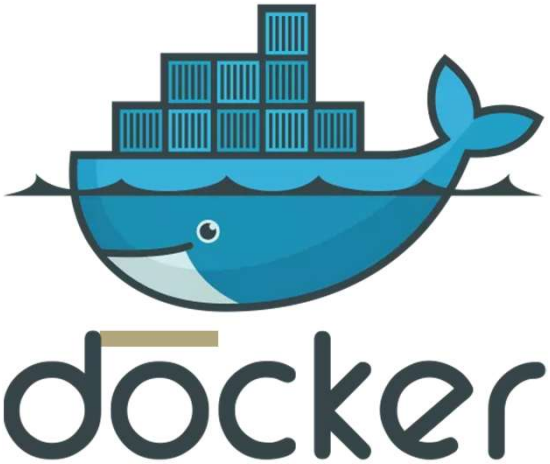


Docker

기본 개념

강사 : 신 인 호 교수

Docker 개념

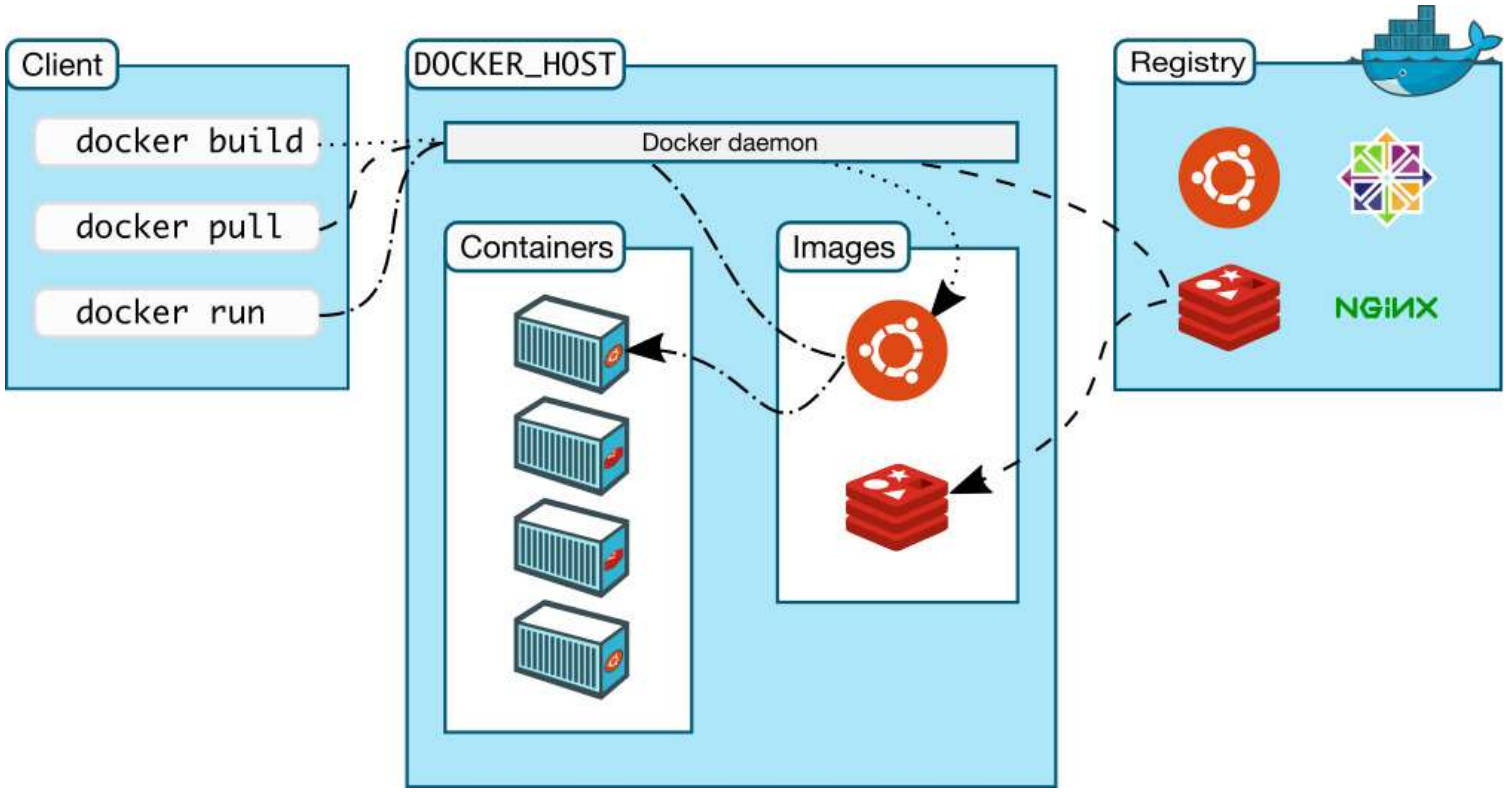
- 도커Docker란 작은 운영체제를 포함한 가상화 기술을 의미
- 도커에서 사용되는 작은 컴퓨터를 컨테이너Container라고 부른다.
- 도커는 Linux OS상에 오픈 소스 가상화 플랫폼으로, 애플리케이션 및 그에 따른 환경을 격리된 컨테이너에 패키징하여 개발, 배포, 실행을 쉽게 지원하는 툴이다.
- 도커는 가상 컴퓨터와 거의 비슷한 기능을 갖고 있으면서도 훨씬 가볍게 생성하고 운영

Docker 는.

https://docs.docker.com/get-started/

- PyCon 2013 Solomon Hykes - 'The Future of Linux Container'
- Made by Google's Go Language
- Immutable(불변), Stateless(무상태), Scalable(확장가능)
- Community Edition and Enterprise Edition ($1000 per year)
- Linux Base*
- 64-bit OS only

Docker 구성요소

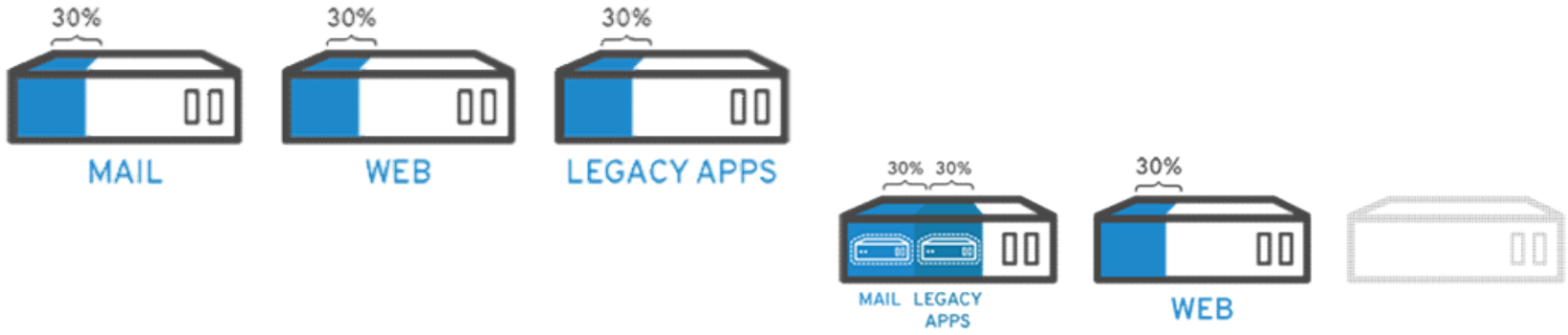

Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGINX

Docker 구성요소

- **Docker Client** -> 도커를 설치하면 그것이 Client며 build, pull, run 등의 도커 명령어를 수행합니다.
- **DOCKER_HOST** -> 도커가 띄워져있는 서버를 의미합니다. DOCKER_HOST에서 컨테이너와 이미지를 관리하게 됩니다.
- **Docker daemon** -> 도커 엔진입니다.
- **Registry** -> 외부(remote) 이미지 저장소입니다. 다른 사람들이 공유한 이미지를 내부(local) 도커 호스트에 pull할 수 있습니다. 이렇게 가져온 이미지를 run하면 컨테이너가 됩니다.

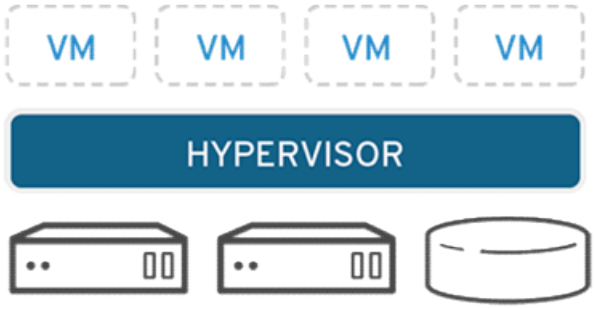
가상화(Virtualiztion)

- 가상화는 전통적으로 하드웨어에 종속된 리소스를 사용해 유용한 IT 서비스를 만들 수 있는 기술입니다
- 가상화를 사용하면 물리적 머신의 기능을 여러 사용자 또는 환경에 배포해 물리적 머신을 최대한 활용할 수 있습니다.



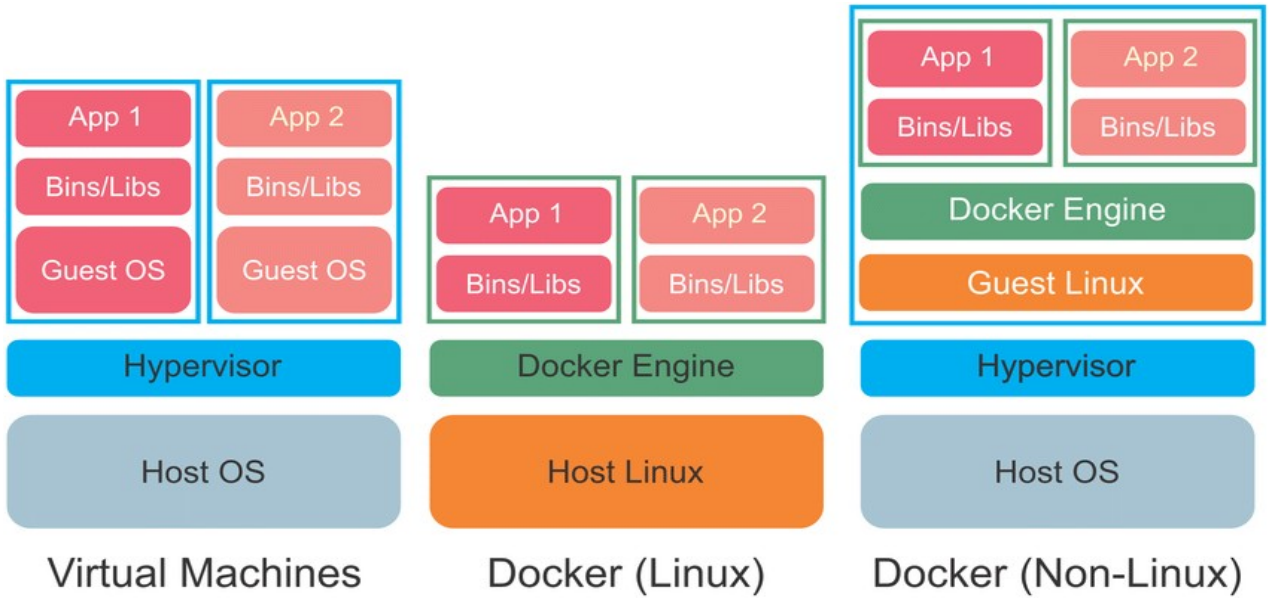
가상화 가상 머신(Virtual Machine)

- 리소스는 필요에 따라 물리 환경에서 여러 가상 환경으로 파티셔닝됩니다.
- 사용자가 가상 환경(일반적으로 게스트 머신 또는 가상 머신이라고 함)과 상호 작용하고 가상 환경 내에서 계산을 실행합니다.
- 하이퍼바이저는 가상 머신(VM)을 생성하고 구동하는 소프트웨어이고, 물리 리소스를 필요로 하는 가상 환경으로부터 물리 리소스를 분리 역할을 합니다



도커와 가상머신(Virtual Machine)

- Docker는 Linux 시스템을 사용하고 프로세스 격리를 통해 응용 프로그램 및 종속성을 모듈 식 Container로 패키지 한다.
- 이로 인해 VM이 필요하지 않으며 App1과 App2의 OS 리소스는 서로 공유된다. 이로 인해 CPU나 Memory를 딱 필요한 만큼만 할당하게 되고 성능의 손실이 거의 발생하지 않는다.



Docker 용어

Docker Image and Container

Docker Engine (Docker Daemon, Core)

Docker Client

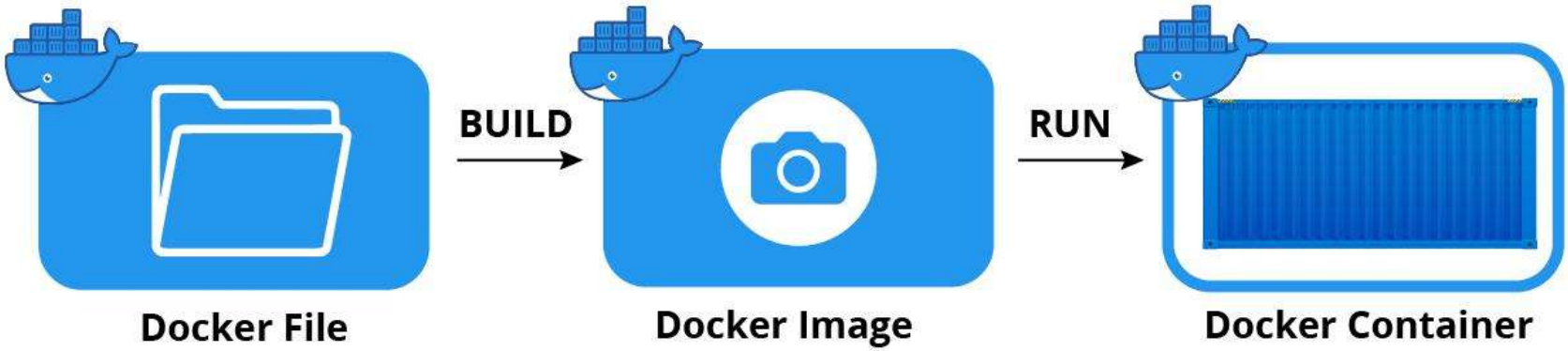
Docker Host OS

Docker Machine (Runtime Environment): 가상 호스트에 Docker Engine을 설치하고, 호스트를 관리할 수 있는 도구.

Docker Compose

Docker Registry, Hub, Swarm

Docker Image and Container



도커 엔진에서 사용하는 기본 단위는 이미지와 컨테이너이며 도커 엔진의 핵심이다.

도커 이미지와 컨테이너는 1:N 관계입니다.

Docker Image

도커 이미지(Docker Image)는 컨테이너를 생성할 때 필요한 요소이며, 가상 머신을 생성할 때 사용하는 iso 파일과 비슷한 개념입니다.

이미지는 컨테이너를 생성하고 실행할 때 읽기 전용으로 사용되며 여러 계층으로 된 바이너리 파일로 존재합니다.

이미지 이름 구성 :
[저장소 이름]/[이미지 이름]:[태그]
Shinho/hadoop3.4.0:latest

Docker Image

Container based on Docker Image

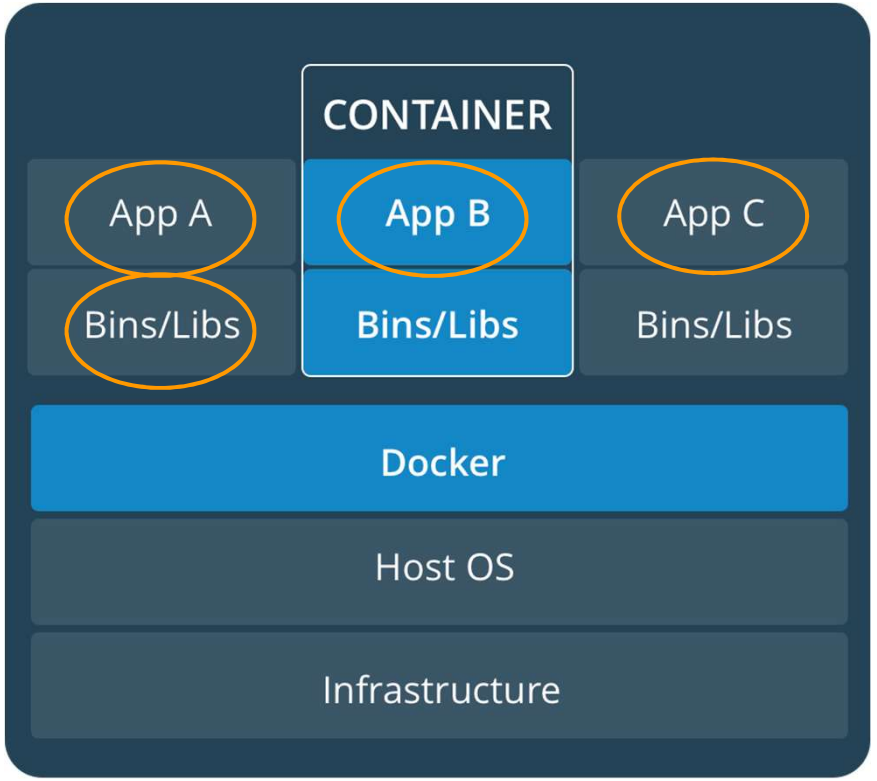
App, Libs, Middleware, OS, NW, etc

ex. MySQL image, Ubuntu image, etc

Image **Build** (Make) : Dockerfile

Image **Ship** (Share)

Docker Hub



Docker Container

- 도커 컨테이너(Docker Container)는 도커 이미지로 생성하고, 해당 이미지의 목적에 맞는 파일이 들어 있는, 호스트와 다른 컨테이너로부터 격리되어 독립된 공간(프로세스)이 생성된다.
- 컨테이너는 생성시에 이미지를 읽기 전용으로 만 사용하고, 이미지에서 변경된 사항만 컨테이너에 저장되므로 컨테이너에서 무엇을 하든지 원래 이미지에는 영향을 주지 않는다.

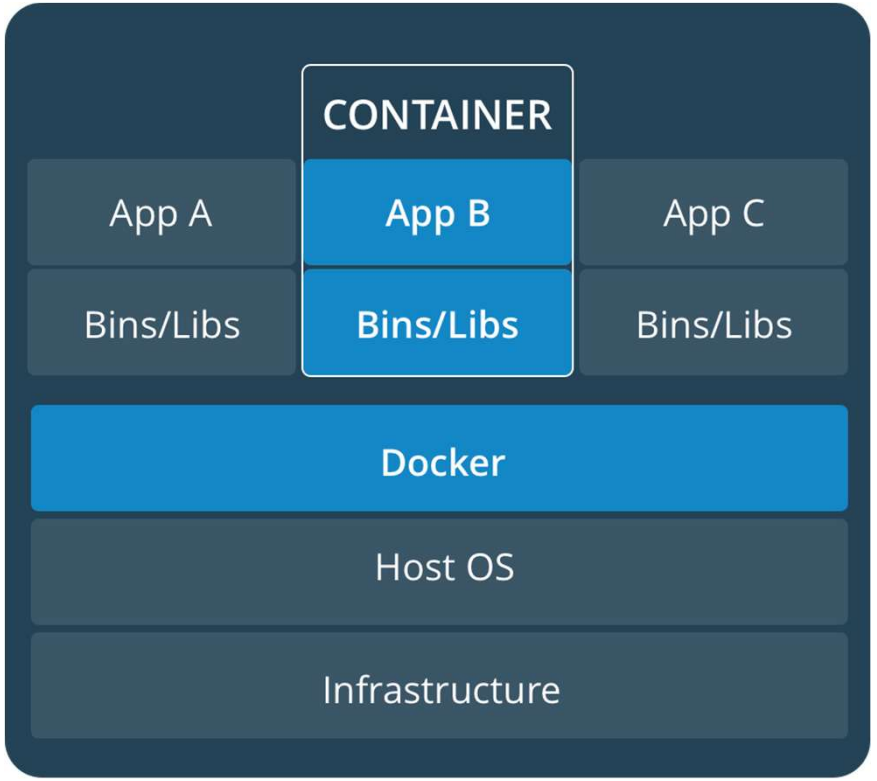
Docker Container

Logical Area in OS(Docker Host OS)
- Process, Network, FS

Like Each Server

Directory, Libraries and IP Shared

cf. VM(Virtual Machine)은 별도의 OS



Docker 설치하기 —

강사 : 신 인 호 교수

Install Docker in Windows

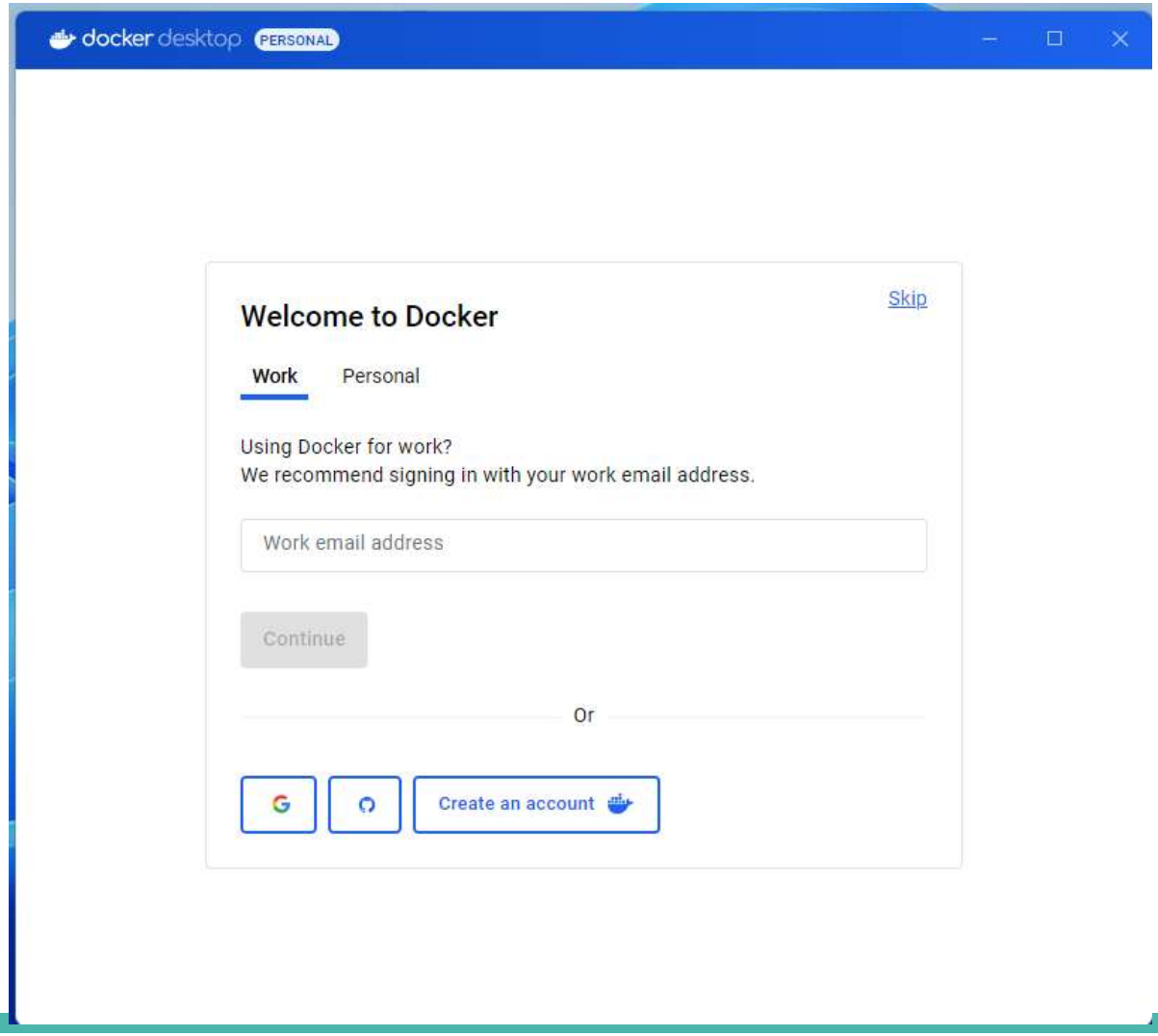
Windows 10

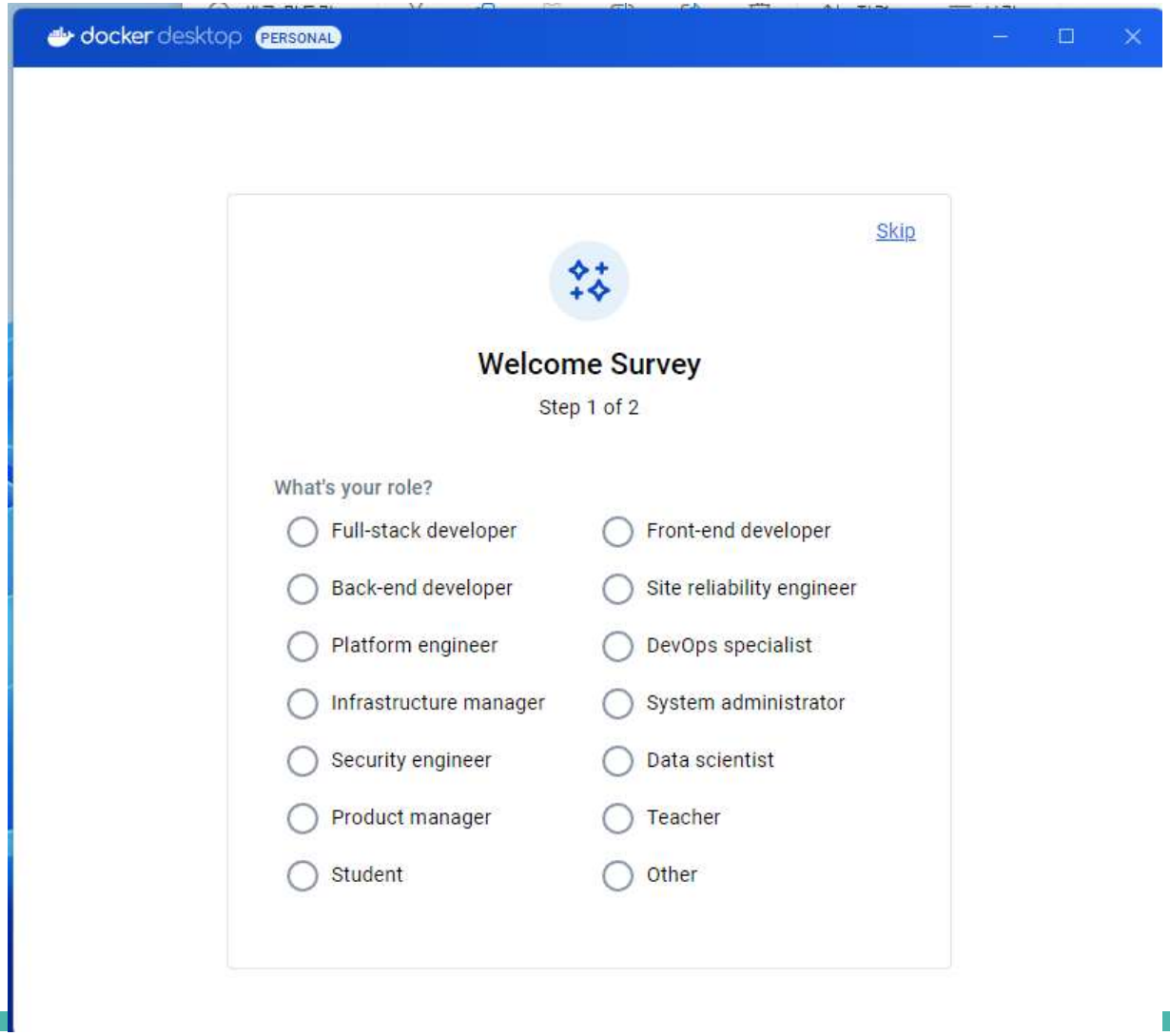
- Need Docker for Windows Only
- 작업관리자 > 성능 > 가상화

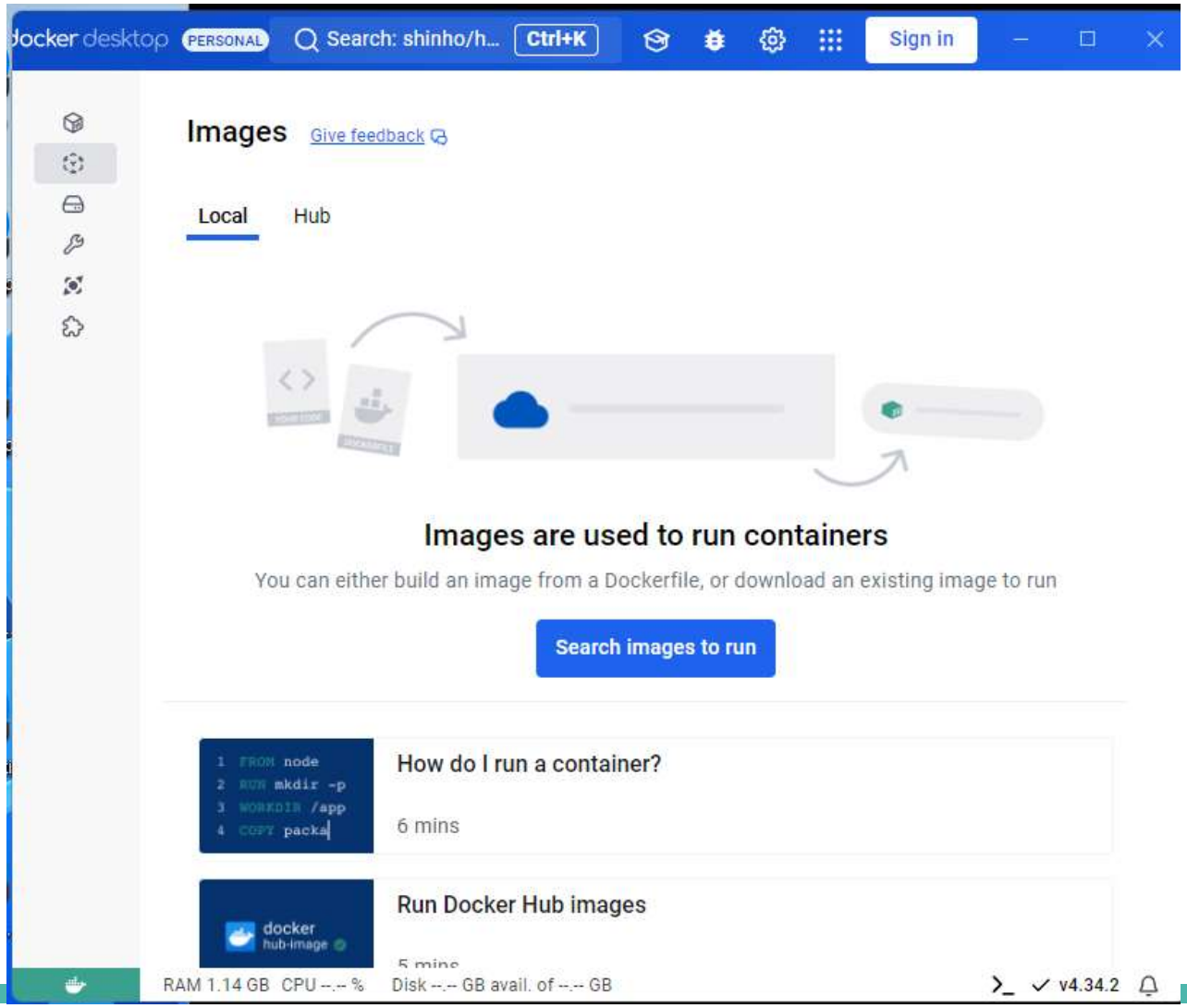
Under Windows 10 (Windows 7)

- Need Docker ToolBox (based on Oracle VirtualBox - Linux VM)
- Check Virtualization Tool

cf. Docker for MAC (By docker account) ⇒ ~/.docker/bin을 PATH에 연결!!


docker desktop
PERSONAL
Skip
Welcome to Docker
Work
Personal
Using Docker for work?
We recommend signing in with your work email address.
Work email address
Continue
Or
Create an account


docker desktop
PERSONAL
Skip
Welcome Survey
Step 1 of 2
What's your role?
Full-stack developer
Front-end developer
Back-end developer
Site reliability engineer
Platform engineer
DevOps specialist
Infrastructure manager
System administrator
Security engineer
Data scientist
Product manager
Teacher
Student
Other


docker desktop
PERSONAL
Search: shinho/h...
Ctrl+K
Sign in
Images
Give feedback
Local
Hub
Images are used to run containers
You can either build an image from a Dockerfile, or download an existing image to run
Search images to run
1 FROM node
2 RUN mkdir -p
3 WORKDIR /app
4 COPY packa
How do I run a container?
6 mins
docker
hub-image
Run Docker Hub images
RAM 1.14 GB CPU --.-- %
Disk --.-- GB avail. of --.-- GB
v4.34.2

Install Docker in Windows

 # wsl 설치 참조

https://www.howtogeek.com/744328/how-to-install-the-windows-subsystem-for-linux-on-windows-11/#how-wsl-works-on-windows-11

```
wsl --install
```

Docker

기본 명령어 —

강사 : 신 인 호 교수

도커 설치 확인

```
# Run Docker Quick Starter

$ docker version        # cf. docker --version
$ docker info           # docker system info
$ docker --help
$ docker run hello-world
```

Docker 활용

1. **Image down**

   Docker [pull | push] ubuntu:latest

2. **이미지를 이용하여 container 생성 및 실행**

   docker [start | stop | restart] <container-name>

3. **Image, container관리**

   docker images, docker **ps**

4. **프로세스 관리**

   docker **ps, docker ps -a**

5. **hub이미지 올리기**

Docker Image downLoad 및 생성

```
# 이미지 다운로드
$> Docker [pull | push ] ubuntu:latest(이미지이름:tag)
$> docker pull <docker-image-name[:tag-name]>

# 이미지 생성
$> docker image bulid -t 이미지명[:태그명] Dockerfile경로
$> docker commit 이미지이름:tag # 실행한 컨테이너를 이미지로 생성
```

https://hub.docker.com

Docker 이미지 명령어

```
docker image <command> [image-name]

$> docker image [pull | push] # image 가져오기|허브에 저장
$> docker image ls            # Local image 목록
$> docker image inspect nginx # image 세부정보 확인
$> docker search python       # 이미지 검색
$> docker image rm nginx      # nginx image 삭제
$> docker login               # login to docker hub
$> docker logout


$> docker image prune         # 사용하지 않는 모든 image 삭제!
```

Docker container 생성 및 실행

```
$> docker container run --name <container> -d -p 80:80 <image>
$> docker [start | stop | restart] <container-name>
$> docker container [pause | unpause] <container-name>
```

https://hub.docker.com

Docker Container 옵션

옵션	설명
-i	표준 입력(stdin)을 활성화 보통 이 옵션을 사용하여 Bash 에 명령을 입력한다.
-t	TTY 모드(pseudo-TTY)를 사용 Bash를 사용하려면 이 옵션을 설정해야 한다. 이 옵션을 설정하지 않으면 명령을 입력할 수는 있지만, Shell이 표시되지 않는다.
-d	데몬 프로세스로 실행(백그라운드)해 프로세스가 끝나도 유지
--name {이름}	컨테이너 이름 지정
-p	호스트와 컨테이너의 포트를 연결 - 외부 접근 설정 (도커 컨테이너는 기본적으로 외부에서 접근할 수 없게 구성되어 있다.)
--rm	컨테이너가 종료되면{내부에서 돌아가는 작업이 끝나면} 컨테이너를 제거
-v	호스트와 컨테이너의 디렉토리를 연결

https://hub.docker.com

Docker Container 실습

```
$> docker pull nginx  $/usr/share/nginx/html
$> docker container run --name webserver -d -p 80:80 nginx
$> docker container ps # 실행 중 컨터이너, -a :전체
$> docker container ls # 컨터이너 목록
# 컨테이너 시작, 접속, 정지, 삭제
$> docker container [start|attach|stop|rm] 컨테이너 이름
```

https://hub.docker.com

Docker Container 실습

```
# 실행중 컨테이너 상태 모니터링
$> docker container [top | stats] webserver #실행 중 컨테이너 상태
# 컨테이너 삭제.
$> docker container rm webserver # rm -f 강제 삭제
# 사용하지 않은 이미지, 컨테이너 삭제.
$> docker container prune
$> docker system df # 디스크 사용상태
```

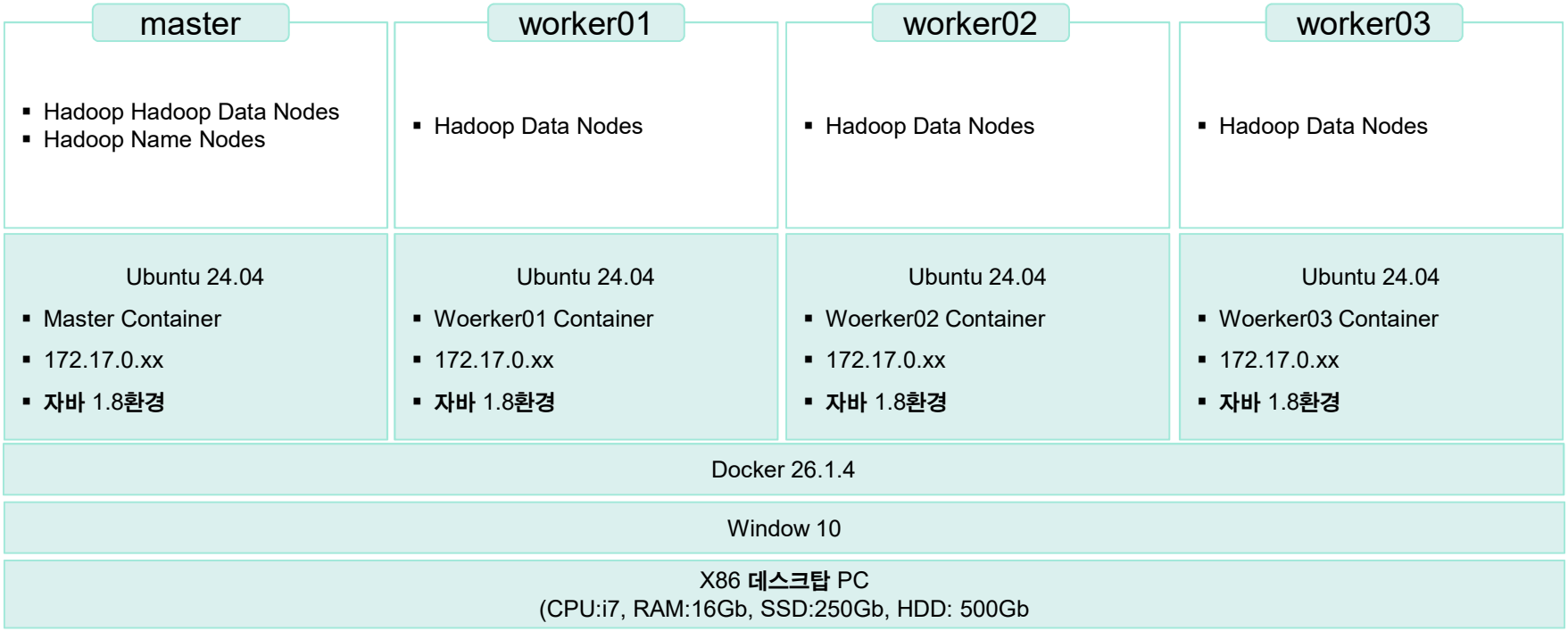
Docker

Hadoop2.4.0 설치

강사 : 신 인 호 교수

1. Hadoop 구성도

https://hub.docker.com



1. Hadoop 및 관련 프로그램 설치

https://hub.docker.com

1. Hadoop Image 다운로드 : Window CMD 창에서 실행

```
C:\>docker pull shinho/hadoop3.4.0
```

1. Hadoop 및 관련 프로그램 설치

https://hub.docker.com

2. Hadoop Image로 각 노드 생성(Container): Window CMD 창에서 실행

```
C:\> docker run -it -h master --name master -p 9870:9870
shinho/hadoop3.4.0

C:\> docker run -it -h worker01 --name worker01 --link
master:master shinho/hadoop3.4.0
C:\> docker run -it -h worker02 --name worker02 --link
master:master shinho/hadoop3.4.0
C:\> docker run -it -h worker03 --name worker03 --link
master:master shinho/hadoop3.4.0
```

2. Hadoop 환경 파일 수정 및 실행 https://hub.docker.com

3. 각 노드의 할당 된 IP확인 : Window CMD 창에서 실행

```
C:\> docker inspect master   | findstr "IPAddress"
C:\> docker inspect worker01 | findstr "IPAddress"
C:\> docker inspect worker02 | findstr "IPAddress"
C:\> docker inspect worker03 | findstr "IPAddress"
```

2. Hadoop 환경 파일 수정 및 실행 https://hub.docker.com

1. Vi /etc/hosts 파일 수정 : master에서 수정한다

```
172.17.0.2 master
172.17.0.3 worker01
172.17.0.4 worker02
172.17.0.5 worker03
```

- IP주소는 각 노드별로 앞에서 확인 한 주소를 입력한다.

Guest OS 현행화

apt update

apt upgrade

2. Hadoop 환경 파일 수정 및 실행 https://hub.docker.com

2. ssh 서비스 실행 및 확인 : 각 노드별로 실행한다.

-. 각 노드(master, worker01,worker02,worker03) bash shell 이용
service ssh restart

-. 접속확인
ssh worker01
ssh worker02
ssh worker03

2. Hadoop 환경 파일 수정 및 실행 https://hub.docker.com

3. 하둡 실행을 위한 초기화(format) 밍 실행

- Master server에서 실행한다.

```
$ cd

$ stop-all.sh
$ Hadoop_init
$ start-all.sh
```

- Browser에서 실행한다.

http://localhost:9870

3. Hadoop 실행 및 확인

https://hub.docker.com

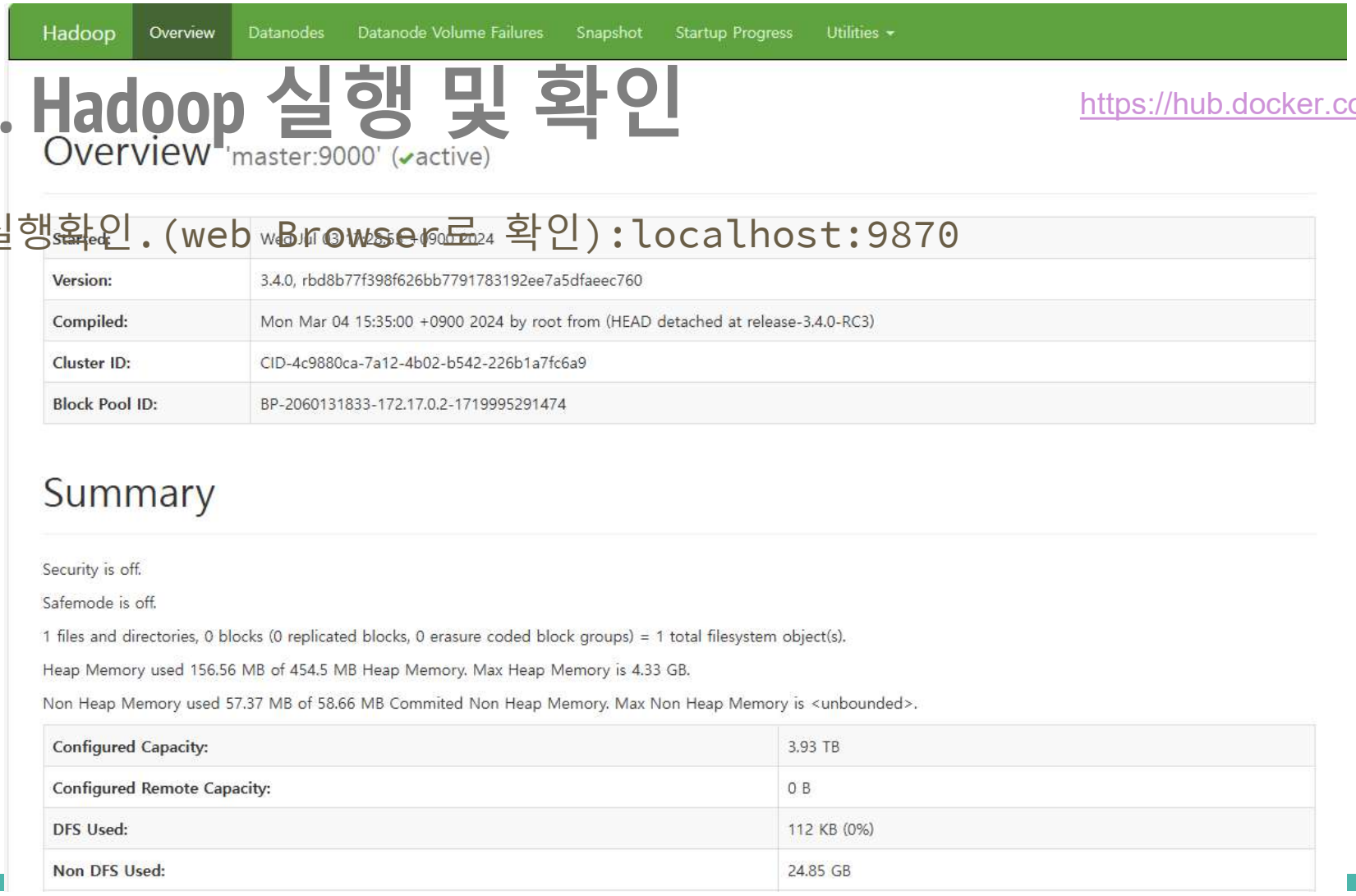
2. 실행확인

```
$ jps
1123 ResourceManager
581 DataNode
839 SecondaryNameNode
1355 NodeManager
1806 Jps
351 NameNode
```

3. Hadoop 실행 및 확인

https://hub.docker.com

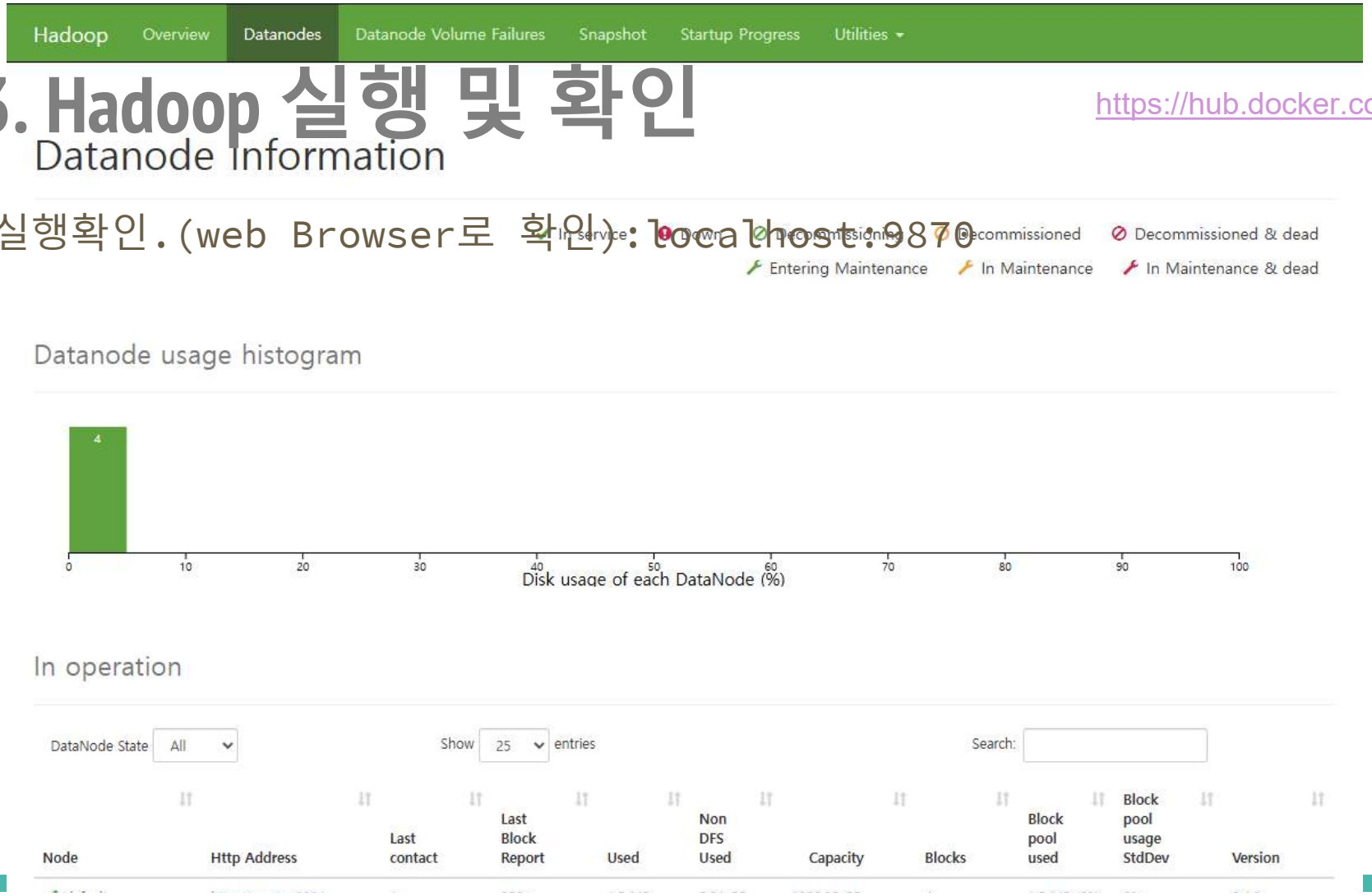
3. 실행확인.(web Browser로 확인):localhost:9870

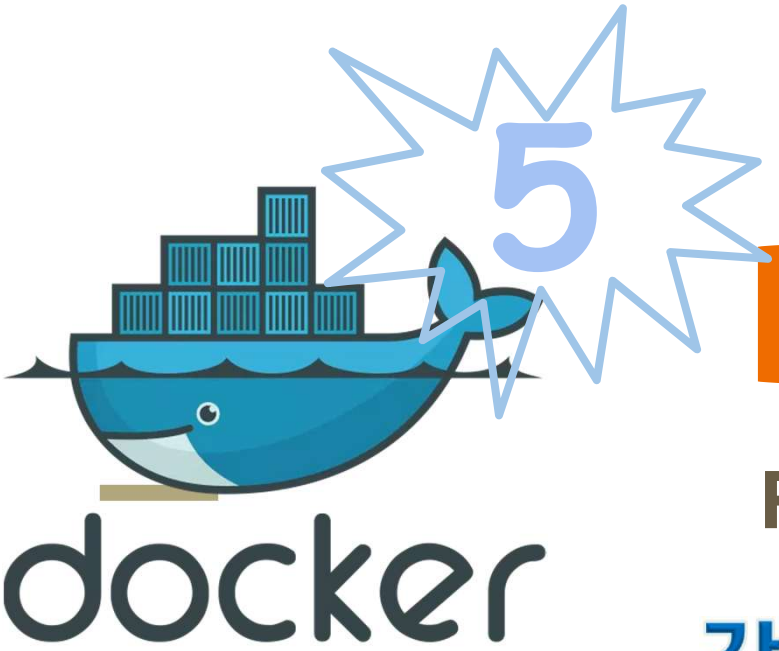


3. Hadoop 실행 및 확인

https://hub.docker.com

3. 실행확인.(web Browser로 확인): localhost:9870





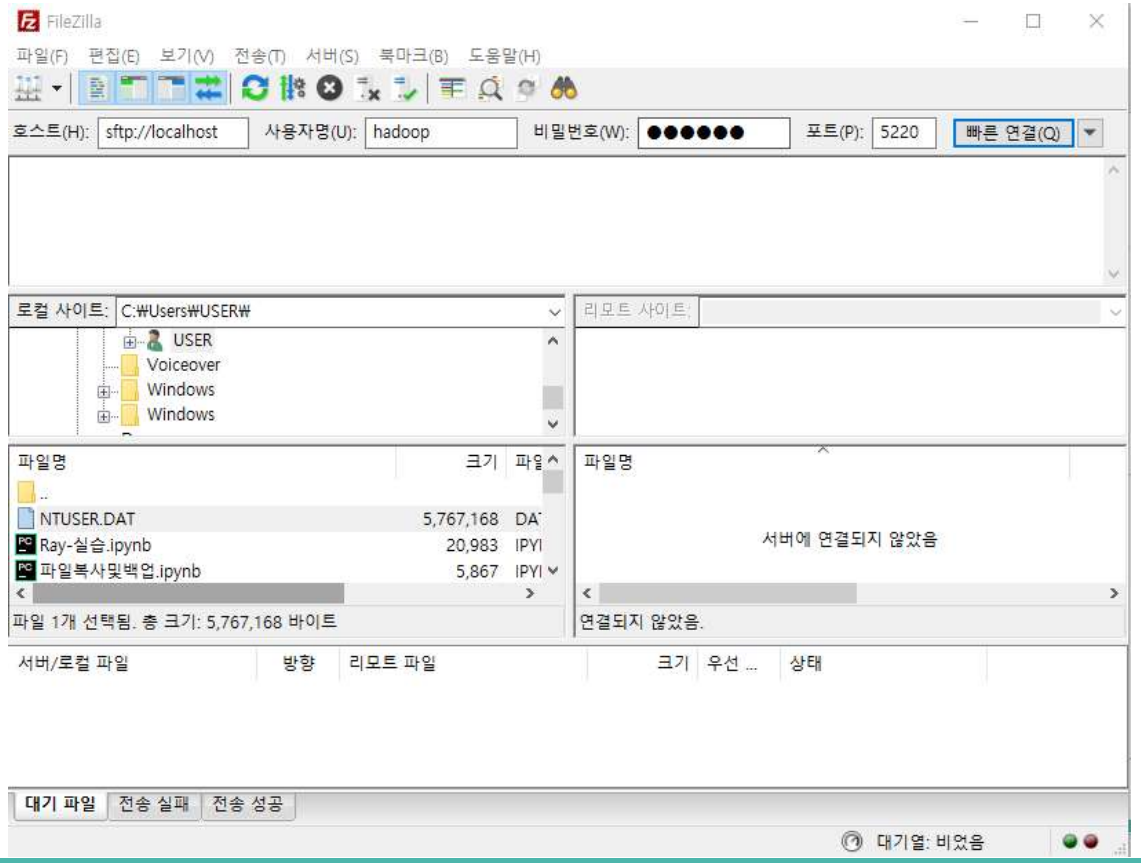
Docker

File Sharing

강사 : 신 인 호 교수

File Copy & Share : FTP사용

- **Host :**
  `sftp://loclahost`
- **계정/암호 :**
  `hadoop / hadoop`
- **Port :**
  `5220`



File Copy & Share : Docker 명령어

docker [container] cp <container-name>:<path> <client-path>

docker [container] cp <client-file> <container-name>:<path>

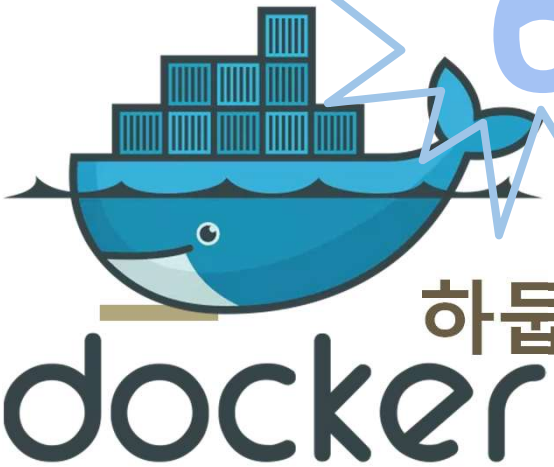
```
$> docker cp mysql5:/backup/dump.sql  .

$> docker cp dump.sql mysql5:/

# Share Directory

$> docker run -v <localpath>:<container-path>
```

cf. $> docker stop `docker ps -q`



6

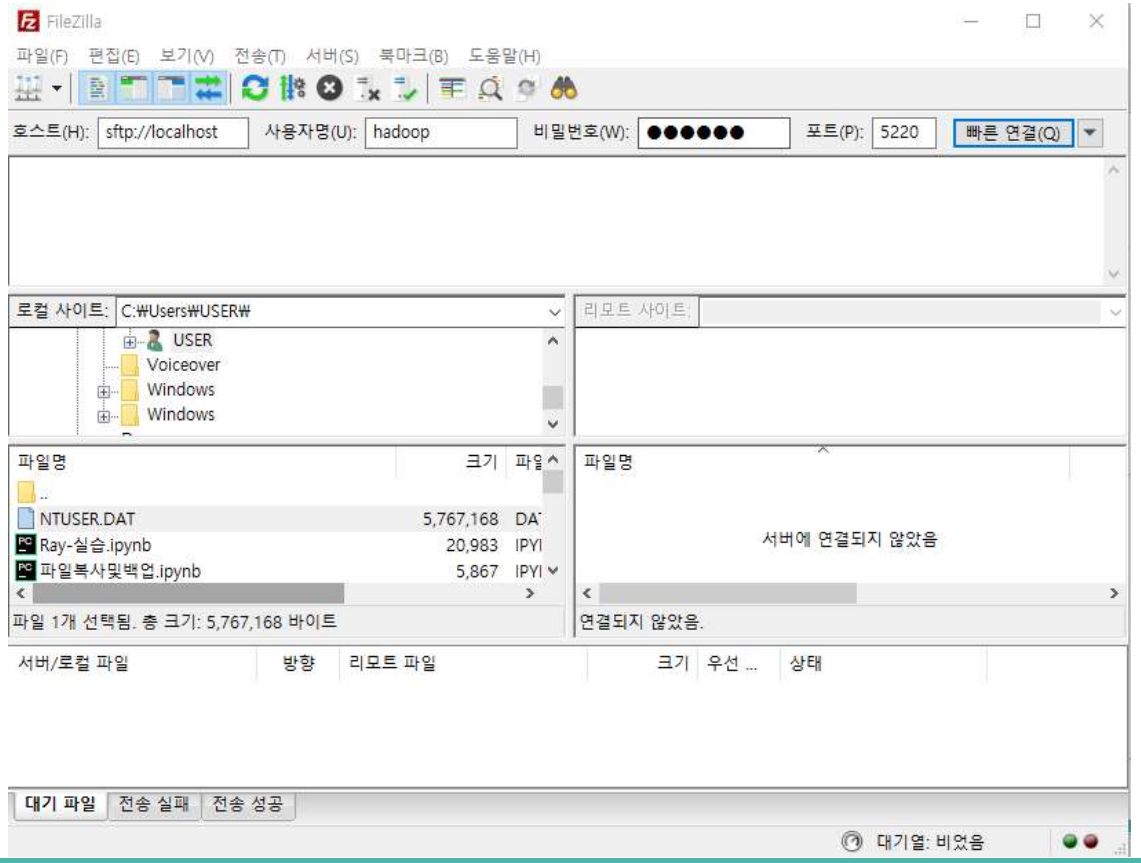
Docker

하둡 Wordcount 실습

강사 : 신 인 호 교수

File Copy & Share : FTP사용

- **Host :**
  `sftp://loclahost`
- **계정/암호 :**
  `hadoop / hadoop`
- **Port :**
  `5220`



1. 분석 관련 파일 복사 : master

https://hub.docker.com

1. Wordcount 관련 파일 master에 Upload:

- Master Directory 생성(분석데이터) Ftp창 에서 실행

```
$ mkdir ./tmp_data
```

- 프로그램 Upload : mapreduce-0.0.1-SNAPSHOT.jar
- 분석용data Upload: alice.txt. Holmes.txt, frankenstein.txt

- 분석할 데이터 master에서 하둡에 Upload

```
Hdfs dfs -put ./tmp_data/* /aysdat/
```

-. Hdfs dfs -put -> 하둡에 분석데이터 적제 명령
-. tmp_data/*.txt -> master 노드에 있는 분석데이터
-. /aysdat/ -> 하둡에 있는 분석데이터

2. Wordcount 프로그램 실행

https://hub.docker.com

1. Wordcount 명령어 실행 : master에서

-. yarn jar mapreduce-0.0.1-SNAPSHOT.jar
bigdata.mapreduce.WordCount /aysdat/*.txt /wcnt

-. 명령어설명
1.yarn jar : 하둡에서 jar파일 실행 명령
2.mapreduce-0.0.1-SNAPSHOT.jar : WordCount 프로그램 패키지
3.bigdata.mapreduce.WordCount : 실행할 클래스 파일
4./aysdat/*.txt : 분석할 데이터 위치
5./wcnt : 분석결과를 보관할 위치

3. Wordcount 결과 확인

https://hub.docker.com

1. Wordcount 분석 결과 : 내 컴퓨터에서 확인

-. 하둡에서 master로 가져오기

```
Hdfs dfs -get /wcnt/part-r-00000
```

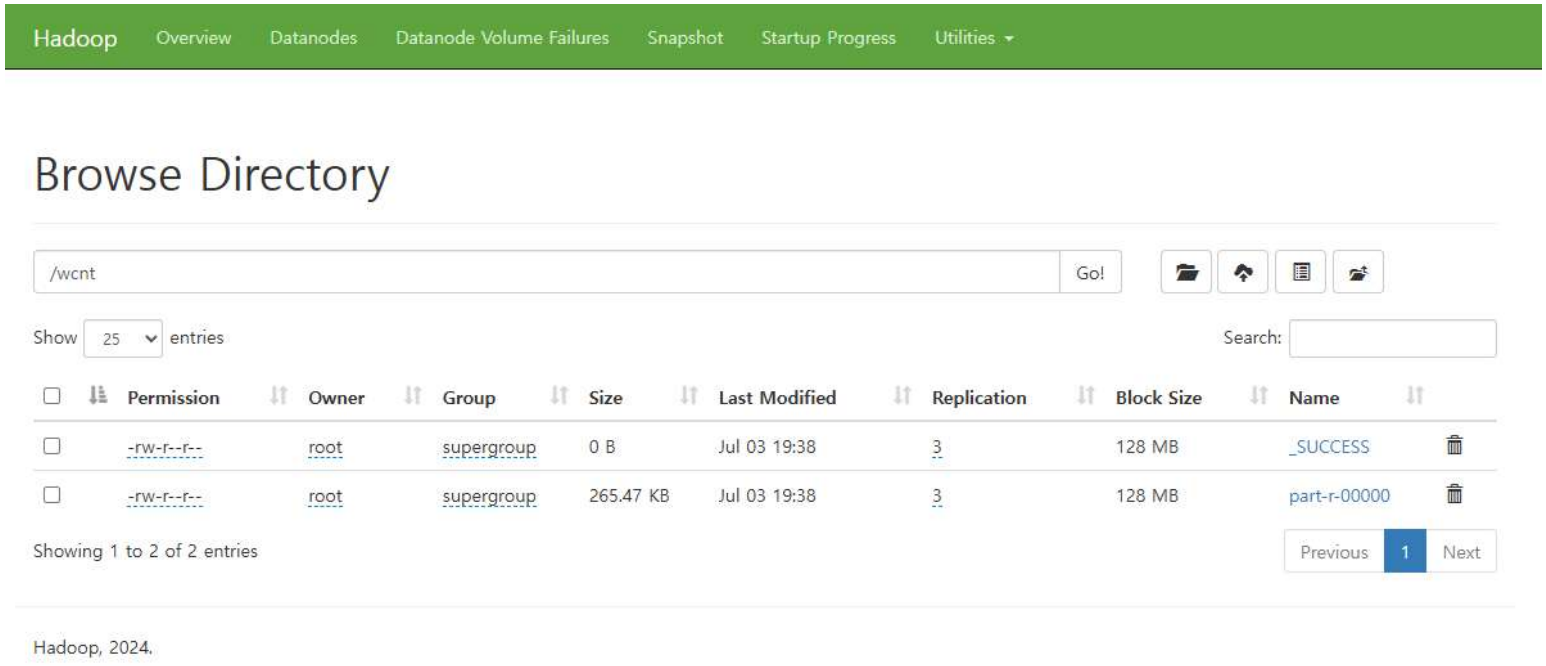
-. 내 컴퓨터로 가져오기 - Ftp창 에서 실행

```
part-r-00000
```

3. Wordcount 결과 확인

https://hub.docker.com

2. Wordcount 분석 결과 : web에서 확인



수고하셨습니다.