Docker 101

A very basic hands-on introduction

Allan Costa
@allanino

February 25, 2016

About the author

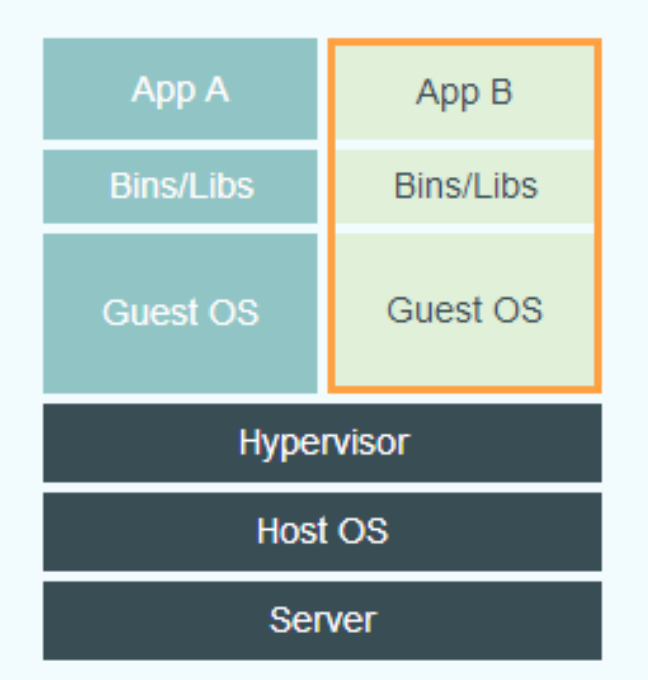
Started to work as developer at CloudWalk in August, 2013, having as goal to research and apply Machine Learning techniques to our business.

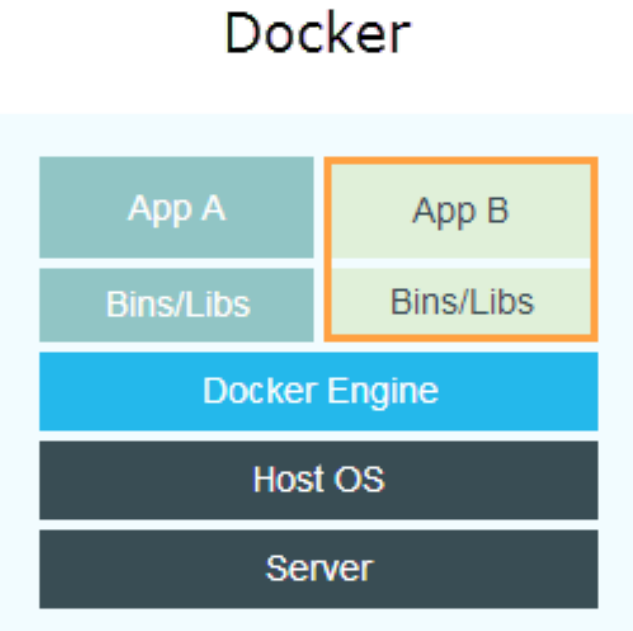
Docker was launched as an open-source project in March, 2013, so I didn't get a chance to live in dependency hell.

Some of my personal projects: `http://allanino.me/projects`

Content

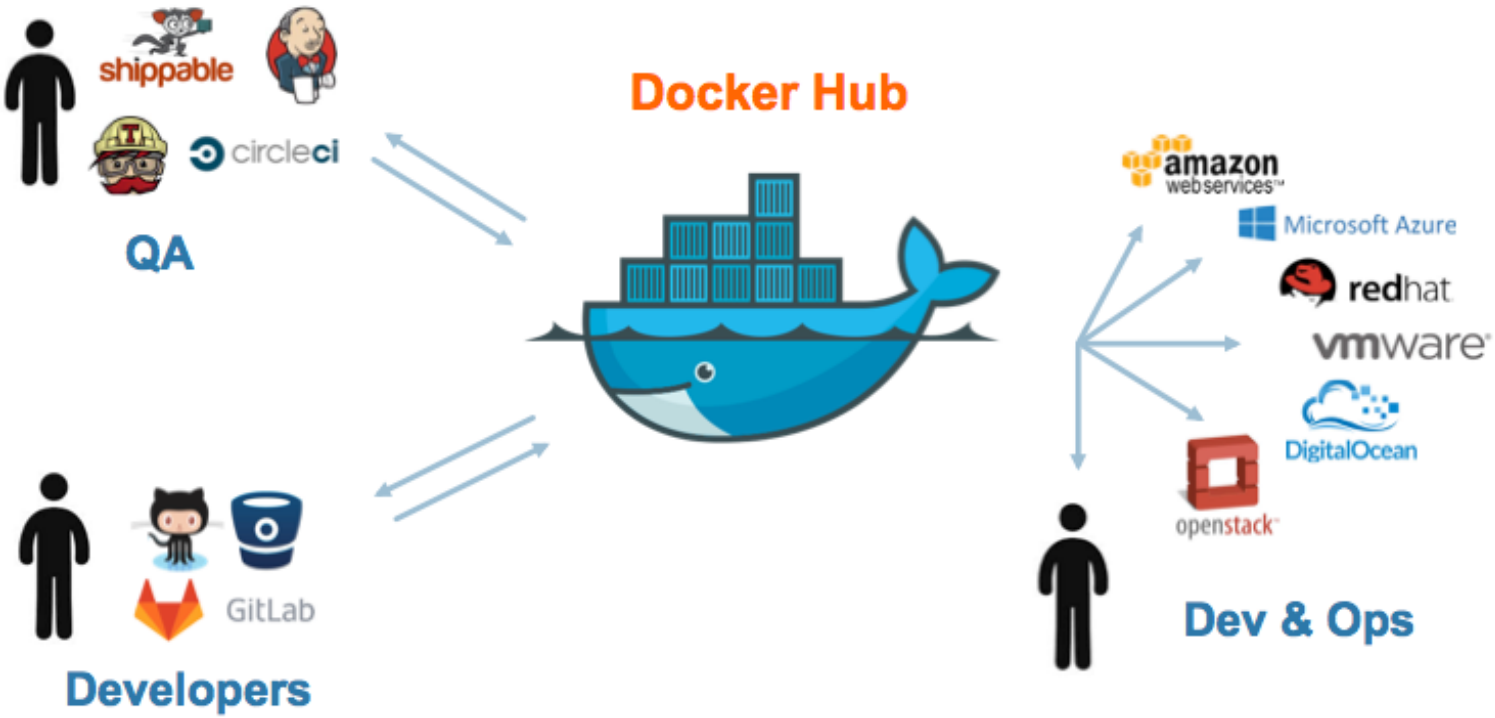
What is Docker?
It's not a VM!





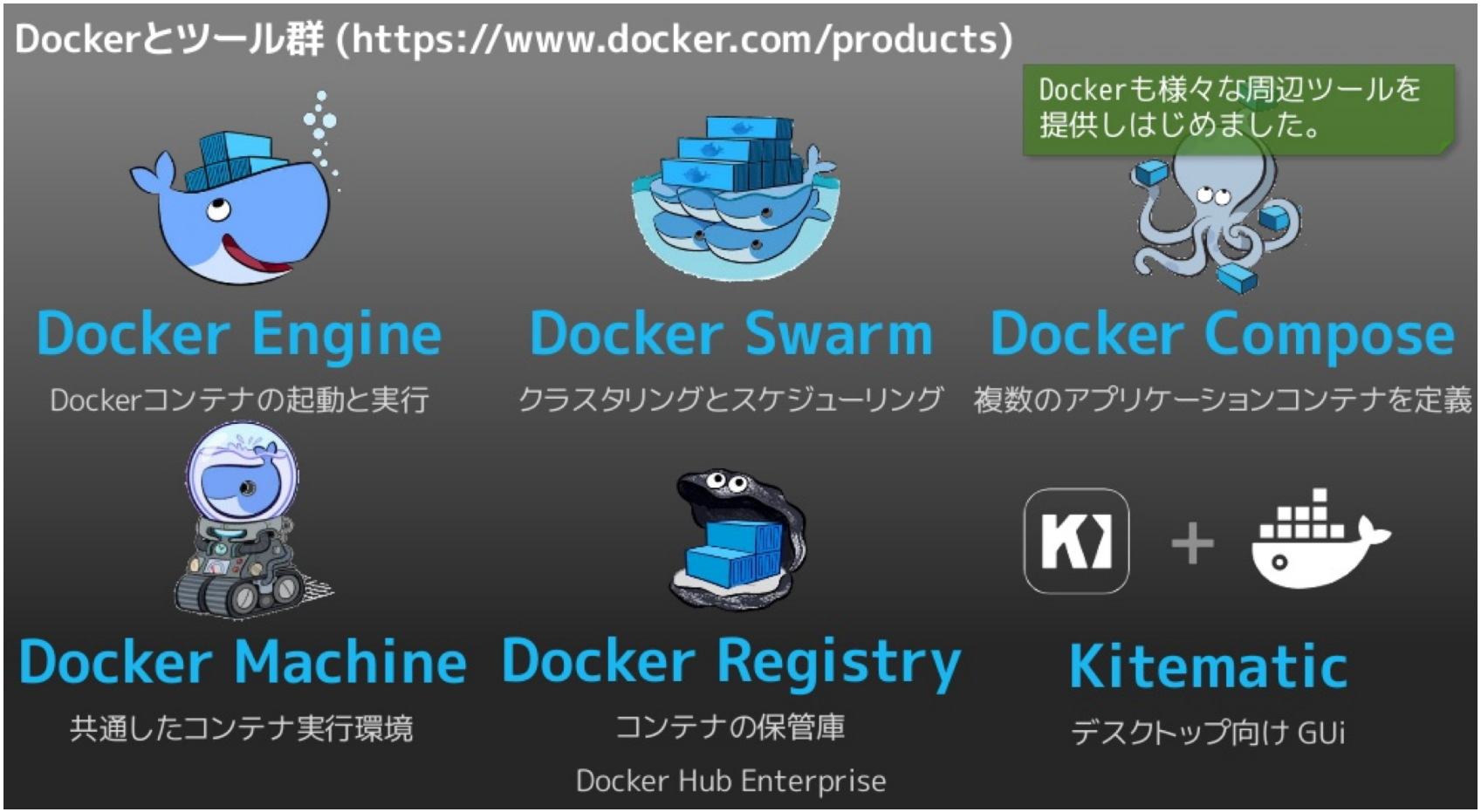
Source: `http://www.jayway.com/2015/03/21/a-not-very-short-introduction-to-docker/`

What is Docker?
It's more than Docker Engine



Source: `https://blog.docker.com/2015/09/docker-hub-2-0/`

What is Docker?
It's a whole ecosytem



Source: `https://gist.github.com/j138/9db77cd23133b72dfbc1`

Installation

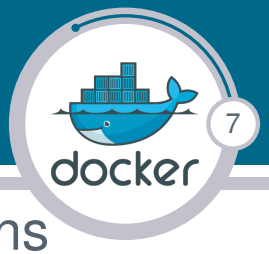
We need for this tutorial to install Docker Engine and Docker Compose. It's really platform specific, so please refer to the docs:

`https://docs.docker.com/engine/installation/`
`https://docs.docker.com/compose/install/`

Non-Linux users should try installing Docker Toolbox, which contains the tools we need plus a nice GUI (Kitematic) and Docker Machine:

`https://www.docker.com/products/docker-toolbox`

Docker images
Using a Dockerfile to build an image



First, let's clone this presentation's repository in GitHub. It contains our example source code:

```
$ git clone git@github.com:allanino/docker-presentation.git
$ cd docker-presentation
```

Take a look at our project's `Dockerfile`:

```
$ cat Dockerfile
FROM gliderlabs/alpine:3.2
MAINTAINER Allan Costa "allaninocencio@yahoo.com.br"
RUN apk --update add python-dev py-pip
ADD . /src/app
WORKDIR /src/app
RUN pip install -r requirements.txt
ENTRYPOINT ["python", "app.py"]
```

Use our Dockerfile to build a Docker image:

```
$ docker build -t allanino/app .
```

To start a container from our image we just need to run this command:

```
$ docker run -p 80:5000 allanino/app
```

The only parameter we passed, using the `-p` flag, was to map container's port 5000 to host's port 80. In this way, the application should be available on `http://localhost`.

WARNING: Don't try to access `http://localhost/counter`.

Docker images
Pushing the image to Docker Hub

We can manually push an image to a registry, but before doing that, you need to login to your account, in this case a Docker Hub account:

```
$ docker login
```

Then push the image to a repository with same name:

```
$ docker push allanino/app
```

After the image is uploaded, anyone can pull it (if it's public) using this command:

```
$ docker pull allanino/app
```

Continuous integration
Automate build

We can integrate source control platforms, such as GitHub or Bitbucket, to have images built automatically on code change.

One platform we use, specially for automatic image tagging based on Git tags, is `quay.io`:

Monolithic servers x Microservices

Orchestration
A multi-container application

What about adding a Redis database?

```
$ docker run --name redis redis
```

Let's give our application access to it:

```
$ docker run --name app --link redis:db -p 80:5000 \
    allanino/app
```

Check container's environment variables:

```
$ docker exec app env
```

Orchestration
Docker Compose: creating a configuration file

Create a `docker-compose.yml` file:

```
version: '2'
services:
  app:
    build: .
    ports:
      - "80:5000"
    volumes:
      - .:/src/app
    links:
      - redis
  redis:
    image: redis
      volumes:
        - redis_data:/data
volumes:
  redis_data:
    driver: local
```

With Compose we can start our entire project with a single command:

```
$ docker-compose up
```

You can scale individual services when running a swarm:

```
$ docker-compose scale app=3
```

Obs: The command above will probably fail for you, because you are probably not running a swarm.

What's next

- **Docker Machine:** provision and manage multiple remote Docker hosts.
- **Docker Swarm:** turns a pool of Docker hosts into a single, virtual Docker host.

Thank you!