Docker기반 DevOps 인프라 구축

1일차 / 황상철

강의 개요

- 강의제목: Docker 기반 DevOps 인프라 구축
- 대상: 도커 초보자, DevOps 초보자
- 교재: 실습자료(하드카피), 강의자료
- 진행방식: 강의(60%), 실습(40%)
- 프리모텀
  - 도커가 무엇인지 설명할 수 있다.
  - 현재 참여중인 프로젝트에서 활용할 수 있을거 같다.
  - 도커기반 DevOps 시스템 구축에 대한 감이 잡힌다.

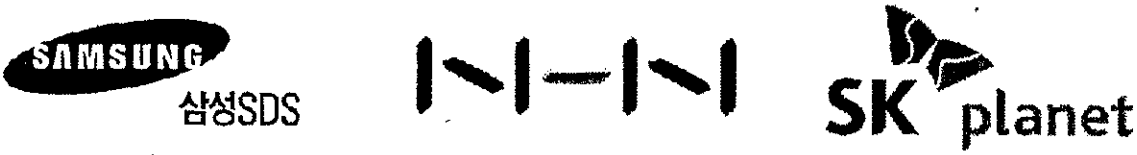
강의 구성

Day	레벨	내용	시간	세부 내용
1 일차	기본	처음 배우는 도커	1	Docker 이해 및 환경구성
			2	Docker 이미지
			3	Docker 컨테이너
		Docker로 꾸미는 로컬 환경	4	Docker 머신
			5	컨테이너 활용
			6	Docker Compose
2 일차	활용	Docker로 꾸미는 프로젝트 환경	1	Docker Hub와 Registry
			2	개발 환경 구성
			3	도커 이미지 & 환경구성
		Docker 멀티 호스트 구성	4	볼륨과 네트워킹
			5	Docker Swarm
			6	SwarmKit과 Moby

강의 구성

Day	레벨	내용	시간	세부 내용
3 일차	기본	Docker 운영환경	1	Docker Security
			2	Docker 모니터링
			3	Redis 클러스터
		DevOps와 도커	4	Container Linux 환경구성
			5	DevOps 환경을 위한 시스템
			6	Docker 오케스트레이션 도구

강사를 소개합니다.



9.7 + 3 + 3.5

= 16.2

책을 좋아하는 개발자

DevOps 엔지니어, Agile 멘토, Java 개발자

espressobook.com 대표(2016.6~현재)

블로그: pragmaticstory.com

수강생 소개

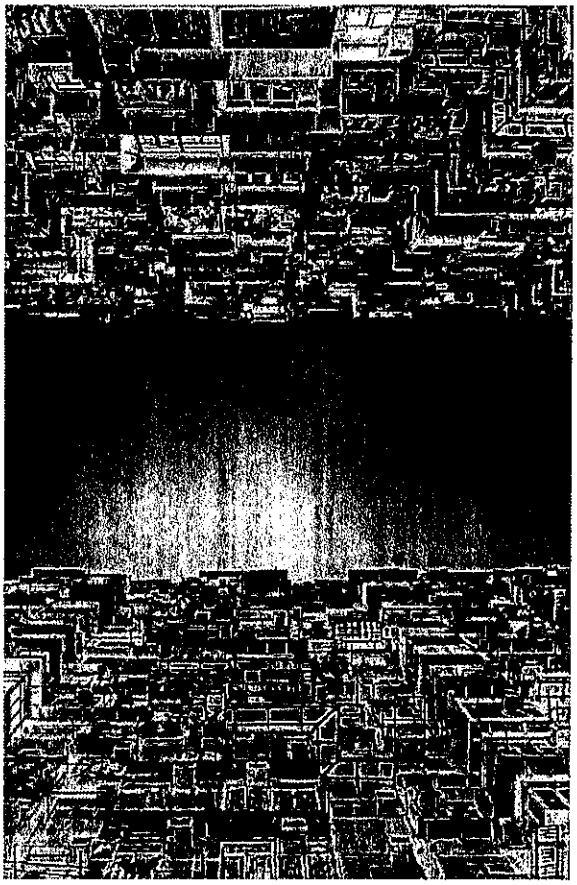
- Q1: 도커에 대한 지식,경험
- Q2: DevOps에 대한 지식,경험

저는 백엔드 개발자 황상철 입니다.
도커는 전혀 모르고, DevOps는 용어만 들어봤습니다.
이번 워크샵을 통해서 도커를 마스터 하고 싶습니다.

강의중 질문,실습 지원을 위한 슬랙 링크

https://fcdocker.slack.com

처음 만나는 Docker

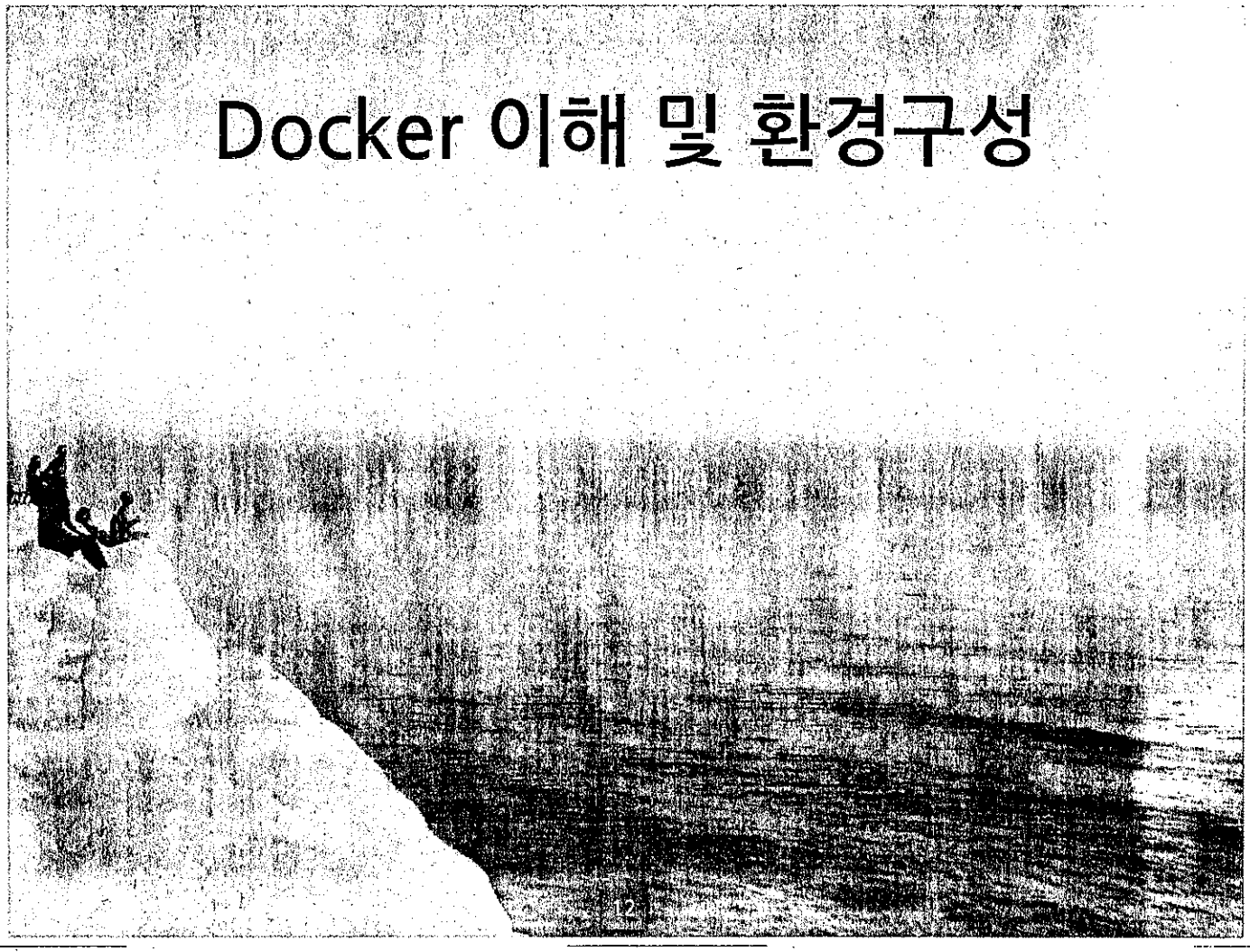


Docker 이해 및 환경구성

Docker 이미지

Docker 컨테이너

Docker 이해 및 환경구성



도커는 무엇인가
Introductions to Docker

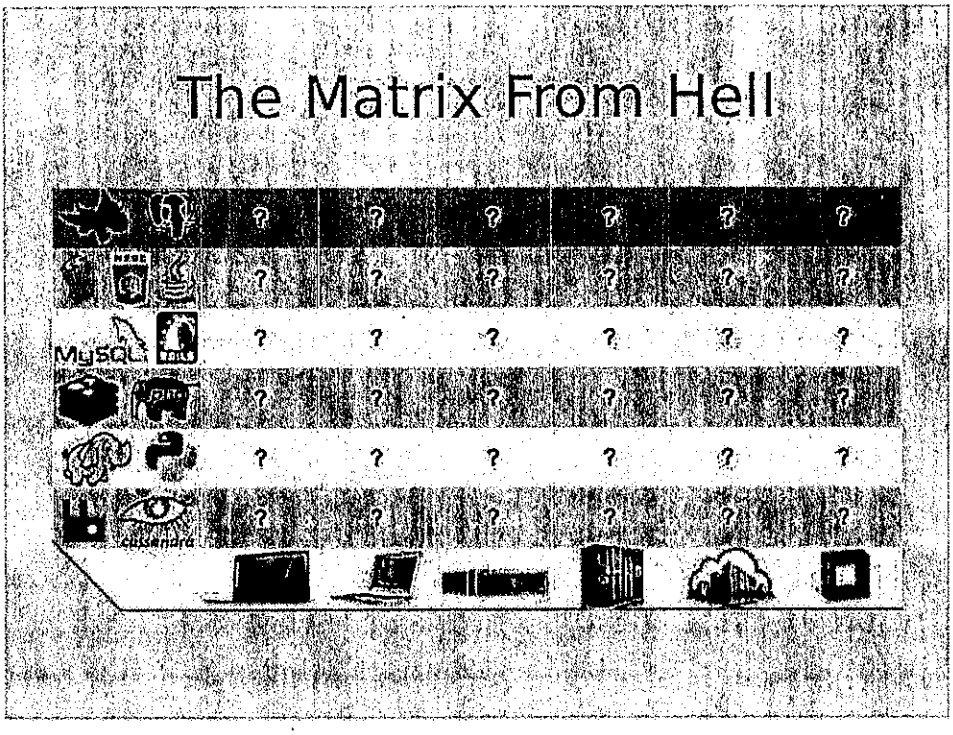
도커(Docker)

리눅스컨테이너 기술을 이용해 어플리케이션 패키징,
배포를 지원하는 경량의 가상화 오픈소스 프로젝트

- 최신버전: v17.03.0(2017.3)
- 라이센스: Apache License 2.0
- Go 언어로 개발: https://golang.org/

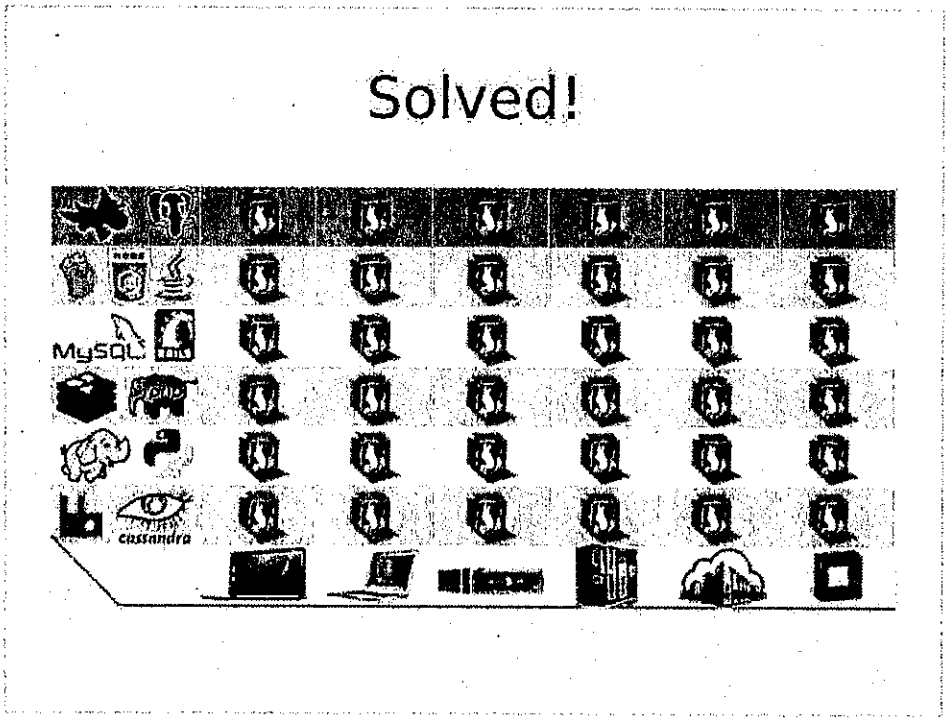
왜 도커를 써야 할까?

Matrix from Hell



http://www.slideshare.net/Docker/docker-lpc-2014cristian

Matrix from Hell



맥에서 아파치 톰캣 설치

1. Java 버전 확인

   $ java -version

2. 톰캣 다운로드

   http://tomcat.apache.org/

3. 폴더 생성 및 복사

   sudo mkdir -p /user/local

   sudo mv ~/Desktop/tomcat /user/local

   sudo rm -f /Library/Tomcat

   sudo ln -s /user/local/tomcat /Library/Tomcat

4. 권한 수정

   sudo chown -R <Id> /Library/Tomcat

   sudo chmod -x /Library/Tomcat/bin/*.sh

5. 톰캣 시작/중지

   sudo /Library/Tomcat/bin/startup.sh

   sudo /Library/Tomcat/bin/shutdown.sh

https://wolfpaulus.com/journal/mac/tomcat8/

윈도우에서 아파치 톰캣 설치

1. Java 버전 확인

   $ java -version

2. 톰캣 다운로드

   http://tomcat.apache.org/

3. 폴더 생성 및 복사

   c:/tomcat

4. 톰캣 시작/정지

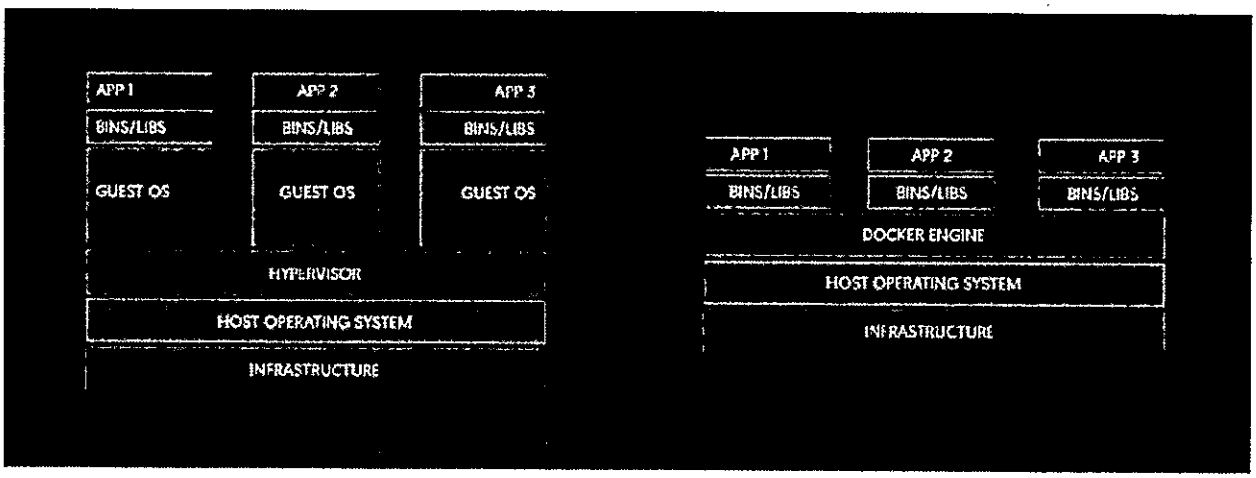
   c:/tomcat/bin/startup.bat

   c:/tomcat/bin/shutdown.bat

Docker로 아파치 톰캣 설치

$ docker run -d -p 8080:8080 tomcat:8.0

Container vs Hypervisor



도커는 가볍다

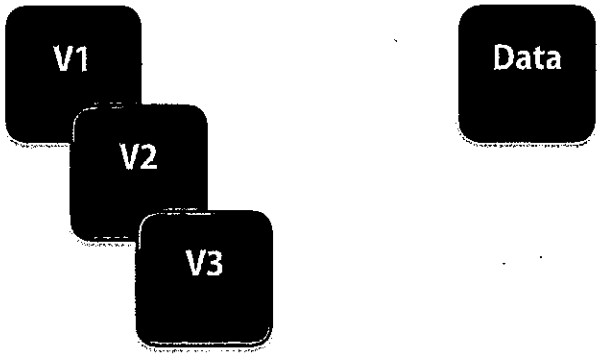
VM에 비해 이미지 파일 크기가 작아서 빠르게 이미지를 만들고 실행할수 있다.

```
Sangcheolui-MacBook-Pro:ebook-admin sangcheolhwang$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest              cf62323fa025        4 days ago          125 MB
centos              latest              05188b417f30        11 days ago         196.8 MB
alpine              latest              4e38e38c8ce0        2 weeks ago         4.799 MB
fedora              latest              f9873d530588        3 weeks ago         204.4 MB
debian              latest              1b088884749b        4 weeks ago         125.1 MB
```

Immutable Infrastruture

이미지 기반의 어플리케이션 배포 파라다임

많은 서버를 동적으로 관리하는 클라우드 환경에서 효과적이고 유연한 배포 방식



https://www.slideshare.net/continuousphp/lets-code-our-infrastructure

Immutable Infrastruture

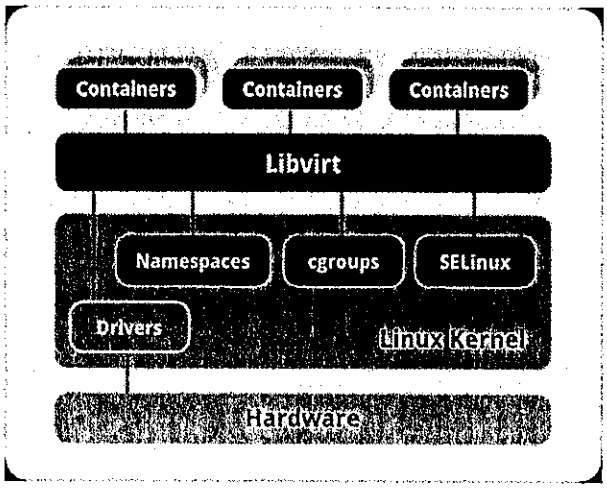
- 관리 가능하고
- Stateless 하고
- Scalabe 한
- 이미지 기반의
- 어플리케이션 배포

http://blog.nacyot.com/articles/2014-04-06-immutable-infrastructure/

리눅스 컨테이너
(LXC, Linux Container)

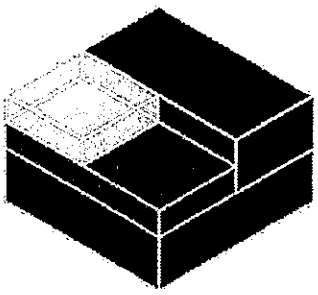
- 단일 리눅스 호스트상에서 여러개의 격리된(isolated) 리눅스 시스템을 실행하기 위한 OS 수준의 가상화

- Libvirt
  - 가상 호스트를 관리하는 툴킷
  - https://libvirt.org



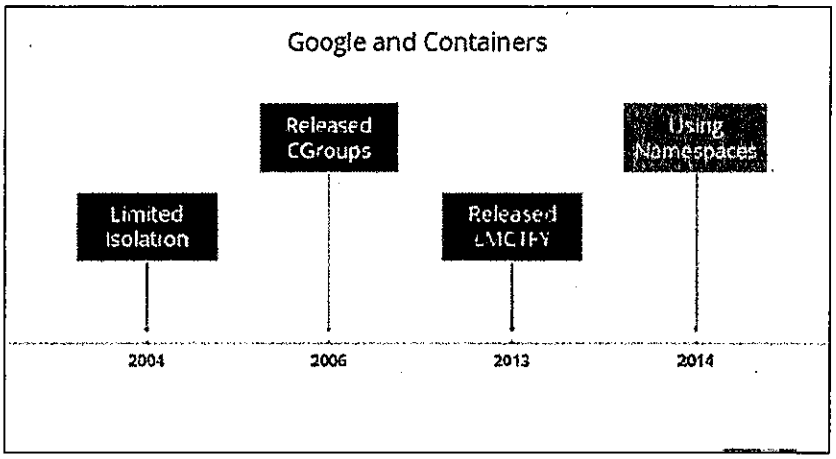
리눅스 컨테이너
(LXC, Linux Container)

- OpenVZ(openvz.org)
- Imctfy(google/Imctfy)
- Warden(cloudfoundry/warden)
- libContainer(docker 기본)
- libvirt, systemd



Everything at google runs in container

매주 20억개 컨테이너가 구동된다.(2014)



https://speakerdeck.com/jbeda/containers-at-scale

도커는 왜 성공했을까?

리눅스 컨테이너는 소개된지 10년이 지난 기술이지만 설정이 복잡하고 사용하기 어려워 널리 사용되지 못했음.

"도커는 사용하기 쉽다."

29

도커의 시작

dotCloud의 Solomon Hykes가 PyCon2013에서 'Hello World' 데모를 통해 처음 공개

* The future of Linux containers

https://www.youtube.com/watch?v=wW9CAH9nSLs

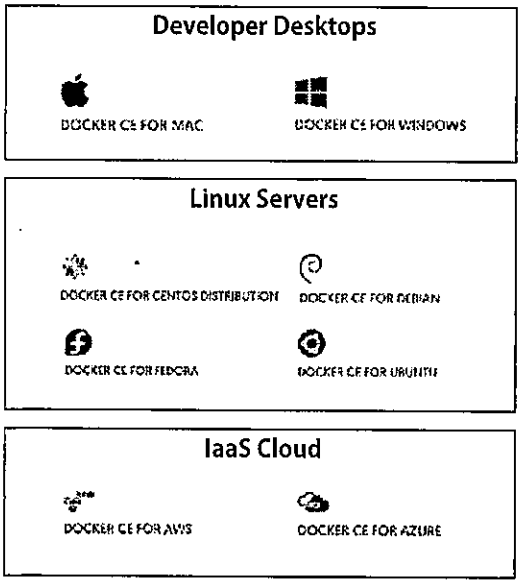
```
$ docker run busybox echo 'Hello World'
```

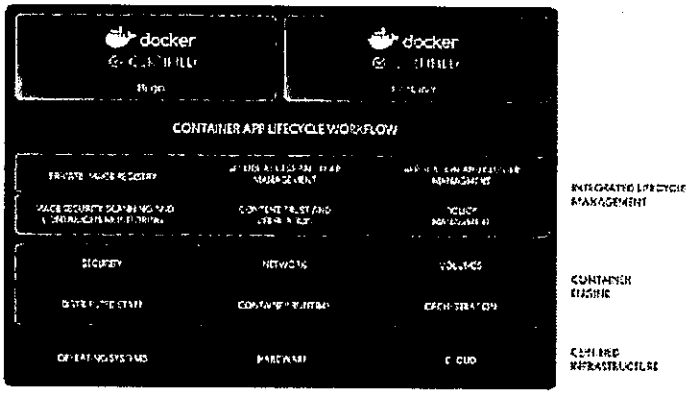
- 로컬에 busybox 이미지 확인
  - 없으면 dockerhub에서 다운로드
- busybox 이미지로 새 컨테이너 생성
- 컨테이너 실행
- TTY로 'Hello World' 출력
- 출력된 결과를 유닉스 소켓을 통해 클라이언트로 전송

도커 구성요소
Docker Components

Product

Community Edition **Enterprise Edition**





Docker CE 버전 특징

- 무료
- 최신 버전
- 클라이언트 위주(개발자, 리눅스, 클라우드)

https://www.docker.com/community-edition

	COMMUNITY EDITION	ENTERPRISE EDITION BASIC	ENTERPRISE EDITION STANDARD	ENTERPRISE EDITION ADVANCED
Container engine and built in orchestration, networking, security	✓	✓	✓	✓
Docker Certified Infrastructure, Plugins and ISV Containers		✓	✓	✓
Image Management (private registry, caching)	Cloud hosted repos		✓	✓
Docker Datacenter Integrated container app management			✓	✓
Docker Datacenter Multi-tenancy with RBAC, LDAP/AD support			✓	✓
Integrated secrets mgmt, image signing policy			✓	✓
Image security scanning	Preview			✓
Support	Community Support	Business Day or Business Critical	Business Day or Business Critical	Business Day or Business Critical
	FREE Docker Cloud for repos, automated builds, scanning as a service	Starting at $750 per year	Starting at $1500 per year	Starting at $2000 per year
		Request Quote	Request Quote	Request Quote
		$75 per month	$150 per month	$200 per month

Docker for Mac

A native application using the OS X sandbox security model which delivers all Docker tools to your Mac.

Docker for Windows

A native Windows application which delivers all Docker tools to your Windows computer.

Docker for Linux

Install Docker on a computer which already has a Linux distribution installed.

Docker Engine

Create Docker images and run Docker containers.

As of v1.12.0-rc1, Engine includes swarm mode container orchestration features.

Docker Compose

Defines applications built using multiple containers.

Docker Hub

A hosted registry service for managing and building images.

Docker Cloud

A hosted service for building, testing, and deploying Docker images to your hosts.

Docker Trusted Registry

(DTR) stores and signs your images.

Docker Universal Control Plane

(UCP) Manage a cluster of on-premises Docker hosts they were a single machines.

Docker Machine

Automate container provisioning on your network or in the cloud. Available for Windows, Mac OS X, or Linux.

Docker Client

- 도커 엔진과 통신하는 프로그램
- 윈도우/맥/리눅스 지원
- Kitemetic
- Docker CE for Linux
- Docker CE for Mac / Docker CE for Widows

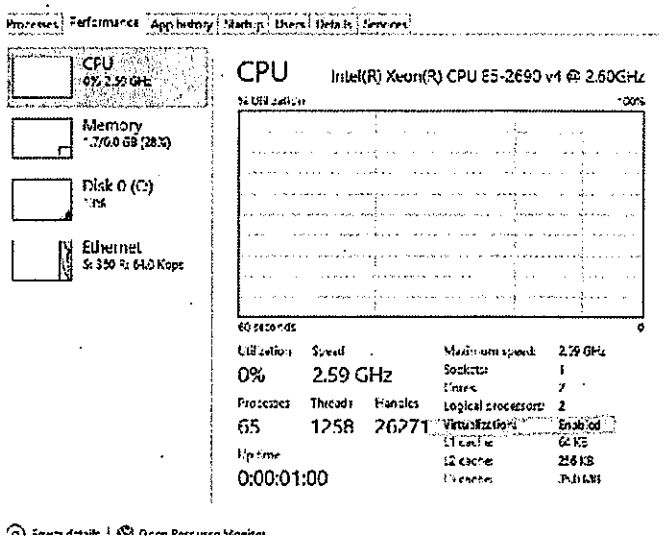
Docker CE for Mac

- xhyve/bhyve를 기반으로 하는 HyperKit 을 임베디드 하이퍼바이저로 사용한다

https://github.com/docker/HyperKit/

Docker CE for Windows

· 윈도우에 내장된 Hyper-V 기술을 기반으로 한다.



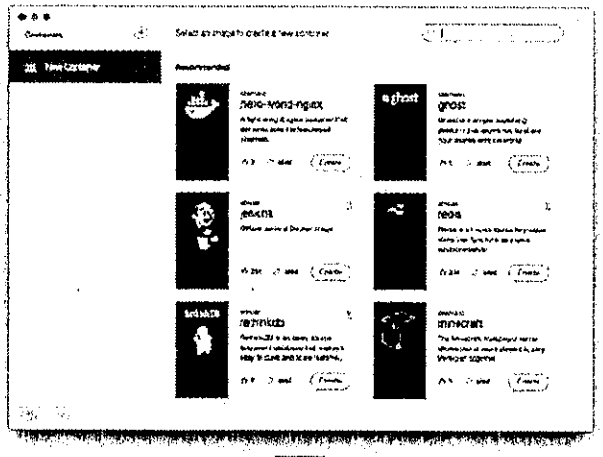
Docker CE for Linux(CenOS 기준)

```
$ sudo yum update
$ curl -fsSL https://get.docker.com/ | sh
$ sudo yum install docker-engine
$ sudo systemctl enable docker.service
$ sudo systemctl start docker
```

https://docs.docker.com/engine/installation/

Kitemetic

템플릿 기반으로 도커 컨테이너를 관리할수 있게 해주는 GUI 도커 클라이언트



41

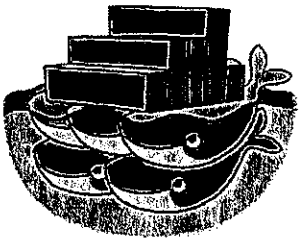
Docker Host OS

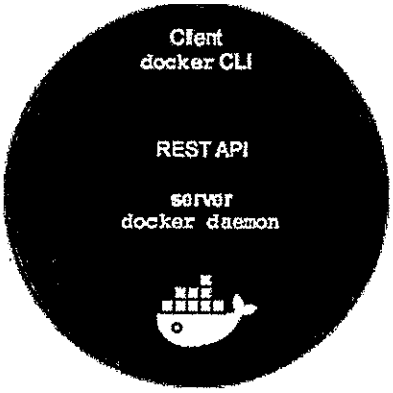
도커 엔진을 설치할수 있는 환경, 64비트 리눅스 커널 버전 3.10 이상 환경

- Ubuntu, CentOS, RHEL
- Container OS(구 CoreOS)
- Debian, SUSE, Fedora, etc

Docker Engine

- 어플리케이션을 컨테이너로 만들고 실행하게 해주는 데몬
- Swarm 통합





윈도우는 도커 호스트 OS일까

- It's not virtualization
  Docker for Windows will not run Linux images

- Building a full union FS with NTFS semantics is hard

- Hybrid model
  Virtual block device + NTFS partition per container
  Symlinks to layers on host FS to keep block devices small

- FROM scratch

- Windows images must derive from Windows base image
  windowsservercore – large, nearly full Win32 compatibility
  nanoserver – small, fast to boot, software may need porting

- Base images are delivered separately from Docker

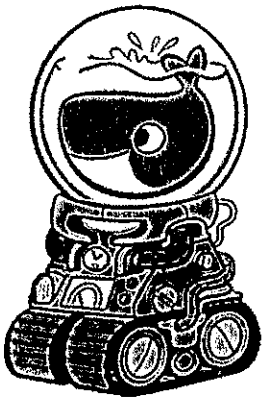
- Docker runs on host

- Launches silo in a stateless, lightweight Hyper-V VM

http://www.slideshare.net/Docker/windows-server-containers-how-we-hot-here-and-architecture-deep-dive

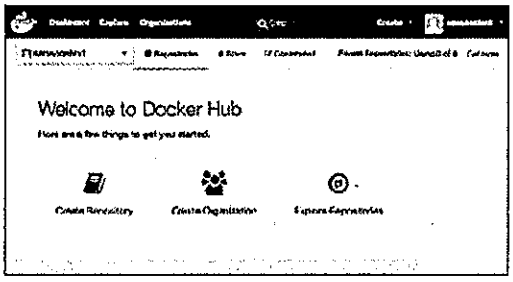
Docker Machine

로컬, 리모트 도커 엔진을 프로비저닝 해주는 클라이언트



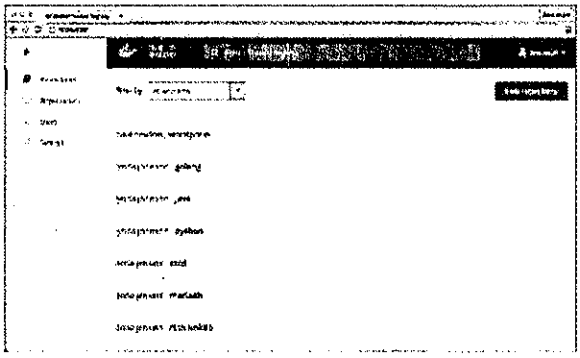
Docker Hub

- 도커 이미지를 관리하는 저장소.
- 오픈소스 공식 이미지 관리.
- 공개(무료), 비공개(유료)
- https://hub.docker.com



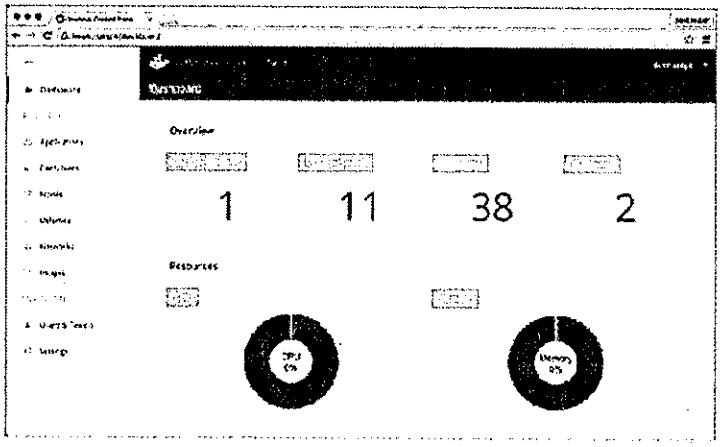
Docker Trusted Registry

- 도커 이미지 저장소를 구축할 수 있다
- 설치형, 인증 지원
- Registry(무료), DTR(유료)



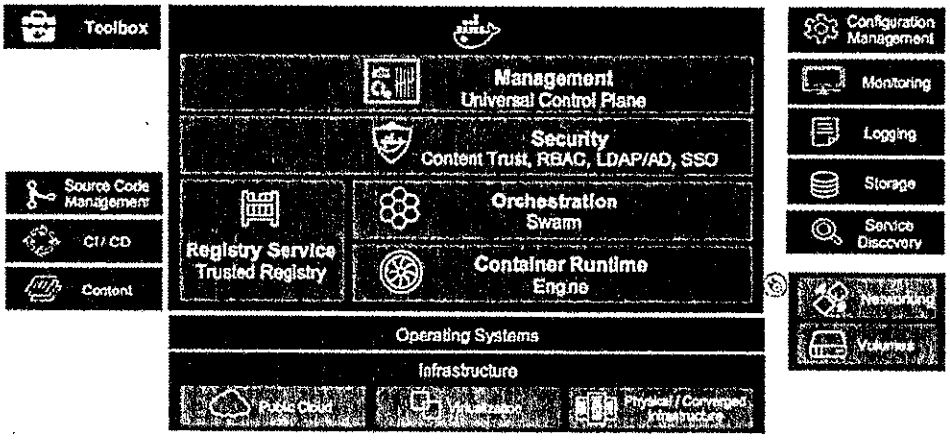
DUCP

- Docker Universal Control Plan
- 도커 클러스터 관리도구
- 설치형(유료)



Docker Data Center

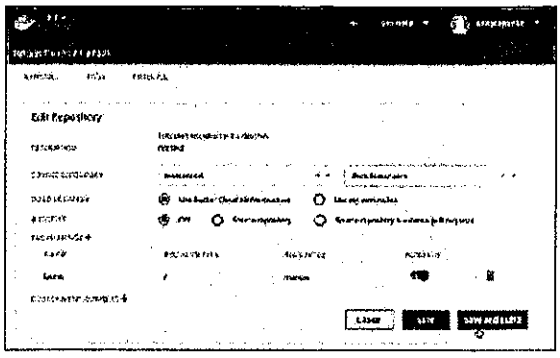
- DUCP + DTR



https://blog.docker.com/2016/02/docker-datacenter-caas/

Docker Cloud

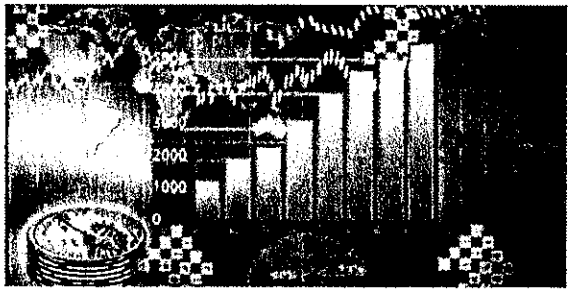
- 도커 이미지,컨테이너 관리를 지원
- 공개/비공개 클라우드 관리 가능
- tutum을 인수해서 통합



도커 어디까지 왔나

도커는 얼마일까

Sources: Microsoft Tried To Buy Docker for $4B



Scott Raynovich
June 27, 2016
1:05 pm ET

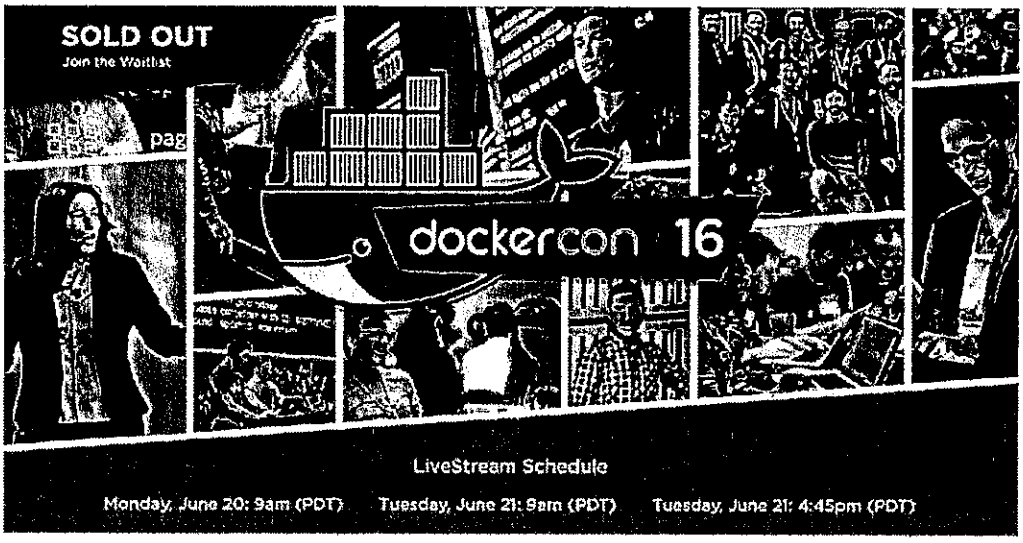
At last week's DockerCon 2016 event in Seattle there was a lot of behind-the-scenes chatter about Microsoft wanting to buy Docker for billions of dollars. Microsoft's bid for Docker was rumored to be as much as $4 billion for the 250-person container technology startup in the last six months, according to multiple sources with contacts close to both companies.

The deal was never completed because the companies could not agree on a price, said several sources under condition of anonymity. The sources said the rumored deal was a hot topic in the startup and venture capital circles several months ago.

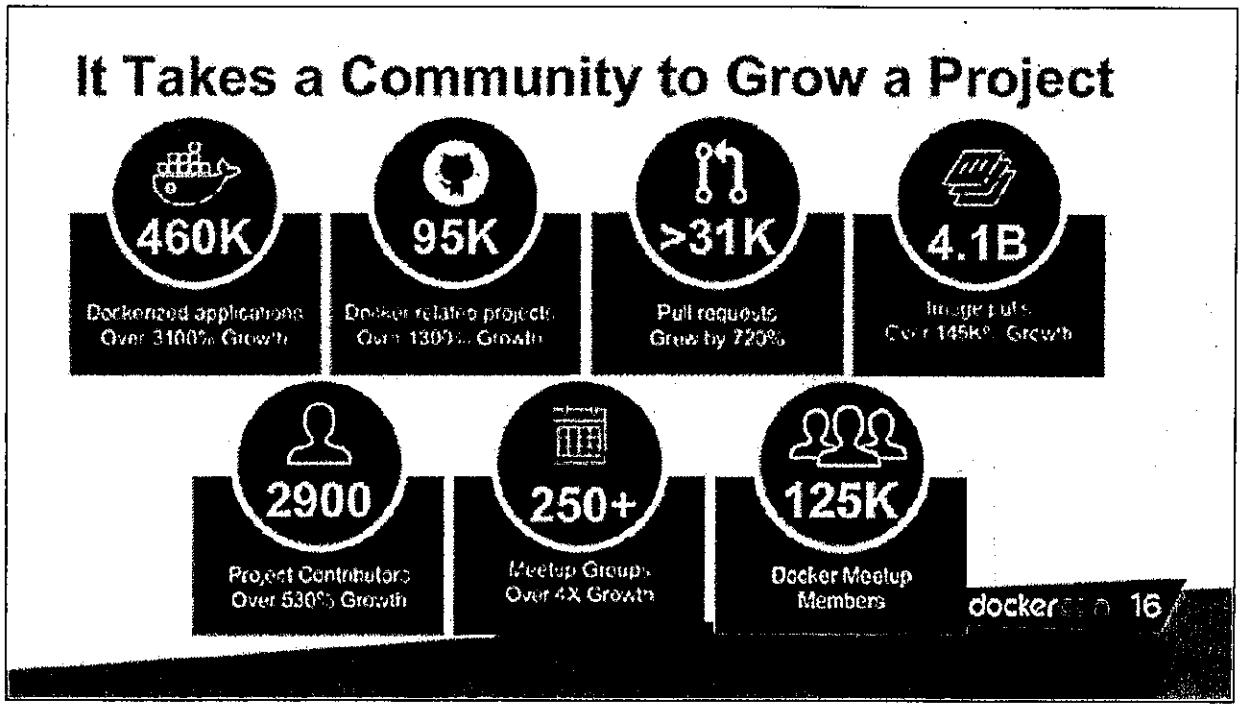
https://www.sdxcentral.com/articles/news/sources-microsoft-tried-to-buy-docker-for-4b/2016/06/

Docker Conference

전세계 도커들의 축제,미국/유럽에서 매년개최



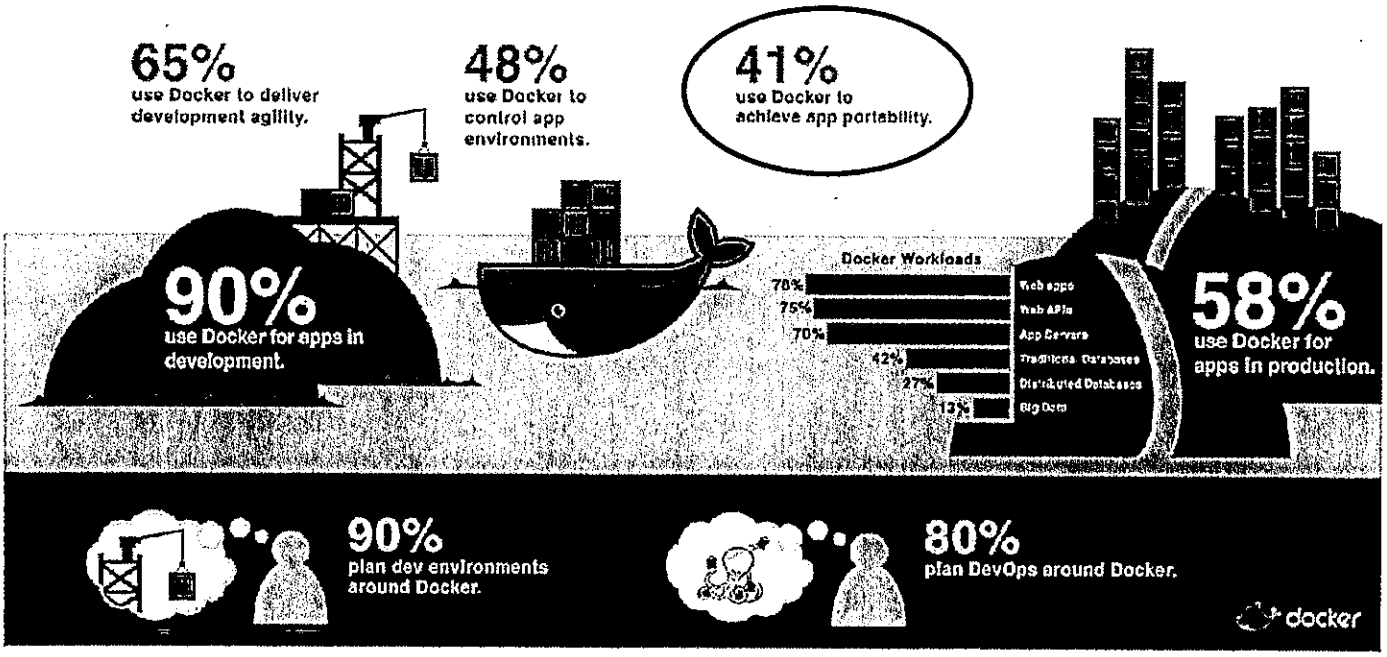
http://2016.dockercon.com/



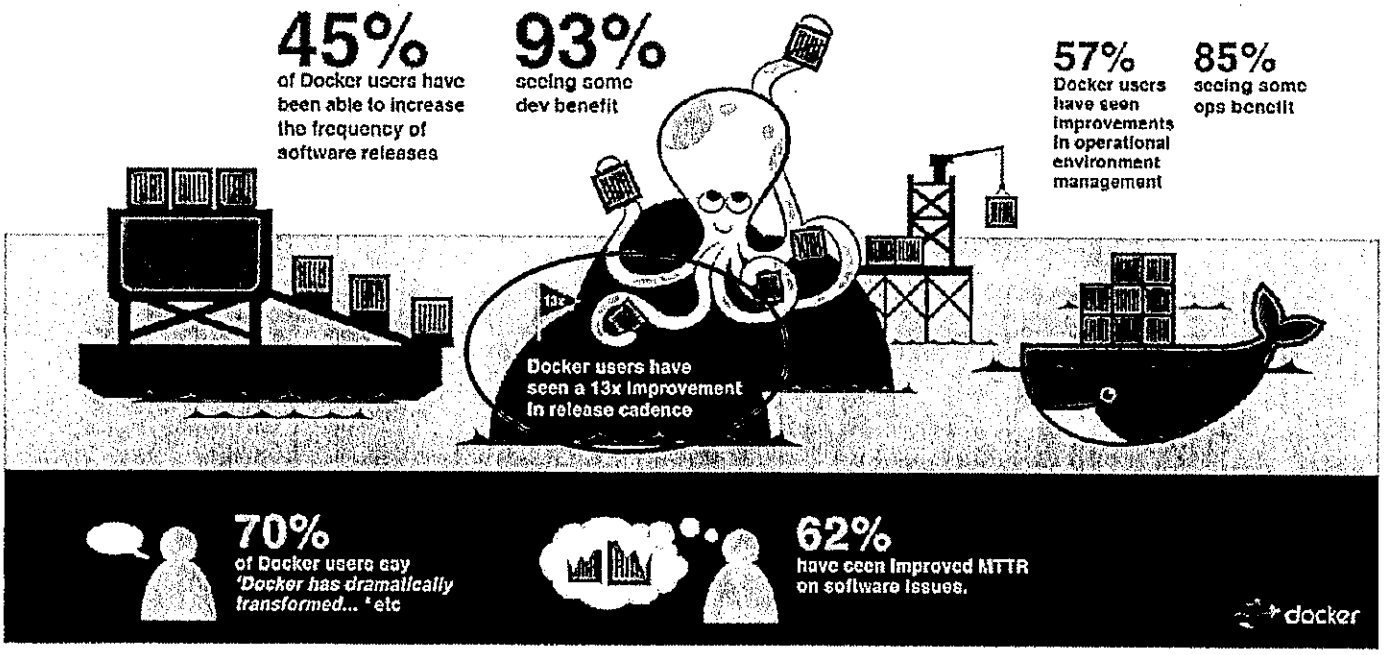
2016 Docker Survey

https://www.docker.com/survey-2016

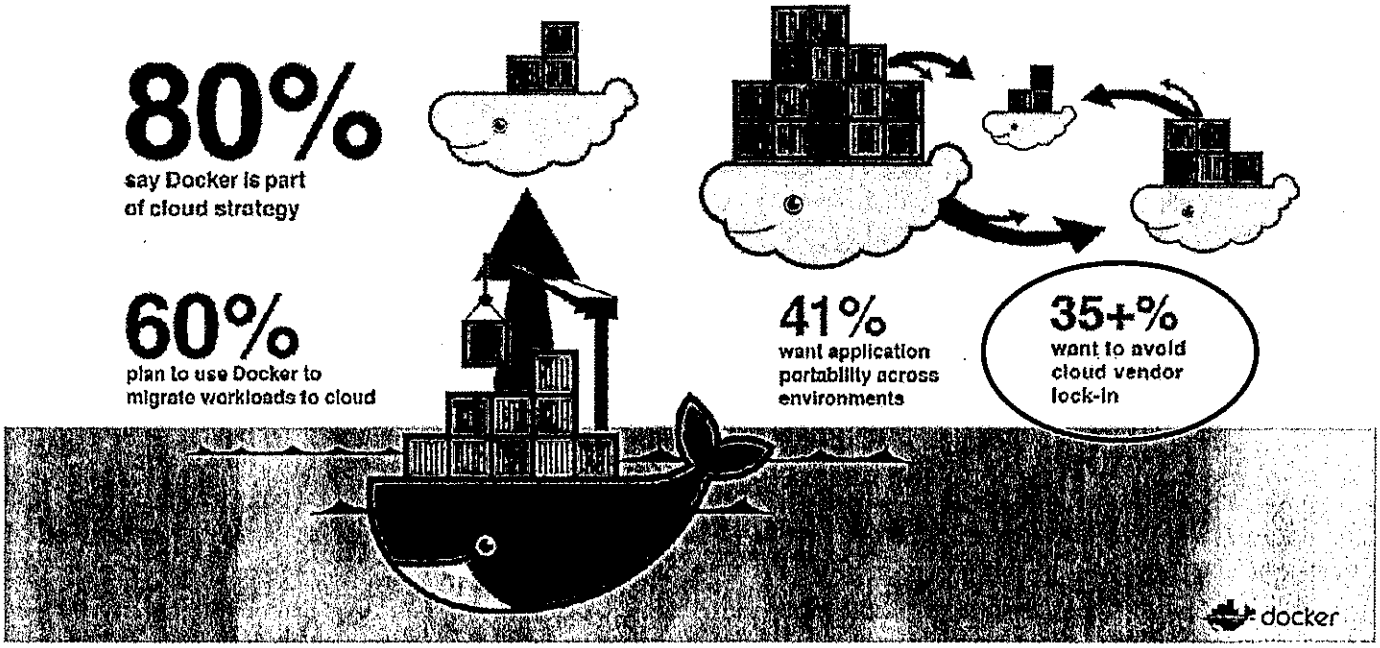
왜 도커를 사용하는가



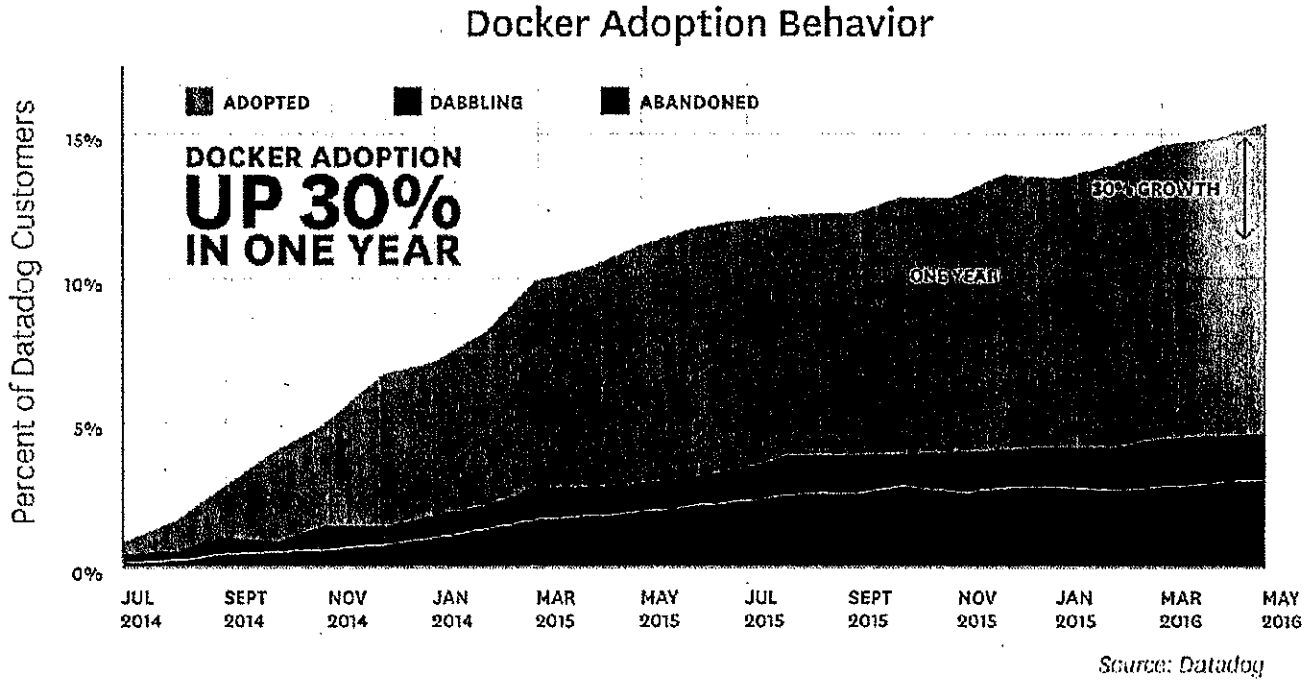
도커가 얼마나 도움이 되었는가



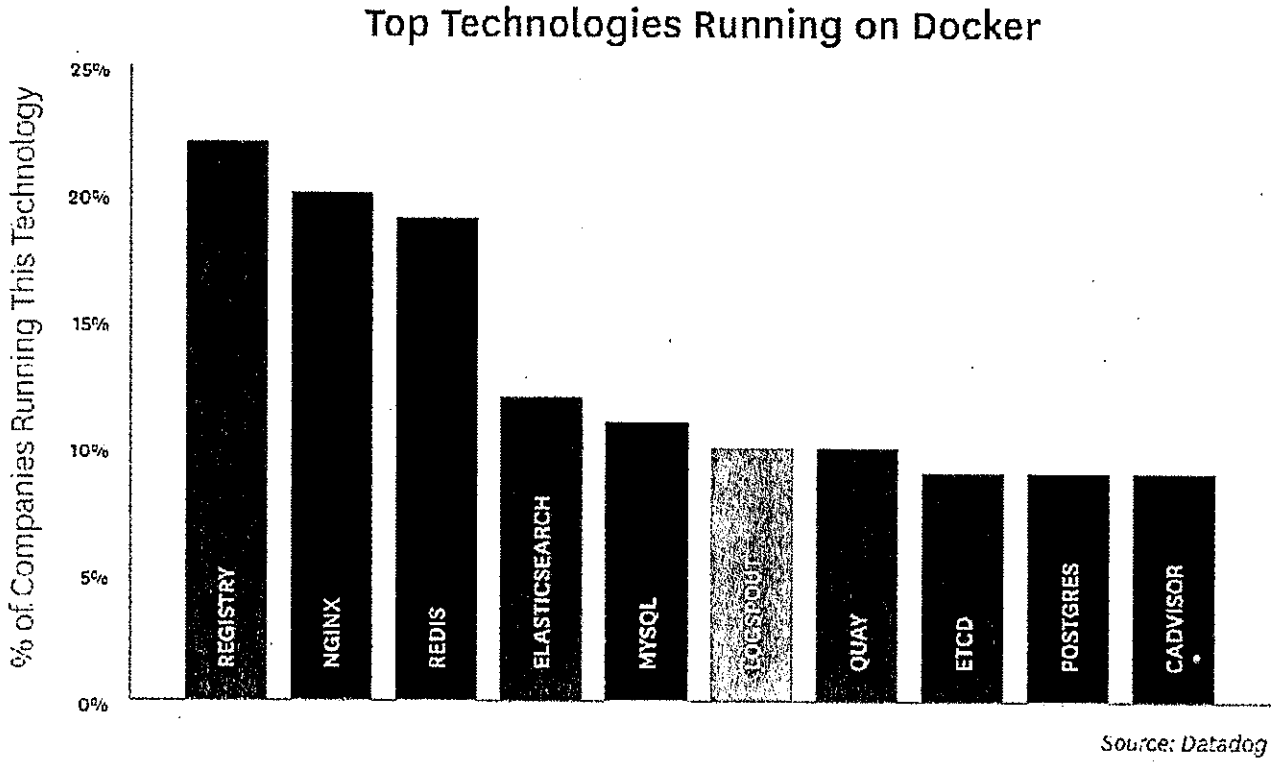
클라우드에서 도커효과



Real Docker Adoption Is Up 30% in One Year

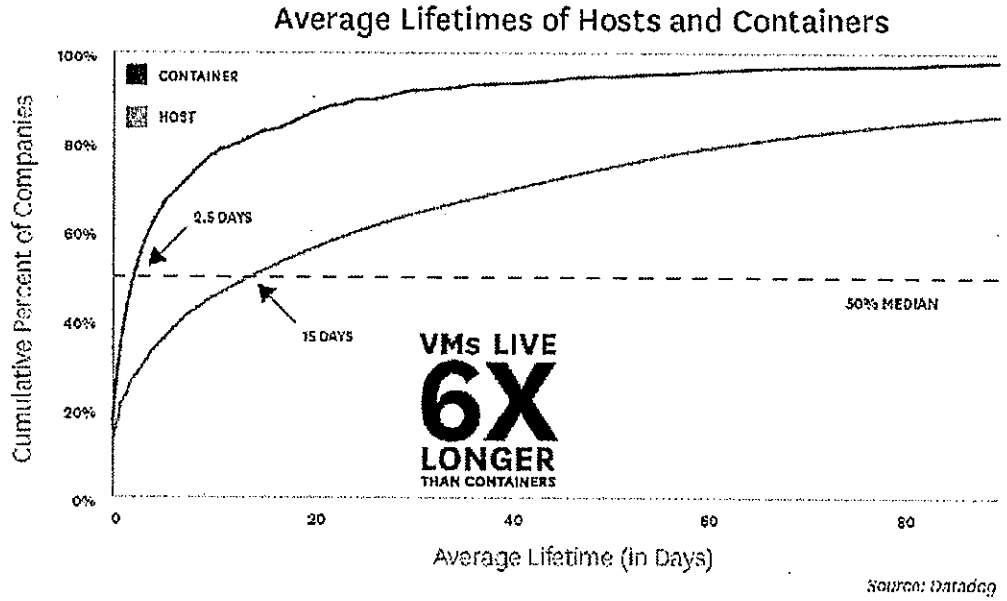


Source: Datadog



https://www.datadoghq.com/docker-adoption/

도커 컨테이너 평균 지속시간 2.5일(VM은 15일)



https://www.datadoghq.com/docker-adoption/

What Non-Users Think of Docker

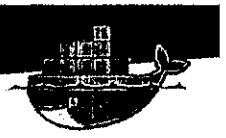
For those who had not used Docker we tried to assess their feeling about the technology.

Sentiment	Count	Percent
Gotta' try it soon	19	52.8
Ground breaking new tech	2	5.6
It's yet another hipster tech - count me out	2	5.6
Not sure	6	16.7
VMs are fine for my development work	2	5.6
Waiting for it to be more matur	2	5.6
NA	3	8.3

Over 50% of the non-users indicated that they needed to use the technology soon, with remaining respondents generally being unsure or not foreseeing a need for LNCDF. Again, this is only 18 percent of the total respondents to the survey.

https://blog.openshift.com/survey-results-developer-usage-docker-containers/

[실습1-1] 환경 구성



윈도우즈

- 64비트 10이상: Docker CE for Windows
- 나머지: Docker Toolbox

맥

- 요세미티 10.10: Docker CE for Mac
- 나머지: Docker Toolbox

https://www.docker.com/products/docker#/mac

[실습1-2] 설치 확인

터미널에서 도커 클라이언트 실행 환경 설정

```
$ eval $(docker-machine env default)
```

도커 엔진 버전을 확인한다.

```
$ docker version
```

도커 엔진 상세정보를 확인한다.

```
$ docker info
```

[실습1-3] hello-world

터미널에서 도커 클라이언트 실행 환경 설정

```
$ eval $(docker-machine env default)
```

테스트 이미지 실행

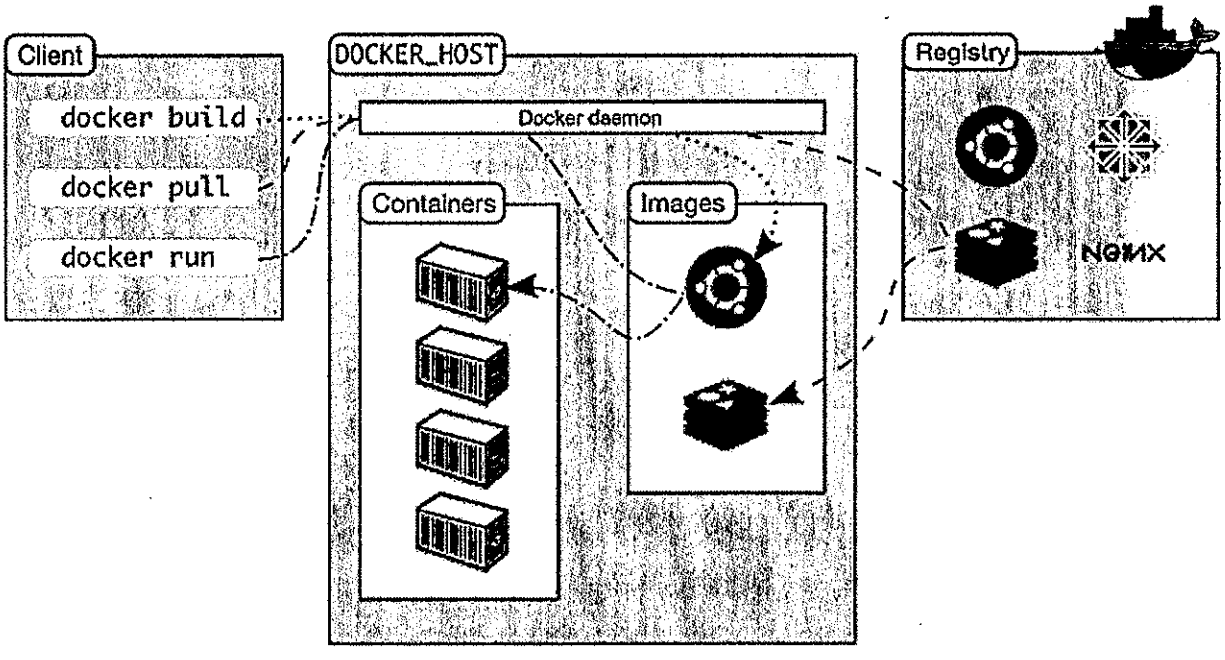
```
$ docker run hello-world
```

도커 컨테이너 확인

```
$ docker ps -a
```

Docker Architecture

도커 아키텍처



기반 기술

Namespaces

- 가벼운 프로세스 가상화 기술
- 격리된 작업공간(컨테이너)을 구성해준다.
- 1992: "The Use of Name Spaces in Plan 9"
- 유형
  - mnt(mount points, 파일 시스템)
  - pid(프로세스)
  - net(네트워크 스택)
  - ipc(System V IPC)
  - uts(hotsname)
  - user(UIDs)

기반 기술

Control groups(==cgroups)

- 실행중인 어플리케이션이 원하는 만큼 리소스를 사용하게 한다.
- 특정 컨테이너가 지정한 만큼 리소스를 쓰도록 제한한다.
- 2006년 구글에서 시작
  - 'process containers' -> 'Control Groups' 변경
- 관리 리소스
  - memory: mm/memocontrol.c
  - cpuset: kernel/spuset.c
  - net_prio: net/core/netprio_cgroup.c
  - devices: security/device_cgroup.c

Namespace: 프로세스별 리소스 격리

Cgroups: 리소스 관리

Union Filesystem

다른 파일 시스템을 Union mount 하도록 해주는 리눅스 파일 시스템 서비스

브랜치라고 하는 여러개 나누어진 파일 시스템을 하나의 파일 시스템처럼 사용할 수 있게 해준다.

read-only 레이어들의 통합

권한수준

root 사용자와 동일한 수준의 권한이 필요

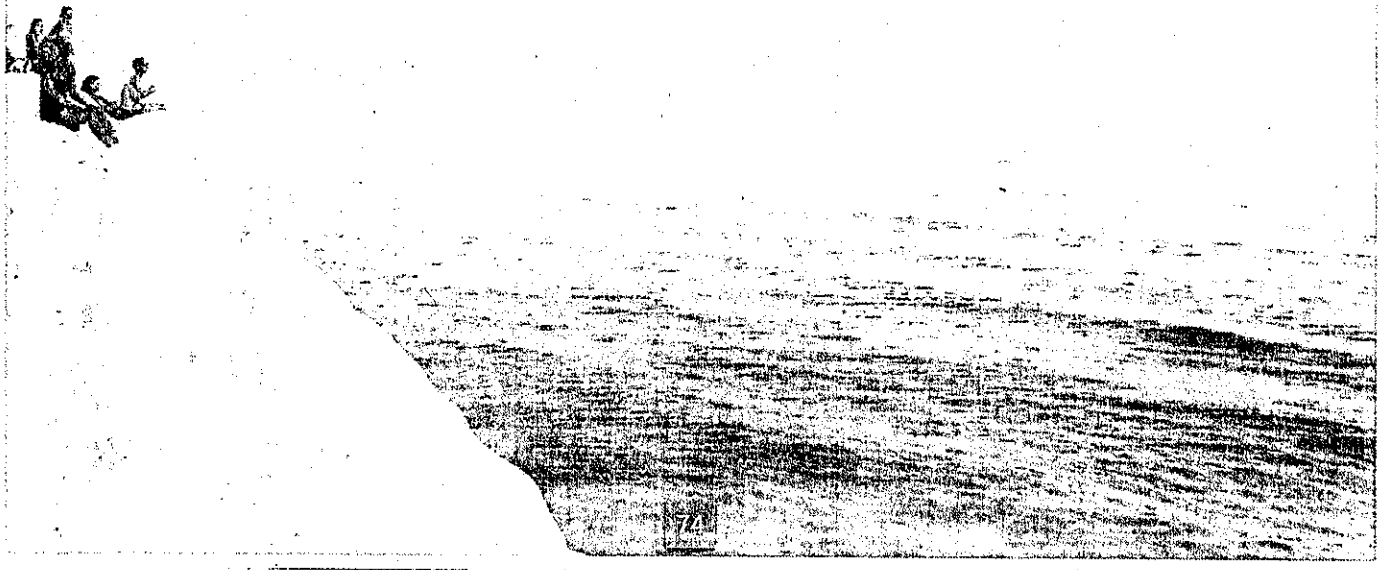
docker 데몬을 실행하기 위해서는 root 사용자 필요

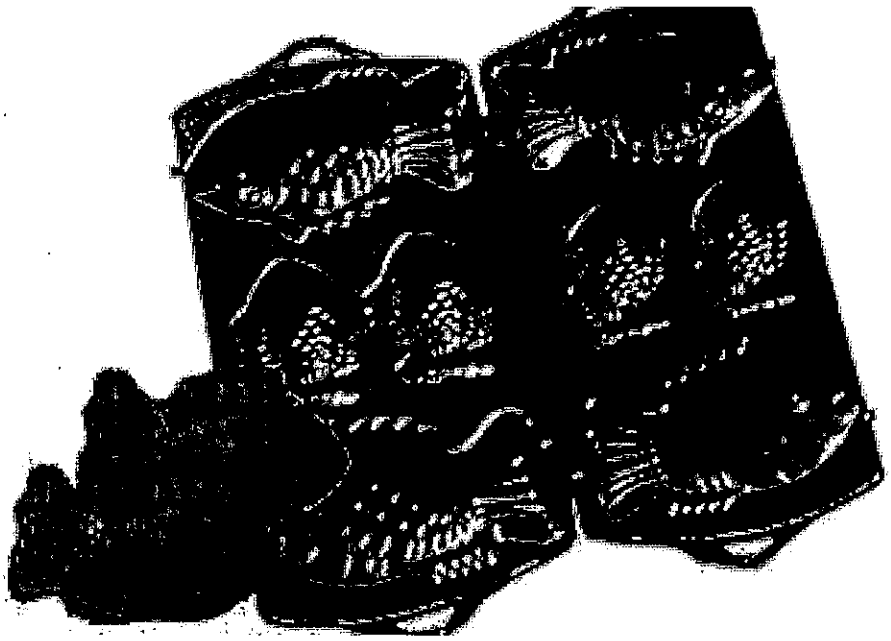
TCP 포트가 아닌 Unix 소켓과 데몬이 바인딩 하기 때문

사용자 그룹: docker

사용자: docker

Docker 이미지

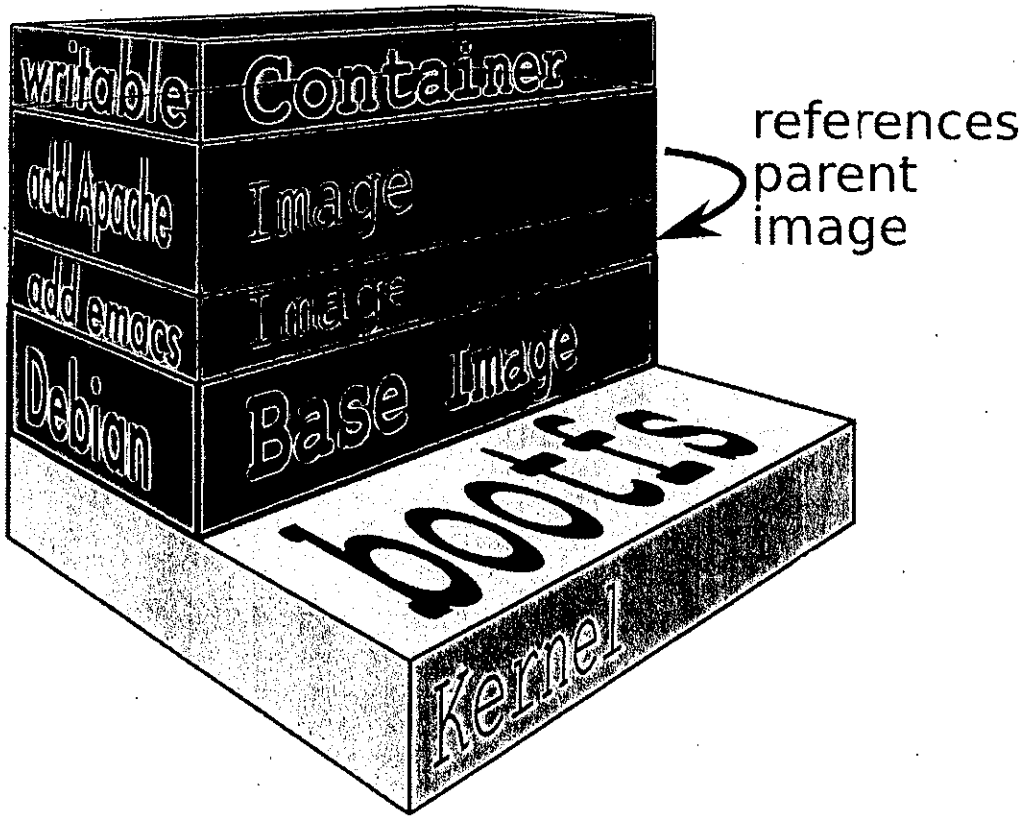




컨테이너가 붕어빵이면
이미지는 붕어빵 틀

도커 이미지

- 컨테이너를 만들기 위한 것
- 레이어 파일 시스템으로 각 파일시스템이 곧 이미지.
- 읽기전용 파일시스템으로 도커가 어플리케이션을 배포하는 단위.



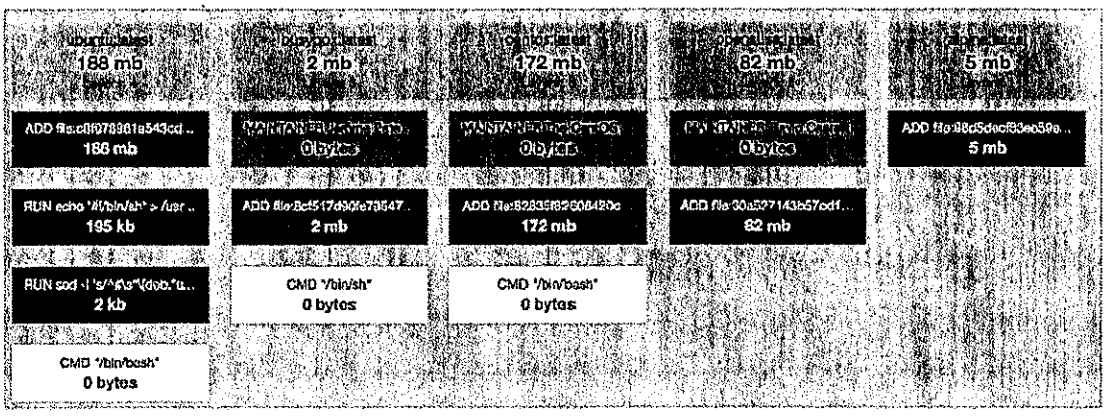
- bootfs: 도커 시스템이 사용하는 파일시스템
- union mount: 여러 파일시스템을 하나의 파일 시스템으로 보이게
- Read-Only Layer, Read-Write Layer

베이스이미지

- 이미지의 부모가 되는 이미지.
- 공식 베이스이미지는 ubuntu 였지만, alpine으로 변경되었음.

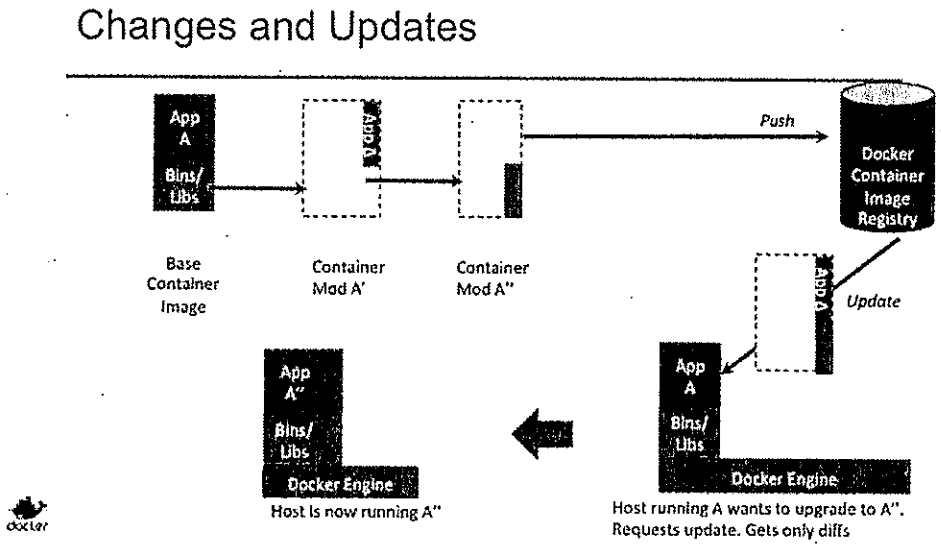
Image Layers

- 도커 이미지를 비쥬얼하게 보여준다.
- https://imagelayers.io/



https://www.brianchristner.io/docker-image-base-os-size-comparison/

어떻게 변경된 부분만 저장하는가



http://slides.com/vichoward/docker-hackathon#/0/5

Dockerfile

도커 이미지를 만들기 위해 설치할 SW, 필요한 설정을 정의하는 파일

- 프로비저닝(Provisioning) 역할
- 이미지 제작 과정을 기록

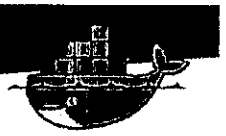
nginx Dockerfile

```
# Version: 0.0.1
FROM ubuntu:14.04
MAINTAINER Sangcheol Hwang "k16wire@이메일"

RUN apt-get update
RUN apt-get install -y nginx
RUN echo 'Hi, I am in your container'

EXPOSE 80
```


[실습1-4] 이미지 검색



DockerHub(https://hub.docker.com/)사이트에서 검색한다.

- https://hub.docker.com/r/docker/whalesay/

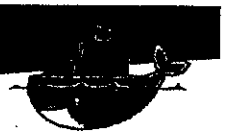
도커 명령어로 검색한다.

```
$ docker search whalesay
```

```
$ docker pull docker/whalesay
```

```
$ docker images
```

[실습1-5] 공개 이미지 사용

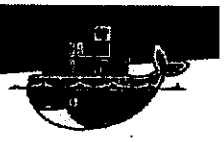


whalesay 이미지를 실행한다.

```
$ docker run docker/whalesay cowsay boo
```

```
$ docker ps -a
```

[실습1-6] 내 이미지 만들기



docker-whale 이미지를 위한 Dockerfile을 만든다.

```
FROM docker/whalesay:latest

RUN apt-get -y update && apt-get install -y fortune

CMD /usr/games/fortune -a | cowsay
```

docker-whale 이미지를 빌드한다.

```
$ docker build -t docker-whale .
```

docker-whale 이미지를 실행한다.

```
$ docker run docker-whale
```

https://docs.docker.com/engine/getstarted/step_four/

Dockerfile Cheat Sheet

From 베이스 이미지를 지정한다.
```
FROM ubuntu:14.04
```

RUN 리눅스 명령어를 실행한다.
```
RUN apt-get update
```

CMD 이미지를 실행하는 명령을 지정한다.
```
CMD [ " mysqld " ]
```

EXPOSE 공개하는 포트를 정의한다.
```
EXPOSE 80 443
```

ENV 환경변수를 정의한다.
```
ENV NGINX_VERSION 1.11.5
```

ADD,COPY 파일을 복사한다.
```
COPY jerkins.sh /user/local/bin/
jenkins.sh
```

ENTRYPOINT 이미지 실행 명령을 재정의 한다.
```
ENTRYPOINT [ " /bin/tini " , " — " ]
```

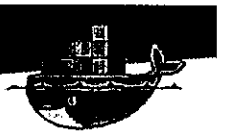
VOLUME 바인딩 디렉토리를 정의한다
```
VOLUME /var/lib/mysql
```

USER 사용자를 지정한다
```
USER jenkins
```

WORKDIR 작업위치를 지정한다
```
WORKDIR /data
```

ONBUILD 이미지 빌드후 실행되는 명령을 지정한다.
```
ONBUILD RUN /user/local/bin/python-
build --dir /app/src
```

[실습1-7] 이미지 삭제



docker-whale 이미지를 삭제한다.

```
$ docker rmi docker-whale:latest
```

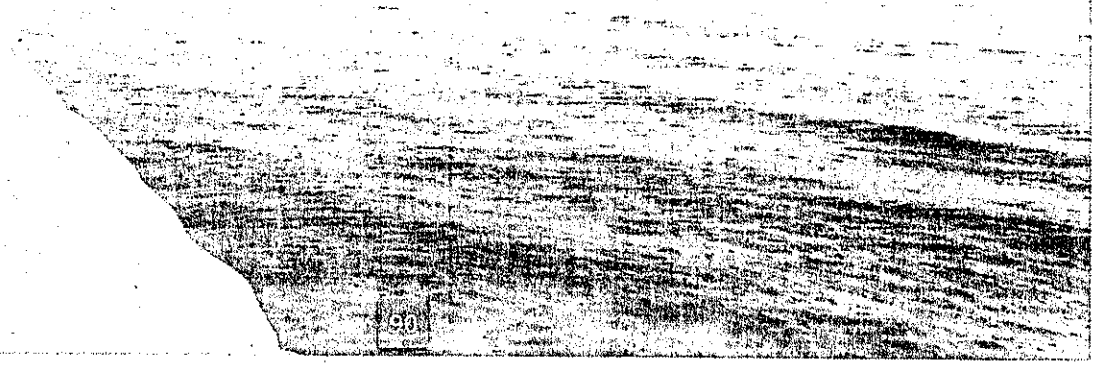
Q: 이미지를 참조하는 컨테이너가 있을때는 어떻게?
Q: 모든 이미지를 삭제하려면?

도커 이미지 크기를 줄이는 방법

- 가벼운 베이스 이미지를 사용한다.
- Dockerfile 명령을 체인으로 사용한다.
- 빌드 도구를 설치하지 않는다.
- 패키지 관리자를 정리한다.

http://pragmaticstory.com/?cat=3&paged=3

Docker 컨테이너



도커 컨테이너

- 이미지를 실행한 상태
- 읽기/쓰기가 가능한 파일 시스템
- 실행된 독립 애플리케이션

컨테이너는 가상서버가 아니다.

[실습1-8] DB 컨테이너 실행

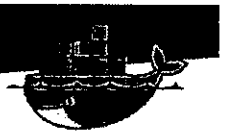
mysql 컨테이너를 실행한다.

```
$ docker run -e MYSQL_ROOT_PASSWORD='Passw0rd' --
name mydb -d mysql:5.6
```

-d 옵션: 컨테이너를 데몬으로 실행한다.

-e 옵션: 환경변수 설정

[실습1-9] 컨테이너 삭제



mysql 컨테이너를 삭제한다.

```
$ docker rm mydb
```

Q: 모든 컨테이너를 삭제하려면?

[실습1-10] nginx 컨테이너 실행

nginx 컨테이너를 실행한다.

```
$ docker run --name myweb -d -p 80:80 nginx
```

-p 옵션: 포트를 외부에 오픈한다.

Q: -P 옵션과 어떻게 다른가?

[실습1-11] nginx 컨테이너 설정

nginx 컨테이너를 원하는 설정에 맞춰 실행한다.

```
$ docker run --name myweb -v /home/docker/
nginx.conf:/etc/nginx/nginx.conf -d -p 80:80
nginx
```

윈도우즈는 /home/docker가 위치가 다를수 있으니 수정해 줘야 한다.

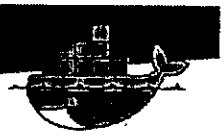
[실습1-12] 컨테이너 로그

nginx 컨테이너에서 출력되는 로그를 확인한다.

```
$ docker logs myweb
```

-f 옵션: 로그를 연속해서 보여준다.

[실습1-13] 컨테이너 조사



nginx 컨테이너 상세 내용을 확인합니다.

```
$ docker inspect myweb
```

[실습1-14] 컨테이너 연결

nginx 컨테이너와 jenkins 컨테이너를 연결한다.

1.jenkins 컨테이너를 실행한다.

```
$ docker run -d -p 8080:8080 -p 5000:5000 --name
myjenkins jenkins
```

2.nginx 컨테이너를 실행하면서 jenkins 컨테이너를 연결한다.

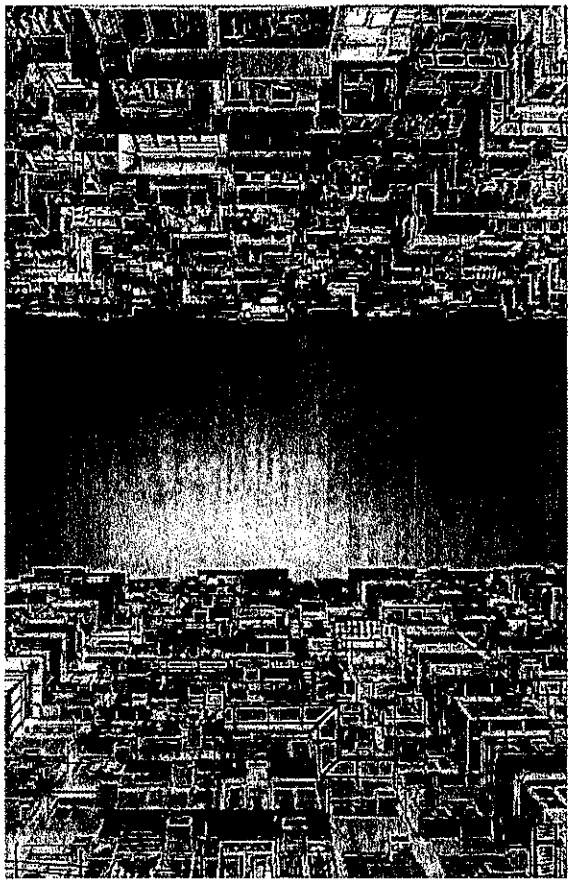
```
$ docker run -p 80:80 -v /home/docker/
nginx.conf:/etc/nginx/nginx.conf --link
myjenkins:jenkins -d nginx
```

Docker로 꾸미는 로컬환경

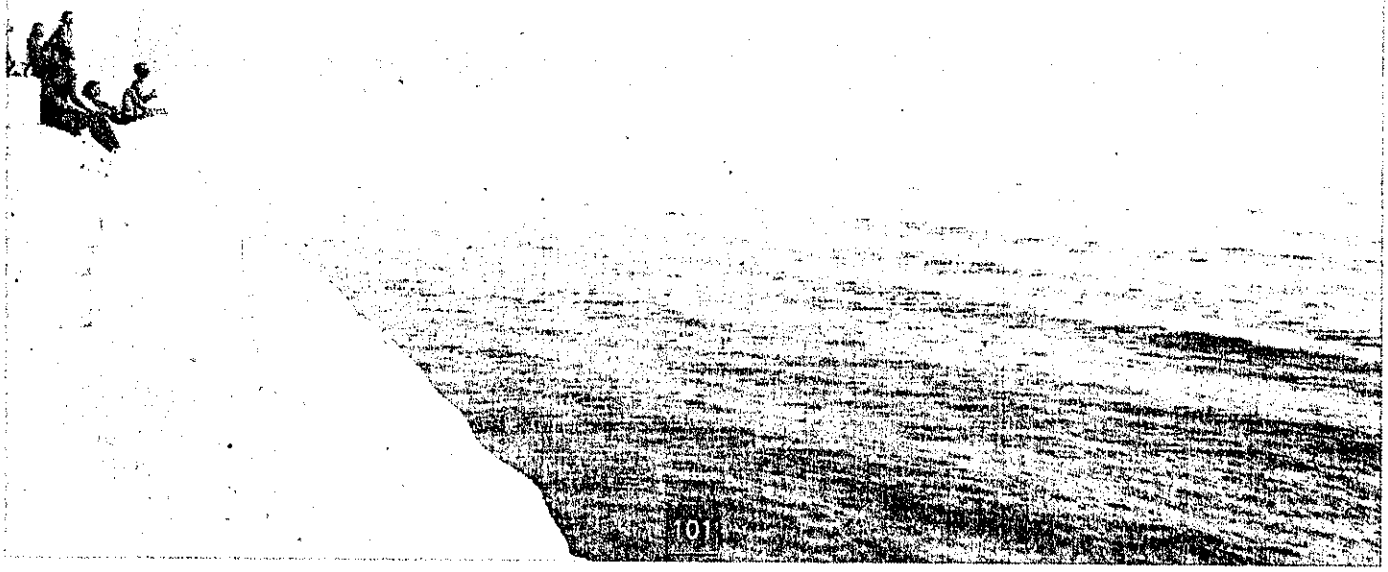
컨테이너 활용

Docker Machine

Docker Compose



컨테이너 활용



도커 호스트 구성

US1:독립된 로컬 개발환경

로컬 설정에 영향을 받지 않는 독립된 개발환경을 구성. Vagrant,Docker Machine을 이용하면 일관된 인터페이스로 구성이 가능

Vagrant Docker provider

(http://docs.vagrantup.com/v2/docker/)

US2:VM기반의 도커 Host

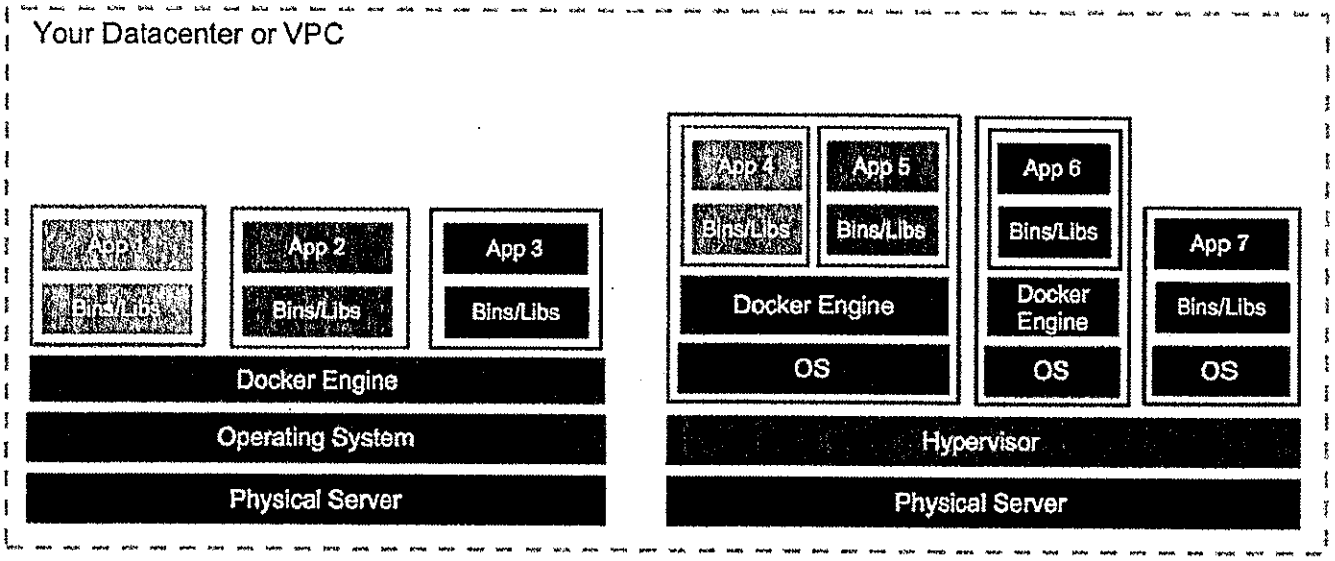
VM서버로 도커 Host 구성한다. Host에는 한개 컨테이너만 운영하는게 효과적이다. VM 유형,환경에 상관없이 배포가 가능하다.

Docker Machine(https://github.com/docker/machine)

US3:싱글서버 도커 Host

물리서버 1대로 도커 Host를 구성한다.

여러 컨테이너를 운영할수 있어서 효율적이다. 구성이 단순하여 관리가 용이하지만 서버 에러에 대한 위험성이 크다.



US4:도커 Host 클러스터

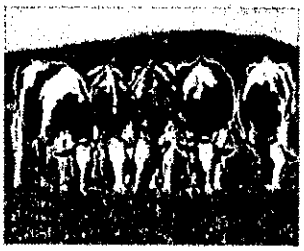
여러 도커 Host 서버를 하나의 Host처럼 관리하기 위해 클러스터를 구성한다. 컨테이너 스케쥴링, Host 모니터링 등이 필요.

- fleet
- swarm
- kubernets
- mesos

컨테이너 데이터 백업

Service Model

- Pets are given names like pussinboots.cern.ch
- They are unique, lovingly hand raised and cared for
- When they get ill, you nurse them back to health



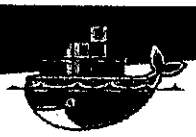
- Cattle are given numbers like vm0042.cern.ch
- They are almost identical to other cattle
- When they get ill, you get another one

- Future application architectures should use Cattle but Pets with strong configuration management are viable and still needed

Gavin McCance, CERN 17

http://www.it20.info/2012/12/vcloud-openstack-pets-and-cattle/

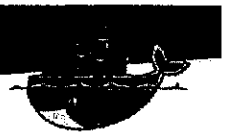
[실습1-15] 데이터 볼륨



mysql 컨테이너 데이터를 호스트 디렉토리 mydqldata에 저장한다.

```
$ docker run -e MYSQL_ROOT_PASSWORD='Passw0rd' --
name mydb -d -v /mysqldata:/var/lib/mysql mysql:
5.6
```

-v 옵션: 볼륨 바인딩, 컨테이너 특정 디렉토리와 호스트 특정 디렉토리를 연결한다.



mysql 컨테이너 설정을 원하는 대로 변경한다.

```
$ docker run -e MYSQL_ROOT_PASSWORD='Passw0rd' --
name mydb -d -v /mysqldata:/var/lib/mysql -v /
home/docker/my.cnf:/etc/mysql/conf.d/my.cnf
mysql:5.6
```

-v 옵션

데이터 컨테이너

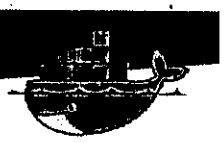
- 다른 컨테이너에게 볼륨을 공유하기 위해 만드는 컨테이너
- 애플리케이션을 실행하지 않는다.

[실습1-17] 데이터 컨테이너 생성

/data/app1 디렉토리를 공유하는 데이터 컨테이너를 만든다.

```
$ docker run --name mydata -v /data/app1 busybox
true
```

[실습1-18] 데이터 컨테이너 활용

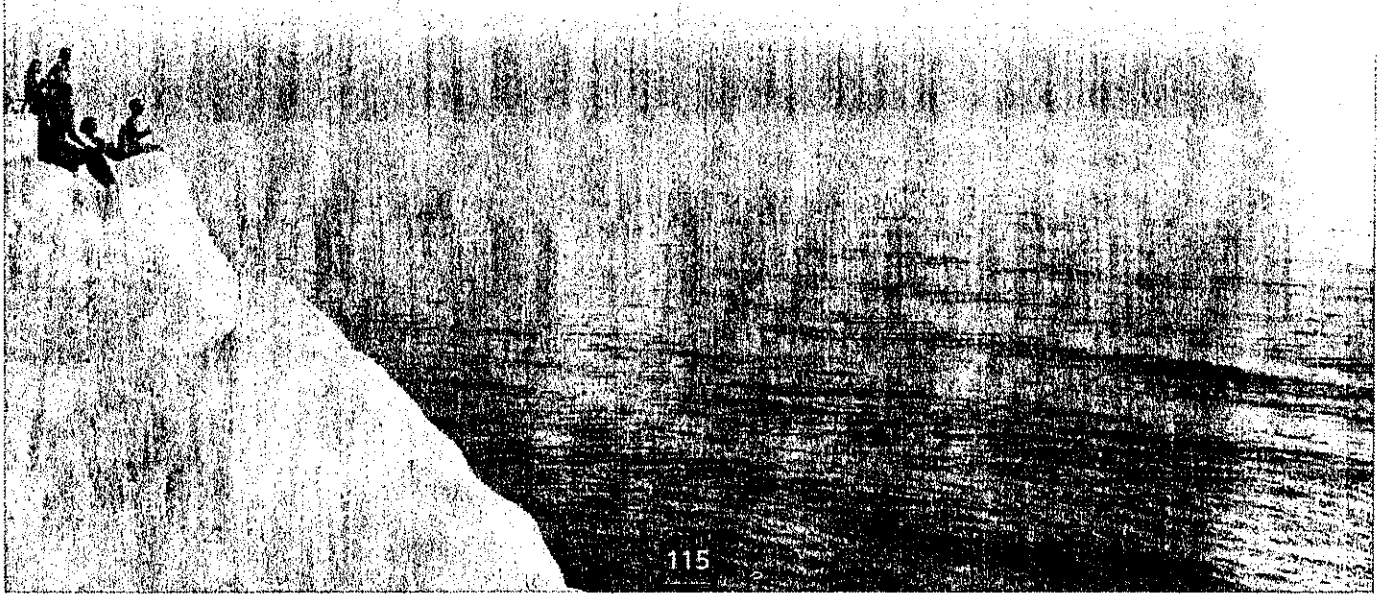


mydata 컨테이너를 참조하는 컨테이너를 만든다.

```
$ docker run -it --volumes-from mydata ubuntu:
14.04
```

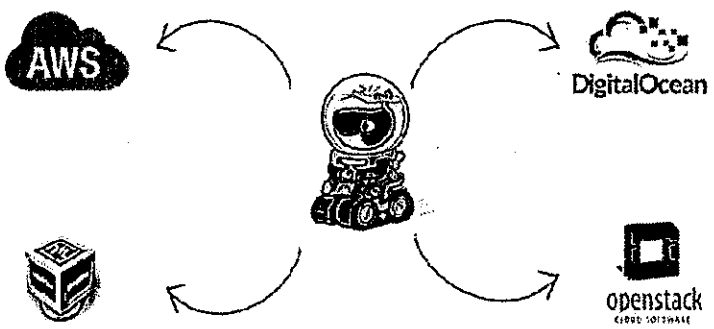
-it옵션: 인터렉티브 모드

Docker Machine

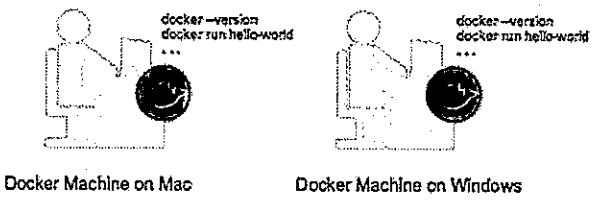


Docker Machine

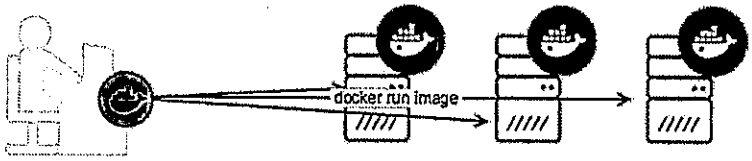
가상서버에 도커엔진을 설치할수 있게 도와주는 도구



유스케이스



로컬에 도커엔진 설치



리모트 서버에 도커엔진 설치

boot2docker

도커 컨테이너를 실행하기 위한 가벼운 리눅스 배포판

- Tiny Core Linux 기반
- Docker Machine의 기본설정
- https://github.com/boot2docker/boot2docker

도커 머신 주요 명령어 1/2

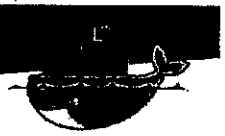
- docker-machine config
- docker-machine env
- docker-machine inspect
- docker-machine ip
- docker-machine kill
- docker-machine restart
- docker-machine provision

도커 머신 주요 명령어 2/2

- docker-machine ssh
- docker-machine start
- docker-machine stop
- docker-machine status
- docker-machine upgrade
- docker-machine url

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

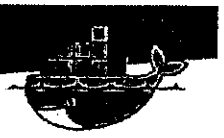
[실습1-19] 머신 조회



생성된 서버를 조회한다.

```
$ docker-machine ls
```

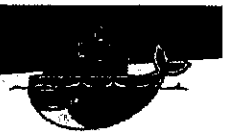
[실습1-20] 머신 생성



my-dev 서버를 생성한다.

```
$ docker-machine create --driver virtualbox my-
dev
```

[실습1-21] 머신 연결



my-dev 서버 정보를 조회한다.

```
$ docker-machine env my-dev
```

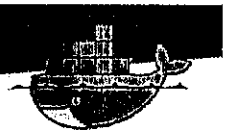
현재 쉘에 서버 연결정보를 설정한다.

```
$ eval $(docker-machine env my-dev)
```

```
$ docker-machine.exe env --shell cmd my-dev
```

https://docs.docker.com/machine/reference/env/

[실습1-22] ssh 연결



my-dev 서버 정보에 ssh로 연결한다.

```
$ docker-machine ssh my-dev
```

Q: 계정과 암호는?

https://docs.docker.com/machine/reference/env/

[실습1-23] scp 복사

my-dev 서버에 my.cnf 파일을 scp로 복사한다.

```
$ docker-machine scp my.cnf my-dev:~/
```

Q: 인증정보가 필요한 경우에는?

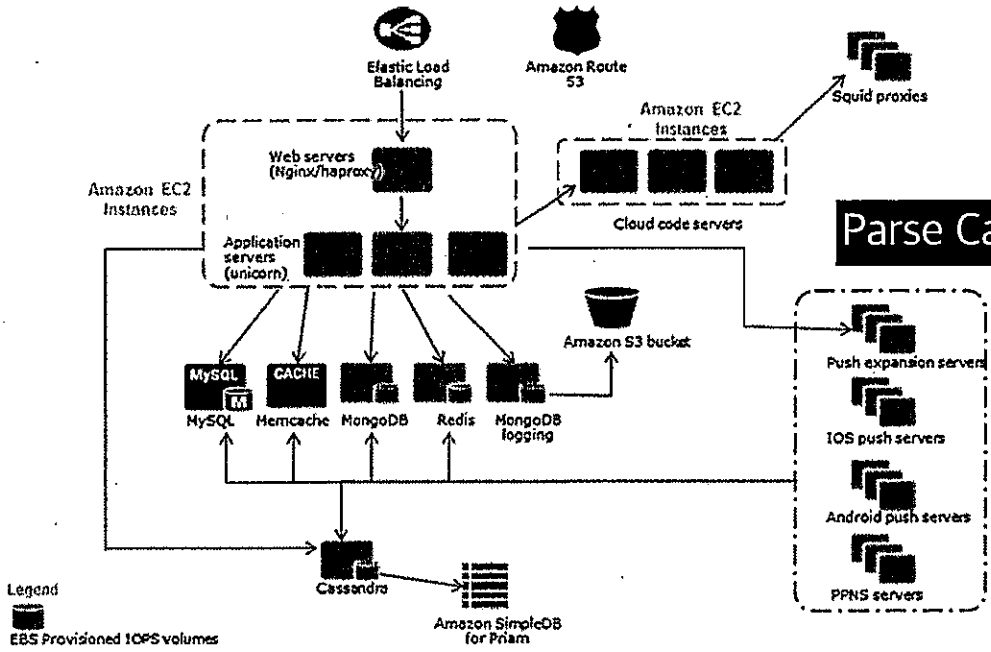
https://docs.docker.com/machine/reference/env/

도커 머신 Drivers

- Amazon Web Services
- Microsoft Azure / Hyper-V
- Digital Ocean
- Google Compute Engine
- IBM Softlayer
- Oracle VirutalBox
- VMware vCloud Air / Fusion / vSphere

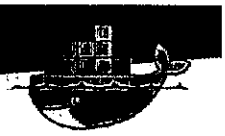
https://docs.docker.com/machine/drivers/

아마존 AWS



https://aws.amazon.com/ko/solutions/case-studies/parse/

[실습1-24] AWS에 머신생성



AWS ec2에 머신을 생성한다.

```
$ docker-machine create --driver amazonec2 \
--amazonec2-access-key AKI-xxx \
--amazonec2-secret-key 8T9-xxx \
--amazonec2-region us-east-1a \
--amazonec2-instance-type "t2.micro" \
--amazonec2-ami ami-56d4ad31 \
--amazonec2-security-group default-xxx \
 aws-sandbox
```

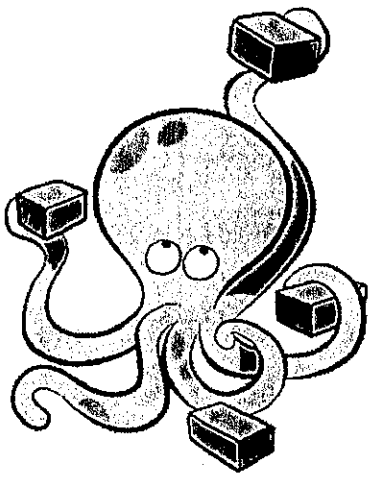
https://docs.docker.com/machine/examples/aws/

Docker Compose

Docker Compose

여러개 컨테이너 구성된 어플리케이션을 만들고 관리 할수 있게 해주는 도구

- 웹 프런트엔드
- 사용자 관리
- 결재
- 데이터 베이스



설치 방법

docker-toolbox와 docer CE for xxx 설치시 함께 설치된다.

- 수동설치

```
$ curl -L "https://github.com/docker/compose/releases/download/1.11.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

$ chmod +x /usr/local/bin/docker-compose
```


docker-compose.yml

어플리케이션을 만드는 서비스를 정의하는 yaml 파일

Version	Compose Version	Docker version	특징
1	xx < 1.6.0		volumes, networks, build 를 사용할 수 없다.
2	1.6.0 < xx < 1.10.0	1.10.0+	depends_on variable substitution
2.1		1.12.0+	link local ips, isolation, labels, userns mode, healthcheck, sysctls
3	1.10.0 < xx		제거 : volume_driver, volumes from, cpu_shares, cpu_quota, cpuset, mem_limit, memswap_limit 추가: deploy

docker-compose.yml 예제

```
web:
  build: .
  ports:
   - "5000:5000"
  volumes:
   - .:/code
  links:
   - redis
redis:
  image: redis
```

Version 1

```
version: '2'
services:
  web:
    build: .
    ports:
     - "5000:5000"
    volumes:
     - .:/code
  redis:
    image: redis
```

Version 2

```
version: "3"
services:
  db:
    image: postgres
    volumes:
      - data:/var/lib/postgresql/data
volumes:
  data:
    driver: mydriver
```

Version 3

docker-compose.yml Cheat Sheet 1/2

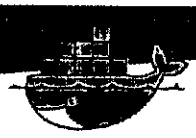


image build 할 이미지를 지정한다.

```
image: webapp:tag
```

build 빌드에 적용할 옵션을 설정한다.

```
build:
  context: ./dir
  dockerfile: mysql.yml
  args:
    buildno: 1
```

links 다른 서비스에 컨테이너를 연결한다.

SERVICE:ALIAS

```
web:
  links:
    - mysql_1:db
    - redis
```

external-links 현재 docker-compose.yml 외부에서 시작된 컨테이너를 연결한다.

```
external_links:
  - mysql_1:mysql
```

ports 포트를 드러낸다.

```
ports:[
  - " 8000:8000 "
  - " 9090-9091:8080-8081 "
```

expose 호스트 바인딩없이 포트를 드러낸다.

```
exposeUME:
  - " 3000 "
```

volumes 호스트와 컨테이너 특정 패스를 마운트한다.

```
volumes:
  - /opt/data:/var/lib/mysql
```

volumes_from 다른 서비스나 컨테이너로부터 볼륨을 마운트한다.

```
volumes_from:
  - mysql_dev
  - service_name:ro
```

docker-compose.yml Cheat Sheet

environment 환경변수를 정의한다.

```
image: webapp:tag
web:
  environment:
    - TAG=1
web:
  image: " webapp:${TAG} "
```

env_file 환경변수를 정의하는 파일을 지정한다.

```
env_file: .env

env_file:
  - ./common.env
```

extends 다른 서비스 설정을 상속받는다.

```
extends:
  file: common.yml
  service: webapp
```

net 네트워크 모드

```
net: " bridge "
net: " host "
```

dns 사설 dns 서버를 지정한다.

```
dns: 8.8.8.8
dns:
  - 8.8.8.8
  - 9.9.9.9
```

dns_search 사설 dns 서버를 도메인으로 지정한다.

```
dns_search: example.com
dns_search:
  - dc1.example.com
  - dc2.example.com
```

도커 컴포즈 주요 명령어

- docker-compose up -d
- docker-compose up -d mydb
- docker-compose up -d —no-recreate
- docker-compose down
- docker-compose rm

link의 원리

↳ 권장하지 않음

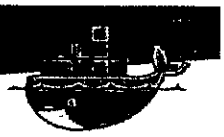
link를 걸면 도커 엔진이 컨테이너에 환경 변수로 필요한 정보를 주입 시킨다.

```
docker-compose run SERVICE  env
```

* 이 환경변수를 직접 사용하지 말것

https://docs.docker.com/compose/link-env-deprecated/

[실습1-25] compose를 컨테이너로 실행



컨테이너로 설치한다.

```
$ $ curl -L https://github.com/docker/compose/
releases/download/1.11.2/run.sh > /usr/local/bin/
docker-compose

$ chmod +x /usr/local/bin/docker-compose
```

[실습1-26] DB 실행

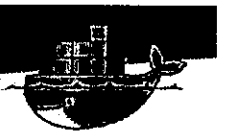
mysql.yml 파일을 정의한다.

```
mydb:
  image: mysql:5.6
  environment:
   - MYSQL_ROOT_PASSWORD=Passw0rd
  ports:
   - 3306:3306
  volumes:
   - /home/docker/my.cnf:/etc/mysql/conf.d/my.cnf
   - /db_master:/var/lib/mysql
```

mysql 컨테이너를 실행한다.

```
$ docker-compose -f mysql.yml up -d
```

[실습1-27] Web/Was 실행



docker-compose.yml 파일을 정의한다.

```
myweb:
  image: nginx:latest
  ports:
   - 80:80
  volumes:
   - /home/docker/nginx.conf:/etc/nginx/nginx.conf
  links:
   - mywas:tomcat
mywas:
  image: tomcat:8.0
  ports:
   - 8080:8080
```

```
docker-compose up -d
```

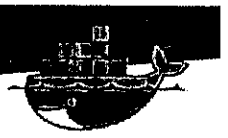
[실습1-28] WAS 스케일아웃

mywas 컨테이너를 3개로 늘린다.

```
docker-compose scale mywas=3
```

Q: myweb이 mywas를 번갈아 바라보도록 하려면?

[실습28] 워드프레스 실행



version2 docker-compose.yml 파일을 정의한다.

```
version: '2'
services:
  db:
    image: mysql:5.7
    volumes:
      - "./.data/db:/var/lib/mysql"
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: wordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    links:
      - db
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_PASSWORD: wordpress
```

[실습29] Web/DB 로깅

myweb, mydb 컨테이너를 로그를 확인한다.

```
docker-compose logs
```

○ CSP Download 지원 20시 ~08 : 6간 11초

- 조회, TRMS
- Singe Daemon

○ Disk I/O / DB plan

- Date 필요
- Download
- CBH :

Q&A

espressobook