УДК 004.04’416

Мартынов Т. Е.

БЫСТРАЯ И ЕСТЕСТВЕННАЯ НЕПРЕРЫВНАЯ ИНТЕГРАЦИЯ С GITLAB CI

Тимофей Евгеньевич Мартынов

студент магистратуры

timtim96@bk.ru

ФГАОУ ВО «Российский государственный профессионально-педагогический университет», Россия, Екатеринбург

FAST AND NATURAL CONTINUOUS INTEGRATION WITH GITLAB CI

Timofej Evgenyevich Martynov

Russian State Vocation Pedagogical University, Russia, Yekaterinburg

Аннотация*. В статье представлен обзор непрерывной интеграции и доставки GitLab, а также основные функции инструмента.*

Abstract*. This article provides an overview of the continuous integration and delivery of GitLab, as well as the main features of the tool.*

Ключевые слова: *непрерывная интеграция, непрерывная доставка, gitlab.*

Keywords: *continuous integration, continuous delivery, deployment, gitlab.*

Введение

Системы контроля версий (их ещё называют системами управления версиями) — один из инструментов, который использует в своей работе любой программист от первокурсника до опытного разработчика с сотнями успешных проектов:

- ссылки на примеры кода в репозиториях помещают в своё портфолио;

- большие проекты с открытым программным кодом получают тысячи улучшений, благодаря размещению на специализированном хостинге;
- от 30 до 70% кода, использованного в программном продукте, профессиональные разработчики могут скопировать с проектов, представленных в открытых репозиториях.

Системы контроля версий незаменимы в командной разработке, где синхронизация процесса играет важную роль.

GitLab — не только git репозиторий, он обладает богатым функционалом, с которым знаком далеко не каждый его пользователь.

Эта статья предлагает обзор того, как разработчики GitLab реализовали основные концепции непрерывной интеграции/доставки и представили основные функциональные инструменты.

Благодаря бурному развитию функционала GitLab используется многими компаниями. Часть этой популярности исходит от того, что это проект с открытым исходным кодом, и его цикл выпуска обновлений очень короткий — он получает множество новых функций каждый месяц на 22-й день.

Рисунок 1 — Жизненный цикл приложения

Иногда случается, что люди просто придерживаются известных для них инструментов и не замечают или не исследуем альтернативы, которые могут улучшить их повседневную работу. Непрерывная интеграция относится к некоторым из менее известных функций GitLab. Программное обеспечение GitLab намного больше, чем традиционный сервер системы контроля версий. В прошлом году они придумали свой так называемый «мастер-план» расширить продукт, чтобы он охватывал каждый этап цикла разработки, или, по их словам, переход от идеи к производству. И в предыдущих и будущих выпусках они объединяют некоторые действительно полезные дополнительные интеграции, как, например, Mattermost ChatOps или Prometheus monitoring.

Различия с другими инструментами непрерывной интеграции

Рассмотрим полную картину функций, которые делают его настолько мощным. Прежде всего, его модель основана на облегченном файле конфигурации YAML, хранящемся в корне каждого репозитория. Это имеет некоторые плюсы:

1. Тесто сопряженные системы (и непрерывная интеграция и система контроля версий — единый продукт).
2. Конфигурация непрерывной интеграции становится версионной.
3. Применение различных ветвей с различными конфигурациями.
4. Возможность участникам также сотрудничать в настройке интеграции.
5. Интеграция Docker из коробки, включая частный реестр Docker для каждого проекта.
6. Браузер артефактов, который позволяет получить доступ к этапам, выводится так же, как и локальный.
7. Отсутствие времени, потраченного на обслуживание кошмара сервера CI.

Конечно, с точки зрения пользователя, он также имеет некоторые недостатки по сравнению с другими системами непрерывной интеграции. Отсутствие плагинов и интеграций, к которые все привыкли в других инструментах, является одним из них (например, создание заданий, требующих нескольких репозиториев, становится нетривиальным). С другой стороны, большинство вариантов использования достаточно охвачены, и есть такие функции, как настраиваемые оповещения по электронной почте, браузер истории или программируемые сборки.

Однако, они не изобрели колесо и не единственные, кто им пользуется. Многие другие программные решения непрерывной интеграции также лежат в аналогичных парадигмах (Трэвис, Дженкинс...). Но в отличии от них, GitLab

легче настроить, и он позволяет контролировать большинство деталей на одной вкладке браузера. Разработчики взяли некоторые из лучших функций каждого инструмента и включили их в свое решение.

Основные возможности

Далее будет краткая иллюстрация основных функции этого инструмента и демонстрация, как легко просто начать работу с ним даже без опыта системного администратора или DevOps. Все крутится вокруг *.gitlab_ci.yml* файла, содержащего определение различных этапов (шагов), которые должны быть завершены в целом, для того, чтобы проект был успешно доставлен. Файловая структура естественна для чтения, и после изучения нескольких примеров, можно писать свои собственные без особых усилий.

Сначала необходимо изучить документацию по GitLab CI, где найдется много информации о том, как воплотить конкретные потребности проекта. Но для краткого обзора возможностей, непрерывная интеграция включена по умолчанию в каждом проекте.

Настройки CI/CD

Страница настроек дает обзор того, насколько легко настроить все, что необходимо для создания непрерывной интеграции, т. е.:

Runners

Runners позволяют забыть о настройке и управлении подчиненными машинами, которые общаются с CI сервером через SSH, как балансировать нагрузку между всеми собирающими машинами, и многие другие вещи, которые так утомительны и часто сложны. Кроме того, можно пометить runners в зависимости от их возможностей (например, docker, базы данных, и т. д.), чтобы выбрать их для конкретных заданий, когда они требуются.

Runners может реализовать множество исполнителей, т. е. способов запуска сборочных скриптов/кода в них; от самого основного исполнителя SSH, через хост контейнера и прямо до самого большого кластера kubernetes, который можно поддерживать даже через Powershell/Batch в системах Windows.

Управление секретами

В эпоху микросервисов, когда проект может быть интегрирован и взаимодействовать с десятками API, которые требуют токенов, секретов, паролей и многих других способов авторизации, способ справиться с этой сложностью элегантным образом становится приоритетом. Очень плохой пример, наблюдаемый во многих проектах, заключается в том, чтобы хранить все это в файлах конфигурации на удаленной машине или даже держать в каком-то куске кода. Проекты GitLab предоставляют простое хранилище ключей в своих настройках непрерывной интеграции, доступ к которым можно получить из скриптов интеграции, чтобы помочь участникам проекта обрабатывать и настраивать все эти секреты.

Pipelines

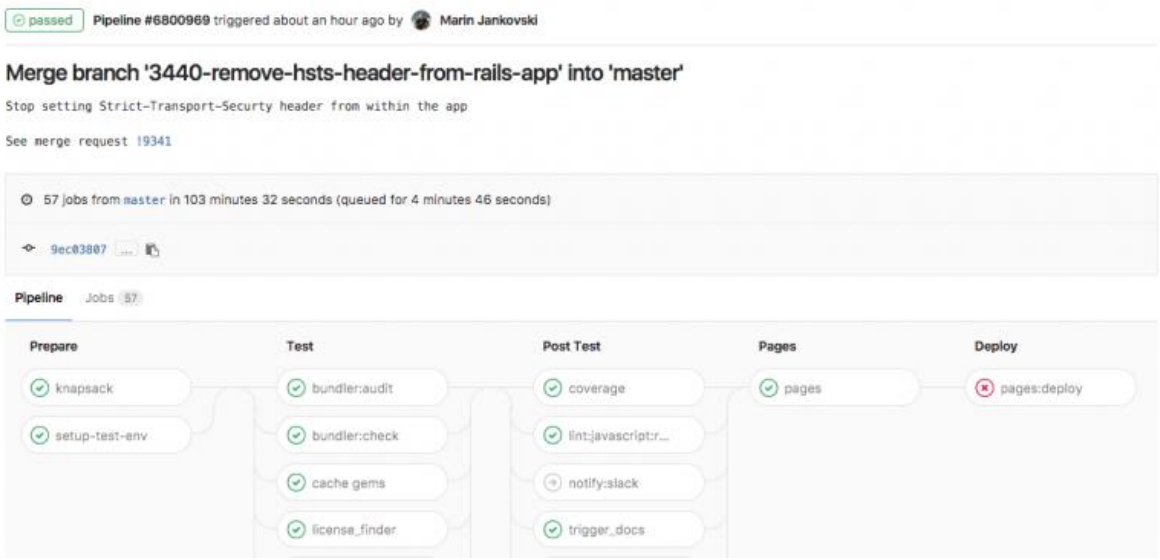

Рисунок 2 — Этапы и задания, выполняемы при отправке кода в репозиторий

Это основная функция любой системы непрерывной интеграции, и в то же время это очень простая концепция, которая представляет из себя шаги, которые будут выполняться с момента получения исходного кода, до момента развертывания приложения.

В них можно включить все, что каждый считает нужным, чтобы убедиться, что код соответствует правилам оформления (линтинг), может быть

собран, работает правильно (тестирование), интегрируется с другими системами и все, что можно придумать, чтобы предпринять последние шаги и отправить код.

Реестр контейнеров

Реестр контейнеров можно представить, как будто есть собственный частный центр Docker, где можно хранить образы проекта и обновлять их, когда это необходимо, без необходимости предоставлять их общественности и иметь возможность скачивать их из любого места в реестре. Можно иметь образ, готовый для каждого этапа сборки, и скачивать его мгновенно. Это становится очень удобно и позволяет избежать инициализации среды и, следовательно, ускорить общее время, необходимое конвейеру для запуска.

Среды и приложения для просмотра

Достаточном популярным подходом является gitflow. Для этого есть несколько веских причин. Он построен на предположении, что, если каждая ветвь разрабатывается изолированно, новые функции не будут мешать друг другу или стабильной версии, пока они не будут объединены обратно в главную ветвь. Это помогает как при разработке новых функциональных возможностей, так и при их тестировании.

Когда появились контейнеры, было очевидно, как можно облегчить и улучшить процесс развертывания различных независимых сред с индивидуальными настройками для каждой функции. Справедливо будет сказать, что теперь контейнеры используются в качестве стандарта де-факто для шаблонов среды. Большинство основных решений для непрерывной интеграции были выпущены задолго до того, как был создан первый контейнер, поэтому они не были построены с учетом Docker, но для GitLab CI все наоборот: он использует контейнеры как путь, основанный на многих преимуществах, которые он приносит в рабочий процесс, как на рисунке 3.

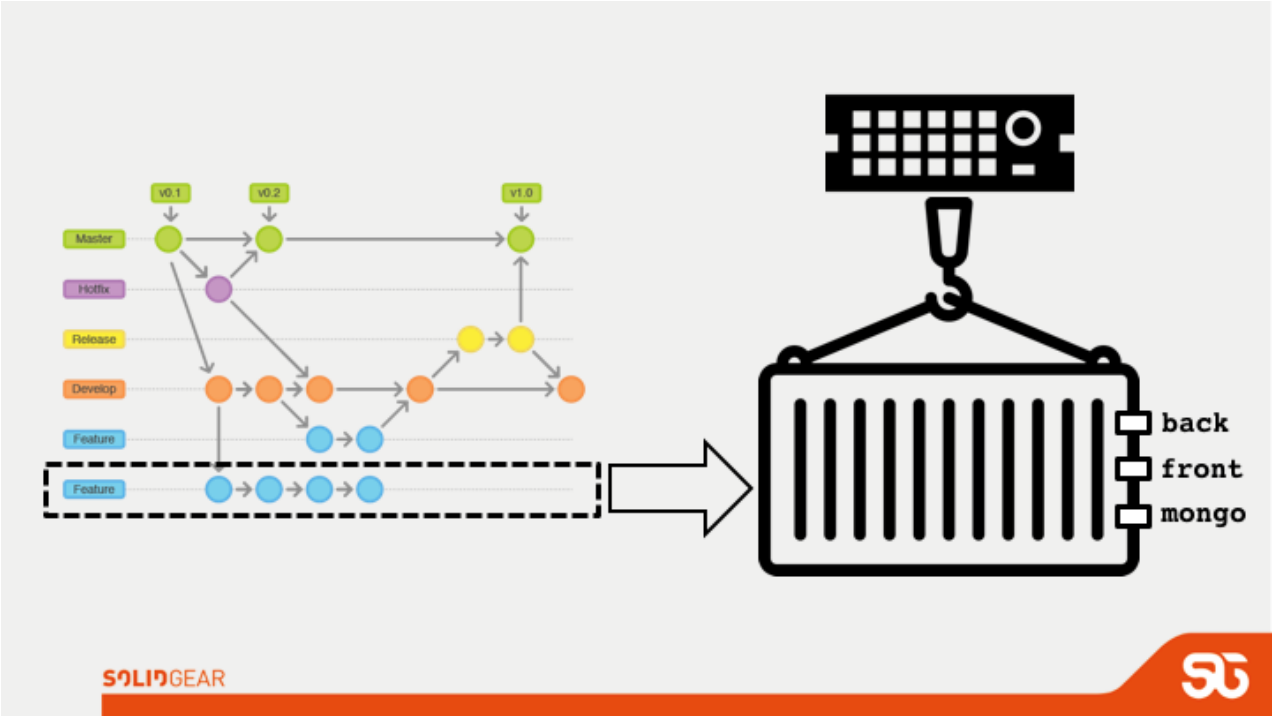

Рисунок 3 — Рабочий процесс выпуска новых версий

Очень распространенный сценарий в рабочем процессе: разработчик собирается объединить ветвь функций, которая должна быть протестирована командой QA, которая вводит некоторые новые библиотеки и новую службу (например, Redis, mongoDB...) для развертывания. Он просто обновляет Dockerfile, чтобы включить новые слои, которые управляют этими новыми зависимостями, и отправляет образ в локальный реестр. Кроме того, добавляет строку для включения новой службы через docker hub в .gitlab_ci.yml. Перспективная ветка, будучи отправленной на удаленный сервер, будет иметь все необходимое для развертывания в среде тестирования.

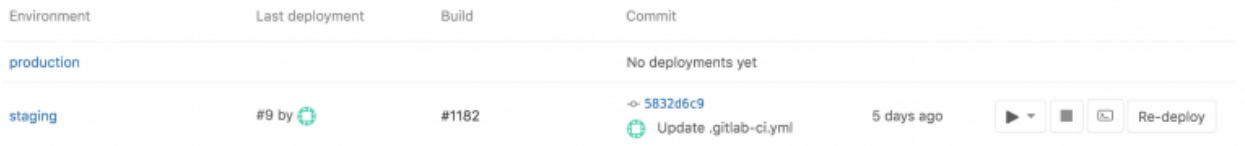

Рисунок 4 — Панель быстрых действий и результатов задания

Здесь вступает в действие приложение Review, это просто отличный способ вызова динамических сред для каждой ветви, созданных для проверки изменений и просмотра их в реальном времени. Каждая ветвь получает мгновенную поддержку развертывания при перемещении в репозиторий. Этот процесс можно ускорить с помощью таких средств, как dpl, которые абстрагируют

многие сведения для основных служб развертывания, или перейти к свободному стилю и вызвать пользовательский скрипт развертывания оттуда. Он также включает в себя интерактивный терминал в браузере для анализа сборок и устранения неполадок, если это необходимо.

Еще одна важная функция — это браузер истории, изображенный на рисунке 5. Например, при развертывании некоторых изменений в демо-среде и за 10 минут до выхода в сеть, обнаруживается, что код содержит серьезную ошибку.

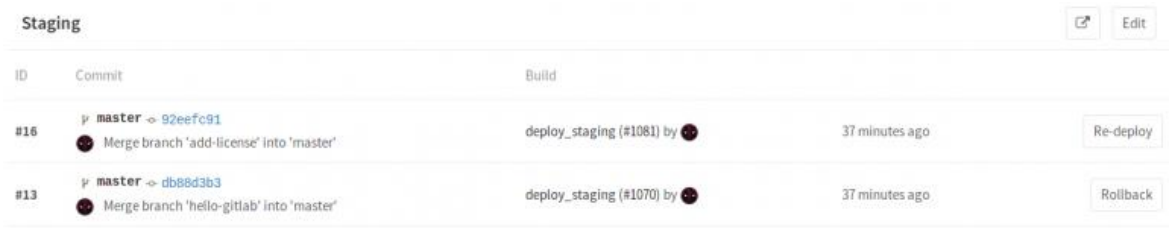

Рисунок 5 — Список развертываний в тестовом окружении

Эту ситуацию можно быстро исправить: очень легко получить доступ к истории того, что было развернуто в каждой среде, и выполнить откаты или повторно развернуть в любое предыдущее воспроизводимое состояние.

Список литературы

1. Continuous Integration для новичков [Электронный ресурс]. – Режим доступа: https://habr.com/post/352282/ (дата обращения: 22.12.2018).

2. CI/CD: принципы, внедрение, инструменты [Электронный ресурс]. – Режим доступа: https://medium.com/southbridge/ci-cd-принципы-внедрение-инструменты-f0626b9994c8 (дата обращения: 22.12.2018).

3. Непрерывная интеграция [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Непрерывная_интеграция (дата обращения: 22.12.2018).

4. Что такое непрерывная интеграция [Электронный ресурс]. – Режим доступа: https://ru.stackoverflow.com/questions/470453/Что-такое-непрерывная-интеграция/471034 (дата обращения: 22.12.2018).