Version control and DevOps for accelerator and experiments: experience and outlook

Ismael P. Trobo[1*], *Konstantinos* Evangelou[1], *Alexandre* Lossent[1] and *Andreas* Wagner[1]

[1]CERN IT Department, PW Group, Esplanade des Particules 1, 1217 Meyrin, Switzerland

Abstract. GitLab has been in operation at CERN since 2012. It is a self-service code hosting application based on Git that provides collaboration and code review features, becoming one of the key infrastructures at CERN. It is extensively utilised at CERN, with more than 17 000 active users, hosting more than 120 000 projects and triggering more than 5 000 jobs per hour. During the initial stages, a custom-made solution was deployed. However, with the exponential increase of projects, workflows, and continuous integrations, the GitLab infrastructure became hard and complex to scale and maintain. The recent migration, which involved adopting a new supported Cloud Hybrid infrastructure, has enabled CERN to align its GitLab infrastructure with industry standards and best practices. This has resulted in a significantly more robust and high-performing infrastructure, leading to notable benefits throughout the entire deployment process. This paper will address how this deployment process, on the road to success, has presented a series of challenges and pitfalls that have been faced during this complex migration process.

1 Introduction

GitLab [1] is a self-service code hosting application based on Git that provides collaboration and code review features for users and CERN service providers. It is one of the critical infrastructures running at CERN, being widely used daily for more than 17 000 users, and hosting more than 120 000 projects, fitting the needs of the entire community at CERN.

Initially, a custom-made infrastructure was used with success for several years and proved to be stable and reliable. However, in view of the end of life of several underlying components, and the maintainability cost increase over the past years, it was decided to start planning a migration to a more modern infrastructure, and to take the opportunity to align it to the recommended guidelines and best practices provided by the supplier GitLab, Inc.

2 Infrastructures Overview

In its initial stage, the GitLab application has been running in an OpenShift version 3 [2] cluster with dedicated nodes. It has been a custom-made deployment built several years ago when a Helm [3] deployment was not offered by GitLab, Inc. [4], consequently neither

* Corresponding author: ismael.posada.trobo@cern.ch

supported nor aligned with upstream recommendations. Several components have been externalised, such as the database, based on PostgreSQL [5] and provided as a service by the DBoD Team (DataBase on Demand) at CERN, as well as the NFS (Network File System) storage and the S3 solution that runs on Ceph [6], that follows as close as possible the S3 protocol specifications provided by the Storage Team at CERN. Several of these components reached the end of life back in 2022.

The final production setup after the migration advocates for following a similar architecture in terms of computational resources, however aligned with the requirements and recommendations provided upstream [7]. This final production setup uses, however, Magnum Kubernetes [8] provided at CERN as the base cloud infrastructure, CephFS [9] as the storage system provided by the Storage Team at CERN and the database system remained as is (figure 1).

The sum of all these systems forms what we call the GitLab infrastructure at CERN.

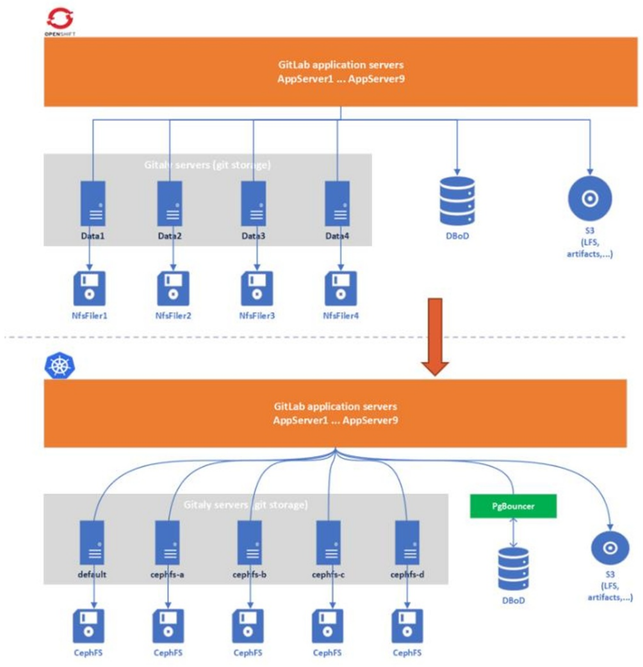

Figure 1. Pre and Post migration infrastructures overview.

A series of challenges had to be overcome, since it was desired to minimise the downtime foreseen at the time of moving all components and all git repositories from one infrastructure to another. There was also the wish to preserve the availability, data integrity and operations running while the migration was being performed, avoiding a major disruption of the service during a period of hours, preventing CERN's community users moving on their workflows.

This level of availability and data integrity was accomplished by having both infrastructures running at the same time, and key components of the GitLab application interconnected between one infrastructure and the other (figure 2).

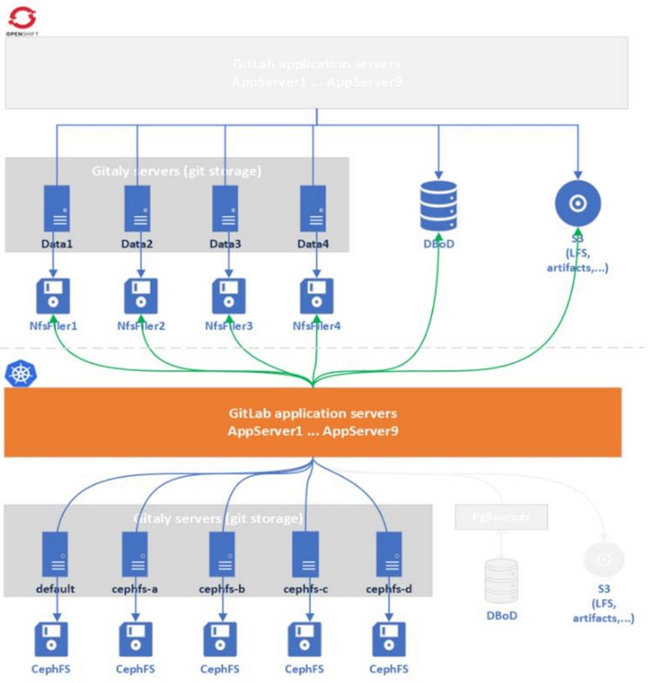

Figure 2. GitLab infrastructures with interconnected components.

3 Migration process

The migration process to a brand-new Magnum Kubernetes cloud-based infrastructure, with the goal of benefiting from high performance, cost efficiency, high availability, among others [10], can be split into three well-differentiated parts: pre-migration steps, migration of the GitLab application and migration of the repository data.

3.1 Pre-migration steps

Pre-migration steps were extremely important to guarantee the availability and the reliability of the solution, and to minimise the impact of the downtime foreseen for the migration of the application at the time of starting migrating all components.

The steps followed in this stage were mainly to provision the new Magnum Kubernetes cluster with the minimum components needed to make a smooth transition, namely the Redis [11] Sentinel cache deployment and the Gitaly [12] deployment, together with five persistent volumes [13] linked to the CephFS storage infrastructure. It was also preserved the same number of computational resources as in the infrastructure based on OpenShift 3.

On one hand, the Redis deployment has been deployed internally within the OpenShift 3 cluster and not exposed to intranet nor internet, i.e., only accessible within the cluster, therefore the need of deploying a separate deployment in the new Magnum Kubernetes cluster was needed. In future migrations, a consideration is being made to externalise and decouple this component making it reachable at least in the intranet area, thus reducing even more the time needed to switch between infrastructures and especially the number of manual actions, thus minimising the downtime expected and the risk of the operation.

On the other hand, a Gitaly deployment has been deployed to facilitate making these storages available and viewable by both the infrastructure based on OpenShift 3 and the one based on Magnum Kubernetes. This was possible thanks to the use of the Endpoints [14] and the NodePorts [15] Kubernetes resources, aiming at exposing these storages outside the Magnum Kubernetes cluster, but only viewable from the intranet.

Apart from that, a tweak has also been made to the GitLab Workhorse [16] component, by pointing it to the root hostname instead of the internal service, in order to make this component aware of the extra storages attached, located outside the OpenShift 3 cluster. As per the official network architecture for the Gitaly component [17], and to be able to move repository's data between the different storages, every single Gitaly storage must be interconnected and cross-communicated.

3.2 Migration of the GitLab application

The GitLab application is a single application that provides a complete DevOps lifecycle solution, that is composed of several components interconnected between them. A simplified architecture diagram as in [18] can be used to understand the GitLab architecture (figure 3).

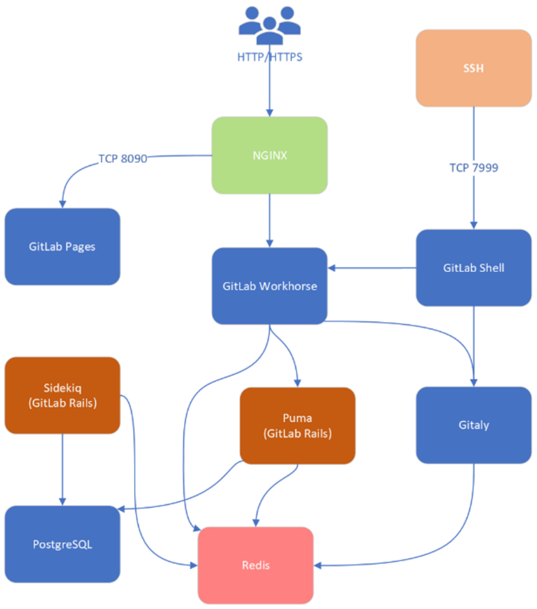

Figure 3. GitLab application high-level architecture

At a glance, the impression given is that migrating the GitLab application from one infrastructure to another looks like an easy process, however its main constraint comes with the complexity of moving the Redis data between infrastructures, among other complex operations explained in detail afterwards. This process must be accurately scheduled to avoid any data loss and to make sure Redis data contains the latest cached data produced by the GitLab application running on OpenShift 3 infrastructure. It will represent the only scheduled downtime that the GitLab application will show up and is meant to last the time it is spent between copying data over from two infrastructures.

To accomplish this, the GitLab application running on OpenShift 3 was stopped for approximately twenty minutes, in order to prevent GitLab to keep generating more cached data. At that moment, Redis saves a snapshot of the data automatically, therefore as of this moment, the Redis snapshot can be fetched and moved to a different instance of Redis, this time running in the new infrastructure based on Magnum Kubernetes.

Once this Redis data movement has been achieved, it is time to bring the Magnum Kubernetes infrastructure up and start running the service again.

3.2.1 Migration of the GitLab application: errors, bugs and pitfalls

The process of migrating the GitLab application incurred several unforeseen errors, bugs and pitfalls that were unfortunately only detected and spotted in the production stage.

Several indentation problems of the configuration files were not spotted in due time, leading to default values being applied for diverse components. This led to some of the components running properly on development instances, but not being able to handle appropriately the load occasioned in the production environment infrastructure.

Keytabs [19] of some servers were absent while configuring Kerberos due to a misconfiguration issue, leading to users not being able to work appropriately with Kerberos and executing common git actions such as, git push, git clone, to name a few.

Configuration of the Nginx web server had to be tuned up and accommodated by allowing larger client buffers and increasing body sizes for requests on-the-go since the load present in the development environment did not require such extra configuration in place.

3.3 Migration of the repository data

As previously stated, a prerequisite to execute this migration was to have both Gitaly storage deployments up and running, and cross-communicated.

To accomplish the migration of the repository data, internal tools [20] provided by GitLab have been used, such as the GitLab API. Four hundred projects were initially successfully migrated manually. Once this process was verified and the integrity of the data was guaranteed, a bulk operation for the remaining projects was executed to move repositories from the NFS storage provided by the infrastructure based on OpenShift 3 to the CephFS storage provided by the infrastructure based on Magnum Kubernetes.

3.3.1 Errors, bugs and pitfalls

The second phase of migration encountered one of the most challenging errors, the implications of which were unanticipated.

To migrate the first bunch of projects showed no evidence that something was going wrong in the process. However, at the time of initiating the bulk migration of the remaining projects, and therefore making the new GitLab application running on Magnum Kubernetes start acquiring a significant increase of load, a series of reboots started happening under the nodes hosting the Gitaly deployment.

The initial analysis of the causes of the reboots did not show anything clear, but it was clear that nodes running the Gitaly component were rebooted for unknown reasons. A posteriori analysis revealed that we have been hitting a kernel bug [21] specifically for the version of Fedora it has been running.

This bug provoked the corruption of a huge number of repositories and led us to start yet another analysis about how to mitigate such a big issue. On a preliminary basis analysis, it was discussed the possibility of acting two ways:

- Either setting the application in maintenance mode, therefore writing operations in disk would have been stopped, aiming at minimising the risk of repositories corruption.
- Or upgrade the kernel version in place as a short-term solution, while a long-term solution was being analysed in parallel.

A long-term solution is proposed for creating a separate Nodegroup [22] with a specific version of the kernel, allowing the corruptions to cease and therefore to mitigate the issue.

4 Stabilization and beyond

Once the above issues had been resolved, we proceeded to stabilise the Magnum Kubernetes infrastructure, looking for any sign of instability, if any.

This process helped us improving our Monitoring system based on the ELK stack [23], giving the administrators a clear view and more insight into things happening behind the scenes. Monitoring in the OpenShift 3 infrastructure could have been better and due to the magnitude of the change, most of the metrics used back then needed to be re-adapted. This change and adaptation revealed yet another issue concerning the performance of the database used by the GitLab application: a snapshot scalability issue.

4.1 Snapshot scalability issues

Benefiting from the Monitoring system set in place, a series of peaks have been identified in the graphs coming from the underlying database system (figure 4).

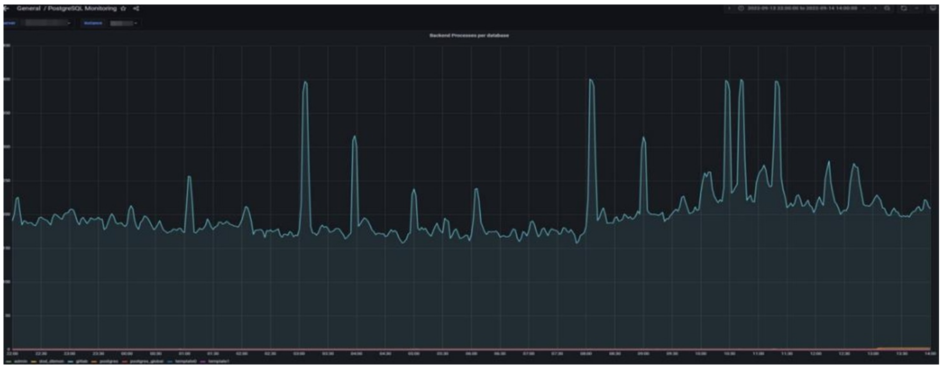

Figure 4. Backend connections peaking up suddenly in the database server.

These peaks represent the number of backend connections in the database server, huge spikes happening in a short period of time, mainly seconds, and all of a sudden, provoking, among many other things, a loss in performance at the database level, as well as a degradation of the user experience when accessing the main instance.

With the goal of understanding and reducing these peaks, an investigation was initiated in consonance with the Database at CERN and GitLab support teams, so as to find the root cause of these spikes. Different components were analysed and benchmarked to ensure that our systems were fully aligned with the specifications that GitLab, Inc. provided for customers.

After six months of daily investigations, analysis, comparisons, etc., it was pinpointed that there was evidence that it had been hitting a snapshot scalability issue. As stated in [24], snapshot scalability is related to the database's handling of concurrent transactions and its use of MVCC (Multi Version Concurrency Control).

4.1.1 Mitigating the snapshot scalability issues

To mitigate snapshot scalability issues, it is essential to minimise the impact of long-running transactions, to emphasise the optimization of query execution, excessive dead row accumulation, as well as reduce the number of snapshots created per transaction.

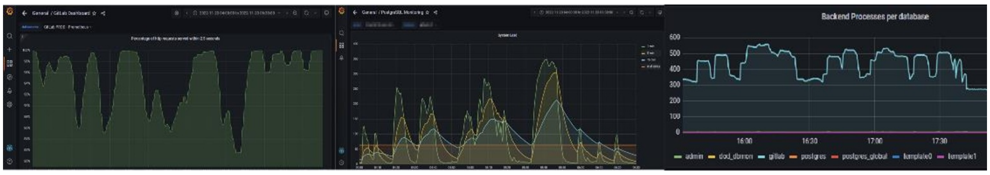

Figure 5. Consequences of a snapshot scalability issue: low rate of http requests served and poor performance at the database level.

There are many solutions widely discussed to help mitigating these issues, such as proper indexing, implementing appropriate data archival and clean up strategies, tuning the database server, especially the auto-vacuum configuration, and implementing a transaction pooling mechanism, to name just a few. These are considered key solutions [24].

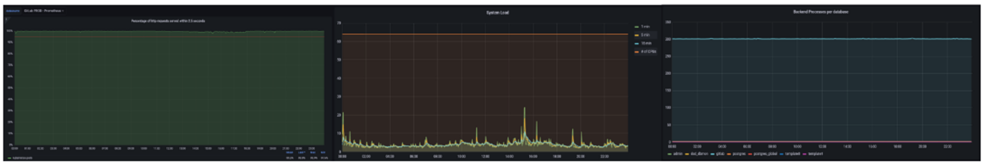

Figure 6. Result of tuning the database server and use of a transaction pooling mechanism.

Several techniques, aiming at delivering a high performant database [25], have been applied with successful results as shown in the comparison between figures 5 and 6, where there is clear evidence of the improvement in terms of performance gained by the database server linked to the GitLab application. All these techniques and know-how have been transmitted to GitLab, Inc. via its support line, in order to allow their customers to benefit from it and to serve as a case study. A contribution made by CERN is still ongoing at the time of writing this paper for GitLab, Inc., to start offering our implementation of the transaction pooling mechanism widely, as can be seen in [26].

In brief, tuning the database server, and using a transaction pool mechanism has helped minimising and reducing the load of the database server, improving substantially its performance, and enhancing the user experience for the CERN community. This has been also accompanied by setting some throttling at the application level, in order to avoid a misuse and/or abuse from the community, that parallel investigations revealed and helped the GitLab administrators to contain the load of the infrastructure, in benefit of an improvement of the general user experience.

5 Conclusion and future work

After having migrated our GitLab application to a more modern cloud-based infrastructure based on Magnum Kubernetes, and after aligning with upstream recommendations and best practices, it can be confirmed that unavailability has been substantially reduced and performance and reliability of the service has been improved, permitting a smooth and soft user experience that has been enhanced notoriously since then, improving the quality of the service.

This reduction of the maintenance and performance-gaining has contributed to several improvements within the team, such as freeing up resources, making procedures easy to maintain and follow, due to its proximity to the ones provided upstream; and, most importantly, to allow the members of the GitLab service at CERN team to focus on new features in order to keep ensuring a good quality of the service, being ready for the upcoming challenges of the future.

A clear roadmap has been elaborated, with special emphasis on the scalability of the solution and especially due to the exponential growth that is foreseen. To name a few, it is under study to move in the high availability direction for the different components, such as the underlying database, allowing the GitLab infrastructure to scale and to make it more robust by removing single points of failure. Furthermore, it is also under investigation to move from CephFS to a solution called Gitaly cluster, which will allow having replication of data and high availability of the repository's storage.

Moreover, to have a replicated system by using the GitLab component Geo [27], in order to follow best practices and recommendations concerning business continuity and disaster recovery methodologies, aiming at following industry standards and, by definition, keep improving user experience and performance.

An exciting future of challenges shows in the horizon.

References

[1] GitLab, Inc., “GitLab at CERN Portal,” 2023. [Online]. Available: https://gitlab.cern.ch. [Accessed 2023].

[2] Red Hat, Inc., “OpenShift Portal,” [Online]. Available: https://access.redhat.com/products/openshift/. [Accessed 2023].

[3] The Linux Foundation, “Helm Portal,” [Online]. Available: https://helm.sh/. [Accessed 2023].

[4] GitLab, Inc., “GitLab Portal,” [Online]. Available: https://about.gitlab.com/. [Accessed 2023].

[5] The PostgreSQL Global Development Group, “Main Portal,” [Online]. Available: https://www.postgresql.org/. [Accessed 2023].

[6] “Ceph Object Gateway,” [Online]. Available: https://docs.ceph.com/en/quincy/radosgw/index.html. [Accessed 2023].

[7] GitLab, Inc., “Cloud Nativer Hybrid reference architecture,” [Online]. Available: https://docs.gitlab.com/ee/administration/reference_architectures/10k_users.html#cloud-native-hybrid-reference-architecture-with-helm-charts-alternative. [Accessed 2023].

[8] Red Hat, Inc., “Magnum User Guide,” [Online]. Available: https://docs.openstack.org/magnum/latest/user/index.html. [Accessed 2023].

[9] Ceph authors and contributors, “Ceph File System,” [Online]. Available: https://docs.ceph.com/en/latest/cephfs/. [Accessed 2023].

[10] A. Sreeramaneni, B. Seo and C. KOH, "A business driven scalable cloud computing service platform (PaaSXpert)," *Journal of KIIT,* vol. 15, no. 1, pp. 35-44, 2017.

[11] R. Reagan, “Redis Cache,” *Web Applications on Azure: Developing for Global Scale,* pp. 257-300, 2018.

[12] GitLab, Inc., “Gitaly and Gitaly Cluster,” [Online]. Available: https://docs.gitlab.com/ee/administration/gitaly/. [Accessed 2023].

[13] The Linux Foundation, “Persistent Volumes,” [Online]. Available: https://kubernetes.io/docs/concepts/storage/persistent-volumes/. [Accessed 2023].

[14] The Linux Foundation, “Networking - Endpoints,” [Online]. Available: https://kubernetes.io/docs/concepts/services-networking/service/#endpoints. [Accessed 2023].

[15] The Linux Foundation, “Networking - NodePort,” [Online]. Available: https://kubernetes.io/docs/concepts/services-networking/service/#type-nodeport. [Accessed 2023].

[16] GitLab, Inc., “GitLab Workhorse,” [Online]. Available: https://docs.gitlab.com/ee/development/workhorse/. [Accessed 2023].

[17] GitLab, Inc., “Gitaly - Network architecture,” [Online]. Available: https://docs.gitlab.com/ee/administration/gitaly/configure_gitaly.html#network-architecture. [Accessed 2023].

[18] GitLab, Inc., “GitLab Architecture - Simplified component overview,” [Online]. Available: https://docs.gitlab.com/ee/development/architecture.html#simplified-component-overview. [Accessed 2023].

[19] MIT, “MIT Kerberos Documentation,” [Online]. Available: https://web.mit.edu/kerberos/krb5-devel/doc/basic/keytab_def.html. [Accessed 2023].

[20] GitLab, Inc., “Moving repositories managed by GitLab,” [Online]. Available: https://docs.gitlab.com/ee/administration/operations/moving_repositories.html. [Accessed 2023].

[21] X. Li, “kclient: BUG: kernel NULL pointer dereference, address: 0000000000000008,” 14 04 2022. [Online]. Available: https://tracker.ceph.com/issues/55327. [Accessed 10 2022].

[22] Red Hat, Inc., "Magnum Nodegroups," [Online]. Available: https://specs.openstack.org/openstack/magnum-specs/specs/stein/magnum-nodegroups.html. [Accessed 2022].

[23] Elasticsearch B.V., "What is the ELK Stack?," [Online]. Available: https://www.elastic.co/what-is/elk-stack. [Accessed 2023].

[24] A. Freund, "Improving Postgres Connection Scalability: Snapshots," 25 10 2020. [Online]. Available: https://techcommunity.microsoft.com/t5/azure-database-for-postgresql/improving-postgres-connection-scalability-snapshots/ba-p/1806462. [Accessed 2023].

[25] C. Chauhan and D. Kumar, PostgreSQL High Performance Cookbook, Packt, 2017.

[26] I. P. Trobo and M. D. Giorgi, "Add PgBouncer Helm Chart," CERN, 13 02 2023. [Online]. Available: https://gitlab.com/gitlab-org/charts/gitlab/-/merge_requests/2973. [Accessed 2023].

[27] GitLab, Inc., "GitLab Geo," [Online]. Available: https://about.gitlab.com/solutions/geo/. [Accessed 2023].