iocell Documentation

Release 1.7.3

iocell

Nov 23, 2018

Contents

iocell is a zero dependency drop in jail/container manager amalgamating some of the best features and technologies FreeBSD operating system has to offer. It is geared for ease of use with a simple and easy to understand command syntax.

FEATURES:

- Templates, clones, basejails, fully independent jails
- Ease of use
- Zero configuration files
- Rapid thin provisioning within seconds
- Automatic package installation
- Virtual networking stacks (vnet)
- Shared IP based jails (non vnet)
- Resource limits (CPU, MEMORY, etc.)
- Filesystem quotas and reservations
- Dedicated ZFS datasets inside jails
- Transparent ZFS snapshot management
- Binary updates
- Differential jail packaging
- Export and import
- And many more!

CHAPTER 1

Documentation:

1.1 Basic usage

This section is about basic use and is a suitable quick howto for newcomers.

1.1.1 Fetch a release

The very first step with iocell is to fetch a RELEASE. By default iocell will attempt to fetch the host's current RELEASE from the freebsd.org servers. Once the RELEASE bits are downloaded, the most recent patches are applied too.

Simply run:

```
iocell fetch
```

If a specific RELEASE is required run:

```
iocell fetch release=9.3-RELEASE
```

In case a specific download mirror (FTP or HTTP) is required simply run:

```
iocell fetch ftphost=ftp.hostname.org
```

You can also specify a directory to fetch the base files from:

```
iocell fetch ftpdir=/dir/
```

1.1.2 Create a jail

There are three supported basic jail types: full, clone and base jail. In addition to these three there are two more which are discussed later (empty and templates). Depending on requirements the *create* subcommand can be tweaked to create any of the three types. By default iocell will create a fully independent jail of the current host's RELEASE and set the TAG property to todays date.

Creating a jail is real simple, just run:

```
iocell create
```

This will create a fully independent jail.

To create a lightweight jail (clone) run:

```
iocell create -c
```

To create a base jail:

```
iocell create -b
```

To create a jail and set its IP address and tag property run:

```
iocell create -c tag=myjail ip4_addr="em0|10.1.1.10/24"
```

For more information please read iocell(8).

1.1.3 Listing jails

To list all jails run:

```
iocell list
```

To see all downloaded RELEASEs run:

```
iocell list -r
```

To see available templates run:

```
iocell list -t
```

1.1.4 Start, stop or restart a jail

To start or stop any jail on the system both the UUID or TAG can be used interchangeably. To simplify UUID handling iocell accepts a partial UUID too with any subcommand.

Start

To start a jail tagged www01 simply run:

```
iocell start www01
```

To start a jail with a full UUID run:

```
iocell start 26e8e027-f00c-11e4-8f7f-3c970e80eb61
```

Or to start the jail only with a partial UUID enter the first few characters only:

```
iocell start 26e8
```

Stop

To stop a jail just use the `stop` subcommand instead of start:

```
iocell stop www01
```

Restart

To restart a jail run:

```
iocell restart www01
```

Note: Short UUIDs are supported with all operations and subcommands within iocell.

1.1.5 Configure a jail

Any property can be reconfigured with the `set` subcommand.

Set property

To set the jail's TAG property run:

```
iocell set tag=www02 26e8e027
```

Get property

To verify any property simply run the `get` subcommand:

```
iocell get tag 26e8e027
```

Get all properties:

Or to display all supported properties run:

```
iocell get all 26e8e027
```

1.1.6 System wide defaults

Starting with version 1.6.0 system wide defaults can be set. These defaults will be re-applied for all newly created jails. To create a system wide default override for a property simply specify the `default` keyword instead of a jail UUID or TAG.

Example, to turn off VNET capability for all newly created jails run:

```
iocell set vnet=off default
```

1.1.7 Destroy a jail

To destroy a jail, simply run:

```
iocell destroy www02
```

Warning: this will irreversibly destroy the jail!

1.2 Networking

1.2.1 Intro

Jails have multiple networking options based on what features are desired. Traditionally jails only supported IP alias based networking where an IP address is assigned to the host's interface which is then utilized by the jail for network communication. This is known as "shared IP" based jails.

Anoter option emerged in recent years, called VNET or sometimes referred to as VIMAGE. VNET is a fully virtualized, isolated per jail networking stack. VNET abstracts virtual network interfaces to jails, which behave the same way as physical interfaces.

iocell will try to guess whether VNET support is available in the system and if it is will enable it by default for newly created jails.

1.2.2 Shared IP

Stability: It is rock solid and battle tested well over a decade.

System requirements

None, everything is built into the default GENERIC kernel.

Usage

Make sure VNET is disabled

`iocell get vnet UUID | TAG`

If set to "on" disable it

`iocell set vnet=off UUID | TAG`

A system wide default can be configured if required. This will take effect for newly created jails only.

`iocell set vnet=off default`

Configure an IP address

`iocell set ip4_addr="em0|10.1.1.10/24" UUID| TAG`

If multiple addresses are desired just separate the configuration directives with a comma.

Example:

`iocell set ip4_addr="em0|10.1.1.10/24,em0|10.1.1.11/24" UUID| TAG`

Allow raw sockets

`iocell set allow_raw_sockets=1 UUID | TAG`

This allows the prison root to create raw sockets, which allows utilites like ping and tracroute to operate.

Start jail:

`iocell start UUID | TAG`

Verify visible IP configuration in the jail

(jail must be running for this to work)

`iocell exec UUID | TAG ifconfig`

1.2.3 VIMAGE/VNET

Stability: VIMAGE is considered experimental, unexpected system crashes can occur (for details please see known issues section)

System requirements

Kernel

Rebuild the kernel with the following options:

(also disable SCTP if not required)

```
nooptions       SCTP   # Stream Control Transmission Protocol
options         VIMAGE # VNET/Vimage support
options         RACCT  # Resource containers
options         RCTL   # same as above
```

/etc/rc.conf

Add bridge configuration to /etc/rc.conf:

(on the host node)

```
# set up two bridge interfaces for iocell
cloned_interfaces="bridge0 bridge1"

# plumb interface em0 into bridge0
ifconfig_bridge0="addm em0 up"
ifconfig_em0="up"
```

/etc/sysctl.conf

Add these tunables to /etc/sysctl.conf:

```
net.inet.ip.forwarding=1        # Enable IP forwarding between interfaces
net.link.bridge.pfil_onlyip=0   # Only pass IP packets when pfil is enabled
net.link.bridge.pfil_bridge=0   # Packet filter on the bridge interface
net.link.bridge.pfil_member=0   # Packet filter on the member interface
```

Configure default GW for jail

Example: `iocell set defaultrouter=10.1.1.254 UUID | TAG`

Configure an IP address

`iocell set ip4_addr="vnet0|10.1.1.10/24" UUID | TAG`

Start jail and ping default gateway

Start the jail:

`iocell start UUID | TAG`

Drop into jail:

`iocell console UUID | TAG`

Ping default gateway, example:

```
ping 10.1.1.254
```

Gotchas

Routes

Make sure default gateway knows the route back to the VNET subnets.

If using VLANs

If you are using VLAN interfaces for the jail host you not only have to add the vlan interface as bridge member but the parent interface of the VLAN as bridge member as well.

1.2.4 Configuring Network Interfaces

iocell handles network configuration for both, shared IP and VNET jails transparently.

Configuring a shared IP jail

IPv4

```
iocell set ip4_addr="em0|192.168.0.10/24" UUID|TAG
```

IPv6

```
iocell set ip6_addr="em0|2001:123:456:242::5/64" UUID|TAG
```

This will add an IP alias 192.168.0.10/24 to interface em0 for the shared IP jail at start time, as well as 2001:123:456::5/64.

Configuring a VNET jail

To configure both IPv4 and IPv6:

```
iocell set ip4_addr="vnet0|192.168.0.10/24" UUID|TAG
```

```
iocell set ip6_addr="vnet0|2001:123:456:242::5/64" UUID|TAG
```

```
iocell set defaultrouter6="2001:123:456:242::1" UUID|TAG
```

NOTE: For VNET jails a default route has to be specified too.

Hints

To start a jail with no IPv4/6 address whatsoever set these properties:

```
iocell set ip4_addr=none ip6_addr=none UUID|TAG
```

```
iocell set defaultrouter=none defaultrouter6=none UUID|TAG
```

1.3 Jail types

iocell supports five different jail types:

- thick (default)

- thin
- base
- template
- empty

All types have their pros & cons and serves different needs.

1.3.1 Full (thick)

Full (thick) jail is the default type and it is created with the following command:

```
iocell create
```

A full jail has a fully independent ZFS dataset suitable for network replication (ZFS send/recv).

1.3.2 Clone (thin)

Thin jails are lightweight clones created with:

```
iocell create -c
```

Thin jails are cloned from the appropriate RELEASE at creation time and consume only a fraction of space, preserving only the changing data.

1.3.3 Base

The original basejail concept based on nullfs mounts got popularized by ezjail. iocell basejails use independent read-only ZFS filesystem clones to achieve the same functionality.

To create a basejail execute:

```
iocell create -b
```

Basejails re-clone their base filesystems at each startup. They are ideal for environments where patching or upgrades are required at once to multiple jails.

1.3.4 Template

Template is just another jail where the "template" property is set to "yes".

To turn a jail into a template simply execute:

```
iocell set template=yes UUID|TAG
```

After this operation the jail can be listed with:

```
iocell list -t
```

To deploy a jail from this template, execute:

```
iocell clone TEMPLATE_UUID tag=mynewjail
```

Templates can be converted back and forth with setting the "template" property.

1.3.5 Empty

Empty jails are intended for unsupported jail setups or testing. To create an empty jail run:

```
iocell create -e
```

These are ideal for experimentation with unsupported RELEASES or Linux jails.

1.4 Best practices

These are some generic guidelines for working with iocell managed jails.

Use PF as a module

This is the default setting in the `GENERIC` kernel. There seems to be a VNET bug which is only triggered when PF is directly compiled into the kernel.

Always tag your jails and templates!

This will help you avoid mistakes and easily identify jails.

Set the notes property

Set the `notes` property to something meaningful, especially for templates and jails you might use only once in a while.

VNET

`VNET` will give you more control and isolation. Also allows to run per jail firewalls. See known issues about VNET.

Don't overuse resource limiting!

Unless really needed, let the OS decide how to do it best. Set limits with the "log action" before enforcing "deny". This way you can check the logs before creating any performance bottlenecks.

Discover templates!

Templates will make your life easy, try them!

Use the restart command instead of start/stop

If you wish to restart a jail use the `restart` command which performs a soft restart and it leaves the `VNET` stack alone, less stressful for the kernel and you.

Check your firewall rules

When using `IPFW` inside a `VNET` jail put `firewall_enable="YES" firewall_type="open"` into `/etc/rc.conf` for a start. This way you can exclude the firewall from blocking you right from the beginning! Lock it down once you've tested everything. Also check PF firewall rules on the host if you happen to mix both.

Get rid of old snapshots

Remove snapshots you don't need, especially from jails where data is changing a lot!

Use the chroot sub-command

In case you need to access or modify files in a template or a jail which is in a stopped state, use `iocell chroot UUID | TAG`. This way you don't need to spin up the jail or convert the template.

1.5 Advanced usage

1.5.1 Clones

When a jail is cloned, iocell creates a ZFS clone filesystem. In a nutshell clones are cheap lightweight writable snapshots.

A clone depends on its source snapshot and filesystem. If you'd like to destroy the source jail and preserve its clones you need to promote the clone first, otherwise the source jail cannot be destroyed.

Create a clone

To clone www01 to www02 run:

```
iocell clone www01 tag=www02
```

To clone a jail from an existing snapshot:

```
iocell clone www01@snapshotname tag=www03
```

Promoting a clone

To promote a cloned jail, simply run:

```
iocell promote UUID | TAG
```

The above step will reverse the clone and source jail relationship. Basically the clone will become the source and the source jail will be demoted to a clone.

Now you can remove the demoted jail with:

```
iocell destroy UUID | TAG
```

1.5.2 Updating jails

Updates are handled with the freebsd-update(8) utility. Jails can be updated while they are stopped or running.

To update a jail to latest patch level run:

```
iocell update UUID | TAG
```

This will create a back-out snapshot of the jail automatically.

When finished with updating and the jail is working OK, simply remove the snapshot:

```
iocell snapremove UUID|TAG@snapshotname
```

In case the update breaks the jail, simply revert back to the snapshot:

```
iocell rollback UUID|TAG@snapshotname
```

If you'd like to test updating without affecting a jail, create a clone and update the clone the same way as outlined above.

To clone run:

```
iocell clone UUID|TAG tag=testupdate
```

1.5.3 Upgrading jails

Upgrades are handled with the freebsd-update(8) utility. By default the upgrade command will try to upgrade the jail to the hosts RELEASE version (uname -r).

Based on the jail "type" property, upgrades are handled differently for basejails and non basejails.

Upgrade non-basejail

To upgrade a normal jail (non basejail) to the hosts RELEASE run:

```
iocell upgrade UUID | TAG
```

This will upgrade the jail to the same RELEASE as the host.

To upgrade to a specific release run:

```
iocell upgrade UUID|TAG release=10.1-RELEASE
```

Upgrade basejail

To upgrade a basejail:

Verify whether the jail is a basejail:

```
iocell get type UUID|TAG
```

Should return type "basejail".

The upgrade can be forced while the jail is online with executing:

```
    iocell upgrade UUID|TAG
```

This will forcibly re-clone the basejail filesystems while the jail is running (no downtime) and update the jails /etc with the changes from the new RELEASE.

```
iocell set release=10.1-RELEASE UUID|TAG
```

This will cause the jail to re-clone its filesystems from 10.1-RELEASE on next jail start. This will not update the jails /etc files with changes from the next RELEASE.

1.5.4 Auto boot

Make sure `iocell_enable="YES"` is set in `/etc/rc.conf`.

To enable a jail to auto-boot during a boot, simply run:

```
iocell set boot=on UUID|TAG
```

Boot priority

Boot order can be specified by setting the priority value:

```
iocell set priority=20 UUID|TAG
```

Lower value means higher boot priority.

1.5.5 Snapshot management

iocell supports transparent ZFS snapshot management out of the box. Snapshots are point-in-time copies of data, a safety point to which a jail can be reverted at any time. Initially snapshots take up almost no space as only changing data is recorded.

List snapshots for a jail with:

`iocell snaplist UUID|TAG`

To create a new snapshot run:

`iocell snapshot UUID|TAG`

This will create a snapshot based on current time.

If you'd like to create a snapshot with custom naming run:

`iocell snapshot UUID|TAG@mysnapshotname`

1.5.6 Resource limits

iocell can enable optional resource limits for a jail. The outlined procedure should provide enough for a decent starting point.

Limit core or thread

Limit a jail to a single thread or core number 1:

`iocell set cpuset=1 UUID|TAG` `iocell start UUID|TAG`

List applied rules

List applied limits:

`iocell limits UUID|TAG`

Limit DRAM use

Limit a jail to 4G DRAM memory use (limit RSS memory use can be done on-the-fly):

`iocell set memoryuse=4G:deny UUID|TAG`

Turn on resource limits

Turn on resource limiting for jail:

`iocell set rlimits=on UUID|TAG`

Apply limits

Apply limit on-the-fly:

`iocell cap UUID | TAG`

Check limits

Check active limits:

`iocell limits UUID | TAG`

Limit CPU use by %

Limit CPU execution to 20%:

`iocell set pcpu=20:deny UUID|TAG` `iocell cap UUID|TAG`

Check limits:

`iocell limits UUID | TAG`

Resetting a jail's properties

If you have many properties on a jail that you would like to reset back to defaults, iocell easily allows that!

To reset to defaults:

`iocell reset UUID | TAG`

You can also reset every jail to the default properties:

`iocell reset ALL`

Resetting a jail will retain the jails UUID and TAG. Everything else will be lost. Make sure to set any custom properties back that you need. If you have set anything via `iocell set PROPERTY default` You have nothing left to do!

1.5.7 Automatic package installation

Packages can be installed automatically at creation time!

Specify the `pkglist` property at creation time, which should point to a text file containing one package name per line. Please note you need to have Internet connection for this to work as `pkg install` will try to get the packages from online repositories.

Example:

Create a pkgs.txt file and add package names to it.

`pkgs.txt:`

```
nginx
tmux
```

Now simply create a jail and supply the pkgs.txt file:

`iocell create pkglist=/path-to/pkgs.txt tag=myjail`

This will install `nginx` and `tmux` in the newly created jail.

1.6 How to create and use templates

Templates can save you precious time!

Set up a jail any way you like, and create a template from it. All packages and pre-configured settings will be available for deployment next time within seconds.

Any jail can be converted to a template and back to a jail again as required. In fact a template is just another jail which has the property `template` set to "yes". The difference is that templates are not started by iocell.

Here is how to create a template with iocell:

1. Create a new jail `iocell create tag=mytemplate`
2. Configure the jail's networking
3. Install any package you like and customize jail
4. Once finished with customization stop the jail `iocell stop UUID | TAG`
5. A good idea is set some notes `iocell set notes="customized PHP,nginx jail" UUID | TAG`
6. Turn the template property on `iocell set template=yes UUID | TAG`
7. List your template with `iocell list -t`

Here is how to use the created template:

To create a new jail from this template simply clone it!

1. `iocell clone UUID-of-mytemplate tag=mynewjail`
2. List new jail `iocell list`
3. Start jail `iocell start UUID | TAG`

Done!

If you need to make further customization in the template or want to patch it, you have two options.

- convert template back to jail with `iocell set template=no UUID-of-template`, and start the jail
- if you don't need network access to make the changes simply run `iocell chroot UUID-of-template`, make the changes and exit

1.7 Create a jail package!

What is a jail package? A jail package is basically a small differential image template which can be deployed on top of vanilla jails. The RELEASE and patch level has to match between the package and a vanilla jail.

iocell uses the **record** function for this, which is a **unionfs** mount under the hood.

The resulting package can be stored on a web server with a checksum file ready to be deployed anywhere.

1. create a new jail `iocell create -c tag=nginx`
2. start jail `iocell start UUID | TAG`
3. configure networking to enable internet access for this jail
4. issue `iocell record start UUID | TAG`, from now on every change will be recorded under `/iocell/jails/UUID/recorded`
5. install nginx with `pkg install nginx`

6. install any other software you might require
7. customize configuration files
8. once finished, stop recording changes with `iocell record stop UUID | TAG` optionally stop jail
9. examine `/iocell/jails/UUID/recorded`, run `find /iocell/jails/UUID/recorded -type f`
10. remove any unnecessary files, make final customization/changes
11. run `iocell package UUID | TAG`, this will create a package in `/iocell/packages` with a SHA256 checksum file
12. optionally discard the jail now with `iocell destroy UUID | TAG`

The resulting `UUID.tar.xz` can now be deployed on top of any new vanilla jail!

1. create new jail `iocell create -c`
2. deploy package `iocell import UUID tag=myjail`
3. list jail `iocell list|grep myjail`, grab UUID
4. start jail `iocell start UUID | TAG`
5. examine your changes and packages - they are all there!

Enjoy!

1.8 Create a Debian Squeeze jail (GNU/kFreeBSD)

In this howto we will set up a Debian (GNU/kFreeBSD) jail. GNrUkFreeBSD is a Debian userland tailored for the FreeBSD kernel.

Don't forget to replace UUID with your jail's full UUID!

Create an empty jail with linux specifics:

`iocell create -e tag=debian exec_start="/etc/init.d/rc 3" exec_stop="/etc/init.d/rc 0"`

Install debootstrap on the host:

`pkg install debootstrap`

Grab the mountpoint for our empty jail, append /root/ to it and run debootstrap:

`iocell get mountpoint UUID | TAG`

`debootstrap squeeze /iocell/jails/UUID/root/` (you can replace squeeze with wheezy if that is what you need)

Edit the jail's fstab and add these lines:

`/iocell/jails/UUID/fstab`

```
linsys   /iocell/jails/UUID/root/sys         linsysfs  rw           0  0
linproc  /iocell/jails/UUID/root/proc        linprocfs rw           0  0
tmpfs    /iocell/jails/UUID/root/lib/init/rw tmpfs     rw,mode=777  0  0
```

Start the jail and attach to it:

`iocell start UUID | TAG`

```
iocell console UUID | TAG
```

What you gain is a 64bit Debian Linux userland. Please note this is not recommended for production use. The intention was to show that iocell will let you do almost anything you want with your jails.

If you wish to install a Linux only Debian jail you can follow this tutorial: debian-linux-freebsd-jail-zfs

1.9 Known Issues

This is a short list of known issues.

1.9.1 88 character mount path limitation

There is a know mountpoint path length limitation issue on FreeBSD which is set to a historical 88 character limit.

This issue does not affect iocell jails from functioning properly, but can present challenges when diving into ZFS snapshots (cd into .zfs/snapshots, tar, etc.).

ZFS snapshot creation and rollback is not affected.

To workaround this issue iocell 1.6.0 introduced a `hack88` property.

Example:

Shut down jail:

```
iocell stop myjail
```

Set the `hack88` property to “1”:

```
iocell set hack88=1
```

Start jail:

```
iocell start myjail
```

To revert back to full paths repeat the procedure but set `hack88=0`.

To create a system wide default (introduced in 1.6.0) for all newly created jails use:

```
iocell set hack88=1 default
```

1.9.2 Property validation

iocell does not validate properties right now. Please refer to man page to see what is supported for each property. By default iocell pre-configures each property with a safe default.

1.9.3 VNET/VIMAGE issues

VNET/VIMAGE can cause unexpected system crashes when VNET enabled jails are destroyed - that is when the jail process is killed, removed, stopped.

As a workaround iocell allows a warm restart without destroying the jail. By default the restart sub-command will execute a warm restart.

Example:

```
iocell restart UUID
```

FreeBSD 10.1-RELEASE is stable enough to run with VNET and warm restarts. There are production machines with iocell and VNET jails running well beyond 100 days of uptime running both PF and IPFW.

1.9.4 VNET/VIMAGE issues w/ ALTQ

As recent as FreeBSD 10.1-RELEASE-p10, there is some *interesting* interaction between VNET/VIMAGE and ALTQ, which is an ALTernate Queueing system used by PF and other routing software. Should you compile a kernel, make sure that you do not have any of the following lines in your kernconf (unless you want to disable VNET):

```
options     ALTQ
options     ALTQ_CBQ
options     ALTQ_RED
options     ALTQ_RIO
options     ALTQ_HFSC
options     ALTQ_CDNR
options     ALTQ_PRIQ
```

Otherwise, should you try to start a jail with VNET support enabled, your host system will more than likely crash. You can read a little more at the mailing list post here.

1.9.5 IPv6 host bind failures

In some cases a jail with an ip6 address may take too long adding the address to the interface, and services defined to bind specifically to that address may fail. In such cases, adding the following sysctl do disable DAD (duplicate address detection) probe packets.

To set, `sysctl net.inet6.ip6.dad_count=0`. To make it permanent, add the setting to sysctl.conf.

```
# disable duplicated address detection probe packets for jails
net.inet6.ip6.dad_count=0
```

You can read a little more about this here and at the mailing list post here.

1.10 FAQ

What is iocell? iocell is jail management script aiming to simplify jail administration tasks as much as possible.

What is a jail? Jail is a FreeBSD **OS virtualization** technology allowing to run multiple copies of the operating system. Some operating systems use the term **Zones** or **Containers** for OS virtualization.

What is VNET? VNET is an independent per jail virtual networking stack.

How do I configure network interfaces in a VNET or shared IPjail? You configure both the same way: `iocell set ip4_add="interface|IP/netmask" UUID | TAG`. For more info please refer to the documentation.

Do I need to set my default gateway? Only if VNET is enabled. You need to assign an IP address to the **bridge** where the jail interface is attached to. This IP essentially becomes your default gateway for your jail.

Can I run a firewall inside a jail? Yes in a VNET jail **IPFW is supported**. PF is not supported inside the jail - though you can still enable PF for the host itself. If you plan on using **IPFW** inside a jail make sure **securelevel** is set to **2**

Can I enable both IPFW and PF at the same time? Yes, make sure you allow traffic on both in/out for your jails.

Can I create custom jail templates? Yes, and thin provision them too! Starting with version 1.3 there is also a package option for jail packaging.

What is a jail clone? **Clones** are ZFS clones, these are fully writable copies of the source jail.

Can I limit the CPU and Memory use? Yes. (refer to manual page)

Is there a way to display resource consumption? Yes, `iocell inuse UUID | TAG`

Is NAT supported for the jails? Yes. This is built into FreeBSD. Treat your server as a core router/firewall. Check documentation section on NAT.

Will iocell work on a generic system with no ZFS pools? No. ZFS is a must, if you run a FreeBSD server you should be using ZFS!

Is ZFS jailing supported? Yes, please refer to man page.

1.11 Indices and tables

- genindex
- modindex
- search