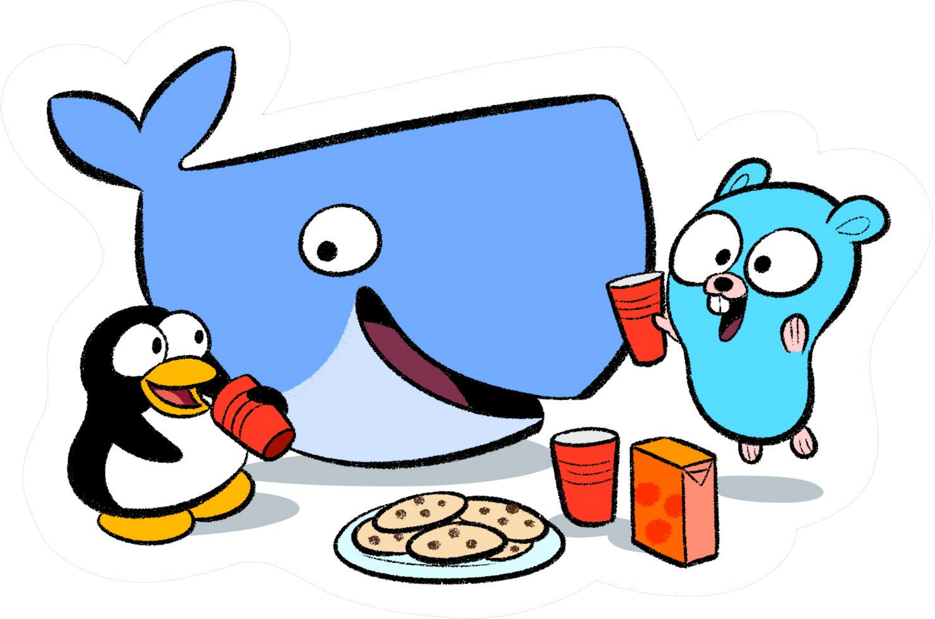
Introduction to Docker

Can Güney Aksakalli

April 20, 2016



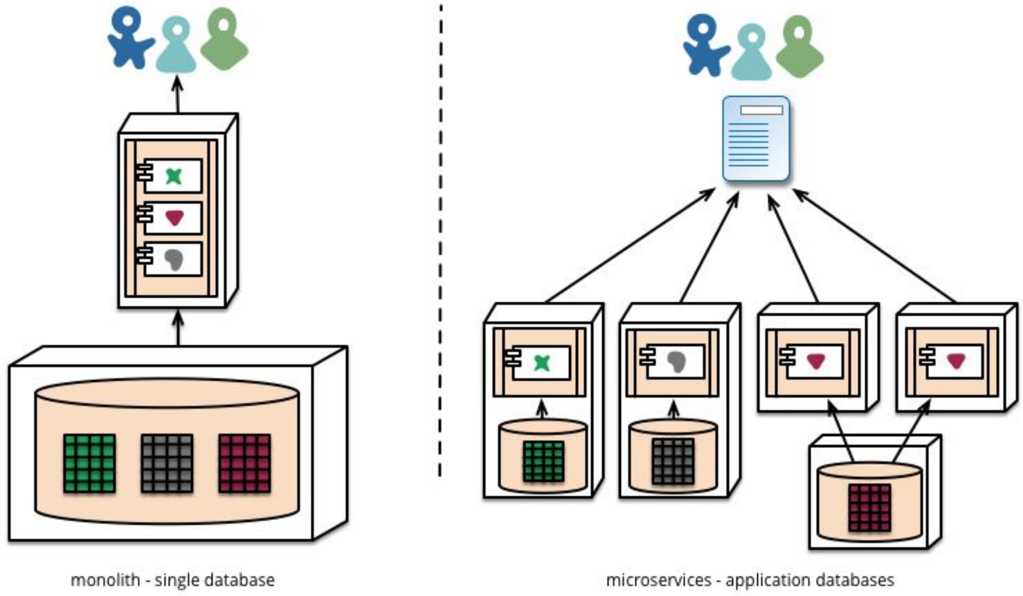
Content

- Motivation
- What is Docker?
- A Short Demo

Shipping code is hard

- Conflict runtimes
- Mapping ports and Services
- Something working in your machine might fail in another

It is harder for modern server architectures

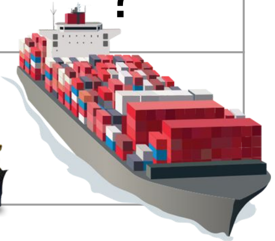

monolith - single database
microservices - application databases

Source: Martin Fowler “Microservices”

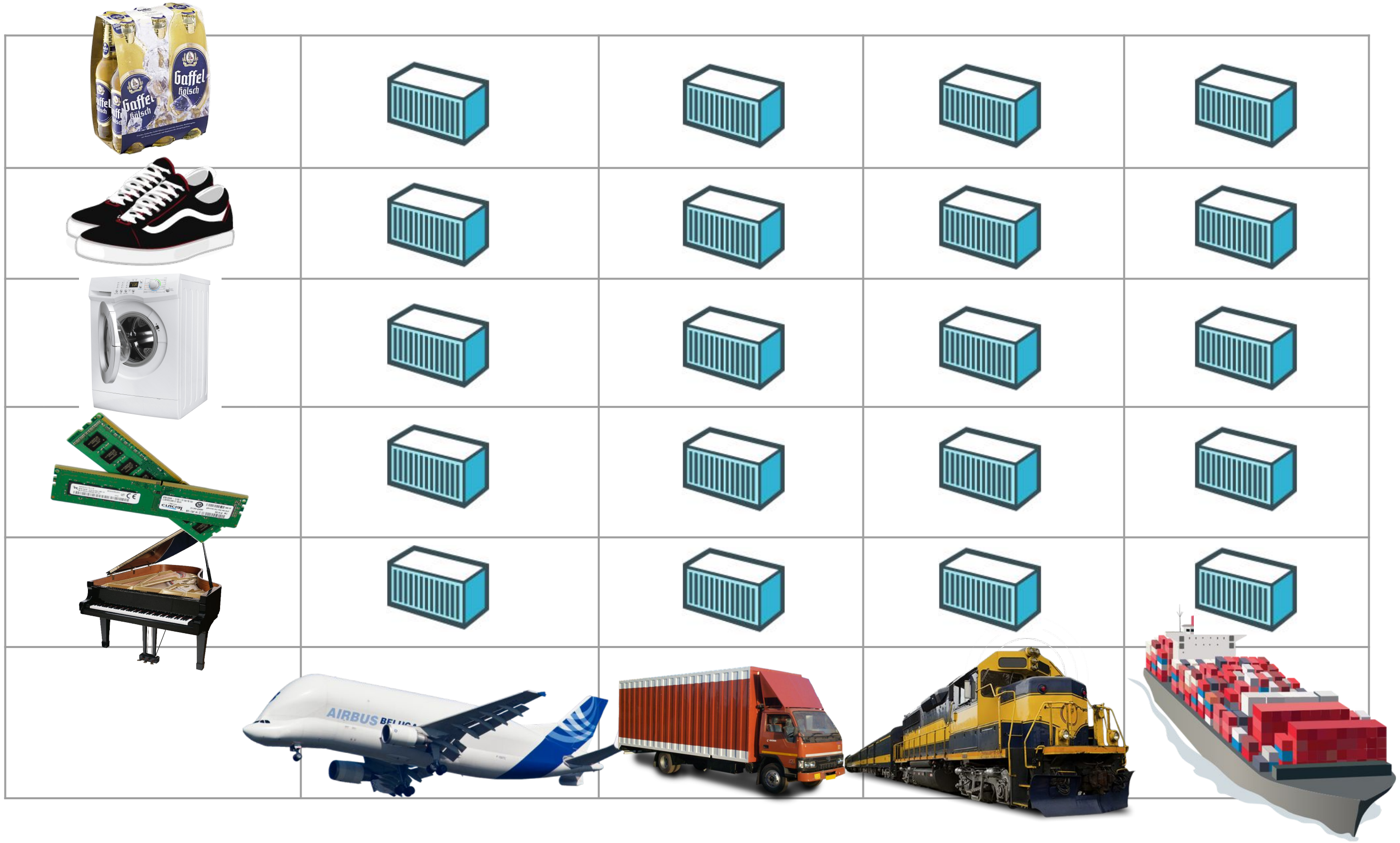
The Matrix From Hell

cassandra	?	?	?	?
node JS	?	?	?	?
Java	?	?	?	?
	?	?	?	?
NGINX	?	?	?	?
				☁

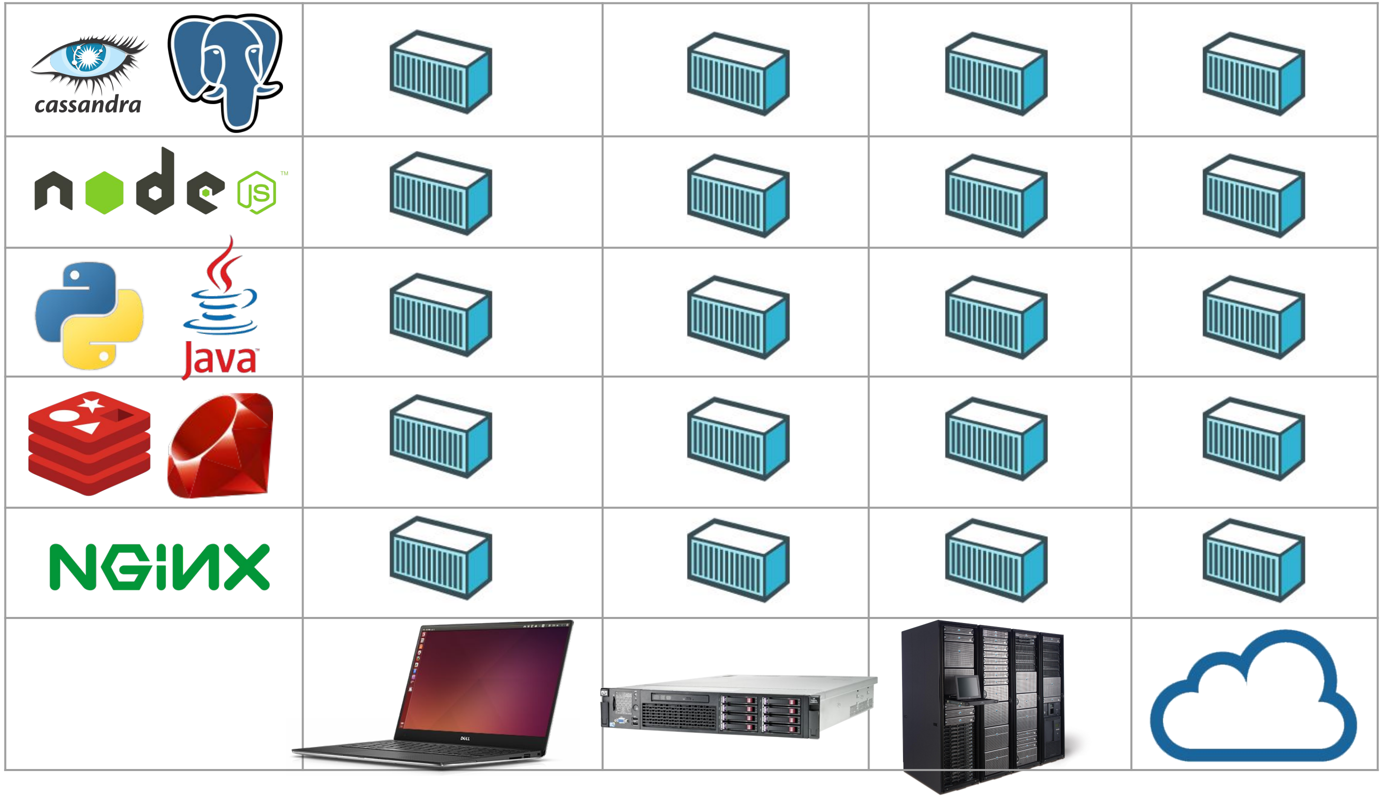
Another Matrix From Hell

	?	?	?	?
	?	?	?	?
	?	?	?	?
	?	?	?	?
	?	?	?	?
				

Solved with intermodal shipping container

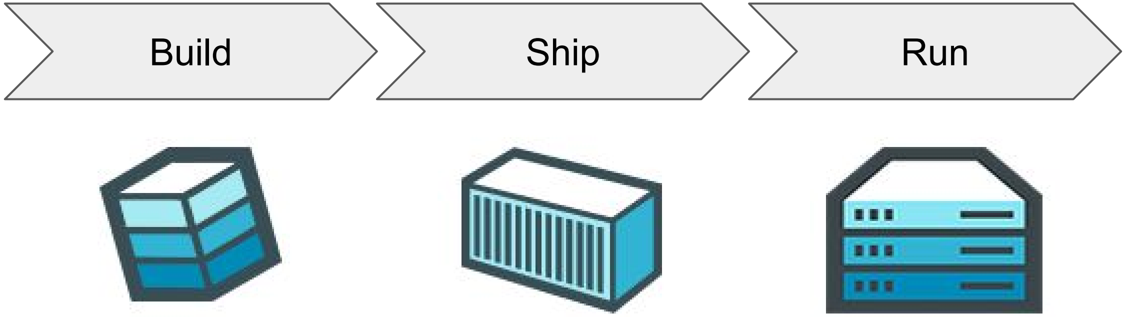


Solved with Docker Containers



What is docker ?

An **open*** platform for distributed applications for **developers** and **sysadmins**



Build

Ship

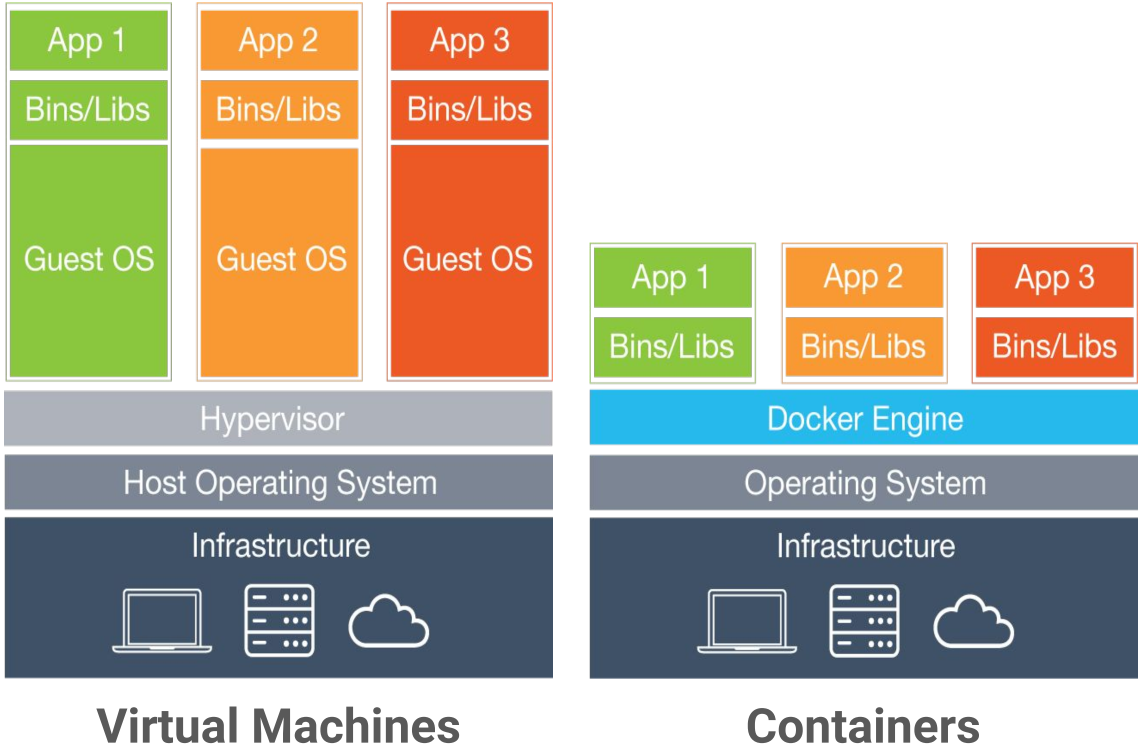
Run

Image

Container
(running image)

*licensed under the Apache License, Version 2.0

How is this different from virtual machines?

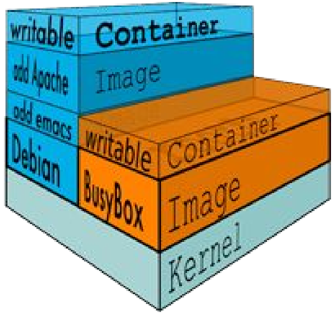


With Containers:

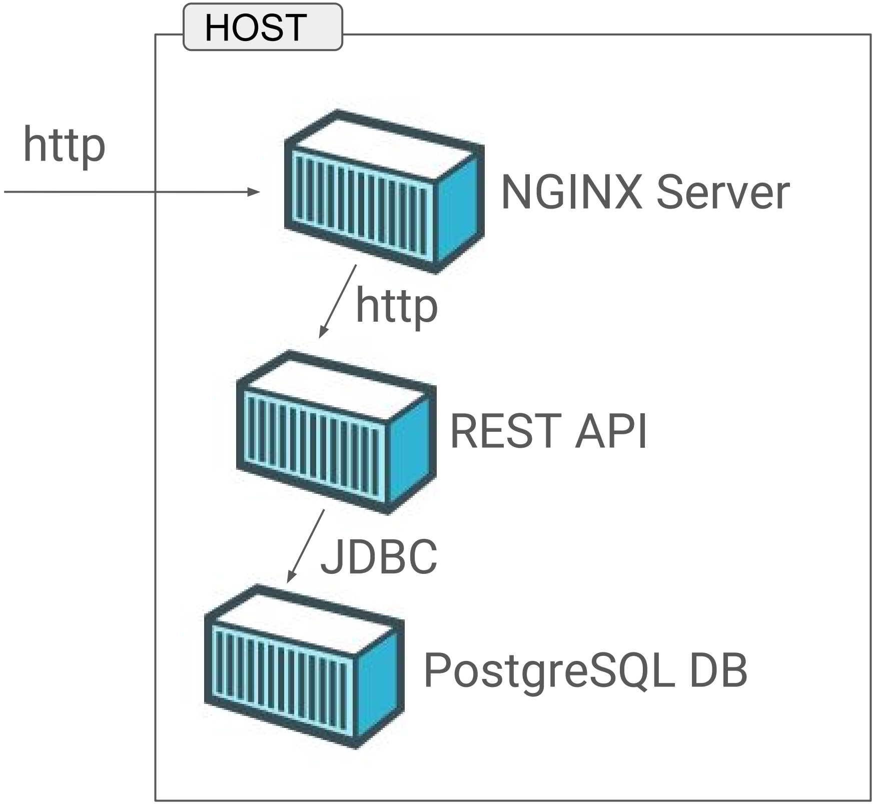
- Kernel level virtualization
  - Portable
  - Efficient
- Versioning
  - Only ship the diff
  - Semantic in images

What does Docker provide?

- Isolation: each container has isolated
  - Kernel namespaces
  - Network
  - Memory, CPU and disk I/O
- Security: a single container
  - cannot affect others
  - cannot bring the system down
- Portability across machines
  - Run your image in any Docker Host
- Rapid application deployment with consistent environment
  - Deploy quickly though Development→CI→Stage→Production



Let’s deploy a web app, shall we?



https://github.com/aksakalli/todo-spring-angular

A Simple TODO Web App Consist of

- **NGINX** serves static files of *SPA* and provides a *back proxy* to REST API.
- **REST API** is a Java Spring App and requires Java platform. It connects to DB through JDBC.
- **PostgreSQL DB** is responsible for persistency.

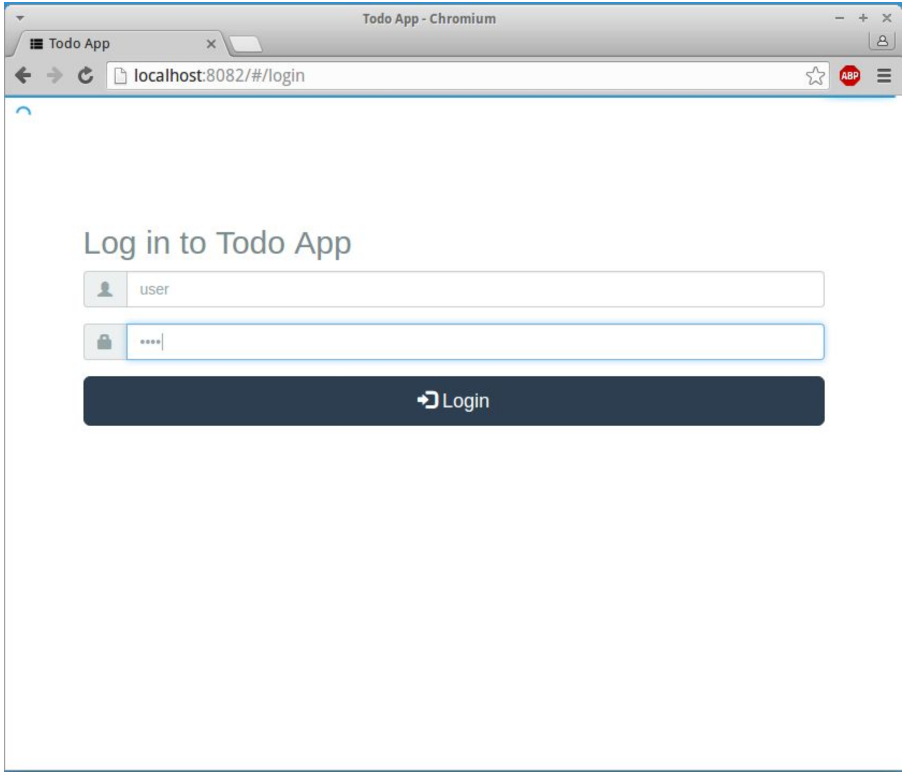
Running The Images from Docker Hub

Run The Simple Todo App with Docker

```
# first install Docker

# run the images from Docker Hub
docker run -d --name todo-db \
  -e POSTGRES_USER=postgres \
  -e POSTGRES_PASSWORD=postgres \
  -e POSTGRES_DB=todo postgres:9.4.5

docker run -d --name todo-rest \
  --link todo-db:todo-db aksakalli/todo-rest

docker run --name todo-nginx \
  -p 8082:80 --link todo-rest:todo-rest -d aksakalli/todo-nginx
```

Enjoy Your ~~Over Engineered~~ Beautiful Todo App

- On http://localhost:8082
  - User: user
  - Password: user



Log in to Todo App

user

••••

Building and Running Your Own Images

Simple Development & Deployment

```
# clone the project
git clone https://github.com/aksakalli/todo-spring-angular.git
cd todo-spring-angular

# build the project binaries
mvn package

# build and run images with docker-compose (need to be installed)
docker-compose up
```

docker-compose.yml

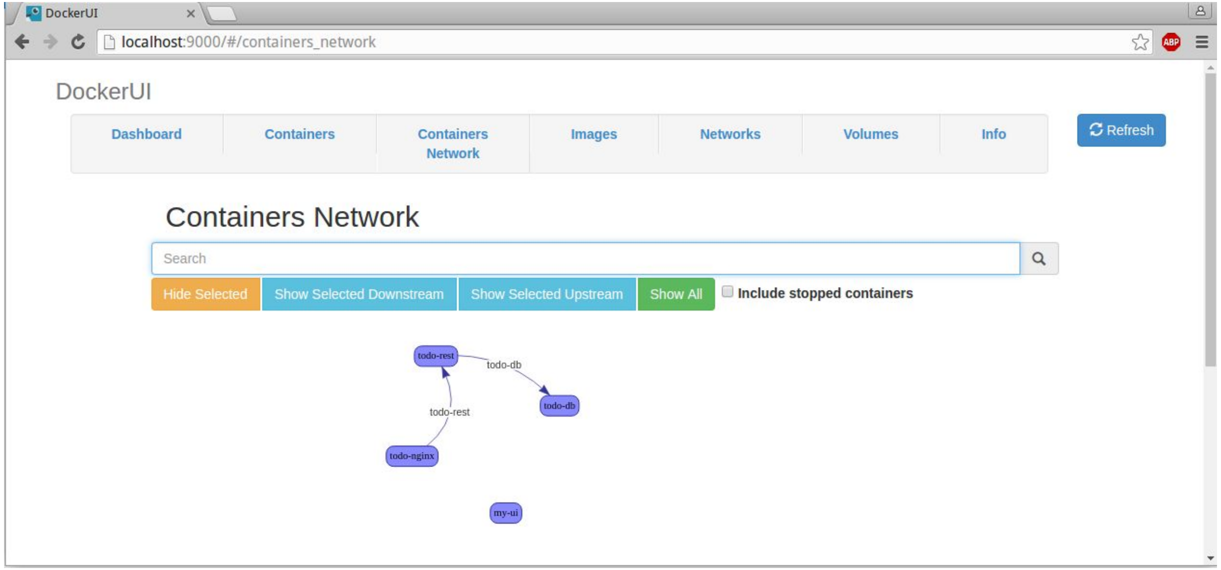
```
version: '2'
services:
 todo-nginx:
   build: todo-frontend/.
   ports:
     - "80:80"
   depends_on:
     - todo-rest
 todo-rest:
   build: todo-rest/.
   depends_on:
     - todo-db
 todo-db:
   image: postgres:9.4.5
   environment:
     POSTGRES_USER: postgres
     POSTGRES_PASSWORD: postgres
     POSTGRES_DB: todo
```

Alternative Way (without docker-compose)

```
# after building the project binaries

# build the images
docker build -t todo-rest todo-rest/.
docker build -t todo-nginx todo-frontend/.

# run the images
docker run -d --name todo-db \
  -e POSTGRES_USER=postgres \
  -e POSTGRES_PASSWORD=postgres \
  -e POSTGRES_DB=todo postgres:9.4.5
docker run -d --name todo-rest \
  --link todo-db:todo-db todo-rest
docker run --name todo-nginx \
  -p 8082:80 --link todo-rest:todo-rest -d todo-nginx
```

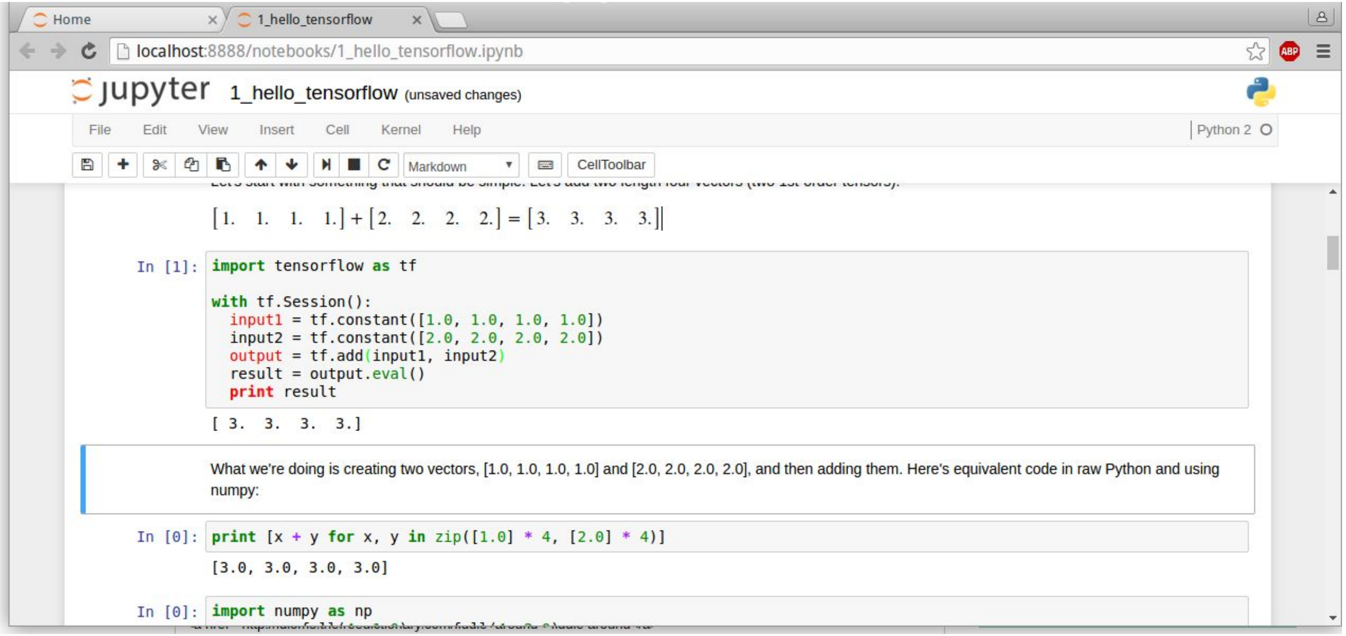
You Can Run Web-UI for Docker using Docker

```
docker run -d -p 9000:9000 --privileged \
  -v /var/run/docker.sock:/var/run/docker.sock kevan/dockerui
```



Or Fiddle Around with TensorFlow in a Minute

```
docker run --name my-tensor -p 8888:8888 \
  -v /home/can/notebooks:/notebooks b.gcr.io/tensorflow/tensorflow
```



Let's start with something that should be simple: Let's add two length four vectors (two 1st-order tensors):

$$[1.\quad 1.\quad 1.\quad 1.] + [2.\quad 2.\quad 2.\quad 2.] = [3.\quad 3.\quad 3.\quad 3.]$$

In [1]:
```
import tensorflow as tf

with tf.Session():
  input1 = tf.constant([1.0, 1.0, 1.0, 1.0])
  input2 = tf.constant([2.0, 2.0, 2.0, 2.0])
  output = tf.add(input1, input2)
  result = output.eval()
  print result
```

```
[ 3.  3.  3.  3.]
```

What we're doing is creating two vectors, [1.0, 1.0, 1.0, 1.0] and [2.0, 2.0, 2.0, 2.0], and then adding them. Here's equivalent code in raw Python and using numpy:

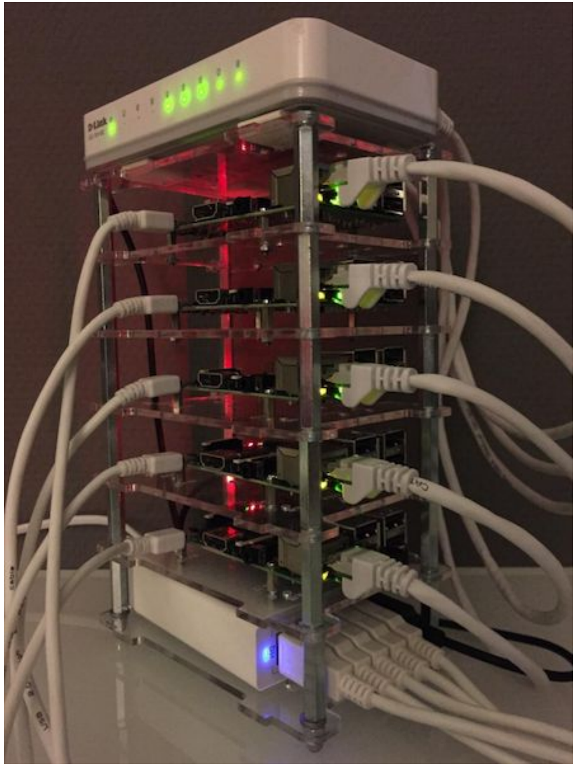
In [0]:
```
print [x + y for x, y in zip([1.0] * 4, [2.0] * 4)]
```

```
[3.0, 3.0, 3.0, 3.0]
```

In [0]:
```
import numpy as np
```

Management of Containers

- Orchestration Tools for Clusters
  - Kubernetes - http://kubernetes.io
    - Used in Google Cloud Platform
    - 708 contributors - 13k stars - 4k forks on GitHub
  - Docker Swarm - https://github.com/docker/swarm
  - Apache Mesos - http://mesos.apache.org
  - Helios - https://github.com/spotify/helios
- Monitoring Tools
  - Open-Source: cAdvisor https://github.com/google/cadvisor
  - Commercial: AppDynamics https://www.appdynamics.com
- Public/Private Docker Image Registry
  - Official: https://hub.docker.com
  - Open Source: https://github.com/docker/distribution



a Raspberry Pi cluster running with Kubernetes

Questions?

Thanks for your attention!