Bausteine Forschungsdatenmanagement
Empfehlungen und Erfahrungsberichte für die Praxis von Forschungsdatenmanager*innen

Combining cloud and Git tools in a research data management strategy for team science.

Julien Colomb[i] Robert Mies[ii]

2026

Zitiervorschlag

Colomb, Julien und Mies, Robert. 2026. Combining cloud and Git tools in a research data management strategy for team science. *Bausteine Forschungsdatenmanagement. Empfehlungen und Erfahrungsberichte für die Praxis von Forschungsdatenmanager*innen* Nr. 1/2026. DOI: 10.17192/bfdm.2026.1.8721.

[i]Humboldt Universität zu Berlin, Technische Universität Berlin. ORCID: 0000-0002-3127-5520
[ii]Technische Universität Berlin. ORCID: 0000-0003-0195-7405

Abstract

In project management of collaborative research projects, there is an increasing demand for an information and communication technology (ICT) infrastructure because people are distributed in time and space, including a safe and legal data sharing infrastructure. Data managers want to foster the production of FAIR data by implementing best research data management (RDM) practices. In practice, the use of a cloud service is the easiest to implement, but shared folders tend to become huge and unorganised, which greatly limits the findability and reuse of the shared data. In order for data managers to regularly monitor activities on the shared folder, they need a better version control system than what cloud systems provide.

Here we present a strategy where the data manager uses the power of Git on a local copy of the shared folder. By spotting new and modified files, they can intervene very early and pro-actively to keep files organised, produce useful metadata, or publish data on behalf of the researchers. In particular, the data manager can move large files outside of the data synchronised on the researchers' computers. This strategy was successfully implemented in two projects that collect relatively small datasets. Because most of the collected data is available and organised, publication and archival of the whole data can be performed by the data manager, potentially making this data FAIR during and after the project.

1 Introduction

Since research teams in collaborative projects are distributed in time and space, online tools for communication and data sharing are necessary. Proper data management in research that includes the monitoring and improvement of data quality is particularly efficient (Lowndes et al., 2017; The Turing Way Community, 2022). To this end, data managers should timely intervene in the data organisation, its formatting and its annotation, in order to make the data more meaningful, and facilitate interactions between project members. This requires access to the data and interactions with the researchers, at the latest when new data is collected.

The use of version control, especially Git and Git forges like GitLab, has been proposed to enable better research data management (Cyra et al., 2022). However, in practice, the need for technical knowledge that allows the effective use of Git collaboratively is a huge barrier, it lacks common document editing functionalities and it is pretty impossible to get every member of a large team to use Git. In particular, synchronisation is a manual task such that the use of Git needs a continuous and active participation of every team member.

In contrast, the use of cloud systems, where digital files are automatically uploaded to a server, or directly saved on the server, is an appriciated tool to share data (see

for example Brzeźniak et al., 2017; Labrador et al., 2019). A fairly large amount of the data can actually be then accessed by the data manager. However, the version control in these systems is usually not very developed and it is very difficult to monitor what is shared and when. It is therefore difficult to give timely feedback. At the time the data managers are aware of the new set of data, its sheer size can make it difficult to modify its organisation or format.

Figure 1 summarises the pros and cons of using Git versus cloud systems in research consortiums. By combining the two approaches, we developed a strategy to facilitate collaboration and the production of FAIR data. We present here two use cases that illustrate both the advantages and limitations of this approach, where a special care needs to be taken with large files management.

Features and issues	**Git and Git forge**	**cloud tool**
Technological knowledge requirement	-	++
Access, backup all files	-	+
Big files support	-	+
Synchronisation between local computer and server backup	+ (manual)	++ (automatic)
See what is new	++	-
Access previous versions	++	+
See differences inside files	+	-
Collaborative writing	+	++ (integrated collabora)
Project management tools	++	+

Figure 1: Overview of the tool advantages and problems. A scale from 0 (-) to two (++) represents the relative quality of that feature for each tool, as estimated qualitatively by the authors.

2 Concept

Researchers are using the cloud solution, accessing data via the browser or directly on their local computer using a specialised application (blue square in Figure 2). The folder organisation is meant to facilitate both team work, and give the possibility for the users to restrict synchronization to folders they are working on. For instance, researchers should not synchronise the folder for big files to limit the amount of data storage needed on their computer (in Figure 2, note the folder is empty on the local version shown on the right side of the figure).

On the other hand, the data manager is using Git locally to follow modifications in the shared folder (orange square in Figure 2). Using a Git management software, they can easily control data quality and organisation and proactively move files, or modify data format or file names in discussion with the researchers (orange arrow in Figure 2). Once cleaned, modifications are 'commited' and pushed on a Git forge, which creates an extra backup of the files and their history. Large files are not added to Git and backed up independently on a HD drive. The figure shows Nextcloud and GitLab tools, because these are the tools we used in our implementation. One could also build a similar workflow using Dropbox or Seafile (Brzeźniak et al., 2017) as a cloud tool, and GitHub, or a self hosted Gitlea or Forgejo as a Git forge.

2.1 Researchers' work

At the start of the project, everyone is given access to the cloud server, filled with a pre-defined folder organisation, and asked to upload relevant files in it. By sharing project management and meeting minutes files in one place, it is easier to organise team work. Files can also be shared with collaborators who would not have access to the cloud tool by using the "share file" functionality of Nextcloud. By receiving early feedback on their file naming, folder organisation or file format, they can produce data that is easier to re-use. Collaborative editing of text and spreadsheets is easier done online directly in the cloud tool for instance with Collabora (the online version of LibreOffice; Timmar 2020), or elsewhere and archived on the cloud system.

In a second phase, researchers should use the synchronisation tool and share all project files with each other and with the data manager. Team members' main motivation may be to read files while being offline, to automate file backup or sharing, or to get early feedback from the data manager. Since they are usually involved only in specific tasks in the project, they can synchronise only part of the dataset and reduce the data storage needed on their computer, as well as their network usage.

If they need to access the history of the files, the data manager can lead them through all versions available in the Git system, via the available curated history of the files on the Git forge. Importantly, the Git history of the repository is not shared via the cloud,

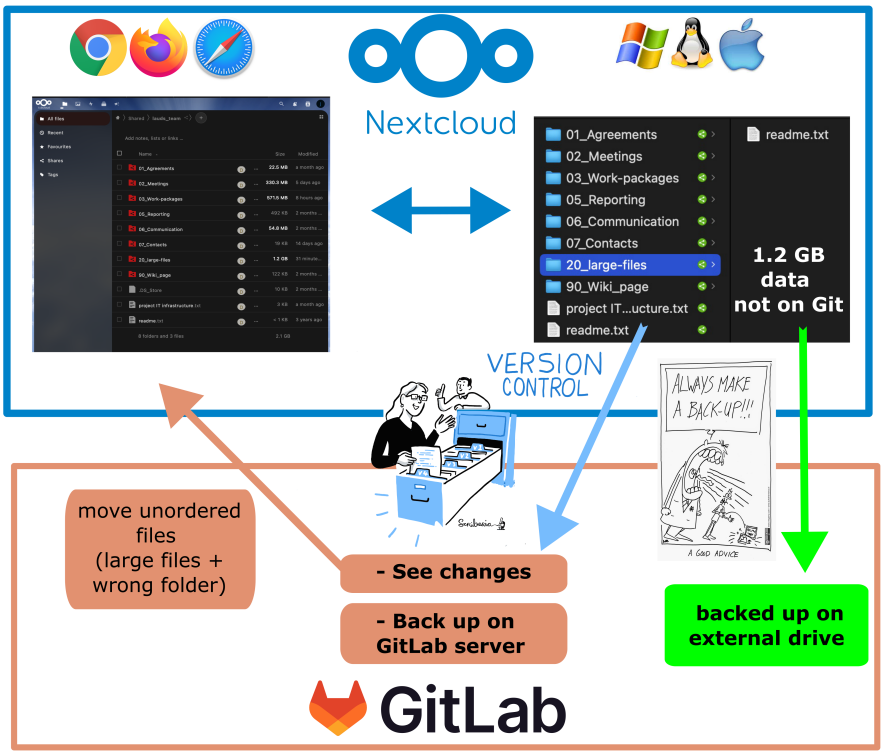

Figure 2: Overview of the RDM strategy. On the top in the blue square, team members use Nextcloud to upload, download and access data either via their browser (left) or via a synchronized folder on their computer (right). On the computer, one can choose what folders will be synchronized. In this example, the “20_large_files” folder contains 1.2 GB of data online, but is empty on the computer. Below the orange square represents the work of the data manager who uses Git version control. They can see changes thanks to Git and propose or make changes in the dataset using the cloud version. As an extra backup, the Git repository is pushed to a GitLab instance and larges files are saved on an extra drive. Image credits: Herrema, 2014; The Turing Way Community and Scriberia, 2020; both CC BY.

and researchers do not use Git. If researchers want to use Git, they may use the Git forge directly, in discussion with the data manager.

2.2 Data manager's work

At the start of the project, the data manager provides a folder structure template, manages the cloud and Git forge infrastructure choice and access, but not its maintenance. The choice of the infrastructure should be made in accordance with the institutions involved in the project and follow local legal requirements. When setting up Git on their machine, the data manager should keep the large ".git" folder outside of the shared cloud folder, using the `---separate-git-dir` option of `git init`.

During the project, they regularly monitor changes in the dataset and react accordingly. In discussion with the researchers, they may change file names and formats, add metadata, or follow up in an absence of data on the cloud. When moving files, they should use the cloud functions in the browser to limit network usage and prevent share links stopping to work. They then curate the file history by adding a descriptive message to their Git commits, and run the "git gc" command to reduce the size of the .git folder on their machine. The data manager should also convince and teach researchers to use the cloud application and save all their files.

2.3 Data management

The described strategy allows the cloud data manager to be directly in charge of data backup, publication and archival. They have themselves three copies at three different locations: one on the cloud server, one on the data manager's computer and one in the Git forge. The data manager can also organize specific folders for data that will be published. They can also prepare the archival of the data that will happen at the end of the project. By acting early and having access to most of the data, the chances that the data can be made findable, accessible, interoperable and reusable (FAIR) improve.

With time, the size of the dataset will grow, especially when people share large files, like photos and videos. These files should not be added into Git, and also not be synchronised on every computer via the cloud tool. Therefore, the strategy provides a specific folder for these files. This folder is ignored both by the cloud synchronisation tool and by Git, and is backed up independently. A link to the file is left at the old location, such that the researchers know the file exists on the server.

Importantly, the strategy does not depend on the usage of text-based files (markdown for text, comma separated values for spreadsheets), which would work best with Git. Here, most of the collaborative writing happens in cloud text editing tools, and binary files are shared or backed up in the cloud tool. When these files are small, the data manager is commiting them to the Git system, as it is important to see when they were

modified. This does not increase the size of the shared folder, as the Git history is kept outside of it.

While the workflow does not per se induce the production of FAIR data, it facilitates the work of the data manager toward that goal. On one hand, the data manager will have access to datasets early enough to provide feedback at the beginning or even before data collection. This can facilitate the use of open data formats, the implementation of standard terminologies, and the production of metadata early in the research process. This process potentially makes the data easier to analyse as well as more interoperable and reusable.

3 Implementations in two projects

We have been using this strategy in the relatively small Berlin University Alliance project Open.Make (started in 2019) and the larger EU funded LAUDS Factories project (started in January 2024). We have been using the infrastructure of the TU and FU Berlin (Nextcloud and Gitlab instances). At the end of the project, we will archive the data on a long time archive solution provided by the TU. The amount of data for both projects reaches 27 GB which is saved in the same Nextcloud instance. The data manager synchronises only about 3 GB of data on their local copy, using a Git UI software (sourcetree) to monitor and commit data modifications on a single computer. Git was initialized with the '–separate-git-dir' option to save the large ".git" folder outside from the shared folders and have two different Git projects. Importantly, the Open.Make project has data absent of the cloud system and present only in Git forges (GitHub and Codeberg), as specific sub-projects require a fine-tuned version control.

In both projects, the data manager is caring for the repository about once every two weeks. While most issues are related to moving big files to the large file folder, we have also encountered problems when derived data instead of raw data was shared, an inefficient data format was used (an 84 MB excel file could be replaced with a 130 kb open office file, large images were present in office files for no good reason, spreadsheet design could be reviewed), data was duplicated, or empty folders were uploaded (the files supposed to be inside were absent). This can be well tackled when the data manager checks the size of the files newly added or modified.

The Open.Make team is small and local, so we discuss everything in person and save all our data in the cloud system. In contrast, we provided the LAUDS Factories team with a walk-through in the form of a public wiki hosted on the GitLab platform. The wiki presents all IT tools used in the consortium. We also organized onboarding sessions and used the data management plan meetings to stress the need for saving all files in the cloud. One-to-one meetings are organised for the use of the Nextcloud application.

In the LAUDS Factory project, the data manager created a specific folder for data to be published. This folder is linked to a second Git repository, which is linked to Zenodo. Data related to our first deliverable could then be published (Colomb et al., 2024) without the involvement of the researchers, apart from their agreement to the publication. At the end of the project, data which will not have been published will be archived in a single archive.

4 Conclusions

4.1 Advantages

The strategy allows for an automatic backup and sharing of data online, and for a version control system creating an annotated history of changes. All team members can have access to the whole dataset, and the data manager can use the history to keep track of data uploads and control its quality. This also allows the data manager to access and backup all data collected and prepare data publication and archiving.

4.2 Limitations

While Nextcloud is easy to use, users still have to build new workflows and habits. Agreement on data organisation is sometimes difficult to achieve and requires to invest time in communicating with the researchers. On the technical part, users need to get access to the central cloud tool, which may be time-consuming. Nextcloud is also not meant to be used to share huge amounts of data, and this strategy should only be used for relatively small datasets

Because there is no fine-grained data access restriction, this strategy requires team members to agree with sharing their data with the whole team. In some cases (unwilling researchers or personal data collected), only a fraction of the data will or can be shared.

4.3 Lessons learned

It is important to regularly assess the data shared and move large files outside of the core data controlled with Git and synchronised with every researcher's computer. Indeed, when the amount of data grows, people would otherwise be synchronizing unnecessary files.

It is not realistic to implement the use of text-based workflows that would work better with Git. Collaborative text editing is best done with specific cloud software, either using a tool available in the cloud system (in our case Collabora in Nextcloud), or using an external tool and saving copies in the cloud system.

The local history size may be reduced using the "git gc" command. It should be backed up outside of the shared cloud data, using the '–separate-git-dir' option. (Other strategies only allow one project to follow this strategy).

Communication with the researcher is both primordial and the most difficult part of the work of the data manager. While this strategy lowers the technical barrier for data backup and sharing, one still needs to convince researchers to change their habits. The publication of datasets demonstrates the advantage of the strategy and each time the data manager can point out and solve data issues, the data quality increases, making datasets FAIR, one advice at a time.

Funding

RM and JC were funded by the European Union, 2024-2026, GA 101135986 LAUDS Factories, and by the Federal Ministry of Education and Research (BMBF) and the state of Berlin under the Excellence Strategy of the Federal Government and the Länder / www.berlin-university-alliance.de inside the Open.Make project. JC was funded by the DFG – project number 327654276 SFB 1315.

References

Brzeźniak, M., Wadówka, K., Woszuk, P., & Meyer, N. (2017). Storage back-ends for scalable sync & share based on seafile. *Zenodo*. https://doi.org/10.5281/zenodo.439564

Colomb, J., Mies, R., Hofer, M., Unterfrauner, E., & Gurgurovci, S. (2024). Lauds factories open data (version 2.0) [data set]. https://doi.org/10.5281/zenodo.15535394

Cyra, M., Politze, M., & Timm, H. (2022). A push for better rdm: Erfahrungsbericht aus dem einsatz von git für forschungsdaten. *Bausteine Forschungsdatenmanagement*, *2*, 1–17. https://doi.org/10.17192/bfdm.2022.2.8435

Herrema, A. (2014). No title [drawing]. https://aukeherrema.nl/

Labrador, H. G., Alexandropoulos, G., Bocchi, E., Castro, D., Chan, B., Contescu, C., Lamanna, M., Lo Presti, G., Mascetti, L., Moscicki, J., Musset, P., Karavakis, E., Pelletier, R., & Valverde, R. (2019). Cernbox: The cern cloud storage hub. *EPJ Web Conferences*, *214*, 04038. https://doi.org/10.1051/epjconf/201921404038

Lowndes, J. S. S., Best, B. D., Scarborough, D., Afflerbach, J. C., Frazier, M. R., O'Hara, C. C., Jiang, N., & Halpern, B. S. (2017). Our path to better science in less time using open data science tools. *Nature Ecology & Evolution*, *1*(6), Article 0160. https://doi.org/10.1038/s41559-017-0160

The Turing Way Community. (2022). The turing way: A handbook for reproducible, ethical and collaborative research (version 1.0.2). https://doi.org/10.5281/zenodo.7625728

The Turing Way Community & Scriberia. (2020). Illustrations from the turing way book dashes. https://doi.org/10.5281/zenodo.3695300

Timmar, A. (2020). Integrate collabora online with web applications [video]. https://doi.org/10.5446/47479