

A Claude Code subagent that reviews your code

A read-only subagent that reviews the files you changed against a fixed checklist and returns findings with file, line and severity. Includes a file with deliberate bugs to try it on and a script that shows which tools it really used.

FOR	Claude Code on macOS, Linux or Windows
TESTED WITH	Claude Code 2.1.238 · Windows 11 · Haiku session, Sonnet reviewer
UPDATED	26 October 2026
DOCUMENTATION	code.claude.com/docs/en/sub-agents
ARTICLE	ecomlabs.dev/blog/claude-code-subagents-code-reviewer
AUTHOR	Luis Morales · @luismoralesdev

Contents

Before you start	What you will build
---------------------	---------------------

Step 01	Create the agents folder
---------	--------------------------

Step 02	The reviewer file
---------	-------------------

Step 03	Adapt the checklist to your project
---------	-------------------------------------

Step 04	Try it on the example file
---------	----------------------------

Step 05	Check what it really did
---------	--------------------------

Step 06	Check that it does not edit
---------	-----------------------------

Day to day	How to ask for a review
---------------	-------------------------

If something fails	Troubleshooting
--------------------------	-----------------

When you finish	Checklist
--------------------	-----------

What you will build

A subagent is an assistant the main Claude Code session calls for a specific task. It starts with a fresh context: it does not see your conversation or the files Claude already read. It does its work with its own tools and only the result comes back to your session.

The one in this guide is a read-only code reviewer. You hand it the files you changed, it goes through them with a five-point checklist and returns a report in a fixed format. It cannot edit anything because it does not have the tools to.

What is in the ZIP

FILE	WHAT IT DOES
<code>.claude/agents/code-reviewer.md</code>	The subagent: frontmatter, checklist and report format.
<code>example/coupons.ts</code>	Twenty lines with deliberate bugs to try the reviewer on. Do not use it in your store.
<code>test/show-tools.sh</code>	Shows, from the transcript, which tools the subagent used (bash).
<code>test/show-tools.ps1</code>	The same in PowerShell.

Requirements

- Claude Code installed and a project where you want code reviewed.
- For the check script: bash (macOS, Linux or Git Bash) or PowerShell 5.1 or 7.

STEP 01

Create the agents folder

A project's subagents live in `.claude/agents/`, at the repository root. Those in `~/ .claude/agents/` apply to all your projects; if two share a name, the project one wins.

```
# macOS, Linux or Git Bash
mkdir -p .claude/agents

# PowerShell
New-Item -ItemType Directory -Force .claude\agents
```

CAUTION

Claude Code watches the agents folders that existed when the session opened. If you just created `.claude/agents/` with Claude Code open, close it and open it again once. After that, any change to the file is picked up within a few seconds.

The reviewer file

Save it as `.claude/agents/code-reviewer.md`. It is the same file that comes in the ZIP.

`.claude/agents/code-reviewer.md`

```
---
name: code-reviewer
description: Read-only code reviewer. Use proactively when a change is finished, before a commit or a PR, or
when the user asks for a review. Pass it the list of files that changed.
tools: Read, Grep, Glob
model: sonnet
---
```

You are a code reviewer. You only read: you have no tools to edit files and you do not rewrite whole files in your answer.

Review only the files named in the task. If none are named, answer "I need the list of files that changed" and stop. You may open other files to understand the context (what they import, their tests), but do not review them.

Go through this list for each file:

1. Security: secrets in the code, unvalidated input, URLs, queries or commands built from text that comes from outside, personal data in logs.
2. Error handling: empty catch blocks, HTTP responses that are never checked, errors that get lost, code that carries on as if nothing failed.
3. Tests: new or changed logic with no test covering it. Before saying there are no tests, search for them with Glob and put the pattern you used in the finding.
4. Naming: functions or variables whose name does not say what they do.
5. Performance: network calls or queries inside a loop, work repeated for no reason.

Answer in this format and nothing else:

Summary

One or two sentences: what you reviewed and whether you would let it into a commit.

Findings

- [blocker|major|minor] file:line – what happens. Why it matters. How to fix it.

Order them from blocker to minor. If there are none, write "No findings".

Checklist

- Security: ok | N findings
- Error handling: ok | N findings
- Tests: ok | N findings
- Naming: ok | N findings
- Performance: ok | N findings

Do not invent problems to fill the list. If a finding depends on code you did not see, say so instead of guessing.

The frontmatter, field by field

FIELD	VALUE	WHY
name	code-reviewer	The name you call it by. Unique, no <code>:</code> .
description	When to use it and what it needs	Claude reads it to decide whether to delegate on its own. A vague description never gets used.
tools	Read, Grep, Glob	What makes it read-only. Leave the field out and it inherits every tool, Edit and Bash included.
model	sonnet	The reviewer's model, independent of your session's. Also accepts <code>opus</code> , <code>haiku</code> , <code>fable</code> , a full model ID or <code>inherit</code> .

The prompt

Below the frontmatter goes the subagent's prompt. It has three parts: what to review (only the files named in the task), how to review (the five-point list) and how to answer (a fixed format). The last line tells it not to invent problems to fill the list, and to say when a finding depends on code it did not see.

STEP 03

Adapt the checklist to your project

The five checks fit almost any JavaScript or TypeScript project. You can change them, but keep two rules: each check says what to look for, not just a topic, and the report format stays the same.

Example checks for a store

- **Checkout:** "Any change in `src/checkout/` or to prices, discounts or taxes is at least major. Flag floating-point arithmetic on money."
- **Shopify themes:** "In `.liquid` files, flag loops over `collection.products` without `limit` and visible text that skips translations."
- **Accessibility:** "Images without `alt`, buttons without accessible text, form fields without a `label`."
- **Analytics:** "Changes that remove or rename `dataLayer` events."

If a check grows large, split it into another subagent (for example `accessibility-reviewer`) with its own `description`. Write descriptions that do not overlap, so Claude knows which one gets each task.

The evidence rule

In our tests, the first version of the reviewer claimed it had searched for tests with Glob without doing so: its transcript only had a read of the file. Since check 3 asks it to write down the pattern it used, the search shows up in the transcript. If you add checks that depend on searching for something, ask for the same evidence.

Try it on the example file

1. Copy `example/coupons.ts` to `src/coupons.ts`.
2. Open Claude Code in the project and type: "Use the code-reviewer subagent to review `src/coupons.ts` and show me its report as is".
3. Wait for the report. Ask for it "as is" because otherwise the main session tends to summarise it in its own words.

The file has five deliberate problems. A correct report mentions all of them:

PROBLEM	LINE	CHECKLIST ITEM
Key written in the code and sent in the URL	4 and 15	Security
<code>code</code> put into the URL unencoded	15	Security
Empty <code>catch (e) {}</code>	18	Error handling
One <code>fetch</code> per line inside the loop	10 to 12	Performance
No test for <code>calculate</code>	—	Tests

In our runs, the reviewer found all five and also pointed out that no `fetch` response checks `r.ok` and that the total can go negative. Naming varied: once it flagged the function name as too generic and the other times it left it "ok". Each review of this file cost about USD 0.15 with the session on Haiku and the reviewer on Sonnet.

Check what it really did

Claude Code keeps each subagent's transcript in `~/.claude/projects/<project>/<session>/subagents/`, next to a `.meta.json` with its name. These scripts read the most recent one and list every tool it used.

`test/show-tools.sh`

```
#!/usr/bin/env bash
# show-tools.sh: shows which tools a subagent really used, reading the transcript
# Claude Code keeps in ~/.claude/projects.
# Usage:  bash test/show-tools.sh                (the most recent subagent)
#         bash test/show-tools.sh <agent-xxxx.jsonl>

file=${1:-$(ls -t "$HOME"/.claude/projects/*/subagents/agent-*.jsonl 2>/dev/null | head -n 1)}
[ -f "$file" ] || { echo "Cannot find any subagent transcripts in ~/.claude/projects" >&2; exit 1; }

echo "Transcript: $file"
meta="${file%.jsonl}.meta.json"
[ -f "$meta" ] && grep -o '"agentType": "[^"]*" ' "$meta" | sed 's/"agentType": "/Subagent: /; s/"$//'

# One line per call: the tool and what it received.
grep -o '{"type": "tool_use"[^}]*}' "$file" | sed -E 's/.*"name": "([^\"]+)", "input": /\1 /'
```

`test/show-tools.ps1`

```

# show-tools.ps1: shows which tools a subagent really used, reading the transcript
# Claude Code keeps in ~/.claude/projects.
# Usage: powershell -NoProfile -ExecutionPolicy Bypass -File test\show-tools.ps1
#         (add -Path <agent-xxxx.jsonl> to check a specific one)

param([string]$Path)

if (-not $Path) {
    $latest = Get-ChildItem -Path "$HOME\.claude\projects\*\*\subagents\agent-*.jsonl" -ErrorAction
SilentlyContinue |
    Sort-Object LastWriteTime -Descending | Select-Object -First 1
    if (-not $latest) {
        [Console]::Error.WriteLine('Cannot find any subagent transcripts in ~/.claude/projects')
        exit 1
    }
    $Path = $latest.FullName
}
if (-not (Test-Path -LiteralPath $Path)) {
    [Console]::Error.WriteLine("Not found: $Path")
    exit 1
}

"Transcript: $Path"
$meta = $Path -replace '\.jsonl$', '.meta.json'
if (Test-Path -LiteralPath $meta) {
    'Subagent: ' + (Get-Content -LiteralPath $meta -Raw | ConvertFrom-Json).agentType
}

# One line per call: the tool and what it received.
foreach ($line in Get-Content -LiteralPath $Path -Encoding UTF8) {
    $m = $line | ConvertFrom-Json
    foreach ($c in @($m.message.content)) {
        if ($c.type -eq 'tool_use') { '{0} {1}' -f $c.name, ($c.input | ConvertTo-Json -Compress) }
    }
}

```

```

bash test/show-tools.sh
powershell -NoProfile -ExecutionPolicy Bypass -File test\show-tools.ps1

```

For our run of the English reviewer, the output was this (path shortened):

```

Subagent: code-reviewer
Glob {"pattern":"**/coupons.ts"}
Read {"file_path":"...\src\coupons.ts"}
Glob {"pattern":"**/*coupon*"}
Glob {"pattern":"**/*.test.ts"}

```

If the report says it searched for tests and there is no `Glob` line here, the report is not reliable on that point. `Edit`, `Write` or `Bash` should never appear.

STEP 06

Check that it does not edit

Start a session with the reviewer as the main agent and ask it for a change, with edits pre-approved:

```
claude --agent code-reviewer --permission-mode acceptEdits
```

Type: "Fix the empty catch in `src/coupons.ts`". In our test (with the Spanish reviewer) it answered that it cannot edit files because it is a read-only reviewer with Read, Grep and Glob, and gave the suggested fix for us to apply by hand. Afterwards, `git status` should show no changes to `src/coupons.ts`.

When you are done, delete `src/coupons.ts`.

DAY TO DAY

How to ask for a review

WHEN	WHAT TO TYPE
Before a commit	"Use the code-reviewer subagent to review the files that changed according to <code>git diff --name-only</code> "
A specific folder	<code>@code-reviewer (agent)</code> review <code>src/checkout/</code>
A review-only session	<code>claude --agent code-reviewer</code>
After the review	"Apply the blocker findings from the report; leave the rest for later"

You pick the `@` mention from the list that appears when you type the at sign. It makes sure the task goes to that subagent and not another. The reviewer does not fix: the main session makes the fixes, with the report in front of it, and you decide which.

Troubleshooting

SYMPTOM	LIKELY CAUSE	FIX
Claude cannot find the subagent	The folder was created with the session open	Restart Claude Code once
Claude reviews it itself instead of delegating	You did not name the subagent	Use its name or the @ mention
The reviewer edits files	<code>tools</code> is missing from the frontmatter	Add <code>tools: Read, Grep, Glob</code>
It answers "I need the list of files that changed"	The task named no files	Give it paths or ask it to use <code>git diff --name-only</code>
The report arrives summarised or in another format	The main session rewrote it	Ask "show me its report as is"
It says it searched for tests but there is no <code>Glob</code>	The model claimed something it did not do	Keep the evidence rule from step 3
<code>show-tools</code> finds nothing	No subagent has run yet	Run a review first

Checklist

- ☐ `.claude/agents/code-reviewer.md` is in the project and committed.
- ☐ The frontmatter has `name`, `description` and `tools: Read, Grep, Glob`.
- ☐ The `description` says when to use it and that it needs the file list.
- ☐ The review of `example/coupons.ts` found all five problems.
- ☐ `show-tools` shows only `Read`, `Grep` and `Glob`.
- ☐ With `--agent`, it refused to edit and `git status` stayed clean.
- ☐ You deleted `src/coupons.ts`.