Self Releases Workflow (Nexus3 - Docker)

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"
    b. Teams that have added these jobs for Nexus2 releases, will need to use a docker node if they want to use docker self releases. For example: https://gerrit.onap.org/r/#/c/ci-management/+/95775/
3. Once a release candidate is build using gerrit-maven-docker-stage, a new file must be added to your project repo describing the release and referring to the gerrit-maven-docker-stage log:
    a. **Please create 1 releases file per Gerrit repo.** The job will throw an error if found more than 1 release files: https://github.com/lfit/releng-global-jjb/blob/master/shell/release-job.sh#L49
    b. This file needs to be located in the root repo under a "releases" folder.
    c. An example can be viewed here: https://gerrit.onap.org/r/#/c/clamp/+/95770/
        i. Name of the file should match the semantic version of the release being published. (For example: releases/1.0.0-container.yaml) This is not enforced, but is good to follow a convention across repos.
            1. In reality, the file can be named whatever but we want to keep consistency and recommend teams to use #.#.#-container.yaml (Docker) and #.#.#.yaml (Maven)
        ii. distribution_type: 'container'
        iii. container_release_tag: '#.#.#' (Release semantic version, release version that will appear in the docker.releases repo)
        iv. tag_release: false (By default, tagging a repo with the container_release_tag 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_docker_stage_job/build_number/' (for example: 'ccsdk-parent-maven-docker-stage-master/2/')
        vii. ref: 'commit that the team would like to tag' (If the repo was already tagged when maven releases was processed, no re-tagging will happen. This job just tags using the version number that is being released, it does not tag official ONAP releases like 5.0.0-ONAP)
        viii. containers: list of image names and tags to be pulled for the releases. For example:

```
- name: test-backend
  version: 1.0.0-20190806T184921Z
- name: test-frontend
  version: 1.0.0-20190806T184921Z
```

1. 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.
2. Tech team needs to +2 this new change and merge it. **Please do not override any -1 Verify from Jenkins.**
3. 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 docker-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 properly calling the Docker profile in pom while running docker-stage-release. If there are no docker images pushed, the job will fail.
- 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"