FreeBSD Release Engineering

Abstract

This article describes the release engineering process of the FreeBSD Project.

Table of Contents

1. Introduction to the FreeBSD Release Engineering Process . . . 1
2. General Information and Preparation . . . 2
3. Release Engineering Terminology . . . 4
4. Website Changes During the Release Cycle . . . 5
5. Release from main . . . 6
6. Release from stable/ . . . 8
7. Building FreeBSD Installation Media . . . 10
8. Publishing FreeBSD Installation Media to Project Mirrors . . . 11
9. Wrapping up the Release Cycle . . . 12
10. Release End-of-Life . . . 13

1. Introduction to the FreeBSD Release Engineering Process

Development of FreeBSD has a very specific workflow. In general, all changes to the FreeBSD base system are committed to the main branch, which reflects the top of the source tree.

After a reasonable testing period, changes can then be merged to the stable/ branches. The default minimum timeframe before merging to stable/ branches is three (3) days.

Although a general rule to wait a minimum of three days before merging from main, there are a few special circumstances where an immediate merge may be necessary, such as a critical security fix, or a bug fix that directly inhibits the release build process.

After several months, and the number of changes in the stable/ branch have grown significantly, it is time to release the next version of FreeBSD. These releases have been historically referred to as "point" releases.

In between releases from the stable/ branches, approximately every two (2) years, a release will be cut directly from main. These releases have been historically referred to as "dot-zero" releases.

This article will highlight the workflow and responsibilities of the FreeBSD Release Engineering Team for both "dot-zero" and "point"' releases.

The following sections of this article describe:

General Information and Preparation

General information and preparation before starting the release cycle.

Website Changes During the Release Cycle

Website Changes During the Release Cycle

Release Engineering Terminology

Terminology and general information, such as the "code slush" and "code freeze", used throughout this document.

Release from main

The Release Engineering process for a "dot-zero" release.

Release from stable/

The Release Engineering process for a "point" release.

Building FreeBSD Installation Media

Information related to the specific procedures to build installation medium.

Publishing FreeBSD Installation Media to Project Mirrors

Procedures to publish installation medium.

Wrapping up the Release Cycle

Wrapping up the release cycle.

2. General Information and Preparation

Approximately two months before the start of the release cycle, the FreeBSD Release Engineering Team decides on a schedule for the release. The schedule includes the various milestone points of the release cycle, such as freeze dates, branch dates, and build dates. For example:

Milestone	Anticipated Date
main slush:	May 27, 2016
main freeze:	June 10, 2016
main KBI freeze:	June 24, 2016
`doc/` tree slush [1]:	June 24, 2016
Ports quarterly branch [2]:	July 1, 2016
stable/13 branch:	July 8, 2016
`doc/` tree tag [3]:	July 8, 2016
BETA1 build starts:	July 8, 2016
main thaw:	July 9, 2016

Milestone	Anticipated Date
BETA2 build starts:	July 15, 2016
BETA3 build starts [*]:	July 22, 2016
releng/13.0 branch:	July 29, 2016
RC1 build starts:	July 29, 2016
stable/13 thaw:	July 30, 2016
RC2 build starts:	August 5, 2016
Final Ports package builds [4]:	August 6, 2016
Ports release tag:	August 12, 2016
RC3 build starts [*]:	August 12, 2016
RELEASE build starts:	August 19, 2016
RELEASE announcement:	September 2, 2016



Items marked with "[*]" are "as needed".

1. The `doc/` tree slush is coordinated by the FreeBSD Documentation Engineering Team.
2. The Ports quarterly branch used is determined by when the final `RC` build is planned. A new quarterly branch is created on the first day of the quarter, so this metric should be used when taking the release cycle milestones into account. The quarterly branch is created by the FreeBSD Ports Management Team.
3. The `doc/` tree is tagged by the FreeBSD Documentation Engineering Team.
4. The final Ports package build is done by the FreeBSD Ports Management Team after the final (or what is expected to be final) `RC` build.



If the release is being created from an existing stable/ branch, the KBI freeze date can be excluded, since the KBI is already considered frozen on established stable/ branches.

When writing the release cycle schedule, a number of things need to be taken into consideration, in particular milestones where the target date depends on predefined milestones upon which there is a dependency. For example, the Ports Collection release tag originates from the active quarterly branch at the time of the last `RC`. This in part defines which quarterly branch is used, when the release tag can happen, and what revision of the ports tree is used for the final `RELEASE` build.

After general agreement on the schedule, the FreeBSD Release Engineering Team emails the schedule to the FreeBSD Developers.

It is somewhat typical that many developers will inform the FreeBSD Release Engineering Team about various works-in-progress. In some cases, an extension for the in-progress work will be requested, and in other cases, a request for "blanket approval" to a particular subset of the tree will be made.

When such requests are made, it is important to make sure timelines (even if estimated) are

discussed. For blanket approvals, the length of time for the blanket approval should be made clear. For example, a FreeBSD developer may request blanket approvals from the start of the code slush until the start of the RC builds.



To keep track of blanket approvals, the FreeBSD Release Engineering Team uses an internal repository to keep a running log of such requests, which defines the area upon which a blanket approval was granted, the author(s), when the blanket approval expires, and the reason the approval was granted. One example of this is granting blanket approval to release/doc/ to all FreeBSD Release Engineering Team members until the final RC to update the release notes and other release-related documentation.



The FreeBSD Release Engineering Team also uses this repository to track pending approval requests that are received just prior to starting various builds during the release cycle, which the Release Engineer specifies the cutoff period with an email to the FreeBSD developers.

Depending on the underlying set of code in question, and the overall impact the set of code has on FreeBSD as a whole, such requests may be approved or denied by the FreeBSD Release Engineering Team.

The same applies to work-in-progress extensions. For example, in-progress work for a new device driver that is otherwise isolated from the rest of the tree may be granted an extension. A new scheduler, however, may not be feasible, especially if such dramatic changes do not exist in another branch.

The schedule is also added to the Project website, in the doc/ repository, in ~/website/content/en/releases/13.0R/schedule.adoc. This file is continuously updated as the release cycle progresses.



In most cases, the schedule.adoc can be copied from a prior release and updated accordingly.

In addition to adding schedule.adoc to the website, ~/shared/releases.adoc is also updated to add the link to the schedule to various subpages, as well as enabling the link to the schedule on the Project website index page.

The schedule is also linked from ~/website/content/en/releng/_index.adoc.

Approximately one month prior to the scheduled "code slush", the FreeBSD Release Engineering Team sends a reminder email to the FreeBSD Developers.

3. Release Engineering Terminology

This section describes some of the terminology used throughout the rest of this document.

3.1. The Code Slush

Although the code slush is not a hard freeze on the tree, the FreeBSD Release Engineering Team requests that bugs in the existing code base take priority over new features.

The code slush does not enforce commit approvals to the branch.

3.2. The Code Freeze

The code freeze marks the point in time where all commits to the branch require explicit approval from the FreeBSD Release Engineering Team.

The FreeBSD Git repository contains several hooks to perform sanity checks before any commit is actually committed to the tree. One of these hooks will evaluate if committing to a particular branch requires specific approval.

To enforce commit approvals by the FreeBSD Release Engineering Team, the Release Engineering Team must approve any changes to the branch, in which case the commit log must include an `Approved by: re (login)` line, where "login" is the login ID of the approver.



During the code freeze, FreeBSD committers are urged to follow the Change Request Guidelines.

3.3. The KBI/KPI Freeze

KBI/KPI stability implies that the caller of a function across two different releases of software that implement the function results in the same end state. The caller, whether it is a process, thread, or function, expects the function to operate in a certain way, otherwise the KBI/KPI stability on the branch is broken.

4. Website Changes During the Release Cycle

This section describes the changes to the website that should occur as the release cycle progresses.



The files specified throughout this section are relative to the `main` branch of the `doc` repository.

4.1. Website Changes Before the Release Cycle Begins

When the release cycle schedule is available, these files need to be updated to enable different functionalities on the FreeBSD Project website:

File to Edit	What to Change
~/shared/releases.adoc	Change `beta-upcoming` from `IGNORE` to `INCLUDE`
~/shared/releases.adoc	Change `beta-testing` from `IGNORE` to `INCLUDE`

4.2. Website Changes During BETA or RC

When transitioning from PRERELEASE to BETA, these files need to be updated to enable the "Help Test" block on the download page. All files are relative to head/ in the doc repository:

File to Edit	What to Change
~/shared/releases.adoc	Update betarel-vers to BETA1
~/website/data/en/news/news.toml	Add an entry announcing the BETA
~/website/static/security/advisory-template.txt	Add the new BETA, RC, or final RELEASE to the template
~/website/static/security/errata-template.txt	Add the new BETA, RC, or final RELEASE to the template

Once the releng/13.0 branch is created, the various release-related documents need to be added to the doc/ repository.



The relevant release-related documents exist in the doc repository for FreeBSD 12.x and later.

4.3. Ports Changes During BETA, RC, and the Final RELEASE

For each build during the release cycle, the MANIFEST files containing the SHA256 of the various distribution sets, such as base.txz, kernel.txz, and so on, are added to the misc/freebsd-release-manifests port. This allows utilities other than bsdinstall(8), such as ports-mgmt/poudriere, to safely use these distribution sets by providing a mechanism through which the checksums can be verified.

5. Release from main

This section describes the general procedures of the FreeBSD release cycle from the main branch.

5.1. FreeBSD “ALPHA” Builds

Starting with the FreeBSD 10.0-RELEASE cycle, the notion of “ALPHA” builds was introduced. Unlike the BETA and RC builds, ALPHA builds are not included in the FreeBSD Release schedule.

The idea behind ALPHA builds is to provide regular FreeBSD-provided builds before the creation of the stable/ branch.

FreeBSD ALPHA snapshots should be built approximately once a week.

For the first ALPHA build, the BRANCH value in sys/conf/newvers.sh needs to be changed from CURRENT to ALPHA1. For subsequent ALPHA builds, increment each ALPHAN value by one.

See Building FreeBSD Installation Media for information on building the ALPHA images.

5.2. Creating the stable/13 Branch

When creating the stable/ branch, several changes are required in both the new stable/ branch and the main branch. The files listed are relative to the repository root. To create the new stable/13 branch in Git:



Make sure that you are in the main branch

```
% git checkout -b stable/13
```

Once the stable/13 branch has been created, make the following edits:

File to Edit	What to Change
UPDATING	Update the FreeBSD version, and remove the notice about `WITNESS`

contrib/jemalloc/include/jemalloc/jemalloc_FreeBSD.h	`#ifndef MALLOC_PRODUCTION`
	`#define MALLOC_PRODUCTION`
	`#endif`

lib/clang/llvm.build.mk	Uncomment `-DNDEBUG`
sys/*/conf/GENERIC*	Remove debugging support
sys/*/conf/MINIMAL	Remove debugging support
release/release.conf.sample	Update `SRCBRANCH`
sys/*/conf/GENERIC-NODEBUG	Remove these kernel configurations
sys/arm/conf/std.arm*	Remove debugging options
sys/conf/newvers.sh	Update the `BRANCH` value to reflect `BETA1`
share/mk/src.opts.mk	Move `REPRODUCIBLE_BUILD` from `__DEFAULT_NO_OPTIONS` to `__DEFAULT_YES_OPTIONS`
share/mk/src.opts.mk	Move `LLVM_ASSERTIONS` from `__DEFAULT_YES_OPTIONS` to `__DEFAULT_NO_OPTIONS`
libexec/rc/rc.conf	Set `dumpdev` from `AUTO` to `NO` (it is configurable via for those that want it enabled by default)
release/Makefile	Remove the `debug.witness.trace` entries

Then in the main branch, which will now become a new major version:

File to Edit	What to Change
UPDATING	Update the FreeBSD version
sys/conf/newvers.sh	Update the `BRANCH` value to reflect `CURRENT`, and increment `REVISION`

File to Edit	What to Change
Makefile.inc1	Update `TARGET_TRIPLE` and `MACHINE_TRIPLE`
sys/sys/param.h	Update `__FreeBSD_version`
gnu/usr.bin/cc/cc_tools/freebsd-native.h	Update `FBSD_MAJOR` and `FBSD_CC_VER`
contrib/gcc/config.gcc	Append the `freebsdversion.h` section
lib/clang/llvm.build.mk	Update the value of `OS_VERSION`
lib/clang/freebsd_cc_version.h	Update `FREEBSD_CC_VERSION`
lib/clang/include/lld/Common/Version.inc	Update `LLD_REVISION_STRING`
Makefile.libcompat	Update `LIB32CPUFLAGS`

6. Release from stable/

This section describes the general procedures of the FreeBSD release cycle from an established stable/ branch.

6.1. FreeBSD `stable` Branch Code Slush

In preparation for the code freeze on a `stable` branch, several files need to be updated to reflect the release cycle is officially in progress. These files are all relative to the top-most level of the stable branch:

File to Edit	What to Change
sys/conf/newvers.sh	Update the `BRANCH` value to reflect `PRERELEASE`
Makefile.inc1	Update `TARGET_TRIPLE`
lib/clang/llvm.build.mk	Update `OS_VERSION`
Makefile.libcompat	Update `LIB32CPUFLAGS`

6.2. FreeBSD `BETA` Builds

Following the code slush, the next phase of the release cycle is the code freeze. This is the point at which all commits to the stable branch require explicit approval from the FreeBSD Release Engineering Team. This is enforced by gitadm@FreeBSD.org who handles the repository.



There are two general exceptions to requiring commit approval during the release cycle. The first is any change that needs to be committed by the Release Engineer to proceed with the day-to-day workflow of the release cycle, the other is security fixes that may occur during the release cycle.

Once the code freeze is in effect, the next build from the branch is labeled `BETA1`. This is done by updating the `BRANCH` value in sys/conf/newvers.sh from `PRERELEASE` to `BETA1`.

Once this is done, the first set of BETA builds are started. Subsequent BETA builds do not require updates to any files other than sys/conf/newvers.sh, incrementing the BETA build number.

6.3. Creating the releng/13.0 Branch

When the first RC (Release Candidate) build is ready to begin, the releng/ branch is created. This is a multi-step process that must be done in a specific order, to avoid anomalies such as overlaps with __FreeBSD_version values, for example. The paths listed below are relative to the repository root. The order of commits and what to change are:



Make sure that you are in the stable/13 branch

```
% git checkout -b releng/13.0
```

File to Edit	What to Change
sys/conf/newvers.sh	Change BETAX to RC1
sys/sys/param.h	Update __FreeBSD_version
sys/conf/kern.opts.mk	Move REPRODUCIBLE_BUILD from DEFAULT_NO_OPTIONS toDEFAULT_YES_OPTIONS
etc/pkg/FreeBSD.conf	Replace latest with quarterly as the default package repository location
release/pkg_repos/release-dvd.conf	Replace latest with quarterly as the default package repository location
sys/conf/newvers.sh	Update BETAX with PRERELEASE
sys/sys/param.h	Update __FreeBSD_version

Then, gitadm@FreeBSD.org adds new approvers for the releng branch as did for the stable branch.

```
% git add .
% git commit
```

Now that two new __FreeBSD_version values exist, also update ~/documentation/content/en/books/porters-handbook/versions/chapter.adoc in the Documentation Project repository.

After the first RC build has completed and tested, the stable/ branch can be "thawed" by gitadm@FreeBSD.org.

Following the availability of the first RC, FreeBSD Bugmeister Team should be emailed to add the new FreeBSD -RELEASE to the versions available in the drop-down menu shown in the bug tracker.

7. Building FreeBSD Installation Media

This section describes the general procedures producing FreeBSD development snapshots and releases.

7.1. Release Build Scripts

Prior to FreeBSD 9.0-RELEASE, src/release/Makefile was updated to support bsdinstall(8), and the src/release/generate-release.sh script was introduced as a wrapper to automate invoking the release(7) targets.

Prior to FreeBSD 9.2-RELEASE, src/release/release.sh was introduced, which heavily based on src/release/generate-release.sh included support to specify configuration files to override various options and environment variables. Support for configuration files provided support for cross building each architecture for a release by specifying a separate configuration file for each invocation.

As a brief example of using src/release/release.sh to build a single release in /scratch:

```
# /bin/sh /usr/src/release/release.sh
```

As a brief example of using src/release/release.sh to build a single, cross-built release using a different target directory, create a custom release.conf containing:

```
# release.sh configuration for powerpc/powerpc64
CHROOTDIR="/scratch-powerpc64"
TARGET="powerpc"
TARGET_ARCH="powerpc64"
KERNEL="GENERIC64"
```

Then invoke src/release/release.sh as:

```
# /bin/sh /usr/src/release/release.sh -c $HOME/release.conf
```

See release(7) and src/release/release.conf.sample for more details and example usage.

7.2. Building FreeBSD Releases

During the release cycle, a copy of CHECKSUM.SHA512 and CHECKSUM.SHA256 for each architecture are stored in the FreeBSD Release Engineering Team internal repository in addition to being included in the various announcement emails. Each MANIFEST containing the hashes of base.txz, kernel.txz, etc. are added to misc/freebsd-release-manifests in the Ports Collection, as well.

In preparation for the release build, several files need to be updated:

File to Edit	What to Change
sys/conf/newvers.sh	Update the `BRANCH` value to `RELEASE`
UPDATING	Add the anticipated announcement date
lib/csu/common/crtbrand.S	Replace `__FreeBSD_version` with the value in sys/sys/param.h

After building the final `RELEASE`, the releng/13.0 branch is tagged as release/13.0.0 using the revision from which the `RELEASE` was built. Similar to creating the stable/13 and releng/13.0 branches, this is done with `git tag`. From the repository root:



Make sure that you are in the releng/13.0 branch

```
% git tag release/13.0.0
```

8. Publishing FreeBSD Installation Media to Project Mirrors

This section describes the procedure to publish FreeBSD development snapshots and releases to the Project mirrors.

8.1. Staging FreeBSD Installation Media Images

Staging FreeBSD snapshots and releases is a two part process:

- Creating the directory structure to match the hierarchy on `ftp-master`

  If `EVERYTHINGISFINE` is defined in the build configuration files, main.conf in the case of the build scripts referenced above, this happens automatically in the chroot(8) after the build is complete, creating the directory structure in ${DESTDIR}/R/ftp-stage with a path structure matching what is expected on `ftp-master`. This is equivalent to running the following in the chroot(8) directly:

  ```
  # make -C /usr/src/release -f Makefile.mirrors EVERYTHINGISFINE=1 ftp-stage
  ```

  After each architecture is built, thermite.sh will rsync the ${DESTDIR}/R/ftp-stage from the build to /snap/ftp/snapshots or /snap/ftp/releases on the build host, respectively.

- Copying the files to a staging directory on `ftp-master` before moving the files into pub/ to begin propagation to the Project mirrors

  Once all builds have finished, /snap/ftp/snapshots, or /snap/ftp/releases for a release, is polled by `ftp-master` using rsync to /archive/tmp/snapshots or /archive/tmp/releases, respectively.

  

  On `ftp-master` in the FreeBSD Project infrastructure, this step requires `root`

level access, as this step must be executed as the `archive` user.

8.2. Publishing FreeBSD Installation Media

Once the images are staged in /archive/tmp/, they are ready to be made public by putting them in /archive/pub/FreeBSD. To reduce propagation time, pax(1) is used to create hard links from /archive/tmp to /archive/pub/FreeBSD.



For this to be effective, both /archive/tmp and /archive/pub must reside on the same logical filesystem.

There is a caveat, however, where rsync must be used after pax(1) to correct the symbolic links in pub/FreeBSD/snapshots/ISO-IMAGES which pax(1) will replace with a hard link, increasing the propagation time.



As with the staging steps, this requires `root` level access, as this step must be executed as the `archive` user.

As the `archive` user:

```
% cd /archive/tmp/snapshots
% pax -r -w -l . /archive/pub/FreeBSD/snapshots
% /usr/local/bin/rsync -avH /archive/tmp/snapshots/* /archive/pub/FreeBSD/snapshots/
```

Replace *snapshots* with *releases* as appropriate.

9. Wrapping up the Release Cycle

This section describes general post-release tasks.

9.1. Post-Release Errata Notices

As the release cycle approaches conclusion, it is common to have several EN (Errata Notice) candidates to address issues that were discovered late in the cycle. Following the release, the FreeBSD Release Engineering Team and the FreeBSD Security Team revisit changes that were not approved prior to the final release, and depending on the scope of the change in question, may issue an EN.



The actual process of issuing ENs is handled by the FreeBSD Security Team.

To request an Errata Notice after a release cycle has completed, a developer should fill out the Errata Notice template, in particular the `Background`, `Problem Description`, `Impact`, and if applicable, `Workaround` sections.

The completed Errata Notice template should be emailed together with either a patch against the releng/ branch or a list of revisions from the stable/ branch.

For Errata Notice requests immediately following the release, the request should be emailed to both the FreeBSD Release Engineering Team and the FreeBSD Security Team. Once the releng/ branch has been handed over to the FreeBSD Security Team as described in Handoff to the FreeBSD Security Team, Errata Notice requests should be sent to the FreeBSD Security Team.

9.2. Handoff to the FreeBSD Security Team

Roughly two weeks following the release, the Release Engineer updates the Git repository changing the approver from the Release engineering team to the security officer for the releng/13.0 branch.

10. Release End-of-Life

This section describes the website-related files to update when a release reaches EoL (End-of-Life).

10.1. Website Updates for End-of-Life

When a release reaches End-of-Life, references to that release should be removed and/or updated on the website:

File	What to Change
~/website/themes/beastie/layouts/index.html	Remove `u-relXXX-announce` and `u-relXXX-announce` references.
~/website/content/en/releases/_index.adoc	Move the `u-relXXX-*` variables from the supported release list to the Legacy Releases list.
~/website/content/en/releng/_index.adoc	Update the appropriate releng branch to reflect the branch is no longer supported.
~/website/content/en/security/_index.adoc	Remove the branch from the supported branch list.
~/website/content/en/security/unsupported.adoc	Add the branch to the unsupported branch list.
~/website/content/en/where.adoc	Remove the URLs for the release.
~/website/themes/beastie/layouts/partials/sidenav.html	Remove `u-relXXX-announce` and `u-relXXX-announce` references.
~/website/static/security/advisory-template.txt	Remove references to the release and releng branch.
~/website/static/security/errata-template.txt	Remove references to the release and releng branch.