Proposal for modernization of various components of our infrastructure

Summary

Our existing infrastructure has served us well, and been in service now for four and half years. However, changes in the way we now develop software, the emergence of new software, and the passage of time have exposed a number of shortcomings which need to be addressed. These shortcomings have included limitations in scalability, excess resource consumption, information not being synced quickly or at all and a lack of integration between our systems. Due to these issues it is necessary to replace our entire Git infrastructure - changing one part (Git hosting, project management, code review, etc.) alone will not resolve these problems.

Sysadmin therefore proposes that we implement Phabricator, an integrated solution. It is able to provide code hosting and review, allowing for both pre and post commit review to be conducted. The repository browser is highly performant in comparison to the Chiliproject instance currently in use. The code review functionality is also an improvement over Reviewboard, providing a list of commits used to produce the code in question as well as a list of reviews which affect the same files. With minor modifications it will also be possible to offer creation of scratch repositories directly from the web interface.

Phabricator also offers task tracking, wiki pages, and many more applications, which could be used to replace Kanboard (todo.kde.org) and could potentially allow projects to have comprehensive and turnkey project management. We can therefore provide an effective software forge using Phabricator, in contrast to our existing split approach for infrastructure components. A component known as Herald may also permit us to offer an equivalent to commit filter, with the added bonus that it also covers tasks and code reviews.

Importantly, upstream has been extremely responsive to our communication and have already proved helpful in our evaluation of Phabricator. Their roadmap for future development is very promising and indicates that more of the requirements suggested by the community will be satisfied as time goes forward. Phabricator is also being used by a number of other large entities in the open source community including Wikimedia, Clang, FreeBSD and Blender.

In conclusion, we believe Phabricator will best be able to meet the overall needs of our community going forward. We would therefore like to invite a couple of projects (2-3) to work with sysadmin to evaluate the use of Phabricator in a production scenario. Once this has been completed a final decision can be undertaken by the community as to the future of our Git infrastructure.

A short history

Much of KDE's current infrastructure was conceived by the sysadmin team in June 2010 in response to a request to set up a Gitorious instance. After Gitorious proved unsuitable due to scaling issues and inflexible access control capabilities, the final implemented solution was and is comprised of Reviewboard for code review, Chiliproject (was Redmine at the time) for project management, and Gitolite for managing the actual Git repositories on the git.kde.org. It was at this time that KDE Identity was also established. Since the migration to Git was just occurring, it was also unclear how future development would be conducted (whether to split repositories or not had yet to be decided, and a date for trunk/KDE/ to migrate did not exist either).

Four and half years have passed since then. Some of the goals we had at the time were not accomplished and some items have been achieved which were not considered at the time. One of the notable failures included greater use of project management tools (particularly those inbuilt to Redmine/Chiliproject); although many projects, particularly extragear projects, had communicated to us their desire for better-integrated project management tools that could allow their teams to work more efficiently, the fallout from the Redmine fork hampered development of Redmine and Chiliproject both, and they proved too fragile and limited to be serious contenders for project management.

We have however managed to build tools which monitor our projects for build failures (build.kde.org) and which are able to checkout a large number of Git repositories in a familiar folder layout conveniently.

Problems with our existing systems

Unfortunately the passage of time has also unveiled significant problems with the tools we selected. Although a sensible choice at the time due to a majority of the core Redmine developers joining the effort, Chiliproject has been abandoned upstream and we are unable to return to Redmine due to the level of difference between the two products. In addition, some of our modifications (which, among other things, provide the ability to set the i18n branches and the tree structure within the url) would have to be completely rewritten.

Lack of developer support in the creation of a proper commit filter solution has led to a haphazard solution within projects.kde.org. This frequently breaks – usually whenever a repository is renamed or deleted. It is also extremely coarse and creates additional maintenance problems for projects.kde.org. It would pose an additional obstacle to any migration to Redmine.

Further, such a migration would not solve the scalability issues currently associated with Chiliproject, which produce significant server load. These are caused by the repository browser (which uses an extremely slow method to access repositories, leading to page timeouts on occasion) and the code which produces the kde_projects.xml file.

The replication system which supports the anongit nodes has also reached design and scalability limitations. This is likely a result of the 1,954 Git repositories we now host, which occupy 30gb of disk space across 840k files and includes mirrors of both Qt 4 and Qt 5 (due to reliability problems with the infrastructure provided by qt-project.org). Currently new repositories are only synced onto (or old repositories removed from) the anongit nodes when a cronjob runs. Due to round-trip delay one of the anongit nodes is only able to run this once an hour, while the others can only run it once every 20 minutes.

In addition, a failsafe mechanism added by sysadmin into this cronjob following the June 2013 disk failure of Shrek can also disable this cronjob. Manual intervention is required to correct this once activated - which can be triggered by something as simple as a large enough number of scratch or clone repositories being deleted. To date this has not been a significant problem due to the instant-update mechanism, however it can only handle regular pushes (not branch deletions for instance).

Integration problems our existing approach creates

As a result of many disparate components being used to provide a complete solution it was necessary to provide glue to allow them to work together. This has itself caused extra work. These have included:

- Limitations in the use of LDAP by both Reviewboard and Redmine/Chiliproject requiring the implementation of an external daemon to keep email and names updated when they change on Identity.
- Updates to Reviewboard have broken portions of the commit hook which closes reviews. No upstream code exists to close reviews based on a Git commit.
- Projects.kde.org is completely oblivious to the existence of Reviewboard making it difficult for contributors and others to find existing reviews.
- Repositories have to be manually registered within Reviewboard when they're created by a sysadmin (after waiting ~20 minutes due to the anongit sync delay).
- No connection exists between the CI system and Reviewboard. The only possible option would be ReviewBot, which requires quite a bit of infrastructure – as well as customisations to get it talking to Jenkins. Sysadmin lacks the resources to work on this and the final solution would be excessively complicated given the end result.
- To create a new Git repository sysadmin is reliant on being able to access Chiliproject, which then generates the Gitolite configuration which is deployed on git.kde.org. This makes it difficult to customise permissions on a module level - requiring the same configuration to either be repeated globally for all repositories or repeated once per repository in a module (a list which must be manually maintained).

Feedback we have received on the existing infrastructure

Whilst initially without problem, complaints have become more common in recent times. Chiliproject was the original focus of problems due to the poor performance of the Git repository browser – a problem which sysadmin mitigated with quickgit.kde.org and the commits.kde.org redirect service. While this is duplication of infrastructure, as well as extra infrastructure to manage, it is necessary as we cannot eliminate Chiliproject without drastic changes to our infrastructure.

Reviewboard has also become a problem in recent times. Rbtools is unable to handle the kde: alias and from time to time cannot submit patches to Reviewboard. Web uploads have also been known to fail from time to time as well. Sysadmin has been unable to determine why this occurs. Once uploaded, the process of accessing patches is also imperfect. Patches generated using “git format-patch” or “git show” when uploaded to Reviewboard will often be mangled when downloaded, requiring manual intervention to correct the problem. This creates issues for both potential and current contributors as well as developers.

Members of the community have also commented on the inability of Reviewboard to subscribe them to reviews by default. While rbtools and .reviewboardrc files circumvent this issue to a certain extent, patches can still end up with no reviewers selected – meaning they get lost until someone notices them in the web interface. Additionally problems exist with how Reviewboard sends email – with some commenting on how it is rather abusive.

Our existing infrastructure also lacks a convenient area for collections of repostiories to have a “mini-website”. This would facilitate providing some basic information on the project and allow links to be set to various relevant items (such as the bug tracker, code review area, API documentation, wiki, forum and so forth). Some projects have also indicated they would like to have used issue tracking functionality which is built into such a forge. While projects.kde.org is able to support this, community and sysadmin policy in 2010 required sysadmin to disable the wiki and issue tracking functionality it has.

Research into alternative solutions

Sysadmin has been aware of many of the above problems for some time. When evaluating solutions we did not focus exclusively on just Git hosting, but rather on something which could provide a significant improvement to how the broader community interacts with our code. This would help ensure new contributors are able to find and acquire the resources they need easily and efficiently, lowering the barrier to entry and minimizing the risk of loss due to an inability to find what is needed.

While simple hosting solutions which have web interfaces are potential options, an ideal solution would be composed of a set of well-integrated tools. We discussed a number of options as part of this. Unfortunately some possibilities had to be eliminated due to being non-FOSS, lacking functionality or having scalability concerns. We have continued to monitor options we have eliminated from time to time to reassess whether their status has changed. We considered:

- Atlassian Stash: Rejected as non-FOSS.
- Github: Rejected as non-FOSS.
- Bitbucket: Rejected as non-FOSS.
- Gitorious: Rejected based on a variety of factors.
  Please see our prior report in 2010 to kde-scm-interest for more information on this.
- Gogs.io: Rejected due to lack of necessary functionality.
- Gitblit: Rejected due to lack of necessary functionality.
- Gitlab: Rejected due to scalability and security concerns, as well as difficult-to-modify code.
- Gerrit: Please see below for our rationale.

Both Gitblit and Gogs.io are solely code hosting solutions and do not provide any form of code review at this time. In the case of Gitblit it does not appear to support scratch repositories in any form as well. Gogs.io is an emergent Github style clone which does natively support scratch and clone repositories as a result of this.

In early 2013 sysadmin evaluated Phabricator as a direct replacement for projects.kde.org. The possibility of it replacing other parts of our infrastructure was also considered. Unfortunately at the time it was not possible to adopt it despite the potential it showed due to some issues which would have significantly hindered adoption. Most notably it required you to login in order to simply browse it, and once logged in a user could do nearly anything (see https://secure.phabricator.com/T603; although marked as fixed in late 2013 some blockers remained and additional functionality has been added since). Further, our ability to integrate it into our systems would be limited (see https://github.com/phacility/phabricator/issues/282). These have since been fixed by upstream.

In 2014 sysadmin undertook an evaluation of Gitlab. This is a Github-type clone which is available under a FOSS license, and was evaluated by Sprinter and GCompris. Sysadmin discovered some serious problems which eliminated it as a contender. These included code which sent debug information to an email address specified by the Gitlab developers, and the sending of an email to all (KDE) developers when access to a new repository was granted. It was also not designed to work with external hooks, necessitating hacks to facilitate this integration. The code was extremely middleware-driven and very difficult to add even basic modifications due to assumptions and abstractions happening at many middleware layers. Finally, we were unable to reach their developers via any form of real-time communication.

Our analysis of Gerrit

The primary problem with Gerrit is that it does not provide a solution to many of the issues we are currently experiencing. It only deals with code review and can therefore only replace Reviewboard. While it is possible to use Gerrit as the primary code host, this would eliminate the possibility to have scratch repositories as it does not support them out of the box. Significant modifications to our web software have in the past caused sysadmin significant issues, so we would like to avoid patches which would not be accepted by upstream where possible.

As a result of this it would be necessary to combine several different components to produce a complete Git solution. This would require further effort to integrate them with both each other and parts of KDE infrastructure such as Identity. Even after such effort is completed a certain degree of synchronisation between the tools will need to be maintained, such as registering repositories in both the code hosting tool and Gerrit.

The voting system provided by Gerrit is more fine grained than that allowed by Phabricator. The split between +1/+2/-1/-2 does allow a greater degree of flexibility in comparison to the Accept Revision / Request Changes mechanism which is provided by Phabricator. This is important to note considering the problems we currently experience with the Ship It! mechanism in Reviewboard.

As an additional positive point, Gerrit does offer native git client integration. While this does require configuration on end user systems before use, it does eliminate the need for system specific tools such as arcanist to be installed. It should be noted though that changes have been requested upstream with Phabricator already for functionality along these lines to be provided, so it is quite reasonable to expect it will be able to provide this at some point in the near future.

What does sysadmin propose?

We have kept our eye on Phabricator ever since our initial test. We performed another round of testing of its updated functionality, and given its extensive improvements we are proposing Phabricator. We think that it has both an excellent feature set today and a very good roadmap going forwards. Additionally, in the time since we last tested it, a number of large and successful projects have adopted Phabricator and are using it successfully. These include the Blender Foundation, Wikimedia, Clang and Haskell. It is also in use by very large non-FOSS firms, originally as Facebook's internal collaboration tool. While it originated at Facebook, it is now independent of this and is a self-sustaining open source project. As a result of its origins, we do not anticipate any scaling problems. Their developers have indicated that it has handled upwards of 2 million commits without a problem, which is similar to what we expect KDE to generate once all repositories are imported. There are also development going on around clustered Phabricator.

Phabricator itself is a framework and common interface paradigm providing an extensive set of independent but well-integrated applications that can be individually enabled, disabled, or access controlled as needed. Initially, Phabricator would serve as a replacement for KDE Projects, Git, and Reviewboard (and as knock-on effects, QuickGit, WebSVN, and Commits). Due to the extensive amount of functionality it offers, however, it could eventually serve as a replacement for Bugzilla, Todo (Kanboard), and more. Eventual use of that functionality would also increase the benefits we receive from the strong integration between the various applications within Phabricator.

Projects within Phabricator can have their own resources attached to them, and certain items such as issues and reviews can belong to multiple projects. Each project can also have a workboard which allows for associated issues to be managed using a workflow similar to Kanboard. This may provide a useful space to openly collaborate depending on the extent to which we adopt its various applications.

The code review functionality appears to be a large improvement to Reviewboard. Particular improvements it would provide include a listing of reviews which affect the same files and a listing of commits made to produce the relevant diff which has been submitted. Based on our research, it should also be possible to offer CI builds on posted reviews, subject to the availability of the resources to support these builds.

A component of Phabricator would also allow post-commit reviews to be opened conveniently (from the repository browser). This functionality works in the same fashion as pre-commit code review, providing line by line comments and overall comments along with the ability to express an overall opinion on the commit. A comprehensive listing of these reviews is also available and can be searched.

Additionally it is able to support a form of scratch repositories with minimal modifications from sysadmin. At this time it is not able to support the way scratch repositories work, however depending on how a new feature is implemented by upstream we may be able to return to this way of working in the long term (minimum of 6 months). This would include full code review and browsing capabilities, making scratch repositories first class citizens within our Git infrastructure.

One of the most versatile, powerful and compelling components of Phabricator is known as Herald. Herald manages rules which can take a variety of actions - including adding people to the CC list for a review or flagging a commit for later review. This would allow reviewers to ensure they are always aware of reviews for their repositories, maintainers to be aware of changes being made to their repositories and may offer a solution for a Git-based commit filter.

Finally, Phabricator comes with a tool called ‘Arcanist’ which can be installed on developer machines. This functions similar to rbtools and allows for patches to be uploaded and updated, provides access to the list of tasks (also allowing adding and closing them) and other parts of the instance. It also has an extensive Web API called ‘Conduit’ which can be used to build tools to retrieve information from the system. This is the same mechanism used by ‘Arcanist’.

We have also been able to successfully engage with upstream. Development is extremely active at this time with numerous people involved and a number of the core developers are currently working on providing a commercial service around it. They were very responsive to our questions and gave helpful advice on how we may be able to implement functionality such as scratch repositories both now and in the future.

Finally we would like to note that the Phabricator interface should be easy for newcomers to adopt and understand, while still providing the power to do sophisticated operations (such as saved queries) should you wish to do so. It is also very usable on mobile devices such as phones and tablets.

It should be noted that Phabricator does not provide a replacement to the anongit system in any form. We will build this ourselves using new approaches we have thought over the past few years to hopefully eliminate the scalability problems. As this will require a deeper level of integration with the main Git repository management system, we cannot commence work on this until the future of git.kde.org has been determined.

Caveats

The following issues may interfere with our adoption of Phabricator. Please see the following upstream issues for more information. If necessary we are proposing working with other projects which are interested in these features in coordination with upstream to get these features added to Phabricator.

- Phabricator T4245 (Eliminating the requirement to have repository callsigns)
- Phabricator T5000 (Supporting direct use of ‘git’ for interacting with reviews)
- Phabricator T4333 (Preserving author metadata when landing changes)

Next Steps Forward

Should the proposal be accepted, sysadmin would like to invite projects to test out Phabricator to ensure it is able to handle the workflow patterns of KDE Developers. Such issues are to be expected of any tool we adopt, as only actual use can discover problems which may not be readily apparent. This will also provide sysadmin the necessary time to work on infrastructure integration problems and to assess how we would host it over the long term. We would also need to consider whether it is necessary to migrate any data from Reviewboard (in consultation with the community).

Given the complex nature of our SCM infrastructure and the number of systems involved we expect the entire evaluation and migration process (if successful) to take several months. Affected systems include anongit, the Continuous Integration system, LXR, Reports, EBN and api.kde.org among others as well as the ones Phabricator would replace.

Additional Notes

Sysadmin expects opposing proposals to be published. Such proposals must address the concerns we have about existing infrastructure, which will mean replacing projects.kde.org and its functionality (kde_projects.xml, Gitolite metadata generation, etc.) in some form. Additionally the community has indicated that scratch repositories must remain, an element which competing proposals must also deal with.

Proposals which consist solely of a code review solution will be considered incomplete and not accepted by the sysadmin team. Additionally, any proposal which contains components which have an uncooperative/hostile upstream or inactive development will be considered invalid as it will not provide a sustainable solution in the long run.