repository: shanraisshan/claude-code-best-practice

Claude Code Best Practice:

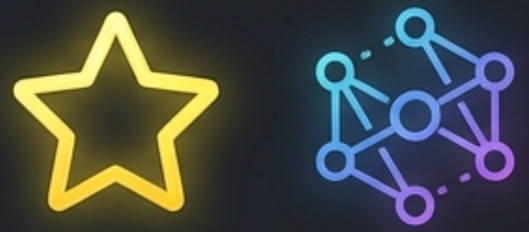
単なるチャットボットから「自律型開発パートナー」へ

Deep Dive for Technical Leaders & Developers


```
{
  "status": "verified",
  "stars": "1.5k+",
  "version": "2.0"
}
```

なぜ今、ベストプラクティスが必要なのか

The Source (Fact)



GitHub: shanraisshan/claude-code-best-practice

Metrics: **1.5k+ Stars**

Validation: Highly cited by @tom_doerr & global dev community.
「検証済みのワークフローとアーキテクチャパターン」

The Problem (Interpretation)

Treating Claude like a web chat leads to:

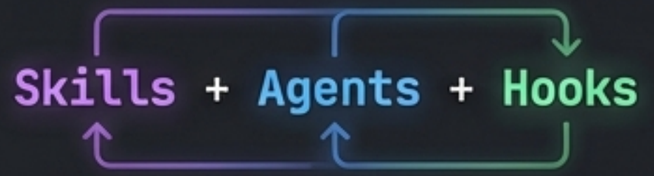
- **Context Bloat (肥大化)**
- **Token Burn (無駄遣い)**
- **Hallucination**

"Context Rot" degrades output quality rapidly.

The Solution (Action)

Shift from "One-Shot Prompting" to "Agentic Workflows".

Adopt the architecture:



>_ _ready for implementation

パラダイムシフト：チャットからアーキテクチャへ

The Old Way (Amateur)

- **Giant Context Window** (Single global context)
- **Manual Copy-Paste**
- **“Slot Machine Method”**
  (Save → Run → Revert → Retry)
- Result: Unstable, "Vibe Coding”

The New Way (Professional)

- **Progressive Disclosure**
  (必要な時に必要な情報だけ / Only necessary info)
- **The Holy Trinity**
  (Skills / Agents / Hooks)
- **Deterministic Execution**
- Result: Reliable Engineering

>_ _ready for implementation

アーキテクチャの核心「The Holy Trinity」

Skills (The Knowledge)

Reusable domain knowledge loaded on-demand.

Path: `.claude/skills/`
Invoke: `/skill-name`

Agents (The Workforce)

Isolated execution contexts for specific tasks.

Separates "Planning" from "Implementation".

Hooks (The Guardrails)

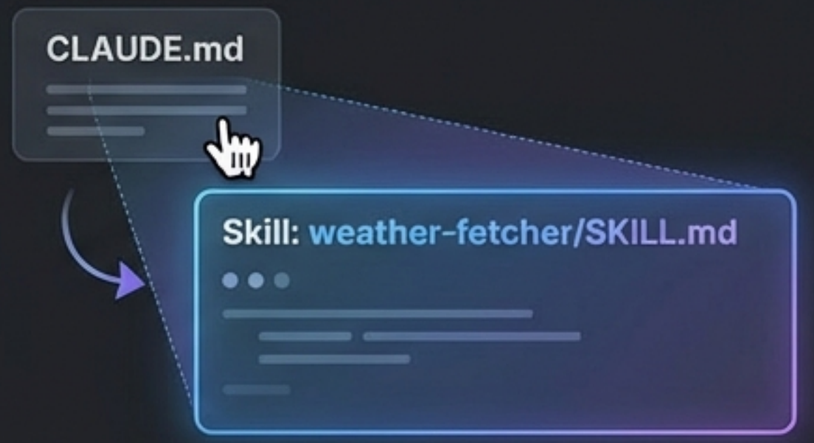
Deterministic scripts running on specific events.

Ensures rules are enforced (e.g., pre-commit checks).

> __ready for implementation

Skills: "Progressive Disclosure"の実現

```
.claude/
├── CLAUDE.md (Keep small!)
└── skills/
    ├── weather-fetcher/
    │   └── SKILL.md
    ├── gmo-payment/
    │   └── SKILL.md
    └── azure-infra/
        └── SKILL.md
```



The Core Concept
Don't stuff everything in `CLAUDE.md`.
Loading all docs at once causes **"Context Rot"**

Best Practice
1. Create domain-specific Markdown files.
2. Load them *only* when the task requires it via **Custom Slash Commands**.
3. Keep the main context clean.

> __ready for implementation

Agents: 実行コンテキストの分離

Definition: Specialized sub-processes for distinct tasks (Backend, Testing, Security).

CRITICAL WARNING (from AGENTS.md)

❌ **Don't**: Ask Claude to run `claude task` via bash. (It will fail).

✅ **Do**: Use the programmatic **"Task tool"** for orchestration.

Separate "Planning" from "Implementation" to drastically reduce error rates.

> __ready for implementation

Memory & Hooks: 運用を支えるガードレール

The 150-Line Rule

`CLAUDE.md` **must be < 150 lines.**

- ***Insight:*** "If your `CLAUDE.md` is too long, Claude ignores half of it."
- ***Solution:*** Move details to Skills. Keep `CLAUDE.md` for high-level directives only.

Hooks (Deterministic Rules)

Use for non-negotiable checks.

Hook Examples

Example 1: PreToolUse

```
"Prevent committing .env files"
```

Example 2: PostToolUse

```
"Auto-run linters/formatters after
code generation"
```

❯ __ready for implementation

運用における「黄金のルール」(The Golden Rules)

****The 50% Rule****

If context usage > 50%:

Run `/compact` manually.

Monitor usage aggressively.

****Task Sizing****

A task must fit within **50% of the context window**.

"Vanilla Claude Code with small tasks > Complex workflows."

****Atomic Commits****

Commit immediately after task completion.

Never drag context across distinct tasks.

****Plan Mode****

Always start with:

`/plan`

Do not code without a plan.

```
› __ready for implementation
```

推奨ワークフロー：RPIモデル

R - Research (調査)

Gather context.
Read files.
Understand the codebase.
Don't guess.

P - Plan (計画)

Define the path.
Always start with `/plan` mode.
Define the architecture before writing code.

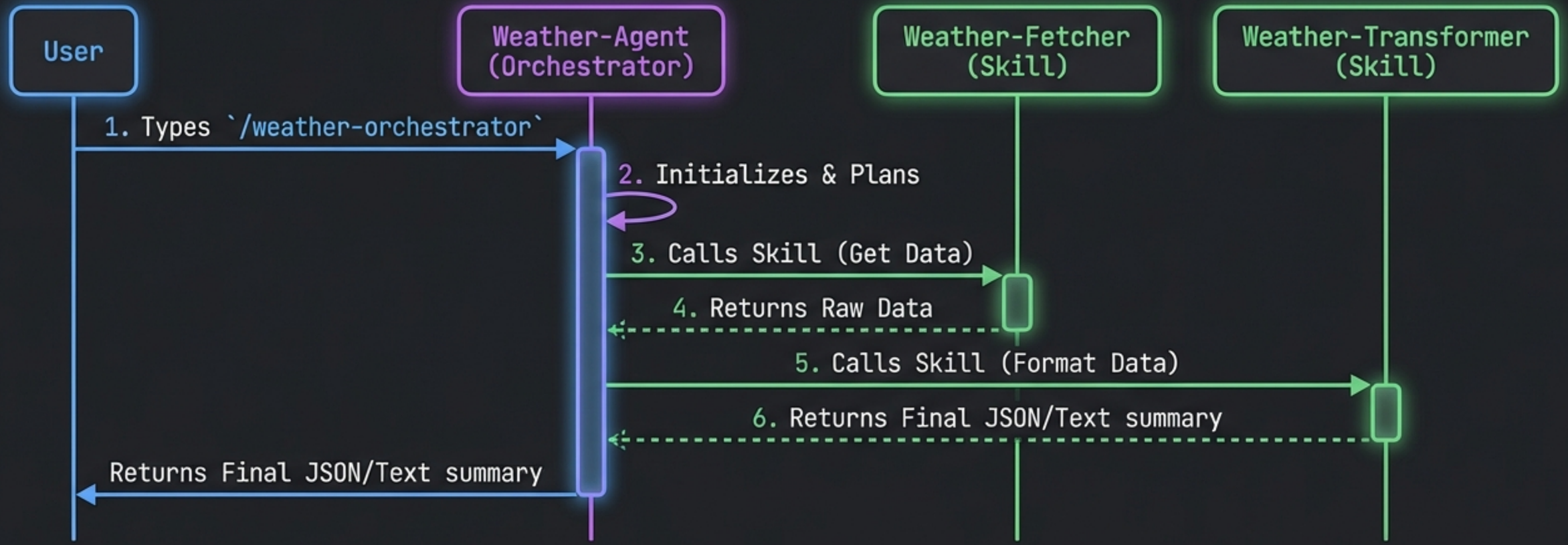
I - Implement (実装)

Execute the code changes.
Builder Mode.

Reference: "Human Layer RPI" / "Boris Feb26 workflow" — Prevents "Vibe Coding".

```
› __ready for implementation
```

実装例：Weather Orchestration



Key Takeaway

Clean separation of concerns. The **Agent** orchestrates, **Skills** provide logic, **User** provides intent.

> ___ready for implementation

開発環境とツールセット (Environment & Toolset)

Terminal: iTerm2

Recommended for MacOS.

Avoid: VS Code integrated terminal (Known crash issues with Claude Code).

Productivity: Wispr Flow

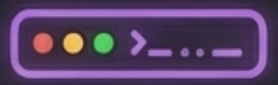
Voice prompting tool.

**Claim:* 10x productivity via high-speed voice input.*

Configuration

Mode: `bypassPermissions` (or `/config` "don't ask permission")

Reduces friction in trusted environments.



Status Line

Enable for constant context awareness.

> ___ready for implementation

トラブルシューティングとデバッグ

```
> /doctor
```

Diagnosis

Run `/doctor` command.

Performs immediate environment health checks.

Visibility

Screenshots: Provide images to Claude when UI issues occur.

Background Tasks: Ask Claude to run servers/terminals in background so it can read the logs itself.

Browser Testing

Use **MCP** (Model Context Protocol).

Claude in **Chrome** / Playwright.

Let Claude see the Chrome console logs directly.

> ___ready for implementat

高度なトピックとレポート (Advanced Topics)

1.2 KB 2 minutes ago

****CLI vs Agent SDK.md****

Deep dive into why outputs differ based on System Prompt architecture.

1 line

1.1 kB 2 minutes ago

****Monorepo Strategy.rpt****

How to handle `CLAUDE.md` loading: *Ancestor vs Descendant behavior.*

1 lir

6 KB 2 minutes ago

****Browser Automation.test****

Comparison Report: Playwright vs Chrome DevTools vs Claude in Chrome.

> ___ready for implementation

アクションプラン：明日からの変革

1. Immediate Cleanup

- Trim `CLAUDE.md` to < 150 lines.
- Extract excess rules into `.claude/skills/`.

2. Adopt RPI

- Research → Plan → Implement.
- Always start every session with `/plan`.

3. Monitor & Commit

- Watch the context gauge.
- Run `/compact` at 50%.
- Atomic commits after every task.

結論 (Conclusion)

「Claudeが仕事をしやすい状態にすることは、
人間も仕事をしやすくなること」

(Making it easier for Claude to work makes it easier for humans to work.)

`Small Context` + `RPI Workflow` + `Frequent Commits` = **High Quality Code**

`github.com/shanraisshan/claude-code-best-practice`