BridgePoint Long Term Support

October 2017

Why LTS Releases?

- Validating releases for production use is hard
  - some users perform their own internal tests of each new release, requiring effort and time
  - LTS releases dramatically limit scope of revalidation
- Avoid bugs that come along with "bleeding edge"
- Concern about regression in existing functionality due to new features
- Limit user confusion about new features and need for retraining

LTS Release Goals

- LTS versions are:
  - stable
  - have a longer lifecycle
  - slow to change: only important bug fixes, or compelling changes

- LTS versions do not change regularly. They are not:
  - feature driven
  - cutting edge

Existing Process

- Repositories
  - *bridgepoint*, *mc*, *packaging*, *pt_antlr*, *bptest*
  - Many developer forks of main xtUML repository
  - Branches: master, issue development, release version
  - Pull requests to master serviced by core committers
- Engineering Builds
  - nightly (master)
  - branch (issue development)
  - limited or no testing
- Releases
  - BridgePoint Pro at least twice a year
  - Full testing (manual and automated)

Ubuntu-like Time Based Approach (Option 1)

- Create a new major LTS version every two years
- LTS versions live for three years
- Provides a one year overlap for users to move from one LTS version to the next
- This approach is a significant deviation from current release process

Debian-like Continuous Multi-branch Approach (Option 2)

- Multiple release and pre-release branches coexist at the same time
- Features and fixes are promoted to *testing* branch
- Eventually *testing* is frozen, undergoes focused testing & hardening, then becomes *stable* branch
  - *stable* is only updated for important security and usability fixes
  - *testing* is then reopened for ongoing development

Development Process Effects

- Managing LTS branches requires oversight and adds overhead to the development process
- Requires tracking of fixes not yet applied to stable
- Candidates for promotion to stable require consultation and review by dev team and customer
- Retest of *stable* after promotion of feature/fix
- This work becomes more difficult over time due to
  - divergences in the code base
  - build infrastructure changes

Proposal

- Modify current development process to add stability-driven, customer-supported LTS branches
- Create LTS branch off BP Pro release branch in response to customer request or community-driven compelling event
- Stakeholders agree on the starting point and collaborate on what changesets are applied to the LTS branch
- Customer requests special release of BridgePoint Pro from the LTS branch

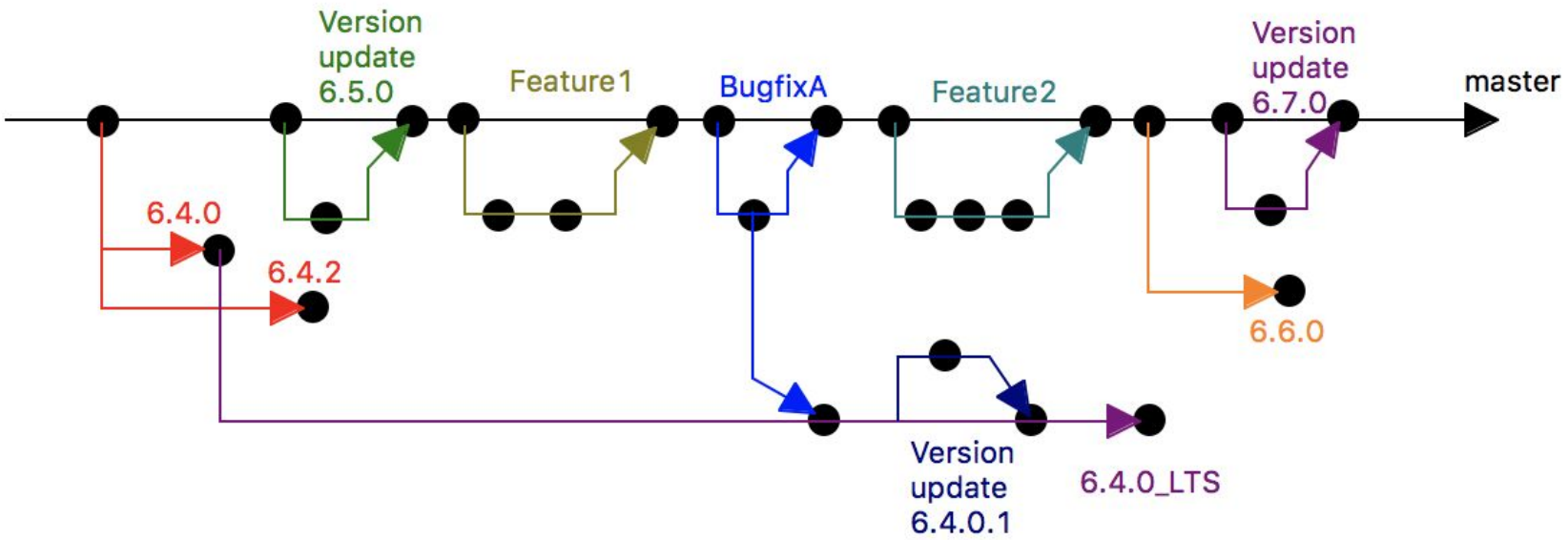
Proposed Branch Management

- *master == testing == experimental*
  - gets new features and fixes from developers
  - we require a measure of functional test as part of the acceptance process
  - users have early access to these engineering builds
  - releases are tagged

- *stable*
  - ongoing protected branch
  - gets promotions from master through due process, including user input and promotion criteria
  - … and after feature has had some mileage in the master branch

Flow of Changesets

- Regardless of the exact strategy implementation, the LTS (*stable*) branch moves forward much slower than *master*



Discussion

- Is the proposed strategy reasonable ?
- Are you interested in using LTS releases versus the current BridgePoint Pro release?

onefact.net