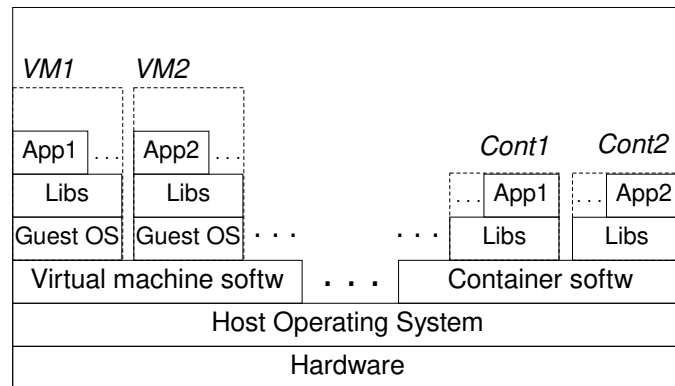# *DOCKER*

Essentials

Virtual machines vs containers

- virtual machines
  - provide isolation (when necessary)
- container:
  - provide isolation (when necessary) in a small, efficient scale

Virtual machine model   *Container model*

| VM1	VM2			Cont1	Cont2
App1 ...	App2 ...			... App1	... App2
Libs	Libs			Libs	Libs
Guest OS	Guest OS ···		···		
Virtual machine softw		**· · ·**	Container softw		
Host Operating System					
Hardware					

Computer

Docker[1]

Docker's vision

- developer writes Program in *Development environment* and creates an "*image*" of it, with all needed software

- other developers and users *build container* with Program's *image* (*Testing* environment) and use, test and change Program in that environment

- feedback to developers cause alterations of Program in the *Development* environment and new *image* is created and "pushed" to Testing environment

- when all testing is complete, final *image* of Program is created, ready to be used (or "pushed"?) to customer in Production environment

---

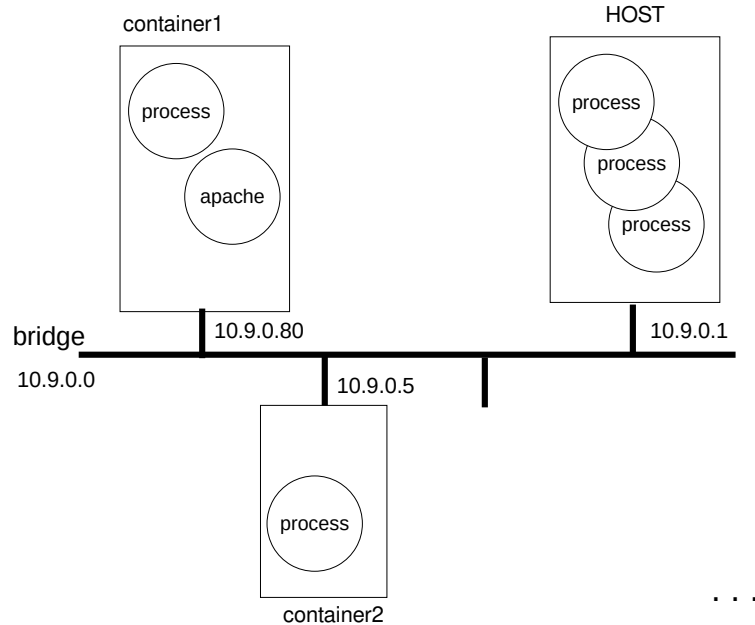1   www.docker.com ; www.docker.com/community/open-source

---

Docker's usage

- programs (utilities, daemons,...), objects (containers, images, networks,...) support environment and operate as specified by developer and user

in docs.docker.com/get-started/overview/#docker-architecture
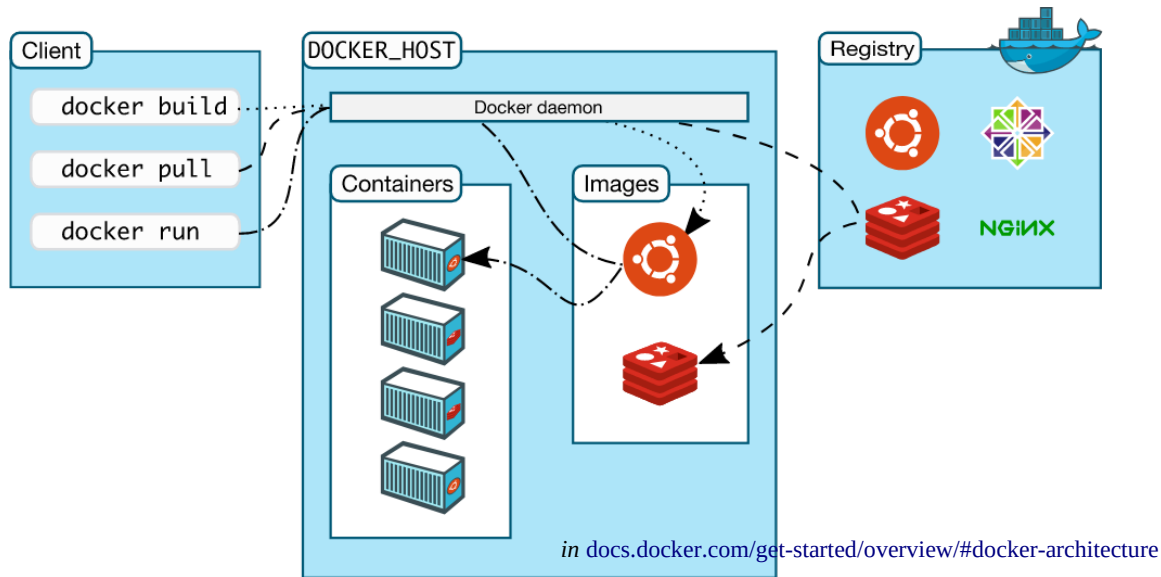
- processes (almost entire operating systems) run in containers, isolated from other containers, but being able to inter-communicate

container1

process

apache

HOST

process

process

process

bridge 10.9.0.80 10.9.0.1

10.9.0.0 10.9.0.5

process

container2

. . .

- developer specifies:

  - Dockerfile

    - instructions for building a container

Dockerfile

```
FROM handsonsecurity/seed-server:apache-php

COPY bash_shellshock /bin/
COPY vul.cgi getenv.cgi /usr/lib/cgi-bin/
COPY server_name.conf  /etc/apache2/sites-available

RUN  chmod 755 /bin/bash_shellshock \
     && chmod 755 /usr/lib/cgi-bin/*.cgi  \
     && a2ensite server_name.conf

CMD service apache2 start && tail -f /dev/null
```

. . . *Docker's usage (cont.): developer specifies*

- o `docker-compose.yml`

  - instructions for building a number of interconnected containers (each with its own *Dockerfile*)

docker-compose.yml

```
version: "3"
services:
    victim:
        build: ./image_www
        image: seed-image-www-shellshock
        container_name: victim-10.9.0.80
        tty: true
        networks:
            net-10.9.0.0:
                ipv4_address: 10.9.0.80
networks:
    net-10.9.0.0:
#        name: net-10.9.0.0
        ipam:
            config:
                - subnet: 10.9.0.0/24
```

...Docker's usage (cont.)

- user useful commands:

  - `docker-compose up`
    - builds, (re)creates, starts and attaches to containers images

  - `docker-compose down`
    - stops containers and removes containers, networks, volumes, and images created by `docker-compose up`

  - `docker ps`
    - lists containers

  - `docker exec -it <id> cmd`
    - Run a command in a running container. `cmd` example: `/bin/bash`

  - `docker network ls`
    - lists networks

  - (`ifconfig` : displays the status of the currently active interfaces.)