The Release Engineering of FreeBSD 4.4

Murray Stokely murray@FreeBSD.org

Wind River Systems

Abstract

This paper describes the approach used by the FreeBSD release engineering team to make production-quality releases of the FreeBSD operating system. It details the methodology used for the release of FreeBSD 4.4 and describes the tools available for those interested in producing customized FreeBSD releases for corporate rollouts or commercial productization.

1 Introduction

The development of FreeBSD is a very open process. FreeBSD is comprised of contributions from thousands of people around the world. The FreeBSD Project provides anonymous CVS[1] access to the general public so that others can have access to log messages, diffs between development branches, and other productivity enhancements that formal source code management provides. This has been a huge help in attracting more talented developers to FreeBSD. However, I think everyone would agree that chaos would soon manifest if write access were opened up to everyone on the Internet. Therefore, only a "select" group of nearly 300 people are given write access to the CVS repository. These *committers*[6] are responsible for the bulk of FreeBSD development. An elected *core-team*[7] of very senior developers provides some level of direction over the project.

The rapid pace of FreeBSD development leaves little time for polishing the development system into a production quality release. To solve this dilemma, development continues on two parallel tracks. The main development branch is the HEAD or *trunk* of our CVS tree, known as *FreeBSD-CURRENT*. A more stable branch is maintained, known as *FreeBSD-STABLE*. Both branches live in a master CVS repository in California and are replicated via CVSup[2] to mirrors all over the world. FreeBSD-CURRENT[8] is the "bleeding-edge" of FreeBSD development where all new changes first enter the system. FreeBSD-STABLE is the development branch from which major releases are made. Changes go into this branch at a different pace, and with the general assumption that they have first gone into FreeBSD-CURRENT and have been thoroughly tested by our user community.

In the interim period between releases, nightly snapshots are built automatically by the FreeBSD Project build machines and made available for download from `ftp://stable.FreeBSD.org`. The widespread availability of binary release snapshots, and the tendency of our user community to keep up with -STABLE development with CVSup and "make world"[8] helps to keep FreeBSD-STABLE in a very reliable condition even before the quality assurance activities ramp up pending a major release.

Bug reports and feature requests are continuously submitted by users throughout the release cycle. Problem reports are entered into our GNATS[9] database through email, the send-pr(1) application, or via a web-based form. In addition to the multitude of different technical mailing lists about FreeBSD, the FreeBSD quality-assurance mailing list `freebsd-qa@FreeBSD.org` provides a forum for discussing the finer points of release-polishing.

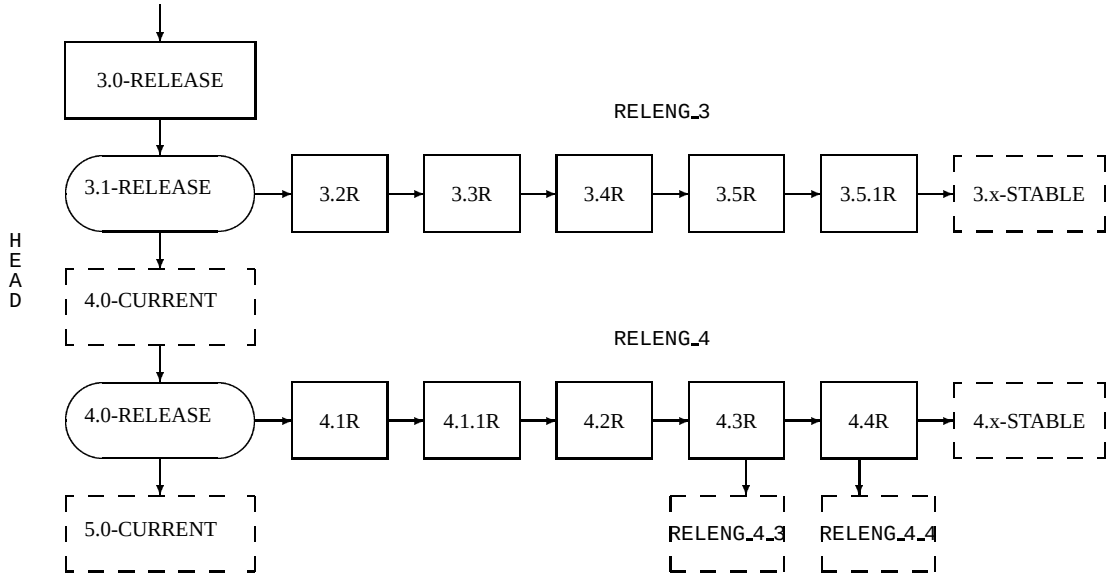
To service our most conservative users, individual release branches were introduced with FreeBSD 4.3. These release branches are created shortly before a final release is made, and after the release goes out only the most critical security fixes are merged onto the release branch. In addition to source updates via CVS, binary patchkits are available to keep systems on the `RELENG_4_3` and `RELENG_4_4` branches updated.

Section 2 discussed the different phases of the release engineering process leading up to the actual system build and section 3 describes the actual build process. Section 4 describes how the base releases may be extended by third-parties and Section 5 details some of the lessons learned through the release of FreeBSD 4.4. Finally, section 6 presents future directions of development.

2 Release Process

New releases of FreeBSD are released from the -STABLE branch at approximately four month intervals. The

Figure 1: FreeBSD Development Branches



FreeBSD release process begins to ramp up 45 days before the anticipated release date when the release engineer sends an email to the development mailing lists to remind developers that they only have 15 days to integrate new changes before the code freeze. During this time, many developers perform what have become known as "MFC sweeps". MFC stands for "Merge From CURRENT" and it describes the process of merging a tested change from our -CURRENT development branch to our -STABLE branch.

2.1 Code Review

Thirty days before the anticipated release, the source repository enters a "code slush". During this time, all commits to the -STABLE branch must be approved by the release engineer (`re@FreeBSD.org`). The kinds of changes that are allowed during this 15 day period include :

- Bug-fixes.
- Documentation updates.
- Security-related fixes of any kind.
- *Minor* changes to device drivers, such as adding new device IDs.
- Any additional change that the release engineering team feels is justified given the potential risk.

After the first 15 days of the code slush, a release candidate is released for widespread testing and the code enters a "code freeze" where it becomes much harder to justify new changes to the system unless a serious bug-fix or security issue is involved. During the code freeze, at least one release candidate is released per week until the final release is ready. During the days leading up to the final release, the release engineering team is in constant communication with the security-officer team, the documentation maintainers, and the ports managers, to make sure that all of the different components required for a successful release are available.

2.2 Final Release Checklist

When several release candidates have been made available for widespread testing and all major issues have been resolved, the final release "polishing" can begin.

Creating the Release Branch

As described in the introduction, the `RELENG_X_Y` release branch is a relatively new addition to our release engineering methodology. The first step in creating this branch is to ensure that you are working with the newest version of the `RELENG_X` sources that you want to branch *from*.

```
/usr/src# cvs up -rRELENG_4 -P -d
```

The next step is to create a branch point tag[1], so that diffs against the start of the branch are easier with CVS :

```
/usr/src# cvs rtag -rRELENG_4 RELENG_4_4_BP src
```

[1] A "tag" is CVS vernacular for a label that identifies the source at a specific point in time. By tagging the tree, we ensure that future release builders will always be able to use the same source we used to create the official FreeBSD Project releases.

And then a new branch tag is created with :

```
/usr/src# cvs rtag -b -rRELENG_4_4_BP \
    RELENG_4_4 src
```

The RELENG_* tags are restricted for use by the CVS-meisters and release engineers.

Bumping up the Version Number

Before the final release can be tagged, built, and released, the following files need to be modified to reflect the correct version of FreeBSD :

- `src/sys/conf/newvers.sh`
- `src/sys/sys/param.h`
- `src/release/doc/share/sgml/release.ent`
- `src/gnu/usr.bin/groff/tmac/mdoc.local`
- `doc/share/sgml/freebsd.ent`
- `doc/en_US.ISO8859-1/books/handbook/mirrors/chapter.sgml`
- `www/en/releases/*`
- `src/UPDATING`

Creating the Release Tags

When the final release is ready, the following command will create the `RELENG_4_4_0_RELEASE` tag.

```
/usr/src# cvs rtag -rRELENG_4_4 \
    RELENG_4_4_0_RELEASE src
```

The Documentation and Ports managers are responsible for tagging the respective trees with the `RELEASE_4_4_0` tag.

Occasionally, a last minute fix may be required *after* the final tags have been created. In practice this isn't a problem, since CVS allows tags to be manipulated with `cvs tag -d` *`tagname`* `filename`. It is very important that any last minute changes be tagged appropriately as part of the release. FreeBSD releases must always be reproduceable. Local hacks in the release engineer's environment are not acceptable.

3 Release Building

FreeBSD releases can be built by anyone with a fast machine and access to a source repository[2]. The only special requirement is that the `vn`[3] device must be available. If the device is not loaded into your kernel, then the kernel module should be automatically loaded when `vnconfig` is executed during the boot media creation phase. All of the tools necessary to build a release are available from the CVS repository in `src/release`. These tools aim to provide a consistent way to build FreeBSD releases. A complete release can actually be built with only a single command, including the creation of ISO images suitable for burning to CDROM, installation floppies, and an FTP install directory. This command is aptly named "make release".

3.1 "make release"

To successfully build a release, you must first populate `/usr/obj` by running "make world" or simply "make buildworld". The release target requires several variables be set properly to build a release :

- `CHROOTDIR` - The directory to be used as the chroot environment for the entire release build.
- `BUILDNAME` - The name of the release to be built.
- `CVSROOT` - The location of a CVS repository.
- `RELEASETAG` - The CVS tag corresponding to the release you would like to build.

There are many other variables available to customize the release build. Most of these variables are documented at the top of `src/release/Makefile`. The exact command used to build the official FreeBSD 4.4 (x86) release was :

```
make release CHROOTDIR=/local3/release \
  BUILDNAME=4.4-RELEASE \
  CVSROOT=/host/cvs/usr/home/ncvs \
  RELEASETAG=RELENG_4_4_0_RELEASE
```

The release Makefile can be broken down into several distinct steps.

- Creation of a sanitized system environment in a separate directory hierarchy with "make installworld".
- Checkout from CVS of a clean version of the system source, documentation, and ports into the release build hierarchy.

[2] That should be everyone, since we offer anonymous CVS! See `http://www.FreeBSD.org/handbook` for details.

[3] On -CURRENT, this device has been replaced by the new `md` memory disk driver.

- Population of `/etc` and `/dev` in the chrooted environment.
- chroot into the release build hierarchy, to make it harder for the outside environment to taint this build.
- "make world" in the chrooted environment.
- Build of Kerberos-related binaries.
- Build 'GENERIC' kernel.
- Creation of a staging directory tree where the binary distributions will be built and packaged.
- Build and installation of the documentation toolchain needed to convert the documentation source (SGML) into HTML, and text documents that will accompany the release.
- Build and installation of the actual documentation (user manuals, tutorials, release notes, hardware compatibility lists, etc...)
- Build of the "crunched" binaries used for installation floppies.
- Package up distribution tarballs of the binaries and sources.
- Create the boot media and a fixit floppy.
- Create FTP installation hierarchy.
- (optionally) Create ISO images for CDROM/DVD media.

3.2 Contributed Software ("ports")

The FreeBSD Ports collection[4] is a collection of nearly 6,000 third-party software packages available for FreeBSD. The ports team (`portmgr@FreeBSD.org`) is responsible for maintaining a consistent ports tree that can be used to create the binary packages that accompany a given FreeBSD release.

The Ports Cluster

In order to provide a consistent set of third-party packages for FreeBSD releases, every port is built in a separate chroot environment, starting with an empty `/usr/local` and `/usr/X11R6`. The requisite dependencies are installed as packages before the build proceeds. This enforces *consistency* in the package build process. By starting the package build in a pristine environment, we can assure that the package metadata (such as required dependencies) is accurate, and so we will never generate packages that might work on some systems and not on others depending on what software was previously installed.

The "Ports Cluster"[3] for the x86 architecture currently consists of a master node (Dual Pentium III 733Mhz) and 8 slave nodes (Pentium III 800Mhz) to do the actual package builds. With this configuration, a complete package build takes over 24 hours. These machines are co-located with the other FreeBSD Project equipment at Yahoo's corner of Exodus in Santa Clara, CA.

The "Ports Cluster" for the Alpha architecture consists of 7 PWS 500A machines donated by Compaq and co-located in the BSD Lab at Wind River Systems.

The Package Split

For FreeBSD 4.4 over 4.1 gigabytes of packages were created. This causes a problem for CDROM distributions because we would like to ship as many packages as possible without making the user insert another disc to satisfy dependencies. The solution is to create "clusters" of like packages with similar dependencies onto specific discs. The package split is performed by the `portmgr@FreeBSD.org` team in coordination with the wishes of the general user community with respect to which packages get to appear on the first CD.

3.3 Release ISOs

Starting with FreeBSD 4.4, the FreeBSD Project decided to release all four ISO images that were previously sold on the BSDi/Wind River Systems "official" CDROM distributions. Each of the four discs must contain a `README.TXT` file that explains the contents of the disc, a `CDROM.INF` file that provides meta-data for the disc so that sysinstall can validate and use the contents, and a `filename.txt` file that provides a manifest for the disc. This manifest can be created with a simple command :

```
/stage/cdrom#  find . -type f |
  sed -e 's/\^.\///' | sort > filename.txt
```

The specific requirements of each CD is outlined below.

Disc #1

The first disc is almost completely created by "make release". The only changes that should be made to the `disc1` directory are the addition of a 'tools' directory, XFree86, and as many popular third party software packages as will fit on the disc. The 'tools' directory contains software that allow users to create installation floppies from other operating systems. This disc should be made bootable so that users of modern PCs do not need to create installation floppy disks.

If an alternate version of XFree86 is to be provided, then sysinstall must be updated to reflect the new location and installation instructions. The relevant code is contained in `src/release/sysinstall` on -STABLE or

`src/usr.sbin/sysinstall` on -CURRENT. Specifically, the files `dist.c`, `menus.c`, and `config.c` will need to be updated.

Disc #2

The second disc is also largely created by "make release". This disc contains a "live filesystem" that can be used from sysinstall to troubleshoot a FreeBSD installation. This disc should be bootable and should also contain a compressed copy of the CVS repository in the `CVSROOT` directory and commercial software demos in the `commerce` directory.

Discs #3 and 4

The remaining two discs contains additional software packages for FreeBSD. The packages should be clustered so that a package and all of its dependencies are included on the same disc.

4 Extensibility

Although FreeBSD forms a complete operating system, there is nothing that forces you to use the system exactly as we've packaged it up for distribution. We have tried to design the system to be as extensible as possible so that it can serve as a platform that other commercial products can be built on top of. The only "rule" we have about this is that if you're going to distribute FreeBSD with non-trivial changes, we encourage you to document your enhancements! The FreeBSD community can only help support users of the software we provide. We certainly encourage innovation in the form of advanced installation and administration tools, for example, but we can't be expected to answer questions about it.

4.1 Creating Customized Boot floppies

Many sites have complex requirements that may require additional kernel modules or userland tools be added to the installation floppies. The "quick and dirty" way to accomplish this would be to modify the staging directory of an existing "make release" build hierarchy :

- Apply patches or add additional files inside the chroot release build directory.
- `rm ${CHROOTDIR}/usr/obj/usr/src/\`
  `release/release.[48]`
- rebuild sysinstall, the kernel, or whatever parts of the system your change affected.
- `chroot ${CHROOTDIR} ./mk release.4`
- `chroot ${CHROOTDIR} ./mk release.8`

New release floppies will then be located in ${CHROOTDIR}/R/stage/floppies.

Alternatively, the "boot.flp" make target can be called or the filesystem creation script, `src/release/scripts/doFS.sh` may be invoked directly.

Local patches may also be supplied to a release build by defining the LOCAL_PATCH variable in "make release".

4.2 Scripting Sysinstall

The FreeBSD system installation and configuration tool, sysinstall, can be scripted to provide automated installs for large sites. This functionality can be used in conjunction with Intel's PXE[13] to bootstrap systems from the network, or via custom boot floppies with a sysinstall script. An example sysinstall script is available in the CVS tree as `src/release/sysinstall/install.cfg`.

5 Lessons Learned from FreeBSD 4.4

The release engineering process for 4.4 formally began on August 1st, 2001. After that date all commits to the RELENG_4 branch of FreeBSD had to be explicitly approved by `re@FreeBSD.org`. The first release candidate for the x86 architecture was release on August 16, followed by 4 more release candidates leading up to the final release on September 18th. The security officer was very involved in the last week of the process as several security issues were found in the earlier release candidates. A total of over **500** emails were sent to `re@FreeBSD.org` in little over a month.

Our user community has made it very clear that the security and stability of a FreeBSD release should not be sacrificed for any self-imposed deadlines or target release dates.

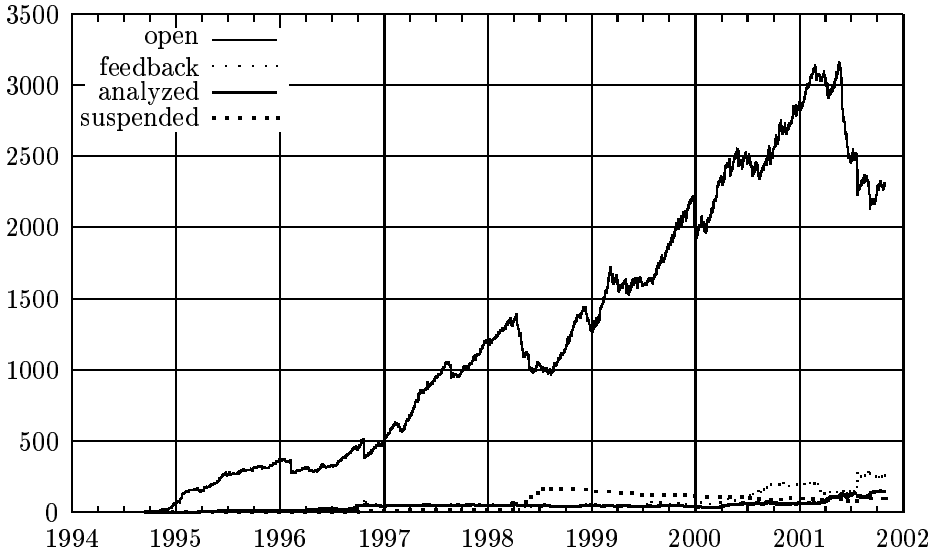
The FreeBSD Project has grown tremendously over its lifetime and the need for standardized release engineering procedures has never been more apparent. This will become even more important as FreeBSD is ported to new platforms.

6 Future Directions

It is imperative for our release engineering activities to scale with our growing userbase. Along these lines we are working very hard to document the procedures involved in producing FreeBSD releases.

- *Parallelism* - Certain portions of the release build are actually "embarrassingly parallel". Most of the tasks are very I/O intensive, so multiple high-speed disk drives is actually more important than multiple processors in speeding up the "make release" process. If multiple disks are used for different hierarchies in the chroot environment, then the CVS checkout of the

Figure 2: Project Lifetime PR Statistics [10]



ports and doc trees can be happening simultaneously to the "make world" on another disk. Using a RAID solution (hardware or software) can significantly decrease the overall build time.

- *Cross-building releases* - Building IA-64 or Alpha releases on x86 hardware? "make TARGET=ia64 release"

- *Regression Testing* - We need better automated correctness testing for FreeBSD.

- *Installation Tools* - Our installation program has long since outlived its intended life span. Several projects are under development to provide a more advanced installation mechanism. One of the most promising is the libh project[5] which aims to provide an intelligent new package framework and GUI installation program.

Acknowledgments

I would like to thank Jordan Hubbard for giving me the opportunity to take on some of the release engineering responsibilities for FreeBSD 4.4 and also for all of his work throughout the years making FreeBSD what it is today. Of course the release wouldn't have been possible without all of the release-related work done by Satoshi Asami, Steve Price, Bruce Mah, Nik Clayton, David O'Brien, Kris Kennaway, John Baldwin, and the rest of the FreeBSD developer community. I would also like to thank Rod Grimes, Poul-Henning Kamp, and others who worked on the release engineering tools in the very early days of FreeBSD.

This paper was influenced by release engineering documents from the CSRG[14], the NetBSD Project[11], and John Baldwin's proposed release engineering process notes[12].

References

[1] CVS – Concurrent Versions System `http://www.cvshome.org`

[2] CVSup – The CVS-Optimized General Purpose Network File Distribution System `http://www.polstra.com/projects/freeware/CVSup/`

[3] `http://bento.FreeBSD.org`

[4] FreeBSD Ports Collection `http://www.FreeBSD.org/ports/`

[5] The libh Project `http://www.FreeBSD.org/projects/libh.html`

[6] FreeBSD Commiters `http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/contributors/staff-committers.html`

[7] FreeBSD Core-Team `http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/contributors/staff-core.html'`

[8] FreeBSD Handbook http://www.FreeBSD.org/handbook

[9] GNATS : The GNU Bug Tracking System http://sources.redhat.com/gnats/

[10] Poul-Henning's FreeBSD PR Statistics http://phk.freebsd.dk/Gnats/

[11] NetBSD Developer Documentation: Release Engineering http://www.netbsd.org/developers/releng/index.html

[12] John Balwin's FreeBSD Release Engineering Proposal http://people.freebsd.org/~jhb/docs/releng.txt

[13] PXE Jumpstart Guide http://www.freebsd.org/doc/en_US.ISO8859-1/articles/pxe/index.html

[14] Marshall Kirk McKusick, Michael J. Karels, and Keith Bostic *The Release Engineering of 4.3BSD*