Docker Image Build Guidelines

Based on the ONAP community decision to move toward a decoupled versioning and release model (i.e. each project team will version and release their own artifacts on their own schedules), we determined that it makes the most sense for individual project teams to fully manage their own docker image builds. As such, the Integration team will no longer maintain a central docker build process like we did before in OPEN-O.

This is set of recommendations and guidelines for each ONAP project to follow as they work on creating a docker image build process.

High Level Artifact Build Flow

Ideally, the docker image build process should be fully separated from the Java/Python artifact build process. This means:

- One set of Jenkins jobs will build and deploy Java/Python artifacts to Nexus
- One set of Jenkins jobs will build the docker images using Java/Python artifacts already deployed to Nexus before
- Jenkins job dependencies should be set up so that the former can trigger the latter

Once we include considerations of the SNAPSHOT/STAGING/RELEASE docker tags, we end up with the following general development/release flow:

1. Produce SNAPSHOT Java artifact. Test this in a SNAPSHOT docker image.
2. Produce staging (release candidate) Java artifact. Test this in a SNAPSHOT docker image.
3. Produce release Java artifact by picking one of the candidates from staging.
4. Produce STAGING docker image using the release Java artifact. Use this in E2E test flows.
5. Produce RELEASE docker image by picking one of the candidate STAGING docker images.

Migrating from OPEN-O Docker Builds

The OPEN-O Integration team used to centrally run all the docker image builds using files source controlled under integration.git/test/csit/docker/. A copy of the scripts used in OPEN-O has been placed within ONAP under integration.git/packaging/docker/scripts/. However, the individual image scripts (instanc e-config.sh, etc.) were not carried over. The OPEN-O docker build process copied those files into the docker build directory, and auto-generated a Dockerfile definition and supporting scripts (e.g. docker-entrypoint.sh) that called those scripts as appropriate.

If you still have a copy of the OPEN-O integration repo available, the best way to migrate from the OPEN-O docker image definition is as follows:

1. Run integration/test/csit/docker/scripts/gen-all-dockerfiles.sh. This will create the docker definitions for all the microservices defined in the test/csit /docker/ directory.
2. Go to the subdirectory for your microservice. Inside, you'll find that a target/ directory was created by the above script. This is the auto-generated docker image definition for your microservice. You should be able to run "docker build" directly against this directory.
3. Copy the contents of this directory (except for *.txt) to your ONAP repo, under a docker/ directory. The contents of *.txt are concatenated into Dockerfile, so you no longer need *.txt, and should just manually update Dockerfile going forward. Everything left in this docker directory is your docker image definition. Source control these files.
4. Update the Dockerfile for ONAP. In particular, replace the wget commands that download your Nexus artifacts with the ONAP groupId/artifactId /path information.
5. You should be able to run "docker build" locally to build the docker image from this directory.
6. TBD: Add a JJB job that will use this docker/ directory to build the docker image.

Externalize Service Dependencies (MySQL, Redis, etc.)

The OPEN-O docker images were created specifically for CSIT only, which is why it was ok to directly incorporate services like MySQL / Redis in the microservice docker images themselves. In ONAP, the docker images are intended to be run for production, so any necessary services should be run as separate docker containers outside your microservice container. So, please define your docker images so that they can have things like MySQL IP addresses passed in from the outside.

Docker Requirements from OOM

The OOM project may place additional requirements on the docker image definitions for things like service discovery, etc. TBD