BP-13 - Time Based Release Plan

Status

Current state: *Accepted*

Discussion thread: http://mail-archives.apache.org/mod_mbox/bookkeeper-dev/201708.mbox/%3CCAO2yDyZbJwqATZb12vL9fN8VcTi7MxGconoiACxO4QNoqpYVzw%40mail.gmail.com%3E

JIRA: N/A

Released: Starting from 4.6.0

This proposal is to adopt Kafka's Time Based Release Plan.

Motivation

The benefits of moving to a time based release are:

1. A quicker feedback cycle and users can benefit from features shipped quicker
2. Predictability for contributors and users:
    a. Developers and reviewers can decide in advance what release they are aiming for with specific features.
    b. If a feature misses a release we have a good idea of when it will show up.
    c. Users know when to expect their features
3. Transparency - There will be a published cut-off date (AKA feature freeze) for the release and people will know about it in advance. Hopefully this will remove the contention around which features make it.
4. Quality - we've seen issues pop up in release candidates due to last-minute features that didn't have proper time to bake in. More time between feature freeze and release will let us test more, document more and resolve more issues.

Since nothing is ever perfect, there will be some downsides:

1. Most notably, features that miss the feature-freeze date for a release will have to wait few month for the next release. Features will reach users faster overall as per benefit #1, but individual features that just miss the cut will lose out
2. More releases a year mean that being a committer is more work - release management is still some headache and we'll have more of those. Hopefully we'll get better at it. Also, the committer list is growing and hopefully it will be less than once-a-year effort for each committer.
3. For users, figuring out which release to use and having frequent new releases to upgrade to may be a bit confusing.
4. Frequent releases mean we need to do bugfix releases for older branches. Right now we only do bugfix releases to latest release.

We decided to experiment with time based releases and see if the benefits for us as a community exceed the drawbacks. We will regularly iterate to improve our release process with the goal of having a community of happy developers and users as well as regular high-quality releases.

How will Apache BookKeeper development process look like?

At this stage we are planning to make a release **every 3 month**.

- **A month before the release date**, the release manager will cut branches and also publish (preferably on the wiki) a list of features that will be included in the release (these will typically be BPs, but not always).
- We will leave **another week** for "minor" features to get in (see below for definitions), but at this point we will start efforts to stabilize the release branch and contribute mostly tests and fixes.
- **Two weeks after branch cutting**, we will announce code-freeze and start rolling out RCs, after which only fixes for blocking bugs will be merged.

Definitions

- **Major Feature** - Feature that takes more than 2 weeks to develop and stabilize. Requires more than one PR to complete the feature.
- **Minor Feature** - Feature that takes less than 2 weeks to develop and stabilize. Requires mostly one PR to complete the feature.
- **Stabilization** - This phase would involve writing incremental system tests for features and fixing any bugs identified in the checked in feature.
- **Feature Freeze** - Major features should have only stabilization remaining. Minor features should have a PR in progress that can be checked in by a week. A release branch will be cut at this point.
- **Code Freeze** - Development is stopped. Blocker bugs will be fixed after code freeze. First RC will be created at this point.

For time based release, we will strictly ensure that a release happens on a given date. For example, in the first release, we will decide to have a release by middle of October and we will stick to it. We will drop features that we think will not make it into the release at the time of feature freeze and also avoid taking on new features into the release branch. Trunk development can continue as usual and those features will be in the following release. Ideally, we would have started stabilization around this time. About two weeks before the release date, we would call for code freeze and code checkins will be allowed only if any blocker bugs are identified. In a rare scenario, we could end up with a feature that passed the feature freeze bar but still fails to complete on time. Such features will also be dropped from the release at the end to meet the release deadline.

What happens if features don’t complete?

Features tend to be of different complexity. Some of them can be implemented within a single release while some span multiple releases. With time based release, we would need to ensure that ongoing features do not affect the stability of the release. There are couple of options –

1. **Ensure that every feature is broken down into testable units and only testable units get checked in.** This means that good set of unit test and system tests are written for sub tasks before they are checked in. This will ensure that trunk is in a relatively stable state to release at any point of time.
2. **Use feature branches.** Create branches from trunk that are focused on a specific feature. The feature developers ensure that the branch is in sync with trunk from time to time. Once there is high level of confidence on the stability, it can be merged into trunk. This approach has the additional overhead of branching and performing merges from time to time.

In practice, the right approach would be a mix of both 'a' and 'b'. The feature developers need to make this call depending on the complexity of the feature.

What are the gaps that we need to focus on?

There are some gaps in the development process and testing infrastructure that we need to close to ensure that we can successfully do a time based release. This is going to be a long term effort but will simplify time based releases as we make more progress in closing them.

1. **Testing coverage with check ins.** Code reviews in Apache BookKeeper need to ensure we have good unit test coverage and enforce system tests be written along with the check ins (when applicable). Testing should not be at the end. We should also encourage documentation be accompanied with check ins where applicable.
2. **Compatibility testing.** Although we have basic backward compatibility testing, it is not a very good framework. We need to come up with a better solution during testing.
3. **Documentation coverage with check ins**. Code reviews in Apache BookKeeper need to ensure we have good documentation coverage with check ins. Otherwise we will end up spending too much time on documentation at the end of each release.

What Is Our EOL Policy?

Currently Apache BookKeeper doesn't have a good story about EOL policy. We kept almost all the releases.

If we switched to Time based release plan, we will have more releases. We need to define an EOL Policy. Given 4 releases a year and the fact that no one upgrades three times a year, we propose making sure (by testing!) that rolling upgrade can be done from each release in the past year (i.e. last 4 releases) to the latest version.

We will also attempt, as a community to do bugfix releases as needed for the last 4 releases.

Who Manages The Releases?

As usual, a committer shall volunteer. If no committer volunteers, we'll cancel a release due to lack of interest.

What About Version Numbers?

Currently the Apache BookKeeper version number is comprised with 3 components: *major.minor.bug*

Feature releases will be a minor release by default (e.g. 4.4.0 => 4.5.0) - unless

- *We break compatibility (i.e. remove deprecated public methods after a reasonable period), in which case we bump major version (e.g. 4.4 => 5.0)*
- *We do something totally amazing (e.g. removing zookeeper dependency) and decide to bump major version*

Schedule

The proposed schedule for Apache BookKeeper is shown below.

- August 2017 - November 2017
- November 2017 - February 2018
- February 2018 - May 2018
- May 2018 - August 2018