AGENT ENGINEERING

Building Custom Agents

Design, build, and deploy specialized AI agents that integrate with the ACOS ecosystem.

v7.0 — by Frank Guzman — frankx.ai

01 Anatomy of an Agent

An agent is a **markdown file** in `.claude/agents/` that defines a specialized persona for Claude Code's Task tool.

Agent Definition

```
# .claude/agents/content-optimizer.md
name: Content Optimizer
role: Content quality and SEO specialist
## Core Capabilities
- SEO analysis (keyword density, meta tags)
- Readability scoring (Flesch-Kincaid)
- Voice consistency checking
## Output Format
Return structured report: score, improvements, suggestions
```

Required Fields

- **name** — Display name
- **role** — One-line description
- **capabilities** — What it can do

Recommended Fields

- **tools** — Which tools it uses
- **output format** — Expected deliverable
- **triggers** — Activation keywords

02 Agent Design Patterns

Pattern 1: Domain Expert

Single-domain agent with deep knowledge. Best for specific, repeatable tasks.

Examples: SEO Auditor, Music Producer, Security Reviewer

Pattern 2: Pipeline Agent

Multi-step agent that orchestrates a workflow. Takes input, processes through stages.

Examples: Factory, Product Launch, Publishing Pipeline

Pattern 3: Reviewer Agent

Analyzes existing work and provides feedback. Read-only, returns structured reports.

Examples: Code Reviewer, Content Reviewer, Accessibility Auditor

Pattern 4: Coordinator Agent

Meta-agent that manages other agents. For complex tasks requiring multiple specialties.

Examples: Starlight Orchestrator, Council, Master Story Architect

BEST PRACTICE

Start simple. Build a single-purpose agent first. Get it working well. Then add complexity incrementally.

03 Building Your First Agent

Step-by-step: a **Blog Post Optimizer** agent.

1. Create the file

```
$ touch .claude/agents/blog-optimizer.md
```

2. Write the definition

```
# Blog Post Optimizer
You are the Blog Post Optimizer. Analyze MDX
blog posts and provide actionable improvements.
## Process
1. Read the target MDX file
2. Analyze: word count, headings, keywords
3. Check: meta, schema, internal links
4. Score: readability, engagement hooks
5. Report: structured findings
## Output
- Overall score (1-100)
- Top 5 high-impact improvements
- Keyword opportunities
```

3. Register in skill-rules.json (optional)

```
# Add to "agents" section:
"blog-optimizer": {
  "promptTriggers": {
    "keywords": ["optimize blog", "improve post"]
  }
}
```

4. Use it

The agent activates when triggers match, or invoke directly by describing what you need.

04 Advanced Techniques

Multi-Agent Composition

Create coordinator agents that delegate to specialists. Synthesize findings from multiple domain agents into a single action plan.

Context-Aware Agents

Agents that detect CLAUDE.md, package.json, or .frankx/ to adapt behavior to the current project's conventions and tech stack.

Self-Improving Agents

Use `/agentic-jujutsu` to record input/approach/output/feedback after each task. Extract patterns and update agent definitions with successful strategies.

MCP Integration

Reference MCP servers in agent definitions: Playwright for visual testing, Nano Banana for image generation, Memory for cross-session persistence.

Validation Checklist

- Consistent output format across invocations
- Handles edge cases (empty files, missing data)
- Respects read-only vs write permissions
- Integrates with existing workflows
- Clear enough for Claude to follow reliably

Need help building agents?

hello@frankx.ai

FRANKX.AI · GITHUB.COM/FRANKXAI