

Claude Code in your CI with GitHub Actions

Three workflows so Claude Code reviews every pull request, explains why tests fail and writes documentation when someone types @claude, with minimum permissions, time limits and a script to test the explanation step locally before pushing anything.

FOR	github.com repositories with admin access
TESTED WITH	Claude Code 2.1.238 locally · workflows validated, not run on GitHub
UPDATED	26 October 2026
DOCUMENTATION	code.claude.com/docs/en/github-actions · /headless
ARTICLE	ecomlabs.dev/blog/claude-code-github-actions
AUTHOR	Luis Morales · @luismoralesdev

Contents

Before
you start

What you will set up

Step 01

Test the explanation step locally

Step 02

Install the app and the secret

Step 03

Review every pull request

Step 04

An explanation when tests fail

Step 05

@claude for documentation

Step 06

Check that it works

Going
further

Costs and variations

If
something
fails

Troubleshooting

When you
finish

Checklist

What you will set up

The official `anthropics/claude-code-action@v1` action runs Claude Code inside a GitHub Actions job. This guide sets up three uses for it:

WORKFLOW	WHEN IT RUNS	WHAT IT DOES
<code>claude-review.yml</code>	When a PR is opened or updated	Reviews the PR and comments on its lines
<code>ci.yml</code>	On every push and PR, only if tests fail	Explains the cause in the run log
<code>claude.yml</code>	When someone types <code>@claude</code>	Answers and can push commits, documentation for example

CAUTION

The workflows follow the examples in the official docs and their YAML is validated, but we did not run them on GitHub for this guide. What is tested is the explanation step, locally, with the scripts from step 1.

Requirements

- A github.com repository where you have admin access.
- A Claude Console API key, or a Claude subscription to use `claude setup-token`.
- Claude Code on your machine, to test locally and, if you want, for the quick installer.
- For the quick installer, the GitHub CLI (`gh`) logged in.

Test the explanation step locally

Before spending CI minutes, check that the prompt gives a useful explanation with your tests. These scripts do the same as the "Explain the failure" step in `ci.yml`, but with `claude -p` in your terminal:

test/explain-failure.sh

```
#!/usr/bin/env bash
# explain-failure.sh: runs the tests and, if they fail, asks Claude Code to
# explain the cause, just like the "Explain the failure" step in ci.yml.
# Usage, from the project root:  bash test/explain-failure.sh [claude options]
#   for example:  bash test/explain-failure.sh --model haiku
# Needs Claude Code with a logged-in session, Node and npm.

output=$(npm test 2>&1)
code=$?
if [ "$code" -eq 0 ]; then
  echo "The tests pass: nothing to explain."
```

```

    exit 0
fi

prompt="The tests in this project failed; their output arrives on standard input. \
Explain in under 15 lines: which test fails, the most likely cause and which file to \
look at. Do not change any file."

json=$(printf '%s\n' "$output" | tail -n 200 | claude -p "$prompt" --max-turns 3 --output-format json "$@" ) ||
{ echo "explain-failure: claude -p did not finish cleanly" >&2; exit 1; }

# Three fields of the claude -p JSON matter here: result, num_turns and total_cost_usd.
printf '%s' "$json" | node -e '
    let s = "";
    process.stdin.on("data", (d) => (s += d)).on("end", () => {
        const r = JSON.parse(s);
        console.log(r.result);
        console.log(`\n${r.num_turns} turns, ${r.total_cost_usd.toFixed(4)} USD`);
    });'
exit "$code"

```

test/explain-failure.ps1

```

# explain-failure.ps1: runs the tests and, if they fail, asks Claude Code to
# explain the cause, just like the "Explain the failure" step in ci.yml.
# Usage, from the project root:
# powershell -NoProfile -ExecutionPolicy Bypass -File test\explain-failure.ps1 [claude options]
# for example: ... -File test\explain-failure.ps1 --model haiku
# Needs Claude Code with a logged-in session, Node and npm.

# PowerShell 5.1 sends text to a program as ASCII and reads it in the console
# code page; UTF-8 both ways keeps accented characters intact.
$OutputEncoding = [System.Text.UTF8Encoding]::new($false)
[Console]::OutputEncoding = [System.Text.UTF8Encoding]::new($false)

# With 2>&1, PowerShell wraps every stderr line in an ErrorRecord;
# TargetObject keeps the original text.
$output = & npm test 2>&1 | ForEach-Object {
    if ($_ -is [System.Management.Automation.ErrorRecord]) { "$($_.TargetObject)" } else { "$_" }
}
$code = $LASTEXITCODE
if ($code -eq 0) {
    'The tests pass: nothing to explain.'
    exit 0
}

$prompt = 'The tests in this project failed; their output arrives on standard input. ' +
    'Explain in under 15 lines: which test fails, the most likely cause and which file to ' +
    'look at. Do not change any file.'

$json = $output | Select-Object -Last 200 | & claude -p $prompt --max-turns 3 --output-format json @args
if ($LASTEXITCODE -ne 0) {

```

```
[Console]::Error.WriteLine('explain-failure: claude -p did not finish cleanly')
exit 1
}

# Three fields of the claude -p JSON matter here: result, num_turns and total_cost_usd.
$r = ($json -join "`n") | ConvertFrom-Json
$r.result
''

'({0} turns, {1:N4} USD)' -f $r.num_turns, $r.total_cost_usd
exit $code
```

From the project root:

```
bash test/explain-failure.sh --model haiku
powershell -NoProfile -ExecutionPolicy Bypass -File test\explain-failure.ps1 --model haiku
```

If the tests pass, the script says so and exits with 0. To see an explanation, break something on purpose. We raised the demo store's shipping from 490 to 590 cents: the explanation found the two failing tests, the 100-cent difference and the file to look at. Each run with Haiku cost between USD 0.048 and 0.059.

The PowerShell version sets UTF-8 for input and output: without it, PowerShell 5.1 hands `claude` any accented characters in the test output as question marks.

STEP 02

Install the app and the secret

Quick path

Open `claude` in the repository and run `/install-github-app`. It installs the Claude app, saves the secret and opens a PR with the workflow you choose. It only works with github.com repositories.

Manual path

1. Install the app from github.com/apps/claude on the repository.
2. Under *Settings* → *Secrets and variables* → *Actions*, create one of these secrets:
 - `ANTHROPIC_API_KEY`: a key from platform.claude.com. Billed by usage.
 - `CLAUDE_CODE_OAUTH_TOKEN`: run `claude setup-token` on your machine and paste the result. Uses your subscription.
3. Copy the workflows into `.github/workflows/`.

The ZIP's workflows use `ANTHROPIC_API_KEY`. If you chose the token, change the `anthropic_api_key` line in each one to:

```
claude_code_oauth_token: ${ secrets.CLAUDE_CODE_OAUTH_TOKEN }
```

For an organization, the docs recommend an API key as an organization secret: the token is tied to the subscription of whoever generated it.

Review every pull request

.github/workflows/claude-review.yml

```
# Review every pull request with Claude Code's code-review plugin.
# Based on the "Run a skill" example at code.claude.com/docs/en/github-actions.
# Claude comments on the PR's lines, or leaves one comment if it finds nothing.
name: Claude review

on:
  pull_request:
    types: [opened, synchronize, ready_for_review, reopened]

# A new push to the same PR cancels the previous review: you do not pay twice.
concurrency:
  group: claude-review-${{ github.event.pull_request.number }}
  cancel-in-progress: true

jobs:
  review:
    runs-on: ubuntu-latest
    timeout-minutes: 15
    permissions:
      contents: read
      pull-requests: read
      issues: read
      id-token: write
    steps:
      - uses: actions/checkout@v6
        with:
          fetch-depth: 1
      - uses: anthropics/claude-code-action@v1
        with:
          anthropic_api_key: ${ secrets.ANTHROPIC_API_KEY }
          plugin_marketplaces: "https://github.com/anthropics/claude-code.git"
          plugins: "code-review@claude-code-plugins"
          prompt: "/code-review:code-review --comment ${ github.repository }}/pull/${{
github.event.pull_request.number }}"
          claude_args: '--allowedTools "mcp_github_inline_comment__create_inline_comment"'
```

- `--comment` makes the review show up on the PR: one comment per finding on its line, or a single one if it finds nothing. Without it, it only stays in the log.
- `--allowedTools "mcp_github_inline_comment__create_inline_comment"` is needed even though the plugin already names that tool: the action only starts the comment server if it appears here.
- `concurrency` cancels the previous review when a new push lands on the same PR.

Per the docs, Claude skips draft and closed PRs and those that already have a comment from it. In public repositories it does not run on PRs from forks, because GitHub withholds secrets from them.

An explanation when tests fail

.github/workflows/ci.yml

```
# CI with an explanation: if the tests fail, Claude reads their output and
# explains the cause in the run log. It does not change any file.
name: CI

on:
  pull_request:
  push:
    branches: [main]

concurrency:
  group: ci-${{ github.ref }}
  cancel-in-progress: true

jobs:
  test:
    runs-on: ubuntu-latest
    timeout-minutes: 15
    permissions:
      contents: read
      id-token: write
    steps:
      - uses: actions/checkout@v6
      - uses: actions/setup-node@v7
        with:
          node-version: 22
      - run: npm ci
      - name: Tests
        id: tests
        # shell: bash turns on pipefail: if npm test fails, the step fails even though tee does not.
        shell: bash
        run: npm test 2>&1 | tee test-output.txt
      - name: Explain the failure
        if: failure() && steps.tests.outcome == 'failure'
        uses: anthropics/claude-code-action@v1
        with:
          anthropic_api_key: ${ secrets.ANTHROPIC_API_KEY }
          prompt: |
            The tests in this repository failed. Their output is in test-output.txt.
            Explain in under 15 lines: which test fails, the most likely cause and
            which file to look at. Do not change any file.
          claude_args: '--max-turns 5 --allowedTools "Read,Grep,Glob"'
```

- `shell: bash` turns on `pipefail`. Without it, `npm test | tee` succeeds even when tests fail, and Claude never runs.
- `steps.tests.outcome == 'failure'` keeps Claude from running when what failed was `npm ci`.

- `--allowedTools "Read,Grep,Glob"` lets it read the output and the code, with no tools to edit.
- `--max-turns 5` and `timeout-minutes: 15` put a ceiling on the run.

If you already have a CI workflow, do not copy this one whole: add the `id` and the `tee` to your test step, the "Explain the failure" step, and `id-token: write` to the job's permissions.

STEP 05

@claude for documentation

.github/workflows/claude.yml

```
# Claude answers when someone with write access writes @claude in an issue or
# a PR: for example, "@claude document the summary function in docs/".
# Based on the "Respond to @claude mentions" example in the documentation.
name: Claude Code

on:
  issue_comment:
    types: [created]
  pull_request_review_comment:
    types: [created]

jobs:
  claude:
    if: contains(github.event.comment.body, '@claude')
    runs-on: ubuntu-latest
    timeout-minutes: 30
    permissions:
      contents: write
      pull-requests: write
      issues: write
      id-token: write
      actions: read
    steps:
      - uses: actions/checkout@v6
        with:
          fetch-depth: 1
      - uses: anthropics/claude-code-action@v1
        with:
          anthropic_api_key: ${ secrets.ANTHROPIC_API_KEY }
          claude_args: "--max-turns 20"
```

This workflow waits for someone with write access to mention `@claude`. Some documentation requests:

```
@claude document in docs/checkout.md how free shipping works and add an example
@claude update the README with the new npm run guides:build script
@claude explain in a comment what this PR changes for someone who did not write it
```

Claude answers in a comment it keeps updating and pushes the changes as commits. That is why it asks for `contents: write`, and why those commits get reviewed before merging.

CAUTION

Do not pass `github_token: ${ secrets.GITHUB_TOKEN }` to the action. GitHub does not trigger workflows on commits made with that token, so your CI would not run on Claude's changes.

STEP 06

Check that it works

1. Push the workflows on a branch and open a PR. `claude-review.yml` should run and comment on the PR.
2. On that branch, break a test on purpose and push. In the *Actions* tab, the CI job should fail and the "Explain the failure" step should show the explanation in its log.
3. Write on the PR: "@claude summarise in a comment what this PR changes". Claude should answer in a comment.
4. Fix the test, and merge only when everything is green and you have reviewed what Claude proposed.

Check each run's log for how many turns it used and how long it took. That is how you tune `--max-turns` and `timeout-minutes` for your repository.

GOING FURTHER

Costs and variations

What each run costs

Two things: GitHub Actions minutes and Claude tokens, or your subscription with the token. As a reference from our local tests: explaining a failure with Haiku, about USD 0.05; reviewing a 20-line file with Sonnet, about USD 0.15. It grows with the size of the diff, of `CLAUDE.md` and of what it reads. The docs recommend keeping `CLAUDE.md` short because it is read on every run.

Your own reviewer

If you have a reviewer subagent in `.claude/agents/`, the `checkout` puts it on the runner. You can change the review's `prompt` to use it, with the comment tool in `--allowedTools` if you want it to post on the PR.

Planning on an issue

With `--permission-mode plan` in `claude_args`, Claude proposes a plan without editing. Useful for having it leave a plan on an issue before anyone approves it.

Troubleshooting

SYMPTOM	LIKELY CAUSE	FIX
Claude does not answer <code>@claude</code>	The app or the secret is missing, or the author lacks write access	Check the app, the secret and the commenter's permissions
You wrote <code>/claude</code> or <code>@claude-bot</code>	The action looks for the whole word <code>@claude</code>	Write <code>@claude</code>
CI does not run on Claude's commits	You passed <code>github_token</code> with the default token	Remove that line
Authentication error	Invalid key or token	Test them locally with <code>claude</code> before debugging the workflow
Tests fail but Claude does not run	<code>shell: bash</code> or the step's <code>id</code> is missing	Compare with the ZIP's <code>ci.yml</code>
The review does not show on the PR	<code>--comment</code> or the tool in <code>--allowedTools</code> is missing	Copy both lines from the example
A bot triggers Claude and it fails	The action rejects bots by default	Add it to <code>allowed_bots</code> only if needed

Checklist

- ☐ `explain-failure` gave a useful explanation with your tests locally.
- ☐ The Claude app is installed and the secret saved; the key appears in no YAML.
- ☐ Each workflow has only its own permissions, plus `id-token: write`.
- ☐ Jobs have `timeout-minutes` and Claude steps have `--max-turns`.
- ☐ A test PR got the review, the failure explanation and the answer to `@claude`.
- ☐ `main` requires human review before merging.