Moving CSIT to project repositories

Description

- See Project-specific CSIT structure

Motivation

The most obvious solution for taking full advantage of CSIT is to add the test cases under the same repository with the functionalities that they are testing (instead of having them in separate centralized CSIT repository as we currently have). This would have the following benefits:

- CSIT could be triggered by any commit to the project repo
- CSIT tests the code (or specifically, docker images that have been built) from the committed branch
- CSIT could have a vote on the commit based on the result of the test run
- If the implementation changes require changes in CSIT tests (to cover new functionality or to pass in the first place), that could be handled within the same commit
- ideally, local verification would become less complex (no need to work between CSIT repo and project repo)
- No need for the integration team to merge any changes related to your project

Issues

Given the fact that CSITs are currently a very colourful collection of various suites with different scopes and strategies, the transition of CSITs to project repositories is not necessarily trivial:

- CSIT suites that test components from multiple project repositories at the same time
    - such CSIT tests may have to be separated using additional simulators, or
    - project repository structures themselves may have to be reconsidered, or
    - the possibility of combining branches from multiple repositories under the same commit needs to be provided (if gerrit allows?)
    - In any case, the division between the images *under test* and images that are just necessary dependencies should be clearly made and documented
        - the images under test should be coming from the commit branch
        - in the case of necessary dependencies it must be decided whether they should be provided as simulators or real components
            - if provided as real components, they should be referred to with a fixed, unchanging released version number and should be stable and mature enough to develop on
                - ONAP's (unintentional?) practice of allowing the same versioned image to be changed is problematic
        - project-specific simulators that are built on the fly are trivial case from dependency handling point of view, but if common simulators are in use, dependency considerations for them are the same as with real components
- Jenkins templates have to be redesigned to support unified approach for triggering review branch-specific artifact and docker builds, CSIT execution and voting chain as part of review verification
    - The redesign should also allow testing locally built docker images in local environment with CSIT as effortlessly as possible
- CSITs will become blockers for merging code
    - local pre-commit verification should be supported better by common CSIT tools
    - are all projects and suites mature enough to deal with that?
- Docker image production practices should be unified (see Docker Image Build Guidelines, Independent Versioning and Release Process and Release Versioning Strategy)

Technical decisions

- New templates have been introduced for all verification steps (see Project-specific CSIT structure) while all the existing ones have been left untouched
    - review-verification template
        - triggered by review commit
        - has verify vote on the review
        - triggers artifact builds
            - Should maybe include also Sonar analysis in the future - those are currently still completely separate jobs executed only on merged code
        - triggers docker image build - different types of docker builds each require their own templates (identified and separated by "artifact-type"), and so far we have them for
            - maven docker build
            - golang docker build
            - images *are not* pushed to Nexus3
        - triggers CSIT execution that tests the produced docker images
    - merge-verification template
        - triggered by merge
        - triggers artifact build in the same way as review-verification
        - triggers docker image build in the same way as review-verification but builds the images from master
        - triggers CSIT execution that tests the produced docker images in the same way as review-verification
        - No images are pushed to Nexus3 as a result of merge-verification either; publication of new images still remains the responsibility of separate docker staging etc. jobs
- Execution of CSIT tests and incorporating locally built test images should be made as easy as possible following common guidelines

    - Local docker build instructions should be maintained and easily accessible (for example, as part of CSIT README.md in the project repository)
    - Test environment setup should be as automated as possible (project-specific dependencies should be handled by the setup scripts)
    - Need for specific environment variables (like , for example, GERRIT_BRANCH) should be minimized and preferably avoided altogether
- Artifact builds should take care of code coverage/Sonar analysis
    - apparently currently there are no common templates dealing with Sonar? (instead, all the project have their custom Sonar JJB definition)
    - all the Sonars run on daily schedule instead of being triggered
    - are projects mature enough to define the various Sonar violations as review blockers?
        - can we give each project the power to define their own quality gates?

Project status and readiness at the end of Honolulu

- See CSIT status and readiness per project at the end of Honolulu

Project status and readiness at the end of Guilin

- Moved under CSIT status and readiness per project at the end of Guilin for reference