CLAUDE CODE DEEP DIVE WORKSHOP

Chapter 1 – Claude Code, 설치부터 Headless 자동화까지

Ren, Yongzhe
Sr. Solutions Architect
AWS Korea
yongzher@amazon.com

Agenda

01	**설치와 인증**	Native 설치, doctor, 조직 버전 통제, Bedrock 연동과 자격증명 우선순위
02	**기본 사용**	프롬프트 원칙, 슬래시 명령, 디버깅·기능추가·리팩토링 시나리오, 권한 모드와 되감기
03	**도구 시스템**	도구 5개 축, Git·PR 워크플로, 권한 규칙과 샌드박스
04	**CLAUDE.md 및 메모리**	두 메모리 체계, 4계층 구조, rules 분리와 준수율 높이기
05	**워크플로 패턴**	Explore-Plan-Code, 코드 리뷰, 병렬화, 헤드리스 자동화, 비용·컨텍스트 관리
06	**핸즈온 랩**	

CHAPTER 01 / PART 03

Installation

Native 설치가 새 표준, 그리고 AWS 고객을 위한 Bedrock 연동

- 시스템 요구사항과 설치 방법 5종 비교
- Native install: macOS, Linux, WSL, Windows
- Homebrew, WinGet, apt, dnf, apk 저장소
- 릴리스 채널, 버전 고정, 검증과 제거

시스템 요구사항

지원 플랫폼과 하드웨어 (2026.07)

OS	macOS 13+, Windows 10 1809+, Server 2019+	Ubuntu 20.04+, Debian 10+, Alpine 3.19+
하드웨어	4GB+ RAM, x64 또는 ARM64	Apple Silicon 네이티브 지원
셸	Bash, Zsh, PowerShell, CMD	Windows는 CMD도 공식 지원
네트워크	인터넷 연결 필수, HTTPS 443	프록시, 커스텀 CA 지원
의존성	ripgrep 기본 포함, Node.js 불필요	npm 경로 사용 시에만 Node 18+

설치 방법 비교

다섯 가지 경로, 무엇을 선택할까

Native (권장)	curl / irm 원라이너, 자동 업데이트	개인, 팀 기본 선택
Homebrew	claude-code(stable), claude-code@latest	macOS 패키지 일원화
WinGet	winget install Anthropic.ClaudeCode	Windows 표준 관리
apt / dnf / apk	서명된 공식 저장소, stable과 latest 채널	리눅스 플릿 배포
npm (레거시)	@anthropic-ai/claude-code, Node 18+ 필요	기존 파이프라인 호환

Native
공식 권장

Auto-update
Native만 백그라운드

Signed
리눅스 저장소 GPG 서명

2 채널
stable / latest

Native Install / macOS, Linux, WSL

권장 설치, 원라이너로 완료

INSTALL SCRIPT

```
# 설치 (macOS, Linux, WSL 공통)
curl -fsSL https://claude.ai/install.sh | bash

# 프로젝트에서 첫 실행
cd your-project
claude
# 처음 실행 시 브라우저 로그인 안내가 표시됩니다

# 특징
#  Node.js 불필요, 단일 바이너리
#  백그라운드 자동 업데이트 기본 활성
#  403, syntax error 발생 시 troubleshoot-install 문서 참조
```

1 line	No Node	Auto	install.sh
설치 명령	런타임 의존성 없음	백그라운드 업데이트	claude.ai 공식 스크립트

Native Install / Windows

PowerShell과 CMD를 구분

POWERSHELL / CMD

```
# Windows PowerShell (프롬프트가 PS C:\)
irm https://claude.ai/install.ps1 | iex

:: Windows CMD (프롬프트가 C:\)
curl -fsSL https://claude.ai/install.cmd -o install.cmd
install.cmd && del install.cmd

# 판별 팁
#  '&&' is not a valid statement -> PowerShell
#  'irm' is not recognized       -> CMD
```

GIT FOR WINDOWS

- 설치 권장, Bash 도구 활성화
- 없으면 PowerShell 도구로 대체
- WSL 설치라면 불필요
- 경로 지정: CLAUDE_CODE_GIT_BASH_PATH

버전 지정과 채널 선택

특정 버전, stable 채널로 설치

PINNED INSTALL

```
# stable 채널로 설치 (설치 채널이 이후 업데이트 기본값)
curl -fsSL https://claude.ai/install.sh | bash -s stable

# 특정 버전 설치
curl -fsSL https://claude.ai/install.sh | bash -s 2.1.89

# Windows PowerShell에서 버전 지정
& ([scriptblock]::Create((irm https://claude.ai/install.ps1))) 2.1.89

# 채널 개념
#  latest: 릴리스 즉시   stable: 약 1주 지연, 회귀 제외
```

-s stable
채널 지정 인자

-s 2.1.89
버전 고정 인자

채널 승계
설치 채널이 업데이트 기준

2 채널
latest / stable

Docker 컨테이너

격리 실행과 이미지 표준화

DOCKERFILE (EXAMPLE)

```
FROM ubuntu:24.04
RUN apt-get update && apt-get install -y curl git ca-certificates
RUN curl -fsSL https://claude.ai/install.sh | bash
ENV PATH="/root/.local/bin:${PATH}"
WORKDIR /workspace

# 실행 예시 (키는 런타임 주입)
# docker run -it --rm \
#   -e ANTHROPIC_API_KEY \
#   -v $(pwd):/workspace claude-dev claude
```

Isolation
프로세스, FS 격리

Runtime -e
키는 이미지에 미포함

Volume
작업 디렉토리 마운트

Ch.3
엔터프라이즈 이미지 심화

Devcontainer 설정

팀 일관 개발 환경

.devcontainer/devcontainer.json

```
{
  "name": "claude-dev",
  "image": "mcr.microsoft.com/devcontainers/base:ubuntu",
  "features": {},
  "postCreateCommand":
    "curl -fsSL https://claude.ai/install.sh | bash",
  "remoteEnv": {
    "ANTHROPIC_API_KEY": "${localEnv:ANTHROPIC_API_KEY}"
  }
}
```

postCreate
컨테이너 생성 시 설치

localEnv
호스트 키 전달

VS Code
Reopen in Container

Team
환경 일관성 확보

설치 검증

version과 doctor로 상태 확인

VERIFY

```
# 버전 확인
claude --version

# 종합 자가 진단
claude doctor
#  설치 상태, 설정, 최근 업데이트 결과 점검
# f 키를 누르면 발견된 문제를 Claude가 수정

# 즉시 수동 업데이트
claude update
```

--version	doctor	f	update
버전 출력	종합 진단	진단 후 자동 수정	즉시 갱신

자동 업데이트와 릴리스 채널

settings.json으로 채널과 하한 지정

settings.json

```
{
  "autoUpdatesChannel": "stable",
  "minimumVersion": "2.1.100",
  "env": {
    "DISABLE_AUTOUPDATER": "0"
  }
}

# autoUpdatesChannel: latest(기본) 또는 stable
# minimumVersion: 이 값 미만으로는 다운그레이드 금지
# /config 의 Auto-update channel 항목으로도 변경 가능
```

latest
기본 채널

stable
약 1주 지연 채널

minimumVersion
다운그레이드 하한

/config
대화형 변경 경로

조직 차원 업데이트 통제

managed settings의 버전 정책

minimumVersion	업데이트가 이 값 미만 설치 거부	사용자 설정으로 무력화 불가
requiredMinimumVersion	미만 버전은 실행 자체 거부	managed 전용
requiredMaximumVersion	초과 버전 실행 거부, 업데이트 상한	검증된 범위 고정
DISABLE_AUTOUPDATER	백그라운드 확인만 중지	update 명령은 동작
DISABLE_UPDATES	수동 포함 모든 업데이트 차단	자체 배포 채널 조직용

CHAPTER 01 / PART 04

Authentication

구독부터 Bedrock, Gateway까지 6가지 경로

- Claude 구독 OAuth와 사용량 확인
- Console API Key와 보안 수칙
- Bedrock, Vertex AI, Microsoft Foundry
- apps gateway, 우선순위, CI용 장기 토큰

인증 방법 한눈에

여섯 가지 경로 요약

구독 OAuth	claude 실행 후 브라우저 로그인	Pro, Max, Team, Enterprise
Claude Console	Console 계정, API 종량 청구	Claude Code 전용 롤 지원
Amazon Bedrock	AWS 자격증명, IAM 통제	엔터프라이즈 표준 경로
Google Vertex AI	GCP ADC 자격증명	GCP 표준화 조직
Microsoft Foundry	Azure 자격증명	Azure 표준화 조직
apps gateway	사내 IdP SSO, 게이트웨이 토큰	자체 호스팅 중앙 관문

Method 1 / Claude 구독 OAuth

가장 간단한 개인 경로

FIRST LOGIN

```
$ claude
# 브라우저가 열리며 로그인 진행

# 브라우저가 안 열리면
#  c 키: 로그인 URL 클립보드 복사

# WSL2, SSH, 컨테이너처럼 콜백이 막힌 환경
#  브라우저에 표시된 코드를
#  'Paste code here if prompted'에 붙여넣기

$ /logout   # 로그아웃 후 재인증
```

ACCOUNT TYPES

- Pro, Max: 개인 구독
- Team, Enterprise: 조직 초대 계정
- Free 플랜은 사용 불가
- 자격증명은 OS 보안 저장소에 보관

Method 2 / Claude Console

API 종량 청구 조직 경로

CONSOLE PATH

Console 조직에 사용자를 초대하고 롤을 부여하면, 구성원이 Console 계정으로 로그인해 API 청구 기반으로 사용합니다.

STEP 1

조직 초대

Settings, Members에서 대량 초대 또는 SSO 연동.

STEP 2

롤 부여

Claude Code 롤은 Code 전용 키만, Developer 롤은 전체 키 생성.

STEP 3

사용자 로그인

설치 후 Console 자격증명으로 로그인.

BILLING

종량 청구

조직 Console에 사용량 집계, 예산 알림 설정 가능.

2 Roles
Claude Code / Developer

SSO
Console SSO 지원

Usage
조직 단위 집계

Invite
관리자 초대 필수

API Key 환경변수

셸과 컨테이너 환경 통합

ENV SETUP

```
# Bash / Zsh
export ANTHROPIC_API_KEY="sk-ant-..."

# PowerShell
$env:ANTHROPIC_API_KEY = "sk-ant-..."

# direnv (.envrc, 반드시 .gitignore에 추가)
export ANTHROPIC_API_KEY="sk-ant-..."
# Docker / Kubernetes
docker run -e ANTHROPIC_API_KEY ...
kubectl create secret generic claude-key \
  --from-literal=ANTHROPIC_API_KEY=sk-ant-...
```

X-Api-Key
전송 헤더

1회 승인
대화형 최초 사용 시

.gitignore
.envrc 필수 등록

Secret
K8s는 시크릿으로

Method 3 / AWS Bedrock

AWS 자격증명으로 추론 경로 전환

BEDROCK ENV

```
# Bedrock 경로 활성화
export CLAUDE_CODE_USE_BEDROCK=1
export AWS_REGION=ap-northeast-2

# 자격증명은 표준 AWS 체계 그대로
aws sso login --profile dev   # 또는 IAM Role, 환경변수
export AWS_PROFILE=dev

# 실행
claude
# /status 로 활성 공급자와 모델 확인
```

USE_BEDROCK=1	SSO/Role	Region	/status
경로 스위치	표준 AWS 자격증명	모델 제공 리전 확인	활성 경로 검증

Bedrock 모델 해석과 지정

alias 매핑과 상위 모델 사용

opus / sonnet alias	Bedrock에서는 Opus 5, Sonnet 5로 해석	보수적 기본 매핑
상위 모델 사용	전체 모델명 또는 inference profile ARN 지정	claude --model로 전달
기본값 고정	ANTHROPIC_DEFAULT_OPUS_MODEL 등 환경변수	alias가 가리킬 버전 통제
소형 모델	ANTHROPIC_SMALL_FAST_MODEL로 배경 작업 지정	Haiku 계열 권장
리전 전략	모델별 제공 리전 상이, cross-region profile	지연과 가용성 균형

Bedrock IAM 정책

최소 권한 예시

IAM POLICY (JSON)

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "ClaudeCodeBedrock",
    "Effect": "Allow",
    "Action": [
      "bedrock:InvokeModel",
      "bedrock:InvokeModelWithResponseStream"
    ],
    "Resource": "arn:aws:bedrock:*::foundation-model/anthropic.*"
  }]
}
```

2 Actions	anthropic.*	SCP	Ch.3
Invoke + Stream	리소스 범위 한정	조직 가드레일 병행	SSO 매핑 심화

Claude apps gateway

사내 SSO로 로그인하는 자체 호스팅 관문

SELF-HOSTED GATEWAY

게이트웨이가 IdP로 개발자를 인증하고, 추론을 설정된 클라우드로 라우팅합니다.
세션의 유일한 자격증명은 게이트웨이 토큰입니다.

STEP 1

/login SSO

개발자가 사내 IdP로 로그인

›

STEP 2

Gateway

그룹별 모델 접근, 지출 한도 적용

›

STEP 3

Provider

Bedrock, GCP, Foundry로 라우팅

›

STEP 4

Telemetry

OTLP로 사용량과 비용 집계

자격증명 우선순위 복습

충돌 시 무엇이 이기는가

0 Gateway 세션	apps gateway 로그인 상태면 아래 전부보다 우선
1 Cloud 스위치	USE_BEDROCK / VERTEX / FOUNDRY 환경변수
2-3 Token / Key	ANTHROPIC_AUTH_TOKEN, 그다음 ANTHROPIC_API_KEY
4 apiKeyHelper	스크립트 반환 동적 키, 볼트 연동
5-6 OAuth	CLAUDE_CODE_OAUTH_TOKEN, 마지막이 /login 구독

CHAPTER 01 / PART 05

Quick Start

첫 세션부터 세션 관리까지

- claude 명령 기본형과 대화형 REPL
- 첫 프롬프트 작성 요령과 슬래시 명령 지도
- 권한 모드 운용과 resume, branch, rewind

claude 명령 기본형

대화형과 헤드리스, 두 얼굴

COMMAND SHAPES

```
claude                  # 대화형 REPL 시작
claude "질문 또는 작업"      # 초기 프롬프트와 함께 시작
claude -p "작업"           # 헤드리스, 결과만 출력 후 종료

claude --model opus        # 모델 지정 시작
claude --continue          # 최근 세션 이어서
claude --resume            # 세션 선택해 재개

cat data.csv | claude -p "요약" # 파이프 입력
claude --help              # 전체 플래그
```

REPL
기본 대화형

-p
헤드리스 모드

--model
시작 모델 지정

Pipe
stdin 입력 지원

대화형 REPL 입문

첫 세션의 화면 구성

FIRST SESSION

```
$ cd my-project && claude

> 이 프로젝트가 뭘 하는지 설명해줘
# Claude가 Glob, Read로 구조 탐색하며 답변

> src/api의 에러 처리 방식을 알려줘
# 후속 질문은 이전 맥락을 그대로 활용

# 조작 키
#  Esc: 즉시 중단   Esc Esc: 되감기
#  Shift+Tab: 권한 모드 순환
#  Ctrl+R: 명령 이력 검색
```

REPL BASICS

- 프롬프트에 자연어로 지시
- 도구 호출 과정이 실시간 표시
- 대화는 자동 저장, 재개 가능
- / 입력으로 명령 목록 필터

첫 프롬프트 작성 요령

공식 Best Practices 4원칙

DELEGATE, DON'T DICTATE

유능한 동료에게 위임하듯 맥락과 방향을 주고 세부 경로는 맡깁니다.
어떤 파일을 읽을지까지 지시할 필요는 없습니다.

TIP 1
구체적으로

관련 경로, 제약, 참고 패턴을 처음부터 명시해 교정 횟수 절감.

TIP 2
검증 기준 제공

테스트 케이스, 기대 출력 등 스스로 확인할 기준 제공.

TIP 3
탐색과 분리

복잡한 문제는 Plan 모드로 조사 먼저, 코딩은 그 다음.

TIP 4
대화로 교정

완벽한 첫 프롬프트보다 중간 개입과 반복 교정이 빠름.

경로 명시
src/payments/ 처럼

Verify
확인 가능한 기준

Plan 먼저
큰 변경 2단계

Esc
언제든 개입

명확한 vs 모호한 지시

같은 목표, 다른 결과

SPECIFICITY WINS

범위와 완료 기준이 담긴 지시가 1회 성공률을 결정합니다.

모호한 지시 (교정 비용 큼)

- 로그인 버그 고쳐줘
- 테스트 좀 추가해줘
- 코드 정리해줘
- 탐색 비용과 방향 이탈 위험 증가

명확한 지시 (1회 성공률 상승)

- 만료 카드 결제 실패, src/payments/ 토큰 갱신부터 조사
- validateEmail 구현, 3개 케이스 통과 후 테스트 실행
- src/legacy만, 동작 변경 없이 이름 규칙 통일
- 범위와 완료 기준이 함께 전달됨

슬래시 명령 지도

워크플로 단계별 핵심 명령

첫 세션	/init, /memory, /mcp, /agents, /permissions	프로젝트 셋업
작업 중	/plan, /model, /effort, /context, /compact, /btw	진행과 컨텍스트
병렬 실행	/agents, /tasks, /background, /batch	위임과 분산
배포 전	/diff, /code-review, /security-review	품질 게이트
세션 간	/clear, /resume, /branch, /teleport	전환과 재개
문제 시	/rewind, /doctor, /debug, /feedback	복구와 진단

/help, /status, /doctor

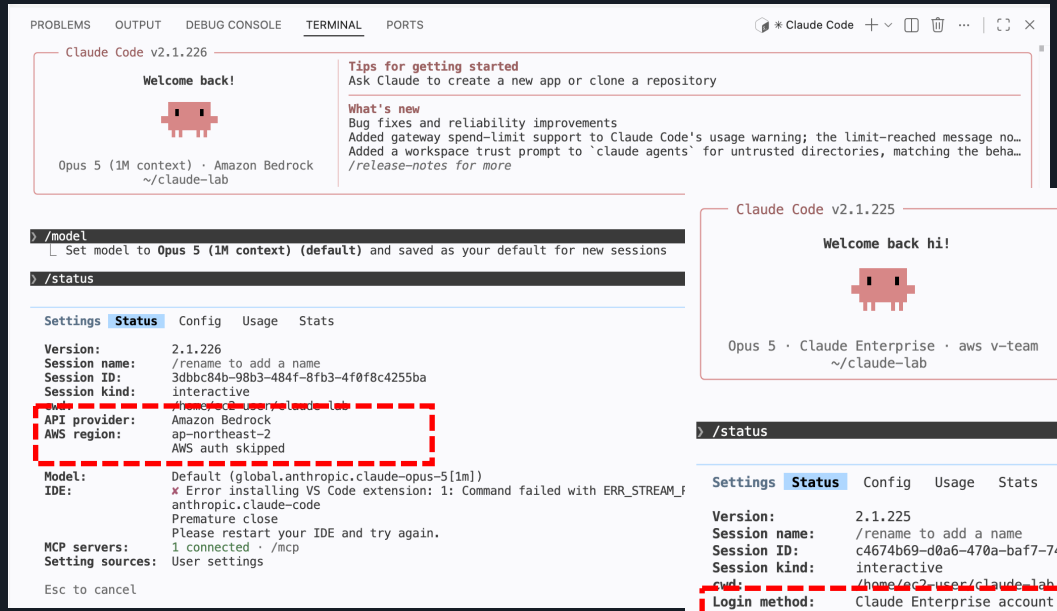
상태 확인 3종 세트

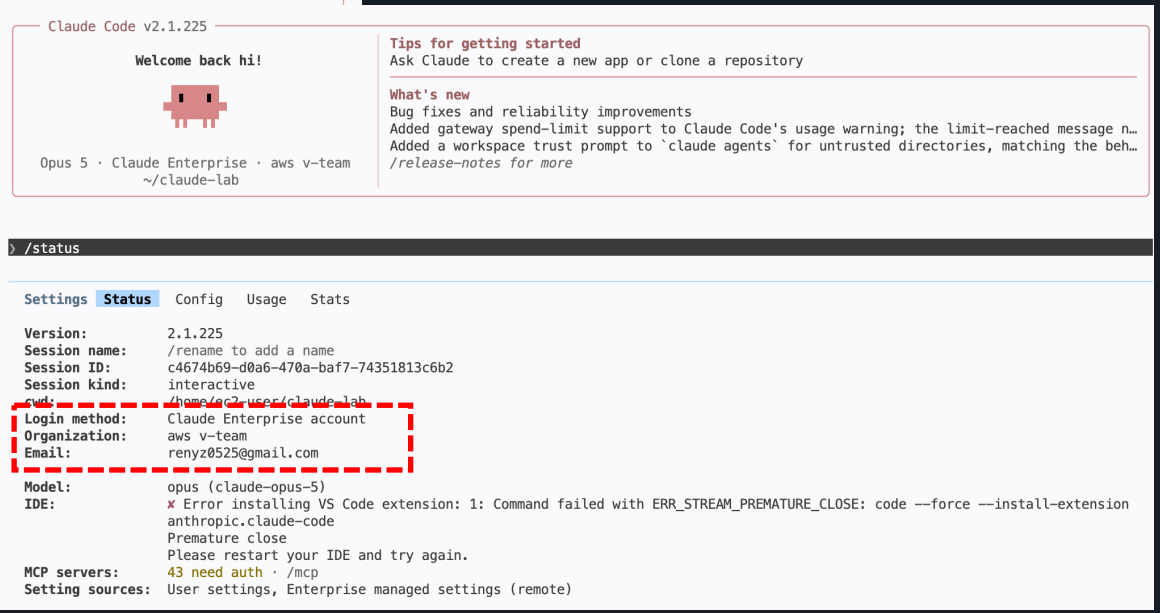
STATUS TRIO

```
> /help
# 사용 가능한 명령 전체와 설명

> /status
# 인증 방법, 공급자, 모델, 계정, 세션 정보

> /doctor
# 설치와 설정 종합 진단, 상태 아이콘 표시
# f 키로 발견된 문제 자동 수정
```

/help	/status	/doctor	f
명령 카탈로그	인증, 모델 확인	종합 진단	자동 수정 트리거

/status 확인 예시





/init 프로젝트 초기화

CLAUDE.md 자동 생성

INIT

```
$ claude
> /init
# 코드베이스를 분석해 CLAUDE.md 초안 생성
#  빌드, 테스트 명령과 컨벤션을 자동 발견
#  AGENTS.md, .cursorrules 있으면 내용 반영
#  기존 CLAUDE.md가 있으면 개선안 제안

# 대화형 다단계 흐름 (선택)
$ CLAUDE_CODE_NEW_INIT=1 claude
> /init
#  CLAUDE.md, skills, hooks 중 무엇을 만들지 질문
```

/init	AGENTS.md	NEW_INIT=1	Part 8
초안 자동 생성	타 도구 설정 흡수	대화형 다단계	CLAUDE.md 심화

/clear vs /compact vs /btw

컨텍스트를 다루는 세 가지 선택

CONTEXT TRIAGE

새 작업이면 clear, 같은 작업의 공간 확보면 compact, 흐름을 깨지 않을 곁가지 질문이면 btw입니다.

/clear

새 대화 시작

컨텍스트 초기화, 이전 대화는 /resume에 보존. 별칭 /new.

/compact

요약해 지속

같은 작업을 이어가며 공간 확보. 초점 지시 전달 가능.

/btw

곁가지 질문

대화 이력에 남지 않는 사이드 질문. 컨텍스트 오염 방지.

작업 전환
clear 선택 기준

공간 부족
compact 선택 기준

잡담, 확인
btw 선택 기준

/rewind
clear 이전 복귀 가능

/context 시각화

윈도우 사용량을 그리드로 확인

CONTEXT VIEW

```
> /context
# 컨텍스트 사용량을 색상 그리드로 표시
#  System, CLAUDE.md, Auto memory, MCP,
#  대화 이력, 도구 출력의 점유 비율
# 과도한 항목에 대한 최적화 제안 표시

> /context all
# 풀스크린 모드에서 항목별 상세 확장

> /mcp
# 서버별 컨텍스트 비용 개별 확인
```

Grid	제안	all	/mcp
색상 점유 시각화	최적화 힌트 포함	상세 확장 인자	서버별 비용

권한 모드 실전 운용

Shift+Tab 순환과 상황별 선택

MODE SWITCHING

모드는 세션 중 언제든 전환됩니다.

큰 변경 전 Plan, 반복 작업엔 Accept Edits, 신뢰 저장소에선 Auto가 기본 감각입니다.

Shift+Tab

Plan

아키텍처 변경, 50+ 파일, 의사결정

›

Shift+Tab

Default

일반 작업 기본값, 승인 흐름 학습

›

Shift+Tab

Accept Edits

소규모 반복 수정, 빠른 루프

›

Shift+Tab

Auto

신뢰 저장소, 분류기 안전 검사

세션 관리 / resume, continue

대화를 이어가는 방법

SESSIONS

```
claude --continue      # 이 디렉토리의 최근 세션 재개
claude --resume        # 세션 목록에서 선택 재개
claude --from-pr 123   # PR을 만든 세션 찾아 재개

> /resume              # 세션 안에서 피커 열기
# 현재 워크트리 기준, 단축키로 범위 확장

> /clear 결제모듈-리팩토링 # 이전 대화에 이름 붙여 보관

# 세션은 ~/.claude/projects/ 에 JSONL로 저장
```

--continue
최근 세션 즉시

--resume
피커에서 선택

--from-pr
PR로 세션 추적

JSONL
로컬 저장 형식

세션 분기 / branch와 fork

다른 방향을 시험하는 두 방법

BRANCH VS FORK

branch는 내가 사본으로 갈아타 다른 방향을 시도하고, fork는 사본을 백그라운드 서브에이전트에게 맡깁니다.

/branch [이름]

- 현 시점에서 대화 사본 생성
- 나는 사본으로 전환해 작업
- 원본은 /resume으로 복귀 가능
- 실험적 방향 전환에 적합

/fork <지시>

- 전체 맥락을 물려받은 서브에이전트 생성
- 나는 원본에서 계속 진행
- 완료되면 결과가 내 대화로 회수
- 곁가지 작업 위임에 적합

/rewind 체크포인트 복구

코드와 대화를 시점 단위로 롤백

REWIND

```
# 빠른 되감기
Esc Esc          # 최근 체크포인트로

> /rewind
# 체크포인트 목록에서 시점 선택
#  코드만 / 대화만 / 둘 다, 복구 범위 선택
#  /clear 이전 대화로도 복귀 가능

# 한계
#  파일 변경만 복구, DB나 배포 등
#  외부 부작용은 대상 아님
```

Esc Esc
즉시 되감기

3 범위
코드, 대화, 둘 다

clear 복귀
이전 대화 복원

Files only
외부 부작용 제외

/usage 비용과 한도

세션과 플랜 소비 확인

USAGE

```
> /usage      # /cost 는 별칭
# 플랜 한도 대비 사용량과 리셋 시점
# 스킬, 서브에이전트, MCP 서버별 소비 분해

# 소비 절감 루틴
#  /model haiku     가벼운 작업 전환
#  /compact         이력 요약으로 토큰 절감
#  /context         과소비 항목 식별

# 조직 집계는 OTel 텔레메트리로 (챕터 3)
```

/usage
한도와 분해 뷰

haiku
경량 전환 절감

/compact
이력 요약 절감

OTel
조직 단위 집계

CHAPTER 01 / PART 06

Core Capabilities

내장 도구를 손끝까지

- 파일, 검색, 실행, 웹, 인텔리전스 5축 도구
- LSP, Monitor, AskUserQuestion 등 신규 도구
- Git 워크플로와 gh CLI 통합
- 권한 규칙 문법과 샌드박스 가드레일

도구 카테고리 지도

이 파트의 학습 경로

AXIS 1
File Ops
Read, Edit, Write, NotebookEdit

›

AXIS 2
Search
Glob, Grep

›

AXIS 3
Execution
Bash, PowerShell, Git

›

AXIS 4
Web
WebSearch, WebFetch

›

AXIS 5
Intel + Orchestration
LSP, Monitor, Agent, Ask

도구 이름 문자열은 권한 규칙, 서브에이전트 tools 목록, Hook 매처에 그대로 사용됩니다

5 Axes
학습 경로

정확한 이름
규칙, 훅에 재사용

승인 구분
부작용 도구만 확인

MCP
외부 도구 확장 축

Read / 파일 읽기와 멀티모달

텍스트, 이미지, 노트북까지

READ PATTERNS

```
> src/auth/session.ts 읽고 만료 로직 설명해줘
# 큰 파일은 필요한 라인 범위만 선택적으로 읽음

> 이 스크린샷의 에러를 봐줘 (이미지 드래그, 붙여넣기)
# PNG, JPG 등 이미지를 직접 이해

> design.pdf의 3장 요구사항을 정리해줘
# PDF 문서 읽기 지원

# 특성: 승인 불필요, 접근은 작업 디렉토리 기준
#  외부 경로는 --add-dir 또는 /add-dir로 확장
```

No approve
읽기는 승인 불필요

Image/PDF
멀티모달 입력

Range
라인 범위 선택 읽기

--add-dir
외부 경로 확장

Edit / 정밀 수정

diff 승인 기반의 안전한 변경

EDIT PATTERNS

```
> session.ts의 TTL 상수를 3600으로 바꾸고, 이 값을 참조하는 모든 곳을 함께 정리해줘

# Edit 승인 화면
#  변경 전후 diff 표시
#  y 수락 / n 거절 / e 직접 수정

# 멀티 파일 수정
#  관련 파일들을 연속 Edit로 일괄 처리
#  각 편집 전 스냅샷 자동 저장 (rewind 가능)
```

diff 승인
변경 검토 후 수락

Multi-file
연속 편집 일괄 처리

Snapshot
편집 전 자동 저장

Accept Edits
모드로 자동화 가능

Write와 NotebookEdit

신규 파일과 Jupyter 셀

WRITE / NOTEBOOK

> tests/checkout.test.ts 파일을 새로 만들어 만료 카드 케이스 3개를 작성해줘
Write: 신규 파일 생성, 승인 필요

> analysis.ipynb의 2번 셀을 pandas 2.x 문법으로 업데이트하고 실행 결과도 갱신해줘
NotebookEdit: 셀 단위 수정

실행 가능한 파일 생성 시
acceptEdits 모드에서도 확인 프롬프트 표시

Write
신규 파일 생성

NotebookEdit
Jupyter 셀 수정

승인 필요
두 도구 공통

실행 파일 주의
추가 확인 동작

Bash / 명령 실행

포그라운드와 백그라운드

BASH PATTERNS

```
> npm run build 실행하고 오류 있으면 고쳐줘
# 명령별 승인, allow 규칙으로 자주 쓰는 것 통과

> 개발 서버를 백그라운드로 띄우고 로그에 에러가 보이면 알려줘
# 장시간 프로세스는 백그라운드 실행
#  이후 출력 감시는 Monitor 도구와 연계

# 세션 특성
#  작업 디렉토리 유지, 타임아웃 존재
#  위험 명령은 deny 규칙으로 사전 차단
```

승인 필요
Bash 기본 정책

Background
장시간 프로세스

allow 규칙
반복 명령 통과

deny
위험 명령 차단

PowerShell 도구

Windows 네이티브 셸 지원

POWERSHELL

```
# Git for Windows가 없는 네이티브 Windows
#  셸 명령이 PowerShell 도구로 실행됨

> Get-Process에서 메모리 상위 5개 보여줘
> 이 폴더의 오래된 로그 파일을 정리하는 PowerShell 스크립트를 만들어 실행해줘

# Git Bash가 있는 환경에서의 선택적 사용
$env:CLAUDE_CODE_USE_POWERSHELL_TOOL = 1
# 권한 규칙에서는 PowerShell(...) 로 매칭
```

Native Win
기본 셸 도구

Opt-in
Git Bash 병행 시 선택

PowerShell()
권한 규칙 표기

GPO 친화
Windows 운영 자동화

Grep과 Glob / 검색

내용 검색과 파일 패턴

SEARCH PATTERNS

> processPayment를 호출하는 곳을 전부 찾아줘
Grep: ripgrep 기반 내용 검색, 정규식 지원

> src 아래 *.test.ts 파일 목록 보여줘
Glob: 파일명 패턴 매칭

조합 패턴
> deprecated 주석이 달린 export 함수들을 찾아 사용처 개수와 함께 표로 정리해줘
Grep + Glob + Read를 스스로 조합
두 도구 모두 승인 불필요, 탐색이 빠른 이유

ripgrep
Grep 엔진

Regex
정규식 지원

No approve
검색 승인 불필요

조합
탐색 도구 자동 연계

LSP / 코드 인텔리전스

언어 서버 기반의 정적 이해

LANGUAGE SERVER TOOL

편집 직후 타입 오류와 경고를 확인하고, 정의로 이동하며, 참조를 찾습니다.
코드 인텔리전스 플러그인으로 언어를 확장합니다.

USE 1

편집 후 진단

Edit 직후 타입 오류를 확인해 즉시 재수정 루프.

USE 2

정의 이동

심볼의 정의 위치로 이동해 정확한 컨텍스트 확보.

USE 3

참조 찾기

변경 영향 범위를 참조 목록으로 정량 파악.

SETUP

플러그인

언어별 code intelligence 플러그인 설치로 활성.

Post-edit
수정 직후 진단

Definitions
정의, 참조 탐색

Plugins
언어별 확장

정확도
텍스트 검색 보완

WebSearch와 WebFetch

외부 지식을 세션 안으로

WEB PATTERNS

```
> 이 에러 메시지 최근 알려진 원인 검색해줘: ERR_OSSL_EVP_UNSUPPORTED
# WebSearch: 웹 검색 결과 요약

> https://code.claude.com/docs/en/hooks 읽고 PreToolUse 입력 형식을 정리해줘
# WebFetch: URL 콘텐츠 조회

# 통제
#  WebFetch(domain:docs.example.com) 형식으로
#  권한 규칙에서 도메인 단위 allow, deny
```

Search
최신 정보 조회

Fetch
문서 원문 반영

domain:
도메인 단위 통제

프록시
사내망 규칙 준수

Monitor 도구

출력을 지켜보다 반응하는 에이전트

MONITOR

> npm run dev를 Monitor로 띄우고, 로그에 UnhandledPromiseRejection이 보이면 원인 파일을 찾아서 바로 고쳐줘

```
# Monitor 동작
#  백그라운드 명령의 출력 라인을 실시간 수신
#  각 라인이 이벤트로 Claude에게 전달
#  WebSocket 연결의 수신 메시지도 이벤트화

# 활용: dev 서버 감시, 파일 변경 폴링,
#  장시간 배치의 진행 관찰과 자동 대응
```

Line events
출력 라인 단위 반응

WebSocket
수신 메시지 이벤트

승인 필요
실행 계열 정책

Loop 연계
관찰 후 자동 대응

Agent 도구 / 서브에이전트

챕터 2 미리보기

DELEGATION PRIMITIVE

Agent 도구는 독립 컨텍스트를 가진 서브에이전트를 생성합니다.
곁가지 작업이 본 대화를 부풀리지 않고, 완료 시 요약만 돌아옵니다.

WHY
컨텍스트 격리

탐색성 작업의 대량 출력이 메인 대화를 오염하지 않음.

HOW
자동, 명시 호출

작업 성격에 따라 자동 위임, 또는 명시적으로 지정 호출.

SCALE
병렬 실행

독립 작업 여러 개를 동시에, /batch는 워크트리 격리까지.

NEXT
챕터 2

맞춤 서브에이전트 정의와 5대 실전 패턴을 심화.

Isolated
독립 컨텍스트 윈도우

Summary
완료 시 요약 회수

Parallel
동시 실행 가능

Ch.2
심화 챕터

AskUserQuestion과 스케줄 도구

묻는 에이전트, 예약하는 에이전트

INTERACTIVE + SCHEDULED

모호하면 객관식으로 묻고, 반복 작업은 세션 안에서 예약합니다.

AskUserQuestion

1. 요구사항 모호 시 객관식 질문 제시
2. 잘못된 가정으로 달리는 것을 방지
3. 선택지가 곧 설계 옵션 정리
4. 승인 불필요, 흐름 자연 통합

CronCreate / List / Delete

1. 세션 내 반복, 일회성 프롬프트 예약
2. /loop로 폴링형 반복 실행
3. resume 시 미만료 예약 복원
4. 머신 독립 예약은 Routines

Git 워크플로 / 일상

상태 파악부터 스테이징까지

GIT DAILY

> 지금 변경사항 요약해줘
git status, git diff를 읽고 자연어 요약

> 이 변경을 논리 단위로 나눠서 스테이징해줘
관련 파일끼리 묶어 git add 제안

> main과 충돌났어, 해결해줘
충돌 마커를 읽고 양쪽 의도를 병합

Claude는 git 상태를 항상 인지
현재 브랜치, 미커밋 변경, 최근 커밋 이력

status/diff
상태 자동 인지

논리 단위
스테이징 분할

Conflict
충돌 의도 병합

History
최근 커밋 참조

커밋 메시지와 PR 생성

이력을 읽고 컨벤션대로

COMMIT / PR

```
> 설명적인 메시지로 커밋해줘
# 최근 커밋 스타일을 읽어 컨벤션 준수
#  conventional commits 사용 저장소면 그 형식으로

> 이 브랜치로 PR 만들어줘. 리뷰어가 봐야 할 변경 요점과 테스트 방법 포함해서.
# 브랜치 push, PR 본문 작성까지 일괄

> /code-review --comment
# 리뷰 결과를 PR 인라인 코멘트로 게시
```

Convention
저장소 스타일 준수

PR 본문
요점과 테스트 방법

--comment
인라인 코멘트 게시

Ship loop
커밋부터 리뷰까지

gh CLI 통합

GitHub 작업을 대화로

GH INTEGRATION

```
# gh CLI가 설치되어 있으면 그대로 활용

> 나에게 할당된 열린 이슈 보여줘
# gh issue list --assignee @me

> 123번 이슈 읽고 이 저장소에서 재현되는지 확인해줘
# gh issue view 123 후 코드 탐색

> CI 실패한 최근 PR의 로그 분석해줘
# gh run list, gh run view --log 조합

# 인증은 gh auth login 상태를 그대로 사용
```

gh
설치 시 자동 활용

Issues
조회와 재현 확인

CI logs
실패 원인 분석

gh auth
기존 인증 재사용

권한 규칙 문법

settings.json의 allow, ask, deny

.claude/settings.json

```
{
  "permissions": {
    "allow": ["Bash(npm test)",
      "Bash(npm run lint:*)", "Read(./src/**)"],
    "ask": ["Bash(git push:*)"],
    "deny": ["Read(./.env*)",
      "Bash(rm -rf:*)", "WebFetch"]
  }
}
```

Tool(pattern)
규칙 기본형

:*
접두 와일드카드

구체성 우선
충돌 시 해석 원칙

deny 최우선
차단이 항상 승리

도구별 권한 매트릭스

무엇이 승인을 요구하는가

Read, Glob, Grep	승인 불필요	탐색 속도의 원천
Edit, Write, NotebookEdit	승인 필요	diff 검토 후 수락
Bash, PowerShell, Monitor	승인 필요	allow 규칙으로 통과 가능
WebSearch, WebFetch	규칙 기반	도메인 단위 통제 권장
Agent, AskUserQuestion	승인 불필요	위임과 질문은 안전 동작

CHAPTER 01 / PART 08

CLAUDE.md & Memory

세션을 넘어 지식을 잇는 두 시스템

- CLAUDE.md와 Auto memory의 분업
- 4계층 메모리와 로딩 순서
- Import 문법과 .claude/rules 디렉토리
- AGENTS.md 상호운용과 실전 예시

두 가지 메모리 시스템

내가 쓰는 것과 Claude가 쓰는 것

COMPLEMENTARY MEMORY

둘 다 매 세션 시작 시 로드됩니다.

지시와 규칙은 CLAUDE.md에, Claude가 발견한 학습은 Auto memory에 쌓입니다.

CLAUDE.md (사용자 작성)

- 코딩 표준, 워크플로, 아키텍처 지시
- 프로젝트, 사용자, 조직 스코프
- 버전 관리로 팀과 공유
- 지시가 반복되면 여기에 기록

AUTO MEMORY (Claude 작성)

- 빌드 명령, 디버깅 인사이트 자동 축적
- 저장소 단위, 워크트리 간 공유
- 매 세션 첫 200줄 또는 25KB 로드
- 교정할수록 알아서 똑똑해짐

메모리 계층 구조

넓은 스코프에서 좁은 스코프로

1 Managed	macOS /Library/Application Support/ClaudeCode, Linux /etc/claude-code 의 CLAUDE.md, 조직 정책
2 User	~/.claude/CLAUDE.md, 모든 프로젝트에 적용되는 개인 선호
3 Project	./CLAUDE.md 또는 ./.claude/CLAUDE.md, 버전 관리로 팀 공유
4 Local	./CLAUDE.local.md, .gitignore 대상, 개인 프로젝트 노트

로딩 순서와 병합

디렉토리 트리를 걸어 올라가며

HOW CLAUDE.MD LOADS

```
# foo/bar/ 에서 claude 실행 시
# 상위로 걸어 올라가며 발견한 파일을 모두 병합

/etc/claude-code/CLAUDE.md     (조직)
~/.claude/CLAUDE.md            (사용자)
foo/CLAUDE.md                  (상위)
foo/bar/CLAUDE.md              (작업 디렉토리)
foo/bar/CLAUDE.local.md        (개인, 마지막)

# 하위 디렉토리의 CLAUDE.md는 그 폴더의
# 파일을 읽을 때 온디맨드 로드
# HTML 주석은 컨텍스트 주입 전에 제거됨
```

병합
덮어쓰기 아님

가까운 순 뒤
작업 위치가 마지막

On-demand
하위 폴더 파일

주석 제거
HTML 코멘트 무료

Project Memory 실전

./CLAUDE.md 표준 골격

./CLAUDE.md

```
# MyService

## Commands
- Build: npm run build
- Test: npm test (커밋 전 필수)
- Lint: npm run lint:fix
## Conventions
- 들여쓰기 2칸, 세미콜론 사용
- API 핸들러는 src/api/handlers/ 에 위치
- 에러는 AppError 클래스로 래핑
## Architecture
- 인증은 src/auth, 세션은 Redis 보관
```

200줄 이하
권장 상한

검증 가능
구체적 문장

헤더 구조
스캔 가능성

팀 공유
버전 관리 포함

.claude/rules 디렉토리

주제별 파일로 규칙 분리

RULES LAYOUT

```
your-project/
  .claude/
    CLAUDE.md          # 본체 지시
    rules/
      code-style.md    # 스타일 규칙
      testing.md       # 테스트 컨벤션
      security.md      # 보안 요구사항
      frontend/        # 하위 폴더 재귀 탐색
        components.md

# paths 없는 규칙은 세션 시작 시 로드
# 심볼릭 링크 지원: 사내 표준 공유 가능
```

rules/
주제별 분리

재귀 탐색
하위 폴더 포함

Symlink
사내 표준 링크

~/.claude/rules
개인 전역 규칙

Auto Memory 운용

Claude가 스스로 적는 학습 노트

MEMORY.md

작업 중 발견한 빌드 명령, 프로젝트 패턴, 사용자 교정 사항을 Claude가 저장소 단위로 기록합니다.
매 세션 앞부분이 자동 로드됩니다.

WRITE	LOAD	SCOPE	CONTROL
자동 축적	로드 상한	저장소 단위	관리
교정과 발견이 자동 기록, 별도 작성 노력 불필요.	첫 200줄 또는 25KB, 초과분은 필요 시 조회.	워크트리 간 공유, 프로젝트별 독립.	내용 확인과 정리 요청 가능, 서브에이전트도 자체 보유.

MEMORY.md	200줄/25KB	Repo 단위	지시 아님
저장 파일	세션 로드 상한	워크트리 공유	학습 기록 성격

/memory와 # 단축

세션 안에서 메모리 다루기

MEMORY COMMANDS

```
> /memory
# 로드된 메모리 파일 목록과 편집 진입

> # 커밋 전에는 반드시 npm test 실행
# 문장 앞 # 은 빠른 메모리 추가 단축
#  어느 파일에 넣을지 선택 프롬프트 표시

> /init
# 초안 생성 또는 기존 파일 개선안

# 반복해서 말하게 되는 지시는
# 그 순간 # 으로 바로 박제하는 습관
```

/memory
목록과 편집

#
빠른 추가 단축

파일 선택
스코프 지정 프롬프트

습관화
반복 지시 즉시 기록

작성 베스트 프랙티스

지켜지는 메모리의 조건

CONTEXT, NOT CONFIG

메모리는 강제 설정이 아니라 컨텍스트입니다.
구체적이고 간결하며 일관될수록 잘 지켜집니다.

SIZE	STRUCT	SPECIFIC	CONSIST
200줄 이하	헤더와 불릿	검증 가능하게	모순 제거
길수록 준수율 하락, 넘치면 rules와 skills로 분리.	스캔 가능한 구조, 밀집 문단 지양.	2칸 들여쓰기처럼 확인 가능한 표현 사용.	상충 규칙은 임의 선택 유발, 주기적 정리.

짧게
준수율과 반비례 길이

구체적
모호함 제거

일관성
상충 규칙 청소

Hook
강제 필요 시 대안

CHAPTER 01 / PART 09

Workflow Patterns

반복 가능한 실무 패턴 10선

- Explore-Plan-Code와 TDD
- code-review, 멀티 에이전트, 비주얼 워크플로
- Headless, 파이프라인, CI, Routines
- goal 지속 실행과 비용, 컨텍스트 전략

Pattern 1 / Explore-Plan-Code

가장 기본이 되는 3단계 리듬

EPC RHYTHM

탐색으로 이해를 만들고, 계획으로 방향을 합의한 뒤, 코드로 실행합니다.
큰 변경일수록 단계 분리 효과가 큽니다.

STEP 1

Explore

관련 코드 읽기, 흐름 추적, 아직 코딩 금지

›

STEP 2

Plan

Plan 모드로 접근 설계, 계획 협상과 승인

›

STEP 3

Code

승인된 계획대로 구현, 테스트로 검증

Pattern 2 / TDD Workflow

실패하는 테스트가 곧 명세

TDD LOOP

```
> validateCoupon 함수를 TDD로 만들자.
  1) 먼저 실패하는 테스트 작성:
     만료 쿠폰 거부, 중복 사용 거부,정상 쿠폰 할인율 반환
  2) 테스트 실행해서 실패 확인
  3) 통과할 최소 구현 작성
  4) 재실행으로 전부 통과 확인 후 리팩토링

# 기대 출력을 명세로 제공하는 것이 핵심
# mock 데이터가 아닌 실제 실행 결과로 검증
```

Red 먼저
실패 확인 단계

명세 = 테스트
케이스가 요구사항

최소 구현
과설계 방지

Green 후 정리
리팩토링 순서

Pattern 3 / Code Review

/code-review 명령 계열

REVIEW COMMANDS

```
> /code-review            # 현재 diff 리뷰
> /code-review high       # 노력 수준 지정
> /code-review --fix      # 발견 사항 자동 수정
> /code-review --comment  # PR 인라인 코멘트 게시
> /code-review ultra      # 클라우드 멀티에이전트 심층

> /simplify              # 버그 탐색 없이 정리만
> /security-review       # 보안 관점 읽기 전용 검사

# 머지 전 로컬 게이트로 습관화
```

--fix
결과 자동 적용

ultra
클라우드 심층 리뷰

/simplify
정리 전용 모드

머지 전
로컬 품질 게이트

Pattern 4 / Multi-Agent와 /batch

큰 변경을 병렬 단위로

PARALLEL SCALE

```
# 곁가지 위임
> /fork 이 변경의 문서 업데이트를 맡아줘

# 대규모 분할 실행
> /batch src/ 전체를 Solid에서 React로 마이그레이션
# 동작:
#  1. 코드베이스 조사, 5-30개 독립 단위로 분해
#  2. 계획 승인 후 단위별 서브에이전트 생성
#  3. 각자 격리된 git worktree에서 구현과 테스트
#  4. 단위별 PR 오픈

> /tasks    # 백그라운드 작업 현황
```

/batch	Worktree	PR each	Ch.2
5-30 단위 분해	단위별 격리	리뷰 가능한 산출	설계 심화 챕터

Pattern 5 / Visual Workflow

스크린샷이 명세가 되는 흐름

VISUAL LOOP

```
> (디자인 시안 이미지 붙여넣기)
  이 시안대로 ProfileCard 컴포넌트를 구현해줘.
  구현 후 스토리북 스크린샷과 시안을 비교해서 다른 부분을 스스로 찾아 고쳐줘.

# 프론트엔드 검증 강화
> /chrome 연결 후:
> localhost:3000/profile 열어서 콘솔 에러 확인, 버튼 클릭했을 때 모달 동작도 검증해줘
```

이미지 입력
시안이 곧 명세

자가 비교
결과와 시안 대조

Chrome
실동작 검증

UI 루프
구현, 확인, 수정

Pattern 6 / Headless 자동화

-p 플래그로 스크립트 편입

HEADLESS BASICS

```
# 기본형
claude -p "package.json에서 outdated 의존성 요약"

# 파이프와 리다이렉트
tail -200 app.log | claude -p "이상 징후 보고"
claude -p "이 저장소 의존성 분석" > deps-report.md

# 도구 제한으로 안전한 무인 실행
claude -p "보안 이슈 스캔" \
  --allowed-tools "Read,Grep,Glob" \
  --permission-mode plan
claude -p "간단 분류 작업" --model haiku   # 비용 절감
```

-p	allowed-tools	permission-mode	haiku
비대화 실행	도구 화이트리스트	무인 안전선	배치 비용 절감

Pattern 7 / Pipeline과 JSON

구조화 출력으로 후처리

JSON OUTPUT

```
claude -p "현재 디렉토리 스택 분석" \
  --output-format json | jq '.result'

# result 메시지에 포함되는 것
#  result: 최종 응답 텍스트
#  modelUsage: 실제 사용 모델
#  usage: 토큰 사용량, 비용 근거
#  session_id: 후속 --resume 연계

# 실시간 이벤트 스트림
claude -p "..." --output-format stream-json
# 이벤트 단위 파싱으로 진행 표시 구현
```

json	stream-json	jq	session_id
구조화 결과	이벤트 스트림	표준 후처리 도구	체인 실행 연계

Pattern 8 / CI 통합

GitHub Actions 최소 예시

.github/workflows/claude-review.yml

```
on: [pull_request]
jobs:
  review:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - run: curl -fsSL https://claude.ai/install.sh | bash
      - run: |
          claude -p "이 PR의 diff를 리뷰하고 위험을 보고" \
            --allowed-tools "Read,Grep,Glob,Bash(git diff:*)"
        env:
          ANTHROPIC_API_KEY: ${{ secrets.ANTHROPIC_API_KEY }}
```

Actions
공식 액션도 제공

Secrets
키는 시크릿 주입

읽기 위주
CI 도구 제한

Ch.5
패턴 심화 챕터

Pattern 9 / Routines 예약 실행

머신이 꺼져도 도는 자동화

ROUTINES

```
> # 예약 실행 — 세션 유지형 (이 버전에서 동작 확인됨)
  > 평일 09:00에 열린 PR 리뷰 상태를 요약해서 Slack #dev로 보내줘
  # → CronCreate 도구로 등록됨 (슬래시 명령 아님, 자연어로 요청)
  # REPL이 idle일 때 발화 → 세션이 열려 있어야 함
  # durable: true 지정 시 .claude/scheduled_tasks.json에 저장돼 재시작 후에도 생존

  # 등록된 예약 확인

  > 예약된 작업 보여줘      # CronList

# 세션 내 반복은 /loop (머신 필요)
> /loop 10m CI 상태 확인하고 실패하면 원인 요약
```

CronCreate	Managed	이벤트	/loop
Routine 생성	관리형 인프라 실행	API, GitHub 트리거	세션 내 반복 폴링

Pattern 10 / goal 지속 실행

조건이 참이 될 때까지

GOAL

```
> /goal 전체 테스트 스위트가 통과하고 lint 경고가 0이 될 때

# 동작
#  Claude가 턴을 넘겨도 목표 달성까지 계속 작업
#  각 턴 종료 시 조건 충족 여부 스스로 점검

> /goal          # 현재 목표 확인
> /goal clear    # 목표 조기 해제

# Fable 5와 조합하면 장시간 자율 작업에 최적
```

/goal
완료 조건 선언

Cross-turn
턴 경계 지속

자가 점검
조건 충족 확인

Fable 5
장기 자율 조합

비용 관리 전략

모델, 컨텍스트, 캐시의 3축

COST LEVERS

비용은 모델 선택, 컨텍스트 크기, 캐시 적중의 함수입니다.

/usage 분해 뷰로 소비처를 찾고 세 레버를 조정합니다.

MODEL

작업별 모델

일상은 sonnet, 난제만 opus, 배치는 haiku로 배분.

CONTEXT

가볍게 유지

clear, compact, 경로 스코프 rules로 윈도우 최소화.

CACHE

캐시 친화

잦은 모델 전환과 세션 중 CLAUDE.md 수정은 캐시 미스 유발.

MONITOR

관측

/usage 분해와 OTel로 팀 단위 소비 추적.

3 Levers	/usage	Cache miss	OTel
모델, 컨텍스트, 캐시	소비처 분해	모델 전환 비용	팀 단위 추적

컨텍스트 관리 전략

언제 비우고 언제 요약하는가

CONTEXT DISCIPLINE

작업 경계마다 정리하는 습관이 긴 세션의 품질을 지킵니다.

언제 무엇을

- 작업 전환: **/clear** (+이름)
- 같은 작업 지속: **/compact** 초점 지시
- 곁가지 질문: **/btw**
- 탐색 위임: 서브에이전트로 격리
- 상태 점검: **/context** 주기 확인

영속화 원칙

- 대화 초반 지시는 소실 가능
- 영속 규칙은 **CLAUDE.md**로
- Compact Instructions 섹션 활용
- 반복 절차는 skill로 승격
- 경로 한정 지식은 rules로

토큰 효율 베스트 프랙티스

같은 결과를 더 적은 토큰으로

TIP 1

경로 지목

저장소 전체 탐색 대신 관련 경로를 직접 지목.

TIP 2

출력 형식 지정

표로, 다섯 줄로 등 출력 길이를 명시적으로 통제.

TIP 3

MCP 다이어트

안 쓰는 서버 해제, /mcp 로 서버별 비용 점검.

TIP 4

메모리 다이어트

CLAUDE.md 200줄 유지, rules 경로 스코프 분리.

지목
탐색 토큰 절감

형식
출력 토큰 통제

/mcp
서버 비용 가시화

200줄
메모리 상한 습관

자주 마주치는 안티 패턴

비효율의 신호들

안티 패턴

- 한 세션에 서로 다른 작업 뒤섞기
- 거대 파일 통째로 붙여넣기
- 검증 기준 없는 막연한 지시
- 실패한 방향을 수동 재타이핑으로 반복
- 모든 작업을 opus로

교정

- 작업 경계마다 /clear
- 경로 지목과 라인 범위 활용
- 테스트, 기대 출력 명세 동봉
- /rewind로 시점 복귀 후 재지시
- 난이도별 모델 배분

추가 학습 자료

이 챕터 다음에 읽을 것들

공식 문서	code.claude.com/docs/ko/overview	본 덱의 1차 출처
Best Practices	docs 내 best-practices 문서	패턴의 원전
GitHub 저장소	github.com/anthropics/claude-code	이슈, 릴리스 노트
Bedrock 안내	aws.amazon.com/bedrock	AWS 경로 심화
MCP 표준	modelcontextprotocol.io	챕터 4 예습
워크샵 코드	github.com/whchoi98/claude-code-workshop	본 과정 실습 자료

Thank you!

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