처음 뵙겠습니다 ?!

▷ 아직 설문 조사 안해주신 분들 꼭 참여해주세요

- https://forms.gle/qhnDSrXqNyNn2rvr9

▷ Camera는 가급적 켜 주시면 대단히 감사하겠습니다 !!!

- 너무 부끄러우면 Snap Camera를 사용하시는 것 까지는~ ^^

▷ 오늘 수업 자료는 아래 링크에서 다운로드 받으실 수 있어요.

- https://github.com/whatwant-school/kubernetes

1st Week

I am ground

Survey

수업은 이렇게 진행하려구요

▷ 2시간 수업이지만, 조금 넘을 수도 있어요 (배고프겠지만, 공부에 대한 열정으로 극복해보아요!)

▷ 마지막 수업은 오프라인(강남캠퍼스)으로 했으면 좋겠어요 (상품 증정식도 있을거에요!)

▷ 로컬 실습 환경은 꼭 구성해서 직접 다루면서 공부할 수 있으면 좋겠습니다.

▷ 2시간 中 1시간은 플립-러닝으로 진행하고, 30분은 summary, 30분은 퀴즈 및 소통의 시간

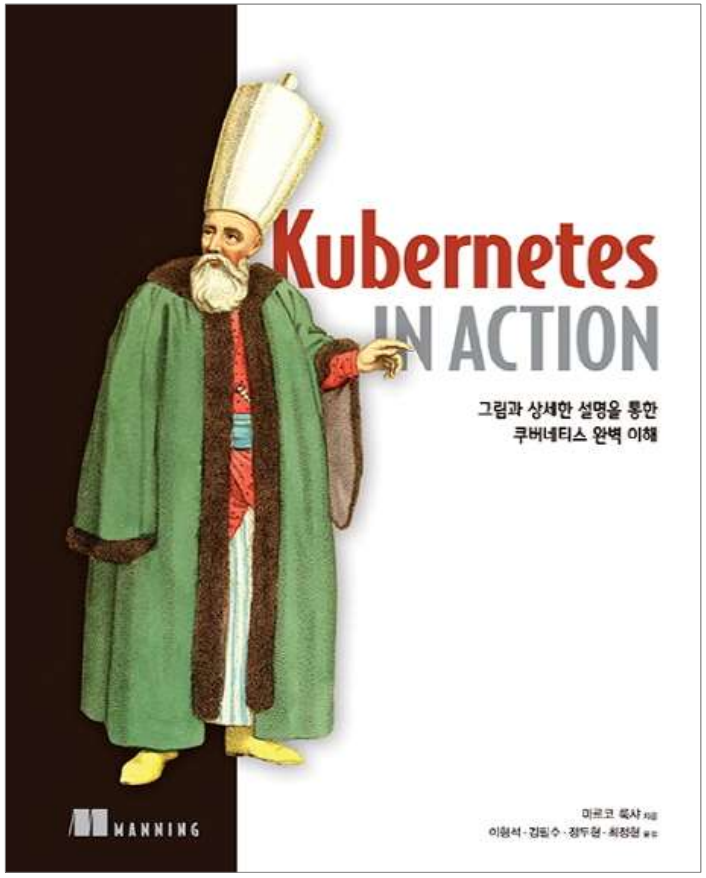
▷ 예습은 선택, 복습은 필수 !!!

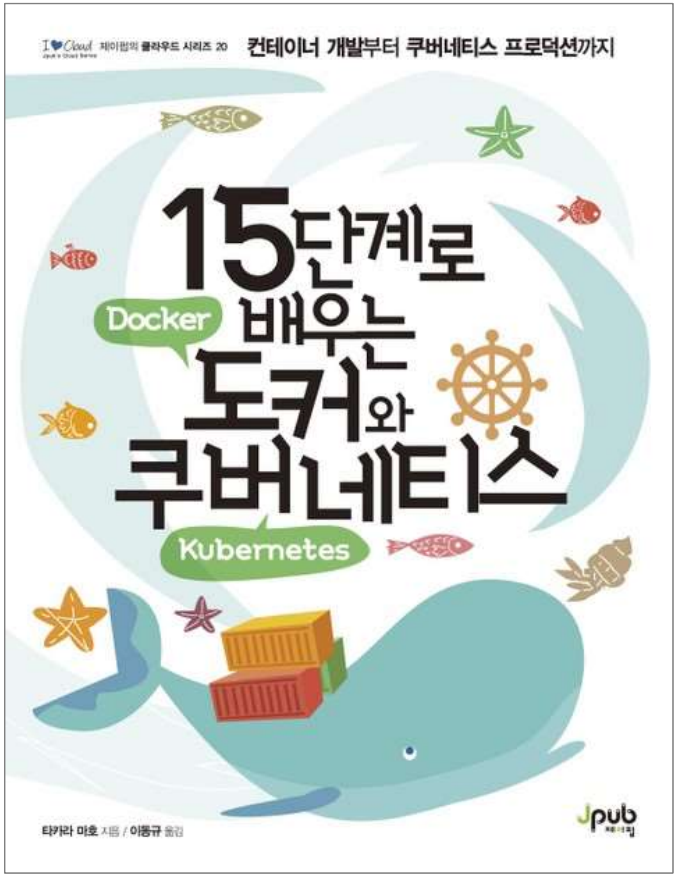
▷ Camera On은 필수, Mic On은 선택 !!!

▷ 질문을 창피해 하지 말아요. 같이 소통하면서 알찬 수업을 만들어요 !!!

▷ 공부도 중요하지만, 우리 서로의 인맥이 되어봐요 !!!

Books


Kubernetes
IN ACTION
그림과 상세한 설명을 통한
쿠버네티스 완벽 이해
MANNING


컨테이너 개발부터 쿠버네티스 프로덕션까지
15단계로
Docker 배우는
도커와
쿠버네티스
Kubernetes
Jpub

but ...

Our goal is not to be ~~**user**~~,

to be **administrator** of **K8s**

///

Study Environment

S/W Development Environment

It's Linux !!

///

Breaktime

///

Container

A Brief History of Containers: From the 1970s Till Now

1979: Unix V7 – chroot 도입

2000: FreeBSD Jails – 서비스와 고객 서비스를 구분하기 위해 여러 개의 독립적이고 작은 시스템(jails)으로 분할

2001: Linux VServer - Jails와 유사하게, 리소스(파일 시스템, 네트워크 주소, 메모리)를 분할 할 수 있는 운영 체제 가상화를 Linux 커널 패치로 구현

2004: Solaris Containers – 첫 번째 공개 베타 출시

2005: Open VZ (Open Virtuzzo) - 가상화, 격리, 리소스 관리 및 체크 포인트를 위해 패치 된 Linux 커널을 사용하는 Linux 용 운영 체제 수준의 가상화 기술

2006: Process Containers - 2006년 Google 출시. 리소스 사용량(CPU, Mem, Disk I/O, NW)을 제한, 계산 및 격리하도록 설계. 1년 후 "cgroups"으로 이름 변경.

2008: LXC (LinuX Containers) – 컨테이너 관리자의 가장 완벽한 최초 구현. cgroups & namespace를 사용하여 구현.

2011: Warden – CloudFoundry에서 초기는 LXC를 사용하고 나중에 자체 구현으로 대체. cgroups, namespace 및 프로세스 수명주기 관리 서비스 포함.

2013: LMCTFY (Let Me Contain That For You) – Linux 애플리케이션 컨테이너를 제공하는 Google 컨테이너 스택의 오픈 소스 버전. 2015년 중단.

2013: Docker - 컨테이너 인기 폭발. 초기 단계 LXC 사용, 추후 자체 라이브러리 libcontainer로 대체.

2014: Kubernetes (Google)

2015: Kubernetes to CNCF

2016: The Importance of Container Security Is Revealed – DevSecOps

2017: Container Tools Become Mature – 컨테이너 도구의 성숙

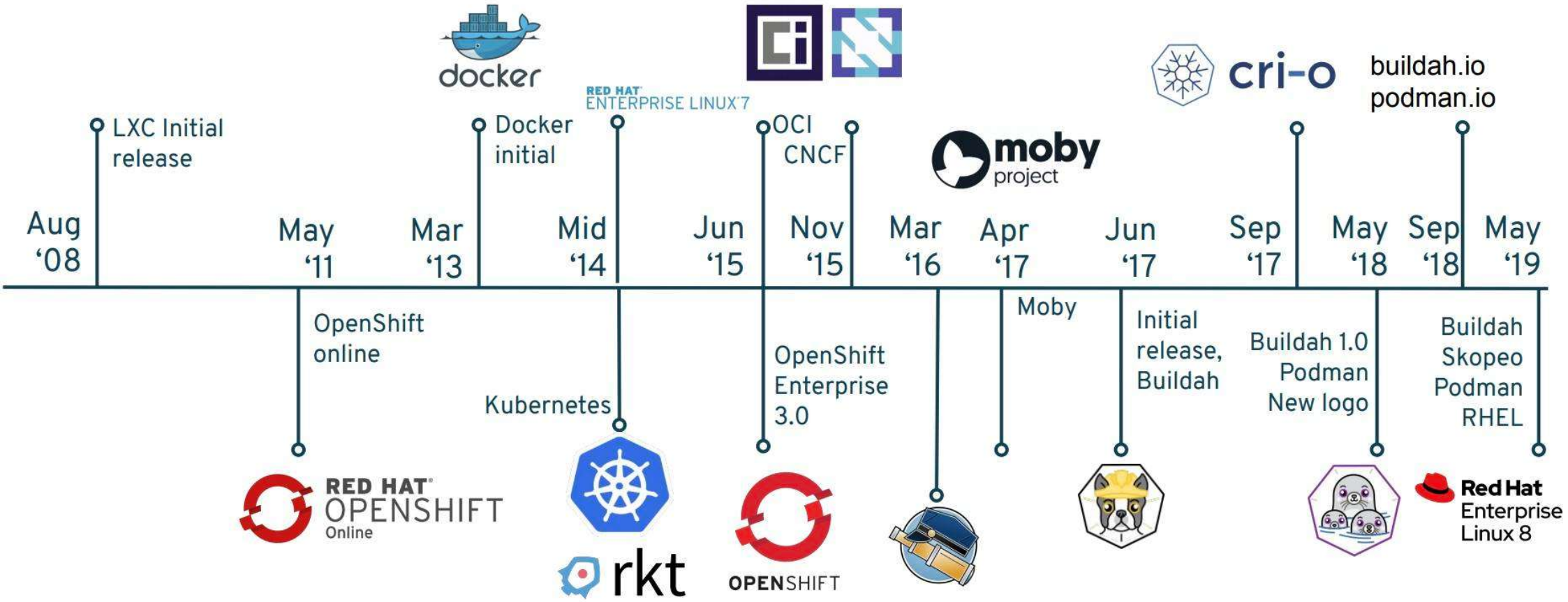
2017: containerd to CNCF (Docker)

2018: The Gold Standard – 시장 표준

2019: A Shifting Landscape - 변화

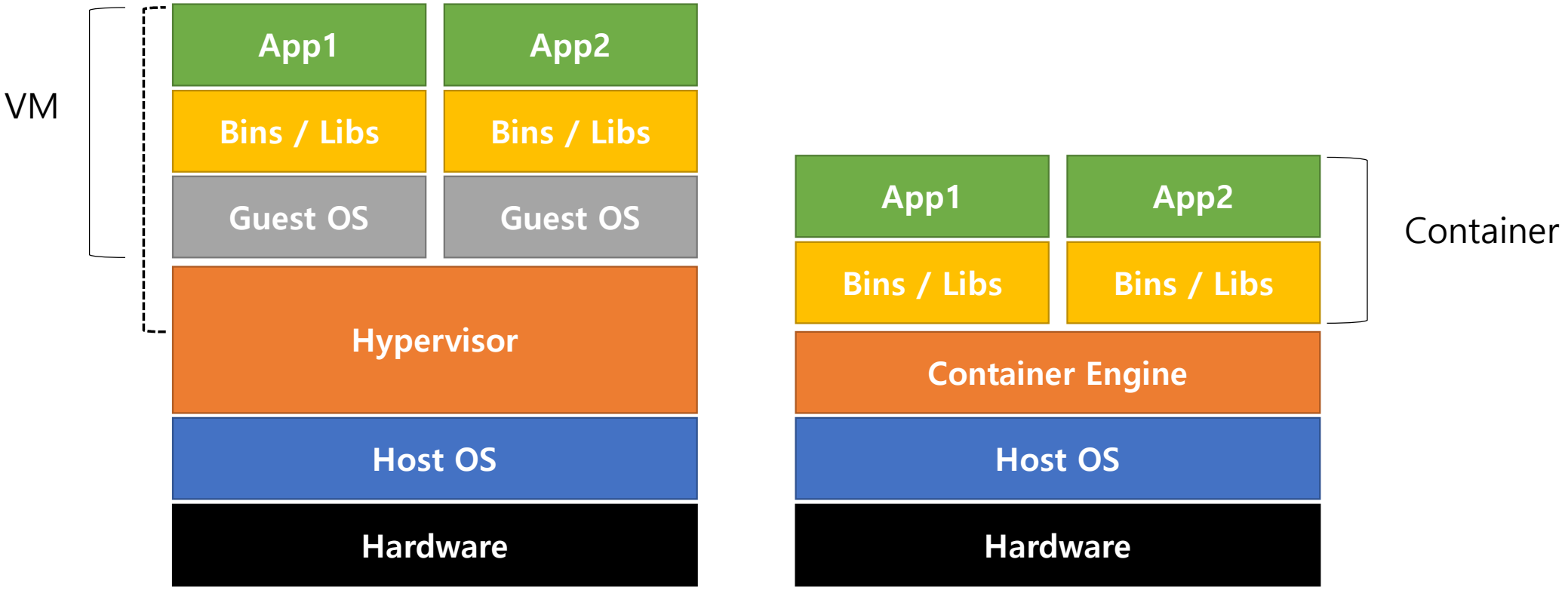
※ 참고 : https://blog.aquasec.com/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016

Evolution of the open-source container

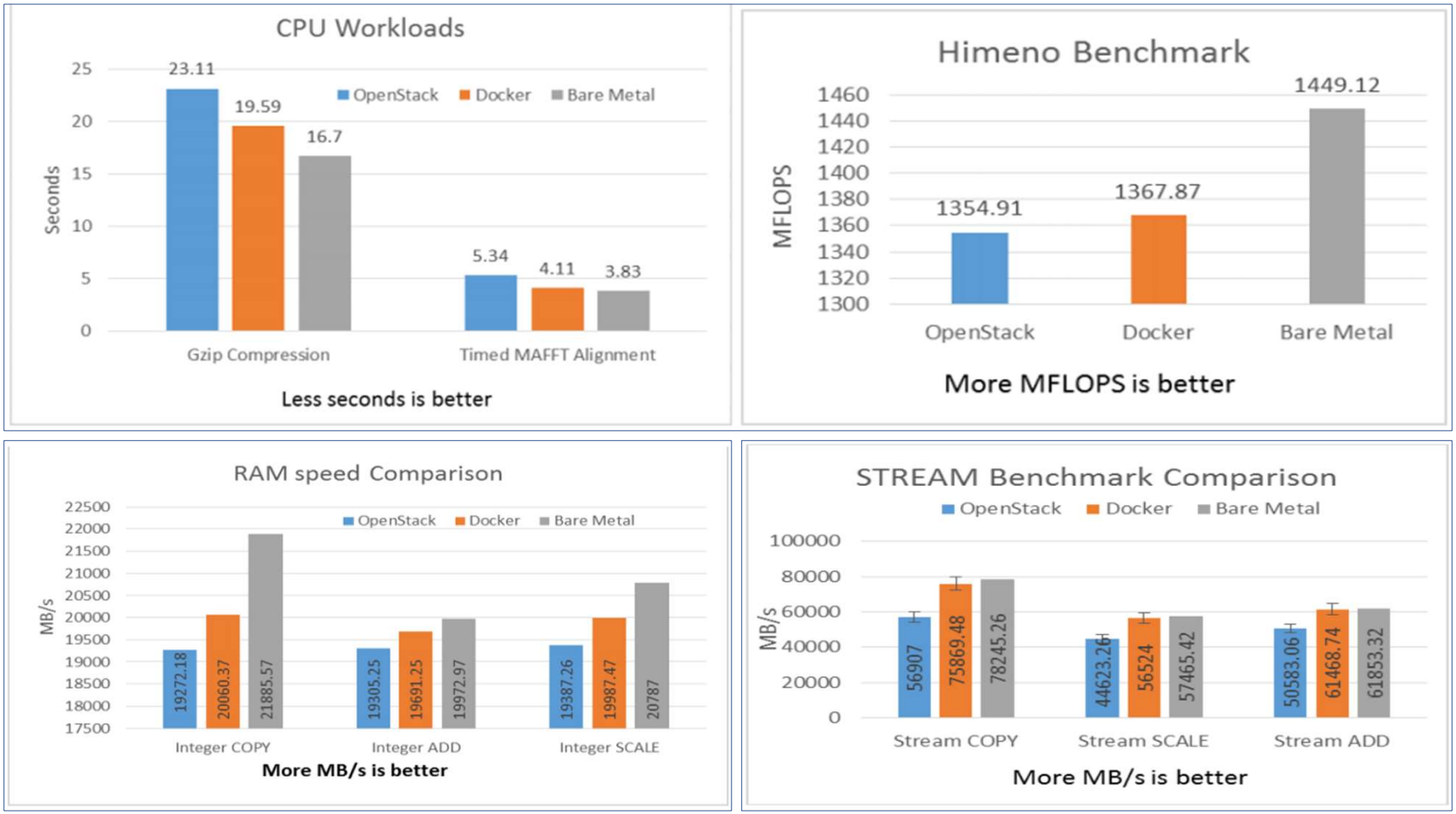


※ 참고 : https://developer.ibm.com/tutorials/multi-architecture-cri-o-container-images-for-red-hat-openshift/

Virtual Machine vs. Containers

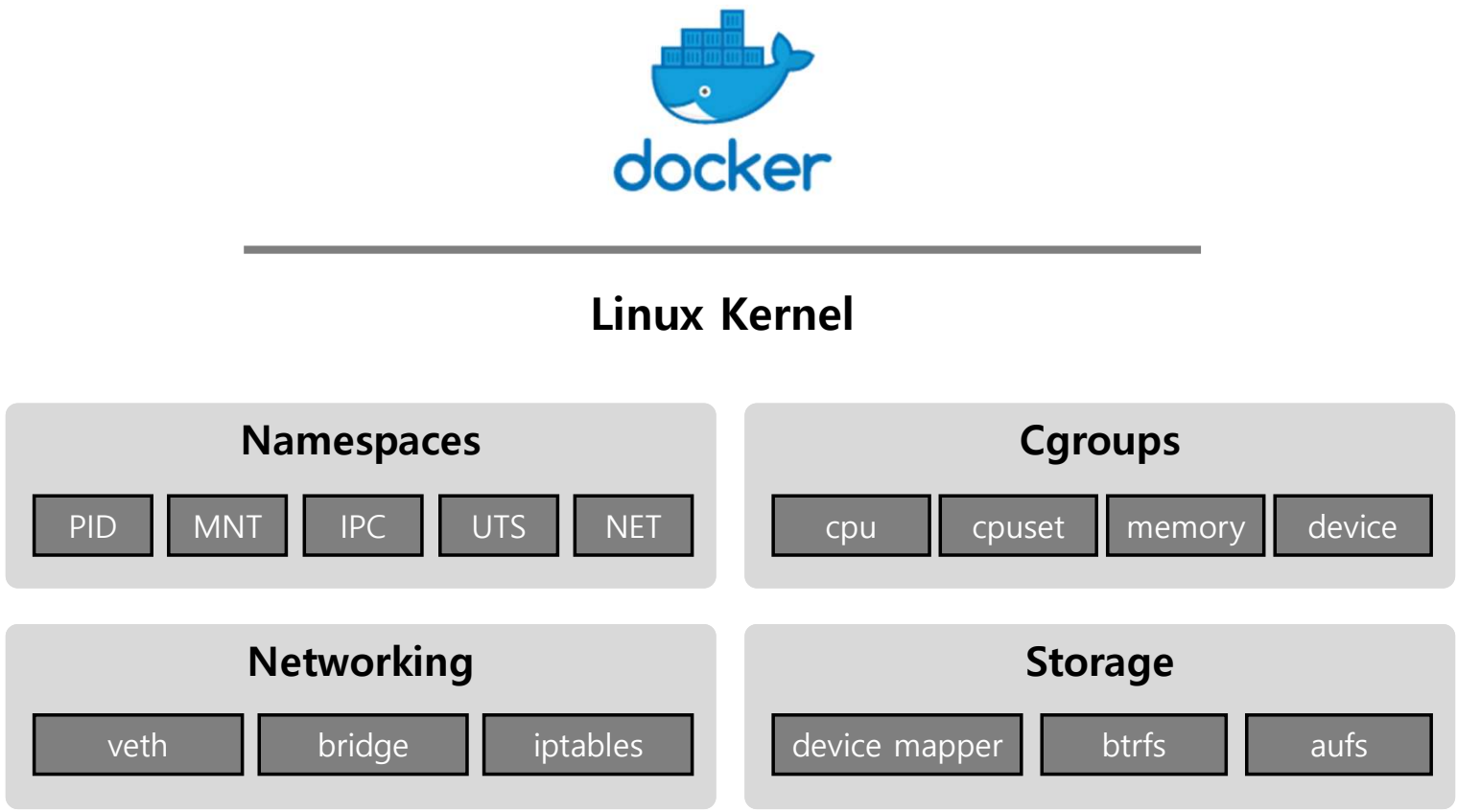


Docker Container vs. Openstack VM vs. Bare Metal Server

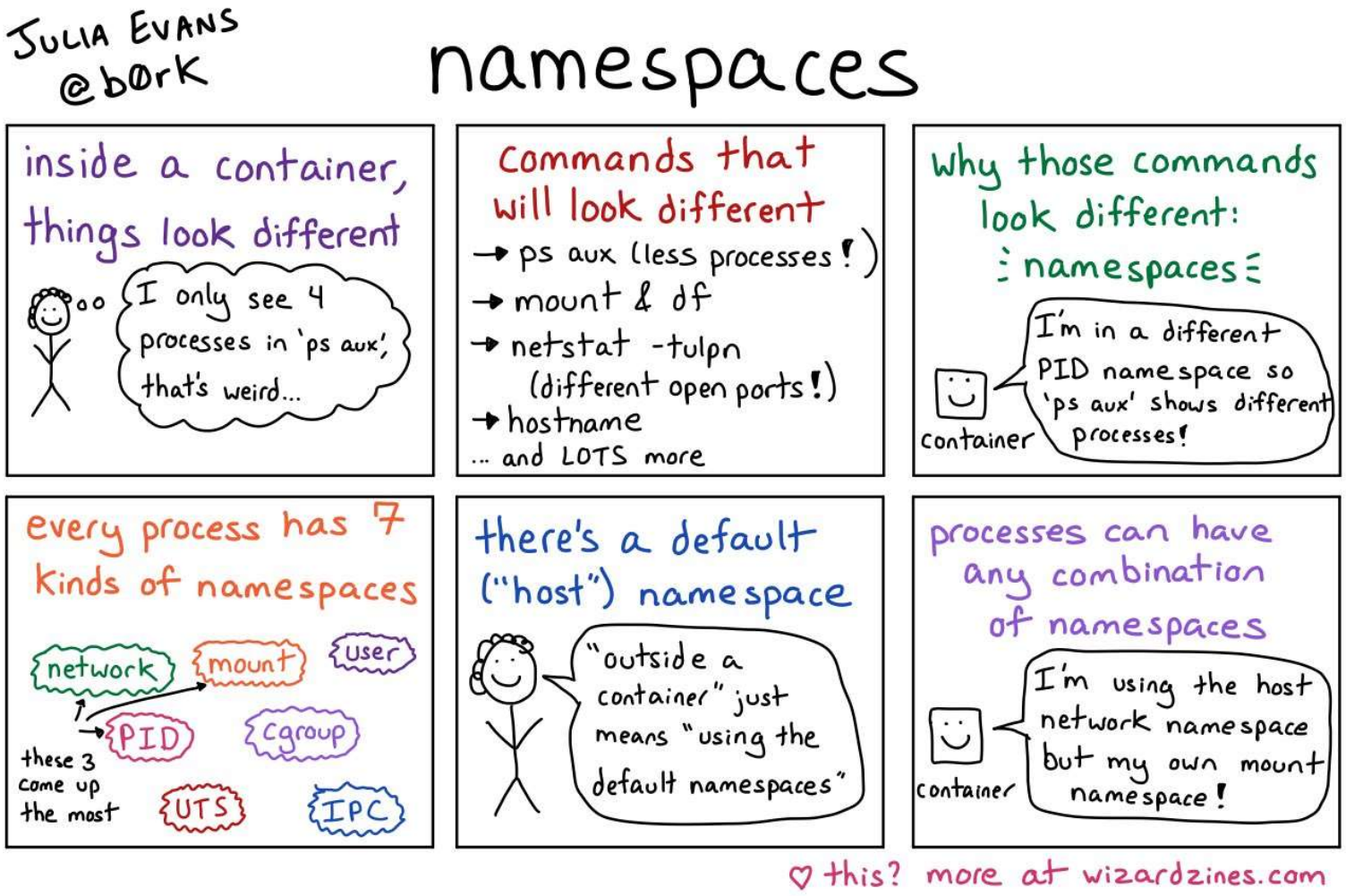


※ 참고 : http://ijeecs.iaescore.com/index.php/IJEECS/article/view/7925

Docker with Kernel



namespaces



※ 참고 : https://twitter.com/b0rk/status/1240364585766576128

namespaces

▷ a feature of the Linux kernel that **partitions kernel resources** such that one set of processes sees one set of resources while another set of processes sees a different set of resources.

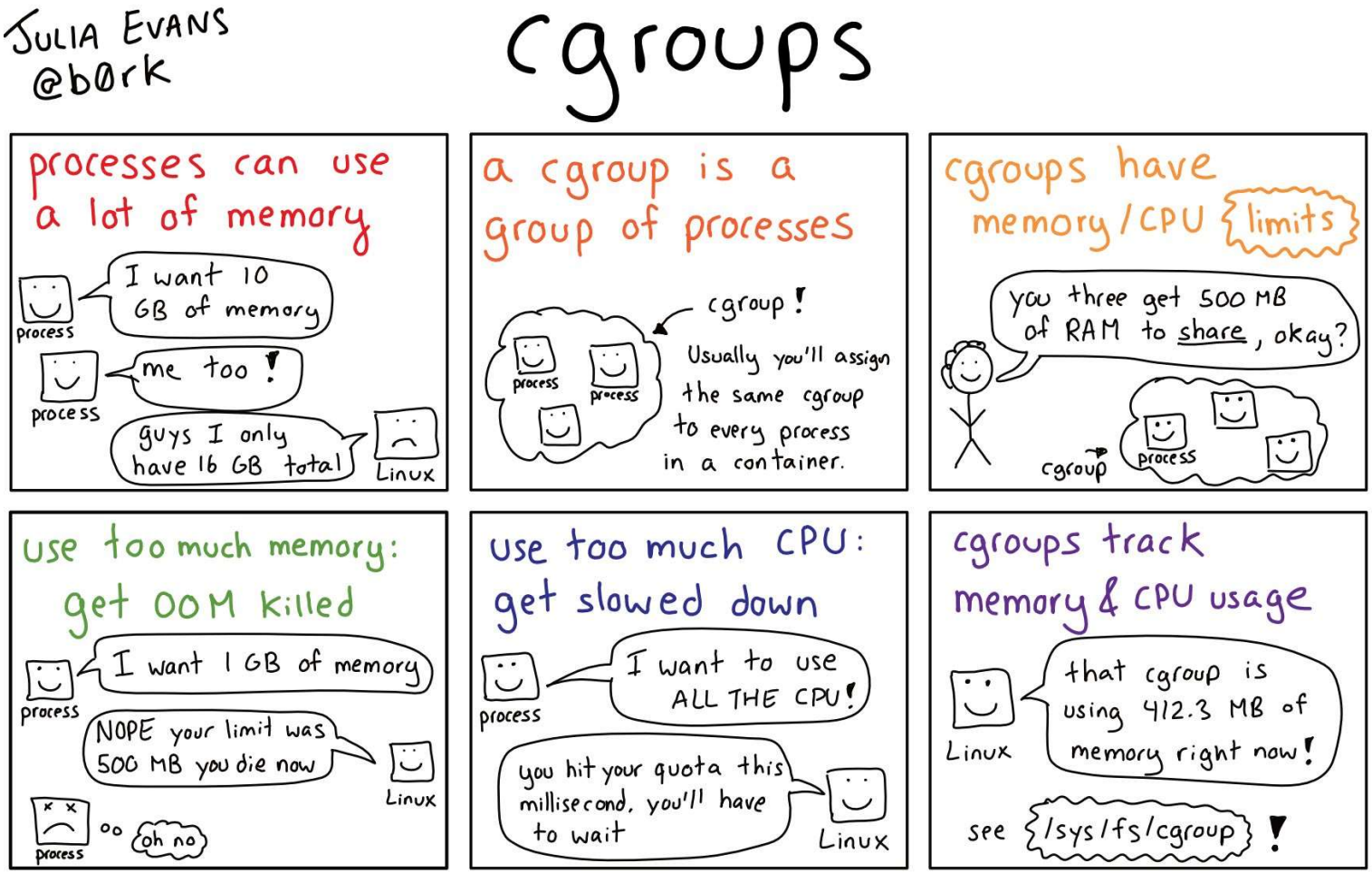
```
❯ sudo ls -al /proc/1/ns

합계 0
dr-x--x--x 2 root root 0 12월 24 23:32 .
dr-xr-xr-x 9 root root 0 12월 24 23:02 ..
lrwxrwxrwx 1 root root 0 12월 24 23:32 cgroup -> 'cgroup:[4026531835]'
lrwxrwxrwx 1 root root 0 12월 24 23:32 ipc -> 'ipc:[4026531839]'
lrwxrwxrwx 1 root root 0 12월 24 23:32 mnt -> 'mnt:[4026531840]'
lrwxrwxrwx 1 root root 0 12월 24 23:32 net -> 'net:[4026531992]'
lrwxrwxrwx 1 root root 0 12월 24 23:32 pid -> 'pid:[4026531836]'
lrwxrwxrwx 1 root root 0 12월 24 23:32 pid_for_children -> 'pid:[402
lrwxrwxrwx 1 root root 0 12월 24 23:32 user -> 'user:[4026531837]'
lrwxrwxrwx 1 root root 0 12월 24 23:32 uts -> 'uts:[4026531838]'
```

same namespaces

```
❯ sudo ls -al /proc/2/ns

합계 0
dr-x--x--x 2 root root 0 12월 24 23:36 .
dr-xr-xr-x 9 root root 0 12월 24 23:02 ..
lrwxrwxrwx 1 root root 0 12월 24 23:36 cgroup -> 'cgroup:[4026531835]'
lrwxrwxrwx 1 root root 0 12월 24 23:36 ipc -> 'ipc:[4026531839]'
lrwxrwxrwx 1 root root 0 12월 24 23:36 mnt -> 'mnt:[4026531840]'
lrwxrwxrwx 1 root root 0 12월 24 23:36 net -> 'net:[4026531992]'
lrwxrwxrwx 1 root root 0 12월 24 23:36 pid -> 'pid:[4026531836]'
lrwxrwxrwx 1 root root 0 12월 24 23:36 pid_for_children -> 'pid:[4026531836]'
lrwxrwxrwx 1 root root 0 12월 24 23:36 user -> 'user:[4026531837]'
lrwxrwxrwx 1 root root 0 12월 24 23:36 uts -> 'uts:[4026531838]'
```

※ 참고 : https://en.wikipedia.org/wiki/Linux_namespaces

cgroups (control groups)



※ 참고 : https://twitter.com/b0rk/status/1214341831049252870

cgroups (control groups)

▷ Linux kernel feature that **limits**, accounts for, and isolates the resource usage (**CPU, memory, disk I/O, network**, etc.) of a **collection of processes**.

```
❯ ls -al /sys/fs/cgroup
합계 0
drwxr-xr-x 15 root root 380  7월 15 00:32 .
drwxr-xr-x  9 root root   0  7월 15 00:32 ..
dr-xr-xr-x  5 root root   0  7월 15 00:32 blkio
lrwxrwxrwx  1 root root  11  7월 15 00:32 cpu -> cpu,cpuacct
dr-xr-xr-x  5 root root   0  7월 15 00:32 cpu,cpuacct
lrwxrwxrwx  1 root root  11  7월 15 00:32 cpuacct -> cpu,cpuacct
dr-xr-xr-x  3 root root   0  7월 15 00:32 cpuset
dr-xr-xr-x  5 root root   0  7월 15 00:32 devices
dr-xr-xr-x  3 root root   0  7월 15 00:32 freezer
dr-xr-xr-x  3 root root   0  7월 15 00:32 hugetlb
dr-xr-xr-x  5 root root   0  7월 15 00:32 memory
lrwxrwxrwx  1 root root  16  7월 15 00:32 net_cls -> net_cls,net_prio
dr-xr-xr-x  3 root root   0  7월 15 00:32 net_cls,net_prio
lrwxrwxrwx  1 root root  16  7월 15 00:32 net_prio -> net_cls,net_prio
dr-xr-xr-x  3 root root   0  7월 15 00:32 perf_event
dr-xr-xr-x  5 root root   0  7월 15 00:32 pids
dr-xr-xr-x  3 root root   0  7월 15 00:32 rdma
dr-xr-xr-x  6 root root   0  7월 15 00:32 systemd
dr-xr-xr-x  6 root root   0  7월 15 00:32 unified
```

```
❯ cat /proc/cgroups

#subsys_name    hierarchy   num_cgroups   enabled
cpuset          10          1             1
cpu             5           65            1
cpuacct         5           65            1
blkio           8           65            1
memory          12          99            1
devices         2           65            1
freezer         9           1             1
net_cls         7           1             1
perf_event      6           1             1
net_prio        7           1             1
hugetlb         3           1             1
pids            11          68            1
rdma            4           1             1
```

※ 참고 : https://en.wikipedia.org/wiki/Cgroups

Container Network Model (CNM)

▷ Sandbox

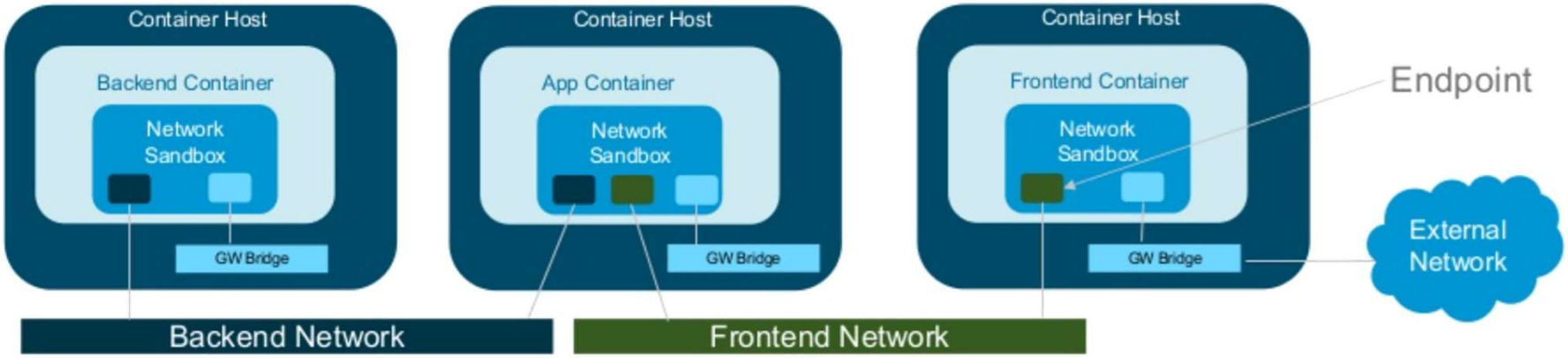
- A Sandbox contains the configuration of a container's network stack.
- This includes management of the container's interfaces, routing table and DNS settings.
- An implementation of a Sandbox could be a Linux Network Namespace, a FreeBSD Jail or other similar concept.

▷ Endpoint

- An Endpoint joins a Sandbox to a Network.
- An implementation of an Endpoint could be a veth pair, an Open vSwitch internal port or similar

▷ Network

- A Network is a group of Endpoints that are able to communicate with each-other directly.
- An implementation of a Network could be a VXLAN Segment, a Linux bridge, a VLAN, etc.

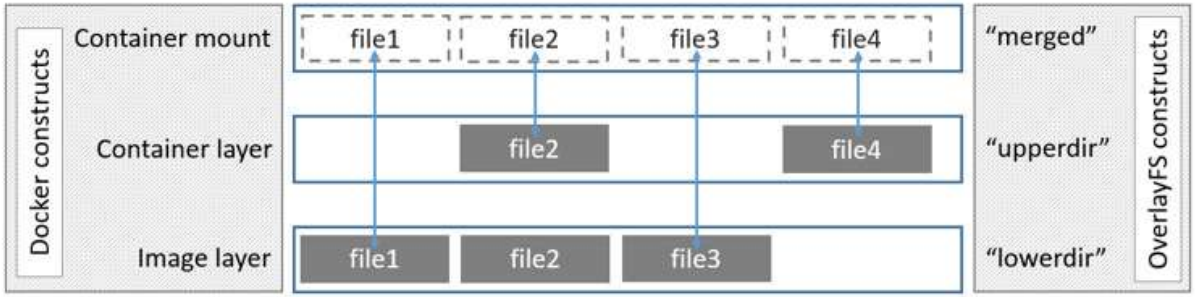


※ 참고 : https://www.slideshare.net/OpenNetworkingSummit/container-networking-deep-dive

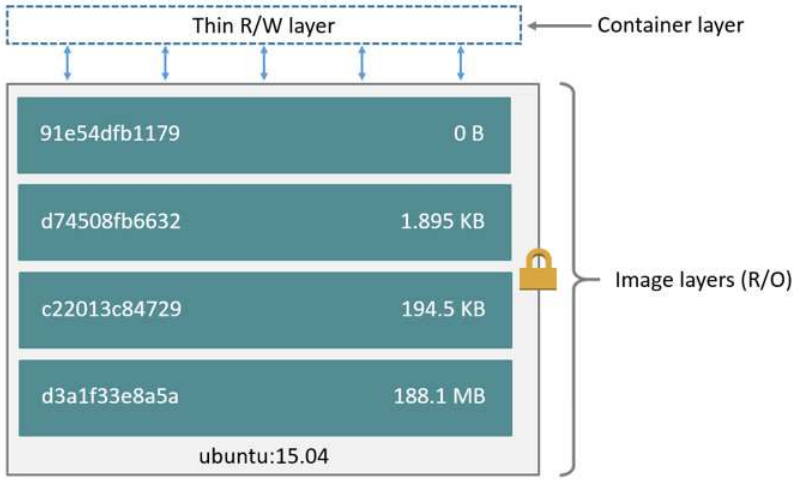
Docker Storage Driver

▷ Docker supports the following storage drivers

- **overlay2** : 기본 드라이버
- **aufs** : Docker 18.06 및 이전 버전에서 사용
- **fuse-overlayfs** : Rootless 지원 안되는 호스트에서 Rootless Docker를 사용할 때
- **devicemapper** : production 환경을 위해서는 direct-lvm 필요.
- **btrfs** and **zfs** : "snapshots" 같은 고급 기능을 지원하지만 설치와 유지보수가 까다로움.
- **vfs** : 테스트 목적으로만 사용하는 것을 권장



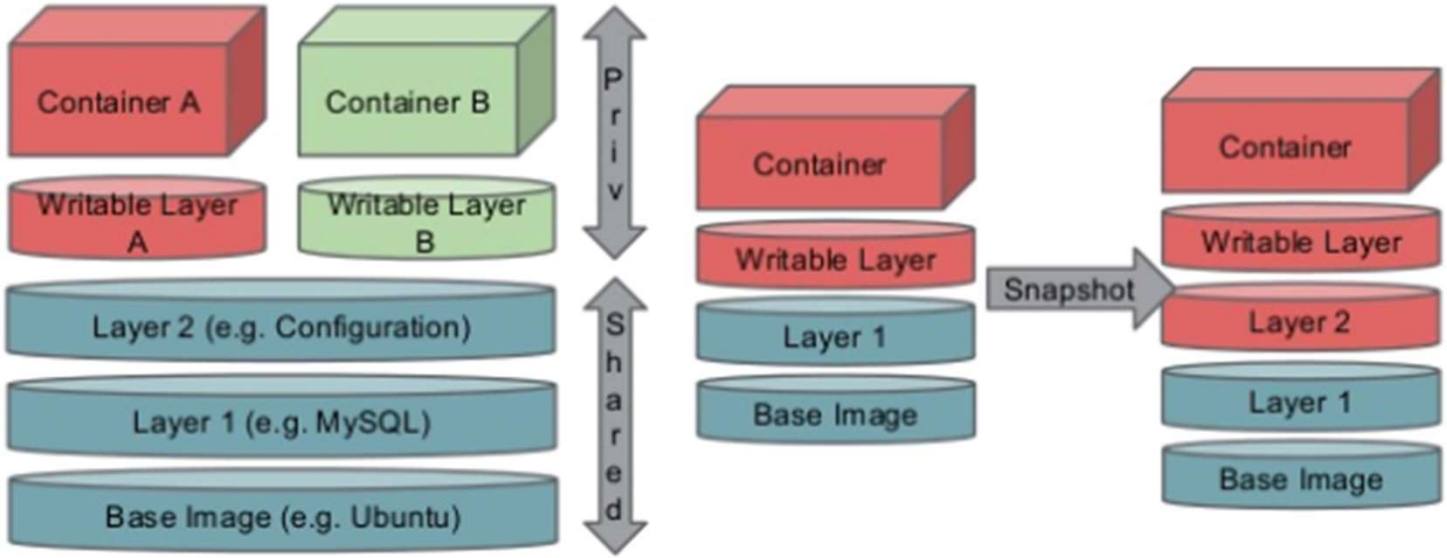
How the overlay driver works



Container
(based on ubuntu:15.04 image)

※ 참고 : https://docs.docker.com/storage/storagedriver/overlayfs-driver/

Docker Storage Driver

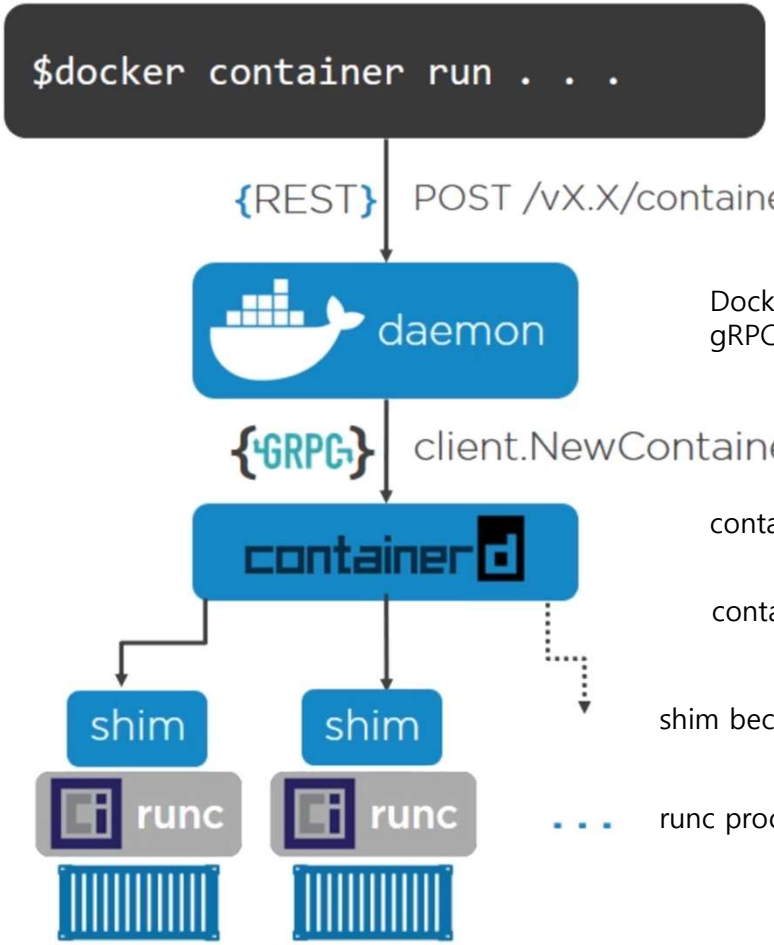


※ 참고 : https://docs.docker.com/storage/storagedriver/overlayfs-driver/

///

Docker

Docker Engine Architecture



Docker CLI client makes a REST API call to the Docker daemon server

Docker responds to the request and spins up a container by passing the message through gRPC protocol to the containerd process

containerd manages images and lifecycle of the containers: stop, pause, start, delete

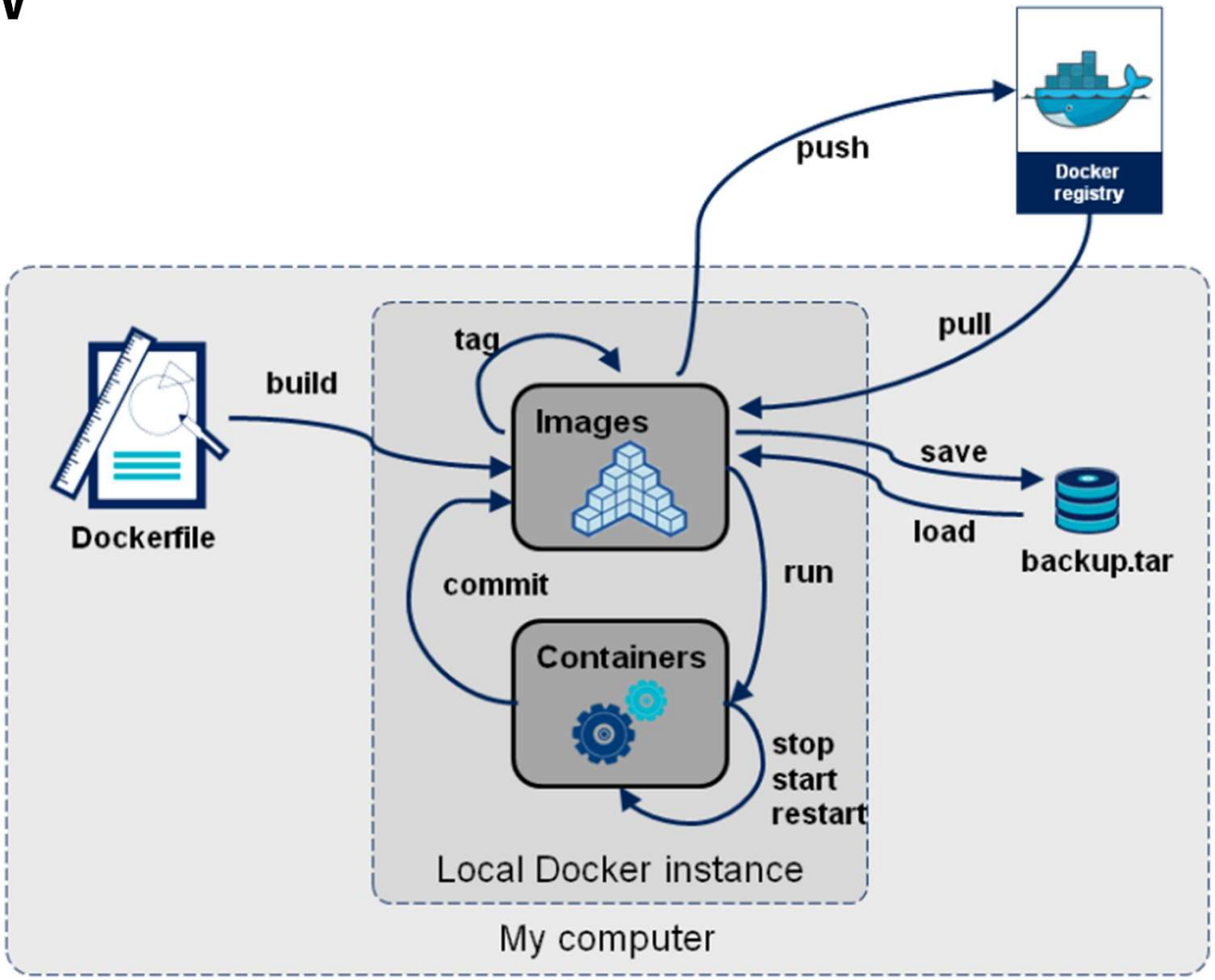
containerd forks the instance of runc for new containers

shim becomes the parent of the running container

runc process exits after creating container

※ 참고 : https://betterprogramming.pub/docker-for-front-end-developers-c758a44e622f

Docker flow



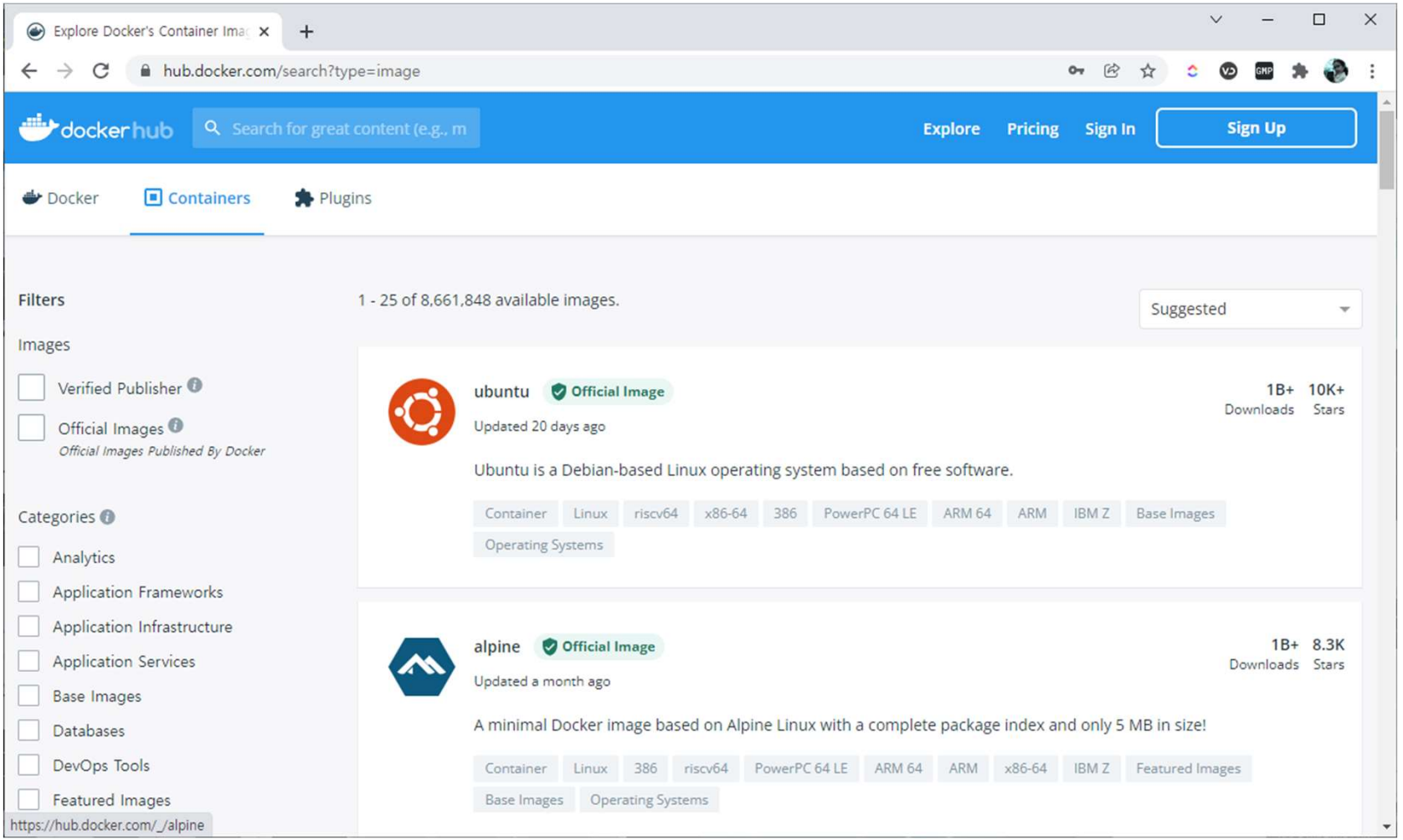
※ 참고 : https://blog.wonizz.tk/2019/07/31/docker-dockerfile/

Dockerfile

```
# fetch node v4 LTS codename argon
FROM node:argon

# Request samplename build argument
ARG samplename

# Create app directory
RUN mkdir -p /usr/src/spfx-samples
WORKDIR /usr/src/spfx-samples

#Install app dependencies
RUN git clone https://github.com/SharePoint/sp-dev-fx-webparts.git .
WORKDIR /usr/src/spfx-samples/samples/$samplename

# install gulp on a global scope
RUN npm install gulp -g

# RUN ["npm", "install", "gulp"]
RUN npm install
RUN npm cache clean

# Expose required ports
EXPOSE 4321 35729 5432

# Run sample
CMD ["gulp", "serve"]

```

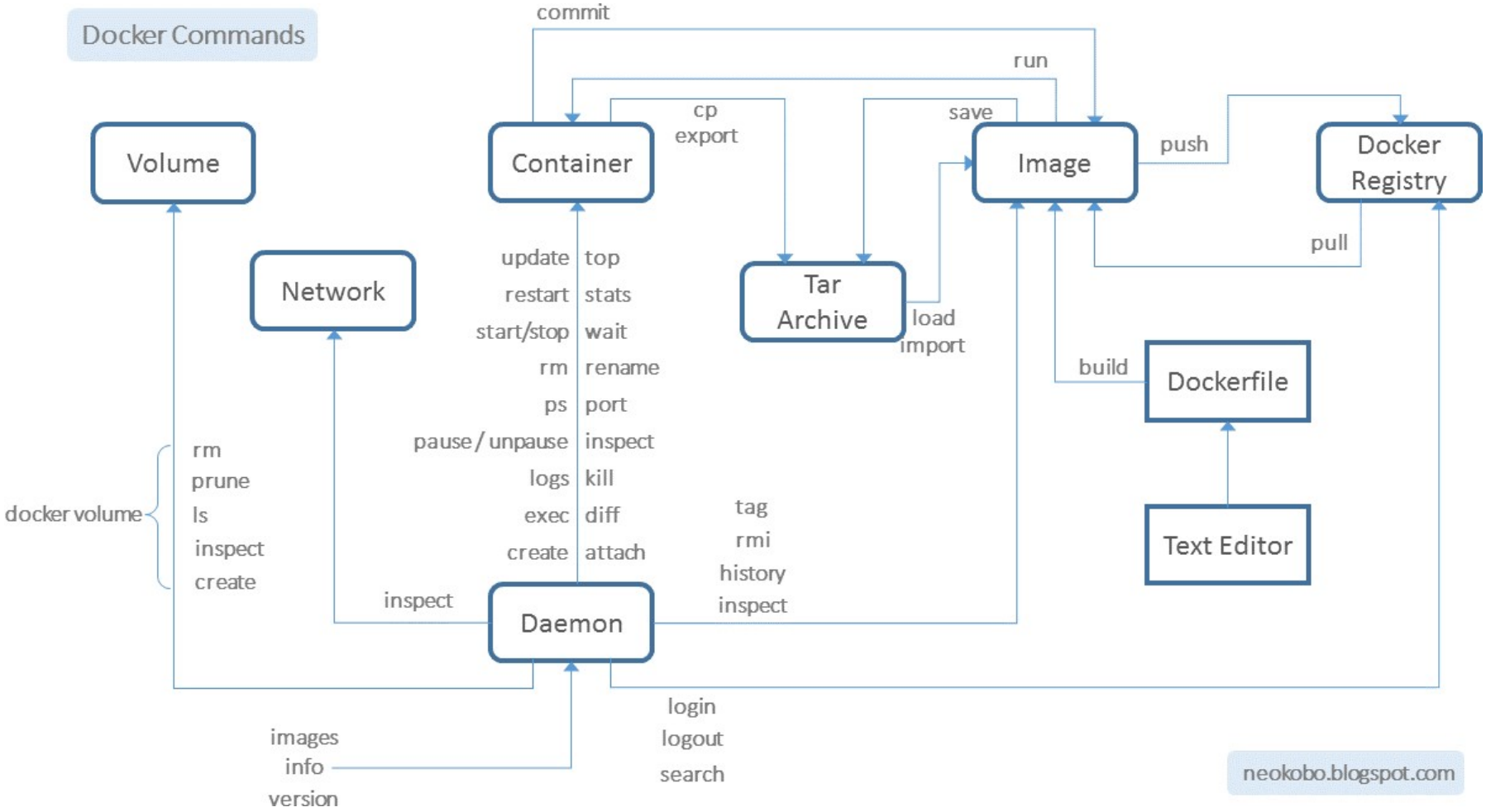
※ 참고 : https://n8d.at/how-to-run-sharepoint-pattern-and-practices-samples-through-docker/

Docker Hub (Registry)



※ 참고 : https://hub.docker.com/

Docker command



※ 참고 : http://neokobo.blogspot.com/2017/12/docker-command-flowchart.html

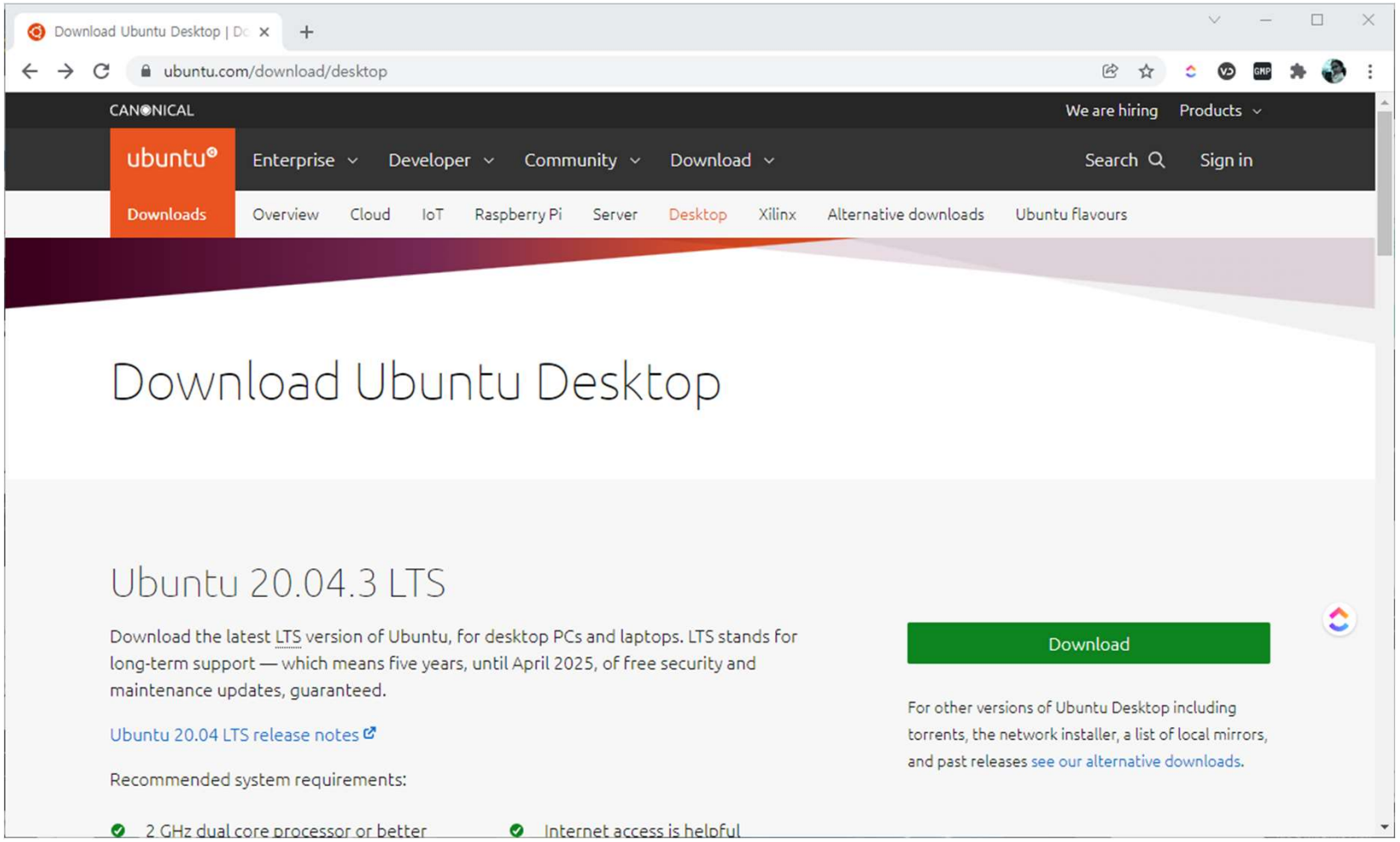
///

Docker
Hands-On

VirtualBox Install

※ 참고 : https://www.virtualbox.org/

Ubuntu Install



※ 참고 : https://ubuntu.com/download/desktop

Docker Install

Ubuntu 배포판 확인

```
❯ lsb_release -a

No LSB modules are available.
Distributor ID:         Ubuntu
Description:            Ubuntu 18.04.6 LTS
Release:    18.04
Codename:   bionic
```

패키지 및 버전 확인

https://download.docker.com/linux/ubuntu/dists/

```
❯ wget https://download.docker.com/linux/ubuntu/dists/bionic/pool/stable/amd64/containerd.io_1.4.12-1_amd64.deb
❯ wget https://download.docker.com/linux/ubuntu/dists/bionic/pool/stable/amd64/docker-ce-cli_20.10.12~3-0~ubuntu-bionic_amd64.deb
❯ wget https://download.docker.com/linux/ubuntu/dists/bionic/pool/stable/amd64/docker-ce_20.10.12~3-0~ubuntu-bionic_amd64.deb

❯ sudo dpkg --install ./containerd.io_1.4.12-1_amd64.deb
❯ sudo dpkg --install ./docker-ce-cli_20.10.12~3-0~ubuntu-bionic_amd64.deb
❯ sudo dpkg --install ./docker-ce_20.10.12~3-0~ubuntu-bionic_amd64.deb

❯ sudo usermod -aG docker $USER

❯ docker --version
Docker version 20.10.12, build e91ed57

❯ docker run hello-world
```

Just do - docker build

index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Kubernetes</title>
</head>
<body>
  <h2>Hello from Nginx container</h2>
</body>
</html>
```

Dockerfile

```
FROM nginx:latest

COPY ./index.html /usr/share/nginx/html/index.html
```

예제 파일 내려 받은 후 docker image 빌드

```
❯ git clone https://github.com/whatwant-school/kubernetes.git
❯ cd kubernetes/01-Container-Docker/hands-on

❯ docker build -t webserver .
...

❯ docker images

REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
webserver     latest    6223db426adf   42 seconds ago   141MB
nginx         latest    f6987c8d6ed5   3 days ago       141MB
hello-world   latest    feb5d9fea6a5   3 months ago     13.3kB
```

※ 참고 : https://www.docker.com/blog/how-to-use-the-official-nginx-docker-image/

Just do - docker run / ps

빌드한 이미지를 (container로) 실행하자

```
❯ docker run -it --rm -d -p 8080:80 --name web webserver

4ade5015b7f84e3f115331072a18038819e39dbf65c54cd7413d153900b93264


❯ docker ps -al

CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS          PORTS                                   NAMES
4ade5015b7f8   webserver   "/docker-entrypoint.…"   42 seconds ago   Up 41 seconds   0.0.0.0:8080->80/tcp, :::8080->80/tcp   web
```

Chrome을 통해서 웹페이지 확인

Just do - docker stop / images / rmi

동작하고 있는 container를 중단해보자

```
❯ docker ps -al

CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS          PORTS                                   NAMES
4ade5015b7f8   webserver   "/docker-entrypoint.…"   42 seconds ago   Up 41 seconds   0.0.0.0:8080->80/tcp, :::8080->80/tcp   web


❯ docker stop 4ade5015b7f8

4ade5015b7f8
```

등록되어 있는 image를 확인하고, 삭제해보자

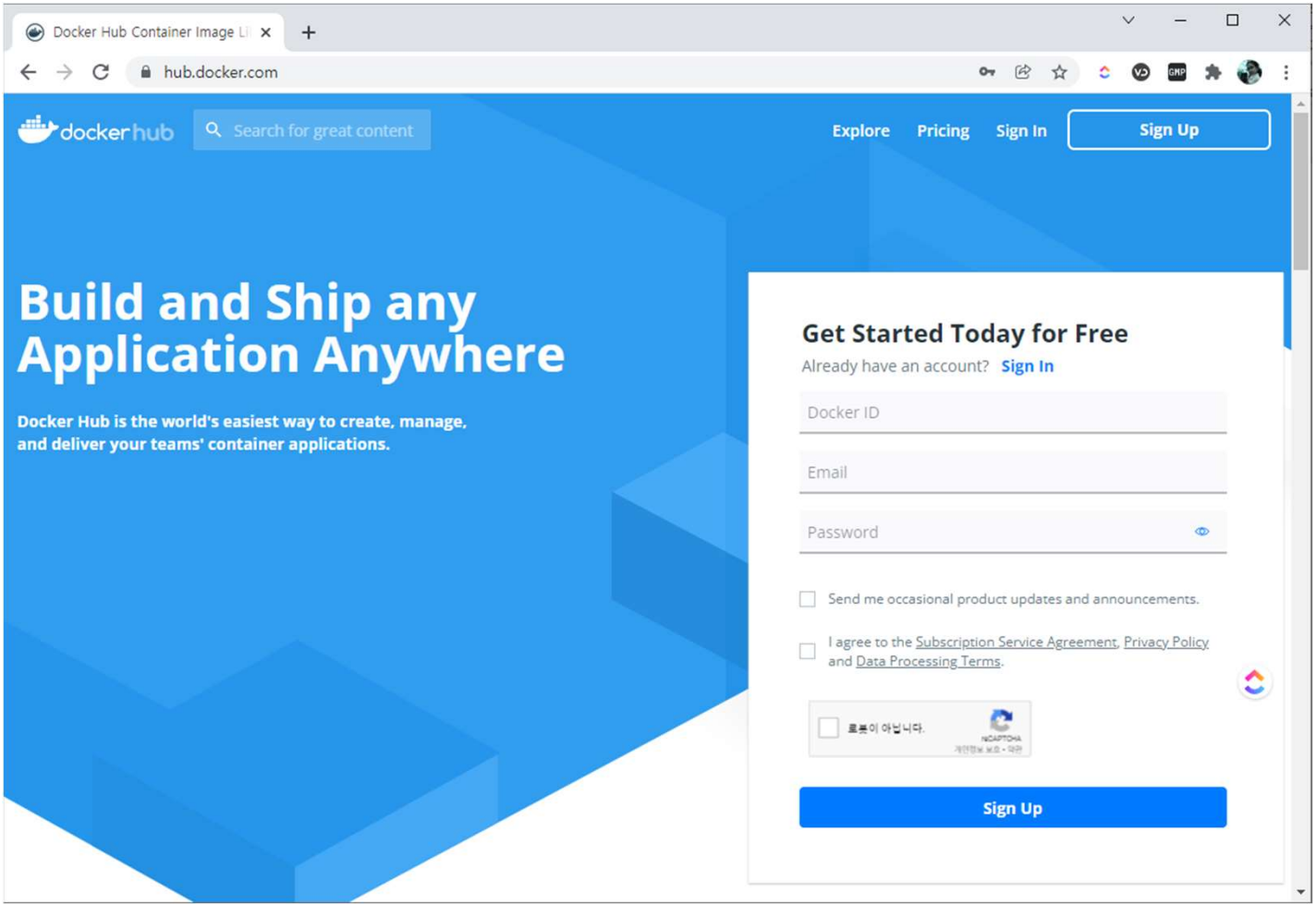
```
❯ docker images

REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
webserver    latest    6223db426adf   7 hours ago   141MB
nginx        latest    f6987c8d6ed5   4 days ago    141MB


❯ docker rmi webserver

Untagged: webserver:latest
Deleted: sha256:6223db426adf9a43a506b324924f450b6c466ea2ee60cf8cc165d923d9806f3c
Deleted: sha256:572dc68dceacf746d58efe16951ec47936d46d68e2d27e089a05f8cd79738895
```

///

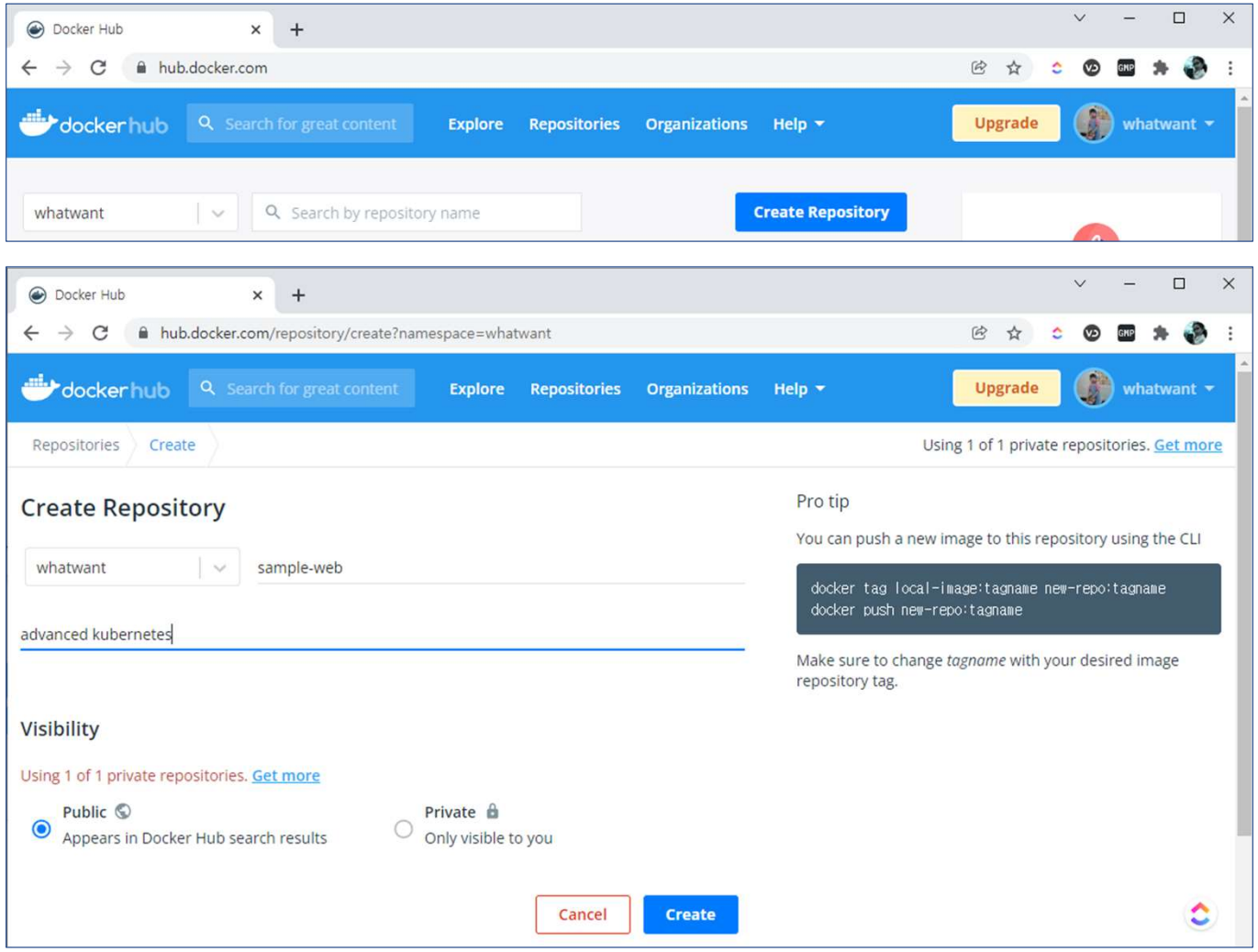
Tip #1 - DockerHub

DockerHub - Sign Up / In

https://hub.docker.com/



DockerHub - Create Repository



DockerHub - docker login / tag / push

DockerHub 권한을 위해 로그인이 필요하다

```
❯ docker login

...
Login Succeeded
```

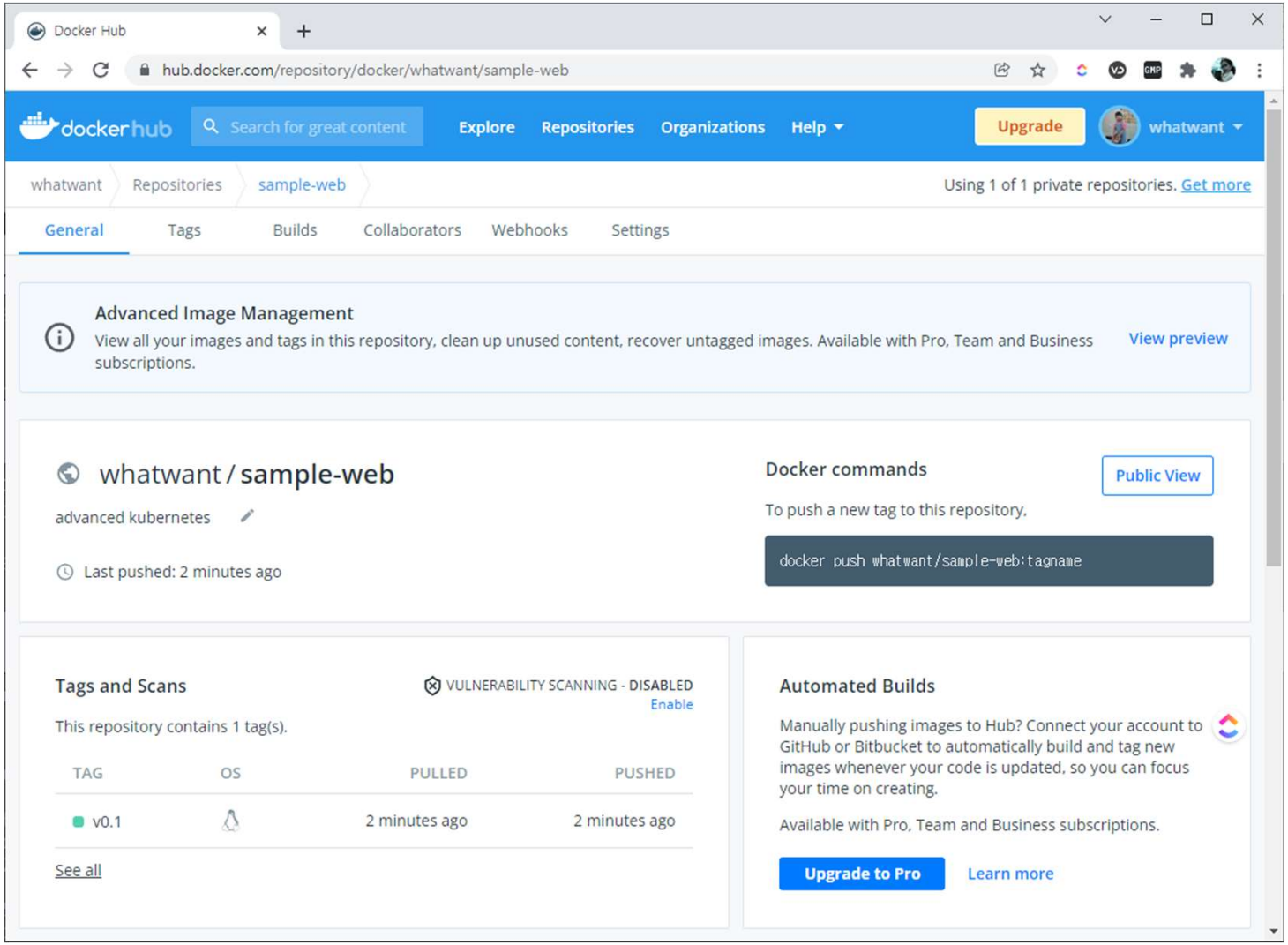
앞에서 진행해봤던 이미지를 재활용 해보자

```
❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
❯ cd advanced-kubernetes/01-week/

❯ docker build -t webserver .
...
```

업로드 하기 전에 tagging을 하고 push

```
❯ docker tag webserver:latest whatwant/sample-web:v0.1


❯ docker push whatwant/sample-web:v0.1

The push refers to repository [docker.io/whatwant/sample-web]
ba032a7dca37: Pushed
51a4ac025eb4: Mounted from library/nginx
...
2edcec3590a4: Mounted from library/nginx
v0.1: digest: sha256:f47f5ecb4f828d28f930a9c262f33066c5ca59e6b3f72c2ac882c71e3e981e31 size: 1777docker build -t webserver .
```

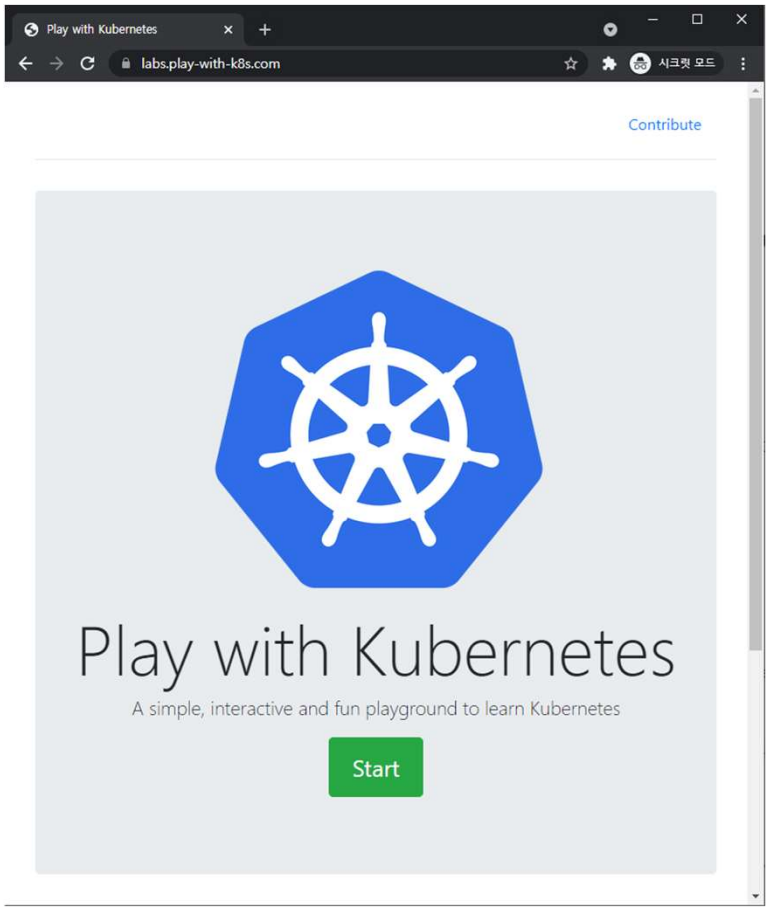
DockerHub - Repository

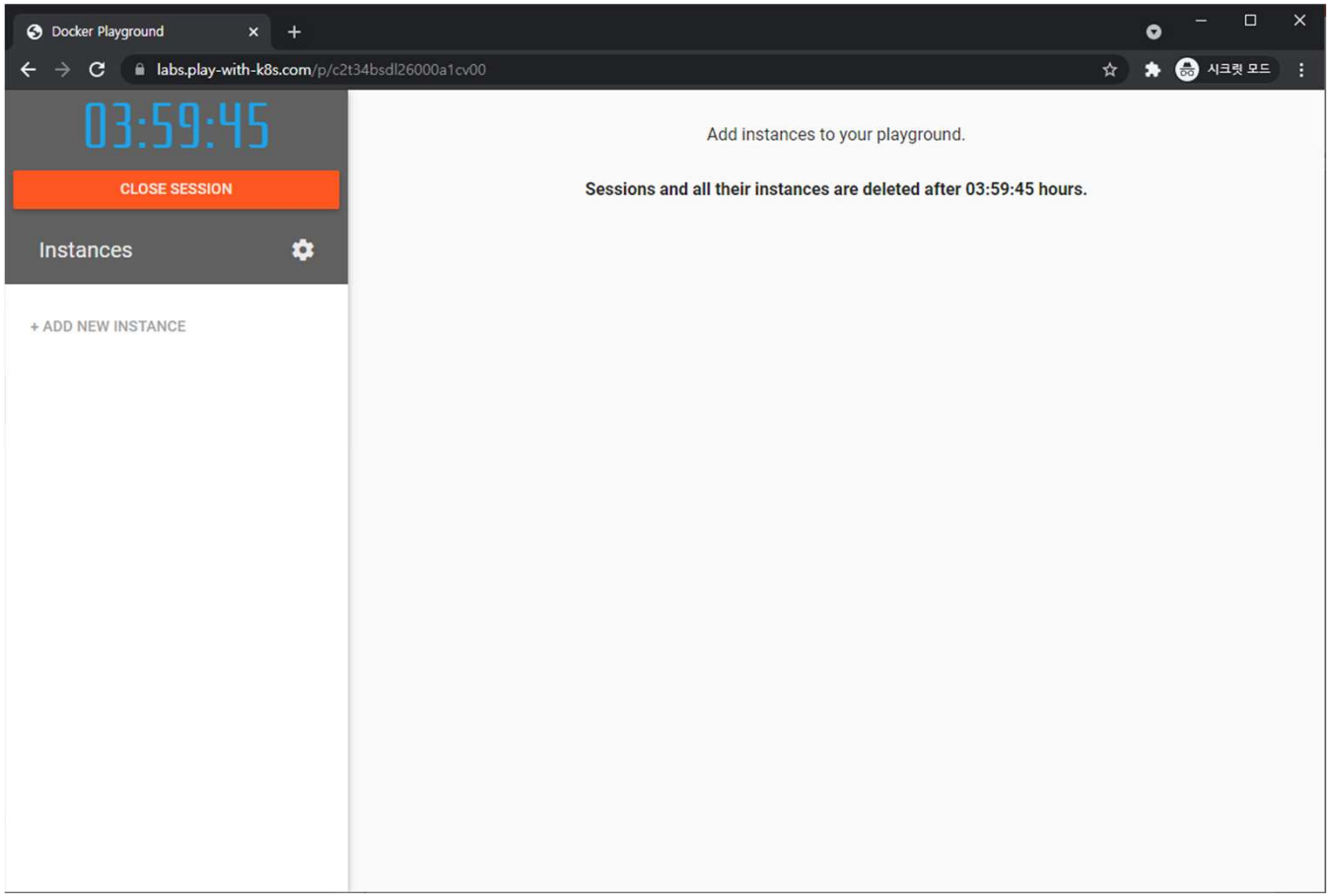


Tip #2 - Play with Kubernetes

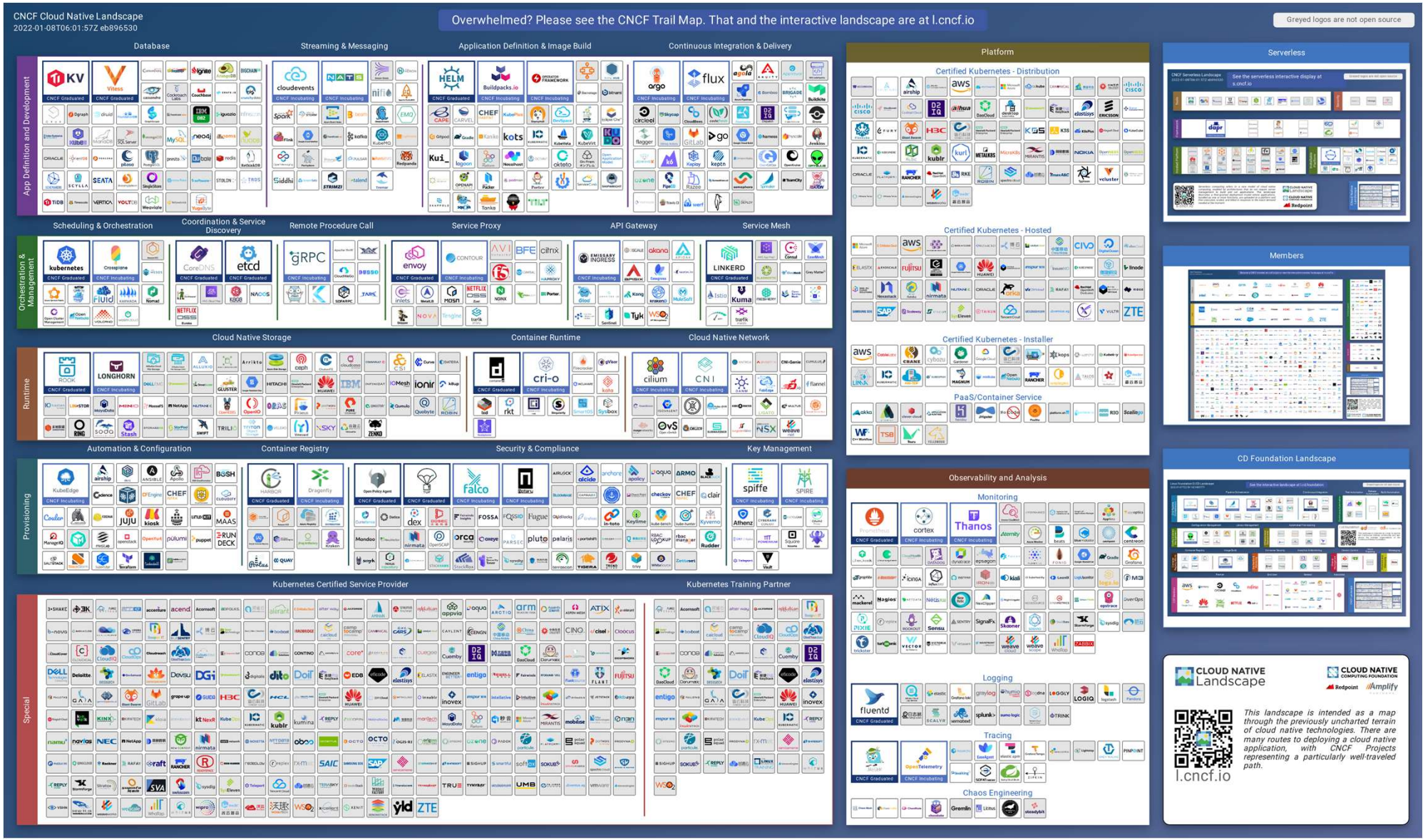
실습 환경 : https://labs.play-with-k8s.com/

- 여러 개의 instance 생성 가능, but 4시간 무료 사용
- disk 관련된 제약 等 불편함은 존재





Tip #3 - CNCF



※ 참고 : https://landscape.cncf.io/images/landscape.png

///

https://kahoot.it/