Docker for Development: Getting Started

Lisa H. Ridley

Savas Labs

DrupalCamp Atlanta

October 22, 2016

Who am I?

Lisa Ridley, Director of Client Success, Savas Labs

- Lead Developer and Project Manager
- Automated Testing Champion (Unit, Functional, User Acceptance, Visual Regression)
- drupal.org: lhridley
- twitter: lhridley
- GitHub: github.com/lhridley & github.com/codementality
- LInkedIn: https://www.linkedin.com/in/lisahridley
- Blog: http://savaslabs.com/blog/

What is your development environment?

Development Environment	*# users*
MAMP / WAMP / Locally installed LAMP/LEMP Stack	*13*
Vagrant / VMWare / Parallels / VirtualBox — Virtual Machines	*8*
Remote server	*8*
Docker / Containers	*3*

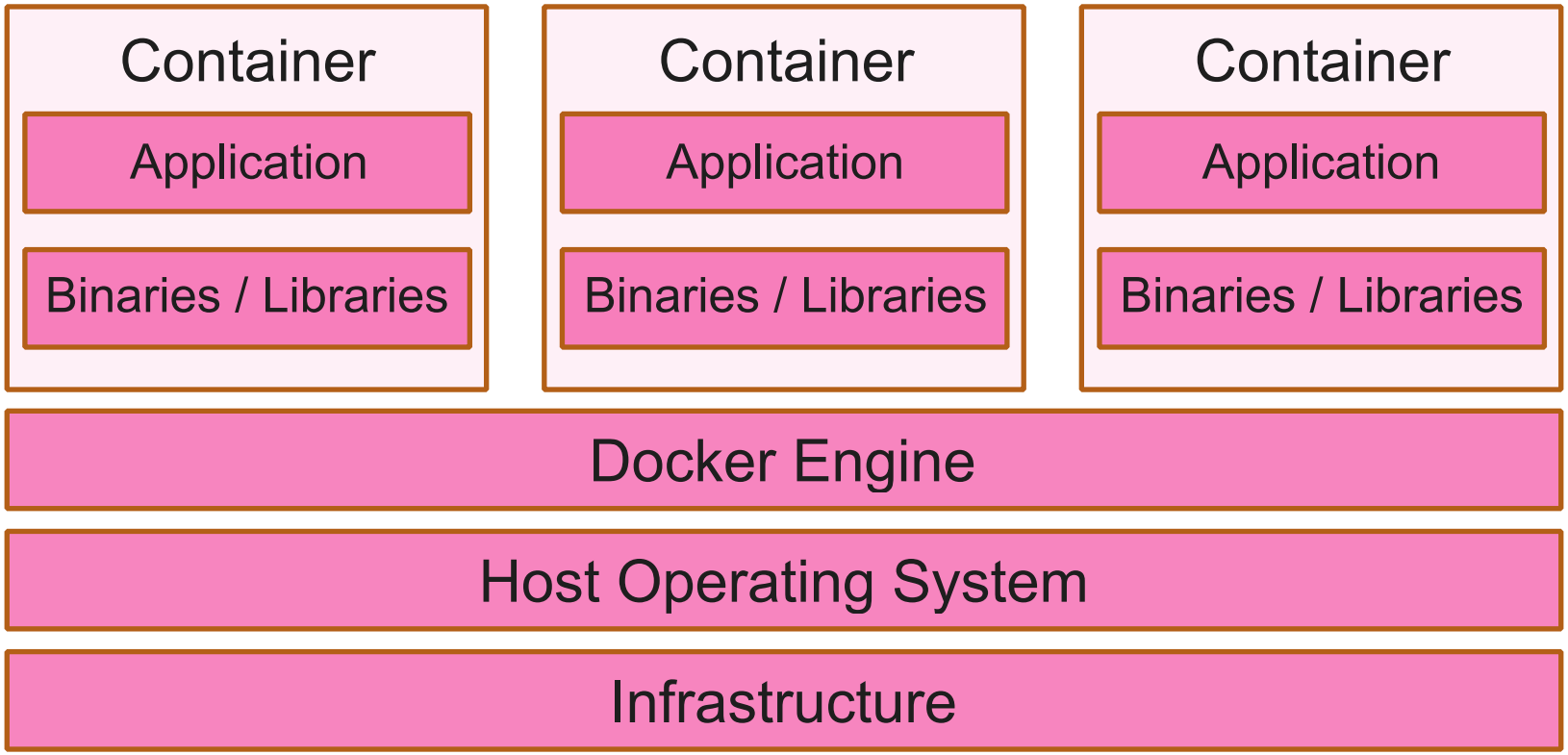
What is Docker?

- Automates the deployment of applications in **software containers**
- Operating system abstraction layer
- Open Source Project (May 2013)

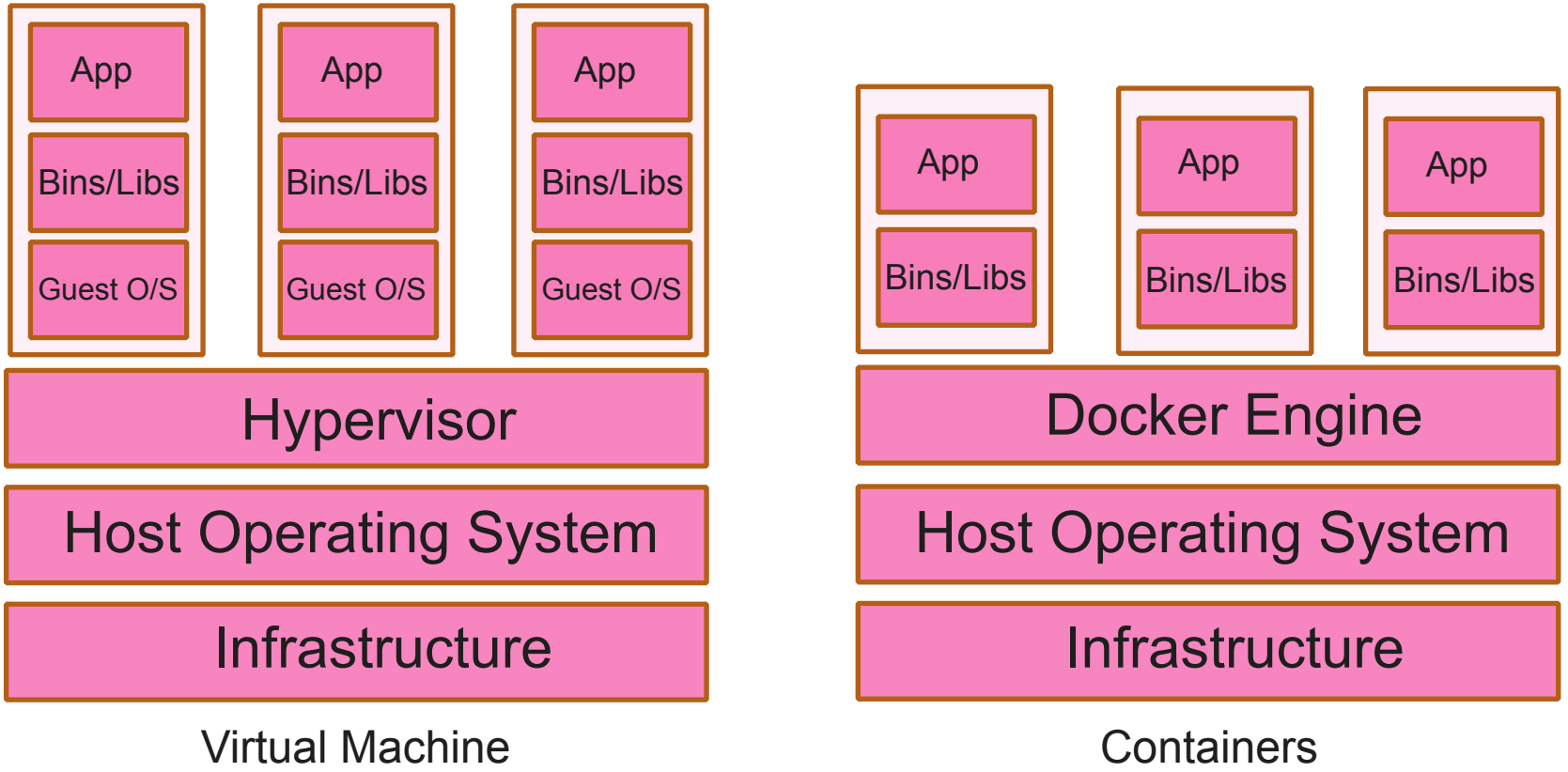
What is a Docker software container

- An isolated user-space instance — that contains a complete application with all dependencies and components needed to run properly
- Shares the underlying O/S kernel
- Looks and feels like a real server

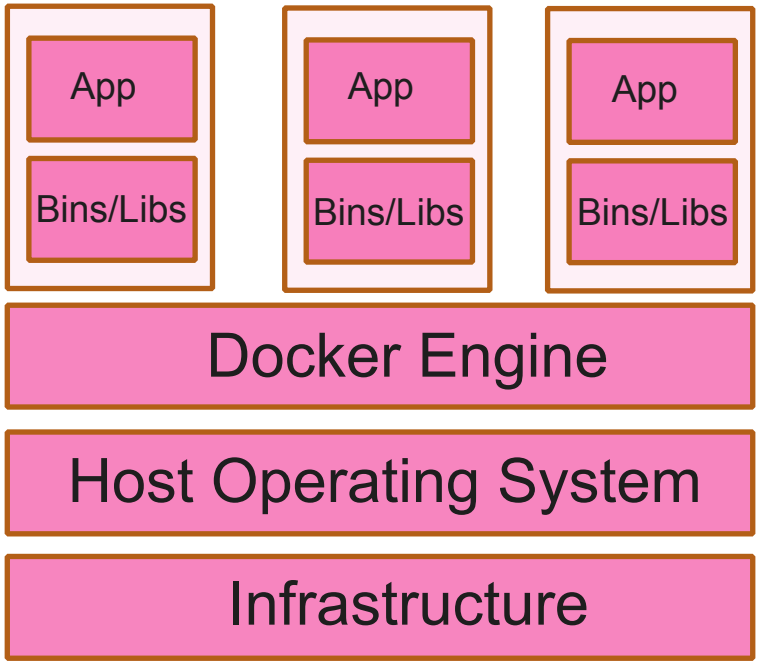
What does that look like?



Containers vs. Virtual Machines



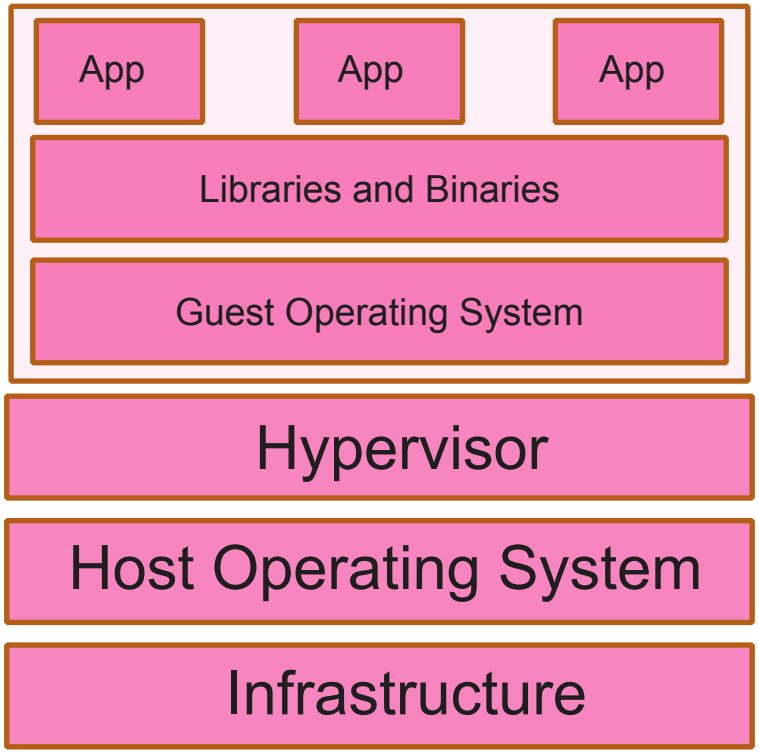
Containers vs. Virtual Machines



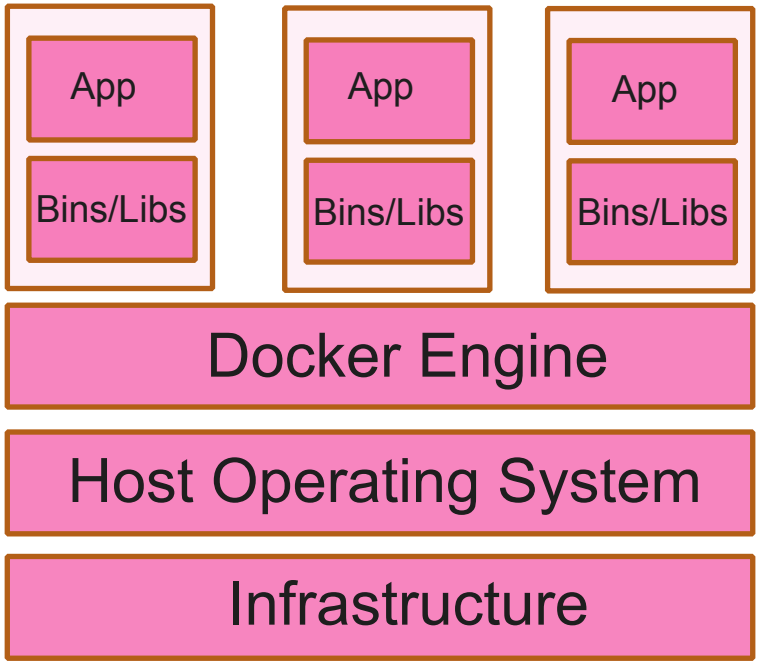
Virtual Machine

Containers

Containers vs. Virtual Machines



Virtual Machine



Containers

As a Developer, how does this benefit me?

Benefits of Containerization

- Development environment consists of components
- Components can be swapped as needed for different clients / applications
- If production environment is containerized, deployment is as simple as dropping a completed application in a fully tested container on the production server

Containers as components

Apache Container

Client App Container v1

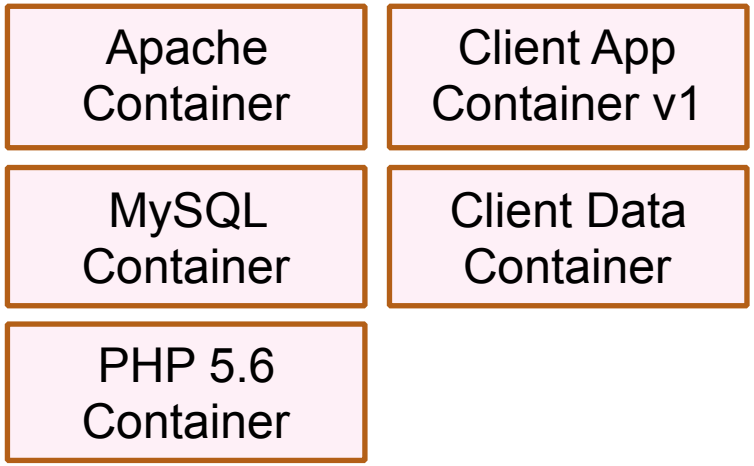
MySQL Container

Client Data Container

PHP 5.6 Container

Docker Engine

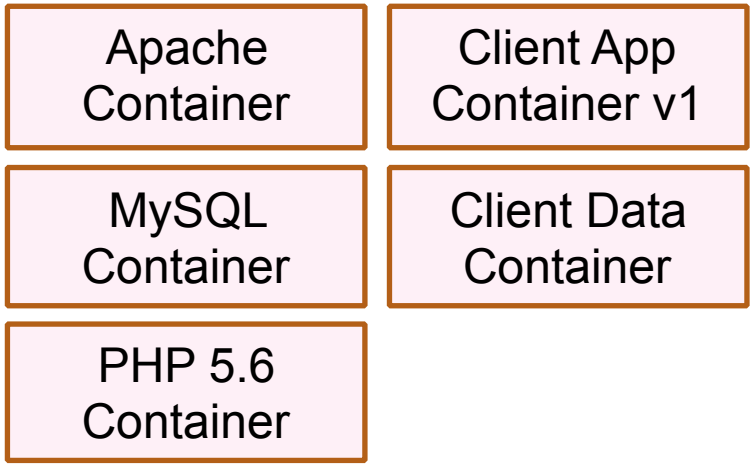
Containers as components



- Applications are isolated

Docker Engine

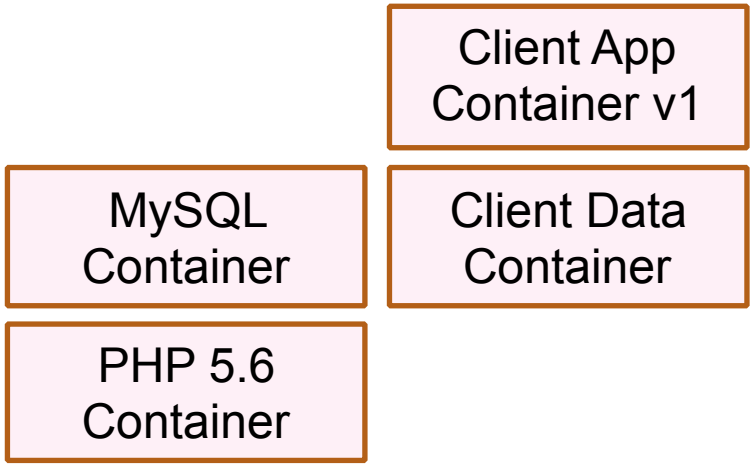
Containers as components



- Applications are isolated
- Reconfiguration is painless

Docker Engine

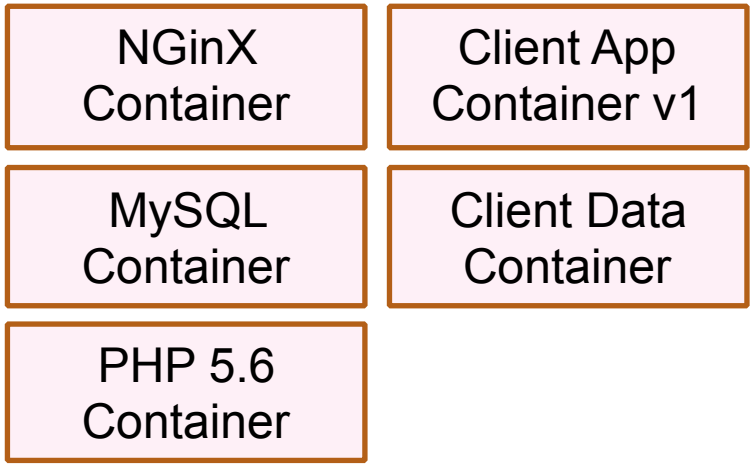
Containers as components



- Applications are isolated
- Reconfiguration is painless

Docker Engine

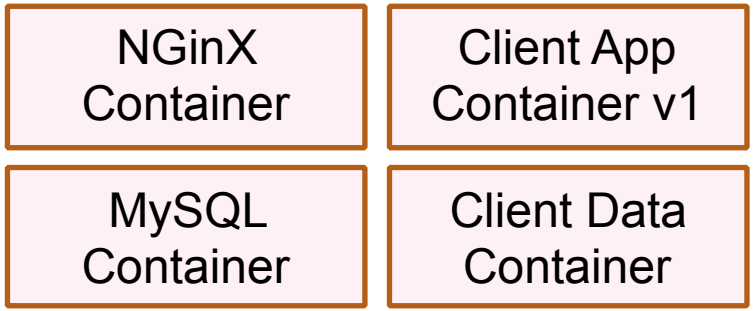
Containers as components



- Applications are isolated
- Reconfiguration is painless

Docker Engine

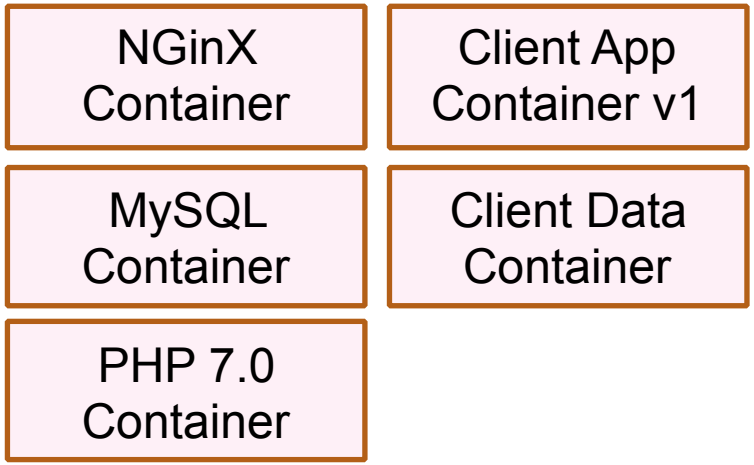
Containers as components



- Applications are isolated
- Reconfiguration is painless

Docker Engine

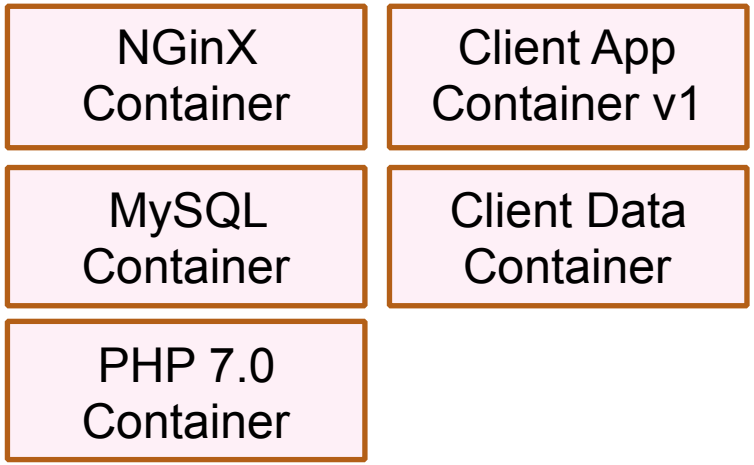
Containers as components



- Applications are isolated
- Reconfiguration is painless

Docker Engine

Containers as components

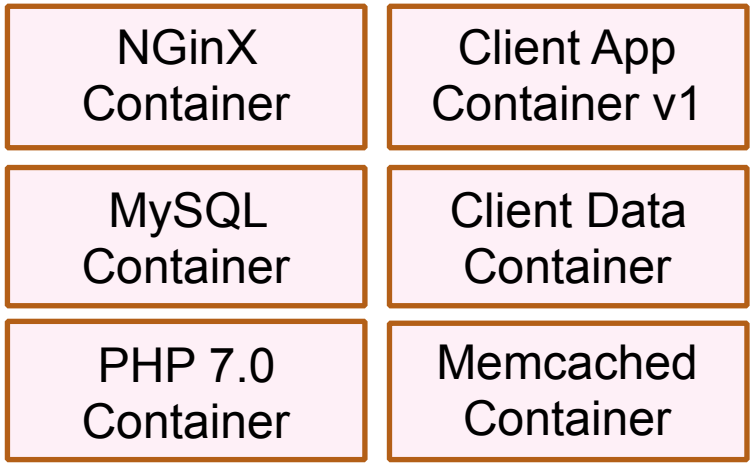


- Applications are isolated
- Reconfiguration is painless

…Really painless

Docker Engine

Containers as components

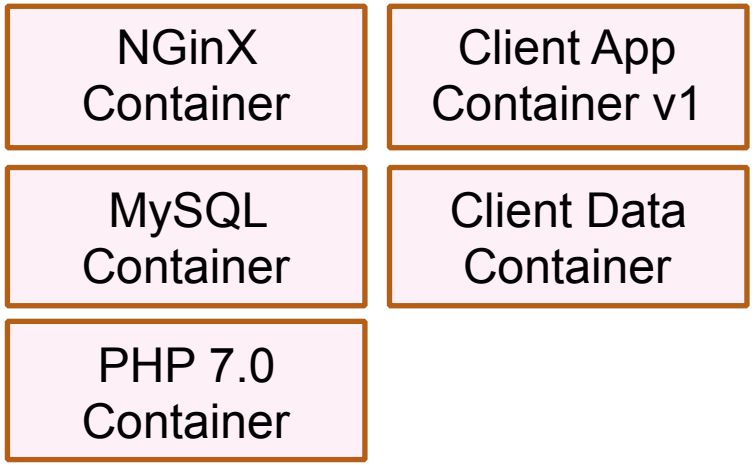


- Applications are isolated
- Reconfiguration is painless

  …Really painless

Docker Engine

Containers as components

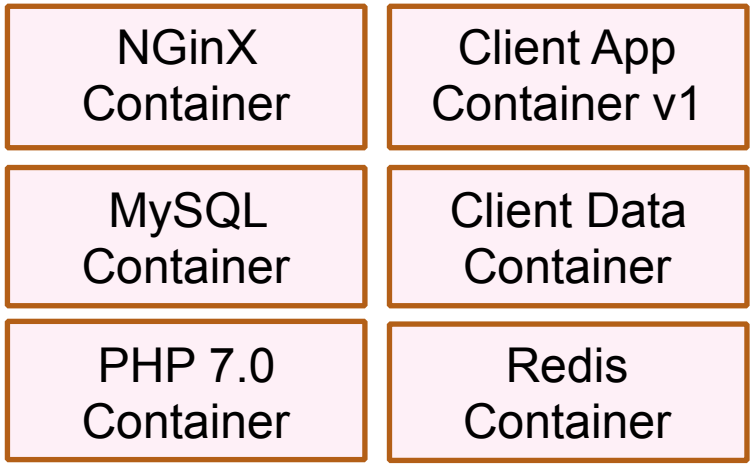


- Applications are isolated
- Reconfiguration is painless

…Really painless

Docker Engine

Containers as components

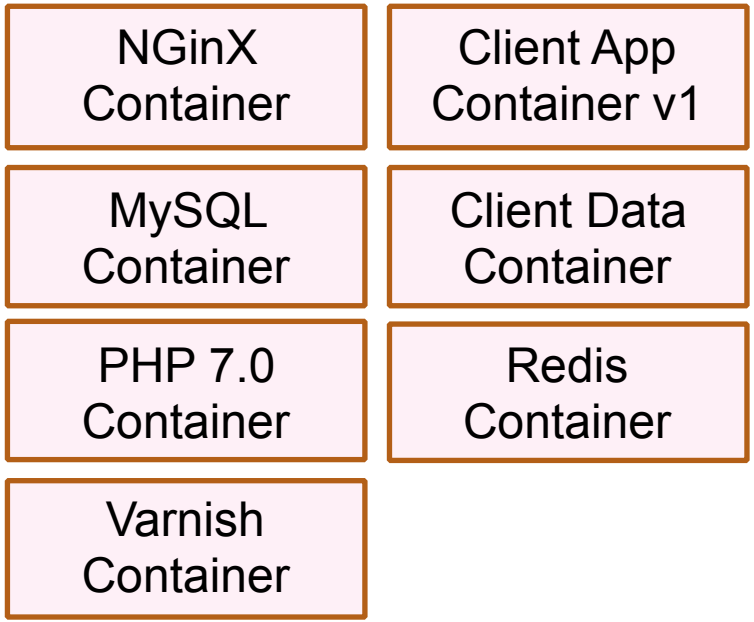


- Applications are isolated
- Reconfiguration is painless

…Really painless

Docker Engine

Containers as components

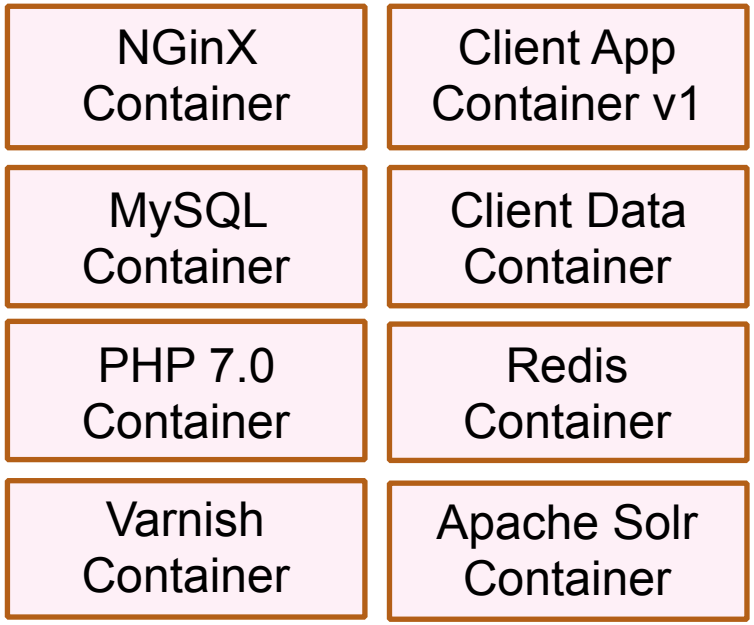


- Applications are isolated
- Reconfiguration is painless

…Really painless

Docker Engine

Containers as components

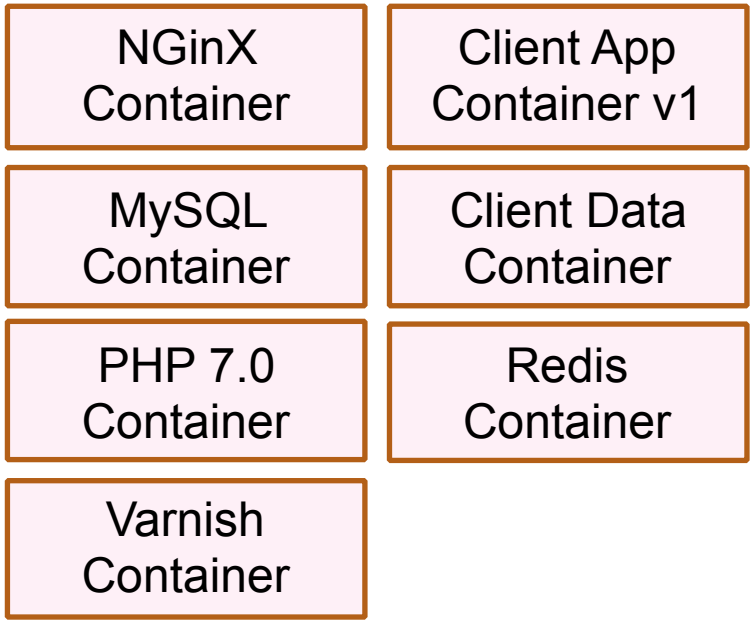


- Applications are isolated
- Reconfiguration is painless

…Really painless

Docker Engine

Containers as components

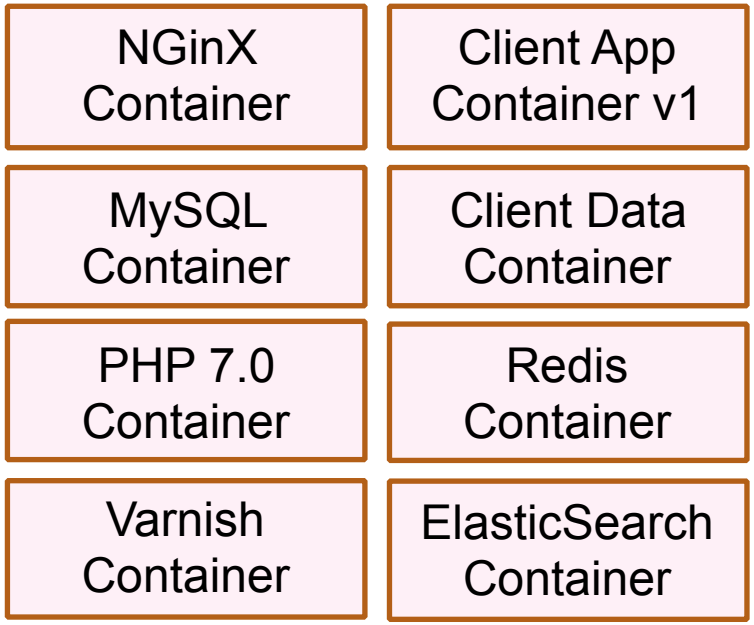


- Applications are isolated
- Reconfiguration is painless

…Really painless

Docker Engine

Containers as components

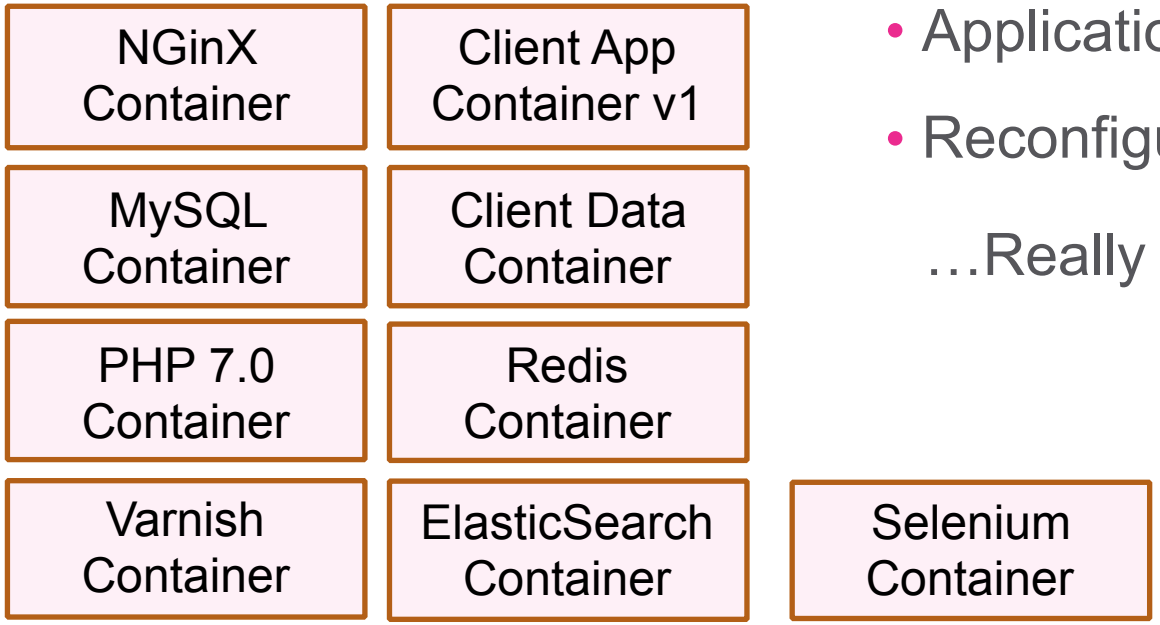


- Applications are isolated
- Reconfiguration is painless

…Really painless

Docker Engine

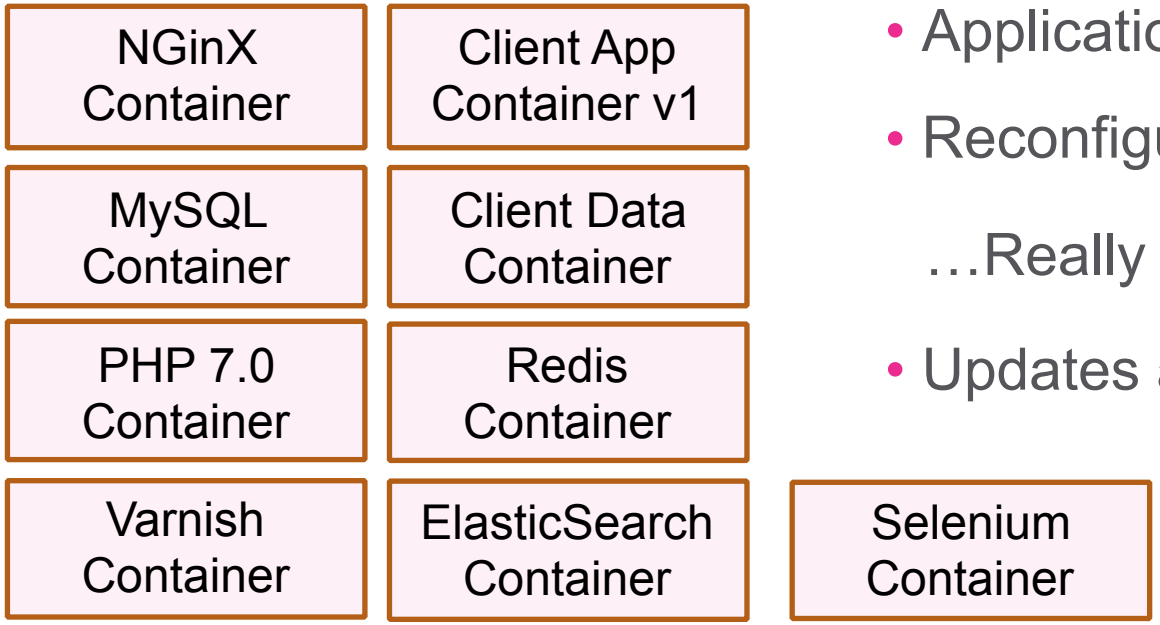
Containers as components



- Applications are isolated
- Reconfiguration is painless

…Really painless

Containers as components



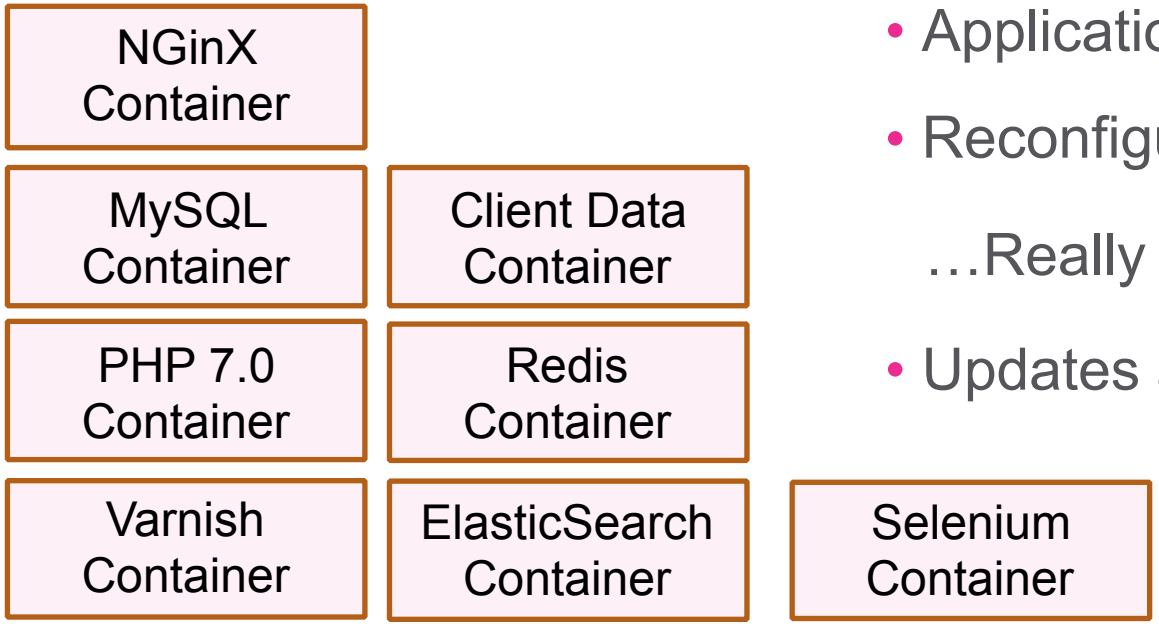
- Applications are isolated
- Reconfiguration is painless

  …Really painless

- Updates are simplified

Docker Engine

Containers as components

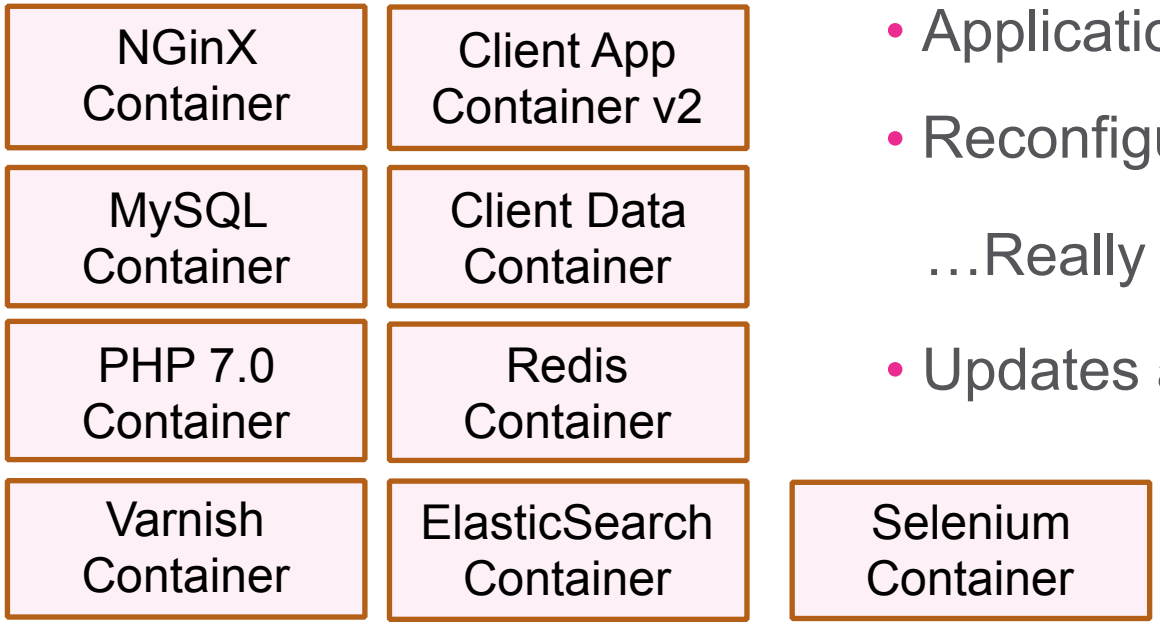


- Applications are isolated
- Reconfiguration is painless

  …Really painless
- Updates are simplified

Docker Engine

Containers as components



- Applications are isolated
- Reconfiguration is painless

  …Really painless

- Updates are simplified

Docker Engine

Docker Stack Configuration (docker-compose)

```
version: '2'
services:
  web:
    domainname: local.hptn.org
    hostname: local.hptn.org
    container_name: local.hptn.org
    ports:
      - '443:443'
      - '80:80'
    environment:
      - RUN_AS_UID=1000
      - GH_TOKEN
      - IMGUR_API_KEY
    image: savaslabs/hptn-web:1.6
    depends_on:
      - db
      - memcached
    volumes:
      - ./docker/hptn-web/drupal:/conf
      - ./docker/hptn-web/php:/etc/php5
  memcached:
    container_name: hptn_memcached
    image: memcached:1.4.21
    environment:
      - MEMCACHED_MEMORY_LIMIT=128
  db:
    container_name: hptn_db
    image: mysql:5.5.47
    volumes:
      - ./docker/mysql:/etc/mysql/conf.d/
      - mysql-data:/var/lib/mysql
    environment:
      MYSQL_DATABASE: 'hptn_docker'
      MYSQL_ROOT_PASSWORD: 'root'
    ports:
      - '33308:3306'
  selenium:
    container_name: hptn_selenium
    image: selenium/standalone-firefox:2.53.0
    volumes:
      - /dev/shm:/dev/shm
volumes:
  mysql-data:
    driver: local
```

How can I get started using Docker for development

- Install the Docker Engine (Linux), Docker for Windows, or Docker for Mac
- Install Docker Compose (Linux)
- Create a docker-compose.yml file with your stack configuration
- Execute `docker-compose up -d`

Projects to jumpstart your development efforts

- Docker For Drupal (docker4drupal.org)
- Docker based development environment (drupal.org), https://www.drupal.org/node/2736447
  - https://github.com/peperoni60/drupal-docker
- Tons more on Github, pick your flavor
- Docker Hub: https://hub.docker.com
- Docker Sync (for OS X) (http://docker-sync.io/)

Questions?