How To Release Hadoop-Thirdparty

The Hadoop-thirdparty artifacts have a different release number, different repo. But apart from that, it is mostly the same as the main Hadoop releases.

Preparation

1. If you have not already done so, append your code signing key to the KEYS file. Once you commit your changes, they will automatically be propagated to the website. Also upload your key to a public key server if you haven't. End users use the KEYS file (along with the web of trust) to validate that releases were done by an Apache committer. For more details on signing releases, see Signing Releases and Step-By-Step Guide to Mirroring Releases.
2. Make sure to update the various LICENSE and NOTICE files per Apache policy.
3. Bulk update JIRA to unassign from this release all issues that are open non-blockers.
4. Send follow-up notification to the developer list that this was done.
5. 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

When releasing Hadoop X.Y.Z, the following branching changes are required. Note that a release can match more than one of the following if-conditions. For a major release, one needs to make the changes for minor and point releases as well. Similarly, a new minor release is also a new point release.

1. If this is a new major release (i.e., Y = 0 and Z = 0)
    a. Create a new branch (branch-X) for all releases in this major release.
    b. Update the version on trunk to (X+1).0.0-SNAPSHOT

        ```
        mvn versions:set -DnewVersion=(X+1).0.0-SNAPSHOT
        ```

    c. Commit the version change to trunk.

        ```
        git commit -a -m "Preparing for (X+1).0.0 development"
        ```

2. If this is a new minor release (i.e., Z = 0)
    a. Create a new branch (branch-X.Y) for all releases in this minor release.
    b. Update the version on branch-X to X.(Y+1).0-SNAPSHOT

        ```
        mvn versions:set -DnewVersion=X.(Y+1).0-SNAPSHOT
        ```

    c. Commit the version change to branch-X.

        ```
        git commit -a -m "Preparing for X.(Y+1).0 development"
        ```

3. If this is a new point release (i.e., always)
    a. Create a new branch (branch-X.Y.Z) for this release.
    b. Update the version on branch-X.Y to X.Y.(Z+1)-SNAPSHOT

        ```
        mvn versions:set -DnewVersion=X.Y.(Z+1)-SNAPSHOT
        ```

    c. Commit the version change to branch-X.Y.

        ```
        git commit -a -m "Preparing for X.Y.(Z+1) development"
        ```

4. Release branch (branch-X.Y.Z) updates:
    a. Update the version on branch-X.Y.Z TO X.Y.Z

        ```
        mvn versions:set -DnewVersion=X.Y.Z
        ```

Now, for any branches in {trunk, branch-X, branch-X.Y, branch-X.Y.Z} that have changed, push them to the remote repo taking care of any conflicts.

```
git push <remote> <branch>
```

Creating the release candidate (X.Y.Z-RC<N>)

These steps need to be performed to create the _N_th RC for X.Y.Z, where *N* starts from 0.

1. In JIRA, ensure that only issues in the "Fixed" state have a "Fix Version" set to release X.Y.Z.
2. Verify that $HOME/.gpg defaults to the key listed in the KEYS file.
3. For the Apache release, a machine capable of running Docker- and Internet- capable, build the release candidate with `create-release`. Unless the `--logdir` is given, logs will be in the `patchprocess/` directory. Artifacts will be in the target/artifacts NOTE: This will take quite a while, since it will download and build the entire source tree, including documentation and native components, from scratch to avoid maven repository caching issues hiding issues with the source release.

   ```
   dev-support/bin/create-release --asfrelease --docker --dockercache
   ```

4. While it should fail `create-release` if there are issues, doublecheck the rat log to find and fix any potential licensing issues.

   ```
   grep 'Rat check' patchprocess/mvn_apache_rat.log
   ```

5. Check that release files look ok - e.g. install it somewhere fresh and run examples from tutorial, do a fresh build, read the release notes looking for WARNINGs, etc.
6. Set environment variable version for later steps. `export version=X.Y.Z-RCN`
7. Tag the release candidate:

   ```
   git tag -s release-$version -m "Release candidate - $version"
   ```

8. Push branch-X.Y.Z and the newly created tag to the remote repo.
9. Deploy the maven artifacts, on your personal computer. Please be sure you have completed the prerequisite step of preparing the `settings.xml` file before the deployment. You might want to do this in private and clear your history file as your gpg-passphrase is in clear text.

   ```
   mvn deploy -Psign,dist -DskipTests -DskipShade
   ```

10. Copy release files to a public place and ensure they are readable. Note that `home.apache.org` only supports SFTP, so this may be easier with a graphical SFTP client like Nautilus, Konqueror, etc.

    ```
    sftp home.apache.org
    > cd public_html
    > mkdir hadoop-thirdparty-${version}
    > put -r /home/hadoop/hadoop-thirdparty-${version}
    ....
    > bye
    ```

11. Log into Nexus, select "`Staging` Repositories" from the left navigation pane, select the check-box against the specific hadoop repository, and `close` the release. At the time of deploying the maven artifacts, if there are different hadoop items of this release candidate at “Staging Repositories”, drop the stale RC first.
12. Call a release vote on common-dev at hadoop.apache.org. It's usually a good idea to start the release vote on Monday so that people will have a chance to verify the release candidate during the week. Example
13. If the release candidate contains a serious issue, withdraw the vote, make necessary changes, and repeat this process.
14. If non-trivial changes are committed to the release branch, ensure the commits are present in the upstream branches.

Publishing

In 5 days if the release vote passes, the release may be published.

1. In JIRA, "release" the version, setting the date to the end-of-vote date. Visit the "Administer Project" page, then the "Manage versions" page. You need to have the "Admin" role in HADOOP, HDFS, MAPREDUCE, and YARN.
2. Set environment variable version for later steps. `export version=X.Y.Z`
3. Tag the release. Do it from the release branch and push the created tag to the remote repository:

   ```
   git tag -s rel/release-${version} -m "Hadoop Thirdparty ${version} release"
   git push origin rel/release-${version}
   ```

4. Copy release files to the distribution directory
   a. Check out the corresponding svn repo if need be

      ```
      svn co https://dist.apache.org/repos/dist/release/hadoop/thirdparty/ hadoop-thirdparty-dist
      ```

   b. Copy the release files to hadoop-thirdparty-dist/thirdparty-${version}
   c. Add the files
      svn add thirdparty-$version}
   d. Commit the changes (it requires a PMC privilege)

      ```
      svn ci -m "Publishing the bits for Hadoop Thirdparty release ${version}"
      ```

5. Update upstream branches to make them aware of this new release:
   a. Copy and commit the CHANGES.md and RELEASENOTES.md:

```
git commit -a -m "Make upstream aware of ${version} release."
```

6. In Nexus
    a. effect the release of artifacts by selecting the staged repository and then clicking `Release`
    b. If there were multiple RCs, simply drop the staging repositories corresponding to failed RCs.
7. Wait 24 hours for release to propagate to mirrors.