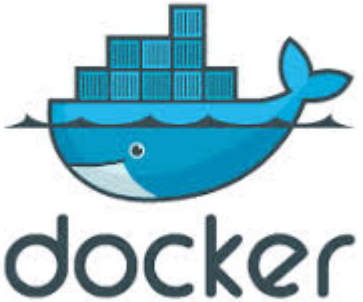
Lesson iut.univ-paris8.fr 2018-05

# Introduction to VMs & Containers




This Document:
http://arnaud-nauwynck.github.io/docs/Intro-VM-Container.pdf

arnaud.nauwynck@gmail.com

Hardware
→ Software

"All problems in computer science can be solved by another level of indirection"

(the "fundamental theorem of software engineering")

David Wheeler

Hardware

→ Assembly Langage

→ Compiler & Langages

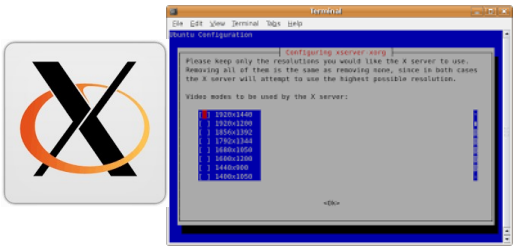
→ Shared Libraries

→ Frameworks

# Problem 1 To Solve :
# Libraries Dependencies Hell

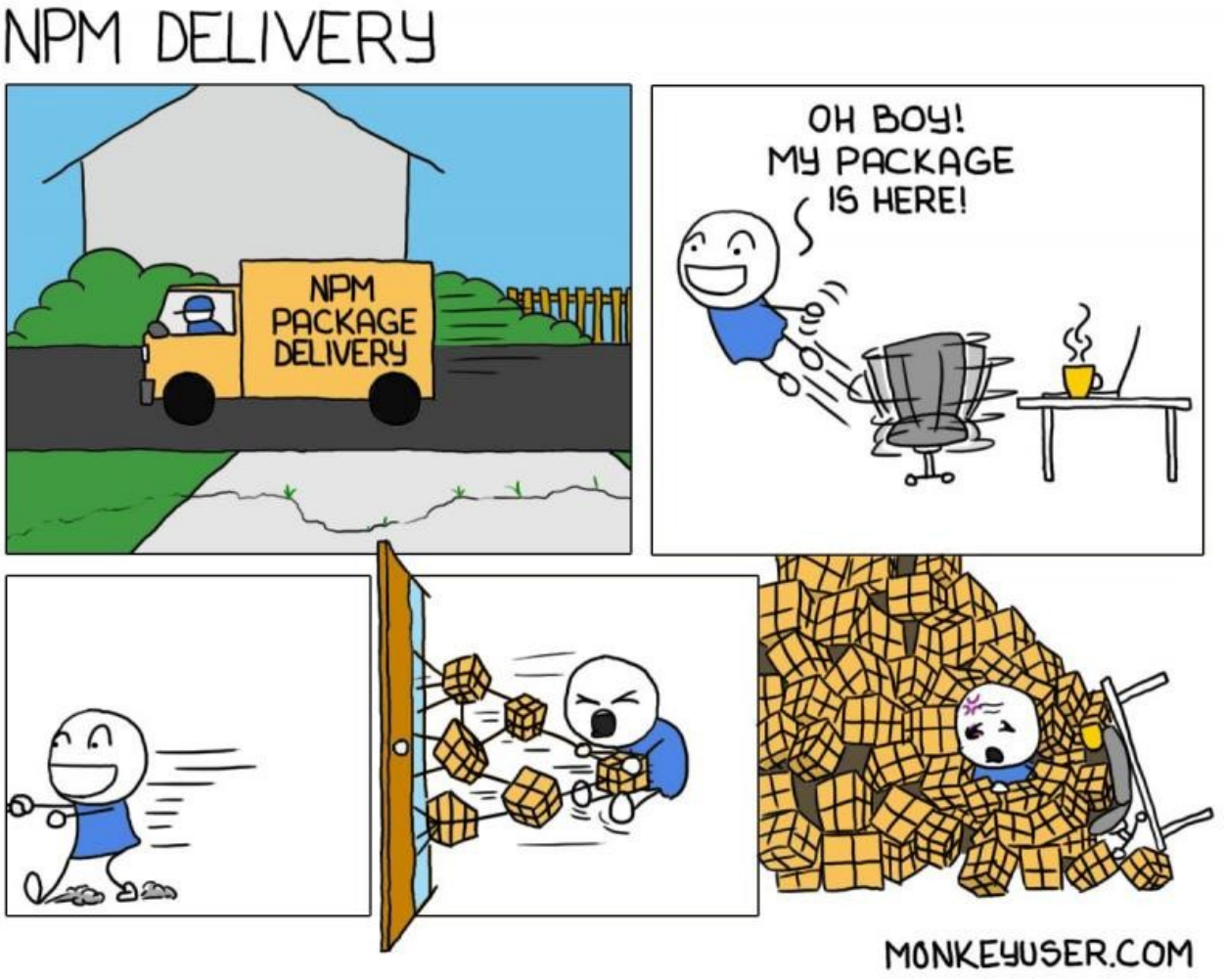
Think “Windows” …
to understand “Dll Dependency Hell”


Think “Debian” …
to upgrade Xserver driver



Think “NPM” / “NodeJS” …
package.json node_modules and versionning

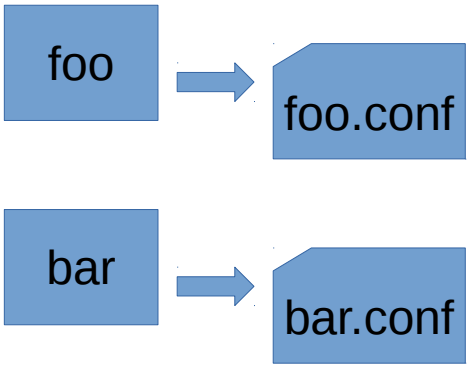
# Example: Npm modules hell



# Problem 2 To Solve : Configuration Conflicts / Reuse Component

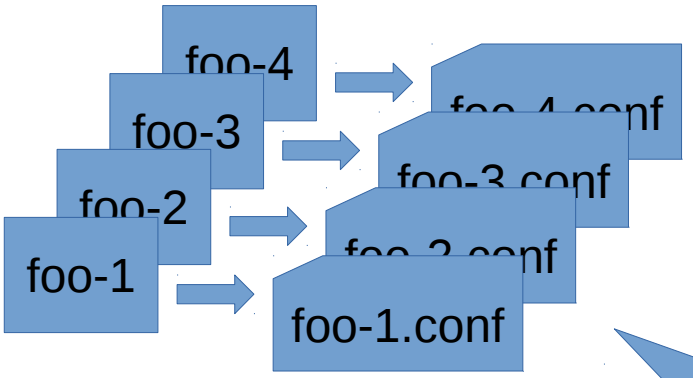
Think “/etc/<app>.conf.d” …
to understand “Configuration” for “<app>”

OK



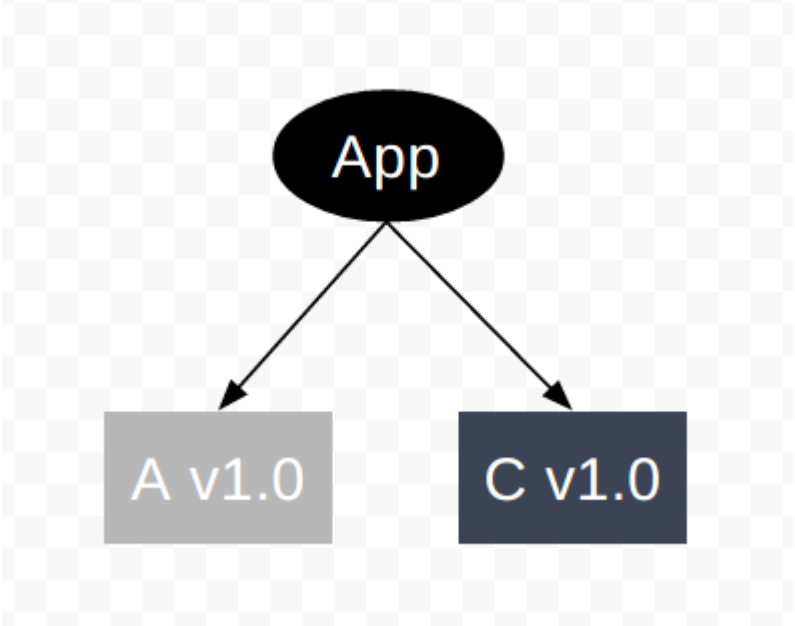

How to have multiple instances ??
<app.1>, <app.2> … , <app.N>

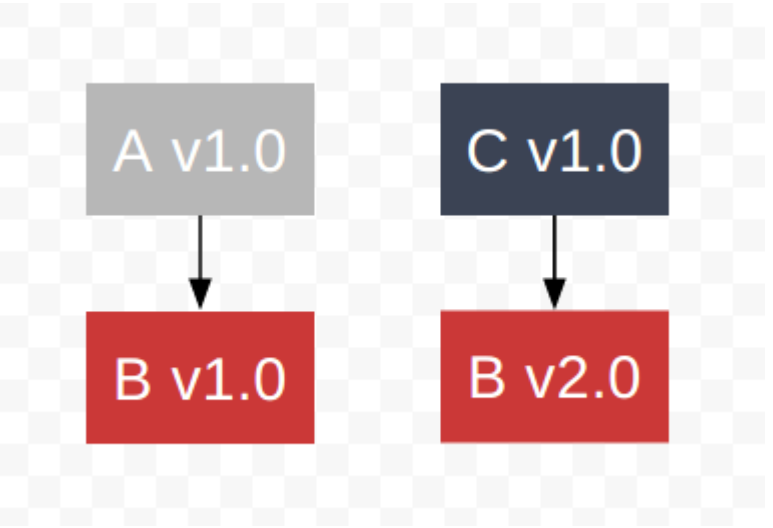
KO !



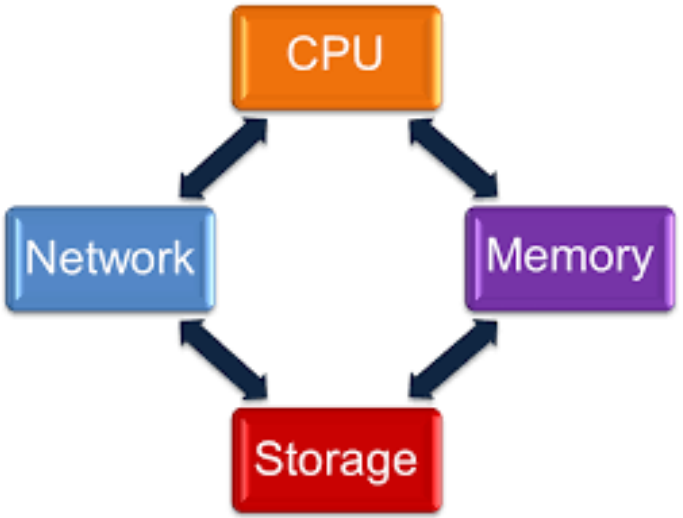

Isolate shared / specific part?

# Problem 3 : Versionning
# Multiple Versions Co-Existing



# Problem 4 : Resource Sharing




Hardware

→ OS

→ Virtual Machine

→ Containers

→ Orchestrators

# Problem 5 : Resource Scheduling

# Virtualisation – Containers History

2003
Qemu
Xen

2005
Intel VT-x

2007
KVM

VirtualBox
( Innotek
→ Sun 2008
→ Oracle 2010)

2008
LXC

2010
dotCloud
company
created by
Solomon Hykes

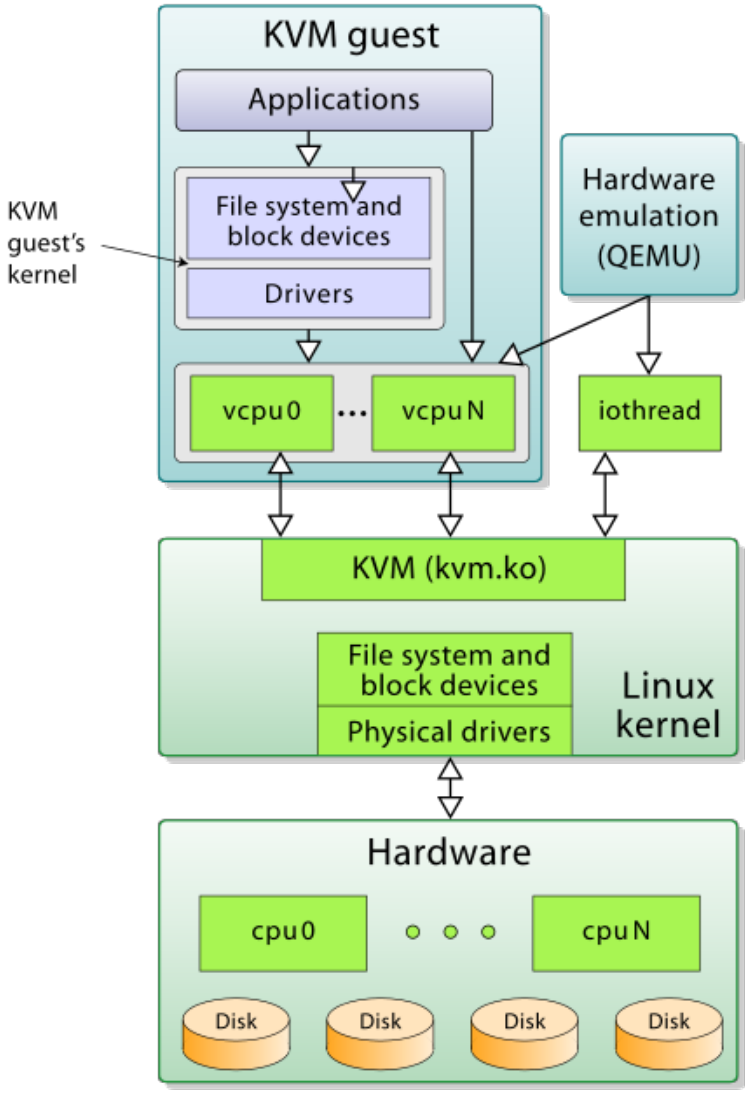
2013
Docker released
DotCloud renamed
Docker Inc

2016
Docker
on Windows

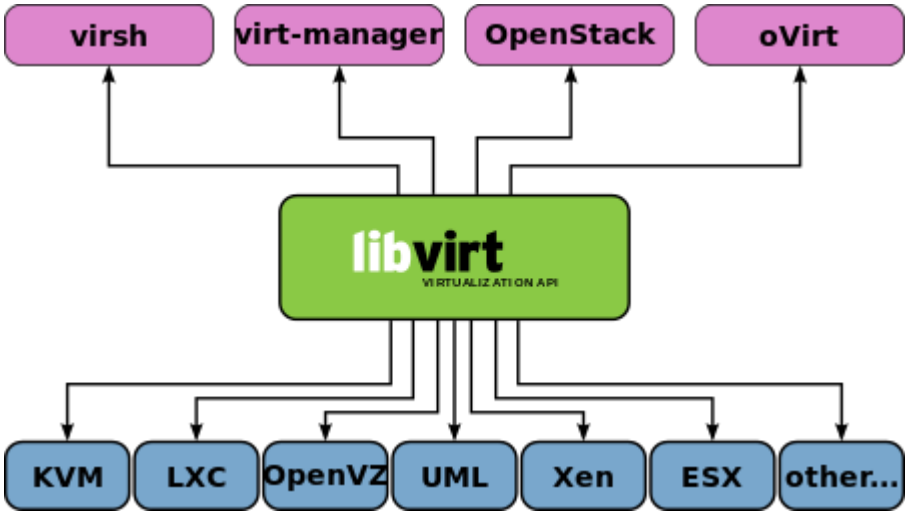
2018
Solomon
quits Docker
.. what's next?

https://en.wikipedia.org/wiki/Timeline_of_virtualization_development

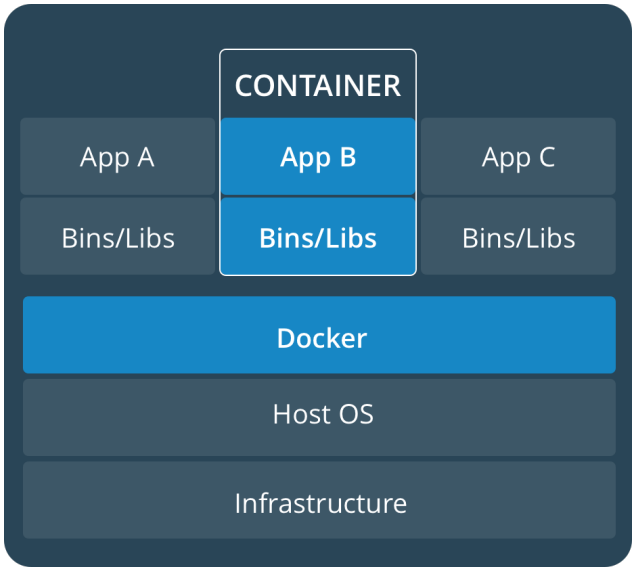
# Qemu – KVM – VirtualBox ...

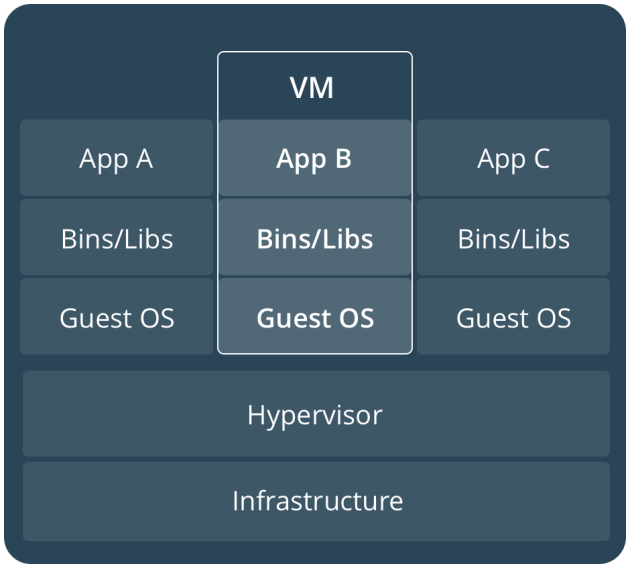



# LibVirt ...


virsh
virt-manager
OpenStack
oVirt
libvirt
VIRTUALIZATION API
KVM
LXC
OpenVZ
UML
Xen
ESX
other...

# Lightweight Containers ↔ Virtual Machines

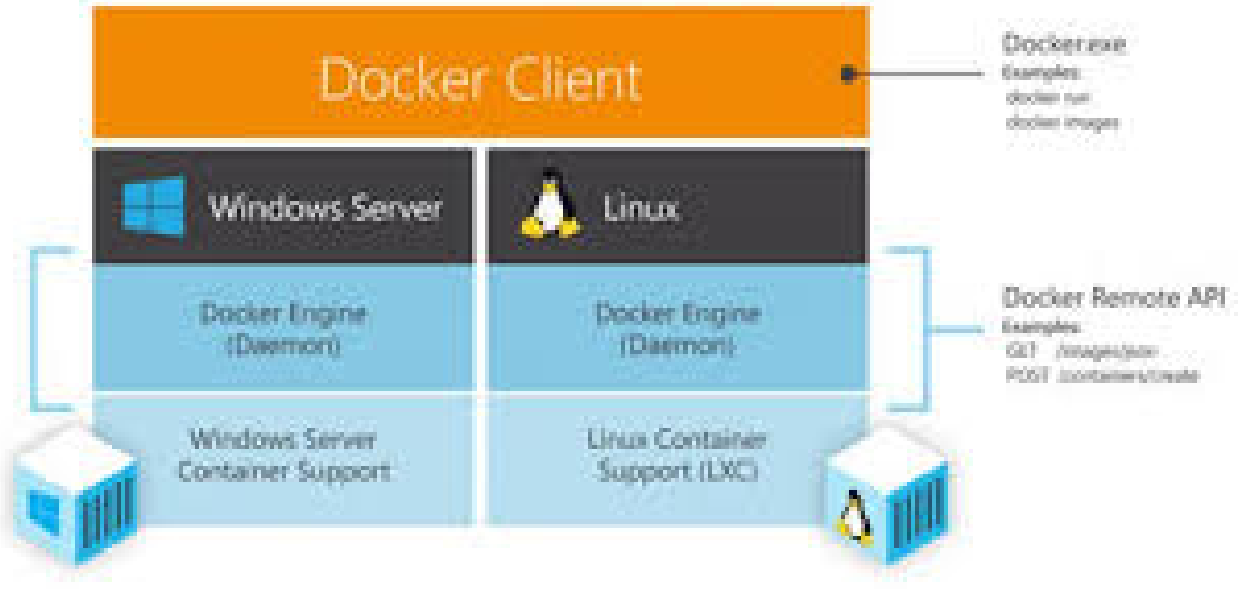



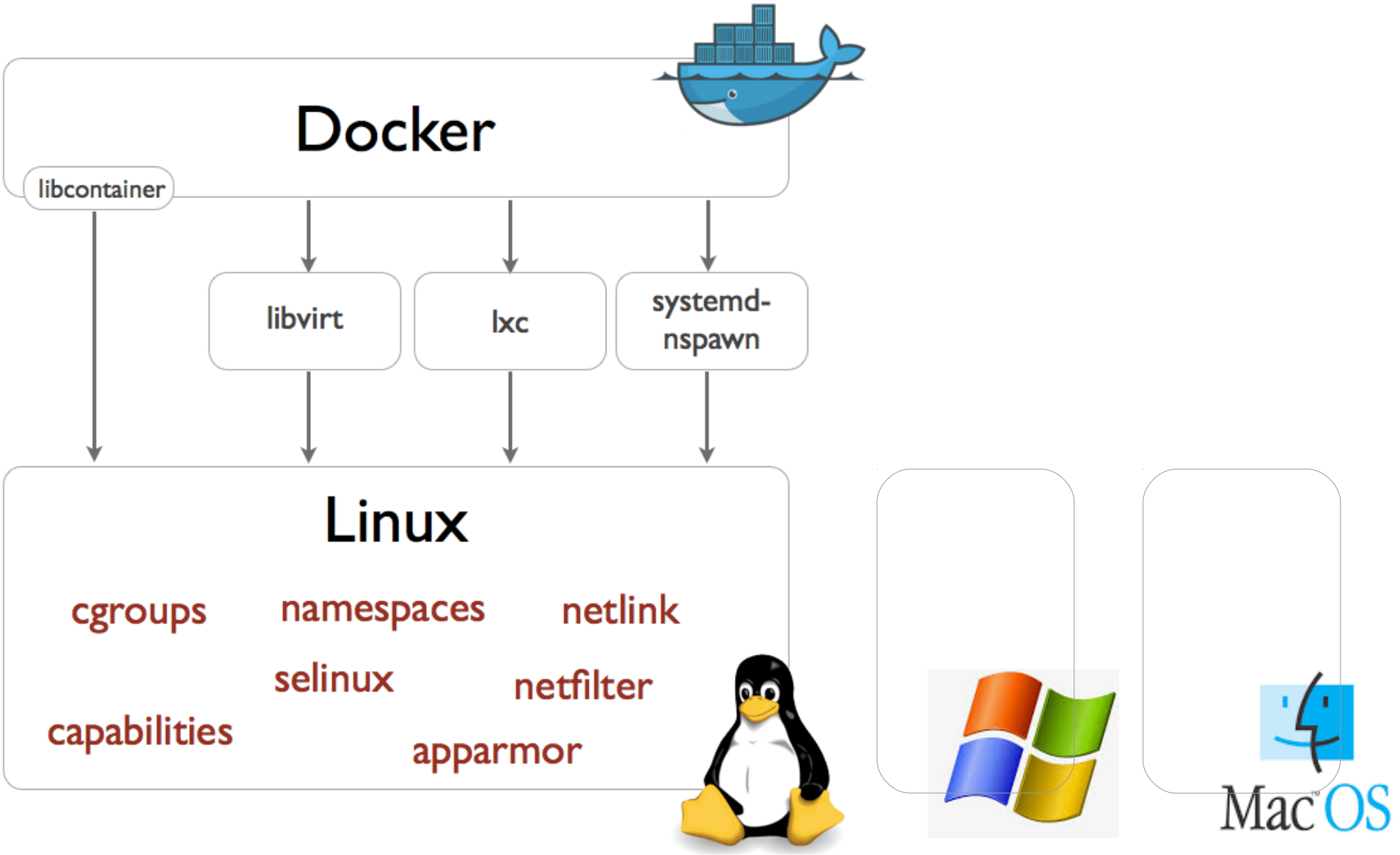


https://docs.docker.com/get-started/#containers-and-virtual-machines

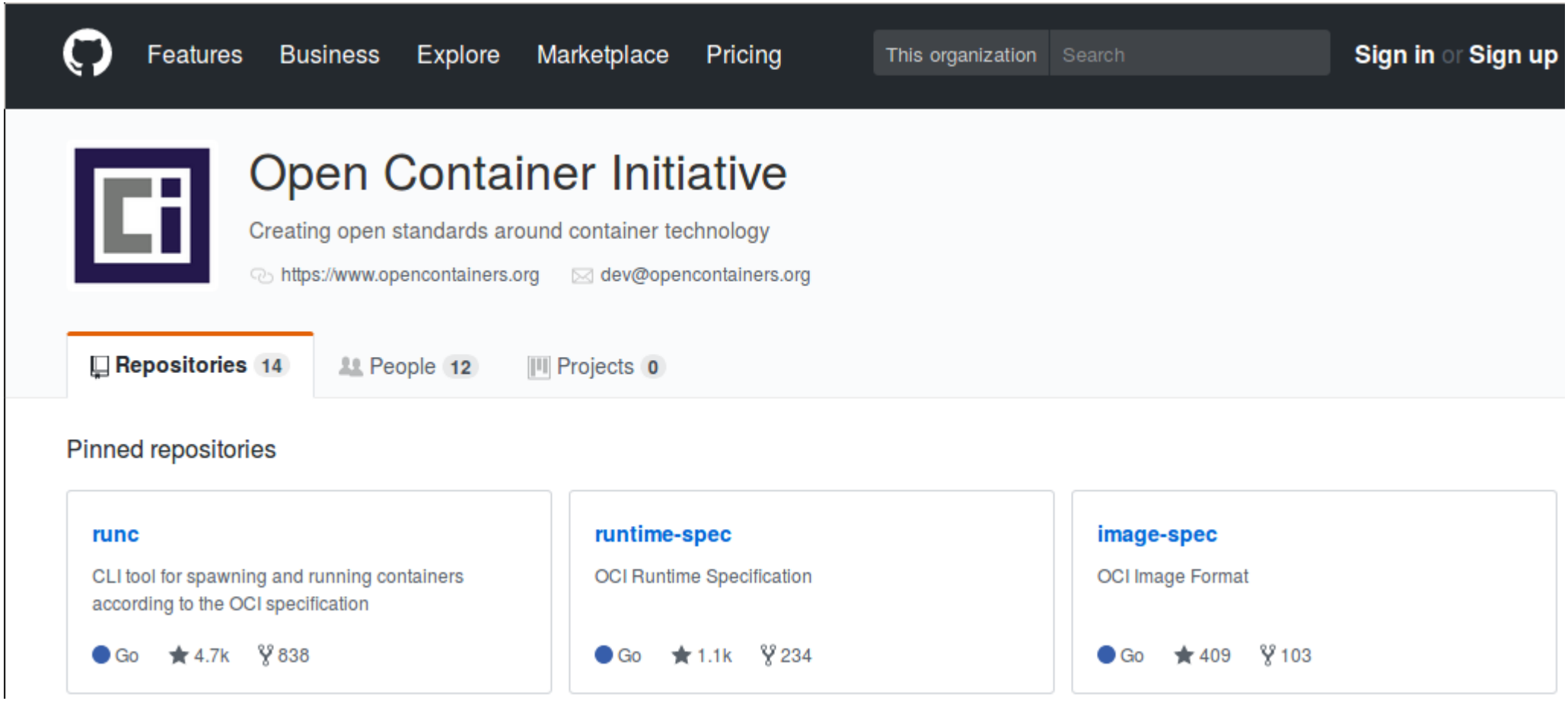
# Docker : Linux Container Then Windows & Mac



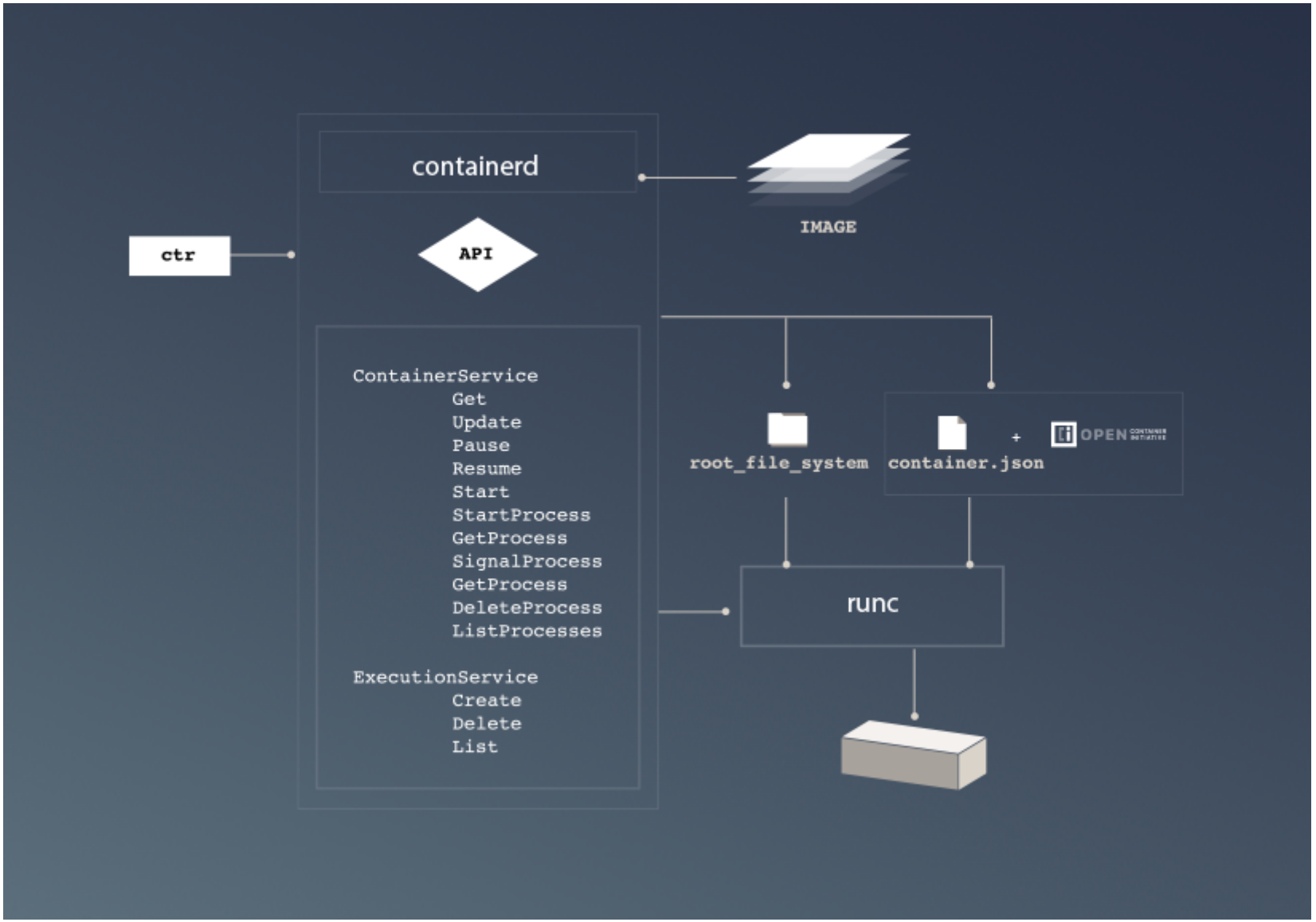
# Docker – Libcontainer



# Libcontainer, Open Container

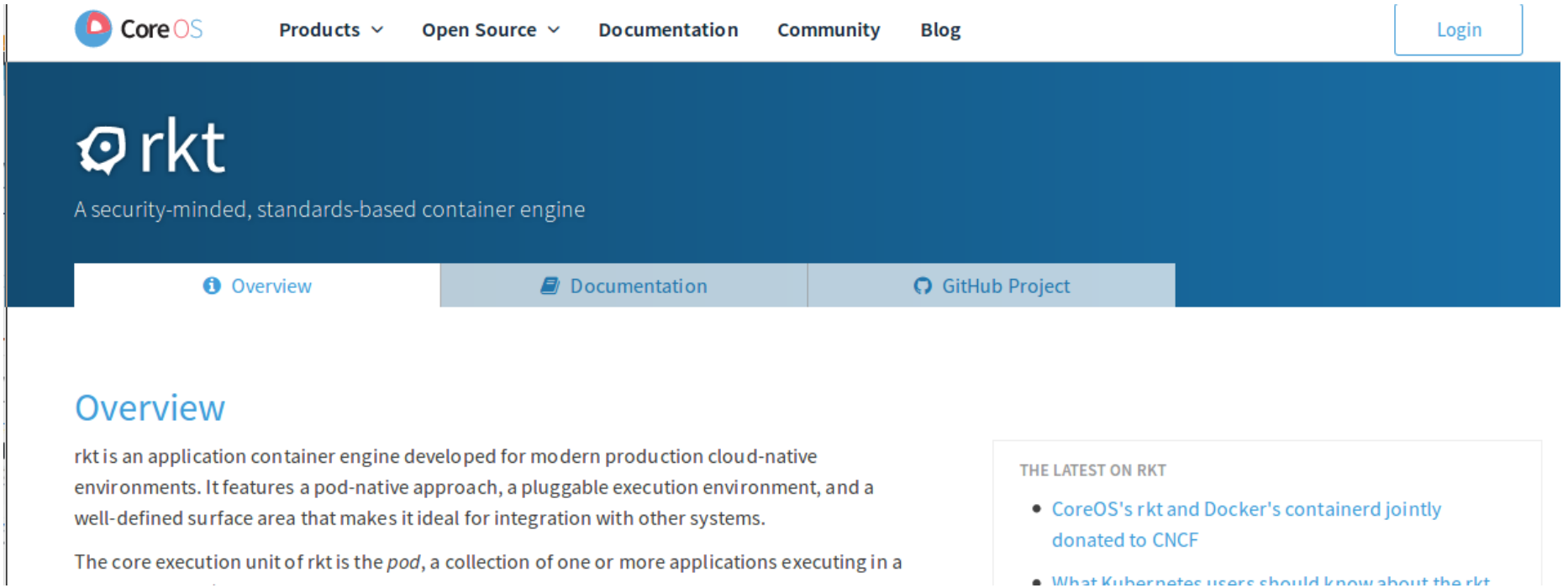


# https://containerd.io/



# Other Container Technologies..

https://coreos.com/rkt/



# Linux Isolations primitives for Containers

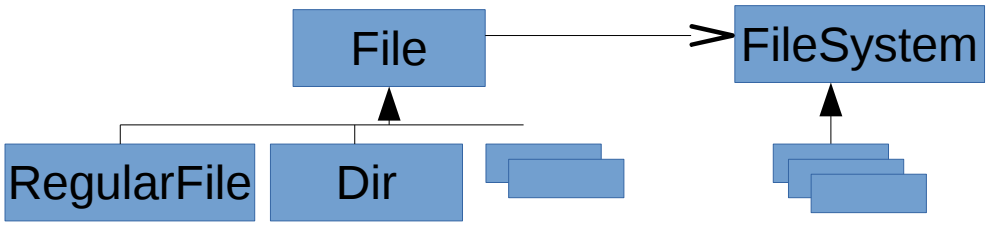
# FileSystem

```
READ(2)                                    Linux Programmer's Manual

NAME
       read - read from a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t read(int fd, void *buf, size_t count);

DESCRIPTION
       read() attempts to read up to count bytes from file descriptor fd int
```

```
WRITE(2)                                   Linux Programmer's Manual

NAME
       write - write to a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t write(int fd, const void *buf, size_t count);

DESCRIPTION
       write()  writes  up  to  count  bytes from the buffer pointed buf to the file r
       descriptor fd.
```

# FileSystem : Class / Design-Pattern Interpretation

**Facade, Delegate** design-pattern : all System I/O goes to a facade API

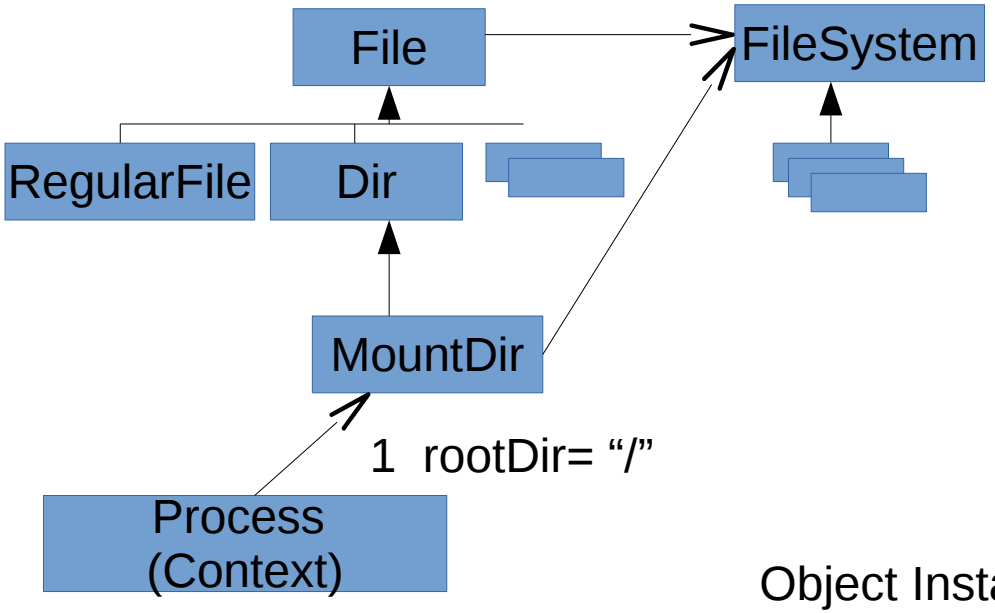



```
public static abstract class File {
    public abstract FileSystem getFileSystem();
    public abstract int getFD();
}

public static abstract class RegularFile extends File {
    // read(), write() => cf FileSystem.readFile(this), writeFile(this) ..
}

public static abstract class Dir extends File {
    // list(), mkdir() => cf FileSystem.listDir(), mkdir() ..
}

public static abstract class FileSystem {
    // readFile(), writeFile(), ..
    // listDir(), mkdir(), ..
}
```

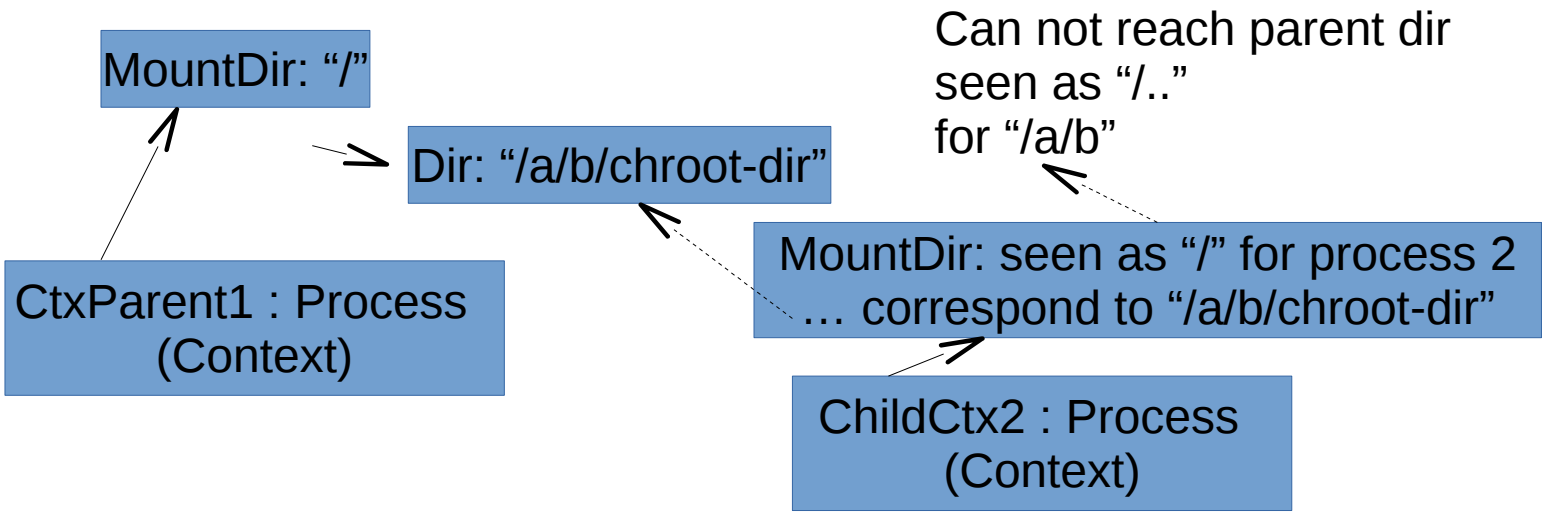
# ChRoot

```
CHROOT(8)                          User Commands                          CHROOT(8)

NAME
       chroot - run command or interactive shell with special root directory

SYNOPSIS
       chroot [OPTION] NEWROOT [COMMAND [ARG]...]
       chroot OPTION

DESCRIPTION
       Run COMMAND with root directory set to NEWROOT.

       --groups=G_LIST
              specify supplementary groups as g1,g2,..,gN

       --userspec=USER:GROUP
              specify user and group (ID or name) to use

       --skip-chdir
              do not change working directory to '/'

       --help display this help and exit
```

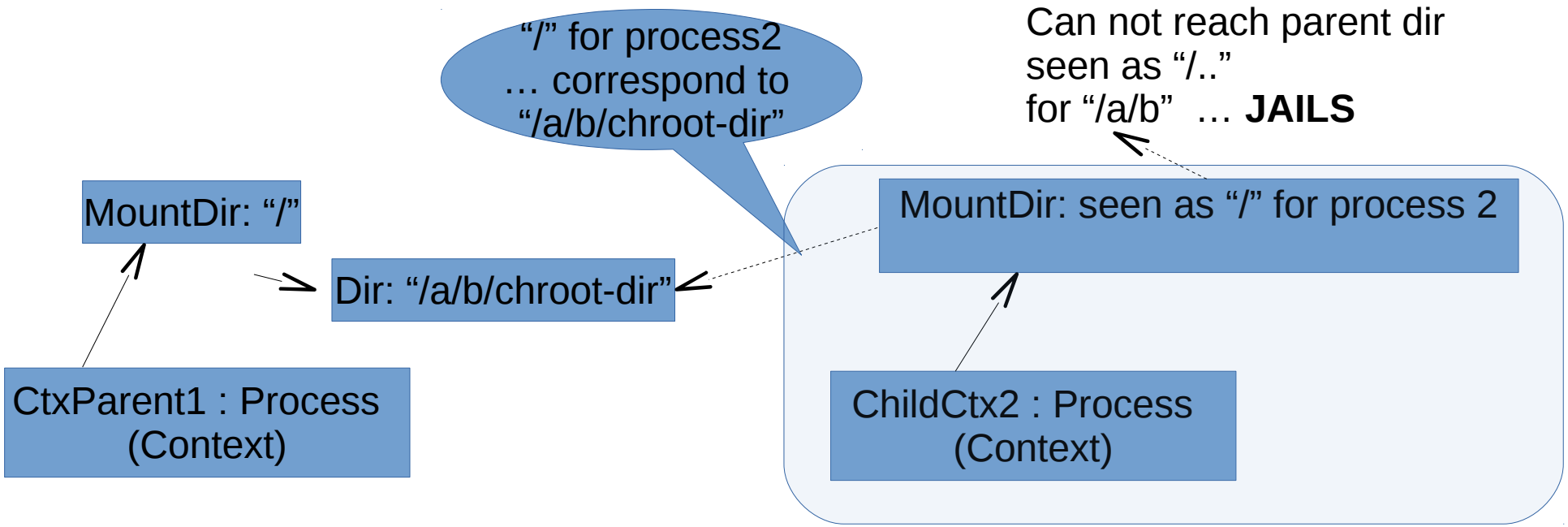
# ChRoot – Class/Pattern Interpretation




Object Instance Diagram : forking a ch-rooted process



# ChRoot – Object Diagram for ch-rooted child process

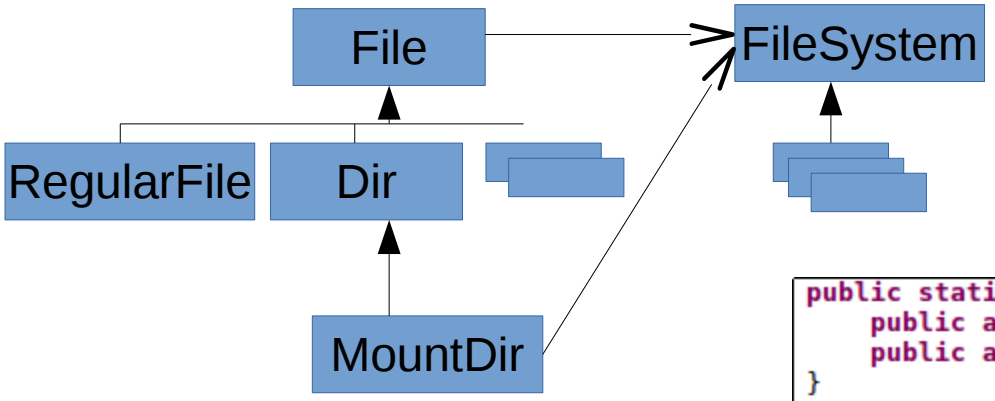


# Mount

```
$ mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=8164088k,nr_inodes=2041022,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=1636904k,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
```

# Mount : Class / Design-Pattern Interpretation

**Adapter** design-pattern : adapt a Dir sub-class to delegate to FileSystem
MountDir → FileSystem




```
public static abstract class File {
    public abstract FileSystem getFileSystem();
    public abstract int getFD();
}

public static abstract class RegularFile extends File {
    // read(), write() => cf FileSystem.readFile(this), writeFile(this) ..
}

public static abstract class Dir extends File {
    // list(), mkdir() => cf FileSystem.listDir(), mkdir() ..
}

public static abstract class FileSystem {
    // readFile(), writeFile(), ..
    // listDir(), mkdir(), ..
}

public static abstract class MountDir extends Dir {
    FileSystem fileSystem;
    Map<String,Object> mountOptions;
    // => delegate file/dir operations to fileSystem..
}
```

# “Mount“ usage in Docker : Volumes

Example:
$ docker run -v /data:/redis-1/data  --name redis-1 redis:latest

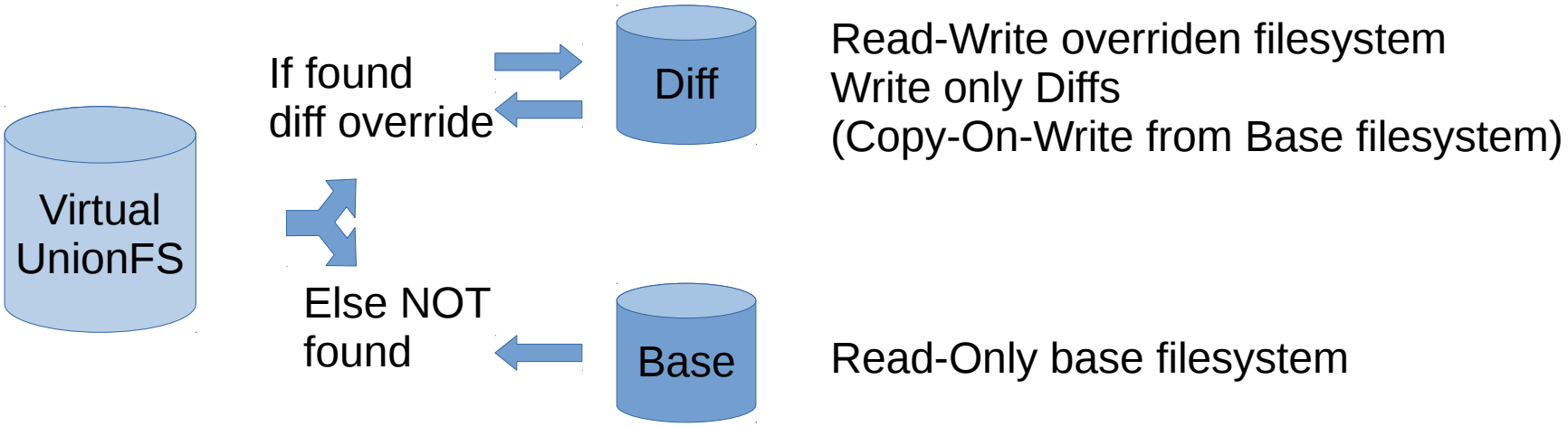
# UnionFS, AuFS

```
aufs(5)                          Linux Aufs User's Manual                          aufs(5)

NAME
       aufs - advanced multi layered unification filesystem. version 4.x-rcN-20160111

DESCRIPTION
       Aufs  is  a stackable unification filesystem such as Unionfs, which unifies sev-
       eral directories and provides a merged single directory.   In  the  early  days,
       aufs  was  entirely  re-designed  and re-implemented Unionfs Version 1.x series.
       After many original ideas, approaches and improvements, it becomes totally  dif-
       ferent  from  Unionfs while keeping the basic features.  See Unionfs Version 1.x
       series for the basic features.  Recently, Unionfs Version 2.x series begin  tak-
       ing some of same approaches to aufs's.
```

Example:
$ mkdir /tmp/rw && mkdir /tmp/aus
& sudo mount -t aufs -o br=/tmp/rw=rw:/home/user=ro none /tmp/aus/

… will show in “/tmp/aus”
the union of “/tmp/rw”
and “/home/user” (kept as read-only)

# UnionFS, AuFs

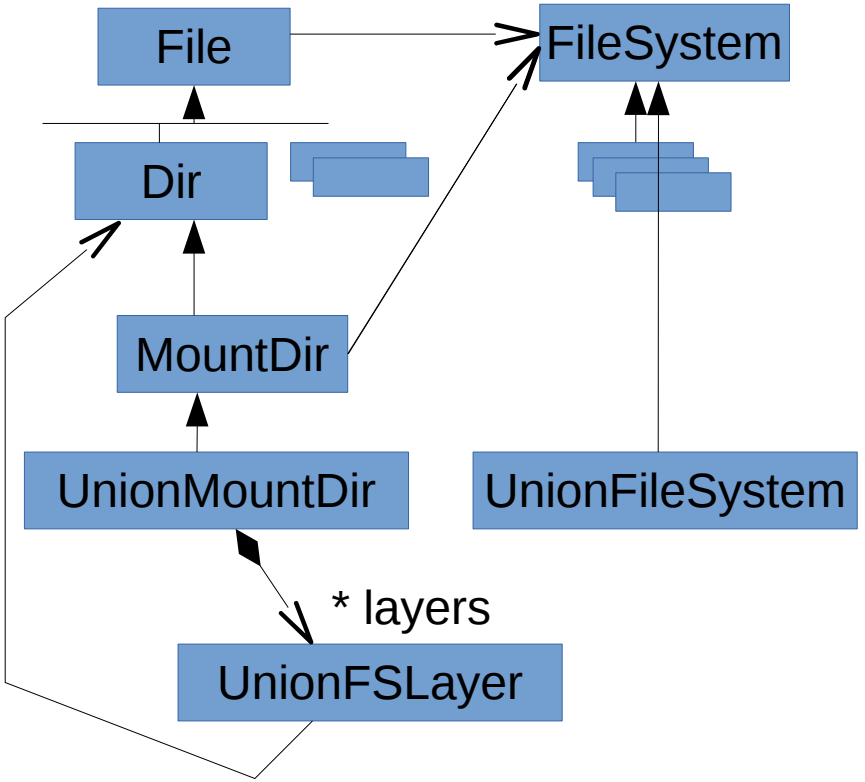



Example: Live linux distribution for test (no persistent files), on Boot USB disk

# UnionFS : Class / Design Pattern

**Decorator design-pattern ?**
Example: 1 layer Read-Write / Above 1 layer Read-Only
decorate Read-Write over a Read-Only

**Delegate / Chain-Of-Responsibility** design-pattern ?
Example: multiple layers … when match pattern, then delegate else bubbles up

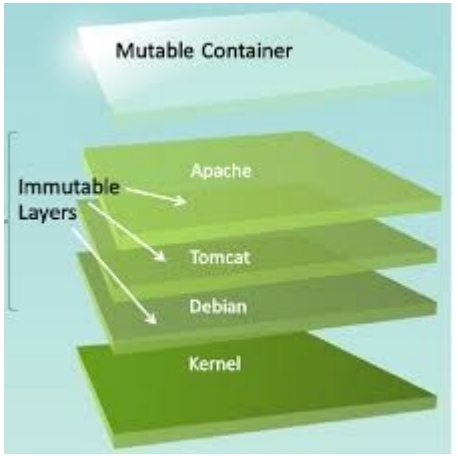



```
public static abstract class MountDir extends Dir {
    FileSystem fileSystem;
    Map<String,Object> mountOptions;
    // => delegate file/dir operations to fileSystem..
}

public static abstract class UnionMountDir extends MountDir {
    List<UnionFSLayer> layers; // <= init from mountOptions
    static class UnionFSLayer {
        Dir delegateDir;
        String readWriteMode; Map<String,Object> layerOptions;
    }
}
public static class UnionFsMountFileSystem extends FileSystem {
    // => interpret mountOptions as layers for read-write, read-only
    // ... and delegate read(),write(),list(),..  to underlying real dirs
}
```

# UnionFS usage in Docker : Image Layers Stack (Dockerfile)

Dockerfile

```
FROM debian:latest

RUN apt-get install tomcat

RUN apt-get install apache
```




$ docker build -t my-image .

# Loopback Device, Losetup

```
LOSETUP(8)                                        System Admi

NAME
       losetup - set up and control loop devices

SYNOPSIS
       Get info:

            losetup loopdev

            losetup -l [-a]

            losetup -j file [-o offset]

       Detach a loop device:

            losetup -d loopdev...

       Detach all associated loop devices:

            losetup -D

       Print the name of the first unused loop device:

            losetup -f

       Set up a loop device:

            losetup [-o offset] [--size
                    [-Pr] [--show] -f|
```

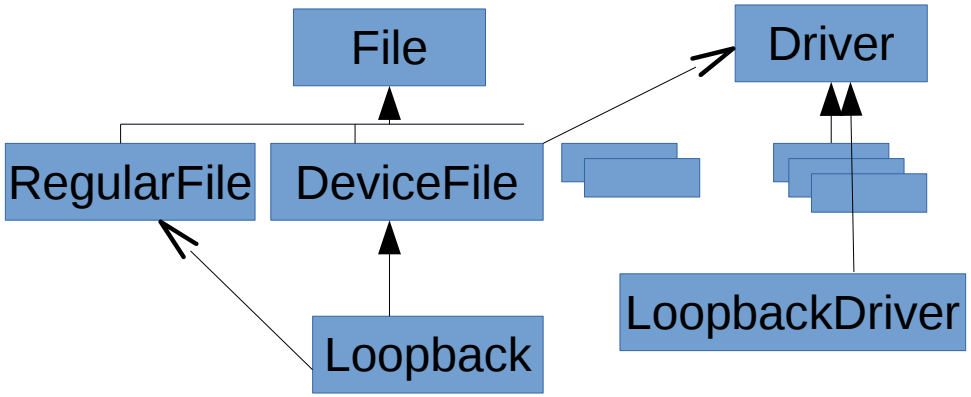
```
The  loop  device is a block device that maps its data blocks not
disk or optical disk drive, but to the blocks of a regular file in
device.   This can be useful for example to provide a block device
file, so that it can be mounted with the mount(8) command.  You co

       $ dd if=/dev/zero of=file.img bs=1MiB count=10
       $ sudo losetup /dev/loop4 file.img
       $ sudo mkfs -t ext4 /dev/loop4
       $ sudo mkdir /myloopdev
       $ sudo mount /dev/loop4 /myloopdev
```

# Loopback – Class / Design Pattern

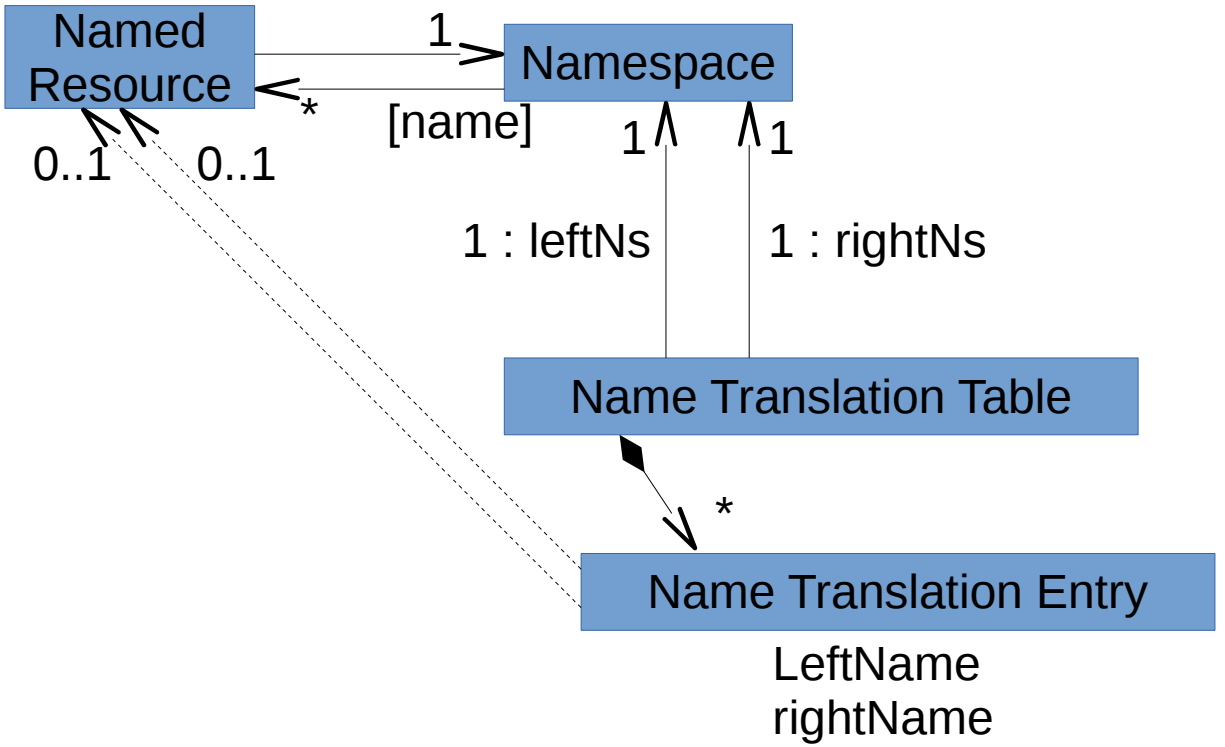
Adapter design-pattern : DeviceFile → File

Class Diagram:

```
public static abstract class Driver {
}

// example: "/dev/sdc0" ... device for accessing hard-disk driver
public static abstract class DeviceFile extends File {
    Driver driver;
    int minor, major; Map<String,Object> deviceOptions;
    // readblock(), writeblock()  or readchar(), writechar()
}

public static abstract class LoopbackDeviceFile extends DeviceFile {
    File loopFile;
    Map<String,Object> loopbackOptions;
    // => delegate device read..()/write..() to loopFile
}
```

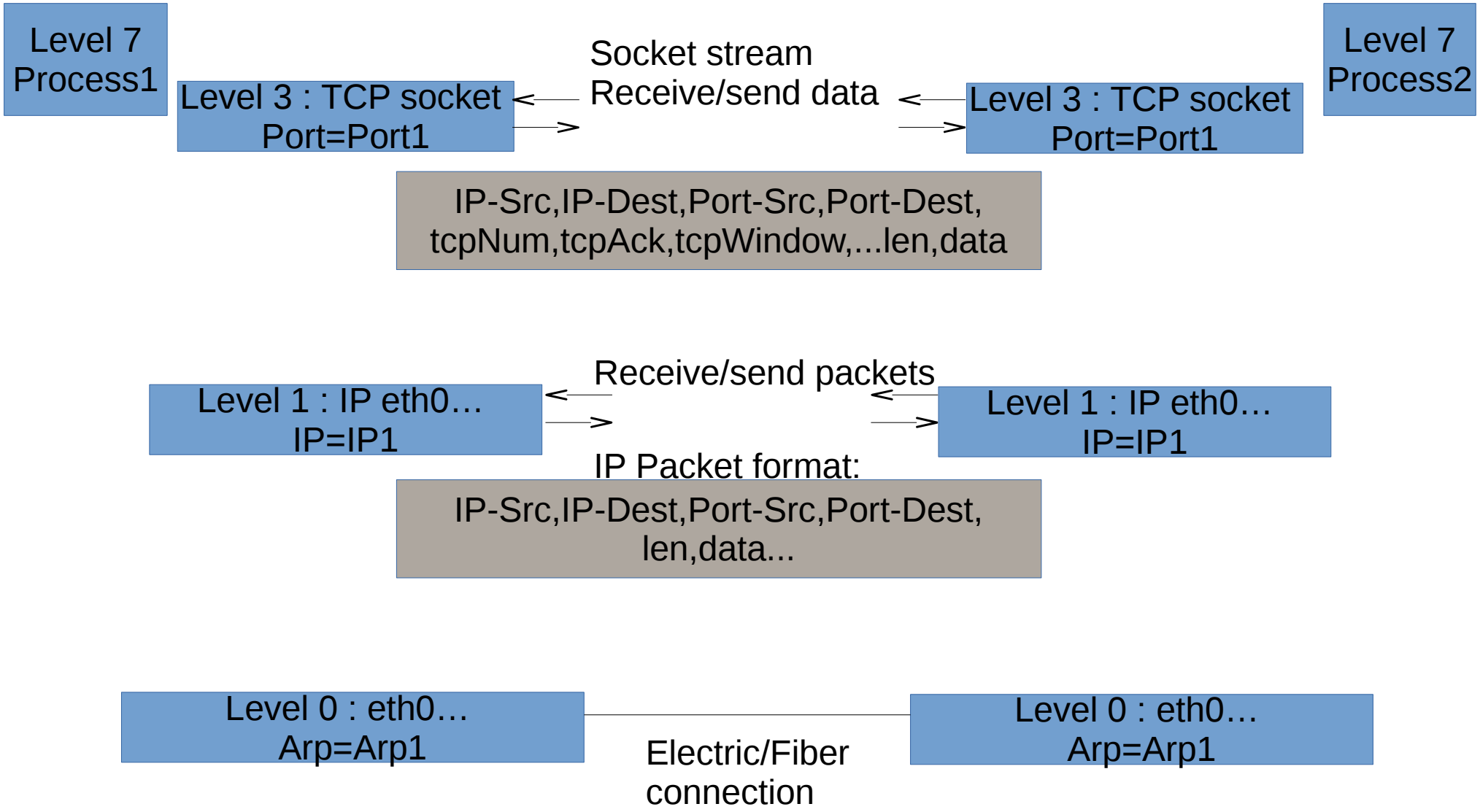


# Namespaces (UTS, PIDs, ..)



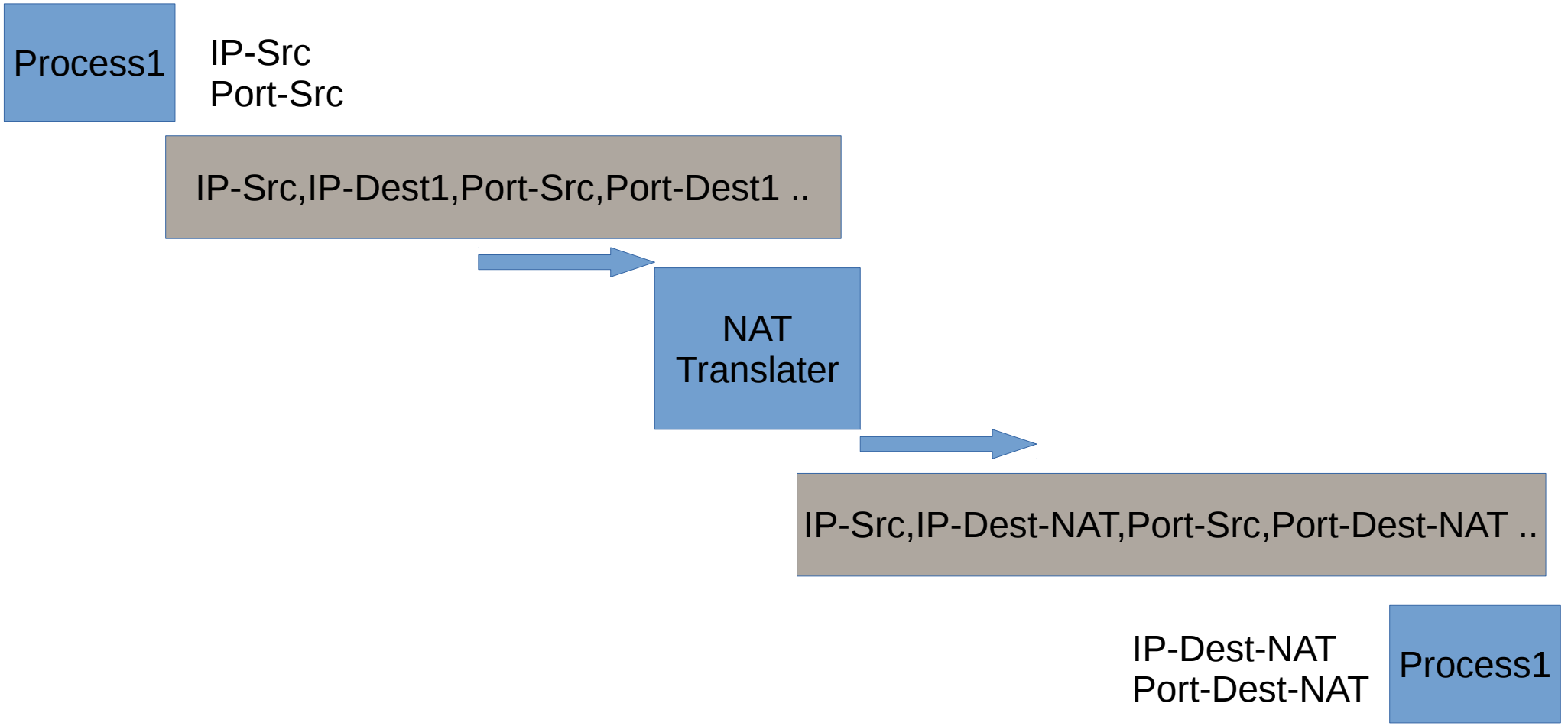

Example of Namespaces in Linux: PIDs, Hostnames, ...

# Reminder on TCP-IP
# Socket, Packet (IP:Port → IP:Port)




Notice: can have Host1=Host2,  IP1=IP2, ...

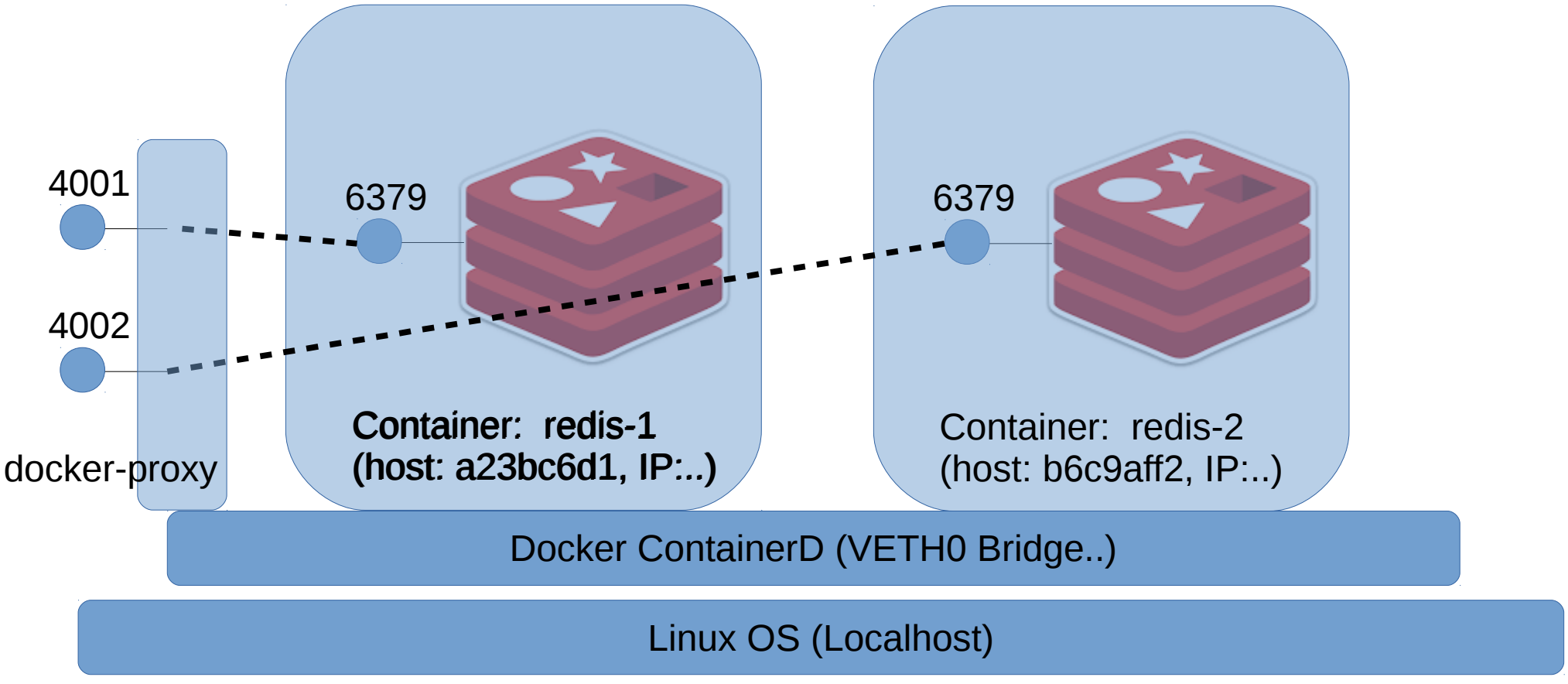
# NAT : Network Address Translation



# Example: Docker Port Export

$ docker run **-p 4001:6379** --name redis-1 redis:latest

$ docker run **-p 4002:6379** --name redis-2 redis:latest



# .. Docker -p <extport>:<intport>

```
$ docker run -p 4001:6379 --name redis-1 redis:latest
1:C 03 May 21:07:08.779 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 03 May 21:07:08.779 # Redis version=4.0.9, bits=64, commit=00000000, modified=
1:C 03 May 21:07:08.779 # Warning: no config file specified, using the default con
 redis-server /path/to/redis.conf
1:M 03 May 21:07:08.780 * Running mode=standalone, port=6379.
```

Check connecting manually (telnet) to redis on 4001 … not on 6379 !!

```
$ telnet localhost 4001
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
```

execute redis commands

It works ..

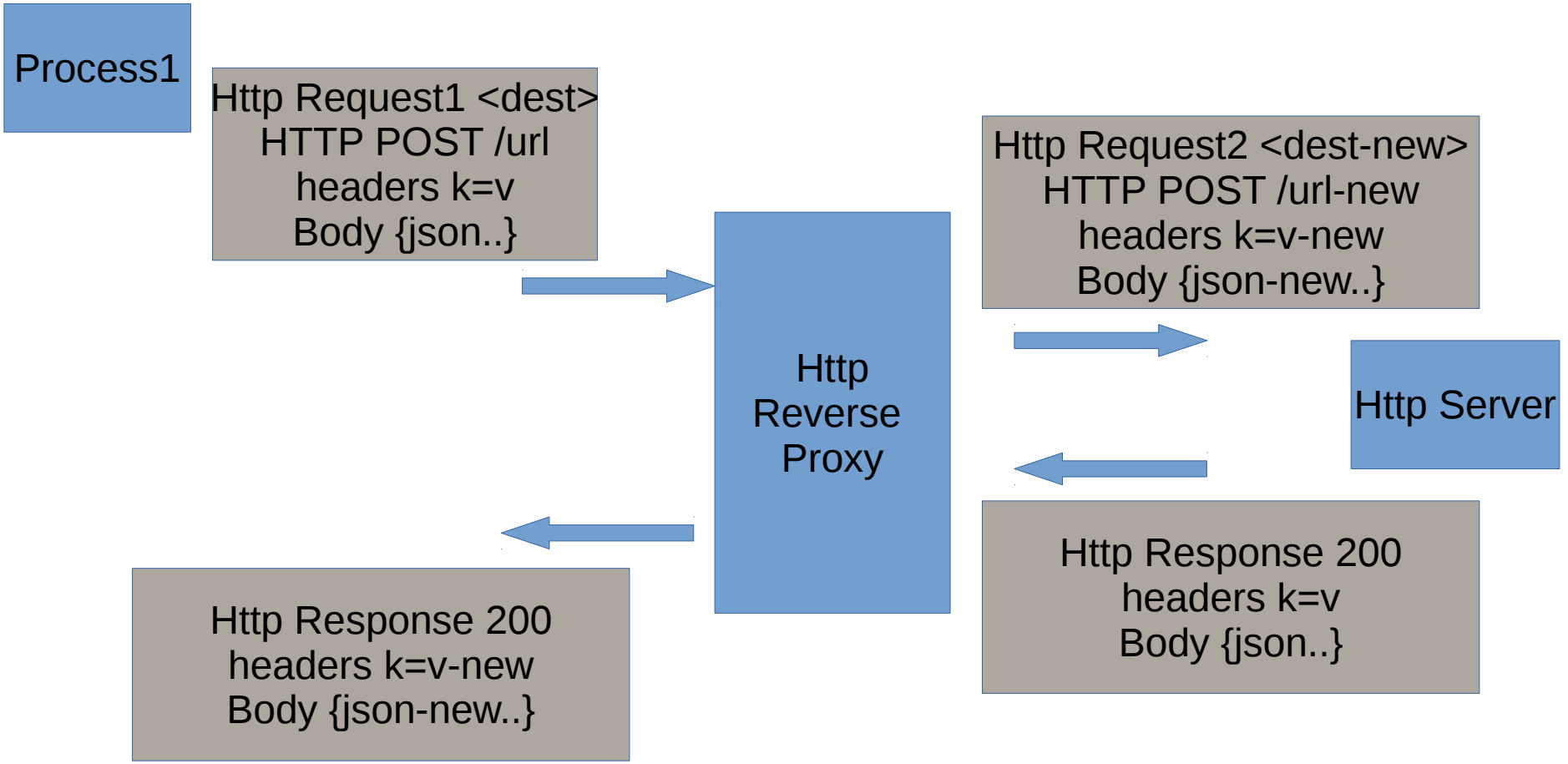
```
Escape character is '^]'.
incr mycounter
:1
incr mycounter
:2
decr mycounter
:1
```

# Result Linux processes ..

```
$ sudo netstat -nlap | grep 4001
tcp6       0      0 :::4001                 :::*                    LISTEN      13958/docker-proxy
```

```
$ sudo pstree -p 1705
dockerd(1705)─┬─docker-containe(1764)─┬─docker-containe(13963)─┬─redis-server(13980)─┬─{redis-server}(14026)
              │                       │                        │                     ├─{redis-server}(14027)
              │                       │                        │                     └─{redis-server}(14028)
              │                       │                        ├─{docker-containe}(13964)
              │                       │                        ├─{docker-containe}(13965)
              │                       │                        ├─{docker-containe}(13966)
              │                       │                        ├─{docker-containe}(13967)
              │                       │                        ├─{docker-containe}(13968)
              │                       │                        ├─{docker-containe}(13969)
              │                       │                        ├─{docker-containe}(13970)
              │                       │                        └─{docker-containe}(14272)
              │                       ├─{docker-containe}(1765)
              │                       ├─{docker-containe}(1766)
              │                       ├─{docker-containe}(1767)
              │                       ├─{docker-containe}(1768)
              │                       ├─{docker-containe}(1769)
              │                       ├─{docker-containe}(1770)
              │                       ├─{docker-containe}(1771)
              │                       ├─{docker-containe}(1773)
              │                       ├─{docker-containe}(2854)
              │                       ├─{docker-containe}(3015)
              │                       ├─{docker-containe}(3132)
              │                       ├─{docker-containe}(3186)
              │                       └─{docker-containe}(14033)
              ├─docker-proxy(13958)─┬─{docker-proxy}(13959)
              │                     ├─{docker-proxy}(13960)
              │                     ├─{docker-proxy}(13961)
              │                     └─{docker-proxy}(13962)
              ├─{dockerd}(1758)
              ├─{dockerd}(1759)
              ├─{dockerd}(1760)
```

# Idem NAT … Level 7 : Http Reverse Proxy



# DNS : Host to IP resolver

```
$ cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 127.0.1.1
```

```
NSLOOKUP(1)                                                    BIND9

NAME
       nslookup - query Internet name servers interactively

SYNOPSIS
       nslookup [-option] [name | -] [server]

DESCRIPTION
       Nslookup is a program to query Internet domain name servers.
       non-interactive. Interactive mode allows the user to query nar
       domains or to print a list of hosts in a domain. Non-interacti
       requested information for a host or domain.
```

```
$ nslookup www.google.com
Server:         127.0.1.1
Address:        127.0.1.1#53

Non-authoritative answer:
Name:   www.google.com
Address: 172.217.21.68
```

# Ingress Networks..
# (= Http Reverse Proxy + DNS + ..)

Next Chapter 2/3 :
Docker

Next Chapter 3/3 : Kubernetes Orchestration