Integration & Test Strategy for Lithium

Luis Gomez, Brocade

Facts

- OpenDaylight projects are quite heterogeneous: platform, tools, plugins, apps, etc… -> they have different test needs.
- Projects define one or more features -> we have to test them individually and together.
- We leave users freedom to select features -> we need an strategy to test a non-rigid distribution.
- OpenDaylight has a common CI process for all projects -> tests should in general be part of the CI.
- Number of projects duplicates every release -> We need an strategy that can scale.

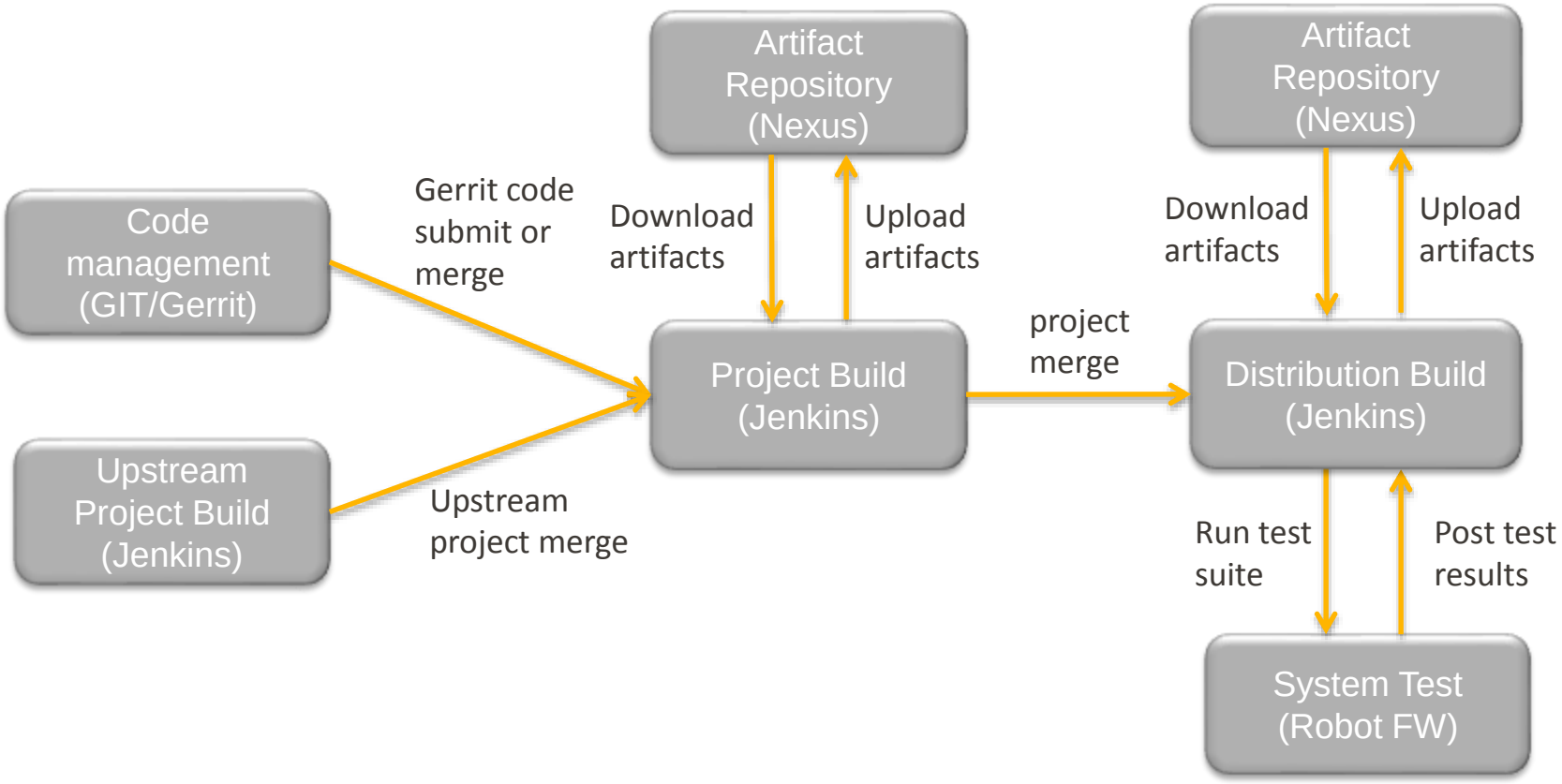
High Level Strategy

- Instead of verifying projects, test user-facing features.
  - Test single feature enabled to identify issues in a feature itself.
  - Test all features enabled to identify feature interferences.
- For a given project, test after:
  - A change in the project code (submit or merge).
  - A change (merge) in an upstream project.
- Be selective with the tests we run:
  - A change in a “stem” project will trigger test on many features.
  - A change in a “leaf” project will trigger test on few concerned features.
- Projects will be responsible and will own their system test.

Distribution Assembly

- We define 2 integration features for test purposes:
  - compatible-with-all -> useful for projects to test their features with other features.
  - all = compatible-with-all + incompatible features -> used to test features/bundles definition in integration.
- Integration features are currently hidden to users.

CI Infrastructure



Build Test

Build test occurs during project code build. Examples are Junit, PAX-EXAM or single feature install tests

Every project in OpenDaylight defines 3 build jobs:

- Verify job -> builds the project code and passes build tests on code submit
- Merge job -> builds the project code and passes build tests on code merge. In addition it will upload generated artifacts to Nexus repository.
- Integration job -> builds the project code and passes build tests after a merge in an upstream project

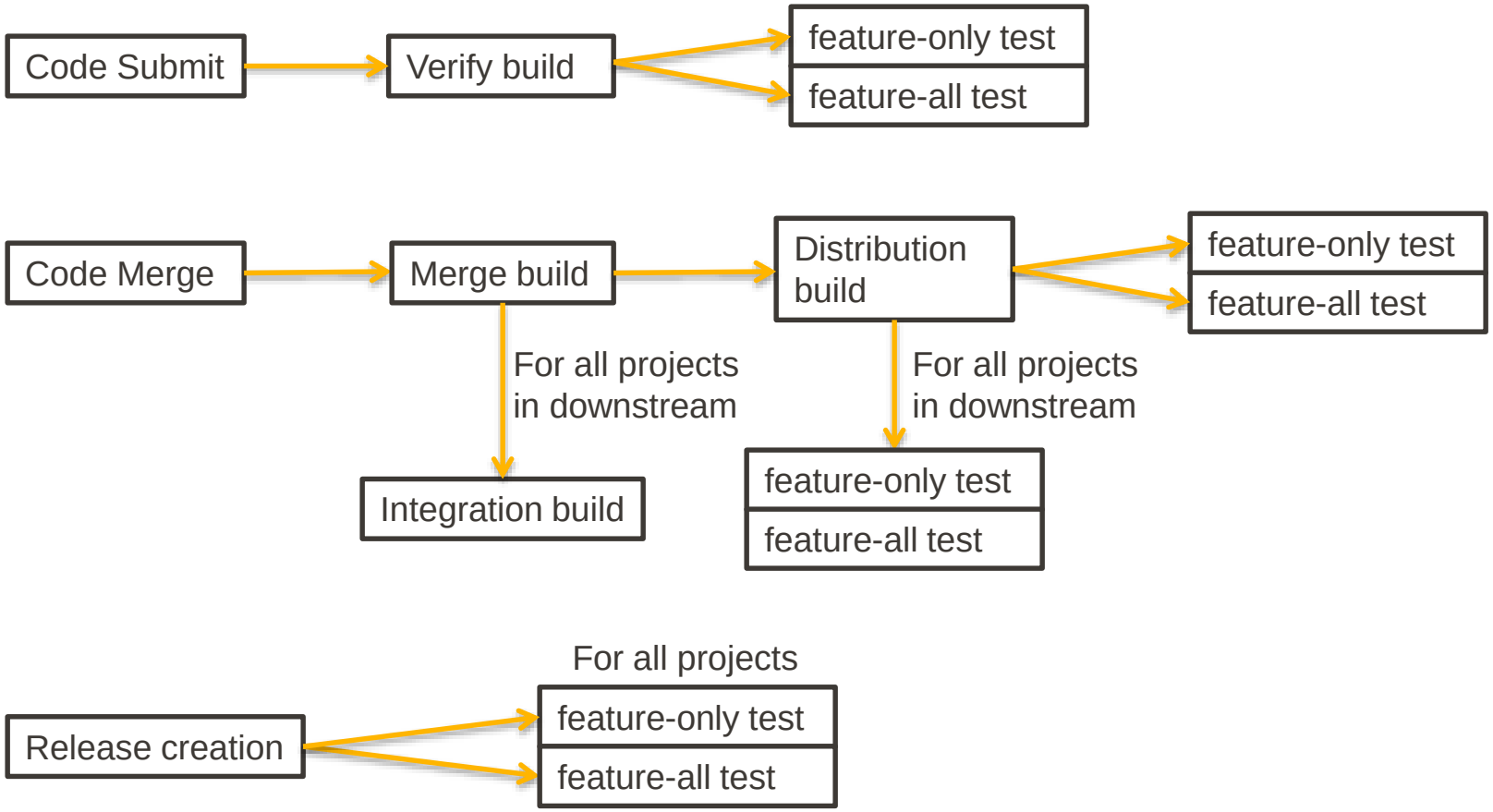
System Test

System test requires a distribution and an automation that installs controller and executes tests against network tools.

Every project in OpenDaylight will define 2 jobs:

- Distribution job -> builds the integration code and therefore the official distribution
- System Test Jobs -> For every project user-facing feature (essential functionality) we will create 2 system test jobs:
  - feature-only test -> Installs single feature with required dependencies and runs functionality planned test
  - feature-all test -> Installs odl-integration-compatible-with-all + feature and runs functionality planned test

Test Triggers



Quality metrics

- Code sanity and code coverage reports are collected daily to Sonar
- All system test code will be stored in the Integration repo, we can quantify the amount of test per project
- Number of critical bugs per project and bugs trends
- Performance numbers