Paper TT06

Using Git and GitLab to Empower QC Workflow and Project Management in Clinical Reporting

Kieran Martin, Roche, Welwyn Garden City, UK

ABSTRACT

Git ® has been an extremely popular tool in software engineering over the last 2 decades. In this talk I will look at how we are taking advantage of this tool and Gitlab in the context of QC workflow. I will explore the advantages of the integrated project management tools, the clear audit trail, and the ability to collaborate effectively when using Git. I will discuss some of the difficulties of using this new tool, both in terms of the specific needs of our workflow, and the learning curve of Git, and how we have overcome these.

INTRODUCTION

At Roche over the last 4 years we have been transitioning to a new data science platform, which we call OCEAN. Part of this transition has been moving our data science stack to more modern, open source solutions. In this paper, I am going to focus on our adoption of Git as a tool for version control.

I will discuss what version control is, and why it is important in the context of clinical reporting. I will give a brief introduction to Git and Gitlab ®, what the tools are and why they are so popular. I will then discuss why we chose to use those tools. I will outline how we use those tools in our day to day work, and finish by discussing some of the challenges inherent with these tools.

VERSION CONTROL AND GIT

VERSON CONTROL

Version control is a solution for tracking the changes in software over time. It allows users to follow the evolution of a document or piece of code as it is altered during the course of project.

Using version control allows users to more easily understand the state of a project, and not worry about keeping old versions of code as additional files (myproject_v2, etc.). It can also provide more detail for auditors who wish to understand the history of a project.

GIT

Git is probably the most popular tool for version control in software engineering. A survey done by Stack Overflow (Stack Overflow 2022) recorded 93.87% of developers as using Git as their tool for version control. Git is a free tool developed for the Linux Kernel, but available on all operating systems. It was designed to be fast and reliable.

Unlike many other solutions available when it was developed, Git was developed as a self-contained solution. Provided a user has the software installed, they can conduct version control entirely on their own system, without reference to a central server. This makes development fast and easy, and in principle can continue without connection to a network.

Git works by taking “snapshots” of a set of files and folders (commonly called a repository). That is, rather than tracking individual files over time, Git tracks the state of files in relation to one another. This allows a fuller picture of how a project evolved over time, which is not always as easy with single file version control.

While Git can be used as a stand-alone tool, in practice users will want to collaborate with one another, and this is facilitated by the excellent web servers available.

GITLAB

Gitlab is just one of many similar web solutions for collaborating with Git. Fundamentally, it operates as a web server hosting Git Repositories, but offers many features on top of this

- Live rendering of code and documents

- Merge/Pull requests which allows line by line code review
- In built project management tools, such as issues, cross linking between issues and code, and Kanban boards to explore and manage these issues
- More recently, the development of in built continuous integration/development (ci/cd) tooling

The exact nature of how these features work will vary a little depending on the platform used. In this paper I discuss Gitlab because this is the tool being used in our internal solution, but most of what I say will apply to any of these platforms.

WHY USE GIT IN QC?

Within clinical reporting across the pharmaceutical industry we have developed quality standards we expect our clinical data, and the table listings and graphs to adhere to. In particular, we want to

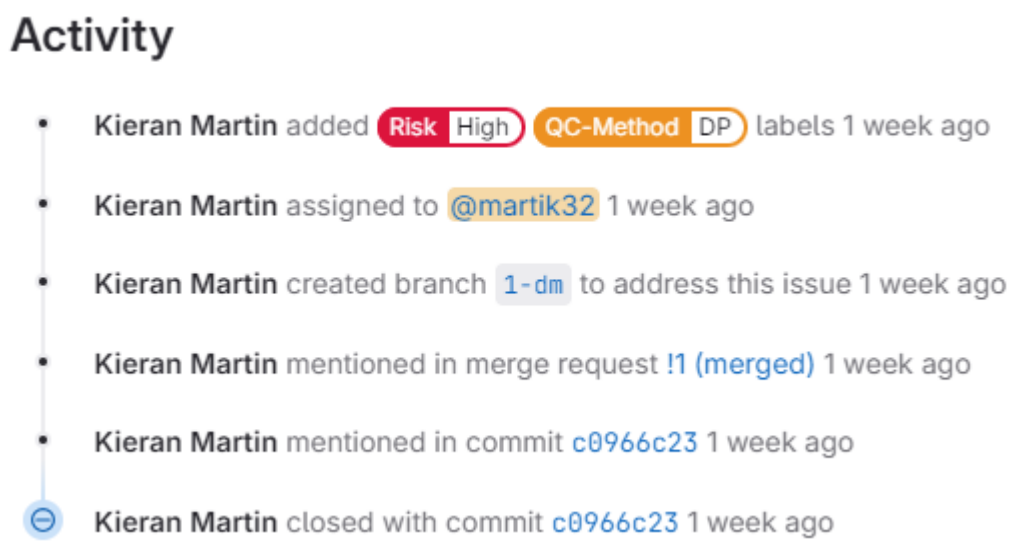
- Track what is being QCed, who Is performing the QC, and how they will be doing it
- Provide evidence that QC happened, and what the outcomes were
- Have an auditable history of our work

These core requirements were traditionally split across multiple different tools. Project tracking would be in one location, QC evidence another, and the code yet another. This made understanding what was done, and providing a clear picture when it comes to an audit challenging. In addition, because the system was quite inconvenient, compliance could be quite low.

CREATING A STORY

Gitlab provides us with the ability to sync all of what we are doing in one place. Code development, QC tracking and project management can all be done in one place.

One of the most powerful benefits we get from this is a clear story of the code for a particular project over time. We can use “issues” to track individual pieces of work. Each of those issues will be assigned to the first line programmer, and will use “labels” to indicate what level of risk is and QC method. Then when code is introduced, the changes can be cross linked to the issue. See the image below for an example



This simple bit of functionality; the ability to cross link between the code being added and the problem being solved, makes it very easy to understand when code was added and why.

These kinds of benefits go beyond purely meeting regulatory requirements; it makes it easier to understand the work colleagues are doing, and the work we did previously; we can understand why changes were made at a particular point in time.

Note that in Git every single commit (snapshot of the code) must be associated with a message. This allows even more information about why changes were made at a particular time.

BEING FAIR

One goal as we moved to a new data science platform is making our data Findable, Accessible, Interoperable and Reusable (FAIR). Part of doing this relies on code which empowers these standards.

One key change switching to Git has empowered has been the separation of code and data. Putting code on Gitlab makes it much easier to find; Gitlab search is very powerful, and we have additional metadata first driven solutions which ensure that we understand what work is being done where. This separation also means that code can be more exposed; previously, when data was stored with code, only individuals who had permissions to read clinical data could see it. This was an unnecessary restriction, and now we can share more readily with our colleagues.

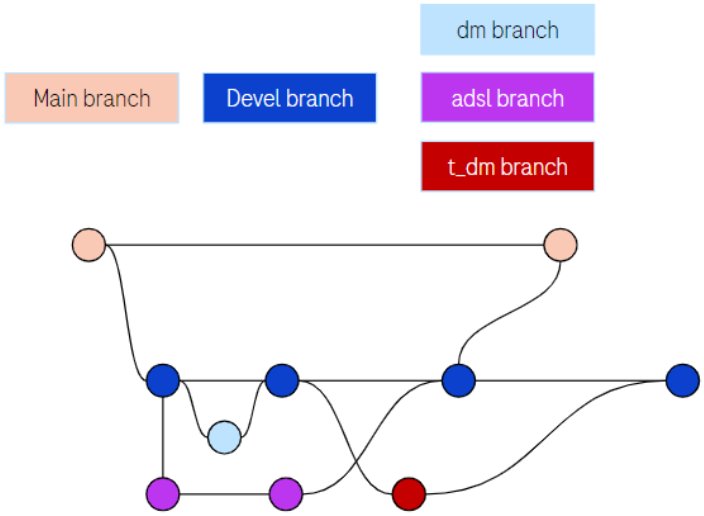
Finally an important goal was to make our work reproducible. Git as a tool is excellent for empowering the reproduction of work, storing as it does snapshots of code over time. Gitlab also allows the creation of “releases”, special snapshots which can be tagged to indicate that they were used for a particular delivery. These kinds of changes we think will reap long term benefits when we need to recreate work that was done many years prior.

HOW WE USE GIT

In this section I will give a brief summary of how we use Git in clinical reporting.

STANDARD WORKFLOW

For each delivery, we create a code repository, with issues tracking the work to be done. In particular, we have issues for each SDTM ®, ADaM ® and tlg. Those issues are labelled to track who will be working on the code for this particular deliverable, and what level of QC is required. In particular, we flag where a piece of work is low or high risk, which will indicate if it needs double programming or not.



On each repository, we have two permanent branches, devel and main. The expectation is that code on both devel and main have completed QC. Main should have production code; code which has completed QC and is ready to be used to deliver final results. Devel exists to confirm that the code has worked in an integrated fashion; once each deliverable is coded, that all of the code works together with no issues.

Code development then primarily happens in feature branches, named after the issue they are created for. Both first line and second line programming should happen in these branches. Any QC comments should be done on the associated merge requests, to allow a clear story of what changes were made.

If a deliverable is determined to need updates after completion, the related issue is reopened, and a new feature branch is created to address these issues. This does mean that there may be multiple merge requests associated with a particular issue, but we believe this is a feature; by splitting different changes into clear deliverables, the flow of work is much easier to track. The associated merge requests will be linked to the original issue to ensure that the evolution of the program over time is easy to track.

ENSURING CONFORMANCE

For this approach to be successful, we need programmer to comply with the framework as defined. We have a number of ways that this is enforced.

- Training and process. There is a clearly defined process for how individuals must work, and training is provided to each individual to ensure understanding. We also provide tools to help set up and ensure that standards are followed.
- Checks and balances. We have a number of checks to protect against accidents.
    - The main and devel branch are protected from being pushed to directly
    - Delivery of final work can only be done with code on the main branch
    - Git hooks to encourage people work correctly locally
    - Our solution for running code does a number of checks to ensure code meets the expected quality

CHALLENGES WITH USING GIT

Switching to a new way of working is not without challenges. I will discuss a few of these in this section as well as how we are looking at mitigating them.

LEARNING GIT

The majority of our users have not used Git before, and it can be quite tricky to learn. There is a very steep learning curve as you begin using the tool, as you must understand some quite novel concepts. During the beginning of the transition we saw many issues as users who simply did not understand the tool started making quite understandable mistakes.

We have tried to resolve this in a few ways

1. Creating tooling and processes which made following our workflow more easy. These “guard rails” help protect users from the beginner errors someone might fall into
2. Providing training

We developed in house training on both the subject of Git more generally, but also how we used it. We made sure when developing this material to do it in such a way that it could act as a resource in future. While we did prioritise in person training sessions, we wanted to ensure that for future joiners the material would also be available for offline learning.

We did this by building comprehensive slide decks paired with making recordings explaining each section in simple terms. We made sure that this material was partnered with comprehensive exercises which allowed users to practice and embed their learning.

While we are still having challenges, we are noting our user base becoming more comfortable with using Git and believe that this will only improve with time.

DETERMINING THE BEST WORKFLOW

One real challenge has been the design of Gitlab. Gitlab was designed with software engineering in mind, where some of the specifics of how we work is very particular to our industry.

This has led to some difficulties when it comes to certain aspects of how we work. For example, the fact that when one branch is merged into the other, the full content is brought through can be difficult when the QC of code may not happen straight away. The solution we opted for was to require QC to be complete before a deliverable is pushed into

the devel branch. That solves one problem, but does create another, as dry runs become harder to do if QC is not completed.

Some of these difficulties are driven by the current paradigms within clinical reporting. We may look at changing how we work in future to reflect new practices and technologies. In particular, moving away from double programming, and focusing more on code review, unit testing and assertion based programming. This would be more in tune with software engineering, and could help us move towards more automated workflows in future.

We have already updated our workflow based on feedback once, and expect to do it again in future; using Git for QC is a novel application, and we anticipate changes over time as we get a better understanding of what works best.

CONCLUSION

Git as a software allows the creation of a rich history which both helps us meet our regulatory requirements, but also build a better understanding of our own code base. Using it will make our work more reproducible and findable, and we are confident that it is the correct tool to use in the future.

There are challenges inherent in using Git with the QC process, but also opportunities. We will continue to explore how to best use Git, and also how we can update our general processes to take advantage of developments within and outside of our industry.

REFERENCES

Stack Overflow 2022,*Stack Overflow website*, accessed 16/10/2024
https://survey.stackoverflow.co/2022/#technology-version-control

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Kieran Martin
Roche
Hexagon Place, Shire Park, Falcon Way, Welwyn Garden City AL7 1TW
Kieran.martin@roche.com