CHEF Release Engineering OSS Management

In open source software projects, it is important that the software is released early and often. It is essential to have a well-defined process in place for creating these releases. This does not mean that a release will be complete or that it will be bug-free. However, it does mean the status and changeset of each release is well documented to manage user expectations.

This document details the processes the Release Engineering team at CHEF uses to triage, merge, and release its software projects. This process may differ from other teams, so please consult their release process as necessary. This process is always under refinement, so please leave a comment if you have a suggested change.

This team supports the following projects:

- https://github.com/opscode-cookbooks/jenkins
- https://github.com/opscode-cookbooks/omnibus
- https://github.com/opscode-cookbooks/remote_install
- https://github.com/opscode/artifactory-client
- https://github.com/opscode/bento
- https://github.com/opscode/mixlib-versioning
- https://github.com/opscode/omnibus-harmony
- https://github.com/opscode/omnibus-ruby
- https://github.com/opscode/omnibus-software

Process

This team manages all projects using GitHub Issues and Pull Requests. For more information on "the GitHub flow", please see http://guides.github.com/overviews/flow/.

With exceptions given to holidays and conferences, this team triages issues on a weekly basis as part of a standing team meeting. As time permits, issues may receive earlier attention. While we would love to spend all our time working on open source projects, please be aware that we maintain these projects alongside other pressing work. Issues are triaged in the following priority order:

- Bugfixes (Pull Request)
- Bug Reports
- Questions
- Documentation
- New Features (Pull Request)
- Feature Requests

Because all projects include a minimum viable test suite that automatically runs on a continuous integration service, we will not merge any Pull Requests that do not have a green build. By proxy, we will not release a new version that does not have a new build.

This team does not schedule releases on a predetermined schedule, but the following cases will receive special releasing priority:

- Security fixes
- Breaking API changes that violate semantic versioning

All projects follow the semantic versioning specification as documented at http://semver.org/ with the following exceptions:

- Pre-release versions for cookbooks is not supported

This team will announce new release via the following channels:

- Twitter (@chef)
- Chef Mailing List

Merging

The following section documents the merging process. This document is primarily for internal use or by contributors to a project.

It is highly recommended that you install the `hub` gem for interacting with Pull Requests:

```
gem install hub
```

First, ensure you have the latest version of the code:

```
git fetch --all && git pull origin master
```

Use the `hub` gem to checkout the Pull Request:

```
hub checkout <link-to-pull-request>
```

This will checkout a new branch and pull the code from GitHub. Rebase this feature branch against the latest `master` branch:

```
git rebase origin/master
```

Checkout the master branch and merge the feature branch:

```
git checkout master && git merge <branch>
```

If you are prompted for a merge commit message, something went wrong with the rebase. You should not be prompted for a message; the merge should be clean.

When the author's commit messages are not descriptive, you squash the Pull Request into a single commit when merging.

```
git merge --squash <branch>
git commit -m "<descriptive message of all the changes>"
```

Push the code back to the `master` branch on GitHub:

```
git push origin master
```

Depending on the rebase situation, you may need to manually close the Pull Request via the GitHub interface. Leave the following comment:

```
Ohai @<username>. Thank you for the Pull Request. This was merged in
<SHA>.
```

If applicable, delete the remote branch (this only applies when the branch exists on the parent repository).

Releasing

The following section documents the releasing process. This document is primarily for internal use or by contributors to a project.

Cookbooks

This team uses Stove for releasing cookbooks. You can install Stove from Rubygems:

```
gem install stove
```

Stove exposes a `bake` command that automates a series of releasing tasks. Among other things, Stove will:

- Bump the version
- Prompt for a changelog
- Upload the cookbook to the community site
- Tag and push the changes to git
- Publish a release artifact to GitHub

Keeping in mind semantic versioning, pick the new version of the released cookbook. Then run the `bake` command:

> You will need to be a collaborator on the cookbook with release permission. If you do not have release permissions, please ask the cookbook maintainer to add you.

```
bake <x.y.z>
```

Announce new release on;

- Twitter
- Chef Mailing List

Gems

This team uses Bundler for releasing Ruby gems. Bundler is already installed on your system.

Keeping in mind semantic versioning, update the version in the `lib/version.rb` to the next appropriate version. Add and commit the changes to git:

```
git add lib/version.rb
git commit -m "Version bump to <x.y.z>"
```

Release the gem to Rubygems.org.

> You will need release permissions for the gem on RubyGems. If you do not have release permissions, please contact one of the gem owners to add you.

```
rake release
```

Announce new release on;

- Twitter
- Chef Mailing List