Using Boot Environments at Scale

Allan Jude – allanjude@freebsd.org
Klara Systems

Introduction

- FreeBSD Server Admin since 2001
- 4 Years as FreeBSD committer
  - ZFS, installer, boot loader, GELI (FDE)
- FreeBSD Core Team (2016 - Present)
- Co-Author of "*FreeBSD Mastery: ZFS*" and "*FreeBSD Mastery: Advanced ZFS*" with Michael W. Lucas – ZFSBook.com
- Host of BSDNow.tv Podcast
- GSoC Mentor for bectl(8)

What is a Boot Environment (BE)

- Similar in concept to NanoBSD
  - Divides the disk into 2 partitions (firmware images)
  - Install the stock image to both
  - At upgrade time, overwrite the inactive image
  - Boot-once to the newer image. If it fails, or is otherwise unserviceable, reboot to good image
  - If the new image is accepted, configure it as the default for all future reboots
  - Repeat process for next upgrade

ZFS Boot Environments

- ZFS takes this concept further
- ZFS allows you to have many filesystems, without needing to partition your disk
- Separate the OS (root FS) from user data (home directories, logs, databases)
- ZFS has instant snapshots and clones
- Snapshot and clone the root filesystem before you make changes or upgrade

How?

- Now you have multiple different ‘versions’ of your root filesystem to choose from
- Modern FreeBSD boot loader allows you to choose from different rootfs at boot
- Now you can ‘revert’ an upgrade without losing changes to home directories, logs, databases or other filesystems, further separating the ‘OS’ from the ‘Data’

Control

- The flexibility of ZFS puts you in control
- Any files in the filesystem mounted as / are treated as part of the operating system
- Any files in other filesystems, are retained, no matter what ‘version’ of the OS you boot
- Packages (/usr/local) and the pkg database (/var/db/pkg) are included in /. This allows you to ‘undo’ a `pkg upgrade`

Default BE layout

```
NAME                 USED      REFER  MOUNTPOINT
zroot                19.5G        88K  /zroot
zroot/ROOT           1.67G        88K  none
zroot/ROOT/default    1.67G     1.67G  /
zroot/tmp              88K        88K  /tmp
zroot/usr            12.3G        88K  /usr
zroot/usr/obj        12.3G     8.03G  /usr/obj
zroot/usr/home       140M      140M  /usr/home
zroot/var             153M        88K  /var
zroot/var/audit        88K        88K  /var/audit
zroot/var/crash      152M      152M  /var/crash
zroot/var/log        352K      352K  /var/log
zroot/var/mail       132K      132K  /var/mail
zroot/var/tmp          88K        88K  /var/tmp
```

That's Great, But I Already Knew That

Going Further

- When upgrading a system, we wanted to replace the entire OS with a newer version
- So we just install a new boot environment
- But what about /etc? My machine needs to have a configured network for puppet to replace the rest of the configuration
- Let’s make /etc its own filesystem, it can persist through the upgrade this way

What Could Possibly Go Wrong?

Not So Fast...

- A lot of boot things depend upon /etc
- No /etc/fstab, no /etc/rc, no /etc/ttys
- Don’t want to `etcupdate` or `mergemaster`
- Another Idea: Steal from NanoBSD: A read-only /etc recreated at boot from /cfg
- Then learned about `init_script` see loader(8)
- Use `init_script` to mount /cfg. Replace persistent files in /etc with symlinks to /cfg

What do you run from init_script?

```
mount -p | while read _dev _mp _type _rest;
do
        [ $_mp = "/" ] || continue
        if [ $_type = "zfs" ] ; then
                pool=${_dev%%/*}
                zfs mount ${pool}/cfg
        fi
        break
done
```

So how does that work?

- /cfg populated with ~10 files we care about
- Configure network (rc.conf.*), sysctls, SSHd keys, fstab (for jails), etc
- Rest of /etc can be replaced with stock files
- Never have to merge /etc/rc.d files
- Originally had to manually recreate symlinks because our BE images were stock
- Used a VM and a script to make new BEs

How do you deploy a Boot Env?

1. Create an image:
    a. `zfs snapshot zroot/ROOT/bename@snapname`
    b. `zfs send -pec zroot/ROOT/bename @snapname | xz -T0 -9 > bename.zfs.xz`
2. Apply the image:
    a. `fetch -o - https://svr/bename.zfs.xz | unxz | zfs recv zroot/ROOT/newbe`
3. Boot Once:
    a. `zfsbootcfg zfs:zroot/ROOT/newbe:`

Shortcomings

- We were still doing `pkg upgrade -f` in a chroot for the base system BE plus each jail
- Building images was painfully manual
- Missing a step or file almost every time
- Bootstrapping a fresh install was still a bunch of manual work, over slow IPMI
- Not usable by anyone else, too many rough edges and sharp corners

Using BEs at Scale

- Over 100 servers, 38 DCs, 11 countries
- Only myself and 1 full time sysadmin
- Mix of versions, 10.4, 11.1, 12-CURRENT
- `freebsd-update upgrade` too manual
- `zfs recv; zfsbootcfg; reboot` takes less than a minute, and fails gracefully
- Upgrade remote machines with confidence even without console access

Not Just For Packages Anymore

- Poudriere is the tool used to build the official FreeBSD binary packages, very quickly
- Uses Jails, and optionally ZFS and TMPFS
- Starts 1 jail per core, builds one package in each jail, only dependencies installed, no network
- You can use it to build your own customized package (ports tree * freebsd version * arch * set)
- Supports: iso, iso+(z)mfs, usb, usb+(z)mfs, rawdisk, zfsrawdisk, tar, firmware, embedded

A Better Way to Build

- During the development of this upgrade procedure, I happened to be talking with Baptiste Daroussin (bapt@) who informed me of his work on `poudriere image`
- Designed to create customized VM or USB images. Used at Gandi to build FreeBSD images for their Public Cloud Customers
- Supports overlays and preinstalled packages

Poudriere Image ZFS BE Support

- After discussing it, we decided that `zfs send` should be added as an output format
- Add `-t zfssend` (full pool replication stream) and `-t zfssend+be` (just the BE)
- Modified overlay support to handle symlinks
- Added support for a ‘ZFS Layout’ config file, in the same format used by `bsdinstall`
- Control what files are part of the Boot Env.

What About Brand New Systems?

- Previously, we used IPMI Remote Media feature to run bootonly.iso on each machine and manually ran through `bsdinstall`
- No PXEBOOT with only 1-3 servers per DC
- Now we make our own iso+mfs image
- Prompts for some config details (no DHCP)
- Partition disks and create an empty pool
- Then `zfs recv` a full pool image on to it

Poudriere Image for Everyone

- Many recent enhancements upstreamed
- Work-in-Progress can be found on my github
- Use it to create your own custom images
- Builds from poudriere jails you have already created to build packages. Can create from releases without having to compile!
- New Image Formats? vmdk, qcow2, vhd, MBR (CSM & EFI), GPT (CSM, EFI, both), <yours>

Enhancing Poudriere Image

- Needs better naming for image types
- Should support many more combinations
- Replace tools/boot/rootgen.sh
- Should integrate various ‘Cloudware’
- Replicate features of ‘release’ building bits
- Support for post-build scripts (chroot)
- More appliance building features - talk to me
- What features do you need?

Improvements to Come

- Automate the process of confirming an image is good, some combination of:
  - Uptime, Minimum level of served, self-tests
- Use `bectl` or `libbe` to set the shiney new boot environment as the default for future boots
- Extend `zfsbootcfg`
  - Currently just a string
  - Delphix uses failure count down. If counter reaches zero, boots into a phone-home rescue mode
  - Use a structured format to support both and more

Nested Boot Environments

- The stretch goal of the bectl(8) project was to add support for ‘nested’ BEs (bdrewery style)
- Example: have a /usr/src that matches the running kernel/world in each boot environment
- Would like better support for recursive cloning (ZFS Channel Program would be great for this)
- What filesystems would you like separate from the root FS, but part of the boot environment?

What Are Your Questions?

BSDNow.tv

- Weekly video podcast about the latest news in the BSD and IllumOS world
- Always looking for developers to interview
- Our archives are full of goodies (100+ Interviews):
  - Matt Ahrens
  - George Wilson
  - Bryan Cantrill
  - Adam Leventhal
  - Richard Yao
  - Alex Reese
  - Kirk McKusick
  - Josh Paetzel
  - Justin Gibbs
  - Paweł Jakub Dawidek
  - Sean Chittenden
  - Ryan Zezeski