

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Мегафакультет компьютерных технологий и управления
Факультет программной инженерии и компьютерной техники

СОГЛАСОВАНО

УТВЕРЖДАЮ

Руководитель профильной организации,
наименование организации, должность,
подпись

Академический руководитель
образовательной программы
«Вычислительные системы и сети 2017»
доцент факультета программной
инженерии, канд. пед. наук

«_____» _____ 2021 г.

_____ Е. Ю. Авксентьева
«_____» _____ 2021 г.

**АВТОМАТИЗАЦИИ УПРАВЛЕНИЯ ЖИЗНЕННЫМ
ЦИКЛОМ ВЕБ-СЕРВИСА**

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ (ПРЕДДИПЛОМНОЙ) ПРАКТИКИ

Исполнитель: Студент группы Р3555

_____ С. А. Федюкович

«_____» _____ 2021 г.

Содержание

ВВЕДЕНИЕ	3
1 ИНСТРУКТАЖ ОБУЧАЮЩЕГОСЯ	4
2 ОБЗОР ЛИТЕРАТУРЫ ПО ТЕМЕ ВКР	5
3 ОБЗОР И АНАЛИЗ СУЩЕСТВУЮЩИХ СИСТЕМ	8
4 ПРОЕКТИРОВАНИЕ СИСТЕМЫ	10
5 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ	12
6 ОФОРМЛЕНИЕ ВКР В СООТВЕТСТВИИ С ТРЕБОВАНИЯМИ К ВКР УНИВЕРСИТЕТА ИТМО	15

ВВЕДЕНИЕ

Согласно полученному заданию, автор данного отчета проходил преддипломную практику в организации “Университет ИТМО”. В рамках практики проводилась разработка программного комплекса по автоматизации управлением жизненного цикла веб-сервиса.

Перед началом выполнения работы было разработано техническое задание на проект, на основании которого была определена цель исследования: создание действующего опытного образца низко бюджетного программного комплекса предназначенного для подготовки рабочего окружения развёртки веб-сервисов. Для развёртки был получен проприетарный веб-сервис, разработанный на языке программирования Node.js, и использующий Docker образы. Процесс разработки и его результаты кратко описаны в данном отчете.

1. ИНСТРУКТАЖ ОБУЧАЮЩЕГОСЯ

22.11.2021 на территории профильной организации был проведен инструктаж обучающегося по ознакомлению с требованиями охраны труда, техники безопасности, пожарной безопасности, а также правилами внутреннего трудового распорядка.

После инструктажа, был составлен график выполнения работы, который был согласован с руководителем практики от профильной организации.

2. ОБЗОР ЛИТЕРАТУРЫ ПО ТЕМЕ ВКР

Данный этап практики выполнялся с 23.11.2021 по 27.11.2021. Автор отчета ознакомился с требованиями к выпускным квалификационным работам (утвержденным УС ИТМО 24.03.2020), подобрал литературу (книги, научные статьи, электронные ресурсы) не старше 5 лет по теме ВКР.

Для реализации цели работы, описанной во введении, были поставлены следующие задачи:

- 1) рассмотреть полученный для развёртки веб-сервис,
- 2) рассмотреть имеющиеся на рынке системы по автоматизации управления жизненным циклом веб-сервисов,
- 3) разработать принцип построения сервисной части комплекса,
- 4) выбрать компоненты для сервисной части комплекса,
- 5) произвести настройку составляющих элементов сервисной части комплекса,
- 6) разработать программную часть комплекса,
- 7) собрать опытный стенд, провести тестирование опытного образца комплекса.

Согласно поставленным задачам, предметами исследований в рамках работы являются параметры доступных технологических решений с последующим выбором наиболее подходящих для проекта, а также параметры доступных электронных и программных компонентов в рамках выбранного решения.

Объектами исследования являются системы автоматизации развёртки, представленные на рынке на момент выполнения данной работы, а также другие доступные для реализации программные средства. В процессе выполнения работы были просмотрены 22 источника, в том числе 12 электронных ресурсов, которые перечислены в соответствующем разделе отчета.

Актуальность темы.

Веб технологии широко распространились в нашем мире и на сегодняшний день почти каждая ВС взаимодействует со всемирной паутиной. В свою очередь, поддержка и разработка наиболее популярной архитектуры «Клиент-Сервер» таких систем требует существенных временных затрат, поскольку уже с самого начала проектирования требуется решить ряд следующих задач:

- вертикальное и горизонтальное масштабирование системы,
- доставка обновлений сервиса на рабочие ЭВМ,
- управление окружением ВС,
- осуществления контроля качества поступающих изменений,
- управление версиями ВС,
- бесшовное развёртывание отдельных компонентов системы,
- оперативная загрузка срочных исправлений.

Подавляющее большинство этих задач решается при помощи методологии SaaS (Software as a Service, «Программное обеспечение как услуга»). Основная идея заключается в предоставлении программного обеспечения по подписке. SaaS — это облачное решение при использовании которого, пользователь получает доступ к сервису, как правило, через браузер или по API. Оплачивает доступ и максимально быстро получает на руки готовый инструмент.

В данной работе сделан акцент на применение готовых SaaS решений, поскольку в условиях современной коммерческой разработки игнорирование таких технологий по большей части нецелесообразно и влечёт серьёзные временные, и как следствие, денежные потери.

Помимо SaaS так же используется методология IaC (Infrastructure as Code, «Инфраструктура как код») — это процесс управления и позиционирования дата центров и серверов с помощью машиночитаемых файлов определений, созданный как альтернатива физическому конфигурированию оборудования и оперируемым человеком инструментам. Теперь, вместо того, чтобы запускать сотню различных файлов конфигурации, IaC позволяет просто запускать скрипт, который управляет инфраструктурой и масштабированием системы.

Большинство таких средств использует в качестве основы Docker, который предоставляет инструментарий для контейнеризации программного обеспечения.

Так как в полученном для развёртки сервисе уже используется Node.js и Docker, то для корректной работы разрабатываемого комплекса требуется наличие регистров Node пакетов и Docker образов.

Так же в наши дни крайне распространена методология разработки CI/CD (Continuous Integration/Continuous Deployment, «Продолжительная интеграция/Продолжительная развёртка»). Она представляет собой автоматизацию тестирования и доставки новых проектов разрабатываемого проекта разработчикам, аналитикам, инженерам качества, конечным пользователям и другим заинтересованным сторонам. Метод обеспечивает оперативность вывода новой функциональности продукта и повышение качества разрабатываемого решения. Методология предоставляет интерфейс конвейеров (Pipelines), состоящий из последовательно выполняемых задач (Jobs).

В зависимости от решаемых задач и условий эксплуатации разработчиками выбираются конкретные инструменты из области рассмотренных выше методологий.

Разрабатываемая в рамках данной работы система автоматизации управления жизненным циклом веб-сервиса, согласно техническому заданию, должна отвечать следующим требованиям:

- взаимодействие с конечным пользователем по методологии SaaS,
- возможность добавления в систему не описанных заранее сервисов по методологии IaC,
- использование методологии CI/CD для взаимодействия с инфраструктурой и организации контроля качества,
- наличие конфигурации прав доступа (приватизации исходного кода),
- наличие хранилища NPM пакетов и регистра Docker образов,
- стоимость установки и обслуживания системы — не более 12 500 рублей за установку и не более 5 000 рублей в месяц за обслуживание на момент написания данной работы.

3. ОБЗОР И АНАЛИЗ СУЩЕСТВУЮЩИХ СИСТЕМ

С 28.11.2021 по 02.12.2021 выполнялся этап обзора и анализа существующих аналогичных программных комплексов.

Для корректного формулирования требований к разрабатываемому комплексу автоматизации управления жизненным циклом веб-сервиса, необходимо проанализировать и сравнить характеристики схожих по назначению решений, применяемых в данный момент на архитектуре «Клиент-Сервер», либо по своим характеристикам подходящих для такого применения.

Рассматривая рынок развёртки программного обеспечения, стоит начать с такого ключевого элемента, как, системы контроля версий (или VCS — Version Control System,). Так как в мире современных технологий большая часть практических задач решается именно такими системами.

Наиболее популярной системой контроля версий на момент написания работы является git. Git — система управления версиями программного обеспечения с распределенной архитектурой. В отличие от некогда популярных систем вроде CVS и Subversion (SVN), где полная история версий проекта доступна лишь в одном месте, в Git каждая рабочая копия кода сама по себе является репозиторием. Это позволяет всем разработчикам хранить историю изменений в полном объеме. Разработка в Git ориентирована на обеспечение высокой производительности, безопасности и гибкости распределенной системы.

Сам по себе git предоставляет только инструменты для локальной разработки и имеет весьма ограниченный функционал. Для работы в большинстве случаев выбирается SaaS Git хостинг исходного кода. Такие сервисы предоставляют удобный масштабируемый хостинг для Git-репозиториев с веб-интерфейсом для просмотра и редактирования кода, а также гибкими настройками доступа. Современные и простые способы организации процессов CI/CD и решения самых разных задач с их помощью.

В настоящее время на рынке имеется несколько лидирующих представителей, отличающихся в основном стоимостью и набором дополнительных инструментов.

Наиболее популярным из таких хостингов является GitHub от международной компании Microsoft. Для проектов с открытым исходным кодом сервис бесплатен. Функции приватизации и контроля доступа для команд предоставляются за отдельную плату. Для развёртки проектов по методологии CI/CD, GitHub предоставляет инструмент Actions. С его помощью автоматизация процессов развёртки и тестирования производится в декларативном стиле путём описания YAML файлов. Хранилище образов и пакетов предоставляется с ограничением и только проектам с открытым исходным кодом. В целом, данный хостинг лучше всего подходит для таких проектов и редко используется командами разработчиков в коммерческой сфере деятельности.

Следующим из таких хостингов является Bitbucket от австралийской компании Atlassian. Данный хостинг более акцентирован на коммерческую разработку и предоставляет функции приватизации бесплатно для небольших команд. Возможности CI/CD аналогичны GitHub Actions, только предоставляются продуктом Bamboo. Хранилище образов и пакетов отсутствует. Bitbucket следует рассматривать, как альтернативу GitHub только для коммерческой разработки.

Одним из наиболее подходящих хостингов является GitLab от украинских разработчиков (ныне зарубежной компании GitLab Inc). Ключевой особенностью является то, что этот сервис изначально разрабатывался, как полноценная система для управления жизненным циклом программного обеспечения на всех этапах разработки. Благодаря этому в нём реализовано большинство основных функций для обеспечения полноценного рабочего окружения по методологии CI/CD. Бесплатно сервис предоставляет, как приватизацию и управление доступом к исходному коду, так и хранилища образов и пакетов. Возможности непрерывной интеграции и доставки включают в себя отчеты о тестировании в реальном времени, параллельное выполне-

ние, локальный запуск скриптов, поддержку Docker. Несмотря на это, для ввода его в рабочий процесс всё равно необходима дополнительная разработка механизмов развёртки, начиная от описания YAML файлов конфигураций до разработки дополнительных программных средств.

Так же был рассмотрен продукт от разработчиков из Санкт-Петербурга (международной компании JetBrains) — Space. Данное решение является интегрированной средой для командной работы, которая включает управление исходным кодом, постановку и работу над задачами, управление командами разработчиков и инструменты коммуникации. Все функции Space так же предоставляет бесплатно, но с ограничениями в разумных пределах, подходящими небольшим командам разработчиков. Несмотря на большое количество возможностей, данный продукт существует на рынке сравнительно небольшое количество времени и большинство ключевых функций ещё не реализовано.

Помимо git хостинга данному проекту так же потребуется механизм масштабирования системы. Технология контейнеризации (Docker) позволяет запускать приложения в отдельных независимых средах — контейнерах. Они упрощают развертывание приложений, изолируют их друг от друга и ускоряют разработку. Но когда контейнеров становится слишком много, ими трудно управлять. Тут на помощь приходят IaC системы оркестрации.

У Docker есть стандартный инструмент оркестрации — Docker Swarm. Он поставляется вместе с Docker, довольно прост в настройке и позволяет создать кластер в кратчайшие сроки. Из минусов следует отметить узкую функциональность: возможности ограничены Docker API. Это значит, что Swarm способен сделать лишь то, что позволяют возможности Docker.

Альтернативным решением является Kubernetes — это универсальное средство для создания распределенных систем. Это комплексная система с большим количеством возможностей, разработка и поддержка которой самостоятельно значительно усложняется. Kubernetes мощный инструмент, имеющий много возможностей, которые позволяют строить действительно комплексные распределенные системы. К минусам относится управление, как оно использует отдельный набор команд и инструментов, несовместимых с Docker CLI. Гибкость данного решения излишне и лишь затруднит введение проекта в рабочее состояние в контексте данной работы.

Подробное сравнение git хостингов и оркестраторов представлено в тексте ВКР в виде таблиц.

Подводя итоги исследования доступных решений систем автоматизации управления жизненным циклом веб-сервисов, можно сделать следующие выводы:

- анализ источников показывает, что на рынке отсутствуют подходящие под требования проектируемой системы готовые решения, позволяющие осуществлять дальнейшие доработки системы под конкретные задачи;
- анализ использованных источников показывает, что цели и задачи работы возможно реализовать на основе Git хостинга GitLab, разработав программные механизмы части системы. Правильный выбор сервисных компонентов системы позволит выполнить требования технического задания и достигнуть цели работы.

4. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

Этап проектирования происходил в период с 03.12.2021 по 07.12.2021.

Проводя обзор доступных на рынке git хостингов, можно сделать вывод, что наиболее распространенным git хостингом на сегодняшний день является хостинг компании GitLab Inc. К тому же, по соотношению цена-функционал хостинг этой компании существенно обходит конкурентов. Также, к существенному преимуществу можно отнести наличие обширного сообщества пользователей и разработчиков программных решений на основе git хостинга GitLab, что позволяет иметь доступ к множеству готовых решений и получать помощь в разработке при необходимости.

Для реализации поставленной в данной работе задачи гибкость настройки всей инфраструктуры окружения не требуется, а также ставится в приоритет скорость ввода в рабочее состояние. Поэтому в качестве оркестратора вместо гибкости Kubernetes был выбран Docker Swarm. Но так как в данной работе делается акцент на гибкость всей системы, то далее будет рассмотрена описание конфигурации для использования с Kubernetes, не считая уставноку и настройку самого кластера. Для работы будет подготовлена одна вершина Docker Swarm в статусе менеджер, поскольку более не требуется на данном этапе.

Как было сказано ранее, для развёртки в работе используется Docker, поэтому необходим простой инструмент удаленного доступа к сокету Docker сервиса на рабочем сервере. Для этих целей будет использоваться GitLab Runner с установленным исполнителем задач Docker. Кратко описать работу GitLab Runner можно следующим образом: GitLab Runner запускается в отдельном контейнере с добавленным volume на сокет Docker, таким образом получается избежать достаточно сложного и нецелесообразного запуска Docker внутри Docker, так как в этом случае GitLab Runner получает доступ напрямую к сокету Docker сервера. Данное решение имеет потенциальную проблему с безопасностью, поскольку если злоумышленник получит доступ к описанию задач GitLab CI/CD, то он сможет запускать на рабочем сервере любое ПО. Для избежания данной проблемы будут установлены настройки доступа внутри GitLab. Так же для избежания потери полезного времени работы GitLab Runner, необходимо будет произвести настройку кэша GitLab Runner. Ключевой настройкой является политика загрузки образов для задач, поскольку по умолчанию GitLab Runner в любом случае будет загружать образ из регистра, даже если образ представлен локально. Согласно требованиям GitLab Runner должен будет запускать минимум три задачи за единицу времени, данное значение будет отражено в конфигурации на этапе реализации.

Автоматизация контроля качества будет реализована путём запуска описанных разработчиками тестовых сценариев внутри Pipelines. Отчёты по прохождению сценариев будут представлены в веб интерфейсе GitLab путём интеграции с CI/CD.

Для контроля версий будет использована модель ветвления git flow со следующими окружениями:

- develop — инсценировка рабочего окружения веб сервиса для разработчиков,
- testing — окружение для проведения ручного тестирования,
- release — рабочее окружение сервиса для реальных пользователей.

Основная проблема заключается в обновлении пакетов зависимостей проекта, поскольку для обновления необходимо пройти по зависимым сервисам и построить граф. Для решения этой проблемы будет разработано программное решение на базе CI/CD с применением bash скриптов при помощи package.json файлов веб сервиса.

Для проведения тестирования был составлен план содержащий набор ключевых функций для тестирования. Подробнее план представлен в виде таблицы с описанием сценариев, входных данных и ожидаемого результата в тексте работы ВКР.

5. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

В период 08.12.2021 - 14.12.2021 происходила программная реализация проекта.

Перед реализацией необходимо произвести базовую настройку окружения. Для этого был зарегистрирован GitLab аккаунт, установлен SSH ключ и создана группа проекта.

Следующим этапом было произведено создание необходимых репозиторий с последующей загрузкой исходного кода предоставленного для работы веб-сервиса:

- web-client — репозиторий для хранения исходного кода веб-клиента проекта,
- api — репозиторий для хранения исходного кода API проекта.
- node packages — репозиторий для хранения исходного кода общих npm зависимостей проекта.

Так же в корне каждого репозитория был загружен Dockerfile для развёртки данного сервиса. Так как в качестве модели ветвления была выбрана модель git flow, то так же были подготовлены соответствующие ветки в репозиториях под разные окружения: develop, testing и release. Установка доступа к данным репозиториям предоставляется только разработчикам данных программных решений в целях безопасности.

На следующем этапе был подготовлен репозиторий deployment для хранения общих скриптов и конфигураций окружения и развёртки. Установка доступа к этому репозиторию предоставляется только команде обеспечения развёртки в целях безопасности.

Подготовка хранилищ npm пакетов и Docker образов не требуется, так как GitLab берёт на себя данную ответственность и не требует дополнительных действий от пользователя.

Базовая подготовка окружения завершена. Следующим этапом был установлен и настроен Docker Swarm кластер. Одной и ключевой настройкой является открытие портов на уровне операционной системы сервера:

- TCP порт 2377 для коммуникации между менеджерами кластера,
- TCP и UDP порты 7946 для взаимодействия между нодами кластера,
- UDP порта 4789 для управления сетевым трафиком.

Дальше была произведена инициализация кластера на сервера. В данном этапе нет необходимости при использовании Kubernetes.

После была произведена установка и настройка GitLab Runner на рабочем сервере. В качестве дистрибутива на этапе проектирования был выбран Docker, как самый быстрый и удобный. После установки необходимо зарегистрировать GitLab Runner, для этого на странице настроек CI/CD группы в GitLab был получен регистрационный токен. Конфигурация runner производится путём редактирования config.toml файла в соответствии с этапом проектирования, основными настройками являются:

- Установка executor — docker executor,
- Установка volumes — /var/run/docker.sock,
- Установка pull-policy — if-not-present,
- Установка concurrent — 3.

На данном этапе runner полностью готов к работе и ожидает входящих задач.

На следующем этапе необходимо подготовить общие для работы сервисов конфигурации запуска, которые будут храниться в репозитории deployment:

- build.yaml — набор универсальных задач, предназначенный для сборки Docker образа и последующей загрузки в регистр контейнеров,
- publish.yaml — набор универсальных задач, предназначенный для оповещения кластера об обновлении сервиса для загрузки новой версии,
- publish-api.yaml — расширенная версия publish.yaml, содержащая дополнительные задачи для проведения миграция базы данных в случае необходимости,
- build-package.yaml — набор задач для отслеживания к какому конкретному сервису принадлежит npm пакет путём построения графа зависимостей на основании package.json файлов,
- publish-package.yaml — набор задач для загрузки собранного npm пакета.

Далее была произведена конфигурация на уровне сервисов компонентов, в репозитории api и web-client были добавлены конфигурации CI/CD путём создания .gitlab-ci.yml файла, содержащего следующие задачи:

- lint — осуществление контроля качества изменений путём проверки семантических правил написания кода,
- test — осуществление контроля качества путём запуска описанных разработчиками тестов кода,
- build — импортированная из репозитория common задача для сборки Docker образа и загрузки в регистр,
- publish — импортированная из репозитория common задача для оповещения кластера о загрузке новой версии (опциональная задача, требует подтверждения пользователем).

Аналогичные задачи были описаны для репозитория node packages.

Завершающим этапом реализации является описание конфигурационных файлов Docker Swarm. Для этого в настройках CI/CD репозитория deployment были добавлены переменные окружения (секреты) на все рабочие окружения (develop, testing и release), содержащие аргументы сервисов компонентов системы (доступы к базе данных, секрет ключа авторизации и так далее). Аналогично были добавлены Docker Swarm конфигурационные файлы под каждый окружения:

- develop — каждый сервис запускается на сервере в одном экземпляре, ресурсы сервера сильно ограничены во избежание лишней нагрузки,
- testing — аналогичен develop, только используется для целей ручного тестирования веб-сервиса,
- release — web-client запускается в одном экземпляре, база данных и api запускаются в трёх, большая часть ресурса отведена под эти сервисы.

Так же был подготовлен пример описания аналогичных файлов при использовании с Kubernetes.

Основная часть работы выполнена в полном объёме, дальше было произведено ручное тестирование функций программного комплекса согласно плану тестирования, составленному на этапе проектирования. В ходе тестирования все компоненты системы работали исправно и тесты ошибок не выявили.

Итоговая стоимость разработки с учётом только стоимости использования сервисов и обслуживания составляет 0 рублей за установку и 0 рублей в месяц за обслуживание.

6. ОФОРМЛЕНИЕ ВКР В СООТВЕТСТВИИ С ТРЕБОВАНИЯМИ К ВКР УНИВЕРСИТЕТА ИТМО

С 15.12.2021 по 18.12.2021 текст ВКР оформлялся в соответствии с требованиями к ВКР Университета ИТМО.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1) Сентерев Ю.А.: Выпускная квалификационная работа в вопросах и ответах. – С.: Университет ИТМО, 2017. – 64 с.
- 2) Скотт Граннеман: Linux. Карманный справочник. – С.: Вильямс, 2019. – 464 с..
- 3) Ким Д: Проект "Феникс". Как DevOps устраняет хаос и ускоряет развитие компании. – С.: Бомбора, 2020. – 384 с..
- 4) Дэвис Дженнифер, Дэниелс Кэтрин: Философия DevOps. Искусство управления IT. – С.: Питер, 2017. – 416 с..
- 5) Бетси Бейер, Site Reliability Engineering. Надежность и безотказность как в Google. – С.: Питер, 2019. – 592 с..
- 6) Арундел Джон, Kubernetes для DevOps: развертывание, запуск и масштабирование в облаке. – С.: Питер, 2020. – 494 с..
- 7) Джин Ким, Ускоряйся! Наука DevOps. – С.: Интеллектуальная Литература, 2020. – 224 с..
- 8) Вольф Эберхард, Continuous delivery. Практика непрерывных апдейтов. – С.: Питер, 2018. – 320 с..
- 9) Хамбл Джек, Непрерывное развертывание ПО. Автоматизация процессов сборки, тестирования и внедрения новых версий программ. – С.: Вильямс, 2016. – 432 с..
- 10) Документация GitLab [Электронный ресурс]. — <https://docs.gitlab.com> (дата обращения 25.11.2021).
- 11) Swarm mode overview [Электронный ресурс]. — <https://docs.docker.com/engine/swarm/> (дата обращения 25.11.2021).
- 12) Документация npm [Электронный ресурс]. — <https://docs.npmjs.com> (дата обращения 25.11.2021).
- 13) Документация Kubernetes [Электронный ресурс]. — <https://kubernetes.io/ru/docs/home/> (дата обращения 25.11.2021).
- 14) Документация Docker [Электронный ресурс]. — <https://docs.docker.com> (дата обращения 25.11.2021).
- 15) GitHub [Электронный ресурс]. — <https://github.com/> (дата обращения 25.11.2021).
- 16) JetBrains Space [Электронный ресурс]. — <https://www.jetbrains.com/space/> (дата обращения 25.11.2021).
- 17) Git — Book [Электронный ресурс]. — <https://git-scm.com/book/en/v2/> (дата обращения 25.11.2021).
- 18) Использование BitBucket [Электронный ресурс]. — <https://bitbucket.org/product/ru/guides/> (дата обращения 25.11.2021).

- 19) Рейтинг сервисов-репозиторийов для хранения кода 2018 [Электронный ресурс]. — <https://tagline.ru/source-code-repository-rating/> (дата обращения 25.11.2021).
- 20) GitLab [Электронный ресурс]. — <https://gitlab.com> (дата обращения 25.11.2021).
- 21) Чем полезен Docker Swarm и в каких случаях лучше использовать Kubernetes [Электронный ресурс]. — <https://mcs.mail.ru/blog/docker-swarm-ili-kubernetes-cto-luchshe> (дата обращения 25.11.2021).