

AUNOVA 수동 사전 설치 및 통합 설정 가이드

Codex CLI · Claude Code CLI · Antigravity CLI · Obsidian · GitHub · Google Drive · NAS

AUNOVA Harness

2026-07-21

차례

문서 목적	4
문서의 위치	4
파일 2와 파일 3의 순서	5
1. 먼저 알아야 할 구분	6
1.1 Obsidian만 설치할 때와 통합 환경을 만들 때	6
1.2 설치 중요도 표기	6
1.3 Windows 권장 기준	6
2. 설치 전 전체 체크리스트	7
2.1 계정	7
2.2 경로 메모	7
3. Windows 기본 도구 설치	8
3.1 WinGet 확인	8
역할	8
설치 여부 확인	8
WinGet이 없을 때	8
등록 문제 복구	8
3.2 Windows Terminal	8
중요도	8
설치	8
확인	8
3.3 PowerShell 7	9
중요도	9
WinGet 설치	9
확인	9
실행 정책	9
3.4 Git	9
중요도	9
설치	9
설치 옵션 권장	10
확인	10
사용자 설정	10
안전 기본값	10
3.5 GitHub CLI	10
중요도	10
설치	10
확인	10
로그인	10
3.6 Node.js LTS	11
중요도	11
설치	11
확인	11
권장 원칙	11
3.7 Python	11
중요도	11
설치	11
설치 프로그램 옵션	11
확인	12
프로젝트별 가상환경	12
3.8 Visual Studio Code 또는 다른 편집기	12
중요도	12
설치	12
4. 세 CLI 설치 순서	13
5. Codex CLI 설치	14
5.1 공식 문서	14

5.2 설치 방식	14
5.3 확인	14
5.4 로그인	14
5.5 초기 확인 명령	14
5.6 Windows 샌드박스	14
5.7 테스트	15
6. Claude Code CLI 설치	16
6.1 공식 문서	16
6.2 Windows PowerShell 설치	16
6.3 Git Bash 연동	16
6.4 확인	16
6.5 로그인	16
6.6 프로젝트 실행	16
6.7 Skills 위치	17
7. Antigravity CLI 설치	18
7.1 공식 문서	18
7.2 설치	18
7.3 확인	18
7.4 로그인	18
7.5 프로젝트 실행	18
7.6 Antigravity IDE와 CLI 구분	18
8. 세 CLI 공통 검증	19
8.1 명령 확인	19
8.2 실행 파일 위치	19
8.3 새 터미널에서 재확인	19
8.4 읽기·쓰기 테스트	19
8.5 CLI Registry 예시	19
8A. Multi-PC · Multi-RAW 설치 전 설계	20
8A.1 경로를 PC마다 다르게 두는 원칙	20
메인 데스크톱	20
D 드라이브가 없는 PC	20
8A.2 PC별 로컬 Device Profile	20
8A.3 Canonical Project와 여러 RAW Workspace	21
8A.4 NAS의 정확한 역할	21
9. Obsidian 설치	22
9.1 공식 다운로드	22
9.2 Windows 설치	22
9.3 Vault 선택	22
9.4 NAS에 Vault를 돌지 여부	22
9A. PC별 Obsidian Vault 설정과 동기화	23
9A.1 Vault 생성	23
9A.2 Vault 내부 링크	23
9A.3 Vault의 PC 간 동기화	23
10. AUNOVA Obsidian 폴더 생성	24
10.1 기본 구조	24
10.2 PowerShell 생성 예시	24
10.3 기본 설정 파일	25
11. 기존 프로젝트 등록	26
11.1 프로젝트 등록 원칙	26
11.2 프로젝트 설정 예시	26
11.3 프로젝트 내부 최소 구조	26
11A. 여러 RAW Workspace 수동 등록	27
11A.1 현재 RAW 폴더에 식별 파일 생성	27

11A.2 Device Profile에 경로 등록	27
11A.3 동일 프로젝트 동시 작업	27
12. GitHub 연결	28
12.1 기존 저장소 확인	28
12.2 .gitignore 기본 예시	28
12.3 원격 연결	28
12.4 하네스 정책	28
13. Google Drive 연결	29
13.1 폴더 역할	29
13.2 동기화 원칙	29
13.3 인증	29
14. NAS 설정	30
14.1 접근 확인	30
14.2 드라이브가 연결되지 않았을 때	30
14.3 NAS의 역할	30
15. Obsidian 기본 설정	31
15.1 Files & Links	31
15.2 Core Plugins	31
15.3 Templates	31
15.4 Properties	31
16. 세 CLI와 Obsidian 연동	32
16.1 역할 분리	32
16.2 CLI에 Vault 전체를 주지 않기	32
16.3 Context Pack 예	32
17. 다중 CLI 오케스트레이션 수동 테스트	33
17.1 계획 라운드	33
17.2 교차검토	33
17.3 실행자와 검토자	33
17.4 충돌 방지	33
17A. 업무시작·업무마무리 수동 테스트	34
17A.1 업무시작	34
17A.2 업무마무리	34
17A.3 동기화 결과 예	34
18. 통합 검증 체크리스트	35
18.1 기본 도구	35
18.2 CLI	35
18.3 경로	35
18.3A Multi-PC · Multi-RAW	35
18.4 GitHub	35
18.5 Obsidian	35
18.6 최종 READY 판정	35
19. 대표 문제와 해결	37
19.1 명령을 찾을 수 없음	37
19.2 PowerShell 스크립트 차단	37
19.3 Git 줄바꿈 문제	37
19.4 NAS 속도 문제	37
19.5 Obsidian 중복·충돌	37
19.6 CLI가 너무 많은 파일을 읽음	37
19.7 D 드라이브가 없는 PC	37
19.8 다른 PC에서 최신 RAW가 보이지 않음	38
19.9 같은 프로젝트의 RAW 폴더가 여러 개	38
20. macOS 보충 설치	39

기본 도구	39
21. Linux 보충 설치	40
공통	40
22. 설치 자동화를 더 쉽게 사용하는 방법	41
파일 2: 세 CLI 자동 설치	41
파일 3: Obsidian 자동 설치·연동	41
자동 설치 안전 규칙	41
23. 최종 권장 설치 시나리오	42
개인 Windows PC + NAS + GitHub + Drive	42
NAS를 쓰지 않는 사용자	42
GitHub를 쓰지 않는 프로젝트	42
모든 외부 채널을 건너뛰는 세션	42
24. 공식 링크 모음	43
Windows·터미널	43
개발 기본 도구	43
CLI	43
Obsidian	43
25. 문서 업데이트 정책	44
부록 A. Windows 한 번에 확인하는 명령	45
부록 B. 경로 검사 예시	46
부록 C. 설치 기록 템플릿	47
참고문헌 및 검증 기준	48

문서 목적

이 문서는 AUNOVA의 **다중 바이브코딩 툴 + 다중 CLI + Obsidian + 선택형 저장소** 환경을 사용자가 직접 설치하고 설정할 때 사용하는 수동 기준 가이드다.

이 문서에서 구축하려는 최종 흐름은 다음과 같다.

```

바이브코딩 툴
  ↓ 사용자와 대화·계획·결과 표시
Harness
  ↓ 프로젝트·권한·순서·검증 통제
Codex CLI / Claude Code CLI / Antigravity CLI
  ↓ 탐색·수정·테스트·상호 검토
Obsidian
  ↓ 지식·결정·프로젝트 상태·Handoff
현재 PC의 RAW Workspace
  ↓ 실제 작업
NAS Sync Hub / GitHub / Google Drive
  ↓ PC 간 전달·버전·공유
PC별 로컬 Obsidian Vault
  ↓ 지식·결정·상태·Handoff
  
```

문서의 위치

이 문서는 전체 파일 체계에서 **파일 6**이다.

- 01 전문가 이론·지식 그물망 아키텍처
- 02 세 CLI 자동 설치 지시 파일
- 03 Obsidian 자동 설치·연동 지시 파일
- 04 Obsidian·Harness 폴더 및 Skill 구조
- 05 다중 바이브코딩·다중 CLI 오케스트레이션
- 06 수동 사전 설치 및 통합 설정 가이드 ← 현재 문서
- 07 Codex용 Obsidian·다중 CLI 하네스

파일 2와 파일 3의 순서

설치와 연동은 다음 순서로 진행한다.

1. 운영체제·터미널·Git 등 기본 도구 준비
2. Codex CLI 설치
3. Claude Code CLI 설치
4. Antigravity CLI 설치
5. 세 CLI 실행·로그인·권한 검증
6. Obsidian 설치
7. Vault와 AUNOVA 폴더 생성
8. 프로젝트·CLI·저장소 연동

따라서 자동화 파일의 번호도 다음처럼 정리한다.

02_AUTO_INSTALL_ALL_CLI_FOR_VIBE_CODING.md
03_AUTO_INSTALL_OBSIDIAN_FOR_VIBE_CODING.md

1. 먼저 알아야 할 구분

1.1 Obsidian만 설치할 때와 통합 환경을 만들 때

Obsidian 앱 자체를 설치하는 데 Git, Node.js, Python이 반드시 필요한 것은 아니다.

하지만 다음 기능을 모두 사용하려면 추가 프로그램이 필요하거나 강하게 권장된다.

- 세 CLI 설치 및 실행
- 프로젝트 폴더 탐색과 수정
- Git 체크포인트와 브랜치
- 하네스 자동 설치 스크립트
- Markdown·PDF·문서 변환
- 프로젝트 상태 검사
- NAS와 로컬 폴더 동기화
- 다중 CLI 오케스트레이션

1.2 설치 중요도 표기

표기	의미
필수	통합 시스템 기본 실행에 필요
강력 권장	없어도 일부 실행되지만 기능·안전성이 떨어짐
선택	프로젝트 유형에 따라 설치
조건부	특정 CLI·개발 스택·자동화에서만 필요

1.3 Windows 권장 기준

이 가이드는 사용자의 현재 환경을 고려해 Windows를 중심으로 작성한다.

권장 기준:

- Windows 11 권장
- Windows 10은 1809 이상
- 64비트 x64 또는 ARM64
- 관리자 권한을 사용할 수 있는 개인 PC 또는 승인된 회사 PC
- 인터넷 연결
- 최소 10GB 이상의 설치·캐시 여유 공간
- 프로젝트와 Vault를 저장할 별도 여유 공간

회사에서 관리하는 PC는 방화벽, 실행 정책, 관리자 권한, 로컬 사용자 생성 정책 때문에 일부 설치나 샌드박스 설정이 제한될 수 있다.

2. 설치 전 전체 체크리스트

2.1 계정

다음 계정을 준비한다.

계정	용도	필수 여부
OpenAI/ChatGPT	Codex CLI 로그인	Codex 사용 시 필수
Anthropic/Claude	Claude Code 로그인	Claude Code 사용 시 필수
Google 계정	Antigravity 로그인, Drive 사용	Antigravity 사용 시 필수
GitHub 계정	원격 저장소·PR·Issue	GitHub 사용 시 필수
NAS 사용자 계정	NAS 폴더 접근	NAS 사용 시 필수

보안 원칙:

- 비밀번호와 API 키를 Markdown 파일에 기록하지 않는다.
- .env, 토큰, 인증 파일을 Obsidian Vault나 GitHub에 올리지 않는다.
- 브라우저 로그인 또는 운영체제의 보안 자격 증명 저장소를 우선 사용한다.
- 공용 PC에서는 로그인 상태를 남기지 않는다.

2.2 경로 메모

설치 전에 다음 경로를 정한다.

Obsidian Vault 경로:

예) D:\Obsidian\AUNOVA_AI_OS

프로젝트 기본 경로:

예) Z:\HDD1\yongusb\ aunova

또는 D:\Projects

작업용 임시 경로:

예) D:\AUNOVA_WORK

백업 경로:

예) Z:\HDD1\yongusb\ aunova_backup

경로 원칙:

- 경로를 하네스 파일에 하드코딩하지 않는다.
- 사용자별 절대경로는 `paths.local.yaml`에 저장한다.
- 공통 프로젝트 설정에는 저장소 ID와 상대경로만 저장한다.
- NAS 드라이브 문자가 달라져도 저장소 루트 한 곳만 수정하게 한다.

3. Windows 기본 도구 설치

3.1 WinGet 확인

역할

WinGet은 Windows에서 프로그램을 검색·설치·업그레이드·제거하는 공식 패키지 관리자다.

설치 여부 확인

PowerShell 또는 명령 프롬프트에서 실행한다.

```
winget --version
```

정상 출력 예:

```
v1.x.x
```

WinGet이 없을 때

1. Microsoft Store를 연다.
2. **앱 설치 관리자(App Installer)**를 검색한다.
3. 설치 또는 업데이트한다.
4. Windows에서 로그아웃 후 다시 로그인하거나 재부팅한다.
5. 다시 `winget --version`을 실행한다.

공식 안내:

- <https://learn.microsoft.com/windows/package-manager/winget/>
- <https://apps.microsoft.com/detail/9nblggh4nns1>

등록 문제 복구

최신 Windows에서 App Installer가 설치되어 있는데 WinGet이 인식되지 않으면 PowerShell에서 다음 명령을 시도한다.

```
Add-AppxPackage -RegisterByFamilyName -MainPackage Microsoft.DesktopAppInstaller_8wekyb3d8bbwe
```

3.2 Windows Terminal

중요도

강력 권장.

PowerShell, 명령 프롬프트, Git Bash, WSL을 탭으로 사용할 수 있어 다중 CLI 관리가 쉬워진다.

설치

```
winget install --id Microsoft.WindowsTerminal --exact
```

공식 다운로드:

- <https://apps.microsoft.com/detail/9n0dx20hk701>
- <https://learn.microsoft.com/windows/terminal/>

확인

시작 메뉴에서 Terminal을 실행한다.

권장 설정:

- 기본 프로필: PowerShell 7
- 시작 디렉터리: 사용자 홈 또는 프로젝트 기본 폴더
- 복사 시 일반 텍스트 사용
- 긴 출력이 잘리지 않도록 스크롤 기록 확대

3.3 PowerShell 7

중요도

필수에 가까운 강력 권장.

Windows에는 Windows PowerShell 5.1이 기본 설치되어 있지만, 자동화와 최신 CLI 실행에는 PowerShell 7 사용을 권장한다. PowerShell 7은 기존 5.1을 제거하지 않고 나란히 설치된다.

WinGet 설치

```
winget install --id Microsoft.PowerShell --source winget
```

MSI 형식이 필요한 경우:

```
winget install --id Microsoft.PowerShell --source winget --installer-type wix
```

공식 안내:

- <https://learn.microsoft.com/powershell/scripting/install/install-powershell-on-windows>
- <https://github.com/PowerShell/PowerShell/releases>

확인

새 터미널을 열고 실행한다.

```
pwsh --version
```

또는 PowerShell 7 내부에서:

```
$PSVersionTable
```

실행 정책

설치 스크립트나 npm 명령에서 다음과 같은 오류가 날 수 있다.

```
running scripts is disabled on this system
```

먼저 현재 정책을 확인한다.

```
Get-ExecutionPolicy -List
```

개인 PC에서 현재 사용자 범위에만 적용하려면:

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

주의:

- 회사 PC에서는 임의로 변경하지 말고 관리자에게 문의한다.
- Unrestricted나 Bypass를 전역으로 상시 설정하지 않는다.
- 문제가 해결되면 필요한 최소 범위로 유지한다.

3.4 Git

중요도

필수.

Git은 코드뿐 아니라 문서·프롬프트·설정 변경을 체크포인트로 남기고, CLI가 수정한 내용을 비교·복구하는 기준이 된다.

설치

```
winget install --id Git.Git --exact
```

공식 다운로드:

- <https://git-scm.com/install/>
- <https://git-scm.com/download/win>

설치 옵션 권장

Git for Windows 설치 프로그램에서:

- 기본 편집기: 사용 중인 편집기 또는 Visual Studio Code
- PATH: 명령 프롬프트와 타사 소프트웨어에서 Git 사용
- HTTPS 전송: OpenSSL 기본값 권장
- 줄바꿈: Windows checkout, Unix commit 기본 권장
- 터미널: Windows Terminal 또는 MinTTY 선택 가능
- Credential Manager: 활성화 권장

확인

```
git --version
```

사용자 설정

```
git config --global user.name "사용자 이름"
git config --global user.email "GitHub에 사용할 이메일"
```

확인:

```
git config --global --list
```

안전 기본값

```
git config --global init.defaultBranch main
git config --global pull.rebase false
git config --global core.autocrlf true
```

팀이나 기존 저장소 규칙이 있으면 프로젝트 규칙을 우선한다.

3.5 GitHub CLI

중요도

GitHub를 사용할 때 강력 권장.

GitHub CLI는 터미널에서 저장소, PR, Issue, Actions, 인증을 다룬다.

설치

```
winget install --id GitHub.cli --exact
```

공식 링크:

- <https://cli.github.com/>
- <https://cli.github.com/manual/>

확인

```
gh --version
```

로그인

```
gh auth login
```

권장 선택:

GitHub.com

HTTPS

브라우저 로그인

GitHub 자격 증명으로 Git 설정 허용

확인:

```
gh auth status
```

3.6 Node.js LTS

중요도

조건부 강력 권장.

현재 세 CLI는 각자의 네이티브 설치 경로를 제공하지만, Node.js는 다음 상황에서 유용하다.

- 기존 프로젝트가 npm, pnpm, yarn을 사용
- 일부 MCP·Skill·자동화 스크립트 실행
- Markdown/문서 도구 실행
- 프론트엔드 프로젝트 테스트
- npm 방식으로 Codex를 설치하거나 복구

설치

```
winget install --id OpenJS.NodeJS.LTS --exact
```

공식 다운로드:

- <https://nodejs.org/en/download>

확인

```
node --version
```

```
npm --version
```

권장 원칙

- 특별한 이유가 없으면 LTS 사용
- 여러 프로젝트가 서로 다른 Node 버전을 요구하면 버전 관리자 사용
- 전역 npm 패키지는 최소화
- 관리자 PowerShell에서 무조건 npm을 실행하지 않는다

3.7 Python

중요도

조건부 강력 권장.

Python은 폴더 분석, PDF·이미지·데이터 처리, 자동화, 테스트 스크립트에 유용하다.

설치

현재 지원되는 버전을 설치한다. 예시:

```
winget search Python.Python
```

설치 예시:

```
winget install --id Python.Python.3.14 --exact
```

공식 다운로드:

- <https://www.python.org/downloads/>
- <https://www.python.org/downloads/windows/>

설치 프로그램 옵션

수동 설치 시 다음을 확인한다.

- Add Python to PATH
- Install launcher for all users 또는 현재 사용자
- pip 포함

확인

```
python --version
py --version
pip --version
```

프로젝트별 가상환경

```
python -m venv .venv
.\.venv\Scripts\Activate.ps1
python -m pip install --upgrade pip
```

가상환경과 대형 패키지 폴더는 Git에 올리지 않는다.

```
.venv/
__pycache__/
*.pyc
```

3.8 Visual Studio Code 또는 다른 편집기

중요도

선택이지만 권장.

Obsidian은 지식과 상태를 관리하고, 코드 편집과 터미널 확인은 VS Code 또는 현재 사용하는 바이브코딩 툴이 담당한다.

설치

```
winget install --id Microsoft.VisualStudioCode --exact
```

공식 다운로드:

- <https://code.visualstudio.com/download>

권장 확장:

- Markdown All in One
- YAML
- GitLens 또는 기본 Git 기능
- 프로젝트에서 필요한 언어 확장

확장은 너무 많이 설치하지 않는다. CLI와 바이브코딩 툴이 동일 기능을 중복 수행하면 충돌과 리소스 사용이 늘어난다.

4. 세 CLI 설치 순서

설치 순서:

Codex CLI

→ Claude Code CLI

→ Antigravity CLI

→ 공통 실행 검증

→ Obsidian 설치·연동

세 CLI를 모두 설치하는 이유:

- 동일 작업에 대해 서로 다른 계획을 제안할 수 있다.
- 한 CLI의 계획을 다른 CLI가 검토할 수 있다.
- 실행 담당과 검증 담당을 분리할 수 있다.
- 특정 CLI 장애 시 다른 CLI로 전환할 수 있다.
- 작업 성격에 맞는 실행자를 선택할 수 있다.

하지만 매 작업마다 세 CLI를 모두 호출하는 것은 아니다.

간단 작업 → 1개 CLI

일반 작업 → 1개 실행 + 1개 검토

복잡 작업 → 2~3개 계획 비교

중요 작업 → 3개 계획 + 교차검토 + 독립검증

5. Codex CLI 설치

5.1 공식 문서

- CLI: <https://developers.openai.com/codex/cli>
- Windows: <https://developers.openai.com/codex/windows/windows-sandbox>
- 설정: <https://developers.openai.com/codex/config-basic>
- AGENTS.md: <https://developers.openai.com/codex/guides/agents-md>

5.2 설치 방식

공식 CLI 페이지의 운영체제 탭에서 최신 설치 방식을 확인한다.

macOS·Linux

```
curl -fsSL https://chatgpt.com/codex/install.sh | sh
```

Windows Windows에서는 공식 CLI 페이지의 **Windows 탭** 또는 공식 설치 프로그램을 우선 사용한다.

Node.js가 설치되어 있고 npm 방식을 선택할 경우 공식 페이지의 npm 탭에서 최신 패키지명을 확인한 뒤 설치한다. 일반적으로 다음 형식이 사용된다.

```
npm install -g @openai/codex
```

설치 방식은 업데이트될 수 있으므로 명령 실행 전에 반드시 공식 페이지의 현재 Windows/npm 탭과 일치하는지 확인한다.

5.3 확인

```
codex --version
```

5.4 로그인

프로젝트 폴더에서 실행한다.

```
cd "Z:\HDD1\yongusb\anovaa\프로젝트명"
codex
```

첫 실행에서 ChatGPT 로그인 또는 제공되는 인증 방식을 선택한다.

5.5 초기 확인 명령

Codex 내부에서:

```
/status
/permissions
/model
```

프로젝트에 AGENTS.md가 없다면:

```
/init
```

다만 AUNOVA 하네스에서는 자동 생성 전에 기존 AGENTS.md와 프로젝트 규칙을 먼저 확인한다.

5.6 Windows 샌드박스

권장:

```
[windows]
sandbox = "elevated"
```

관리 PC에서 elevated 설정이 막히면:

```
[windows]
sandbox = "unelevated"
```

원칙:

- 기본은 프로젝트 폴더 내부만 쓰기
- 삭제·외부 업로드·배포는 승인 필요
- Full Access를 상시 사용하지 않음
- NAS 읽기 권한은 필요한 프로젝트 경로만 허용

5.7 테스트

테스트 프로젝트에서:

이 프로젝트의 폴더 구조를 읽기 전용으로 분석하고 파일은 수정하지 마세요.

그 다음 작은 테스트 파일을 만들어 쓰기 권한을 검증한다.

6. Claude Code CLI 설치

6.1 공식 문서

- 개요: <https://docs.anthropic.com/en/docs/claude-code/overview>
- 설치: <https://docs.anthropic.com/en/docs/claude-code/setup>
- 빠른 시작: <https://docs.anthropic.com/en/docs/claude-code/quickstart>
- 설정: <https://docs.anthropic.com/en/docs/claude-code/settings>
- Skills: <https://docs.anthropic.com/en/docs/claude-code/skills>

6.2 Windows PowerShell 설치

```
irm https://claude.ai/install.ps1 | iex
```

WinGet 설치 방식:

```
winget install Anthropic.ClaudeCode
```

둘 중 하나만 사용한다.

- 네이티브 설치: 자동 업데이트가 편리
- WinGet: 설치·업그레이드 관리가 명확

6.3 Git Bash 연동

Git for Windows가 설치되어 있으면 Claude Code가 Bash 도구를 사용할 수 있다.

자동 인식이 안 되면 Claude Code 설정에 경로를 지정한다.

```
{
  "env": {
    "CLAUDE_CODE_GIT_BASH_PATH": "C:\\Program Files\\Git\\bin\\bash.exe"
  }
}
```

Git Bash가 없어도 PowerShell 도구로 실행할 수 있다.

6.4 확인

```
claude --version
```

6.5 로그인

```
claude
```

브라우저 로그인을 완료한다.

다시 로그인:

```
/login
```

로그아웃:

```
/logout
```

6.6 프로젝트 실행

```
cd "Z:\HDD1\yongusb\anovaa\프로젝트명"
claude
```

첫 요청 예:

파일을 수정하지 말고 프로젝트의 구조와 현재 Git 상태만 분석하세요.

6.7 Skills 위치

프로젝트 범위 예:

`.claude/skills/<skill-name>/SKILL.md`

AUNOVA 공통 Skill을 설치할 때는 도구 전용 형식과 공통 원본을 구분한다.

7. Antigravity CLI 설치

7.1 공식 문서

- 제품: <https://antigravity.google/product/antigravity-cli>
- 설치: <https://antigravity.google/docs/cli-install>
- 공식 사이트: <https://antigravity.google/>
- 실습: <https://codelabs.developers.google.com/antigravity-cli-hands-on>

7.2 설치

Antigravity CLI는 운영체제별 공식 설치 스크립트 또는 다운로드 페이지를 사용한다.

1. 공식 설치 페이지를 연다.
2. Windows 항목을 선택한다.
3. 제공되는 PowerShell 설치 명령 또는 설치 파일을 사용한다.
4. 설치 명령은 복사 전에 도메인이 antigravity.google인지 확인한다.
5. 설치 완료 후 새 터미널을 연다.

공식 설치 페이지의 명령은 업데이트될 수 있으므로 이 문서에 고정된 비공식 명령을 사용하지 않는다.

7.3 확인

```
agy --version
```

버전 옵션이 다르면:

```
agy --help
```

7.4 로그인

```
agy
```

첫 실행에서 다음 중 하나를 선택할 수 있다.

Google OAuth

Google Cloud 프로젝트 사용

개인 사용자는 일반적으로 Google OAuth가 간단하다.

7.5 프로젝트 실행

```
cd "Z:\HDD1\yongusb\anovaa\프로젝트명"
```

```
agy
```

읽기 전용 테스트:

현재 프로젝트 구조를 분석하고 파일을 변경하지 마세요.

7.6 Antigravity IDE와 CLI 구분

- Antigravity: 여러 에이전트를 관리하는 상위 앱
- Antigravity IDE: 편집기·터미널·에이전트 통합 환경
- Antigravity CLI: 터미널 중심 실행

AUNOVA 구조에서는 현재 사용 중인 바이브코딩 툴이 오케스트레이터가 되고, agy는 CLI 작업자 풀의 하나로 등록한다.

8. 세 CLI 공통 검증

8.1 명령 확인

```
codex --version
claude --version
agy --version
```

8.2 실행 파일 위치

```
Get-Command codex
Get-Command claude
Get-Command agy
```

8.3 새 터미널에서 재확인

설치 직후에는 PATH가 현재 터미널에 반영되지 않을 수 있다.

1. 모든 터미널을 닫는다.
2. Windows Terminal을 다시 연다.
3. 버전 명령을 재실행한다.
4. 그래도 안 되면 재부팅한다.

8.4 읽기·쓰기 테스트

테스트 폴더:

```
New-Item -ItemType Directory -Path "$HOME\AUNOVA_CLI_TEST" -Force
Set-Location "$HOME\AUNOVA_CLI_TEST"
"AUNOVA TEST" | Set-Content README.md
```

각 CLI에 다음을 요청한다.

README.md를 읽고 내용을 요약하되 파일은 수정하지 마세요.

쓰기 테스트는 별도 승인 후 진행한다.

TEST_RESULT.md를 만들고 도구 이름과 현재 시간을 기록하세요.

8.5 CLI Registry 예시

Obsidian 설치 이후 다음 설정을 만들 예정이지만, 수동 메모로 먼저 준비할 수 있다.

```
tools:
  codex:
    command: codex
    enabled: true
  claude_code:
    command: claude
    enabled: true
  antigravity:
    command: agy
    enabled: true
```

8A. Multi-PC · Multi-RAW 설치 전 설계

8A.1 경로를 PC마다 다르게 두는 원칙

메인 데스크톱과 다른 PC의 드라이브 구성이 같을 필요는 없다.

메인 데스크톱 Vault: D:\AUNOVA_AI_OS
다른 PC Vault: C:\AUNOVA_AI_OS
하네스 논리 Root: \${AUNOVA_AI_OS_ROOT}

각 PC에서 다음 User 환경변수를 설정할 수 있다.

메인 데스크톱

```
[Environment]::SetEnvironmentVariable(  
    "AUNOVA_AI_OS_ROOT",  
    "D:\AUNOVA_AI_OS",  
    "User"  
)
```

D 드라이브가 없는 PC

```
[Environment]::SetEnvironmentVariable(  
    "AUNOVA_AI_OS_ROOT",  
    "C:\AUNOVA_AI_OS",  
    "User"  
)
```

확인:

```
[Environment]::GetEnvironmentVariable("AUNOVA_AI_OS_ROOT", "User")
```

새 터미널을 연 뒤 \$env:AUNOVA_AI_OS_ROOT도 확인한다.

8A.2 PC별 로컬 Device Profile

다음 폴더를 만든다.

```
New-Item -ItemType Directory -Force "$HOME\.aunova"
```

메인 PC 예:

```
# %USERPROFILE%\.aunova/device.local.yaml  
device:  
  id: main-desktop  
  name: AUNOVA Main Desktop
```

```
paths:  
  aunova_ai_os_root: "D:/AUNOVA_AI_OS"  
  nas_root: "Z:/HDD1/yongusb/aunova"
```

```
workspaces: {}
```

다른 PC 예:

```
device:  
  id: laptop-01  
  name: AUNOVA Laptop
```

```
paths:  
  aunova_ai_os_root: "C:/AUNOVA_AI_OS"  
  nas_root: "Z:/HDD1/yongusb/aunova"
```

```
workspaces: {}
```

이 파일은 PC별 설정이다. GitHub에 커밋하거나 다른 PC의 값으로 덮어쓰지 않는다.

8A.3 Canonical Project와 여러 RAW Workspace

프로젝트 하나가 다음 여러 폴더에 존재할 수 있다.

```
D:\CodexProjects\MindWeather
D:\Antigravity\MindWeather
C:\ClaudeProjects\MindWeather
Z:\HDD1\yongusb\anovna\MindWeather
```

프로젝트를 한 폴더로 강제 이동하지 않고 다음처럼 등록한다.

```
project:
  id: mind-weather
  uuid: prj-mw-2026-001
  name: Mind Weather

workspaces:
  mw-main-codex:
    device_id: main-desktop
    path: "D:/CodexProjects/MindWeather"
  mw-main-antigravity:
    device_id: main-desktop
    path: "D:/Antigravity/MindWeather"
  mw-laptop-claude:
    device_id: laptop-01
    path: "C:/ClaudeProjects/MindWeather"
```

8A.4 NAS의 정확한 역할

NAS는 모든 PC가 같은 파일을 실시간 공동 편집하는 위치가 아니다.

업무시작

NAS 최신 Manifest·Handoff 확인

→ 로컬 RAW와 비교

→ 승인 후 Pull

업무마무리

로컬 Validation

→ Obsidian 상태 갱신

→ 승인 후 NAS Push

→ Manifest 갱신

NAS가 끊기면 로컬 작업은 가능하지만 다른 PC로 이어서 작업하기 전에 pending sync를 해결해야 한다.

9. Obsidian 설치

세 CLI가 정상 실행된 다음 Obsidian을 설치하고 연동한다.

9.1 공식 다운로드

- 다운로드: <https://obsidian.md/download>
- 설치 가이드: <https://obsidian.md/help/install>
- Help: <https://obsidian.md/help>

9.2 Windows 설치

1. <https://obsidian.md/download> 에 접속한다.
2. Windows의 Universal 설치 파일을 내려받는다.
3. 설치 파일을 실행한다.
4. Obsidian을 실행한다.
5. 기존 Vault 또는 새 Vault를 선택한다.

9.3 Vault 선택

새 Vault 권장 예:

D:\Obsidian\AUNOVA_AI_OS

기존 Vault 기존 Vault를 사용할 경우:

- 먼저 전체 백업
- 기존 폴더 이동 금지
- 누락 폴더만 추가
- .obsidian 설정을 무조건 덮어쓰지 않음
- 커뮤니티 플러그인은 사용자 승인 후 추가

9.4 NAS에 Vault를 둘지 여부

권장:

Vault = 로컬 SSD

프로젝트 원본·대용량 파일 = NAS

이유:

- Obsidian은 많은 작은 Markdown 파일을 자주 읽고 쓴다.
- NAS 연결 지연이나 끊김은 충돌과 인덱싱 문제를 만들 수 있다.
- Vault는 로컬에서 빠르게 쓰고 별도 방식으로 백업하는 편이 안정적이다.

NAS에 Vault를 두어야 한다면:

- 한 번에 한 기기에서만 쓰는 정책 검토
- 안정적인 네트워크 연결
- 파일 잠금과 동기화 충돌 확인
- 정기 스냅샷 활성화
- .obsidian/workspace*와 캐시 충돌 주의

9A. PC별 Obsidian Vault 설정과 동기화

9A.1 Vault 생성

현재 PC에서 환경변수 또는 Device Profile의 경로에 Vault를 만든다.

```
$Vault = [Environment]::GetEnvironmentVariable("AUNOVA_AI_OS_ROOT", "User")  
New-Item -ItemType Directory -Force $Vault
```

Obsidian에서 **Open folder as vault**를 선택해 해당 폴더를 연다.

9A.2 Vault 내부 링크

Vault 내부에서는 상대 링크를 사용한다.

```
[[10_Project_Index/MindWeather/PROJECT_OVERVIEW]]
```

PC 고정 절대경로는 노트 본문에 넣지 않는다.

9A.3 Vault의 PC 간 동기화

RAW 프로젝트의 NAS 동기화와 Vault 동기화는 별도 정책이다.

선택 가능:

- Obsidian Sync
- Private Git 기반 Markdown 동기화
- 사용자가 선택한 파일 동기화 방식
- PC별 독립 Vault 후 Handoff만 교환

공통 동기화 시 장치별 UI 상태는 제외하는 것을 권장한다.

```
.obsidian/workspace.json  
.obsidian/workspace-mobile.json  
.obsidian/cache  
.trash
```

device.local.yaml은 Vault 밖 %USERPROFILE%/.aunova/에 두므로 Vault 동기화 충돌을 피할 수 있다.

10. AUNOVA Obsidian 폴더 생성

10.1 기본 구조

```
AUNOVA_AI_OS/  
├── 00_System/  
│   ├── config/  
│   ├── projects/  
│   ├── tools/  
│   ├── adapters/  
│   ├── sessions/  
│   ├── locks/  
│   └── logs/  
├── 01_Inbox/  
│   ├── global/  
│   ├── pending/  
│   ├── conflicts/  
│   └── processed/  
├── 10_Project_Index/  
├── 20_Knowledge/  
├── 30_Workflows/  
├── 40_MOCs/  
├── 50_Templates/  
├── 70_Outputs/  
├── 80_AI/  
│   ├── harness/  
│   ├── skills/  
│   ├── adapters/  
│   └── prompts/  
└── 90_Archive/
```

10.2 PowerShell 생성 예시

Vault 경로를 수정한 뒤 실행한다.

```
$Vault = "D:\Obsidian\AUNOVA_AI_OS"
```

```
$Folders = @(  
    "00_System\config",  
    "00_System\projects",  
    "00_System\tools",  
    "00_System\adapters",  
    "00_System\sessions",  
    "00_System\locks",  
    "00_System\logs",  
    "01_Inbox\global",  
    "01_Inbox\pending",  
    "01_Inbox\conflicts",  
    "01_Inbox\processed",  
    "10_Project_Index",  
    "20_Knowledge",  
    "30_Workflows",  
    "40_MOCs",  
    "50_Templates",  
    "70_Outputs",  
    "80_AI\harness",  
    "80_AI\skills",  
    "80_AI\adapters",  
    "80_AI\prompts",  
    "90_Archive"  
)  
  
foreach ($Folder in $Folders) {  
    New-Item -ItemType Directory -Path (Join-Path $Vault $Folder) -Force | Out-Null  
}
```

실행 전에 경로를 반드시 확인한다.

10.3 기본 설정 파일

00_System/config/paths.local.yaml

```
obsidian:  
  vault_path: "D:/Obsidian/AUNOVA_AI_OS"  
  
storages:  
  aunova_nas:  
    type: nas  
    root: "Z:/HDD1/yongusb/aunova"  
    enabled: true  
  
  local_projects:  
    type: local  
    root: "D:/Projects"  
    enabled: false
```

이 파일은 사용자 컴퓨터별 절대경로를 포함하므로 Git 공유 여부를 신중히 결정한다.

00_System/config/system.yaml

```
system:  
  name: AUNOVA_AI_OS  
  version: 1  
  
workflow:  
  cli_before_obsidian_link: true  
  require_user_approval_for_write: true  
  require_validation: true  
  use_jit_context: true
```

00_System/tools/cli-registry.yaml

```
tools:  
  codex:  
    command: codex  
    enabled: true  
  claude_code:  
    command: claude  
    enabled: true  
  antigravity:  
    command: agy  
    enabled: true
```

11. 기존 프로젝트 등록

사용자 NAS 예:

Z:\HDD1\yongusb\ aunova

하위에 프로젝트별 폴더가 있다고 가정한다.

Z:\HDD1\yongusb\ aunova\ MindWeather

Z:\HDD1\yongusb\ aunova\ TrueBusan

Z:\HDD1\yongusb\ aunova\ AutoRepair

11.1 프로젝트 등록 원칙

- 프로젝트 전체를 Vault로 복사하지 않는다.
- 실제 파일은 원래 위치가 Source of Truth다.
- Obsidian에는 프로젝트 개요·상태·결정·Handoff를 둔다.
- 기존 프로젝트 구조는 강제로 바꾸지 않는다.
- .aunova와 Inbox는 승인 후 최소한으로 추가한다.

11.2 프로젝트 설정 예시

00_System/projects/mind-weather.yaml

```
project:
  id: mind-weather
  name: Mind Weather
  storage_id: aunova_nas
  relative_path: MindWeather

obsidian:
  note_root: "10_Project_Index/MindWeather"

folders:
  inbox: "01_Inbox"
  system: ".aunova"
  outputs: "outputs"

channels:
  github:
    enabled: true
  google_drive:
    enabled: true
  external_storage:
    enabled: true
```

11.3 프로젝트 내부 최소 구조

```
프로젝트/
├─ 기존 파일과 폴더/
├─ 01_Inbox/
│   ├─ pending/
│   ├─ processed/
│   ├─ conflicts/
│   └─ external/
└─ .aunova/
    ├─ project.yaml
    ├─ folder-map.yaml
    ├─ context/
    ├─ sessions/
    ├─ task-packets/
    ├─ handoff/
    ├─ validation/
    └─ logs/
```

자동 생성 전에 Dry Run 목록을 먼저 확인한다.

11A. 여러 RAW Workspace 수동 등록

11A.1 현재 RAW 폴더에 식별 파일 생성

```
$Raw = "D:\CodexProjects\MindWeather"
New-Item -ItemType Directory -Force "$Raw\.aunova\handoff"
New-Item -ItemType Directory -Force "$Raw\.aunova\validation"
New-Item -ItemType Directory -Force "$Raw\.aunova\conflicts"
```

.aunova/project.yaml:

```
project_id: mind-weather
project_uuid: prj-mw-2026-001
project_name: Mind Weather
```

.aunova/workspace.yaml:

```
workspace_id: mw-main-codex
device_id: main-desktop
tool_hint: codex
workspace_type: local-raw
```

11A.2 Device Profile에 경로 등록

```
workspaces:
  mw-main-codex:
    project_id: mind-weather
    path: "D:/CodexProjects/MindWeather"
    status: active
```

다른 RAW 폴더도 다른 workspace_id로 추가한다.

11A.3 동일 프로젝트 동시 작업

- 기본은 한 Workspace만 쓰기 활성화한다.
- 다른 Workspace는 읽기 전용 검토에 사용할 수 있다.
- Workstream이 완전히 다르면 Workstream Lock을 사용한다.
- 동일 파일의 양방향 변경은 자동 덮어쓰기하지 않는다.

12. GitHub 연결

12.1 기존 저장소 확인

프로젝트 폴더에서:

```
git status
```

Git 저장소가 아니면:

```
git init
```

하지만 기존 원격 저장소가 있다면 먼저 복제 또는 원격 연결 방식을 확인한다.

12.2 .gitignore 기본 예시

```
.env  
.env.*  
*.key  
*.pem  
node_modules/  
.next/  
dist/  
build/  
.venv/  
__pycache__/  
.aunova/logs/  
.aunova/context/  
01_Inbox/external/  
*.mp4  
*.mov  
*.psd  
*.blend
```

대용량 파일을 무조건 Git에 넣지 않는다.

12.3 원격 연결

```
git remote -v
```

새 원격 예:

```
git remote add origin https://github.com/사용자/저장소.git
```

첫 Push:

```
git add .  
git commit -m "Initialize AUNOVA project"  
git branch -M main  
git push -u origin main
```

12.4 하네스 정책

- Fetch는 자동 가능
- Pull은 로컬 변경 확인 후
- Commit은 변경 요약 확인 후
- Push는 사용자 승인 후
- Force Push는 기본 금지
- 브랜치 삭제는 사용자 승인 필수

13. Google Drive 연결

Google Drive는 코드 저장소가 아니라 다음에 사용한다.

- 사업계획서
- 발표자료
- PDF
- Excel
- 외부 공유 문서
- 검토용 산출물

13.1 폴더 역할

```
Google Drive/  
└─ AUNOVA/  
    └─ 프로젝트명/  
        ├── Incoming/  
        ├── Working-Documents/  
        ├── Deliverables/  
        └─ Archive/
```

13.2 동기화 원칙

- 코드 전체를 Drive와 GitHub에 동시에 동기화하지 않는다.
- Drive Desktop 동기화 폴더 안에서 개발 저장소를 운영하면 파일 잠금과 임시파일 충돌을 확인한다.
- 최종 문서 중심으로 업로드한다.
- 동일 이름 덮어쓰기보다 버전 또는 날짜 규칙을 사용한다.

예:

```
사업계획서_v03_2026-07-21.docx  
사업계획서_FINAL_2026-07-21.pdf
```

13.3 인증

바이브코딩 툴 또는 MCP에서 Google Drive를 연결할 때:

- 프로젝트별 폴더만 허용
- 전체 Drive 접근을 최소화
- 업로드 전 사용자 확인
- 공개 링크 생성은 별도 승인

14. NAS 설정

사용자 기본 경로:

Z:\HDD1\yongusb\apunova

14.1 접근 확인

Test-Path "Z:\HDD1\yongusb\apunova"

하위 폴더 목록:

Get-ChildItem "Z:\HDD1\yongusb\apunova" -Directory

14.2 드라이브가 연결되지 않았을 때

- 파일 탐색기에서 NAS 주소 확인
- VPN 필요 여부 확인
- NAS 전원·네트워크 확인
- 저장된 Windows 자격 증명 확인
- 드라이브 문자 변경 여부 확인

네트워크 경로 예:

\\NAS이름\HDD1\yongusb\apunova

드라이브 매핑 예:

net use Z: "\\NAS이름\HDD1" /persistent:yes

계정정보를 스크립트에 평문으로 넣지 않는다.

14.3 NAS의 역할

권장:

- 대용량 원본
- 영상·이미지·3D 파일
- ComfyUI 모델과 결과
- 프로젝트 백업
- 장기 보관

비권장:

- 작은 임시 파일을 초당 반복 생성하는 빌드 캐시
- node_modules
- .venv
- .next
- 컴파일 캐시
- 여러 CLI가 동시에 쓰는 임시 세션 파일

임시 실행은 로컬 SSD, 최종 결과와 원본은 NAS가 안정적이다.

15. Obsidian 기본 설정

15.1 Files & Links

권장:

- 새 링크 형식: 상대경로 또는 짧은 링크 중 하나를 일관되게 사용
- 링크 업데이트 자동화 활성화
- 첨부 파일 전용 폴더 지정
- 삭제 파일은 시스템 휴지통 또는 Vault의 Trash 사용

15.2 Core Plugins

초기 권장:

- Backlinks
- Outgoing links
- Graph view
- Properties view
- Templates
- Search
- Command palette
- Bookmarks
- File recovery

처음부터 커뮤니티 플러그인을 많이 설치하지 않는다.

15.3 Templates

템플릿 폴더:

50_Templates

권장 템플릿:

- Project Overview
- Current State
- Decision
- Task Packet
- Session Brief
- Handoff
- Validation Report
- Workflow Note

15.4 Properties

예:

```
type: project-state
project_id: mind-weather
status: "Multi-PC Multi-RAW Manual Guide"
updated: 2026-07-21
owner: user
source_of_truth: obsidian
```

Properties는 검색·자동화·Context Compiler가 읽을 수 있도록 일관되게 유지한다.

16. 세 CLI와 Obsidian 연동

16.1 역할 분리

바이브코딩 툴
= 사용자 대화·계획·오케스트레이션·결과 표시

CLI
= 탐색·수정·테스트·독립 검토

Obsidian
= 지식·결정·현재 상태·워크플로·Handoff

Harness
= 필요한 문맥만 선택·권한 통제·검증

16.2 CLI에 Vault 전체를 주지 않기

CLI 입력은 다음으로 제한한다.

SESSION_CONTEXT.md
TASK_PACKET.yaml
FILE_SCOPE.txt
VALIDATION_PLAN.md

16.3 Context Pack 예

```
task:
  id: MW-DEV-024
  objective: "로그인 화면 모바일 오류 수정"

project:
  root: "Z:/HDD1/yongusb/aunova/MindWeather"

context:
  notes:
    - "10_Project_Index/MindWeather/PROJECT_OVERVIEW.md"
    - "10_Project_Index/MindWeather/CURRENT_STATE.md"
    - "10_Project_Index/MindWeather/DEV_CONTEXT.md"

scope:
  read:
    - "app/src/login"
  write:
    - "app/src/login"
  exclude:
    - "node_modules"
    - "media"
    - "archive"
```

17. 다중 CLI 오케스트레이션 수동 테스트

17.1 계획 라운드

현재 바이브코딩 툴이 동일 Task Packet을 세 CLI에 전달한다.

각 CLI에는 다음만 요청한다.

파일을 수정하지 말고 실행 계획, 위험, 검증 방법을 제안하세요.

17.2 교차검토

계획 원문 전체 대신 구조화된 요약물 다른 CLI에 전달한다.

계획 A의 누락과 위험 검토

계획 B의 불필요한 범위 검토

계획 C의 테스트 적절성 검토

토론은 기본 1회, 최대 2회로 제한한다.

17.3 실행자와 검토자

예:

Codex CLI → 구현

Claude Code CLI → 구조·문서 검토

Antigravity CLI → 사용자 흐름·통합 검증

역할은 고정이 아니라 작업별로 배정한다.

17.4 충돌 방지

코드:

```
git worktree add ..\project-codex -b task/codex
git worktree add ..\project-claude -b task/claude
git worktree add ..\project-agy -b task/agy
```

문서:

```
proposal_codex_draft.md
proposal_claude_review.md
proposal_agy_review.md
```

동일 파일 동시 수정은 금지한다.

17A. 업무시작·업무마무리 수동 테스트

17A.1 업무시작

1. 현재 RAW 폴더의 project_id/workspace_id 확인
2. Device Profile에서 Vault Root 확인
3. Obsidian 최근 CURRENT_STATE/Handoff 확인
4. NAS 연결 확인
5. 로컬과 NAS Manifest 비교
6. 변경·삭제·충돌 목록 확인
7. 사용자 승인 후 NAS 변경을 로컬로 복사
8. Workspace Lock 생성
9. 작업 시작

NAS가 오프라인이면 READY_DEGRADED로 기록한다. 이전 PC의 NAS Push가 pending이라면 새 PC에서 쓰기 작업을 시작하기 전에 해결한다.

17A.2 업무마무리

1. 테스트·Validation 완료
2. Obsidian CURRENT_STATE와 Handoff 갱신
3. 로컬 Manifest 갱신
4. 사용자 승인 후 NAS로 변경 반영
5. NAS Manifest 갱신
6. GitHub·Drive 독립 동기화
7. 실패 채널 pending 기록
8. Lock 해제

17A.3 동기화 결과 예

```
sync_result:
  nas:
    status: success
  github:
    status: pending
    reason: auth_required
  google_drive:
    status: skipped
overall: completed_with_pending_sync
```

18. 통합 검증 체크리스트

18.1 기본 도구

```
winget --version
wt --version
pwsh --version
git --version
gh --version
node --version
npm --version
python --version
```

18.2 CLI

```
codex --version
claude --version
agy --version
```

18.3 경로

```
Test-Path "D:\Obsidian\AUNOVA_AI_OS"
Test-Path "Z:\HDD1\yongusb\ aunova"
```

18.3A Multi-PC · Multi-RAW

- ☐ 현재 PC Device ID가 고유함
- ☐ AUNOVA_AI_OS_ROOT가 이 PC의 실제 Vault 경로를 가리킴
- ☐ 현재 RAW 폴더의 project_id와 workspace_id가 등록됨
- ☐ 같은 프로젝트의 다른 RAW 폴더와 Workspace ID가 중복되지 않음
- ☐ NAS Manifest 비교 Dry Run 성공
- ☐ NAS 오프라인 시 pending 기록 성공
- ☐ Vault 내부 링크가 절대 드라이브 문자에 의존하지 않음

18.4 GitHub

```
gh auth status
git status
git remote -v
```

18.5 Obsidian

- Vault 열림
- 기본 폴더 보임
- 템플릿 폴더 지정됨
- 파일 복구 활성화
- Project Index에서 프로젝트 노트를 열 수 있음
- 링크 클릭이 정상 작동

18.6 최종 READY 판정

```
READY
- 3개 CLI 실행 가능
- Obsidian Vault 열림
- 프로젝트 경로 접근 가능
- 최소 1개 프로젝트 등록
- 읽기·쓰기 권한 검증
```

```
READY WITH OPTIONAL CHANNELS DISABLED
- GitHub, Drive, NAS 중 일부 미사용
- 핵심 CLI·Obsidian·프로젝트는 정상
```

```
READY WITH LIMITATIONS
```

- 특정 CLI 로그인 실패
- NAS 읽기만 가능
- 회사 정책으로 샌드박스 제한

NOT READY

- 프로젝트 경로 접근 불가
- Obsidian Vault 미생성
- 모든 CLI 실행 불가
- 설정 파일 손상

19. 대표 문제와 해결

19.1 명령을 찾을 수 없음

codex is not recognized
claude is not recognized
agy is not recognized

해결 순서:

1. 새 터미널 열기
2. Get-Command <명령> 실행
3. 설치 경로 확인
4. PATH 확인
5. 재설치
6. 재부팅

19.2 PowerShell 스크립트 차단

`Get-ExecutionPolicy -List`
`Set-ExecutionPolicy -Scope CurrentUser RemoteSigned`

회사 PC는 관리자 정책을 우선한다.

19.3 Git 줄바꿈 문제

Windows와 Linux를 교차 사용할 때 .gitattributes를 사용한다.

```
* text=auto
*.sh text eol=lf
*.ps1 text eol=crlf
*.md text eol=lf
```

19.4 NAS 속도 문제

- 임시 빌드 폴더를 로컬 SSD로 이동
- 대용량 파일만 NAS 저장
- 유선 네트워크 사용
- 동일 파일 동시 수정 금지
- 스냅샷 또는 버전 관리 활성화

19.5 Obsidian 중복·충돌

- 동일 Vault를 여러 동기화 서비스에 동시에 맡기지 않음
- .obsidian/workspace* 충돌 확인
- 파일명에 금지문자 사용하지 않음
- 링크 이름 변경 시 자동 업데이트 사용
- Vault 백업 후 구조 변경

19.6 CLI가 너무 많은 파일을 읽음

- Vault 전체 경로 전달 금지
- FILE_SCOPE.txt 사용
- 제외 목록 추가
- 현재 작업용 Context Pack만 제공
- 로그와 Archive 제외

19.7 D 드라이브가 없는 PC

증상:

D:\AUNOVA_AI_OS 경로를 찾을 수 없음

해결:

1. C 또는 다른 로컬 SSD 폴더를 선택한다.
2. AUNOVA_AI_OS_ROOT 환경변수를 새 경로로 설정한다.
3. %USERPROFILE%/.aunova/device.local.yaml을 갱신한다.
4. 하네스 문서의 논리 Root는 바꾸지 않는다.

19.8 다른 PC에서 최신 RAW가 보이지 않음

확인 순서:

1. 이전 PC의 업무마무리 Handoff 확인
2. 이전 PC의 pending-sync.yaml 확인
3. NAS Manifest 수정시각 확인
4. 현재 PC의 Workspace ID 확인
5. NAS Pull을 Dry Run으로 재실행

이전 PC에 pending NAS Push가 있으면 현재 PC에서 자동으로 오래된 NAS 사본을 최신으로 간주하지 않는다.

19.9 같은 프로젝트의 RAW 폴더가 여러 개

- 모두 같은 project_id를 사용한다.
- 각 폴더는 서로 다른 workspace_id를 사용한다.
- 업무시작 시 현재 활성 Workspace를 명시한다.
- 동일 파일 동시 수정은 금지한다.

20. macOS 보충 설치

기본 도구

Homebrew:

- <https://brew.sh/>

Git:

```
xcode-select --install
```

또는:

```
brew install git
```

PowerShell:

```
brew install --cask powershell
```

Node.js:

```
brew install node
```

Python:

```
brew install python
```

Obsidian:

- <https://obsidian.md/download>

Codex:

```
curl -fsSL https://chatgpt.com/codex/install.sh | sh
```

Claude Code:

```
curl -fsSL https://claude.ai/install.sh | bash
```

Antigravity CLI:

- <https://antigravity.google/docs/cli-install>

21. Linux 보충 설치

배포판별 공식 패키지 관리자를 우선한다.

공통

- Git
- curl
- PowerShell이 필요한 경우 pwsh
- Node.js LTS
- Python 3

Obsidian Linux 설치 방식:

- Snap
- AppImage
- Flatpak

공식 안내:

- <https://obsidian.md/help/install>

Codex:

```
curl -fsSL https://chatgpt.com/codex/install.sh | sh
```

Claude Code:

```
curl -fsSL https://claude.ai/install.sh | bash
```

Antigravity CLI:

- <https://antigravity.google/docs/cli-install>

22. 설치 자동화를 더 쉽게 사용하는 방법

이 문서는 사용자가 직접 설치하고 설정하기 위한 전체 기준 가이드다.

설치 작업을 바이브코딩 툴에 맡기려면 다음 파일을 사용한다.

파일 2: 세 CLI 자동 설치

02_AUTO_INSTALL_ALL_CLI_FOR_VIBE_CODING.md

이 파일은 바이브코딩 툴에 다음을 지시한다.

- Windows·macOS·Linux 확인
- WinGet·터미널·PowerShell·Git 확인
- Codex CLI 설치
- Claude Code CLI 설치
- Antigravity CLI 설치
- 기존 설치 중복 방지
- 로그인 안내
- 버전·PATH·프로젝트 접근 검증
- 세 CLI Registry 생성
- 다중 CLI 오케스트레이션 준비 검사

파일 3: Obsidian 자동 설치·연동

03_AUTO_INSTALL_OBSIDIAN_FOR_VIBE_CODING.md

이 파일은 CLI 설치와 검증이 끝난 다음 사용한다.

- Obsidian 설치 여부 확인
- 기존 Vault 또는 새 Vault 선택
- Vault 경로 선택
- AUNOVA 폴더 구조 생성
- 설정 YAML 생성
- 프로젝트 폴더 등록
- NAS·로컬 경로 매핑
- 세 CLI Adapter 연결
- Obsidian Context Pack 테스트

자동 설치 안전 규칙

자동 설치 파일은 다음을 사용자 승인 없이 수행하지 않는다.

- 기존 Vault 삭제
- 기존 프로젝트 이동
- 외부 저장소 덮어쓰기
- Git Push
- Google Drive 업로드
- NAS 파일 삭제
- API 키 저장
- 실행 정책의 전역 완화
- 관리자 권한 명령 실행

자동 설치가 실패하면 이 파일 6의 해당 절차로 돌아와 수동으로 복구한다.

23. 최종 권장 설치 시나리오

개인 Windows PC + NAS + GitHub + Drive

1. WinGet 확인
2. Windows Terminal
3. PowerShell 7
4. Git + GitHub CLI
5. Node.js LTS + Python
6. Codex CLI
7. Claude Code CLI
8. Antigravity CLI
9. 세 CLI 로그인·테스트
10. Obsidian 설치
11. 로컬 SSD에 AUNOVA Vault 생성
12. NAS Z:\HDD1\yongusb\ aunova 등록
13. 프로젝트별 Obsidian Index 생성
14. GitHub 연결
15. Drive 산출물 폴더 연결
16. 다중 CLI 오케스트레이션 테스트

NAS를 쓰지 않는 사용자

프로젝트 루트: D:\Projects 또는 사용자가 선택한 폴더
external_storage.enabled: false
나머지 구조는 동일

GitHub를 쓰지 않는 프로젝트

Git은 로컬 체크포인트로 사용 가능
GitHub Push 단계만 비활성화
Drive·NAS는 독립 사용 가능

모든 외부 채널을 건너뛰는 세션

Obsidian + 로컬 프로젝트 + CLI만 사용
업무마무리 시 Obsidian Handoff와 로컬 Validation만 기록

24. 공식 링크 모음

Windows·터미널

- WinGet: <https://learn.microsoft.com/windows/package-manager/winget/>
- Windows Terminal: <https://learn.microsoft.com/windows/terminal/>
- PowerShell: <https://learn.microsoft.com/powershell/scripting/install/install-powershell-on-windows>

개발 기본 도구

- Git: <https://git-scm.com/install/>
- GitHub CLI: <https://cli.github.com/>
- Node.js: <https://nodejs.org/en/download>
- Python: <https://www.python.org/downloads/>
- VS Code: <https://code.visualstudio.com/download>

CLI

- Codex CLI: <https://developers.openai.com/codex/cli>
- Codex Windows: <https://developers.openai.com/codex/windows/windows-sandbox>
- Claude Code 설치: <https://docs.anthropic.com/en/docs/claude-code/setup>
- Claude Code 빠른 시작: <https://docs.anthropic.com/en/docs/claude-code/quickstart>
- Antigravity CLI: <https://antigravity.google/product/antigravity-cli>
- Antigravity CLI 설치: <https://antigravity.google/docs/cli-install>

Obsidian

- 다운로드: <https://obsidian.md/download>
- 설치: <https://obsidian.md/help/install>
- Help: <https://obsidian.md/help>

25. 문서 업데이트 정책

CLI 설치 명령과 지원 운영체제는 변경될 수 있다.

이 문서는 다음 시점에 공식 링크를 다시 확인해 업데이트한다.

- 자동 설치 파일 제작 전
- 하네스 배포 전
- CLI 주요 버전 변경 후
- 설치 오류가 반복될 때
- Windows 샌드박스 정책 변경 시

반복 오류는 사용자 프롬프트를 길게 만드는 방식이 아니라 다음 중 하나를 수정한다.

설치 검사 규칙
환경 감지 스크립트
Adapter
Skill
Validation
문제 해결 표

부록 A. Windows 한 번에 확인하는 명령

```
$Commands = @(
    "winget",
    "wt",
    "pwsh",
    "git",
    "gh",
    "node",
    "npm",
    "python",
    "codex",
    "claude",
    "agy"
)

foreach ($Command in $Commands) {
    $Found = Get-Command $Command -ErrorAction SilentlyContinue
    if ($Found) {
        Write-Host "[OK] $Command -> $($Found.Source)"
    }
    else {
        Write-Host "[MISSING] $Command"
    }
}
```

부록 B. 경로 검사 예시

```
$Paths = @(
    "D:\Obsidian\AUNOVA_AI_OS",
    "Z:\HDD1\yongusb\ aunova"
)

foreach ($Path in $Paths) {
    if (Test-Path $Path) {
        Write-Host "[OK] $Path"
    }
    else {
        Write-Host "[NOT FOUND] $Path"
    }
}
```

부록 C. 설치 기록 템플릿

AUNOVA Installation Record

시스템

- OS:
- 사용자:
- 설치일:

경로

- Vault:
- 프로젝트 루트:
- NAS:

기본 도구

- WinGet:
- PowerShell:
- Git:
- GitHub CLI:
- Node.js:
- Python:

CLI

- Codex:
- Claude Code:
- Antigravity:

연결

- GitHub:
- Google Drive:
- NAS:

판정

- READY / READY WITH LIMITATIONS / NOT READY

남은 문제

-

참고문헌 및 검증 기준

이 문서는 2026년 7월 21일 기준으로 다음 공식 문서를 중심으로 구성했다.

- Microsoft Learn: WinGet, PowerShell
- Git 공식 설치 문서
- GitHub CLI 공식 문서
- Node.js 공식 다운로드
- Python 공식 다운로드
- OpenAI Codex 공식 문서
- Anthropic Claude Code 공식 문서
- Google Antigravity 공식 문서와 Codelab
- Obsidian 공식 Help

실제 설치 시 링크의 최신 지침을 우선한다.