클라우드 서비스 구축 및 연동구현

목차

PART 1. 도커 사용하기 - 4

- ✓ CHAPTER 1. 도커의 구조와 특징 - 5
- ✓ CHAPTER 2. 도커 설치 - 8
- ✓ CHAPTER 3. 도커 이미지 관리하기 - 15
- ✓ CHAPTER 4. 도커 컨테이너 관리하기 - 28
- ✓ CHAPTER 5. DockerFile 다루기 – 43

PART 2. 쿠버네티스 사용하기 - 55

- ✓ CHAPTER 6. 쿠버네티스의 구조와 특징 - 56
- ✓ CHAPTER 7. 쿠버네티스 설치 - 58
- ✓ CHAPTER 8. Pod 사용하기 - 70
- ✓ CHAPTER 9. Replica Set 사용하기 - 85
- ✓ CHAPTER 10. Deployment 사용하기 - 100
- ✓ CHAPTER 11. Service 사용하기 - 115

목차

PART 3. 애플리케이션 배포 설정 - 131

- ✓ CHAPTER 12. 사용량 관리하기 - 132
- ✓ CHAPTER 13. 스케줄링 설정하기 - 150
- ✓ CHAPTER 14. 컨트롤러 사용하기 - 162
- ✓ CHAPTER 15. 기타 오브젝트들 살펴보기 - 175

PART 1.
도커
사용하기

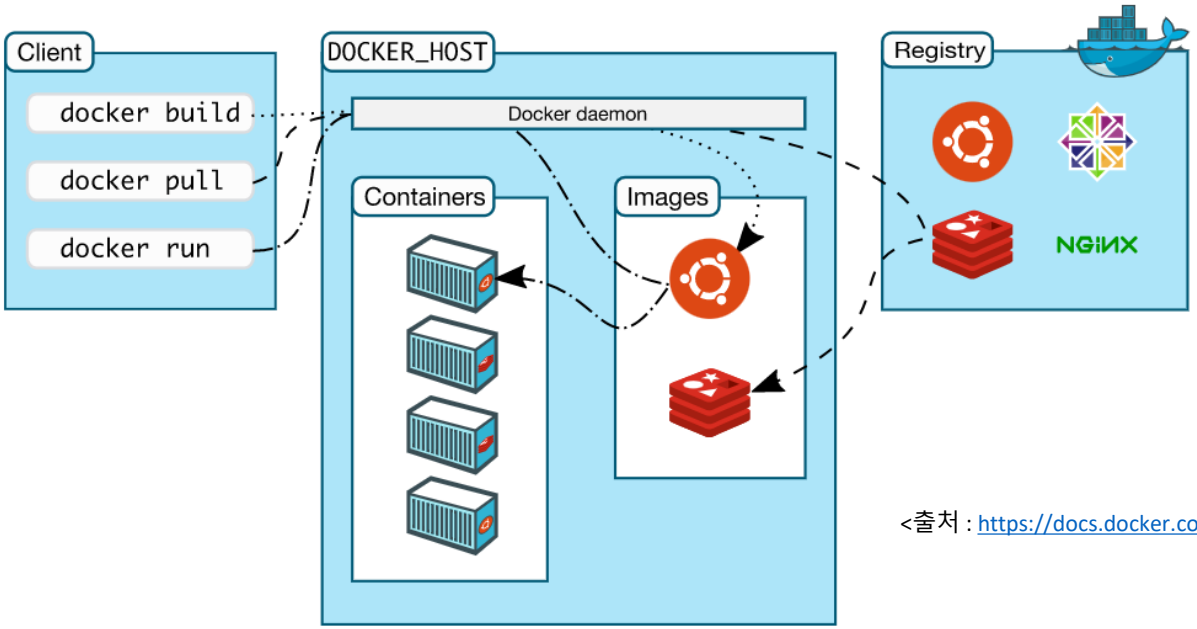
CHAPTER 1.
도커의 구조와 특징

도커의 구조와 특징

✓ 도커(Docker) 란?

애플리케이션을 **개발/배포/실행**하기 위한 플랫폼
컨테이너를 사용하여 애플리케이션 및 지원 구성 요소를 개발
리눅스 자체 기능을 사용하여 프로세스 단위의 격리 환경을 제공
배포 용량과 시간을 단축하고, **성능 손실을 최소화** 시킴

✓ 도커 구조



<출처 : https://docs.docker.com/get-started/overview/ >

도커의 구조와 특징

✓ 도커 엔진과 도커 데몬(dockerd)

외부에서 Docker API 요청을 받아 **Docker 객체와 서비스들을 관리**
컨테이너를 생성하고 관리하는 주체
도커 데몬 : 도커 프로세스가 실행되어 입력 받을 준비가 된 상태

✓ 도커 클라이언트(docker)

입력된 **명령어를 Docker API 형태로 도커 데몬에게 전달**
도커 엔진의 수행 결과를 반환 받아 사용자에게 출력

✓ 도커 레지스트리(Registry)

도커에서 사용되는 **이미지들을 저장하고 공유해주는 원격 저장소**
공개 레지스트리(Docker Hub)와 사설 레지스트리 사용 가능

CHAPTER 2.
도커 설치

도커 설치

✓ 도커 설치

다양한 운영체제에서 사용 가능(리눅스, 윈도우, 맥 OS X)
설치 및 설정 방법은 운영체제에 따라 조금씩 다르지만, 설치된 이후 사용방법은 동일함

✓ 공식 홈페이지

https://www.docker.com/

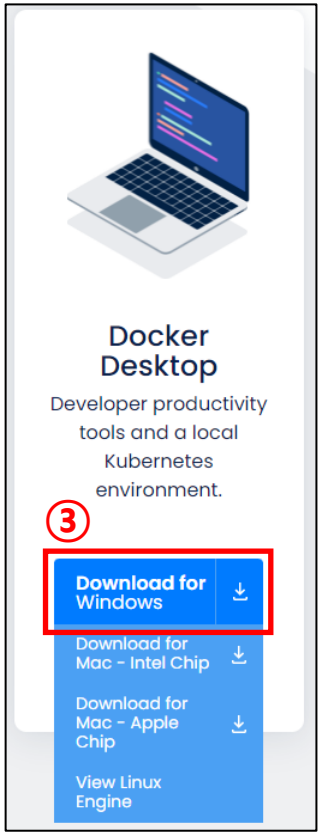
✓ 가이드 홈페이지

https://docs.docker.com/

도커 설치

✓ Docker Desktop for Windows 설치

① 도커 공식 홈페이지(https://www.docker.com) 접속
② 상단 메뉴 중 Developers 클릭
③ Download for Windows 클릭
④ 다운받은 설치 파일 실행



④ Docker Desktop Installer.exe

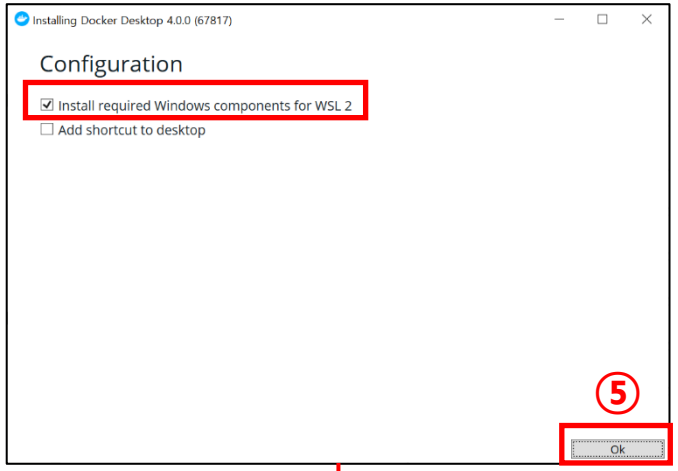
도커 설치

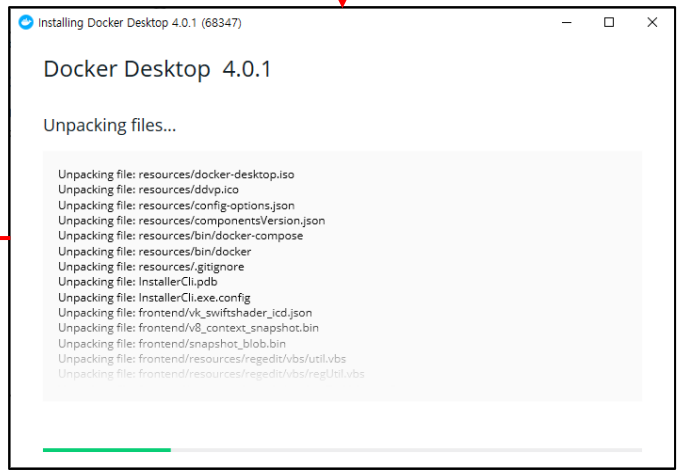
✓ Docker Desktop for Windows 설치

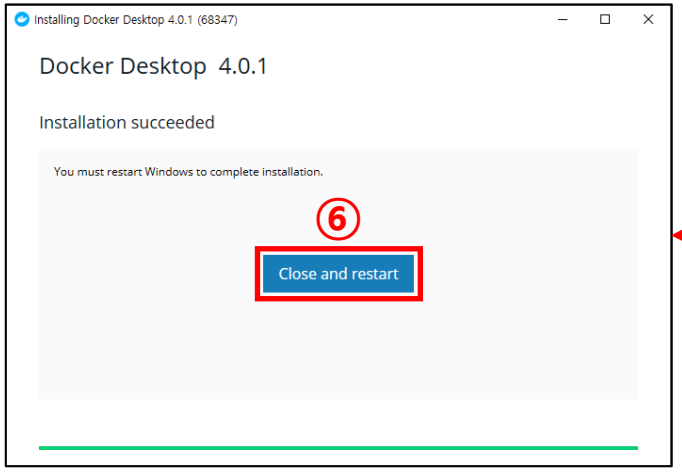
⑤ WSL2 설치 체크 후 OK

⑥ Unpacking 완료 후 윈도우 재부팅

※ WSL2(Windows Subsystem for Linux 2) :
윈도우에서 경량 가상화 기술을 사용해 리눅스를 구동할 수 있도록 도와주는 기능







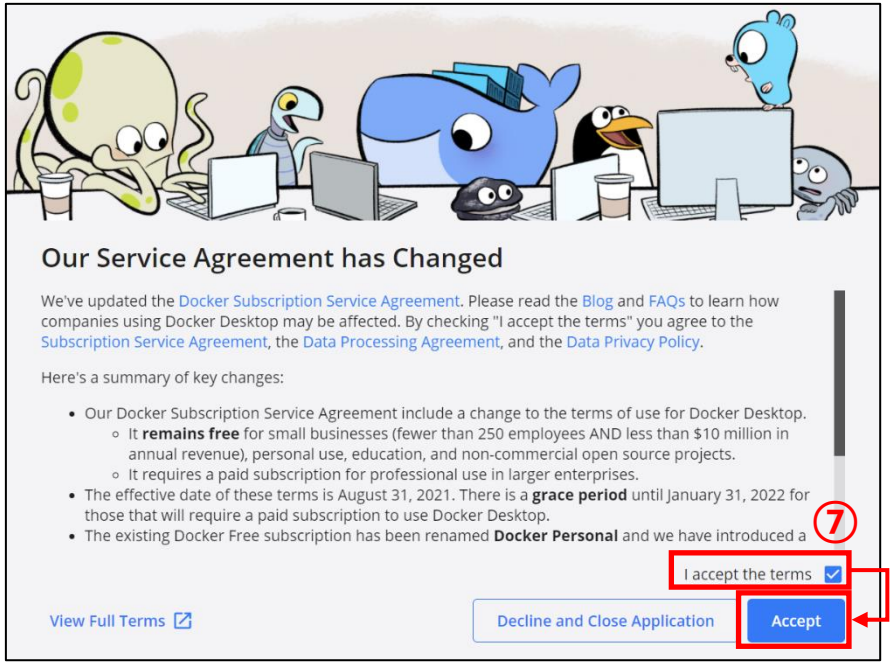
도커 설치

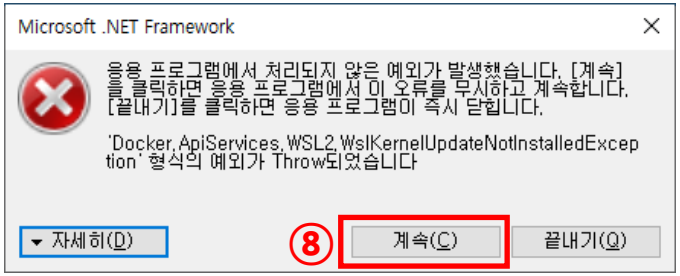
✓ Docker Desktop for Windows 설치

재부팅 후 잠시 기다리면 서비스 계약 변경 동의 창 팝업 됨

⑦ I accept the terms 체크 후 Accept 버튼 클릭

⑧ WSL2 오류 팝업 창에서 계속 버튼 클릭





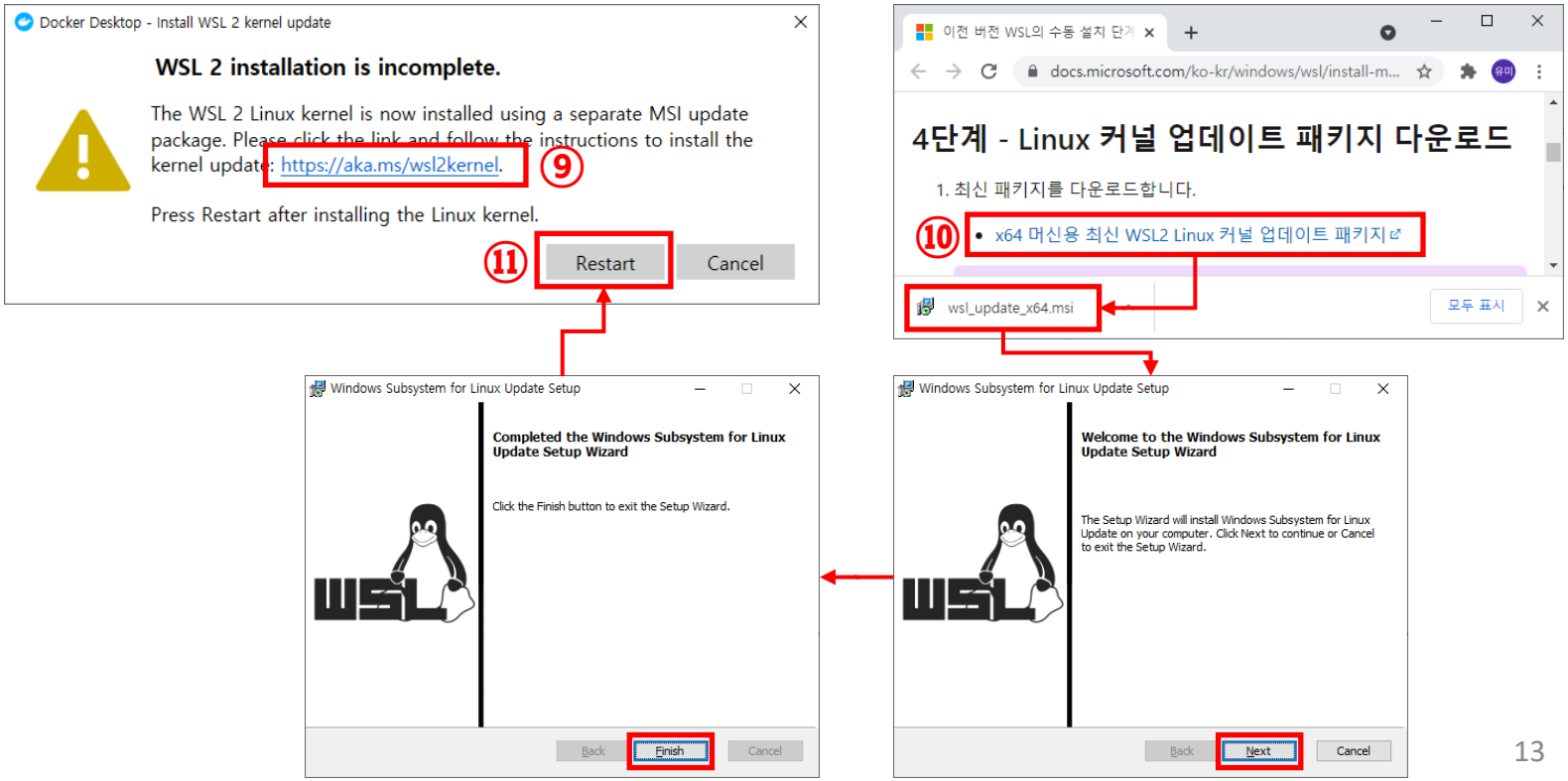
도커 설치

✓ Docker Desktop for Windows 설치

⑨ WSL2 설치 팝업 창에서 링크 클릭

⑩ Linux 커널 업데이트 패키지 다운로드 후 Next로 설치

⑪ WSL2 설치 팝업 창에서 Restart

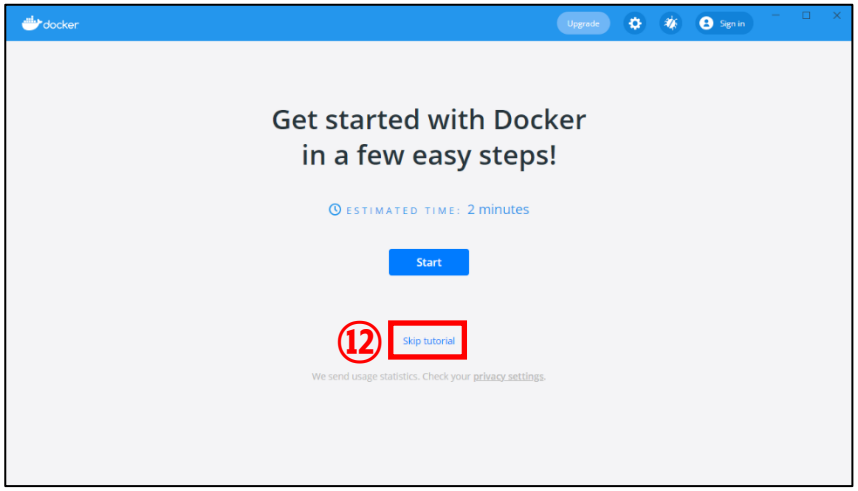


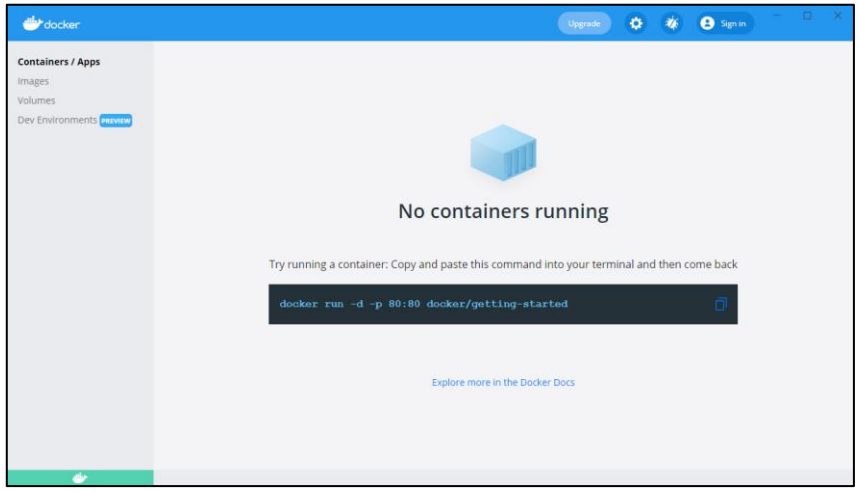
도커 설치

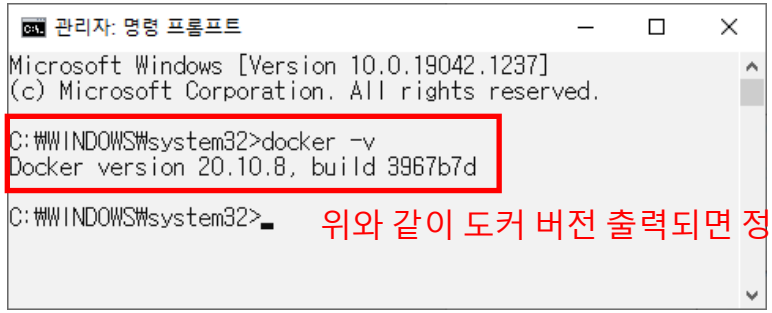
✓ Docker Desktop for Windows 설치

⑫ Docker 첫 실행 화면에서 Skip tutorial 클릭

⑬ 윈도우 명령 프롬프트(cmd) 에서 도커 버전 확인







위와 같이 도커 버전 출력되면 정상 설치 완료

CHAPTER 3.
도커 이미지 관리하기

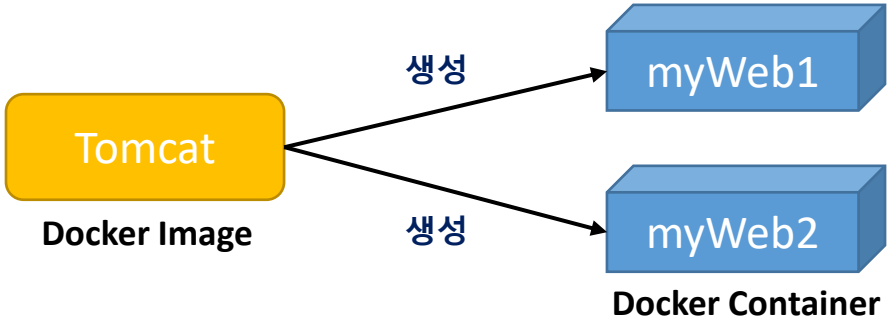
도커 이미지 관리하기

✓ 도커 이미지

컨테이너를 만들고 실행하기 위한 읽기 전용 파일(템플릿)
모든 **컨테이너는 이미지 기반으로 생성**됨
컨테이너 실행에 필요한 파일과 설정 값 등을 포함하고 있음

✓ 도커 이미지와 컨테이너

컨테이너는 필요한 파일과 설정을 이미지에서 읽기전용으로 가져다 사용하고, **변경된 사항만 컨테이너 계층에 별도 저장**
하나의 이미지로 여러 컨테이너에서 사용 가능



도커 이미지 관리하기

✓ 도커 허브(Docker Hub)

도커에서 공식적으로 제공하고 있는 중앙 이미지 저장소
도커 계정을 가지고 있으면 누구든 이미지 업로드/다운로드가 가능
기본적인 리눅스 운영체제부터 웹서버, 데이터베이스, 각종 애플리케이션 등의 다양한 종류의 이미지를 도커 레지스트리에서 내려 받아 컨테이너로 생성

✓ 도커 이미지 이름

[저장소이름/]이미지이름[:태그]

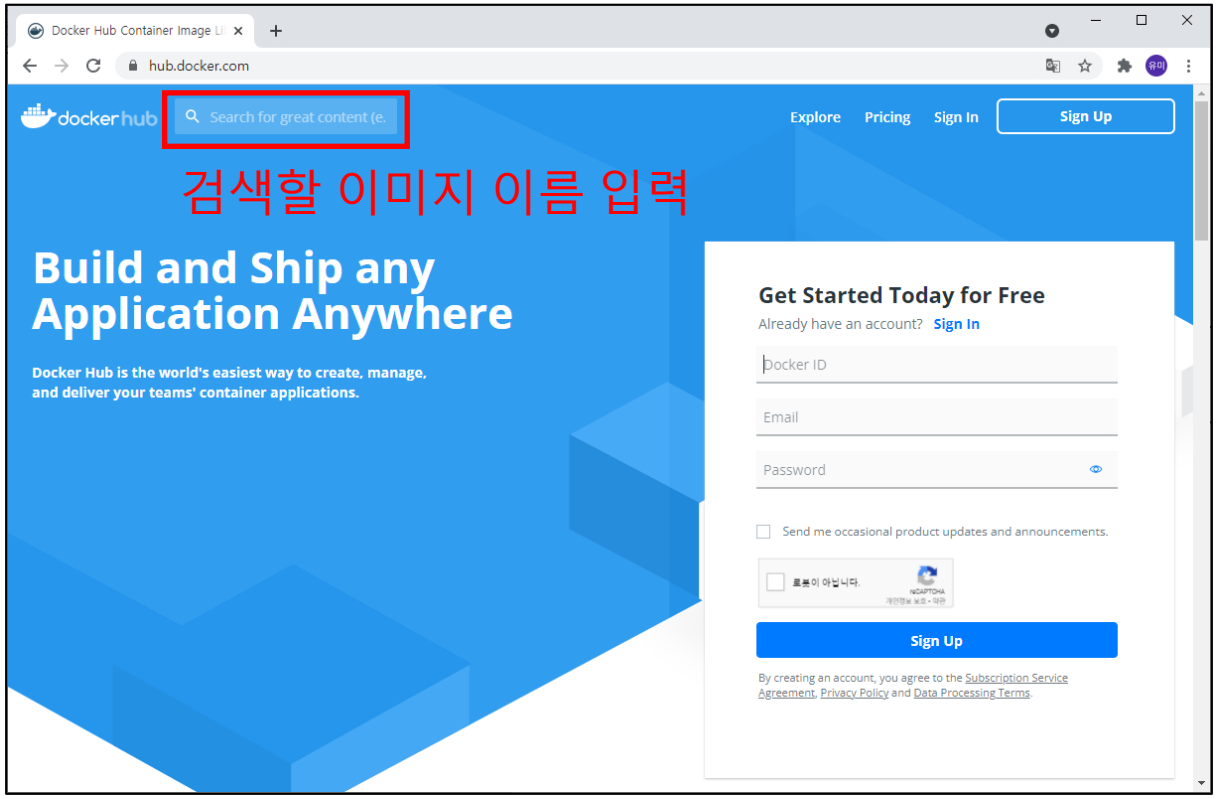
- 저장소 이름 생략 시 도커 허브 공식 이미지로 인식
- 태그는 버전을 나타냄
- 태그 생략 시 최신 버전(latest)으로 인식

도커 이미지 관리하기

✓ 이미지 검색 방법 1

도커 허브 공식 홈페이지에서 검색

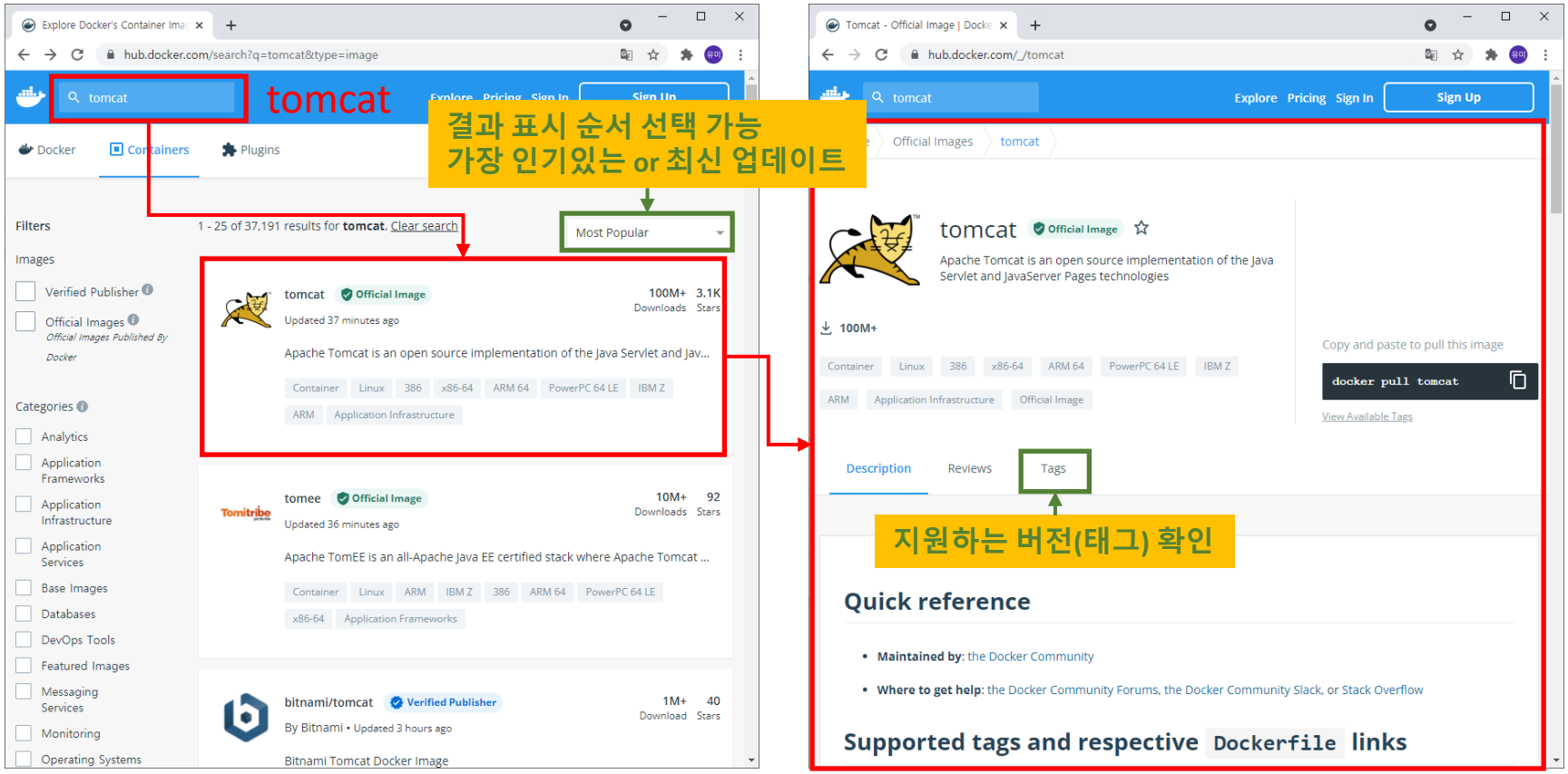
https://hub.docker.com/



도커 이미지 관리하기

✓ 이미지 검색 방법 1

검색 결과 클릭 시 상세 정보 및 지원 버전 확인 가능



도커 이미지 관리하기

✓ 이미지 검색 방법 2

도커 엔진에서 명령어로 검색
docker search <키워드>

> docker search tomcat

```
C:₩>docker search tomcat
NAME                      DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
tomcat                    Apache Tomcat is an open source implementati    3145      [OK]
tomee                     Apache TomEE is an all-Apache Java EE certif    93        [OK]
dordoka/tomcat            Ubuntu 14.04, Oracle JDK 8 and Tomcat 8 base    58                   [OK]
kubeguide/tomcat-app      Tomcat image for Chapter 1                      31
consol/tomcat-7.0         Tomcat 7.0.57, 8080, "admin/admin"              18                   [OK]
cloudesire/tomcat         Tomcat server, 6/7/8                            15                   [OK]
aallam/tomcat-mysql       Debian, Oracle JDK, Tomcat & MySQL              13                   [OK]
arm32v7/tomcat            Apache Tomcat is an open source implementati    11
rightctrl/tomcat          CentOS , Oracle Java, tomcat application ssl    7                    [OK]
arm64v8/tomcat            Apache Tomcat is an open source implementati    6
maluuba/tomcat7-java8     Tomcat7 with java8.                             6
unidata/tomcat-docker     Security-hardened Tomcat Docker container.      5                    [OK]
```

- NAME : [저장소이름/]이미지이름
- DESCRIPTION : 설명
- STARS : 도커 사용자로부터 즐겨찾기 된 수
- OFFICIAL : 공식 이미지
- AUTOMATED : 자동화 빌드 설정 여부

도커 이미지 관리하기

✓ 이미지 다운로드

docker pull [저장소이름/]<이미지이름>[:태그]

> **docker pull tomcat:8.5**

```
C:₩>docker pull tomcat:8.5
8.5: Pulling from library/tomcat
955615a668ce: Pull complete
2756ef5f69a5: Pull complete
911ea9f2bd51: Pull complete
27b0a22ee906: Pull complete
785dffb36c6c: Pull complete
3fccb14f0369: Pull complete
4a0c30fedd9c: Pull complete
54a2fe1e7768: Pull complete
99fc40810699: Pull complete
17fef45f952c: Pull complete
Digest: sha256:658c3c54069e7c969803212191f654d0afb30660cd6702cabc3811ce753d1939
Status: Downloaded newer image for tomcat:8.5
docker.io/library/tomcat:8.5
```

도커 이미지 관리하기

✓ 다운받은 이미지 목록 조회

docker images

> docker images

```
C:₩>docker images
REPOSITORY    TAG       IMAGE ID       CREATED       SIZE
tomcat        8.5       551efb60a0a2   2 weeks ago   678MB
```

IMAGE ID : 16진수 형태의 고유 식별값

도커 이미지 관리하기

✓ 이미지 세부 정보 조회

docker image inspect <IMAGE ID>
image id 는 목록 중 유일하게 구분될 수 있게만 작성하면 됨

> docker image inspect 551e

```
C:₩>docker image inspect 551e
[
    {
        "Id": "sha256:551efb60a0a200cb11039fe4e29a2f78c2acabd4604e5af8cd345e6ce4486a99",
        "RepoTags": [
            "tomcat:8.5"
        ],
        "RepoDigests": [
            "tomcat@sha256:658c3c54069e7c969803212191f654d0afb30660cd6702cabc3811ce753d1939"
        ],
        "Parent": "",
        "Comment": "",
        "Created": "2021-09-14T18:02:37.434645185Z",
        "Container": "5a74e9c47e6dfbe46e2ed2e561e50c499247737aed5cceb17f0cb4a7264c774a",
        "ContainerConfig": {
            "Hostname": "5a74e9c47e6d",
```

도커 이미지 관리하기

✓ 도커 이미지 추출

도커 이미지를 별도의 파일로 저장할 때 사용
docker save -o <파일명> <이미지명>[:태그]
-o 옵션 : 추출될 파일명을 지정

> **docker save -o d:₩₩tomcat_image tomcat:8.5**

```
C:₩>docker save -o d:₩₩tomcat_image tomcat:8.5

C:₩>_
```

D: 드라이브에 파일 생성 확인

tomcat_image	2021-09-29 오전 7:22	파일	674,605KB

도커 이미지 관리하기

✓ 도커 이미지 추출

추출된 파일을 다시 도커 내 이미지로 로드
docker load -i <파일명>
-i 옵션 : 로드할 파일명을 지정

> **docker load -i d:₩₩tomcat_image**

```
C:₩>docker load -i d:₩₩tomcat_image
Loaded image: tomcat:8.5
```

도커 이미지 관리하기

✓ 도커 이미지 이름 추가

기존 이미지를 새로운 이름으로 추가

docker tag [기존저장소/]<기존이미지명>[:기존태그]
[새로운저장소/]<새로운이미지명>[:새로운태그]

> docker tag tomcat:8.5 mytom:1

```
C:₩>docker tag tomcat:8.5 mytom:1

C:₩>docker images
REPOSITORY   TAG        IMAGE ID       CREATED        SIZE
mytom        1          551efb60a0a2   2 weeks ago    678MB
tomcat       8.5        551efb60a0a2   2 weeks ago    678MB
```

IMAGE ID는 같고 이름만 다른 이미지 추가됨

도커 이미지 관리하기

✓ 도커 이미지 삭제

docker rmi <이미지이름:태그 or IMAGE ID>

> **docker rmi mytom:1**

```
C:₩>docker rmi mytom:1
Untagged: mytom:1
```

이름 추가(tag)된 이미지가 있으면 이미지는 삭제되지 않고 이름만 제거(untag) 됨

> **docker rmi tomcat:8.5**

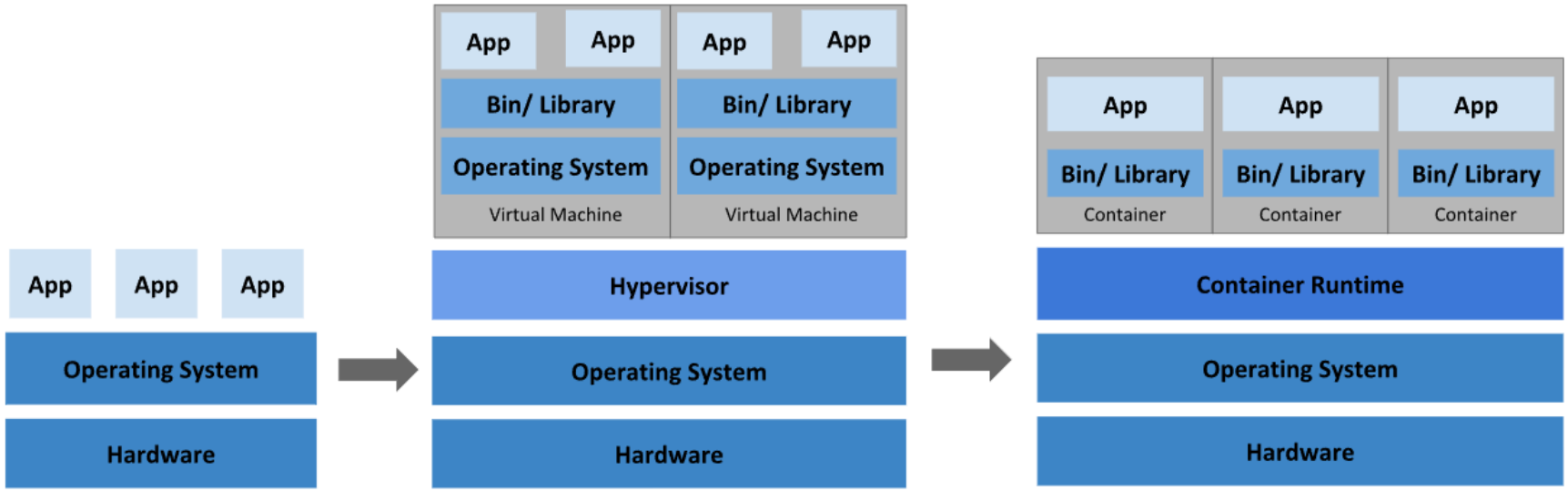
```
C:₩>docker rmi tomcat:8.5
Untagged: tomcat:8.5
Deleted: sha256:551efb60a0a200cb11039fe4e29a2f78c2acabd4604e5af8cd345e6ce4486a99
Deleted: sha256:8a8887e79df466c7590dd80c5d30de0cb7b127470883202b2d6664a026a2d54d
Deleted: sha256:d7d66a00b3cd43b2d195dca6fba2a726a2980e75cdeb94217ad0f5b579c086e7
Deleted: sha256:dce75d71440205925bab1b50a73d42fd6866631b998ad563da8047f67163ddf6
Deleted: sha256:25f34361d25b41cf15bcefe00e320772675539d74547001c46dce046b7ad787e
Deleted: sha256:5ab2e6c924e94042dc07fd33ddb68ad2bd87bb4f25f59f32e08be319700edce9
Deleted: sha256:8ef2b7d4d3c6fa6899e53a70ddeb662fd538fbb8e8ee8bf89341c806caa85d9f
Deleted: sha256:7c8e6d275c16b312f3a9fbd14dac058e0925c3071c61f85c14dd85d070567aa2
Deleted: sha256:dde6172a7948c5a7c96cc46703d13946bfd47b726decc1acf48abeda64f32ac4
Deleted: sha256:f11c0e4e16dc4cc8c0ffac9ac15db9a7c5091f16a86d53cab55306be2d4891a8
Deleted: sha256:799760671c382bd2492346f4c36ee4033cf917400be4354c8b096ecef88df34b
```

CHAPTER 4.
도커 컨테이너 관리하기

도커 컨테이너 관리하기

✓ 도커 컨테이너

도커 이미지의 실행 가능한 인스턴스
개별 애플리케이션의 실행에 필요한 실행환경을 독립적으로 운용



<일반> <가상화> <컨테이너>

<이미지 출처 : https://kubernetes.io/ko/docs/concepts/overview/what-is-kubernetes/ >

도커 컨테이너 관리하기

✓ 컨테이너 생성

docker run [옵션] [저장소이름/]<이미지이름>[:버전]
다운받은 이미지가 없는 경우 자동으로 다운받아서 생성됨
이미지 다운로드(pull) + 컨테이너 생성(create) + 실행(start) + 접속(attach) 까지 한번에 수행됨

[옵션]

옵션	설명
-d	백그라운드에서 컨테이너 실행
-i -t	컨테이너 내부로 진입하여 컨테이너와 상호 입출력
--name <이름>	컨테이너에 이름 지정(지정하지 않으면 16진수 임의 값으로 이름 생성됨)
-p <호스트Port>:<컨테이너Port>	외부에서 컨테이너 내부로 접근할 수 있게 호스트 PC 의 포트와 컨테이너 내부 포트를 연결
-e <환경변수>=<값>	환경변수 설정
--link	다른 컨테이너에 링크 추가

도커 컨테이너 관리하기

✓ 컨테이너 생성

> **docker run -d --name myweb -p 80:8080 tomcat:8.5.72-jdk8**

```
C:₩>docker run -d --name myweb -p 80:8080 tomcat:8.5.72-jdk8
Unable to find image 'tomcat:8.5.72-jdk8' locally
8.5.72-jdk8: Pulling from library/tomcat
df5590a8898b: Pull complete
705bb4cb554e: Pull complete
519df5fceacd: Pull complete
ccc287cbeddc: Pull complete
39a2961e8351: Pull complete
a12df774715e: Pull complete
08f28107b8b3: Pull complete
15e524c21b25: Pull complete
3397cebc9b31: Pull complete
998466ebbeab: Pull complete
Digest: sha256:d26b746135cb7fda083bd18dc6efd1385ef917f75d55c58cca675cdc324e939b
Status: Downloaded newer image for tomcat:8.5.72-jdk8
e95fe3d8aff730ae4f22d2d18459d1fd05b2fff57e7614bf5c56864f96a66ea5
```

도커 컨테이너 관리하기

✓ 컨테이너 확인

docker ps [옵션]

> **docker ps -a**

```
C:₩>docker ps -a
CONTAINER ID   IMAGE                 COMMAND             CREATED          STATUS          PORTS                   NAMES
e95fe3d8aff7   tomcat:8.5.72-jdk8    "catalina.sh run"   30 seconds ago   Up 26 seconds   0.0.0.0:80->8080/tcp    myweb
```

항목	설명
CONTAINER ID	컨테이너에 자동 할당되는 고유 ID 중 일부 출력
IMAGE	컨테이너 생성 시 사용된 이미지
COMMAND	컨테이너 시작 시 자동으로 실행될 명령어 대부분 이미지에 미리 내장되어 있음 컨테이너 생성 시 변경 가능
CREATED	컨테이너 생성 후 흐른 시간
STATUS	컨테이너 상태
PORTS	컨테이너에 개방된 포트, 호스트와 연결된 포트를 출력
NAMES	컨테이너 생성 시 지정한 이름(--name 옵션)

도커 컨테이너 관리하기

✓ 컨테이너 상세 내용 조회

docker inspect <컨테이너 이름 or ID>

> **docker inspect myweb**

```
C:₩>docker inspect myweb
[
    {
        "Id": "e95fe3d8aff730ae4f22d2d18459d1fd05b2fff57e7614bf5c56864f96a66ea5",
        "Created": "2021-10-10T13:28:33.4140483Z",
        "Path": "catalina.sh",
        "Args": [
            "run"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 7492,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2021-10-10T17:22:06.2337005Z",
            "FinishedAt": "0001-01-01T00:00:00Z"
        },
```

도커 컨테이너 관리하기

✓ 컨테이너 실행/종료

docker start/stop/restart <컨테이너 이름 or ID>

```
C:₩>docker stop myweb
myweb
```

```
C:₩>docker start myweb
myweb
```

```
C:₩>docker restart e95f
e95f
```

✓ 컨테이너로 파일 복사

docker cp [옵션] <원본파일경로> <복사할경로>
경로가 컨테이너 내부일 경우 <컨테이너명:경로> 로 지정

> **docker cp d:₩test.txt myweb:/root**

```
C:₩>docker cp d:₩test.txt  myweb:/root
```

도커 컨테이너 관리하기

✓ 컨테이너 명령어 실행

docker exec [옵션] <컨테이너이름 or ID> <명령어>
쉘을 실행하면 컨테이너에 접근 가능
쉘 접근 후 다양한 명령어 사용 가능(종료 시 exit)

> **docker exec -it myweb /bin/bash**

```
C:₩>docker exec -it myweb /bin/bash
root@e95fe3d8aff7:/usr/local/tomcat# ls /root
test.txt
```

✓ 컨테이너 삭제

docker rm [옵션] <컨테이너이름 or ID> [컨테이너 이름 or ID ...]

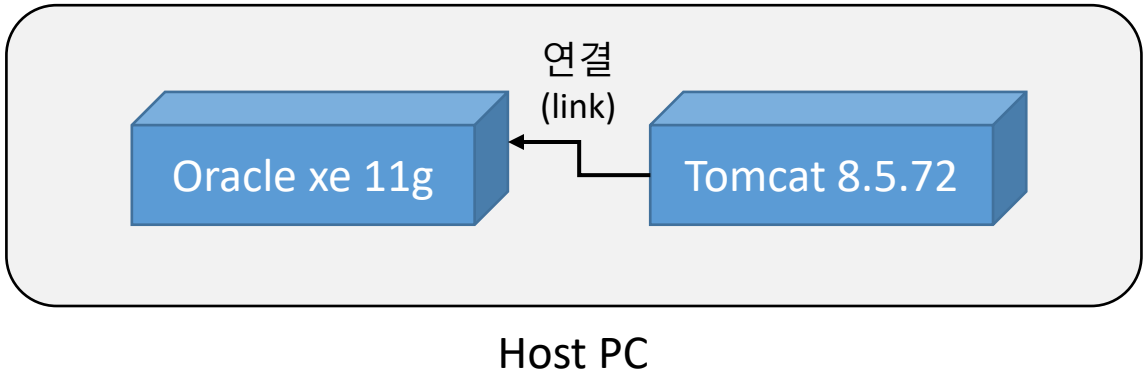
> docker rm –f myweb

```
C:₩>docker rm -f myweb
myweb
```

도커 컨테이너 관리하기

✓ 실습 구성

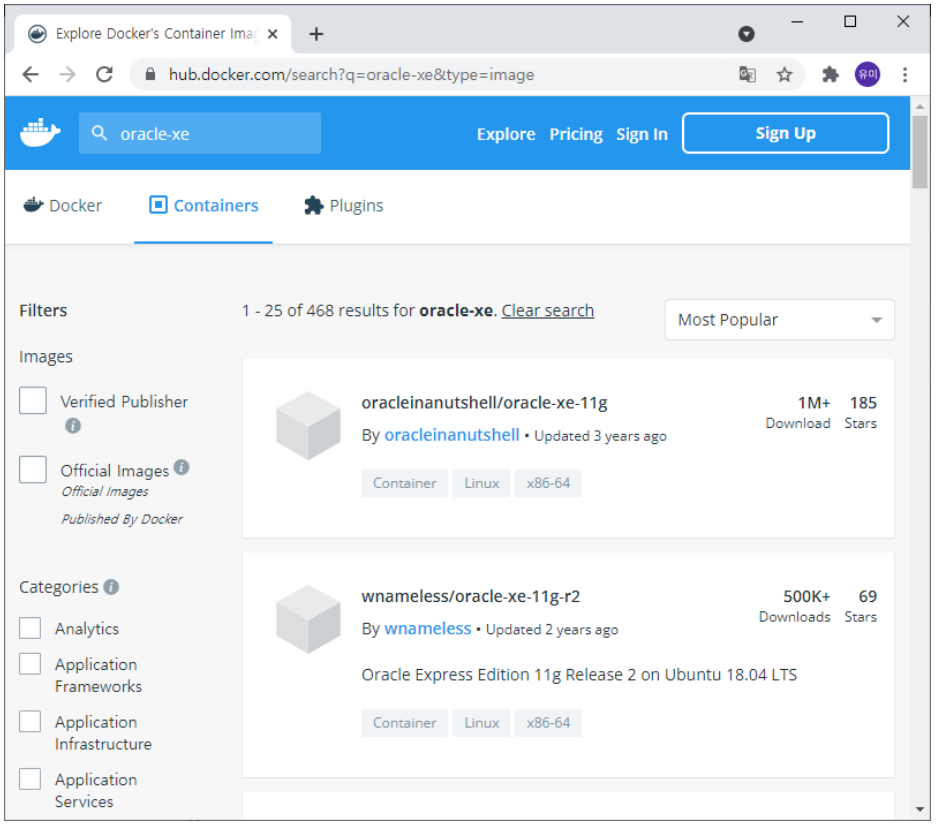
1. Oracle DB 컨테이너 설치
2. DB 사용자 계정 생성 및 테이블 생성
3. Tomcat Server 컨테이너 설치
4. War 파일 Tomcat에 배포



도커 컨테이너 관리하기

✓ Oracle 이미지 검색

목록에서 이미지 클릭 후 상세 설명 확인



Quick Start

Run with 1521 port opened:

```
docker run -d -p 49161:1521 oracleinanutshell/oracle-xe-11g
```

By default, the password verification is disable(password never expired)
Connect database with following setting:

```
hostname: localhost
port: 49161
sid: xe
username: system
password: oracle
```

Password for SYS & SYSTEM

```
oracle
```

도커 컨테이너 관리하기

✓ Oracle 컨테이너 설치

> **docker run -d --name mydb -p 40000:1521**
-e TZ=Asia/Seoul oracleinanutshell/oracle-xe-11g

```
C:₩>docker run -d --name mydb -p 40000:1521 -e TZ=Asia/Seoul oracleinanutshell/oracle-xe-11g
Unable to find image 'oracleinanutshell/oracle-xe-11g:latest' locally
latest: Pulling from oracleinanutshell/oracle-xe-11g
6cf436f81810: Pull complete
987088a85b96: Pull complete
b4624b3efe06: Pull complete
d42beb8ded59: Pull complete
15522cc0fb47: Pull complete
f747bf1d551d: Pull complete
Digest: sha256:8b740e77d4b90add693fedb22938f340821e89665fb58ecaeeb0dace853b9ee5
Status: Downloaded newer image for oracleinanutshell/oracle-xe-11g:latest
f32bb76195ab20912b74589a38d99974f0fe5f2524b66a68e3499e80f2951ef5
```

* TZ : TimeZone 설정(다른 서버와 연동하는 경우 TimeZone 이 맞지 않으면 오류 발생 할 수 있음)

```
C:₩>docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED              STATUS          PORTS
                                   NAMES
f32bb76195ab   oracleinanutshell/oracle-xe-11g   "/bin/sh -c '/usr/sb…"   About a minute ago   Up 58 seconds   22/tcp,
8080/tcp, 0.0.0.0:40000->1521/tcp   mydb
```

도커 컨테이너 관리하기

✓ Oracle 컨테이너로 쿼리파일 복사

일괄처리를 위해 미리 작성한 쿼리파일 컨테이너로 복사하기

> **docker cp d:₩dockerweb₩mydb.sql mydb:/root**

mydb.sql - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

테스트용 쿼리

```
CREATE USER webmaster IDENTIFIED BY webmaster;
GRANT CONNECT, RESOURCE TO webmaster;

CONNECT webmaster/webmaster;

CREATE TABLE member (
    id VARCHAR2(100),
            pw VARCHAR2(100) NOT NULL,
            nick VARCHAR2(100) NOT NULL
);

ALTER TABLE member ADD CONSTRAINT member_pk PRIMARY KEY (id);

DELETE member;
INSERT INTO member ( id, pw, nick ) VALUES ( 'tester', '1234', 'testernick' );
INSERT INTO member ( id, pw, nick ) VALUES ( 'master', 'qwer1234', 'masternick' );
COMMIT;
```

Ln 9, Col 29 | 100% | Windows (CRLF) | UTF-8

```
C:₩>docker cp d:₩dockerweb₩mydb.sql mydb:/root
```

도커 컨테이너 관리하기

✓ Oracle 컨테이너 접속 후 테이블 생성

> **docker exec –it mydb /bin/bash**

미리 복사한 sql 파일 일괄처리

 # ls /root

 # sqlplus sys/oracle as sysdba @/root/mydb.sql

```
C:₩>docker exec -it mydb bash
root@f32bb76195ab:/# ls /root
mydb.sql  oradiag_root
root@f32bb76195ab:/# sqlplus sys/oracle as sysdba @/root/mydb.sql

SQL*Plus: Release 11.2.0.2.0 Production on Sun Oct 10 14:52:21 2021
```

생성된 테이블 확인

```
SQL> SELECT * FROM member;

ID                   PW                   NICK
-------------------- -------------------- --------------------
tester               1234                 testernick
master               qwer1234             masternick
```

도커 컨테이너 관리하기

✓ Tomcat 컨테이너 설치

> **docker run -d --name myweb -p 80:8080**
-e TZ=Asia/Seoul --link mydb:db tomcat:8.5.72-jdk8

```
C:₩>docker run -d --name myweb -p 80:8080 -e TZ=Asia/Seoul --link mydb:db tomcat:8.5.72-jdk8
146f4cecec48aaedbf2e860b5e623a31235ca17bff2dd61b749bd86951e4b8a5
```

--link <컨테이너이름:별칭> : myweb 컨테이너에서 미리 생성한 mydb 컨테이너를 db라는 이름으로 불러서 사용하겠다는 설정

```
C:₩>docker ps -a
CONTAINER ID   IMAGE                              COMMAND                  CREATED          STATUS          PORTS
                                   NAMES
146f4cecec48   tomcat:8.5.72-jdk8                 "catalina.sh run"        59 seconds ago   Up 58 seconds   0.0.0.0:80->8
080/tcp                            myweb
f32bb76195ab   oracleinanutshell/oracle-xe-11g    "/bin/sh -c '/usr/sb…"   3 hours ago      Up 3 hours      22/tcp, 8080
/tcp, 0.0.0.0:40000->1521/tcp   mydb
```

도커 컨테이너 관리하기

✓ Tomcat 컨테이너로 war 파일 복사

> docker cp d:₩ROOT.war myweb:/usr/local/tomcat/webapps

> docker restart myweb

```
C:₩>docker cp d:₩DockerWeb₩ROOT.war myweb:/usr/local/tomcat/webapps

C:₩>docker restart myweb
myweb
```

✓ Host PC IP주소로 웹 접속(http://IP주소/main)

CHAPTER 5.
DockerFile 다루기

DockerFile 다루기

✓ Dockerfile

도커 이미지를 생성할 수 있는 설정 파일

사용자가 이미지를 조합하기 위해 명령줄에서 호출할 수 있는 모든 명령을 포함하는 텍스트 문서

완성된 이미지를 생성하기 위해 필요한 컨테이너, 패키지, 소스코드, 명령어 등을 하나의 파일에 기록

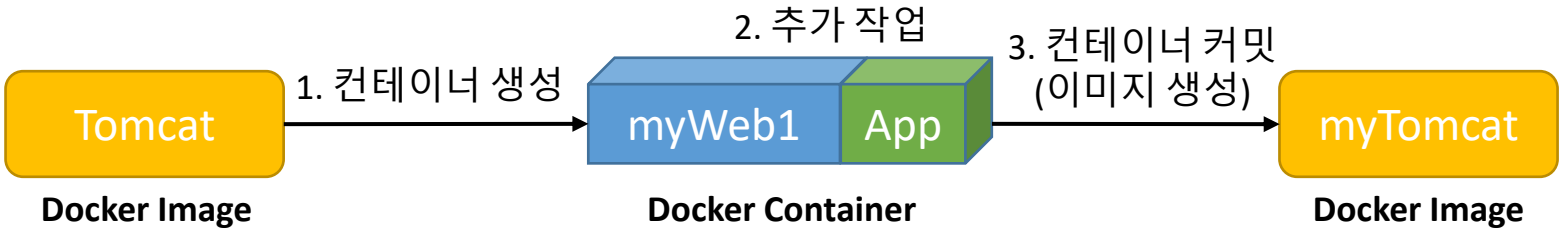
도커에서 이 파일을 읽어 자동으로 작업을 수행한 뒤 완성된 이미지로 만들어 줌

깃과 같은 개발도구를 통해 애플리케이션의 빌드 및 배포를 자동화할 수 있음

DockerFile 다루기

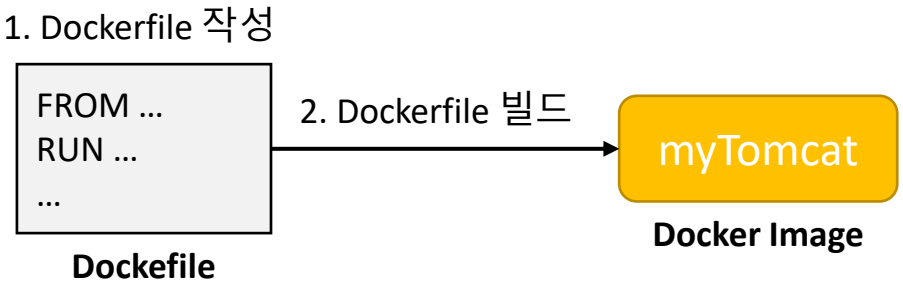
✓ 도커 이미지 생성 방법 1

기존 이미지로 컨테이너 생성 후 작업 완료된 컨테이너를 다시 이미지로 생성



✓도커 이미지 생성 방법 2

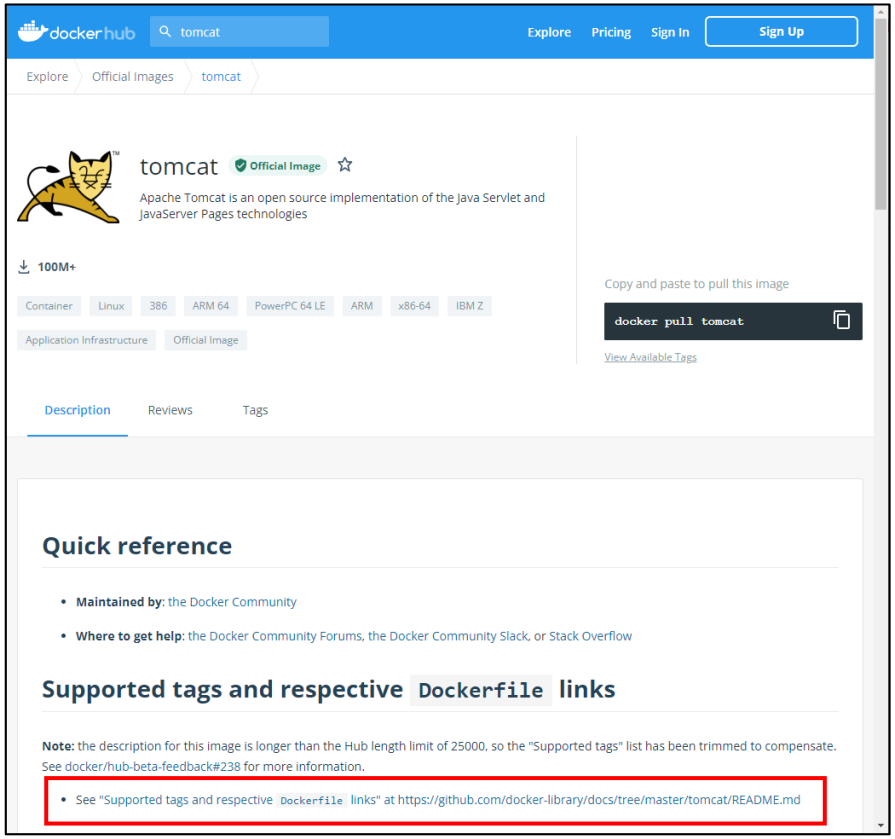
Dockerfile로 필요한 패키지, 소스코드, 명령어 등을 작성 후 빌드

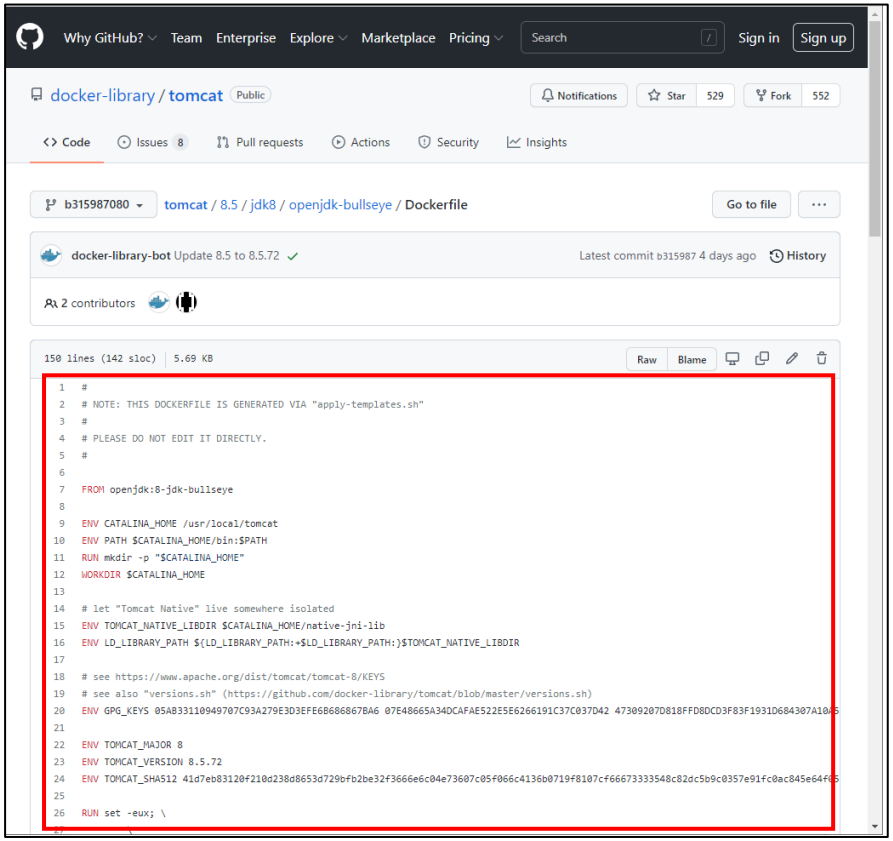


DockerFile 다루기

✓ Tomcat 공식 이미지 Dockerfile 확인

Docker hub tomcat 이미지 검색 시 태그별 Dockerfile을 볼수있는 경로를 제공해 줌





DockerFile 다루기

✓ Dockerfile 작성

컨테이너에서 수행해야 할 작업을 명시
정해진 형식과 명령어로 작성해야 함
문서 파일로 작성
한 줄에 하나의 명령어로 구성
명령어를 명시하고 뒤에 옵션을 추가
명령어는 대/소문자 상관없음(일반적으로 대문자로 표기)
위에서 아래로 한 줄씩 차례대로 실행됨
주석은 # 기호를 사용

DockerFile 다루기

✓ Dockerfile 명령어

명령어	설명
FROM <이미지명>	생성할 이미지의 베이스가 될 이미지 Dockerfile 작성 시 반드시 한번 이상 입력해야 함 이미지 이름은 기본 도커 명령어에서 사용되는 형식과 동일 (저장소이름/이미지이름:태그)
LABEL < " 키:값 " >	이미지에 메타 데이터 추가 "키:값" 형태로 입력 여러 개의 메타데이터 입력 가능 docker inspect 명령어로 확인 가능
ADD <원본경로> <대상경로> ADD ["추가할 파일명", … , "추가될 위치"]	로컬 디렉터리나 외부 URL 및 tar 파일에서 이미지로 파일 복사 기본 Dockerfile이 위치한 디렉터리에서 파일을 가져옴 JSON배열의 형태로 사용 가능 추가할 파일명을 콤마(,)로 구분해서 나열한 뒤 가장 마지막에 추가될 디렉터리 경로를 작성
COPY <원본경로> <대상경로>	로컬 디렉터리에서 이미지로 파일을 복사

DockerFile 다루기

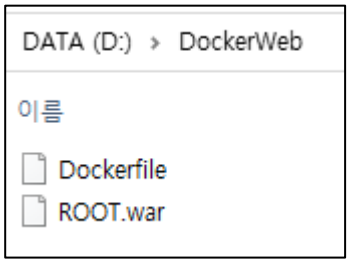
✓ Dockerfile 명령어

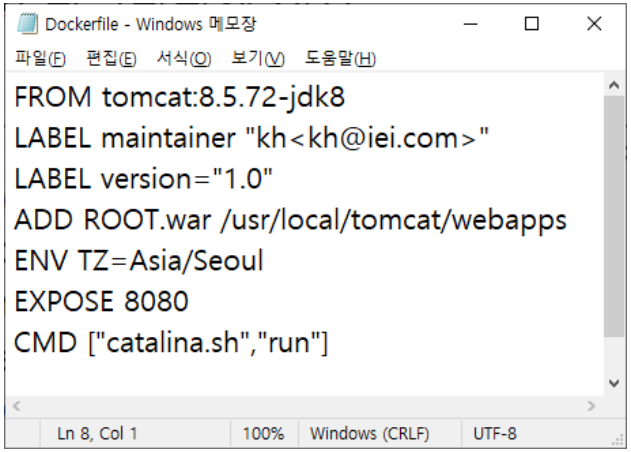
명령어	설명
RUN <명령어> RUN ["실행가능한 파일" , "명령줄 인자 1", "명령줄 인자2" , …]	이미지를 만들기 위해 컨테이너 내부에서 실행할 명령어 입력 명령어 실행 시 대화형 입력이 뜨지 않도록 옵션을 지정해야 함 JSON배열 형태로 사용 가능
WORKDIR <디렉터리>	명령어를 실행할 디렉터리로 이동 cd 명령어를 입력하는 것과 같은 기능
EXPOSE <포트번호>	Dockerfile의 빌드로 생성된 이미지에서 사용될 포트 설정
CMD <명령어> CMD ["실행가능한 파일", "명령줄 인자1", "명령줄 인자 2", …]	컨테이너가 시작될 때 마다 실행할 명령어 설정 Dockerfile 에서 한번만 사용 가능 JSON배열 형태로 사용 가능

DockerFile 다루기

✓ Dockerfile 작성 예제

① D:₩DockerWeb 디렉터리 생성
② ROOT.war 파일 DockerWeb 디렉터리에 저장
③ Dockerfile 작성 후 저장





[Dockerfile]

```
# 베이스 이미지를 tomcat:8.5.72-jdk8로 지정
FROM tomcat:8.5.72-jdk8
# 작성자 라벨 생성
LABEL maintainer "kh<kh@iei.com>"
# 버전 라벨 생성
LABEL version= " 1.0"
# /usr/local/tomcat/webapps 디렉터리에 ROOT.war 파일 추가
ADD ROOT.war /usr/local/tomcat/webapps
# Timezone 환경변수를 Asia/Seoul로 변경
ENV TZ=Asia/Seoul
# 이미지에서 8080포트 사용
EXPOSE 8080
# 컨테이너 실행 시 catalina.sh 파일 자동 실행
CMD ["catalina.sh","run"]
```

DockerFile 다루기

✓ Dockerfile 빌드

작성한 Dockerfile을 이용해 이미지를 생성하는 과정
docker build [옵션] <Dockerfile이 존재하는 디렉터리 경로>

[옵션]

옵션	설명
-f <파일명>	Dockerfile 의 파일명 별도 지정 기본값은 지정된 디렉터리 경로 내 “Dockerfile” 이름으로 사용됨
-t <이미지명:태그> [이미지명:태그 ...]	생성될 이미지 이름 지정 지정하지 않으면 16진수 형태 이름으로 자동 생성됨

DockerFile 다루기

✓ Dockerfile 빌드

> **docker build –t mytomcat:1.0 d:₩DockerWeb**

```
C:₩>docker build -t mytomcat:1.0 d:₩DockerWeb
[+] Building 0.2s (7/7) FINISHED
 => [internal] load build definition from Dockerfile                                                    0.0s
 => => transferring dockerfile: 219B                                                                    0.0s
 => [internal] load .dockerignore                                                                       0.0s
 => => transferring context: 2B                                                                         0.0s
 => [internal] load metadata for docker.io/library/tomcat:8.5.72-jdk8                                   0.0s
 => [1/2] FROM docker.io/library/tomcat:8.5.72-jdk8                                                     0.0s
 => [internal] load build context                                                                       0.0s
 => => transferring context: 32B                                                                        0.0s
 => CACHED [2/2] ADD ROOT.war /usr/local/tomcat/webapps                                                 0.0s
 => exporting to image                                                                                  0.1s
 => => exporting layers                                                                                 0.0s
 => => writing image sha256:041310afa86f51c774f0e3114a2ab5ad31a16fae89ccbf861f351bec32593099            0.0s
 => => naming to docker.io/library/mytomcat:1.0                                                         0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

```
C:₩>docker images
REPOSITORY                      TAG          IMAGE ID       CREATED          SIZE
mytomcat                        1.0          041310afa86f   15 minutes ago   560MB
tomcat                          8.5.72-jdk8  a3defb2c00e5   2 days ago       545MB
oracleinanutshell/oracle-xe-11g latest       ad13c30ec346   2 years ago      2.13GB
```

DockerFile 다루기

✓ 생성된 이미지 상세 정보 보기

> **docker inspect mytomcat:1.0**

```
C:₩>docker image inspect mytomcat:1.0
[
    {
        "Id": "sha256:041310afa86f51c774f0e3114a2ab5ad31a16fae89ccbf861f351bec32593099",
        "RepoTags": [
            "mytomcat:1.0"
        ],
        "RepoDigests": [],
        "Parent": "",
        "Comment": "buildkit.dockerfile.v0",
        "Created": "2021-10-10T18:45:56.2919081Z",
        "Container": "",
        "ContainerConfig": {
            "Hostname": "",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": null,
            "Cmd": null,
```

```
"Labels": {
    "maintainer": "kh<kh@iei.com>",
    "version": "1.0"
}
```

```
    "TZ=Asia/Seoul"
],
```

DockerFile 다루기

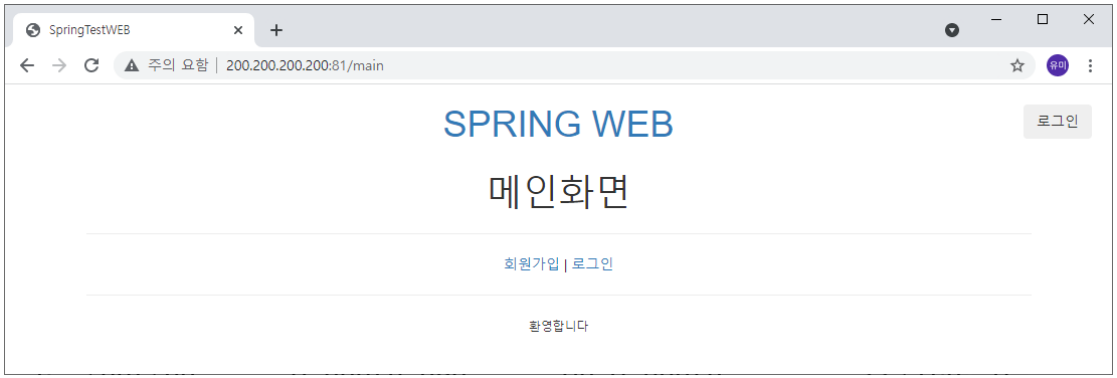
✓ 생성된 이미지로 컨테이너 실행

> **docker run -d --name myweb2 -p 81:8080 --link mydb:db mytomcat:1.0**

```
C:₩>docker run -d --name myweb2 -p 81:8080 --link mydb:db mytomcat:1.0
bf7d1b2f977b6fae226b0e63cc4ba3ca85cb8886d8bfeea538509b60bb8ca27c
```

```
C:₩>docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED          STATUS             PORTS
                                   NAMES
bf7d1b2f977b   mytomcat:1.0                         "catalina.sh run"        16 seconds ago   Up 15 seconds      0.0.0.0:81
->8080/tcp                         myweb2
146f4cecec48   tomcat:8.5.72-jdk8                   "catalina.sh run"        2 hours ago      Up About an hour   0.0.0.0:80
->8080/tcp                         myweb
f32bb76195ab   oracleinanutshell/oracle-xe-11g      "/bin/sh -c '/usr/sb…"   5 hours ago      Up 5 hours         22/tcp, 8
080/tcp, 0.0.0.0:40000->1521/tcp   mydb
```

http://호스트IP:81/main



PART 2.
쿠버네티스 사용하기

CHAPTER 6.
쿠버네티스의 구조와 특징

쿠버네티스의 구조와 특징

✓ 쿠버네티스(Kubernetes)

표준으로 사용되고 있는 컨테이너 오케스트레이션 도구

구글에서 2014년 오픈소스로 공개

여러 대의 도커 호스트를 하나의 클러스터로 만들어 줌

세부적인 기능을 더욱 폭넓게 제공하여 실제 서비스 운영 단계에서 가장 많이 쓰이고 있음

여러 서버의 자원을 클러스터링해 컨테이너를 배치하는 것이 핵심 기능

다른 오케스트레이션 툴 보다 다양한 지식을 필요로 함

✓ 공식 홈페이지

https://kubernetes.io/

CHAPTER 7.
쿠버네티스 설치

쿠버네티스 설치

✓ 쿠버네티스 설치

사용환경과 목적에 따라 설치 방법이 다양하고 복잡함
어떤 용도로 어떤 환경에서 사용할 것인지를 먼저 선택한 뒤 설치

✓ 쿠버네티스 용도에 따른 분류

용도	설치 프로그램
개발 용도	Minikube Docker Desktop 내장 쿠버네티스
서비스 테스트 및 운영 용도	Kops Kubesparay Kubeadm EKS, GKE 등

쿠버네티스 설치

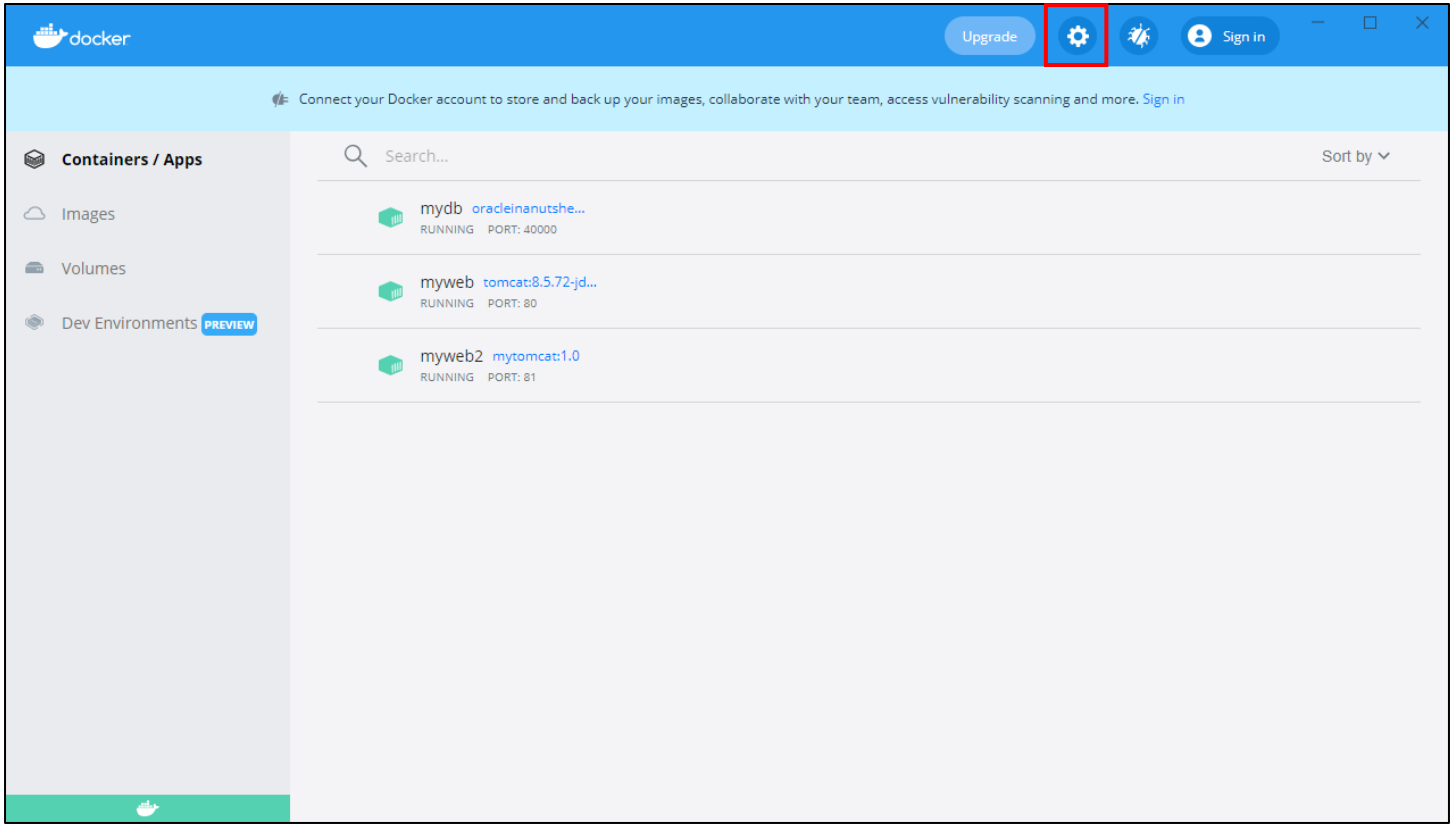
✓ 쿠버네티스 운영 환경

환경		설치 프로그램
자체 서버 환경	on-premise 환경 자체적으로 구축한 서버에 쿠버네티스를 설치하여 사용하는 환경 모든 인프라를 직접 관리	Kubesparay Kubeadm
클라우드 서버 인프라 환경	클라우드 서버에 쿠버네티스를 설치하여 사용하는 환경 서버,네트워크 인프라는 클라우드 제공자가 담당 쿠버네티스 설치 및 관리는 직접 담당	Kops Kubesparay Kubeadm
클라우드 서비스 환경	쿠버네티스 자체를 클라우드 서비스로 제공해 주는 환경 서버, 네트워크, 쿠버네티스 설치 및 관리까지 클라우드 제공자가 담당	EKS, GKE 등 Managed

쿠버네티스 설치

✓ Docker Desktop for Windows 쿠버네티스 설치

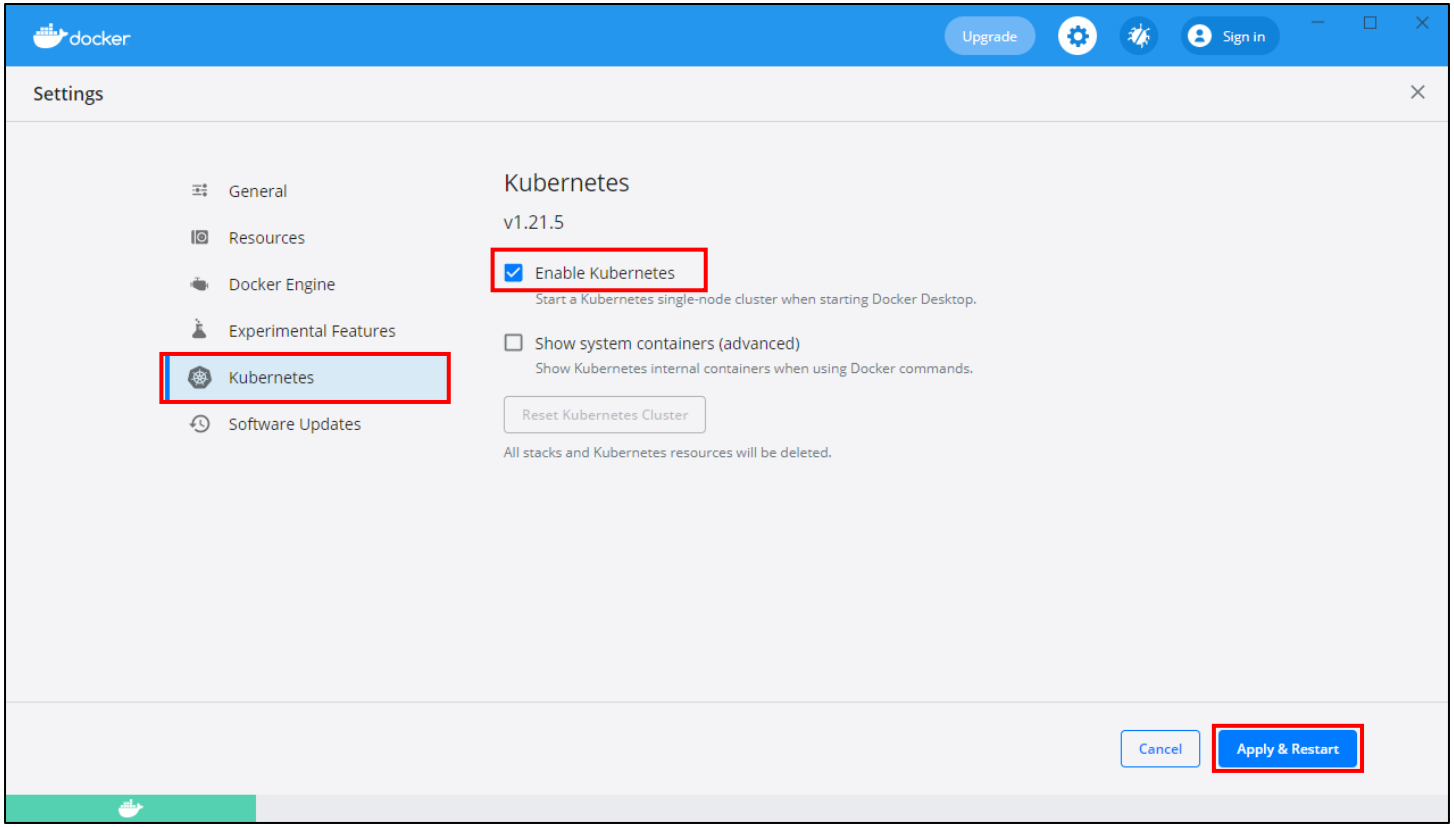
Docker Dashboard 상단 설정 버튼 클릭



쿠버네티스 설치

✓ Docker Desktop for Windows 쿠버네티스 설치

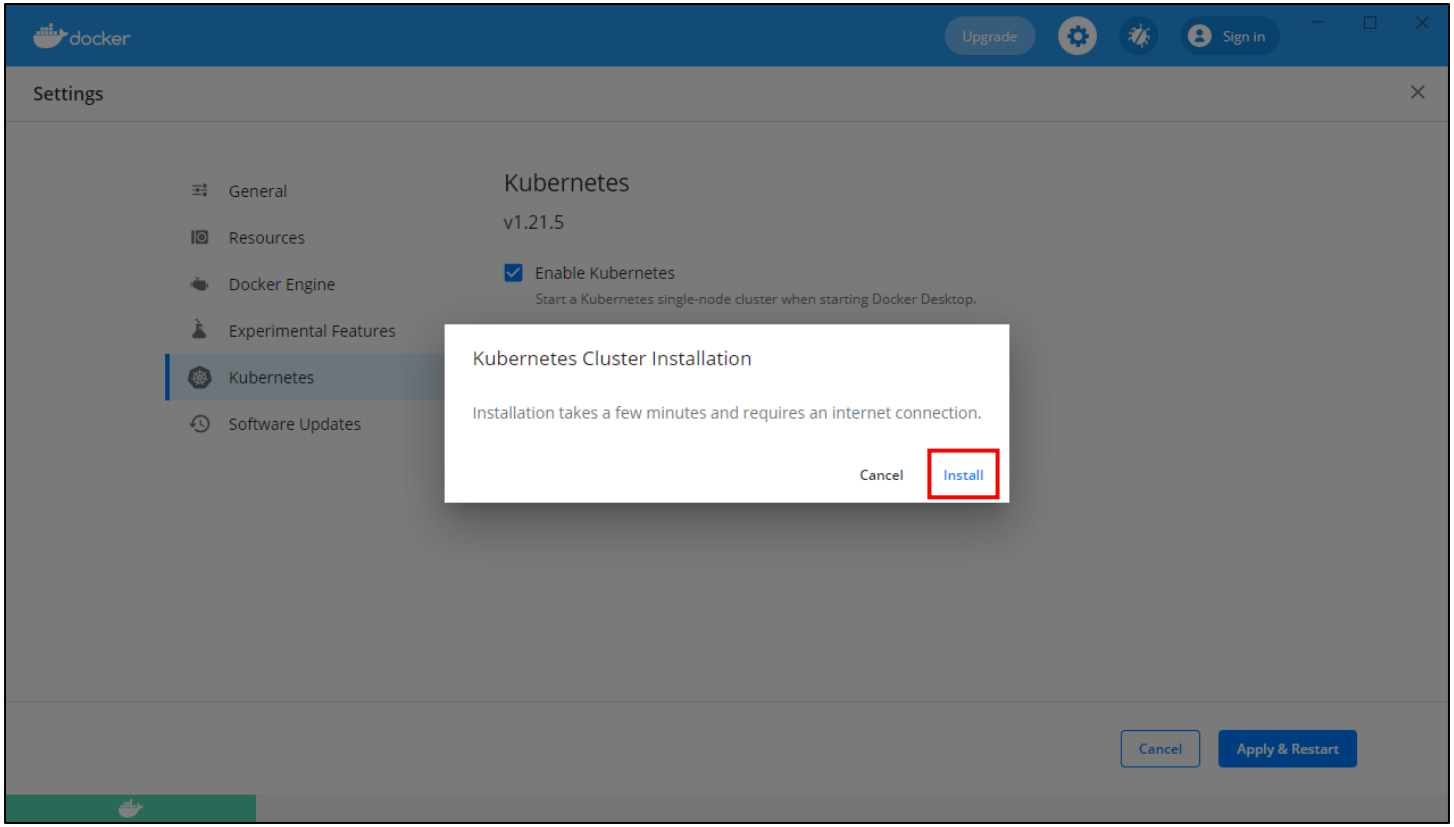
Kubernetes 메뉴 -> Enable Kubernetes 체크 -> Apply&Restart



쿠버네티스 설치

✓ Docker Desktop for Windows 쿠버네티스 설치

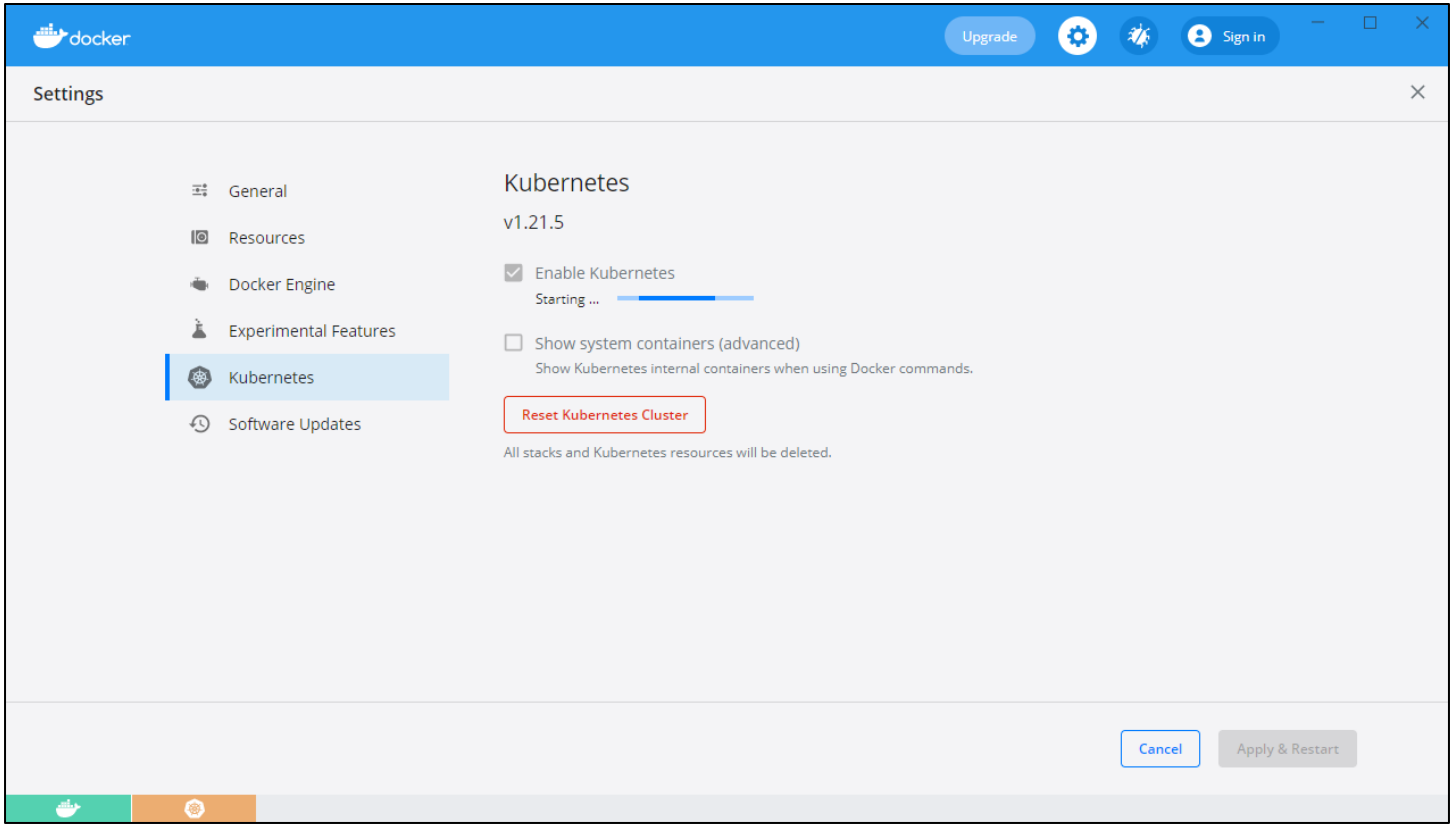
팝업 창에서 install 버튼 클릭



쿠버네티스 설치

✓ Docker Desktop for Windows 쿠버네티스 설치

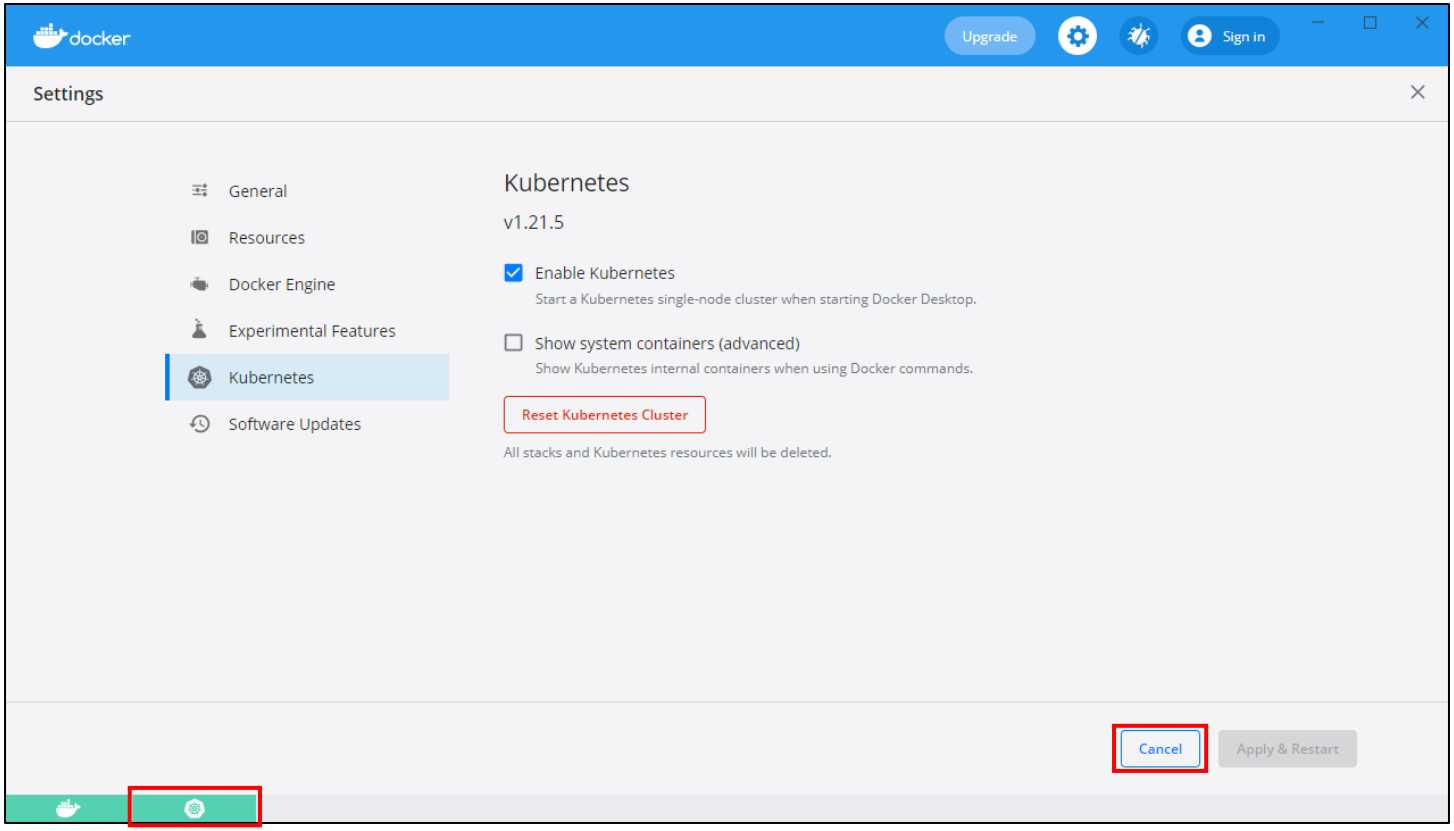
설치 진행 중



쿠버네티스 설치

✓ Docker Desktop for Windows 쿠버네티스 설치

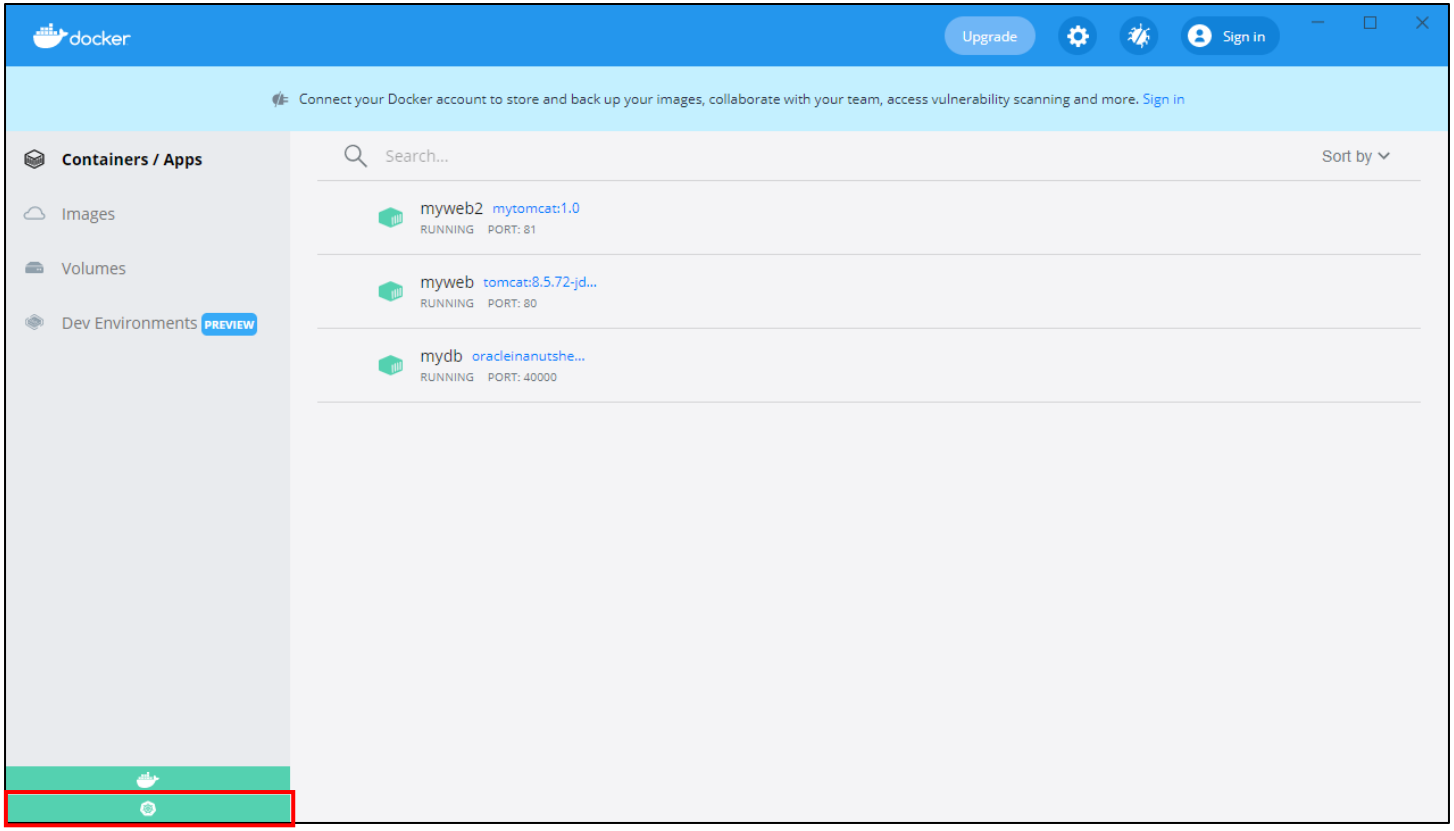
설치 완료되면 하단 쿠버네티스 아이콘 녹색으로 변경됨 -> Cancel



쿠버네티스 설치

✓ Docker Desktop for Windows 쿠버네티스 설치

Dashboard 메인 에서도 쿠버네티스 실행여부 확인



쿠버네티스 설치

✓ 쿠버네티스 버전 확인

> **kubectl version**

```
C:₩>kubectl version
Client Version: version.Info{Major:"1", Minor:"21", GitVersion:"v1.21.5", GitCommit:"aea7bbadd2fc0cd689de94a54e5b7b75886
9d691", GitTreeState:"clean", BuildDate:"2021-09-15T21:10:45Z", GoVersion:"go1.16.8", Compiler:"gc", Platform:"windows/a
md64"}
Server Version: version.Info{Major:"1", Minor:"21", GitVersion:"v1.21.5", GitCommit:"aea7bbadd2fc0cd689de94a54e5b7b75886
9d691", GitTreeState:"clean", BuildDate:"2021-09-15T21:04:16Z", GoVersion:"go1.16.8", Compiler:"gc", Platform:"linux/amd
64"}
```

짧게 버전만 확인 가능

> **kubectl version --short**

```
C:₩>kubectl version --short
Client Version: v1.21.5
Server Version: v1.21.5
```

쿠버네티스 설치

✓ 쿠버네티스 오브젝트 확인

쿠버네티스는 대부분의 리소스를 오브젝트로 관리함
오브젝트 목록 확인 : **kubectl api-resources**

```
C:₩>kubectl api-resources
NAME                              SHORTNAMES   APIVERSION                             NAMESPACED   KIND
bindings                                       v1                                     true         Binding
componentstatuses                 cs           v1                                     false        ComponentStatus
configmaps                        cm           v1                                     true         ConfigMap
endpoints                         ep           v1                                     true         Endpoints
events                            ev           v1                                     true         Event
limitranges                       limits       v1                                     true         LimitRange
namespaces                        ns           v1                                     false        Namespace
nodes                             no           v1                                     false        Node
persistentvolumeclaims            pvc          v1                                     true         PersistentVolumeClaim

persistentvolumes                 pv           v1                                     false        PersistentVolume
pods                              po           v1                                     true         Pod
podtemplates                                   v1                                     true         PodTemplate
replicationcontrollers            rc           v1                                     true         ReplicationController

resourcequotas                    quota        v1                                     true         ResourceQuota
secrets                                        v1                                     true         Secret
serviceaccounts                   sa           v1                                     true         ServiceAccount
services                          svc          v1                                     true         Service
mutatingwebhookconfigurations                  admissionregistration.k8s.io/v1        false        MutatingWebhookConfig
uration
validatingwebhookconfigurations                admissionregistration.k8s.io/v1        false        ValidatingWebhookConf
iguration
customresourcedefinitions         crd,crds     apiextensions.k8s.io/v1                false        CustomResourceDefinit
ion
apiservices                                    apiregistration.k8s.io/v1              false        APIService
```

쿠버네티스 설치

✓ 쿠버네티스 오브젝트 확인

오브젝트 상세 정보 확인 : kubectl explain <오브젝트 이름>

```
C:₩>kubectl explain bindings
KIND:     Binding
VERSION:  v1

DESCRIPTION:
     Binding ties one object to another; for example, a pod is bound to a node
     by a scheduler. Deprecated in 1.7, please use the bindings subresource of
     pods instead.

FIELDS:
   apiVersion   <string>
     APIVersion defines the versioned schema of this representation of an
     object. Servers should convert recognized schemas to the latest internal
     value, and may reject unrecognized values. More info:
     https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources

   kind <string>
     Kind is a string value representing the REST resource this object
     represents. Servers may infer this from the endpoint the client submits
     requests to. Cannot be updated. In CamelCase. More info:
     https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds

   metadata     <Object>
     Standard object's metadata. More info:
     https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata

   target       <Object> -required-
     The target object that you want to bind to the standard object.
```

CHAPTER 8.
Pod 사용하기

Pod 사용하기

✓ Pod

Kubernetes에서 컨테이너를 만들고 관리할 수 있는 가장 작은 배포 가능한 단위

컨테이너를 실행하기 위한 환경

1개의 포드는 1개 이상의 컨테이너가 존재할 수 있음

일반적으로 1개의 컨테이너를 생성해서 사용함

Pod 사용하기

✓ 테스트용 이미지 만들기

① 간단한 웹 페이지 생성

- d:₩KubeTestWeb₩index.jsp 파일 생성
- 서버의 IP 주소를 보여주도록 코드 작성

```
<%@ page language="java" import="java.net.InetAddress" %>
<%  InetAddress inet = InetAddress.getLocalHost(); %>
<h1>TEST WEB v1</h1>
Server IP Address : <%=inet.getHostAddress()%>
```

index.jsp - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
<%@ page language="java" import="java.net.InetAddress" %>
<%  InetAddress inet = InetAddress.getLocalHost(); %>
<h1>TEST WEB v1</h1>
Server IP Address : <%=inet.getHostAddress()%>
```

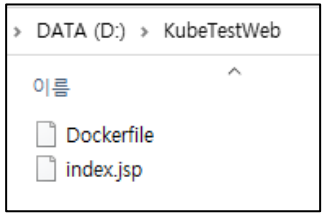
Pod 사용하기

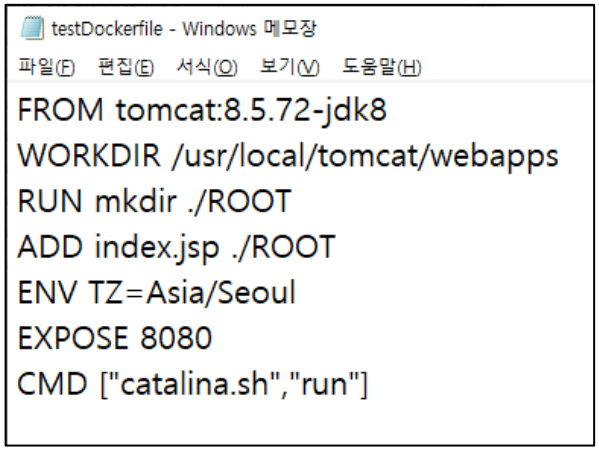
✓ 테스트용 이미지 만들기

② Dockerfile 생성

- d:₩KubeTestWeb₩Dockerfile 파일 생성
- ① 에서 생성한 index.jsp 파일을 추가한 이미지 생성

```
FROM tomcat:8.5.72-jdk8
# 작업 디렉터리 변경
WORKDIR /usr/local/tomcat/webapps
# 작업 디렉터리 내 ROOT 디렉터리 생성
RUN mkdir ./ROOT
# ROOT 디렉터리에 index.jsp 파일 추가
ADD index.jsp ./ROOT
ENV TZ=Asia/Seoul
EXPOSE 8080
CMD ["catalina.sh","run"]
```





Pod 사용하기

✓ 테스트용 이미지 만들기

③ Dockerfile 빌드

> **docker build –t testweb:1.0 d:₩KubeTestWeb**

```
C:₩>docker build -t testweb:1.0 d:₩KubeTestWeb
[+] Building 0.3s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 200B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/tomcat:8.5.72-jdk8
 => [1/4] FROM docker.io/library/tomcat:8.5.72-jdk8
 => [internal] load build context
 => => transferring context: 226B
 => CACHED [2/4] WORKDIR /usr/local/tomcat/webapps
 => CACHED [3/4] RUN mkdir ./ROOT
 => CACHED [4/4] ADD index.jsp ./ROOT
 => exporting to image
 => => exporting layers
 => => writing image sha256:294dc6d36bf30f850fefd995cac132dc7da997c3ea27a384b518f7b0f590178c
 => => naming to docker.io/library/testweb:1.0

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

```
C:₩>docker images
REPOSITORY                          TAG
testweb                             1.0
```

Pod 사용하기

✓ 테스트용 이미지 만들기

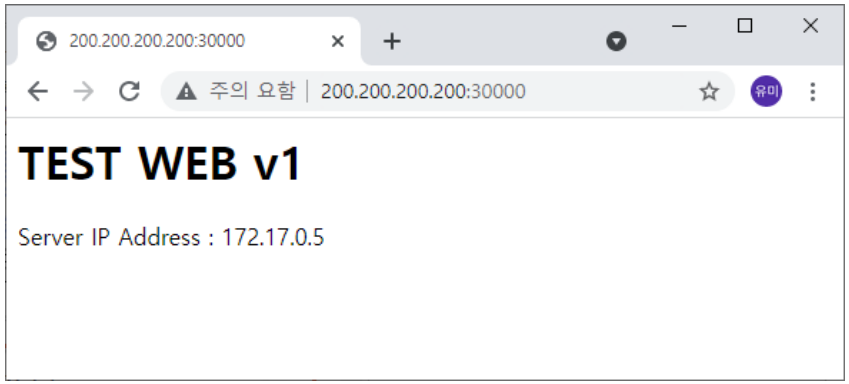
④ 이미지로 컨테이너 생성 후 정상 동작 확인

> **docker run -d --name tweb1 -p 30000:8080 testweb:1.0**

```
C:₩>docker run -d --name tweb1 -p 30000:8080 testweb:1.0
02180a0e5a67ab4bdf4c8519ad1deb8cda406166e3fae56f3d64a136893b8542
```

```
C:₩>docker ps -a
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS
 PORTS                                     NAMES
02180a0e5a67   testweb:1.0                    "catalina.sh run"        20 seconds ago   Up 21 seconds
 0.0.0.0:30000->8080/tcp                   tweb1
```

- http://호스트IP:30000



Pod 사용하기

✓ Pod 생성하기

kubectl run <Pod명> --image=<이미지명> [옵션]

kubectl 명령어로도 생성 가능하지만 일반적으로 Yaml 파일을 이용해서 생성하는 것을 권장함

> **kubectl run kuweb1 --image=testweb:1.0 --port=8080**

```
C:₩>kubectl run kuweb1 --image=testweb:1.0 --port=8080
pod/kuweb1 created
```

* --port : 컨테이너가 노출하는 포트

Pod 사용하기

✓ Pod 목록 확인

kubectl get pods [옵션]

> **kubectl get pods**

```
C:₩>kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
kuweb1    1/1     Running   0          92s
```

> **kubectl get pods -o wide**

- Pod 에서 사용중인 IP 주소 확인 가능

```
C:₩>kubectl get pods -o wide
NAME      READY   STATUS    RESTARTS   AGE     IP          NODE             NOMINATED NODE   READINESS GATES
kuweb1    1/1     Running   0          2m22s   10.1.0.37   docker-desktop   <none>           <none>
```

※ 각 Pod는 **고유한 IP주소**를 가짐

※ 쿠버네티스가 재 기동되면 Pod가 새롭게 생성되면서 **주소가 변경될 수 있음**

Pod 사용하기

✓ Pod 상세 정보 확인

kubectl describe pods [옵션]

> **kubectl describe pods**

```
C:₩>kubectl describe pods
Name:         kuweb1
Namespace:    default
Priority:     0
Node:         docker-desktop/192.168.65.4
Start Time:   Tue, 12 Oct 2021 03:01:15 +0900
Labels:       run=kuweb1
Annotations:  <none>
Status:       Running
IP:           10.1.0.37
IPs:
  IP:  10.1.0.37
Containers:
  kuweb1:
    Container ID:   docker://6d14d19dd95021219ab634ab31b84eab408227280edee41e9319da7ddfa210ab
    Image:          testweb:1.0
    Image ID:       docker://sha256:294dc6d36bf30f850fefd995cac132dc7da997c3ea27a384b518f7b0f590178c
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Tue, 12 Oct 2021 03:01:17 +0900
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-jcqtj (ro)
```

Pod 사용하기

✓ Pod 컨테이너 명령 실행

kubectl exec <Pod명> [-c 컨테이너명] [옵션] -- 명령어 [args ...]

- 외부에서 명령어 실행

  > **kubectl exec kuweb1 -- pwd**

  * pwd : 현재 작업 경로를 출력해주는 명령어

```
C:₩>kubectl exec kuweb1 -- pwd
/usr/local/tomcat/webapps
```

- 컨테이너 내부에 접근해서 명령어 실행

  > **kubectl exec kuweb1 -it -- bash**

  * /bin/bash 쉘로 접근해서 명령어 실행 (종료 시 exit)

```
C:₩>kubectl exec kuweb1 -it -- bash
root@kuweb1:/usr/local/tomcat/webapps# pwd
/usr/local/tomcat/webapps
root@kuweb1:/usr/local/tomcat/webapps# exit
exit
```

Pod 사용하기

✓ 접속 확인

쿠버네티스는 외부에서 접근하려면 서비스 오브젝트를 따로 생성해야 함

내부에서 접속 테스트를 하기 위해 임시로 서버 설치 후 Curl 도구 활용해서 테스트 진행

- CentOS 리눅스 서버 설치

  > **kubectl run -it kuserver --image=centos:7**

- 리눅스 내 Curl 도구 설치

  # yum –y install curl

```
C:₩>kubectl run -it kuserver --image=centos:7
If you don't see a command prompt, try pressing enter.
[root@kuserver /]# yum -y install curl
```

- 접속 확인

  # curl <Pod IP주소>:8080

```
[root@kuserver /]# curl 10.1.0.37:8080


<h1>TEST WEB v1</h1>
Server IP Address : 10.1.0.37
```

Pod 사용하기

✓ Yaml 파일

쿠버네티스에서 오브젝트를 생성하고 관리할 때 명령어보다는 Yaml 파일 사용을 권장함

공백 두 개로 들여쓰기 표현

들여쓰기 기준으로 계층적 구조를 가짐

기호를 주석으로 사용

대소문자 구분해야 함

✓ Yaml 파일 기본 구조

```
apiVersion: 사용할 k8s API의 버전 명세
kind: 생성할 리소스 타입 지정
metadata:
  Name: 생성할 이름
  Labels:
    App: 사용할 라벨
spec:
  생성할 리소스의 구체적인 내용을 정의
```

Pod 사용하기

✓ Pods API 확인

> kubctl api-resources

```
C:₩>kubectl api-resources
NAME                              SHORTNAMES   APIVERSION                             NAMESPACED   KIND
bindings                                       v1                                     true         Binding
componentstatuses                 cs           v1                                     false        ComponentStatus
configmaps                        cm           v1                                     true         ConfigMap
endpoints                         ep           v1                                     true         Endpoints
events                            ev           v1                                     true         Event
limitranges                       limits       v1                                     true         LimitRange
namespaces                        ns           v1                                     false        Namespace
nodes                             no           v1                                     false        Node
persistentvolumeclaims            pvc          v1                                     true         PersistentVolumeClaim
persistentvolumes                 pv           v1                                     false        PersistentVolume
pods                              po           v1                                     true         Pod
podtemplates                                   v1                                     true         PodTemplate
replicationcontrollers            rc           v1                                     true         ReplicationController
resourcequotas                    quota        v1                                     true         ResourceQuota
```

컬럼명	설명
NAME	API 이름
SHORTNAMES	API 짧은 이름(명령어에서 짧은 이름으로 사용 가능)
APIVERSION	API 버전
NAMESPACED	독립적인 서비스 환경을 갖는 영역인지 여부
KIND	쿠버네티스 리소스의 유형

Pod 사용하기

✓ Yaml 파일로 Pod 생성하기

- d:₩KubeTestWeb₩kuweb-pod.yaml 파일 생성

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: kuweb-pod
  label:
    app: kuweb-label
spec:
  containers:
  - name: kuweb2
    image: testweb:1.0
    ports:
    - containerPort: 8080
      protocol: TCP
```

→ 생성할 Pod의 정보

→ Pod안에 생성될 컨테이너의 정보

kuweb-pod.yaml - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: kuweb-pod
  label:
    app: kuweb-label
spec:
  containers:
  - name: kuweb2
    image: testweb:1.0
    ports:
    - containerPort: 8080
      protocol: TCP
```

Pod 사용하기

✓ Yaml 파일로 Pod 생성하기

kubectl apply -f <Yaml 파일 경로> [옵션]

> **kubectl apply -f d:₩KubeTestWeb₩kuweb-pod.yaml**

```
C:₩>kubectl apply -f d:₩KubeTestWeb₩kuweb-pod.yaml
pod/kuweb-pod created
```

```
C:₩>kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
kuserver    1/1     Running   1          34m
kuweb-pod   1/1     Running   0          3m34s
kuweb1      1/1     Running   0          67m
```

✓ Pod 삭제

kubectl delete pods <Pod명 | -l label | --all> [옵션]

> **kubectl delete pods kuweb1 kuweb-pod**

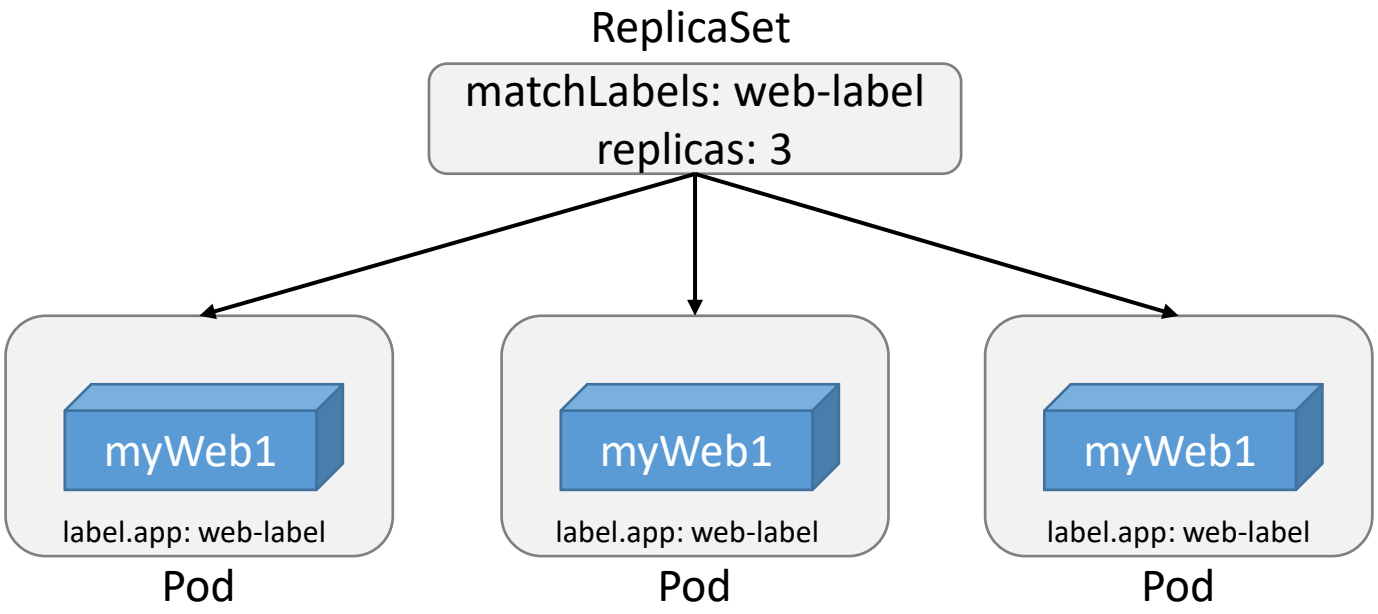
```
C:₩>kubectl delete pods kuweb1 kuweb-pod
pod "kuweb1" deleted
pod "kuweb-pod" deleted
```

CHAPTER 9.
Replica Set 사용하기

Replica Set 사용하기

✓ **ReplicaSet**

주어진 시간에 실행되는 안정적인 복제 Pod 세트를 유지
지정된 수의 동일한 Pod의 가용성을 보장하는 데 사용됨
운영 환경에서 Pod만 단독으로 사용하는 경우는 거의 없음



Replica Set 사용하기

✓ ReplicaSet API 확인

> **kubctl api-resources**

```
C:₩>kubectl api-resources
NAME                              SHORTNAMES   APIVERSION                             NAMESPACED   KIND
bindings                                       v1                                     true         Binding
componentstatuses                 cs           v1                                     false        ComponentStatus
configmaps                        cm           v1                                     true         ConfigMap
endpoints                         ep           v1                                     true         Endpoints
events                            ev           v1                                     true         Event
limitranges                       limits       v1                                     true         LimitRange
namespaces                        ns           v1                                     false        Namespace
nodes                             no           v1                                     false        Node
persistentvolumeclaims            pvc          v1                                     true         PersistentVolumeClaim
persistentvolumes                 pv           v1                                     false        PersistentVolume
pods                              po           v1                                     true         Pod
podtemplates                                   v1                                     true         PodTemplate
replicationcontrollers            rc           v1                                     true         ReplicationController
resourcequotas                    quota        v1                                     true         ResourceQuota
secrets                                        v1                                     true         Secret
serviceaccounts                   sa           v1                                     true         ServiceAccount
services                          svc          v1                                     true         Service
mutatingwebhookconfigurations                  admissionregistration.k8s.io/v1        false        MutatingWebhookConfiguration
validatingwebhookconfigurations                admissionregistration.k8s.io/v1        false        ValidatingWebhookConfiguration
customresourcedefinitions         crd,crds     apiextensions.k8s.io/v1                false        CustomResourceDefinition
apiservices                                    apiregistration.k8s.io/v1              false        APIService
controllerrevisions                            apps/v1                                true         ControllerRevision
daemonsets                        ds           apps/v1                                true         DaemonSet
deployments                       deploy       apps/v1                                true         Deployment
replicasets                       rs           apps/v1                                true         ReplicaSet
statefulsets                      sts          apps/v1                                true         StatefulSet
```

Replica Set 사용하기

✓ ReplicaSet Yaml 파일 작성

- d:₩KubeTestWeb₩kuweb-replicaset.yaml 파일 생성

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: kuweb-replicaset
spec:
  replicas: 3
  selector:
    matchLabels:
      app: kuweb-label
```

생성할 ReplicaSet의 정보

ReplicaSet 정의
replicas – Pod 유지 개수 설정
selector – Pod 선택기(관리할 Pod를 찾는 방법)
matchLabels – 일치하는 label의 Pod를 선택

다음 페이지 내용 이어서 쓰기

Replica Set 사용하기

✓ ReplicaSet Yaml 파일 작성

- d:₩KubeTestWeb₩kuweb-replicaset.yaml 파일 생성

```
template:
  metadata:
    name: kuweb-pod
    labels:
      app: kuweb-label
  spec:
    containers:
    - name: kuweb
      image: testweb:1.0
      ports:
      - containerPort: 8080
        protocol: TCP
```

생성할 Pod의 정보

Pod안에 생성될 컨테이너의 정보

Replica Set 사용하기

✓ ReplicaSet Yaml 파일 작성

- d:₩KubeTestWeb₩kuweb-replicaset.yaml 파일 생성
- 들여쓰기 주의해야 함
- 공백 두 개가 한번 들여쓰기

kuweb-replicaset.yaml - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: kuweb-replicaset
spec:
  replicas: 3
  selector:
    matchLabels:
      app: kuweb-label
  template:
    metadata:
      name: kuweb-pod
      labels:
        app: kuweb-label
    spec:
      containers:
      - name: kuweb
        image: testweb:1.0
        ports:
        - containerPort: 8080
          protocol: TCP
```

Replica Set 사용하기

✓ ReplicaSet 생성

kubectl apply -f <Yaml 파일 경로> [옵션]

> **kubectl apply -f d:₩KubeTestWeb₩kuweb-replicaset.yaml**

```
C:₩>kubectl apply -f d:₩KubeTestWeb₩kuweb-replicaset.yaml
replicaset.apps/kuweb-replicaset created
```

Replica Set 사용하기

✓ ReplicaSet 목록 확인

kubectl get replicasets [옵션]

> **kubectl get replicasets**

- 유지되는 Pod 개수 확인 가능

```
C:₩>kubectl get replicasets
NAME              DESIRED   CURRENT   READY   AGE
kuweb-replicaset  3         3         3       3m19s
```

> **kubectl get rs -o wide**

- replicaset의 Pod 선택기, 컨테이너이름, 이미지이름 확인 가능

* rs : replicasets의 짧은 명령어

```
C:₩>kubectl get rs -o wide
NAME              DESIRED   CURRENT   READY   AGE     CONTAINERS   IMAGES        SELECTOR
kuweb-replicaset  3         3         3       6m11s   kuweb        testweb:1.0   app=kuweb-label
```

Replica Set 사용하기

✓ ReplicaSet 상세 정보 확인

kubectl describe rs [옵션]

> **kubectl describe rs kuweb-replicaset**

```
C:₩>kubectl describe rs kuweb-replicaset
Name:         kuweb-replicaset
Namespace:    default
Selector:     app=kuweb-label
Labels:       <none>
Annotations:  <none>
Replicas:     3 current / 3 desired
Pods Status:  3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=kuweb-label
  Containers:
   kuweb:
    Image:        testweb:1.0
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Events:
  Type    Reason            Age   From                   Message
  ----    ------            ----  ----
  Normal  SuccessfulCreate  11m   replicaset-controller  Created pod: kuweb-replicaset-drmv5
  Normal  SuccessfulCreate  11m   replicaset-controller  Created pod: kuweb-replicaset-tlv9c
  Normal  SuccessfulCreate  11m   replicaset-controller  Created pod: kuweb-replicaset-4jppw
```

Pod 선택기 정보

Pod 유지개수
Pod 상태정보

컨테이너 정보

생성된 Pod 이름

Replica Set 사용하기

✓ ReplicaSet 으로 생성된 Pod 정보 확인

> **kubectl get po**

* po : pods 의 짧은 명령어

```
C:₩>kubectl get po
NAME                      READY   STATUS    RESTARTS   AGE
kuserver                  1/1     Running   1          21h
kuweb-replicaset-4jppw    1/1     Running   0          17m
kuweb-replicaset-drmv5    1/1     Running   0          17m
kuweb-replicaset-tlv9c    1/1     Running   0          17m
```

- Pod 이름 : Replicaset 이름+16진수 랜덤 값으로 자동 생성됨

> **kubectl get po –o wide**

- Pod 에서 사용중인 IP 주소 확인 가능

```
C:₩>kubectl get po -o wide
NAME                      READY   STATUS    RESTARTS   AGE   IP          NODE             NOMINATED NODE   READINESS GATES
kuserver                  1/1     Running   1          21h   10.1.0.42   docker-desktop   <none>           <none>
kuweb-replicaset-4jppw    1/1     Running   0          18m   10.1.0.48   docker-desktop   <none>           <none>
kuweb-replicaset-drmv5    1/1     Running   0          18m   10.1.0.47   docker-desktop   <none>           <none>
kuweb-replicaset-tlv9c    1/1     Running   0          18m   10.1.0.49   docker-desktop   <none>           <none>
```

Replica Set 사용하기

✓ 접속 확인

- 테스트용 리눅스 서버에서 접속 확인

> **kubectl exec -it kuserver -- bash**

```
C:₩>kubectl exec -it kuserver -- bash
[root@kuserver /]#
```

 # curl <Pod IP주소:포트번호>

```
[root@kuserver /]# curl 10.1.0.48:8080


<h1>TEST WEB v1</h1>
Server IP Address : 10.1.0.48

[root@kuserver /]# curl 10.1.0.47:8080


<h1>TEST WEB v1</h1>
Server IP Address : 10.1.0.47

[root@kuserver /]# curl 10.1.0.49:8080


<h1>TEST WEB v1</h1>
Server IP Address : 10.1.0.49
```

동일한 Pod가 여러 개 생성되었기 때문에
모두 동일한 페이지 실행됨

Replica Set 사용하기

✓ Pod 유지 개수 변경

Replicaset 생성 후 Pod의 유지 개수를 변경해야 하는 경우
spec.replicas 속성 값만 변경해 주면 됨
Yaml 파일 수정, kubectl edit/kubectl patch 명령어 사용 등

[기존에 생성한 Yaml 파일]

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: kuweb-replicaset
spec:
  replicas: 5
  selector:
    matchLabels:
      app: kuweb-label

... 생략 ...
```

Replica Set 사용하기

✓ Pod 유지 개수 변경

- 수정된 Yaml 파일 적용

> **kubectl apply -f d:₩KubeTestWeb₩kuweb-replicaset.yaml**

```
C:₩>kubectl apply -f d:₩KubeTestWeb₩kuweb-replicaset.yaml
replicaset.apps/kuweb-replicaset configured
```

* 설정만 변경되었기 때문에 create가 아닌 configured로 표시됨

- 변경된 정보 확인

```
C:₩>kubectl get rs
NAME                DESIRED   CURRENT   READY   AGE
kuweb-replicaset    5         5         5       46m

C:₩>kubectl get po
NAME                      READY   STATUS    RESTARTS   AGE
kuserver                  1/1     Running   1          22h
kuweb-replicaset-4ippw    1/1     Running   0          46m
kuweb-replicaset-8ctlm    1/1     Running   0          2m56s
kuweb-replicaset-8h5zz    1/1     Running   0          2m56s
kuweb-replicaset-drmv5    1/1     Running   0          46m
kuweb-replicaset-tlv9c    1/1     Running   0          46m
```

개수를 맞추기 위해
새롭게 추가된 Pod

Replica Set 사용하기

✓ Pod 오류 시 자동 재 생성

장애가 발생하거나 필요에 의해 특정 Pod를 사용할 수 없는 경우 유지 개수에 맞게 자동으로 재 생성됨

- 특정 Pod 수동 삭제(Pod 목록에서 임의로 1개 선택)

> kubectl delete po kuweb-replicaset-4jppw

```
C:₩>kubectl delete po kuweb-replicaset-4jppw
pod "kuweb-replicaset-4jppw" deleted
```

- Pod 목록 확인 시 새로운 Pod 자동 생성됨

```
C:₩>kubectl get po
NAME                    READY   STATUS    RESTARTS   AGE
kuserver                1/1     Running   1          22h
kuweb-replicaset-8ctlm  1/1     Running   0          8m39s
kuweb-replicaset-8h5zz  1/1     Running   0          8m39s
kuweb-replicaset-drmv5  1/1     Running   0          52m
kuweb-replicaset-ff8hj  1/1     Running   0          7s
kuweb-replicaset-tlv9c  1/1     Running   0          52m
```

Replica Set 사용하기

✓ ReplicaSet 삭제

kubectl delete rs <rs명 | -l label | --all> [옵션]

> kubectl delete rs kuweb-replicaset

```
C:₩>kubectl delete rs kuweb-replicaset
replicaset.apps "kuweb-replicaset" deleted
```

- ReplicaSet에 의해 생성된 Pod도 같이 삭제됨

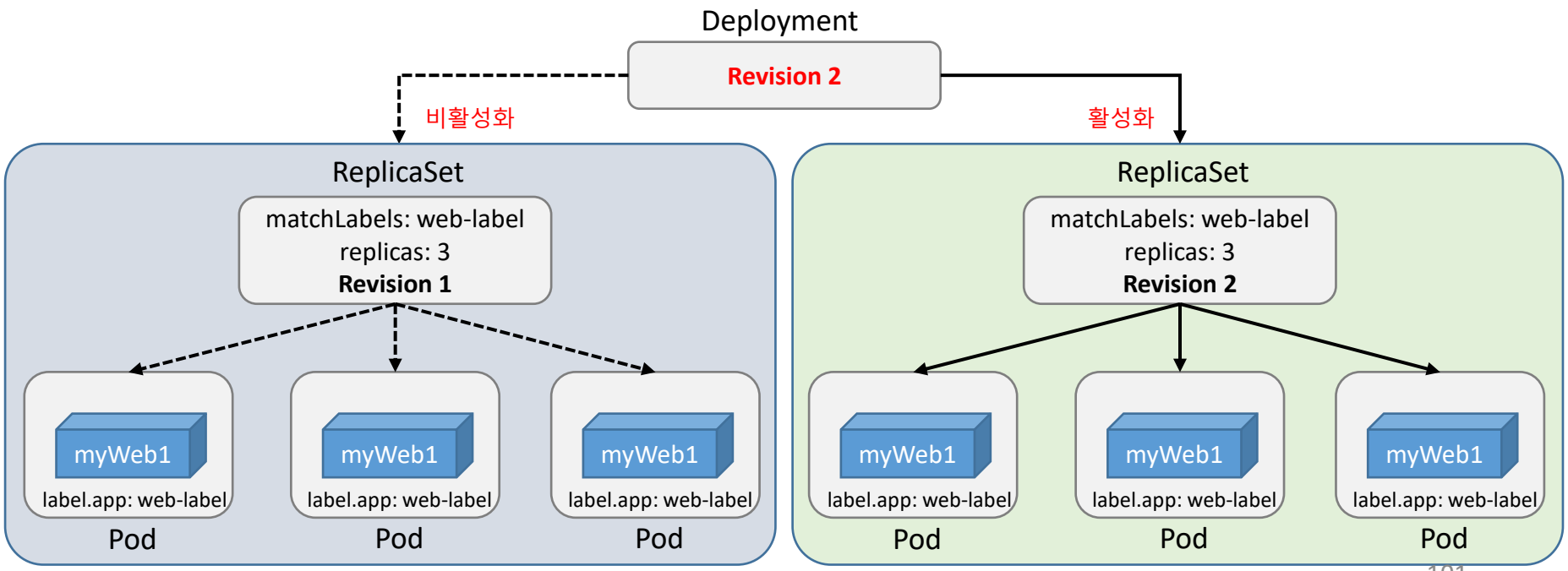
```
C:₩>kubectl get rs
No resources found in default namespace.

C:₩>kubectl get po
NAME        READY   STATUS    RESTARTS   AGE
kuserver    1/1     Running   1          22h
```

CHAPTER 10.
Deployment
사용하기

Deployment 사용하기

✓ Deployment

ReplicaSet과 Pod의 배포 관리
Pod 및 ReplicaSet에 대한 **선언적 업데이트를 제공**
ReplicaSet의 상위 오브젝트
쿠버네티스에서 공식적으로 Deployment를 사용할 것을 권장함



Deployment 사용하기

✓ Deployment API 확인

> kubctl api-resources

```
C:₩>kubectl api-resources
NAME                              SHORTNAMES   APIVERSION                             NAMESPACED   KIND
bindings                                       v1                                     true         Binding
componentstatuses                 cs           v1                                     false        ComponentStatus
configmaps                        cm           v1                                     true         ConfigMap
endpoints                         ep           v1                                     true         Endpoints
events                            ev           v1                                     true         Event
limitranges                       limits       v1                                     true         LimitRange
namespaces                        ns           v1                                     false        Namespace
nodes                             no           v1                                     false        Node
persistentvolumeclaims            pvc          v1                                     true         PersistentVolumeClaim
persistentvolumes                 pv           v1                                     false        PersistentVolume
pods                              po           v1                                     true         Pod
podtemplates                                   v1                                     true         PodTemplate
replicationcontrollers            rc           v1                                     true         ReplicationController
resourcequotas                    quota        v1                                     true         ResourceQuota
secrets                                        v1                                     true         Secret
serviceaccounts                   sa           v1                                     true         ServiceAccount
services                          svc          v1                                     true         Service
mutatingwebhookconfigurations                  admissionregistration.k8s.io/v1        false        MutatingWebhookConfiguration
validatingwebhookconfigurations                admissionregistration.k8s.io/v1        false        ValidatingWebhookConfiguration
customresourcedefinitions         crd,crds     apiextensions.k8s.io/v1                false        CustomResourceDefinition
apiservices                                    apiregistration.k8s.io/v1              false        APIService
controllerrevisions                            apps/v1                                true         ControllerRevision
daemonsets                        ds           apps/v1                                true         DaemonSet
deployments                       deploy       apps/v1                                true         Deployment
replicasets                       rs           apps/v1                                true         ReplicaSet
statefulsets                      sts          apps/v1                                true         StatefulSet
```

Deployment 사용하기

✓ Deployment Yaml 파일 작성

- Kind 외에 ReplicaSet 과 동일한 방식으로 생성
- d:₩KubeTestWeb₩kuweb-deployment.yaml 파일 생성

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kuweb-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: kuweb-label
```

아래 내용 ReplicaSet과 동일

생성할 Deployment 의 정보

ReplicaSet 정의
replicas – Pod 유지 개수 설정
selector – Pod 선택기(관리할 Pod를 찾는 방법)
matchLabels – 일치하는 label의 Pod를 선택

Deployment 사용하기

✓ Deployment Yaml 파일 작성

- d:₩KubeTestWeb₩kuweb-deployment.yaml 파일 생성
- 들여쓰기 주의해야 함
- 공백 두 개가 한번 들여쓰기

kuweb-deployment.yaml - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kuweb-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: kuweb-label
  template:
    metadata:
      name: kuweb-pod
      labels:
        app: kuweb-label
    spec:
      containers:
      - name: kuweb
        image: testweb:1.0
        ports:
        - containerPort: 8080
          protocol: TCP
```

Deployment 사용하기

✓ Deployment 생성

kubectl apply -f <Yaml 파일 경로> [옵션]

> **kubectl apply -f d:₩KubeTestWeb₩kuweb-deployment.yaml**

```
C:₩>kubectl apply -f d:₩KubeTestWeb₩kuweb-deployment.yaml
deployment.apps/kuweb-deployment created
```

Deployment 사용하기

✓ Deployment 목록 확인

kubectl get deployments [옵션]

> **kubectl get deployments**

- 유지되는 Pod 개수 확인 가능

```
C:₩>kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
kuweb-deployment   3/3     3            3           78s
```

> **kubectl get deploy -o wide**

- replicaset의 Pod 선택기, 컨테이너이름, 이미지이름 확인 가능

* deploy : deployments의 짧은 명령어

```
C:₩>kubectl get deploy -o wide
NAME               READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES        SELECTOR
kuweb-deployment   3/3     3            3           91s   kuweb        testweb:1.0   app=kuweb-label
```

Deployment 사용하기

✓ Deployment 상세 정보 확인

kubectl describe deploy [옵션]

> **kubectl describe deploy kuweb-deployment**

```
C:₩>kubectl describe deploy kuweb-deployment
Name:                   kuweb-deployment
Namespace:              default
CreationTimestamp:      Wed, 13 Oct 2021 04:15:41 +0900
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=kuweb-label
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=kuweb-label
  Containers:
   kuweb:
    Image:        testweb:1.0
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   kuweb-deployment-54f4f94f95 (3/3 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  6m58s  deployment-controller  Scaled up replica set kuweb-deployment-54f4f94f95 to 3
```

Pod 배포 상태

이전 ReplicaSet 이름
새로 생성된 ReplicaSet 이름과 Pod 개수

Deployment 사용하기

✓ Deployment로 생성된 ReplicaSet, Pod 정보 확인

- ReplicaSet 이름 : Deployment이름+16진수 랜덤 값

```
C:₩>kubectl get rs -o wide
NAME                          DESIRED   CURRENT   READY   AGE   CONTAINERS   IMAGES        SELECTOR
kuweb-deployment-54f4f94f95   3         3         3       13m   kuweb        testweb:1.0   app=kuweb-label,pod-template-hash=54f4f94f95
```

- Pod 이름 : ReplicaSet이름+16진수 랜덤 값

```
C:₩>kubectl get po -o wide
NAME                                  READY   STATUS    RESTARTS   AGE     IP          NODE             NOMINATED NODE   READINESS GATES
kuserver                              1/1     Running   1          25h     10.1.0.42   docker-desktop   <none>           <none>
kuweb-deployment-54f4f94f95-cqp7f     1/1     Running   0          3m53s   10.1.0.62   docker-desktop   <none>           <none>
kuweb-deployment-54f4f94f95-jf6xs     1/1     Running   0          3m53s   10.1.0.63   docker-desktop   <none>           <none>
kuweb-deployment-54f4f94f95-wwmvg     1/1     Running   0          3m53s   10.1.0.64   docker-desktop   <none>           <none>
```

- 접속 확인
  - 테스트용 리눅스 서버에서 접속 확인
  - > **kubectl exec -it kuserver -- bash**

```
[root@kuserver /]# curl 10.1.0.62:8080


<h1>TEST WEB v1</h1>
Server IP Address : 10.1.0.62
```

```
[root@kuserver /]# curl 10.1.0.63:8080


<h1>TEST WEB v1</h1>
Server IP Address : 10.1.0.63
```

```
[root@kuserver /]# curl 10.1.0.64:8080


<h1>TEST WEB v1</h1>
Server IP Address : 10.1.0.64
```

Deployment 사용하기

✓ Deployment 업데이트

애플리케이션 업데이트 시 중단 없이 업데이트 가능
변경 사항을 Revision에 기록하여 필요한 경우 롤백(rollback) 가능

- 테스트용 업데이트 이미지 생성

  d:₩KubeTestWeb₩index.jsp 파일 수정

```
index.jsp - Windows 메모장
파일(F)  편집(E)  서식(O)  보기(V)  도움말(H)
<%@ page language="java" import="java.net.InetAddress" %>
<%  InetAddress inet = InetAddress.getLocalHost(); %>
<h1>TEST WEB v2</h1>
Server IP Address : <%=inet.getHostAddress()%>
```

- Dockerfile 빌드

  > **docker build –t testweb:2.0 d:₩KubeTestWeb**

```
C:₩>docker images
REPOSITORY                                  TAG
testweb                                     2.0
testweb                                     1.0
```

Deployment 사용하기

✓ Deployment 이미지 업데이트

kubectl set image deploy <Deployment이름>
<컨테이너이름>=<이미지이름> --record=true

* --record : 변경사항을 기록하여 해당 버전의 ReplicaSet을 보존

> **kubectl set image deploy kuweb-deployment kuweb=testweb:2.0 --record=true**

```
C:₩>kubectl set image deploy kuweb-deployment kuweb=testweb:2.0 --record=true
deployment.apps/kuweb-deployment image updated
```

- 변경된 정보 확인

```
C:₩>kubectl get deploy -o wide
NAME                READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES        SELECTOR
kuweb-deployment    3/3     3            3           47m   kuweb        testweb:2.0   app=kuweb-label
```

Deployment 사용하기

✓ Deployment 이미지 업데이트

- ReplicaSet 정보 확인

  기존 ReplicaSet은 보존되고 새로운 ReplicaSet 생성 및 활성화

```
C:₩>kubectl get rs -o wide
NAME                          DESIRED   CURRENT   READY   AGE   CONTAINERS   IMAGES        SELECTOR
kuweb-deployment-54f4f94f95   0         0         0       47m   kuweb        testweb:1.0   app=kuweb-label,pod-template-hash=54f4f94f95
kuweb-deployment-6c886754c4   3         3         3       57s   kuweb        testweb:2.0   app=kuweb-label,pod-template-hash=6c886754c4
```

- Pods 정보 확인

  새롭게 생성되어 Pod이름과 IP주소 모두 변경됨

```
C:₩>kubectl get po -o wide
NAME                                  READY   STATUS    RESTARTS   AGE   IP          NODE             NOMINATED NODE   READINESS GATES
kuserver                              1/1     Running   1          25h   10.1.0.42   docker-desktop   <none>           <none>
kuweb-deployment-6c886754c4-84skl     1/1     Running   0          76s   10.1.0.67   docker-desktop   <none>           <none>
kuweb-deployment-6c886754c4-ktf4v     1/1     Running   0          80s   10.1.0.65   docker-desktop   <none>           <none>
kuweb-deployment-6c886754c4-nqqjn     1/1     Running   0          78s   10.1.0.66   docker-desktop   <none>           <none>
```

- 접속 확인

  > **kubectl exec -it kuserver -- bash**

```
[root@kuserver /]# curl 10.1.0.65:8080


<h1>TEST WEB v2</h1>
Server IP Address : 10.1.0.65
```

```
[root@kuserver /]# curl 10.1.0.66:8080


<h1>TEST WEB v2</h1>
Server IP Address : 10.1.0.66
```

```
[root@kuserver /]# curl 10.1.0.67:8080


<h1>TEST WEB v2</h1>
Server IP Address : 10.1.0.67
```

Deployment 사용하기

✓ Deployment Rollout 기록 확인

변경된 사항은 Revision으로 기록됨

kubectl rollout history deploy [Deployment이름] [옵션]

> **kubectl rollout history deploy kuweb-deployment**

```
C:₩>kubectl rollout history deploy kuweb-deployment
deployment.apps/kuweb-deployment
REVISION  CHANGE-CAUSE
1         <none>
2         kubectl set image deploy kuweb-deployment kuweb=testweb:2.0 --record=true
```

> **kubectl rollout history deploy kuweb-deployment --revision=2**

```
C:₩>kubectl rollout history deploy kuweb-deployment --revision=2
deployment.apps/kuweb-deployment with revision #2
Pod Template:
  Labels:       app=kuweb-label
        pod-template-hash=6c886754c4
  Annotations:  kubernetes.io/change-cause: kubectl set image deploy kuweb-deployment kuweb=testweb:2.0 --record=true
  Containers:
   kuweb:
    Image:      testweb:2.0
    Port:       8080/TCP
    Host Port:  0/TCP
    Environment:        <none>
    Mounts:     <none>
  Volumes:      <none>
```

Revision 별 세부 사항 확인 가능

Deployment 사용하기

✓ Deployment 롤백

배포가 안정적이지 않은 경우 안정적인 이전 버전으로 롤백
kubectl rollout undo deploy <deploy이름> [--to-revision=번호]
--to-revision을 생략하면 바로 이전 버전으로 롤백

> **kubectl rollout undo deploy kuweb-deployment**

```
C:₩>kubectl rollout undo deploy kuweb-deployment
deployment.apps/kuweb-deployment rolled back
```

- 정보 확인 시 이전 버전으로 다시 돌아옴

```
C:₩>kubectl get deploy -o wide
NAME                READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES        SELECTOR
kuweb-deployment    3/3     3            3           30m   kuweb        testweb:1.0   app=kuweb-label
```

Deployment 사용하기

✓ Deployment 삭제

kubectl delete deploy <deploy명 | -l label | --all> [옵션]

> kubectl delete deploy kuweb-deployment

```
C:₩>kubectl delete deploy kuweb-deployment
deployment.apps "kuweb-deployment" deleted
```

- Deployment에 의해 생성된 ReplicaSet, Pod도 같이 삭제됨

```
C:₩>kubectl get deploy
No resources found in default namespace.

C:₩>kubectl get rs
No resources found in default namespace.

C:₩>kubectl get po
NAME        READY   STATUS    RESTARTS   AGE
kuserver    1/1     Running   1          25h
```

CHAPTER 11.
Service 사용하기

Service 사용하기

✓ Service

Pod 집합에서 실행 중인 애플리케이션을 네트워크 서비스로 노출하는 추상적인 방법

Pods에 접근하기 위한 규칙을 정의

하나의 애플리케이션에 하나의 Pod만 사용하지 않고 Pod의 집합을 사용하는 경우 각 Pod는 고유의 IP주소를 가지기 때문에 접근이 까다로움

Deployment를 사용하는 경우 Pod는 동적으로 생성/소멸 될 수 있기 때문에 배포 중 동작되는 **Pod의 IP주소는 변경될 수 있음**

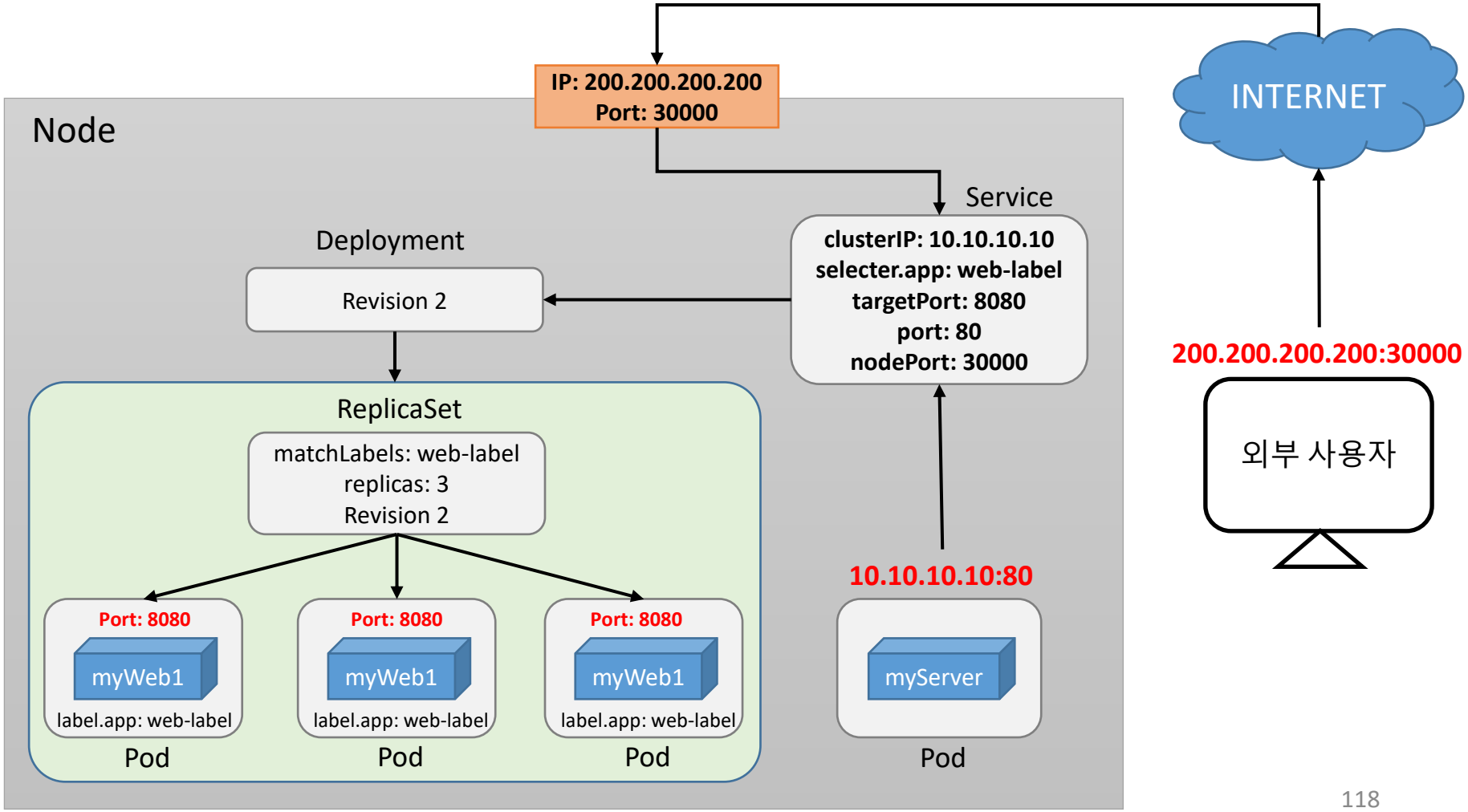
Service 사용하기

✓ Service Type

타입	설명
ClusterIP	클러스터 내부 IP에 서비스를 노출 **클러스터 내에서만 서비스에 접근 가능** 외부로 Pods를 노출하지 않음 기본 ServiceType
NodePort	Pods에 접근할 수 있는 Port를 클러스터의 모든 Node에 동일하게 개방 **외부에서 서비스에 접근 가능** ClusterIP 서비스가 자동으로 같이 생성됨 접근 Port는 기본 랜덤 하게 지정됨 특정 Port로 접근하도록 설정 가능
LoadBalancer	클라우드 공급자의 로드 밸런서를 사용하여 서비스를 외부에 노출 **외부에서 서비스에 접근 가능** **클라우드 환경에서만 사용 가능** NodePort 및 ClusterIP 서비스가 자동으로 같이 생성됨

Service 사용하기

✓ **NodePort Service**



Service 사용하기

✓ Service API 확인

> **kubctl api-resources**

```
C:₩>kubectl api-resources
NAME                              SHORTNAMES   APIVERSION                             NAMESPACED   KIND
bindings                                       v1                                     true         Binding
componentstatuses                 cs           v1                                     false        ComponentStatus
configmaps                        cm           v1                                     true         ConfigMap
endpoints                         ep           v1                                     true         Endpoints
events                            ev           v1                                     true         Event
limitranges                       limits       v1                                     true         LimitRange
namespaces                        ns           v1                                     false        Namespace
nodes                             no           v1                                     false        Node
persistentvolumeclaims            pvc          v1                                     true         PersistentVolumeClaim
persistentvolumes                 pv           v1                                     false        PersistentVolume
pods                              po           v1                                     true         Pod
podtemplates                                   v1                                     true         PodTemplate
replicationcontrollers            rc           v1                                     true         ReplicationController
resourcequotas                    quota        v1                                     true         ResourceQuota
secrets                                        v1                                     true         Secret
serviceaccounts                   sa           v1                                     true         ServiceAccount
services                          svc          v1                                     true         Service
mutatingwebhookconfigurations                  admissionregistration.k8s.io/v1        false        MutatingWebhookConfiguration
validatingwebhookconfigurations                admissionregistration.k8s.io/v1        false        ValidatingWebhookConfiguration
customresourcedefinitions         crd,crds     apiextensions.k8s.io/v1                false        CustomResourceDefinition
apiservices                                    apiregistration.k8s.io/v1              false        APIService
controllerrevisions                            apps/v1                                true         ControllerRevision
daemonsets                        ds           apps/v1                                true         DaemonSet
deployments                       deploy       apps/v1                                true         Deployment
replicasets                       rs           apps/v1                                true         ReplicaSet
statefulsets                      sts          apps/v1                                true         StatefulSet
```

Service 사용하기

✓ Service 연결할 Deployment 생성

- Yaml 파일 작성
- d:₩KubeTestWeb₩kuweb-deployment.yaml 파일 생성

kuweb-deployment.yaml - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kuweb-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: kuweb-label
  template:
    metadata:
      name: kuweb-pod
      labels:
        app: kuweb-label
    spec:
      containers:
      - name: kuweb
        image: testweb:1.0
        ports:
        - containerPort: 8080
          protocol: TCP
```

Service 사용하기

✓ Service 연결할 Deployment 생성

> **kubectl apply -f d:₩KubeTestWeb₩kuweb-deployment.yaml**

```
C:₩>kubectl apply -f d:₩KubeTestWeb₩kuweb-deployment.yaml
deployment.apps/kuweb-deployment created
```

- 생성된 Pods 확인

```
C:₩>kubectl get po -o wide
NAME                                READY   STATUS    RESTARTS   AGE     IP          NODE             NOMINATED NODE   READINESS GATES
kuserver                            1/1     Running   2          2d23h   10.1.0.75   docker-desktop   <none>           <none>
kuweb-deployment-54f4f94f95-bjxcj   1/1     Running   0          4m31s   10.1.0.78   docker-desktop   <none>           <none>
kuweb-deployment-54f4f94f95-p8w6k   1/1     Running   0          4m31s   10.1.0.79   docker-desktop   <none>           <none>
kuweb-deployment-54f4f94f95-vxltk   1/1     Running   0          4m31s   10.1.0.77   docker-desktop   <none>           <none>
```

Service 사용하기

✓ Service Yaml 파일 작성

- d:₩KubeTestWeb₩kuweb-serivce.yaml 파일 생성

```yaml
apiVersion: v1
kind: Service
metadata:
  name: kuweb-service
spec:
  type: NodePort
  selector:
    app: kuweb-label
  ports:
  - name: kuweb-port
    targetPort: 8080
    port: 80
    nodePort: 30000
```

생성할 Service 이름

생성할 Service 정의
type - 서비스 방식
selector - Pod 선택기(연결할 Pod를 찾는 방법)

Port 정의
Name – Port 이름
targetPort – Pods에서 사용중인 Port
port – 추상화된 서비스 Port 지정
nodePort – 외부에서 연결할 Port 지정

kuweb-service.yaml - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```yaml
apiVersion: v1
kind: Service
metadata:
  name: kuweb-service
spec:
  type: NodePort
  selector:
    app: kuweb-label
  ports:
  - name: kuweb-port
    targetPort: 8080
    port: 80
    nodePort: 30000
```

Service 사용하기

✓ Service 생성

kubectl apply -f <Yaml 파일 경로> [옵션]

> **kubectl apply -f d:₩KubeTestWeb₩kuweb-serivce.yaml**

```
C:₩>kubectl apply -f d:₩kubeTestWeb₩kuweb-service.yaml
service/kuweb-service created
```

Service 사용하기

✓ Service 목록 확인

kubectl get services [옵션]

> **kubectl get svc -o wide**

- 유지되는 Pod 개수 확인 가능

* svc : services의 짧은 명령어

```
C:₩>kubectl get svc -o wide
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE     SELECTOR
kubernetes      ClusterIP   10.96.0.1       <none>        443/TCP        3d22h   <none>
kuweb-service   NodePort    10.99.115.180   <none>        80:30000/TCP   51s     app=kuweb-label
```

컬럼	설명
NAME	서비스 이름
TYPE	서비스 타입
CLUSTER-IP	클러스터 내부에서 접근 가능한 IP주소
PORT(S)	서비스 PORT 번호 / 외부 접근용 PORT 번호 / 프로토콜
SELECTOR	연결할 Pods 를 식별하기 위한 정보

Service 사용하기

✓ Service 상세 정보 확인

kubectl describe svc [옵션]

> **kubectl describe svc kuweb-service**

```
C:₩>kubectl describe svc kuweb-service
Name:                     kuweb-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=kuweb-label
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.99.115.180
IPs:                      10.99.115.180
LoadBalancer Ingress:     localhost
Port:                     kuweb-port  80/TCP
TargetPort:               8080/TCP
NodePort:                 kuweb-port  30000/TCP
Endpoints:                10.1.0.77:8080,10.1.0.78:8080,10.1.0.79:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

ClusterIP 주소

Port 정보

현재 연결된 Pods 정보

Service 사용하기

✓ 클러스터 내부에서 접속 확인

<ClusterIP or 서비스이름>:<서비스 Port>

> **kubectl exec -it kuserver – bash**

- 클러스터 내부에서 접근할 경우 추상화된 서비스 Port로 접근
- 동일한 IP로 접근해도 실제 연결되는 Pod가 달라짐

```
[root@kuserver /]# curl 10.99.115.180:80

<h1>TEST WEB v1</h1>
Server IP Address : 10.1.0.78

[root@kuserver /]# curl 10.99.115.180:80

<h1>TEST WEB v1</h1>
Server IP Address : 10.1.0.79

[root@kuserver /]# curl 10.99.115.180:80

<h1>TEST WEB v1</h1>
Server IP Address : 10.1.0.77
```

```
[root@kuserver /]# curl kuweb-service:80

<h1>TEST WEB v1</h1>
Server IP Address : 10.1.0.78

[root@kuserver /]# curl kuweb-service:80

<h1>TEST WEB v1</h1>
Server IP Address : 10.1.0.79

[root@kuserver /]# curl kuweb-service:80

<h1>TEST WEB v1</h1>
Server IP Address : 10.1.0.77
```

Service 사용하기

✓ 클러스터 내부에서 접속 확인

- ClusterIP로는 targetPort(8080번)로 접속 불가능

```
[root@kuserver /]# curl 10.99.115.180:8080
curl: (7) Failed connect to 10.99.115.180:8080; Connection refused
```

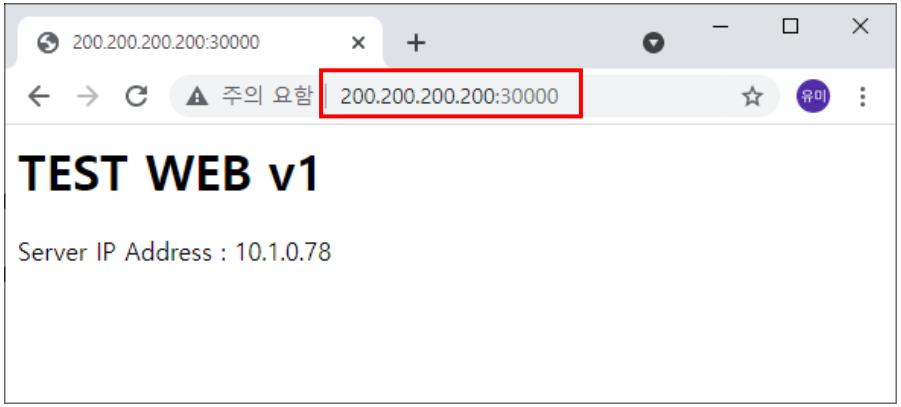
- ClusterIP로는 nodePort(30000번)로 접속 불가능

```
[root@kuserver /]# curl 10.99.115.180:30000
curl: (7) Failed connect to 10.99.115.180:30000; Connection refused
```

Service 사용하기

✓ 클러스터 외부에서 접속 확인

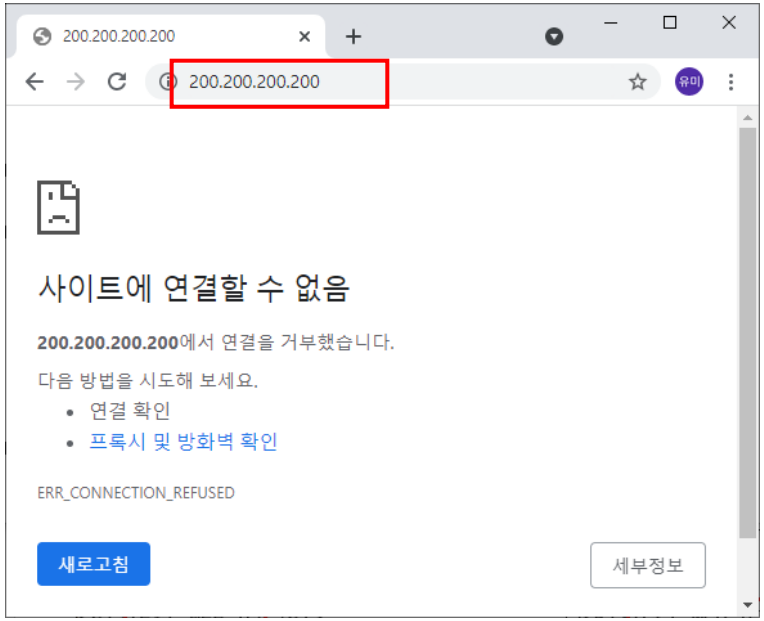
<Node(호스트) IP>:<nodePort>

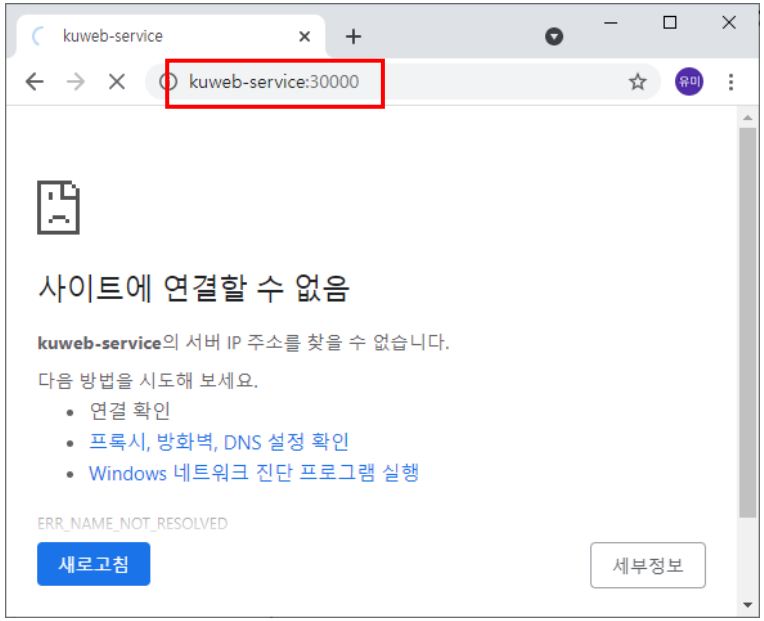


Service 사용하기

✓ 클러스터 외부에서 접속 확인

- 외부에서 접속 시 추상화된 서비스 Port(80번)로 접근 불가능
- 외부에서 접속 시 서비스 이름으로 접속 불가능





Service 사용하기

✓ Service 삭제

kubectl delete svc <svc명 | -l label | --all> [옵션]

> kubectl delete svc kuweb-service

```
C:₩>kubectl delete svc kuweb-service
service "kuweb-service" deleted
```

PART 3.
애플리케이션 배포 설정

CHAPTER 12.
사용량 관리하기

사용량 관리하기

✓ 사용량 관리

클러스터 내부에서 컴퓨팅 자원을 효율적으로 활용하는 것이 중요함

실행 중인 컨테이너와 유휴 상태의 컨테이너의 CPU, 메모리 등의 자원을 효율적으로 할당하기 위해 각 컨테이너의 자원 사용량을 적절히 제한해야 함

사용량 관리하기

✓ Limits

컨테이너와 Pod의 자원 사용량 제한

Pod 자체에 최대 자원 사용량을 명시적으로 지정

지정하지 않으면 Node의 모든 자원 사용 가능

하나의 Pod에 여러 컨테이너가 정의되어 있는 경우 컨테이너 별로 각각 자원 할당 가능

메모리 사용량 단위 : Mi (Mebibyte)

CPU 사용량 단위 : m(밀리코어), CPU 1개가 1000m에 해당

✓ Limits 상세 정보 확인

- Pod가 생성된 Node 확인

```
C:₩>kubectl get po -o wide
NAME                                  READY   STATUS    RESTARTS   AGE    IP          NODE             NOMINATED NODE   READINESS GATES
kuserver                              1/1     Running   2          3d1h   10.1.0.75   docker-desktop   <none>           <none>
kuweb-deployment-54f4f94f95-bjxcj     1/1     Running   0          148m   10.1.0.78   docker-desktop   <none>           <none>
kuweb-deployment-54f4f94f95-p8w6k     1/1     Running   0          148m   10.1.0.79   docker-desktop   <none>           <none>
kuweb-deployment-54f4f94f95-vxltk     1/1     Running   0          148m   10.1.0.77   docker-desktop   <none>           <none>
```

사용량 관리하기

✓ Limits 상세 정보 확인

- Node 상세 정보 확인

> **kubectl describe nodes docker-desktop**

컨테이너별 사용량

```
Non-terminated Pods:          (13 in total)
  Namespace                   Name                                       CPU Requests  CPU Limits  Memory Requests  Memory Limits  Age
  ---------                   ----                                       ------------  ----------  ---------------  -------------  ---
  default                     kuserver                                   0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d1h
  default                     kuweb-deployment-54f4f94f95-bjxcj          0 (0%)        0 (0%)      0 (0%)           0 (0%)         149m
  default                     kuweb-deployment-54f4f94f95-p8w6k          0 (0%)        0 (0%)      0 (0%)           0 (0%)         149m
  default                     kuweb-deployment-54f4f94f95-vxltk          0 (0%)        0 (0%)      0 (0%)           0 (0%)         149m
  kube-system                 coredns-558bd4d5db-2ttvl                   100m (1%)     0 (0%)      70Mi (0%)        170Mi (1%)     4d
  kube-system                 coredns-558bd4d5db-2tfnr                   100m (1%)     0 (0%)      70Mi (0%)        170Mi (1%)     4d
  kube-system                 etcd-docker-desktop                        100m (1%)     0 (0%)      100Mi (0%)       0 (0%)         4d
  kube-system                 kube-apiserver-docker-desktop              250m (3%)     0 (0%)      0 (0%)           0 (0%)         4d
  kube-system                 kube-controller-manager-docker-desktop     200m (2%)     0 (0%)      0 (0%)           0 (0%)         4d
  kube-system                 kube-proxy-9k8xs                           0 (0%)        0 (0%)      0 (0%)           0 (0%)         4d
  kube-system                 kube-scheduler-docker-desktop              100m (1%)     0 (0%)      0 (0%)           0 (0%)         4d
  kube-system                 storage-provisioner                        0 (0%)        0 (0%)      0 (0%)           0 (0%)         4d
  kube-system                 vpnkit-controller                          0 (0%)        0 (0%)      0 (0%)           0 (0%)         4d
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource           Requests    Limits
  --------           --------    ------
  cpu                850m (10%)  0 (0%)
  memory             240Mi (1%)  340Mi (2%)
  ephemeral-storage  0 (0%)      0 (0%)
  hugepages-1Gi      0 (0%)      0 (0%)
  hugepages-2Mi      0 (0%)      0 (0%)
Events:              <none>
```

전체 사용량

* 전체 출력 내용 중 하단 부분에 있음

사용량 관리하기

✓ Limits 상세 정보 확인

- 현재 시스템 정보 확인

> **kubectl describe nodes docker-desktop**

```
Addresses:
  InternalIP:  192.168.65.4
  Hostname:    docker-desktop
Capacity:
  cpu:                8
  ephemeral-storage:  263174212Ki
  hugepages-1Gi:      0
  hugepages-2Mi:      0
  memory:             13038072Ki
  pods:               110
Allocatable:
  cpu:                8
  ephemeral-storage:  242541353378
  hugepages-1Gi:      0
  hugepages-2Mi:      0
  memory:             12935672Ki
  pods:               110
```

* 전체 출력 내용 중 중반 부분에 있음

사용량 관리하기

✓ Limits 설정

- 기존 Deployment 파일을 복사해서 하단에 내용 추가
- d:₩KubeTestWeb₩kuweb-limit.yaml 파일 생성

```
... 상단 내용 생략 ...

spec:
    containers:
    - name: kuweb
      image: testweb:1.0
      ports:
      - containerPort: 8080
        protocol: TCP
      resources:
        limits:
          memory: "512Mi"
          cpu: "1000m"
```

컨테이너 정보 하단 부분에 추가

사용량 제한 설정

사용량 관리하기

✓ Limits 설정

- d:₩KubeTestWeb₩kuweb-limit.yaml 파일 생성

kuweb-limit.yaml - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kuweb-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: kuweb-label
  template:
    metadata:
      name: kuweb-pod
      labels:
        app: kuweb-label
    spec:
      containers:
      - name: kuweb
        image: testweb:1.0
        ports:
        - containerPort: 8080
          protocol: TCP
        resources:
          limits:
            memory: "512Mi"
            cpu: "1000m"
```

사용량 관리하기

✓ Limits 적용

> **kubectl apply -f d:₩KubeTestWeb₩kuweb-limit.yaml**

```
C:₩>kubectl apply -f d:₩kubeTest₩eb₩kuweb-limit.yaml
deployment.apps/kuweb-deployment configured
```

* 이미 이전에 만들어져 있는 상태면 업데이트됨

- 생성된 Pods 확인

```
C:₩>kubectl get po -o wide
NAME                                READY   STATUS    RESTARTS   AGE    IP          NODE             NOMINATED NODE   READINESS GATES
kuserver                            1/1     Running   2          3d1h   10.1.0.75   docker-desktop   <none>           <none>
kuweb-deployment-5b66bb4b88-4xfxn   1/1     Running   0          108s   10.1.0.82   docker-desktop   <none>           <none>
kuweb-deployment-5b66bb4b88-56dvw   1/1     Running   0          112s   10.1.0.80   docker-desktop   <none>           <none>
kuweb-deployment-5b66bb4b88-q4cb9   1/1     Running   0          110s   10.1.0.81   docker-desktop   <none>           <none>
```

사용량 관리하기

✓ Limits 적용 확인

- Node 상세 정보 확인

> **kubectl describe nodes docker-desktop**

```
Non-terminated Pods:          (13 in total)
  Namespace                   Name                                       CPU Requests  CPU Limits  Memory Requests  Memory Limits  Age
  ---------                   ----                                       ------------  ----------  ---------------  -------------  ---
  default                     kuserver                                   0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d1h
  default                     kuweb-deployment-5b66bb4b88-4xfxn          1 (12%)       1 (12%)     512Mi (4%)       512Mi (4%)     2m3s
  default                     kuweb-deployment-5b66bb4b88-56dvw          1 (12%)       1 (12%)     512Mi (4%)       512Mi (4%)     2m7s
  default                     kuweb-deployment-5b66bb4b88-q4cb9          1 (12%)       1 (12%)     512Mi (4%)       512Mi (4%)     2m5s
  kube-system                 coredns-558bd4d5db-2ftvl                   100m (1%)     0 (0%)      70Mi (0%)        170Mi (1%)     4d1h
  kube-system                 coredns-558bd4d5db-2tfnr                   100m (1%)     0 (0%)      70Mi (0%)        170Mi (1%)     4d1h
  kube-system                 etcd-docker-desktop                        100m (1%)     0 (0%)      100Mi (0%)       0 (0%)         4d1h
  kube-system                 kube-apiserver-docker-desktop              250m (3%)     0 (0%)      0 (0%)           0 (0%)         4d1h
  kube-system                 kube-controller-manager-docker-desktop     200m (2%)     0 (0%)      0 (0%)           0 (0%)         4d1h
  kube-system                 kube-proxy-9k8xs                           0 (0%)        0 (0%)      0 (0%)           0 (0%)         4d1h
  kube-system                 kube-scheduler-docker-desktop              100m (1%)     0 (0%)      0 (0%)           0 (0%)         4d1h
  kube-system                 storage-provisioner                        0 (0%)        0 (0%)      0 (0%)           0 (0%)         4d1h
  kube-system                 vpnkit-controller                          0 (0%)        0 (0%)      0 (0%)           0 (0%)         4d1h
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource           Requests     Limits
  --------           --------     ------
  cpu                3850m (48%)  3 (37%)
  memory             1776Mi (14%) 1876Mi (14%)
  ephemeral-storage  0 (0%)       0 (0%)
  hugepages-1Gi      0 (0%)       0 (0%)
  hugepages-2Mi      0 (0%)       0 (0%)
Events:              <none>
```

* 전체 출력 내용 중 하단 부분에 있음

사용량 관리하기

✓ **Requests**

컨테이너가 사용하고자 하면 최소한 보장되어야 할 자원의 양
자원의 오버커밋을 가능하게 만드는 기능

✓ **오버커밋(Overcommit)**

한정된 자원을 효율적으로 사용하기 위한 방법

유휴상태의 컨테이너가 할당 받을 자원의 양을 줄이고, 실행 중인 컨테이너가 더 많은 자원을 사용할 수 있도록 해서 효율성을 높임

컨테이너들의 자원 할당량을 합하면 물리적으로 사용할 수 있는 자원보다 많은 양의 자원이 할당될 수 있음

사용량 관리하기

✓ Requests 설정

- 기존 limits 파일을 복사해서 하단에 내용 추가
- d:₩KubeTestWeb₩kuweb-request.yaml 파일 생성

```
... 상단 내용 생략 ...

spec:
        ... 생략 ...
    resources:
      limits:
        memory: "512Mi"
        cpu: "1000m“
      requests:
        memory: “256Mi”
        cpu: “500m”
```

→ 사용량 제한 설정

사용량 관리하기

✓ Requests 설정

- d:₩KubeTestWeb₩kuweb-request.yaml 파일 생성

kuweb-request.yaml - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kuweb-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: kuweb-label
  template:
    metadata:
      name: kuweb-pod
      labels:
        app: kuweb-label
    spec:
      containers:
      - name: kuweb
        image: testweb:1.0
        ports:
        - containerPort: 8080
          protocol: TCP
        resources:
          limits:
            memory: "512Mi"
            cpu: "1000m"
          requests:
            memory: "256Mi"
            cpu: "500m"
```

사용량 관리하기

✓ Requests 적용

> **kubectl apply -f d:₩KubeTestWeb₩kuweb-request.yaml**

```
C:₩>kubectl apply -f d:₩kubeTestWeb₩kuweb-request.yaml
deployment.apps/kuweb-deployment configured
```

사용량 관리하기

✓ Requests 적용 확인

- Node 상세 정보 확인

> **kubectl describe nodes docker-desktop**

```
Non-terminated Pods:          (13 in total)
  Namespace                   Name                                       CPU Requests  CPU Limits  Memory Requests  Memory Limits  Age
  ---------                   ----                                       ------------  ----------  ---------------  -------------  ---
  default                     kuserver                                   0 (0%)        0 (0%)      0 (0%)           0 (0%)         3d2h
  default                     kuweb-deployment-56f49cd6c4-2ppt6          500m (6%)     1 (12%)     256Mi (2%)       512Mi (4%)     32s
  default                     kuweb-deployment-56f49cd6c4-5qhwt          500m (6%)     1 (12%)     256Mi (2%)       512Mi (4%)     30s
  default                     kuweb-deployment-56f49cd6c4-c9txd          500m (6%)     1 (12%)     256Mi (2%)       512Mi (4%)     34s
  kube-system                 coredns-558bd4d5db-2ftvl                   100m (1%)     0 (0%)      70Mi (0%)        170Mi (1%)     4d1h
  kube-system                 coredns-558bd4d5db-2tfnr                   100m (1%)     0 (0%)      70Mi (0%)        170Mi (1%)     4d1h
  kube-system                 etcd-docker-desktop                        100m (1%)     0 (0%)      100Mi (0%)       0 (0%)         4d1h
  kube-system                 kube-apiserver-docker-desktop              250m (3%)     0 (0%)      0 (0%)           0 (0%)         4d1h
  kube-system                 kube-controller-manager-docker-desktop     200m (2%)     0 (0%)      0 (0%)           0 (0%)         4d1h
  kube-system                 kube-proxy-9k8xs                           0 (0%)        0 (0%)      0 (0%)           0 (0%)         4d1h
  kube-system                 kube-scheduler-docker-desktop              100m (1%)     0 (0%)      0 (0%)           0 (0%)         4d1h
  kube-system                 storage-provisioner                        0 (0%)        0 (0%)      0 (0%)           0 (0%)         4d1h
  kube-system                 vpnkit-controller                          0 (0%)        0 (0%)      0 (0%)           0 (0%)         4d1h
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource           Requests     Limits
  --------           --------     ------
  cpu                2350m (29%)  3 (37%)
  memory             1008Mi (7%)  1876Mi (14%)
  ephemeral-storage  0 (0%)       0 (0%)
  hugepages-1Gi      0 (0%)       0 (0%)
  hugepages-2Mi      0 (0%)       0 (0%)
Events:              <none>
```

사용량 관리하기

✓ ResourceQuota

네임스페이스에서 사용할 수 있는 자원 사용량의 총합을 제한
네임스페이스에서 생성할 수 있는 리소스의 개수를 제한

- 네임스페이스 확인

> **kubectl get namespace**

```
C:₩>kubectl get namespace
NAME              STATUS   AGE
default           Active   4d5h
kube-node-lease   Active   4d5h
kube-public       Active   4d5h
kube-system       Active   4d5h
```

> **kubectl describe deploy kuweb-deployment**

```
C:₩>kubectl describe deploy kuweb-deployment
Name:                   kuweb-deployment
Namespace:              default
```

사용량 관리하기

✓ ResourceQuota 설정

- d:₩KubeTestWeb₩kuweb-quota.yaml 파일 생성

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: kuweb-quota
  namespace: default
spec:
  hard:
    requests.cpu: “1000m”
    requests.memory: “1024Mi”
    limits.cpu: “2000m”
    limits.memory: “2048Mi”
```

→ 적용할 네임스페이스 지정

→ 사용량 제한 설정

사용량 관리하기

✓ ResourceQuota 설정

- d:₩KubeTestWeb₩kuweb-quota.yaml 파일 생성

kuweb-quota.yaml - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: kuweb-quota
  namespace: default
spec:
  hard:
    requests.cpu: "1000m"
    requests.memory: "1024Mi"
    limits.cpu: "2000m"
    limits.memory: "2048Mi"
```

✓ ResourceQuota 적용

> **kubectl apply -f d:₩KubeTestWeb₩kuweb-quota.yaml**

```
C:₩>kubectl apply -f d:₩kubeTestWeb₩kuweb-quota.yaml
resourcequota/kuweb-quota created
```

사용량 관리하기

✓ ResourceQuota 적용 확인

- resoutcequota 조회해도 되고, 그냥 quota로 조회해도 됨

> **kubectl get resourcequota**

```
C:₩>kubectl get resourcequota
NAME          AGE   REQUEST                                                 LIMIT
kuweb-quota   41s   requests.cpu: 1500m/1, requests.memory: 768Mi/1Gi   limits.cpu: 3/2, limits.memory: 1536Mi/2Gi
```

> **kubectl describe quota**

```
C:₩>kubectl describe quota
Name:           kuweb-quota
Namespace:      default
Resource        Used    Hard
--------        ----    ----
limits.cpu      3       2
limits.memory   1536Mi  2Gi
requests.cpu    1500m   1
requests.memory 768Mi   1Gi
```

✓ ResourceQuota 삭제

> **Kubectl delete quota <삭제할 quota이름>**

```
C:₩>kubectl delete quota --all
resourcequota "kuweb-quota" deleted
```

CHAPTER 13.
스케줄링 설정하기

스케줄링 설정하기

✓ 스케줄링

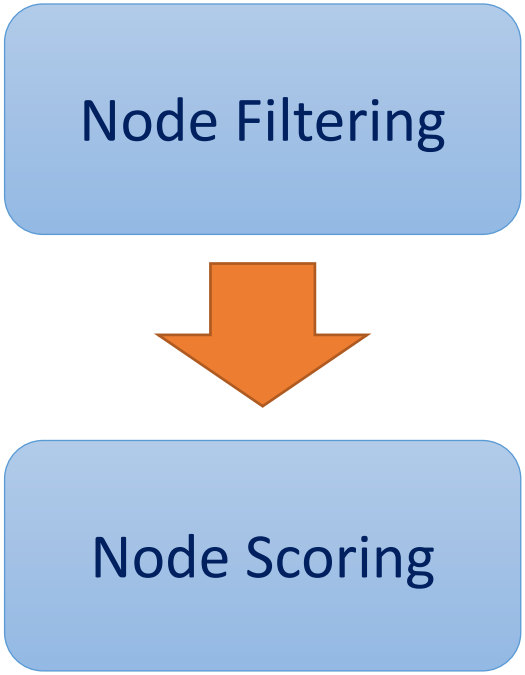
컨테이너를 새롭게 생성할 때 어느 서버에 생성할 것일지 결정하는 것

컨테이너를 생성하기 전에 특정 목적에 최대한 부합하는 Node를 선택하는 작업

여러 개의 Node를 사용하는 경우 해당됨

스케줄링 설정하기

✓ Node 선택 과정



Pod를 할당할 수 있는 Node와 그렇지 못한 Node를 분리해서 Filtering 하는 단계

Filtering 후 선정된 Node목록을 Node Scoring 단계로 전달

미리 정의된 알고리즘의 가중치에 따라 Node의 점수를 계산하여 가장 점수가 높은 Node를 최종적으로 선택

스케줄링 설정하기

✓ nodeName 설정

- Yaml 파일에 Node 이름을 직접 명시하는 방법
- d:₩KubeTestWeb₩kuweb-nodename.yaml 파일 생성

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: kuweb-pod
spec:
  nodeName: docker-desktop
  containers:
  - name: kuweb2
    image: testweb:1.0
    ports:
    - containerPort: 8080
```

→ 사용할 Node 이름 지정

스케줄링 설정하기

✓ nodeName 설정

- d:₩KubeTestWeb₩kuweb-nodename.yaml 파일 생성

kuweb-nodename.yaml - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
apiVersion: v1
kind: Pod
metadata:
  name: kuweb-pod
spec:
  nodeName: docker-desktop
  containers:
  - name: kuweb2
    image: testweb:1.0
    ports:
    - containerPort: 8080
```

스케줄링 설정하기

✓ nodeName 적용

> **kubectl apply -f d:₩KubeTestWeb₩kuweb-nodename.yaml**

```
C:₩>kubectl apply -f d:₩kubeTestWeb₩kuweb-nodename.yaml
pod/kuweb-pod created
```

✓ nodeName 적용 확인

> **Kubectl get po -o wide**

```
C:₩>kubectl get po -o wide
NAME                                READY   STATUS    RESTARTS   AGE     IP          NODE             NOMINATED NODE   READINESS GATES
kuserver                            1/1     Running   2          3d7h    10.1.0.75   docker-desktop   <none>           <none>
kuweb-deployment-56f49cd6c4-2ppt6   1/1     Running   0          4h31m   10.1.0.84   docker-desktop   <none>           <none>
kuweb-deployment-56f49cd6c4-5qhwt   1/1     Running   0          4h31m   10.1.0.85   docker-desktop   <none>           <none>
kuweb-deployment-56f49cd6c4-c9txd   1/1     Running   0          4h31m   10.1.0.83   docker-desktop   <none>           <none>
kuweb-pod                           1/1     Running   0          40s     10.1.0.86   docker-desktop   <none>           <none>
```

* 현재는 Node가 1개밖에 없기 때문에 모두 동일한 Node에 생성됨

스케줄링 설정하기

✓ nodeSelector

Node Label을 이용해 선택하는 방법
모든 Node에는 기본적으로 설정되어 있는 라벨이 있음
라벨을 이용해 Node를 특징 별로 분류해서 선택 가능
필요한 경우 추가 가능

✓ Node에 지정된 Label 확인

> **kubectl get nodes --show-labels**

```
C:₩>kubectl get nodes --show-labels
NAME             STATUS   ROLES                  AGE    VERSION   LABELS
docker-desktop   Ready    control-plane,master   4d5h   v1.21.5   beta.kubernetes.io/arch=amd64,
beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=docker-desktop,kuber
netes.io/os=linux,node-role.kubernetes.io/control-plane=,node-role.kubernetes.io/master=,node.ku
bernetes.io/exclude-from-external-load-balancers=
```

스케줄링 설정하기

✓ Node Label 추가

kubectl label nodes <Node이름> <추가할 Label명=Value>

> **kubectl label nodes docker-desktop kh.iei/project=kuwebnode1**

```
C:₩>kubectl label nodes docker-desktop kh.iei/project=kuwebnode1
node/docker-desktop labeled
```

- 추가된 Label 확인

```
C:₩>kubectl get nodes --show-labels
NAME             STATUS   ROLES                  AGE    VERSION   LABELS
docker-desktop   Ready    control-plane,master   4d6h   v1.21.5   beta.kubernetes.io/arch=amd64,
beta.kubernetes.io/os=linux,kh.iei/project=kuwebnode1,kubernetes.io/arch=amd64,kubernetes.io/hos
tname=docker-desktop,kubernetes.io/os=linux,node-role.kubernetes.io/control-plane=,node-role.kub
ernetes.io/master=,node.kubernetes.io/exclude-from-external-load-balancers=
```

스케줄링 설정하기

✓ nodeSelector 설정

- d:₩KubeTestWeb₩kuweb-nodeselector.yaml 파일 생성

```
apiVersion: v1
kind: Pod
metadata:
  name: kuweb-pod2
spec:
  nodeSelector:
    kh.iei/project: kuwebnode1
  containers:
  - name: kuweb2
    image: testweb:1.0
    ports:
    - containerPort: 8080
```

→ 사용할 Node 이름 지정

스케줄링 설정하기

✓ nodeSelector 설정

- d:₩KubeTestWeb₩kuweb-nodeselector.yaml 파일 생성

kuweb-nodeselector.yaml - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: kuweb-pod2
spec:
  nodeSelector:
    kh.iei/project: kuwebnode1
  containers:
  - name: kuweb2
    image: testweb:1.0
    ports:
    - containerPort: 8080
```

스케줄링 설정하기

✓ nodeSelector 적용

> kubectl apply -f d:₩KubeTestWeb₩kuweb-nodeselector.yaml

```
C:₩>kubectl apply -f d:₩kubeTestWeb₩kuweb-nodeselector.yaml
pod/kuweb-pod2 created
```

✓ nodeName 적용 확인

> Kubectl get po -o wide

```
C:₩>kubectl get po -o wide
NAME                                READY   STATUS    RESTARTS   AGE     IP          NODE             NOMINATED NODE   READINESS GATES
kuserver                            1/1     Running   2          3d7h    10.1.0.75   docker-desktop   <none>           <none>
kuweb-deployment-56f49cd6c4-2ppt6   1/1     Running   0          4h53m   10.1.0.84   docker-desktop   <none>           <none>
kuweb-deployment-56f49cd6c4-5qhwt   1/1     Running   0          4h53m   10.1.0.85   docker-desktop   <none>           <none>
kuweb-deployment-56f49cd6c4-c9txd   1/1     Running   0          4h53m   10.1.0.83   docker-desktop   <none>           <none>
kuweb-pod                           1/1     Running   0          23m     10.1.0.86   docker-desktop   <none>           <none>
kuweb-pod2                          1/1     Running   0          3m1s    10.1.0.87   docker-desktop   <none>           <none>
```

스케줄링 설정하기

✓ Node Label 삭제

Label 뒤에 –(대시)를 입력하면 삭제됨
kubectl label nodes <Node이름> <삭제할Label명>-

> **kubectl label nodes docker-desktop kh.iei/project-**

```
C:₩>kubectl label nodes docker-desktop kh.iei/project-
node/docker-desktop labeled
```

- Label 삭제 확인

```
C:₩>kubectl get nodes --show-labels
NAME             STATUS   ROLES                  AGE    VERSION   LABELS
docker-desktop   Ready    control-plane,master   4d6h   v1.21.5   beta.kubernetes.io/arch=amd64,
beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=docker-desktop,kuber
netes.io/os=linux,node-role.kubernetes.io/control-plane=,node-role.kubernetes.io/master=,node.ku
bernetes.io/exclude-from-external-load-balancers=
```

CHAPTER 14.
컨트롤러 사용하기

컨트롤러 사용하기

✓ 컨트롤러

클러스터의 상태를 관찰한 다음 필요한 경우 변경을 수행하거나 요청하는 제어 루프

각 컨트롤러는 현재 클러스터 상태를 **원하는 상태에 더 가깝게 이동하려고 노력**함

선언적 작업을 수행할 때 컨트롤러에 의해 관리됨

기본 리소스 별 컨트롤러들(Deployment Controller, Node Controller 등)이 **컨트롤러 매니저 컴포넌트에 의해 관리**됨

컨트롤러 사용하기

✓ 쿠버네티스 작업 수행 방식

명령형(Imperative)

kubectl 명령어로 직접 실행하고 결과를 돌려 받는 방식

선언적(Declarative)

Yaml 파일에 최종적으로 도달해야 할 상태를 명시하고 kubectl apply를 통해 수행하는 방식

쿠버네티스는 선언적 방식을 지향함

최종적으로 도달해야 할 상태가 되기 위한 동작을 컨트롤러가 결정함

컨트롤러 사용하기

✓ 컨트롤러 매니저

- kube-system 네임스페이스에서 Pod로 실행됨

```
C:₩>kubectl get po -n kube-system
NAME                                       READY   STATUS    RESTARTS   AGE
coredns-558bd4d5db-2ftvl                   1/1     Running   1          4d12h
coredns-558bd4d5db-2tfnr                   1/1     Running   1          4d12h
etcd-docker-desktop                        1/1     Running   1          4d12h
kube-apiserver-docker-desktop              1/1     Running   1          4d12h
kube-controller-manager-docker-desktop     1/1     Running   1          4d12h
kube-proxy-9k8xs                           1/1     Running   1          4d12h
kube-scheduler-docker-desktop              1/1     Running   1          4d12h
storage-provisioner                        1/1     Running   2          4d12h
vpnkit-controller                          1/1     Running   543        4d12h
```

컨트롤러 사용하기

✓ 커스텀 리소스(Custom Resource)

쿠버네티스에서 기본적으로 제공하는 리소스(Pods, Deployments, Services 등) 외 **사용자가 직접 정의해서 사용하는 리소스**

Deployment, Service 등의 오브젝트의 묶음을 추상화 해서 패키지형태로 사용 가능

별도의 로직을 연동해서 사용하는 것도 가능함

컨트롤러 사용하기

✓ 커스텀 리소스 사용 단계

1. CRD(CustomResourceDefinition) 리소스 생성
2. 커스텀 리소스 생성
3. 컨트롤러 구현 및 실행
4. 컨트롤러가 커스텀 리소스 생성을 감지하고 작업을 수행

컨트롤러 사용하기

✓ CRD 생성

커스텀 리소스의 상세 정보를 정의

> **kubectl api-resources**

```
C:₩>kubectl api-resources
NAME                              SHORTNAMES   APIVERSION                             NAMESPACED   KIND
bindings                                       v1                                     true         Binding
componentstatuses                 cs           v1                                     false        ComponentStatus
configmaps                        cm           v1                                     true         ConfigMap
endpoints                         ep           v1                                     true         Endpoints
events                            ev           v1                                     true         Event
limitranges                       limits       v1                                     true         LimitRange
namespaces                        ns           v1                                     false        Namespace
nodes                             no           v1                                     false        Node
persistentvolumeclaims            pvc          v1                                     true         PersistentVolumeClaim
persistentvolumes                 pv           v1                                     false        PersistentVolume
pods                              po           v1                                     true         Pod
podtemplates                                   v1                                     true         PodTemplate
replicationcontrollers            rc           v1                                     true         ReplicationController
resourcequotas                    quota        v1                                     true         ResourceQuota
secrets                                        v1                                     true         Secret
serviceaccounts                   sa           v1                                     true         ServiceAccount
services                          svc          v1                                     true         Service
mutatingwebhookconfigurations                  admissionregistration.k8s.io/v1        false        MutatingWebhookConfiguration
validatingwebhookconfigurations                admissionregistration.k8s.io/v1        false        ValidatingWebhookConfiguration
customresourcedefinitions         crd,crds     apiextensions.k8s.io/v1                false        CustomResourceDefinition
apiservices                                    apiregistration.k8s.io/v1              false        APIService
controllerrevisions                            apps/v1                                true         ControllerRevision
daemonsets                        ds           apps/v1                                true         DaemonSet
deployments                       deploy       apps/v1                                true         Deployment
replicasets                       rs           apps/v1                                true         ReplicaSet
statefulsets                      sts          apps/v1                                true         StatefulSet
```

컨트롤러 사용하기

✓ CRD 생성

Yaml 파일로 CRD 생성 정의

- d:₩KubeTestWeb₩kuweb-custom.yaml 파일 생성

```
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: khwebs.iei.com
spec:
  group: iei.com
  versions:
    - name: v1
      served: true
      storage: true
```

컨트롤러 사용하기

✓ CRD 생성

Yaml 파일로 CRD 생성 정의

- d:₩KubeTestWeb₩kuweb-custom.yaml 파일 생성

```
schema:
  openAPIV3Schema:
    type: object
    properties:
      replicas:
        type: integer
        minimum: 1
        maximum: 10
      image:
        type: string
```

컨트롤러 사용하기

✓ CRD 생성

Yaml 파일로 CRD 생성 정의

- d:₩KubeTestWeb₩kuweb-custom.yaml 파일 생성

```
scope: Namespaced
  names:
    plural: khwebs
    singular: khweb
    kind: Khweb
    shortNames: ["kh"]
```

컨트롤러 사용하기

✓ CRD 생성

Yaml 파일로 CRD 생성 정의

- d:₩KubeTestWeb₩kuweb-custom.yaml 파일 생성

kuweb-custom.yaml - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```yaml
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: khwebs.iei.com
spec:
  group: iei.com
  versions:
    - name: v1
      served: true
      storage: true
      schema:
        openAPIV3Schema:
          type: object
          properties:
            replicas:
              type: integer
              minimum: 1
              maximum: 10
            image:
              type: string
  scope: Namespaced
  names:
    plural: khwebs
    singular: khweb
    kind: Khweb
    shortNames: ["kh"]
```

컨트롤러 사용하기

✓ CRD 생성

항목	설명
Metadata.name	CRD이름
Spec.group	커스텀 리소스 API 그룹
Spec.versions	커스텀 리소스 API 버전
Spec.versions.schema	커스텀 리소스의 데이터를 정의
Spec.scope	커스텀 리소스가 네임스페이스에 속하는지 여부
Spec.names	커스텀 리소스의 이름

컨트롤러 사용하기

✓ CRD 생성

> **kubectl apply -f d:₩KubeTestWeb₩kuweb-nodeselector.yaml**

```
C:₩>kubectl apply -f d:₩kubeTestWeb₩kuweb-custom.yaml
customresourcedefinition.apiextensions.k8s.io/khwebs.iei.com created
```

- 생성 확인

> **kubectl get crd**

```
C:₩>kubectl get crd
NAME              CREATED AT
khwebs.iei.com    2021-10-15T10:13:29Z
```

CHAPTER 15.
기타 오브젝트들 살펴보기

기타 오브젝트들 살펴보기

✓ 기타 오브젝트

상황에 따라 Pod를 사용하는 상위 오브젝트들

항목	설명
Jobs	특정 동작을 수행하고 종료해야 하는 작업을 위한 오브젝트
DaemonSets	쿠버네티스의 모든 노드에 동일한 포드를 하나씩 생성하는 오브젝트
StatefulSets	상태를 갖는 포드를 관리하기 위한 오브젝트