Управление контейнерами при построении распределенных систем с микросервисной архитектурой

А. С. Бондаренко, К.С. Зайцев

***Аннотация*. Целью настоящей работы является исследование возможности и способов применимости систем управления контейнерами для построения распределенных микросервисных архитектур, так как сегодня программисты и архитекторы всё чаще отдают предпочтение именно такой архитектуре. Причиной этого является широкое распространение облачных технологий, которые с каждым годом становятся все проще в настройке и управлении. Для достижения цели исследования в статье рассмотрены основные особенности микросервисной архитектуры, и проведена оценка применимости системы управления контейнерами на основе инструментов Kebernetes/ Openshift при построении распределенных архитектур. Так же был проведен эксперимент с демонстрацией эффективного управления топологией сети как одного из важнейших требований к разрабатываемым системам такого типа. Результаты применения предложенных инструментов подтвердили правильность использования систем управления и оркестрации контейнерами для построения микросервисов и продемонстрировали эффективность такого управления.**

***Ключевые слова* – микросервис, контейнер, микросервисная архитектура, облачные вычисления, программное обеспечение.**

I. ВВЕДЕНИЕ

Понятие облака для современного человека уже давно ассоциируется с различными интернет-сервисами, которые предоставляют различные услуги, помогая человеку закрыть различные информационные потребности. Облачные вычисления представляют собой различные вычислительные услуги, такие как серверы, хранилища, базы данных, сети, программное обеспечение, аналитику через Интернет-облако без активного прямого управления со стороны потребителя этих услуг. Вместо того, чтобы владеть собственной вычислительной инфраструктурой или центрами обработки данных, компании могут арендовать доступ ко всему вышеперечисленному у поставщика облачных услуг.

Одним из преимуществ использования является то, что компании могут избежать первоначальных затрат и сложностей, связанных с владением и обслуживанием собственной ИТ-инфраструктуры, и вместо этого просто платить за то, что они используют, по требованию.

Одним из преимуществ использования является то, что компании могут избежать первоначальных затрат и сложностей, связанных с владением и обслуживанием собственной ИТ-инфраструктуры, и вместо этого просто платить за то, что они используют, по требованию.

В свою очередь, поставщики услуг облачных вычислений могут извлечь выгоду из значительной экономии за счет масштаба, предоставляя одни и те же услуги широкому кругу клиентов. Центральное место в построении облачных решений занимают контейнеры.

В процессе проектирования современных информационных систем основная задача заключается в управлении сложностью системы. Чтобы система соответствовала классическим критериям качества ПО она должна быть хорошо организована, то есть, иметь хорошую архитектуру.

Облачная архитектура является ключевым элементом построения облачных решений. Она соединяет все компоненты и технологии, необходимые для облачных вычислений, предоставляет возможность изменять и расширять функционал системы без изменения уже существующих модулей. Другими словами, новые функции реализуются через написание нового кода, а существующая кодовая база не будет изменена [1].

Миграция в облако несет множество бизнес-преимуществ по сравнению с локальными средами, от повышенной гибкости и масштабируемости до экономической эффективности. Хотя многие организации могут начать с подхода «подъем и перенос», при котором локальные приложения перемещаются с минимальными изменениями, в конечном итоге потребуется создавать и развертывать приложения в соответствии с потребностями облачных сред. Облачная архитектура определяет, как компоненты интегрируются, чтобы вы могли объединять, совместно использовать и масштабировать ресурсы по сети.

В сфере корпоративных приложений возникла необходимость интеграции и взаимодействия большого количества информационных систем предприятия, или нескольких партнерских предприятий, которая оказывает существенное влияние на используемые программные архитектуры.

В разработке программного обеспечения микросервисная архитектура представляет собой вариант структурного стиля сервис-ориентированной архитектуры. Это архитектурный шаблон, в котором приложение представляет собой набор

слабосвязанных мелкомодульных сервисов, взаимодействующих с помощью облегченных протоколов. Одна из его целей заключается в том, чтобы команды могли разрабатывать и развертывать свои сервисы независимо от других. Монолитное программное обеспечение спроектировано так, чтобы быть автономным, в котором компоненты или функции программы тесно связаны, а не слабо связаны, как в микросервисах. В монолитной архитектуре каждый компонент и связанные с ним компоненты должны присутствовать для выполнения или компиляции кода. Монолитные приложения являются одноуровневыми, что означает объединение нескольких компонентов в одно большое приложение. Следовательно, они, как правило, имеют большие кодовые базы, управлять которыми со временем может быть чересчур сложно. В отличии от этого, в микросервисах сокращаются зависимости в кодовой базе, что позволяет разработчикам развивать свои сервисы с меньшими ограничениями со стороны пользователей и скрывать от пользователей дополнительные сложности. Как следствие, компании могут разрабатывать программное обеспечение постоянно увеличивая скорость вывода в промышленную эксплуатацию и размер системы. Эти преимущества достигаются за счет поддержания независимости и разделения микросервисов. Интерфейсы должны быть тщательно спроектированы и рассматриваться как общедоступный API. Одним из используемых методов является наличие нескольких интерфейсов в одной и той же службе или нескольких версий одной и той же службы, чтобы не нарушать работу существующих пользователей [2].

II. ОСОБЕННОСТИ ПРИМЕНЕНИЯ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ

Архитектура микросервисов (часто сокращается до микросервисов) является одним из видов сервисориентированной архитектуры и позволяют разделить большое приложение на более мелкие независимые части, каждая из которых имеет свою сферу ответственности. Для обслуживания одного запроса пользователя приложение на основе микросервисов может вызывать множество внутренних микросервисов для составления своего ответа (рис. 1) [1]. Основными характеристиками данной архитектуры являются множество компонентов, бизнес-подход (изоляция по бизнес функционалу), простота маршрутизации, децентрализованность отказоустойчивость и эволюционность [2].

В процессе построения решений на основе микросервисной архитектуры необходимо учитывать требования, с которыми могут столкнуться архитекторы и программные инженеры в процессе разработки. Основными требованиями применения микросервисной архитектуры являются [3]:

1. Планирование структуры данных;

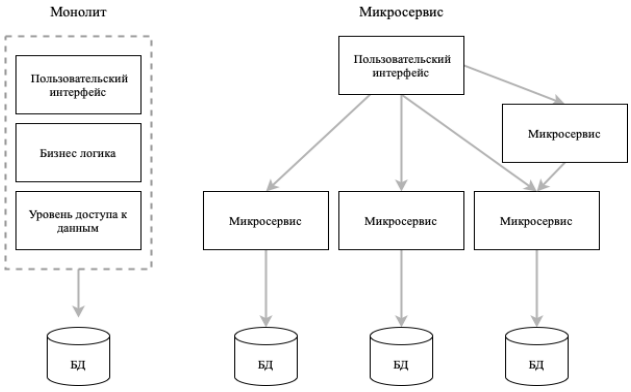

Рис.1. Примеры монолитной и мироксервисной архитектуры

2. Понимание необходимости использования именно микросервисной архитектуры;
3. Планирование процессов мониторинга и логирования, тестовых сценариев и автотестов для проверки работоспособности сервисов;
4. Подготовка среды быстрого развертывания;
5. Понимание того, что разработка микросервисов это не быстрая задача, и над ней стоит весьма долго и кропотливо работать, начиная с мелких компонентов.

Создание системы на основе микросервисов достаточно трудный процесс, в ходе которого надо учитывать различные факторы, возникающие из-за слабой связности модулей между сбой и возможности отказов в них. Если в ходе создания программного решения на основе микросервисов учесть все проблемные места, можно получить отказоустойчивое и надежное решение, имеющее в дальнейшем возможность легкой доработки или исправления ошибок, в случае их появления.

III. КОТЕЙНЕРИЗАЦИЯ ПРИЛОЖЕНИЙ И АРХИТЕКТУРА МИКРОСЕРВИСОВ

Применение контейнеров – один из наиболее динамично развивающихся сегментов ИТ-рынка, ориентированный на активное корпоративное использование облачных услуг. По оценкам Forrester Research, сегодня контейнеры уже получили признание у 41% корпоративного рынка. Согласно прогнозу компании 451 Research, мировой объем продаж контейнеров в ближайшее время будет продолжен. Если в 2016 г. он оценивался в размере 762 миллионов долларов [1], в 2018 в размере 1,2 миллиарда [4], к 2022 цифра выросла до 2.09 миллиарда и в перспективе объем должен достигнуть уровня 5,6 миллиардов долларов к 2027 году, что соответствует среднегодовому темпу роста около 22,7% [5].

Контейнеризация — это форма виртуализации, при которой приложения запускаются в изолированных пользовательских пространствах, называемых контейнерами, при использовании одной и той же общей операционной системы (рис.2). Одним из преимуществ контейнеризации является то, что контейнер представляет собой полностью упакованную и переносимую вычислительную среду. Все, что нужно приложению для запуска — его

двоичные файлы, библиотеки, файлы конфигурации и зависимости — инкапсулировано и изолировано в его контейнере.

Сам контейнер абстрагирован от хостовой операционной системы, с ограниченным доступом к базовым ресурсам — так же, как виртуальная машина. В результате контейнеризированное приложение можно запускать в различных типах инфраструктуры — на «голом железе», в виртуальных машинах и в облаке — без необходимости его рефакторинга для каждой среды. В Unix-подобных операционных системах контейнеризация реализуется через несколько механизмов: cgroups - функции для изоляции и контроля использования ресурсов (например, сколько ЦП и ОЗУ, потоков может использовать данный процесс) в ядре Linux. Механизм namespaces позволяет изолировать пространства имен компонентов, таких как таблицы маршрутизации и файловые системы. Контейнер Linux может монтировать файловую систему, запускать команды от имени root и получать IP-адрес. Он выполняет эти действия в своем личном пользовательском пространстве.

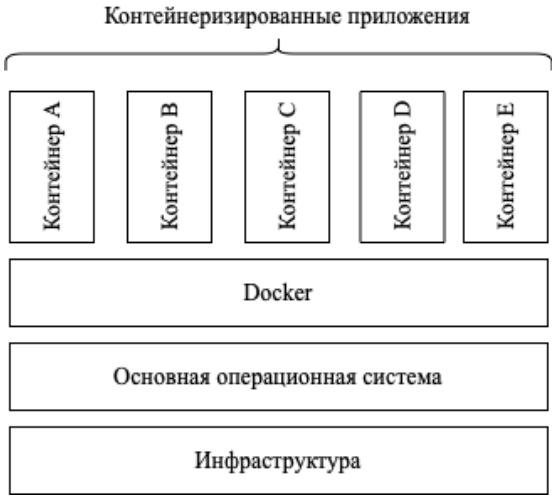

Рис.2. Место контейнера в архитектуре виртуализации

Docker — одна из самых популярных платформ контейнеризации с открытым исходным кодом. Она предлагает высокий уровень безопасности, простое управление и повышенную скорость развертывания. Это в значительной степени облегчает жизнь разработчикам, делая процесс упаковки приложений в контейнеры беспроблемным. Таким образом, процесс становится автоматизированным и быстрым. К тому же она позволяет управлять многочисленными контейнерами на одном хосте.

Кроме того, большая часть популярности Docker заключается в простоте его использования. Благодаря множеству ресурсов и широкому комьюнити пользователей, Docker можно быстро освоить, изучая доступные материалы по теме создания и управления контейнерами.

На текущий момент, технология контейнеризации является одним из самых мощных и современных методов виртуализации. [2, 3].

Отличия технологий контейнейризации от использования традиционной виртуальной машины приведены в таблице 1.

Таблица 1. Сравнение инструментов виртуализации

Технология / Критерий	Контейнер	Виртуальная машина
Определение	Изолированная группа процессов, управляемая общим ядром	Полная ОС, которая совместно использует хост-оборудование через гипервизор.
Тип ОС	Одно ядро ОС, но разные дистрибутивы	Различные независимые ОС
Уровень изоляции	Механизмы namespaces и cgroups	Полная изоляция на уровне ОС
Размер	Образы измеряются в мегабайтах, плюс размер пользовательского приложения	Образы измеряются в гигабайтах, плюс размер пользовательского приложения
Жизненный цикл	Запускается непосредственно в ядре без процесса загрузки, часто недолговечен.	Имеет процесс загрузки и, как правило, долгоживущий

Компания Datadog, занимающаяся решениями по безопасности и мониторингу для облачных платформ, имеет большое количество данных об устройстве облачных инфраструктур в компаниях по всему миру. Она собрала статистику по использованию Docker и смежных технологий среди своих клиентов, которая представляет тысячи компаний и 1.5 миллиарда контейнеров [6, 7]. Список анализируемых компаний-клиентов охватывает большинство отраслей и весь спектр: от стартапов до компаний из списка Fortune 100.

Согласно исследованиям компании ежегодно в период с 2019 по 2022 годы количество клиентов Datadog использующих Docker росло на 38-40%. Больше 60% компаний с количеством серверов 500 и выше используют Docker для решения своих задач. С 2019 года начался рост интереса к Docker со стороны компаний среднего размера (количество серверов от 100 до 500). Доля пользователей Docker среди них приблизилось к показателю как у крупных организаций в 2019 [8], а с 2022 года превзошло их [7].

Дальнейшая эволюция контейнеризированных технологий привела к использованию контейнерами распределённых вычислений облаченных в микросервисную архитектуру [10]. Больше половины компаний-пользователей Docker, используют системы оркестрации.

В 2019 почти 40% пользователей Docker применяли различные системы оркестрации [6], а в 2022 году

именно Kubernetes стал популярен как никогда. Сегодня почти половина организаций использует Kubernetes для развертывания контейнеров и управления ими в растущей экосистеме. Такие инструменты, как Amazon Elastic Kubernetes Services Blueprints и Amazon EKS Anywhere упрощают для команд запуск кластеров в облаке и локально. Роль Kubernetes в качестве платформы – предоставить унифицированные, согласованные и удобные для разработчиков средства развертывания приложений в различных масштабах. [11, 12].

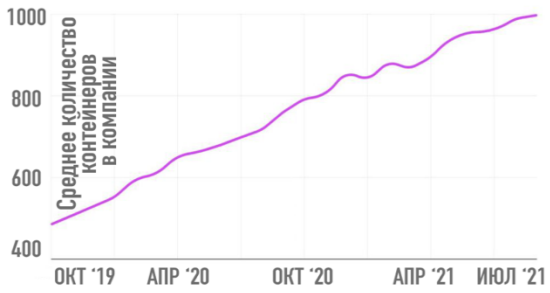

Рис.3. Среднее количество контейнеров в компании удвоилось

Анализируя возрастающую тенденцию внедрения Docker-контейнеров [9, 12] в крупных корпорациях [7, 13], можно сделать вывод об актуальности темы применения контейнеров с использованием облачной архитектуры для построения информационных систем.

IV. ПОСТРОЕНИЕ АРХИТЕКТУРЫ В СИСТЕМЕ УПРАВЛЕНИЯ КОНТЕЙНЕРАМИ

Для построения распределенной системы с микросервисной архитектурой в качестве эксперимента по предмету исследования было решено использовать четыре приложения, написанных на языке Java, представляющих собой четыре отдельных микросервиса. В качестве системы управления контейнерами выбран инструмент OpenShift Online, позволяющий загружать пользовательские образы и строить микросервисную архитектуру.

Клиентский сервис client обращается к серверу по сущности service openshift. За этим сервером-сервисом скрываются 3 версии одного сервиса v1, v2 и v3. Клиент инициирует запрос после обращения к нему (из вне) по REST протоколу, и далее вызывает сервер. По умолчанию в openshift маршрутизация происходит по алгоритму round robin, т. е. перебор серверов внутри одной группы по круговому циклу.

Ожидаемое поведение в текущей конструкции построенной микросервисной архитектуры такое, что при множественных вызовах от клиента к серверу, нагрузка будет равномерно распределена на всех серверах в топологии сети, в нашем случае по трем сервисам.

В рамках эксперимента было инициировано 100 вызовов от клиента к серверу. Все вызовы имели код HTTP ответа 200. Это означает что 100% вызовов завершились успешно.

Для наглядного представления топологии построенной микросервисной архитектуры, а также сетевого взаимодействия сервисов был использован инструмент Kiali, представленный на рисунке 4.

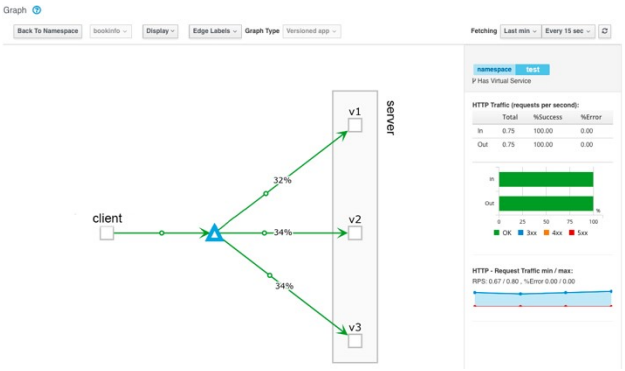

Рис.4. Сетевое взаимодействие компонентов построенной архитектуры.

По полученной схеме видно, что 32% трафика пошло на первую версию, и по 34% на вторую и третью. А значит алгоритм round robin работает, и мы имеем систему с распределенной нагрузкой по серверам (сервисам).

Теперь положим, что сервер v2 перестал отвечать на запросы, выведен из строя. Понаблюдаем как в это случае поведет себя построенная архитектура. После эмуляции вывода одного сервера из строя, Openshift определяет, что сервис больше не доступен, и оперативно выводит его из топологии сети, чтобы запросы от клиента не маршрутизировались на недоступный сервер. На этом этапе были инициированы дополнительные 100 запросов, после чего получены другие результаты (рис. 5).

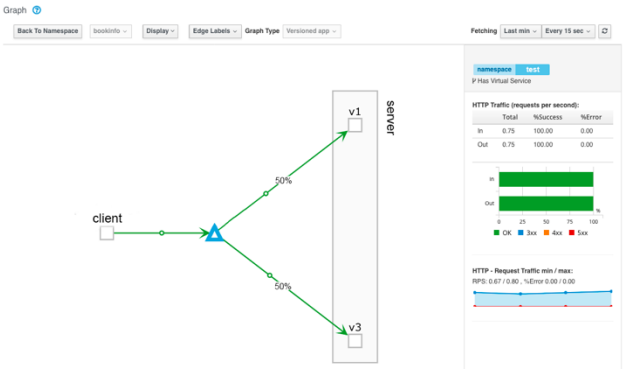

Рис.5. Сетевое взаимодействие компонентов после отключения сервера v2.

Запросы распределились равномерно между серверами v1 и v2 и успешно ответили кодом 200.

В этом эксперименте была продемонстрирована и исследована оркестрация контейнеров, успешно выполняемая системой управления OpenShift, в котором были развернуты созданные микросервисы.

Как было отмечено, OpenShift с установленным инструментом Kiali позволяет легко наблюдать построенную микросервисную архитектуру и анализировать запросы, передаваемые между сервисами.

Исследуемый пример хорошо показал, как система управления контейнерами может реализовать такие

базовые принципы микросервисов как обеспечение обнаружения сервисов (service discovery) и канареечные релизы, т.е. снижение риска внедрения новой версии программного обеспечения в «промышленную среду». Проблемы управления инфраструктурой в больших масштабах с успехом решает специальный журнал событий, в который серверы и компоненты инфраструктуры сообщают о своих адресах и состоянии, как только они появляются в кластере. Любые системы, которым нужно знать обо всех серверах, в курсе, что всё это можно узнать в этом журнале при наличии соответствующих прав доступа. Такая система называется службой обнаружения (service discovery). В нашем примере ее исполняет OpenShift.

С помощью этого же механизма решаются вопросы маршрутизации при использовании подхода канареечных релизов, когда путем плавного развертывания изменений для небольшой группы пользователей, значительно снижается риск внедрения новой версии программного обеспечения в промышленную среду.

Если нам необходимо запустить трафик извне в наш кластер, Kubernetes предоставляет и такую возможность. Для этого необходимо использовать сущность Kubernetes / Openshift Ingress, ведь сегодня более 35% организаций для управления внешним доступом к объектам в проекте используют Ingress, который стал общедоступным с момента выпуска Kubernetes версии 1.19 в августе 2020 года. [7] Многие первые пользователи Kubernetes использовали облачные балансировщики нагрузки для маршрутизации трафика к своим сервисам. Но Ingress часто более экономичен, и его внедрение неуклонно растет с момента его выпуска. Kubernetes Gateway API, бета-версия которого вышла в июле 2022 года, — это следующий шаг в эволюции управления сетью для контейнеров.

V. ДИСКУССИЯ ПО ТЕМЕ ИССЛЕДОВАНИЙ

На сегодняшний момент темы запуска крупных микросервисных архитектур широко обсуждается научным сообществом и прикладными инженерами по всему миру. В статье авторы широко освещают использование контейнеризации и подчеркивают важность использования подхода на основе микросервисов при построении систем для облачных вычислений.

В настоящее время технология Docker де-факто стала стандартом для контейнеризации приложений, о чем подчеркивает авторы публикаций [8, 12]. Кроме того, при использовании технологий в крупных системах реального времени никак нельзя обойтись без применения программного обеспечения оркестрации для управления и синхронизацией систем, состоящих из множества контейнеров, о чем докладывают авторы в отчете [11] и статье [14].

Большое распространение получили системы оркестрации на основе Kubernetes и Openshift, что подробно отражено в публикациях [14, 15].

На данный момент все описанные выше технологии и системы активно развиваются, как сообществом, так и вендорами, выпускающие enterprise версии, поэтому в ближайшее время количество исследований в данной тематике будет только увеличиваться, возможно изменяя или дополняя существующие факты, принятые на текущий момент. Миграция приложений в плоскость горизонтального масштабирования так же ускоряет процесс развития технологий и подходов, используемых при построении микросервисной архитектуры.

VI. ЗАКЛЮЧЕНИЕ

В результате изучения различных источников, мы пришли к выводу, что системы управления контейнерами играют важную роль при построении облачных распределённых систем в крупных корпорациях и внедрение микросервисной архитектуры является наиболее современным подходом.

Проведенный эксперимент показал, что система управления контейнерами (в исследуемом случае Openshift) позволяет строить распределенные микросервисные архитектуры, поскольку инструменты, предоставляемые технологией, дают возможность реализовать базовые принципы микросервисной архитектуры. Нам удалось распараллелить классический клиент-серверный подход, где каждый компонент является изолированным от других компонентов, имеет возможность быть перемещенным из одного пространства в другое, использует собственное место для хранения данных, отделенное от других микросервисов. Механизм service discovery представлен по типу черного ящика (а значит облегчающий установку и развертывания сервисов, без погружения в принципы его работу) позволяет легко обнаруживать сервисы для дальнейшего их взаимодействия и поддерживать постоянно актуальную топологию сети.

В процессе проведения эксперимента был сделан вывод о том, что управление доступом в кластерах улучшается с применением новых инструментов, но все же пока остается не простой задачей. В Openshift мы использовали управление доступом на основе ролей, чтобы разрешить субъектам (пользователям, группам или учетным записям служб) получать доступ или изменять ресурсы в кластере. В эксперименте мы столкнулись с тем, что не всегда достаточно базовых прав для полноценной работы построенной архитектуры. В нашем случае это не несет особых рисков, однако большие корпорации могут столкнуться с трудностями использования такого подхода, ведь в соответствии с рекомендациями по безопасности субъекты должны иметь только необходимые разрешения, и администраторы должны соблюдать осторожность при предоставлении привилегий, которые связаны с рисками эскалации. К ним относятся разрешения, которые позволяют субъектам перечислять все секреты или создавать рабочие нагрузки, сертификаты или запросы токенов, которые могут позволить им изменять свои собственные привилегии.

Возможно, этот вызов могут принять алгоритмы, созданные с применением различных методов ИИ, например с использованием многоагентных моделей для автоматизации управления и развертывания контейнеризированных приложений, что является сегодня весьма актуальным.

Независимо от того, что на текущий момент существует большое количество различных научных публикаций по теме упаковки приложений в контейнеры, как современного метода виртуализации, тема применения методов искусственного интеллекта для управления кластерами микросервисов-контейнеров, практически нет. По мнению авторов, такой эффект вызван новизной микросервисного подхода и постоянной эволюцией систем оркестрации контейнеров, поэтому тема является актуальной и будет активно развиваться в ближайшем будущем.

БЛАГОДАРНОСТИ

Авторы выражают благодарность Высшей инжиниринговой школе НИЯУ МИФИ за помощь в возможности опубликовать результаты выполненной работы.

БИБЛИОГРАФИЯ

1. Новиков И. Kubernetes, Docker Swarm и Mesos: лучшее для оркестрации контейнеров [Электронный ресурс]: https://www.itweek.ru/infrastructure/article/detail.php?ID=196419 (Дата обращения: 13.06.2022).
2. Бедняк С.Г., Симакова В.Е. Информационные технологии. Виртуализация платформ и ресурсов //Актуальные направления научных исследований XXI века: теория и практика. – 2015. – Т. 3. – No. 7-3. – с. 346-349.
3. Шурупов Д. Какие известные компании используют Docker в production и для чего? [Электронный ресурс]: https://habr.com/ru/company/flant/blog/326784/ (Дата обращения: 13.06.2022).
4. Application container market [Электронный ресурс]: https://www.marketsandmarkets.com/Market-Reports/application-container-market-182079587.html (Дата обращения: 30.12.2022)
5. Containers as a Service Market by Service Type [Электронный ресурс]: https://www.researchandmarkets.com/reports/4402288/containers-as-a-service-market-by-service-type?utm_source=GNOM&utm_medium=PressRelease&utm_code=xnrj7f&utm_campaign=1791213+-+Global+Containers+as+a+Service+Market+Report+2022+to+2027%3a+Benefits+of+Cost-Effectiveness+and+Increased+Productivity+Drive+Growth&utm_exec=jamu273prd (Дата обращения: 30.12.2022)
6. Kumar R., Charu S. An importance of using virtualization technology in cloud computing //Global Journal of Computers & Technology. – 2015. – Т. 1. – №. 2.
7. 9 Insights on Real-World Container Use | Datadog [Электронный ресурс]: https://www.datadoghq.com/container-report/?utm_source=organic&utm_medium=display&utm_campaign=dg-organic-websites-ww-corpsite-announcement-report-container2022 (Дата обращения: 30.12.2022)
8. 8 surprising facts about real Docker adoption [Электронный ресурс]: https://www.datadoghq.com/docker-adoption/ (Дата обращения: 13.06.2022).
9. ERN: Processing Petabytes of Data More Efficiently with Kubernetes [Электронный ресурс]: https://kubernetes.io/case-studies/cern/ (Дата обращения: 30.12.2022)
10. Carter E. 2018 Docker usage report. [Электронный ресурс]: https://sysdig.com/blog/2018-docker-usage-report/ (Дата обращения: 13.01.2022).
11. Carter E. Sysdig 2019 Container Usage Report: New Kubernetes and security insights [Электронный ресурс]: https://sysdig.com/blog/sysdig-2019-container-usage-report/ (Дата обращения: 13.06.2022).
12. Morgan T. Inside Ebay's shift to Kubernetes and containers atop Openstack [Электронный ресурс]: https://www.nextplatform.com/2015/11/12/inside-ebays-shift-to-kubernetes-and-containers-atop-openstack/(Дата обращения: 13.06.2022).
13. Орлов Д. Как жить с Docker, или почему лучше с ним, чем без него? [Электронный ресурс]: https://habr.com/ru/post/250469/ (Дата обращения: 13.01.2022).
14. Oliveira C. et al. Evaluating raft in docker on kubernetes //International Conference on Systems Science. – Springer, Cham, 2016. – P. 123-130.
15. Lossent A., Wagner A., Rodriguez Peon A. IOP: PaaS for web applications with OpenShift Origin // J. Phys.: Conf. Ser. – 2017. – Т. 898. – P. 082037.

Статья получена 12.05.2023

Бондаренко Александр Сергеевич, Национальный исследовательский ядерный университет «МИФИ» (НИЯУ МИФИ), аспирант, sasha-bond-95@mail.ru

Зайцев Константин Сергеевич, Национальный исследовательский ядерный университет «МИФИ» (НИЯУ МИФИ), профессор, kszajtsev@mephi.ru

Using container management systems to build distributed cloud information systems with microservice architecture

A.S. Bondarenko, K.S. Zaytsev

Annotation **– The purpose of this work is to study the possibility and applicability of container management systems for building distributed systems with a microservice architecture, because nowadays programmers and architects are increasingly giving their preference to such an architecture. The reason is the widespread use of cloud technologies, which every year are becoming easier to set up and manage. To achieve the goal of the study, the authors considered the main features of the microservice architecture and evaluated the applicability of a container management system based on Kebernetes / Openshift for building distributed systems with such an architecture. An experiment was also conducted to demonstrate the effective management of the network topology as one of the most important requirements for the developed systems of this type. The results of applying the tools proposed by the authors showed the importance of using container management and orchestration systems for building microservices and demonstrated the convenience and efficiency of this management.**

Keywords **– distributed systems, containers, microservices, system architecture, cloud information system.**

REFERENCES

1. Novikov I. Kubernetes, Docker Swarm и Mesos: best for container orchestration. Avaliable at: https://www.itweek.ru/infrastructure/article/detail.php?ID=196419 (Date of access: 13.06.2022).
2. Bednyak S.G., Simakova V.E. Information Technology. Virtualization of Platforms and Resources // Actual directions of scientific research of the XXI century: theory and practice. – 2015. – Vol. 3. – No. 7-3. – P. 346-349.
3. Shurupov D. Which well-known companies use Docker in production and why? Avaliable at: https://habr.com/ru/company/flant/blog/326784/ (Date of access: 13.06.2022).
4. Application container market. Avaliable at: https://www.marketsandmarkets.com/Market-Reports/application-container-market-182079587.html (Date of access: 30.12.2022)
5. Containers as a Service Market by Service Type. Avaliable at: https://www.researchandmarkets.com/reports/4402288/containers-as-a-service-market-by-service-type?utm_source=GNOM&utm_medium=PressRelease&utm_code=xnrj7f&utm_campaign=1791213+-+Global+Containers+as+a+Service+Market+Report+2022+to+2027%3a+Benefits+of+Cost-Effectiveness+and+Increased+Productivity+Drive+Growth&utm_exec=jamu273prd (Date of access: 30.12.2022)
6. Kumar R., Charu S. An importance of using virtualization technology in cloud computing //Global Journal of Computers & Technology. – 2015. – Vol. 1. – No. 2.
7. 9 Insights on Real-World Container Use | Datadog Avaliable at: https://www.datadoghq.com/container-report/?utm_source=organic&utm_medium=display&utm_campaign=dg-organic-websites-ww-corpsite-announcement-report-container2022 (Date of access: 30.12.2022)
8. 8 surprising facts about real Docker adoption Avaliable at: https://www.datadoghq.com/docker-adoption/ (Date of access: 13.06.2022).
9. ERN: Processing Petabytes of Data More Efficiently with Kubernetes Avaliable at: https://kubernetes.io/case-studies/cern/ (Date of access: 30.12.2022)
10. Carter E. Sysdig 2019 Container Usage Report: New Kubernetes and security insights Avaliable at https://sysdig.com/blog/sysdig-2019-container-usage-report/ (Date of access: 13.06.2022).
11. Carter E. 2018 Docker usage report. Avaliable at: https://sysdig.com/blog/2018-docker-usage-report/ (Date of access: 13.01.2022).(Date of access: 13.06.2022).
12. Morgan T. Inside Ebay's shift to Kubernetes and containers atop Openstack Avaliable at: https://www.nextplatform.com/2015/11/12/inside-ebays-shift-to-kubernetes-and-containers-atop-openstack/(Date of access: 13.06.2022).
13. Orlov D. How to live with Docker, or why is it better with it than without it? Avaliable at: https://habr.com/ru/post/250469/ (Date of access: 13.01.2022).
14. Oliveira C. et al. Evaluating raft in docker on kubernetes //International Conference on Systems Science. – Springer, Cham, 2016. – P. 123-130.
15. Lossent A., Wagner A., Rodriguez Peon A. IOP: PaaS for web applications with OpenShift Origin // J. Phys.: Conf. Ser. – 2017. – Vol. 898. – P. 082037.