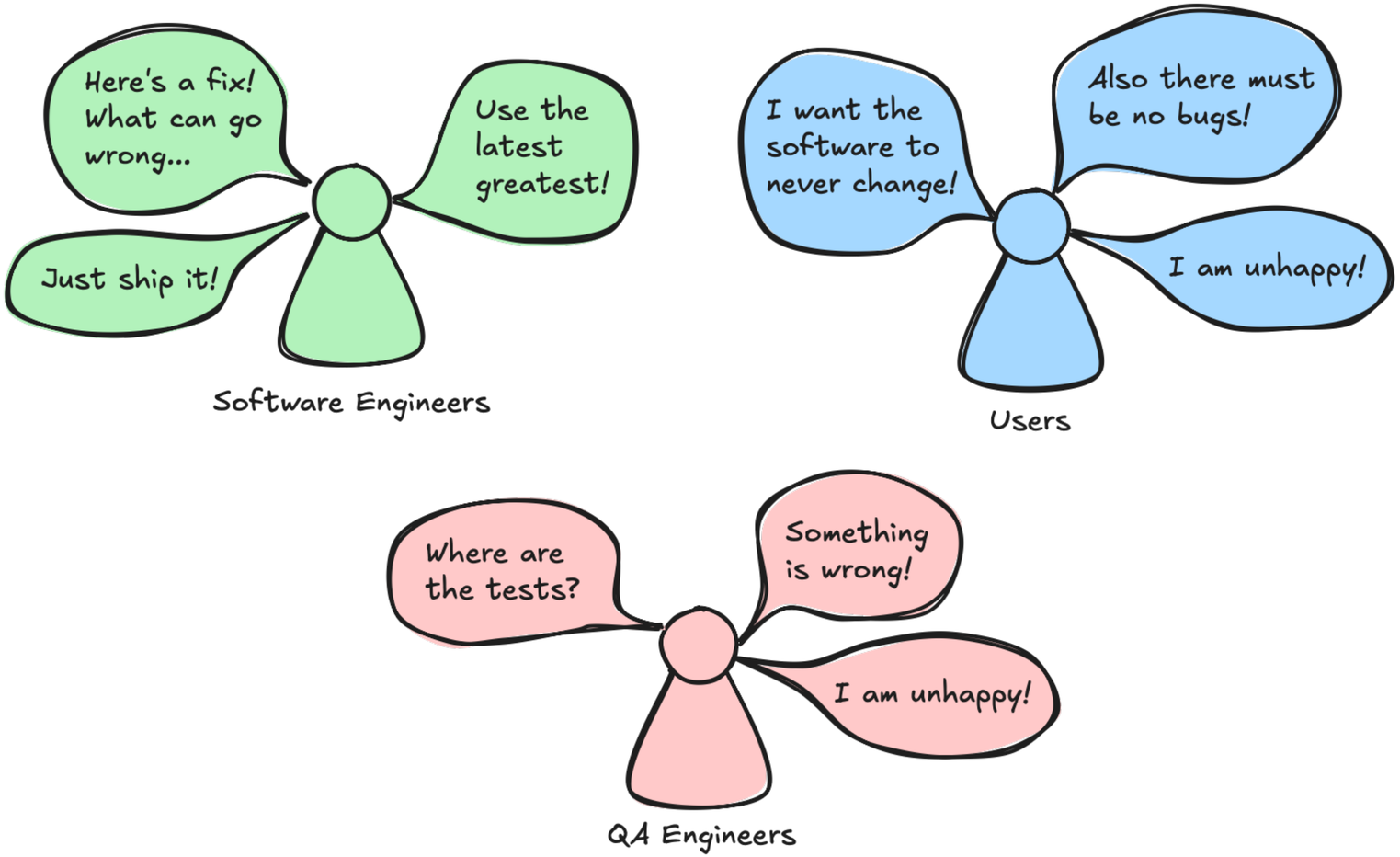
Open-Source Release Engineering

Streamlining the release process from git

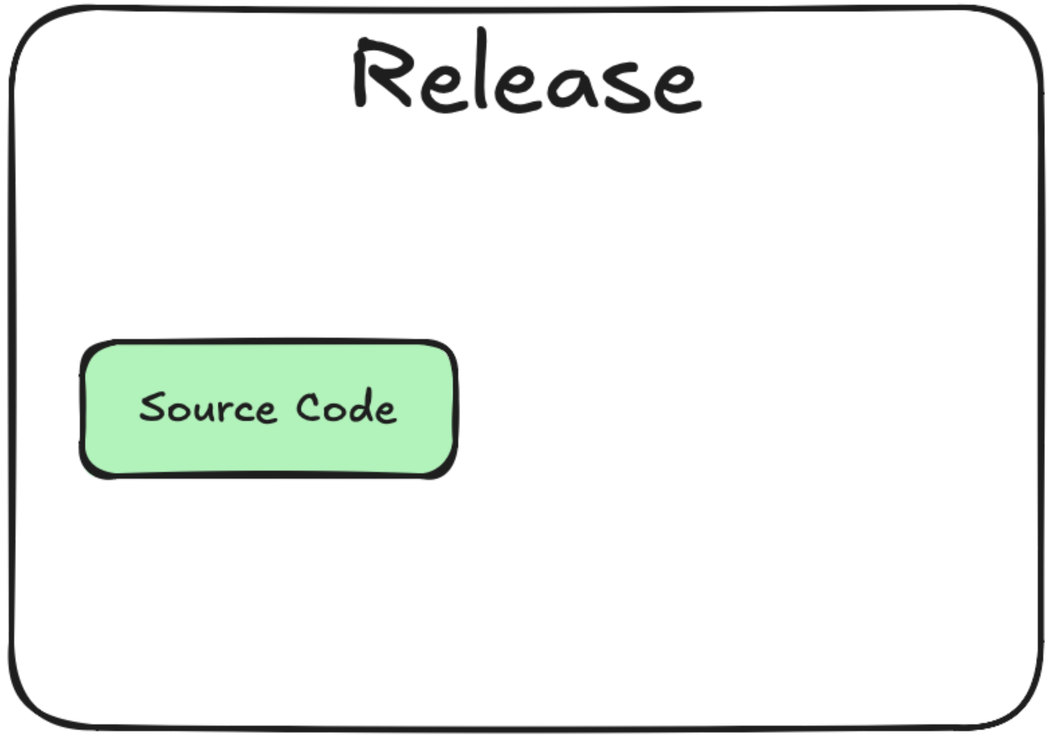
Ondřej Surý, ISC; RIPE 91 22.10.2025

The Everlasting Clash

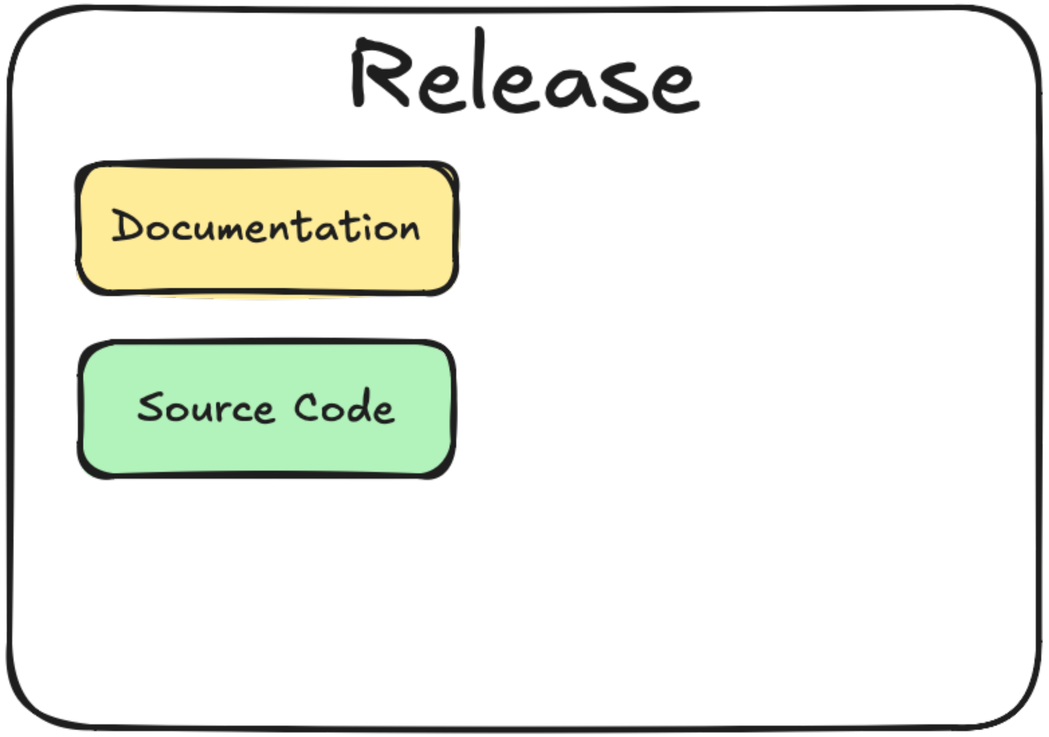
Software Engineers vs QA vs Users



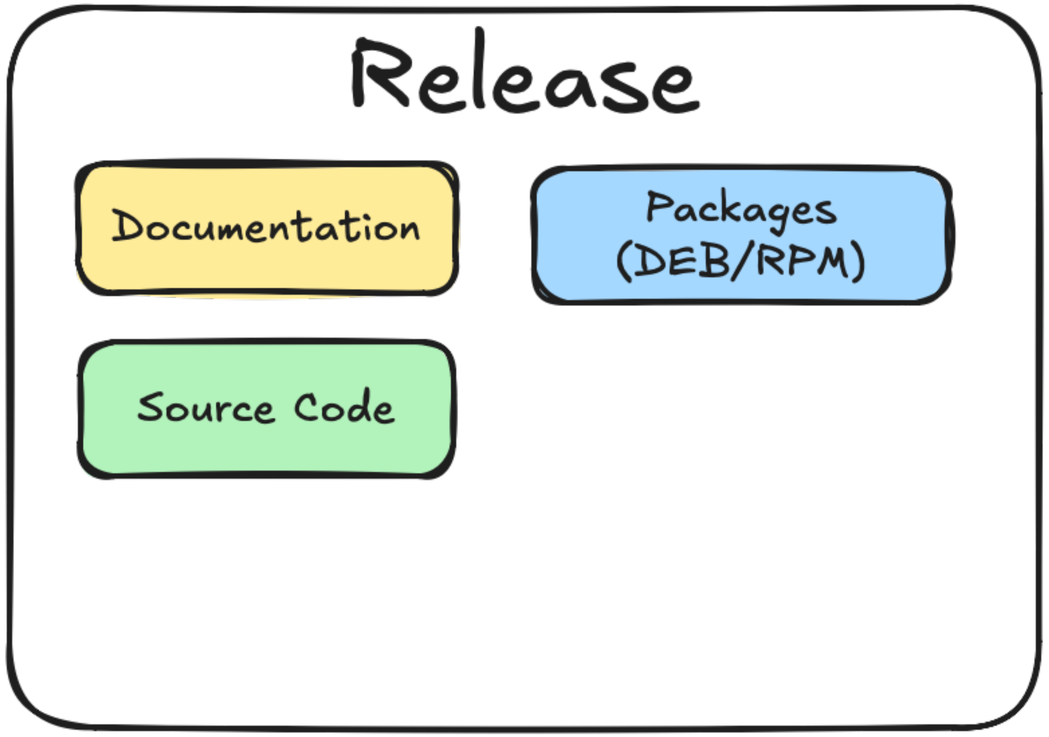
Release Engineering



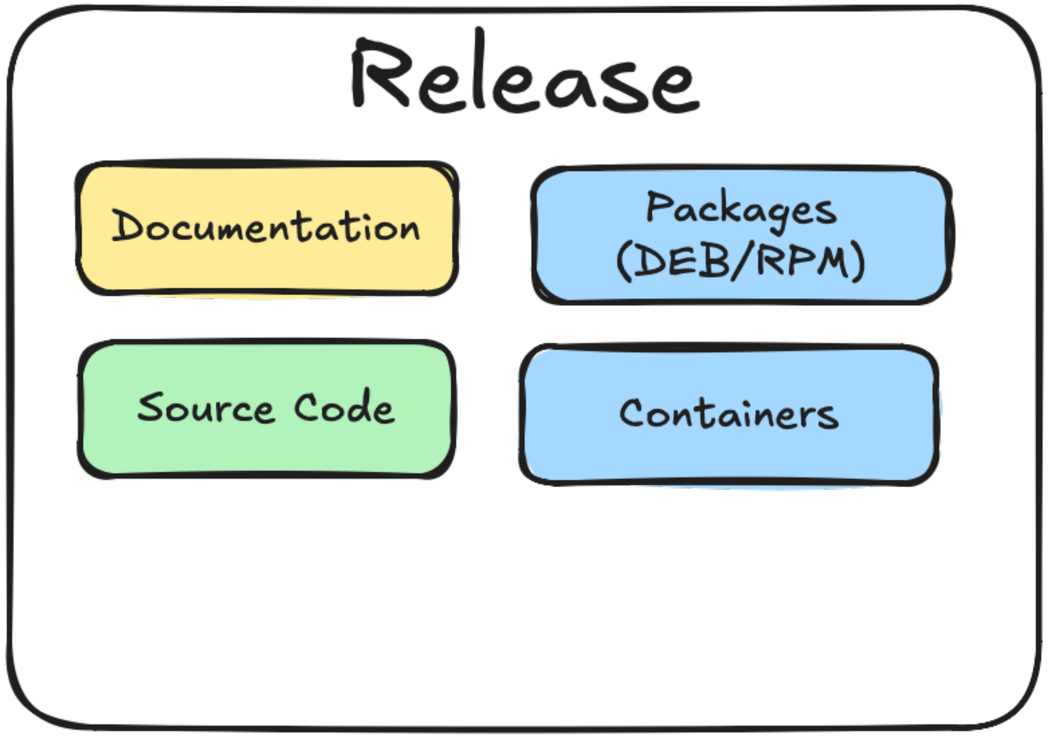
Release Engineering



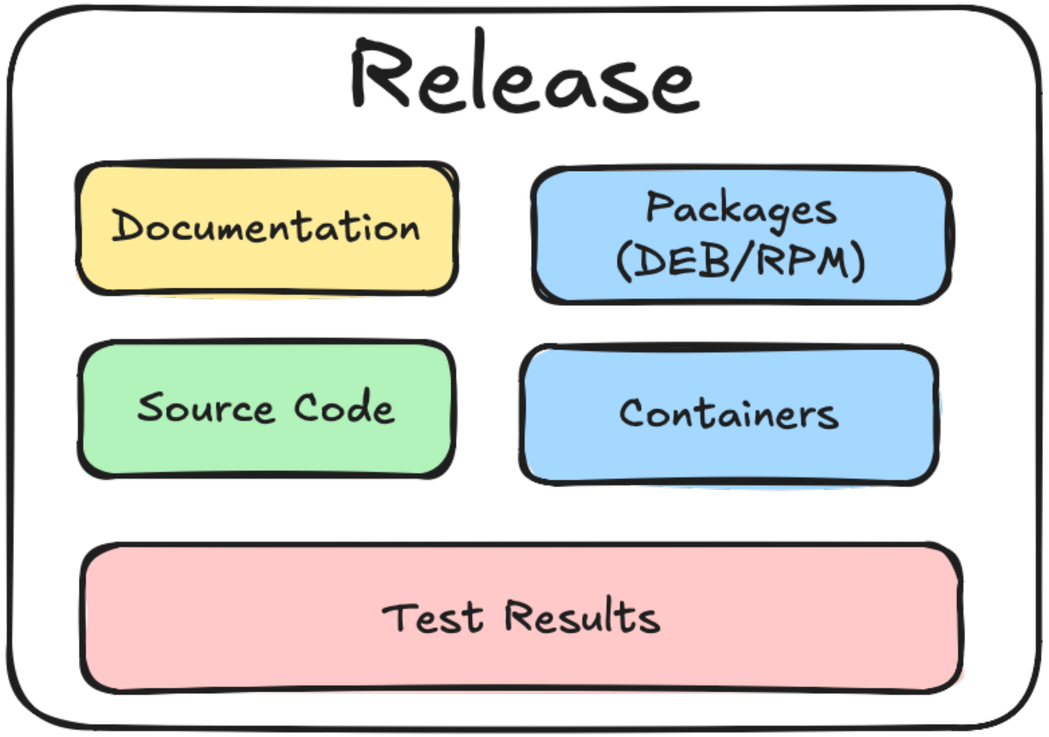
Release Engineering



Release Engineering

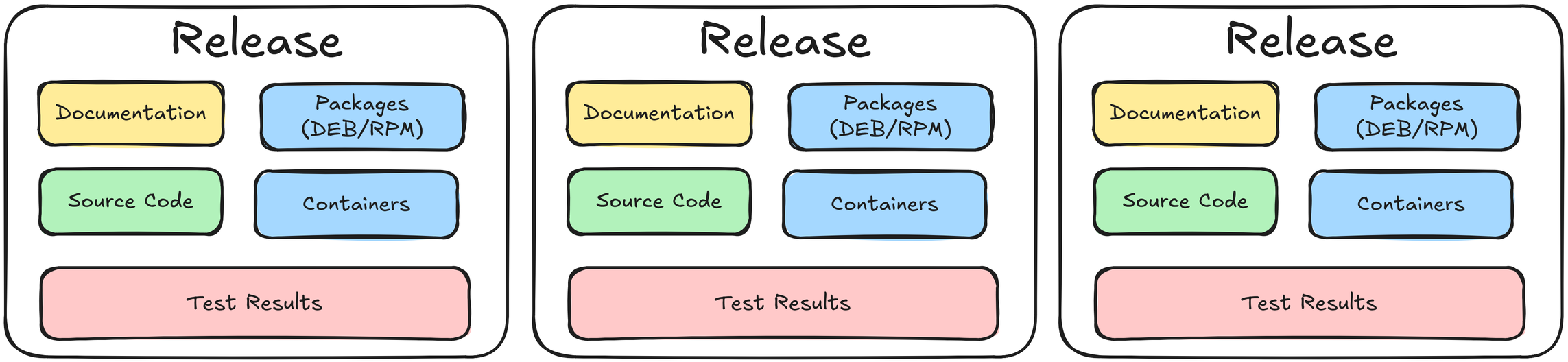


Release Engineering



Release Engineering

Multiple Releases



Release Checklist for BIND 9

Open · Issue created 3 weeks ago by **Michał Kępień** · Show sidebar

Release Checklist for BIND 9.18.40, 9.18.40-S1, 9.20.14/9.20.15, 9.20.14-S1/9.20.15-S1, 9.21.13

Release Schedule

Code Freeze: Wednesday, 1 October 2025

Tagging Deadline: Monday, 6 October 2025

EVN Release: Wednesday, 8 October 2025

Public Release: Wednesday, 22 October 2025

Documentation Review Links

Closed issues assigned to the milestone without a release note:

- 9.18.40
- 9.18.40-S1
- 9.20.14/9.20.15
- 9.20.14-S1/9.20.15-S1
- 9.21.13

Merge requests merged into the milestone without a release note:

- 9.18.40
- 9.18.40-S1
- 9.20.14/9.20.15
- 9.20.14-S1/9.20.15-S1
- 9.21.13

Merge requests merged into the milestone without a changelog entry:

- 9.18.40
- 9.18.40-S1
- 9.20.14/9.20.15
- 9.20.14-S1/9.20.15-S1
- 9.21.13

Release Checklist

Before the Code Freeze

- [x] *(QA)* Rebase -S editions on top of current open-source versions: `git checkout bind-9.18-sub && git rebase --rebase-merges origin/bind-9.18`
- [x] *(QA)* Inform Support and Marketing of impending release (and give estimated release dates).
- [x] *(QA)* Ensure there are no permanent test failures on any platform. Check public and private scheduled pipelines.
- [ ] ~~*(QA)* Check charts from `shotgun:*` jobs in the scheduled pipelines to verify there is no unexplained performance drop for any protocol. (Shotgun tests are currently b0rked, so there is no way to check this.)~~
- [x] *(QA)* Check Perflab to ensure there has been no unexplained drop in performance for the versions being released.

Release Checklist

Before the Code Freeze

- [x] *(QA)* Rebase -S editions on top of current open-source versions: `git checkout bind-9.18-sub && git rebase --rebase-merges origin/bind-9.18`
- [x] *(QA)* Inform Support and Marketing of impending release (and give estimated release dates).
- [x] *(QA)* Ensure there are no permanent test failures on any platform. Check public and private scheduled pipelines.
- [ ] ~~*(QA)* Check charts from `shotgun:*` jobs in the scheduled pipelines to verify there is no unexplained performance drop for any protocol. (Shotgun tests are currently b0rked, so there is no way to check this.)~~
- [x] *(QA)* Check Perflab to ensure there has been no unexplained drop in performance for the versions being released.
- [ ] ~~*(QA)* Update GitLab settings for all maintained branches to disallow merging to them: public, private~~
- [ ] ~~*(QA)* Announce (on Mattermost) that the code freeze is in effect.~~
  - ▸ ~~Details~~

Before the Tagging Deadline

- [x] *(QA)* Ensure all merge requests marked for backporting have been indeed backported.
- [x] *(QA)* Check whether all issues and merge requests assigned to the release milestone are resolved[1].
- [x] *(QA)* Branch-off open-source release branches (e.g. `v9.20.X-release`) from all maintained branches and push them to the private repository.
- [x] *(QA)* Merge any outstanding merge requests in the private repository[1] to the open-source release branches. (e.g. security fixes)
- [x] *(QA)* Rebase -S editions on top of the open-source release branches and branch-off -S release branches (e.g. `v9.20.X-S1-release`).
- [ ] ~~*(QA)* Merge any outstanding -S specific merge requests in the private repository[1] to the -S release branches. (-S specific fixes)~~
- [ ] ~~*(QA)* Inspect the current output of the `cross-version-config-tests` job to verify that no unexpected backward-incompatible change was introduced in the current release cycle.~~
- [ ] ~~*(QA)* Reword any -S specific commits from previous release that would generate a release note or changelog entry.~~
- [x] *(QA)* Generate changelog and release notes from git log. (Run prep_doc_mr.py for instructions.)
- [x] *(QA)* Ensure release notes are correct, ask Support and Marketing to check them as well. Example
- [x] *(QA)* Update BIND 9 version in `configure.ac` (9.18+)
- [x] *(QA)* Tag the releases in the private repository (`git tag -s -m "BIND 9.x.y" v9.x.y`).
- [x] *(QA)* Assign the merged MRs to versioned milestones.
- [ ] ~~*(QA)* Update GitLab settings for all maintained branches to allow merging to them again: public, private~~
- [ ] ~~*(QA)* Announce (on Mattermost) that the code freeze is over.~~
  - ▸ ~~Details~~

Before the EVN Deadline

Wednesday, 8 October 2025

- [x] *(QA)* Check that the formatting is correct for the HTML version of release notes.
- [x] *(QA)* Verify GitLab CI results for the tags created and sign off on the releases to be published.
- [x] *(QA)* Prepare and merge MRs updating the version string for each maintained branch (using `version_bump.py`).
- [x] *(QA)* Rebase the Subscription Edition branches (including recent release prep commits) on top of the open source branches with updated version strings.
- [x] *(QA)* Request signatures for the tarballs; ask signers on Mattermost.
- [x] *(Signers)* Ensure that the contents of tarballs and tags are identical (using `release-tarball-comparison.sh`).
- [x] *(Signers)* Sign tarballs and upload signatures.
- [x] *(QA)* Verify tarball signatures and check tarball checksums again: Run `publish_bind.sh` on repo.isc.org to pre-publish.
- [x] *(QA)* Prepare the `patches/` subdirectory for each security release (if applicable).
- [x] *(QA)* Pre-publish EVN and/or Subscription Edition tarballs so that packages can be built.

On the Day of Public Release

Wednesday, 15 October 2025

- [ ] *(QA)* Ask for clearance from Incident Managers to p
- [ ] *(QA)* Update Known Issues lists (9.18, 9.20).
- [ ] *(QA)* Ensure all new tags are annotated and signed
- [ ] *(QA)* Push tags to customer git repositories by trig … s in the private repo.
- [ ] *(QA)* Push tags for the published releases to the public repository.
- [ ] *(QA)* Push the `stable` tag to be the same as the latest `v9.20.x`.
- [ ] *(QA)* Place tarballs in public l… tion on FTP s… e
- [ ] *(QA)* Inform Marketing of th…
- [ ] *(QA)* Use the Printing Press proje… to prepare a release announcement email.
- [ ] *(Marketing)* Before proceeding with publication, check any CVE checklists associated with this release to ensure they are ready for publication.
- [ ] *(Marketing)* Publish links to downloads on ISC website. Update hidden security releases table in the downloads data file in case of security release. `Example`
- [ ] ~~*(Marketing)* Update the BIND -S information ticket in the support portal with download links to the new versions. (If this is a security release, this will have already been done as part of the EVN process.)~~
- [ ] *(Marketing)* Open a ticket in the read-only BIND-Announce queue in the support portal to announce the release to support customers if a major release or significant change to a stable version (or packages).
- [ ] *(Marketing)* Send the release announcement email to the *bind-announce* mailing list (and to *bind-users* if a major release - example).
- [ ] *(Marketing)* Announce release on social media sites.
- [ ] *(Marketing)* Update Wikipedia entry for BIND.
- [ ] *(Marketing)* Update RT article informing customers of current versions and release schedule.
- [ ] *(Support)* Add the new releases to the vulnerability matrix in the Knowledge Base.
- [ ] *(Support)* Update tickets in case of waiting support customers.
- [ ] *(QA)* Build, test, and publish any outstanding private packages in private repo. `Example`
- [ ] *(QA)* Build public RPMs. `Example commit` which triggers Copr builds automatically
- [ ] *(SwEng)* Build Debian/Ubuntu packages.
- [ ] *(SwEng)* Update Docker files here and make sure push is synchronized to GitHub. Docker Hub should pick it up automatically. `Example`
- [ ] *(QA)* Using `merge_tag.py`, merge published release tags back into the their relevant development/maintenance branches.
- [ ] *(QA)* Ensure `allow_failure: true` is removed from the `cross-version-config-tests` job if it was set during the current release cycle.
- [ ] *(QA)* Sanitize confidential issues which are assigned to the current release milestone and do not describe a security vulnerability, then make them public.
- [ ] *(QA)* Sanitize confidential issues which are assigned to older release milestones and describe security vulnerabilities, then make them public if appropriate[2].
- [ ] *(QA)* Update QA tools used in GitLab CI (e.g. Black, PyLint, Sphinx) by modifying the relevant `Dockerfile`.
- [ ] *(QA)* Manually create a pipeline and start all jobs in it to rebuild all images used in GitLab CI.
- [ ] *(QA)* Update `metadata.json` with the upcoming release information.
- [ ] *(QA)* Close all GitLab milestones for this release cycle (using `close_milestones.py`).

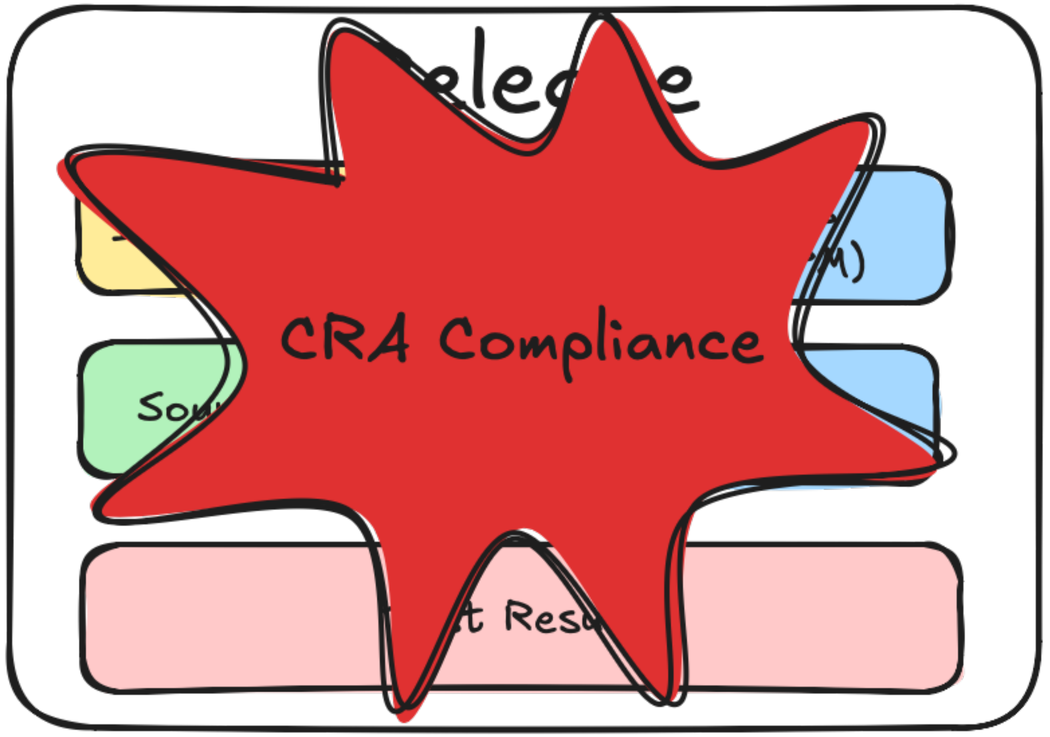
---

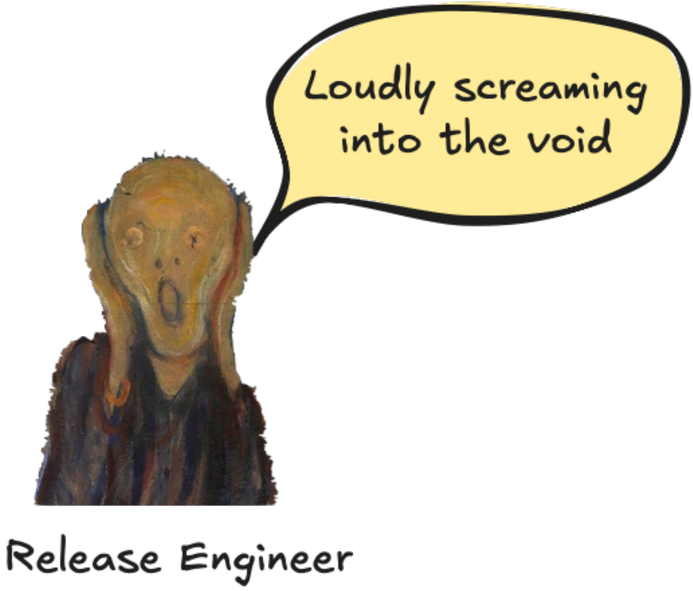
1. If not, use the time remaining until the tagging deadline to ensure all outstanding issues are either resolved or moved to a different milestone. ↩ ↩2 ↩3
2. As a rule of thumb, security vulnerabilities which have reproducers merged to the public repository are considered okay for full disclosure. ↩

27 of 57 checklist items completed · Edited 1 day ago by Michał Kępień

https://gitlab.isc.org/isc-projects/bind9/-/issues/5534

Release Engineering





Does it have to be this way?

What is the "Release" anyway?

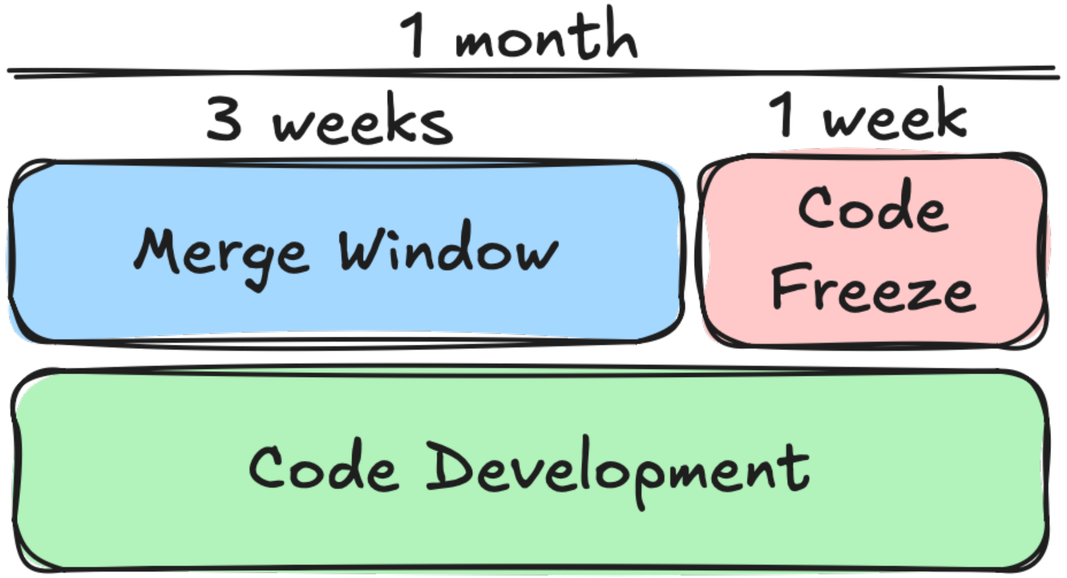
- Contains:
  - Source Code
  - Documentation
  - Build System
- Artifacts:
  - Source Code Tarball
  - Binary Packages
  - Release Notes
  - Documentation
  - Test Results

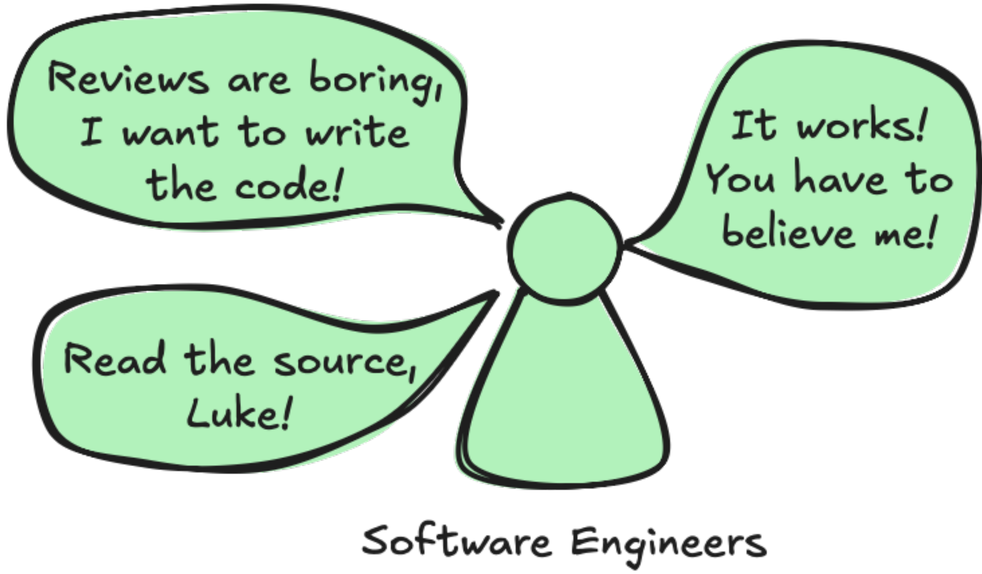
Reviewed and Tested Source Code

Reviewed and Tested Source Code

Takes a lot of extra time if people don't work together!

- Freeze the release branches (for up to a week)
- Run a comprehensive test suites
- Check the results

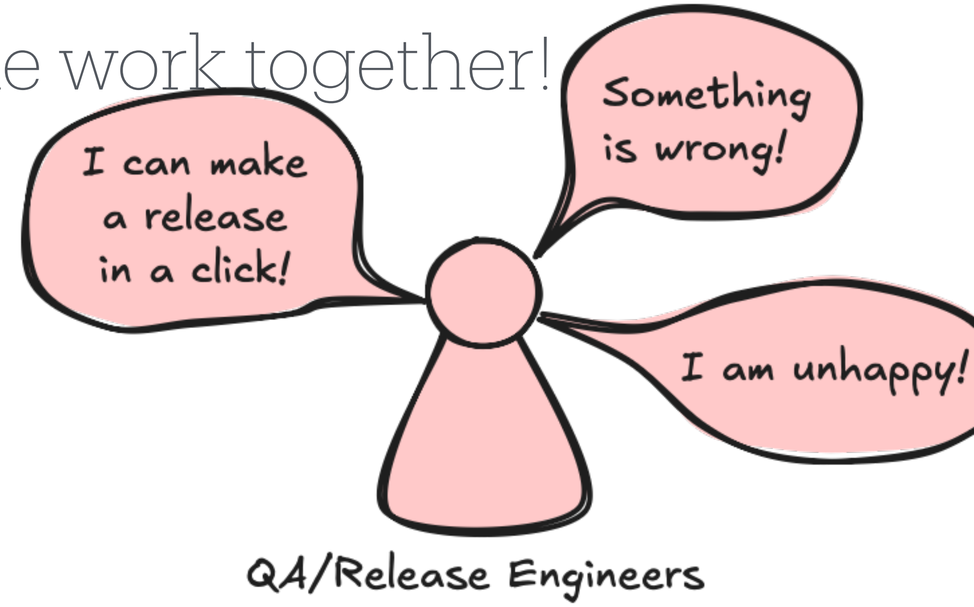


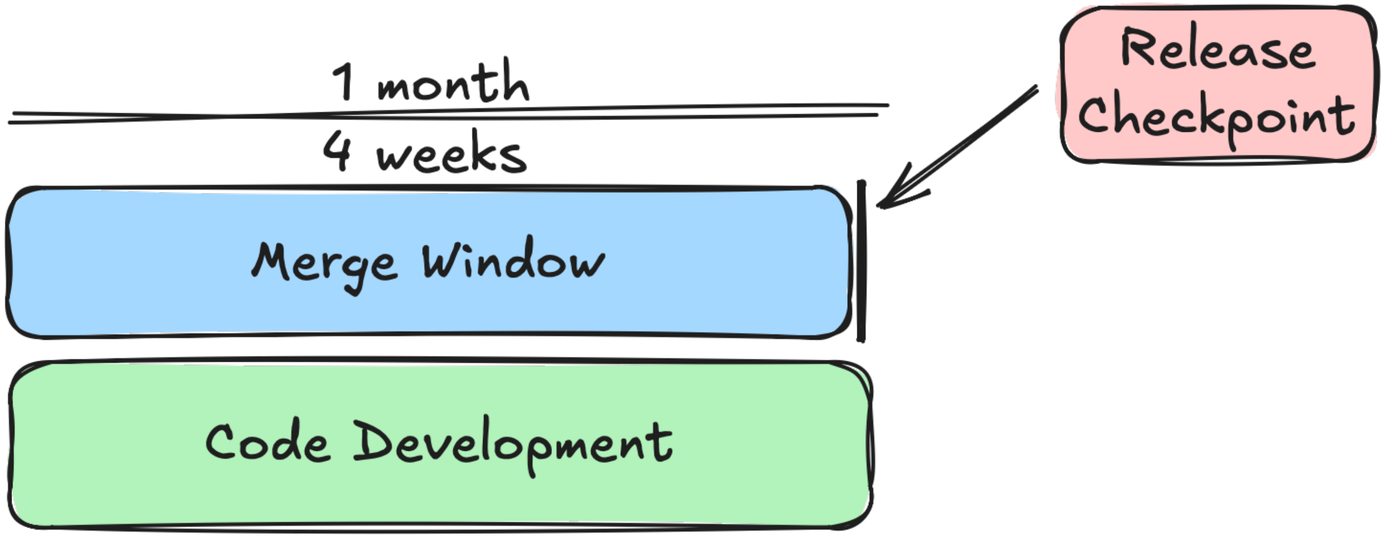


Reviewed and Tested Source Code

Doesn't have to be time-consuming if people work together!

- Release Branches Quality Assurance
  - Always stable
  - Always reviewed
  - Always tested
  - Always documented
  - **Always ready for the release!**





What is testing anyway?

Source Code Testing

- Unit Testing
- System Testing
- Cross-Version Testing
- Interoperability Testing
- Performance Testing
- User-Deployment Testing (the BEST ONE)

Quality Assurance Extras

Not just tests

- Style and Formatting
  - Agree on a single source code style and formatting
  - **Use tools to help** and enforce
- Source Code Style Tools (BIND 9 toolbox)
  - C – clang-format, clang-tidy, coccinelle
  - Python – black, pylint, vulture, mypy
  - Shell – checkbashism, shfmt

Documentation

Software Engineers

- Commit Messages
- Comments in the Code
- Source itself

"We have seen that computer programming is an art, because it applies accumulated knowledge to the world, because it requires skill and ingenuity, and especially because it produces objects of beauty. A programmer who subconsciously views himself as an artist will enjoy what he does and will do it better."

Knuth D., Computer Programming as an Art, CACM, December 1974

Documentation

End Users

- Reference Manual
  - For long winter evenings
- Knowledge Base
  - How do I do?
- Manual Pages
  - How do I do?
- Release Notes
  - Why is this not working?

Support Engineers

- Reference Manual
- Knowledge Base
  - Written by Support Staff
- Release Notes
- Changelog
  - More detailed than release notes, but less detailed than git

Release Notes

Before

1. Modify the source code
2. Modify the `doc/notes/notes-<relver>.rst`
3. Create merge request
4. Support Engineers spell-check our non-native speaker English
5. Rebase the branch
6. Resolve the rebase conflict
7. Go to 5. (ad infinitum until finally merged)

Release Notes

Now with git-changelog

1. Modify the source code
2. Create the merge request
3. Write solid merge request description with markup in the title
4. Support Engineers spell-check our non-native speaker English
5. Merge the merge request

Release Notes Examples

New User Feature

new: usr: Add manual mode configuration option to dnsec-policy

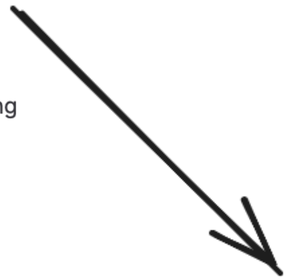
Merged **Matthijs Mekking** requested to merge `4606-dnssec-policy-dry-run` into `main` 2 months ago

Overview 17 | Commits 13 | Pipelines 8 | Changes 52 | All

Add a new option `manual-mode` to :any:`dnssec-policy`. The intended use is that if it is enabled, it will not automatically move to the next state transition, but instead the transition is logged. Only after manual confirmation with `rndc dnssec -step` the transition is made.

Closes #4606 (closed)

Edited 1 month ago by Matthijs Mekking



Notes for BIND 9.20.13

New Features

- Add a new option `manual-mode` to **`dnssec-policy`**.

  When enabled, **`named`** will not modify DNSSEC keys or key states automatically. The proposed change will be logged and only after manual confirmation with `rndc dnssec -step` will the modification be made. [GL #4606]

Release Notes Examples

Removed Feature

rem: usr: Deprecate the "tkey-gssapi-credential" statement

Merged **Michał Kępień** requested to merge `4204-deprecate-tkey-gssapi…` into `main` 2 months ago

Overview 13 | Commits 1 | Pipelines 5 | Changes 3

The :any:`tkey-gssapi-keytab` statement allows GSS-TSIG to be set up in a simpler and more reliable way than using the :any:`tkey-gssapi-credential` statement and setting environment variables (e.g. `KRB5_KTNAME`). Therefore, the :any:`tkey-gssapi-credential` statement has been deprecated; :any:`tkey-gssapi-keytab` should be used instead.

For configurations currently using a combination of both :any:`tkey-gssapi-keytab` *and* :any:`tkey-gssapi-credential`, the latter should be dropped and the keytab pointed to by :any:`tkey-gssapi-keytab` should now only contain the credential previously specified by :any:`tkey-gssapi-credential`.

See #4204 (closed)

Edited 2 months ago by Michał Kępień

Removed Features

- Deprecate the **`tkey-gssapi-credential`** statement.

  The **`tkey-gssapi-keytab`** statement allows GSS-TSIG to be set up in a simpler and more reliable way than using the **`tkey-gssapi-credential`** statement and setting environment variables (e.g. `KRB5_KTNAME`). Therefore, the **`tkey-gssapi-credential`** statement has been deprecated; **`tkey-gssapi-keytab`** should be used instead.

  For configurations currently using a combination of both **`tkey-gssapi-keytab`** *and* **`tkey-gssapi-credential`**, the latter should be dropped and the keytab pointed to by **`tkey-gssapi-keytab`** should now only contain the credential previously specified by **`tkey-gssapi-credential`**. **[GL #4204]**

Release Notes Examples

Bug Fix

fix: usr: Fix the default interface-interval from 60s to 60m

Merged **Ondřej Surý** requested to merge `5246-fix-default-interface…` into `main` 7 months ago

Overview 2 | Commits 1 | Pipelines 4 | Changes 2

When the interface-interval parser was changed from uint32 parser to duration parser, the default value stayed at plain number `60` which now means 60 seconds instead of 60 minutes. The documentation also incorrectly states that the value is in minutes. That has been fixed.

Closes #5246 (closed)

Edited 7 months ago by Ondřej Surý

Bug Fixes

- Correct the default `interface-interval` from 60s to 60m.

  When the `interface-interval` parser was changed from a `uint32` parser to a duration parser, the default value stayed at plain number `60` which now means 60 seconds instead of 60 minutes. The documentation also incorrectly states that the value is in minutes. That has been fixed. [GL #5246]

Our git-changelog syntax

- Action
  - '**chg**' is for refactor, small improvement, cosmetic changes...
  - '**fix**' is for bug fixes
  - '**new**' is for new features, big improvement
  - '**rem**' is for removed features
  - '**sec**' is for security fixes

- Audience:
  - Included in release notes and changelog:
    - '**usr**' is for final users
    - '**pkg**' is for packagers
  - Included in changelog:
    - '**dev**' is for developers
  - Omitted from changelog:
    - '**test**' is for test-only changes
    - '**doc**' is for doc-only changes
    - '**nil**' is for ... just nobody should care about this

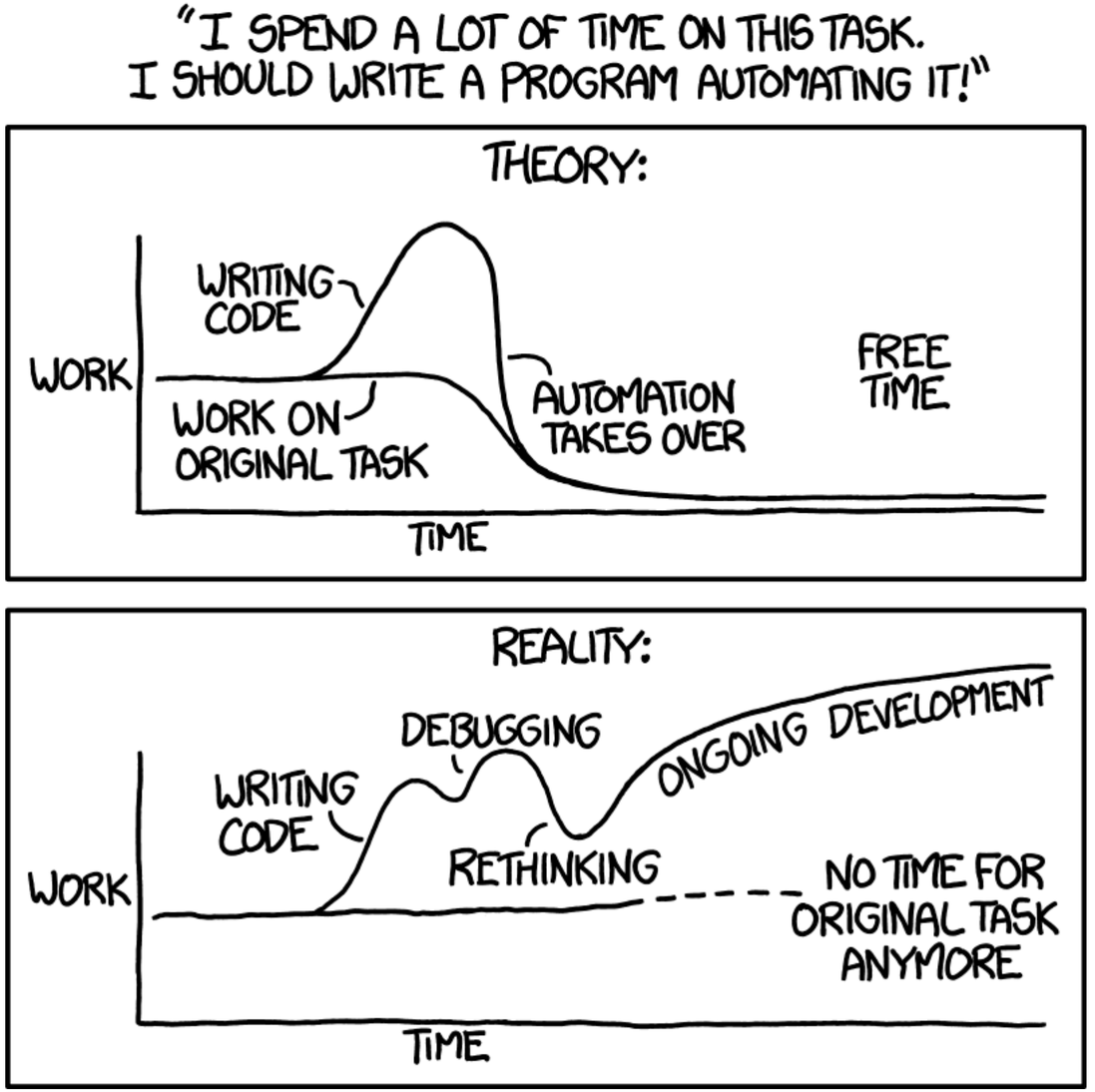
Security Release

- Security Releases are same but different
- Everything needs to be done in the private
  - Means only internal testing is done…
  - It helps to have small group of outside testers for early deployment (tyvm)
- Coordinated security releases are the worst
  - Fixed schedule – between other vendors and security researchers
  - If you make a mistake, everyone has to reschedule

Release Engineering

Summary

- Release Branches always in good shape
- Release Engineering Automated
  - Ideally – just click here and magic happens
  - Takes a lot of time to automate, but saves time
- Release Notes
  - Use tools, not a manual process
  - Fixing merge conflicts is annoying and error-prone



https://xkcd.com/1319/

Thank you