Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

УДК 658.51: 004.42

На правах рукописи

ЯРОШЕНКО
Александр Леонидович

АВТОМАТИЗАЦИЯ ПРОЦЕССОВ CONTINUOUS INTEGRATION И CONTINUOUS DELIVERY НА БАЗЕ ОПЕРАЦИОННОЙ СИСТЕМЫ LINUX

АВТОРЕФЕРАТ
диссертации на соискание степени
магистра техники и технологий

по специальности 1-39 81 01 – Компьютерные технологии
проектирования электронных систем

Минск 2019

Работа выполнена на кафедре проектирования информационно-компьютерных систем учреждения образования «Белорусский государственный университет информатики и радиоэлектроники»

Научный руководитель: **ПИСКУН Геннадий Адамович,**
кандидат технических наук, доцент, доцент кафедры проектирования информационно-компьютерных систем учреждения образования «Белорусский государственный университет информатики и радиоэлектроники»

Рецензент **НАПРАСНИКОВ Владимир Владимирович,**
кандидат технических наук, доцент, доцент кафедры «Программное обеспечение вычислительной техники и автоматизированных систем» учреждения образования «Белорусский национальный технический университет»

Защита диссертации состоится «26» июня 2019 г. года в $11^{\underline{00}}$ часов на заседании Государственной экзаменационной комиссии по защите магистерских диссертаций в учреждении образования «Белорусский государственный университет информатики и радиоэлектроники» по адресу: 220013, Минск, ул. П. Бровки, 6, копр. 1, ауд. 408, тел. 293-20-80, e-mail: kafpiks@bsuir.by

С диссертацией можно ознакомиться в библиотеке учреждения образования «Белорусский государственный университет информатики и радиоэлектроники».

ВВЕДЕНИЕ

В настоящее время повсеместно при разработке программного продукта необходима его сборка, тестирование и последующая доставка. Однако, с развитием программного продукта повышается и сложность его ручного обновления, и временные затраты на эти действия. Для автоматизации данного процесса существуют методологии *Continuous Integration* и *Continuous Delivery*.

Также, в процессе работы часто приходится обновлять сервисы и разворачивать их на конечные сервера. При разработке программного продукта в контексте микросервисной архитектуры встает вопрос о том, чтобы развертывания происходило часто и с запуском автоматических тестов. И, при расширении функционала, со временем увеличивается и выполнение однотипных задач, которое занимает все больше времени.

Реализация данных процессов представляет собой понимание архитектуры программного продукта для последующей автоматизации. Также, необходимо учитывать временные затраты на выбор подходящих для автоматизации инструментов и рефакторинг данного программного продукта для внедрения данной автоматизации. Однако, внедрение и затраты на реализацию текущих методологий повлечет за собой автоматизацию операций сборки, тестирования и доставки каждого отдельного компонента программного продукта для его последующей работы, что уменьшит временные затраты однотипных действий разработчиками, сосредоточив их внимание на написание программного кода, а также ускорит процессы нахождения ошибок и уязвимостей программного продукта, и ускорит его обновление.

Также, реализация предполагает наличие знаний в различных направлениях: компьютерные системы и сети, глубокое понимание процессов виртуализации, программировании, а также внутреннего устройства операционной системы.

Помимо этого, реализация предполагает наличие знаний в специализированных программных продуктах и инструментах, предназначенных для проведения вышеназванной автоматизации процесса разработки.

Учитывая все вышеперечисленное, разработка архитектуры автоматизации процессов разработки и внедрение их в работу предприятий повлечет за собой повышение качества выпускаемого программного продукта, что доказывает актуальность темы диссертации.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы исследования

Процесс автоматизации находит все более широкое применение в сфере информационных технологий, поскольку позволяет сократить время взаимодействия при работе с определенной системой. Одними из самых эффективных процессов автоматизации являются *Continuous Integration* и *Continuous Delivery*. Данные направления автоматизации являются быстроразвивающимися. В течение последнего времени стало появляться множество статей и исследований на данную тематику. В частности, данные процессы автоматизации активно поддерживают и применяют такие крупные компании, как *Amazon* и *Google*.

Степень разработанности проблемы

Исследование методологий *Continuous Integration* и *Continuous Delivery* осуществлялось на основе построения практических моделей с использованием работ как отечественных, так и зарубежных специалистов. В частности, стоит отметить работы Столярова Д.С., Азимова Р.М., Куковерова А.В., Алексеева В.Ф. и т.д.

Одним из недостатков исследований, представленных в современной технической литературе, является нехватка практических примеров, позволяющих охватить весь спектр задач одного рабочего процесса в контексте одного программного продукта.

Предложенное исследование направлено на устранение этого недостатка на основе разработки полноценной архитектуры, покрывающей процесс от программного кода до конечной его доставки на целевой сервер с определенным окружением.

Цель и задачи исследования

Целью магистерской диссертации является автоматизация процессов *Continuous Integration* и *Continuous Delivery* на базе операционной системы *Linux*, что позволит сократить временные затраты на сборку, тестирование и доставку исходного кода разрабатываемого программного продукта при минимальных временных затратах разработчика на различные выполняемые действия.

Поставленная цель работы определяет следующие основные задачи:

– провести обзор и анализ специфики методологий *Continuous Integration* и *Continuous Delivery*, а также указать их основные особенности в контексте автоматизации;

– разработать алгоритм функционирования процессов *Continuous Integration* и *Continuous Delivery*;

– экспериментально выявить преимущества спроектированной архитектуры относительно временных затрат на ручные действия, которые бы выполняли разработчики.

Область исследования

Содержание диссертации соответствует образовательному стандарту высшего образования второй ступени (магистратуры) специальности 1-39 81 01 «Компьютерные технологии проектирования электронных систем».

Теоретическая и методологическая основа исследования

В основу диссертации легли работы белорусских и зарубежных авторов в направлении *DevOps* (*Development Operations*), системного администрирования, сетевого администрирования, специалиста по базам данных, а также специалистов по работе с большими объемами данных.

Научная новизна

Научная новизна и значимость полученных результатов работы заключается в разработке архитектуры для автоматизации процессов *Continuous Integration* и *Continuous Delivery* на базе операционной системы *Linux*.

Теоретическая значимость работы заключается в детальном анализе процессов *Continuous Integration* и *Continuous Delivery*.

Практическая значимость диссертации состоит в разработанной архитектуре для автоматизации процессов *Continuous Integration* и *Continuous Delivery* на базе операционной системы *Linux*.

Основные положения, выносимые на защиту

1 Классификация методологий разработки программного продукта и их практическая значимость относительно автоматизации процесса *Continuous Integration*, а также целесообразность процесса *Continuous Delivery* для микросервисной архитектуры.

2 Разработка алгоритма для построения архитектуры, позволяющей автоматизировать процессы *Continuous Integration* и *Continuous Delivery*, а также осуществление этапа подготовки для его реализации.

3 Эксплуатация разработанной архитектуры с учетом динамического изменения нагрузки на каждый отдельный компонент программного продукта путем горизонтального масштабирования системы, а также с учетом балансировки нагрузки входящего трафика.

Публикации

Изложенные в диссертации основные положения и выводы опубликованы в 8 печатных работах. В их числе 2 работы опубликованы в сборниках материалов научных конференций и 6 тезисов приняты к опубликованию.

Структура и объем работы

Диссертация состоит из введения, общей характеристики работы, трех глав, заключения, библиографического списка и приложений.

В первой главе приведен обзор специфики процессов *Continuous Integration* и *Continuous Delivery*. Показаны основные особенности данных процессов, а также способы их применения на базе двух сервисов. **Во второй главе** представлены и обоснованы основные используемые инструменты, разработан алгоритм функционирования процессов *Continuous Integration* и *Continuous Delivery*, а также построена архитектура для автоматизации данных процессов. **В третьей главе** представлены результаты автоматизации процессов *Continuous Integration* и *Continuous Delivery*, а также показана их эффективность использования в рамках программного продукта.

Общий объем диссертационной работы составляет 118 страниц. Из них 76 страниц основного текста, 44 иллюстрации на 40 страницах, 2 таблицы на 2 страницах, библиографический список из 64 наименований на 6 страницах, список собственных публикаций соискателя из 8 наименований на 2 страницы, 5 приложений на 26 страницах.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во **введении** рассмотрено современное состояние проблемы отсутствия автоматизации процессов *Continuous Integration* и *Continuous Delivery*, а также представлено обоснование актуальности темы диссертации.

В **общей характеристике** работы показана актуальность проводимых исследований, степень разработанности проблемы, сформулированы цель и задачи диссертации, обозначена область исследований, научная (теоретическая и практическая) значимость исследований.

В **первой главе** рассмотрена основная сущность процессов *Continuous Integration* и *Continuous Delivery*, а также приведен общий вид процесса *Continuous Delivery*.

Представлены основные особенности автоматизации процессов *Continuous Integration* и *Continuous Delivery*, включающие в себя практическое сравнение ручного и автоматизированного развертывания двух сервисов: *Jenkins* и *GitLab*. Установлено, что сложность автоматизации зависит от

сложности разработанного программного продукта и его необходимых зависимых компонентов, из чего следует, что при отсутствии необходимости в горизонтальном масштабировании, итогового результата можно достичь стандартной установкой компонента.

Рассмотрены основные способы развертывания с использованием *Docker* и *Docker Compose*.

Во **второй главе** проведен анализ основных инструментов, необходимых для автоматизации процессов *Continuous Integration* и *Continuous Delivery*, а также обоснован выбор каждого из них.

На базе выбранных инструментов разработан алгоритм функционирования процессов *Continuous Integration* и *Continuous Delivery*, а также его последующие сборка метрик и агрегации выходных данных, необходимые разработчикам.

Алгоритм включает в себя несколько этапов:

1 Разработка программного продукта в виде микросервисной архитектуры.

2 Сохранение исходного кода в сервисе *GitLab* для последующего взаимодействия с ним.

3 Написание *Ansible*-плейбука, который будет взаимодействовать с репозиторием проекта и при вызове подставлять определенные значения для каждого конкретного микросервиса с использование зашифрованного хранилища *Ansible Vault*.

4 Написание инструкций *Dockerfile* для каждого микросервиса с учетом требований к безопасности локальной системы (чтобы процесс с *PID* равным единице не вышел за пределы контейнера под пользователем *root*) и оптимизации размера итогового образа для его последующей загрузки в *Docker*-репозиторий, а также с учетом особенностей разработки микросервиса.

5 Создание *Docker Compose* файла для запуска интеграционного тестирования всего кластера зависимостей программного продукта.

6 Создание и настройка *Jenkins CI* сервера, на котором будет производиться сборка *Docker*-образа, тестирование (модульное и интеграционное) внутри *Docker*-контейнера, а также запуск зависимых компонентов (например, база данных *PostgreSQL*), необходимых для проведения интеграционных тестов.

7 Создание и настройка кластера *Kubernetes* для работы с *Harbor Registry*, а также настройка основных компонентов кластера *Kubernetes*.

Основные компоненты, которые при этом будут использоваться, следующие: *Git*, *GitLab*, *Ansible*, *Jenkins CI*, *Docker*, *Harbor Registry*, *Kubernetes*.

Помимо автоматизации развертывания программного продукта необходимо производить мониторинг и сбор метрик работы различных сервисов, служб, структурных компонентов *Kubernetes*, баз данных, веб-сервера или отдельных программ и их визуализация в привычном человеку формате – в виде графиков и трендов (рисунок 1).

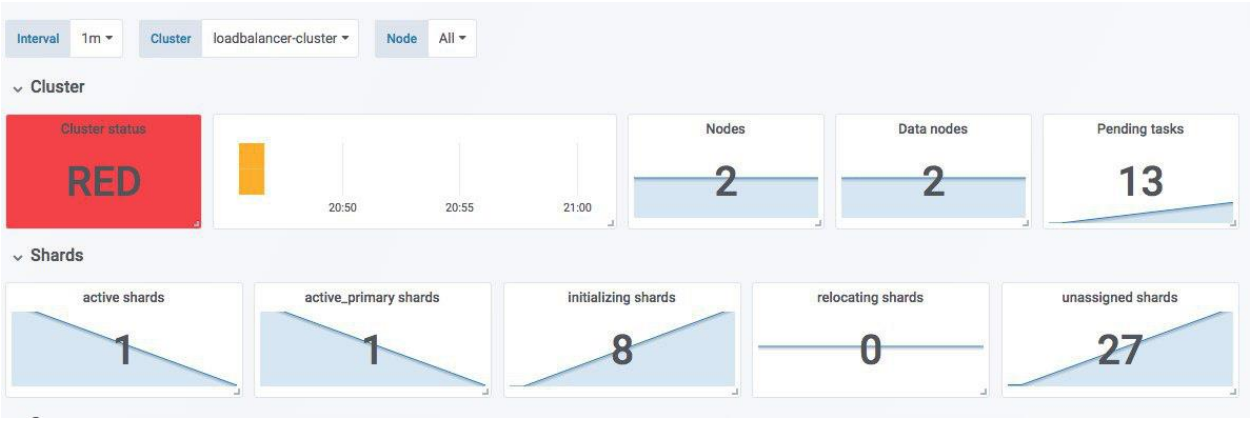

Рисунок 1 – Агрегированный сбор метрик

Основные компоненты, которые при этом будут использоваться, следующие: *Fluentd*, *Elasticsearch*, *Kibana*, *Prometheus*, *Grafana*.

Кроме того, важным фактором является возможность отслеживать потенциальные предупреждения и ошибки, которые возникли в процессе создания, обновления или возвращения на прошлую версию и оповещение об этом в виде сообщения в каком-либо сервисе сообщений или по электронной почты.

Проектирование архитектуры начинается с разработки программного продукта. Существуют два направления при разработке программного продукта: в виде монолитной (единой) архитектуры, а также в виде микросервисной архитектуры.

Монолитная архитектура представляет собой целостный программный продукт, все компоненты которого работают в качестве единого модуля. Из этого вытекает ряд недостатков:

– долгий процесс запуска программного продукта;

– при каких-либо изменениях необходимо перезагрузить весь программный продукт, а не отдельный компонент;

– невозможность распределенного развертывания программного продукта для более гибкого масштабирования системы;

– продолжительная отладка ошибки из-за большого количества модулей.

Микросервисная архитектура представляет собой совокупность компонентов программного продукта, реализованных в виде модуля, отдельного компонента, реализующего одну из функциональных частей программного продукта. Преимущества данного подхода обратно пропорциональны недостаткам монолитной архитектуры.

Однако, при всех этих преимуществах, присутствует один большой недостаток, заключающийся в больших количествах зависимостей между сервисами, из-за чего усложняется процесс разработки программного продукта (рисунок 2).

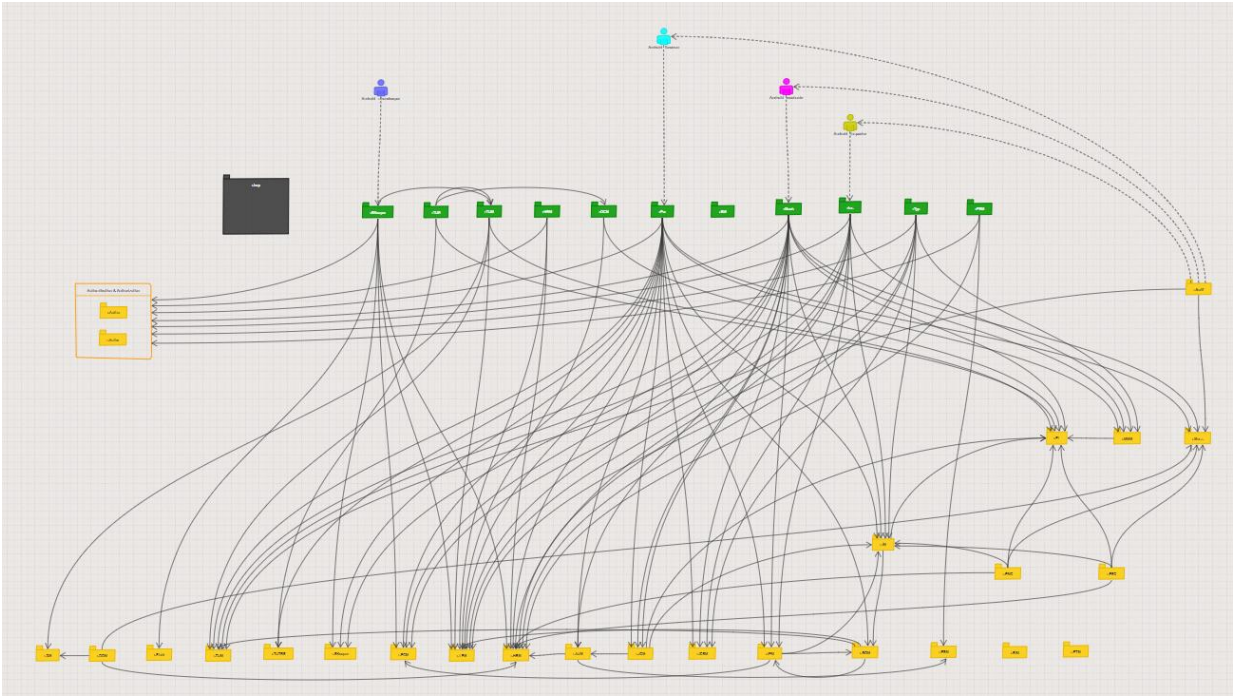

Рисунок 2 – Связь между микросервисами

Монолитная архитектура представляет собой целостный программный продукт, все компоненты которого работают в качестве единого модуля. Из этого вытекает ряд недостатков:

– долгий процесс запуска программного продукта;

– при каких-либо изменениях необходимо перезагрузить весь программный продукт, а не отдельный компонент;

– невозможность распределенного развертывания программного продукта для более гибкого масштабирования системы;

– продолжительная отладка ошибки из-за большого количества модулей.

При настройке *CI/CD* процессов следует учитывать, что помимо автоматической сборки производится процесс автоматического тестирования, позволяющее увидеть состояние результатов тестирования.

Для внедрения приложения в кластер *Kubernetes* необходимо создать конфигурационные файлы компонентов *deployment* и *service* в формате *yml*.

Для простого мониторинга кластера *Kubernetes* существует дополнение, которое называется *Kubernetes Dashboard*. Она представляет собой веб-интерфейс, взаимодействующий с кластером, на котором был запущен *Kubernetes Dashboard* (рисунок 3).

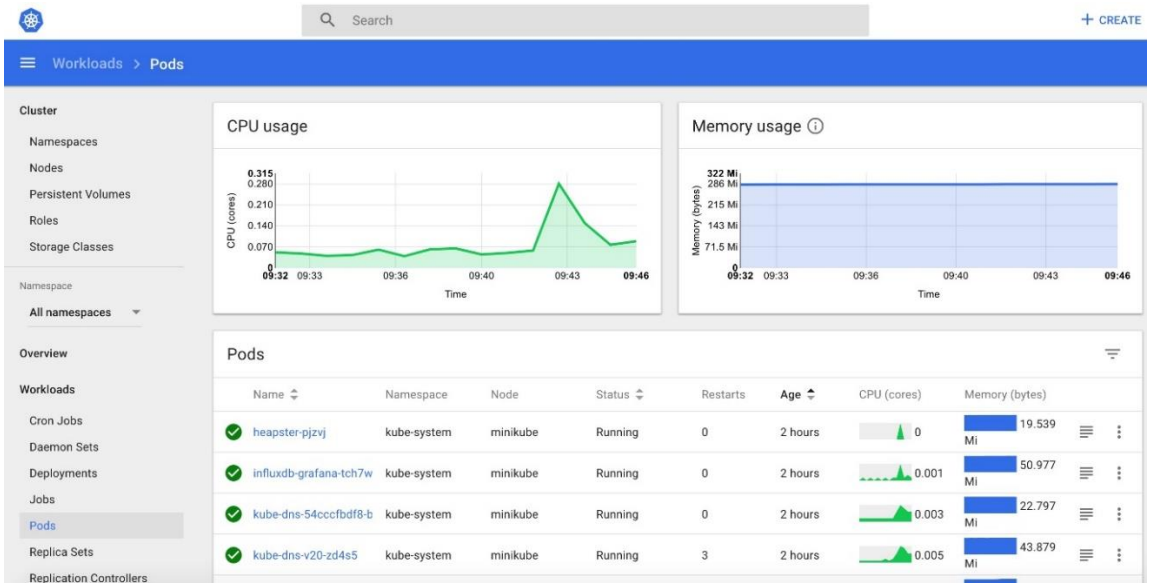

Рисунок 3 – Веб-интерфейс *Kubernetes Dashboard*

Данное дополнение позволит следить за различными компонентами кластера *Kubernetes* и отслеживать потенциальные предупреждения или ошибки. Однако, данная система является актуальной, когда не требуется распределенная система кластеров, а достаточно лишь одного. В ином случае, используются другие решения для мониторинга кластера *Kubernetes* и других компонентов системы.

В **третьей главе**, описаны результаты автоматизации, включающие в себя описание процесса внедрения архитектуры в контексте компании.

Постепенный переход всего программного продукта на автоматизированную систему означает изучение архитектуры самого программного продукта и изучение его ручной конфигурации, после чего адаптация под автоматизированную систему.

Поскольку программный продукт содержит несколько десятков микросервисов, то для каждого из них необходимо выделить следующие файлы:

1 *Ansible Playbook* для динамической замены переменных окружения и генерации конфигурационных файлов.

2 *Dockerfile* для сборки необходимого изолированного окружения для каждого микросервиса.

3 *Docker Compose* файл для запуска всего кластера необходимых микросервису зависимостей, а также проведения *JUnit*-тестов и интеграционных тестов.

4 *Jenkinsfile* для более гибкой настройки *Jenkins CI*.

5 *Kubernetes*-конфигурации, необходимые для работы микросервисов, содержатся в папке *.kube*/.

Результат разработки архитектуры представляет собой работающее в кластере *Kubernetes* приложение. Для этого используется *Kubernetes Plugin* для *Jenkins CI*, позволяющий взаимодействовать с кластерами *Kubernetes* через *Kubernetes API*, применяя конфигурационные файлы для кластера, тем самым производя обновление или создание подов, сервисов и реплик, позволяющие начать работу микросервису.

Основываясь на разработанной архитектуре, был сделан вывод, что после внедрения архитектуры автоматизации разработчик может сосредоточить внимание на разработке программного кода, следуя лишь настроенному в компании рабочему процессу, не выполняя лишних действий.

ЗАКЛЮЧЕНИЕ

Основные научные результаты диссертации

В магистерской диссертации были исследованы основные методологии, используемые при автоматизации процессов разработки: *Continuous Integration*. *Continuous Inspection*, *Continuous Delivery, Continuous Deployment*. Данные методологии образуют единый рабочий процесс интеграции, сборки, тестирования, доставки, управления, мониторинга и агрегация необходимых для разработчиков данных, что снизит как денежные затраты, так и затраты на внешние команды разработчиков, не связанных с разработкой программного кода.

Основными задачами при разработке архитектуры для автоматизации являются:

– анализ и исследование основных процессов, используемых для разработки текущей архитектуры;

– проработка основные требований и разработка дорожной карты (*roadmap*), позволяющий определить основные этапы и сложность их реализации на пути к итоговый архитектуре;

– выбор и обоснование основных инструментов и языков программирования, необходимых для построения текущей архитектуры;

– разработка алгоритма функционирования *CI*/*CD* процессов на базе выбранных инструментов и языков программирования;

– проектирования архитектуры с возможностью последующего горизонтального масштабирования программного продукта;

– тестирование и демонстрационное использование разработанной архитектуры в размах *development*-окружения исключая работу с большими объемами данных, но включая возможность их потери;

– использование принципа *DRY* (*Don't Repeat Yourself*), позволяющее шаблонизировать оставшиеся участки повторяющего кода или файлов;

– введение в эксплуатацию на *production*-окружении включая работу с большими объемами данных и нагрузкой.

Описанные выше задачи были решены, что привело к достижению поставленной цели: разработать архитектуру, позволяющую покрыть этапы интеграции, сборки, тестирования и доставки, а также осуществить динамическое распределение нагрузки и входящего трафика в зависимости от нагрузки на каждый отдельный компонент программного продукта путем горизонтального масштабирования.

Таким образом, разработанная архитектура позволит избавить разработчиков от каких-либо ручных действий, связанных с разработкой программного продукта.

Рекомендации по практическому использованию результатов

Полученные результаты внедрены в учебный процесс на кафедре проектирования информационно−компьютерных систем учреждения образования «Белорусский государственный университет информатики и радиоэлектроники» в учебный курс «Разработка *Web*-приложений для мобильных систем».

СПИСОК СОБСТВЕННЫХ ПУБЛИКАЦИЙ

1. Лось, Н.А. Автоматизация тестирования *Web*-приложений с использованием *Selenium WebDriver* / Н.А. Лось, А.Л. Ярошенко, В.Ф. Алексеев // материалы 13-ой международной молодежной научно-технической конференции «Современные проблемы радиоэлектроники и телекоммуникаций, РТ – 2017», Севастополь, Российская Федерация / УО «СГУ». – Севастополь, 2017. – С. 250.

2. Ярошенко, А.Л. Особенности обеспечения безопасности в программном средстве *Docker* / А.Л. Ярошенко, Н.А. Лось, Г.А. Пискун //

материалы 13-ой международной молодежной научно-технической конференции «Современные проблемы радиоэлектроники и телекоммуникаций, РТ – 2017», Севастополь, Российская Федерация / УО «СГУ». – Севастополь, 2017. – С. 296.

3. Лось, Н.А. Автоматизация модульного тестирования *Web*-приложений с помощью программных библиотек *xUnit* / Н.А. Лось, А.Л. Ярошенко // материалы 3-ей международной открытой конференции «Современные проблемы анализа динамических систем. Приложения в технике и технологиях», Воронеж, Российская Федерация / ФГБОУ ВО «Воронежский государственный лесотехнический университет имени Г.Ф. Морозова. – Воронеж, 2018. – С. 226-227.

4. Ярошенко, А.Л. Оптимизация образов *Docker* / А.Л. Ярошенко, Н.А. Лось // материалы 3-ей международной открытой конференции «Современные проблемы анализа динамических систем. Приложения в технике и технологиях», Воронеж, Российская Федерация / ФГБОУ ВО «Воронежский государственный лесотехнический университет имени Г.Ф. Морозова. – Воронеж, 2018. – С. 428-430.

5. Лось, Н.А. Автоматизация тестирования *REST* сервисов с помощью *REST Assured* / Н.А. Лось, А.Л. Ярошенко // материалы 54−ой науч. конф. аспирантов, магистрантов и студентов «Проектирование информационно−компьютерных систем», Минск, Респ. Беларусь, 23–27 апреля 2018 г. / УО «БГУИР». – Минск, 2018. – С. 74.

6. Ярошенко, А.Л. Сетевые решения *Docker* / **А**.Л. Ярошенко, Н.А. Лось // материалы 54−ой науч. конф. аспирантов, магистрантов и студентов «Проектирование информационно−компьютерных систем», Минск, Респ. Беларусь, 23–27 апреля 2018 г. / УО «БГУИР». – Минск, 2018. – С. 235.

7. Лось, Н.А. Автоматизация тестирования *Web*-приложений как часть процесса *Continuous Integration* / Н.А. Лось, А.Л. Ярошенко // материалы 55−ой науч. конф. аспирантов, магистрантов и студентов «Проектирование информационно−компьютерных систем», Минск, Респ. Беларусь, 22–26 апреля 2019 г. / УО «БГУИР». – Минск, 2019. – С. 138.

8. Ярошенко, А.Л. Основные абстракции оркестратора *Kubernetes* / **А**.Л. Ярошенко, Н.А. Лось // материалы 55−ой науч. конф. аспирантов, магистрантов и студентов «Проектирование информационно−компьютерных систем», Минск, Респ. Беларусь, 22–26 апреля 2019 г. / УО «БГУИР». – Минск, 2019. – С. 164.

РЭЗЮМЭ

Ярашэнка Аляксандр Леанідавіч

Аўтаматызацыя працэсаў Continuous Integration і Continuous Delivery на базе аперацыйнай сістэмы Linux

Ключавыя словы: Continuous Integration, Continuous Delivery.

Мэта працы: аўтаматызацыя працэсаў Continuous Integration і Continuous Delivery на базе аперацыйнай сістэмы Linux, якое дазволіць скараціць часовыя затраты на зборку, тэставанне і дастаўку зыходнага кода распрацоўванага праграмнага прадукту пры мінімальных часавых выдатках распрацоўніка на розныя дзеянні.

Атрыманыя вынікі і іх навізна: прыведзены агляд спецыфікі працэсаў Continuous Integration і Continuous Delivery. Паказаны асноўныя асаблівасці дадзеных працэсаў, а таксама спосабы іх прымянення на базе двух сэрвісаў.

Прадстаўлены і абгрунтаваныя асноўныя выкарыстоўваюцца інструменты, распрацаваны алгарытм функцыянавання працэсаў Continuous Integration і Continuous Delivery, а таксама пабудавана архітэктура для аўтаматызацыі дадзеных працэсаў.

Прадстаўлены вынікі аўтаматызацыі працэсаў Continuous Integration і Continuous Delivery, а таксама паказана іх эфектыўнасць выкарыстання ў рамках праграмнага прадукту.

Ступень выкарыстання: вынікі ўкаранены ў навучальны працэс на кафедре праектавання інфармацыйна-камп'ютэрных сістэм ўстановы образованія «Беларускі дзяржаўны універсітэт інфарматыкі і радиоэлектроники» у навучальны курс «Распрацоўка Web-прыкладанняў для мабільных сістэм».

Вобласць ужывання: аўтаматызацыя працэсаў распрацоўкі, распрацоўка праграмнага прадукту.

РЕЗЮМЕ

Ярошенко Александр Леонидович

Автоматизация процессов Continuous Integration и Continuous Delivery на базе операционной системы Linux

***Ключевые слова*:** Continuous Integration, Continuous Delivery.

***Цель работы*:** автоматизация процессов Continuous Integration и Continuous Delivery на базе операционной системы Linux, которое позволит сократить временные затраты на сборку, тестирование и доставку исходного кода разрабатываемого программного продукта при минимальных временных затратах разработчика на различные действия.

***Полученные результаты и их новизна*:** приведен обзор специфики процессов Continuous Integration и Continuous Delivery. Показаны основные особенности данных процессов, а также способы их применения на базе двух сервисов.

Представлены и обоснованы основные используемые инструменты, разработан алгоритм функционирования процессов Continuous Integration и Continuous Delivery, а также построена архитектура для автоматизации данных процессов.

Представлены результаты автоматизации процессов Continuous Integration и Continuous Delivery, а также показана их эффективность использования в рамках программного продукта.

***Степень использования*:** результаты внедрены в учебный процесс на кафедре проектирования информационно−компьютерных систем учреждения образования «Белорусский государственный университет информатики и радиоэлектроники» в учебный курс «Разработка Web-приложений для мобильных систем».

***Область применения*:** автоматизация процессов разработки, разработка программного продукта.

SUMMARY

Yarashenka Aliaksandr Leonidovich

Automation of Continuous Integration and Continuous Delivery processes based on Linux operating system

***Keywords*:** Continuous Integration, Continuous Delivery.

***The object of study*:** automation of the Continuous Integration and Continuous Delivery processes based on the Linux operating system, which will reduce the time spent on building, testing and delivering the source code of the software being developed with minimal developer time for various activities.

***The results and novelty*:** provides an overview of the specifics of the Continuous Integration and Continuous Delivery processes. The main features of these processes, as well as methods of their application on the basis of two services, are shown.

Presented and substantiated the main tools used, developed an algorithm for the functioning of the Continuous Integration and Continuous Delivery processes, as well as built an architecture to automate these processes.

The results of the automation of the Continuous Integration and Continuous Delivery processes are presented, and their effectiveness in the use of the software product is also shown.

***Degree of use*:** the results were introduced into the educational process at the Department of Design of Information and Computer Systems of the Educational Institution «Belarusian State University of Informatics and Radioelectronics» in the training course «Development of Web Applications for Mobile Systems».

Sphere of application: development process automation, software development.