Release Process

Template to set the workflow for any given release. The goal of the template here is to use well known and tested methods, simplify the process, and reduce any ambiguity that we can. This helps to keep the focus on the product where it belongs by taking cognitive load out of managing releases. This template should also not be seen as a final word; if someone has an objectively better method that improves this process then we should generally adopt that.

Some well known and tested methods we will adopt immediately

1. Semantic Versioning http://semver.org/
   - Use of semver scheme for EdgeX individual services
   - Discuss further whether to version the named release (ie, Barcelona)
2. Nexus Repositories (adopted from ONAP project)
   a. **Snapshot repo**: used for merged artifacts. After the committer has performed the code review (+2), has merged the code and the build is successful, the build artifact is within the Snapshot repo. It is expected to have multiple snapshots for a single repo per day. All artifacts have same version number.
   b. **Staging repo**: used for Release candidate. Once a day, a new clean build is automatically performed. All artifacts have same version number. The Staging artifact is used for the release testing (Testing beyond unit tests).
   c. **Release repo**: this is the place where the project team (or Linux Foundation Releng Team) stores the artifacts that are deemed stabled for being consumed by the other project teams. Each team decides when to release. It is not expected to get a new release every day. No TSC approval is required for getting a new release artifact.
3. Dependencies on named release
   - Barcelona for example will contain a number of individual services that can be independently released. We should manifest the particular services and corresponding version of that service that will be contained in Barcelona.

General Development

1. All general work goes against the master branch in each project.
2. Individual Pull Requests will generate artifacts that are pushed to the Nexus snapshot repository.
3. Daily builds will build master and push an artifact into the staging repository, these should be considered release candidates.

Enumerate Services in Release

1. Simple list of all services that will be in the release.

Release Process

1. Branch Cutting
   a. At some point before an official artifact release (at least 2 weeks). A release branch will be created off of the master branch.
   b. The master branch version will be rev'd to the desired version of a future release.
   c. Only bug and security fixes can now make it into the release branch unless truly exceptional.
   d. Jenkins jobs will be instantiated for the release branch.
2. Official Release
   a. A leading representative from a particular project will initiate the release by doing the following:
      i. Submit a Project Service / Release Artifacts ticket via JIRA - https://support.linuxfoundation.org/ - with the following information:
         1. Link to Jenkins staging job that produced artifact to be released
         2. Type of artifacts to be released (jar, go bin, container)
   b. The EdgeX Release Engineer will release signed artifacts as requested and git tag the repository found at the sha checked out in the aforementioned Jenkins staging job.
   c. RE will push the tag to https://github.com/edgexfoundry/$PROJECT_NAME
   d. Project representation will PATCH bump the version of the project in the release branch in prep for next patch.

2. Generate or update named release manifest

   - This is the document which will enumerate each service and version that will be contained in a named release.

3. Individual service release

   - Each individual service can release separately if they wish. If at any time a service is updated then we should also be sure to update the manifest.