

Jail

frank

What is jail?

Lightweight container that isolate processes/
access to file system/network stack

- Introduced in FreeBSD 4.X (March, 2000)
- Build upon the concept of chroot
- Operating-system-level virtualization

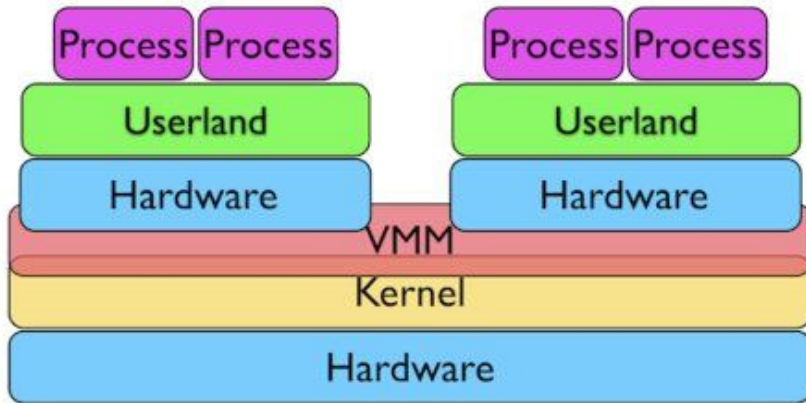


Jail v.s. chroot

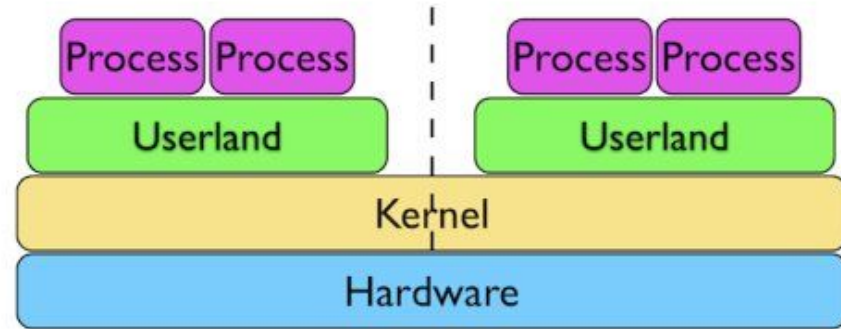
Jails differs from chroot by the level of separation:

Chroot limits scope of filesystem for the process
While Jail isolates processes from the rest of system

Jail v.s. VM



VM



Jail

Jail: Advantage

Jail

- Easy to deploy and maintain
- Boot up fast (within seconds, while VM takes minutes)
- Much less overhead (Disk I/O, emulated NIC, CPU)

How to create a jail?

A jail is characterized by four elements:

- Path: the starting point from which jail is entered
- Hostname: Hostname used by jail
- IP address: IP assigned to the jail
- Command: the path name of executable to run inside the jail

How to create a jail?

Basic Jail Environment:

/myjail: place to put our jails, the “*Path*” of jail

/myjail/bin, /myjail/sbin: basic tools

/myjail/lib,/myjail/libexec: library dependencies

/myjail/dev: devfs

```
# jail -c jid=1 name=myjail path=/myjail mount.devfs \  
  ip4.addr="127.0.0.1" host.hostname=myjail \  
  command="/bin/sh"
```

How to create a jail?

There must be at least one process attached to specific jail. Otherwise, set *persist* parameter.

```
# jls // list running jails
```


When will a jail be destroyed?

Jail will be destroyed when no process is running in jail, unless *“persist”* is specified.

Jails with “persist” propertie may be destroyed manually using “jail -r <JID/Name>”

Create a jail with freebsd world

Download pre-built base system

```
# fetch
```

```
“http://ftp.tw.freebsd.org/pub/releases/amd64/10.1-RELEASE/base.txz”
```

```
# mkdir /myjail // jail path
```

```
# tar -xf base.txz -C /myjail // extract base into jail path
```

```
# freebsd-update -b /myjail fetch install // upgrade base system
```

Create a jail with freebsd world

```
# jail -c name=myjail path=/myjail \  
ip4.addr=vmx0|127.0.0.10/24 mount.devfs \  
host.hostname=myjail.urcrazy.net \  
exec.start="sh /etc/rc" \  
exec.stop="sh /etc/rc.shutdown"
```

Managing Jails

Start jails upon system boot:

In rc.conf: jail_enable="YES"

Saving jail configurations in /etc/jail.conf:

```
myjail {  
    path=/myjail;  
    mount.devfs;  
    host.hostname=myjail.urcrazy.net;  
    exec.start="sh /etc/rc";  
    exec.shutdown="sh /etc/rc.shutdown";  
    ip4.addr="127.0.0.1";  
}
```

Managing Jails

Start/Stop all jails:

```
# service jail start/stop
```

Start/Stop specific jails:

```
# service jail start/stop <jail name>
```

Managing multiple Jails

per jail base system:

- waste of space
- adds complexity to system administration

Solution?

Shared-base-system Jails

Managing multiple Jails

Common Implementation:

- Using unionfs
 - easy to implement
 - caveats!
- Using ZFS (recommended)
 - easy to backup/snapshot/clone

Managing multiple Jails using UnionFS

```
# mkdir -p /jail/basejail
```

```
# tar -xf base.txz -C /jail/basejail
```

```
# freebsd-update -b /jail/basejail fetch install
```

for each jail: create directory

```
/jail/${jail_name} and /jail/${jail_name}_rw
```


Managing multiple Jails using UnionFS

for each jail: create file

`/jail/fstab.${jail_name}` with content below

```
/jail/basejail /jail/${jail_name} nullfs ro 0 0
```

```
/jail/${jail_name}_rw unionfs rw,noatime 0 0
```

Managing multiple Jails using UnionFS

Add jail configuration to `/etc/jail.conf`:

```
jail01 {  
    path=/jail/jail01;  
    mount.devfs;  
    mount.fstab=/jail/fstab.jail01;  
    devfs_ruleset=4;  
    host.hostname=jail01.urcrazy.net;  
    exec.start="sh /etc/rc";  
    exec.shutdown="sh /etc/rc.shutdown";  
    ip4.addr="127.0.0.1";  
}
```

```
jail02 {  
    path=/jail/jail02;  
    mount.devfs;  
    mount.fstab=/jail/fstab.jail01;  
    devfs_ruleset=4;  
    host.hostname=jail02.urcrazy.net;  
    exec.start="sh /etc/rc";  
    exec.shutdown="sh /etc/rc.shutdown";  
    ip4.addr="127.0.0.2";  
}
```

Managing multiple Jails using UnionFS

Further simplify jail.conf using variables and wildcard

```
* {
    path=/jail/$name;
    mount.devfs;
    mount.fstab=/jail/fstab.$name;
    devfs_ruleset=4;
    host.hostname=$name.urcrazy.net;
    exec.start="sh /etc/rc";
    exec.shutdown="sh /etc/rc.shutdown";
}

jail01 {
    ip4.addr="127.0.0.1";
}

jail02 {
    ip4.addr="127.0.0.2";
}
```

Managing multiple Jails using UnionFS

Caveaaaat!!!

The mounting of devfs **WILL FAIL** using jail.conf due to the mounting sequence of unionfs and devfs

FYI:

<https://forums.freebsd.org/threads/freebsd-10-jail-on-nullfs-unionfs-devfs-causing-pain.46610/>

PR: <http://www.freebsd.org/cgi/query-pr.cgi?pr=186360>

Managing multiple Jails using UnionFS

Temporary Solutions:

1. adding `exec.prestart="ls /jail/$name/dev";` to `jail.conf` (recommended)
2. mount devfs to each jail using `rc.local`
3. manually mount devfs (bad idea)

Managing multiple Jails using UnionFS

Recap

To create new jail:

1. create jail and jail_rw directory
2. mount read-only basejail to jail directory
3. mount jail_rw to jail using unionfs
4. do 2 and 3 with individual fstab file
5. add jail configuration into jail.conf

Managing multiple Jails using ZFS

Separate read-only and read-write areas

Read-write area:

- /usr/local
- /usr/X11R6
- /usr/home or /home
- /tmp
- /var
- /root
- /usr/ports/distfiles

Managing multiple Jails using ZFS

```
# zfs create -o mountpoint=/jail zroot/jail
```

```
# zfs create zroot/jail/basejail
```

```
# zfs create zroot/jail/skel
```

```
# tar -xf base.txz -C /jail/basejail
```

```
# freebsd-update -b /jail/basejail fetch install
```


Managing multiple Jails using ZFS

```
# cd /jail/basejail
```

```
# mv etc tmp var root ../skel
```

```
# mv usr/local ../skel/usr-local
```

```
# mkdir s
```

Managing multiple Jails using ZFS

```
# ln -s /s/etc etc
```

```
# ln -s /s/home home
```

```
# ln -s /s/root root
```

```
# ln -s /s/usr-local usr-local
```

```
# ln -s /s/distfiles usr/ports/distfiles
```

```
# ln -s /s/tmp tmp
```

```
# ln -s /s/var var
```

Managing multiple Jails using ZFS

```
# cd ../skel
```

```
# mkdir home usr-X11R6 distfiles
```

Managing multiple Jails using ZFS

for each jails:

create directory

`/jail/${jail_name}` and `/jail/${jail_name}_rw`

copy file in `/jail/skel` into `/jail/${jail_name}_rw`

`cpdup /jail/skel/ /jail/${jail_name}_rw`

Managing multiple Jails using ZFS

create individual fstab file `fstab.${jail_name}`

```
/jail/basejail /jail/${jail_name} nullfs ro 0 0
```

```
/jail/${jail_name}_rw /jail/${jail_name}/s nullfs rw 0 0
```

Managing multiple Jails using ZFS

add jail configuration to /etc/jail.conf
following steps in [p.19](#)

Managing multiple Jails using ZFS

Recap

To create new jail:

1. create jail and jail_rw directory
2. copy /jail/skel/ into jail_rw directory
3. mount read-only basejail to jail directory
4. mount jail_rw to jail/s
5. do 2 and 3 with individual fstab file
6. add jail configuration into jail.conf

Managing Jail using tools

There are tools designed to aid jails managements

- qjail
- ezjail

Managing Jail using ezJail

```
# pkg install ezjail
```

```
# ezjail-admin install -p
```

To create jail

```
# ezjail-admin create <jail name> <jail ip>
```

To list jail

```
# ezjail-admin list
```

Managing Jail using ezJail

To enter jail console:

```
# ezjail-admin console <jail name>
```

To create archive for jail

```
# ezjail-admin archive <jail name>
```

To deploy jail from archive

```
# ezjail-admin create -a <archive>
```

Jail Networking

Using NAT

```
# ifconfig lo1 create
```

```
jail1 ip: lo1|10.0.0.1/24
```

```
jail2 ip: lo1|10.0.0.2/24
```

Jail Networking

Using NAT

pf.conf:

```
ext_if="vmx0"
```

```
jail_if="lo1"
```

```
nat on $ext_if from $jail_if to any -> $ext_if
```

Jail Networking

**Ping command returned
Operation not permitted**

Solution:

```
allow.raw_sockets = 1;
```

Fine tuning Jail

Jail has many restrictions, some of them can be tuned via setting `sysctl` and `jail.conf` parameters.

Make sure you know exactly what you're doing

```
# sysctl -d security.jail.param
```

Advanced Topic - Resource limiting

Assign processors to specific jail:

```
# cpuset -j <jailid> ....
```

Advanced Topic - Resource limiting using RACCT/RCTL

Recompile kernel with:

options RACCT

options RCTL

Advanced Topic - Resource limiting using RACCT/RCTL

Add/Remove rule:

```
rctl -a <rule>
```

Rule Syntax:

```
subject:subject-id:resource:action=amount/per
```

Advanced Topic - Resource limiting using RACCT/RCTL

subject

process, user, loginclass or **jail**

subject-id

process id, user id, login class name, **jail name**

Advanced Topic - Resource limiting using RACCT/RCTL

resources

cputime, datasize, stacksize, maxporc
vmemoryuse, swapuse, pcpu.....

action

deny, log, devctl, sig*, throttle

Advanced Topic - Resource limiting using RACCT/RCTL

Add/Remove rule:

```
rctl -a <rule>
```

Rule Syntax:

```
subject:subject-id:resource:action=amount/per
```

Advanced Topic - VNET

Reference:

<https://wiki.freebsd.org/VIMAGE>