FreeBSD Release Engineering: A New Sheriff is in Town

BY COLIN PERCIVAL

On November 17, 2023, Glen Barber retired from the position of FreeBSD Release Engineering Lead after a decade of managing FreeBSD releases, and with the endorsement of the FreeBSD Core Team, I took over the role.

Continuity, and

Glen did an excellent job as a release engineer, so when I took over my priority was to provide continuity: To do things, as much as possible, the same way as he had done previously. To this end I was aided by three years of experience as Release Engineering Deputy Lead: While I had only managed one release before (FreeBSD 13.2-RELEASE, after Glen was hospitalized with pneumonia), I had watched how Glen did things enough to have a general sense of how to continue.

But that is not to say that I made no changes—if that were the case, this would be a very short article. My first goal was to streamline the release process to avoid a repeat of FreeBSD 13.2, which arrived almost a month late after needing six release candidates before all the issues were ironed out enough to proceed to the release. To that end, I set out some ground rules for release cycles:

- A Release Candidate should be exactly that—a candidate for a release—and so by the time we get to RC1 there should be no further known release-blocking issues. The goal is to build RC1, have a week of testing (or a bit under a week given the time for builds to complete), and then build the final RELEASE images. If problems are found in RC1 we can do a second or third release candidate—but this should only happen with new problems; any previously-known problems should have already been ironed out before RC1.
- While developers often have code that they "need" to get into the next release, I'm not going to hold up the release process if they're late—it's simply not fair to all the other developers who got their code into the tree on time and want to get it into the hands of users. The tree is open for anyone to merge patches until -BETA1 (we used to "freeze" the stable branch, but that hasn't happened for many years now) and between BETA1 and BETA2 I'm generally willing to accept changes that aren't particularly large or dangerous, but between BETA2 and BETA3 it becomes a matter of "are we *sure* this patch won't cause any new problems" and between BETA3 and RC1 the bar for patches rises to "... and does it fix an actual problem which users are running into" since if a problem is purely theoretical it's not worth the risk that something might be wrong with the patch. After RC1, of course, only the most critical patches will be accepted—and ideally

no patches—since any changes beyond that point will require adding another Release Candidate and pushing back the release date.
- If new features arrive close to the release—say, after we enter "code slush", two weeks before BETA1—and problems are found that can't be immediately and trivially fixed, *I will remove those features from the release*. Part of the reason that past release schedules often dragged on is that features arrived late and then needed multiple rounds of bug fixes before the release could ship; as with late-arriving code, it's not fair to everyone else if one developer is delaying the release because they merged buggy code at the last minute.

I also asked FreeBSD developers—with inconsistent success, although it does seem to be gradually improving—to be far more proactive in alerting the release engineering team to anything they intended to get into the upcoming release; and once I knew about issues, I became more proactive in emailing developers to ask for status updates. More than once I had replies along the lines of "I didn't realize BETA2 was already out; I'll get that code merged ASAP"—FreeBSD is a volunteer project so it's entirely reasonable that other things in developers' lives distract them from working on FreeBSD, but the last thing we want to do is have a release schedule slip just because someone lost track of time.

Once I knew about issues, I became more proactive in emailing developers to ask for status updates.

Release Scheduling

Once it became clear that the FreeBSD project could do releases in a predictable amount of time, I turned my mind towards scheduling. Three BETA images and one RC collectively take a month, and we have a two-week "code slush" before BETA1 and a two-week warning (aka "hurry and get your code merged") before that. That adds up to 2 months if everything goes perfectly; but since the schedule will *sometimes* slip no matter how hard we try (if nothing else, because we'll always hold the release for last-minute security fixes) we need a third month to allow for slack in the schedule—and to allow the release engineering team a bit of time between releases.

Knowing that we can effectively do one release per 3 months makes a schedule obvious: Do one release per calendar quarter. If we start with the 2 week "warning" and 2-week "code slush" in the first month of the quarter and have BETAs and RC1 weekly through the second month of the quarter, we end up with the RELEASE announcement landing around the start of the third month. If the release slips by a few weeks, we still finish before the end of the third month. This gives us a target schedule for future releases—we're never going to ship a release that isn't ready, but knowing what we're aiming for is an essential starting point:
- BETA1 - 28 days: Warning to developers.
- BETA1 - 14 days: Code slush starts.
- First Friday in second month of the quarter: BETA1.
- BETA1 + 7 days: BETA2.
- BETA1 + 14 days: BETA3.
- BETA1 + 21 days: RC1.
- BETA1 + 28 days: RELEASE builds start.

- BETA1 + 32 days: RELEASE announcement goes out.
- BETA1 + 39 days: The release branch is handed over to the security team.

The one exception to this is .0 releases: These start with ALPHA builds before the stable branch is created, and there's simply no way to fit the entire process for a .0 release into three months; so, in the interest of keeping the schedule otherwise aligned with calendar quarters, I decided we should set aside six months—two calendar quarters—for those releases. The exact schedule of when things will happen within those six months, I haven't decided yet—and since I've never done a .0 release before, will probably get wrong for 15.0—but I do hope to eventually establish a repeatable schedule for these as well.

Once the amount of time for each release was determined, all that remained was to set the order in which releases would occur and decide how often to do .0 releases. The first question was easily answered: There's no point doing a release 3 months after a release from the same stable branch, so we alternate between stable branches, i.e. 14.0, 13.3, 14.1, 13.4, 14.2, et cetera. The answer to the second question came from many discussions amongst FreeBSD developers over the years: We'd like to have a new major version every 2 years. In the past, it has worked out very well to have the .0 release cycle start soon after a FreeBSD developer summit—that ensures a large number of developers are on hand to discuss any features that we want to have completed in time for the release—and since the largest developer summit has typically been at BSDCan in May or June, it makes sense to schedule the .0 releases for the second half of the year, with the release landing around November or December of every odd year.

We'd like to have a new major version every 2 years.

This gives us a schedule for when each FreeBSD release will take place:

- June 2025: FreeBSD 14.3
- (September 2025: No release this quarter; working on 15.0)
- December 2025: FreeBSD 15.0
- March 2026: FreeBSD 14.4
- June 2026: FreeBSD 15.1
- September 2026: FreeBSD 14.5
- December 2026: FreeBSD 15.2
- March 2027: FreeBSD 14.6
- June 2027: FreeBSD 15.3
- (September 2027: No release; working on 16.0)
- December 2027: FreeBSD 16.0

Support Periods

Starting with FreeBSD 11.0-RELEASE, FreeBSD's policy for supporting releases has been that minor releases are supported for three months past the date of the next release from the same stable branch, while stable branches are supported for five years from the date of the .0 release.

The first half of that—support durations for minor releases—fits neatly with the new quarterly release schedule: Just as we have a new release at the end of nearly every quarter, a release reaches its end-of-life at the end of nearly every quarter. Moreover, since releases now always occur in the third month of each quarter, the policy of "three months of overlap between minor releases" simply translates to "one quarter of overlap."

Where the new release schedule doesn't fit with the established support timelines so well is the support period for stable versions: Doing a new .0 release (and thus a new stable branch) every 2 years and supporting each stable branch for 5 years would result in 3 stable branches being supported at the same time—which experience has taught us isn't something we can do very well as a project.

Consequentially, to better align support timelines with the release schedule the Core team, security team, and ports management team have approved adjusting the support timelines for stable branches: Starting from 15.x, stable branches will be supported for 4 years instead of 5. This may be less of a change than it sounds: In practice stable branches were never supported very well in their fifth year.

Starting from 15.x, stable branches will be supported for 4 years instead of 5.

With this change, there are always two supported stable branches, aside from a window of a few weeks every second year where there are three supported branches when (N+2).0 is released shortly before the N.x branch reaches its end-of-life.

Legacy Releases

For many years parts of the FreeBSD website—sometimes commented out—have referred to "legacy" releases, but as far as I could find there was never any clear definition of what this meant. During Glen's time as a release engineer, the concept mostly fell into disuse, as he felt that calling a release "legacy" gave users a misleadingly poor sense of its quality, when in fact we aim for the same quality from all FreeBSD releases.

I decided to bring this concept back to help users (especially new users) decide which release they should be installing. To this end, I have a definition:

> *A "legacy" release is a release that has not yet reached its End of Life, but which the Release Engineering team recommends against deploying for new systems.*

In other words, it exists for the benefit of users who already have FreeBSD installed and don't want to upgrade to the latest release; but if you're installing a new system, you should probably be reaching for a newer release.

In general, once FreeBSD N.1 is released, the (N-1).x stable branch will be considered "legacy", and any time a new minor version is released from a stable branch, the previous version from that branch is immediately "legacy". Or put another way: Normally everything except the most recent release from the latest stable branch is "legacy", but immediately after a new stable branch launches with a .0 release the most recent release from the previous stable branch will remain "non-legacy".

Ultimately FreeBSD users can use whatever versions they like, and this is in no way intended to limit FreeBSD developers' decisions about merging features and bug fixes; it's purely a matter of helping new and inexperienced users pick the right version to download.

What's Next?

During 2024 I've made the changes mentioned above and managed the 13.3, 14.1, 13.4, and 14.2 releases. What comes next?

Well, for a start we have three releases scheduled for 2025: 13.5, 14.3, and 15.0. The minor releases I've done so far have each taken 50-100 hours of my time (13.x at the low end of the range and 14.x at the high end, since more new code lands in 14.x) and I've been very lucky to have sponsorship from Amazon for my FreeBSD work (both for maintaining FreeBSD specifically on the EC2 platform and for general release engineering work)—without that, FreeBSD 14.2 would have shipped without some late-landing features simply because it wouldn't have been possible for issues to get fixed in time. FreeBSD 15.0, of course, will be a new major version—something I've never done before—and I'm sure it will take considerably more release engineering time.

But beyond the "routine" process of pushing out weekly snapshots and (mostly) quarterly releases, there are two big items on the horizon: First, FreeBSD 15.0 should ship with a packaged-based system—pkgbase has been "coming soon" for far too long already—and second, the FreeBSD Foundation, with funding from the Sovereign Tech Agency, is funding a project to modernize the FreeBSD build process (in particular, to make it possible to build as much as possible without root privileges). Both items will involve significant changes in the release engineering process, but neither of them should affect our goal of producing stable and well-tested releases on a predictable schedule.

COLIN PERCIVAL is the FreeBSD Release Engineering Lead and maintainer of the FreeBSD/EC2 platform, for which he was recognized as an "AWS Hero" in 2019. His day job is as the founder and primary developer of Tarsnap, an online backup service.