ChatGPT

Cloud Run에서 FastAPI 앱 배포 오류 진단 및 해결

1. `server_quali.py` 모듈이 인식되지 않는 원인

오류 메시지: “Error loading ASGI app. Could not import module 'server_quali'” – 이는 Uvicorn이 `server_quali` 모듈을 가져오는 데 실패했음을 의미합니다. 주요 원인은 다음과 같습니다:

- **작업 디렉토리/모듈 경로 문제:** Uvicorn을 실행하는 현재 디렉토리에 `server_quali.py`가 없거나, 모듈 경로가 잘못 지정된 경우 발생합니다 [1] [2]. 예를 들어, 코드가 `admin/` 폴더에 있고 모듈 경로를 지정하지 않았다면 Uvicorn이 해당 모듈을 찾지 못합니다.
  해결: Dockerfile에서 `WORKDIR`를 앱 코드 위치로 설정하고, `server_quali.py`를 이미지에 복사했는지 확인하세요. 또한 Uvicorn 실행 시 현재 경로에 파일이 없다면, **모듈 경로를 포함**해야 합니다 (예: 파일이 `admin` 폴더에 있다면 `uvicorn admin.server_quali:app` 형태로 실행하거나, 실행 전에 `cd admin`으로 워킹 디렉토리를 변경합니다 [1]).

- **내부 임포트 실패:** 모듈 내부에서 필요한 패키지를 찾지 못해 ImportError가 발생한 경우에도 동일한 증상이 나타날 수 있습니다. `server_quali.py`를 살펴보면 `.env` 파일 로딩을 위해 `python-dotenv`가 필요하지만, **requirements.txt에 누락**되어 있습니다. 이처럼 필수 패키지가 설치되지 않으면 모듈 import 단계에서 예외가 발생하고, Cloud Run 로그에는 단순히 모듈 로드 실패만 나타날 수 있습니다 [3].
  해결: `requirements.txt`에 누락된 패키지를 추가합니다. 특히 `from dotenv import load_dotenv`를 사용하고 있으므로 `python-dotenv` 패키지를 추가하여 재배포하세요. 추가로, `auth_utils`나 `logging_setup`과 같은 내부 모듈 임포트가 있다면 해당 파일들도 컨테이너에 포함되어 있어야 합니다. ImportError의 상세 내용은 Cloud Run 로그에 바로 표시되지 않을 수 있으므로, 로그와 의존 패키지 목록을 꼼꼼히 대조해 누락된 모듈을 모두 설치해야 합니다 [3].

- **FastAPI 앱 객체 미정의:** Uvicorn은 `<모듈>:<변수>` 형식으로 ASGI 앱을 찾습니다. 만약 `server_quali.py` 내에 FastAPI 앱 인스턴스 (`app`)가 **전역(scope)으로 정의되지 않은 경우** 모듈 임포트에 실패하거나 앱을 찾지 못합니다. 예를 들어 `app = FastAPI()` 선언이 파일 수준에 없거나, `if __name__ == "__main__":` 블록 내에만 존재하면 문제가 됩니다 [4].
  해결: `server_quali.py` 최상단에 `app = FastAPI()`를 선언해 **전역 변수로 앱 객체를 정의**해야 합니다 [4]. 또한 Uvicorn 실행 명령에서 이 변수 이름(`app`)과 모듈명을 정확히 지정해야 합니다. (예: `uvicorn server_quali:app` – 여기서 `app`은 코드 내 FastAPI 인스턴스 이름과 일치해야 합니다.) 만약 코드에서 앱 객체 이름을 다르게 지었다면 (`api = FastAPI()` 등), Uvicorn 명령에서도 동일한 이름을 사용해야 합니다.

2. Cloud Run 시작 커맨드(ENTRYPOINT/CMD) 설정 문제

Cloud Run은 **컨테이너의 ENTRYPOINT/CMD**로 애플리케이션을 실행합니다. 이 설정이 올바르지 않으면 앱이 정상적으로 구동되지 않습니다:

- **잘못된 실행 커맨드:** Dockerfile의 CMD가 `uvicorn`을 통해 앱을 실행하지 않거나 형식이 틀린 경우 문제가 됩니다. 예를 들어 CMD를 `python server_quali.py`로 지정했다면, 이 스크립트는 **서버를 기동하지 않고** 앱 객체만 정의한 뒤 종료되므로 Cloud Run에서 포트를 리슨하지 않게 됩니다. Uvicorn으로 ASGI 서버를 띄우지 않으면 클라우드 런은 준비 상태에 진입하지 못합니다 [5].
  해결: Dockerfile의 CMD를 올바르게 설정하세요. 일반적으로 FastAPI 앱은 Uvicorn으로 실행하므로,

`CMD ["uvicorn", "server_quali:app", "--host", "0.0.0.0", "--port", "8080"]` 와 같이 지정합니다. 이때 `server_quali:app`은 `<모듈이름>:<FastAPI인스턴스>` 형식입니다. 이미 Dockerfile에서 Uvicorn을 사용하고 있었다면, 모듈명이나 경로 철자가 정확한지 (`server_quali` 철자나 경로) 확인하세요. 또한 colon(`:`) 다음의 앱 변수 이름도 정확히 일치해야 합니다. 형식 오류가 있는 경우 "Import string must be in format '<module>:<attribute>'" 같은 에러가 발생하거나 모듈을 못 찾게 됩니다 [1].

- **ENTRYPOINT 중복 실행 문제:** 일부 경우, `server_quali.py` 내부에서 `uvicorn.run()`을 호출하는 `__main__` 블록을 넣고, Dockerfile에서 `python server_quali.py`로 실행하려는 시도가 있습니다. Cloud Run 환경에서는 **이중으로 서버를 실행하는 설정을 피해야** 합니다. 예컨대, 코드 내 `if __name__ == "__main__": uvicorn.run(app, ...)`와 Dockerfile의 `CMD uvicorn ...`를 함께 쓰면 하나만 사용하거나 정리해야 합니다.
  해결: Cloud Run에서는 보통 **Dockerfile의 CMD로 Uvicorn을 실행**하고, 코드 내에서는 `uvicorn.run`을 호출하지 않습니다 [6]. 만약 코드에 `if __name__ == "__main__":` 블록이 있다면, Dockerfile에서 직접 Uvicorn을 실행할 경우 그 블록은 불필요하므로 제거하세요 [6]. 반대로, Dockerfile에서 Python으로 스크립트를 실행할 경우에는 그 스크립트가 Uvicorn을 기동하는지 확인해야 합니다. 종합적으로, **하나의 경로로만 서버를 띄우도록 일원화**해야 합니다. 일반적인 권장사항은 "Dockerfile에서 Uvicorn을 CMD로 실행하고, 코드내 main블록에서는 Uvicorn을 호출하지 않는 것"입니다 [7]. 이렇게 하면 Cloud Run이 컨테이너 시작 시 Uvicorn 프로세스를 곧바로 실행하여 ASGI 앱을 서빙하게 됩니다.

3. requirements.txt 패키지 누락 및 버전 호환성

환경 설정 문제로 인한 오류 가능성도 검토해야 합니다. 요구사항 파일의 패키지 구성이 잘못되면 앱이 정상 구동되지 않을 수 있습니다:

- **필수 패키지 누락:** 앞서 언급한 대로 `python-dotenv` 등이 누락되어 **ImportError**가 발생하면 애플리케이션 로드에 실패합니다. Cloud Run 빌드 로그에서 pip 설치 경고나 실패가 있었는지 확인하고, 모든 import가 충족되도록 해야 합니다. 예를 들어, `server_quali.py`에서 사용하는 `fastapi`, `uvicorn[standard]`, `pydantic`, `python-multipart`, `aiofiles` 등이 requirements.txt에 모두 포함되어 있는지 살펴보세요. FastAPI 애플리케이션을 실행하기 위해 **FastAPI와 Uvicorn이 모두 설치**되어 있어야 함은 기본 전제입니다 [8].
  해결: 누락된 패키지를 추가하고 이미지 재빌드 후 배포합니다. 특히 `dotenv` 관련 오류를 막으려면 `python-dotenv`**를 추가**하세요. 또한 Cloud Run 로그에 다른 모듈을 찾을 수 없다는 메시지가 없는지 확인하세요 (예: `ModuleNotFoundError: No module named 'X'`). 그런 메시지가 있다면 해당 모듈이 requirements.txt에 있는지 점검하면 됩니다.

- **버전 불일치:** requirements.txt의 버전 범위도 검토하세요. 현재 FastAPI와 Uvicorn 버전 제한이 (`fastapi>=0.110,<1`, `uvicorn[standard]>=0.24,<1`)로 되어 있는데, 서로 호환되는지 확인해야 합니다. FastAPI 0.8x~0.9x 버전대와 Uvicorn 0.22+는 일반적으로 호환되며, FastAPI 0.100+도 Uvicorn 최신 버전들과 함께 동작합니다. 다만 **Pydantic 버전**은 FastAPI와 직접 연관되므로, FastAPI 0.110.x는 Pydantic 1.x를 사용합니다. 이미 요구사항에 `pydantic==1.*`로 고정해두었지만, 만약 잘못된 버전(예: Pydantic 2.x)이 설치되었다면 FastAPI 모델에서 오류가 날 수 있습니다.
  해결: FastAPI와 Uvicorn은 특정 호환성 이슈가 보고되지 않았으므로 현재 제약으로 충분해 보입니다. 다만 **설치된 실제 버전**을 확인하여 의도한 범위 내인지 보세요. Cloud Run 배포 이미지 안에서 `pip show fastapi uvicorn pydantic` 등을 실행해 버전을 검증할 수 있습니다. 필요하다면 버전 범위를 조정하거나 고정하여, 예컨대 FastAPI와 Uvicorn의 known stable 조합을 사용하도록 합니다. (FastAPI 문서나 릴리즈 노트를 참고하세요.) 현재 발견된 문제로는 패키지 누락이 더 의심되므로, 우선 누락 패키지 보완 후 재테스트를 권장합니다.

4. `app = FastAPI()` 선언 및 Uvicorn 앱 인식 문제

구조적인 문제로, `server_quali.py` 내부에 FastAPI **앱 객체(**`app`**)가 정의되어 있는지** 확인이 필요합니다. Uvicorn은 모듈을 임포트한 후 해당 모듈에서 지정된 앱 객체를 찾아 실행합니다. 이 과정에서 다음 사항을 점검하세요:

- **전역 범위 앱 선언:** `app` 객체 생성이 함수 내부나 `__main__` 블록 안이 아닌, **모듈 전역에 있어야** 합니다 [4]. 질문 내용상 `server_quali.py`에 `app = FastAPI()` 선언이 있는지가 관건인데, 만약 없었다면 Uvicorn은 `app`이라는 변수를 찾지 못해 실행에 실패합니다. (이 경우 오류 메시지는 module import error와는 별도로 "`module 'server_quali' has no attribute 'app'`" 형태로 나왔을 것입니다.) 코드 확인 결과 다행히 `app = FastAPI(title="...")`가 정의되어 있으므로, **앱 객체 부재는 직접적 원인은 아니지만**, 실수로 이 부분을 제거하거나 잘못된 위치에 두지 않도록 해야 합니다.

- **이름 불일치:** FastAPI 앱 객체의 이름과 Uvicorn 명령 인자가 일치해야 합니다. 예를 들어 코드를 수정하여 `app`이 아니라 `api`라는 변수에 FastAPI 인스턴스를 할당했다면, Uvicorn 실행 명령도 `server_quali:api`로 맞춰야 합니다 [9]. 이름이 어긋나면 "attribute not found" 오류가 발생합니다. 현재 코드는 `app`으로 맞춰져 있으므로, Dockerfile의 CMD에서도 `:app`으로 지정한 것이 맞는지 재확인하세요.

- `__main__` **가드 처리:** 앞서 언급한 대로, 코드 내부에 `if __name__ == "__main__": ...` 블록이 있는 경우 **앱 객체 생성은 그 바깥에서 이뤄져야** 합니다. 앱 객체 생성까지도 해당 블록 안에 들어있다면, Uvicorn이 모듈 import할 때 앱이 만들어지지 않으므로 빈 모듈이 됩니다. 우리의 `server_quali.py`에서는 `app = FastAPI()`가 최상단에 있으므로 이 부분은 문제가 없었습니다. 다만, `uvicorn.run()` 호출이 만약 `__main__` 블록에 있었다면 (지금은 없는 것으로 확인), Docker 환경에서는 제거하거나 무시됩니다 [6]. **정리하면,** Cloud Run에서는 앱 객체는 전역에, Uvicorn 호출은 코드 외부(컨테이너 커맨드)에서 하는 구조가 가장 안정적입니다.

5. Cloud Run 8080 포트 리슨 설정

포트 문제: Cloud Run은 **환경 변수** `$PORT` **(기본값 8080)**로 지정된 포트에서 어플리케이션이 수신 대기(listen)하기를 기대합니다. 현재 증상으로 볼 때, 컨테이너가 올바른 포트에서 리슨하지 않아 `Revision not ready` 상태가 된 것입니다 (즉, 헬스체크 실패) [5]. 고려해야 할 사항:

- **포트 번호 설정:** Uvicorn의 기본 포트는 8000이므로, 별도 지시 없으면 컨테이너 내부에서 8000으로 열립니다. Cloud Run은 외부에서 8080으로 접속을 시도하므로, 내부 서버가 8080을 사용하지 않으면 연결되지 않습니다. 따라서 **Uvicorn 실행 시** `--port 8080`**으로 지정**해야 합니다 [7]. (8080 이외의 포트로 Cloud Run 서비스를 구성하지 않는 한, 8080로 고정해도 무방합니다. 혹 향후 Cloud Run 설정에서 포트를 변경할 가능성을 대비한다면 `$PORT` 환경변수를 사용하는 편이 좋습니다.)

- **호스트 설정:** 기본으로 Uvicorn은 localhost(127.0.0.1)에서만 바인딩합니다. Cloud Run에서는 트래픽을 수신하려면 `--host 0.0.0.0`로 **모든 인터페이스에 바인딩**해야 외부 요청을 받을 수 있습니다. 이 설정이 빠지면, 컨테이너 내부에서만 열리고 외부에선 닿지 않아 마찬가지로 헬스체크에 실패합니다. `--host 0.0.0.0` 옵션을 반드시 포함하세요 [7].

- **Dockerfile 내 포트 표기:** Cloud Run은 Dockerfile의 `EXPOSE` 지시자를 필수로 요구하지는 않지만, **관례상** `EXPOSE 8080`을 넣어주면 해당 포트가 서비스 포트임을 명시하게 됩니다. 일부 로컬 테스트나 다른 환경에서 이미지 재사용 시 이 정보가 유용할 수 있습니다. (Cloud Run 자체는 `$PORT` 환경변수로 포트를 전달하므로 EXPOSE 유무와 상관없이 동작합니다.)

해결: Dockerfile에 `EXPOSE 8080`을 추가하고, CMD 또는 ENTRYPOINT에서 **포트 설정을 8080으로 지정**합니다. 예를 들면:

```
EXPOSE 8080
CMD ["uvicorn", "server_quali:app", "--host", "0.0.0.0", "--port", "8080"]
```

위와 같이 설정하면 Cloud Run 환경에서 자동으로 8080 포트를 듣게 되고, "PORT=8080"을 찾지 못해 실패하는 일이 없습니다. 만약 `$PORT` 환경 변수를 직접 활용하고 싶다면, Dockerfile에 `ENV PORT 8080`을 명시한 뒤 CMD를 쉘 모드로 `uvicorn ... --port $PORT` 실행하는 방법도 있습니다 [7]. 하지만 Cloud Run에서는 기본적으로 `$PORT`가 8080으로 주어지므로 큰 차이는 없습니다. 중요한 것은 **컨테이너가 8080포트를 수신하도록 Uvicorn을 설정**하는 것입니다. 설정을 수정한 후에는 Cloud Run에 이미지를 다시 배포하고, **로그에 "Uvicorn running on 0.0.0.0:8080"** 등의 메시지가 뜨는지 확인하세요. 정상적으로 8080에서 시작됐다면, `Revision Ready` 상태로 전환되어 서비스가 응답을 받기 시작할 것입니다.

위의 모든 진단 포인트를 하나씩 적용하고 수정하면, Cloud Run 환경에서 QualiJournal FastAPI 앱이 정상 구동될 것입니다. **요약하면**, `server_quali.py` 모듈을 올바르게 인식시키기 위해 파일 경로 및 모듈 임포트 환경을 바로잡고, **필요한 패키지를 모두 설치**하며, **Dockerfile CMD**를 통해 Uvicorn이 올바른 호스트(0.0.0.0)와 포트(8080)에서 앱을 실행하도록 해야 합니다. 이러한 변경 후에 배포를 다시 수행하면, Cloud Run 서비스가 “Ready” 상태로 전환되고 FastAPI 엔드포인트들이 정상적으로 응답할 것입니다 [5] [7].

---

[1] [9] python - FastAPI throws an error (Error loading ASGI app. Could not import module "api") - Stack Overflow
https://stackoverflow.com/questions/60819376/fastapi-throws-an-error-error-loading-asgi-app-could-not-import-module-api

[2] [4] [8] FastAPI Error loading ASGI app. Could not import module 'main' | bobbyhadz
https://bobbyhadz.com/blog/python-fastapi-error-loading-asgi-app-could-not-import-module

[3] [5] [6] [7] Cloud Run deployment failing for FastAPI - Stack Overflow
https://stackoverflow.com/questions/78623517/cloud-run-deployment-failing-for-fastapi