

리액트 애플리케이션 빌드, 배포



1. 리액트 애플리케이션 빌드

❖ 빌드 명령어

- `npm run build`
- 빌드 결과물
 - 디렉토리
 - Vite 기반 프로젝트 : `dist`
 - CRA 기반 프로젝트 : `build`
 - 일반적인 산출물
 - `dist/index.html`
 - `dist/assets/index.xxxx.css`
 - `dist/assets/index.xxxx.js`
 - 기타 정적 파일
- 대규모 앱일 경우
 - 모든 코드가 `index.xxxx.js` 파일 하나로 빌드됨
 - 첫 화면 로딩시 LCP 값이 커짐 ---> 사용자 경험 훼손

1. 리액트 애플리케이션 빌드

❖ 빌드 결과물을 성능 최적화로 만드는 방법

- react-router를 이용하는 애플리케이션의 경우 Lazy Loading 적용
 - 4장 8절에서 학습
 - lazy loadin을 위해 각 페이지 컴포넌트를 위한 청크(Chunk)가 분리됨
 - 해당 컴포넌트가 필요한 시점에 .js 파일 로딩
 - lazy loading으로도 해결되지 않는 모듈들
 - node_modules에 컴포넌트, 모듈들
 - vendor가 제공하는 것들
- 개발자가 직접 청크를 분리
 - vendor가 제공하는 모듈들도 직접 분리할 수 있음
 - 선결 조건 : 어떤 모듈을 분리할 것인가를 알아야 함
 - Vite나 CRA에 청크 분리를 위한 내장된 기능을 이용해 분리
 - 예) Vite.config.ts 파일에 분리 기능을 정의함

1. 리액트 애플리케이션 빌드

❖어떤 모듈을 분리할 것인가?

- 모든 모듈을 분리할 수도 있지만 너무 많은 청크가 만들어질 수 있음. 오히려 로딩 성능에 나쁜 영향을 줌
- 분리대상 모듈
 - 첫 화면 로딩할 때 필요하지 않은 모듈
 - 특정 화면에서만 사용되는 사이즈가 큰 모듈
- 어떤 모듈이 분리대상이 될지 결정을 위해 bundle-analyzer를 사용할 것을 권장
 - CRA : webpack-bundle-analyzer
 - Vite : vite-bundle-visualizer, Rollup Plugin Visualizer

1. 리액트 애플리케이션 빌드

❖ vite-bundle-visualizer 사용

- 3가지 타입
 - sunburst : 순환 계층 다이어그램. 사이즈가 큰 모듈을 파악하기 좋음
 - treemap : 직사각형 계층 다이어그램. 사이즈가 큰 모듈을 파악하기 좋음
 - network : 모듈과 모듈의 참조 관계를 파악하기 좋음
- vite-bundle-visualizer 사용법

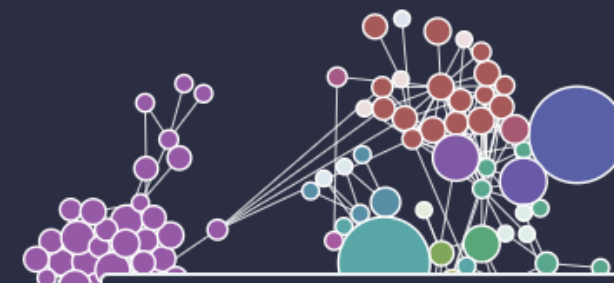
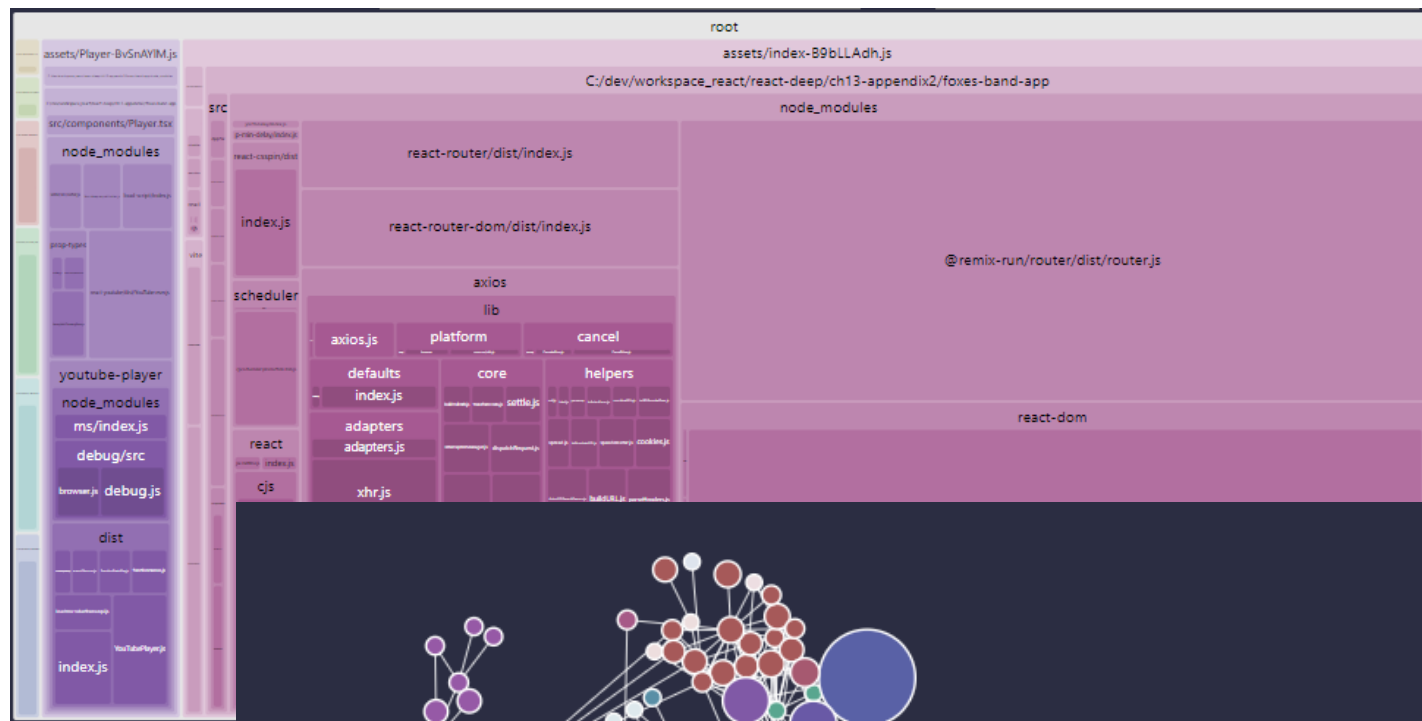
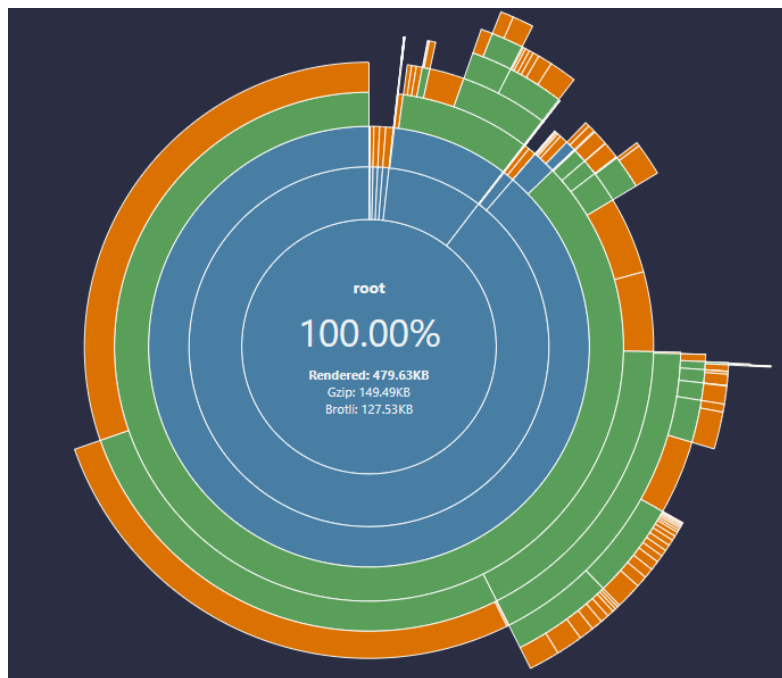
```
//npx vite-bundle-visualizer -t 타입 -o 저장파일
```

```
예시1) npx vite-bundle-visualizer -t sunburst -o stats-sunburst.html
```

```
예시2) npx vite-bundle-visualizer -t network -o stats-network.html
```

1. 리액트 애플리케이션 빌드

■ 다이어그램 형태



C:/dev/workspace_react/react-deep/ch13-appendix2/foxes-band-app/node_modules/axios/index.js
Imported By:
C:/dev/workspace_react/react-deep/ch13-appendix2/foxes-band-app/src/main.tsx
C:/dev/workspace_react/react-deep/ch13-appendix2/foxes-band-app/src/loaders/index.ts
C:/dev/workspace_react/react-deep/ch13-appendix2/foxes-band-app/src/components/ErrorBoundary.tsx
C:/dev/workspace_react/react-deep/ch13-appendix2/foxes-band-app/src/actions/index.ts

1. 리액트 애플리케이션 빌드

❖ Vite 기반 청크 분리 실습

- axios, react-router, react, react-youtube 모듈을 청크 분리
- 준비된 예제는 4장에서 작성했던 foxes-band-app
 - 만일 작성한 예제가 없다면 완성된 예제 4장의 foxes-band-app-8-lazy-loading 예제를 이용함
- vite.config.ts를 다음과 같이 변경하고 npm run build 수행

```
.....(생략)
export default defineConfig({
  plugins: [react()],
  build: {
    rollupOptions: {
      output: {
        manualChunks: {
          "vendor-axios": ["axios"],
          "vendor-react": ["react", "react-dom"],
          "vendor-react-youtube": ["react-youtube"],
          "vendor-react-router": ["react-router-dom"],
        },
      },
    },
  },
});
```

```
vite v5.2.11 building for production...
✓ 136 modules transformed.
dist/index.html                                0.72 kB | gzip: 0.37 kB
dist/assets/index-B6vsFp2-.css                 243.55 kB | gzip: 32.67 kB
dist/assets/Home-BhSwJPr.js                    0.27 kB | gzip: 0.21 kB
dist/assets/About-DRx26AXJ.js                  0.28 kB | gzip: 0.22 kB
dist/assets/Members-C_PDD2nr.js                 0.82 kB | gzip: 0.43 kB
dist/assets/Player-I46YskKB.js                 1.06 kB | gzip: 0.53 kB
dist/assets/SongList-0ikoQNLm.js               1.29 kB | gzip: 0.60 kB
dist/assets/AddSong-G8Bw6qPu.js               1.32 kB | gzip: 0.62 kB
dist/assets/UpdateSong-AeZ7DSrx.js            1.63 kB | gzip: 0.73 kB
dist/assets/index-Dg-UF14_.js                  9.80 kB | gzip: 4.08 kB
dist/assets/vendor-react-youtube-DqjsmdKb.js  18.04 kB | gzip: 6.44 kB
dist/assets/vendor-axios-Cq0ey0Vy.js          29.93 kB | gzip: 12.06 kB
dist/assets/vendor-react-router-CrY5220J.js   65.19 kB | gzip: 21.97 kB
dist/assets/vendor-react-vVDM-wBe.js         141.33 kB | gzip: 45.47 kB
✓ built in 1.66s
```


1. 리액트 애플리케이션 빌드

❖만일 모든 모듈을 분리하고 싶다면?

- manualChunks 옵션에 함수를 지정해 함수가 리턴하는 청크명으로 분리함
 - 함수의 인자 id : 모듈의 전체 경로
 - 모듈의 경로에 node_modules가 포함되어 있으면 모두 분리시킴
- vite.config.ts를 다음과 같이 변경하고 npm run build 수행

```
.....(생략)
export default defineConfig({
  plugins: [react()],
  build: {
    rollupOptions: {
      output: {
        manualChunks: (id) => {
          if (id.indexOf("node_modules") !== -1) {
            const module = id.split("node_modules/").pop().split("/")[0];
            return `vendor-${module}`;
          }
        },
      },
    },
  },
});
```

```
vite v5.2.11 building for production...
✓ 136 modules transformed.
dist/index.html                                1.68 kB  gzip: 0.52 kB
dist/assets/index-DWvpyoBr.css                 0.79 kB  gzip: 0.44 kB
dist/assets/vendor-react-cssspin-BhN80_AN.css  9.37 kB  gzip: 1.48 kB
dist/assets/vendor-bootstrap-DRQo3kaA.css     233.39 kB  gzip: 30.83 kB
dist/assets/vendor-yoctodelay-DB-hd04a.js      0.12 kB  gzip: 0.13 kB
dist/assets/vendor-p-min-delay-D0BEVQ8s.js     0.17 kB  gzip: 0.16 kB
dist/assets/Home-DPHyvcdW.js                   0.23 kB  gzip: 0.19 kB
dist/assets/About-Bul043Fg.js                  0.24 kB  gzip: 0.20 kB
dist/assets/vendor-sister-DVqyZZES.js          0.46 kB  gzip: 0.31 kB
dist/assets/vendor-fast-deep-equal-BvdMVOJB.js  0.77 kB  gzip: 0.38 kB
dist/assets/vendor-load-script-DghppdYS.js     0.78 kB  gzip: 0.42 kB
dist/assets/vendor-prop-types-CkTJOC_5.js      0.80 kB  gzip: 0.51 kB
dist/assets/Members-BKKp-kl0.js                0.90 kB  gzip: 0.46 kB
dist/assets/vendor-ms-nT7xfLs5.js              1.23 kB  gzip: 0.62 kB
dist/assets/Player-CALfDyvm.js                 1.48 kB  gzip: 0.69 kB
dist/assets/AddSong-DHk1_eNB.js                1.49 kB  gzip: 0.69 kB
dist/assets/SongList-0IV0akHX.js               1.51 kB  gzip: 0.68 kB
dist/assets/UpdateSong-CNwDozhp.js             1.84 kB  gzip: 0.80 kB
dist/assets/vendor-react-cssspin-qALiy_rl.js   2.88 kB  gzip: 1.54 kB
dist/assets/vendor-debug-bhRFUqb8.js           3.26 kB  gzip: 1.52 kB
dist/assets/vendor-scheduler-CzFDRTuY.js       4.10 kB  gzip: 1.78 kB
dist/assets/vendor-youtube-player-YrpmPaAi.js  5.40 kB  gzip: 1.98 kB
dist/assets/vendor-react-youtube-ecetz5LP.js   5.92 kB  gzip: 1.90 kB
dist/assets/index-DqBqJLzd.js                  6.81 kB  gzip: 2.74 kB
dist/assets/vendor-react-XGzZVkvQ.js           8.06 kB  gzip: 3.06 kB
dist/assets/vendor-react-router-BLjj0TZ0.js    9.89 kB  gzip: 3.52 kB
dist/assets/vendor-react-router-dom-SJjussz2.js 10.41 kB  gzip: 4.05 kB
dist/assets/vendor-axios-Cq0ey0Vy.js          29.93 kB  gzip: 12.06 kB
dist/assets/vendor-@remix-run-13oXxjU0.js     45.31 kB  gzip: 15.94 kB
dist/assets/vendor-react-dom-Do0ZKzj6.js      130.29 kB  gzip: 41.87 kB
✓ built in 1.67s
```

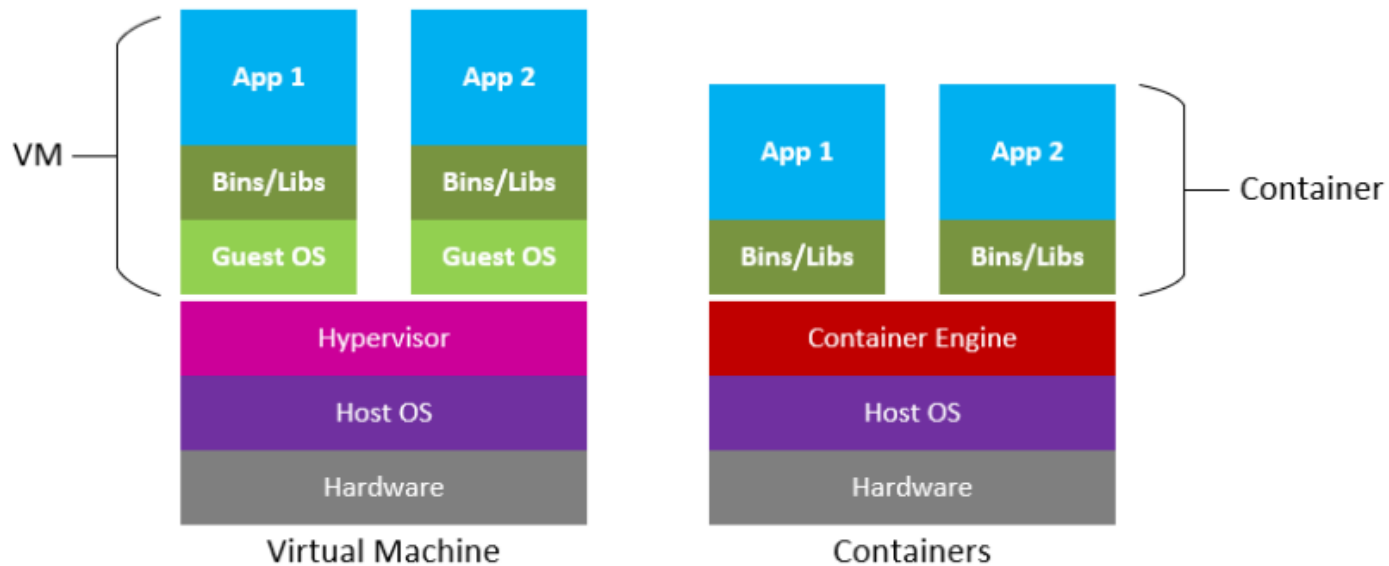

2. 컨테이너 개요

❖ 컨테이너란?

- 실행하기 위해 필요한 런타임, 애플리케이션 코드, 구성 설정, 의존 모듈 등을 포함하는 표준화되고 규격화된 소프트웨어 패키지
- 컨테이너를 실행하기 위한 엔진만 준비되어 있다면 어떤 환경, 어떤 플랫폼에서든지 실행할 수 있음

❖ 대표적인 컨테이너 기술

- Docker : Go언어로 작성된 가장 널리쓰이는 컨테이너 엔진
- OPENVZ, LXC, LXD



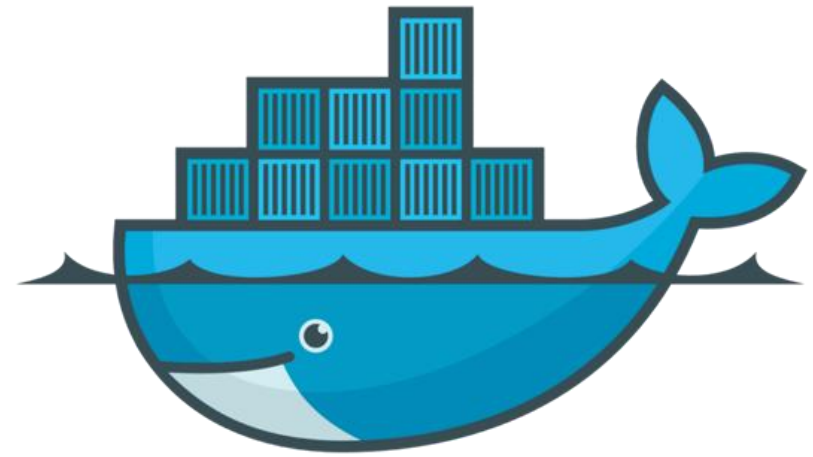
2. 컨테이너 개요

❖ Docker란?

- 컨테이너를 빌드, 배포, 실행, 관리할 수 있는 기능을 제공하는 오픈소스 플랫폼, 엔진

❖ Docker의 장점

- 표준화, 규격화
 - 이식성, 이동성 극대화
 - 어디서나 Docker 엔진만 있다면 실행 가능
 - 지속적 배포를 적용하기에 용이함
 - Dev환경 = Test 환경 = Prod 환경
- 애플리케이션 기동 시간 감소 (VM과 비교하여)
 - 미리 설정된 소프트웨어 패키지이므로 ...
- 아키텍처의 변경 없이 빠르게 확장
 - 컨테이너 수만 늘리면 확장
 - 컨테이너 실행 시간이 VM의 실행시간보다 짧음



docker

3. 컨테이너 사용을 위한 설정

❖가장 간단한 배포 방법

- npm run build 후 dist 디렉토리의 파일을 웹서버에 복사

❖컨테이너 기반 애플리케이션에 배포하려면?

- 최근 많이 배포되는 형태 : ex) k8s, docker swarm
- 수동 또는 CI 도구를 이용해 자동화된 테스트, 빌드 프로세스를 통해 컨테이너 이미지를 생성해야 함

❖이 과정에서 사용할 이미지 빌드 도구, 배포 플랫폼

- docker desktop
- docker desktop이 제공하는 싱글 인스턴스의 k8s

3. 컨테이너 사용을 위한 설정

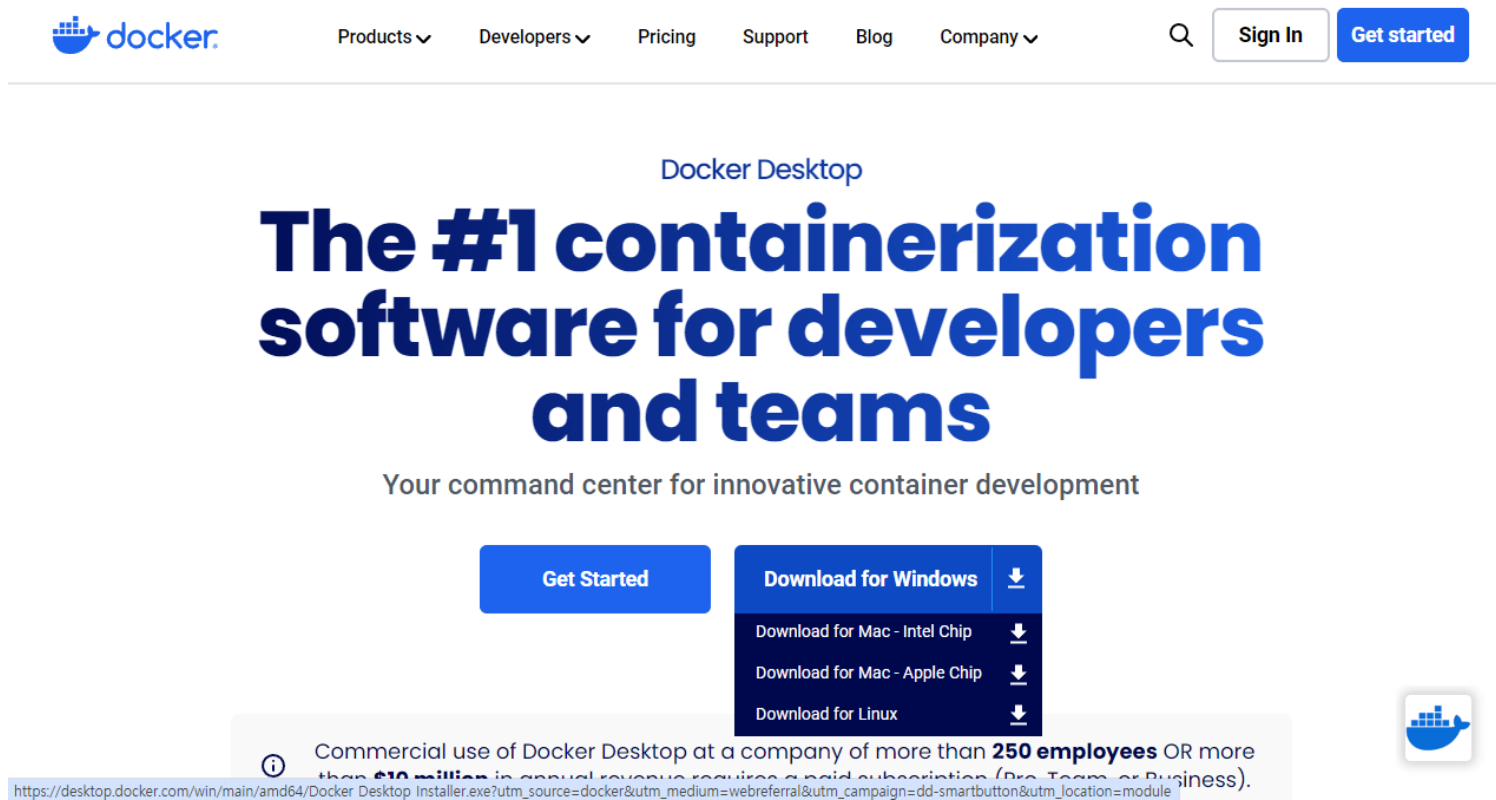
❖ Docker desktop 설치

- 운영체제별 설치 가능 여부 및 사전 설치 사항
 - mac, linux : 설치 가능
 - window 10,11 Pro : 설치 가능
 - window 10 Home : 20H1, 20H2, 21H1 이상의 버전에서만 가능. 사전에 wsl2가 설치되어야 함
 - window 11 Home : 설치 가능하지만 사전에 wsl2가 설치되어야 함
- window 10,11 Home에 사전 설치해야 할 항목 : wsl2 (Windows Subsystem Linux 2)
 - windows 10의 Powershell을 관리자 권한으로 실행
 - `dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart`
 - `dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart`
 - `wsl --install` : wsl 환경 설치
 - `wsl --install -d Ubuntu` : Ubuntu 리눅스 설치
 - 다음 경로에서 WSL2를 다운로드 받아 리눅스 커널 업데이트
 - https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi
 - `wsl --set-default-version 2` : 기본 버전을 wsl2로 변경
 - Ubuntu를 실행하여 아이디, 패스워드 설정

3. 컨테이너 사용을 위한 설정

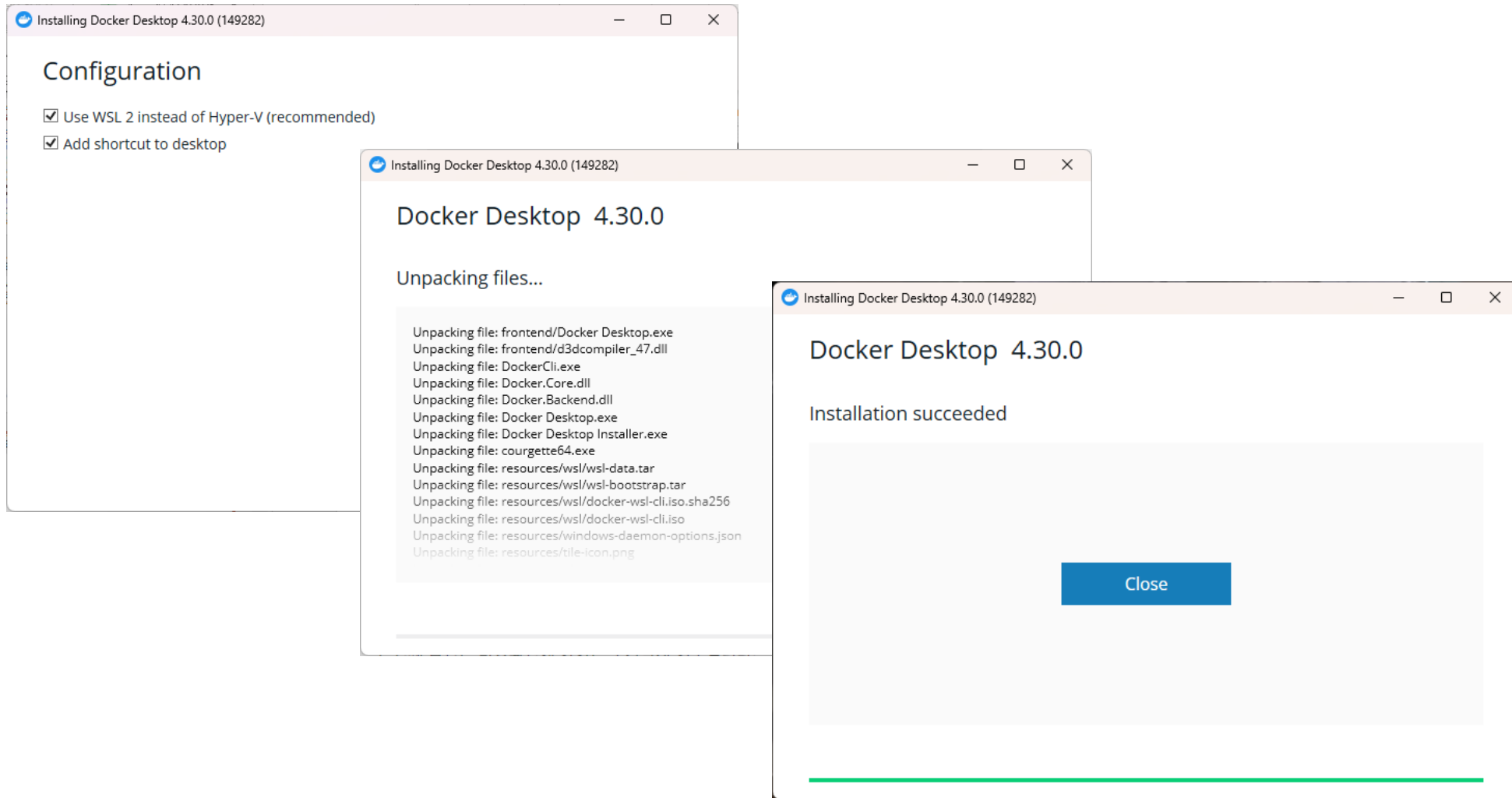
❖ Docker Desktop 설치

- 설치파일 다운로드
 - <https://docker.com/products/docker-desktop>



3. 컨테이너 사용을 위한 설정

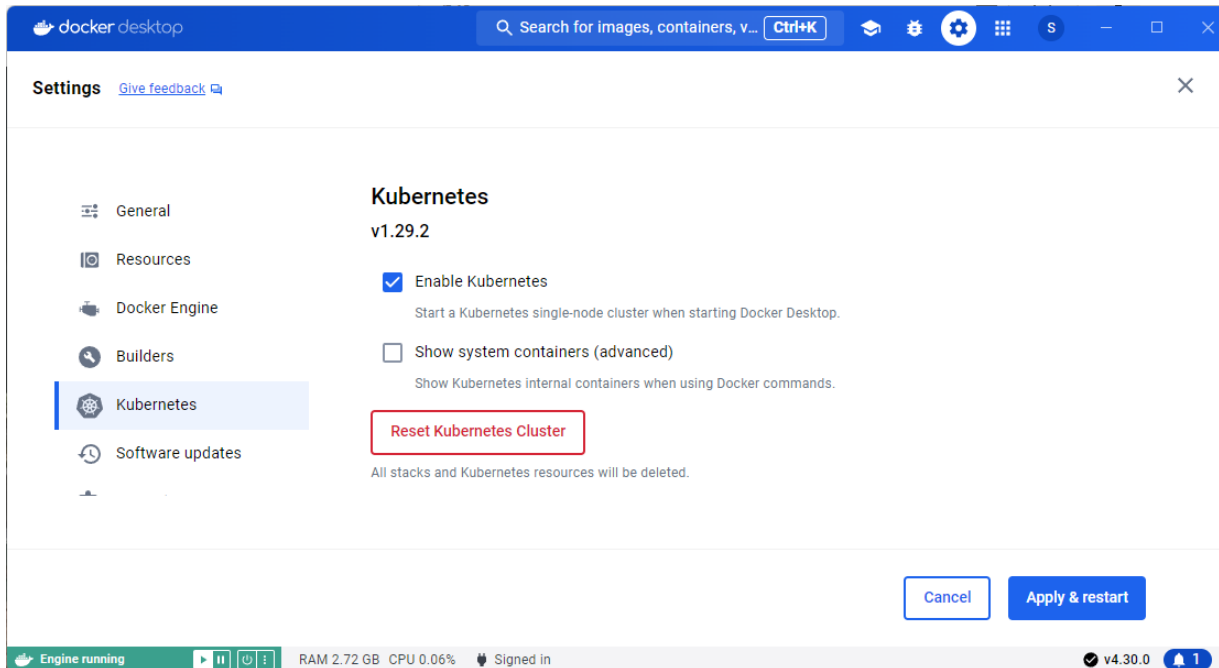
❖ Docker Desktop 설치



3. 컨테이너 사용을 위한 설정

❖ Docker Desktop 환경 설정

- Docker Desktop이 실행된 후 오른쪽 상단의 톱니바퀴 아이콘을 클릭하여 설정화면으로 이동
- 왼쪽 패널에서 Kubernetes 클릭 후
 - Enable Kubernetes 체크후 Apply & Restart 버튼 클릭
 - k8s 서비스가 시작되지 않고 계속 시작중이면 컴퓨터를 재부팅할 것



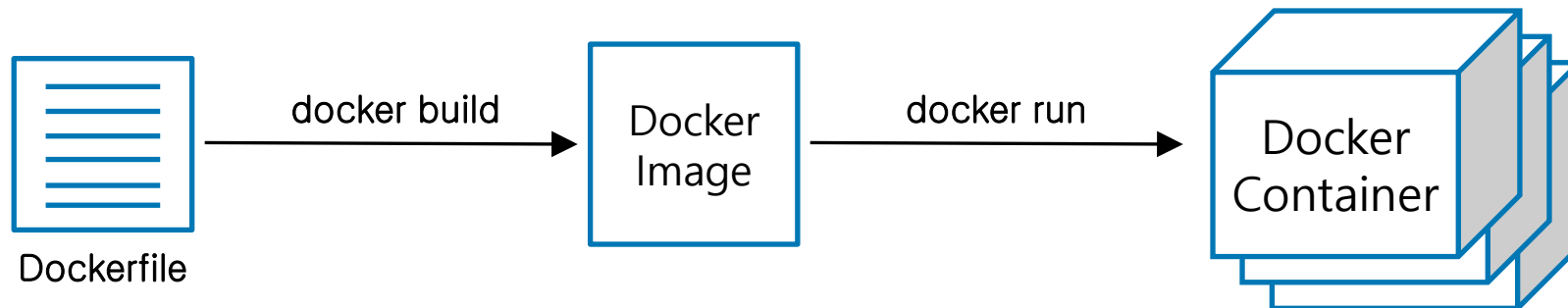
4. 컨테이너 이미지 빌드

❖ Container Image란?

- Container를 생성하기 위한 청사진이자 일종의 템플릿
 - Image 하나로 여러 개의 Container를 생성할 수 있음
- Image를 빌드하기 위해서는 Dockerfile을 작성하고 docker build 명령으로 빌드해야 함

❖ Image VS Container

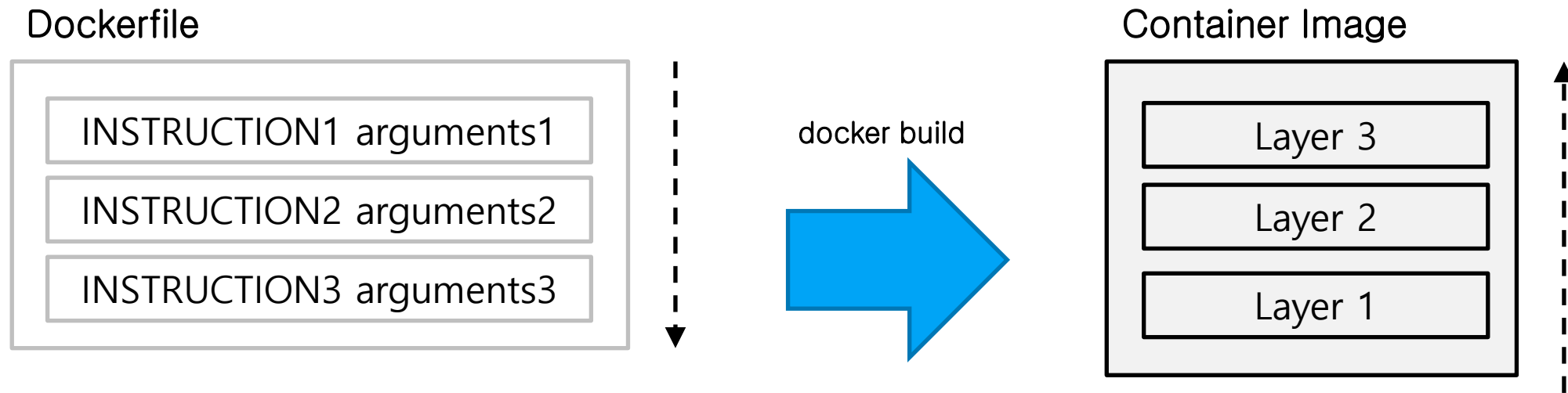
- 1 Image --> N Container
- 특성 비교
 - Image는 변경불가(Immutable)
 - Container는 Image Layer위에 쓰기가 가능한 Container Layer가 추가됨
 - Container를 종료, 재시작하면 Container Layer에 썼던 콘텐츠는 모두 날라감(휘발성)



4. 컨테이너 이미지 빌드

❖ Dockerfile

- Docker Image를 빌드하기 위한 Instruction의 집합
- Dockerfile 내부는 Instruction(명령, 지시어)으로 구성됨
- Image는 Instruction의 역순으로 계층화됨



4. 컨테이너 이미지 빌드

❖ foxes-band-app2 리액트 앱을 배포를 위한 Dockerfile 예시

- foxes-band-app2 예제 : 반드시 다단계 빌드,
 - 백엔드 API를 사용하지 않는 예제이므로 단독 배포 후 실행 확인 가능

```
## stage1 : 관련된 모듈 다운로드 후 빌드
FROM node:20 AS BUILD
WORKDIR /app-build
COPY . /app-build
RUN npm install
RUN npm run build

## stage2 : stage1에서 빌드한 결과물을 이용해 이미지 빌드
FROM nginx:stable-alpine
# nginx 설정 중에서 404 오류시에 /index.html을 응답하도록 fallback UI 설정한 것으로 교체
COPY --from=BUILD /app-build/nginx-conf/default.conf /etc/nginx/conf.d/
# npm run build로 만들어진 dist 폴더의 빌드 결과물을 nginx 루트로 복사
COPY --from=BUILD /app-build/dist /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

4. 컨테이너 이미지 빌드

❖ 이미지 빌드

- foxes-band-app 폴더에서 다음 명령어 실행

```
docker build . -t foxes-band-app:1.0.0
```

- 만일 docker hub 등과 같은 이미지 리포지토리에 푸시하고 싶다면?

```
docker tag foxes-band-app:1.0.0 [dockerhub-id]/foxes-band-app:1.0.0  
docker push [dockerhub-id]/foxes-band-app:1.0.0
```

- 테스트용 컨테이너 실행

```
docker run -d -p 8080:80 --name foxes foxes-band-app:1.0.0
```

//브라우저를 열어서 http://localhost:8080 으로 실행 여부 확인

//테스트가 끝났다면 컨테이너 실행 종료후 삭제

```
docker stop foxes
```

```
docker rm foxes
```

4. 컨테이너 이미지 빌드

❖ k8s 애플리케이션 실행

- deployment로 배포 : 예제 제공 - foxes-band-app-deploy.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: foxes-band-app-deploy
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: foxes-band-app
  template:
    metadata:
      name: foxes-band-app
      labels:
        app: foxes-band-app
    spec:
      containers:
        - name: foxes-band-app
          image: stepanow/foxes-band-app:1.0.0
          ports:
            - containerPort: 80
```

4. 컨테이너 이미지 빌드

❖k8s에 배포

```
kubectl apply -f foxes-band-app-deploy.yaml
```

```
//실행을 테스트하기 위해 nginx pod 배포
```

```
kubectl run test-pod --image nginx --port=80
```

```
//다음명령어로 배포한 앱의 내부 IP 확인
```

```
kubectl get pods -o wide
```

```
-----
NAME                                READY  STATUS   RESTARTS  AGE    IP             ...
foxes-band-app-deploy-dcc8fb7d6-fl29k  1/1    Running  0         6m17s  10.1.0.15     ...
test-pod                               1/1    Running  0         69s    10.1.0.17     ...
```

```
//test-pod 내부로 접속하여 배포된 앱 pod의 IP로 접속 시도
```

```
kubectl exec -it test-pod -- /bin/bash
```

```
root@test-pod:/# curl 10.1.0.15/asdf
```

```
<!doctype html>
<html lang="en">
```

```
.....
</html>
```

4. 컨테이너 이미지 빌드

❖ 배포한 k8s 애플리케이션 삭제

```
kubectl delete pods test-pod  
kubectl delete deploy foxes-band-app-deploy
```


5. github + 무료 배포 플랫폼

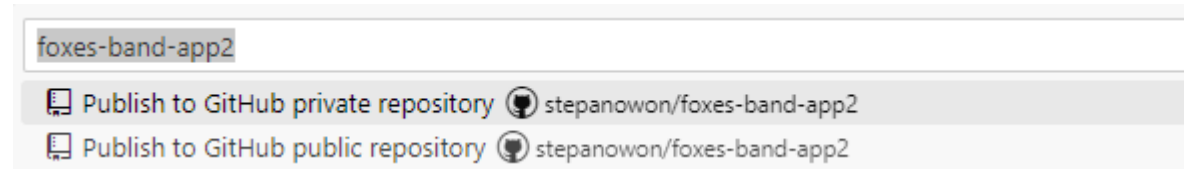
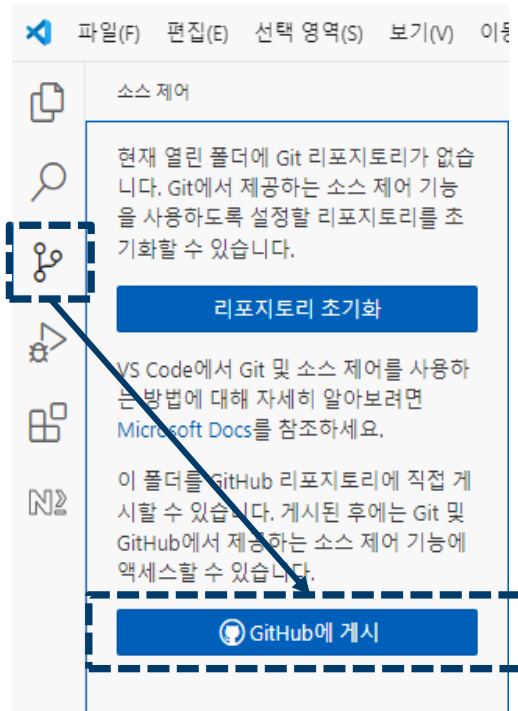
❖사전 준비

- github 계정 등록 : <https://github.com/> 에서 sign up
- git 클라이언트 도구 설치
 - mac, linux : 이미 설치되어 있음
 - window : <https://git-scm.com/downloads> 에서 다운로드 후 기본값으로 설치
- git 도구 설정
 - 사용자명과 이메일 주소 설정
 - `git config --global user.name "사용자명"`
 - `git config --global user.email "사용자 이메일 주소"`
 - git 설정 에디터를 VSCode로 설정
 - `git config --global core.editor "code --wait"`
 - 기본 브랜치를 main으로 설정
 - `git config --global init.defaultBranch main`

5. github + 무료 배포 플랫폼

❖github에 새로운 리포지토리 생성

- foxes-band-app2 이라는 이름의 Private 리포지토리 생성
- vscode에서 github에 게시 버튼 클릭하여 인증
- 브라우저 창에서 인증 후 게시할 리포지토리 선택 - Private repository 선택
 - github에 게시된 애플리케이션 코드 확인



5. github + 무료 배포 플랫폼

❖vercel이란?


- <https://vercel.com>


Vercel is the Frontend Cloud. Build, scale, and secure a faster, personalized web.

❖vercel 에 접속하여 계정 생성

- github 계정을 이용해 vercel에 로그인
- 대시보드 화면에서 AddNew 버튼 클릭후 Project 선택
- Git 리포지토리에서 foxes-band-app2 리포지토리 선택후 import 함

Import Git Repository

 stepanowon

 foxes-band-app2 · 5m ago

Import

Missing Git repository? [Adjust GitHub App Permissions →](#)

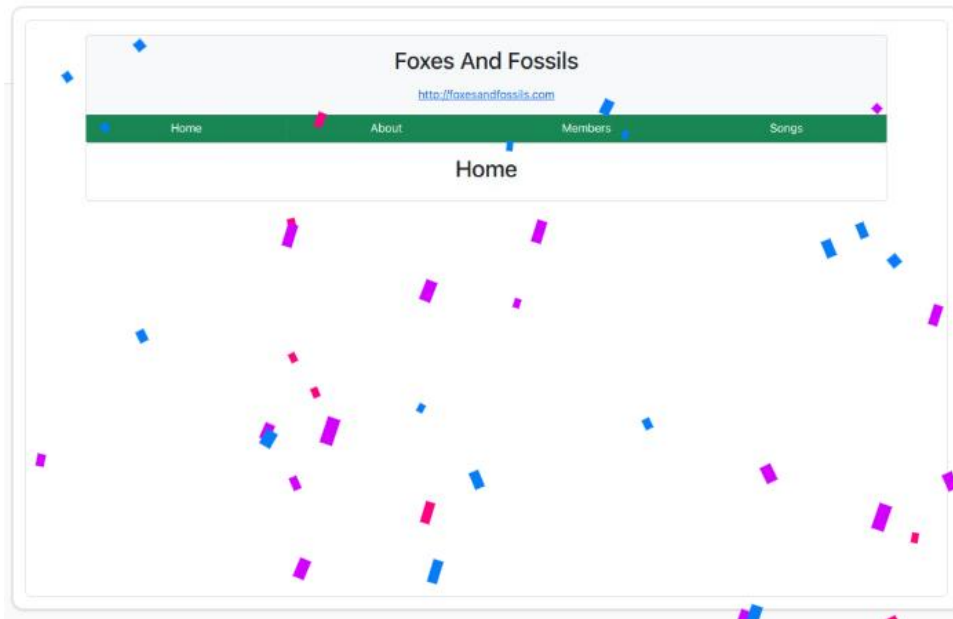
5. github + 무료 배포 플랫폼

- ❖ Vercel Project 설정 화면에서 기본값으로 "Deploy"
- ❖ 배포 완료 화면에서 continue to dashboard 클릭 후 주어진 Domain 을 클릭하여 기능 테스트

Congratulations!

You just deployed a new Project to Vercel.

Continue to Dashboard →



Next Steps

Instant Previews

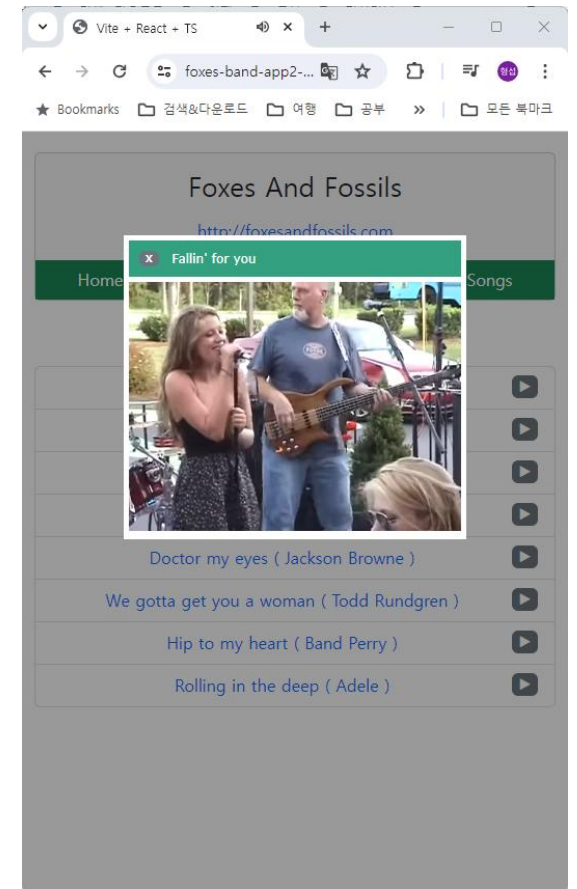
Push a new branch to preview changes instantly

Add Domain

Add a custom domain to your project

Enable Speed Insights

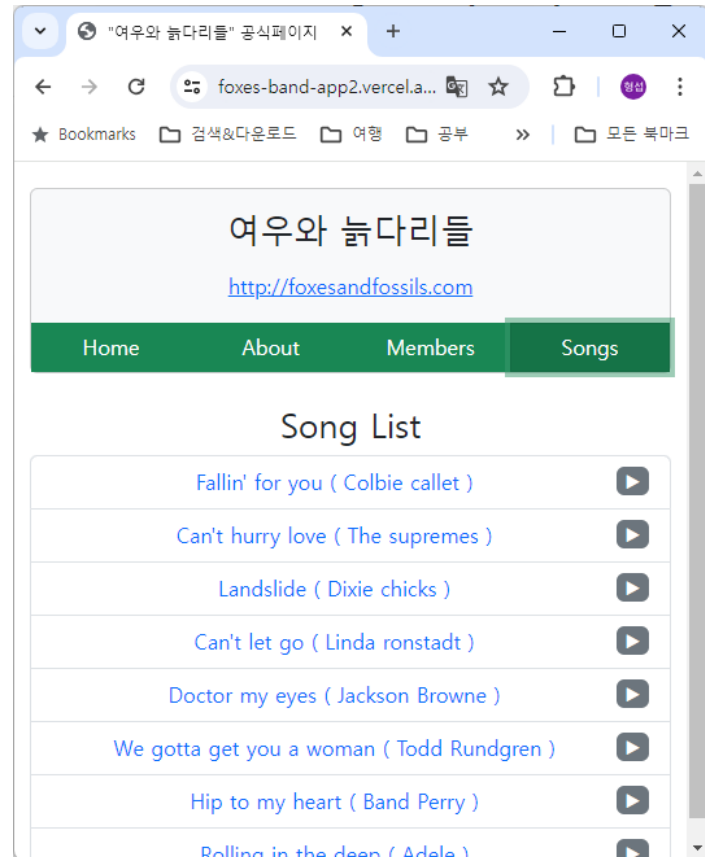
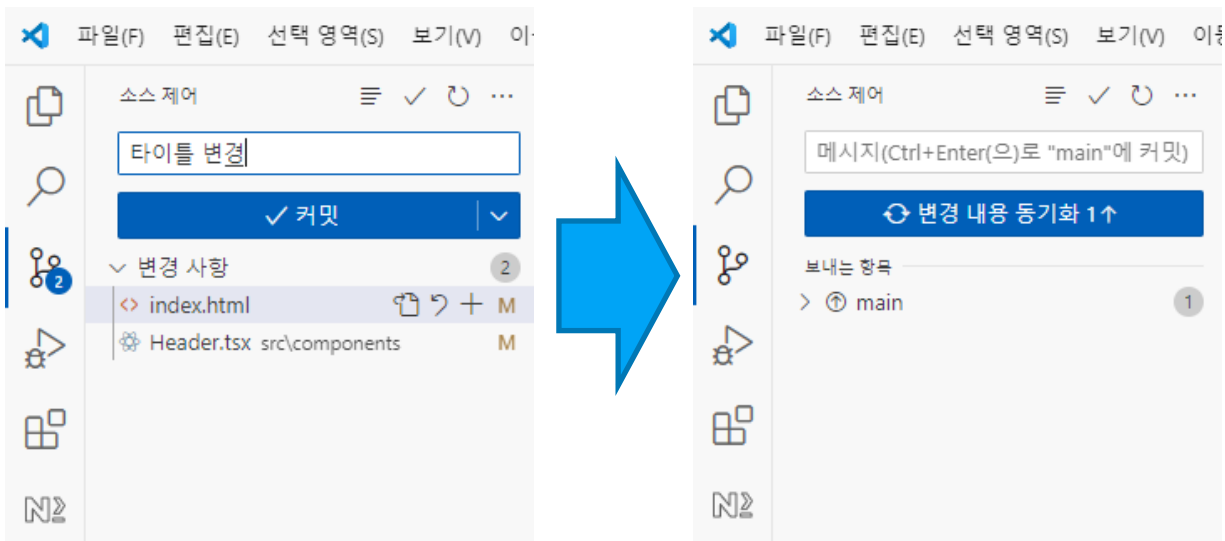
Track how users experience your site over time



5. github + 무료 배포 플랫폼

❖ 애플리케이션 코드를 변경 후 재배포 하기

- 변경 사항
 - foxes-band-app2에서 index.html의 title을 변경함
 - "여우와 늑다리들" 공식 페이지
 - src/components/Header.tsx 의 제목을 변경
 - "Foxes And Fossils" --> "여우와 늑다리들"
- VSCode 소스제어 화면에서 적절한 커밋메시지와 함께 변경사항을 커밋
- Vercel에서 자동 배포된 애플리케이션 확인



6. 배포시 고려사항

❖ 애플리케이션의 여러가지 설정 정보는 어떻게 할 것인가?

- 일반적으로 환경변수를 활용함
- 개발 환경에서의 환경 변수 설정
 - 개발시에 사용할 환경 변수는 .env 파일 이용
 - 이 파일은 git 리포지토리로 등록되면 안됨 --> .gitignore 파일에 .env 파일 등록
 - .env 파일에는 key = value 형식으로 설정함
- 리액트 코드에서 환경 변수 이용
 - process.env.ENV1 과 같이 환경 변수에 접근
 - 환경변수가 없을 때를 위해 기본값을 설정하도록 함

```
const BASEURL = process.env.BASEURL || "http://localhost:3000/todolist_long"
```

- 배포 환경에서의 환경 변수
 - vercel과 같은 배포 플랫폼에는 환경 변수를 설정할 수 있는 옵션이 있음
 - Vercel의 경우 Project Settings에 Environment Variables 라는 메뉴가 있음
 - k8s 환경 : 다음 페이지 예시
 - k8s의 ConfigMap, Secret을 이용해 환경 변수를 등록하고 k8s 애플리케이션에서 참조하도록 함.

6. 배포시 고려사항

❖SPA 배포시 404 오류 문제

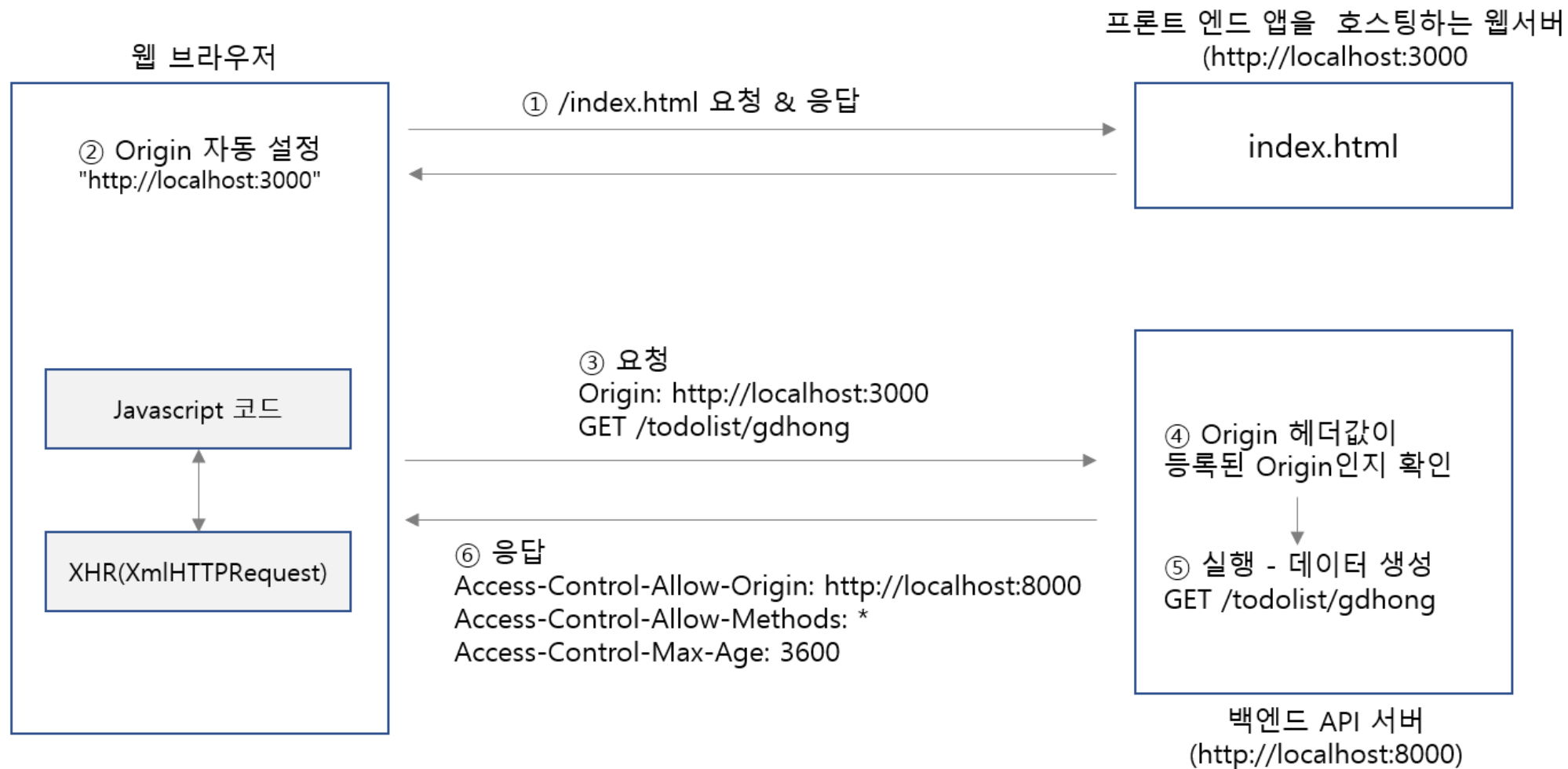
- 서버에서 404 오류시 기본 페이지(index.html)를 응답하도록 설정해야 하고
- 리액트 앱 내부에서 react-router 를 이용해 404 페이지를 별도로 구성해야 함
 - 4장 react-router에서 다룬 내용

❖리액트앱이 별도의 백엔드 API 서버를 접근할 때

- Cross Origin 문제가 발생할 수 있음
- 문제 해결 방법
 - 1. 백엔드 API 서버에서 CORS를 지원하여 해결
 - 2. 리액트 앱을 호스팅하는 웹서버에 proxy 설정하여 해결

6. 배포시 고려사항

■ 1. 백엔드 API 서버에 CORS 지원 설정



6. 배포시 고려사항

- 각 백엔드 API 프레임워크 별 CORS 설정
 - Spring MVC, Spring Boot : @CrossOrigin Annotation

```
@CrossOrigin(origins={ "http://client:5173" })  
@GetMapping  
public ContactList getContactList() {  
    return contactService.getContactList();  
}
```

- Spring Boot : Cors Filter 지정

```
@Configuration  
public class WebMvcConfig implements WebMvcConfigurer {  
    @Override  
    public void addCorsMappings(CorsRegistry registry) {  
        registry.addMapping("/contacts/**")  
            .allowedOrigins("http://client:5173","http://client.com")  
            .allowedMethods("GET","POST","PUT","DELETE","HEAD", "OPTIONS")  
            .allowCredentials(true)  
            .maxAge(3600);  
    }  
}
```

6. 배포시 고려사항

- 각 백엔드 API 프레임워크 별 CORS 설정(이어서)
 - node.js + express

```
import express from 'express';
import cors from 'cors';

.....
const app = express();
app.use(cors());
.....
```

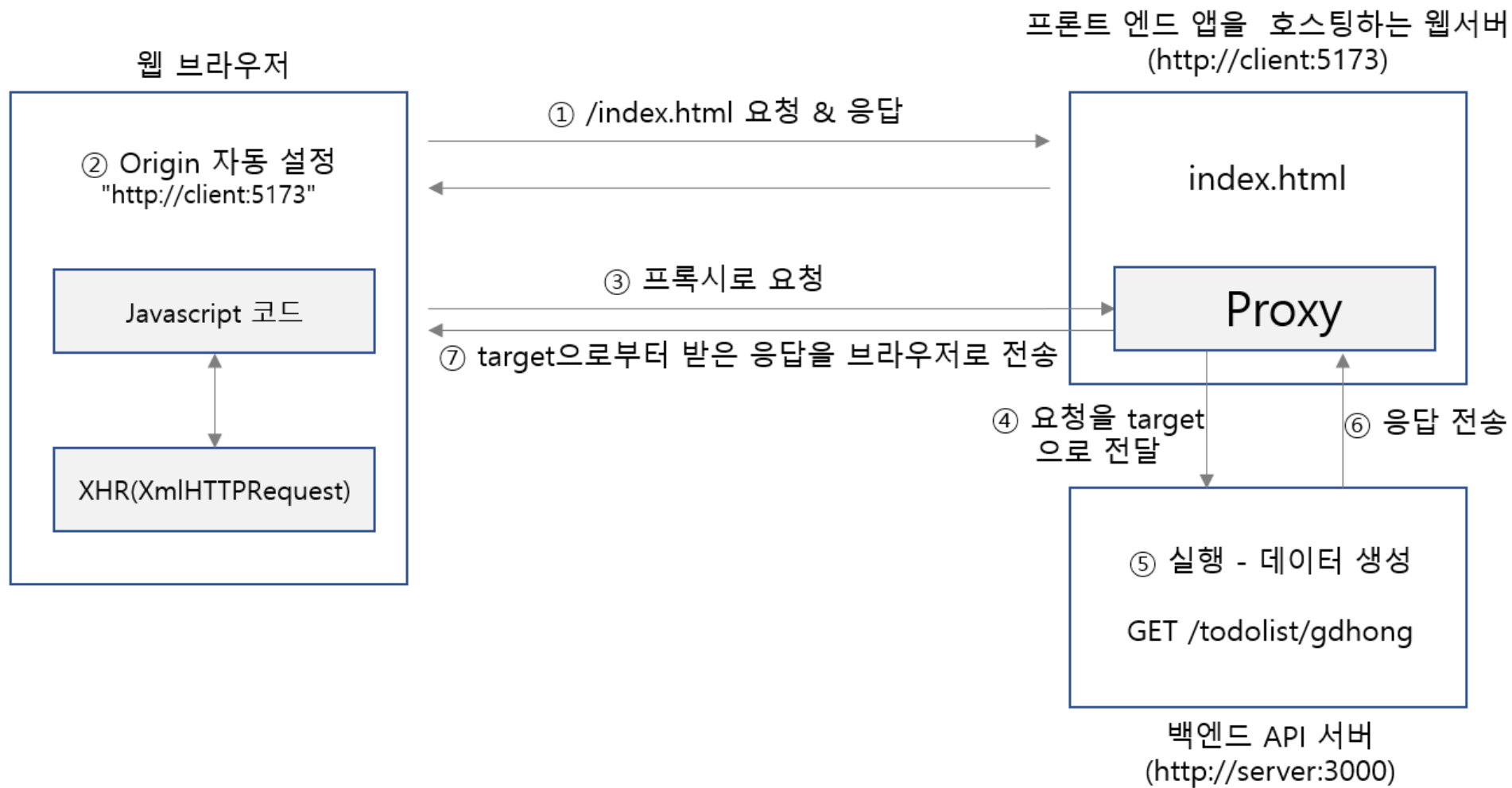
```
import express from 'express';
import cors from 'cors';

.....
const app = express();
const whitelist = ["http://client:5173", "http://client.com"]
const corsOptions = {
  origin: function (origin, callback) {
    if (whitelist.indexOf(origin) !== -1) { callback(null, true) }
    else { callback(new Error('CORS에 의해 접근이 허가되지 않았음')); }
  }
}

app.get('/contacts/:id', cors(corsOptions), (req, res, next) => {
  res.json({ msg: "이 엔드포인트는 화이트 리스트에 등록된 오리진에 매칭될때 접근이 가능합니다."})
})
```

6. 배포시 고려사항

■ 2. 리액트 앱을 호스팅하는 서버에 Proxy 설정



6. 배포시 고려사항

- Vite 개발용 서버에 Proxy를 설정하는 방법

```
// /api/contacts 로 요청하면 http://server:3000/contacts로 전달함
.....(생략)
export default defineConfig({
  plugins: [react()],
  server: {
    proxy: {
      "/api": {
        target: "http://server:3000",
        changeOrigin: true,
        rewrite: (path) => path.replace(/^W/api/, ""),
      },
    },
  },
});
```

6. 배포시 고려사항

- 각 서버별로 Proxy 설정 기능
 - Tomcat + JavaServlet
 - <https://github.com/mitre/HTTP-Proxy-Servlet>
 - Node.js + Express 기반의 경우
 - <https://github.com/chimurai/http-proxy-middleware>
 - Spring MVC Proxy
 - <https://www.baeldung.com/spring-rest-with-zuul-proxy>
 - Apache
 - <http://jinhokwon.tistory.com/121>
 - Python Django
 - <https://github.com/mjumbewu/django-proxy>