Git Ultimate Cheat Sheet

Distributed Version Control System Essentials

1 Setup & Config

Configuration

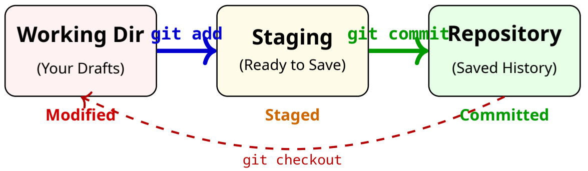
```
# Set global username
git config --global user.name "Your Name"

# Set global email
git config --global user.email "you@ex.com"

# Set default editor (e.g., VS Code)
git config --global core.editor "code --wait"

# List config
git config --list
```

Initialization

```
git init        # Start a new repo
git clone <url> # Copy remote repo
```

2 Basic Workflow

The Three Stages



Staging Changes

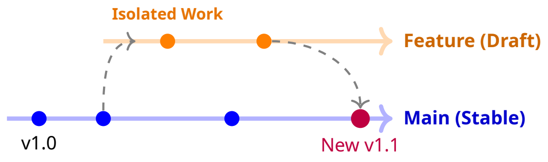
```
git status         # Check file status
git add <file>     # Stage specific file
git add .          # Stage all changes
```

Committing

```
# Commit with message
git commit -m "Brief description"

# Stage and commit (skip git add for tracked files)
git commit -am "Message"
```

3 Branching & Merging

Branches isolate development work.



Managing Branches

```
git branch              # List branches
git branch <name>       # Create new branch
git checkout <name>     # Switch to branch
git switch <name>       # Safer switch (modern)
git checkout -b <name>  # Create & switch
git branch -d <name>    # Delete branch
```

Merging Combines history of one branch into another.

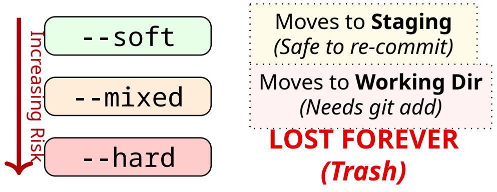
```
# Switch to target (e.g., main)
git checkout main
# Merge feature branch
git merge feature-branch
```

4 Undoing Changes

Discarding Changes (Safe)

```
# Unstage a file (keep changes)
git reset HEAD <file>

# Discard local changes (Dangerous!)
git checkout -- <file>
git restore <file>
```

Resetting (Time Travel)

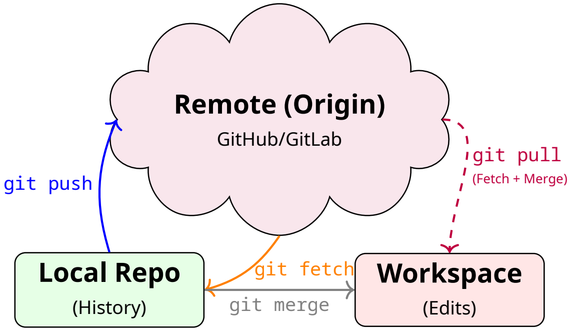


```
# Soft: Move HEAD back, keep changes staged
git reset --soft HEAD~1

# Mixed: Move HEAD, keep changes unstaged
git reset HEAD~1

# Hard: DESTROY changes (Dangerous!)
git reset --hard HEAD~1
```

Reverting (Public History) Creates a *new* commit that undoes changes. Safe for shared branches.

```
git revert <commit_hash>
```

5 Remote Interaction



Connecting

```
# View remotes
git remote -v

# Add origin
git remote add origin <url>
```

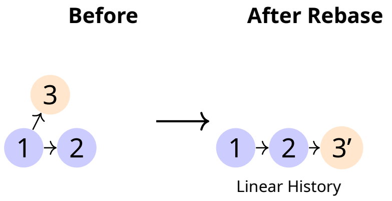
Syncing

```
# Download changes (no merge)
git fetch

# Download & Merge (Fetch + Merge)
git pull origin main

# Upload changes
git push origin main
```

6 Advanced Concepts

Merge vs. Rebase

- **Merge**: Preserves history/chronology. Creates ”merge commits”.
- **Rebase**: Rewrites history for a linear path. Cleaner, but dangerous on shared branches.



```
# Rebase current branch onto main
git checkout feature
git rebase main
```

Stashing Temporarily save changes to switch branches.

```
git stash           # Save changes
git stash pop       # Apply & drop
git stash list      # View stashes
```

Cherry Picking Pick a specific commit from another branch.

```
git cherry-pick <commit_hash>
```

7 Inspection

```
# Commit history
git log

# One line summary
git log --oneline

# Visual graph
git log --graph --oneline --all

# Show changes
git diff            # Unstaged changes
git diff --staged   # Staged changes
```

8 Ignoring Files

Create a `.gitignore` file to exclude files.

```
# Ignore logs
*.log

# Ignore node_modules
node_modules/

# Ignore secrets
.env
```