ChatGPT

Gemini 2.5 Flash Image 및 이미지 생성 MCP 정리

Gemini 2.5 Flash Image 공식 문서 및 API 출처

Gemini 2.5 Flash Image 개요: Google이 새롭게 공개한 최첨단 이미지 생성 및 편집 모델로, 코드명 "nano-banana"로도 불립니다 [1]. 이 모델은 기존 Gemini 2.0 Flash에서 지원하던 **네이티브 이미지 생성** 기능을 한층 강화하여 더 높은 화질의 이미지를 생성하고, 여러 이미지를 합성하거나 스토리텔링에 필요한 캐릭터 일관성 유지, 자연어를 통한 세밀한 이미지 편집 등을 지원합니다 [1]. 특히 **대화형 멀티턴 이미지 편집**까지 가능하여 마치 채팅하듯이 이미지 생성과 수정 작업을 반복할 수 있습니다 [2]. 모든 Gemini 2.5 Flash Image 모델이 생성하거나 편집한 이미지에는 **눈에 보이지 않는 SynthID 워터마크**가 삽입되어 추후 AI 생성 이미지임을 식별할 수 있습니다 [3].

공식 문서 출처: Gemini 2.5 Flash Image에 대한 공식 가이드는 Google Cloud Vertex AI와 Google AI Studio를 통해 제공됩니다. 주요 참고 문서는 다음과 같습니다:

- **Vertex AI 모델 카드:** Google Cloud Vertex AI 공식 문서의 Model Garden 섹션에 "Gemini 2.5 Flash Image Preview" 모델이 소개되어 있습니다. 이 문서에는 모델의 출시 단계(퍼블릭 프리뷰), 입력/출력 유형, 기능 등이 요약되어 있습니다 [2]. 예를 들어, Gemini 2.5 Flash Image Preview는 **신속한 크리에이티브 작업(flow)**을 위해 기존 Flash 모델을 확장한 것으로 소개됩니다 [2].

- **Google 개발자 블로그:** 2025년 8월 26일자 Google Developers Blog 포스트에서도 Gemini 2.5 Flash Image 출시 소식과 특징을 자세히 다루고 있습니다. 이 글에 따르면 해당 모델은 현재 **개발자를 위해 Gemini API 및 Google AI Studio에서**, **엔터프라이즈 사용자를 위해 Vertex AI에서** 프리뷰로 제공됩니다 [4]. 가격 정책도 공개되어 있는데, 100만 출력 토큰당 $30이며 **이미지 1장 생성은 약 1290 출력 토큰**으로 계산됩니다 [4].

- **Gemini API 개발자 문서:** Gemini 2.5 Flash Image는 Google의 새로운 Gemini API를 통해 접근 가능합니다. Google AI for Developers 사이트에 공개된 Gemini API 문서에는 이 모델을 **"고효율 고정밀 이미지 생성 모델"**로 설명하고 있으며 [5], Python/Node.js 등 다양한 환경에서 호출하는 예제가 포함되어 있습니다. 예를 들어, "Native Image Generation (aka Nano Banana)" 섹션에서는 Gemini 2.5 Flash Image로 **맥락적 이미지 생성과 편집을 수행**하는 방법을 안내합니다 [6]. 또한 Google AI Studio의 **Gemini 모델 페이지**에서도 해당 모델을 찾아볼 수 있습니다 [1].

- **Google AI Studio 인터페이스:** AI Studio 웹 UI에서도 Gemini 2.5 Flash Image Preview를 사용해볼 수 있습니다. AI Studio의 Create Prompt 화면에서 모델을 `gemini-2.5-flash-image-preview`로 선택하고, 출력 형태를 "Image and text"로 지정하면 텍스트 프롬프트에 따라 이미지를 생성해 줍니다 [7] [8]. (AI Studio 상의 안내 또한 Vertex AI 문서와 유사한 워터마크 및 사용 제한 사항을 명시합니다.)

以上의 공식 자료들을 통해 **REST API 엔드포인트와 Python SDK 사용법**도 확인할 수 있습니다. REST API의 경우 **Vertex AI 엔드포인트** 또는 **Generative Language API 엔드포인트**를 활용하게 되며, Python SDK로는 최신 Google GenAI SDK (`google-genai` 라이브러리)나 Vertex AI Python SDK를 사용할 수 있습니다. 다음 섹션에서 코드 예시와 함께 자세히 설명합니다.

Gemini 2.5 Flash Image 이미지 생성 코드 예시 (REST API & Python SDK)

Gemini 2.5 Flash Image는 **Vertex AI** 환경과 **Gemini API**를 통해 프로그래밍적으로 이미지 생성을 요청할 수 있습니다. 아래에 **Python SDK** 활용 예시와 **REST API (cURL)** 예시를 함께 제공합니다. 또한 코드와 함께 지원되는 주요 옵션들을 설명합니다.

Python SDK 예시 (Google GenAI SDK 활용)

공식 문서에서는 `google-genai` SDK를 통해 Gemini 모델을 호출하는 방법을 소개합니다 [9] [10]. 먼저 패키지를 설치하고 필요한 환경 변수를 설정한 뒤, `Client` 객체로 모델을 호출합니다:

```
# (전제) SDK 설치 및 환경변수 설정:
# pip install --upgrade google-genai
# export GOOGLE_CLOUD_PROJECT=<YOUR_PROJECT_ID>
# export GOOGLE_CLOUD_LOCATION=global
# export GOOGLE_GENAI_USE_VERTEXAI=True  # Vertex AI로 호출할 경우

from google import genai
from google.genai.types import GenerateContentConfig, Modality
from PIL import Image
from io import BytesIO

client = genai.Client()

response = client.models.generate_content(
    model="gemini-2.5-flash-image-preview",
    contents=("에펠탑 뒤편 밤하늘에 불꽃놀이가 펼쳐지는 이미지를 생성해줘."),
    config=GenerateContentConfig(
        response_modalities=[Modality.TEXT, Modality.IMAGE],
        candidate_count=1,
        safety_settings=[
            {"method": "PROBABILITY"},
            {"category": "HARM_CATEGORY_DANGEROUS_CONTENT"},
            {"threshold": "BLOCK_MEDIUM_AND_ABOVE"},
        ],
    ),
)

# 응답에서 텍스트와 이미지를 추출하여 저장
for part in response.candidates[0].content.parts:
    if part.text:
        print(part.text)   # 생성된 이미지에 대한 설명 텍스트
    elif part.inline_data:
        image = Image.open(BytesIO(part.inline_data.data))
        image.save("output_image.png")
```

위 예시는 **Vertex AI로 요청**하는 모드입니다 (`GOOGLE_GENAI_USE_VERTEXAI=True` 설정). `model` 파라미터에 `gemini-2.5-flash-image-preview`를 지정하여 Gemini 2.5 Flash Image 프리뷰 모델을 사용하고, `contents`에는 **텍스트 프롬프트 문자열**을 넣습니다.

`GenerateContentConfig` 설정에서 중요한 옵션들은 다음과 같습니다:

- `response_modalities` – 출력 유형을 지정합니다. 반드시 `["TEXT", "IMAGE"]` 두 가지를 모두 포함해야 합니다. **이미지만 단독으로 출력하도록 요청하는 것은 지원되지 않으므로** 텍스트 응답을 함께 요구해야 합니다 [11]. 모델은 일반적으로 생성 결과에 대한 **간략한 설명문**(텍스트)과 **이미지 데이터**를 인터리브(interleave)하여 반환합니다.

- `candidate_count` – 생성할 결과 후보 개수입니다. 기본값은 1이며 최대 8까지 요청 가능하여, 한 번의 호출로 다수의 이미지를 얻을 수 있습니다 [12]. 단, 후보를 여러 개 요청하면 그만큼 토큰 비용이 증가합니다 (이미지 한 장당 약 1290 output tokens 소모 [4]).

- `safety_settings` – 콘텐츠 안전성 필터를 위한 옵션입니다. 예시에서는 위험 콘텐츠에 대해 중간 수준 이상을 차단하도록 설정하고 있습니다 (`BLOCK_MEDIUM_AND_ABOVE`) [13]. 이를 통해 폭력적이거나 부적절한 이미지 생성 요청을 자동 거절하거나 블러 처리합니다. 필요에 따라 OpenAI의 content filter와 유사하게 이 임계값을 조정할 수 있습니다.

- **그 외 옵션:** `GenerateContentConfig`에서는 추가로 temperature, topP 등의 생성 다양성 파라미터를 설정할 수 있으나, 이미지 생성에 직접적으로 어떤 영향을 주는지는 공식 문서에 명시되어 있지 않습니다. (텍스트 생성과 마찬가지로 일부 변동성을 주는 역할을 할 수 있습니다.) 기본적으로 temperature=1.0, top_p=0.95 등이 적용됩니다 [14]. 또한 Gemini 모델은 **프롬프트에서 여러 이미지를 입력으로 제공**하여 편집이나 변환을 할 수 있는데, Python SDK에서는 `contents` 리스트에 PIL Image 객체 등을 추가해 이미지+텍스트 복합 입력이 가능합니다 [15] (위 블로그 예시 코드 참고).

REST API (cURL) 예시: REST API를 통해 직접 호출할 때는 Google Cloud OAuth 토큰 또는 Gemini API 키를 사용합니다. 아래는 Vertex AI API를 OAuth 토큰으로 호출하는 cURL 예시입니다 (이때 엔드포인트와 요청 구조는 Vertex AI의 Generative AI API 형식):

```
# Vertex AI API로 이미지+텍스트 생성 요청 (OAuth2 액세스 토큰 사용 예시)
curl -X POST \
 -H "Authorization: Bearer $(gcloud auth print-access-token)" \
 -H "Content-Type: application/json" \
 https://us-central1-aiplatform.googleapis.com/v1/projects/PROJECT_ID/
locations/us-central1/publishers/google/models/gemini-2.5-flash-image-
preview:generateContent \
 -d '{
    "contents": {
     "role": "USER",
     "parts": { "text": "세 단계로 이루어진 PB&J 샌드위치 만드는 튜토리얼을 만들어줘." }
    },
    "generation_config": {
     "response_modalities": ["TEXT", "IMAGE"]
    },
    "safetySettings": {
     "method": "PROBABILITY",
```

```
    "category": "HARM_CATEGORY_DANGEROUS_CONTENT",
    "threshold": "BLOCK_MEDIUM_AND_ABOVE"
   }
  }'
```

> **Note:** 만약 Google의 **Gemini API 키**로 호출할 경우, 엔드포인트는 `https://generativelanguage.googleapis.com/v1beta2/models/gemini-2.5-flash-image-preview:generateContent` 형태이고, 헤더에 `x-goog-api-key: YOUR_GEMINI_API_KEY`를 포함하면 됩니다 [16] [17]. 요청 JSON 구조는 위와 동일하며, `PROJECT_ID/locations/...` 부분이 없고 버전이 약간 다를 수 있습니다.

위 cURL 요청에서도 Python SDK와 마찬가지로 `response_modalities`에 `"IMAGE"`와 `"TEXT"`를 모두 지정해야 함을 유의하세요. **이미지 단독 출력 모드는 지원되지 않으며**, 만약 `"IMAGE"`만 요청하면 에러가 발생하거나 요청이 거부됩니다 [11].

주요 옵션 설명:

- **프롬프트 구성:** 프롬프트 내부에 이미지 설정을 추가로 지정하여 세부 출력 제어가 가능합니다. 예를 들어 "... 각 단계별 이미지는 1:1 비율로 생성해줘."와 같이 지시하면 실제 출력 이미지의 가로세로 비율에 영향을 줄 수 있습니다 [18] [19]. 다만 해상도나 구체적인 픽셀 크기를 직접 지정하는 파라미터는 제공되지 않으며, 모델이 **내부적으로 정해진 해상도** (예를 들어 약 1024x1024 등 추정)를 사용합니다. 필요 시 출력 후 후처리(업스케일 등)를 해야 합니다.

- **여러 이미지/멀티턴 생성:** Gemini 2.5 Flash Image는 한 번의 응답에 여러 이미지와 텍스트를 **섞어서 (interleaved)** 반환할 수 있는 것이 특징입니다 [18] [20]. 예를 들어 한 번의 프롬프트에서 "1단계: ..., (이미지); 2단계: ..., (이미지); ..." 이런 식으로 단계별 설명과 이미지를 교차하여 생성할 수 있습니다. 이를 위해 프롬프트를 구상할 때 원하는 출력 형식을 서술하면, 모델이 알아서 **중간중간 이미지를 삽입한 응답**을 생성합니다. 이 역시 `response_modalities`에 TEXT와 IMAGE를 모두 포함했을 때만 가능한 기능입니다.

- **토큰 및 비용:** 이미지 출력은 텍스트 토큰으로 환산되어 과금됩니다. Google 설명에 따르면 **이미지 한 장은 1290 토큰**으로 간주되며 [4], 이는 비교적 긴 답변 생성과 맞먹는 비용입니다. 따라서 한 번에 과도하게 많은 이미지를 요청하거나 (`candidate_count` 크게 설정 등) 불필요한 이미지를 여러 번 생성하면 비용이 빠르게 증가할 수 있으니 주의해야 합니다.

- **제한 사항 요약:** Gemini 2.5 Flash Image는 강력하지만 아직 **제한 사항**도 존재합니다. **(a)** 모델 출력 이미지에는 사람이 읽을 수 있는 글자를 넣지 않도록 설계되었으며(워터마크 제외), 장문의 텍스트(예: 책 페이지 사진 등)를 렌더링하는 능력은 제한적입니다. Google은 현재 "이미지 내 긴 텍스트 렌더링, 캐릭터 일관성, 사실적 표현 향상" 등을 지속 개선 중이라고 밝히고 있습니다 [21]. **(b)** 생성된 이미지는 DeepMind의 SynthID 기술로 워터마크가 포함되며 [3], 사용자가 이를 임의로 제거할 수 없습니다. **(c)** 안전성 필터가 적용되어 폭력적이거나 성인용 등의 부적절한 내용은 프롬프트에 포함해도 모델이 거부하거나 우회적인 일반 이미지로 대체할 수 있습니다. Vertex AI Studio를 통해 사용할 때는 Google의 이용약관 및 콘텐츠 정책을 따르며, 프리뷰 단계에서는 응답 거부율이 다소 높을 수도 있습니다 (일부 개발자 피드백에 따르면 Vertex AI 경유 시 AI Studio보다 엄격한 필터가 적용된 사례도 있다고 합니다). **(d)** 현재 프라이빗 프리뷰 또는 한정적 공개 단계이므로, 사용을 위해서는 Google Cloud 프로젝트에 Vertex AI API를 활성화하고 권한 요청을 해야 할 수 있습니다 [22].

요약하면, **Gemini 2.5 Flash Image**는 **REST API와 Python SDK를 통해 쉽게 호출**할 수 있으며, 텍스트와 이미지 응답을 조합해 풍부한 출력 생성이 가능합니다. 다만 이미지 생성 요청 시에는 반드시 텍스트 응답을 포함해야 하고, **워터마크나 콘텐츠 필터 등의 제한을 준수**해야 합니다.

유명한 이미지 생성 MCP 서버 (GitHub 사례 조사)

이미지 생성 기능을 LLM 어시스턴트에 통합하기 위해 등장한 **MCP**(Model Context Protocol) 서버들의 오픈소스 구현을 조사했습니다. MCP는 대화형 AI 모델이 외부 툴(예: 이미지 생성기)을 호출할 수 있게 해주는 **표준 프로토콜**로, Claude Desktop이나 Cursor IDE, ChatGPT 플러그인 등에서 지원되고 있습니다. 아래에서는 **GitHub 상에서 인기 있고 안정적으로 동작하는 이미지 생성 MCP 서버**들의 특징과 제공 기능을 정리합니다. 각 MCP 프로젝트별로 **지원 기능 (프롬프트 최적화, 멀티모델 지원, 대기열 처리 등)**과 **안정적인 작동을 위한 주요 설계요소**를 함께 설명합니다.

1. Image Gen MCP (멀티-모델 이미지 생성 서버 by lansespirit)

- **지원 모델:** OpenAI의 DALL-E 시리즈(DALL-E 2, DALL-E 3, GPT-4 Vision의 이미지 생성 `gpt-image-1`)와 Google의 Imagen 시리즈(Imagen 3, **Imagen 4**, Imagen 4 Ultra 등)를 모두 통합 지원하는 **다중 모델** MCP 서버입니다 [23] [24]. 하나의 서버로 OpenAI와 Google 양측의 최신 이미지 생성 API를 호출할 수 있어, **멀티-프로바이더 지원**이 큰 강점입니다.

- **핵심 기능:** 텍스트 입력으로 **고화질 이미지 생성**(text-to-image) 및 **이미지 편집**(image-to-image)을 제공합니다. OpenAI 모델의 경우 입력 이미지와 마스크를 받아 편집하는 기능까지 포함하고 있습니다 [25]. 출력 이미지는 **PNG, JPEG, WebP 등 다양한 포맷**을 선택 가능하고 [26], **화질 옵션**도 "auto, high, medium, low"로 조절할 수 있습니다 [27]. 배경을 투명하게 생성할지 여부도 `transparent/opaque` 등으로 지정 가능합니다 [28].

- **프롬프트 최적화:** 흔히 좋은 이미지를 얻기 위해 프롬프트 엔지니어링이 중요한데, 이 MCP 서버에는 **10종 이상의 프롬프트 템플릿**이 내장되어 있어 자주 쓰는 이미지 생성 요구사항을 손쉽게 구성할 수 있습니다 [29]. 예를 들어 "사진 스타일"이나 "일러스트 스타일" 같은 템플릿을 선택하면, 해당 스타일에 맞춰 프롬프트를 자동 보정해줄 수 있습니다. 또한 **동적으로 사용 가능한 모델과 기능을 런타임에 조회**할 수 있는 API도 제공하여, 클라이언트가 현재 서버에서 어떤 모델들을 쓸 수 있는지 확인할 수 있습니다 [30].

- **MCP 통합 & 통신:** FastMCP Python SDK 기반으로 구현되어 MCP 표준 툴 인터페이스를 따릅니다 [31]. **STDIO, HTTP, SSE** 3가지 통신 방식을 모두 지원하여, 터미널 기반 Claude Desktop 연동부터 웹서버 형태까지 유연하게 활용 가능합니다 [31]. 응답은 Pydantic 스키마로 검증된 **구조화 출력**을 내보내어, LLM 측에서 결과를 파싱하기 쉽게 해둔 것도 특징입니다 [29]. (예: 툴 호출 결과로 `{ "image_data": "...", "image_url": "..." }` 같은 정형 JSON이 반환될 수 있음)

- **저장 및 캐싱:** 생성된 이미지는 **로컬 디스크에 체계적으로 저장**하며, 요청별 메타데이터를 함께 관리합니다 [32]. 또한 **메모리 기반 캐시**를 두어 동일한 요청에 대해 중복 생성을 피하거나 바로 제공할 수 있고, Redis를 이용한 캐시 백엔드도 선택 가능합니다 [33]. 일정 시간 후 이미지 파일을 자동 삭제하는 **보존 기간 정책(autoclean)**도 설정되어 있어 디스크 용량을 안정적으로 관리합니다 [33].

- **운영 및 모니터링:** 프로덕션 환경에서의 안정적 운영을 위해 **Docker 컨테이너** 배포를 지원하고, Nginx 리버스 프록시를 통해 **SSL/TLS 처리 및 속도 제한(레이트 리미팅)**을 설정할 수 있습니다 [34]. 또 Prometheus, Grafana와 연계한 **모니터링 대시보드** 구성이 포함되어 서버 상태와 성능 지표를 추적할 수 있습니다 [34]. 이러한 운영상의 편의 기능 덕분에, 여러 사용자가 동시에 이미지를 생성하더라도 **큐 관리와 부하 분산**이 수월하고 장애를 예방할 수 있습니다. 로그와 에러도 자세히 남기며, Pydantic 검증과 예외 처리로 **에러 핸들링이 체계적**입니다 [35].

  안정적 작동 관련 Tip: 본 서버는 고화질 이미지 생성 시 시간과 비용이 많이 들 수 있기에 **레이트 리밋과 캐싱** 전략으로 안정성을 확보합니다. 예를 들어 Nginx 레이트리밋 설정으로 사용자별 초당 호출 수를 제한하고, 동일 프롬프트 반복 호출 시 캐시된 이미지를 제공함으로써 API 쿼터 낭비를 막습니다. 또한 STDIO 모드(동기 통신)와 HTTP 모드를 분리하여, 챗봇 내부에서 사용할 때는 STDIO로 빠르게 결과를 주고받고, 외부 웹서비스로 확장할 때는 HTTP 모드로 확장하는 식의 **유연한 아키텍처**를 취한 점

도 안정성에 기여합니다. 개발 시에는 테스트용 **가짜 프롬프트**나 **디버그 모드**(hot-reload 등)를 지원하므로, 프로덕션 배포 전에 충분히 검증할 수 있습니다 [35] [36].

2. OpenAI ImageGen MCP (OpenAI 이미지 생성 서버 by spartanz51)

- **지원 모델:** OpenAI의 이미지 생성 API를 MCP로 감싸서 제공하는 경량 서버입니다. **DALL-E 2**, **DALL-E 3** 및 ChatGPT Vision의 내부 모델로 추정되는 **gpt-image-1**까지 지원하며, 사용 가능한 모델은 환경변수나 실행 옵션으로 선택 가능합니다 [37] [38]. (예: `--models dall-e-3` 또는 `--models gpt-image-1` 옵션으로 특정 모델만 활성화 가능 [39].)

- **핵심 기능: 텍스트→이미지 생성**(`text-to-image`)과 **이미지 편집**(`image-to-image`) 두 가지 MCP 툴을 제공합니다 [37]. 이미지 편집은 입력 이미지와 편집용 프롬프트(그리고 필요한 경우 마스크 이미지)를 받아 OpenAI의 편집 API를 호출하며, DALL-E 계열의 고해상도 편집을 활용합니다 [40]. 이 MCP 서버는 **추가 플러그인 없이** 바로 작동하도록 설계되었으며 (Node.js 기반), npm 패키지로도 배포되어 `npx imagegen-mcp`만으로 실행할 수 있을 정도로 간편합니다 [41].

- **파라미터 및 출력:** 프롬프트 외에 **이미지 크기(size)**, **품질(quality)**, **스타일(style)**, **포맷(format)** 등을 환경변수 또는 인자로 설정할 수 있습니다 [42]. 예를 들어 DALL-E API의 기본 출력 크기(1024px)를 변경하거나, 결과 이미지를 JPEG로 변환하여 저장하는 등의 옵션이 제공됩니다. 생성된 이미지는 **임시 파일로 저장**되고, MCP 응답으로는 **파일 경로와 base64 인코딩 데이터**를 함께 전달합니다 [43]. 이렇게 두 형태를 모두 주는 이유는, 일부 클라이언트(예: Cursor 에디터)는 base64 이미지를 바로 보여주고, 다른 일부는 파일 경로를 받아 로드하기 때문입니다 [44] [45].

- **클라이언트 통합:** 이 프로젝트는 VS Code 스타일의 AI 코딩 비서인 **Cursor**와의 연동 예제를 상세히 제공하고 있습니다. Cursor 설정에서 MCP 커스텀 서버로 `npx imagegen-mcp`를 등록하면 에디터 내에서 `@image` 툴로 바로 이미지 생성 결과를 삽입할 수 있습니다 [46] [47]. Anthropic Claude 등과 연동할 때도 유사하게 설정 파일에 경로와 API키를 등록하여 사용 가능합니다.

- **안정성 및 기타:** OpenAI API의 경우 호출 횟수 제한이나 지연이 있을 수 있으므로, 이 MCP 서버는 **단일 스레드 이벤트 루프** 환경에서 요청을 순차 처리합니다. 자체적인 큐잉 시스템은 명시적으로 언급되어 있지 않지만, Node.js 특성상 동시에 여러 요청을 보내기보다는 MCP 툴 호출마다 순서대로 이미지를 생성해 주도록 동작합니다. 또한 **환경변수로 OpenAI API 키를 설정**하도록 하여 키 유출 위험을 낮추고 (`OPENAI_API_KEY` 필요) [48], 지원되는 모델명 외에 잘못된 모델을 지정하면 서버가 실행되지 않도록 **입력 검증**을 수행합니다. 전체적으로 구현이 단순명료하여 유지보수가 쉬운 장점이 있습니다.

  안정적 운영 팁: OpenAI ImageGen MCP를 사용할 때는 **OpenAI API의 요금**과 **레이트한도**를 고려해야 합니다. 여러 사용자가 이미지를 요청하는 경우, **큐**를 별도로 운영하거나 모델별 **동시 호출 제한**을 둘 필요가 있습니다 (DALL-E API는 한 계정당 분당 생성 제한 등이 있음). Spartanz51의 구현은 간소하지만, 필요하다면 이 앞단에 **간단한 대기열 (예: BullMQ)**을 붙이거나, 요청당 임시파일을 정리하는 **크론 작업** 등을 추가하여 운영상의 안정성을 높일 수 있습니다. 현재 코드베이스 자체는 경량이므로, 주로 OpenAI 측의 안정성에 의존하며, 서버 내에서는 **에러 발생 시 재시도 로직** 등을 넣어줄 수 있습니다 (예: OpenAI API 일시적 오류 시 n초 후 재시도). 마지막으로, **환경변수로 모델 접근 제한**을 걸 수 있으므로 (예: 기업용 키에는 DALL-E3 권한이 없을 시 `--models dall-e-2`만 사용), 실제 사용 가능한 모델만 노출하는 것이 안전합니다 [39].

3. Gemini Image MCP (Google Gemini 이미지 생성 서버 by qhdrl12)

- **지원 모델:** Google의 **Gemini 2.x Flash 모델**을 통해 이미지 생성 및 편집 기능을 제공하는 MCP 서버입니다. (초기 버전은 Gemini 2.0 Flash를 사용하였으며, 추후 2.5 Flash Image 지원도 업데이트될 것으로 보입니다.)

Google Cloud의 Vertex AI API를 호출하므로 **서비스 계정 키(JSON)**가 필요하며, 환경변수로 경로를 설정해 사용합니다.

- **핵심 기능: 텍스트→이미지 생성**(`generate_image_from_text`)과 **이미지 변환**(`transform_image_from_file` 또는 `...from_encoded`) 툴을 제공합니다 [49] [50]. `generate_image_from_text`는 순수 텍스트 프롬프트로 새 이미지를 생성하며, 결과로 **원본 이미지 바이트와 저장된 파일 경로의 튜플**을 반환합니다 [51] [52]. `transform_image` 계열 툴은 입력 이미지 + 프롬프트로 **기존 이미지를 변환/편집**합니다. 예를 들어 "이 사진을 스케치 풍으로 바꿔줘" 같은 요청을 구현할 수 있습니다. 이때 이미지는 **파일 경로 입력** 방식과 **base64 문자열 입력** 두 가지를 모두 지원하여, 클라이언트 상황에 맞게 쓸 수 있습니다 [53] [54].

- **프롬프트 최적화:** 이 MCP 서버의 특이점은 **프롬프트 전처리**를 해준다는 것입니다. 사용자가 한국어나 기타 언어로 프롬프트를 넣어도 **자동으로 영어로 번역**하여 Gemini 모델에 보냅니다 [54]. (Gemini가 다국어를 지원하지만 영어로 입력 시 가장 안정적인 결과를 내놓을 가능성이 높기 때문입니다.) 또한 프롬프트 상에 **이미지에 글자가 나타나지 않도록** 유도하는 문구를 자동 추가해 줍니다 [55]. 예컨대, 많은 이미지 생성 모델들이 의도치 않게 이상한 글자나 숫자를 이미지에 삽입하는 문제가 있는데, 이 MCP는 내부적으로 "text in image: none" 같은 제약을 줘서 **엄격하게 텍스트가 출력되지 않도록** 합니다 [53] [55].

- **파일 관리:** 생성된 이미지는 지정된 로컬 폴더에 저장하며, 프롬프트 내용 등을 기반으로 **지능형 파일 이름**을 만들어 붙여줍니다 [56] [57]. 예를 들어 "날아다니는 돼지" 이미지를 생성하면 `flying_pig_<해시>.png` 식으로 파일명을 지어주는 식입니다. 이렇게 하면 파일을 봤을 때 어떤 프롬프트였는지 추측이 가능하고 중복도 방지됩니다. 반환 결과에 파일 경로와 이미지 바이트 데이터를 모두 포함시키므로, **AI 모델은 경로를 기억했다가 필요하면 후속 요청에 사용**할 수도 있습니다 (예: "이전에 만든 flying_pig 이미지를 리터치해줘" 같은 식으로 경로 전달) [52].

- **안정성 및 이슈:** 개발자가 밝혀놓은 알려진 이슈로, **Claude Desktop Host와 연동 시** 약간의 성능 및 경로 해석 문제가 있었다고 합니다 [58] [59]. 대용량 base64 이미지를 STDIO로 주고받을 때 지연이 발생할 수 있고, Claude 측이 반환된 파일 경로를 제대로 불러오지 못하는 경우가 있다고 합니다 [58] [60]. 이러한 문제를 해결하기 위해 **권장되는 방법은** 가능하면 base64 대신 **파일 경로로 이미지를 주고받는 것**입니다. 즉, `transform_image_from_file` 방식을 쓰면 성능상 유리하고 경로 문제도 최소화됩니다 [59] [60]. 또한 여러 클라이언트 중 Claude Desktop에 특화된 문제가 있을 뿐, **다른 MCP 지원 클라이언트에서는 정상 동작**한다고 하니, 환경에 따라 대안을 고려해야 합니다.

  안정적 작동 팁: Gemini Image MCP의 경우 Google Cloud API를 쓰므로 **요청 지연**이나 **API 한도**를 염두에 둬야 합니다. 대용량 이미지 생성은 수 초 이상 걸릴 수 있으므로, MCP 호출 타임아웃을 충분히 길게 잡거나 **비동기 처리**로 짜는 것이 좋습니다. 또한 에러 상황(예: API quota 초과나 이미지 생성 실패 등)에서 **명확한 오류 메시지를 반환**하도록 구현되어 있으므로, 이를 LLM이 인지해 재시도 또는 사용자에게 안내하도록 프롬프트를 설계하면 좋습니다. 마지막으로, 프롬프트 자동 번역 기능은 편리하지만 간혹 뉘앙스 손실이 있을 수 있으므로, 중요한 경우 사용자가 직접 영어 표현을 조정할 수 있게 옵션을 줄 수도 있습니다. (현재 구현상 자동번역은 기본 활성화되어 있음 [54].)

4. AI Image Gen MCP (Replicate 기반 이미지 생성 서버 by mikeyny/GongRzhe)

- **지원 모델:** 이 MCP 서버는 **Replicate** 플랫폼의 이미지 생성 API를 래핑한 것입니다. 기본 설정으로는 Replicate에 공개된 **Black Forest Labs의 "Flux Schnell"** 모델을 사용하도록 되어 있으며, 이는 Stable Diffusion 계열의 최신 고속 생성 모델입니다 [61]. 환경 변수 `MODEL`을 변경하면 Replicate 허브의 다른 모델도 쓸 수 있게 해두었으며 (예: `"stability-ai/sdxl"` 같은 경로를 넣어 다른 모델 사용) [62] [63], 기본값이 `"black-forest-labs/flux-schnell"`로 지정되어 있습니다 [63].

- **설치 및 통합:** Node.js 환경에서 동작하며 npm 패키지로 배포됩니다 (`@gongrzhe/image-gen-server`). Claude Desktop과의 연계를 예로 들어, **Smithery** 패키지 매니저로 손쉽게 설치 및 설정할 수 있도록 가이드하고 있습니다 [64] [65]. Claude 설정파일의 `mcpServers` 항목에 이 패키지를 `npx`로 실행하는 커맨드를 넣고, Replicate API 토큰을 환경변수로 지정하면 바로 Claude의 도구로 등록됩니다 [66]. 로컬 설치 후 `node index.js`로 직접 돌릴 수도 있습니다.

- **기능: generate_image**라는 MCP 툴을 제공하며, 텍스트 프롬프트를 받아 **Flux 이미지 생성 모델**에 요청하고 결과 이미지를 반환합니다 [67]. Replicate API 특성상 비동기로 호출되며, 완료된 이미지 URL을 받아와서 MCP 응답으로 이미지 데이터를 전송합니다. 이때 **자동 프롬프트 승인(autoApprove)** 목록 등을 설정할 수 있어, 특정 툴 호출은 사용자 확인 없이 바로 실행되게 할 수도 있습니다 [68]. (Claude Desktop에서는 외부 MCP 툴 호출 시 보안상 사용자 승인을 요구하는데, `autoApprove`에 툴 이름을 넣으면 이 확인을 스킵함)

- **안정성:** 이 MCP는 Replicate 서비스에 의존하기 때문에, **안정성의 대부분은 Replicate API의 가용성**에 달려 있습니다. 자체적으로는 비교적 단순한 래퍼이므로, 큰 문제는 적습니다. 다만 **네트워크 오류나 API 응답 지연**에 대비한 타임아웃 설정, 오류 재시도 로직 등이 고려될 수 있습니다. 현재 구현에서는 환경변수로 토큰과 모델만 설정하면 되고 별도 큐 처리 로직은 없습니다 [69] [63]. 여러 사용자가 동시에 요청을 보내면 Node.js 이벤트 루프에서 비동기로 처리하면서 Replicate 쪽 큐에 쌓이게 되므로, **자체 서버 내에서 요청을 직렬화하거나 하는 기능은 없는** 점에 유의해야 합니다. 필요하면 이 MCP 앞단에 간단한 **rate limiter**를 둘 수 있습니다.

  안정적 작동 팁: Replicate 기반 MCP를 운용할 때는 **API 호출 비용**(Replicate는 요청 횟수나 사용량 기반 과금)과 **모델 가용성**을 모니터링해야 합니다. Flux Schnell 모델이 비교적 빠르지만, 고해상도 이미지를 요청하면 속도가 느려질 수 있으므로 **프롬프트에 불필요하게 고화질을 요구하지 않도록** 안내하는 것이 좋습니다. 또한 Replicate 토큰은 곧 비용과 직결되므로, MCP 서버에 **간단한 인증**을 걸어 임의의 사람이 함부로 사용하지 못하게 할 필요도 있습니다. 예를 들어 MCP 서버를 로컬에서만 돌리고 외부 포트는 막아두거나, HTTP 모드로 쓸 경우 자체 API키 검증 로직을 추가하는 식입니다. 마지막으로, `MODEL` 파라미터를 통해 대체 모델을 사용할 경우, 그 모델의 입력 요구사항(예: 특정 프롬프트 형식이나 해상도 제한)을 사전에 숙지하고 MCP 코드에 반영해야 일관된 운영이 가능합니다 [61].

以上이 **나노 바나나 (Gemini 2.5 Flash Image) MCP 제작**에 참고할 수 있는 자료 정리입니다. 공식 문서를 통해 모델의 사용법과 제한사항을 숙지하고, 오픈소스 MCP 구현들의 기능과 안정화 기법을 참고하여, 원하는 MCP를 설계 및 개발할 수 있을 것입니다. 특히 멀티 모델을 지원하려는 경우 lansespirit의 Image Gen MCP 구조가 큰 도움이 될 것이고, 단일 모델에 집중한다면 각 전용 MCP의 간결한 구현들이 좋은 레퍼런스가 됩니다. 필요한 경우 GitHub 링크 및 문서를 참조하여 세부 구현을 살펴보시기 바랍니다 [24] [37] [70] [63]. 각 출처의 라이선스와 이용약관을 준수하는 것도 잊지 마세요.

참고 자료: Google Cloud Vertex AI 공식 문서 [2] [12], Google Developers 블로그 [1] [4], Google AI Studio/Gemini API 문서 [6], 및 GitHub MCP 오픈소스 프로젝트 (lansespirit [24], spartanz51 [37], qhdrl12 [70], mikeyny [63] 등).

---

[1] [3] [4] [15] [21] Introducing Gemini 2.5 Flash Image, our state-of-the-art image model - Google Developers Blog
https://developers.googleblog.com/en/introducing-gemini-2-5-flash-image/

[2] Google models | Generative AI on Vertex AI | Google Cloud
https://cloud.google.com/vertex-ai/generative-ai/docs/models

[5] [6] [16] [17] Gemini API | Google AI for Developers
https://ai.google.dev/gemini-api/docs

7 8 9 10 11 12 13 18 19 20 Generate images with Gemini | Generative AI on Vertex AI | Google Cloud
https://cloud.google.com/vertex-ai/generative-ai/docs/multimodal/image-generation

14 22 Gemini 2.5 Flash | Generative AI on Vertex AI | Google Cloud
https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-flash

23 24 25 26 27 28 29 30 31 32 33 34 35 36 GitHub - lansespirit/image-gen-mcp: An MCP server that integrates with gpt-image-1 & Gemini imagen4 model for text-to-image generation services
https://github.com/lansespirit/image-gen-mcp

37 38 39 40 41 42 43 44 45 46 47 48 GitHub - spartanz51/imagegen-mcp: MCP server for OpenAI Image Generation & Editing — text-to-image, image-to-image (with mask), no extra plugins.
https://github.com/spartanz51/imagegen-mcp

49 50 51 52 53 54 55 56 57 58 59 60 70 GitHub - qhdrl12/mcp-server-gemini-image-generator: MCP server for AI image generation and editing using Google's Gemini Flash models. Create images from text prompts with intelligent filename generation and strict text exclusion. Supports text-to-image generation with future expansion to image editing capabilities.
https://github.com/qhdrl12/mcp-server-gemini-image-generator

61 62 63 64 65 66 67 68 69 GitHub - GongRzhe/Image-Generation-MCP-Server: This MCP server provides image generation capabilities using the Replicate Flux model.
https://github.com/GongRzhe/Image-Generation-MCP-Server