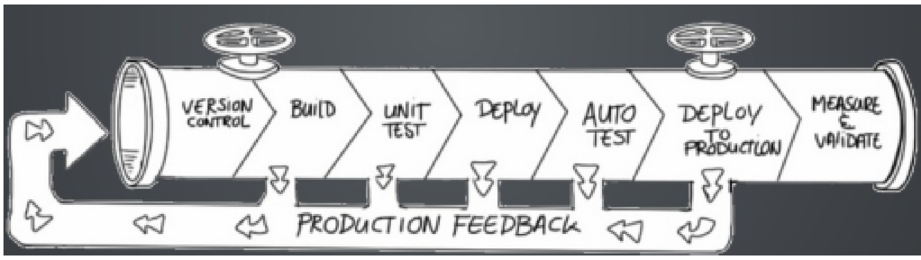
Docker

Boris Teabe

boris.teabedjomgwe@enseeiht.fr

Docker in general

- Virtualization system
  - Allow building very light VMs (containers)
  - OS level virtualization
  - Very small VMs and small overhead
- Set of user-friendly tools for managing containers
  - Much used for continuous integration
  - No live migration
- Widely used
  - Versions for Linux, Mac, Windows
  - Opensource

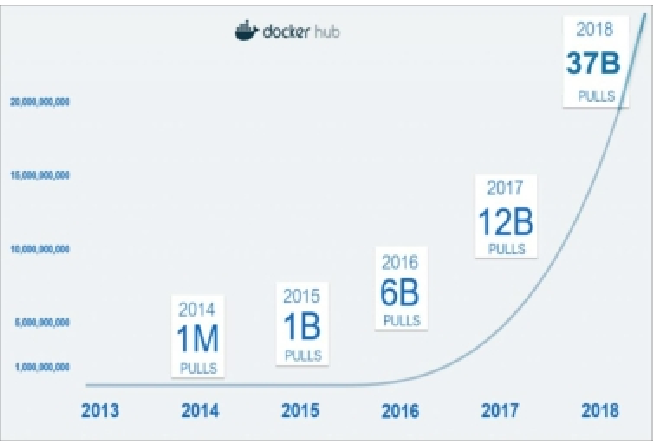


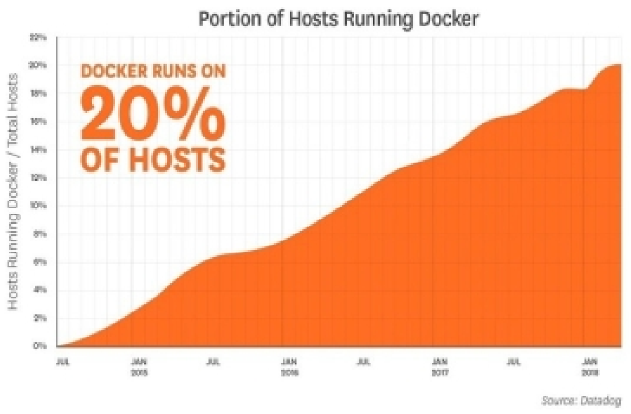
Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers.

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels.

Some numbers

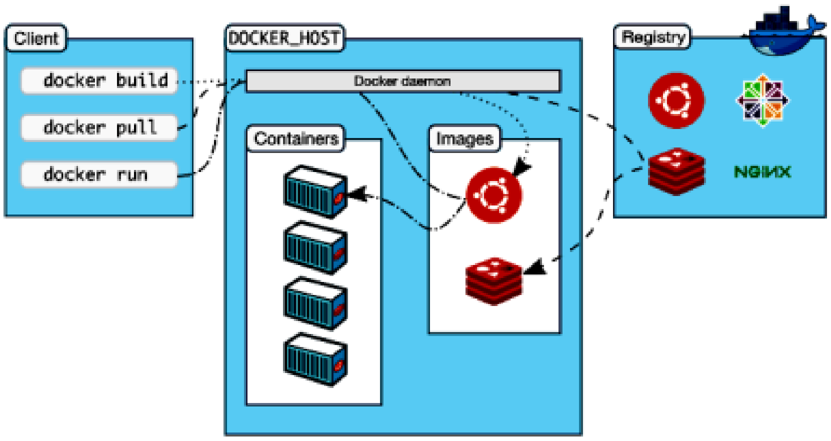




GROUPON grubHub PayPal Nerdalize
ORBITZ BUSINESS INSIDER UBER lyft
ebay Yandex Baidu 百度 shopify
Spotify yelp New Relic. BBC NEWS
ING bleacher report GILT

At the beginning of April 2018, 23.4 percent of Datadog customers had adopted Docker, up from 20.3 percent one year earlier. Since 2015, the share of customers running Docker has grown at a rate of about 3 to 5 points per year.

Architecture



- Client-server architecture
  - Registry
    - Server of VM images (Internet site)
  - Docker client (a shell)
  - Docker host (docker daemon)
    - The heart of the system
      - Building of VM images
      - Instance creation
    - A local image registry (cache)

Docker uses a client-server architecture. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface.

The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes.

The Docker client (docker) is the primary way that many Docker users interact with Docker. When you use commands such as docker run, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API. The Docker client can communicate with more than one daemon.

A Docker registry stores Docker images. Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry.

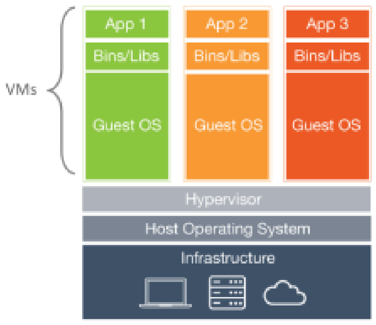
Docker images

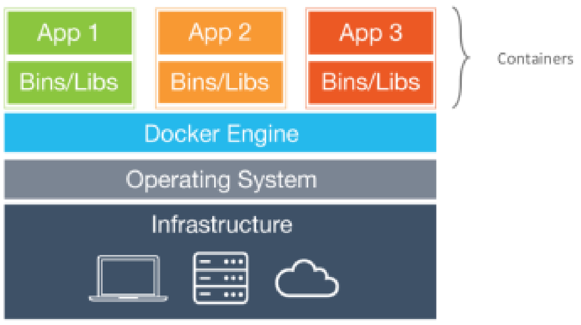
- The image of a VM
  - Docker relies on Union File System for the representation of image
    - An image is represented as a set of layers
    - Each layer describes a modification of the file system (like diff)
  - Advantages of this representation
    - Allows building a file system
      - From a standard image
      - With small additional data (tens of Mb instead of hundreds of Mb)
      - Efficiently
    - The same set of standard images can be reused
    - The modification of a file system does not generate a full file system (only a layer)
      - Only diffs are saved
      - A means for versioning
- Docker allows sharing images
    - https://hub.docker.com

An image is a read-only template with instructions for creating a Docker container. Often, an image is based on another image, with some additional customization. For example, you may build an image which is based on the ubuntu image, but installs the Apache web server and your application, as well as the configuration details needed to make your application run.

You might create your own images or you might only use those created by others and published in a registry. To build your own image, you create a Dockerfile with a simple syntax for defining the steps needed to create the image and run it. Each instruction in a Dockerfile creates a layer in the image. When you change the Dockerfile and rebuild the image, only those layers which have changed are rebuilt. This is part of what makes images so lightweight, small, and fast, when compared to other virtualization technologies.

Virtual Machines vs. Containers



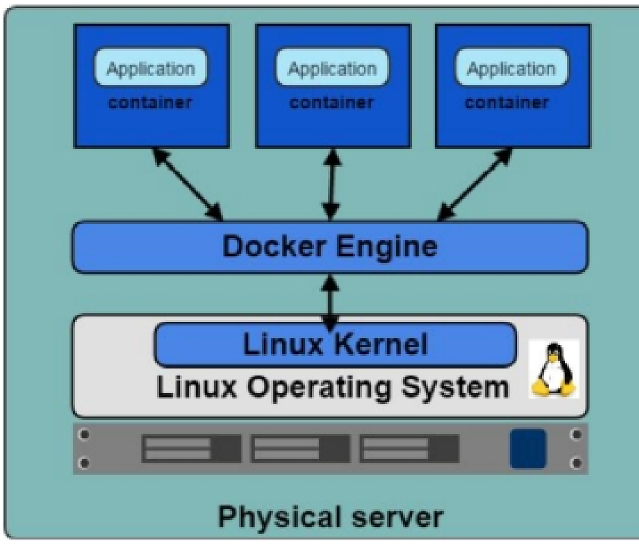


- **Virtual machines**
  - Each virtual machine (VM) includes the app, the necessary binaries and libraries and an entire guest operating system

- **Containers**
  - Containers include the app & all of its dependencies, but share the kernel with other containers.
  - Run as an isolated process in userspaceon the hostOS
  - Not tied to any specific infrastructure–containers run on any computer, infrastructure and cloud.

Docker Engine

- Container execution and admin
- Uses Linux Kernel namespaces and control groups
- Namespaces provide for isolated workspace

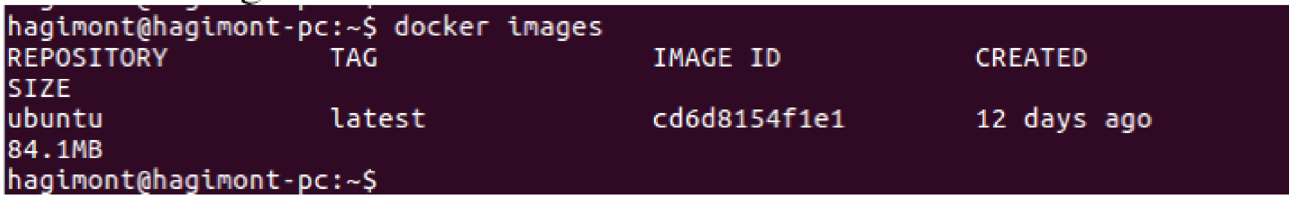


First steps

- Installation under Linux
  - wget -qO- https://get.docker.com/ | sh
- Starting a container
  - docker $\underset{\uparrow\,\text{start}}{\text{run}}$ $\underset{\uparrow\,\text{flags}}{\text{-it}}$ $\underset{\uparrow\,\text{image}}{\text{ubuntu}}$ $\underset{\uparrow\,\text{application}}{\text{bash}}$
  - Lookup the image
    - If the image is not in the local registry, download from the hub
    - Ubuntu: pre-existing image in the hub
  - Build the Linux file system
  - Start the container
  - Configure the IP address of the container
    - Also communication between outside and the container

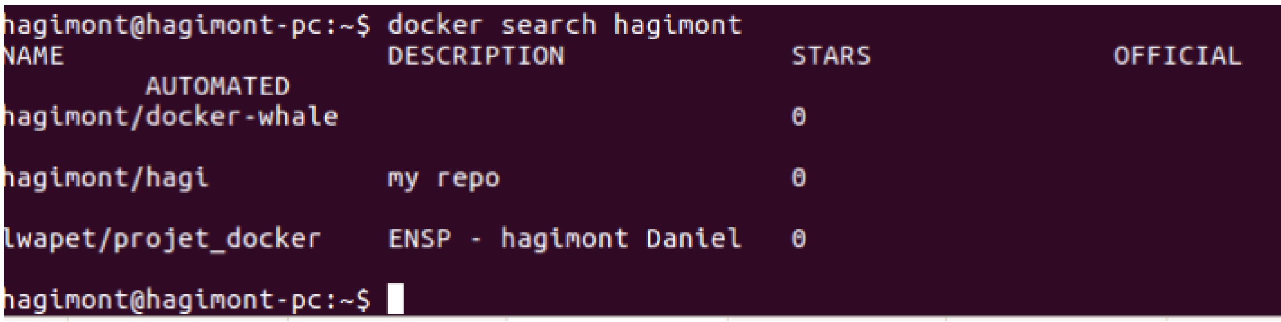
Management of images

- List local images
  - docker images



```
hagimont@hagimont-pc:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
ubuntu              latest              cd6d8154f1e1        12 days ago
84.1MB
hagimont@hagimont-pc:~$
```

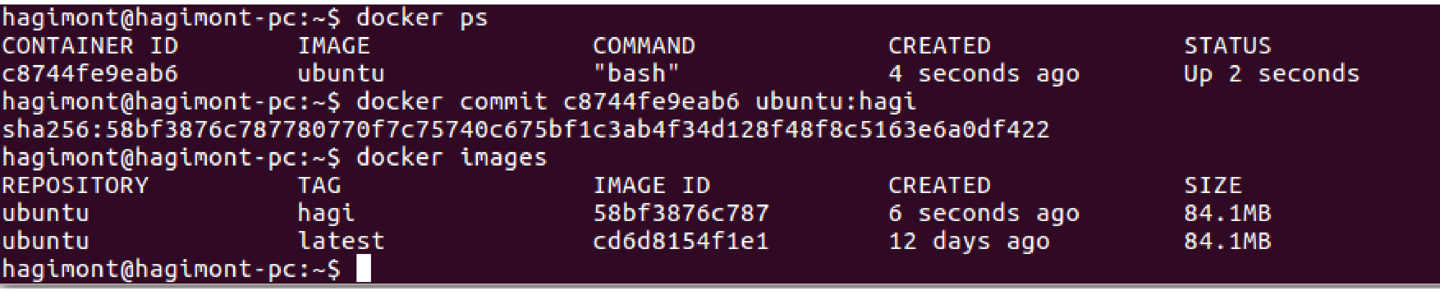
- Log in the hub
  - docker login/logout
- Lookup an image in the hub
  - docker search hagimont



```
hagimont@hagimont-pc:~$ docker search hagimont
NAME                   DESCRIPTION              STARS               OFFICIAL
         AUTOMATED
hagimont/docker-whale                           0

hagimont/hagi          my repo                  0

lwapet/projet_docker   ENSP - hagimont Daniel   0

hagimont@hagimont-pc:~$
```

Management of images

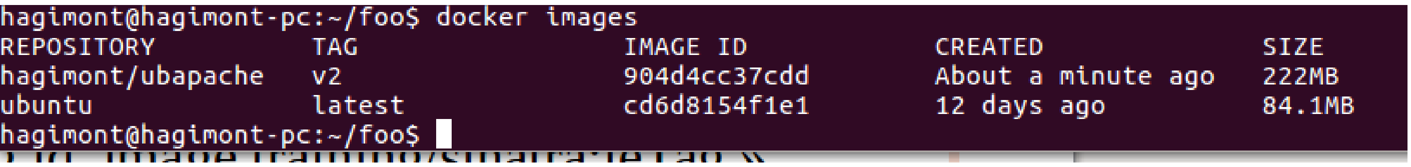
- Creation of an image
- From a container instance
  - Start the container (from an initial standard image)
  - Modify the file system (apt-get install ...)
  - Commit the instance with a new image name
    - docker commit c8744fe9eab6 ubuntu:hagi



```
hagimont@hagimont-pc:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
c8744fe9eab6        ubuntu              "bash"              4 seconds ago       Up 2 seconds
hagimont@hagimont-pc:~$ docker commit c8744fe9eab6 ubuntu:hagi
sha256:58bf3876c787780770f7c75740c675bf1c3ab4f34d128f48f8c5163e6a0df422
hagimont@hagimont-pc:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              hagi                58bf3876c787        6 seconds ago       84.1MB
ubuntu              latest              cd6d8154f1e1        12 days ago         84.1MB
hagimont@hagimont-pc:~$ 
```

Management of images

- Creation of an image
- From a Dockerfile
  - mkdir foo
  - cd foo
  - Create a file Dockerfile
    - # This is a comment
    - FROM ubuntu
    - RUN apt-get update && apt-get install -y apache2
  - docker build -t hagimont/ubapache:v2 .



```
hagimont@hagimont-pc:~/foo$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED              SIZE
hagimont/ubapache   v2                  904d4cc37cdd        About a minute ago   222MB
ubuntu              latest              cd6d8154f1e1        12 days ago          84.1MB
hagimont@hagimont-pc:~/foo$ 
```

Management of images

- Management of images in the hub
  - You must be logged in
  - Save the image in the hub
    - docker push hagimont/ubapache:v2
  - Download an image from the hub
    - docker pull hagimont/ubapache:v2
- Tag an image (versioning)
  - docker tag id_image training/sinatra:thetag

Data volumes

- Goal of data volumes
  - make visible in one or more containers a directory or file from the host file system
  - Allows file sharing between several containers
- Persistent even after container destruction
- Any modification is immediately effective
- Command:
  - `docker run -it -v /tmp/host_file:/tmp/container_file ubuntu bash`

Management of containers

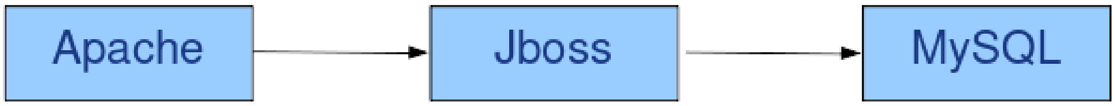
- It’s a VM in the Docker dialect
  - Philosophically, execute a single process
    - One container = one application (or process)
    - No execution of daemons, services, ssh, etc.
- it’s file system is not persistent (after container destruction
- Docker implements its own container format
  - Libcontainer (instead of Linux’LXC which is way more complex)
  - Advantage
    - Enables portability to other OS and also other implementations

Management of containers

- Start a container
  - docker run
- List containers
  - docker ps
- Stop (clean) a container
  - docker stop
    - SIGTERM followed by a SIGKILL
- Stop (force) a container
  - docker kill
    - SIGKILL
- Restart a previously stopped container
  - docker start
- Remove a container
  - docker rm
- Help
  - docker help

Linking containers

- Docker help linking container
- Consider a JEE application structured as follows



- Apache requires Jboss' IP address
- Jboss requires MySQL' IP address

Linking containers

- Links between containers
  - docker run -d --name db hagimont/mysql
  - docker run -d --name jboss --link db hagimont/jboss
    - The db host name is known in the Jboss container
  - docker run -d --name apache --link jboss hagimont/apache
    - The jboss host name is known in the apache container
- Better method
  - Define a network (bridge)
    - docker network create mynet
  - Start a container in this network
    - docker run -d --name db --net mynet hagimont/mysql
  - The db host name is known in other containers in mynet

Linking containers

- Port redirection
  - Example of link: host → container
    - docker run -d -p 80:5000 hagimont/apache
    - Any connection on port 80 of the host is forwaded to port 5000 of the container

Ecosystem

- Docker machine
  - Allow to easily install Docker hosts in a network
- Docker compose
  - Allow defining and running multi-container applications
- Kitematic
  - Graphical interface for the administration of a Docker host
- Docker swarm
  - Allow the management of a cluster of Docker hosts (container replication, load-balancer, elasticity, recovery …)

Docker compose

- The docker cli is used when managing individual containers on a docker engine.
- The docker-compose cli can be used to manage a multi-container application.
- It works as a front end "script" on top of the same docker apiused by docker.

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application’s services. Then, with a single command, you create and start all the services from your configuration.

What is Docker Compose?

- Define and run multi-container applications
- Specify images and configuration in a simple YAML file
  - *docker-compose.yml*
- One command to get it all running:
  - *$ docker-compose up*

Exemple of yml file :

```
version: "3.8"
services:
  web:
    build: .
    ports:
      - "5000:5000"
    links:
      - redis
  redis:
    image: redis
```

What is Docker Compose?

docker-compose up:

- Builds images from Dockerfiles
- Pulls images from registries
- Creates and starts containers
- Streams their logs

Builds, (re)creates, starts, and attaches to containers for a service.

What is Docker Compose?

Make your development environments:

- Repeatable
- Isolated
- Fast

Docker Compose File

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

Web service

The web service uses an image that's built from the Dockerfile in the current directory.

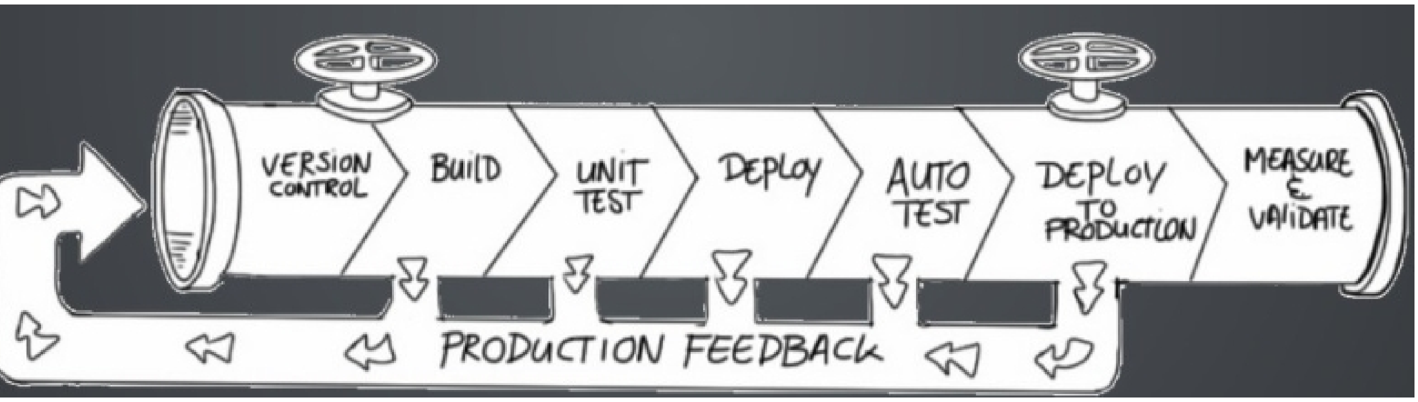
Redis service

The redis service uses a public Redis image pulled from the Docker Hub registry.

docker-compose up:

Usecase: continuous integration

- Docker is widely used for continuous integration
  - Quick transition from code to production



Continuous integration (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project.

Usecase: continuous integration

- Continuous integration: software engineering techniques which aim at accelerating the delivery of software by reducing integration time
  - Code verification and compiling
  - Execution of unit tests
  - Delivery of a version to test (including the most recent modifications)
  - Possibility to automatically generate periodic reports about the code quality, test coverage, etc.
- Some tools: Anthill Pro., Atlassian Bamboo, Build Forge, Cruise Control, Apache Continuum, Luntbuild, JetBrains TeamCit, Jenkins

Continuous integration with Docker

- A Docker image captures dependencies (libraries, other software …) of software to be executed in a container
- Such images/containes are used for
  - Compiling
  - Verifications
  - Testing
  - Deploying
  - Delivery