

Claude Code Skills: 지능의 확장과 표준화

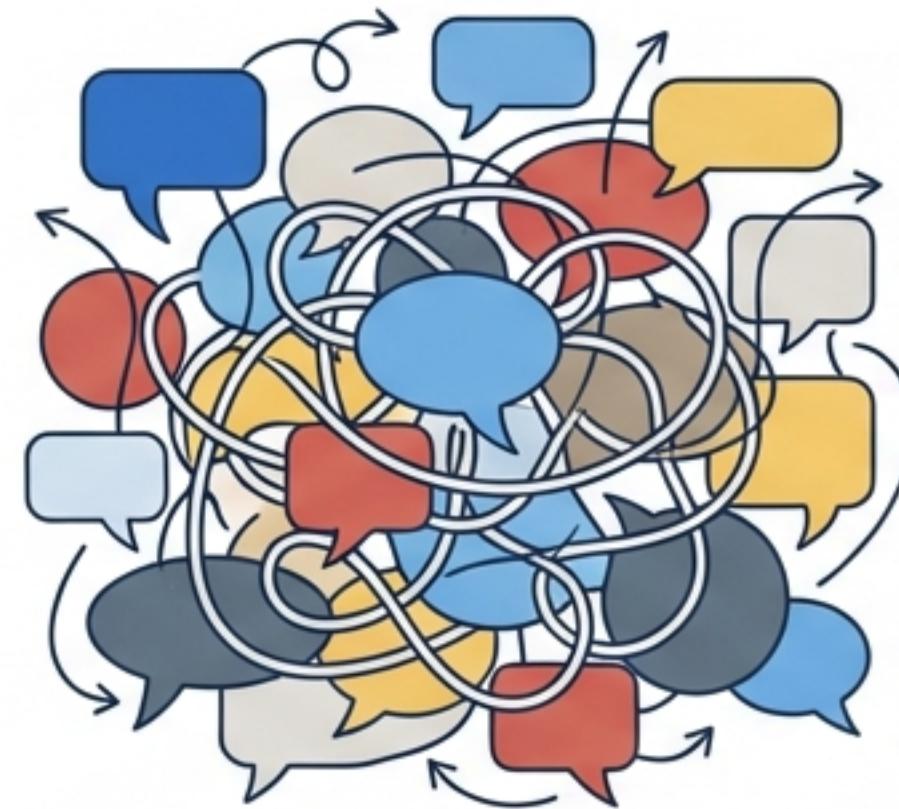
단순 프롬프트를 넘어, 재현 가능한 조직의 표준 운영 절차(SOP)로



STRATEGIC TECHNICAL PLAYBOOK

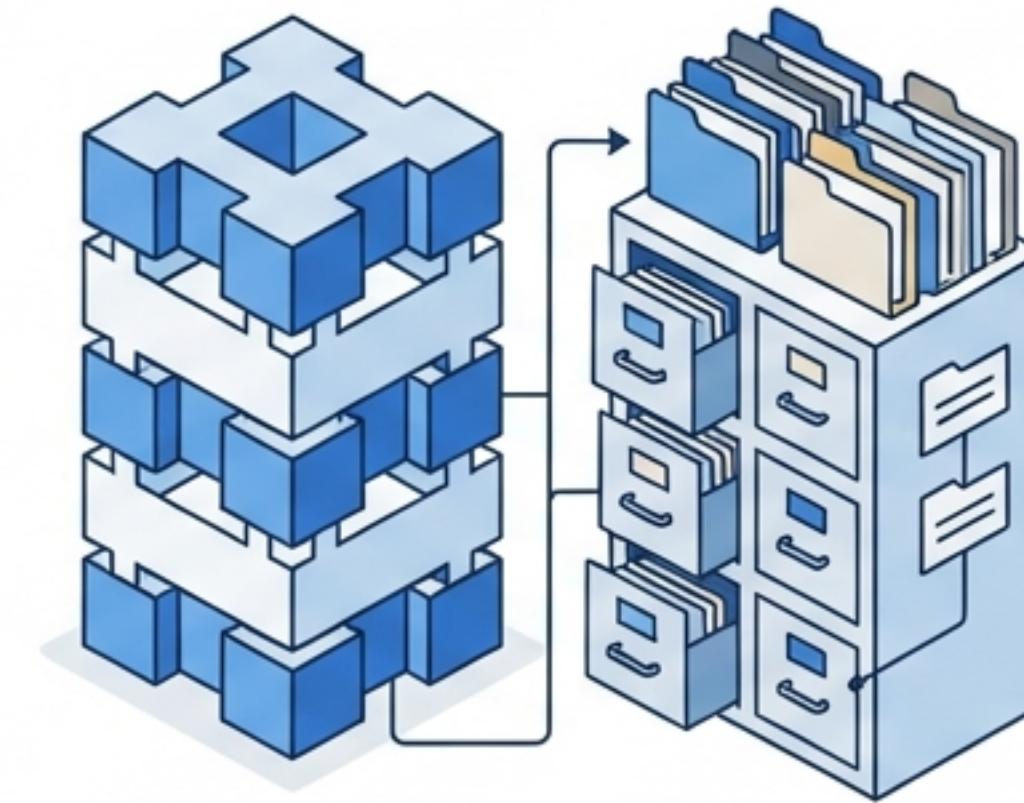
Contrast

BEFORE: Prompting



- 일회성 (One-off)
- 가변적 결과 (Variable)
- 개인의 노하우 (Tribal Knowledge)

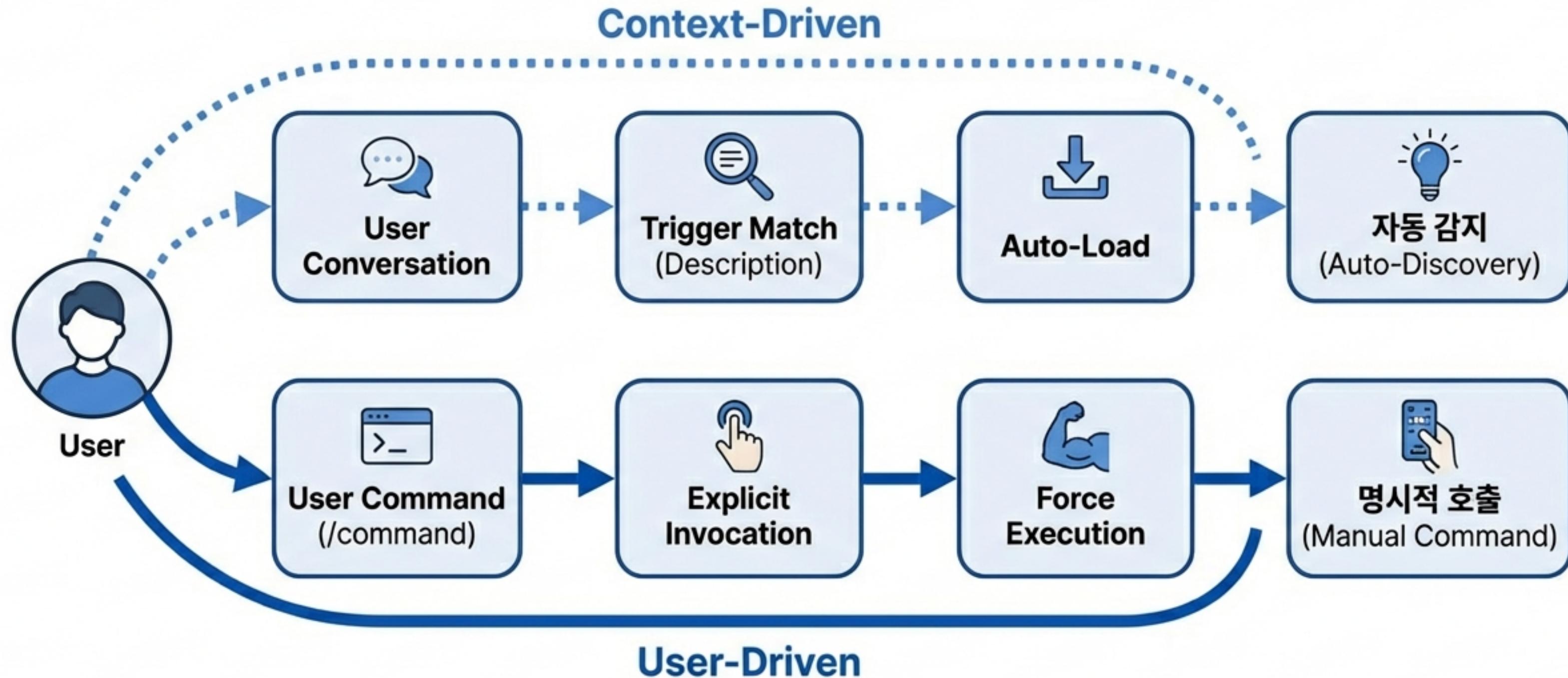
AFTER: Engineering Skills



- 반복 가능 (Reproducible)
- 고정된 품질 (Standardized)
- 팀 자산 (Organizational Asset)

Engineering Blue Skill의 정의: Claude가 특정 작업을 수행할 때 반드시 참조해야 하는 지식, 절차, 품질 기준을 패키지화한 것입니다.

이중 실행 메커니즘 (Dual Mechanism)



스킬은 필요할 때 스스로 개입하거나, 사용자의 의도에 따라 도구처럼 사용될 수 있습니다.

The Brain: SKILL.md 구조

제어 장치(Control Panel):
언제, 어떻게 실행될지 결정
(설정, 권한)

실행 지침
(Instruction Manual):
작업 절차, 규칙,
출력 형식 정의

SKILL.md

```
---
```

name: explain-code
description: Explains code logic
disable-model-invocation: false

Instructions

1. Analyze the input code.
2. Create a flow diagram.
3. Explain step-by-step.

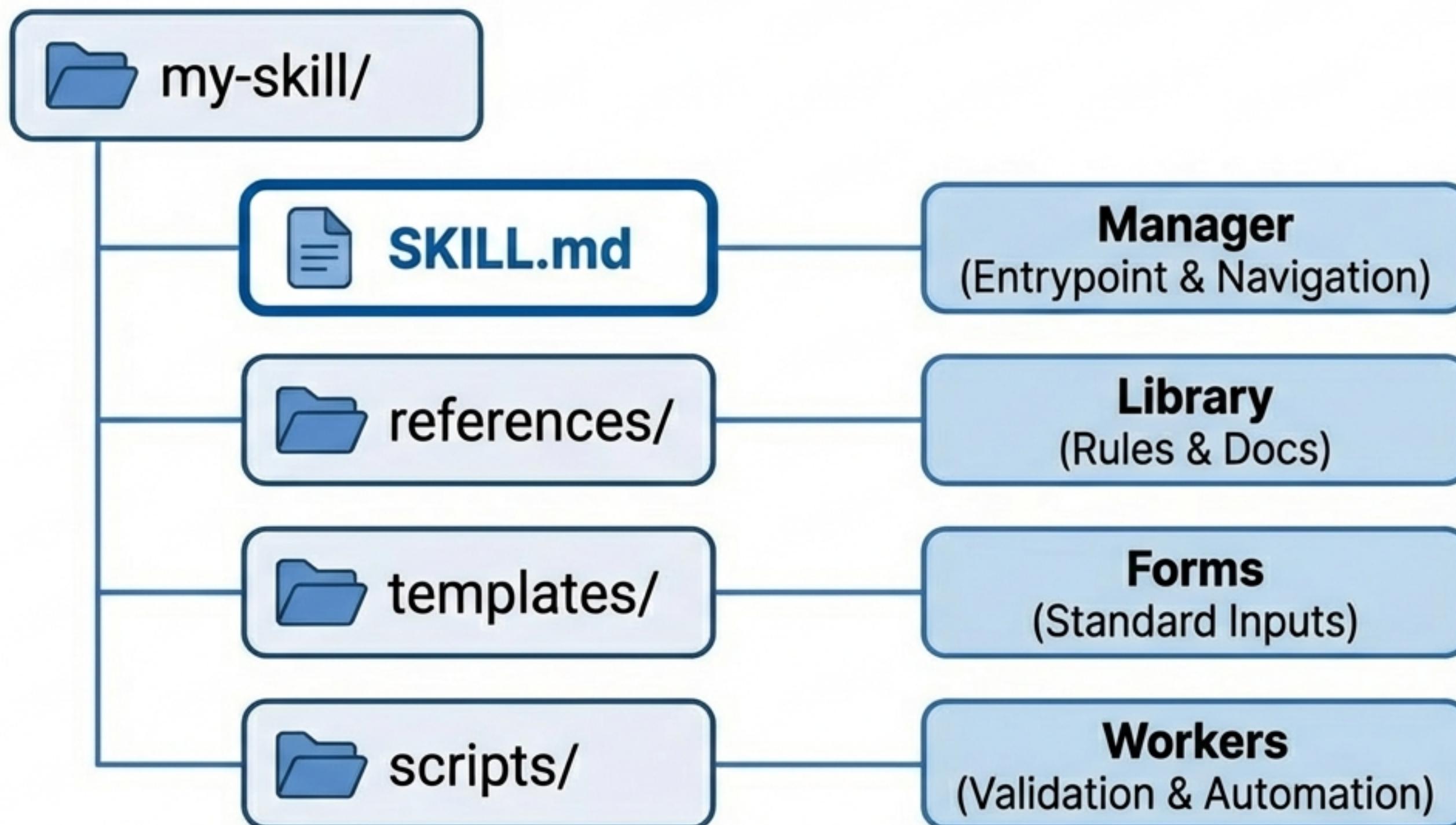
30%

70%

모든 스킬의
진입점(Entry point)은
SKILL.md입니다.

The Ecosystem: 디렉토리 구조

SKILL.md는 관리자(Manager)이며, 세부 작업은 위임합니다.



Scope & Context: 스킬의 저장 위치



Enterprise

Managed Settings
조직 전체 표준 강제

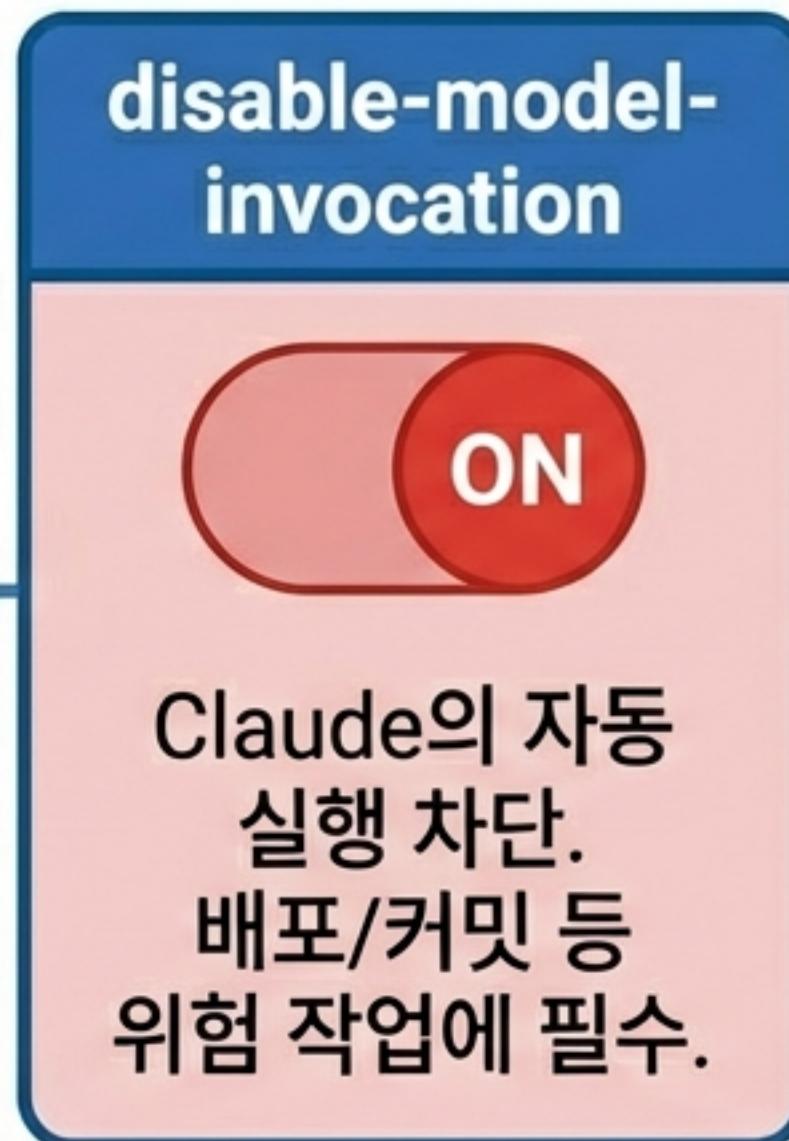
Personal

~/.claude/skills/
개인의 모든 프로젝트

Side Note (Key Rule):

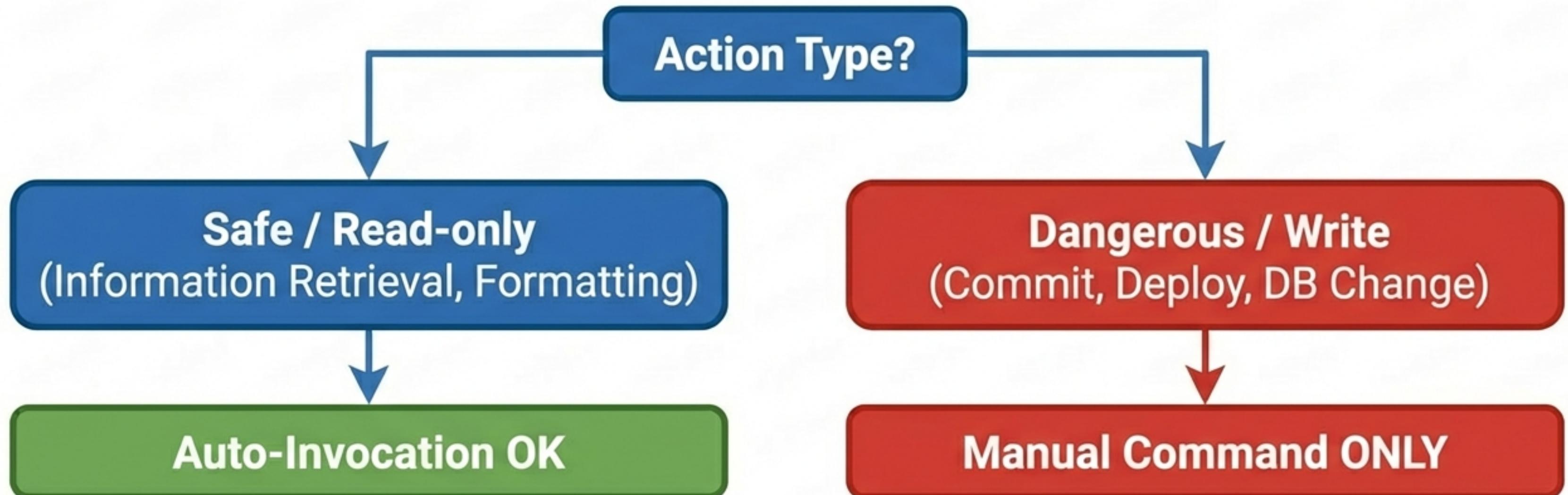
우선순위 (Priority Rule):
Enterprise > Personal > Project

Controlling Behavior: Frontmatter 설정



호출 제어 정책 (Invocation Policy)

자동화와 인간 통제의 경계 설정



부작용(Side Effect)이 있는 작업은 반드시 인간의 명시적 승인을 거치도록 설계합니다.

Two Archetypes: Reference vs. Task



Reference Archetype (Knowing)

- Focus: 지식, 규칙, 가이드라인
- Example: 코딩 컨벤션, UI 스타일 가이드
- Use Case: Inline Knowledge

Claude에게 '무엇을 아는지'를 제공하거나, '무엇을 해야 하는지'를 지시합니다.

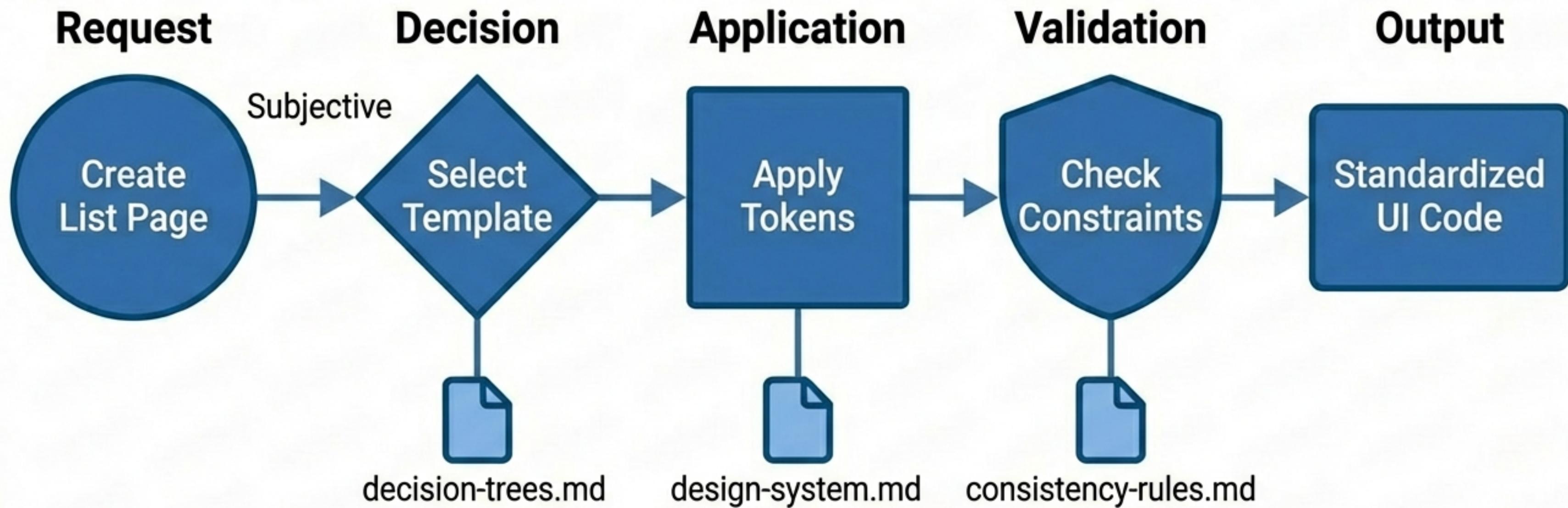


Task Archetype (Doing)

- Focus: 절차, 행동, 파이프라인
- Example: 배포, 마이그레이션, OCR 처리
- Use Case: Step-by-step Actions

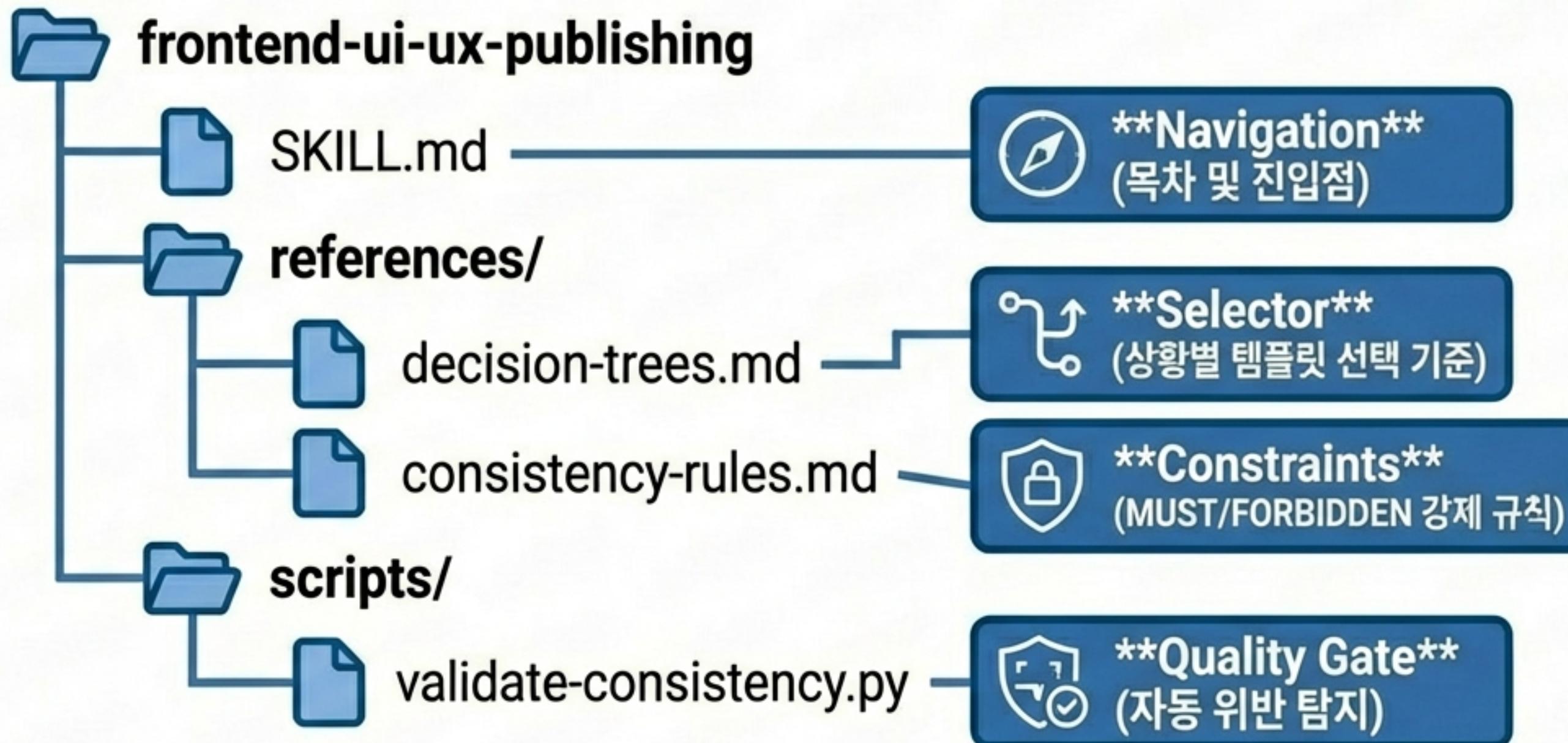
Case Study A: Frontend System

주관적 디자인의 객관화 (Objectifying Subjective Design)



의사결정, 규칙, 검증을 분리하여 일관성 확보.

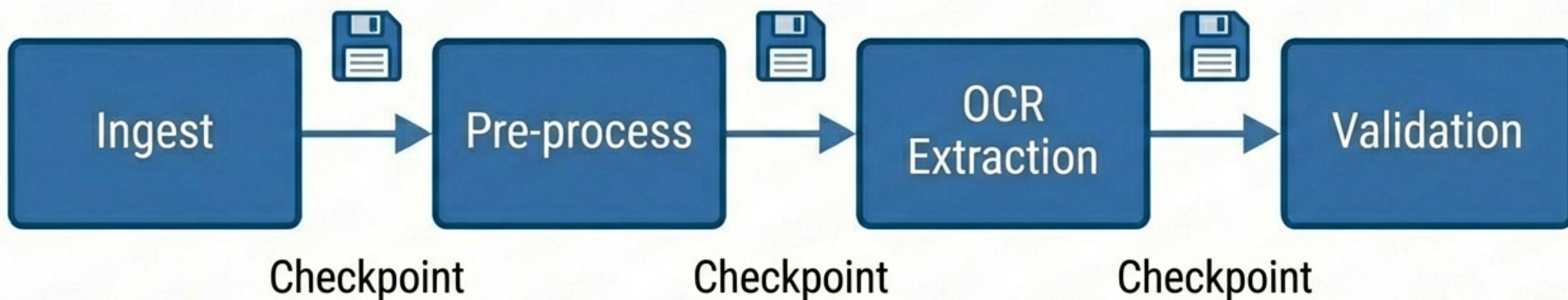
Case Study A: The File Strategy



분리의 미학: `references`는 역할별 표준을 쪼개 관리하며,
스크립트는 사람이 놓치기 쉬운 규칙 위반을 사전에 차단합니다.

Case Study B: OCR Pipeline

실패를 전제로 한 운영형 설계 (Fail-Safe Engineering)



- **Stage Separation:** 원인 파악을 위해 프로세스 분리
- **Checkpoints:** 각 단계 완료 시 상태 저장 (재시작 비용 최소화)
- **Contract:** `references/stages.md`에 정의된 입출력 스키마 준수

Engineering Rigor: 검증(Verification)과 복구(Recovery)

Error Handling Strategy

오류 발생 시: Fallback → Retry → Alert



Verification Checklist (Quality Gate)

- Chunk Count Match
- Embedding Dimensions Valid
- JSON Schema Validated

Fail-Safe: 성공을 기대하는 것이 아니라,
실패를 감지하고 복구하도록 설계합니다.

스킬 설계를 위한 디자이너 체크리스트

코드를 작성하기 전, 다음 5가지 질문에 답하십시오.

- 1 Problem Definition:** 이 스킬이 해결하려는 구체적인 문제는 무엇인가?
- 2 Trigger Design:** 사용자는 어떤 키워드나 표현으로 이것을 요청하는가?
- 3 Success Criteria:** 성공적인 결과물의 형태와 품질 기준은 무엇인가?
- 4 Failure & Recovery:** 실패할 수 있는 지점은 어디이며, 어떻게 복구할 것인가?
- 5 Control Policy:** 자동 실행이 안전한가, 아니면 수동 승인이 필요한가?

결론: ‘마법’을 ‘메서드(Method)’로 전환하십시오.



‘마법’
(Magic / Tribal Knowledge)

‘메서드’
(Method / System)

Skill은 조직 최고의 엔지니어의 두뇌를 복제하여 팀 전체에 공유하는 기술입니다.
일회성 대화에서 벗어나, 검증되고 재현 가능한 시스템을 구축하십시오.