

Системы оркестрации, как средства управления контейнерами

М.А. Жижченко¹, Г.М. Михайлов¹, А. Н. Сальников^{1,3}, А.М. Чернецов^{1,2}

¹ *Вычислительный центр им. А.А. Дородницына ФИЦ ИУ РАН*

² *Национальный исследовательский университет «МЭИ»*

³ *Московский государственный университет имени М.В. Ломоносова*

Аннотация. В докладе рассматриваются технологии оркестрации для автоматизированного развертывания, управления и масштабирования контейнерных приложений. В работе представлен ряд лучших решений по созданию инструментов и сервисов оркестрации контейнеров, получивших широкое применение в производственной среде, таких как KUBERNETES, OPENSIFT, ROCHER, DOCER SWARM.

Ключевые слова: контейнеры, оркестрация, DevOps, Docker, KUBERNETES, OPENSIFT, ROCHER, DOCKER SWARM.

Orchestration systems as container management tools

M.A. Zhyzchenko¹, G.M. Mikhaylov¹, A.N. Salnikov^{1,3}, A.M. Chernetsov^{1,2}

¹ *Dorodnicyn Computing Centre FRC CSC RAS*

² *National Research University "MPEI"*

³ *Lomonosov Moscow State University*

Abstract. The report discusses orchestration technologies for automated deployment, management and scaling of containerized applications. The paper presents a number of the best solutions for creating container orchestration tools and services that are widely used in the production environment, such as KUBERNETES, OPENSIFT, ROCHER. DOCER SWARM.

Keywords: containers, orchestration, DevOps, Docker, KUBERNETES, OPENSIFT, ROCHER, DOCKER SWARM.

Введение

Название доклада включает в себя два взаимосвязанных определения: контейнеры и оркестрация, как средства управления контейнерами. Контейнер приложения - это экземпляр исполняемого программного обеспечения (ПО), который объединяет двоичный код приложения со всеми связанными файлами конфигурации, библиотеками, зависимостями и средой выполнения. Смысл и главное преимущество этой технологии в том, что контейнер абстрагирует приложение от ОС хоста, оставаясь автономным, как следствие, легко переносимом и способным работать на любой платформе [1]. Контейнеры могут запускаться, приостанавливаться, возобновляться и останавливаться, что приводит к представлению о сопоставимости с виртуальными машинами (ВМ). Однако есть принципиальная разница: ВМ работают под управлением некоей ОС хоста с названием гипервизор и требуют полной установки ОС для каждого экземпляра, в то время как каждый контейнер - это экземпляр ПО к имеющимся ресурсам отдельного ядра с достаточно тонким уровнем изоляции. Это позволяет запускать намного больше контейнеров в отдельной машине по сравнению с ВМ и при этом используют меньше ресурсов. Как описано в работах [2,3], задачи реализации основной идеи упаковки и запуска программного обеспечения в ограниченной среде, именуемой контейнерами, выполняются набором инструментов технологии Docker. На самом верхнем уровне Docker предоставляет следующие возможности:

- *Упаковку.* Способность паковать программное обеспечение в повторно используемом формате, именуемым *образами*.
- *Возможность управления временем исполнения.* Способность запуска, приостановки, перезапуска или останова упакованного программного обеспечения надёжным и повторяемым образом.
- *Оркестрация и масштабирование.* Задачи настройки, запуска и обновления множества контейнеров, которые составляют некое приложение во взаимодействии, выполняет процесс, именуемый как оркестрация контейнеров, а в качестве инструментов исполнения процесса выступают оркестраторы.

Микросервисы – еще одна технология, получившая широкое распространение как способ описания приложений в виде набора независимо развертываемых сервисов [4]. Эти сервисы могут быть написаны на разных языках программирования и использовать разные технологии хранения данных. Каждый микросервис в кластере имеет свои данные, свою модель, он автономен и, как правило, помещается в свой контейнер. Чем крупнее проект и больше сервисов, тем больше количество контейнеров, которыми необходимо управлять. Эта задача управления ложится на оркестраторов.

Они же управляют жизненными циклами контейнеров микросервисных приложений.

В настоящее время в цифровом пространстве представлено более десятка реализованных платформ оркестрации контейнерами, как декларативных, так и императивных. В нашей работе представлены результаты исследований по сравнению наиболее распространенных платформ в рамках поиска оптимального инструмента по развертыванию системы администрирования сети. Практический опыт имеется по работе с Kubernetes и в меньшей степени с Docker Swarm.

Системы оркестрации контейнеров. Преимущества и недостатки

При выборе инструмента оркестрации контейнеров или управляемой службы оркестровки контейнеров необходимо учитывать следующие ключевые моменты:

- сеть;
- доступность;
- простота развертывания и обслуживания;
- масштабируемость;
- обнаружение служб;
- безопасность и соответствие;
- поддержка (сообщество и предприятие);
- административные накладные расходы.

Цель нашей работы заключается в сопоставлении наиболее известных продуктов, числе которых, по общему признанию IT-сообщества, наиболее продвинутым является Kubernetes [3].

Kubernetes – это платформа с открытым исходным кодом, изначально разработанная Google и в настоящее время поддерживаемая Cloud Native Computing Foundation (CNCF). Kubernetes поддерживает как декларативную конфигурацию, так и автоматизацию. Он поставляется с отличным планировщиком и менеджером ресурсов для более эффективного развертывания широкодоступных контейнеров. Согласно CNCF, существует более 109 инструментов для управления контейнерами, но 89% используют различные формы Kubernetes. Проект Kubernetes поддерживается облачным фондом с участниками по всему миру. В число участников входят как крупные организации, так и отдельные разработчики с открытым исходным кодом.

Kubernetes предлагает множество функций, которых нет в родных инструментах Docker [2]. Управляемые сервисы здесь играют ключевую роль.

Важное примечание. Kubernetes удалил среду выполнения контейнера Docker после версии 1.20. Это не означает, что образы докеров не будут работать в Kubernetes. Просто базовая среда выполнения контейнера переместилась из Docker в среду выполнения.

Отметим следующие особенности:

- обнаружение служб и балансировка нагрузки;
- оркестрация системы хранения данных;
- автоматизированные развертывания и откаты;
- горизонтальное масштабирование;
- управление секретом и конфигурацией;
- самовосстановление;
- пакетное выполнение;
- двойной стек IPv4/IPv6;
- автоматическая упаковка ячеек.

В качестве следующего инструмента оркестрации рассмотрим платформу **Docker Swarm**. Встроенная утилита Docker Swarm (Docker in swarm mode, «Docker в режиме роя») — самый простой в развертывании и управлении оркестратор. Это хороший выбор для организации, которая только начинает использовать контейнеры в производстве. Swarm легко интегрируется с остальной частью набора инструментов Docker, например, с Docker Compose (*инструментальное средство, входящее в состав Docker. предназначено для решения задач, связанных с развёртыванием проектов*) и Docker CLI (*консольный клиент, позволяющий управлять Docker через интерфейс командной строки*). Тем самым обеспечивается привычный пользовательский интерфейс. Для контейнеров Docker Swarm считается более безопасным и простым в устранении неполадок, чем Kubernetes.

Преимущества **Docker Swarm**

Удобная установка. Для работы Docker Swarm использует тот же интерфейс командной строки, что и Docker. Это упрощает настройку и дальнейшую работу пользователей.

Совместимость. Docker Swarm работает поверх Docker и идеально совместим с другими инструментами этой экосистемы. Пользователи работают с одним и тем же интерфейсом командной строки Docker, который обеспечивает простую в использовании структуру команд.

Скорость. Docker Swarm предоставляет динамичную среду, которая позволяет приложениям быстро запускаться в виртуальном пространстве.

Хорошая документированность. Docker постоянно обновляет свою документацию, чтобы давать пользователям самую последнюю информацию по изменениям в своей экосистеме.

Контроль версий. Пользователи могут легко отслеживать текущие версии docker-контейнера, чтобы контролировать расхождения с предыдущими версиями.

Недостатки **Docker Swarm**

Зависимость от платформы. Docker Swarm поддерживает несколько операционных систем, но только на базе Linux.

Хранилище. В Docker нет встроенной реализации хранилища, поэтому Docker Swarm — не самое простое решение для подключения контейнеров к хранилищу.

Мониторинг. В Docker Swarm нет встроенных инструментов расширенного мониторинга, позволяющих собрать больше данных в режиме реального времени.

Следующей распространенной цифровой платформой является **OPENSIFT**. Он позволяет автоматизировать приложения на безопасных и масштабируемых ресурсах в гибридных облачных средах. Предоставляет платформы корпоративного уровня для создания, развертывания и управления контейнерными приложениями.

Сервис построен на базе Redhat Enterprise Linux и Kubernetes. Openshift имеет различные функциональные возможности для управления кластерами через интерфейс пользователя и интерфейс командной строки. Redhat предоставляет Openshift еще в двух вариантах:

- Openshift Online - предлагается как программное обеспечение в качестве услуги (SaaS);
- выделенный OpenShift - предлагается как управляемые услуги.

Openshift Origin (Origin Community Distribution) - родительский проект сообщества с открытым исходным кодом, который используется в OpenShift Container Platform, Openshift Online и OpenShift Distributed.

В завершение, из ряда платформ оркестрации контейнеров рассмотрим **Rancher**. Rancher — это инструмент оркестрации контейнеров с открытым исходным кодом. Он обеспечивает выполнение следующих функций:

- централизованная подготовка кластера, поддерживающая локальные, облачные и периферийные ресурсы;

- оптимизация операций Kubernetes за счет управления операциями кластера с единой консоли;
- централизованная безопасность Kubernetes с помощью централизованных пользовательских политик;
- интуитивно понятное управление рабочей нагрузкой с помощью собственного API Kubernetes или утилиты Kubectl;
- интегрированный мониторинг и ведение журналов с использованием Prometheus, Fluentd и Grafana;
- поддержка управления кластерами Amazon EKS (Amazon Elastic Container Service) и Google Kubernetes Engine (GKE);
- глобальный каталог приложений для упрощения установки и обновления приложений.

Дополнительно к изложенному приведем преимущества и недостатки инструментов оркестрации *управляемых контейнеров* и оркестрации *контейнеров с собственным размещением*.

Общими свойствами преимуществ для обоих видов являются:

- возможность развертывания в один клик;
- интеграция с родными сервисами облачной платформы;
- высокая масштабируемость;
- безопасность и соответствие требованиям корпоративного уровня.

Дополнительно, оркестрации *управляемых контейнеров* свойственна высокая доступность, а с *собственным размещением* - экономия административных расходов.

К недостаткам обоих видов реализации следует отнести:

- отсутствие контроля над плоскостью управления;
- необходимость придерживаться вариантов, предоставляемых поставщиком облачных услуг.

Заключение

В заключение, как выбрать инструмент оркестрации контейнеров? На этот вопрос нет однозначного ответа. Выбор инструмента оркестрации контейнеров зависит от различных факторов, таких как размер команды, бюджет, доступные SME (Small and medium-sized enterprises) и совместимость приложений, а также соответствие требованиям безопасности.

Литература

1. Андрей Маркелов. Введение в технологии контейнеров и Kubernetes. ДМК Пресс, 2019 - 194 с. ISBN 978-5-97060-775-6

2. Рой Айзенберг. Docker для разработчиков Rails. Pragmatic Programmers, LLC., 9650 Strickland Rd, #103-255. Raleigh, NC 27615. USA. ISBN 978-1-68050-273-2. 2019-02-23.
<http://creativecommons.org/licenses/by/3.0/legalcode>
3. Сайфан Джиджи. Осваиваем Kubernetes. Оркестрация контейнерных архитектур. - СПб.: Питер, 2019 -400 с. ISBN 978-5-4461-0973-9
4. Сэм Ньюмен. Создание микросервисов – СПб.: Питер, 2016 -304с. ISBN 978-5-496-02011- 4

References

1. Andrej Markelov. Vvedenie v tehnologii kontejnerov i Kubernetes. DMK Press, 2019 - 194 s. ISBN 978-5-97060-775-6
2. Roj Ajzenberg. Docker dlja razrabotchikov Rails. Pragmatic Programmers, LLC., 9650 Strickland Rd, #103-255. Raleigh, NC 27615. USA. ISBN 978-1-68050-273-2.2019-02-23.
<http://creativecommons.org/licenses/by/3.0/legalcode>
3. Sajfan Dzhidzhi. Osvaivaem Kubernetes. Orkestracija kontejnernih arhitektur. - SPb.: Piter, 2019 -400 s. ISBN 978-5-4461-0973-9
4. Sem N'jumen. Sozдание mikroservisov – SPb.: Piter, 2016 -304s. ISBN 978-5-496-02011- 4