Git Worktrees + Claude Code
Cheatsheet

Run multiple Claude Code sessions in parallel on different branches — no stashing, no branch switching, no conflicts.

Common Commands

Command	Description
`git worktree list`	Show all active worktrees
`git worktree add <path> <branch>`	Create worktree from existing branch
`git worktree add <path> -b <branch>`	Create worktree with a new branch
`git worktree remove <path>`	Remove a worktree
`git worktree prune`	Clean up stale worktree references
`git branch -d <branch>`	Delete branch after worktree removed
`git fetch origin`	Fetch latest remote branches

Day to Day Workflow

```
# Add a new worktree
cd ~/projects/my-project-worktrees
git worktree add cool-feature -b feature/cool-feature

# Work in it
cd cool-feature
claude

# Push when ready (from inside the worktree)
git push origin feature/cool-feature

# Open a PR on GitHub/GitLab, get it reviewed, merge remotely

# Pull latest main
cd ../main
git pull

# Clean up finished worktree
cd ..
git worktree remove cool-feature
git branch -d feature/cool-feature
```

Merging

Worktrees don't merge into each other. Each pushes to its own remote branch. Merging happens through your normal PR workflow on GitHub/GitLab.

```
feature-auth/  →  push  →  origin/feature/auth  →  PR  →  merge into main
bugfix-nav/    →  push  →  origin/bugfix/nav    →  PR  →  merge into main
```

Standard Setup

```
# From inside your repo
git worktree add ../feature-auth feature/auth # existing branch
git worktree add ../bugfix-nav -b bugfix/nav # new branch
```

Umbrella Folder Pattern

```
mkdir ~/projects/my-project-worktrees
# Move or clone your repo inside it, then:
cd ~/projects/my-project-worktrees/my-project
git worktree add ../feature-auth feature/auth
git worktree add ../bugfix-nav -b bugfix/nav
```

Result:

```
my-project-worktrees/
my-project/ ← main repo (main branch)
feature-auth/ ← worktree
bugfix-nav/ ← worktree
```

Bare Repo Setup (Recommended)

No branch is "special" — every branch is an equal worktree. The bare repo is just the git database with no working files.

```
# 1. Create umbrella folder
mkdir ~/projects/my-project-worktrees
cd ~/projects/my-project-worktrees

# 2. Clone as bare repo
git clone --bare git@github.com:you/my-project.git .bare

# 3. Point .git to the bare repo
echo "gitdir: ./.bare" > .git

# 4. Fix remote fetch config (important!)
git config remote.origin.fetch "+refs/heads/*:refs/remotes/origin/*"
git fetch origin

# 5. Create worktrees
git worktree add main main
git worktree add feature-auth -b feature/auth
git worktree add bugfix-nav -b bugfix/nav
```

Result:

```
my-project-worktrees/
.bare/ ← git database (history, objects, refs)
.git ← tiny file pointing to .bare
main/ ← worktree
feature-auth/ ← worktree
bugfix-nav/ ← worktree
```

Running Claude Code

Open separate terminals and launch Claude Code in each worktree:

```
# Terminal 1
cd ~/projects/my-project-worktrees/feature-auth
claude

# Terminal 2
cd ~/projects/my-project-worktrees/bugfix-nav
claude

# Terminal 3 (headless mode)
cd ~/projects/my-project-worktrees/main
claude -p "refactor the auth module" --output-format stream-json
```

Managing Untracked Files (.env, etc.)

Worktrees share git history but each get their own working directory, so untracked files like `.env` won't carry over. Here are some approaches:

Symlink to a Shared File (Simplest)

Keep one `.env` outside the worktrees and symlink to it from each. Change it once, all worktrees pick it up.

```
# Create a shared env file in the umbrella folder
cp main/.env ../shared.env

# In each worktree
ln -s ~/projects/my-project-worktrees/shared.env .env
```

Copy Script

Write a small script that copies untracked files into a new worktree after creating it:

```
#!/bin/bash
# new-worktree.sh
git worktree add "$1" -b "$2"
cp ~/projects/my-project-worktrees/main/.env "$1/.env"
cp ~/projects/my-project-worktrees/main/.env.local "$1/.env.local"
```

.env.example + direnv

Check in a `.env.example` with placeholder values, then use **direnv** (direnv.net) which auto-loads `.envrc` files per directory. Each worktree can have its own `.envrc` that sources shared secrets.

Post-checkout Hook

Add a post-checkout hook in `.bare/hooks/` that auto-copies certain files whenever a worktree is created. Since all worktrees share the same `.git` (or `.bare`), one hook covers everything.

Tips

- **2–3 concurrent Claude sessions** is the sweet spot — you still need to review the output
- **CLAUDE.md** at the repo root is shared across all worktrees automatically
- **Name worktrees descriptively** so you know which terminal is which
- **Short-lived worktrees** — create for a task, merge the PR, remove
- A branch **cannot be checked out in multiple worktrees** at the same time