pot: FreeBSD containers on FreeBSD

Luca Pizzamiglio
pizzamig@FreeBSD.org
FOSDEM 2018

whoami(1)

- Luca Pizzamiglio aka pizzamig@
- FreeBSD enthusiast
- Port committer since August 2017
- Building packages at trivago

Motivations 1/2

I needed a tool to easily create/run FreeBSD “instances” to

- build/develop/test ports
- develop/test Saltstack tests
- run web services

Several really good solutions already available, even if not perfect for my use cases:

- ezjail, iocage, ...

Motivations 2/2

It should run on a laptop

- limited hardware resources
- flexible network configuration (DHCP)

I wanted to

- imitate docker, FreeBSD containers for FreeBSD
- force automation → user oriented CLI
- experiment different solutions/layouts/concepts
- use and learn more about FreeBSD features

So, what is pot?

pot is a tool to automate the management of those container

- **Currently, pot is a bunch of shell scripts**
- **Basic features are covered by standard tools**
- **Advanced features will be implemented with a proper programming language**

Why ‘pot’?

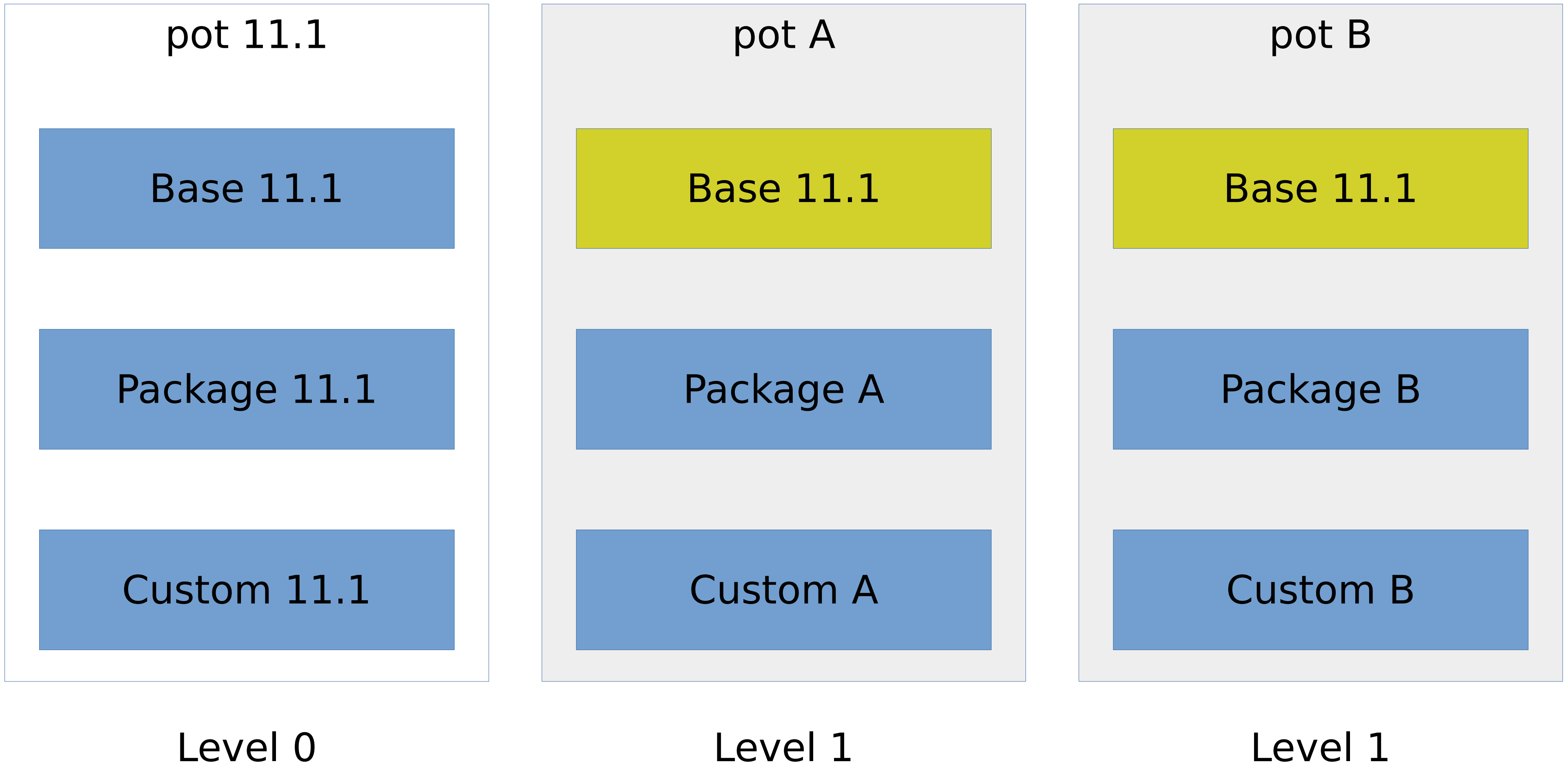
Pasta analogy [1/2]

File system components

Split the file system in several logic components:

- **FreeBSD base**
  - **It determines the FreeBSD version**
- **Packages**
  - **Installed packages**
- **Customization**
  - **Configuration files, home directories, /var**

Pot: level 1


pot 11.1
Base 11.1
Package 11.1
Custom 11.1
Level 0
pot A
Base 11.1
Package A
Custom A
Level 1
pot B
Base 11.1
Package B
Custom B
Level 1

CL workflow

```
# pot init
# pot create-base -r 11.1
```

Download of FreeBSD 11.1

Create base 11.1 datasets

Create pot base-11_1

```
# pot create -p A -b 11.1
# pot create -p B -b 11.1
# pot start A
```

Mounts ZFS datasets via nullfs(5)

Starts the jail

```
# pot stop A
```

Stop the jail

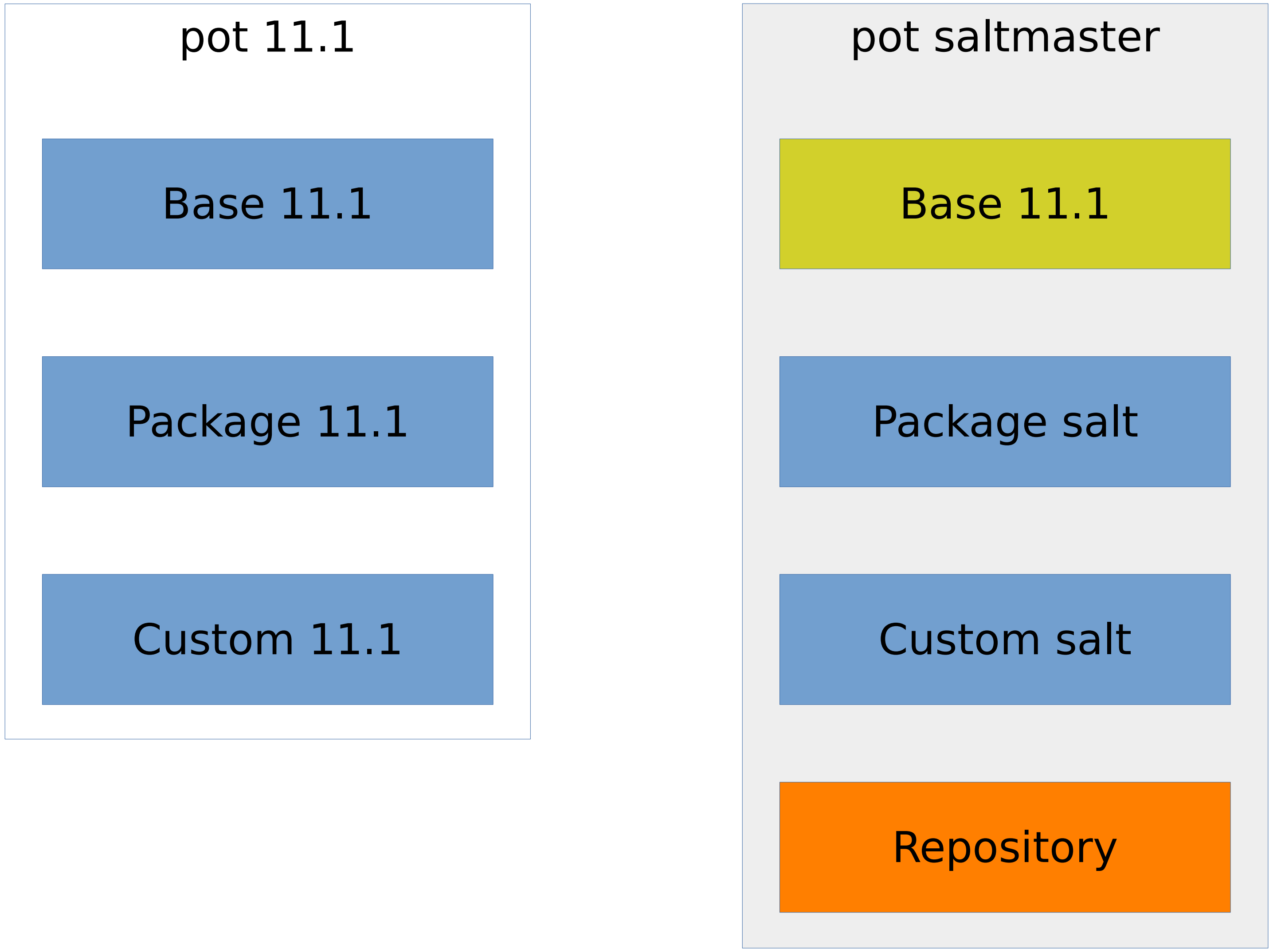
Unmounts ZFS datasets

File system components

File system components as building blocks

- Mandatory
  - Base
  - Package
  - Customization
- Whatever you need
  - Code repository
  - Databases
  - Caches
  - ...

Example: saltmaster

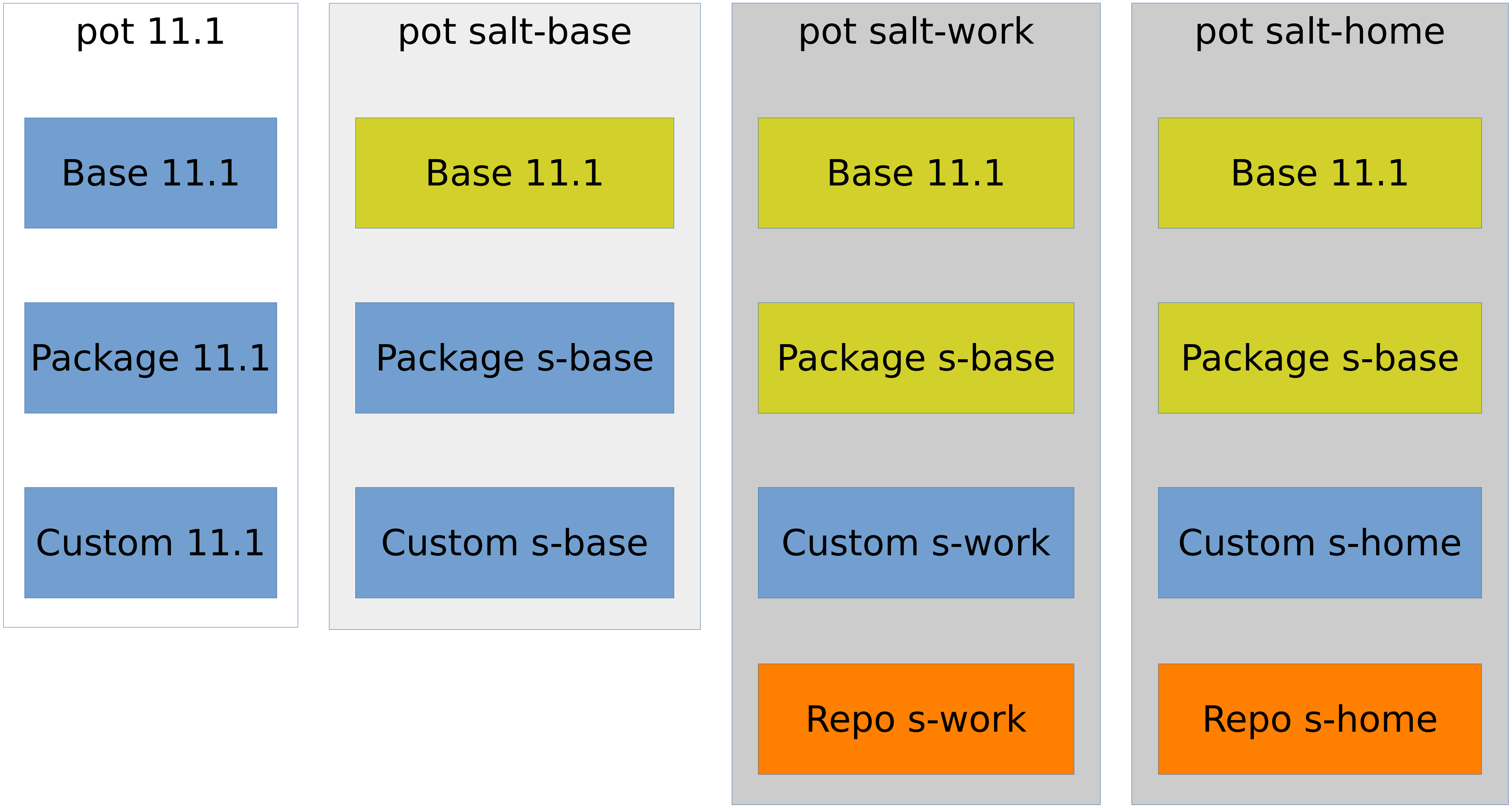


CL workflow

```
# pot init
# pot create-base -r 11.1
# pot create-fscomp -f repository
# pot create -p saltmaster -b 11.1
# pot add-fscomp -p saltmaster \
                 -f repository \
                 -m /mnt
```

Pasta analogy [2/2]

pot: level 2


pot 11.1
Base 11.1
Package 11.1
Custom 11.1
pot salt-base
Base 11.1
Package s-base
Custom s-base
pot salt-work
Base 11.1
Package s-base
Custom s-work
Repo s-work
pot salt-home
Base 11.1
Package s-base
Custom s-home
Repo s-home

CL workflow

```
# pot init
# pot create-base -r 11.1
# pot create-fscomp -f repo-work
# pot create-fscomp -f repo-home
# pot create -p salt-base -b 11.1
# pot create -p salt-work -P salt-base -l 2
# pot create -p salt-home -P salt-base -l 2
# pot add-fscomp -p salt-work -f repo-work -m /mnt
# pot add-fscomp -p salt-home -f repo-home -m /mnt
```

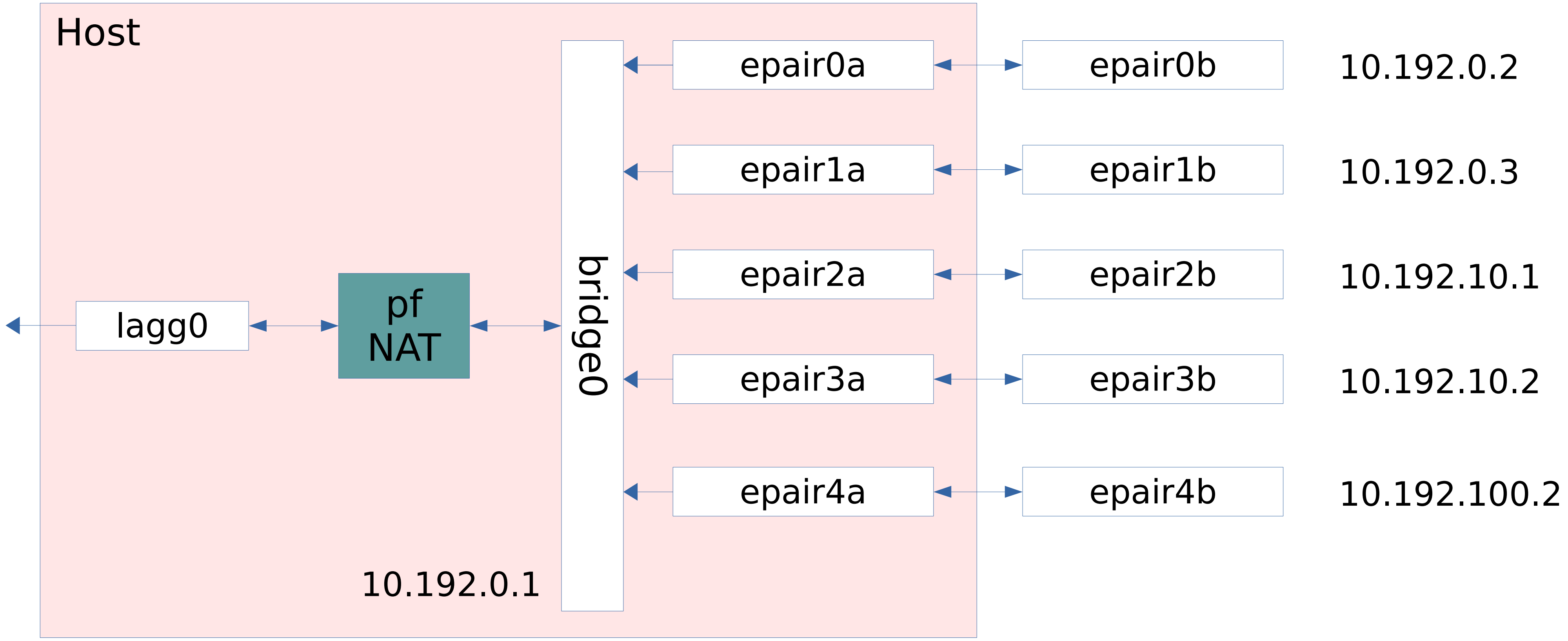
Network

Two network configurations available:

- **Inherit**
  - **Inherit the network stack of the host**
- **static IP in the internal virtual network**
  - **Exploits VNET(9) (kernel manually rebuilt)**
  - **NAT supported by pf(4)**
  - **the physical network interface as default gateway**
  - **all network interfaces are on the same bridge**

Internal virtual network

Network: 10.192.0.0/10



Network: missing features

- Add support to static IP without NAT
  - As currently provided by jails
- SHCP: Static DHCP
  - Currently, IP addresses have to be manually specified
  - SHCP would be a tool to provide valid static IP addresses
- Expose network services
  - A special dns pot running dnsmasq and consul
  - Network services registration to consul
  - haproxy running in the host can redirect request to the right pot using the information provided by the dns pot

pot is ZFS!

A pot is a bunch of ZFS datasets!

- **zfs snapshot => pot snapshot**
- **zfs rollback => pot rollback**
- **zfs clone => pot clone**
- **zfs rename => pot rename**

Work in progress

- **zfs promote => pot promote**

Pot flavor

Two kinds of flavors

- **A typical shell script, executed inside the container**
  - **Ideal for provisioning**
  - **A default flavor is also available**
- **A set of pot commands, to enrich the pot configuration**
  - **Ideal to attach file system components**
  - **Possibility to enforce priority between pots**

Pot flavor

Imitating poudriere(8)

```
# pot create -p builder -b 11.1 -f buildport
## buildport
  add-fscomp -f svnport -m /usr/ports
  add-fscomp -f distfiles -m /usr/ports/distfiles
  add-fscomp -f ccache -m /mnt
## buildport.sh
  #!/bin/sh
  pkg install -y ccache
  pkg clean -ayq
  echo "setenv CCACHE_DIR /mnt" >> /root/.cshrc
```

pot add-dep : Runtime dependency

Add dynamic dependencies between container

Example: salt-test needs saltmaster

- salt-test is the client
- saltmaster is the server
- `pot add-dep -p salt-test -P saltmaster`
- `pot start salt-test`
  - **saltmaster will start automatically**
  - **saltmaster will start first**
  - **Then, salt-test will start**

Resource limitation: cpuset(1)

Limiting CPU usage

- **Statically assign a pot to one or more CPUs**

```
# pot set-rss -p pot -C 0,2
```

Implemented via cpuset(1)

- **Applied immediately after the start of the jail**

Possible improvement

- **Set the number of CPUs wanted**
  - **During the start, a static allocation is performed that balance the load between CPUs**

Resource limitation: rctl(8)

- **rctl(8) is a relatively new resource limitation framework implemented in FreeBSD 9, but not enabled by default**
- **To be enabled at boot time via `kern.racct.enable=1` in `/boot/loader.conf`**
- **Used to show used resources and set specific limits**

Resource limitation: rctl(8) memoryuse

To limit the physical memory used by a pot

- How much?
- If the limit is reached, what happen?
  - Out of memory?
  - Soft limit?

Example: pot saltmaster

- Physical memory used: 430MB
  - pot show is the command showing the resource used by a pot

Resource limitation: rctl(8) memoryuse

Physical memory used: 430MB

- **Limit 400MB → still working, memory 400MB**
- **Limit 200MB → still working, memory ~200MB, sometimes above**
- **Limit 50MB → still working, memory ~52MB, often above**
- **Limit 10MB → still working, memory ~11MB, often a lot above the limit**

The memory limit reduce the RSS of a process to fit the constraint

The processes “working set” are drastically reduced

Possible big performance penalty

Resource limitation: rctl(8) pcpu

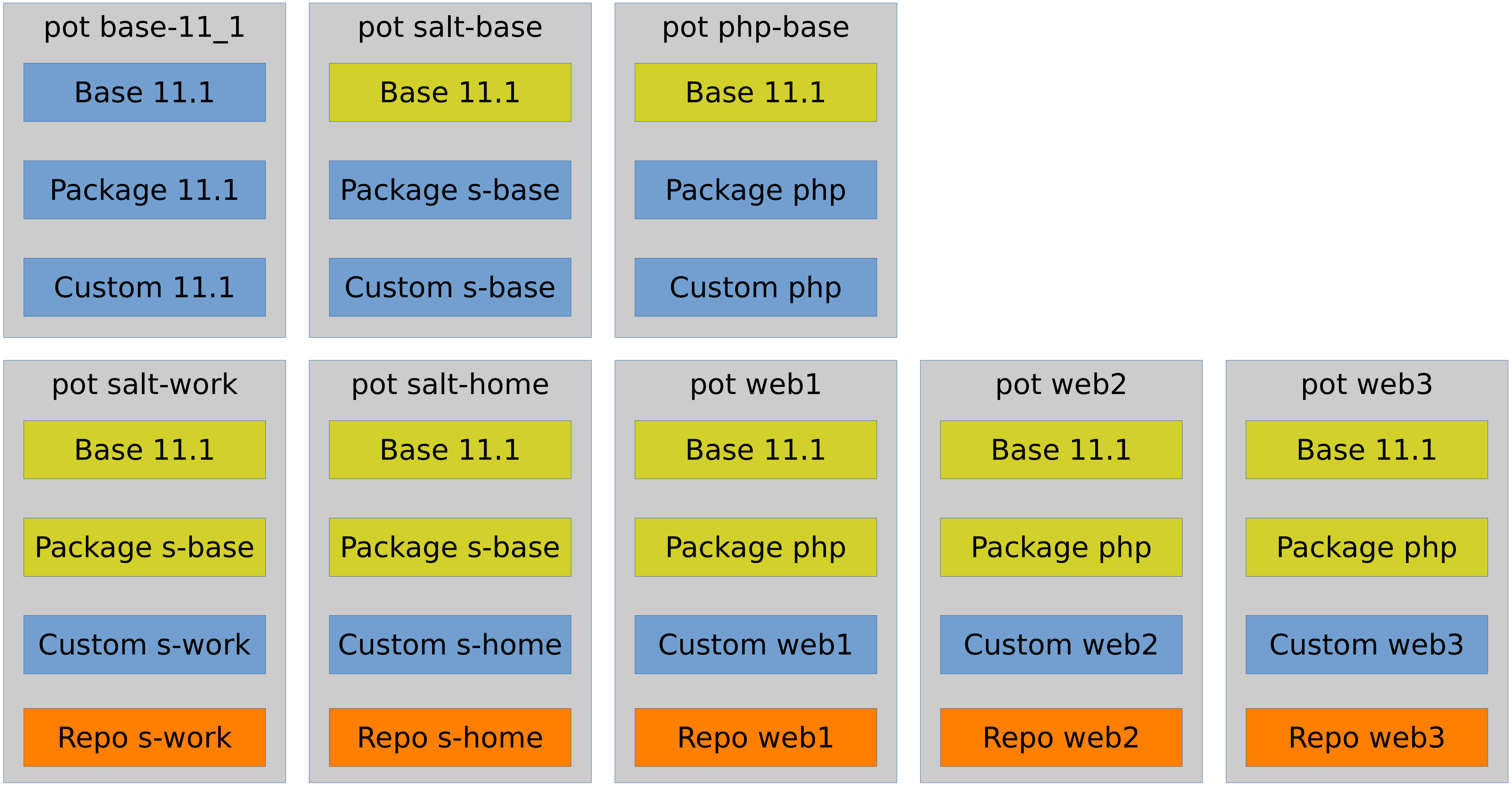
To limit the cpu percentage used by a pot

- I wasn’t able to find a proper setup
  - pcpu counter in kernel space has an odd behavior
    - 20k % of CPU usage?
- To enforce the CPU% limits, the processes are simply blocked
  - Delay of seconds observed, causing timeouts to expire

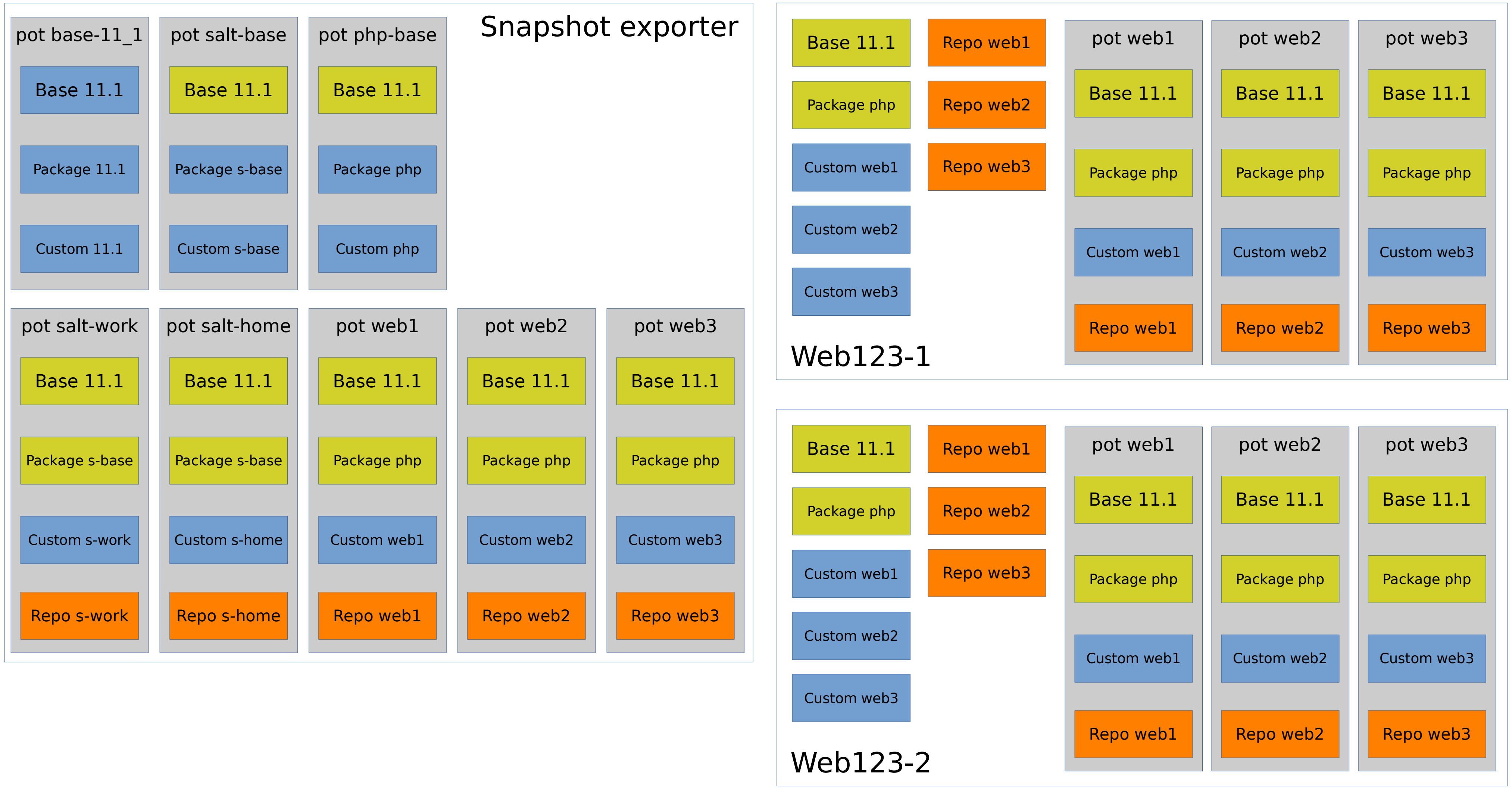
Not adopted in pot and probably it won’t in the future

Moonshot : the big picture

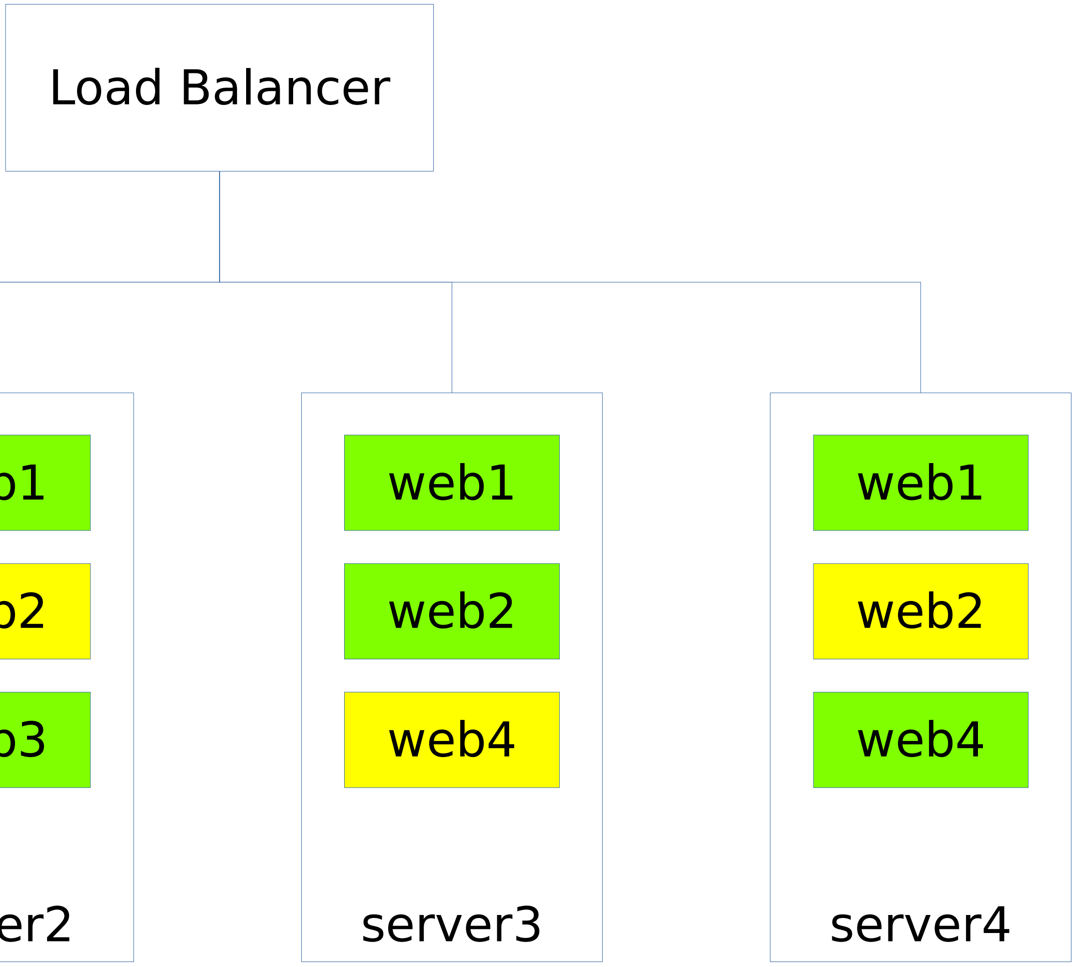
pot migration : a look to the future



pot migration : a look to the future



Orchestration?



Conclusion → TILs

- pot is a possible implementation of a container model entirely based on FreeBSD
- The project is on github
  - `https://github.com/pizzamig/pot`
  - Fork it and submit pull requests
  - Submit issues (it’s still full of bugs, help!)
- TIL: containers cannot be better than the host Operating System

Thanks!

Thanks a lot!

Questions?

Contributions

[1] pot logo

Daniela Spoto

https://danielaspoto.wixsite.com/illustrations

[2] Pasta

Junya Ogura

https://www.flickr.com/photos/sooey/5089711764

[3] spaghetti carbonara

Martin Krolikowski

https://www.flickr.com/photos/martinkrolikowski/6302915547

[4] Pici with ragù

Luca Nebuloni

https://www.flickr.com/photos/nebulux/8524965788

[5] The Moonshot

Diego Torres Silvestre

https://www.flickr.com/photos/3336/6039485059