Releasing Rails

In this document, we'll cover the steps necessary to release Rails. Each section contains steps to take during that time before the release. The times suggested in each header are just that: suggestions. However, they should really be considered as minimums.

10 Days before release

Today is mostly coordination tasks. Here are the things you must do today:

Is the CI green? If not, make it green. (See "Fixing the CI")

Do not release with a Red CI. You can find the CI status here:

`https://buildkite.com/rails/rails`

Is Sam Ruby happy? If not, make him happy.

Sam Ruby keeps a test suite that makes sure the code samples in his book (Agile Web Development with Rails) all work. These are valuable system tests for Rails. You can check the status of these tests here:

https://intertwingly.net/projects/dashboard.html

Do not release with Red AWDwR tests.

Do we have any Git dependencies? If so, contact those authors.

Having Git dependencies indicates that we depend on unreleased code. Obviously Rails cannot be released when it depends on unreleased code. Contact the authors of those particular gems and work out a release date that suits them.

Contact the security team (either tenderlove or rafaelfranca)

Let them know of your plans to release. There may be security issues to be addressed, and that can impact your release date.

Notify implementors.

Ruby implementors have high stakes in making sure Rails works. Be kind and give them a heads up that Rails will be released soonish.

This is only required for major and minor releases, bugfix releases aren't a big enough deal, and are supposed to be backward compatible.

Send a message just giving a heads up about the upcoming release to these lists:

- team@jruby.org
- community@rubini.us

- rubyonrails-core

Implementors will love you and help you.

3 Days before release

This is when you should release the release candidate. Here are your tasks for today:

Is the CI green? If not, make it green.

Is Sam Ruby happy? If not, make him happy.

Contact the security team. CVE emails must be sent on this day.

Create a release branch.

From the stable branch, create a release branch. For example, if you're releasing Rails 3.0.10, do this:

```
[aaron@higgins rails (3-0-stable)]$ git checkout -b 3-0-10
Switched to a new branch '3-0-10'
[aaron@higgins rails (3-0-10)]$
```

Update each CHANGELOG.

Many times commits are made without the CHANGELOG being updated. You should review the commits since the last release, and fill in any missing information for each CHANGELOG.

You can review the commits for the 3.0.10 release like this:

```
[aaron@higgins rails (3-0-10)]$ git log v3.0.9..
```

If you're doing a stable branch release, you should also ensure that all of the CHANGELOG entries in the stable branch are also synced to the main branch.

Put the new version in the RAILS_VERSION file.

Include an RC number if appropriate, e.g. `6.0.0.rc1`.

Build and test the gem.

Run `rake verify` to generate the gems and install them locally. `verify` also generates a Rails app with a migration and boots it to smoke test with in your browser.

This will stop you from looking silly when you push an RC to rubygems.org and then realize it is broken.

Release to RubyGems and npm.

IMPORTANT: Several gems have JavaScript components that are released as npm packages, so you must have Node.js installed, have an npm account (npmjs.com), and be a package owner for `@rails/actioncable`, `@rails/actiontext`, `@rails/activestorage`, and `@rails/ujs`. You can check this by making sure your npm user (`npm whoami`) is listed as an owner (`npm owner ls <pkg>`) of each package. Do not release until you're set up with npm!

The release task will sign the release tag. If you haven't got commit signing set up, use https://git-scm.com/book/en/v2/Git-Tools-Signing-Your-Work as a guide. You can generate keys with the GPG suite from here: https://gpgtools.org.

Run `rake changelog:header` to put a header with the new version in every CHANGELOG. Don't commit this, the release task handles it.

Run `rake release`. This will populate the gemspecs and npm package.json with the current RAILS_VERSION, commit the changes, tag it, and push the gems to rubygems.org.

Send Rails release announcements

Write a release announcement that includes the version, changes, and links to GitHub where people can find the specific commit list. Here are the mailing lists where you should announce:

- rubyonrails-core
- rubyonrails-talk
- ruby-talk@ruby-lang.org

Use Markdown format for your announcement. Remember to ask people to report issues with the release candidate to the rails-core mailing list.

NOTE: For patch releases, there's a `rake announce` task to generate the release post. It supports multiple patch releases too:

```
VERSIONS="5.0.5.rc1,5.1.3.rc1" rake announce
```

IMPORTANT: If any users experience regressions when using the release candidate, you *must* postpone the release. Bugfix releases *should not* break existing applications.

Post the announcement to the Rails blog.

If you used Markdown format for your email, you can just paste it into the blog.

- https://rubyonrails.org/blog

Post the announcement to the Rails Twitter account.

Time between release candidate and actual release

Check the rails-core mailing list and the GitHub issue list for regressions in the RC.

If any regressions are found, fix the regressions and repeat the release candidate process. We will not release the final until 72 hours after the last release candidate has been pushed. This means that if users find regressions, the scheduled release date must be postponed.

When you fix the regressions, do not create a new branch. Fix them on the stable branch, then cherry pick the commit to your release branch. No other commits should be added to the release branch besides regression fixing commits.

Day of release

Many of these steps are the same as for the release candidate, so if you need more explanation on a particular step, see the RC steps.

Today, do this stuff in this order:

- Apply security patches to the release branch
- Update CHANGELOG with security fixes
- Update RAILS_VERSION to remove the rc
- Build and test the gem
- Release the gems
- If releasing a new stable version:
  - Trigger stable docs generation (see below)
  - Update the version in the home page
- Email security lists
- Email general announcement lists

Emailing the Rails security announce list

Email the security announce list once for each vulnerability fixed.

You can do this, or ask the security team to do it.

Email the security reports to:

- rubyonrails-security@googlegroups.com
- oss-security@lists.openwall.com

Be sure to note the security fixes in your announcement along with CVE numbers and links to each patch. Some people may not be able to upgrade right away, so we need to give them the security fixes in patch form.

- Blog announcements
- Twitter announcements

- Merge the release branch to the stable branch
- Drink beer (or other cocktail)

Misc

Fixing the CI

There are two simple steps for fixing the CI:

1. Identify the problem
2. Fix it

Repeat these steps until the CI is green.