

Docker Containers 소개

November, 2017

Younggyu Kim

Principal Sales Consultant
Sales Consulting, Cloud Platform
Seoul, Korea

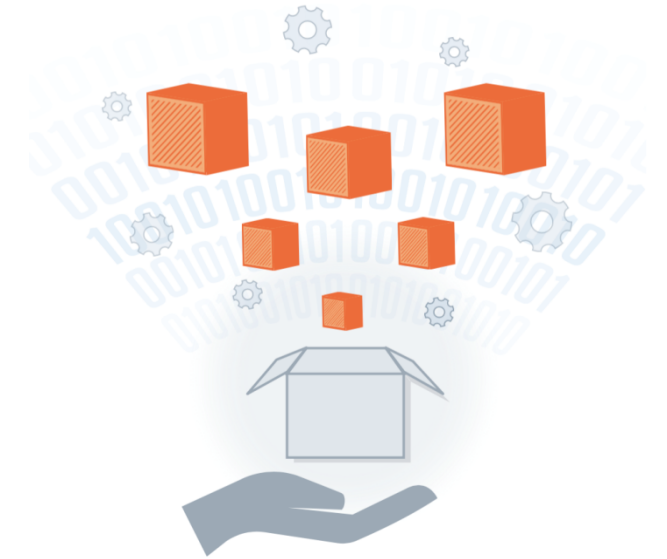


Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

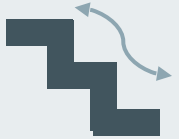







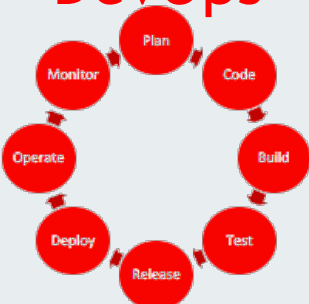
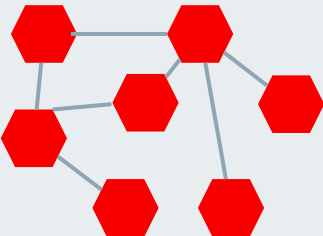
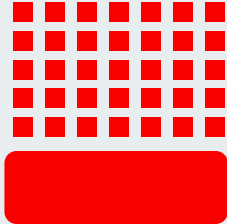
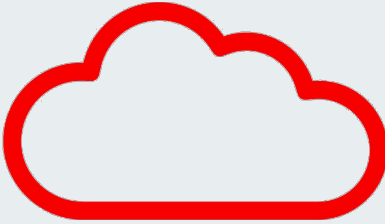
Program Agenda

- 1 Container의 역사
- 2 Container를 이용한 우수한 Use Case
- 3 기본 아키텍처 및 용어 설명
- 4 Docker가 인기있는 이유
- 5 추가 리소스
- 6 Q & A



Container의 역사

컴퓨팅의 역사와 다차원 적 진화

개발 프로세스	응용 프로그램 아키텍처	배포 및 패키징	애플리케이션 인프라
<p>Waterfall</p> 	<p>Monolithic</p> 	<p>Physical Server</p> 	<p>Datacenter</p> 
<p>Agile</p> 	<p>N-Tier</p> 	<p>Virtual Servers</p> 	<p>Hosted</p> 
<p>DevOps</p> 	<p>Microservices</p> 	<p>Containers</p> 	<p>Cloud</p> 

유닉스 Container의 타임라인

Docker는 회사이자 기술입니다.

*Docker*는 *Linux* 컨테이너 기술 도입의 핵심적인 역할을 했지만, 컨테이너 개념을 발명 한 것은 아닙니다.

그러나 그들은 사람이 이 기술을 소비가능하게끔 만들었습니다.



Containers를 이용한 우수한 사용 사례

Ready to Run Application Stacks

- Dev/ Test 설정에 우수함
- Hours/Days가아닌 초단위의 배포
- 빠른 Start Up, Tear Down

New App Dev & Microservices

- 기존 앱의 전체 또는 일부 리팩터링
- Microservices에 적합한 Containers

One-Time Run Jobs and Analytics

- 작업 / 분석을 실행하고 종료

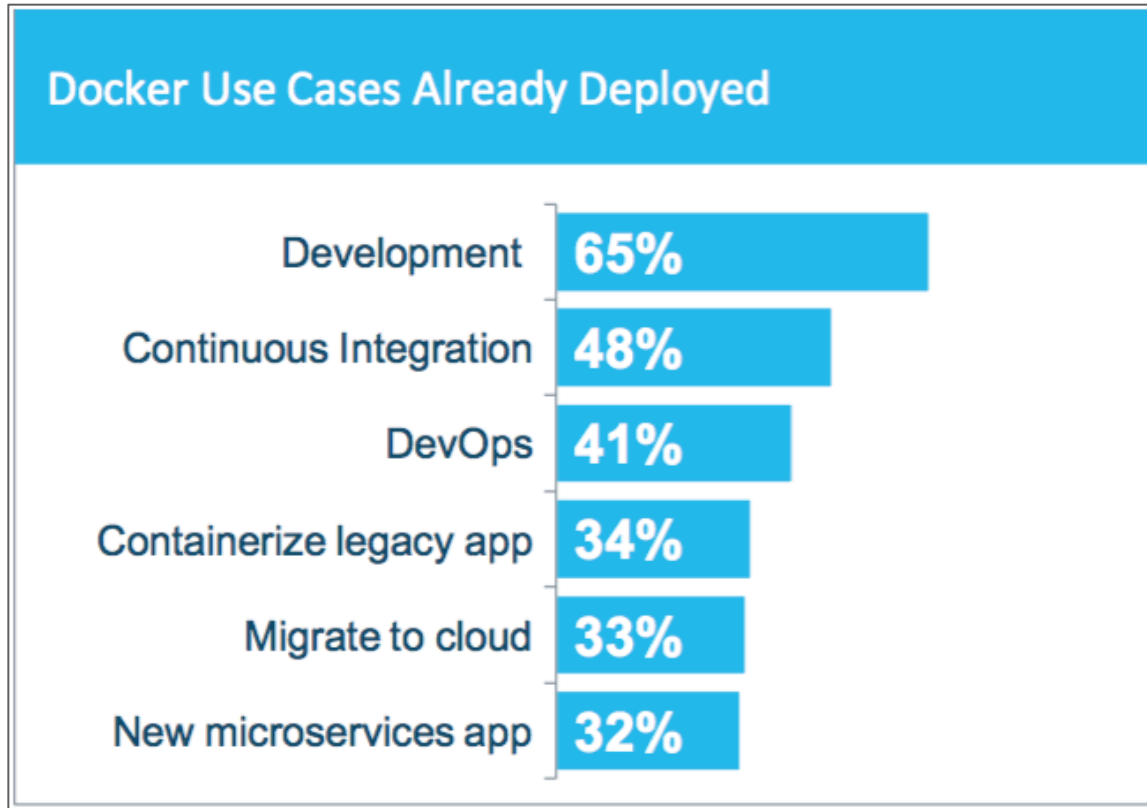
Front-End App Servers

- Highly horizontally scalable
- Cattle Not Pets
- Fast A/B, Rolling Deployments
- CX 최적화
- Traditional Technologies - MW/Backend

Server Density

- 동적 포트를 사용하는 Containers
- 서버에서 동일한 앱을 여러 개 실행

컨테이너가 사용 되고 있는 사례- 설문 조사 :

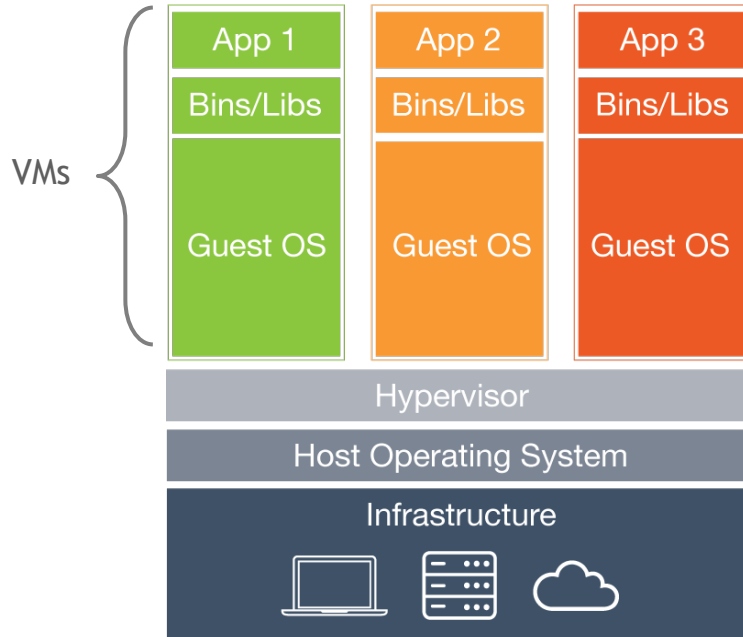


SOURCE: THE EVOLUTION OF THE MODERN SOFTWARE SUPPLY CHAIN, DOCKER SURVEY 2016

- 개발자 생산성은 오늘날 최고의 사용 사례
- CI / CD 파이프 라인 구축
 - 일관된 container image가 파이프 라인을 통해 이동
 - “개발할 땐 동작했는데요?” 신드롬을 예방
- 어플리케이션 현대화와 이식성이 도입을 이끈 핵심 요인 (Prem <---> Cloud)

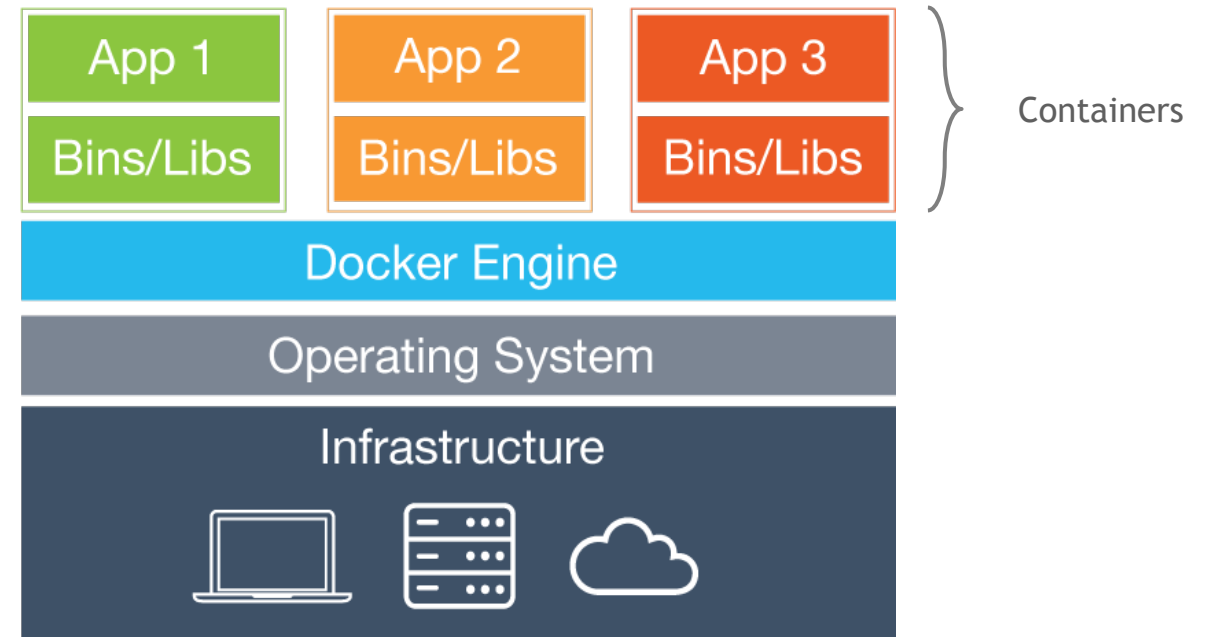
기본 아키텍처 및 용어 설명

가상머신 vs. Container



가상머신

- 각 가상 컴퓨터 (VM)에는 응용 프로그램, 필요한 바이너리, 라이브러리 및 전체 게스트 운영 체제가 포함되어 있습니다.

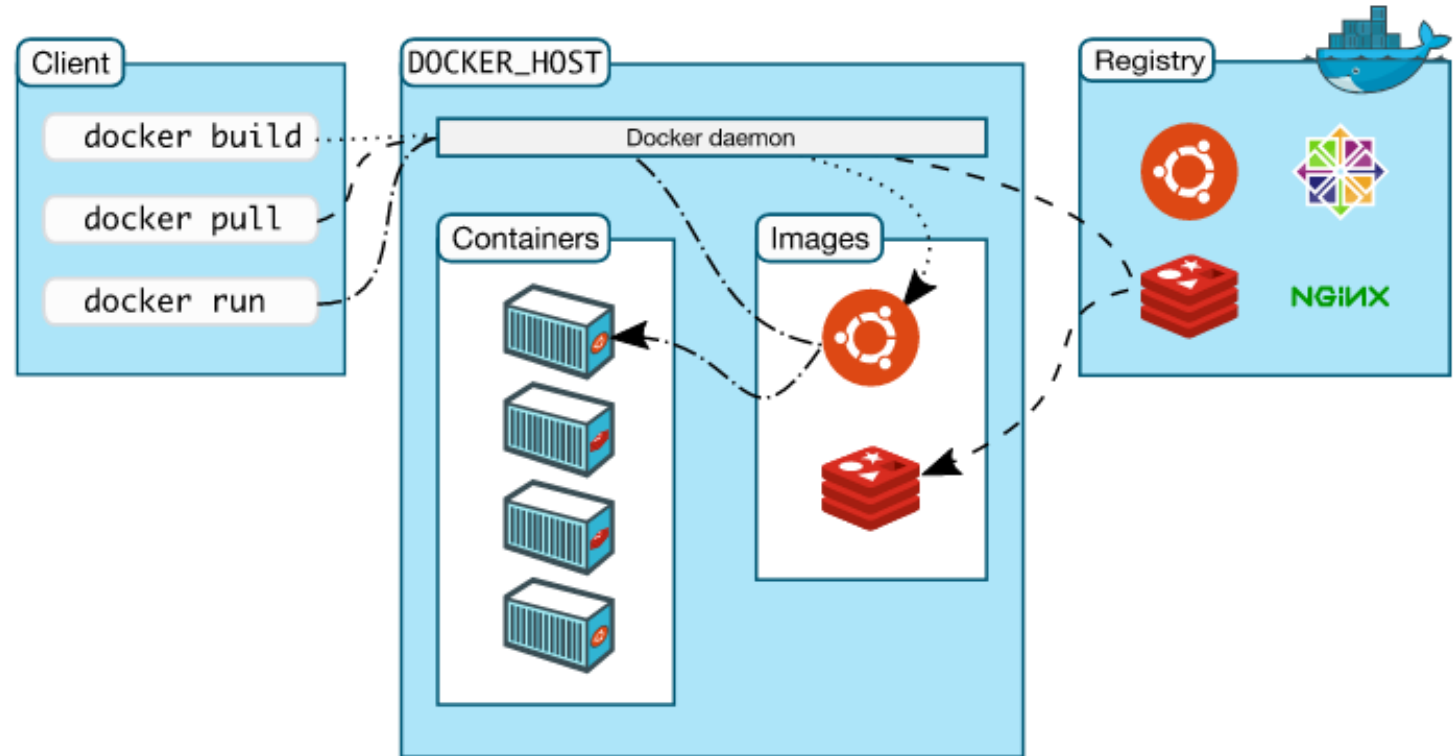


Container

- 컨테이너는 app과 앱에 관련된 dependencies를 포함하지만 커널은 다른 컨테이너와 공유합니다.
- 호스트 OS의 사용자 공간에서 격리된 프로세스로서 실행
- 특정 인프라에 묶여 있지 않음. 컨테이너는 모든 컴퓨터, 인프라 및 클라우드에서 실행 가능

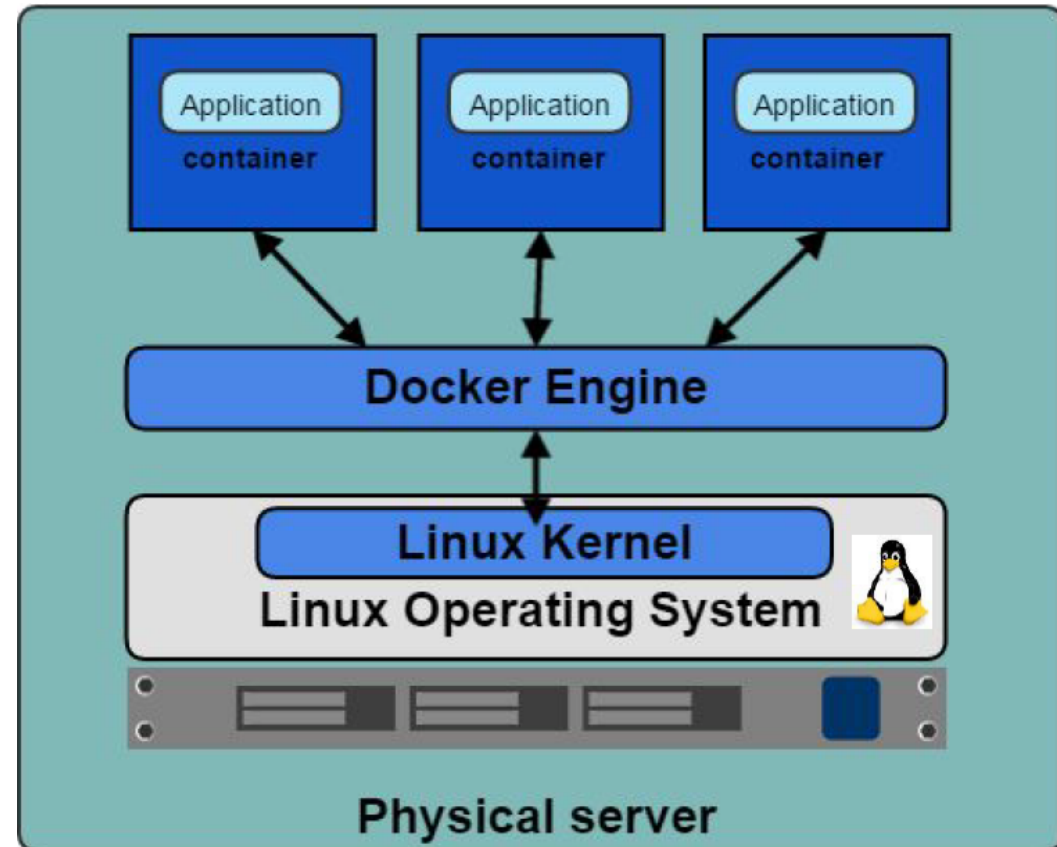
Docker 아키텍처

- Docker 클라이언트 - Docker와의 인터페이스를 위한 명령 행 인터페이스 (CLI)
- Dockerfile - Docker 이미지를 어셈블하는 데 사용되는 Docker 명령으로 구성된 텍스트 파일
- Image - Dockerfile에 정의된 명령어들을 docker build CLI를 통해 빌드된 계층적구조의 파일
- Container - docker run 명령을 사용하여 이미지 인스턴스 실행
- Registry - Image 저장소



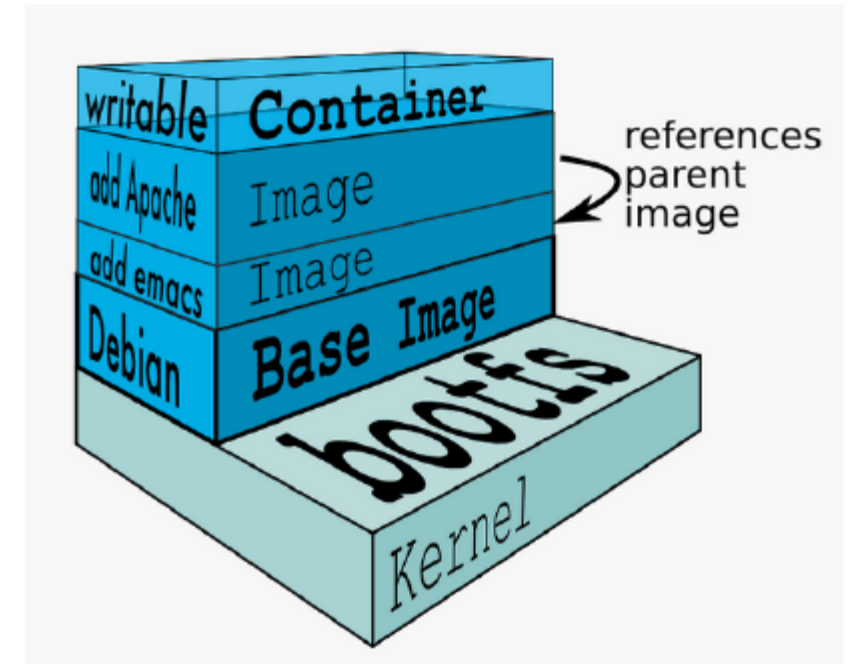
Docker 엔진

- 컨테이너 실행 및 관리
- 리눅스 커널 네임 스페이스와 컨트롤 그룹 사용
- 고립 된 작업 공간을위한 네임 스페이스 제공



Docker Images

- 이미지는 파일들과 일부 메타 데이터의 모음
- 이미지는 다중 레이어이며, 이 다중 레이어는 다른 이미지를 참조 / 기반으로함(유니온 파일 시스템)
- 각 이미지에는 실행할 소프트웨어가 포함되어 있음.
- 모든 이미지에는 기본 레이어가 포함되어 있음
- 레이어는 읽기 전용



Dockerfile - 도커이미지를 만드는 텍스트 파일 (라서피)

Example Hello World Dockerfile

```
FROM nginx:1.10.1-alpine  
Add index.html /usr/share/nginx/html/  
index.html  
# Override the nginx start from the base  
container  
COPY start.sh /start.sh  
RUN chmod +x /start.sh  
ENTRYPOINT ["/start.sh"]
```

Source: <https://github.com/scottsbaldwin/docker-hello-world/blob/master/Dockerfile>

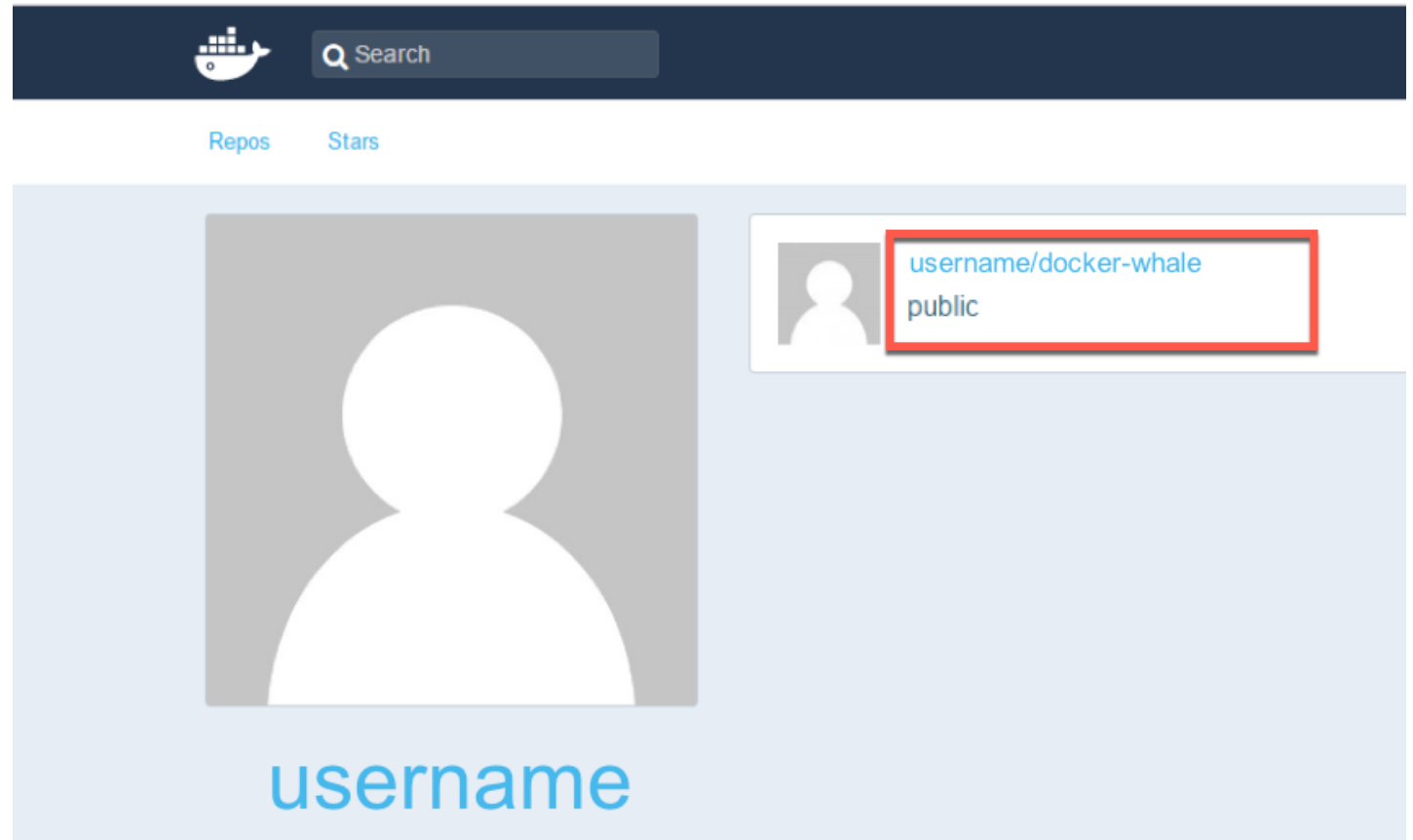
Docker build image CLI example

```
$ docker build -t helloworld:1.0 .
```

NOTE: The “.” references Dockerfile in local directory

Docker Hub

- Docker Inc.
 - Repository
 - public and private images
- 랩탑에서 이미지 공유 및 이동
- Example usage:
 - `$ docker tag docker-whale:latest username/docker-whale:latest`
 - `$ docker push username/docker-whale:latest`
 - `$ docker pull username/docker-whale:latest`



Docker CLI - 일반적인 / 유용한 명령

- `docker build` : build docker image from Dockerfile
- `docker run` : run docker image
- `docker logs` : show log data for a running or stopped container
- `docker ps` : list running docker containers (analogous to `ps`)
- `docker ps -a` : list all containers including not running
- `docker images` : list all images on the local volume
- `docker rm` : remove/delete a container | `docker rmi` : remove/delete an image
- `docker tag` : name a docker image
- `docker login` : login to registry
- `docker push/pull` : push or pull volumes to/from Docker Registries
- `docker inspect` : return container run time configuration parameter metadata

See the docs here: <https://docs.docker.com/edge/engine/reference/commandline/docker/>

Docker Run

이미지를 Repository에서 가져와 컨테이너로 실행.

Examples:

– Simple:

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

– Complex:

```
$ docker run -d --restart=always -p=443:5000/tcp  
-e="REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt"  
-e="REGISTRY_HTTP_TLS_KEY=/certs/registry.example.com.key"  
-e="REGISTRY_AUTH=htpasswd"  
-e="REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd"  
-e="REGISTRY_AUTH_HTPASSWD_REALM=Our Test Registry"  
-v=/home/opc/certs:/certs -v=/home/opc/auth:/auth  
-v /home/opc/registry:/var/lib/registry "registry:2"
```

Docker Compose

- Docker Compose
 - multi-container Docker를 정의하고 실행하기 위한 Docker Tool
 - YAML에 정의된 참조 파일
 - docker-compose.yml
- \$ docker-compose up -d

```
version: '2'

services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/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
    ports:
      - "80:80"
    restart: always
    volumes:
      - /var/www/html:/var/www/html:rw
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_PASSWORD: wordpress
  volumes:
    db_data:
```

Docker가 인기있는 이유

Docker가 인기있는 이유 - Devs가 좋아하는 간단함



레거시 앱 테스트 / 개발

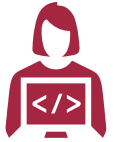
새로운 앱 개발 (기존 앱의 일부 포함)

Code민첩성, CI / CD 파이프 라인, DevOps

오픈 소스 도입

마이크로 서비스 및 클라우드 네이티브 앱

왜 Container를 사용할까요?



Developers가 좋아하는 이유:

- 바로 실행할 수 있는 패키지 응용 프로그램을 신속하게 생성하고, 저렴한 비용으로 배포 및 재생 가능.
- 테스트, 통합, 패키징 자동화
- 플랫폼 호환성 문제("Dev에서는 작동해요") 를 줄이거 나 없앨 수 있음.
- 차세대 애플리케이션 지원 (마이크로 서비스)



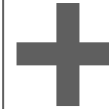
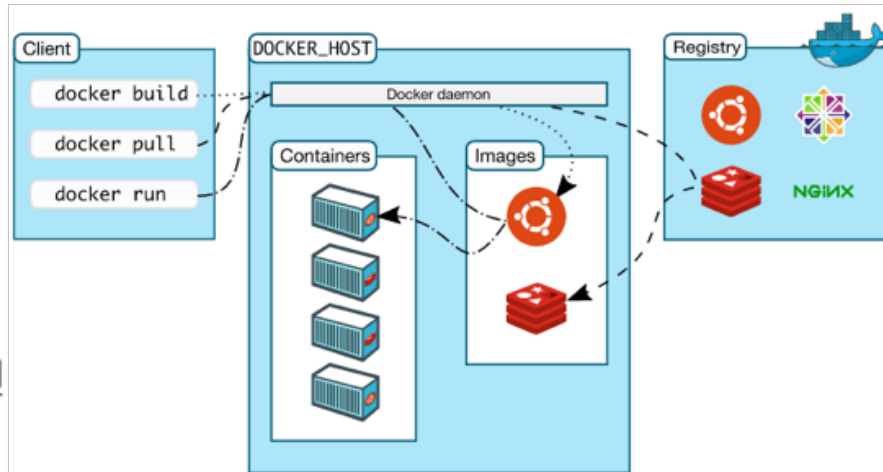
IT가 좋아하는 이유:

- 배포 속도 및 릴리즈 빈도 개선과 배포 안정성 향상
- 앱 라이프 사이클을 효율적이고 일관성 있게 반복 가능하게 할 수 있음. 한 번 구성하고 여러 번 실행.
- 개발, 테스트, 생산 간의 환경 불일치 제거
- 애플리케이션의 탄력성 향상과 온디맨드 scale in/out

컨테이너는 이동 가능 하지만 고급 기능은 어떨까요?

Docker 아키텍처

- Docker 클라이언트 - Docker와의 인터페이스를 위한 명령 행 인터페이스 (CLI)
- Dockerfile - Docker 이미지를 어셈블하는 데 사용되는 Docker 명령으로 구성된 텍스트 파일
- Image - docker 빌드 커맨드의 인풋인 Dockerfile을 통해 빌드된 계층적구조의 파일
- Container - docker run 명령을 사용하여 이미지 인스턴스 실행
- Registry - Image 저장소



Advanced Functions

- 오케스트레이션, 모니터링, 운영, 서비스 검색
- Docker 환경 프로비저닝

Fragmented Market Solutions

- Kubernetes
- Swarm, Docker Data Center, Docker Cloud
- Consul, ETCD, Docker Networking
- etc

Oracle Cloud와 Docker Containers

Compute CS



DIY Container Management

IaaS

Container CS



ORACLE Container Cloud Service

Dashboard

4 Deployments OK
All deployments have passed their health checks.

2 Hosts OK
All hosts are active and reachable.

3 Resource Pools

Name	Hosts
default	2
Development	0
Production	0

Category	Count
Services	27
Stacks	4
Deployments	4
Resource Pools	3

Oracle Managed Container Service

CaaS

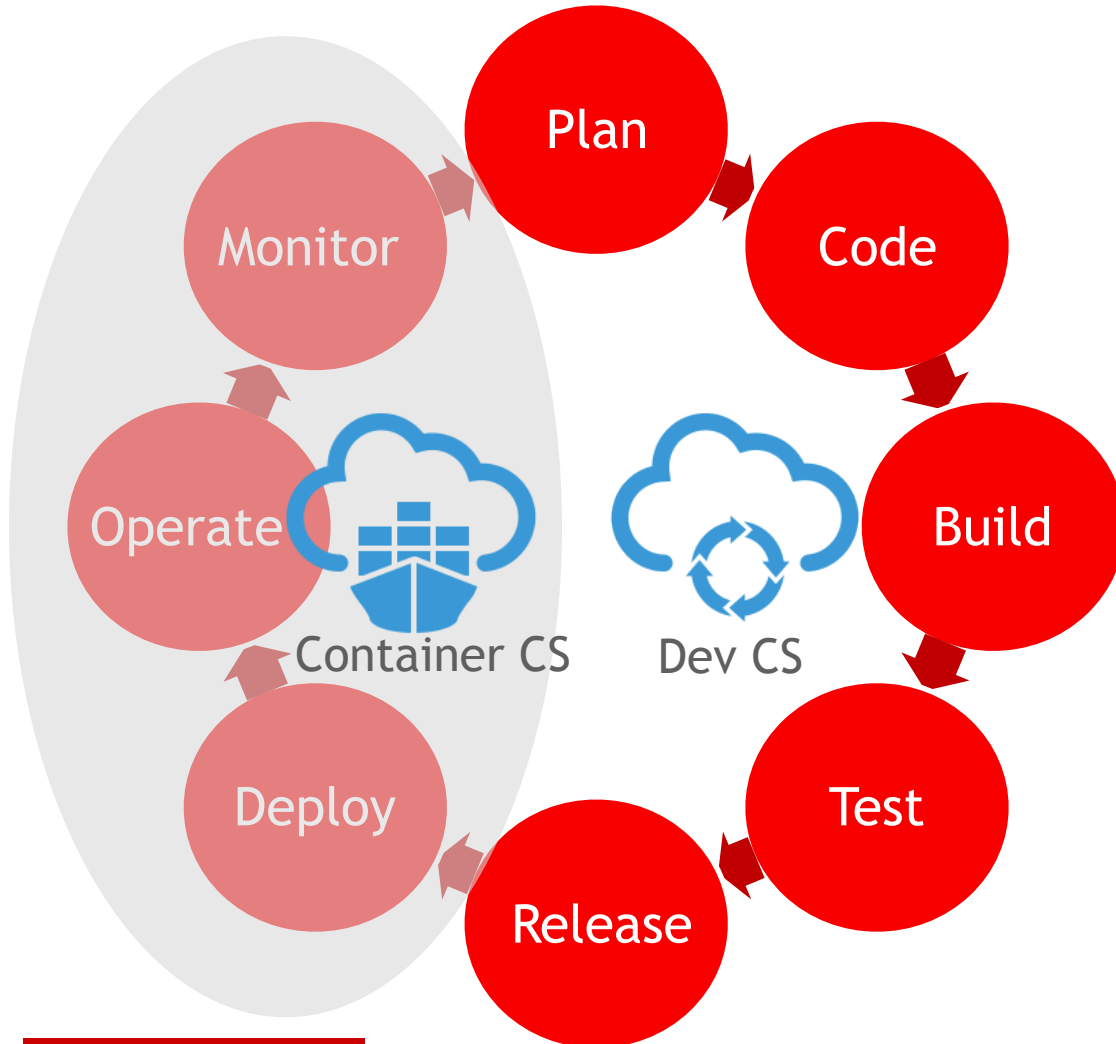
Application Container CS



Docker-based Cloud Polyglot Platform

PaaS

The Docker DevOps Cycle



OPC services featuring:

- Oracle Developer Cloud Service
 - Docker Image Build 기능 포함 *
- Oracle Container Cloud Service
 - 배포, 운영, 모니터링

추가 리소스

Resource	Location
Entry Level Hands-on Lab	https://github.com/oracle/cloud-native-devops-workshop/tree/master/containers/docker001
Oracle Container Cloud Service	https://cloud.oracle.com/en_US/container
Official Image Registries	Oracle Images on the Docker Store Oracle Container Registry

질문있으세요?

ORACLE®

Hands On Lab

ORACLE®

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |



Install Docker on VirtualBox

VirtualBox 실행

Install docker & docker-compose

```
$ sudo apt-get install docker.io
```

```
$ sudo docker --version
```

```
Docker version 1.13.1, build 092cba3
```

```
$ sudo apt-get install docker-compose
```

```
docker-compose version 1.8.0, build unknown
```

Add docker group & add user to docker group

```
$ docker image ls (it causes Permission error)
```

```
$ cat /etc/group
```

```
$ sudo groupadd docker (in case 'docker' group does not exists in the above file.)
```

```
$ sudo gpasswd -a dku docker
```

```
$ sudo service docker restart
```

==> Log out and Log in again

```
$ docker image ls
```