

CHAPTER

4

AN EXPERIENCE OF SYSTEM DEVELOPERS USING GITLAB

*Lina Fatini Azmi, Wan Lokman Wan Beduridin,
Farizie Abdul Razak, and Nazmona Mat Ali*

4.1 INTRODUCTION

This chapter aims to describe the best practise of GitLab implementation in a research university, in the context of the experiences of the developer team. The contribution of this chapter is to provide lessons learned to others about GitLab implementation in the system development team.

The development of innovative electronic projects for research universities is known for its complexity and requires a large number of resources. They are complex because of the combination of innovation, information, and communication technologies, politics, and social impacts (Wen et al., 2020). Handling such collaborative initiatives can be difficult. Poor project management is one of the most common reasons for project failure, owing to issues such as a lack of creativity and vision, poor communication and organisational skills, a hazy work breakdown, poor scope definition, ineffective workload management, and change management, as well as shifting requirements, technical complexity and poor delegation and tracking (Wen et al., 2020). Other particular issues emerge, such as coordinating the project's teamwork, identifying goals, and synchronising the pace of work among developers (Goldfinch, 2007). As a

result, proper management of the project should be a key focus in order to develop effective project solutions.

The significant success of the GitLab project in delivering complicated and high-quality products demonstrates the benefit of incorporating its principles and methods into collaborative situations (Wen et al., 2020). The benefits of this strategy are reduced software and business development expenses, faster time-to-market, improved performance, and a focus also economic profit (Kourtesis et al., 2012). Complex and large-scale projects in research universities cause delays or even failures in the delivery of digital services. One way to tackle these challenges is to use agile methodologies in system development to update large-scale existing infrastructure and adjust to environmental changes and stakeholder requirements more quickly (Mergel, 2016).

The adoption of agile practises in a research university is an important topic to cover in this chapter. For a successful GitLab implementation, the authors will highlight managerial, organisational, process, and technology concerns that must be rethought and reconfigured in a research university. According to the researchers, agile methodologies are better suited to projects with a wide range of requirements, technical capabilities, and technologies. In terms of development activities, agile practises place a premium on increased communication and collaboration (Bheri & Vummenthala, 2019). The researchers emphasised the importance of GitLab values and practises in addition to the well-known agile methodologies.

Moreover, this chapter suggests approaches to solve issues that occurred in the system development section of a research university. The researchers also propose a framework of process flow to implement GitLab to manage our teams and resources. To do so, the researchers need to gather the ideas and experiences of system development teams in a research university. Based on those experiences, it helps the researchers to create the base of the framework and support all the processes in the framework.

This chapter is organised by firstly highlighting the purpose of this paper and describing the problems in system development teams in Section 4.1. Then, the next section describes the introduction of GitLab

and presents related work in Section 4.2. Section 4.3 describes the GitLab framework in a research university, while Section 4.4 explains the issues and challenges faced by developers in a research university. Section 4.5 discusses the results and advantages of implementing GitLab as a lifecycle tool, and finally, Section 4.6 of this chapter provides the conclusion for this chapter.

4.2 REVIEW OF GITLAB IMPLEMENTATION IN SYSTEM DEVELOPMENT LIFECYCLE

A comprehensive review of GitLab and the implementation approaches by other organisations can be found in this section. The brief history of GitLab and description of its features are also included in this section.

4.2.1 DevOps Framework for Agile Development

DevOps is a diverse and collaborative endeavor within a company that automates the continuous delivery (CD) of new software versions while ensuring their correctness and performance (Leite et al., 2019). DevOps culture is a method of facilitating communication amongst IT teams while also speeding up and improving the maintenance cycle. DevOps encourages rapid and continuous feedback from operations to development in order to identify problems before they may affect users (Lopez-Pena et al., 2020).

Moreover, DevOps is an advancement of the Agile movement, and it combines Agile methods with an automated toolchain infrastructure to allow continuous integration (CI), testing, quality assurance, and deployment. Small release iterations with user feedback are encouraged in agile system development. It is expected that the team will be able to release the system regularly in a production-like environment (Leite et al., 2019). Therefore, in this chapter, an explanation of a DevOps infrastructure for agile development is provided based on the GitLab code repository.

4.2.2 History of GitLab

GitLab originated as a web-based open-source Git repository manager and rapidly evolved into a single tool for the whole DevOps life cycle (Vesselov & Davis, 2019). GitLab is also recognised as a source code management solution for storing and controlling the source code of projects. It is an open-source platform that aims to provide a comprehensive solution that helps developers get from concept to production on a single platform.

It was initially made public on GitHub in October 2011 and has since grown into a powerful tool. When redistributing the GitLab code, the author's identity must be included. Dmitriy Zaporozhets, the project's primary developer, along with Valery Sizov was originally wrote GitLab in Ruby and some parts in Go (Than & Phyu, 2019).

Users who contribute work to a project (editing code, creating chapters, etc.) will have a copy of the project downloaded, checked out, or cloned on their local computer because GitLab is built on top of git. It offers a web interface for many of git's more advanced tasks, as well as a workflow for working with git that maximises productivity, efficiency, and ease of use.

4.2.3 GitLab Features

Built-in continuous integration and continuous deployment, configurable permissions, tracking of activity in progress work, moving issues between projects, group-level milestones, building new branches from issues, dashboard, and time tracking are just a few of GitLab's unique features (Wen et al., 2020). From a single application interface, it tracks changes and performs continuous integration and delivery of products.

Other features include deploy keys that allow a remote machine to access a GitLab repository by importing SSH public key, deploy tokens that allow to download or push and pull packages and container registry images from a project without a user or a password, branches as a version of a project's working tree and create permission to the branches to protect it, pipeline that comprise of jobs and stages.

4.2.3.1 Repositories

A repository is the place where codes are stored and where changes are made to it. Every change is tracked with version control. In order to create a repository, we can either create a new project or fork an existing project. A fork is a personal copy of the repository and all its branches. The place to store and share bits of code with other users is called snippets. Then, to make changes and keep it parallel without affecting other changes, we can create branches. To impose further restrictions on certain branches, they can be protected by write permission to the branches, while the default branch is protected by default. We can also make changes tracking by file blame and checking commit history using file history. A repository can be moved to or from an external source by mirroring. After changing the source code into a branch, we can request merging, and we can visualise and collaborate on the code changes before merging whenever we open a merge request.

4.2.3.2 Continuous Integration or Continuous Deployment

Using Auto DevOps, GitLab CI/CD can automatically develop, test, deploy, and monitor our application. GitLab Auto DevOps is a set of pre-configured tools and integrations that work together to facilitate system delivery processes. Interestingly, GitLab CI/CD can be used to discover bugs and errors early in the development process, guaranteeing that all the code deployed to production adheres to the defined code standards. There are a number of GitLab CI/CD concepts. One of the concepts is called pipeline refers to the top-level component of continuous integration, delivery, and deployment that structures the CI/CD processes. Pipeline configuration begins with jobs that are defined with constraints stating under what conditions they should be executed. Then, CI/CD variables are a type of environment variable that control the behaviour of jobs and pipelines, and also store values that we want to reuse. Besides, there is another concept in CI/CD workflow, which is called Docker container, that allows jobs to run in it.

4.2.3.3 Runner

GitLab Runner is a pipeline-running program that interacts with CI/CD. GitLab Runner is also capable of running inside a Docker container. Many features of GitLab Runner include the ability to run multiple jobs at the same time, use multiple tokens with multiple servers, limit the number of concurrent jobs per token, customise the job running environment, automatic configuration reload without restart, and enable Docker container caching. Based on their accessibility, there are three types of runners: (1) Shared runners for all projects, (2) Group runners for all projects and subgroups in a group, and (3) Specific runners for individual projects. GitLab Runner includes a variety of executors that may be used to run builds in a variety of scenarios. The environment in which each job runs is determined by an executor. An executor also supports a variety of platforms and processes for project development.

4.2.4 GitLab Implementation in Other Organisations

Many organisations use the GitLab application to develop and manage their system development projects. Since GitLab has many features, each organisation implements GitLab with a different approach based on its needs. Examples of GitLab practises are shown as follows:

- (1) A study by Wang et al. (2019) used Gitlab to provide developers with searching and profiling based on code quantity, code quality, contribution, and score of projects to measure the developers' expertise.
- (2) Bai et al. (2019) applied GitLab in an educational context to help students gain the skills needed for working in industry. It also allowed educators to collect data for identifying problematic student behaviour, and at once to improve the learning outcomes of the students.
- (3) Wen et al. (2020) studied the case of the Brazilian Public Software portal project, a government-academia collaboration that uses FLOSS practises and agile methods for project management. In this study, Gitlab was used to overcome collaboration barriers faced by stakeholders involved.

- (4) A study by Yang et al. (2020) developed a novel education information system at ECNU based on DevOps and related techniques, especially in four aspects: Continuous Integration (CI), Continuous Deployment (CD), log management, and code quality.

4.3 A PROPOSED GITLAB PRACTISE FRAMEWORK IN A RESEARCH UNIVERSITY

A framework of Gitlab practises is proposed to give an idea and guidelines for those who are interested in implementing this application in their system development projects. Nevertheless, this proposed framework is applicable for a research university setting in the meantime.

4.3.1 Case Study Overview

The Deputy Vice Chancellor's Department (Research and Innovation), or known as the DVCRI Department, aims to manage innovation research activities for Universiti Teknologi Malaysia (UTM). In addition, the main services related to technology and information systems (TIS) are developed and maintained by the Research Computing Section, or known as UTMRC.

UTMRC's main portfolio is to provide TIS services to UTM researchers, for example, research-related system development, innovation, commercialisation, community, and networking. Among the systems that have been successfully developed by UTMRC are Research and Development Information System (RADIS 4.0) for research lifecycle and grant management, Innovation and Commercialisation System (INNOCOMMS 2.0) for intellectual property and commercialisation tracking, UTMLAB Phase 1 for managing laboratory equipment, and e-Press for digital scholarly publishing. Additionally, the section maintains e-Activity to monitor staff involvement in academic tasks, UTMScholars and UTMEexperts to showcase researcher profiles and expertise, Books Online as a digital repository for academic publications, and the TNCPI Official Web Portal as the department's primary information hub. Beyond

system development, UTMRC manages essential data for national frameworks such as Malaysia Research Assessment (MyRA), Rating System for Malaysian Higher Education (SETARA), and Ministry of Education Data Collection System (MyMoheS).

UTMRC implements GitLab in its system development life cycle and aims to create an environment that can easily manage all source code from all projects. Moreover, we believe collaboration between team members is a key to the success of system development, and this can be strengthened with GITLAB implementation. Heretofore, GitLab provides services that suit our needs in many aspects in a research university situation.

4.3.2 GitLab Practise Framework

The basic structure of our GitLab practise framework has three layers. The first one is Branch A for programmers to push their code to the repository, the second one is Branch B for development, and the last one is Branch C, which is for the admin to deploy code to production.

Each of these layers has its own access level to Branch A whereby which is a platform where programmers assemble and get the latest source code. All of their source code that has already been completed will be stored here and shared with other programmers. Concurrently, Branch B is for the admin to pull and deploy source code to the development server. It is essential because at this point, the system will be tested before it can be released to production. Then, system analysts will check and test the system. If there are any issues occur, the system analysts will inform programmers involved to make corrections and modifications until the system is fully tested without any errors or faults. The final layer is Branch C, where admins are allowed to deploy source code to production and further release the system to the end users. Figure 4.1 shows how the proposed GitLab Practise framework works, involving three layers of Branch A, B, and C.



Figure 4.1

4.3.3 Issues and Challenges

There are some issues and challenges that occurred, particularly at the beginning of GitLab implementation. This is mainly because we were not familiar with GitLab, even had no experience using it in system development. The following are the issues and challenges that should be considered while implementing GitLab.

(1) Merging and Editing the Conflict File

Programmers commit and push code without pulling the latest code from the git repository. This will affect the latest code push to the git repository, i.e., Branch B, and consequently will affect other programmers when they pull the latest code from the git repository. Therefore, admins should use a rollback function to check out a git repository to the desired checkpoint ID.

(2) Multiple Admin

Normally, only one admin has full access to manage a git repository in GitLab. Problems occur when there are many projects that need to be managed simultaneously and in parallel. This will cause delays in system development and confusion in communication between members. Hence, it is important to create multiple admins for more effective and efficient project management.

(3) Multiple Roles

In some cases, the owner of the system is also the developer of the system. Therefore, there are several roles, and each role has a different access level and function in our proposed framework. In other words, users with multiple roles are allowed in our proposed GitLab Practise framework.

(4) Protected Branch

Only users with the role of owner and maintainer of the assigned branch can commit and push the code. Thus, admins will assign the desired users to the selected branch to prevent users from committing and pushing to the wrong branch.

(5) **GitLab Runner**

When developing a lot of systems at the same time and in a short period of time, runners are really needed to fasten the process without knowing how it works from the backend. A runner is an agent that runs the CI/CD jobs that come from GitLab. This is where communication takes place between our Gitlab instance and the machine to speed up the operation process for all existing projects towards micro services.

(6) **Docker Container**

Since we use an agile approach in our projects of system development projects, all possibilities and uncertainties are taken into account. For example, the possibility of downtime taking too long to recover, and deployment taking too much time to apply due to unexpected things. Therefore, we use Docker containers as it is an easy, fast, and minimal step of configuration in development, staging, and production servers, i.e., one-time configuration only.

(7) **Authentication Key**

When involving a lot of system development simultaneously, there is a possibility of slowing down in the authentication process. Hence, we identify runners for each server. Each runner has their own key to run the process includes the process of encrypting and decrypting for security reasons. When we use keys to authenticate to the GitLab remote server, we do not need to supply username and password each time.

4.3.4 Evaluation and Findings

Using GitLab has improved many aspects in managing source code for UTMRC, particularly in situations where many systems are developed simultaneously in a given limited time. Source code can be easily accessed, managed, and securely stored in the git repository. Moreover, we applied features in GitLab such as CI/CD/Pipelines and DevOps for fast deployment to the development server, staging server, and production server. The CI/CD/Pipelines are one of the parts in DevOps,

while DevOps is one of the elements in the Agile approach that we currently practise in many systems development projects.

In addition, the git repository offers insights into the data it represents in the form of a dashboard that can help us make some relevant analysis towards University Data Driven. We believe the use of GitLab can benefit the University as a whole, particularly to maintain its status as a Research University (RU) and sustain its performance as well as ranking year by year.

4.4 CONCLUSION

This chapter shares issues and solutions that occurred throughout the process of system development in the research university that can be a guideline for those who are interested and in the same setting. We use GitLab and are further proposing a framework of process flow to implement GitLab in order to manage our teams and resources. This proposed framework is an illustration of GitLab practises that have been validated during the system development. Importantly, this effort is well documented as lessons learned and best practises that could be considered as part of knowledge management.

REFERENCES

- Bai, X., Pei, D., Li, M., & Li, S. (2019). *The DevOps lab platform for managing diversified projects in educating agile software engineering* [Conference session]. 2018 IEEE Frontiers in Education Conference (FIE) (pp. 1–9). IEEE. <https://doi.org/10.1109/FIE.2018.8658955>
- Bheri, S., & Vummenthala, S. (2019). *An introduction to the DevOps tool related challenges* [Master's thesis, Blekinge Institute of Technology]. DiVA Portal.
- Goldfinch, S. (2007). Pessimism, computer failure, and information systems development in the public sector. *Public Administration Review*, 67(5), 917–929. <https://doi.org/10.1111/j.1540-6210.2007.00778.x>

- Kourtesis, D., Bratanis, K., Bibikas, D., & Paraskakis, I. (2012). Software co-development in the era of cloud application platforms and ecosystems: The case of CAST. *IFIP Advances in Information and Communication Technology*. (Vol. 380, pp. 196–204). Springer.
- Leite, L., Rocha, C., Kon, F., Milojevic, D., & Meirelles, P. (2019). A survey of DevOps concepts and challenges. *ACM Computing Surveys*, 52(6).
- Lopez-Pena, M. A., Diaz, J., Perez, J. E. & Humanes, H. (2020) DevOps for IoT systems: Fast and continuous monitoring feedback of system availability. *IEEE Internet of Things Journal*. 7(10), 10695–10707.
- Mergel, I. (2016). Agile innovation management in government: A research agenda. *Government Information Quarterly*. 33(3), 516–523.
- Than, P. P., & Phyu, M. P. (2019). *Continuous integration for Laravel applications with GitLab* [Conference session]. Proceedings of the 1st International Conference on Advanced Information Science and System. <https://doi.org/10.1145/3373477.3373479>
- Vesselov, S., & Davis, T. (2019). *Building design systems: Unify user experiences through a shared design language*. Apress. <https://doi.org/10.1007/978-1-4842-4514-9>.
- Wang, J., Meng, X., Wang, H., & Sun, H. (2019). An online developer profiling tool based on analysis of GitLab repositories. *Communications in Computer and Information Science*, 1128, 331–342. https://doi.org/10.1007/978-981-15-1377-0_32
- Wen, M., Siqueira, R., Lago, N., Camarinha, D., Terceiro, A., Kon, F., & Meirelles, P. (2020). Leading successful government-academia collaborations using FLOSS and agile values. *Journal of Systems and Software*, 164, 110548.
- Yang, Dawei, Wang, D., Yang, Dongming, Dong, Q., Wang, Y., Zhou, H., & Daocheng, H. (2020). DevOps in practise for education management information system at ECNU. *Procedia Computer Science*, 176, 1382–1391.

