EPJ Web of Conferences **251**, 02017 (2021) https://doi.org/10.1051/epjconf/202125102017
CHEP 2021

Containerization in ATLAS Software Development and Data Production

Nurcan Ozturk[1,*], *Alex* Undrus[2], *Marcelo* Vogel[1,3,**], and *Alessandra* Forti[4]

[1]University of Texas at Arlington, Physics Department, Arlington, Texas, USA
[2]Brookhaven National Laboratory, Physics Department, Upton, NewYork, USA
[3]University of Wuppertal, Faculty of Mathematics and Natural Sciences, Wuppertal, Germany
[4]University of Manchester, Physics and Astronomy Department, Manchester, UK

Abstract. The ATLAS experiment's software production and distribution on the grid benefits from a semi-automated infrastructure that provides up-to-date information about software usability and availability through the CVMFS distribution service for all relevant systems. The software development process uses a Continuous Integration pipeline involving testing, validation, packaging and installation steps. For opportunistic sites that can not access CVMFS, containerized releases are needed. These standalone containers are currently created manually to support Monte-Carlo data production at such sites. In this paper we will describe an automated procedure for the containerization of ATLAS software releases in the existing software development infrastructure, its motivation, integration and testing in the distributed computing system.

1 Introduction

The ATLAS experiment [1] at the Large Hadron Collider (LHC), similarly to other experiments, distributes its software releases to grid sites using CVMFS [2] as the software distribution service. Production and analysis jobs access the software from a /cvmfs space. The ATLAS Nightly and Continuous Integration Systems [3] use pipelines for building and testing releases with software updates early and often. When a software release is ready, it is published in CVMFS, registered in the Computing Resource Information Catalogue (CRIC) [4] and appropriate tags that describe the release and its purpose are created in the ATLAS Metadata Interface (AMI) database [5]. These AMI-tags are eventually used by the ATLAS production system [6] to select the software needed to run grid jobs in producing physics data.

The containerization effort began with the production of containers consisting of thin Operating System (OS) layers. Current production jobs use these containers, obtaining all needed software and detector specific data via CVMFS and the Frontier service [7], which is another service requiring network connectivity. This system works well at sites that have

*e-mail: Nurcan.Ozturk@cern.ch
**University of Texas at Arlington is the current affiliation.

This work was supported by the U.S. Department of Energy, Office of Science, High Energy Physics contract No. DE-SC0012704 and by the U.S. National Science Foundation.

public networking and access to the CVMFS file system. At sites that don't have this capability, as is the case with most High Performance Computing (HPC) centers, the software needs to be installed manually. Given that the only distribution method is currently via CVMFS, this has always represented a main road blocker for a wider use of HPCs. One way to bypass this problem is to build standalone containers, which provide all the needed software and data. The main advantage of this method is the ease of building containers with a single software release from the release's RPMs (with a well defined dependency tree) and a dedicated, simpler release setup script. Building from RPMs results in significantly smaller containers (30 GBs) as compared to the multi-release ones (200 GBs), which were considered in the past yet proven to be difficult to distribute and install.

The standalone container design developed in a previous work [8] was proven successful by submitting simulation jobs with the PanDA (Production and Distributed Analysis) user tools [9] and the production system tools in the PanDA system [10].

In this paper we describe the work done over the past year, which was dedicated to fully integrating this prototype with the ATLAS Nightly Build System, for an automatic production of software releases together with their associated containers, as well as with the CRIC system, the AMI database and the PanDA system. Additional effort was placed into developing a framework for testing the nightly release containers with its own monitoring system. A procedure was streamlined for container environment setup and for extraction of the necessary detector specific data from the trigger and conditions databases. The work done has motivated so far several HPC centers to provide resources for ATLAS in a more stable manner.

The possibility to add these standalone containers to CVMFS automatically opens the door to the idea of flattening software distribution to grid sites, HPCs and clouds by using containers in the future. It would also simplify the production system components that currently have to distinguish between containerized and non-containerized software.

2 Strategy for container building and registration

Docker is a light weight, software based virtualization technology [11] that encapsulates applications inside a complete file system containing software, system tools and libraries, and the runtime needed for the applications to run. The application processes run natively in the host yet isolated in containers. This isolation guarantees that applications will run in the same way regardless of a particular configuration of the host. In the Docker framework, containers or images are built by adding layers to pre-existing images through the execution of sets of sequential instructions specified in special configuration files called Dockerfiles. Dockerfiles facilitate image building by providing an automatic and standardized way of installing and configuring software on top of pre-existing images, such as base images with minimal OS installations. The fully configured applications and runtime packaged in containers can, in principle, run on any minimally configured host.

2.1 Streamlined environment setup and packing of conditions and trigger data

In this work, two Dockerfiles are used in a pipeline that begins with the installation of a single software release on top of the matching OS base image, and then proceeds with the installation of detector conditions and geometry databases. This two-step image building process gives us the flexibility to install different versions of the database package (which takes only a few minutes) with the same installation of the ATLAS software release (which can take more than an hour). The pipeline currently supports the creation of images that can be used in detector simulation workflows in fully standalone mode, that is, with no need

for a network connection. This independence is largely due to the installation of a custom-made conditions database package (DBRelease) that supports most simulation workflows (Figure 1).

Figure 1. The image production pipeline adds each layer on top of pre-existing ones.

The detector conditions database contained in the DBRelease package is prepared by copying COOL folders from a central Oracle conditions database to a SQLite file, which is then included in the package. COOL and POOL are components of the LCG Persistency Framework [12]. COOL consists of libraries for handling time variation and versioning of the experiment conditions data, while POOL is a storage solution for complex ROOT [13] objects. This SQLite database file has the minimal payload necessary to execute the job, which is specified by the requirements of the data to be processed. A series of python scripts are presently used to scan logs generated by standard production jobs running the same workflows on the grid and targeting the same data. The information extracted from the logs in this manner corresponds to a set of specific COOL folders and POOL files referenced in the folders, which are needed for processing the input files. This information is then used to build the custom-made DBRelease packages that are also distributed officially in CVMFS. The last step in image creation involves the generation of a python script used to set up the software release and the runtime for payload execution. This script represents a streamlined alternative to the standard packages used for release setup. The pipeline described above is currently integrated into the ATLAS Nightly System described in Section 4.

An effort is currently underway to add support for the deployment of images for running MC reconstruction in containers. The added complexity of this workflow is the need for a separate dedicated local database for the trigger configuration, which is not a COOL schema database. Significant progress has been made lately by the generation of a comprehensive SQLite copy of the Oracle trigger configuration database (TriggerDB), which can be packaged in the DBRelease package and supports most MC reconstruction workflows. Additionally, patches to the software releases have been completed to support access to a local trigger database.

2.2 Container registration in AMI and Docker Hub registry

The containers are registered in the AMI system using a naming convention and uploaded to the Docker Hub registry [14]. The naming convention follows the OCI (Open Container Initiative) image name specifications [15], and consists of the following fields: <repository user>/<repository name>:<cacheName>.<AMItag>-<counter>. As an example, the standalone container made for the simulation workflow is tagged as atlas/athena:21.0.15.sw6-0. For the repository user and repository name fields the current hierarchy tree of the Docker Hub registry is kept, however, these fields are free strings that can accommodate a change in the future to the way ATLAS organizes its software projects and branches. The cacheName field shows which version of the ATLAS software release is used. The AMItag field (e.g. sw-tag) represents a software stack identifier, which uniquely identifies how to prepare a software environment for a given ATLAS production workflow. The software stack identifiers consist of the following fields in AMI: the architecture, the platform, the compiler, the payload, the production step, the geometry version, the version of the conditions tag, the software release, the distributed software release, the version of the identifier (patched or not), the comment and the state (used, updated, obsoleted, etc.). As an example, the content of the sw6-tag in AMI can be seen in Figure 2.

id	51
tagType	sw
tagNumber	6
tagName	sw6
imageArch	x86_64
imagePlatform	slc6
imageCompiler	gcc49-opt
owner	atlas
payload	simulation production
prodstep	simul
geometryVersion	
conditionTag	
swRelease	AtlasOffline_21.0.15
dbRelease	31.8.1
distRelease	
version	patched
comment	content
state	USED

Figure 2. The content of a software stack identifier (sw6-tag) in AMI.

The images are then registered in a table in AMI with the following fields: the name of the repository source (Docker Hub, GitLab, etc.), the name of the repository user, the name of the repository, the repository tag (sw-tag), the image name, the image type (Docker, etc.)

and the hash digest of the image. The repository counter is incremental and used for rebuilds (when an image is corrupted).

The standalone containers made for official production are tested, validated and manually registered in AMI. The nightly release containers are automatically registered in AMI by the machinery of the ATLAS Nightly System. The developmental containers are not registered in AMI and kept only in the Docker Hub registry. Due to the recent changes in the Docker Hub registry policies ATLAS is transitioning to use the GitLab registry at CERN [16].

3 Standalone containers in Monte-Carlo data production

A production campaign defines which software release(s) and other data (conditions, geometry and trigger) will be used, then the corresponding docker container(s) are built to run at grid sites and HPCs in a standalone mode. The containers are fully integrated in the ATLAS production system components. Once the containers are uploaded to the Docker Hub registry they are copied (synchronized) to CVMFS automatically. The containers are then tagged in the CRIC system for all grid sites as well as for HPC sites which have available resources to join the campaign. The installation of a container at a HPC site is done manually by the site expert via "pull" or download of the container from Docker Hub. The container is specified in a production request on the DEfT (Database Engine for Tasks) [6] interface for a particular workflow (simulation, reconstruction, etc.) to run in the container mode. The production jobs are created and assigned to grid sites or to the participating HPC sites by PanDA (automatically brokered for the European HPCs and manually done for the U.S. HPCs). The staging-in of the input data, the job execution and the staging-out of the output data are identical between the conventional mode (at grid sites) and the container mode (at HPC sites) in the production system.

ATLAS currently runs Monte-Carlo full detector simulation production at HPC sites using containers. The container made for this specific workflow is tagged as atlas/athena:21.0.15.sw6-0. It contains the ATLAS offline software release, version 21.0.15, and it passed the full physics validation. The total size of this container is 28.7 GB (including the release size of 15.7 GB and the DBRelease of 13 GB) which is small enough to easily distribute and install. The size of the DBReleases is minimal by construction (custom-made), however the size of the software release can be optimized by either reducing the size of the release's RPMs at some level or by choosing a slimmer project of a release (e.g. AthSimulation versus full Athena) depending on the needs of production. The following HPC sites have been contributing to this production:

- sites in Germany: LRZ C2PAP, LRZ LMU, MPG Draco,
- sites in the Czech Republic: IT4Innovations Barbora and Salomon,
- sites in the U.S.: ALCF Theta, NERSC Cori, TACC Frontera.

Figure 3 shows the number of CPU cores (multiplied by their HS06 power (HEP-Spec06 benchmark [17])) used by the simulation jobs in running status at these HPCs between February 17-23, 2021. Each HPC site can have multiple resources (queues) as seen in the legend of the plot.

ATLAS also considers running Monte-Carlo reconstruction on simulated data at HPC sites. A container has already been built for demonstration purposes which runs reconstruction with release 21.0.77 on simulated data from release 21.0.15. Once the validation of this container is finalized and HPC sites report readiness for this data-intensive workflow, the container will be tagged in AMI and incorporated into the production system.

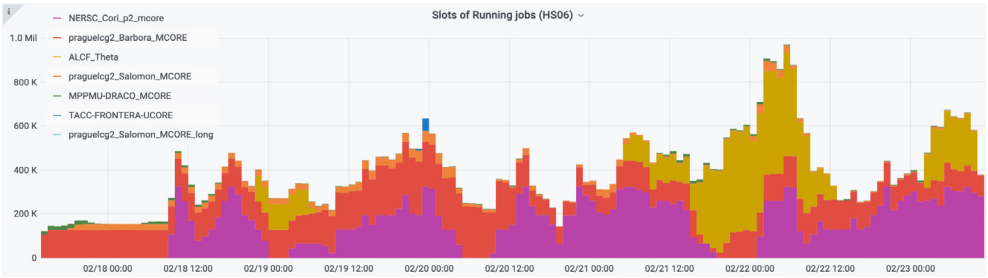

Figure 3. The number of running job slots at HPC sites versus time (between February 17-23, 2021).

4 Container builds in the ATLAS Nightly System

The ATLAS Nightly and Continuous Integration (CI) Systems are the major components of the ATLAS software development infrastructure, synchronizing efforts of several hundred software developers working around the world and around the clock. Interconnected with the ATLAS GitLab [18] code management service, the ATLAS Jenkins-based [19] CI system performs up to 100 ATLAS software builds probing code changes proposed in GitLab merge requests. The ATLAS Jenkins-based Nightly System (separate from CI) performs daily builds of ATLAS software on top of the code repository. It probes how the changes from accepted merge requests work together. In addition, it supports migrations to new platforms and compilers and verifies patches to external tools. The Nightly System maintains a multi-stream, parallel development environment with up to 30 multi-platform branches. The nightly releases are installed on the CVMFS file system and rigorously tested in the ART grid based framework [20]. Selected nightly releases undergo physics validation that involves the production of relatively large samples of Monte-Carlo data. The successful nightly releases are transformed into stable releases.

Building nightly release containers allows easy deployment to multiple different operating systems and hardware platforms including those lacking CVMFS access. The flowchart of the nightly container build process is shown in Figure 4. Once the nightly release build is completed, the RPM package is created. This package is installed on the CVMFS file system and in the release container. Then ATLAS conditions data (DBRelease) and additional external tools are added to the container. The container is tagged using the naming convention as explained above and registered in AMI. The nightly containers are currently stored in the Docker Hub registry and on the CVMFS file system for a retention period of 30 days. The transition from the Docker Hub to the GitLab registry is planned. The nightly containers are tested in the ATNC (ATLAS Testing for Nightly Containers) framework described in the next section.

Information about nightly releases builds, tests, and containers is stored in the ATLAS Nightlies Database residing in the ATLAS database production cluster dedicated to offline analysis (ATLR) [21]. It relies on the Oracle RDBMS (Relational Database Management System) and is supported by the CERN IT-DB group. The ATLAS Nightlies Database is the source of dynamic content for the nightlies dashboards on the BigPanDA monitoring web application [22], which provides numerous aggregated reports, dashboards for ATLAS software and distributed production jobs.

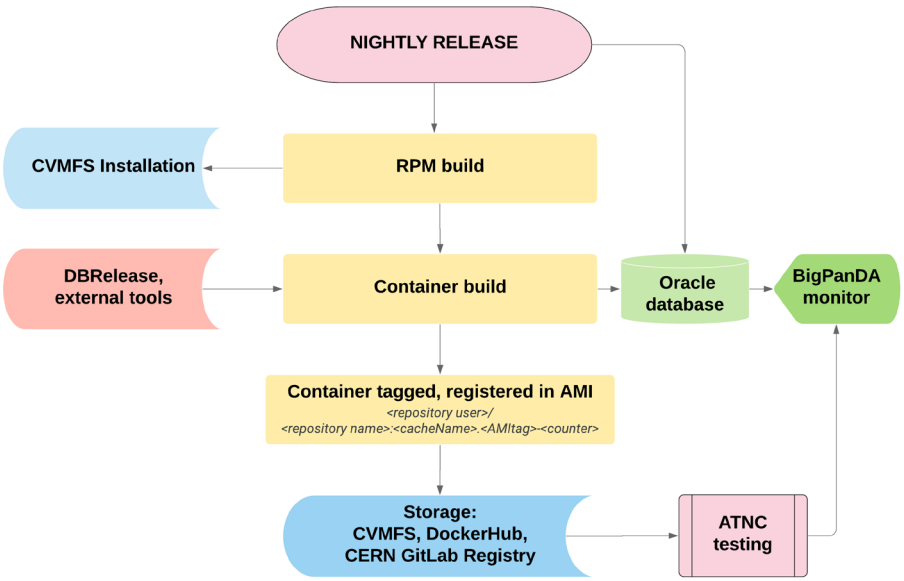

Figure 4. Diagram showing the stages of nightly container builds.

5 ATNC container testing framework

The ATNC framework runs validation tests of ATLAS nightly release containers. The goals of testing are the following:

- Check that all needed software, externals, databases are included and work together,
- Test the environment setup shipped in the container,
- Check compatibility with ATLAS distributed computing systems and tools,
- Check basic functionalities for which the container is designated (e.g., reduced physics validation tasks).

The ATNC uses YAML [23] configuration files to specify test settings such as input dataset names, production request pattern numbers and job options. The test jobs are run on conventional grid sites in the ATLAS production system. ATNC submits the production request to the DEfT interface via the curl-based API. The submission status is recorded in the Nightlies Database. The test results are monitored in the BigPanDA monitoring system. A dedicated dashboard is being developed for the display of summary and detailed views of ATNC test results.

The framework is extensible to test the containers for intermediate stable releases on the grid and selected HPC sites before using them in large scale data production campaigns. This is of particular importance for the increased utilization of HPC resources that often bring additional challenges to HEP computing.

6 Summary and outlook

An automated procedure or pipeline has been developed for the containerization of ATLAS software releases. These include stable releases as well as those produced by the ATLAS Nightly System. The nightly release containers are currently built 2-3 times a week, although the frequency can be adjusted as needed.

A testing framework has been put together for the nightly release containers to run validation tests at grid sites. The tests can be automatically triggered from the machinery of the Nightly Build System once its frequency is decided. The framework is extensible to submit tests for the validation of intermediate stable releases and stable releases for production.

A procedure has been streamlined for building and deploying the containers, which include single ATLAS software releases and additional data needed via DBReleases. The latter contain conditions and trigger information to run production workflows (both Monte-Carlo detector simulation and reconstruction) in standalone mode (no access to CVMFS or outside network connectivity) at grid sites and HPCs.

In light of the recent changes in the Docker Hub registry policies, ATLAS is transitioning to use the GitLab registry at CERN. The nightly release containers will provide a more realistic use case in testing the limitations of the GitLab registry regarding pull rates, the CVMFS hook for copying containers to CVMFS as well as their deletion from storage. In addition, a transition of the ATLAS Jenkins-based build system to the GitLab based CI is planned for 2022. The pipeline of container creation described in this work can run in that system in the future.

References

[1] The ATLAS Collaboration, J. Inst. **3** S08003 (2008)

[2] J. Blomer, P. Buncic, R. Meusel, G. Ganis, I. Sifiligoi, D. Thain, in Computing in Science and Engineering, vol.17, no. 6, pp. 61–71 (2015)

[3] J. Elmsheuser, A. Krasznahorkay, E. Obreshkov, A. Undrus, J. Phys.: Conf. Ser. **898** 072009 (2017)

[4] *ATLAS CRIC Web Portal*, http://atlas-cric.cern.ch/ (2021), accessed: 2021-02-01

[5] S. Albrand, J. Fulachier, F. Lambert, J. Phys.: Conf. Ser. **219** 042030 (2010)

[6] F. H. Barreiro, M. Borodin, K. De, D. Golubkov, A. Klimentov, T. Maeno, R. Mashinistov, S. Padolski, T. Wenaus, J. Phys.: Conf. Ser. **898** 052016 (2017)

[7] D. Dykstra, J. Phys.: Conf. Ser. **331**, 042008 (2011)

[8] M. Vogel, M. Borodin, A. Forti, L. Heinrich, EPJ Web of Conferences **245** 07010 (2020)

[9] *PanDA client tools*, https://panda-wms.readthedocs.io/en/latest/client/client.html (2020), accessed: 2021-02-01

[10] F. H. Barreiro Megino, K. De, A. Klimentov, T. Maeno, P. Nilsson, D. Oleynik, S. Padolski, S. Panitkin, T. Wenaus, J. Phys.: Conf. Ser. **898** 052002 (2017)

[11] *Docker*, https://www.docker.com (2021), accessed: 2021-02-01

[12] A. Valassi, M. Clemencic, D. Dykstra, M. Frank, D. Front, G. Govi, A. Kalkhof, A. Loth, M. Nowak, W. Pokorski, A. Salnikov, S. A. Schmidt, R. Trentadue, M. Wache, Z. Xie, J. Phys.: Conf. Ser. **331** 042043 (2011)

[13] *ROOT*, https://root.cern/ (2021), accessed: 2021-02-01

[14] *Docker Hub registry*, https://hub.docker.com/u/atlas/ (2021), accessed: 2021-02-01

[15] *The OCI (Open Container Initiative) image name specs*, https://github.com/opencontainers/image-spec (2021), accessed: 2021-02-01

[16] *ATLAS GitLab registry*, https://gitlab.cern.ch/atlas (2021), accessed: 2021-02-01

[17] *HEP-Spec06 benchmark*, http://w3.hepix.org/benchmarking.html (2017), accessed: 2021-02-01
[18] *GitLab*, https://about.gitlab.com (2021), accessed: 2021-02-01
[19] *Jenkins*, https://www.jenkins.io/ (2021), accessed: 2021-02-01
[20] T. Donszelmann, W. Lampl, G. Stewart, EPJ Web of Conferences **245** 05015 (2020)
[21] G. Dimitrov, L. Canali, M. Blaszczyk, R. Sorokoletov, J. Phys.: Conf. Ser. **396** 052027 (2012)
[22] A. Alekseev, A. Klimentov, T. Korchuganova, S. Padolski, T. Wenaus, J. Phys.: Conf. Ser. **1085** 032043 (2018)
[23] *YAML*, https://en.wikipedia.org/wiki/YAML (2021), accessed: 2021-02-01