Kubernetes

Предпосылки появления Kubernetes

Переход от монолитной архитектуры к микросервисам значительно увеличивает общую сложность системы. Вместо одного приложения необходимо управлять множеством сервисов, каждый из которых имеет свои зависимости, конфигурации и требования к развертыванию.Такой масштаб требует автоматизации:

- развертывания и обновления контейнеров;
- мониторинга их состояния;
- масштабирования и восстановления после сбоев;
- управления сетевыми и хранилищными ресурсами.

Для решения подобных задач был создан Kubernetes Kubernetes — универсальный инструмент управления контейнерными приложениями.

Назначение и возможности Kubernetes

Kubernetes предназначен для автоматизации всех этапов жизненного цикла контейнеров, включая:

- развертывание и обновление приложений;
- автоматическое масштабирование и балансировку нагрузки;
- обеспечение отказоустойчивости;
- управление сетевыми подключениями и хранилищами;
- контроль доступа и интеграцию с внешними системами.

Kubernetes может быть развернут как в публичных и частных облаках, так и на физических серверах. Платформа обладает высокой степенью расширяемости и активной экосистемой.

Под и его свойства

Под (Pod) — минимальная единица развертывания в Kubernetes, которая содержит один или несколько контейнеров с общими ресурсами. Обычно в нём группируют тесно связанные компоненты одного приложения.

Поду характерны следующие свойства:

- Является минимальной сущностью для размещения приложения в Kubernetes.
- Представляет собой абстракцию над одним или несколькими контейнерами, совместно использующими хранилище, сетевое пространство и конфигурацию.
- В большинстве случаев содержит один контейнер.
- Обладает собственным IP-адресом, действительным до удаления или пересоздания Pod'а.
- При пересоздании Pod получает новый IP-адрес, что необходимо учитывать при организации сетевого взаимодействия.

Далее рассмотрены некоторые другие компоненты Kubernetes, которые помогают управлять жизненным циклом подов.

Архитектура Kubernetes

Архитектура Kubernetes включает два типа узлов: **управляющие** (control plane) и **рабочие** (worker nodes).

Управляющие узлы (control plane)

Управляющие узлы отвечают за глобальное состояние кластера, планирование задач и принимают решения о размещении и масштабировании приложений.

Компоненты управляющего узла

- **API Server (kube-apiserver)** — API-интерфейс всего кластера, обрабатывает внешние и внутренние запросы, обращается к etcd, валидирует манифесты и инициирует действия других компонентов.
- **Scheduler (kube-scheduler)** — планирует размещение новых подов на доступных узлах на основе ресурсов, политик, приоритетов и ограничений. Выбирает наиболее подходящий узел для запуска пода.
- **Controller Manager (kube-controller-manager)** — запускает контроллеры, которые отслеживают состояние ресурсов и стремятся привести его в соответствие с желаемым. Далее перечислены контроллеры в составе Controller Manager:
  - **Replication Controller** - контролирует количество реплик приложения. Автоматически масштабирует поды вверх или вниз, чтобы соответствовать указанному количеству реплик.
  - **Node Controller** — отслеживает состояние узлов и инициирует восстановление при их сбое. Обрабатывает связанные с узлами события, например помечает узлы как

недоступные, если они перестают отвечать.

    - **Volume Controllers** — управляют привязкой и отсоединением постоянных хранилищ.
    - **Service Controller** — отвечает за доступность сервисов и их маршрутизацию.
    - **Access Controllers** — реализуют контроль доступа (RBAC).
    - **Configuration Controller** — отслеживает изменения в ConfigMap и Secret.
- **etcd** — распределенное хранилище key-value, в котором Kubernetes хранит всё состояние кластера: манифесты подов, данные о конфигурации и статусах. Хранилище обеспечивает надёжность и отказоустойчивость, сохраняет состояние кластера.

Рабочие узлы (worker nodes)

Рабочие узлы в Kubernetes отвечают за запуск контейнеров с приложениями.

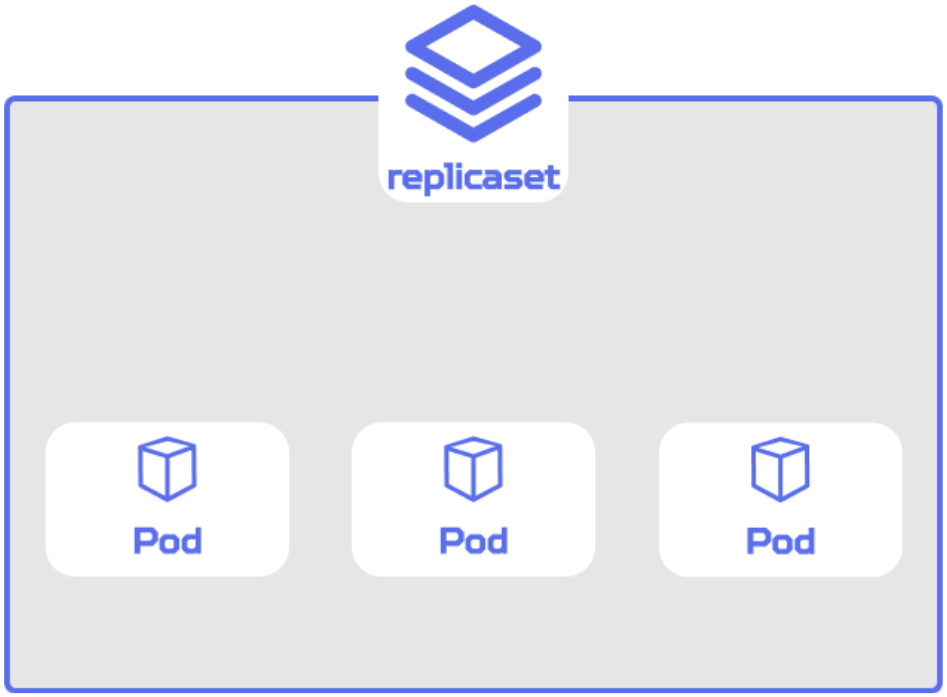
Компоненты рабочего узла

- **Kubelet** - агент, отслеживающий состояние подов, запущенных на узле, и взаимодействующий с управляющим уровнем.
- **Kube-proxy** - запущен на всех узлах и поддерживает сетевые правила на узлах. Эти правила разрешают сетевое взаимодействие с вашимии подами из сетевых сессий внутри и снаружи кластера.

Сервисы, контроллеры и компоненты

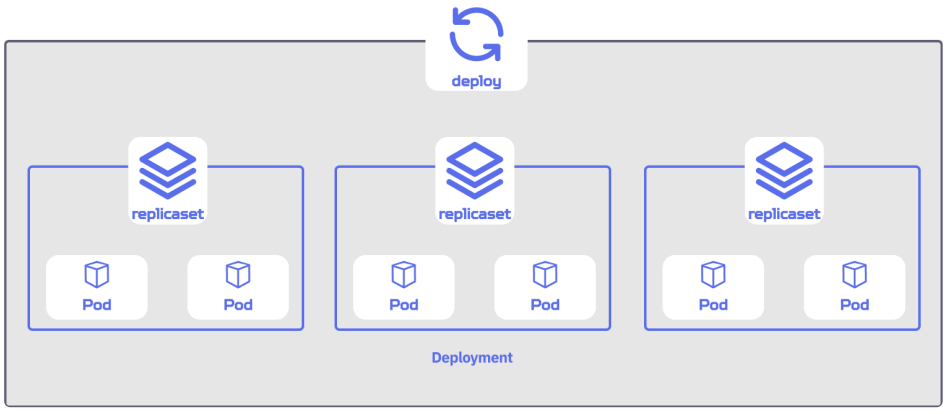
ReplicaSet

ReplicaSet в Kubernetes — это контроллер, который помогает определить и поддерживать заданное количество экземпляров подов в кластере. Он обеспечивает работу необходимого количества модулей и автоматически заменяет любые модули, которые выходят из строя, завершают работу или удаляются.



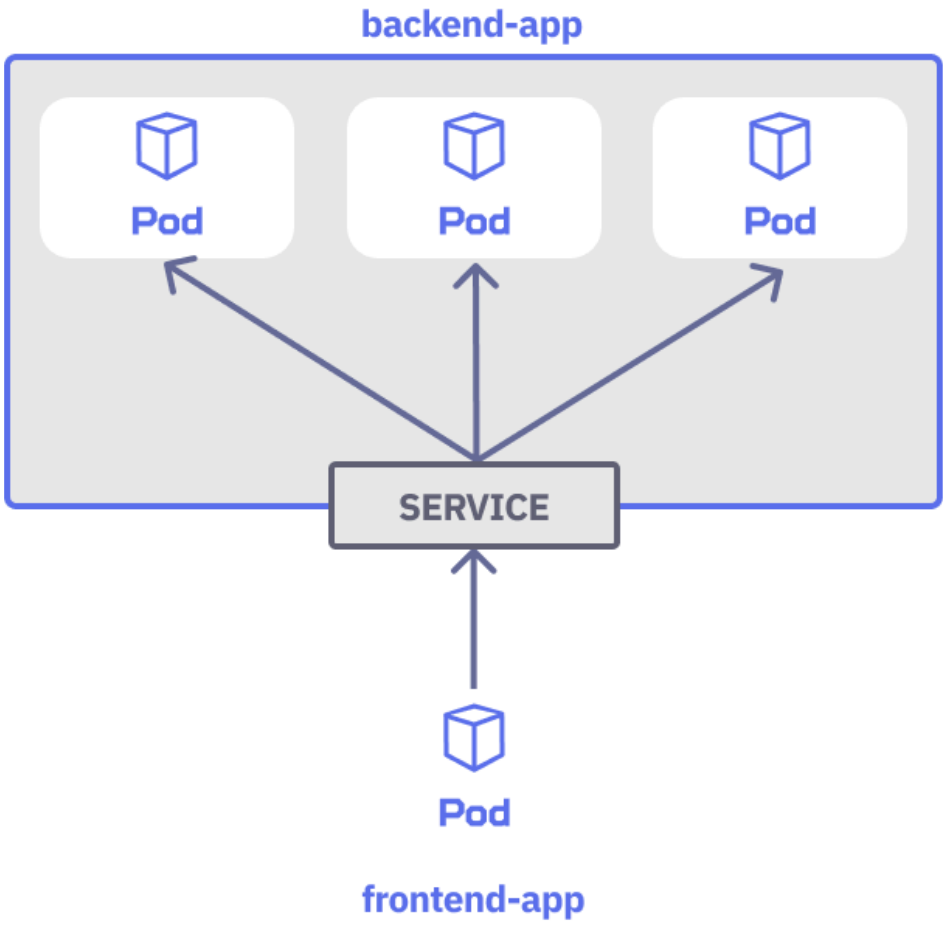
Deployment

Deployment в Kubernetes — это высокоуровневый ресурсный объект, который управляет развертыванием и масштабированием приложений. Он поддерживает желаемое состояние приложения и позволяет масштабировать контейнеры вверх и вниз в зависимости от входящего трафика



Сетевые абстракции Service в Kubernetes

Service в Kubernetes — это абстракция, которая определяет логическую группу подов (Pods) и политику, по которой к этим подам можно обращаться.



Основные задачи сервисов:

- Позволяют внешним и внутренним компонентам общаться с приложением;
- Предоставляют стабильные IP-адреса и DNS-имена для доступа к подам;
- Могут балансировать нагрузку между несколькими подами.

Далее рассмотрим типы серивиса в Kubernetes.

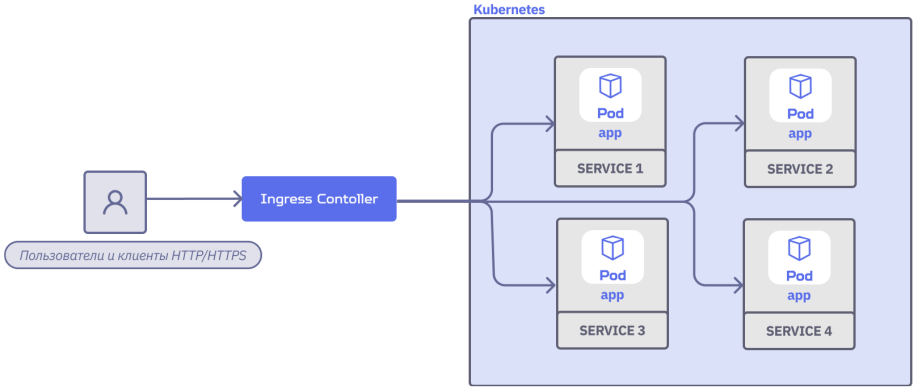
- **ClusterIP** — стандартный тип сервиса в Kubernetes, который обеспечивает внутреннюю связь между различными компонентами приложения. Данному типу сервиса характерны свойства:
  - Предоставляет стабильный внутренний IP-адрес и DNS-имя для доступа к приложению.
  - Используется для взаимодействия между компонентами внутри кластера.
  - Выполняет балансировку нагрузки между репликами.
  - Не доступен извне без дополнительной настройки (например, через Ingress или NodePort).
- **NodePort** — более открытый тип сервиса в Kubernetes, который позволяет получать доступ к подам через фиксированный порт на каждом узле кластера. Данному типу

сервиса характерны свойства:

  - Открывает доступ к приложению из внешней сети.
  - Использует фиксированный диапазон портов (по умолчанию 30000–32767).
  - Требует знания IP-адресов рабочих узлов.
  - Открытие портов напрямую на узлах требует дополнительного контроля и может не соответствовать требованиям безопасности.
- **LoadBalancer** — тип сервиса в Kubernetes, который обеспечивает доступ пода к внешнему трафику. Данному типу сервиса характерны свойства:
  - Предоставляет внешний IP-адрес или доменное имя для доступа к приложению.
  - Интегрируется с провайдерами облачных платформ (например, AWS, GCP, Azure).
  - В средах без встроенного балансировщика (on-premise) требует установки стороннего решения, например, MetalLB.

Ingress-контроллер

Ingress Controller в Kubernetes — это компонент, который управляет внешним доступом к сервисам внутри кластера, обычно трафиком по протоколам HTTP или HTTPS.



Функции и задачи

- **Маршрутизация трафика.** Контроллер анализирует входящие запросы, собирает данные (имя хоста, путь и атрибуты заголовка) и на основе правил Ingress-ресурса определяет службу, куда пересылать запрос.
- **Балансировка нагрузки.** Контроллер распределяет входящий трафик между несколькими репликами сервиса, что обеспечивает высокую доступность и масштабируемость.
- **Шифрование/дешифрование SSL/TLS-соединений.** При необходимости контроллер может выполнять шифрование и дешифрование SSL/TLS-сертификатов.
- **Проверка состояния сервисов.** Контроллер может проводить проверки состояния сервисов бэкенда, чтобы направлять трафик только на рабочие экземпляры.

- Может быть реализован с использованием Nginx, HAProxy, Traefik и других контроллеров.

Управление конфигурацией и секретами

- **ConfigMap** — это ресурс, который хранит неконфиденциальные данные конфигурации в виде пар key-value. Эти данные могут включать в себя параметры конфигурации, настройки окружения, конфигурационные файлы и другие сведения, необходимые приложению для корректной работы.
- **Secret** — это объект, предназначенный для безопасного хранения конфиденциальной информации. С помощью Secrets можно хранить такие данные, как пароли, токены API и сертификаты. Secrets позволяют отделить чувствительные сведения от кода приложения и конфигурационных файлов, что упрощает управление безопасностью и снижает риск утечек данных.

Эти компоненты образуют основу для построения, конфигурации и управления приложениями в Kubernetes.
