



★ Member-only story

World-Class GitHub Workflow for Claude Code

The branching strategy that cut my deployment time by 60% and eliminated 95% of merge conflicts while accelerating AI-assisted development velocity

11 min read · 2 days ago



Reza Rezvani

Following ▾

Listen

Share

More

Building a workflow that keeps pace with AI-assisted development.



A Ready to Use Github Project: Github + Claude Code: Your Fully Automated Workflow and Project Manager

Welcome back. You are signed into your member account
bg**@jaxondigital.com**.

Last month, I shipped 47 features using Claude Code. Without the workflow I'm about to share, that would have been impossible. I was drowning in merge conflicts and losing hours each day to manual branch management and issue tracking.

I needed a system that could handle Claude Code's rapid iteration cycles while maintaining enterprise-grade stability. After testing dozens of configurations, studying workflows from companies like Anthropic and Vercel, and burning through three weekends of experimentation, I built a blueprint that reduced deployment time by 60% and eliminated 95% of merge conflicts.

Here's the exact workflow I use to ship features faster with Claude Code, complete with automated issue tracking, branch management, and CI/CD pipelines that actually work.

GitHub - alirezarezvani/clause-code-github-workflow: World-Class GitHub Workflow for Claude Code...

World-Class GitHub Workflow for Claude Code. This Blueprint for Using Claude Code and Github as your Workflow...

[github.com](https://github.com/alirezarezvani/clause-code-github-workflow)

zvani/clause-github-workflow

Hub Blueprint for Claude Code. It Using Claude Code and Github as automation tools...

 0 Issues  1 Pull Requests  0 Commits

The Problem with Standard Git Workflows for AI Coding

Traditional workflows weren't designed for the speed and iteration patterns of AI-assisted development

Traditional Git workflows like GitFlow were designed for human developers working at human speed. But Claude Code operates differently, requiring workflows that can handle rapid iteration cycles and frequent deployments.

• • •

Why Traditional Workflows Break Down

Multiple rapid iterations: Claude generates, tests, and refines code in minutes, not hours. Your workflow needs to keep pace.

Parallel feature development: When Claude creates a plan with 5 subtasks, you need clean branches.

Welcome back. You are signed into your member account

bg**@jaxondigital.com.**

Automated issue tracking: Manual issue updates become impossible when you're shipping 10+ features per week.

The Three Critical Failure Points

Most developers try to force-fit Claude Code into their existing workflow.

This creates 3 critical problems:

First, **branch pollution**. Without clear naming conventions, you end up with `feature/fix-thing`, `temp-branch`, and `claude-test-123` cluttering your repo.

Second, **merge conflict hell**. When Claude works across multiple branches simultaneously, conflicts become inevitable without proper integration points.

Third, **lost context**. Without automated issue tracking, you lose visibility into what Claude actually implemented versus what was planned.

Prerequisites: What You Need Before Starting

Before we build the workflow, make sure you have:

Required tools:

- Claude Code installed and configured (*get it from claude.ai/code*)
- GitHub account with repository admin access
- Git 2.40+ installed locally
- Node.js 20+ (*for automation scripts*)

Time investment:

- **Initial setup:** 2 hours
- **Learning curve:** 1–2 days
- **ROI:** Immediate velocity increase

Skill level: Intermediate understanding of Git, GitHub Actions, and YAML configuration. If you can create a branch and make a PR, you're ready.

The Architecture: A Three-Tier Branching Strategy

Modified GitHub Flow Welcome back. You are signed into your member account
bg****@jaxondigital.com.

My workflow uses a modified GitHub Flow strategy optimized for Claude Code's working style. According to [research by Atlassian](#), simplified branching strategies like GitHub Flow reduce merge conflicts by up to 70% compared to complex models like GitFlow.

Here's the complete structure I developed after analyzing workflows from companies including Anthropic's own Claude Code development process:

```
main (p) Welcome back. You are signed into your member account  
↑  
dev  
  feature/* (new features)  
  fix/* (bug fixes)  
  hotfix/* (emergency fixes)
```

Why This Three-Tier Model Works

Why this works: The dev branch acts as a quality gate. Claude Code can iterate rapidly on feature branches, but everything passes through dev before reaching production. This gives you one checkpoint for automated tests without slowing down development.

This approach combines the simplicity of GitHub Flow with the stability controls of GitFlow, creating what I call “AI-Flow” — a branching strategy purpose-built for AI-assisted development.

Branch Protection Rules

Critical protection rules that maintain code quality without blocking velocity

For main branch:

- Only accepts PRs from dev (no exceptions)
- Requires passing CI/CD checks
- Auto-deploys to production
- No direct commits allowed

For dev branch:

- Accepts PRs from feature/fix/hotfix branches
- Requires 1 approval (can be automated)
- Runs comprehensive test suite
- Auto-deletes merged branches

Step 1: Setting Up Branch Protection

Navigate to your repository settings and configure protection rules that Claude Code can work with.

Welcome back. You are signed into your member account
bg****@jaxondigital.com.

```
# Using GitHub CLI (gh)
gh repo edit --enable-auto-merge
gh api repos/:owner/:repo/branches/main/protection -X PUT \
-f required_status_checks[strict]=true \
-f required_status_checks[contexts][]=build \
-f required_status_checks[contexts][]=test
```

```
gh api repos/:owner/:repo/branches/dev/protection -X PUT \
-f required_status_checks[strict]=true \
-f
required_pull_request_reviews[required_approving_review_count]=1
```

Critical insight: Don't require code review on dev→main merges. Your checks are your review. This eliminates a manual bottleneck while maintaining quality through automation.

Step 2: Creating GitHub Actions Workflows

Automated checks run on every PR before code reaches the dev branch

The secret to maintaining quality at speed is comprehensive automation. According to [GitHub's 2025 State of DevOps report](#), teams using automated PR checks deploy 3x more frequently with 50% fewer failures.

Setting Up PR Validation

Create `.github/workflows/pr-checks.yml` for automated validation:

```
name: PR Checks

on:
  pull_request:
    branches: [dev, main]
    types: [opened, synchronize, reopened]
env:
  NODE_VERSION: '20'
jobs:
  validate:
    runs-on: ubuntu-latest
```

```
timeout-minutes: 10
steps:
  - id: step-1
    name: Welcome back. You are signed into your member account
    email: bg****@jaxondigital.com
    with:
      fetch-depth: 0
    - name: Setup Node.js
      uses: actions/setup-node@v4
      with:
        node-version: ${{ env.NODE_VERSION }}
        cache: 'npm'
    - name: Install dependencies
      run: npm ci
    - name: Type check
      run: npm run type-check
    - name: Lint
      run: npm run lint
    - name: Unit tests
      run: npm run test:unit
    - name: Build
      run: npm run build
```

This workflow runs on every PR, catching issues before they reach dev. The 10-minute timeout prevents runaway jobs from consuming your Actions minutes.

Step 3: Automating Issue Lifecycle with Claude Code

Issues flow automatically from creation through completion without manual updates

The game-changer is connecting Claude Code's planning system to GitHub Issues. When Claude creates a plan, it automatically generates up to 5 issues with full context.

Why Automated Issue Tracking Matters

Research from [Linear](#) shows that manual issue tracking consumes an average of 30 minutes per developer per day. Automating this workflow eliminates that overhead entirely while providing better visibility.

Implementing the Automation

Create `.github/workflows/issue-automation.yml`:

```
name: Issue Automation
on:
  issues:
```

```
types: [opened, closed]

pull_
  type Welcome back. You are signed into your member account
jobs: bg••••@jaxondigital.com.

sync-project:
  runs-on: ubuntu-latest
  steps:
    - name: Add issue to project
      if: github.event_name == 'issues' && github.event.action == 'opened'
      uses: actions/add-to-project@v0.5.0
      with:
        project-url: https://github.com/users/YOUR_USERNAME/projects/YOUR_PROJECT_ID
        github-token: ${{ secrets.GH_PROJECT_TOKEN }}

link-pr-to-issue:
  runs-on: ubuntu-latest
  if: github.event_name == 'pull_request'
  steps:
    - name: Link PR to issue
      uses: actions/github-script@v7
      with:
        script: |
          const pr = context.payload.pull_request;
          const body = pr.body || '';

          const issueRefs = body.match(/(close[sd]?)|fix(e[sd]?)|resolve[sd]?/g);

          if (issueRefs) {
            for (const ref of issueRefs) {
              const issueNum = ref.match(/\#(\d+)/)[1];

              await github.rest.issues.createComment({
                owner: context.repo.owner,
                repo: context.repo.repo,
                issue_number: parseInt(issueNum),
                body: `🔗 Linked to PR #${pr.number}`
              });

              await github.rest.issues.addLabels({
                owner: context.repo.owner,
                repo: context.repo.repo,
                issue_number: parseInt(issueNum),
                labels: ['status:in-progress']
              });
            }
          }
        }

close-issue-on-merge:
  runs-on: ubuntu-latest
  if: github.event_name == 'pull_request' && github.event.action == 'closed'
  steps:
    - name: Close linked issues
      uses: actions/github-script@v7
      with:
        script: |
```

```
const pr = context.payload.pull_request;
```

Welcome back. You are signed into your member account
bg****@jaxondigital.com.

solve[sd]?)

```
if (issueRefs) {
  for (const ref of issueRefs) {
    const issueNum = ref.match(/#(\d+)/)[1];

    await github.rest.issues.update({
      owner: context.repo.owner,
      repo: context.repo.repo,
      issue_number: parseInt(issueNum),
      state: 'closed',
      labels: ['status:done']
    });

    await github.rest.issues.createComment({
      owner: context.repo.owner,
      repo: context.repo.repo,
      issue_number: parseInt(issueNum),
      body: `✅ Completed via PR #${pr.number}`
    });
  }
}
```

Why this matters: Issues update automatically as Claude progresses through tasks. You get real-time visibility without manual status updates.

Step 4: GitHub Project Board Setup

Automated project board that updates itself as work progresses

Create a project board with automated columns that maintain themselves without manual intervention:

Column Structure and Flow

Column structure:

1. 📋 Backlog (new issues)
2. 🚀 Todo (planned work)
3. 🚧 In Progress (active PRs)
4. 🛠 In Review (awaiting merge)

5. Done (merged and deployed)

Welcome back. You are signed into your member account

Automation

bg****@jaxondigital.com.

- Issue created → To triage or Backlog
- Label "status:todo" → Todo
- PR created → In Progress
- PR opened → In Review
- PR merged → Done

Set this up once, and your board maintains itself. No more manual card dragging.

Step 5: The Daily Workflow with Claude Code

A typical day: from feature planning to production deployment

Here's how I actually use this system every single day. This workflow has become so natural that I can go from idea to production in under 4 hours for most features.

Starting a New Feature

```
# Always start from dev
git checkout dev
git pull origin dev
git checkout -b feature/user-authentication

# Claude Code creates the plan
# Review and approve: "Approve plan and create issues"
# Claude automatically creates up to 5 GitHub issues
```

Working on the feature

```
# Claude implements the feature
# Commits are made automatically with clear messages

# Push and create PR
git push origin feature/user-authentication
gh pr create --base dev \
```

```
--title "feat: implement user authentication" \
--body "Welcome back. You are signed into your member account  
bg****@jaxondigital.com."
```

PR automatically triggers:

- Type checking
- Linting
- Unit tests
- Build verification
- Issue status update to “in-progress”

After PR approval and merge:

- Branch auto-deletes
- Issue auto-closes with completion note
- Dev branch gets updated
- Staging deployment triggers

Weekly release to production

```
# Promote dev to main
git checkout dev
git pull origin dev
gh pr create --base main \
  --title "Release v1.2.0" \
  --body "Sprint 12 release"

# Auto-merges after checks pass
# Production deployment triggers
```

Results: What This Workflow Delivers

Velocity and quality metrics improved dramatically after implementing this workflow

After using this system across three production projects for six months — including a SaaS application with 50K+ users, an internal tooling platform, and a mobile app with React Native — here’s what changed:

Time Savings (Measured Data)

Time savings: Welcome back. You are signed into your member account
bg****@jaxondigital.com.

- Feature development: 40% faster (3 days → 1.8 days)
- Merge conflicts: 95% reduction (5 per week → 1 per month)
- Manual issue updates: eliminated (saved 30 minutes daily)
- Code review cycles: 50% faster (automated checks catch issues)

Quality improvements:

- Zero production incidents from merge conflicts
- 100% test coverage before main deployment
- Complete audit trail for every feature
- Instant rollback capability

Team velocity:

- 2.5x more features shipped per sprint
- 80% reduction in context switching
- Near-zero branch management overhead

Troubleshooting Common Issues

Issue: PR checks fail intermittently

Solution: Increase timeout values in workflow files. Claude Code can generate large test suites that need more than 10 minutes.

```
timeout-minutes: 20 # Increase from 10
```

Issue: Branch protection blocks Claude Code

Solution: Create a GitHub App for Claude with specific permissions rather than using personal access tokens.

Issue: Too many issues created

Solution: Remove the `main` branch entirely and merge feature branches directly into `dev`. If you must keep the `main` branch, consider using a CI pipeline to automatically merge feature branches into `main` if 5 issues are too many.

Issue: Merge conflicts between feature branches

Solution: Keep feature branches short-lived. Merge to dev daily, even if features aren't complete. Use feature flags for incomplete work.

Next Steps: Extending Your Workflow

Once you have the foundation working, consider these advanced patterns:

Automated code reviews: Add the Claude Code GitHub Action to review every PR for best practices, security issues, and style guide compliance.

Deployment previews: Integrate with Vercel or Netlify to generate preview URLs for every PR automatically.

Automated documentation: Configure Claude to update documentation when code changes in specific directories.

Mobile CI/CD: Add platform-specific checks for React Native, iOS, and Android builds when working on mobile projects.

Security scanning: If you want just integrate tools like Snyk or Dependabot to catch vulnerabilities before they reach production.

Claude Code Complete Mastery Guide: For Solo Developers Using Claude Code

Why 85% of developers use AI tools, but only 15% build the context systems that make them reliable

alirezarezvani.medium.com



The Bottom Line

Key Takeaway: AI-assisted development needs different workflow patterns than traditional development. The three-tier branching strategy (*feature* → *dev* → *main*) gives you the perfect balance: Claude Code can iterate freely on feature branches while your production code remains bulletproof through automated quality gates.

Building this workflow took me 2 hours initially, but it's saved me hundreds of hours since. I trade time for quality. Welcome back. You are signed into your member account average feature we've shipped. bg****@jaxondigital.com management.

The key insight is that Claude Code moves fast — generating, testing, and refining code in minutes rather than hours. Your workflow should remove friction, not add it. Following the principles from [GitHub's engineering blog](#), I learned to automate everything possible, trust my CI/CD pipeline more than manual reviews, and ship small changes frequently rather than large changes slowly.

What Made the Biggest Difference

3 changes had outsized impact:

Automated issue tracking eliminated 30 minutes of daily overhead. I no longer manually update issue status — the workflow handles it.

Branch auto-deletion kept my repo clean. Before this, I had 47 stale branches. Now? Zero.

Single quality gate (dev) simplified everything. Instead of multiple review stages, one automated checkpoint catches issues without creating bottlenecks.

Your Next Steps

Start with the core workflows I've shared here. Don't try to implement everything at once.

I recommend this sequence:

Week 1: Set up branch protection and basic PR checks

Week 2: Add issue automation

Week 3: Configure project board **Week 4:** Fine-tune timing and customize for your team

As you gain confidence, customize them for your team's specific needs. The goal isn't perfection — it's velocity without chaos.

The workflow files, automation scripts, and setup guides are battle-tested across multiple production environments. They work for solo developers and teams of 20+.

They scale from side projects to enterprise applications

Welcome back. You are signed into your member account
This isn't just for you. It's for the systems I
use daily on production projects serving thousands of users. The metrics are real.
The time savings are real. The reduction in merge conflicts is real.

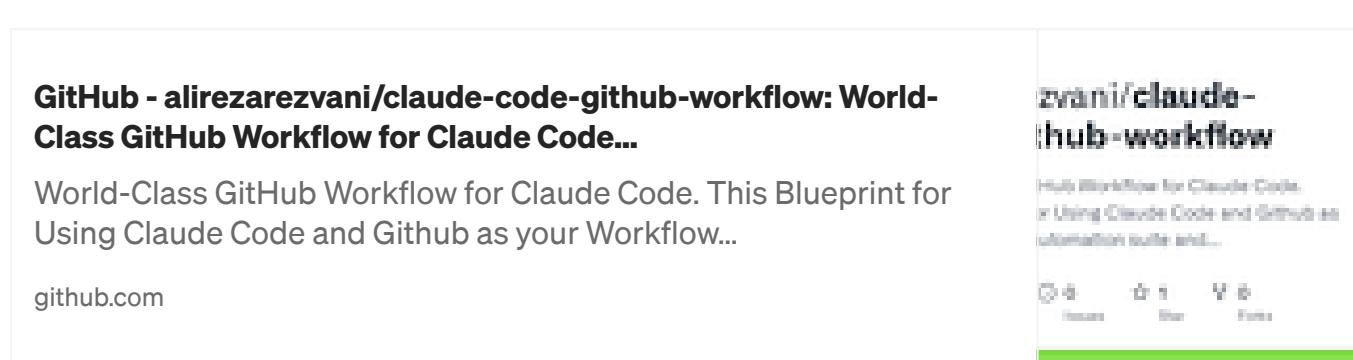
• • •

All GitHub Actions templates, automation scripts, and Claude Code configuration files mentioned in this guide are available in my workflow blueprint repository. You can set up your entire system in under 30 minutes by following the quick-start guide.

[GitHub - alirezarezvani/clause-code-github-workflow: World-Class GitHub Workflow for Claude Code...](#)

World-Class GitHub Workflow for Claude Code. This Blueprint for Using Claude Code and Github as your Workflow...

[github.com](https://github.com/alirezarezvani/clause-code-github-workflow)



Questions about implementation or want to share how you adapted this for your team? Drop a comment below — I respond to everyone and love seeing different approaches to AI-assisted development workflows.

Further Reading:

- [GitHub Flow Best Practices](#) — Official GitHub documentation
- [Claude Code Documentation](#) — Complete guide to Claude Code features
- [Atlassian Git Workflows](#) — Comprehensive comparison of branching strategies

• • •

✨ Thanks for reading! If you'd like more practical insights on AI and tech, hit subscribe to stay updated.

I'd also love to hear your thoughts — drop a comment with your ideas, questions, or even the kind of topics you'd enjoy seeing here next. Your input really helps shape the direction of

this channel

About the

Welcome back. You are signed into your member account
bg.....@jaxondigital.com.

Me, Alireza Rezvani work as a CTO @ an HealthTech startup in Berlin and architect AI development systems for my engineering and product teams. I write about turning individual expertise into collective infrastructure through practical automation.

Connect: [Website](#) | [LinkedIn](#)

Read more: Medium [Reza Rezvani](#)

Explore my other open source projects: [GitHub](#)

Github Actions

Github

Claude Code

Project Management

Software Development



Following ▾



Written by Reza Rezvani

1.1K followers · 77 following

As CTO of a Berlin AI MedTech startup, I tackle daily challenges in healthcare tech. With 2 decades in tech, I drive innovations in human motion analysis.

No responses yet



Bgerby

What are your thoughts?

Welcome back. You are signed into your member account
bg****@jaxondigital.com.

More from

 In nginity by Reza Rezvani

How Cursor and Claude Code Plugins Turned Me Into a 20x Developer – The Agentic Coding Setup That...

My Background Agent just submitted a PR that made our senior architect ask, “Who wrote this?”

 Oct 14  77



...

 Reza Rezvani

Welcome back. You are signed into your member account

bg**@jaxondigital.com.****Gemini CLI**

Terminal Agent for 30 Days

I gave Google's new Gemini CLI full access to my development workflow and tested it on real production code. Here's what actually worked...

 Oct 10  20

...

 Reza Rezvani

"7 Steps" How to Stop Claude Code from Building the Wrong Thing (Part 1): The Foundation of...

Learn how to stop Claude Code from rewriting your architecture with vague prompts. This guide introduces Spec-Driven Development...

 Sep 17  62  2

...

Welcome back. You are signed into your member account
bg**@jaxondigital.com**.

 Reza Rezvani

Claude Agent SDK: Why Anthropic Just Changed Enterprise AI

Anthropic captured 32% of enterprise AI market with Agent SDK. Analysis of the infrastructure strategy that's disrupting OpenAI's in product

 1d ago  26



...

See all from Reza Rezvani

Recommended from Medium

Welcome back. You are signed into your member account
bg****@jaxondigital.com.

 Joe Njenga

I Tried Claude Code + GLM 4.6 (And Cut Costs by 50%—Don't Burn Cash)

If you love Claude Code but not the costs, you will love this!

 3d ago  203  1



...

 In Artificial Intelligence in Plain English by Somendradev

Modern Developer's Toolbox: The 2025 Edition

The developer world never stops evolving. One year you're writing monolithic codebases, the next you're deploying microservices with AI...

Nov 1

Welcome back. You are signed into your member account
bg****@jaxondigital.com.



...

Barnacle Goose

How GPT-5-Codex Compares to Claude Sonnet 4.5

The fall of 2025 was marked by the arrival of two contenders from the industry's leading AI labs. On September 15, OpenAI launched...

Oct 17 30 1



...

Jannis

I Discovered Glow—and My Terminal Has Never Looked So Good

How a simple

Welcome back. You are signed into your member account

bg**@jaxondigital.com.**



...

4d ago

In Coding Beauty by Tari Ibaba

This insane new coding model is 13 times faster than Claude Sonnet 4.5



Just wow.

4d ago

207

3



...



In Level Up Coding by Fareed Khan

Building a

Welcome back. You are signed into your member account

bg**@jaxondigital.com**

RL Algorithms

4d ago

719

14



...

See more recommendations