



# Jails: Como encarcerar *Userlands* em FreeBSD

**Francisco Alves Cabrita tcc include**  
include@npf.pt.freebsd.org  
(versão 1.1 de 30 de Novembro de 2006)

## Resumo

Neste documento escrevo sobre o "mundo fantástico das **Jails**": o que são, para que servem e como as implementar.

Este "Como Fazer (CF)" destina-se aos utilizadores de FreeBSD 5.X, no entanto; a partir do FreeBSD 4.X é possível construir **Jails** com a mesma simplicidade. Boa leitura.

## 1 Introdução

As **Jails** foram criadas por Poul-Henning Kamp para a R&D Associates e daí introduzidas no FreeBSD 4.0. Robert Watson esteve envolvido na documentação e na elaboração de alguns "re-mendos".

### 1.1 O que são Jails?

A tradução à letra de **Jail** recai sobre o termo "prisão", mas para uma melhor adaptação do termo ao tema que aqui trago, podemos também correctamente falar sobre encarceramento da área de utilizadores, ou seja, criar *Userlands* encarceradas.

Como é sabido, o FreeBSD é composto por duas áreas distintas, o núcleo *Kernelland* e área de utlizadores *Userland*; é portanto sobre a criação de novas áreas de utilizadores que falaremos neste CF.

As **Jails** são ambientes auto-contidos que partilham recursos (" *CPU / Memória / Sistema de ficheiros /etc* ") com o Anfitrião dessa mesma máquina, mas que possuem o seu próprio endereço IP, configurações e programas. Podemos chamar às **Jails** Servidores Virtuais, pois estas vão disponibilizar serviços como um servidor nativo o faria, mas de forma isolada.

O Anfitrião é aquele que "segura" as prisões, este comunica com os serviços contidos nessas por diversas vias (IP, IPC, Raw Sockets, etc) dependendo da configuração. No presente CF iremos ver algumas destas vias em detalhe.

## 1.2 Para que servem as Jails ?

Numa primeira e directa abordagem, estas prisões foram criadas a pensar em segurança (de que falarei adiante); no entanto, podemos encaixá-las noutro tipo de necessidades que os administradores de sistemas encontram diariamente, como por exemplo:

- Recursos Físicos - Poucos recursos físicos (máquinas) para alojar separadamente serviços.

Normalmente um simples utilizador em casa não tem o material que desejaria para poder alocar uma maquina para cada serviço: uma para o servidor de mail, outra para web, uma outra apenas para um motor de base de dados, e por aí adiante<sup>1</sup>, daí que o uso deste sistema se pode tornar um aliado na redução de custos e recursos, no entanto não só em casa encontramos vantagens para usar **Jails**; hoje cada vez mais, "Internet Service Providers, Shell Providers", PME's, entre outros, estão cada vez mais a apostar em FreeBSD e nesta tecnologia.

- Ambientes Isolados - Necessidade de criar ambientes específicos que não entrem em conflito com outros serviços a correr na mesma máquina.

Vários administradores ficam mais descansados se, por exemplo, implementarem dois servidores web distintos, um com suporte para PHP (mod\_php) e outro com suporte para Perl (mod\_perl) (*é claro que se pode compilar um Apache com suporte a estas duas linguagens estática ou dinamicamente*).

Poderíamos descrever aqui inúmeros exemplos para justificar o uso destes ambientes isolados, como dar privilégios de `root` ao Administrador de Base de Dados, para administrar a sua prisão onde correm os devidos motores, (**MySQL, PostgreSQL, etc**).

Um outro exemplo seria, termos a correr projectos distintos no mesmo servidor sem que as Equipas/Projectos se tocassem ou sequer soubessem da existência uns dos outros no mesmo servidor.

- Laboratório - Criar aquilo a que eu chamo "zona de destruição", que como o nome diz são **Jails** para "partir" com testes.

Pegando outra vez na falta de recursos, ou mesmo a criação de ambientes isolados para atingir maiores níveis de segurança e independência sobre o sistema Anfitrião, porque não pensar em ter um laboratório virtual.

Instalar uma nova instância demora apenas 5 minutos.

Desta forma, neste ambiente, podemos elaborar todo o tipo de testes e avaliar o seu resultado sem que afectemos o servidor Anfitrião ou as restantes **Jails**/projectos em produção no mesmo.

Como exemplos finais dou um caso prático usado por muitos "Hosting Providers" e outro que uso quer em casa como no trabalho:

---

<sup>1</sup>Atenção, no caso dos IDS's - (Sistemas de Detecção de Intrusos), aprisionando-o não obtemos resultados sobre a rede, mas apenas dessa prisão, visto que este nestas condições só é capaz de auscultar o próprio IP.

- *Hosting Providers* - Disponibilizam *shells* com acesso a máquinas virtuais.

Estes, usando **Jails**, estão a alugar mais do que um simples `/home/user_xpto` mas sim toda uma máquina com a sua administração e acesso a `root`, claro que sempre por baixo de uma administração mais poderosa que é a do próprio `root` do servidor Anfitrião.

- Construção de Ports - Uma prisão exclusiva para construir pacotes e a partir dessa distribuí-los para as restantes prisões na mesma ou numa outra máquina.

A utilização duma plataforma deste tipo dá-nos a capacidade de parameterizar um "*Cluster*" capaz de construir de forma distribuída e segura pacotes previamente escolhidos, que de seguida serão também distribuídos por servidores e **Jails**.

Tal método permite-nos eliminar a redundância ao compilar várias vezes os mesmos ports em servidores/**Jails** diferentes.

## 2 Jails

### 2.1 Criação de Jails

Depois de termos um `Kernel` correctamente configurado e instalado podemos dar início à instalação das nossas "prisões"; para isso temos que criar os respectivos directórios para onde vamos instalar as **Jails**. Neste caso, `node1` e `node2` serão as nossas duas **Jails** em causa:

```
$ mkdir /usr/jails
$ mkdir /usr/jails/node1
$ mkdir /usr/jails/node2
```

E iniciamos a criação e instalação da **Jail**, o alerta aqui vai para dois pontos:

1. Caso tenhamos eliminado os objectos `/usr/obj` no final da instalação da *Userland* (*world*, teremos que executar não um `installworld` mas sim um `buildworld`, ou na substituição dos dois apenas `world` sem nenhuma outra palavra concatenada a esta.
2. O outro alerta vai para o parâmetro `DESTDIR`, que é de extrema importância e cujo uso é imperativo na criação de **Jails**; caso não o usemos iremos reconstruir um sistema em cima do sistema Anfitrião, o que resultará na perda de configurações (como por exemplo os grupos de utilizadores do sistema registados em `/etc/groups`).

```
$ cd /usr/src
$ make installworld DESTDIR=/usr/jails/node1
$ make installworld DESTDIR=/usr/jails/node2
$ cd etc
$ make distribution DESTDIR=/usr/jails/node1
$ make distribution DESTDIR=/usr/jails/node2
```

Assim terminamos a instalação das **Jails**.

## 2.2 Anfitrião: Configurações relativas às Jails

Depois de termos as **Jails** instaladas temos que informar o Anfitrião da existência das mesmas, desta forma precisamos de criar aliases de IP para cada uma das **Jail** e, de seguida, apenas com um simples ficheiro (`/etc/rc.conf.local`), configurar onde e como estas serão executadas.

Criação de aliases de IP.

Em `/etc/rc.conf` adicionamos, no final do ficheiro:

```
ifconfig_bfe0_alias0="inet 10.1.1.55 netmask 0xffffffff" #node1
ifconfig_bfe0_alias1="inet 10.1.1.51 netmask 0xffffffff" #node2
```

**Nota 1)** Na linha acima `bfe0` é o interface de rede para a placa de rede em causa. No seu caso poderá ser outro, verifique com o comando `ifconfig` ou nas mensagens do sistema (`/var/log/messages`) qual o nome do interface e substitua na linha acima. (Exemplo: `ifconfig_NOME_INTERFACE_alias0="inet IP netmask 0xffffffff"`)

**Nota 2)** Verão que em todo o documento, usei IPs da classe A (`10.1.1.X`), para comunicar com as **Jails**, no entanto pode também preferir que as ditas **Jails** apenas sejam acedidas pela própria máquina em causa, para isso reservamos aliases de IP locais através do Interface de Rede Local *"loopback"*. As vantagens deste método são várias.

Exemplo: Quando pensarmos ter um serviço como o **MySQL** a correr numa **Jail**, este só poderá ser interrogado por serviços a correr na própria máquina, quer sejam **Jails** ou não, ficando assim completamente isolado do "mundo exterior".

O processo é o mesmo que o exemplo acima, vejamos como ficaria o nosso `/etc/rc.conf`:

```
ifconig_lo0_alias0="inet 127.0.0.2 netmask 0xffffffff" #node1
ifconig_lo0_alias1="inet 127.0.0.3 netmask 0xffffffff" #node1
```

Repetimos o processo caso sejam criadas mais **Jails**.

Entretanto, as instruções a adicionar ao ficheiro `/etc/rc.conf.local` serão:

```
jail_enable="YES"
jail_set_hostname_allow="NO"
jail_socket_unixiproute_only="YES"
jail_sysvipc_allow="YES"
jail_stop_jailer="NO"
jail_list="node1 NOME_DA_JAIL"

jail_node1_rootdir="/usr/jails/node1"
jail_node1_hostname="node1"
jail_node1_ip="10.1.1.55"
```

```
jail_node1_exec="/bin/sh /etc/rc"  
jail_node1_devfs_enable="YES"  
jail_node1_procfs_enable="YES"
```

O último bloco de configurações é repetido para cada `jail` a inicializar, alterando `node1` por `NOME_DA_JAIL` e respectivo directório/IP.

Note que, em `jail_list="node1 NOME_DA_JAIL"`, deverá adicionar consecutivamente o nome da `jail` que pretende ver iniciada no arranque do sistema, neste caso já temos lá listada a `jail NOME_DA_JAIL`.

Alguns serviços necessitam privilégios especiais para poderem ser executados dentro de sistemas enclausurados, como o caso do **PostgreSQL**. Desta forma, devemos parameterizar o Anfitrião de que estes estão autorizados a usar instruções `sysvipc` (neste caso). Para isso basta:

```
$ echo "security.jail.sysvipc_allowed=1" >> /etc/sysctl.conf
```

Adiante iremos abordar em maior detalhe este tema "Segurança" através das "System Calls" `-(sysctl)`

## 2.3 Configurar as Jails

Na configuração de uma `jail` teremos que efectuar as seguintes manobras:

- Criar um ficheiro `fstab` vazio para cada **Jail**  

```
$ touch /usr/jails/node1/etc/fstab  
$ touch /usr/jails/node2/etc/fstab
```
- Configurar a resolução de nomes para cada uma das **Jail**, em `/usr/jails/NOME_DA_JAIL/etc/resolv.conf` inserir `nameserver ip_servidor_de_dns`.
- Junto do `rc.conf` de cada `jail`, residente em `/usr/jails/NOME_DA_JAIL/etc/rc.conf` devemos verificar os seguintes parâmetros:
  1. Alterar o nome do `hostname`
  2. Adicionar `network_interfaces=""` desta forma desabilitamos no arranque, a configuração e alertas dadas pelo `ifconfig(8)`.
  3. Activar o serviço de SSH (caso pretendido)  

```
sshd_enable="YES"
```
  4. Definir `rpcbind_enable="NO"`
  5. Desligar serviços de e-mail `sendmail_enable="NONE"`
  6. Adiante veremos como configurar o "timezone" entre outras coisas, assim que tenhamos a **Jail** a correr.

- Forçar o serviço SSH a escutar em apenas um IP. Para tal, o procedimento será o mesmo para cada **Jail**:

```
Editar: /usr/jails/node1/etc/ssh/sshd_config
```

```
Rectificar: ListenAddress IP_DA_JAIL
```

**Nota:** Devemos ter especial atenção aos "binds" de IPs para cada serviço/**Jail**, pois a solução de um problema no arranque de algum deles (p.e. MySQL, Apache, OpenLDAP), pode passar por este ítem.

- Criar o directório `ports` para cada **Jail**:  

```
$ mkdir /usr/jails/NOME_DA_JAIL/usr/ports
```
- Criar o directório `BUILD` em `/tmp` também para cada `jail`:  

```
$ mkdir /usr/jails/NOME_DA_JAIL/tmp/BUILD
```
- Criar um `make.conf` para cada uma, com a indicação onde serão construídos os pacotes a instalar:  

```
$ touch /usr/jails/NOME_DA_JAIL/etc/make.conf
$ cd /usr/jails/NOME_DA_JAIL
$ echo "WRKDIRPREFIX=/tmp/BUILD" > etc/make.conf
```
- Montar o `devfs` dentro de cada `jail`:  

```
$ mount_devfs devfs /usr/jails/NOME_DA_JAIL/dev
```
- Ligar o dispositivo `null` ao núcleo residente na `jail`:  

```
$ cd /usr/jail/NOME_DA_JAIL
$ ln -sf dev/null kernel
```

O final das configurações termina com o momento em que entramos nas **Jails** para criar um utilizador; as configurações gerais ainda não terminam aqui, mas com este utilizador ficamos capazes de aceder às **Jails** via `ssh`, como veremos adiante.

(Exemplo: `$jail /usr/jails/NOME_DA_JAIL/ JAIL_HOSTNAME IP_JAIL /bin/sh`)

```
$ jail /usr/jails/node1/ node1 10.1.1.55 /bin/sh
```

**Nota:** Para usar o `JAIL_HOSTNAME` deverá ter adicionado o nome/IP de cada `jail` ao `/etc/hosts` do Anfitrião.

Entretanto, já dentro da **Jail** os 3 primeiros e mais importantes passos são:

1. O primeiro passo será definir uma palavra-passe para o administrador do sistema (`root`), de preferência diferente da palavra-passe do Anfitrião.

2. De seguida, executando o comando `adduser` adicionamos um utilizador; repetimos o processo para cada **Jail**.
3. Por fim, para as restantes configurações (como acima alertei que iria falar), podemos executar o nosso conhecido `/usr/sbin/sysinstall` dentro da **Jail**. Desta forma configurar o "timezone" da máquina entre outras coisas como criar utilizadores e seus grupos torna-se mais fácil ainda.

Para sair de cada **Jail** bastará executar o comando `exit`.

De seguida inicializamos as **Jails**.

```
$ cd /etc/rc.d
$ ./jail start
Configuring jails:.
Starting jails: node1 node2
```

O comando seguinte, lista as **Jails** activas no sistema.

```
$ jls
JID IP Address  Hostname  Path
2   10.1.1.55   node1     /usr/jails/node1
1   10.1.1.51   node2     /usr/jails/node2
```

## 3 Outras Configurações

### 3.1 Segurança, `sysctl`'s

Veremos agora como e quais parametros (MIB's) - "*Management Information Base*" usar para alterar o estado do núcleo do sistema e o tratamento dado por este às **Jails**.

Presentemente a alteração destas MIBs propaga-se a todas as **Jails** existentes no sistema, no futuro esta funcionalidade será afinada para que possa haver maior distinção entre cada **Jail** e suas configurações.

Para receber a listagem de todas as "MIBs" usadas pelo núcleo basta executar como `root` o comando: `sysctl -a | more`

No nosso caso iremos só pesquisar as que interagem directamente com o mundo das **Jails**.

```
$ sysctl -a | grep 'jail'
security.jail.set_hostname_allowed: 0
security.jail.socket_unixiproute_only: 1
security.jail.sysvipc_allowed: 1
```

```
security.jail.getfsstatroot_only: 1
security.jail.allow_raw_sockets: 0
security.jail.chflags_allowed: 0
security.jail.jailed: 0
```

Resumidamente, estas MIBs significam:

- `security.jail.allow_raw_sockets`  
Esta entrada determina se o `root` é capaz ou não de criar *"raw sockets"*. Activando esta MIB, irá permitir que utilitários como `ping` e `traceroute` operem dentro das **Jails**.
- `security.jail.getfsstatroot_only`  
Define se dentro de uma **Jail**, os processos são capazes de ver ou não todos os *"mount-points"*.
- `security.jail.set_hostname_allowed`  
Esta MIB assegura se os processos dentro da **Jail** podem ou não alterar o nome do nó via comando `hostname`.
- `security.jail.socket_unixiproute_only`  
A tecnologia de enclausuramento permite auscultar um IP versão 4 em cada **Jail**, e limitar o acesso a outros endereços de rede no espaço IPv4 que possam estar disponíveis no ambiente. No entanto, presentemente a **Jail** não é capaz de limitar o acesso a outras pilhas de protocolos de rede que não tenham esta tecnologia de enclausuramento. Assim, por defeito, processos dentro de **Jails** apenas podem aceder a protocolos nos seguintes domínios: (`PF_LOCAL`, `PF_INET` e `PF_ROUTE`, permitindo-os aceder a sockets de UNIX, endereços IPv4 e sockets de roteamento).
- `security.jail.sysvipc_allowed`  
Com esta MIB definimos se os processos residentes nas **Jails** irão ter acesso a primitivas *"System V IPC"*.  
Como nota especial para esta MIB, dou como exemplo o caso de se ter uma instância de **PostgreSQL** a correr dentro de uma **Jail** o qual obriga a activação da MIB em causa.
- `security.jail.chflags_allowed`  
Esta entrada determina se um utilizador privilegiado dentro da **Jail** é tratado por `chflags(2)`. Se estiver a zero, este utilizador é tratado como não privilegiado não podendo alterar ficheiros com *"flags"* definidas.

Entretanto também temos as seguintes MIB's que devem ser alvo de atenção:

```
$ root@fac:/etc/rc.d$ sysctl -a | grep 'bsd.see_other'
security.bsd.see_other_uids: 1
security.bsd.see_other_gids: 1
```

Acima, como o próprio nome indica, cada utilizador ou grupo de utilizadores, só poderá ver os seus/grupo processos.

As considerações finais para este tema vão para:

- Uma MIB com o valor zero (0) estará inactiva, uma com o valor um (1) estará activa.
- Para que as "sysctl's" premanecam correctamente configuradas após reiniciarmos o servidor, deveremos lista-las em `/etc/sysctl.conf` da seguinte forma:

```
security.jail.sysvipac_allowed=1
```

## 3.2 Partilhar recursos: Anfitrião - Jails

Antes de acedermos às **Jails** vamos torná-las capazes de ler a Árvore de Ports do sistema Anfitrião, desta forma, partilhando recursos deste tipo, vamos poupar espaço em disco, tempo e possíveis dores de cabeça.

```
$ mount_nullfs /usr/ports /usr/jails/NOME_DA_JAIL/usr/ports
```

Tentaremos criar este "mount point" sempre que necessário, não o incluindo no `/usr/fstab` do sistema Anfitrião por questões de segurança (ou esquecimento na hora de eliminar alguma **Jail**).

### 3.2.1 Compatibilidade binária com Linux

Como é sabido, em FreeBSD podemos executar binários construídos para Linux. Por defeito esta compatibilidade não está activa no sistema e, por conseguinte, numa **Jail** não estará. Aqui iremos ver como de uma forma simples vamos activar essa funcionalidade:

```
$ cd /usr/ports/compat/linux_base
$ make install clean
$ mount_nullfs /compat/ /usr/jails/node1/compat/
$ mount_linprocfs linprocfs /usr/jails/node1/compat/linux/proc
```

Estes volumes deverão ser configurados no `/etc/fstab` de cada jail em causa para que possam ser estabelecidos no arranque do sistema; para isso basta adicionar as duas linhas seguintes ao ficheiro acima mencionado.

```
/compat /usr/jails/node1/compat          nullfs    rw  0  0
linproc /usr/jails/node1/compat/linux/proc linprocfs rw  0  0
```

A seguinte linha deverá ser adicionada quer no Anfitrião quer nas **Jails** que terão activado o suporte para binários linux:

```
$ echo linux_enable="YES" >> /etc/rc.conf
```

### 3.2.2 Actualizar as Jails

A actualização de cada prisão é em muito semelhante à actualização de um servidor, quer no que diz respeito a aplicações (porte) quer no que diz respeito à sua *Userland*. Neste CF iremos abordar apenas esta segunda componente.

Desta forma, depois de um típico `make update` e devidas compilações que poderá ver em detalhe em [CF\_makeconf - "Truques e valores uteis para o /etc/make.conf"], executamos:

```
$ cd /usr/src
$ make installworld DESTDIR=/usr/jails/node1
$ make installworld DESTDIR=/usr/jails/node2
$ mergemaster -D /usr/jails/node1
$ mergemaster -D /usr/jails/node2
$ cd /etc/rc.d
$ ./jail restart
```

**Nota:** Atenção para ficheiros como `/etc/groups` e `/etc/hosts` para que não sejam sobrepostos com ficheiros limpos "*default*" aquando a sua fusão "*merge*".

### 3.3 Desfazer todo o trabalho

Para desfazermos todo o trabalho (eliminar **Jails**), em primeiro lugar devemos efectuar as devidas cópias de segurança dos nossos dados e configurações.

De seguida desligamos a **Jail** e confirmamos se está algum volume `nullfs` ou outro montado para a nossa **Jail** (exemplo: `/usr/jails/node1/usr/ports`). Por fim, devemos verificar se há algum link simbólico ou não a atravessar a **Jail**.

Com estes cuidados básicos podemos prosseguir com a destruição da mesma.

```
$ tar -czvf ~/node2_etc.tar.gz node2/etc/*
$ tar -czvf ~/node2_usr_local_etc.tar.gz node2/usr/local/etc/*
$ tar -czvf ~/node2_var.tar.gz node2/var/*
$ tar -czvf ~/node2_home.tar.gz node2/usr/home/*
$ cd /etc/rc.d
$ ./jail stop node2
$ umount /usr/jails/node2/usr/ports
$ cd /usr/jails
$ chflags -R noschg node2
$ unlink node2/home
$ unlink node2/kernel
$ rm -rf node2
```

Resta agora, eliminar as entradas relativas a esta **Jail** em `/etc/rc.conf.local` para finalizar este ponto.

**Alerta:** Deverá efectuar "Backups" em conformidade com as suas necessidades.

Antes de proceder à eliminação da **Jail** não deixe de confirmar se o restauro dos mesmos corre como planeado.

Mais vale prevenir.

## 4 Jails para o impaciente

```
$ ifconfig NOME_INTERFACE inet alias IP_DA_JAIL netmask 0xffffffff
$ mkdir /usr/jails ; mkdir /usr/jails/NOME_DA_JAIL
$ cd /usr/src ; make -j4 world DESTDIR=/usr/jails/NOME_DA_JAIL
$ cd etc ; make distribution DESTDIR=/usr/jails/NOME_DA_JAIL

$ cd /usr/jails/NOME_DA_JAIL ; mkdir tmp/BUILD
$ touch etc/fstab ; echo "hostname=\"NOME\"" >> etc/rc.conf
$ echo "WRKDIRPREFIX=/tmp/BUILD" >> etc/make.conf
$ echo "ListenAddress IP_DA_JAIL" >> etc/ssh/sshd_config

$ mount_devfs devfs /usr/jails/NOME_DA_JAIL/dev
$ cd /usr/jails/NOME_DA_JAIL ; ln -sf dev/null kernel
$ mount_procfs proc /data/jail/NOME_DA_JAIL/proc
$ jail /usr/jails/NOME_DA_JAIL NOME IP_DA_JAIL /bin/sh /etc/rc
```

Nos passos acima descritos terá uma **Jail** pronta a funcionar em poucos minutos. No entanto, para que esta possa inicializar automaticamente após reiniciar o servidor, deverá configurar o `rc.conf.local` do Anfitrião, estas informações estão descritas no ponto **2.2** deste CF.

## 5 Informações Adicionais

Pode encontrar a última versão deste CF em <http://npf.pt.freebsd.org>. Entretanto, se tiver algum comentário, crítica ou correcção a fazer sobre este CF, pode e deve fazê-lo para [include@npf.pt.freebsd.org](mailto:include@npf.pt.freebsd.org). Obrigado.