# HowToReleasePreDSBCR

> **Deprecated since 2.8.0**
>
> **This doc refers to the releases already end of life. For current releases, please see HowToRelease.**

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

**WARNING: These instructions use the ASF Jenkins servers to build a release artifact. This is against the ASF release policies!**

These instructions have been updated for Hadoop 2.5.1 and later releases to reflect the changes to version-control (git), build-scripts and mavenization.

Earlier versions of this document are at HowToReleaseWithSvnAndAnt and 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, 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.
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>
```

4. Verify that CHANGES.txt reflect all relevant commits since the previous release. Add and commit missing ones to CHANGES.txt.

## 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. Add the release X.Y.Z to CHANGES.txt files if it doesn't already exist (leave the date as unreleased for now). Commit these changes to any **live** upstream branch. For example, if you are handling 2.6.2, commit the changes to trunk, branch-2, branch-2.6, and branch-2.7 (provided branch-2.7 is an active branch). Starting from 2.8.0 release CHANGES.txt are obsolete, the step is required only for 2.7.* series of branches.

```
git commit -a -m "Adding release X.Y.Z to CHANGES.txt"
```

2. 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"
```

3. 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"
```

4. 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"
```

5. Release branch (branch-X.Y.Z) updates:
   a. Update `hadoop-project/src/site/markdown/index.md.vm` to reflect the right versions, new features and big improvements.
   b. Update the version on branch-X.Y.Z TO X.Y.Z

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

   c. Generate `releasenotes.html` with release notes for this release. You generate these with:

```
python ./dev-support/relnotes.py -v ${vers}
```

      - If your release includes more then one version you may add additional -v options for each version. By default the previousVersion mentioned in the notes will be X.Y.Z-1, if this is not correct you can override this by setting the --previousVer option.

   d. Update `releasenotes.html`

```
mv releasenotes.${vers}.html ./hadoop-common-project/hadoop-common/src/main/docs/releasenotes.html
```

      - Note that the script generates a set of notes for HDFS, HADOOP, MAPREDUCE, and YARN too, but only common is linked from the html documentation so the indavidual ones are ignored for now.

   e. Commit these changes to branch-X.Y.Z

```
git commit -a -m "Preparing for release 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. Run mvn rat-check and fix any errors

```
mvn apache-rat:check
```

2. Set environment variable version for later steps. `export version=X.Y.Z-RCN`
3. Set the release date for X.Y.Z to the current date in each CHANGES.txt file in branch-X.Y.Z and commit the changes.

```
git commit -a -m "Set the release date for $version"
```

4. Tag the release candidate:

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

5. Push branch-X.Y.Z and the newly created tag to the remote repo.
6. 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 clean deploy -Psign,src,dist,native -Dtar -DskipTests -Dgpg.passphrase=<your-gpg-passphrase>
mvn site site:stage -DskipTests
```

7. ~~Use this Jenkins job to build the artifacts~~
8. Check that release files

```
hadoop-dist/target/hadoop-${version}.tar.gz
hadoop-dist/target/hadoop-${version}-src.tar.gz
```

look good - e.g. install it and run examples from tutorial.

9. Generate the checksums of the release file.

```
gpg --print-mds hadoop-${version}-src.tar.gz > hadoop-${version}-src.tar.gz.mds
gpg --print-mds hadoop-${version}.tar.gz > hadoop-${version}.tar.gz.mds
```

10. Sign the release. Please be sure you have completed the prerequisite step of preparing the key before signing.

```
gpg --armor --output hadoop-${version}-src.tar.gz.asc --detach-sig hadoop-${version}-src.tar.gz
gpg --armor --output hadoop-${version}.tar.gz.asc --detach-sig hadoop-${version}.tar.gz
```

11. Copy release files to a public place and ensure they are readable.

```
ssh people.apache.org mkdir public_html/hadoop-${version}
scp -p hadoop-${version}*.tar.gz* people.apache.org:public_html/hadoop-${version}
ssh people.apache.org chmod -R a+r public_html/hadoop-${version}
```

12. Log into Nexus, select "`Staging` Repositories" from the left navigation pane, select the check-box against the specific hadoop repository, and `close` the release.
13. 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
14. If the release candidate contains a serious issue, withdraw the vote, make necessary changes, and repeat this process.
15. 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. Update the release date in CHANGES.txt to the final release vote passage date, and commit them to all live upstream branches (e.g., trunk, branch-X, branch-X.Y) to reflect the one in branch-X.Y.Z. Commit and push those changes.

```
git commit -a -m "Set the release date for X.Y.Z"
```

2. Tag the release. Do it from the release branch and push the created tag to the remote repository:

```
git tag -s rel/release-X.Y.Z -m "Hadoop X.Y.Z release"
git push origin rel/release-X.Y.Z
```

3. ~~Use this Jenkins job to create the final release files~~
   Create final release files

```
mvn clean deploy -Psign,src,dist,native -Dtar -DskipTests
mvn site site:stage -DskipTests
```

4. Make sure that on Nexus all artifacts have corresponding sources and javaDoc jars.
5. 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/common/ hadoop-dist
```

   b. Generate new .mds files referring to the final release tarballs and not the RCs
   c. Copy the release files to hadoop-dist/hadoop-${version}
   d. Update the symlinks to current2 and stable2. The release directory usually contains just two releases, the most recent from two branches.
   e. Commit the changes (it requires a PMC privilege)

```
svn add hadoop-${version}
svn ci -m "Publishing the bits for release ${version}"
```

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.
8. Edit the website.
   a. Checkout the website if you haven't already

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

   b. Update the documentation links in `author/src/documentation/content/xdocs/site.xml`.
   c. Update the release news in `author/src/documentation/content/xdocs/releases.xml`.
   d. Update the news on the home page `author/src/documentation/content/xdocs/index.xml`.
   e. Copy the new release docs to svn and update the `docs/current` link, by doing the following:

```
tar xvf /www/www.apache.org/dist/hadoop/core/hadoop-${version}/hadoop-${version}.tar.gz
cp -rp hadoop-${version}/share/doc/hadoop publish/docs/r${version}
rm -r hadoop-${version}
cd publish/docs
# Update current2, current, stable and stable2 as needed.
# For example
rm current2 current
ln -s r${version} current2
ln -s current2 current
```

   f. Similarly update the symlinks for stable if need be.
   g. Add the documentation changes.

```
svn add publish/docs/r${version}
```

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

```
ant -Dforrest.home=$FORREST_HOME -Djava5.home=/usr/local/jdk1.5
firefox publish/index.html
svn commit -m "Updated site for release X.Y.Z."
```

9. Send announcements to the user and developer lists once the site changes are visible.

10. In Jira, ensure that only issues in the "Fixed" state have a "Fix Version" set to release X.Y.Z.
11. 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.
12. In Jira, close issues resolved in the release. Disable mail notifications for this bulk change.

# See Also

- Apache Releases FAQ