KNOWLEDGE-WORKER REFERENCE

Custom Slash Commands

A slash command is a paragraph of your own judgment you stopped retyping. Write it once; type a slash and a name after that. Full walkthrough and copy-paste versions at artificial-ideas.com.

artificial-ideas.com

Custom commands · part 2 of 2
Current as of June 2026 · Claude Code 2.1.x

TWO WAYS TO MAKE ONE

Quick: a markdown file at `.claude/commands/name.md` becomes `/name`. The file's contents are the instructions.

The real system: a Skill at `.claude/skills/name/SKILL.md` also becomes `/name`. Custom commands merged into Skills; both work, and a folder can hold templates and scripts too.

FRONTMATTER

At the top of `SKILL.md`, between `---` lines:

`description:` what it does and when. This is how Claude reaches for it on its own when it's relevant.

`disable-model-invocation: true` means only you can run it. Use it for anything that sends, posts, or overwrites.

PULL IN LIVE DATA

`$ARGUMENTS` — whatever you type after the command name.

`!`command`` — runs a shell command and drops its output in before Claude reads it.

`@file` — reads that file before answering, instead of you describing it.

A gallery to steal

One recipe per corner of knowledge work, none of them about code. The first is the two-minute version (a plain command file); the rest are full Skills. Drop one into the matching folder, change the specifics, and it's yours.

/tidy-notes

the two-minute version — a plain command file at `.claude/commands/tidy-notes.md`, no frontmatter

```
Turn the notes I'm sharing into a clean summary:
a one-line headline, three to five bullet points of what was
decided, and a separate list of action items with owners.
Keep it short. Don't invent anything that isn't in the notes.
```

/summarize-sources

for the research pile — the shape of ten links before you read every word

```
---
description: Summarize a set of sources into a comparison.
  Use when the user has gathered several articles, papers, or
  reports and wants the gist and the disagreements.
---

For each source I share, give me: a two-sentence summary, the
single strongest claim it makes, and how current it is. Then,
across all of them: where they agree, where they disagree, and
which one I should read in full first and why.
```

/numbers-check

for finance and data — checks a document against the actual figures

```
---
description: Check the numbers in a document against a data file.
  Use before sending anything with figures in it.
disable-model-invocation: true
---

## The data
@$ARGUMENTS

## Your task
Read the file above. Then check the document I'm sharing against
it: verify every figure, flag any number that doesn't reconcile,
and call out claims the data doesn't actually support. List the
discrepancies first, with the exact figures, before anything else.
```

/vet-source

the journalist's and analyst's reflex, written down once

```
---
description: Vet a source for reliability. Use when the user
  shares a link, a study, or a claim and asks whether to trust it.
---

Assess the source I'm sharing. Cover, briefly:
- who published it and whether they have a stake in the conclusion
- when it's from, and whether that still holds
- what the actual evidence is, versus what's asserted
- one thing I should independently check before relying on it

If you can't verify something, say so plainly. Don't pad.
```

/draft-from-transcript

for content people — a first draft, not a cleanup of someone's talking

```
---
description: Turn a raw transcript into a first draft. Use when
  the user has an interview, call, or talk transcript and wants
  an article, summary, or post out of it.
---

## The transcript
@$ARGUMENTS

## Your task
Pull a first draft out of the transcript above. Find the actual
through-line, not just the order things were said in. Quote the
sharp lines verbatim and attribute them. Cut the filler. Mark
anything that needs a fact-check with [CHECK] so I can see it.
```

/weekly-digest

the one that uses live data — nothing to paste in

```
---
description: Draft this week's digest from recent files.
  Use on a regular cadence to summarize what changed.
---

## This week's updated files
!`find . -name "*.md" -mtime -7 -not -path "*/.*"`

## Your task
For each file listed above, read it and pull what's new or
decided. Write a digest grouped by theme, newest first, with a
two-line "what to pay attention to" at the top. Keep it to one
screen.
```

Your turn

the test: a paragraph you retype more than twice a week

Any instructions you paste into chat over and over — a checklist, a five-step procedure, the way you always want notes cleaned up — is a command waiting to happen.

The first one takes about two minutes. Once you have a handful, a fair amount of your week runs on instructions you only had to write down once.