Git Worktrees Tutorial 🌳

Supercharge your development workflow with parallel Git branches

Press Space for next page `space`

The Problem 🤔

Traditional Git Workflow Pain Points

- **Context Switching Overhead**: Constantly stashing and switching branches
- **Lost Mental Context**: What were you working on before the urgent bug fix?
- **Merge Conflicts**: Stashing complex changes can cause issues
- **Development Interruption**: Can't work on multiple features simultaneously
- **AI Development Bottleneck**: Single branch limits parallel AI agent workflows

```
# The typical painful workflow
git add .                    # Working on feature-a
git stash                    # Urgent bug reported!
git checkout main            # Switch context
git pull                     # Update main
git checkout -b hotfix-123   # Create hotfix branch
# Fix bug, commit, merge ...
git checkout feature-a       # Back to feature work
git stash pop               # What was I working on again? 🤯
```

The Solution: Git Worktrees 🌟

Multiple Working Directories, One Repository

What are Git Worktrees?

- **Separate directories** for each branch
- **Shared repository** history and configuration
- **Independent working states** per directory
- **No context switching** required

Key Benefits

- 🚀 **Parallel Development**: Multiple features at once
- 🔥 **Instant Hotfixes**: No stashing or switching
- 🤖 **AI Agent Workflows**: Multiple AI assistants working simultaneously
- 👀 **Clean Code Reviews**: Dedicated review environment

```
# Git Worktree Workflow
cd ~/myproject                # Main directory: feature-a
git worktree add ../myproject-hotfix main
cd ../myproject-hotfix        # New directory: hotfix
# Fix bug here while feature-a stays intact!

# Directory structure:
~/workspace/
├── myproject/            # feature-a branch
├── myproject-hotfix/     # hotfix branch
├── myproject-review/     # main branch (clean)
└── myproject-exp/        # experimental branch
```

🎯 **Result:** Work on urgent fixes without losing your feature work context!

Real-World Scenarios 🎭

Scenario 1: The Urgent Bug Fix 🚨

- You're deep in a complex feature implementation
- **Critical production bug** is reported
- With worktrees: Keep feature work untouched
- Create hotfix in separate directory
- Deploy fix without context loss

```
# No interruption to your feature work!
git worktree add ../myproject-hotfix main
cd ../myproject-hotfix
# Fix bug, test, deploy
cd ../myproject  # Back to feature work
```

Scenario 2: AI-Assisted Parallel Development 🤖

- Multiple AI agents working simultaneously
- Each agent gets its own worktree
- **No conflicts** between AI workflows
- **Parallel progress** on different features

```
# Main: Claude Code on auth refactor
# Worktree 1: GPT-4 building dashboard
git worktree add ../project-dashboard feature-dash

# Worktree 2: Cursor on API optimization
git worktree add ../project-api feature-api

# Review: Clean main for code reviews
git worktree add ../project-review main
```

Tutorial Repository 📚

Comprehensive Learning Experience

🎯 5 Hands-on Exercises

- Basic worktree operations
- Parallel feature development
- Hotfix workflows
- AI agent coordination
- Cleanup and maintenance

📖 Rich Documentation

- Command cheatsheet
- Best practices guide
- AI development workflows

🛠️ Helper Scripts

- Setup and cleanup automation

Exercise 1: Basic Operations 🚀

Master the Fundamentals

Learning Objectives

- Create and list worktrees
- Navigate between directories
- Remove worktrees safely
- Understand directory structure

Key Commands

```
# Create new worktree
git worktree add <path>

# Create with specific branch
git worktree add <path> -b <branch>

# List all worktrees
git worktree list
```

Hands-on Practice

```
# Step 1: Check current state
git worktree list

# Step 2: Create first worktree
git worktree add ../tut-feat-a

# Step 3: Explore the new environment
cd ../tut-feat-a
git branch  # You're on tut-feat-a
ls -la      # Same files, different branch

# Step 4: Make independent changes
echo "# Feature A Notes" > feat-a-notes.md
git add feat-a-notes.md
git commit -m "Add feature A notes"

# Step 5: Verify independence
cd ../GitWorktreesTutorial
ls -la  # No feat-a-notes.md here!
```

Exercise 4: AI Agent Workflows 🤖

Parallel AI Development Made Easy

The Challenge

- Multiple AI coding assistants
- Different tasks requiring isolation
- Avoiding merge conflicts
- Coordinated development workflow

The Worktree Solution

- **Claude Code**: Authentication refactor (main directory)
- **GPT-4**: New dashboard (worktree 1)
- **Cursor**: API optimization (worktree 2)
- **Review Environment**: Clean main branch (worktree 3)

Implementation

```
# Main directory: Claude Code working
# on authentication refactor
git checkout -b auth-refactor

# Worktree 1: GPT-4 building dashboard
git worktree add ../project-dashboard -b feature-dashboa

# Worktree 2: Cursor optimizing API
git worktree add ../project-api -b optimize-api

# Worktree 3: Clean environment for reviews
git worktree add ../project-review main

# Each AI agent works independently!
# No conflicts, no context switching
```

🚀 **Pro Tip:** Use different terminals/IDEs for each worktree to maximize AI agent efficiency!

Best Practices & Tips 💡

📁 Directory Organization

```
~/workspace/
├── myproject/            # main
├── myproject-feat-auth/  # features
├── myproject-feat-dash/  # features
├── myproject-hotfix/     # hotfixes
└── myproject-review/     # reviews
```

Consistent naming helps identify purpose

⚡ Git Aliases

```
# Add to ~/.gitconfig
[alias]
    wt = worktree
    wtlist = worktree list
    wtadd = worktree add
    wtremove = worktree remove
    wtprune = worktree prune
```

Speed up your worktree commands

🔧 VSCode Integration

- Each worktree = separate folder
- Multiple VS Code windows
- Independent extensions/settings
- Parallel development environments

Perfect for multi-tasking developers

Sample Project: Real-World Practice 🌐

Web Application with Intentional Learning Opportunities

Project Structure

```
sample-project/
├── src/
│   ├── frontend/     # React app
│   │   ├── App.js
│   │   └── App.css
│   └── backend/      # Node.js server
│       └── server.js
├── tests/
│   ├── unit/         # Unit tests
│   └── integration/  # Integration tests
├── config/           # Configuration
├── docs/            # Documentation
└── package.json
```

Practice Scenarios

- 🐛 **Bug Fixes**: Critical issues to resolve
- ✨ **New Features**: Authentication, dashboard
- 🔧 **Refactoring**: Code improvements
- 📊 **Performance**: Optimization tasks
- 🧪 **Testing**: Add test coverage

Learning Outcomes

- Practice worktree workflows on real code
- Experience parallel development benefits
- Handle realistic merge scenarios
- Build confidence with complex projects

Advanced Workflows 🔥

Code Review Workflow

The Setup

```
# Dedicated review environment
git worktree add ../project-review main
cd ../project-review

# Always clean main branch for reviews
# Continue feature work in main directory
```

Benefits

- 🎯 **Clean Environment**: No work-in-progress files
- 🔍 **Focus**: Dedicated space for reviews
- ⚡ **Fast Switching**: No stashing required
- 📊 **Context**: Keep feature work visible

Experimentation Workflow

The Setup

```
# Try risky experiments safely
git worktree add ../project-experiment -b experiment-new

# Compare approaches
git worktree add ../project-comparison -b comparison-app
```

Benefits

- 🧪 **Safe Testing**: Original work protected
- 📈 **A/B Comparison**: Side-by-side evaluation
- 🔄 **Easy Rollback**: Just remove the worktree
- 💡 **Innovation**: Try bold ideas without fear

Ready to Transform Your Workflow? 🚀

Get Started with the Git Worktrees Tutorial

📚 GitHub Repository

github.com/kriscoleman/GitWorktreesTutorial

🎯

5 Hands-on Exercises

From basics to advanced workflows

🤖

AI Development Ready

Perfect for modern AI-assisted coding

🛠️

Production Ready

Real-world scenarios and best practices

Start with Exercise 1: Basic Worktree Operations

Master the fundamentals in 15 minutes

Thank You! 🙏

Questions & Discussion

Resources

- 📖 **Tutorial**: GitHub Repository
- 📚 **Documentation**: Command cheatsheet, best practices, AI workflows
- 🛠️ **Scripts**: Setup, cleanup, and validation helpers
- 💬 **Support**: GitHub Issues and Discussions

Ready to supercharge your development workflow with Git worktrees?