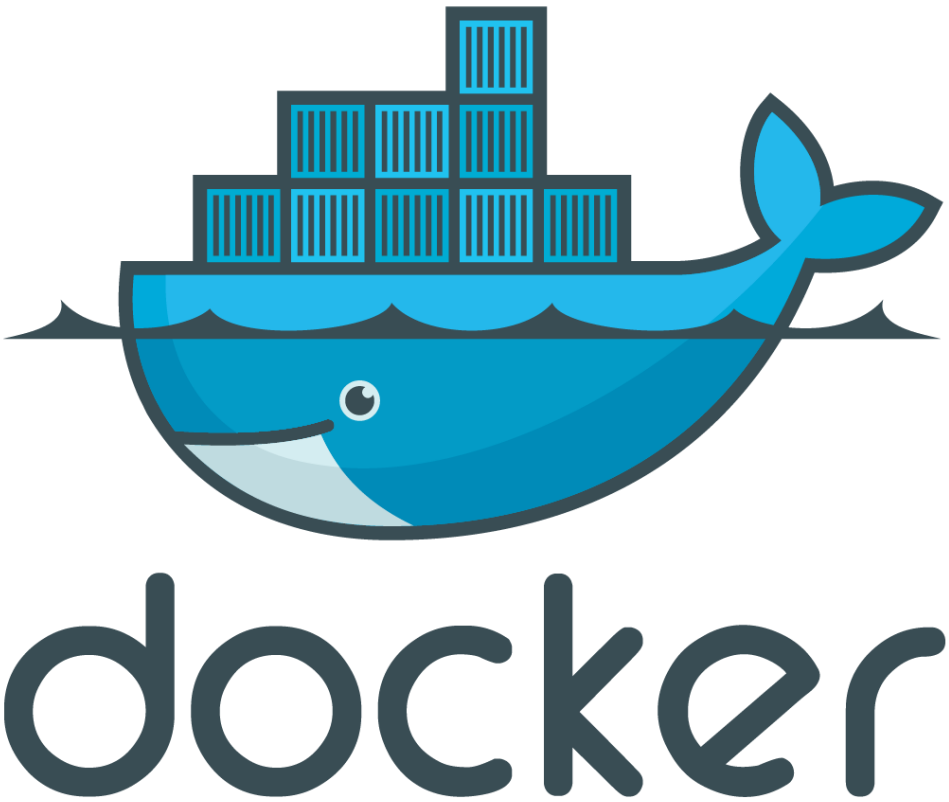
8 min read

초보를 위한 도커 안내서 - 도커란 무엇인가? SERIES 1/3

subicura on 19 Jan 2017



docker logo

어느날 친구한테 메시지를 받았습니다.

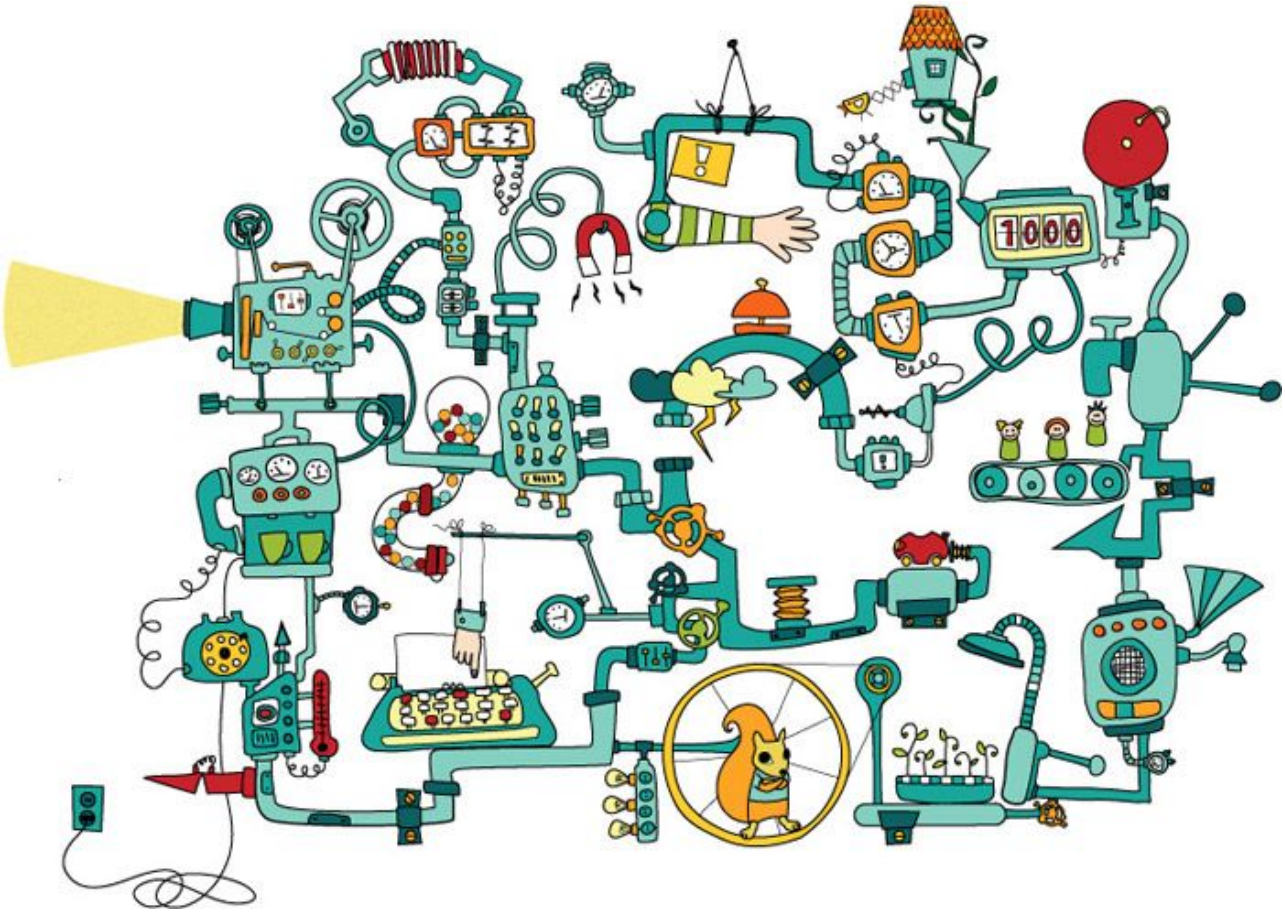
> 도커 공부 좀 하려는데 `hello world` 문서 어떤 거 보면서 시작하는 게 좋음?

ㄱ글에서 docker 검색ㄱㄱ 라고 말하려다 인터넷에 있는 도커 관련 글과 동영상 중에 입문자 링크를 몇 개 전달해 주었습니다. 이후에도 몇 번 비슷한 요청을 받으면서 도커에 대해 가볍게 정리해보자 라는 생각이 들었고 예전 글(도커를 이용한 웹서비스 무중단 배포하기)과 밋업 때 발표했던 내용, 그리고 그동안의 사용경험을 모아 글을 작성하게 되었습니다.

이 글은 도커에 대해 1도 모르는 시스템 관리자나 서버 개발자를 대상으로 도커 전반에 대해 얕고 넓은 지식을 담고 있습니다. 도커가 등장한 배경과 도커의 역사, 그리고 도커의 핵심 개념인 컨테이너와 이미지에 대해 알아보고 실제로 도커를 설치하고 컨테이너를 실행해 보도록 하겠습니다.

- **초보를 위한 도커 안내서 - 도커란 무엇인가? ✓** SERIES 1/3
- 초보를 위한 도커 안내서 - 설치하고 컨테이너 실행하기 SERIES 2/3
- 초보를 위한 도커 안내서 - 이미지 만들고 배포하기 SERIES 3/3

서버를 관리한다는 것



복잡하고 어려운 서버관리

일반적으로 서버를 관리한다는 건 복잡하고 어려우며 고오급 개발자들의 섬세한 작업이 필요한 영역입니다.

2006년 병역특례를 시작하고 맨 처음 했던 일은 매뉴얼을 보고 Redhat Enterprise Linux 4에 Oracle 10g을 설치하는 일이였습니다. 정확히 기억이 나지는 않지만 설치 매뉴얼은 길고 복잡했고 알 수 없는 이유로 자꾸 설치를 실패하였습니다. 제대로 설치가 되지 않으면 다시 OS를 설치하는 것부터 반복하여 몇 번을 재설치한 끝에 성공하곤 했습니다. 회사에서 사용하는 리눅스와 오라클 버전은 딱 정해져 있었고 버전을 업데이트 하는 건 엄청난 리스크였기 때문에 서버는 최대한 건드리지 않고 그대로 두는 게 최선의 방법이였습니다.

새로운 서버를 셋팅하는 날은 밤을 새는 날이였고 몇 번 밤을 새다보니 `./configure` 와 `make && make install` 의 달인이 되어 있었습니다. 어느 정도 익숙해졌다고 생각한 시점에도 리눅스 배포판이 바뀌거나 환경이 달라지면 꼭 문제가 생기곤 했습니다.

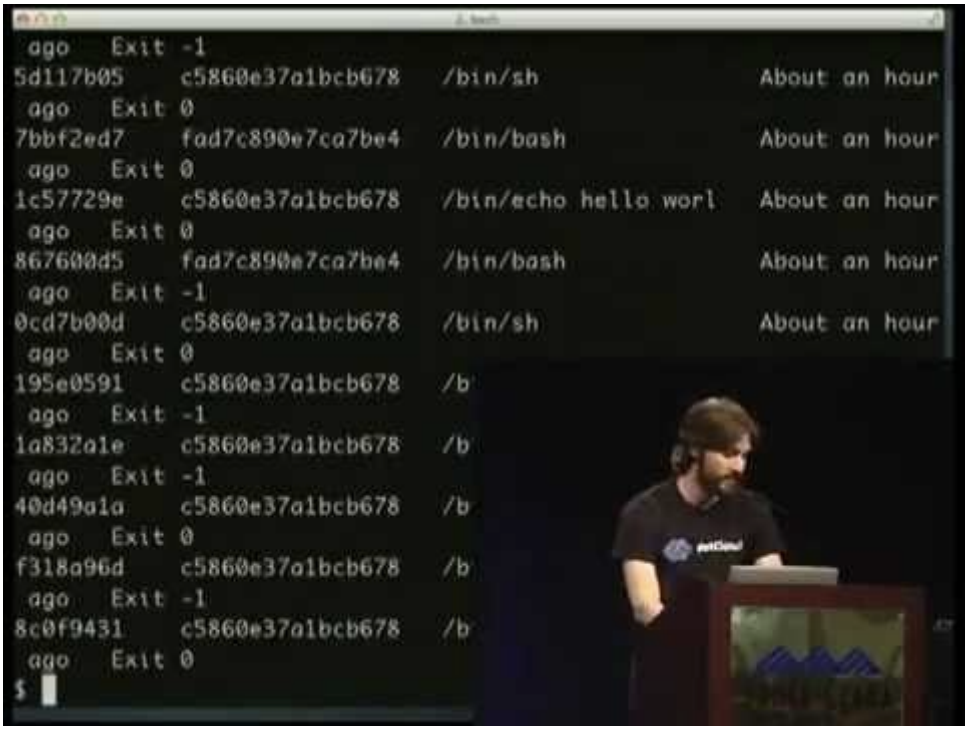
하나의 서버에 여러개의 프로그램을 설치하는 것도 문제였는데 서로 사용하는 라이브러리의 버전이 다르거나 동일한 포트를 사용하는 경우는 설치가 굉장히 까다로웠습니다. 차라리 서로 다른 서버에 설치하는게 나았고 그렇게 조립PC는 늘어나고 자원은 낭비됩니다.

시간이 흐르면서 서버 환경이 계속 바뀌는데 CentOS에 익숙해지면 Ubuntu를 써야하는 일이 생기고 AWS에 익숙해지면 Azure를 써야하는 일이 생깁니다. Chef의 cookbook에 익숙해지

ㄲ Ansible의 playbook을 작성해야 하는 일이 생깁니다. ~~야호~~

DevOps의 등장으로 개발주기가 짧아지면서 배포는 더 자주 이루어지고 마이크로서비스 아키텍쳐가 유행하면서 프로그램은 더 잘게 쪼개어져 관리는 더 복잡해집니다. 새로운 툴은 계속해서 나오고 클라우드의 발전으로 설치해야 할 서버가 수백, 수천대에 이르는 `1` 나누기 `0` ~~답이 없는~~ 같은 상황에서 도커(Docker) 가 등장하고 서버관리 방식이 완전히 바뀌게 됩니다.

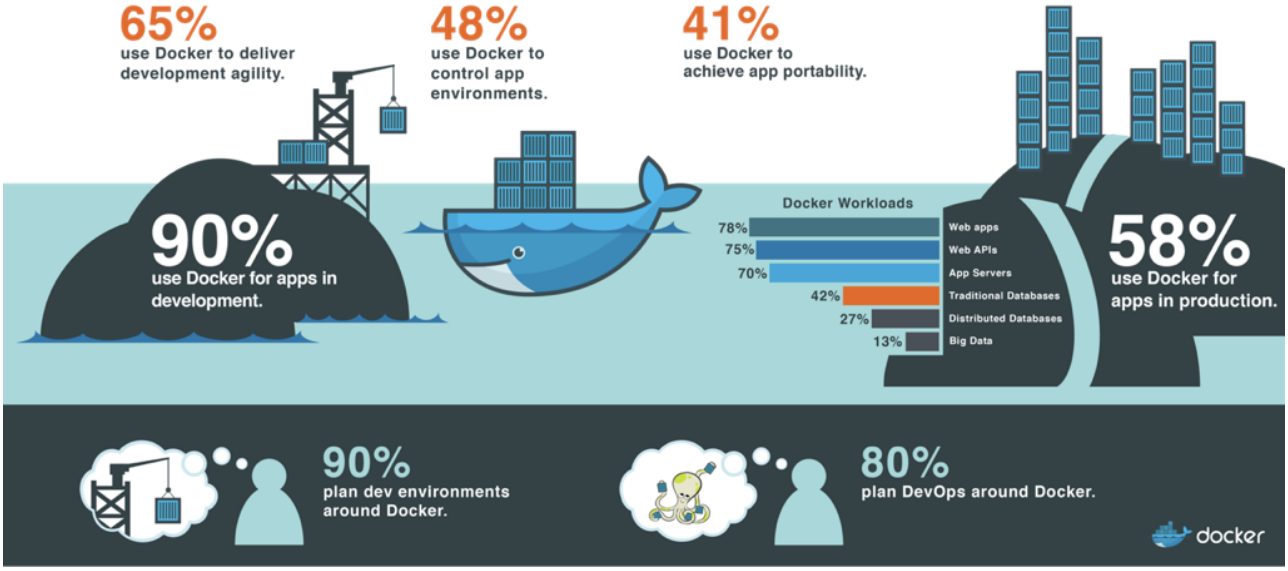
도커의 역사



The future of Linux Containers

도커는 2013년 3월 산타클라라에서 열린 Pycon Conference에서 dotCloud의 창업자인 Solomon Hykes가 The future of Linux Containers 라는 세션을 발표하면서 처음 세상에 알려졌습니다.

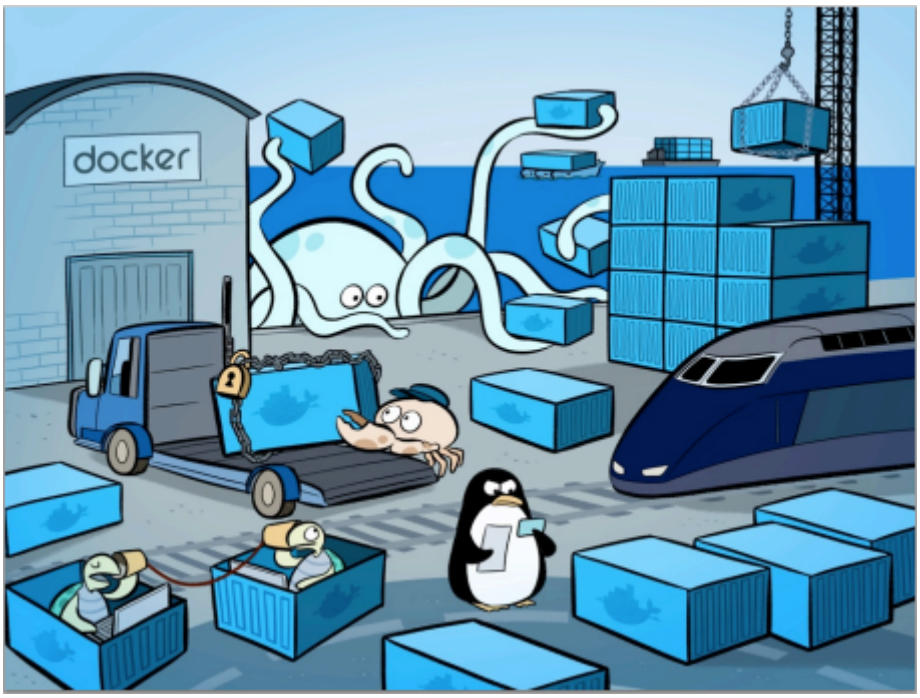
이 발표 이후 도커가 인기를 얻으면서 2013년 10월 아예 회사이름을 도커(Docker Inc.)로 바꾸고 2014년 6월 도커 1.0을 발표합니다. 2014년 8월 도커에 집중하기 위해 dotCloud 플랫폼을 매각하고 2015년 4월 $95M(약 1,100억원) 투자를 유치한 후 계속해서 빠르게 성장하고 있습니다. (현재까지 총 투자액은 $180M이며 2016년 6월 MS에서 $4B/₩4조에 인수하려 했다는 기사가 있습니다.) ~~누가 오픈소스는 돈이 되지 않는다고 했는가?!~~



The Evolution of the Modern Software Supply Chain - The Docker Survey, 2016

~~도커에서한~~ 2016년 설문조사에서 90%가 개발에 사용중이고 80%가 DevOps에 사용할 예정이며 58%가 운영환경에서 사용중이라고 합니다. 2014년 도커 서울 밋업을 시작할 때만 해도 대부분의 사람들이 도커를 잘 모르고 개념도 이해하지 못했는데 이제는 거의 모르는 사람이 없을 정도로 널리 쓰이고 있습니다.

도커란?



도커는 컨테이너를 관리하는 플랫폼

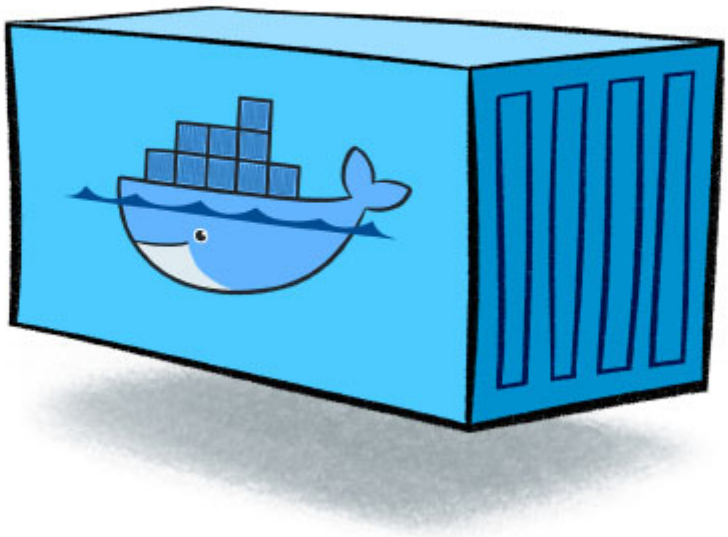
도커는 **컨테이너 기반의 오픈소스 가상화 플랫폼**입니다.

테이너라 하면 배에 실는 네모난 화물 수송용 박스를 생각할 수 있는데 각각의 컨테이너 안에는 옷, 신발, 전자제품, 술, 과일등 다양한 화물을 넣을 수 있고 규격화되어 컨테이너선이나 트레일러등 다양한 운송수단으로 쉽게 옮길 수 있습니다.

서버에서 이야기하는 컨테이너도 이와 비슷한데 다양한 프로그램, 실행환경을 컨테이너로 추상화하고 동일한 인터페이스를 제공하여 프로그램의 배포 및 관리를 단순하게 해줍니다. 백엔드 프로그램, 데이터베이스 서버, 메시지 큐등 어떤 프로그램도 컨테이너로 추상화할 수 있고 조립PC, AWS, Azure, Google cloud등 어디에서든 실행할 수 있습니다.

컨테이너를 가장 잘 사용하고 있는 기업은 구글인데 2014년 발표 에 따르면 구글은 모든 서비스들이 컨테이너로 동작하고 매주 20억 개의 컨테이너를 구동 한다고 합니다. ~~갓구글~~

컨테이너(Container)



docker container

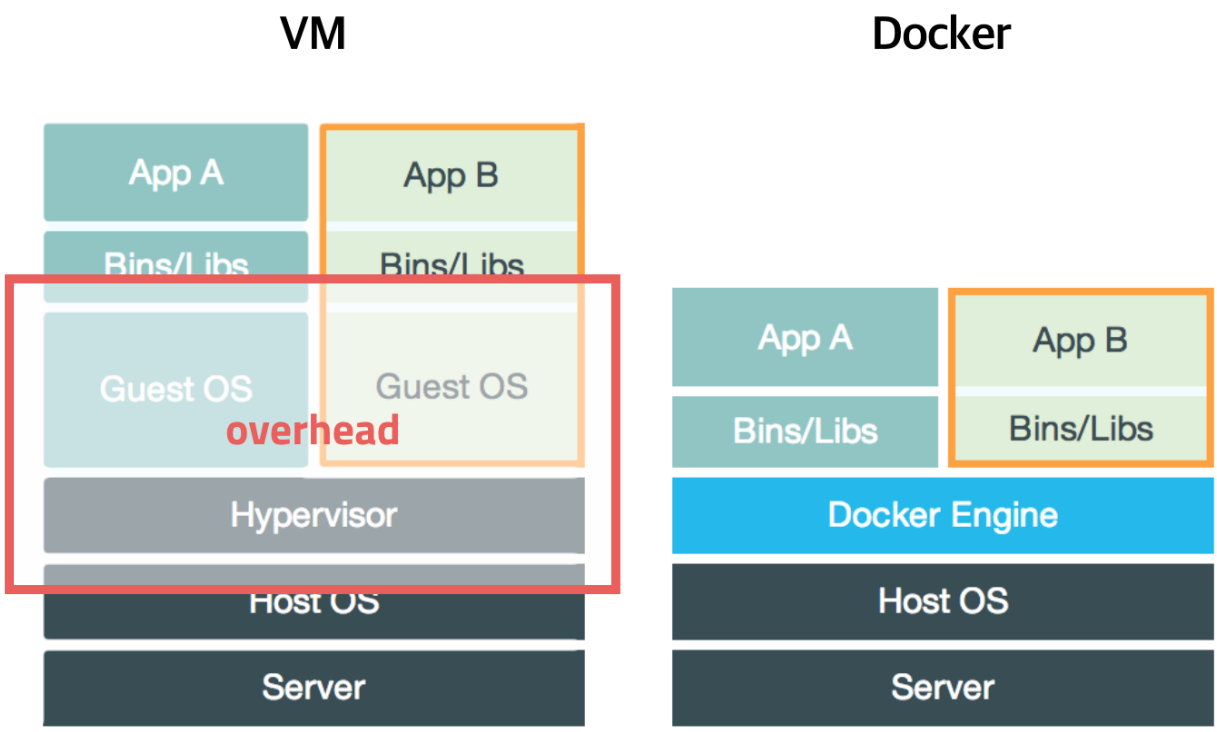
컨테이너는 격리된 공간에서 프로세스가 동작하는 기술입니다. 가상화 기술의 하나지만 기존 방식과는 차이가 있습니다.

기존의 가상화 방식은 주로 **OS를 가상화**하였습니다.

우리에게 익숙한 VMware나 VirtualBox같은 가상머신은 호스트 OS위에 게스트 OS 전체를 가상화하여 사용하는 방식입니다. 이 방식은 여러가지 OS를 가상화(리눅스에서 윈도우를 돌

ㄱ'다던가) 할 수 있고 비교적 사용법이 간단하지만 무겁고 느려서 운영환경에선 사용할 수 없습니다.

이러한 상황을 개선하기 위해 CPU의 가상화 기술(HVM)을 이용한 KVM Kernel-based Virtual Machine과 반가상화 Paravirtualization 방식의 Xen이 등장합니다. 이러한 방식은 게스트 OS가 필요하긴 하지만 전체OS를 가상화하는 방식이 아니였기 때문에 호스트형 가상화 방식에 비해 성능이 향상되었습니다. 이러한 기술들은 OpenStack이나 AWS, Rackspace같은 클라우드 서비스에서 가상 컴퓨팅 기술의 기반이 되었습니다.



가상머신과 도커

전가상화든 반가상화든 추가적인 OS를 설치하여 가상화하는 방법은 어쨌든 성능문제가 있었고 이를 개선하기 위해 **프로세스를 격리** 하는 방식이 등장합니다.

리눅스에서는 이 방식을 리눅스 컨테이너라고 하고 단순히 프로세스를 격리시키기 때문에 가볍고 빠르게 동작합니다. CPU나 메모리는 딱 프로세스가 필요한 만큼만 추가로 사용하고 성능적으로도 거어어어의 손실이 없습니다.

> **도커의 기본 네트워크 모드는 `Bridge` 모드로 약간의 성능 손실이 있습니다. 네트워크 성능이 중요한 프로그램의 경우 `--net=host` 옵션을 고려해야 합니다.**

하나의 서버에 여러개의 컨테이너를 실행하면 서로 영향을 미치지 않고 독립적으로 실행되어 마치 가벼운 VM Virtual Machine을 사용하는 느낌을 줍니다. 실행중인 컨테이너에 접속하여 명

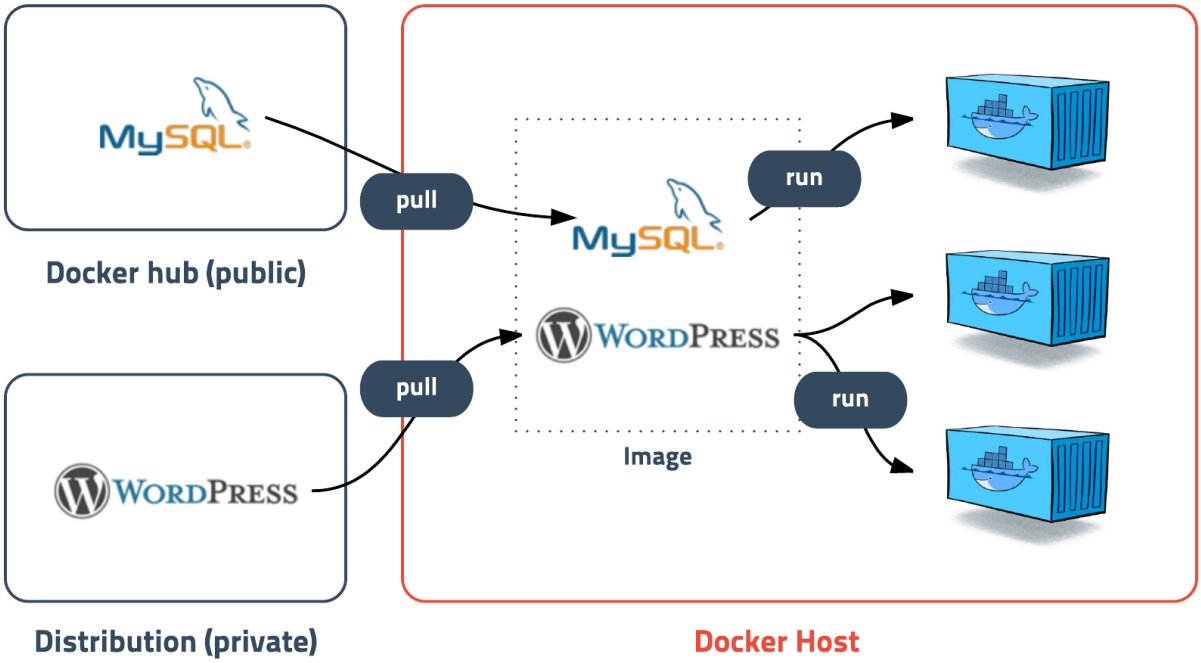
커를 입력할 수 있고 `apt-get` 이나 `yum` 으로 패키지를 설치할 수 있으며 사용자도 추가하 여러개의 프로세스를 백그라운드로 실행할 수도 있습니다. CPU나 메모리 사용량을 제한할 수 있고 호스트의 특정 포트와 연결하거나 호스트의 특정 디렉토리를 내부 디렉토리인 것처럼 사용할 수도 있습니다.

새로운 컨터이너를 만드는데 걸리는 시간은 겨우 1-2초~~(체감 0.001초)~~로 가상머신과 비교도 할 수 없이 빠릅니다.

이러한 컨테이너라는 개념은 도커가 처음 만든 것이 아닙니다. 도커가 등장하기 이전에, 프로세스를 격리하는 방법으로 리눅스에서는 cgroups(control groups)와 namespace를 이용한 LXC(Linux container)가 있었고 FreeBSD에선 Jail, Solaris에서는 Solaris Zones이라는 기술이 있었습니다. 구글에서는 고오오급 기술자들이 직접 컨테이너 기술을 만들어 사용하였고 lmctfy (Let Me Contain That For You)라는 ~~뭐라고 읽어야 할지 알 수 없는~~ 오픈소스 컨테이너 기술을 공개했지만 성공하진 못했습니다.

도커는 LXC를 기반으로 시작해서 0.9버전에서는 자체적인 libcontainer 기술을 사용하였고 추후 runC기술에 합쳐졌습니다.

이미지(Image)



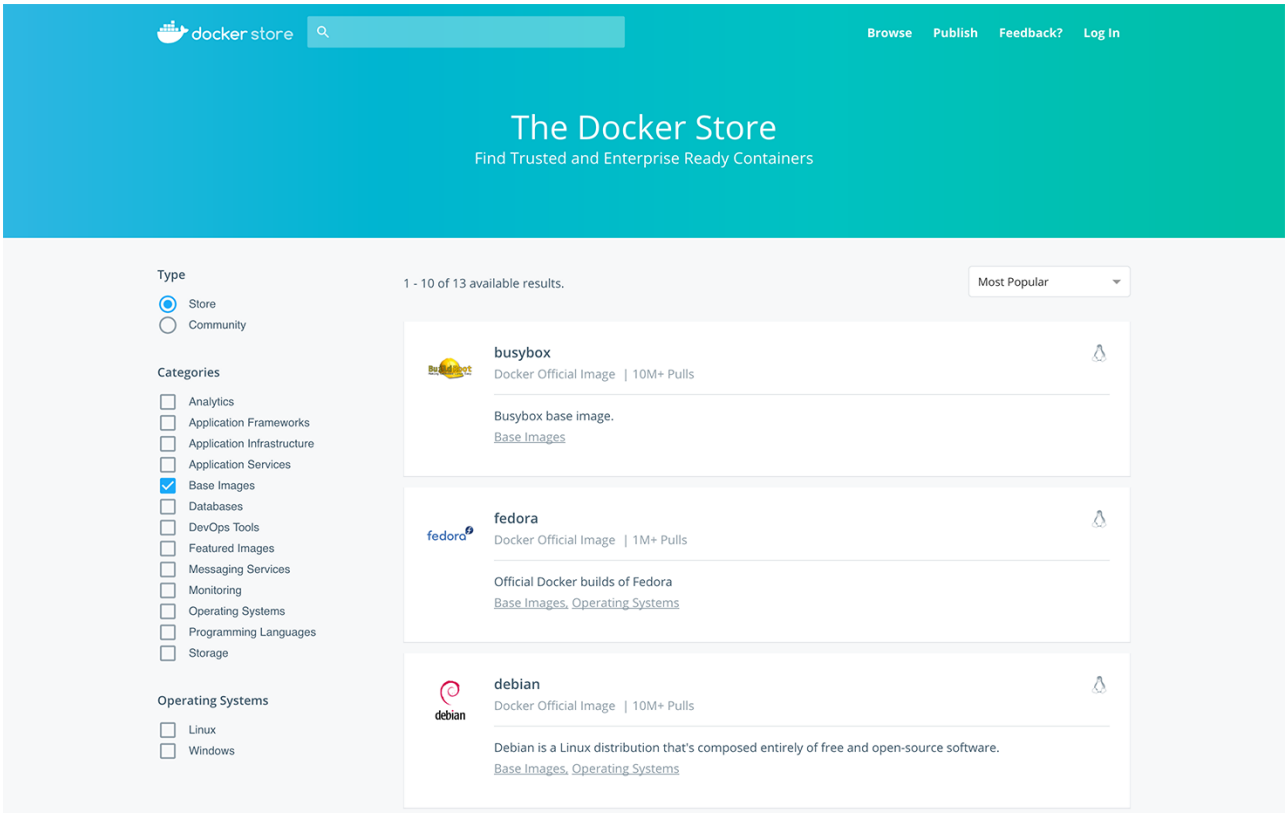
Docker image

도커에서 가장 중요한 개념은 컨테이너와 함께 이미지라는 개념입니다.

이미지는 **컨테이너 실행에 필요한 파일과 설정값등을 포함하고 있는 것**으로 상태값을 가지지 고 변하지 않습니다(Immutable). 컨테이너는 이미지를 실행한 상태라고 볼 수 있고 추가되거나 변하는 값은 컨테이너에 저장됩니다. 같은 이미지에서 여러개의 컨테이너를 생성할 수 있고 컨테이너의 상태가 바뀌거나 컨테이너가 삭제되더라도 이미지는 변하지 않고 그대로 남아있습니다.

ubuntu이미지는 ubuntu를 실행하기 위한 모든 파일을 가지고 있고 MySQL이미지는 debian을 기반으로 MySQL을 실행하는데 필요한 파일과 실행 명령어, 포트 정보등을 가지고 있습니다. 좀 더 복잡한 예로 Gitlab 이미지는 centos를 기반으로 ruby, go, database, redis, gitlab source, nginx등을 가지고 있습니다.

말그대로 이미지는 컨테이너를 실행하기 위한 모오오오오든 정보를 가지고 있기 때문에 더 이상 의존성 파일을 컴파일하고 이것저것 설치할 필요가 없습니다. 이제 새로운 서버가 추가되면 미리 만들어 놓은 이미지를 다운받고 컨테이너를 생성만 하면 됩니다. 한 서버에 여러개의 컨테이너를 실행할 수 있고, 수십, 수백, 수천대의 서버도 문제없습니다.



Docker Store

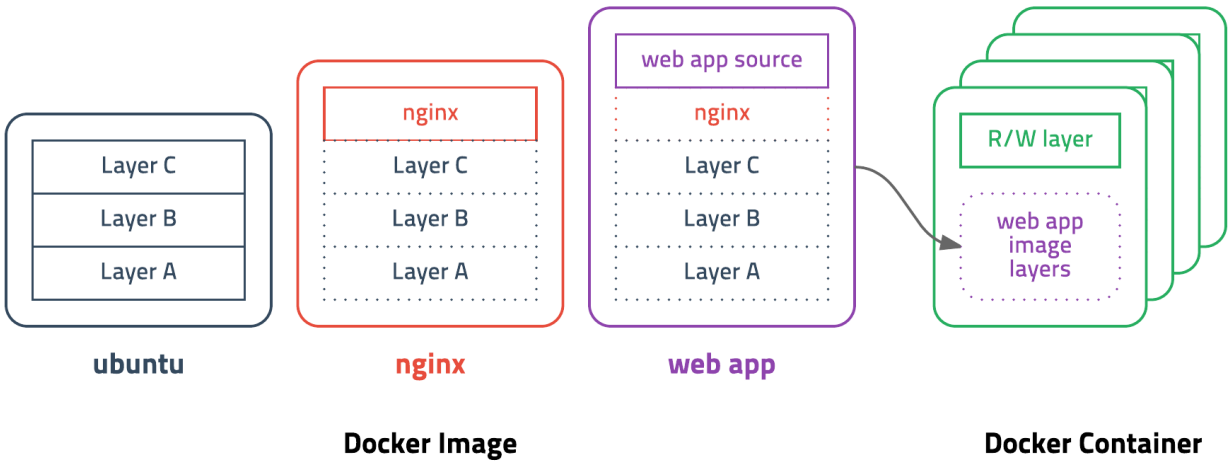
도커 이미지는 Docker hub에 등록하거나 Docker Registry 저장소를 직접 만들어 관리할 수 있습니다. 현재 공개된 도커 이미지는 50만개가 넘고 Docker hub의 이미지 다운로드 수는 80억회에 이릅니다. 누구나 쉽게 이미지를 만들고 배포할 수 있습니다.

이렇게 핫한가?

도커는 완전히 새로운 기술이 아니며 이미 존재하는 기술을 잘 포장했다고 볼 수 있습니다. ~~(마치 최신기술을 잘 포장해서 만든 아이폰처럼)~~

컨테이너, 오버레이 네트워크overlay network, 유니온 파일 시스템union file systems등 이미 존재하는 기술을 도커처럼 잘 조합하고 사용하기 쉽게 만든 것은 없었고 사용자들이 원하는 기능을 간단하지만 획기적인 아이디어로 구현하였습니다.

레이어 저장방식



Docker Layer

도커 이미지는 컨테이너를 실행하기 위한 모든 정보를 가지고 있기 때문에 보통 용량이 수백메가MB에 이릅니다. 처음 이미지를 다운받을 땐 크게 부담이 안되지만 기존 이미지에 파일 하나 추가했다고 수백메가를 다시 다운받는다면 매우 비효율적일 수 밖에 없습니다.

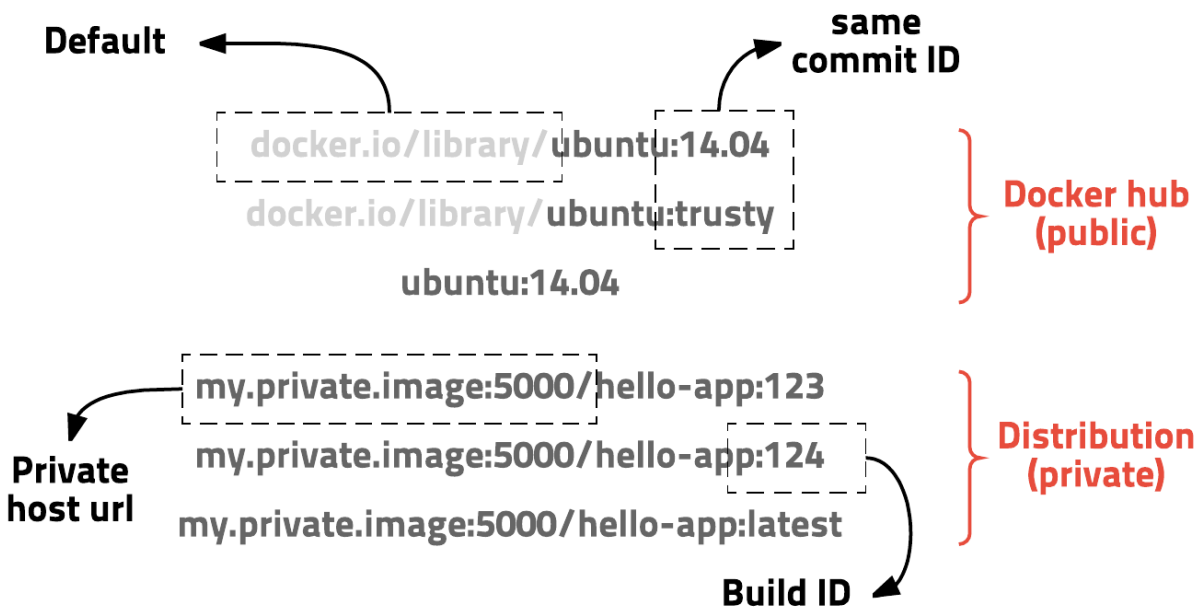
도커는 이런 문제를 해결하기 위해 **레이어layer**라는 개념을 사용하고 유니온 파일 시스템을 이용하여 여러개의 레이어를 하나의 파일시스템으로 사용할 수 있게 해줍니다. 이미지는 여러개의 읽기 전용read only 레이어로 구성되고 파일이 추가되거나 수정되면 새로운 레이어가 생성됩니다. ubuntu 이미지가 `A + B + C` 의 집합이라면, ubuntu 이미지를 베이스로 만든 nginx 이미지는 `A + B + C + nginx` 가 됩니다. webapp 이미지를 nginx 이미지 기반으로 만들었다면 예상대로 `A + B + C + nginx + source` 레이어로 구성됩니다. webapp 소스를 수정하면 `A`, `B`, `C`, `nginx` 레이어를 제외한 새로운 `source(v2)` 레이어만 다운받으면 되기 때문에 굉장히 효율적으로 이미지를 관리할 수 있습니다. ~~(개이득)~~

컨테이너를 생성할 때도 레이어 방식을 사용하는데 기존의 이미지 레이어 위에 읽기/쓰기read-write 레이어를 추가합니다. 이미지 레이어를 그대로 사용하면서 컨테이너가 실행중에 생성하

- 파일이나 변경된 내용은 읽기/쓰기 레이어에 저장되므로 여러개의 컨테이너를 생성해도 최소한의 용량만 사용합니다.

가상화의 특성상 이미지 용량이 크고 여러대의 서버에 배포하는걸 감안하면 단순하지만 엄청나게 영리한 설계입니다.

이미지 경로



Docker image url

이미지는 url 방식으로 관리하며 태그를 붙일 수 있습니다. ubuntu 14.04 이미지는 `docker.io/library/ubuntu:14.04` 또는 `docker.io/library/ubuntu:trusty` 이고 `docker.io/library` 는 생략가능하여 `ubuntu:14.04` 로 사용할 수 있습니다. 이러한 방식은 이해하기 쉽고 편리하게 사용할 수 있으며 태그 기능을 잘 이용하면 테스트나 롤백도 쉽게 할 수 있습니다.

Dockerfile

```
1  # vertx/vertx3 debian version
2  FROM subicura/vertx3:3.3.1
3  MAINTAINER chungsub.kim@purpleworks.co.kr
4
5  ADD build/distributions/app-3.3.1.tar /
6  ADD config.template.json /app-3.3.1/bin/config.json
7  ADD docker/script/start.sh /usr/local/bin/
8  RUN ln -s /usr/local/bin/start.sh /start.sh
9
10 EXPOSE 8080
11 EXPOSE 7000
12
```

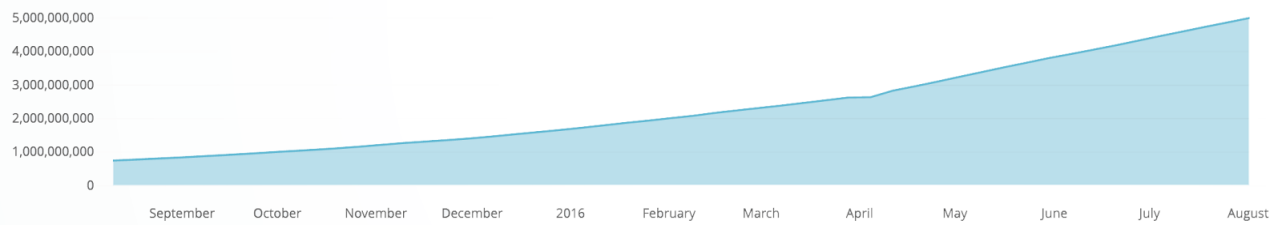
```
13  CMD ["start.sh"]
```

ockerfile hosted with ♡ by GitHub view raw

도커는 이미지를 만들기 위해 `Dockerfile` 이라는 파일에 자체 DSLDomain-specific language 언어를 이용하여 이미지 생성 과정을 적습니다. 추후에 문법에 대해 자세히 다루겠지만 위 샘플을 보면 그렇게 복잡하지 않다는 걸 알 수 있습니다.

이것은 굉장히 간단하지만 유용한 아이디어인데, 서버에 어떤 프로그램을 설치하려고 이것저것 의존성 패키지를 설치하고 설정파일을 만들었던 경험이 있다면 더 이상 그 과정을 블로깅하거나 메모장에 적지 말고 `Dockerfile` 로 관리하면 됩니다. 이 파일은 소스와 함께 버전 관리 되고 원한다면 누구나 이미지 생성과정을 보고 수정할 수 있습니다.

Docker Hub



Docker Hub Hits 5 Billion Pulls(2016/08)

도커 이미지의 용량은 보통 수백메가로 수기가가 넘는 경우도 흔합니다. 이렇게 큰 용량의 이미지를 서버에 저장하고 관리하는 것은 쉽지 않은데 도커는 Docker hub를 통해 공개 이미지를 무료로 관리해 줍니다. 하루에도 엄청난 용량의 이미지가 전세계에서 다운로드 되고 트래픽 비용만 해도 어마어마 할 것 같은데 그것이 다 무료!입니다.

Command와 API

도커 클라이언트의 커맨드 명령어는 정말 자아아알 만들어져 있습니다. 대부분의 명령어는 직관적이고 사용하기 쉬우며 컨테이너의 복잡한 시스템 구성을 이해하지 못하더라도 편하게 사용할 수 있습니다. 또한 http기반의 Rest API도 지원하여 확장성이 굉장히 좋고 훌륭한 3rd party 툴이 나오기 좋은 환경입니다.

유용한 새로운 기능들

도커는 발전속도가 아주 빠른 오픈소스입니다. 사용하면서 부족하다고 느꼈던 부분은 빠르게 개선되고 새로운 버전이 나오면 유용한 기능이 대폭 추가됩니다. 어떻게 보면 프로그램을 작은 조각으로 나누고 여러개의 프로그램을 조합하여 동작시키는 유닉스의 철학에는 맞지 않지만.. 너무 잘합니다. ㅠㅠ

이번 1.13버전에서는 Docker stacks이라는 여러개의 컨테이너를 한번에 관리하는 기능이 정식으로 릴리즈 되었고 system 커맨드가 추가되어 이미지, 컨테이너 관리가 더 편해졌습니다.

crets Management라는 비밀정보를 관리하는 기능도 추가됩니다.

새로운 기능이 계속 추가되고 있고 다음 릴리즈가 기대됩니다.

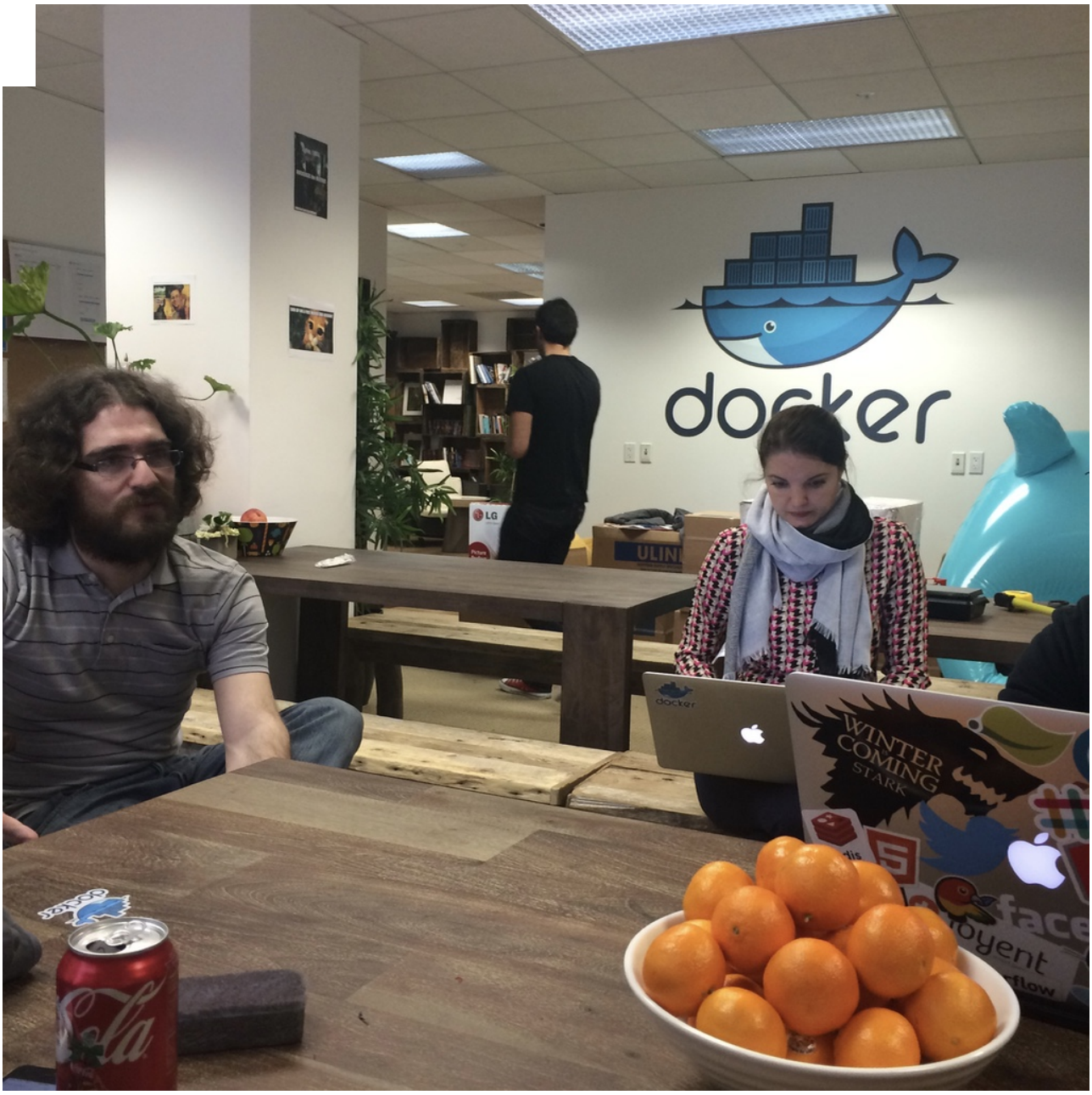
훌륭한 생태계

도커는 굉장히 큰 생태계를 가지고 있고 커다란 기업과 협력하여 사실상 클라우드 컨테이너 세계의 de facto가 되었습니다. 로깅, 모니터링, 스토리지, 네트워크, 컨테이너 관리, 배포등 다양한 분야에서 다양한 툴들이 존재하며 아예 도커를 위한 OS(coreos-> container linux)도 존재합니다.

현재 도커를 기반으로한 오픈소스 프로젝트는 10만개가 넘고 굉장히 활발하게 진행되고 있습니다.

커뮤니티 지원

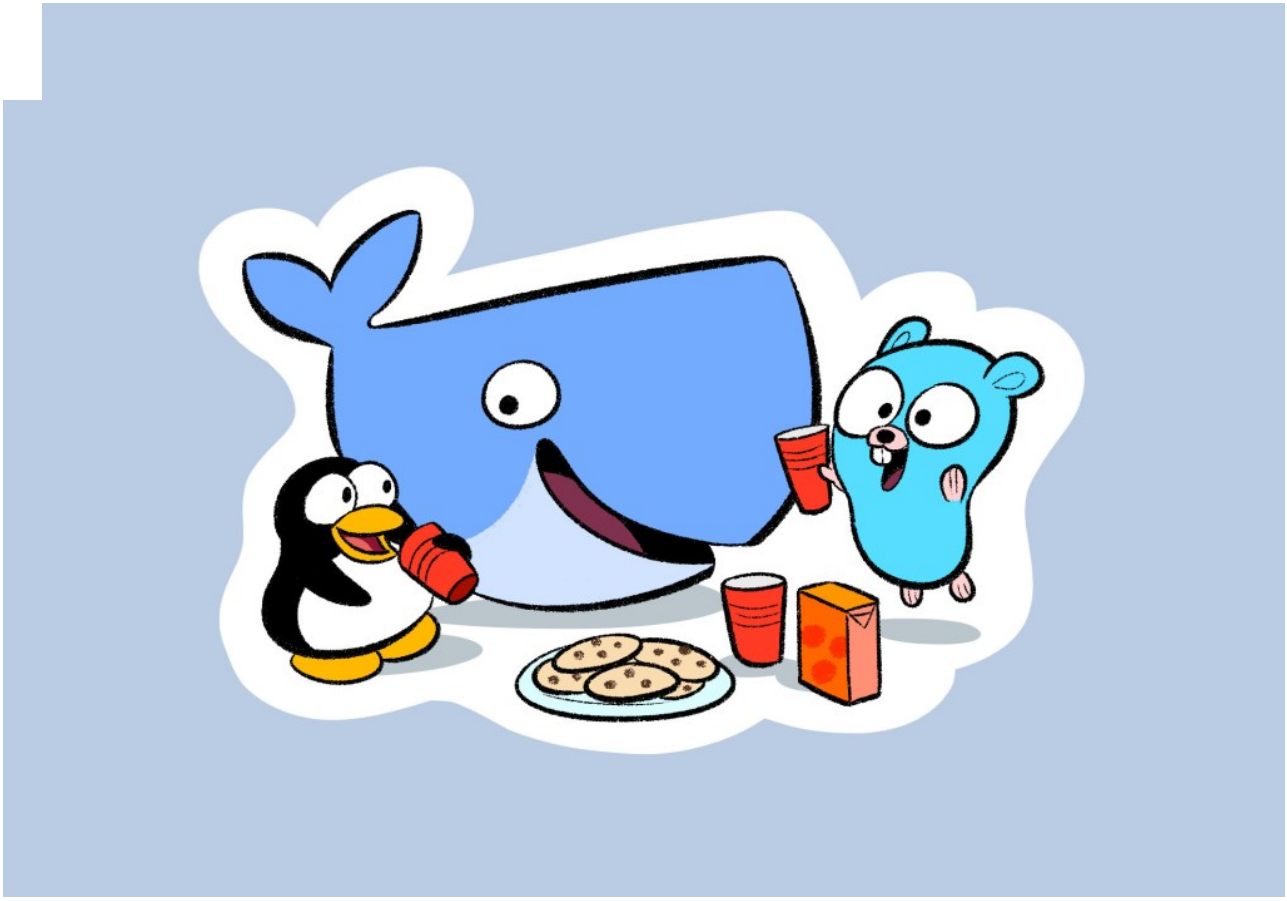
도커는 기술기업답지 않게 홍보와 커뮤니티 관리에 굉장히 신경쓰고 있습니다. 커뮤니티를 위한 스티커나 티셔츠를 무료로 제공하고 필요하면 연사요청도 할 수 있습니다. 홈페이지에서는 전세계에서 열리는 밋업 상황을 볼 수 있고 일주일마다 발송되는 뉴스레터에는 다양한 개발자들의 글이 실려있습니다.



Docker office에서 만난 Jérôme Petazzoni

운이 좋았지만 도커 오피스에 방문해서 사무실을 구경할 수도 있었고 고오오오급 개발자 Jérôme Petazzoni와 직접 이야기를 나눌수 있었습니다! ~~아 또 가고 싶다~~

moby dock

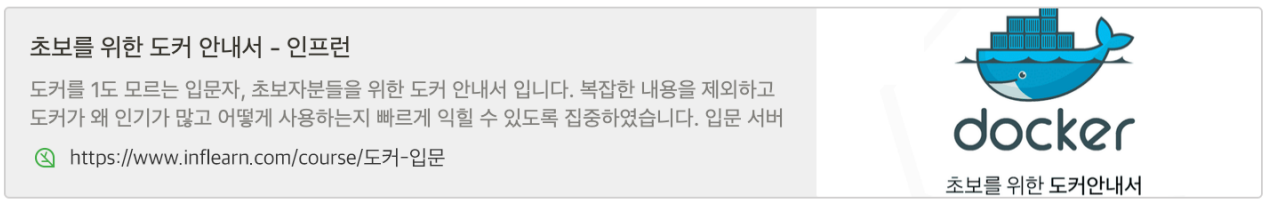


Tux(linux) - Moby Dock(docker) - Gopher (golang)

도커는 넘나 귀여운 고래를 로고로 하고 있습니다. 로고 스티커는 항상 인기가 넘치고 로고가 그려진 티셔츠는 입고 돌아다녀도 개발자처럼 보이지 않습니다. 도커가 성공한 가장 큰 이유는 귀여운 고래 덕분이라고 생각합니다.(엄근진) ~~오픈소스가 성공하려면 귀여운 동물 캐릭터를 사용하세요!~~

정리

여기까지 도커에 대해 기본적인 내용을 아주 얕게 살펴보았습니다. 이제 실전으로 들어가봅시다!



이제 도커안내서를 영상으로 만나보세요!

초보를 위한 도커 안내서 - 설치하고 컨테이너 실행하기

초보를 위한 도커 안내서 2번째 글입니다. 도커의 기본적인 내용을 이야기 했던 첫번째 글에 이어 실제로...

어느덧 2016년 마지막 날입니다. 올 한해 역시 정신없이 지나갔고 작년에 태어난 딸도 돌을 지나고 어느새...

초보를 위한 도커 안내서 - 도커란 무엇인가? ✓ SERIES 1/3

초보를 위한 도커 안내서 - 설치하고 컨테이너 실행하기 SERIES 2/3

- 초보를 위한 도커 안내서 - 이미지 만들고 배포하기 SERIES 3/3

Docker DevOps Server

WRITTEN BY

subicura

Published 19 Jan 2017

SUPPORTED BY

Proudly published with Jekyll

Feedly 구독하기
RSS 구독하기
Email 구독하기

댓글 65개 정렬 기준 인기순

댓글 달기...

윤재형
한방에 이해할 수 있게 해주셔서 감사합니다.
좋아요 · 답글 달기 · 6주

이지형
최고의 설명
좋아요 · 답글 달기 · 10주

정훈
좋은 글 고맙습니다. 책보다 낫네요
좋아요 · 답글 달기 · 20주

용은재

좋은 글 이었습니다 도커에대해 쉽게 설명되어있었습니다. 감사합니다.

좋아요 · 답글 달기 · 22주

송형선

잘읽었습니다.

좋아요 · 답글 달기 · 40주

최단비

잘 읽고갑니다~

좋아요 · 답글 달기 · 1년

다은송

lmctfy(Let Me Contain That For You) 이거는 진짜 뭐라고 읽나요?

좋아요 · 답글 달기 · 1년

정태혁

[엘엠시티피] 래요..
https://github.com/google/lmctfy/blob/master/README.md

좋아요 · 답글 달기 · 1년

주은지

쉽게 설명해주셔서 도움 많이 됐습니다.

좋아요 · 답글 달기 · 1년

이승규

잘읽었습니다!

좋아요 · 답글 달기 · 1년

김성권

글이 무척 쉽게 쓰여있어서 덕분에 도커에 대해 알게되었습니다~ ^^ 감사합니다.

좋아요 · 답글 달기 · 1년

댓글 10개 더 읽어들이기

Facebook 댓글 플러그인

READ THIS NEXT

YOU MIGHT ENJOY

2016년 블로그 회고