1st Week

Agenda

- ▷ [30m] Ice-Breaking
- ▷ [10m] Ground-Rule
- ▷ [10m] Prerequisite
- ▷ [30m] Container / Docker Overview
- ▷ [20m] Docker Hands-On
- ▷ [10m] Breaktime
- ▷ [50m] Kubernetes Overview
- ▷ [20m] Quiz

Ice-Breaking

▷ 자기 소개

▷ 설문조사 결과 공유

▷ 인증샷

Ground-Rule

▷ 가급적 지각/결석 하지 않기

▷ 가급적 Camera 켜 놓고 수업 참여하기

▷ 가급적 적극적으로 참여하기

▷ 3시간이 넘더라도 배고프다고 화내지 않기

▷ Slack 잊지 않기

▷ 꼭 끝까지 함께하기

Prerequisite

▷ 자료 공유

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

▷ Desktop/Laptop

- OS: Windows 10 (VirtualBox 설치 가능 환경)
- Mem: 8GB 이상 (16GB 추천)
- Network: 공유기

///

Breaktime

///

Container / Docker Overview

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

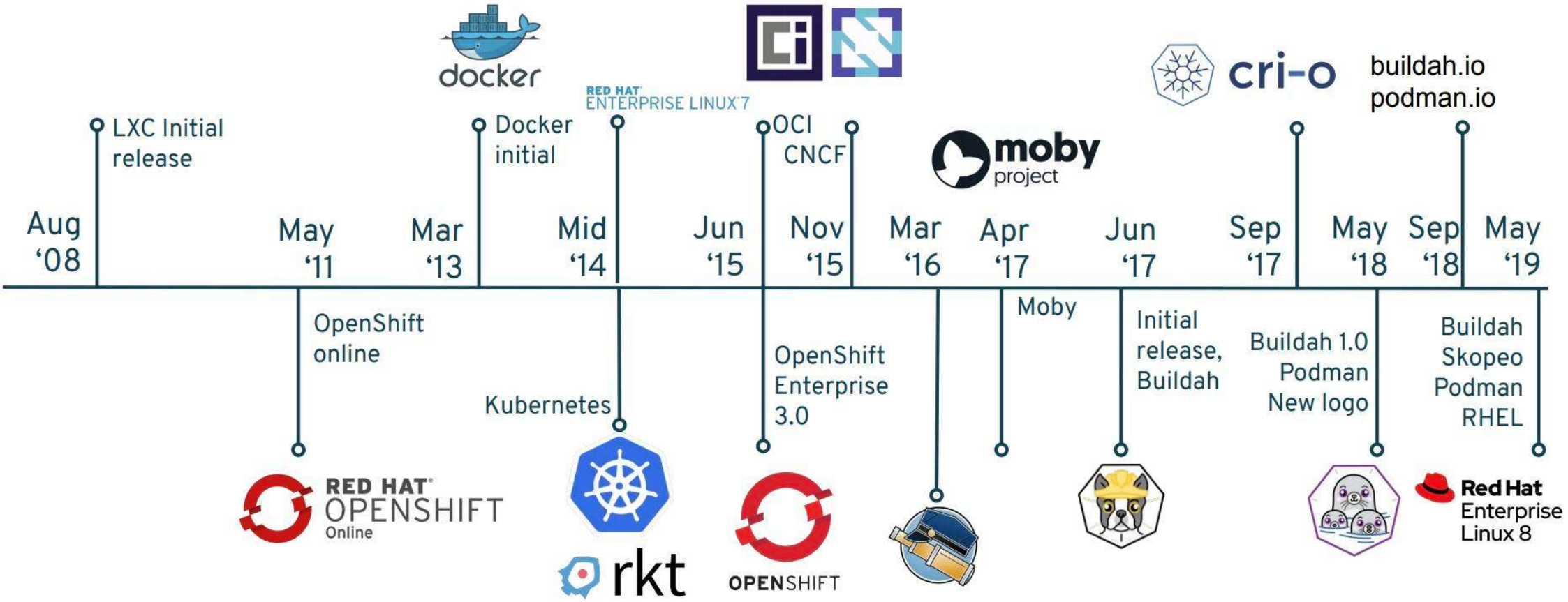
2017: containerd to CNCF (Docker)

2018: The Gold Standard – 시장 표준

2019: A Shifting Landscape - 변화

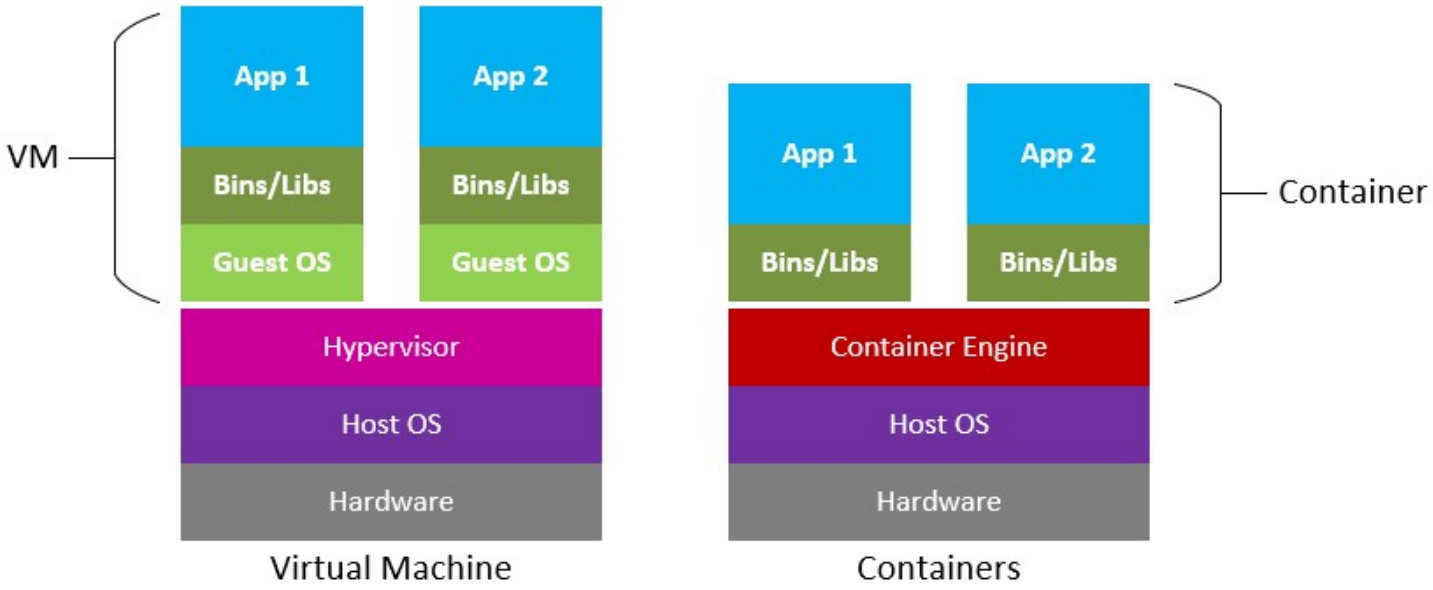
※ 참고 : https://blog.aquasec.com/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016

Evolution of the open-source container



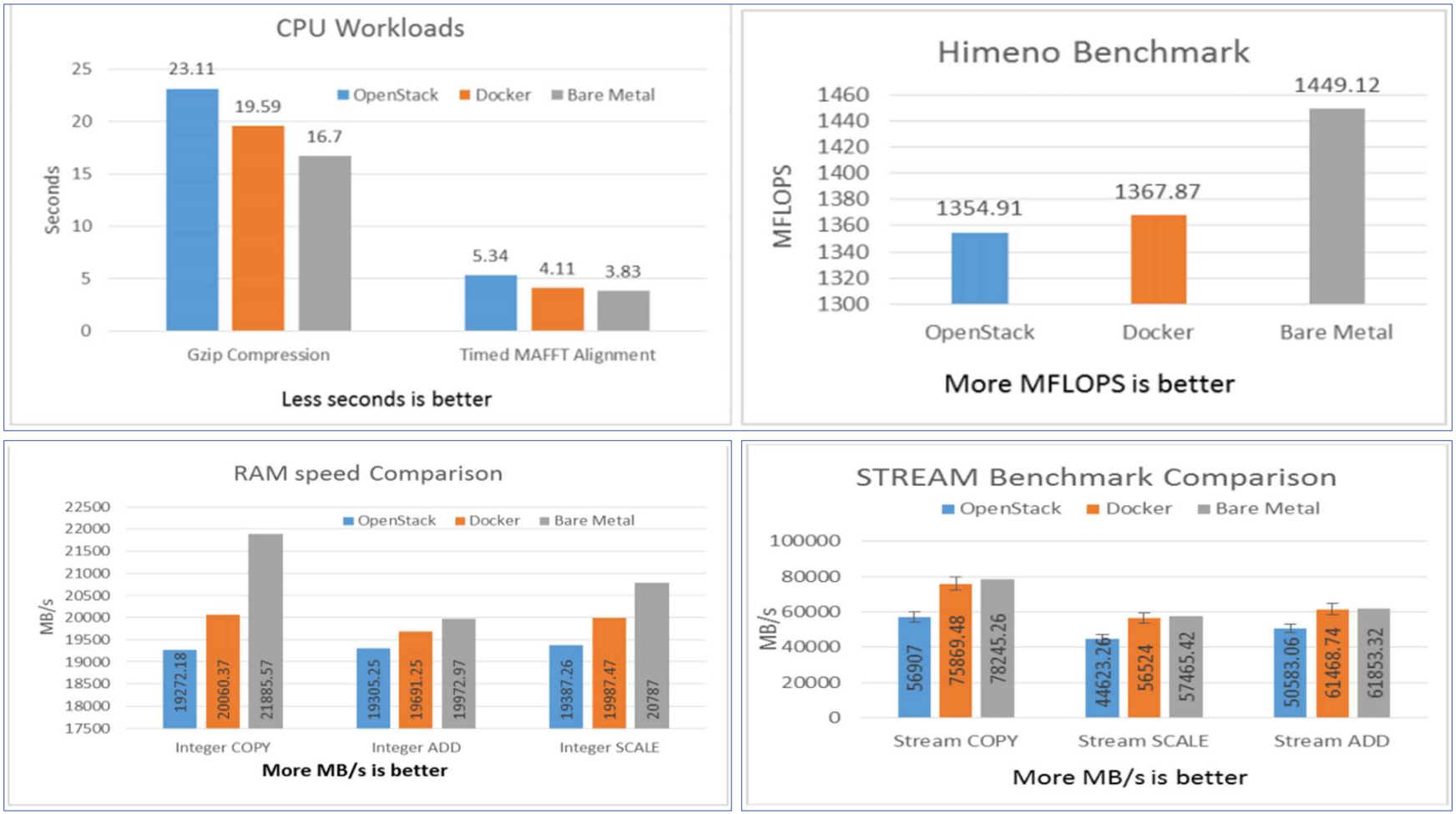
※ 참고 : https://developer.ibm.com/tutorials/multi-architecture-cri-o-container-images-for-red-hat-openshift/

Virtual Machine vs. Containers



※ 참고 : https://cloudblogs.microsoft.com/opensource/2019/07/15/how-to-get-started-containers-docker-kubernetes/

Docker Container vs. Openstack VM vs. Bare Metal Server



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

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**.

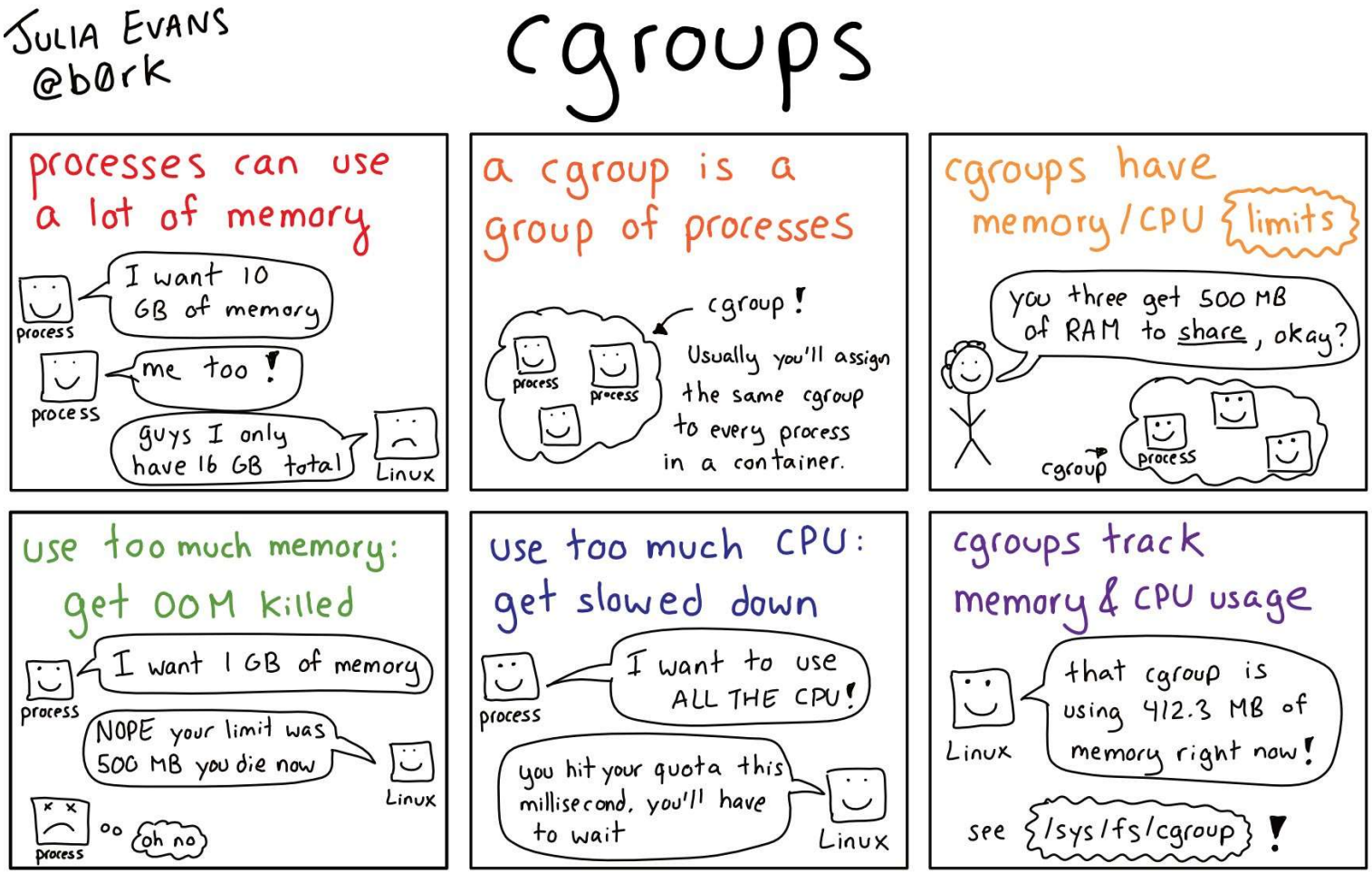
```
❯ cat /proc/cgroups

#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
```

- cgroups를 관리하는 방법으로 `cgroupfs`와 `systemd` 2 종류 존재
- 또한 cgroups 정책도 `v1`, `v2` 2 종류가 공존
- Kubernetes v1.22에서 이와 관련 정책 변경이 있음

```
❯ ls -al /sys/fs/cgroup

합계 0
drwxr-xr-x 15 root root 380 12월 24 23:02 .
drwxr-xr-x  9 root root   0 12월 24 23:02 ..
dr-xr-xr-x  4 root root   0 12월 24 23:02 blkio
lrwxrwxrwx  1 root root  11 12월 24 23:02 cpu -> cpu,cpuacct
dr-xr-xr-x  4 root root   0 12월 24 23:02 cpu,cpuacct
lrwxrwxrwx  1 root root  11 12월 24 23:02 cpuacct -> cpu,cpuacct
dr-xr-xr-x  2 root root   0 12월 24 23:02 cpuset
dr-xr-xr-x  4 root root   0 12월 24 23:02 devices
dr-xr-xr-x  2 root root   0 12월 24 23:02 freezer
dr-xr-xr-x  2 root root   0 12월 24 23:02 hugetlb
dr-xr-xr-x  4 root root   0 12월 24 23:02 memory
lrwxrwxrwx  1 root root  16 12월 24 23:02 net_cls -> net_cls,net_prio
dr-xr-xr-x  2 root root   0 12월 24 23:02 net_cls,net_prio
lrwxrwxrwx  1 root root  16 12월 24 23:02 net_prio -> net_cls,net_prio
dr-xr-xr-x  2 root root   0 12월 24 23:02 perf_event
dr-xr-xr-x  4 root root   0 12월 24 23:02 pids
dr-xr-xr-x  2 root root   0 12월 24 23:02 rdma
dr-xr-xr-x  5 root root   0 12월 24 23:02 systemd
dr-xr-xr-x  5 root root   0 12월 24 23:02 unified
```

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

cgroups (control groups)



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

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.

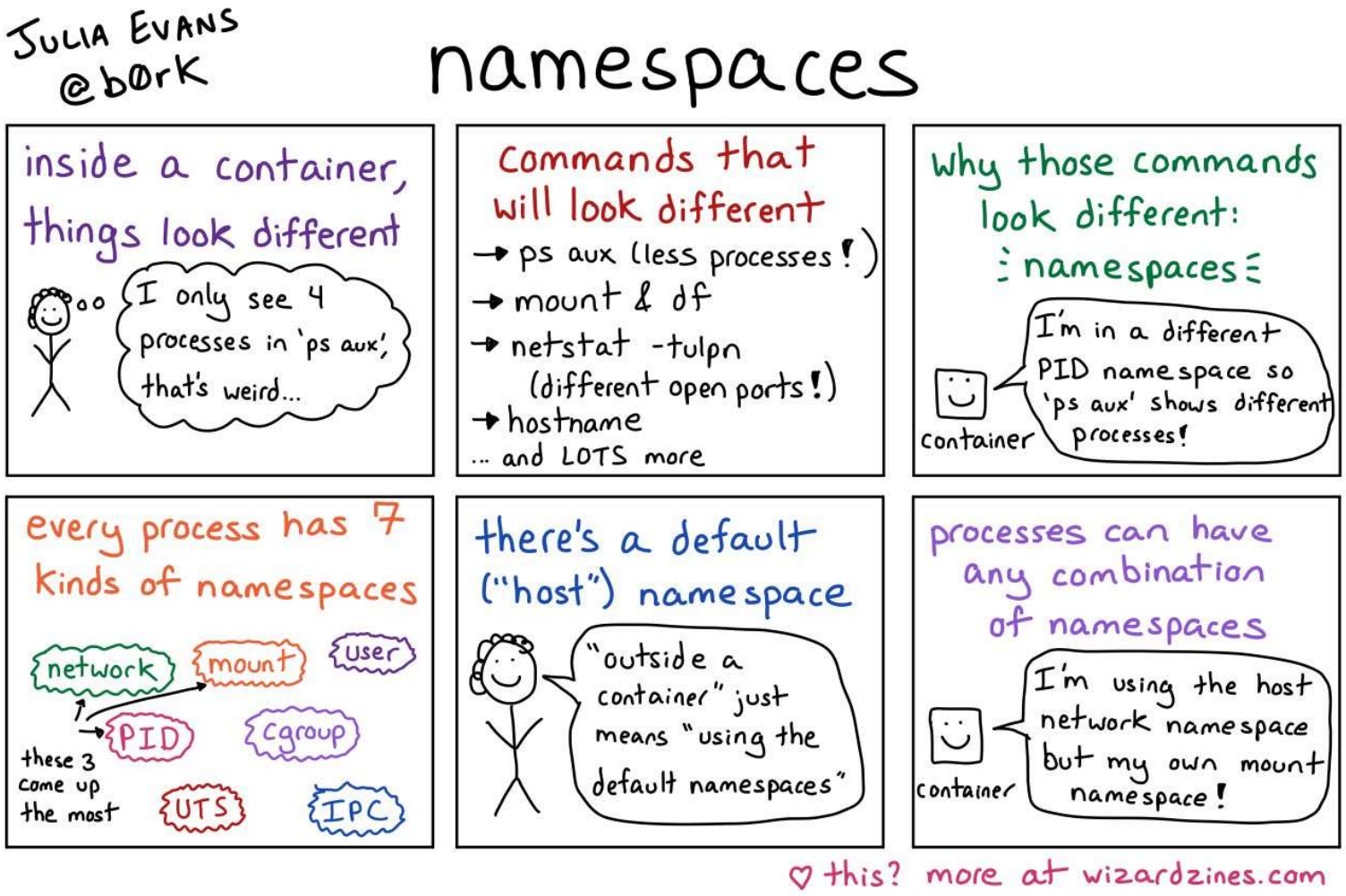
```
❯ sudo ls -al /proc/1/ns

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:[40
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

namespaces

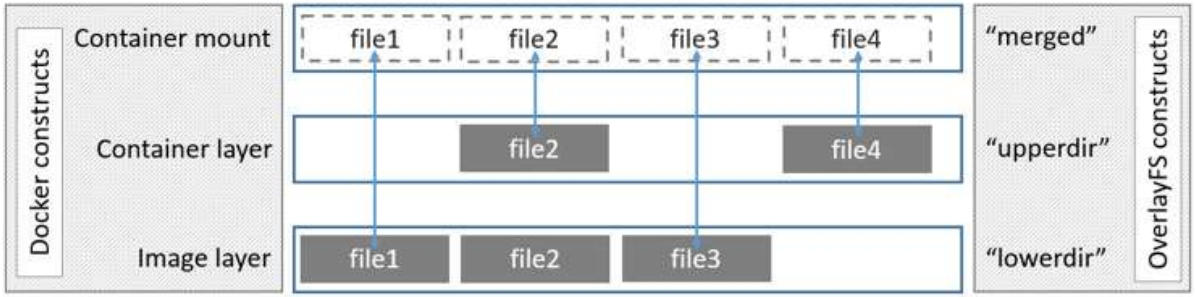


※ 참고 : https://twitter.com/b0rk/status/1240364585766576128

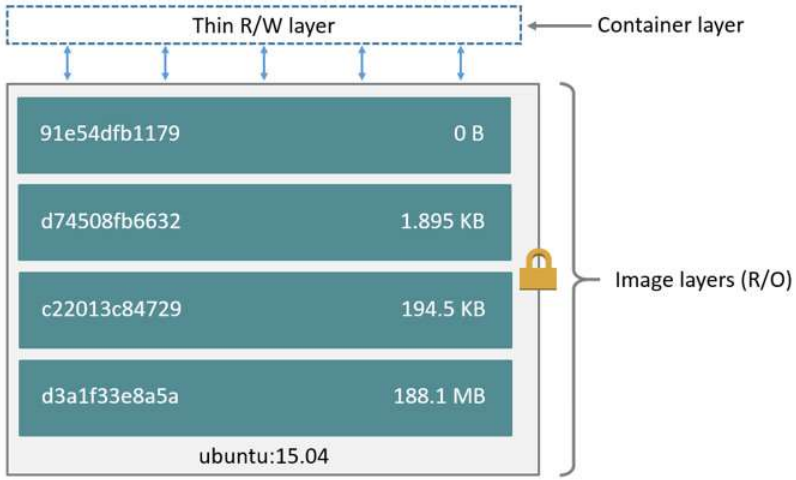
Docker Storage Driver

▷ 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



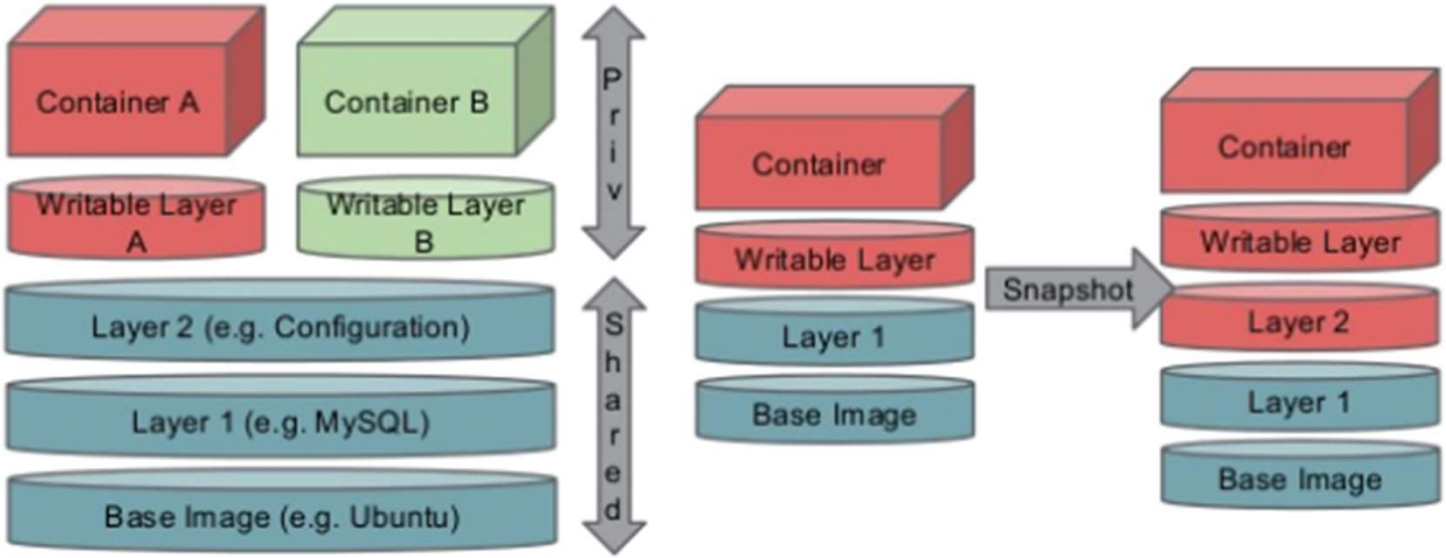
How the overlay driver works



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/

Container Network Model (CNM)

▷ Sandbox

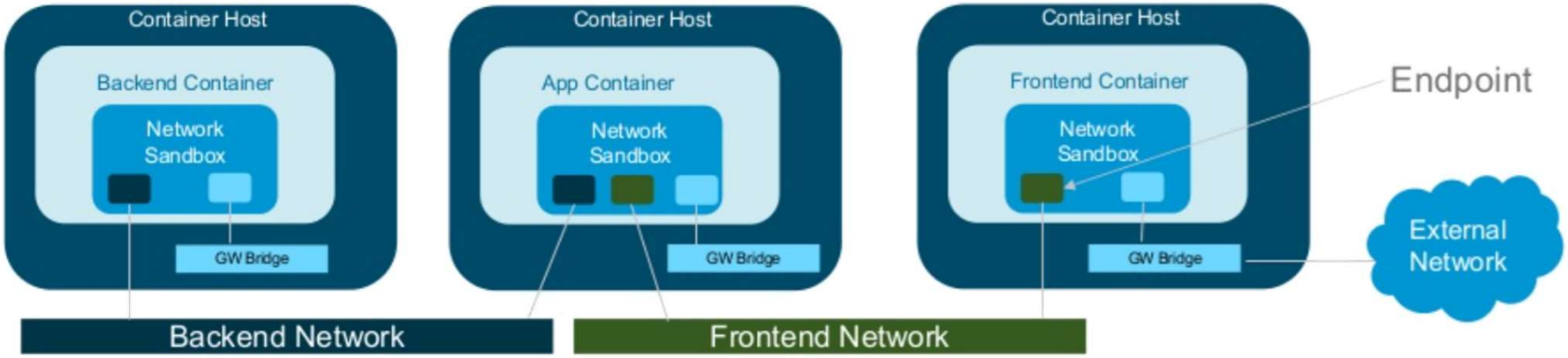
- A Sandbox contains the configuration of a container's network stack.
- This includes management of the container's interfaces, routing table and DNS settings.
- An implementation of a Sandbox could be a Linux Network Namespace, a FreeBSD Jail or other similar concept.

▷ 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

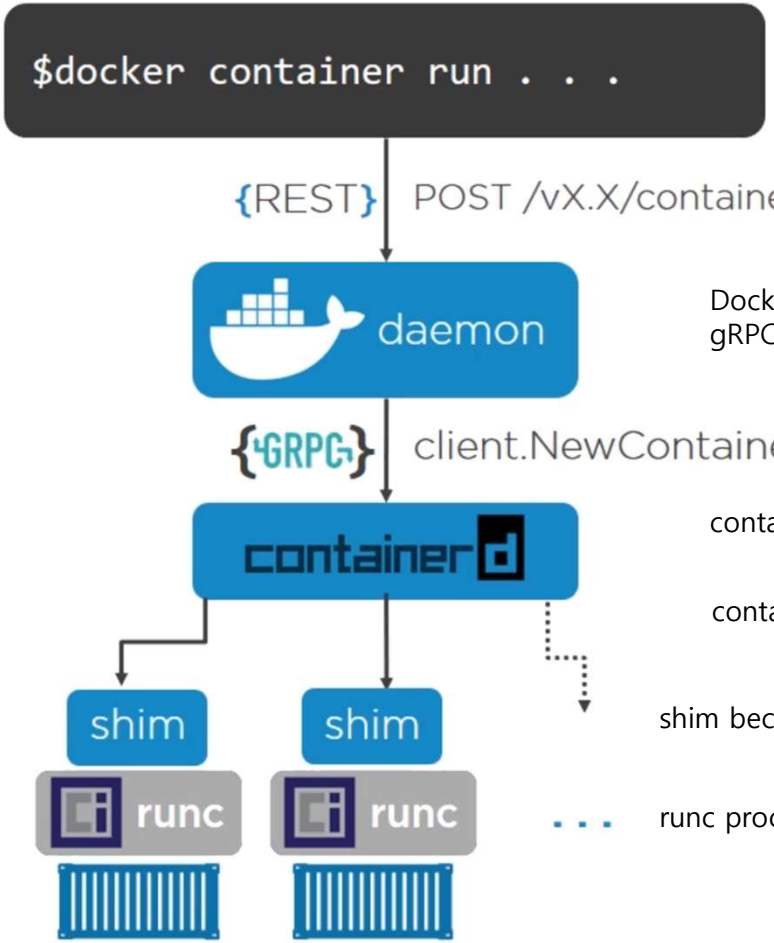
▷ Network

- 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 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

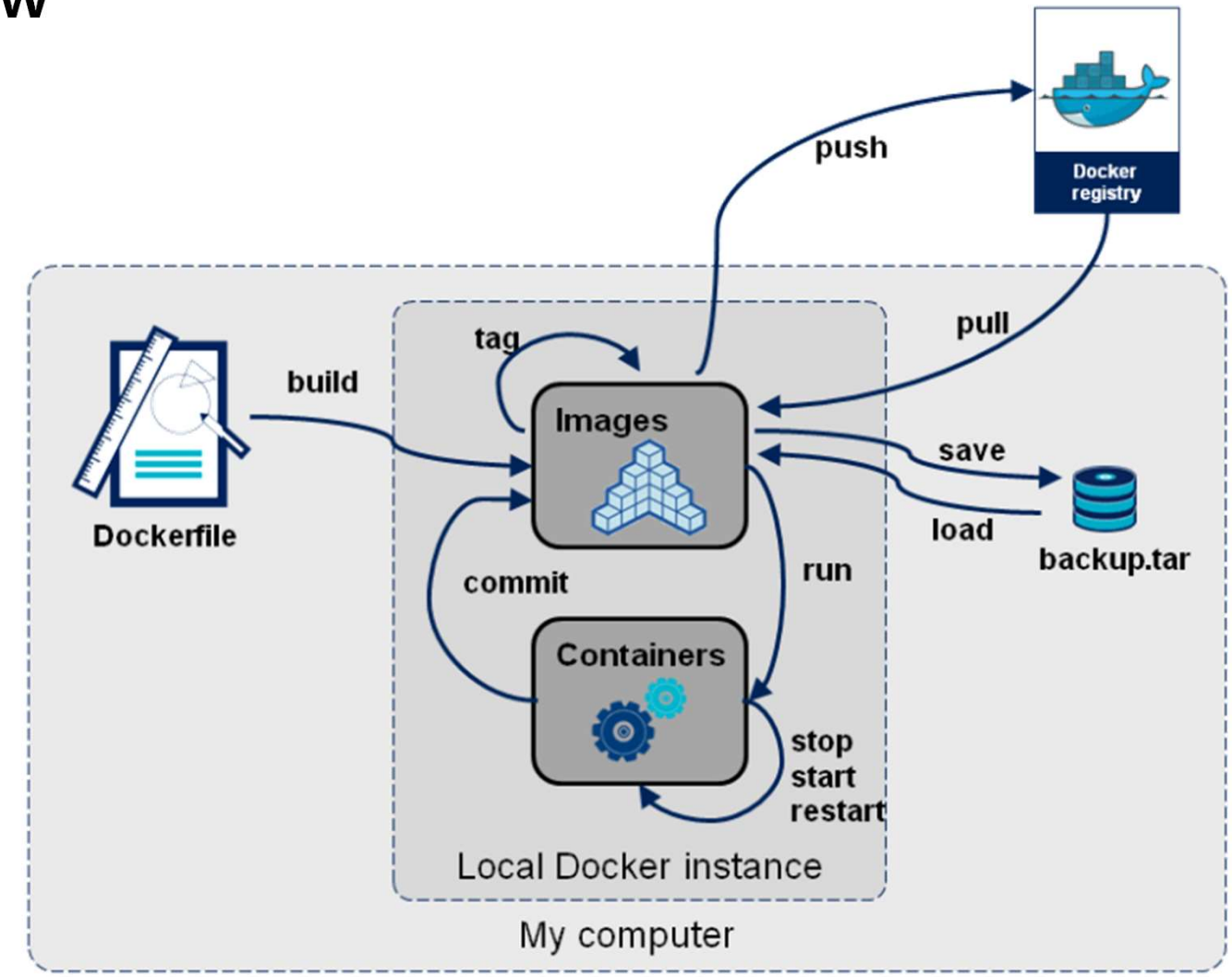
containerd forks the instance of runc for new containers

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



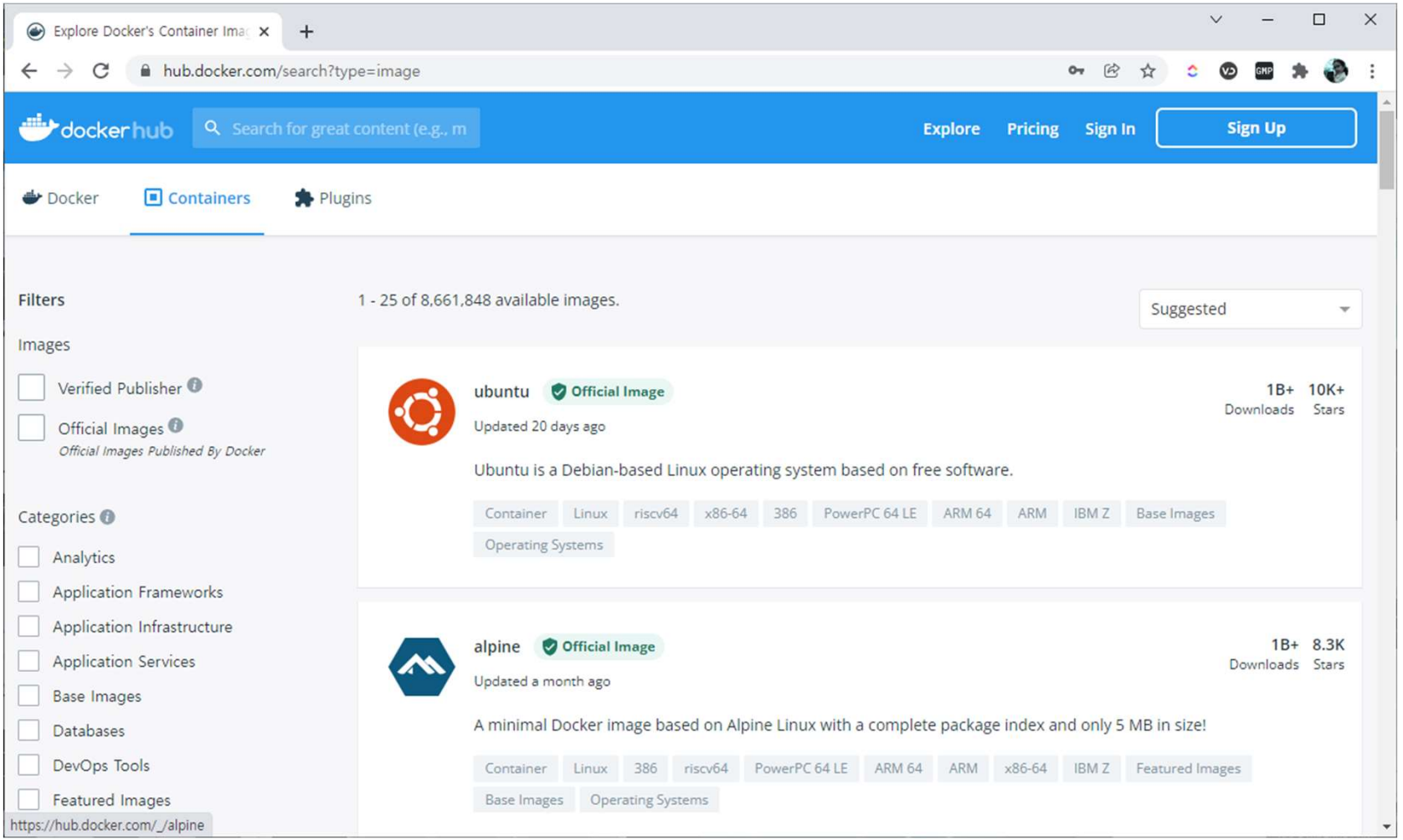
※ 참고 : https://blog.wonizz.tk/2019/07/31/docker-dockerfile/

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"]

```

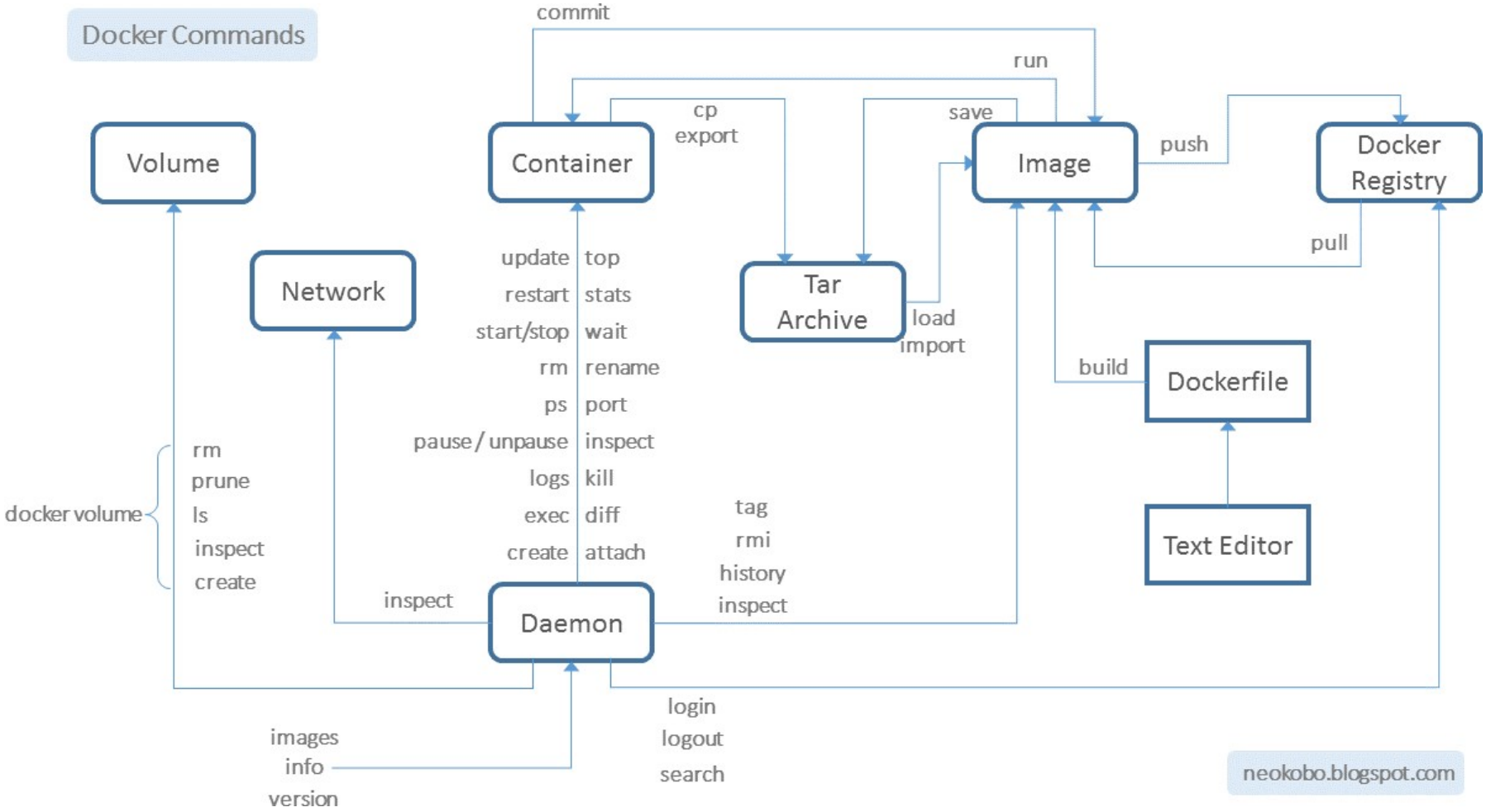
※ 참고 : https://n8d.at/how-to-run-sharepoint-pattern-and-practices-samples-through-docker/

Docker Hub (Registry)



※ 참고 : https://hub.docker.com/

Docker command



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

///

Docker
Hands-On

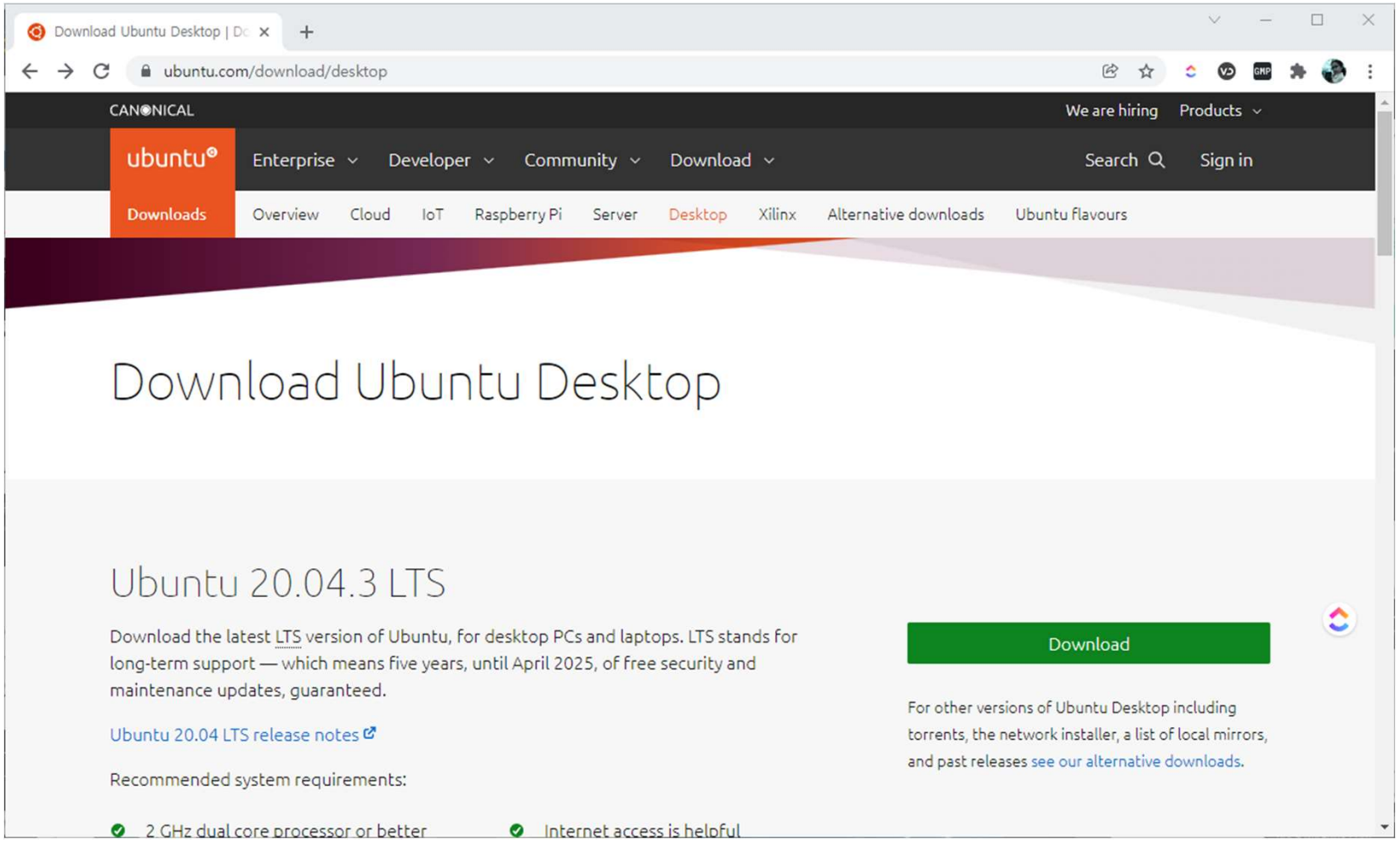
Agenda

- ▷ VirtualBox Install
- ▷ Ubuntu Install
- ▷ Docker Install
- ▷ Hands-On
- ▷ Tip #1 - DockerHub
- ▷ Tip #2 - Katacoda

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>Advanced 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/advanced-kubernetes.git
❯ cd advanced-kubernetes/01-week/

❯ 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

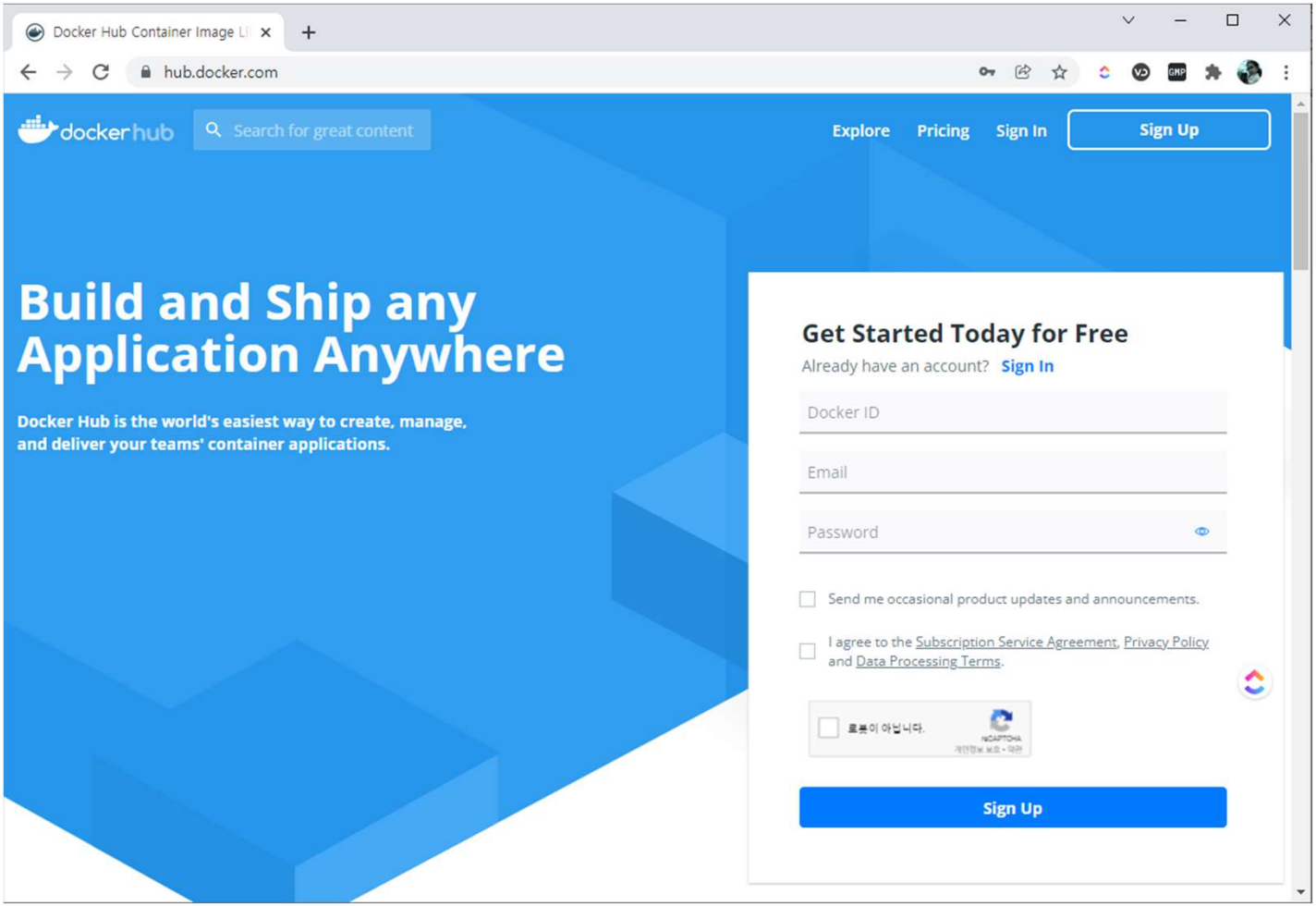
```
❯ docker images

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
```

///

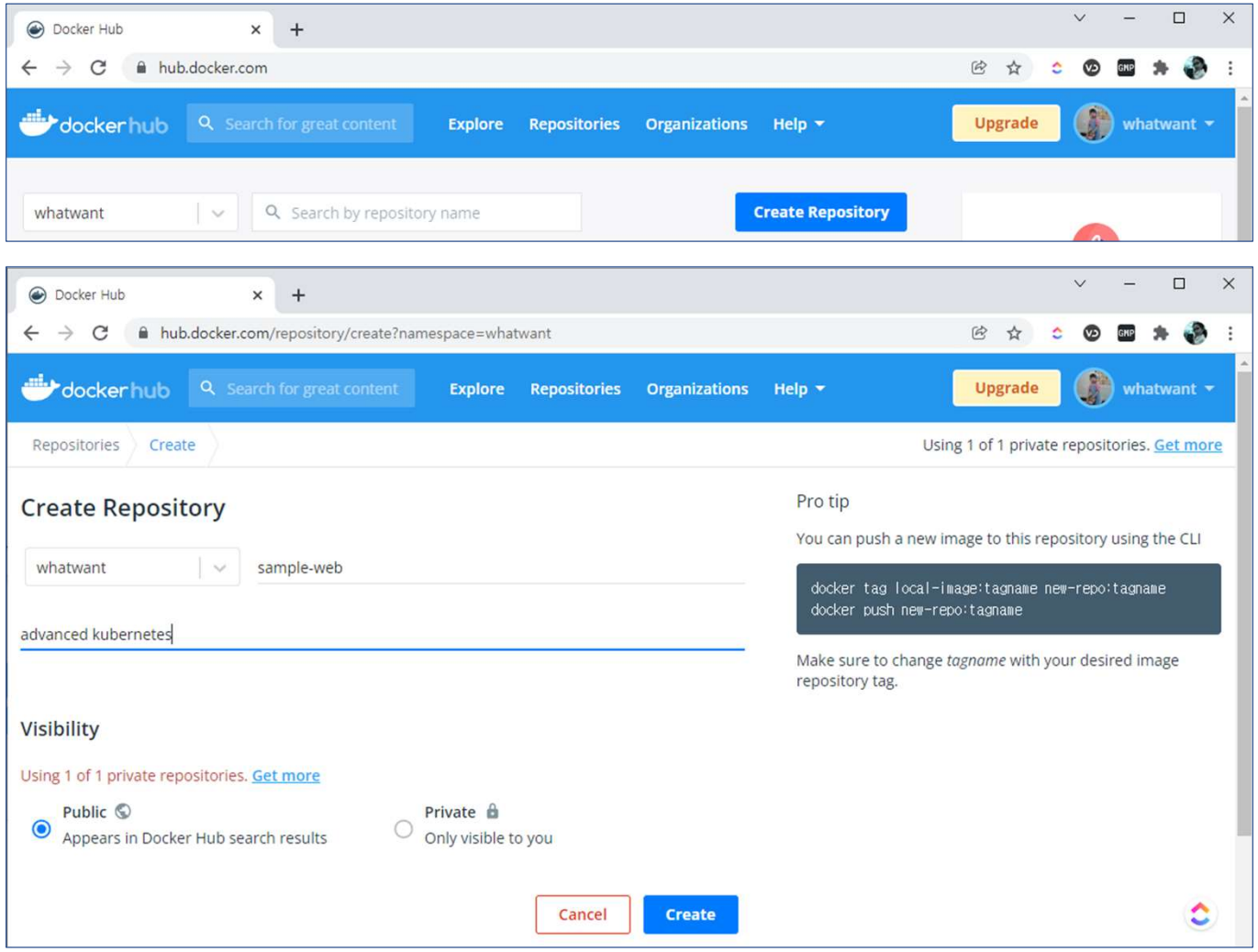
Tip #1 - DockerHub

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
```

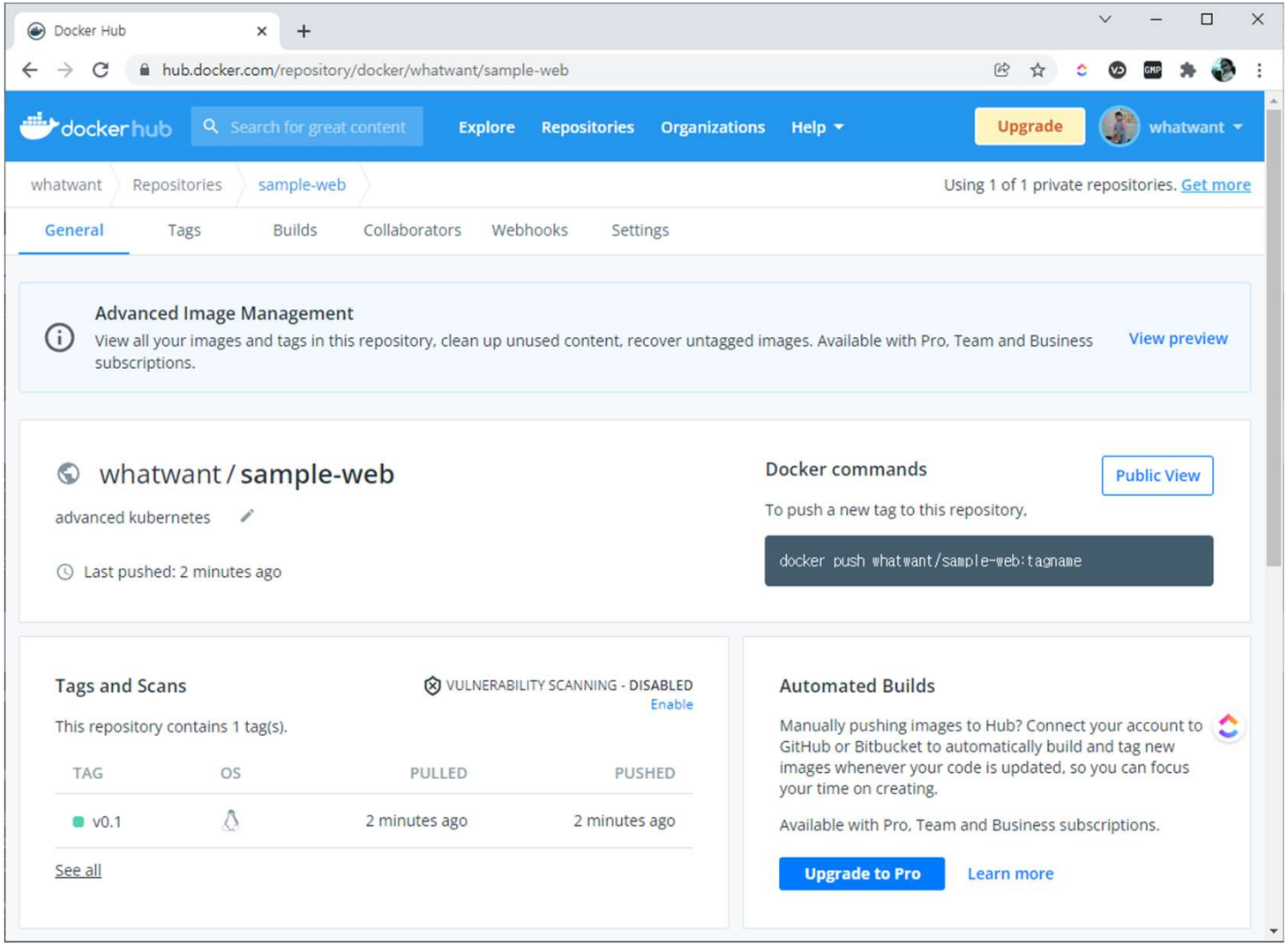
앞에서 진행해봤던 이미지를 재활용 해보자

```
❯ 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

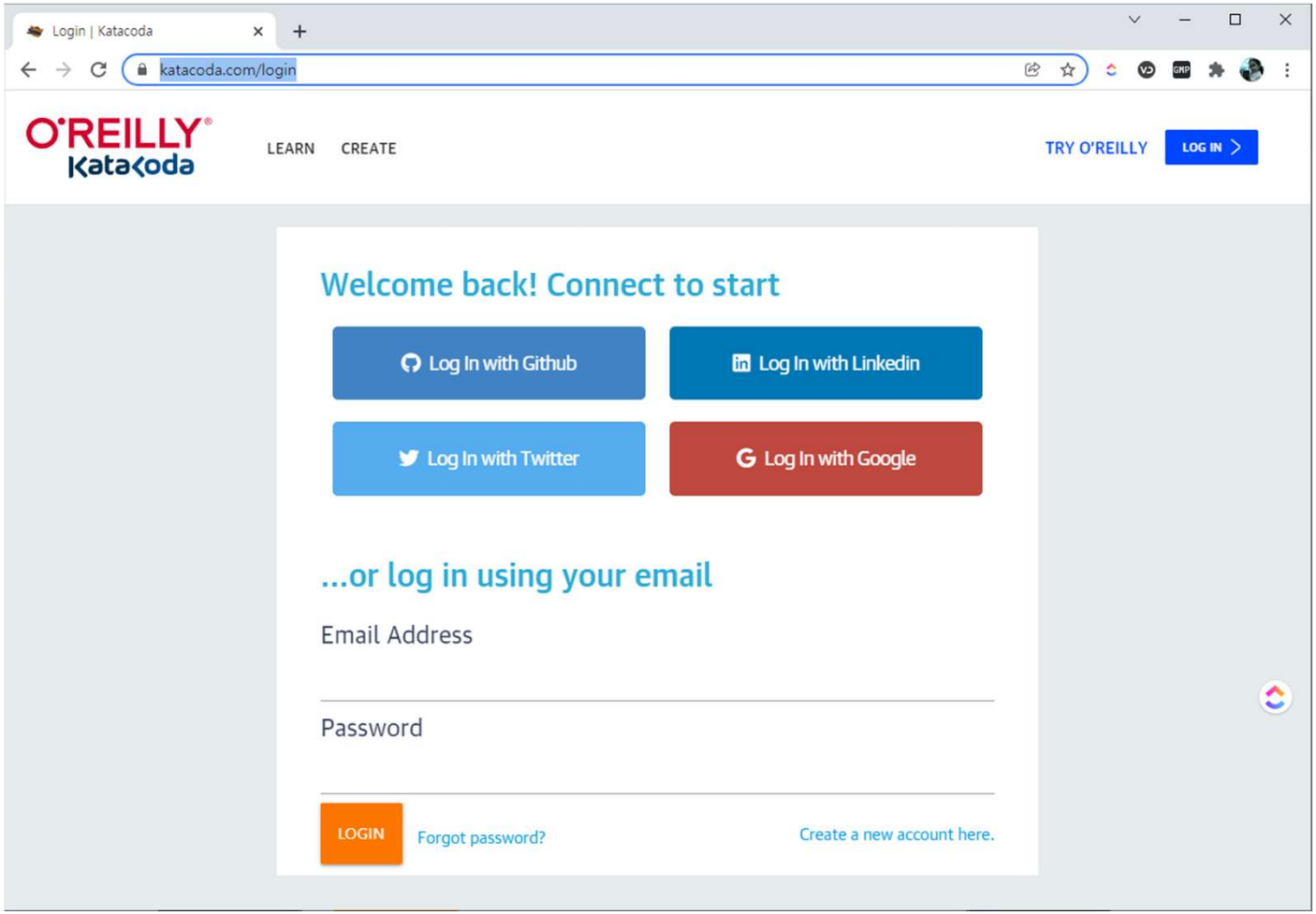


///

Tip #2 - Katacoda

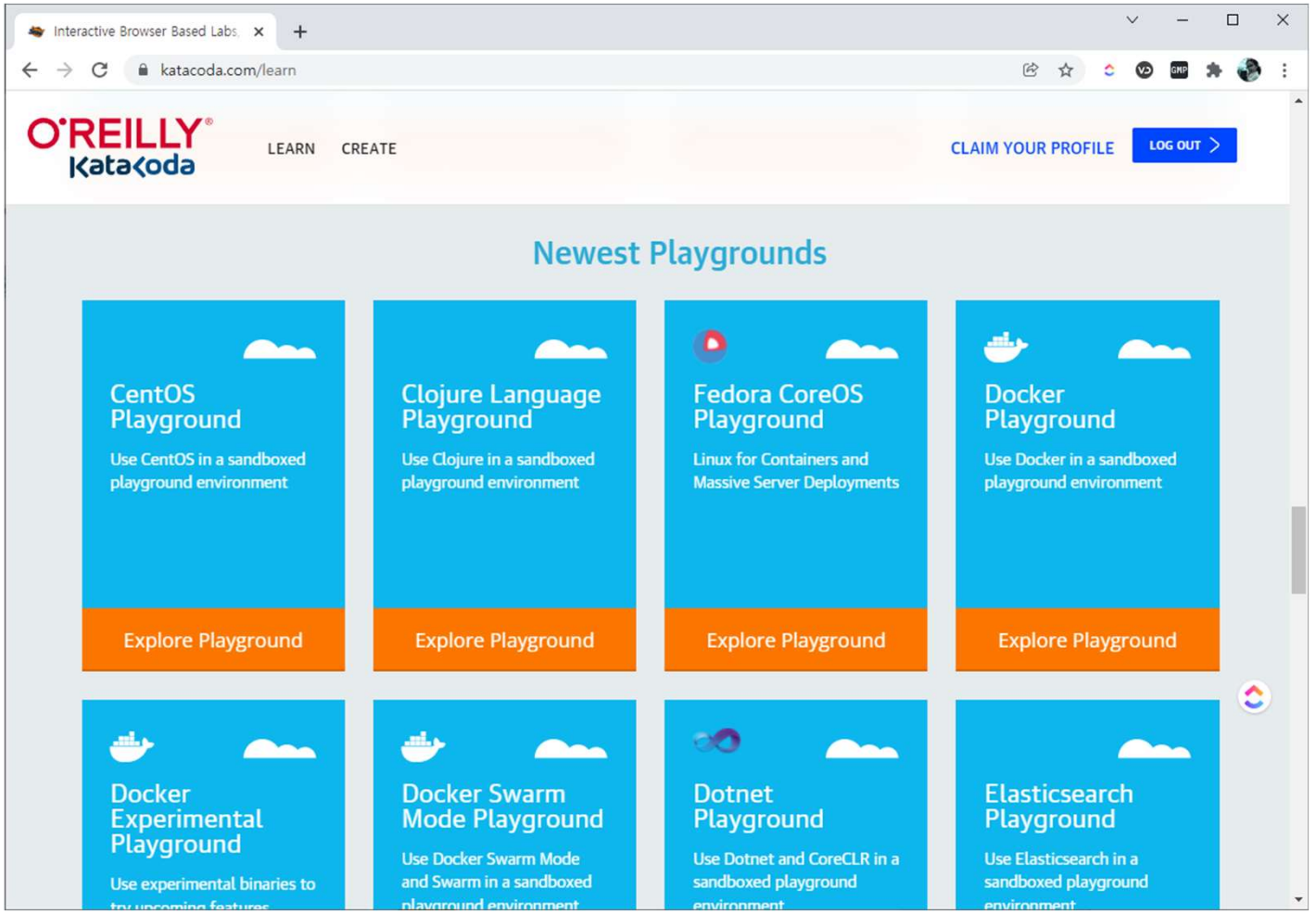
Katacoda - Sign Up / In

https://www.katacoda.com/login



Katacoda - Docker Playground

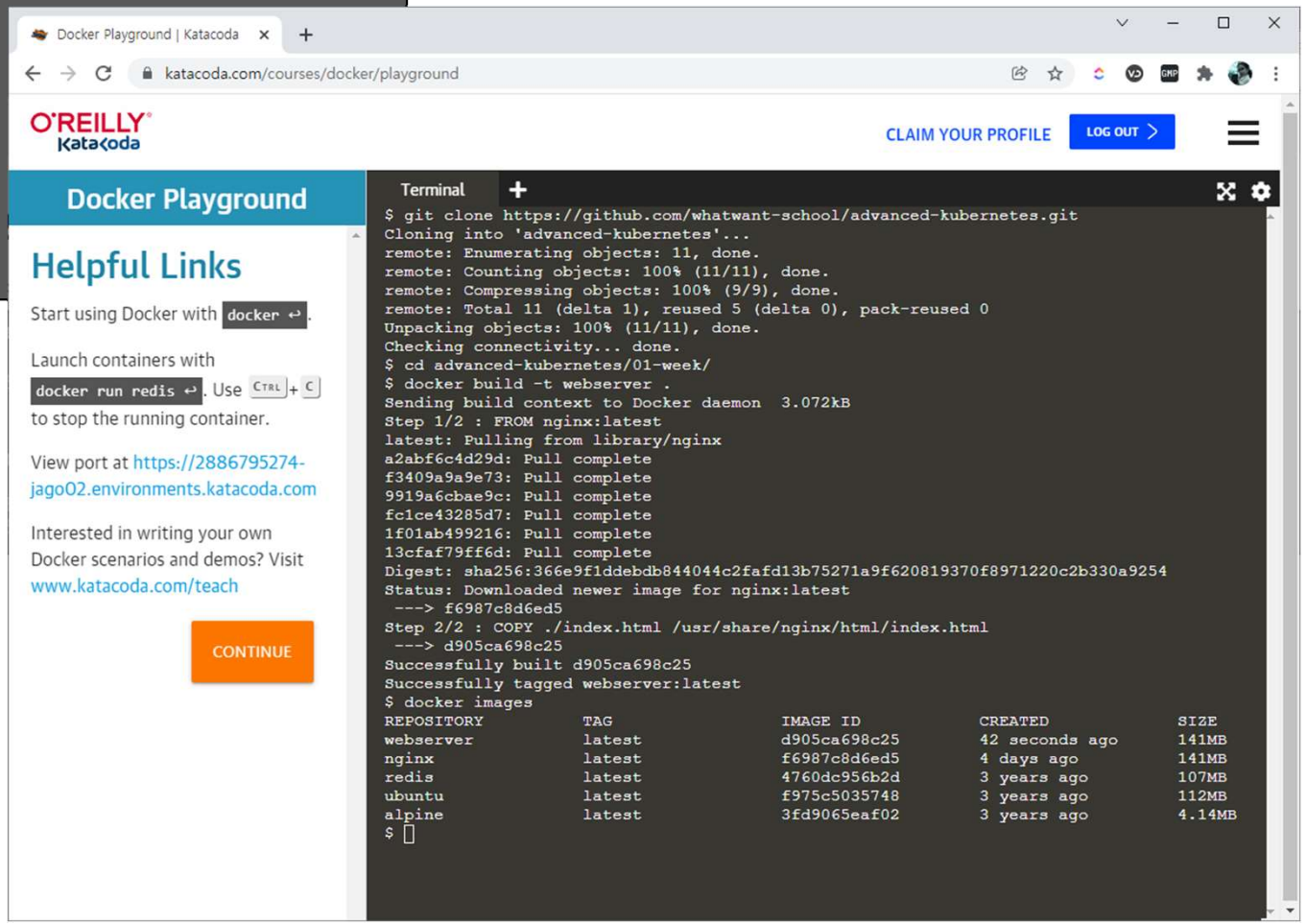
https://www.katacoda.com/courses/docker/playground



Katacoda - docker build / images

정말 고맙게도, git도 지원 해준다.

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

❯ docker build -t webserver .
...

❯ docker images
...
```

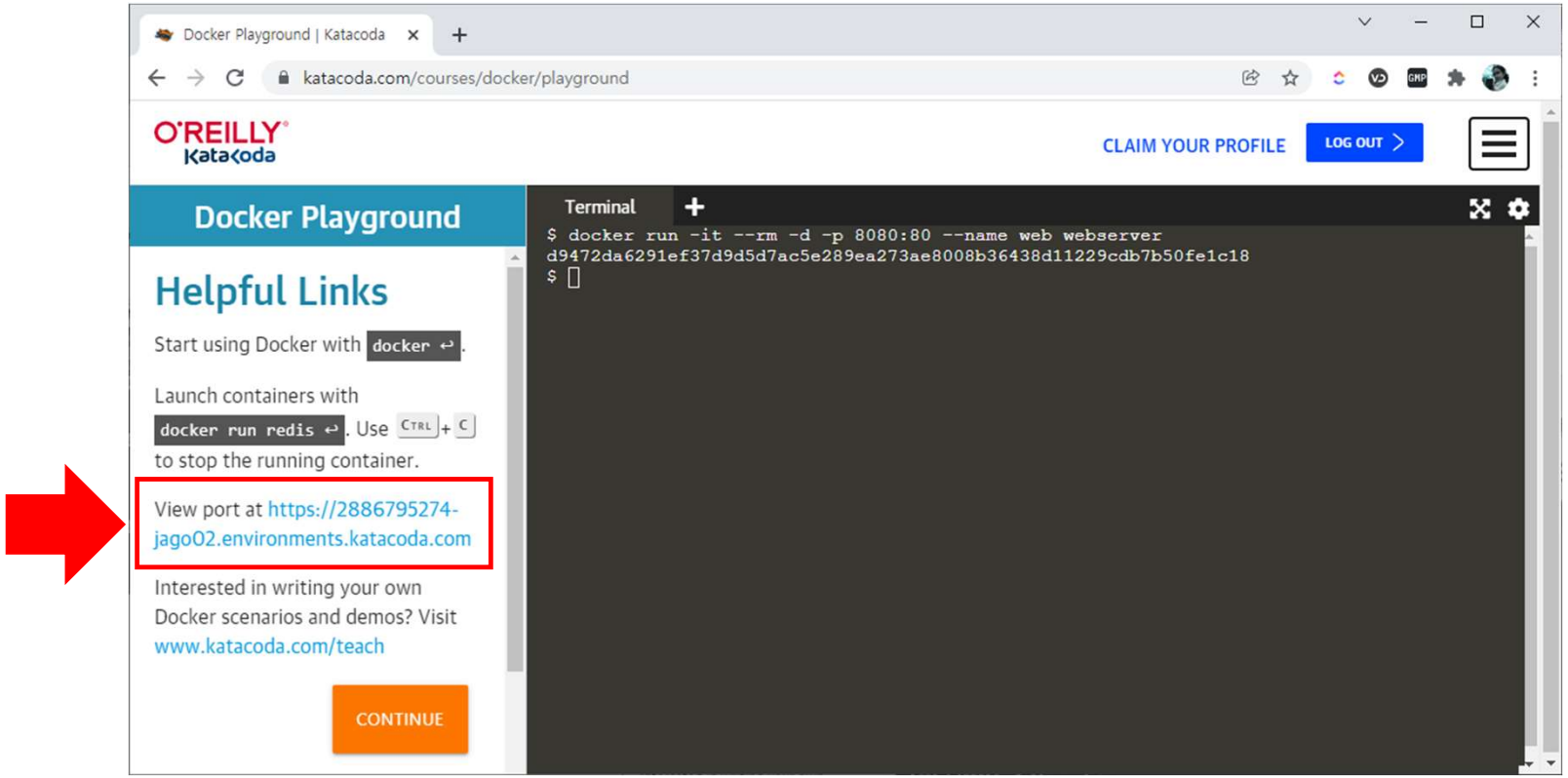


Katacoda - docker run

실행 !!!!

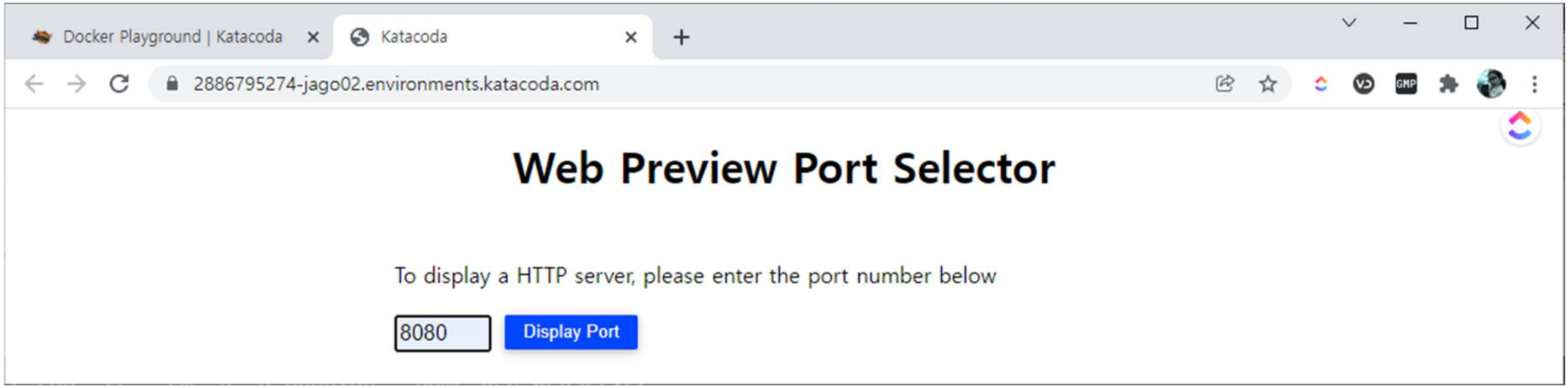
```
❯ docker run -it --rm -d -p 8080:80 --name web webserver
...
```

그런데, 이렇게 띄운 웹서버를 어떻게 확인해야 할까?



Katacoda - Browsing

8080 port로 실행을 했으니, 아래와 같이 적어주고 "Display Port" 클릭 !?!?



///

Breaktime

///

Kubernetes Overview

Kubernetes is ...

Kubernetes, also known as K8s,

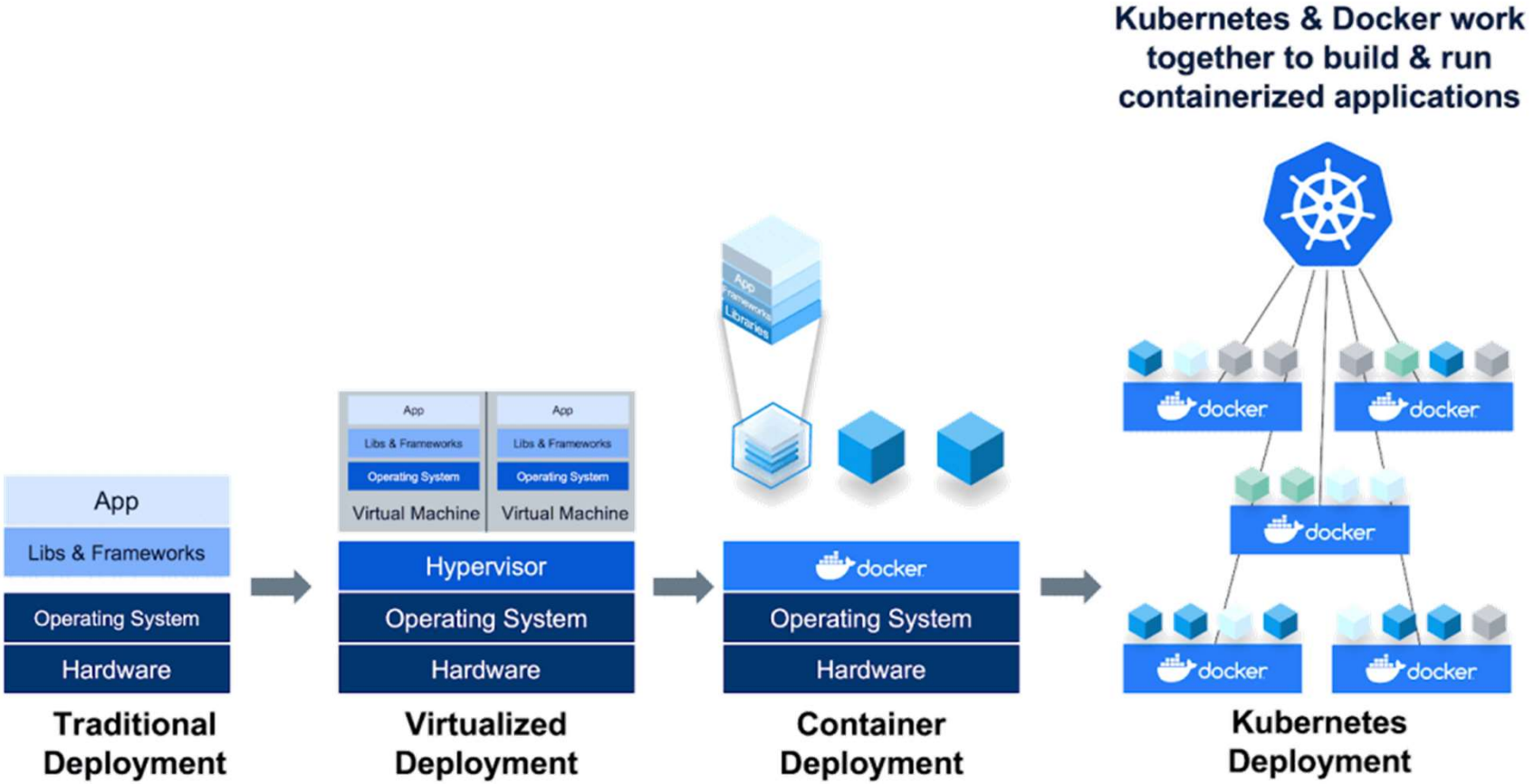
is an open-source system for

automating deployment, scaling, and management

of containerized applications.

※ 참고 : https://kubernetes.io/

Docker & Kubernetes

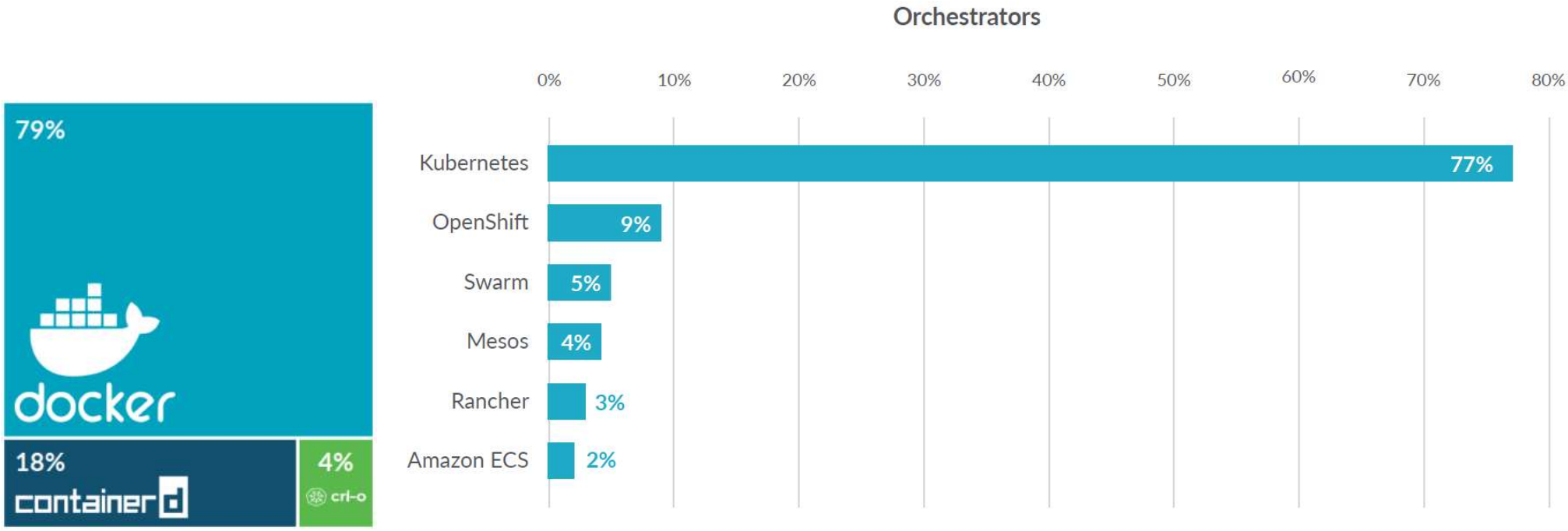


※ 참고 : https://www.docker.com/blog/top-questions-docker-kubernetes-competitors-or-together/

Features of Kubernetes

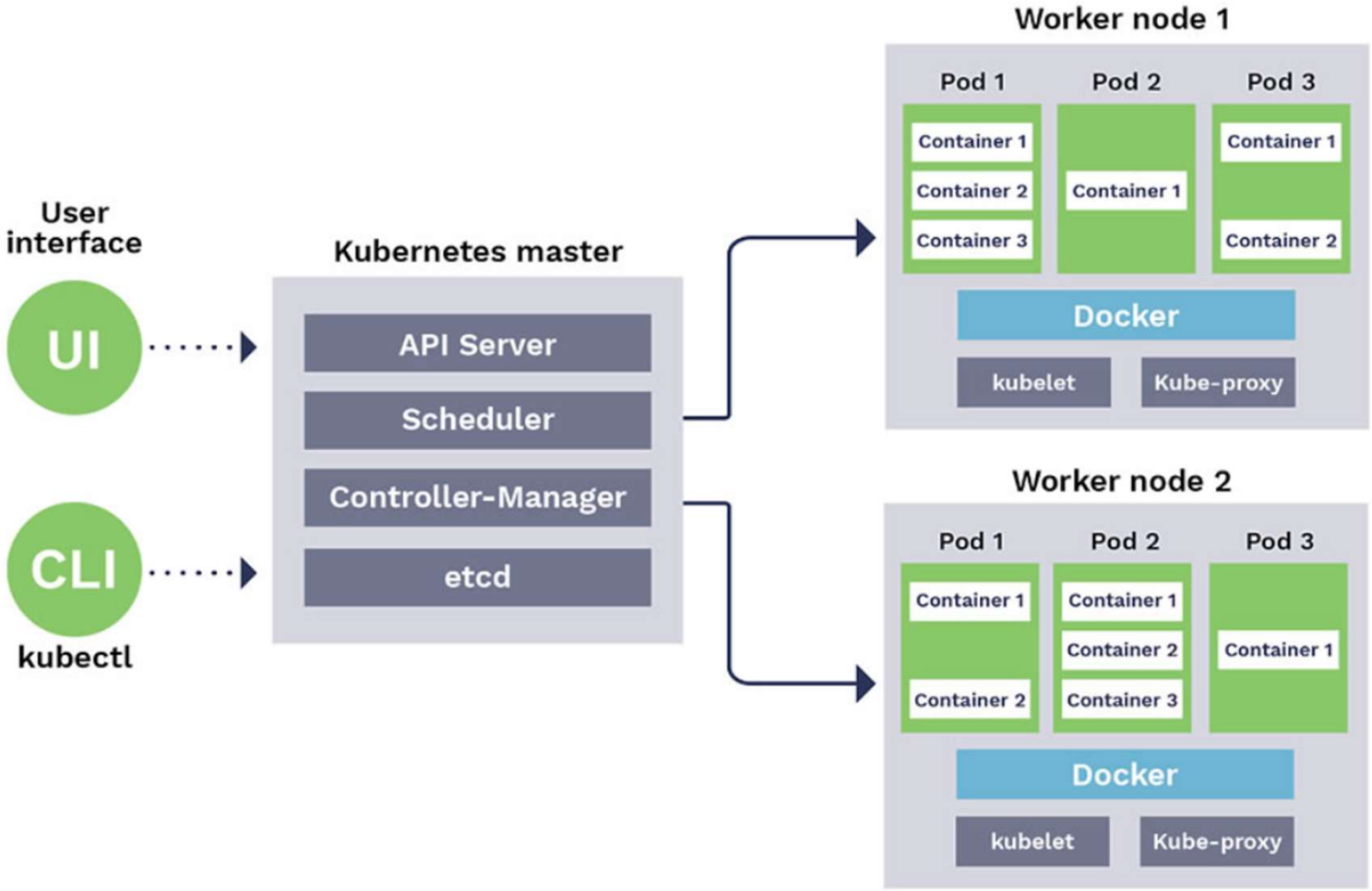
※ 참고 : https://www.xenonstack.com/blog/debug-application-running-in-kubernetes/

Market Share



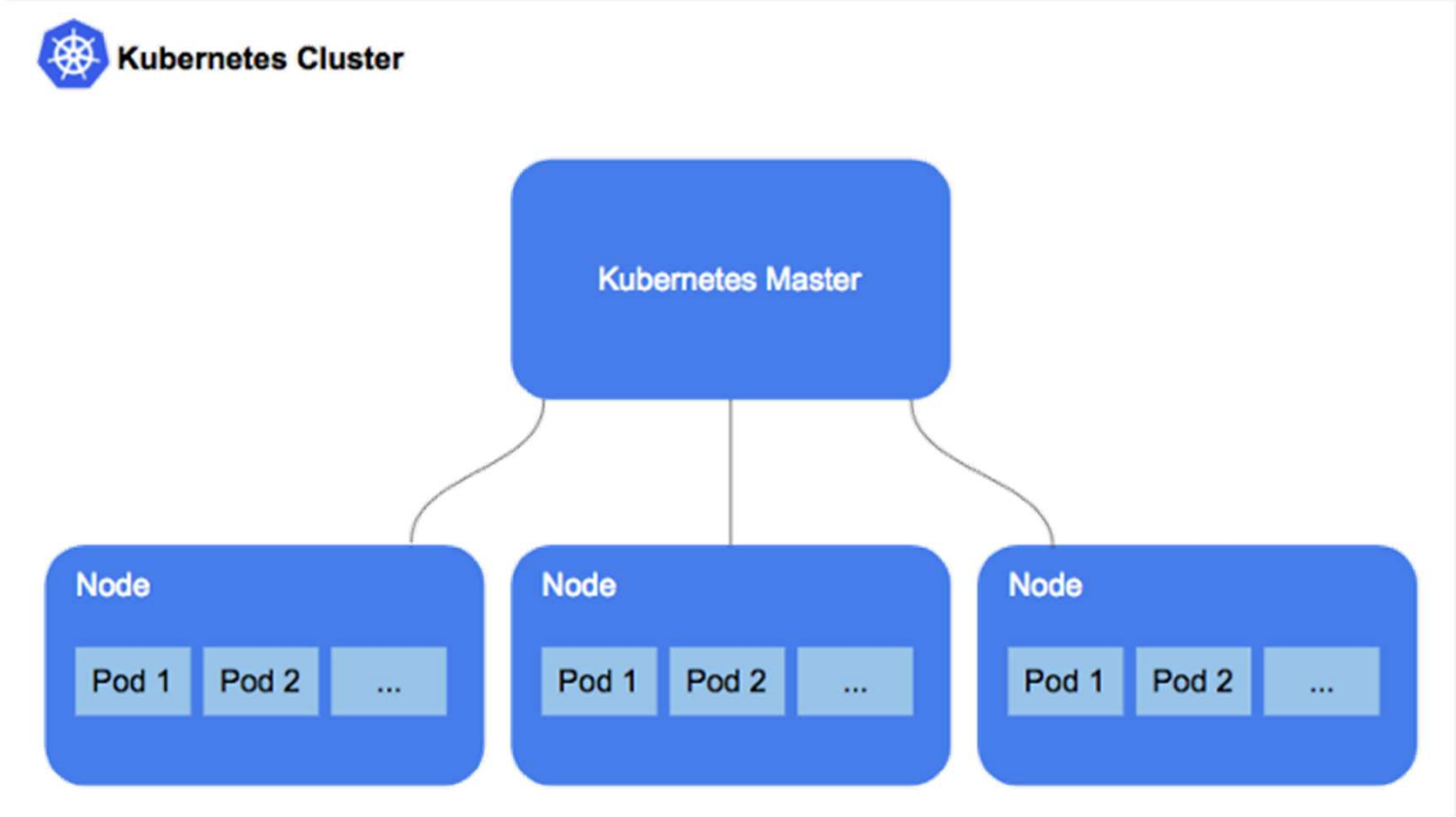
※ 참고 : https://sysdig.com/blog/sysdig-2019-container-usage-report/

Kubernetes Architecture



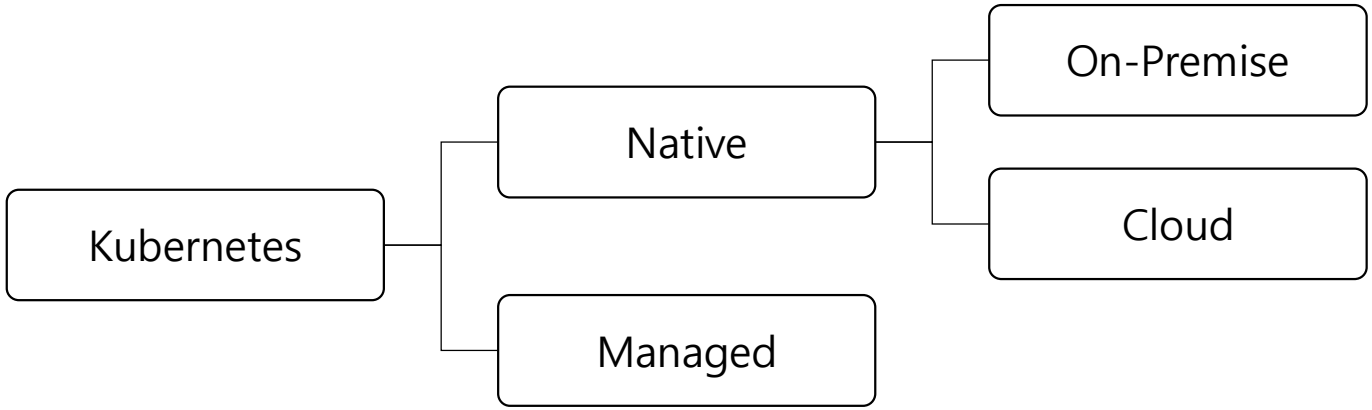
※ 참고 : https://keetmalin.wixsite.com/keetmalin/post/understanding-container-orchestration-with-kubernetes

Cluster



※ 참고 : https://medium.com/@tomerf/so-you-want-to-configure-the-perfect-db-cluster-inside-a-kubernetes-cluster-a4d2c26aca7a

How ...



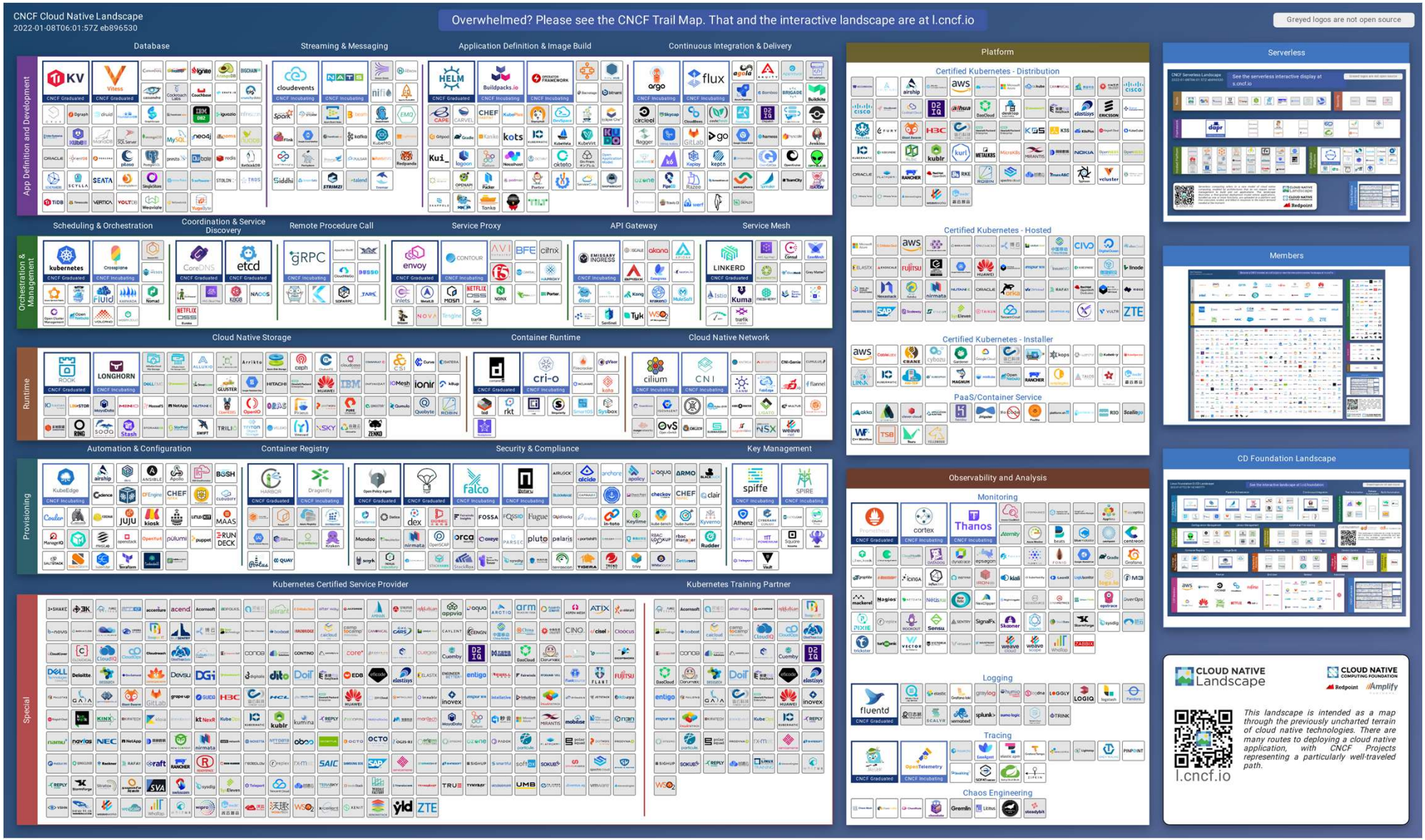
항목	자체 서버환경	Cloud 환경	Managed K8s
인프라 관리	사용자	벤더사	벤더사
K8s 설치, 관리	사용자	사용자	벤더사
K8s(클러스터) 백업	사용자	사용자	벤더사
K8s(클러스터) 스케일링	사용자	사용자	벤더사
워커노드 프로비저닝	사용자	사용자	벤더사
어플리케이션 스케일링	사용자	사용자	사용자
어플리케이션 배포	사용자	사용자	사용자

AWS	EKS (Elastic Kubernetes Service)
Azure	AKS (Azure Kubernetes Service)
GCP	GKE (Google Kubernetes Service)
NCP	NKS (Naver Kubernetes Service)

※ 참고 : https://www.cloocus.com/insight-kubernetes_2/

///

Tip #3 - CNCF



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

///

https://kahoot.it/

자습 (복습)

▷ Docker

▷ VirtualBox