Вишняков Родион
Студент 2 курса ПИ, группы БПИ213

Технологии контейнеризации

background

Системы контейнеризации

Источник: IHS Markit, 2019

Технологии контейнеризации приложений помогают сделать приложения более безопасными, облегчают их развертывание и улучшают возможности по их масштабированию.

Контейнерная виртуализация

Контейнер это не вполне виртуальная машина

Были разные решения

• FreeBSD Jail (2000), Virtuozzo Containers (2000), Solaris Containers (2005), Linux-VServer[en], OpenVZ (2005), LXC (2008), iCore Virtual Accounts (2008)

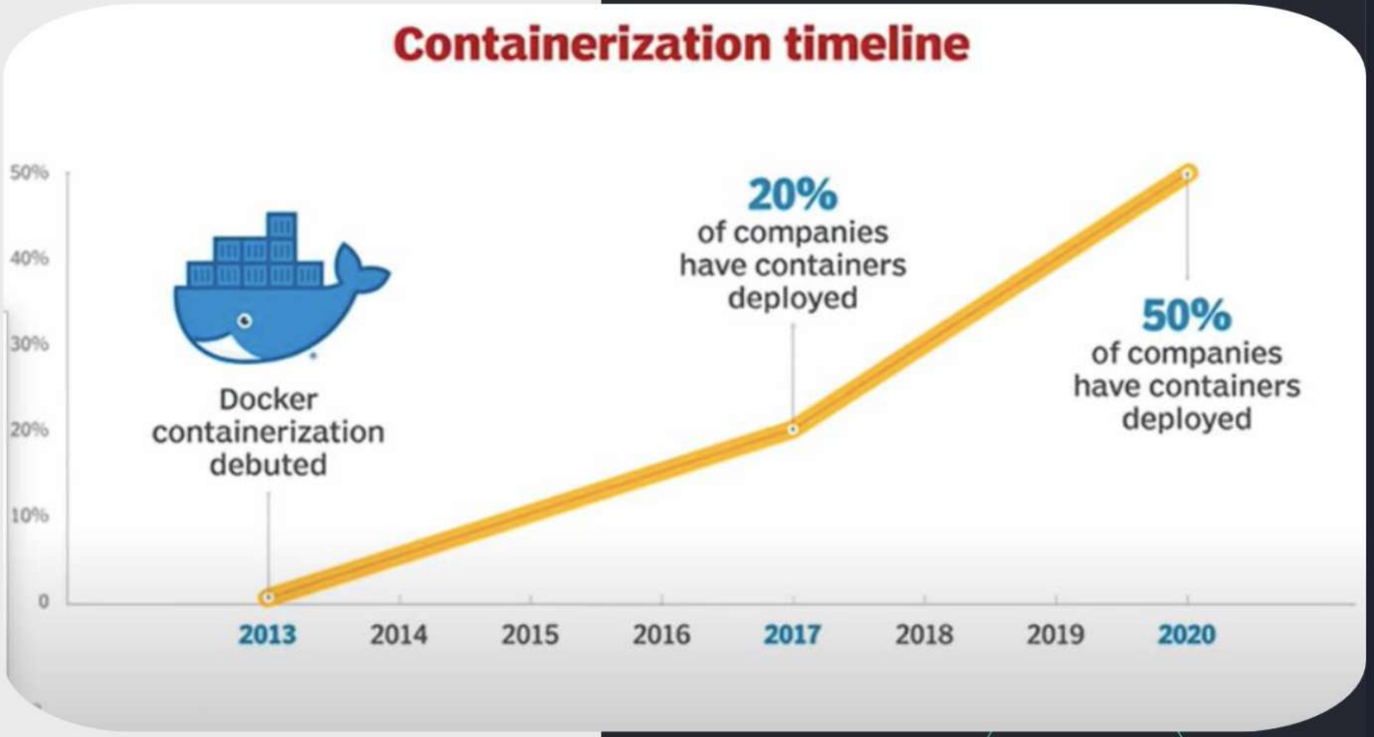
Но потом появился Docker (2013)

• и достаточно быстро стал наиболее используемым приложением для контейнеризации • (и завоевал мир)

Контейнерная виртуализация

Сейчас использование docker распространилось очень широко

- Работа с docker - must-have навык



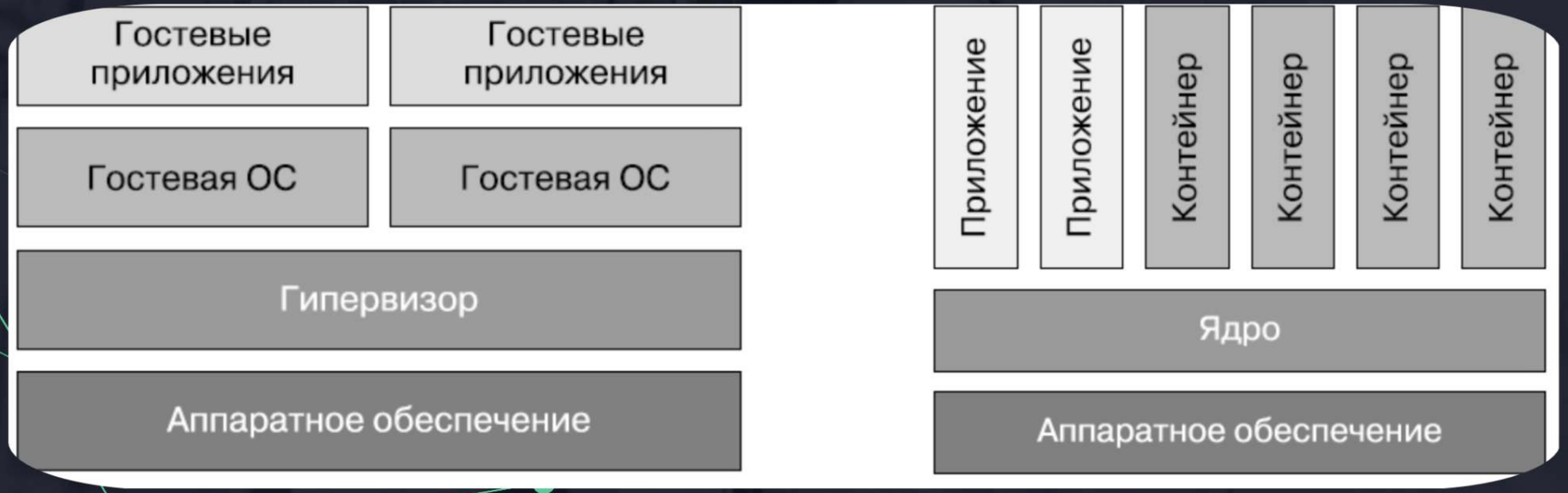
- Почему это так популярно?
- В чем идея?
- Рассмотрим аналогию с появлением стандартных морских контейнеров

Использование стандартных контейнеров для упаковки программных решений имеет схожие плюсы

- Поддержка на разных платформах
- Возможность написать dockerfile и запустить его в любом месте, получив гарантированный повторяемый результат
- Можно упаковать свое приложение в стандартный docker контейнер и запустить его сколько угодно раз
- Можно развернуть, например, в docker compose

Контейнеры vs виртуальные машины

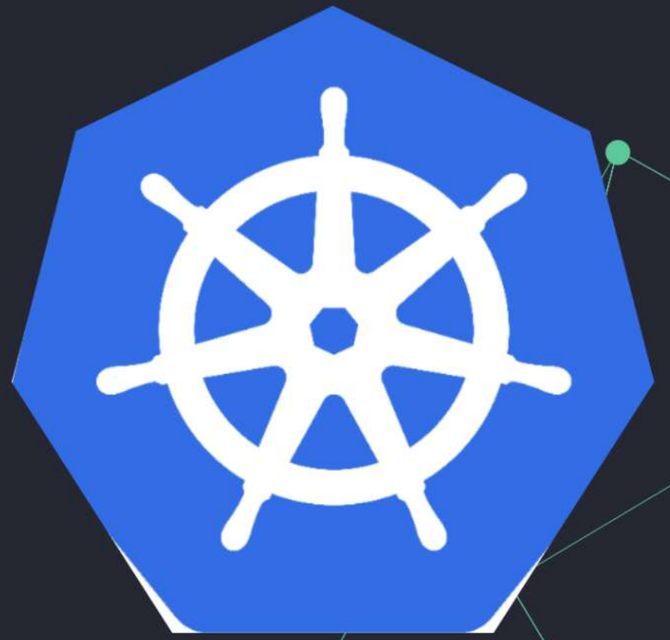
- Виртуальные машины – диспетчер (гипервизор) выделяет ресурсы для гостевых ОС (у каждой – свое ядро)
- Контейнеры – делят одно ядро с хостом



Оркестрация

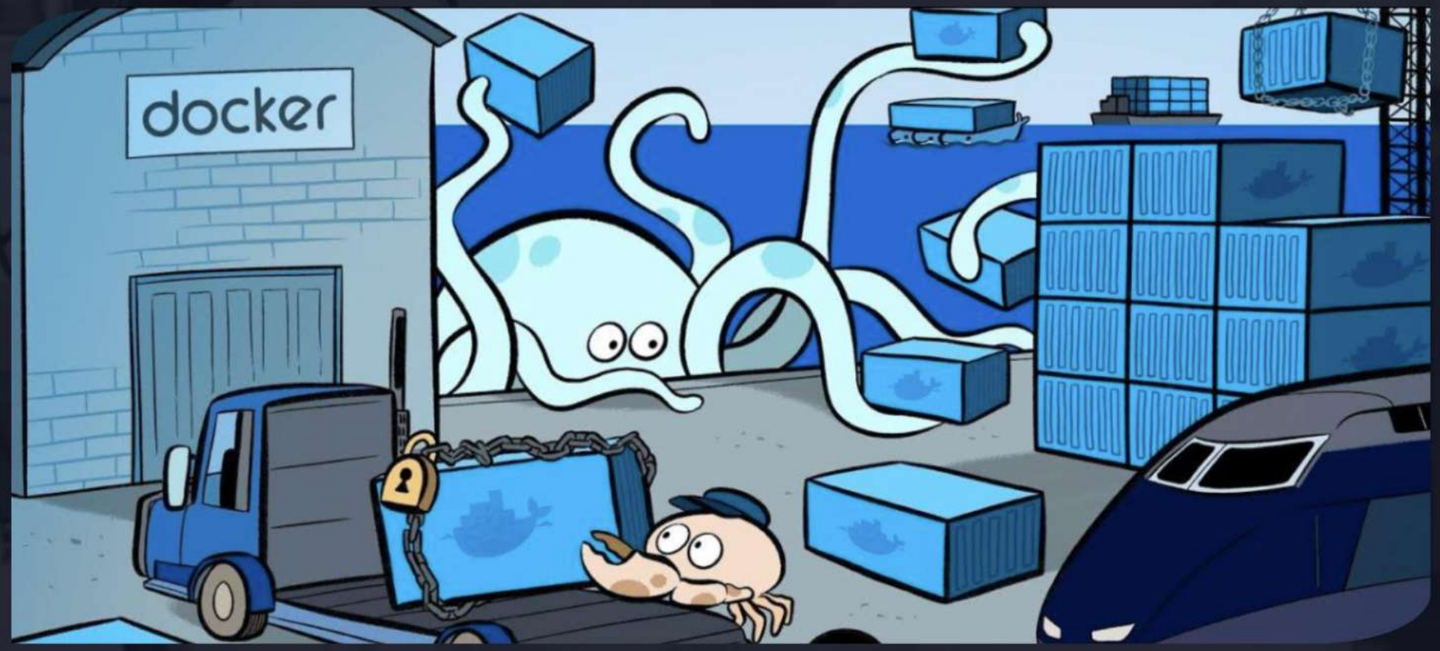
- Управление большим количеством контейнеров
- На самом деле большим – сотнями, тысячами
- Контроль их состояния, выделение ресурса, перезапуск, балансировка, развертывание дополнительных серверов (масштабирование) и т.п.
- Для этого так же есть специализированные решения

Прежде всего - Kubernetes

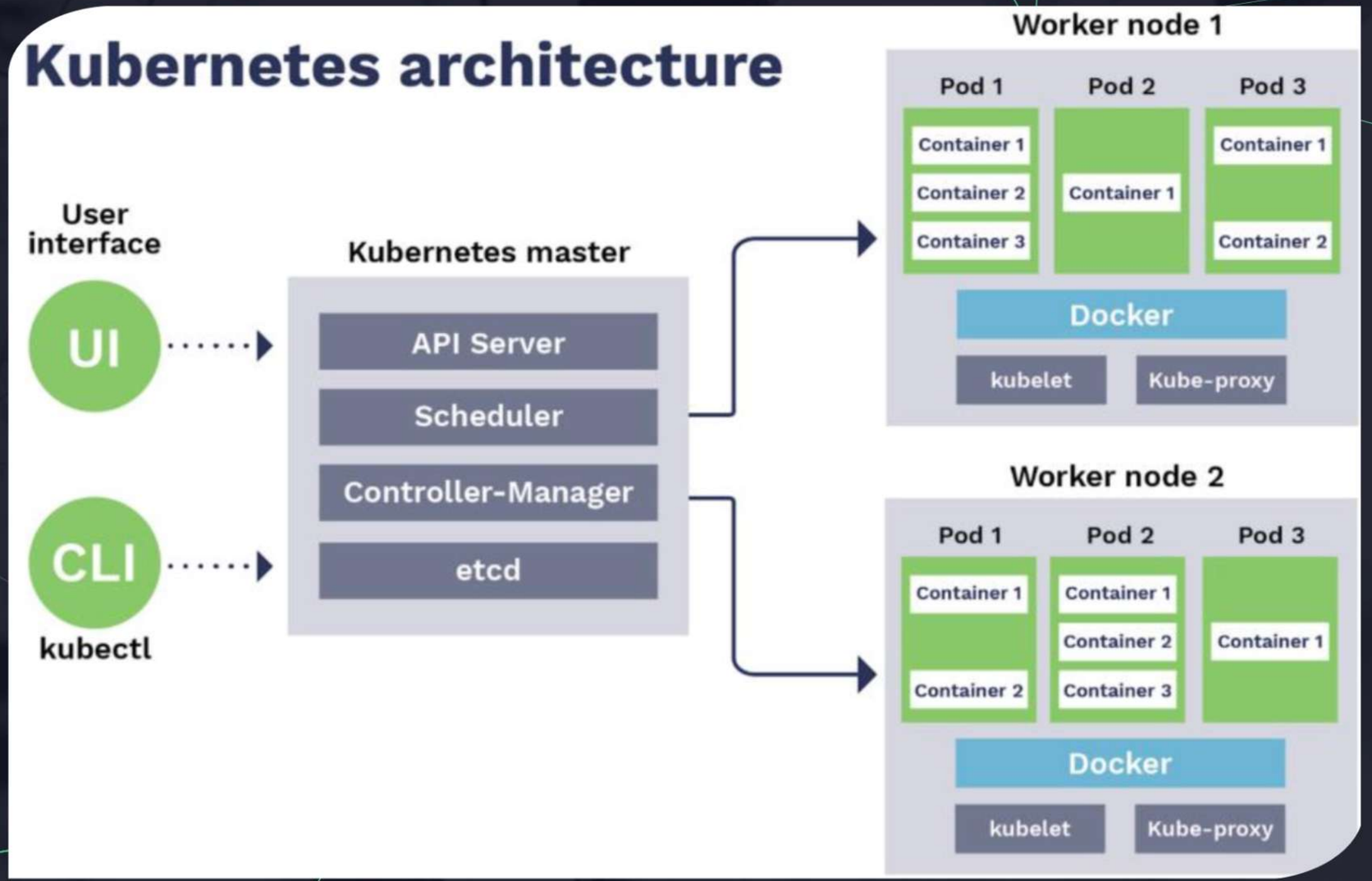


Оркестрация

- Стандартная схема работы
  - Разрабатываем приложение
  - Упаковываем его в docker контейнер
  - Разворачиваем в Kubernetes (частный, или в публичное облако)



Оркестрация

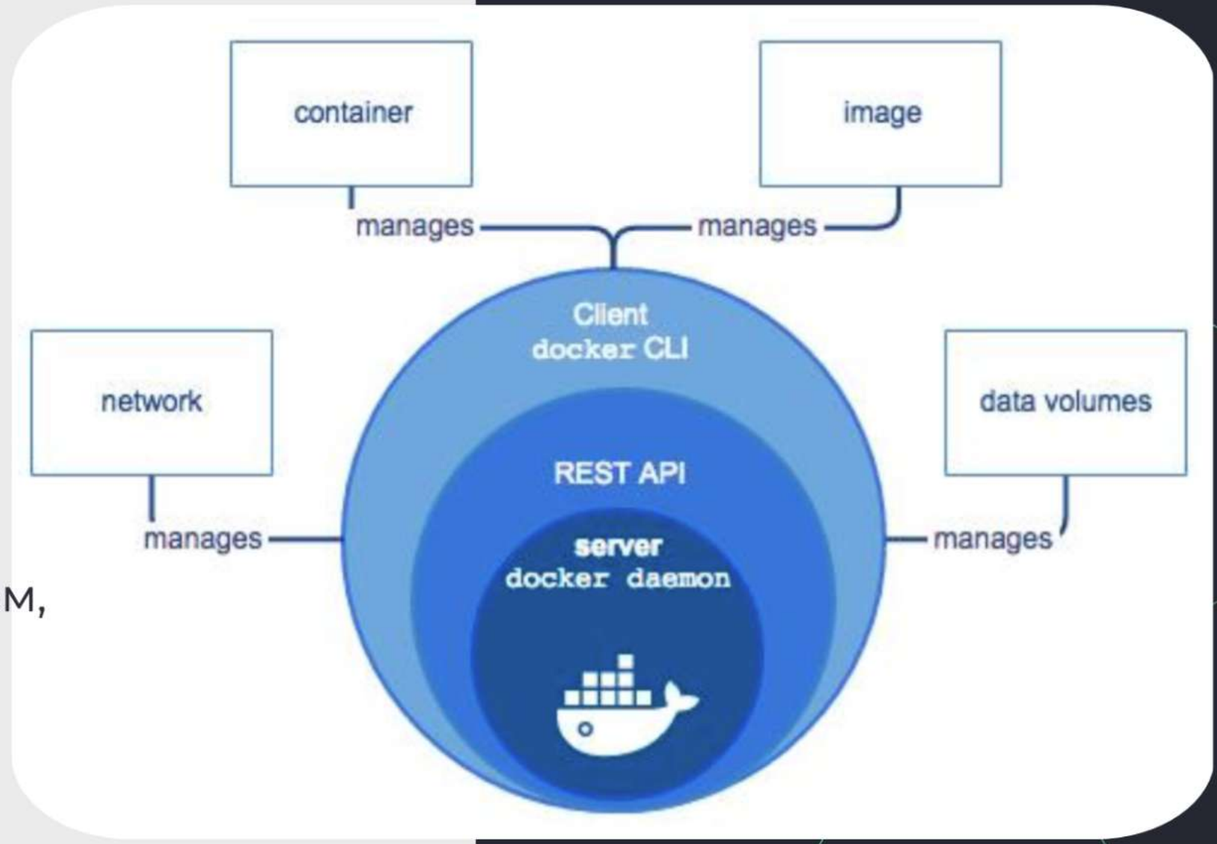


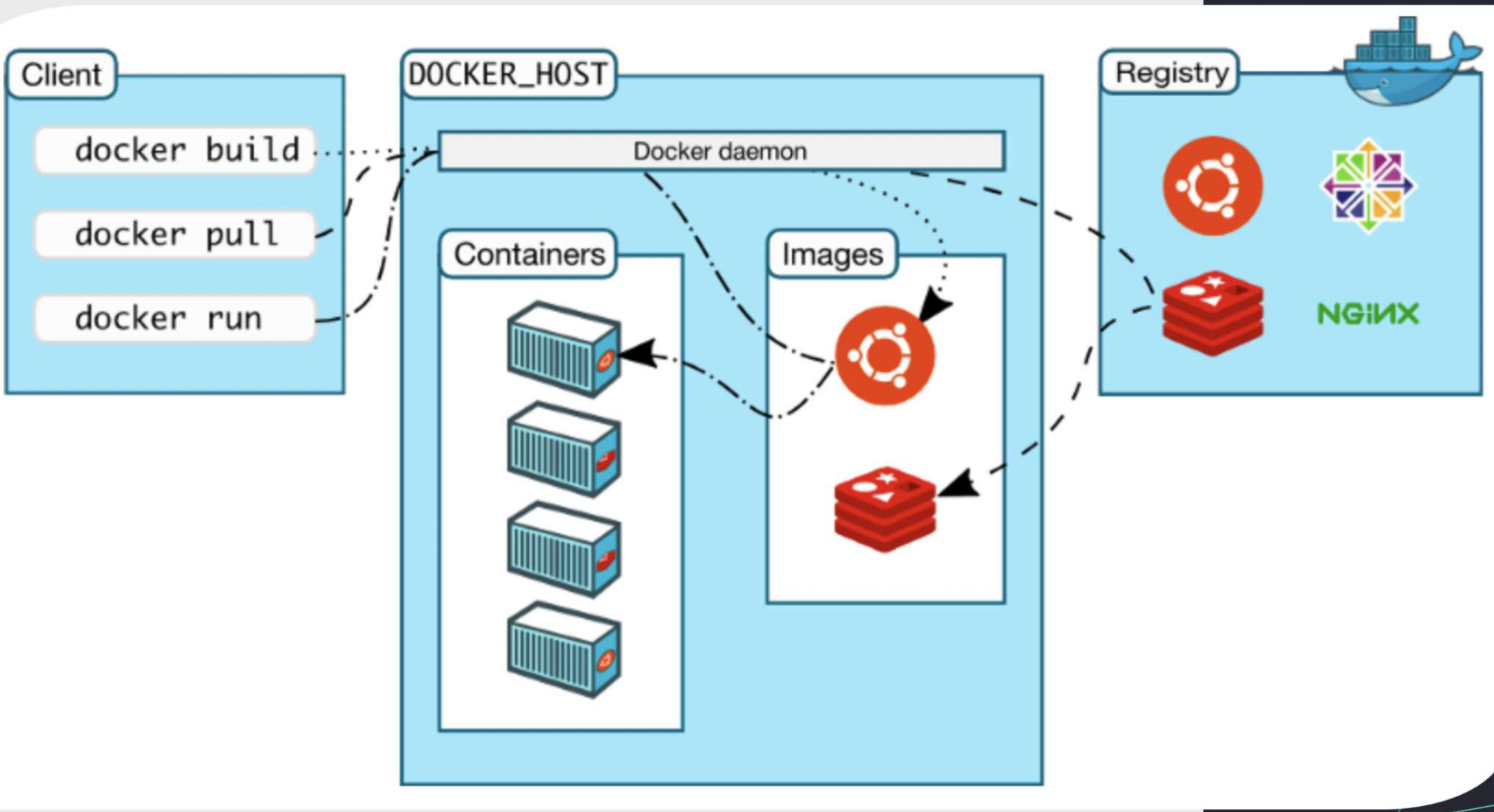
1 контейнер изолирует:
пользовательское окружение

2 контейнер делит с хостом:
- ядро
- пространство памяти

ядра

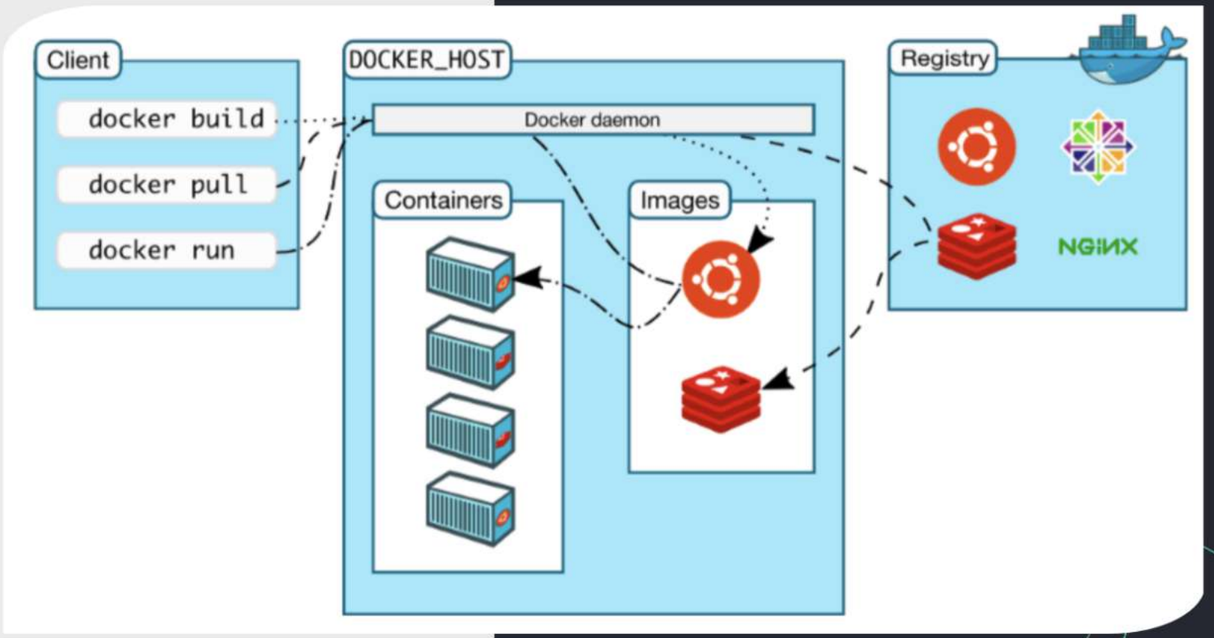
3 Docker (Docker Engine) - это одна из систем, позволяющих контейнеризировать приложения, предназначена для разработки, развертывания и запуска приложений в контейнерах





Клиент Docker (Docker Client) — это основное средство, которое используют для взаимодействия с Docker

Демон Docker (Docker Daemon) — это сервер Docker, который ожидает запросов к API Docker. Демон Docker управляет образами, контейнерами, сетями и томами



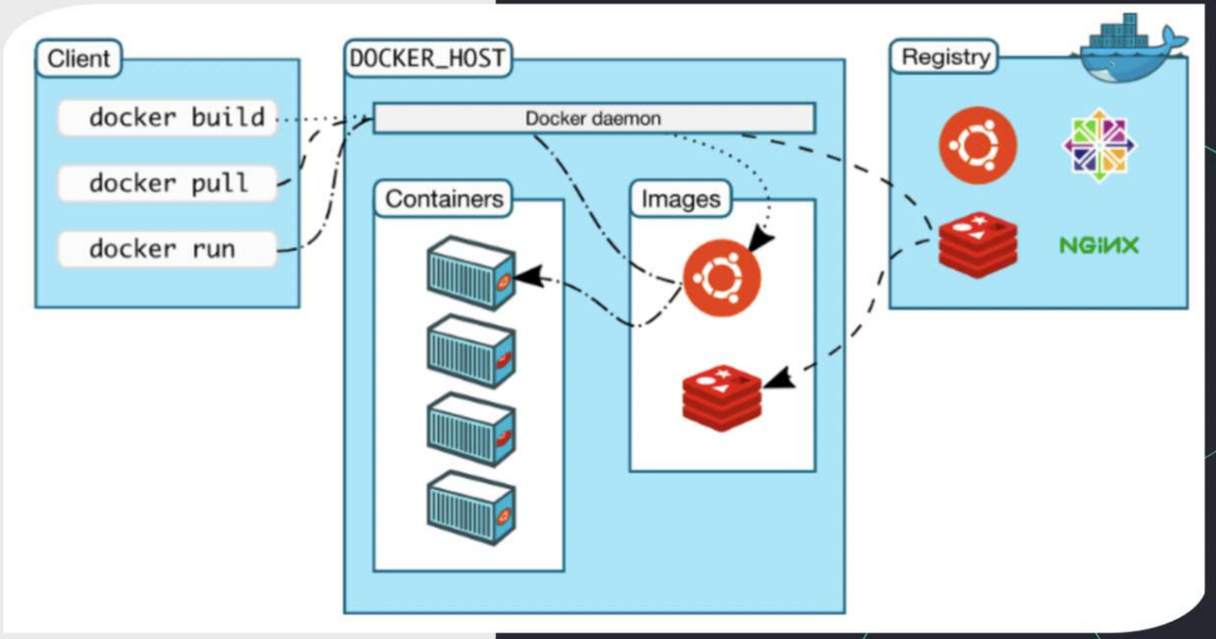
Тома Docker (Docker Volumes) – наиболее предпочтительный механизм постоянного хранения данных, потребляемых или производимых приложениями.

Реестр Docker (Docker Registry) – удаленная платформа, используемая для хранения образов Docker. В ходе работы с Docker образы отправляют в реестр и загружают из него.

Docker Хаб (Docker Hub) — это самый крупный реестр образов Docker, используется при работе с Docker по умолчанию.

Репозиторий Docker (Docker Repository) – набор образов Docker, обладающих одинаковыми именами и разными тегами.

Теги — идентификаторы образов.



Контейнер — среда, где разворачивается и запускается сохраненный заранее образ с отдельным ПО \ программой \ сервисом.

Образ — полностью работоспособный исполняемый пакет определенного ПО для запуска в контейнере Docker.

Dockerfile — файл инструкций для демона Docker под соответствующий образ. Для **нового** образа, сначала готовится Dockerfile после чего сам пакет образа.

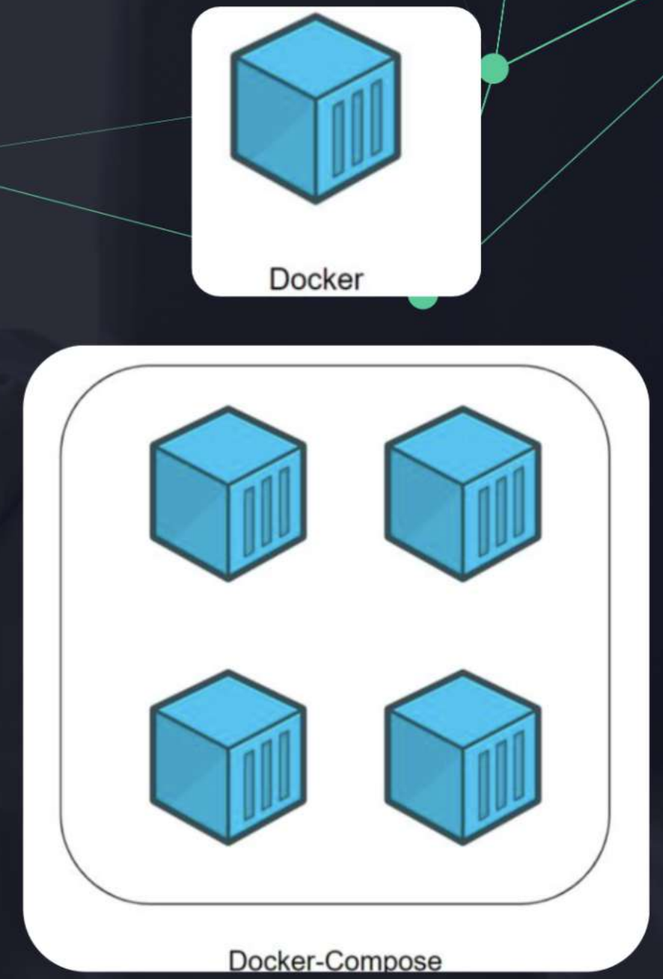
Docker vs VM

	Docker	Виртуальные машины (VM)
Время загрузки	Загрузка через несколько секунд.	Загрузка виртуальных машин занимает несколько минут.
Работает на	Docker используют механизм исполнения.	ВМ используют гипервизор.
Эффективность памяти	Для виртуализации не требуется места, а значит, и меньше памяти.	Требуется загрузка всей ОС перед запуском поверхности, поэтому она менее эффективна.
Изоляция	Нет условий для систем изоляции.	Возможность вмешательства минимальна из-за эффективного механизма изоляции. Изолируется средствами процессора.
Развертывание	Развертывание легко, так как только одно изображение в контейнере может использоваться на всех платформах.	Развертывание сравнительно длительное, поскольку за выполнение отвечают отдельные экземпляры.

Docker-compose

Docker Compose — инструмент, упрощающий развёртывание приложений, для работы которых требуется несколько контейнеров Docker, позволяет выполнять команды, описываемые в файле docker-compose.yml. Эти команды можно выполнять столько раз, сколько потребуется.

Интерфейс командной строки Docker Compose упрощает взаимодействие с многоконтейнерными приложениями.



Dockerfile

- Контейнер Docker — самодостаточная операционная система, в которой имеется только самое необходимое и код приложения.
- Образы Docker являются результатом процесса их сборки, а контейнеры Docker — это выполняющиеся образы.
- Dockerfile файлы сообщают Docker о том, как собирать образы, на основе которых создаются контейнеры.
- Каждому образу Docker соответствует файл, который называется Dockerfile.
- При запуске команды docker build для создания нового образа подразумевается, что Dockerfile находится в текущей рабочей директории. Если этот файл находится в каком-то другом месте, его расположение можно указать с использованием флага -f.

Dockerfile

- В файлах Dockerfile содержатся инструкции по созданию образа (набираются заглавными буквами). С них начинаются строки этого файла.

После инструкций идут их аргументы.

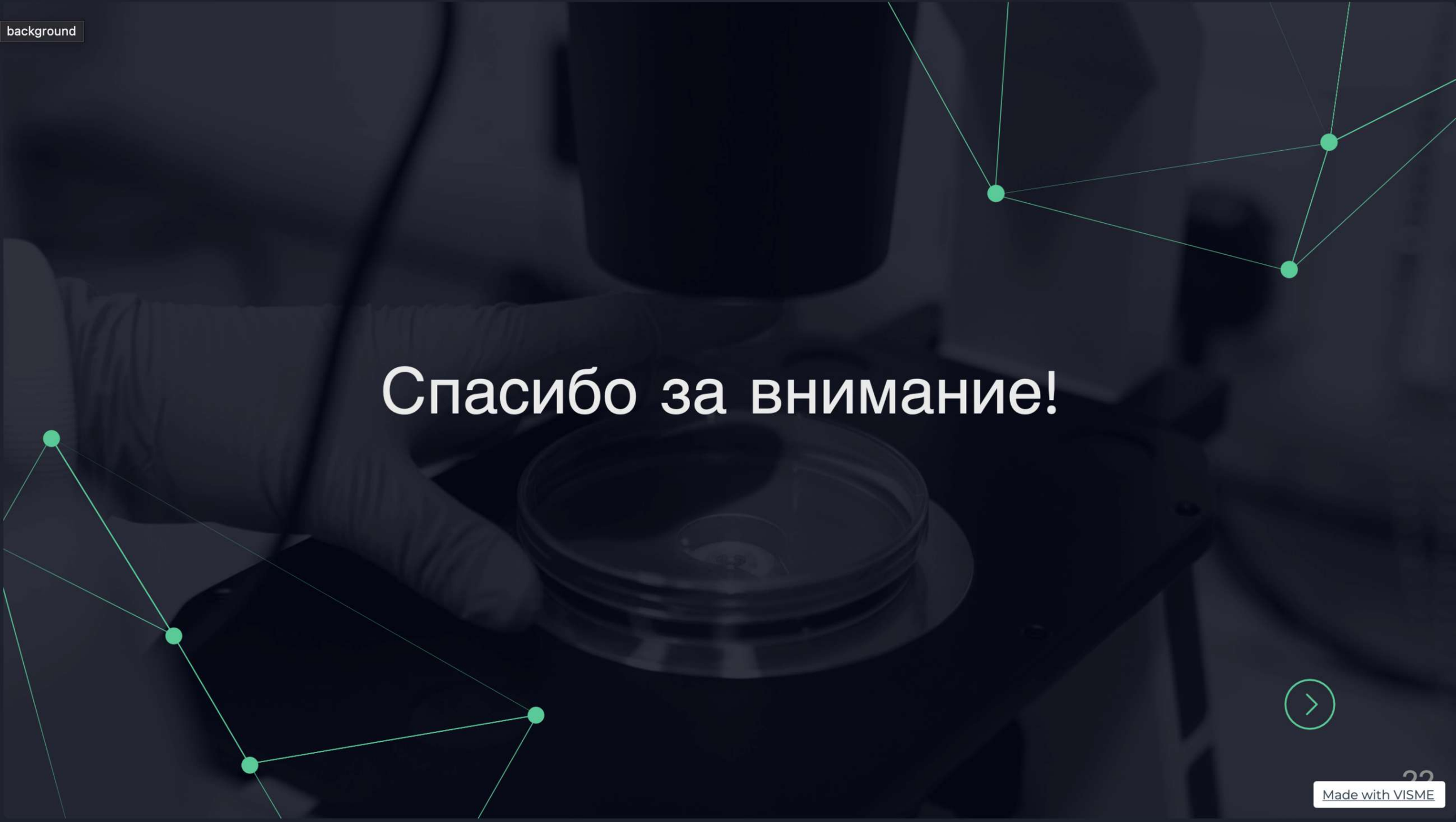
- Инструкции, при сборке образа, обрабатываются сверху вниз.
- Слои в итоговом образе создают только инструкции FROM, RUN, COPY, и ADD.

Основные инструкции

FROM	Задаёт базовый (родительский) образ.
COPY	Копирует в контейнер файлы и папки.
ADD	Копирует файлы и папки в контейнер, может распаковывать локальные .tar-файлы.
WORKDIR	Задаёт рабочую директорию для следующей инструкции.
ENV	Устанавливает постоянные переменные среды.
ARG	Задаёт переменные для передачи Docker во время сборки образа.
LABEL	Описывает метаданные. Например — сведения о том, кто создал и поддерживает образ.
RUN	Выполняет команду и создаёт слой образа. Используется для установки в контейнер пакетов.
CMD	Описывает команду с аргументами, которую нужно выполнить когда контейнер будет запущен. Аргументы могут быть переопределены при запуске контейнера. В файле может присутствовать лишь одна инструкция CMD.
ENTRYPOINT	Предоставляет команду с аргументами для вызова во время выполнения контейнера. Аргументы не переопределяются.
EXPOSE	Указывает на необходимость открыть порт.
VOLUME	Создаёт точку монтирования для работы с постоянным хранилищем.

Создаем образ

```
FROM python:3.7.2-alpine3.8
LABEL maintainer="boss@example.com"
ENV ADMIN="boss"
RUN apk update && apk upgrade && apk add bash
COPY . ./app
ADD https://raw.githubusercontent.com/discdiver/pachy-vid/master/sample_vids/vid1.mp4 \
/my_app_directory
RUN ["mkdir", "/a_directory"]
CMD ["python", "./my_script.py"]
```


background
Спасибо за внимание!
Made with VISME