Linux 없이 Linux 에서 개발하기

Great Technology For Great Games

김진욱

jinuk.kim@ifunfactory.com

IFUN FACTORY

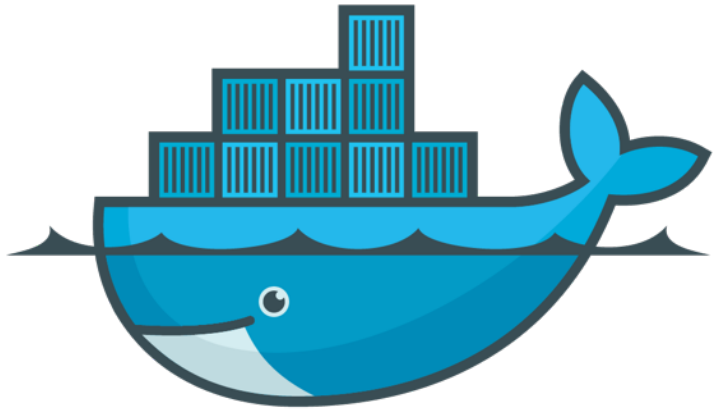
개발 환경: 원하는 것

- Linux 기반의 라이브러리와 서비스를 쓰고 싶다
- Windows / macOS 에서 개발하고 싶다
- GUI 툴이 더 좋다
- 여러 대의 가상 환경을 개발 PC/랩탑 하나에서 모두 /한꺼번에 다루고 싶다

01 Docker 로 linux 개발환경 만들기

Docker

- Linux 기반*
- 얇고/가벼운 가상화 환경
- 개별 디스크 (=이미지) 를 각각의 얇은 VM (=컨테이너) 으로 사용하는 방식



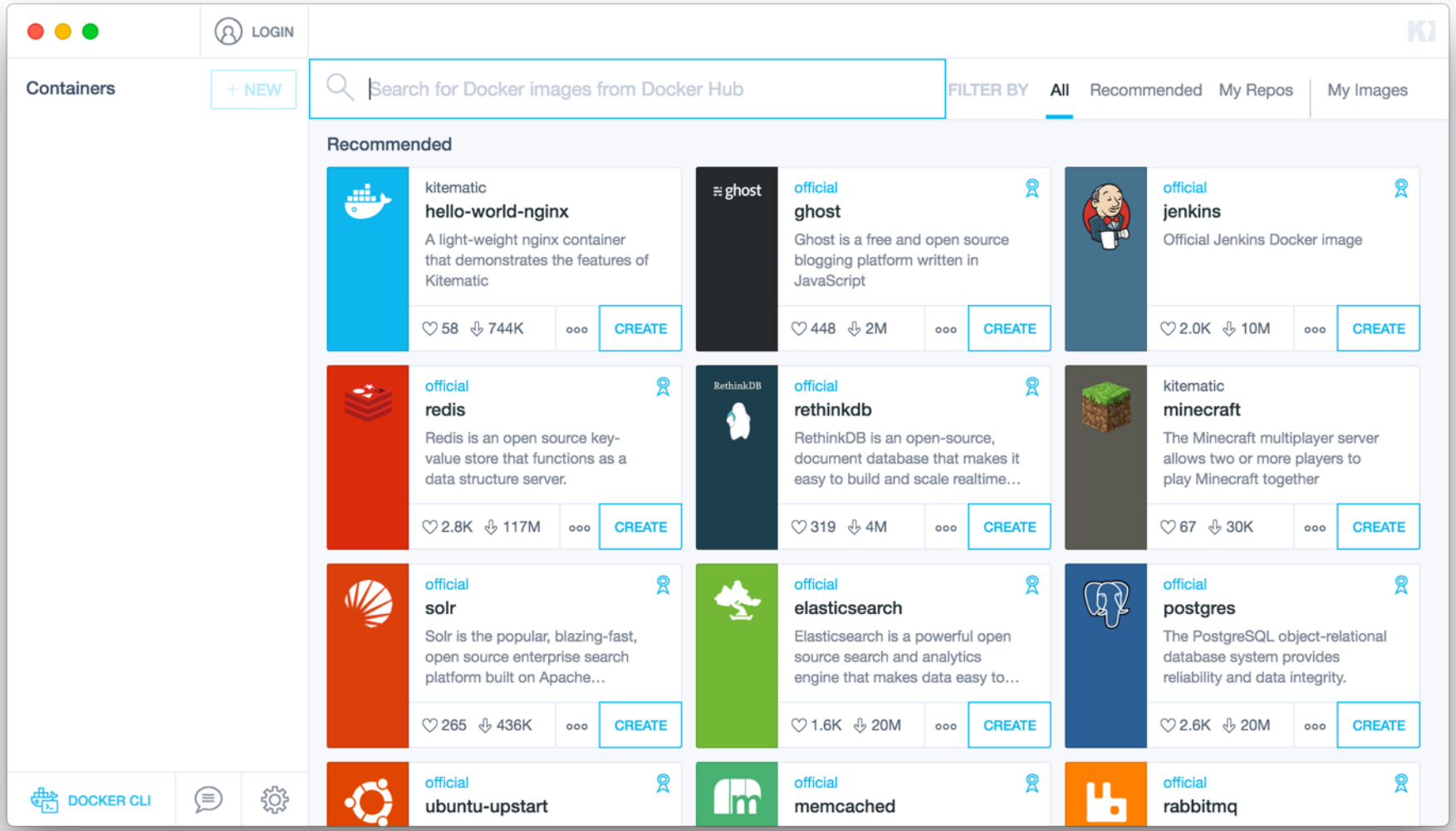
가벼운 가상화 (?)

VM 보다 더 적은 자원 (메모리, 디스크) 을 쓴다

랩탑에서도 동시에 여러 docker 컨테이너 사용 가능

가벼운 디스크 형식

공통 이미지를 수정하면 변경 사항이 덧 씌워지는 방식



OS 지원

- Windows / macOS 정식 지원 시작 (2016-07)

  Docker For Mac And Windows Is Now Generally Available And Ready For Production

- Windows 를 컨테이너 안에서 쓰기:

  Windows Containers on Windows Server (Windows Server 2016; 출시전)

I FUN FACTORY

C++ 프로젝트를 빌드하는 Docker 컨테이너를 띄워봅시다

I FUN FACTORY

환경 설정: Dockerfile

- 하나의 docker 컨테이너를 설정하는 파일
- 다음과 같은 구성 요소를 갖는다:
  - 기본 컨테이너 이미지 (FROM)
  - 추가 설정 작업 (쉘 명령; RUN …)
  - 외부에 열어놓을 포트 목록 (EXPOSE)
  - 외부에서 가져와서 공유할 디렉터리 (VOLUME)
  - 실제로 할 작업 (ENTRYPOINT …)

Example: 이미지 만들기

- Ubuntu 16.04 (xenial)
- g++, cmake 를 설치
- /workspace 공유 디렉터리
- 8000 포트를 연다

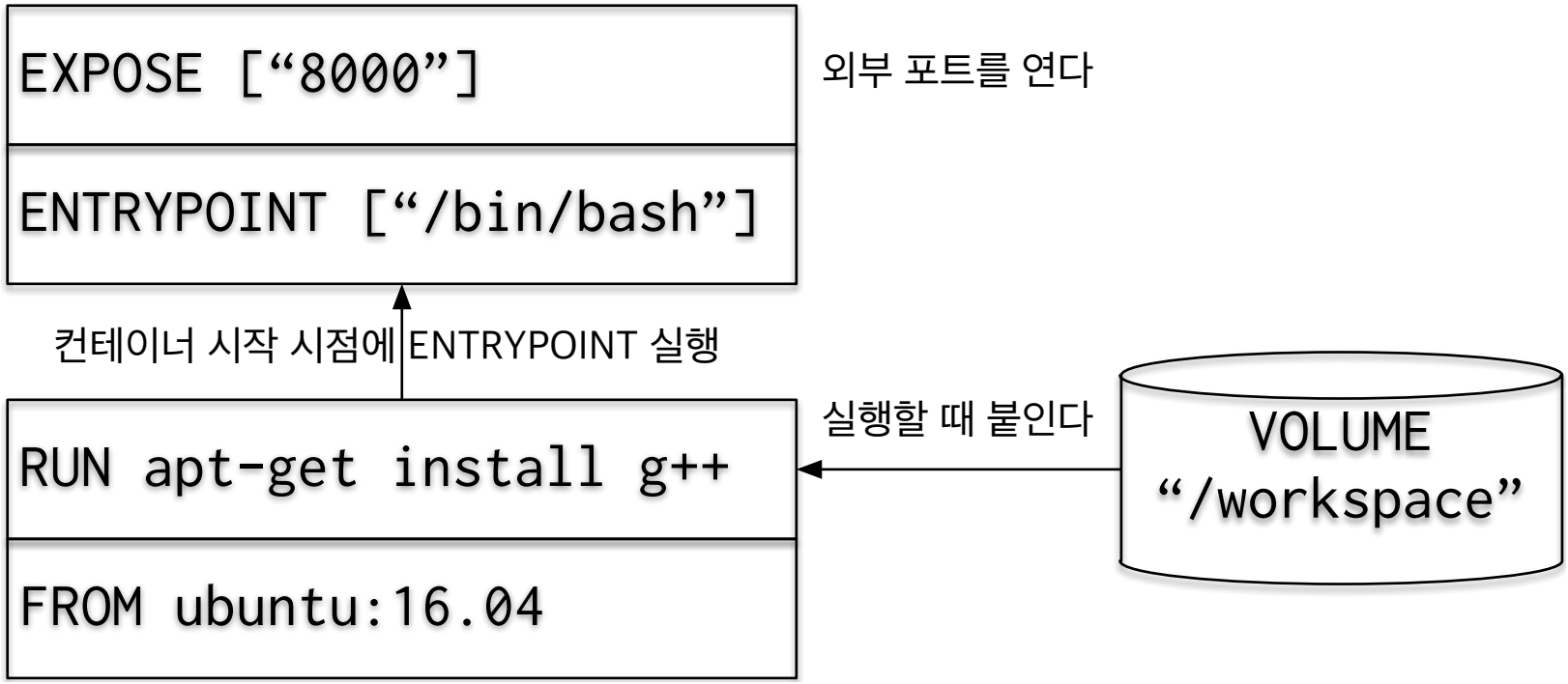
Dockerfile

```
1  FROM ubuntu:16.04
2  RUN apt-get -qq update \
3    && apt-get -qq install -y g++ cmake \
4    && apt-get clean
5
6  EXPOSE 8000
7  VOLUME ["/workspace"]
8  ENTRYPOINT ["/bin/bash"]
```

- `docker build --rm -t cpp-builder .`

  (Dockerfile 이 있는 디렉터리에서 실행)

Example: 이미지 만들기 (2)



Example: 컨테이너 띄우기

- 빌드한 이미지는 컨테이너로 띄워서 사용합니다.
- 예를 들어 crow 라는 프로젝트를 빌드해봅시다:

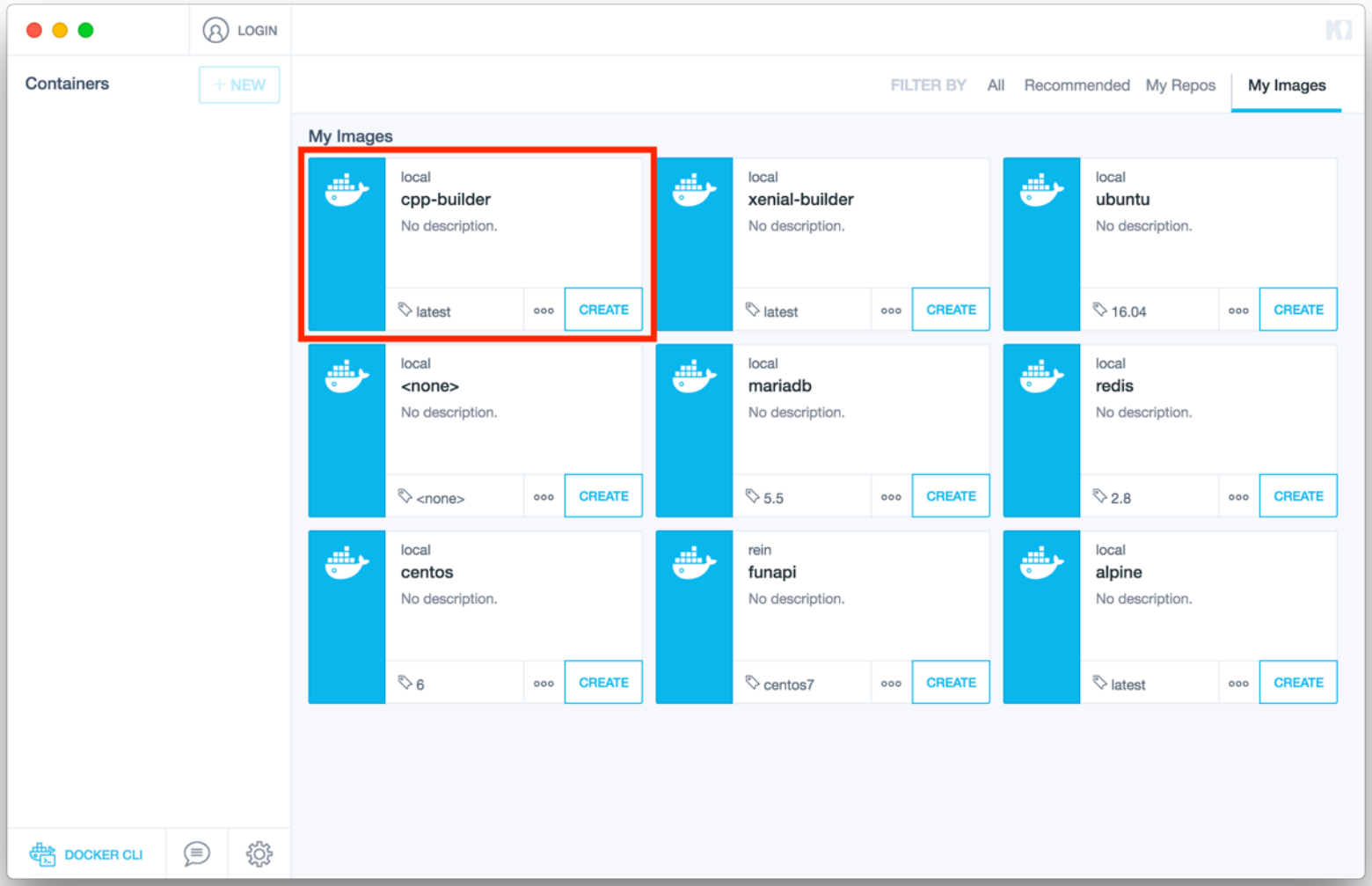
```
# 데스크탑 혹은 랩탑에서 실행
$ docker run -it --rm -v ~/work/crow:/workspace cpp-builder
# 여기서부터는 docker 컨테이너 안
$ apt-get update
$ apt-get install -y libgoogle-perftools-dev libssl-dev libboost-all-dev
$ cd /workspace
$ mkdir build; cd build; cmake ..; make -j2
```

02 GUI를 쓰고 싶습니다

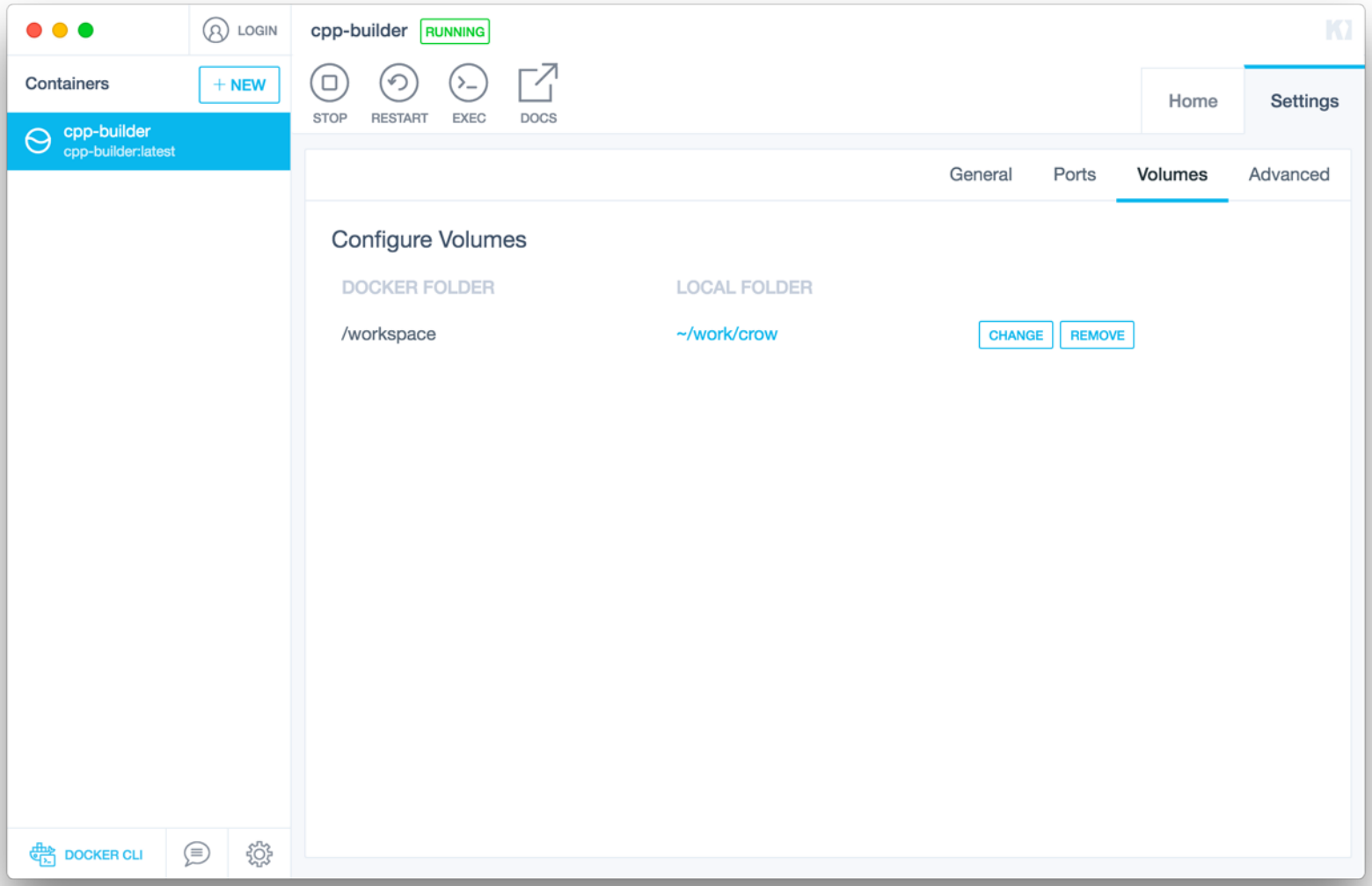
Remind:

- 명령행에서 작업하는 걸 피하고 싶다
- GUI 툴은 없을까?

Kitematic: docker run



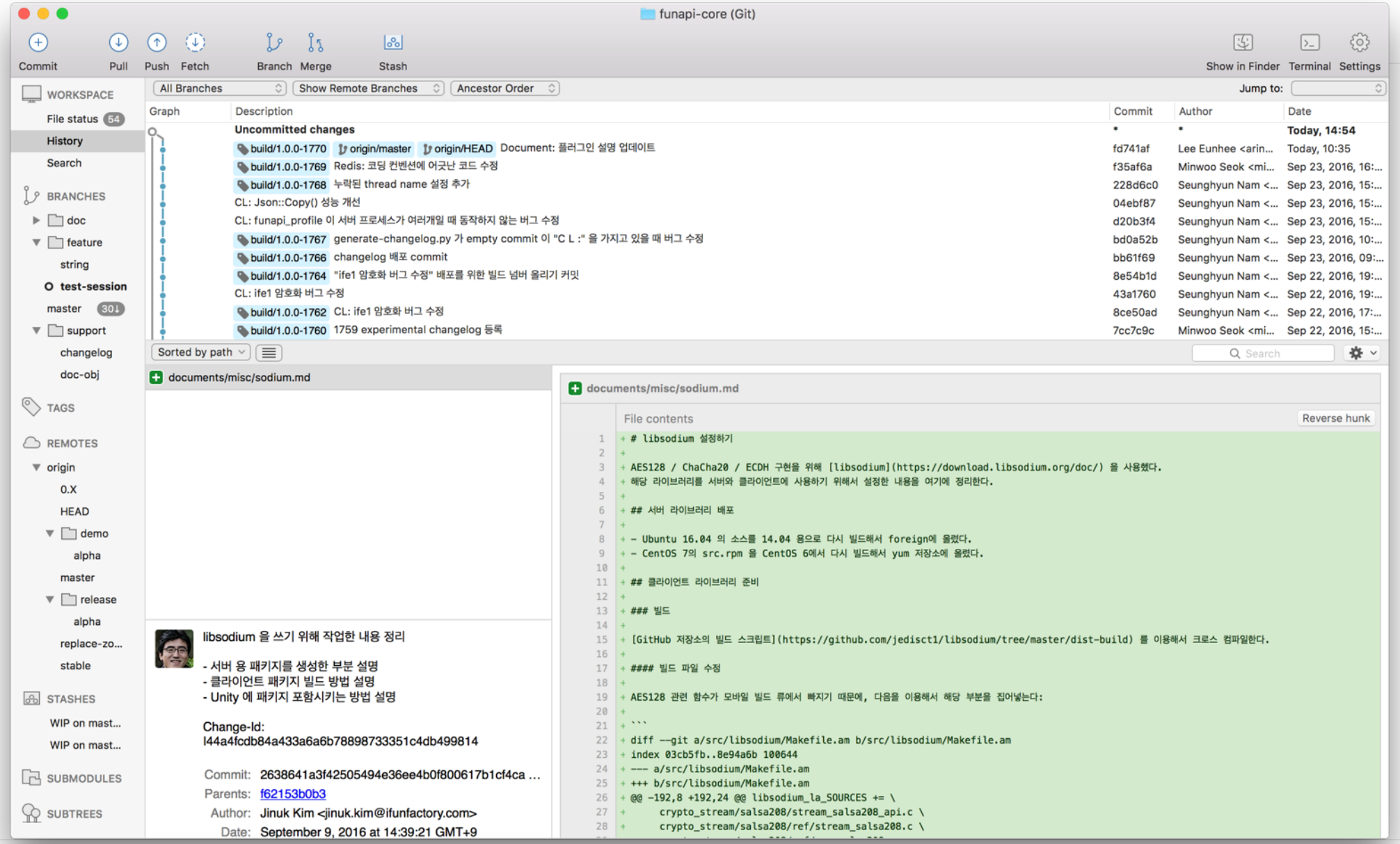
Kitematic: GUI로 대체하자



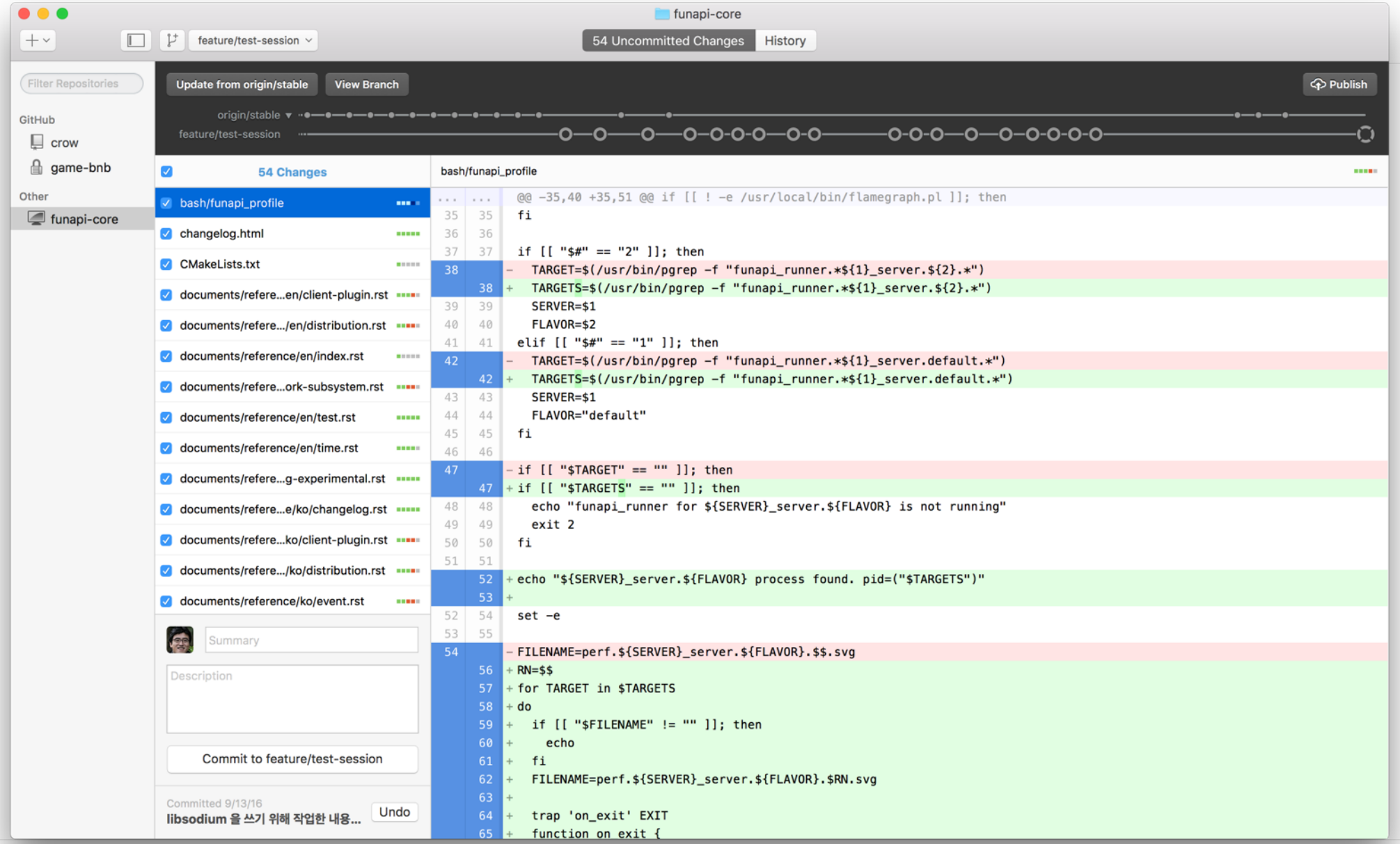
다른 작업은?

- Dockerfile 의 Volume: 공유 디렉터리처럼 동작
- 소스코드를 OS / Docker 컨테이너 사이에 공유하고, GUI 도구는 Windows / macOS에서 실행
- 예를 들어,

SourceTree (git client)



GitHub Desktop



…

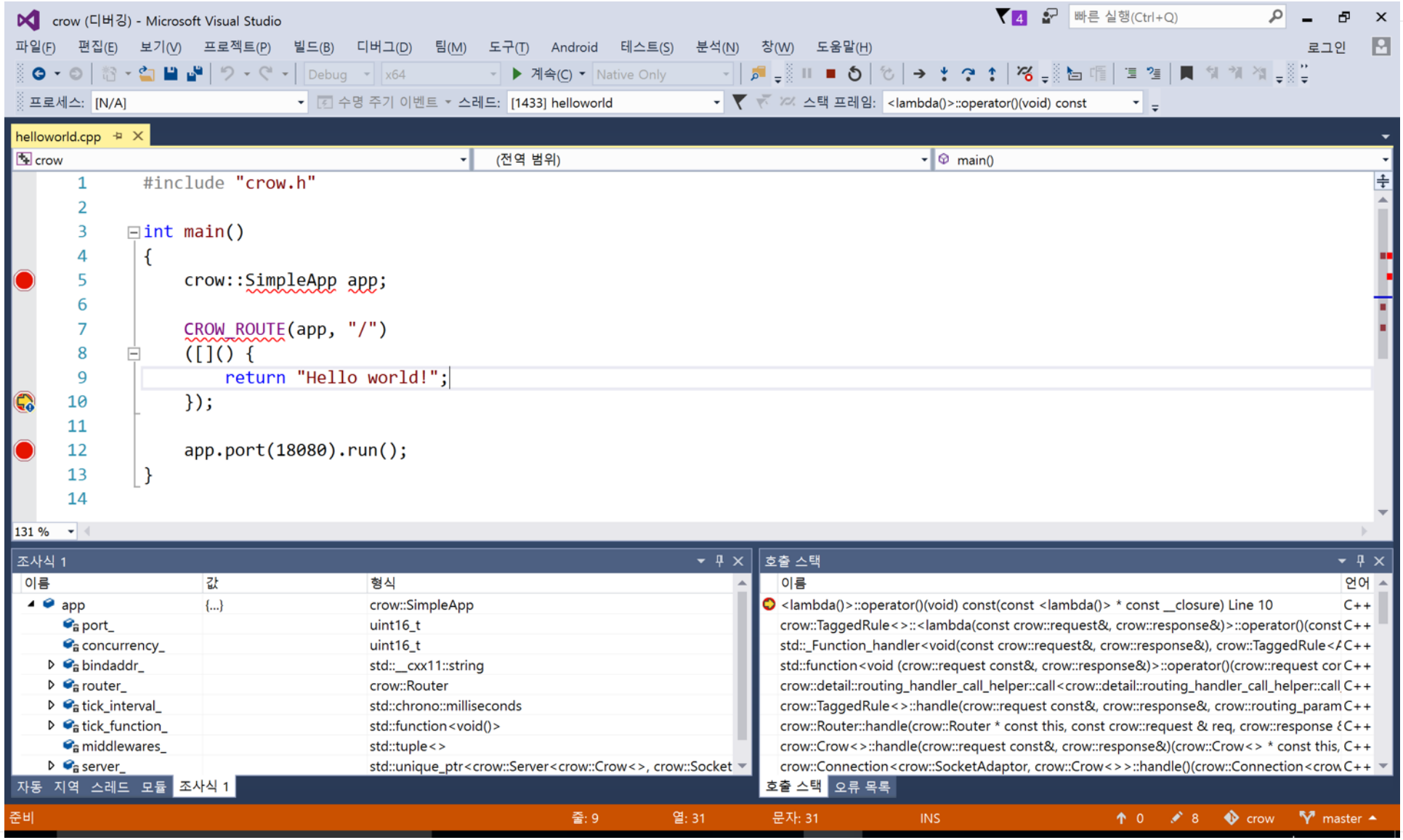
- 컴파일러나 링커와 직접 상호작용하지 않는 대부분의 툴은 개발하는 PC에서 바로 사용한다
- 빌드 툴 (컴파일러/링커…) 만 docker 안에서 명령행으로 돌린다 (?)

Docker + IDE 환경?

- Windows / macOS 사용자에게 GUI가 더 편하다
- 다음 작업은 GUI에서 하고 싶다:
  - 편집 (+자동 완성)
  - 다른 심볼 (함수, 변수, 클래스 정의 …) 참조
  - 디버깅
- 어떻게 해야할까?

03 – 개발 환경도 GUI를 쓰고 싶습니다
On Windows

Windows: VisualStudio

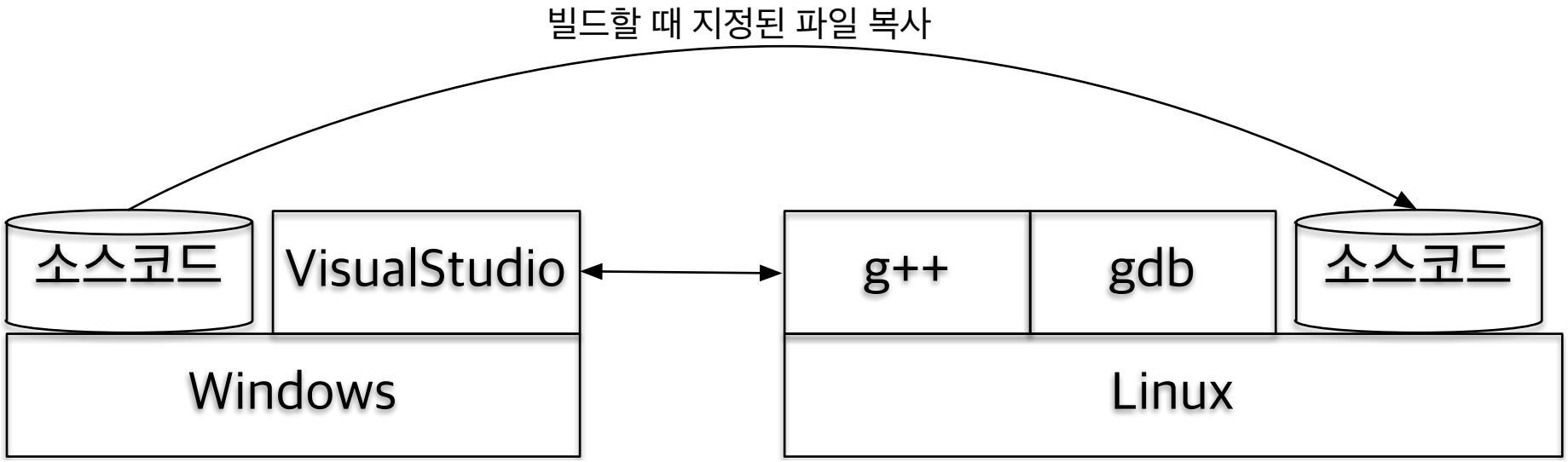


Visual C++ for Linux Development

- Microsoft 에서 제작하는 linux 개발용 VS 확장기능
- C++ 원격 개발 지원
- Windows + VisualStudio에서 원격으로
  - 빌드
  - 소스코드 편집
  - 디버깅

어떻게 동작하는가?

- VisualStudio 가 linux 서버와 직접 통신 (SSH)
- 사용자가 지정한 빌드/디버깅 명령을 실행
- 소스코드는 Windows 쪽에서 linux 쪽으로 복사



컨테이너로 바꿉시다

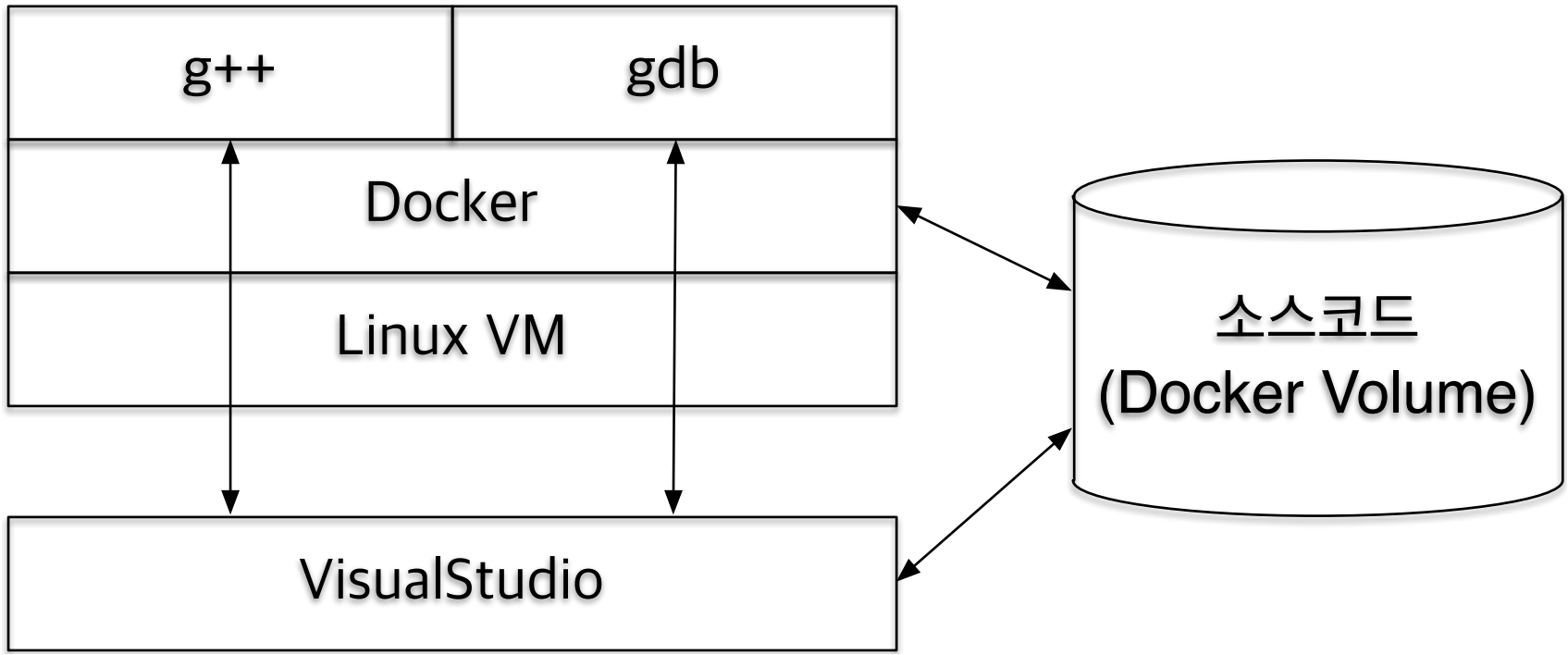
- g++, gdb, … 을 설치 (필요한 빌드툴/라이브러리)
- SSH 서버를 설치 + 설정
- 컨테이너를 실행하면 SSH 서버를 실행
- 소스코드를 공유할 volume 정의
- SSH용 22 포트를 외부로 연다

컨테이너 띄우기

- SSH 포트 지정
- Volume 지정이 주의*
- 디버거가 동작하기 위해서 권한이 더 필요

Windows: VisualStudio (1)

- 소스코드는 docker volume 으로 공유
- Port 는 docker 에서 포워딩한 ssh 포트를 이용

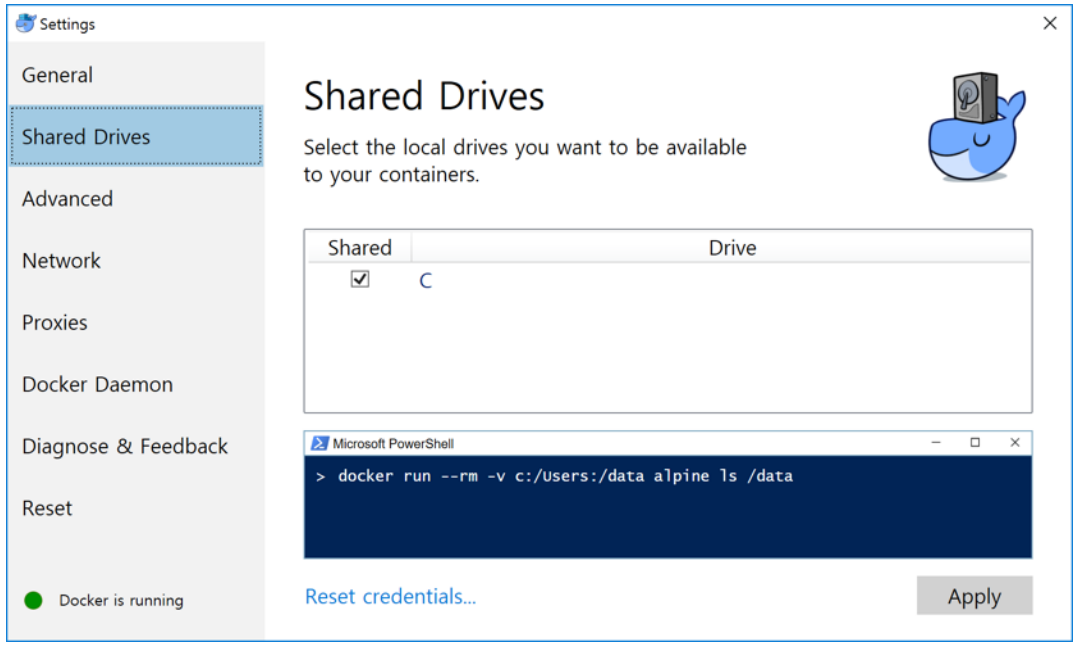


Windows: VisualStudio (2)

- 소스코드 복사는 불필요 (volume 을 양쪽에서 공유)
  대신, 소스코드 디렉터리를 프로젝트 설정에서 지정
- 직접 지정할 프로젝트 설정이 잔뜩 있습니다:
  - 빌드, 디버깅 명령
  - 빌드 결과물 나열
  - …

Windows: misc.

- Docker 가 디버거에 필요한 일부 권한을 컨테이너에게 주지 않는다: 컨테이너를 privileged 상태로 실행
- Docker For Windows 의 공유 드라이브를 꼭 설정

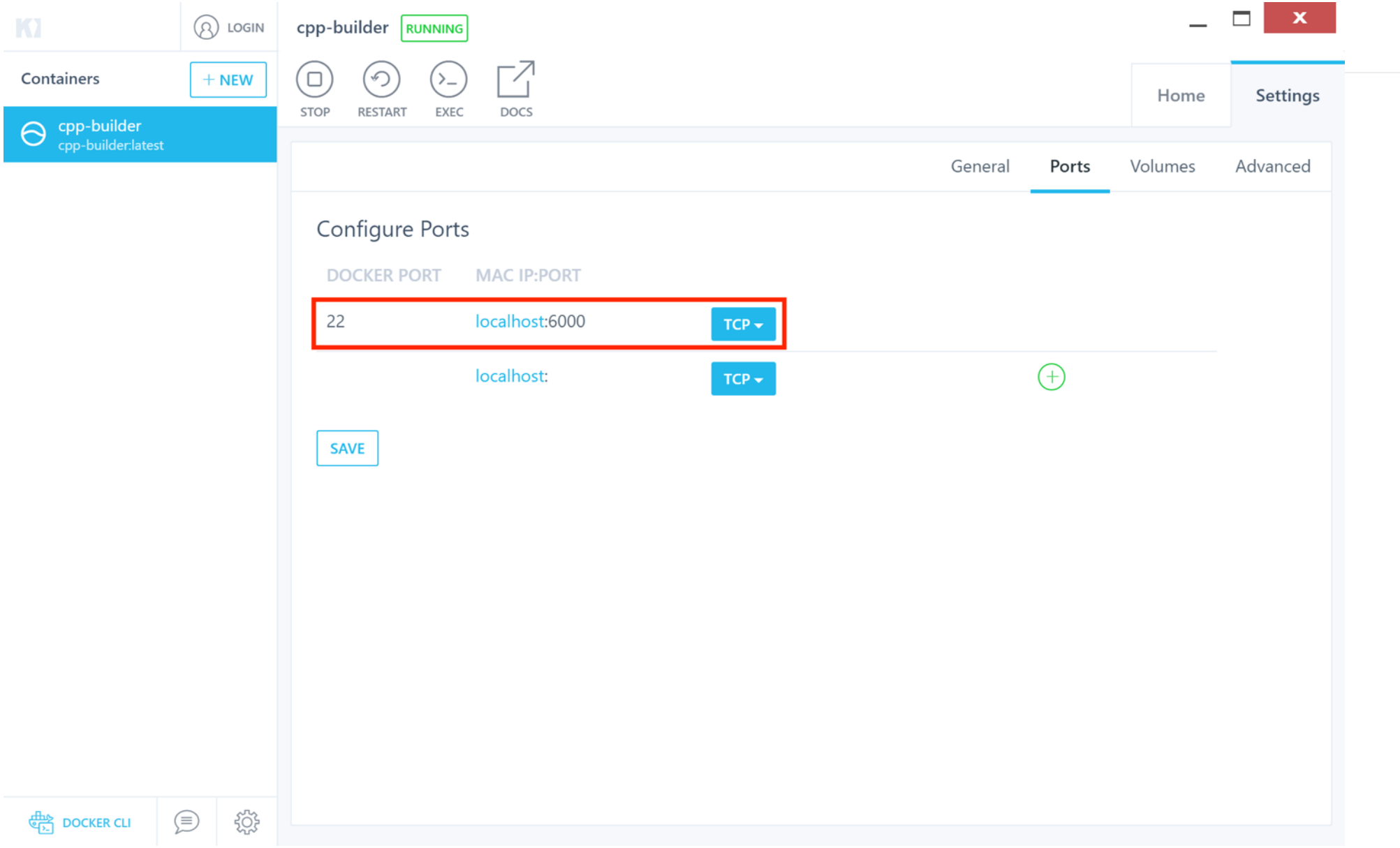


Example:
C++ 프로젝트에 디버거를 붙여봅시다

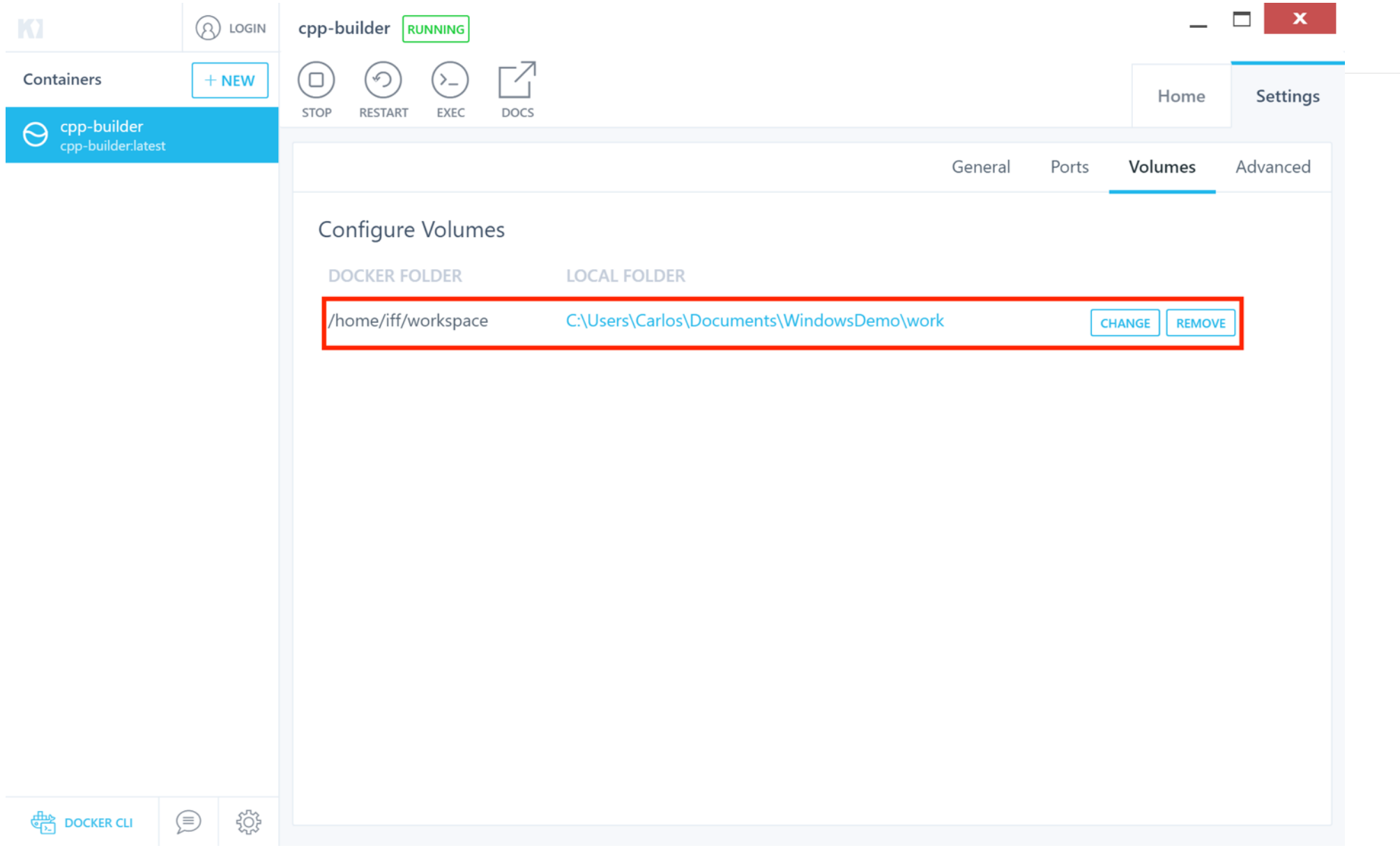
Example

- 필요한 툴들을 넣어서 컨테이너를 띄웁니다 (예제로 사용한 Dockerfile)
- 이미지에는 g++, gdb… 등을 포함
- Kitematic 에서 디버깅을 위한 설정
- VisualStudio 에서 빌드/디버깅/소스코드 위치를 하나씩 지정 (…)

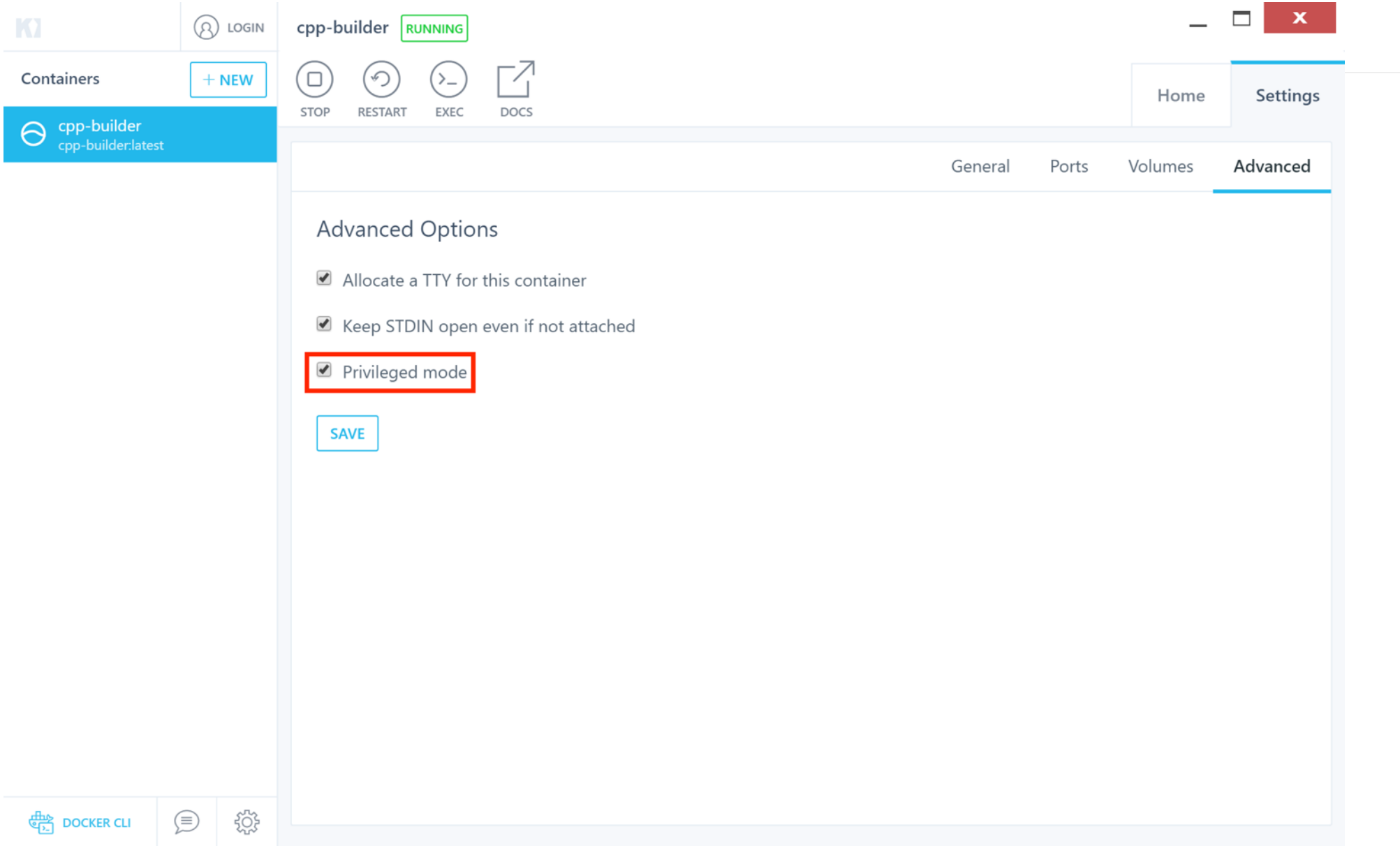
Example: Kitematic 설정 (ports)



Example: Kitematic 설정 (volume)



Example: Kitematic 설정 (advanced)

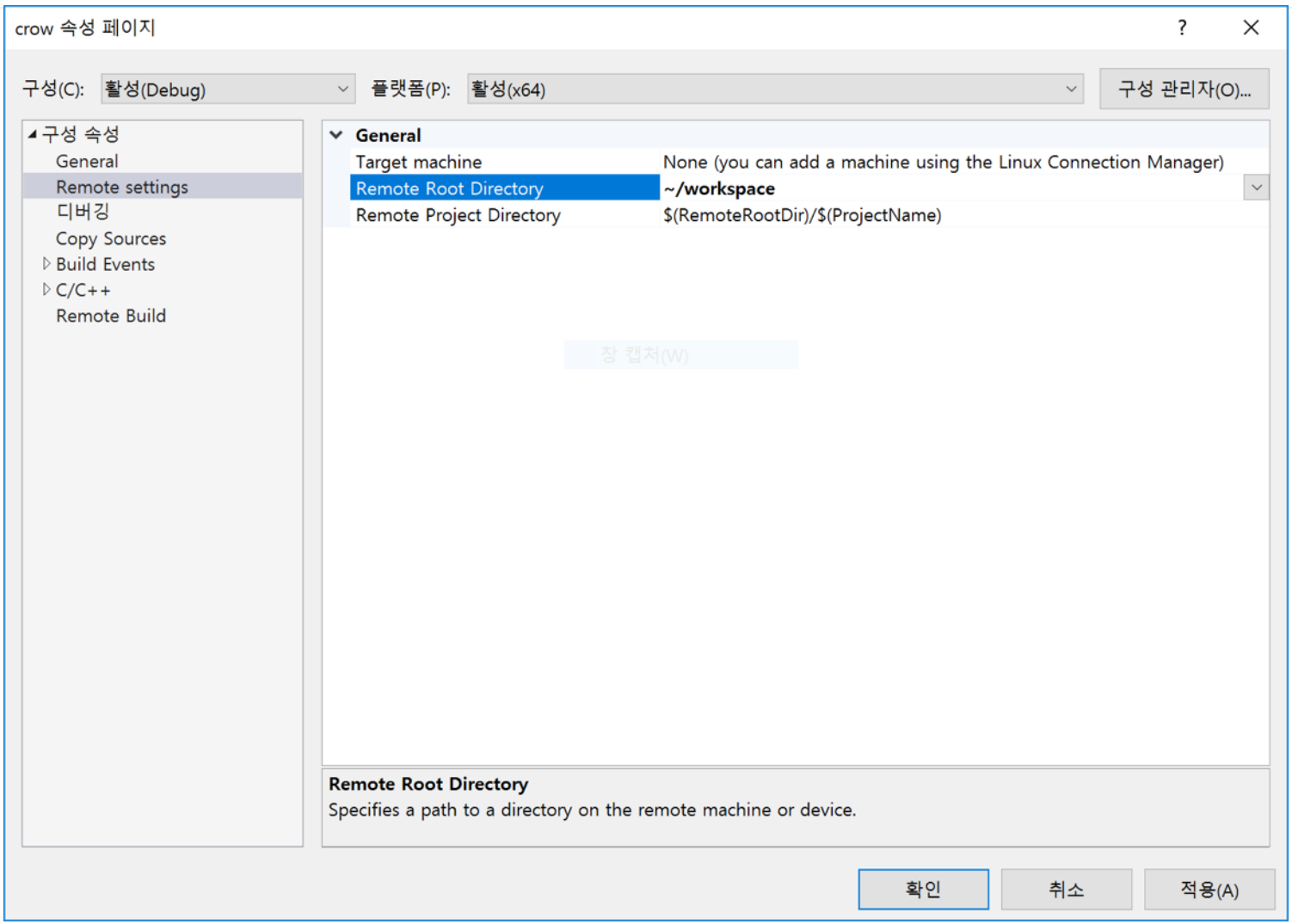


Example: VS 설정

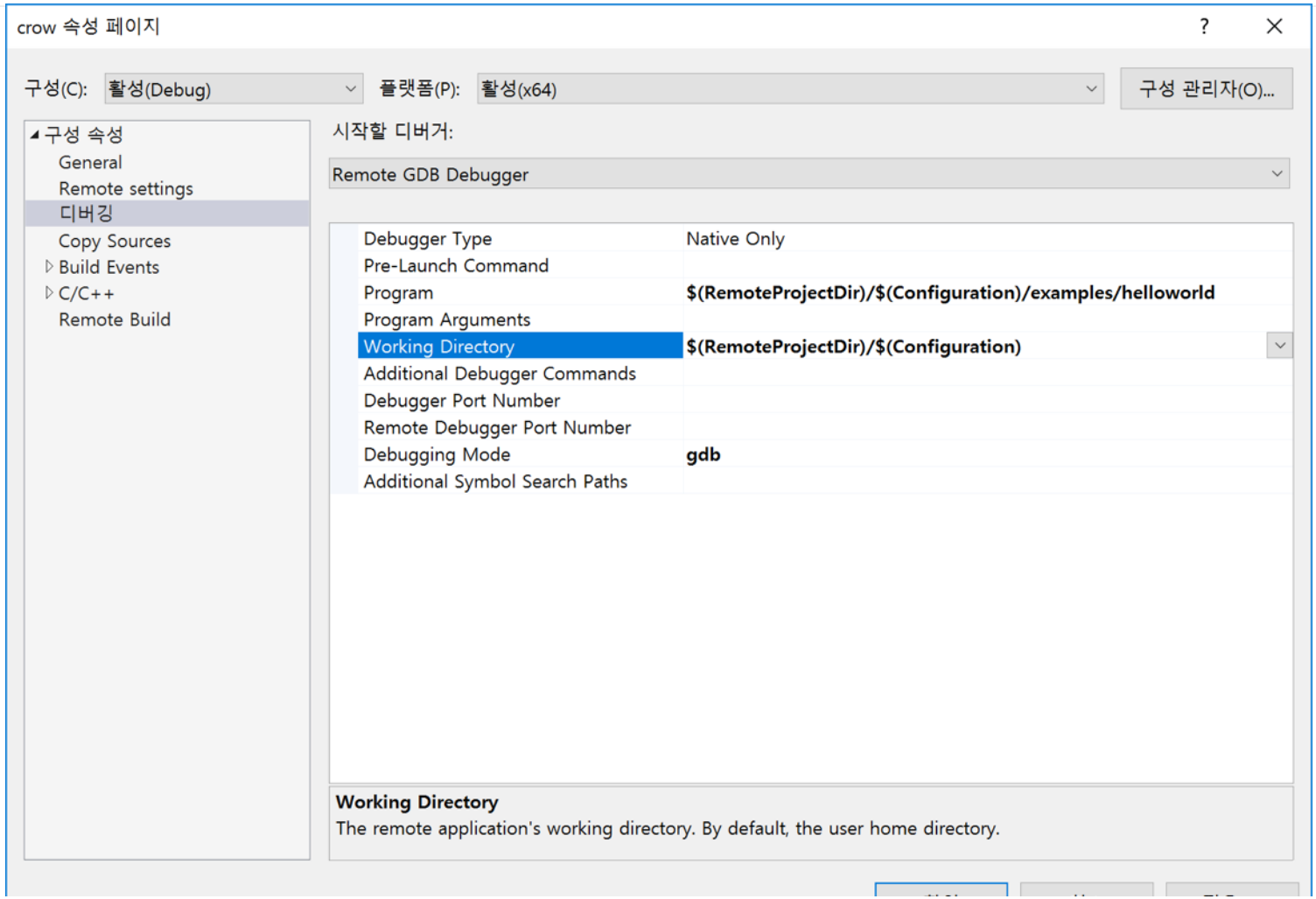
원격 프로젝트의

- 소스코드 디렉터리
- 디버깅 대상 (실행 파일, 인자, CWD, …)
- 서버로 소스코드 복사해야할지 여부
- 빌드 전에 할 작업
- 빌드 / 초기화 / 다시 빌드 명령어

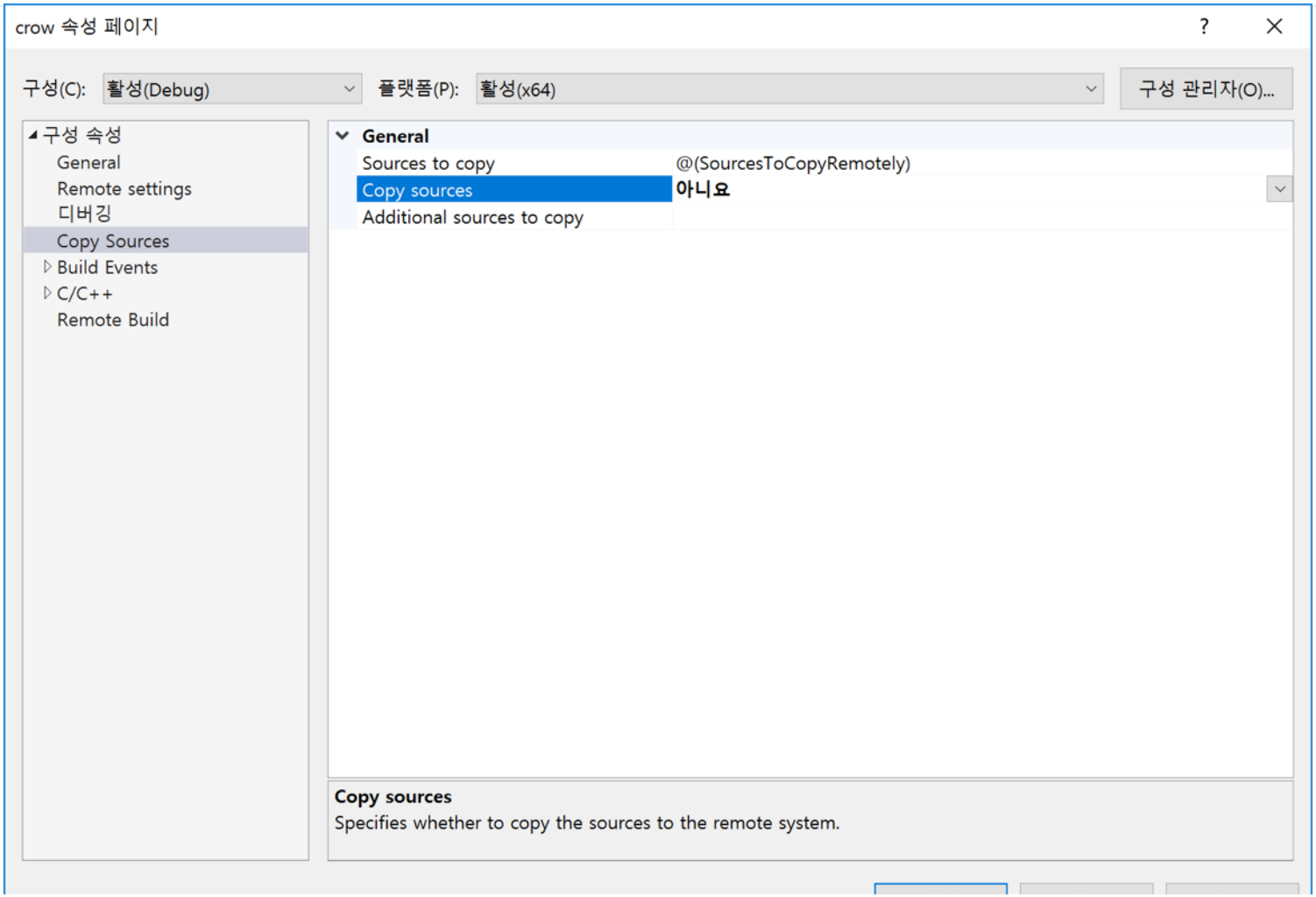
Example: VS 설정 (1/5)



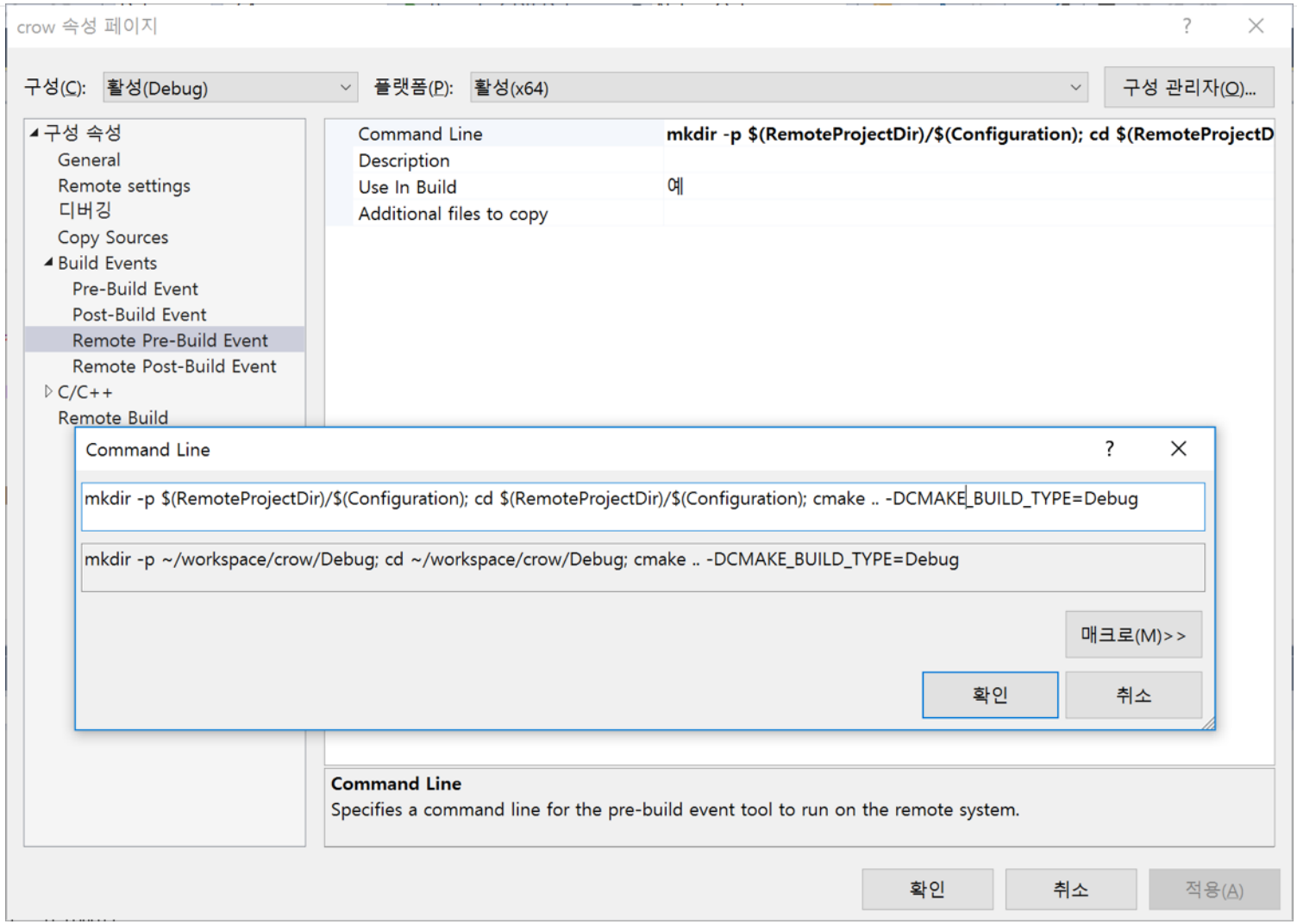
Example: VS 설정 (2/5)



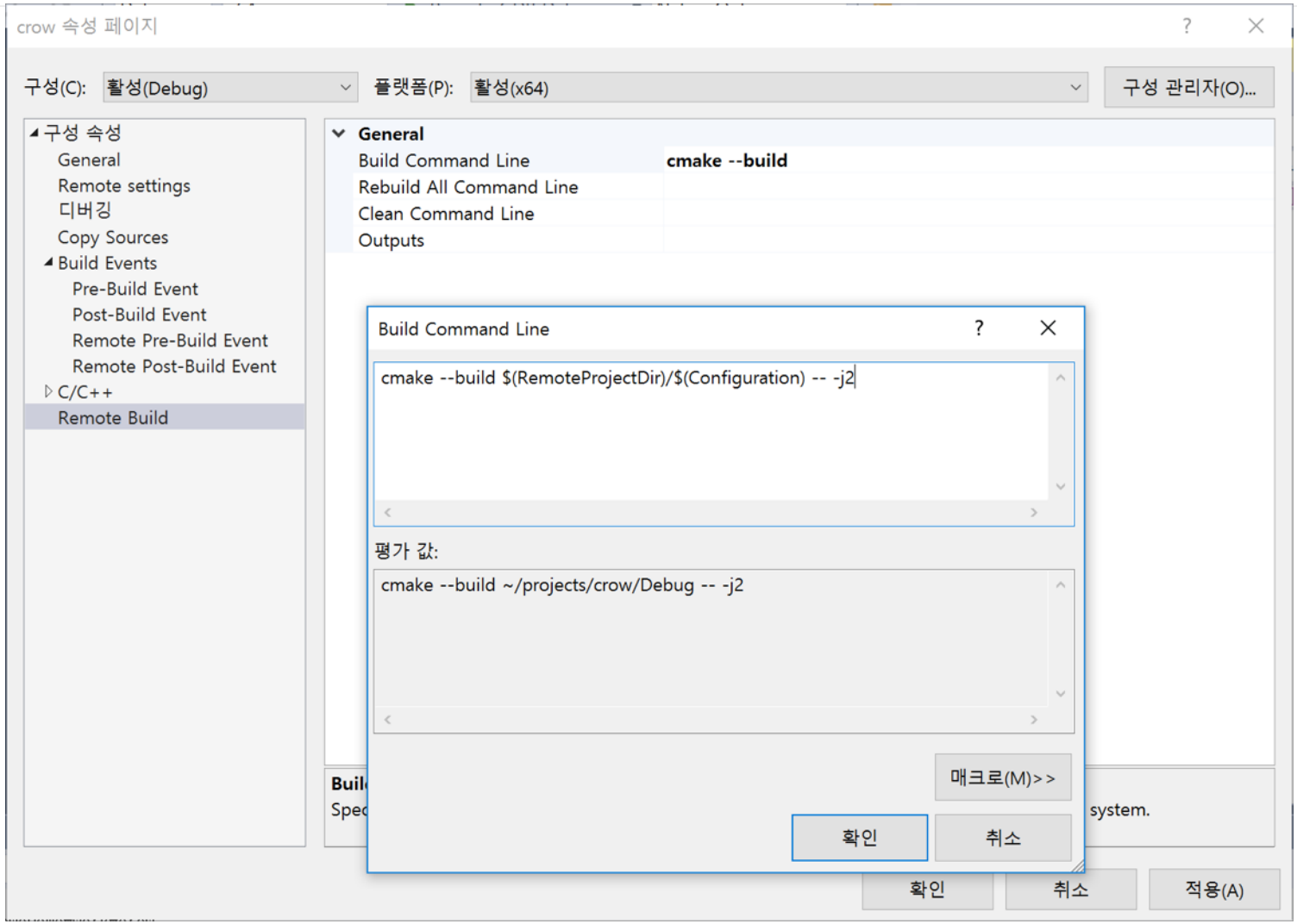
Example: VS 설정 (3/5)



Example: VS 설정 (4/5)

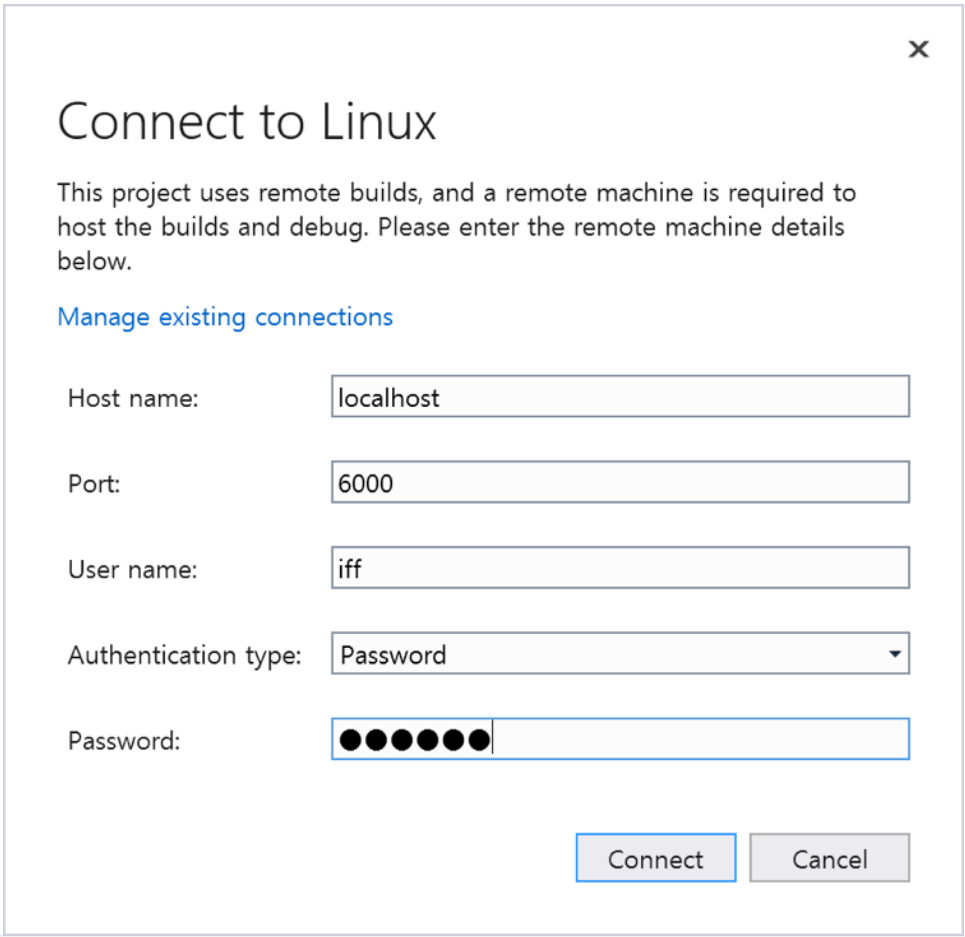


Example: VS 설정 (5/5)



Example: Linux 서버 연결 설정

처음 빌드할 때 서버와 어떻게 연결할지 지정



아직 해결 안된 것

- Visual C++ for Linux 가 아직 beta
- 빌드 / 디버깅 명령을 하나씩 지정해줘야 하는 문제
- Docker for Windows 쪽에서 공유 드라이브 설정을 꼭 해줘야 volume 이 동작

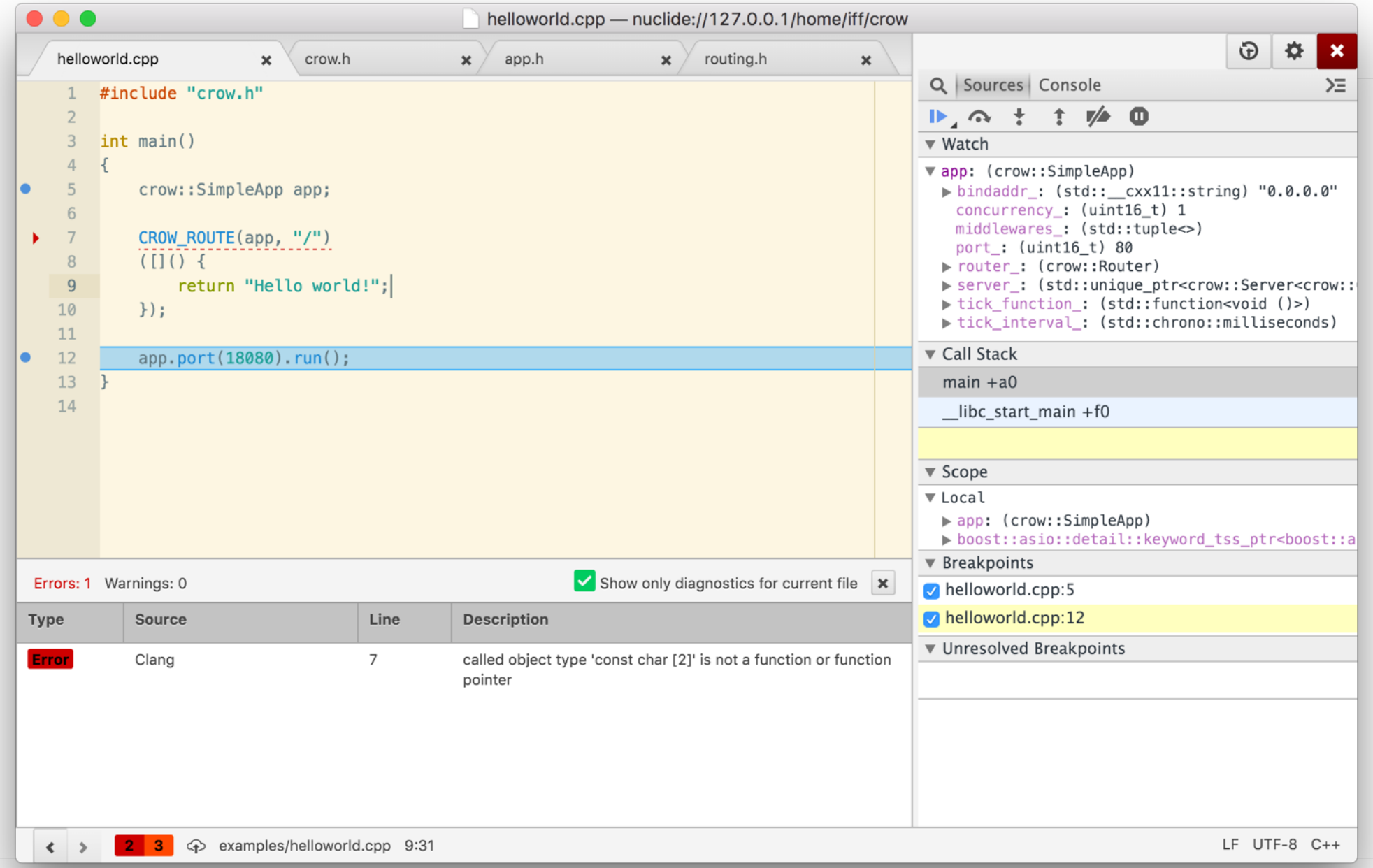
광고: 아이펀엔진

- 아이펀엔진도 VS 확장 기능을 써서 개발 가능
- VisualStudio 확장 기능을 한 번 감싸서 디렉터리/빌드/디버거 설정 등을 자동으로 처리

비슷하게, 다른 프로젝트에서도 내부 설정을 위한 플러그인을 만들면 해결됩니다 (?)

04 – 개발 환경도 GUI를 쓰고 싶습니다
On macOS

macOS: Nuclide in Action

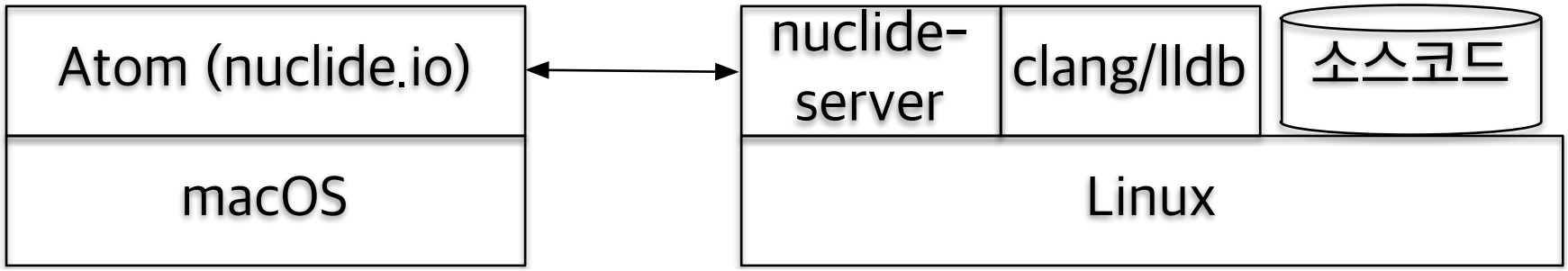


macOS: Nuclide

- Facebook 의 OpenSource IDE (macOS / linux)
- Atom 에 IDE 기능을 추가
- 원격 편집, 디버거 (lldb) 연동 지원, 일부 환경의 빌드
- C++ / Hack / Python / JavaScript / … 언어 지원
- Native app. / ReactNative / Android / iOS 지원

어떻게 동작하는가?

- IDE가 원격 서버와 SSH / JSON RPC 로 통신
- 원격 서버의 buck 을 이용해서 빌드
- lldb 를 이용해서 디버깅
- 소스코드는 원격 서버에 있는 시나리오

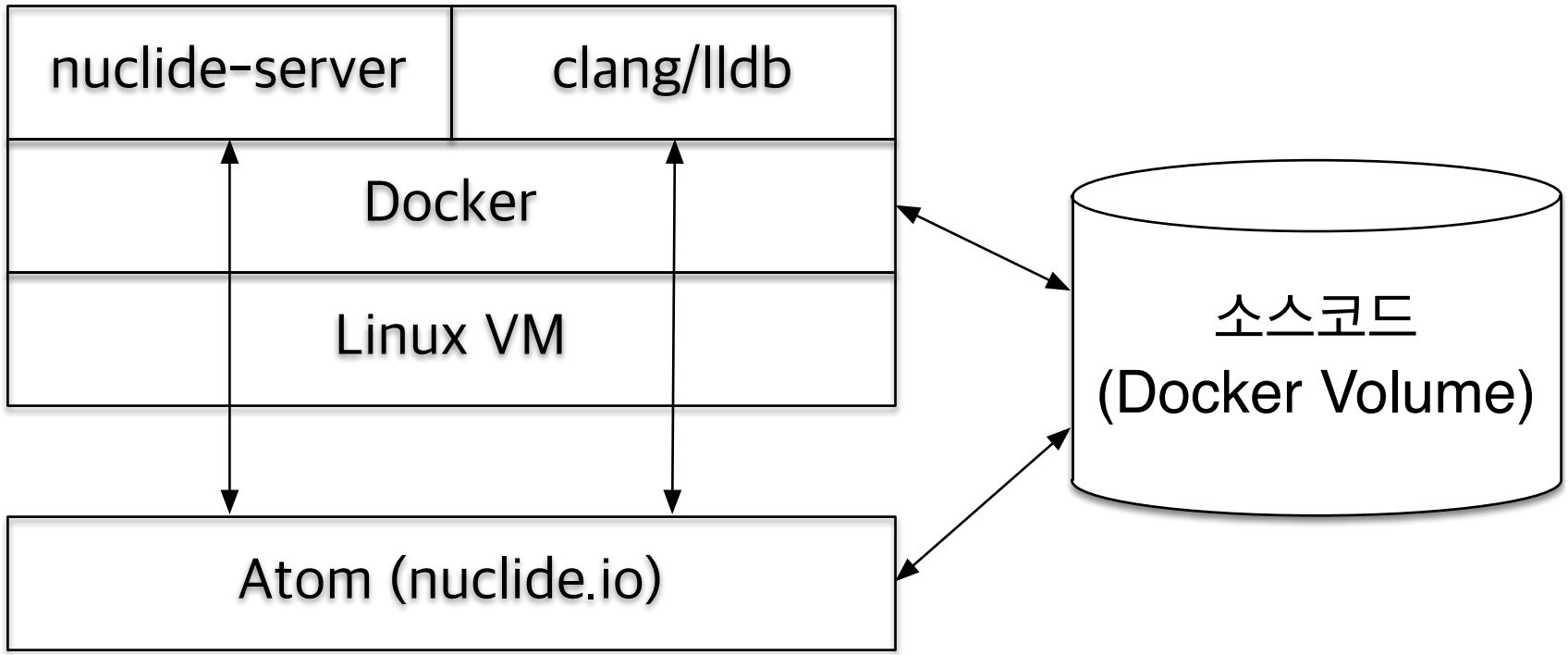


컨테이너로 바꿉시다

- `node.js`, `python`, `clang`, … 을 설치
- 여기서도 SSH 서버가 필요
- SSH용 22, 디버거가 사용할 9090-9093 포트 준비
- 소스코드를 공유할 volume 정의

macOS: Nuclide 설정 (1)

- 소스 코드는 docker volume 으로 공유한다
- Nuclide에서 준비한 컨테이너에 연결해서 작업한다



macOS: Nuclide 설정 (2)

- Docker 컨테이너 쪽에 SSH 공개키 로그인하게 수정
- Atom 의 원격 프로젝트 (remote project) 로 docker 쪽의 주소를 추가해준다
- 현재는 아래 기능이 원격으로 동작한다:
  - 디버깅 (lldb)
  - 코드 분석 (clang 기반)
  - 빌드: buck 을 쓰는 경우만

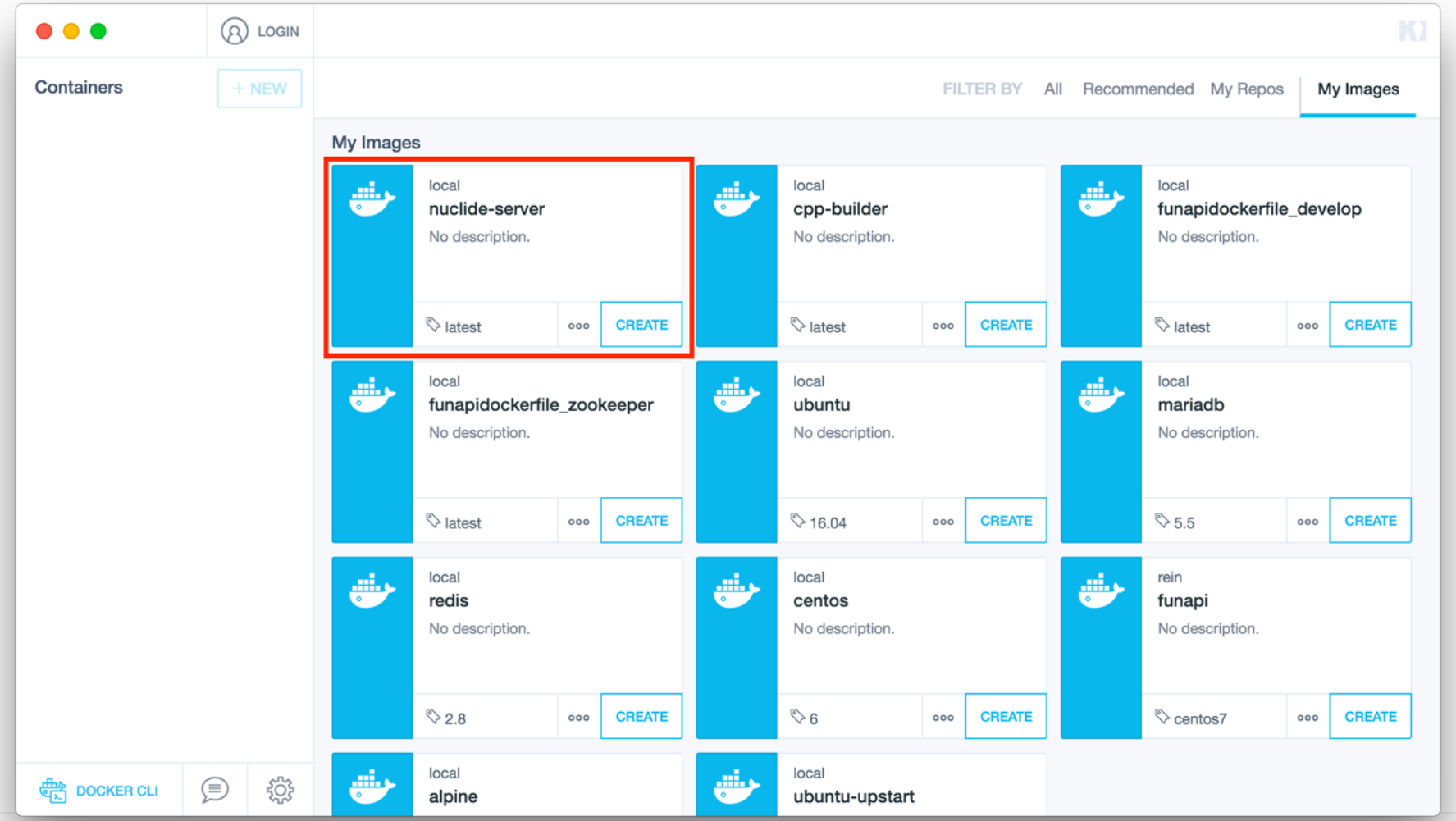
macOS: misc.

- Docker 가 디버거에 필요한 일부 권한을 컨테이너에게 주지 않는다: 컨테이너를 privileged 상태로 실행
- Nuclide 가 디버거/분석 툴과 통신하는 port (9090-9093) 는 호스트 쪽에도 해당 포트로 열려있어야 정상 동작한다

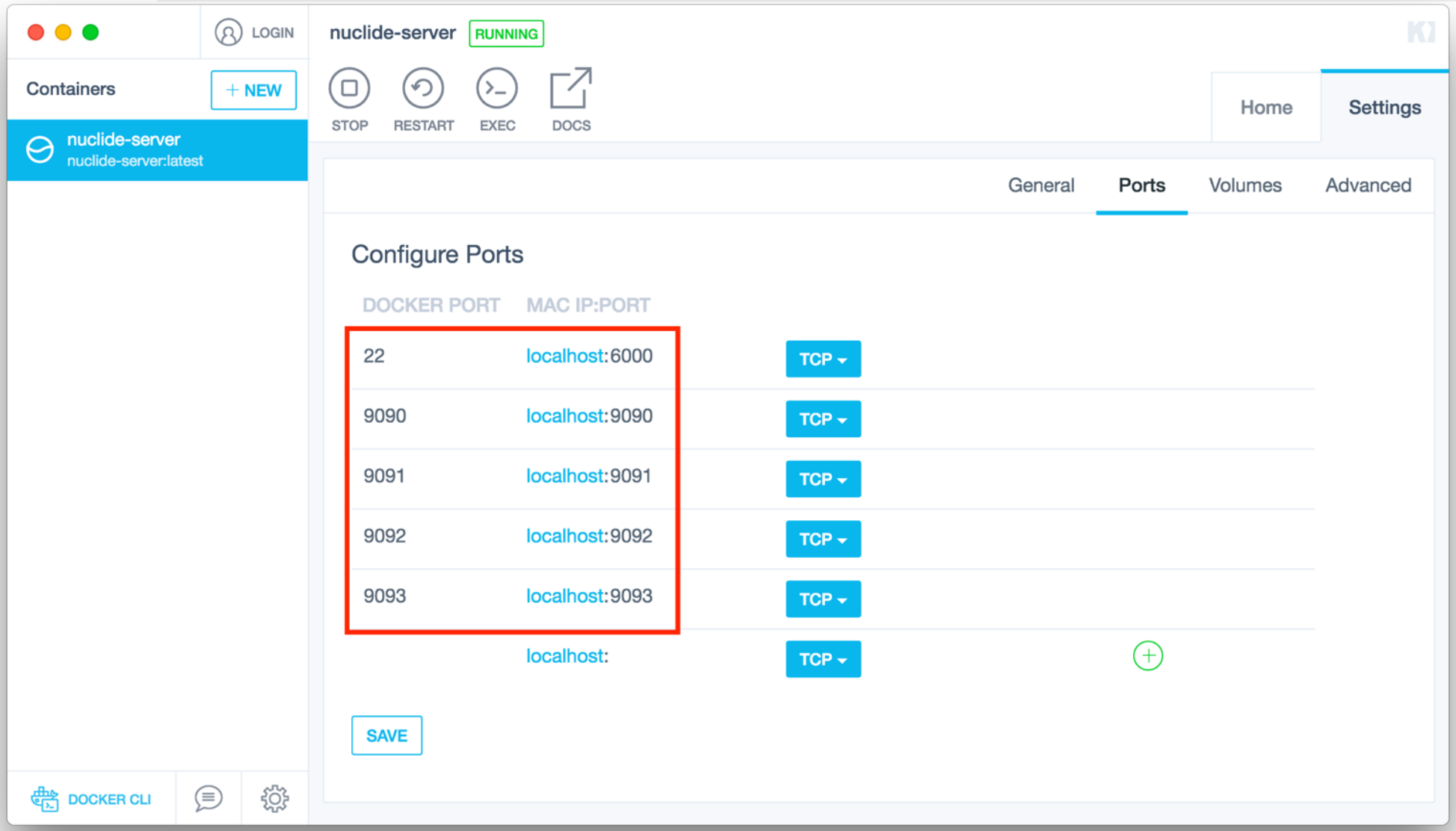
Example:
C++ 프로젝트에 디버거를 붙여봅시다

Example: 컨테이너 생성

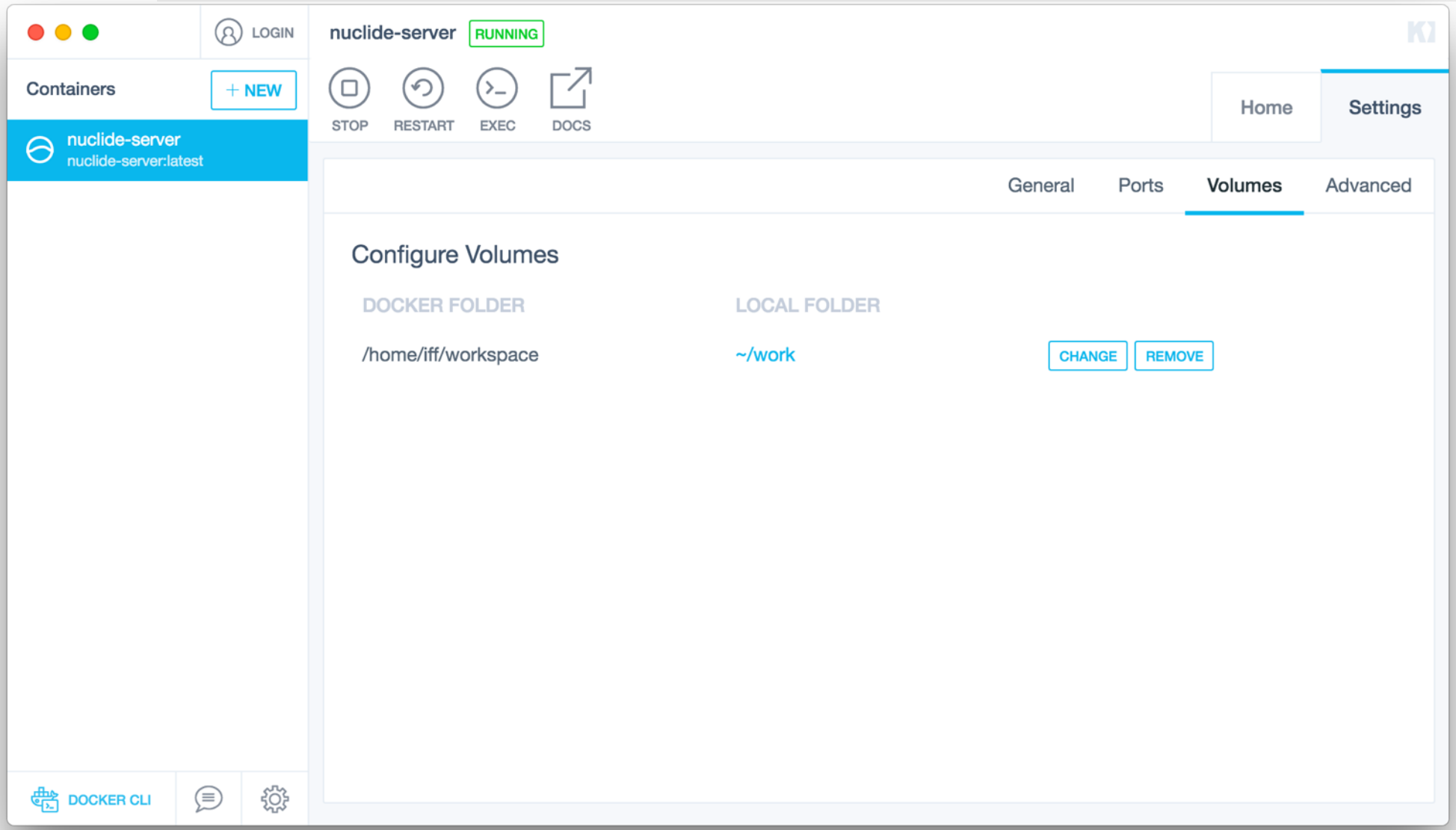
- 앞에서 설명한 컨테이너를 띄웁니다 (Dockerfile)



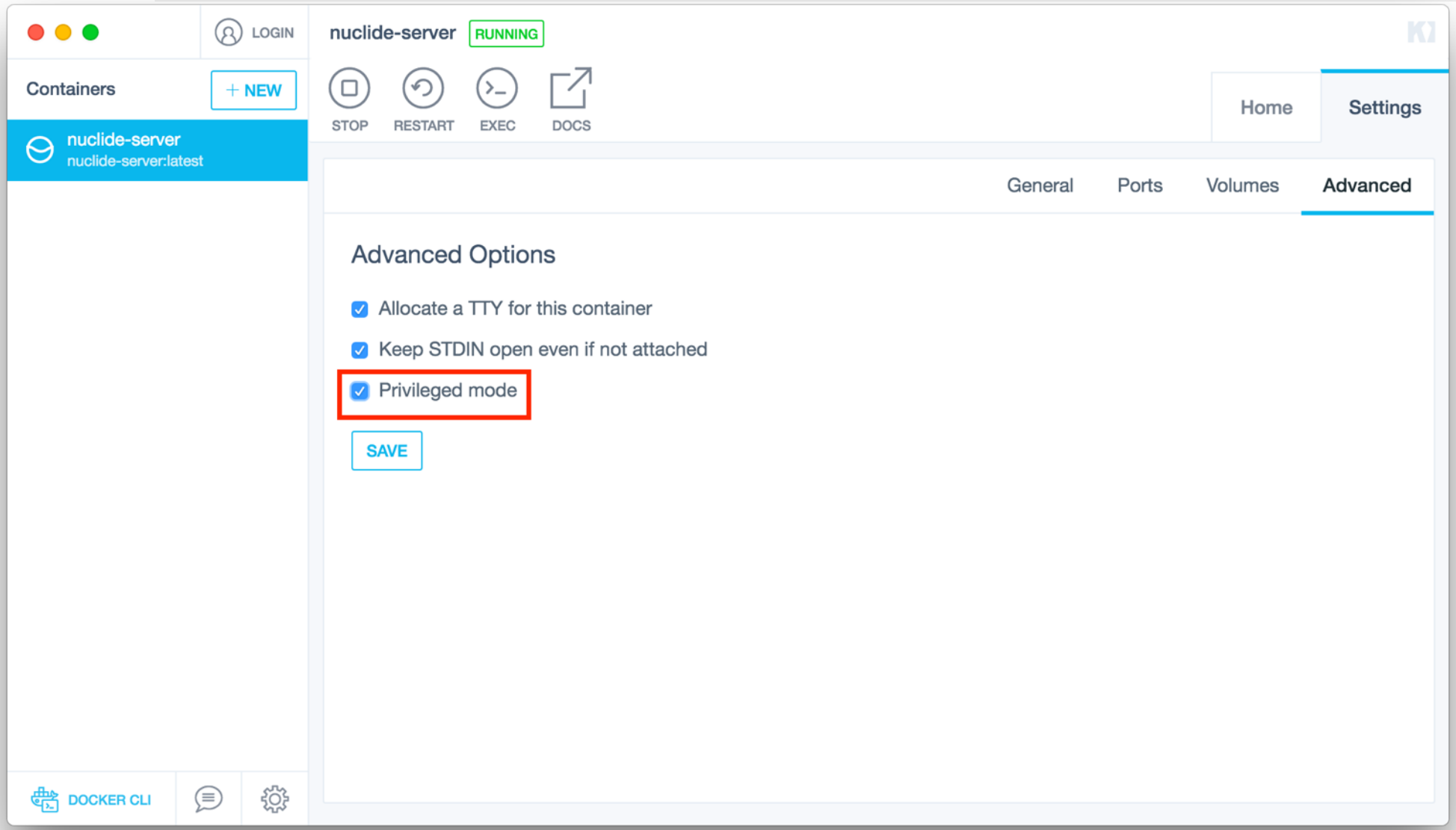
Example: Kitematic 설정 (ports)



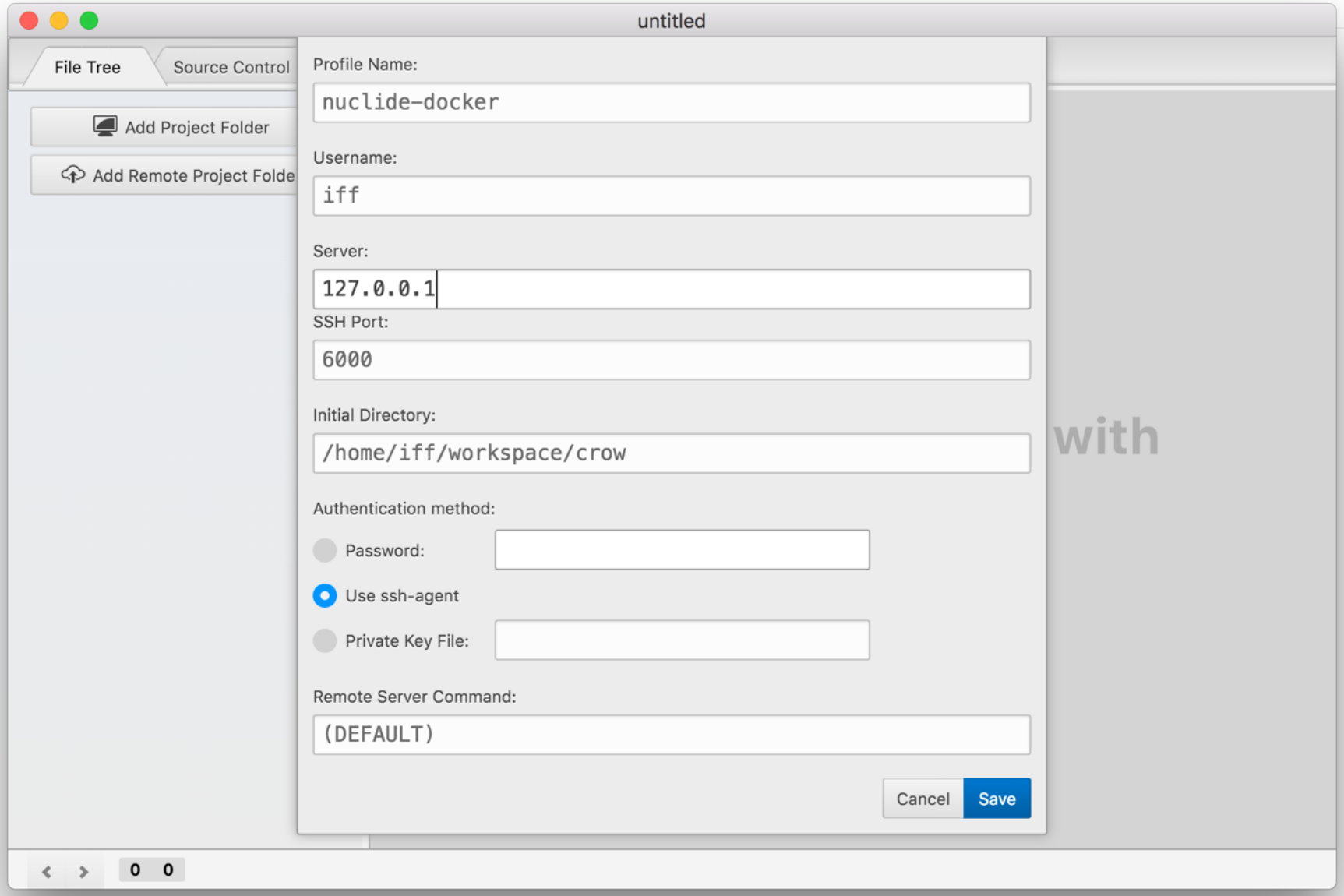
Example: Kitematic 설정 (volume)



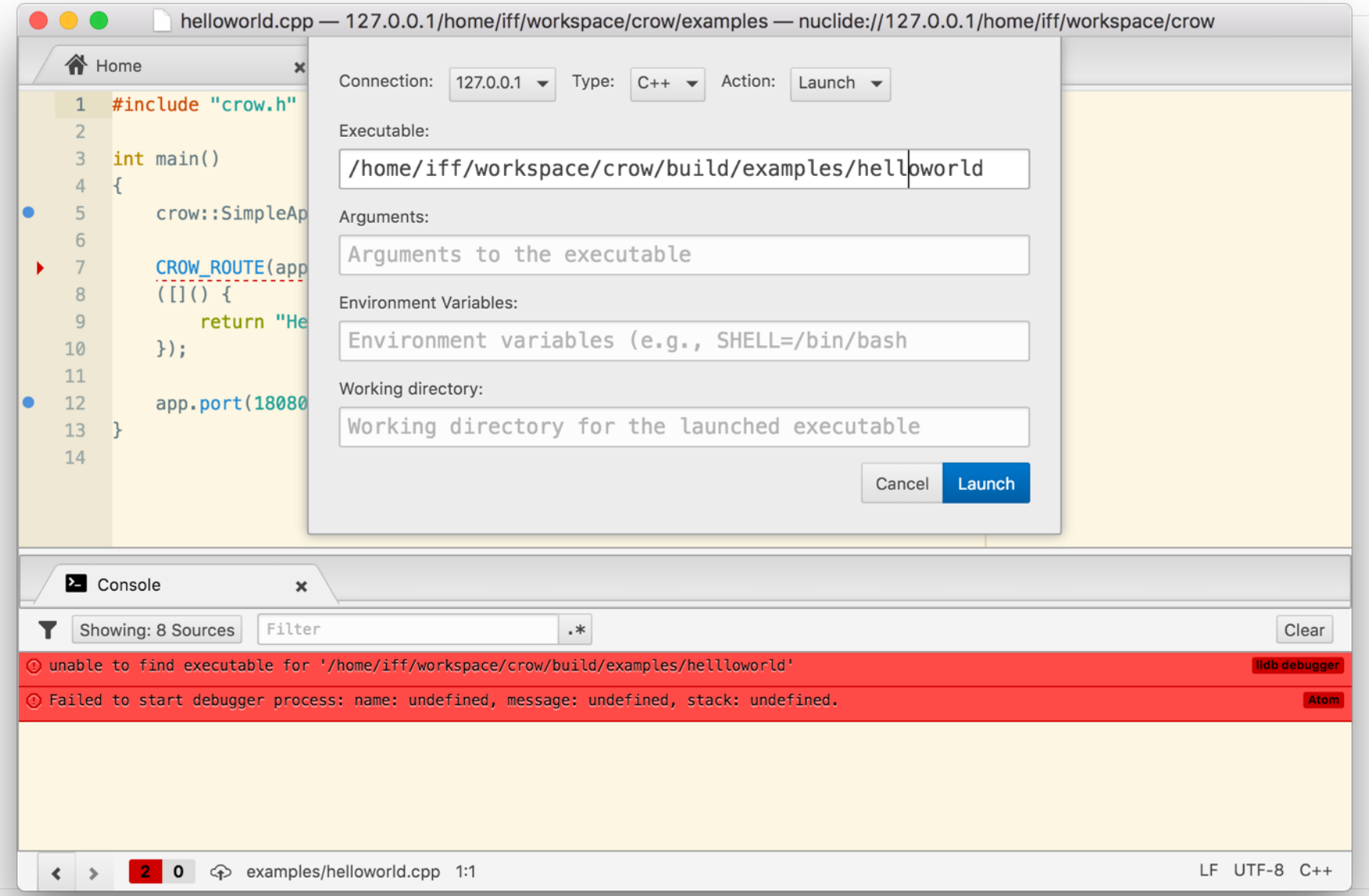
Example: Kitematic 설정 (advanced)



Example: Nuclide 설정 (remote)



Example: Nuclide 설정 (debugger)



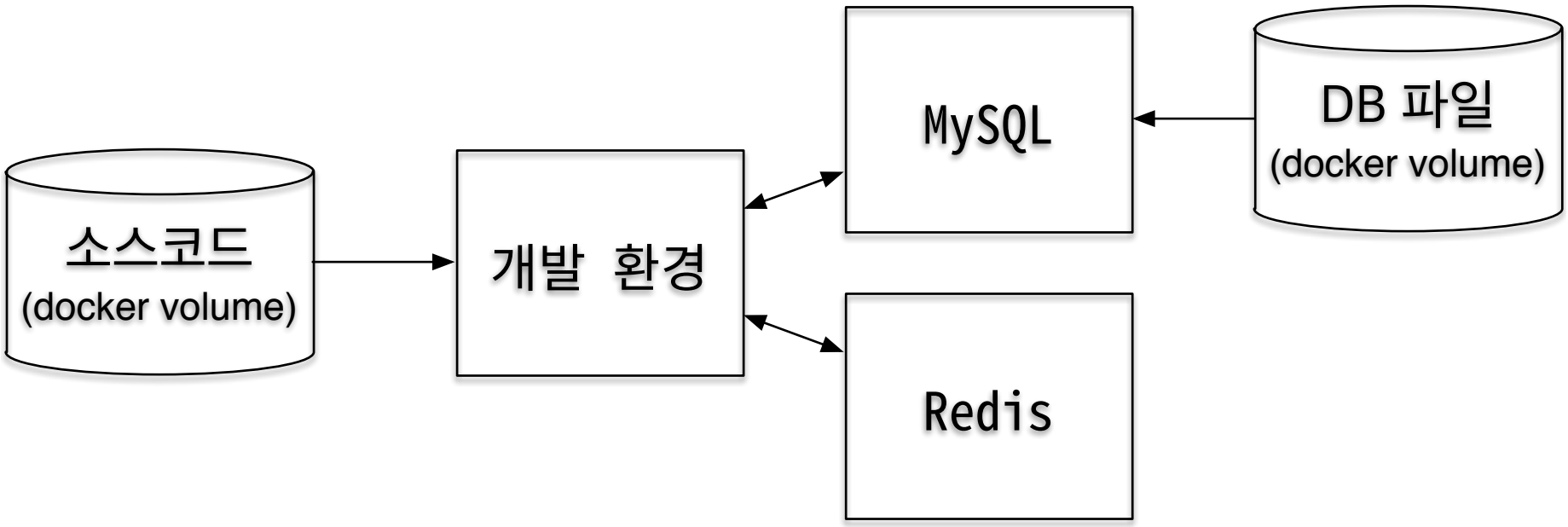
미묘한 부분들

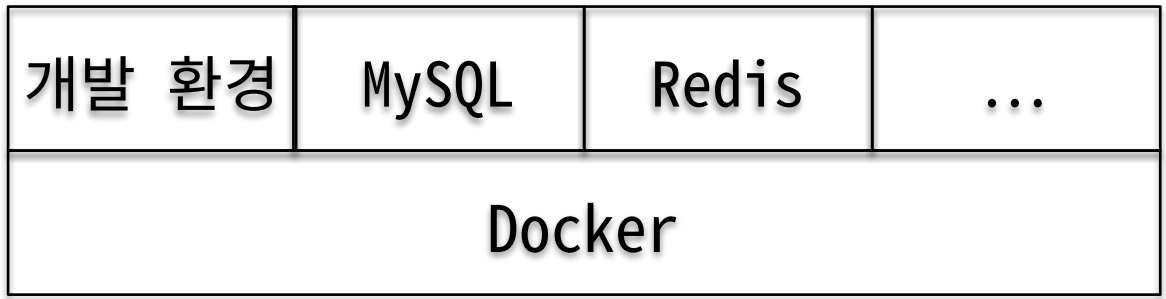
- clang 분석 기능
  - 헤더 검색 경로에 컴파일러 기본 경로가 미탑재
  - Buck 을 안 쓰면 표준 헤더에서 경고 띄울 때가 있다
- clang/llvm 배포판에 따라 동작 안하는 문제
  - libclang.so, libLLVM.so … 등이 경로를 수동으로 수정해줘야 한다

05 – 실제 개발 환경은 어떻게?

외부 서비스를 포함한 구성 지원

Docker Compose





Docker Compose

- 여러 개의 docker 컨테이너를 설정하기 위한 용도
- 하나의 compose 설정은 여러 개의 서비스를 정의:
  - 서비스 = docker 컨테이너
  - 각 서비스가 포트, 볼륨, … 등을 노출한다
  - 각 서비스가 서로 다른 docker 이미지를 이용해서 서비스를 정의한다

Example: 설정

- 각각의 서비스를 지정
- 필요한 포트를 열고
- 유지할 데이터를 볼륨으로 컨테이너에 붙이고
- 각 컨테이너는 links 로 나열한 서비스를 DNS 로 접근해서 사용

```
services:
  redis:
    hostname: redis
    image: redis:2.8
    expose:
      - 6379

  mysql:
    hostname: mysql
    image: mariadb:5.5
    volumes:
      - ./mysql-data:/var/lib/mysql
    expose:
      - 3309

  develop:
    hostname: develop
    build: base-dev
    ports:
      - "22"
    links:
      - redis
      - mysql
    volumes:
      - /Users/rein/work/pong:/home/iff/work
```

Example: 컨테이너 띄우기

- compose 설정 + 개별 이미지 설정
- `docker-compose up -d`: 컨테이너를 모두 띄운다
  - 이미지가 없다면 개별 이미지 설정 써서 빌드
  - 이미지가 있다면 해당 이미지를 빌드

TIPS: DockerHub 컨테이너 쓰기

- 이미 만들어놓은 이미지는 쉽게 띄울 수 있다
- MySQL / MariaDB 혹은 redis 처럼 널리 쓰는 서비스는 공식 이미지가 존재한다
- 해당 이미지를 compose 설정 파일에 넣어주면 (자동으로 가져와서) 컨테이너로 띄워준다

06 – 테스트 환경 만들기

SW 엔지니어 작업 환경을 넘어서기

개발 환경에서 테스트 환경으로

- SW 엔지니어가 아닌 사람에게도 독립된 환경을 제공하려면?
- 문제:
  - 어떻게 docker 컨테이너를 배포할까?
  - 제한된 자원으로 어떻게 서비스를 여러개 띄울까?

Docker Swarm

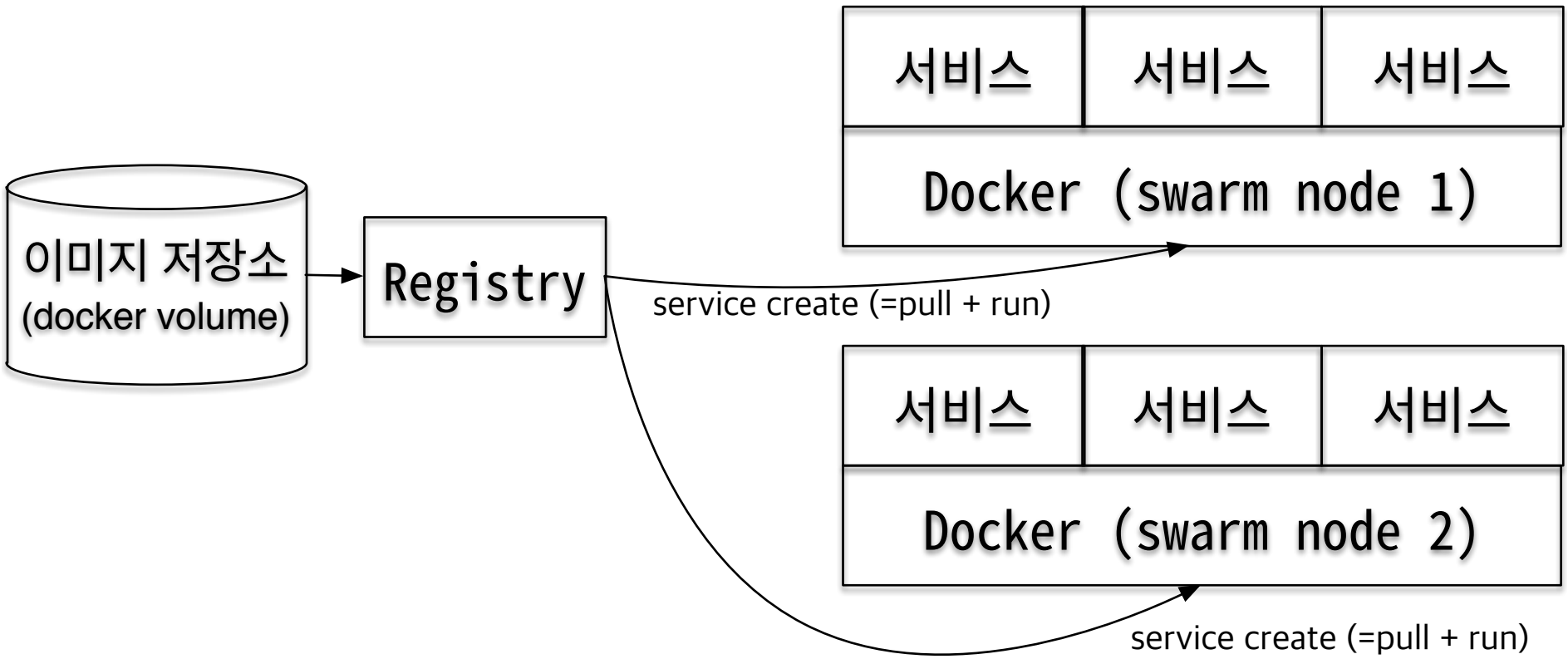
- Docker 를 여러 대의 서버에 나눠서 띄워주는 환경
- 서비스: swarm에 떠 있는 컨테이너 (…보다는 복잡)
- docker service create …
  - swarm 어딘가에 서비스를 생성한다
  - 따로 사용할 수 만큼 (이름을 붙여서) 생성

Docker Registry

- Docker 이미지들을 보관/변경/얻어오는 저장소
- Docker 이미지 형태로 제공된다 (예: registry:2)
  - 이미지를 실제로 저장할 volume 필요
  - HTTP 요청을 보호할 TLS 인증서 필요*

```
docker run -d -p 5000:5000 \
    -v /path/to/volume:/var/lib/registry \
    -v /path/to/cert:/certs
```

Docker Swarm + Registry



Q&A

Great Technology For Great Games, iFunFactory

iFunFactory

jinuk.kim@ifunfactory.com

www.ifunfactory.com

+82-70-4923-6566

THANKS!

Great Technology For Great Games, iFunFactory

iFunFactory

jinuk.kim@ifunfactory.com

www.ifunfactory.com

+82-70-4923-6566