How-to Commit

Commits applicable to multiple versions are atomically pushed forward merges.

The fix lands on the oldest release branch and is then forward-merges it into each newer branch using an ours merge to record branch lineage and amends that merge commit to include the branch-appropriate patch.

This keeps a clean, traceable history and a single logical unit of work per ticket per branch, while preventing unintended diffs from being pulled forward automatically.

Git branch based Contribution

How to commit and merging git-based contributions.

For example, a hypothetical `CASSANDRA-12345` ticket is a bug fix that requires different code for cassandra-4.0, cassandra-4.1, cassandra-5.0 and trunk. The contributor supplied git fork+branches `12345/4.0`, `12345/4.1`, `12345/5.0` and `12345/trunk`.

On cassandra-4.0

```
git cherry-pick <sha-of-4.0-commit>
ant realclean && ant jar # rebuild to make sure code compiles
```

On cassandra-4.1

```
git merge cassandra-4.0 -s ours --log
git cherry-pick -n <sha-of-4.1-commit>
ant realclean && ant jar # rebuild to make sure code compiles
git commit --amend # this will squash the 4.1 applied patch into the forward merge commit
```

On cassandra-5.0

```
git merge cassandra-4.1 -s ours --log
git cherry-pick -n <sha-of-5.0-commit>
ant realclean && ant jar check # rebuild to make sure code compiles
git commit --amend # this will squash the 5.0 applied patch into the forward merge commit
```

On trunk

```
git merge cassandra-5.0 -s ours --log
git cherry-pick -n <sha-of-trunk-commit>
ant realclean && ant jar check # rebuild to make sure code compiles
git commit --amend # this will squash the trunk applied patch into the forward merge commit
```

To Push

```
git push origin cassandra-4.0 cassandra-4.1 cassandra-5.0 trunk --atomic -n # dryrun check
git push origin cassandra-4.0 cassandra-4.1 cassandra-5.0 trunk --atomic
```

Contributions only for release branches

If the patch is for an older branch, and doesn't impact later branches (such as trunk), we still need to merge up and atomic push.

On cassandra-4.0

```
git cherry-pick <sha-of-4.0-commit>
ant realclean && ant jar # rebuild to make sure code compiles
```

On cassandra-4.1

```
git merge cassandra-4.0 -s ours --log
ant realclean && ant jar # rebuild to make sure code compiles
```

On cassandra-5.0

```
git merge cassandra-4.1 -s ours --log
ant realclean && ant jar check # rebuild to make sure code compiles
```

On trunk

```
git merge cassandra-4.1 -s ours --log
ant realclean && ant jar check # rebuild to make sure code compiles
```

To Push

```
git push origin cassandra-4.0 cassandra-4.1 trunk --atomic -n # dryrun check
git push origin cassandra-4.0 cassandra-4.1 trunk --atomic
```

Patch based Contribution

How to commit and merging patch-based contributions.

For example, a hypothetical `CASSANDRA-12345` ticket is a bug fix that requires different code for cassandra-4.0, cassandra-4.1, cassandra-5.0 and trunk. The contributor supplied provided the patch for the root branch `12345-4.0.patch`, and patches for the remaining branches `12345-4.1.patch`, `12345-5.0.patch` and `12345-trunk.patch`.

On cassandra-4.0

```
git am -3 12345-4.0.patch
ant realclean && ant jar # rebuild to make sure code compiles
git commit --amend # Notice this will squash the 4.0 applied patch into the forward merge commit
```

On cassandra-4.1

```
git merge cassandra-4.0 -s ours --log
git apply -3 12345-4.1.patch
ant realclean && ant jar # rebuild to make sure code compiles
git commit --amend # this will squash the 4.1 applied patch into the forward merge commit
```

On cassandra-5.0

```
git merge cassandra-4.1 -s ours --log
git apply -3 12345-5.0.patch
ant realclean && ant jar check # rebuild to make sure code compiles
git commit --amend # this will squash the 4.1 applied patch into the forward merge commit
```

On trunk

```
git merge cassandra-5.0 -s ours --log
git apply -3 12345-trunk.patch
ant realclean && ant jar check # rebuild to make sure code compiles
git commit --amend # this will squash the trunk applied patch into the forward merge commit
```

To Push

```
git push origin cassandra-4.0 cassandra-4.1 cassandra-5.0 trunk --atomic -n # dryrun check
git push origin cassandra-4.0 cassandra-4.1 cassandra-5.0 trunk --atomic
```

Commit Message

Tip
The commit message is to be in the format:

```
<One sentence description, usually Jira title or CHANGES.txt summary>

<Optional lengthier description>

 patch by <Authors>; reviewed by <Reviewers> for CASSANDRA-#####


Co-authored-by: Name1 <email1>
Co-authored-by: Name2 <email2>
```

This format is used by the contribulyze pages.

Tips

Tip
Notes on git flags: `-3` flag to am and apply will instruct git to perform a 3-way merge for you. If a conflict is detected, you can either resolve it manually or invoke git mergetool - for both am and apply.

`--atomic` flag to git push does the obvious thing: pushes all or nothing. Without the flag, the command is equivalent to running git push once per each branch. This is nifty in case a race condition happens - you won't push half the branches, blocking other committers' progress while you are resolving the issue.

Tip
The fastest way to get a patch from someone's commit in a branch on GH - if you don't have their repo in remotes - is to append .patch to the commit url, e.g. `curl -O https://github.com/apache/cassandra/commit/7374e9b5ab08c1f1e612bf72293ea14c959b0c3c.patch`

Tip
`git format-patch -1 <sha-of-X.X-commit> ; git apply -3 <sha-of-X.X-commit>.patch` can be used in place of the `git cherry-pick -n <sha-of-X.X-commit>` steps.