ChatGPT

QualiJournal 관리자 시스템 Cloud Run 배포 및 도메인 연결 가이드

QualiJournal 관리자 웹 애플리케이션을 Google Cloud Run에 배포하고 사용자 도메인 **standardai.co.kr**에 연결하는 절차를 설명합니다. 이 가이드는 개발자 및 운영자를 위한 종합 지침으로, 배포 명령부터 도메인 설정, 기능 점검, 백업 및 자동화, 보안 확인, 그리고 배포 실패 시 대처 방안을 순서대로 다룹니다. 각 섹션에는 단계별 명령 예시와 예상 결과, 주의사항이 포함되어 있습니다.

1. Cloud Run 소스 코드 배포 및 정적 HTML 리소스 포함 설정

QualiJournal의 백엔드(FastAPI, 예: `server_quali.py`)를 Cloud Run에 소스 기반으로 배포하려면, GCP Cloud SDK의 **gcloud** 명령어를 사용합니다. 이 명령은 Cloud Build를 통해 소스 코드를 빌드하고 Cloud Run에 컨테이너를 배포합니다. 정적 웹 자산(HTML/JS/CSS 등)을 포함시키기 위해 **.gcloudignore**와 **.dockerignore** 설정에 유의해야 합니다.

- **배포 명령 예시:** 프로젝트 소스 루트 디렉토리에서 다음 명령을 실행합니다. 서비스 이름, GCP 프로젝트, 리전을 실제 값으로 바꾸세요. (리전은 서울의 경우 `asia-northeast3` 등 사용 가능):

```
gcloud run deploy qualijournal-admin \
 --source . \
 --project <PROJECT_ID> \
 --region <REGION> \
 --platform managed \
 --allow-unauthenticated
```

위 명령을 실행하면 Cloud Run이 현재 디렉토리의 소스를 빌드하여 컨테이너 이미지를 생성하고, 지정한 서비스(`qualijournal-admin`)로 배포합니다 [1]. 배포 중 필요한 API가 활성화되지 않았다면 `Y`를 눌러 Cloud Build, Cloud Run API 등을 활성화합니다 [2]. 빌드 및 배포가 완료되면 터미널에 "Deploying... done"과 함께 서비스 URL(예: `https://qualijournal-admin-xxxxxx.run.app`)이 출력됩니다. 이 URL은 Cloud Run이 제공하는 기본 도메인으로, 이후 사용자 도메인과 연결할 수 있습니다.

- **정적 HTML 포함을 위한 설정:** 소스 배포 시 **.gcloudignore** 파일의 규칙을 확인해야 합니다. Cloud Run은 배포 시 `.gcloudignore` 파일이 없으면 기본적으로 `.gitignore` 규칙을 따라 소스 코드를 업로드하는데, 이 경우 정적 자산이 `.gitignore`에 의해 제외될 수 있습니다 [3]. 따라서 정적 HTML/JS 파일이 `.gitignore`에 의해 무시되고 있다면, **.gcloudignore** 파일을 프로젝트 루트에 생성하여 해당 파일들을 포함시켜야 합니다. .gcloudignore 파일을 만들면 .gitignore의 기본 제외 규칙이 적용되지 않으므로, 필요한 자산을 배포에 포함할 수 있습니다 [4]. 예를 들어, `build/` 디렉터리에 새로운 관리자 UI HTML이 있다면 .gcloudignore에 `!build/**`와 같이 예외 규칙을 추가합니다.

Dockerfile을 사용하는 경우 **.dockerignore**도 확인해야 합니다. Cloud Build는 소스를 업로드할 때 .gitignore를 참고하지만, 도커 이미지 빌드 단계에서는 .dockerignore에 정의된 파일들을 제외합니다 [5]. 따라서 정적 파일 폴더가 .dockerignore에 의해 제외되지 않도록 설정해야 합니다. 요약하면, **정적 HTML 및 자산 파일이 빌드/배포에서 누**

락되지 않도록 .gcloudignore와 .dockerignore에 포함 규칙을 정확히 설정해야 합니다. (예: `.gcloudignore`에 `node_modules/` 등 불필요 파일은 제외하되 HTML/정적 파일 경로는 제외 리스트에 넣지 않음).

- **주의사항:** 배포 명령 실행 전 **requirements.txt** 또는 **poetry.lock** 등 종속 패키지 목록이 최신인지 확인하고, FastAPI 앱이 환경변수 PORT를 사용해 해당 포트로 실행되도록 설정되었는지 점검합니다. Cloud Run은 자동으로 `$PORT` 환경 변수를 컨테이너에 주입하며, 일반적으로 uvicorn 등으로 FastAPI를 실행할 때 이 PORT를 사용해야 합니다. 또한, **첫 배포 시 Cloud Run 및 Cloud Build API**가 프로젝트에서 활성화되어 있어야 합니다. 명령 실행 중 요청받는 경우 승인하여 활성화합니다. 배포 완료 후 출력된 URL은 우선 Cloud Run의 기본 도메인이므로, 다음 절차에서 사용자 도메인 연결을 설정합니다.

2. 환경 변수 구성 (.env 파일 및 Cloud Run 등록)

QualiJournal 관리자 시스템의 민감한 설정 값들은 **환경 변수**로 관리합니다. 예를 들어 관리자 인증 토큰 `ADMIN_TOKEN`, 외부 API 키 `API_KEY`, 기타 설정 등을 코드에 하드코딩하지 않고 배포 시 환경 변수로 주입해야 합니다.

- **.env 예시:** 로컬 개발 또는 배포 준비를 위해 프로젝트 루트에 `.env` 파일을 작성합니다 (실제 배포에는 이 파일을 직접 사용하지 않고, Cloud Run 설정에 값만 반영합니다). 예를 들어 `.env` 파일 내용은 다음과 같을 수 있습니다:

```
ADMIN_TOKEN=초기관리자토큰값123!
API_KEY=abcd1234efgh5678ijkl
DB_URL=postgresql://user:pass@host:5432/dbname
```

주의: `.env` 파일 자체는 버전관리(git 등)에 커밋하지 말고, 배포용 참고 파일로 유지합니다. 위 예시는 ADMIN_TOKEN, API_KEY, (필요시) DB 연결 문자열 등을 포함한 형태입니다. 실제 값은 운영 환경에 맞게 설정하세요.

- **Cloud Run에 환경 변수 등록:** Cloud Run에 배포할 때 위 환경변수들을 설정해야 애플리케이션이 정상 동작합니다. 환경 변수 설정은 **gcloud CLI 플래그** 또는 **콘솔 UI** 둘 다 가능합니다.

- CLI 이용: `gcloud run deploy` 명령에 `--set-env-vars` 플래그를 사용하면 배포와 동시에 환경 변수를 지정할 수 있습니다. 예를 들면:

```
gcloud run deploy qualijournal-admin \
 --image gcr.io/<PROJECT_ID>/qualijournal:latest \
 --region <REGION> \
 --platform managed \
 --set-env-vars "ADMIN_TOKEN=초기관리자토큰값123!,API_KEY=abcd1234efgh5678ijkl"
```

(소스 배포인 경우 `--image` 대신 `--source .`를 사용하며, 동일하게 `--set-env-vars` 사용 가능). 쉼표로 구분하여 여러 변수를 한 번에 설정할 수 있습니다 [6]. 값에 특수문자나 쉼표가 있는 경우 적절히 이스케이프하거나 `--set-env-vars`를 반복 사용할 수 있습니다 [7]. **예상 결과:** 새로운 리비전이 배포되며 설정된 환경 변수들은 해당 리비전에 적용됩니다. 배포 후 Cloud Run 콘솔의 서비스 상세 화면 > **Variables & Secrets** 탭에서 설정된 환경 변수를 확인할 수 있습니다.

- 콘솔 UI 이용: GCP 콘솔에서 Cloud Run 서비스를 배포하거나 수정할 때 **콘테이너 > 변수 & 보안** 섹션에서 키-값 쌍을 추가할 수도 있습니다 [8]. 이미 배포된 서비스의 환경 변수를 수정하려면 **Edit & Deploy New Revision**을 선택하고 해당 탭에서 값을 편집한 후 새 리비전을 배포합니다.

- **주의사항:** 환경 변수에 민감한 값(특히 ADMIN_TOKEN, API_KEY 등)을 넣을 때에는 **시크릿 매니저** 등도 고려해야 하나, 간단히 환경 변수로 관리하는 경우 반드시 프로젝트 내 접근 권한을 최소화하고, 값 노출에 유의해야 합니다. Cloud Run 환경 변수는 해당 서비스 내부에서만 조회 가능하며, 다른 서비스나 사용자에게 직접 노출되지 않습니다. 다만, **배포 로그나 어플리케이션 로그에 환경 변수 값을 출력하지 않도록** 주의합니다. (.env 파일은 배포에 직접 사용되지 않지만, `gcloud run deploy` 시 `--env-vars-file` 옵션으로 .env 파일을 지정하여 일괄 등록할 수도 있습니다 [9].)

3. 사용자 도메인 연결 설정 (가비아 DNS, A레코드/CNAME 및 HTTPS 인증)

기본 도메인으로 서비스가 배포된 후, **standardai.co.kr** 같이 사용자의 커스텀 도메인으로 Cloud Run 서비스를 연결할 수 있습니다. 이 가이드에서는 도메인 등록 업체로 **가비아(Gabia)**를 사용하는 경우를 예로 들어 DNS 레코드 설정과 HTTPS 인증 과정을 설명합니다.

- **Cloud Run 도메인 매핑:** 먼저 Cloud Run 서비스에 사용자 도메인을 매핑해야 합니다. GCP 콘솔의 Cloud Run 서비스 상세 화면에서 **커스텀 도메인** 또는 **도메인 매핑** 기능을 이용하거나, gcloud 명령어를 사용합니다. 예시 (gcloud beta 명령어):

  ```
  gcloud beta run domain-mappings create --service qualijournal-admin --domain standardai.co.kr --region <REGION>
  ```

  이 명령은 Cloud Run에 현재 서비스와 도메인 매핑을 생성하는 것으로, 실행 후 도메인 소유권 확인 및 DNS 레코드 정보가 생성됩니다. (콘솔 UI에서는 **Domain Mappings** 메뉴에서 새 매핑 추가를 진행하며, 처음 사용하는 도메인의 경우 도메인 소유 확인 절차가 안내됩니다. 일반적으로 도메인 소유자 확인을 위해 GCP에서 제공하는 TXT 레코드를 DNS에 등록해야 할 수 있습니다.)

- **DNS 레코드 설정 (가비아):** 도메인 매핑이 생성되면, Cloud Run 측에서 이 도메인을 가리키는 **DNS 레코드 정보**를 제공합니다 [10] [11]. 예를 들어, **A 레코드**와 **AAAA 레코드** (IPv6용) 또는 **CNAME 레코드** 형태로 몇 개의 레코드가 제시됩니다. 이러한 레코드를 가비아 DNS 설정에 추가해야 합니다. Gabia에 로그인하여 **My가비아 > 도메인 관리**로 이동한 뒤, 해당 도메인의 **DNS 관리** 또는 **레코드 편집** 메뉴를 엽니다.

Cloud Run이 제공한 레코드에 따라 다음을 설정합니다 [12]: - **A 레코드**: 이름은 `@` (루트 도메인 표기)로 하고, 값으로 제공된 IPv4 주소를 입력합니다. Cloud Run은 글로벌 로드밸런서를 통해 여러 개의 IP를 제공할 수 있으므로 안내된 A레코드들을 모두 추가합니다. - **AAAA 레코드**: (선택) IPv6 접근을 위한 레코드도 제공된 경우 동일하게 추가합니다. - **CNAME 레코드**: 만약 서브도메인 (예: `www.standardai.co.kr`)을 사용하거나 Cloud Run에서 CNAME을 제시한 경우, 이름 필드에 해당 서브도메인을, 값에는 Cloud Run이 안내한 도메인 (예: `ghs.googlehosted.com` 또는 Cloud Run 서비스 URL) 등을 설정합니다. 단, Apex 도메인(`standardai.co.kr`)은 CNAME으로 설정할 수 없으므로 A/AAAA 방식으로 해야 합니다.

설정을 저장한 후 **DNS 변경 사항 전파**를 기다립니다. 보통 몇 분 내에 적용되지만, 레지스트리 TTL에 따라 수시간까지 걸릴 수 있습니다 [13]. 적용 여부는 `nslookup` 또는 `dig` 툴을 사용하여 확인할 수 있습니다 (예: `dig standardai.co.kr A` 질의가 Cloud Run에서 제시한 IP를 반환하면 성공).

- **HTTPS 인증 및 검증:** Cloud Run의 도메인 매핑은 자동으로 **Let's Encrypt SSL 인증서** 발급을 시도합니다. DNS 설정이 올바르다면 수분 내에 인증서가 발급되어 HTTPS로 서비스가 제공됩니다 [14]. 웹 브라우저에서

`https://standardai.co.kr`로 접속하여 인증서가 유효한지, 보안 연결(자물쇠 아이콘)이 표시되는지 확인하세요. 만약 Cloud Run 콘솔에서 도메인 상태가 **Pending** 상태라면 DNS 설정이 제대로 되었는지 재확인해야 합니다. 인증서 발급 전에는 브라우저가 연결을 거부하거나 비보안 경고를 표시할 수 있으므로, **DNS 적용 후 충분히 시간을 두고 재시도**합니다.

- **주의사항:** Cloud Run 도메인 매핑 기능은 현재 미리보기(Preview) 단계일 수 있으므로 간혹 지연이나 이슈가 있을 수 있습니다. 가비아 DNS를 사용할 때 **클라우드플레어(Cloudflare)** 등의 별도 CDN을 사용 중이라면, Cloud Run의 인증 요청이 방해받지 않도록 Cloudflare의 "Always use HTTPS"와 같은 설정을 일시적으로 끄라는 권고가 있습니다 [15]. 대부분의 경우 가비아 DNS에 직접 설정하면 문제없이 적용됩니다. 또한, **www 서브도메인**도 함께 운영하려면 `www.standardai.co.kr`에 대한 별도 매핑과 DNS 설정(CNAME 또는 A)을 추가로 해야 합니다. 도메인 연결 후에는 항상 HTTPS로 접속되도록(HTTP 요청시 HTTPS로 리디렉션) Cloud Run 서비스 설정의 **HTTPS만 사용** 옵션을 확인하는 것도 좋습니다.

4. 배포 후 서비스 동작 확인 및 신규 UI(신판 HTML) 적용 검증

도메인 연결까지 마쳤다면, **배포가 성공적으로 이루어졌는지와 최신 관리자 UI 변경사항(신판 HTML)이 제대로 적용되었는지** 확인해야 합니다. 이 단계에서는 웹 애플리케이션이 실제로 정상 작동하고, 최신 프론트엔드 자산이 사용되고 있는지를 검증합니다.

- **서비스 접근 및 응답 확인:** 브라우저에서 커스텀 도메인 (`https://standardai.co.kr`) 또는 배포 직후 제공된 Cloud Run 기본 도메인으로 접속합니다. 페이지가 로드되며 관리자 로그인 또는 대시보드 화면이 보여야 합니다. 초기 로딩 시간 동안 Cloud Run이 컨테이너 인스턴스를 기동하므로 약간 지연될 수 있습니다 (Cold Start). 정상적인 경우 QualiJournal 관리자 UI의 메인 화면이 나타납니다. **예상 결과:** UI의 로고, 메뉴, 대시보드 구성요소 등이 보이고, 백엔드와 통신하여 KPI 지표 등이 표출됩니다 (초기 ADMIN_TOKEN 인증이 필요한 경우 로그인 화면이나 인증 요청이 나타날 수 있습니다). HTTP 200 응답과 함께 페이지 리소스(HTML, JS, CSS, 이미지)가 모두 로드되어야 합니다.

- **신판 HTML 적용 여부 확인:** 최근에 UI에 변경을 가했다면 (예: 새로운 디자인 적용, HTML/CSS 업데이트 등), 배포 후 실제 사이트에 그 변경사항이 반영됐는지 확인합니다. 변경 전후 UI를 비교하거나, 개발자 도구를 열어 로드된 정적 파일(예: `app.js`나 `index.html` 등)의 버전이나 내용이 최신인지 검증합니다. **주의사항:** Cloud Run 소스 배포시 Buildpacks 사용으로 정적 파일의 Last-Modified timestamp가 1980년으로 설정되는 이슈가 있어, 브라우저가 캐시를 잘못 사용할 가능성이 있습니다 [16]. 만약 새로운 UI가 보이지 않고 이전 버전이 계속 보인다면, 브라우저 캐시를 강력 새로고침(CTRL+F5)하거나 시크릿 모드로 접속해봅니다. 필요한 경우 Cloud Run에서 static 파일 서빙 시 ETag/Last-Modified 헤더를 보내지 않도록 설정하면 브라우저 캐시 문제를 완화할 수 있습니다 [16].

- **오류 로그 확인:** UI에 오류가 있거나 빈 페이지가 나온다면 **Cloud Run 로그**를 확인해야 합니다. GCP 콘솔의 Cloud Run > 해당 서비스 > 로그 에서 최신 로그를 살펴봅니다. 정적 파일 경로 404 에러가 있다면 .gcloudignore/.dockerignore 설정 미스로 파일이 누락된 것이므로 배포 패키지 구성을 고쳐야 합니다. 또는 Python 애플리케이션 예외가 발생했다면 traceback 로그가 찍혔을 수 있으니 원인을 분석합니다. Cloud Run 로그는 기본적으로 stdout/stderr 출력을 수집하며, FastAPI의 기본 로깅(uvicorn)이 요청 및 에러를 로깅하므로 이를 통해 어플리케이션 상태를 파악할 수 있습니다.

- **종합 확인:** 최종적으로 **페이지의 주요 요소**(예: KPI 수치, 차트, 버튼 등)가 예상대로 동작하고 최신 코드가 반영되어 있으면 배포가 성공한 것입니다. 이 때까지 도메인 연결, 인증서 적용, UI 로드에 문제 없다면 다음 단계로 넘어갑니다.

5. 백엔드 API 스모크 테스트 (상태 확인용 엔드포인트 검사)

배포 후 백엔드 서비스의 주요 API들이 정상적으로 작동하는지 간단한 **스모크 테스트**를 수행합니다. CLI 도구인 `curl`을 사용하여 Cloud Run 서비스의 엔드포인트를 호출하고 올바른 응답이 오는지 확인합니다. 특히 `/api/status` 같이 상태를 알려주는 엔드포인트나, `/api/export`처럼 핵심 기능 관련 엔드포인트를 점검합니다.

- `/api/status` **엔드포인트:** 애플리케이션이 구동 중인지 확인하기 위한 헬스체크/상태 확인 API가 있다면 호출해봅니다. 예를 들어 GET 요청으로 호출 가능하다면 다음과 같이 실행합니다:

  ```
  curl -X GET "https://standardai.co.kr/api/status" \
    -H "Admin-Token: 초기관리자토큰값123!"
  ```

  위 명령에서 `-H` 옵션은 ADMIN_TOKEN을 헤더로 포함한 예시입니다. 실제 구현에 따라 `Authorization: Bearer <token>` 형태나 `Admin-Token` 헤더를 사용하고 있다면 맞게 넣습니다. **예상 결과:** HTTP 200 상태 코드와 함께 `{ "status": "ok" }` 또는 관련 JSON 응답이 반환되거나, 간단히 "OK" 문자열이 응답될 것입니다. 응답이 정상이면 백엔드 애플리케이션이 잘 작동하고 있다는 뜻입니다. 인증 토큰이 잘못되었거나 누락된 경우 401 Unauthorized가 응답되므로, 이 경우 ADMIN_TOKEN 값을 확인해야 합니다.

- `/api/export` **엔드포인트:** 리포트 데이터를 파일로 추출하는 API로서, Markdown이나 CSV 내보내기 기능을 담당합니다. 이 엔드포인트는 데이터베이스 또는 내부 저장소에서 최신 리포트를 가져와 내보내거나, 요청 시 새로운 리포트를 생성해 반환할 수 있습니다. 이 API를 테스트하여 **백엔드 핵심 기능**이 동작함을 확인합니다:

  ```
  curl -X GET "https://standardai.co.kr/api/export?format=csv" \
    -H "Admin-Token: 초기관리자토큰값123!" \
    -o test_export.csv
  ```

  위 요청은 CSV 형식의 내보내기를 요청하는 예시입니다 (`?format=csv`는 구현에 따라 다를 수 있습니다). `-o` 옵션은 응답을 파일로 저장하는 기능입니다. **예상 결과:** 응답 HTTP 200과 함께 파일이 다운로드되고, 파일 크기가 0보다 크며 내용에 CSV 데이터(또는 Markdown 텍스트 등)가 들어 있어야 합니다. 예컨대 CSV의 헤더 행과 데이터 행들이 존재하거나, Markdown의 경우 보고서 텍스트가 포함됩니다.

동일한 방법으로 Markdown 형식(`?format=md` 등)도 호출해볼 수 있습니다. 혹은 `/api/export`가 요청 시점의 리포트를 생성한다면, 연속 호출 시 시간 차이에 따른 데이터 변화가 있음을 확인합니다. 응답 헤더의 `Content-Type`도 `text/csv` 또는 `text/markdown` 등 적절히 설정되었는지 확인하세요.

- **기타 엔드포인트:** 필요에 따라 `/api/report` (리포트 생성 트리거), `/api/summary` (요약 생성) 등의 엔드포인트가 있다면 비슷한 방식으로 호출해봅니다. POST 메서드가 요구되는 경우 `curl -X POST`로 호출하고, JSON 바디가 필요하면 `-d '{"param": "value"}' -H "Content-Type: application/json"` 등의 옵션으로 테스트합니다. **예상 결과:** 각 엔드포인트가 정의된 기능에 맞게 성공 응답(예: 200 OK 또는 작업 결과)이 반환되면 이상 없습니다.

- **주의사항:** 모든 API 호출에는 관리자 인증을 위해 **ADMIN_TOKEN을 헤더에 포함**해야 합니다. 이를 깜빡 잊으면 401 오류가 발생하므로, 스크립트나 툴에서 헤더 설정을 정확히 해야 합니다. 또한, curl 테스트 시 `-v` 옵션을 주면 상세 요청/응답 헤더와 상태를 볼 수 있어 문제 분석에 도움이 됩니다. 만약 특정 API가 제대로 동작

하지 않는다면 Cloud Run 로그에서 에러 메시지를 확인하고, 환경변수 누락이나 의존 서비스(DB, 외부 API) 연결 문제는 없는지 점검합니다.

6. 관리자 UI 주요 기능 점검

이제 관리자 웹 UI에서 제공하는 **핵심 기능들이 정상적으로 동작하는지** 검증합니다. QualiJournal 관리자 페이지에는 리포트 생성 및 미리보기, KPI 자동 새로고침, 슬라이더 조작, SSE(서버발송이벤트) 기반 실시간 업데이트, 요약 생성, 내보내기 등의 기능 버튼이 있을 것입니다. 각 기능을 하나씩 테스트하여 프론트엔드와 백엔드 간 연계가 올바른지 확인합니다:

- **보고서 생성 버튼:** 관리자 UI에서 "보고서 생성" 버튼을 클릭합니다. 이 기능은 새로운 품질 리포트를 생성하도록 백엔드 `/api/report` 등을 호출할 것입니다. 버튼을 누른 후 **예상 결과:** 보고서 생성 작업이 시작되면 UI에 진행 상태 표시(예: 로딩 스피너 or "생성 중..." 메시지)가 나타나고, 완료 시 성공 알림 또는 보고서 미리보기 업데이트가 이루어집니다. 백엔드에서는 보고서 생성 완료 시 결과를 반환하거나 SSE로 진행 상황 이벤트를 보낼 수 있습니다. **주의사항:** 보고서 생성에 시간이 다소 걸릴 경우 UI가 이를 반영하고 있는지 (예: 버튼 비활성화, 진행바 등) 확인합니다. 만약 보고서 생성이 실패하면 UI에 오류 메시지가 표시되어야 합니다. 이 과정에서 Cloud Run 로그에 생성 작업 관련 로그가 실시간으로 쌓이는지도 모니터링하면 좋습니다.

- **미리보기 기능:** 보고서 생성 후 "미리보기" 기능이 있다면, 최신 리포트를 화면에 보여주는 역할일 것입니다. **테스트:** 보고서 생성이 완료된 다음 "미리보기" 버튼(또는 탭)을 눌러 생성된 보고서 내용을 확인합니다. **예상 결과:** 최근에 생성된 리포트의 제목, 항목, 차트 등이 UI에 렌더링되어야 합니다. 예를 들어 Markdown이 HTML로 변환되어 표시되거나, 특정 리포트 요약 내용이 보일 것입니다. 만약 미리보기 시점에 백엔드에서 `/api/report/latest`와 같은 호출이 일어난다면 응답 지연 없이 데이터가 잘 오는지 확인합니다. **주의사항:** 미리보기 내용이 이전 버전 데이터로 나타나면 캐시 문제일 수 있으니, 개발자 콘솔 네트워크 탭에서 해당 요청의 응답이 최신인지 확인하고 필요시 강제 새로고침합니다.

- **KPI 자동 새로고침:** 대시보드에 KPI 지표(예: 주요 통계 숫자들)가 있고 이를 일정 주기마다 자동 갱신해주는 기능(토글 스위치 등)이 있다면, **테스트:** 해당 스위치를 ON으로 설정합니다. **예상 결과:** 활성화 시 UI에서 주기적으로 (예: 30초 or 1분 간격) KPI 수치가 백엔드 API를 통해 새로고침됩니다. 이를 확인하기 위해 잠시 기다리면서 변화 여부를 관찰합니다. 백엔드에서 KPI 값을 제공하는 API(`/api/kpi` 등)가 주기적으로 호출되고 200 응답을 주는지 브라우저 개발자 도구(Network 탭)로 볼 수 있습니다. **주의사항:** 자동 새로고침 간격이 너무 짧게 설정되면 과도한 요청을 유발할 수 있으므로 적절한지 확인하고, 토글 OFF시 추가 요청이 중지되는지도 확인합니다.

- **슬라이더 기능:** UI에 슬라이더 컨트롤이 있는 경우 (예: 날짜 범위 선택 슬라이더, 임곗값 조절 등), **테스트:** 슬라이더를 좌우로 움직여 봅니다. **예상 결과:** 슬라이더 값이 변함에 따라 연동된 데이터(그래프나 표 등)가 실시간으로 업데이트되거나, "적용" 버튼이 있다면 눌렀을 때 변경사항이 반영됩니다. 예를 들어 기간 슬라이더라면 해당 기간의 KPI나 리포트 내용이 바뀌어야 합니다. **주의사항:** 슬라이더 조작 중 콘솔 오류가 발생하지 않는지 확인하고, 극단값 (최소/최대)에서도 UI가 잘 동작하는지 테스트합니다. 백엔드 호출이 동작 안 하거나 JS 오류가 있다면 슬라이더와 이벤트 연동 코드를 수정해야 합니다.

- **SSE (Server-Sent Events) 스트림:** 관리자 UI에서 **실시간 업데이트**가 필요한 기능 (예: 보고서 생성 진행 로그, 서버 상태 모니터링 등)이 SSE로 구현되어 있을 수 있습니다. SSE는 HTTP 지속연결을 통해 서버가 보내는 이벤트를 실시간 수신하는 방식입니다. **테스트:** 보고서 생성 버튼을 눌렀을 때 혹은 별도의 "실시간 로그" 창이 있다면 SSE 연결이 이루어집니다. 개발자 콘솔 Network 탭에서 `EventStream` 또는 SSE 연결 요청이 보이는지 확인합니다 (예: `GET /api/stream` 요청이 Pending 상태로 유지). **예상 결과:** SSE 연결 후 서버 측에서 주기적으로 보내는 이벤트 데이터가 UI에 즉시 반영됩니다. 예를 들어 "챕터 1 생성 완료", "요약 생성 중..." 등의 중간 상태 메시지가 UI 로그 창에 실시간 추가됩니다. **주의사항:** SSE 연결이 아예 수립되지 않거나 이벤트가 오지 않는다면, 백엔드 SSE 구현(FastAPI의 `StreamingResponse` 등)과 프론트엔드

EventSource 리스너를 점검해야 합니다. 또한 Cloud Run은 기본적으로 SSE를 지원하지만, 시간이 너무 오래 걸리는 연결은 끊길 수 있으므로 이벤트 간격 및 연결 지속 시간을 적절히 설계해야 합니다.

- **요약 생성 기능:** 리포트에 대한 요약본을 AI 등을 통해 생성하는 기능이 있다면, **테스트:** "요약 생성" 버튼을 눌러봅니다. **예상 결과:** 클릭 시 백엔드에 요약 생성 요청(/api/summary 등)이 보내지고, 완료되면 UI에 요약 텍스트가 나타납니다. 이 과정이 SSE로 진행될 수도 있고, 완료 시 한꺼번에 결과를 받아 표시할 수도 있습니다. UI에는 "요약 생성 중..."와 같은 안내가 있다가 완료 후 요약 내용이 채워져야 합니다. **주의사항:** 요약 생성에는 외부 AI API 키(API_KEY 환경변수)가 사용될 수 있으므로, 만약 기능이 동작하지 않으면 API_KEY 설정 및 해당 외부 API의 응답 여부를 확인합니다. 오류 발생 시 UI에 에러 메시지가 표시되는지, 연속 클릭 방지 처리 등이 되어있는지도 확인하세요.

- **내보내기 기능:** "내보내기" 버튼은 현재 보고서나 데이터를 파일로 다운로드하게 해줍니다. **테스트:** 버튼을 클릭하여 Markdown 내보내기 (`.md`)와 CSV 내보내기가 각각 동작하는지 확인합니다. **예상 결과:** 클릭 시 브라우저 다운로드가 시작되고 파일이 저장됩니다. Markdown의 경우 `.md` 파일로, CSV의 경우 `.csv` 파일로 다운로드 되어야 합니다. 파일을 열어보면 현재 보고서 내용이 Markdown 형식 혹은 CSV 형식으로 정확히 들어있습니다 (예: 표, 리스트, 텍스트 등이 원본과 일치). **주의사항:** 내보내기 버튼을 누를 때 백엔드 요청 (`/api/export`)이 정상 응답해야 하며, 만약 다운로드가 시작되지 않으면 팝업 차단 등 브라우저 이슈가 아닌지 확인합니다. 파일명도 일반적으로 `report_20231015.md`처럼 날짜가 포함되도록 되어 있을 수 있는데, 실제 이름 형식이 의도한 대로인지 확인하세요. 연속 클릭 시 중복 다운로드 방지나 로딩 상태 표시 등이 있다면 그 역시 확인합니다.

- **전반적인 UI 검사:** 위 주요 기능 외에도, 관리자 UI의 링크, 필터 입력, 페이지 전환 등의 부가 기능도 간단히 점검합니다. 예를 들어 페이지 새로고침 시 로그인 유지 여부, 브라우저 콘솔에 오류 메시지 존재 여부, 모바일 화면에서의 표시 등도 확인하면 좋습니다. 모든 기능을 하나씩 확인함으로써 **최종 사용자에게 제공되는 관리자 시스템이 완전하고 오류 없이 작동**함을 보증할 수 있습니다.

7. 일일 백업 스크립트 설정 (PowerShell 예시)

운영 환경에서 생성된 리포트와 데이터를 **주기적으로 백업**하여 보관하는 것은 중요합니다. 이를 위해 Windows 환경이라면 PowerShell 스크립트를 사용하여 **매일 자동으로 MD/CSV 보고서를 다운로드**하고 저장할 수 있습니다. 예를 들어, 새벽 시간에 관리자 API의 `/api/export`를 호출하여 Markdown과 CSV를 받아 로컬 또는 파일 서버에 저장하는 스크립트를 작성하고 Windows 작업 스케줄러에 등록하는 방법을 소개합니다.

- **PowerShell 백업 스크립트 예시:** 아래는 매일 실행되어 **Markdown 보고서와 CSV 데이터를 저장**하는 PowerShell 스크립트입니다. 해당 파일을 예를 들어 `backup-qualijournal.ps1`로 저장합니다:

```
# 설정: API 엔드포인트 및 인증 토큰
$baseUrl = "https://standardai.co.kr/api"
$adminToken = "초기관리자토큰값123!"  # 환경 변수나 별도 안전한 저장소에서 가져오는 것을 권장
$today = (Get-Date).ToString("yyyy-MM-dd")

# 헤더 구성 (Authorization 헤더에 토큰 전달)
$headers = @{ "Admin-Token" = $adminToken }

try {
    # 1. Markdown 보고서 백업
    $mdUri = "$baseUrl/export?format=md"
    $mdOutput = "QualiJournal_Report_$today.md"
    Invoke-WebRequest -Uri $mdUri -Headers $headers -OutFile $mdOutput
```

```
    Write-Host "Markdown 보고서 다운로드 완료: $mdOutput"

    # 2. CSV 데이터 백업
    $csvUri = "$baseUrl/export?format=csv"
    $csvOutput = "QualiJournal_Data_$today.csv"
    Invoke-WebRequest -Uri $csvUri -Headers $headers -OutFile $csvOutput
    Write-Host "CSV 데이터 다운로드 완료: $csvOutput"
}
catch {
    Write-Error "백업 중 오류 발생: $($_.Exception.Message)"
}
```

스크립트 설명: 먼저 `baseUrl`과 `adminToken` 변수를 설정합니다. 실제 운영 환경에서는 토큰을 스크립트에 직접 쓰기보다는 환경변수나 보안 파일에서 읽어오는 것이 안전합니다. `Get-Date`를 이용해 오늘 날짜 문자열을 생성하고, 이를 파일명에 포함시켜 매일 다른 이름으로 저장되도록 했습니다. 각 백업 단계에서 `Invoke-WebRequest` cmdlet을 사용하여 GET 요청을 보내고 `-OutFile`로 결과를 파일로 저장합니다. 요청 시 `-Headers` 파라미터를 통해 ADMIN_TOKEN을 포함한 헤더를 전달하고 있습니다. (일부 API에서 Authorization Bearer 토큰 방식을 쓴다면 `$headers = @{ "Authorization" = "Bearer $adminToken" }` 형태로 변경해야 합니다 [17].) `Write-Host`를 통해 각 단계 완료를 콘솔에 출력하며, try/catch로 감싸 예외 발생 시 에러를 표시하게 했습니다.

- **수동 실행 테스트:** 스크립트를 작업 스케줄러에 등록하기 전에, 관리자 권한 PowerShell 프롬프트에서 수동으로 실행해봅니다: `PS C:\> .\backup-qualijournal.ps1`. **예상 결과:** 현재 디렉터리에 `QualiJournal_Report_YYYY-MM-DD.md`와 `QualiJournal_Data_YYYY-MM-DD.csv` 두 파일이 생성되고, 각각 Markdown 보고서와 CSV 데이터가 들어있어야 합니다. 파일을 열어 내용이 정상적으로 보이는지 확인합니다. 만약 API 호출에 실패하면, PowerShell이 오류를 출력할 것입니다. 이 경우 `Invoke-WebRequest` 실행 시 응답 상태 코드를 확인하거나 (`$?`와 `$LASTEXITCODE` 등 활용), `Invoke-RestMethod`를 사용해 JSON 응답 메시지를 파싱하여 문제 원인을 파악할 수도 있습니다.

- **Windows 작업 스케줄러 등록:** 스크립트가 정상 작동하면 Windows Task Scheduler에 일일 작업으로 추가합니다. **설정 예:** 새 작업 만들기 > 트리거: 매일 새벽 3시, 동작: 프로그램 시작 - `powershell.exe` 그리고 인수로 `-File "C:\경로\backup-qualijournal.ps1"`. 작업 실행 계정은 권한이 있는 사용자로 설정하고, 성공적으로 동작하는지 다음 날 로그(작업 히스토리)와 결과 파일 생성을 확인합니다. Windows가 아닌 리눅스 환경이라면 `crontab`에 `curl` 명령을 넣어주거나, 해당 서버에서 주기적 백업 스크립트를 실행할 수도 있습니다.

- **주의사항:** 백업 받은 파일들은 주기적으로 정리하거나 보관 정책을 정해야 합니다. 예를 들어 30일치 보관 후 오래된 파일 삭제 등을 고려합니다. 또한, 백업 스크립트를 저장한 경로의 보안도 중요합니다(토큰이 노출되므로 해당 파일 권한을 제한). 더 안전하게 하려면 ADMIN_TOKEN을 Windows 자격 증명 저장소나 Azure Key Vault와 연계하거나, PowerShell SecureString으로 암호화하여 보관하는 방법도 있습니다. 그러나 기본적으로 위 스크립트 방식은 편의성과 빠른 구현을 위한 것이며, 토큰이 노출되지 않는 내부망 환경에서 사용하기를 권장합니다.

8. 자동화 고려사항 (Cloud Scheduler를 통한 리포트 자동 생성)

QualiJournal 시스템 운영을 자동화하기 위해, **Cloud Scheduler**를 사용하여 정해진 스케줄에 Cloud Run 서비스의 특정 API를 호출할 수 있습니다. 예를 들어 **매일 아침 정기적으로 리포트가 생성**되도록 `/api/report` 엔드포인트를

Cloud Scheduler로 트리거하면, 사람이 수동으로 버튼을 누르지 않아도 보고서 생성 작업이 수행됩니다. 이 절에서는 Cloud Scheduler로 Cloud Run을 호출하는 방법과 주의사항을 설명합니다.

- **Cloud Scheduler 작업 생성:** GCP 콘솔에서 Cloud Scheduler 서비스를 이용하거나 gcloud CLI로 작업을 만들 수 있습니다. 여기서는 gcloud 명령어 예시를 보여줍니다. 매일 오전 7시에 `/api/report`를 POST로 호출하는 작업을 생성한다고 가정합니다:

```
gcloud scheduler jobs create http daily-report-job \
 --schedule "0 7 * * *" \
 --timezone "Asia/Seoul" \
 --http-method POST \
 --uri "https://standardai.co.kr/api/report" \
 --headers "Admin-Token: 초기관리자토큰값123!"
```

명령 설명: `jobs create http`는 HTTP 호출 작업을 생성하는 명령입니다. `--schedule`에는 Cron 형식의 스케줄을 지정합니다 (위 예시는 매일 7:00). `--timezone`은 해석할 시간대를 서울로 지정한 것입니다. `--http-method`는 사용할 HTTP 메서드로, 보고서 생성은 리소스 생성 동작이므로 POST로 지정했습니다. `--uri`는 호출할 URL이고, `--headers` 플래그로 요청에 포함할 헤더(인증 토큰)를 넣었습니다 [18]. 이 때 헤더는 하나만 직접 지정할 수 있는데, 필요한 경우 여러 헤더를 JSON 또는 && 구분으로 넣을 수도 있습니다. Cloud Scheduler는 기본적으로 User-Agent 등의 헤더를 자체 설정하며, **Content-Type**이 필요하면 `--headers "Content-Type: application/json"` 식으로 추가 가능합니다. (단순 GET/POST로 토큰만 보내는 경우 Content-Type는 생략 가능). 작업 생성 후 `gcloud scheduler jobs describe daily-report-job`으로 확인하면 설정이 올바르게 등록되었는지 검증할 수 있습니다.

- **Cloud Scheduler 인증 설정:** Cloud Run 서비스가 **허용된 비공개 서비스**로 설정되어 IAM 인증이 필요하다면, Cloud Scheduler에서 **인증된 호출**을 구성해야 합니다. 그러나 현재 QualiJournal ADMIN_TOKEN으로 자체 인증을 하고 있으므로 Cloud Run 서비스는 **--allow-unauthenticated** 상태일 가능성이 높습니다. 이 경우 Cloud Scheduler에서 별도의 OIDC 인증 설정 없이 위와 같이 Admin-Token 헤더만으로 호출이 가능합니다. (만약 Cloud Run이 IAM 인증을 요구하도록 설정되었다면, Cloud Scheduler 작업 생성시 `--oauth-service-account-email` 및 `--oauth-token-scope` 옵션을 설정하여 서비스 계정 OIDC 토큰이 포함되도록 해야 합니다 [19] [20].)

- **예상 결과:** 스케줄러 작업이 등록되고 활성화되면, 매일 지정된 시간에 Cloud Run 엔드포인트가 호출됩니다. 이를 확인하는 방법:

- Cloud Scheduler 콘솔에서 해당 job의 **최근 실행** 시간을 확인하고 **최근 실행 상태**가 성공으로 표시되는지 본다.
- Cloud Run 로그에서 `/api/report`에 대한 요청 로그가 매일 7:00 경 기록되는지 확인한다. (요청자가 Cloud Scheduler일 경우 특별한 User-Agent 또는 `X-CloudScheduler` 헤더가 있을 수 있음).
- 생성된 보고서가 의도대로 DB나 파일에 저장되거나, 이후 `/api/export`로 다운로드 시 최신 날짜로 반영되는지 등을 검증한다.

- **주의사항:** Cloud Scheduler에서 HTTP 호출 실패시 (예: Cloud Run 쪽 오류로 500 응답 등) **재시도 정책**을 설정할 수 있습니다. 기본값으로 3번까지 재시도 등이 있으니 필요한 경우 조정합니다. 또한 ADMIN_TOKEN이 URL이나 로그 등에 노출되지 않도록 헤더로 전달하고, GCP Workflow나 Cloud Tasks 등을 통해 더 복잡한 시나리오(예: 호출 후 후속 처리)를 구현할 수도 있습니다. 단순 스케줄링은 Cloud Scheduler가 적합하며, 작업이 실행되지 않으면 서비스 계정 권한(Cloud Scheduler가 Cloud Run을 호출하려면 기본 Compute 서비스 계정에 Cloud Run Invoke 권한 필요 등) 문제일 수 있으니 권한을 확인합니다. 마

지막으로, 자동 생성된 보고서가 너무 자주 쌓이면 저장소 용량 문제 등이 발생할 수 있으므로 보관 주기나 정리 작업도 고려해야 합니다.

9. 인증 및 보안 점검 사항

QualiJournal 관리자 시스템은 **관리자 전용**으로 설계되어 있으므로, **보안 및 인증** 설정을 철저히 점검하는 것이 중요합니다. 특히 ADMIN_TOKEN을 이용한 인증 메커니즘을 사용하므로, 토큰 관리와 노출 방지, 로그 모니터링에 신경써야 합니다.

- **ADMIN_TOKEN 인증 적용 확인:** 우선 Cloud Run에 배포된 FastAPI 백엔드가 **모든 민감 API 요청에 대해 ADMIN_TOKEN 검증을 수행**하는지 재확인합니다. 일반적으로 FastAPI dependency나 미들웨어를 통해 입력 받은 토큰을 환경 변수 `ADMIN_TOKEN`과 비교하여 일치하지 않으면 401 에러를 반환하도록 구현했을 것입니다. `/api/status` 같이 헬스체크 용도를 제외한 **모든 관리자 기능 API에 인증 데코레이터가 누락되지 않았는지** 확인합니다. 코드 수정이나 배포 과정에서 혹시 일부 엔드포인트에 토큰 검증이 빠져있다면 잠재적인 보안 구멍이 될 수 있습니다.

- **토큰 노출 방지:** ADMIN_TOKEN 값은 절대로 공개 레포지토리나 클라이언트 코드에 노출되면 안 됩니다. 프론트엔드 JS에서 해당 토큰을 직접 사용한다면 (예: AJAX 호출 시 헤더로 포함), 이 토큰이 브라우저에 노출되므로 **출입 통제**가 된 환경에서만 쓰여야 합니다. 이상적으로는 별도의 로그인 절차를 두고 세션을 관리하는 것이 바람직하나, 현재는 ADMIN_TOKEN 자체가 비밀번호 역할을 하는 간단 인증으로 보입니다. 따라서 **토큰 값을 추측하기 어렵게 충분히 복잡하게 만들고 정기적으로 변경**하는 것을 고려해야 합니다. 예를 들어 영문+숫자+특수문자 조합의 32자 이상 랜덤 문자열이면 무작위 대입 공격을 피할 수 있습니다. 토큰을 변경할 때는 새로운 값으로 환경변수를 업데이트하고, 프론트엔드 설정(만약 토큰을 프론트엔드에서 상수로 들고 있다면)도 재배포해야 하므로 계획을 가지고 실행합니다.

- **로그 점검 방법:** Cloud Run은 모든 요청에 대해 로그를 남길 수 있으므로, 주기적으로 로그를 분석하여 **비정상적인 접근 시도**나 오류를 모니터링해야 합니다. GCP 로그 뷰어에서 해당 서비스의 로그를 필터링해 `status: 401`로 검색하면, 인증 실패 내역이 나옵니다. 만약 ADMIN_TOKEN이 노출되어 누군가 오남용한다면 비정상적인 시간대나 IP에서 다수의 호출이 발생할 수 있으니 유의합니다. 또한 **토큰을 쿼리스트링으로 전달하지 않는 것**이 중요합니다. 쿼리에 넣으면 Cloud Run 로그에 URL 전체가 기록되어 토큰이 노출될 수 있습니다. 가능하면 Authorization 헤더나 쿠키를 이용하는 것이 보안상 안전합니다. 현재 구현에서 Admin-Token을 헤더로 받고 있으므로 올바른 접근입니다. 로그에 민감정보가 남지 않도록 `INFO` 수준 이상의 애플리케이션 로그에는 토큰 값을 출력하지 않는지 확인합니다 (예: `logging.info(f"Admin token {token} authenticated")` 같은 코드가 없어야 함).

- **HTTPS 및 인증서:** 앞서 설정한대로 사용자 도메인에 HTTPS가 적용되었는지 항상 확인하고, 인증서 갱신 만료일을 챙겨야 합니다. Cloud Run의 자동 인증서는 갱신을 관리해주지만, 도메인 매핑이 Preview 기능임을 감안하여 만약 만료 이슈가 있었다면 사전에 알려주는 모니터링을 하면 좋습니다.

- **역할 및 권한:** 운영 팀 내에서 Cloud Run 서비스나 Cloud Scheduler 등을 다루는 사람이 여럿이라면, GCP IAM 권한을 최소 권한 원칙으로 부여합니다. 예를 들어 Cloud Run 관리자, 뷰어 등을 적절히 나누고, Cloud Build 권한도 관리합니다. 특히 Cloud Run에 새로운 리비전을 배포하는 권한은 제한된 인원에게만 주고, 배포 Pipeline을 통해 이루어지도록 하면 사람이 실수로 잘못된 설정으로 배포하는 일을 줄일 수 있습니다.

- **기타 보안 점검:** Cloud Run 특성상 외부에 공개된 엔드포인트이므로, ADMIN_TOKEN이 있다 해도 **과도한 요청이나 DDoS에 대한 대비**가 필요할 수 있습니다. 간단한 대책으로는 Cloud Run 설정에서 동시 요청 처리 제한이나 최대 인스턴스 수를 설정해 두어 서비스가 폭주하지 않게 하는 것입니다. 또한 IP 제한이 필요한 경우 Cloud Run을 VPC 내부로 넣거나, Cloud Armor와 Load Balancer를 앞단에 구성해야 하지만 이는 아키텍처가 복잡해지므로 현재 스코프에서는 토큰 기반 인증만으로 충분한지 판단해야 합니다. 마지막으로,

ADMIN_TOKEN 방식에서 더 나아가야 한다면 **JWT 기반 인증**이나 **OAuth** 등을 도입하는 방안도 고려할 수 있으나, 이는 별도 개발이 수반됩니다.

10. 배포 실패 시 Plan-B: 수동 Docker 이미지 빌드 및 배포 스크립트

가끔 Cloud Run의 소스 빌드/배포가 오류가 나거나, 문제를 진단하기 어려울 때가 있습니다. 이런 경우를 대비하여, **Plan-B**로서 로컬에서 Docker 이미지를 빌드하고 Cloud Run에 수동으로 배포하는 방법을 갖춰두면 유용합니다. 이 접근법은 CI/CD 없이도 문제를 재현하고 이미지를 커스터마이징하여 배포할 수 있으며, Cloud Build를 우회하기 때문에 빌드 환경을 완전히 통제할 수 있다는 장점이 있습니다. 아래는 Docker 이미지 수동 배포 절차와 예시 스크립트입니다.

- **Docker 이미지 수동 빌드:** 로컬 개발 PC나 빌드 서버에서 Docker를 이용해 이미지를 빌드합니다. 우선 프로젝트 루트에 올바른 **Dockerfile**이 있어야 합니다 (예: Python 이미지를 베이스로 uvicorn으로 server_quali.py를 실행하도록 한 Dockerfile). 터미널에서 다음 명령을 실행하여 이미지를 빌드합니다:

```
docker build -t gcr.io/<PROJECT_ID>/qualijournal-admin:manual .
```

이 명령은 현재 디렉토리의 Dockerfile을 기반으로 이미지를 빌드하고, 태그를 GCP 컨테이너 레지스트리(혹은 Artifact Registry) 경로로 지정합니다. `<PROJECT_ID>`는 GCP 프로젝트 ID로, 이미지를 push하려면 프로젝트 컨테이너 레지스트리에 대한 태그여야 합니다. (Container Registry `gcr.io`를 쓰거나, Artifact Registry를 사용할 경우 `REGION-docker.pkg.dev/PROJECT/REPO/이미지명:태그` 형태를 사용합니다.) 빌드가 성공하면 로컬 Docker 데몬에 이미지가 생성됩니다.

- **이미지 레지스트리 푸시:** 빌드한 이미지를 GCP로 업로드해야 Cloud Run에서 사용할 수 있습니다. 우선 한 번만 `gcloud auth configure-docker`를 실행해 도커가 GCR/AR에 푸시할 권한을 설정합니다. 그런 다음:

```
docker push gcr.io/<PROJECT_ID>/qualijournal-admin:manual
```

를 실행하여 이미지를 푸시합니다. **예상 결과:** 푸시가 진행되며 이미지 레이어들이 업로드되고, 완료 후 GCP 콘솔의 Artifact Registry(또는 Container Registry)에 해당 이미지가 나타납니다. (만약 Artifact Registry를 사용했다면, `<PROJECT_ID>`와 `<REGION>`에 해당하는 경로로 push해야 하고, 사전에 Artifact Registry 리포지토리를 생성해두어야 합니다. 기본 Cloud Run 소스 배포를 한 번이라도 했다면 `cloud-run-source-deploy`라는 리포지토리가 자동 생성되어 있을 수 있습니다 [21] .)

- **Cloud Run에 이미지 배포:** 이제 업로드된 컨테이너 이미지를 사용하여 Cloud Run 서비스를 업데이트합니다. gcloud 명령에서 `--image` 플래그를 사용하면 소스 대신 특정 이미지를 배포할 수 있습니다:

```
gcloud run deploy qualijournal-admin \
 --image gcr.io/<PROJECT_ID>/qualijournal-admin:manual \
 --region <REGION> \
 --platform managed \
 --allow-unauthenticated \
 --update-env-vars ADMIN_TOKEN=초기관리자토큰값
123!,API_KEY=abcd1234efgh5678ijkl
```

예상 결과: Cloud Run이 새로운 리비전을 생성하고 수동 빌드한 이미지로 서비스가 교체됩니다. `--update-env-vars`를 사용하여 기존 환경 변수 값들을 유지 또는 업데이트할 수 있습니다 [22]. (만약 모든 변수를 재지정하지 않으면 `--clear-env-vars`로 싹 지운 후 하나만 넣는 실수가 생길 수 있으니 `--update-env-vars` 사용을 권장합니다.) 배포 완료 후 서비스가 정상 동작하는지, 이전 소스 빌드와 동등한 기능이 제공되는지 테스트합니다. 이 시나리오에서는 빌드 문제가 있었던 경우라면, 로컬에서 빌드 시 로그를 확인해 원인을 찾을 수 있고 수정된 이미지를 바로 배포함으로써 장애 시간을 줄일 수 있습니다.

- **배포 스크립트 예시:** 위의 수동 과정은 일련의 커맨드로 구성되어 있으므로, 스크립트로 만들어두면 편리합니다. 예를 들어 배포를 자동화한 `deploy_manual.sh` (Linux/macOS) 내용을 소개합니다:

```
#!/bin/bash
PROJECT_ID="your-gcp-project"
IMAGE_NAME="qualijournal-admin"
REGION="asia-northeast3"
TAG="manual-$(date +%Y%m%d%H%M)"  # 현재 시간으로 태그 생성 (옵션)

# 1. Docker 이미지 빌드
echo "Building Docker image..."
docker build -t "gcr.io/$PROJECT_ID/$IMAGE_NAME:$TAG" . || { echo "Docker build 실패"; exit 1; }

# 2. 이미지 푸시
echo "Pushing Docker image to gcr.io..."
docker push "gcr.io/$PROJECT_ID/$IMAGE_NAME:$TAG" || { echo "Docker push 실패"; exit 1; }

# 3. Cloud Run 배포
echo "Deploying to Cloud Run..."
gcloud run deploy $IMAGE_NAME \
 --image "gcr.io/$PROJECT_ID/$IMAGE_NAME:$TAG" \
 --region $REGION --platform managed --allow-unauthenticated \
 --update-env-vars "ADMIN_TOKEN=초기관리자토큰값123!,API_KEY=abcd1234efgh5678ijkl" || {
echo "gcloud deploy 실패"; exit 1; }

echo "배포 완료 - 이미지 태그: $TAG"
```

이 스크립트를 사용하면 하나의 명령으로 빌드부터 배포까지 진행됩니다. 태그에 타임스탬프를 붙이면 매 배포마다 고유한 이미지를 배포하여 추적하기 쉽습니다. 배포 후 문제가 없다면, 필요시 Cloud Run 콘솔에서 이전 리비전을 삭제하거나 트래픽을 모두 새 리비전에 할당했는지 확인합니다.

- **주의사항:** 수동 배포를 사용할 때는 Cloud Build/Source 기반 배포와 병행하면 이미지 관리가 혼란스러울 수 있으므로, **긴급 대처용**으로 활용하는 것이 좋습니다. 장기적으로는 Cloud Build YAML을 만들어 빌드 과정을 코드화하거나, CI 파이프라인에서 이미지 빌드 & 배포 단계를 관리하는 것을 고려하세요. 그래도 Plan-B가 필요한 경우, Dockerfile 및 관련 리소스를 최신 상태로 유지해야 막상 문제 발생 시 바로 빌드해서 배포할 수 있습니다. 또한, 수동으로 배포한 이미지에 문제가 있으면 **이전 정상 리비전으로 롤백**하는 것도 하나의 방법입니다. Cloud Run 콘솔의 **Revision** 탭에서 이전 리비전으로 100% 트래픽을 돌리거나, CLI로 `gcloud run services update-traffic --to-revisions` 명령을 사용해 롤백할 수 있습니다. 항상 배포 전후에 기능 테스트를 철저히 해서 문제 발생 시 신속히 대응하도록 합니다.

以上の 가이드에 따라 QualiJournal 관리자 시스템을 안정적으로 배포, 운영 및 관리할 수 있습니다. 각 단계마다 명령 실행 결과를 확인하고, 주의사항을 준수하여 운영하면 도메인 연결된 Cloud Run 서비스에서 원활히 관리자 기능을 제공할 수 있을 것입니다. 필요시 Google Cloud 공식 문서와 레퍼런스를 추가로 참고하며 지속적으로 환경을 개선해 나가십시오.

---

1 2 16 21 Deploy services from source code | Cloud Run | Google Cloud
https://cloud.google.com/run/docs/deploying-source-code

3 Cloud Run doesn’t work with Express static middleware - Stack Overflow
https://stackoverflow.com/questions/58532968/cloud-run-doesn-t-work-with-express-static-middleware

4 5 docker - `gcloud builds submit` for Cloud Run - Stack Overflow
https://stackoverflow.com/questions/55846611/gcloud-builds-submit-for-cloud-run

6 7 8 9 22 Configure environment variables for services | Cloud Run | Google Cloud
https://cloud.google.com/run/docs/configuring/services/environment-variables

10 11 12 13 14 15 Mapping custom domains | Cloud Run | Google Cloud
https://cloud.google.com/run/docs/mapping-custom-domains

17 Invoke-RestMethod with Authorization header - Stack Overflow
https://stackoverflow.com/questions/54191266/invoke-restmethod-with-authorization-header

18 20 Verifying Cloud Scheduler requests in Google Cloud Run with ...
https://jackcuthbert.dev/blog/verifying-google-cloud-scheduler-requests-in-cloud-run-with-typescript

19 Use authentication with HTTP targets | Cloud Scheduler
https://cloud.google.com/scheduler/docs/http-target-auth