etc/group`** → group membership.
  - **`/etc/gshadow`** → group password and security info.

---

2. User Lifecycle Management

2.1 Creating Users

```
bash


# Minimal
useradd alice

# With home, shell, group, comment
useradd -m -d /home/alice -s /bin/bash -c "Alice Johnson" alice

# With UID/GID
useradd -u 1501 -g developers alice
```

2.2 Password Management

```
bash


passwd alice
passwd -l alice   # lock account
passwd -u alice   # unlock
chage -l alice    # view password aging
chage -E 2025-12-31 alice   # set expiry
```

2.3 Modifying Users

```bash

usermod -aG sudo,docker alice    # add to groups
usermod -s /bin/zsh alice        # change shell
usermod -L alice                 # lock account
usermod -U alice                 # unlock account
```

2.4 Removing Users

```bash

userdel alice            # leaves home dir
userdel -r alice         # removes home + mail spool
```

---

3. Group Lifecycle Management

3.1 Creating Groups

```bash

groupadd developers
groupadd -g 2001 testers
```

3.2 Modifying Groups

```bash

gpasswd -a alice developers    # add user
gpasswd -d alice developers    # remove user
groupmod -n engineers developers  # rename
```

3.3 Deleting Groups

```
bash

groupdel testers
```

---

4. Specialized Scenarios

4.1 Service Accounts

- Created without login shell or password:

```
bash

useradd -r -s /usr/sbin/nologin -d /var/lib/mysql mysql
```

- Restricted privileges. Use **`systemd`** unit directives (**`User=`**, **`Group=`**).
- Files owned by service accounts should be strictly controlled.

4.2 Guest Accounts

- For shared terminals/kiosks:
  - Ephemeral home directory, reset at logout.
  - Can be implemented with **`guest-account`** PAM module.
  - Example:

```
bash

useradd -m -d /tmp/guest-home -s /bin/bash guest
```

  - Hook a cleanup script in **`/etc/skel`** or via cron to purge home.

4.3 Temporary Delegates

- Accounts with automatic expiration:

```
bash

useradd -m -e 2025-09-30 tempdelegate
```

- Use **`chage`** for aging and lockout policy.
- For sudo delegation:
  - Create a file in **`/etc/sudoers.d/delegate`**:

```
pgsql

tempdelegate ALL=(ALL) /usr/bin/systemctl restart nginx
```

  - Limits scope to specific commands.

4.4 Management Users with subUIDs/subGIDs

- For container runtimes and user namespaces:
  - **`/etc/subuid`** and **`/etc/subgid`** define ranges:

```
makefile

alice:100000:65536
```

  - Allows containers to map their own root → unprivileged host UIDs.
- Critical for Podman, LXC, rootless Docker.
- Verify with:

```
bash

podman info | grep uid
```

4.5 Daily-driver Users

- Should not be **`root`**.
- Added to necessary groups: **`sudo`**, **`audio`**, **`video`**, **`plugdev`**, **`docker`**, **`libvirt`**.
- Login shells with customized profiles in **`~/.bashrc`** or **`~/.zshrc`**.
- Principle of least privilege: remove unused group memberships.

---

5. Authentication and Security Layers

5.1 PAM (Pluggable Authentication Modules)

- Central to Linux auth.
- Located in **`/etc/pam.d/`**.
- Common stacks: **`pam_unix.so`**, **`pam_tally2.so`**, **`pam_faillock.so`**.
- Controls: login attempts, lockouts, password quality.

5.2 SSH Access

- Restrict root login: **`/etc/ssh/sshd_config`** → **`PermitRootLogin no`**.
- User-based restrictions:

```
nginx


AllowUsers alice bob
DenyGroups guests
```

- Key-based auth in **`~/.ssh/authorized_keys`**.

5.3 Password Policies

- **`libpam-pwquality`**: enforce length/complexity.

- `/etc/security/faillock.conf`: brute-force mitigation.

5.4 SELinux / AppArmor

- Label user domains for finer controls.
- Example: `staff_u`, `user_u`, `sysadm_u`.

---

6. Advanced Group Concepts

6.1 Access Control Lists (ACLs)

- Go beyond user/group/other model.
- Example:

```bash
setfacl -m u:alice:rwx /srv/project
setfacl -m g:developers:rx /srv/project
getfacl /srv/project
```

6.2 Special Groups

- `wheel`: sudo-equivalent (RHEL/CentOS).
- `adm`: read system logs.
- `disk`: raw disk access (dangerous).
- `docker`: access Docker socket (root-equivalent).
- `systemd-journal`: journalctl read access.

---

7. Auditing and Monitoring

- Check users:

```
bash

getent passwd
```

- Check groups:

```
bash

getent group
```

- Who is logged in:

```
bash

who
w
last
```

- Open sessions:

```
bash

ps -u alice
```

- Audit login failures:

```
bash

faillog -a
journalctl _SYSTEMD_UNIT=sshd.service
```

---

8. Enterprise Integration

8.1 Centralized Identity

- LDAP, FreeIPA, Active Directory integration via SSSD or Winbind.

- Kerberos for SSO.
- NIS (legacy).

8.2 Role-Based Access

- Map enterprise roles → Linux groups.
- Example: `dev`, `qa`, `ops`, `audit`.

---

9. Best Practices

1. **Principle of least privilege**: No one gets more than needed.
2. **No shared accounts**: Use individual accounts with sudo/logging.
3. **Service separation**: Each service has its own UID/GID.
4. **Expire temporary accounts**.
5. **Lock dormant accounts** with `usermod -L`.
6. **Audit group membership regularly**.
7. **Enforce key-based SSH**.
8. **Use subUIDs for containers** to sandbox.

---

10. Quick Reference Commands

- `id alice` → UID/GID and groups.
- `groups alice` → groups only.
- `getent passwd alice` → user entry.
- `getent group developers` → group entry.
- `lslogins -u` → detailed user info.

- `vipw` / `vigr` → safe edit passwd/group.

Would you like me to expand this into a **step-by-step "playbook style" document** (with actual recommended workflows and policies for different environments like servers, desktops, container hosts, enterprise domains), or keep it as a **reference encyclopedia** as above?