

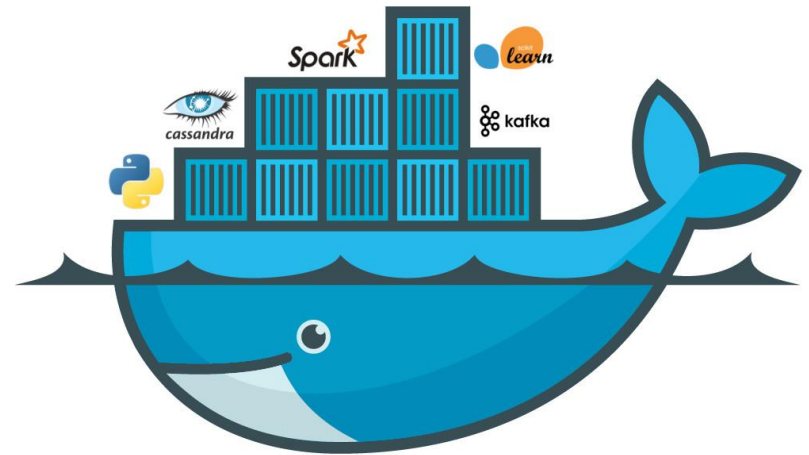


# D-Tux 연구주제: Docker

[hsc080808@naver.com](mailto:hsc080808@naver.com)

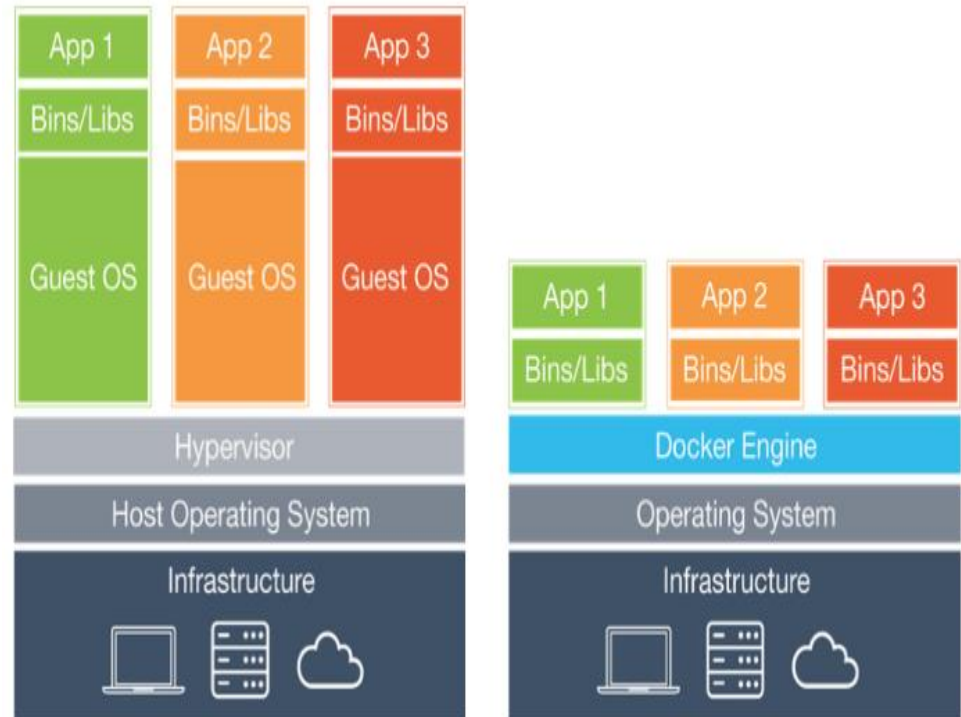
# What is Docker?

- 리눅스 **컨테이너** 기술을 기반으로 하는 오픈 소스 소프트웨어 플랫폼
- 애플리케이션과 실행에 필요한 시스템 환경을 모아서 컨테이너로 관리
- 컨테이너 안에 가상공간을 만들고 실행 파일을 호스트에서 직접 실행
  - 리눅스 커널이 기술 제공
    - cgroup
    - namespaces



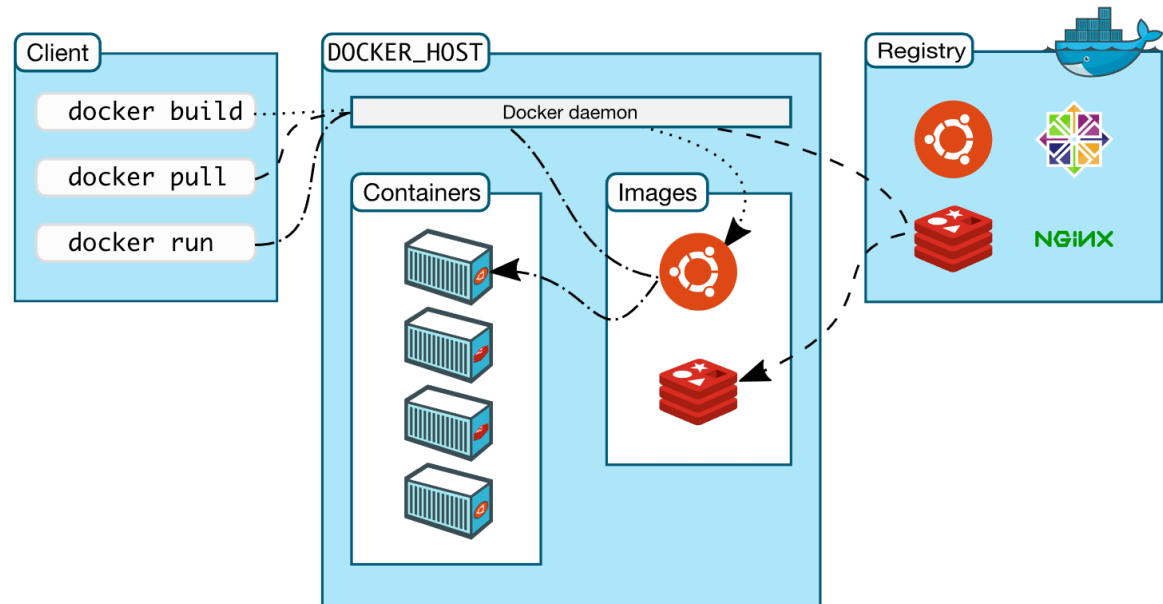
# VM vs Container

- VM: 가상화된 하드웨어 위에 OS 가 올라가는 형태로 거의 완벽하게 host와 분리
- Container: 전체 운영체제를 가상화 하지 않고 단지 소프트웨어가 실행할 때 필요한 라이브러리 및 설정들만 포함.
- 빠르다
- 가볍다
- 비용절감
- etc



# Why Container?

- 개발자는 프로젝트에 적합한 환경과 툴을 사용가능.
  - Ex) OS: 우분투, DB: MYSQL 등의 설정 후 이미지 생성
  - 해당 이미지를 통해 컨테이너 생성.



<https://docs.docker.com/engine/docker-overview/#docker-architecture>

# Container 작동 원리

- Cgroup (Control Group)
  - 시스템의 CPU시간, 메모리, 네트워크 대역폭등의 자원을 제한하고 격리할 수 있는 커널 기능
- Namespace
  - 다른 프로세스와의 격리 시키는 기능
- Container
  - Cgroup + namespace 의 기술을 이용한 프로세스 격리 기술.

# Cgroup (sys/fs/cgroup)

- Subsystem 중 cpu 사용 그룹당
  - Time period
  - Cpu 점유율 등의 설정 가능
  - /sys/fs/cgroup/ 경로에 subsystems 위치.
- Process: matho-primes
  - 지정된 범위까지 소수값을 출력
  - Cpu 사용량을 확인하기 위해 사용.

# Cgroup 실습

- 총 4개의 control group 생성 후 각각의 group 당 cpu 점유율 지정 (default: 1024) 각 그룹당 cpu 사용량 비율 (4:2:1:1)

```
root@hansol-VirtualBox:/home/hansol# cgcreate -g cpu:/cpushare_1
root@hansol-VirtualBox:/home/hansol# cgcreate -g cpu:/cpushare_2
root@hansol-VirtualBox:/home/hansol# cgcreate -g cpu:/cpushare_3
root@hansol-VirtualBox:/home/hansol# cgcreate -g cpu:/cpushare_4
root@hansol-VirtualBox:/home/hansol# cgset -r cpu.shares=512 cpushare_2
root@hansol-VirtualBox:/home/hansol# cgset -r cpu.shares=256 cpushare_3
root@hansol-VirtualBox:/home/hansol# cgset -r cpu.shares=256 cpushare_4
root@hansol-VirtualBox:/home/hansol#
```



# Cgroup 실습

- cgexec 명령어를 통해 총 4개의 matho-primes process 실행

```
root@hansol-VirtualBox:/home/hansol# cgexec -g cpu:cpushare_1 /usr/bin/matho
matho      matho-mult  matho-pascal matho-primes matho-sum    matho-sumsq  mathomatic
root@hansol-VirtualBox:/home/hansol# cgexec -g cpu:cpushare_1 /usr/bin/matho-primes 0 999999999 > /dev/null &
[1] 3833
root@hansol-VirtualBox:/home/hansol# cgexec -g cpu:cpushare_2 /usr/bin/matho-primes 0 999999999 > /dev/null &
[2] 3836
root@hansol-VirtualBox:/home/hansol# cgexec -g cpu:cpushare_3 /usr/bin/matho-primes 0 999999999 > /dev/null &
[3] 3837
root@hansol-VirtualBox:/home/hansol# cgexec -g cpu:cpushare_4 /usr/bin/matho-primes 0 999999999 > /dev/null &
[4] 3838
root@hansol-VirtualBox:/home/hansol# top
```



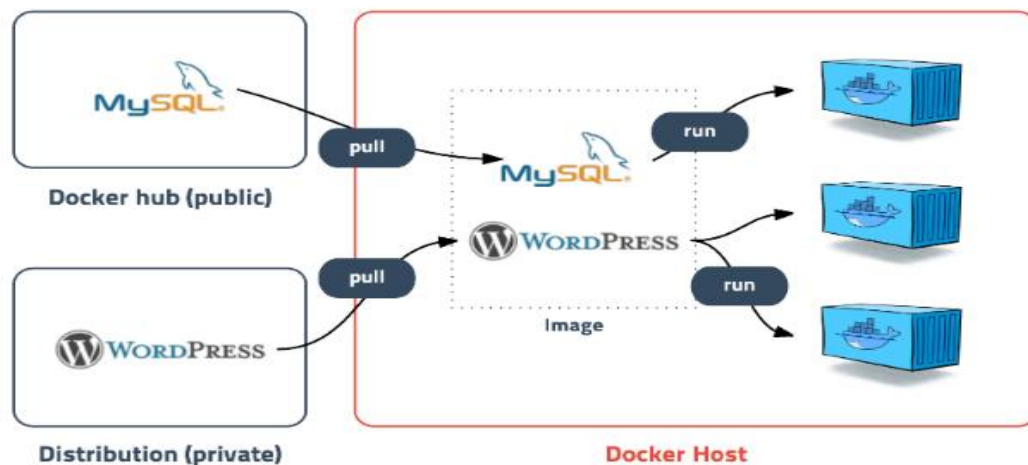
# Cgroup 실습

- Top 명령어를 통해 cpu 점유율 확인
  - 모두 100% 사용시 강제로 지정된 cpu만을 사용해 실행
  - taskset 명령어 사용
    - Ex) taskset -c 0
      - 해당 명령어를 사용해 core 0번만을 사용해 프로세스 실행

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
5457	root	20	0	10188	3564	1684	R	49.8	0.0	0:16.37	matho-primes
5458	root	20	0	10188	3736	1864	R	24.9	0.0	0:05.69	matho-primes
5460	root	20	0	10188	3572	1696	R	12.6	0.0	0:01.71	matho-primes
5459	root	20	0	10188	3572	1692	R	12.3	0.0	0:02.18	matho-primes
5461	root	20	0	43208	3984	3212	R	0.3	0.0	0:00.09	top

# Docker

- Image
  - 도커 이미지를 통해 여러 개의 컨테이너 생성 가능
  - 컨테이너 실행에 필요한 파일과 설정값 등을 포함.
    - Ex) MySQL, Ubuntu
  - 이미지 실행 → 컨테이너
  - 컨테이너의 변경/삭제 등이 일어나더라도 이미지는 변하지 않음.
- Docker hub
  - 쉽고 빠르게 이미지의 배포 및 공유 가능



# Image 빌드

- Docker 설치
  - Curl -fsSL <https://get.docker.com/> | sudo sh
  - docker version 을 통해 도커 설치 확인
- Dockerfile 을 통해 이미지 만들기
  - FROM: base image 지정
  - WORKDIR: working directory 지정
  - RUN: 패키지 설치 시 사용
  - EXPOSE: 컨테이너 외부에 노출할 포트번호
  - CMD: 컨테이너가 실행될 때 실행할 커맨드

# IMAGE 만들기

- Python 사용
- Pip 을 통해 해당 컨테이너에 필요한 패키지 설치
- 컨테이너 실행시 python app.py 실행

```
# Use an official Python runtime as a parent image
FROM python:2.7-slim

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]
```

# Requirement.txt

- Redis: in-memory 기반 데이터 베이스
- Flask: python 기반 웹 어플리케이션
- App.py
  - 컨테이너 id 와 redis 연결 상태 확인

# app.py

```
from flask import Flask
from redis import Redis, RedisError
import os
import socket

# Connect to Redis
redis = Redis(host="redis", db=0, socket_connect_timeout=2, socket_timeout=2)

app = Flask(__name__)

@app.route("/")
def hello():
    try:
        visits = redis.incr("counter")
    except RedisError:
        visits = "<i>cannot connect to Redis, counter disabled</i>"

    html = "<h3>Hello {name}!</h3>" \
        "<b>Hostname:</b> {hostname}<br/>" \
        "<b>Visits:</b> {visits}"
    return html.format(name=os.getenv("NAME", "world"), hostname=socket.gethostname(), visits=visits)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80)
```

# build

- `docker build --tag=[image id]`
- `docker images` 을 통해 이미지 빌드 확인
- `Docker run -p [mappin할 포트번호]:80 [image id]`
- `localhost:4000` 을 통해 확인

**Hello World!**

**Hostname:** aedecf35015e

**Visits:** *cannot connect to Redis, counter disabled*





# 이미지 관리 및 배포

- Docker hub 에 가입
  - <https://hub.docker.com/>
- Docker login 을 통해 로그인 (깃 허브와 유사함)
- 이미지 배포
  - Docker tag [image] [username]/[repository]:[tag]




```
root@hs-desktop:/home/hs/docker_example/python_execute# docker push hsc0808/gettingstart:part1
The push refers to repository [docker.io/hsc0808/gettingstart]
a41daa89d927: Mounted from hsc0808/redis
c0be62c6e1a8: Mounted from hsc0808/redis
fc7000c6a8ff: Mounted from hsc0808/redis
af9628477752: Mounted from hsc0808/redis
f1bd403e5041: Mounted from hsc0808/redis
b7fcb2747224: Mounted from hsc0808/redis
7b4e562e58dc: Mounted from hsc0808/redis
part1: digest: sha256:3f60aea1f8755a11ccbff846771bea9f88410815008ca05ba14a9fb1f440322d size: 1788
```

# Docker hub 예시

 Explore Repositories Organizations Get Help hsc0808 

Repositories hsc0808 / gettingstart Using 0 of 1 private repositories. [Get mo](#)



General Tags Builds Timeline Collaborators Webhooks Settings

 **hsc0808 / gettingstart**  
*This repository does not have a description*   
 Last pushed: 3 minutes ago

**Docker commands**  
To push a new tag to this repository,  

```
docker push hsc0808/gettingstart:tagname
```

**Tags**  
This repository contains 1 tag(s).

part1 3 minutes ago

[See all](#)

# Docker:service

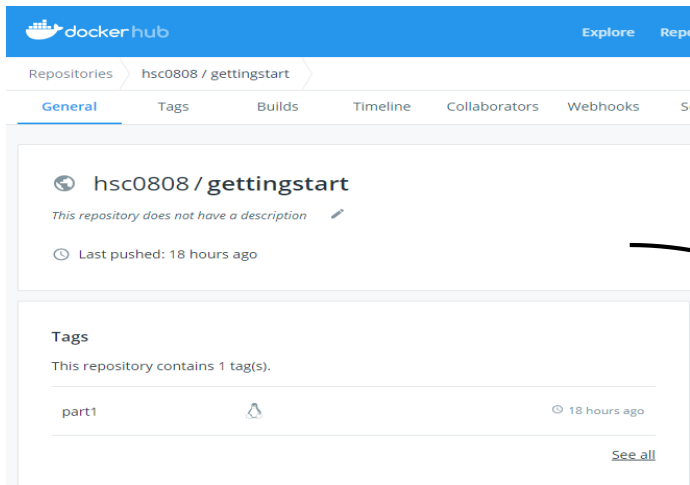
- 자원 관리 및 확장
- Service 란?
  - Distributed Application 에서 각각의 기능을 하는 개개의 app
  - Ex) video webpage
    - Video streaming
    - Database
    - Front-hand 반응 등
    - 각각이 서비스
- Docker에서는 **docker-compose.yml** 파일을 통해 서비스의 자원 관리 및 확장 가능

# docker-compose.yml

```
4 version: "3"
5 services:
6   web:
7     image: hsc0808/gettingstart:part1
8     deploy:
9       replicas: 5
10      resources:
11        limits:
12          cpus: "0.1"
13          memory: 50M
14        restart_policy:
15          condition: on-failure
16      ports:
17        - "4000:80"
18      networks:
19        - webnet
20 networks:
21   webnet:
```

- Service: 이전에 push 한 도커 이미지 tag 와 mapping
- Replicas: 5 → 5개의 cpu 사용 10%, memory 50M 로 제한된 컨테이너 실행

# Docker:service



```
root@hs-desktop:/home/hs/docker_example/python_execute# docker pull hsc0808/gettingstart:part1
part1: Pulling from hsc0808/gettingstart
177e7ef0df69: Pull complete
f6b2167b8d5a: Pull complete
432b044db3f9: Pull complete
7356f8556c46: Pull complete
d24429c45008: Pull complete
345ffaafbc478: Pull complete
23933064fb4e: Pull complete
Digest: sha256:3f60aealf8755a11ccbff846771bea9f88410815008ca05ba14a9fb1f440322d
Status: Downloaded newer image for hsc0808/gettingstart:part1
root@hs-desktop:/home/hs/docker_example/python_execute# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hsc0808/gettingstart	part1	58cd062ea0ce	44 hours ago	131MB

# Docker:service

```
root@hs-desktop:/home/hs/docker_example/python_execute# docker stack deploy -c docker-compose.yml getstartedlab
Creating network getstartedlab_webnet
Creating service getstartedlab_web
```

```
root@hs-desktop:/home/hs/docker_example/python_execute# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
a0bf587e6042	hsc0808/gettingstart:part1	"python app.py"	9 minutes ago	Up 9 minutes	80/tcp	getstartedlab_web.4.m6zwk62xln5658
g4wpj2i3ir1						
10386ede0d24	hsc0808/gettingstart:part1	"python app.py"	9 minutes ago	Up 9 minutes	80/tcp	getstartedlab_web.5.l96u9ubaergfic
x15pagphgl1						
9ca77d39a5fb	hsc0808/gettingstart:part1	"python app.py"	9 minutes ago	Up 9 minutes	80/tcp	getstartedlab_web.1.jnswg18b3md1k1
08m62yxap4j						
270ab666c2ea	hsc0808/gettingstart:part1	"python app.py"	9 minutes ago	Up 9 minutes	80/tcp	getstartedlab_web.3.n6z1ojfg6w13st
qh1tmny77kn						
0ca260cb402a	hsc0808/gettingstart:part1	"python app.py"	9 minutes ago	Up 9 minutes	80/tcp	getstartedlab_web.2.rs974fgsos1hv1
mnnjj0km462						

- Container 5개 실행 확인
- Localhost:4000 을 통해 5개 container id 가 번갈아 출력

# 실행화면

Hello World!

Hostname: 0ca260cb402a

Visits: *cannot connect to Redis, counter disabled*

Hello World!

Hostname: 10386ede0d24

Visits: *cannot connect to Redis, counter disabled*

Hello World!

Hostname: 9ca77d39a5fb

Visits: *cannot connect to Redis, counter disabled*



# Docker:swarm

- 현재 서버의 확장 진행중
  - 여러 대의 서버가 확장되어 관리 되고 있음.
  - 서비스를 효율적으로 관리하기 위해 서버의 역할을 부여
    - 서버가 늘어나고 컨테이너가 늘어날 때마다 관리가 어려움
    - Ex) 하나의 서버에 컨테이너가 몰릴 때
    - 역할이 모호한 애플리케이션의 경우 등
- Solution) Server orchestration 중 하나: Docker swarm
  - 여러대의 서버와 여러 개의 서비스를 편리하게 관리해주는 작업.
  - Ex) kubernetes, docker swarm 등

# 참고

- 도커 image 배포 및 실행, service 실습
  - <https://docs.docker.com/get-started/>
- 도커 기초 지식
  - <https://subicura.com/2017/01/19/docker-guide-for-beginners-2.html>