

Linux 없이 Linux 에서 개발하기

Great Technology For Great Games



김진욱

jinuk.kim@ifunfactory.com



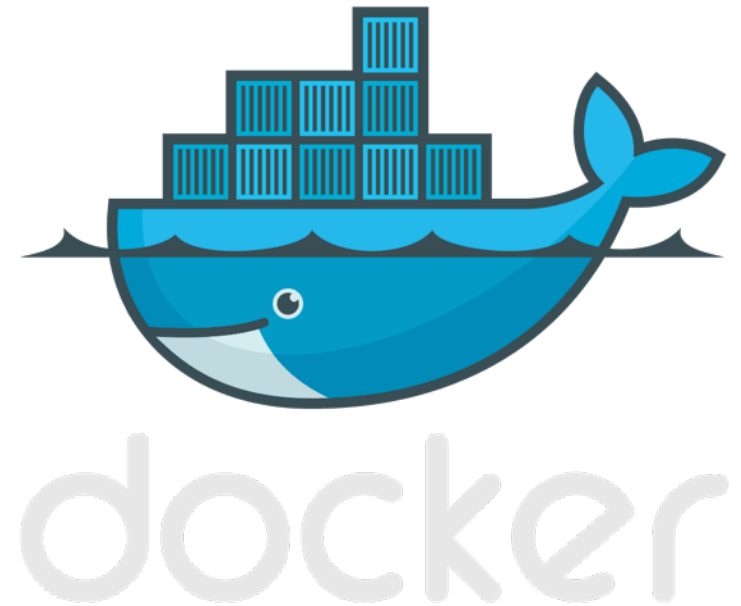
개발 환경: 원하는 것

- 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

서비스	서비스				
Guest OS	Guest OS	서비스	서비스	서비스	서비스
하이퍼바이저	하이퍼바이저	Docker			
OS					
서버 하드웨어					

가벼운 디스크 형식



공통 이미지를 수정하면 변경 사항이 덧 씌워지는 방식

The screenshot shows the Docker Hub interface with a search bar and a grid of recommended Docker images. The interface includes a 'Containers' sidebar, a search bar, and a 'FILTER BY' dropdown menu. The recommended images are:

Image Name	Description	Downloads	Actions
kitematic/hello-world-nginx	A light-weight nginx container that demonstrates the features of Kitematic	58 ↓ 744K	CREATE
ghost/official-ghost	Ghost is a free and open source blogging platform written in JavaScript	448 ↓ 2M	CREATE
jenkins/official-jenkins	Official Jenkins Docker image	2.0K ↓ 10M	CREATE
redis/official-redis	Redis is an open source key-value store that functions as a data structure server.	2.8K ↓ 117M	CREATE
rethinkdb/official-rethinkdb	RethinkDB is an open-source, document database that makes it easy to build and scale realtime...	319 ↓ 4M	CREATE
kitematic/minecraft	The Minecraft multiplayer server allows two or more players to play Minecraft together	67 ↓ 30K	CREATE
solr/official-solr	Solr is the popular, blazing-fast, open source enterprise search platform built on Apache...	265 ↓ 436K	CREATE
elasticsearch/official-elasticsearch	Elasticsearch is a powerful open source search and analytics engine that makes data easy to...	1.6K ↓ 20M	CREATE
postgres/official-postgres	The PostgreSQL object-relational database system provides reliability and data integrity.	2.6K ↓ 20M	CREATE
ubuntu-upstart/official-ubuntu-upstart			
memcached/official-memcached			
rabbitmq/official-rabbitmq			



OS 지원

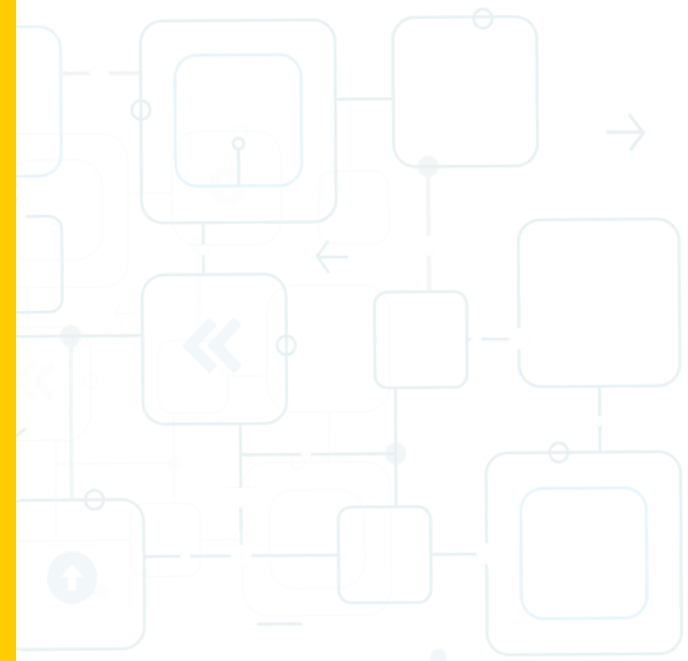
- Windows / macOS 정식 지원 시작 (2016-07)

[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)

C++ 프로젝트를 빌드하는 Docker 컨테이너를 띄워봅시다





환경 설정: Dockerfile

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



Example: 이미지 만들기

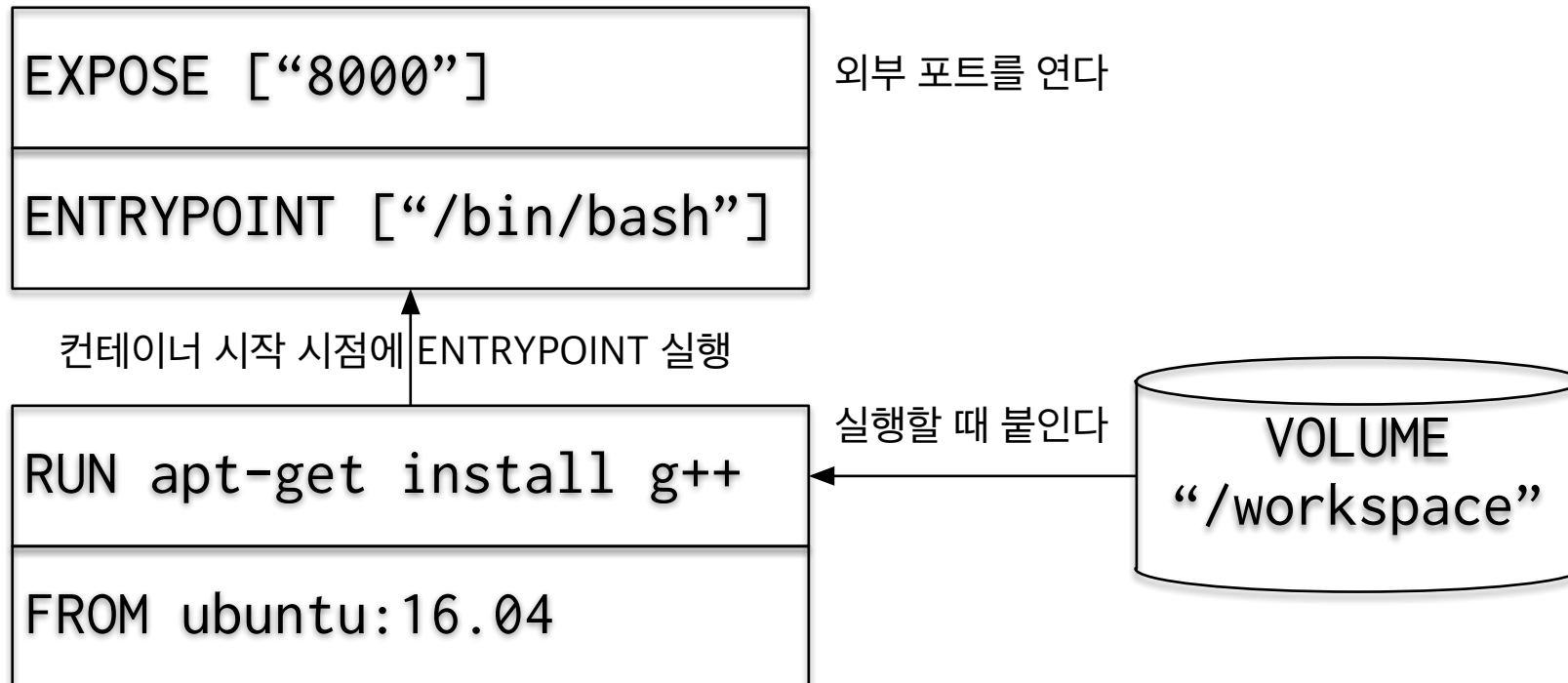
- Ubuntu 16.04 (xenial)
- g++, cmake 를 설치
- /workspace 공유 디렉터리
- 8000 포트를 연다

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:

```
1 # 데스크탑 혹은 랩탑에서 실행
2 $ docker run -it --rm -v ~/work/crow:/workspace cpp-builder
3 # 여기서부터는 docker 컨테이너 안
4 $ apt-get update
5 $ apt-get install -y libgoogle-perftools-dev libssl-dev libboost-all-dev
6 $ cd /workspace
7 $ mkdir build; cd build; cmake ..; make -j2
```

02 - GUI를 쓰고 싶습니다



Remind:

- 명령행에서 작업하는 걸 피하고 싶다
- GUI 툴은 없을까?

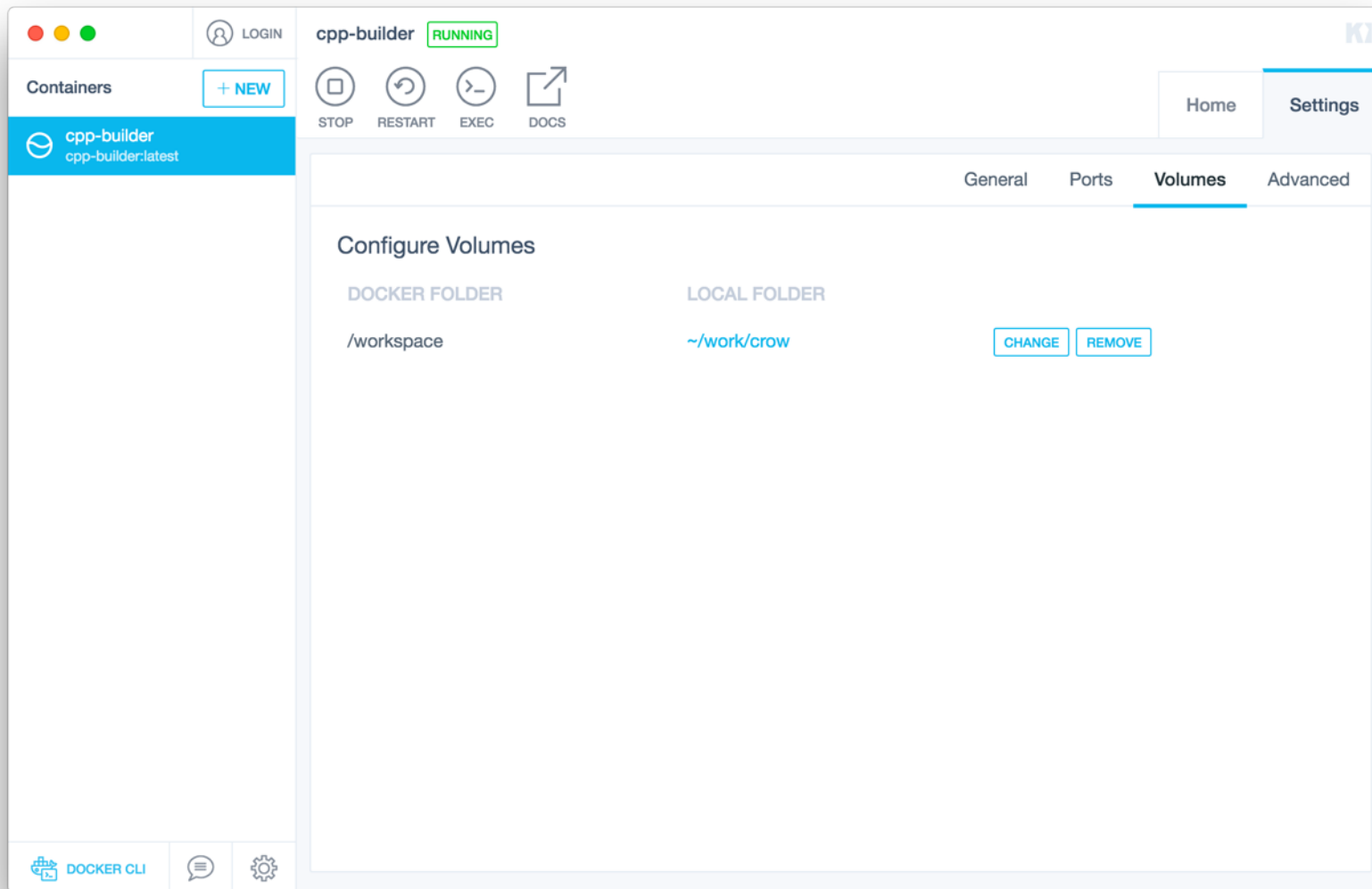
Kitematic: docker run



The screenshot shows the 'My Images' section of the Kitematic interface. It features a grid of Docker images, each with a blue background and a white Docker logo icon. The images are arranged in three rows and three columns. The first image in the top row, 'local cpp-builder', is highlighted with a red border. Each image card displays the name, a description (or 'No description.'), a tag (like 'latest' or '6'), and a 'CREATE' button. The interface also includes a 'CONTAINERS' sidebar on the left, a 'LOGIN' button, and a 'FILTER BY' dropdown menu at the top right.

Image Name	Description	Tag	Action
local cpp-builder	No description.	latest	CREATE
local xenial-builder	No description.	latest	CREATE
local ubuntu	No description.	16.04	CREATE
local <none>	No description.	<none>	CREATE
local mariadb	No description.	5.5	CREATE
local redis	No description.	2.8	CREATE
local centos	No description.	6	CREATE
rein funapi	No description.	centos7	CREATE
local alpine	No description.	latest	CREATE

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)



The screenshot displays the SourceTree Git client interface for a repository named 'funapi-core (Git)'. The interface is divided into several sections:

- Workspace:** Shows the current working directory as 'documents/misc/sodium.md'. The file contents are displayed in a code editor, showing a Makefile for building 'libsodium' on various Linux distributions (Ubuntu 16.04, CentOS 7).
- Branches:** A sidebar on the left shows the local branches: 'doc', 'feature', 'string', 'test-session', 'master' (30 commits), 'support', 'changelog', and 'doc-obj'. The 'origin' remote is also visible with branches '0.X', 'HEAD', 'demo', 'alpha', 'master', 'release', 'alpha', 'replace-zo...', and 'stable'.
- Commit History:** A central panel shows a graph of commits. The 'Uncommitted changes' section lists several commits with their descriptions and commit hashes. The 'Commit' table on the right provides a summary of recent commits, including the author and date.
- Stashes:** A sidebar on the left shows two stashes: 'WIP on mast...' and 'WIP on mast...'.
- Submodules and Subtrees:** Empty sections at the bottom left of the interface.

The commit history table includes the following data:

Commit	Author	Date
*	*	Today, 14:54
fd741af	Lee Eunhee <arin...>	Today, 10:35
f35af6a	Minwoo Seok <mi...>	Sep 23, 2016, 16:...
228d6c0	Seunghyun Nam <...>	Sep 23, 2016, 15:...
04ebf87	Seunghyun Nam <...>	Sep 23, 2016, 15:...
d20b3f4	Seunghyun Nam <...>	Sep 23, 2016, 15:...
bd0a52b	Seunghyun Nam <...>	Sep 23, 2016, 10:...
bb61f69	Seunghyun Nam <...>	Sep 23, 2016, 09:...
8e54b1d	Seunghyun Nam <...>	Sep 22, 2016, 19:...
43a1760	Seunghyun Nam <...>	Sep 22, 2016, 19:...
8ce50ad	Seunghyun Nam <...>	Sep 22, 2016, 17:...
7cc7c9c	Minwoo Seok <mi...>	Sep 22, 2016, 15:...

GitHub Desktop



funapi-core

54 Uncommitted Changes History

Update from origin/stable View Branch Publish

origin/stable
feature/test-session

54 Changes

File	Change	Line	Code
bash/funapi_profile	...	35	@@ -35,40 +35,51 @@ if [[! -e /usr/local/bin/flamegraph.pl]]; then
bash/funapi_profile	...	35	fi
bash/funapi_profile	...	37	if [["\$#" == "2"]]; then
bash/funapi_profile	-	38	TARGET=\$(/usr/bin/pgrep -f "funapi_runner.*\${1}_server.\${2}.*")
bash/funapi_profile	+	38	TARGETS=\$(/usr/bin/pgrep -f "funapi_runner.*\${1}_server.\${2}.*")
bash/funapi_profile	...	39	SERVER=\$1
bash/funapi_profile	...	40	FLAVOR=\$2
bash/funapi_profile	...	41	elif [["\$#" == "1"]]; then
bash/funapi_profile	-	42	TARGET=\$(/usr/bin/pgrep -f "funapi_runner.*\${1}_server.default.*")
bash/funapi_profile	+	42	TARGETS=\$(/usr/bin/pgrep -f "funapi_runner.*\${1}_server.default.*")
bash/funapi_profile	...	43	SERVER=\$1
bash/funapi_profile	...	44	FLAVOR="default"
bash/funapi_profile	...	45	fi
bash/funapi_profile	-	47	if [["\$TARGET" == ""]]; then
bash/funapi_profile	+	47	if [["\$TARGETS" == ""]]; then
bash/funapi_profile	...	48	echo "funapi_runner for \${SERVER}_server.\${FLAVOR} is not running"
bash/funapi_profile	...	49	exit 2
bash/funapi_profile	...	50	fi
bash/funapi_profile	...	51	fi
bash/funapi_profile	+	52	echo "\${SERVER}_server.\${FLAVOR} process found. pid=("\${TARGETS}")"
bash/funapi_profile	...	53	+
bash/funapi_profile	...	52	set -e
bash/funapi_profile	...	53	+
bash/funapi_profile	-	54	FILENAME=perf.\${SERVER}_server.\${FLAVOR}.\$\$.svg
bash/funapi_profile	+	56	RN=\$\$
bash/funapi_profile	+	57	for TARGET in \$TARGETS
bash/funapi_profile	+	58	do
bash/funapi_profile	+	59	if [["\$FILENAME" != ""]]; then
bash/funapi_profile	+	60	echo
bash/funapi_profile	+	61	fi
bash/funapi_profile	+	62	FILENAME=perf.\${SERVER}_server.\${FLAVOR}.\$RN.svg
bash/funapi_profile	...	63	+
bash/funapi_profile	+	64	trap 'on_exit' EXIT
bash/funapi_profile	+	65	function on_exit {

Summary

Description

Commit to feature/test-session

Committed 9/13/16
libsodium 을 쓰기 위해 작업한 내용... Undo

...

- 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



crow (디버깅) - Microsoft Visual Studio

빠른 실행(Ctrl+Q)

파일(F) 편집(E) 보기(V) 프로젝트(P) 빌드(B) 디버깅(D) 팀(M) 도구(T) Android 테스트(S) 분석(N) 창(W) 도움말(H) 로그인

프로세스: [N/A] 수명 주기 이벤트 스레드: [1433] helloworld 스택 프레임: <lambda()>::operator()(void) const

```
helloworld.cpp
crow (전역 범위) main()
1  #include "crow.h"
2
3  int main()
4  {
5      crow::SimpleApp app;
6
7      CROW_ROUTE(app, "/")
8      ([]() {
9          return "Hello world!";
10     });
11
12     app.port(18080).run();
13
14 }
```

131 %

이름	값	형식
app	{...}	crow::SimpleApp
port_		uint16_t
concurrency_		uint16_t
bindaddr_		std::__cxx11::string
router_		crow::Router
tick_interval_		std::chrono::milliseconds
tick_function_		std::function<void()>
middlewares_		std::tuple<>
server_		std::unique_ptr<crow::Server<crow::Crow<>, crow::Socket>

호출 스택

이름	언어
<lambda()>::operator()(void) const(const <lambda()> * const __closure) Line 10	C++
crow::TaggedRule<>::<lambda(const crow::request&, crow::response&>)::operator()(const C++	
std::function_handler<void(const crow::request&, crow::response&), crow::TaggedRule<C++	
std::function<void (crow::request const&, crow::response&>)::operator()(crow::request cor C++	
crow::detail::routing_handler_call_helper::call<crow::detail::routing_handler_call_helper::call C++	
crow::TaggedRule<>::handle(crow::request const&, crow::response&, crow::routing_param C++	
crow::Router::handle(crow::Router * const this, const crow::request & req, crow::response {C++	
crow::Crow<>::handle(crow::request const&, crow::response&)(crow::Crow<> * const this, C++	
crow::Connection<crow::SocketAdaptor, crow::Crow<>>::handle()(crow::Connection<crow C++	

자동 지역 스레드 모듈 조사식 1

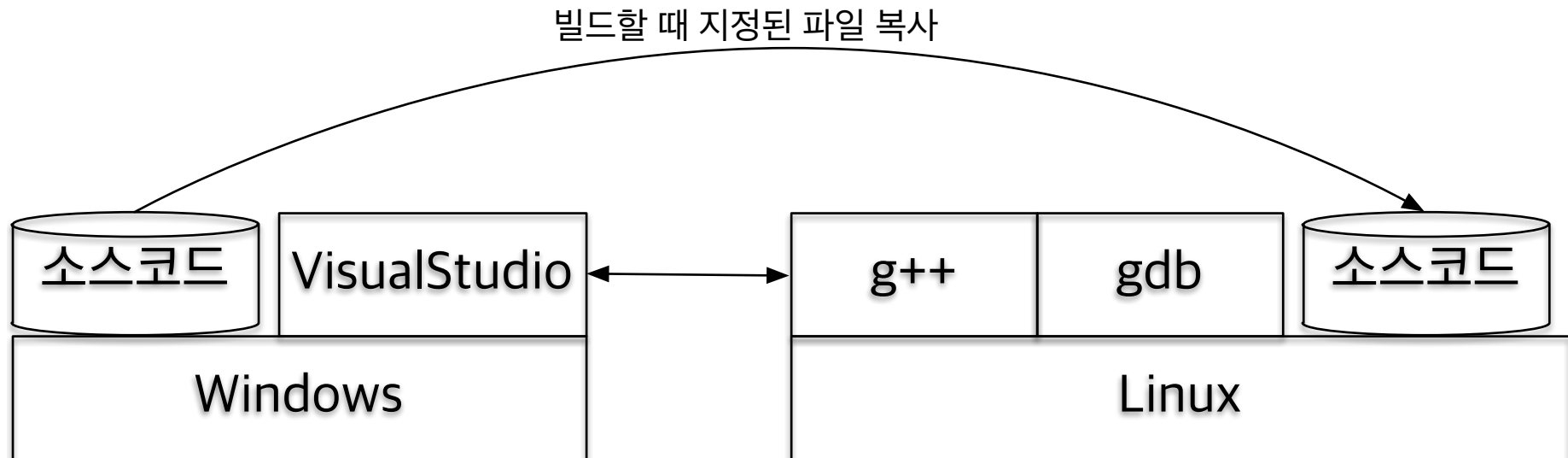
준비 줄: 9 열: 31 문자: 31 INS ↑ 0 ↵ 8 crow master

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



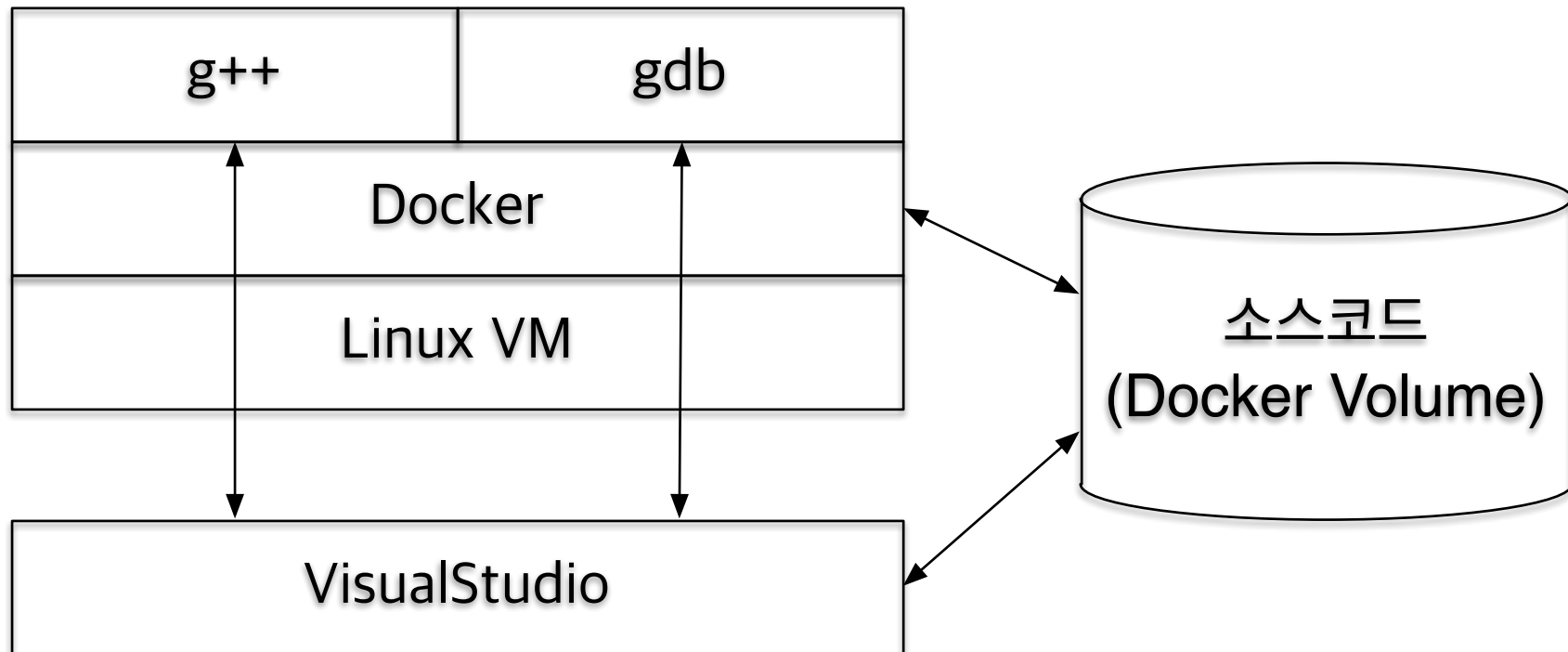
컨테이너 띄우기

- 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



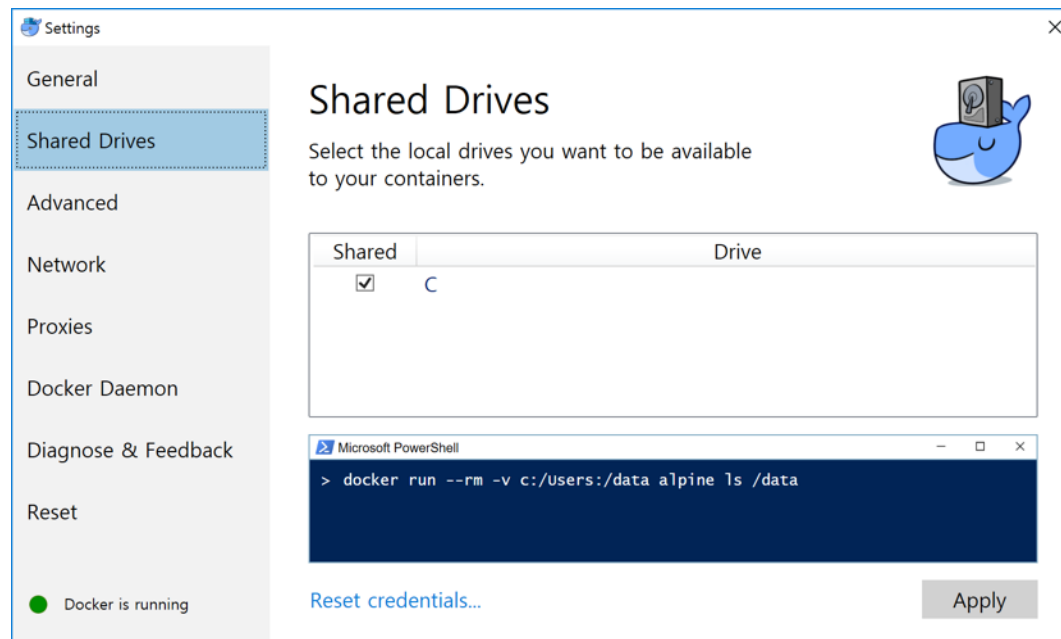
Windows: VisualStudio (2)



- 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)



Containers + NEW

cpp-builder cpp-builder:latest

STOP RESTART EXEC DOCS

Home Settings

General **Ports** Volumes Advanced

Configure Ports

DOCKER PORT	MAC IP:PORT	
22	localhost:6000	TCP ▾
	localhost:	TCP ▾

+

SAVE

DOCKER CLI

Example: Kitematic 설정 (volume)



The screenshot shows the Kitematic interface for a container named 'cpp-builder' which is in a 'RUNNING' state. The interface includes a sidebar with 'Containers' and a '+ NEW' button. The main area is divided into 'Home' and 'Settings' tabs, with 'Settings' selected. Under 'Settings', there are tabs for 'General', 'Ports', 'Volumes', and 'Advanced', with 'Volumes' selected. The 'Configure Volumes' section has two columns: 'DOCKER FOLDER' and 'LOCAL FOLDER'. A single volume configuration is shown, with the Docker folder set to '/home/iff/workspace' and the local folder set to 'C:\Users\Carlos\Documents\WindowsDemo\work'. The 'CHANGE' and 'REMOVE' buttons for this volume are highlighted with a red rectangular box. At the bottom of the interface, there are icons for 'DOCKER CLI', a chat bubble, and a gear icon.

Example: Kitematic 설정 (advanced)



The screenshot shows the Kitematic interface for a container named 'cpp-builder' which is in a 'RUNNING' state. The interface includes a sidebar with 'Containers' and a '+ NEW' button. The main area displays the 'Advanced' settings for the container, with tabs for 'General', 'Ports', 'Volumes', and 'Advanced'. Under 'Advanced Options', the following options are checked:

- Allocate a TTY for this container
- Keep STDIN open even if not attached
- Privileged mode

A 'SAVE' button is located below the 'Privileged mode' option. The 'Privileged mode' checkbox and its label are highlighted with a red rectangular border.



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)



crow 속성 페이지

구성(C): **활성(Debug)** 플랫폼(P): **활성(x64)** 구성 관리자(O)...

- 구성 속성
 - General
 - Remote settings**
 - 디버깅
 - Copy Sources
 - Build Events
 - C/C++
 - Remote Build
- General
 - Target machine: None (you can add a machine using the Linux Connection Manager)
 - Remote Root Directory**: **~/workspace**
 - Remote Project Directory: $\$(RemoteRootDir)/\$(ProjectName)$

참 캡처(W)

Remote Root Directory
Specifies a path to a directory on the remote machine or device.

확인 취소 적용(A)

Example: VS 설정 (2/5)



crow 속성 페이지

구성(C): **활성(Debug)** 플랫폼(P): **활성(x64)** 구성 관리자(O)...

- 구성 속성
 - General
 - Remote settings
 - 디버깅**
 - Copy Sources
 - Build Events
 - C/C++
 - Remote Build

시작할 디버거: Remote GDB Debugger

Debugger Type	Native Only
Pre-Launch Command	
Program	<code>\$(RemoteProjectDir)/\$(Configuration)/examples/helloworld</code>
Program Arguments	
Working Directory	<code>\$(RemoteProjectDir)/\$(Configuration)</code>
Additional Debugger Commands	
Debugger Port Number	
Remote Debugger Port Number	
Debugging Mode	gdb
Additional Symbol Search Paths	

Working Directory
The remote application's working directory. By default, the user home directory.

Example: VS 설정 (3/5)



crow 속성 페이지

구성(C): **활성(Debug)** 플랫폼(P): **활성(x64)** 구성 관리자(O)...

- 구성 속성
 - General
 - Remote settings
 - 디버깅
 - Copy Sources**
 - Build Events
 - C/C++
 - Remote Build
- General
 - Sources to copy @ (SourcesToCopyRemotely)
 - Copy sources** 아니요
 - Additional sources to copy

Copy sources
Specifies whether to copy the sources to the remote system.

Example: VS 설정 (4/5)



crow 속성 페이지

구성(C): **활성(Debug)** 플랫폼(P): **활성(x64)** 구성 관리자(O)...

- 구성 속성
 - General
 - Remote settings
 - 디버깅
 - Copy Sources
 - Build Events
 - Pre-Build Event
 - Post-Build Event
 - Remote Pre-Build Event**
 - Remote Post-Build Event
 - C/C++
 - Remote Build

Command Line	<code>mkdir -p \$(RemoteProjectDir)/\$(Configuration); cd \$(RemoteProjectDir)</code>
Description	
Use In Build	예
Additional files to copy	

Command Line

```
mkdir -p $(RemoteProjectDir)/$(Configuration); cd $(RemoteProjectDir)/$(Configuration); cmake .. -DCMAKE_BUILD_TYPE=Debug
```

```
mkdir -p ~/workspace/crow/Debug; cd ~/workspace/crow/Debug; cmake .. -DCMAKE_BUILD_TYPE=Debug
```

매크로(M)>>

확인 취소

Command Line
Specifies a command line for the pre-build event tool to run on the remote system.

확인 취소 적용(A)

Example: VS 설정 (5/5)



crow 속성 페이지

구성(C): **활성(Debug)** 플랫폼(P): **활성(x64)** 구성 관리자(O)...

- 구성 속성
 - General
 - Remote settings
 - 디버깅
 - Copy Sources
 - Build Events
 - Pre-Build Event
 - Post-Build Event
 - Remote Pre-Build Event
 - Remote Post-Build Event
 - C/C++
 - Remote Build

General

Build Command Line	cmake --build
Rebuild All Command Line	
Clean Command Line	
Outputs	

Build Command Line

```
cmake --build $(RemoteProjectDir)/$(Configuration) -- -j2
```

평가 값:

```
cmake --build ~/projects/crow/Debug -- -j2
```

확인 취소

확인 취소 적용(A)

Example: Linux 서버 연결 설정



처음 빌드할 때 서버와 어떻게 연결할지 지정

Connect to Linux

This project uses remote builds, and a remote machine is required to host the builds and debug. Please enter the remote machine details below.

[Manage existing connections](#)

Host name:

Port:

User name:

Authentication type:

Password:



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: 아이편엔진

- 아이편엔진도 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



The screenshot shows the Nuclide IDE interface. The code editor displays the following C++ code:

```
1 #include "crow.h"
2
3 int main()
4 {
5     crow::SimpleApp app;
6
7     CROW_ROUTE(app, "/")
8     ([]() {
9         return "Hello world!";
10    });
11
12    app.port(18080).run();
13 }
14
```

The IDE shows a breakpoint at line 12. The console window on the right displays the following watch data:

```
app: (crow::SimpleApp)
  ▶ bindaddr_: (std::__cxx11::string) "0.0.0.0"
  ▶ concurrency_: (uint16_t) 1
  ▶ middlewares_: (std::tuple<>)
  ▶ port_: (uint16_t) 80
  ▶ router_: (crow::Router)
  ▶ server_: (std::unique_ptr<crow::Server<crow::
  ▶ tick_function_: (std::function<void ()>)
  ▶ tick_interval_: (std::chrono::milliseconds)
```

The Call Stack shows the following frames:

```
main +a0
__libc_start_main +f0
```

The Scope window shows the following local variables:

```
app: (crow::SimpleApp)
boost::asio::detail::keyword_tss_ptr<boost::a
```

The Breakpoints window shows the following breakpoints:

```
helloworld.cpp:5
helloworld.cpp:12
```

The diagnostics table shows the following error:

Type	Source	Line	Description
Error	Clang	7	called object type 'const char [2]' is not a function or function pointer



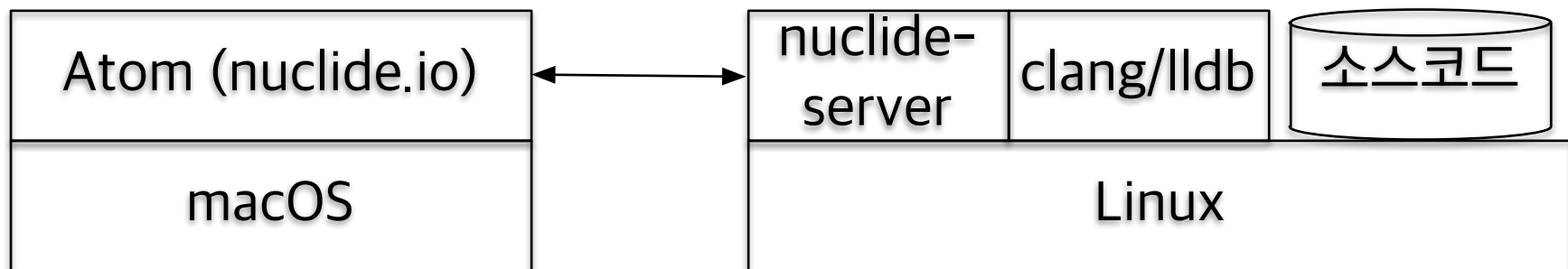
macOS: Nuclide

- 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





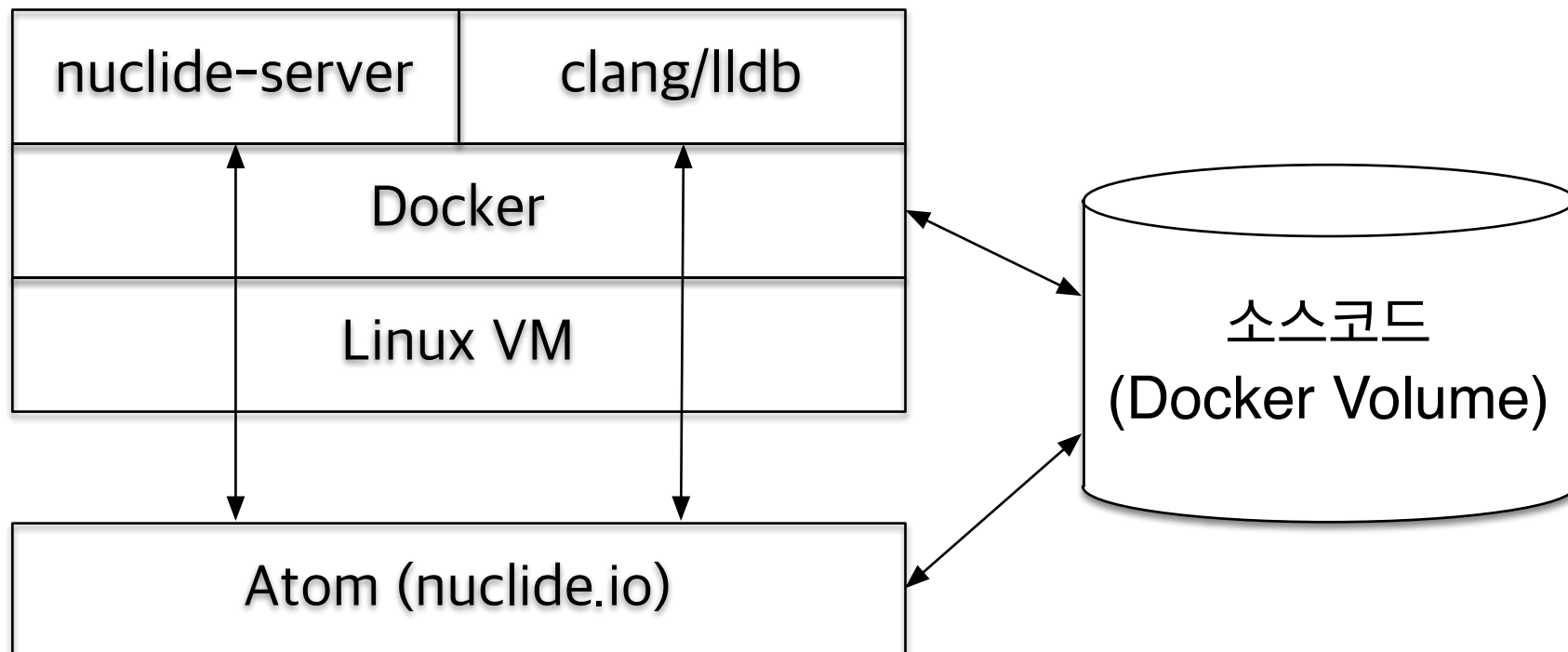
컨테이너로 바꿉시다

- 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)

The screenshot shows a web interface for managing containers. The main area is titled 'My Images' and displays a grid of Docker images. The first image in the grid, 'local nuclide-server', is highlighted with a red border. Each image card includes a Docker logo, the image name, a description (or 'No description.'), a tag (e.g., 'latest', '16.04', '5.5', '2.8', '6', 'centos7', 'alpine'), and a 'CREATE' button. The interface also features a 'Containers' sidebar, a '+ NEW' button, and a 'FILTER BY' dropdown menu with options for 'All', 'Recommended', 'My Repos', and 'My Images'. The bottom of the interface has a 'DOCKER CLI' button and a settings gear icon.

Example: Kitematic 설정 (ports)

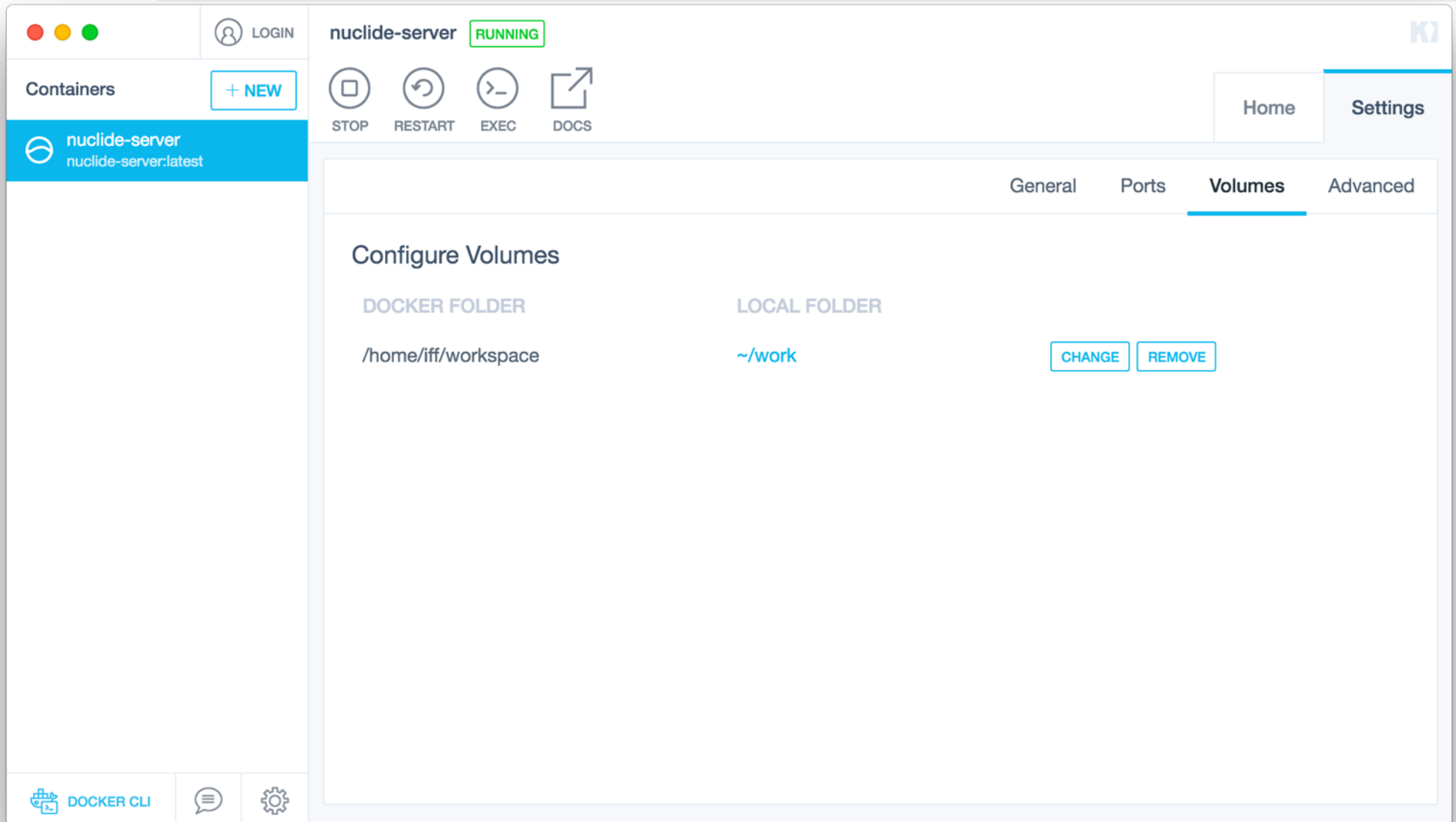


The screenshot shows the Kitematic interface for a container named 'nuclide-server' in a 'RUNNING' state. The 'Ports' tab is selected, displaying a 'Configure Ports' section. A table lists the following port mappings:

DOCKER PORT	MAC IP:PORT	Protocol
22	localhost:6000	TCP
9090	localhost:9090	TCP
9091	localhost:9091	TCP
9092	localhost:9092	TCP
9093	localhost:9093	TCP
	localhost:	TCP

The first five rows of the table are highlighted with a red border. A 'SAVE' button is located at the bottom left of the configuration area. The interface also includes a sidebar with 'Containers' and 'nuclide-server' listed, and a top navigation bar with 'Home' and 'Settings' options.

Example: Kitematic 설정 (volume)



The screenshot displays the Kitematic interface for a container named "nuclide-server" in a "RUNNING" state. The interface is divided into several sections:

- Top Bar:** Includes a "LOGIN" button, the container name "nuclide-server" with a "RUNNING" status indicator, and navigation tabs for "Home" and "Settings".
- Containers List:** Shows the selected container "nuclide-server" with the image "nuclide-server:latest".
- Actions:** Buttons for "STOP", "RESTART", "EXEC", and "DOCS" are visible.
- Configuration Tabs:** The "Volumes" tab is selected, with other tabs for "General", "Ports", and "Advanced".
- Configure Volumes:** A table-like structure showing the mapping between the "DOCKER FOLDER" and the "LOCAL FOLDER".

DOCKER FOLDER	LOCAL FOLDER	Actions
/home/iff/workspace	~/work	CHANGE REMOVE
- Bottom Bar:** Contains icons for "DOCKER CLI", a chat bubble, and a settings gear.

Example: Kitematic 설정 (advanced)



The screenshot shows the Kitematic interface for a container named 'nuclide-server' in a 'RUNNING' state. The container is running the image 'nuclide-server:latest'. The interface includes a sidebar with 'Containers' and a '+ NEW' button. The main area shows 'Advanced Options' with three checked checkboxes: 'Allocate a TTY for this container', 'Keep STDIN open even if not attached', and 'Privileged mode'. The 'Privileged mode' checkbox is highlighted with a red border. A 'SAVE' button is located below the checkboxes. The interface also features a 'Home' and 'Settings' tab, and a 'General', 'Ports', 'Volumes', and 'Advanced' sub-tab. The bottom of the interface has a 'DOCKER CLI' button and a settings gear icon.

Example: Nuclide 설정 (remote)



untitled

File Tree Source Control

Add Project Folder

Add Remote Project Folder

Profile Name:
nuclide-docker

Username:
iff

Server:
127.0.0.1

SSH Port:
6000

Initial Directory:
/home/iff/workspace/crow

Authentication method:

Password:

Use ssh-agent

Private Key File:

Remote Server Command:
(DEFAULT)

Cancel Save

with

Example: Nuclide 설정 (debugger)



The screenshot displays the Nuclide IDE interface. On the left, a code editor shows a C++ file named `helloworld.cpp` with the following content:

```
1 #include "crow.h"
2
3 int main()
4 {
5     crow::SimpleApp app;
6
7     CROW_ROUTE(app, "/") {
8         return "Hello World!";
9     };
10
11
12     app.port(18080);
13 }
14
```

In the center, a configuration dialog is open for launching an executable. The fields are as follows:

- Connection: 127.0.0.1
- Type: C++
- Action: Launch
- Executable: `/home/iff/workspace/crow/build/examples/helloworld`
- Arguments: Arguments to the executable
- Environment Variables: Environment variables (e.g., SHELL=/bin/bash)
- Working directory: Working directory for the launched executable

At the bottom, the console shows two error messages:

```
unable to find executable for '/home/iff/workspace/crow/build/examples/helloworld'
Failed to start debugger process: name: undefined, message: undefined, stack: undefined.
```

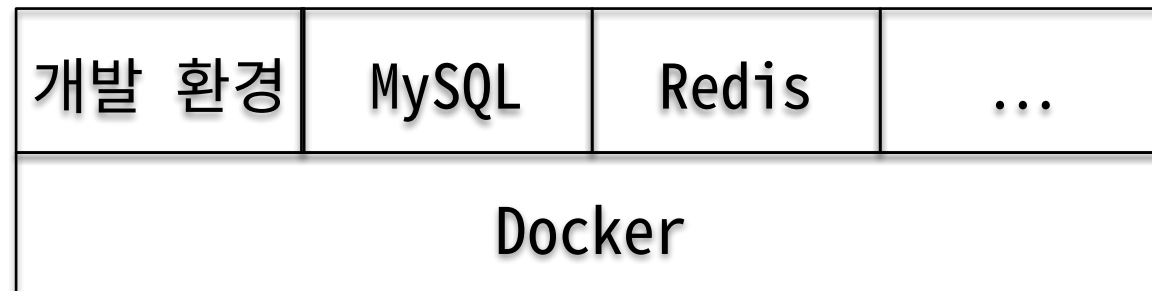
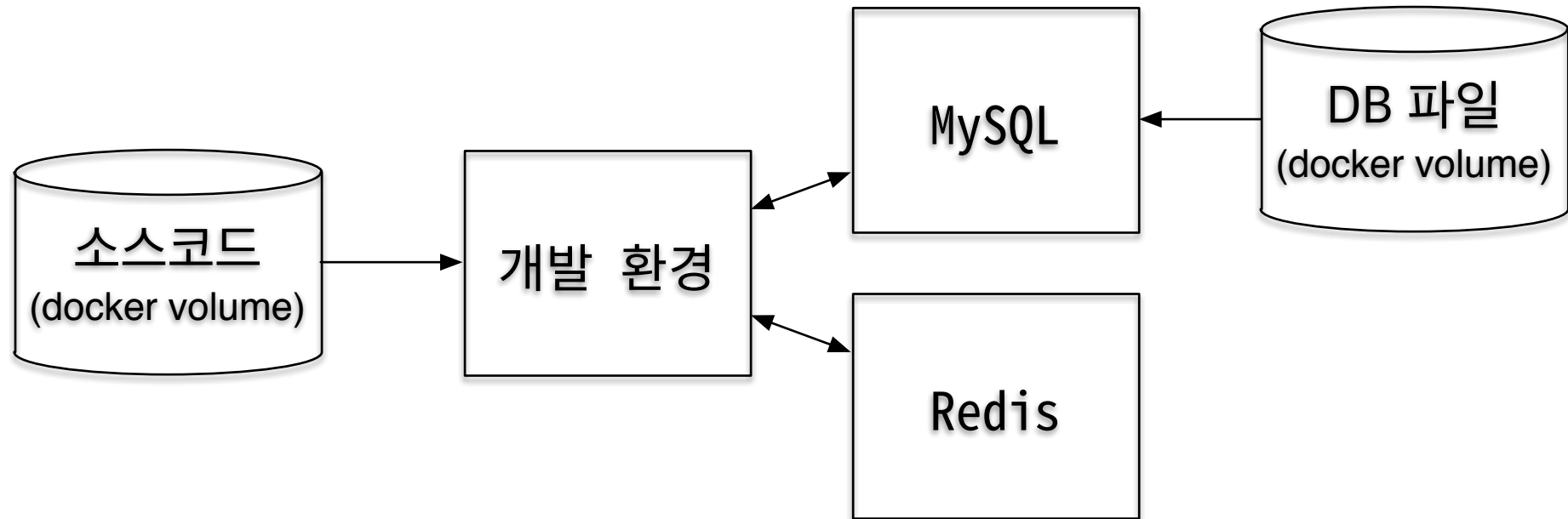
The status bar at the bottom indicates the current file is `examples/helloworld.cpp` at line 1:1, with a cursor at line 2, column 0. The encoding is UTF-8 and the language is C++.

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

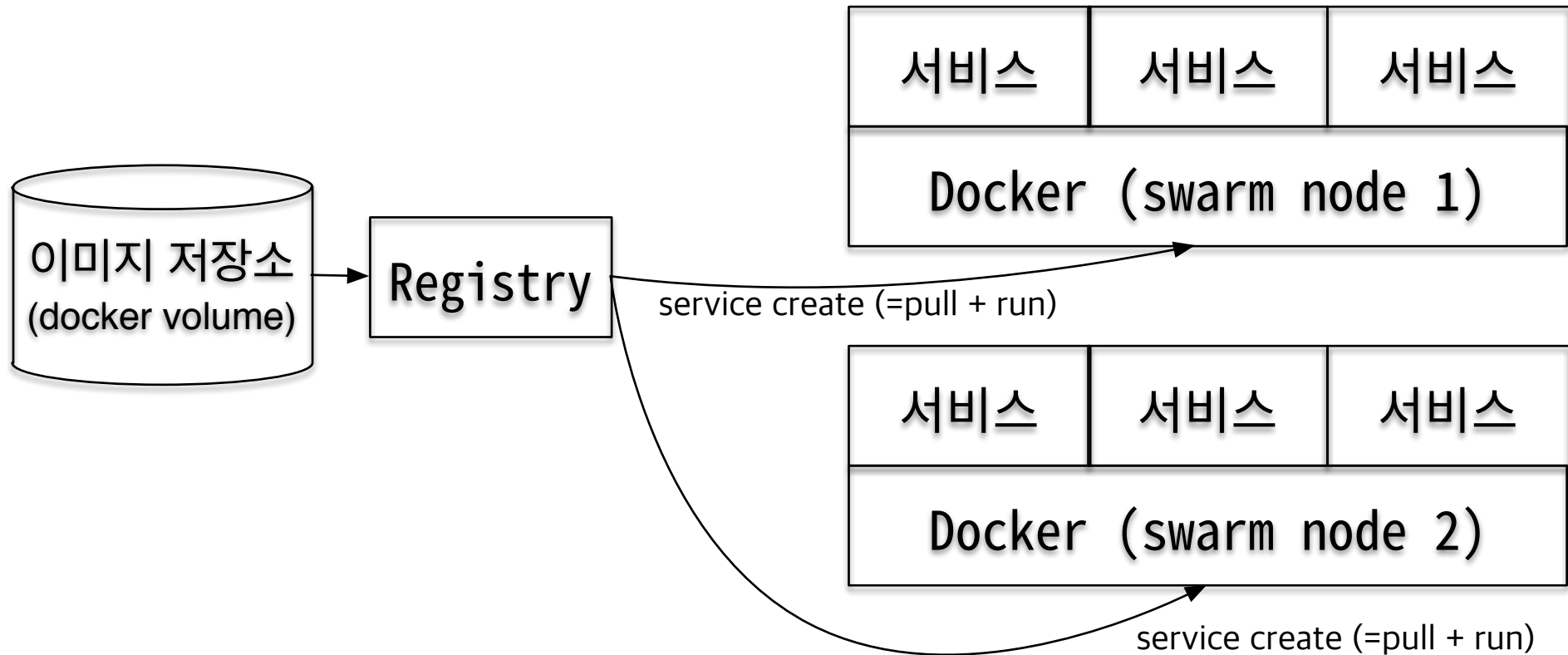


Docker Registry

- 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