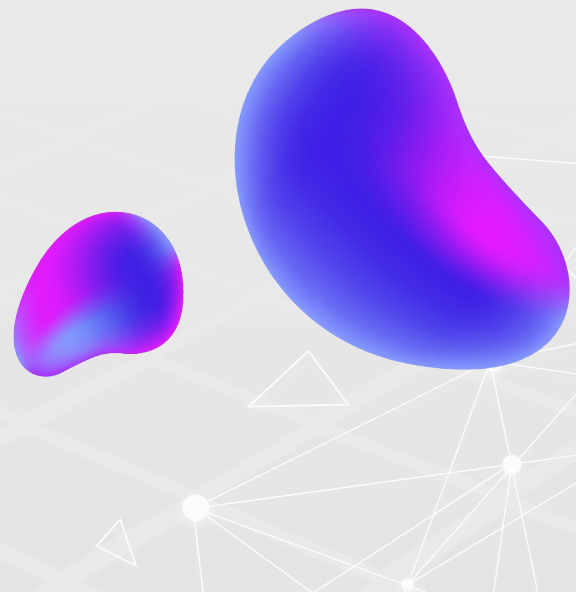



Введение в Kubernetes День 1

Летняя школа ВШЭ 2024.





Содержание

01

Эволюция
архитектур
приложений

02

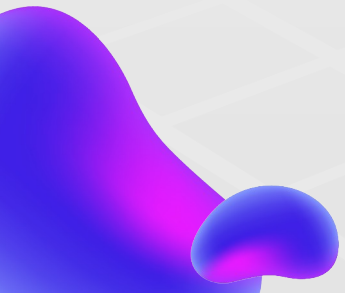
Виртуализация и
контейнеры

03

Kubernetes

04

Декларативный
подход k8s





Обо мне

Общий стаж в ИТ – 13 лет, из них 7 лет в кибербезопасности.

Сейчас работаю в отделе безопасности инфраструктуры Okko.



01

Эволюция архитектур приложений

Краткая история

1990's

SPAGUETTI ARCHITECTURE
(aka Copy & Paste Mess)



2000's

LASAGNA ARCHITECTURE
(aka Layered Monolith)



2010's

RAVIOLI ARCHITECTURE
(aka Microservices)



2020's

RISOTTO ARCHITECTURE
(aka Serverless)



Архитектура программного обеспечения

Структура системы

Определяется архитектурными стилями и паттернами, такими как микросервисы, многослойная архитектура и т.д.

Архитектурные решения

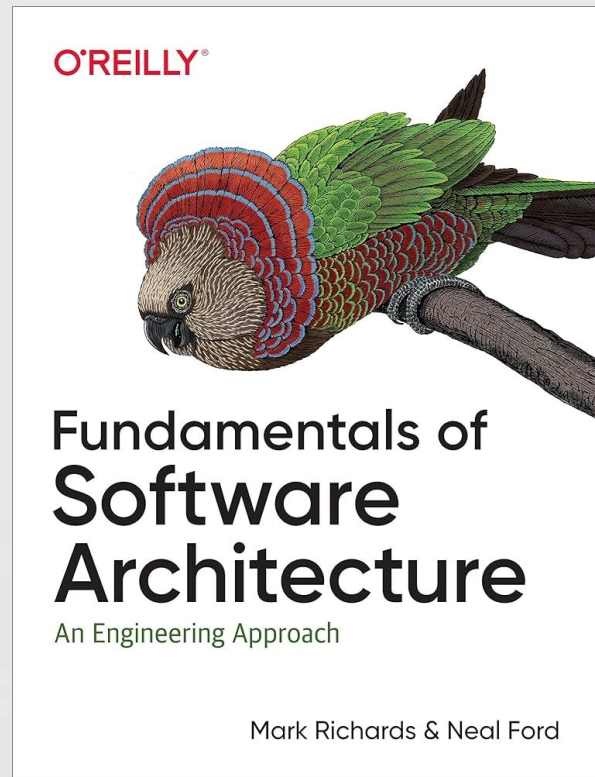
Определяют, как система должна быть построена. Они включают в себя правила и ограничения, направленные на достижение определенных архитектурных целей

Архитектурные характеристики

Нефункциональные требования, например, масштабируемость, отказоустойчивость и т. д.

Принципы проектирования

Руководящие указания, которые помогают при создании системы, являются подходами для решения типичных задач.



Монолитная архитектура

Структура системы

Вся система — это единое приложение, где все компоненты работают вместе.

Архитектурные характеристики

Простота разработки, но проблемы с масштабируемостью и отказоустойчивостью.

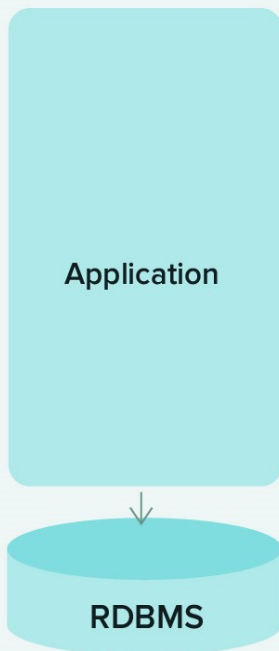
Архитектурные решения

Использование единого стека технологий для всего приложения

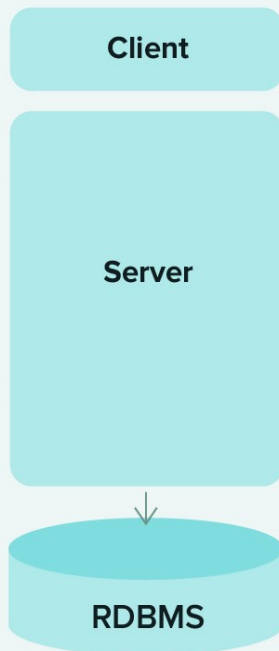
Принципы проектирования

Модульность внутри одного приложения с риском влияния изменений на всю систему

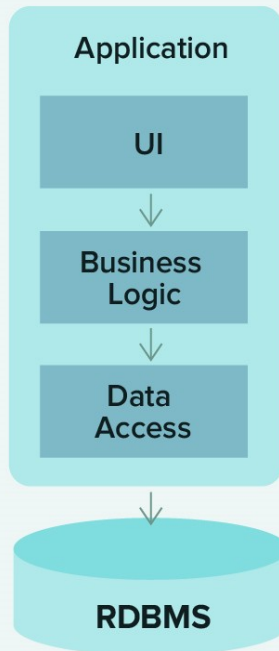
MONOLITHIC 1



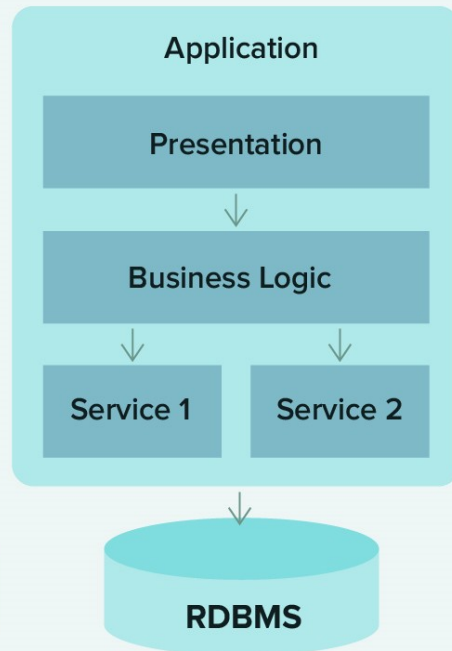
MONOLITHIC 2



MONOLITHIC 3



MONOLITHIC 4



Микросервисная архитектура

Структура системы

Система разделена на автономные сервисы, взаимодействующие через API.

Архитектурные характеристики

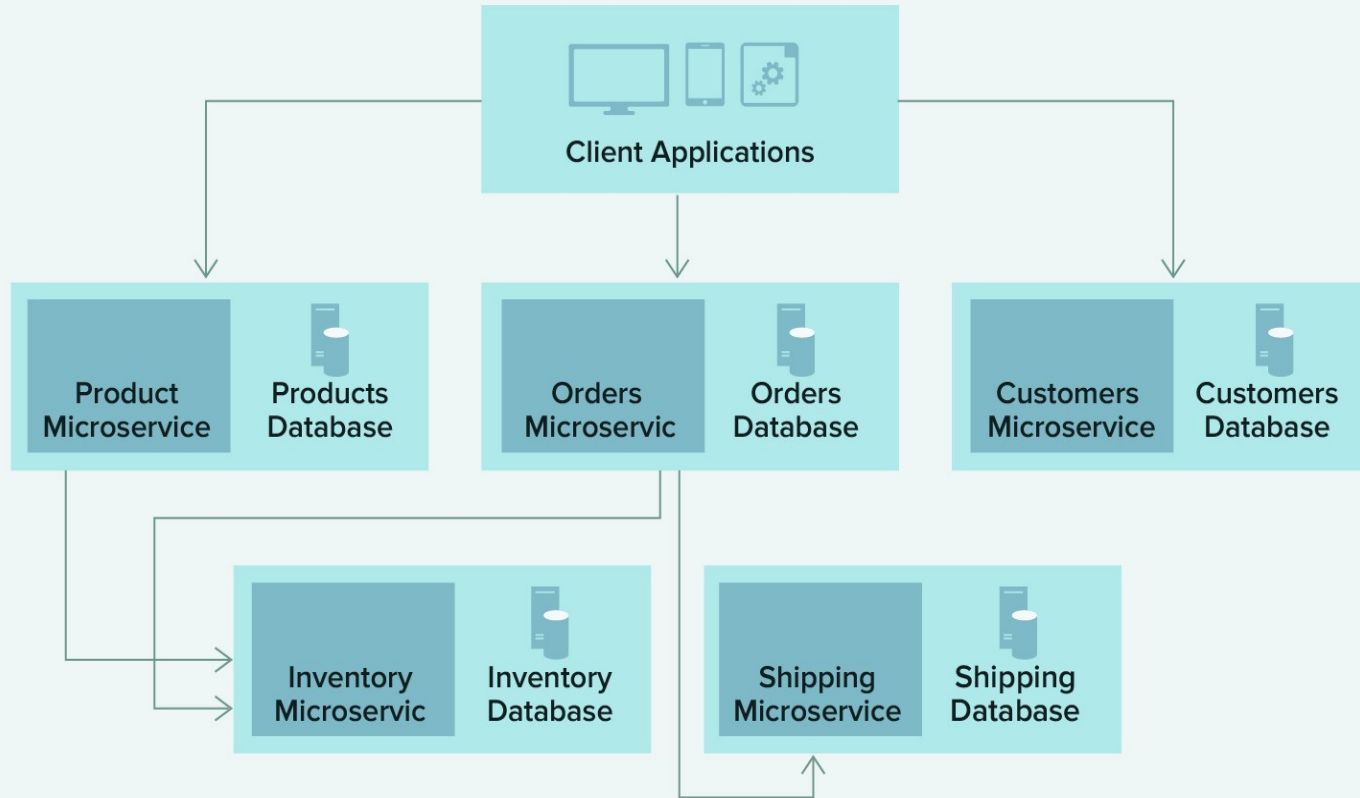
Легкое масштабирование и высокая отказоустойчивость, но сложность управления

Архитектурные решения

Высокая производительность за счет отсутствия сетевых взаимодействий между компонентами

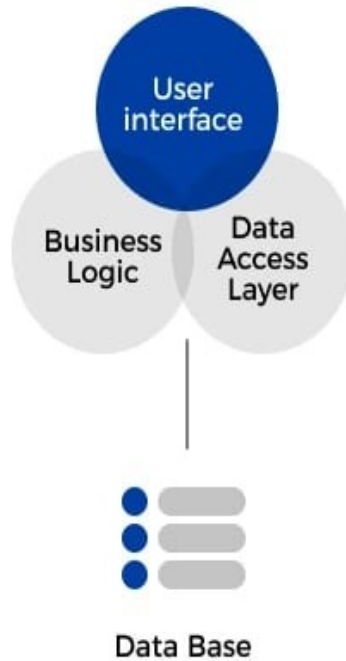
Принципы проектирования

Каждый сервис выполняет одну функцию, упрощая изменение и тестирование

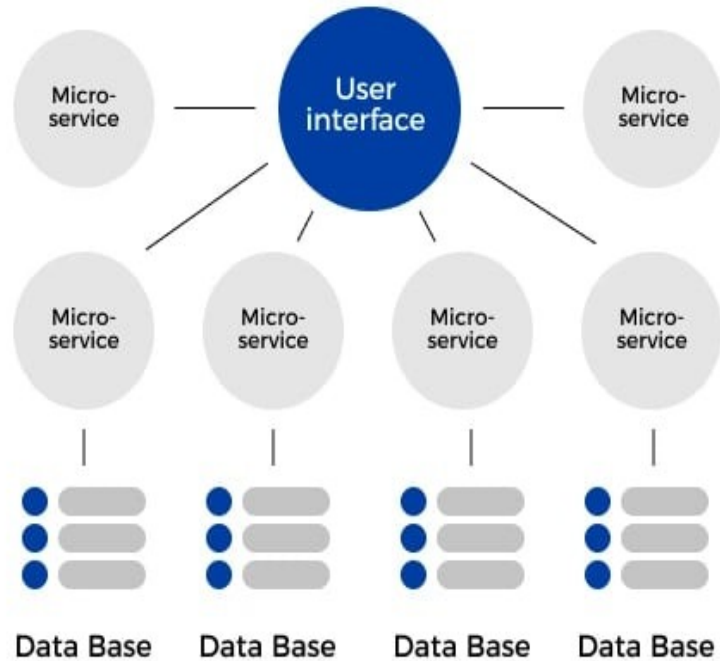


Фактор	Монолитная архитектура	Микросервисная архитектура
Масштабируемость	Требуется масштабирование всей системы	Масштабирование отдельных сервисов
Поддерживаемость	Простая отладка, но сложное обновление всего приложения	CI/CD уменьшает ошибки, требует сложного мониторинга
Отказоустойчивость	Сбой одного модуля может вызвать сбой всей системы	Сбой одного сервиса не влияет на всю систему
Технологическая зависимость	Использует один стек технологий	Возможность использования разных технологий для каждого сервиса

MONOLITHIC ARCHITECTURE



MICROSERVICE ARCHITECTURE

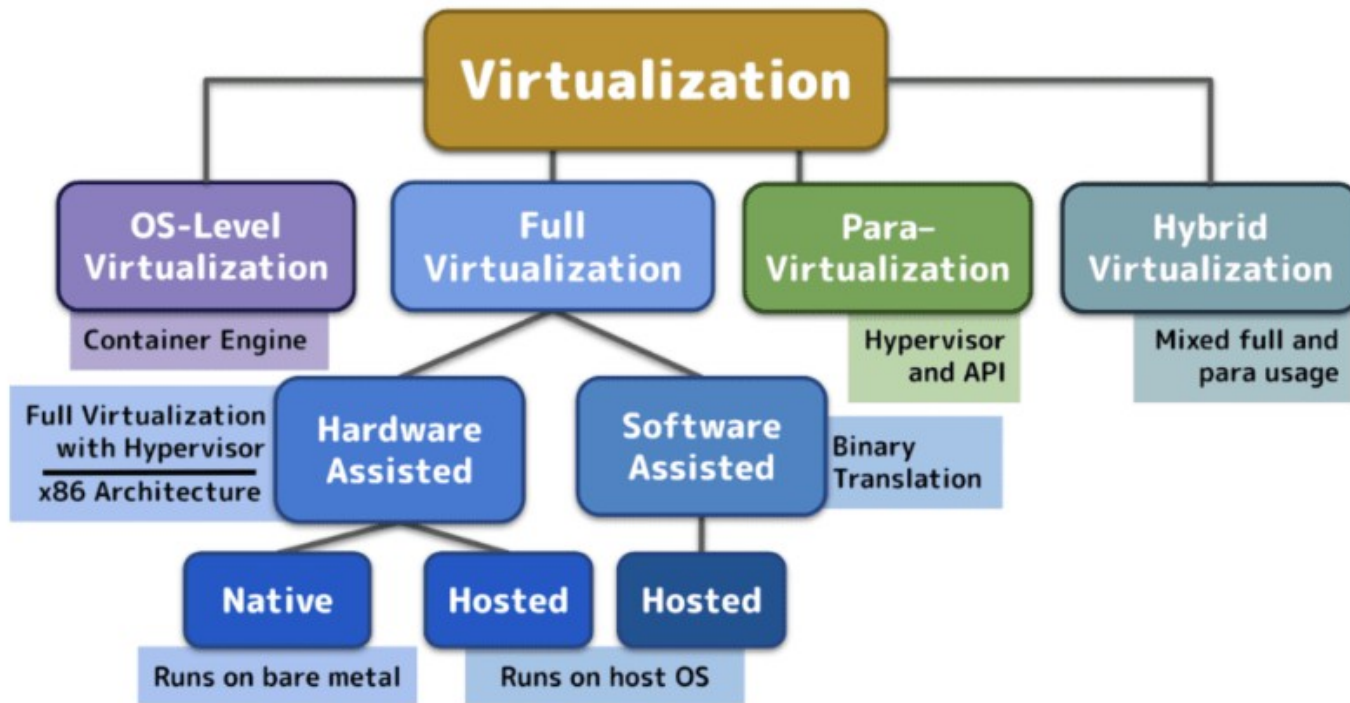




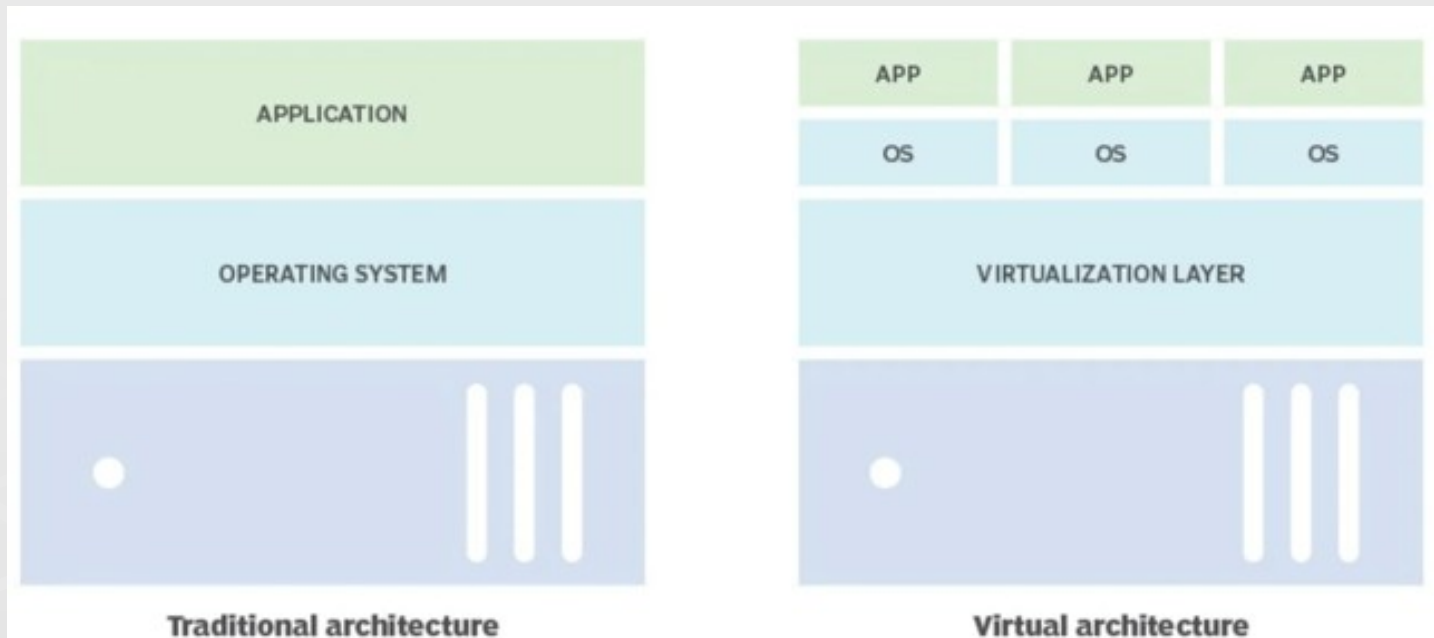
02

Виртуализация и контейнеры

Типы виртуализации



Традиционная и виртуальная архитектура



The slide features decorative geometric patterns in the corners. The top-left corner has a network of lines and triangles. The bottom-left corner has a repeating diamond-shaped grid pattern. The bottom-right corner has a network of lines and triangles.

Что такое контейнер?

Контейнер это (Docker)...

Контейнер — это стандартная единица программного обеспечения, которая упаковывает код и все его зависимости, обеспечивая быстрое и надёжное выполнение приложения в различных вычислительных средах. Образ контейнера Docker — это лёгкий, автономный, исполняемый пакет программного обеспечения, включающий всё необходимое для запуска приложения: код, среду выполнения, системные утилиты, системные библиотеки и настройки.

<https://www.docker.com/resources/what-container/>

Что такое OCI?

OCI (Open Container Initiative) - организация созданная для разработки и внедрения открытых индустриальных стандартов вокруг форматов контейнеров и сред их выполнения (runtime).

Основные спецификации:

- runtime-спес - определяет, как должен запускаться контейнер, представляющий собой файловую систему, которая распакована на диск
- image-спес
- distribution-спес

<https://github.com/opencontainers>

Контейнер это (OCI)...

Среда для выполнения процессов с настраиваемой изоляцией и ограничениями ресурсов. Например, пространства имен, лимиты ресурсов и монтирования — всё это элементы контейнерной среды.

Цель стандартного контейнера — инкапсулировать программный компонент и все его зависимости в формате, который является самодокументируемым и переносимым, так что любой совместимый рантайм может запустить его без дополнительных зависимостей, независимо от аппаратной платформы и содержимого контейнера.

<https://github.com/opencontainers/runtime-spec/blob/20a2d9782986ec2a7e0812ebe1515f73736c6a0c/glossary.md#container>



Преимущества виртуализации ОС

Разделение ресурсов

Позволяет одной ОС разделить свои ресурсы на несколько изолированных сред, каждая из которых функционирует как отдельная виртуальная среда

Эффективность

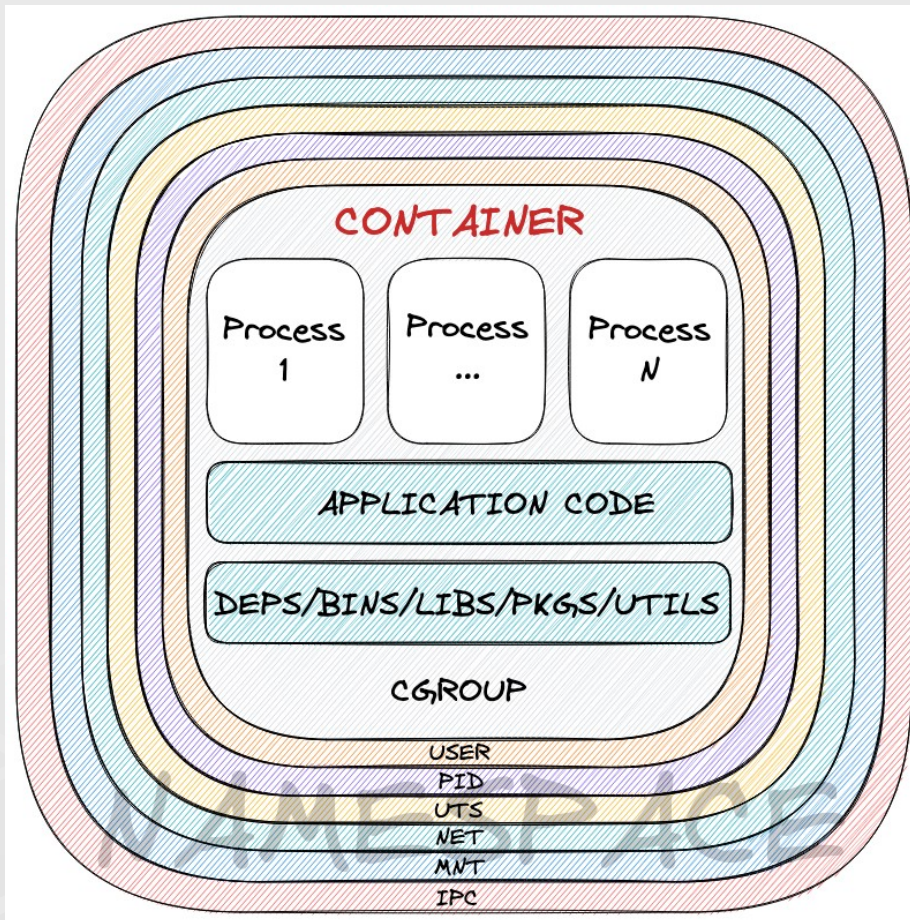
Виртуальные среды используют ядро хост-системы, что делает их легковесными и быстрыми

Изоляция

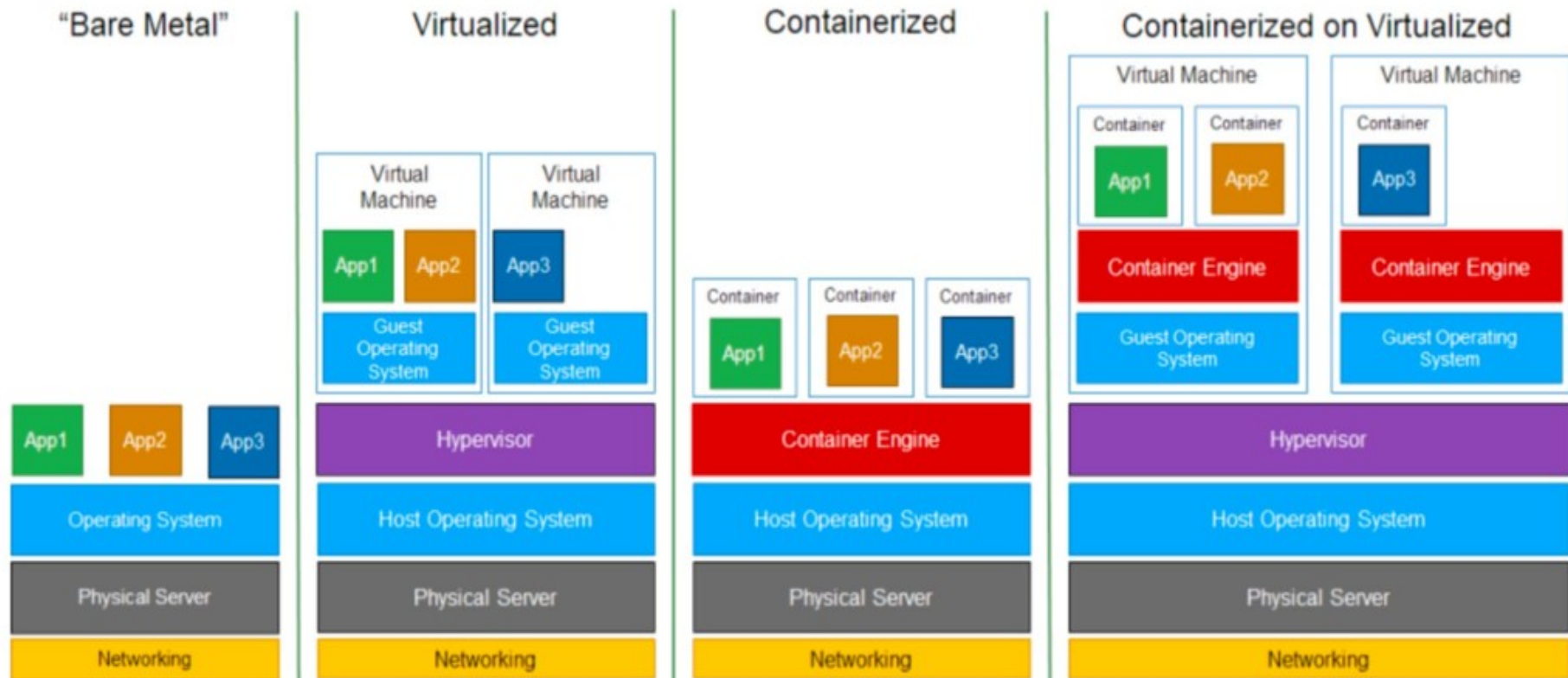
Виртуальные среды изолированы друг от друга, что предотвращает конфликты и повышает безопасность



Что такое контейнер?

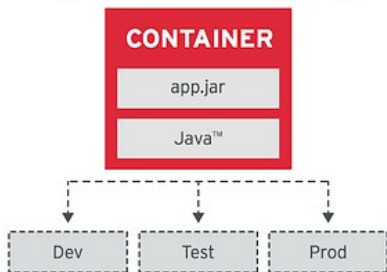


Виртуализация\контейнеризация

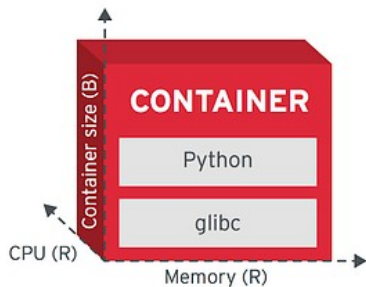


Принципы проектирования контейнеров (OCI)

Image Immutability Principle



Runtime Confinement Principle



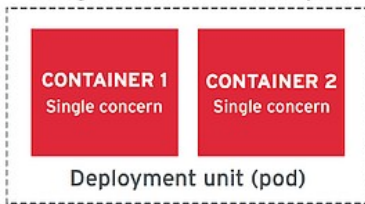
High Observability Principle



Lifecycle Conformance Principle



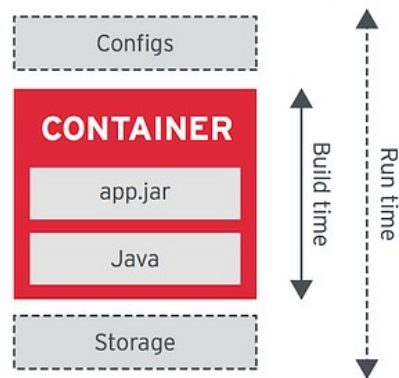
Single Concern Principle



Process Disposability Principle

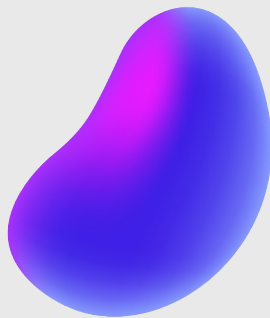


Self-Containment Principle



03

Kubernetes



Проблемы до пришествия K8s

Сложность управления
масштабируемыми
средами

Самовосстановление и
отказоустойчивость

Отсутствие
оркестрации на уровне
кластера

Балансировка нагрузки
и управление
трафиком

А разве Docker-Compose не решил часть проблем?

Масштабирование

Нет поддержки автоматического масштабирования контейнеров на основе нагрузки

Балансировка нагрузки

Базовая балансировка, нет встроенной поддержки L7-балансировки (требуются сторонние решения).

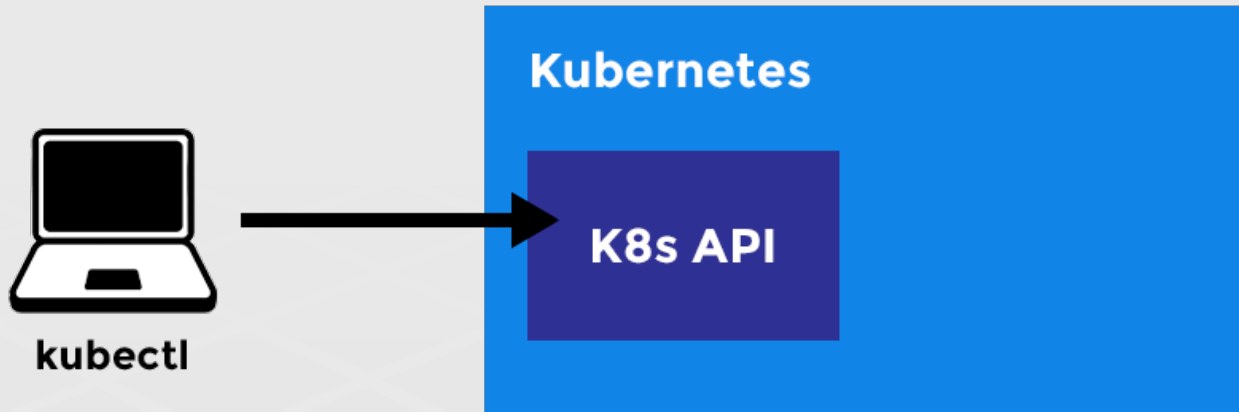
Управление на уровне кластера

Ограничено одним узлом, отсутствие поддержки для управления контейнерами на нескольких серверах

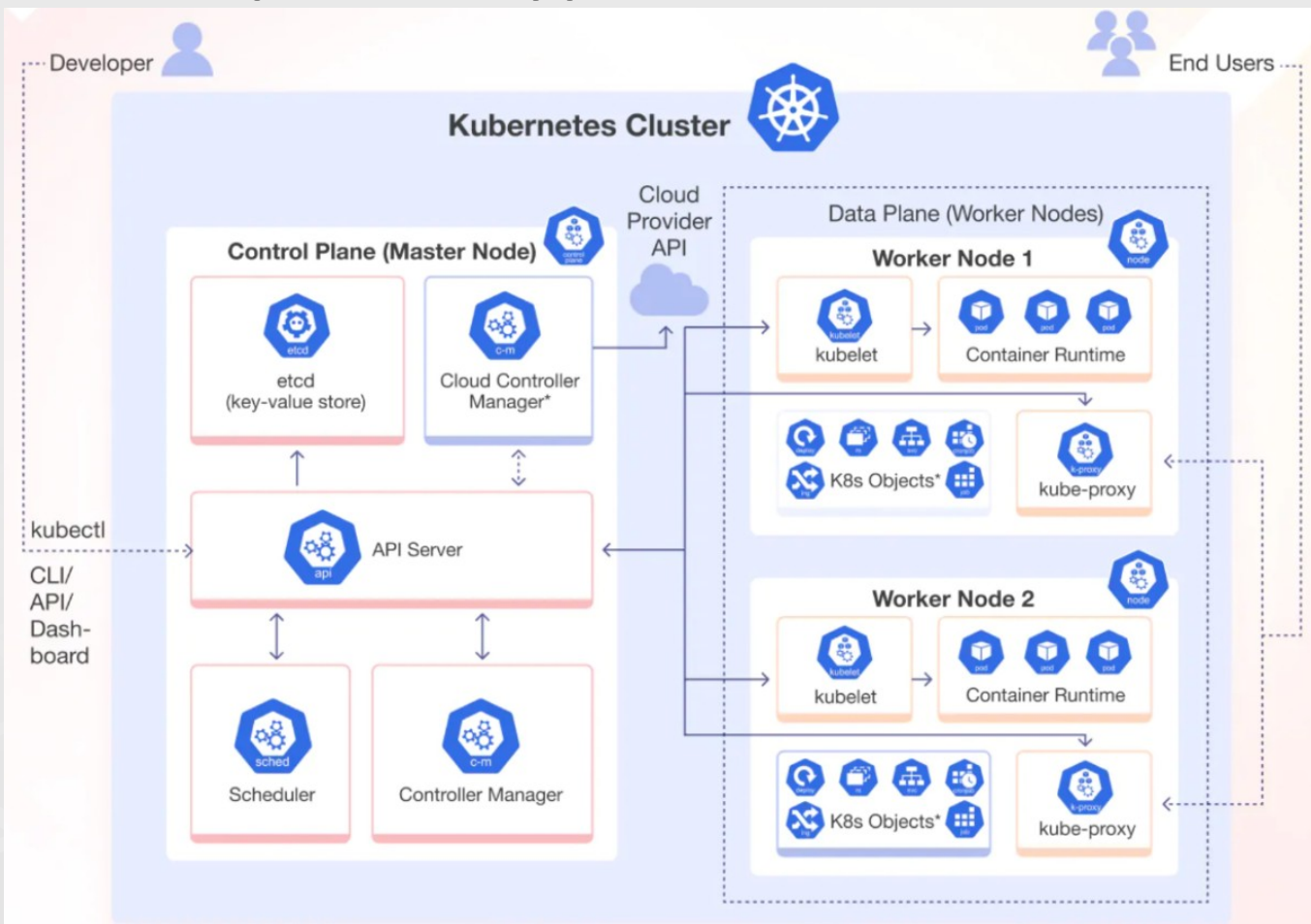
Самовосстановление

Нет автоматического перезапуска контейнеров в случае сбоев.

Kubectl

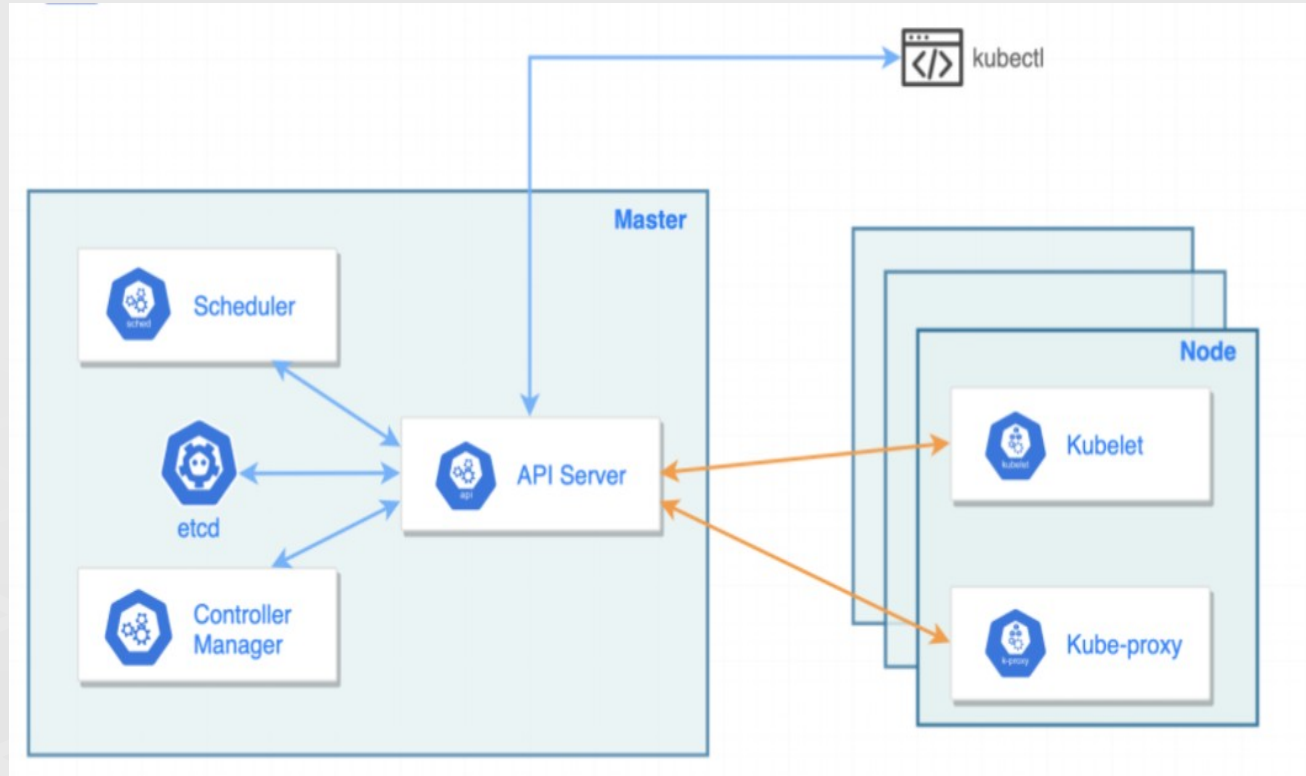


Архитектура Kubernetes



K8s - Control Plane

- Основная машина, управляющая узлами.
- Основная точка входа для всех административных задач.
- Он отвечает за оркестровку рабочих узлов.





K8s - Control Plane



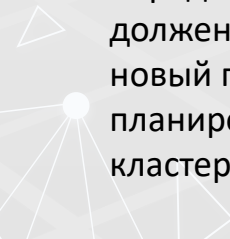
etcd

Обеспечивает согласованность данных в кластере, используется для хранения информации о состоянии всех объектов Kubernetes

kube-apiserver

Служит интерфейсом между всеми компонентами кластера и внешними клиентами, валидирует и конфигурирует данные API-объектов

kube-scheduler



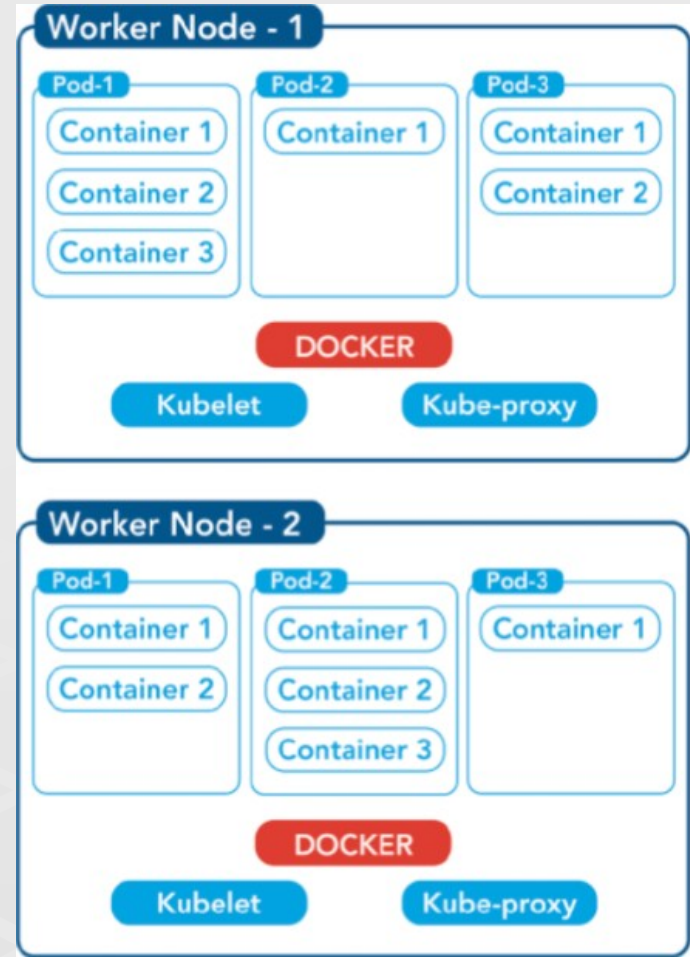
Анализирует ресурсы нод и требования подов, чтобы определить, на какой ноде должен быть развернут каждый новый под, ответственный за планирование подов на ноды кластера

kube-controller-manager

Выполняет фоновые задачи управления, такие как поддержание желаемого состояния подов, реплик, конечных точек и других объектов в соответствии с заданными спецификациями

K8s – Data Plane

- Worker Node выполняет запрошенные master задачи.
- Каждый узел контролируется главным узлом.
- Запускает pods (содержащие контейнеры внутри)
- Здесь запускается контейнерный runtime, который отвечает за загрузку образов и запуск контейнеров.





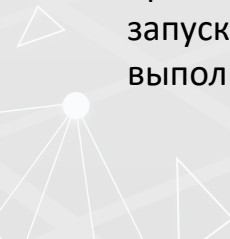
K8s – Data Plane



kubelet

Запуск подов, обеспечение их работы и поддержание их в заданном состоянии. Kubelet получает инструкции от API-сервера и управляет запуском контейнеров через контейнерный runtime

Pods



Выполняют пользовательские приложения и сервисы. Поды запускаются на узлах и обеспечивают выполнение конкретных задач.

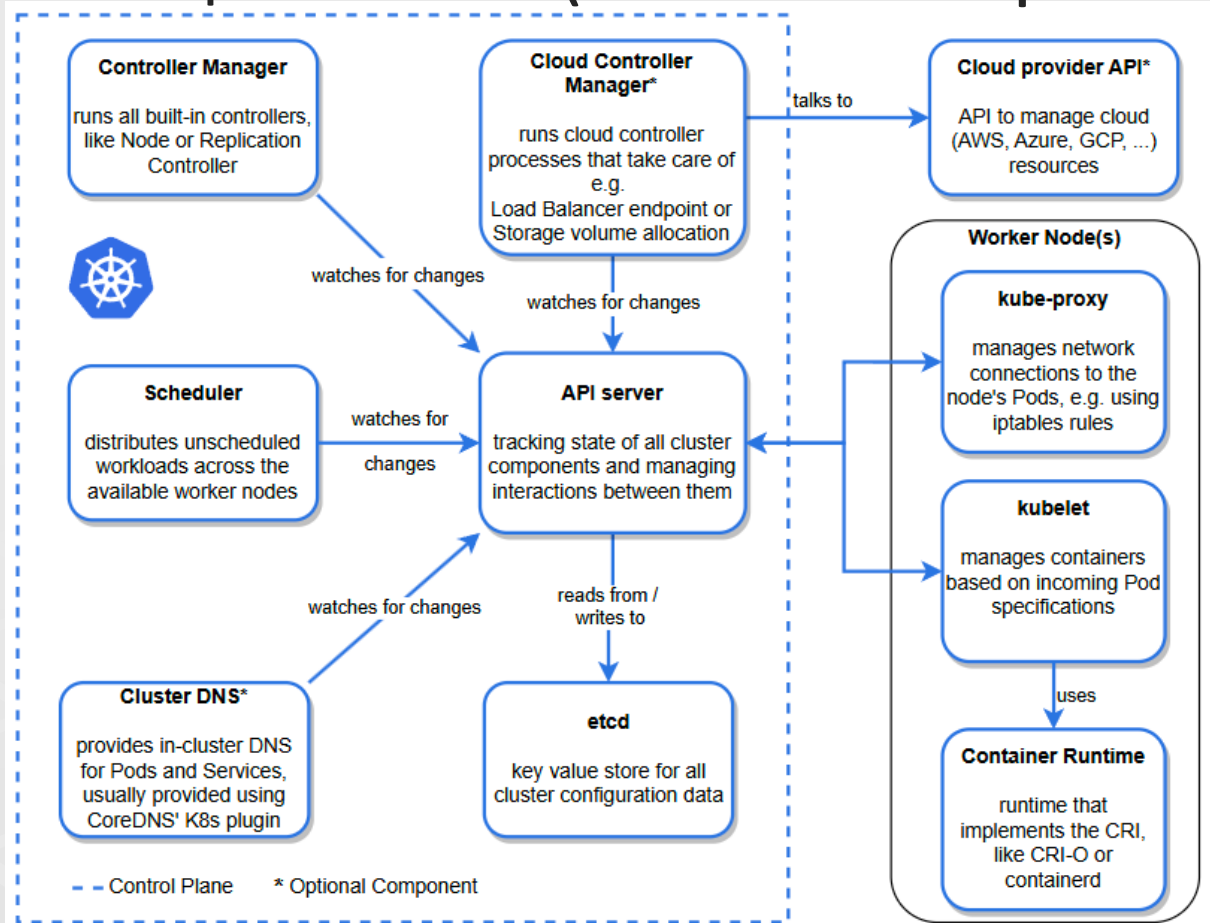
Container Runtime

Запускает и управляет жизненным циклом контейнеров.

kube-proxy

Обеспечивает сетевую маршрутизацию и балансировку нагрузки между сервисами и подами. Kube-proxy поддерживает связь между различными подами и сетевыми объектами, управляя правилами iptables или ipvs в зависимости от конфигурации.

Общая схема (с комментариями)





04


Декларативный подход k8s



Базовая основа k8s



Все в Kubernetes определяется с помощью текстовых файлов в формате YAML или JSON.



Платформа запускает образы OCI декларативным способом, настраивая все через эти файлы.

Этот подход используется также для настройки сетевых правил, аутентификации, авторизации (RBAC) и других аспектов

Изучив один синтаксис и структуру YAML/JSON, вы сможете управлять всеми аспектами Kubernetes

Создание и управление контейнерами в Docker

```
FROM alpine:3.15.4
RUN apk add --no-cache mysql
ENTRYPOINT ["/usr/bin/mysqld"]
```

```
FROM python:3.7
WORKDIR /myapp
COPY src/requirements.txt ./
RUN pip install -r requirements.txt
COPY src /myapp
CMD [ "python", "mysql-custom-client.py" ]
```

```
docker build -t my-mysql-server .
```

```
docker run -d --name mysql-server my-mysql-server
```

Создание и управление контейнерами в Kubernetes

```
apiVersion: v1
kind: Pod
metadata:
  name: mysql-pod
spec:
  containers:
  - name: mysql-server
    image: myregistry.com/mysql-server:v1.0
```

```
apiVersion: v1
kind: Pod
metadata:
  name: python-client-pod
spec:
  containers:
  - name: python-client
    image: myregistry.com/python-client:v1.0
    command: ['tail', '-f', '/dev/null']
```

```
kubectl apply -f mysql.yaml
```




Декларативный подход k8s

Kubernetes


Конфигурации
сохраняются в YAML-
файлах и описывают
желаемое состояние
системы.

автоматически
поддерживает и
стабилизирует
состояние кластера на
основе этих описаний


- ☐ Kubernetes сам решает, где и как запускать контейнеры.
 - ☐ Кластер поддерживает заданное состояние даже при сбоях.
 - ☐ Легко масштабировать и управлять большим количеством контейнеров.
- 

Базовая терминология

1. CNI (Container Networking Interface) и CSI (Container Storage Interface) – сетевой интерфейс контейнеров и интерфейс хранилища для контейнеров соответственно. Позволяют подключать модули Pod (с контейнерами), работающие в Kubernetes, к сетям и хранилищам.
2. Контейнер (Container) – образ Docker или OCI (Open Container Initiative), который обычно запускает приложение.
3. Плоскость управления (Control plane) – мозг кластера Kubernetes, осуществляющий планирование контейнеров и управляющий всеми объектами Kubernetes (которые иногда называют мастер-объектами).
4. Набор демонов (DaemonSet) – аналог разворачивания (Deployment), но выполняется на каждом узле кластера.
5. Разворачивание (Deployment) – набор модулей, которыми управляет Kubernetes.
6. kubectl – инструмент командной строки для взаимодействия с панелью управления Kubernetes.
7. kubelet – агент Kubernetes, работающий на узлах кластера. Обеспечивает поддержку плоскости управления.
8. Узел (Node) – машина, на которой запущен процесс kubelet.
9. OCI (Open Container Initiative) – общий формат образа для создания выполняемых автономных приложений. Также называется образами Docker.
10. Pod (модуль) – объект Kubernetes, инкапсулирующий контейнер.



ДЗ будет в канале telegram.





А на сегодня все.

