Introduction to Worktrees

Charles Daniels
May 14, 2026

About Me – Charles Daniels

- Computer Engineering B.S.E. – UofSC 2019
- Computer Science M.S. - UofSC 2021
- Backend Software Engineer – Synqly Inc.

All views and opinions expressed are my own, and do not represent those of Synqly.

Want the Slides?

- https://cdaniels.net/talks

Introduction

- Prerequisites
  - Basic git usage
  - Branches, PRs, commits
- Worktrees
  - Have multiple branches checked out at the same time
- Use cases
  - Work on multiple tasks simultaneously without stashing
  - Run a long running job in one branch while working on others
  - Work in parallel with AI agents

Background – Git Repo Structure

- `$GIT_DIR`
  - `.git` directory location
  - Bare repos are this directory only – no worktrees by default
- $GIT_WORK_TREE
  - Checked-out files live here (e.g. `README.md`, source files)
  - In a “normal” clone – the parent folder of `$GIT_DIR`
  - You can have more than 1

Git Repo Structure – Illustrated

```
$ pwd
/Users/cad/src/demo-orig
$ ls -lah
total 24
drwxr-xr-x   6 cad  staff   192B May 13 21:25 .
drwxr-xr-x   3 cad  staff    96B May 13 21:28 ..
drwxr-xr-x  12 cad  staff   384B May 13 21:25 .git
-rw-r--r--   1 cad  staff    35B May 13 21:24 go.mod
-rw-r--r--   1 cad  staff    80B May 13 21:25 main.go
-rw-r--r--   1 cad  staff    21B May 13 21:24 README.md
$ git rev-parse --git-dir
.git
$ git worktree list
/Users/cad/src/demo-orig  3815feb [main]
```

Add Another Worktree

- `git worktree add ../some-feature`
  - Creates branch some-feature, worktree in ../some-feature
- `git worktree add ../foldername branchname`
  - Check out an existing branch in a new worktree
- `git worktree add -b branchname ../foldername`
  - Create a new named branch in a folder with a different name

Add Another Worktree (demo)

```
$ git branch
* main
$ git branch new-feature
$ git worktree add ../new-tree new-feature
Preparing worktree (checking out 'new-feature')
HEAD is now at 3815feb first commit
$ git worktree list
/Users/cad/src/demo-orig  3815feb [main]
/Users/cad/src/new-tree   3815feb [new-feature]
$ cd ../new-tree
$ git rev-parse --git-dir
/Users/cad/src/demo-orig/.git/worktrees/new-tree
$ cat .git
gitdir: /Users/cad/src/demo-orig/.git/worktrees/new-tree
```

Delete a Worktree

- `git worktree remove ../some-tree`
  - Explicit removal.
- `rm -rf ../some-tree && git worktree prune`
  - Implicit, post-hoc removal of orphaned worktrees

Delete a Worktree (demo)

```
$ git worktree list
/Users/cad/src/demo-orig  3815feb [main]
/Users/cad/src/new-tree   3815feb [new-feature]
$ test -d ../new-tree ; echo $?
0
$ git worktree remove ../new-tree
$ git worktree list
/Users/cad/src/demo-orig  3815feb [main]
$ test -d ../new-tree ; echo $?
1
```

Day-to-Day Operations

- Most operations work identically when using worktrees, just `cd` to the tree you want
- commit, fetch, checkout, branch, rebase, cherry-pick, push – all work the same
- **But! Only 1 worktree per branch**
  - Checking out the same branch in more than one worktree will error
  - `fatal: 'branchname' is already used by worktree at '/path/to/worktree'`
- Caveat: .git =/= “top level directory”
  - Scripts should use rev-parse

Bare-Hub Pattern

- No need to have a “main” worktree
- Bare repos can have worktrees too!
- Worktree commands we already covered work fine inside of a bare repo’s `$GIT_DIR`

Bare Hub Pattern (demo 1/2)

```
$ pwd
/Users/cad/src/bare-hub-demo
$ git clone --bare ../demo-orig ./hub.git
Cloning into bare repository './hub.git'...
done.
$ cd hub.git
$ git config remote.origin.fetch '+refs/heads/*:refs/remotes/origin/*'
$ git fetch origin
From /Users/cad/src/bare-hub-demo/../demo-orig
 * [new branch]      main        -> origin/main
 * [new branch]      new-feature -> origin/new-feature
```

Bare Hub Pattern (demo 2/2)

```
$ git worktree add ../tree-new-feature new-feature
Preparing worktree (checking out 'new-feature')
HEAD is now at 3815feb first commit
$ git worktree list
/Users/cad/src/bare-hub-demo/hub.git              (bare)
/Users/cad/src/bare-hub-demo/tree-new-feature  3815feb [new-feature]
$ ls ../tree-new-feature
go.mod          main.go         README.md
$ cat ../tree-new-feature/.git
gitdir: /Users/cad/src/bare-hub-demo/hub.git/worktrees/tree-new-feature
```

Worktree Pool Pattern

- Managing a worktree per branch can be tedious
- You can create re-usable “slots” with placeholder branches, checkout as needed
- Usefulness depends on workflow and language ecosystem
  - Go’s shared build cache works well
  - node_modules may not

Worktree Pool Pattern (demo 1/2)

```
$ git worktree add ../sandbox-00
Preparing worktree (new branch 'sandbox-00')
HEAD is now at 3815feb first commit
$ cd ../sandbox-00
$ git worktree list
/Users/cad/src/bare-hub-demo/hub.git           (bare)
/Users/cad/src/bare-hub-demo/sandbox-00        3815feb [sandbox-00]
/Users/cad/src/bare-hub-demo/tree-new-feature  3815feb [new-feature]
```

Worktree Pool Pattern (demo 2/2)

```
$ git checkout -b task-1
Switched to a new branch 'task-1'
$ # do some work
$ git commit
$ git checkout sandbox-00
$ git checkout -b task-2
Switched to a new branch 'task-2'
$ # do more work
$ git commit
$ git checkout sandbox-00
Switched to branch 'sandbox-00'
$ git fetch
$ git rebase main
Current branch sandbox-00 is up to date.
```

Conclusion

- The checked-out files at a particular ref (a worktree) are separate from git’s internal files (gitdir)
- You can have 0 or more worktrees per gitdir – not just the 1 you get by default
- Useful for working on multiple tasks, collaborating with agents, letting slow builds run without trampling your active work

End.

- Questions?

- Want to connect?
  - contact@cdaniels.net
  - https://www.linkedin.com/in/charles-daniels-932767bb/