Simple Web Application Workflow on Gitlab

Contents

INTRODUCTION

In the modern software applications world, projects are getting wider and the traffic scale is hitting the sky, and hence, modern planning and implementation methodologies are needed to take place. To build and publish such a modern software product, the whole workflow of the development process must be well planned, implemented, and monitored.

In this document, the development workflow of *an example web application* will be demonstrated from the application's first task to the production deployment.

APPLICATION

The software application demonstrated in this document is assumed to be a simple web page that does not do anything except showing a small rectangle holding a sample video. Now for the sake of organizing the whole process of building and publishing the application, we will be implementing the project workflow on the well-known platform *GitLab*.

Why GitLab

GitLab is a Git-based repository manager and a powerful complete application for software development. With GitLab, we can get a complete CI/CD toolchain in a single platform for our project. This includes having one interface, one conversation, and one permission model. By planning, building, working on, testing, and deploying our project on GitLab, we will not need any other software or platforms, even though GitLab offers a large area of integration with other software.

NOTE: This document will not dive into the very technical details, instead, will mention GitLabs features and services needed for each step.

First Step

Before anything, the first thing to do with the project after completing its idea is creating it, which will be the first step in the project workflow on Gitlab.

Groups

GitLab offers a service of creating and managing projects, and this is done by either create an individual project or a group of projects. A group here is a collection of several projects. When the projects are well organized, the group works like a folder. Here we can manage our group member's permissions and access to each project in the group.

Projects

There are four ways to create a new project in GitLab, and those are:

- A blank project, this is the normal way of creating a project from scratch
- From a template, a pre-populated project with the necessary files to get started
- Import project by migrating data from an external source like GitHub or Bitbucket
- Connect external repository to GitLab CI/CD

In our case, the second point will be great, because we can create a project with the needed files for the needed technology (Spring, .Net...etc.)

Flow Diagram

The following Flow diagram shows the project development workflow stages explained in the next sections.

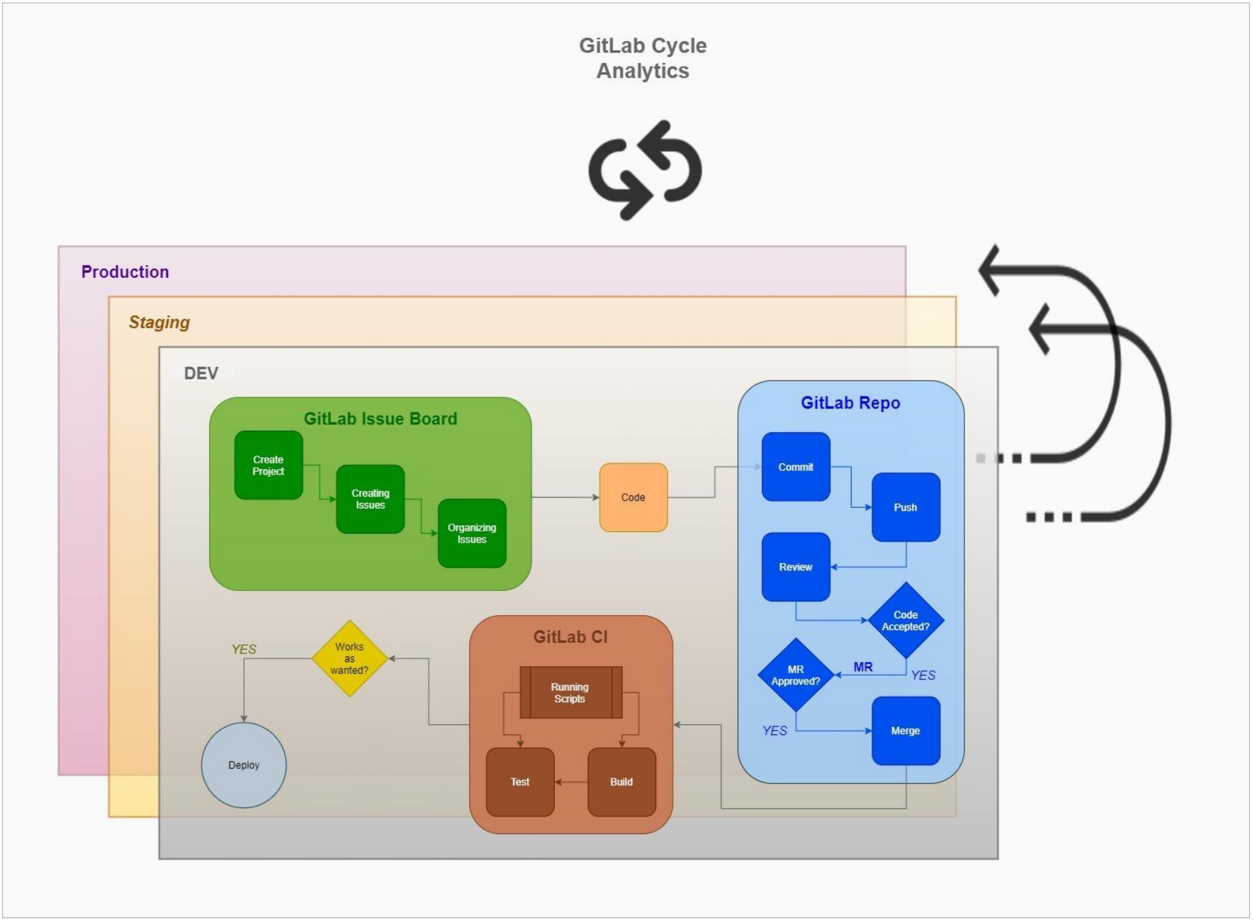

Figure 1: Web application development workflow on GitLab.

TRACK ISSUES

As the work must be divided and subdivided into individual workpieces to start implementing the project, this step needs to be well implemented. GitLab says "*The most effective way to discuss an idea is creating an issue for it*". An issue here can be a task, a new story, a group of features, or a defect or bug. We can move to implement our project workpieces into issues using *GitLab Issue Tracker*.

Issue Tracker

Each project hosted by GitLab has an issue tracker. There is an easy interface to create our first issue, which can be described using Markdown and there will be pro tips to enhance the issue description. The

GitLab Issue Tracker presents extra functionalities to make it easier to organize and prioritize project actions, including *Confidential Issues*, *Due Dates*, *Assignee*, *Labels*, *Issue Weight*, and so on.

PLAN WORK

After the project idea and first workpieces where set, it is time to organize the work. Every bundle of features or resolved bugs due to a certain period of time makes what is known as *Sprint*, and each sprint begins with a planning meeting. In sprint-planning meetings, sprint issues (tickets or cards) get rechecked, prioritized, and assigned to the right resource or team.

Issue Board

The GitLab Issue Board is a tool ideal for planning and organizing project issues according to the project's workflow. It consists of a board with lists corresponding to its respective labels. Each list contains their corresponding labeled issues, displayed as cards (or tickets). In our project, we are going to have the following card lists:

- **TODO**: a list where cards to be done are held
- **IN PROGRESS**: a list where cards currently being worked on are held
- **ON HOLD**: a list where cards that need more information or decided to be paused are held
- **IN QA**: a list where cards are done working on and needs to be reviewed by the QA team are held
- **DONE**: a list where cards are done working on and confirmed to be working as expected by the QA team are held

CODE, COMMIT, AND REVIEW

After the work has been separated into many cards, the cards were reviewed and prioritized, they were distributed to the card lists, and the whole work was divided into most probably several sprints, the next step is to get their work done.

Code

Coding is where all of the business is implemented. In our project, Spring Boot and Angular will be used as the main technologies to do the coding.

Commit

As mentioned before, GitLab is a Git-based repository manager, which will be used to organize every code piece and code version of the project. At first, we write the project’s code locally and, once we are done with the first iteration, which is assumed to be done here, we commit the code and push to the project’s GitLab repository.

Review

Now once the commit and push are done to our code changes to some work branch, GitLab will identify this change and will prompt us to create a *Merge Request* (MR).

Merge Request (MR)

A Merge Request (MR) is a request to merge one branch into another. We will use our created MR to visualize and collaborate on proposed changes to source code.

WIP MR

A WIP MR, which stands for Work in Progress Merge Request, is a technique used at GitLab to prevent ME from getting merged before it's ready. By just adding "*WIP:*" to the beginning of the title of an MR, the MR will not be merged unless it is removed from there.

Approval

Once we have created a merge request, it is s time to get feedback from the team or collaborators. With GitLab, using the diffs available on the UI, we can add inline comments, reply to them and resolve them. One more thing, the commit history is available from the UI, the changes between the different versions of a certain file can be tracked and viewed flexibly. Finally, if the changes work as expected, they will be approved and merged with the source code, and the issue they were made for will be closed.

GitLab Flow

The following figure shows how our Git-based management strategy can be improved with the GitLab Flow. Notice that the last two nodes are not important in this context, the path we are interested in here is the path from the issue to the code merge.

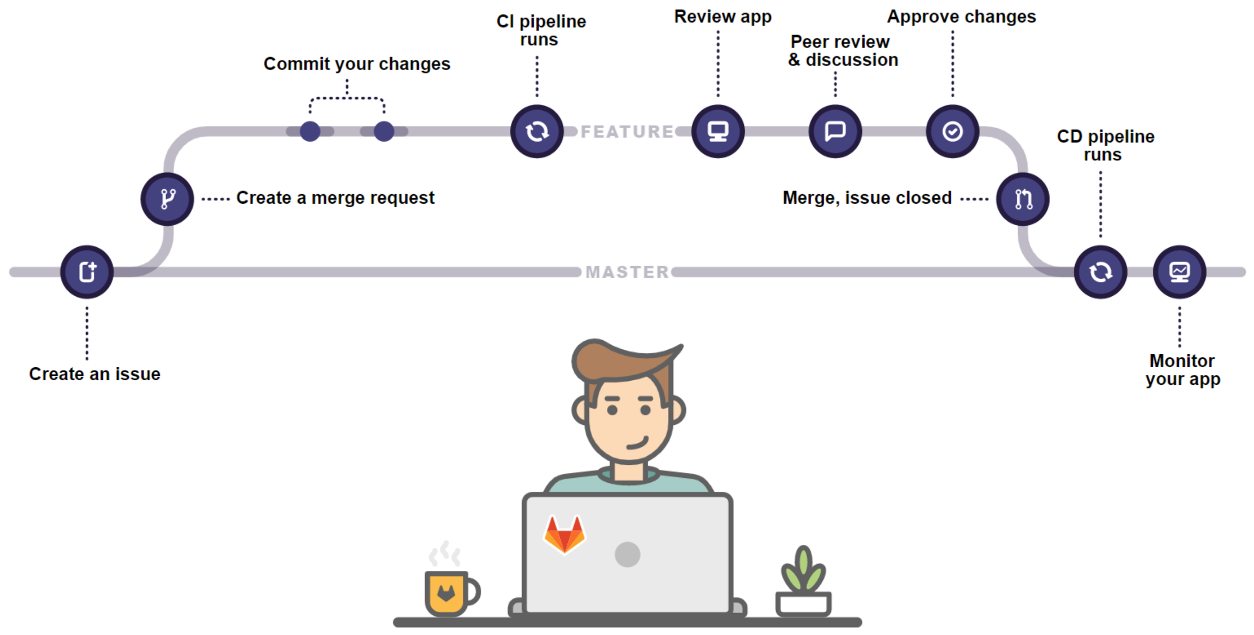

Figure 2: GitLab Flow, the path of a code change from the issue to the merge.

BUILD AND TEST

After the merge has been done, the code needs to be built. A *build* is the process of converting source code files into standalone software artifact(s) that can be run on a computer, or the result of doing so. For the sake of both build and test, we are going to use GitLab CI here.

GitLab CI

GitLab CI is a powerful built-in tool for *Continuous Integration*, *Continuous Deployment*, and *Continuous Delivery*, which can be used to run scripts as wished. The possibilities are endless, we think of it as if it was our own command-line running the jobs for this project, and It's all set by one Yaml file called, "*.gitlab-ci.yml". All jobs of building, testing, and deploying of this project will be scripts run using GitLab CI.

Test

After the code is finally built and is ready to be tested, several tests will be applied to the environment that had the changes built on.

Manual vs Automated testing

Many tests could be applied to the built environment, and those tests can be either manual or automated. Manual testing is done in person, by clicking through the application or interacting with the software and APIs with the appropriate tooling. Automated tests, on the other hand, are performed by a specialized software that executes a test script that has been written in advance.

In this project, more than one test will be applied, among them; Smoke, Integration, and performance tests.

DEPLOY

After the tests were applied and accepted, it is time do deploy changes on the needed *environments* (envs) or server tiers, and in this project, they will be:

- Development (DEV)
- Staging
- Production (Prod)

Deploying into different environments ensures that many stages of code verification and validation will be made to make sure that the whole application will work as expected when it goes live.

DEV

In this environment, developers test code and check whether the application runs successfully with that code. All development coding and testing are done in this environment, which is the first environment the code exists on after the developer's local machine.

Staging

This environment is made to look exactly like the production server environment. The application is tested on the staging server to check for reliability and to make sure it does not fail on the actual production server. The testing on the staging server is the final step before the application could be deployed on a production server.

Prod

The production environment is where software and other products are put into operation for their intended uses by end-users.

As mentioned above, GitLab CI will be used as the Command-line tool to deploy the code into all of the three environments.

FEEDBACK

Now after we followed the GitLab Workflow, we can gather feedback with *GitLab Cycle Analytics* on the time the project team took to go from the first issue to production, for each key stage of the process:

- **Issue**: the time from creating an issue to assigning the issue to a milestone or adding the issue to a list on the Issue Board
- **Plan**: the time from giving an issue a milestone or adding it to an Issue Board list, to pushing the first commit
- **Code**: the time from the first commit to creating the merge request
- **Test**: the time CI takes to run the entire pipeline for the related merge request
- **Review**: the time from creating the merge request to merging it
- **Staging**: the time from merging until deploy to production
- **Production**: The time it takes between creating an issue and deploying the code to production

SUMMARY

In this simple video-streaming web application, the whole development workflow with GitLab was demonstrated. With GitLab, we could create the project from a pre-built template on its Git-based repository. After that, we used GitLab Issue Tracker to express the project's ideas as issues, which are workpieces or items. After the first project issues were set, we could organize and prioritize them with GitLab Issue Dashboard, which allowed us to set each issue's priority, due date, and the assignee.

The next thing after the project was set was the coding. We used Spring Boot and Angular to build the application's API and GUI. Now after coding was done and reviewed, we committed and pushed our first version of code to the GitLab repository, which generated an MR. The MR was approved and the code changes were merged into the project's source code. What we did next is using GitLab CI to run our scripts to build, test, and deploy the project on the DEV environment, which was the first environment after local machines.

Deploying the code on staging, is next to deploying it on DEV. Staging was the code simulation on the real environment so that everything was confirmed to be working as wanted. The last thing before starting to get feedback is deploying on Prod, which is the real environment the application was set on.

Finally, getting feedback is the method followed to check what needs to be enhanced or improved. With GitLab Cycle Analytics, we were able to monitor and audit our application behavior on different stages of the whole process workflow from issues to Prod.