

---

# **zedenv Documentation**

*Release 0.2.0-alpha*

**John Ramsden**

**Jul 04, 2018**



---

## Contents:

---

<b>1</b>	<b>Documentation</b>	<b>3</b>
1.1	Install . . . . .	3
1.2	System Setup . . . . .	4
1.3	Basic Usage . . . . .	5
1.4	Commands . . . . .	7
1.5	Plugins . . . . .	9
<b>2</b>	<b>Indices and tables</b>	<b>13</b>



zedenv is a utility to manage Boot Environments using ZFS. zedenv is still in the alpha stage on Linux and FreeBSD. Take caution when using.



## 1.1 Install

zedenv requires python 3.6+, `pyzfscmds`, and ZFS running as the root filesystem.

It can be installed a few ways:

- From the `setup.py` directly.
- From the `Makefile`.
- From the [Arch AUR](#).

First, clone the git repos.

```
git clone https://github.com/johnramsdend/pyzfscmds
git clone https://github.com/johnramsdend/zedenv
```

### 1.1.1 Makefile and setup.py

To install without polluting your system, you can also create a directory somewhere and install in a `venv`, otherwise install to the system.

Optionally, create a `venv` and activate.

```
python3.6 -m venv venv
. venv/bin/activate
```

#### **setup.py**

Enter the repos and install.

```
cd pyzfscmds
python setup.py install

cd ../zedenv pyzfscmds
python setup.py install
```

## Makefile

Enter the packaging directory in the repos run make, pyzfscmds must be installed first.

```
cd pyzfscmds/packaging
make

cd ../../zedenv/packaging
make
```

## 1.2 System Setup

In order to use boot environments your system needs to be set up in a certain manner.

The main dataset that is used for your root file system, can also be thought of as your boot environment. Anything in this dataset, or in a dataset under it, is what constitutes a boot environment.

### 1.2.1 Dataset Configuration

To put your system in a compatible configuration, your boot environments for your system should be kept in a ‘Boot Environment root’. In most configurations this would be <pool>/ROOT. However its location is not important, and it can be located anywhere within a pool. What’s important is that it does not have any child datasets that are not in a boot environment.

The common practice is to start with a ‘default’ boot environment. This would be the dataset <pool>/ROOT/default. If a system is setup in this manner, it would be the most basic boot environment compatible system.

This ‘default’ dataset could have the entire system installed into it. Upon creating new boot environments, it would be cloned and the entire system would be in the new boot environment. A better practice would be to keep some datasets separate from the boot environment. Putting parts of the system that can be shared between boot environments, in these separate datasets is good practice. For example, one might want to keep their log, or home directories separate.

### 1.2.2 Examples

Here are a few examples of possible setups with a few different systems.

#### FreeBSD

The default FreeBSD configuration is a great example of a hierarchy that is setup to use boot environments by default. After a root on ZFS install, the system has a new boot environment, that is usable by default.

Any dataset that is set to `canmount=off`, means it will not be mounted and its data will be stored in the boot environment.

In the default setup this means the data of `/usr`, and `/var` will change between boot environments, but any dataset set `canmount=on`, will not be in the boot environment, and the data will be persistent between every boot environment.

NAME	CANMOUNT	MOUNTPOINT
zroot	on	/zroot
zroot/ROOT	on	none
zroot/ROOT/default	noauto	/
zroot/tmp	on	/tmp
zroot/usr	off	/usr
zroot/usr/home	on	/usr/home
zroot/usr/ports	on	/usr/ports
zroot/usr/src	on	/usr/src
zroot/var	off	/var
zroot/var/audit	on	/var/audit
zroot/var/crash	on	/var/crash
zroot/var/log	on	/var/log
zroot/var/mail	on	/var/mail
zroot/var/tmp	on	/var/tmp

## Arch Linux

Here is an Arch Linux system, with an extensive dataset setup.

NAME	CANMOUNT	MOUNTPOINT
vault/ROOT	on	none
vault/ROOT/default-3	noauto	/
vault/ROOT/default-4	noauto	/
vault/home	on	legacy
vault/usr	off	/usr
vault/usr/local	on	legacy
vault/var	off	/var
vault/var/cache	on	legacy
vault/var/cache/pacman	on	legacy
vault/var/lib	off	/var/lib
vault/var/lib/docker	on	legacy
vault/var/lib/libvirt	on	legacy
vault/var/lib/systemd	off	/var/lib/systemd
vault/var/lib/systemd/coredump	on	legacy
vault/var/log	on	legacy
vault/var/log/journal	on	legacy

In this example, while it looks like `/var`, `/var/lib` `/var/lib/systemd`, and `/usr` are outside of the boot environment, they have actually been set to `canmount=off` meaning they're not mounted and are only there to create the ZFS dataset structure. This will put their data in the boot environment dataset. Their properties will be inherited by any child datasets.

This allows us to take any datasets we would like to share between boot environments, and create them under these datasets in hierarchy that is clear and easy to understand. It means the user's `/home`, `/usr/local` and `/var/log` directories, among others, data will stay the same when switching between boot environments.

## 1.3 Basic Usage

zedenv can be used to manage boot environments using ZFS. If your system is set up in a way compatible with boot environments, you can start using them right away.

Create and activate a new Boot Environment.

```
$ zedenv create default-0
$ zedenv activate default-0
```

This will make it the Boot Environment used on reboot.

```
$ zedenv list
```

Name	Active	Mountpoint	Creation
default	N	-	Wed-May-23-23:48-2018
default-0	R	/	Thu-May-24-23:54-2018

This can be shown with a list, command. The boot environment currently being used will have a 'N' in the active column signifying the boot environment is being used now. An 'R' in the active column means this environment will be used on reboot.

In order to integrate with a bootloader, an extra flag '-b/--bootloader' must be used to specify a bootloader plugin. The plugin will make the necessary changes to boot from the new Boot Environment.

If you expect you will always be using a certain bootloader, you can set the `org.zedenv.bootloader` property on your boot environments, and the bootloader plugin will be used without you having to specify.

```
$ zedenv set org.zedenv.bootloader=<bootloader plugin>
```

Plugins available for your system can be listed with `zedenv --plugins`.

If you're using `zedenv` to activate a boot environment, and a plugin isn't available, you may need to edit some config files to specify the new dataset, depending on your bootloader.

Usage information can be given at any time by running `zedenv --help`.

```
Usage: zedenv [OPTIONS] COMMAND [ARGS]...

ZFS boot environment manager cli

Options:
  --version
  --plugins List available plugins.
  --help   Show this message and exit.

Commands:
  activate Activate a boot environment.
  create   Create a boot environment.
  destroy  Destroy a boot environment or snapshot.
  get      Get boot environment properties.
  list     List all boot environments.
  mount    Mount a boot environment temporarily.
  rename   Rename a boot environment.
  set      Set boot environment properties.
  umount   Unmount a boot environment.
```

More specific information about a specific subcommand can be requested as well.

```
zedenv create --help
```

```
Usage: zedenv create [OPTIONS] BOOT_ENVIRONMENT
```

(continues on next page)

(continued from previous page)

```
Create a boot environment.
```

```
Options:
```

```
-v, --verbose      Print verbose output.
-e, --existing TEXT Use existing boot environment as source.
--help            Show this message and exit.
```

## 1.4 Commands

The following commands are available

- `activate` - Activate a boot environment.
- `create` - Create a boot environment.
- `destroy` - Destroy a boot environment or snapshot.
- `get` - Print boot environment properties.
- `list` - List all boot environments.
- `mount` - Mount a boot environment temporarily.
- `rename` - Rename a boot environment.
- `set` - Set boot environment properties.
- `umount` - Unmount a boot environment.

### 1.4.1 Activate

The `activate` is used to enable an already created boot environment. After activation, the boot environment will be used upon reboot.

```
zedenv activate [OPTIONS] BOOT_ENVIRONMENT
```

Option	Description
<code>-v, --verbose</code>	Print verbose output.
<code>-b, --bootloader TEXT</code>	Use bootloader type.
<code>-y, --noconfirm</code>	Assume yes in situations where confirmation is needed.
<code>-n, --noop</code>	Print what would be destroyed but don't apply.
<code>--help</code>	Show this message and exit.

### 1.4.2 Create

Create a new boot environment.

```
zedenv create [OPTIONS] BOOT_ENVIRONMENT
```

Option	Description
<code>-v, --verbose</code>	Print verbose output.
<code>-e, --existing TEXT</code>	Use existing boot environment as source.
<code>--help</code>	Show this message and exit.

### 1.4.3 Destroy

Destroy a boot environment or snapshot.

```
zedenv destroy [OPTIONS] BOOT_ENVIRONMENT
```

Option	Description
-v, --verbose	Print verbose output.
-b, --bootloader TEXT	Use bootloader type.
-y, --noconfirm	Assume yes in situations where confirmation is needed.
-n, --noop	Print what would be destroyed but don't apply.
--help	Show this message and exit.

### 1.4.4 Get

Print boot environment properties.

```
zedenv get [OPTIONS] BOOT_ENVIRONMENT
```

### 1.4.5 List

List all boot environments.

```
zedenv list [OPTIONS]
```

Option	Description
-v, --verbose	Print verbose output.
-D, --spaceused	Display the full space usage for each boot environment.
-H, --scripting	Scripting output.
-O, --origin	Display origin.
--help	Show this message and exit.

### 1.4.6 Mount

Mount a boot environment temporarily.

```
zedenv mount [OPTIONS] BOOT_ENVIRONMENT [MOUNTPOINT]
```

Option	Description
-v, --verbose	Print verbose output.
--help	Show this message and exit.

### 1.4.7 Rename

Rename a boot environment.

```
zedenv rename [OPTIONS] BOOT_ENVIRONMENT NEW_BOOT_ENVIRONMENT
```

Option	Description
<code>-v, --verbose</code>	Print verbose output.
<code>--help</code>	Show this message and exit.

### 1.4.8 Set

Set boot environment properties.

```
zedenv set [OPTIONS] BOOT_ENVIRONMENT
```

Option	Description
<code>-v, --verbose</code>	Print verbose output.
<code>--help</code>	Show this message and exit.

### 1.4.9 Umount

Unmount a boot environment.

```
zedenv umount [OPTIONS] BOOT_ENVIRONMENT
```

Option	Description
<code>-v, --verbose</code>	Print verbose output.
<code>--help</code>	Show this message and exit.

## 1.5 Plugins

As of now plugins for the following bootloaders exist:

- FreeBSD's loader - `freebsdloader`
- `systemdboot` - `systemdboot`

In order to integrate with a bootloader, an extra flag `'-b/--bootloader'` must be used to specify a bootloader plugin. The plugin will make the necessary changes to boot from the new Boot Environment.

If you expect you will always be using a certain bootloader, you can set the `org.zedenv:bootloader` property on your boot environments, and the bootloader plugin will be used without you having to specify.

```
$ zedenv set org.zedenv:bootloader=<bootloader plugin>
```

Plugins available for your system can be listed with `zedenv --plugins`.

#### 1.5.1 freebsdloader

The `freebsdloader` plugin is quite simple. During activation, it will respect `/etc/rc.d/zfsbe` if it exists, and set, all datasets to `canmount=noauto`, otherwise the datasets will be set `canmount=on`.

It will also change the root dataset to mount from in `/boot/loader.conf`, and `/boot/loader.conf.local`.

The current `/boot/zfs/zpool.cache` will also be copied into the newly activated boot environment.

## 1.5.2 systemdboot

Multiple kernels can be managed with this plugin and systemd-boot, but it will require changing the mountpoint of the esp (EFI System Partition).

### Problem With Regular Mountpoint

Usually the `$esp` would get mounted at `/boot` or `/boot/efi`. The kernels would sit in the root of the `$esp`, and the configs for systemdboot in `$esp/loader/`.



The configs would then reference kernels in the root directory.

```

title           Arch Linux
linux           vmlinuz-linux
initrd          intel-ucode.img
initrd          initramfs-linux.img
options         zfs=bootfs rw
  
```

The problem with this method, is multiple kernels cannot be kept at the same time. Therefore this hierarchy is not conducive to boot environments.

### Alternate Mountpoint

First, remount the `$esp` to a new location, the default is `/mnt/efi`.

If you would like to explicitly specify the mountpoint used, you can set the `org.zedenv.systemdboot:esp` property on your current boot environment, and the plugin will use the specified location:

```
zfs set org.zedenv.systemdboot:esp='/mnt/efi' zpool/ROOT/default
```

Don't forget to change the mount point in `/etc/fstab`.

```
UUID=9F8A-F566          /mnt/efi  vfat      rw,defaults,errors=remount-ro 0 2
```

Now, make a subdirectory `$esp/env`, kernels will be kept in a subdirectory of this location.

The bootloader configuration can now use a different path for each boot environment.

So the 'default' boot environment config, located at `$esp/loader/entries/zedenv-default.conf`, would look something like:

```

title           Arch Linux
linux           /env/zedenv-default/vmlinuz-linux
initrd          /env/zedenv-default/intel-ucode.img
  
```

(continues on next page)

(continued from previous page)

```
initrd      /env/zedenv-default/initramfs-linux.img
options    zfs=zpool/ROOT/default rw
```

To make the system happy when it looks for kernels at `/boot`, this directory should be bindmounted to `/boot`.

**Bindmount** `/mnt/efi/env/zedenv-default` to `/boot` in `/etc/fstab`.

```
/mnt/efi/env/zedenv-default /boot none rw,defaults,errors=remount-ro,bind ↵
↵0 0
```

If this directory is not here, the kernels will not be updated when the system rebuilds the kernel.

Once our system is set up in the proper configuration, `zedenv` will update the bootloader, and `fstab` - if requested - when a new boot environment is activated.

It will also update the configuration described above, asking you if the modifications that made are correct. You will have a chance to inspect and change them if they are not.

If you are confident and the changes it is making, and do not wish to inspect them, adding the `--noconfirm/-y` flag will run the command without asking for confirmation.



## CHAPTER 2

---

### Indices and tables

---

- `genindex`
- `search`



**B**

Basic Usage, 5

**C**

Commands, 7

**I**

Install, 3

**P**

Plugins, 9

**S**

System Setup, 4