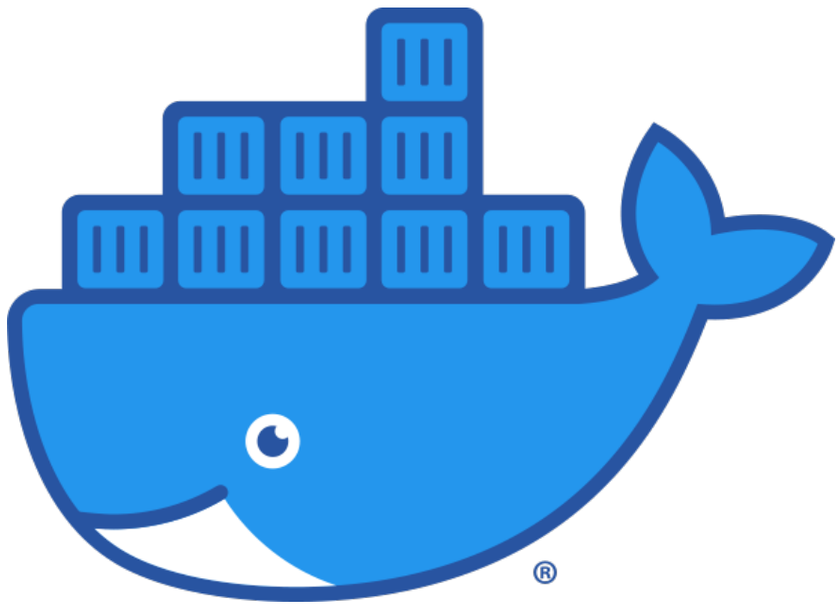


Containerisation with Docker

Aurélien Coet
March 2020

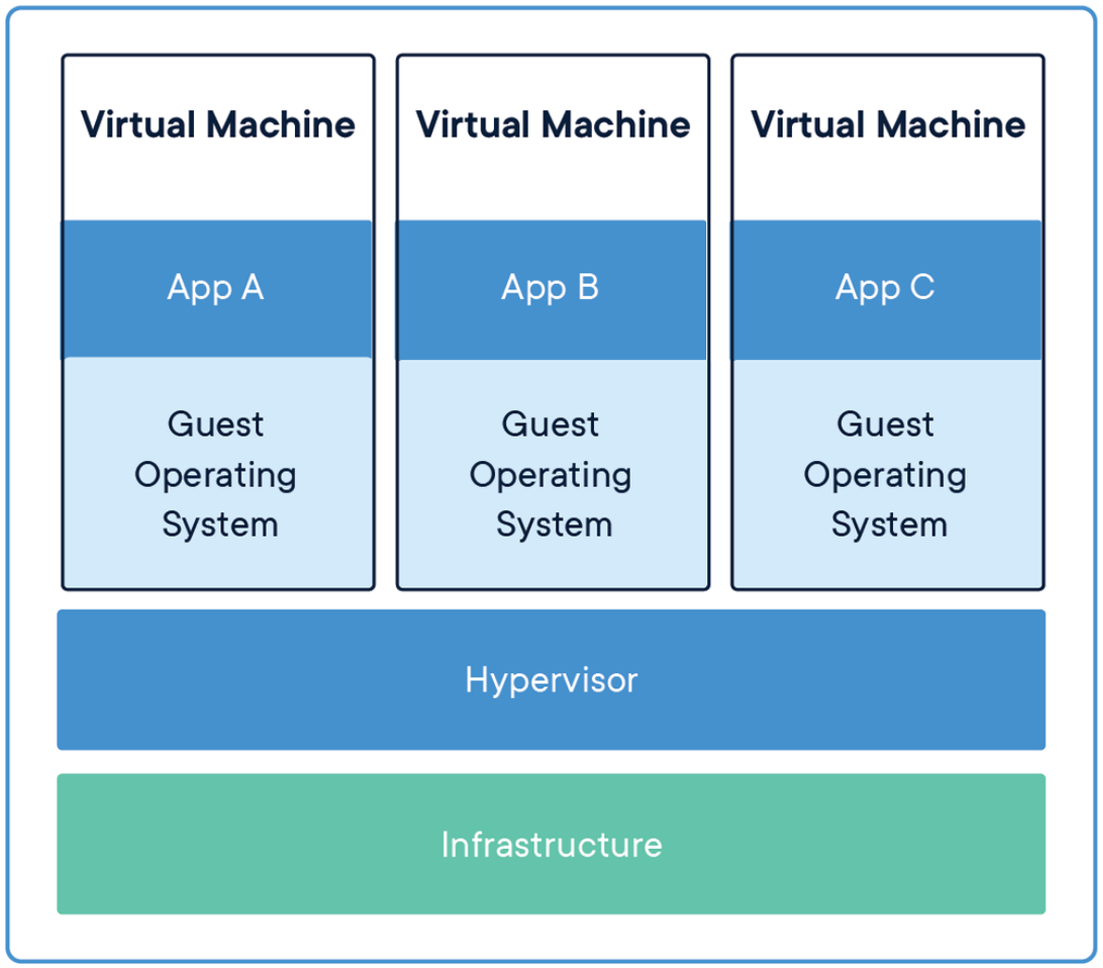
UNIVERSITÉ DE GENÈVE
FACULTÉ DES SCIENCES

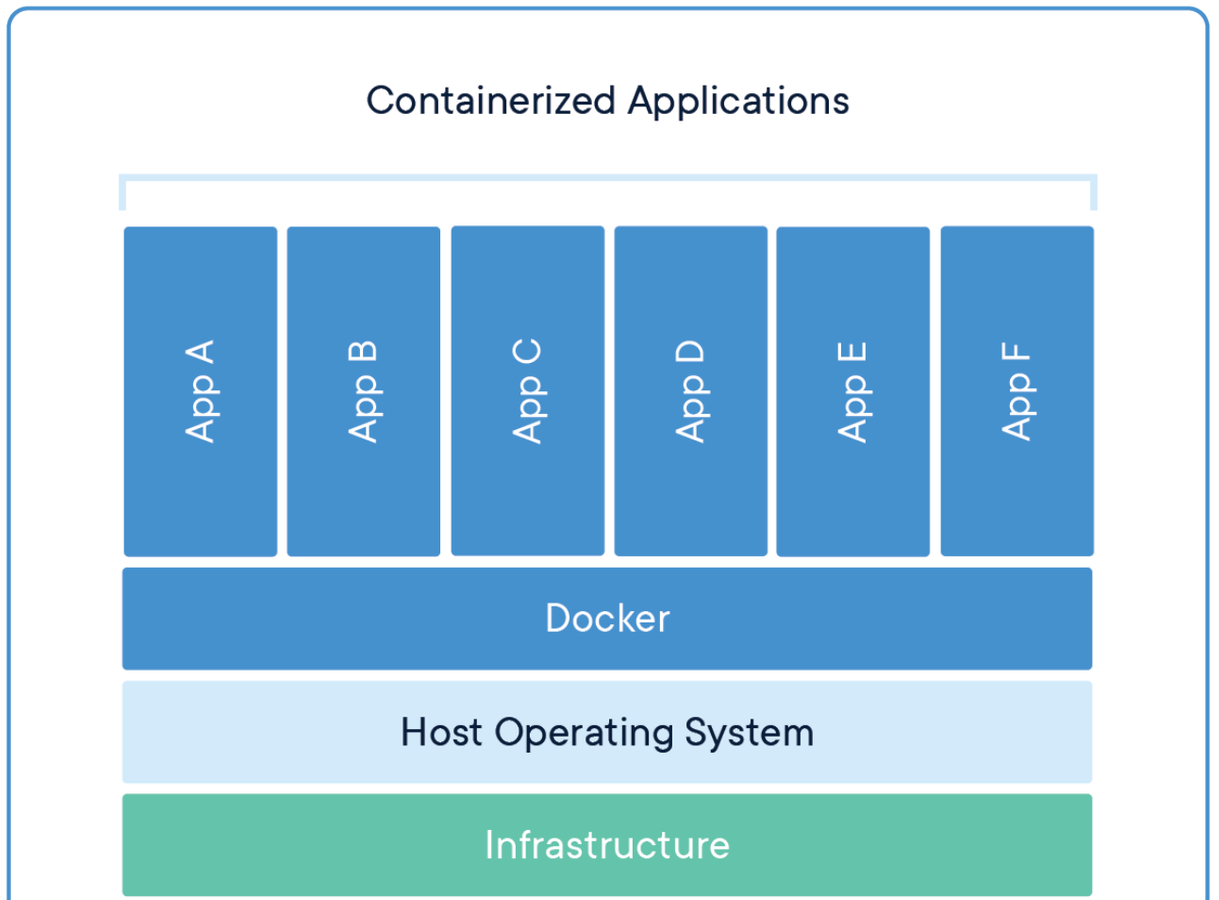
What is *Docker* ?

What is *Docker* ?

- **A tool designed to make software deployment easier by bundling it in *containers***
- **Containers are software units that can be directly run on a machine, without additional installation or configuration steps**
- **They package together configuration files, libraries and programs in isolated environments**

Virtual Machines vs Docker Containers





https://www.docker.com/resources/what-container

Docker Basics

Docker Concepts

- **Docker containers are built from *images***
- **Images define what a container will contain and execute when it is spawned**
- **They are like *templates* for containers: an image is unique, and multiple containers can be spawned from the same image**

Docker Images

- **In the *microservices* example:**

```
unige/regulatory-service      latest    5859668ecfb1      12 seconds ago       778MB
unige/valuation-service       latest    93516633b7b3      48 seconds ago       814MB
unige/instrument-service      latest    b1bded92050c      About a minute ago   814MB
unige/counterparty-service    latest    1789c8543673      2 minutes ago        780MB
unige/api-gateway             latest    b355613b0bbd      32 hours ago         371MB
```

  are docker images containing the microservices and their dependencies (JVM, libraries, etc…)

- ***myCounterpartyService* is a container started from the counterparty-service image**

Docker Containers

- **Docker containers are spawned with the command:**

```
$ docker run -p x:y --name=z img:tag
```

where:

- ***x:y*** **is a mapping from OS port x to container port y**
- ***z*** **is the name given to the container**
- ***img*** **is the name of the image from which the container must be spawned**
- ***tag*** **is the tag (version) of the image**

Docker Containers

- **When docker run is called on an image, the *docker daemon* first searches for it locally on the user's machine**
- **If it doesn't find any image with the input name, it looks for it in a *docker registry***
- **By default, the registry in which docker looks up for images is DockerHub**

Docker Containers

- **Running docker containers are listed with the command:**

```
$ docker ps
```

- **All the containers on a machine can be listed with:**

```
$ docker ps -a
```

Docker Containers

- **Docker containers are stopped with the command:**

```
$ docker kill id
```

where:

  - ***id*** **is the name of the container to stop (or its auto-generated identifier if no name was provided when it was started)**

- **All running docker containers can be stopped with:**

```
$ docker kill $(docker ps -q)
```

Docker Containers

- **Docker containers are deleted with the command:**

```
$ docker rm id
```

where:

- ***id*** **is the name of the container to stop (or its auto-generated identifier if no name was provided when it was started)**

- **All existing docker containers can be stopped with:**

```
$ docker rm $(docker ps -a -q)
```

Dockerfiles

- **Docker images are defined with *Dockerfiles***

```
FROM nginx
COPY ./dist /usr/share/nginx/html
COPY nginx.default /etc/nginx/conf.d/default.conf
COPY htpasswd /etc/nginx/htpasswd
```

https://docs.docker.com/engine/reference/builder/

Dockerfiles - Build stages

- **Dockerfiles are split into *build stages* (or *layers*) that define a set of actions to incrementally build an image**
- **Build stages start from a *base image* defining the environment of the stage**
- **Each stage can use a different base image**
- **The instruction:**

```
FROM nginx AS stage-name
```

  - **Defines *nginx* as the stage's base image**
  - **Names the stage *stage-name***

Dockerfiles - Build stages

```
FROM node:latest AS build-stage
…
```
} 1st layer

```
FROM nginx:latest AS deploy-stage
…
```
} 2nd layer

More on build stages:

https://docs.docker.com/develop/develop-images/multistage-build/

Dockerfiles - Workdirs

- **Build stages operate on *working directories***

- **The instruction:**

```
WORKDIR somedir
```

  - **Defines *somedir* as the stage's workdir**
  - **Creates the directory in the image if it doesn't exist**

Dockerfiles - Copying files

- **The instruction:**

```
COPY ./localdir ./somedir
```

  - **Copies the files in *localdir* on the user's machine to the build stage's *somedir* working directory**

Dockerfiles - Running commands

- **The instruction:**

```
RUN command
```

- **Runs the command *command* inside the image**

.dockerignore

- ***.dockerignore*** **files work similarly to** ***.gitignore*** **files in git**

- **They indicate to the docker daemon that it must ignore some files on the user's machine when building an image**

- **This prevents leaking sensitive or useless data into docker images**

Building a docker image

- A docker image is built with the command:

```
$ docker build -t name:tag path
```

where:

- *name* is the name to give to the image
- *tag* is the tag for the image
- *path* is the path to a Dockerfile

Publishing a docker image

- **A docker image is pushed to the docker registry with the command:**

```
$ docker push name:img
```

where:

- ***name*** **is the name of the image**
- ***tag*** **is the tag of the image**

Listing docker images

- **All docker image on the machine can be listed with the command:**

```
$ docker image ls
```

Deleting a docker image

- A docker image is deleted with the command:

```
$ docker rmi -t name:tag
```

  where:

  - *name* is the name of the image
  - *tag* is the tag for the image

- All existing docker images can be deleted with:

```
$ docker rmi $(docker image ls)
```

- An image cannot be deleted if a container using it is running !

Docker command line reference:

https://docs.docker.com/engine/reference/commandline/docker/

Dockerfile reference:

https://docs.docker.com/engine/reference/builder/