Release for Lithium

George Zhao, Ed Warnicke, Colin Dixon,
Mathieu Lemey, Robert Varga, An Ho

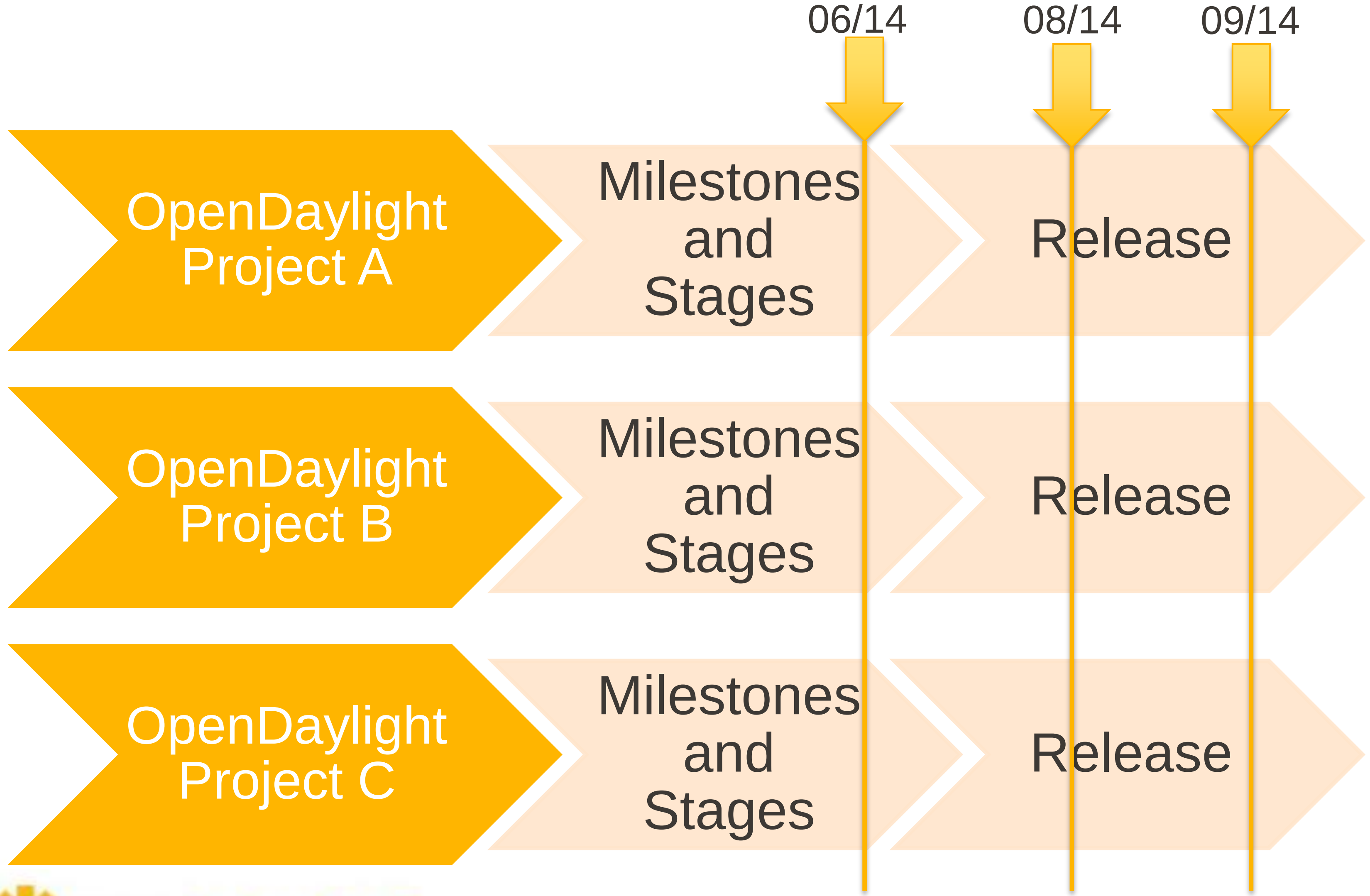
Content

- Opendaylight release (Ed, George)
- Lessons learned (Colin, Robert)
- Proposal for Lithium release
  - Release process, schedule and milestones (George, Robert)
  - Auto-release move to Gerrit? (Ed)
  - A La Carte download portal (Mathieu)
  - Quality of project (An, George)
  - Other discussions (version control, introducing Gradle, Zuul etc)

Opendaylight Release

- Auto-release in Hydrogen
  - Build locally
  - Cannot handle mutual dependencies
  - Use maven version plugin, release plugin and wagon plugin
- Auto-release in Helium
  - auto-release-helium-worker: the main auto release scripts
  - auto-release-helium-test: time control, no publish artifacts
  - auto-release-helium-nightly: publish artifacts
  - auto-release-helium-release: publish to staging, no date suffix

Current Release Pipeline



06/14

08/14

09/14

OpenDaylight Project A

Milestones and Stages

Release

OpenDaylight Project B

Milestones and Stages

Release

OpenDaylight Project C

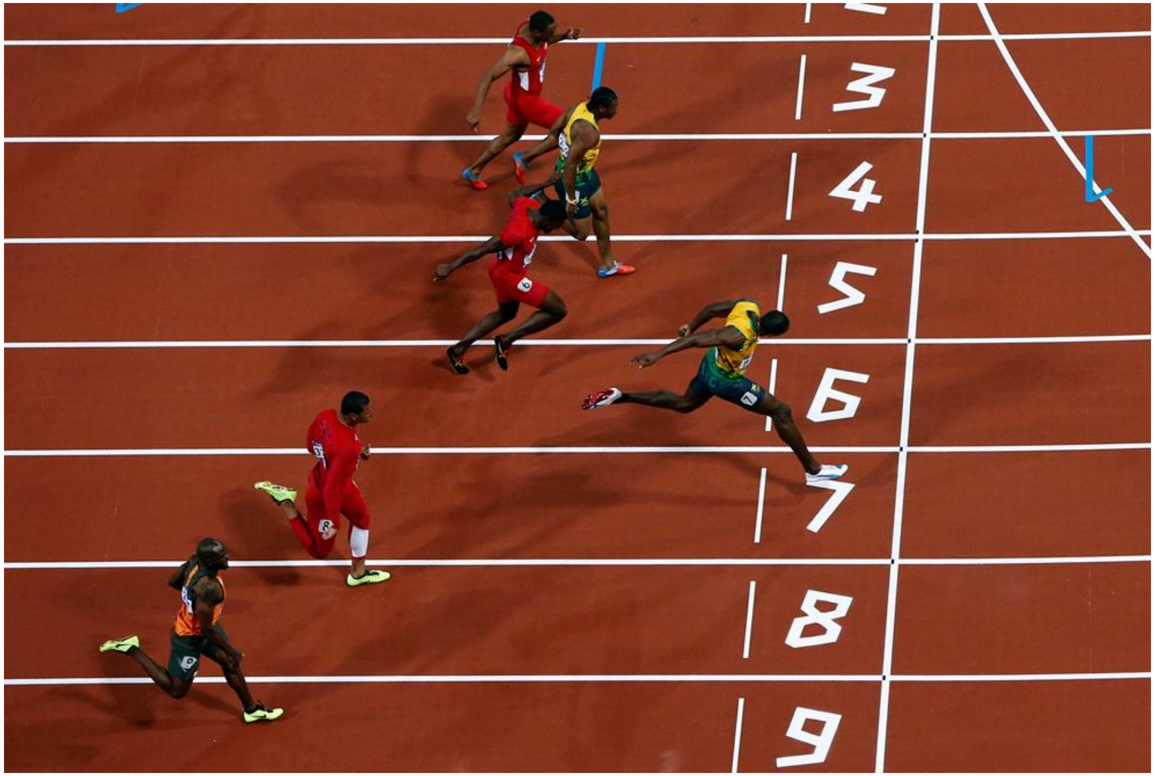
Milestones and Stages

Release

Current Release Pipeline

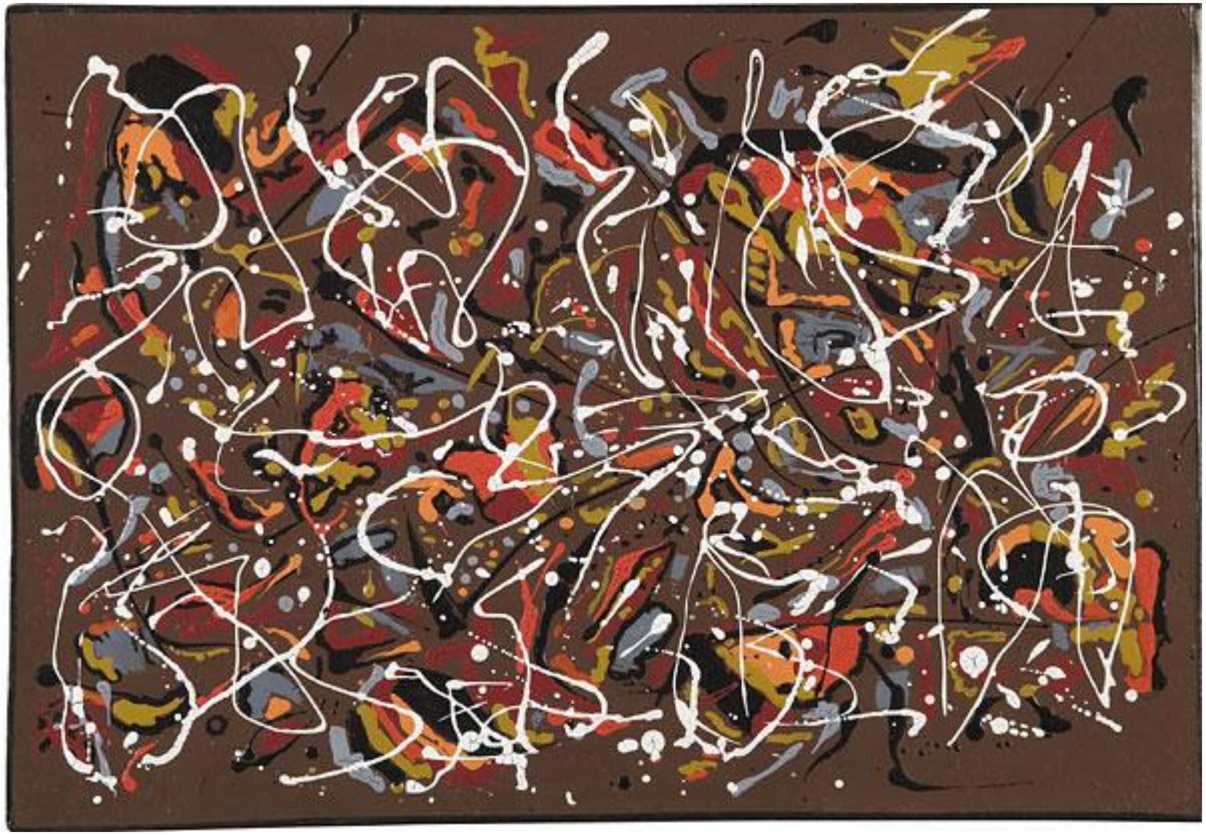
What’s wrong with this pipeline?

Current Release Pipeline



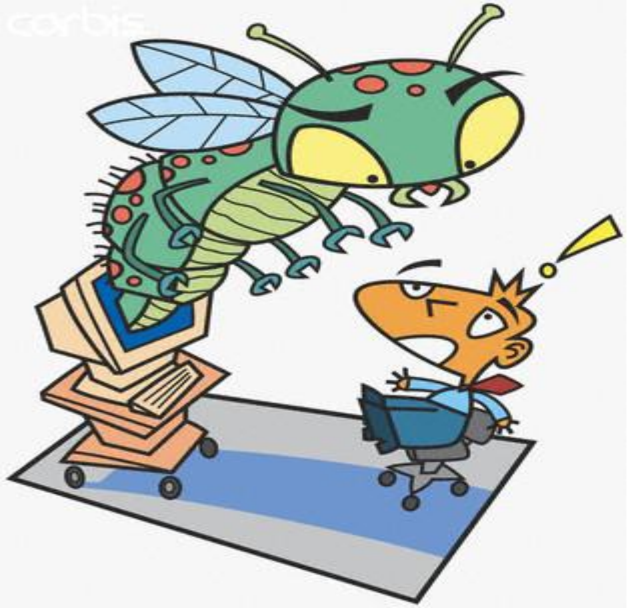
All projects race to finish line at the same time!

Current Release Pipeline



Problem is … Inter-project dependencies some things need to happen before others!

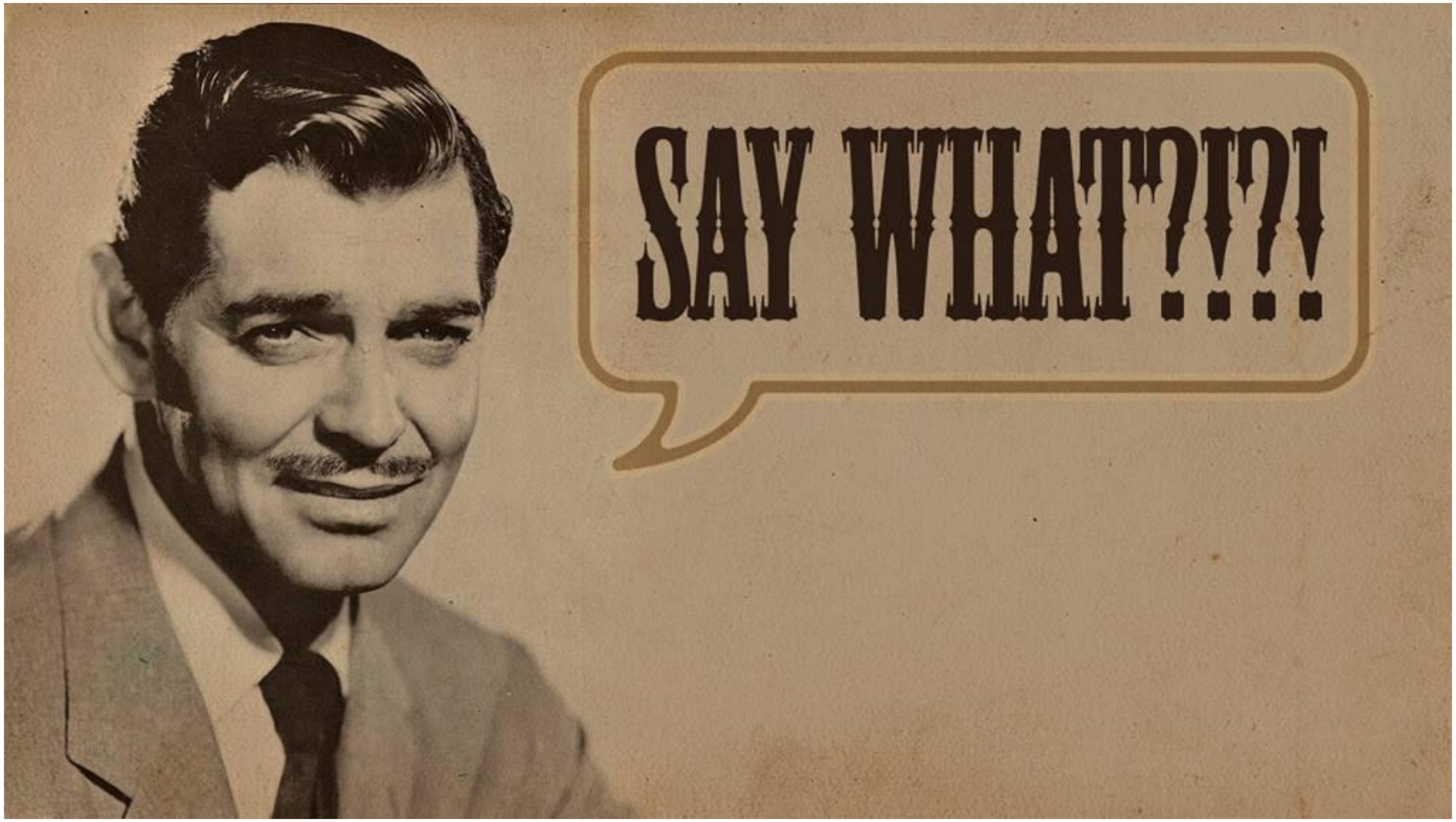
Current Release Pipeline



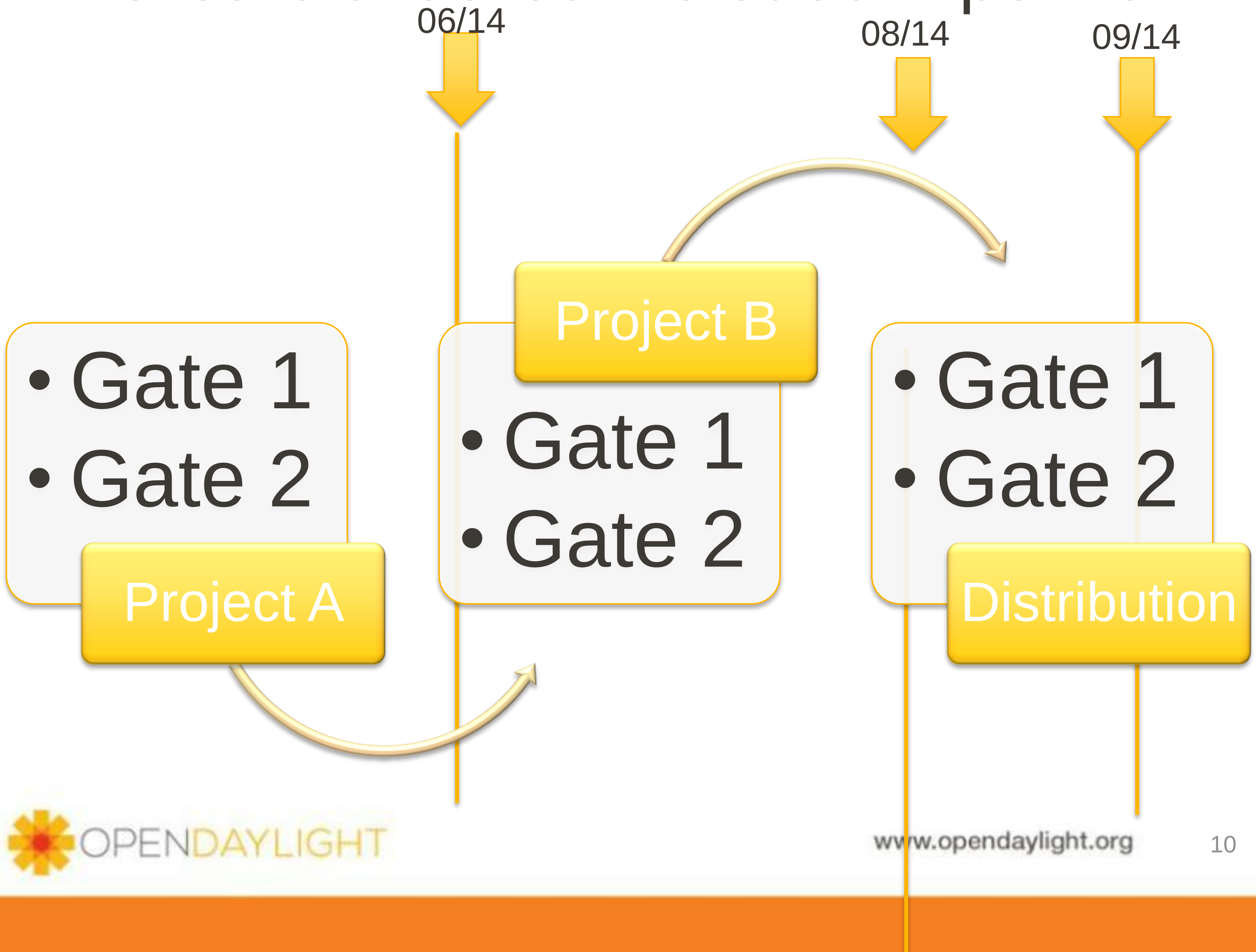
Outcome … cat hearding, pain and… last minute BUGS..

Suggestion

No more simultaneous release!



Create a Gated Release Pipeline



06/14

08/14

09/14

Project A

- Gate 1
- Gate 2

Project B

- Gate 1
- Gate 2

Distribution

- Gate 1
- Gate 2

What is missing?

- Proper Modularity
- Tools to get proper insights on “Gates” (Gates can be documentation, test code coverage, etc.) It should also help classify and promote the projects.
- Continuous Deployment
- Some changes to automated release
- Pipeline management…

Goals

- Fail often and Early (catch CI failures faster for integration work)
- No rush at the finish line
- Less wasted time for everyone
- Better Docs

Lessons Learned

- Details in Opendaylight wiki: Lithium Release Plan
- We should make it clear that participation in Service Releases is not optional
- We should make it clear what we expect in terms of timely responses from project primary contacts for a release
- We need a longer time between code freeze and release candidates because developers don't focus on tests (especially system and integration tests) until after code freeze
- Status reports for each milestone should include more than a Boolean for tests
  - In general, the templates for status reports should probably be developed more in advance.

Problems We Agree On

- Moving to new infra, toolchains, etc. is really painful
  - This time it was karaf, docs, next time it will be…
    - Gradle, new version of maven, etc.
  - Possible solution, is to declare the things we need to use several months out
    - Phil points out that when we add things, we need to either make it optional or push release dates out
  - Need how tos written for this stuff before it's even suggested
- The milestones are broken
  - We can't be taking 3 months to get to the point that we're doing integration testing
  - Maybe we should have it be the case that people are already working on and maybe finishing up the features for the *next* release during this one
- A lot of discussion of "core" code and how to differentiate them from leaf or incubation
  - Only release mature/core code
  - Other projects can release as "ancilliary" parts that tag along
- Service releases are something we've yet to do
- We eally need to make sure core/mature/etc. is not on a project granularity

Lessons Learned (cont.)

- We really need somebody who groks the things that need to be accomplished at each milestone and can take a glance at the code and jenkins jobs for each project to get an idea of whether they're on track or not.
- Requirements to meet at different stages (and especially RCs) should be set and enforced with clearly explained consequences for missing them
- We need to pre-declare when RCs and final release artifacts will be cut (both dates and times for clarity)
- Need to add an EOL-plans section to release plan to understand user impact of EOLed features/components/APIs at the start of a development cycle

Release Process for Lithium

- Feedback and suggestions from lessons learned discussion.
- Longer time between code freeze and release?

  M5 3/23/2015 RC0 3/30/2015
- Stable branch cut at RC0 instead of after release?

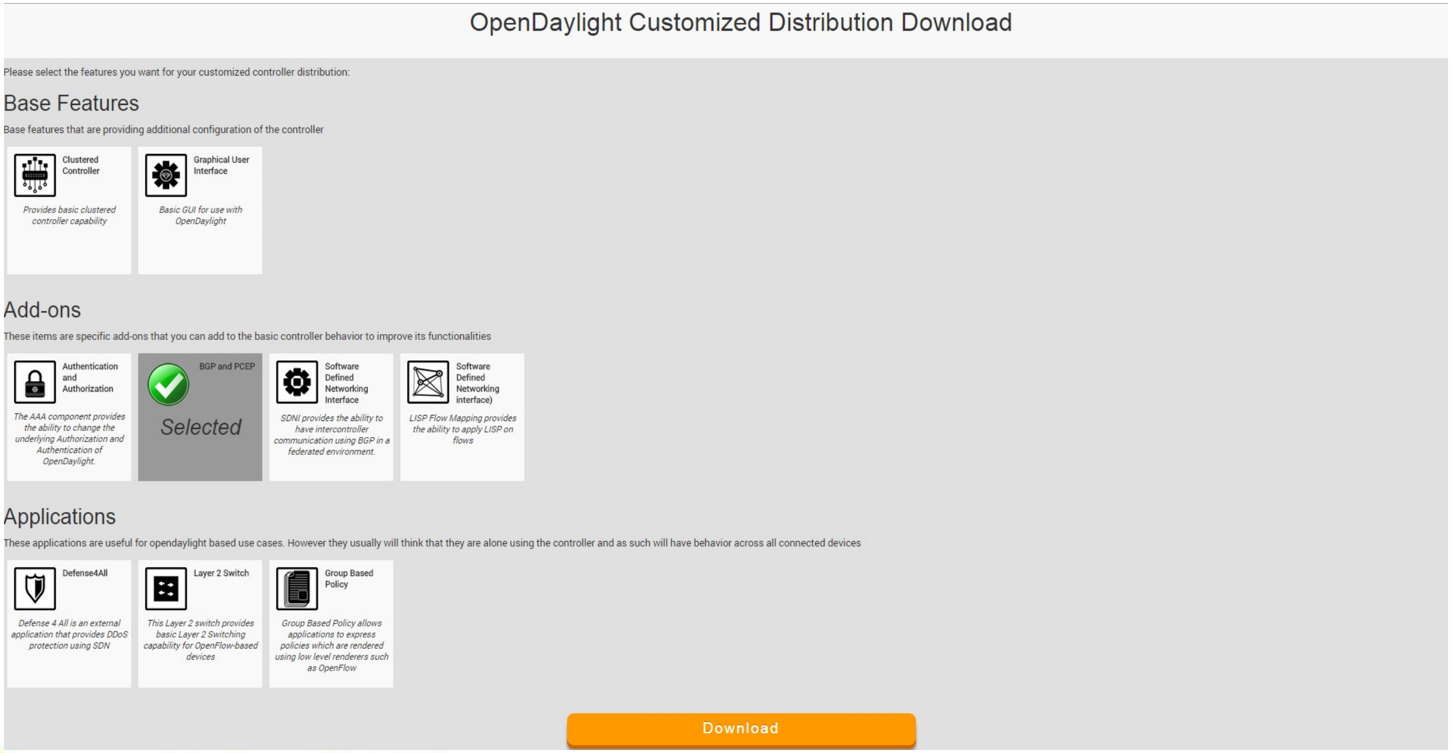
  Currently auto-release will pick up whatever check in HEAD

Autorelease

- Version Control
  - Put autorelease scripts into Gerrit.(?)
  - Scripts modification should go through Gerrit peer review process.
  - Enforce versioning control and release control.
- Tag trigger Release Build
  - Cut stable branch at RC timeframe and tag trigger release verification.
- Regular Builds
  - How and strategy to maintain three regular builds. Hydrogen, Helium and later Lithium.
- Add two VMs to Jenkins/autorelease

A La Carte Download

- Do we put this into release project (github)?
- How do we update and test it automatically?



OpenDaylight Customized Distribution Download

Please select the features you want for your customized controller distribution:

Base Features

Base features that are providing additional configuration of the controller

Clustered Controller

Provides basic clustered controller capability

Graphical User Interface

Basic GUI for use with OpenDaylight

Add-ons

These items are specific add-ons that you can add to the basic controller behavior to improve its functionalities

Authentication and Authorization

The AAA component provides the ability to change the underlying Authorization and Authentication of OpenDaylight.

BGP and PCEP

✓ *Selected*

Software Defined Networking Interface

SDNI provides the ability to have intercontroller communication using BGP in a federated environment.

Software Defined Networking interface)

LISP Flow Mapping provides the ability to apply LISP on flows

Applications

These applications are useful for opendaylight based use cases. However they usually will think that they are alone using the controller and as such will have behavior across all connected devices

Defense4All

Defense 4 All is an external application that provides DDoS protection using SDN

Layer 2 Switch

This Layer 2 switch provides basic Layer 2 Switching capability for OpenFlow-based devices

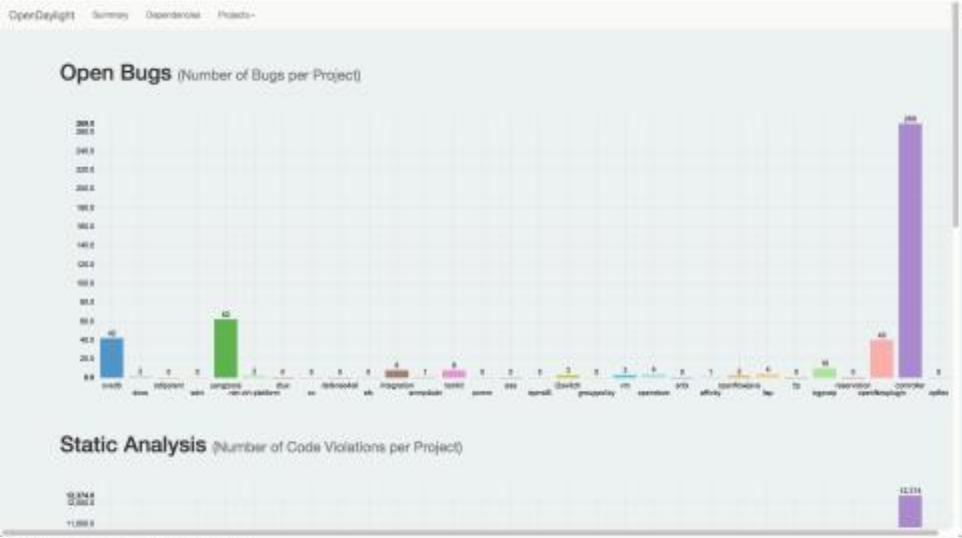
Group Based Policy

Group Based Policy allows applications to express policies which are rendered using low level renderers such as OpenFlow

Quality of Projects

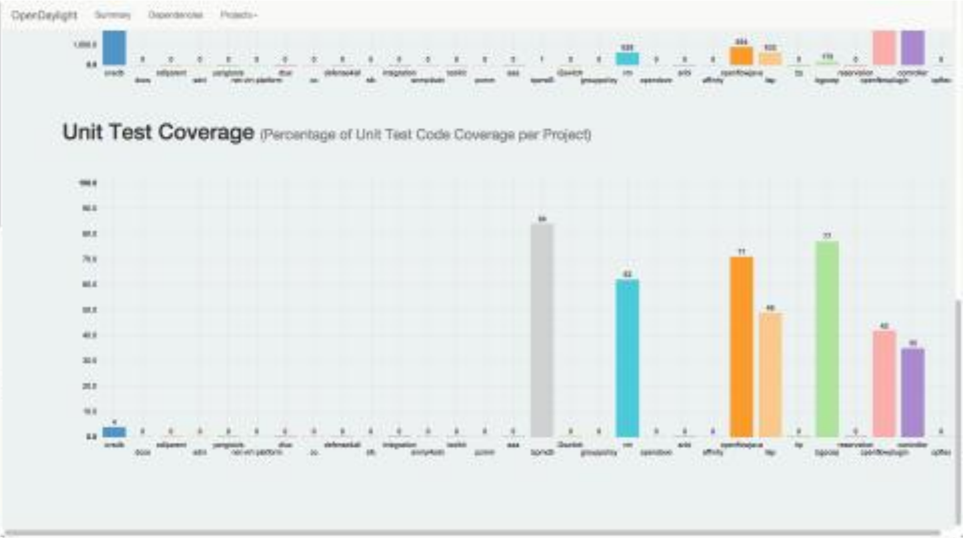
- Portal
  - Have a central web portal with an overview of the project status and release status
  - This should be used for release decision.
- Data Driven
  - Portal should give people an objective fact based summary of project quality based on criteria such as
    - Number of Defects Open
    - Number of Static Analysis Violations
    - Percentage of Code Coverage
- Visualization
  - Portal should display data in a manner that is easily consumed.

Quality of Projects

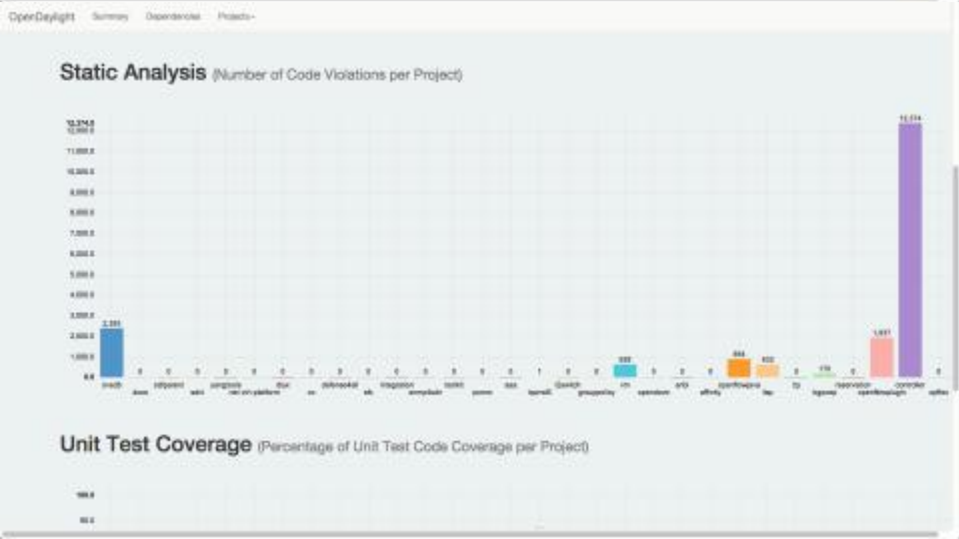


Open Bugs (Number of Bugs per Project)

Static Analysis (Number of Code Violations per Project)

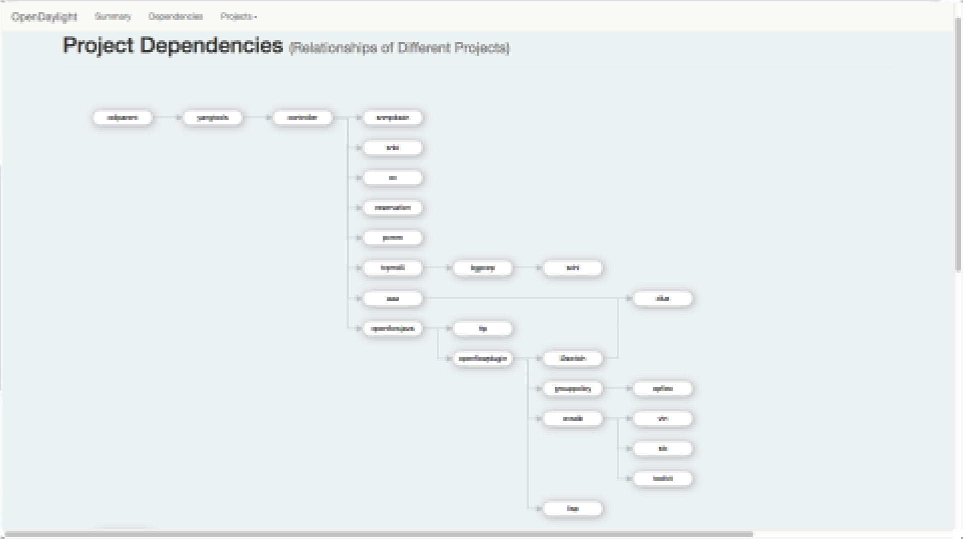


Unit Test Coverage (Percentage of Unit Test Code Coverage per Project)



Static Analysis (Number of Code Violations per Project)

Unit Test Coverage (Percentage of Unit Test Code Coverage per Project)



Project Dependencies (Relationships of Different Projects)

Discussions

- Release project for Lithium? (do we need one?)
- Release milestone dates modification? (if yes, which one?)
- Release policy (template, enforce milestone, declaration…)
- Other

Interested?

- **Contact:** George.Y.Zhao@huawei.com
- **Mailing List:** release@lists.opendaylight.org
- **IRC Channel:** #opendaylight-release

OPEN DAYLIGHT

Thank You

LINUX FOUNDATION
COLLABORATIVE PROJECTS