Extending software repository hosting to code review and testing

2015 J. Phys.: Conf. Ser. 664 062018

(http://iopscience.iop.org/1742-6596/664/6/062018)

Journal of Physics: Conference Series **664** (2015) 062018 doi:10.1088/1742-6596/664/6/062018

Extending software repository hosting to code review and testing

A. Gonzalez Alvarez[1], B. Aparicio Cotarelo[1] A. Lossent[1], T. Andersen[1], A. Trzcinska[1], D. Asbury[1], N. Hł imyr[1], H. Meinhard[1]

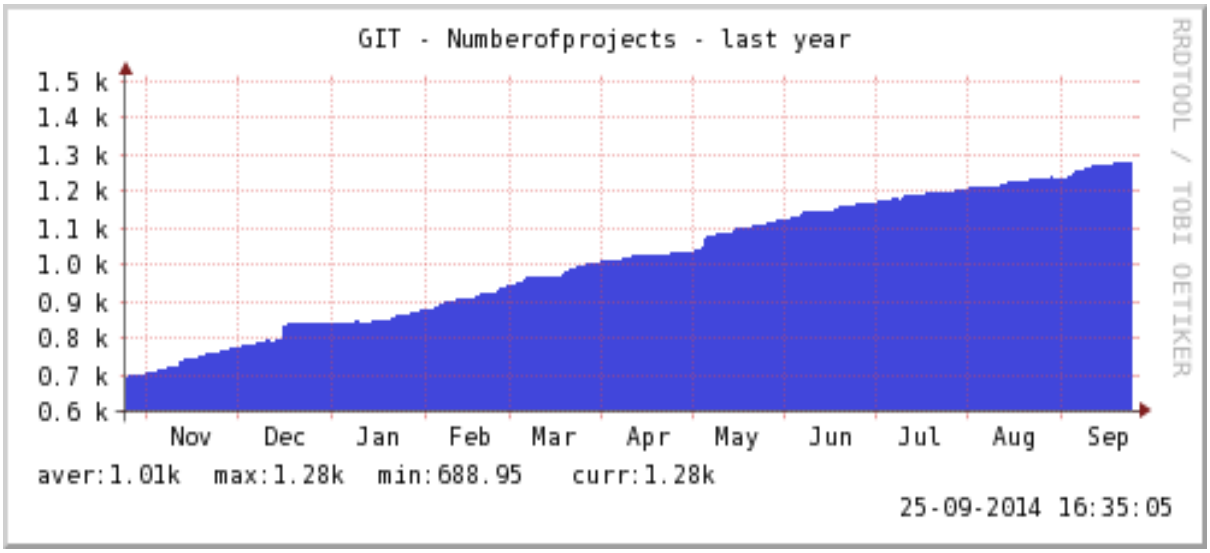
[1] CERN, CH-1211 Geneva, Switzerland

E-mail: Alvaro.Gonzalez.Alvarez@cern.ch

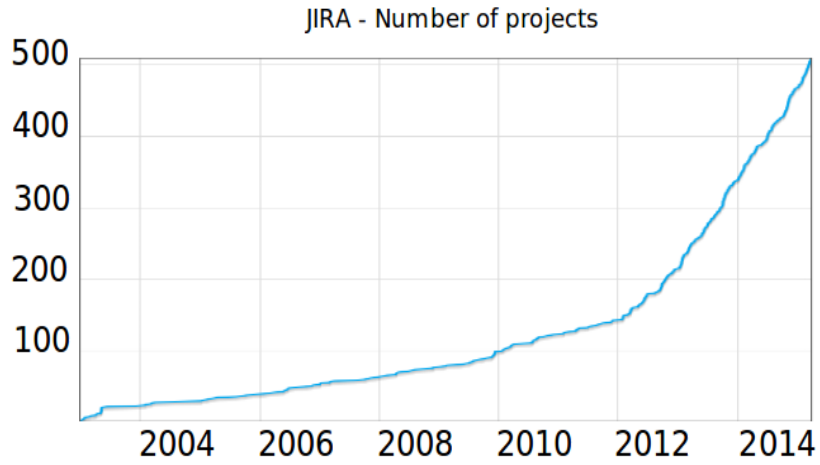
Abstract. We will describe how CERN's services around Issue Tracking and Version Control have evolved, and what the plans for the future are. We will describe the services main design, integration and structure, giving special attention to the new requirements from the community of users in terms of collaboration and integration tools and how we address this challenge when defining new services based on GitLab for collaboration to replace our current Gitolite service and Code Review and Jenkins for Continuous Integration. These new services complement the existing ones to create a new global "development tool stack" where each working group can place its particular development work-flow.

1. Introduction

A version control system handles the management of changes to computer files, mostly software code. [1] Changes are stored in revisions, which specify when this change was done, by whom and most importantly why. This is regarded as a key service needed in order to develop software. Starting with CVS [2] and currently Subversion [3] and Git [4], CERN IT has provided a central version control service to the user community at CERN and world-wide in the experiment collaborations.



An issue tracking system [8] is a computer software package that manages and maintains lists of issues, a unit of work to accomplish an improvement in a system. An issue could be a bug, a requested feature, task, missing documentation and so forth. Issue tracking systems have been in use in different groups at CERN since many years, and since 2012, CERN IT provides a central issue tracking service based on Atlassian JIRA. It was born with the objective of, on one hand consolidate the different existing installations already present at CERN (notably Savannah [11]) and to provide the service to new users who were asking for such a system.



Even though external services exists for code hosting (notably GitHub [5]), in-house services are needed notably for code that has to be deployed on systems at CERN and kept private with restricted access. Also, in order to be well integrated with the CERN computing environment with Single Sign on and e-groups, a central customised deployment is required. These systems have all seen continuous growth both in projects and users over the last years, and have proven to be very useful for the organization. In this paper, we describe the set-up of the services for collaborative software development at CERN, and the future plans for service evolution.

The current efforts of the team are focused in expanding the offered services to include a continuous integration system [12] and a code review one [13]. Continuous integration is the software engineering practice of continuously and automatically build and test a software. This process can be done after every single change or in a periodic fashion. Several continuous integration solutions are being worked on [14] [15]. Code review on the other hand, is the systematic examination of computer source code by a different person than the author of the code. This is intended to increase the quality of the code by detecting and solving errors as soon as they are introduced in the code as well as ensure the code legibility, a given code design style and share code techniques knowledge within the community of developers.

2. Version control

Software development is a key activity for many projects at CERN, as well as operations. Information Technology infrastructure from administrative application to control systems in charge of the functioning of the accelerator complex, engineering applications and dedicated software for data acquisition from the different experiments or its posteriori analysis are just examples of the multiple scenarios at CERN that implies the development and maintenance of computer software.

The task of keeping the source code of all those systems safe, accessible and version-controlled used to be responsibility of each group so independent Version Control Systems had to be maintained in local environments dedicated for individual projects or groups of projects . This task is time consuming and requires dedicated staff in each group to run such systems or the projects responsible were assigned to that task investing time in keeping their code, rather than working in the code itself. Such task is common regardless of the source code to be kept. Due to this reason, the CERN IT Department provides Version Control Systems as a service providing a secure, private and maintained platform where all the different software projects could sit releasing each project responsible from such task. Building up on the experience from the central CVS service, started in 2002 [2], the central service later on evolved to SVN from 2008 [3] and later Git (Gitolite) from 2013.[4]

As methods of software development have evolved, the centralised approach used by SVN (and CVS in the past) has given way to a distributed approach with Git, where each user commits to a local repository and then push to a central master branch. SVN and Git repositories on the central service can be requested by users on a self-service basis via the so-called "CernForge" application. [4] CernForge uses the different services APIs and the central LDAP e-groups to create, administer and delete the different projects and its integrations across the services.

The CERN Version Control services are targeting code that must be hosted at CERN and remain private. We encourage the use of Github for open source projects with external collaborators from the general public.

3. *Issue tracking*

The central Issue Tracking service at CERN is based on Atlassian JIRA [7], and was set up in 2012. JIRA was chosen after a survey conducted among users of issue tracking tools at CERN in 2011, where its configurability and user friendliness was highlighted. The service allows project teams to track bugs, tasks or other issues that may be relevant for their processes via the JIRA web application. Access authentication to JIRA is provided via the regular CERN Single Sign-on (SSO), while authorization is done via a combination of e-groups, mapped to JIRA-groups, and the roles that are defined within the JIRA application. The usage of the service is growing, and currently about 500 projects with about 360k issues are hosted in the central JIRA instance, with additional projects in custom hosted instances. JIRA is a rather flexible and fast evolving tool and allows for the use of a number of add-ons, such as a module “JIRA-Agile”, for Agile project work-flows and a Gantt chart plugin for project planning purposes. Each project may use different settings for issue types and permission schemes, allowing certain flexibility. However, the issue types, permission schemes and work-flow schemes can only be defined or customised by the system administrators. Also certain

operations like linking a JIRA project to an SVN or Git repository, where issues point to related commits, requires JIRA system privileges to set-up the regular pull of the version control repository. Thus having a way to grant project administrators a way to change some of their project settings is potentially interesting. This has been addressed via the CernForge administrator portal at CERN, that allows for JIRA project requests and other administrative operations in the system via a JIRA REST API [4]. We had to develop our own REST API plugin in order to add missing features in JIRA's system API, notably the possibility of creating projects.

4. Improving the quality of the software process

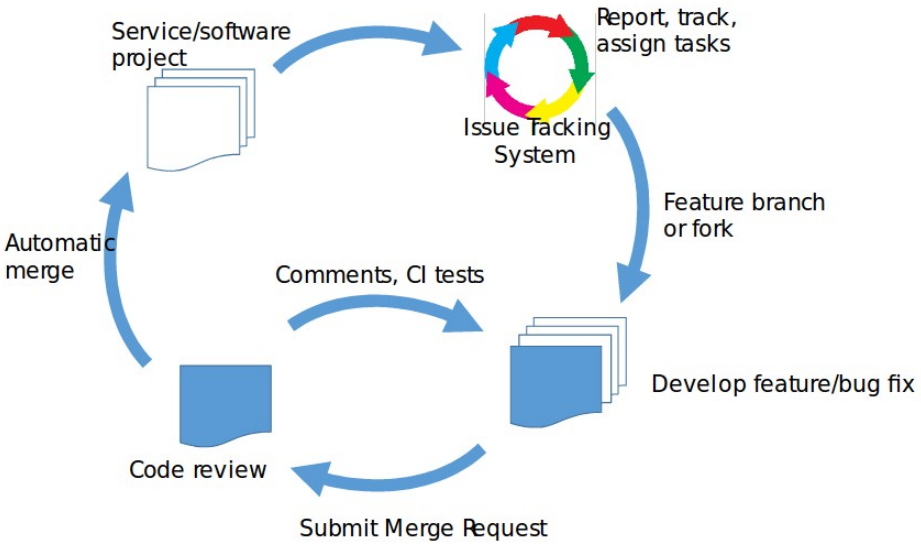
The combination of SVN/Git + JIRA allows the tracking of the development process of the code used in a project, and to evolve the code to add new features; fix bugs and other maintenance tasks, while keeping the history of evolution of the code itself as well as requests for improvements.

Version Control allows the generation of versions that can easily be deployed or rolled back. Code and software quality is ensured after the code has been compiled/packed and/or deployed, by testing the different releases in dedicated environments before a stable version is produced and deployed in the production environment. This procedure allows to detect errors that have to be corrected by the author(s) of the code producing patches or rewriting the problematic fragment of the code, however, it doesn't prevent the same errors to be produced again: the experience learned during this process doesn't get back to the whole team and the responsibility of producing software of a good quality still an individual task of each member of the team and the new knowledge doesn't get propagated.

However, it would be desirable to automate further the improvements of software quality, by adding synthetic tests as well as provisions for quality control by different groups of project participants. Such processes have been implemented in many large software projects, with local rules and procedures.

4.1. Code review

Git provides per branch development and merge mechanisms between branches or mirrors of the same repository of software. This opens the possibility of building review work-flows of the code and testing procedures before it becomes a deployable version. By applying review cycles of the code within a team, possible bugs are easier to spot, code design is ensured to be aligned and the coding skills are shared and acquired by all the members increasing the quality of the final product produced.



The increasing demand of the CERN user community of tools for code review, led to a search for a system to replace our Gitolite installation, that would scale, with flexibility to support different processes. Gitolite uses a path based authorization system, similar to SVN's, but notably the pull-merge request method offered by GitHub and similar tools is gaining popularity and it is proving to be the most effective way for code collaboration, so support for this process is considered essential.

4.1.1. GitLab

GitLab [15] is a self-service open source code review hosting application based on Git that provides collaboration and code review features. It allows the integration with issue trackers as Jira as well as continues integration platforms such as Jenkins which facilitates the creation of a fully defined set of integrated tools for software development. At the time Git was introduced with Gitolite in 2013, Gitlab was not considered mature enough to fit our needs, but in the last years Gitlab has shown a remarkable evolution.

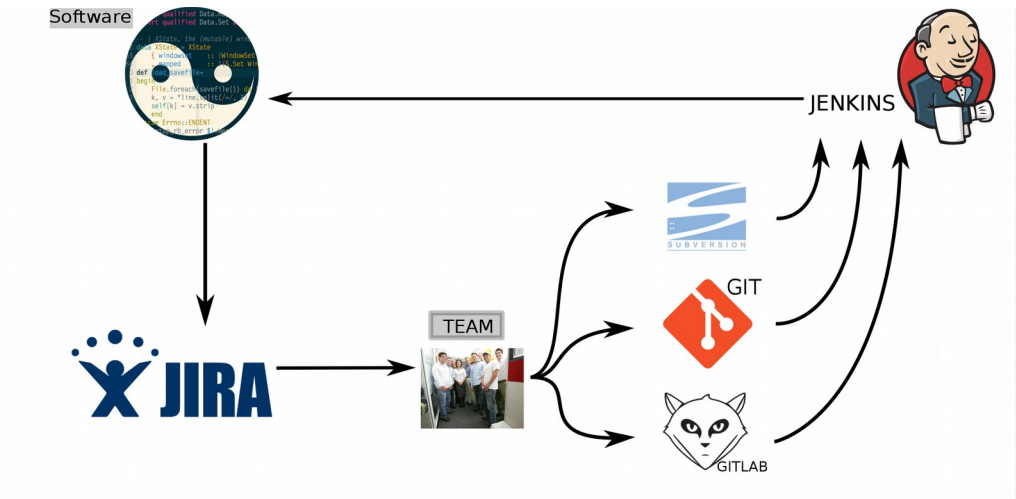
Within the project, GitLab provides a hierarchy of responsibilities map to users based on roles. Software projects may involve several levels of 'trust' for developers: some developers may be trusted with full access to all parts of a project, while other developers are only allowed to modify certain parts. This is an scenario that happens in big collaborations such experiments where community of developers can grow a lot. Projects hosted on GitLab use a model based on Merge Requests (similar to GitHub's Pull Requests) and a code review work-flow: the Feature Branch work flow: Clone project – Create branch with your feature – Write code and commit changes – Push the branch – Review you code on commits page – Create merge request – Team lead will review the code and merge it to the main branch. This model also enables contributions from developers who do not have any write permission on a repository (such as contributors from outside the team) via the Forking Work-flow where the developer creates his/her own copy of the project and submits a merge request from it to the original project.

By using as a central system for merge requests, developers are invited to review the set of changes submitted, comment and discuss the changes with the author, and eventually decide to apply (or not) the set of changes to the project's main branch enforcing a good quality of the code and sharing the coding within the team.

4.2. Continuous Integration

Continuous integration(CI) techniques complement the code review process before code changes are promoted from the development stage to a production stage by running a set of tests that can go from a simple build or compilation to a complex set of functional tests. This process is entitled to ensure that no bugs are introduced creating collateral effects on previous stable versions of the software and new features behave as designed. The CI application will get a fresh copy of the code triggered by, a commit, on a regular basis (nightly builds) or manually, it will perform all the preconfigured steps and report the results back to the users. If there was any kind of problem, the report is typically done via a JIRA issue. If successful, it is also common, to deploy the compiled results to a software repository or directly to the application, using a process called continuous delivery (CD). Many CI applications exists, where Jenkins [14] is widely used in Open source projects, and also by many software and experiment groups at CERN. Building upon the experience from the Agile Infrastructure project at CERN[16], a pre-configured Jenkins continues integration environment can be requested via the CernForge portal as a Platform as a Service. The aim of a Central CI service is to target most of the working groups at CERN, releasing their manager of the task of setting up the infrastructure and installation by themselves, making a CI solution easy to deploy to any large- and medium-sized software projects, however for smaller projects a more lightweight solution without the need for dedicated server infrastructure and administration would be preferable.

GitLab offers a built in solution for continuous integration, GitLab CI [15] that is fully integrated with GitLab. This lightweight CI is still rather young, but we are exploring this avenue as an option for smaller projects.



5. Conclusion

With the experience gained during the last years in terms of software repository hosting as well as issue tracking services, we have a set of collaboration tools for software projects at CERN and in the HEP community. Based on demands from the community of users and tools such as Git and Jenkins, the services are being expanded to address the whole software life-cycle, including quality control and testing. This stack of integrated services allows each group developing software applications to get, in a self-service way, the set of tools needed to keep and maintain, ensuring best practices, their software projects during its whole life cycle, yet maintaining confidentiality and integration with CERN computing infrastructure. For projects that cannot use an external solution like GitHub, the deployed stack removes the need for multiple private installations for software development, while benefiting from a central support that concentrates the experience on the different tools at the laboratory.

6. Acknowledgements

We would like to thank the different user communities of our services for all the input and collaboration. We would also like to thank the authors of the software packages and IT infrastructure teams who provide the foundation for the set-up of the VCS infrastructure.

References:

[1] *https://en.wikipedia.org/wiki/Revision_control*
[2] CVS at CERN Lopienski, S e.a. 10.5170/CERN-2005-002.1192
[3] SVN at CERN Hugosson, Hugo (IT-DES) CERN-CNL-ARTICLE-2008-049
[4] Husejko M, Gonzalez A et al 2014 J. Phys.: Conf. Ser. 513 052012
[5] Github URL https://github.com
[6] Sourceforge URL http://sourceforge.net/
[7] Atlassian JIRA URL https://www.atlassian.com/software/jira
[8] https://en.wikipedia.org/wiki/Issue_tracking_system
[9] https://en.wikipedia.org/wiki/Software_project_management#Issue
[11] http://en.wikipedia.org/wiki/GNU_Savannah
[12] https://en.wikipedia.org/wiki/Continuous_integration
[13] https://en.wikipedia.org/wiki/Code_review
[14] https://jenkins-ci.org/
[15] https://about.gitlab.com/
[16] Agile Infrastructure project: https://agile-infrastructure.web.cern.ch/