오픈소스 SW 입문

w7 GitHub 2(Web Page)

2023-0504, 강의실(과학관213)

강사 : 박노헌

https://github.com/nparkcourage/2023-kau-0504

한국항공대학교

W7 GitHub 2

목차

7교시 15:00	◆ **Web 기본** • Web 개요 • Web Browser • Web Page 구성
8교시 16:00	◆ **Web 개발** • 컨테이너와 VS Code를 이용한 개발환경 준비 • VS Code를 이용한 Web Page 개발 • GitHub Pages
9교시 17:00	◆ **복습** • issues feedback • 수업 내용 정리 • 이후 계획

W7 GitHub 2

첫째 시간

7교시 15:00	◆ **Web 기본** • Web 개요 • Web Browser • Web Page 구성
8교시 16:00	◆ **Web 개발** • 컨테이너와 VS Code를 이용한 개발환경 준비 • VS Code를 이용한 Web Page 개발 • GitHub Pages
9교시 17:00	◆ **복습** • issues feedback • 수업 내용 정리 • 이후 계획

Web 개요

인터넷(Internet)

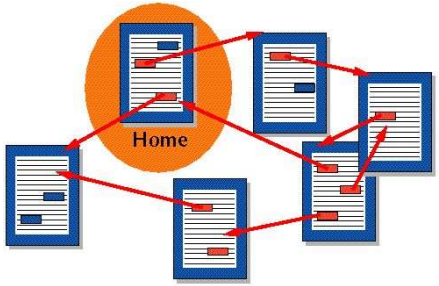
- 컴퓨터로 연결하여 TCP/IP(Transmission Control Protocol/Internet Protocol)라는 통신 프로토콜을 이용해 정보를 주고받는 컴퓨터 네트워크
- 네트워크와 네트워크를 연결해 주는 네트워크

월드 와이드 웹(World Wide Web, WWW, W3)

- 인터넷에 연결된 컴퓨터를 통해 사람들이 정보를 공유할 수 있는 전 세계적인 정보 공간(위키백과)
- 인터넷에서 HTTP 프로토콜, 하이퍼텍스트, HTML형식 등을 사용하여 그림과 문자를 교환하는 전송방식
- HTTP(**H**yper**T**ext **T**ransfer **P**rotocol) : 클라이언트와 서버 사이에 이루어지는 요청/응답(request/response) 프로토콜
- **하이퍼텍스트**(Hypertext) : 참조(하이퍼링크)를 통해 독자가 한 문서에서 다른 문서로 즉시 접근할 수 있는 텍스트, 하이퍼링크를 따라 다른 문서로 이동, HTML 문서라고도 함
- **하이퍼 텍스트 마크업 언어**(Hyper Text Markup Language, **HTML**) : 제목, 단락, 목록 등과 같은 본문을 위한 구조적 의미를 나타내는 것뿐만 아니라 링크, 인용과 그 밖의 항목으로 구조적 문서를 만들 수 있는 방법을 제공

https://ko.wikipedia.org/wiki/%EC%9B%94%EB%93%9C_%EC%99%80%EC%9D%B4%EB%93%9C_%EC%9B%B9

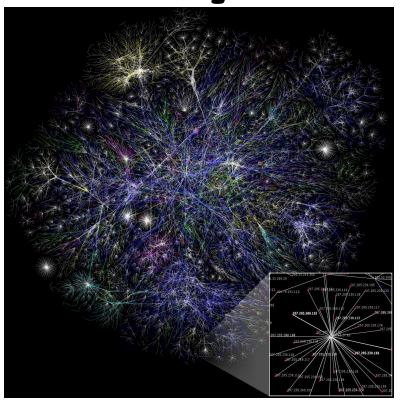
Web 개요



HTML

HTTP

TCP/IP



Web(WWW)

- 웹은 인터넷이라는 물리적 네트워크를 기반으로 함
- 인터넷 상에 TCP/IP라는 네트워크 전송 계층의 프로토콜을 사용
- TCP/IP 네트워크 전송 계층위에 HTTP라는 네트워크 응용계층 프로토콜을 사용
- HTTP 프로토콜을 이용해 HTML이라는 마크업 언어로 작성된 하이퍼 텍스트 데이터를 전송

Web Server(정적)

- 하이퍼 텍스트 파일등의 웹 페이지 구성 파일들을 제공(Publish)하는 서버 프로그램
- HTTP 서버라고도 하며 다양한 제품들이 있음
    - Apache, nginx

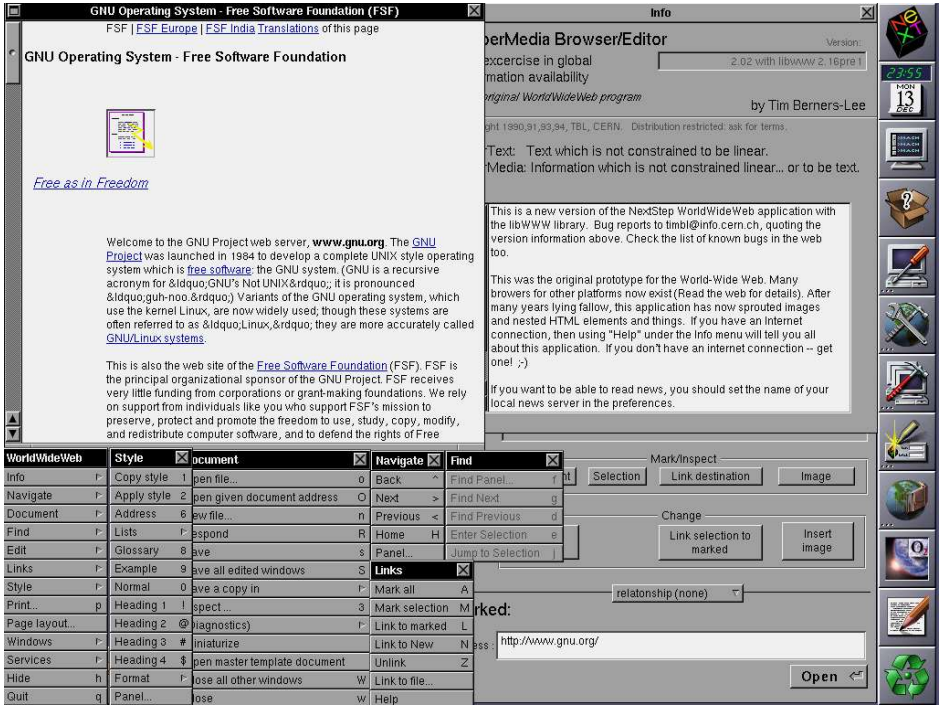
(Web) Application Server(WAS, AS)(동적)

- 서버에서 프로그램이 실행되고 그 결과로 HTML 파일을 생성하여 제공하는 서버 프로그램
- 주로 Database에서 데이터를 조회하고, 결과를 HTML에 표현하는 기능을 많이 함
    - JAVA : Weblogic, Apache Tomcat
    - C# : IIS
    - PHP : PHP
    - JavaScript : Node.js

Web 브라우저

Web 브라우저

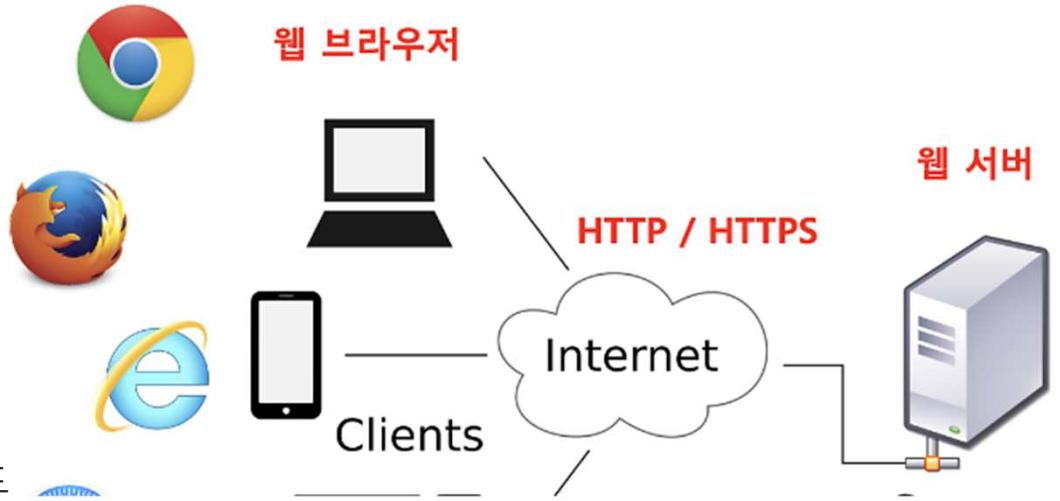
- 하이퍼 텍스트 문서(HTML 문서)를 렌더링하여 GUI로 보여주는 웹 클라이언트 프로그램
- 주요기능
  - 웹서버에게 주어진 URL의 웹페이지(하이퍼 텍스트)를 요청
  - 웹페이지를 다운로드 받아서 그래픽으로 렌더링
  - 자바스크립트(ECMAScript 표준 준수) 인터프리터로 자바스크립트 실행
  - 멀티 미디어 표시 : 이미지, 음악, 동영상



Web Page 구성

Web Page 구성 요소

- 문서의 내용과 구조를 제공하는 HTML이 기본 페이지이며 여기에 모양을 변형하는 CSS와 동작을 제공하는 JavaScript가 추가됨
- HTML만으로 이루어진 페이지는 존재하나 CSS나 JavaScript만으로 구성된 페이지는 존재할 수 없음
- **HTML : 내용(Content)**
  - 웹 페이지에 표시할 내용들을 포함
  - 웹 페이지의 구조를 기술
  - 엘리먼트(요소)들로 구성
- **CSS(Cascading Style Sheets) : 모양(Style)**
  - 웹사이트의 레이아웃을 정함
  - 글꼴, 색상 등의 디자인을 입히는 역할
- **JavaScript(ECMAScript) : 동작(Behavior)**
  - 객체 기반의 스크립트 프로그래밍 언어
  - 브라우저에서 실행되어 동적인 웹페이지 구성
  - 동적 클라이언트 사이드 -> Node.js는 서버 사이드
  - HTML과 CSS를 수정하는 역할



https://m.post.naver.com/viewer/postView.naver?volumeNo=33234329&memberNo=6255089

Web Page 구성

HTML

- 시작과 끝을 표시하는 태그 쌍들로 이루어짐
- 선언 : html을 선언하는 태그 쌍 내부에 내용이 들어감

```
<!doctype html></html>
```

- 헤드 : 화면에 표시되지 않는 메타데이터 기록 부분

```
<head>
  메타데이터
</head>
```

- 바디 : 화면에 표시되는 문서의 내용과 구조가 위치하는 부분

```
<body>
  웹 표현 내용(구조)
</body>
```

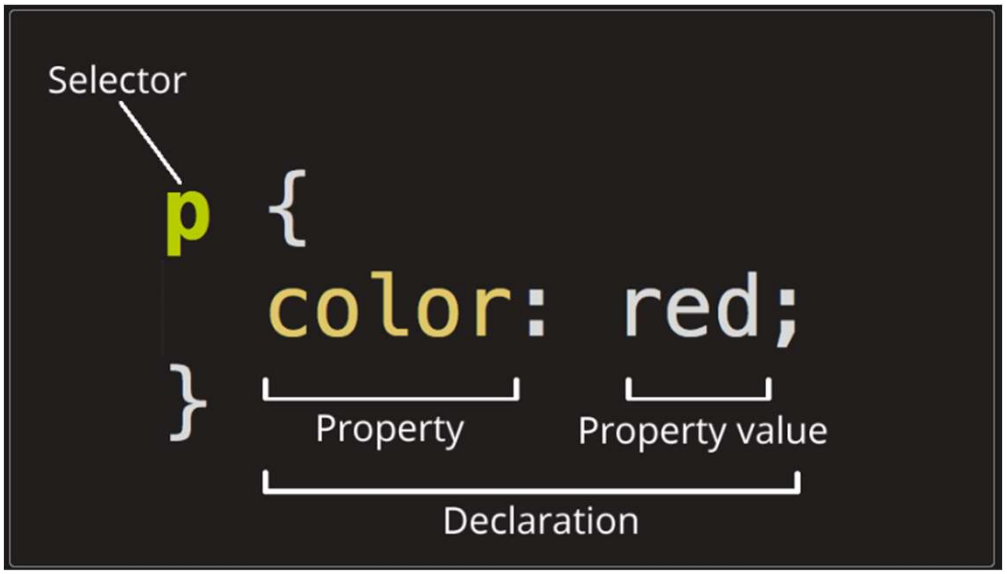
hello1.html(기본 html 파일)

```
<!doctype html>
<html>
  <head>
    <title>Hello HTML</title>
  </head>
  <body>
    <h1>Hello HTML!</h1>
    <p>OSS Intro 2023</p>
  </body>
</html>
```

Web Page 구성

CSS

- html 요소별 프라퍼티 지정
  - 요소이름(selector) {property:property value;}
- 3가지 설정 방식이 있음
  - Inline : HTML 요소에 직접 설정
  - Internal : HTML 파일의 내부에 CSS 설정
  - External : 외부파일에 CSS 설정하고 불러서 사용
- Inline CSS 예



https://developer.mozilla.org/ko/docs/Learn/Getting_started_with_the_web/CSS_basics

hello2.html(Inline CSS)

```
<!doctype html>
<html>
  <head>
    <title>Hello HTML</title>
  </head>
  <body style="background-color:lightblue;">
    <h1 style="color: white;text-align: center;">Hello inline CSS</h1>
    <p style="font-family: verdana;font-size: 20px;">OSS Intro 2023</p>
  </body>
</html>
```

Web Page 구성

CSS

- Internal CSS 예

hello3.html(Internal CSS)

```
<!doctype html>
<html>
  <head>
    <title>Hello HTML</title>
    <style>
        body {
          background-color: lightblue;
        }
        h1 {
          color: white;
          text-align: center;
        }
        p {
          font-family: verdana;
          font-size: 20px;
        }
    </style>
  </head>
  <body>
    <h1>Hello internal CSS!</h1>
    <p>OSS Intro 2023</p>
  </body>
</html>
```

Web Page 구성

CSS

- External CSS 예

hello4.html(External CSS)

```
<!doctype html>
<html>
  <head>
    <title>Hello HTML</title>
    <link rel="stylesheet" href="hello.css"/>
  </head>
    <h1>Hello external CSS</h1>
    <p>OSS Intro 2023</p>
  </body>
</html>
```

hello.css

```
body {
  background-color: lightblue;
}
h1 {
  color: white;
  text-align: center;
}
p {
  font-family: verdana;
  font-size: 20px;
}
```

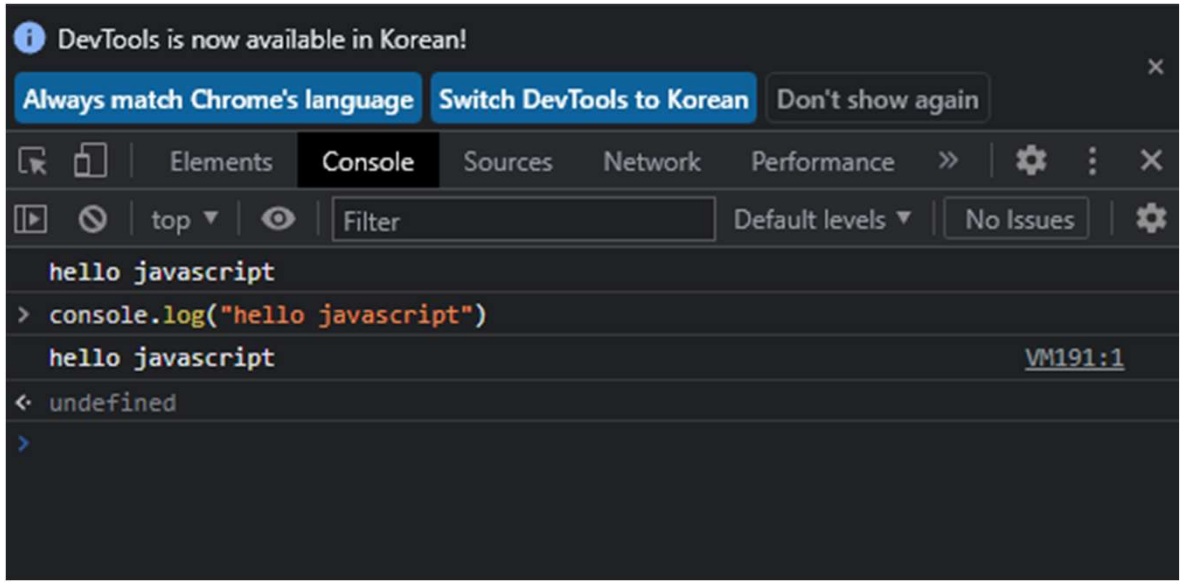
Web Page 구성

JavaScript

- 인터프리터 방식의 스크립트 언어 : 런타임에 코드 해석(번역)
- 표준 웹 브라우저에 인터프리터 내장
- 클라이언트(웹 브라우저)에서 페이지 내용과 모양을 변경
- ECMAScript 표준 준수
- HTML 파일 내부에 코딩하는 방법과 외부 파일에서 불러서 사용하는 방법 모두 지원
    - Internal : <script></script> 태그 사용
    - External : <header>에 외부 파일 지정

        <script type="text/javascript" src="message.js"></script>

- html의 구조를 알아야 함
- 개발자 콘솔(F12)에서 인터프리터 사용 가능
    - F12 클릭
    - Console 탭 선택
    - console.log("hello javascript")



Web Page 구성

JavaScript

- hello javascript 예
- 내부 코딩(Internal Coding) ascript")

hello_javascript.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Hello JavaScript</title>
  </head>
  <body>
    <h1>Hello JavaScript를 JavaScript로 출력</h1>
    <p id="hello-javascript"></p>
    <script>
      document.getElementById("hello-javascript").innerText = "Hello JavaScript";
    </script>
  </body>
</html>
```

Web Page 구성

동적 Client Web Page 예

clock.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>시계</title>
      <style>
             p {
               font-family: verdana;
               font-size: 30px;
               color: blue;
              }
      </style>
  </head>
  <body>
    <h1>시계</h1>
    <p id="clock"></p>
    <script>
      // 시계를 업데이트하는 함수
      function updateClock() {
        const now = new Date(); // 현재 날짜와 시간을 가져옵니다.

        const hours = now.getHours(); // 현재 시간 (0부터 23까지)
        const minutes = now.getMinutes(); // 현재 분
        const seconds = now.getSeconds(); // 현재 초

        const clock = document.getElementById("clock"); // 시계를 표시할 요소를 찾습니다.

        // 시간을 표시할 형식을 지정합니다.
        const timeString = `${hours}:${minutes < 10 ? "0" + minutes : minutes}:${seconds < 10 ? "0" + seconds : seconds}`;

        clock.textContent = timeString; // 시계 요소의 텍스트를 업데이트합니다.
      }

      // 1초마다 시계를 업데이트합니다.
      setInterval(updateClock, 1000);
    </script>
  </body>
</html>
```

Web Page 구성

클라이언트 사이드 동적 Web Page 개발

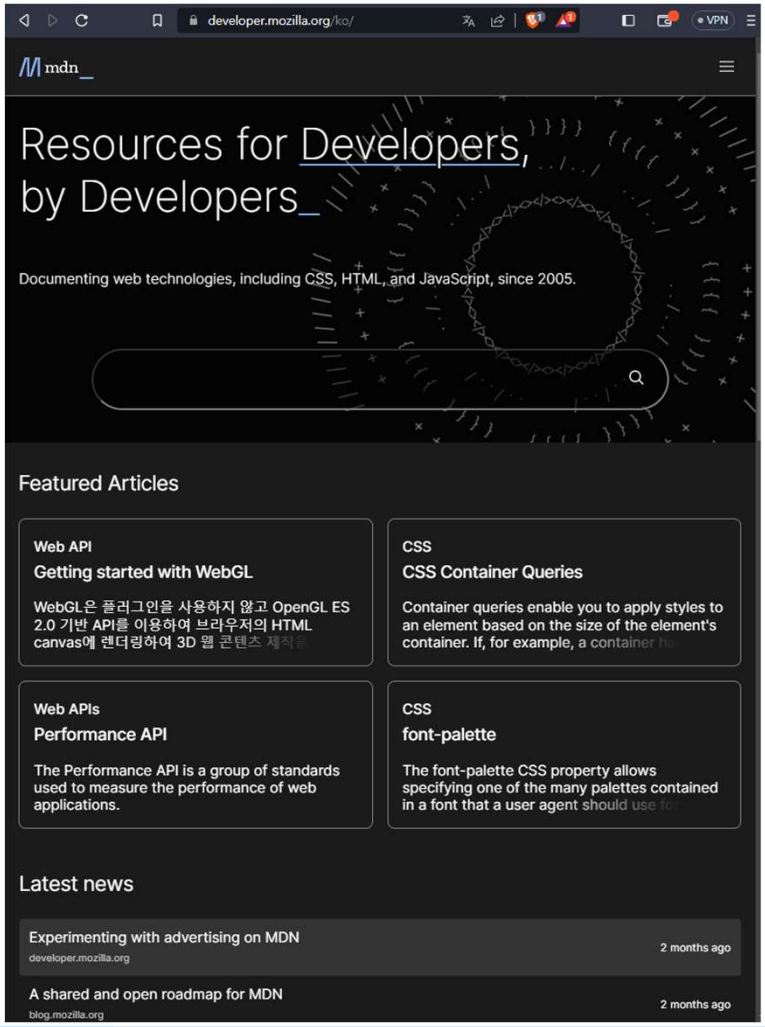
- 자바스크립트 언어를 이용하여 클라이언트 사이드의 동적인 페이지 구성 가능
- 자바스크립트는 브라우저에서 실행되므로 웹서버는 파일들(HTML, CSS, JavaScript)을 제공하는 역할만 함

  -> 서버에서 페이지 파일들을 만들거나 변경할 필요가 없음, 있는 파일들만 제공
- 자바스크립트 언어로 클라이언트(웹 브라우저)에서 페이지 내용과 모양을 변경
- 웹 페이지 UI(프론트 엔드) 개발을 위한 많은 라이브러리와 프레임워크 존재(JavaScript 인터프리터에서 실행)
  - Vue.js
  - ReactJS
  - Angular

Web Page 구성

참고 사이트

- https://www.w3schools.com
- https://developer.mozilla.org/ko



둘째 시간

7교시 15:00	◆ Web 기본 • Web 개요 • Web Browser • Web Page 구성
8교시 16:00	◆ **Web 개발** • 컨테이너와 VS Code를 이용한 개발환경 준비 • VS Code를 이용한 Web Page 개발 • GitHub Pages
9교시 17:00	◆ 복습 • issues feedback • 수업 내용 정리 • 이후 계획

컨테이너와 VS Code를 이용한 개발환경 준비

컨테이너 준비

- WSL 실행(Mac 사용자는 터미널 열기)
- 새로운 실습용 컨테이너 실행

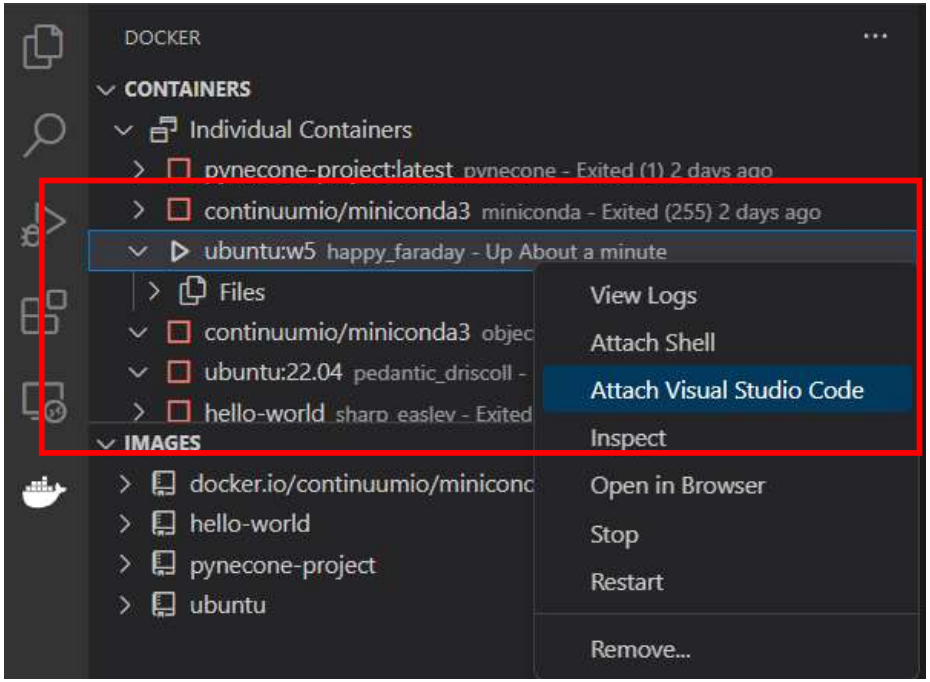
```
docker run -itd --name w7 -p 7070:8080 -v ~/projects:/workspace ubuntu:22.04 bash
```

- home으로 이동하여 VS Code 실행

```
cd
code .
```

- VS Code에서 도커 컨테이너로 새로운 VS Code 어태치(Attach)
  - 왼쪽의 도커(고래) 아이콘 클릭
  - 풀다운 메뉴에서 "Attach Visual Studio Code" 선택

  -> 새로운 VS Code 열림

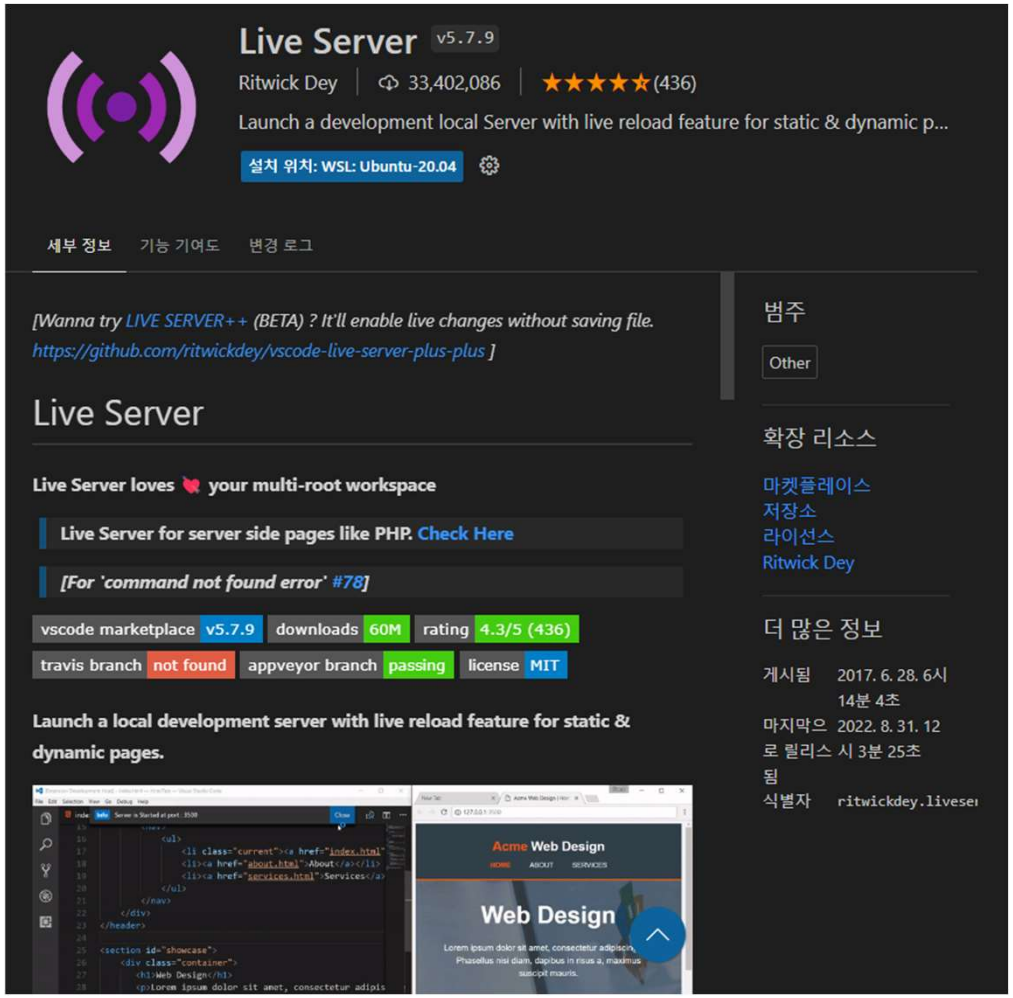


컨테이너와 VS Code를 이용한 개발환경 준비

컨테이너 준비

- 컨테이너 VS Code에 확장 설치

  Live Server



컨테이너와 VS Code를 이용한 개발환경 준비

컨테이너 준비

- SSH 개인키 복사

  host(WSL)의 ~/.ssh/id_ed25519의 내용을

  컨테이너의 ~/.ssh/id_ed25519 파일로 복사 : 파일을 새로 만들고 내용 복사

  ~/.ssh/id_ed25519 파일의 모드 수정

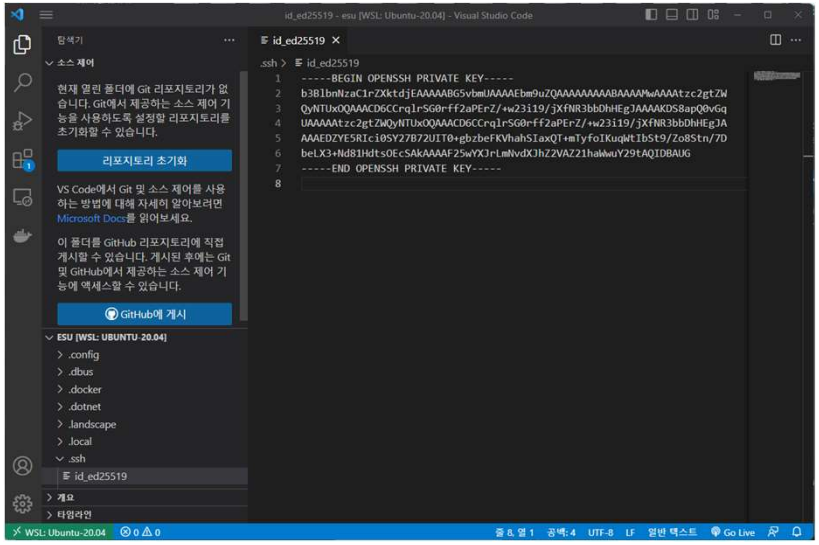
  ```
  chmod 600 ~/.ssh/id_ed25519
  ```

  확인

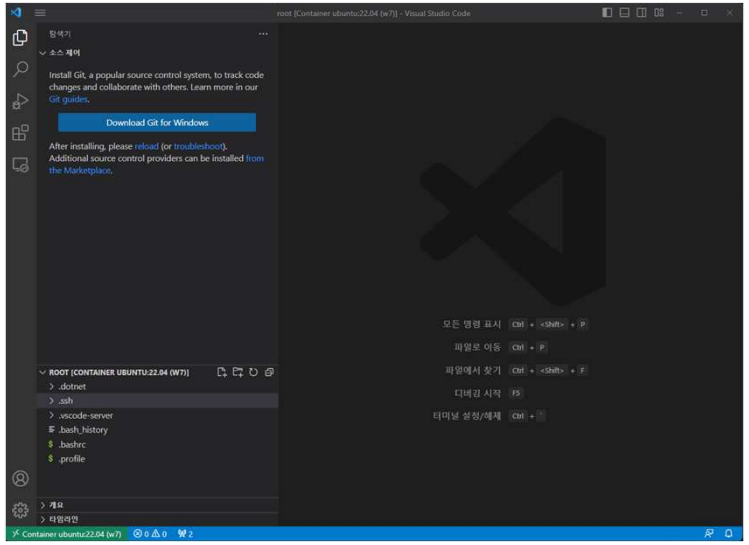
  ```
  ls -l ~/.ssh/id_ed25519
  ```

  ```
  -> -rw------- 1 root root 419 Apr 18 07:23 /root/.ssh/id_ed25519
  ```

host(wsl)에서 열린 VS Code



컨테이너에서 열린 VS Code



컨테이너와 VS Code를 이용한 개발환경 준비

컨테이너 준비

- 컨테이너에 git 설치

```
apt update
apt install git
```

- 컨테이너에 git 사용자 설정

```
git --global user.name "<user name>"
git --global user.email <user email address>
```

VS Code를 이용한 Web Page 개발

GitHub 저장소 만들기

- Repository name : w7
- Public : 체크(공개 저장소)
- Add a README file : 체크하지 않음
- Add .gitignore : None
- Choose a license : None

컨테이너에서 클론

- 컨테이너의 /workspace에서 클론

```
cd /workspace
git clone ssh-url
```

웹 파일 편집

- 클론한 디렉토리에서 웹파일 편집
- HTML, CSS, JavaScript 등의 웹용 파일들을 생성하고 편집

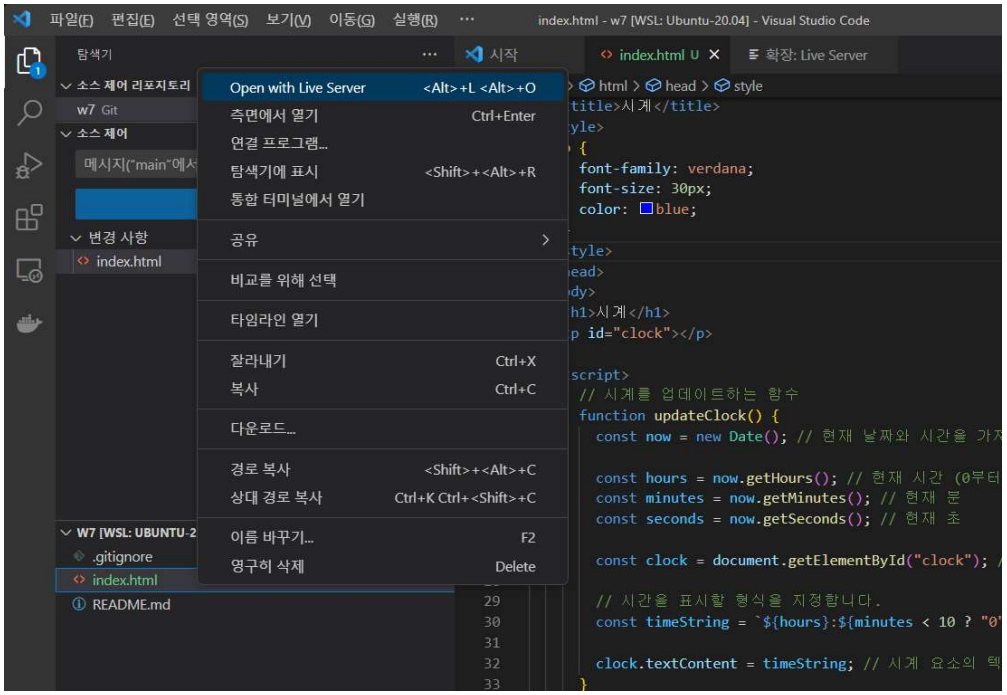
VS Code를 이용한 Web Page 개발

Web Page 구성

- 웹서버의 기본(root)디렉토리가 있음
- 기본(root) 디렉토리에 있는 index.html이 기본 페이지가 됨
- 같은 웹서버에 있는 페이지는 상대경로로 링크 가능
- 그 외의 링크는 URL 이용

Web Page 예

- 클론한 w7 디렉토리에 웹 파일들 추가
- https://github.com/nparkcourage/2023-kau-0504/w7 참고



웹 파일 확인

- 라이브 서버를 이용하여 웹브라우저에서 웹페이지 확인

GitHub 저장소로 업로드

- VS Code에서
  - add
  - commit
  - push(동기화)

GitHub Pages

GitHub의 Web Page Hosting

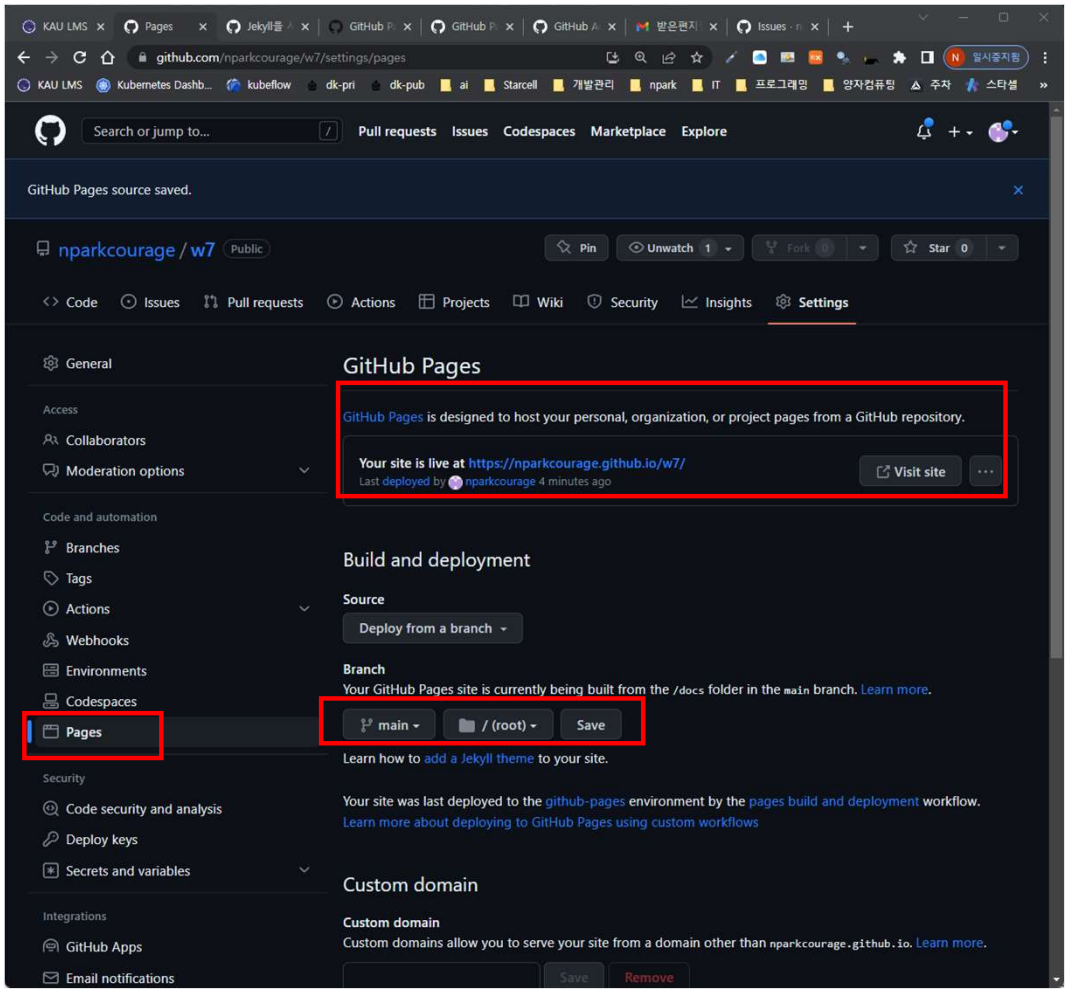
- GitHub에서는 Web Page Hosting Service 제공
- 정적인 Hosting
  - HTML, CSS, JavaScript 파일 서비스 기능
  - 서버 사이드 동적 호스팅 불가

GitHub Pages는 2가지 페이지 서비스 제공

- Project Pages Hosting
  - 각 프로젝트(저장소) 별 페이지 호스팅
- User Page Hosting
  - 각 사용자 계정 당 하나의 페이지 호스팅

Project Page만들기

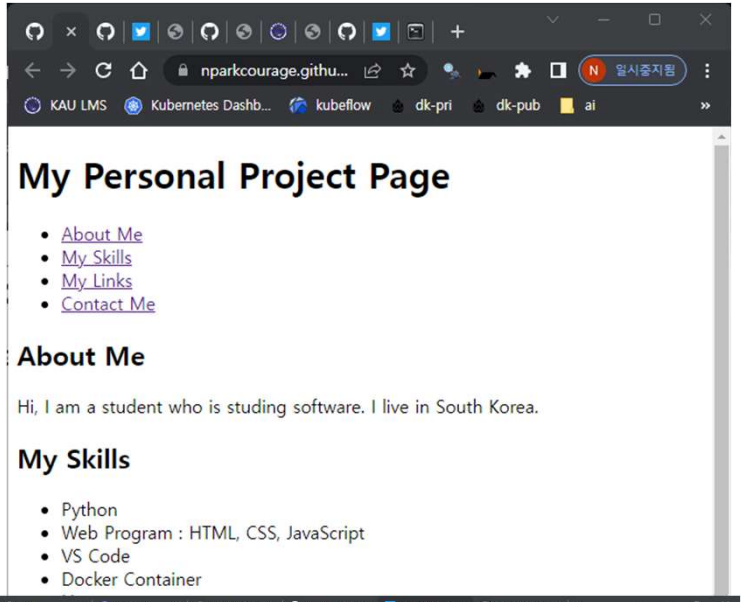
- 저장소의 "Settings" 메뉴에서 "Pages"에서 "Branch"에서 설정
- https://<username>.github.io/<저장소이름>으로 publishing 됨



GitHub Pages

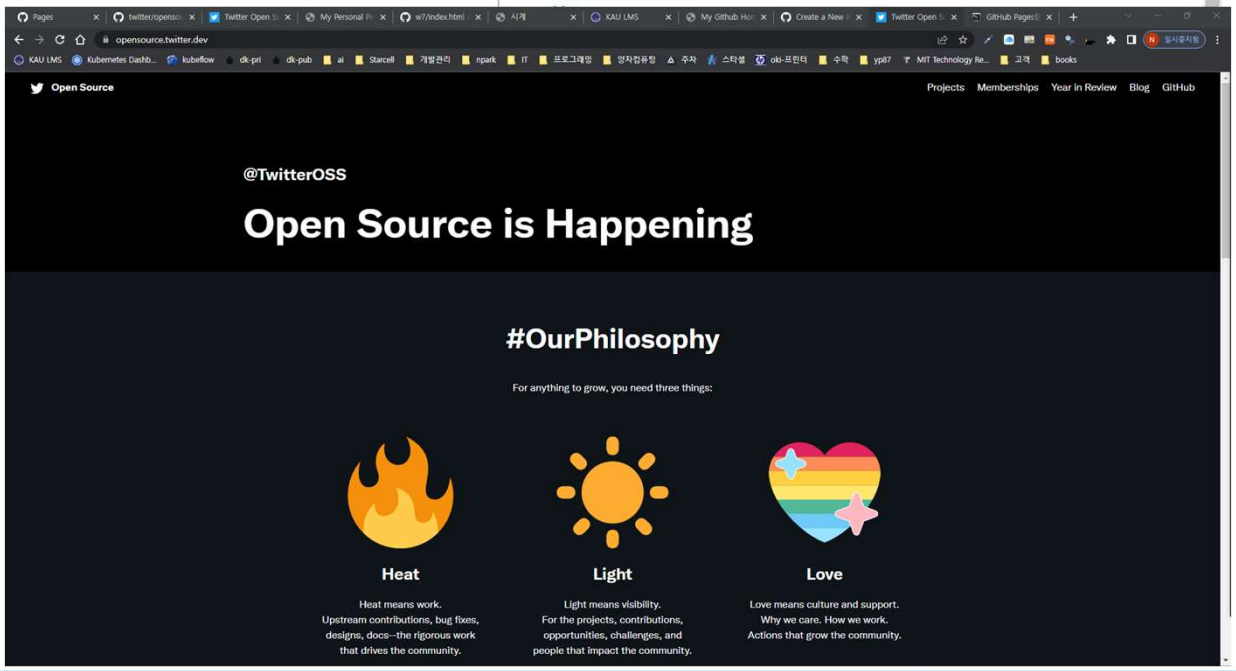
GitHub Pages URL

- 다음과 같은 방식으로 접속 가능
  - /index.html 파일은 https://사용자이름.github.io/저장소이름/
  - /aaa/index.html 파일은 https://사용자이름.github.io/저장소이름/aaa
  - /bbb.html 파일은 https://사용자이름.github.io/저장소이름/bbb
- 저장소 URL : https://github.com/nparkcourage/w7
- Web Page URL : https://nparkcourage.github.io/w7
- **git push 후에 즉시 반영되지 않음**



GitHub 페이지 예

- 트위터 오픈 소스 페이지
- https://github.com/twitter/opensource-website



GitHub Pages

GitHub Pages 제한 사항

- 상업적 사이트 금지
- 거래에 대한 위험성 주의
- 사용 제한
  - GitHub Pages 소스 저장소의 권장 제한은 1GB입니다.
  - 게시 된 GitHub 페이지 사이트는 1GB를 초과 할 수 없습니다.
  - GitHub 페이지 사이트의 대역폭 제한은 한 달에 100GB입니다.
  - GitHub 페이지 사이트의 builds 제한은 시간당 10회 입니다.

https://wepplication.github.io/programming/github-pages/

세째 시간

7교시 15:00	◆ Web 기본 • Web 개요 • Web Browser • Web Page 구성
8교시 16:00	◆ Web 개발 • 컨테이너와 VS Code를 이용한 개발환경 준비 • VS Code를 이용한 Web Page 개발 • GitHub Pages
9교시 17:00	◆ 복습 • issues feedback • 수업 내용 정리 • 이후 계획

issues feedback

1. 긍정의견

- 실무적인 내용들을 실습할 수 있어서 좋았다.
- 혼자 하면 막막하거나 시간이 많이 걸릴 내용들을 할 수 있어서 좋았다.
- 특히 Linux, Docker 컨테이너 등 중요하다고 이야기 들었던 것을 직접 사용해 볼 수 있어서 좋았다.

2. 부정의견

- 한 번 놓치면 따라가기 어렵다.
- 도커 컨테이너와 같은 것이 왜 필요한 지 어떻게 활용되는 지 설명이 있으면 좋겠다.
- 새로운 용어나 개념이 많이 나오는데 설명이 좀 부족하다.
    - 특히 네트워크 매핑 관련 설명이 더 있었으면 좋겠다.

issues feedback

1. 컨테이너를 사용하는 이유

- 여러 대의 컴퓨터가 필요한 상황

  한 대의 컴퓨터에 많은 소프트웨어를 설치해서 사용할 경우 서로 영향을 미칠 수 있으므로 독립적인 컴퓨터를 원함

  - 다양한 용도 : 개발용, 빌드용, 테스트용
  - 다양한 환경 : 다양한 소프트웨어와 다양한 버전에서 개발, 실행, 테스트가 필요한 경우
  - 다양한 기능 : DB Server, Web Server, Application Server, 각 서비스별 시스템 분리(마이크로 서비스)

- 소프트웨어 설치를 쉽게 하기

  새로운 소프트웨어를 설치할 때 간편하게 할 수 있음

  - 필요한 소프트웨어를 설치해서 사용할 때 의존성에 따라 많은 소프트웨어와 라이브러리를 설치해야 함
  - 해당 소프트웨어의 컨테이너 이미지를 받아서 실행

- 컴퓨터 환경을 이전해야 하는 경우

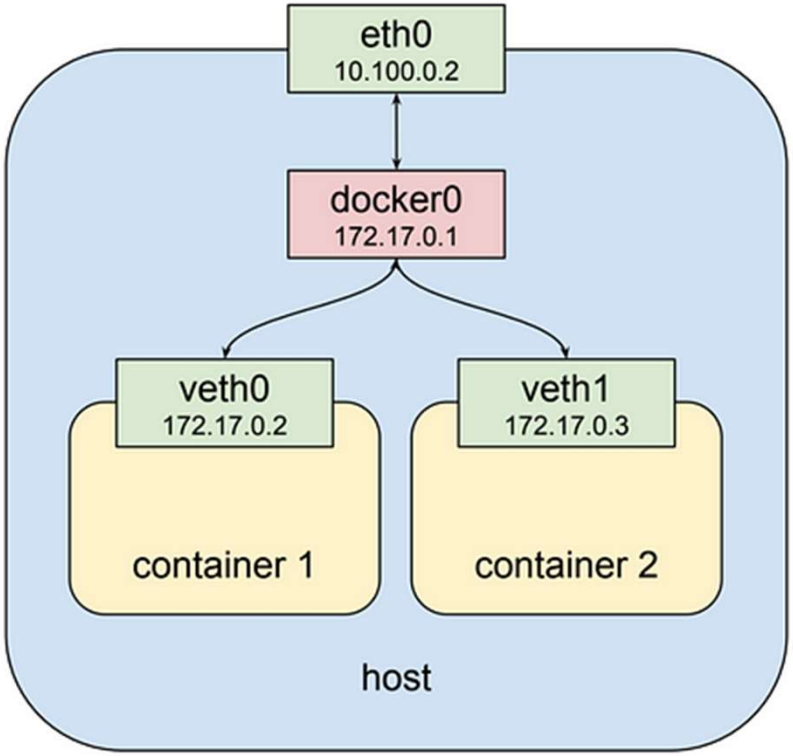
  소프트웨어를 다른 컴퓨터에 동일한 환경으로 다시 설치해서 사용해야 하는 경우

  - 많은 소프트웨어들을 다시 설치해야 함(대부분의 경우 OS까지 맞추어서 다시 설치 필요)
  - 많은 소프트웨어들에 대해 모두 설정을 다시 해야 함
  - 개발 시스템에서 서비스 시스템으로 이전(배포)
  - 성능이 좋은 새로운 서버로 이전(마이그레이션)
  - 다른 클라우드로 이전

issues feedback

2. 네트워크 매핑

- 컨테이너는 호스트의 내부에서 작동
- 내부에서만 통신하도록 되어 있음
- 외부에서 내부로는 들어갈 통로가 없음
- 호스트에서 받아서 전달해 줄 수 있는 통로를 만들어야 함
  - 포트 매핑 또는 포트 전달(forwarding)
  - docker -p <호스트 포트>:<컨테이너 포트>



https://medium.com/finda-tech/kubernetes-%EB%84%A4%ED%8A%B8%EC%9B%8C%ED%81%AC-%EC%A0%95%EB%A6%AC-fccd4fd0ae6

W7 GitHub 2 : 복습

수업 내용 정리

w1. Open Source License

- 반환의무(reciprocal) : 소스 공개 의무
- 반환의무불필요(permissive) :
- 반환의무가 있는 라이선스와 불필요한 라이선스 구별

w2. Linux, CLI, shell script

- OS의 커널이 무엇인지?
- Linux 명령어(CLI)
  - 실행 중인 프로세스 보기
  - 문자열 다루기 : tail, cut 등을 활용
  - 파일의 권한, 파일 권한 수정 명령
  - ssh key 만드는 명령
  - 주소가 175.117.50.9인 원격 시스템에 포트 번호 10022번으로 edu 유저로 접속하기 위한 명령어는?

```
ssh edu@195.117.50.9 -p 10022
```

w3. git, GitHub

- git 명령
  - git add
  - git commit
  - git push
  - git config
- GitHub 사용을 위한 키등록과 clone, push 등
- 마크다운

수업 내용 정리

w4. 도커 1

- 도커를 사용하는 이유
- 도커 명령

w5. 도커 2

- 도커 이미지 커밋
- 도커에서 네트워크 매핑
- 도커에서 볼륨 매핑(바인드 마운트)

w6. VS Code

- 사용 관련 단축키 및 메뉴, 아이콘
- VS Code를 사용하여 직접 할 수 있는 작업
  - 텍스트 파일 편집
  - git 연동
  - 원격 사용(컨테이너 포함)
  - 터미널 사용
  - 주피터 노트북 파일 실행

w7. GitHub 2

- web과 web 서비스
- web page 구성 요소
- GitHub Pages

이후 계획

GitHub을 이용한 협업

- GitHub에 팀프로젝트 저장소 만들기
- 분담하여 개발
- 개발한 소스 합치기
- GitHub를 이용한 커뮤니케이션

GitHub에 Team Page 만들기

- 팀별로 주제를 정해서 웹 페이지 제작

GitHub에 User Page 만들기

- 개인용 웹 페이지 만들기

동적 웹서버

- pynecone을 이용한 동적 웹 서비스 만들기

RDBMS

- MariaDB : open source RDBMS
- dbeaver : open source DB관리 툴
- SQL 기본

웹서버 프로그램에 RDBMS연동

- RDBMS의 데이터를 조회하여 표시하는 웹 서비스 개발
- pynecone과 MariaDB 사용

그 외 원하는 내용

- 학생들이 원하는 내용들 중 가능한 내용
- 준비할 시간 필요