# QClick 배포 준비 체크리스트

## 🚀 배포 전 필수 준비사항

### Phase 1: 환경 설정 및 보안 (1-2일)

#### 1.1 환경변수 설정

- [ ] \*\*프로덕션 환경변수 파일 생성\*\*

```bash

# .env.production

DATABASE\_URL=postgresql://username:password@host:port/database

JWT\_SECRET=강력한\_시크릿\_키\_최소\_32자

GEMINI\_API\_KEY=실제\_GEMINI\_API\_키

OPENAI\_API\_KEY=실제\_OPENAI\_API\_키

NAVER\_CLIENT\_ID=실제\_NAVER\_CLIENT\_ID

NAVER\_CLIENT\_SECRET=실제\_NAVER\_CLIENT\_SECRET

CORS\_ORIGINS=https://your-domain.com,https://www.your-domain.com

```

- [ ] \*\*API 키 보안 검토\*\*

- [ ] 모든 API 키가 환경변수로 관리되는지 확인

- [ ] 하드코딩된 API 키가 없는지 검색

- [ ] API 키 권한 설정 확인

#### 1.2 보안 강화

- [ ] \*\*Rate Limiting 적용\*\*

```python

# main-api/api/utils/rate\_limiting.py

from fastapi import HTTPException

from slowapi import Limiter, \_rate\_limit\_exceeded\_handler

from slowapi.util import get\_remote\_address

from slowapi.errors import RateLimitExceeded

limiter = Limiter(key\_func=get\_remote\_address)

@limiter.limit("100/minute")

async def rate\_limited\_endpoint(request):

# 엔드포인트 구현

pass

```

- [ ] \*\*CORS 설정 검토\*\*

```python

# 허용된 도메인만 설정

CORS\_ORIGINS = [

"https://your-domain.com",

"https://www.your-domain.com"

]

```

- [ ] \*\*입력 검증 강화\*\*

- [ ] 모든 API 엔드포인트에 Pydantic 모델 적용

- [ ] SQL Injection 방지 확인

- [ ] XSS 방지 확인

#### 1.3 데이터베이스 준비

- [ ] \*\*PostgreSQL 스키마 생성\*\*

```sql

-- 데이터베이스 생성

CREATE DATABASE qclick\_production;

-- 사용자 생성

CREATE USER qclick\_user WITH PASSWORD 'strong\_password';

GRANT ALL PRIVILEGES ON DATABASE qclick\_production TO qclick\_user;

```

- [ ] \*\*마이그레이션 스크립트 준비\*\*

```python

# scripts/migrate.py

from sqlalchemy import create\_engine

from models import Base

engine = create\_engine(DATABASE\_URL)

Base.metadata.create\_all(engine)

```

### Phase 2: 클라우드 서비스 설정 (1일)

#### 2.1 Vercel (프론트엔드)

- [ ] \*\*Vercel 계정 생성\*\*

- [ ] \*\*프로젝트 연결\*\*

```bash

npm install -g vercel

vercel login

vercel --prod

```

- [ ] \*\*환경변수 설정\*\*

```bash

vercel env add REACT\_APP\_API\_URL

vercel env add REACT\_APP\_ENVIRONMENT

```

- [ ] \*\*도메인 설정\*\*

- [ ] 커스텀 도메인 연결

- [ ] SSL 인증서 확인

#### 2.2 Railway (백엔드)

- [ ] \*\*Railway 계정 생성\*\*

- [ ] \*\*프로젝트 배포\*\*

```bash

npm install -g @railway/cli

railway login

railway init

railway up

```

- [ ] \*\*환경변수 설정\*\*

```bash

railway variables set DATABASE\_URL=postgresql://...

railway variables set JWT\_SECRET=...

railway variables set GEMINI\_API\_KEY=...

```

- [ ] \*\*PostgreSQL 데이터베이스 연결\*\*

- [ ] Railway PostgreSQL 서비스 생성

- [ ] 연결 문자열 확인

#### 2.3 Render (마이크로서비스)

- [ ] \*\*Render 계정 생성\*\*

- [ ] \*\*QName 서비스 배포\*\*

- [ ] GitHub 저장소 연결

- [ ] 빌드 명령어 설정

- [ ] 환경변수 설정

- [ ] \*\*QText 서비스 배포\*\*

- [ ] 동일한 과정 반복

### Phase 3: CI/CD 파이프라인 (1일)

#### 3.1 GitHub Actions 설정

- [ ] \*\*워크플로우 파일 생성\*\*

```yaml

# .github/workflows/deploy.yml

name: Deploy to Production

on:

push:

branches: [main]

jobs:

test:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v3

- name: Test Backend

run: |

cd services/main-api

pip install -r requirements.txt

pytest

- name: Test Frontend

run: |

cd frontend

npm install

npm run build

deploy:

needs: test

runs-on: ubuntu-latest

steps:

- name: Deploy to Vercel

uses: amondnet/vercel-action@v20

with:

vercel-token: ${{ secrets.VERCEL\_TOKEN }}

vercel-org-id: ${{ secrets.ORG\_ID }}

vercel-project-id: ${{ secrets.PROJECT\_ID }}

vercel-args: '--prod'

```

#### 3.2 자동 배포 설정

- [ ] \*\*Vercel 자동 배포\*\*

- [ ] \*\*Railway 자동 배포\*\*

- [ ] \*\*Render 자동 배포\*\*

### Phase 4: 성능 최적화 (1일)

#### 4.1 프론트엔드 최적화

- [ ] \*\*번들 크기 분석\*\*

```bash

npm install -g webpack-bundle-analyzer

npm run build

webpack-bundle-analyzer dist/stats.json

```

- [ ] \*\*이미지 최적화\*\*

- [ ] WebP 형식 사용

- [ ] 이미지 압축

- [ ] Lazy loading 적용

- [ ] \*\*캐싱 전략\*\*

```javascript

// service-worker.js

const CACHE\_NAME = 'qclick-v1';

const urlsToCache = [

'/',

'/static/js/bundle.js',

'/static/css/main.css'

];

```

#### 4.2 백엔드 최적화

- [ ] \*\*API 응답 캐싱\*\*

```python

from fastapi\_cache import FastAPICache

from fastapi\_cache.backends.redis import RedisBackend

@app.get("/api/cached-data")

@cache(expire=300) # 5분 캐시

async def get\_cached\_data():

return {"data": "cached"}

```

- [ ] \*\*데이터베이스 쿼리 최적화\*\*

- [ ] 인덱스 추가

- [ ] N+1 쿼리 문제 해결

- [ ] 쿼리 성능 분석

### Phase 5: 모니터링 및 로깅 (1일)

#### 5.1 로깅 시스템

- [ ] \*\*구조화된 로깅\*\*

```python

import structlog

logger = structlog.get\_logger()

@app.middleware("http")

async def log\_requests(request: Request, call\_next):

start\_time = time.time()

response = await call\_next(request)

process\_time = time.time() - start\_time

logger.info(

"request\_processed",

path=request.url.path,

method=request.method,

status\_code=response.status\_code,

process\_time=process\_time

)

return response

```

#### 5.2 모니터링 도구

- [ ] \*\*Sentry 에러 트래킹\*\*

```python

import sentry\_sdk

from sentry\_sdk.integrations.fastapi import FastApiIntegration

sentry\_sdk.init(

dsn="your-sentry-dsn",

integrations=[FastApiIntegration()],

traces\_sample\_rate=1.0,

)

```

- [ ] \*\*성능 모니터링\*\*

- [ ] Vercel Analytics 설정

- [ ] Railway 메트릭스 확인

- [ ] Google Analytics 설정

### Phase 6: 테스트 및 검증 (1일)

#### 6.1 기능 테스트

- [ ] \*\*API 엔드포인트 테스트\*\*

```bash

# Postman 컬렉션 실행

newman run qclick-api-tests.postman\_collection.json

```

- [ ] \*\*사용자 플로우 테스트\*\*

- [ ] 회원가입 → 로그인 → 서비스 사용 → 결제

- [ ] 관리자 기능 테스트

- [ ] 에러 시나리오 테스트

#### 6.2 성능 테스트

- [ ] \*\*부하 테스트\*\*

```bash

# Apache Bench 테스트

ab -n 1000 -c 10 https://your-api-domain.com/api/health

```

- [ ] \*\*응답 시간 측정\*\*

- [ ] API 응답 시간 < 2초

- [ ] 페이지 로딩 시간 < 3초

#### 6.3 보안 테스트

- [ ] \*\*OWASP Top 10 검사\*\*

- [ ] \*\*API 보안 스캔\*\*

- [ ] \*\*SSL/TLS 설정 확인\*\*

## 🔍 배포 후 검증 체크리스트

### 즉시 확인 (배포 직후)

- [ ] \*\*헬스체크 엔드포인트\*\*

```bash

curl https://your-api-domain.com/health

```

- [ ] \*\*기본 기능 동작 확인\*\*

- [ ] 홈페이지 로딩

- [ ] 로그인/회원가입

- [ ] API 응답

- [ ] \*\*SSL 인증서 확인\*\*

```bash

openssl s\_client -connect your-domain.com:443

```

### 24시간 모니터링

- [ ] \*\*에러율 모니터링\*\* (< 1%)

- [ ] \*\*응답 시간 모니터링\*\* (< 2초)

- [ ] \*\*서버 리소스 모니터링\*\*

- [ ] \*\*사용자 행동 분석\*\*

### 1주일 후 검토

- [ ] \*\*성능 지표 분석\*\*

- [ ] \*\*사용자 피드백 수집\*\*

- [ ] \*\*개선점 도출\*\*

- [ ] \*\*다음 단계 계획 수립\*\*

## 🚨 롤백 계획

### 롤백 트리거 조건

- [ ] 에러율 > 5%

- [ ] 응답 시간 > 5초

- [ ] 치명적 보안 취약점 발견

- [ ] 데이터 손실 발생

### 롤백 절차

1. \*\*즉시 롤백 실행\*\*

```bash

# Vercel 롤백

vercel rollback

# Railway 롤백

railway rollback

```

2. \*\*문제 분석 및 수정\*\*

3. \*\*수정된 버전 재배포\*\*

4. \*\*검증 후 정상 운영\*\*

## 📊 성공 지표

### 기술적 지표

- [ ] \*\*업타임\*\*: > 99.5%

- [ ] \*\*응답 시간\*\*: < 2초

- [ ] \*\*에러율\*\*: < 1%

- [ ] \*\*페이지 로딩\*\*: < 3초

### 비즈니스 지표

- [ ] \*\*사용자 등록\*\*: 목표 달성

- [ ] \*\*서비스 사용률\*\*: 목표 달성

- [ ] \*\*결제 전환율\*\*: 목표 달성

## 🎯 다음 단계

### 배포 완료 후 1주일

1. \*\*사용자 피드백 수집\*\*

2. \*\*성능 최적화\*\*

3. \*\*추가 기능 개발 계획\*\*

### 배포 완료 후 1개월

1. \*\*사용자 행동 분석\*\*

2. \*\*수익화 모델 검토\*\*

3. \*\*확장 계획 수립\*\*

---

\*\*체크리스트 완료 후 배포를 진행하세요!\*\* ✅

**현재 확인해야 할 사항**

**QName 서비스와 QText 서비스가 실행되고 있는지 확인해주세요:**

* QName 서비스: http://localhost:8002
* QText 서비스: http://localhost:8003

**현재 어떤 기능들이 정상 작동하는지 알려주세요:**

1. 프론트엔드 접속이 잘 되나요?
2. 회원가입/로그인이 가능한가요?
3. QName, QText 서비스 페이지에 접속할 수 있나요?
4. API 문서(http://localhost:8001/docs)에 접속할 수 있나요?

이 정보를 바탕으로 다음 단계를 계획하겠습니다! 🚀