ATLAS offline software

ePIC Committee on Collaborative Software Development Guidelines and Policies, Meeting #4

3 October 2025

Johannes Elmsheuser (BNL)

With some material from Alex Undrus (BNL)
Slides updated/extended from talks at ATLAS/CMS/LHCb/ALICE joint session in Feb 2025

Introduction - setting the scene

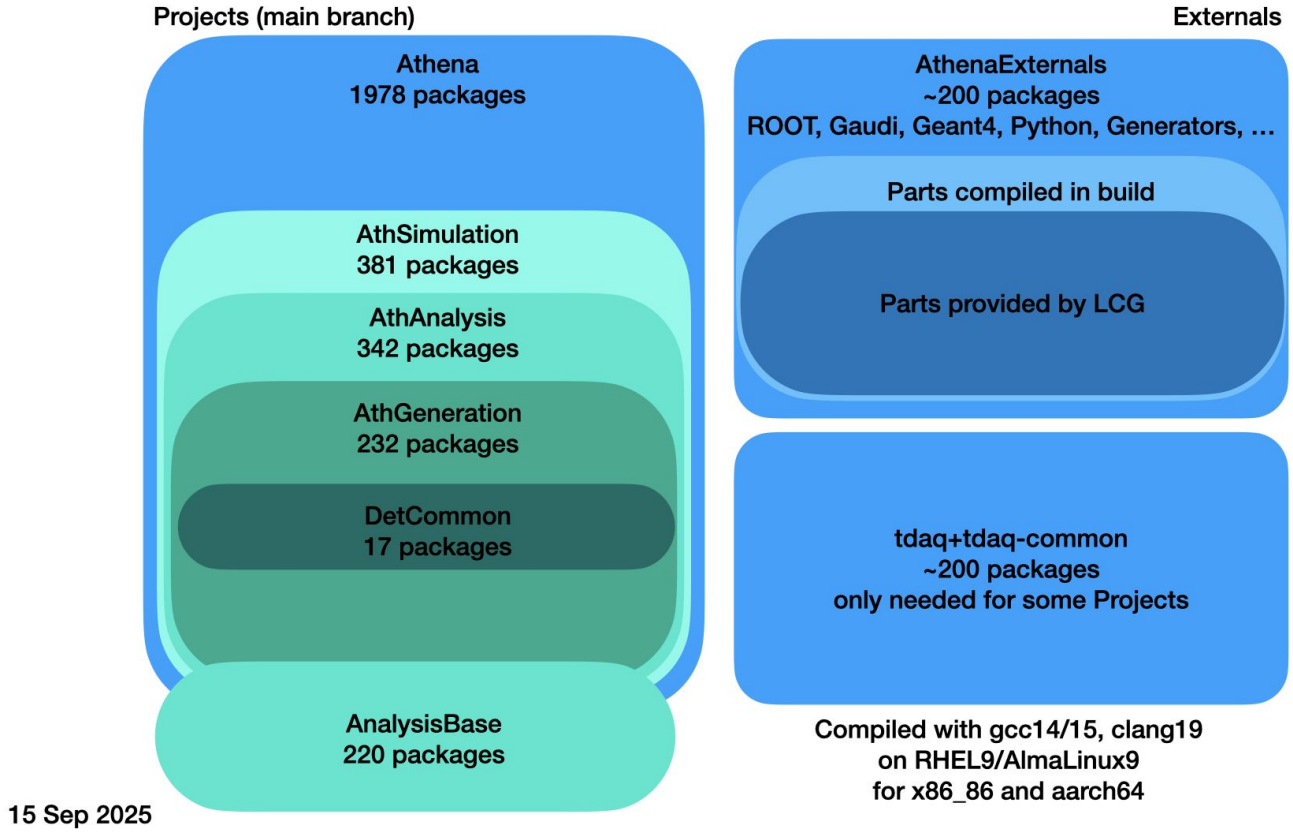
List of topics in this talk:

- Overview - building blocks
- Software repository
- Merge request reviews
- Shifts
- Lessons learnt
- CMake
- Externals and LCG releases
- Projects
- Platforms
- Compilers
- Test builds
- Jenkins, Gitlab, CI
- Nightly builds
- Machines
- Numbered releases
- EOS+CVMFS installations
- ART tests

Documentation

- Section 6.1 of Run3 ATLAS software and computing paper
  - https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PAPERS/SOFT-2022-02/
  - https://arxiv.org/pdf/2404.06335
- Athena releases and nightly builds
  - https://atlassoftwaredocs.web.cern.ch/athena/developers/releases/
- Merge review shifter procedure: https://atlas-software.docs.cern.ch/software-infrastructure/code-review/
- ATLAS C++ coding guidelines: https://atlas-software.docs.cern.ch/coding-guidelines/
- Athena Guidelines: https://atlas-software.docs.cern.ch/athena/guidelines/
- ATLAS cmake configuration:
  - https://atlassoftwaredocs.web.cern.ch/athena/developers/cmake/
- ATLAS cmake package:
  - https://gitlab.cern.ch/atlas/atlasexternals/-/tree/main/Build/AtlasCMake
- How to build a release
  - https://atlassoftwaredocs.web.cern.ch/athena/developers/building/
- ATLAS offline software in gitlab
  - https://gitlab.cern.ch/atlas/athena and https://gitlab.cern.ch/atlas/atlasexternals
- Hep_OSlibs meta-package
  - https://gitlab.cern.ch/linuxsupport/rpms/HEP_OSlibs/-/blob/el9/README-el9.md

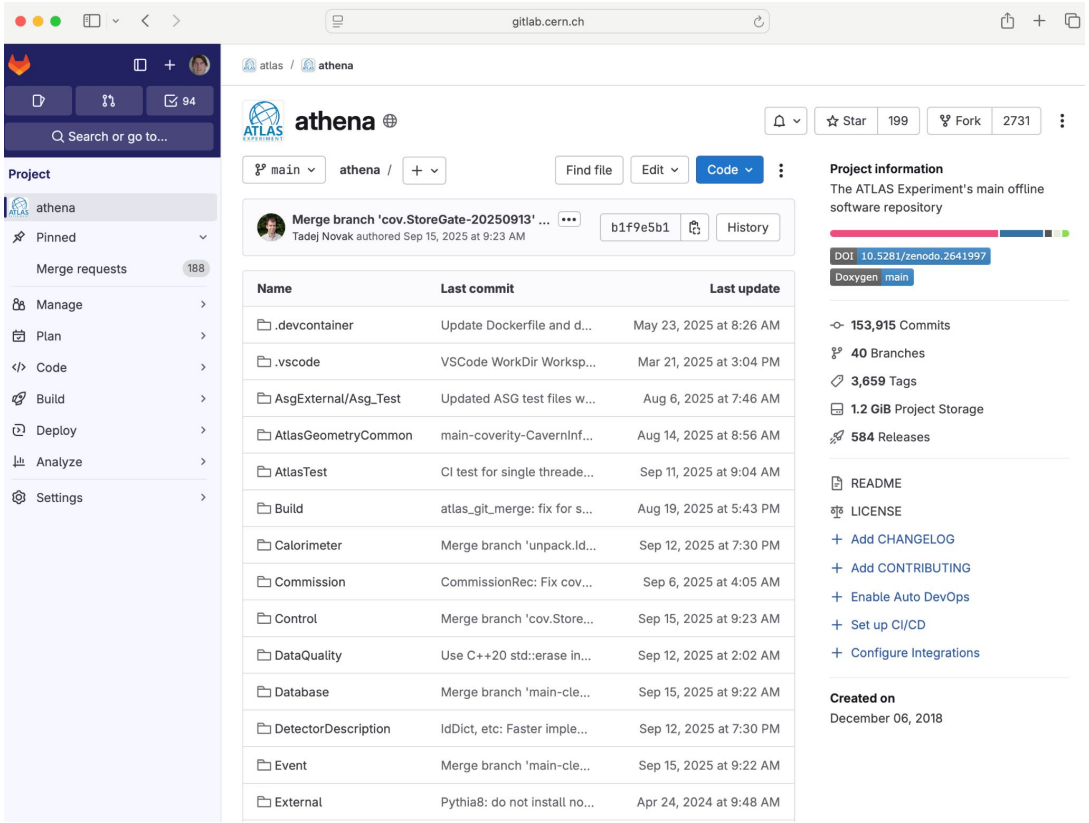
ATLAS software schematic overview



3 major blocks built by different teams:

- **Projects:** ATLAS offline software
- **Externals:** CERN EP/SFT through LCG layers and ATLAS offline software
- **TDAQ**: ATLAS trigger/DAQ

ATLAS offline software repository



ATLAS offline software is hosted at:

https://gitlab.cern.ch/atlas/athena

And some of the CMake “glue” code for externals at:
https://gitlab.cern.ch/atlas/atlasexternals

ATLAS offline software contributions

Everyone in ATLAS can open a merge request (MR) to the ATLAS code (Git workflow, Shift documentation)

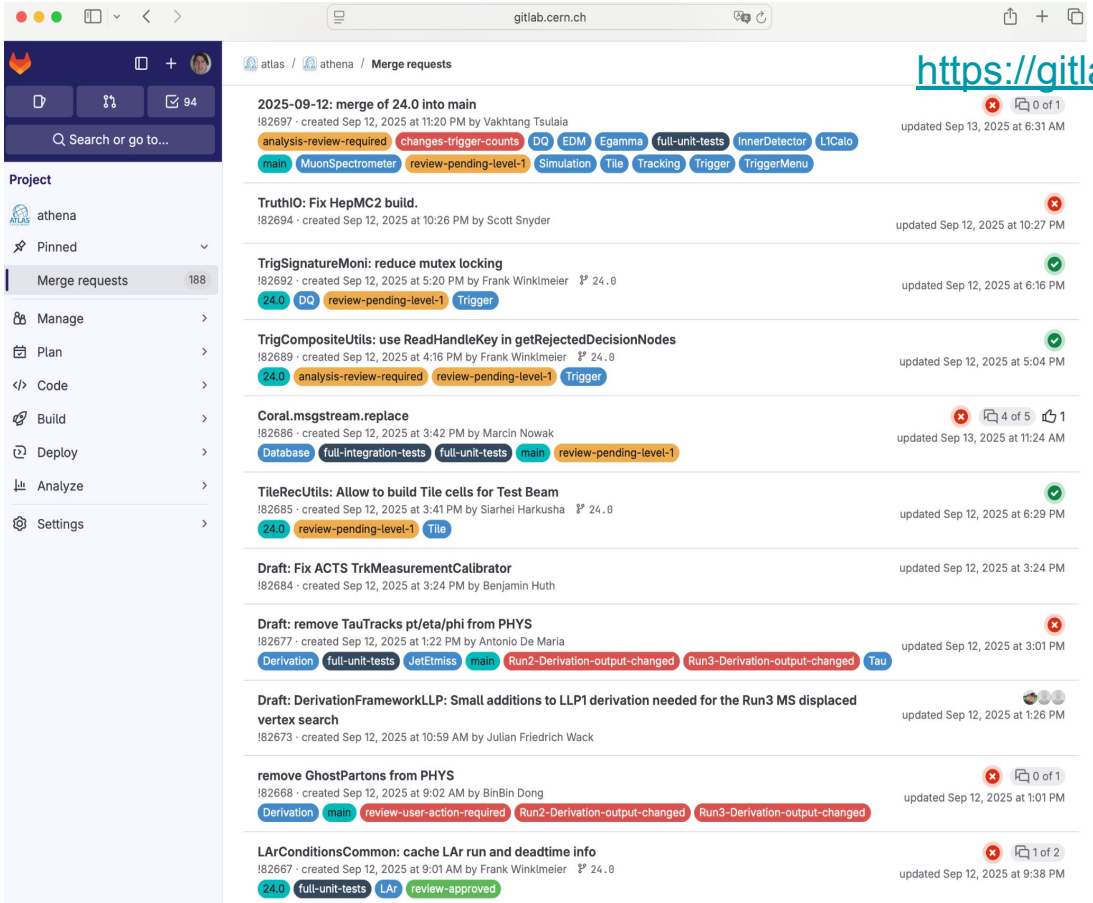
Local code development:

- Make local change in git feature branch of athena repository
- Compile and test a handful of packages against relevant nightly build on CVMFS

MR procedure:

- Incremental CI build of the code changes in MR against current gitlab branch
- CI with unit tests (few seconds to minutes) for affected packages and ~100 CI tests with longer “production” workflows and output validation (1-4h)
- Output validation: comparison of outputs (HITS, AOD, DAODs,...) to be binary identical in physics variables - files are stored on CERN EOS and synced to CVMFS
- Allow “output-changes” only if coordinators agree or meeting discussion in case of production reconstruction
- MR checks by MR shifter level 1 - can escalate to Level 2 shifter or experts
- Release coordinator accepts ~30-60 MRs per work day
- Nightly builds for different “projects” and distributed to CVMFS
- Several hundred longer “integration” tests are run through PanDA overnight (ART)

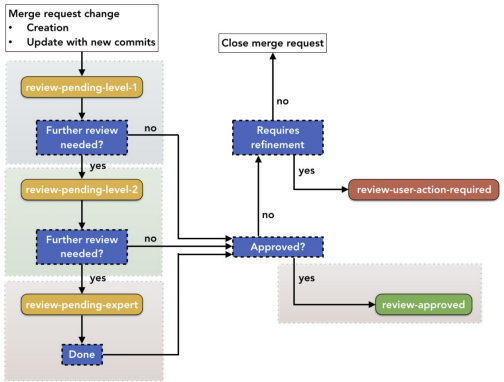
ATLAS offline software contributions



https://gitlab.cern.ch/atlas/athena/-/merge_requests

Merge request reviews by shift team

- Workdays, two level-1 and two level-2 shifters on shift: 9-13:00,13-17:00
- Release Coordination shifter all week for main and 24.0 branch each (very expert level)
- Mattermost channel for communication
- Merge review shifter procedure: https://atlas-software.docs.cern.ch/software-infrastructure/code-review/
- ATLAS coding guidelines:
  - C++: https://atlas-software.docs.cern.ch/coding-guidelines/
  - Athena Guidelines: https://atlas-software.docs.cern.ch/athena/guidelines/
  - Checked in CI with CheckerGccPlugins package (link) and flake8 (link)



- CI and Shifter mark MR status and needed actions through **labels** in gitlab

Practical lessons learnt and Policy decisions

- Practical considerations:
  - Keep MRs small and break them up into several pieces if needed
  - Decide early on coding guidelines and check/enforce them automatically in the CI with e.g. with CheckerGccPlugins package (link), flake8 (link) and static code checker like cppcheck (link) or coverity
  - Avoid working with too many specialized production branches
    - In Run2 ATLAS worked with at least 5 (or more ?) different dedicated branches (Tier0 reconstruction, Trigger, upgrade, derivations+analysis, simulation, main)
    - Extremely tedious to merge them back together after several years of separate progress in the branches
    - For Run3 decided to only have 2 active branches: (a) Tier0 reconstruction, HLT trigger, simulation (b) event generation, derivations, analysis, upgrade
  - It might be worth checking with the ACTS team about their CI practices: https://github.com/acts-project/acts and https://acts.readthedocs.io/en/latest/contribution/contribution.html

Policy decisions

- Collaboration policy decisions:
  - Licence
    - Add an open source licence that is compatible with all dependencies early on , like e.g. Apache 2.0 - see also https://hepsoftwarefoundation.org/activities/licensing.html
  - Copyright
    - Similarly add a copyright of the code - use something like "`Copyright (C) 2002-2025 CERN for the benefit of the ATLAS collaboration`" - but on behalf for BNL and/or JLab
  - Check the Open Source resources
    - CERN's Open Source Program Office: https://opensource.web.cern.ch/welcome-ospo-page

Operating system, git and platforms

- Operating system
  - Run3 (2022-2026): RHEL9/AlmaLinux9 + Hep_OSlibs meta-package (link)
    - Bare metal machines (64 core AMD EPYC) provided by CERN IT for nightlies and CI
  - Run2 (2015-2018): CentOS7 nightlies once per 10 days built in container
- Offline software repository and CI
  - Open source and hosted at https://gitlab.cern.ch/atlas/athena/ and https://gitlab.cern.ch/atlas/atlasexternals
  - Using jenkins CI for athena MRs and gitlab CI for atlasexternals MRs
  - 2 gitlab branches of athena under active development:
    - main (Event generation, derivations, Analysis)
    - 24.0 (Tier0 reconstruction, MC simulation, HLT at Point1)
- Platforms
  - x86_64-v2 and aarch64 (Arm v8) (N.B. about 1.5% of Grid jobs with x86_64-v2)
  - Main: gcc 14.2 (Run3 production), gcc15.2, clang19 with cuda 12.8.1, python 3.12.11
  - 24.0: gcc 13.1, python 3.11.9, cuda 12.4
  - Run2 legacy builds with gcc11 and gcc8
- Containers
  - Created for AnalysisBase/AthAnalysis every week
  - Created on demand for other releases occasionally

BONUS

Projects

- Full Athena build of ~2000 packages supports (theoretically) most of the production workflows
  - Athena, AthSimulation, AthAnalysis/AnalysisBase, AthGeneration, DetCommon
  - Projects with dedicated use cases developed years ago to make builds a bit more light-weight and save build time (when build machines were not yet so powerful)
  - Uses LCG_108 as basis and tdaq/tdaq-common (Athena, DetCommon) for externals
  - Can be used via CVMFS
- AnalysisBase is an exception:
  - Full stand-alone ROOT based release (no LCG and tdaq dependencies)
  - Work horse for end-user analysis
  - Can be used via CVMFS and also with containers on e.g. a laptop

LCG releases and AtlasExternals

- Software stack provided by CERN EP/SFT with
  - Content overview: https://lcginfo.cern.ch
  - More details in “2025 CERN EP/SFT program of work” overview talk at link
  - Provides compilers and consistent software stacks: 800+ external packages
  - Major LCG versions in sync with major/bugfix versions of ROOT
  - ATLAS requests LCG layers usually with minor version updates of MC generators and a few other external packages every few weeks
- AtlasExternals
  - Dedicated builds of externals ATLAS prefers to control version/patches:
    - Geant4 (10.6 used for Run3, and 11.3 for R&D)
    - Gaudi
    - Several python packages, onnxruntime, gdb and more
    - ROOT for AnalysisBase

CMake and Packages

- Historically ATLAS offline code is organized in ~2000 packages, i.e. translates into ~2000 subdirectories with dedicated CMakeLists.txt files that represent e.g. a particular reconstruction algorithm+tools
- CMake:
  - Used to build libraries and executables together with python configuration and data files
  - A dedicated set of CMake macros was developed to ease usage of package structure vs. libraries (link and link)
  - EOS and CVMFS are used for large data files for CI tests
  - User can develop code and build small parts against a full pre-build nightly (located on CVMFS)
  - CTest is used for a total of ~2400 unit tests
  - CPack is used to create RPMs used for nightly installation on CVMFS
  - If deemed good a nightly can be declared a numbered stable release, will separately installed on CVMFS with ‘unlimited lifetime and can be directly used in production
  - N.B. nighlies can also be used in production but will unless pinned disappear after 30 days)

CMakeLists.txt examples for ATLAS package

main / athena / Control / AthenaExamples / AthExHelloWorld / **CMakeLists.txt** — Find file | Blame | History | Permalink

Cleanup AthExHelloWorld
Tadej Novak authored 8 Mar 2023 at 18:56 — Verified — 7905c150

Code owners Assign users and groups as approvers for specific file changes. Learn more. — Manage branch rules

CMakeLists.txt 1.26 KiB — Edit | Lock | Replace | Delete

```
# Copyright (C) 2002-2023 CERN for the benefit of the ATLAS collaboration

# Declare the package name:
atlas_subdir( AthExHelloWorld )

# Component(s) in the package:
atlas_add_component( AthExHelloWorld
                     src/*.cxx
                     src/components/*.cxx
                     LINK_LIBRARIES GaudiKernel AthenaBaseComps )

# Install files from the package:
atlas_install_joboptions( share/*.py )
atlas_install_python_modules( python/HelloWorldConfig.py
                              POST_BUILD_CMD ${ATLAS_FLAKE8} )

# Test(s) in the package:
atlas_add_test( AthExHelloWorld
                ENVIRONMENT THREADS=0
                SCRIPT test/test_AthExHelloWorld.sh )

atlas_add_test( AthExHelloWorldMT_1
                ENVIRONMENT THREADS=1
                SCRIPT test/test_AthExHelloWorld.sh )

atlas_add_test( AthExHelloWorldMT_2
                ENVIRONMENT THREADS=2
                SCRIPT test/test_AthExHelloWorld.sh
                LOG_IGNORE_PATTERN "AthenaHiveEventLoopMgr.* processing event|^HelloWorld .*(INFO|WARNING A WARNIN
)

atlas_add_test( AthExHelloWorldCA
                SCRIPT python -m AthExHelloWorld.HelloWorldConfig
                POST_EXEC_SCRIPT nopost.sh )
```

main / athena / Reconstruction / Jet / JetRec / CMakeLists.txt — Find file | Blame | History | Permalink

JetRec: thread-checker cleanup
Frank Winklmeier authored 23 Jan 2025 at 08:29 — e81bbbda

Code owners Assign users and groups as approvers for specific file changes. Learn more. — Manage branch rules

CMakeLists.txt 2.99 KiB — Edit | Lock | Replace | Delete

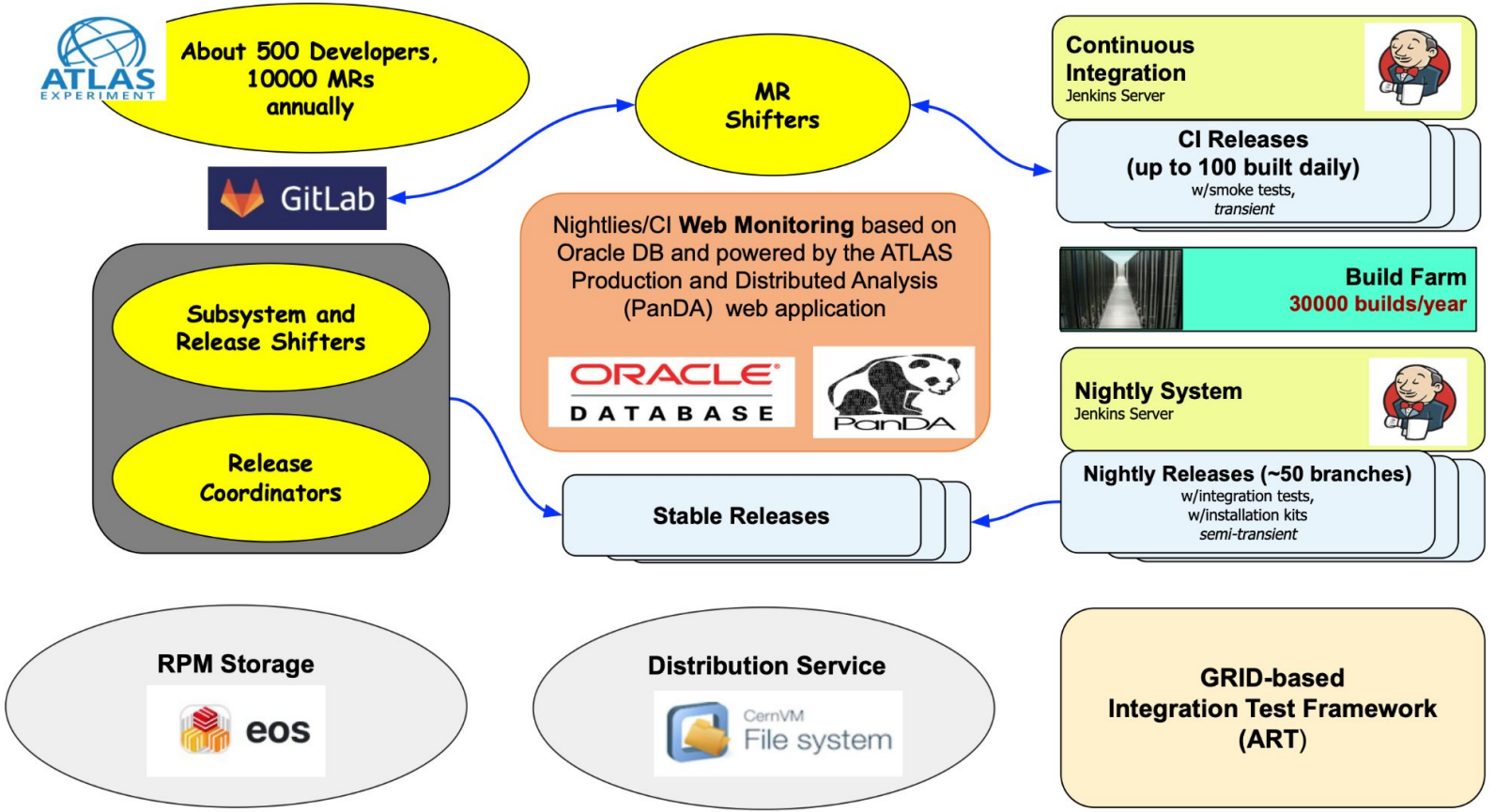
```
# Copyright (C) 2002-2025 CERN for the benefit of the ATLAS collaboration

# Declare the package name:
atlas_subdir( JetRec )

# Extra dependencies, based on the environment:
set( extra_libs )
if( NOT GENERATIONBASE )
   list( APPEND extra_libs xAODPFlow )
   if( NOT XAOD_ANALYSIS )
      list( APPEND extra_libs AthenaMonitoringKernelLib )
   endif()
   if( NOT XAOD_STANDALONE )
      list( APPEND extra_libs StoreGateLib )
   endif()
endif()

# External dependencies:
find_package( FastJet COMPONENTS fastjetplugins fastjettools siscone siscone_spherical )
find_package( FastJetContrib COMPONENTS VariableR RecursiveTools )
find_package( GTest )

# Component(s) in the package:
atlas_add_library( JetRecLib
  JetRec/*.h Root/*.h Root/*.cxx src/*.cxx
  PUBLIC_HEADERS JetRec
  INCLUDE_DIRS ${FASTJET_INCLUDE_DIRS} ${FASTJETCONTRIB_INCLUDE_DIRS}
  LINK_LIBRARIES ${FASTJET_LIBRARIES} ${FASTJETCONTRIB_LIBRARIES} AsgDataHandlesLib AnaAlgorithmLib AsgTools  AthL
  PRIVATE_LINK_LIBRARIES CxxUtils xAODTracking )

if( NOT XAOD_STANDALONE )
  atlas_add_component( JetRec
    src/components/*.cxx
    LINK_LIBRARIES ${FASTJET_LIBRARIES} AthContainers AthenaBaseComps GaudiKernel JetEDM JetInterface JetRecLib St
endif()
```

Left: https://gitlab.cern.ch/atlas/athena/-/blob/main/Control/AthenaExamples/AthExHelloWorld/CMakeLists.txt
Right: https://gitlab.cern.ch/atlas/athena/-/blob/main/Reconstruction/Jet/JetRec/CMakeLists.txt

Nightly build flavours

- Building nightlies from 24.0 and main gitlab branches
  - Athena, AthSimulation, AthGeneration, AthAnalysis, AnalysisBase, DetCommon
  - x86_64 and aarch64
  - Athena nightly with clang19
- Development nightlies
  - dev3LCG (ROOT head), dev4LCG (currently v6-36-00-patches)
  - Test new versions of ROOT, external packages, cmake, cuda etc.
- Several dedicated experimental special nightlies
  - ACTS, Geant4 11.3, archflag (x86_64-v3), lto, gccchecker, HepMC2, Gaudi
- Legacy nightlies
  - Run2 CentOS7 builds in containers

ATLAS Offline Software Development Workflow at a Glance



ATLAS CI system

Key component of ATLAS offline software workflow since 2017

- Jenkins based build and testing system interconnected with GitLab
- CI build for each GitLab Merge Request (MR) creation/update
  - Up to 100 CI jobs daily
  - ~16500 CI jobs completed in 2024
- Rapid unit and integration testing
- Efficient pipelines with dynamic optimization of build and test scale
- Comprehensive feedback to developers
  - Dynamic monitoring is based on the Oracle DB technology and integrated with the ATLAS BigPanDA web service
  - Job results are posted directly to GitLab MR views
- ATLAS teams use CERN GitLab CI for smaller projects

ATLAS Nightly System

- Validates accepted code changes every night
- Dedicated Jenkins automation server (separate from the CI system)
- ~ 50 branches - production, experimental, testing new externals, legacy
  - Optimized scheduling - some branches do not run daily
- Support for Alma9 and ARM platforms
  - Legacy builds within CentOS7 containers
- ~ 13200 Nightly jobs completed in 2024
- Employs the same dynamic monitoring as the CI system
- Release installation on the CVMFS file system
  - Worldwide accessibility
  - See details in the next slides
- Rapid fast unit testing, executed locally on build nodes
- Comprehensive GRID-based integration testing in the ART framework
  - See details in the next slides

ATLAS Software Build Farm

- 18 powerful 64-core Alma9 BM nodes
  - 251 GB RAM, 1.7 TB SSD
- 10 16-core Alma9 VM nodes
  - 114 GB RAM, 160 GB SSD, 490 GB eph SSD, 490 GB Ceph
- 4 20-core ARM VM nodes
  - 57 GB RAM, 200 GB SSD, 490 GB Ceph
- Efficient use through sharing nodes between the CI and Nightly systems
  - Priority for CI jobs at day time, nightly jobs at night time

Release Installation

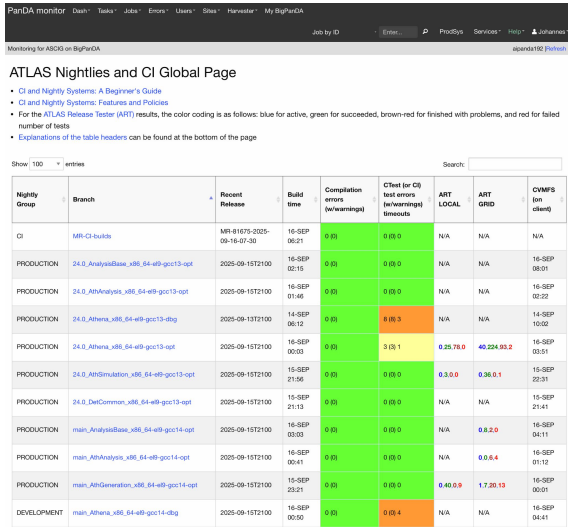
- Nightly installations
  - Executed on three CVMFS release managers
  - Created from RPM files using dnf5
  - Procedure based on Bash shell scripts, featuring:
    - Parallelization mechanisms, including lock-based synchronization
    - Support for all CVMFS transaction types, with results analysis and notifications
    - Storage space management
      - Standard nightly release retention policy: 30 days
- Stable releases installations performed on a separate CVMFS server
- Additional installations maintained
  - Data files for test frameworks
  - External packages and tools
- Key statistics
  - Release installation time: 10 to 100 minutes, depending on system load and release size
  - Peak Activity: About 40 release installations on peak days
  - Storage utilization: 7.2 TB out of 15 TB available

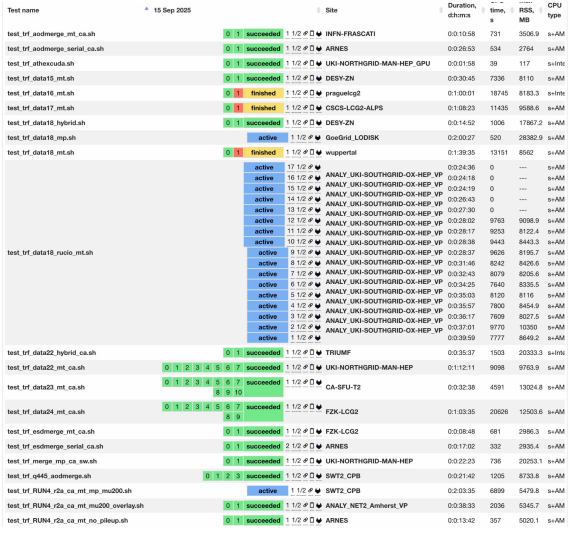
ART: ATLAS Release Testing Framework

- Python testing framework designed to detect defects and bugs in nightly builds
- Executes tests locally on the dedicated VMs or on the GRID
- Triggered by Jenkins Nightly Server and runs on nightly builds installed on CVMFS
- Jenkins initiates the GitLab-CI API to start a pipeline
- Jobs can run for up to 24 hours, processing a high number of events, which helps uncover rarer bugs
- Current operations:
  - Runs for approximately 20 nightly releases each day
  - Executes thousands of grid jobs and hundreds of local node jobs daily
  - GRID job results are published to the BigPanDA web service
  - Local job results are hosted on EOS local web but will soon be integrated into BigPanDA

ATLAS BigPanDA Monitoring System

- Django-based web application that aggregates data from Oracle DB and other sources
- Provides a wide range of dashboards, from high-level summaries to detailed views of individual computational jobs and their logs
- Supports dashboards for ART, CI, Nightly Systems, and many other workflows





Summary and Conclusions

- Provided a brief overview of the ATLAS offline software build system, dependencies and build procedures
- More details in the documentation links provided on slide 3