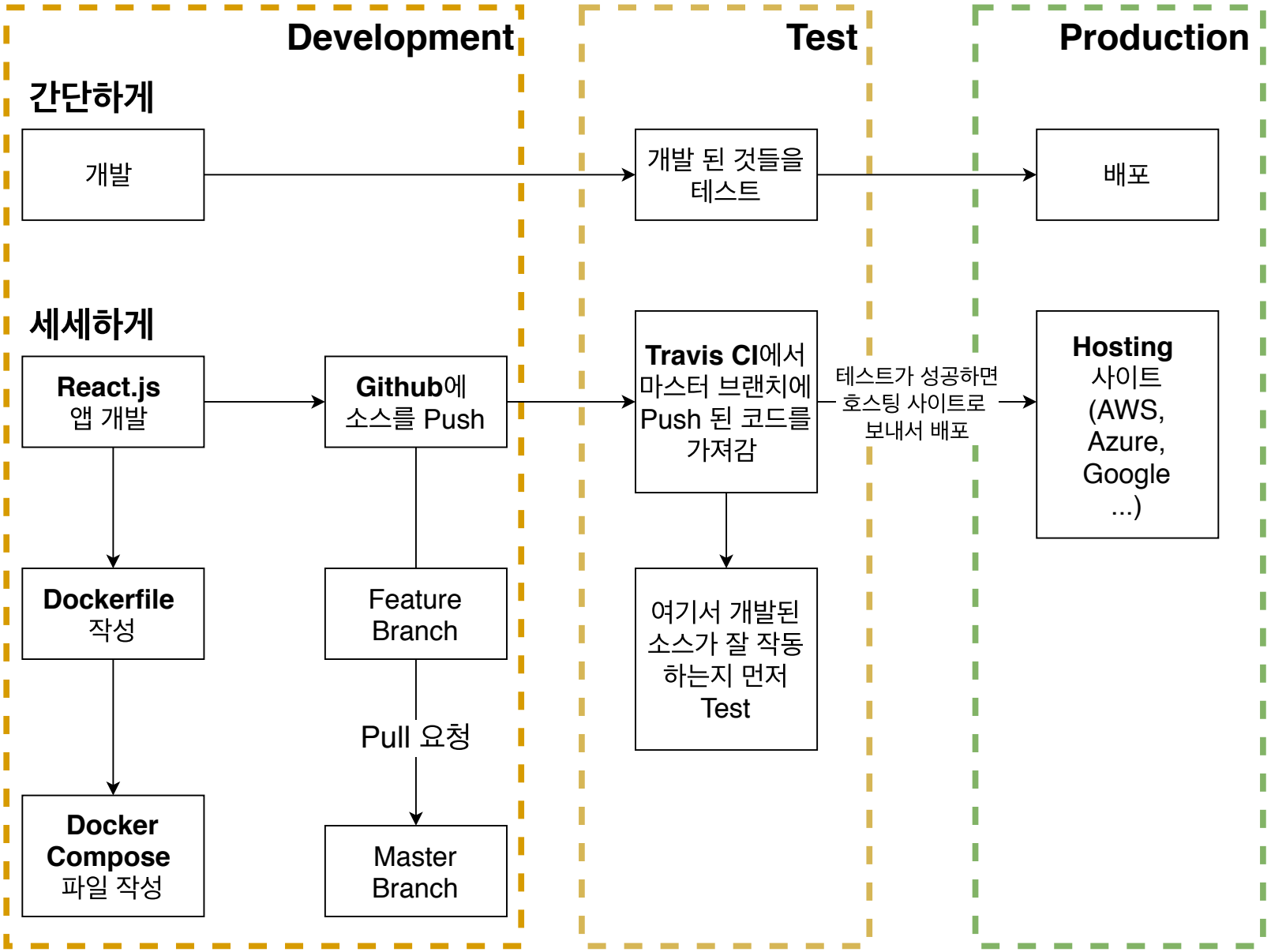
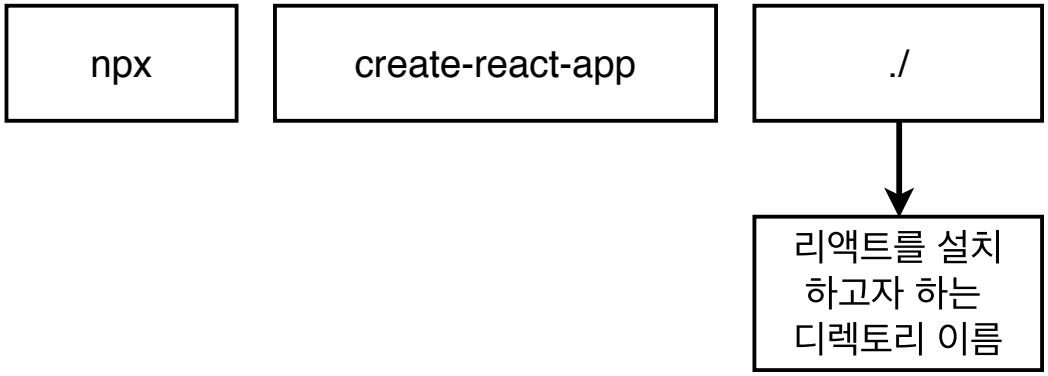
간단한 어플을 실제로 배포해보기(개발 부분)

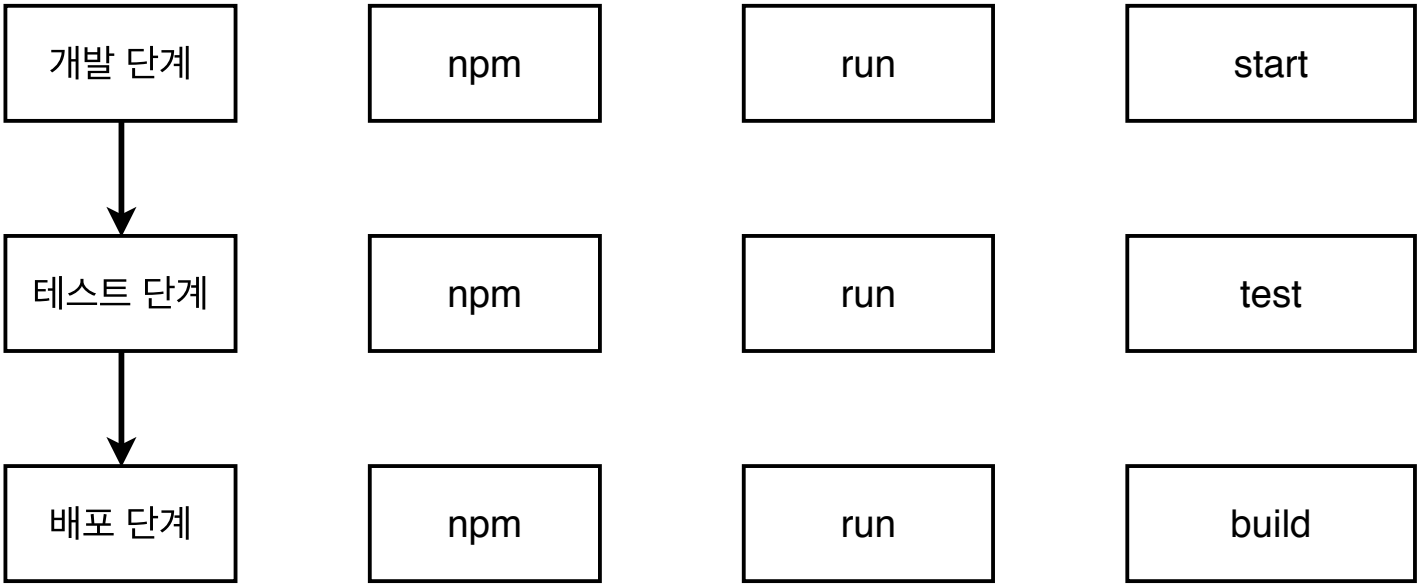
이번 섹션 설명



리액트 앱 설치하기

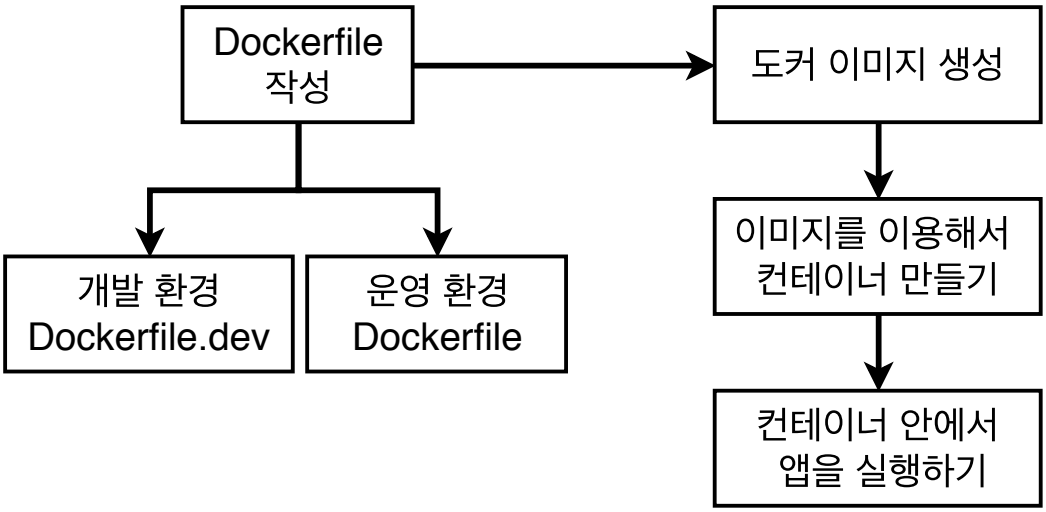
리액트를 설치하기 위한 명령어

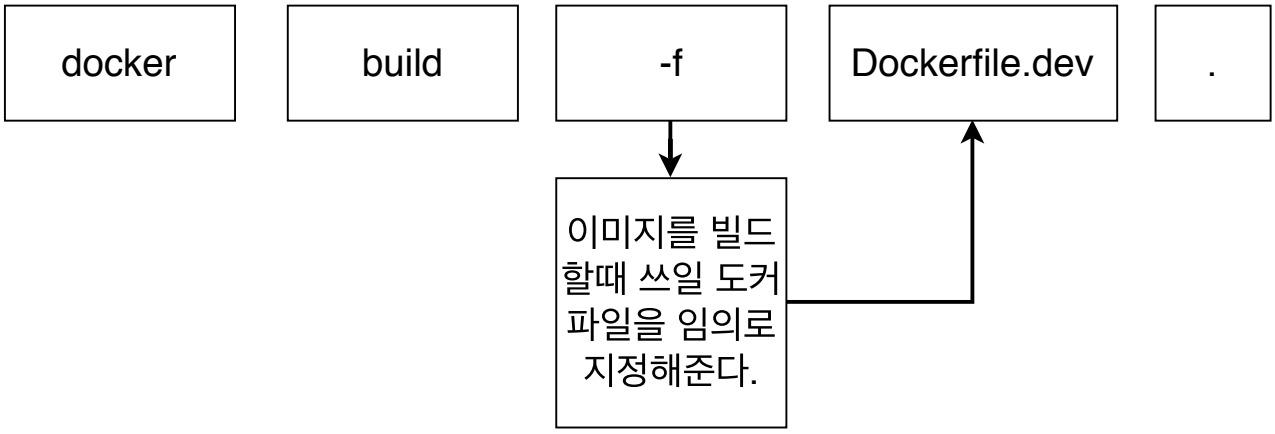


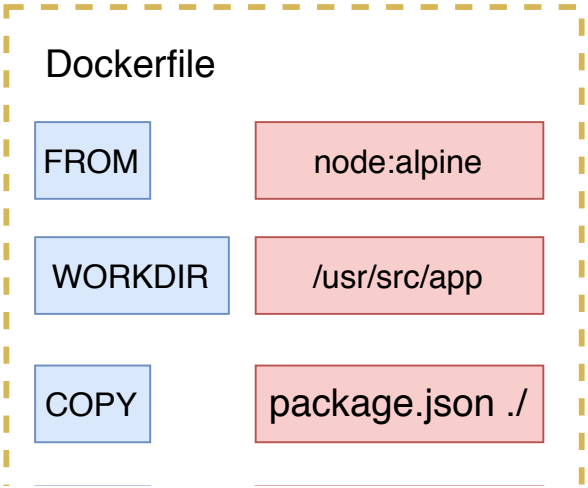


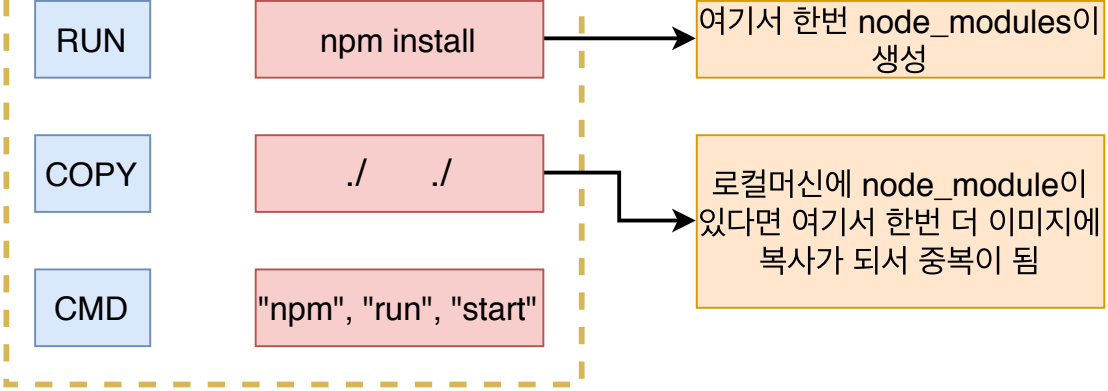
도커를 이용하여 개발단계에서 리액트 실행하기

도커로 어플을 실행하기 위해서는...

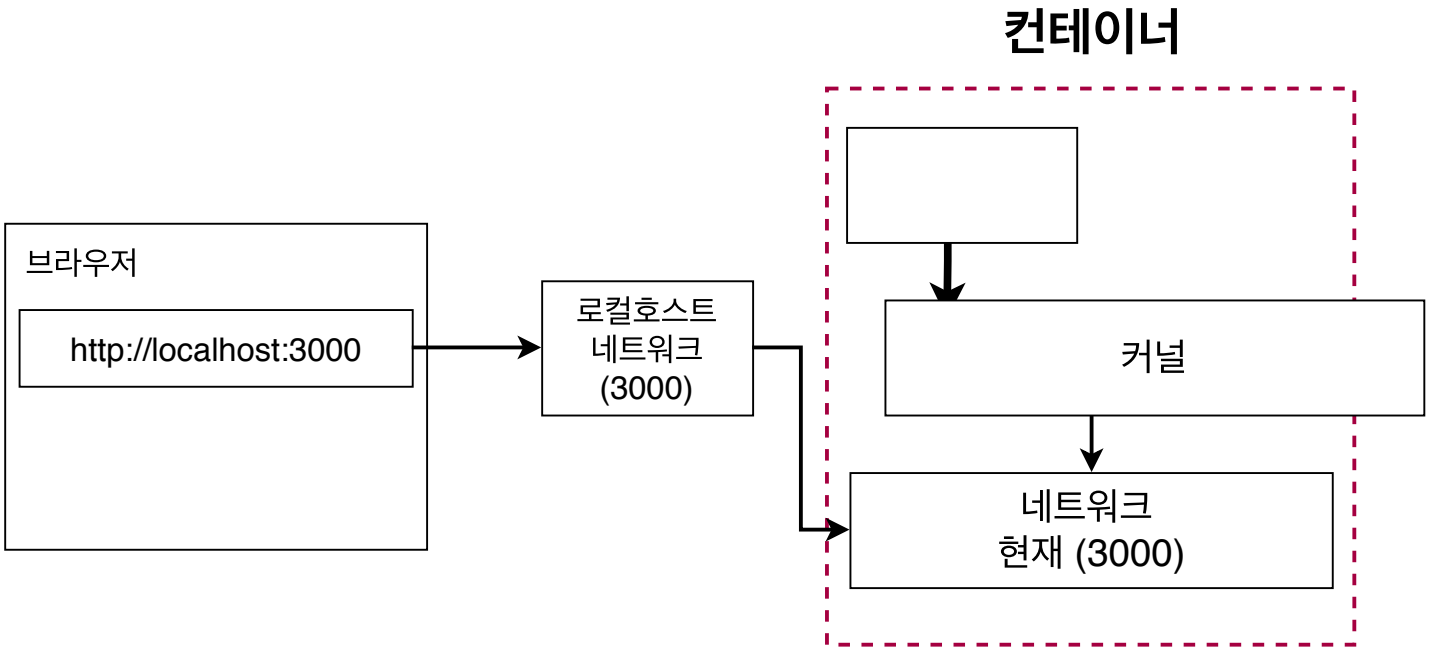


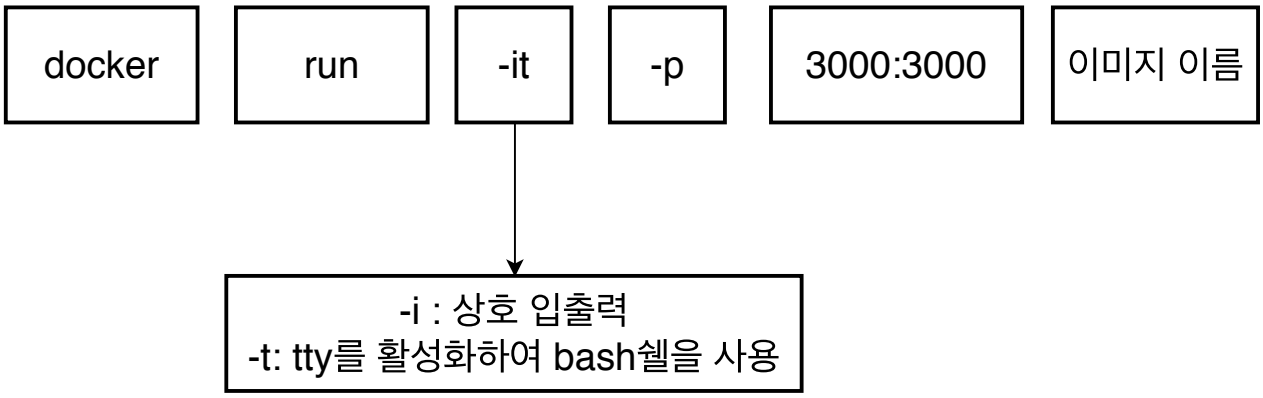





RUN
npm install
여기서 한번 node_modules이 생성
COPY
./ ./
로컬머신에 node_module이 있다면 여기서 한번 더 이미지에 복사가 되서 중복이 됨
CMD
"npm", "run", "start"

생성된 도커 이미지로 리액트 실행해보기



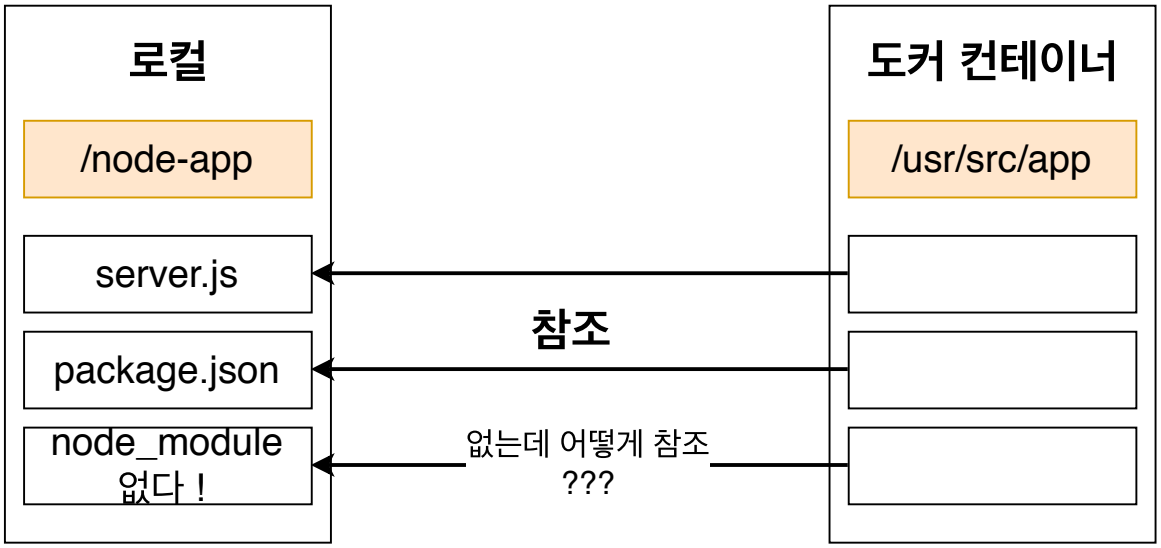


볼륨을 이용한 소스 코드 변경

Volume 사용해서 어플리케이션 실행하는 법

docker run -p 3000:3000	-v /usr/src/app/node_modules	-v $(pwd):/usr/src/app	<이미지 아이디>

Volume



도커 컴포즈로 좀 더 간단하게 앱 실행하기

docker-compose.yml
버전 3

컨테이너 이름
react

1. 도커 파일 사용
2. 포트 맵핑

version	도커 컴포즈의 버전
services	이곳에 실행하려는 컨테이너들을 정의
react	컨테이너 이름
build	현 디렉토리에 있는 Dockerfile 사용
context	도커 이미지를 구성하기 위한 파일과 폴더들이 있는 위치
dockerfile	도커 파일 어떤 것인지 지정
ports	포트 맵핑 로컬 포트 : 컨테이너 포트
volumes	로컬 머신에 있는 파일들 맵핑
stdin_open	리액트 앱을 끌때 필요(버그 수정)

리액트 앱 테스트 하기

이미지 생성

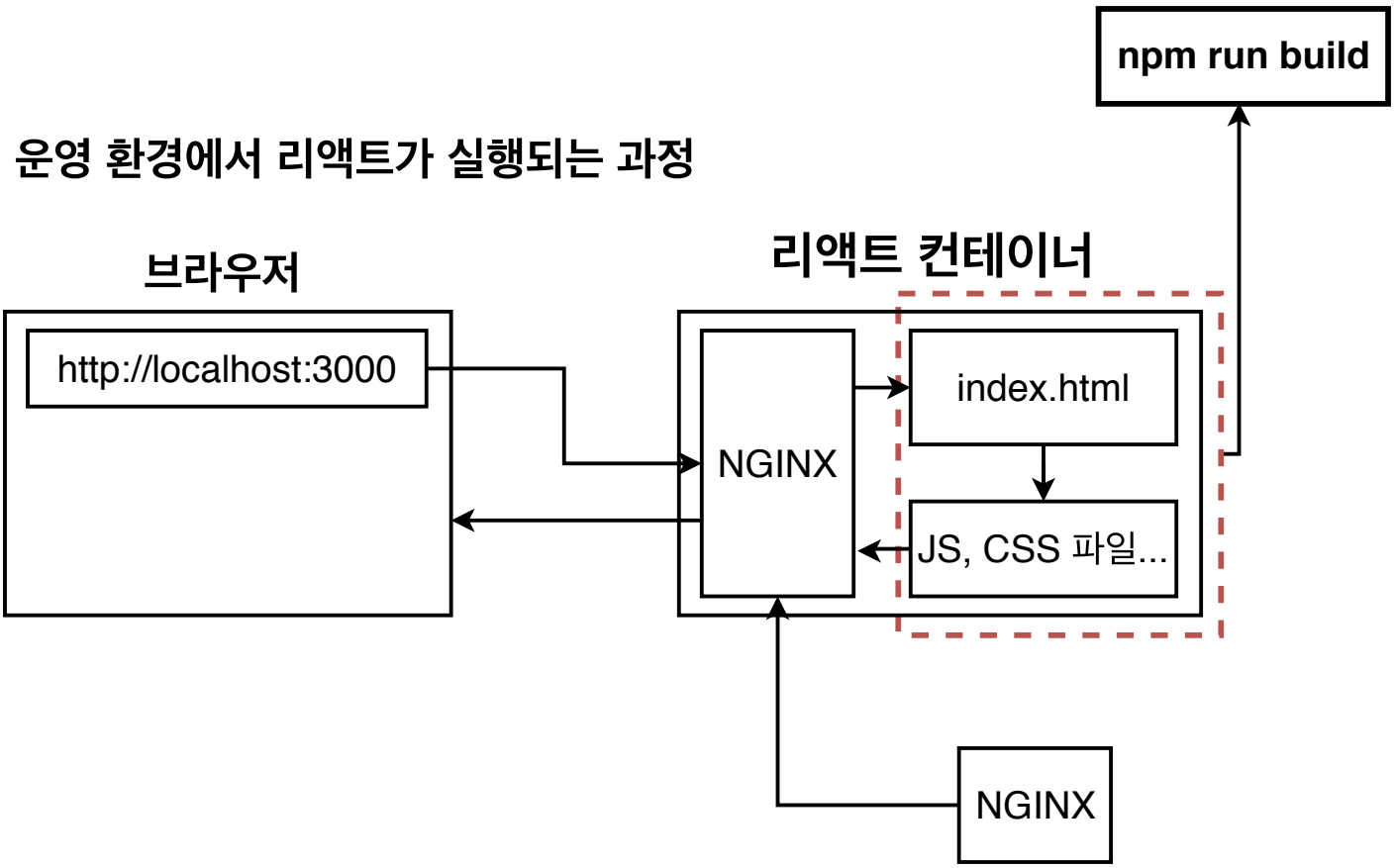
```
docker build -f dockerfile.dev .
```

앱 실행

```
docker run -it 이미지 이름 npm run test
```

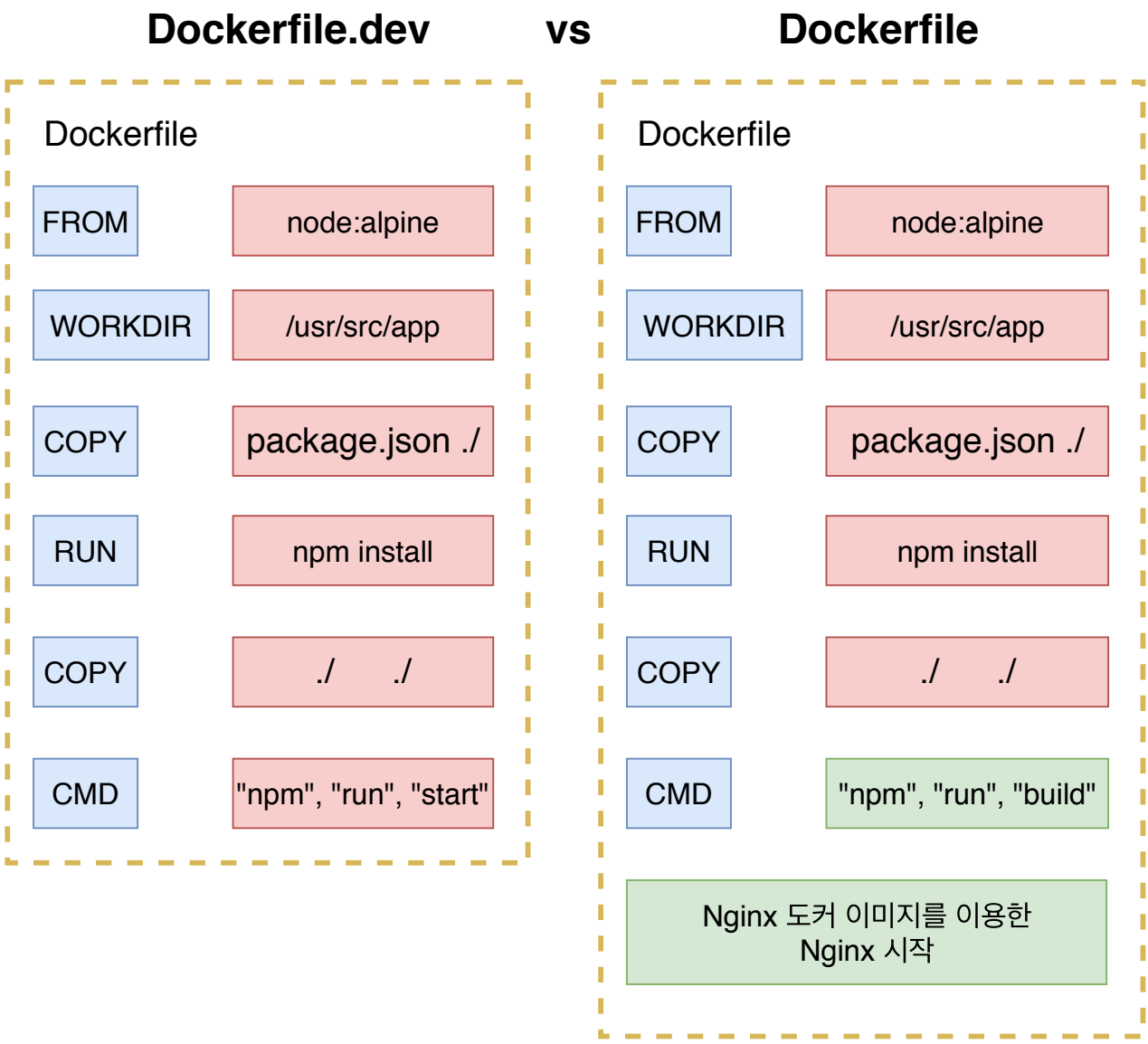
- test → 컨테이너 이름
- build → 현 디렉토리에 있는 Dockerfile 사용
- context → 도커 이미지를 구성하기 위한 파일과 폴더들이 있는 위치
- dockerfile → 도커 파일 어떤 것인지 지정
- volumes → 로컬 머신에 있는 파일들 맵핑
- command → 테스트 컨테이너 시작할때 실행 되는 명령어

운영환경을 위한 Nginx

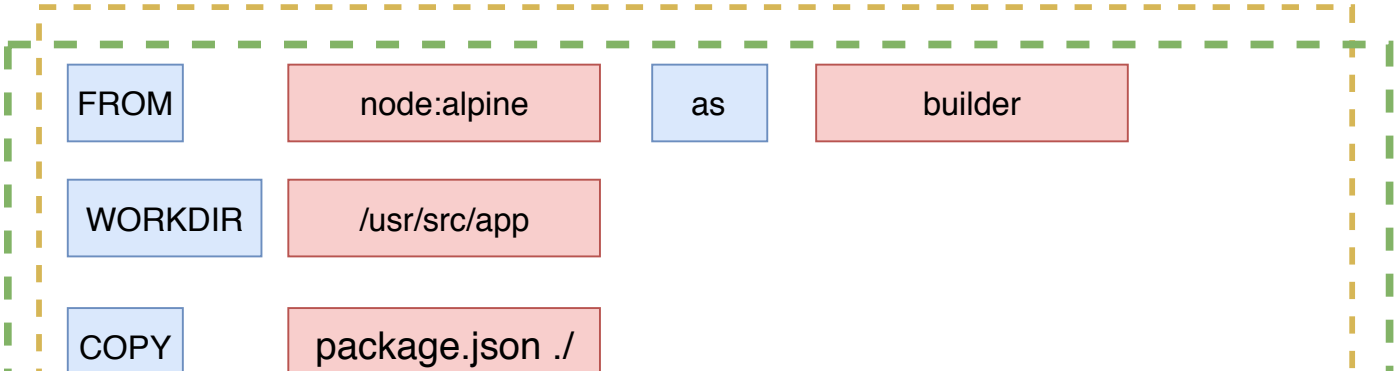


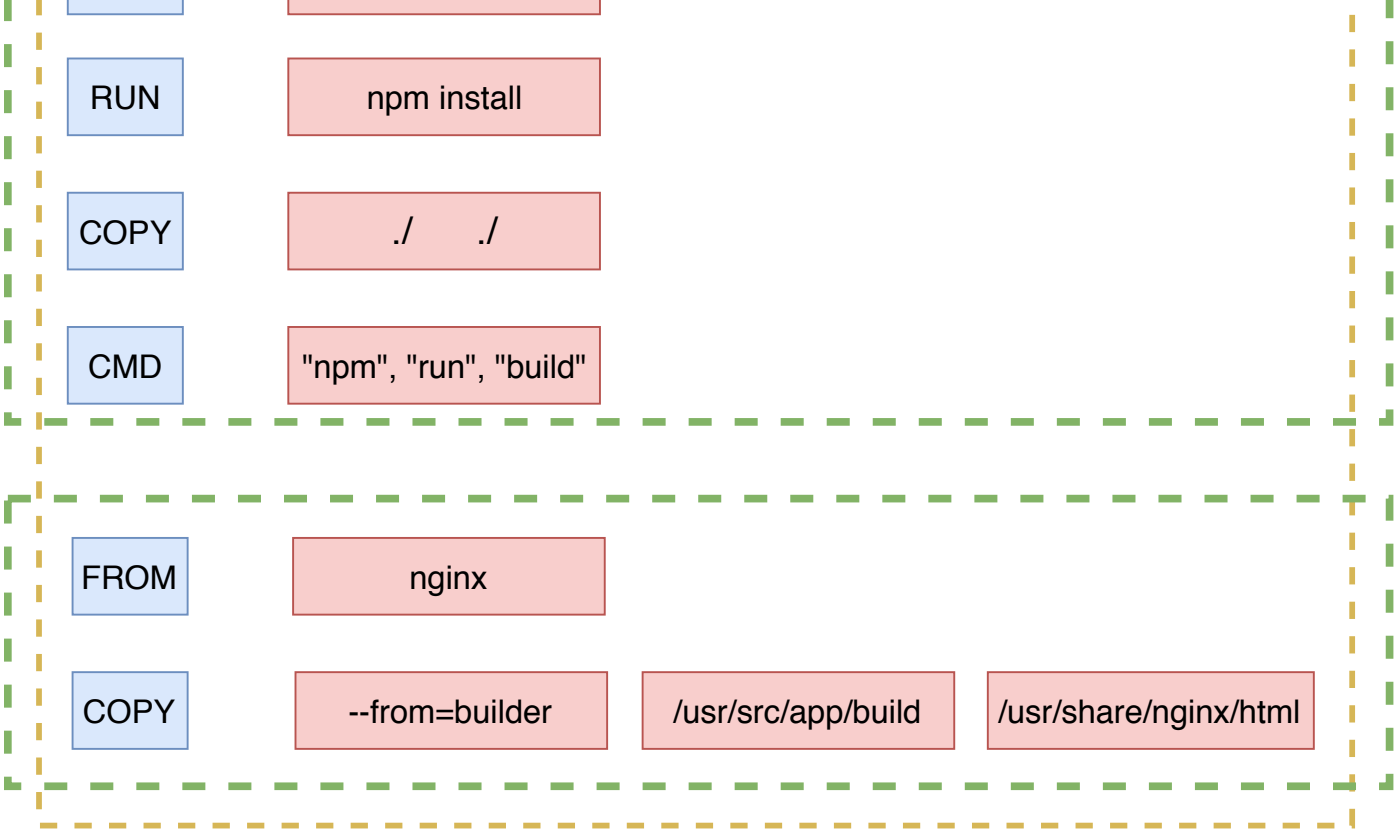
운영환경 도커 이미지 위한 Dockerfile 작성하기

개발환경 도커파일과 운영환경 도커 파일 비교



Dockerfile




RUN
npm install
COPY
./ ./
CMD
"npm", "run", "build"
FROM
nginx
COPY
--from=builder
/usr/src/app/build
/usr/share/nginx/html