HowToCommitWithSvn

Guide for Hadoop Core Committers

This page contains Hadoop Core-specific guidelines for committers.

New committers

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

- Git access to Apache codebases
- Apache Committer FAQ
- Apache New Committer Guide - Some of this information applies only to projects that use Subversion.

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/hadoop/common/site/main/author/src/documentation/content/xdocs/who.xml. Once done, update the Hadoop website as described here.

Review

Hadoop 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 Hadoop Review Queue. 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.

For non-trivial changes, it is best to get another committer to review your own patches before commit. Use "Submit Patch" like other contributors, and then wait for a "+1" from another committer before committing. Please also try to frequently review things in the patch queues:

- Hadoop Common Review Queue
- Hadoop HDFS Review Queue
- Hadoop MapReduce Review Queue
- Hadoop YARN Review Queue

Reject

Patches should be rejected which do not adhere to the guidelines in HowToContribute and to the CodeReviewChecklist. 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.

Commit

When you commit a patch, please:

1. Add an entry in CHANGES.txt, at the end of the appropriate section. This should include the JIRA issue ID, and the name of the contributor.
2. Include the JIRA issue id in the commit message, along with a short description of the change and the name of the contributor if it is not you. 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).
3. 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. **Special case**- when committing to a *non-mainline* branch (such as branch-0.22 or branch-0.23 ATM), please set fix-version to either 2.x.x or 3.x.x appropriately too.
4. Use the -E option to make sure that empty files are removed during the commit.

You can find a guidance on how to commit a patch to the Hadoop SVN repository in this video.

Committing Documentation

Hadoop'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/hadoop/common/site
cd site
ant
firefox publish/index.html # preview the changes
svn stat                   # check for new pages
svn add                    # add any new pages
svn commit
ssh people.apache.org
cd /www/hadoop.apache.org/common
svn up
```

Changes to website (*via svn up*) might take up to an hour to be reflected on Apache Hadoop site.

Backporting commits to previous branches

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/hadoop/common/trunk .
# Now resolve any merge conflicts.
# If major edits are needed, produce a new patch and upload it to the JIRA.
svn diff CHANGES.txt # get all JIRA numbers included in this merge
svn commit -m "merge <list all JIRA numbers here>"
```

Please be sure to include JIRA number(s) in the commit message for merge commits. Sometimes developers just put something like "merge -r 4000:4001" in the merge message, which fails to trigger the JIRA/Subversion integration, so the JIRA doesn't record the branch commit. It is important to link to the JIRA number, so that when looking at the JIRA it will be clear that this patch has been merged to this branch.

Patches that break HDFS, YARN and MapReduce

In general, the process flow is that Jenkins notices the checkin and automatically builds the new versions of the common libraries and pushes them to Nexus, the Apache Maven repository.

However, to speed up the process or if Jenkins is not working properly, developers can push builds manually to Nexus. To do so, they need to create a file in ~/.m2/settings.xml that looks like:

```
<settings>
  <servers>
    <!-- To publish a snapshot of some part of Maven -->
    <server>
      <id>apache.snapshots.https</id>
      <username> <!-- YOUR APACHE SVN USERNAME --> </username>
      <password> <!-- YOUR APACHE SVN PASSWORD --> </password>
    </server>
    <!-- To publish a website of some part of Maven -->
    <server>
      <id>apache.website</id>
      <username> <!-- YOUR APACHE SSH USERNAME --> </username>
      <filePermissions>664</filePermissions>
      <directoryPermissions>775</directoryPermissions>
    </server>
    <!-- To stage a release of some part of Maven -->
    <server>
      <id>apache.releases.https</id>
      <username> <!-- YOUR APACHE SVN USERNAME --> </username>
      <password> <!-- YOUR APACHE SVN PASSWORD --> </password>
    </server>
    <!-- To stage a website of some part of Maven -->
    <server>
      <id>stagingSite</id>
      <!-- must match hard-coded repository identifier in site:stage-deploy -->
      <username> <!-- YOUR APACHE SSH USERNAME --> </username>
      <filePermissions>664</filePermissions>
      <directoryPermissions>775</directoryPermissions>
    </server>
  </servers>
</settings>
```

After you have committed the change to Common, do an "ant mvn-publish" to publish the new jars.

As a security note, since the settings.xml file contains your Apache svn password in the clear, I prefer to leave the settings file encrypted using gpg when I'm not using it. I also don't ever publish from a shared machine, but that is just me being paranoid. 🙂

Dialog

Committers should hang out in the #hadoop 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.