Hack the whale

DIETI UNINA

NAPOLI GNU/LINUX USERS GROUP

Tips and Tricks to have fun with services like containers

A new IT paradigm

let's talk about

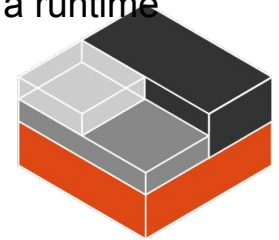
- **why?**
  New IT professionals must face and take an interest about full microservices lifecycle combined with continuous deployment and containers.

- **who?**
  Architects: that want to know how to design their system around microservices;
  Operators: that want know how to apply configuration management practices in containers;
  Developers: that want take the process back into their hands!
  Manager: that gain a better understanding of project and of it delivery.

- **what?**
  ***aka* toolbox**: containers and orchestration tools

Short happy story

Historically, UNIX-style operating system have used the term ***jail*** to describe a modified runtime environment for a program that have protected resources.
Only since 2005, thanks to **Sun Solaris 10**, ***container*** has become the preferred term for such a runtime environment.



Something like this before was seen only with ***chroot*** ... but it was not enough!

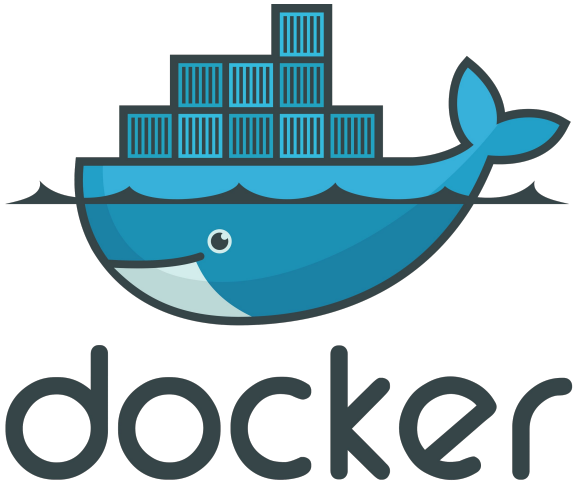
Since 2008, an interesting technology has appeared: **LXC**
LXC is a method to running multiple isolated Linux system (containers, right!) on a control host using a single Linux kernel.
One of the key features is the isolation of the namespaces that allows you to isolate groups of processes between them.

Short happy story

Using containers has been a best practice for a long time. But manually building containers can be challenging and easy to do incorrectly.
These errors and the sense of security must be avoided.

Docker solves this problem!

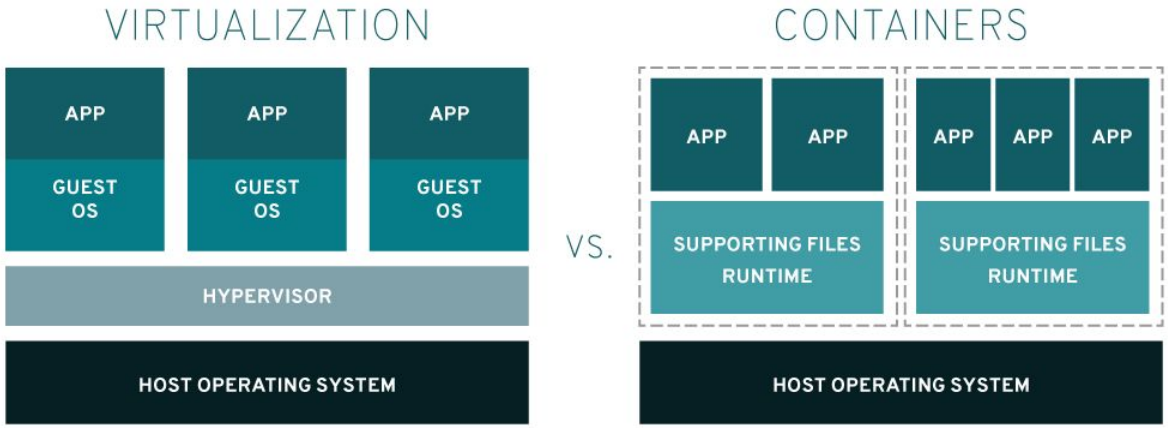
Any software run with Docker is run inside a container. Docker uses existing container engines to provide consistent containers build according to best practices.



Containers are not virtualization

Without container technology (*in particular a cool technology like Docker*), businesses typically use virtualization to provide isolation.
Virtual machines **take a long time** to create and require significant resource overhead because they run a whole copy of an operating system in addition to the software you want to use.

Unlike VM, **programs running inside Docker containers interface directly with the host's Linux kernel**. There isn't additional layer between the program running inside the container and the computer's operating system, no resources are wasted by running redundant software or simulating virtual hardware.

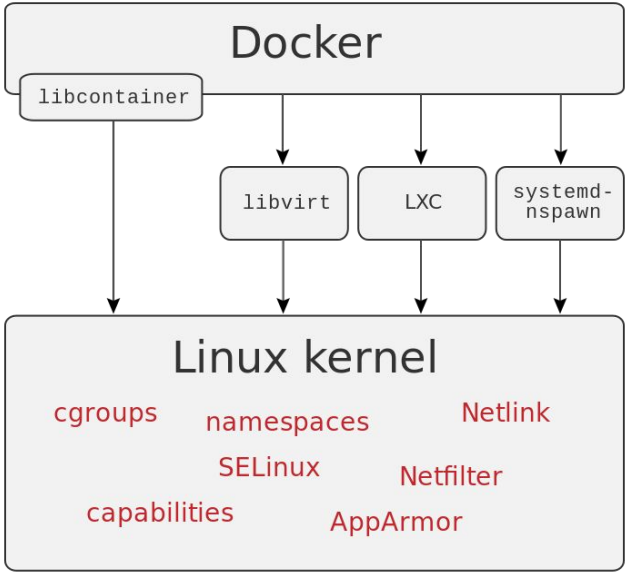


Welcome to Docker world

Recapping

- **What is Docker?**

Docker is a whale … ops … docker is a CL program, a background daemon and a set of remote services that take a logical approach to solving common software problems and simplifying your experience installing, running, publishing and removing software.
It accomplishes this using a UNIX technology called containers.



What problem does Docker solve?

- **Organized software park:**

  Using software is complex!
  Before installation you have to consider what operating system you are using, the resources the software requires, what other software is already installed and what other software depends on.

- **Improving portability:**

  The application is agnostic! It is executable on any operating system!

- **Protecting your infrastructure:**

  Isolation is the key! Anything inside a container can only access things that are inside it as well.

Why is Docker important?

- **1**
  Docker provides what is called an **abstraction**.
  Abstractions allow you to work with complicated things in simplified terms.

- **2**
  Docker is **significant push** in the software community to adopt container technology. This push is so strong in companies like Amazon, Microsoft and Google. They contribute to its development and adopt it in their own cloud offering.

- **3**
  It has made software installation, compartmentalization and removal **very simple**. Better yet, Docker does it in a cross-platform and open way.

- **4**
  We are finally starting to see better adoption of some of the more advanced **isolation** features of operating system.

Keywords of Docker world: Dockerfile

Docker can build images (*later I will explain what an image is*) automatically by reading the instructions from a **Dockerfile**.

A Dockerfile is a **text document** that contains all the commands a user could call on the command line to assemble an image.
It is composed of simple commands like:

`FROM` #indicates the source of docker image
`RUN` #execute any commands in a new layer on top of the current image and commit the results
`COPY` #copies new files or directories from a src and adds them to the filesystem of the container at the destination path
`EXPOSE` #informs Docker that the container listens on the specified network ports at runtime
`VOLUME` #creates a mount point with the specified name
...

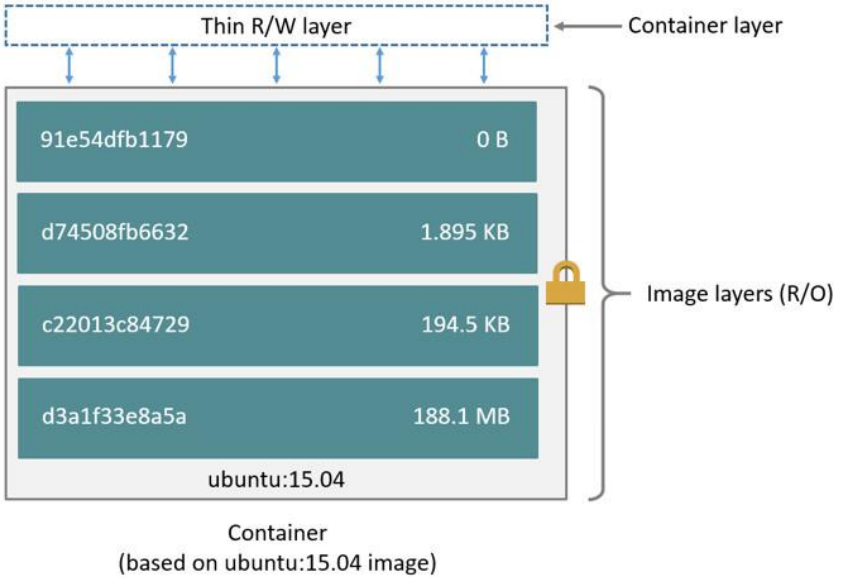
Using **`docker build`** (*remember -t and -f option*) users can create an automated build that executes several command-line instructions in succession.

Keywords of Docker world: docker image

A Docker Image is the **template** needed to build a running Docker Container (*the running instance of that image*).

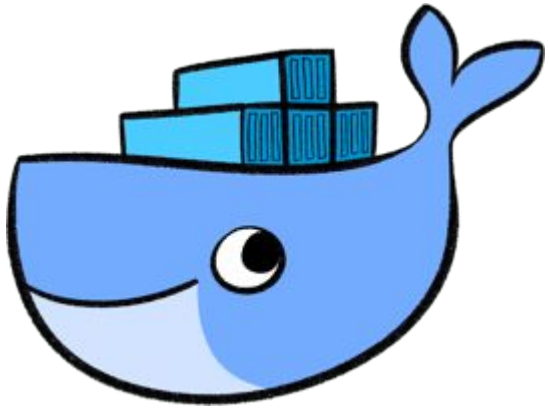
An Image is an ordered collection of root filesystem changes and the corresponding execution parameters for use within a container runtime. An image typically contains a **union of layered filesystems** stacked on top of each other.

Attention:
if we make a change to the image, when we commit we will update only that layer and not the whole image



Keywords of Docker world: docker container

A container is a **runtime instance** of a docker image.



A Docker container consists of

- A Docker image
- An execution environment
- A standard set of instructions

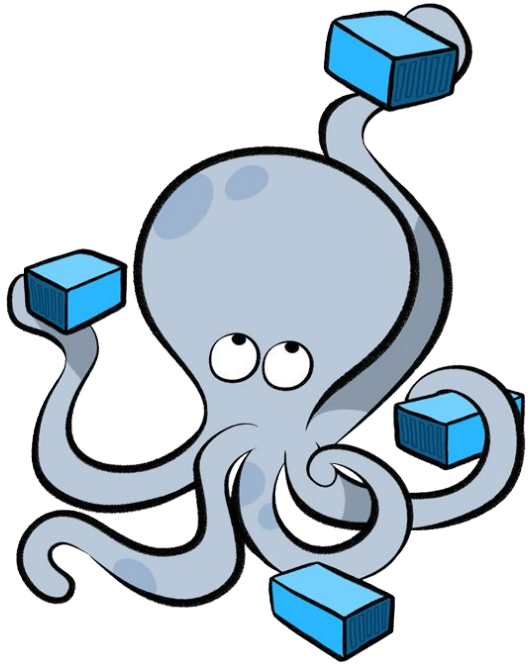
The concept is borrowed from **Shipping Containers**, which define a standard to ship goods globally. Docker defines a standard to ship software.


SANTA CRUZ
MONROVIA
HAMBURG SÜD

Docker tools: Docker Compose

Docker Compose is a tool for defining and running multi-container Docker applications in a single file

It uses YAML files to configure the application's services and performs the creation and start-up process of all the containers with a single command.



smart example:

```
# Filename: docker-compose.yml
wordpress:
    image: wordpress:4.2.2
    links:
  -   db:mysql
   ports:
  -   8080:80

db:
    image: mariadb
    environment:
        MYSQL_ROOT_PASSWORD: example
```

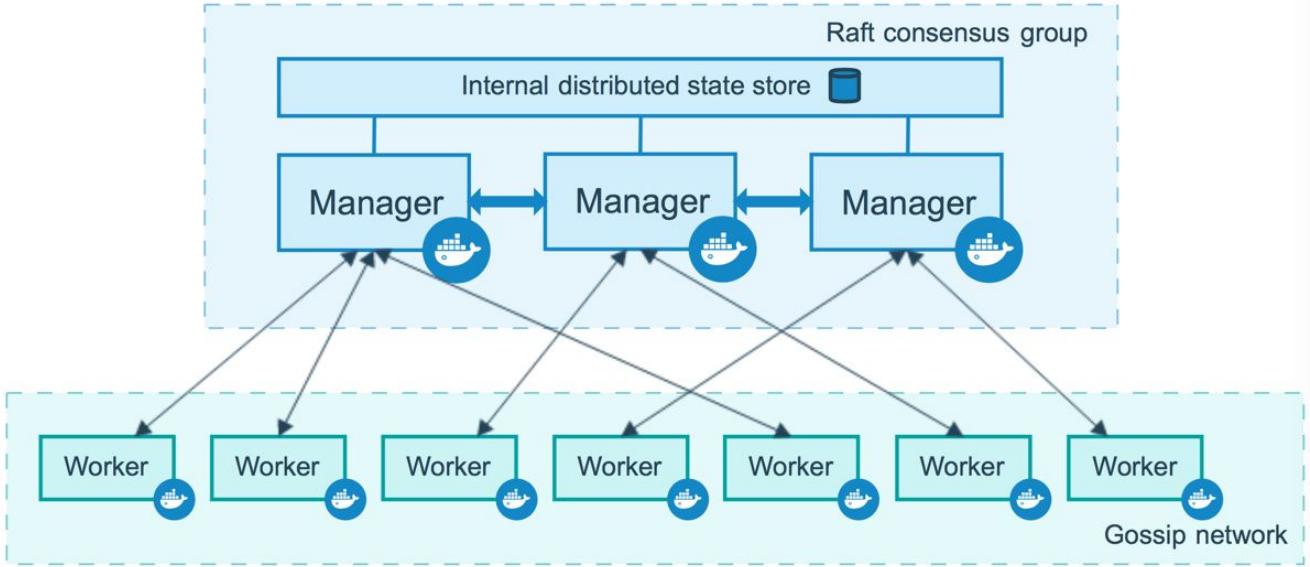
Docker tools: Docker Swarm

Docker Swarm is the name of a **standalone native clustering tool for Docker**. Docker Swarm pools together several Docker hosts and exposes them as a single virtual Docker host. It serves the standard Docker API, so any tool that already works with Docker can now transparently scale up to multiple hosts.

Docker node:
A node is a physical or virtual machine running an instance of the Docker Engine in swarm mode

Manager:
perform swarm management and orchestration duties. By default manager nodes are also worker nodes.

Worker:
execute tasks.

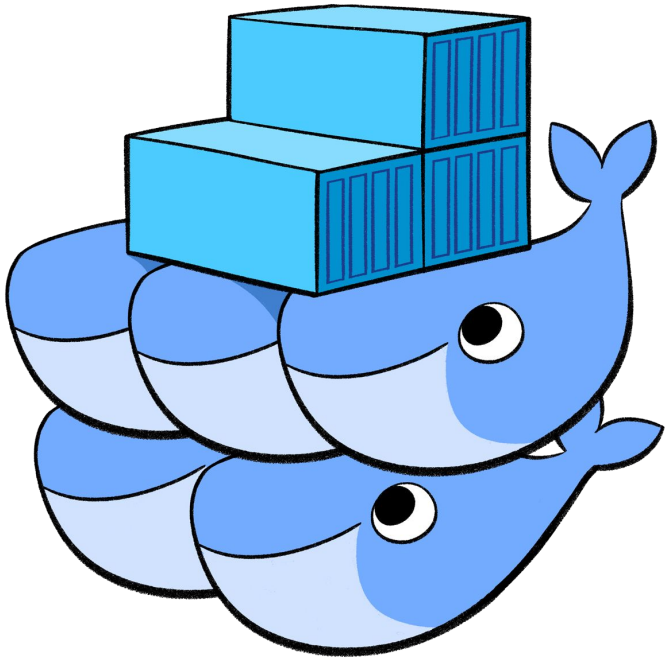


Docker tools: Docker Swarm

In Docker Swarm we introduce the concept of **service**.

For each service coincide more containers that run the same application.

It is a concept very similar to the clustering within which to guarantee fault-tolerance and high available.



End of first part

Are you ready for hands on session?

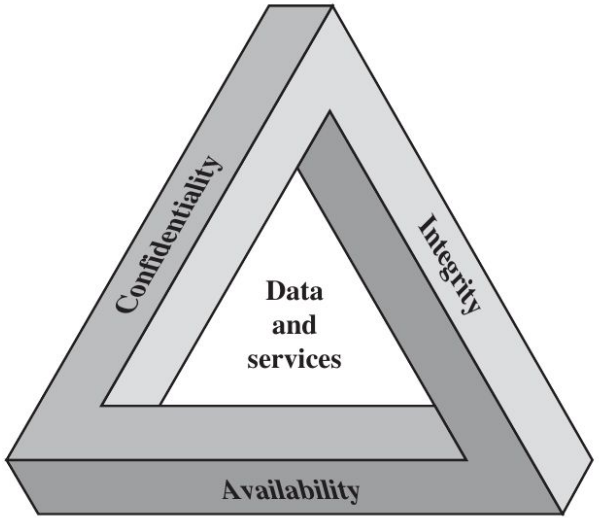
Intro at Network Security

Network Security understood as the security of applications distributed over the IP network and security of network infrastructures

Intro at Network Security

Three pillars of network security:

1) Confidentiality
2) Integrity
3) Availability



Remember if you want to make your system safe, know that you must be the first enemy of your system

Step 0: Read “The Art of War” di “SUN TZU”

“The victorious warriors first win and then go to war, while the defeated warriors first go to war and then try to win.”

“As in war, the rule is to avoid what is strong and to hit what is weak”

The other Step

low profile

high profile (more intrusive)

- Fingerprint
- Scanning
- Enumeration

Some Tools

1) Fingerprint :
   a) Fantasy, all information is good

2) Scanning & Enumeration
   a) NMAP
   b) NETCAT

Brute Force

It's the more simple attack technique that we can image.

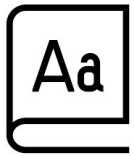
when we want login into a protected system and don't have credentials, we can try to login making a password combining ASCII characters

Try, try and try again

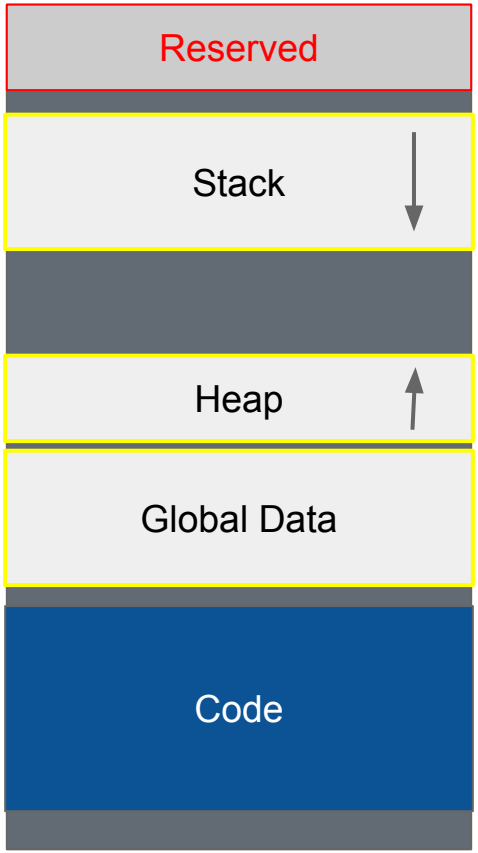
“It isn’t the more efficient way to obtain a credentials to access at the system, but with a good dictionary (simple text file that contains the password to be tested) it’s work in most cases,especially for weak passwords. ”

Some Tools

1. To make dictionary:
   a. Know the target;
   b. Google;
   c. Fantasy.
2. Brute Force tools:
   a. Hydra ();
   b. John the ripper.



Simple Model Memory



Each area contains different information and each process can only access certain portions of these areas

Stack: Area that contain local variables and other information for function call.

Heap: Area that contain variables that are dynamically allocated.

Global Data: Area that contain global variables.

Buffer Overflow

A condition at an interface under which more input can be placed into a buffer or data holding area than the capacity allocated, -overwriting other information.
Attackers exploit such a condition to crash a system or to insert -specially crafted code that allows them to gain control of the system.

[STALLINGS]

Buffer Overflow

Usually a buffer overflow attack have as scope modify the content of variables near buffer, or load and execute a malicious code.

Buffer Overflow: Example

What is wrong in this function's code?

```
void readMessage(int socket,char * buffer){
      FILE *fp=fdopen(socket,"r");
      fscanf(fp,"%s",buffer);
      fclose(fp);
}
```

Useful references

1. Course materials by Simon Pietro Romano

   docenti.unina.it;

2. Hacking Exposed 7: Network Security Secrets and Solutions;

3. Cryptography and Network Security: Principles and Practice ;

4. Google;

5. Exploit-DB;

6. Docker web-site.

7. Buffer Overflow gitlab project: http://gitlab.comics.unina.it/Previtera/BufferOverflow/tree/master

8. Jabba The Hack gitlab project:

   http://gitlab.comics.unina.it/NS-Projects/JabbaTheHack/tree/master/JabbaDocker