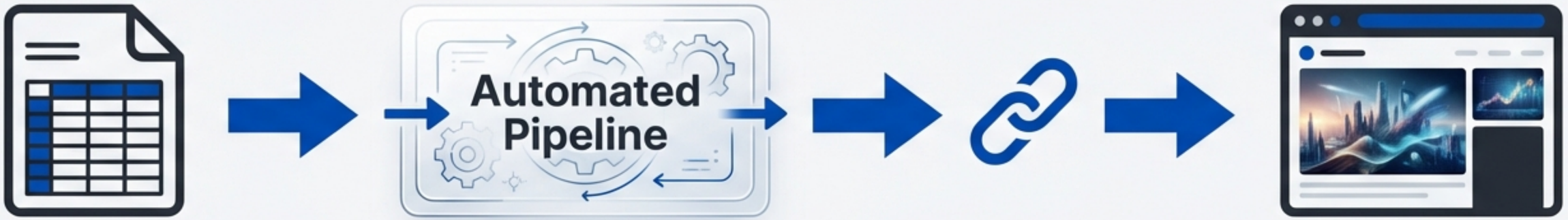
AI가 이미지를 만들어도, 관리는 여전히 사람의 몫이었습니다.

저는 바이브코딩으로 세일즈 랜딩페이지를 자주 만듭니다.
그럴 때마다 매번 반복되는, 지루한 수작업이 있었습니다.
이미지는 AI가 만드는데, 그 관리와 연결은 전부 사람이 하고 있었죠.

"이 모든 과정을 하나의 자동화 파이프라인으로 만들 수 없을까?"

그래서 목표를 명확하게 정의했습니다.
이미지 생성부터 웹 게시까지, 사람의 개입을 최소화하는 시스템을 만들기로 했습니다.



이미지가 생성되는 순간, **자동**적으로 **오브젝트 스토리지**에 저장되고 →
공개 URL이 생성되며 → 스프레드시트에 기록되고 → **웹에서 즉시 사용할 수 있게 만들자.**

여정의 시작: 에이전트 빌더 Lindy와 길잡이가 되어준 AI

이 목표를 달성하기 위해 선택한 툴은 에이전트 빌더 Lindy입니다.
우선 전체적인 구조에 대한 아이디어를 얻기 위해 GPT에게 물었습니다.

Lindy

GPT 왈:

"실제 파일은 오브젝트 스토리지(R2/S3)에 두는 구조가 운영과 확장에 가장 안정적 입니다."

클로드(Claude)의 친절한 부연 설명 덕분에
개념을 확실히 이해할 수 있었습니다.

Claude

5분 만에 완성한 첫 번째 워크플로우 설계

개념을 이해한 뒤, Lindy에게 자연어로 워크플로우 생성을 요청했습니다.
복잡성을 피하기 위해 GPT가 제안한 DB 구조 중 핵심만 추려 단순하게 시작했습니다.

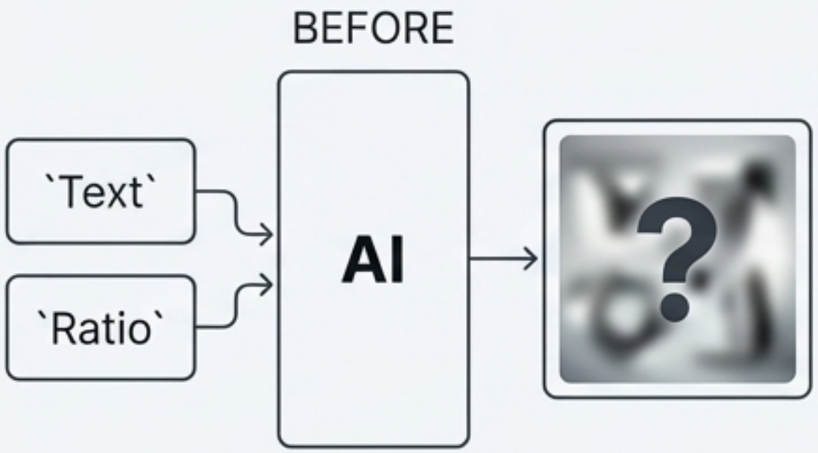
1. **트리거**: 스프레드시트 `status` 컬럼이 "Generate"로 변경
2. **프롬프트 작성**: `landing_section_text`와 `aspect_ratio`로 프롬프트 생성
3. **이미지 생성**: Image Generator 노드 호출, `Temp_Image_URL` 획득
4. **업로드**: `Temp_Image_URL`을 다운로드하여 R2 버킷에 업로드
5. **URL 반환**: R2에서 발급된 `public_url` 반환
6. **DB 업데이트**: 시트의 해당 레코드에 `public_url` 업데이트

Initial DB Schema

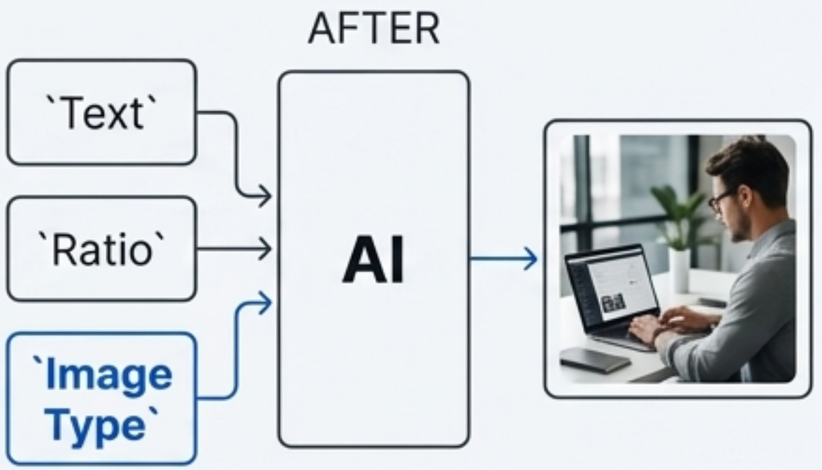
Field	Description
`row_id`	고유 ID
`landing_section_text`	이미지 용도 텍스트
`aspect_ratio`	이미지 비율
`status`	Generate / Done

첫 번째 장벽: 텍스트와 비율만으로는 프롬프트가 너무 모호했습니다.

몇 번의 테스트 후, 중대한 문제를 발견했습니다. 사용자가 입력한 텍스트와 이미지 비율 정보만으로는 **AI**가 맥락에 맞는 고품질 이미지를 생성하기에 정보가 부족했습니다. 결과물의 품질이 들쑥날쑥했습니다.



그래서 DB에 `image_type` 컬럼을 추가했습니다.



문제의 핵심을 관통한 한 줄의 컬럼: ‘image_type’

이미지의 ‘용도’를 명확히 정의하는 `image_type` 컬럼 하나를 추가하자
프롬프트의 품질이 극적으로 향상되었습니다.

A. Hero/Banner Type (가로로 긴 배너)

용도: 웹사이트 최상단(Hero Section), 섹션 중간 띠 배너

B. Spot/Icon Type (오브젝트 중심)

용도: 특징 소개(Feature), 3단 칼럼, 강조 포인트

C. Context/Card Type (상황/장면 중심)

용도: 블로그 썸네일, 후기 카드
(사람이 무언가를 하거나, 기기 화면이 보이는 등 '맥락'이 담기는 이미지)

'타입' 정보 하나가 만들어낸 극적인 변화

BEFORE image_type

입력 landing_section_text: "데이터 자동화 솔루션",
aspect_ratio: "16:9"

결과

AFTER image_type

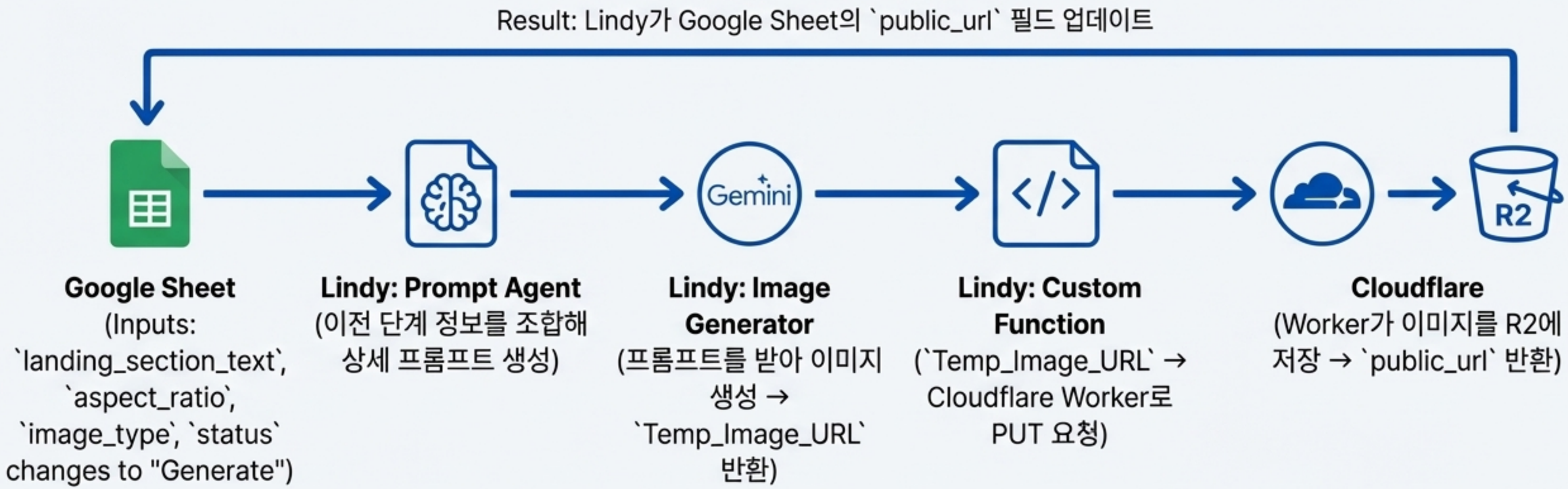
입력 landing_section_text: "데이터 자동화 솔루션",
aspect_ratio: "16:9",
image_type: "Context/Card"

결과

같은 입력값이라도, 이미지 '용도'를 지정해주자 의도에 맞는 결과물을 얻을 수 있었습니다.

최종 완성된 자동화 워크플로우의 전체 구조

`image_type`을 추가하여 완성된 전체 파이프라인입니다. 구글 시트의 `status`를 "Generate"로 바꾸는 것만으로 모든 과정이 자동으로 실행됩니다.



여기가 핵심: Lindy와 R2를 잇는 Cloudflare Worker

이 워크플로우의 가장 중요한 기술적 연결 지점입니다. Lindy가 이미지를 R2에 직접 올리는 대신, Cloudflare Worker 엔드포인트에 요청을 보내는 구조를 선택했습니다.

1. **임시 이미지 URL 다운로드**
   Lindy의 Image Generator가 생성한 임시 URL(`Temp_Image_URL`)을 받습니다.

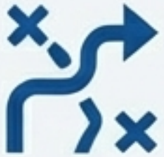
2. **R2에 PUT 요청**
   받은 이미지를 Cloudflare R2 버킷에 저장합니다.

3. **Public URL 반환**
   R2에 저장된 파일의 최종 공개 URL을 Lindy에게 돌려줍니다.

왜 직접 업로드하지 않고 Worker를 사용했을까?

이 구조는 몇 가지 중요한 이점을 가집니다.

의존성 최소화: `boto3`와 같은 특정 SDK나 라이브러리가 필요 없습니다.



실행 환경 제약 회피: Lindy Function의 실행 환경에 제약받지 않고 유연하게 파일을 처리할 수 있습니다.

안정적인 서빙: R2를 Public Access로 열지 않고도 Worker를 통해 CDN처럼 안정적으로 이미지를 서빙할 수 있습니다.

Author's Note: Cloudflare 설정과 코드 디버깅에 시간이 제법 걸렸지만, 전체 시스템이 동작하는 것을 두 눈으로 확인한 순간, 의미 있는 시도였다고 확신했습니다.

마침내 손에 넣은 자동 생성 Public URL

모든 파이프라인이 성공적으로 실행된 후, 구글 시트에는 최종 결과물인 공개 URL이 자동으로 기록됩니다.

C1 | fx

	A	B	C	D	E
1					
2			status	public_url	
3			Done	https://calm-cloud-0676r2-image-proxy.gkhan-f1c.workers.dev/landing_images/20251219_072318.png	
4					

https://calm-cloud-0676r2-image-proxy.gkhan-f1c.workers.dev/landing_images/20251219_072318.png

이제 남은 마지막 한 걸음: 자동화된 에셋을 웹페이지로.

이제 바이브코딩으로 만들어진 웹페이지가 이 스프레드시트의 `public_url`을 실시간으로 읽어서 렌더링하도록 연결하면, 진정한 의미의 이미지 에셋 자동화가 완성됩니다.

부푼 꿈을 안고, 다음에 더 업그레이드된 사례로 돌아오겠습니다.