Moving towards Continuous Integration

ATLAS Software TIM

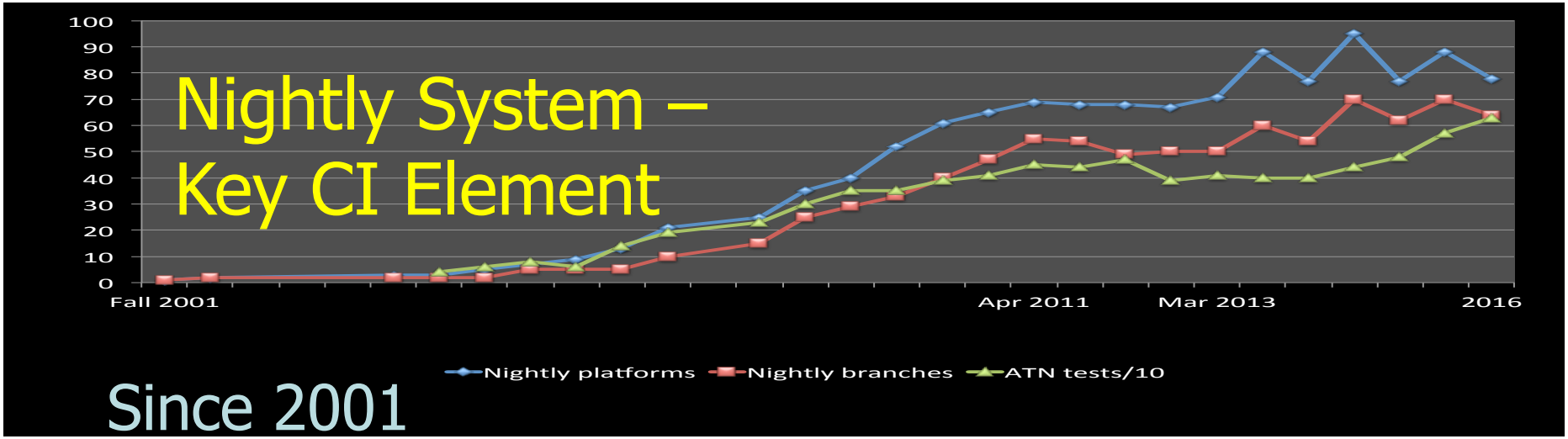
June 8, 2016

Alexander Undrus (BNL)

Continuous Integration (CI) Definitions

- **Continuous Integration** is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible*(http://www.martinfowler.com/articles/continuousIntegration.html, 2006)*
- **Integration** is the act of submitting your personal work (modified code) to the common work area ("Learning Continuous Integration with Jenkins", by Nikhil Pathania, 2016)
- CI was first named and proposed by Grady Booch in "*Object Oriented Design: With Applications*" (1991), although Booch did not advocate integrating several times a day (*https://en.wikipedia.org/wiki/Continuous_integration*)

Nightly System – Key CI Element



Since 2001

Currently:

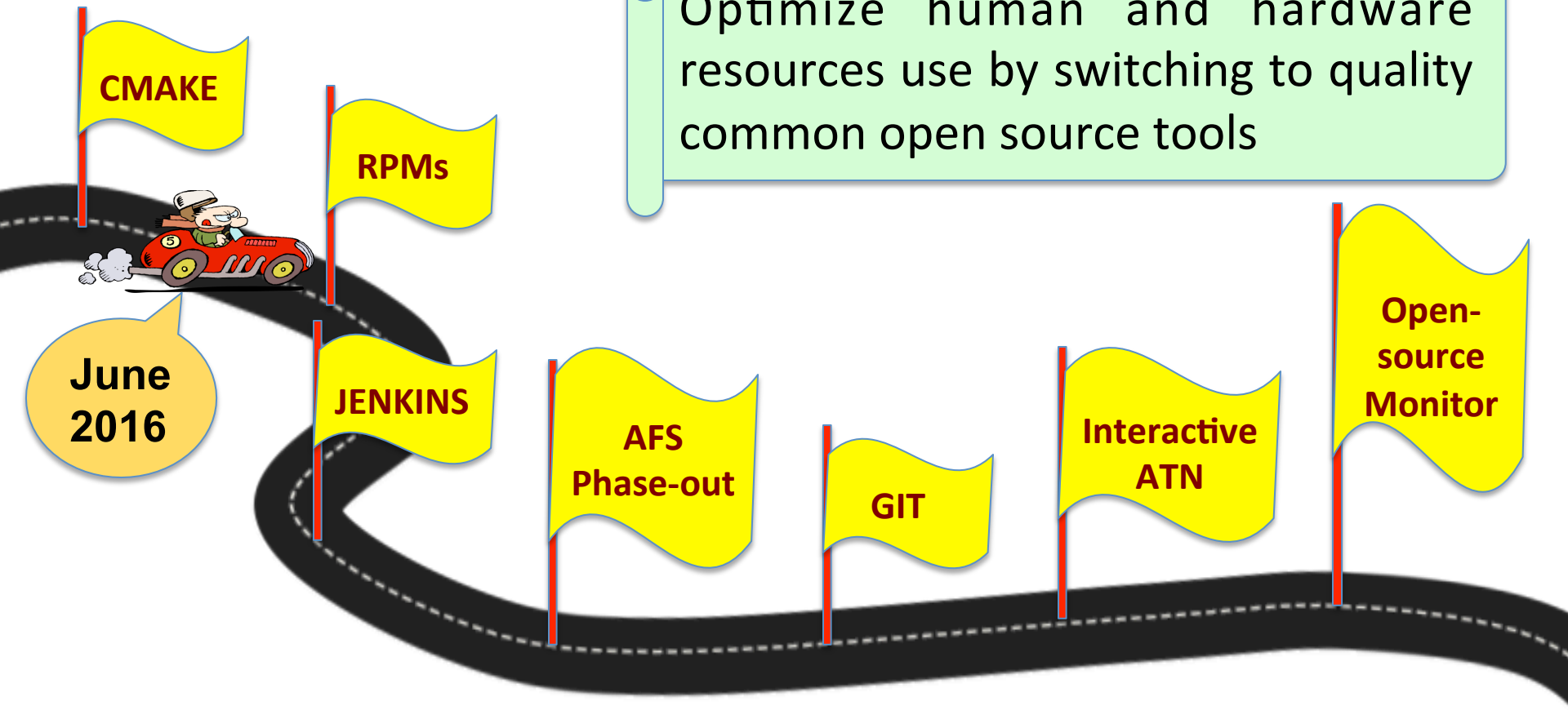
- Developers do integrate their work frequently
  - 60 branches rebuilt daily (most) and on-demand (few)
  - New on-demand functionality is popular
    - 30 privileged coordinators make few daily restarts
- The LS1 upgrade brought the Oracle-based Web UI
  - Flexible error analysis and monitoring
- 100% home made
  - ~ 60 RH and virtual machines (incl. distscc farm)

Missing CI Elements

- Many CI features and procedures are in place
- There are multiple integrations per day
  - In multiple branches with convoluted packages tags sweeps
  - No automatic rebuilds on changes
- Common work area exists (SVN and TC)
  - Complicated tag approval bureaucracy
  - Several releases types ("full" Athena, analysis and simulation, derivation, RootCore releases)
  - Bidirectional traceability is manually supported
- Heavy use of human and hardware resources
  - Substantial SVN and TC admin effort
  - Arduous management of > 2000 packages
  - Fixed job schedule means substantial idle machines time
  - Home-made tools, often without alternative experts

Nightly System Roadmap to CI
(Inspired by the 2015 Build Review)

Optimize human and hardware resources use by switching to quality common open source tools



4-phases Plan of the Nightly System CI Upgrade

- Discussed and approved at Febriary 2016 S&C week, https://twiki.cern.ch/twiki/bin/viewauth/AtlasComputing/AtlasNightliesPlans
  - ① Focus on CMake, Jenkins integration
  - ② Put Jenkins on top of the System
  - ③ Focus on new Build Analytics tool
  - ④ Finalization
- Small, incremental changes, iterations
- Project completion by LS2
- Detailed plan is essential for a success
  - *Success metrics*
  - *Means of implementation*
  - *Milestones*
  - *Workshops, review panel*

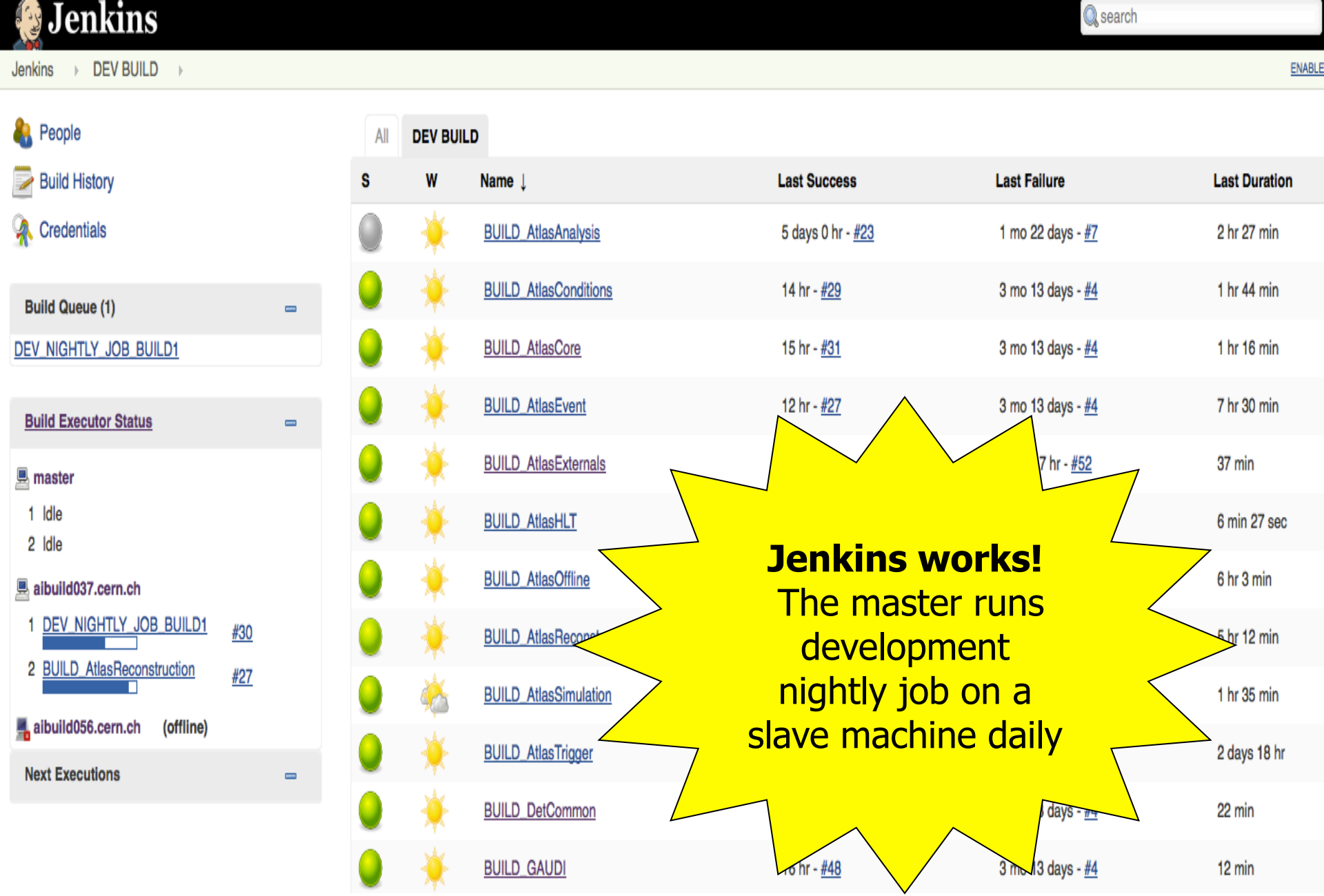
1st Phase Progress (Completion 4Q 2016)

Issues in Epic

ATLINFR-873	establish build time environment for CMAKE builds		OPEN	Shuwei Ye
ATLINFR-966	zsh support in cmake setup scripts		OPEN	Attila Krasznahorkay
ATLINFR-1010	Show CMake configuration and installation logs in the NICOS webpages		RESOLVED	Alexander Undrus
ATLINFR-1011			CLOSED	Alexander Undrus
ATLINFR-1012			RESOLVED	Alexander Undrus
ATLINFR-1013			CLOSED	Alexander Undrus
ATLINFR-1015			OPEN	Emil Obreshkov
ATLINFR-1025			OPEN	Shuwei Ye
ATLINFR-1033			CLOSED	Attila Krasznahorkay
ATLINFR-1060			CLOSED	Alexander Undrus
ATLINFR-1062			RESOLVED	Alexander Undrus
ATLINFR-1065			RESOLVED	Alexander Undrus
ATLINFR-1072	Make ATN execute "make atlas_tests" before running tests in CMake nightlies		OPEN	Alexander Undrus
ATLINFR-1075	Make dev/devval use the production version of AtlasSetup, while CMAKE(-VAL) should use the beta		RESOLVED	Alexander Undrus
ATLINFR-1096	Teach NICOS to notice library loading failures in CMake builds		RESOLVED	Alexander Undrus
ATLINFR-1097	Add package version information to CMake log files		RESOLVED	Alexander Undrus

Priority for NICOS Updates for CMake: (epic ATLINFR-1009):

- Transformation of NICOS workflow for CMAKE builds is completed
- 11 out of 16 tickets closed
- Many remaining tickets are perennial task (environment, rpms ...)



BROOKHAVEN NATIONAL LABORATORY

1st Phase Items under Development

- ATLAS Jenkins master server
  - Works OK (accessible inside CERN only)
  - Starts cmake-based nightly jobs on a slave machine daily
  - Project builds are synchronized by Jenkins (Multijob plugin)
- New nightly release name format (YYYY-MM-DDTTTT instead of rel_N): supported by asetup, a test release is available on nightlies CVMFS
- RPM kits are created and installed locally in cmake-based nightlies, ready for outside-of-AFS installation

Jenkins Caveats

- Key functionalities are provided by plugins
  - ~ 50 active plugins on the ATLAS master
  - Essential "multijob" and "multiplatform" plugins
- Stability is not granted by default installation
  - Automatic updates ruin the master
    - *Update kills all slave jobs*
    - *New Jenkins versions ~ twice a week*
    - *Plugins backward compatibility is not guaranteed*
  - Jenkins configuration backup is required (plugin available)
  - Jenkins slave agent can crash – results in stalled jobs
  - Build timeout plugin is needed to stop stalled jobs
- Jenkins puppet module (from CERN IT) is disruptive
  - Restarts service (killing all jobs) every time puppet runs

Jenkins: Next Steps

- Security scan, opening ports in the firewall
  - Help of ATLAS CSOPS is essential
  - Resolve puppet issue (or disable permanently)
- Achieve high stability for the Jenkins-managed test nightly jobs
  - Less than 1 Jenkins hiccup per month
- Assure optimum synchronization of different build steps (no idle periods)
- Switch few CMake-based 'regular' nightly branches to Jenkins scheduling by December 2016
  - Optionally: new nightly release names 'YYYY-MM-DDTTTT'

Moving Nightlies Installation from AFS

- CVMFS seems like a natural choice
- Existing ATLAS nightlies CVMFS server:
  - can not be regarded as the model for AFS replacement
  - holds few nightly branches used by few users
  - copes with data size and load on it is incomparable with that on AFS
- CVMFS and AFS use cases are different
  - CVMFS is a distribution file system
  - Better solution is needed to allow several users trigger installations and publications on a CVMFS server
- Solutions for CVMFS nightlies installations are explored in ATLINFR-1050
  - Several servers with a common base path
  - **NOT FREE: additional support effort seems inevitable**

2nd Phase (completion 4Q 2017)

- Continue to add cmake-based builds to Jenkins
- Retire RH build machines
- Use VM SSD machines for builds
  - All with the same configuration
  - Reduce number of CPUs ~ 30% at the build farm
- Automate creation of nightly jobs for new nightly branches (possibly with a special Jenkins plugin)
- Keep the Nightly System in sync with ATLAS transition from SVN to git
  - Transition schedule with dated milestones would be very helpful
  - Evaluate Jenkins plugin for git SCM (the Report R.2)
- Evaluate incremental capabilities of CMake builds

3rd Phase (completion 4Q 2018)

- Evaluate and choose an open source tool for nightlies monitoring
- Move cmt and RootCore based builds to Jenkins
- Support git SCM as needed
- Deliver a beta version of the new ATN framework allowing testing on remote farms and in individual developers sessions

4th Phase (completion by LS2?)

- Remove dependence of the ATLAS Nightly System on AFS
- Replace the NICOS Web UI with open source monitoring tool
- Finalize the support of the workflow based on git SCM
- Deliver beta-version of the new ATN framework

Conclusions

- Good progress achieved in the ongoing 1st phase of the CI Nightly System upgrade
  - The Nightly System is transformed for cmake-based builds
  - ATLAS Jenkins instance is created and works in test mode
  - Jenkins instabilities are investigated and addressed