

Release Day - Fedora Release Process

- Determine which modules will be released.
- Building the release artifacts
 - Perform a clean checkout of the code from Github and prepare the release.
 - Preparing the release commits
 - Inspect/Verify local updates
 - Push the changes to Github.
- Sonatype Release
 - Troubleshooting
 - Publishing on Sonatype
- Docker Release (fcrepo only)
- Make checksums for artifacts
- Github Release
- Fedora Developer Documentation
 - Troubleshooting
- Push Release Branch to Maintenance
- Update the release spreadsheet
- Repeat for each item/module you are releasing
- Last sanity checks :
- Announce release

Determine which modules will be released.

You should have created a new page in the [fedora release spreadsheet](#) which will show what you need to release.

Building the release artifacts

These variables will be used in the examples that follow. The exact values of `${ORG}`, `${REPO}`, `${CURR}` and `${NEXT}` will vary depending on which module and version is being released.

```
${ORG} = fcrepo          # the github organization for the repository being released
${REPO} = fcrepo         # the github repository name
${CURR} = 6.0.0          # the version currently being released
${NEXT} = 6.1.0-SNAPSHOT # the next snapshot version, this is normally the next minor release. i.e. 6.0.0 ->
6.1.0-SNAPSHOT or 8.3.2 -> 8.4.0-SNAPSHOT
```

Perform a clean checkout of the code from Github and prepare the release.

```
git clone https://github.com/${ORG}/${REPO}
cd ${REPO}
git checkout -b ${CURR}-RC origin/${CURR}-RC # check out the release branch
mvn release:clean
```

If `mvn release:clean` fails, you may need to revert the RC commit with `git revert HEAD`. If the parent snapshot is not available, build an old version of `fcrepo` to populate it locally.

Preparing the release commits

The `mvn release:prepare` command will update all the POM files to set the tags and versions to the release version, create a commit along with a Git tag. Then change all the versions to the next development version and change the tag back to HEAD. This will be committed to the current branch as well.

```
mvn release:prepare -DreleaseVersion=${CURR} -DdevelopmentVersion=${NEXT} -Dtag=${REPO}-${CURR} -
DautoVersionSubmodules=true -DpushChanges=false
```

The above command arguments are:



- **-DreleaseVersion** - this is the numeric version you are releasing, i.e. 6.1.0, 5.6.1, etc
- **-DdevelopmentVersion** - this is the next snapshot version. It should be one minor version above the release version and it should **always** end with *-SNAPSHOT*. i.e. 6.2.0-SNAPSHOT, 5.7.0-SNAPSHOT
- **-Dtag** - this is the git release tag which will point to the release commit. It is `<repository name>-<release version>`. i.e. `fcrepo-camel-toolbox-6.1.0`, `fcrepo-import-export-6.2.0`, `fcrepo-6.5.0`
- **-DautoVersionSubmodules=true** - because Fedora is made up of a bunch of modules, this ensure they also get processed.
- **-DpushChanges=false** - this ensure changes are **not** automatically pushed to Github.

You may be warned to update project versions. It will give you an option of which versions to update with options like (0: All, 1: Project Dependencies,). **1 (Project Dependencies)** is the default and is the correct choice to select.

For any dependency it may ask for:

1. the release version, this should be set to the correct released version (i.e. no *-SNAPSHOT*). This is the most current released version of that project, **not** the one you are currently releasing
2. the next development version, this will normally be one minor version above the release version with *-SNAPSHOT* appended. (i.e. for release version 6.3.0, the development version is 6.4.0-SNAPSHOT)

Your GPG passphrase may not be masked in terminal.



If you have more than one personal key on your GPG keyring, you can specify the correct key by adding

```
-Darguments=-Dgpg.keyname=<Your Key ID>
```

to the above *mvn* command.

for example

```
mvn release:prepare -DreleaseVersion=${CURR} -DdevelopmentVersion=${NEXT} -DautoVersionSubmodules=true -DpushChanges=false -Darguments=-Dgpg.keyname=<your signing key ID>
```

Inspect/Verify local updates

```
git diff HEAD~1
```

This will show changes from the commit before HEAD to HEAD. It should only show the changes from the release version to the new snapshot version and in at least one place the tag from `$(REPO)-$(CURR)` to *HEAD*.

i.e. From 6.0.0 6.1.0-SNAPSHOT and `fcrepo-6.0.0` HEAD

See <https://github.com/fcrepo/fcrepo/commit/8be5b22d9bb0f42b52ff0acf9294fd73fc493b99> for an example

```
git diff HEAD~2 HEAD~1
```

This will show changes from the commit 2 before HEAD to the commit before HEAD. It should only show the changes from the old *SNAPSHOT* version to the final release version and in at least one place the tag from *HEAD* to `$(REPO)-$(CURR)`.

i.e. from 6.0.0-SNAPSHOT 6.0.0 and HEAD `fcrepo-6.0.0`


See <https://github.com/fcrepo/fcrepo/commit/6a13e29dfd5931d3d7d182c841a23763bd8bf7bd> for an example

Remove your local copies of Fedora artifacts to be sure of a clean build, and build the release.

```
rm -rf ~/.m2/repository/org/fcrepo
git checkout $REPO-$CURR # detached head state i.e. > git checkout fcrepo-6.0.0
mvn clean install
```

This ensures that all the dependencies can be downloaded from maven to build the artifacts.

Up until this point, all of the changes made are strictly in your local repository and working directory.


 From this point on, the changes you make will be visible to the world and in some cases difficult to back-out of.

Be aware.

Push the changes to Github.

```
git push origin --tags
```

The maven release task relies on the tag existing in the repository, make sure there is not a branch with the same name or it will be used instead and cause errors.

 This does **not** push the changes on to any branch. The commits exist on Github but are not associated with a branch, this will come later in case something goes wrong with Sonatype.

Sonatype Release

Release the build artifacts to the Sonatype repository.

```
mvn release:perform -DperformRelease -Dgoals=deploy
```

You will be prompted to enter your GPG passphrase. Ensure you have setup your `~/.m2/settings.xml` file with the servers defined on [Fedora Release Process](#)

Your GPG passphrase may not be masked in terminal.

 If you have more than one personal key on your GPG keyring, you can specify the correct key by adding

```
-Darguments=-Dpgp.keyname=<Your Key ID>
```


to the above *mvn* command.

for example

```
mvn release:perform -DperformRelease -Dgoals=deploy -Darguments=-Dpgp.keyname=<your signing key ID>
```

Troubleshooting

If you get a warning like:


[ERROR] Provider message:
[ERROR] The git-clone command failed.
[ERROR] Command output:
[ERROR] Cloning into '/working_directory/fcrepo/fcrepo-storage-ocfl/target/checkout'...
[ERROR] [git@github.com](#): Permission denied (publickey).
[ERROR] fatal: Could not read from remote repository.
[ERROR]
[ERROR] Please make sure you have the correct access rights
[ERROR] and the repository exists.

You may need to generate a SSH key, associate it with your Github account and with your local ssh client. More instructions are available on [Github](#)


Publishing on Sonatype

The above has pushed artifacts to Sonatype in a staging repository. Now you can log in to Sonatype and release them if everything went well

- Go to <https://s01.oss.sonatype.org> and **log in**
- Click **Staging Repositories** in left navigation
- Search for "fcrepo" in upper right search box (project will not have the repository name in title)
- Select the resource and verify that it appears correct.
Ensure the published date and user appear correct, it should be you.
Click on the **Content** tab and check that inside the modules you see artifacts with the correct version numbers.
- Click **Close**, enter a brief description. Normally something like "Release X.Y.Z" (using the actual version i.e. 6.0.0) is fine.


- Click **Refresh**, you might have to do this a couple of times until it is in a Closed state.

Final step - Be aware

 If anything seems wrong, you should select the resource and click **Drop**. Then nothing is published to Sonatype. This will require re-running the release process to publish a new artifact to Sonatype, but better to be safe than sorry.

If everything went smoothly and you are ready to release, select the resource and click **Release**. Again enter a brief description, the same "Release X.Y.Z" is fine as above.

You have not pushed your commits to a branch on Github yet (See [push changes to github](#)).

 If you are **NOT** working on a past version release, i.e. you are releasing the latest version of the software. Now is a good time to push your *main* branch onto the repository.

This should **always** be a fast-forward push and should **never** require a "force push"

```
git push origin ${CURR}-RC:main
```

Now browsing Github you should see two commits with "[maven-release-plugin]" in the label. See examples in [Verify local updates](#)

On some repositories (like fcrepo/fcrepo) you are not able to push directly on to **main**, in this case you should push your release branch up to **origin**

```
git checkout ${CURR}-RC
git push origin ${CURR}-RC # i.e. git push origin 6.5.0-RC
```


Then open a pull request from the release branch to main, it will have the 2 release related commits. This pull request should **NOT** be squash merged.

--

If you **are** working on a past version, see [Push Release Branch to Maintenance](#)

Docker Release (fcrepo only)

You should still have your built Fedora webapp war file. (i.e. fedora-webapp-\${CURR}.war)

 Checkout the tagged version of Fedora and build it.

```
git checkout fcrepo-${CURR}
mvn clean install -Pfast-build
```

This assumes you fcrepo and fcrepo-docker repositories are checked out as siblings.

/home_directory

/fcrepo

/fcrepo-docker

If you don't have a fcrepo-docker repository checked out then run

```
git clone https://github.com/fcrepo-exts/fcrepo-docker.git
```

Change into the fcrepo-docker directory

We want to release with 3 tags for each release.



- MAJOR-tomcat9
- MAJOR.MINOR-tomcat9
- MAJOR.MINOR.PATCH-tomcat9

For example - for version 6.5.0

```
* 6-tomcat9
* 6.5-tomcat9
* 6.5.0-tomcat9
```

These tags are provided at the end of command below separated by spaces

For example - for version 6.5.0

```
./build-and-push-to-dockerhub.sh <path to your fcrepo repository>/fcrepo-webapp/target/fcrepo-webapp-${CURR}.
war 6-tomcat9 6.5-tomcat9 6.5.0-tomcat9
```

```
cd fcrepo-docker
```

```
DOCKER_PASSWORD=<password> DOCKER_USERNAME=<username> ./build-and-push-to-dockerhub.sh <path to your fcrepo
repository>/fcrepo-webapp/target/fcrepo-webapp-${CURR}.war ${TAGS as defined above}
```

Make checksums for artifacts

Locate/build the artifacts

For Fedora core:

The WAR file will be in the *fcrepo/fcrepo-webapp/target* directory. It will have the name *fcrepo-webapp-\${CURR}.war*

Copy it to a new location outside of the *fcrepo* directory, then build the one-click artifact.

```
mvn clean install -Pone-click -Pfast-build
```

The `-Pfast-build` flag skips all the tests to make it build much faster.

This builds an executable JAR file in the *fcrepo/fcrepo-webapp/target* directory. It will have the name *fcrepo-webapp-\${CURR}-jetty-console.jar*

Copy this file beside the WAR file from above.

For all other projects

The JAR file will be in the repository's *target* directory.

Next we want to make a SHA-1 and MD5 checksum for the artifact(s).

Note: The checksum files should have lines of the format "[checksum] [filename]"

Example (OSX)

```
shasum -al <artifact name> >> <artifact name>.sha1
md5 -r <artifact name> >> <artifact name>.md5
```

Fedora example (OSX)

```
shasum -al fcrepo-webapp-6.0.0.war >> fcrepo-webapp-6.0.0.war.sha1
md5 -r fcrepo-webapp-6.0.0.war >> fcrepo-webapp-6.0.0.war.md5
```

Example (*nix)

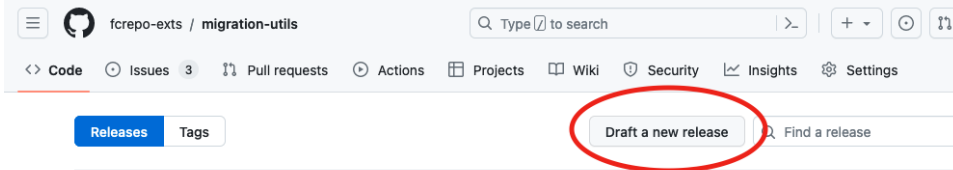
```
shasum <artifact name> >> <artifact name>.sha1
md5sum <artifact name> >> <artifact name>.md5
```

Fedora example (*nix)

```
shasum fcrepo-webapp-6.0.0.war >> fcrepo-webapp-6.0.0.war.sha1
md5sum fcrepo-webapp-6.0.0.war >> fcrepo-webapp-6.0.0.war.md5
```

Github Release

- Go to [https://github.com/\\$ORG/\\$REPO/releases](https://github.com/$ORG/$REPO/releases)
- Click the **Draft New Release** button.



- Click **Choose a tag** and find and select the new release tag.
- Set the title to something like "Release \$CURR" (i.e. "Release 6.0.0")
- Click the **Generate release notes** button to generate some default changes, edit as necessary.
- Attach the artifacts and their checksum files from the previous step.
- Ensure **Set as the latest release** is checked
- Click **Publish Release**

Fedora Developer Documentation

This is done to update the Github Pages documentation:

```
mvn site-deploy -DskipTests
```

fcrepo pages will be visible at [http://docs.fcrepo.org/site/\\$CURR/\\$REPO/](http://docs.fcrepo.org/site/$CURR/$REPO/)

Other module pages will be located at: [http://\\$ORG.github.io/\\$REPO/site/\\$CURR/fcrepo/\\$REPO](http://$ORG.github.io/$REPO/site/$CURR/fcrepo/$REPO)

For fcrepo/fcrepo and fcrepo-exts/fcrepo-camel, manually add links to the current releases. The easiest way to do this is to search for an old version number and copy/update for the current release.

Troubleshooting

Error creating blob: API rate limit exceeded

GitHub only allows a certain number of requests per hour. Once that number is hit you'll have to wait an hour before resuming your operation. The site documentation may exhaust this limit several times.
If you get the following error:

Error creating blob: You have triggered an abuse detection mechanism and have been temporarily blocked from calling the API. Please retry your request again later. (403)

You may consider using the patched version of site-maven-plugin: <https://github.com/github/maven-plugins/commit/d4ccf887098b18e9a23b7316ecf96348a2c73d0a>

Or a 405 error or a 502 error:

You may consider using the patched version of site-maven-plugin: <https://github.com/github/maven-plugins/commit/09b282544a1208be63bd012c2cdfd4d58e3f2d22>

If you use two factor authentication with Github and have a **Personal Access Token** setup for Maven. Ensure that this token has the **repo** and **user:email** permissions.


If you get the following error:

```
Failed to execute goal com.github.github:site-maven-plugin:0.12:site (github) on project fcrepo-module-auth-rbac1: Error creating commit: Invalid request.
```

```
[ERROR]
[ERROR] For 'properties/name', nil is not a string.
[ERROR] For 'properties/name', nil is not a string. (422)
```

You will need to ensure that the "Name" field of your github profile is not null. Fix it by going to github.com and updating your profile.

Error creating blob: cannot retry due to server authentication, in streaming mode

 This is an authentication error, check your password or token in your Maven settings.xml file. If you use 2-factor you can create a new token with the following permissions *notifications, public_repo, repo:status, repo_deployment, user:email*

Push Release Branch to Maintenance

This is **only** necessary for past versions of software when a bug fix is being applied. Otherwise you would have updated the project's *main* branch at the end of [Publish on sonatype](#) above.

The release branch has changes made since code freeze. It also contains the update to the version numbers for future development.

```
git checkout ${CURR}-RC # this is your local copy of the release branch
git log
```

Ensure that your commit history matches the release branch's commit history, except for the two additional commits.

1. Changing from SNAPSHOT version to release version. Something like **[maven-release-plugin] prepare release \$REPO-\$CURR**
2. Changing from release version to next development version. Something like **[maven-release-plugin] prepare for next development iteration**

If this appears correct, you can push your release branch on to the maintenance branch.

```
git push origin ${CURR}-RC:${CURR}-maintenance
```

Update the release spreadsheet

Hopefully you did this as you went along, but if not review your [Fedora release spreadsheet](#) page and ensure you didn't miss anything.

Repeat for each item/module you are releasing

Depending on your comfort you may attack some of these in different ways. (i.e. perform all builds and sonatype releases, then generate all the Github releases last)

Last sanity checks :

1. Assuming a Fedora core release: Download and run the *fcrepo-webapp-\${CURR}-jetty-console.jar*
2. Make sure that the checksums of the artifacts published in the github release page match those in sonatype.
3. If we're talking about a maintenance release, make sure that all commits that went into the maintenance release are also on main (where appropriate)

Announce release

Let [Arran Griffith](#) and/or [Dan Field](#) know that the release is complete and can be announced.