How To Release

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

Build environment

I often use Docker to build a clean and consistent build environment:

```
# Build a clean docker image from scratch.
DOCKER_RUN_ENTRYPOINT=true DOCKER_BUILD_XTRA_ARGS=--no-cache DOCKER_IMAGE_NAME=avro-build:X.Y-rcN ./build.sh
docker

# Run the docker container, automatically mapping ~/.gnupg, ~/.m2 and the current directory to the container.
DOCKER_IMAGE_NAME=avro-build:X.Y-rcN ./build.sh docker

# Omitting DOCKER_IMAGE_NAME will use avro-build-$USER:latest
```

Branching

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

1. Create a release branch for the release series (from the master):

   ```
   git checkout -b branch-X.Y
   ```

2. Update the version:

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

   a. Update the version number in **`share/VERSION.txt`** to be "X.Y.0'. Be sure NOT to leave a trailing newline.
   b. Update the version number in **`lang/js/package.json`** to X.Y.0
   c. Update the version number in **`pom-XXXX.xml`** used for unit tests in the **`lang/java/maven-plugin/src/test/resources`** directory too.
3. Update the release branch.

   ```
   git commit -am "Preparing for release X.Y.0"
   git push apache branch-X.Y
   ```

4. Return to the master branch.

   ```
   git checkout master
   ```

5. Update the master version to match the next major version:

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

   a. Update the version number in **`share/VERSION.txt`** to be "X.(Y+1).0-SNAPSHOT'. Be sure NOT to leave a trailing newline.
   b. Update the version number in **`lang/js/package.json`** to X.(Y+1).0-SNAPSHOT

    c. Update the version number in **`pom-XXXX.xml`** used for unit tests in the **`lang/java/maven-plugin/src/test/resources`** directory too.
6. Update the master branch

```
git commit -am "Preparing for X.(Y+1).0 development"
git push apache master
```

Updating Release Branch

These operations take place in the release branch.

1. Check out the branch with:

```
git checkout branch-X.Y
```

2. Update the version:

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

    a. Update the version number in **`share/VERSION.txt`** to be `X.Y.Z`. Be sure NOT to leave a trailing newline.
    b. Update the version number in **`lang/js/package.json`** to `X.Y.Z`. Update or regenerate **`lang/js/package-lock.json`**.
    c. Update the version number in **`pom-XXXX.xml`** used for unit tests in the **`lang/java/maven-plugin/src/test/resources`** directory too.
    d. Update the versions in **`doc/examples/java-example`** and **`doc/examples/mr-example`**.
3. Commit the update to the release branch.

```
git commit -am "Preparing for release X.Y.Z"
```

4. Add the fix version X.Y.Z to the Avro JIRA
5. If not already done, cherry-pick desired patches from master into the branch and commit these changes.

```
git checkout branch-X.Y
git cherry-pick <commit>
```

6. For each patch merged, change the fix version for the JIRA issue to be X.Y.Z
7. Go through JIRA, and git log to be sure that the issues included in the branch match in each location.
8. Update the version number in **`lang/c/version.sh`** (the variables **`libavro_micro_version`**, **`libavro_interface_age`** and **`libavro_binary_age`**) according to the libtool versioning rules as described in that file. Note the libtool version number is completely unrelated to the Avro release version number.
9. Commit these changes.

```
git commit -am "Preparing to build X.Y.Z"
```

10. Tag the release candidate (R is the release candidate number):

```
git tag -s release-X.Y.Z-rcR -m "Avro X.Y.Z rcR release."
```

11. Push the release tag and branch changes:

```
git push apache release-X.Y.Z-rcR
git push apache branch-X.Y
```

Building

Unless you have set up the required dependencies to build Avro for all languages, the following should be run inside a Docker container (see **`BUILD.md`** for instructions).

1. Build the release & run unit tests. It is strongly recommended to do this inside the docker container.

```
# To launch a bash shell inside the docker container.
./build.sh docker

# Once inside the container.
./build.sh clean test dist
```

2. Check that release files look ok - e.g. unpack the sources and run tests.
3. Sign the release (see Step-By-Step Guide to Mirroring Releases for more information).

```
./build.sh sign
# Or if using a non-default key (AVRO-3858):
GPG_LOCAL_USER=B0CCBDD6 ./build.sh sign
```

To sign a release, your key must be present in the dist/KEYS file. See the Apache guide to Signing Releases for more details. Also, if you've updated the dist/KEYS file, be sure to update the public Apache KEYS file as well:

```
svn co --depth=files https://dist.apache.org/repos/dist/release/avro/ archive-avro/
cd archive-avro/

head -n 14 KEYS
This file contains the PGP keys of various developers.

Users: pgp < KEYS
or
       gpg --import KEYS


Developers:
    pgp -kxa <your name> and append it to this file.
or
    (pgpk -ll <your name> && pgpk -xa <your name>) >> this file.
or
    (gpg --list-sigs <your name>
    && gpg --armor --export <your name>) >> this file.
```

As you can see, the header of the KEYS file has instructions to add your key. Then commit using: svn commit -m "Add key for ..."

4. Copy release files to the public staging area https://dist.apache.org/repos/dist/dev/avro/

```
svn co https://dist.apache.org/repos/dist/dev/avro/ avro-dev-dist
mkdir avro-dev-dist/avro-X.Y.Z-rcR
cp -pr avro/dist/* avro-dev-dist/avro-X.Y.Z-rcR
cd avro-dev-dist
svn add avro-X.Y.Z-rcR
svn commit -m "Artifacts for Avro X.Y.Z RCR"
```

5. Stage the default *Hadoop 2* version of Java artifacts to the Maven repository:

```
mvn clean -P dist,sign deploy -DskipTests=true -Davro.version=X.Y.Z
```

Make sure that you've followed to guide to configure Maven: http://www.apache.org/dev/publishing-maven-artifacts.html

6. Find the Staging Repository, and close it.
7. Call a release vote on dev at avro.apache.org. Include the URL of the staging repository.

Publishing

Once three PMC members have voted for a release, it may be published. Please prepare in advance your account and request correct permissions to be able to publish in the different services: PyPi, RubyGems, NuGet, npm and CPAN

1. Tag the release:

```
git checkout release-X.Y.Z-rcR
git tag -s release-X.Y.Z -m "Avro X.Y.Z release."
git push apache release-X.Y.Z
```

2. Copy release files to the release repository *(PMC permissions required)*:

```
svn copy https://dist.apache.org/repos/dist/dev/avro/avro-X.Y.Z-rcR \
https://dist.apache.org/repos/dist/release/avro/avro-X.Y.Z -m "Avro X.Y.Z release."
```

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 *(PMC permissions required)*:

```
svn co https://dist.apache.org/repos/dist/release/avro/ avro-release-dist
cd avro-release-dist
svn rm avro-A.B.C; rm stable
ln -s avro-X.Y.Z stable
svn commit -m  "Avro X.Y.Z release."
```

4. Publish **Java** artifacts to the Maven repository:
   Find the Staging Repository and release it.
5. Publish **Python** artifacts to **PyPI**. To do this you'll need an account on PyPi, and write access to the Avro package - ask the existing owners for permission if you don't have it. Also see the build and twine documentation.

```
# Install build and twine in the docker image
export PATH=~/.local/bin/:$PATH
python3 -m pip install --user --upgrade build pip twine

mkdir -p tmp/py
cd tmp/py
tar xvfz ../../dist/py/avro-X.Y.X.tar.gz
cd avro-X.Y.Z
python3 -m build  # build a sdist, and then the wheel from the sdist.
twine upload dist/*
```

6. Publish **Ruby** artifacts to **RubyGems**. Again, you'll need an account and you need to be an owner.

```
gem push dist/ruby/avro-X.Y.Z.gem
```

7. Publish **JavaScript** artifacts to **npm**. Again, you'll need an account and you need to be an owner.

```
npm login
npm publish dist/js/avro-js-X.Y.Z.tgz
```

8. Publish **C#** artifacts to **NuGet**. Again, you'll need an account and you need to be an owner. You also need to generate an additional key to publish.

```
mkdir -p tmp/csharp
cd tmp/csharp
tar xvfz ../../dist/csharp/avro-csharp-X.Y.Z.tar.gz
dotnet nuget push main/Apache.Avro.X.Y.Z.nupkg -k YOUR_KEY -s https://api.nuget.org/v3/index.json
dotnet nuget push codegen/Apache.Avro.Tools.X.Y.Z.nupkg -k YOUR_KEY -s https://api.nuget.org/v3/index.
json
dotnet nuget push codec/Avro.File.Snappy/Apache.Avro.File.Snappy.X.Y.Z.nupkg -k YOUR_KEY -s https://api.
nuget.org/v3/index.json
dotnet nuget push codec/Avro.File.BZip2/Apache.Avro.File.BZip2.X.Y.Z.nupkg -k YOUR_KEY -s https://api.
nuget.org/v3/index.json
dotnet nuget push codec/Avro.File.XZ/Apache.Avro.File.XZ.X.Y.Z.nupkg -k YOUR_KEY -s https://api.nuget.org
/v3/index.json
dotnet nuget push codec/Avro.File.Zstandard/Apache.Avro.File.Zstandard.X.Y.Z.nupkg -k YOUR_KEY -s
https://api.nuget.org/v3/index.json
```

9. Publish **Perl** artifacts to **CPAN**. Again, you'll need an account and you need to be an owner.

```
cpan-upload -u YOUR_ID Avro-X.Y.Z.tar.gz
```

10. Wait 24 hours for release to propagate to mirrors.
11. Prepare to edit the website. **THIS SECTION IS OUT OF DATE. TODO!**

```
svn co https://svn.apache.org/repos/asf/avro/site
```

12. Update the documentation links in **`author/content/xdocs/site.xml`**.
13. Update the release news in **`author/content/xdocs/releases.xml`**.
14. Run Apache Forrest using Apache Ant. Example is with forrest installed using homebrew: http://macappstore.org/apache-forrest/

```
cd site
ant -Dforrest.home=/usr/local/Cellar/apache-forrest/0.9/

# Alternatively, using forrest inside the avro-build docker.
ant -Dforrest.home=/usr/local/apache-forrest
```

15. Copy the new release docs, which are generated at the dist step, to website and update the `docs/current` link:

```
tar xzf dist/avro-doc-X.Y.Z.tar.gz
mv avro-doc-X.Y.Z ../site/publish/docs/X.Y.Z
cd ../site/publish/docs
svn add X.Y.Z
rm current
ln -s X.Y.Z current
svn commit -m "Adding documentation for release X.Y.Z."
```

16. Send an announcement email to the the mailing lists: announce@apache.org, user@avro.apache.org. and dev@avro.apache.org once the site changes are visible. Use your personal `@apache.org` email address or it will be refused.
17. In Jira, "release" the version. Visit the Manage versions page. You need to have the "Admin" role in Avro's Jira for this step and the next.
18. In Jira, close issues resolved in the release. Disable mail notifications for this bulk change.
19. Update the release information in Apache Committee Report Helper (you should receive a reminder by email).
20. Create a new release on **Github:**
    a. Draft a new release, using the release-X.Y.Z tag created in step 1.
    b. Release title: Apache Avro X.Y.Z

Post-release steps:

Prepare next release cycle

1. Update the version where "N" is one greater than the release just made:

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

    a. Update the version number in **`share/VERSION.txt`** to be "X.Y.N-SNAPSHOT'. Be sure NOT to leave a trailing newline.
    b. Update the version number in **`lang/js/package.json`** to X.Y.N-SNAPSHOT
    c. Update the version number in **`pom-XXXX.xml`** used for unit tests in the **`lang/java/maven-plugin/src/test/resources`** directory too.

Update to the release branch.

```
git commit -am "Preparing for release X.Y.N"
git push apache branch-X.Y
```

Update Documentation

1. Update the Apache Avro wikipedia page
2. Update this guide with the changes or issues you found while releasing the version. Extra points if you can reduce or automate some steps!

Validate published artifacts

You can check the published version of the artifacts is correctly available in the URL or if you want to double check that the artifacts can be downloaded /used you can use the suggested CLI

1. Java - https://search.maven.org/artifact/org.apache.avro/avro

```
docker run -it maven:3 mvn dependency:get -Dartifact=org.apache.avro:avro:X.Y.Z
```

2. Python 2 / 3 - https://pypi.org/project/avro / https://pypi.org/project/avro-python3

```
docker run -it python:2 pip install 'avro==X.Y.Z'
docker run -it python:3 pip install 'avro==X.Y.Z'
```

3. Ruby - https://rubygems.org/gems/avro

```
docker run -it ruby:2.3 gem install avro:X.Y.Z
```

4. Javascript - https://www.npmjs.com/package/avro-js

```
docker run -it node:10 npm install avro-js@X.Y.Z
```

5. C# - https://www.nuget.org/packages/Apache.Avro / https://www.nuget.org/packages/Apache.Avro.Tools

```
docker run -it mcr.microsoft.com/dotnet/core/sdk:3.1
$ dotnet new console -o test-project && cd test-project && dotnet add package Apache.Avro --version X.Y.Z
$ dotnet tool install --global Apache.Avro.Tools --version X.Y.Z
```

6. Perl - https://metacpan.org/release/Avro

```
docker run -it perl:5 cpan Avro
```

See Also

- Apache Releases FAQ
- Verifying Apache Software Foundation Releases
- Apache Infrastructure
- Apache Site Publishing Details