ReleaseProcess

This document provides a more detailed explanation of the steps involved in releasing Apache Marmotta and what happens when they are performed.

Prerequisites

- you need a PGP/GPG key
- you need the gpg-agent running (or otherwise you'll need to enter your password about 160 times)
- you need the passwords for the Apache Maven repositories (apache.releases.https, apache.snapshots.https) configured in your settings.xml
- you need to configure GIT so the passwords for the upstream repository need not be entered

Please be also aware of the following documentation on the Apache website:

- Check http://www.apache.org/dev/#release
- Read http://www.apache.org/dev/release-publishing.html
- Read http://www.apache.org/dev/publishing-maven-artifacts.html

The whole release process is completely based on Maven and the "Maven Release Plugin", i.e. I added all the necessary steps to a Maven profile that can be used to run the release. Note however that this does not release you from the burden of doing manual checking of the result afterwards!

The release in the following guide is `3.0.0-incubating`, so you have to replace that value in all commands with the actual version number!

Check that system builds, carry out some automatic license checking

Start with the latest snapshot build. Make sure your local GIT repository is committed and in sync with the remote repository.

Clean build

Run:

```
mvn clean install
```

Do not skip tests. The purpose is to see if the build is working. If it is not, it saves you a lot of trouble doing this easy Maven build. In any later phase you might need to meddle with GIT, clean up repositories, or worse.

RAT Checking

Do:

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

and possibly for creating a html report

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

This will check if all files have appropriate license headers, except those that have explicitly been included in the parent POM. If a file is excluded in the parent POM, add a comment why this is done, so others can later see the reasons. If the check fails, fix the affected source files, and start over (don't forget to GIT commit and push).

Prepare the Release

The Maven Release Plugin basically has two phases: preparation and the actual release. The prepare phase will

- update all your POM files to the release version number (e.g. 3.0.0-incubating)
- run a complete build deploying to a local repository
- if successful, commit these changes to the GIT repository (by default only local commit without a push)
- add to this commit the release tag in GIT (e.g. "3.0.0-incubating")
- update all your POM files to the next snapshot version number (e.g. 3.1.0-incubating-SNAPSHOT)
- commit these changes again to the GIT repository (again only local commit)

This means that after preparation, both your local working copy and your local repository will already be the new snapshot version! To make it a bit easier to undo such changes, I disabled the automatic GIT push to the upstream GIT repository. Undoing changes in your local repository requires only a bit of GIT meddling. Undoing changes in the remote repository is simply impossible (and dangerous).

In the "prepare" phase, it is possible to do a dry run. Use this last opportunity offered by the system to check if everything is ok:

```
mvn release:prepare -DreleaseVersion=3.0.0-incubating -DdevelopmentVersion=3.1.0-incubating-SNAPSHOT -
DautoVersionSubmodules=true -DdryRun=true
```

This command will do all the steps of a normal release preparation except running any automatic GIT commands. Can make your live a lot less troublesome in case of errors.

If all goes well, do a "mvn release:clean" and then do the actual preparation:

```
mvn release:prepare -DreleaseVersion=3.0.0-incubating -DdevelopmentVersion=3.1.0-incubating-SNAPSHOT
```

Perform the Release

The second step of the Maven Release Plugin is actually performing the release. No dry-run is possible here, so failure is bad. Make sure you have GPG agent running and all passwords configured properly. The phase will carry out the following steps:

- check out from your GIT repository the commit that was tagged with the release tag (e.g. "3.0.0-incubating")
- switch to this directory and run a complete build with the following profiles activated: "marmotta-release,installer,sign"

```
mvn release:perform -DconnectionUrl=scm:git:file://`pwd`/.git
```

The connectionUrl property will redefine the connection url for the GIT repository to the local clone. This reduces the time because the content needs not be retrieved from the remote repository.

The build will be completely clean and simulate the same environment that any user will get who checks out the release tag for the first time on his machine. The three release profiles together have been configured to carry out the following steps:

- build and install all the project, with tests in the separate checkout directory target/checkout
- do a last check with apache-rat:check to ensure the licenses are correct
- sign and deploy all Maven artifacts (with GPG signatures) to the "staging" Apache Maven Repository
- build and sign all distribution assemblies (i.e. currently the -src, -ldpath, -webapp, -installer assemblies)
- create a directory under ${root}/target/checkout/target with the name of the release (e.g. 3.0.0-incubating)
- copy all distribution assemblies to this directory
- create a template VOTE email in ${root}/vote.txt
- print further instructions on stdout

Already quite cool, but a few manual steps are still remaining to finish the release.

Upload Distribution Assemblies

First of all, check the distribution files in target/checkout/target/3.0.0-incubating, especially regarding license files, readme, and installability.

Distribution files are managed by a dedicated Subversion server. There are two distribution directories in this subversion server that are interesting for Apache Marmotta:

- https://dist.apache.org/repos/dist/dev/marmotta/ will contain the distribution assemblies for the voting process on the mailinglist
- https://dist.apache.org/repos/dist/release/marmotta/ will contain the distribution assemblies after the voting process has been finished with positive result

For continuing with the release, please check out both Subversion repositories, e.g. with:

```
svn co https://dist.apache.org/repos/dist/dev/marmotta/ marmotta-releases-voting
svn co https://dist.apache.org/repos/dist/release/marmotta/ marmotta-releases-final
```

Now copy over the directory containing the distribution assemblies created by the build process (see above) to the "voting" working copy:

```
cp -r <MARMOTTA-WORKING-COPY>/target/checkout/target/3.0.0-incubating <VOTING-WORKING-COPY>
```

add the resulting directory to subversion:

```
svn add 3.0.0-incubating
svn commit -m "new Apache Marmotta release candidate"
```

This will upload the release artifacts to the subversion server.

Close Staging Repository

The Maven artifacts are uploaded to a "staging" repository on the ASF Maven Nexus. Staging repositories are separate spaces meant for checking releases before really distributing them. To be able to access this staging repository, it first needs to be **closed**, so

- go to https://repository.apache.org and log in with your Apache ID
- select "Staging Repositories" on the left, it should display the Marmotta Staging Repository
- check the Marmotta repository (checkbox) and click on "Close" in the menu above; provide a sensible commit message!

Closing the repository will carry out some simple consistency and error checks, specifically it will check GPG signatures. You will receive an email once all checks have passed.

While in Nexus, remember the URL of the staging repository by clicking on its name. You will need it later (in my current case this is https://repository.apache.org/content/repositories/orgapachemarmotta-013/)

Push Local GIT to Upstream

Changes have been performed in your local GIT repository, but since we disabled this feature for security reasons, they are not automatically pushed to the upstream repository. So don't forget to do a

```
git push --tags
```

Send VOTE E-Mail

The final step in the release process is to send out the VOTE E-Mail to the Marmotta Developer mailinglist. A template for this mail has been prepared for you at

target/checkout/target/vote.txt

This template needs to be completed manually by some information. Particularly, you need to

- write a bit of introduction
- insert the correct URL of the Maven Staging Repository (you copied it in step 5)
- add the release notes (copy & paste from Jira Release Notes)

The voting period is 72 hours. While in incubation, when we have concluded a vote on the developer mailinglist, we also need to run an additional vote on the incubator general mailinglist, so the whole process will be 144 hours (or 6 days).

After the VOTE

VOTE failed: Try again

In case the vote fails, clean up, fix the errors and create a new release candidate (RC)

- Clean up:
  - **Drop** staging repository
  - Clean dist-server

```
svn rm https://dist.apache.org/repos/dist/dev/marmotta/3.0.0-incubating
```

- Fix the errors:
  - create a branch starting from the `3.0.0-incubating` tag

    ```
    git checkout -b release-3.0.0-incubating 3.0.0-incubating
    ```

  - do whatever fixes you need to do and commit them

    ```
    edit ...
    git add ...
    git commit -m 'fixing foo for the 3.0.0-incubating release'
    ```

- update the release tag

  ```
  git tag -f 3.0.0-incubating
  ```

- restart the next iteration with Perform the Release

VOTE passes: Announce the release

If the vote passes, the first thing is to prepare the distribution:

- Send a [RESULT][VOTE] mail
- Clean up JIRA:
  - Reassign all *open* issues to the next Version
  - Transition all *resolved* issues to **closed**
- Move the distribution files to the the release subversion; this will mirror the files to the different locations around the world

  ```
  svn move https://dist.apache.org/repos/dist/dev/marmotta/3.0.0-incubating https://dist.apache.org/repos
  /dist/release/marmotta/
  ```

- Update the downloads page on the Marmotta Webpage.
- If you created a `release-3.0.0-incubating` branch, merge it back (and delete the release branch)
- **Release** the staging repository in Nexus; that moves the artifacts to the releases repository, which is synchronized with maven central
- Push the .deb package to the stack.linkeddata.org repository (see MARMOTTA-431)

This process usually takes 24 hours. After that the release can be announced 🙂