A Roadmap to Continuous Integration for ATLAS
Software Development

J Elmsheuser\textsuperscript{1}, A Krasznahorkay\textsuperscript{2}, E Obreshkov\textsuperscript{3}, A Undrus\textsuperscript{1}

on behalf of the ATLAS Collaboration

\textsuperscript{1} Brookhaven National Laboratory, USA
\textsuperscript{2} CERN, Switzerland
\textsuperscript{3} University of Texas, Arlington, USA

E-mail: undrus@bnl.gov

Abstract. The ATLAS software infrastructure facilitates efforts of more than 1000 developers working on the code base of 2200 packages with 4 million C++ and 1.4 million python lines. The ATLAS offline code management system is the powerful, flexible framework for processing new package versions requests, probing code changes in the Nightly Build System, migration to new platforms and compilers, deployment of production releases for worldwide access and supporting physicists with tools and interfaces for efficient software use. It maintains multi-stream, parallel development environment with about 70 multi-platform branches of nightly releases and provides vast opportunities for testing new packages, for verifying patches to existing software and for migrating to new platforms and compilers. The system evolution is currently aimed on the adoption of modern continuous integration (CI) practices focused on building nightly releases early and often, with rigorous unit and integration testing. This paper describes the CI incorporation program for the ATLAS software infrastructure. It brings modern open source tools such as Jenkins and GitLab into the ATLAS Nightly System, rationalizes hardware resource allocation and administrative operations, provides improved feedback and means to fix broken builds promptly for developers. Once adopted, ATLAS CI practices will improve and accelerate innovation cycles and result in increased confidence in new software deployments. The paper reports the status of Jenkins integration with the ATLAS Nightly System as well as short and long term plans for the incorporation of CI practices.

1. Introduction

ATLAS (A Toroidal LHC Apparatus) \cite{ATLAS} is one of the largest collaborative efforts ever attempted in the physical sciences. The ATLAS software and computing systems deal with data volume exceeding 200 PB processed on up to 300k cores around the world. The current code development activity is focused on efficient use of new computing vector-processing and multi-threading architectures. The ATLAS Software Infrastructure copes with 7000 – 8000 monthly code update commits from hundreds of software developers. It includes tools and services for software storage, code approval, builds, and validation. The developer workflow optimization is crucial for ATLAS code developments which are often parallel and long. Throughout 2016 there were periods with 70 parallel development streams, each served by a nightly build branch. The merges of numerous long parallel developments are difficult, time and effort consuming. Continuous Integration (CI), which was first named and proposed by Grady Booch in ”Object Oriented Design: With Applications” in 1994 \cite{Booch}, calls for frequent integrations of personal work with
frequent validations by automated build and test systems. Modern CI-oriented open source tools, such as GitLab [3] and Jenkins [4], support pull request workflows, provide fast evaluation of the code base health, reduce integration problems allowing to achieve goals rapidly, lower human and hardware cost of code development. The paper describes the current ATLAS Software Infrastructure and outlines plans for CI practices adoption.


Current ATLAS collaborative software infrastructure provides vast opportunities to consider, make and test code changes. The ATLAS Nightly System Framework facilitates coordination between several hundred software developers working around the world and around the clock [5]. The central part of this Framework is the Nightly Build System managed by NICOS Nightly Control Tool [6]. The ATLAS nightly build system supports up to 70 nightly release branches of different types (percentage of each type is indicated in parentheses):

- Major integration (∼7.5%)
- Major validation (∼7.5%)
- Probing new architectures and compilers (∼10%)
- Testing new versions of external tools (∼10%)
- Personal, assigned to advanced developers (∼15%)
- Patches validation (∼25%)
- Physics analysis tools development (∼25%)

Nightly jobs are scheduled daily at fixed time, with manual restart option available. An ATLAS software release [7] comprises a large number of packages with specific version tags stored in the Tag Collector [8] web interfaced database application. Developers are able to interactively select the version tags from the ATLAS SVN code repository for the nightly releases. ATLAS Nightlies web user interfaces provide dynamic information about the nightly system status, build and test results, allow release coordinators restarting nightly branches with a button click. The key system components are the Nightlies Oracle Database and Nightlies Web Server [9]. The Nightlies Database stores nightly jobs data and serves as a mediator between the Nightly and other ATLAS systems. The ATLAS Nightlies web server is an Apache server managed by CERN IT. It is powered by the Panda Web Platform [10], which supports Python plugins capable of accessing data, generating and publishing both the web content as well as user interface. The JavaScript front-ends are powered by the JQuery([11])-based ThemeRoller web application [12] which provides web theme designs with consistent look and feel.

ATLAS nightly releases are rebuilt on 1 to 4 platforms every day for each branch (in some cases several times for day) on the ATLAS nightly computing farm at CERN equipped with 60 powerful multi-core nodes. Builds are accelerated by file-level and package-level parallelism and by running tests in parallel. The largest builds take up to 9 hours. ATLAS nightly releases are installed on AFS and CernVM-FS [13] distributed file systems for worldwide access. The CernVM-FS is a fuse-based http, read-only file system which guarantees file de-duplication and on-demand file transfer with caching, scalability and performance. Nightly releases are kept for 2 to 7 days. When certain development goals are achieved, the successful nightly release is transformed into the stable release by the team of ATLAS offline release shifters. Stable releases have unique numeric identifiers and indefinite lifetime.

The Nightly System is connected with ATN [14] and RTT [15] testing scaffolds that run tests of different granularity levels. The ATN test tool is embedded within the Nightly System and launches tests concurrently with compilations for faster results delivery. As a fast feedback to developers is one of the most important functionalities of nightly systems, NICOS automatically
posts the information about the progress of nightly builds and tests, identifies problems, and
creates the summary web pages reflecting the system status. Automatic e-mail notifications
about problems are sent out to responsible developers.

3. ATLAS CI Incorporation Program

The current workflow system is based on 24-hour nightly releases cycle and deals with many
components (Tag Collector, SVN, AFS) and manual steps. The large number of nightly branches
results in a heavy load on the build farm machines and operators. CI mechanisms provide faster
feedback to developers and accelerate development cycles by building nightly releases early and
often, rational hardware resource allocation, rigorous unit and integration testing. ATLAS CI
incorporation program is based at and takes an advantage of CI-friendly open source tools, such
as CMake, Jenkins, GitLab.

3.1. CMake
At the beginning of 2016 the transition to a CMake-based [16] build system started. With
somewhat faster full builds, and much faster partial builds, especially when using Ninja [17] as
the build tool, this makes it possible to run CI tests more frequently.

3.2. CTest
CTest is a unit test framework distributed as a part of CMake. It is used extensively in the
ATLAS offline code to set up unit tests for the software which can be run conveniently to check
the success of an integration attempt.
Tests can be assigned labels. Since running every single test in the offline software takes
many hours even on a fast machine, only a fraction of the tests will be labeled to run as part
of CI. It is also under evaluation to develop a system that would allow us to run all of the tests
defined for the packages for which some build operation took place as part of the CI attempt.
Such system allows to get finer grained results for the part of the code that was most affected
by the update in question.

3.3. CPack
CPack is a powerful, easy to use, cross-platform software packaging tool distributed with CMake.
The new CMake based build system of ATLAS makes use of it to build RPM packages of the
software that can be installed very conveniently, with just a single command.
RPMs generated from the full offline build are stored on CERN’s EOS system, and served
through a dedicated webserver allowing the installation of the software in a straightforward way
anywhere in the world using ATLAS’s own extension to yum [18], ayum [19].

3.4. Jenkins
Jenkins is an open source CI and build automation tool with multiple benefits:
• Various build triggers: cron expression, SCM (source code management) poll, build after
  other projects are built, posting to URL
• Feature-rich web user interface for triggering and configuring builds, managing resources,
  adding plugins
• Broad community that supports over 400 Jenkins plugins coming with detailed built-in
  documentation
• Load optimization for build nodes
• Archive of build results and node performance
The ATLAS Jenkins master server is set, integrated with CERN single sign-on authentication, with a hot spare for increased reliability. It triggers jobs on 50 slave nodes. All ATLAS nightly jobs are expected to be scheduled in Jenkins by the end of 2016.

3.5. Git, GitLab

Git [20] is an open source distributed version control system. The migration of ATLAS software code repository from SVN to Git is planned in 2017. The Git-based workflow models match ATLAS needs for handling multi-stream development, fits well in CI Jenkins-based build system, simplifies the creation and improves quality of stable software releases. ATLAS uses the GitLab [3] central repository service at CERN which also provides web based code management.

3.6. CDash

CDash [21] is an open source, web-based software testing server. CDash aggregates, analyzes and displays the results of software testing processes. It is capable to collect results of builds and tests executed in Jenkins and publish them on the Web. CDash integrates well with CMake, CTest, and CPack tools. It is being evaluated for the replacement of the ATLAS Nightlies web user interfaces, promising substantial improvements in monitoring and reduction of support effort.

4. Conclusion

Over the last decade the ATLAS Nightly System served as a major tool in the ATLAS collaborative software organization and management schemes. Adoption of CI practices and replacement of aging System elements with modern open source tools such as Jenkins and GitLab will allow rationalizing hardware resource allocation and administrative operations, providing improved software development workflow for developers, accelerating innovation cycles and increasing confidence in new software deployments. It will further improve ATLAS capability of sustaining increasing numbers of developers and groups testing demands.

5. Acknowledgements

The authors wish to thank members of the ATLAS Software Infrastructure and Git teams for much valuable advice and useful discussions. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of High Energy Physics.

References

[6] Undrus A 2003 Proc. Int. Conf. on Computing in High Energy and Nuclear Physics CHEP03 (La Jolla, USA) eConf C0303241 pp TUJT006 (e-Print hep-ex/0305087)
[10] https://twiki.cern.ch/twiki/bin/view/PanDA/PandaPlatform


[17] https://ninja-build.org


[19] https://gitlab.cern.ch/rhauser/ayum
