HowToReleaseWithSvnAndAnt

This page is prepared for Hadoop Core committers. You need committer rights to create a new Hadoop Core release.

DEPRECATED! This doc refers to the releases already end of life. For current releases, please see HowToRelease.

These instructions have been updated for Hadoop 0.20.x and 1.x releases. For earlier releases, check out an older revision of this page. For 0.21.0 and later many of the steps need to be done in turn for Common, HDFS, and MapReduce. For 0.23.x and 2.x releases, there is a new version of this page at HowToReleasePostMavenization.

Preparation

1. Bulk update Jira to unassign from this release all issues that are open non-blockers and send follow-up notification to the developer list that this was done.
2. If you have not already done so, update your @apache.org account via id.apache.org with your key; also add and commit your public key to the Hadoop repository KEYS, appending the output of the following commands:

   ```
   gpg --armor --fingerprint --list-sigs <keyid>
   gpg --armor --export <keyid>
   ```

   and publish your key at http://pgp.mit.edu/. Once you commit your changes, log into `people.apache.org` and pull updates to `/www/www.apache.org/dist/hadoop/core`. For more details on signing releases, see Signing Releases and Step-By-Step Guide to Mirroring Releases.
3. To deploy artifacts to the Apache Maven repository create `~/.m2/settings.xml`:

   ```
   <settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
                         http://maven.apache.org/xsd/settings-1.0.0.xsd">
     <servers>
       <server>
        <id>apache.staging.https</id>
        <username>Apache username</username>
        <password>Apache password</password>
       </server>
     </servers>
   </settings>
   ```

Branching

Skip this section if this is NOT the first release in a series (i.e. release X.Y.0).

1. Notify developers on the #hadoop IRC channel that you are about to branch a release.
2. Update `CHANGES.txt` to include the release version and date (use `Unreleased` for the date if it is unknown) and remove `Trunk (unreleased changes)`.
3. Commit these changes to trunk.

   ```
   svn commit -m "Preparing for release X.Y.Z"
   ```

4. Create a branch for the release series:

   ```
   svn copy https://svn.apache.org/repos/asf/hadoop/common/trunk \
   https://svn.apache.org/repos/asf/hadoop/common/branches/branch-X.Y -m "Branching for X.Y releases"
   ```

5. Update `CHANGES.txt` to add back in `Trunk (unreleased changes)`.
6. Update the default version in `build.xml` on trunk to X.Y+1.0-dev.

7. Update the `hadoop.version` number in `ivy/libraries.properties` on trunk to X.Y+1.0.
8. Commit these changes to trunk.

```
svn commit -m "Preparing for X.Y+1.0 development"
```

Updating Release Branch

These operations take place in the release branch.

1. Check out the branch with:

```
svn co https://svn.apache.org/repos/asf/hadoop/common/branches/branch-X.Y
```

2. Update `CHANGES.txt` to include the release version and date (this change must be committed to trunk and any intermediate branches between trunk and the branch being released).
3. Update `src/docs/releasenotes.html` with release notes for this release. You generate these with:

```
cd src/docs
jira.sh -s https://issues.apache.org/jira -u $user -p $pw \
   -a getIssueList --search \
   "project in (HADOOP,HDFS,MAPREDUCE) and fixVersion = '$vers' and (resolution = Fixed OR 'Target
Version/s' = '$vers') ORDER BY KEY" \
   | ./relnotes.py > $vers.html
```

   edit the releasenotes.html with the list of items from $vers.html.
4. Update the version number in `build.xml` to be *hadoop-X.Y.N-dev*, where *N* is one greater than the release being made.
5. Update the `hadoop.version` number in `ivy/libraries.properties` to be the same as the release being made.
6. Commit these changes.

```
svn commit -m "Preparing for release X.Y.Z"
```

7. If not already done, merge desired patches from trunk into the branch and commit these changes. You can find the revision numbers using `svn log CHANGES.txt` in the branch and in trunk.

```
cd branch-X.Y
svn merge -rR1:R2 ../trunk .
svn commit -m "Merge -r R1:R2 from trunk to X.Y branch. Fixes: HADOOP-A, HADOOP-B."
```

8. Tag the release candidate (R is the release candidate number, and starts from 0):

```
svn copy https://svn.apache.org/repos/asf/hadoop/common/branches/branch-X.Y \
https://svn.apache.org/repos/asf/hadoop/common/tags/release-X.Y.Z-rcR -m "Hadoop X.Y.Z-rcR release."
```

Build Requirements

To build an official release, you must:

1. Use a 64-bit Linux system, so that we can build native code for both 32- and 64-bit architectures.
2. Have a recent version of Eclipse installed, so that the eclipse plugin will build.
3. Have Xerces C 2.8.x installed.

HADOOP-6846 has some scripts that make it easier to build and smoke test a release for 0.21.0 and later.

Building

1. Build the release & run unit tests. This is captured in part in http://svn.apache.org/viewvc/hadoop/nightly/hudsonBuildHadoopRelease.sh?view=markup. The three parts of this command are intended to be run cumulatively:

```
## build 32-bit
export JAVA_HOME=/path/to/32bit/jdk
export CFLAGS=-m32
export CXXFLAGS=-m32
ant \
  -Dforrest.home=/usr/local/forrest \
  -Djava5.home=/usr/local/jdk1.5 \
  -Dfindbugs.home=/usr/local/findbugs \
  -Declipse.home=/usr/lib/eclipse \
  -Dxercescroot=/usr/local/xerces-c \
  -Dversion=X.Y.Z \
  -Dhadoop.version=X.Y.Z \
  -Dcompile.native=true \
  -Dcompile.c++=true \
  -Dlibhdfs=true \
  -Dlibrecordio=true \
  -Dtest.junit.output.format=xml \
  veryclean task-controller rpm deb \
     | tee build_log_dir/build32-X.Y.Z.log

## build 64-bit
export JAVA_HOME=/path/to/64bit/jdk
export CFLAGS=-m64
export CXXFLAGS=-m64
ant \
  -Dforrest.home=/usr/local/forrest \
  -Djava5.home=/usr/local/jdk1.5 \
  -Dfindbugs.home=/usr/local/findbugs \
  -Dversion=X.Y.Z \
  -Dhadoop.version=X.Y.Z \
  -Dcompile.native=true \
  -Dcompile.c++=true \
  -Dlibhdfs=true \
  -Dlibrecordio=true \
  tar rpm deb | tee build_log_dir/build64-X.Y.Z.log

## run tests (back in 32-bit mode)
export JAVA_HOME=/path/to/32bit/jdk
export CFLAGS=-m32
export CXXFLAGS=-m32
ant \
  -Dforrest.home=/usr/local/forrest \
  -Djava5.home=/usr/local/jdk1.5 \
  -Dfindbugs.home=/usr/local/findbugs \
  -Declipse.home=/usr/lib/eclipse \
  -Dxercescroot=/usr/local/xerces-c \
  -Dversion=X.Y.Z \
  -Dhadoop.version=X.Y.Z \
  -Dcompile.native=true \
  -Dcompile.c++=true \
  -Dlibhdfs=true \
  -Dlibrecordio=true \
  -Dtest.junit.output.format=xml \
  test test-c++-libhdfs | tee build_log_dir/build32tests-X.Y.Z.log
```

2. Check that release file looks ok - e.g. install it and run examples from tutorial.
3. Generate the checksums of the release file.

```
gpg --print-mds hadoop-X.Y.Z.tar.gz > hadoop-X.Y.Z.tar.gz.mds
```

4. Sign the release

```
gpg --armor --output hadoop-X.Y.Z.tar.gz.asc --detach-sig hadoop-X.Y.Z.tar.gz
```

5. Copy release files to a public place.

```
ssh people.apache.org mkdir public_html/hadoop-X.Y.Z-candidate-0
scp -p hadoop-X.Y.Z.tar.gz* people.apache.org:public_html/hadoop-X.Y.Z-candidate-0
```

6. Stage the release candidate to the maven repository:

```
ant \
  -Dforrest.home=/usr/local/forrest \
  -Djava5.home=/usr/local/jdk1.5 \
  -Dfindbugs.home=/usr/local/findbugs \
  -Dversion=X.Y.Z \
  -Dhadoop.version=X.Y.Z \
  -Drepo=staging \
  mvn-deploy
## Be ready to respond to the interactive request for your GPG pass-phrase, for signing the artifacts.
```

   For Hadoop-2.x use maven deploy

```
mvn -Psign deploy -DskipTests
## Be ready to respond to the interactive request for your GPG pass-phrase, for signing the artifacts.
```

7. Enter Nexus, and perform the following steps:
   a. Click on `Log In` in the upper right corner. Log in using your apache user name and password.
   b. In the left navigation pane, select `Staging Repositories`.
   c. Identify the release candidate you just pushed, by your user name (in parentheses as part of the "Repository" name) and the "Created On" date. Click on the check box to the left of your Repository name to select it. (If you accidentally click on the Repository name itself, another tab will pop open. If so, just close it.)
   d. Click the `Close` button above the Repository names. This makes your release candidate available at the Staging level.
   e. If you have previously staged an older release candidate with the same version number, and it is still showing in the Repository list, you must select and `Drop` the old one now.
   f. Confirm that your new release candidate is visible at https://repository.apache.org/content/groups/staging/org/apache/hadoop/hadoop-core/X.Y.Z/ (for 1.x and 0.20.x), or https://repository.apache.org/content/groups/staging/org/apache/hadoop/hadoop-common/X.Y.Z/ (for 0.22.x, 0.23.x, and 2.x), with correct file modification dates.
8. Call a release vote on common-dev at hadoop.apache.org.

Publishing

In 7 days if the release vote passes, the release may be published.

1. Tag the release:

```
svn move https://svn.apache.org/repos/asf/hadoop/common/tags/release-X.Y.Z-rcR \
https://svn.apache.org/repos/asf/hadoop/common/tags/release-X.Y.Z -m "Hadoop X.Y.Z release."
```

2. Copy release files to the distribution directory and make them writable by the hadoop group.

```
ssh people.apache.org
cp -pr public_html/hadoop-X.Y.Z-candidate-0 /www/www.apache.org/dist/hadoop/core/hadoop-X.Y.Z
cd /www/www.apache.org/dist/hadoop/core
chgrp -R hadoop hadoop-X.Y.Z
chmod -R g+w hadoop-X.Y.Z
```

3. The release directory usually contains just two releases, the most recent from two branches, with a link named 'stable' to the most recent recommended version.

```
ssh people.apache.org
cd /www/www.apache.org/dist/hadoop/core
rm -rf hadoop-A.B.C; rm stable
ln -s hadoop-A.B.D stable
```

4. In Nexus, promote the artifacts to 'Released' status by right-clicking the staged repository and select `Release`
5. Wait 24 hours for release to propagate to mirrors.
6. Prepare to edit the website.

```
svn co https://svn.apache.org/repos/asf/hadoop/common/site ~/hadoop-site
```

7. Update the documentation links in

```
 main/author/src/documentation/content/xdocs/site.xml
```

.

8. Update the release news in

```
 main/author/src/documentation/content/xdocs/releases.xml
```

.

9. Regenerate the site, review it, then commit it.

```
cd ~/hadoop-site/main
ant -Dforrest.home=/usr/local/forrest -Djava5.home=/usr/local/jdk1.5 update
svn commit -m "Updated site for release X.Y.Z."
```

10. It is not usually necessary to update the site front page (http://hadoop.apache.org), but if it is needed, update `main/author/src/documentation/content/xdocs/index.xml`, then do

```
cd ~/hadoop-site/main
ant -Dforrest.home=/usr/local/forrest -Djava5.home=/usr/local/jdk1.5 update
svn commit -m "Updated site front page for release X.Y.Z."
```

11. Publish the new release docs, by doing the following:

```
ssh people.apache.org
svn co --depth immediates https://svn.apache.org/repos/asf/hadoop/common/site/main/publish/docs/
cd docs
tar xzf /www/www.apache.org/dist/hadoop/core/hadoop-X.Y.Z/hadoop-X.Y.Z.tar.gz --wildcards '*/docs'
mv hadoop-X.Y.Z/docs rX.Y.Z
svn add rX.Y.Z
svn commit -m "Publishing docs for release X.Y.Z."
rm -r hadoop-X.Y.Z
```

12. If the `docs/current` and/or `docs/stable` links should be updated to the new release, do one or both of the following:

```
## update current
rm current
ln -s rX.Y.Z current
svn commit -m "Updating link to current docs."

## update stable
rm stable
ln -s rX.Y.Z stable
svn commit -m "Updating link to stable docs."
```

13. Generate the jdiff API data for the new release by, in the branch directory, running

```
ant -Dversion=X.Y.Z api-xml
```

then commit the new XML file generated in lib/jdiff to both trunk and to the branch (and any intermediate branches between trunk and the branch being released).

```
svn add lib/jdiff/hadoop_X.Y.Z.xml
svn commit -m "JDiff output for release X.Y.Z"
```

14. Update the `jdiff.stable` value in the X.Y+1 branch's `build.xml` (which may be trunk) to be the published release (ie. X.Y.Z).
15. Send announcements to the user and developer lists once the site changes are visible.
16. In Jira, ensure that only issues in the "Fixed" state have a "Fix Version" set to release X.Y.Z.
17. In Jira, "release" the version. Visit the "Administer Project" page, then the "Manage versions" page. You need to have the "Admin" role in Hadoop Core's Jira for this step and the next.
18. In Jira, close issues resolved in the release. Disable mail notifications for this bulk change.

See Also

- Apache Releases FAQ