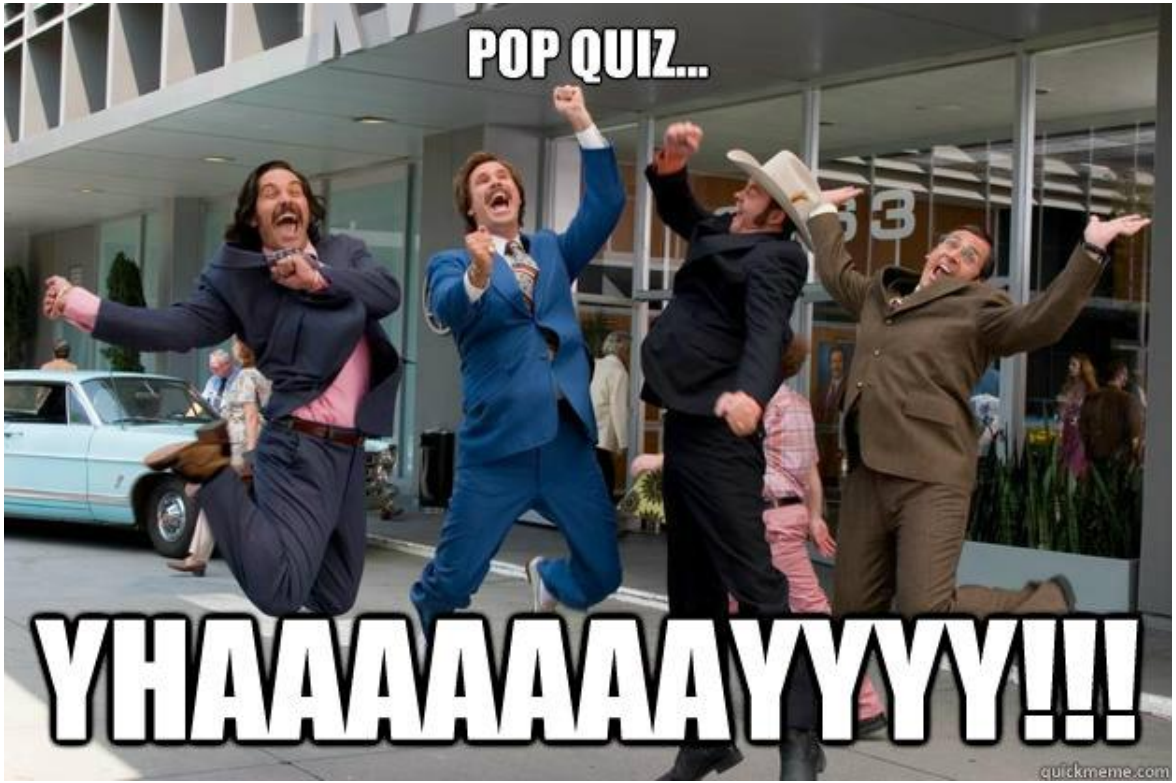
Docker Compose

Blacksburg Docker Meetup

October 12, 2016

Pop Quiz Time!

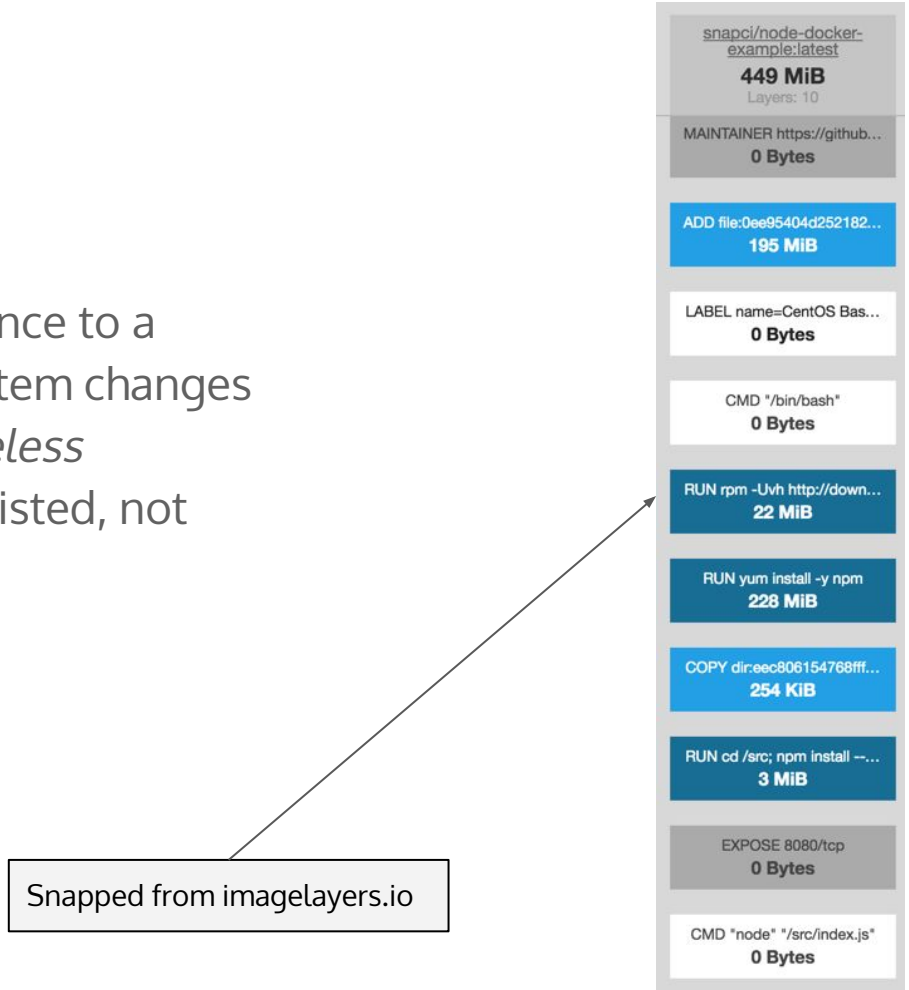


Q1: What’s an image?

(Not a JPG, GIF, or PNG)

Q1: What's an image?

- Layers of filesystem changes
  - Each layer contains a reference to a parent image and its filesystem changes
- **Images** are *immutable* and *stateless*
  - Only filesystem data is persisted, not running processes



Q2: How can we make new images?

(Hint: there are two ways we talked about. One's obviously preferred...)

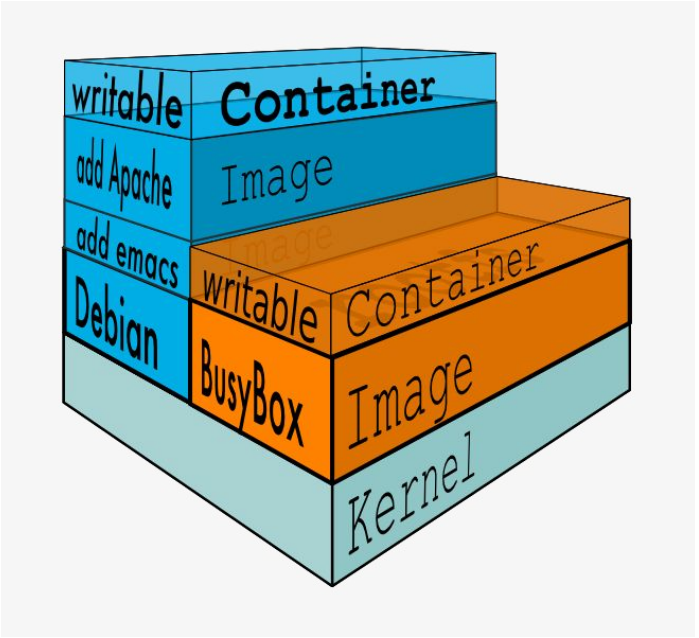
Q2: How can we make new images?

1. `docker commit [containerId] [tag]`
    a. Less preferred as it requires a container to exist with the changes
    b. Suffers from share-ability - can't easily hand a script off to reproduce the script
2. Use a Dockerfile
    a. Simply a script that provides steps to create new image
    b. Script can be shared and version controlled
    c. Definitely preferred method

Q3: What’s a container?

Q3: What's a container?

- Running *instances* of an image
- Each container is given its own writeable layer
    - Supports multiple concurrently running containers based on a single image



Q4: How do we run a container?

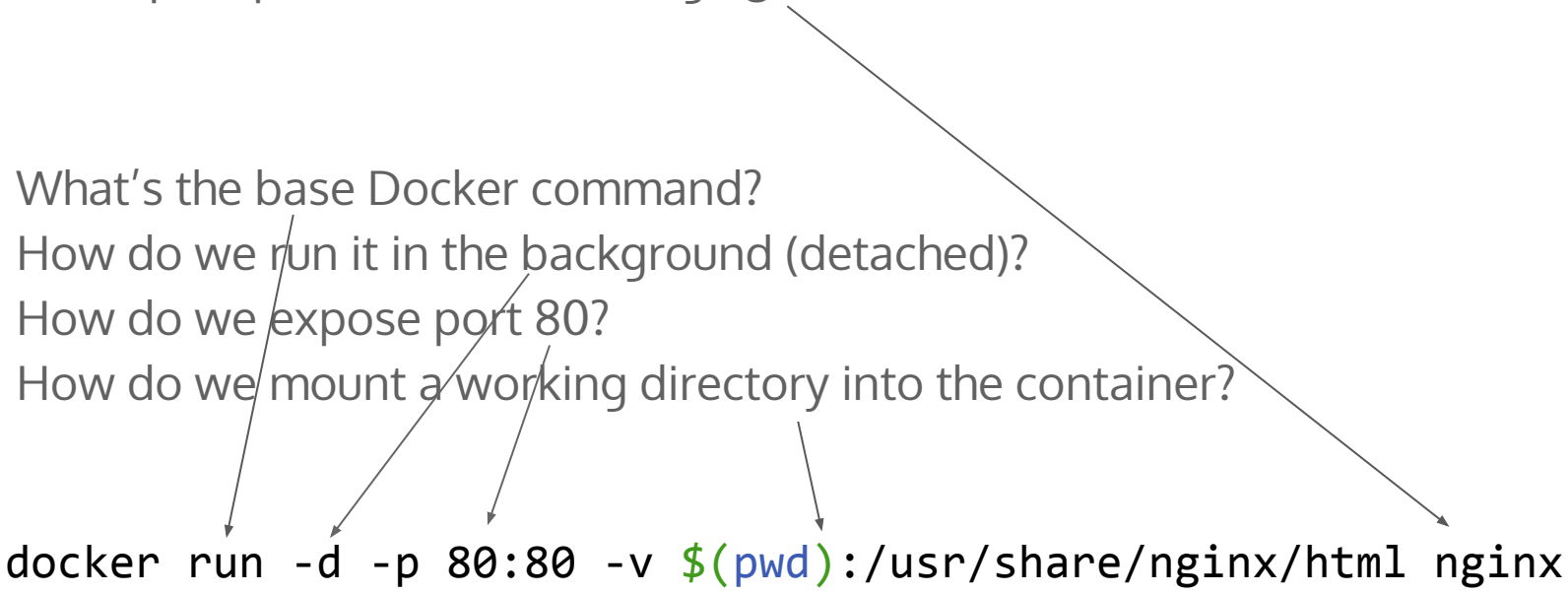
- Let's spin up a `nginx` image

- What's the base Docker command?
- How do we run it in the background (detached)?
- How do we expose port 80?
- How do we mount a working directory into the container?

Q4: How do we run a container?

- Let's spin up a container running `nginx`

- What's the base Docker command?
- How do we run it in the background (detached)?
- How do we expose port 80?
- How do we mount a working directory into the container?



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

But wait...


What about apps that aren’t fully self-contained and depend on databases, message brokers, caches, etc.??

External dependencies...

- What's the hostname/IP we're going to connect to?
- What port should we connect to?
- Are they running the expected version?

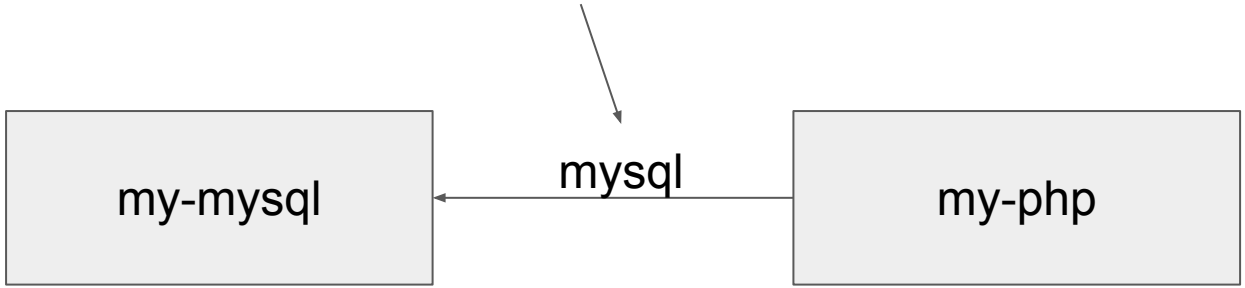

DISCOVERED
CONTAINER LINKING

Container Linking

- Provides the ability for one container to talk **directly** to another container
- Allows communication using a simple host name
  - Defaults to name of linked container, but can be changed
- Can communicate with ports not exposed on the host
  - Example - allow app to connect to MySQL without exposing MySQL to the outside world

An Example!

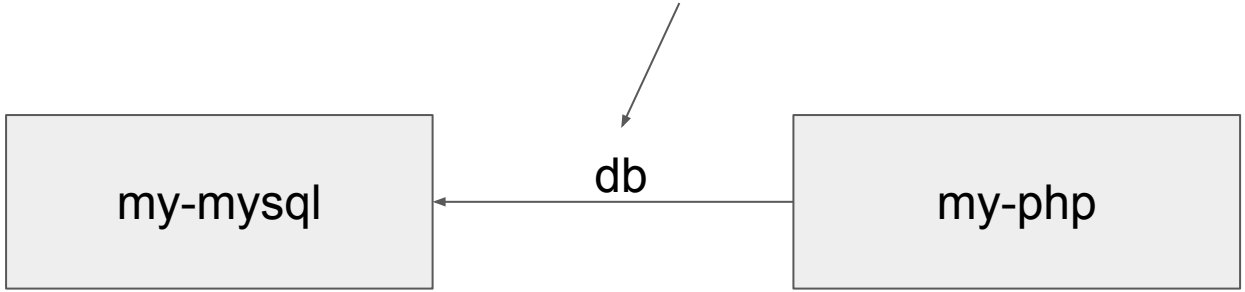
```
docker run -d --name mysql -e MYSQL_ROOT_PASSWORD=secret my-mysql
docker run -p 80:80 --link mysql my-php
```




```
/etc/hosts
172.17.0.2    mysql fca1535bb6b0
```

Changing the hostname (aliasing)

```
docker run -d --name mysql -e MYSQL_ROOT_PASSWORD=secret my-mysql
docker run -p 80:80 --link mysql:db my-php
```




```
/etc/hosts
172.17.0.2    db fca1535bb6b0 mysql
```

That gets complicated quickly...

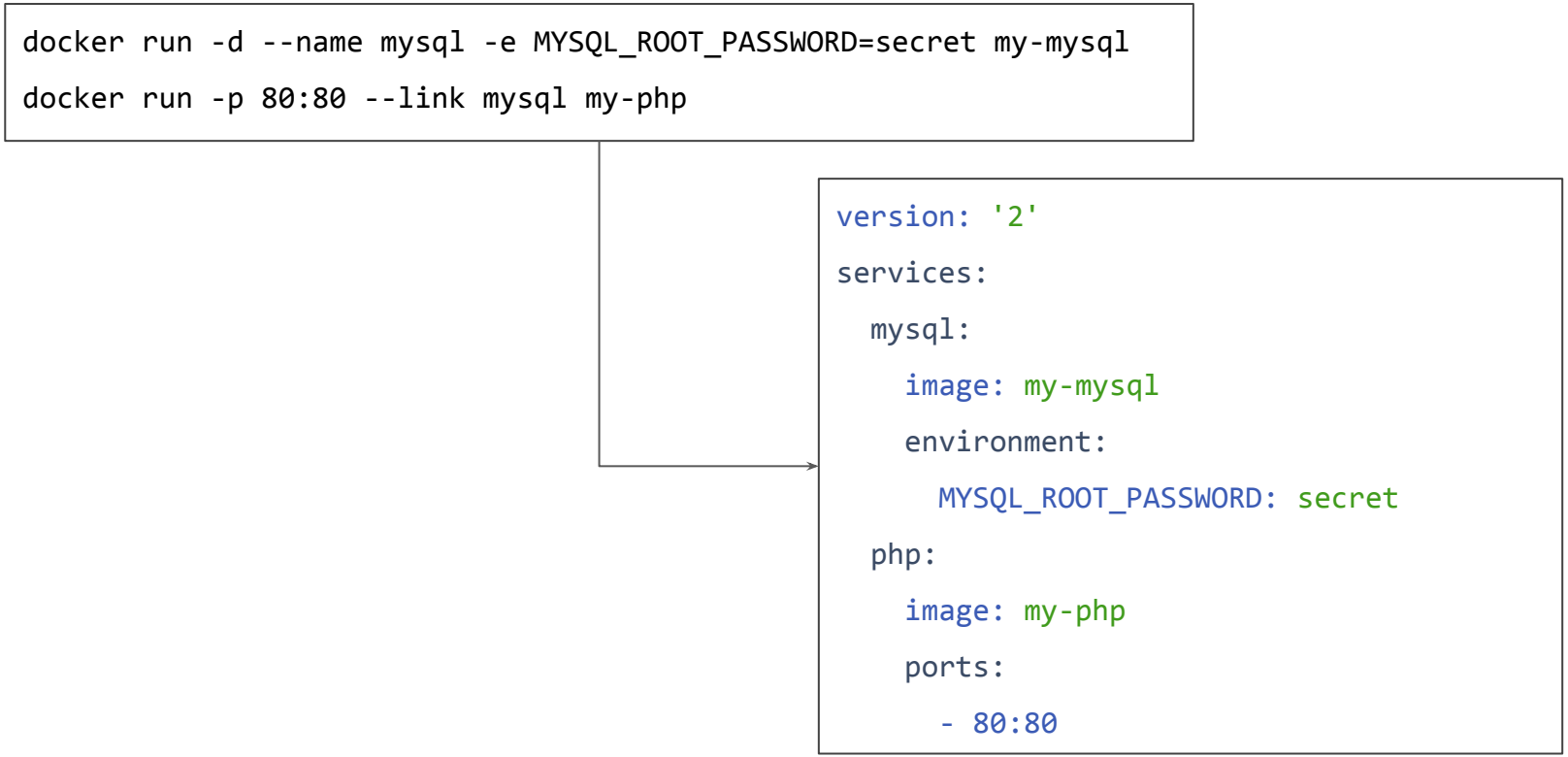

DISCOVERED
DOCKER COMPOSE

Quick intro to Docker Compose

- Uses YAML structure to outline the application stack
  - Default filename is docker-compose.yml
- Allows configuration for each container (image, ports, etc.)
- There are two versions (Version 1 and 2)
  - Version 2 is current (obviously) and adds support for networking and storage
  - Will still see many docker-compose.yml still using Version 1

First docker-compose.yml

```
docker run -d --name mysql -e MYSQL_ROOT_PASSWORD=secret my-mysql
docker run -p 80:80 --link mysql my-php
```



```yaml
version: '2'
services:
  mysql:
    image: my-mysql
    environment:
      MYSQL_ROOT_PASSWORD: secret
  php:
    image: my-php
    ports:
      - 80:80
```

Spinning it up

- Start the application stack by running: **`docker-compose up`**
- Will pull images (if needed) and start everything up
- All output from the container is combined for console output

```
02 -> docker-compose up
Creating 02_php_1
Creating 02_mysql_1
Attaching to 02_mysql_1, 02_php_1
mysql_1  | Initializing database
php_1    | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.19.0.2.
mysql_1  | 2016-10-12T04:04:19.735163Z 0 [Warning] TIMESTAMP with implicit DEFAULT value is deprecated. Please use --
php_1    | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.19.0.2.
php_1    | [Wed Oct 12 04:04:19.764720 2016] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.10 (Debian) PHP/7.0.11
php_1    | [Wed Oct 12 04:04:19.764768 2016] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
mysql_1  | 2016-10-12T04:04:20.644099Z 0 [Warning] InnoDB: New log files created, LSN=45790
mysql_1  | 2016-10-12T04:04:20.823593Z 0 [Warning] InnoDB: Creating foreign key constraint system tables.
mysql_1  | 2016-10-12T04:04:20.922285Z 0 [Warning] No existing UUID has been found, so we assume that this is the fir
mysql_1  | 2016-10-12T04:04:20.935522Z 0 [Warning] Gtid table is not ready to be used. Table 'mysql.gtid_executed' co
mysql_1  | 2016-10-12T04:04:20.936480Z 1 [Warning] root@localhost is created with an empty password ! Please consider
mysql_1  | 2016-10-12T04:04:24.666903Z 1 [Warning] 'user' entry 'root@localhost' ignored in --skip-name-resolve mode.
mysql_1  | 2016-10-12T04:04:24.666963Z 1 [Warning] 'user' entry 'mysql.sys@localhost' ignored in --skip-name-resolve
mysql_1  | 2016-10-12T04:04:24.666986Z 1 [Warning] 'db' entry 'sys mysql.sys@localhost' ignored in --skip-name-resolv
mysql_1  | 2016-10-12T04:04:24.667022Z 1 [Warning] 'proxies_priv' entry '@ root@localhost' ignored in --skip-name-res
mysql_1  | 2016-10-12T04:04:24.667139Z 1 [Warning] 'tables_priv' entry 'sys_config mysql.sys@localhost' ignored in --
mysql_1  | Database initialized
mysql_1  | MySQL init process in progress...
mysql_1  | MySQL init process in progress...
```

But wait… we skipped the link!

- Contents of `/etc/hosts` doesn't have an entry
- Container is configured to use Docker-provided DNS
  - DNS server is specific to the network
  - When starting up Docker compose, a new overlay network is created

```
root@260aa8dcaa8a:/var/www/html# cat /etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.19.0.2      260aa8dcaa8a
```

```
root@260aa8dcaa8a:/var/www/html# cat /etc/resolv.conf
search local
nameserver 127.0.0.11
options ndots:0
```

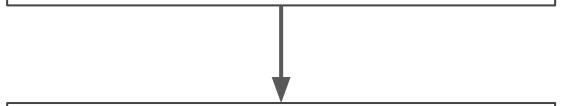
```
root@260aa8dcaa8a:/var/www/html# nslookup mysql
Server:         127.0.0.11
Address:        127.0.0.11#53

Non-authoritative answer:
Name:   mysql
Address: 172.19.0.3
```

What else can we do?

- We still have the image specified, which is an image not in the central hub
- We can use the `build` configuration to specify the location of a Dockerfile that can build the image

```
version: '2'
services:
  mysql:
    image: my-mysql
    environment:
      MYSQL_ROOT_PASSWORD: secret
  php:
    image: my-php
```



```
version: '2'
services:
  mysql:
    build: ./docker/mysql
    environment:
      MYSQL_ROOT_PASSWORD: secret
  php:
    build: ./docker/php
```

Wrapping up

- Docker Compose allows declarative application stacks in YAML
  - Default is docker-compose.yml
  - Allows for version controlling and easy sharing with others
- Can use either images available from a repo or build images
- Each composition is in its own network, making it easy to discover services

Questions?