

Архитектура вычислительных систем

Лекция 10. Слой Virtualisation



Artem Beresnev

t.me/ITSMDao

t.me/ITSMDaoChat

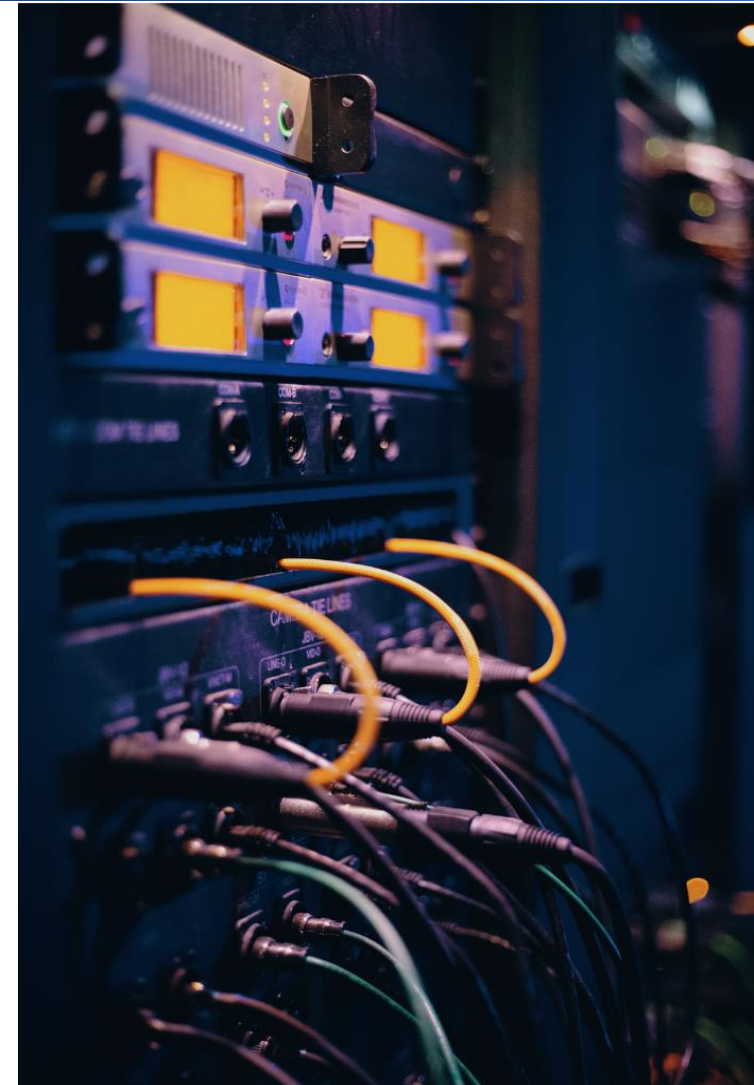
План

- Задачи виртуализации
- Виды виртуализации
- Примеры использования
- Контейнеризация
- Особенности использования

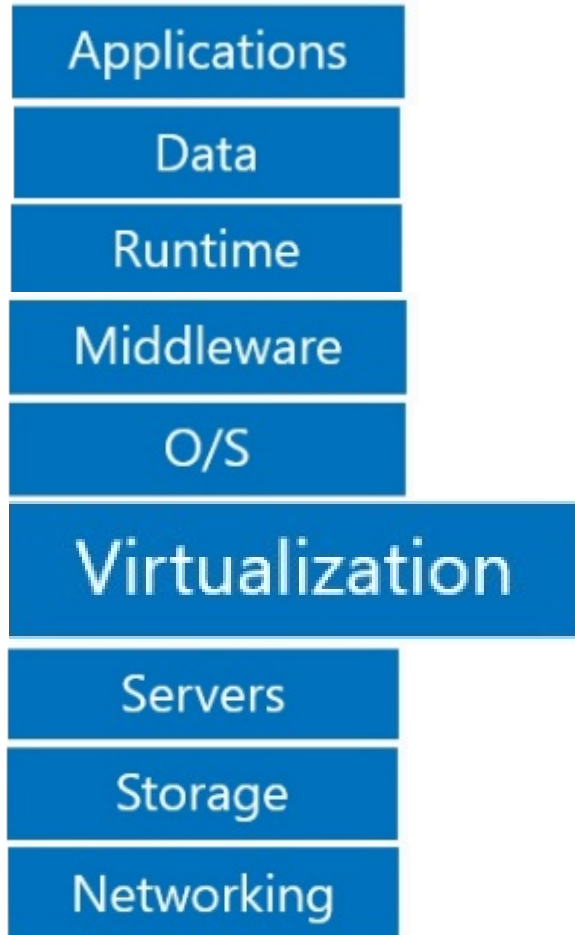


Слои ИТ-инфраструктуры

Что там было про Virtualisation?



ИТ-инфраструктура

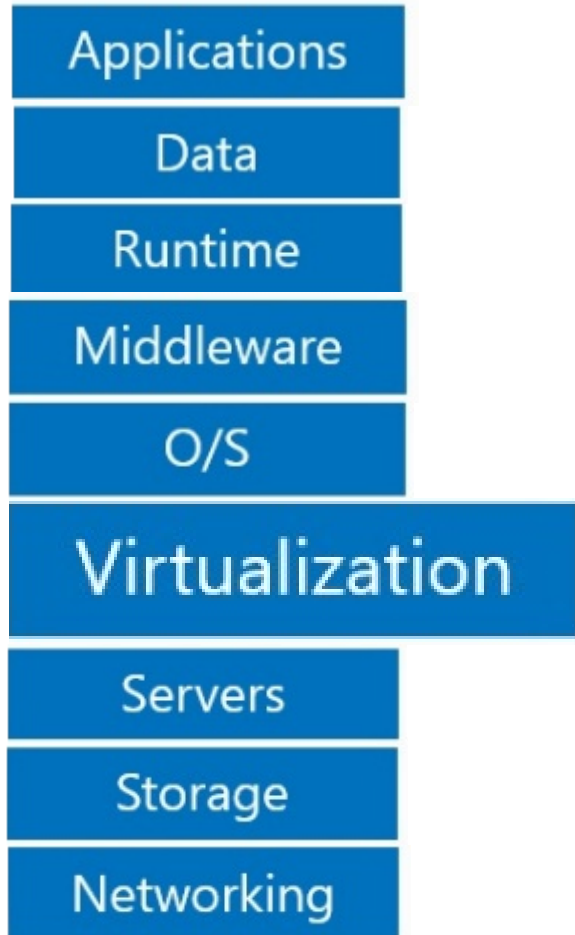


Virtualization – слой виртуализации и контейнеризации.

Этот слой абстрагирует физические ресурсы и предоставляет их в виде виртуальных машин или контейнеров.



ИТ-инфраструктура



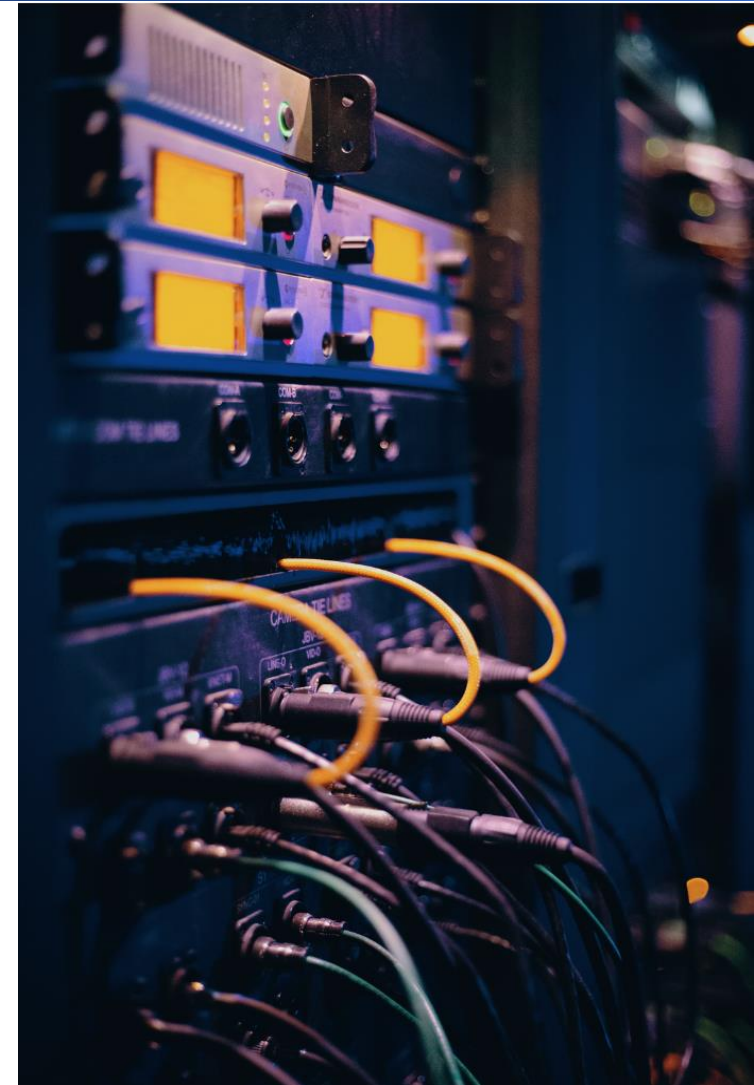
Примеры:

- VMware vSphere
- Hyper-V
- KVM (Kernel-based Virtual Machine)
- VirtualBox
- Parallels
- Docker \ podman
- K8



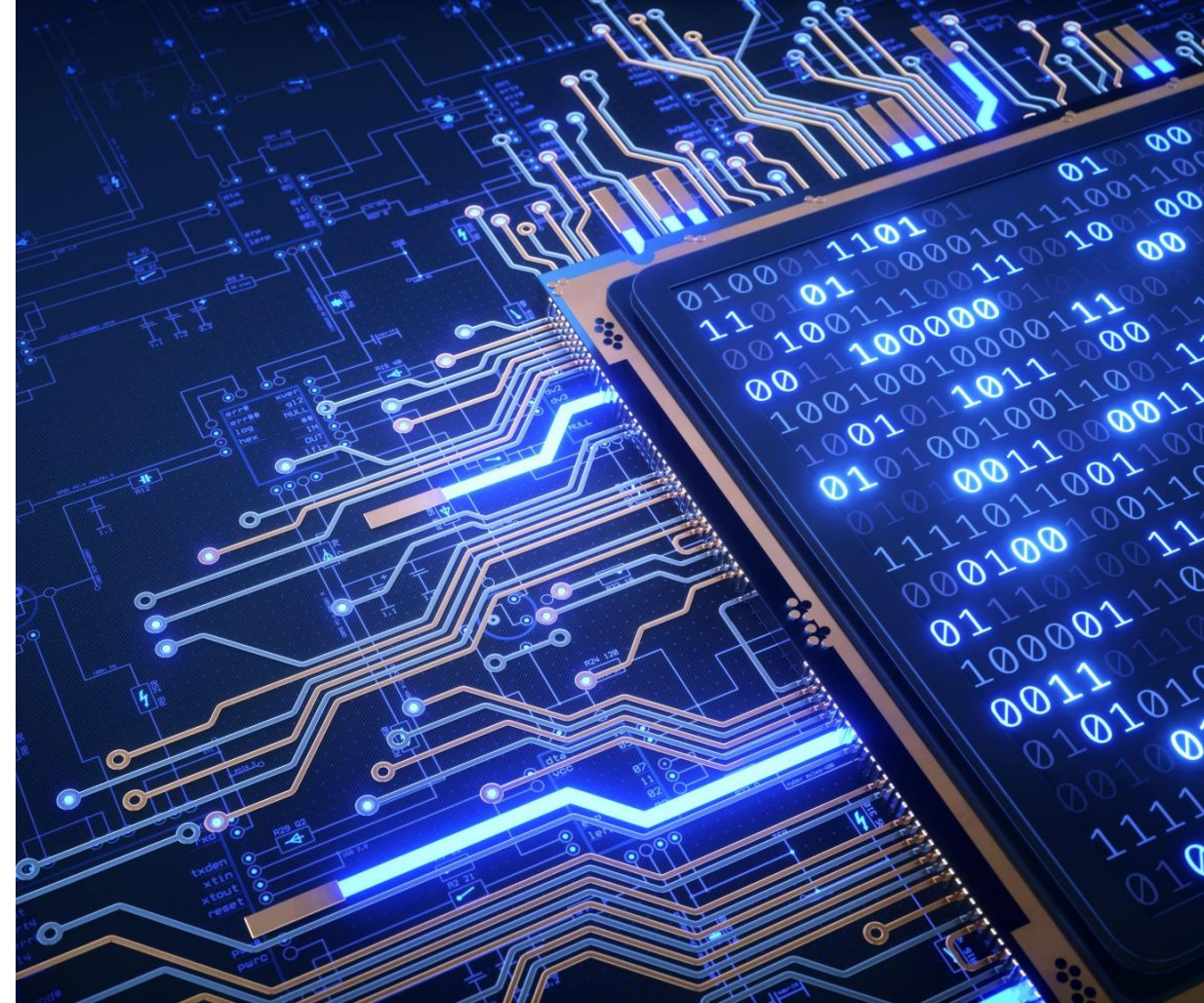
Что такое виртуализация?

Какие задачи решает виртуализация? Какой она бывает?

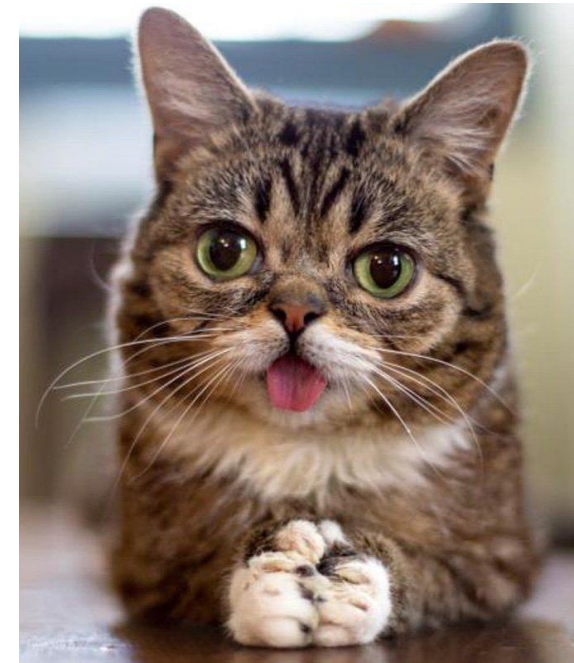
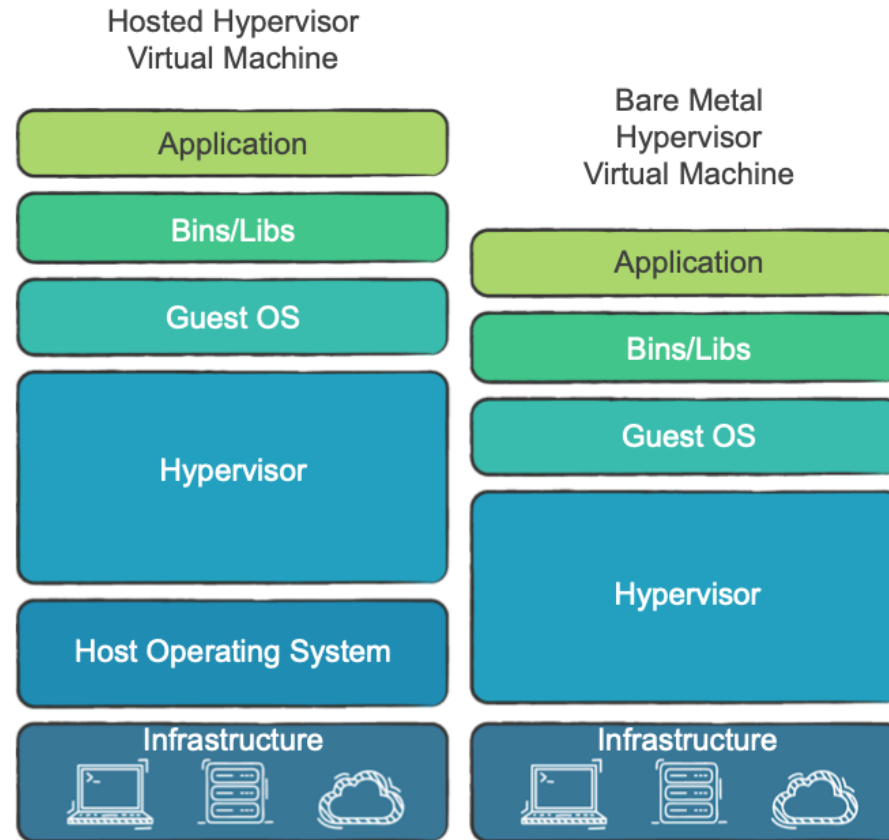


Виртуализация

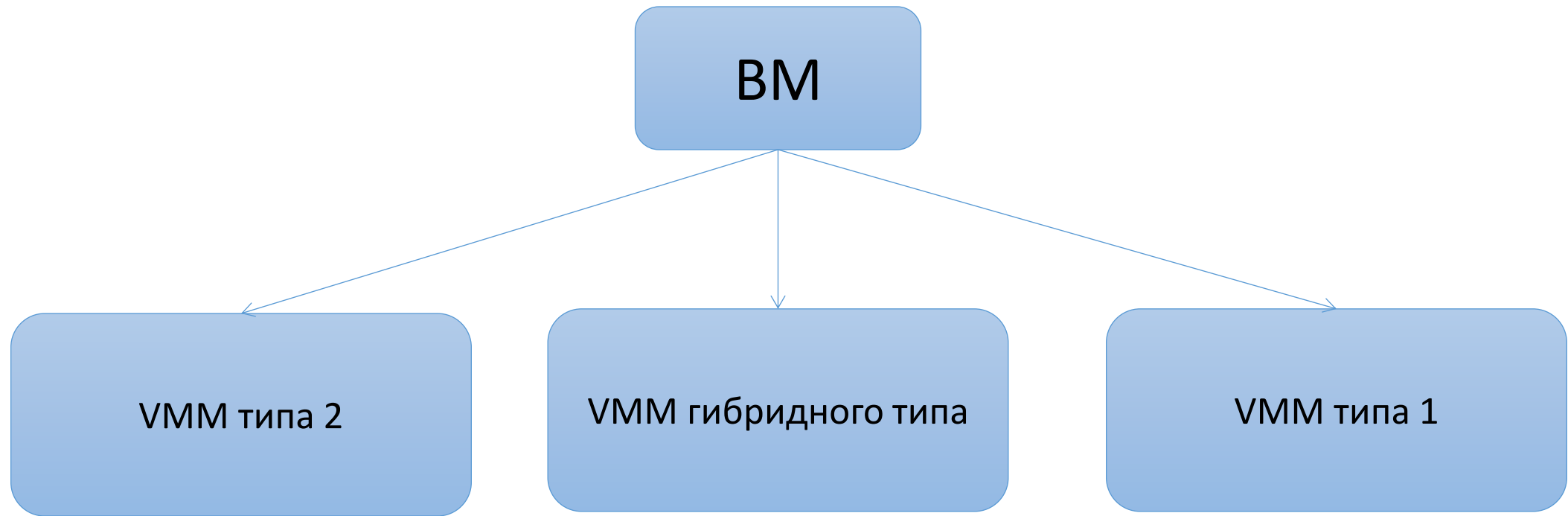
- **Виртуализация** — предоставление набора вычислительных ресурсов или их логического объединения, абстрагированное от аппаратной реализации, и обеспечивающее при этом логическую изоляцию вычислительных процессов, выполняемых на одном физическом ресурсе
- Виртуальная машина (VM, от англ. virtual machine) — программная и/или аппаратная система, эмулирующая аппаратное обеспечение некоторой платформы (guest — гостевая платформа) и исполняющая программы для guest-платформы на host-платформе (host — хост-платформа, платформа-хозяин) или виртуализирующая некоторую платформу и создающая на ней среды, изолирующие друг от друга программы и даже операционные системы



Виртуализация ОС

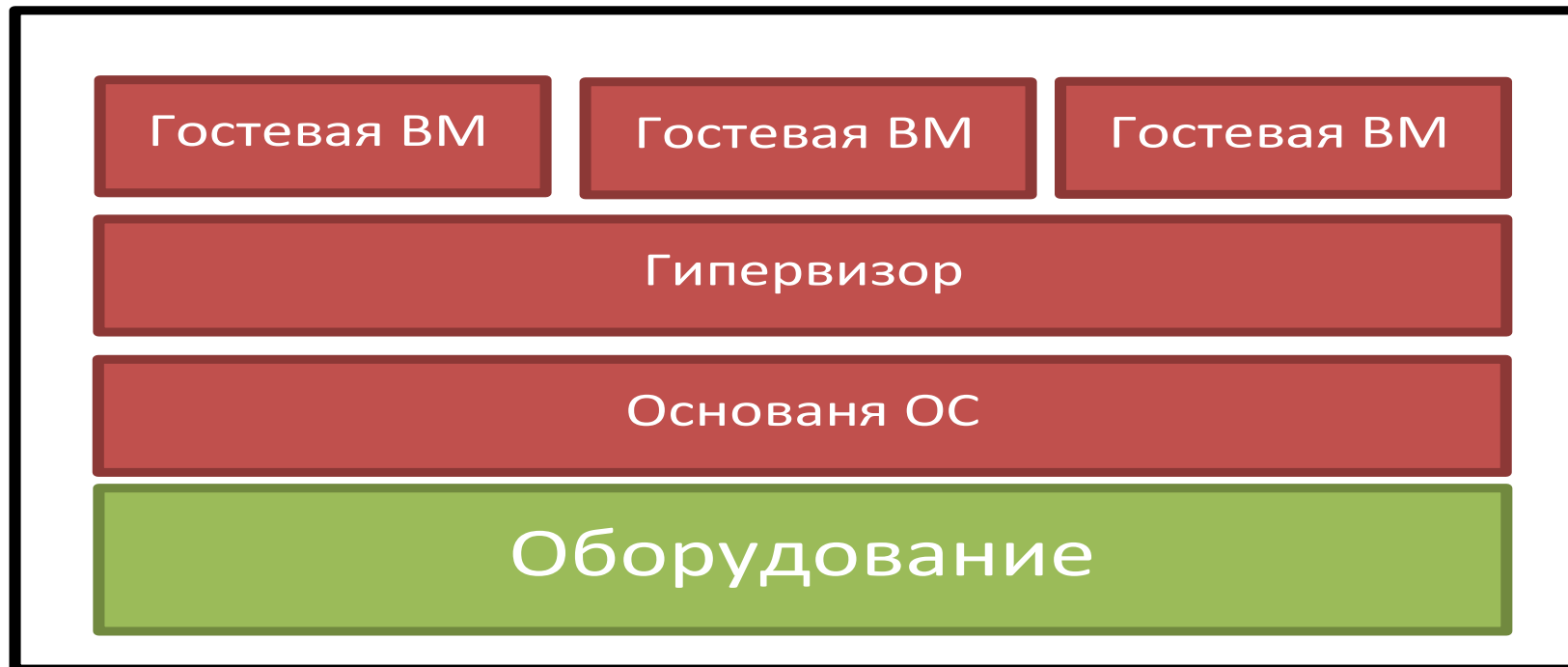


Классификация архитектур виртуализации

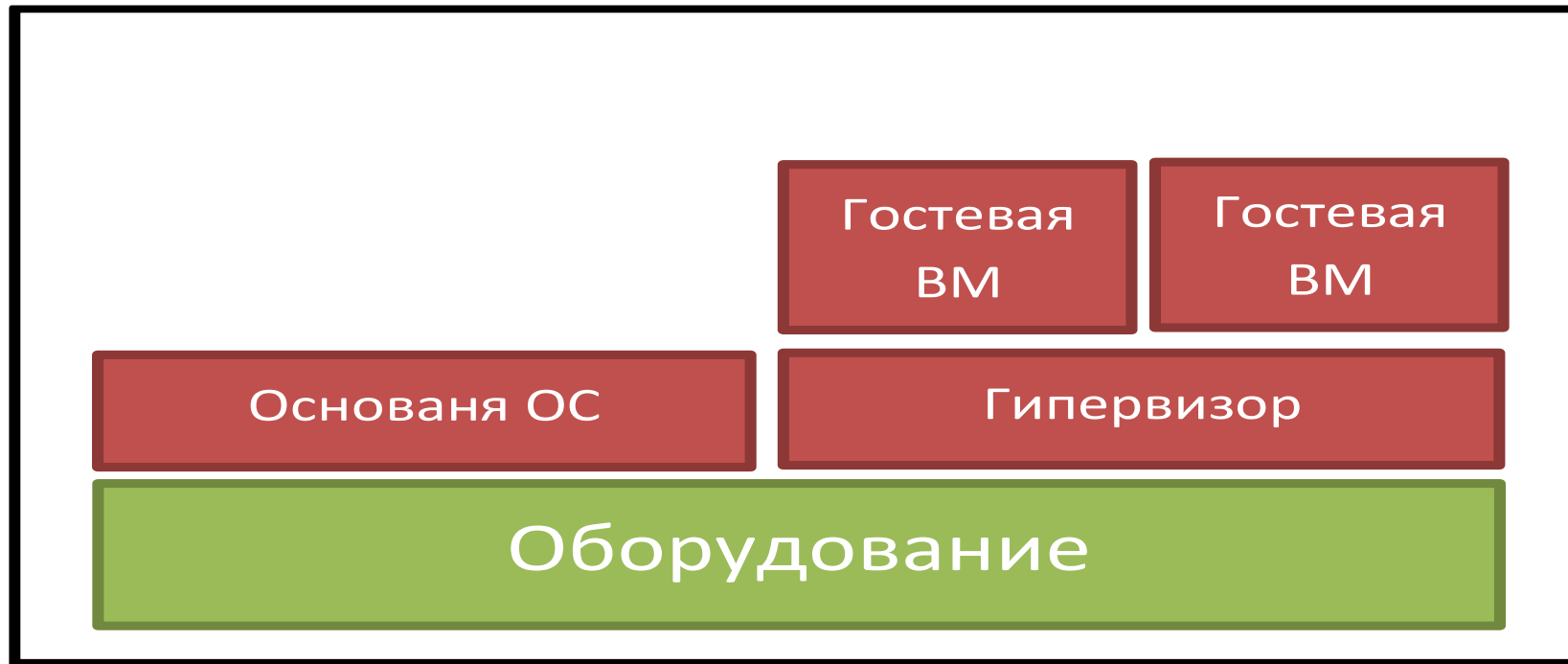


VMM - Virtual Machin Manager, или гипервизор

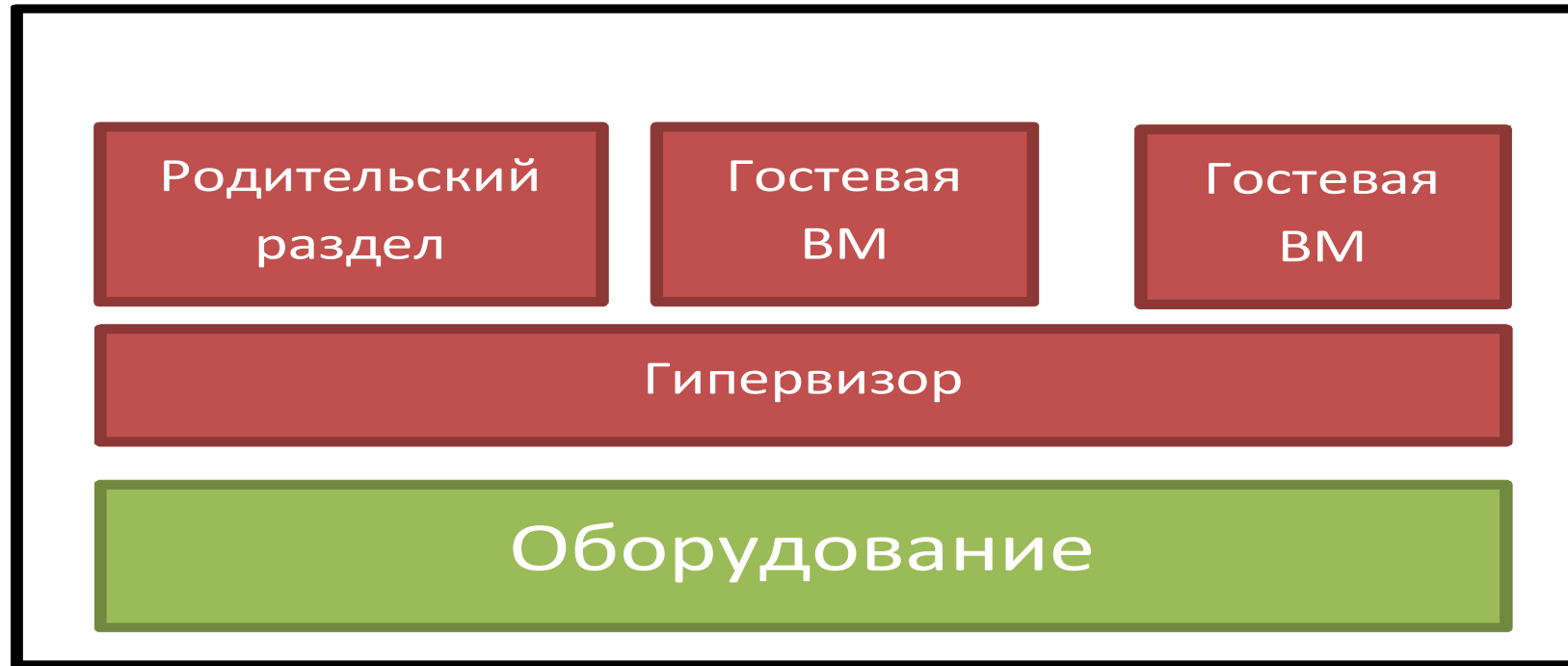
VMM типа 2



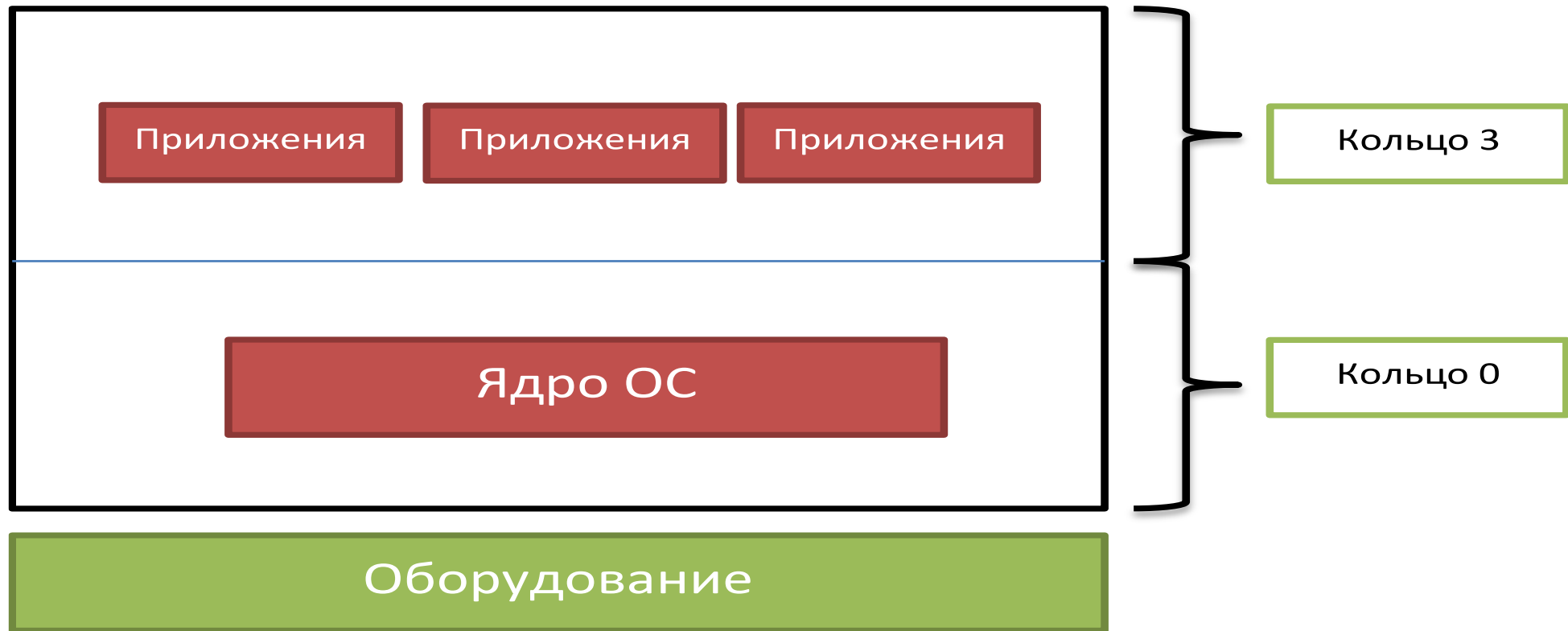
VMM гибридного типа



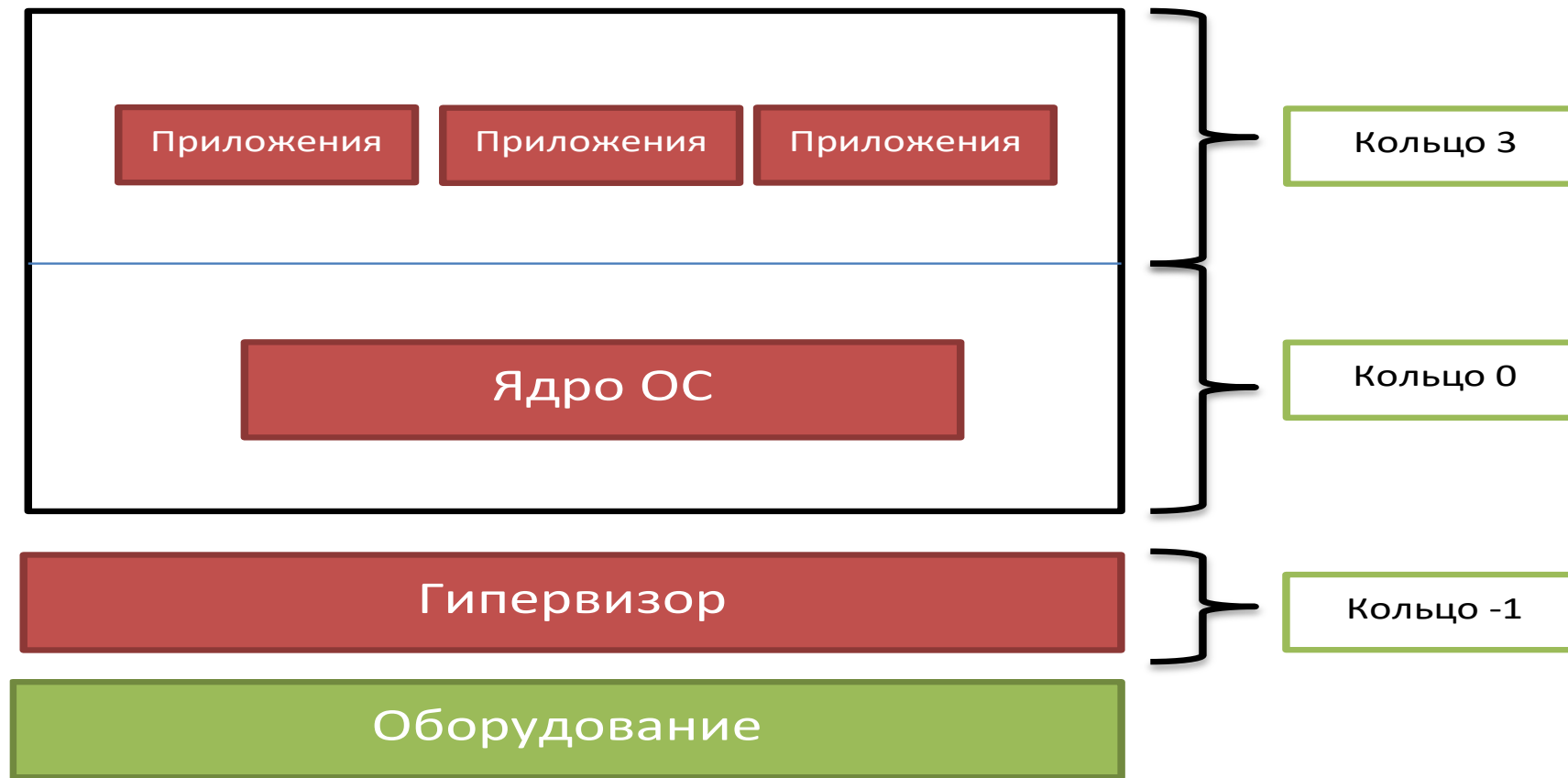
VMM типа 1



Обобщенная архитектура



Аппаратная виртуализация



Intel-VT



AMD-V

Основные термины

Гипервизор

Стек виртуализации

Раздел

Эмулируемый раздел

Просветленный раздел

Гипервизор

Программный интерфейс между разделами и оборудованием. Отвечает за разделение ресурсов, изоляцию разделов и взаимодействие разделов.



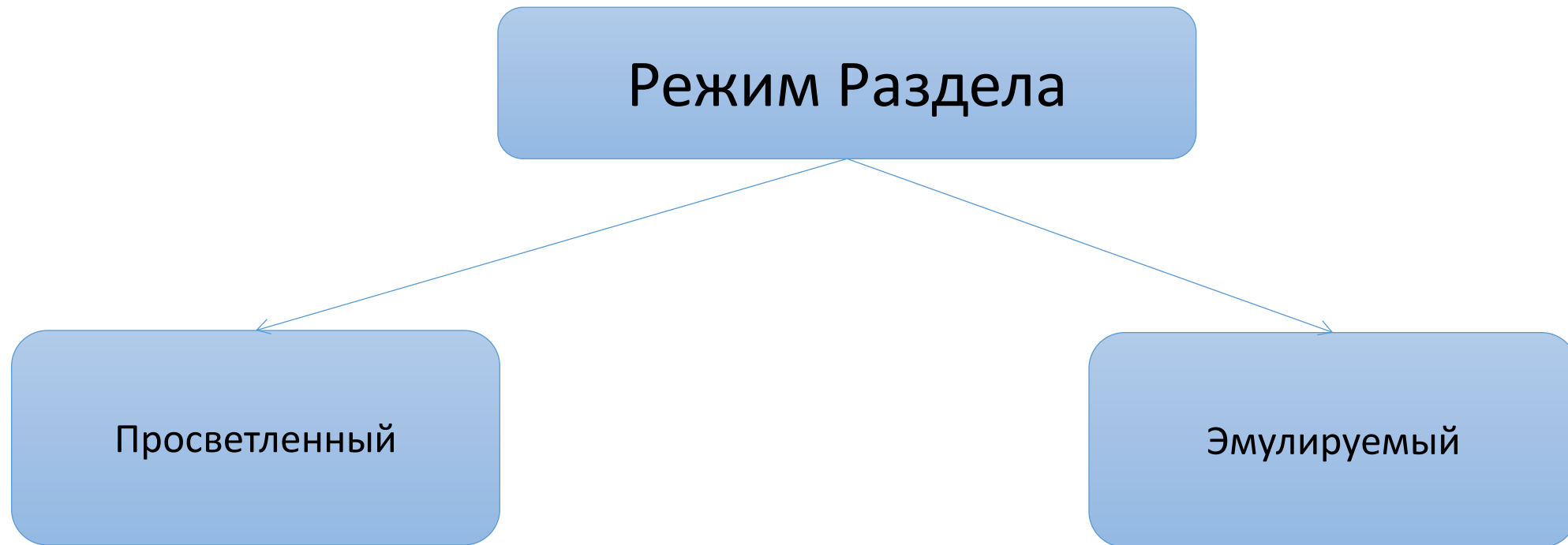
Стек виртуализации

Исполняемый в родительском разделе набор компонентов по управлению дочерними разделами и гипервизором

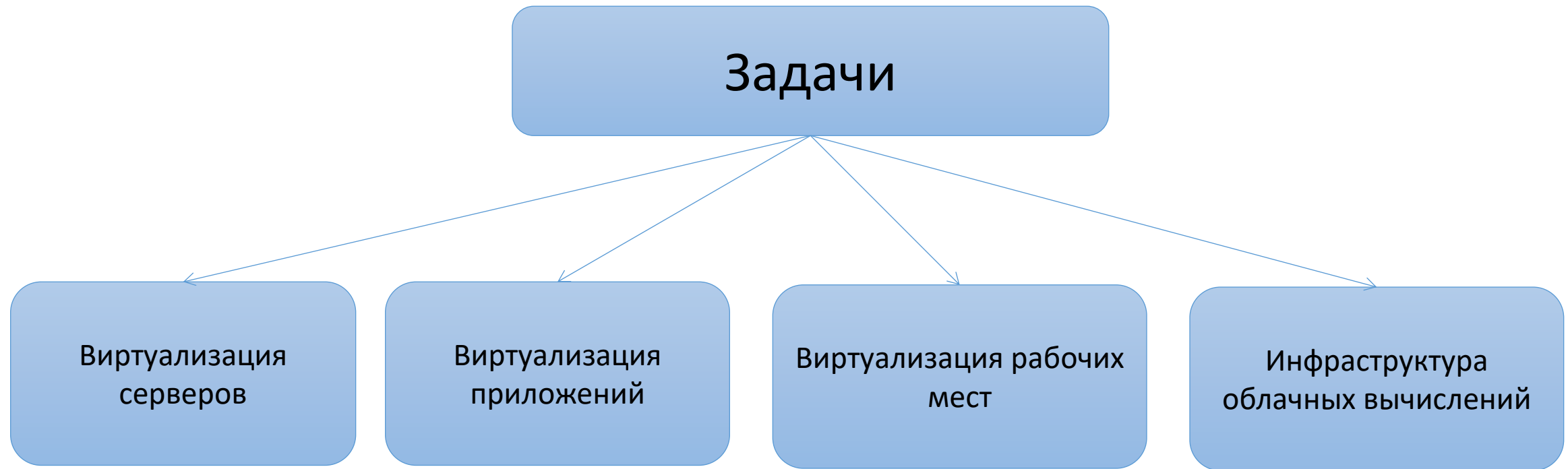
Для Hyper-V: VSP, VMBus, VDI

Раздел

Работающая в среде виртуализации VM Раздел
м б родительским (управляющим) или гостевым
(дочерним)



Задачи виртуализации



Виртуализация серверов и рабочих мест



- ❑ VMware ESX Server (ESXi)
- ❑ Hyper-V
- ❑ XEN

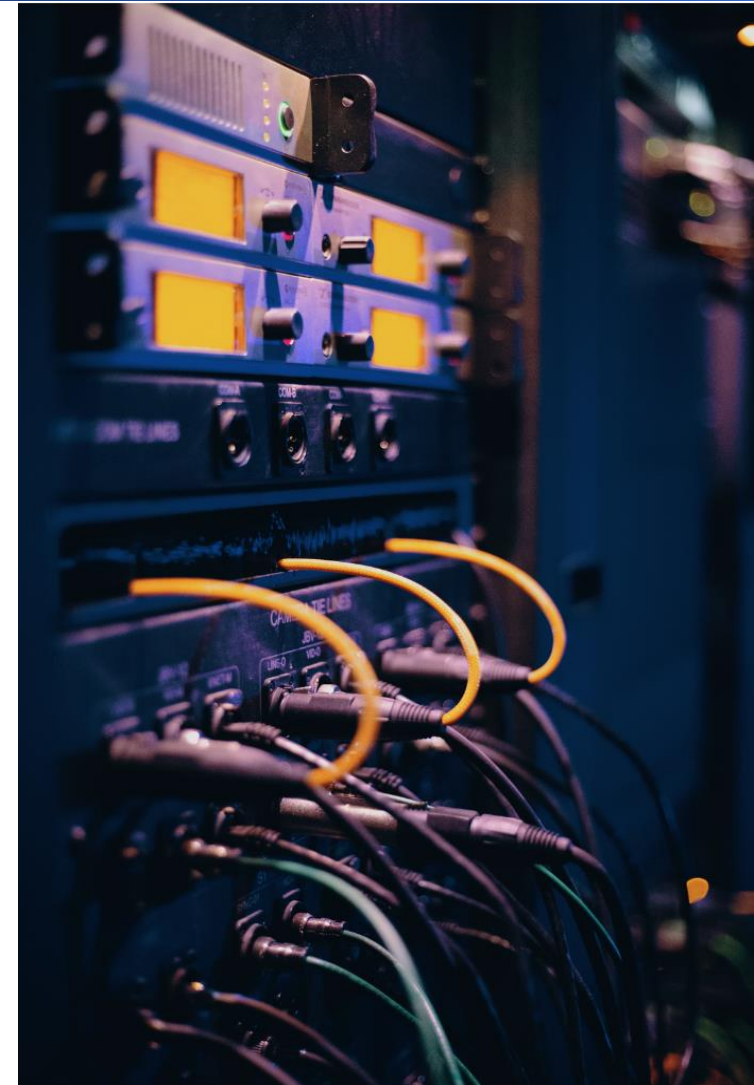
Виртуализация приложений



- ❑ Parallels Virtuozzo Containers
- ❑ VMware ThinApp
- ❑ Microsoft Application Virtualization (App-V)
- ❑ Citrix XenApp

Что такое контейнеризация?

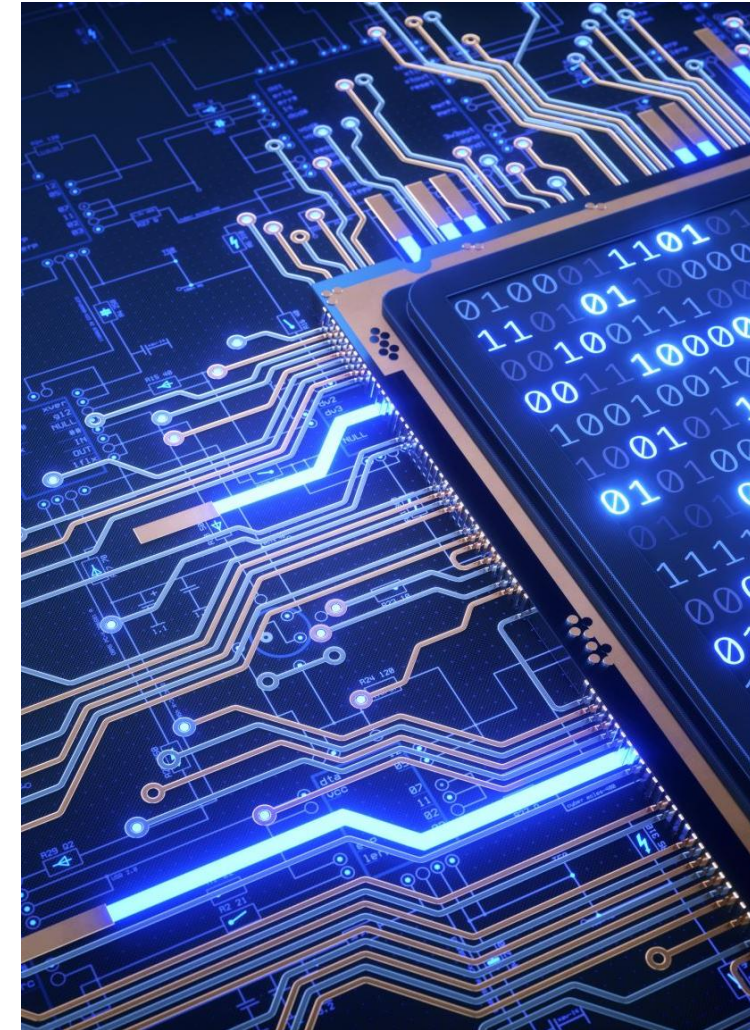
Какие задачи решает контейнеризация? Чем она отличается от виртуализации?



Контейнеризация

Контейнеризация (виртуализация на уровне операционной системы, контейнерная виртуализация, зонная виртуализация) — метод виртуализации, при котором ядро операционной системы поддерживает несколько изолированных экземпляров пространства пользователя вместо одного.

Эти экземпляры (обычно называемые контейнерами или зонами) с точки зрения выполняемых в них процессов идентичны отдельному экземпляру операционной системы



Контейнеры. Что под капотом?



Основные технологии Namespaces и Control Groups



Namespaces

Namespaces PID ограничивает процессы своим деревом процессов со своим корнем (PID 1)
Networking Namespace позволяет ограничить/изолировать сеть
Mount — это ограничение по файловой системе
User — ограничение по юзерам



Control Groups

Управляют ресурсами для контейнера.
Control Groups: Memory, CPU, IOPS, Network



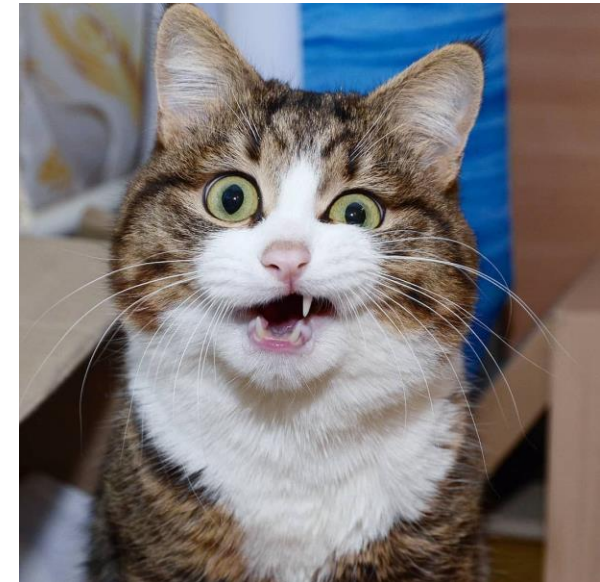
Linux Containers или LXC

Docker

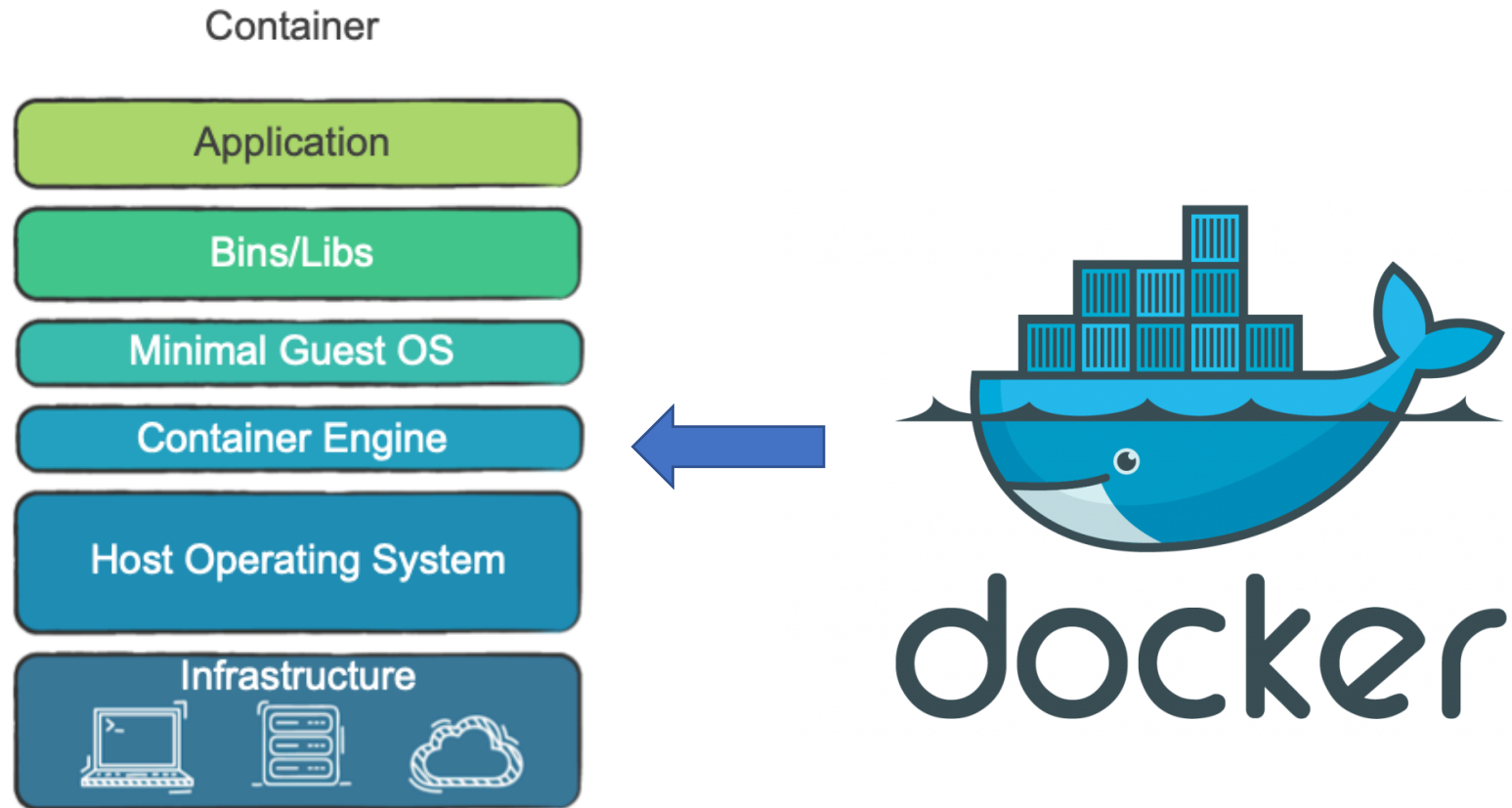
When other
people
describe
Docker



When I
use
Docker

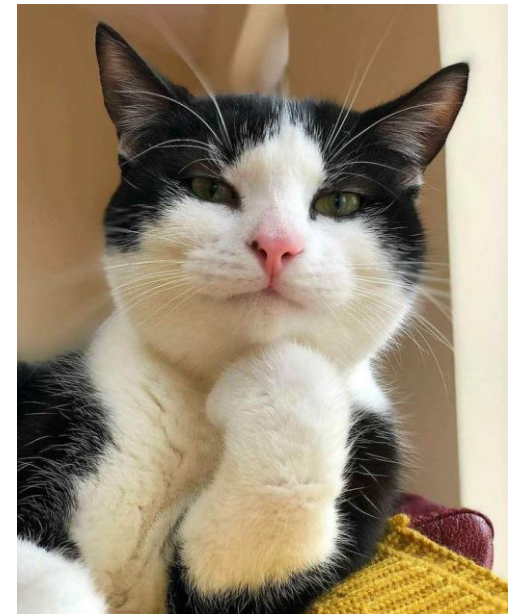


Контейнеризация



Преимущества контейнеризации

- Легкость: минимальные сопутствующие издержки
- Скорость: для запуска приложений требуется меньше времени
- Изоляция: процессы одного контейнера полностью (на самом деле нет) изолированы от общей инфраструктуры и других контейнеров. Изоляция позволяет делать независимое обновление, а также точно вносить изменения
- Инкапсуляция: необходимая для запуска приложений инфраструктура (файлы, настройки и зависимости) собирается в отдельный образ-капсулу. Его можно легко перенести из одной среды в другую
- Повторное использование: уже собранные готовые образы можно использовать повторно. Поскольку это ускоряет время запуска, для часто используемых приложений виртуальный контейнер — это оптимальный вариант



Недостатки контейнеризации

- Сложность в управлении: каждый контейнер модерируется по отдельности. Для одновременной работы с большим количеством экземпляров или управления взаимосвязанными контейнерами потребуется оркестратор
- Контейнеризация Linux: большинство технологий контейнеризации создавались специально под Linux
- Уровень изоляции: контейнеры слабее изолированы, поскольку их изоляцию обеспечивает ядро хост-системы. Поэтому при использовании контейнеризации стоит уделять больше внимания настройкам безопасности
- Новизна: контейнеризация — относительно новая (нет) технология



Принципы использования контейнеризации

- Один контейнер обслуживает одну функцию
- Статичность образа Не стоит вносить изменения в контейнер, если образ уже сформирован Это может привести к полному уничтожению или потере части данных Также статичность контейнера позволяет параллельно проводить тестирования в CI/CD системах
- Лимитирование ресурсов Чтобы процессы в контейнерах работали максимально эффективно, желательно настроить трекинг лимитов (CPU и RAM) Это позволит отслеживать состояние ресурсов и вовремя реагировать на их избыточное потребление



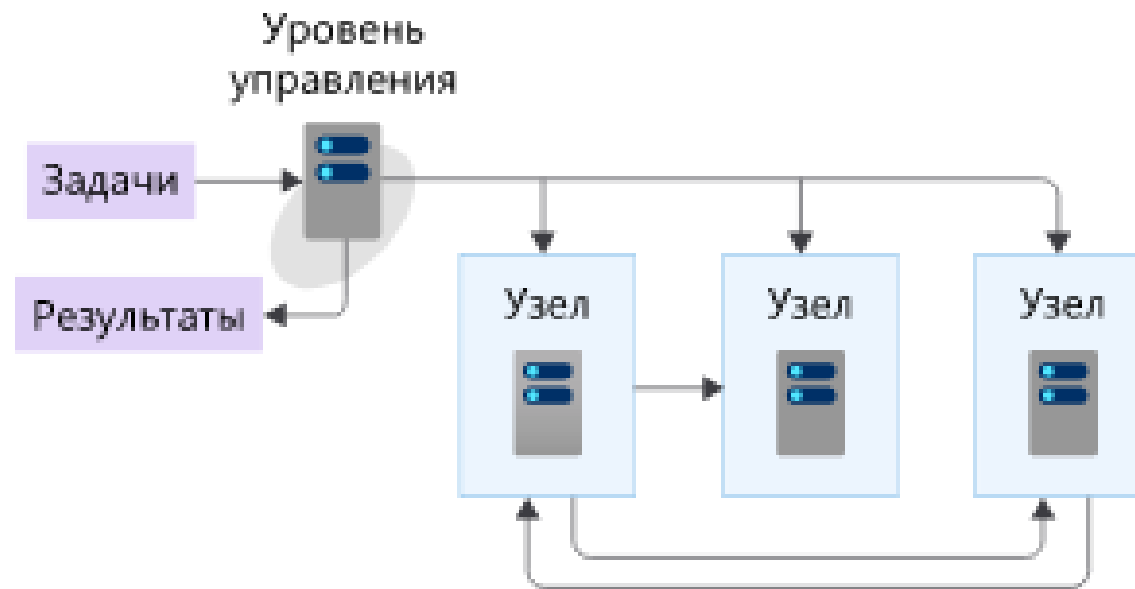
Оркестрация



kubernetes

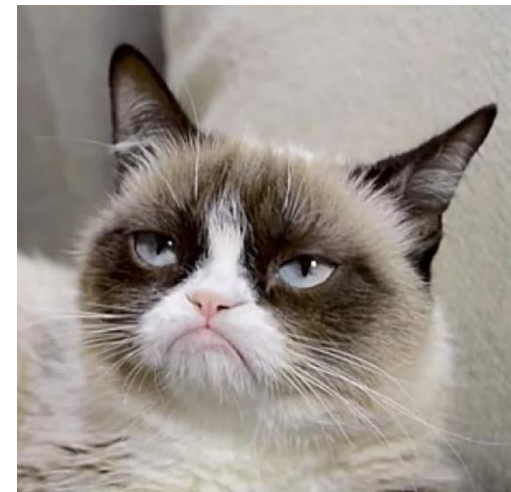


OPENSIFT



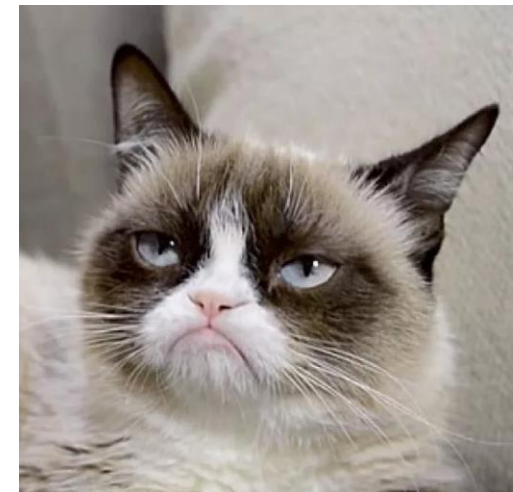
Задачи оркестратора контейнеров

- Развёртывание контейнеров с гибким управлением процессом развертывания.
- Настройка и поддержание балансировки нагрузки;
- Сетевое подключение;
- Самостоятельное восстановление контейнеров. Например, перезапуск контейнеров, в которых произошел сбой, или замена контейнеров.
- Масштабирование количества развернутых контейнеров приложений динамически в зависимости от спроса.
- Автоматические последовательные обновления и откат контейнеров;
- Управление хранилищем;
- Управление сетевым трафиком;
- Хранение конфиденциальных данных, таких как пароли, и управление ими.

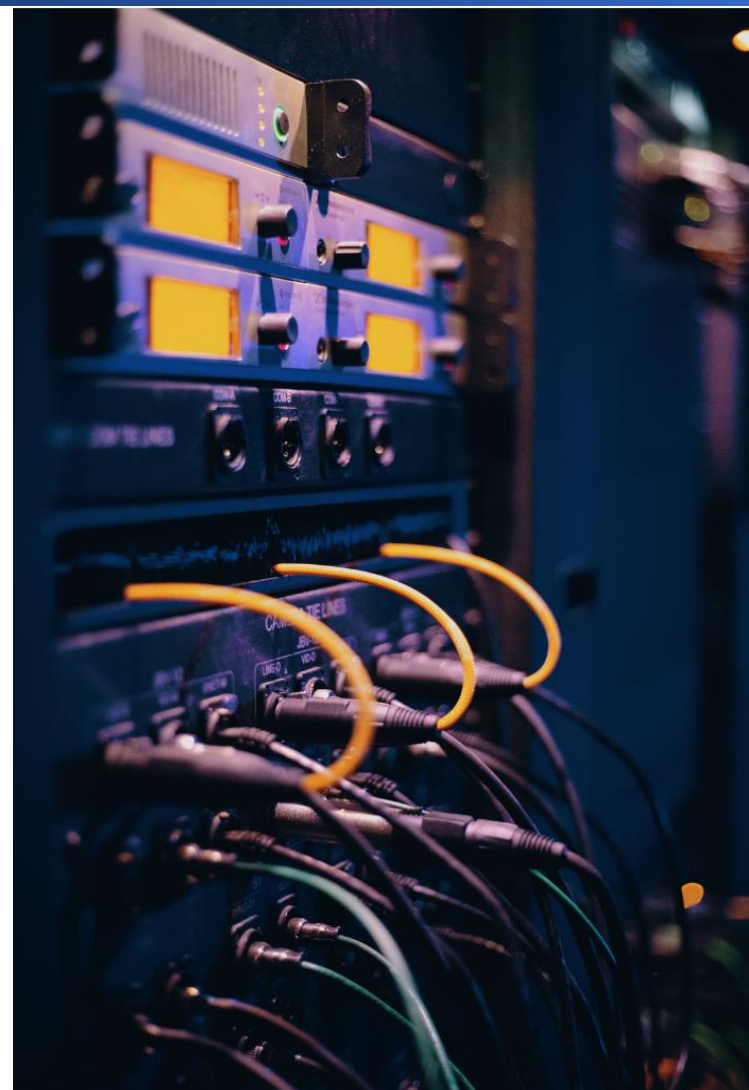


Примеры в студию!

- Установка:
<https://docs.docker.com/engine/install/debian/#install-using-the-repository>
- Запуск контейнера:
- `docker run --name mynginx -d -p 80:80 nginx`
- Docker ps stop start kill
- Осторожно! Volume и секреты.



ИТОГИ



- Виртуализация бывает разная
- Контейнеризация эффективней, но имеет ограничения