Container technology is not very old

The most famous : docker

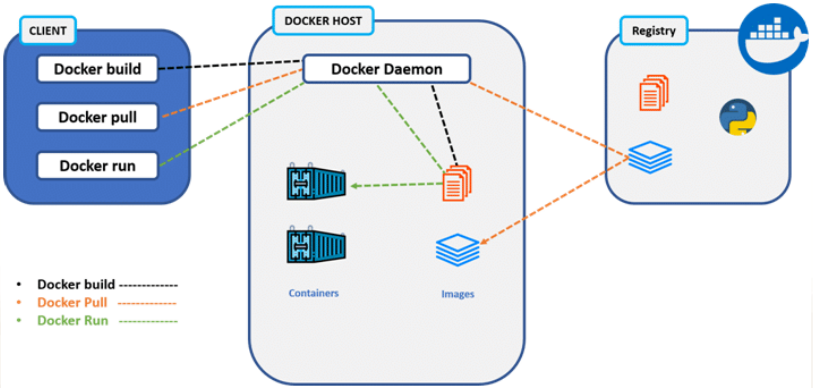
Solomon Hykes was inspirated by container port in the world travel

Docker is an open source project, a community and a private company

- Born in 2010
- First public release in 2013
- V 1.0 in 2014
- Open source and free
- Packaged to Ubuntu in 2014 (V14.04)

Term definitions

- Docker image –> "snapshot" immutable file
  - Set of libraries, functions
  - Static state
  - Online Store or share
  - Automatically build
- Docker container –> instance of an image
  - Result of the image activation
  - Can be modified
  - Can be tunred into an image
  - 1 image –> multiple containers


CLIENT
Docker build
Docker pull
Docker run
DOCKER HOST
Docker Daemon
Containers
Images
Registry
Docker build ----------
Docker Pull ----------
Docker Run ----------

client-server architecture

1. Client talk to a daemon (docker background program)

Client

```
$ docker build [path][url]
docker build https://github.com/docker/rootfs.git#container:docker
$ docker pull [image\_name]
 docker pull biocontainers/samtools
$ docker run [image\_name]
docker run biocontainers/samtools
```

Docker Cheat Sheet

docker

Build

Build an image from the Dockerfile in the current directory and tag the image
`docker build -t myimage:1.0 .`

List all images that are locally stored with the Docker Engine
`docker image ls`

Delete an image from the local image store
`docker image rm alpine:3.4`

Share

Pull an image from a registry
`docker pull myimage:1.0`

Retag a local image with a new image name and tag
`docker tag myimage:1.0 myrepo/myimage:2.0`

Push an image to a registry
`docker push myrepo/myimage:2.0`

Run

Run a container from the Alpine version 3.9 image, name the running container "web" and expose port 5000 externally, mapped to port 80 inside the container.
`docker container run --name web -p 5000:80 alpine:3.9`

Stop a running container through SIGTERM
`docker container stop web`

Stop a running container through SIGKILL
`docker container kill web`

List the networks
`docker network ls`

List the running containers (add `--all` to include stopped containers)
`docker container ls`

Delete all running and stopped containers
`docker container rm -f $(docker ps -aq)`

Print the last 100 lines of a container's logs
`docker container logs --tail 100 web`

Docker Management

All commands below are called as options to the base `docker` command. Run `docker <command> --help` for more information on a particular command.

app*	*Docker Application*
assemble*	*Framework-aware builds (Docker Enterprise)*
builder	*Manage builds*
cluster	*Manage Docker clusters (Docker Enterprise)*
config	*Manage Docker configs*
context	*Manage contexts*
engine	*Manage the docker Engine*
image	*Manage images*
network	*Manage networks*
node	*Manage Swarm nodes*
plugin	*Manage plugins*
registry*	*Manage Docker registries*
secret	*Manage Docker secrets*
service	*Manage services*
stack	*Manage Docker stacks*
swarm	*Manage swarm*
system	*Manage Docker*
template*	*Quickly scaffold services (Docker Enterprise)*
trust	*Manage trust on Docker images*
volume	*Manage volumes*

*Experimental in Docker Enterprise 3.0

www.docker.com

DOCKER COMPOSE CHEAT SHEET

File

structure

```
services:
  container1:
    properties: values

  container2:
    properties: values

networks:
  network:

volumes:
  volume:
```

Types

value
```
key: value
```

array
```
key:
  - value
  - value
```

dictionary
```
master:
  key: value
  key: value
```

Properties

build
build image from dockerfile in specified directory
```
container:
  build: ./path
  image: image-name
```

image
use specified image
```
image: image-name
```

container_name
define container name to access it later
```
container_name: name
```

volumes
define container volumes to persist data
```
volumes:
  - /path:/path
```

command
override start command for the container
```
command: execute
```

environment
define env variables for the container
```
environment:
  KEY: VALUE
---
environment:
  - KEY=VALUE
```

env_file
define a env file for the container to set and override env variables
```
env_file: .env
---
env_file:
  - .env
```

restart
define restart rule (no, always, on-failure, unless-stopped)
```
expose:
  - "9999"
```

networks
define all networks for the container
```
networks:
  - network-name
```

ports
define ports to expose to other containers and host
```
ports:
  - "9999:9999"
```

expose
define ports to expose only to other containers
```
expose:
  - "9999"
```

network_mode
define network driver (bridge, host, none, etc.)
```
network_mode: host
```

depends_on
define build, start and stop order of container
```
depends_on:
  - container-name
```

Other

idle container
send container to idle state + container will not stop
```
command: tail -f /dev/null
```

named volumes
create volumes that can be used in the volumes property
```
services:
  container:
    image: image-name
    volumes:
      - data:
volume:/path/to/dir

volumes:
  data-volume:
```

networks
create networks that can be used in the networks property
```
networks:
  frontend:
    driver: bridge
```