

# Claude Code Agent Teams 완전 도입 가이드 — 멀티에이전트로 개발 생산성 극대화

## Claude Code의 Agent Teams(멀티에이전트)를 실전 도입해보자

### 왜 지금 Agent Teams인가?

지난 2월 5일, **Claude Opus 4.6**이 공개되었습니다.

OpenAI 역시 같은 날 GPT-5.3-Codex를 출시하며 AI 기술 경쟁은 더욱 치열해지고 있습니다.

이번 글에서는 Claude Code의 핵심 신기능인 **Agent Teams**의 개념과 단계별 도입 방법을 정리합니다.

### 사전 준비

- WSL2 (Ubuntu) 환경
- Claude Code 설치 완료

## Agent Teams란?

Agent Teams는 여러 Claude Code 인스턴스를 하나의 팀으로 묶어 협력하며 동작시키는 **실험적 기능 (Research Preview)**입니다.

하나의 세션이 **Team Lead(리더)** 역할을 맡아 여러 **Teammate(팀원)**를 생성하고, 다음을 담당합니다.

- 태스크 할당
- 진행 상황 관리
- 결과물 통합

각 팀원은 **독립된 컨텍스트**에서 작업하면서 서로 직접 소통하며 태스크를 함께 진행합니다.

단일 인스턴스 방식과 달리, 여러 관점에서 동시에 문제에 접근할 수 있다는 점이 핵심 강점입니다.

세부 동작 원리는 공식 문서를 참고해 주세요.

## 도입 절차

Claude Code가 이미 설치되어 있다면 Agent Teams 도입은 설정 파일 수정 한 줄로 시작할 수 있습니다.

### Step 1: Agent Teams 활성화

Agent Teams는 기본적으로 **비활성화** 상태입니다.

환경 변수 `CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS`를 `1`로 설정하면 활성화됩니다.

`settings.json`에 다음 내용을 추가하세요.

## ~/claude/settings.json

```
{
  "env": {
    "CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS": "1"
  }
}
```

## Step 2: 실행 모드 선택 후 Claude Code 시작

Claude Code를 실행할 때 파라미터로 2가지 표시 모드 중 하나를 선택합니다.

모드	설명	필요 환경
<b>in-process</b> (기본값)	모든 팀원이 하나의 터미널 안에서 동작	없음
<b>split pane</b>	팀원별로 분리된 터미널 패널이 열림	tmux 또는 iTerm2

처음 사용한다면 **in-process** 모드로 시작하는 것을 권장합니다.

```
# in-process 모드로 실행 (기본값)
claude

# 명시적으로 지정하는 경우
# claude --teammate-mode in-process
```

WSL 환경에 tmux가 설치되어 있다면 split pane 모드도 꼭 써보세요.  
각 팀원의 작업 흐름을 패널별로 실시간 확인할 수 있어 훨씬 직관적입니다.

```
claude --teammate-mode tmux
```

settings.json으로 기본 모드를 고정할 수도 있습니다.

## ~/claude/settings.json

```
{
  "teammateMode": "in-process"
}
```

**⚠ 참고:** 필자의 환경에서는 settings.json의 teammateMode 설정이 반영되지 않는 경우가 있었습니다.

claude 명령 실행 시 --teammate-mode 파라미터를 직접 지정하는 방법이 더 안정적입니다.

## Step 3: 팀 구성하기

Claude Code가 실행되면, 팀 생성을 자연어로 요청합니다.  
역할과 인원수를 구체적으로 지정할수록 원하는 팀 구성을 얻을 수 있습니다.

TODO 앱을 개발하기 위한 Agent Team을 구성해주세요.  
팀 구성은 다음 4명입니다. 아직 작업은 시작하지 말고, 팀 구성만 완료해주세요.

- 기술 리서치 담당: 프레임워크 및 DB 기술 조사 수행
- UI/UX 리서치 담당: UI 패턴 및 기능 요건 조사 수행
- 아키텍처 담당: 디렉토리 구성 및 API 설계 수행
- 비판적 리뷰어: 나머지 3명의 산출물에서 문제점 및 누락 사항 지적



#### 팁: 비판적 리뷰어의 효과

팀에 비판적 리뷰어를 포함시키면 결과물의 전반적인 완성도가 눈에 띄게 향상됩니다.  
초반에 문제를 발견해 수정하는 비용이 훨씬 적기 때문입니다.

구성에 성공하면 4명의 팀원 + 1명의 team-lead, 총 **5명**이 멤버로 생성됩니다.

기본적으로는 team-lead와의 채팅 화면에서 시작되며,  
`Shift + ↑/↓` 키로 대화 상대 팀원을 전환할 수 있습니다.

tmux split pane 모드를 활성화한 경우 아래와 같은 메시지가 출력됩니다.

```
View teammates: `tmux -L claude-swarm-XXXX a`
```

이 명령을 별도의 WSL 터미널에서 실행하면 팀원별로 분리된 패널을 확인할 수 있습니다.  
각 팀원의 작업 진행 상황 모니터링이나 직접 대화도 이 화면에서 가능합니다.

#### tmux 패널 전환 단축키:

- 다음 패널로 이동: `Ctrl+B` → `o`
- 원하는 방향의 패널로 이동: `Ctrl+B` → 방향키 (`↑↓↔`)

## Step 4: 팀에 태스크 전달하기

팀 구성이 완료되면 team-lead에게 작업 지시를 내리기만 하면 됩니다.



#### 중요: delegate mode 설정

team-lead가 팀원의 작업 완료를 기다리지 않고 단독으로 구현을 시작하는 경우가 있습니다.  
이를 방지하려면 `Ctrl+Tab`으로 **delegate mode on**을 선택하세요.

아래는 실전 태스크 전달 예시입니다.

※ 아래 프롬프트는 Docker 환경 기준입니다. 본인 환경에 맞게 수정해주세요.

구성된 팀으로 다음 작업을 진행해주세요.

#### 【중요 제약사항】

- 개발 및 실행 환경은 전부 Docker 컨테이너 내에서 완결시킬 것.  
로컬 PC 환경에는 일절 변경을 가하지 않는다.
- 최종 산출물에는 반드시 Dockerfile과 docker-compose.yml을 포함하며,  
`docker compose up --build` 명령 하나로 실행 가능한 상태로 만들 것.
- 배포는 불필요. localhost에서 브라우저로 동작 확인이 가능하면 충분.

#### 【페이지 1: 조사 (병렬 진행)】

- 기술 리서치 담당: 2025년 기준 TODO 앱에 적합한

경량 프레임워크(Hono, Express, Fastify 등)와 DB(SQLite, LowDB 등)를 조사하고, Docker 환경과의 호환성도 고려한 비교표를 docs/tech-comparison.md에 작성.

- UI/UX 리서치 담당: 심플한 TODO 앱의 UI 패턴을 조사하고, 필요 기능(추가, 완료, 삭제, 필터링, 우선순위)을 정리하여 docs/ui-spec.md에 사양서 작성.
- 아키텍처 담당: 프로젝트 디렉토리 구성, API 설계(엔드포인트 목록 및 요청/응답 사양), Docker 컨테이너 구성(단일 or 멀티 컨테이너)을 docs/architecture.md에 작성.
- 비판적 리뷰어: 위 3명의 산출물이 완성된 후 리뷰를 수행하고, 다음 관점에서 docs/review.md에 지적 사항을 정리.
  - 기술 선정의 리스크 및 누락 사항(패키지 유지보수성, 라이선스 등)
  - UI 사양의 모순이나 부족한 기능
  - 아키텍처의 과잉 설계 또는 확장성 문제
  - Docker 구성의 보안 및 운영상 우려 사항

#### 【페이지 2: 구현】

리드는 3명의 조사 결과와 비판적 리뷰어의 피드백을 종합하여 최종 기술 스택을 선정하고, TODO 앱을 구현해주세요. 최종적으로 ``docker compose up --build``로 실행되며, localhost에서 브라우저로 동작 확인이 가능한 상태까지 완성해주세요.

team-lead는 내용을 파악하여 각 팀원에게 적절한 태스크를 배분하기 시작합니다.

tmux 패널에서는 각 팀원의 실시간 작업 진행 상황도 한눈에 확인할 수 있습니다.

이후에는 간헐적으로 명령어 실행 허용이나 플랜 확인 요청에 응답해주면서, 커피 한 잔 여유롭게 마시다 보면 SI가 완성까지 알아서 처리해줍니다! ☕

## 마치며

Agent Teams를 활용하면 Claude Code의 작업을 여러 인스턴스에 분산시켜 **병렬·협력 방식**으로 태스크를 효율적으로 진행할 수 있습니다.

단, **토큰 비용은 팀원 수에 비례해 증가**한다는 점을 기억하세요.

단순한 순차 처리 작업에 적용하면 오히려 비효율적일 수 있습니다.

#### 추천 활용 순서:

1. 코드 리뷰 등 **읽기 전용 태스크**로 먼저 감각 익히기
2. 소규모 기능 구현 태스크로 확장
3. 복잡한 풀스택 개발 프로젝트에 본격 활용

또한, 본 기능은 **Research Preview(실험적 기능)** 단계이므로 향후 사양 변경이 예상됩니다.

도입 전후로 반드시 **공식 문서**의 최신 내용을 확인하시기 바랍니다.