УДК 004

Сравнение инструмента CI/CD двух сервисов Jenkins и GiLab

Голубь Илья Сергеевич
Приамурский государственный университет имени Шолом-Алейхема
Магистрант

Глаголев Владимир Александрович
Приамурский государственный университет имени Шолом-Алейхема
к.г.н., доцент кафедры информационных систем, математики и правовой информатики

Аннотация
Целью данной работы являет сравнение CI/CD инструментов Jenkins и GiLab. Методами исследования выбраны исследование теоретической информации по обоим продуктам и сравнение результатов. Изучив литературу и сравнив инструменты, были сделаны выводы о том, что данные инструменты имеют сильные и слабые стороны. Каждый из рассмотренных CI/CD-инструментов отличается некоторыми особенностями.
Ключевые слова: CI / DC, Jenkins, Pipeline, GiLab, -Doker.

Comparison of CI / CD tool of two services Jenkins and GitLab

Golub Ilya Sergeevich
Sholom Aleichem Priamursky State University
Undergraduate

Glagolev Vladimir Alexandrovich
Sholom Aleichem Priamursky State University
Candidate of Geographical Sciences, Associate Professor of the Department of Information Systems, Mathematics and Legal Informatics

Abstract
The purpose of this paper is to compare the CI / CD tools of Jenkins and GiLab. The research methods are the study of theoretical information on both products and the comparison of the results. After studying the literature and comparing the tools, it was concluded that these tools have strengths and weaknesses. Each of the considered CI / CD tools has some special features.
Keywords: CI / DC, Jenkins, Pipeline, GiLab, -Doker.

В своей работе J. F.Rey говорит о GitLab CE (Community Edition), это открытое ядро, бесплатное и полное решение для управления разработкой пакетов R, доставкой и многим другим. Мы фокусируемся на его части конвейера непрерывной интеграции и доставки с использованием Docker и

VirtualBox для упаковки R. Это локальное решение, используемое в нашей лаборатории, позволяет нам разрабатывать коды R в режиме совместной работы и автоматизировать проверку и сборку пакетов R. Этот процесс позволяет нам совместно использовать частные и / или разрабатываемые пакеты R на нескольких ОС и ускорить отправку CRAN за счет уменьшения количества ошибок проверки[3].

M. Berguerand рассказывает о реализация тестового конвейера, который должен иметь возможность компилировать код, программировать цель и устройство, моделирующее ее рабочую среду, и проверять правильное функционирование цели после моделирования[2].

В статье A. Tsalolikhin говориться о том, как реализовать конвейеры CI / CD в двух популярных инструментах, Jenkins и GitLab CI[4].

S. Arachchi, I. Perera рассказывают о том, как использование (CICD) инструментов повысило эффективность проектов с гибкими практиками и конвейерным подходом непрерывной интеграции и непрерывной доставки. В гибкой разработке новые функции вводятся в систему при каждой доставке спринта, и, хотя она может быть хорошо развита, сбои при доставке возможны из-за проблем с производительностью. Учитывая сроки доставки, переход на масштабирование системы является обычным решением в таких ситуациях. Но насколько масштабировать систему? Для масштабирования системы требуется текущий тестовый статус системы и ожидаемый статус системы. Тестирование производства - важная задача, поскольку она прерывает работу системы. Новая версия должна пройти нагрузочный тест, чтобы измерить ожидаемое состояние системы. Традиционные методы нагрузочного тестирования не могут определить поведение производительности в производственной среде, поскольку моделируемые шаблоны трафика сильно отличаются от производственных. Чтобы преодолеть эти проблемы, этот подход расширил конвейер CICD до трех этапов автоматизации: эталонного тестирования, нагрузочного тестирования и масштабирования. Он сводит к минимуму прерывание работы системы за счет использования тестового стенда при тестировании системы и использует производственный трафик для нагрузочного тестирования, что дает более точные результаты. После завершения этапов эталонного и нагрузочного тестирования можно оценить масштабирование системы. Изначально конвейер был разработан с использованием Jenkins CI-сервера, репозитория Git и репозитория Nexus с автоматизацией Ansible. Затем GoReplay используется для дублирования трафика из рабочей среды в среду тестирования. Мониторинг Nagios используется для анализа поведения системы на каждой фазе, и результаты тестового стенда показали, что масштабирование способно выдерживать ту же нагрузку при изменении прикладного программного обеспечения, но это не так. • оптимизировать время отклика приложения на значительном уровне и помогает снизить риск развертывания приложения за счет интеграции этого трехэтапного подхода в качестве расширенной функции автоматизации CICD. Таким образом,

исследование обеспечивает эффективный способ управления проектами CICD на основе Agile[4].

Целью данной работы являет сравнение CI/CD инструментов Jenkins и GiLab.

Аббревиатура CI (Continuous Integration) /CD (Continuous Delivery) означает – инструменты непрерывной интеграции и непрерывного развёртывания программного обеспечения. В последнее время данные инструменты получили большую известность. Развитие технологий интеграции разработки и эксплуатации программного обеспечения привело к росту спроса на CI/CD-инструменты. Существующие решения постоянно обновляются и оптимизируются, стремясь идти в ногу со временем, в мире контроля качества ПО постоянно появляется большое количество новых продуктов. При таком разнообразии выбора подбор подходящего инструментария оказывается нелёгкой задачей

Среди всех существующих инструментов CI/CD существуют два проекта, которые в данный момент являются самыми популярными.Это такой проект как Jenkins и инструмент GitLab CI/CD, который является частью платформы GitLab. У Jenkins имеется более 16000 звёзд на GitHub. Репозиторий GitLab же набрал на gitlab.com чуть больше 2000 звёзд. Если сравнить популярность данных репозиториев, то окажется, что проект Jenkins набрал в 8 раз больше звёзд, чем платформа, в состав которой входит GitLab CI/CD. Но при выборе CI/CD-инструмента это — далеко не единственный показатель, на который стоит обращать внимание. Есть и масса других, и это объясняет то, что во многих сравнениях Jenkins и GitLab CI/CD оказываются очень близко друг к другу.

Для более подробного сравнения необходимо рассмотреть инструменты подробнее.

Jenkins - это очень популярный и гибкий CI/CD-инструмент, автоматизирующий множество задач, которые связанны с программными проектами. Jenkins написан с помощью языка программирования Java и выпущен под лицензией MIT. Он обладает большим набором возможностей, направленных на автоматизацию задач, связанных со сборкой, тестированием, развёртыванием, интеграцией, выпуском программного обеспечения. Данный инструмент поддерживается большинством операционных систем таких как Widows, MacOS, и большенством дистрибутивов Linux. Существуют установочные пакеты Jenkins, предназначенные для различных ОС, этот инструмент можно установить в Docker и в любой системе, где есть JRE (Java Runtime Environment).

От разработчиков Jenkins существует ещё один проект, Jenkins X, который предназначен для работы в среде Kubernetes. В Jenkins X интегрированы Helm, сервер Jenkins CI/CD, Kubernetes и некоторые другие инструменты, предназначенные для создания CI/CD-конвейеров.

Скрипты Jenkins скрипты очень хорошо структурированы и понятны, что облегчает их чтение и понимание. Создатели Jenkins дополнительно написали более 1500 плагинов, которые направлены на организацию

интеграции Jenkins с самыми разными технологиями. В скриптах есть возможность реализовать систему аутентификации, что, позволит подключать различные закрыте системамы.

В процессе работы конвейера Jenkins можно наблюдать за тем, что происходит на каждом его шаге, за тем, успешно ли завершились те или иные этапы работы. Наблюдать за всем этим можно, но не применяя некий графический интерфейс, а пользуясь возможностями терминала.

Среди наиболее известных особенностей Jenkins можно выделить простоту настройки, высокий уровень автоматизации различных операций и отличную документацию. Если говорить о решении DevOps-задач, то здесь Jenkins считается весьма надёжным инструментом, используя который, как правило, нет смысла пристально наблюдать за всем процессом обработки проекта. В случае с другими CI/CD-инструментами это не так. Давайте поговорим о некоторых важнейших возможностях Jenkins.

Во-первых, это бесплатный инструмент, открытый исходный код, поддержка множества платформ. Как говорилось раньше, Jenkins использует лицензию MIT, что делает его бесплатным, а его исходный код открытым. Так же данный инструмент, работает на самых популярных операционных системах и может выполняться в виде сервлета в контейнерах, поддерживающих Java, в таких, как Apache Tomcat и GlassFish. Установка Jenkins качественно документирована.

Во-вторых, экосистема плагинов Jenkins гораздо более развита по сравнению с другими экосистемами подключаемых модулей CI/CD-инструментов. Эти плагины направлены на решение большого количества задач, с их помощью можно, например, произвести автоматизацию самых разных проектов. Богатство выбора бесплатных подключаемых модулей означает, что у того, кто использует Jenkins, нет острой необходимости в покупке дорогостоящих платных плагинов.

В-третьих, Jenkins довольно прост в установке и настройке. При этом процесс обновления тоже устроен очень органично. В документации Jenkins, можно найти ответы на самые разные вопросы, связанные с установкой и настройкой Jenkins.

В-четвертых, в Jenkins реализован REST API, что еще больше расширяет возможности системы. API представлен в трёх вариантах: XML, JSON с поддержкой JSONP, Python.

В-пятых, Jenkins поддерживает параллельное выполнение DevOps-задач. Он легко интегрируется с соответствующими инструментами, а так же отправляет уведомления о результатах выполнения задач. Выполнение тестирования кода можно ускорить за счёт организации параллельной сборки проекта с использованием различных виртуальных машин.

В-шестых, проект Jenkins позволяет использовать одновременно распределённые сборки с использованием нескольких компьютеров. Данная возможность используется в крупных проектах и завязана на схему работы, в соответствии с которой есть всего один главный сервер Jenkins и несколько подчинённых машин. Подчинённые машины могут использоваться и в

ситуациях, когда нужно организовать тестирование проекта в разных средах. Эти возможности выгодно отличают Jenkins от других подобных проектов.

GitLab CI/CD –это один самых новых и самых часто используемых DevOps-инженерами инструментов. Этот бесплатный инструмент, с открытым исходным кодом, встроен в систему контроля версий GitLab. У данной платформы есть community-версия, она поддерживает управление репозиториями, средства для отслеживания проблем, организацию код-ревью, механизмы, ориентированные на создание документации. Компании могут устанавливать GitLab локально, связывая эту систему с Active Directory и с LDAP-серверами для организации безопасной авторизации и аут

Инструмент GitLab CI/CD написан на Ruby и на Go. Они выпущены под лицензией MIT как и основной проект GitLab.

Интеграция данного инструмента в проект очень проста. Так же, при использовании этого инструмента обработка кода проекта разделяется на стадии, которые в свою очередь могут состоят из нескольких задач. Задачи поддаются тонкой настройке.

Задачи могут выполняться параллельно. После настройки последовательности стадий и задач CI/CD-конвейер готов к работе. В результате пользоваться GitLab CI/CD очень удобно, пожалуй, удобнее, чем другими подобными инструментами.

Проект отличается подробной и качественной документацией, его инструментами легко и удобно пользоваться. Для знакомства с GitLab CI/CD рассмотрим несколько возможностей этого инструмента Многие из этих возможностей интегрированы в саму платформу GitLab.

Во-первых, GitLab CI/CD это достаточно новый инструмент, нашедший широкое применение. GitLab CI/CD постепенно стал крайне популярным CI/CD-инструментом. Данный инструмент используют для автоматизированного тестирования и развёртывания программного обеспечения. Его просто настраивать. Это, к тому же, бесплатный CI/CD-инструмент, встроенный в платформу GitLab.

Во-вторых, это поддержка GitLab Pages и Jekyll. Jekyll — это генератор статических сайтов, который можно использовать в рамках системы GitLab Pages для создания сайтов на основе GitLab-репозиториев. Данная система использует материалы находящиеся в репозитории и автоматически генерирует на их основе статический сайт. Управлять внешним видом подобных сайтов можно, редактируя файл config.yml, используемый Jekyll.

В-третьих, благодаря масштабирование раннеров, которые выполняют конкретные задачи, можно получить экономию в части аренды серверных мощностей. Это важный фактор, особенно для проектов, которые тестируются параллельно. Кроме того, это важно для крупных проектов, состоящих из нескольких репозиториев.

В-четвертых, средства для отслеживания проблем имеют мощный функционал, а это привело к тому, что эту платформу используют многие опенсорсные проекты. GitLab CI/CD позволяет выполнять параллельное

тестирование различных веток кода. Это выгодно отличает GitLab CI/CD от Jenkins.

В-пятых, платформа GitLab поддерживает разграничение прав в репозитории. Это особенно актуально для корпоративных проектов.

В-шестых, вокруг GitLab сложилось большое сообщество, которое способствует развитию этой платформы и её инструментов, в частности - GitLab CI/CD. Глубокая интеграция GitLab CI/CD и GitLab, кроме прочего, упрощает нахождение ответов на вопросы, возникающие при работе с GitLab CI/CD.

В-седьмых, GitLab CI/CD - это система, которая работет не только с кодом, размещенным в его репозиториях. А, например, код можно хранить в репозитории похожего проекта – GitHub. При этом CI/CD-конвейер можно организовать на базе GitLab с использованием GitLab CI/CD.

Таблица 1 – Сравнение инструментов CI\CDГ

Характеристика	**Jenkins**	**GitLab CI/CD**
Открытый/закрытый исходный код	Открытый	Открытый
Инсталляция	Требуется	Не требуется, так как это — встроенная возможность платформы GitLab.
особенности	Поддержка плагинов	Глубокая интеграция в систему управления версиями.
Поддержка платформы	Отсутствует	Имеется
Сложности с установкой и настройкой	Сложностей не вызывают	Сложностей не вызывают
Самостоятельное развёртывание системы	Это - единственный вариант использования системы	Поддерживается
Создание CI/CD-конвейеров	Поддерживается, используется Jenkins Pipeline	Поддерживается

Мониторинг производительности приложений	Отсутствует	Имеется
Экосистема	Существует более 1000 плагинов	Система развивается в рамках GitLab
API	Поддерживает развитую систему API	Предлагает API для более глубокой интеграции в проекты
Поддержка JavaScript	Имеется	Имеется
Интеграция с другими инструментами	Поддерживается интеграция с другими инструментами и платформами (Slack, GitHub)	Множество средств для интеграции со сторонними системами, в частности — с GitHub и Kubernetes.
Контроль качества кода	Поддерживается — с помощью плагина SonarQube и других плагинов	Поддерживается

Описав и сравнив Jenkins и GitLab CI/CD, можно сосредоточиться их на различиях.

- GitLab CI/CD может полностью контролировать Git-репозитории. Данный инструмет может управлять ветками репозиториев и о некоторыми другими возможностями. А вот Jenkins, хотя и умеет работать с репозиториями, не даёт такого же уровня контроля над ними, как GitLab CI/CD.
- Jenkins — это бесплатный опенсорсный проект. Тот, кто его выбирает, разворачивает его самостоятельно. А GitLab CI/CD включён в состав платформы GitLab, это готовое решение.
- GitLab CI/CD поддерживает развитые средства управления задачами, работающие на уровне проектов. Эта сторона Jenkins развита слабее.

Так же кроме обычных необходимо рассмотреть сильные и слабые стороны этих двух проектов.

Сильными сторонами Jenkins являются:

- Достаточно большое количество плагинов.
- Контроль над установкой инструмента.
- Легкая отладка раннеров.
- Доступная настройка узлов.

- Простое развёртывание кода.
- Хохошо продуманная система управления учётными данными.
- Гибкость и универсальность.

А слабые стороны Jenkins это:

- Использование Jenkins для маленьких проектов может оказаться неоправданным по временным затратам.
- Отсутствие общих аналитических сведений по CI/CD-конвейерам.

Сильные стороны GitLab CI/CD:

- интеграция с Docker.
- Масштабирование раннеров.
- Параллельное выполнение задач, входящих в состав стадий CI/CD-конвейера.
- Использование модели ориентированного ациклического графа при настройке взаимоотношений задач.
- Лёгкость добавления задач.
- Надёжная система безопасности.

Слабые стороны GitLab CI/CD:

- Для каждой из задач нужно загружать/выгружать артефакты.
- Нельзя протестировать результаты объединения веток до их фактического объединения.
- При описании стадий CI/CD-конвейера в них пока нельзя выделять отдельные этапы.

Подводя итог сравнения Jenkins, и GitLab CI/CD можно сделать выводы, что данные инструменты имеют сильные и слабые стороны. Каждый из рассмотренных CI/CD-инструментов отличается некоторыми особенностями, хотя созданы эти инструменты для решения однотипных задач. При этом Jenkins является автономным инструментом, а GitLab CI/CD - это часть большой платформы, предназначенной для совместной работы над кодом.

Библиографический список

1. Arachchi S., Perera I. Continuous integration and continuous delivery pipeline automation for agile software project management // 2018 Moratuwa Engineering Research Conference (MERCon). 2018. С. 156-161.
2. Berguerand M. Gitlab Intégration Continue avec «On Target Testing». Haute Ecole d'Ingénierie, 2019.
3. Rey J. F., Houde L. R packaging development using GitLab CI/CD // useR! 2019. 2019. С. 1.
4. Tsalolikhin A. Setting Up CI/CD Pipelines with GitLab {CI} and Jenkins. 2018.