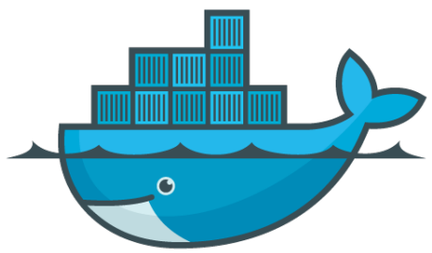
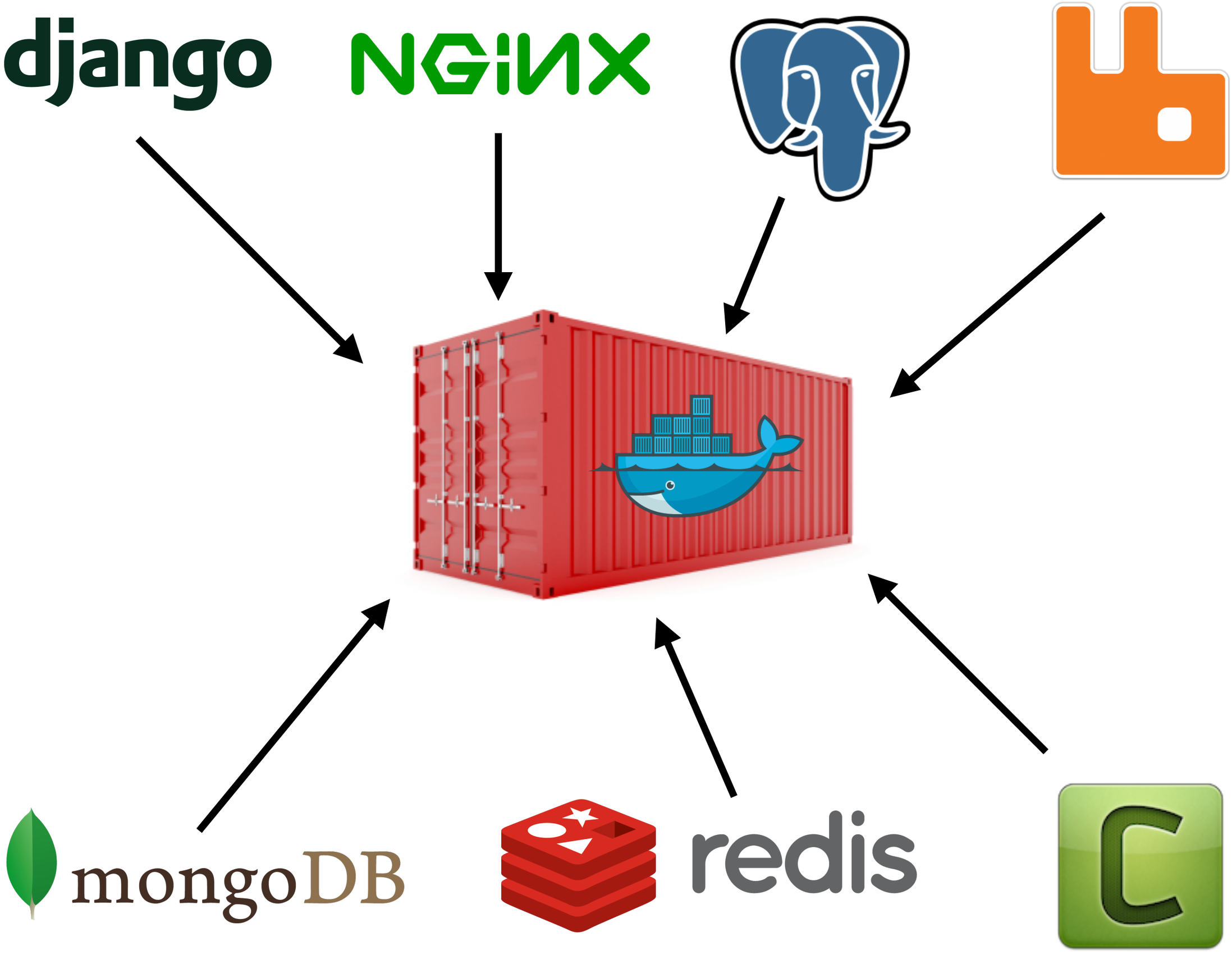
Introduction to 

Marconi Moreto

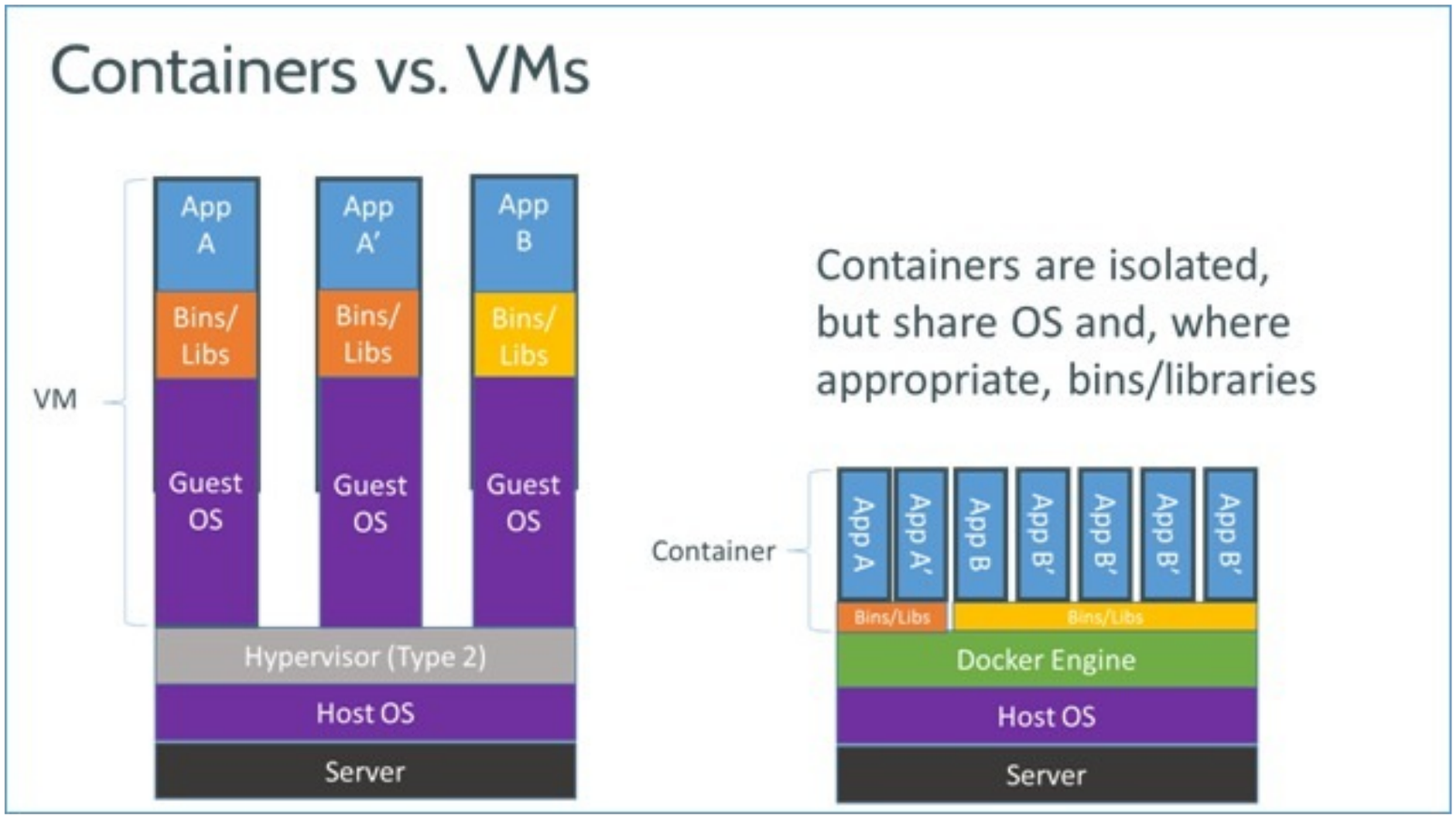
@marconimjr

What's Docker?

"An open source project to pack, ship and run any application as a lightweight container" - https://www.docker.io/

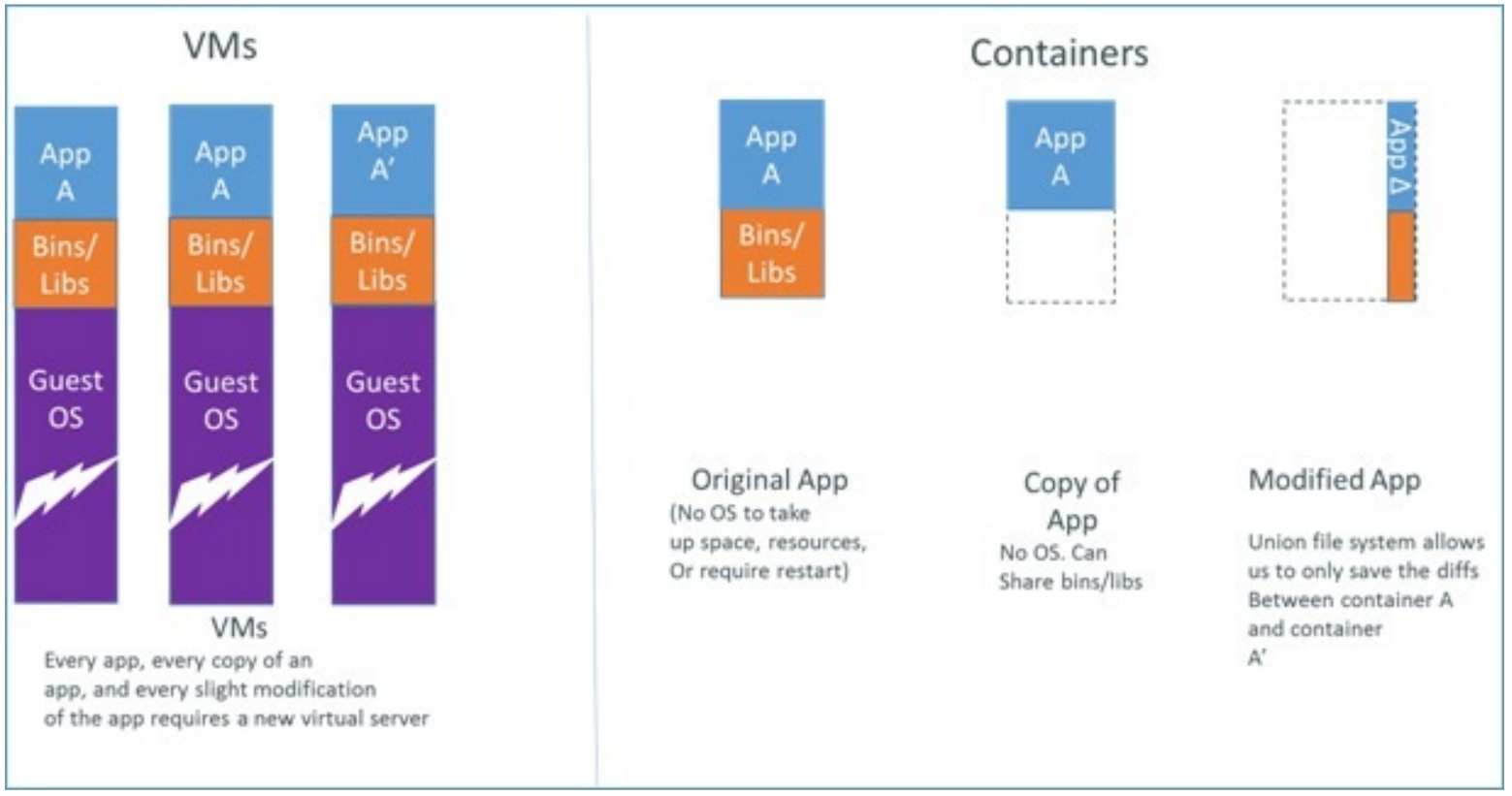

django
NGINX
mongoDB
redis

Containers VS VMs



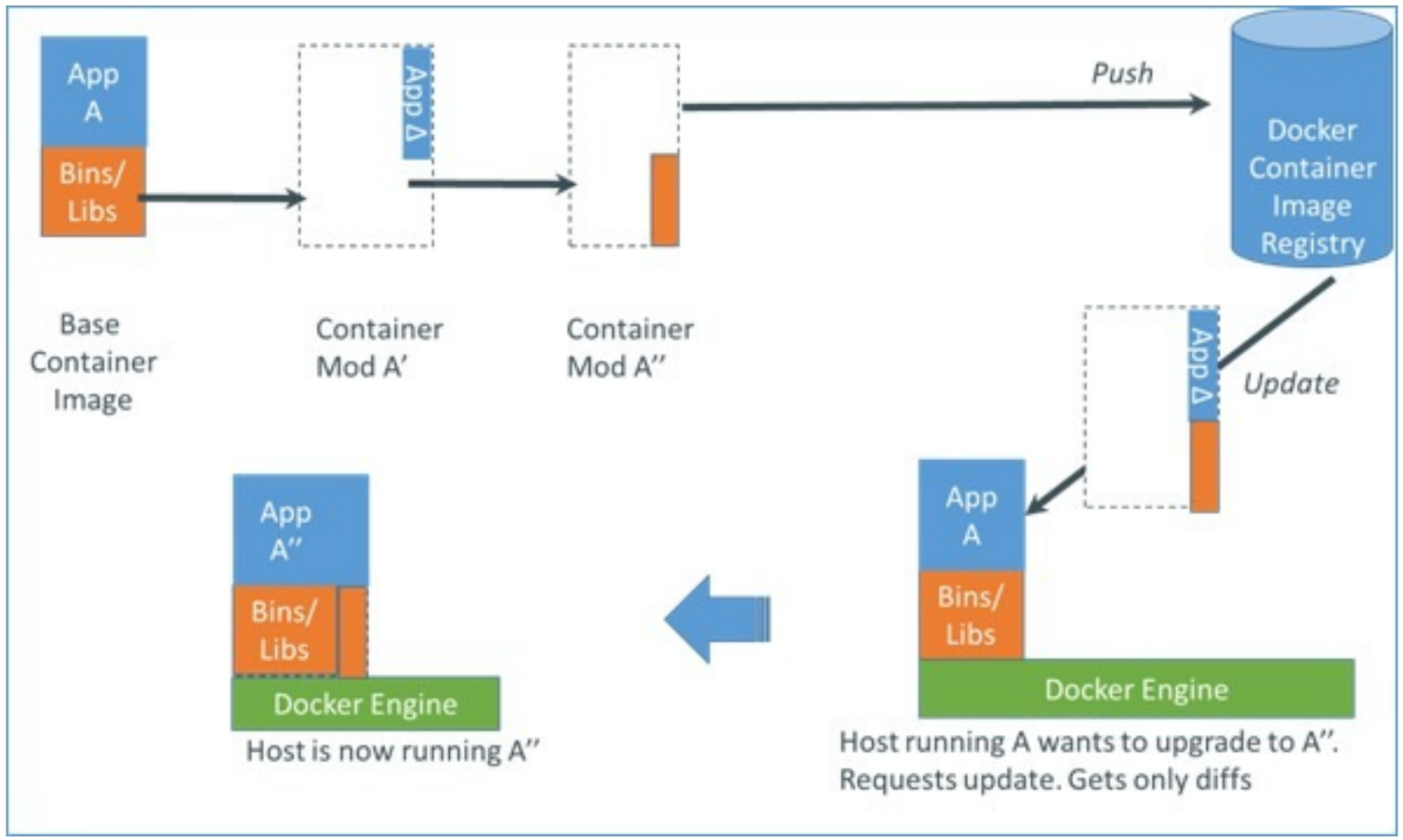
Source: https://www.docker.io/the_whole_story/

Containers are lightweight

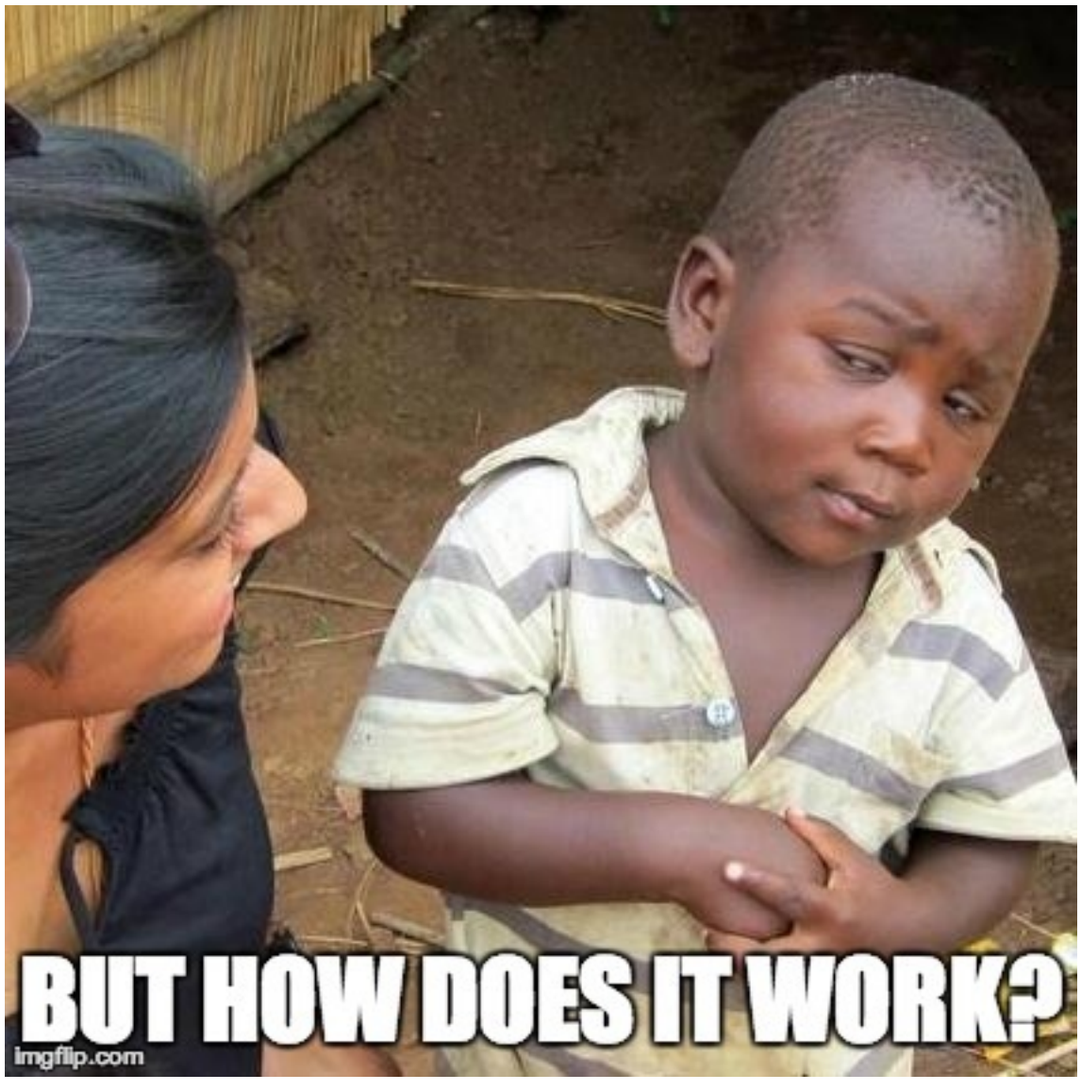


Source: https://www.docker.io/the_whole_story/

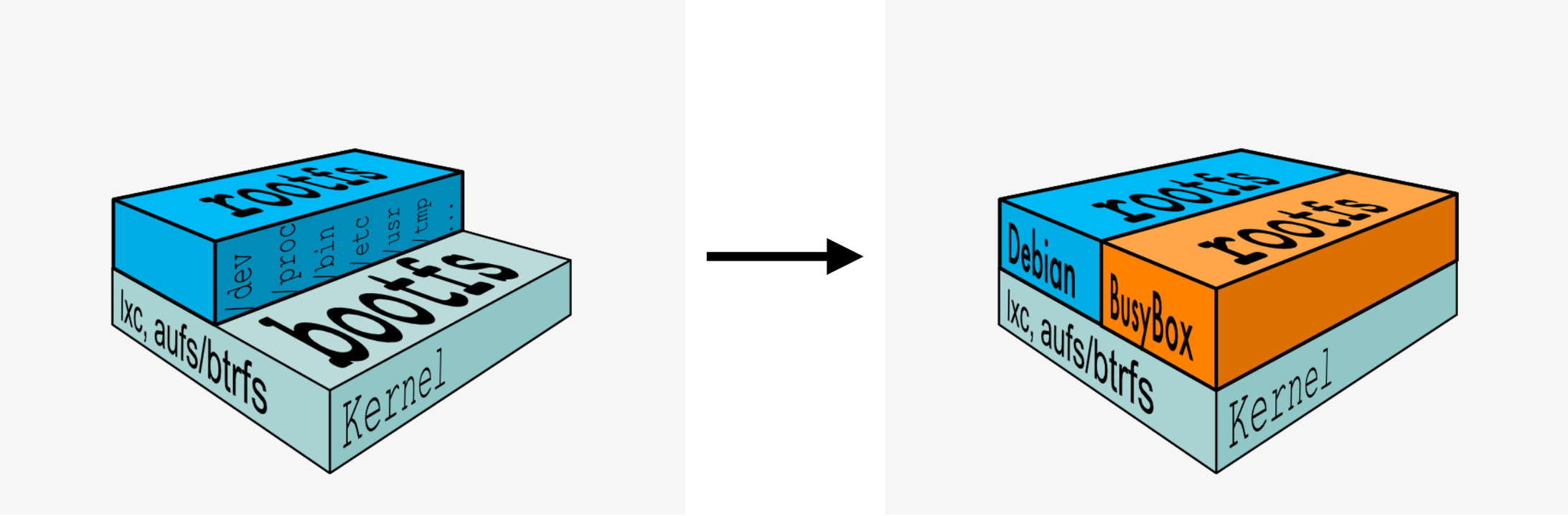
Changes and Updates



Source: https://www.docker.io/the_whole_story/

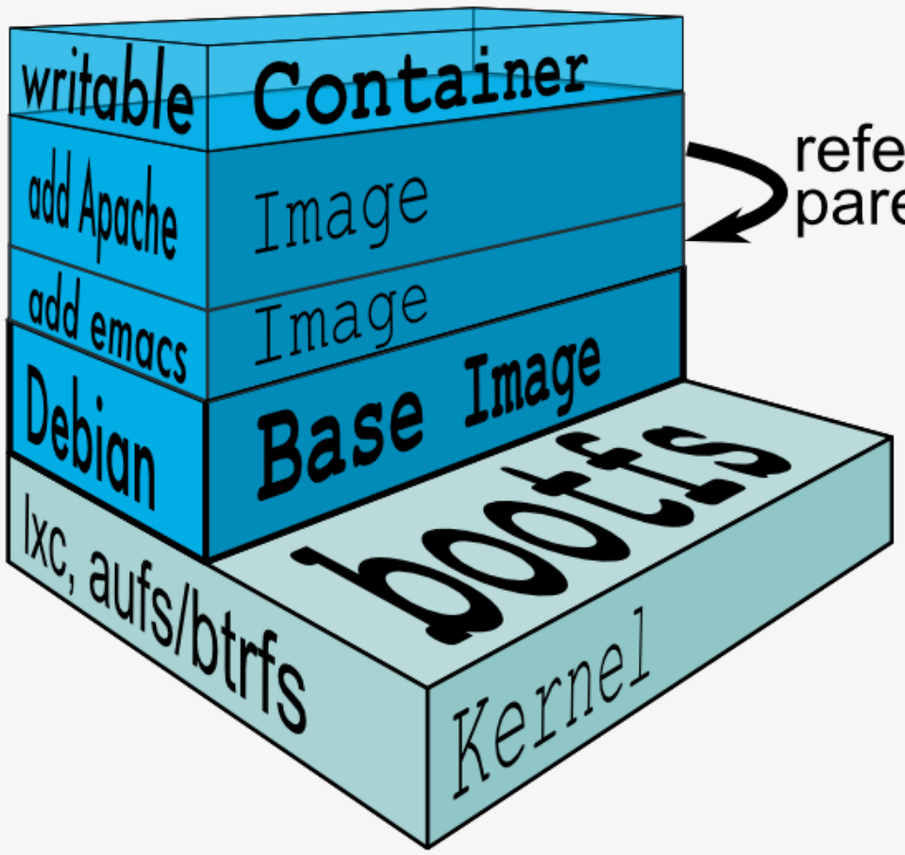

BUT HOW DOES IT WORK?
imgflip.com

File System



Source: http://docs.docker.io/en/latest/terms/filesystem/

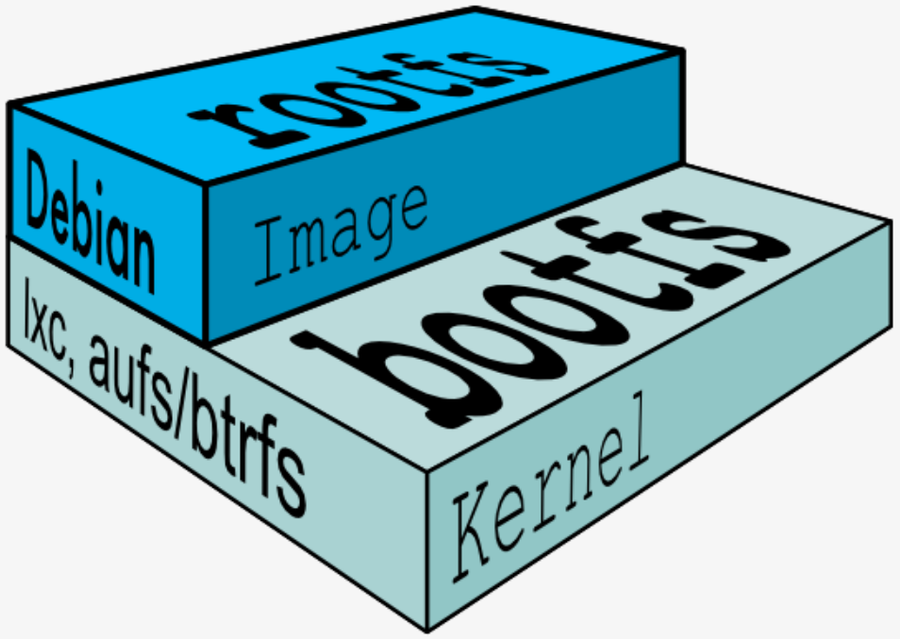
Layers



- Mounts rootfs as Read-Only
- Adds Read-Write on top of Read-Only layers
- Writes happens on top
- Any write from lower layers, will copy file to top and write happens on the copy

Source: http://docs.docker.io/en/latest/terms/layer/

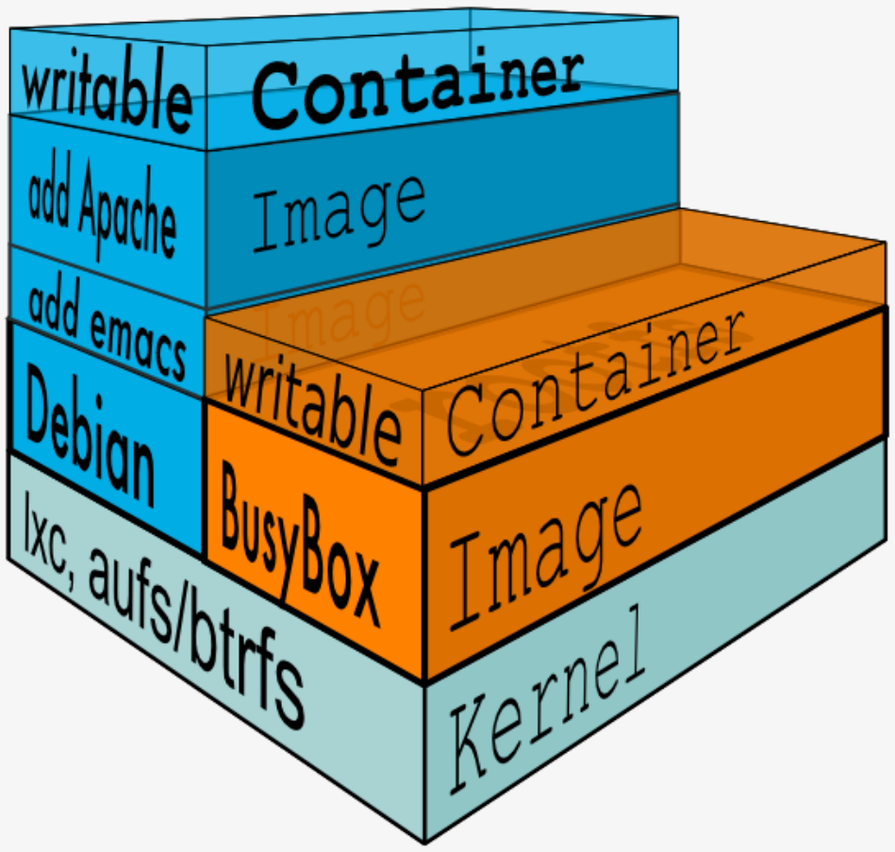
Image



- Images are read-only layers
- Images never changes
- Images can depend on another layers below it
- Image without parent is called base image

Source: http://docs.docker.io/en/latest/terms/image/

Container

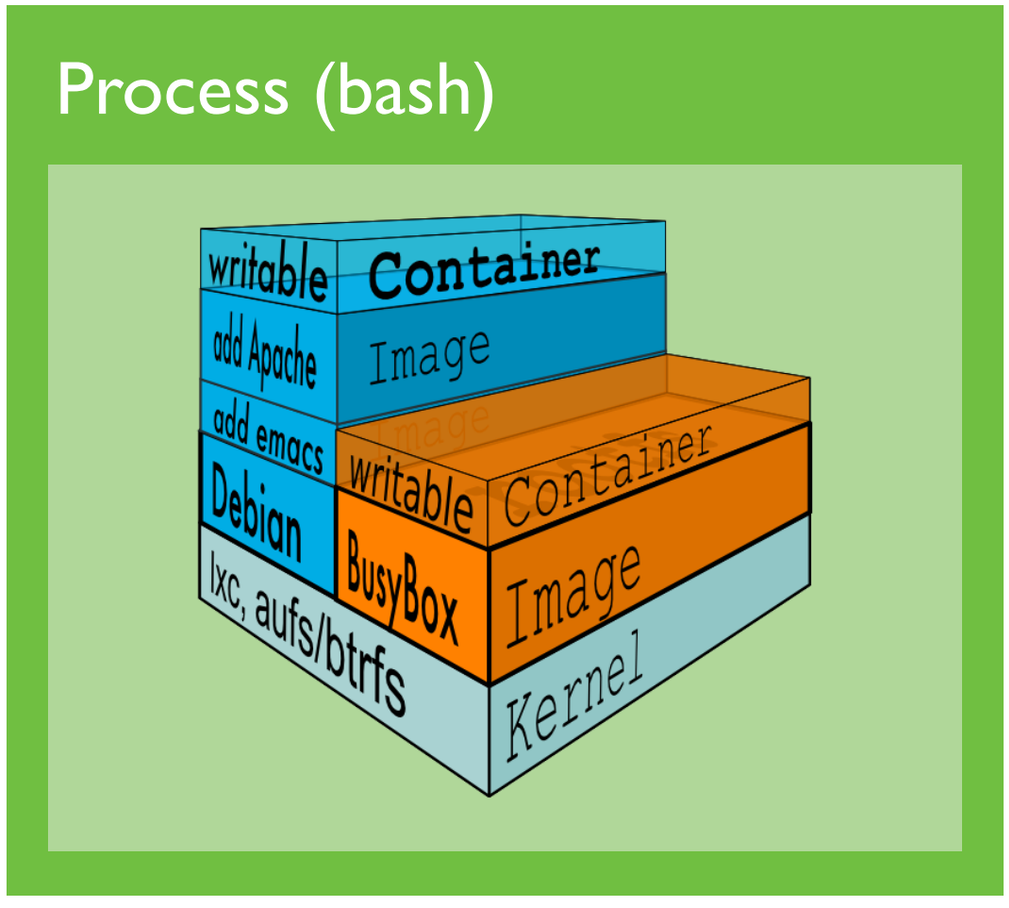


- Container: the Read-Write top layer + information about parent images, etc.
- Container have state: running or exited
- Container can be promoted to become an image via `docker commit`

Source: http://docs.docker.io/en/latest/terms/container/

Lets run it!

```
$ docker run -t -i ubuntu:12.04 /bin/bash
root@ca4af8c4c13f:/#
```



- Spawning a container based off of image will fetch that image, parent images, up-to its base image

- Then Docker adds the Read-Write layer on top which is now the container

Why you should care

Build once… (finally) run anywhere

- A clean, safe, hygienic and portable runtime environment for your app
- No worries about missing dependencies, packages and other pain points during subsequent deployments
- Run each app in its own isolated container, so you can run various versions of libraries and other dependencies for each app without worrying
- Automate testing, integration, packaging…anything you can script
- Reduce/eliminate concerns about compatibility on different platforms, either your own or your customers
- Cheap, zero-penalty containers to deploy services? A VM without the overhead of a VM? Instant replay and reset of image snapshots? That’s the power of Docker

What people have already built using Docker

Use Case	Examples	Link
Build your own PaaS	Dokku - Docker powered mini-Heroku. The smallest PaaS implementation you've ever seen	http://bit.ly/191Tgsx
Web Based Environment for Instruction	JiffyLab – web based environment for the instruction, or lightweight use of, Python and UNIX shell	http://bit.ly/12oaj2K
Easy Application Deployment	Deploy Java Apps With Docker = Awesome	http://bit.ly/11BCvvu
	Running Drupal on Docker	http://bit.ly/15MJS6B
	Installing Redis on Docker	http://bit.ly/16EWOKh
Create Secure Sandboxes	Docker makes creating secure sandboxes easier than ever	http://bit.ly/13mZGJH
Create your own SaaS	Memcached as a Service	http://bit.ly/11nL8vh
Automated Application Deployment	Push-button Deployment with Docker	http://bit.ly/1bTKZTo
Continuous Integration and Deployment	Next Generation Continuous Integration & Deployment with dotCloud's Docker and Strider	http://bit.ly/ZwTfoy
Lightweight Desktop Virtualization	Docker Desktop: Your Desktop Over SSH Running Inside Of A Docker Container	http://bit.ly/14RYL6x


IMMA BUILD
MY OWN SERVICE
imgflip.com

Name Service

app.py

```
import names
from flask import Flask

app = Flask(__name__)

@app.route("/name")
def name():
    return names.get_full_name()

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=9000)
```

requirements.txt

```
Flask==0.10.1
names==0.3.0
```

Dockerfile

```
FROM    ubuntu:12.04

# bundle our app
ADD . /src

# install requirements
RUN apt-get install python-pip -y
RUN cd /src; pip install -r requirements.txt

# run our app
EXPOSE    9000
CMD ["python", "/src/app.py"]
```

Build the image

```
$ cd path/to/our/files
$ docker build -t marconi/name-service .
…
```

Check image

```
$ docker images
(docker)
REPOSITORY                  TAG                 IMAGE ID
marconi/name-service        latest              9d52dd56be0f
…
```

Spawn container

```
$ docker run -p 9001:9000 -d marconi/name-service
f7991864b1dbf37ec5c5ddbf5e01a617779f491346dfde0ed32cf8180e
```

Check the container

```
$ docker ps
CONTAINER ID        IMAGE                              COMMAND                  ...   PORTS
f7991864b1db        marconi/name-service:latest   python /src/app.py   ...   0.0.0.0:9001-
>9000/tcp
```

Lets test it!

```
$ curl http://localhost:9001/name
Catherine Lewis
```

Lets share our image

```
$ docker push marconi/name-service
The push refers to a repository [marconi/name-service]
(len: 1)
Sending image list
Pushing repository marconi/name-service (1 tags)
511136ea3c5a: Image already pushed, skipping
Image 6170bb7b0ad1 already pushed, skipping
Image 9cd978db300e already pushed, skipping
9cfc6d137909: Image successfully pushed
b1b72907d441: Image successfully pushed
b6d2ecc5c0d2: Image successfully pushed
ac2907342d6b: Image successfully pushed
74bb3c8ebb13: Image successfully pushed
Pushing tag for rev [74bb3c8ebb13] on {https://
registry-1.docker.io/v1/repositories/marconi/name-service/
tags/latest}
```

Try it on your machine!

```
$ docker pull marconi/name-service
```

Thank you

Marconi Moreto

@marconimjr

http://marconijr.com

https://github.com/marconi