61. Spring Framework 운용하기

① 'spring-webmvc' 라이브러리를 프로젝트에 적용

spring 6.x → Jakarta EE 사용 9.x jakarta.* Tomcat 10.x

✓ spring 5.x → JavaEE (8.x) = Jakarta EE (8.x) javax.* Tomcat 9.x

② 스프링 애노테이션으로 교체

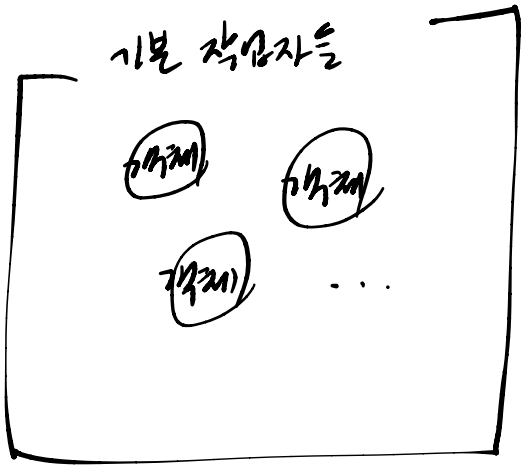
③ 〃 필터로 교체

④ 〃 사용법에 맞춰 AppConfig 수정

⑤ 〃 프론트 컨트롤러로 교체

⑥ 〃 IoC 컨테이너로 교체

* Spring Framework



+ 기능 추가?

⇓

그 기능을 수행할 객체를 등록
(작업자)

⇓ 방법

① 직접 객체를 생성해서 등록

② (애노테이션 / XML) 설정을 통해

어떤 기능을 처리할 때
작업자가 등록되어 있으면
(객체)
그 객체를 실행 (메서드호출)하여
처리한다.
없으면 에러를 던지거나 기능을 무시한다.

* 요청 파라미터 와 요청 핸들러의 파라미터

```
<form>
  <input name="email">
  <input name="password">
  <button> 로그인 </button>
</form>
```

post+요청 →

요청 파라미터

email=aaa@test.com & password=1111

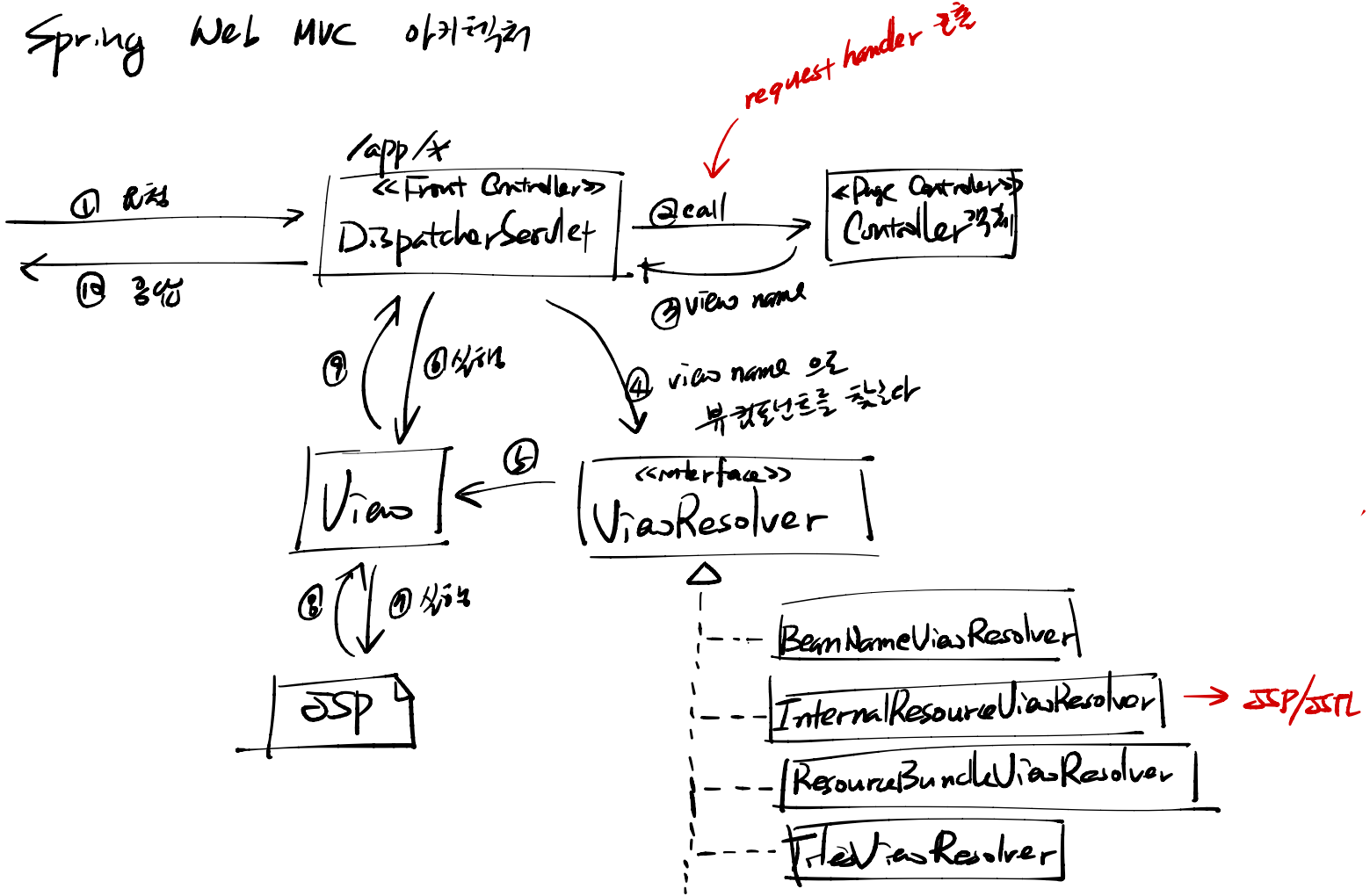
```
String login(String email, String password) {
}
```

요청 핸들러

≡

요청 핸들러의 파라미터

* Spring Web MVC 아키텍처



* InternalResourceViewResolver

"index" ← view name

↓

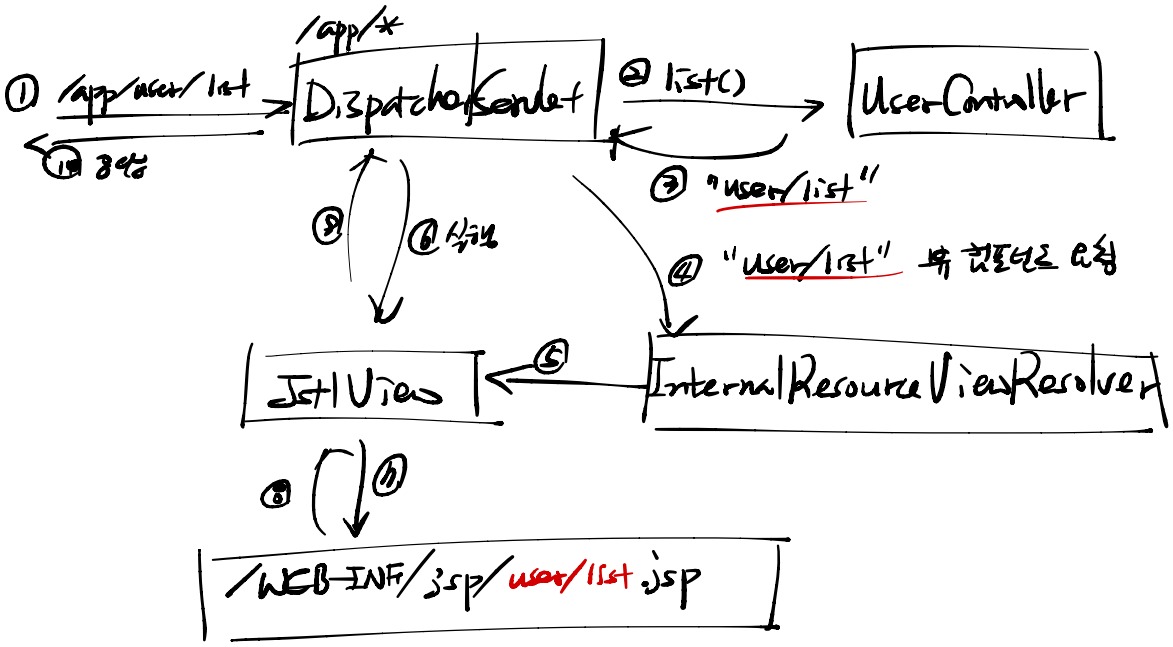
InternalResourceViewResolver

"/WEB-INF/jsp/" ↓ ".jsp"

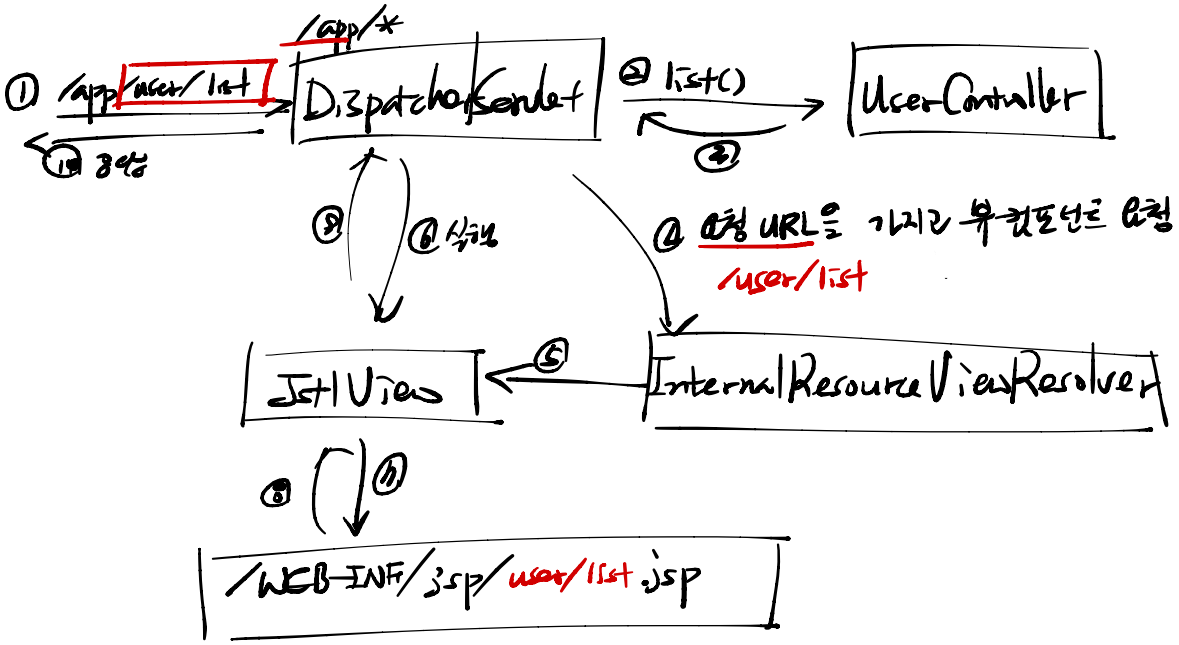
"/WEB-INF/jsp/index.jsp"

prefix suffix

* view name 있을 때

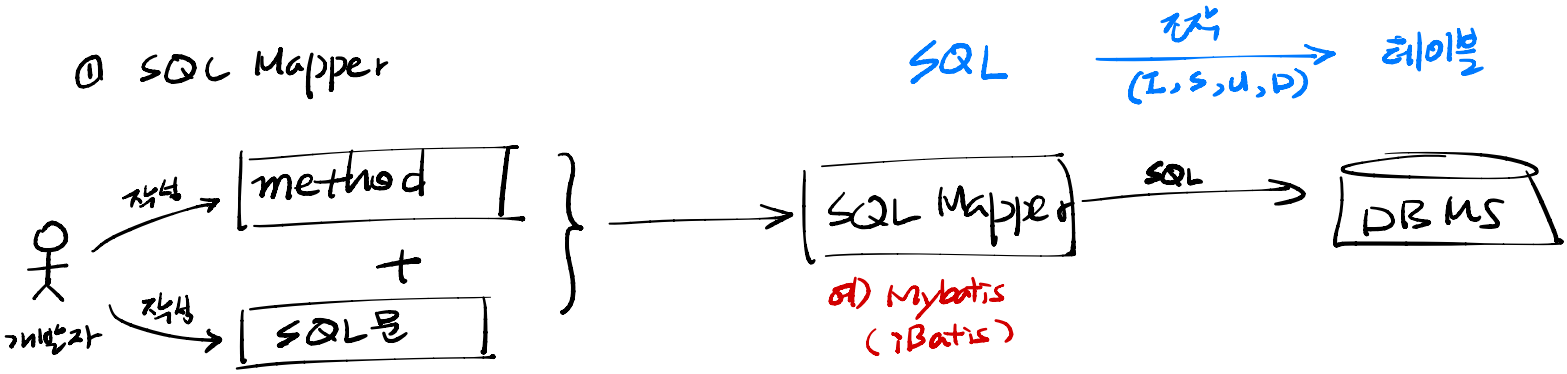


* view name 없을 때

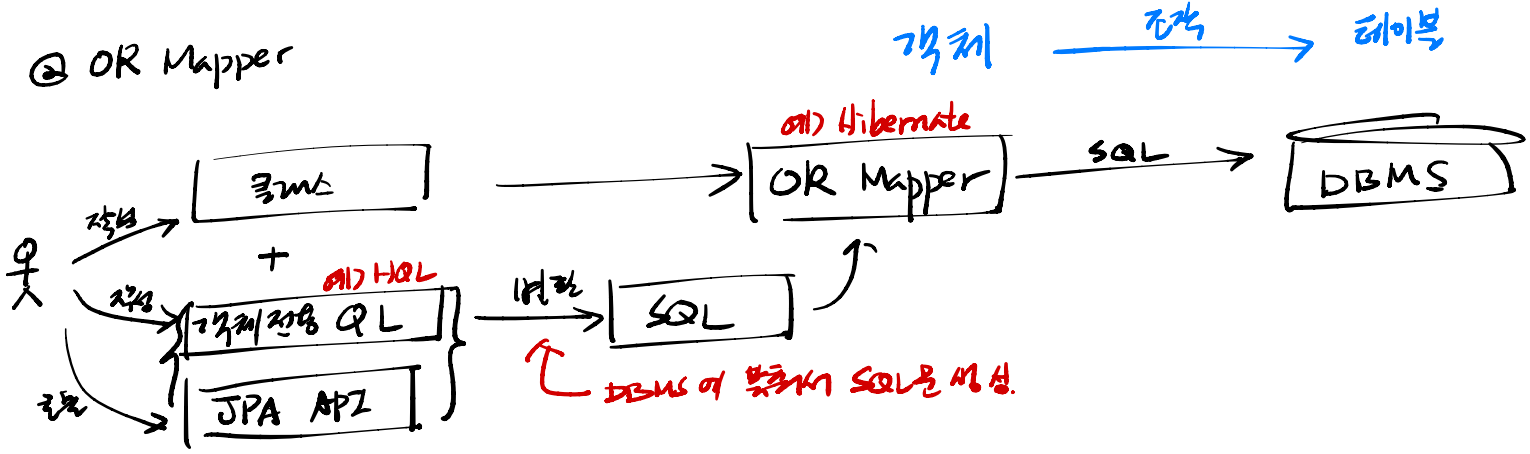


* SQL Mapper vs OR Mapper

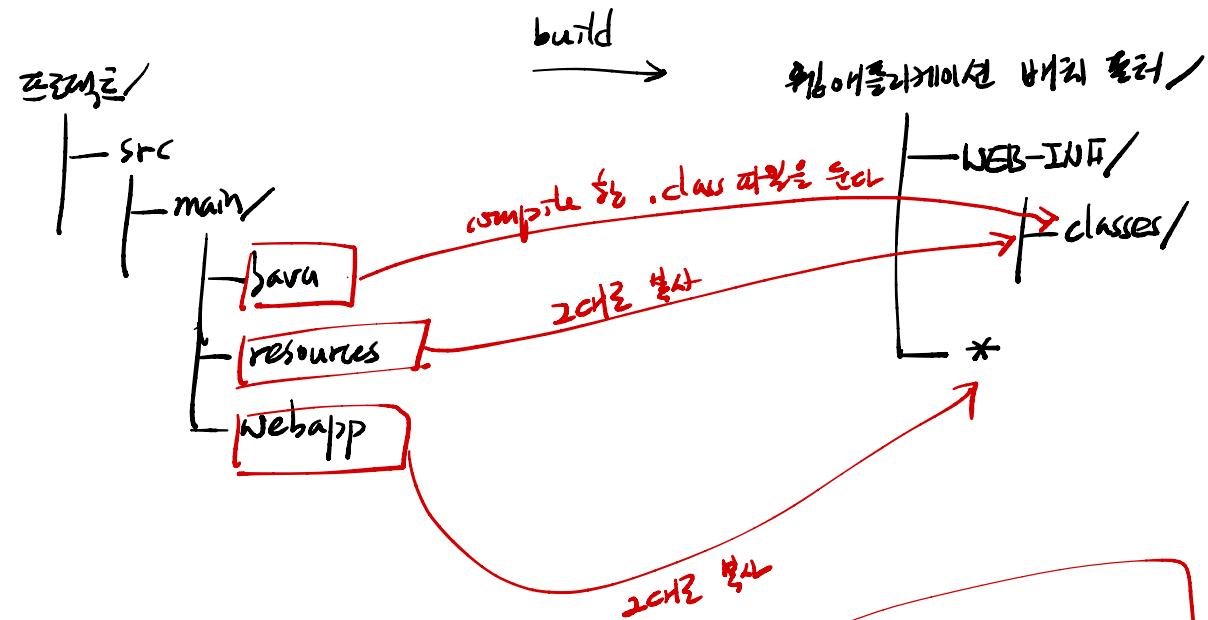
① SQL Mapper



② OR Mapper



* build 와 classpath



src/main/java/bitcamp/myapp/App.java → /WEB-INF/classes/bitcamp/myapp/App.class

src/main/resources/config/jdbc.properties → /WEB-INF/classes/config/jdbc.properties

* DaoFactory 와 SqlSessionTemplate

우리가 만든 DAO 팩토리

MyBatis 자체 공장을 DAO 팩토리

BoardDao
insert()

→ DaoFactory

SQL 맵퍼 파일에서 SQL 문을 찾을 때 사용하는 값

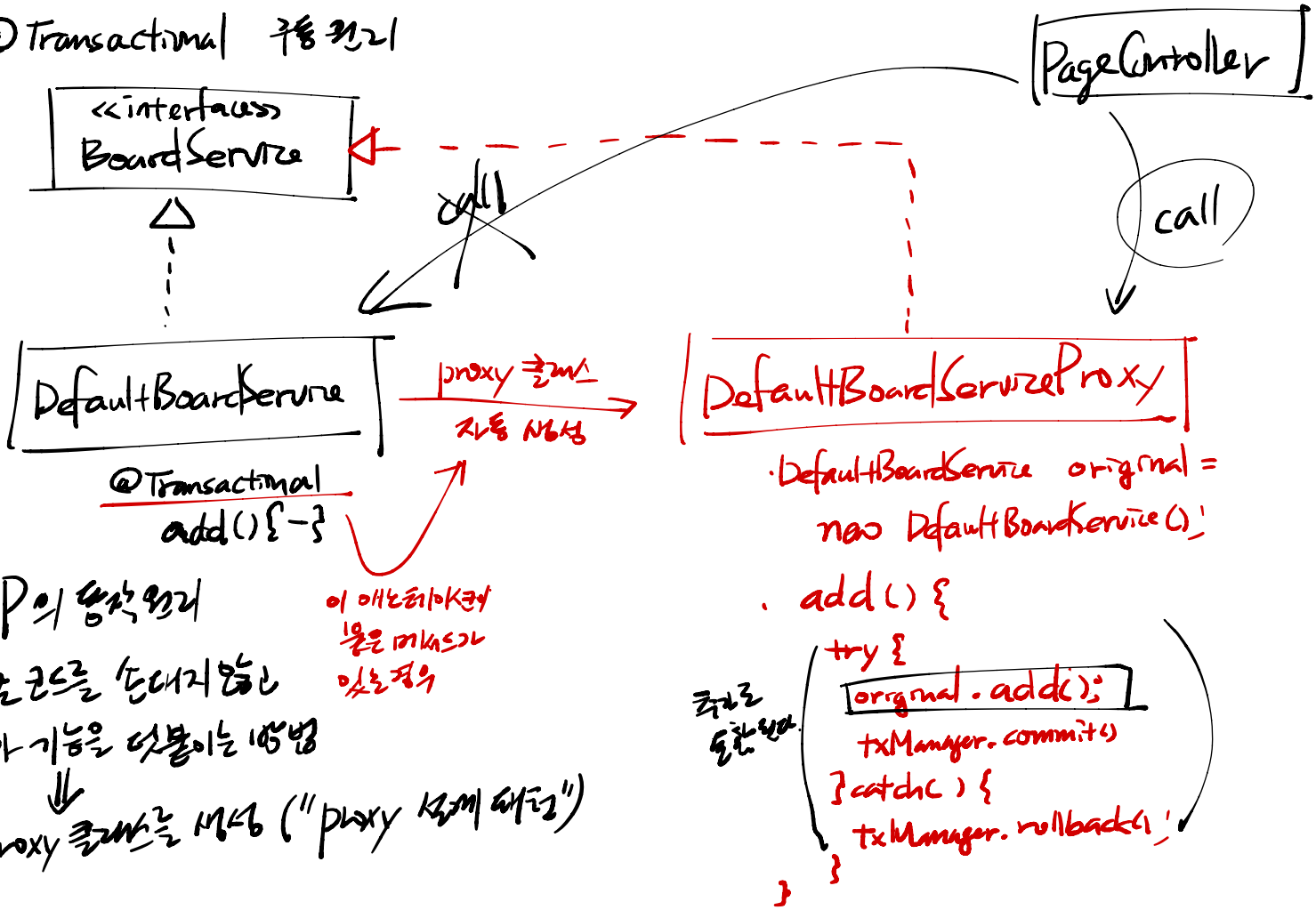
↳ sqlSession.insert("BoardDao.insert", —);

→ SqlSessionTemplate

패키지 이름이 포함된다.

↳ sqlSession.insert(
"biz.com.myapp.dao.BoardDao.insert",
—);

* @Transactional 구동원리



* AOP의 동작원리

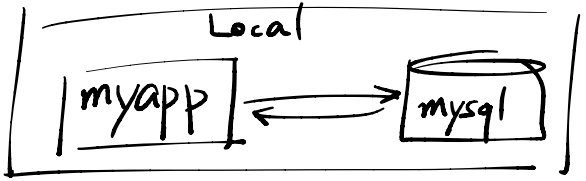
↳ 기존 코드를 손대지 않고
추가 기능을 덧붙이는 방법

⇓

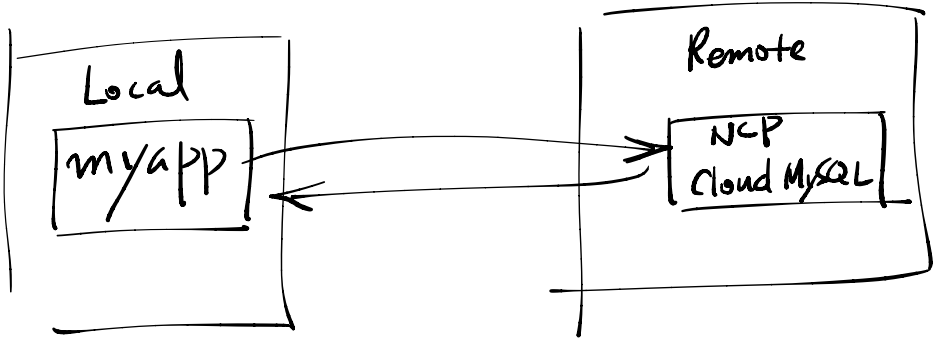
proxy 클래스를 생성 ("proxy 설계 패턴")

62. Naver Cloud Platform 서비스 도입

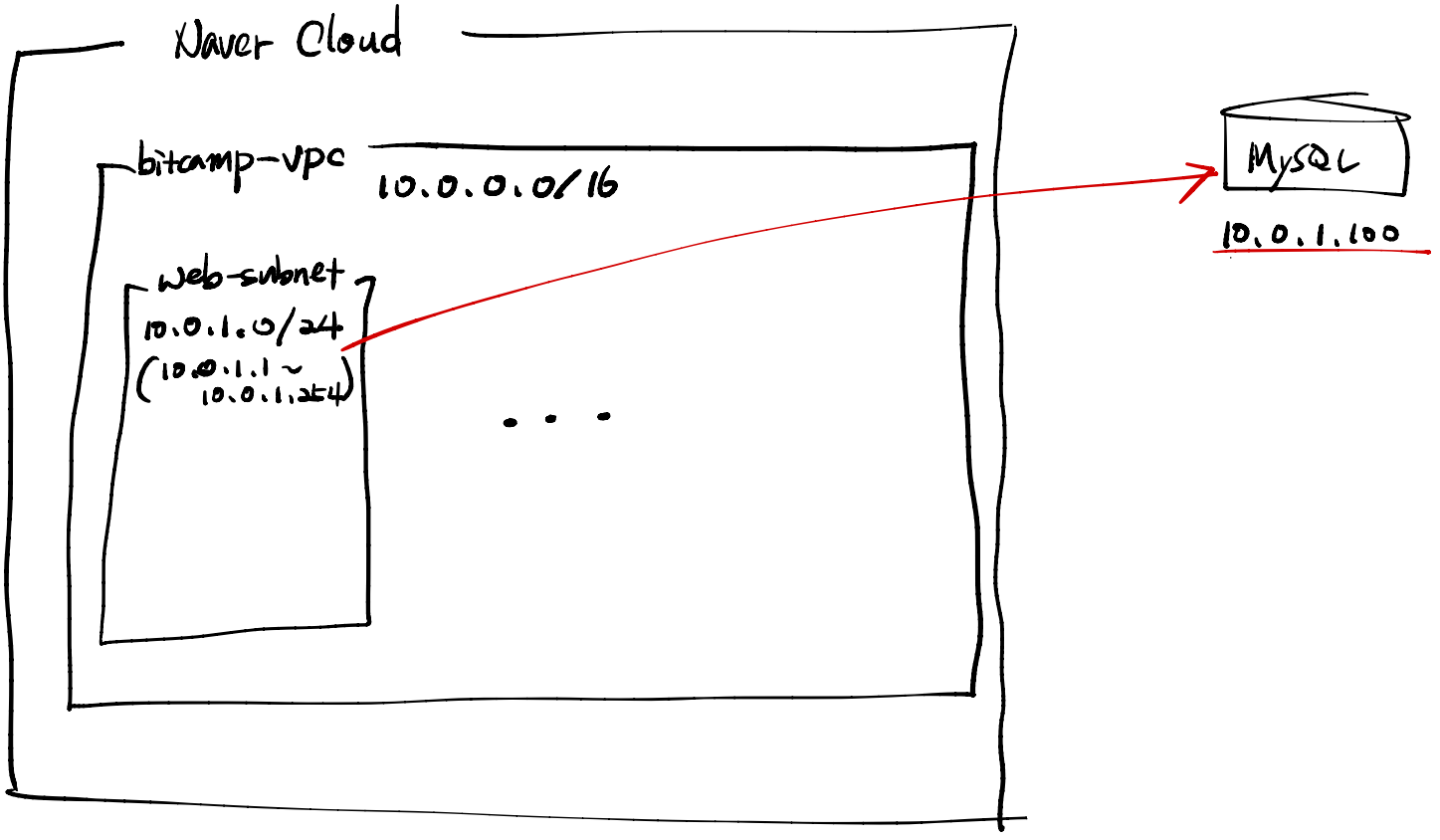
* 이전



* 변경



* VPC (Virtual Private Cloud) — 조직이나 개인이 별도로 관리할 사설 네트워크 망을 설정



* RESTful API

File 또는 Data

↓ 주소

URL

예) http://localhost:8888/app/board/view?no=3 GET 요청

↓ Representational 표현상

예) http://localhost:8888/app/boards/3 GET 요청

메서드의 성격이 강한 URL

데이터 이름의 성격이 강함
+
GET, POST, PUT, DELETE, PATCH

representational state transfer
"REST"

* function → REST API

① function → ② Remote Procedure Call (RPC) → ③ Remote Method Invocation (RMI)

다른 프로세스

통신

C 등 절차적 프로그래밍

기업의 업무처리

↓

App의 기능을 분산 (분산 컴퓨팅)

↓

한 App이 하던 일을

여러 App으로 쪼개서 되도록

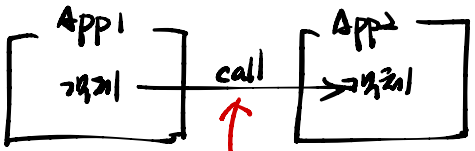
App 간의 연동이 필요하다

↓

그래서 다른 App의

function을 호출할 수 있는

기술이 등장하게 된 것.



통신

C++ 등 객체지향 프로그래밍

↓

OOP 프로그래밍의 특징이 발휘

RPC 를 개선

* function → REST API

③ RMI → ④ CORBA → ⑤ web-service

같은 언어로 만든 App끼리 함수를 호출할 수 있는 기술

C → C

~~→~~ PASCAL

common Object Request Broker Architecture

⇓

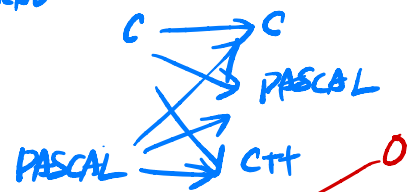
다른 언어로 만든 App끼리 함수를 호출할 수 있는 기술

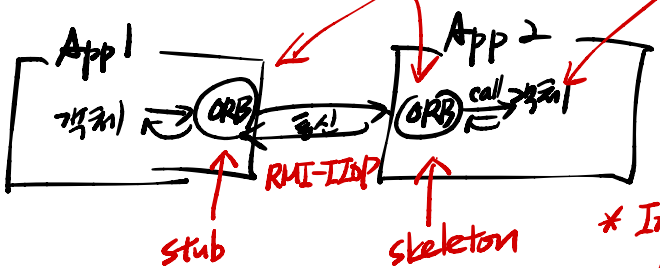
✓ 통신할 때 HTTP 프로토콜을 사용

✓ ORB를 클라이언트 언어에 맞춰 자동생성함

⇓

원격 객체를 만드는 개발자의 부담을 덜어줌!





ORB 생성을 원격 객체를 생성하는 측에서 맡음

"Remote object"라 (원격객체) 부름

* Internet Inter-ORB Protocol

↳ IIOP

⑥ Enterprise JavaBeans (EJB) 등장

자바언어 전용 RMI 기술

↑

단점!

2000년 초반에는 PC라는 단말 디바이스를 쓰던 사람이기에 이런 단점이 크게 문제되지 않았다.

* function → REST API

(Web-Service
EJB) → REST API

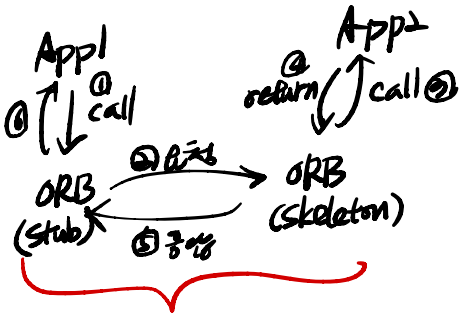
↓

프로그래밍 언어에 맞춰
ORB를 생성해야 한다

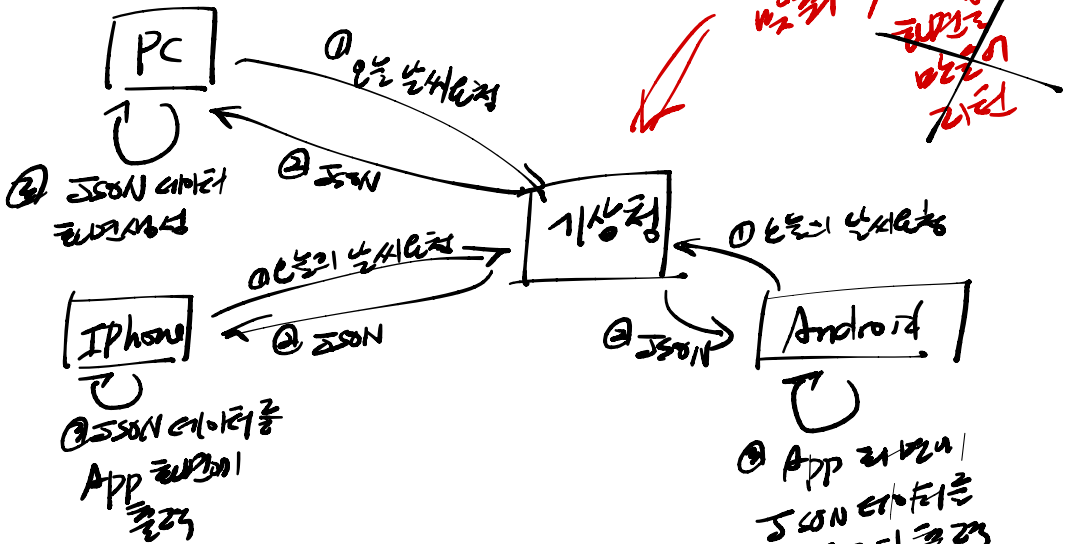
App1 —①요청 HTTP→ App2

← 응답 (XML/JSON 등)

여러 디바이스 환경에
맞춰 데이터 리턴
~~해당 디바이스 리턴~~



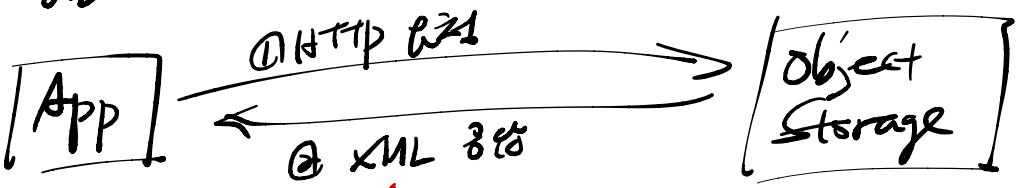
ORB를 통해
원격 객체의 메소드 호출



* Object Storage REST API 사용

↳ HTTP 요청/응답 규칙!

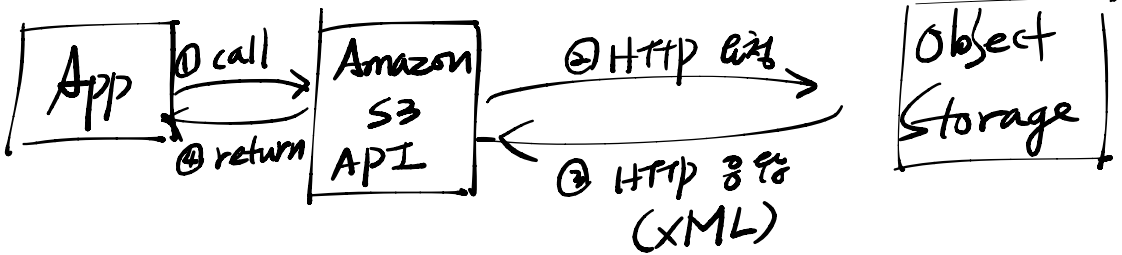
① 직접 호출



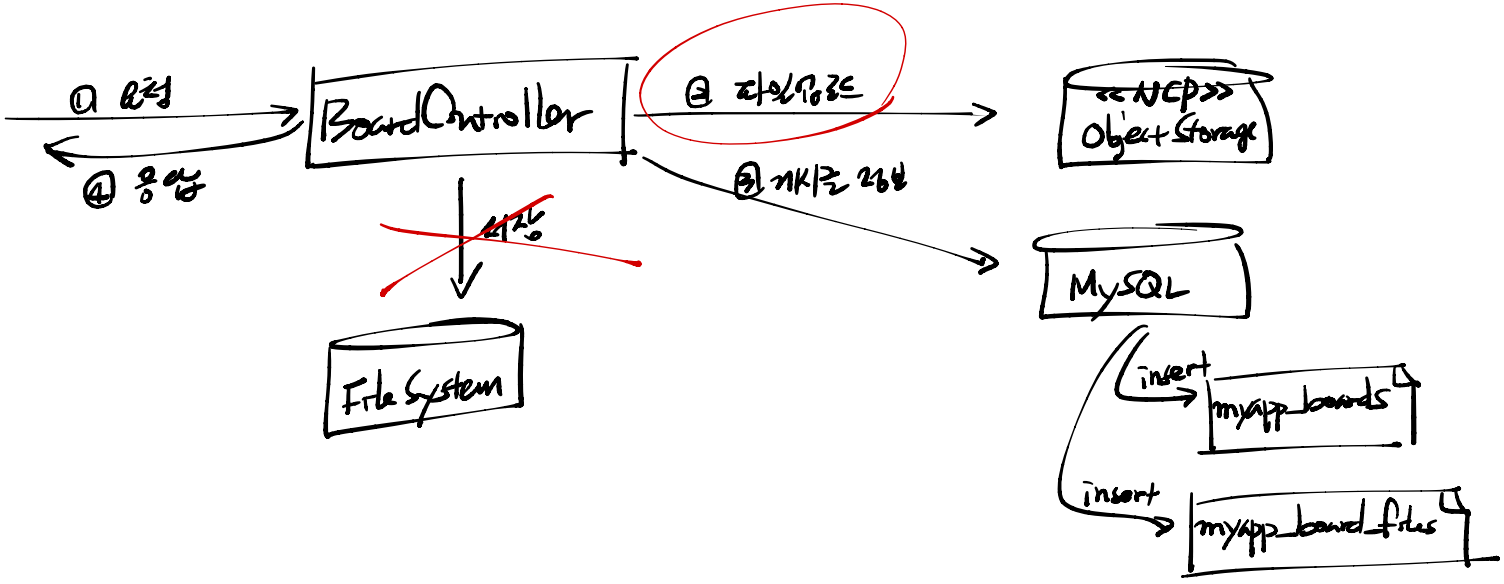
HTTP 요청을 직접 다루기가 번거롭다

② 간접 호출

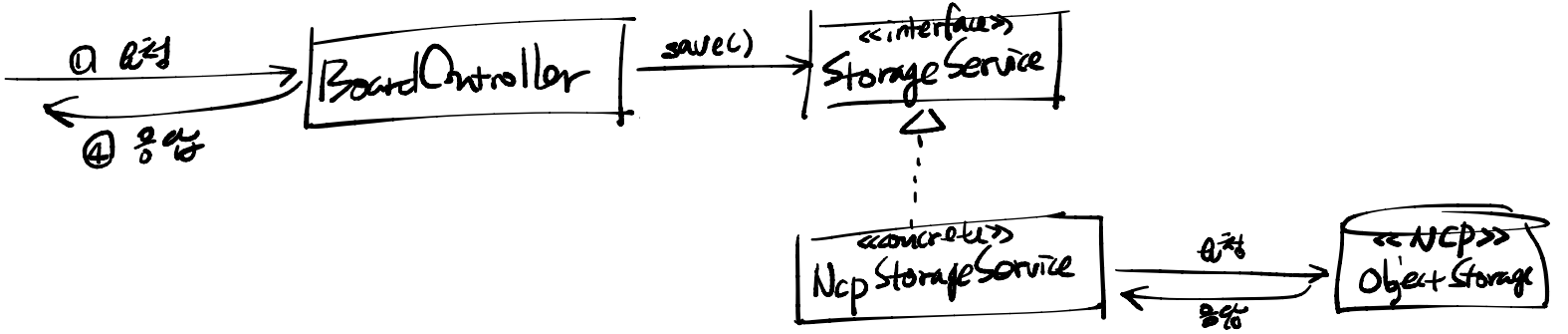
HTTP 프로토콜을 직접 다룰 필요가 없어
프로그래밍이 쉬워진다.



* 게시글 첨부파일을 NCP의 Object Storage에 저장



* 게시글 첨부파일은 NCP의 Object Storage에 저장 → NCP 연동 로직을 서비스 객체로 분리



* 사진관리와 보안

① 보안에 안좋은 예 : 회원정보 변경

client —(회원정보 + 새사진)→ UserController

기존 사진 파일명을 포함

- i) 기존 사진 삭제 → 클라이언트가 보낸 파일명으로 파일을 찾아 삭제
- ii) 새 사진 저장 → 새파일 저장
- iii) 정보변경 → DB 변경

A라는 사람이 로그인 한 후에
자신의 정보를 바꾼 여 사진 파일명을
다른 사람의 것으로 전달하다면?

↳ 다른 사람의 사진을 삭제할 수 있다.

⇓

입력, 변경, 삭제 시
개인 정보를 사용할 때는
항상 서버 쪽에서 조회하여 사용하라!
절대 개인정보를 클라이언트에서 받지 말라! → 또는 입력, 변경, 삭제할 때 권한을 검사하라!

* Transaction Propagation

Propagation \ Caller	Transaction	
	X	O (Tx1)
(default) REQUIRED	Tx1	Tx1
REQUIES_NEW	Tx1	Tx2
MANDATORY	에러	Tx1
SUPPORTS	트랜잭션이 무의미 함으로 none	Tx1
NOT_SUPPORTED	실행	현재 트랜잭션 중지한 상태로 실행
NEVER	실행	에러

* propagation 종류 예:

① REQUIRED

UserController.delete() ×

↓ call

Tx1 { DefaultUserService.delete()

Tx1 { UserController.delete()

↓ call

DefaultUserService.delete()

② REQUIRES_NEW

UserController.delete() ×

↓ call

Tx1 { DefaultUserService.delete()

Tx1 { UserController.delete()

↓ call

Tx2 { DefaultUserService.delete()

* 한번 삭제시 트랜잭션 문제

① 현황

X UserController.delete() X

call → Tx1 { DefaultUserService.delete() — REQUIRED

→ X NcpObjectStorageService.delete() — X

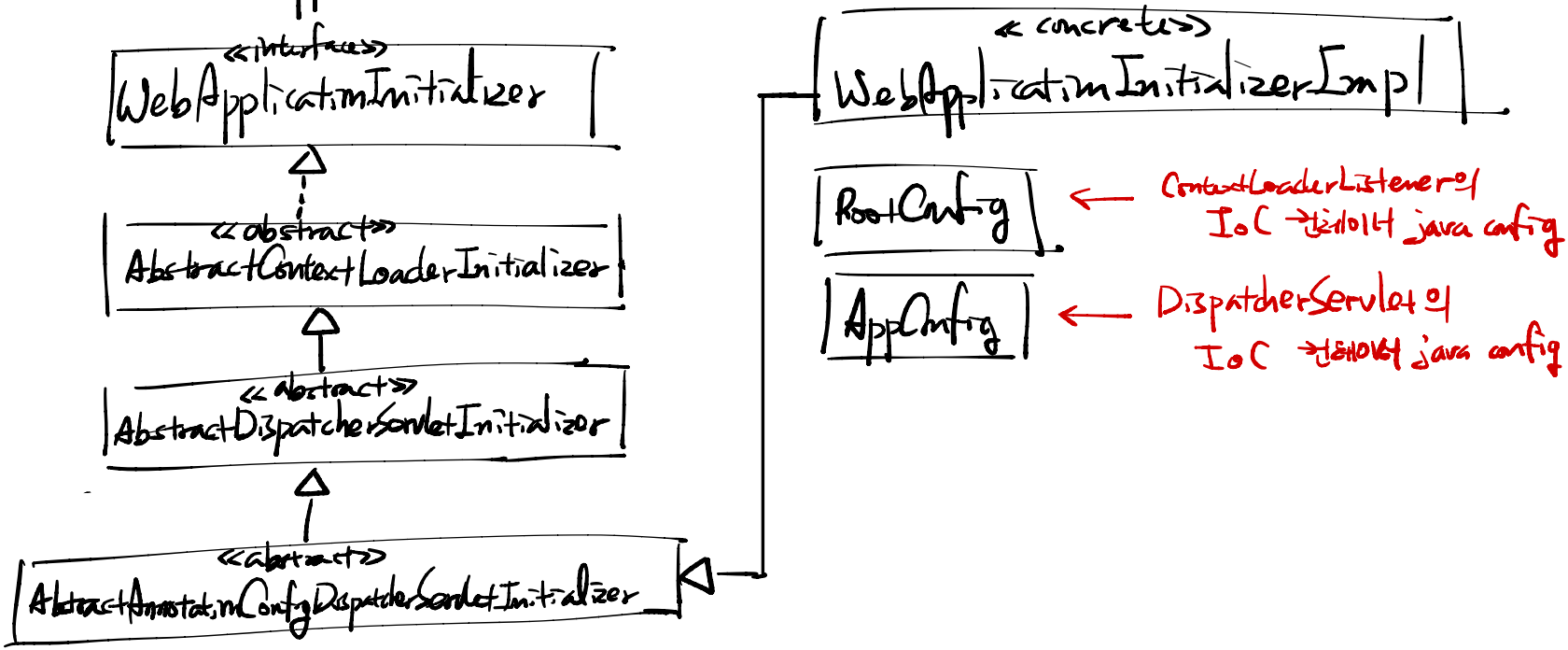
② 해결

Tx1 UserController.delete() — REQUIRED

call → Tx1 : DefaultUserService.delete() — REQUIRED

→ Tx1 : NcpObjectStorageService.delete() — REQUIRED

63. WebApplicationInitializer



65. Spring Boot 개념

Spring Framework

+

Tomcat

} Spring Boot

66. 페이징 처리

select

from

where

order by

limit 행번호, 개수

페이지의 시작 행번호 = (이전페이지수 × 개수)
= 이전페이지번호

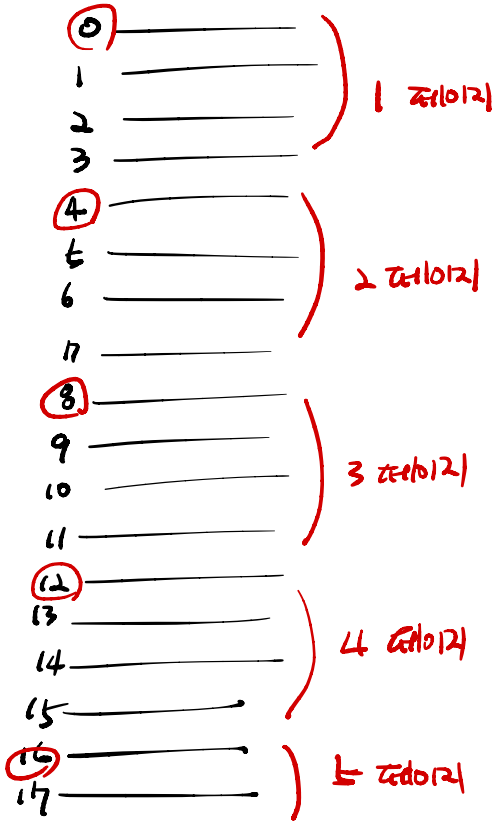
limit 0, 4

limit 4, 4

limit 8, 4

limit 12, 4

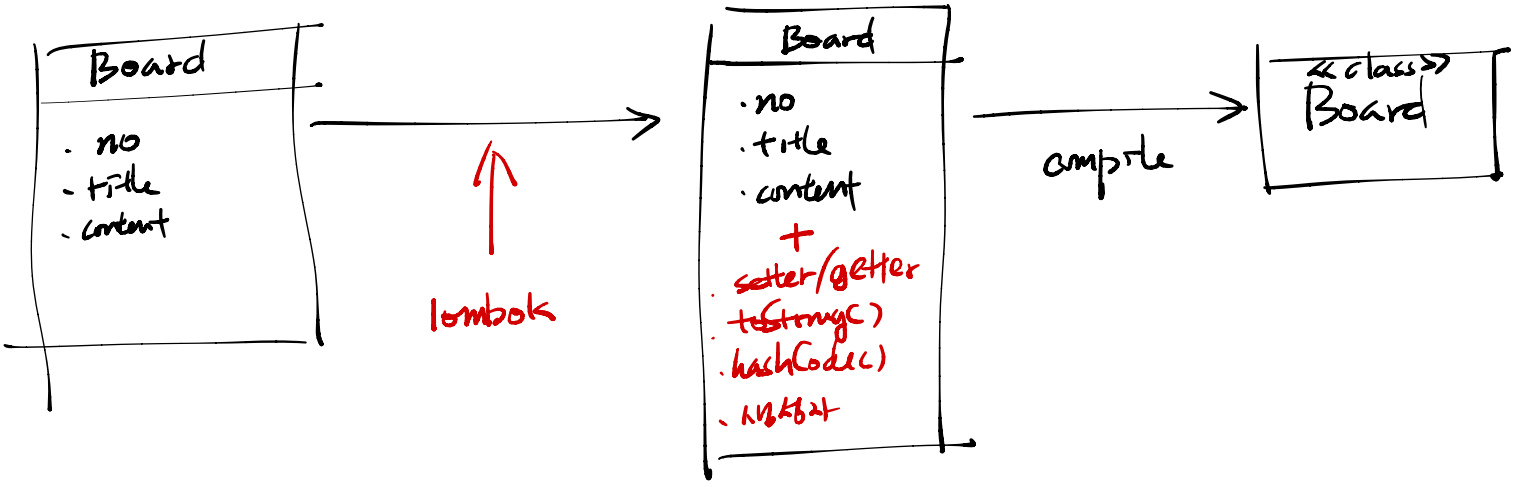
limit 16, 4



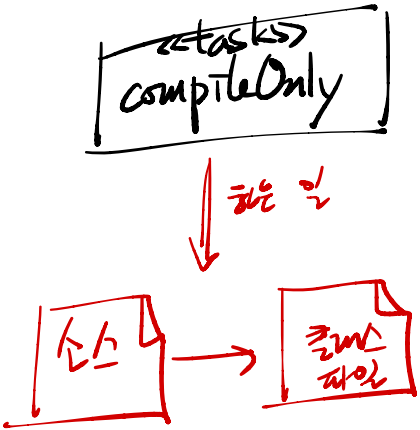
67. lombok 라이브러리 적용

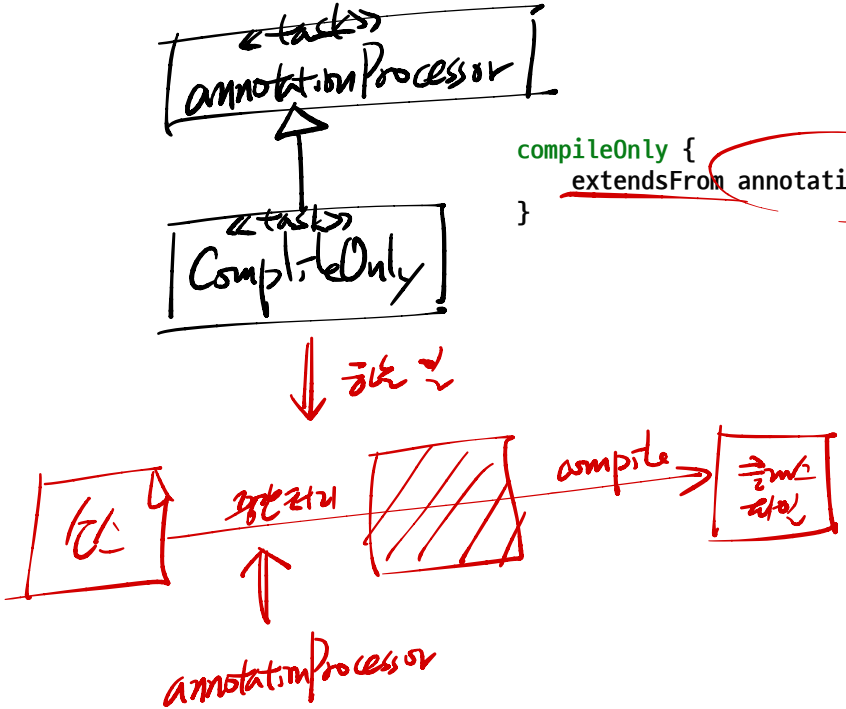
↳ Domain 클래스 생성 편리 → setter/getter, toString(), hashCode(), 생성자 자동 생성기능 제공

Gradle
⇓
Build



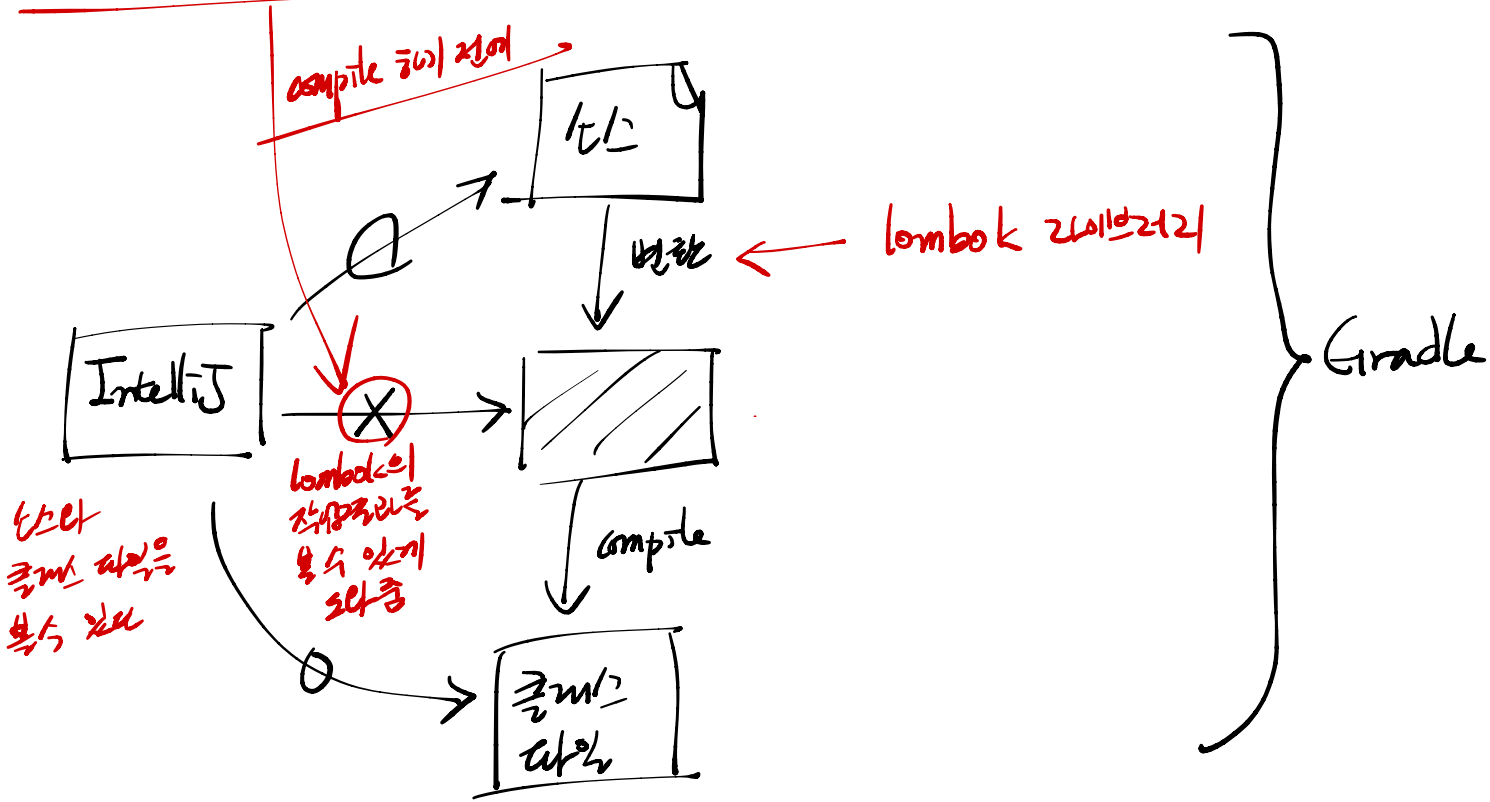
* Gradle 의 Task



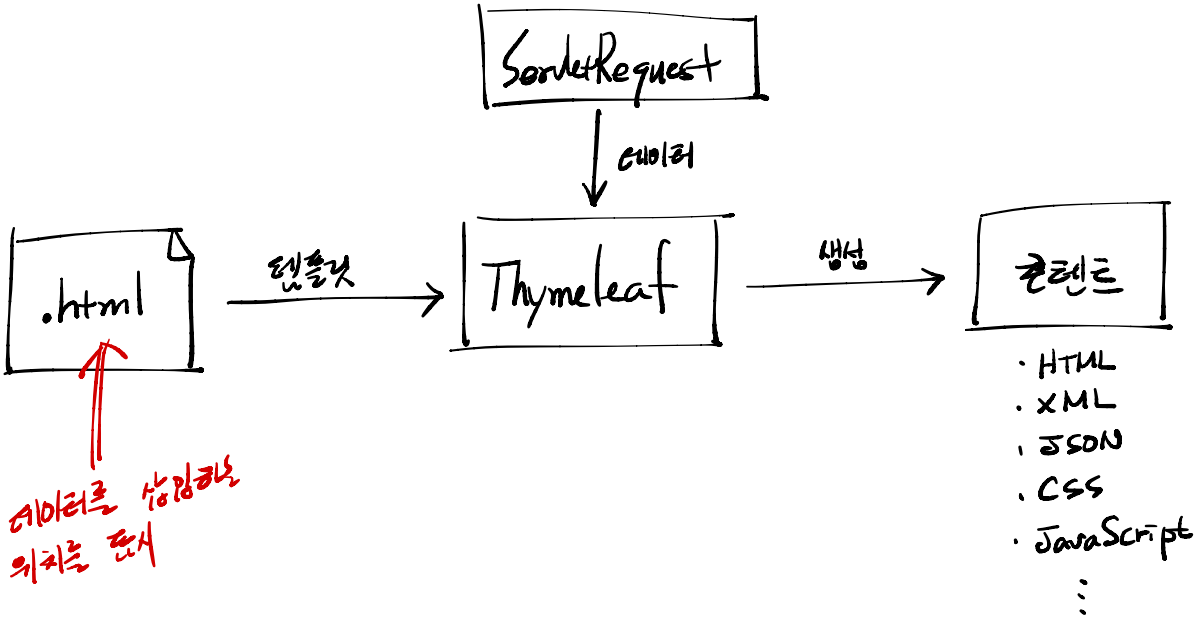



```
compileOnly {
    extendsFrom annotationProcessor
}
```

* lombok IntelliJ 플러그인의 역할



* Thymeleaf 템플릿 엔진



* Thymeleaf 템플릿과 HTML

템플릿 파일

.html

```
<p th:tex="#{OGNL}">예제 텍스트</p>
```

HTML 확장 속성

JSP → ${객체명}

Mybatis → #{객체명}

Thymeleaf → ${객체명}

Web Browser

Thymeleaf의 코드는 무시

HTML 렌더링 화면을 출력

디자이너

디자이너는 개발자가 삽입한 코드에 방해받지 않는 화면을 확인할 수 있다

Thymeleaf 의 특징

① JSP와 다르게 템플릿이 HTML 파일이다.

↳ 템플릿 엔진의 실행 없이 소스 그대로 웹브라우저에서 읽어서 출력 결과를 대략적으로 확인할 수 있다.

⊖ Thymeleaf 문법이 HTML 문법을 준수하기 때문에 소스 그대로 출력할 때 화면에 보이지 않는다.

* Thymeleaf 템플릿과 HTML

 예제데이터

대체

소스 그대로 읽을 때

Web Browser

예제 데이터 출력

Thymeleaf 엔진

 데이터

* HTML에 데이터 삽입하기

<태그명 th:text="표현식">표현식</태그명>

대치

표현식 실행 결과

⇓

<태그명>실행결과</태그명>

* Thymeleaf Namespace

namespace 별칭

```
<html xmlns:th="http://www.thymeleaf.org">
```

→ Thymeleaf 엔진에게 Thymeleaf 태그나 속성이 th 네임스페이스로 시작한다는 것을 알려준다

⇓

Thymeleaf 엔진은 HTML 문서에서 th:* 태그나 속성을 만나면 Thymeleaf 전용 태그나 속성으로 인식하여 처리할 것이다

```
<p th:text="${welcome}">환영합니다!</p>
```

※ xml namespace 선언을 생략하면 th: 를 기본으로 인식하고 그대로 가능하도록 생략하는 방법!

* HTML5 와 Thymeleaf

예제

HTML5 에서는 무효한 문법이다

HTML5 문법에 맞게 변환

예제

HTML5 문법에 맞춰 변환한 코드

"data-속성명"

* HTML5 의 확장속성

data-속성명 = "기능"

* th-text vs th-utext

welcome = 안녕하세요.

data-th-text

escape 문자를 처리한다

HTML문을 많이 쓰일 수 있는 HTML 키워드에 해당하는 문자를 변환하여 HTML 해석에 영향을 끼치지 않게 하기 위함이다.

<b>안녕</b>하세요.

HTML Entity

(HTML 상수)

↳ HTML keyword와 충돌나는 문자를 표현할 때 사용하는 특별한 상수

HTML 키워드가 아닌 순수 문자로 취급하기 위함.

data-th-utext

unescaped

escape 문자로 처리하지 않는다

HTML 해석에 포함된다.

안녕하세요.

언뮤테스트가 아닌 HTML 문법으로 간주한다.

* Link

data-th-href="@{url}"

<a href="">

현재페이지 URL: http://localhost:8888/app/board/list

server root path ⇒ /

contex path ⇒ /app

(웹 애플리케이션 경로)

@{http://localhost:8888/app/board/view} → href="http://localhost:8888/app/board/list"

절대 경로

@{~/board/view} → href="/board/view"

서버 root path 기준

@{/board/view} → href="/app/board/view"

context path 기준

@{board/view} → href="board/view"

/app/board/board/view

현재 페이지 기준

69. HandlerMethodArgumentResolver 추가하기

request handler

파라미터 ← 생성해주는 객체

목적?

request handler의 파라미터 값을 제공하기 위해

```
String add(
    Board board,
    MultipartFile[] files,
    HttpSession session) {

    ≡

}
```

1번으로 제공해주는 파라미터 값

- ServletRequest
- ServletResponse
- HttpSession
- @RequestParam
- ⋮

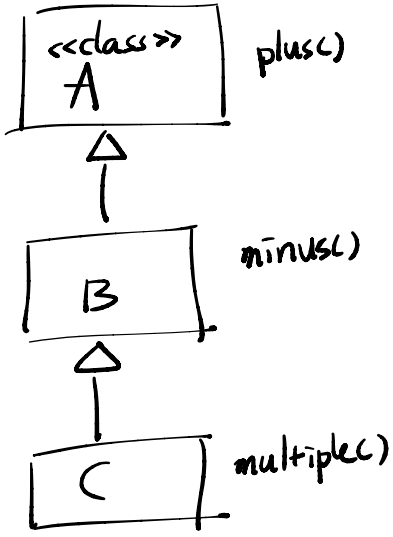
* 로그인 사용자의 정보를 리턴해주는 HandlerMethodArgumentResolver 만들기

```
public String add(
        Board board,
        MultipartFile[] files,
        @LoginUser User loginUser
) throws Exception {
    …
}
```

① LoginUser 애노테이션 정의

② LoginUserArgumentResolver 클래스 정의

* Class.isAssignableFrom()



A.class.isAssignableFrom(B.class)

A type의 레퍼런스에 B type의 인스턴스를 할당할 수 있는가?

```
A obj = new B();
```

A+α

```
obj.plus()
```

```
B obj ≠ new A();
```

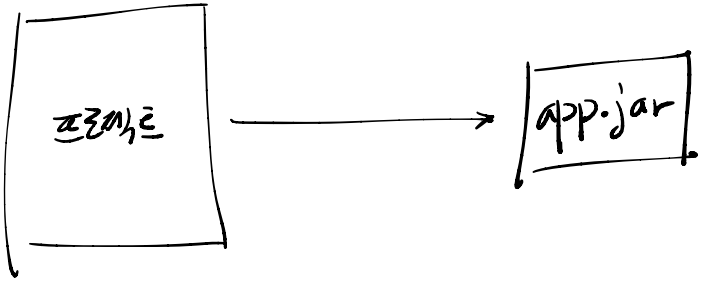
```
obj.minus();
obj.plus();
```

10. Jenkins 와 Docker를 이용하여 배치 자동화

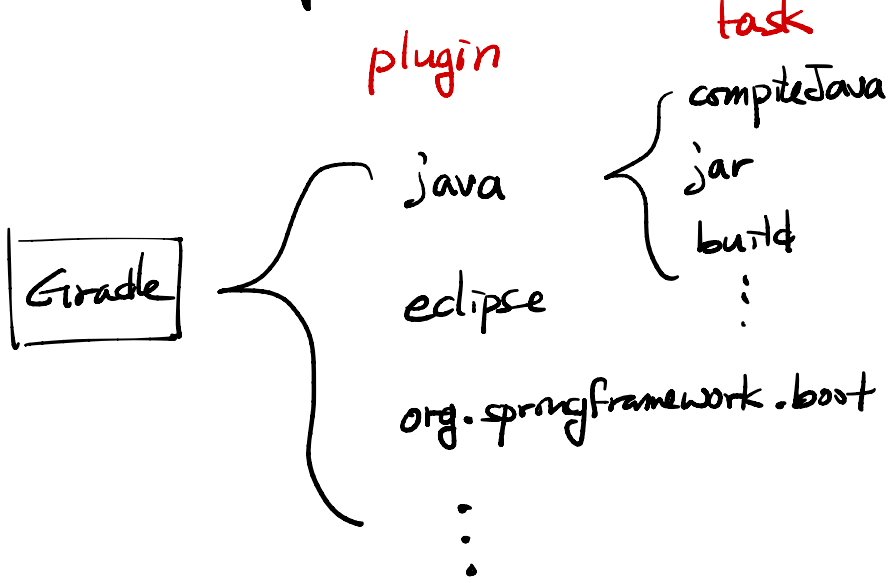
① 배포파일 생성 및 실행 확인

task 실행 확인

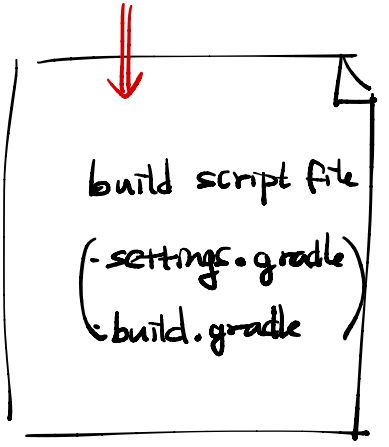
```
$ gradle build --console=plain

$ java -jar .../myapp.jar
```



* Gradle – plugin – task – build script file



(플러그인 태스크) 에 대한 설정



java { – }

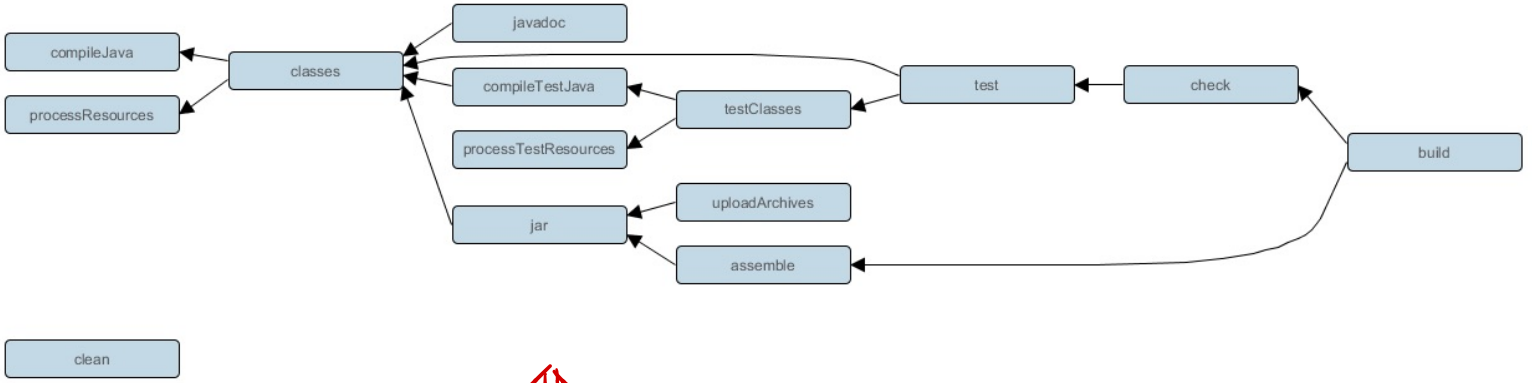
bootJar { – }

⋮

* Gradle 실행 과정

$ gradle build

← build task에 종속된 작업들이 순서대로 실행된다



⇑

'java' gradle 플러그인의 태스크와 종속관계

10. Jenkins 와 Docker를 이용하여 배치 자동화

② 스프링부트 설정 파일을 운영과 개발 모드로 분리

src/main/resources

application-dev.properties

application-prod.properties

⇑

실행할 때 지정

(dev
prod

```
$ java -Dspring.profiles.active=dev jar myapp.jar
```

JVM argument

```
$ java jar myapp.jar --spring.profiles.active=dev
```

Program Argument

* Program argument 와 JVM argument

공백(space)

$ java 클래스명 값 값 값

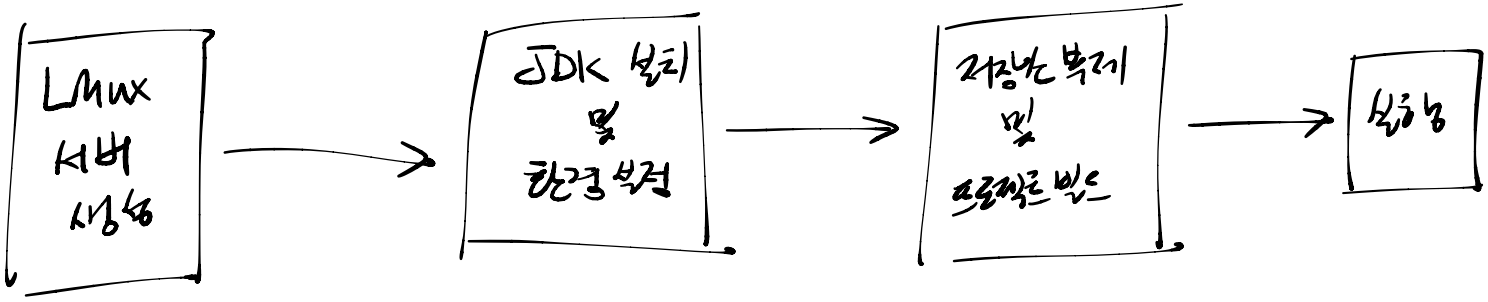
프로그램 아규먼트

$ java -D이름=값 -D이름=값 클래스명

JVM 아규먼트

10. Jenkins 와 Docker를 이용하여 배치 자동화

① 클라우드에서 리눅스 서버 구축의 어플리케이션 배치 및 실행



* 프로젝트 Repo 분리

bitcamp-mystudy (Repository)

- myapp (project) → 독립적인 Repository로 분리
  - 변경내역을 관리하기 쉽다
  - 빌드 및 배치를 다루기 쉽다.
- java-web (Project)
- java-lang (Project)

⋮

Repo. = Project

pull 과 push는

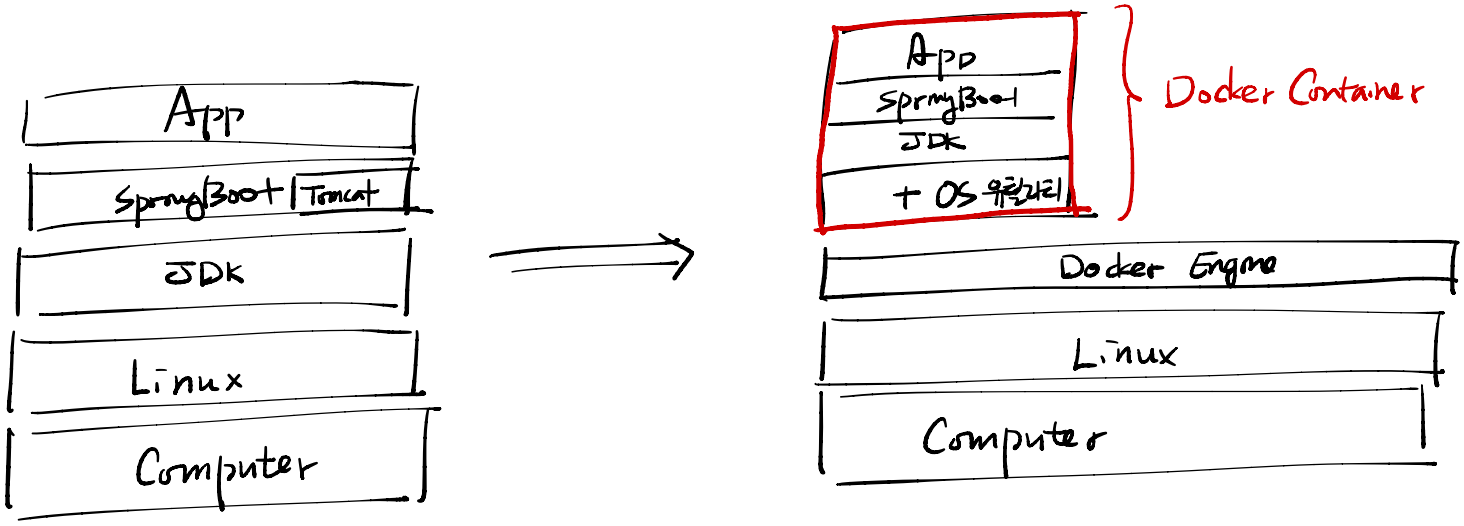
Repo. 단위로

수행된다

↳ 배치 관리가 불편

10. Jenkins 와 Docker를 이용하여 배치 자동화

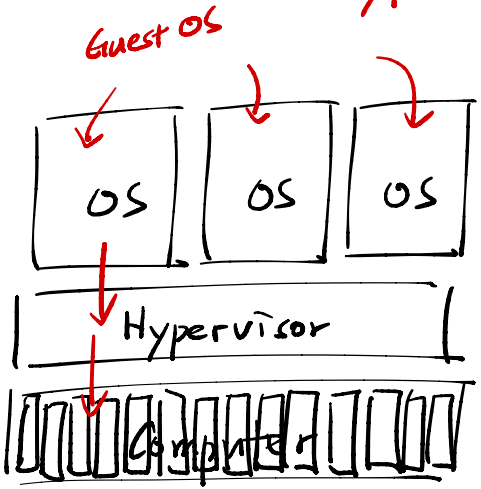
④ Docker 컨테이너에 배치하기



* 가상화 와 Docker 컨테이너

① Type 1 (bare metal)

예) Xen, KVM, Hyper-V



Guest OS + Computer를 사용하는 것을 통제 및 제어

예) 클라우드 서버

* 가상화

H/W를 논리적으로 여러 개로 분할하여 사용, 공유하는 기술

⇓

H/W 최대한 활용

⇓

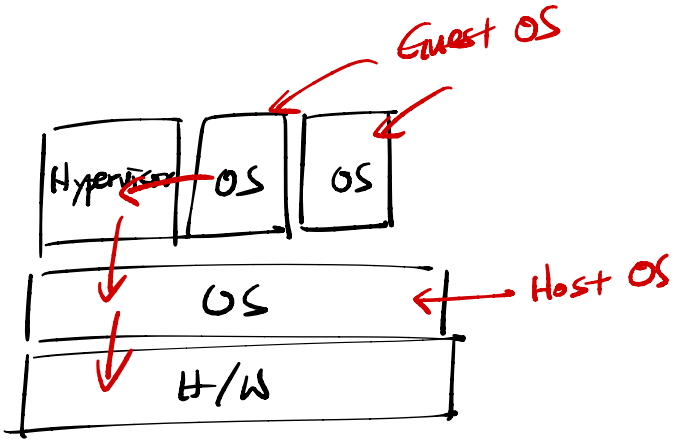
Who? (H/W 앞대 사용자 서버 운영자

여러 OS가 같은 H/W (cpu, memory, 내부 저장장치 등)를 효율적이 공유해서 사용하도록 제어

* 가상화 와 Docker 컨테이너

② Type 2

예) VMWare, VirtualBox, Parallels 등

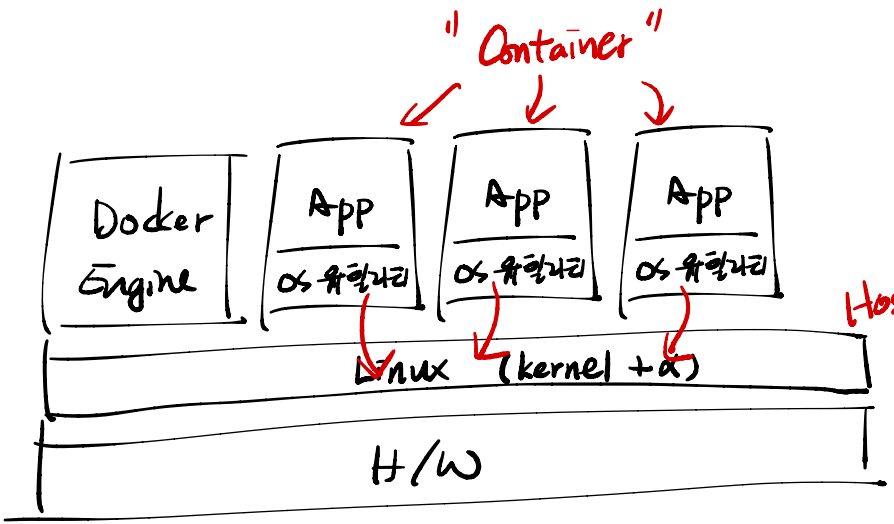


* 가상화 와 Docker 컨테이너

③ Container — 가상화의 단점을 보완

↳ OS를 전체 설치 → (HDD, RAM) 사용이 많다

중복 코드 로딩 (OS 커널)



Host OS의 핵심 기능을 공유

⇓

Guest OS를 전체 설치할 필요가 없다

⇓

설치파일 크기가 작다

⇓

HDD 사용량이 적다

OS kernel 로딩 필요없다

⇓

RAM 사용량이 적다

* container 실행

접속 터미널모드

docker **rm** -i -t ubuntu:14.04

image 이름 tag 이름

⇓

① local 에서 해당 도커 이미지를 찾는다

⇓ 없으면

② docker 허브에 이미지를 다운로드 한다

⇓

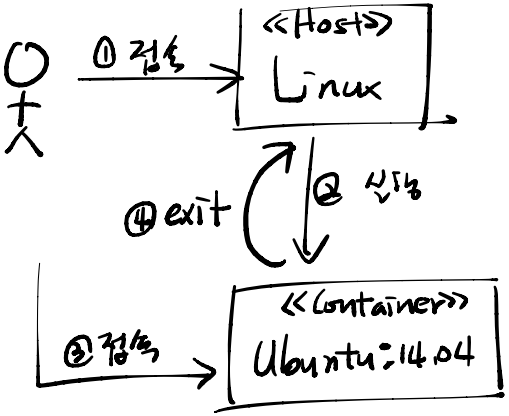
③ 이미지를 가지고 도커 컨테이너를 설치한다.

⇓

④ 컨테이너를 실행한다

⇓

⑤ 터미널 모드로 접속한다



* Docker Image 설치 및 실행

도커 이미지 다운로드

docker pull centos:7

도커 컨테이너 생성

docker create -i -t --name mycentos centos:7

도커 이미지

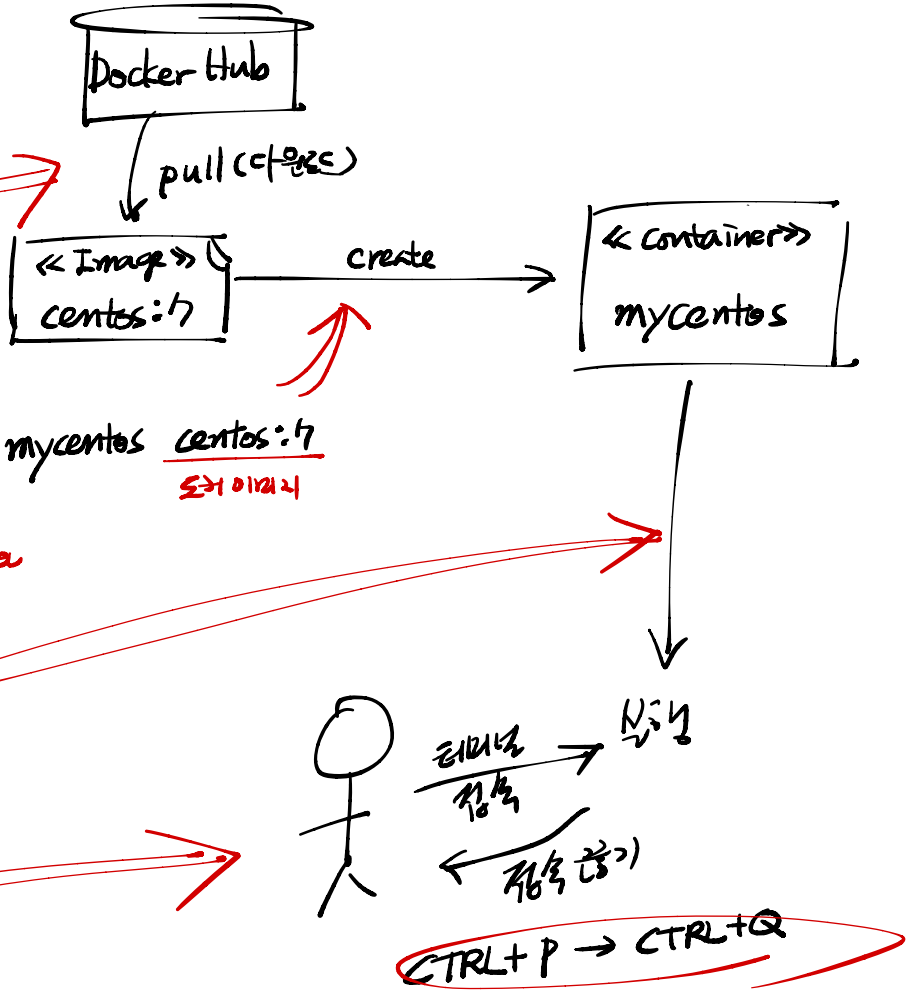
attach 하려면
이 옵션으로 생성해야 함

도커 컨테이너 실행

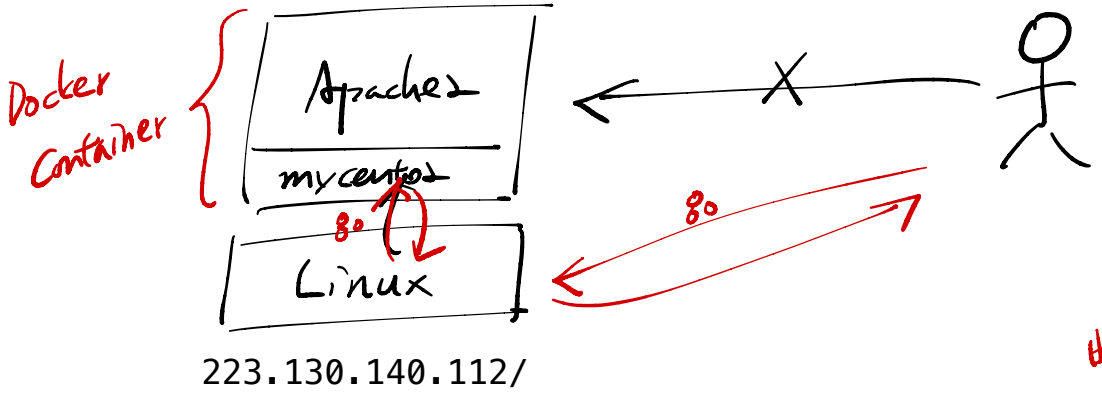
docker start mycentos

도커 컨테이너에 접속

docker attach mycentos



* 외부에서 컨테이너 접속하기 → 포트 포워딩으로 처리

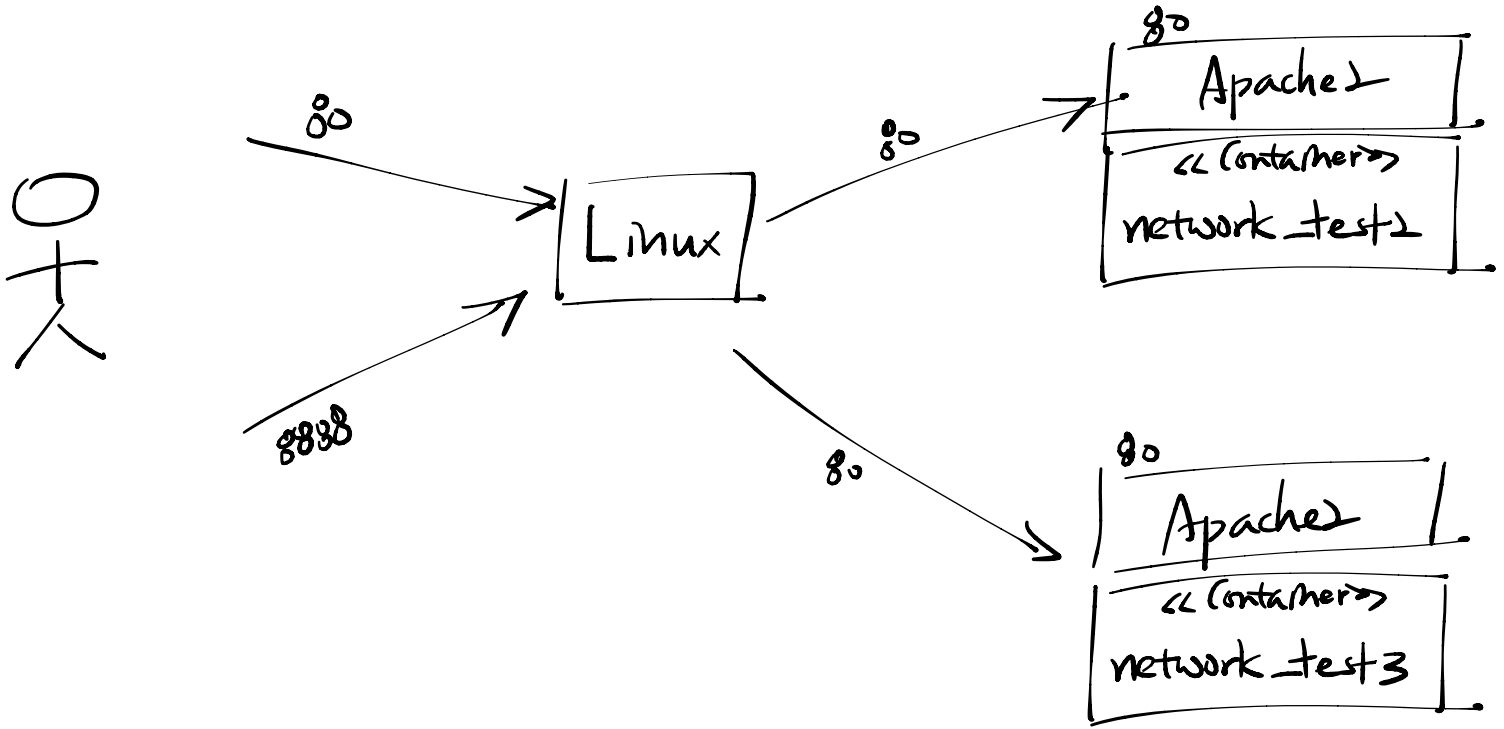


Host Container

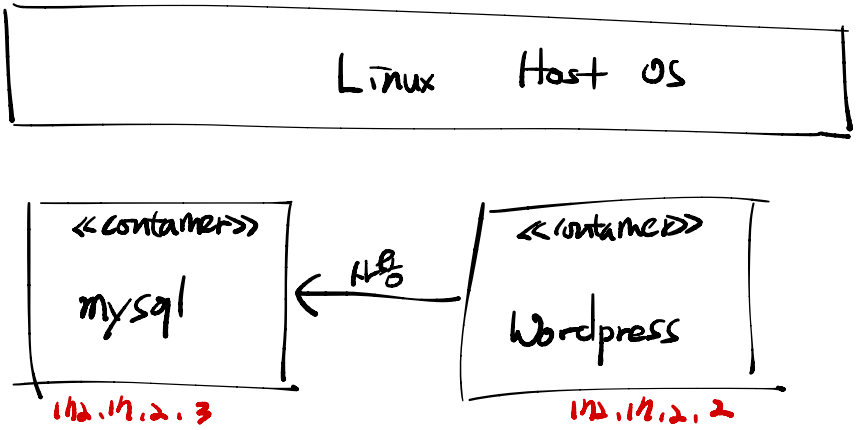
```
docker run -i -t --name mycentos2 -p 80:80 ubuntu:14.04
```

포트 포워딩설정

* 네트워크 컨테이너 생성하기 → 포트 포워딩으로 처리 II



* 컨테이너 활용 - Wordpress 구축

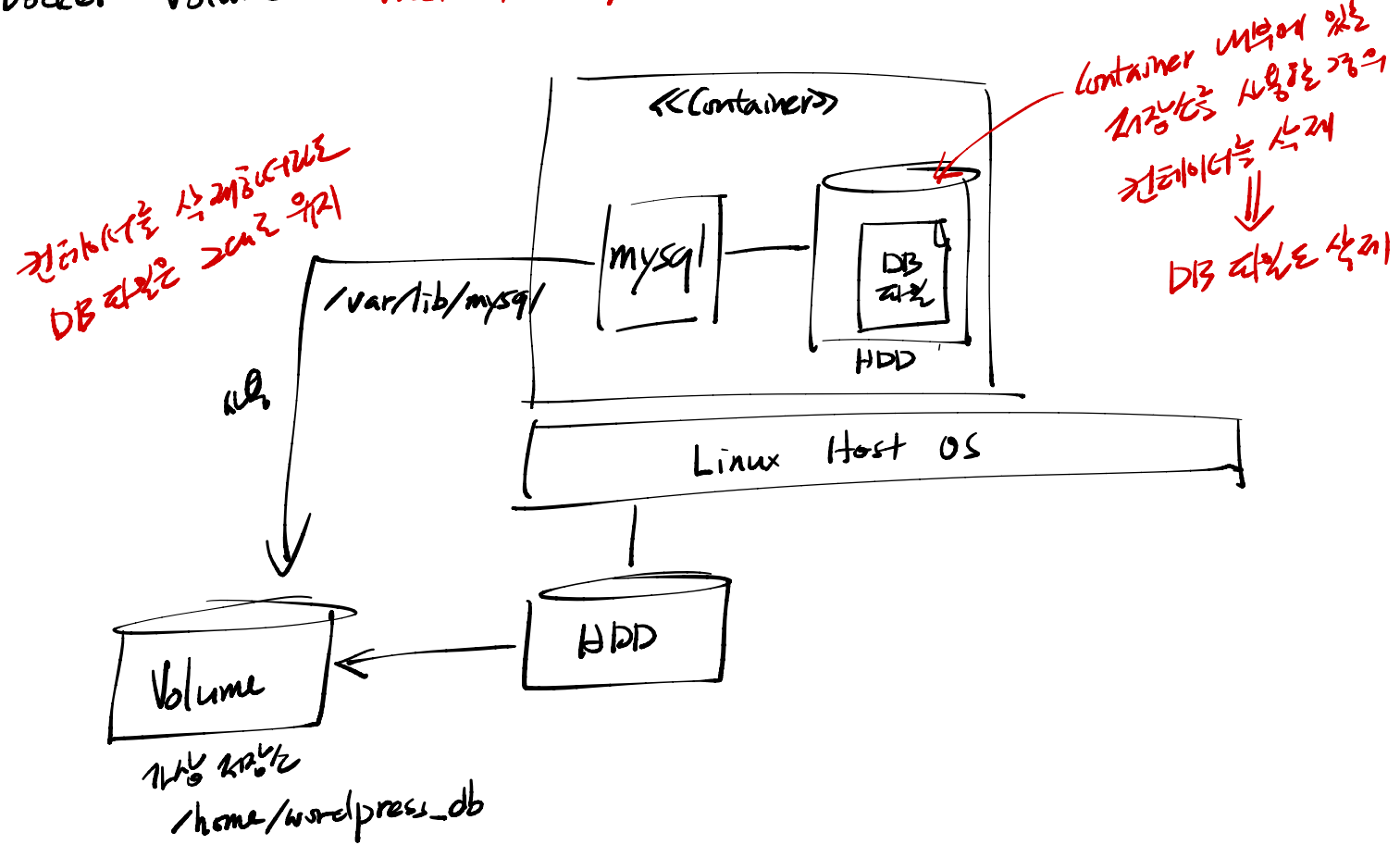


--link 컨테이너명:별명

↑

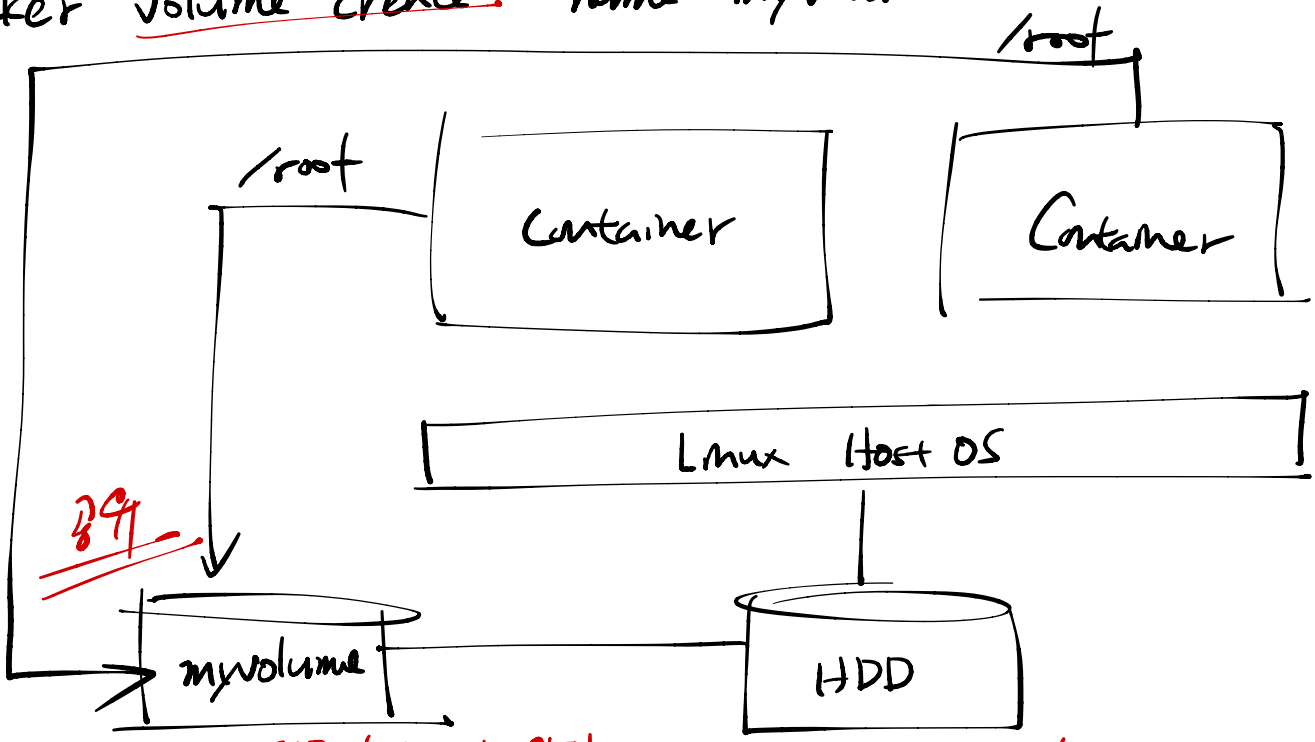
IP 주소를 알 필요가 없다.
컨테이너 이름으로 연결할 수 있다.

* Docker Volume - Host 의 디렉토리를 지정하는 경우



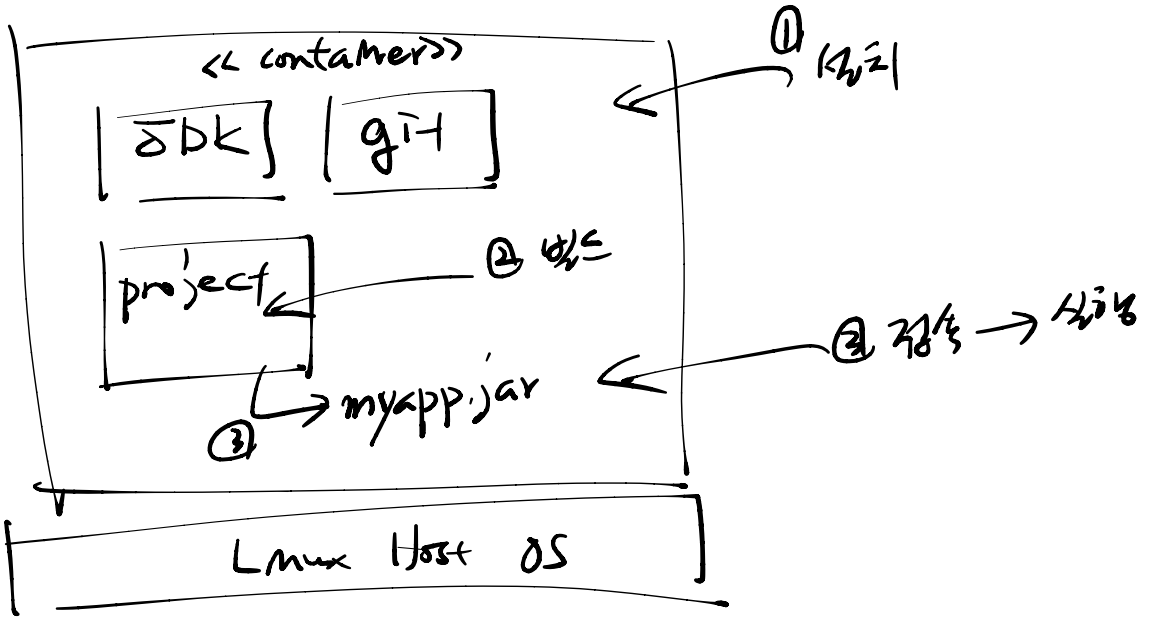
* Docker Volume — Docker에서 Volume 폴더 관리

docker volume create --name myvolume



위치? docker가 관리
↳ 디렉토리 위치를 특정할 필요가 없을 때!

* 애플리케이션 배치 container 만들기

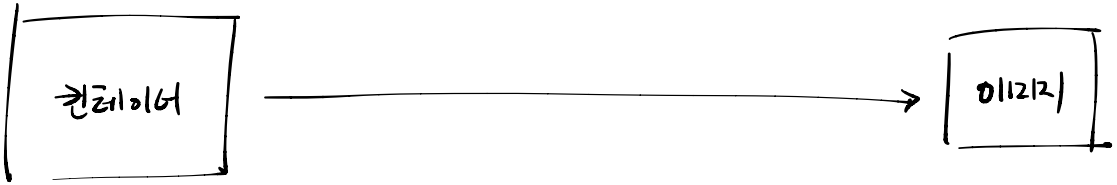


* 도커 이미지 만들기

↳ 컨테이너 설치 파일

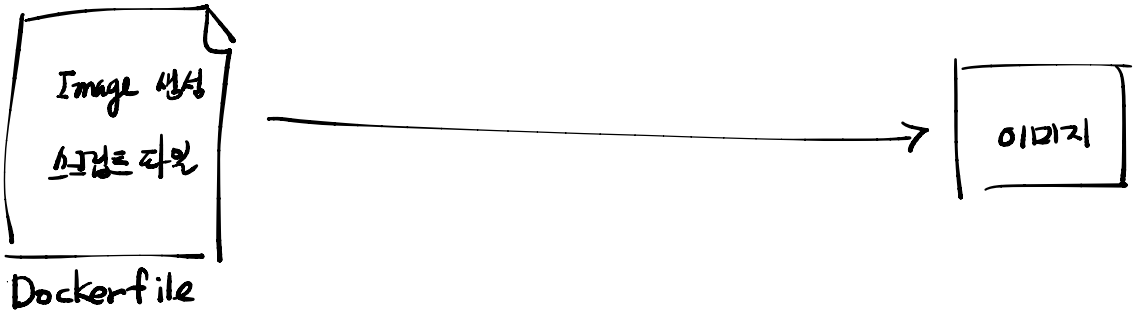
① docker commit 옵션 컨테이너명 REPOSITORY:TAG

예) sudo docker commit -a "제작자" -m "설명" 컨테이너명 이미지명:태그명

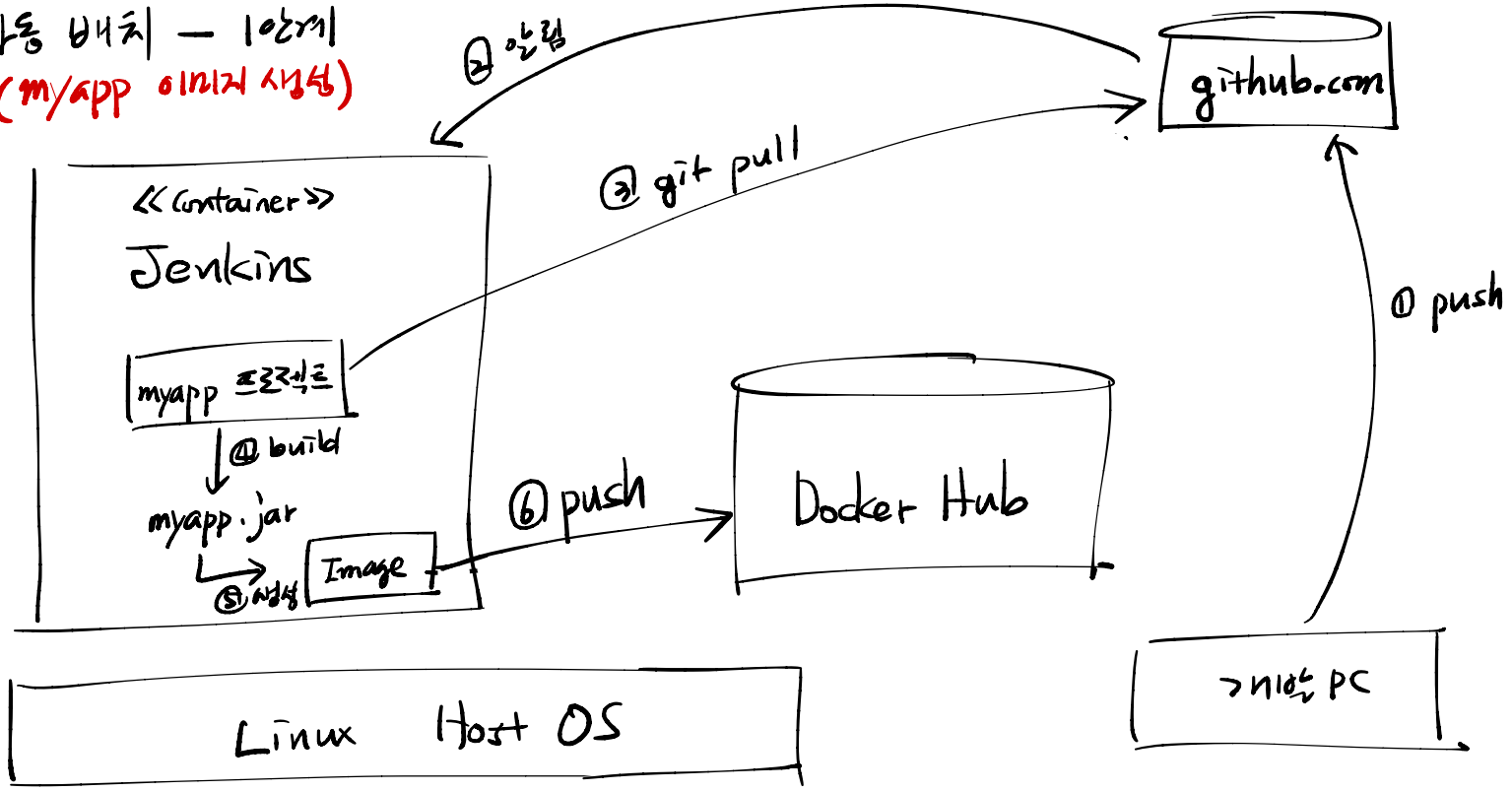


② docker build -t eomjinyoung/bitcamp:myapp .

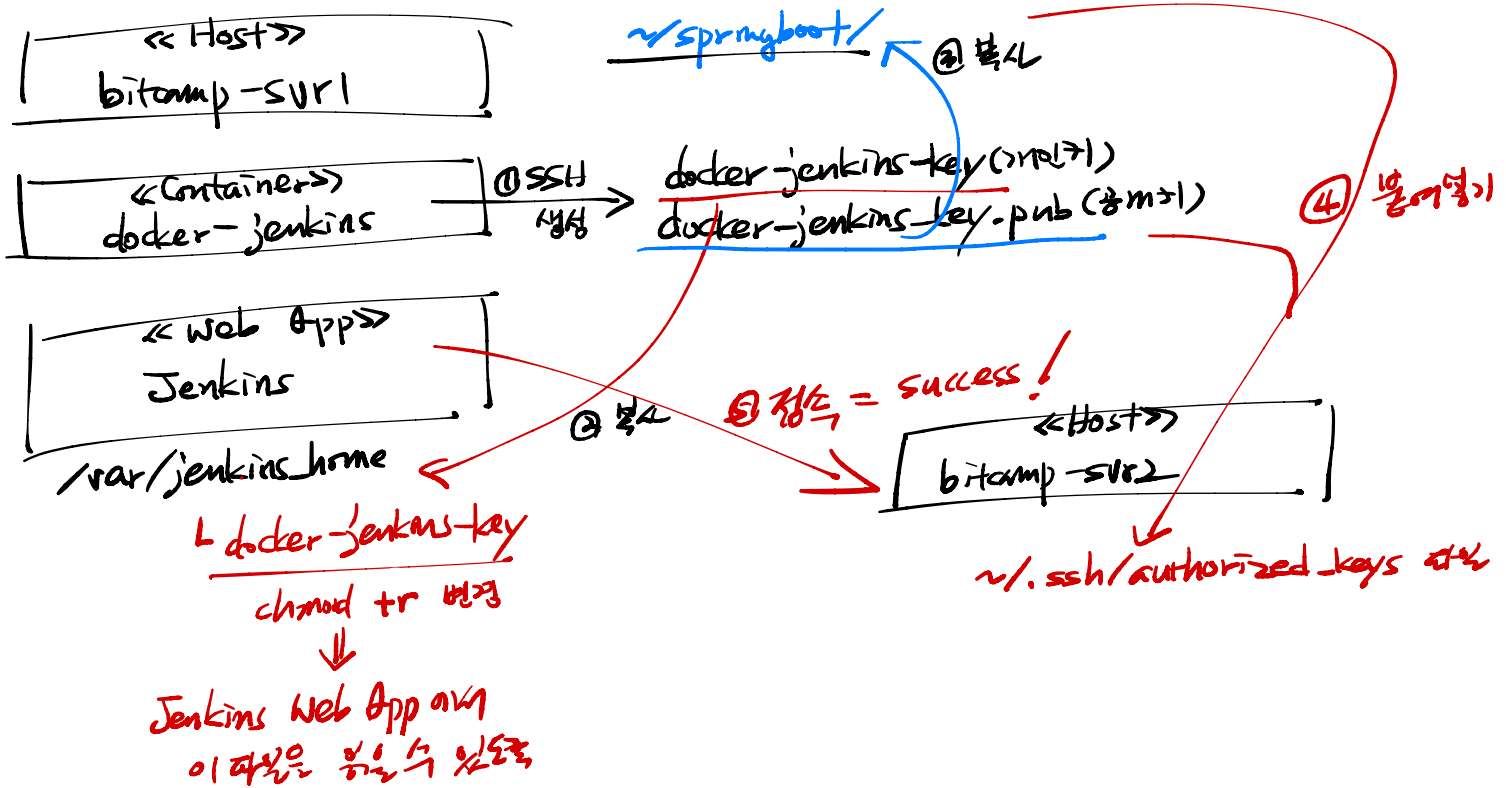
도커허브아이디 이미지명 태그명



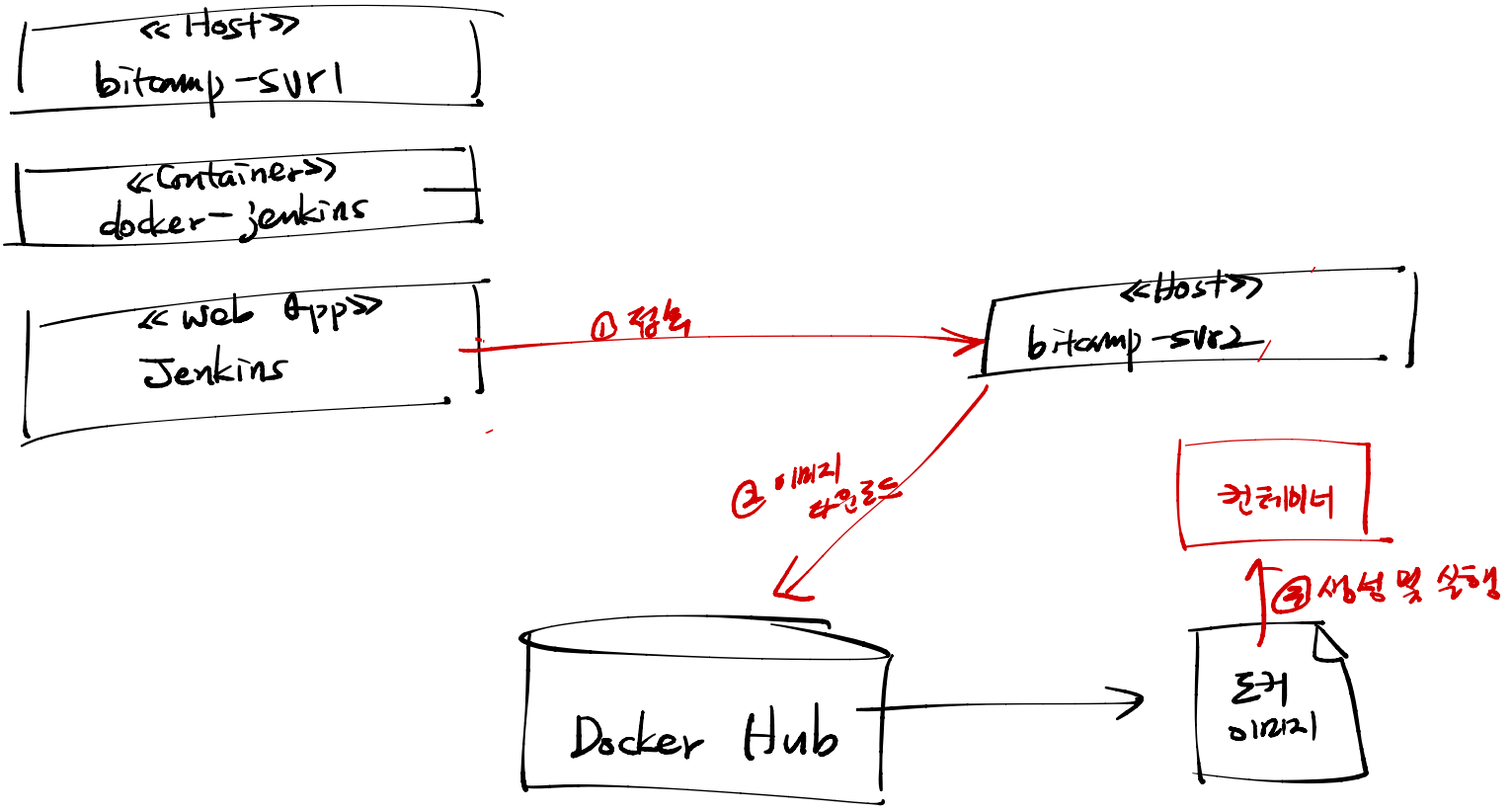
* 자동 배치 — 1단계
(myapp 이미지 생성)



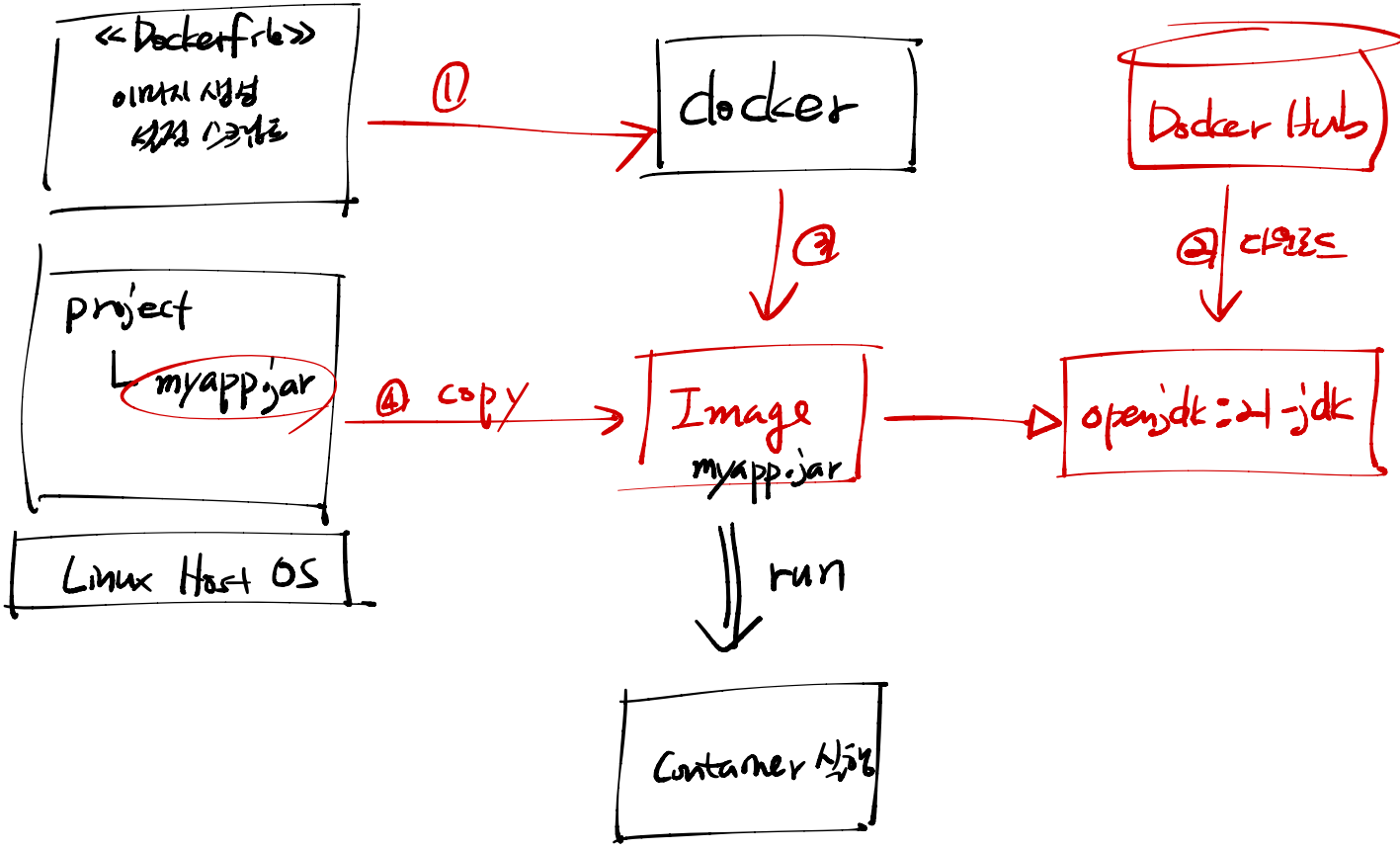
* 자동 배치 - 2단계 (배치 환경 설정)



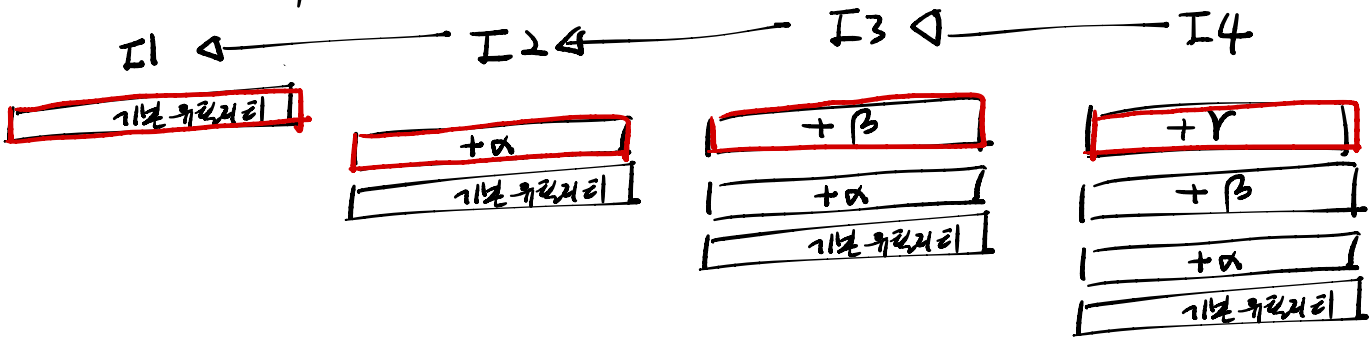
* 자동 배치 — 3단계 (배치)


<<Host>>
bitcamp-svr1
<<Container>>
docker-jenkins
<<web App>>
Jenkins
① 접속
<<Host>>
bitcamp-svr2
② 이미지 다운로드
Docker Hub
도커 이미지
③ 생성 및 실행
컨테이너

* 애플리케이션 배치 Container 만들기



* Docker 이미지 구조



Layerd 방식으로
Image 생성

컨테이너 생성

I1 다운로드
I2 다운로드
I3 다운로드
I4 다운로드

설치