HowToRelease

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

Hadoop Version Warning

This page assumes you are releasing from the master branch, and thus omits the use of Maven profiles to determine which version of Hadoop you are building against. If you are releasing from branch-1, you will need to add `-Phadoop-2` to most of your Maven commands.

Storage API Release

The Hive projects has two products that are released separately:

1. Storage-API – the vectorization and predicate push down classes
2. Hive – the rest of Hive

Most Hive releases will require a new storage-api release and the storage-api currently releases faster than Hive, so has higher version numbers.

Storage API Prepare Master Branch

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

1. Check out the master branch

```
git checkout master
```

2. Increment the value of the `version` property in the storage-api/pom.xml file. For example, if the current value is 2.5.0`-SNAPSHOT`, the new value should be 2.6.0`-SNAPSHOT`. Please note that the `SNAPSHOT` suffix is required in order to indicate that this is an unreleased development branch.
3. Update the storage-api.version property in the root pom.xml and standalone-metastore/pom.xml to the new value from the step above.
4. Verify that the build is working with changes.
5. Commit these changes to master with a comment "Preparing for storage-api X.Y+1.0 development".

Storage API 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 #hive IRC channel and dev@hive mailing lists that you are about to branch a release.
2. Create a branch for the release series:

```
git checkout -b storage-branch-X.Y origin/master
```

3. Update the `version` property value in the storage-api/pom.xml file. You should remove the `SNAPSHOT` suffix and set `version` equal to `X.Y.Z` w here `Z` is the point release number in this release series (0 for the first one, in which case this step is a no-op since you already did this above when creating the branch). Use Maven's Versions plugin to do this as follows:
4. Verify that the build is working with changes.
5. Commit these changes with a comment "Preparing for storage-api X.Y.Z release".
6. Tag the release candidate (`R` is the release candidate number, and also starts from 0):

```
git commit -a -m "Preparing for storage-api X.Y.Z release"
git push -u origin storage-branch-X.Y
git tag -a storage-release-X.Y.Z-rcR -m "Hive Storage API X.Y.Z-rcR release."
git push origin storage-release-X.Y.Z-rcR
```

Making Storage API Release Artifacts

1. Make sure your release notes have been updated for any new commits, and go through the previous steps if necessary.
2. Create and publish the tag:

```
git tag storage-release-X.Y.Z-rcR -m "Hive Storage API X.Y.Z-rcR release."
git push origin storage-release-X.Y.Z-rcR
```

3. Build the release (binary and source versions) after running unit tests. Manually create the sha file.

```
% wget https://github.com/apache/hive/archive/storage-release-X.Y.Z-rcR.tar.gz
% tar xzvf storage-release-X.Y.Z-rcR.tar.gz
% mv storage-release-X.Y.Z-rcR/storage-api hive-storage-X.Y.Z
% tar czvf hive-storage-X.Y.Z-rcR.tar.gz hive-storage-X.Y.Z
% shasum -a 256 hive-storage-X.Y.Z-rcR.tar.gz > hive-storage-X.Y.Z-rcR.tar.gz.sha256
```

4. Setup your PGP keys for signing the release, if you don't have them already.
   a. See https://www.apache.org/dev/release-signing.html, https://www.apache.org/dev/openpgp.html.
5. Sign the release (see Step-By-Step Guide to Mirroring Releases for more information).

```
% gpg --armor --detach-sig hive-storage-X.Y.Z-rcR.tar.gz
```

6. Check the signatures.

```
% shasum -c hive-storage-X.Y.Z-rcR.tar.gz.sha256
hive-storage-X.Y.Z-rcR.tar.gz: OK

% gpg hive-storage-X.Y.Z-rcR.tar.gz.asc
gpg: assuming signed data in `hive-storage-X.Y.Z-rcR.tar.gz'
gpg: Signature made Fri Apr 28 12:50:03 2017 PDT using RSA key ID YOUR-KEY-ID
gpg: Good signature from "Your Name <YOUR-APACHE-ID@apache.org>"
```

7. Copy release files to a public place.

```
% sftp YOUR-APACHE-ID@home.apache.org

sftp> cd public_html
sftp> mkdir hive-storage-X.Y.Z
sftp> cd hive-storage-X.Y.Z
sftp> put hive-storage-X.Y.Z-rcR.tar.gz*
sftp> quit
```

8. Send email to dev@hive.apache.org calling the vote.

Publishing the Storage API Artifacts

1. After the release vote passes, push the artifacts to Nexus. (If you get an error `gpg: signing failed: Inappropriate ioctl for device` trying doing `export GPG_TTY=$(tty)` .)

```
% git checkout storage-release-X.Y.Z-rcR
% cd storage-api
% mvn -Papache-release -DskipTests clean deploy
```

2. Login to Nexus and close the repository. Mark the repository as released.
3. Create the final tag (be very careful, tags in "rel/" are not changeable).

```
% git checkout storage-release-X.Y.Z-rcR
% git tag -s rel/storage-release-X.Y.Z -m "Hive Storage API X.Y.Z"
% git push origin rel/storage-release-X.Y.Z
```

4. Add the artifacts to Hive's dist area. There might be a problem with the size of the artifact. INFRA-23055 - Getting issue details... STATUS solved the issue.

```
% svn checkout https://dist.apache.org/repos/dist/release/hive hive-dist
% cd hive-dist
% mkdir hive-storage-X.Y.Z
% cd hive-storage-X.Y.Z
% wget https://home.apache.org/~YOUR-APACHE-ID/hive-storage-X.Y.Z/hive-storage-X.Y.Z-rcR.tar.gz{,.sha256,.asc}
% svn add .
% svn commit
```

Preparing Branch for further development

1. Edit storage-api/pom.xml to change version to X.Y.Z+1-SNAPSHOT.
2. Edit pom.xml to change storage-api.version to X.Y.Z+1-SNAPSHOT.
3. Commit the changes back

```
% git commit -a -s -m 'Preparing for development post-X.Y.Z.'
% git push origin storage-branch-X.Y
```

Cleaning Up Storage API Artifacts

1. Delete the storage-release-X.Y.Z-rcR tags.
2. Delete the artifacts from home.apache.org.

Hive Release

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. There are two kinds of JIRAs that need to be taken care of:
    a. Unresolved JIRAs with Target Version/s or Fix Version/s (legacy) set to the release in question.
    b. Resolved/closed(!) JIRAs with Target Version/s, but not Fix Version/s set to the release in question (e.g. a JIRA targets 2.0.0 and 1.3.0, but was only committed to 2.0.0).
2. Run 'mvn clean apache-rat:check' and examine the generated report for any files, especially .java files which should all have Apache license headers. Note also, that each individual component will have a rat.txt inside it when you run this – be sure to check ql/target/rat.txt, for example. Add the license header to any file that is missing it (open a jira and submit a patch).
3. Update copyright date in NOTICE. If any components mentioned in them have updated versions, you would need to update the copyright dates for those. (Thejas comment: It sounds like entries are needed in NOTICE only if the license requires such attribution. See https://www.apache.org/legal/src-headers.html#notice.)

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 #hive IRC channel and dev@hive mailing lists that you are about to branch a release.
2. Create a branch for the release series:

```
git checkout -b branch-X.Y origin/master
git push -u origin branch-X.Y
```

3. Increment the value of the `version` property in all pom.xml files. For example, if the current value is `0.7.0-SNAPSHOT`, the new value should be `0.8.0-SNAPSHOT`. Please note that the `SNAPSHOT` suffix is required in order to indicate that this is an unreleased development branch. This can be accomplished with a single command using Maven's Versions plugin as follows:

```
mvn versions:set -DnewVersion=X.Y.0-SNAPSHOT -DgenerateBackupPoms=false
```

4. Make changes to metastore upgrade scripts. See HIVE-6555 on how this was done for HIVE 0.13.
5. Verify that the build is working with changes.
6. Commit these changes to master with a comment "Preparing for X.Y+1.0 development".

Updating Release Branch

These operations take place in the release branch.

1. Check out the release branch with:

   ```
   git clone https://git-wip-us.apache.org/repos/asf/hive.git/ <hive_src_dir>
   cd <hive_src_dir>
   git checkout branch-X.Y
   ```

2. Update the `version` property value in all pom.xml files. You should remove the `SNAPSHOT` suffix and set `version` equal to `hive-X.Y.Z` where Z is the point release number in this release series (0 for the first one, in which case this step is a no-op since you already did this above when creating the branch). Use Maven's Versions plugin to do this as follows:

   ```
   mvn versions:set -DnewVersion=0.7.0 -DgenerateBackupPoms=false
   ```

   Make sure to update the version property in standalone-metastore/pom.xml and upgrade-acid/pom.xml.
3. Remove the storage-api from the list of modules to build in the top level pom.xml. Set the storage-api.version property to the release of storage-api that you are using for your release. Make sure to set the storage-api.version property in standalone-metastore/pom.xml as well.
4. Update the the value of the `TRACKING_BRANCH` field in the `.reviewboardrc` file to point to the `origin/branch-X.Y`.
5. Verify that the build is working with changes.
6. Commit these changes with a comment "Preparing for X.Y.Z release".

7. If not already done, merge desired patches from trunk into the branch and commit these changes. Avoid usage of "git merge" to avoid too many merge commits. Either request the committer who committed that patch in master to commit to this branch, or commit it yourself, or try doing a git cherry-pick for trivial patches. Specifics of this step can be laid down by the release manager.
8. You probably also want to commit a patch (on both trunk and branch) which updates README.txt to bring it up to date (at a minimum, search+replacing references to the version number). Also check NOTICE to see if anything needs to be updated for recent library dependency changes or additions.
   a. Select all of the JIRAs for the current release that aren't FIXED and do bulk update to clear the 'Fixed Version' field.
   b. Likewise, use JIRA's Release Notes link to generate content for the RELEASE_NOTES.txt file. Be sure to select 'Text' format. (It's OK to do this with a direct commit rather than a patch.)
   c. Update the release notes in trunk with the release notes in branch.
9. Tag the release candidate (R is the release candidate number, and also starts from 0):

   ```
   git tag -a release-X.Y.Z-rcR -m "Hive X.Y.Z-rcR release."
   git push origin release-X.Y.Z-rcR
   ```

Building

1. Make sure your release notes have been updated for any new commits, and go through the previous steps if necessary.
2. Build the release (binary and source versions) after running unit tests. Manually create the sha256 files.

   ```
   % mvn install -Pdist,iceberg -DskipTests -Dmaven.javadoc.skip=true -DcreateChecksum=true

   % cd packaging/target
   % shasum -a 256 apache-hive-X.Y.Z-bin.tar.gz > apache-hive-X.Y.Z-bin.tar.gz.sha256
   % shasum -a 256 apache-hive-X.Y.Z-src.tar.gz > apache-hive-X.Y.Z-src.tar.gz.sha256
   ```

3. Verify that the SHA 256 checksums are valid:

   ```
   % shasum -a 256 -c apache-hive-X.Y.Z-bin.tar.gz.sha256
   apache-hive-X.Y.Z-bin.tar.gz: OK

   % shasum -a 256 -c apache-hive-X.Y.Z-src.tar.gz.sha256
   apache-hive-X.Y.Z-src.tar.gz: OK
   ```

4. Check that release file looks ok -- e.g., install it and run examples from tutorial.
5. Setup your PGP keys for signing the release, if you don't have them already.
   a. See https://www.apache.org/dev/release-signing.html, https://www.apache.org/dev/openpgp.html.

```
% gpg --full-generate-key
% gpg --keyserver hkp://keyserver.ubuntu.com --send-keys <PUB_KEY>
% svn co --depth files https://dist.apache.org/repos/dist/release/hive
% cd hive
% (gpg --list-sigs <NAME> && gpg --armor --export <NAME>) >> KEYS
% svn add KEYS
% svn commit -m 'Adding <FullName>'s key'
```

6. Sign the release (see Step-By-Step Guide to Mirroring Releases for more information).

```
% gpg --armor --output apache-hive-X.Y.Z-bin.tar.gz.asc --detach-sig apache-hive-X.Y.Z-bin.tar.gz
% gpg --armor --output apache-hive-X.Y.Z-src.tar.gz.asc --detach-sig apache-hive-X.Y.Z-src.tar.gz
```

7. Follow instructions in https://www.apache.org/dev/release-publishing.html#distribution to push the new release artifacts (tar.gz, tar.gz.asc, tar.gz.sha256) to the SVN staging area of the project (https://dist.apache.org/repos/dist/dev/hive/). Make sure to create a new directory for the release candidate. You may need PMC privileges to do this step – if you do not have such privileges, please ping a PMC member to do this for you.

```
svn co --depth empty https://dist.apache.org/repos/dist
cd dist

svn update --set-depth empty dev
svn update --set-depth empty dev/hive
mkdir dev/hive/rX.Y.Z/

cp <hive-source-dir>/packaging/target/apache-hive-X.Y.Z*.tar.gz* dev/hive/r.X.Y.Z/
svn add dev/hive/rX.Y.Z
svn commit -m "Hive X.Y.Z release"
```

8. Publish Maven artifacts to the Apache staging repository. Make sure to have this setup for Apache releases. Use committer setting.xml.

   Note: If you get an error `gpg: signing failed: Inappropriate ioctl for device`, try doing `` `export GPG_TTY=$(tty)` ``

   Note: if you have multiple gpg keys, you may need to specify which key to use via -Dgpg.keyname=<PRIV_KEY>

```
% mvn deploy -DskipTests -Papache-release,iceberg -Dmaven.javadoc.skip=true
```

9. Login to the Apache Nexus server and "close" the staged repository. This makes the artifacts available at a temporary URL.

Voting

1. Call a release vote on dev at hive.apache.org.

```
From: you@apache.org
To: dev@hive.apache.org
Subject: [VOTE] Apache Hive X.Y.Z Release Candidate N

Apache Hive X.Y.Z Release Candidate N is available here:

https://people.apache.org/~you/hive-X.Y.Z-candidate-N

The checksums are these:
- ff60286044d2f3faa8ad1475132cdcecf4ce9ed8faf1ed4e56a6753ebc3ab585  apache-hive-4.0.0-alpha-1-bin.tar.gz
- 07f30371df5f624352fa1d0fa50fd981a4dec6d4311bb340bace5dd7247d3015  apache-hive-4.0.0-alpha-1-src.tar.gz

Maven artifacts are available here:

https://repository.apache.org/content/repositories/orgapachehive-121/

The tag release-X.Y.Z-rcR has been applied to the source for this release in github, you can see it at
https://github.com/apache/hive/tree/release-X.Y.Z-rcR

The git commit hash is:

https://github.com/apache/hive/commit/357d4906f5c806d585fd84db57cf296e12e6049b

Voting will conclude in 72 hours.

Hive PMC Members: Please test and vote.

Thanks.
```

Verifying the Release Candidate

1. Verifying the PGP signature:

```
#get the hive committers keys file
wget https://www.apache.org/dist/hive/KEYS
or
wget https://people.apache.org/keys/group/hive.asc

gpg --import <keys file>
gpg --verify hive-X.Y.Z-bin.tar.gz.asc  hive-X.Y.Z-bin.tar.gz
gpg --verify hive-X.Y.Z.tar.gz.asc  hive-X.Y.Z.tar.gz
```

2. Verifying the sha256 checksum:
   See the step under Building.

Publishing

Once three PMC members have voted for a release, it may be published.

1. Tag the release and delete the release candidate tag:

```
git tag -s rel/release-X.Y.Z release-X.Y.Z-rcR -m "HiveX.Y.Z release."  # where -rcR was the last tagged
release candidate that passed the vote
git push origin rel/release-X.Y.Z
git tag -d release-X.Y.Z-rcR
git push origin :release-X.Y.Z-rcR
```

2. Move the release artifacts to the release area of the project (https://dist.apache.org/repos/dist/release/hive/). Using svn mv command is important otherwise you may hit size limitations applying to artifacts(**INFRA-23055** - Getting issue details... STATUS)

```
svn mv https://dist.apache.org/repos/dist/dev/hive/hive-X.Y.Z https://dist.apache.org/repos/dist/release
/hive/hive-X.Y.Z -m "Move hive-X.Y.Z release from dev to release"
```

3. Wait till the release propagates to the mirrors and appears under: https://dlcdn.apache.org/hive/
4. In your base hive source directory, generate javadocs as follows:

   ```
   mvn clean install javadoc:javadoc javadoc:aggregate -DskipTests -Pjavadoc
   ```

   After you run this, you should have javadocs present in your <hive_source_dir>/target/site/apidocs
5. Check out the javadocs svn repository as follows:

   ```
   svn co --depth empty https://svn.apache.org/repos/infra/websites/production/hive/content/javadocs
   ```

6. Copy the generated javadocs from the source repository to the javadocs repository, add and commit:

   ```
   mkdir <hive_javadocs_repo_dir>/rX.Y.Z/
   cd <hive_javadocs_repo_dir>
   cp -r <hive_source_dir>/target/site/apidocs ./rX.Y.Z/api
   svn add rX.Y.Z
   svn commit
   ```

   If this is a bugfix release, svn rm the obsoleted version. (For eg., when committing javadocs for r0.13.1, r0.13.0 would have been removed)
7. Prepare to edit the website.

   ```
   git clone https://github.com/apache/hive-site.git
   ```

8. Edit files content/downloads.mdtext and javadoc.mdtext to appropriately add entries for the new release in the appropriate location. For example, for 1.2.0, the entries made were as follows:

   ```
   ./downloads.md:### 18 May 2015 : release 1.2.0 available
   ./downloads.md:You can look at the complete [JIRA change log for this release][HIVE_1_2_0_CL].
   ./downloads.md:[HIVE_1_2_0_CL]: https://issues.apache.org/jira/secure/ReleaseNote.jspa?
   version=12329345&styleName=Text&projectId=12310843
   ./javadoc.md:  * [Hive 1.2.0 Javadocs][r1.2.0]
   ./javadoc.md:[r1.2.0]: /javadocs/r1.2.0/api/index.html
   ```

   As you can see, you will need a release note link for this release as created previously for this section.
9. Push your changes to the https://github.com/apache/hive-site/tree/gh-pages branch, and you can preview the results at https://apache.github.io/hive-site/. If everything is ok, then you can push your changes to https://github.com/apache/hive-site/tree/main branch and see the results at https://hive.apache.org/ site.
10. Update JIRA
    a. Ensure that only issues in the "Fixed" state have a "Fix Version" set to release X.Y.Z.
    b. Release the version. Visit the releases page. Select the version number you are releasing, and hit the release button. You need to have the "Admin" role in Hive's Jira for this step and the next.
    c. Close issues resolved in the release. Disable mail notifications for this bulk change.

11. Login to the Apache Nexus server and mark the release candidate artifacts as released.
12. Add the release in Apache Committee Report Helper for the next board report to pick that up automatically.
13. Send a release announcement to Hive `user` and `dev` lists as well as the Apache `announce` list. This email should be sent from your Apache email address:

```
From: you@apache.org
To: user@hive.apache.org, dev@hive.apache.org, announce@apache.org
Subject: [ANNOUNCE] Apache Hive X.Y.Z Released

The Apache Hive team is proud to announce the release of Apache Hive
version X.Y.Z.

The Apache Hive (TM) data warehouse software facilitates querying and
managing large datasets residing in distributed storage. Built on top
of Apache Hadoop (TM), it provides, among others:

* Tools to enable easy data extract/transform/load (ETL)

* A mechanism to impose structure on a variety of data formats

* Access to files stored either directly in Apache HDFS (TM) or in other
  data storage systems such as Apache HBase (TM)

* Massively parallel query execution via Apache Tez

For Hive release details and downloads, please visit:
https://hive.apache.org/downloads.html

Hive X.Y.Z Release Notes are available here: [UPDATE THIS LINK]
https://issues.apache.org/jira/secure/ReleaseNote.jspa?projectId=12310843&version=12316178

We would like to thank the many contributors who made this release
possible.

Regards,

The Apache Hive Team
```

Archive old releases

According to the INFRA archival guidelines old releases should be removed from the main download site of the project following. Check the respective guidelines and perform the necessary cleanup.

```
svn del -m "Archiving release Apache Hive 4.0.0-beta-1" https://dist.apache.org/repos/dist/release/hive/hive-
4.0.0-beta-1/
```

Preparing Branch for Future Maintenance Release

After the release has been completed, prepare the branch for the next development cycle.

1. Check out the release branch with:

   ```
   git clone https://git-wip-us.apache.org/repos/asf/hive.git/ <hive_src_dir>
   cd <hive_src_dir>
   git checkout branch-X.Y
   ```

2. Increment the `version` property value in all pom.xml files and add the `SNAPSHOT` suffix. For example, if the released version was `0.7.0`, the new value should be `0.7.1-SNAPSHOT`. Please note that the `SNAPSHOT` suffix is required in order to indicate that this is an unreleased development branch. Use Maven's Versions plugin to do this as follows:

   ```
   mvn versions:set -DnewVersion=0.7.1-SNAPSHOT -DgenerateBackupPoms=false
   ```

3. If the release number you are preparing moves the major (first) or minor (second) number, update the Hive version name in the poms. In both `pom.xml` and `standalone-metastore/pom.xml` search for the property `hive.version.shortname`. This should match the new version number.

For example, if you are working on branch-3 and have just released Hive 3.2 and are preparing the branch for Hive 3.3 development, you need to update both poms to have `<hive.version.shortname>3.3.0</hive.shortname.version>`. If however you are working on branch-3.1 and have just released Hive 3.1.2 and are preparing the branch for 3.1.3 development, this is not necessary.

4. Verify that the build is working with changes.
5. Commit these changes with a comment "Preparing for X.Y.Z+1 development".

See Also

- Apache Releases FAQ