linode

Containers And Docker

Lunch and Learn - September 2018

Part 1 of N

linode

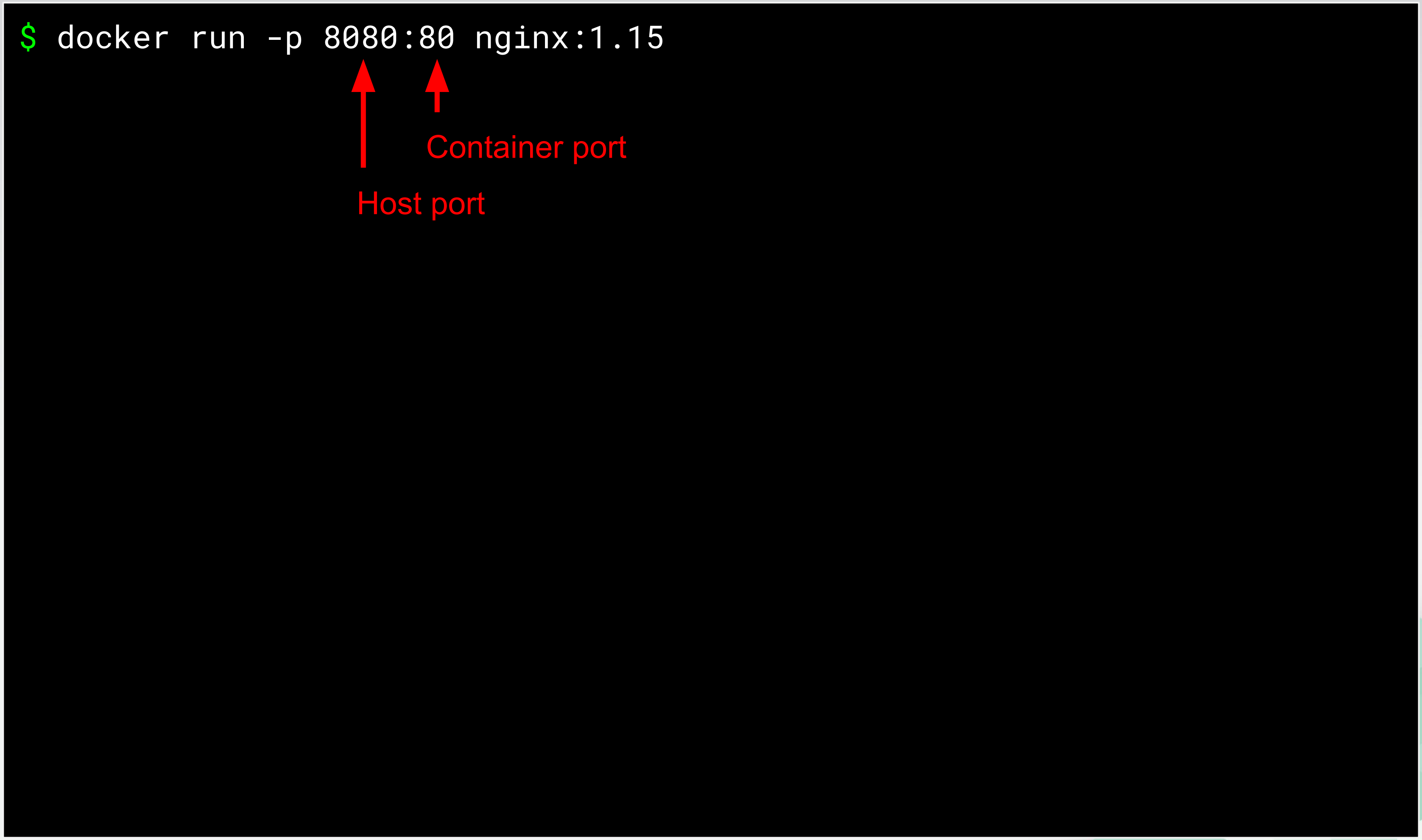
Follow Along!

https://github.com/jfrederickson/containers-presentation

Why?

Demo

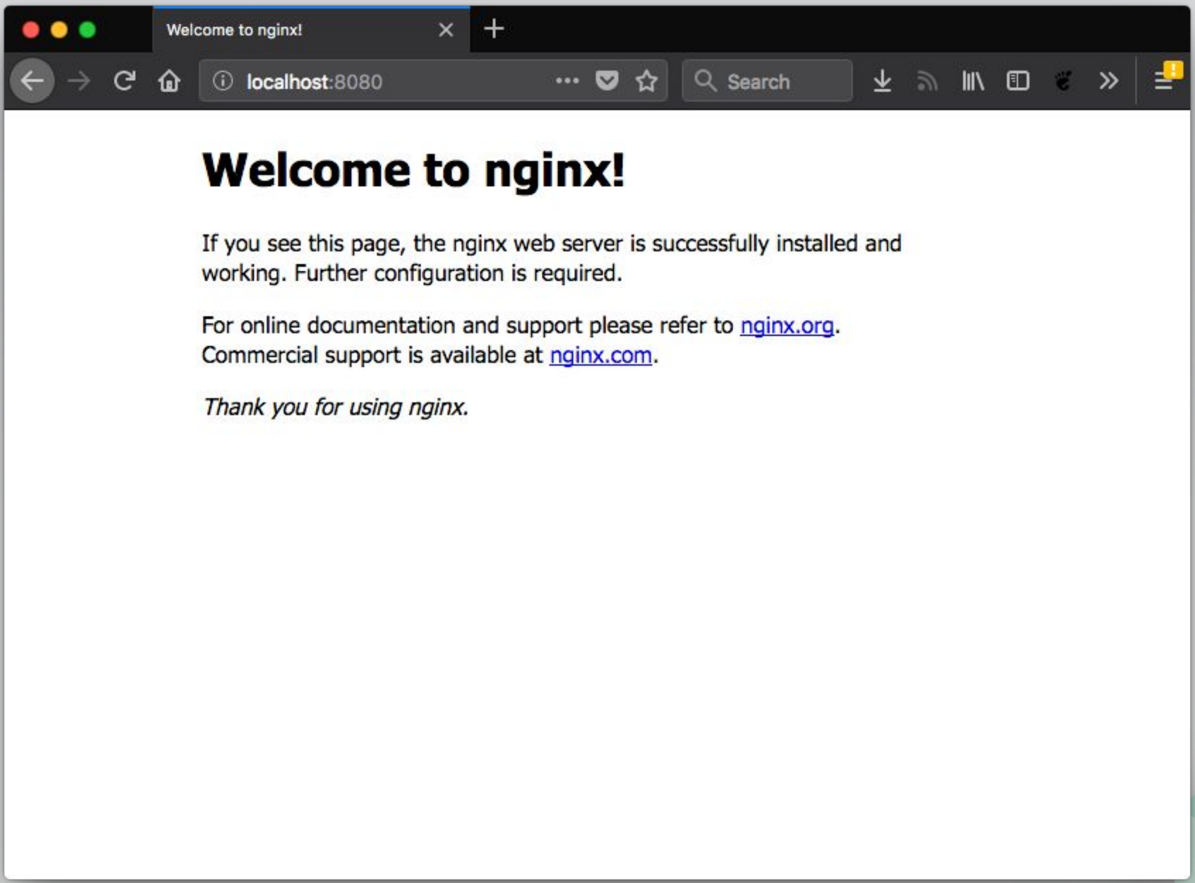
Using Containers



```
$ docker run -p 8080:80 nginx:1.15
```

linode

Using Containers



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

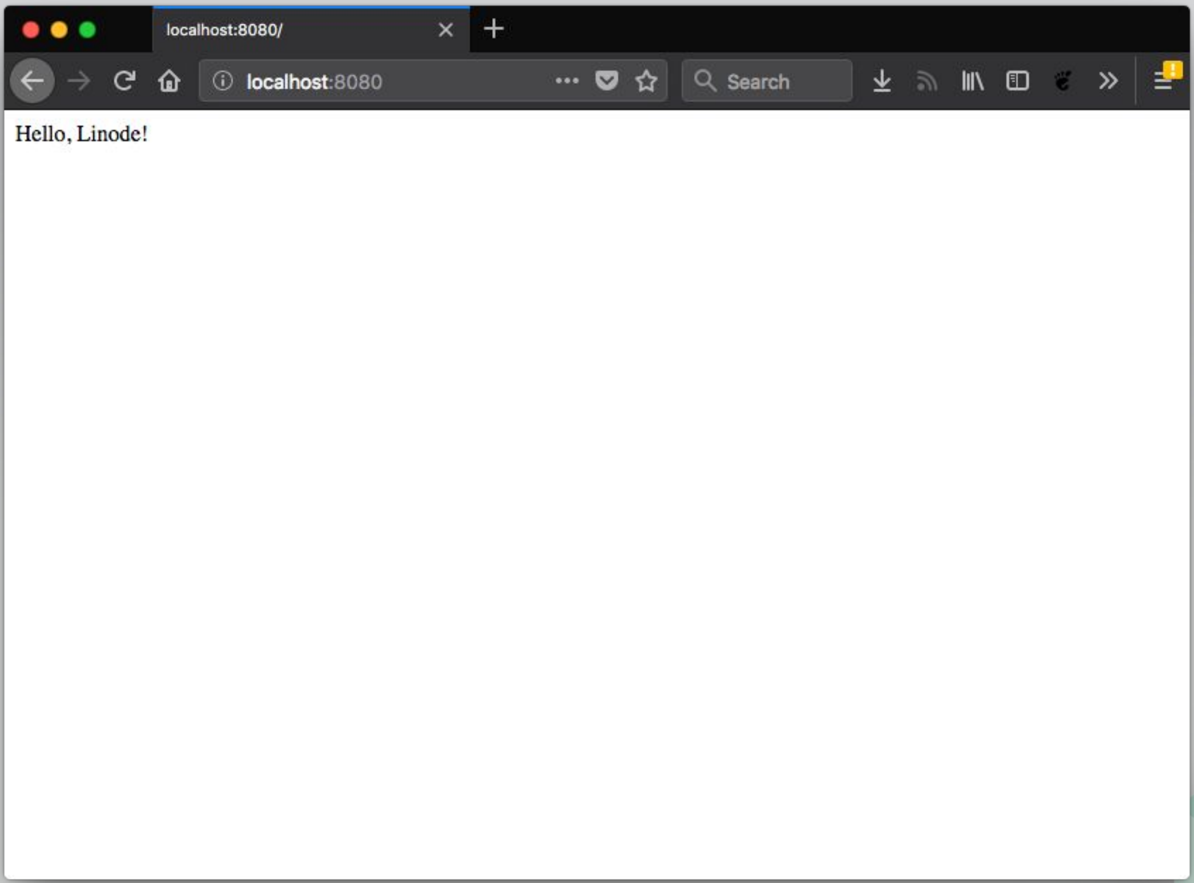
Thank you for using nginx.

linode

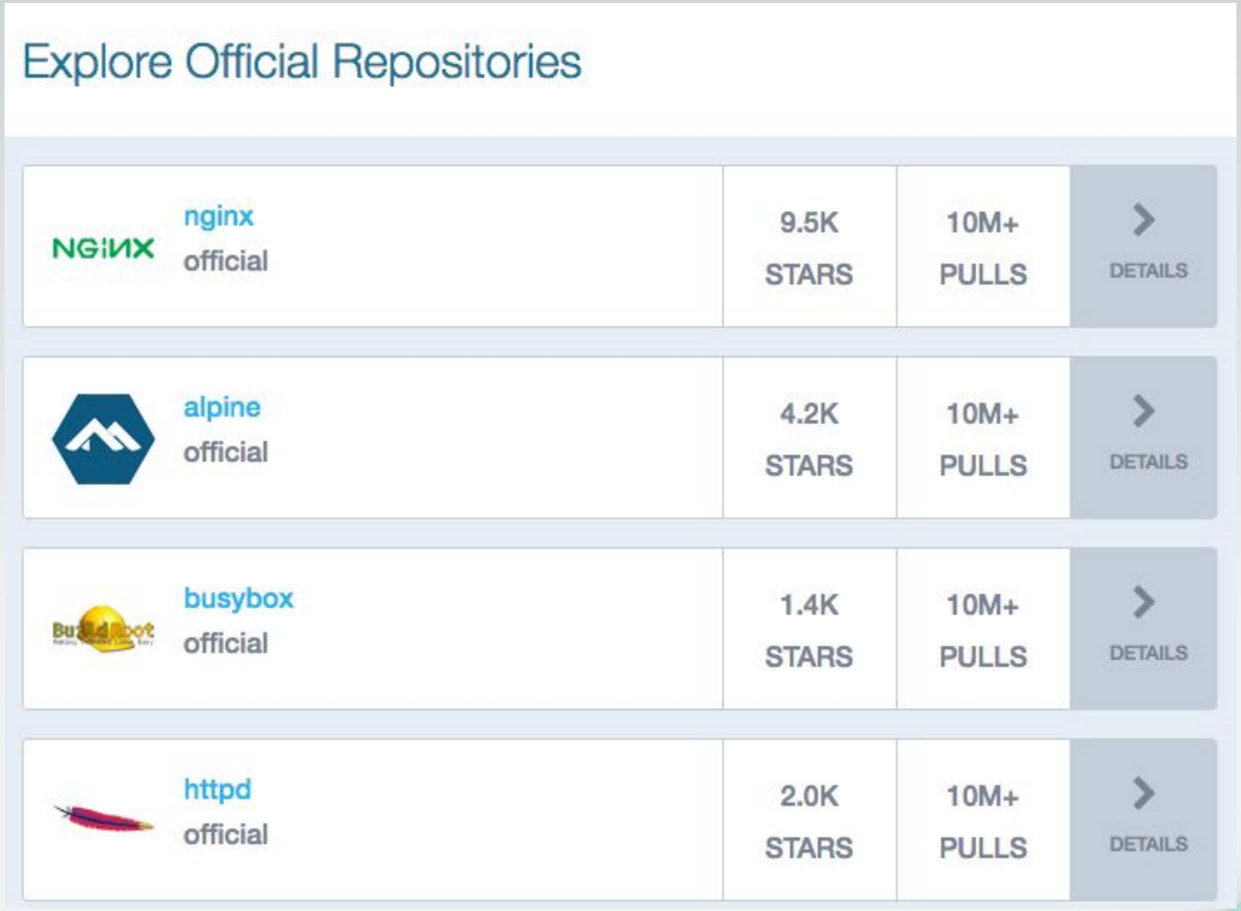
Building Containers

```
01-build$ cat Dockerfile
FROM nginx:1.15
COPY index.html /usr/share/nginx/html/index.html

01-build$ cat index.html
<html>
  <body>Hello, Linode!</body>
</html>

01-build$ docker build . -t lunchlearn:v1
Sending build context to Docker daemon  3.072kB
Step 1/2 : FROM nginx:1.15
 ---> b175e7467d66
Step 2/2 : COPY index.html /usr/share/nginx/html/index.html
 ---> 39452ffcd55d
Successfully built 39452ffcd55d
Successfully tagged lunchlearn:v1

01-build$ docker run -p 8080:80 lunchlearn:v1
```

Building Containers



Hello, Linode!



Explore Official Repositories

nginx official	9.5K STARS	10M+ PULLS	DETAILS
alpine official	4.2K STARS	10M+ PULLS	DETAILS
busybox official	1.4K STARS	10M+ PULLS	DETAILS
httpd official	2.0K STARS	10M+ PULLS	DETAILS

Anyone can upload to Docker Hub - look for official images!

Upgrades

Upgrades (Without Containers)

- Application specific!
- Could be a package upgrade, the app might have an upgrade UI, it might be interactive...
- Potentially hard to automate

Volumes make state explicit!

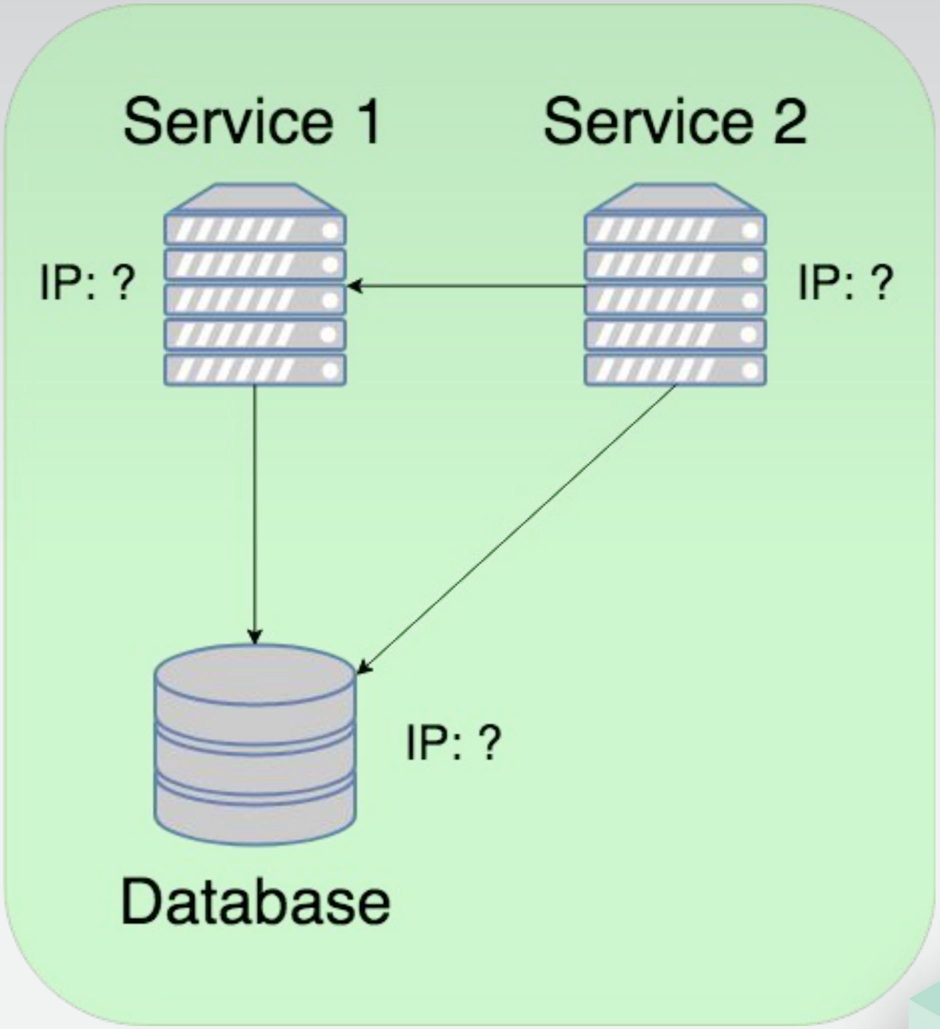
```
02-volumes$ cat nginx-conf/default.conf
server {
    listen       80;
    location / {
    root   /usr/share/nginx/html;
    index  test.html test.htm;
    }
}
02-volumes$ cat nginx-root/test.html
<html><body>Hello volumes!</body></html>

02-volumes$ docker run -p 8080:80 -v $PWD/nginx-conf:/etc/nginx/conf.d
-v $PWD/nginx-root:/usr/share/nginx/html nginx:1.15
```

Rollbacks - just start an older image

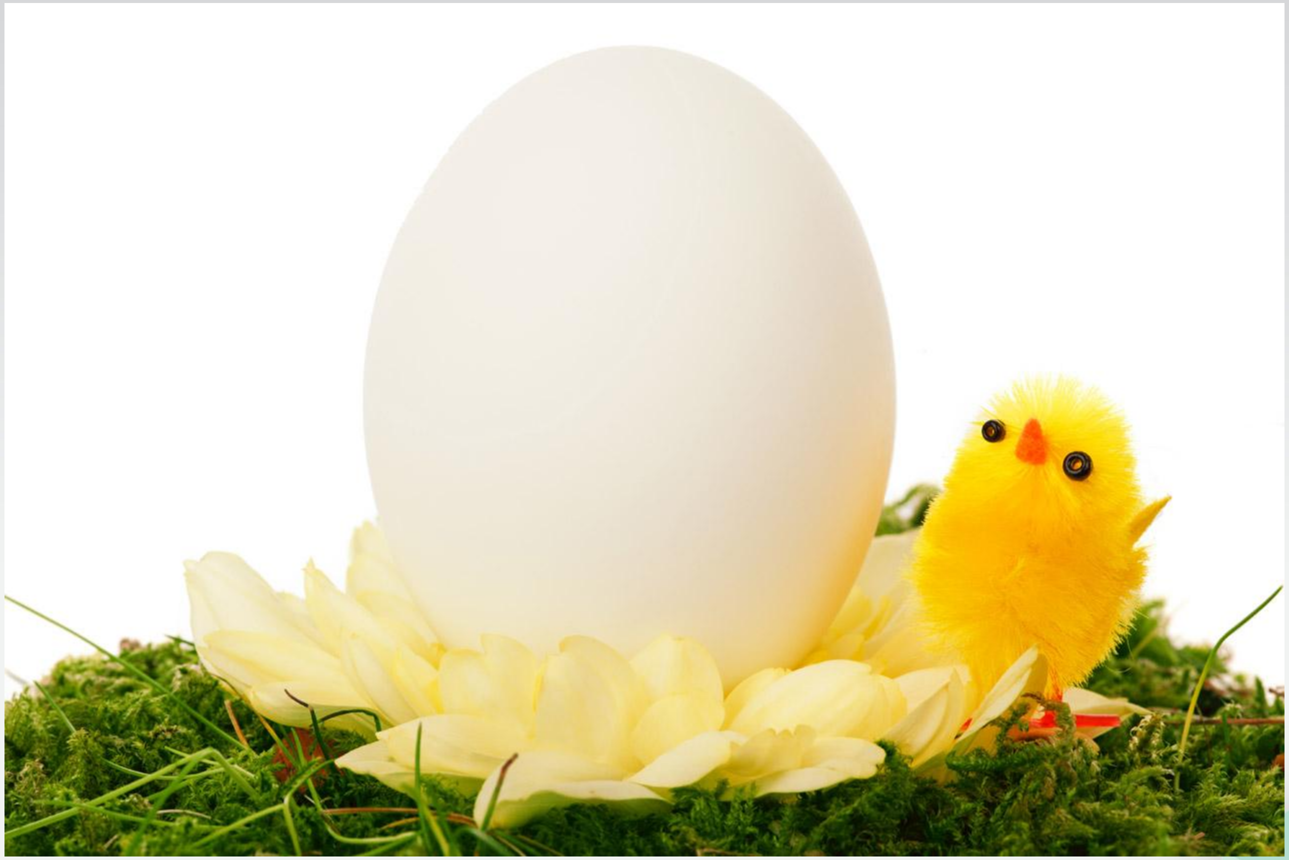
```
02-volumes$ cat nginx-conf/default.conf
server {
    listen       80;
    location / {
    root   /usr/share/nginx/html;
    index  test.html test.htm;
    }
}
02-volumes$ cat nginx-root/test.html
<html><body>Hello volumes!</body></html>

02-volumes$ docker run -p 8080:80 -v $PWD/nginx-conf:/etc/nginx/conf.d
-v $PWD/nginx-root:/usr/share/nginx/html nginx:1.15

02-volumes$ docker run -p 8080:80 -v $PWD/nginx-conf:/etc/nginx/conf.d
-v $PWD/nginx-root:/usr/share/nginx/html nginx:1.14
```

Orchestration

Orchestration (Manually)



Orchestration (Manually)



https://www.publicdomainpictures.net/en/view-image.php?image=13301&picture=chicken-and-egg

linode

Orchestration with Containers

```
03-orchestration$ cat docker-compose.yml
version: "3"
services:
  wordpress:
    image: wordpress:4.9-php5.6-apache
    environment:
      WORDPRESS_DB_HOST: "mysql"
      WORDPRESS_DB_PASSWORD: "secret"
    ports:
      - "8080:80"
  mysql:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: "secret"

03-orchestration$ docker-compose up
```

linode

Orchestration (You can build here too!)

```
04-orchestration-build$ cat docker-compose.yml
version: "3"
services:
  nginx:
    image: lunchlearn:v1
    build: .
    ports:
      - "8080:80"
  mysql:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: "secret"

04-orchestration-build$ cat Dockerfile
FROM nginx:latest
COPY index.html /usr/share/nginx/html/index.html

04-orchestration-build$ docker-compose up
```

You might need to build or pull between runs

- `docker-compose build` will rebuild images
- `docker-compose pull` will pull new image versions
- `docker-compose down -v` will shut down containers and delete their volumes

Scaling (Manually)

- Deploy a new machine, then
- Install your software on it manually, OR
- Enroll the new box in config mgmt and provision with an existing config
- Add its IP/hostname to the config of any system that wants to use it

linode

Scaling with Containers

```
05-scaling$ cat docker-compose.yml
version: "3"
services:
  wordpress:
    image: wordpress:4.9-php5.6-apache
    environment:
      WORDPRESS_DB_HOST: "mysql"
      WORDPRESS_DB_PASSWORD: "secret"
  mysql:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: "secret"
  loadbal:
    image: nginx:1.15
    volumes:
      - ./nginx-conf.d:/etc/nginx/conf.d
    ports:
      - "8080:80"

05-scaling$ docker-compose up
05-scaling$ docker-compose up --scale wordpress=3 # try this one too!
```

Best Practices

- For Docker specifically: https://docs.docker.com/develop/develop-images/dockerfile_best-practices/
  - Search “dockerfile best practices”
- For containers in general: https://12factor.net/

Keep applications stateless

Configure through environment variables or config files

linode

Keep dev and prod environments as close as possible

What NOT to do

- Download/build code on container startup
  - Makes rollbacks less reliable - if the remote server is down you won’t be able to roll back
  - Makes rollbacks take longer
  - Less confidence that you’re running exactly the same thing you were before
- Run code from a volume (in production)
  - Your containers should have all the code they need to start baked in
  - This is fine for local dev - that’s usually how it works (with e.g. hot reloading)

linode

Go forth and containerize!