МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра информатики и программирования

РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЧЕСКОГО РАЗВЕРТЫВАНИЯ ТЕСТИРУЮЩЕЙ СРЕДЫ ПО ПРОВЕРКЕ ПРАКТИЧЕСКИХ НАВЫКОВ УДАЛЕННОЙ РАБОТЫ С LINUX СЕРВЕРАМИ

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 271 группы
направления 09.04.01 — Информатика и вычислительная техника
факультета КНиИТ
Полинского Артема Владиславовича

Научный руководитель
старший преподаватель _____ М. С. Портенко

Заведующий кафедрой
доцент, к. ф.-м. н. _____ М. В. Огнева

Саратов 2019

ВВЕДЕНИЕ

Оценка персонала это действие по определению ценности работника для организации. В качестве основания для определения данной ценности выступает эффективность достижения целей организации. Тем самым оценка персонала является самым важным фактором при приеме на работу.

Тестирование широко применяется для оценки уровня знаний при приеме на работу, для оценки квалификации персонала. Испытуемому предлагается ряд вопросов (тест), на которые он должен ответить.

В рамках данной работы была поставлена цель: разработать систему автоматического развертывания тестирующей среды по проверке практических навыков удаленной работы с Linux серверами.

Для этого необходимо решить следующие задачи:

— выбрать язык, систем управления конфигурациями;
— рассмотреть системы Service Discovery;
— выбрать в качестве основы виртуальные машины или контейнеры для запуска практических заданий;
— реализовать систему предварительного тестирования на языке Go;
— реализовать систему для автоматической разворачивания инфраструктуры и проверку практических заданий.

Магистерская работа состоит из введения, 7 разделов, заключения, списка использованных источников и 5 приложений. Общий объем работы – 138 страниц, из них 89 страниц – основное содержание, включая 59 рисунков и 3 таблиц, цифровой носитель в качестве приложения, список использованных источников информации – 69 наименований.

1 КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Обзор и анализ существующего инструментария: языки, платформы, среды» посвящен анализу существующих систем повышения квалификации сотрудников ИТ сферы, описанию технология проведения тестирования и классификация тестирования, описанию языка программирования Golang: синтаксис Golang, использование массивов Golang, инициализации переменных в языке Go и отличие конструкций For,Switch,If, использование функции в языке Go.

Одной из компаний занимающиеся непосредственно созданием сетевого оборудования, используемого инженерами связи по всему миру является американская компания Cisco — мировой лидер в области сетевых технологий. Компания Cisco проводит обучение по сетевым технологиям дистанционно. Для этого организованна Сетевая Академия CISCO (CNA). Студентам академии выдаются логин и пароль, с помощью которого они получают доступ к информационным ресурсам (лекциям), выполняют лабораторные работы и сдают экзамены. Обучение состоит из четырех ступеней. После успешного завершения обучения студентам выдается сертификат, подтверждающий квалификацию студента. Система дистанционного обучения CNA разработана самой компанией CISCO и организована через веб-интерфейс, что делает ее доступной для всех пользователей сети Интернет, желающих стать специалистом в области сетевой инженерии [1].

Корпорация Oracle – крупнейший в мире поставщик программного обеспечения для управления информацией и вторая в мире компания по поставке программного обеспечения. В 2015-2016 годах компания Oracle запустила свой проект под названием Oracle Academy для развития обучения компьютерным технологиям, обеспечению их доступности для учащихся по всему миру. Курсы доступны в нескольких вариантах: учебные центры при университетах или специальные семинары, которые проходят по записи в регионах [2].

Существуют также готовые решения для организации дистанционного обучения. К таким системам можно отнести проект Moodle [3]. Он распространяется по бесплатной лицензии. К плюсам Moodle можно отнести то, что процесс обучения в этой системе, как и в «Сетевой Академии CISCO», организован через веб-интерфейс, что позволяет использовать систему большему

количеству пользователей. Эта система создана инициативной группой программистов, которая пыталась объединить потребности всех образовательных учреждений, желающих проводить обучение дистанционно. Именно поэтому проект при первом знакомстве может показаться немного сложным и непонятным. Существует так же множество модулей, которые могут и не понадобятся в процессе работы системы.

Go – это язык программирования, который был разработан корпорацией Google. Более распространенное название «Golang» – сокращение от “Google language”. Это компилируемый и многопоточный язык, релиз которого состоялся в ноябре 2009 года. Одной из ключевых фигур, которые принимали участие в создании Golang, является Роб Пайк, известный разработчик языков программирования, а также операционных систем, в данный момент работающий в Google. Выступая на одной из конференций, он отметил, что язык Go – это попытка перенять лучшие стороны таких языков, как C++ и Java.

Язык Go разрабатывался как язык для создания различных высокоэффективных программ, однако лучше всего он подходит для создания веб-приложений (в качестве back-end). При этом Go дает возможности писать и другие проекты. Применение языка Go ограничивается тремя основными направлениями: сетевое программное обеспечение, консольные утилиты и бэкенд.

Одной из отличительных особенностей языка является оригинальная система типов: в языке отсутствует наследование, а это один из принципов объектно-ориентированного программирования. Также в Go используется сокращенный синтаксис определения переменных и синтаксис анонимных функций.

Еще одна особенность этого языка – параллелизм, который заключается в том, что любая функция может быть выполнена одновременно с другой.

Второй раздел «Сравнительный анализ систем управления конфигурациями: Ansible, Puppet, Chef;» посвящен анализ средств управления конфигурацией серверов : Ansible, Puppet, Chef, описанию использования инструмента управления конфигурациями Ansible: структура Ansible, использование переменных в Ansible, использование Playbooks, защита данных и создание зашифрованных файлов в Ansible.

Puppet, отличается от Chef и Ansible, тем что предоставляет наиболее обширный набор доступных действий, модулей и пользовательских интерфейсов. Puppet представляет больше возможности для оркестровки центров обработки данных, охватывая практически каждую операционную систему и предлагая инструменты для основных ОС. Puppet настраивается относительно просто, требуется установка master server и agents для каждой системы, которая должна управляться.

Chef похож на Puppet с точки зрения общей концепции, поскольку на управляемых узлах устанавливается agent. Но Chef отличается в развертывании, к требованию развертывания master-server требуется рабочая станция для управления мастером. Агенты могут быть установлены с рабочей станции с помощью SSH. После этого agents аутентифицируются с master-server при помощи сертификатов.

Конфигурация Chef построена вокруг технологии Git, поэтому знание того, как работает Git, является необходимым условием для работы с Chef.

Опрос agents мастера называется pull, отправка инструкций от master server к agents называется push.

Как и Puppet, Chef основан на Ruby, поэтому знание Ruby также требуется. Как и в случае с Puppet, модули могут быть загружены или написаны с нуля и развернуты на управляемых узлах.

В отличие от Puppet, Chef еще не имеет хорошо сформированной функции push. Это означает, что agent периодически опрашивает master-server на наличие изменений и применяют их.

Ansible – это инструмент развертывания и управления ресурсами с открытым исходным кодом. Он во многом уникален по сравнению с другими инструментами управления из-за стремления обеспечить прирост производительности для широкого круга задач автоматизации. В то время как Ansible предоставляет более производительные аналоги основных возможностей для автоматизации по сравнению с другими продуктами в этой сфере, она также стремится решать другие основные нерешенные задачи: объединить конфигурацию, развертывание и комплексную оркестровку процессов [4].

Установка Ansible может быть выполнена через клон репозитория Git. После этого настраивается SSH ключи и пользователи для управляемых узлов. Как только это будет сделано, master сервер может связываться с узлом

через SSH и выполнять все требуемые задачи. Чтобы работать с операционными системами Ansible автоматически выполняет все команды из-под sudo для запуска команд с правами root [5].

При более внимательном рассмотрении можно выделить ряд особенностей изученных систем, таких как: высокая степень масштабируемости CFEngine за счет сильно распределенной архитектуры, удобство управления инфраструктурой со множеством различных конфигураций Puppet с помощью специфического языка описания управления узлов, наличие максимально большого количества готовых решений Chef, отсутствие необходимости дистрибуции дополнительного ПО на управляемый узел Ansible.

Согласно общей архитектуре системы автоматического развертывания тестирующей среды по проверке практических навыков удаленной работы с Linux серверами можно утверждать, что для выполнения текущей задачи необходима система с централизованной архитектурой, без агентов, при этом, возможностью максимально дистанцироваться от взаимодействия с ними. С учетом данных требований, отмеченных особенностей инструментов, для дальнейшего практического изучения в рамках текущей задачи был выбран Ansible, как средство управления.

Третий раздел «Рассмотрение систем Service Discovery: etcd, Consul, и расширения Consul-Template, связку confd и etcd» посвящен анализу и сравнению существующих систем Service Discovery, описания принципа работы ZooKeeper, Etcd, Consul, Confd.

Service discovery – это то, как приложения или службы обнаруживают друг друга в сети. Концепция service discovery – это старая концепция, которая эволюционировала по мере развития компьютерных архитектур. На заре эпохи создания сети разные компьютеры должны были найти друг друга, и это было сделано через один глобальный текстовый файл HOSTS. Адреса добавлялись вручную, поскольку новые хосты редко добавлялись. По мере роста Интернета, хосты добавлялись все чаще, и возникла необходимость автоматизации и масштабируемости системы. Это привело к изобретению и широкому распространению DNS.

ZooKeeper – это один из первых проектов service discovery. Он возник из проекта Apache Hadoop, Apache Hadoop был написан для помощи в обслуживании различных компонентов в кластере Hadoop. Он надежный и ис-

пользуется многими крупными компаниями (YouTube, eBay, Yahoo). Формат хранящихся данных аналогичен организации файловой системы. Если запустить на кластере серверов, Zookeeper будет делиться состоянием конфигурации со всеми узлами. Каждый кластер выбирает лидера, и клиенты могут подключаться к любому из серверов для извлечения данных [6, 7].

Etcd – key/value хранилище, доступное через HTTP. Оно распределено и имеет иерархическую конфигурационную систему, которая может использоваться для создания service discovery. Данное хранилище очень простое в развертывании, настройке и использовании, обеспечивает надежную сохранность данных, безопасность и имеет подробную документацию.

Confd - упрощённый инструмент управления конфигурацией. Данный инструмент хранит файлы конфигурации в рабочем состоянии, используя данные, хранящиеся в etcd, consul. Он также может использоваться для перезагрузки приложений при изменении конфигурационных файлов. Другими словами, мы можем использовать его для перенастройки и перезапуска компонентов с информацией, хранящейся в etcd.

Consul – это сильное децентрализованное хранилище данных, которое использует gossip для формирования динамических кластеров. Он имеет иерархическое key/value хранилище, которое может использоваться не только для хранения данных, но и для запуска скриптов, которые могут использоваться для различных задач, от отправки уведомлений об изменениях данных до запуска проверок работоспособности и в зависимости от их вывода выполнять действия.

Четвертый раздел «Преимущества и недостатки виртуализации при помощи контейнеров и виртуальных машин» посвящен анализу виртуализций и их разновидностей, выбору между контейнерами и виртуальными машинами, проведению сравнения виртуальных машин и контейнеров.

Виртуализация на уровне операционной системы или контейнерная виртуализация – позволяет использовать несколько виртуальных сред, не отходя от основной операционной системы. Она использует функциональность ядра для изоляции групп процессов друг от друга и остальной части системы. Изолированные среды называются контейнерами. Они имеют одно и то же ядро ОС, поэтому гипервизор не требуется. Данный тип виртуализации так-

же называют контейнеризацией (рисунок **??**).

Контейнеры – это платформы для разработки и развертывания приложений, не обращая внимания на инфраструктуру. Контейнерная методология позволяет разработчикам выполнять быстрое развертывание и значительно сокращает задержку между написанием кода и его внедрением. Контейнеры обеспечивают возможность упаковки и запуска приложения в слабо изолированной среде, что позволяет одновременно запускать несколько контейнеров на данном сервере. В отличие от виртуальных машин, контейнеры не имеют накладных расходов на гипервизоры.

Контейнеры имеют много преимуществ по сравнению с виртуальными машинами. В следующем списке представлены некоторые из них:

— легковесность из-за отсутствия гипервизора;
— простота развертывания;
— производительность и быстрота – поскольку приложения запускаются непосредственно на ядре, запуск приложения происходит почти мгновенно;
— изоляция контейнеров друг от друга и от хоста для повышения производительности и безопасности;
— снижение затрат за счет оптимизации ресурсов;
— инкапсуляция и портативность – код приложения со всеми зависимостями может быть перемещен внутрь контейнера и может быть запущен на любом хосте с установленным Docker и ядром Linux;
— социальный обмен - хранение и совместное использование образов;
— масштабируемость.

Пятый раздел «Развертывание приложения при помощи Docker» посвящен анализу возможностей контейнеризации инструмента Docker, описание принципа работы Docker.

В данный момент Docker – это инструмент, который позволяет полностью изолировать приложение. Обеспечивает автоматизацию развёртывания и управления приложениями. Предоставляет удобный интерфейс для работы с LXC. Позволяет создавать переносимые контейнеры с окружением и всеми зависимостями. Также есть возможность управления вычислительными ресурсами контеийнера.

Приложения в Docker работают в изолированной среде (построенной с

помощью пространств имён, namespaces, и групп процессов, cgroups), с одной стороны изолируя процессы друг от друга, с другой стороны, не прибегая при этом к таким избыточным средствам как виртуализация или эмуляция. Docker использует файловую систему AuFS, благодаря которой контейнеры могут совместно использовать одинаковые части файловой системы, доступные только на чтение.

Docker является альтернативой виртуальным машинам на основе гипервизора. Он полезен при высоких нагрузках, например, при разработке собственного облака или PaaS. Его также можно использовать для средних и малых приложений, когда требуется получить больше от имеющихся ресурсов.

Шестой раздел «Изучение распределенной транзакционной база данных CockroachDB» посвящен описанию и особенностям использования распределенной базы данных CockroachDB.

CockroachDB — авто-масштабируемая распределенная SQL-база данных. CockroachDB разработан под впечатлением от технологий Google Spanner, но в отличие от него является полностью открытым продуктом. Код проекта написан на языке Go и распространяется под лицензией Apache 2.0 [8]. Из наиболее подходящих для CockroachDB применений отмечается организация хранения данных приложений, от которых требуется постоянная доступность и целостность данных, а также гибкая масштабируемость, расширение сводится к добавлению новых узлов, которые автоматически включаются в кластер [9].

CockroachDB предоставляет средства для автоматической репликации, ребалансировки хранилища, обнаружения сбоев и восстановления, при минимальной настройке и обслуживании. CockroachDB подходит для создания распределённых хранилищ и облачных решений, развёртываемых поверх нескольких центров обработки данных. Система обработки транзакций соответствует требованиям ACID (Atomicity, Consistency, Isolation, Durability) [10]. Обмен данными между узлами производится с использованием шифрования. Аутентификация выполняется на основе SSL-сертификатов. Для клиентов предусмотрена система разделения привилегий. Для приложений предоставляется высокоуровневый SQL API (урезанное подмножество SQL), совместимый с клиентскими драйверами для PostgreSQL [11].

Седьмой раздел «Реализация системы для проверки теоритических и практических навыков» посвящен описанию непрерывной интеграции и непрерывной доставке, свойств и функций непрерывной интеграции, сравнению фреймворков для непрерывной интеграции, реализации системы предварительного тестирования на языке Go, реализации модуля для получения доступу к выполнению практического задания, реализации системы для автоматической конфигурации и разворачивания инфраструктуры для системы проверки практических заданий.

Jenkins, TeamCity и Bamboo – это инструменты для непрерывной интеграции, автоматизированных сборок, автоматизированного тестирования и непрерывной доставки. Все они предлагают интеграции, которые расширяет их возможности. Все три инструмента предоставляют поддержку и документацию.

Jenkins – самый популярный инструмент CI/CD с открытым исходным кодом на рынке. Проект был начат Kohsuke Kawaguchi в 2004 году, и первоначально он назывался Hudson, но в 2011 году он был переименован в Jenkins из-за споров с Oracle. [12–14]

Bamboo – это CI/CD-сервер от Atlassian. Как и другие серверы CI/CD, Bamboo позволяет разработчикам автоматически создавать, интегрировать и тестировать исходный код, а затем использовать приложение для развертывания. Bamboo также бесперебойно работает с другими инструментами Atlasian, такими как Jira (управление проектомами) и Hipchat (групповое общение).

TeamCity – это еще один коммерческий сервер CI/CD, разрабатывается фирмой JetBrains. Он известен своей невероятно простой настройкой и красивым пользовательским интерфейсом. Он имеет мощный набор функций и большое количество плагинов.

Проектируемая система разрабатывается для ПРЦ НИТ СГУ (Поволжский региональный центр новых информационных технологий СГУ), в которых одной из форм проверки знаний при приеме на работу является тестирование.

Система будет состоять из нескольких частей – интерфейса для тестирующего с возможностями добавления заданий и просмотра результатов тестирования и интерфейса, тестируемого для непосредственного прохождения

теста по теме, и организация доступа к практическим заданиям.

Интерфейс тестирующего предоставляе возможность выполнять следующие действия:

— добавлять и удалять вопросы;
— выделять правильный вариант ответа;
— выделять правильный вариант ответа.

Интерфейс для тестируемого обладает следующими возможностями:

— работа с базой данных заданий;
— прохождение тестирования;
— отображение вопросов и ответов;
— случайный порядок вопросов для тестирования;
— отображение подсчета правильных и неправильных ответов.

Аутентификация пользователей настроена с использованием метода входа через логин и пароль. Пароль хранится в базе данных в виде sha256(sha256(пароль) + соль).

Terraform – это инструмент от компании Hashicorp, помогающий декларативно управлять инфраструктрой. Преимущества использования Terraform вместо Ansible, Chef или Puppet:

— Оркестровка, а не просто конфигурация ресурсов;
— Неизменная инфраструктура;
— Декларативный, а не процедурный подход.

Все вышеперечисленные инструменты были созданы для настройки сервера, что означает, что их основная цель – установить программное обеспечение на уже существующие серверы и управлять им. Terraform больше концентрируется на подготовке серверов, а развертывание контейнеров программного обеспечения остается за Docker или Ansible.

Terraform использует подход неизменяемой инфраструктуры, при котором каждое новое обновление любого параметра создает отдельный моментальный снимок конфигурации, что означает развертывание нового сервера и удаление старого. Таким образом, обновление среды разработки проходит легко и полностью защищено от ошибок, а возврат к одной из предыдущих конфигураций так же прост, как выбор моментального снимка конфигурации и подготовка новой среды в соответствии с ним [15].

В ходе работы была реализована система для автоматически развер-

тывания и конфигурации при помощи Ansible для настройки серверов под управлением Linux и серверов под управлением Ubuntu. Данная система настраивает 5 серверов под различные нужды: Consul сервер, два сервера для контейнеров Docker и с приложением для тестирования, сервер с Jenkins для автоматической сборки проекта, сервер с базой данных CockroachDB. После запуска Ansible настроит сервера, и они будут готовы для использования.

На рисунке 1 изображена диаграмма развертывания системы. На машине под названием Ansible server запускаются Terraform для создания всей инфраструктуры: создания Virtual Private Cloud. запуском виртуальных машин. созданием inventory файла для Ansible. После завершения создания инфраструктуры, запускается Ansible и конфигурирует сервера в соответствие с их ролью.

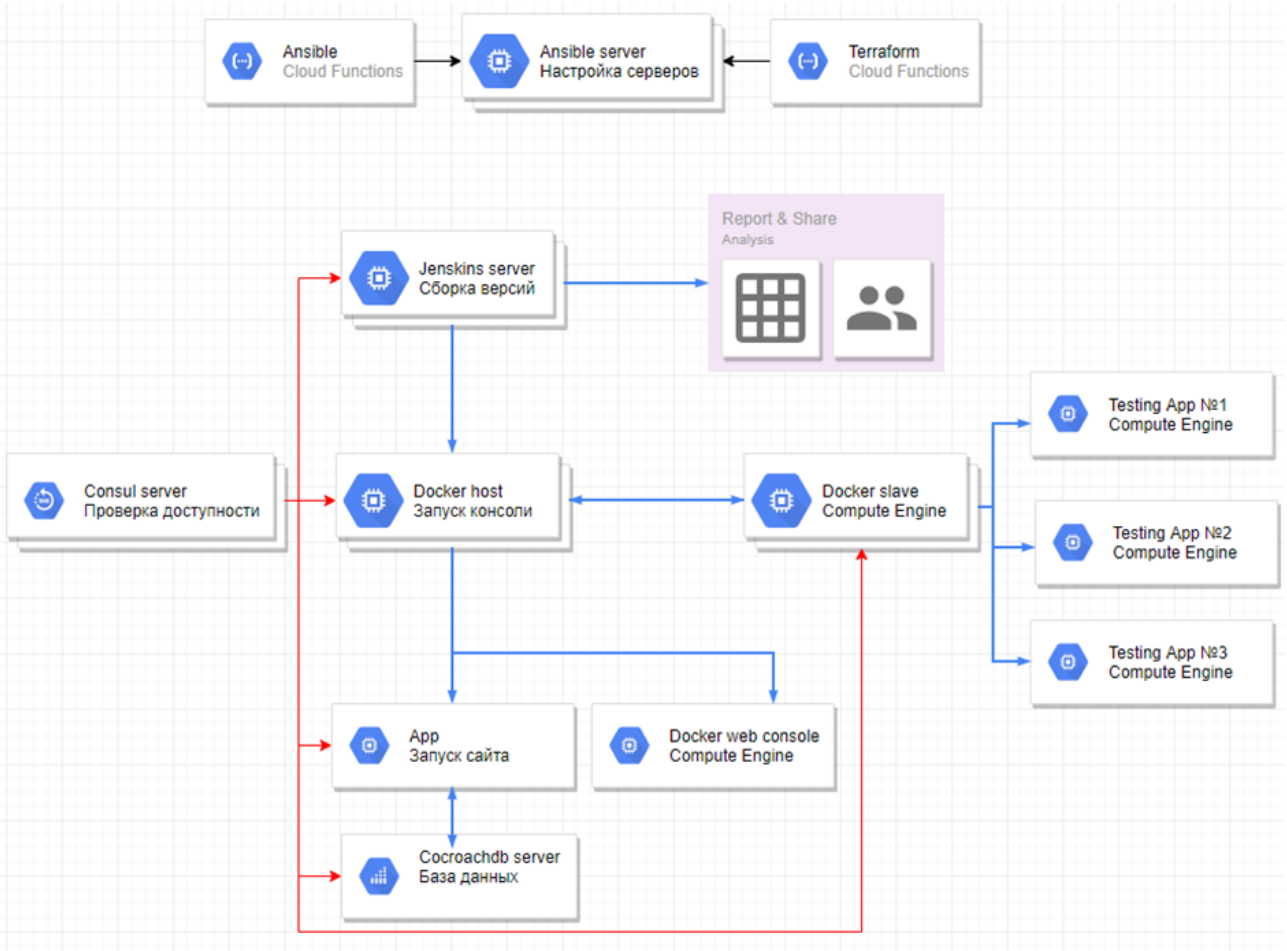

Рисунок 1 – Схема конфигурации и работы системы.

ЗАКЛЮЧЕНИЕ

В работе выполнен анализ существующих систем повышения квалификации сотрудников ИТ сферы таких как: Oracle Academy и Сетевая Академия CISCO. Мною было проведено исследование, направленное на выявления различия между виртуализацией при помощи виртуальных машин и контейнеров, было проведено сравнение средств управления конфигурации и средств service discovery.

Создана система автоматического развертывания тестирующей среды по проверке практических навыков удаленной работы с Linux серверами. Рассмотрены: особенности использование языка Go и архитектура авто масштабируемой распределённой SQL-базы данных *Cockroach DB*; процесс непрерывной интеграции и инструменты для непрерывной интеграции: Jenkins, TeamCity и Bamboo; особенности и недостатки систем service discovey; особенности и недостатки инструментов для управления конфигурацией.

Разработаны: модуль тестирования; модуль проверки практических заданий; модуль доступа к контейнерам для системы автоматического развертывания тестирующей среды по проверке практических навыков удаленной работы с Linux серверами на языке Go; проект Ansible для создания автоматического развертывания системы; проект Terraform для создания инфраструктуры в облаке; использован Consul для обнаружения и для отслеживания запущенных контейнеров.

В ходе работы было подтверждено, что виртуализации при помощи виртуальным машин и контейнеров – не соперники, поскольку решают разные задачи. Контейнеры эффективны там, где нужно исполнять как можно больше приложений на как можно меньшем числе серверов при относительно небольшом разнообразии ОС. Виртуальные машины лучше использовать, когда приходится иметь дело сразу со многими разными ОС. Результаты тестирования подтверждают общий тезис: расходы виртуализации при помощи контейнеров в целом существенно ниже, чем при помощи виртуальных машин. Контейнеры лучше используют память и при передаче данных в сеть накладные расходы контейнера по пропускной способности ниже чем у виртуальной машины.

Разработанная в магистерской работе система автоматического развертывания тестирующей среды по проверке практических навыков удаленной

работы с Linux серверами позволяет повысить эффективность выявления наличия необходимых навыков и позволит приобрести новые навыки при помощи практических и теоретических заданий.

Результаты работы вошли в статью «Автоматическая система тестирования для проверки теоретических и практических навыков у студентов» представленную на Всероссийской научно-практической конференции «Образование. Технологии. Качество» 2019 года и в статью «Возможности облачных платформ для студентов» представленную на IX Всероссийской научно-практической конференции «Информационные технологии в образовании» 2017 года [16, 17].

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Сетевая Академия CISCO [Электронный ресурс]. — URL: https://www.netacad.com/ru/about-networking-academy/curriculum/ (Дата обращения 24.11.2017).

2 Oracle Academy [Электронный ресурс]. — URL: https://academy.oracle.com/ru/training-workshops.html (Дата обращения 24.11.2017).

3 Moodle is a Learning Platform or course management system [Электронный ресурс]. — URL: https://moodle.org/?lang=ru (Дата обращения 24.11.2017).

4 *Hall, D.* Ansible Configuration Management / D. Hall. — Packt Publishing, 2013. — Das einzige Buch zu Ansible; nur 92 Seiten, lohnt vermutlich nicht.

5 Документация Ansible. [Электронный ресурс]. — URL: http://docs.ansible.com/ (Дата обращения 28.05.2018.).

6 Zetcd от CoreOS: Заменяя ZooKeeper на хранилище etcd. [Электронный ресурс]. — URL: https://habr.com/company/flant/blog/329224/ (Дата обращения 28.05.2018.).

7 Twelve-Factor Applications with Consul. [Электронный ресурс]. — URL: http://www.hashicorp.com/blog/twelve-factor-consul.html (Дата обращения 28.05.2018.).

8 CockroachDB - the open source, cloud-native SQL database [Электронный ресурс]. — URL: https://github.com/cockroachdb/cockroach/ (Дата обращения 28.05.2018.).

9 Start a Node CockroachDB [Электронный ресурс]. — URL: https://www.cockroachlabs.com/docs/start-a-node.html (Дата обращения 28.05.2018.).

10 ACID [Электронный ресурс]. — URL: https://en.wikipedia.org/wiki/ACID (Дата обращения 28.05.2018.).

11 Learn CockroachDB SQL [Электронный ресурс]. — URL: https://www.cockroachlabs.com/docs/learn-cockroachdb-sql.html (Дата обращения 28.05.2018.).

12 *Smart, J. F.* Jenkins: The definitive guide / J. F. Smart. — O'Reilly Media, Inc., 2011.

13 *Muli, J.* Jenkins fundamentals: Accelerate deliverables, manage builds, and automate pipelines with jenkins / J. Muli, A. Okoth. Packt Publishing. — 2018.

14 *Soni, M.* Jenkins essentials / M. Soni. Packt Publishing. — 2017.

15 *Brikman, Y.* Terraform: Up and Running: Writing Infrastructure As Code / Y. Brikman. — O'Reilly Media, Incorporated, 2019.

16 *Полинский, А. В.* Возможности облачных платформ для студентов / А. В. Полинский // *IX Всероссийская научно-практическая конференция "Информационные технологии в образовании".* — 2017. — Pp. 277–281.

17 *Полинский, А. В.* Автоматическая система тестирования для проверки теоретических и практических навыков у студентов / А. В. Полинский // *Всероссийская научно-практическая конференция "Образование. Технологии. Качество".* — 2019.