**УДК 004.421**

**Альмухаметов Амур Артурович,**

*студент направления подготовки «Разработка и управление в программных проектах»,*

*amur\_almukhametov@mail.ru*

*Уральский федеральный университет имени первого Президента России Б.Н. Ельцина,*

*РФ, г. Екатеринбург*

**Кислицын Евгений Витальевич,**

*руководитель образовательной программы*

ev.kislitsyn@urfu.ru

*Уральский федеральный университет имени первого Президента России Б.Н. Ельцина,*

*РФ, г. Екатеринбург*

*научный руководитель:* ***Кислицын Евгений Витальевич****,* *к.т.н., доцент кафедры информационных технологий и систем управления УрФУ ev.kislitsyn@urfu.ru*

*Уральский федеральный университет имени первого Президента России Б.Н. Ельцина,*

*РФ, г. Екатеринбург*

**СИСТЕМЫ ОРКЕСТРАЦИИ КАК ИНСТРУМЕНТЫ ДЛЯ УПРАВЛЕНИЯ КОНТЕЙНЕРАМИ**

***Аннотация:***

Доклад посвящён разбору технологии оркестрации для управления, развёртывания и масштабирования контейнеризированных приложений. Рассмотрено понятие микросервисов и микросервисной архитектуры. Так же в работе представлен краткий обзор возможностей и сравнение трёх самых часто используемых оркестраторов, таких как Kubernetes, Docker Swarm и Nomad.

***Ключевые слова****:* контейнеры, оркестрация, Docker, KUBERNETES, DOCKER SWARM, Nomad, микросервисы

**Введение**

Системы оркестрации являются специально созданными для управления контейнерами инструментами, из-за появления контейнеров появилась надобность в системах оркестрации. Рассмотрим оба определения. Контейнер [1] приложения — это легкая, изолированная среда выполнения, которая позволяет разрабатывать, тестировать и запускать приложения с их зависимостями в едином пакетном формате. Контейнеры помогают обеспечить согласованность между различными средами (разработка, тестирование, продакшен) благодаря изоляции каждой среды. В контексте изоляции можно провести аналогию с виртуальными машинами, но есть принципиальная разница. Виртуальные машины [2] работают под управлением гипервизора, который является операционной системой хоста, и требуют полной установки ОС для каждого экземпляра. В отличие от этого, каждый контейнер представляет собой экземпляр приложения, использующий ресурсы единственного ядра с достаточно легким уровнем изоляции. Это позволяет запускать значительно большее количество контейнеров на одной машине по сравнению с виртуальными машинами, при этом затрачивая меньше ресурсов. Для создания и работы с контейнерами используется Docker [3], предоставляющий следующие возможности:

- Создание образов. Способность консолидации программного обеспечения в повторно используемом формате, называемом образами.

- Управление временем исполнения. Возможность надежного и повторяемого запуска, приостановки, перезапуска или остановки упакованного программного обеспечения.

- Оркестрация и масштабирование. Процесс настройки, запуска и обновления множества контейнеров, которые составляют взаимодействующее приложение, выполняется через оркестрацию контейнеров, а для осуществления этого процесса используются оркестраторы.

Необходимость использования оркестраторов появилась в связи с развитием микросервисной архитектурой и микросервисов [4]. Микросервисы представляют собой технологию, которая получила широкое распространение как метод описания приложений в виде набора независимо развертываемых сервисов. Эти сервисы могут быть реализованы на различных языках программирования и использовать разные технологии для хранения данных. Каждый микросервис в кластере обладает собственной моделью и данными, он является автономным и обычно размещается в отдельном контейнере. Чем больше проект и количество сервисов, тем больше контейнеров необходимо управлять. Ответственность за управление этой задачей возлагается на оркестраторов, которые также контролируют жизненные циклы контейнеров микросервисных приложений.

В настоящее время существует более десятка платформ для оркестрации контейнеров [5], как декларативного, так и императивного подхода. В данной работе представлены результаты исследований, направленных на сравнение наиболее популярных из этих платформ с целью поиска оптимального инструмента для развертывания системы администрирования сети.

**Системы оркестрации контейнеров. Преимущества и недостатки**

При выборе инструмента для оркестрации контейнеров или управляемой службы оркестровки следует учитывать несколько ключевых аспектов:

- сетевые возможности;

- уровень доступности;

- легкость развертывания и обслуживания;

- масштабируемость;

- механизмы обнаружения служб;

- безопасность и соответствие стандартам;

- уровень поддержки (как со стороны сообщества, так и от предприятий);

- административные затраты.

Цель данной работы заключается в сопоставлении наиболее известных в индустрии продуктов. Всего их будет три: Kubernetes [6], Docker Swarm [7] и Nomad [8].

**Kubernetes (K8s)** — это система с открытым исходным кодом для автоматизации развертывания, масштабирования и управления контейнерными приложениями. Он группирует контейнеры, которые составляют приложение, в логические единицы для удобного управления и обнаружения. Kubernetes опирается на 15-летний опыт управления производственными нагрузками в Google в сочетании с лучшими в своем классе идеями и практиками сообщества. Использование данного программного средства представляется возможным в широком спектре различных сценариев, таких как:

− обнаружение сервисов и балансировка нагрузки - нет необходимости изменять приложение, чтобы использовать незнакомый механизм обнаружения сервисов. Kubernetes предоставляет модулю свои собственные IP-адреса и одно DNS-имя для набора модулей и может распределять нагрузку между ними;

− топология обслуживания - маршрутизация сервисного трафика на основе кластерной топологии;

− оркестрирование хранения – поддерживает различные системы хранения, будь то локальное хранилище, поставщик общедоступного облака, такой как GCP или AWS, или сетевая система хранения, такая как NFS, iSCSI, Gluster, Ceph, Cinder или Flocker;

− самовосстановление - Kubernetes перезапускает контейнеры, которые выходят из строя, заменяет и перепланирует контейнеры, когда узлы умирают, убивает контейнеры, которые не отвечают на вашу пользовательскую проверку работоспособности, и не объявляет их клиентам, пока они не будут готовы к обслуживанию;

− Автоматические развертывания и откаты - Kubernetes постепенно внедряет изменения в ваше приложение или его конфигурацию, одновременно отслеживая работоспособность приложения, чтобы убедиться, что оно не уничтожает все его экземпляры одновременно. Если что-то пойдет не так, Kubernetes отменит изменения;

− Управление ключами и конфигурацией - развертывайте и обновляйте секреты и конфигурацию приложения, не перестраивая образ и не раскрывая секреты в конфигурации стека;

− Автоматическая упаковка бинов. Происходит автоматическое размещает контейнеры на основе их требований к ресурсам и других ограничений, не жертвуя при этом доступностью. Комбинируйте критические и максимально эффективные рабочие нагрузки, чтобы повысить эффективность использования и сэкономить еще больше ресурсов;

− Пакетное исполнение. Kubernetes может управлять пакетными рабочими нагрузками и рабочими нагрузками CI, заменяя отказавшие контейнеры, если это необходимо;

− Горизонтальное масштабирование. Происходит упрощенное управление масштабированием вверх и вниз с помощью команд, пользовательского интерфейса или автоматически в зависимости от загрузки процессора.

**Docker Swarm mode** – абстракция Docker, которая привносит возможность объединения нескольких серверов для работы в режиме кластера. Docker Swarm объединяет множество Docker хостов в один виртуальный хост. И все клиенты, которые работают с Docker API могут продолжать работать с этим API, даже не подозревая, что обращение происходит не к одному хосту, а уже к кластеру.

Основные функции, предлагаемые оркестратором Docker Swarm:

− Управление кластерами, интегрированное с Docker Engine;

− Масштабирование (можно указать количество реплик сервиса);

− Безопасность по умолчанию – создаются самоподписанные сертификаты, можно создавать свои (например, каждый час будут генерироваться и обновляться сертификаты);

– Отказоустойчивость, благодаря нескольким manager;

– Балансировка нагрузки – настраивается Ingress;

– Service discovery – обнаружение сервисов происходит внутри оркестратора, без прямого участия разработчика;

– Можно менять топологию кластера прямо в runtime;

В кластере существует Manager node, worker node. Manager node находится в сети поддержания согласованности по алгоритму Raft. Raft – алгоритм, использующийся для решения задач нахождения консенсуса в сети ненадёжных вычислений. Для обеспечения отказоустойчивости в Docker Swarm обычно создают несколько серверов с ролью менеджера, и из них будет избираться лидер по алгоритму Raft, и эта нода теперь имеет роль Leader node.

Рабочие нагрузки запускаются на worker node. Соответственно, если у нас для разработки есть только одна нода, то она будет иметь сразу 3 роли: и manager, и leader, и worker.

Также Swarm работает как балансировщик нагрузки между несколькими рабочими узлами, равномерно распределяя запросы, приходящие с любой из сторон кластера. Например, если в кластере существует 3 сервера, на которых исполняется рабочая нагрузка, а определенное приложение есть только на 2, то если запрос придет на 3 сервер, Ingress с Load Balancer’ом перенаправит трафик на ноду с нужным приложением.

Плюсы использования кластера на базе **Docker Swarm**

– простота использования – если разработчик уже знаком с Docker и Docker Compose, то с помощью всего одной команды можно запустить свои сервисы в режиме кластера;

– нет необходимости в дополнительном ПО, так как Swarm режим уже включен по умолчанию в Docker Engine;

Минусы использования **Docker Swarm**:

– Docker Swarm дает ограниченные возможности разработки, например, в нем нет автоматического горизонтального масштабирования сервисов; – нужно самому создавать сети для работы приложений в разныx сетях;

**Nomad.** Nomad – это всего один двоичный файл, как для клиента, так и для сервера, и для него не требуется никаких внешних зависимостей. Nomad сочетает в себе легкий менеджер ресурсов и сложный планировщик в единую систему. По умолчанию Nomad является распределенным, высокодоступным, и функционально простым. Плюсы использования Nomad относительно Kubernetes:

– требует меньше ресурсов для работы control plane. У Kubernetes это обязательно 3 мастера 2 ядра /4 gb памяти, что подходит для большинства систем;

– не нужна плоская сетевая модель;

– возможность запуска различных артефактов, в том числе бинарных файлов, контейнеров;

– не нужно отдельно устанавливать minikube для создания окружения для разработки, требуется всего лишь одна команда: nomad agent –dev.

Минусы **Nomad** относительно **Kubernetes**:

– экосистема еще не так полноценна, поэтому нет определенного стиля разработки инструментов, и разработчику приходится учить совершенно разные инструменты для включения их в свою инфраструктуру;

– пока что мало готовых решений, как и что запускать;

– за безопасность ответственность несет администратор кластера;

– пока нет инструментов для запуска stateful приложений.

Nomad подходит для небольших команд, где архитектура взаимодействия приложения не столь обширна. Но со временем, если система разрастается, появляются новые заказчики, то скорее всего компания захочет переехать на Kubernetes.

Явным преимуществом над Kubernetes в том, что у него можно запускать различные артефакты, будь то контейнеры, Java файлы или бинарные исполняемые файлы. У Kubernetes есть поддержка различных движков контейнеров, таких как Docker, ContainerD, rkt (проект уже закрылся, но как поддержка старых проектов работает) и все приложения нужно будет упаковывать. А это бывает очень сложно сделать с legacy проектами, в которых не предполагалась возможность контейнеризации.

**Заключение**

Можно ли выделить самый лучший инструмент для оркестрации контейнеров? Однозначного ответа нет, каждый из рассмотренных инструментов подходит для своих задач:

Docker swarm – лучший вариант, когда используется Docker Compose и приложения без сложных связей между собой. Он требует меньше всего прилагаемых усилий для изучения.

Nomad – хороший вариант, в котором уже можно запускать production кластера с множеством сервисов. Здесь отсутствует общий стиль разработки инструментов, что делает изучение новых более сложным.

Kubernetes – это фреймворк, который имеет наибольшее сообщество разработчиков. Это один оркестратор, который «из коробки» уже имеет горизонтальное и вертикальное автомасштабирование подов. Также большим преимуществом является то, что Kubernetes стал первым проектом организации CNCF (Cloud Native Computing Foundation), которая позже сделала для него целую экосистему из проектов (Prometheus, Helm и т.д.).

**СПИСОК ЛИТЕРАТУРЫ**

1. Нанян, С. М. Виртуальные контейнеры Docker: назначение и особенности применения / С. М. Нанян, Т. Н. Ничушкина // Инженерный вестник. – 2015. – № 2. – С. 2. – EDN TVWZCF.
2. Савенков, И. В. Механизмы виртуализации в операционных системах / И. В. Савенков // Молодежь и наука: шаг к успеху : сборник научных статей 7-й Всероссийской научной конференции перспективных разработок молодых ученых, Курск, 21–22 марта 2024 года. – Курск: Закрытое акционерное общество "Университетская книга", 2024. – С. 108-111. – EDN AVRHPN.
3. Захарченок, В. Ф. Контейниризация и развертывание приложений с помощью Docker и Docker-compose / В. Ф. Захарченок, А. Н. Бизюк // Материалы докладов 56-й Международной научно-технической конференции преподавателей и студентов : В ДВУХ ТОМАХ, Витебск, 19 апреля 2023 года. Том 2. – Витебск: Витебский государственный технологический университет, 2023. – С. 80-82. – EDN AYMVDY.
4. Морозов, Н. Д. Введение в управление микросервисами / Н. Д. Морозов, Д. Д. Романов, К. В. Пугин // Современные проблемы лингвистики и методики преподавания русского языка в ВУЗе и школе. – 2022. – № 40. – С. 689-702. – EDN DMLLTW.
5. Системы оркестрации как средства управления контейнерами / М. А. Жижченко, Г. М. Михайлов, А. Н. Сальников, А. М. Чернецов // Научный сервис в сети Интернет. – 2022. – № 24. – С. 183-189. – DOI 10.20948/abrau-2022-34. – EDN DKHIRU.
6. Изучаем Kubernetes, совершенствуем Python, знакомимся с Kotlin // Системный администратор. – 2023. – № 7-8(248-249). – С. 94-98. – EDN VIZLHW.
7. Афанасьева, Ж. О. Применение технологии Docker и средства управления Docker-приложениями / Ж. О. Афанасьева // Гагаринские чтения - 2020 : Сборник тезисов докладов, Москва, 27 декабря 2019 года – 17 2020 года. – Москва: Московский авиационный институт (национальный исследовательский университет), 2020. – С. 279-280. – EDN BEKLPI.
8. Шкабара, Н. В. Передовые инструменты и технологии разработки программных приложений: взгляд на Kubernetes, Docker Swarm и Hashicorp Nomad / Н. В. Шкабара // Цифровые инструменты обеспечения устойчивого развития экономики и образования: новые подходы и актуальные проблемы : Сборник научных трудов III-й Национальной научно-практической конференции (с международным участием). В 2-х томах, Орел, 01 апреля 2024 года. – Орел: Российская академия народного хозяйства и государственной службы при Президенте РФ, 2024. – С. 263-267. – EDN GKQOYZ.

**Almukhametov A.A.**

**ORCHESTRATION SYSTEMS AS TOOLS FOR CONTAINER MANAGEMENT**

***Annotation:***

*The report is devoted to the analysis of orchestration technology for managing, deploying and scaling containerized applications. The concept of microservices and microservice architecture is considered. The work also provides a brief overview of the capabilities and a comparison of the three most commonly used orchestrators, such as Kubernetes, Docker Swarm and Nomad.*

***Keywords:*** *containers, orchestration, Docker, KUBERNETES, DOCKER SWARM, Nomad, microservices*