ReleaseDeployment

DRAFT

Discussion page on release deployment strategy. It's easier to keep track of proposals and amend them in the Wiki Once agreed, this page will need to be incorporated into the Commons website

Introduction

The current release process for tarballs (e.g. bin.zip, src.tar.gz) is a bit long-winded and error prone. This page is intended to discuss some possible options and keep track of progress.

Goals of release process

There are several goals that the release process needs to fulfil.

- create the artifacts (tarball / jar) and hashes
- RM signs the artifacts
- Provide staging area(s) for both tarballs and Maven jars
- Clearly identify the artifacts that are to be voted on
- Make it easy for reviewers to check the artifacts forming part of the release
- Traceability of artifacts from vote to release directories
- Maven jars are released to Maven Central via Nexus
- Component tarballs are released to the ASF mirror system via https://dist.apache.org/repos/dist/release/commons/<component>

Maven jar process

[This is just for completeness, changes to the Maven release process are not being considered here]

These are deployed to the Nexus staging area via Maven deploy. The staging area is then published in the VOTE e-mail. If the RC vote succeeds, the staging area is promoted to release state, and Nexus ensures that the artifacts are copied to Maven Central. If the RC vote fails, the staging area is deleted.

Current tarball process

The tarballs are created using the Maven assembly plugin. This is currently configured to attach the artifacts to the build, which means that they are included in the signing process and hashes will be created.

It also means that the tarballs are included in the deploy phase, which in turn means that the tarballs are uploaded to the Nexus staging area. This has some advantages and some disadvantages.

Advantages

- all the artifacts for a release are in a single staging directory
    - so it is easier to download them for RC vote checking.
    - if the RC fails, there is no other directory to clean up
- GPG signing includes the tarballs in the set of files it signs
- hashes are created for the tarballs

Disadvantages

- once the RC completes, the tarballs have to be copied from Nexus to the release directory in SVN.

This process is quite awkward.

- Alternatively, the tarballs can be copied from the local repo (they appear to be staged here as part of the deploy); again this is awkward.
- the tarballs should then be deleted from Nexus; this stage is error-prone as it's easy to delete the wrong files.

Improving the manual parts

Using Maven to create and deploy all the artifacts is quite straightforward; no manual intervention is required. The tricky part of the process is tidying up the staging areas and copying the tarballs to the dev or release SVN trees.

Alternative deploy process

Maven deploy can easily create all the required files in a local directory; this is what the -Ptest-deploy profile does.

So an alternate approach would be to create the Nexus and dist staging areas directly from the local directory. This should allow the tarballs to be uploaded to the dist/dev area, and everything else would be uploaded to Nexus.

Assuming ths can be automated, it would eliminate most of the error-prone stages. The Nexus staging area would still need to be closed (or maybe that can be automated?) but there would be no need to tidy up Nexus.

Experiments show that the scm-publish:publish-scm goal can be used to upload arbitrary folders to the SVN dist/dev tree

And in theory there is a Nexus Maven plugin that can deploy a previously created staging tree.

So the process would be:

- mvn deploy -Ptest-deploy
- use some script (e.g. Ant as part of commons-build plugin) to move the tarballs into their own directory structure
- use scm-publish:publish-scm or svnmucc to publish the tarballs
- use Nexus Maven plugin to deploy & close the jars - the plugin does not seem to work. See below.

The script to move the tarballs could probably be written in Ant and invoked by Maven, e.g. using a specific profile.

Nexus Maven plugin seems not to work and the docs have not been updated recently. However, it is quite easy to use HTTP POST to upload a bundle to Nexus, and it's easy to create a bundle (basically just a jar of the files to be uploaded; no need for hashes, but sigs with a public id are needed).

Alternatively, deploy:deploy-file can be used to upload the main jar plus all the additional files (this is what deploy:deploy does internally, AFAICT).

However when using either bundle upload or a generated deploy-file script there is a risk that some files might be omitted.

One can either use mvn deploy -Prelease -Ptest-deploy to create all the files, sigs and hashes under target/deploy. Or "mvn verify -Prelease" will create the files and sigs (but no hashes) under target.

It's easy to create hashes in Ant; these are only needed for the svnpubsub/dist upload so can be done as part of that process.

Alternative tarball process

An alternative approach would be to detach the tarballs from the build.

This can easily be done by defining **-Dassembly.attach=false** on the command-line or adding **<attach>false</attach>** to the plugin config in the **release** profile

The tarballs can then be created using:

mvn assembly:single

However it's not currently possible to use the gpg:sign plugin to sign them without first executing the package or verify phases. But these phases always recreate the jars, so the assemblies would end up with different jars from Maven. And potentially different classes as well, because incremental compilation is currently broken when package-info.java files are used.

Advantages

- no need to copy/delete the tarballs from Nexus
- SVN traceability of tarballs from dev to release folder
- the deploy plugin ensures only the correct artifacts are uploaded

Disadvantages

- will need to update pom(s) to sign the tarballs (hopefully just once in CP)
- update poms to create the hashes (hopefully just once in CP)
- tarballs will need to be uploaded to https://dist.apache.org/repos/dist/dev/commons/<component> (will need script/new Maven plugin to do this)
- If RC fails, tarballs have to be removed (easy enough to delete containing directory)
- if RC succeeds, tarballs still have to be moved from dist/dev to dist/release directory; this needs a script/plugin for svnmucc
- deploy still adds useless .asc hashes to the staging repo; however this bug might be fixed one day

Longshot: Update Nexus to allow it to stage tarballs as well as Maven jars

This was proposed a long time ago, but in spite of initial enthusiasm, nothing came of it. See https://issues.apache.org/jira/browse/INFRA-4144

This would be a good solution, but unfortunately it seems very unlikely to happen.