Kubernetes vs Docker: Объяснение разницы

28.04.2023

Если вы пытаетесь выбрать между Docker и Kubernetes, вы вряд ли придете к однозначному ответу. Эти две технологии настолько фундаментально различны, что их нельзя сравнивать напрямую. Однако “одна против другой” подчеркивает важность понимания этих двух технологий. Что они делают? Как они это делают? Какие преимущества дает каждый из них? В этой статье мы рассмотрим эти вопросы, чтобы помочь вам понять, какое место каждый инструмент занимает в вашем процессе разработки.

Kubernetes и Docker: Разные инструменты для разных целей

Современные приложения сложны и требуют установки различных фреймворков и библиотек на ваше устройство. К счастью, вы можете объединить ваше приложение и необходимые для него ресурсы. Этот процесс называется контейнеризацией, и он подразумевает объединение приложений и всех необходимых зависимостей в отдельный блок — контейнер. Такая упаковка приложения делает его гораздо более портативным и удобным для развертывания и управления.

Кроме того, контейнеризация может облегчить некоторые трудности, с которыми вы можете столкнуться при попытке воспроизвести среду развертывания для тестирования. В отличие от традиционной архитектуры приложений, которая требует создания отдельной среды тестирования вручную, контейнерные приложения позволяют проводить тестирование в среде, идентичной той, в которой будет развернуто приложение.

Контейнеры также позволяют развертывать и запускать несколько компонентов приложения в архитектуре микросервисов. Это

означает, что ресурсы вашего приложения используют одно и то же оборудование, и вы сохраняете больший контроль над каждым компонентом и его жизненным циклом. Контейнеры легче виртуальных машин, поскольку они используют ядро операционной системы (ОС) хоста и не требуют гипервизоров. В облачных средах технология контейнеризации позволяет обеспечить операционную эффективность, переносимость при миграции, согласованность среды и плавное масштабирование.

Что такое Docker?

Хотя существует множество технологий контейнеризации, Docker остается самой популярной и широко известной. Это инструмент контейнеризации с открытым исходным кодом, который создает экосистему, где можно развертывать, управлять и обмениваться приложениями. Docker появился в 2013 году, обеспечив контейнеризацию с непревзойденной эффективностью и простотой использования. Его инновационные функции позволили решить несколько проблем, которые ранее мешали разработчикам заниматься разработкой на основе контейнеров.

Основным компонентом Docker является Docker Engine, на котором размещаются контейнеры. Docker Engine запускается на хостовой ОС и взаимодействует с контейнерами для доступа к системным ресурсам. Docker также использует конфигурационные файлы YAML, которые определяют, как собрать контейнер и что в нем запускается. Это одна из причин портативности Docker и простоты устранения неполадок. Контейнеры Docker могут взаимодействовать друг с другом по определенным каналам, и каждый контейнер имеет уникальный набор приложений, библиотек и конфигурационных файлов. Они могут содержать любое приложение и работать на любом сервере. Это повышает гибкость и переносимость приложения, позволяя запускать его в различных условиях, включая локальное, публичное или частное облако.

Оркестровка контейнеров с помощью Kubernetes

Современное программное обеспечение в значительной степени опирается на микросервисы – независимо работающие компоненты, которые можно легко развернуть и быстро обновить. Контейнеры полезны для размещения микросервисной архитектуры. Однако по мере того, как приложения становятся все более сложными, ими трудно управлять вручную, поддерживать и переносить в различные среды. Это привело к появлению решений для оркестровки контейнеров. Оркестровка контейнеров – это процесс автоматизации таких операций, как развертывание, администрирование, масштабирование, балансировка нагрузки и сетевое взаимодействие, которые необходимы для работы контейнерных рабочих нагрузок. Все это происходит в масштабе нескольких узлов, известных как кластеры, что позволяет развертывать приложения в различных средах без перебоев.

Kubernetes, иначе известный как K8s, – это масштабируемая система оркестровки контейнеров с открытым исходным кодом, которая использует API для автоматизации процесса запуска приложений в сети и решения возникающих при этом сложностей. Компания Google разработала его и в 2015 году передала его с открытым исходным кодом в Cloud Native Computing Foundation. Вы создаете ресурсы Kubernetes декларативно. Сначала вы определяете все требования в конфигурационном файле YAML. Чтобы развернуть контейнер, Kubernetes находит лучший хост (машину, на которой размещается узел), отвечающий всем требованиям в файле Manifest.yml. Затем он автоматически планирует развертывание кластера на этом узле. Kubernetes также управляет жизненным циклом контейнера на основе заданных конфигураций.

Фреймворк Kubernetes использует следующие ключевые компоненты для обеспечения оркестровки контейнеров:

- **Node** – рабочая машина, на которую Kubernetes развертывает контейнеры.
- **Cluster** – группа связанных узлов. Наличие нескольких узлов помогает сбалансировать рабочие нагрузки, обеспечивая работу приложения даже в случае отказа одного из узлов.
- **Kubelet** – агент, который запускается на каждом узле и обеспечивает работу контейнеров в соответствии с ожиданиями.
- **Control Plane** – набор процессов, которым поручено контролировать все операции.
- **Pod** – объект, в котором заключены контейнеры, развернутые на узле. По сути, pod – это экземпляр приложения и самый маленький объект, который можно создать в Kubernetes.

Kubernetes – это отличный вариант для организаций, которым необходимо развертывать и управлять большим количеством контейнеров. Управление жизненным циклом контейнеров с помощью инструментов оркестровки приносит пользу командам DevOps, которые интегрируют их в рабочие процессы непрерывной интеграции/непрерывной разработки.

Docker Swarm

Docker Swarm – это собственное решение Docker для оркестровки контейнеров с открытым исходным кодом и альтернатива Kubernetes. Он предлагает масштабирование, многохостовую сеть, автоматическую балансировку нагрузки и все другие функции, необходимые для массового развертывания и администрирования контейнеров, не завися от сторонних инструментов оркестровки. У него простой процесс установки, он легкий, и его легко интегрировать, если вы уже привыкли к экосистеме Docker. Docker Swarm – отличный вариант при работе с несколькими узлами и относительно простыми приложениями. Однако если вы организуете работу крупных узлов для критически важных приложений, вам больше подойдут функции безопасности,

постоянного мониторинга, гибкости и отказоустойчивости Kubernetes.

Docker против Kubernetes

Уже очевидно, что Docker и Kubernetes имеют разные сценарии использования. Вы используете Docker для упаковки и доставки приложений и работы с одним узлом. Kubernetes же развертывает и масштабирует приложения на кластере узлов. Кроме того, Kubernetes управляет только контейнерами, а для их создания требуется отдельное программное обеспечение. Однако, несмотря на различие Kubernetes и Docker, их объединяет общая цель — создание масштабируемых контейнерных приложений. Они не являются ни конкурентами, ни взаимоисключающими. Они составляют идеальную команду.

Docker и Kubernetes

Docker создает и развертывает приложения на одном узле, а Kubernetes управляет приложениями на кластере узлов. При совместном развертывании Docker и Kubernetes могут использовать преимущества друг друга, обеспечивая приложениям масштабируемость, гибкость и отказоустойчивость. Kubernetes может сделать контейнеры Docker более устойчивыми, отслеживая состояние каждого узла в кластере. Он автоматически перезапускает, заменяет вышедшие из строя узлы и уничтожает не отвечающие на запросы узлы, которые не прошли проверку работоспособности. Балансировка нагрузки также гарантирует, что узлы не будут перегружены работой.

Kubernetes и Docker также предлагают богатый набор функциональных возможностей, которые помогают определить, как будут работать различные компоненты приложения. Это позволяет легко обновлять приложение по своему усмотрению. Кроме того, масштабирование происходит без проблем, поскольку вы можете быстро создавать контейнеры Docker, а Kubernetes может масштабировать кластеры с минимальным ручным вмешательством.

К другим преимуществам относятся:

- Оптимальное использование ресурсов
- Мониторинг работоспособности программного обеспечения
- Автоматизированные операции, такие как автоматическое развертывание и самовосстановление
- Оркестровка хранилищ

Кроме того, контейнеры Docker не зависят от системы и могут работать в любой среде, поддерживающей Docker Engine, что делает миграцию необременительной.

Заключение

Стратегическая интеграция Docker и Kubernetes не имеет границ. Обе технологии являются мощными и способны решать широкий спектр задач. Этот динамичный дуэт добился большого успеха в бессерверных вычислениях, развертывании в нескольких облаках, администрировании микросервисов и машинном обучении. Вместе Docker и Kubernetes — это лучший способ создать адаптируемую и эффективную среду разработки программного обеспечения. Docker обеспечивает быстродействие и системную независимость ваших приложений, а Kubernetes гарантирует максимальную работоспособность, правильную балансировку нагрузки и возможность масштабировать кластер по своему усмотрению.