Docker 101

Python Girona - October 2018

Jordi Bagot (https://github.com/jbagot), Xavi Torelló (https://github.com/XaviTorello)

What is a Container?

A `container` embeds an application / service and all of its dependencies in a single space.

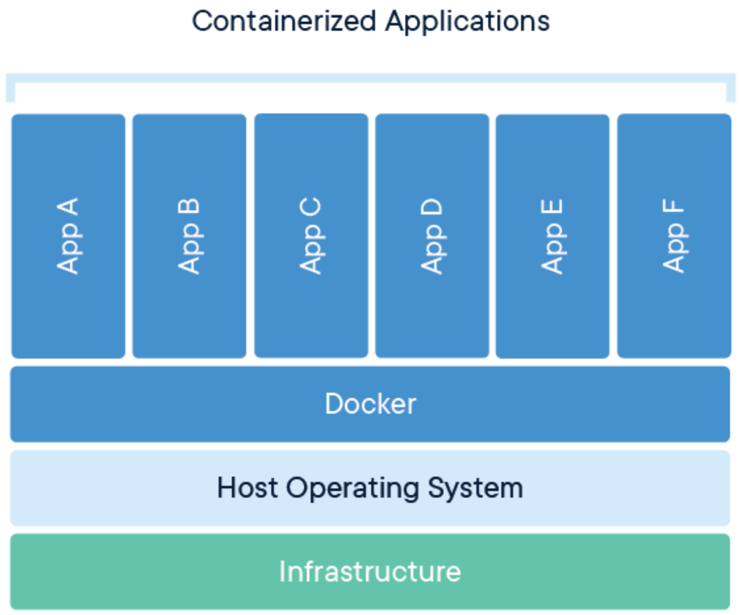
- Isolation
- Ready to run
- Standardization //run everywhere

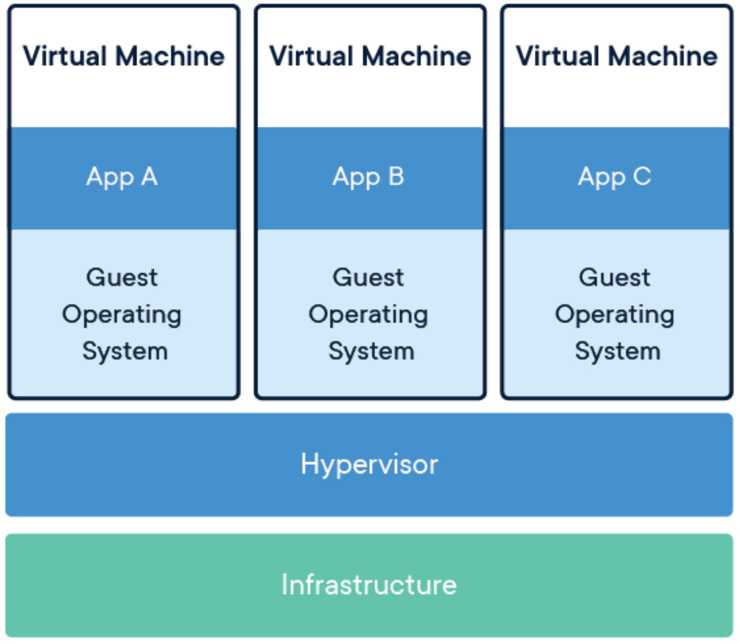
What is Docker?

`Docker` is a platform for developers and sysadmins to develop, ship, and run applications

- `Docker Engine`: open source containerization technology
- `Docker Hub`: SaaS service for sharing and managing app stacks

Docker vs VM?





images from https://www.docker.com/resources/what-container (https://www.docker.com/resources/what-container)

Benefits

- Smaller costs (by default)
- Fast delivery
    - Compatibility
    - Maintainability
    - CI
- Rapid deployment
- Scalability
- Security
- Virtualenvs not needed
- Keep your machine clean

How to use it?

Dockerfile

- Defines the behaviour of the image
- It's like a Makefile that prepares everything
- Also starts the related service

```
FROM alpine:latest
COPY . /app
RUN make /app
CMD python /app/app.py
```

Docker Image

- File
- Comprised of multiple layers //~ snapshot incremental diffs
- Used to execute code in a `Docker container`

```
$ docker images
REPOSITORY                          TAG                 IMAGE ID            CREAT
ED              SIZE
shodand_scanner_scanner             latest              82daf18d5d92        11 da
ys ago          551MB
empireproject/empire                latest              527d5d78e7fc        3 mon
ths ago         1.19GB
redis                               alpine              05097a3a0549        12 da
ys ago          30MB
redis                               2.8                 481995377a04        2 yea
rs ago          186MB
elpaso/qgis-testing-environment     master              334775a61a4f        2 wee
ks ago          3.39GB
docker_erp                          latest              285af92a3352        4 wee
ks ago          1.05GB
ubuntu                              16.04               b9e15a5d1e1a        5 wee
ks ago          115MB
python                              2.7                 4ee4ea2f0113        5 wee
ks ago          908MB
mongo                               3.0                 fdab8031e252        5 mon
ths ago         232MB
```

Running images in containers

- Idempotence?
- Do **not persist** (normally)
- Runs as **`root`** (normally)
- Does **not expose** any container **port** to the host by default
- Does **not map** any host **resource** to the container by default

Running a `python:2.7` image in a temporal container

```
docker run --rm -it python:2.7 python
Python 2.7.15 (default, Sep  5 2018, 04:46:44)
[GCC 6.3.0 20170516] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

, once the execution ends, the container will be autodestroyed.

We can also get a shell for this container:

```
$ docker run --rm -it python:2.7 bash
root@2a57575d1807:/# python -V
Python 2.7.15

root@2a57575d1807:/# ls -la
total 72
drwxr-xr-x   1 root root 4096 Oct 15 10:54 .
drwxr-xr-x   1 root root 4096 Oct 15 10:54 ..
-rwxr-xr-x   1 root root    0 Oct 15 10:54 .dockerenv
drwxr-xr-x   1 root root 4096 Sep  4 22:35 bin
drwxr-xr-x   2 root root 4096 Jun 26 12:03 boot
drwxr-xr-x   5 root root  360 Oct 15 10:54 dev
drwxr-xr-x   1 root root 4096 Oct 15 10:54 etc
drwxr-xr-x   2 root root 4096 Jun 26 12:03 home
drwxr-xr-x   1 root root 4096 Sep  4 22:35 lib
drwxr-xr-x   2 root root 4096 Aug 31 00:00 lib64
drwxr-xr-x   2 root root 4096 Aug 31 00:00 media
drwxr-xr-x   2 root root 4096 Aug 31 00:00 mnt
drwxr-xr-x   2 root root 4096 Aug 31 00:00 opt
dr-xr-xr-x 341 root root    0 Oct 15 10:54 proc
drwx------   1 root root 4096 Sep  5 04:45 root
```

Reviewing existing containers:

```
$ docker ps
CONTAINER ID        IMAGE                  COMMAND                  CREATED
      STATUS              PORTS                  NAMES
7c06579979f3        redis:2.8              "docker-entrypoint.s…"   5 seconds ago
      Up 4 seconds        6379/tcp               quirky_hawking
```

Use `-a` flag to see all (not just the started ones)

```
$ docker ps -a
CONTAINER ID        IMAGE                              COMMAND                  CR
EATED              STATUS                            PORTS                  NAMES
7c06579979f3        redis:2.8                          "docker-entrypoint.s…"   22
 seconds ago       Up 22 seconds                     6379/tcp               quirky_hawkin
g
a22ce5c1986b        docker_erp                         "/entrypoint.sh open…"   22
 hours ago         Exited (0) 20
...
...
---
```

Start a container

```
 $ docker start docker_erp
```

Stop it!

```
 $ docker stop docker_erp
```

Delete it!

```
 $ docker rm docker_erp
```

But I want to communicate with my container!!!!

- **Expose** ports with
    - `- p $HOST_PORT:$CONTAINER_PORT` at run time
        - i.e `-p 8080:8081` to expose the 8080 container port to 8081's host
    - use the `EXPOSE $PORT` command in your `Dockerfile`

- **Mount** paths
    - `- v $HOST_PATH:$CONTAINER_PATH` at run time
        - i.e `-p 8080:8081` to expose the 8080 container port to 8081's host
    - see the difference mount vs `ADD` `Dockerfile` command

But my service needs more than one container...

docker-compose is your friend!

Docker compositions

- run multiple containers at same time
- define your stack using a YAML file

```
version: '3'
services:
  web:
    build: .
    ports:
     - "5000:5000"
  redis:
    image: "redis:alpine"
```

this will provide two containers

- `web`: that uses local `Dockerfile` definition and binds the TCP#5000
- `redis`: that runs an `alpine` tagged `redis` image

- the services can be interconnected with a network
- configure environment variables, different enviroments -> CI

Docker Hub

- repository with thousands of images ready to use

- pull images from `Docker Hub` with Dockerfile
- Create your own repository. Ex: Gitlab registry
- push your images to your repository or to `Docker Hub` with `docker push`
- add a tag in the image name
- version control for your images

```
docker101:latest
docker101:1.0.0
```

Share your images!!! Share your knowledge!!!

Orchestrators

What are orchestrators?

- Help you to keep your containers healthy
- Help you with the deployments and sync the containers
- and much more, we will explain in the next talk ;)

Examples of orchestrators

- Kubernetes
- Docker Swarm

Thank you!

Questions?