

Toolly Multi-Agent Harness

설치부터 운영까지 — 사용 설명서

버전 2.7.7 · 2026-09-20

이 문서는 라이선스 발급 메일에 함께 옵니다.
라이선스 키는 재발급이 번거로우니 이 메일과 함께 보관하십시오.

문의 support@mytoolly.com

사이트 <https://multiagentharness.kr>

1. 이 제품이 하는 일

한 화면을 여러 패널로 나눠 패널마다 에이전트 하나를 두고, 패널 사이에 오가는 말을 하네스가 대신 옮기고 확인하고 남깁니다. 사람이 창을 옮겨 다니며 복사해 붙일 필요가 없습니다.

메시지	위조할 수 없는 기록을 거쳐서만 오갑니다. 화면에 글자를 출력하는 것은 전달이 아닙니다.
역할	설정이 정합니다. 팀 구성을 바꿔도 일하고 검증하는 방식은 그대로입니다.
격리	구현 담당 패널은 각자 자기 git 브랜치에서 일하고, 검증을 통과한 것만 본선에 합쳐집니다.
검증	작업자의 완료 보고는 검증 역할에게 자동으로 넘어갑니다. 사람이 중계하지 않습니다.

에이전트 CLI(claude · codex · gemini)는 이 패키지에 들어 있지 않습니다. 설치와 로그인은 4장을 보십시오. 무엇이 빠졌는지는 toolly doctor 가 이름으로 알려 줍니다.

필요한 것

운영체제	Linux x86_64 — Debian/Ubuntu(.deb) 또는 RHEL/Fedora(.rpm)
함께 깔리는 것	필요한 도구는 설치할 때 같이 들어옵니다. 따로 준비하실 것은 없습니다.
권한	패키지 설치에 관리자 권한이 필요합니다. 하네스 자체는 계정 홈에 들어갑니다.

2. 설치

받으신 라이선스 키로 한 줄이면 끝납니다. 설치할 계정으로 로그인해 그대로 실행하십시오.

```
bash -c "$(curl -fsSL https://mytoolly.com/install/<라이선스키>)"
```

앞에 sudo 를 붙이지 마십시오. 패키지 설치가 필요한 순간에 스크립트가 직접 묻습니다. sudo 권한이 없는 계정이면 어떻게 해야 하는지도 화면에 안내됩니다.

이 한 줄이 아래를 순서대로 끝냅니다.

1 받을 곳 등록	배포처가 진짜인지 확인한 뒤 내려받을 곳을 등록합니다.
2 설치	설치한 뒤 파일이 배포본과 한 글자도 다르지 않은지 대조합니다.
3 계정 준비	이 계정에서 쓸 수 있도록 홈 아래에 한 벌을 만듭니다.
4 활성화	이 기기를 등록하고 라이선스를 받아 적용합니다.
5 프로젝트	프로젝트 폴더 이름을 물어봅니다. 답하면 나머지 준비까지 함께 끝냅니다.

여기까지 끝나면 남는 명령은 toolly up 하나입니다. 폴더 이름을 묻는 자리에서 그냥 Enter 를 누르면 건너뛰고, 나중에 toolly setup 으로 정하면 됩니다.

프로젝트 폴더를 미리 정해 두려면

묻지 않고 바로 잡고 싶으면 폴더 이름을 넘기십시오. 홈 아래에 만들어집니다.

```
HARNESS_PROJECT=public bash -c "$(curl -fsSL https://mytoolly.com/install/<라이선스키>)"
```

관리자가 다른 계정에 설치할 때

```
sudo env HARNESS_TARGET_USER=<계정명> bash -c "$(curl -fsSL https://mytoolly.com/install/<키>)"
```

설치는 하네스만 갑니다. 패널이 여는 대화 창(claude · codex · gemini)은 각 회사의 CLI 이고 따로 설치·로그인하셔야 합니다 — 4장을 보십시오.

저장소만 등록하고 직접 설치하려면

```
# Debian · Ubuntu
curl -fsSL https://mytoolly.com/apt/install.sh | bash

# RHEL · Fedora
curl -fsSL https://mytoolly.com/rpm/install.sh | bash
```

3. 라이선스

라이선스 키를 잃어버리지 마십시오. 이 문서가 온 메일에 키가 함께 있습니다. 메일을 보관하시거나 키만 따로 안전한 곳에 적어 두십시오.

라이선스는 기기와 계정에 묶입니다. 활성화할 때만 서버와 통신하고, 그 뒤로는 받아 둔 라이선스만으로 스스로 확인합니다. 인터넷이 끊겨도 동작합니다.

상태 확인

toolly license

활성 여부와 만료일, 이 기기를 가리키는 지문이 나옵니다.

다른 기기나 계정으로 옮기려면

설치 가능 대수를 넘으면 활성화가 거부됩니다. 쓰던 기기를 정리하고 옮기실 때는 support@mytoolly.com 로 알려 주시면 기존 기기 연결을 해제해 드립니다. 그 뒤 새 기기에서 같은 설치 명령을 다시 실행하시면 됩니다.

받게 되는 안내 메일

발급	키와 설치 명령, 이 설명서
새 기기 활성화	시각·호스트·지문. 본인이 한 것이 아니면 바로 알려 주십시오.
정지 · 해제 · 회수	상태가 바뀌면 그때마다

4. 에이전트 CLI 준비

패널 하나하나가 여는 것은 에이전트 CLI 의 대화 창입니다. 하네스는 그 창을 패널 안에 띄우고, 창과 창 사이의 메시지를 대신 옮기고 확인하고 기록합니다. 창 자체는 각 회사의 CLI 이고, 하네스에 들어 있지 않습니다 — 직접 설치하고 로그인하셔야 합니다. 패키지 의존성에도 넣지 않았습니다(계정·요금제가 고객 것이기 때문입니다).

필요한 CLI 는 고르신 팀 구성이 정합니다. 지금 무엇이 필요한지는 toolly doctor 의 「에이전트 CLI」 항목이 이름으로 알려 줍니다. 없으면 toolly up 은 세션을 만들기 전에 멈춥니다 — 빈 패널이 남지 않게 하기 위해서입니다.

설치

세 CLI 모두 Node.js 18 이상을 씁니다. 배포처가 안내하는 설치 방법이 바뀔 수 있으니 각 공식 문서를 우선하십시오. 널리 쓰이는 방법은 다음과 같습니다.

```
npm install -g @anthropic-ai/claude-code # claude (Anthropic)
npm install -g @openai/codex           # codex (OpenAI)
npm install -g @google/gemini-cli      # gemini (Google)
```

설치되면 명령 이름이 각각 claude · codex · gemini 입니다. 하네스는 이 이름으로 찾습니다. sudo 없이 설치하셨다면 그 경로가 PATH 에 있어야 합니다.

로그인 — 계정 인증

인증은 CLI 마다 자기 방식이 있고, 하네스는 관여하지 않습니다. 공통된 확실한 방법은 하나입니다 — 터미널에서 그 CLI 를 한 번 직접 실행하면 로그인 화면이 뜹니다. 브라우저로 계정 로그인을 마치면 자격이 홈 아래에 저장되고, 그 다음부터는 묻지 않습니다.

claude	claude auth login 으로 로그인하고 claude auth status 로 확인합니다. 브라우저를 열 수 없는 서버라면 claude setup-token 으로 장기 토큰을 만듭니다. 자격은 ~/.claude/.credentials.json 에 저장됩니다.
codex	codex 를 한 번 실행하면 로그인 방법을 고르는 화면이 나옵니다(계정 로그인 또는 API 키). 명령으로 하려면 codex login 이며, 자격은 ~/.codex/ 아래에 남습니다.
gemini	gemini 를 한 번 실행하면 인증 방법을 고르는 화면이 나옵니다(Google 계정 로그인 또는 API 키). 대화 창 안에서는 /auth 로 다시 고를 수 있고, 자격은 ~/.gemini/ 아래에 남습니다.

하위 명령 이름은 CLI 판마다 다를 수 있습니다. 확실한 확인은 claude --help · codex --help · gemini --help 입니다.

로그인은 계정마다 한 번

자격은 그 CLI 를 실행한 OS 계정의 홈에 저장됩니다. 그래서 패널을 여러 OS 계정으로 나누는 uid 격리를 쓰시면, 그 계정마다 한 번씩 로그인해야 합니다. 한 계정으로 모두 돌리는 기본 구성이라면 한 번이면 됩니다.

```
# uid 격리를 쓰는 경우, 패널 계정마다
sudo -u hx-backend -H claude auth login
sudo -u hx-qa      -H codex
```

확인 — 하네스가 띄우기 전에

하네스가 패널에서 실제로 치는 것은 다음과 같은 한 줄입니다. 프로젝트 폴더로 옮긴 뒤 그 CLI 를 확인 프롬프트 없이 띄우고, 역할 지침을 함께 넘깁니다.

```
cd <작업 폴더> && claude --dangerously-skip-permissions \
    --model <모델> --append-system-prompt "$(cat <역할 지침>)"

cd <작업 폴더> && codex --yolo --model <모델> -- "$(cat <역할 지침>)"
cd <작업 폴더> && gemini --approval-mode yolo --model <모델> -i "..."
```

확인을 건너뛰는 옵션을 쓰는 이유는 패널에 사람이 붙어 매번 눌러 줄 수 없기 때문입니다. 대신 하네스의 가드 정책이 위험한 명령을 흑에서 막습니다(9장 안전장치).

처음 보는 폴더에서는 CLI 가 「이 폴더를 신뢰하는가」를 먼저 묻습니다. 하네스가 화면을 읽어 자동으로 답하므로 손댈 것은 없습니다. 다만 로그인에 안 되어 있으면 로그인 화면에서 멈추고, 하네스는 준비 신호를 보지 못한 채 기다립니다 — 패널 하나만 조용히 멈춰 있다면 거의 이 경우입니다.

```
claude --version      # 설치·경로 확인
toolly doctor         # 「에이전트 CLI」 항목에 판번호가 나오면 정상
toolly attach         # 기동 뒤 각 패널이 대화 창까지 떠 있는지 눈으로 확인
```

요금제·사용량은 고객의 계정에 붙습니다. 다섯 패널이면 다섯 개의 대화가 동시에 돌아가므로, 쓰시는 요금제의 동시 사용 한도를 미리 확인하시는 편이 좋습니다.

5. 첫 실행

설치할 때 프로젝트를 정하셨다면 이 장은 건너뛰고 바로 기동하십시오. 건너뛰셨다면 지금 정합니다 — 폴더 이름만 답하면 홈 아래에 잡히고, 나머지 준비까지 함께 끝냅니다.

```
toolly setup      # 세션 이름·폴더·팀을 물어봅니다
toolly doctor     # 준비 상태 점검
```

doctor 는 막고 있는 것만 짚어 줍니다. 빠진 명령, 없는 에이전트 CLI, 활성화되지 않은 라이선스처럼 고칠 것이 있으면 해결 명령까지 함께 보여 줍니다.

지침 파일이 놓이는 자리

에이전트는 작업 폴더에서 상위 폴더로 올라가며 지침 파일을 읽습니다. 그래서 두 자리로 나눠 둡니다 — 멀티에이전트 규약은 계정 홈에 한 벌, 프로젝트에 관한 것은 그 프로젝트에.

홈 — 지침 파일 ~/CLAUDE.md · ~/AGENTS.md · ~/GEMINI.md — 멀티에이전트 공유 규약. toolly sync-rules 가 관리하며 계정당 한 벌입니다.

홈 — 훅·스킬 ~/.claude/ · ~/.codex/ · ~/.agents/ — 훅, 스킬, 서브에이전트. 역시 하네스가 관리합니다.

프로젝트 — 메모리 <프로젝트>/CLAUDE.md · AGENTS.md — 그 프로젝트의 메모리입니다. 없으면 골격만 만들어 드리고 이후 내용은 건드리지 않습니다. 프로젝트의 구조·규약·결정을 여기에 적으십시오.

프로젝트 폴더를 여러 개 쓰시면 프로젝트마다 자기 메모리를 갖습니다. 공유 규약은 홈에 한 벌만 있으므로 프로젝트를 옮겨도 같은 규약으로 움직입니다.

기동

```
toolly up        # 세션·패널·엔진을 한 번에 + 화면까지 붙입니다
toolly attach    # 빠져나온 뒤 다시 붙기 (빠져나오기: Ctrl-b d)
toolly status    # 엔진 상태와 작업 현황
toolly down      # 엔진과 세션을 함께 내리기 (stop 도 같습니다)
```

up 은 기동을 마치면 그 화면에 바로 붙여 줍니다. Ctrl-b d 로 빠져나와도 세션은 계속 돌고, toolly attach 로 다시 붙입니다. 붙지 않고 명령만 돌리려면 toolly up --no-attach 입니다(스크립트·원격 실행).

처음 기동하면 각 패널에서 에이전트 CLI 가 뜨고, 폴더를 신뢰하겠냐는 물음에 자동으로 답합니다. 로그인이 안 되어 있으면 그 패널만 멈춰 있으니 toolly doctor 로 확인하십시오 — 4장을 보십시오.

up 은 만들기 전에 먼저 확인합니다. 활성화되지 않았으면 아무것도 만들지 않고 멈추면서 이 기기의 지문을 보여 줍니다. 규약이 빠져 있으면 조용히 채워 넣습니다.

6. 팀 구성

기본 팀은 다섯입니다. 역할·담당 모델은 설정에서 바꿀 수 있습니다.

ORCH	지휘·배정·병합. 사용자 지시가 들어오는 유일한 창구입니다.
BACKEND	서버·API·데이터 구현. 자기 워크트리에서 일합니다.
FRONTEND	화면·클라이언트 구현. 자기 워크트리에서 일합니다.
QA	백엔드 결과 검증. PASS / FAIL 을 판정합니다.
FE_REVIEW	프론트엔드 결과 검토. PASS / FAIL 을 판정합니다.

백엔드가 완료를 보고하면 엔진이 QA 에게 검증을 넘기고, FAIL 이면 원 작업자에게 되돌립니다. 프론트엔드도 같은 방식입니다. 사람이 중계하지 않습니다.

팀 바꾸기

준비된 구성이 함께 들어 있습니다(webdev · design · api). toolly setup 이 물어보므로 설정 파일을 열 필요가 없습니다.

```
toolly setup          # 팀 구성도 함께 물어봅니다
toolly down -y && toolly up # 이미 돌고 있다면 - 패널을 새 구성으로 다시 만듭니다
```

지금 팀만 보고 싶으면 toolly config --show, 이름을 알고 바로 바꾸려면 toolly team design 입니다.

역할·검증자·흐름이 한 벌로 바뀌고, 새 패널이 쓸 준비까지 함께 끝납니다. 고른 구성은 업그레이드 뒤에도 유지됩니다.

설정 바꾸기

세션 이름·프로젝트 폴더·팀 구성은 toolly setup 이 물어봅니다. 처음 설치할 때든 나중에도 같은 명령입니다. 지금 값이 [] 안에 나오므로 그대로 둘 것은 Enter 만 치면 됩니다.

```
toolly setup          # 물어보며 바꿉니다 (바꿀 것만 요약해 확인합니다)
toolly config --show  # 지금 값만 보기
toolly config session <이름> # 값 하나만 (스크립트용)
toolly config --edit   # 그 밖의 값은 파일을 열어서
```

바꾸기 전 설정을 harness.local.json.bak.<시각> 으로 남기고, 값마다 검사해서 통과하지 못하면 그 값은 되돌립니다. 끝에는 지금 세션이 떠 있는지 보고 무엇을 쳐야 반영되는지 알려 드립니다.

7. 명령

toolly 만 치면 자주 쓰는 것이 나옵니다. 드물게 쓰는 것까지 보려면 `toolly help --all` 입니다. 아래는 그 전부입니다.

준비

setup	세션 이름·프로젝트 폴더·팀 구성을 물어봅니다. 처음 설치할 때든 나중에 바꿀 때든 이 명령 하나입니다. 지금 값이 기본으로 나오므로 그대로 둘 것은 Enter 만 칩니다.
license	라이선스가 살아 있는지, 언제까지인지, 이 기기를 가리키는 지문이 무엇인지 봅니다. 기기를 옮기실 때 이 지문을 배포처에 알려 주시면 됩니다.
activate --file <파일>	배포처에서 받은 라이선스 파일을 적용합니다. 설치할 때 자동으로 끝나므로, 보통은 기기를 옮기거나 기간을 연장했을 때만 씁니다.
install-account	이 계정의 트리를 처음 만듭니다. 계정마다 한 번입니다.
update	이미 설치된 새 판에 이 계정을 맞춥니다. 보통은 칠 일이 없습니다 — <code>up</code> · <code>setup</code> · <code>doctor</code> · <code>team</code> 을 칠 때 런처가 알아서 맞춥니다. 자동 갱신이 실패했을 때만 씁니다. 저장소에서 내려받는 명령이 아닙니다.

기동과 종료

up	세션을 만들고 패널마다 에이전트를 띄우고 엔진을 켜 뒤, 그 화면에 붙여 줍니다. 활성화되지 않았거나 에이전트 CLI 가 없으면 아무것도 만들지 않고 멈춥니다. 붙지 않으려면 <code>up --no-attach</code> .
attach	Ctrl-b d 로 빠져나온 뒤 다시 그 화면으로 들어갑니다. 빠져나와도 하던 일은 계속 돕니다.
down	엔진을 멈추고 세션을 닫습니다. 패널에 떠 있던 에이전트까지 함께 내려갑니다. <code>stop</code> 도 같습니다. 확인 없이 내리려면 <code>down -y</code> .
restart	엔진만 다시 켵니다. 패널의 에이전트는 그대로 두므로 하던 대화가 끊기지 않습니다.
recover	엔진을 다시 세우고, 죽어 있는 패널만 골라 되살립니다. 살아 있는 패널은 건드리지 않습니다. 한두 패널이 멈춰 보일 때 먼저 써 보십시오.

일 시키고 들여다보기

send	특정 역할에게 직접 지시합니다. 보통은 ORCH 패널에 말로 시키면 되고, 이 명령은 한 사람에게 바로 꽂고 싶을 때입니다. 쓰는 법은 이 장 끝에 있습니다.
summary	지금 무엇이 돌고 있는지 사람이 읽는 형태로 요약합니다. 패널마다 살아 있는지, 무슨 작업을 들고 있는지 한눈에 봅니다.
report [--days N]	최근 기록에서 누가 무엇을 했고 어디서 막혔는지 뽑아 줍니다. 기본 7일이고, --md <파일> 로 파일에 받을 수 있습니다.
logs [-f]	엔진이 남긴 로그입니다. -f 는 새로 찍히는 것을 따라갑니다. 무엇이 안 될 때 <code>doctor</code> 다음으로 볼 곳입니다.
help / help --all	자주 쓰는 명령만 보여 줍니다. --all 은 드물게 쓰는 것까지 전부입니다.

작업 합치기

구현 담당 패널은 각자 자기 브랜치에서 일합니다. 아래는 그 결과를 보고 합치는 명령입니다.

worktree status

패널마다 어느 브랜치에서 무엇을 고쳤는지 봅니다.

worktree merge

검증을 통과한 작업을 본선에 합칩니다.

worktree sync

본선의 최신 내용을 각 패널 쪽으로 내려 줍니다.

worktree reset

그 패널의 작업을 버리고 본선과 같게 되돌립니다.

점검

doctor

막고 있는 것만 짚어 주고 해결 명령을 함께 보여 줍니다. 무엇이 안 될 때 제일 먼저 치실 명령입니다.

version

깔려 있는 판을 셋 다 보여 줍니다 — 엔진, 이 계정, 패키지. 업그레이드한 뒤 install-account 를 안 돌려 계정만 낡은 상태도 여기서 드러납니다.

config --show

지금 설정값과 패널 구성을 봅니다. 바꾸는 것은 setup 입니다.

seal write | verify

설치물이 처음 상태에서 바뀌었는지 확인합니다. verify 는 확인, write 는 지금 상태를 기준으로 다시 잡는 것입니다. 정당하게 업그레이드하신 뒤에만 write 하십시오.

audit

지금까지 오간 기록이 중간에 고쳐지거나 빠지지 않았는지 전부 확인합니다.

ready

패널마다 에이전트가 실제로 답할 수 있는 상태인지 봅니다. doctor 가 설치와 설정을 본다면 ready 는 지금 살아 있는지를 봅니다.

check [--explain]

지금 팀에 걸린 자동 점검을 돌립니다. --explain 은 돌리지 않고 무엇을 보지만 보여 줍니다. 팀마다 다릅니다 — 코드 팀은 테스트 러너를, 문서 팀은 링크와 산출물을 봅니다.

catalog

역할·팀·모델 카탈로그를 고친 뒤 팀 구성과 규약 문서를 다시 만듭니다. 카탈로그를 직접 고치지 않으셨다면 칠 일이 없습니다.

uninstall [--dry-run]

이 계정에서 toolly 를 지웁니다. 무엇이 지워지는지 --dry-run 으로 먼저 보십시오. 프로젝트 폴더 안의 data/ 와 plans/ 도 대상이라, 그 폴더를 홈으로 잡아 두셨다면 위험합니다(그래서 홈은 프로젝트 폴더로 받지 않습니다).

rc [--names]

휴대폰 원격 제어가 켜져 있는지와 각 패널이 어떤 이름으로 보이는지 확인합니다.

보통은 쓸 일이 없는 것

아래는 setup 과 up 이 알아서 하는 단계이거나, 스크립트·진단용입니다. toolly help --all 에 같이 나옵니다. 없앤 것이 아니라 옮긴 것이고 전부 그대로 동작하므로, 무엇이 잘못됐을 때 한 단계씩 손으로 짚어 볼 수 있습니다.

team

지금 팀과 고를 수 있는 구성을 봅니다. team <이름> 이면 그 구성으로 바꾸고, 새 패널이 쓸 준비까지 함께 끝냅니다. setup 이 물어보므로 이름을 미리 알 때만 씁니다.

config <키> <값>

값 하나만 바꿉니다. 바꿀 수 있는 것은 session 과 project 둘입니다. 묻지 않고 바로 반영하되 검사에 걸리면 되돌립니다. 스크립트용입니다.

config --edit

설정 파일을 편집기로 엽니다. 위 둘로 안 되는 값(패널 구성·건강 검사 주기 같은 것)을 직접 고칠 때입니다. 닫으면 바로 검사하고, 잘못됐으면 되돌릴지 물어봅니다.

sync-rules

지침 파일과 혹·스킬을 에이전트가 읽는 자리에 다시 놓습니다. 홈에 공유 규약을, 프로젝트에 그 프로젝트의 메모리를 둡니다. setup 과 up 이 이미 합니다.

trust

작업 폴더를 각 CLI 의 신뢰 목록에 미리 넣습니다. 넣어 두지 않으면 첫 기동에서 「이 폴더를 신뢰하는가」 물음에 걸립니다. up 이 합니다.

layout

세션과 패널만 만듭니다. 에이전트는 띄우지 않고 엔진도 켜지 않습니다. up 의 첫 단계입니다.

panes [--pane N]

패널의 에이전트를 띄웁니다. 이미 돌고 있는 패널은 건너뛰고, --force 면 끝내고 다시 띄웁니다. --pane N 으로 그 패널 하나만 고를 수 있습니다. 한 패널만 이상할 때 씁니다.

worktree setup	패널마다 자기 브랜치와 작업 사본을 처음 만듭니다. up 이 합니다.
start	엔진만 켭니다. 세션이 이미 있어야 합니다. 패널은 건드리지 않습니다.
status	엔진 상태와 작업 현황을 기계가 읽는 형태로 그대로 냅니다. 사람이 보실 것은 summary 입니다.
ident list	패널 등록부를 봅니다 — 어느 패널이 어떤 지문으로 서명하는지. ident init 은 키를 새로 발급하고(전에 남은 기록은 확인할 수 없게 됩니다), ident check 는 등록부가 지난 기동 때와 같은지 봅니다.
fingerprint	이 기기와 계정을 가리키는 지문만 한 줄로 냅니다. license 가 같은 값을 함께 보여 주므로 보통은 그쪽을 쓰십시오.
brief --pane N	그 패널이 기동할 때 받은 지침 문서를 그대로 출력합니다. 에이전트가 엉뚱하게 움직일 때 무엇을 읽고 있는지 확인하는 용도입니다.
bench [건수]	메시지를 그만큼 흘려 전달 성능을 재 봅니다. 기기가 느릴 때 원인이 하네스인지 보려는 것입니다.
selftest	엔진이 자기 기능을 스스로 점검합니다. 설치가 온전한지 볼 때 씁니다.

패널에 일 시키기

ORCH 패널에 하실 일을 그대로 말씀하시면 됩니다. ORCH 가 쪼개서 배정하고, 검증까지 끝난 결과만 보고합니다. 직접 특정 역할에게 보내려면 아래를 씁니다.

```
toolly send --to BACKEND --task T20260914_API-1 \
  "GOAL: ... | 범위: ... | 완료 조건: ... | 제약: ..."
```

상태(DONE · PASS · FAIL · NEED_INPUT · BLOCKED)는 기록에 남는 값입니다. 본문에 완료라고 적는 것으로는 아무 일도 일어나지 않고, 반대로 본문에 그 단어가 들어가도 오판되지 않습니다.

8. 업그레이드

판을 올리는 일은 두 단계입니다. 기기에 패키지를 받는 것은 관리자가 한 번, 계정마다 자기 트리를 맞추는 것은 자동입니다.

1단계 — 기기에 새 판을 받는다 (관리자)

```
sudo apt update && sudo apt upgrade toolly
# RHEL 계열: sudo dnf upgrade toolly
```

apt update 를 빼면 안 됩니다. 캐시에 남은 옛 목록을 보고 "이미 최신" 이라고 답합니다. 실제로 올라갔는지는 toolly version 이 알려 줍니다.

2단계 — 계정을 맞춘다 (보통 자동)

계정마다 자기 트리를 홈 아래에 따로 갖고 있고, 실제로 도는 것은 그 트리입니다. 런처가 판을 비교해서, 다음에 up · setup · doctor · team 을 치실 때 스스로 맞춥니다. 따로 치실 것이 없습니다. 화면에 이렇게 지나갑니다.

```
[toolly] 새 판이 설치돼 있습니다 (2.7.7-21 → 2.7.7-22). 이 계정을 맞춥니다.
[toolly] 계정 트리를 2.7.7-22 로 갱신했습니다.
```

send · restart · down 에서는 맞추지 않습니다. 에이전트가 일하는 도중에 규약과 설정이 발밑에서 바뀌면 안 되기 때문입니다. 자동 갱신이 실패했을 때만 toolly update 를 직접 치십시오. 저장소에서 내려받는 명령이 아니라, 이미 깔린 새 판에 이 계정을 맞추는 명령입니다.

규약·스킬이 바뀐 판이라면 한 줄 더

판올림은 제품 파일을 바꿀 뿐, 각 패널 자리에 놓인 지침과 스킬까지 다시 깔지는 않습니다. 갱신 화면 끝에 그 안내가 나옵니다. 계정마다 한 번 치십시오.

```
toolly sync-rules && toolly restart
# sync-rules 지침·혹·스킬을 각 패널 자리에 다시 놓는다
# restart 엔진만 다시 켜다 - 패널의 대화는 끊기지 않는다
```

계정이 여럿이면 관리자가 한 번에 돌릴 수 있습니다. 그 계정의 패널이 돌고 있으면 먼저 내리십시오.

```
for u in 계정1 계정2 계정3; do
  sudo -u "$u" -H /usr/bin/toolly sync-rules
done
```

올린 뒤 확인

```
toolly version # 엔진 · 계정 · 패키지 세 판이 같은지
toolly doctor  # 남은 것이 있으면 여기서 짚어 줍니다
toolly seal write # 설치물이 바뀌었으니 봉인을 다시 씁니다
```

계정 설정·서명키·라이선스·고른 팀 구성은 그대로 보존됩니다. 다시 활성화할 필요가 없습니다. 봉인만은 자동으로 다시 쓰지 않습니다 — 자동으로 쓰면 그 사이에 누가 파일을 바꿨더라도 그대로 통과시키게 됩니다.

9. 안전장치

봉인

toolly seal verify — 설치물이 봉인한 때와 달라졌는지 파일 단위로 확인합니다.

기록

패널 사이의 모든 메시지가 고칠 수 없는 상태로 남습니다. toolly audit 로 전체 확인.

가드

파괴적 명령·비밀 열람·경계 밖 쓰기를 실행 전에 막습니다.

역할 제약

검증 역할은 코드를 고치지 못하고, 지휘 역할은 직접 구현하지 못합니다.

정당하게 파일을 고치셨다면 toolly seal write 로 다시 봉인하십시오. 에이전트가 스스로 다시 봉인하는 것은 가드가 막습니다.

10. 자주 겪는 일

sudo 를 쓸 수 없다고 나옵니다

그 계정에 관리자 권한이 없는 것입니다. 화면에 두 가지 방법이 함께 안내됩니다 — 권한이 있는 다른 계정에서 이 계정으로 설치하거나, 관리자가 그 계정을 sudo 그룹에 넣어 주는 것입니다(넣은 뒤 다시 로그인해야 적용됩니다).

다른 패키지 때문에 설치가 멈춥니다

기기에 원래 고장 나 있던 패키지가 있으면 apt 가 오류로 끝납니다. 설치기는 우리 패키지가 제대로 깔렸는지 따로 확인하고 계속 진행합니다. 남은 문제는 아래로 정리하십시오.

```
sudo dpkg --configure -a
```

패널이 하나 비어 있습니다

그 패널의 에이전트 CLI 가 설치되지 않았거나 로그인되지 않은 경우가 대부분입니다. toolly doctor 가 어느 CLI 가 없는지 알려 줍니다. 설치·로그인 뒤 toolly recover 로 되살립니다.

작업이 진행되지 않습니다

toolly status 로 어느 역할이 무엇을 쥐고 있는지 봅니다. 보고 없이 멈춰 있으면 toolly logs -f 에 전달 실패나 타임아웃이 남아 있습니다. toolly recover 로 엔진과 죽은 패널을 되살릴 수 있습니다.

활성화가 거부됩니다

설치 가능 대수를 넘었거나 라이선스가 정지·회수된 경우입니다. 화면에 사유가 그대로 표시됩니다. support@mytoolly.com 로 문의해 주십시오.

11. 복구

무엇이 잘못됐는지 모를 때는 toolly doctor 가 먼저입니다. 막고 있는 것만 골라 고칠 명령과 함께 보여 줍니다. 아래는 doctor 가 짚어 준 뒤에 하실 일입니다.

패널이 멈췄거나 비어 있다

가벼운 것부터 씁니다. 위에서 안 되면 아래로 내려가십시오. 위의 둘은 하던 대화를 지키고, 마지막 것만 전부 새로 시작합니다.

```
toolly recover          # 죽은 패널만 되살린다 (살아 있는 것은 건드리지 않는다)
toolly panes --pane 2   # 그 패널 하나만 다시 띄운다
toolly restart          # 엔진만 다시 켜다 (패널의 대화는 유지)
toolly down -y && toolly up # 전부 새로
```

에이전트가 아예 뜨지 않고 셸로 빠진다

CLI 자체가 시작하지 못한 것입니다. 그 패널에 찍힌 오류를 먼저 읽으십시오. 가장 흔한 둘은 이렇습니다.

CLI 가 없다	toolly doctor 가 어디에 있는지 찾아 알려 줍니다. root 로 설치하셨다면 다른 계정에서는 보이지도 쓰이지도 않습니다 — 모두가 읽을 수 있는 자리에 두거나 그 계정으로 설치하십시오.
----------	---

설정을 읽지 못한다	그 CLI 의 설정 파일이 깨진 것입니다. 아래 「신뢰 등록이 깨졌다」를 보십시오.
------------	--

인증은 계정마다 따로입니다. 바이너리를 공용 자리에 두어도 계정마다 한 번은 그 CLI 에 로그인하셔야 합니다. 로그인 전에는 패널이 뜨기만 하고 답하지 않습니다.

신뢰 등록이 깨졌다

toolly 는 패널이 일할 폴더를 각 CLI 의 신뢰 목록에 미리 적어 둡니다. 묻는 창에서 팀 전체가 멈추지 않게 하기 위해서입니다. 그 경로에 눈에 보이지 않는 글자가 섞여 들어가면 codex 는 설정을 읽지 못해 시작하자마자 죽고, claude 는 신뢰 확인 창에서 멈춥니다. 화면에는 멀쩡한 경로로 보이므로 눈으로는 찾지 못합니다.

2.7.7-22 부터 toolly doctor 가 이것을 찾아 ^H 처럼 드러내 보여 줍니다. 손으로 확인하려면:

```
sed -n '1,8p' ~/.codex/config.toml | cat -A
# [projects."..."] 줄에 ^H 나 ^I 가 보이면 그 줄이 원인입니다
```

고치는 순서입니다. 깨진 등록을 지운 뒤 경로를 다시 정하면 다음 기동에서 새로 등록됩니다. 두 파일을 손보는 일이라 먼저 내려 두고 하십시오.

```
toolly down -y
```

1) codex 쪽입니다. 편집기로 열어 제어문자가 보이는 [projects."..."] 줄과 바로 아래 trust_level 줄을 함께 지웁니다. 다른 항목은 건드리지 마십시오.

```
cp ~/.codex/config.toml ~/.codex/config.toml.bak
nano ~/.codex/config.toml      # 또는 쓰시는 편집기
```

2) claude 쪽도 같습니다. ~/.claude.json 의 projects 에서 같은 경로의 키를 지웁니다. JSON 이라 심표가 어긋나기 쉬우니 먼저 복사해 두십시오. 어느 키가 깨졌는지는 아래 한 줄이 짚어 줍니다.

```
cp ~/.claude.json ~/.claude.json.bak
grep -n 'projects' ~/.claude.json | cat -v | head
# ^H 나 ^I 가 보이는 키가 지울 대상입니다
```

3) 경로를 다시 정하고 올립니다. 경로는 붙여넣기 말고 직접 입력하십시오 — 보이지 않는 글자는 대개 붙여넣기로 들어옵니다.

```
toolly setup --project ~/work
toolly doctor          # 경고가 사라졌는지 확인
toolly sync-rules && toolly up
```

지우기가 어려우시면 배포처로 연락 주십시오. 두 파일만 보면 되는 일이라 원격 없이 안내해 드릴 수 있습니다.

판을 올려도 이미 저장된 값과 이미 쓰인 등록은 사라지지 않습니다. 새 판은 「앞으로 그런 값이 들어가는 것」을 막을 뿐입니다.

프로젝트 폴더를 잘못 잡았다

홈 디렉터리 자체는 프로젝트 폴더로 쓸 수 없습니다. toolly 가 그 안에 data/ 와 plans/ 와 프로젝트 메모리를 만들고, 나중에 uninstall 이 그것들을 지우기 때문입니다. 홈 아래 한 칸을 정하십시오.

```
toolly setup --project ~/work
toolly sync-rules
```

설치물이 바뀌었다고 한다

toolly seal verify 가 어떤 파일이 달라졌는지 알려 줍니다. 업그레이드나 설정 변경처럼 정당한 이유로 바뀐 것이면 다시 봉인하십시오. 짝이는 데가 없으면 그대로 두고 배포처에 알려 주십시오 — 다시 봉인하면 증거가 사라집니다.

```
toolly seal verify
toolly seal write  # 확인한 뒤에만
```

기록이 의심스럽다

패널 사이에 오간 모든 메시지는 서명된 채로 남습니다. toolly audit 가 중간에 고쳐지거나 빠진 것이 없는지 전부 대조합니다. 무엇이 실제로 오갔는지는 toolly report 로 읽으십시오.

되돌릴 수 없는 것

worktree reset 그 패널의 작업을 버리고 본선과 같게 만듭니다. 되돌릴 수 없습니다.

uninstall 프로젝트 폴더의 data/ 와 plans/ 까지 지웁니다. --dry-run 으로 먼저 보십시오.

둘 다 확인 절차가 있습니다. 확인 문구를 읽지 않고 넘기지 마십시오 — 지워진 뒤에는 배포처도 되살릴 수 없습니다.

12. 실제로 쓰는 법

여기까지가 설치와 관리였습니다. 이 장은 일을 시키는 쪽입니다. 핵심은 하나입니다 — 사람은 ORCH 에게만 말하고, 나머지는 ORCH 가 나눕니다.

하루가 이렇게 흘러갑니다

```
toolly up          # 팀을 띄우고 화면에 붙는다
(ORCH 패널에 하실 일을 말로 적는다)
Ctrl-b d          # 빠져나와도 팀은 계속 일한다
toolly summary     # 지금 누가 무엇을 들고 있는지
toolly attach      # 다시 들어간다
toolly report --days 1 # 오늘 무엇이 오갔는지
```

일을 어떻게 적어야 하는가

ORCH 에게는 평소 말투로 적으시면 됩니다. 다만 네 가지가 있으면 되묻지 않고 바로 나눕니다. 없으면 ORCH 가 먼저 물어봅니다.

무엇을	만들거나 고칠 것. "주문 API 에 중복 방지를 넣어라"
어디까지	건드려도 되는 파일과 범위. 넓게 주면 넓게 고칩니다.
끝난 기준	무엇이 되면 끝인가. 테스트가 통과하는 것인지, 화면이 뜨는 것인지.
하지 말 것	건드리면 안 되는 것. 계약·스키마·배포 같은 것들.

특정 역할에게 바로 쫓고 싶을 때만 send 를 씁니다. 평소에는 ORCH 에게 말하는 편이 낫습니다 — 검증까지 붙여서 돌려주기 때문입니다.

결과를 검증을 거쳐 옵니다

작업자가 끝냈다고 해서 끝이 아닙니다. 검증 역할이 같은 명령을 스스로 다시 돌려 보고, 통과해야 작업이 닫힙니다. 통과하지 못하면 무엇이 왜 미달인지와 함께 작업자에게 되돌아갑니다. 사람이 보는 것은 그 뒤입니다.

같은 일을 두 번 되돌려 보내고도 안 되면 ORCH 가 사람에게 묻습니다. 그때는 판단이 필요한 것이지 에이전트가 게으른 것이 아닙니다.

팀을 고르는 기준

역할 이름이 아니라 「무엇을 만드는가」로 고르십시오. toolly team 을 치면 팀마다 설명과 역할, 사람이 해야 하는 단계가 함께 나옵니다.

코드를 만든다	web-development · app-development · game-development · api
글을 만든다	publishing · education · marketing · business
조사하고 판단한다	research · idea-planning · design
시스템을 바꾼다	operations — 승인 단계가 강제됩니다
판단이 무거운 일	hybrid-code · hybrid-doc — 검증을 서로 다른 계열로 두 번 합니다

팀은 언제든지 바꿀 수 있습니다. 바꾸면 역할·검증자·흐름이 한 벌로 바뀝니다. 하던 작업이 있으면 먼저 정리하십시오.

토큰을 아끼거나 더 쓰고 싶을 때

같은 팀을 더 센 모델로 돌리거나 더 싸게 돌릴 수 있습니다. 검증 자리의 강도는 일정 아래로 내려가지 않습니다 —

검증이 무너지면 나머지가 의미가 없기 때문입니다.

```
toolly team                # 지금 팀과 고를 수 있는 것
toolly team research --tier max
toolly team --tier low      # 팀은 그대로, 강도만
```

팀 이름만 주고 tier 를 주지 않으면 그 팀의 기본값으로 돌아갑니다. 골라 두셨던 tier 는 지워지고 화면이 그 사실을 알려 줍니다.

작업을 합치기

코드 팀은 패널마다 자기 브랜치에서 일합니다. 서로의 파일을 덮지 않기 위해서입니다. 검증을 통과한 것만 본선으로 올리십시오.

```
toolly worktree status     # 패널별로 무엇이 얼마나 바뀌었는지
toolly worktree merge      # 통과한 것을 본선에 합친다
toolly worktree sync       # 본선의 최신을 각 패널로 내린다
```

사람이 해야 하는 단계

배포·스토어 제출·실기기 플래싱·촬영처럼 되돌릴 수 없거나 기계가 할 수 없는 일은 가드가 막습니다. 막는 것이 고장이 아닙니다. 그 자리에서 에이전트는 무엇을 해야 하는지 적어 사람에게 넘깁니다. toolly team 의 팀 설명에 「사람이 하는 단계 있음」으로 미리 표시됩니다.

문의 support@mytoolly.com · 이 메일에 그대로 답장하셔도 됩니다.

사이트 <https://multiagentharness.kr>

Toolly Multi-Agent Harness 2.7.7