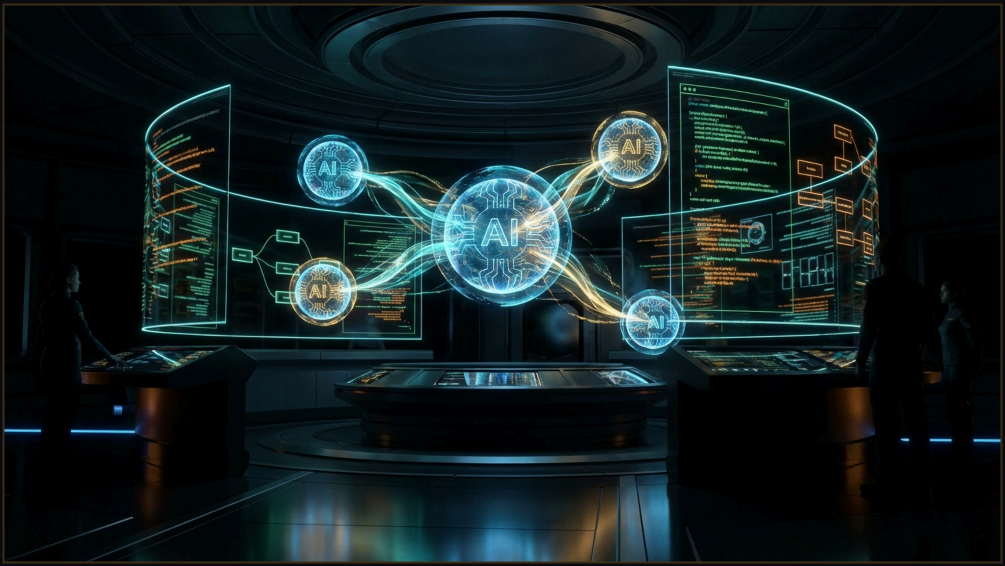
ADVANCED GUIDE

The AI Project Orchestrator Blueprint

Stop prompt engineering messages.
Start prompt engineering your entire workflow.



By Luis Aviles // Operations Architect

INTRODUCTION

The Problem With Prompt Engineering

Everyone's obsessed with prompt engineering.

"Write better prompts." "Use chain-of-thought." "Add system instructions."

Great. That helps with your first message. **What about the next 100?**

Real projects aren't one-shot. They're hundreds of back-and-forths across multiple sessions. Multiple agents. Multiple days. You can't prompt engineer every exchange.

> *"I stopped prompt engineering my messages. I started prompt engineering my workflow. The instructions. The stages. The checkpoints. Baked into the system. So I don't have to remember."*

This guide shows you how to build that system.

SECTION 01

The CLAUDE.md File: Your Project's Brain

Every project I work on has a single file that controls everything: CLAUDE.md

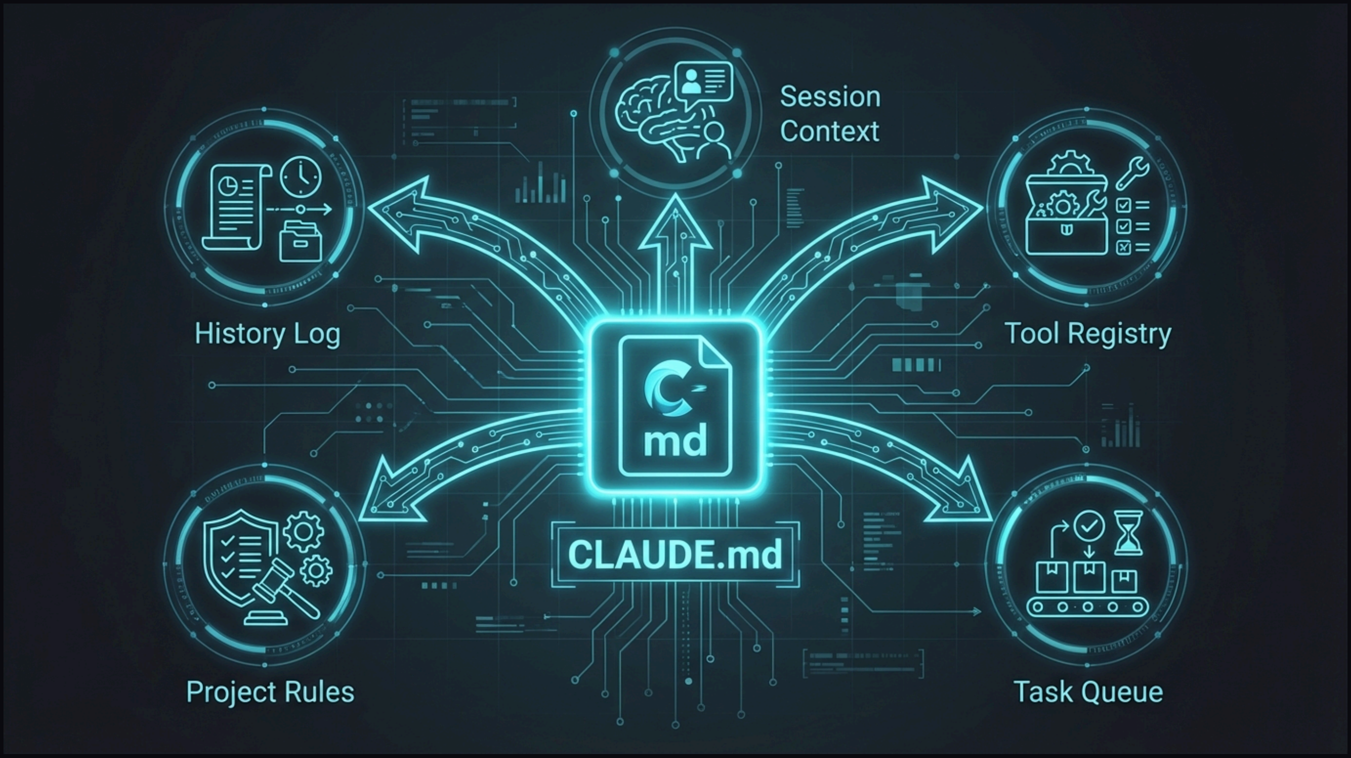
This isn't documentation. It's **operational memory**. When I start a new session, Claude reads this file and knows:

- **What this project is** - purpose, goals, current state
- **What's been done** - completed work, decisions made
- **What's in progress** - active tasks, blockers
- **What tools exist** - Python scripts, agents, skills
- **How to behave** - project-specific rules and patterns

Example CLAUDE.md Structure

```
# Project: Business Operations System # Status: Active Development # Last Session: 2026-02-03
## What This Project Does AI-powered operations infrastructure for service businesses.
Financial dashboards, CRM automation, SOP management. ## Current State - [x] Financial
ingestion pipeline (500+ rules) - [x] CRM scraper (saves 130+ hrs/year) - [ ] Owner dashboard
v2 - [ ] Mobile notifications ## Available Tools | Tool | Location | Purpose | |------|-------
---|---------| | refresh_data.py | /tools/ | Pulls latest CRM data | | classify.py | /tools/ |
Categorizes transactions | ## Session Rules - Always run security review before marking
complete - Use /parallel for independent tasks - Python for mechanical ops, AI for
intelligence ops
```

This file is the difference between "starting fresh every session" and "picking up exactly where you left off."



SECTION 02

The 9-Stage Pipeline

Every project runs through the same stages. Not because I remember them - because they're encoded in the system.

01 **/analyze**
Understand requirements. Explore codebase. Map dependencies. No coding yet.

02 **/plan**
Design the approach. Break into tasks. Identify parallel opportunities.

03 **/parallel**
Launch independent tasks simultaneously. Multiple agents working together.

04 **/implement**
Build the solution. Follow the plan. Track progress.

05 **/security**
Review for vulnerabilities. Check API keys. Validate inputs.

06 **/test**
Run tests. Verify edge cases. Confirm requirements met.

07 **/cleanup**
Remove dead code. Organize files. Update documentation.

08 **/review**
Final check. Does it match the original request? Any gaps?

09 **/complete**
Update CLAUDE.md. Archive session. Ready for next time.

Average user skips steps 5-8. They get excited when "it works" and move on. The pipeline ensures nothing gets skipped.

SECTION 03

Skills + Agents + Python: The Toolkit

AI alone isn't always the answer. Sometimes a Python script is faster. Sometimes an agent is smarter. The key is knowing when to use what.

CORE PRINCIPLE

Mechanical Ops → Python
Intelligence Ops → LLM

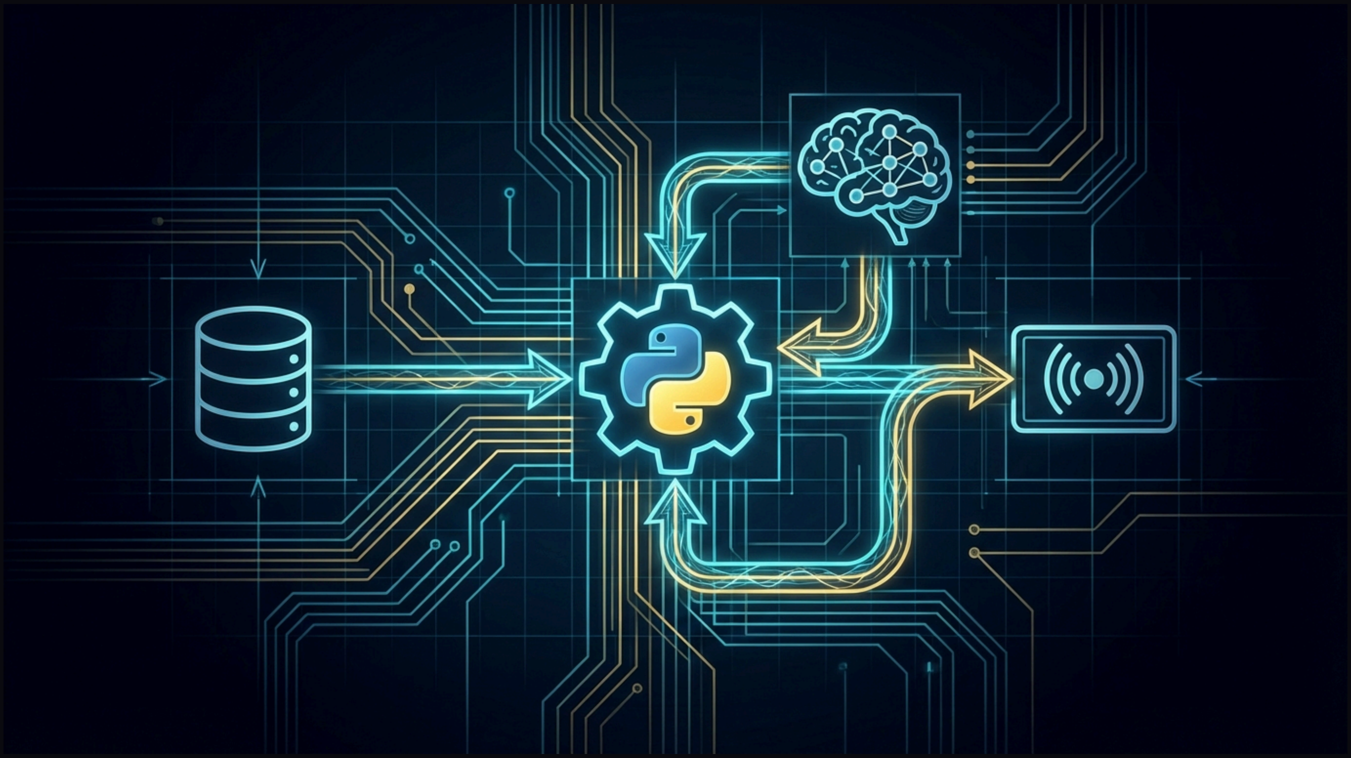
When to Use Python

- **File operations** - Moving, renaming, organizing
- **Data transformation** - Parsing, formatting, calculations
- **Batch processing** - Same operation on many items
- **Scheduled tasks** - Things that run without you
- **State management** - Tracking progress, updating files

When to Use AI

- **Contextual reasoning** - Understanding nuance
- **Edge cases** - Handling the unexpected
- **Content generation** - Writing, summarizing
- **Decision making** - Weighing options
- **Exploration** - Understanding new codebases

The Hybrid Pipeline



Real example: CRM data extraction - Selenium scrapes at machine speed, Python flags discrepancies, Claude reviews only the 5% that need judgment. **45 minutes manual → 5 minutes automated.**

SECTION 04

Session Persistence: Never Start Fresh

The biggest waste in AI workflows: losing context between sessions.

Without Persistence

- Re-explain project every session
- AI forgets decisions made
- Duplicate work happens
- "Where were we?" syndrome

With Persistence

- AI reads CLAUDE.md, knows everything
- Decisions are documented
- Progress tracked automatically
- Pick up exactly where you stopped

The Persistence Stack

1. **CLAUDE.md** - Project-level memory (what is this)
2. **SESSIONS.lock** - Who's working on what (prevents conflicts)
3. **WORK_QUEUE.md** - Tasks waiting to be done
4. **PIPELINE_STATE.md** - Current stage and progress

Together, these files create **institutional memory** for your AI workflow. The system remembers so you don't have to.

SECTION 05

Putting It All Together

Here's what a typical workflow looks like:

```
// Day 1: Start new feature > /analyze "Add user authentication" // AI explores codebase, identifies
files, maps dependencies > /plan // AI creates task list, identifies what can run in parallel // Day
2: Continue work (AI reads CLAUDE.md, knows where we left off) > /implement // Continues from saved
state, no re-explanation needed > /complete // Updates CLAUDE.md, archives session, ready for
production
```

THE MINDSET SHIFT

You're not prompting an AI.
You're orchestrating a team.

INCLUDED IN YOUR DOWNLOAD

- **CLAUDE.md template** - Ready to customize for your project
- **State files** - WORK_QUEUE.md, PIPELINE_STATE.md, SESSIONS.lock
- **Python tools** - new_project.py, update_state.py, parse_plan.py
- **Example configuration** - See how a real project is set up
- **Alternatives guide** - For non-Claude Code users

See the **/template** folder in this download package.

NEXT STEPS

Start Building Your Orchestrator

1. **Open the template folder** - Copy it to your project
2. **Customize CLAUDE.md** - Fill in the PROJECT-SPECIFIC section
3. **Try one pipeline run** - /analyze → /plan → /implement on a small task
4. **Add Python where needed** - Identify mechanical operations to script

This isn't about using AI more. It's about using AI *systematically*.

The most efficient AI workflow isn't always more AI. It's having AI build durable automation you own, and deploying it strategically.

Created by Luis Aviles // Operations Architect

From Spreadsheets to Systems