CLAUDE CODE DEEP DIVE WORKSHOP

Chapter 2 - Agents (Subagents)

맞춤 서브에이전트로 작업을 분담하고 병렬화하기

Update : 2026.09 / Claude Code v2.1.251 최신 채널 기준, 공식 문서 code.claude.com/docs 반영

Junseok Oh
Sr. Solutions Architect
AWS Korea
atomoh@amazon.com

Chapter 2 Agenda

10 Parts

PART 01	Subagents 개념과 원리	격리 컨텍스트, Built-in, 실행 모델
PART 02	Subagents 정의 방법	스코프 5계층, 프론트매터 17필드
PART 03	Agent 도구와 디스패치	자동 위임, @멘션, fork, 중첩, resume
PART 04	Pattern 1 / Code Reviewer	리뷰 에이전트와 영속 메모리
PART 05	Pattern 2 / Tester	테스트 작성과 worktree 격리
PART 06	Pattern 3 / Security Scanner	읽기 전용 스캐너와 훅 검증
PART 07	Pattern 4 / Docs Writer	문서 에이전트와 skills 프리로드
PART 08	Pattern 5 / Migration Bot	대규모 마이그레이션 안전 장치
PART 09	병렬 오케스트레이션	Agent Teams, Workflows, loop과 graph
PART 10	Recap & Q&A	핵심 요약, FAQ, Labs, Q&A

Chapter 2 Learning Objectives

본 챕터에서 달성할 학습 목표

AT THE END OF THIS CHAPTER

서브에이전트의 격리 모델을 설명하고, 프론트매터로 맞춤 에이전트를 정의하며, 자동 위임과 명시 호출, 병렬 실행을 실무 워크플로에 적용할 수 있습니다.

OBJ 1

격리 모델 이해

컨텍스트 격리와 요약 회수, fork와 Teams, Workflows의 경계를 설명.

OBJ 2

에이전트 정의

프론트매터 17필드로 도구, 모델, 메모리, 훅을 설계.

OBJ 3

디스패치 운용

자동 위임, @멘션, 병렬과 체이닝, resume, 6대 병렬화 수단.

OBJ 4

실전 패턴 구축

리뷰어부터 마이그레이션 봇까지 5종을 직접 구현.

10 Parts	4 Labs	5 Patterns	~1.2h + 0.5h
구조화된 학습 경로	Chapter 2 실습 랩	실전 에이전트 구현	강의와 실습 시간

02

Agents (Subagents)

위임과 병렬화로 컨텍스트를 지키는 기술

CHAPTER 02 / PART 01

Subagents 개념과 원리

격리 컨텍스트와 위임의 가치

- Subagent 정의와 컨텍스트 격리 모델
- Built-in 에이전트: Explore, Plan, general-purpose, claude
- Fork 기본 모드와 Background 실행 모델 (v2.1.232)
- Fork, Teams, Workflows, Cross-session의 경계

Subagent란 무엇인가

자기만의 컨텍스트 윈도우를 가진 격리 인스턴스

DEFINITION

Subagent는 특정 작업을 처리하는 격리된 Claude 인스턴스입니다.

자체 컨텍스트 윈도우, 맞춤 시스템 프롬프트, 별도 도구 권한으로 독립 작업 후 요약만 돌려줍니다.

ISOLATED

자체 컨텍스트

메인 대화와 분리된 윈도우에서 작업, 상호 오염 없음.

FOCUSED

전용 프롬프트

역할에 특화된 시스템 프롬프트로 한 가지 일에 집중.

SCOPED

도구 격리

허용 도구를 좁혀 읽기 전용 등 안전한 권한 설계.

SUMMARY

요약 회수

수십 개 도구 호출의 결과가 요약 한 덩이로 귀환.

브라우저 탭
겹가지 추적의 비유

1 Session
한 세션 안의 워커

Markdown
정의 파일 형식

Agent tool
생성 도구 이름

왜 위임하는가

컨텍스트 오염이 품질을 깎는다

THE PROBLEM

검색 결과, 로그, 파일 내용이 메인 대화를 채우면 모델의 주의가 분산되고 응답 품질이 떨어집니다.
다시 참조하지 않을 출력이 특히 낭비입니다.

위임 없이 (메인에서 직접)

- 30개 파일 읽기가 그대로 이력에 축적
- 테스트 로그 수천 줄이 윈도우 점유
- Compaction이 빨리 오고 초기 지시 소실
- 긴 세션일수록 응답 품질 하락

위임하면 (Subagent 경유)

- 대량 읽기는 자식 컨텍스트에서 소화
- 메인에는 실패 테스트 요약만 도착
- 본 대화는 결정과 방향에만 사용
- 긴 세션에도 컨텍스트가 가볍게 유지

무엇이 넘어오고, 무엇이 안넘어오나

Subagent 초기 컨텍스트의 여섯 구성

System Prompt	에이전트 자신의 프롬프트 + 작업 디렉토리 등 환경 정보, Claude Code 전체 프롬프트는 미포함
Task Message	메인 Claude가 작성한 위임 지시문, 대화 이력은 오지 않음
CLAUDE.md + Memory	메인이 로드한 메모리 계층 전체, 단 Explore와 Plan은 생략
Git Status	부모 세션 시작 시점 스냅샷, Explore와 Plan은 생략
Preloaded Skills	skills 필드에 지정한 스킬의 본문 전체 + 형제 로스터(v2.1.206): 이름이 붙은 다른 에이전트 목록, SendMessage 대상

Built-in / Explore

읽기 전용 탐색 전문 에이전트

EXPLORE AGENT

코드베이스 검색과 분석에 최적화된 읽기 전용 에이전트입니다.

Claude가 수정 없이 코드를 이해해야 할 때 자동으로 위임합니다.

MODEL
메인 모델 상속

v2.1.198부터 상속.
Claude API는 Opus 상한,
Bedrock 등은 직접 상속.

TOOLS
읽기 전용

Write와 Edit 거부, 검색과
읽기 도구만 허용.

DEPTH
3단계 강도

quick, medium, very
thorough를 호출 시 지정.

LIGHT
경량 설계

CLAUDE.md와 git 상태를
생략해 빠르고 저렴.

상한 Opus
Claude API 캡

직접 상속
Bedrock 등 타 공급자

model: haiku
같은이름의 정의로 오버라이드

One-shot
resume 불가 특성

실행 모델 / Foreground vs Background

v2.1.232부터 fork 모드가 기본, 그래서 백그라운드가 기본

BACKGROUND BY DEFAULT

대화형 세션은 fork 모드가 기본이라 서브에이전트가 항상 백그라운드에서 돕니다.

fork 모드에서는 포그라운드 요청 자체가 불가하며, 권한 프롬프트는 메인 세션에 떠오릅니다.

FG
Foreground

완료까지 메인 차단. fork 모드가 꺼진 세션과 -p, SDK에서만.

BG
Background

동시 진행, 완료 시 결과가 메시지로 도착. 대화형 기본값.

PERMS
권한 표면화

승인 요청이 메인에 뜨고 요청 에이전트 표기.
v2.1.186 이전은 자동 거부

CONTROL
조작

Ctrl+B 백그라운드 전환. 백그라운드는 도구 집합이 좁혀짐(2차 필터).

v2.1.232+	Ctrl+B	Esc	CLAUDE_CODE_DISABLE_BACKGROUND_TASKS=1
fork 기본화 버전	백그라운드 전환 키	개별 승인 거부	

모델 해석 우선순위

Subagent가 어떤 모델로 도는가 (v2.1.251에서 순서 변경)

1 호출 파라미터	메인 Claude가 Agent 도구 호출에 함께 넘기는 model 값, resume에도 유지
2 Frontmatter	정의 파일의 model 필드: sonnet, opus, haiku, fable, 전체 ID, inherit
3 환경변수	CLAUDE_CODE_SUBAGENT_MODEL, inherit 외 값일 때만. 내장 Explore/Plan엔 무효
4 메인 모델	위가 모두 없으면 메인 대화의 모델을 그대로 상속 (기본값)

경계 지도 / 6가지 병렬화 수단

판단 기준: 세션이 하나인가, 소통이 필요한가, 계획을 누가 들고 있는가

Subagent	한 세션 안의 워커, 새 컨텍스트, 요약 회수	곁가지 격리, 병렬 리서치
Fork	대화 전체를 물려받은 서브에이전트, /subtask	설명 없이 맡기는 곁가지
Agent view	독립 백그라운드 세션을 한 화면에서 조종, claude agents	무관한 작업들의 병렬 운영
Agent Teams	공유 태스크와 직통 메시지로 협업, 실험 기능	소통이 필요한 지속 병렬
Dynamic Workflows	스크립트가 수십~수백 에이전트를 조율	전수 감사, 대규모 마이그레이션
Cross-session	내가 띄운 세션끼리 발견과 상태를 주고받음	워크트리 병행 작업 조율

Cost & Latency 고려사항

위임에도 가격표가 있다

OVERHEAD AWARENESS

서브에이전트는 새 컨텍스트에서 출발하므로 상황 파악 시간이 들고, 결과 회수도 토큰을 씁니다.

이득이 오버헤드를 넘는지 판단이 필요합니다.

STARTUP

기동 비용

새 출발이라 컨텍스트 수집 시간 발생, 빠른 단발 작업엔 비효율.

RETURN

회수 비용

다수 에이전트의 상세 결과가 메인을 다시 채울 수 있음.

MODEL

모델 배분

탐색성 워커는 haiku 지정으로 비용 절감.

CACHE

캐시 관점

fork는 메인 캐시 공유. 팬아웃은 첫 에이전트 캐시를 형제가 재사용.

Fresh start
기동 오버헤드 원인

요약 지시
회수 비용 통제법

haiku
탐색 워커 권장

Cache hit
fork의 비용 이점

Subagent 안티 패턴

위임하지 말아야 할 순간

피해야 할 것

- 잦은 왕복이 필요한 대화형 작업 위임
- 계획, 구현, 테스트가 맥락을 공유하는 일 분리
- 한 줄 수정 같은 초단발 작업 위임
- 전 단계 결과에 의존하는 조사들의 병렬화
- 만능 에이전트 하나로 모든 역할 해결

대신 이렇게

- 왕복형 작업은 메인 대화에서 직접
- 맥락 공유 단계는 한 흐름으로 유지
- 빠른 확인은 /btw, 맥락 질문에 적합
- 의존 조사는 체이닝으로 순차 실행
- 역할별 단일 책임 에이전트로 분리

CHAPTER 02 / PART 02

Subagent 정의 방법

Markdown 한 장으로 나만의 워커 만들기

- 정의 파일 해부와 스코프 5계층
- 필수 필드: name, description 작성법
- 도구, 모델, 권한 모드, 메모리 필드
- skills 프리로드, mcpServers, experimental

정의 파일 해부

YAML Frontmatter + 시스템 프롬프트 본문

.claude/agents/code-reviewer.md

```
---
name: code-reviewer
description: Reviews code for quality and best practices
tools: Read, Glob, Grep
model: sonnet
---

You are a code reviewer. When invoked, analyze the
code and provide specific, actionable feedback on
quality, security, and best practices.
```

Frontmatter
메타데이터와 설정

본문
시스템 프롬프트

필수 2
name, description

파일명 자유
정체성은 name 필드

스코프 5계층과 우선순위

같은 이름이 충돌하면 위가 이긴다

1 Managed	조직 관리 설정 디렉토리의 .claude/agents/, 관리자가 배포
2 --agents 플래그	실행 시 JSON으로 주입, 세션 한정, 디스크 저장 없음
3 Project	.claude/agents/, 버전 관리로 팀 공유, 실무의 중심
4 User	~/.claude/agents/, 내 모든 프로젝트에서 사용
5 Plugin	플러그인의 agents/, 스코프 이름 my-plugin:name으로 등록

name과 description

자동 위임을 좌우하는 필수 2필드

THE TRIGGER FIELDS

Claude는 description을 읽고 위임 여부를 판단합니다.
언제 이 에이전트를 써야 하는지가 명확할수록 자동 위임의 정확도가 올라갑니다.

NAME
소문자와 하이픈

고유 식별자, 훅에는 agent_type으로 전달됨.

WHEN
시점 서술

무엇을 하는지보다 언제 쓰는지를 담아야 위임 판단 가능.

PROACTIVE
능동 문구

use proactively, MUST BE USED가 자동 위임을 강화.

SPECIFIC
구체 신호

auth, payments 변경 전과 같은 트리거 조건을 명시.

lowercase-
name 규칙

언제
description의 핵심

proactively
능동 위임 키워드

agent_type
훅 매처 값

description 작성 비교

위임되는 설명 vs 무시되는 설명

무시되는 설명 (막연함)

- 코드를 리뷰하는 에이전트
- 테스트 관련 도우미
- 보안 전문가
- 역할만 있고 시점이 없음
- Claude가 위임 시점을 못 찾음

위임되는 설명 (시점 명확)

- Use proactively after writing or modifying code
- MUST BE USED when tests fail or coverage drops
- Use before commits touching auth, payments, or user data
- 행동 트리거가 문장에 내장
- 매칭 조건이 곧 자동 위임 규칙

tools 필드 / 허용 목록

역할에 필요한 도구만 부여

tools allowlist

```
---
name: safe-researcher
description: Research agent with restricted capabilities
tools: Read, Grep, Glob, Bash
---

# 생략 시: 메인의 전체 도구를 상속 (MCP 포함)
# 지정 시: 나열한 도구만 사용 가능

# 1차 필터 - 모든 subagent에서 원천 불가 (9종)
#  AskUserQuestion, EnterPlanMode, ExitPlanMode,
#  EndConversation, ScheduleWakeup, TaskOutput,
#  WaitForMcpServers, Workflow, 깊이 한계의 Agent
# 2차 필터 - 백그라운드는 내장 도구가 더 좁아짐 (fork는 면제)
```

Allowlist
tools 의미

생략=상속
MCP까지 전체

필터 2단
1차 9종 + BG 축소

Agent 포함
중첩 스폰 허용 스위치

model 필드

역할별 모델 배분

alias	sonnet, opus, haiku, fable	일반적인 선택
전체 ID	claude-opus-5, claude-sonnet-5	버전 고정 필요 시
inherit	메인 대화의 모델 그대로	생략 시 기본값
검증	조직 availableModels 허용 목록 통과 필수	제외 모델은 상속으로 폴백
사고 설정	확장 사고는 v2.1.198부터 메인 설정을 상속	에이전트별 개별 설정 없음

inherit
생략 시 기본

fable
최상위 alias 지원

haiku
탐색 워커 권장

allowlist
조직 통제 적용

permissionMode 필드

에이전트별 권한 모드 오버라이드

default / manual	표준 확인 프롬프트, manual은 별칭 v2.1.200+	생략 시 메인 상속, Pro/Max/Team은 auto가 기본
acceptEdits	작업 경로의 편집 자동 수락	반복 수정 워커
auto	분류기가 명령과 보호 경로 쓰기 검토	신뢰 저장소
dontAsk	확인 프롬프트를 자동 거부	무인, 읽기 중심
plan	읽기 전용 탐색	조사 전용 워커
부모 우선	부모가 bypass, acceptEdits, auto면 그쪽 우선	프론트매터 무시됨

skills 프리로드

도메인 지식을 시작부터 주입

skills preload

```
---
name: api-developer
description: Implement API endpoints following team conventions
skills:
  - api-conventions
  - error-handling-patterns
---

# 나열한 스킬의 본문 전체가 시작 컨텍스트에 주입
# 나열되지 않은 스킬도 Skill 도구로 실행 중 호출은 가능
# 스킬 호출 자체를 막으려면 tools에서 Skill 제외
```

본문 전체
설명이 아닌 풀 주입

발견 불필요
탐색 단계 생략

Skill 도구
미나열분 온디맨드

컨텍스트 점유
스킬 수 만큼 늘어남

memory 필드 / 영속 메모리

세션을 넘어 학습이 쌓이는 에이전트

persistent memory

```
---
name: code-reviewer
description: Reviews code for quality and best practices
memory: project
---

You are a code reviewer. As you review code, update
your agent memory with patterns, conventions, and
recurring issues you discover.

# 스코프: user / project(권장) / local
# 첫 200줄 또는 25KB 중 먼저 닿는 쪽까지 프롬프트에 포함
# 자동 메모리를 끄면 이 필드는 무효
```

project
권장 기본 스코프

agent-memory/
project/user 경로, local은 -local

200줄/25KB
자동 로드 상한

R/W/E 자동
관리 도구 자동 부여

hooks / 프론트매터 훅

이 에이전트가 도는 동안만의 검증

db-reader with PreToolUse hook

```
---
name: db-reader
description: Execute read-only database queries
tools: Bash
hooks:
  PreToolUse:
    - matcher: "Bash"
      hooks:
        - type: command
          command: "./scripts/validate-readonly-query.sh"
---
# 스크립트가 INSERT, UPDATE 등 감지 시 exit 2로 차단
```

수명 한정
에이전트 활성 중만

exit 2
차단 + 사유 회신

Stop 변환
SubagentStop으로

Plugin 무시
보안상 제외 필드

CHAPTER 02 / PART 03

Agent 도구와 디스패치

부르고, 병렬로 돌리고, 이어서 깨우기

- 자동 위임과 명시 호출 3단계 사다리
- 병렬 리서치, 고볼륨 격리, 체이닝
- resume과 SendMessage, 중첩 실행
- /fork 상세와 관측, 디버깅

자동 위임의 판단 재료

Claude는 무엇을 보고 넘기는가

AUTOMATIC DELEGATION

요청의 성격, 각 에이전트의 description, 현재 컨텍스트 세 가지를 종합해 위임을 결정합니다.

description의 능동 문구가 판을 기울입니다.

INPUT 1

요청 성격

사용자 프롬프트의 작업 서술

INPUT 2

description

각 에이전트의 시점 서술과 능동 문구

INPUT 3

컨텍스트

현재 대화와 작업 상태

DECIDE

위임 결정

매칭 에이전트에 지시문 작성 후 스폰

호출 방법 3단계

제안에서 보장, 세션 전체까지

ESCALATION LADDER

한 번의 제안은 자연어, 이번 작업의 보장은 @멘션, 세션 전체의 기본값은 --agent입니다. 강도가 한 칸씩 올라갑니다.

LEVEL 1

자연어

test-runner 서브에이전트로 고쳐줘, Claude가 판단

LEVEL 2

@멘션

@agent-이름 지목, 이번 작업은 반드시 그 에이전트

LEVEL 3

--agent

세션 자체가 그 에이전트의 프롬프트와 도구로 실행

패턴 / 병렬 리서치

독립 조사를 동시에

~/proj $ claude

> 인증, 데이터베이스, API 모듈을 각각 별도 서브에이전트로 병렬 조사해줘. 각자 담당 영역의 구조와 핵심 흐름을 요약해서 보고해.

```
# 동작
#  독립 에이전트 3개가 동시에 탐색
#  각자 자기 컨텍스트에서 파일을 소화
#  완료되는 대로 요약이 메인에 도착
#  메인 Claude가 세 보고를 종합

# 조건: 조사 경로가 서로 의존하지 않을 것
```

동시 스폰
병렬 실행

독립 조건
의존 없는 경로

종합
메인이 합성

회수 주의
상세 결과 다발 비용

패턴 / 대량 출력 격리와 체이닝

대량 출력 소화와 순차 연결

~/proj $ claude

```
# 대량 출력 격리
> 서브에이전트로 전체 테스트 스위트를 돌리고, 실패한 테스트와 에러 메시지만 보고해줘
# 수천 줄 로그는 자식에서 소화, 실패만 귀환

# 체이닝 (의존 작업의 순차 연결)
> code-reviewer 서브에이전트로 성능 이슈를 찾고, 그다음 optimizer 서브에이전트로 고쳐줘
# 앞 결과를 메인이 받아 다음 지시문에 반영
```

격리
로그는 자식 소화

실패만
회수 형태 명시

체인
결과가 다음 입력

메인 중계
에이전트 간 직통 없음

Resume과 SendMessage

종료된 에이전트를 이어서 깨우기

```
~/proj $ claude

> code-reviewer 서브에이전트로 인증 모듈 리뷰해줘
# 완료, agent ID가 메인에 귀환

> 그 리뷰 이어서 이번엔 인가 로직을 분석해줘
# Claude가 SendMessage로 해당 에이전트를 재개
# 이전 도구 호출과 추론을 전부 가진 채 계속

# 세부
#  Claude가 멈춘 에이전트는 메시지 수신 시 자동 재개
#  내가 x로 중지한 것은 자동 재개 안 됨 (v2.1.191)
#  이름 재사용 충돌 시 전송 거부 후 대상 안내
#  Explore, Plan은 일회성이라 재개 불가
```

SendMessage
재개 수단 도구

전체 이력
이어지는 컨텍스트

자동 재개
중지 상태 수신 시

v2.1.199
이름 충돌 검사

/subtask 상세

대화 전체를 물려받는 특수 서브에이전트

```
~/proj $ claude

> /subtask 지금까지의 파서 변경에 대한 단위 테스트 초안 작성
# v2.1.212부터 /subtask, 그 이전은 /fork
# 이제 /fork는 세션 전체를 백그라운드로 복제

# 패널 조작
#  위아래 화살표  행 이동
#  Enter          트랜스크립트 열람, 후속 지시
#  x              중지 또는 완료 정리
#  Esc            프롬프트로 복귀

# 열람 중 /model, /fast는 메인 대상임을 안내
```

/subtask
v2.1.212+ 명령 이름

전체 상속
이력, 도구, 모델

캐시 공유
fork 모드 기본 v2.1.232

worktree
격리 옵션 가능

선택 가이드 / Main vs Subagent vs /btw

상황별 최적 수단

잦은 왕복, 반복 다듬기	Main 대화	격리가 걸림돌
계획, 구현, 테스트가 맥락 공유	Main 대화	한 흐름 유지
대량 출력 작업, 요약만 필요	Subagent	격리의 본령
도구, 권한 제약을 강제할 작업	Subagent	역할 설계
대화 맥락에 대한 빠른 질문	/btw	전체 참조, 이력 미기록
재사용할 프롬프트, 워크플로	Skill	메인 컨텍스트에서 실행

CHAPTER 02 / PART 04

Pattern 1 / Code Reviewer

가장 먼저 만들게 되는 리뷰어

- 시나리오와 정의 완성본
- 시스템 프롬프트 심화와 영속 메모리 결합
- 대화형, 헤드리스, GitHub Actions 세 경로
- 확장 변형과 자주 만나는 함정

시나리오 정의

무엇을 자동화하려는가

THE SCENARIO

커밋 전 셀프 리뷰를 표준화합니다.

사람 리뷰어에게 가기 전에 명백한 결함과 보안 이슈를 걸러, 리뷰 왕복 횟수를 줄이는 것이 목표입니다.

TRIGGER

발동 시점

코드 작성이나 수정 직후, 커밋과 PR 생성 전.

SCOPE

검토 범위

git diff의 변경 파일 중심, 저장소 전수 검사 아님.

OUTPUT

산출 형식

치명, 경고, 제안 3단계와 파일, 라인, 수정 예시.

SAFETY

안전 원칙

읽기 전용, 코드를 직접 고치지 않고 보고만.

Pre-commit
핵심 발동 지점

diff 중심
검토 범위

3단계
우선순위 출력

Read-only
권한 원칙

정의 완성본

시나리오를 파일로 옮기면

.claude/agents/code-reviewer.md

```
---
name: code-reviewer
description: Expert code review specialist. Proactively
  reviews code. Use immediately after writing or modifying code.
tools: Read, Grep, Glob, Bash
model: inherit
memory: project
---
You are a senior code reviewer ensuring high standards
of code quality and security.
When invoked: run git diff, focus on modified files,
begin review immediately.
```

No Edit
읽기 전용 강제

proactively
자동 위임 문구

memory
project 스코프

Bash 포함
git diff 실행용

시스템 프롬프트 심화

체크리스트와 출력 형식을 본문에

system prompt body

```
Review checklist:
- 명확성: 함수와 변수 이름, 중복 코드
- 안전성: 에러 처리, 입력 검증, 시크릿 노출
- 품질: 테스트 커버리지, 성능 고려

Provide feedback organized by priority:
- Critical (must fix): 파일:라인 + 수정 예시
- Warnings (should fix)
- Suggestions (consider improving)

결론에 머지 가능 여부를 한 줄로 판정.
```

체크리스트
검토 항목 고정

파일:라인
위치 특정 강제

수정 예시
실행 가능한 피드백

판정 한 줄
머지 가능 여부

영속 메모리 결합

리뷰할수록 똑똑해지는 리뷰어

LEARNING REVIEWER

memory: project로 이 저장소의 반복 이슈와 컨벤션이 축적됩니다.

팀은 메모리 디렉토리를 버전 관리에 올려 학습까지 공유합니다.

WRITE

축적 지시

본문에 발견 패턴을 메모리에 기록하라는 지시 포함.

READ

선참조 지시

리뷰 전에 과거 패턴을 먼저 확인하라고 호출 시 요청.

SHARE

팀 공유

.claude/agent-memory/ 커밋으로 학습 자체를 공유.

CURATE

정리 주기

MEMORY.md 상한 초과 시 에이전트가 스스로 큐레이션.

project
권장 스코프

반복 이슈
축적 대상

커밋 가능
학습의 팀 자산화

자가 정리
상한 초과 시

호출 경로 3 / GitHub Actions

PR마다 자동 리뷰

.github/workflows/agent-review.yml

```
on: [pull_request]
jobs:
  review:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4   # with fetch-depth: 0
      - run: curl -fsSL https://claude.ai/install.sh | bash
      - run: |
          claude -p "code-reviewer 서브에이전트로 이 PR
            diff 리뷰, 결과를 review.md로 저장" \
            --allowed-tools "Agent,Read,Grep,Glob,Bash,Write"
        env: { ANTHROPIC_API_KEY: ${{ secrets.ANTHROPIC_API_KEY }} }
```

PR 트리거
pull_request 이벤트

정의 동봉
저장소의 agents 사용

review.md
코멘트 게시 소스

Secrets
키는 시크릿 주입

자주 만나는 함정

현장에서 걸리는 지점

함정

- description에 시점이 없어 자동 위임 불발
- Bash 누락으로 git diff 실행 불가
- 출력 형식 미지정, 리뷰가 산문으로 흩어짐
- 저장소 전수 검사로 시간과 토큰 폭증
- CI에서 Agent 도구 미허용으로 위임 실패

솔루션

- Use immediately after... 시점 문구 명시
- tools에 Bash 포함, 훅으로 diff만 허용
- 우선순위 3단계와 파일:라인 형식 강제
- diff 중심 범위를 본문에 명문화
- allowed-tools에 Agent 포함 확인

test-writer 정의

쓰고 실행하고 고치는 워커

.claude/agents/test-writer.md

```
---
name: test-writer
description: Test coverage specialist. Use proactively
  when new code lacks tests or coverage drops.
tools: Read, Write, Edit, Bash, Grep, Glob
model: sonnet
isolation: worktree
maxTurns: 40
---
You are a test automation expert. Workflow: measure
coverage, find gaps, write tests matching project
conventions, run them, fix failures preserving intent.
```

Write 포함
테스트 파일 생성

worktree
격리 실행

maxTurns 40
폭주 상한

의도 보존
실패 수정 원칙

security-scanner 정의

읽기 전용 + PreToolUse 이중 잠금

.claude/agents/security-scanner.md

```
---
name: security-scanner
description: Security specialist. MUST BE USED before
  commits touching auth, payments, or user data.
tools: Read, Grep, Glob, Bash
model: opus
memory: project
permissionMode: dontAsk
hooks: { PreToolUse: [{ matcher: "Bash", hooks:
  [{ type: command, command: "./scripts/ro-guard.sh" }] }] }
---
```

MUST BE USED
강제 위임 문구

opus
판단 난도 상향

dontAsk
무인 자동 거부

ro-guard
Bash 읽기 전용 검증

docs-writer 정의

skills 프리로드가 스타일을 고정

.claude/agents/docs-writer.md

```
---
name: docs-writer
description: Documentation specialist. Use proactively
  when public APIs change or docs drift from code.
tools: Read, Write, Edit, Grep, Glob, Bash
model: sonnet
skills: [docs-style-guide, api-doc-template]
---
You are a technical writer. Read the code first,
document what it actually does, and follow the
preloaded style guide exactly.
```

skills 2종
스타일, 템플릿 주입

코드 우선
실제 동작 기준

Write 포함
문서 파일 생성

drift 문구
괴리 감지 시 발동

migration-bot 정의

안전장치 3중의 실행 워커

.claude/agents/migration-bot.md

```
---
name: migration-bot
description: Large-scale migration specialist. Use for
  library upgrades, import paths, deprecated API swaps.
tools: Read, Write, Edit, Bash, Grep, Glob
model: sonnet
isolation: worktree
maxTurns: 60
memory: project
---
Migrate in small verifiable batches: run build and tests
each batch, record progress to memory, stop on threshold.
```

worktree
안전장치 1 격리

maxTurns 60
안전장치 2 상한

배치 검증
안전장치 3 절차

memory
진행 상황 기록

CHAPTER 02 / PART 09

병렬 오케스트레이션

Teams, Workflows, 그리고 loop과 graph

- Agent Teams: 공유 태스크와 직통 메시지
- Dynamic Workflows: 계획을 스크립트로
- loop, fan-out, fan-in의 실제 구현
- 6대 병렬화 수단 선택 가이드

Agent Teams / 세션들이 팀을 이룬다

실험 기능, 환경변수로 켜야 동작

EXPERIMENTAL, OFF BY DEFAULT

settings.json의 env에 CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS=1을 넣어야 켜집니다.

켜면 lead가 이름을 붙인 서브에이전트가 teammate로 뜨므로, 의도치 않게 팀이 생길 수 있습니다.

LEAD
팀 리드

메인 세션이 리드. 태스크를 만들고 배분하고 결과를 종합.

MATE
팀메이트

각자 독립 세션과 컨텍스트. 대화 이력은 상속하지 않음.

TASKS
공유 태스크

~/.claude/tasks/. 의존 관계 지원, 파일 락으로 선점 경합 방지.

MAIL
메일박스

SendMessage로 팀메이트끼리 직통. 리드를 거치지 않음.

3~5명
권장 팀 규모

in-process
teammateMode 기본값

세션당 1팀
중첩 팀 불가

토큰 급증
인원수만큼 선형 증가

Agent Teams 운용과 한계

현장에서 먼저 부딪히는 여섯 가지

팀메이트 화면 배치	teammateMode: in-process 기본	split pane은 tmux나 iTerm2
정의 재사용	서브에이전트 타입을 지목해 스폰	skills는 미적용, 본문은 추가
품질 게이트	TeammateIdle, TaskCreated, TaskCompleted	exit 2로 되돌려 계속 일하게
권한	리드의 권한 모드를 그대로 물려받음	승인 프롬프트는 리드에 표시
세션 재개	/resume, /rewind로 복원되지 않음	리드에게 재스폰 요청
금지 사항	중첩 팀 불가, 리드 교체 불가	같은 파일 동시 편집 금지

Dynamic Workflows / 계획을 코드로

스크립트가 수십~수백 에이전트를 조율

.claude/workflows/audit-routes.js

```
export const meta = {
  name: 'audit-routes',
  description: 'Audit route handlers for missing auth checks',
}

const found = await agent('List every .ts file under src/routes/.', {
  schema: { type: 'object', required: ['files'], properties: {
    files: { type: 'array', items: { type: 'string' } } } } })
const audits = await pipeline(found.files, file =>
  agent(`Audit ${file} for missing auth checks.`, { label: file }))
return audits.filter(Boolean)
```

agent / pipeline
parallel / phase / log

결과는 변수에
메인 컨텍스트엔 최종만

ultracode
프롬프트 키워드로 발동

/workflows
실행 목록과 저장(s)

loop과 graph는 어떻게 만드나

전용 문법은 없다, 자바스크립트가 문법이다

LOOP

반복

pipeline로 목록 순회, while로 조건 반복. 전용 loop 문법 없음.

BRANCH

분기

if와 else 그대로. 스크립트가 판단을 들고 있음.

FAN-OUT

동시 실행

parallel이 한 묶음을 동시에 띄우고 전부 기다림.

FAN-IN

합성

앞 결과를 변수로 받아 다음 agent 지시문에 넣음.

DEPENDS

의존

await 순서가 곧 의존 관계. dependsOn이나 DAG 문법은 없음.

PHASE

표시용

phase는 진행 화면의 묶음 제목일 뿐, 의존과 무관.

선택 가이드 / 6대 병렬화 수단

누가 계획을 들고 있는가로 갈린다

한 대화 안에서 위임하고 결과만 회수	Subagent	계획은 Claude, 턴 단위
배경 설명이 긴 곁가지를 그대로 맡김	Fork / subtask	이력과 캐시를 상속
무관한 작업을 던져두고 나중에 확인	Agent view	claude agents, 리서치 프리뷰
워커끼리 논의하고 서로를 반박해야 함	Agent Teams	공유 태스크 + 직통 메시지
전수 감사, 500파일 마이그레이션, 교차검증	Workflows	계획을 스크립트가 보유
내가 띄운 세션끼리 발견을 전달	Cross-session	ListAgents + SendMessage

Chapter 2 핵심 요약

열 파트를 여섯 문장으로

CONCEPT
개념
격리 컨텍스트에서 일하고 요약만 돌려주는 세션 내 워커.

DEFINE
정의
Markdown 한 장, 필수는 name과 description, 필드 17종.

FIELDS
필드
tools, model, memory, skills, hooks, isolation의 조합 설계.

DISPATCH
디스패치
자연어, @멘션, --agent 사다리와 병렬, 체이닝, resume.

ADVANCED
고급
fork 기본 모드, 중첩 깊이 3, 동시 20, SendMessage.

PATTERNS
패턴 5종
리뷰어, 테스터, 스캐너, 문서, 마이그레이션의 검증된 골격.

FAQ / 자주 나오는 질문

현장 질문 여섯 가지

Subagent끼리 대화가 되나요	이름이 붙으면 SendMessage 가능	본격 협업은 Teams, Part 9
정의를 고치면 재시작해야 하나요	파일 워처가 수 초 내 반영	새 폴더, --add-dir가 예외
/agents 위저드가 안 보여요	v2.1.198에서 제거됨	Claude에게 생성 위임
비용이 두 배로 들지 않나요	haiku 배분, fork 캐시, 팬아웃 캐시	Part 1, 8 비용 슬라이드
에이전트가 폭주하면요	maxTurns, 훅, 깊이 3, 동시 20	Part 5, 8 안전장치
팀 표준은 어떻게 배포하나요	.claude/agents 커밋 + managed	Part 2 스코프 계층

추가 학습 자료

이 챕터 다음에 읽을 것들

Subagents 공식 문서	code.claude.com/docs/ko/sub-agents	본 덱의 1차 출처
Agent Teams	docs 내 agent-teams 문서	세션 간 협업 확장
Dynamic Workflows	docs 내 workflows 문서	loop과 대규모 팬아웃
Cross-session messaging	docs 내 cross-session-messaging	세션 간 메시지 전달
Hooks, Skills, Worktrees	docs 내 hooks / skills / worktrees	필드 상세 규격
워크샵 코드	github.com/whchoi98/claude-code-workshop	본 과정 실습 자료

Thank you!

Junseok Oh / Sr. Solutions Architect, AWS Korea

atomoh@amazon.com | linkedin.com/in/oh-junseok

Workshop code: github.com/whchoi98/claude-code-workshop