Managing Container Image Tags

We want to document the process we will use for managing docker images in OOM helm charts for Dublin and subsequent releases.

The primary goal is to make installation of ONAP as simple as practical.

Having the image tag in the values.yaml is the simplest method for the release version and that will be the result of the process we outline below.

Integration still needs to be able to over ride those values upon occasion and may use supporting scripts and docker-manifest-staging.csv files to have a internal process but only ONAP folks doing installs off the SNAPSHOT version in our integration Labs will need to be aware of this.

Dublin Release

The process will be:

1. At RC0 we will convert the OOM branch to use release tag’s for docker image (e.g NOT SNAPSHOT/STAGING)
   a. PTL’s will publish their release artifacts to the release repo per normal process
   b. All release images should be based on SNAPSHOT/STAGING images that at least pass health check and pair wise tests.
   c. After the release artifacts are published, the PTL will update the OOM helm charts for their project to point to the release tagged containers.
      i. The sequence is important since the OOM verify job will confirm the image is in nexus3.onap.org
   d. Between RC0 and final release, the OOM team will ensure same day service for simple release tag update merges
      i. Integration team members will provide additional committer support for this phase
      ii. Only simple values.yaml image tag references will get this expedited approach where we don’t do an ONAP install to confirm the helm chart change
   e. The integration team may assist in converting the current docker-manifest.csv values into a helm chart update as we get close to RC0 if that makes sense.
   f. We will remove docker-manifest.csv after RC0 in Dublin.

2. Before RC0, in Dublin, the PTLs should update the OOM master branch helm charts with SNAPSHOT or STAGING tags that you have for your master branch.
   a. Docker-manifest-staging.csv may still be used by the integration team to override values when we need to rollback to a previous version of the container but PTLs should focus on the OOM master branch updates.
   b. Integration will work with OOM to do a single values.yaml update to the oom helm charts so that for Dublin the master branch of OOM will match the current docker-manifest-staging.csv early next week so PTL’s do not have to do that change right now.

3. Reminder that after RC2, the only changes should be Integration approved JIRAs of High and Highest priority since we run 72 hour stability tests after RC2 and will not have time to accept retests unless its release breaking

4. Reminder that after M4, the changes should be bug fixes only and not new features so that we are focused on getting to reliable operation and not introducing new defect opportunities.

Example Use Case to Update docker image between RC0 and final release

1. RC0 has passed and the project has a release container A.B.1 in the nexus3 repo.
2. A Jira of high/highest is opened so a new container will be required
3. Project builds a A.B.2-STAGING-latest or equivalent STAGING/SNAPSHOT version
4. docker-manifest-staging.csv would be updated for this version
5. Integration with project tests the new container using docker-manifest-staging.csv and integration-override.yaml to use the A.B.2-STAGING-latest image
6. If the new container fails testing:
   a. the project can rebuild using the same tag A.B.2-STAGING-latest
   b. this repeats until the container passes
7. Once the container passes tests:
   a. The project will release the container as A.B.2
   b. oom values.yaml image tag will be updated from A.B.1 to A.B.2 after the new container is available in nexus3

El Alto Release

1. oom master branch allows only released images going forward.
2. A new oom staging branch has been created.
3. The docker-staging-manifest.csv in the integration repo has been decommissioned. The staging image tags that teams used to put in the staging manifest should instead be committed into the oom staging branch helm charts directly. This also provides the benefit of allowing supporting helm chart changes to go in together with the image tag changes.
4. The integration daily staging deployments into the Wind River Integration-Staging-Daily tenant deploy using oom staging branch helm charts.
5. Based on results of the staging deployment tests, teams may decide to release their images as they see fit.
6. Once LF has released the images, the release image tag changes (along with corresponding chart changes as necessary) can be submitted into oom master branch.
7. Once the oom master branch has been updated, the staging branch image tags should be bumped to the next staging version being built (i.e. the staging version after the current release version).