# Multi‑Agent Workspace (Gemini + Codex) 운영 원칙 및 구조 분석

## 프로젝트 개요

**Multi‑Agent Workspace (Gemini + Codex)**는 하나의 개발 워크스페이스에서 **여러 AI CLI 에이전트(Gemini, Codex, 추후 Claude 포함)**를 동시에 활용하고 쉽게 전환할 수 있도록 설계된 시스템이다[[1]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L3-L10). Windows 환경에서 Python 기반의 *Invoke* 태스크 러너를 사용하여 구축되었으며, 이를 통해 **재현성**, **보안**, **신속한 인수인계**를 주요 목표로 운영 표준을 정립하고 있다[[2]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L3-L10). 이 레포지토리는 예컨대 일반적인 대화/계획에 강점이 있는 Gemini 에이전트와 코드 작성에 특화된 Codex 에이전트를 한데 모아, 각자의 강점을 적재적소에 활용함으로써 개발 생산성과 정확성을 높이고자 한다. 실제로 Gemini와 Codex를 하나의 시스템에서 구동하며, 필요에 따라 **에이전트를 전환**하거나 **병렬 작동**시켜 협업할 수 있는 토대를 마련한다[[3]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L76-L81). (Claude 에이전트는 향후 추가 예정으로, 현 단계에서는 관련 구조와 지침만 문서화되어 있다[[4]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L106-L114).)

## 운영 원칙 및 정책

본 워크스페이스는 **“GEMINI.md”** 문서에 정의된 운영 표준을 모든 에이전트에 공통 적용한다[[5]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L139-L143). 주요 운영 원칙과 정책은 다음과 같다:

* **Windows 우선 & UTF-8 고정:** Windows PowerShell 7 환경에서 동작하도록 최적화되어 있으며, PowerShell 래핑 없이 Python 스크립트를 직접 실행한다. 모든 표준 입출력은 **UTF-8**로 고정하여 인코딩 혼선을 방지한다[[6]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L8-L10). (예: PYTHONIOENCODING=utf-8 환경변수 설정 유지[[7]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L34-L42)[[8]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L36-L44).)
* **파일 작업 경계:** *레포지토리 디렉터리 내부*의 파일만을 대상으로 작업하며, 원칙적으로 외부 경로의 파일은 수정하지 않는다[[6]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L8-L10). 이를 통해 워크스페이스 외부 시스템에 부작용을 주지 않도록 격리한다.
* **보안/비밀 관리:** API 키, 토큰 등의 **민감 정보는 절대 Git 커밋에 포함하지 않는다**는 엄격한 정책을 적용한다[[9]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L8-L11). 실제로 .gemini/ 디렉터리 내 인증정보 파일(\*.creds\*.json, \*oauth\*.\* 등)이나 secrets/ 폴더 내 파일은 **로컬 전용**으로 취급하여 버전 관리에서 제외하며, projects/ 작업 공간도 **커밋 금지** 경로로 지정되어 있다[[10]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L20-L27). 만약 실수로 비밀정보가 노출될 경우 **키 회전**(재발급 및 폐기)과 **Git 이력 정리** 등의 대응 절차를 문서화하여 준수하도록 규정한다[[9]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L8-L11). 또한 Git 커밋 시 **pre-commit 훅**을 통해 스테이징된 내용의 diff를 출력하고 사용자에게 변경 내역 확인 및 승인을 요구함으로써, 민감정보나 의도치 않은 변경이 포함되지 않았는지 마지막으로 점검하도록 강제한다[[11]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README.md#L38-L41). (CI 같은 비대화식 환경에서는 SKIP\_DIFF\_CONFIRM=1 환경변수나 invoke commit\_safe --skip-diff-confirm 명령으로 이 훅을 건너뛸 수 있다[[11]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README.md#L38-L41)[[12]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L63-L66).)
* **Invoke 태스크 일관 사용:** 작업 수행 시 가급적 수동 작업 대신 정의된 Invoke 커맨드 태스크를 사용한다. 예를 들어, 에이전트 상태 확인, 파일 변경 *프리뷰*, HUB 전송, Inbox 확인 등은 각각 제공되는 Invoke 태스크(invoke agent.status, invoke review, invoke hub.send, invoke agent.inbox 등)를 통해 실행하도록 권장된다[[13]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L15-L19)[[14]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L17-L20). 이러한 **태스크 중심 운영**은 모든 액션에 로그 및 검증을 일관되게 남겨주며, 작업 재현과 추적을 용이하게 한다.
* **에이전트 전환 및 환경변수 활용:** 현재 활성화된 에이전트 상태는 설정 파일 .agents/config.json의 active 필드로 관리되며, invoke agent.status 명령으로 확인 가능하다[[15]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L135-L143). 에이전트를 전환할 때는 invoke agent.set --name <gemini|codex> 명령을 사용하며, 이는 해당 설정을 갱신하여 이후 태스크들이 지정된 에이전트를 통해 수행되도록 한다[[15]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L135-L143). 추가로, 개별 터미널 프로세스 단위로 **ACTIVE\_AGENT 환경변수**를 설정하면 전역 설정을 변경하지 않고도 그 프로세스에서만 다른 에이전트 라벨을 적용할 수 있다[[3]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L76-L81). 예를 들어 한 터미널에서는 ($env:ACTIVE\_AGENT='codex'); invoke start로 Codex 에이전트를 활성화한 세션을 실행하고, 다른 터미널에서는 기본 Gemini로 동작시켜 **두 에이전트의 병렬 작업**도 가능하다[[3]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L76-L81). (단, 이 경우 동일한 파일을 동시에 수정하는 상황은 피하도록 문서에서 권고된다[[16]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README_PATCH.md#L17-L20).)

## 멀티에이전트 구조

여러 에이전트를 지원하기 위한 **멀티에이전트 구조**의 핵심은, 한 워크스페이스 내에서 에이전트들이 **공존(Coexistence)**하면서도 충돌 없이 역할을 분담하도록 하는 것이다[[15]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L135-L143). 주요 구성 요소 및 설계 원리는 다음과 같다:

* **활성 에이전트 스위칭:** 앞서 언급한 대로 .agents/config.json의 active 값을 통해 현재 사용할 에이전트를 전역 관리한다[[17]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L137-L141). 사용자는 invoke agent.status로 현재 에이전트를 확인하고 invoke agent.set --name gemini|codex로 전환할 수 있으며, 전환 시 로컬 프로세스 로그에도 해당 에이전트명이 프리픽스(예: AGENT=codex)로 기록되어 구분된다[[17]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L137-L141). 이처럼 **명시적인 에이전트 라벨링**을 통해 로그와 프로세스 출력에서 어떤 에이전트가 행동하는지 일목요연하게 파악할 수 있다. 또한 개별 프로세스 환경변수로 ACTIVE\_AGENT를 지정하여 동시에 다른 에이전트를 구동할 수 있으나, 내부 **SQLite DB**는 WAL 모드로 동시 접근을 개선한 정도이므로 과도한 동시 기록 작업은 피해 안정성을 유지하도록 안내하고 있다[[3]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L76-L81)[[16]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README_PATCH.md#L17-L20).
* **공통 규칙 상속:** 멀티에이전트라 해서 에이전트별로 제각기 다른 규칙을 따르는 것은 아니다. **Windows-first**, **Python 직접 호출 & UTF-8**, **레포 내부 작업 범위**, **비밀 커밋 금지** 등의 **기본 원칙은 모든 에이전트에 동일하게 적용**된다[[5]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L139-L143). 따라서 새로운 에이전트를 추가하더라도 GEMINI.md에 정의된 이러한 운영 표준을 그대로 준수해야 한다[[4]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L106-L114). 이는 Claude 와 같은 추후 에이전트 통합 시에도 마찬가지로, 동일한 규칙을 상속하고 .agents/config.json에 에이전트명을 등록하는 정도의 방식으로 일관성을 유지하도록 설계되어 있다[[4]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L106-L114).
* **역할 분담 및** 파일 수정 프로토콜**:** 멀티에이전트 환경에서 **Gemini**와 **Codex**는 역할이 약간 다르게 규정되어 있다. 문서에 따르면 **“시스템 파일의 직접 수정은 원칙적으로 Codex 에이전트가 전담”**하도록 정하고, Gemini는 코드 수정이 필요할 경우 바로 편집하지 않고 **사용자에게 문제점과 수정안을 제안하여 승인을 받는 역할**을 한다[[18]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L141-L145). Gemini가 사용자로부터 명시적인 승인(confirm)을 받으면, 스스로 코드를 고치지 않고 대신 invoke agent.msg 명령을 통해 **Codex에게 작업 요청 메시지**를 남긴다[[18]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L141-L145). 이를 통해 **Gemini는 분석과 의사결정**, **Codex는 실제 코드 반영**이라는 분업이 이루어지며, 위험성 높은 변경은 Gemini가 함부로 수행하지 못하게 **이중 검증** 절차를 갖춘 것이다. (예: Gemini가 “파일 A의 버그를 수정해야 합니다”라고 판단하면, Codex에게 그 수정을 수행하도록 메시지를 남기고 Codex가 이를 실행에 옮기는 식이다.)
* **교차 에이전트 메시징:** 서로 다른 에이전트 사이의 소통은 **파일 기반의 메시지 큐**를 통해 이루어진다. 중앙 **context/messages.jsonl** 파일이 바로 그 용도로, 여러 에이전트가 공유하는 **JSON Lines 포맷 로그**다[[19]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L146-L154). 각 라인에는 타임스탬프 ts, 송신/수신 에이전트 from/to, 태그 배열 tags, 본문 body 필드가 포함되며, 예를 들어 Gemini가 Codex에게 남긴 요청은 {"ts": "...", "from": "gemini", "to": "codex", "tags": ["task","context"], "body": "…내용…"} 형태로 기록된다[[20]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L119-L125)[[21]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L140-L145). 이러한 메시지는 명령어로 다룰 수 있는데, invoke agent.msg --to <agent> --body "<내용>"으로 메시지를 남기고, 대상 에이전트 측에서는 invoke agent.inbox --agent <name> 명령으로 새 메시지를 확인하여 .agents/inbox/<agent>.md 요약 파일을 갱신할 수 있다[[19]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L146-L154). 필요하면 invoke agent.watch --agent <name>로 실시간 감시도 가능하며(--ack 옵션으로 수신과 동시에 자동 읽음 확인 응답까지 전송), 처리한 메시지는 invoke agent.read로 읽음 처리해나간다[[19]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L146-L154). 이 **교차 에이전트 메시지 시스템**을 통해 한 에이전트가 다른 에이전트에게 작업 맥락이나 지시를 전달하고 협업할 수 있다. (참고로 Gemini처럼 Invoke 명령을 직접 실행하기 어려운 에이전트의 경우, 이 JSONL 파일에 수동으로 한 줄을 append하는 방식으로도 동일한 효과를 낼 수 있도록 설계되었다고 문서에 언급되어 있다[[22]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L126-L131).)
* **에이전트 허브 (agents\_hub)와 작업 대기열:** 에이전트 간 메시지 외에, 사용자가 특정 에이전트에 수행시킬 **태스크**를 등록하고 추적하는 용도로 **에이전트 허브** 기능이 제공된다. 이는 .agents\_hub/ 디렉터리 아래 **queue/, processing/, archive/** 등의 폴더 구조를 통해 **파일 시스템 기반 작업 큐**를 구현한 것이다[[23]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L2-L5). 예를 들어 invoke hub.send --to gemini --title "작업" --body "설명" --type task 명령으로 Gemini 에이전트의 할 일 큐에 새로운 작업을 JSON 파일로 등록하면[[24]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README.md#L24-L29), 곧바로 해당 에이전트가 이를 **클레임(claim)**하여 processing/<agent>/ 폴더로 이동시킨 뒤 작업을 수행한다[[23]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L2-L5). 작업 완료 시에는 성공 또는 실패 여부와 간략한 노트와 함께 결과를 archive/<날짜>/success/ 또는 archive/.../failed/ 경로로 옮겨 **기록을 보존**한다[[23]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L2-L5). 이러한 수명주기는 invoke hub.inbox, invoke hub.claim, invoke hub.complete 등의 태스크로 자동화되어 관리된다[[24]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README.md#L24-L29). 이 **에이전트 허브**를 통해 사용자나 시스템이 다수의 작업을 큐에 등록해 두고, 각 에이전트가 순차적으로 이를 처리하게 하거나 작업 내역을 나중에 추적하는 것이 가능하다. (비대화식 환경에서 이 큐를 사용할 경우에도 안전성을 위해 JSON 파일 쓰기를 **원자적으로** 수행하도록 권장하고 있어, 멀티프로세스에서 일관성을 유지하려는 노력이 보인다[[23]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L2-L5).)
* **확장성과 Claude 통합:** 멀티에이전트 구조는 설계 단계부터 새로운 에이전트를 쉽게 추가할 수 있도록 고려되었다. 문서 *Extension Guide*에 따르면 향후 **Claude**와 같은 에이전트를 추가할 때도, 기존 Gemini의 운영 원칙(Windows/UTF-8/보안 규칙 등)을 그대로 따르고 .agents/config.json에 해당 에이전트를 등록하면 된다고 되어 있다[[4]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L106-L114). 실제로 레포지토리에는 간단한 Claude 실행 스크립트(claude.ps1)와 Anthropic API 키(GROQ\_API\_KEY) 설정 방식이 소개되어 있는데[[25]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L174-L182), 이는 Claude를 시범적으로 붙여볼 수 있는 **기초 토대**가 이미 마련되었음을 보여준다. 요약하면, 이 시스템의 구조는 **새로운 LLM 기반 에이전트들도 최소한의 작업으로 편입**시켜 공존할 수 있게끔 유연하게 설계되어 있다.

## 프로세스 및 워크플로우

### 세션 시작과 종료

일반적인 **세션(session)**의 lifecycle은 다음과 같이 이루어진다[[26]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L48-L56)[[27]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L60-L63):

* **시작 (Start):** 사용자가 invoke start 명령을 실행하면, 먼저 시스템 **점검** 태스크(doctor)가 동작하여 Python venv, 권한, 네트워크, 경로, 인코딩 등 환경을 체크한다[[26]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L48-L56). 다음으로 중앙 **HUB 문서**(docs/HUB.md)의 Active/Paused 작업 목록과 현재 Git 변경사항 요약(git status --porcelain)을 브리핑한다[[26]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L48-L56). 이어서 프로젝트 **문맥 인덱스**를 생성/업데이트하는 invoke context.build를 수행하여, 코드베이스의 내용이 최신 인덱싱되어 에이전트가 참고할 수 있게 한다[[26]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L48-L56). 마지막으로 이전 세션에서 사용되던 \_\_lastSession\_\_ 블록(이전 세션 요약 메모)이 있으면 이를 정리하고 새로운 세션을 시작한다[[28]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L50-L56). (빠른 시작이 필요할 경우 invoke start --fast 옵션을 사용하면 위 과정을 약식으로 수행하여 **신속 부팅**할 수 있고, 인덱싱 작업은 백그라운드에서 진행된다[[29]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L133-L137).)
* **진행 중 (During):** 세션이 진행되는 동안, 에이전트는 **작업 복원력(workflow resilience)**을 높이기 위한 몇 가지 표준을 따른다. 첫째, **복잡한 작업은 명확한 하위 목표로 분할**하여 순차적으로 진행함으로써 한번에 많은 것을 시도하지 않는다[[30]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L54-L59). 둘째, 파일 수정이나 명령 실행 전후에는 그 **의도와 결과를 즉각적으로 로그에 기록**하여 모든 변경 사항에 대한 추적을 남긴다[[30]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L54-L59). 셋째, 동일한 접근 방법으로 두 차례 연속 실패가 발생하면 **즉시 다른 해결책을 모색**하도록 하여 똑같은 시도를 세 번 반복하지 않게 한다(이전 “3-strikes rule”을 강화하여 이제 두 번 실패 시 전환)[[30]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L54-L59). 넷째, 불확실한 상황에서는 **가정을 명시적으로 기술**하고 이를 검증하는 계획까지 함께 제시함으로써, 추후 그 가정이 틀렸을 경우를 대비한다[[31]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L56-L59). 이러한 원칙들은 에이전트가 자동화된 작업 중에도 최대한 **신뢰성**을 유지하도록 하는 운영 프로토콜이다.
* **종료 (End):** 작업이 모두 완료되거나 사용자가 세션을 종료하고자 하면 invoke end 명령으로 마무리한다. 이 때 **.gitignore 상태와 미커밋 변경사항**을 최종 점검하고, 필요한 경우 invoke wip 커밋 등을 통해 진행 중인 변경을 임시 저장할 것을 권고한다[[27]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L60-L63). 또한 HUB 문서의 현재 Active 작업을 Paused로 옮기는 등 **작업 상태를 갱신**하고, 이번 세션의 주요 내용 요약을 \_\_lastSession\_\_ 블록으로 기록해 둔다[[27]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L60-L63). 이렇게 하면 다음 세션 시작 시 이전 맥락을 쉽게 파악할 수 있어 연속성이 보장된다.

### 로깅 및 인수인계

이 시스템은 **다층적인 로그 기록 체계**를 통해 작업 내용을 투명하게 남기고, 향후 분석이나 인수인계에 활용하도록 설계되어 있다. 기본적으로 모든 Invoke 태스크 실행 내역은 내부 **SQLite 데이터베이스(usage.db)**에 자동 기록된다 (Invoke 실행을 담당하는 scripts/runner.py가 태스크 이름, 시간, 에이전트 등을 로깅)[[32]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L90-L98). 이 DB의 쓰기모드는 WAL로 설정되어 여러 세션의 동시 접근 시에도 비교적 안전하게 기록되며[[3]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L76-L81), 필요하다면 사용자 스크립트를 통해 수동 로그 입력도 가능하다[[32]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L90-L98).

중앙 **HUB 문서(docs/HUB.md)**는 전체 작업 이력을 한눈에 볼 수 있는 **대시보드 역할**을 한다. 이 문서에는 진행 중인 작업이 **Active**, 대기 중인 작업이 **Staging/Planned**, 보류된 작업은 **Paused**, 완료된 작업은 **Completed** 섹션에 분류되어 표시된다[[33]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L65-L71). 사용자가 “작업 목록”을 요청하면 에이전트가 이 HUB 문서를 참조하여 각 섹션별 작업 현황과 agents\_hub의 대기열 정보를 요약해 알려줄 수도 있다[[34]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L22-L26). 또한 각 개별 작업에 대해서는 고유 ID를 부여하고 docs/tasks/<task\_id>/ 디렉터리에 **로그 파일(log.md)**을 별도로 유지하는데, 여기에는 해당 작업의 진행 로그를 **Append-only** 형태로 시간순 기록한다[[35]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L70-L73). (로그 내용의 수정이 필요할 경우 원본을 지우지 않고 하단에 추가 기록하는 규칙으로 **이력의 무결성**을 지킨다.) 이러한 구조 덕분에 추후에 새로운 팀원이나 에이전트(예: Claude)가 합류하더라도 HUB 문서와 개별 작업 로그를 읽는 것만으로 과거 맥락을 파악할 수 있어 **인계가 용이**하다.

추가로, **터미널 대화 녹화 기능**(옵션)도 지원된다. PowerShell 세션에서 $env:AI\_REC\_AUTO=1로 설정한 뒤 invoke start를 실행하면 자동으로 그 세션의 모든 콘솔 입출력이 terminal\_logs/<날짜>/session\_<시각>\_\_agent-<이름>\_\_term-<환경>\_\_pid-<pid>.txt 경로에 **Transcript**로 저장된다[[36]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L40-L48). 이 기능은 ai-rec-start.ps1/ai-rec-stop.ps1 스크립트를 통해 수동으로 토글할 수도 있으며, VS Code 터미널이나 Windows Terminal 등 다양한 콘솔 환경에서 세션 아이디를 붙여 구분 저장한다[[36]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L40-L48). 녹화된 대화 로그는 문제 발생 시 **디버깅 자료**로 활용하거나, 모델의 의사결정 과정을 검토하는 데 사용할 수 있다. (단, PowerShell Start-Transcript의 제약으로 하나의 세션 내에서만 완전한 녹화가 가능하므로 에이전트별 별도 세션 녹화를 권장한다는 팁이 있다[[37]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L46-L50).)

### 주요 Invoke 태스크 및 워크플로우

이 워크스페이스에서 제공하는 Invoke 커맨드는 **개발 워크플로우 전반을 자동화**한다. 예를 들어 다음과 같은 태스크들이 있다[[38]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L29-L37):

* invoke start – 앞서 설명한 세션 시작 절차를 수행 (환경 점검, HUB 브리핑, 컨텍스트 인덱싱 등).
* invoke doctor – 시스템 진단 (Python 버전, 경로, 권한, 네트워크, Git 설정 등 점검).
* invoke search -q "<질문>" – OpenAI API 등의 툴을 통해 웹 검색을 수행하고 결과 요약 출력[[39]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L92-L100).
* invoke context.build – 레포지토리 내 문맥 정보(코드, 문서)를 스캔하여 인덱스를 생성/갱신.
* invoke context.query "<질의>" – 사전에 구축된 컨텍스트 인덱스에 대해 질문하여 관련 정보를 얻음.
* invoke test – 프로젝트의 테스트 스위트(pytest)를 실행.
* invoke wip -m "<메시지>" – 작업 도중의 변경사항을 “WIP(작업중)” 커밋으로 임시 저장.
* invoke end – 세션을 종료하고 로그 아카이브 및 HUB 상태 갱신 (종료 절차 자동화).

그 외에도 Git 연동을 위한 invoke git.commit\_safe (안전 커밋), 편집 제안 워크플로우(invoke edits.capture/propose/diff/apply) 등 다양한 태스크가 준비되어 있다[[40]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README.md#L30-L38). 이러한 명령들은 **일관된 인터페이스**로 제공되어 사용자가 복잡한 Git 명령이나 파일 조작을 직접 하지 않고도, AI 에이전트의 안내에 따라 필요한 작업을 수행할 수 있게 해준다. 예를 들어 *사전 Diff 워크플로우*의 경우 invoke edits.capture로 변경 초안을 캡처하고, 사용자의 수동 편집 후 invoke edits.propose로 제안을 등록하면, invoke edits.diff로 제안과 기존 코드의 차이를 확인하고 최종 invoke edits.apply로 검토된 변경만 반영하는 식이다[[40]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README.md#L30-L38). 이처럼 **단계별 태스크**를 통해 항상 변경 전에 미리보기(diff)를 거치게 함으로써, 에이전트의 코드 수정이 사용자의 통제와 검토 하에 이루어지도록 하고 있다.

### Standard Operating Procedure (표준 작업 절차)

문서에는 모든 주요 기능 추가나 시스템 변경 작업 시 따르는 **4단계 표준 절차(SOP)**도 정의되어 있다[[41]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L119-L127). 이는 대규모 변경 시 사람과 AI 에이전트가 협업하는 방법론을 제시한 것으로, 단계는 다음과 같다:

1. **Phase 1: 분석 자료 준비** – *Gemini* 에이전트가 현재 시스템 상황과 목표, 그리고 해결해야 할 문제점 등을 상세히 정리한 **“분석 요청서”**를 작성한다. 사용자는 이 요청서를 외부의 심층 분석 LLM(예: GPT)에게 전달하여 현 코드베이스에 대한 **"분석 보고서"**를 획득한다[[42]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L121-L129). (Gemini가 내부 맥락을 정리하고, 외부 AI로부터 추가적인 통찰을 얻는 단계)
2. **Phase 2: 작업 지시 요청** – 사용자는 Phase1에서 얻은 *분석 요청서*와 *분석 보고서*를 종합하여, 최종적으로 신뢰하는 **컨설팅 LLM**에게 **"이 자료들을 바탕으로 Gemini-CLI가 수행해야 할 구체적인 작업 지시서를 작성해달라"**고 요청한다[[43]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L123-L127). 즉, 무엇을 어떻게 해야 할지에 대한 **실행 계획 초안**을 외부 AI로부터 얻는 단계이다.
3. **Phase 3: 작업 지시서 수령 및 실행 계획 수립** – 컨설팅 LLM이 작성한 **"작업 지시서"**(Step-by-step directives)가 나오면, 사용자는 이를 Gemini에게 전달한다. *Gemini*는 지시서를 검토하여 실제 코드 변경과 작업 순서를 담은 구체적인 **실행 계획(Action Plan)**을 수립한다[[44]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L128-L131). 이 때 각 단계별로 어떻게 구현할 것인지 제안을 내고 사용자의 확인을 받는다. (Gemini가 외부 지시서를 현실적인 개발 계획으로 변환하는 단계)
4. **Phase 4: 계획 실행** – 최종 실행 계획이 사용자에게 **승인**되면, Gemini는 그 계획에 따라 실제 코딩, 문서 수정, 테스트 등을 수행한다[[45]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L132-L134). 계획에 없거나 사용자가 승인하지 않은 임의의 행동은 하지 않으며, 모든 변경 내역은 앞서 말한 워크플로우에 따라 기록되고 공유된다. (계획된 작업을 차례대로 이행하여 목표를 달성하는 단계)

이 SOP는 사람이 Gemini와 상호작용하며 외부 LLM들의 도움까지 받아가면서 **신중하고 체계적으로 중요한 변경 작업을 수행**하도록 고안된 프로세스다. 이를 통해 대규모 업데이트 시에도 충분한 사전 분석과 계획 수립을 거치게 되어, **변경 실패율을 낮추고 학습된 모범 패턴을 축적**할 수 있다. 실제로 Gemini는 이러한 과정을 거치면서 동일한 목표에 대해 “두 번 실패 후 한 번 성공”한 접근법을 규칙으로 채택하고, 세 번 연속 성공하면 그것을 표준화(메타러닝)하는 등 지속적으로 개선해나간다는 원칙도 문서에 명시되어 있다[[46]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L113-L116).

## 장점과 한계

### 장점

* **재현성과 일관성 확보** – 운영 표준 문서(GEMINI.md)를 통해 시스템 동작을 세세히 규정하고, 모든 작업을 Invoke 태스크로 수행하게 함으로써 **항상 동일한 절차**를 따르게 했다. 이는 작업 결과의 **재현성(reproducibility)**을 높여주며, 예측 불가능한 행동을 줄여준다[[2]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L3-L10). 예를 들어, 자동 로그 기록과 pre-commit 훅 확인 절차는 어떤 환경에서 누가 작업하더라도 일정한 로그와 검증 단계를 거치게 해준다.
* **보안 및 품질 강화** – 비밀정보가 커밋되는 것을 기술적으로 차단하고 발견 시 알람을 주는 *시크릿 가드*가 적용되어 있으며[[10]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L20-L27), 커밋 전 diff 확인을 의무화해 **휴먼 에러로 인한 실수**를 줄인다[[11]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README.md#L38-L41). 또한 Windows CI 통과를 모든 PR의 필수 조건으로 하는 등[[47]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L87-L95) 품질 게이트를 설정하여 개발 산출물의 **품질 보증**을 강화했다. 이러한 다층 품질/보안 장치는 실제 운영 시 사고를 예방하고 신뢰성을 높이는 강점으로 작용한다.
* **신속한 인수인계와 협업** – 중앙 HUB 문서와 풍부한 로그, 그리고 표준화된 SOP 절차 덕분에, 새로운 사용자나 AI 에이전트가 합류하더라도 비교적 짧은 시간 내에 프로젝트의 상태와 이력이 파악 가능하다[[2]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L3-L10). 문서에 “신속한 인수인계”가 핵심 목표로 명시되어 있듯이, 모든 작업맥락을 기록하고 공유하는 구조는 **지식 손실 없이 팀 협업**을 이어갈 수 있게 돕는다. 특히 HUB.md에 현재 진행중/계획중인 작업들이 정리되고 각 작업별 로그가 남는 것은, 외부 컨설팅 AI가 그 정보를 바탕으로 조언을 해주거나 새로운 에이전트가 이어받아 작업하기에 용이한 환경을 제공한다.
* **에이전트 병행 활용 및 전문화** – 멀티에이전트 설계를 통해 하나의 작업공간에서 **서로 다른 강점을 지닌 AI 에이전트들**을 활용할 수 있다. 사용자는 상황에 따라 Gemini와 Codex를 전환하며 쓸 수 있고, 심지어 두 에이전트를 **병렬로 작동**시켜 각각 별도의 작업을 진행시킬 수도 있다[[3]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L76-L81). Codex는 코드 변경과 같은 루틴 작업에 강하고 Gemini는 사용자와의 상호작용 및 계획 수립에 능하므로, 역할에 맞게 분업시킴으로써 **작업 효율과 정확성**을 높일 수 있다. 또한 한 에이전트가 수행하기 곤란한 작업은 다른 에이전트에게 메시지를 남겨 처리하도록 하는 등, **에이전트 간 보완 협력**이 가능하도록 한 점도 이 시스템의 돋보이는 강점이다.
* **확장성과 미래 대비** – 시스템 구조가 유연하여 새로운 에이전트를 쉽게 추가할 수 있도록 되어 있다[[4]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L106-L114). 실제로 Anthropic의 Claude 모델을 통합하려는 시도가 진행 중이며, claude.ps1 스크립트와 API 키 관리, 여러 라우팅 명령(/think, /code 등)까지 기본적인 틀을 마련해 두었다[[25]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L174-L182). 이는 곧 **다양한 AI 모델의 발전에 따라 시스템을 확장**할 수 있음을 의미하며, 사용자는 특정 작업에 가장 적합한 AI를 선택해 붙여넣는 **모듈식 활용**이 가능해진다. 이러한 확장성은 향후 요구사항 변화나 새로운 LLM 등장에 유연하게 대응할 수 있게 해준다.
* **철저한 문서화와 메타 학습** – 운영 원칙부터 세부 정책, 변경이력(Patch 노트)까지 문서로 남기는 문화는 장기적으로 품질 향상에 기여한다. 예를 들어 SOP를 통해 한 번 성공한 문제 해결 패턴을 표준으로 채택하고, 3회 연속 성공 시 정식 프로세스로 삼는 등의 **메타러닝 규칙**을 적용하여 시스템 자체가 진화하도록 되어 있다[[46]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L113-L116). 또, 문제 발생 시 원인을 신속히 찾을 수 있도록 Troubleshooting 가이드와 각종 TIP들도 문서에 축적되고 있어[[48]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L101-L108), 시스템이 **스스로 개선되는 자기관리(self-refinement)**의 면모도 갖추고 있다.

### 한계

* **구조 복잡도 및 학습 곡선** – 체계가 정교한 만큼 **초기 설정과 학습이 어렵고 복잡**할 수 있다. Windows/Python/Invoke 환경에 익숙하지 않은 개발자나 에이전트에게는 venv 설정, PowerShell 프로필 적용, 각종 환경변수 활용 등 넘어야 할 허들이 있다. 또한 멀티에이전트 운영 자체가 한 번에 많은 개념(에이전트 허브, 교차 메시징, 편집 워크플로우 등)을 요구하기 때문에, 시스템 전반을 이해하는 데 시간이 소요될 수 있다. 이러한 복잡성을 줄이기 위해 .vscode/settings.json 자동 적용이나 PowerShell 설정 스크립트 제공 등의 노력이 있지만[[49]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L58-L62), **사용자는 상당한 분량의 문서를 숙지**해야 원활히 활용 가능하다는 단점이 있다. 특히 SOP와 같이 인간-AI 협업을 위한 절차까지 포함되어 있어 **절차 준수에 대한 부담**이 생길 수 있다.
* **패치에 의한 누더기화 가능성** – 멀티에이전트 지원은 기존 Gemini 단일 에이전트 환경에 **v0.1 패치 형태로 추가**되었다[[50]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README_PATCH.md#L3-L10). 이를 위해 여러 모듈을 변경하고(agent\_manager.py, tasks.py, runner.py 등) 문서를 대폭 수정하였는데[[50]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README_PATCH.md#L3-L10), 이러한 후속 통합 과정에서 구조가 다소 복잡하게 겹쳐질 위험이 있다. 예를 들어, agent 전환 스위처가 도입되면서 .agents/config.json 파일과 ACTIVE\_AGENT 환경변수, Invoke 태스크 간의 **상태 동기화**를 관리해야 하는데, 이 로직이 향후 에이전트 종류가 늘어나면 더욱 복잡해질 수 있다. 또한 여러 에이전트가 같은 자원(DB나 파일)을 다루는 상황에서 잠재적인 경합 조건(race condition)을 완전히 배제하기 어려우며 (WAL 설정 등으로 완화는 했으나), 이는 구조가 복잡해질수록 **운영 상의 리스크**로 작용할 수 있다.
* **문서와 실제 구현 간 괴리 가능성** – 본 시스템은 문서 중심으로 엄격한 가이드라인을 제시하고 있지만, 실제 코드 구현이 항상 문서와 완벽히 일치한다고 보장하기는 어렵다. 예를 들어 AGENTS.md에는 *“v0.2에서 에이전트 전환 스위처 추가 예정”*이라고 쓰여 있었지만, 실제로는 v0.1 패치에서 이미 agent.status/set 기능이 도입되었다[[51]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L112-L115). 또, *“추후 에이전트별 태스크 분기 기능 예정”*이라 명시되어 있으나 현 시점(v0.1)에서는 아직 모든 Invoke 태스크가 에이전트 구분 없이 동일하게 동작하고 있다[[52]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L152-L154). 이런 경우 문서를 그대로 믿고 새로운 기능이나 동작을 기대하면 실망할 수 있으므로, **문서의 선의(plan)와 코드의 현실(actual)** 사이에 검증이 필요하다. 다행히도 Patch 노트나 업데이트 요약에 이러한 차이가 기록되고 있지만[[53]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README_PATCH.md#L7-L14)[[54]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L154-L161), 사용자나 에이전트는 항상 최신 상태의 코드를 기준으로 문서를 해석해야 한다는 점이 한계로 지적된다. 요약하면, 광범위한 문서화는 양날의 검으로서 **문서 유지보수 부담**과 동기화 이슈를 초래할 수 있다.
* **Windows 환경 종속성** – *Windows-first*라는 기조 때문에, 비Windows (예: Linux/Mac) 환경에서 이 워크스페이스를 동일하게 활용하기는 어렵다. PowerShell 7을 전제한 각종 설정(ps1 프로필, 녹화 스크립트 등)과 Windows 전용 경로(venv\Scripts\...) 표기 등이 많아, 타 OS 사용자는 상당한 수정 없이는 시스템을 활용하기 힘들다[[55]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L6-L10). 따라서 이 구조는 현재 **Windows에 종속적**이며, 크로스플랫폼 지원이 미흡한 점도 고려해야 한다. 기업 또는 팀 내에 Windows 환경이 아닌 개발자가 있다면 별도 포팅 작업이나 제한된 기능 사용만 가능할 수 있다.
* **운영 오버헤드** – 높은 보안성과 절차 준수를 확보하는 대신, 실제 운영 시 **속도 및 유연성이 저하**될 수 있다. 예를 들어, 커밋 하나에도 diff 승인을 요구하는 훅이나, 간단한 코드 수정에도 편집 제안→승인→적용의 여러 단계를 거쳐야 하는 워크플로우는 **소규모 변경엔 다소 번거롭게** 느껴질 수 있다. 긴 로그와 문서 작성 역시 개발 속도보다는 기록을 우선시하게 해, 급한 수정이나 실험적 프로토타이핑에는 방해가 될 수 있다. 또한 사람과 외부 LLM을 거치는 SOP는 철저하지만 즉각적인 대응이 어려워 **긴급 패치 상황**에는 부적합할 수 있다. 이러한 오버헤드를 어떻게 줄이면서 핵심 원칙을 지킬지에 대한 균형점 마련이 과제로 남아 있다.

## 사용 맥락 요약

정리하면, **Multi‑Agent Workspace**는 Gemini와 Codex라는 두 종류의 AI 개발 도우미 에이전트를 하나의 환경에서 운영하며, 규칙 기반의 작업 절차와 파일 시스템을 활용한 메시징/작업 큐로 그 협업을 관리하는 시스템이다. 본 분석 문서는 **Claude Code**와 같은 새로운 에이전트가 이 레포지토리의 구조와 원리를 **신속하게 파악**하고, 향후 **사용자와 협업하여 시스템을 적용·개선**해나갈 수 있도록 돕는 것을 목적으로 작성되었다. 이를 통해 Claude와 같은 LLM 에이전트가 기존 Gemini/Codex 워크스페이스에 원활히 적응하고 기여함으로써, 멀티에이전트 개발 보조 환경을 한층 발전시킬 수 있을 것으로 기대된다.[[2]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L3-L10)[[1]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L3-L10)

[[1]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L3-L10) [[3]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L76-L81) [[4]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L106-L114) [[6]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L8-L10) [[7]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L34-L42) [[8]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L36-L44) [[9]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L8-L11) [[12]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L63-L66) [[13]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L15-L19) [[14]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L17-L20) [[20]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L119-L125) [[21]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L140-L145) [[22]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L126-L131) [[23]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L2-L5) [[25]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L174-L182) [[29]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L133-L137) [[32]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L90-L98) [[34]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L22-L26) [[36]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L40-L48) [[37]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L46-L50) [[49]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L58-L62) [[51]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md#L112-L115) AGENTS.md

<https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/AGENTS.md>

[[2]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L3-L10) [[5]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L139-L143) [[10]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L20-L27) [[15]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L135-L143) [[17]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L137-L141) [[18]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L141-L145) [[19]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L146-L154) [[26]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L48-L56) [[27]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L60-L63) [[28]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L50-L56) [[30]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L54-L59) [[31]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L56-L59) [[33]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L65-L71) [[35]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L70-L73) [[38]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L29-L37) [[39]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L92-L100) [[41]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L119-L127) [[42]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L121-L129) [[43]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L123-L127) [[44]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L128-L131) [[45]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L132-L134) [[46]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L113-L116) [[47]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L87-L95) [[48]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L101-L108) [[52]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L152-L154) [[54]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L154-L161) [[55]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md#L6-L10) GEMINI.md

<https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/GEMINI.md>

[[11]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README.md#L38-L41) [[24]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README.md#L24-L29) [[40]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README.md#L30-L38) README.md

<https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README.md>

[[16]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README_PATCH.md#L17-L20) [[50]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README_PATCH.md#L3-L10) [[53]](https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README_PATCH.md#L7-L14) README\_PATCH.md

<https://github.com/etloveaui/multi-agent-workspace/blob/31b0105d66b544a46410212c77899d3c5194a2ca/README_PATCH.md>