Independent Versioning and Release Process

Overview

- No cross-project SNAPSHOT dependencies
- Teams to version and release their own artifacts on their own schedule
- Version numbers should conform to the Semantic Versioning 2.0.0 Specification
- ONAP community "simultaneous release" is composed of a collection of artifact versions defined in a "version manifest" in source control
    - Teams to declare and update the “correct version” for cross-project use and for inclusion in simultaneous release
    - Manifests located in integration repo: https://git.onap.org/integration/tree/version-manifest/src/main/resources
    - TSC to approve version manifest for simultaneous release, e.g. Amsterdam
- ONAP Version Manifest Maven Plugin (Decommissioned) to check against outdated dependencies vs. the manifest
    - Teams to version bump their dependencies per their convenience

Benefits

- No unexpected build failures due to upstream SNAPSHOT changes
- No need for cross-project consolidated “mega builds” to check dependency issues
- Avoids issues with trying to synchronize artifact version numbers across projects
- Improved dev and build cycle time

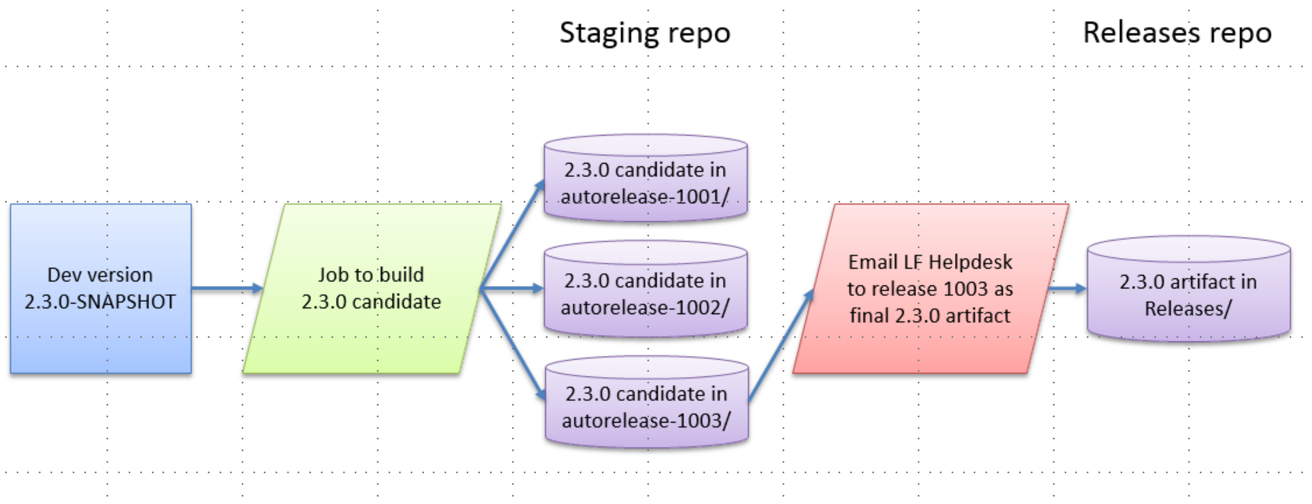
Rationale

Details on the history and discussions that led to this process can be found at Release Versioning Strategy.

Roll-out Schedule

- Due to potential cascading dependencies, and the fact that some teams may not be ready to formally release artifacts before M4 Code Freeze, we expect some teams to need more time to be able to remove all their SNAPSHOT dependencies.
- Ideally, by RC0 all project teams should have already formally released at least one version of their artifacts. So by this time we should be able to start enforcing the no SNAPSHOTs requirement.

Java (Maven) Artifacts Release Process


Staging repo
Releases repo
Dev version 2.3.0-SNAPSHOT
Job to build 2.3.0 candidate
2.3.0 candidate in autorelease-1001/
2.3.0 candidate in autorelease-1002/
2.3.0 candidate in autorelease-1003/
Email LF Helpdesk to release 1003 as final 2.3.0 artifact
2.3.0 artifact in Releases/

- Ensure that your artifacts inherit from O-Parent (oparent)
    - If not possible, please maintain your own implementations of the various configurations and checks provided by oparent
- Remove all external SNAPSHOT dependencies
    - External = cross-project (including oparent) or 3rd party
    - Check version manifest at https://git.onap.org/integration/tree/version-manifest/src/main/resources for the right version to use for your cross-project dependencies
        - The Integration team is providing a Maven plugin to warn you of outdated references: ONAP Version Manifest Maven Plugin (Decommissioned). To use it, run the following command:

```
mvn org.onap.integration:version-manifest:version-check
```

    - If your upstream cross-project dependencies haven't entered their artifacts in the manifest above, please contact the respective project team to get them to version/release their artifacts and add their entries to the manifest
    - Remove ecomp-staging Nexus repo from your local ~/.m2/settings.xml repositories list; all release-versioned artifact dependencies should be fulfilled from the ecomp-releases repo only going forward
- Set up gerrit-maven-stage (staging job from global-jjb) to deploy candidate artifacts to Staging in Nexus2
    - Generates candidate "autorelease-xxxx" directories in Nexus
    - Ensure that your version.properties file has the right version number defined for the intended release
- Ensure that the staging jobs above have completed and generated candidate artifacts
- Perform any necessary testing against the candidate artifacts
- Create a service request to LF Releng (support.linuxfoundation.org) to request a release of the staging candidate
    - Specify the specific Jenkins build job that generated the selected candidate build, e.g. https://jenkins.onap.org/view/aai-aai-common/job/aai-aai-common-maven-stage-master/23/
    - LF will release the artifacts to Releases repo
    - LF will place a GPG signed tag on the specific commit in Gerrit repo
- Update the declared version numbers for your respective artifacts in the java version manifest: https://git.onap.org/integration/tree/version-manifest/src/main/resources/java-manifest.csv
- Update the CHANGELOG to describe the changes that were part of this release
    - TBD: CHANGELOG structure and update process is being developed by the Documentation project
- Bump your own version numbers for ongoing development
    - SNAPSHOT versions in pom.xml
    - Staging/Release version in version.properties
    - The gerrit-maven-stage job provides a patch that the teams can apply for their convenience. e.g. https://logs.onap.org/production/vex-yul-ecomp-jenkins-1/aai-aai-common-maven-stage-master/23/patches/

Docker Images Release Process

Prepare your docker images. Here is the high level flow relating the various Java artifact versions to Docker tags:

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.

Docker image release process:

- Set up gerrit-maven-docker-stage (docker staging job from global-jjb) to produce the STAGING docker images.
- Ensure that the docker staging jobs have completed and generated candidate artifacts
- Perform any necessary testing against the candidate artifacts
- Create a service request to LF Releng (support.linuxfoundation.org) to request a release of the staging candidate

        - Specify the specific Jenkins build job that generated the selected candidate build, e.g. https://jenkins.onap.org/view/clamp/job/clamp-maven-docker-stage-master/52/
        - LF to re-tag the selected STAGING docker image with a RELEASE version tag
    - Update the declared version number for your docker image in the docker version manifest: https://git.onap.org/integration/tree/version-manifest/src/main/resources/docker-manifest.csv
    - Update the CHANGELOG to describe the changes that were part of this release
        - TBD: CHANGELOG structure and update process is being developed by the Documentation project
    - Bump your own version numbers for ongoing development
        - Staging/Release version in version.properties

Standardized Docker Tagging

TODO: Michael to send out onap-discuss meet for 2nd next PTL meet - based on brainstorming the docker versioning in terms of the CD process (Marco, Gildas, Gary, Michael)

Adding the page ONAP Docker Tagging.

OOM-500 - Getting issue details... STATUS

DI 1: Standard Tag Format

Current status within Docker

ONAP Nexus 3 which contains ONAP Docker images is structured as below:

- Docker Release: ONAP released Images
- Docker Staging: currently empty
- Docker Snapshot: the big pile

What do we Need stanardardized Docker format

As Docker Snapshot is a cumulative repository, given a version, a keyword and the name of an image, the need is a systematic method to sort chronologically images based on Nexus version field.

This will help to community in providing an automated deployment using tag sets on a standard format.

Proposal

The proposed docker tag format to align across all the teams is the following:

```
x.y.z-KEYWORD-yyyymmddThhmmssZ
```

X.Y.Z follows the Semantic Versionning

KEYWORD: SNAPSHOT or STAGING

Example

```
1.1.2-SNAPSHOT-20181231T234559Z
```

See similar examples of the current tag structure by browsing the links below. Unfortunaltly there is no consistency among projects. Note these examples have additional keyword (such as SNAPSHOT) that are currently not deemed necessary.

https://nexus3.onap.org/#browse/browse:docker.snapshot

Just picking any 2 random here - issue is not any particular project

https://nexus3.onap.org/#browse/browse:docker.snapshot:v2%2Fonap%2Faai-traversal%2Fmanifests%2F1.2.0-SNAPSHOT-STAGING-20180122T124321

https://nexus3.onap.org/#browse/browse:docker.snapshot:v2%2Fonap%2Fbase_sdc-cassandra%2Fmanifests%2F1.2.0-SNAPSHOT-2018.01.29.15.22

Raw notes from 20180202 - scheduling a PTL meet for after the vF2F

20180202 Notes:
- Alexis, Marco proposal;
1) which docker tag set to test - latest or timestamp based?

Action: could force either on teams
2) how do we report -1 failure of CD - check docker merge build tag version
(leave how to fix - original tagset or latest tagset to the teams)

tag once on image production -

Discussion
- central/integration uses released docker manifest - known tested as good
- team declares a released manifest version as good (for now manually by team - later after the CD has marked the version as tested)
- component teams deploy using their snapshot + same released docker manifest for other teams' docker

Proposal: follow nexus.onap.org naming
- latest uses snapshot - follow maven format
staging
https://nexus.onap.org/#view-repositories;staging~browsestorage
name-1.2.0

snapshot
https://nexus.onap.org/#view-repositories;snapshots~browsestorage
name-1.2.0-timestamp

proposal: use symantic version v1.1.1 - is released (no latest tag) - unchanged
1.1.2-snapshot is the next under test
1.1.2-timestamp (here we need LF to pick up the format)
proposal: Timestamp-yyyy-mm-dd-Thh-mm-ssZ
or version-yyyymmddThhmmssZ (no :)
1.1.2-20181231T234559Z
fill out and merge Gildas slides
to
https://wiki.onap.org/display/DW/Independent+Versioning+and+Release+Process#IndependentVersioningandReleaseProcess-StandardizedDockerTagging
send mail to discuss and meeting 2nd monday

1.1.2-snapshot - points to the "latest" of 1.1.2-timestamp - (are the versions under continuous testing - CD is here)
developers working together can override ext components "v1.1.1" and pick whatever 1.1.2-snapshot to use" - need to see how teams use the staging /snapshot version

fully validated

observation
- teams only have visibility on changes marked in the manifest file - not "in-development" merges
- we will likely need to group commits under test (lack of resources) - developers will need to triage together a test set that marked (n) commits in a timeframe as breaking CD
-

References

- https://lists.onap.org/g/onap-discuss/message/3454
- https://lists.onap.org/pipermail/onap-discuss/2017-August/003454.html (deprecated link- see above)
- https://lists.onap.org/g/onap-discuss/message/3967
- https://lists.onap.org/pipermail/onap-discuss/2017-August/003967.html (deprecated link- see above)