Self Releases Workflow (Nexus2)

Self releases are now possible after the migration to latest global-jjbs staging jobs

gerrit-maven-stage and gerrit-maven-docker-stage

Before teams start using self releases ...

Please note that this process will be possible by committers with Merge/+2 Rights in the repos.

Please make sure your INFO.yaml files are updated and that all committers have been added.

More information of these procedure can be found here. (Please make sure to read this and ask any

questions to support.linuxfoundation.org:

https://github.com/lfit/releng-global-jjb/blob/master/docs/jjb/lf-release-jobs.rst

How does it work?

The process is very simple. It will require for tech teams to post a new "releases" folder with a yaml file for each release

they need. This file gets verified and, once merged, the release will be posted.

Here are the steps:

1. The tech team needs to make sure their committer list is up to date as these will be the people approved to make releases.
    a. The reason behind this, only committers can +2 and merge changes in their tech team repo.
    b. The releases merge job will be triggered by files merged in the repo and will execute a release.
2. Tech team needs to add "{project-name}-gerrit-release-jobs" to their ci-management yaml files.
    a. This group introduces "gerrit-release-verify" and "gerrit-release-merge"
3. Once a release candidate is build using gerrit-maven-stage, a new file must be added to your project repo describing the release and referring to the gerrit-maven-stage log:
    a. This file needs to be located in the root repo under a "releases" folder.
    b. An example can be viewed here: https://github.com/lfit/releng-global-jjb/blob/master/docs/jjb/lf-release-jobs.rst
        i. Name of the file should match the semantic version of the release being published. (For example: releases/1.0.0.yaml)
        ii. distribution_type: 'maven' (Future expansion will allow "container" to be provided"
        iii. version: '#.#.#' (Release semantic version)
        iv. tag_release: false (By default, tagging a repo with a release version is set to true. If you want to skip it, set it to false)
        v. project: 'project-name' (Project name, for example 'ccsdk-parent')
        vi. log-dir: 'pointer_to_maven_stage_job/build_number/' (for example: 'ccsdk-parent-maven-stage-master/2/')
        vii. maven_central_url: 'oss.sonatype.org' (Optional, in case the team want's to publish to Maven Central)
4. This file will trigger "{project-name}-release-verify-{stream}"
    a. This job can also be triggered using the comment "recheck|reverify"
    b. The verify job will make sure the release file contains the needed information and that the candidate exists.
5. Tech team needs to +2 this new change and merge it. **Please do not override any -1 Verify from Jenkins.**
6. The merge will trigger "{project-name}-release-merge-{stream}"
    a. This job can also be triggered using the comment "remerge"
    b. This job will push the release and tag the repo.

After your self release ...

Once your self release file was merged and processed by "gerrit-release-verify" and "gerrit-release-merge"...

- DO NOT attempt to revert files in releases/ Even if the release was not needed, we need to keep track on what happened in that repo also the tag needs to be kept in the repo
- DO NOT attempt to re-tag the repo with the same version this will fail as the gerrit-release-merge job already tagged it
- DO NOT modify releases files Once a releases file is merged, it is immediately processed. If the team needs to release a new build number, please bump your versions and generate another stage-release
- DO NOT re-use the same stage-release build number for multiple releases files Once an autorelease package is pushed, it is closed and it cannot be re-released.
- MAKE SURE your project is 100% free from using nexus-staging-maven-plugin. This was a must do in the global-jjb migration and teams still using it WILL BE facing issues deploying all their artifacts.

    - Using this plugin, pre-deploys image in a bad way before lftools has a chance to compile all artifacts needed for the release candidate
- DO NOT cherry-pick release files across branches. Duplicated release files will make the verify and merge job fail as those release were already processed.
    - Releases from master should be created in master, releases from sub-branches like "el-alto" should be created in "el-alto"