Release Process

There are two types of release for the `go.opentelemetry.io/contrib` repo and submodules.

1. **Case 1** A release due to changes independent of the `go.opentelemetry.io/otel` module, e.g. perhaps a critical bug fix in one of the contrib modules.
2. **Case 2** A release due to a breaking API change in `go.opentelemetry.io/otel` which all modules in this repo depend on.

Pre-Release

Update go.mod for submodules to depend on the upcoming new release of the module in this repo, `go.opentelemetry.io/contrib`. Decide on the next version of the semantic tag to apply to the contrib module based on whether you fall into Case 1 or Case 2.

Case 1

If the changes are all internal to this repo, then the new tag will most often be a patch or minor version upgrade to the existing tag on this module. Let's call this `<new_contrib_tag>`.

Case 2

If a new release is required due to breaking changes in `go.opentelemetry.io/otel`, then the new semantic tag for this repo should be bumped to match the `go.opentelemetry.io/otel` new tag. Let's call this `<new_otel_tag>`. The script checks that `go.opentelemetry.io/otel@v<new_otel_tag>` is a valid tag, so you need to wait until that tag has been pushed in the main repo.

In nearly all cases, `<new_contrib_tag>` should be the same as `<new_otel_tag>`.

1. Run `pre_release.sh` script to create a branch `pre_release_<new_contrib_tag>`. The script will also run `go mod tidy` and `make ci`.
   - **Case 1** `./pre_release.sh -t <new_contrib_tag>`
   - **Case 2** `./pre_release.sh -o <new_otel_tag> [-t <new_contrib_tag>]`
2. If you used `-o <new_otel_tag>` to rewrite the modules to depend on a new version of `go.opentelemetry.io/otel`, there will likely be breaking changes that require fixes to the files in this `contrib` repo. Make the appropriate fixes to address any API breaks and run through the

   ~~~
   ```
   git commit -m "fixes due to API changes"
   make precommit
   ```
   ~~~

   cycle until everything passes

3. Push the changes to upstream.

   ```
   git diff main
   git push
   ```

4. Create a PR on github and merge the PR once approved.

Tag

Now create a `<new_contrib_tag>` on the commit hash of the changes made in pre-release step,

1. Run the tag.sh script.

   ```
   ./tag.sh <new_contrib_tag> <commit-hash>
   ```

2. Push tags upstream. Make sure you push upstream for all the sub-module tags as well.

   ```
   git push upstream <new_contrib_tag>
   git push upstream <submodules-path/new_contrib_tag>
   ...
   ```

Release

Now create a release for the new `<new_contrib_tag>` on github. The release body should include all the release notes in the Changelog for this release. Additionally, the `tag.sh` script generates commit logs since last release which can be used to suppliment the release notes.