Table of Contents

$ adduser

Beginner

Add a user to the system (interactive, Debian-based)

adduser is an interactive, user-friendly command for creating new user accounts. It is a higher-level wrapper around useradd that automatically creates the home directory, copies skeleton files, prompts for a password, and asks for user details.

adduser is specific to Debian/Ubuntu systems. On R...

Options & Flags

`username`	Create a new user interactively
`--disabled-password`	Create user without password prompt
`--system`	Create a system user
`--group`	Create a group instead of user
`user group`	Add existing user to group
`--home`	Specify home directory

Practical Examples

Example: Create user interactively

```
$ sudo adduser jdoe
Adding user jdoe...\nEnter new UNIX password:
```

Creates user with interactive prompts for password, name, phone, etc.

Example: Add user to group

```
$ sudo adduser jdoe sudo
Adding user jdoe to group sudo...
```

Adds existing user jdoe to the sudo group.

Example: Create without password

```
$ sudo adduser --disabled-password --gecos "John Doe" jdoe
```

Creates user non-interactively (for scripts). --gecos sets the name.

Example: Create system user

```
$ sudo adduser --system --group --no-create-home nginx
```

Creates a system user and group for running a service.

Example: Add to Docker group

```
$ sudo adduser $USER docker
```

Adds current user to docker group (logout/login required to take effect).

Tips & Best Practices

Pro Tip: adduser for groups: adduser username groupname adds a user to a group. This is simpler than usermod -aG groupname username.

Note: Debian/Ubuntu only: The interactive adduser is a Debian/Ubuntu tool. On RHEL/CentOS, adduser is just a symlink to useradd.

Warning: Group changes need re-login: After adding a user to a group, they must log out and back in (or use newgrp) for the change to take effect.

$ chage

Intermediate

Change user password expiry information

chage changes user password aging information. It manages password expiration policies including maximum age, minimum age, warning days, and account expiration.

chage is essential for enforcing password policies, meeting compliance requirements (PCI-DSS, HIPAA), and managing contractor/temporary...

Options & Flags

`-l`	List password aging info for user
`-M`	Maximum days between password changes
`-m`	Minimum days between password changes
`-W`	Warning days before password expires
`-E`	Set account expiration date
`-d`	Set date of last password change
`-I`	Days of inactivity after password expires before account ...

Practical Examples

Example: View password info

```
$ sudo chage -l jdoe
Last password change: Jan 01, 2024
Password expires: Apr 01, 2024
Account expires: never
```

Shows all password aging information for a user.

Example: Set 90-day password policy

```
$ sudo chage -M 90 -m 7 -W 14 jdoe
```

Password expires after 90 days, cannot change within 7 days, warns 14 days before.

Example: Force password change

```
$ sudo chage -d 0 jdoe
```

Sets last change date to epoch 0, forcing password change on next login.

Example: Set account expiration

```
$ sudo chage -E 2025-06-30 contractor
```

Sets the account to expire on June 30, 2025.

Example: Remove expiration

```
$ sudo chage -E -1 jdoe
```

Removes account expiration date (account never expires).

Tips & Best Practices

Pro Tip: Compliance policies: For PCI-DSS: chage -M 90 -m 1 -W 14 -I 30 user. This sets 90-day expiry, 1-day minimum, 14-day warning, 30-day inactive lock.

Note: Force change vs expire account: chage -d 0 forces password change at next login. chage -E date expires the entire account.

Warning: Users can view their own info: Users can run chage -l username to see their own password aging info. This is by design for self-service.

$ groupadd

Intermediate

Create a new group

groupadd creates a new group on the system. Groups are fundamental to Linux permissions, allowing multiple users to share access to files and resources.

groupadd requires root privileges. It creates an entry in /etc/group with an automatically assigned GID (or a specified one). After creating a ...

Options & Flags

`-g`	Specify GID (group ID)
`-r`	Create system group (low GID)
`-f`	Exit with success if group exists

Practical Examples

Example: Create a group

```
$ sudo groupadd developers
```

Creates a new group with an auto-assigned GID.

Example: Create with specific GID

```
$ sudo groupadd -g 1500 developers
```

Creates the group with GID 1500.

Example: Create system group

```
$ sudo groupadd -r myapp
```

Creates a system group with a low GID (for services).

Example: Add users to new group

```
$ sudo groupadd project-x && sudo usermod -aG project-x alice && sudo usermod -aG project-x bob
```

Creates a group and adds two users to it.

Example: Set directory group

```
$ sudo groupadd webteam && sudo chgrp webteam /var/www/project && sudo chmod g+ws /var/www/project
```

Creates group, assigns it to directory, and sets setgid.

Tips & Best Practices

Pro Tip: Use setgid for shared directories: After creating a group and assigning it to a directory, set the setgid bit: chmod g+s dir/. New files inherit the group.

Note: System vs regular groups: System groups (-r) get low GIDs and are for services. Regular groups get higher GIDs and are for users.

Warning: Group changes need re-login: After adding a user to a group, they must log out and back in. Or use newgrp groupname for the current session.

$ groups

Beginner

Print group memberships for a user

groups displays the group memberships for a user. Without arguments, it shows groups for the current user. With a username, it shows that user's groups.

groups shows the active groups in the current session, which may differ from what is stored in /etc/group if group changes have been made recen...

Options & Flags

`(no args)`	Show groups for current user
`username`	Show groups for specific user
`multiple`	Show groups for multiple users

Practical Examples

Example: Show my groups

```
$ groups
user sudo docker www-data
```

Lists all groups the current user belongs to.

Example: Show user groups

```
$ groups www-data
www-data : www-data
```

Shows groups for the www-data user.

Example: Check sudo access

```
$ groups | grep -q sudo && echo "Has sudo" || echo "No sudo"
Has sudo
```

Checks if current user has sudo group membership.

Example: Compare users

```
$ groups alice bob
alice : alice sudo docker\nbob : bob developers
```

Shows group memberships for multiple users.

Example: Verify group addition

```
$ sudo usermod -aG docker user; groups user
```

Adds user to docker group and verifies (must re-login to take effect).

Tips & Best Practices

Warning: Re-login for changes: Group changes do not take effect in the current session. Log out and back in, or use newgrp groupname.

Pro Tip: newgrp for immediate effect: After being added to a group, use newgrp groupname to activate it in the current session without re-login.

Note: groups vs id -Gn: groups shows active session groups. id -Gn shows stored groups from /etc/group. After changes, they may differ until re-login.

$ id

Beginner

Print user and group IDs

id displays user and group identity information. Without arguments, it shows the current user's UID, GID, and all group memberships. With a username argument, it shows that user's information.

id is essential for verifying user identity, checking group memberships, debugging permission issues, a...

Options & Flags

`-u`	Print only effective user ID
`-g`	Print only effective group ID
`-G`	Print all group IDs
`-n`	Print name instead of number (with -u, -g, -G)
`-r`	Print real (not effective) ID
`username`	Show info for specific user

Practical Examples

Example: Show current identity

```
$ id
uid=1000(user) gid=1000(user) groups=1000(user),27(sudo),998(docker)
```

Displays UID, GID, and all groups for the current user.

Example: Show specific user

```
$ id www-data
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Shows identity info for the www-data user.

Example: Get username

```
$ id -un
user
```

Prints just the effective username.

Example: Get numeric UID

```
$ id -u
1000
```

Prints just the numeric user ID.

Example: List all groups

```
$ id -Gn
user sudo docker www-data
```

Lists all group names the current user belongs to.

Tips & Best Practices

Pro Tip: Verify group changes: After usermod -aG, run id username to verify the group was added. The user must log out/in for groups to take effect in their session.

Note: Real vs effective ID: Effective ID determines permissions. Real ID is the original user. They differ when running SUID programs or using sudo.

Warning: Current session vs stored groups: id shows stored groups from /etc/group. Your current session may differ until re-login. Use groups command for session groups.

$ last

Intermediate

Show listing of last logged in users

last displays a list of recent user logins and system events by reading the /var/log/wtmp file. It shows login times, logout times, session duration, and the source IP or terminal.

last is essential for security auditing (who logged in and when), troubleshooting (when was the system rebooted), a...

Options & Flags

`username`	Show logins for specific user
`-n N`	Show only last N entries
`-a`	Show hostname in last column
`-i`	Show IP addresses instead of hostnames
`-x`	Show system shutdown/runlevel changes
`-d`	Translate IPs to hostnames
`-F`	Show full login/logout times

Practical Examples

Example: Show recent logins

```
$ last -n 10
admin  pts/0  192.168.1.50  Mon Jan 1 10:00  still logged in
jdoe   pts/1  10.0.0.100    Mon Jan 1 09:30 - 10:15 (00:45)
```

Shows the 10 most recent login entries.

Example: Show user logins

```
$ last admin
```

Shows all login history for the admin user.

Example: Show reboots

```
$ last reboot
reboot  system boot  5.15.0  Mon Jan 1 08:00  still running
```

Shows system reboot history.

Example: Show with IPs

```
$ last -ai -n 20
```

Shows last 20 logins with IP addresses in the last column.

Example: Show shutdowns

```
$ last -x shutdown
```

Shows system shutdown events.

Tips & Best Practices

Pro Tip: Security monitoring: Run last -ai regularly to monitor login activity. Unexpected IPs or times may indicate unauthorized access.

Note: lastb for failures: sudo lastb shows failed login attempts (from /var/log/btmp). Useful for detecting brute force attacks.

Warning: wtmp can be cleared: An attacker with root access can clear /var/log/wtmp. Use remote syslog and log monitoring for reliable auditing.

$ passwd

Beginner

Change user password

passwd changes user passwords. Without arguments, it changes your own password. With a username argument (requires root), it changes another user's password.

passwd can also lock and unlock accounts, set password expiration policies, and force password changes on next login. It interacts with /e...

Options & Flags

`username`	Change password for specific user (root only)
`-l`	Lock account (disable password)
`-u`	Unlock account
`-d`	Delete password (allow passwordless login)
`-e`	Force password change on next login
`-S`	Show password status
`-n`	Minimum days between password changes
`-x`	Maximum days before password expires

Practical Examples

Example: Change own password

```
$ passwd
Changing password for user.\nCurrent password:
```

Prompts for current password then new password.

Example: Change user password

```
$ sudo passwd jdoe
Enter new UNIX password:
```

Sets a new password for user jdoe (root only).

Example: Lock account

```
$ sudo passwd -l compromised_user
passwd: password expiry information changed
```

Locks the account - user cannot log in with password.

Example: Force password change

```
$ sudo passwd -e jdoe
```

Forces user to change password on next login.

Example: Check password status

```
$ sudo passwd -S jdoe
jdoe P 2024-01-15 0 90 7 -1
```

Shows password status: set/locked, last change, expiry info.

Tips & Best Practices

Pro Tip: Force rotation on compromise: If an account may be compromised: sudo passwd -e username forces a password change on next login.

Warning: -d removes security: passwd -d deletes the password entirely, allowing login without one. Only use for system accounts or testing.

Note: passwd vs chage: passwd sets and manages passwords. chage provides more detailed control over aging policies (min/max days, warnings).

$ su

Beginner

Switch user or become superuser

su (substitute user) switches the current user identity to another user, requiring that user's password. Without a username, it switches to root.

su creates a new shell running as the specified user. The - flag (su -) is important - it simulates a full login, loading the target user's environmen...

Options & Flags

`-`	Full login shell (load user environment)
`-c`	Execute a single command
`-s`	Use specific shell
`-l`	Same as - (login shell)
`-p`	Preserve current environment

Practical Examples

Example: Switch to root

```
$ su -
Password:
```

Switches to root with full login shell. Requires root password.

Example: Switch to user

```
$ su - www-data
```

Switches to the www-data user with their environment loaded.

Example: Run single command

```
$ su -c 'cat /etc/shadow' root
```

Runs one command as root without opening an interactive shell.

Example: Switch with specific shell

```
$ su -s /bin/bash nginx
```

Opens bash as nginx user (useful when user's default shell is /sbin/nologin).

Example: Return to original user

```
$ exit
```

Returns to the previous user after su.

Tips & Best Practices

Warning: Always use su - (with dash): su without dash keeps your current environment, which can cause confusion. su - loads the target user environment properly.

Note: sudo vs su: sudo uses YOUR password and logs commands. su requires the TARGET user password. sudo is preferred for admin tasks.

Pro Tip: Service user shells: Service accounts often have /sbin/nologin as shell. Use su -s /bin/bash username to get a shell.

$ sudo

Beginner

Execute a command as another user (typically root)

sudo (superuser do) executes a command as another user, typically root. It is the primary mechanism for administrative privilege escalation in Linux, providing fine-grained access control through the /etc/sudoers file.

sudo logs all commands for auditing, requires the user's own password (not ro...

Options & Flags

`-u`	Run as specified user
`-i`	Login shell as root
`-s`	Run a shell as root
`-l`	List allowed commands
`-k`	Invalidate cached credentials
`-v`	Extend credential timeout
`-E`	Preserve environment variables
`-n`	Non-interactive (fail if password needed)

Practical Examples

Example: Run as root

```
$ sudo apt update
```

Runs apt update with root privileges.

Example: Edit protected file

```
$ sudo nano /etc/nginx/nginx.conf
```

Opens a root-owned file for editing.

Example: Run as another user

```
$ sudo -u postgres psql
```

Opens PostgreSQL as the postgres user.

Example: Root login shell

```
$ sudo -i
```

Opens a full root login shell with root environment.

Example: Check permissions

```
$ sudo -l
(ALL : ALL) ALL
```

Shows what commands you are allowed to run with sudo.

Tips & Best Practices

Warning: Use sudo, not root login: Never log in as root directly. sudo provides auditing, limited scope, and uses your own password. Root login disables all these safeguards.

Pro Tip: sudo !! for mistakes: Forgot sudo? Type sudo !! to re-run the last command with root privileges.

Note: Password caching: sudo caches your password for 15 minutes by default. Use sudo -k to clear it immediately.

$ useradd

Intermediate

Create a new user account

useradd creates a new user account on the system. It is a low-level command that creates the user entry in /etc/passwd and /etc/shadow, optionally creating a home directory and setting initial configuration.

useradd requires root privileges. It creates the user but does not set a password - use ...

Options & Flags

`-m`	Create home directory
`-d`	Specify home directory path
`-s`	Set login shell
`-g`	Set primary group
`-G`	Set additional groups
`-c`	Set comment (full name)
`-e`	Set account expiration date
`-r`	Create system account (no home, low UID)
`-u`	Specify UID

Practical Examples

Example: Create user with home

```
$ sudo useradd -m -s /bin/bash -c "John Doe" jdoe
```

Creates a user with home directory, bash shell, and full name.

Example: Set password

```
$ sudo passwd jdoe
```

Sets password for the new user (always do this after useradd).

Example: Create with groups

```
$ sudo useradd -m -s /bin/bash -G sudo,docker developer
```

Creates a user in the sudo and docker groups.

Example: Create system account

```
$ sudo useradd -r -s /sbin/nologin -d /var/lib/myapp myapp
```

Creates a service account that cannot login interactively.

Example: Create temporary user

```
$ sudo useradd -m -e 2025-06-30 contractor
```

Creates a user account that expires on a specific date.

Tips & Best Practices

Warning: Always set a password: useradd does NOT set a password. The account is locked until you run: sudo passwd username.

Pro Tip: Use adduser on Debian/Ubuntu: adduser is an interactive wrapper that creates home directory, prompts for password, and sets up the account in one step.

Note: -m is not always default: On RHEL/CentOS, -m creates the home directory. On Debian/Ubuntu, it may be the default. Always use -m to be safe.