NAME

ezjail — Jail administration framework.

SYNOPSIS

ezjail-admin *command arguments...*

OVERVIEW

The ezjail commands provide a simple way to create multiple jails using FreeBSD's jail system. It simplifies jail administration effort and minimizes jail system resource usage.

If you are not familiar with the FreeBSD jail concept, please refer to `jail`(8) before continuing. For additional design information, see the ezjail site at `http://erdgeist.org/arts/software/ezjail`.

DESCRIPTION

The ezjail system enables the system administrator to create multiple OS-level virtualization containers called jails. Services like web servers, mail servers, FTP servers, are typically under frequent attack from the public Internet and are exposed to possible compromise. The typical usage of jails is to run a single service in each jail and if that service becomes compromised the rest of the jails and the host system are protected from also being compromised.

The major shortcoming of jails is that each jail has its own copy of the world. This eats disk space, inodes, and more importantly, prevents the sharing of binaries images between jails, thus increasing the memory pressure on the host system. In addition, this causes a major administration headache when comes the time to update the host system, as each jail needs to be updated independently.

Ezjail addresses these problems by creating a single basejail (a read-only `nullfs`(4) mounted directory) populated with the same binaries as the host system which is then shared across all the other service jails created by ezjail. Is is possible to update the base jail (and thus all the jails) in a single ezjail command.

Typical usage of jails include separation of services, creating test environments, consolidation of different services on a single physical host, and more.

EZJAIL SYSTEM

The administrative interface to the ezjail system is the `ezjail-admin`(8) command. It is used to install the ezjail environment, create new jails, archive, restore, delete and update jails, open a jail console, and list the status of all the jails. See below for example usage, and refer to its man page for complete usage details.

Ezjail reads its configuration from its `ezjail.conf`(5). Normally it will not be necessary to edit this file, as some sane defaults are provided. A sample configuration is installed as `EZJAIL_PREFIX/etc/ezjail.conf.sample`.

A script is also installed as `ezjail` in the rc.d system to allow jails under ezjails control to be started at boot time, given ezjail is enabled by setting the `rc.conf`(5) variable "`$ezjail_enable`" to "`YES`".

WHAT'S IN A JAIL

The life of an ezjail installation

The base jail is first created by running **ezjail-admin update** or **ezjail-admin install**. Example usage of this command is section **EXAMPLES**. This will create the base jail, setup a template jail used to setting up new jails, install an example flavour (see below) and configure miscellaneous things.

This step is necessary before using the ezjail system. In particular, it is not possible to create new jails without initializing the base jail in advance.

Once the base jail has been created, new jails may be created with **ezjail-admin create**. A new jail is defined by its name and can have one or multiple IP addresses. Creating a new jail involves copying the template jail to the new location, configuring `nullfs`(4) mounts for giving access to the base jail, and little more. A jail that has just be created occupies about 2MB of disk space ; when running, only a handful of daemons (cron, syslog, sendmail mainly) use memory.

After their creation, jails may be archived to a `pax`(1) archive, restored, and eventually deleted.

When a new version of FreeBSD is released, or when an errata is published, only the base jail need to be updated. Both source upgrades and binary upgrades (using `freebsd-update`(8)) are supported. The `ports`(7) collection may also be updated by ezjail, but individual ports need to be upgraded individually by the administrator.

Anatomy of a Jail

In the ezjail system, a jail is defined by a root directory and a couple of configuration values, mainly a name and IP addresses. The root directory of the jail contains only the jail-specific files: configuration files, data files, and ports installed by the administrator. The base system is shared amongst all jails, using a `nullfs`(4) mount. This saves space and inodes (especially when the ports collection in made available to the jails), and also memory, as the kernel is now able to share copies of running programs between the jails.

Unless the variable "`$ezjail_jaildir`" has been set by the administrator, the root directory of the jail is kept in `/usr/jails`, which therefore needs to reside on a partition big enough.

There are also file-based jails, in which the storage space for the jail is kept in a file mounted with `mdconfig`(8). There are two advantages to image jails. The amount of disk space allocated to the jail is limited, while normal jails have no bound on the amount of disk space they use. On the other hand, the space dedicated to the jail is no longer available to the host, even if the jail doesn't use all its allocated space. In addition, image jails contain a full copy of the basejail. This makes them portable between hosts running the same FreeBSD version as the image was created with. Of course, the jail now needs to be updated independently from all other jails, and there is no longer any sharing of common files between the jails.

Image jails may also be encrypted using `bde`(4) or `geli`(8), depending on the options given at creation time.

Using ZFS

To give more precise control over the resources consumed by a jail, ezjail allows putting each jail in its own `zfs`(8) filesystem. See **Jail Creation Examples** for details.

Also, ezjail can be configured to install its basejail and the accompaning template for all new jails into its own filesystem. Set the "`$ezjail_use_zfs`" variable in your `ezjail.conf` to "YES" before running **`ezjail-admin update`** or **`ezjail-admin install`**.

To use any zfs feature in ezjail, you first need to configure the destination ZFS filesystem using the "`$ezjail_jailzfs`" variable.

You can use ZFS jails without installing the basejail into its own ZFS filesystem and vice versa. In order to create ZFS jails by default, set the "`$ezjail_use_zfs_for_jails`" variable to "YES".

Per-Jail options

As we saw earlier, a jail is described by a file in `EZJAIL_PREFIX/etc/ezjail/`. This file has the same name as the jail it configures. It is a set of variables interpreted by `sh`(1), much like `rc.conf`(5) is. This file is created at the same time as the jail, and usually doesn't require tweaking from the administrator.

In addition to the variables described below, any variable used by the init script `/etc/rc.d/jail` may be added manually by the administrator. The following variables are handled by ezjail, replacing JAILNAME with the actual name of the jail:

jail_JAILNAME_hostname
: The hostname of the jail. Defaults to the name of the jail, unless special characters needed to be stripped.

jail_JAILNAME_ip
: The IP addresses the jail is allowed to use.

Since FreeBSD 7.2, several IP addresses may be given, separated by commas.

Since FreeBSD 9.0 each IP address can be prefixed by an interface name followed by the pipe symbol. It will then automatically be configured on that interface when the jail is started and removed from the interface when the jail stops. (You will probably have to escape the pipe symbol, though.)

jail_JAILNAME_rootdir
: The directory holding the jail files (the directory used as a mount point for file-based jails). Defaults to the jail name inside “`$ezjail_jaildir`”.

jail_JAILNAME_exec_start
: The command to run inside the jail when starting it. Defaults to “`$ezjail_exec_start`” or “`/bin/sh /etc/rc`”.

jail_JAILNAME_exec_stop
: The command to run inside the jail when stopping it. Defaults to the empty string, which means “`/bin/sh /etc/rc.shutdown`”.

jail_JAILNAME_mount_enable
: A boolean (“YES” or “NO”), that specifies whether the filesystems in `/etc/fstab.`*`JAILNAME`* are carried out. Set by ezjail to “`YES`”, set to at your own risk.

jail_JAILNAME_devfs_enable
: A boolean specifying whether to mount a `/dev` filesystem inside the jail. Defaults to “`$ezjail_devfs_enable`”, or “`YES`”.

jail_JAILNAME_devfs_ruleset
: The ruleset to apply when mounting a `/dev` filesystem inside a jail. Defaults to “`$ezjail_devfs_ruleset`”, or “`devfsrules_jail`”.

ezjail_JAILNAME_procfs
: A boolean specifying whether to mount a `/proc` filesystem inside the jail. Defaults to “`$ezjail_procfs_enable`”, or “`YES`”.

ezjail_JAILNAME_fdescfs
: A boolean specifying whether to mount a `/dev/fs` filesystem inside the jail. Defaults to “`$ezjail_fdescfs_enable`”, or “`YES`”.

ezjail_JAILNAME_image
: The path to the image file backing the jail, if the jail is file-based; or the empty string.

ezjail_JAILNAME_imagetype
: The type of the image, if the jail is file-based; the empty string otherwise.

ezjail_JAILNAME_attachparams
: The parameters to pass to the tool used to decrypt file-based, encrypted jails. Initialized from the **-C** option when creating such a jail, or the empty string. “`YES`” if the jail requires interaction with the administrator when starting (typically, encrypted jails that needs a password to be decrypted).

ezjail_JAILNAME_forceblocking
: If “`YES`”, start the jail even when it is marked as blocking.

ezjail_JAILNAME_zfs_datasets
: For ZFS jails, additional ZFS datasets to attach to the jail when starting it. Taken from the **-z** option when configuring a jail; the empty string otherwise.

ezjail_JAILNAME_cpuset

The processor set to place the jail in when starting it (see `cpuset`(1)). Taken from the **−c** option when configuring a jail; the empty string otherwise.

ezjail_JAILNAME_fib

The network view to give to the jail (see `setfib`(1)) when starting it. Taken from the **−f** option when configuring the jail; the empty string otherwise.

ezjail_JAILNAME_parameters

The parameter set to be configured to the jail (see `jail`(8)) when starting it. You need to configure this by hand.

ezjail_JAILNAME_post_start_script

The path to a script that will be executed after the jail successfully was created. The script receives two parameters, the jid and the jail name. You need to configure this by hand.

In addition to these `sh`(1)-style variables, the administrator may add comment lines starting with "PROVIDE:", "REQUIRE:" and "BEFORE:". These comments are used by `rcorder`(8) to determine the order in which the jails are started. The default is to keep "REQUIRE" and "BEFORE" empty, meaning the jails are started in no particular order.

Flavours

When a jail is created, it is not configured; in particular you likely want to edit files such as `/etc/resolv.conf`, `/etc/localtime` and others. You may also want to create some system users, maybe enable `sshd`(8). Ezjail solves this problem by using the concept of "flavours". When a flavour is selected at jail creation time, the flavour directory tree is merged into the new jail's directory tree. In addition, the jail is configured so that on its first boot, the file `ezjail.flavour` is executed.

As part of the install sub-command, the flavour base directory was created as `/usr/jails/flavours` and populated with an single flavour named **`example`**. This flavour contains 3 files customized for running in a jail (`etc/make.conf`, `etc/periodic.conf`, `etc/rc.conf`). The example `ezjail.flavour` also show how to create users, and introduce the convention of placing packages in `/pkg` that are installed when the jail is first brought up. You are encouraged to copy the example flavour to create your own flavour. Typical flavour usages include setting up jails with site-specific configuration, creating classes of jails for development or testing (such as a webdev flavour that would install Apache with your favourite web development framework), pre-creating local users, and so on.

Updating the Base Jail

We already mentionned how easy it is to update jails, since only one copy needs to be updated. Ezjail only handles updating the base system; updating the ports is left to the administrator (but see "`ports-mgmt/jailaudit`" for a way to get notified of ports in need of an update). Updates are handled with the **`ezjail-admin update`** command. It is possible to update the base jail from source or from binary packages. If a base jail already exists, the **`update`** command installs the world in a temporary directory before moving it to the basejail, thus leaving intact all installed libraries. After making sure all software running in the jails is linked with the new librairies, you may want to remove the old library versions. It is often a good idea to update the jails when a new kernel is installed in the host, using the same sources.

Starting Jails

Like all `rc`(8) scripts, the ezjail script `EZJAIL_PREFIX/etc/rc.d/ezjail` accepts parameters **`start`**, **`restart`** and **`stop,`** running, restarting and stopping all (non-blocking) jails under ezjail's control by default. When passed an additional list of jails, only these jails are acted upon.

The order in which jails are started is determined by the `rcorder`(8) tool, using cues from the jail configurations in ezjails `EZJAIL_PREFIX/etc/ezjail` control directory.

The script examines its config, attaches and mounts images, and sets variables for each jail in the list before passing its command on to the `/etc/rc.d/jail` script.

To interactively start all crypto image jails (or those depending on them), that were not automatically started during booting, use the **`startcrypto`** parameter.

Note that jails configured to be in the *norun* state (using **`ezjail-admin config -r`** *`norun jailname`*) are never started by the ezjail rc script.

As a convenient shortcut, the **`ezjail-admin`** command invokes the rc.d script and passes the corresponding parameters, if they look like valid parameters.

Even if ezjail is not enabled in the `rc.conf`(5), rc.d/ezjail can be used to start and stop jails by prepending **`force`** or **`one`** to the **`start, restart`** or **`stop`** parameter. Refer to `rc`(8) for details.

Remarks & Tips

Jails can be either accessed from the network, for instance by using `ssh`(1), or from the host system by using the **`console`** command, which gives you an interactive shell inside the jail. It is also possible to edit the files of a running jail, and the modifications will appear immediately inside the jail environment. When dealing image-based, the **`config -i attach`** command allows one to access the disk of a file-based jail without starting it.

Raw sockets are disallowed by default for all jails. This is not a ezjail restriction, but a design default of the jail command. This means the `ping`(8) command will get "Operation not permitted." error when used from inside of a jail. There are `sysctl`(3) knobs for allowing a jail to access raw sockets, see the `jail`(8) man page for details.

Once your jail has network access, then all your normal application install functions are availabe, right from the jails console. In particular, if the ports collection was installed, it can be used as if from the host system. A modified `make.conf` file is installed by the example flavour, that enable the ports collection to work even with a read-only `/usr/ports`.

It is possible to change the IP address of a jail by editing its configuration file in `EZJAIL_PREFIX/etc/ezjail` and restarting the jail.

The jails use the same network stack as the host system. In particular, that means that if a firewall is needed, it must be configured in the host system.

The ezjail system (and the jails it controls) depends on the "`$ezjail_enable`" variable being set to "`YES`" in `rc.conf`. It is possible to set this variable to "`NO`" if the administrator wants to temporarily ezjail, or if she doesn't want the jails to be automatically started on boot.

The ezjail system may be reset to a pristine state by removing all its files, that is:
`/usr/jails/`
`EZJAIL_PREFIX/etc/ezjail/`
`EZJAIL_PREFIX/etc/ezjail.conf`
`/etc/fstab.*` (but check the list of files this matches)

EXAMPLES

The examples below are only that, examples. The reader is encouraged to read the `ezjail-admin`(8) man page for definitive documentation of all the options.

Initial Binary Installation

The ezjail system may be bootstrapped either from binary packages, or by building from source. The **`install`** command allow to bootstrap from binary packages, while the **`update`** deals with installations (and updates) from source.

ezjail-admin install (without any options)
Fetch and install binaries for populating the base jail from the FreeBSD FTP server. If the host is not running a -RELEASE version, you will be asked for the release to install. Neither the man pages nor the source nor the ports tree are installed. Note that the FreeBSD FTP server is sometimes so busy the download times out. Use the **-h** *host* option to specify a less loaded server, or the "`$ezjail_ftphost`" option in `ezjail.conf`(8).

ezjail-admin install -ms
Same behavior as above, except that man pages and sources are installed in the base jail.

ezjail-admin install -p
Same as the first example, but use `portsnap`(8) to fetch and extract a full FreeBSD ports tree from `portsnap.FreeBSD.org` into the base jail. This is necessary if you plan to install ports at later time into service jails.

ezjail-admin install -P (note uppercase P)
Only fetch the current version of the ports tree, adding it to the base jail. This allow to either add the ports tree after the initial installation or update the ports tree in the base jail.

Install from a disk image
Mount and use a downloaded `disc1.iso` CDRom image file.

```
mdconfig -a -f /usr/8.0-RELEASE-i386-disc1.iso md0
mount -v -t cd9660 /dev/md0 /mnt
cd /mnt/8.0-RELEASE
ezjail-admin install -h file:// -sm
```

When the installation finishes, use the following to release the `disc1.iso` `md0` file.

```
cd /usr
umount /mnt
mdconfig -d -u md0
```

Install from a local directory
To fetch the RELEASE base files manually, create a `.netrc` file in your home directory and populate it with this.

```
machine ftp2.jp.FreeBSD.org
login anonymous
password FBSD@home.com
macdef init
prompt off
cd /pub/FreeBSD/releases/i386/8.0-RELEASE
epsv4 off
$ getdir base kernels manpages src
quit
macdef getdir
! mkdir $i
mreget $i/*
```

Then issue this command on the command line. If the FTP download times out re-issue the FTP command again to resume where it left off.

```
mkdir /usr/8.0-RELEASE
cd /usr/8.0-RELEASE
ftp -v ftp2.jp.FreeBSD.org
ezjail-admin install -h file:// -sm
```

Use this option to target the 8.0-RELEASE files you FTP'ed as the source of the running binaries used to populate the base jail. In addition the man pages and sources will be installed into the base jail.

From Source Installation and Update

The **`update`** is used to both install or update from source the base jail, and for updating the base jail from binary packages.

`ezjail-admin update -b`

Build and install a world from source. The sources are taken from `/usr/src` (but see the **`-s`** flag). This can be used both for creating the initial base jail, and for updating it after the host has been upgraded.

`ezjail-admin update -u`

Update the base jail to the next release using `freebsd-update`(8) (i.e. using binary packages). This may be used only to update an existing installation.

`ezjail-admin update -U -s` *`8.0-RELEASE`*

Upgrade the base jail to the host system's release using `freebsd-update`(8). This may be used only to upgrade an existing installation. Tell freebsd-update which OS version to expect in the basejail via the **`-s`** option.

Note: Check `uname`(1) and especially the `UNAME_r` environment variable to upgrade to different versions.

Jail Creation Examples

`ezjail-admin create` *`www.example.com 10.0.10.1`*

Create a new jail. The jail files will reside in directory `www_example_com` in `/usr/jails`, unless the variable "`$ezjail_jaildir`" has been set to some other value. The jail will only be allowed to use the given IP address. A warning will be displayed if this IP address is not already configured in the host, or if some network daemon is already listening on this address. The name of the jail which will appear in the **`list`** command or which will need to be given to the **`console`** command is *`www.example.com`*.

`ezjail-admin create -f` *`example`* **`-r`** *`webserver www.example.com 10.0.10.2,2001:db8:1:9243::80`*

Create a new jail, placing it in directory `webserver` instead of deriving the directory name of the jail from its host name. The jail will be created with the flavour *`example`*. This jail will be given two IP addressses; this is possible only since FreeBSD 7.2.

`ezjail-admin create -i -s` *`600M sandbox2 10.0.10.4`*

This creates a new file-based jail having a file size of 600 megabytes in `/usr/jails/sandbox2.img`. An empty directory, `/usr/jails/sandbox2`, will be created, and used as a mount point when starting the jail.

`ezjail-admin create -c bde -s` *`600M sandbox3 10.0.10.5`*

This creates a new file based image jail, with `gbde`(4) encryption. During the gbde creation process you are asked to enter a passphrase that is used as the prime seed value of the encryption process. Remember this passphrase, you will be asked for the passphrase every time you want to start this jail. As they require administrator interaction, jails backed by an encrypted file are not automatically started when the system boots.

`ezjail-admin create -c` *`zfs`* **`-s`** *`1G sandbox4 em1\|10.0.10.6`*

This creates a new zfs filesystem based jail with a default quota of 1 gigabyte using lzjb compression. It uses the parent ZFS filesystem configured in the "`$ezjail_jailzfs`" variable to create the filesystem in. The jail command will add the ip address 10.0.10.6 as an alias on the device em1 before starting the jail.

FILES

```
EZJAIL_PREFIX/bin/ezjail-admin
EZJAIL_PREFIX/etc/rc.d/ezjail
EZJAIL_PREFIX/etc/ezjail.conf
EZJAIL_PREFIX/share/examples/ezjail/
EZJAIL_PREFIX/etc/ezjail/*
/usr/etc/fstab.*
```

SEE ALSO

`ezjail-admin`(8), `ezjail.conf`(5), `jail`(8), `nullfs`(4), `zfs`(8).

Interesting additional tools include: “`ports-mgmt/jailaudit`”.

AUTHOR

Dirk Engling ⟨erdgeist@erdgeist.org⟩.

The man page is based on a draft by JoeB ⟨joeb1@a1poweruser.com⟩ and was rewritten by Frederic Perrin ⟨frederic.perrin@resel.fr⟩.