Modernising ATLAS Software Build Infrastructure

ACAT 2017
21-25 August 2017, Seattle

Elmar Ritsch (CERN)
for the ATLAS Collaboration
Overview

• The Past
• Review of the Past Infrastructure
• The Solutions
• The Migration Process
The Past: Version Control

- > 2000 packages in **Subversion**
- **Package-level** access control
- **Custom-made tools**
  - *ATLAS Tag Collector* to define list of package tags for each nightly and release
  - *ATLAS SVN Admin* to manage access control lists

### Subversion Repository

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Rev</th>
<th>Age</th>
<th>Author</th>
<th>Last Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnalysisBase</td>
<td>805319</td>
<td>5 weeks</td>
<td>atagcol</td>
<td>Tag Collector commit</td>
<td></td>
</tr>
<tr>
<td>AnalysisSUSY</td>
<td>731690</td>
<td>15 months</td>
<td>atagcol</td>
<td>Tag Collector commit</td>
<td></td>
</tr>
<tr>
<td>AnalysisTop</td>
<td>806778</td>
<td>2 weeks</td>
<td>atagcol</td>
<td>Tag Collector commit</td>
<td></td>
</tr>
<tr>
<td>AsgExternal</td>
<td>806792</td>
<td>2 weeks</td>
<td>krumnack</td>
<td>Tag Collector commit</td>
<td></td>
</tr>
<tr>
<td>AtlasAnalysis</td>
<td>274940</td>
<td>7 years</td>
<td>atagcol</td>
<td>Tag Collector commit</td>
<td></td>
</tr>
<tr>
<td>AtlasAnalysisRelease</td>
<td>768813</td>
<td>10 months</td>
<td>atagcol</td>
<td>Tag Collector commit</td>
<td></td>
</tr>
<tr>
<td>AtlasAnalysisRunTime</td>
<td>192526</td>
<td>8 years</td>
<td>dquarrie</td>
<td>AtlasAnalysisRunTime7-00</td>
<td></td>
</tr>
</tbody>
</table>

### ATLAS Tag Collector

Dependency package version tree of release 20.20.0 for AtlasOffline

<table>
<thead>
<tr>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.20.0</td>
<td>switch to simple tree</td>
</tr>
<tr>
<td>AtlasCommonPolicy-00-00-55</td>
<td></td>
</tr>
<tr>
<td>AtlasCxxPolicy-00-00-77</td>
<td></td>
</tr>
<tr>
<td>AtlasFortranPolicy-00-00-81</td>
<td></td>
</tr>
<tr>
<td>BeamPipeGeoModel-00-00-19</td>
<td></td>
</tr>
</tbody>
</table>
The Past: Building

• Use **CMT** to
  – Define dependencies between packages
  – Generate Makefiles

• **Custom build system** to schedule, execute and distribute daily nightly builds
  – Use distcc and ccache to speed up builds

Configuration Management Tool (CMT)

```plaintext
package AthenaPoolServices

author Marcin Nowak <Marcin.Nowak@cern.ch>

use AtlasPolicy     AtlasPolicy-*      External
use GaudiInterface  GaudiInterface-*   External
use AtlasPOOL       AtlasPOOL-*       External
use DataModelRoot   DataModelRoot-*   Control
use AthenaBaseComps  AthenaBaseComps-* Control
use AtlasROOT       AtlasROOT-*       External

apply_pattern declare_joboptions files="*.py"
apply_pattern named component library library=AthenaRootStreamerSvc
```
The Review

- **Internal review** of ATLAS offline software and build infrastructure at the end of 2015
  - With external experts

- **Custom tools and workflows**
  - Developers have to learn them specifically for ATLAS
  - Have to be maintained by ATLAS

- **Have to wait 1+ day** to see build and test result

- A broken nightly build
  - Is annoying for developers if they wanted to test / build on top of their latest changes in this nightly build
  - Can delay a release by at least a day since we’ll need to wait for the next successful nightly build

- Rely on developers that changes are sufficiently tested by them to **not break the nightly for everyone**
Example

• Adding a change that involves modifications to multiple packages (e.g. interface change) is tedious
  – Access rights to all packages that need modification required
  – Resulting set of Subversion tags needs to be put into the nightly in one go to not break the nightly
  – Difficult to review
The Solutions

- Move build configuration from CMT to **CMake**
- Move version control from Subversion to **Git**
- Establish **ATLAS Flow** as the main development workflow
  - Heavily based on GitLab Flow
- **Code Review** for all changes
  - Using GitLab
- **Continuous Integration Tests** for all changes
  - Using Jenkins
- Create **public Offline Software Documentation webpage**
  - Searchable through Google
Migration from CMT to CMake

• **Custom migration** script to convert CMT requirements files to CMakeLists.txt for every package
  - Very few exceptions needed to be migrated/adjusted manually
• Features migrated from CMT to CMake:
  - **Build RPMs**
  - Building of **lightweight releases (“flavors”)** containing only a subset of all packages
    • Analysis, Detector simulation
Migration from Subversion to Git

- Available migration scripts **not suited** for structure of ATLAS offline software repository
- **Custom migration scripts** developed to move code to Git
  - Only migrated historical package versions **that were in a release** in the last two years
  - Current development branches migrated
  - **Dropped** no longer used packages
  - **Dropped unused files** >100 KiB
  - Put Copyright statements in source files
- Repository size decreased by more than a **factor of 200x**
  - From **47 GiB** in Subversion to **~220 MiB** in Git
    - Working tree ~640 MiB at the moment
- Created **custom git-atlas command** to allow developers to do sparse checkout of specified packages only
ATLAS Development Workflow

- Almost same as **GitLab Flow**
  - Exception is cherry-picking
- **Multi-stage code review** for every merge request
  - Distributes knowledge of codebase within collaboration
- Difficult to apply “upstream first” policy
  - Divergence of code
  - Bug reports often only reproducible in release branches
- Automatically “sweep” (=cherry pick) changes from release branches to master branch

Continuous Integration

- Dedicated build machines
- Incremental builds
  - Reusing build directories to avoid re-building the entire project every time
- Build + test time typically 3h-5h for the entire CI pipeline
- Build and test results posted on GitLab Merge Request page
Nightly Builds

• **Full build** every night
• All release branches + master branch are built
  – Building 7 different branches currently
• Builds for **multiple platforms and 7 different “flavors”** (Simulation, Analysis, ...)
• Version is **tagged in Git repository**
• Generate **RPM packages**
  – Allows us to deploy every build as an official release
• **Binaries installed** on network filesystem
  – Accessible to all developers
Build and Test Results Dynamic Display

- **ATLAS custom tool**
- Performs **error analysis for build and test results** and stores results in Oracle database
- **Web application** dynamically displays information from the database
- Summary displays of **different levels** (system, branch, specific release)
- Used for **continuous integration builds and nightly builds**
- **Improves usability** as developers don’t need to search through raw build and test log files (15+ MiB)
CI Pipeline Interacts with GitLab Merge Request

Add watchers

Add labels

Build and test result summary

Full details available at NICOS MR-3282-2017-07-11-20-23
For experts only: Jenkins output [CI-MERGE-REQUEST 5115] (for remote access see instructions in Jenkins section here)
Code Review & Automated Sweep to master

Review comments and labelling

- Carl Suster @csuster commented a day ago
  - This looks clear and well-tested, and the build looks ok so I'll approve.

- Carl Suster @csuster added `review-approved` and removed `review-pending-level-1` labels a day ago

- Hass AbouZeid @oabouzei merged about 21 hours ago

- Hass AbouZeid @oabouzei mentioned in commit 99f619e5 about 21 hours ago

- Atlas Nightlybuild @atnight added `sweep:done` label about 11 hours ago

- Atlas Nightlybuild @atnight commented about 11 hours ago

  **Sweep summary**
  success:
  
  - master
Tutorial Sessions

• Prepared the collaboration for the new workflow and tools before the change to the infrastructure was made

• Organized multiple Git tutorials before moving the codebase to Git

• Organized multiple code reviewer tutorials before adopting the new workflow
Public Documentation Webpage

ATLAS Software Git Workflow

Introduction
These web pages describe how to work with ATLAS offline code hosted in the git version control system.

If you have no experience with git at all, it’s worth checking out our collection of links in the Help with git section before anything else. There you’ll find a getting started guide for git itself and a leg up from SVN.

As well as using git as the source control management program, ATLAS uses the GitLab service at CERN to provide web access to all of its source code. All changes are reviewed and automatically deployed through a continuous integration & deployment pipeline.

- Collection of tutorials and guides
  - E.g. Development workflow, Code review guide, Release coordinator guide

- Public and Google-indexed
  - More likely that developers will find right page through search as opposed to internal TWiki

- Source in Git repository, changes are reviewed and automatically deployed through continuous integration & deployment pipeline
Summary

• Moved much of our software infrastructure from custom tools to well established third-party tools
  – CMake, GitLab, Git, Jenkins

• Established code reviews and continuous integration tests as part of the standard workflow

• Public documentation webpage

• Held tutorials to prepare the collaboration for the new workflow and tools
The Human Factor

- Migration mainly driven by a few very engaged people
  - High "bus factor"
- Somewhat less resistance from the rest of the collaboration than initially expected
  - Most developers are eager to learn de-facto standard tools that are new to them (e.g. Git)