How to release CCSDK sli artifacts

Releasing initial version for a new ONAP release

When releasing initial versions for a new ONAP release, we must release ALL CCSDK repositories, in the following order:

1. Release maven artifacts for CCSDK parent poms (ccsdk/parent repository)
2. Release maven artifacts for ccsdk/sli/core repository
3. Release maven artifacts for the following repositories:
    - ccsdk/sli/adaptors
    - ccsdk/sli/northbound
    - ccsdk/sli/plugins
    - ccsdk/features
    - ccsdk/oran
4. Release maven artifacts for ccsdk/distribution
5. Release docker containers for ccsdk/distribution

Releasing maven artifacts

To release maven artifacts for a CCSDK repository:

1. Create a new staging version, if one does not already exist:
    a. In gerrit, find the most recent closed review for the repo in the branch you are using. For example, to find the most recent closed review in the master branch for ccsdk/parent, search would be "is:closed project:ccsdk/parent branch:master"
    b. Post the comment "stage-release" to the most recently merged change.
2. Use the self release process to release maven artifacts (see Self Releases Workflow (Nexus2)). To do this:
    a. Create releases/{version}.yaml (e.g. by copying a existing file in that directory to a new one named for the release you are about to build).
    b. Edit the following fields in the yaml:
        i. Change "version" to the version you want to build. **Note: that version must match the version created in the staging build in step 1.**
        ii. Change "log_dir" to match the Jenkins job for the staging build that was run in step 1.
    c. **For repositories WITHOUT docker containers only -** roll all your poms to the next snapshot version. You can do this easily using the script "changeVersion.sh" provided in ccsdk/parent/tools. For example, to change version to 2.1.1-SNAPSHOT, change directory to the top most level of the repository and run changeVersion.sh as follows:
        changeVersion.sh 2.1.1-SNAPSHOT
    d. Push your changes as a code review:
        git add .
        git commit -as -m "Review comment ...."
        git review

These steps will create a gerrit review that, once merged, will cause the maven artifacts in the specified staging version to be released. An example of such a gerrit review is here

Special considerations for ccsdk/parent releases

Before releasing the ccsdk/parent, be sure to check the following files to verify the version property settings for the ccsdk repositories (e.g. ccsdk.sli.core.version):

- ccsdk/parent/oparent/pom.xml
- ccsdk/parent/odlparent/setup/src/main/resources/pom-template.xml
- ccsdk/parent/springboot/springboot-setup/src/main/resources/pom-template.xml

If you change any versions there, be sure to do a local build and push all changes (since odlparent and springboot pom.xmls are generated on the fly from the pom-template.xml files).

After releasing ccsdk/parent, please check the other ccsdk repositories to verify that none of the repositories are using the snapshot version you just released. If they are, then please update the repo(s) in question to use the version of ccsdk/parents you just released. You can make this change using the script updParentVersion.sh in ccsdk/parent/tools. For example, to change to version 2.1.0, you would run the following from the top level directory of the repo to be changed:
updParentVersion.sh . 2.1.0

Special considerations for ccsdk/distribution releases.

Before releasing ccsdk/distribution, be sure to check the versions in ccsdk/distribution/odlsli/odlsli-alpine/pom.xml to make sure you are using the correct released versions of maven artifacts.

Releasing docker containers

To release docker containers for ccsdk/distribution, first follow the steps above to release maven artifacts for this repository. Once that is complete, docker containers can be released via the self release process (see Self Releases Workflow (Nexus3 - Docker)) by the following steps:

1. Create releases/{version}-container.yaml (e.g. by copying a existing file in that directory to a new one named for the release you are about to build).

2. Edit the following fields in the yaml:
    a. Change "version" to the version you want to build. **Note: that version must match the version created in the docker staging build.**
    b. Change "log_dir" to match the Jenkins job for the docker staging build you want to release.
    c. Change "ref" to match the revision id for this change. To find this value, open the console log for the docker staging job above. Search for the string "Checking Out Revision" to find this hexadecimal value. For example:
        `Checking out Revision cf6fa2203af1f9f161029f2e93a5f19bb18c84af (refs/remotes/origin/guilin)`
    d. Verify the list of containers to be released. For each container, update the version to the correct timestamped version to be released. You can find this by searching for the string <release>-STAGING (e.g. 2.1.1-STAGING) in the console log for the staging job.
3. Roll all your poms to the next snapshot version. You can do this easily using the script "changeVersion.sh" provided in ccsdk/parent/tools. For example, to change version to 2.1.1-SNAPSHOT, change directory to the top most level of the repository and run changeVersion.sh as follows:
    changeVersion.sh 2.1.1-SNAPSHOT
4. Push your changes as a code review:
    git add .
    git commit -as -m "Review comment ...."
    git review

These steps will create a gerrit review that, once merged, will cause the maven artifacts in the specified staging version to be released. An example of such a gerrit review is here

Timing considerations

It is important to remember that maven artifacts and docker containers MUST be released in separate reviews. Do not try to combine these into the same review. Also, always make sure that the release-merge job for maven artifacts completes successfully before merging the review to release docker containers to avoid conflicts (both jobs trigger the same Jenkins job, and therefore share a workspace directory).