Flutter Batch Release One Pager

SUMMARY

One pager to consolidate the batch release strategy.

Author: Chun-Heng Tai (chunhtai)
Go Link: flutter.dev/go/batch-release-one-pager
Created: 9/2025 / **Last updated:** 9/2025

ldap	LGTM
Kate Lovett	✅
Stuart Morgan	✅
Justin McCandless	✅

BACKGROUND

This is a one pager for batch release strategy from flutter.dev/go/package-release-strategy. This is an opinionated design to document the options we will take to implement the strategy.

Audience

Flutter TLs
First-party package owners and contributors

DESIGN

There will be a `release` branch apart from the `main` branch for release bot to

release from.

A cron job (through github action) that periodically scans the packages in the `main` branch for unreleased commits. The job then creates a release PR to the `release` branch with the changelog and version bump resulting from parsing the breadcrumb of those commits from the `main` branch.

Once the release PR is reviewed and merged, another release bot will check out the `release` branch and do a release. Once it is done, the release bot will create another PR with a `post-release-<package>` label to sync the `release` branch back to the `main` branch.

Once this post-release PR is reviewed and merged, a release cycle is completed.

Race Condition

Option 3 from First-Party Package Release Strategy
We use a `release` branch to release from. When a bot is creating a release PR, the change targets the `release` branch. Any subsequent merge into main will not affect the release branch.

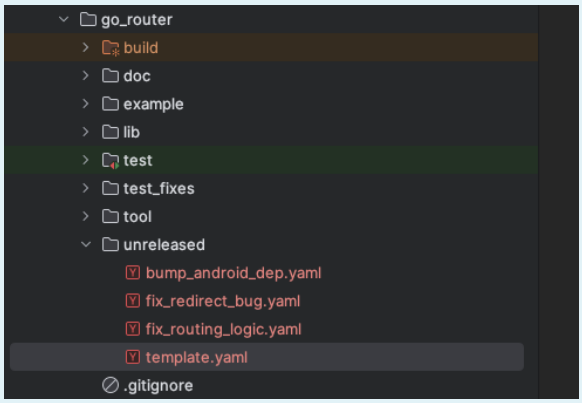
For reverts of unreleased commits, the contributors can revert the commits as straight reverts. The package owners have to postpone any on-going release by closing the release PR (if there is one) and retrigger it after the reverts land.

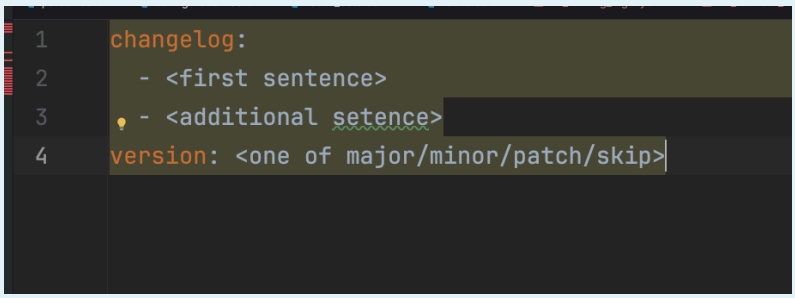
For reverts of released commits (one that has merged to the `release` branch), they should be treated as a new change with proper breadcrumb.

BreadCrumb

Option 3 from First-Party Package Release Strategy
New Change log and version bump will be in its own file as part of the PR, and each PR should create their own file. These files are stored under an `unreleased` folder in the package root.

An example:





The repo tool will be updated to create a separate file for change log and version bump information as part of the PR for the opted-in packages. For contributors who followed the recommended workflow, there will be no change to their workflow.

When the cron job (github action) creates a release, it reads through the unreleased files, generates updates to the real `changelog.md` and version bump in `pubspec.yaml`, and finally deletes these unreleased files in the release PRs.

Multi-Package Update

For the package that opts into the batch release, contributors that author mulit-package updates need to follow the breadcrumb rule and create changelog yaml files in the `unreleased` folder for those packages.

Others

- Package owner can also manually trigger github action to do one-time release
- A check run will flag PRs that touches `changelog.md` and version if they don't have corresponding `post-release-<package>` label
- Contributors can also add `post-release-<package>` if they really need to touch the changelog file and version for emergency situations.