A RESOURCE FOR TEAMS WORKING WITH CODING AGENTS

CLAUDE.md structure template.

The file every session reads first, kept small enough to be read.

Facts in CLAUDE.md. Procedures in skills. Guarantees in hooks.

28	16	200
rules, each with its check	linter rules in the kit	line budget per file

Write down what you would otherwise repeat.

Claude Code reads CLAUDE.md at the start of every session and again after every compaction. Every line is sent with every request, and the docs are plain that longer files "consume more context and reduce adherence" [S01]. Teams get it wrong in two directions: a file that says "this is a React project", or a wiki that Claude skims.

This guide gives the file a fixed structure and a budget, and draws the line the docs draw: CLAUDE.md holds facts needed in every session; file-specific rules, procedures and guarantees go elsewhere [S01, S03]. The kit's linter checks most rules. It reports nothing on the good example and 10 errors and 14 warnings on the bad one.

Version 1.0 / Sources checked 24 September 2026
Field Guide 28 of the Design × AI series, with 29 (skills), 30 (verification) and 31 (context), all using the Acme UI design system as the example. Checked against the Claude Code docs (behavior up to v2.1.281); kit scripts tested on Node.js 22.22 with no dependencies.

Practical guidance, not a standard. Claude Code documents where CLAUDE.md lives and how it loads; the section structure, the linter rules and the 300-line always-loaded budget are this guide's recommendations. Prepared with AI assistance and edited by hand.

START HERE

Know which file does which job.

Claude Code reads several instruction files, and each loads at a different time. Put each instruction where it costs the least and still arrives when it is needed.

File	Loads	Use it for
`CLAUDE.md` or `.claude/CLAUDE.md`	Every session, re-read after compaction	Commands, definition of done, conventions, decisions, gotchas
`CLAUDE.local.md` (gitignored)	Every session, after CLAUDE.md	Your own sandbox URLs and preferences for this repo
`~/.claude/CLAUDE.md`	Every session, every project	Personal preferences that follow you
`.claude/rules/*.md` with `paths`	When Claude reads a matching file	Rules for one part of the tree: components, migrations, docs
`.claude/skills/<name>/SKILL.md`	Description always; body when used	Procedures and long reference material (Field Guide 29)
Hooks in `.claude/settings.json`	On a lifecycle event, outside the context	Anything that must happen every time (Field Guide 30)
`AGENTS.md`	Instead of CLAUDE.md when there is none, or through `@AGENTS.md`	Instructions shared with other coding agents

Load behavior from the Claude Code memory and extension docs [S01, S03]. Auto memory (`MEMORY.md`) is written by Claude, not by you, and is not covered here.

Starting from nothing

Run `/init`, then rewrite what it produced into the template's sections. Keep only lines Claude would get wrong without them.

Cleaning up a long file

Run the linter, then section 04: move procedures to skills and file-specific rules to `.claude/rules/`. The size error goes away as a side effect.

Sharing with other agents

Keep shared instructions in `AGENTS.md` and import it with `@AGENTS.md` (item C24). Claude-specific lines go below the import.

Keeping it honest

Section 06: an owner, the linter in CI and `/context` to confirm what actually loaded.

Label	Meaning
`DOCS`	Behavior or guidance stated in the Claude Code documentation.
`KIT CHECK`	Checked by `lint-claude-md.mjs`; the rule id is in the evidence line.
`PRACTICE`	A recommended working method; the evidence line says what to review.

SECTION 01

Decide what earns a line.

For each line, ask whether removing it would make Claude get something wrong. If not, cut it [S02].

- [ ] **Only what Claude cannot learn from the code**

  C01 DOCS

  Commands it cannot guess, non-default conventions, decisions, quirks, gotchas. Not file listings, standard conventions or API docs.

  Evidence: In review, each line survives the question "would Claude get this wrong without it?"

  Include and exclude table in the best practices. S02.

- [ ] **A line is added when a mistake repeats**

  C02 DOCS PRACTICE

  The trigger is a repeated mistake, a review comment Claude should have anticipated, or a correction you typed last session too.

  Evidence: The commit that adds a line names the correction or review comment behind it.

  When to add to CLAUDE.md. S01.

- [ ] **Every instruction can be checked**

  C03 DOCS KIT CHECK

  "Run `npm test` before committing", not "test your changes". If you cannot tell whether it was followed, Claude cannot either.

  Evidence: Linter rule `vague-instruction`: 4 hits on the bad example, none on the good one.

  Specificity guidance. S01.

- [ ] **No content Claude can derive**

  C04 DOCS KIT CHECK

  Pasted directory trees, dependency lists and architecture overviews go stale and cost tokens. `/doctor` proposes cutting exactly these.

  Evidence: Linter rule `derivable-content`: the bad example's 30-line tree is flagged.

  `/doctor` trim check, v2.1.206 and later. S01.

- [ ] **No dates and no "recently"**

  C05 KIT CHECK

  "We recently moved to Vitest" is true for a month and wrong for a year. State the current rule; delete it when it stops being true.

  Evidence: Linter rule `time-sensitive`: 2 hits on the bad example.

  Anthropic's skill guidance on time-sensitive content, applied to instruction files. S04.

The budget is attention, not disk space.

SECTION 02

Use a fixed set of sections.

The docs require no format [S01]. Fixed headings make files comparable and let a linter find what is missing.

- [ ] **Commands, exactly as typed**

  C06 KIT CHECK

  Install, dev, typecheck, lint, test (and one test file) and build, in one fenced block. These are how Claude verifies its own work.

  Evidence: Linter rules `required-section` and `stale-command`: the bad example names 3 scripts package.json does not have.

  Bash commands Claude cannot guess. S02.

- [ ] **Verification: the definition of done**

  C07 KIT CHECK

  Which commands must exit 0, what to look at for UI changes, what the final message reports. Field Guide 30 turns it into a script and a hook.

  Evidence: Linter rule `required-section`: the bad example has none and fails.

  Give Claude a way to verify its work. S02.

- [ ] **Conventions that differ from the defaults**

  C08 KIT CHECK

  `tone`, never `variant`; string unions for closed sets; one component per folder. Leave out what any TypeScript developer already does.

  Evidence: Linter rule `required-section`; review against the exclude list in C01.

  Code style rules that differ from defaults. S02.

- [ ] **Decisions with their reasons, not a map**

  C09 PRACTICE

  "Semantic tokens only, because core tokens carry no intent", with a link to the record. A reason lets Claude handle cases the line did not foresee.

  Evidence: Every decision has a "because" and a path the linter confirms exists.

  Architectural decisions specific to your project. S02.

- [ ] **Gotchas, and environment names without values**

  C10 PRACTICE

  What has cost someone an afternoon: jsdom cannot test layout, a generated folder is never edited. Variables by name only.

  Evidence: Review: each gotcha is non-obvious and still true.

  Common gotchas and developer environment quirks. S02.

Three required sections, and every other heading earns its place.

SECTION 03

Stay inside the budget.

Claude Code loads up to 4 MiB of CLAUDE.md [S01]. That is a ceiling; the documented target is under 200 lines.

- [ ] **Under 200 loaded lines**

  C11 DOCS KIT CHECK

  Counted after HTML comments are stripped. The linter errors above 200 and warns above 150. The good example is 37 lines.

  Evidence: Linter rule `size-lines`: the bad example's 204 lines fail.

  Target under 200 lines per CLAUDE.md file. S01, S03, S08.

- [ ] **Imports count toward the budget**

  C12 DOCS KIT CHECK

  `@path` imports are expanded at launch, and unscoped rules load every session. Splitting a file organizes it; it does not shrink it.

  Evidence: Linter rule `always-loaded` (local budget 300): the good example loads 59 lines, 37 plus 22 from AGENTS.md.

  Imported files still load at launch. S01.

- [ ] **Imports resolve, and package names are quoted**

  C13 DOCS KIT CHECK

  Imports resolve relative to the importing file, at most four hops deep. Write `@acme/ui` in backticks; unquoted, it parses as an import.

  Evidence: Linter rules `import-missing`, `import-depth`, `import-ambiguous`: 1 missing file and 16 ambiguous names on the bad example.

  Import syntax, depth and code-span rules. S01.

- [ ] **Emphasis on one line, if any**

  C14 DOCS KIT CHECK

  Emphasis such as IMPORTANT can rescue one skipped instruction. On many lines none stands out, and the file is probably too long.

  Evidence: Linter rule `emphasis` warns above two emphasized lines; the bad example has three.

  Best practices on emphasis. S02.

- [ ] **Notes for maintainers go in HTML comments**

  C15 DOCS

  Block-level `<!-- ... -->` comments are stripped before loading, so owner and review notes cost no context. The kit's template uses them.

  Evidence: The linter counts lines after stripping comments: the template loads 48.

  HTML comments in CLAUDE.md. S01.

If a rule keeps getting ignored, the file is probably too long.

SECTION 04

Move it where it belongs.

Most oversized files are not wrong, they are misplaced. Each move makes the file shorter and the instruction more reliable.

- [ ] **File-specific rules go to .claude/rules with paths**

  C16 DOCS

  A rule that only matters for component source goes in `.claude/rules/<topic>.md` with `paths`. It loads with matching files and costs nothing otherwise.

  Evidence: The good example's two rules are path-scoped, so neither counts as always loaded.

  Path-specific rules. S01, S03.

- [ ] **Rules read only `paths`**

  C17 DOCS KIT CHECK

  Other fields are ignored without an error. A rule copied from Cursor with `globs` and `alwaysApply` loads unconditionally, every session.

  Evidence: Linter rule `rule-frontmatter`: 3 warnings on the bad example's Cursor-style rule.

  Rule frontmatter reference. S01.

- [ ] **Procedures go to skills**

  C18 DOCS KIT CHECK

  A release checklist is a skill: only its description is in context until it runs. Leave a pointer: "New component: run `/create-component`".

  Evidence: Linter rule `procedure`: the bad example's 12-step release process is flagged.

  Multi-step procedures belong in a skill. S01, S04, S08.

- [ ] **Guarantees go to hooks**

  C19 DOCS KIT CHECK

  "Run the formatter after every edit" is a request here. A PostToolUse hook is a guarantee and costs no context. Configs are in Field Guide 30.

  Evidence: Linter rule `hook-shaped` flags "after every edit" and "before each commit".

  CLAUDE.md is context, not enforced configuration; use hooks for things that must run at a fixed point. S01, S05.

- [ ] **Personal preferences stay personal**

  C20 DOCS

  Sandbox URLs go in `CLAUDE.local.md` (gitignored); cross-project preferences in `~/.claude/CLAUDE.md`. Across worktrees, import a file from your home folder.

  Evidence: No personal URLs or names in the committed file.

  CLAUDE.md locations and scopes. S01.

A long CLAUDE.md is usually a skill, a rule and a hook that have not been written yet.

SECTION 05

Keep it safe, current and shared.

The file travels: into every clone, every session and every compaction. Make sure what travels is true and harmless.

- [ ] **No secrets, ever**

  C21 KIT CHECK

  The file is committed and sent to the model every session. Name the variable, never its value.

  Evidence: Linter rule `secret`: 2 hits on the bad example.

  Shared through source control. S01.

- [ ] **No stale paths or commands**

  C22 KIT CHECK

  A missing path teaches the wrong layout; a removed script sends Claude to a failing command.

  Evidence: Linter rules `stale-reference`, `stale-command`: 5 findings on the bad example.

  Linter design; build output such as `dist` is skipped.

- [ ] **Say what compaction must keep**

  C23 DOCS

  One line: "keep the files changed and the commands run". Root CLAUDE.md is re-read after `/compact`, so it survives.

  Evidence: After `/compact`, the summary contains what the section names.

  S02, S06, S07.

- [ ] **Share with other agents through AGENTS.md**

  C24 DOCS

  Put what every agent needs in `AGENTS.md`; start CLAUDE.md with `@AGENTS.md`. Claude Code 2.1.277+ reads AGENTS.md itself only when no CLAUDE.md exists.

  Evidence: The good example imports AGENTS.md; the linter finds Commands and Conventions there.

  S01, S09. Field Guide 04 has an AGENTS.md snippet.

Everything in CLAUDE.md is published to everyone who clones the repo.

SECTION 06

Maintain it like code.

The file compounds in value only if someone keeps it true. Treat it like a build config.

- [ ] **It has an owner and goes through review**

  C25 PRACTICE

  One named owner; changes reviewed like code. Anthropic's own advice is the same: under 200 lines, an owner, code review.

  Evidence: A CODEOWNERS entry for CLAUDE.md and `.claude/`.

  S08. S02 on treating CLAUDE.md like code.

- [ ] **The linter runs in CI**

  C26 KIT CHECK

  On every pull request that touches CLAUDE.md, AGENTS.md, `.claude/` or package.json, so stale paths are caught when they happen.

  Evidence: A CI job running `lint-claude-md.mjs`; exit code 1 on any error.

  Kit script.

- [ ] **Confirm what actually loaded**

  C27 DOCS

  `/context` lists the memory files that loaded; `/memory` opens them. An `InstructionsLoaded` hook logs which rules loaded, and why.

  Evidence: A paste of `/context` in the pull request that restructures the file.

  Troubleshooting memory. S01, S10.

- [ ] **Test changes by watching behavior, then prune**

  C28 DOCS PRACTICE

  After an edit, rerun the task the line was meant to fix. If Claude already does it right without the line, delete the line or make it a hook.

  Evidence: The pull request names the task used to test the change.

  Treat CLAUDE.md like code; prune regularly. S02.

A CLAUDE.md nobody owns drifts toward the wiki it was meant to replace.

APPENDIX A

The template, and the example that fills it.

`CLAUDE.template.md` in the kit carries its writing notes in HTML comments, which Claude Code strips before loading. Below is the filled-in version for the Acme UI design system, which imports the shared `AGENTS.md`.

examples/good/CLAUDE.md (excerpt; the file is 37 loaded lines)

```
@AGENTS.md

## Verification

Before you say a task is done:

1. Run `npm run typecheck`, `npm run lint`, `npm test` and `npm run build`. All exit 0.
2. For a component change, run `npm run test-storybook -- --url http://localhost:6006`
   against a running Storybook, and open the story to check it visually.
3. In your final message, list each command with its exit code, the stories you
   checked, and anything you could not verify.

## Workflows

- New component: run `/create-component <Name>`. The skill scaffolds the files.

## Compaction

When compacting, keep the list of files changed, the commands run with their exit
codes, and any contract fields still marked TODO.
```

Commands and conventions live in AGENTS.md, which every agent reads; Claude-specific sections follow the import.

Section	Required	Holds
Project	no	One or two sentences; only what the code cannot tell
Commands	yes	Exact commands, including one test file
Verification	yes	The definition of done and what to report
Conventions	yes	Only rules that differ from defaults
Architecture decisions	no	Decision, reason, record
Gotchas	no	Non-obvious behavior; variable names, not values
Where things live	no	Only paths Claude would guess wrong
Compaction	no	What a summary must keep

Required sections may come from an imported file: the linter reads headings in CLAUDE.md and everything it imports.

APPENDIX B

Run the linter.

Zero dependencies; `node scripts/lint-claude-md.mjs path/to/CLAUDE.md`. The folder of the file is taken as the repository root unless you pass `--root`. Output below is from the kit.

terminal

```
$ node scripts/lint-claude-md.mjs examples/good/CLAUDE.md
examples/good/CLAUDE.md: 37 loaded lines (~455 tokens), 59 lines always loaded
with imports and unscoped rules. 0 error(s), 0 warning(s).

$ node scripts/lint-claude-md.mjs examples/bad/CLAUDE.md
CLAUDE.md          error    size-lines        204 loaded lines; the budget is 200.
CLAUDE.md:204      error    import-missing    @docs/old-setup.md does not resolve to a file.
CLAUDE.md          error    required-section  No "## Verification" section ...
CLAUDE.md:65       error    stale-reference   `src/legacy/theme.ts` does not exist ...
CLAUDE.md:14       error    stale-command     `npm run dev` names a script that ...
CLAUDE.md:25       error    secret            Looks like a secret (credentials in a URL).
CLAUDE.md:69       warning  import-ambiguous  16 unquoted @names parse as imports ...
CLAUDE.md:146      warning  procedure         A 12-step procedure. ... (Field Guide 29)
...
examples/bad/CLAUDE.md: 204 loaded lines (~1785 tokens), 214 lines always loaded
with imports and unscoped rules. 10 error(s), 14 warning(s).
```

Abbreviated; the full run lists all 24 findings with line numbers. Token counts are estimates (characters divided by four), not tokenizer output.

Rule	Severity	Checks
`size-lines`	error	Over 200 loaded lines (warning over 150)
`always-loaded`	warning	CLAUDE.md, imports and unscoped rules over 300 lines
`required-section`	error	No Commands, Verification or Conventions heading
`import-missing`, `import-depth`	error	Broken import; more than four hops
`import-ambiguous`	warning	Unquoted `@scope/name` that matches no file
`stale-reference`, `stale-command`	error	Missing path; `npm run` script not in package.json
`secret`	error	Keys, tokens, private keys, credentials in URLs
`vague-instruction`, `time-sensitive`	warning	Uncheckable or dated wording
`emphasis`, `procedure`, `hook-shaped`	warning	Belongs on one line, in a skill, or in a hook
`derivable-content`, `rule-frontmatter`	warning	Pasted trees; rule fields other than `paths`

Thresholds live in `claude-md-lint.config.json`. The secret patterns are a first line of defense, not a replacement for a secret scanner.

KEEP WITH THE REPOSITORY

Leave a CLAUDE.md review record.

One record per restructure or quarterly review. It shows what was cut, what moved and whether behavior changed.

Repository / owner	**Loaded lines before / after**
Always-loaded lines before / after	**Linter run (link)**
Lines moved to rules (files)	**Lines moved to skills (names)**
Lines replaced by hooks (events)	**Task used to test the change**
/context screenshot (link)	**Next review date**

> **Count what you removed.**
> A review that only adds lines is not a review. The linter's loaded-line count is the number to watch between reviews.

SOURCES / MAINTENANCE

Keep the guide current.

Sources checked 24 September 2026. Claude Code changes often; the docs note version requirements such as v2.1.277 for reading AGENTS.md directly, so check `claude --version` before relying on a recent behavior.

S01 / Claude Code docs, how Claude remembers your project
CLAUDE.md locations, load order, 200-line target, imports and four-hop depth, HTML comments, .claude/rules and paths, AGENTS.md, /doctor trims, compaction.
https://code.claude.com/docs/en/memory

S02 / Claude Code docs, best practices
Include and exclude table, pruning, emphasis, verification, compaction instructions.
https://code.claude.com/docs/en/best-practices

S03 / Claude Code docs, extend Claude Code
CLAUDE.md versus rules, skills, hooks and subagents; context cost by feature.
https://code.claude.com/docs/en/features-overview

S04 / Anthropic, skill authoring best practices
Avoid time-sensitive information; consistent terms.
https://platform.claude.com/docs/en/agents-and-tools/agent-skills/best-practices

S05 / Claude Code docs, hooks guide
Hooks as deterministic automation; re-injecting context after compaction.
https://code.claude.com/docs/en/hooks-guide

S06 / Claude Code docs, explore the context window
What survives compaction: project-root CLAUDE.md and unscoped rules are re-injected from disk.
https://code.claude.com/docs/en/context-window

S07 / Claude Code docs, manage costs
Compact instructions in CLAUDE.md; move workflow instructions to skills.
https://code.claude.com/docs/en/costs

S08 / Claude blog, Steering Claude Code
When to use CLAUDE.md, rules, skills, hooks and subagents; owner and review (June 2026).
https://claude.com/blog/steering-claude-code-skills-hooks-rules-subagents-and-more

S09 / AGENTS.md
The shared instruction file convention for coding agents.
https://agents.md/

S10 / Claude Code docs, commands
/init, /memory, /context and /doctor.
https://code.claude.com/docs/en/commands

Maintenance: recheck the memory, best practices and extension pages at each Claude Code minor release, and rerun the linter against the good and bad examples after any rule change. Update the PDF, HTML, Markdown and JSON together.