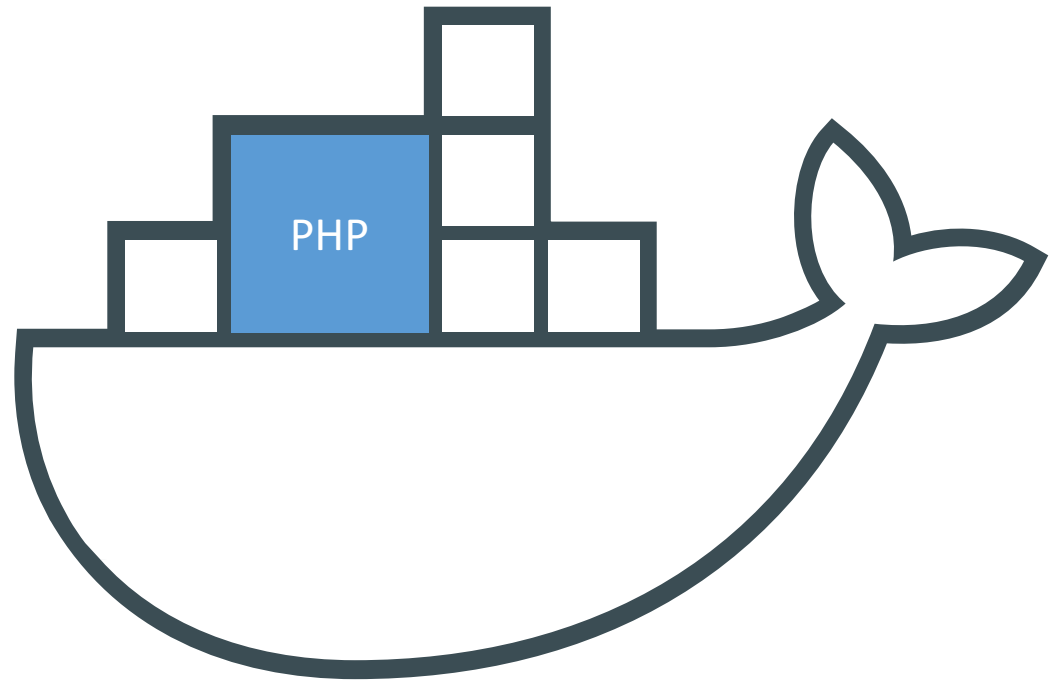


# Modern PHP on Docker



# Table of Contents

이 튜토리얼의 목적

---

도커에 대해서

---

컨테이너 VS 가상머신

---

얼마전까지의 개발 환경

---

현재의 개발 환경

---

과거의 개발환경 VS 현재의 개발환경

---

이미지와 컨테이너

---

도커 명령어 익숙해지기

---

Dockerfile 익숙해지기

---

SSH 키 연결하기

---

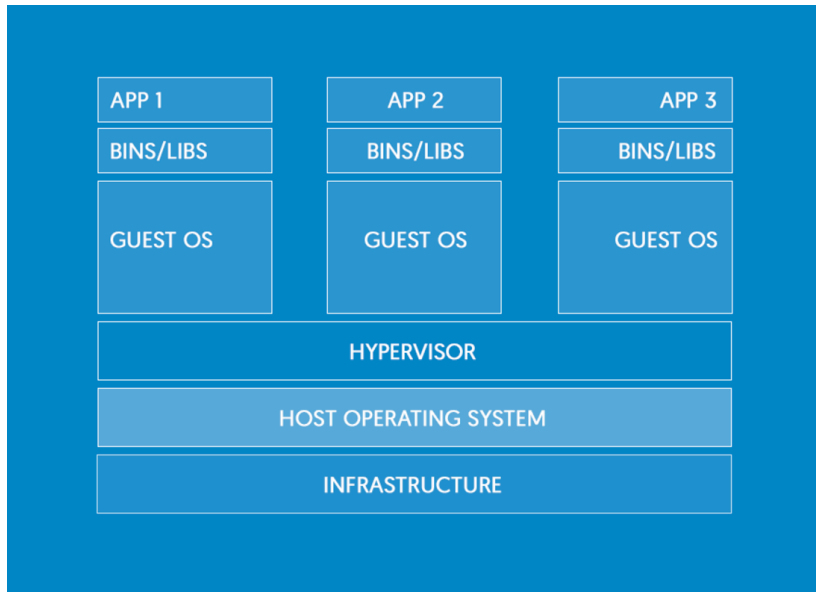
## 튜토리얼 목적

- 실제로 개발환경 구축에 필요한 도커 파악
  - 도커로 무얼 할 수 있는지를 알아봅니다.
  - 도커를 이용해서 개발 환경을 구축 할때 어떤 이점이 있는지 알아봅니다.
- Docker 이미지와 컨테이너 이해
  - 이미지와 컨테이너의 관계를 알아봅니다.
  - 이미지를 만드는 방법에 대해서 알아봅니다.
  - 이미지를 이용해서 컨테이너를 만드는 방법을 알아 봅니다.
- 간단한 웹 개발 환경 구축
  - 라라벨을 이용하여 간단한 웹 애플리케이션을 만들어 봅니다.

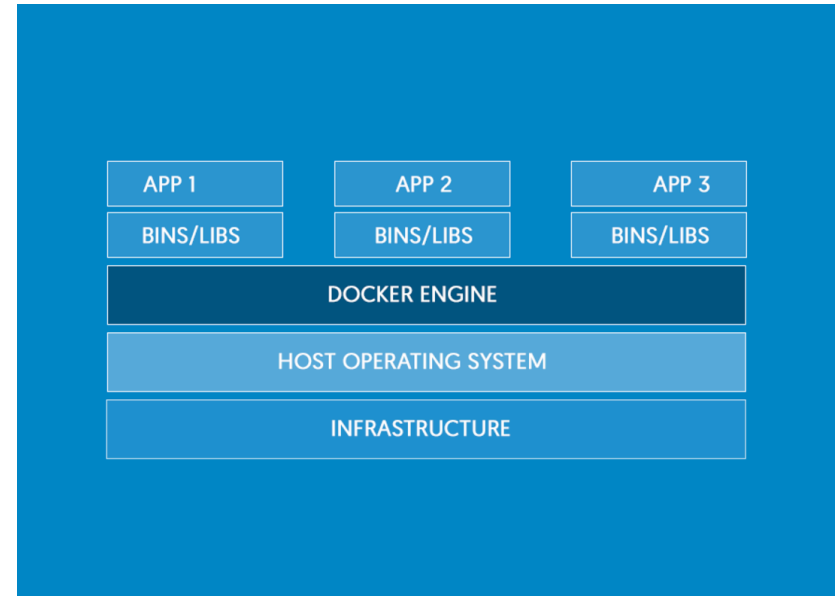
# 도커에 대해서

- 도커는 컨테이너를 손쉽게 관리할수 있게 도와주는 도구
  - 밑에 저변에 깔려있는 기술은 복잡하나,
  - 실제로 몇개의 명령어(CLI)와 이미지와 컨테이너 개념만 익혀도 당장 개발 환경으로 사용하기에 문제가 없습니다.
  - 기본적인 리눅스 명령어만 익혀도 괜찮아요.
  - 먼저 툴에 익숙해지고 관심이 생기면 그때 차근차근 알아가도 괜찮아요.

# 가상머신 VS 컨테이너



가상머신



컨테이너

## 컨테이너는 가볍다.

- 컨테이너는 격리된 환경(Linux Container , LXC)을 제공한다.
  - 공유할 수 있는 리소스는 공유하고 필요한 부분만 생성하기 때문에 좋다.

하지만,

이건 리눅스일때 해당함

## 몇달전에 저의 개발환경



패러럴즈

+



vagrant

+



홈스테드

```
$ vagrant up
```

```
$ vagrant ssh
```

```
$ cd my_project_directory
```

패러럴즈를 지원하는 vagrant에 등록된  
homestead box는 php 5.5 입니다.

하지만,  
Laravel 5.3을 위해서는 php 5.6 이상이 필요한  
상황

기존의 vm을 그대로 두고 새로운 머신이 필요한  
상황에 직면하게 됨.

## 버추얼 머신에 올리고 나서 사용해보니

개발환경으로 버추얼머신 사용해보니 괜찮았다.

- Vagrant를 알게되면서 개발환경으로 버추얼 머신을 적극 활용 했다.
- 팀원들과 동일한 개발환경에서 개발해서 괜찮았다.

하지만,

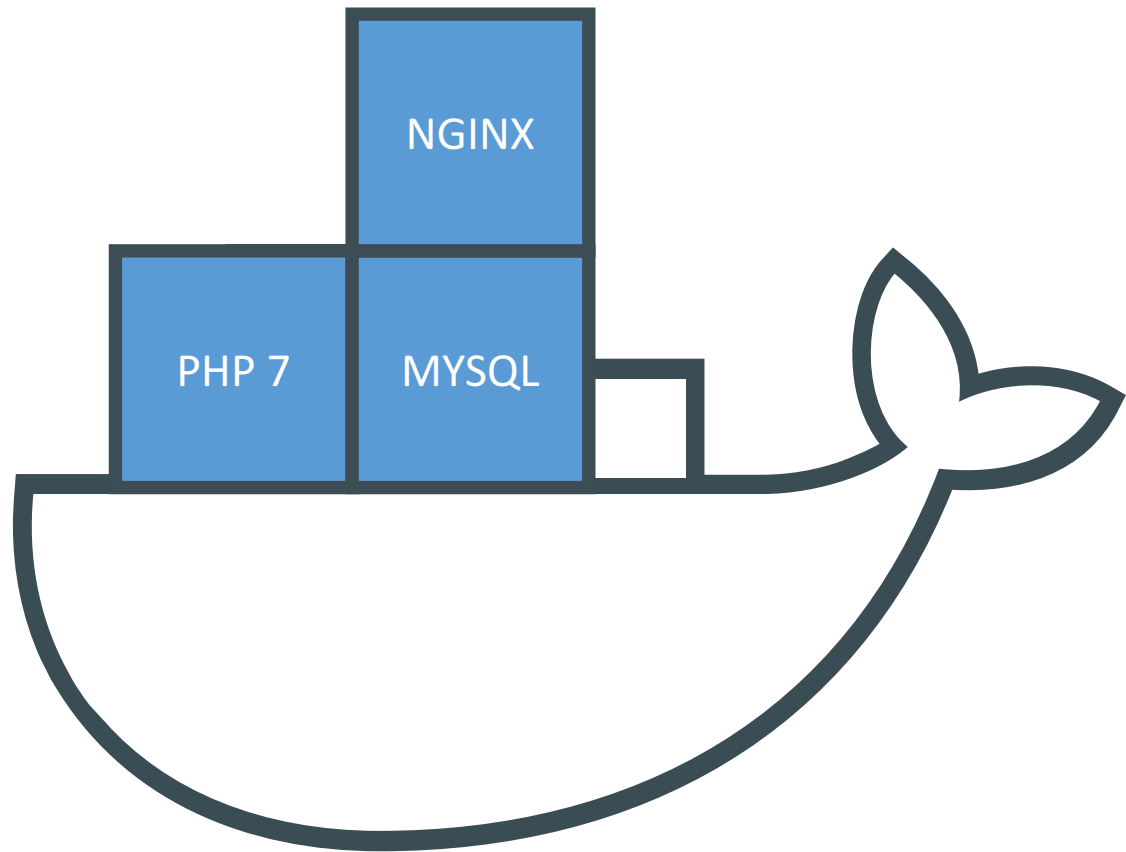
시간이 지날 수록 버추얼머신이 처음과는 다른 상태가 된다.

버추얼머신에 npm버전 업데이트하고 ,

node 버전도 업데이트하고 ...

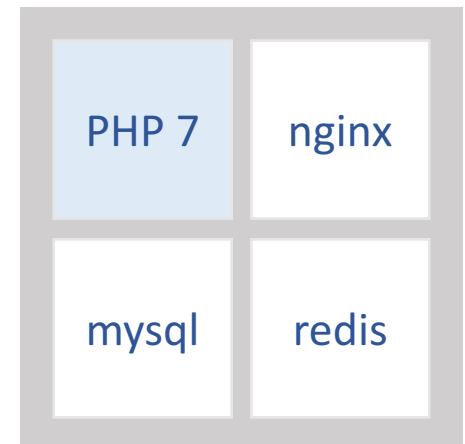
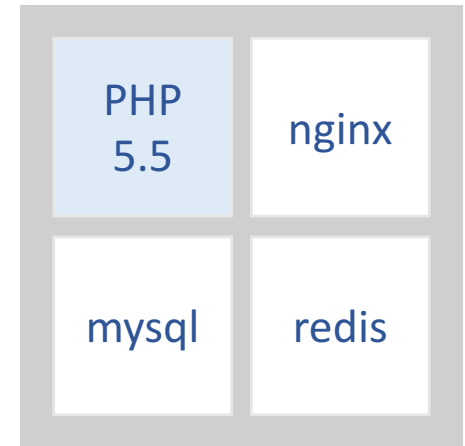


## 현재 제가 사용하고 있는 개발 환경



## 과거의 개발환경 VS 현재의 개발환경

- 프로젝트 별로 독립된 개발 환경을 구축
  - 애플리케이션마다 필요한 컨테이너를 올리면 된다.
  - 로컬 머신이 오염되지 않아서 좋다.



# 이미지와 컨테이너

- 이미지는 컨테이너를 만들기 위한 일종의 템플릿
  - 이미지를 이용해서 컨테이너를 생성합니다.
  - 도커 명령어를 이용해서 이미지를 로컬머신에 다운로드 받습니다.
    - `$ docker pull <이미지이름>`
    - 이미지 이름은 <https://hub.docker.com/>에 찾아보자.
- 컨테이너는 이미지를 이용해서 만들어진 인스턴스 이자 독립된 공간이다
  - 컨테이너를 생성하기 위해서는 이미지가 필요하다.
    - `$ docker images`로 명령어로 호스트 머신에 설치된 이미지를 확인 할 수 있다.
  - 도커 명령어를 이용해서 이미지를 컨테이너로 생성하자.
    - `$ docker run <이미지이름>`

# 도커 명령어 익숙해지기

명령어	설명
\$ docker pull	이미지를 로컬 머신에서 가져온다.
\$ docker images	모든 이미지를 보여준다.
\$ docker ps	현재 실행중인 컨테이너를 보여준다.
\$ docker rm	컨테이너를 삭제한다
\$ docker run	컨테이너를 생성한다.
\$ docker stop	컨테이너를 정지시킨다.

## NGINX 이미지를 받아보자



```
$ docker pull nginx:태그
```

# NGINX 서버를 실행하기



```
$ docker run --name nginx-alpine -v $(pwd):/usr/share/nginx/html:ro -p 8888:80 -d nginx:alpine
```

# PHP 7와 composer가 설치되는 이미지 만들기

## Dockerfile 생성

```
FROM php:7.0-cli
MAINTAINER Jaehee
RUN apt-get update
RUN apt-get install -y zip unzip curl git
RUN php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
RUN php composer-setup.php --install-dir=/usr/bin --filename=composer
RUN php -r "unlink('composer-setup.php');"
```

## PHP 7와 composer가 설치되는 이미지 만들기



```
$ docker build -t composer:latest .
```



# Composer 실행

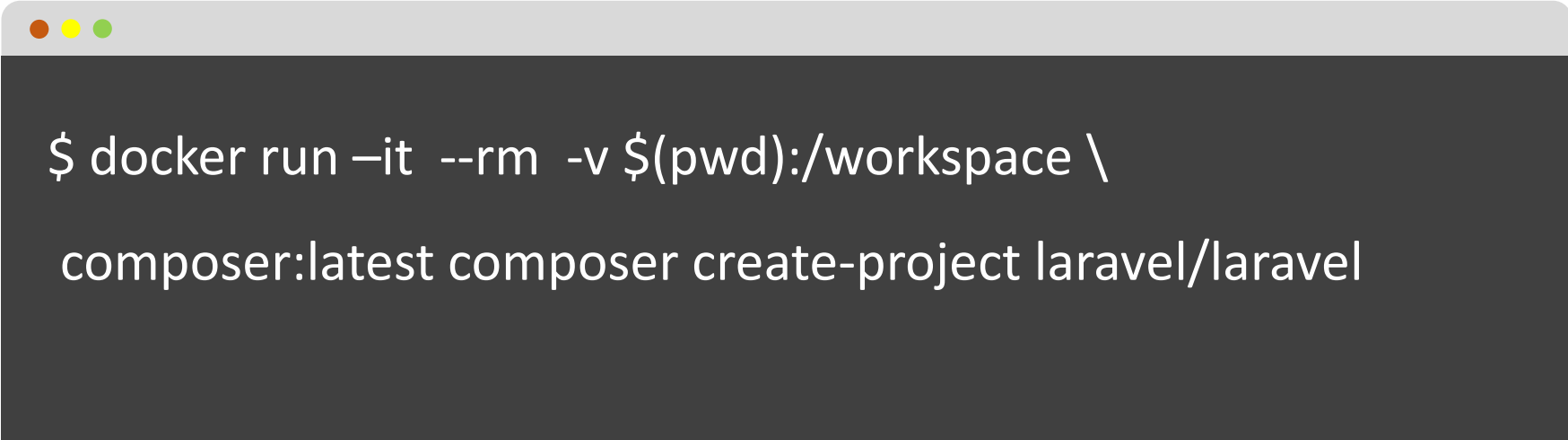


```
$ docker run -it composer:latest composer --version
```

Do not run Composer as root/super user! See <https://getcomposer.org/root> for details

Composer version 1.2.2 2016-11-03 17:43:15

# Laravel 인스톨



```
$ docker run -it --rm -v $(pwd):/workspace \
composer:latest composer create-project laravel/laravel
```

## Php 로 로컬 서버 실행하기

```
$ docker run --rm -it \  
  -p 80:8888 -v $(pwd):/workspace composer:latest \  
  php -S 0.0.0.0:8888 -t /workspace/public
```

# Laravel

[DOCUMENTATION](#)

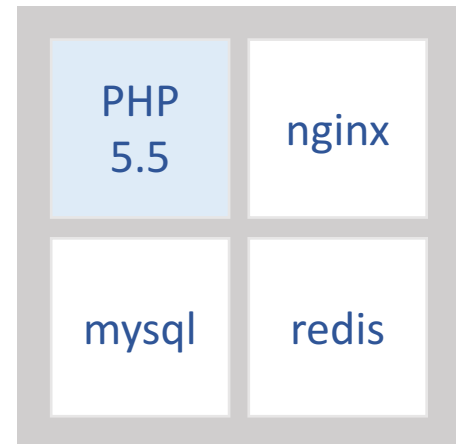
[LARACASTS](#)

[NEWS](#)

[FORGE](#)

[GITHUB](#)

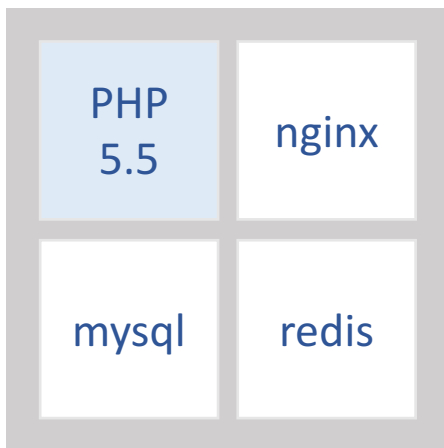
실제로는 하나의 애플리케이션은  
여러개의 컨테이너가 필요하다.



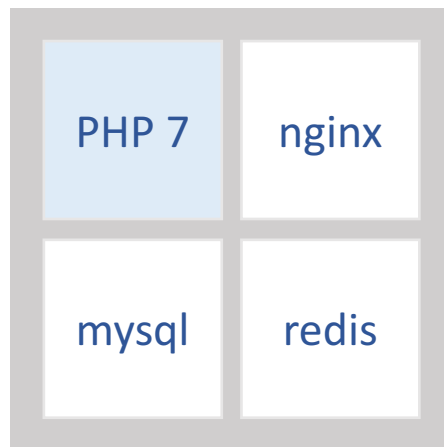
# Docker-compose

- 여러개의 이미지를 레고블록처럼 쉽게 조립해서 사용 할 수 있게 도와준다.

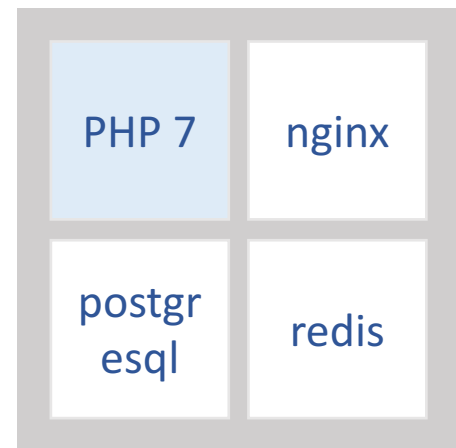
App#1



App#2



App#3




# Laradock

- 라라독은 도커위에서 라라벨을 쉽게 이용 할 수 있도록 도와주는 프로젝트
  - 홈스테드에서 제공하는 역할과 비슷함.
  - Mysql, nginx,apache,caddy,redis등 다양한 이미지를 기본으로 자유로운 조합이 가능하다.

```
$ git clone https://github.com/laradock/laradock
```

## 간단한 투두



```
$ php artisan make:auth
```

```
$ php artisan migrate
```

```
$ php artisan make:migration create_todo_table --create="todos"
```

```
$ php artisan make:model Todo
```

```
$ php artisan make:notification TodoNotification
```

```
$ php artisan make:controller TodoController
```

```
$ php artisan make:request TodoRequest
```