HowToCommit

Guide for Hive Committers

- Guide for Hive Committers
  - New committers
  - Review
  - Reject
  - PreCommit runs, and committing patches
  - Commit
    - Committing Documentation
  - Backporting commits to previous branches
  - Dialog

This page contains guidelines for committers of the Apache Hive project. (If you're currently a contributor, and are interested in how we add new committers, read BecomingACommitter)

New committers

New committers are encouraged to first read Apache's generic committer documentation:

- Apache New Committer Guide
- Apache Committer FAQ

The first act of a new core committer is typically to add their name to the credits page. This requires changing the XML source in http://svn.apache.org/repos/asf/hive/site/author/src/documentation/content/xdocs/credits.xml. Once done, update the Hive website as described in the Documentation section below.

Review

Hive committers should, as often as possible, attempt to review patches submitted by others. Ideally every submitted patch will get reviewed by a committer within a few days. If a committer reviews a patch they've not authored, and believe it to be of sufficient quality, then they can commit the patch, otherwise the patch should be cancelled with a clear explanation for why it was rejected.

The list of submitted patches is in the Hive Patches. This is ordered by time of last modification. Committers should scan the list from top-to-bottom, looking for patches that they feel qualified to review and possibly commit.

Hive committers may not +1 their own patches, i.e. you are allowed to commit your own patch only if the patch first receives a +1 vote from another committer. In the past this rule has typically been ignored when making small changes to the website (e.g. adding a new committer to the credits page), but you should follow the standard process for anything else.

Reject

Patches should be rejected which do not adhere to the guidelines in HowToContribute. Committers should always be polite to contributors and try to instruct and encourage them to contribute better patches. If a committer wishes to improve an unacceptable patch, then it should first be rejected, and a new patch should be attached by the committer for review.

PreCommit runs, and committing patches

1. Run Pre-Commit tests on a patch before committing.
2. If the test run is clean (and there's a +1 from a committer), the patch can be committed.
3. Test runs may not be clean due to issues in the patch itself, or due to flaky tests. These issues must be fixed and patch should not be committed until the run is clean.

If a commit introduces new test failures, the preferred process is to revert the patch, rather than opening a new JIRA to fix the new failures.

Commit

When you commit a patch, please:

1. Ensure that the patch has a +1 vote, and that 24 hours have elapsed since the first +1 vote was cast on JIRA. Note that this rule appears in the Hive Bylaws. Do not ignore it.
2. Include the Jira issue id in the commit message, along with a short description of the change and the name of the contributor. Be sure to get the issue id right, as this causes Jira to link to the change in Subversion (use the issue's "All" tab to see these).
   a. if contributor is you then add the following suffix to commit message "(<you>, reviewed by <reviewer>)". Example: "HIVE-123. Add awesomesauce to the optimizer. (jvs, reviewed by Ashutosh Chauhan)"
   b. if contributor is not you then add the following suffix to commit message "(<contributor> via <you>)". Example: "HIVE-123. Add awesomesauce to the optimizer. (Mike Brakestoner via jvs)"
      i. Additionally `--author="John Doe <john@doe.org>"` may be used to make the author the contributor
3. Don't forget to do 'svn add' on any new files, and 'svn delete' on any files that have been 'deleted' by the patch.

4. Resolve the issue as fixed, thanking the contributor. Always set the "Fix Version" at this point, but please only set a single fix version, the earliest release in which the change will appear. However, if a patch is backported to a point release (such as 1.0.2) then multiple fix versions should be set so that the automated release notes can list the Jira issue for the point release as well as the primary release.
5. Use the -E option to make sure that empty files are removed during the commit.

Committing Documentation

Hive's official documentation is authored using Forrest. To commit documentation changes you must have Forrest installed and the `forrest` executable on your `$PATH`. Note that the current version (0.8) doesn't work properly with Java 6, use Java 5 instead. Documentation is of two types:

1. End-user documentation, versioned with releases; and,
2. The website. This is maintained separately in subversion, republished as it is changed.

To commit end-user documentation changes to trunk or a branch, ask the user to submit only changes made to the *.xml files in `src/docs`. Apply that patch, run `ant docs` to generate the html, and then commit. End-user documentation is only published to the web when releases are made, as described in HowToRelease.

To commit changes to the website and re-publish them:

```
% svn co https://svn.apache.org/repos/asf/hive/site hive-site
% cd hive-site

# Make your changes in the author/src/documentation/content/xdocs subdirectory.
# Then run 'ant' to generate the new website. This will cause
# files to be updated/added in the publish/ subdirectory.
% ant

% svn status
M       author/src/documentation/content/xdocs/credits.xml
X       author/src/documentation/skins
M       publish/credits.html
M       publish/credits.pdf

# Inspect the modified/added files in the publish directory,
# and commit the changes once you are satisfied with them:

% svn commit -m "Add Bob to list of committers on credits page (cws)"
```

Changes committed to the website repository will be automatically published to the website using svnpubsub.

Backporting commits to previous branches

NOTE: the information in this section does not match the current practice, which is to re-apply the patch (or a backport of the patch) directly on the branch.

If a patch needs to be backported to previous branches, follow these steps.

1. Commit the changes to trunk and note down the revision number, say 4001. (Revision number is displayed as response to your *svn commit* command).
   2. Check out the desired branch and execute this command from the root directory.

   ```
   svn merge -r 4000:4001 https://svn.apache.org/repos/asf/hive/trunk .
   svn commit
   ```

Dialog

Committers should hang out in the #hive room on irc.freenode.net for real-time discussions. However any substantive discussion (as with any off-list project-related discussion) should be re-iterated in Jira or on the developer list.