CHAPTER 04 / PART 01

Settings 체계

스코프와 우선순위, 그리고 로드 시점

.claude 디렉토리 — 설정의 위치 지도
스코프 4계층과 우선순위 5단
병합 규칙 — 재정의 · 병합 · 비병합 예외
파싱 · 라이브 리로드 · 진단

설정이 어긋나는 상황

쓰다 보면 반드시 만나는 네 장면

FOUR SCENES

다들 한 번쯤 겪어본 상황들입니다. 증상은 제각각이지만 전부 같은 질문으로 수렴합니다.

장면 1

"세션을 열 때마다 /model부터 치는 게 습관이 됐다"

장면 2

"온보딩한 동료에게 권한 설정을 말로 전수하고 있다"

장면 3

"실험하려고 바꾼 값이 커밋에 딸려 들어갔다"

장면 4

"보안팀: 이 명령은 아무도 못 쓰게, 개인이 되돌릴 수 없게 해 주세요"

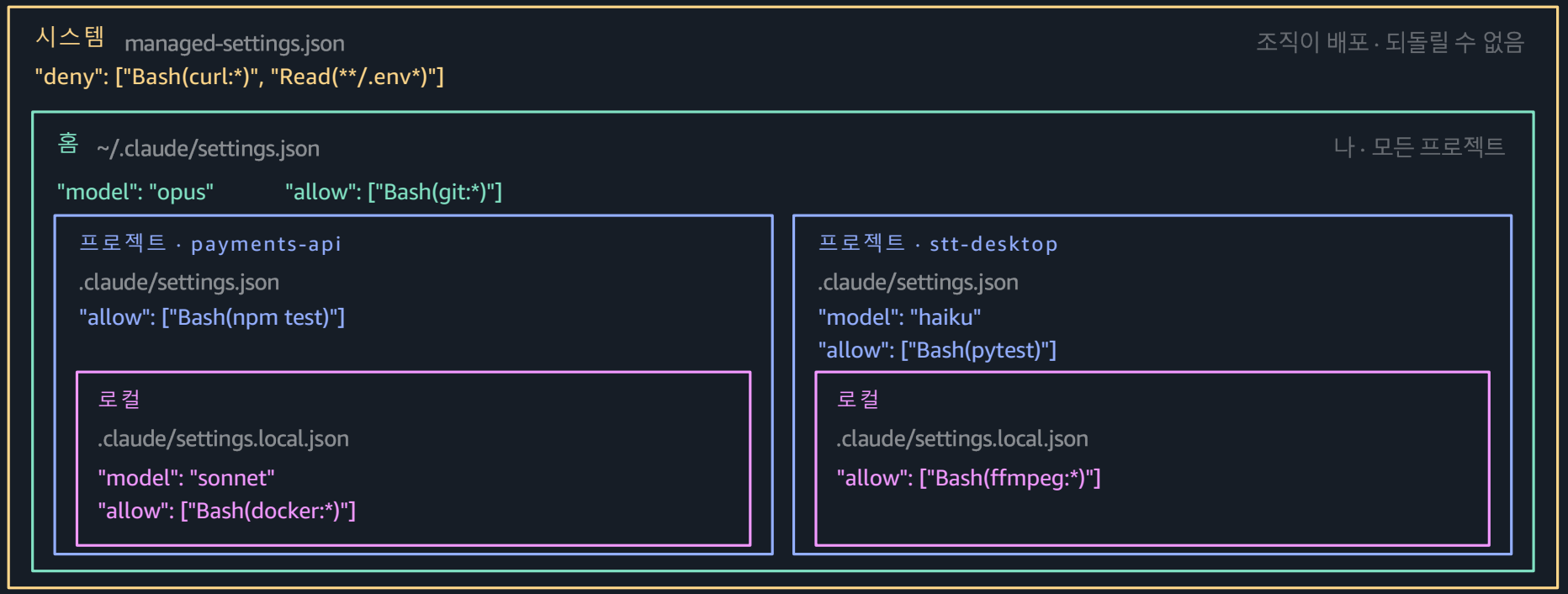
반복 수동 설정
어딘가 적어두면 될 텐데

구전 온보딩
파일도 문서도 아닌 전수

실험값 유출
커밋에 섞여 들어감

강제 정책
개인이 못 되돌리게

설정 스코프와 우선순위



.claude 디렉토리 구조

홈은 유저에게, 저장소는 코드베이스에 귀속

your-project/ + ~/.claude/

HOME — 전역 (모든 프로젝트)

```
~/.claude/                  # 개인 기본값
  └─ settings.json  CLAUDE.md  rules/  skills/  agents/  plugins/
~/.claude.json              # (디렉토리 밖 파일) 앱 상태 · 신뢰 승인 기록
```

REPO — 저장소 (이 프로젝트만)

```
your-project/
  ├─ CLAUDE.md              # 매 세션 로드 지침 (.local.md는 개인)
  ├─ .mcp.json              # 팀 공유 MCP 서버 (루트에 위치)
  └─ .claude/
     ├─ settings.json       # 권한, 훅, env — 이 세션의 대상 (.local.json은 개인 · REPO 전용)
     ├─ rules/              # 경로 조건부 지침 조각 (paths:)
     ├─ skills/  commands/  agents/  workflows/
     └─ output-styles/  agent-memory/
```

귀속 대상
유저 vs 코드베이스

루트 파일
CLAUDE.md, .mcp.json 등

.local 규약
gitignore 개인 오버라이드

신뢰 경계
Repo 설정은 승인 후 발효

설정 스코프 4계층

어디에 두면 누구에게 적용되는가

Managed	서버 관리 설정 · plist/레지스트리 · managed-settings.json	조직 전체 — IT가 배포·강제
User	~/.claude/settings.json	나의 전 프로젝트 — 비공유
Project	저장소의 .claude/settings.json	저장소 협업자 전원 — 커밋 공유
Local	.claude/settings.local.json	이 저장소의 나만 — gitignore

취향은 User
전 프로젝트 개인 기본값

팀 표준은 Project
커밋해서 공유

실험은 Local
커밋에 안 섞이게

강제는 Managed
개인이 못 되돌림

우선순위 5단과 병합

같은 키 충돌 — model은 재정의, permissions는 병합

1	Managed	재정의 불가	—
2	CLI 인자	그 세션 한정	opus ✓
3	Local	개인 로컬 파일	—
4	Project	팀 공유 표준	sonnet ✕
5	User	내 기본값	haiku ✕

기본 규칙의 예외 2종

permissions — 재정의 없이 전 스코프 병합, 충돌 시 deny 우선

fallbackModel — 배열이지만 비병합, 최상위 정의 파일이 체인 전체 공급

model 키 — 세션 선택이 최우선

설정 파일의 model은 강제가 아니라 초기 선택

--model · ANTHROPIC_MODEL — 그 세션 한정

/model — 전환하며 User에 저장 (v2.1.153+)

선택 시 s 키 — 저장 없이 이 세션만 전환

단일 값 키
우선순위로 재정의

permissions
병합 + deny 우선

세션 선택
그 세션에서는 최우선

Managed
파일 우선순위 최상위

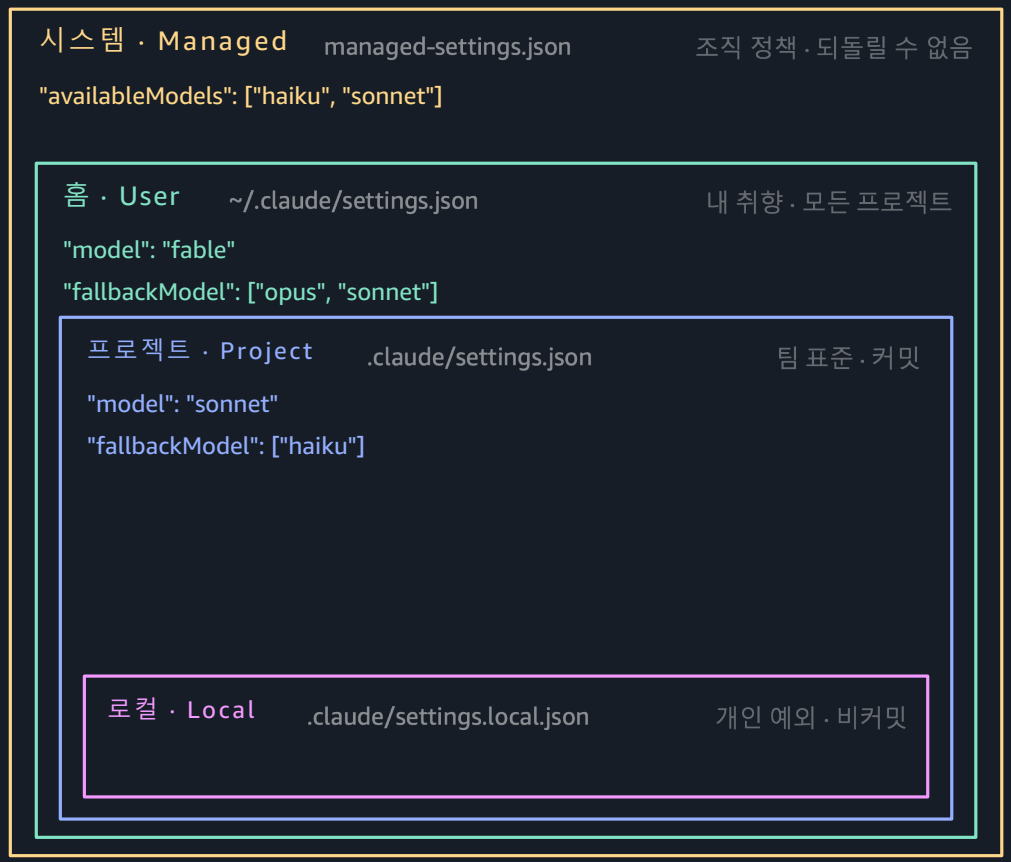
설정 스코프와 우선순위

설정 스코프와 우선순위

model = "sonnet"
단일 값 키 — Project가 User(fable)를 재정의

fable 선택 불가
비병합 — managed 목록만 (v2.1.175+)

설정 스코프와 우선순위



model = "sonnet"
단일 값 키 — Project가 User(fable)를 재정의

fable 선택 불가
비병합 — managed 목록만 (v2.1.175+)

fallbackModel = [haiku]
비병합 — User 체인 무시, 통째로 적용

설정 스코프와 우선순위

model = "sonnet"
단일 값 키 — Project가 User(fable)를 재정의

fable 선택 불가
비병합 — managed 목록만 (v2.1.175+)

fallbackModel = [haiku]
비병합 — User 체인 무시, 통째로 적용

allow = test + build + ~/tools
배열 키 — 전 스코프 연결 후 중복 제거

Bash(npm test *) = 차단
allow에 있어도 deny가 항상 승리

설정 스코프와 우선순위

model = "sonnet"
단일 값 키 — Project가 User(fable)를 재정의

fable 선택 불가
비병합 — managed 목록만 (v2.1.175+)

fallbackModel = [haiku]
비병합 — User 체인 무시, 통째로 적용

allow = test + build + ~/tools
배열 키 — 전 스코프 연결 후 중복 제거

Bash(npm test *) = 차단
allow에 있어도 deny가 항상 승리

무엇이 언제 로드되는가

세션 시작 · 온디맨드 · 이벤트 · 리로드

		COMPACT 후?
세션 시작	CLAUDE.md(전역·루트) · rules(paths 없음) · outputStyle · MCP 도구 목록 · MEMORY.md	재주입
온디맨드	하위 dir CLAUDE.md · rules(paths) · MCP 스키마 · 메모리 토픽	소실
이벤트 발화	hooks — 실행은 컨텍스트 밖, additionalContext만 컨텍스트에 진입	무관
라이브 리로드	permissions · hooks · apiKeyHelper … 예외: model, outputStyle	무관

세션 시작
루트 지침 + 메모리

온디맨드
경로 조건 · 본문 lazy

이벤트
hooks는 컨텍스트 밖

리로드 예외
model, outputStyle

settings.json 해부

이 파일 하나가 남은 세션들의 목차

.claude/settings.json

```
{
"$schema" : "https://json.schemastore.org/claude-code-settings.json",

"permissions" : {
   "allow": ["Bash(npm test *)", "Read"],
   "ask":   ["Bash(git push *)"],
   "deny":  ["Read(./.env)"],
   "defaultMode": "auto"
 },

"hooks" : {
   "PreToolUse":  [ … ],
   "PostToolUse": [ … ]
 },
 “model”: “global.anthropic.claude-fable-5[1m]”,
}
```

permissions

3-2 Permissions

규칙 문법 · 권한 모드

hooks

3-3 Hooks

수명주기 자동화 · 이벤트 30종

env - 모든 세션에 주입되는 환경 변수
model / fallbackModel – 세션 기본 모델과 대체 모델 체인
statusLine - 하단 상태줄 커스텀
outputStyle - 응답 렌더링 스타일

한 파일
남은 챕터의 목차

$schema
자동완성 · 사전 검증

MCP는 제외
.clacude.json은 별도(루트경로)

저장 즉시
리로드 반영 · 예외 2종

settings.local.json 해부

직접 만들기 전에 Claude Code가 먼저 만드는 파일

.claude/settings.local.json

```
{
  "permissions" : {
    "allow": [                  # 대화상자 승인
      "Bash(npm run lint)",
      "WebFetch(domain:github.com)"
    ]
  },
  "skillOverrides": {           # /skills 메뉴
    "deploy": "off"
  },
  "enabledPlugins": {           # 플러그인 옵트아웃
    "context7@marketplace": false
  }
}
```

permissions.allow
팝업을 통해 생성
"다시 묻지 않기" 승인의 저장소

skillOverrides
/skills가 생성
UI 조작의 기록 위치

enabledPlugins
/plugin 또는 직접
팀 표준의 개인 재정의

자동 생성
생성 시 gitignore 등록

git 루트
워크트리 공유 (v2.1.211+)

trust 면제
allow 즉시 적용

무시되는 키
autoMode 등 보안 키

/status

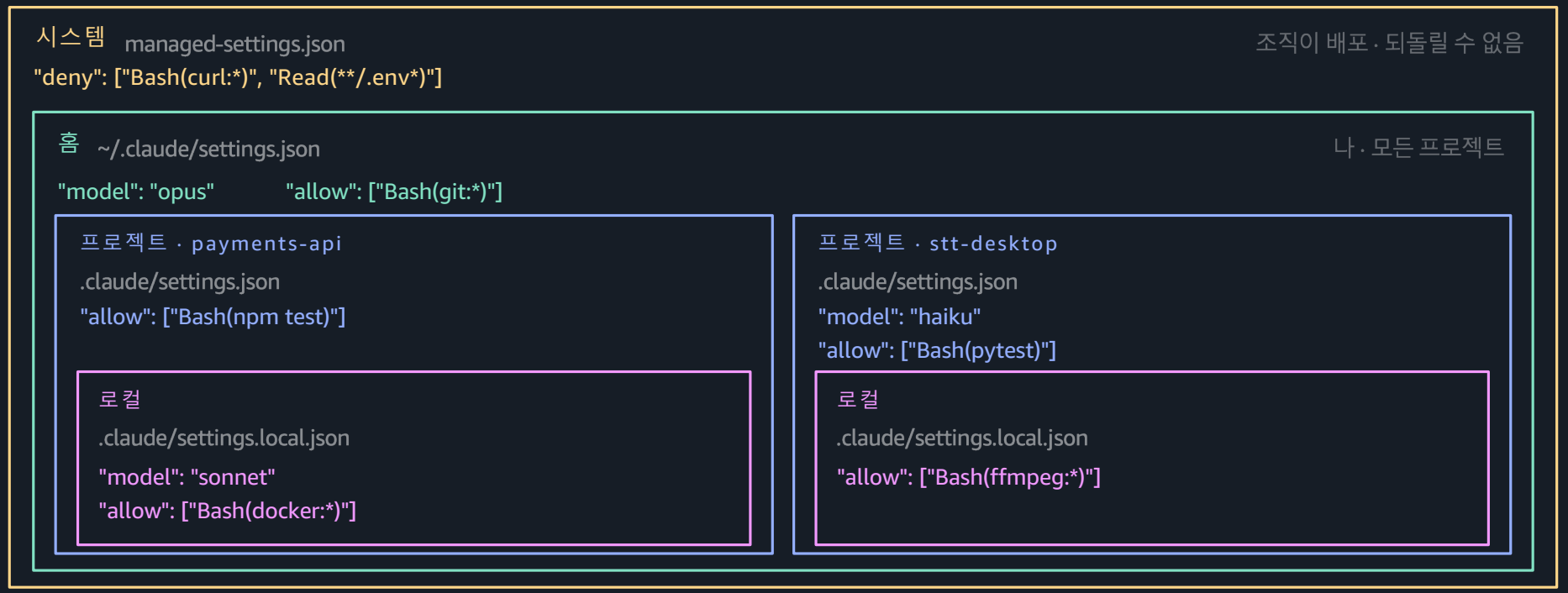
상태 확인의 대화형 창구

~/proj $ claude

```
> /status
# 탭형 상태 UI: 상태 확인과 옵션 토글
  Settings  Status  Config  Usage  Stats
  Version:          2.1.218
  Session name:     /rename to add a name
  Session ID:       5212d292-8a37-478c-8c34-f78fad5
  API provider:     Amazon Bedrock
  AWS region:       us-west-2
  Model:            global.anthropic.claude-fable-5[1m]
  MCP servers:      1 connected · /mcp
  Setting sources:  User settings, Shared project settings, Command line arguments
```

/config 항목별 저장 스코프를 함께 보여 주는 확인 창구 — key=value 직접 지정이 단일 키 변경의 최단 경로

설정 스코프와 우선순위



적용 결과

	payments-api	stt-desktop	둘 다 · 되돌릴 수 없음
model	~~opus~~ sonnet	~~opus~~ haiku	
allow	git:* + npm test + docker:*	git:* + pytest + ffmpeg:*	
deny			curl:*, **/.env*

Part 1 Recap

Settings 체계 핵심 정리

KEY TAKEAWAY

스코프 4계층, 우선순위 5단, permissions는 병합. 저장 즉시 반영되고, managed는 관용, 개인은 엄격. 이 문법 위에 Permissions와 Hooks가 올라갑니다.

핵심 정리

Managed > CLI > Local > Project > User

단일 값은 재정의 · 배열은 병합 · 비병합 예외 2종

$schema로 자동완성 · 전부 아니면 전무 예방

리로드 예외: model · outputStyle

다음 — Part 2 Permissions

규칙 문법 Tool(specifier) 한 형식

deny → ask → allow 평가 순서

권한 모드 6종과 auto 분류기

팀 표준 설계와 흔한 실수

CHAPTER 04 / PART 02

Permissions

allow · ask · deny 규칙 문법과 권한 모드

규칙의 생성 경로와 평가 순서

Bash 패턴 · 경로 패턴 · 특수 지정자

권한 모드 6종과 auto 분류기

설계 전략과 흔한 실수

권한 규칙의 생성 경로

사용 중 승인 프롬프트에서 생성

approval prompt → .claude/settings.local.json

```
> npm run test 수행해
  Bash command: npm run test · This command requires approval
  1. Yes   2. Yes, and don't ask again for: npm run *   3. No
# Bash 승인: 프로젝트+명령 단위 영구 / Edit·Write 승인: 세션 한정, 미기록
# 복합 명령은 하위 명령별 분해 저장 (최대 5)
# 그 외 입구: CLI 플래그(세션 한정) · 파일 수동 편집(라이브 리로드)
```

승인=규칙
승인이 곧 기록

Bash 영구
편집은 세션 한정

복합 분해
하위별 최대 5

출처 확인
/permissions 표시

세 가지 규칙과 평가 순서

deny가 항상 우선

THREE RULES

규칙은 allow, ask, deny 세 종류입니다. 도구 호출은 deny → ask → allow 순으로 확인하며 처음 일치한 규칙이 결과를 정합니다. 일치가 없으면 현재 모드가 동작을 결정합니다.

DENY

무조건 차단

일치 시 즉시 거부, 항상 최우선.

ASK

명시 확인

일치 시 반드시 확인 프롬프트 표시.

ALLOW

무확인 실행

일치 시 확인 프롬프트 없이 즉시 실행.

ELSE

모드 위임

무일치 시 현재 권한 모드의 기본 동작 수행.

deny 우선
평가 원칙

전 스코프 병합
목록 합집합

구체성과 무관
좁은 allow 예외 불가

모드 폴백
무일치 처리

규칙 문법 해부

Tool(specifier) — 하나의 형식

rule anatomy

```
# 형식은 Tool 또는 Tool(specifier), 두 모양뿐
"Bash(npm run lint)"        # 지정자 O → 그 호출만
"WebFetch"                  # 지정자 X → 그 도구 전부 (= WebFetch(*))

# 지정자 없는 규칙은 목록에 따라 의미가 갈린다
"allow": ["WebFetch"]       # 모든 호출을 확인 없이 허용
"deny":  ["Bash"]           # 도구를 컨텍스트에서 제거 (Claude가 못 봄)
"deny":  ["Bash(rm -rf *)"] # 도구는 유지, 이 호출만 차단
```

1 형식
Tool(specifier)

세 목록 공용
allow·ask·deny 같은 문법

지정자 없는 deny
도구를 컨텍스트에서 제거

도구별 의미
지정자 해석이 다름

Bash 패턴

매칭은 정확한 일치와 와일드카드 두 가지 방식으로 동작

bash patterns

```
"Bash(npm run lint)"    # 정확히 이 명령만
"Bash(npm run test *)"  # 접두 + 인자 (공백=단어 경계)
"Bash(git *)"           # git 하위 전부
"Bash(git * main)"      # * 위치 자유, 공백 포함 매칭
# deny 예시
"Bash(curl *)"          # 임의 curl 차단
"Bash(rm -rf *)"
# 복합 명령은 셸 연산자(&& || ; |)로 분해 후 하위 명령별 매칭 / timeout, time, nohup 고정 래퍼 벗김
# npx·docker exec 는 접두 allow가 뒤 명령까지 적용됨 (접두 allow = 임의 실행) / find -exec 는 접두 allow가 적용 안 됨
# 우회 가능성 있음: 1차 방어, 강한 봉쇄는 sandbox 계층
```

정확/와일드카드	* 인자	1차 방어	sandbox
두 매칭 형태	접두 허용 관용구	권한의 역할	강한 봉쇄 계층

쓰지 않아도 되는 규칙

기본 읽기 명령은 이미 허용되어 있음

built-in read-only

```
ls · cat · echo · pwd · head · tail · grep · find · which · diff · stat · du · cd · wc
git status · git log · git diff   # git 읽기 전용 형식 포함
ls *.ts · wc -l src/*.py          # 와일드카드도 통과 (모든 플래그가 읽기 전용 시)
cd packages/api && ls             # 작업 경로 내 cd 복합 통과 (cd+git은 프롬프트)
# find·sort·sed는 와일드카드 시 여전히 프롬프트 (-delete 등 확장 위험)
# 세 목록으로 재정의 (집합 자체는 못 바꿈)
"ask":  ["Bash(git diff *)"]      # 기본 허용에도 확인 강제
"deny": ["Bash(find *)"]          # 기본 허용 위에 차단
# 설계 원칙: 이미 허용된 명령에 allow 규칙을 쓰지 않는다
#  → git 조회 allow, Grep allow는 불필요
```

기본 읽기 명령
구성 불가, 전 모드

와일드카드
읽기 플래그 한정

ask·deny 재정의
기본 허용에도 강제

allow 불필요
통과 명령엔 불필요

파일 경로 패턴

Read, Edit의 gitignore 스타일 경로 패턴

path patterns

```
"Read(./.env)"          # 현재 디렉토리의 .env
"Read(./.env.*)"        # .env.local 등 변형
"Read(./secrets/**)"    # 하위 전체
"Edit(*.ts)"            # 확장자 단위
"Read(.env)"            # 무접두 파일명: 전 깊이 (**/.env와 동일)
"Read(~/.zshrc)"        # 홈 기준 경로
"Read(//etc/passwd)"    # 절대 경로는 // 접두
"Edit(/src/**)"         # / 하나는 설정 파일 위치 기준!
# * 한 경로 구간 안, ** 하위 디렉토리 전체
# 심링크: allow 양쪽 필요, deny 한쪽이면 차단
```

gitignore식	./ vs /	// 절대	~/ 홈
경로 패턴 문법	cwd vs 설정 소스	특수 접두	홈 기준

규칙이 적용되지 않을 때 작동 방식

6가지 모드 — deny와 ask 규칙은 어느 모드에서나 유효

default (Manual)	표준: 위험 작업마다 확인	일상 기본
acceptEdits	작업 경로 편집·파일 명령 자동 수락	반복 수정 세션
plan	읽기 전용 탐색, 계획 산출	설계 단계
auto	분류기(별도 판정 모델)가 심사 후 자동 결정	신뢰 저장소
dontAsk	확인 프롬프트 자동 거부	무인: allow + 읽기 전용만
bypassPermissions	확인 생략 (명시적 deny·ask 규칙은 예외)	격리 한정, 루트·홈 rm은 확인 유지

Shift+Tab 순환
default·acceptEdits·plan 기본

deny·ask 불변
모드가 규칙을 못 이김

bypass도 예외
루트·홈 rm은 확인 유지

dontAsk 진입
--permission-mode 플래그

Auto 모드 — 분류기 판정 원리

패턴이 아니라 속성을 본다

CLASSIFIER

규칙에 걸리는 건 규칙이 먼저 처리하고, 남은 것만 분류기가 봅니다. 정해진 명단이 아니라 되돌릴 수 없는 작업인지, 무언가를 부수는지, 신뢰한 범위를 벗어나는지를 판단합니다.

ORDER

판정 경로

규칙 우선. 읽기·작업 경로 편집 자동 승인. 광역 allow 일시적 해제

BLOCK

기본 차단

curl | bash, force push, git reset --hard류, IaC 파괴.

ALLOW

기본 허용

락파일 의존성 설치, .env 읽고 매칭 API 전송.

FALLBACK

폴백 임계값

연속 3회·총 20회 차단 시 정지, 승인은 연속만 리셋.

차단 사유 전달

Claude가 대안 시도

최근 거부됨 탭

/permissions, r 재시도

auto-mode config

전체 규칙 CLI 출력

추론 결과 미전달

모델에 의한 설득 방지

Auto 분류기 조정

분류기 규칙을 설정으로

settings.json (autoMode)

```
{ "autoMode": {
    "environment": ["$defaults", "Sensitive remote targets: prod-*"],    // 신뢰 + 민감 대상
    "allow":      ["$defaults", "kubectl get 계열"],      // 예외 허용
    "soft_deny":  ["$defaults", "terraform apply 금지"],   // 명시 요청 시 허용
    "hard_deny":  ["$defaults", "IAM 정책 변경"],        // 무조건 차단
    "classifyAllShell": true
} }
# 규칙은 패턴이 아니라 문장으로 작성 / $defaults 누락 = 내장 규칙 통째 교체
# 우선순위: hard_deny → soft_deny → allow → 사용자의 명시적 의도
# 확인: claude auto-mode config (실효 설정) · critique (규칙 검토)
# 가용: 전 프로바이더 · 모델 요건 충족 · Team·Ent는 Owner 활성화
```

문장 규칙
패턴이 아닌 문장

환경 선언
신뢰 대상 + 민감 대상

4단 우선순위
hard·soft·allow·명시 의도

claude auto-mode defaults
$defaults 항목 확인

흔한 실수

가장 잦은 실수 3가지

MISTAKE

정확 일치에 인자를 기대

"Bash(npm run test)"

--watch 붙으면 불일치

교정

"Bash(npm run test *)"

인자 허용은 공백 + 별표

MISTAKE

비밀 파일 변형 누락

"Read(.env)"

.env.local 은 통과

교정

"Read(**/.env*)"

변형과 전 깊이를 차단

MISTAKE

deny에 allow 예외 기대

deny "Bash(aws *)" + allow "Bash(aws s3 ls)"

deny가 항상 먼저: 예외 불가

교정

allow로 좁게 열고 차단은 혹으로

deny에는 예외 구조가 없음

Part 2 Recap

Permissions 핵심 정리

KEY TAKEAWAY

한 형식 Tool(specifier), 세 규칙, deny 우선. 순서가 전부입니다.

핵심 정리

deny → ask → allow → 무일치는 모드

Bash 와일드카드 · gitignore식 경로

모드 6종 — dontAsk는 fail-safe

이미 통과하는 명령에 allow 금지

local 검증 후 project 승격

다음 — Part 3 Hooks 아키텍처

규칙 문법으로 표현 못 하는 판단은?

지시는 확률적, 훅은 결정론적

오늘 본 규칙 문법이 훅의 if에서 재사용

lifecycle · matcher · 입출력 계약

CHAPTER 04 / PART 03

Hooks 아키텍처

훅 하나의 해석 과정을 이해하고, 이벤트 전체로 일반화

Motivation — 확률적 지시 vs 결정론적 실행

Resolution — block-rm 훅의 해석 5단계

Contract — 구성 · matcher · if · 입출력

Boundary — 경계 지도 · 권한과의 역할 분담

왜 훅인가 — 지시의 한계

지시는 확률적, 훅은 결정론적

실패 시나리오

CLAUDE.md에 “파일 수정 후 반드시 포매터 실행”이라고 적어도, 따를지는 모델의 선택이라 실행이 생략될 수 있습니다.

구분	CLAUDE.md 지시	Hook
전달 대상	모델	런타임
실행 보장	확률적 (따를 수도, 안 따를 수도)	이벤트 발생 시 항상
적합 용도	판단이 필요한 지침	예외 없이 강제할 규칙

컴팩션 손실
대화 세부는 요약에서 소실

확률적 생성
따를 수도 안 따를 수도

세션 간 소실
리셋 시 맥락 초기화

훅 = 코드로 강제
런타임이 실행

Hook이란

수명주기 지점에 붙는 자동 실행

HOOKS

훅은 모델에게 부탁하는 지시가 아니라 Claude Code 런타임 경계에서 자동 실행되는 handler다.

command	http	mcp_tool	prompt	agent
셸 명령	HTTP POST	MCP 도구 호출	단발 LLM 판정	서브에이전트 검증
기본 타입 — stdin으로 JSON 수신	이벤트를 지정 URL로 전송	연결된 MCP 서버 도구 실행	모델에 보내 JSON 판정 수신	도구로 조건 확인 후 결정 반환

런타임 실행
모델 아님

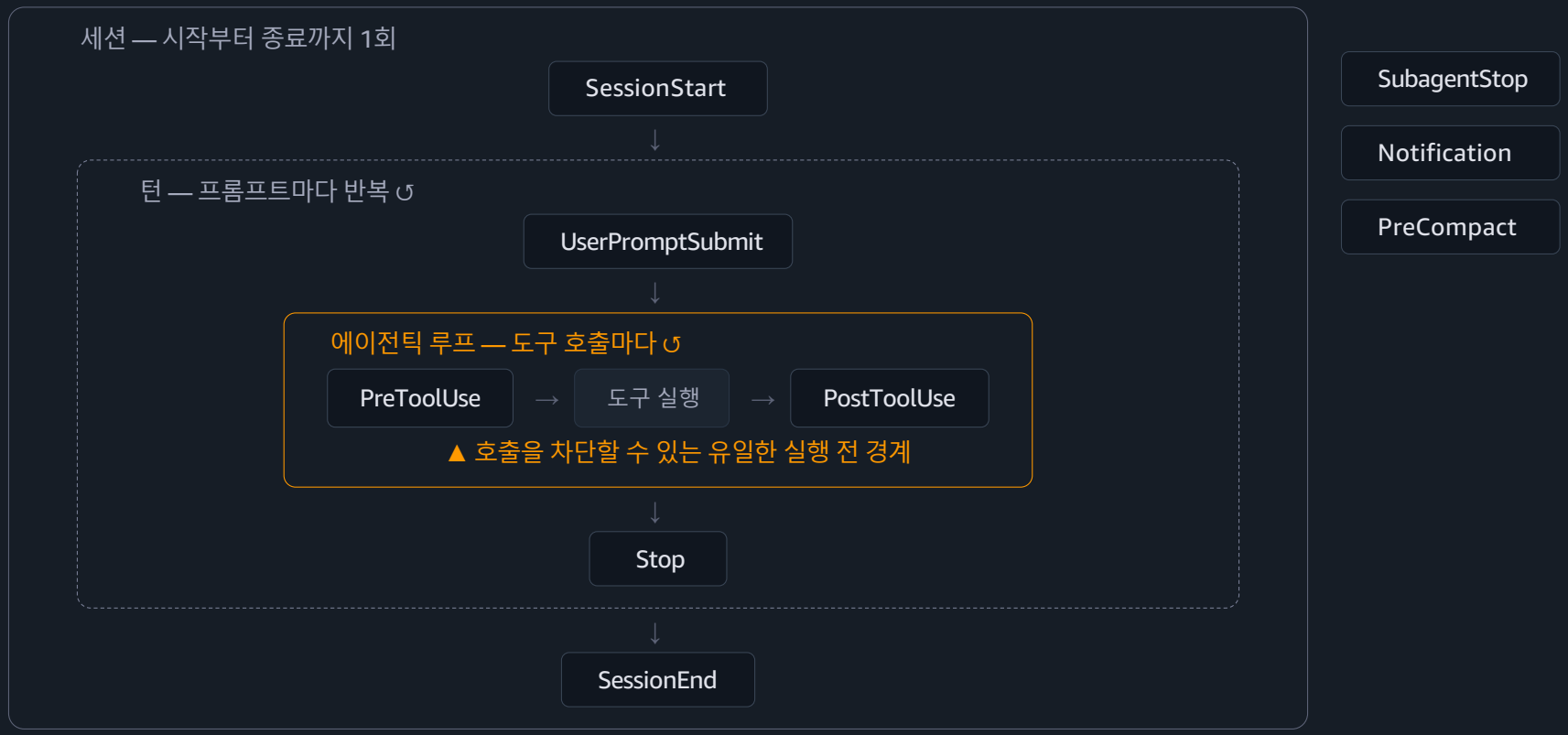
이벤트 JSON 수신
stdin / POST

결정 반환
exit / JSON

5 핸들러 타입
command가 기본

Lifecycle 지도 — 세 개의 중첩 주기

이벤트는 목록이 아니라 구조 — 세션 안에 턴, 턴 안에 도구 호출



구성 3단 중첩

이벤트 · 매처 그룹 · 핸들러

.claude/settings.json (hooks)

```
{ "hooks": {
  "PreToolUse": [    // 1 event
   { "matcher": "Bash", // 2 matcher
     "hooks": [      // 3 handler
      { "type": "command",
        "if": "Bash(rm *)",
        "command":
         "${CLAUDE_PROJECT_DIR}/.claude/hooks/block-rm.sh"
      } ] }
  ] }
}
```

1 Event
수명주기 지점 — PreToolUse 등

2 Matcher group
대상 좁힘 — Bash, Edit|Write

3 Handler
command 등 5타입 중 명시

정의 위치
user·project·local·managed, plugin, frontmatter

3단 구조 동일
어디에 정의하든 같음

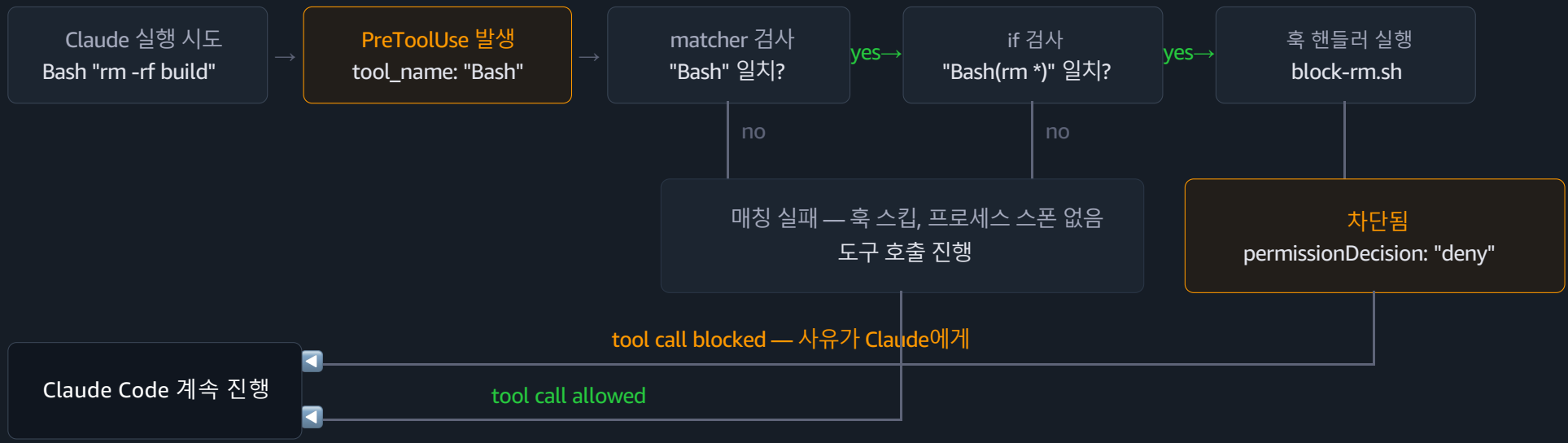
type 필수
5타입 중 명시 지정

if = 권한 문법
파트 2 문법 재사용

등록 확인
/hooks 메뉴 (읽기 전용)

훅 하나의 해석 과정

block-rm 훅이 rm -rf를 차단하기까지 5단계



matcher
키 하나만 비교

if
스폰 없이 인자 검사

handler
전체 JSON을 stdin으로

순서 = 비용
싼 검사가 비싼 스폰을 지킴

Decision Map

지점마다 결정이 미치는 범위가 다름

다중 훅 — 결정 병합

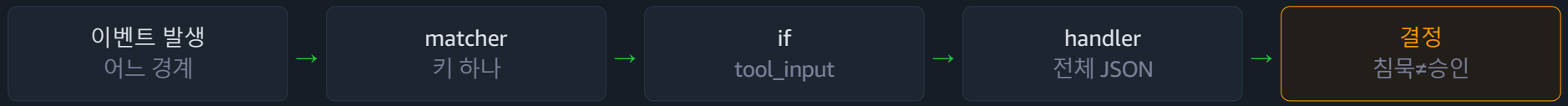
가장 제한적인 결정 우선: deny > defer > ask > allow — additionalContext는 모든 훅의 값을 전달

Permission layer는 항상 평가

deny·ask 규칙은 훅 반환과 무관: hook allow로 우회 불가. 파괴적 동작은 PreToolUse에서

Part 3 Recap

Hooks 아키텍처 핵심 정리



핵심 정리

지시는 확률적, 훅은 런타임이 실행하는 결정론

출력 계약: 차단은 exit 2 또는 deny JSON

hook allow는 권한 정책을 완화하지 못함

이벤트는 6경계 지도로 — 실행 동작까지 설계

다음 — Part 4 MCP

외부 시스템과의 연결 — 마지막 층

mcp__ 접두 이름이 권한·훅의 대상

스코프 3종 — Part 1 파일 지도의 MCP 편

조직 통제와 컨텍스트 예산

CHAPTER 04 / PART 05

MCP 구성

외부 세계로 열린 표준 다리

3 프리미티브와 4 전송

설치 CLI와 스코프 3종

.mcp.json과 OAuth

도구 검색과 컨텍스트 예산

MCP와 3 프리미티브

도구, 리소스, 프롬프트

MODEL CONTEXT PROTOCOL

MCP는 모델과 외부 시스템을 잇는 개방 표준입니다. 서버는 세 종류의 능력을 노출하고, Claude Code가 클라이언트로서 이를 소비합니다.

TOOLS

도구

모델이 호출하는 동작, mcp__서버__도구 이름으로 등장.

RESOURCES

리소스

읽을 수 있는 데이터, @ 멘션으로 컨텍스트에 첨부.

PROMPTS

프롬프트

서버 제공 정형 명령, 슬래시로 노출.

ROLE

역할 구도

Claude Code = 클라이언트, 연결 대상 = 서버.

개방 표준
생태계 호환

mcp__
도구 이름 접두

@ 멘션
리소스 소비

/mcp__
프롬프트 호출

MCP 예시

BEFORE — 그냥 파이썬 함수

```
def get_order_status(order_id: str) -> dict:
    """주문 상태를 조회한다."""
    row = db.query(ORDERS, id=order_id)
    if row.shipped_at: return {"state": "배송중"}
    return {"state": row.state}
```

에이전트가 읽는 것 ↓

```
def get_order_status(order_id: str) -> dict: """주문 상태를 조회한다."""
row = db.query(ORDERS, id=order_id) if row.shipped_at: return {"state": "배송중"}
return {"state": row.state}
```

@mcp.tool()

⟶

AFTER — 데코레이터를 붙인 같은 함수

```
@mcp.tool()
def get_order_status(order_id: str) -> dict :
    """
주문 상태를 조회한다. """
    row = db.query(ORDERS, id=order_id)
    if row.shipped_at: return {"state": "배송중"}
    return {"state": row.state}
```

에이전트에 전달되는 것 ↓

```
{ "name": "get_order_status" ,
  "description":  "주문 상태를 조회한다." ,
  "inputSchema": { "order_id":  "string"  },
  "outputSchema": { "state":  "string" } }
```

설치 / claude mcp add

전송별 추가 문법

install commands

```
# stdio: -- 뒤가 실행 명령
$ claude mcp add playwright -- npx -y @playwright/mcp@latest
# 원격: transport 지정
$ claude mcp add --transport http github https://api.githubcopilot.com/mcp/
# 스코프 지정 (-s): local(기본) | project | user
$ claude mcp add -s project sentry -- npx -y @sentry/mcp
$ claude mcp list / get <name> / remove <name>
```

-- 구분자	--transport	-s 스코프	list/get/remove
stdio 명령 경계	원격 지정	local 기본	관리 3종

스코프 3종과 저장 위치

Part 1 파일 지도의 MCP 편

local (기본)	~/.claude.json의 프로젝트별 항목	이 저장소의 나만, 비공유
project	.mcp.json (저장소 루트)	커밋 공유, 팀 표준
user	~/.claude.json 전역 항목	내 전 프로젝트
우선순위	local > project > user (동명 시, 통째 대체)	설정 계층과 동형
신뢰 확인	project 서버는 최초 사용 승인 절차	저장소발 구성 보호

.mcp.json

팀 공유 서버 정의와 변수 확장

.mcp.json (project scope)

```
{ "mcpServers": {
    "corp-wiki": {
      "type": "http",
      "url": "https://wiki-mcp.corp.example/mcp",
      "headers": { "Authorization": "Bearer ${WIKI_TOKEN}" }
    },
    "db": {
      "type": "stdio",
      "command": "npx",
      "args": ["-y", "@corp/db-mcp"],
      "env": { "DB_HOST": "${DB_HOST:-localhost}" } } } }
```

${VAR}
환경변수 확장

:-기본값
미설정 폴백

비밀 외부화
파일에 토큰 금지

커밋 대상
팀 표준 파일

/mcp 관리 UI

연결의 상황판

MCP DASHBOARD

/mcp는 서버 목록과 연결 상태, 도구 수, 인증을 한 화면에서 다룹니다. 서버가 인

STATUS

연결 상태

connected, failed 등 서버별 헬스 표시.

DETAIL

상세 열람

도구, 리소스 목록과 스코프 출처 확인.

AU

인

O

헬스

연결 진단 1선

출처

스코프 추적

인증

OAuth 창구

Part 8

심화 진단 연결

claude.ai 커넥터

웹에서 연결한 서버가 CLI로

CLOUD CONNECTORS

claude.ai에서 연결한 커넥터가 Claude Code 세션에도 자동 등장합니다. 조직과 저장소는 설정 키로 이를 통제할 수 있습니다.

SYNC

자동 동기

웹 커넥터가 CLI 세션에 자동 페치, 연결.

OPT-OUT

저장소 거부

disableClaudeAiConnectors를 프로젝트에 커밋 가능.

PRECEDENCE

true 우선

어느 소스든 true면 차단, false로 재개방 불가.

MANAGED

조직 결합

managed-mcp.json 배포 시 기본 배제, allowAll 키로 병행.

자동 등장
기본 동작

true 승리
차단 우선 규칙

deniedMcpServers
개별 거부 대안

Ch.3 연결
managed 통제

도구 검색 / Tool Search

대량 도구 시대의 해법

ON-DEMAND TOOLS

서버가 많아지면 도구 정의만으로 컨텍스트가 잠식됩니다. 도구 검색은 설명 대신 검색 인덱스를 두고, 필요한 도구만 온디맨드로 로드합니다.

PROBLEM

정의 비대

수백 도구의 스키마가 시작 컨텍스트를 점령.

HOW

지연 로드

도구는 인덱스로만 존재, 검색 후 정의 로드.

WHEN

발동 조건

기본 활성, auto 모드는 컨텍스트 10% 임계.

EFFECT

체감

다서버 환경에서 시작 토큰 급감, 정확한 도구 선택.

인덱스
정의 대체물

온디맨드
로드 시점

기본 활성
auto: 10% 임계

SDK 문서
tool-search 상세

컨텍스트 예산

서버는 공짜가 아니다

CONTEXT BUDGET

연결된 서버의 도구 설명은 컨텍스트를 소비합니다. 도구 검색이 완화하지만, 스코프 설계로 노출 자체를 줄이는 것이 근본입니다.

잠식 신호

/context에서 MCP tools 비중 확인 (/doctor는 사용률 분석)

시작부터 컨텍스트 수십 퍼센트 점유

안 쓰는 서버가 user 스코프에 상주

무거운 서버를 전 프로젝트에 노출

다이어트 처방

프로젝트별 필요 서버만 .mcp.json

개인 상주는 최소, 나머지는 온디맨드 add

무거운 서버는 서브에이전트 mcpServers로 (Ch.2)

/mcp 토글로 세션 단위 오프

Part 5 Recap

MCP 구성 핵심 정리

KEY TAKEAWAY

3 프리미티브, 4 전송, 3 스코프가 문법의 전부입니다. 팀 표준은 .mcp.json 커밋, 인증은 OAuth 위임, 규모는 도구 검색과 스코프 설계로 다스립니다.

핵심 정리

도구, 리소스, 프롬프트 소비 문법

stdio는 --, 원격은 --transport

local/project/user + 신뢰 승인

${VAR} 확장, 비밀 외부화

도구 검색 + 컨텍스트 예산

다음 — Part 6 MCP 운영과 보안

서버 카탈로그

신뢰 모델과 인젝션

조직 통제 요약

트러블슈팅

CHAPTER 04 / PART 06

MCP 운영과 보안

연결의 규율

인기 서버 카탈로그

신뢰 모델과 프롬프트 인젝션

조직 통제와 훅 결합

성능과 트러블슈팅

인기 서버 카탈로그

현장에서 검증된 출발선

github	이슈, PR, 코드 검색과 조작	http, OAuth
playwright	브라우저 구동, E2E와 스크린샷	stdio, npx
sentry	오류 이벤트 조회와 분석	http, OAuth
slack	채널 검색, 메시지 발신	http, OAuth
postgres 계열	스키마 조회, 쿼리 실행	stdio, 자격은 env
notion, linear, figma	문서, 이슈, 디자인 컨텍스트	http, OAuth

사내 MCP 서버

만드는 쪽의 개요 (상세는 Ch.6)

BUILD YOUR OWN

사내 위키, 배포, 티켓 시스템을 서버로 노출하면 조직 컨텍스트가 도구가 됩니다. 제작은 Agent SDK 계열 문서와 챕터 6에서 심화합니다.

WHAT

노출 대상

위키 검색, 배포 상태, 티켓 CRUD, 사내 API.

STACK

제작 스택

TypeScript, Python MCP SDK로 도구 정의.

SHIP

배포 형태

stdio 패키지 또는 사내 http 엔드포인트.

GOV

운영 결합

OAuth 인증, allowlist 등재, 버전 관리.

3 프리미티브
설계 단위

http 권장
다수 사용자 배포

Ch.6
제작 심화 위치

Part 4 연계
네트워크 준비 (Ch.3)

신뢰 모델

서드파티 서버와 프롬프트 인젝션

TRUST BOUNDARY

서버가 돌려주는 텍스트는 모델이 읽는 입력입니다. 악성 서버나 오염된 데이터가 지시문을 심을 수 있으므로, 서버 선택 자체가 보안 결정입니다.

위험 지점

미검증 서버의 도구 설명, 응답에 지시 삽입

외부 데이터(이슈, 문서) 경유 간접 인젝션

과도한 쓰기 권한을 가진 도구

토큰이 서버 측에 과하게 위임됨

방어 원칙

공식, 사내 제작 서버 우선 채택

프로젝트 서버 최초 사용 승인 신중히

쓰기 도구는 ask, 민감 도구는 deny

외부 데이터 처리 시 훅 검증 결합

조직 통제 요약

Chapter 3 managed 키의 MCP 편

allowedMcpServers	승인 서버 화이트리스트	빈 배열 = 전면 봉쇄
deniedMcpServers	명시 차단, allowlist보다 우선	사고 대응 즉시 배포
allowManagedMcpServersOnly	managed 목록만 유효	최고 수준 통제
managed-mcp.json	조직 표준 서버 정의 배포	커넥터 기본 배제
disableSideloadFlags	--mcp-config 등 우회 거부	v2.1.193+ (Ch.3)
적용 범위	서브에이전트 인라인 정의 포함	Ch.2 우회 봉쇄

훅 결합

mcp__ 매처로 감사와 검증

hooks x mcp

```
{ "hooks": { "PreToolUse": [
  { "matcher": "mcp__.*__(write|create|delete).*",
    "hooks": [{ "type": "command",
      "command": "${CLAUDE_PROJECT_DIR}/.claude/hooks/mcp-write-guard.sh" }] },
  { "matcher": "mcp__corp-wiki__.*",
    "hooks": [{ "type": "command", "async": true,
      "command": "jq -c '{ts:now,tool:.tool_name}' >> ~/mcp-audit.jsonl" }] }
] } }
# 쓰기 계열 정규식 가드(동기) + 서버 전체 비동기 감사 — async 훅은 차단 불가
```

쓰기 정규식
위험 동사 매칭

서버 감사
mcp__s__.* 전량

async 기록
지연 없는 감사 — 차단 불가

Part 3 재사용
매처 문법 그대로

성능 관점

연결이 늘 때 생기는 비용

STARTUP

기동 지연

stdio 다중 스폰과 원격 핸드셰이크가 시작을 늦춤.

CONTEXT

컨텍스트

도구 정의 비대, 도구 검색과 스코프로 완화 (Part 5).

STRICT

구성 고정

--strict-mcp-config로 지정 구성 외 로드 배제.

BARE

무장식 기동

--bare로 MCP 등 확장 없이 최소 기동.

스폰 비용
stdio 다중화

인덱스화
도구 검색 효과

--strict-mcp-config
CI 재현성 — 단독 사용 시 전부 미로드

--bare
진단, 벤치 용도

MCP 트러블슈팅

증상에서 원인으로

서버 안 보임	스코프, 신뢰 승인 미완	/mcp 목록과 출처 확인
connection failed	명령 경로, URL, 네트워크	단독 실행, curl 검증
401, 인증 오류	OAuth 만료, 토큰 변수 미설정	/mcp 재인증, env 확인
도구 호출 거부	권한 규칙 미허용	/permissions에 mcp__ 추가
조직에서 차단	allowlist 미등재	관리자 등재 요청 (Ch.3)
느린 시작	서버 과다, 무거운 stdio	스코프 축소, 도구 검색

운영 모범 사례

연결의 규율 네 가지

RULE 1

표준은 커밋

팀 서버는 .mcp.json으로, 개인 상주는 최소로.

RULE 2

비밀 외부화

토큰은 ${VAR}와 볼트, 파일에는 절대 금지.

RULE 3

쓰기는 관문

쓰기 도구 ask + 훅 가드, 읽기는 넉넉히.

RULE 4

분기 정리

미사용 서버 제거, 버전과 인증 상태 점검.

커밋 표준
팀 정합

0 secrets
파일 원칙

read easy
write hard

분기 점검
위생 주기

Part 6 Recap

MCP 운영과 보안 핵심 정리

KEY TAKEAWAY

서버 응답은 모델의 입력입니다. 서버 선택, 조직 allowlist, 훅 가드의 3중 방어에 성능 다이어트를 더하면 연결의 규율이 완성됩니다.

핵심 정리

카탈로그: SaaS는 OAuth http

인젝션: 응답도 입력이다

3중 방어: 권한, 조직, 훅

--strict-mcp-config, --bare

표준 커밋 + 비밀 외부화

다음 — Part 7 Commands와 Skills

확장 수단 선택 지형

커스텀 명령 문법

SKILL.md 구조

팀 라이브러리