Home

Информационный поиск

Лекция 5. Компьютерные сети и докер

Telegram

GitHub

Ника Зыкова
2023/09/27

Виртуальное окружение

Можно ставить все библиотеки в одно место, но иногда их зависимости конфликтуют: например, sklearn хочет одну версию numpy, а pandas - другую.

Отсюда возникает идея хранить зависимости проектов в разных местах - это и есть **виртуальное окружение**.

Home | Recent Tabs | Current Page

You can type something here...

Python и виртуальные окружения

Менеджер окружений - это инструмент, позволяющий создавать и контролировать виртуальные окружения.

- venv
- pyvenv
- virtualenv (+virtualenvwrapper)
- pipenv
- *poetry*
- conda

Менеджер пакетов позволяет устанавливать и удалять пакеты, контролировать их версии.

- pip
- poetry
- *conda*

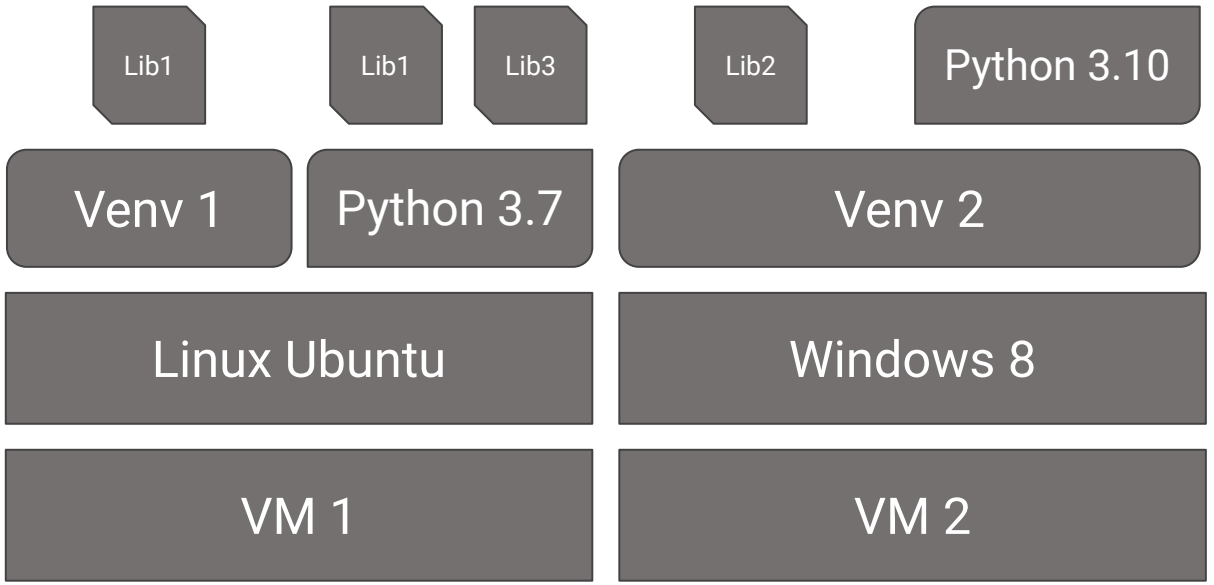
И есть еще **pyenv**:

- не является менеджером пакетов или окружений
- позволяет удобно управлять несколькими версиями питона в одной системе

Виртуальная машина

Часто несовместимость оказывается глобальнее, чем два пакета или две версии питона, которые не дружат между собой.

Хочется уметь отделить и ОС тоже - отсюда возникает идея **виртуальной машины**.



Home | Recent Tabs | Current Page

You can type something here...

Виртуальная машина: пример

Что можно использовать?

Виртуальные машины - полноценное ПО для работы с VM.

Удобно для экспериментов и тестирования.

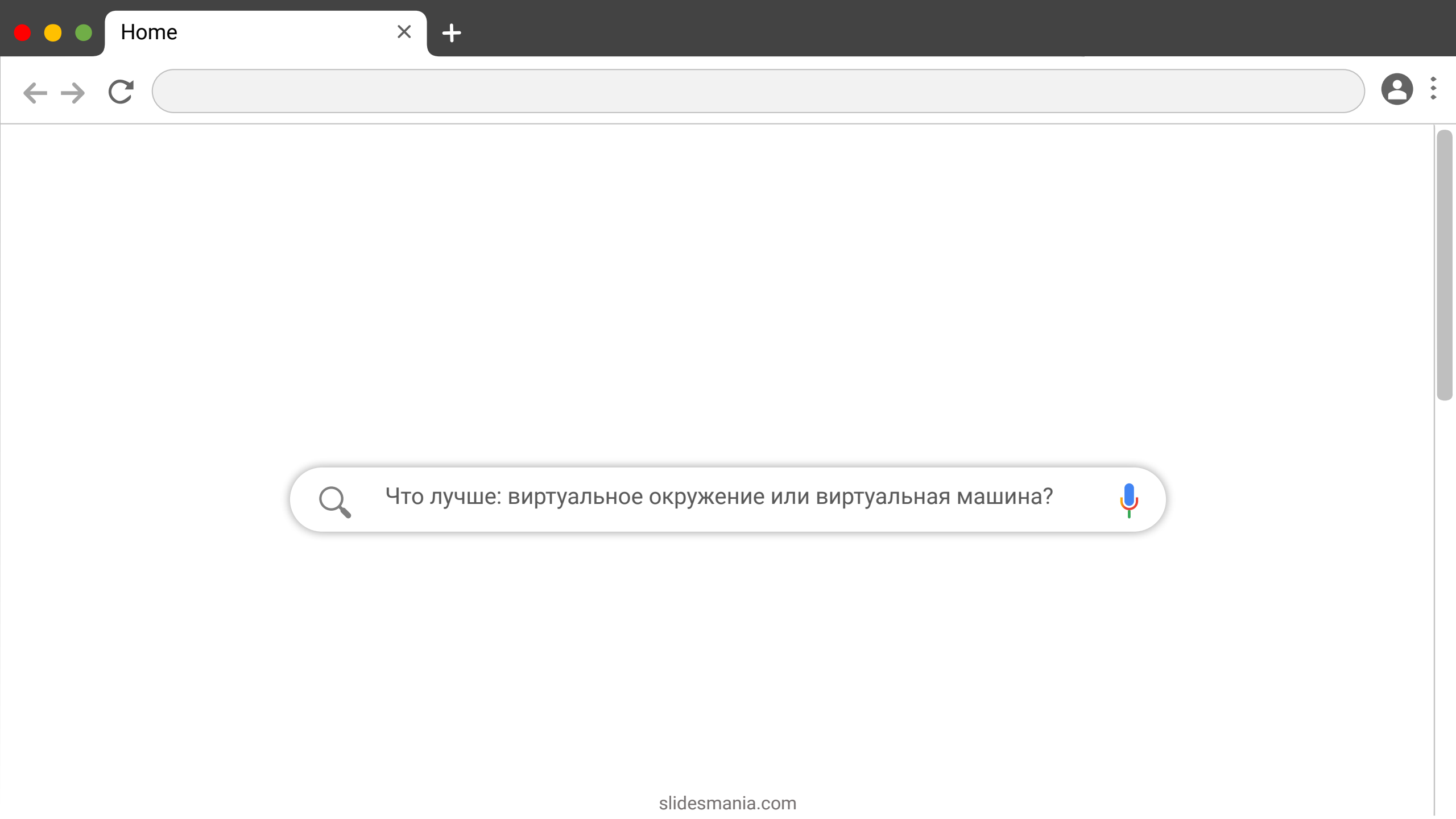
- Hyper-V
- VirtualBox
- VMware

Вторая система - установка необходимой ОС как второй на вашем устройстве.

- Почти невозможно на Mac
- Как вторая рабочая система, не для экспериментов
- Для переключения нужно время, равное времени перезагрузки

Windows Subsystem for Linux (WSL) - утилита для запуска Linux-приложений под Windows.

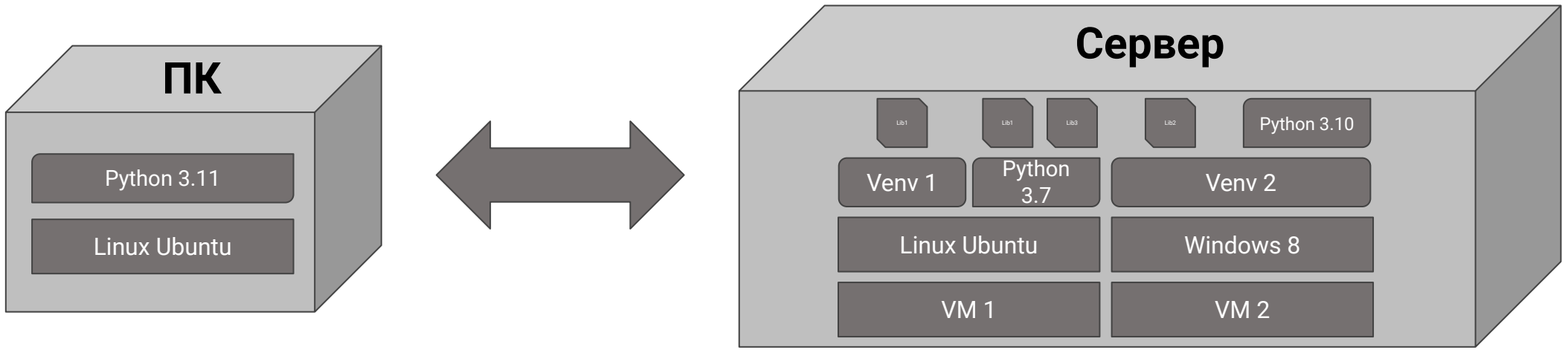
- Только для Windows
- Позволяет работать без тяжелых VM решений
- Дает возможность работать с докером под Windows!



Сервер

А теперь мы не хотим, чтобы все описанное ранее существовало на нашем компьютере: например, у нас мало памяти или нет видеокарты.

Для этого мы подключаемся по сети к удаленной машине - **серверу**.



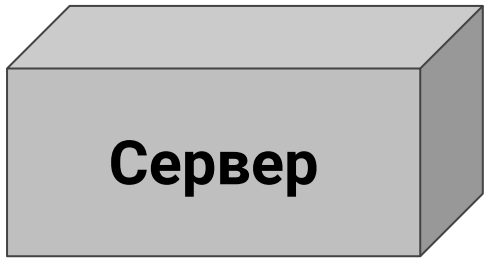
В чем разница?

Сервер - отдельная материальная (!) сущность, другой компьютер где-то далеко.

Виртуальная машина - виртуальный другой компьютер, эмуляция другой ОС на вашей.

Виртуальное окружение - изолированная часть системы со своими зависимостями.

Одно можно сложить в другое, если двигаться снизу вверх (или в картинках справа налево).



Сети: что это?

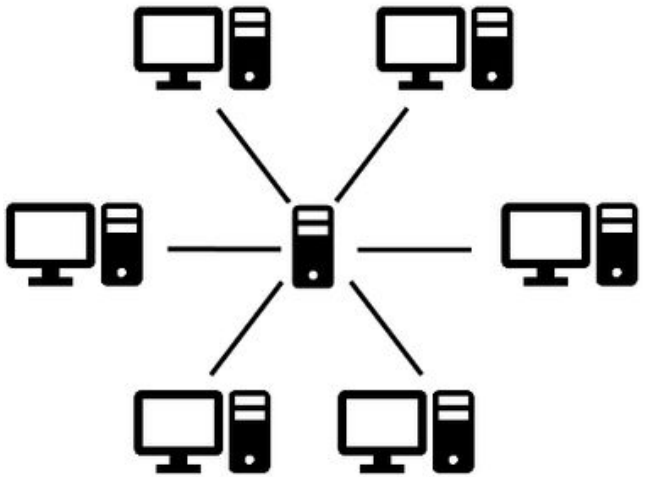
Компьютерная сеть - это набор вычислительных устройств, соединенных в единую систему так, что это дает возможность обмена информацией и совместного использования ресурсов.

Примеры использования сетей:

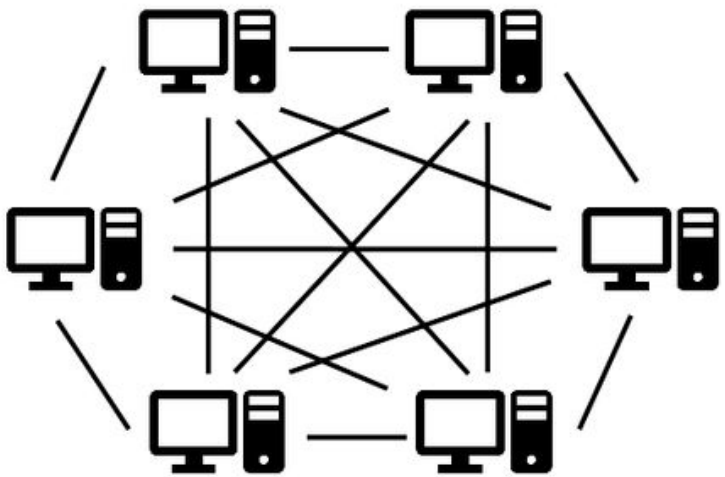
- Игра по bluetooth на телефоне
- Перенос файлов напрямую с одного домашнего компьютера на другой
- Управление умным домом
- Разрешение доступа к ресурсу только для абонентов сети (например, РУЗ и Wi-Fi ВШЭ)
- Запуск программ на сервере
- И вообще интернет в целом :)

Какие бывают сети?

Способов классифицировать сети множество. Один из них - по тому, как глобально организуется связь в сети.



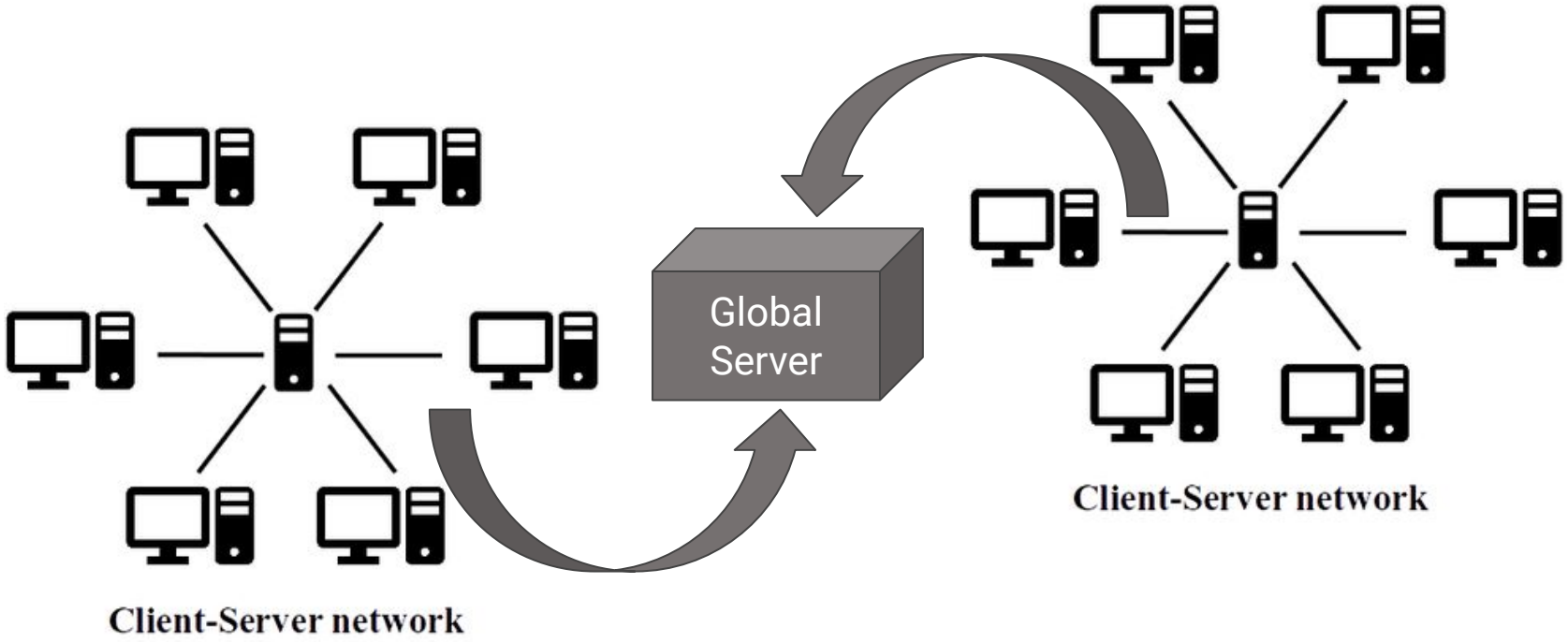
Client-Server network



P2P network

Общение устройств в сети

В сети вида клиент-сервер коммуникацию между абонентами можно представить в виде дерева. Но остается вопрос, как устройства узнаю друг друга.



IP-адрес и порты

В пределах сети для идентификации устройства используется **IP-адрес** - это 4 числа по 1 байту (от 0 до 255), записанные через точку.

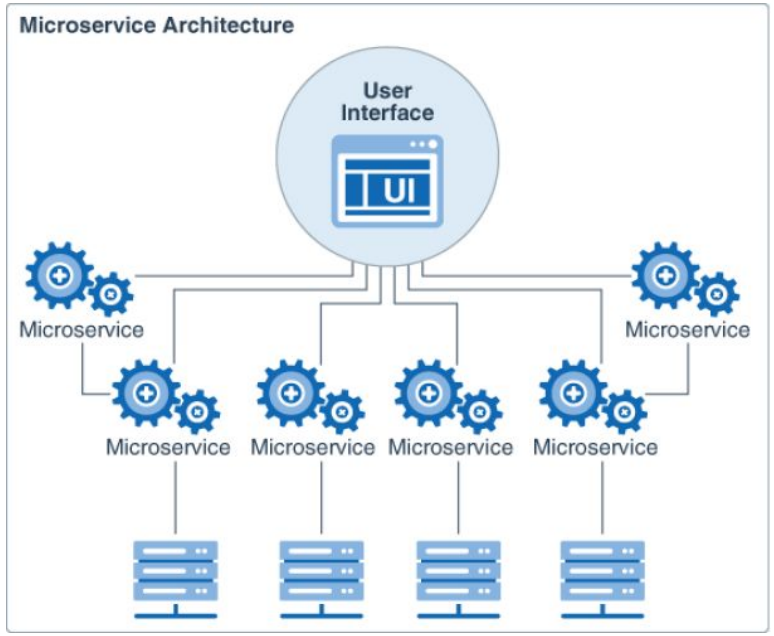
Когда устройство определено, для коммуникации с конкретным приложением на этом устройстве необходим номер **сетевого порта** - 16-битное число, позволяющее идентифицировать получателя внутри одного IP-адреса.

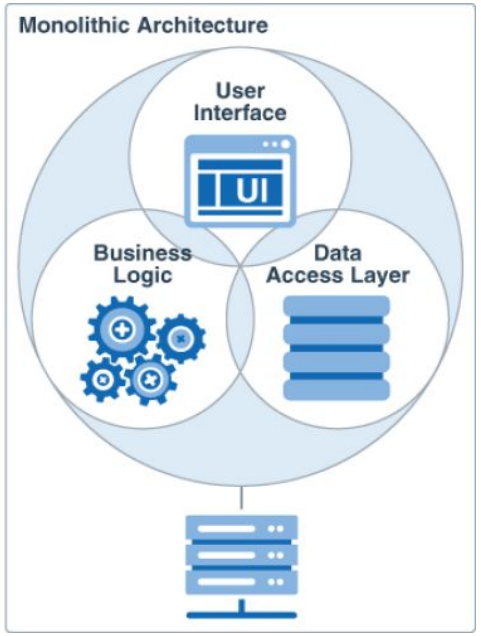
Примеры:

- 8888: Jupyter Notebook
- 22: ssh
- 80: http

Что нам это дает?

Легкая настройка коммуникацию между многими устройствами дала нам возможность развивать **микросервисную архитектуру** взамен **монолитам**, популярным ранее.





Монолит VS микросервис

Монолит

+ Быстрая разработка и развертывание
+ Производительность
+ Легкое тестирование и отладка

- Усложнение разработки, развертывания и масштабирования со временем
- Низкая надежность

Микросервис

+ Гибкость, возможность частичных изменений
+ Высокая надежность
+ Независимость разработки

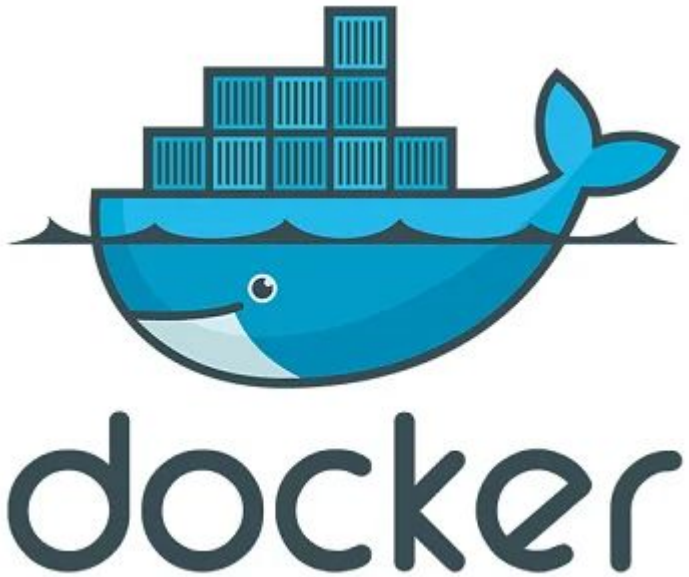
- Сложность первичного развертывания
- Высокие затраты на поддержку
- Сложный вопрос ответственности

Docker & Co

Часто оказывается, что виртуального окружения мало для наших целей (например, хотим развернуть микросервисное приложение), а виртуальной машины - много.

Тогда нам на помощь приходит **docker** - платформа для разработки, доставки и эксплуатации приложений.

По функциональности он находится где-то между виртуальным окружением и виртуальной машиной, но ближе к последней.



Home | Recent Tabs | Current Page

You can type something here...

Понятия docker

Образ (image)

Шаблон для всех контейнеров такого типа, который содержит:

- базовую операционную систему
- версии приложений
- действия для развертки и старта

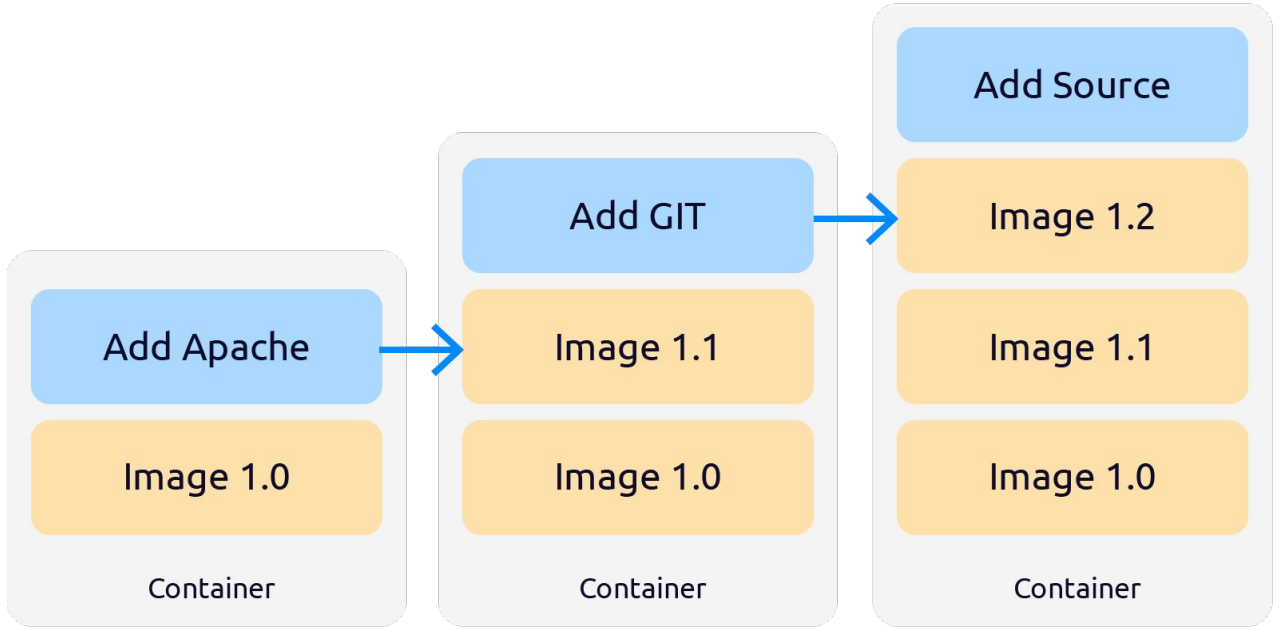
Контейнер (container)

Конкретный исполняемый экземпляр образа. Его можно запустить, остановить, удалить.

Также с ним можно взаимодействовать как с отдельным устройством в сети через IP-адрес и порты.

Как устроен образ?

Образ состоит из слоев - различных частей проекта, которые должны устанавливаться в определенном порядке. Обычно первые два - это операционная система и основное ПО (например Linux + Python). При добавлении каждого слоя получается новый образ.



Два способа создать образ

Руками

Похоже на работу как с обычной ОС:

1. Берется уже существующий базовый образ
2. На его основе запускается контейнер и потом в него вносятся изменения
3. После этого изменения можно сохранить как новый образ

Dockerfile

Создается файл с специальном формате, где прописываются все шаги создания образа:

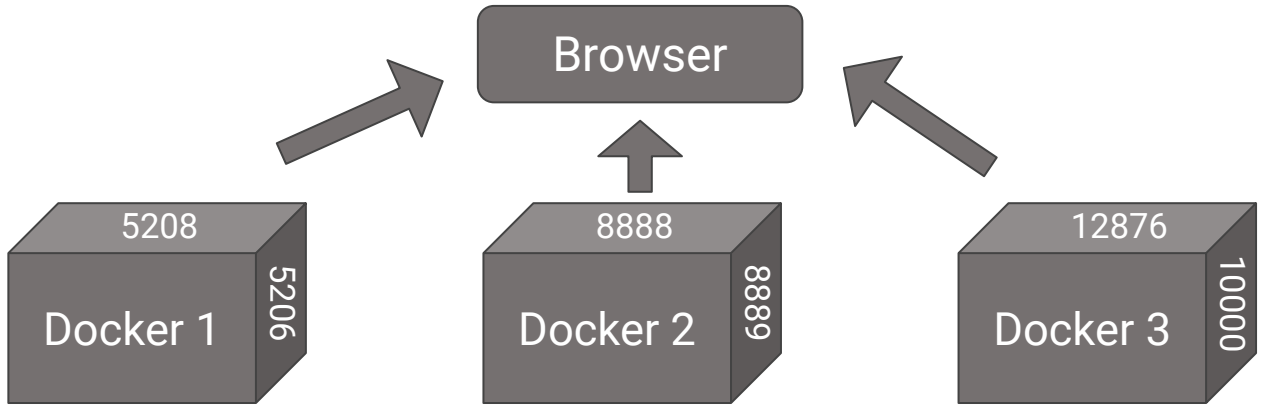
Пример:

```
FROM alpine
COPY my_file /my_file
RUN apk update && apk add vim
CMD cat /my_file
```

А зачем были нужны порты и сети?

Иногда вам нужно запустить в докере программы, отдающие информацию на конкретный порт (например, так себя ведут jupyter notebook, MySQL, neo4j и т.д.) или даже связать вместе несколько контейнеров через локальную сеть.

Для этого необходимо при запуске прописать переход внутренних портов (тех, на которые вещает программа в контейнере) на внешние (те, которые открыты на вашем устройстве).



Про будущее

Следующая пара - практика по докеру. Для этого надо установить сам докер, лучше даже Docker Desktop.

Пара через две недели - knowledge retrieval (плюс возможно что-то еще)

Две пары через три недели - защита проекта.

От меня:

- Выдать вторую домашнюю работу
- Написать требования и критерии к проекту

Пожалуйста, определитесь с темой к концу второй домашки!