

바이브 메이커 · 무료 자료

크래프트 애니씽 설치부터 첫 재현까지

주소 하나를 넣으면 내 클로드코드가 그 웹사이트를 새로 짓고, 원본과 폭마다 견줘 비슷해질 때까지 고치는 오픈소스 도구 Craft Anything. 맥 설치부터 첫 재현, 결과 폴더 읽기까지 실제로 돌린 기록 그대로 따라 합니다.

맥 설치 · 받은 파일 확인 · 클로드코드 연결 · 앱으로 첫 재현 · 명령 한 줄 재현 · 점수 읽기 · 이어서 고치기 · 쓰는 범위

목차

한눈에	이 자료로 하는 일	03
1 · 설치	앱을 받고, 받은 파일이 그대로인지 확인합니다	04
2 · 클로드코드 연결	짓는 사람은 내 Claude Code 입니다	05
3 · 첫 재현 ① 앱	앱에서는 고르고, 적고, 시작을 누릅니다	06
4 · 첫 재현 ② 명령	터미널에서는 명령 한 줄입니다	07
5 · 결과 읽기 ①	끝나면 프로젝트 폴더 하나가 남습니다	08
5 · 결과 읽기 ②	빌더는 비슷해질 때까지 스스로 견주고 고칩니다	09
6 · 이어서 고치기	「추측한 것」 부터 읽고, 이어서 고칩니다	10
7 · 쓰는 범위	내 것, 허락받은 것에만 씁니다	11
막히면	증상별로 할 일	12

한눈에

이 자료로 하는 일

Craft Anything(크래프트 애니씽)은 storytold 가 공개한 오픈소스입니다. 웹사이트 · 앱 · 게임이 어떻게 생겼고 어떻게 돌아가는지 살펴본 뒤, AI 가 **내 버전을 새로 짓게** 합니다. 짓는 일은 내 컴퓨터의 코딩 에이전트(이 자료에선 **Claude Code**)가 하고, Craft Anything 은 원본을 찍고 견주는 쪽을 맡습니다. 아직 초기 알파(early alpha)입니다.

1	**분석 · Study**	원본을 화면 폭마다 찍고, 보이는 글자 · 배치 · 색 · 글꼴 · 열리는 메뉴를 읽습니다
2	**만들기 · Build**	코딩 에이전트가 새 프로젝트 폴더에 내 버전을 짓습니다
3	**검사 · Check**	지은 것과 원본을 견주고 아직 안 맞는 곳을 적습니다
4	**완성 · Ready**	결과를 열어 보고, 바꿀 것을 말하면 더 고칩니다

필요한 것
맥
애플 실리콘 · 인텔 공용판. 윈도우 · 리눅스판도 있습니다

필요한 것
Claude Code
내 컴퓨터에 설치하고 로그인해 둔 상태. 짓는 사람입니다

필요한 것
크롬
웹사이트를 찍을 때 크롬 같은 크로미움 브라우저를 씁니다

실제로 돌린 결과

우리 무료 자료실 페이지 · 8~9쪽

100%
1440 · 768 · 390 px 세 폭 닮은 정도 (도구 검사 기준)

8분 33초
주소를 넣고 완성까지

5번
빌더가 스스로 견준 횟수 (사이사이 4번 고침)

⚡ **이 자료는 클린룸(기본값)만 다룹니다**

클린룸은 원본 속(코드 · 파일 · 페이지 소스)을 읽지 않고, 방문자에게 보이는 것만으로 짓습니다. 쓰는 범위는 11쪽에 따로 정리했습니다.

기준: Craft Anything README · 릴리스 v0.6.1 (2026-11-12) · 2026-11-20 맥에서 실제로 돌린 기록 · Craft Anything 과 바이브 메이커는 제휴 관계가 없습니다

앱을 받고, 받은 파일이 그대로인지 확인합니다

1. GitHub 저장소 **storytold/craft-anything** 의 **Releases** 에서 최신판을 엽니다
   이 자료 기준 v0.6.1 (2026-11-12). 판이 올라갔으면 아래 명령의 숫자만 바꿉니다
2. 맥은 `craft-anything-0.6.1-macos-universal.dmg` 를 받습니다
   애플 실리콘 · 인텔 공용. 윈도우는 `.msi`, 리눅스는 `.AppImage`
3. 같은 곳의 `SHA256SUMS.txt` 도 받아 같은 폴더(다운로드)에 둡니다
4. 터미널에서 아래 명령을 돌려 **OK** 가 나오면 받은 파일이 올린 그대로입니다
5. dmg 를 열고 Craft Anything 을 응용 프로그램 폴더로 끌어 넣습니다

복사해서 그대로 쓰세요

```
cd ~/Downloads
shasum -a 256 -c SHA256SUMS.txt --ignore-missing
spctl --assess --type open --context context:primary-signature -v \
  "craft-anything-0.6.1-macos-universal.dmg"
```

우리 맥에서 실제로 나온 줄

```
craft-anything-0.6.1-macos-universal.dmg: OK
craft-anything-0.6.1-macos-universal.dmg: accepted
source=Notarized Developer ID
```

「accepted」와 「source=Notarized Developer ID」가 보이면 애플 서명 · 공증을 통과한 파일입니다. 하나라도 실패하면 그 파일은 열지 않습니다.

⚡ 명령줄 도구는 앱 안에 들어 있습니다

0.2.0 이후 판은 앱 안에 `craft-anything-cli` 가 같이 들어 있어 zip 을 따로 받지 않아도 됩니다. 아래 세 줄이면 터미널 어디서나 `craft-anything-cli` 로 부를 수 있습니다.

복사해서 그대로 쓰세요

```
mkdir -p ~/.local/bin
ln -sf "/Applications/Craft Anything.app/Contents/MacOS/craft-anything-cli" \
  ~/.local/bin/craft-anything-cli
echo 'export PATH="$HOME/.local/bin:$PATH"' >> ~/.zshrc && source ~/.zshrc
craft-anything-cli --version
```

우리 맥에서 실제로 나온 줄

```
craft-anything-cli 0.6.1
```

2 · 클로드코드 연결

짓는 사람은 내 Claude Code 입니다

Craft Anything 은 코드를 스스로 짓지 않고, 내 컴퓨터에 **설치 · 로그인된 코딩 에이전트**를 빌더로 부릅니다. 지금 실제로 되는 조합이 Claude Code 이고, API 키나 다른 에이전트로 짓는 쪽은 README 기준 「일부만 됨」입니다.

1. Claude Code 가 설치 · 로그인돼 있는지 봅니다
   터미널에서 `claude` 를 쳐서 대화창이 열리면 됩니다
2. `doctor` 로 설치 · 에이전트 · 엔진을 한 번에 점검합니다
   마지막 줄이 **healthy** 면 다음 쪽으로 갑니다
3. `ai list` 에 **Claude Code** 가 보이는지 확인합니다
   CLI 로 돌릴 때 `--backend "Claude Code"` 로 이 이름을 직접 지정할 수 있습니다

복사해서 그대로 쓰세요

```
craft-anything-cli doctor --format text
craft-anything-cli ai list --format text
```

우리 맥에서 실제로 나온 줄

```
[ok] engine:chromium: a Chromium-based browser: /Applications/Google Chrome.app/Contents/MacOS/Google Chrome
[ok] mcp:self-test: MCP server answered (protocol 2025-06-18, 30 tools listed).
[ok] ai:Claude Code: Claude Code answered in 0 ms.
healthy
… (ai list)
enabled: true
kind: agent_cli
model: claude-code
name: Claude Code
```

⚡ 「installed but not connected」 줄은 재현과 상관없었습니다

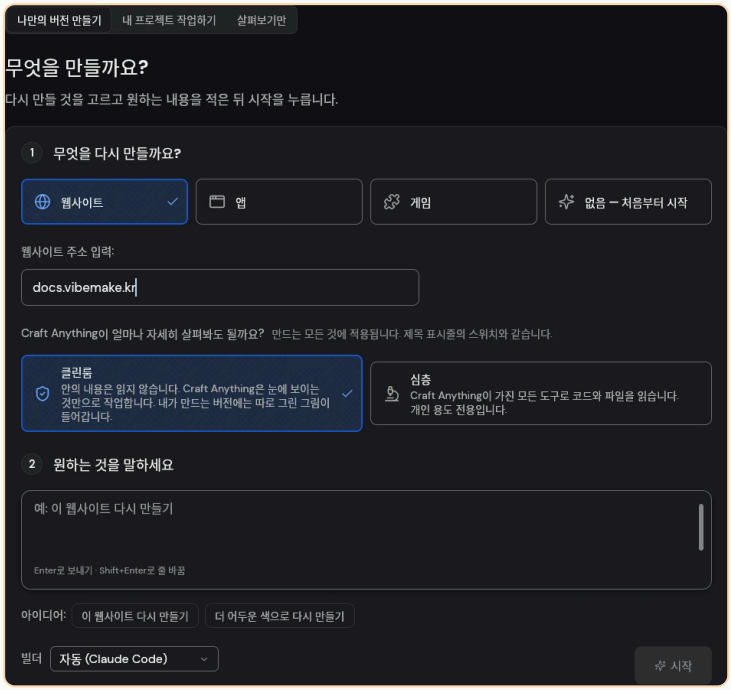
doctor 는 Claude Code · Codex · Cursor 같은 앱마다 `setup --client …` 를 안내합니다. 그 앱 안에서 Craft Anything 도구(MCP)를 쓰게 붙이는 일이라, 재현만 할 때는 하지 않아도 됐습니다. 우리 실행은 이 연결 없이 끝까지 갔습니다.

⚡ 사용량은 내 클로드 플랜에서 나갑니다

실행 기록 첫 줄이 「Using Claude Code (Found Claude Code at …)」 입니다. 앱 오른쪽에 「AI 비용(추정)」 이 뜨는데, 앱 설명 그대로 「API 가격으로 본 코딩 에이전트의 추정치이며 실제 청구액이 아닙니다. 구독 요금제에는 이 금액이 청구되지 않습니다.」 대신 그만큼 내 클로드 사용 한도를 씁니다.

3 · 첫 재현 ① 앱

앱에서는 고르고, 적고, 시작을 누릅니다



「나만의 버전 만들기」 화면 (앱 한국어 화면 · 주소 칸에 우리 자료실)

1. **「나만의 버전 만들기」** 탭 → ① 에서 **웹사이트**
   앱 · 게임 · 처음부터 시작도 같은 자리에 있습니다
2. **웹사이트 주소 입력** 칸에 내 사이트 주소
3. **클린룸** 체크 그대로 둡니다
   심층은 코드와 파일까지 읽는 개인 용도 전용입니다
4. ② **원하는 것을 말하세요** 에 요청을 적습니다
   아이디어 버튼 「이 웹사이트 다시 만들기」 를 눌러도 됩니다
5. **빌더** 가 「자동 (Claude Code)」 인지 보고 **시작**
6. 이 프로젝트를 둘 폴더를 직접 고르려면 아래 **폴더 고르기...**
   기본 = 문서 폴더의 Craft Anything (Settings → General 에서 바꿈)

복사해서 그대로 쓰세요

이 웹사이트 다시 만들기

복사해서 그대로 쓰세요

이 웹사이트를 더 어두운 색으로 다시 만들기

⚡ **시작을 누르면 오른쪽에 4단계가 차례로 켜집니다**

분석 → 만들기 → 검사 → 완성. 빌더가 결정을 물을 때는 질문 카드가 뜨고, 아래쪽 「빌더의 질문」 에서 **나에게 묻기** 와 **알아서 정하기** 중 고를 수 있습니다. 끝난 모습은 8쪽에 있습니다.

4 · 첫 재현 ② 명령

터미널에서는 명령 한 줄입니다

앱 없이도 같은 4단계가 돕니다. 빌더가 묻는 것은 터미널에서 묻습니다. 요청 문장은 README 예시 그대로 쓰고, `example.com` 자리에 내 사이트 주소를 넣습니다.

복사해서 그대로 쓰세요

```
craft-anything-cli rebuild https://example.com "Recreate this website" \
  --clean-room --rounds 2
```

옵션	뜻 (rebuild --help 원문을 옮김)
`--clean-room`	설정과 상관없이 이번 실행은 클린룸. 원본 속은 아무것도 읽지 않습니다
`--rounds 2`	첫 빌드 뒤 다듬는 라운드 수. 기본값이 2 입니다
`--unattended`	질문엔 첫 번째 선택으로 답하고, 승인 요청은 묻지 않고 거절합니다. 자리를 비울 때
`--projects-dir 폴더`	새 프로젝트 폴더가 생길 곳 (기본 = 설정의 프로젝트 폴더)
`--backend "Claude Code"`	빌더를 직접 지정 (`ai list` 의 이름)
`--format jsonl`	출력 형식 (기본 json · jsonl · md · text · csv · tsv 중 고름)

우리가 실제로 돌린 명령

우리 자료실 · 자리 비우고 돌림

참고용 · 우리 사이트에 돌린 명령 그대로 (복사해 쓰지 않습니다)

```
craft-anything-cli rebuild https://docs.vibemake.kr "Recreate this website" \
  --rounds 2 --unattended --clean-room \
  --projects-dir ~/tools/craft-anything/projects --format jsonl > rebuild-1.log 2>&1
```

우리 맥에서 실제로 나온 줄

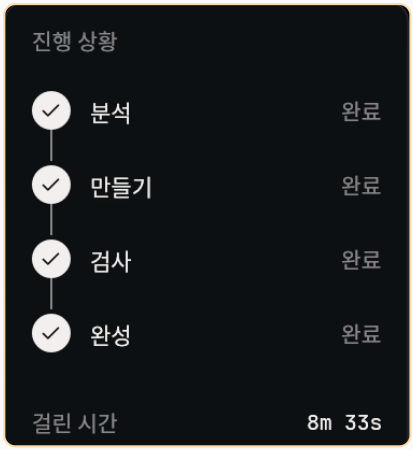
```
Using Claude Code (Found Claude Code at …/bin/claude).
[Study] Opening https://docs.vibemake.kr in a private browser…
[Build] Building the first version…
[Build] The builder checked its work: about 87% like the original, 4 of 4 claims hold.
… (견주고 고치기 4번 더 · 9쪽)
[Ready] Ready: about 100% like the original, 4 of 4 claims hold. 2 questions about the origina
l are still open.
```

명령으로 만든 프로젝트를 앱 왼쪽 목록에서 보려면 Settings → General 에서 그 폴더(`--projects-dir` 로 준 곳)를 목록에 더합니다. 목록에서 빼도 폴더는 지워지지 않습니다 (README).

5 · 결과 읽기

끝나면 프로젝트 폴더 하나가 남습니다

craft.json	요청 · 대상 · 상태 · 기록. **점수**도 여기 있습니다
capture/	원본에서 찍은 것. 우리 실행은 1440 · 768 · 390 px 폭별 전체 사진 3장과 메뉴 · 상태 사진
app/	새로 지은 프로젝트. 우리 실행은 정적 사이트로 지어졌습니다
checks/	원본과 견준 기록. `round-0/` 에 지은 쪽 폭별 사진
investigation.cany/	증거 · 작업 일지 · 남은 질문
inbox/	열린 질문 · 진행 표시



앱 오른쪽 진행 상황 (우리 실행)

점수 읽는 법 craft.json 의 마지막 검사

복사해서 그대로 쓰세요

```
cd "프로젝트 폴더"
python3 - <<'E'
import json; c = json.load(open('craft.json'))['checks'][-1]
print(c['round'], round(c['similarity'], 2), c['note'])
E
```

우리 맥에서 실제로 나온 줄

```
1 99.86 1440 px: 100% alike (layout 100%, detail 100%; page height 8872 vs 8871 px); 768 px: 100% alike (…; page height 8872 vs 8871 px); 390 px: 100% alike (…; page height 10768 vs 10766 px); text 100% alike; structure 100% alike
```

100%	세 폭	1440 · 768 · 390 px 모두 배치(layout) 100% · 세부(detail) 100%. 쪽 길이 차이 1~2 px
100%	글자	보이는 글자(text) 닮은 정도 100% · 구조(structure)도 100% (도구 검사 기준)
99.86	종합	위를 합친 닮은 정도. 첫 라운드에서 끝났습니다 (라운드는 2번까지 허락)
4 / 4	확인	세 폭에서 옆으로 밀리지 않음 · 원본의 영역 구성을 갖춤

5 · 결과 읽기 ②

빌더는 비슷해질 때까지 스스로 견주고 고칩니다

한 라운드 안에서 빌더가 지은 것을 원본과 다섯 번 견줬습니다. 숫자가 잠깐 내려가도(87 → 83) 재고 고치기를 멈추지 않았습니다.

견주기	닮은 정도	다음에 고친 것 (실행 기록 원문 요약)
1번	**약 87%**	원본은 한글을 **단어 단위로만** 줄바꿈한다는 걸 찾아 맞춤 · 숨은 검색 라벨 추가
2번	**약 83%**	제목 · 설명 줄 간격을 잰 값으로 맞춤
3번	**약 84%**	폰 화면은 거의 같아짐 → 맨 위 은은한 주황 빛을 원본과 비교
4번	**약 84%**	제목 줄 간격을 1.375 (설명과 같게)로
5번	**약 100%**	세 폭 모두 100% → 완성



원본 · capture/screenshot-390.png



새로 지은 것 · checks/round-0/screenshot-390.png

같은 실행의 390 px(폰) 사진을 검색칸부터 같은 크기로 잘랐습니다. 직접 비교하려면 두 폴더의 같은 이름 사진을 나란히 열어 봅니다.

6 · 이어서 고치기

「추측한 것」 부터 읽고, 이어서 고칩니다

What's guessed: the screenshots only showed the page standing still, so three behaviours are my best guesses:

- **Search:** typing a number or a word filters the guides. If nothing matches, it shows "검색 결과가 없어요".
- **Scrolling:** the search bar stays pinned at the top.
- **Cards:** they get a light orange outline when you point at them, and the PDFs open in the same tab.

빌더의 마지막 보고 중 「What's guessed」. 원본을 멈춘 사진으로만 봤으니 검색 · 스크롤 · 카드 동작 셋은 추측이라고 밝힙니다.

「남은 질문 2개」 는 `investigation.cany` 에 적혀 있습니다. ① 검색칸에 쳤을 때 무엇을 거르고, 스크롤해도 위에 붙어 있는가 ② 카드에 마우스를 올렸을 때 모양과 PDF 가 새 탭에서 열리는가. 질문마다 **확인하는 법**(검색어를 친 원본 찍기 · 반쯤 스크롤한 원본 찍기)과 **대신 추측한 값**이 같이 남아 있습니다.

앱에서 화면 아래 「변경 요청」 칸

「바꾸기」 를 고르고 바꿀 것을 적어 보냅니다. 「질문만 하기」 는 고치지 않고 묻기만 합니다. 칸 왼쪽 클립 모양 버튼으로 원본 사진 파일 같은 자료를 붙입니다.

아래는 빌더가 보고 끝에 권한 다음 단계(검색어를 친 화면 · 스크롤한 화면을 찍어 견주기)를 옮긴 예시 문장입니다. 우리는 이 요청을 아직 보내지 않았습니다.

복사해서 그대로 쓰세요

남은 질문을 확인해 줘. 원본 검색칸에 12 를 친 화면과 반쯤 스크롤한 화면을 찍어서, 검색과 고정되는 머리줄을 원본에 맞춰 줘.

명령으로 --continue-in · example.com 자리에 내 사이트

복사해서 그대로 쓰세요

```
craft-anything-cli rebuild https://example.com "Make the header blue" \
  --continue-in "프로젝트 폴더"
craft-anything-cli rebuild https://example.com "continue" \
  --continue-in "프로젝트 폴더" --resume
```

첫 줄 = 기존 프로젝트에 후속 요청을 보냅니다(요청이 다음 메시지가 됩니다). 둘째 줄 = 마지막 실행이 멈춘 곳(일시정지 · 앱이나 컴퓨터가 꺼짐)부터 이어 갑니다. 이때 요청 문장은 무시됩니다. (rebuild --help 기준)

7 · 쓰는 범위

내 것, 허락받은 것에만 씁니다

쓰는 곳

내 것 · 허락받은 것

내 사이트 · 내 앱, 살펴봐도 된다고 허락받은 소프트웨어에만 씁니다

하지 않는 것

우회는 없음

로그인 · DRM · 안티치트를 우회하지 않습니다. 도구가 그렇게 만들어져 있습니다

옮기지 않는 것

로고 · 사진 · 글

내 것이거나 허락받았다고 확인해 줄 때만 옮기도록 빌더에게 지시돼 있습니다

방식	무엇을 보나
클린룸 (기본)	원본 속(코드 · 파일 · 페이지 소스)은 읽지 않습니다. 화면 사진 · 보이는 글자 · 잰 배치 · 색 · 글꼴 · 간격 · 열어 본 메뉴처럼 방문자에게 보이는 것만 씁니다. 속을 보는 도구는 거절하고 심층을 켜는 법을 알려 줍니다
심층 (Deep)	모든 도구를 씁니다. 프로그램의 코드 · 파일, 웹사이트의 자바스크립트까지 읽습니다. 앱 화면 표현 그대로 「개인 용도 전용」 입니다

⚡ **클린룸은 컴퓨터를 막아 주는 칸막이가 아닙니다**

클린룸은 Craft Anything 이 무엇을 내주고 무엇을 거절하느냐의 규칙입니다. 내가 돌리는 코딩 에이전트는 여전히 나로서 움직이고, 무엇을 읽고 쓰고 실행하는지는 그 에이전트 자체 권한을 따릅니다 (README).

지금 되는 것 · 아직인 것

README Status · 초기 알파

지금 됨	데스크톱 앱 · CLI · MCP 서버 · Claude Code 를 빌더로 쓰는 웹사이트 · 앱 · 게임 4단계 · 웹사이트 세 폭 검사
일부만	API 키 · 다른 코딩 에이전트로 짓기(만들어졌지만 실제로는 아직 안 돌려 봄) · 앱 · 게임의 동작 검사

기능별 상세와 확인 방법은 저장소의 docs/STATUS.md 에 있습니다. Craft Anything 은 storytold 가 MIT 또는 Apache 2.0(골라 씀)으로 공개한 오픈소스이고, 바이브 메이커와 제휴 관계는 없습니다. 이 자료는 웹사이트 하나(우리 자료실)를 돌린 기록이라 앱 · 게임 재현 결과는 담지 않았습니다.

막히면

증상별로 할 일

증상	할 일
「Apple could not verify craft-anything-cli」	브라우저로 CLI zip 을 받았을 때 뜹니다. 「완료」를 누르고(휴지통으로 옮기지 않음) `xattr -d com.apple.quarantine 파일경로` 또는 시스템 설정 → 개인정보 보호 및 보안 → 「그래도 열기」. 앱 안의 CLI 를 쓰면 이 일이 없습니다 (4쪽)
shasum 결과가 FAILED	그 파일은 열지 않고 다시 받습니다. 계속 실패하면 출력과 릴리스 태그를 적어 저장소에 알립니다
doctor 의 ai 줄이 [ok] 가 아님	Claude Code 를 설치하고 로그인한 뒤 doctor 를 다시 돌립니다 (5쪽)
engine:chromium 이 없음	크롬을 설치합니다. 웹사이트를 찍으려면 브라우저를 띄워야 합니다(리눅스 Flatpak 판은 브라우저를 못 띄워 웹사이트 분석이 안 됩니다)
중간에 멈춤	일시정지 · 앱 종료 · 컴퓨터 꺼짐 모두 `--continue-in "프로젝트 폴더" --resume` 로 이어 갑니다 (10쪽)
묻지도 않고 정해 버림	`--unattended` 는 첫 번째 선택으로 답하고 승인은 거절합니다. 직접 고르려면 빼고 돌립니다. 그러면 터미널에서 묻습니다
프로젝트가 어디 있는지 모름	기본 = 문서 폴더의 Craft Anything (Settings → General 에서 바꿈) · 명령에 `--projects-dir` 를 줬다면 그 폴더
점수가 낮거나 이상함	`checks/` 사진을 `capture/` 의 같은 폭 사진과 나란히 봅니다. 빌더 보고의 「추측한 것」과 남은 질문부터 읽습니다 (9~10쪽)
사진이 흐릿하게 들어감	작은 화면 사진에서 오려 넣은 경우입니다. 우리 실행도 빌더가 「원본 사진 파일을 주면 고쳐진다」고 보고했습니다. 변경 요청에 원본 파일을 붙여 보냅니다

> 더 묻고 싶으면 네이버 카페 「바이브메이커」 자료실 글 댓글로.