ZFS Boot Environments Reloaded

Sławomir Wojciech Wojtczak
vermaden@interia.pl
vermaden.wordpress.com
twitter.com/vermaden

https://is.gd/BECTL
What is ZFS Boot Environment?

Its bootable clone/snapshot of the working system.
What is ZFS Boot Environment?

Its bootable clone/snapshot of the working system.

- In ZFS terminology its clone of the snapshot.
  
  ZFS dataset → ZFS dataset@snapshot → ZFS clone (origin=dataset@snapshot)
What is ZFS Boot Environment?

Its bootable clone/snapshot of the working system.

- In ZFS terminology its clone of the snapshot.
  
  ZFS dataset → ZFS dataset@snapshot → ZFS clone (origin=dataset@snapshot)

- In ZFS (as everywhere) snapshot is read only.
What is ZFS Boot Environment?

Its bootable clone/snapshot of the working system.

- In ZFS terminology its clone of the snapshot.
  
  ZFS dataset → ZFS dataset@snapshot → ZFS clone (origin=dataset@snapshot)

- In ZFS (as everywhere) snapshot is read only.

- In ZFS clone can be mounted read write (and you can boot from it).
What is ZFS Boot Environment?

Its bootable clone/snapshot of the working system.

- In ZFS terminology its clone of the snapshot.  
  ZFS dataset → ZFS dataset@snapshot → ZFS clone (origin=dataset@snapshot)

- In ZFS (as everywhere) snapshot is read only.

- In ZFS clone can be mounted read write (and you can boot from it).

- The BEs are placed in the pool/ROOT ZFS dataset path.
  sys/ROOT/default
  sys/ROOT/safe
  sys/ROOT/pre-upgrade
  (...
Use cases?

Allows **bulletproof** upgrades/changes to the system.
Use cases?

Allows **bulletproof** upgrades/changes to the system.

- Create **safe failback** ZFS Boot Environment before upgrade or major changes to system.
Use cases?

Allows **bulletproof** upgrades/changes to the system.

- Create **safe failback** ZFS Boot Environment before upgrade or major changes to system.

- Update system **inside** new ZFS Boot Environment without touching running system.
Use cases?

Allows **bulletproof** upgrades/changes to the system.

- Create **safe failback** ZFS Boot Environment before upgrade or major changes to system.
- Update system **inside** new ZFS Boot Environment without touching running system.
- Perform upgrade and test the results **inside FreeBSD Jail**.
Use cases?

Allows **bulletproof** upgrades/changes to the system.

- Create **safe failback** ZFS Boot Environment before upgrade or major changes to system.
- Update system **inside** new ZFS Boot Environment without touching running system.
- Perform upgrade and test the results **inside FreeBSD Jail**.
- **Copy/move** ZFS Boot Environment into another machine.
Use cases?

Allows **bulletproof** upgrades/changes to the system.

- Create **safe failback** ZFS Boot Environment before upgrade or major changes to system.
- Update system **inside** new ZFS Boot Environment without touching running system.
- Perform upgrade and test the results **inside FreeBSD Jail**.
- **Copy/move** ZFS Boot Environment into another machine.
- **Major reconfiguration** (Bareos/Postfix/...).
Use cases?

Allows **bulletproof** upgrades/changes to the system.

- Create **safe failback** ZFS Boot Environment before upgrade or major changes to system.
- Update system **inside** new ZFS Boot Environment without touching running system.
- Perform upgrade and test the results **inside FreeBSD Jail**.
- **Copy/move** ZFS Boot Environment into another machine.
- **Major reconfiguration** (Bareos/Postfix/...).
- **Mass populate** large amount of servers with one configured BE.
Use cases?

Allows **bulletproof** upgrades/changes to the system.

- Create **safe failback** ZFS Boot Environment before upgrade or major changes to system.
- Update system **inside** new ZFS Boot Environment without touching running system.
- Perform upgrade and test the results **inside FreeBSD Jail**.
- **Copy/move** ZFS Boot Environment into another machine.
- **Major reconfiguration** (Bareos/Postfix/...).
- **Mass populate** large amount of servers with one configured BE.
- Bare metal **backup** solution.
Can I test and break ZFS BEs without consequences?
Can I test and break ZFS BEs without consequences?

Yes you can! Over and over again.
Can I test and break ZFS BEs without consequences?

Yes you can! Over and over again.

Groundhog Day (1993)
How the World was before BEs?

Vendors used **split mirror** or **copying files** to the other/second disk.

**IBM AIX**
- alt_disk_copy
- alt_disk_install
- nimadm
- unmirrorvg
  (...)

**SUN Solaris Live Upgrade**
- lucreate
- luactivate
- luupgrade
- ludelete
  (...)

**HP-UX**
- lvsplit
- lvmerge
- vgchange
- vgcfgrestore
  (...)

**GNU/Linux**
- mdadm
- mirrorlv
- lvconvert
  (...)

**Dark ages**

2018/11/15
Mistyped command?

Felling lucky?

Raiders of the Lost Ark (1981)

One mistake and you have a disaster ... recovery
The `beadm` command

One simple command - `beadm` - to create/activate/destroy ZFS Boot Environments.

```bash
# beadm
usage:
beadm activate <beName>
beadm create [-e nonActiveBe | -e beName@snapshot] <beName>
beadm create <beName@snapshot>
beadm destroy [-F] <beName | beName@snapshot>
beadm list [-a] [-s] [-D] [-H]
beadm rename <origBeName> <newBeName>
beadm mount <beName> [mountpoint]
beadm { umount | unmount } [-f] <beName>
beadm version
```
The `beadm` is written in POSIX `/bin/sh`
List current BEs and create new one named `newbe`.

```bash
# beadm list
BE     Active Mountpoint  Space Created
11.2-RELEASE NR /       6.3G 2018-11-15 16:01

# beadm create newbe
Created successfully

# beadm list
BE     Active Mountpoint  Space Created
11.2-RELEASE NR /       6.3G 2018-11-15 16:01
newbe  -         -       296.0K 2018-11-15 17:04
```
Verify which **snapshot** is used for this **clone** used as **newbe** BE.

```bash
# beadm list -s
BE/Dataset/Snapshot                         Active Mountpoint  Space Created
11.2-RELEASE
   sys/ROOT/11.2-RELEASE                     NR     /            6.3G 2018-11-15 16:01
   sys/ROOT/11.2-RELEASE@2018-11-15-17:04:22 -      -          288.0K 2018-11-15 10:04
newbe
   sys/ROOT/newbe                           -      -            8.0K 2018-11-15 10:04
   11.2-RELEASE@2018-11-15-17:04:22         -      -          288.0K 2018-11-15 10:04
```

```bash
# zfs get origin sys/ROOT/newbe
NAME            PROPERTY  VALUE                                      SOURCE
sys/ROOT/newbe  origin    sys/ROOT/11.2-RELEASE@2018-11-15-17:04:22 -
```
Rename snapshot used for this clone.

```bash
# zfs rename sys/ROOT/11.2-RELEASE@2018-11-15-17:04:22 sys/ROOT/11.2-RELEASE@newbe

# zfs get origin sys/ROOT/newbe
NAME            PROPERTY  VALUE                        SOURCE
sys/ROOT/newbe  origin    sys/ROOT/11.2-RELEASE@newbe -

# beadm list -s
BE/Dataset/Snapshot           Active Mountpoint  Space  Created
11.2-RELEASE
sys/ROOT/11.2-RELEASE         NR     /            6.3G  2018-11-15 16:01
sys/ROOT/11.2-RELEASE@newbe   -      -          516.0K 2018-11-15 17:04
newbe
sys/ROOT/newbe                -      -            8.0K  2018-11-15 17:04
11.2-RELEASE@newbe            -      -          516.0K 2018-11-15 17:04
```
Activate the newbe BE to be booted after the restart.

```
# beadm list
BE     Active Mountpoint  Space Created
11.2-RELEASE NR / 6.4G 2018-11-15 16:01
newbe  - - 68.8M 2018-11-15 17:04

# beadm activate newbe
Activated successfully

# beadm list
BE     Active Mountpoint  Space Created
11.2-RELEASE N / 187.5M 2018-11-15 16:01
newbe  R - 6.3G 2018-11-15 17:04
```
Remove `newbe`. It will ask for additional confirmation as we renamed snapshot.

```
# beadm list
BE       Active Mountpoint  Space Created
11.2-RELEASE NR     /            6.4G 2018-11-15 16:01
newbe     -      -           68.8M 2018-11-15 17:04

# beadm destroy newbe
Are you sure you want to destroy 'newbe'?  
This action cannot be undone (y/[n]): y
Boot environment 'newbe' was created from existing snapshot
Destroy '11.2-RELEASE@newbe' snapshot? (y/[n]): y
Destroyed successfully

# beadm list
BE       Active Mountpoint  Space Created
11.2-RELEASE NR     /            6.4G 2018-11-15 16:01
```
Selection of BE at boot is integrated into the FreeBSD **loader**.
The **test** BE is selected to boot instead of the **default** one.
Not just FreeBSD loader...

Its integrated into other operating systems as well.

- BSDs
  - FreeBSD
  - HardenedBSD (rolling FreeBSD fork)

- Illumos
- OpenIndiana
- OmniOS
Not just FreeBSD loader ...

Its integrated idea implemented into other operating systems as well.

- BSDs
  - FreeBSD
  - HardenedBSD
    (rolling FreeBSD fork)
  - DragonFly BSD
- Illumos
  - OpenIndiana
  - OmniOS

**EuroBSDcon 2018 | Building Boot Environment Manager for DragonFly BSD**

As many users may be aware, DragonFly BSD’s recently declared the HAMMER2 filesystem to be stable and suitable for use. Since this is a CoW filesystem, and allows mounting of arbitrary snapshots of any PFS (analogous to ZFS datasets), we can define a custom scheme of creating and managing snapshots of any mounted HAMMER2 PFSes and updating the fstab accordingly.

**Turns out beadm(1) is a shell script.**

While investigating how beadm actually gets ZFS dataset information, I discovered it’s actually a very clever mix of sh and awk, which is not what I expected. Since I’m using C, things are a bit more complex. So I’ve had to get into the VFS layer of DragonFly BSD to query which filesystems are mounted, and then get and manipulate their names internally, which has quickly turned into a much more complex task than initially expected.
SUN Solaris and Oracle Solaris use GNU GRUB for the BE selection at boot.
What about Linux?

Instructions are fragmented and complicated.
What about Linux?

Instructions are fragmented and complicated.

- Only ONE Linux distribution allows root on ZFS install.
  - Antergos has ZFS option in installer.
  - Ubuntu comes with ZFS support but not for root.
What about Linux?

Instructions are fragmented and complicated.

- Only ONE Linux distribution allows root on ZFS install.
  Antergos has ZFS option in installer.
  Ubuntu comes with ZFS support but not for root.

- Howtos do not use `beadm` command integration.
What about Linux?

Instructions are fragmented and complicated.

- Only ONE Linux distribution allows root on ZFS install.
  Antergos has ZFS option in installer.
  Ubuntu comes with ZFS support but not for root.

- Howtos do not use `beadm` command integration.

- Howtos are complicated and VERY long.
What about Linux?

Instructions are **fragmented and complicated**.

- Only ONE Linux distribution allows root on ZFS install. Antergos has ZFS option in installer. Ubuntu comes with ZFS support but not for root.

- Howtos do not use `beadm` command integration.

- Howtos are complicated and **VERY long**.

- BTRFS alternative with `snapper` on openSUSE/SUSE. Red Hat deprecated BTRFS recently. Red Hat does not have BTRFS developers. Red Hat has lots of XFS developers. Fedora and CentOS will follow Red Hat.
What about Linux?

Instructions are **fragmented and complicated**.

- Only ONE Linux distribution allows root on ZFS install. Antergos has ZFS option in installer. Ubuntu comes with ZFS support but not for root.

- Howtos do not use `beadm` command integration.

- Howtos are complicated and **VERY** long.

- BTRFS alternative with `snapper` on openSUSE/SUSE. Red Hat deprecates BTRFS recently. Red Hat does not have BTRFS developers. Red Hat has lots of XFS developers. Fedora and CentOS will follow Red Hat.
What about BTRFS?

Can **BTRFS Snapshots** provide same functionality as **ZFS Boot Environments**?
What about BTRFS?

Can BTRFS Snapshots provide same functionality as ZFS Boot Environments?

Nope.
What about BTRFS?

Can BTRFS Snapshots provide same functionality as ZFS Boot Environments?

Nope.

Cite from System Recovery and Snapshot Management with Snapper for OpenSUSE Leap 15 Linux.

- Limitations
  A complete system rollback, restoring the complete system to the identical state as it was in when a snapshot was taken, is not possible.
What about BTRFS?

The **BTRFS Snapshots** limitations/excludes are as follows.

Also from *System Recovery and Snapshot Management with Snapper for OpenSUSE Leap 15 Linux*.

```
| /boot/grub2/*                  | /var/cache                  |
| /home                        | /var/crash                  |
| /opt                         | /var/lib/libvirt/images     |
| /var/opt                     | /var/lib/mailman           |
| /srv                         | /var/spool                  |
| /usr/local                   | /var/lib/named              |
| /tmp                         | /var/lib/mariadb            |
| /var/tmp                     | /var/lib/mysql              |
| /var/log                     | /var/lib/pgqsl              |
```
Default FreeBSD layout supports ZFS BEs

Default Auto (ZFS) `bsdinstall` option supports ZFS BEs.

```
# zfs list
NAME       USED  AVAIL  REFER MOUNTPOINT
zroot      339M  8.87G  88K   /zroot
zroot/ROOT 337M  8.87G  88K   none
zroot/ROOT/default 337M  8.87G  337M  /
zroot/tmp   88K  8.87G  88K   /tmp
zroot/usr   352K  8.87G  88K   /usr
zroot/usr/home 88K  8.87G  88K  /usr/home
zroot/usr/ports 88K  8.87G  88K  /usr/ports
zroot/usr/src  88K  8.87G  88K  /usr/src
zroot/var    596K  8.87G  88K   /var
zroot/var/audit 88K  8.87G  88K  /var/audit
zroot/var/crash 88K  8.87G  88K  /var/crash
zroot/var/log  152K  8.87G  152K  /var/log
zroot/var/mail 92K  8.87G  92K   /var/mail
zroot/var/tmp  88K  8.87G  88K   /var/tmp
```
Default FreeBSD layout supports ZFS BEs

The `/usr` and `/var` filesystems have **canmount** property set to **off**.

```bash
# zfs get -r canmount zroot
NAME                PROPERTY  VALUE     SOURCE
zroot               canmount  on        default
zroot/ROOT          canmount  on        default
zroot/ROOT/default  canmount  noauto    local
zroot/tmp           canmount  on        default
zroot/usr           canmount  off       local
zroot/usr/home      canmount  on        default
zroot/usr/ports     canmount  on        default
zroot/usr/src       canmount  on        default
zroot/var           canmount  off       local
zroot/var/audit     canmount  on        default
zroot/var/crash     canmount  on        default
zroot/var/log       canmount  on        default
zroot/var/mail      canmount  on        default
zroot/var/tmp       canmount  on        default
```
Default FreeBSD layout supports ZFS BEs

This way `/usr` and `/var` are placed on the `/` dataset the `zroot/ROOT/default` BE.

```bash
# df -g
Filesystem         1G-blocks Used Avail Capacity Mounted on
zroot/ROOT/default 9 0 8 4% / ← /usr & /var
devfs              0 0 0 100% /dev
zroot/tmp           8 0 8 0% /tmp
zroot/usr/home      8 0 8 0% /usr/home
zroot/usr/ports     8 0 8 0% /usr/ports
zroot/usr/src       8 0 8 0% /usr/src
zroot/var/audit     8 0 8 0% /var/audit
zroot/var/crash     8 0 8 0% /var/crash
zroot/var/log       8 0 8 0% /var/log
zroot/var/mail      8 0 8 0% /var/mail
zroot/var/tmp       8 0 8 0% /var/tmp
zroot              8 0 8 0% /zroot
```
Add beadm to FreeBSD

Just add beadm package or install sysutils/beadm port ... or download it.

- Package.
  
  ```
  # pkg install -y beadm
  ```

- Port.
  
  ```
  # make -C /usr/ports/sysutils/beadm install clean
  ```

  
  ```
  # fetch https://raw.githubusercontent.com/vermaden/beadm/master/beadm
  # chmod +x beadm
  # ./beadm list
  ```

<table>
<thead>
<tr>
<th>BE</th>
<th>Active</th>
<th>Mountpoint</th>
<th>Space</th>
<th>Created</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.2-RELEASE</td>
<td>NR</td>
<td>/</td>
<td>6.4G</td>
<td>2018-11-15 16:01</td>
</tr>
<tr>
<td>newbe</td>
<td>-</td>
<td>-</td>
<td>80.2M</td>
<td>2018-11-15 17:04</td>
</tr>
</tbody>
</table>
These tools on FreeBSD are `freebsd-update(8)` and `pkg(8)`.
Using update/upgrade tools with BEs

These tools on FreeBSD are `freebsd-update(8)` and `pkg(8)`.

- On FreeBSD by default these tools operate on running system.
Using update/upgrade tools with BEs

These tools on FreeBSD are `freebsd-update(8)` and `pkg(8)`.

- On FreeBSD by default these tools operate on running system.
- By contrast on Solaris/Illumos by default they operate on newly created BE and require reboot into that BE.
Using update/upgrade tools with BEs

These tools on FreeBSD are `freebsd-update(8)` and `pkg(8)`.

- On FreeBSD by default these tools operate on running system.
- By contrast on Solaris/Illumos by default they operate on newly created BE and require reboot into that BE.

**PKG(8)** - [https://man.freebsd.org/pkg](https://man.freebsd.org/pkg)

```
-c (chroot path), --chroot (chroot path) pkg will chroot in the (chroot path) environment.
-r (root directory), --rootdir (root directory) pkg will install all packages within the specified (root directory).
```

**FREEBSD-UPDATE(8)** - [https://man.freebsd.org/freebsd-update](https://man.freebsd.org/freebsd-update)

```
-b basedir    Operate on a system mounted at basedir. (default: /)
-d workdir    Store working files in workdir. (default: /var/db/freebsd-update)
```
Emulate Solaris/Illumos behaviour on FreeBSD

Example **upgrade of packages** in the newly created BE for that purpose.

```bash
# beadm create safe
Created successfully

# beadm mount safe
Mounted successfully on '/tmp/BE-safe.ostSai22'

# pkg -r /tmp/BE-safe.ostSai22 update -f
(…)

# pkg -r /tmp/BE-safe.ostSai22 upgrade
(…)

# pkg -r /tmp/BE-safe.ostSai22 info -s feh
feh-2.27.1  438KiB

# pkg -r / info -s feh
feh-2.27   438KiB

# pkg info -s feh
feh-2.27   438KiB
```
Emulate Solaris/Illumos behaviour on FreeBSD

Example `fetch security updates` in the newly created BE for that purpose.

```bash
# beadm create safe
Created successfully

# beadm mount safe /tmp/safe
Mounted successfully on '/tmp/safe'

# rm -rf /var/db/freebsd-update

# freebsd-update -b /tmp/safe fetch
freebsd-update: Directory does not exist or is not writable: /var/db/freebsd-update

# freebsd-update -b /tmp/safe -d /tmp/safe/var/db/freebsd-update fetch
Looking up update.FreeBSD.org mirrors... 3 mirrors found.
Fetching metadata signature for 11.2-RELEASE from update4.freebsd.org... done.
Fetching metadata index... done.
Inspecting system... done.
Preparing to download files... done.

No updates needed to update system to 11.2-RELEASE-p0.
```
First one was `manageBE` script which had some problems and complicated syntax.

- Create a new BE.
  ```
  # manageBE create -n 9_20120321 -s 9_20120317 -p zroot
  manageBE: cannot create /zroot/ROOT/9_20120321/boot/loader.conf: No such file or directory
  manageBE: cannot create /zroot/ROOT/9_20120321/etc/fstab: No such file or directory
  The new Boot-Environment is ready to be updated and/or activated.
  ```

- List existing BEs.
  ```
  # manageBE list
  Poolname: zroot

  BE                        Active Active Mountpoint                      Space
  Name                      Now    Reboot -                               Used
  ----                      ------ ------ ----------                      -----  
  9_20120321                no     no     /ROOT/9_20120321                 145M
  9_20120317                yes    yes    /                               1.59G
  
  Used by BE snapshots: 1.99G
Current upstream `beadm` source and alternatives/forks.
History/Mods/Forks/Alternatives

Current upstream `beadm` source and alternatives/forks.

- The `manageBE` source - https://outpost.h3q.com/patches/manageBE/manageBE
Current upstream beadm source and alternatives/forks.

- The manageBE source - https://outpost.h3q.com/patches/manageBE/manageBE

- Current beadm implementation - https://github.com/vermaden/beadm ➔ source for beadm package
  - Fork with separate boot pool support - https://bitbucket.org/aasoft/beadm ➔ fork of vermaden/beadm
  - Fork with support for Linux system - https://github.com/b333z/beadm ➔ fork of vermaden/beadm
  - Original HOWTO: FreeBSD ZFS Madness thread - https://forums.freebsd.org/threads/31662/
Current upstream `beadm` source and alternatives/forks.

- The `manageBE` source - https://outpost.h3q.com/patches/manageBE/manageBE
- Current `beadm` implementation - https://github.com/vermaden/beadm
  - Fork with separate boot pool support - https://bitbucket.org/aasoft/beadm
  - Fork with support for Linux system - https://github.com/b333z/beadm
  - Original HOWTO: FreeBSD ZFS Madness thread - https://forums.freebsd.org/threads/31662/
- The `zedenv` in Python 3.6 with support for FreeBSD and Linux - https://github.com/johnramsden/zedenv
  - Currently at alpha stage of development (experimental) - not production ready.
  - Needs `python36` and `py36-setuptools` packages to work.
  - Supports plugins but currently comparable with `beadm` features or its forks.
Current upstream `beadm` source and alternatives/forks.

- The `manageBE` source - [https://outpost.h3q.com/patches/manageBE/manageBE](https://outpost.h3q.com/patches/manageBE/manageBE)
- Current `beadm` implementation - [https://github.com/vermaden/beadm](https://github.com/vermaden/beadm) ➔ source for `beadm` package
  - Fork with separate boot pool support - [https://bitbucket.org/aasoft/beadm](https://bitbucket.org/aasoft/beadm) ➔ fork of `vermaden/beadm`
  - Fork with support for Linux system - [https://github.com/b333z/beadm](https://github.com/b333z/beadm) ➔ fork of `vermaden/beadm`
  - Original HOWTO: FreeBSD ZFS Madness thread - [https://forums.freebsd.org/threads/31662/](https://forums.freebsd.org/threads/31662/)
- The `zedenv` in Python 3.6 with support for FreeBSD and Linux - [https://github.com/johnramsden/zedenv](https://github.com/johnramsden/zedenv)
  - Currently at alpha stage of development (experimental) - not production ready.
  - Needs `python36` and `py36-setuptools` packages to work.
  - Supports plugins but currently comparable with `beadm` features or its forks.
- Ansible `beadm` module - [https://docs.ansible.com/ansible/latest/modules/beadm_module.html](https://docs.ansible.com/ansible/latest/modules/beadm_module.html)
Current upstream `beadm` source and alternatives/forks.

- The `manageBE` source - [https://outpost.h3q.com/patches/manageBE/manageBE](https://outpost.h3q.com/patches/manageBE/manageBE)

- Current `beadm` implementation - [https://github.com/vermaden/beadm](https://github.com/vermaden/beadm) ➔ source for `beadm` package

  - Fork with separate boot pool support - [https://bitbucket.org/aasoft/beadm](https://bitbucket.org/aasoft/beadm) ➔ fork of `vermaden/beadm`
  
  - Fork with support for Linux system - [https://github.com/b333z/beadm](https://github.com/b333z/beadm) ➔ fork of `vermaden/beadm`

  - Original [HOWTO: FreeBSD ZFS Madness](https://forums.freebsd.org/threads/31662) thread - [https://forums.freebsd.org/threads/31662](https://forums.freebsd.org/threads/31662)

- The `zedenv` in Python 3.6 with support for FreeBSD and Linux - [https://github.com/johnramsden/zedenv](https://github.com/johnramsden/zedenv)

  - Currently at alpha stage of development (experimental) - not production ready.

  - Needs `python36` and `py36-setuptools` packages to work.

  - Supports plugins but currently comparable with `beadm` features or its forks.

- Ansible `beadm` module - [https://docs.ansible.com/ansible/latest/modules/beadm_module.html](https://docs.ansible.com/ansible/latest/modules/beadm_module.html)

- New `bectl` FreeBSD 12.x base system utility compatible with `beadm` command.
The `bectl` command

New FreeBSD 12.x base system command - `bectl` - to manage ZFS Boot Environments.

# `bectl`

Usage: `bectl {-h | -? | subcommand [args ...]}

- `bectl activate [-t] beName`
- `bectl create [-e {nonActiveBe | -e beName@snapshot}] beName`
- `bectl create beName@snapshot`
- `bectl destroy [-F] {beName | beName@snapshot}`
- `bectl export sourceBe`
- `bectl import targetBe`
- `bectl jail [{-b | -U}] [{-o key=value | -u key}]... bootenv [utility [argument ...]]`
- `bectl list [-a] [-D] [-H] [-s]`
- `bectl mount beName [mountpoint]`
- `bectl rename origBeName newBeName`
- `bectl {ujail | unjail} (jailID | jailName | bootenv)`
- `bectl {umount | unmount} [-f] beName`
The `bectl` is written in C language

```c
static int
bectl_cmd_activate(int argc, char *argv[])
{
    int err, opt;
    bool temp;

    temp = false;
    while ((opt = getopt(argc, argv, "t")) != -1) {
        switch (opt) {
        case 't':
            temp = true;
            break;
        default:
            fprintf(stderr, "bectl activate: unknown option '-%c'\n",
                    optopt);
            usage(false);
            return (0);
        }
    }

    argc -= optind;
    argv += optind;

    if (argc != 1) {
        fprintf(stderr, "bectl activate: wrong number of arguments\n");
        usage(false);
        return (0);
    }

    /* activate logic goes here */
    if ((err = be_activate(be, argv[0], temp)) != 0)
        /* XXX TODO: more specific error msg based on err */
        printf("did not successfully activate boot environment %s\n", argv[0]);
    else
        printf("successfully activated boot environment %s\n", argv[0]);

    if (temp)
        printf("for next boot\n");

    return (err);
}
```
Difference between `beadm` and `bectl` usage

All commands that work with `beadm` will work with `bectl` tool without modifications.

```
# beadm create ASD
Created successfully
#

# beadm activate ASD
Activated successfully
#

# beadm list
BE   Active Mountpoint  Space   Created
11.2 N      /            7.0G  2018-11-15 16:01
ASD  R      -            6.9M  2018-11-15 17:29
#

# beadm destroy ASD
Are you sure you want to destroy 'ASD'? 
This action cannot be undone (y/[n]): y
Destroyed successfully
#

# beadm rename ASD NEW
Renamed successfully
#
```

```
# bectl create ASD
# (silent creation)
#

# bectl activate ASD
successfully activated boot environment ASD
#

# bectl list
BE   Active Mountpoint  Space   Created
12.0 N      /            471M  2018-11-15 13:15
ASD  R      -            448K  2018-11-15 14:03
#

# bectl destroy ASD
# (no confirmation for destroy)
#

# bectl rename ASD NEW
# (silent rename)
```
New features/commands in **bectl** tool

New **jail/unjail** command to start FreeBSD Jail within ZFS Boot Environment.

```
freebsd12 # hostname
freebsd12.local
freebsd12 # sysctl security.jail.jailed
security.jail.jailed: 0
freebsd12 # bectl jail ASD
# hostname
ASD
# sysctl security.jail.jailed
security.jail.jailed: 1
# (you are directly in newly created FreeBSD Jail within ‘ASD’ ZFS Boot Environment)
```

Meanwhile on the FreeBSD Host ...

```
freebsd12 # mount | grep ASD
zroot/ROOT/ASD on /tmp/be_mount.WR1F (zfs, local, noatime, nfsv4acls)
freebsd12 # jls -a

JID  IP Address      Hostname      Path
1    ASD             /tmp/be_mount.WR1F
```

**New bectl Features**
New features/commands in `bectl` tool

New `export/import` command that sends ZFS Boot Environment into/from plain file.

```
# bectl export ASD
bectl export: must redirect output
# bectl export ASD > ASD.raw
# file ASD.raw | tr ',' '
' ASD.raw: ZFS shapshot (little-endian machine)
  version 17
  type: ZFS
  destination GUID: D9 72 9E 43 9C CF F9 A2
  name: 'zroot/ROOT/NEW@2018-11-15-15:39:25'

# bectl import NEW.raw
bectl import: input can not be from terminal
# bectl import NEW < NEW.raw
# bectl list
BE  Active Mountpoint  Space  Created
12.0  NR   /  905M  2018-11-15 13:24
ASD  -    -  448K  2018-11-15 15:39
NEW  -    -  471M  2018-11-15 16:44
```
New LUA based **loader** in FreeBSD 12.x

New LUA based **loader** that deprecates the old Forth based **loader**.
New LUA based loader in FreeBSD 12.x

New loader menu is not perfect - longer (5!) BE names overlap on the menu border.
New LUA based loader in FreeBSD 12.x

There is no list of BEs - you can only switch between existing BEs in sequence.
New LUA based loader in FreeBSD 12.x

With new **loader** you do not need ZFS boot pool to have GELI encrypted ZFS root!
New LUA based loader in FreeBSD 12.x

With new loader you do not need ZFS boot pool to have GELI encrypted ZFS root!

- Available straight in `bsdinstall` without any HOWTOs.
New LUA based loader in FreeBSD 12.x

With new **loader** you do not need ZFS boot pool to have GELI encrypted ZFS root!

- Available straight in `bsdinstall` without any HOWTOs.
- Setup supported by both `bectl` and `beadm` tools.
New LUA based loader in FreeBSD 12.x

With new **loader** you do not need ZFS boot pool to have GELI encrypted ZFS root!

- Available straight in `bsdinstall` without any HOWTOs.
- Setup supported by both `bectl` and `beadm` tools.
- By default 256-bit AES-XTS is used.
With new **loader** you do not need ZFS boot pool to have GELI encrypted ZFS root!

- Available straight in `bsdinstall` without any HOWTOs.
- Setup supported by both `bectl` and `beadm` tools.
- By default 256-bit AES-XTS is used.
- Works both on UEFI and BIOS (Legacy/CSM/...) boot type.
New LUA based loader in FreeBSD 12.x

With new loader you do not need ZFS boot pool to have GELI encrypted ZFS root!

- Available straight in `bsdinstall` without any HOWTOs.
- Setup supported by both `bectl` and `beadm` tools.
- By default 256-bit AES-XTS is used.
- Works both on UEFI and BIOS (Legacy/CSM/...) boot type.

```bash
# gpart show
⇒  40  16777136  ada0  GPT (8.0G)
    40   1024   1  freebsd-boot (512K)
  1064    984     - free - (492K)
  2048  16773120   2  freebsd-zfs (8.0G)
16775168   2008     - free - (1.0M)

# geli status
Name  Status  Components
ada0p2.eli  ACTIVE  ada0p2
```
New LUA based loader in FreeBSD 12.x

What to choose in `bsdinstall` to create such ZFS root GELI encrypted setup.
Now type in GELI password you want to use.

![FreeBSD Installer](image)

**ZFS Configuration**

Enter a strong passphrase, used to protect your encryption keys. You will be required to enter this passphrase each time the system is booted.

- [Use alpha-numeric, punctuation, TAB or ENTER]
New LUA based loader in FreeBSD 12.x

... and wait till GELI finishes the initialization.
New LUA based loader in FreeBSD 12.x

Here is how boot of such GELI encrypted password prompt looks like on BIOS type.

```
GELI Passphrase for disk0p2: _
```
New LUA based loader in FreeBSD 12.x

Here is how boot of such GELI encrypted password prompt looks like on BIOS type.

```
GELI Passphrase for disk0p2:
Calculating GELI Decryption Key for disk0p2: 745242 iterations...
```
New LUA based loader in FreeBSD 12.x

... and after the password is being accepted you get the loader FreeBSD menu.
Questions?

Sławomir Wojciech Wojtczak
vermaden@interia.pl
vermaden.wordpress.com
twitter.com/vermaden

https://is.gd/BECTL
Thank You!

Sławomir Wojciech Wojtczak
vermaden@interia.pl
vermaden.wordpress.com
twitter.com/vermaden

https://is.gd/BECTL