.CLAUDE/SKILLS/DRAFT-INTERNAL-DOC/SKILL.MD

```
name: draft-internal-doc
description: |
  Create a new Confluence page from scratch. Use this skill when an engineer
  already knows what they want to write and needs help structuring and publishing
  it — doc-type selection, duplicate checking, structure generation, section
  drafting, and final page creation in Confluence. Includes a transcript-to-draft
  shortcut for turning a verbal explanation into a first draft. Trigger on phrases
  like "create a page", "write a doc", "new confluence page", "I need to document
  X", "draft a page about", or "add a page to" when the engineer already has the
  content organized. If the engineer needs help gathering scattered sources or
  doesn't have the content organized yet, use research-doc instead.
```

Page Creation

Guide the user from a raw idea to a published Confluence page. Follows the same two-question intake, duplicate check, structure, draft, and create sequence as the Rovo "Help me write something" scenario — adapted for Claude Code conventions.

Prerequisites

MCP tool	Purpose
`mcp__atlassian__*`	Fetch space structure, create pages, look up parent pages
`mcp__glean_default__search` / `read_document`	Duplicate check, gather existing context

If either MCP is unavailable, note which is missing and continue with reduced capability: skip the duplicate check if Glean is absent; skip page creation if Atlassian is absent and offer to output the body as Markdown instead.

A practical shortcut — start with a transcript

Before asking the two intake questions, check whether the user has a meeting transcript or verbal explanation to work from. If they do, this is faster than starting blank: paste the transcript into the session with a prompt like "turn this into a structured how-to guide for [audience]" and use the output as a first draft to edit. This works especially well for:

- Process documentation that was explained verbally
- Onboarding walkthroughs recorded in Zoom
- Decision records where the rationale was discussed but never written down
- Lunch & Learn sessions worth preserving

If no transcript is available, move directly to Phase 1.

Phase 1 — Intake (two questions)

Ask both questions together in a single message. Do not ask more than two questions at once. If the answers make the doc type obvious, move straight to Phase 2. If not, ask one more targeted question — never a list of follow-ups.

If the user has already answered both in their opening message, extract the answers and skip to Phase 2 without asking.

Q1 — What is this doc for? What should the reader be able to do or understand after reading it?

Q2 — Who is the reader? A new team member, an experienced engineer, a cross-functional stakeholder, an SRE on-call?

Phase 2 — Doc-type identification

Map the intake answers to one of the four Playbook doc types. State the chosen type and one sentence explaining why. If two types are plausible, name both, recommend one, and ask the user to confirm before continuing.

Doc type	When to use	Key signals
Walkthrough	Reader is doing something for the first time	Sequential steps, stated learning outcome, expected result at the end
Task guide	Reader knows the domain and wants concise instructions	Goal, prerequisites, numbered steps, troubleshooting
Explainer	Reader needs to understand, not do	No steps; covers what it is, why it exists, how it works, further reading
Reference	Reader is looking something up	Minimal prose; tables, lists, maximum scannability

Doc types can blend — a runbook might have a short explainer introduction before the task guide steps. That's fine; name the blend deliberately so the structure reflects it ("This page opens with a brief explainer section, then shifts into task guide format for the steps.").

If the structure the user proposes doesn't match what the reader needs, say so directly: "This sounds like a task guide — the reader wants to complete something, not understand something. An explainer structure would leave them without the steps they actually need."

Confirm the doc type with the user before moving to Phase 3.

Phase 3 — Duplicate check

Before generating any structure, search for existing content. Run both searches in parallel.

```
mcp__glean_default__search               query: "<topic> <space or system name>"
mcp__atlassian__searchConfluenceUsingCql  cql: 'title ~ "<key terms>" AND space = "<space key>"'
```

If no close matches found:

> *No duplicate found. Proceeding to structure.*

If close matches found:

> *Found possible overlap: - Page title — one sentence on what it covers - Page title — …*
>
> *Recommendation: [merge into existing / supersede with redirect / proceed as a distinct doc because…]. How would you like to proceed?*

Two pages on the same topic with conflicting information damage trust in Glean search results — do not proceed past this phase until the user has confirmed they want a new page rather than an edit to an existing one.

Phase 4 — Structure

Generate a concrete outline tailored to the confirmed doc type and the specific topic. Use the templates below as a starting point, not a rigid form — adjust section names and depth to fit the content.

Lead with one sentence noting what the structure is optimised for:

> *"Structure below is optimised for an engineer who knows Kubernetes but is new to this team's deployment pipeline."*

Present the outline and offer to adjust before drafting anything.

Section templates by doc type

Walkthrough

```
## Overview
## Prerequisites
## Step 1 — <action>
## Step 2 — <action>
## Step N — <action>
## Verification
## Troubleshooting
## Next steps
```

Task guide

```
## When to use this guide
## Prerequisites
## Steps
## Expected output / verification
## Troubleshooting
## Related tasks
```

Explainer

```
## Summary (≤ 3 sentences — what this is and why it exists)
## Background / motivation
## How it works
## Key concepts
## Trade-offs and limitations
## Further reading
```

Reference

```
## Overview
## <Entity / Flag / Config / API surface> table
## Notes and constraints
## Changelog (if appropriate)
```

Phase 5 — Title and opening paragraph

Produce these two items automatically after the outline is confirmed — do not wait to be asked.

Title

Write a suggested title. If the user supplied one, evaluate it and offer a rewrite if it has a must-fix or should-fix issue (see Output conventions below).

Title conventions: - Specific — include the system, team, or product name - Under 60 characters - Sentence case - **For task guides:** start with a verb ("Configure X," "Set up Y") — the reader is goal-oriented and the verb signals actionability - **For walkthroughs, explainers, reference:** lead with the subject, not the verb ("Kubernetes node draining," not "How to drain Kubernetes nodes") - No dates, version numbers, or "v2" suffixes - Unique in the space — check for near-duplicate titles during Phase 3

The title is the highest-weighted Glean signal. A specific title is the single highest-leverage thing the author can do for discoverability.

Opening paragraph

Draft 1–3 sentences of prose that will appear before any macro, table of contents, or image. This text serves two purposes simultaneously: it orients the reader, and it becomes the Glean search result snippet.

The opening paragraph must: - Include the primary search terms a reader would use to find this page - Include common synonyms for key systems (Glean matches queries against the terms that appear in the doc — synonyms expand recall) - Make clear what the page covers and who it is for - Be self-contained: a reader who only sees this snippet in Glean results should know whether the page is relevant to them

Tell the user that this paragraph doubles as the Glean snippet so they understand why it matters and can edit it with both audiences in mind.

After drafting, briefly note what you assumed about the reader and ask if anything needs adjusting.

Phase 6 — Section drafting

Do not draft all sections unprompted. After presenting the title and opening paragraph, offer:

> *Which section would you like to draft next? Or I can produce the full page in one pass if you'd prefer.*

Draft one section at a time unless asked for more.

Writing standards (apply to every drafted section)

- Second person, present tense, active voice throughout
- Paragraphs 3–5 sentences or shorter
- No jargon without a definition on first use
- American English spelling
- All acronyms defined on first use
- Code font on file names, commands, paths, and literal values
- Code blocks with a language tag
- UI elements bolded (`click **Save**`)
- Link text is descriptive — never "click here"
- Serial commas

Accessibility (apply to every drafted section)

- All images must have meaningful alt text — describe what the image conveys, not just what it shows ("Diagram showing three-tier architecture with load balancer, application server, and database" — not "architecture.png")
- Never rely on color alone to convey meaning — add a label or description
- Do not use all caps for emphasis — screen readers interpret it as an acronym
- Bold and italic are not reliably conveyed by screen readers — use them for formatting only, not as a substitute for structural emphasis
- For charts or complex diagrams, include a text summary or data table alongside
- For video or audio content, include captions or a transcript

Phase 7 — Placement

Recommend where in the space hierarchy the new page should live. Apply the four-folder structure:

Folder	Contents

Getting started	First-time walkthroughs, onboarding paths, prerequisites
How-to guides	Task guides, operational runbooks
Concepts	Explainers, architecture docs, decision records
Reference	Tables, glossaries, API docs, config references

Space landing pages (the top-level index page for a space or system) live at the root of the space, not inside a folder.

State the recommended parent page explicitly:

> *Recommended placement:* `<Space key> / How-to guides / <subsystem if applicable>`

If there is not enough context to pick the right subsystem folder, list the top-level options and ask the user to choose.

Labels

Before recommending labels, check what labels already exist on nearby pages in the same space:

```
mcp__atlassian__searchConfluenceUsingCql  cql: 'space = "<space key>" AND label = "<candidate>"'
```

Use existing labels that apply before recommending new ones — a consistent taxonomy only works if people use the same labels rather than inventing variations.

The CoreWeave label taxonomy is a useful starting framework, not a strict requirement:

Pattern	Example	Purpose
`team-<name>`	`team-ka`, `team-storage`	Feeds team-scoped Glean Collections
`doc-<type>`	`doc-runbook`, `doc-onboarding`, `doc-reference`	Feeds doc-type filtered views
`<tech-name>`	`kubernetes`, `terraform`, `gpu`	Technology surface area

Always add `needs-review` on creation. The page owner removes it once a teammate has signed off.

When suggesting labels, briefly explain what each one does — which Glean Collection it feeds or what filtered view it appears in — so the author understands the value rather than just following instructions.

Never invent labels that fragment the taxonomy or duplicate existing ones with slight variations (e.g., don't add `k8s` if `kubernetes` already exists in the space).

Phase 8 — Creation

Summarise what will be created before calling the MCP tool:

```
Title:    <title>
Space:    <space key>
Parent:   <parent page title>
Labels:   <label list>
Doc type: <type>
Status:   Draft
```

Ask: "Create this page now, or would you like to review the full draft first?"

When the user confirms, create as a draft so they can review before publishing:

```
mcp__atlassian__createConfluencePage
  spaceKey:  <key>
  parentId:  <id of parent page>
  title:     <title>
  body:      <content in Confluence storage format>
  labels:    [<labels>]
  status:    draft
```

After creation, return: - The page URL - The three post-creation steps below

Post-creation checklist (share with the user)

1. **Link from your team landing page** — pages linked from many places rank higher in Glean. Add a link to this page from the space or team index page.
2. **Add to any relevant Glean Collections** — if this page belongs in a curated collection, add it now while the context is fresh.
3. **Verify in Glean once published and set the page's Confluence Page Status to a team-agreed current value** — together these give readers both an in-search and in-page signal that the content is trustworthy. Remove the needs-review label at the same time.

If any sections were not drafted before creation, note them:

> *"Sections left to write: Troubleshooting, Next steps — reply to draft these."*

Output conventions

Tier structure

Use this when evaluating any user-supplied content (title, opening paragraph, proposed outline):

Tier	Meaning
Must fix	Blocks findability or usability — e.g., title conflicts with an existing page, opening paragraph is missing, doc type doesn't match reader need
Should fix	Reduces quality significantly — e.g., title uses generic phrasing, opening paragraph omits the reader, Glean snippet is vague
Nice to have	Polish — e.g., tighter phrasing, stronger section ordering

Automatic rewrites: For any must-fix or should-fix finding affecting the title or opening paragraph, offer a rewrite immediately — give the author something concrete to react to. For longer sections, offer a rewrite but don't produce it unprompted.

Note what's working first. Before listing problems with user-supplied content, lead with one sentence on what is good:

> *"Title is clear and includes the system name — one adjustment will improve task-guide discoverability:"*

Framing: All suggestions are suggestions. Use "consider," "recommend," "one option is" — not "you must" or "this is wrong." The author knows their content and audience better than this skill does.

Consistency with Rovo Docs Assistant

This skill and the Rovo "Help me write something" scenario cover the same surface. Output should be consistent across both so users get the same result regardless of which tool they used.

Shared standards (do not diverge from these): - Same four doc types with the same structural templates - Same four-folder placement logic - Same label taxonomy and the "check existing labels first" rule - Same Glean snippet requirement and synonym inclusion guidance - Same title conventions including verb-first for task guides

If the user references a page they recently created via Rovo, fetch it with `mcp__atlassian__getConfluencePage` or `mcp__glean_default__read_document` and match its formatting conventions before drafting adjacent content.

Pre-creation checklist

Run through this before calling `createConfluencePage`:

- [ ] Doc type confirmed by user
- [ ] Transcript shortcut offered (if applicable)
- [ ] Duplicate check completed — no unresolved overlaps
- [ ] Outline agreed
- [ ] Title follows conventions (specific, correct case, verb-first for task guides)
- [ ] Opening paragraph drafted and works as a Glean snippet
- [ ] Synonyms for key systems included in opening paragraph
- [ ] Accessibility standards applied to all drafted sections
- [ ] Placement confirmed (space + parent page)
- [ ] Existing labels checked before new ones recommended
- [ ] Labels listed including `needs-review`
- [ ] User has seen the creation summary and confirmed
- [ ] Page created as draft, not published