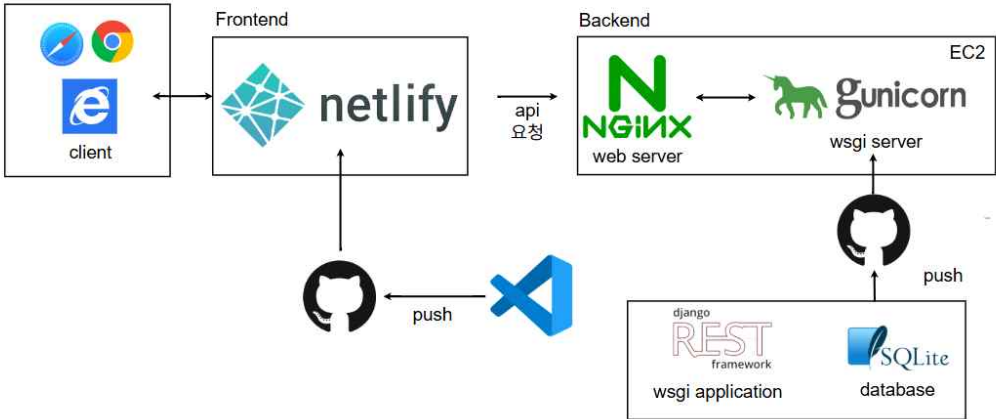


오픈소스프로젝트 제품 구성 배포 운영자료

과 제 명	퀴디: 일기 기반 감정 흐름 시각화 및 감정 기반 콘텐츠 큐레이션 서비스											
팀명	01팀, 라이옹											
팀원	<table><tr><td>오찬주</td><td>2022110699</td><td>정치외교학전공</td></tr><tr><td>한수연</td><td>2022112587</td><td>역사교육과</td></tr><tr><td>황민영</td><td>2022111520</td><td>경영정보학과</td></tr></table>			오찬주	2022110699	정치외교학전공	한수연	2022112587	역사교육과	황민영	2022111520	경영정보학과
오찬주	2022110699	정치외교학전공										
한수연	2022112587	역사교육과										
황민영	2022111520	경영정보학과										
수행기간	2025년 3월~2025년 6월											
전체 구조	<div></div>											
그림 1 배포 구성도												
<p>본 프로젝트는 프론트엔드(React+TypeScript)와 백엔드(Django REST API)가 하나의 GitHub 저장소 내에서 각각 src/frontend와 src/backend 디렉토리를 분리되어 관리되는 모노레포 구조를 따른다. 프론트엔드는 Netlify를 통해 정적 사이트로 자동 배포되며, 백엔드는 AWS EC2 인스턴스에 HTTPS 기반으로 배포되어 운영된다. 두 서비스는 HTTPS를 통해 통신하며 CORS 정책 설정을 통해 안전하게 API 요청 및 응답을 주고 받는 구조로 설계 되어 있다.</p>												
프론트엔드	Netlify 사용											
<p>본 프로젝트의 프론트엔드는 Netlify를 이용해 배포 및 운영된다. Netlify는 정적 웹사이트 및 SPA(Single Page Application)를 위한 자동화된 CI/CD 배포 플랫폼으로, GitHub 저장소와 연동하여 소스코드의 변경 사항을 자동으로 감지하고 빌드 및 배포를 수행할 수 있다. 이를 통해 복잡한 인프라 설정 없이도 빠르게 배포 파이프라인을 구축할 수 있다는 장점이 있다.</p>												

Netlify의 배포 설정은 다음과 같다. Base directory는 프론트엔드 소스코드가 위치한 src/frontend로 지정하였으며, 빌드 명령어는 CI= npm run build를 사용해 React 프로젝트를 빌드한다. 빌드 결과물은 src/frontend/dist 디렉토리에 생성되며, 이 디렉토리가 publish directory로 설정되어 최종적으로 사용자에게 제공된다. Netlify는 이러한 과정을 통해 HTTPS가 적용된 정적 웹사이트를 자동으로 배포하고, 브라우저에서 언제든지 접근 가능한 상태를 유지한다.

배포 절차는 다음과 같다. 개발자가 프론트엔드 코드를 수정한 뒤 main 브랜치에 푸시하면, Netlify는 GitHub Webhook을 통해 변경을 감지하고 지정된 base 디렉토리(src/frontend)에서 빌드 명령어를 실행한다. 빌드된 결과물을 dist 디렉토리에서 추출한 후, 자동으로 Netlify 서버에 배포하고 URL을 갱신한다. 이때 빌드는 프론트엔드 디렉토리에 변경이 감지될 때만 수행되도록 설정되어 있어 불필요한 리소스 사용을 방지한다.

결과적으로 본 프로젝트는 GitHub와 Netlify 간의 자동화된 연동을 통해 수동 개입 없이 빠르고 안정적인 프론트엔드 배포 환경을 구축하였다. Netlify에서 제공하는 무료 HTTPS 기반 커스텀 도메인 기능을 활용하여 <https://liong.netlify.app/> 도메인으로 서비스를 제공하고 있다.

백엔드

AWS EC2 사용

본 프로젝트의 백엔드는 Django REST Framework(DRF)를 기반으로 개발되었으며, AWS EC2 인스턴스를 활용하여 배포 및 운영되고 있다. 백엔드는 RESTful API 방식으로 프론트엔드와 통신하며, 클라이언트의 요청을 받아 데이터베이스와 상호작용하고 그에 대한 응답을 반환하는 역할을 수행한다.

배포 환경은 Ubuntu 기반의 EC2 인스턴스이며 Django 앱을 실행하기 위한 wsgi 서버로는 Gunicorn을 사용하였다. Gunicorn은 Python WSGI(Web Server Gateway Interface) 애플리케이션을 실행하기 위한 서버로 효율적인 백엔드 운영을 가능하게 한다. 또한, Gunicorn은 systemd 서비스 유닛을 통해 백그라운드에서 안정적으로 실행되도록 설정되었으며 서버 재부팅 시 자동으로 시작되어 운영 편의성이 높다.

클라이언트로부터의 실제 요청은 Nginx 웹 서버를 통해 처리된다. Nginx는 외부에서 들어오는 요청을 Gunicorn으로 전달하며, /static/, /media/와 같은 정적 파일 요청을 직접 처리함으로써 응답 속도를 높인다. /etc/nginx/sites-available/default 파일에 프록시 설정을 적용하여 루트(/) 경로로 들어온 요청은 내부의 8000번 포트(Gunicorn이 리스닝 중인 포트)로 전달되도록 구성하였다.

보안을 강화하기 위해 HTTPS 프로토콜을 적용하였다. 이를 위해 무료 SSL 인증서 발급 도구인 Certbot과 인증기관 Let's Encrypt를 활용하였다. SSL 인증서를 발급받고, 이를 Nginx 설정에 자동으로 적용하여 HTTPS 기반의 보안 통신을 지원하도록 하였다.

백엔드 배포 절차는 다음과 같다. 먼저 GitHub에 연동된 저장소에서 최신 소스코드를 EC2 인스턴스에 가져온다. 이후 Python 가상환경을 활성화한 뒤 필요한 패키지를 설치하고 정적 파일을 수집한다. Gunicorn의 systemd 유닛 파일(/etc/systemd/system/gunicorn.service)과 Nginx 설정(/etc/nginx/sites-available/default)을 작성 및 적용한다. 마지막으로, Certbot을 통해 SSL 인증서를 발급받고 Nginx를 재시작하면 HTTPS 기반 배포가 완료된다.

결과적으로 백엔드는 Gunicorn, Nginx, Certbot의 조합을 통해 안정적이고 보안이 강화된 서버 환경에서 운영되고 있으며, 프론트엔드에서 발생하는 모든 API 요청은 HTTPS가 적용된 <https://quddy.shop/api/>를 통해 안전하게 처리된다.