



Scan me
to watch these
commands animate



Initial Commit
made with git-sim on initialcommit.com

The Essential Git Cheat Sheet

Page 1 of 3

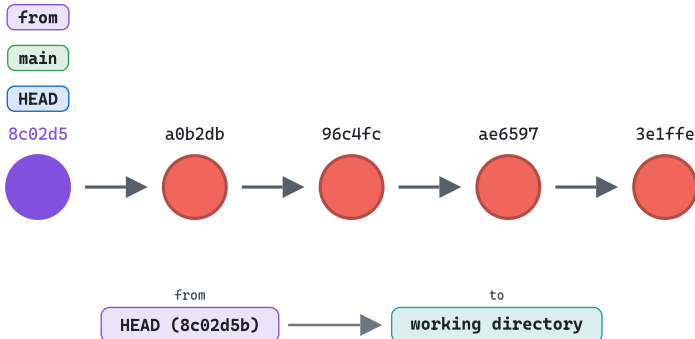
The Essential Git Cheat Sheet

Show unstaged changes

SAFE

```
git diff
```

Shows the changes in the working directory that aren't staged yet, line by line.



Unstaged changes

2 file(s), +2 -2

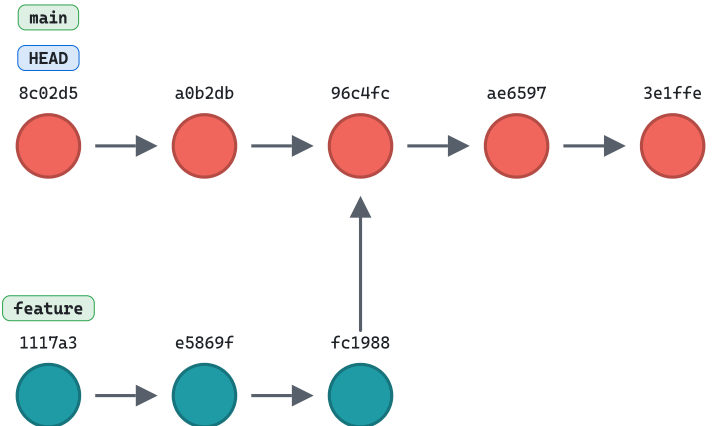
M	app.py	+1 -1	■■■■■
M	styles.css	+1 -1	■■■■■

View all branches as a text graph

SAFE

```
git log --oneline --graph --all
```

Prints the history of all branches as a text graph, one line per commit, with branch and tag names next to their commits.

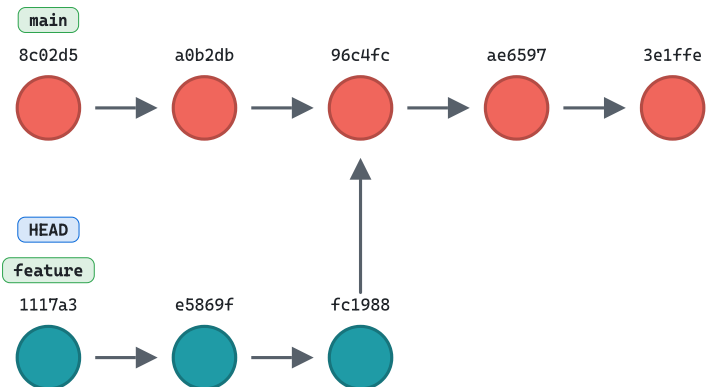


Switch to another branch

SAFE

```
git switch <branch>
```

Points **HEAD** at **<branch>** and updates the staging area and working directory to match its latest commit.

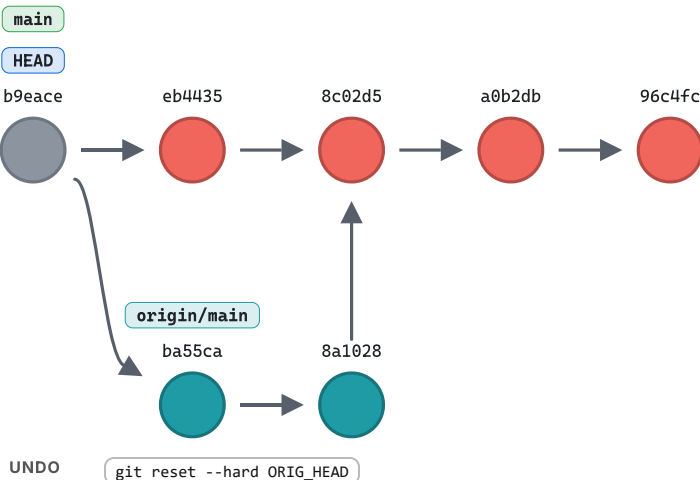


Fetch and merge remote commits

SAFE

```
git pull
```

Fetches the active branch's upstream and merges it into the active branch. When both have new commits, Git creates a merge commit.

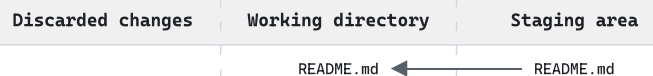


Unstage a file's changes

SAFE

```
git restore --staged <file>
```

Unstages the changes to **<file>**, so they're not in the next commit. The file in the working directory doesn't change.





Scan me
to watch these
commands animate



Initial Commit

made with git-sim on initialcommit.com

The Essential Git Cheat Sheet

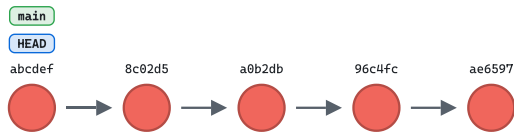
Page 2 of 3

Commit your staged changes

SAFE

```
git commit -m "<message>"
```

Creates a new commit from the staged changes with the message `<message>`, and moves the active branch to it.



Working directory	Staged files	New commit
app.py	README.md	README.md

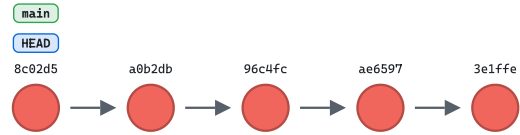
UNDO `git reset --soft HEAD~1`

Stash your uncommitted changes

SAFE

```
git stash
```

Saves the staged and unstaged changes to tracked files as a new stash entry, and resets the staging area and working directory to the last commit.



Stashed changes	Working directory	Staging area
app.py README.md	app.py	README.md

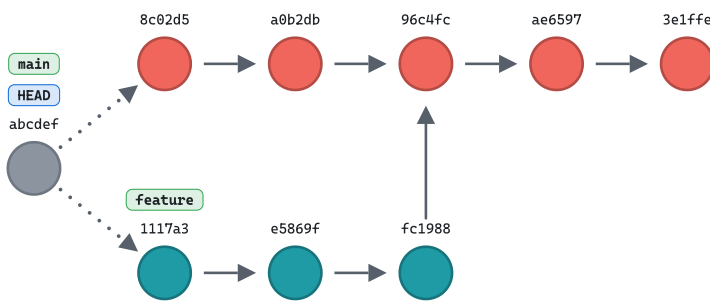
UNDO `git stash pop`

Merge a branch into the active branch

SAFE

```
git merge <branch>
```

Merges `<branch>` into the active branch. When both have new commits, Git creates a merge commit with two parents.



UNDO `git reset --hard ORIG_HEAD`

Discard unstaged changes to a file

DESTRUCTIVE

```
git restore <file>
```

Discards the unstaged changes to `<file>`, restoring it from the staging area.

Discarded changes	Working directory	Staging area
app.py	app.py styles.css	

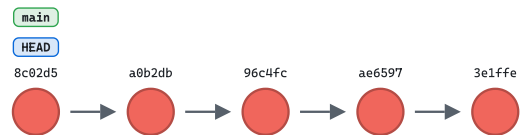
UNDO No way back
WATCH Unstaged edits aren't saved in Git, so they can't be recovered. Run `git stash` first if you're not sure.

Apply and drop the latest stash entry

SAFE

```
git stash pop
```

Applies the latest stash entry to the working directory and removes it from the stash. If applying it conflicts, the entry is kept.



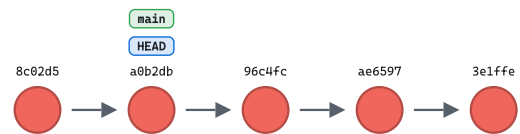
Stashed changes	Working directory	Staging area
app.py	app.py	

Undo the N latest commits, keep their changes staged

CAUTION

```
git reset --soft HEAD~<N>
```

Moves the active branch back `<N>` commits (`HEAD~1` undoes only the last one). Their changes stay staged.



Modified files	Staged files	Undone commits
	requirements.txt (new file)	requirements.txt

UNDO `git reset --soft ORIG_HEAD`



Scan me
to watch these
commands animate



Initial Commit
made with git-sim on initialcommit.com

The Essential Git Cheat Sheet

Page 3 of 3

Check the state of your repository

● SAFE

```
git status
```

Shows the active branch, staged changes, unstaged changes, and untracked files. During a merge, rebase, or cherry-pick, it also shows that operation's state and the next step.

Untracked files	Modified files	Staged files
notes.txt	app.py	README.md

Stage your file changes

● SAFE

```
git add <file>
```

Stages the current changes to `<file>`, so they're included in the next commit.

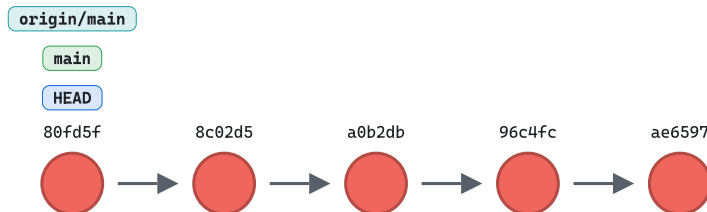
Untracked files	Modified files	Staged files
notes.txt	app.py	notes.txt app.py README.md

Push the active branch's commits

● SAFE

```
git push
```

Sends the active branch's new commits to its upstream branch on the remote.

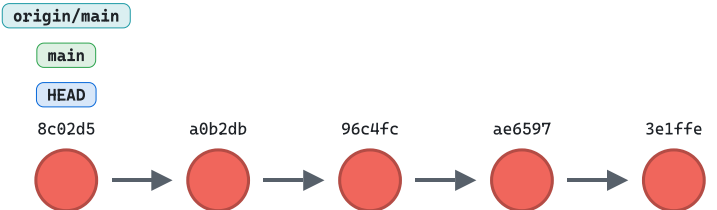


Clone a repository

● SAFE

```
git clone <url>
```

Downloads the repository at `<url>` with its full history into a new folder, checks out its default branch, and adds the source as the remote `origin`.



Create a new branch and switch to it

● SAFE

```
git switch -c <name>
```

Creates a branch named `<name>` at the current commit and switches to it.

