ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV034

MIGRATION OF Tango CONTROLS SOURCE CODE REPOSITORIES

M. Liszcz, K. Kedron, P.P. Goryl, M. Celary, S2Innovation, Kraków, Poland
C. Pascual-Izarra, S.Rubio, A. Sánchez, ALBA-CELLS Synchrotron, Cerdanyola del Vallès, Spain
B. Bertrand, MAX IV Sweden, Lund, Sweden
R. Bourtembourg, A. Götz, ESRF, Grenoble, France
L. Pivetta, Elettra-Sincrotrone Trieste S.C.p.A., Basovizza, Italy
G. Abeille, SOLEIL, Gif-sur-Yvette, France
A.F. Joubert, SARAO, Cape Town, South Africa
T. Braun, Byte Physics, Berlin, Germany

Abstract

At the turn of 2020/2021, the Tango community faced the challenge of a massive migration of all Tango software repositories from GitHub to GitLab. The motivation has been a change in the pricing model of the Travis CI provider and the shutdown of the JFrog Bintray service used for artifact hosting. GitLab has been chosen as a FOSS-friendly platform for storing both the code and build artifacts and for providing CI/CD services. The migration process faced several challenges, both technical, like redesign and rewrite of CI pipelines, and non-technical, like coordination of actions impacting multiple interdependent repositories. This paper explains the strategies adopted for migration, the outcomes, and the impact on the Tango Controls collaboration.

INTRODUCTION

Tango Controls [1] is a free and open-source software ecosystem for building distributed SCADA systems. Nowadays Tango Controls consists not only of the middleware libraries providing the core functionality but also of many supporting applications, GUI toolkits, and other utilities. Tango Controls is developed by numerous contributors and members of the Tango community.

Maintaining such a large and complex project requires best-in-class tools for planning, development, testing, packaging, and version control. From 2016 till 2021 Tango Controls was using GitHub [2] for source code management and various other services, including Travis CI for continuous integration and JFrog Bintray for publishing release artifacts. At the end of 2020, Travis CI announced the shutdown of `travis-ci.org` [3], while `travis-ci.com`, previously focused on commercial software, received a new pricing model [4]. Soon after that, JFrog announced the shutdown of Bintray service [5]. These events resulted in a mass migration of Tango Controls software repositories from GitHub, Travis CI, Bintray, and other previously used service providers. The following sections describe the migration strategy, the challenges faced during the process, the outcomes of the migration, and the impact on the whole Tango Controls collaboration.

MOTIVATION AND STRATEGIC DECISIONS

When in 2015 the tango-controls project was planning its migration out of SourceForge [6], both GitHub.com and GitLab.com were considered as its destination. In favour of GitLab was its licensing policy (GitLab maintains an Open Source Community Edition while GitHub is proprietary software), but GitHub was chosen because, it was then reasoned, its larger popularity would increase the visibility of Tango and facilitate the integration of third party services such as Travis-CI. Over the years, however, GitLab increased its user base considerably–in particular within the Tango developers because of GitLab being installed on-premises in many Tango Collaboration facilities. GitLab also greatly improved its CI integration, making it much more attractive than Travis CI (e.g. for its native support of Docker containers). Meanwhile, in 2018 Microsoft acquired GitHub –an event that already prompted many projects to move from GitHub to GitLab– and launched its own CI service called "GitHub Actions" [7] which also overcame many limitations of Travis CI. The definitive trigger for considering the migration of the tango-controls organization came at the end of 2020 with the announcement that Travis CI would start charging for CI time also to FOSS projects [4], which directly affected the key repositories in tango-controls.

GitHub Actions or GitLab CI?

At this point, two alternatives were considered: either move the CI of the Tango projects to GitHub Actions or move the whole project to GitLab (other possibilities such as using GitLab for the CI but keeping the project infrastructure in GitHub were also considered but quickly discarded as impractical).

Obviously, staying in GitHub and only adapting the CI would entail less effort and, from the technical point of view, the GitHub Actions service was considered to be on-par with GitLab CI in terms of integration and features. However, the proprietary nature of GitHub weighted against investing effort on further integrating with it because of concerns about potential vendor lock-in. In contrast, the fact that GitLab's Community Edition is Open Source and that the Tango Collaboration members are already experienced in both using and maintaining their own GitLab instances, eased the concerns even about the eventuality that GitLab.com

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV034

could restrict its current allowance of free CI time and other support for FOSS project [8].

When evaluating the extra effort required for migrating to GitLab *vs* staying in GitHub, the project import feature provided by GitLab was found to do an excellent job in automatically transferring a whole GitHub project (including its code, issues, wikis, etc). Therefore, it was concluded that the main effort for the migration of a project lies in adapting its CI, which would had to be done also if moving to GitHub Actions, since continuing on Travis CI was not an option.

Regarding the disruption due to the change for the individual users, it was considered to be heavily mitigated by the fact that GitLab's UI is very similar to that of Github, that most of our users were already using GitLab instances on-premises and that GitLab.com allows to log-in using existing GitHub credentials.

Finally, it was also considered that by using the same platform for the collaboration as that used locally in the member facilities would be beneficial for reusing code, configurations and skills, and it would facilitate transferring projects to and from the local GitLab instances and sharing knowledge within the collaboration.

MIGRATION PROCESS

The various projects hosted in the tango-controls GitHub organization involve different subsets of the Tango Community, with different release cycles and development constraints. For this reason each project was allowed to choose its own schedule for the actual migration, but a "migration team" was constituted to coordinate the migrations, support the project administrators and provide both a set of tools and a common procedure for migrating.

The generic procedure for migrating a given project was implemented as follows:

1. If the project uses Continuous integration, its Travis and/or GitHub Actions .yml files are translated to an equivalent gitlab-ci.yml file and the result is tested on a temporary GitLab fork of the repository.

2. A list of participants to the project is compiled by means of a custom-made python script [9] that uses the GitHub REST API to scrape the account names of those who contributed code or participated in the discussions associated to Pull Requests and Issues of this project.

3. A migration date is set by the project administrators and the migration team. Then the participants identified in the previous step are notified that, at least 24h before the migration date, they should ensure that their GitHub account can be associated to a GitLab account (in practice this requires either that they login once into GitLab using their GitHub credentials or that they ensure that their GitHub account is set with a public email address also used in their GitLab account).

4. On the decided date, the migration team performs the actual migration as follows:

   (a) A new empty branch called "*moved-to-gitlab*" is created in the GitHub repository and set as its default branch. A single README file is added to it with a link to the new location in GitLab. The project description is also changed to "Moved to gitlab".

   (b) The GitHub project is "archived" (i.e., set as read-only)

   (c) The GitHub project is imported as a subproject in the GitLab.com tango-controls namespace. This is done using a dedicated "*tango-controls-bot*" account and the standard import tool provided by GitLab, which imports the whole git repository, the issues, the pull requests (merge requests in GitLab), the milestones, the releases the wiki, etc.

   (d) The *moved-to-gitlab* branch is removed from the fresh new GitLab repository and the proper default branch (e.g., "*main*" or "*master*" or "*develop*") is set instead.

   (e) The Tango Community is notified about the project migration via an email and a comment in a dedicated issue.

5. After the migration, the project admins may need to check and adjust the settings of the migrated project in GitLab (e.g., the project members, protected branches, etc.) and also manually adapt the documentation by replacing references to GitHub.com by the equivalent ones in GitLab.com.

C++ Core Library and Related Projects

The migration of the C++ core library repository [10] to GitLab presented multiple challenges due to many interactions with various external services. Travis CI [11] was used for builds on Linux platforms and regression testing, Appveyor [12] was used for builds on Microsoft Windows, Coveralls [13] provided code coverage reports, and Sonar Cloud [14] calculated code quality metrics. Additionally, the ABI compatibility check was implemented using GitHub Actions.

Linux CI The jobs on Travis CI were building and testing the Tango Controls C++ core library on multiple versions of the Debian operating system inside Docker containers. The use of containers was dictated by the limited choice of operating systems available in Travis CI [15]. The images of these containers were maintained by the Tango Controls C++ core library developers and included all the software required to build and test the library. Whilst GitLab Runner provides a Docker Executor and is already running all test jobs inside containers, it was decided to keep existing Debian-based containers and run them using

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV034

Docker-in-Docker [16]. The reuse of the containers allowed to reduce the effort required for the migration. The adoption of Docker-in-Docker increased the complexity of the job configuration but it could not be avoided as currently Docker is anyway required to run the C++ core library tests.

Limited hardware resources of GitLab Build Cloud runners imposed an additional challenge on the migration process. The runners offer just one CPU core [17], compared to two cores available in Travis CI virtual environment [18]. This limitation resulted in failures of some timing-dependent tests designed for multi-core systems. Such tests required case-by-case analysis and minor changes in the implementation.

Windows CI Windows-based jobs from Appveyor were not migrated to GitLab CI/CD. Instead, Appveyor has been configured as an external CI provider for GitLab. There are plans to migrate these jobs to GitLab's Windows Shared Runners, but more work is still needed to enable this migration [19].

External Services During migration, it was decided to stop using external services for code coverage reporting and code quality metrics. For both use cases, GitLab already provides built-in features that integrate better with the rest of the platform. Coveralls was replaced with a CI job that produces a code coverage report in a standard Cobertura XML format. This report is then processed by GitLab and used for test coverage visualization in GitLab's merge request UI [20]. Sonar Cloud was replaced with a CI job that runs clang-tidy [21] and produces a Code Quality report artifact [22]. This allows GitLab to show any code quality changes in the merge request UI.

Related Projects The C++ core library developers also maintain a set of device servers and utilities implemented in C++ like DataBase [23], Starter [24], TangoTest [25], TangoAccessControl [26], tango-idl [27], or tango_admin [28]. These projects do not use continuous integration or any external services, thus their migration followed the standard process of setting up appropriate "moved to" branches and importing the projects in GitLab's web UI.

The Tango Source Distribution [29] is another project closely related to the C++ core library. It used a Travis CI job to build an install-able package with the C++ library, essential device servers, and a set of Java-based tools. For the purpose of the migration, a new CI job was created in GitLab. As Ant tool [30] was used to drive the packaging process, translating the job definition from Travis CI to GitLab CI/CD did not require many changes. Additionally, the resulting package is now stored as a job artifact.

Python Core Library

The Python core library [31] was using Travis CI service for compilation and tests with multiple versions of Python, including 2.x and 3.x releases. All test jobs were utilizing Conda environments [32] to pull the required dependencies and set up a matching version of Python. Since Conda is also available as a Docker image suitable for use in GitLab CI/CD the migration process was relatively straightforward and required only translating the CI pipeline configuration into a format accepted by GitLab. The project was not using any other external services, which simplified the migration. The packaging process was not impacted and the PyTango package is still published to PyPi [33].

Java Core Library and Related Projects

Java core libraries, including JTango, use Apache Maven [34] as a software project management tool to test, build and validate the package. Maven is also used to produce binary artifacts like JAR files. Testing and building the software was automated using Travis CI service that validated all the changes to the source code which was stored in GitHub. The main challenge during migration to GitLab was the rewrite of the CI pipeline into the format expected by Gitlab CI/CD.

During the migration process, the JTango CI/CD pipeline was completely redesigned to optimize the execution time and to automatically deploy artifacts into a remote repository. Also, the obsolete and duplicated CI jobs were removed. With Travis CI, artifacts were deployed automatically to Bintray [35] and manually into Maven Central [36] repositories, as they are very popular in the Java/Maven community. Because of the Bintray shutdown [5], a new location was needed to store packages and artifacts. The free Sonatype Nexus OSS repository hosting [37] was selected for this purpose as it is a popular way to release libraries to Maven Central. As a result, both snapshot and release versions of the Tango Controls core Java libraries are now automatically deployed to OSS and synchronized with Maven Central.

Documentation

Tango Controls documentation is written in the Sphinx-compliant [38] reStructuredText format [39]. The documentation is published on the ReadTheDocs.io service [40].

The sources were kept on GitHub along with other Tango Controls projects. The project used Travis CI to test the documentation build process and to assure its quality in terms of warning-free builds, among others.

Moving the sources to GitLab was straightforward with the use of prepared scripts. In addition to setting up GitLab CI/CD (providing gitlab-ci.yml configuration file), Sphinx configuration file (conf.py) has been updated. The ReadTheDocs.io project was also reconfigured to reflect the new repository location.

The new CI pipeline takes advantage of the features provided by GitLab CI. For each merge request the documentation, including proposed changes, is built and the resulting HTML files are stored as CI job artifacts [41]. This gives the reviewer ability to check the visual quality of

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV034

the merge request directly in GitLab without doing a local build of the documentation on the reviewer's machine.

Other Projects

Other software projects related to the core have been already migrated to GiLab or are in the process of migration (Taurus [42], Sardana [43], fandango [44], panic [45], HDB++ [46]). A developer has been assigned to the migration task of each of these projects. Some of the already migrated projects (fandango, panic) are also in transition from Python 2 to Python 3. The work done on GitLab migration allowed to implement CI/CD on these projects, taking advantage of the Docker images and jobs already configured. Migration of HDB++ Archiving project and the development of Docker images for archiving CI are assigned and expected to be completed in the next year.

IMPACT ON PACKAGING

Debian

Using GitLab enables a direct reuse of the CI pipelines from the official Debian packaging infrastructure [47] which is also based on Gitlab-CI.

RPM

The Tango RPMs used to be built internally by MAX IV. With the migration to GitLab, we took the opportunity to move the Tango SPEC file repository [48] to the tango-controls group on GitLab. RPMs are now built using Copr [49], a build system and infrastructure provided by Fedora. This could have been done on GitHub as well, but using a gitlab-ci pipeline to trigger and test the Copr build made the task easier, as MAX IV has a lot of experience with their internal GitLab server. RPMs are built using the Tango Source Distribution release [29]. This is a single url to change in the SPEC file when next release will be available.

Conda

Conda [32] packages are built using an archive of the repository. The repository url is specified in the *source* field as well as in the *metadata* of the recipe. The impact of the migration to GitLab is thus minimal. Some packages, like tango-database, were created after the migration and already use GitLab as source. Some, like cpptango, were created before. They will be updated when a new version is released.

Bintray

Java core libraries were using Bintray [35] to store binaries and other artifacts like POM files. Publishing to Bintray was the main way of distributing JTango to downstream projects. After Bintray shutdown [5] and repositories migration a GitLab package registry was created for the JTango project [50]. This registry allows for storing JAR files for each JTango release. Additionally, integration of GitLab CI/CD with Maven allowed to refactor the build process and to distribute the binaries directly to OSS Sonatype Nexus [37] which is currently the official remote registry for distributing the JTango Java artifacts.

CONCLUSION

The migration process presented maintainers and developers with many challenges. A common, well-defined strategy and a detailed migration plan allowed for carrying out all necessary activities on time and consistently across multiple projects. A dedicated migration team provided support for administrators of various repositories and coordinated all the efforts related to the migration. Move from Travis CI to GitLab CI/CD caused some technical difficulties but in the end, it allowed for better integration with the rest of the GitLab platform.

The migration of Tango Controls software repositories to GitLab is still a work in progress. At the time of writing 49 out of 67 repositories in the tango-controls organization were already migrated. Critical projects like core software libraries are on GitLab since early 2021 and all development including issue handling, code review and continuous integration is performed there. Remaining projects will be gradually migrated over time or left on GitHub if there is no clear benefit in moving them.

ACKNOWLEDGEMENTS

The authors acknowledge the support of the Tango Controls Collaboration for funding a number of the developments described here as well as the Tango Controls Community for bug reports, fixes, suggestions for new features and contributions.

REFERENCES

[1] https://www.tango-controls.org

[2] https://github.com/tango-controls

[3] https://mailchi.mp/3d439eeb1098/travis-ciorg-is-moving-to-travis-cicom

[4] https://blog.travis-ci.com/2020-11-02-travis-ci-new-billing

[5] https://jfrog.com/blog/into-the-sunset-bintray

[6] https://sourceforge.net/projects/tango-cs/

[7] https://github.com/features/actions

[8] https://about.gitlab.com/solutions/open-source/

[9] https://github.com/tango-controls/gitlab-migration-tools

[10] https://gitlab.com/tango-controls/cppTango

[11] https://travis-ci.org/github/tango-controls/cppTango

[12] https://travis-ci.org/github/tango-controls/cppTango

[13] https://coveralls.io/github/tango-controls/cppTango

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV034

[14] https://sonarcloud.io/dashboard?id=org.tango-controls:cpp-tango:tango-9-lts

[15] https://docs.travis-ci.com/user/reference/overview/#which-one-do-i-use

[16] https://hub.docker.com/_/docker

[17] https://docs.gitlab.com/ee/ci/runners/build_cloud/linux_build_cloud.html

[18] https://docs.travis-ci.com/user/reference/overview/#virtualisation-environment-vs-operating-system

[19] https://gitlab.com/tango-controls/cppTango/-/issues/731

[20] https://docs.gitlab.com/ee/user/project/merge_requests/test_coverage_visualization.html

[21] https://clang.llvm.org/extra/clang-tidy/

[22] https://docs.gitlab.com/ee/user/project/merge_requests/code_quality.html#implementing-a-custom-tool

[23] https://gitlab.com/tango-controls/TangoDatabase

[24] https://gitlab.com/tango-controls/starter

[25] https://gitlab.com/tango-controls/TangoTest

[26] https://gitlab.com/tango-controls/TangoAccessControl

[27] https://gitlab.com/tango-controls/tango-idl

[28] https://gitlab.com/tango-controls/tango_admin

[29] https://gitlab.com/tango-controls/TangoSourceDistribution

[30] https://ant.apache.org/

[31] https://gitlab.com/tango-controls/pytango

[32] https://docs.conda.io

[33] https://pypi.org/project/pytango/

[34] https://maven.apache.org/

[35] https://bintray.com/

[36] https://search.maven.org/

[37] https://oss.sonatype.org/

[38] https://www.sphinx-doc.org/

[39] https://gitlab.com/tango-controls/tango-doc

[40] https://tango-controls.readthedocs.io/en/latest/

[41] https://docs.gitlab.com/ee/ci/pipelines/job_artifacts.html

[42] https://gitlab.com/taurus-org/taurus

[43] https://github.com/sardana-org/sardana

[44] https://gitlab.com/tango-controls/fandango

[45] https://github.com/tango-controls/PANIC

[46] https://github.com/tango-controls-hdbpp

[47] https://salsa.debian.org/salsa-ci-team/pipeline

[48] https://gitlab.com/tango-controls/tango-spec

[49] https://copr.fedorainfracloud.org/coprs/g/tango-controls/tango/

[50] https://gitlab.com/tango-controls/JTango/-/packages/1959228