OTUS

Онлайн образование

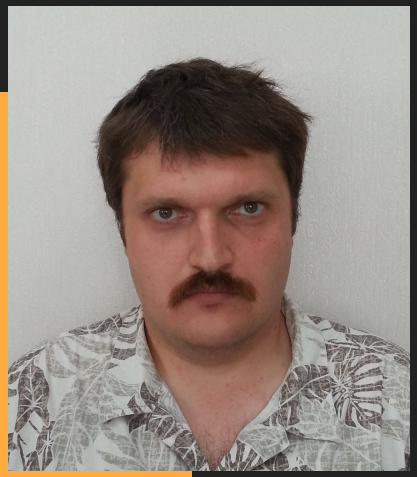
otus.ru

REC Проверить, идет ли запись

Меня хорошо видно && слышно?

Тема вебинара

Docker



Непомнящий Евгений

Java / Kotlin developer

Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в TG

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения

Индивидуально

Время, необходимое
на активность

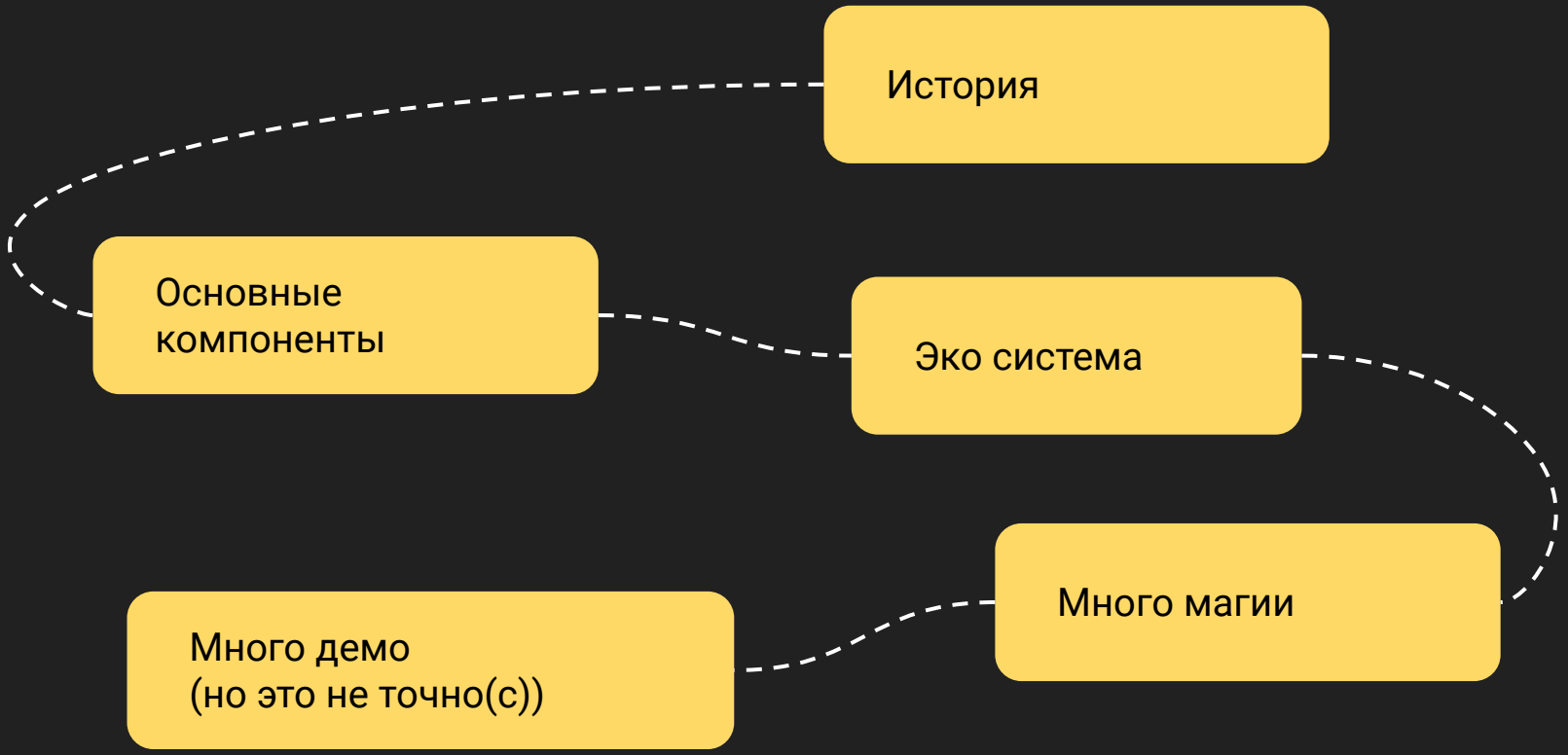
Пишем в чат

Говорим голосом

Документ

Ответьте себе или
задайте вопрос

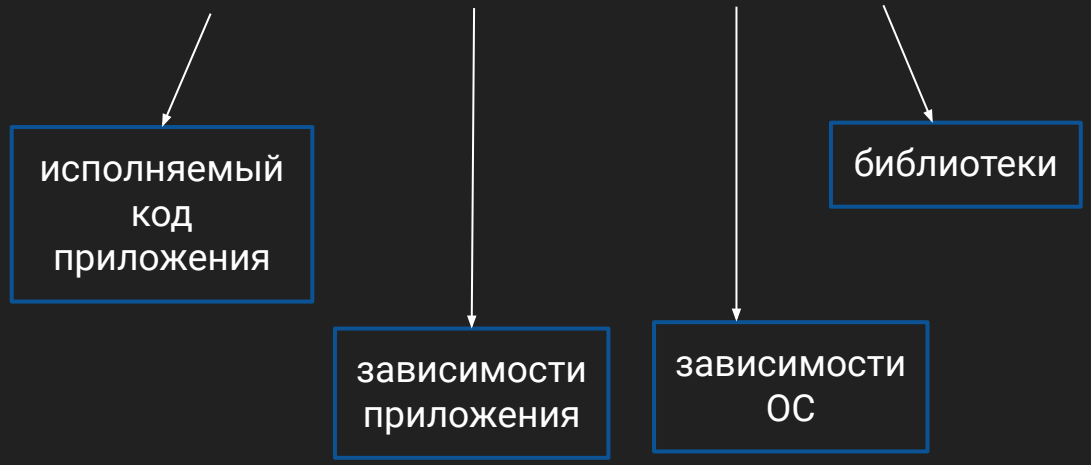
Карта курса


История
Основные компоненты
Эко система
Много магии
Много демо (но это не точно(с))

Зачем нужен Docker?

Docker - технология для создания и управления контейнерами.

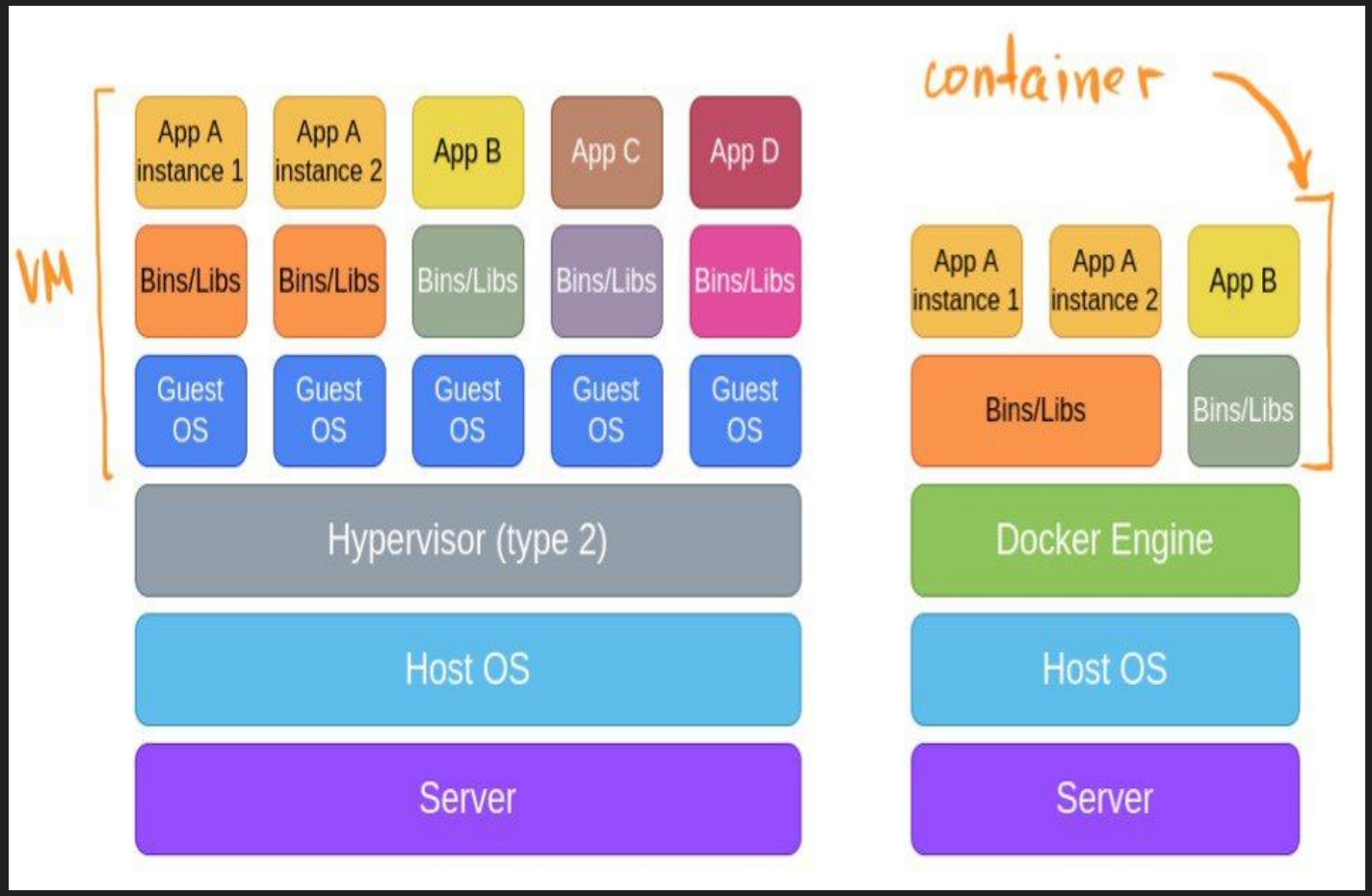
Контейнер (container) содержит в себе все необходимое для запуска и работы приложения



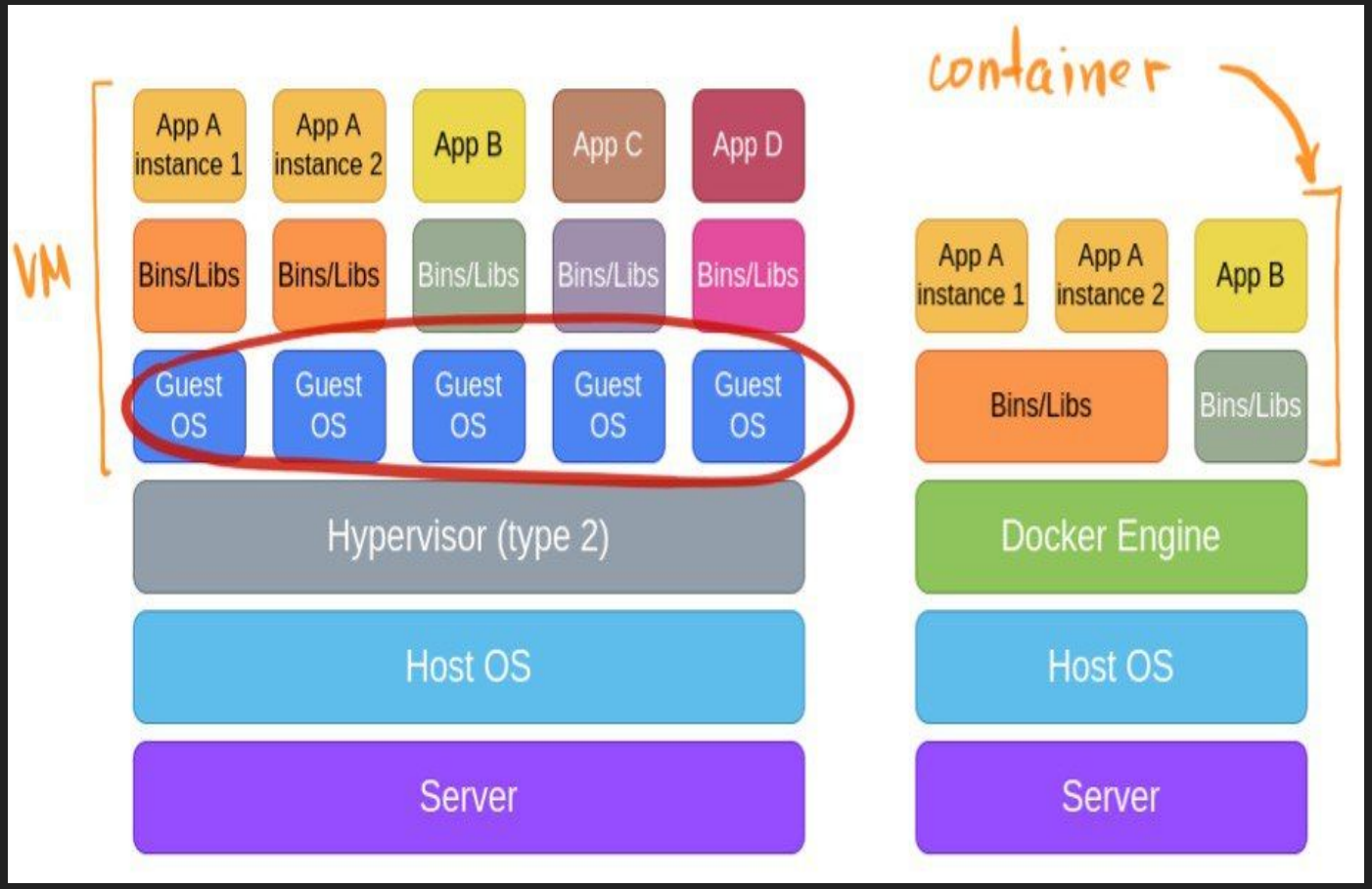
Write Once Run Everywhere

- Переносимость
- Масштабируемость
- Уменьшают издержки на сопровождение инфраструктуры
- Легковесность (об этом чуть далее)
- Позволяет изолировать приложения друг от друга
- Хорошо вписываются в современные архитектурные тенденции

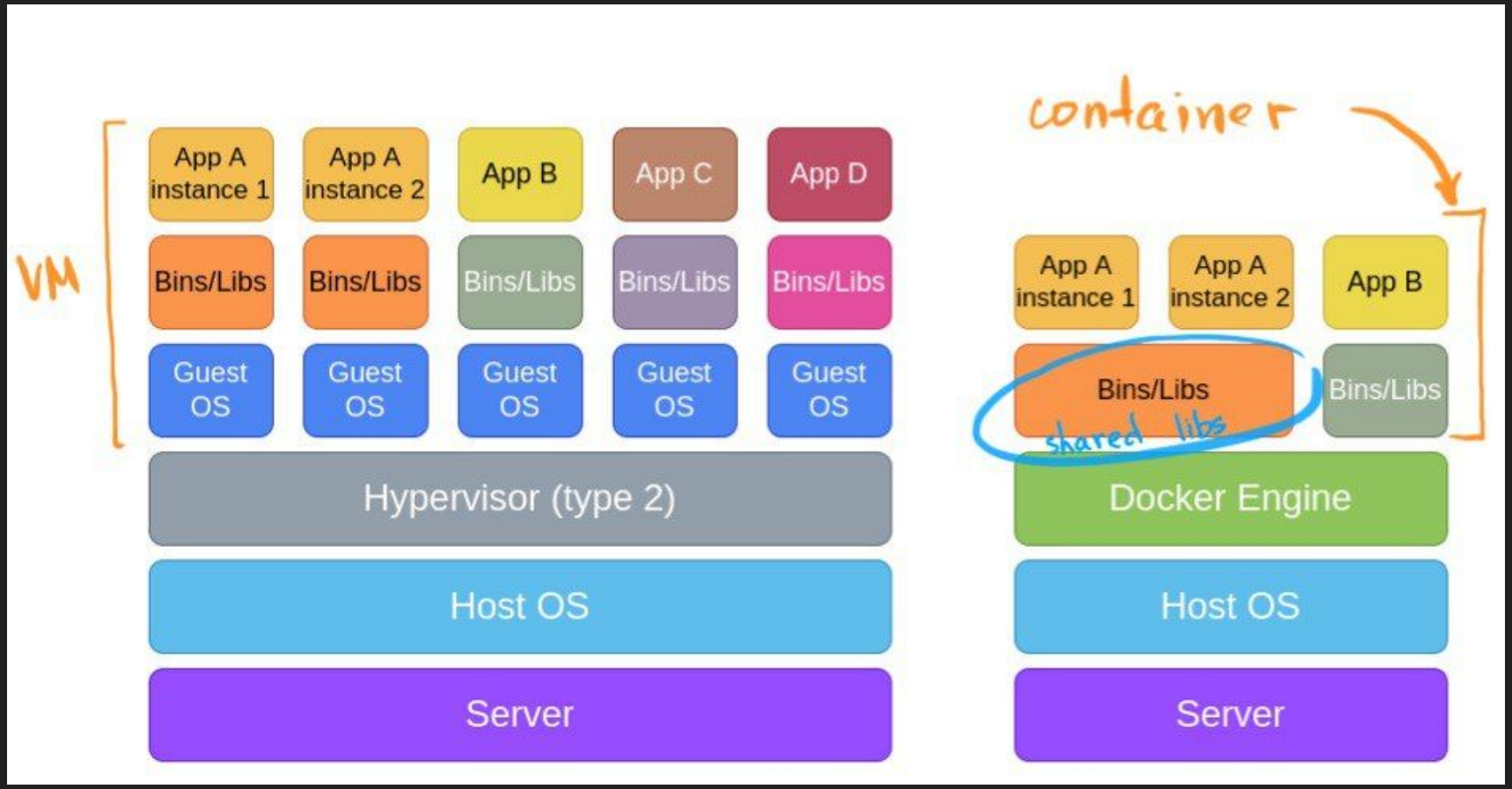
Легковесные контейнеры



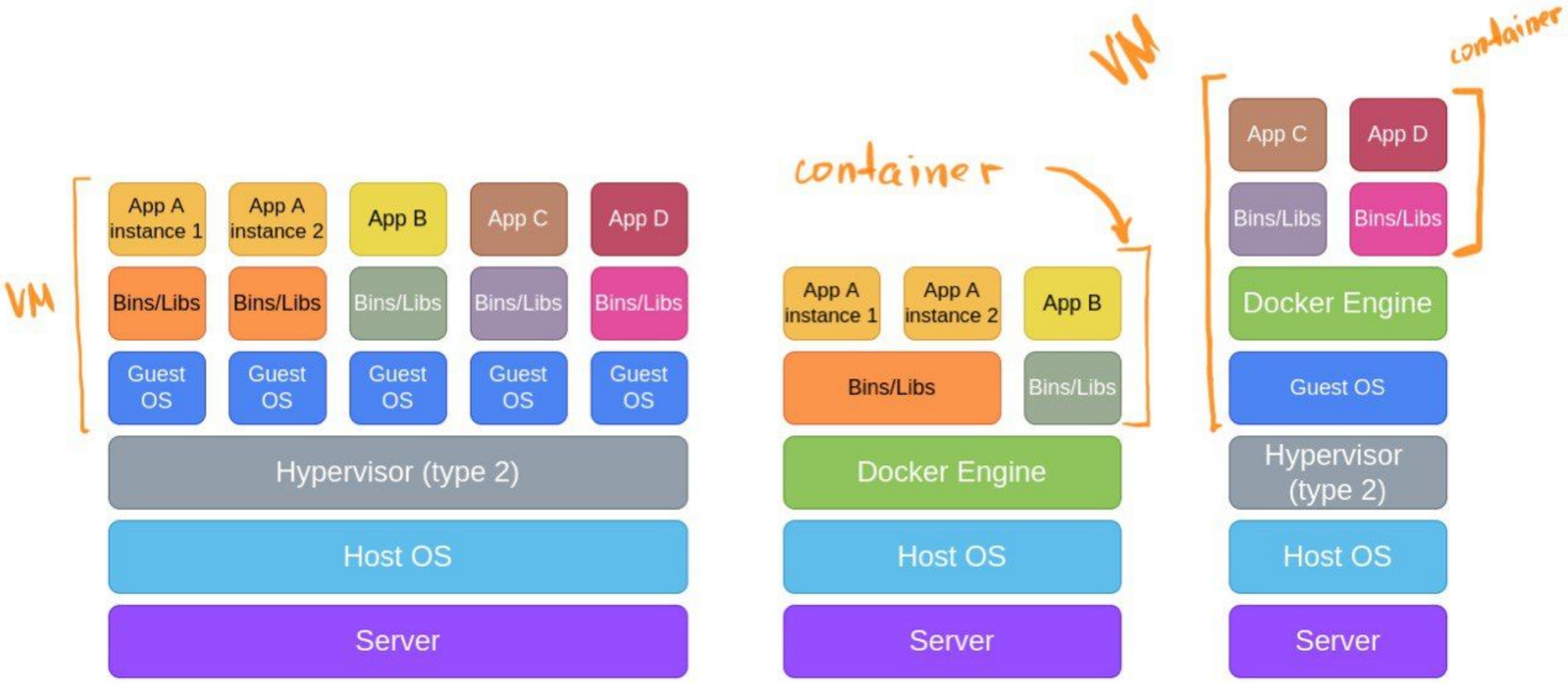
Легковесные контейнеры



Легковесные контейнеры



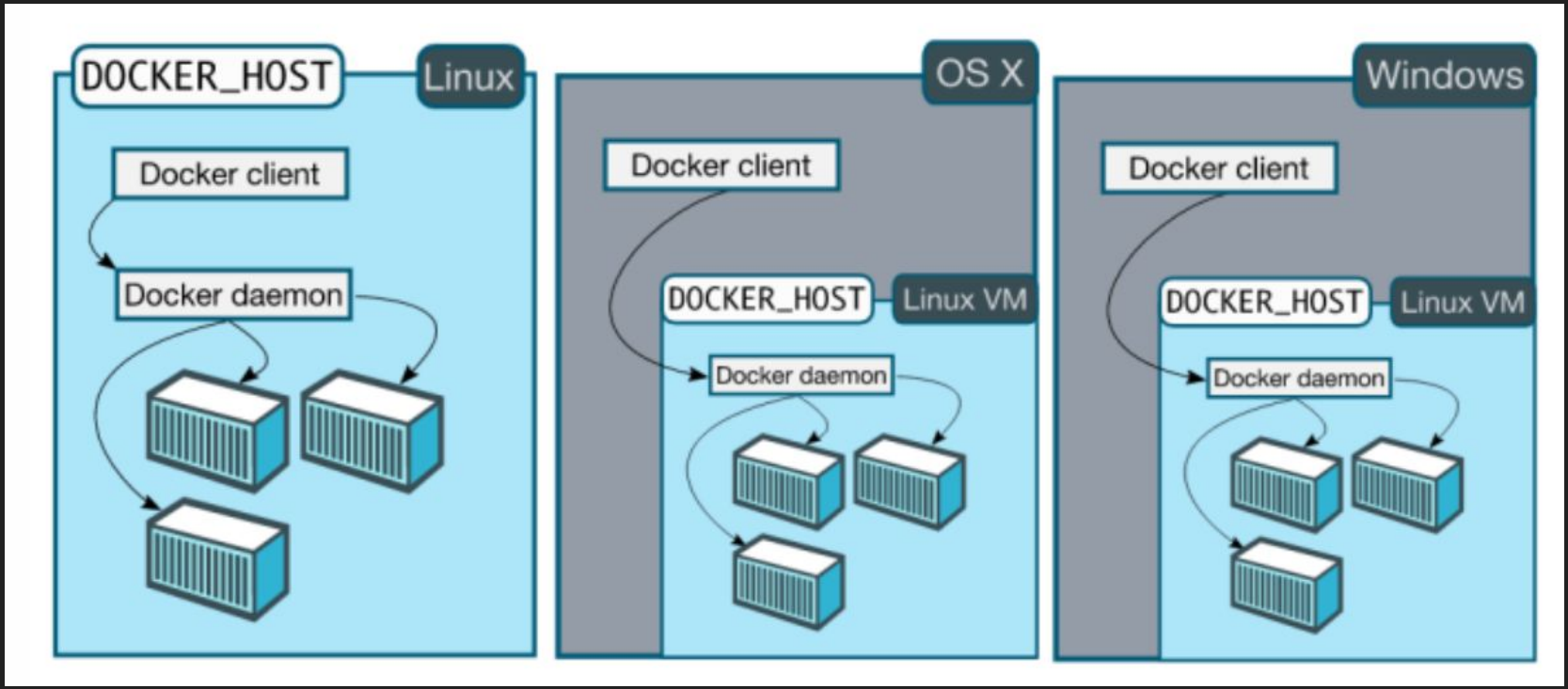
Легковесные контейнеры



Docker

Docker - это не виртуальная машина!

Docker


DOCKER_HOST
Linux
Docker client
Docker daemon
OS X
Docker client
DOCKER_HOST
Linux VM
Docker daemon
Windows
Docker client
DOCKER_HOST
Linux VM
Docker daemon

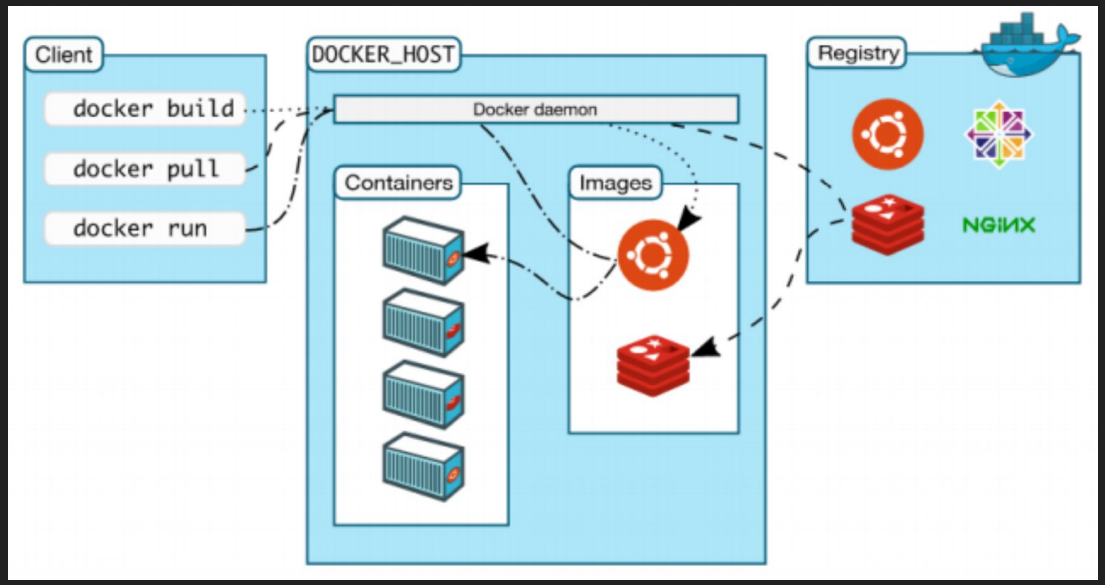
Из каких частей состоит Docker?

Daemon:

- предоставляет api
- управляет объектами
- взаимодействует с другими daemon`ми

cli:

- принимает команды от пользователя
- взаимодействует с api docker daemon



Docker

Так называемый приватный “репозиторий” для хранения image.

Docker hub, Private registry, Docker Store

Запускаем контейнер

```
$> docker run -p 5432:5432 -e POSTGRES_PASSWORD=postgres -d
--name otus_postgres postgres
```

```
Unable to find image 'postgres:latest' locally
latest: Pulling from library/postgres
c229119241af: Pull complete
3ff4ca332580: Pull complete
…
b133d2355caa: Pull complete
b2694eb85faf: Pull complete
503b75e1e236: Pull complete
Digest: sha256:e3d8179786b8f16d066b313f381484a92efb175d1ce8355dc180fee1d5fa70ec
Status: Downloaded newer image for postgres:latest
b3eef16e3432f89bc296188af8b0cb4f3e0fbd6ba6822003f4f72287cf38dde0
```

Структура команды run

```
$> docker run <опции> <образ> <команда>
```

```
$> docker run \
    -p 5432:5432 \
    -e POSTGRES_PASSWORD=postgres \
    -d \
    --name otus_postgres postgres
```

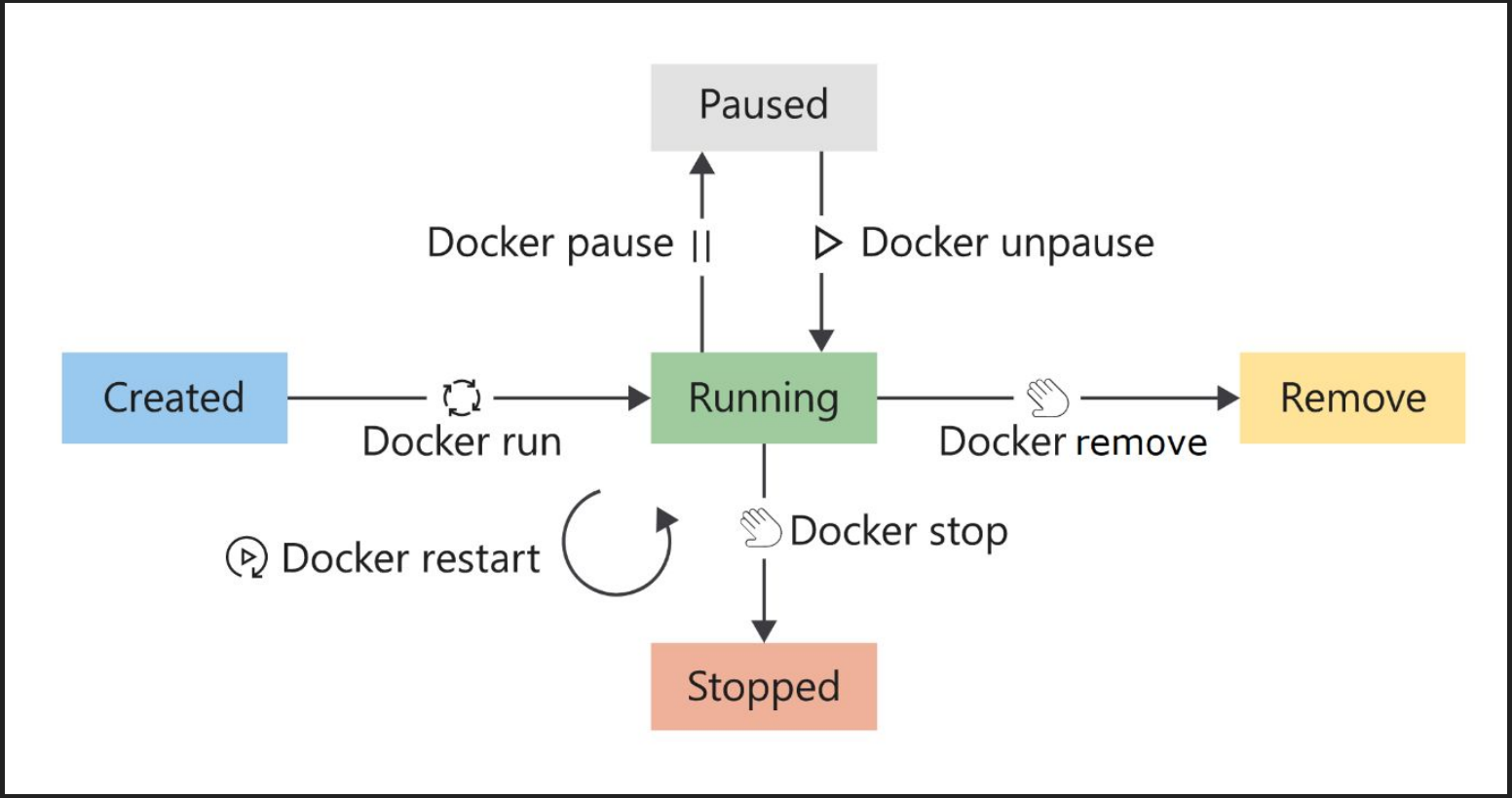
Команды Docker

`$> docker`	выведет полный список команд docker
`$> docker ps`	выведет список всех работающих контейнеров можно добавить опцию `-a` для вывода списка всех контейнеров
`$> docker images`	посмотреть список всех скачанных образов
`$> docker start`	запустить контейнер
`$> docker stop`	остановить контейнер(ы) `docker stop $(docker ps -a -q)` остановит все запущенные контейнеры
`$> docker pull`	скачать образ
`$> docker build`	собрать образ

Docker полезности

docker df - сколько места съел докер
docker stats - смотрим сколько ресурсов потребляет контейнер
docker logs - логи контейнера
docker inspect - информация по контейнеру
docker … prune … - очистить (перед этим смотри доку)

Жизненный цикл Docker Container



Dockerfile

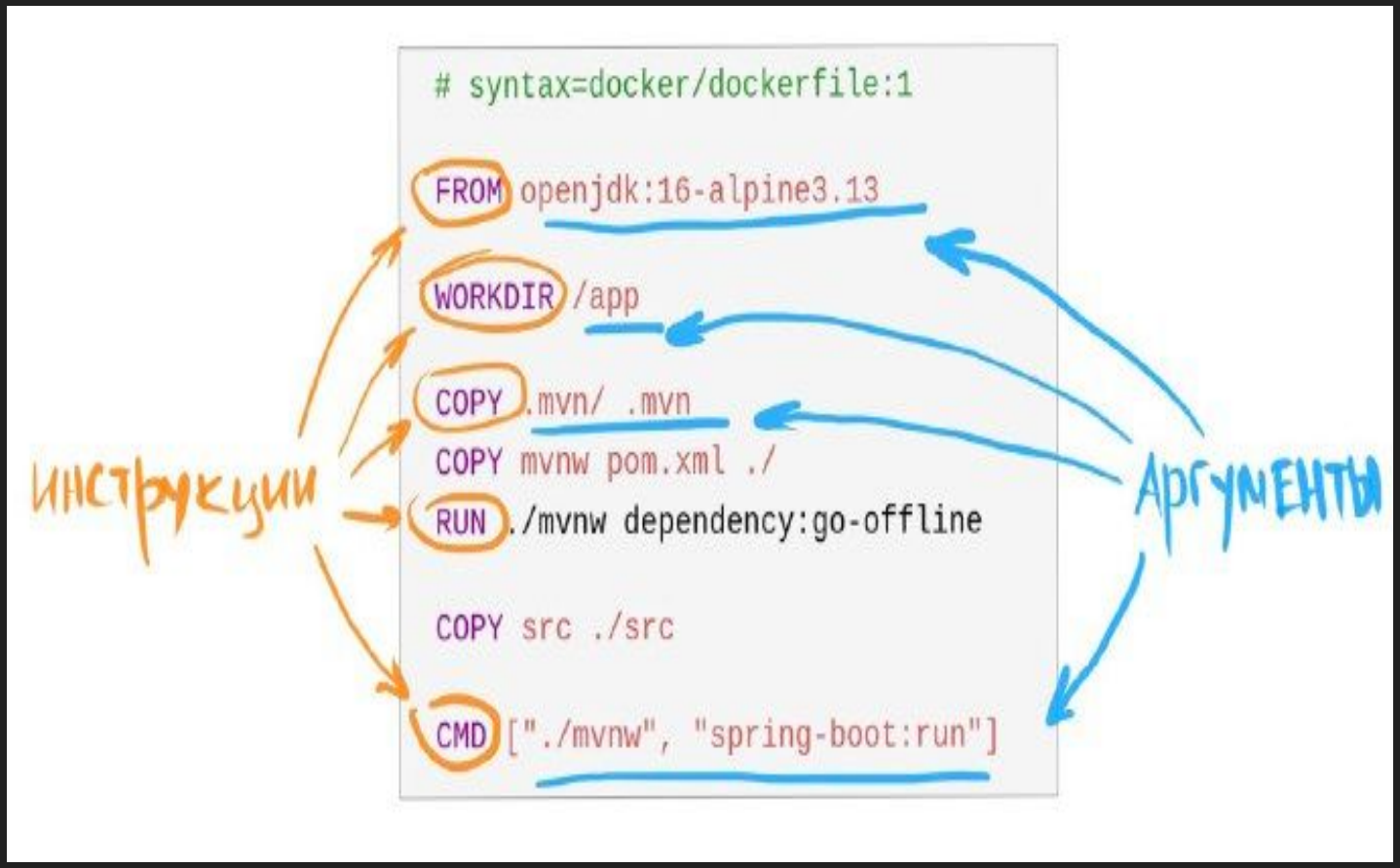
Содержит в себе все инструкции для создания docker образа.

```
# комментарий
<инструкция докер> <аргументы>
...
```

Docker выполняет все инструкции из Dockerfile по порядку, делая коммит результата инструкции

```
# syntax=docker/dockerfile:1

FROM openjdk:16-alpine3.13

WORKDIR /app

COPY .mvn/ .mvn
COPY mvnw pom.xml ./
RUN ./mvnw dependency:go-offline

COPY src ./src

CMD ["./mvnw",
"spring-boot:run"]
```

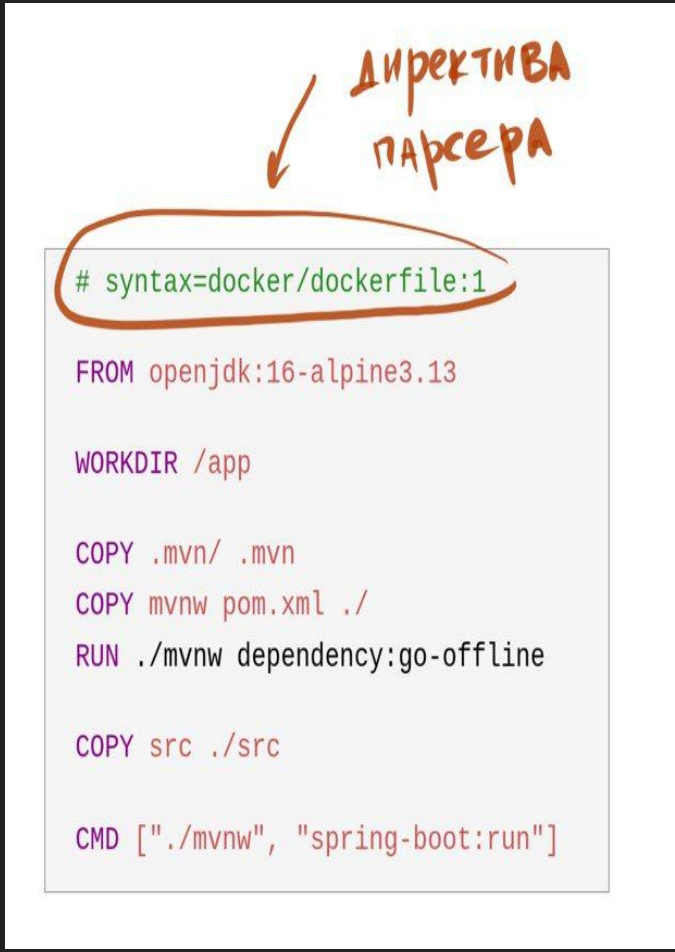
Dockerfile


syntax=docker/dockerfile:1
FROM openjdk:16-alpine3.13
WORKDIR /app
COPY .mvn/ .mvn
COPY mvnw pom.xml ./
RUN ./mvnw dependency:go-offline
COPY src ./src
CMD ["./mvnw", "spring-boot:run"]
инструкции
аргументы

Dockerfile

Директивы парсера:

- Не являются обязательными.
- Влияют на способ, которым обрабатываются следующие инструкции в Dockerfile.
- Не добавляют слои в сборку.
- Не отображаются как отдельные шаги сборки.



Должны быть в самом верху Dockerfile, т.к. как только Docker начнет обработку файла, он перестает искать директивы парсера.

Инструкции Dockerfile

- FROM — задаёт родительский (главный) образ;
- ENV— создаёт переменную окружения;
- RUN — запускает команды;
- COPY — копирует файлы и директории в контейнер;
- ADD — делает всё то же, что и инструкция COPY. Но ещё может распаковывать локальные .tar файлы;
- CMD— указывает команду и аргументы для выполнения внутри контейнера. Параметры могут быть переопределены. Использоваться может только одна инструкция CMD;
- WORKDIR— устанавливает рабочую директорию для инструкции CMD и ENTRYPOINT;
- ARG— определяет переменную для передачи Docker’у во время сборки;
- ENTRYPOINT — предоставляет команды и аргументы для выполняющегося контейнера. Суть его несколько отличается от CMD, о чём мы поговорим ниже;
- EXPOSE — открывает порт;
- VOLUME — создаёт точку подключения директории для добавления и хранения постоянных данных.

Docker

А как запустить то?0_o

CMD
ENTRYPOINT

Режимы работы:
shell

```
FROM alpine
ENTRYPOINT ping www.google.com
```

exec

```
FROM alpine
ENTRYPOINT ["ping", "www.google.com"]
```

Docker

Комбинированное использование

```
FROM alpine
ENTRYPOINT ["ls", "/usr"]
CMD ["/var"]
```

А если я хочу более сложную конструкцию для запуска?

Никто не запрещает использовать bash-скрипт

Dockerfile: Java Hello World

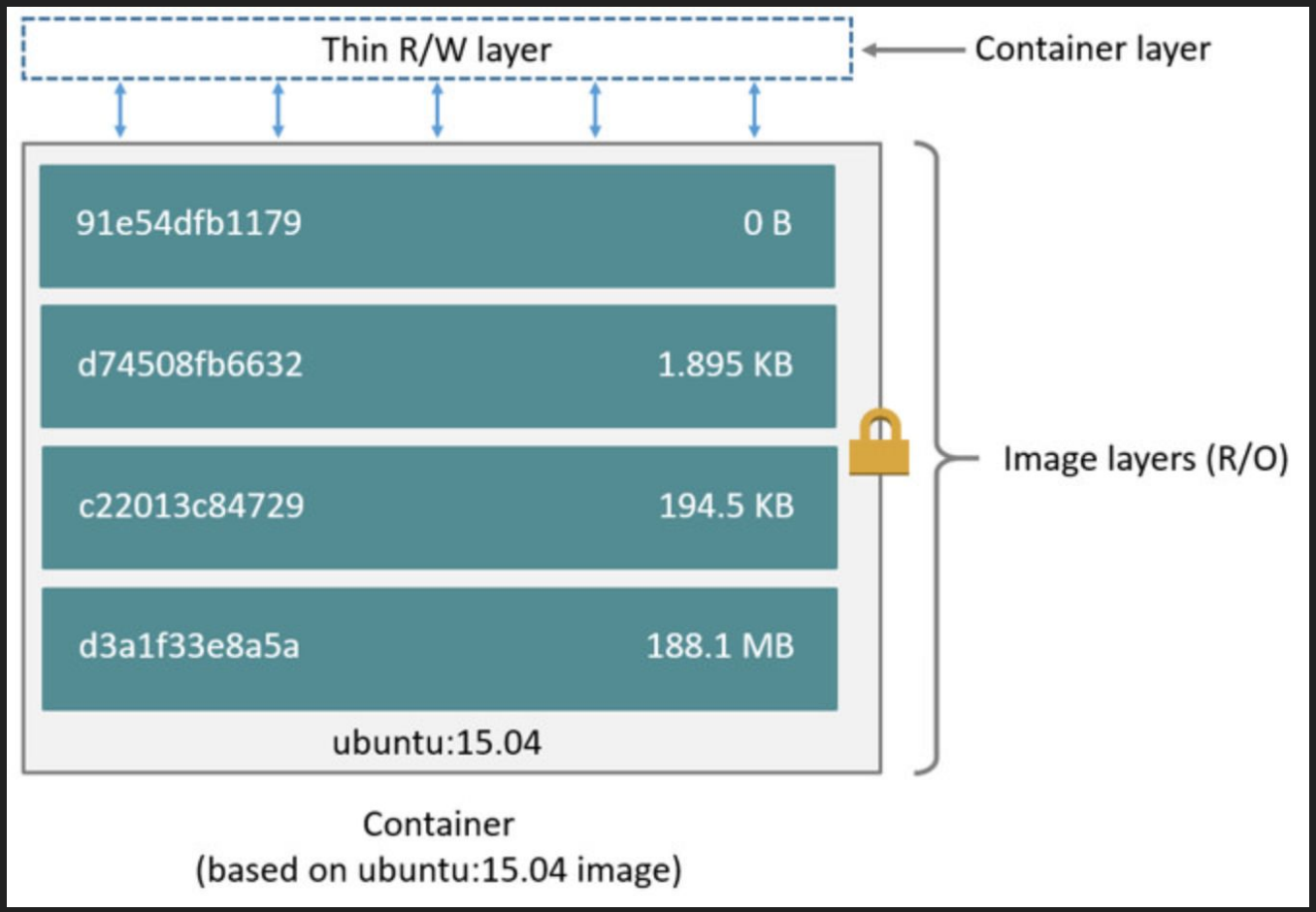
```
FROM openjdk:16-alpine3.13

ADD HelloWorld.java

RUN javac HelloWordl.java

CMD ["java", "HelloWorld"]
```

```
$> docker build -t java-hello-docker:demo .

$> docker run java-hello-docker:demo
```

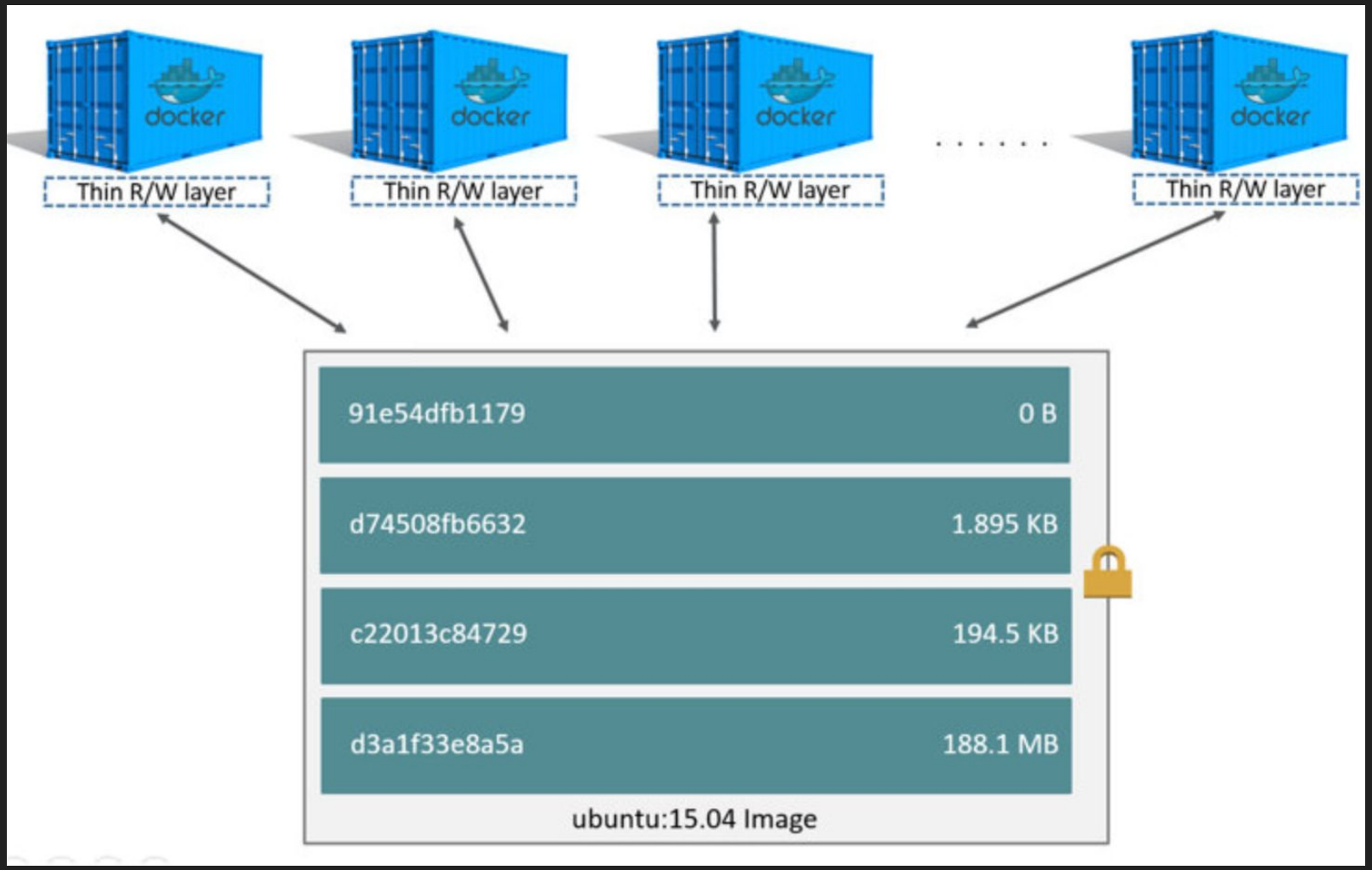
Механизмы Linux, используемые docker

- namespaces (пространства имен) - изоляция и виртуализация глобальных системных ресурсов между процессами
  - прекращает свое существование после окончания работы PID1
  - pid - изоляция процессов (process ID)
  - net - сетевые интерфейсы
  - ipc - IPC-ресурсы (межпроцессное взаимодействие: семафоры, разделяемая память и очереди сообщений)
  - mount - дерево файловой системы
  - uts - изоляция ядра и идентификаторов версий, системных вызовов uname(). (Unix Timesharing System)
  - user - ID пользователей и групп, корневой каталог, ключи и capabilities
- cgroups (контрольные группы)
  - Установка лимитов на ресурсы. Например, ограничение памяти, доступной для контейнера
- UnionFS - “слоеная” файловая система

Docker unionfs



Docker unionfs



Docker union fs

Пример номер раз

Ссылка тут

Docker union fs

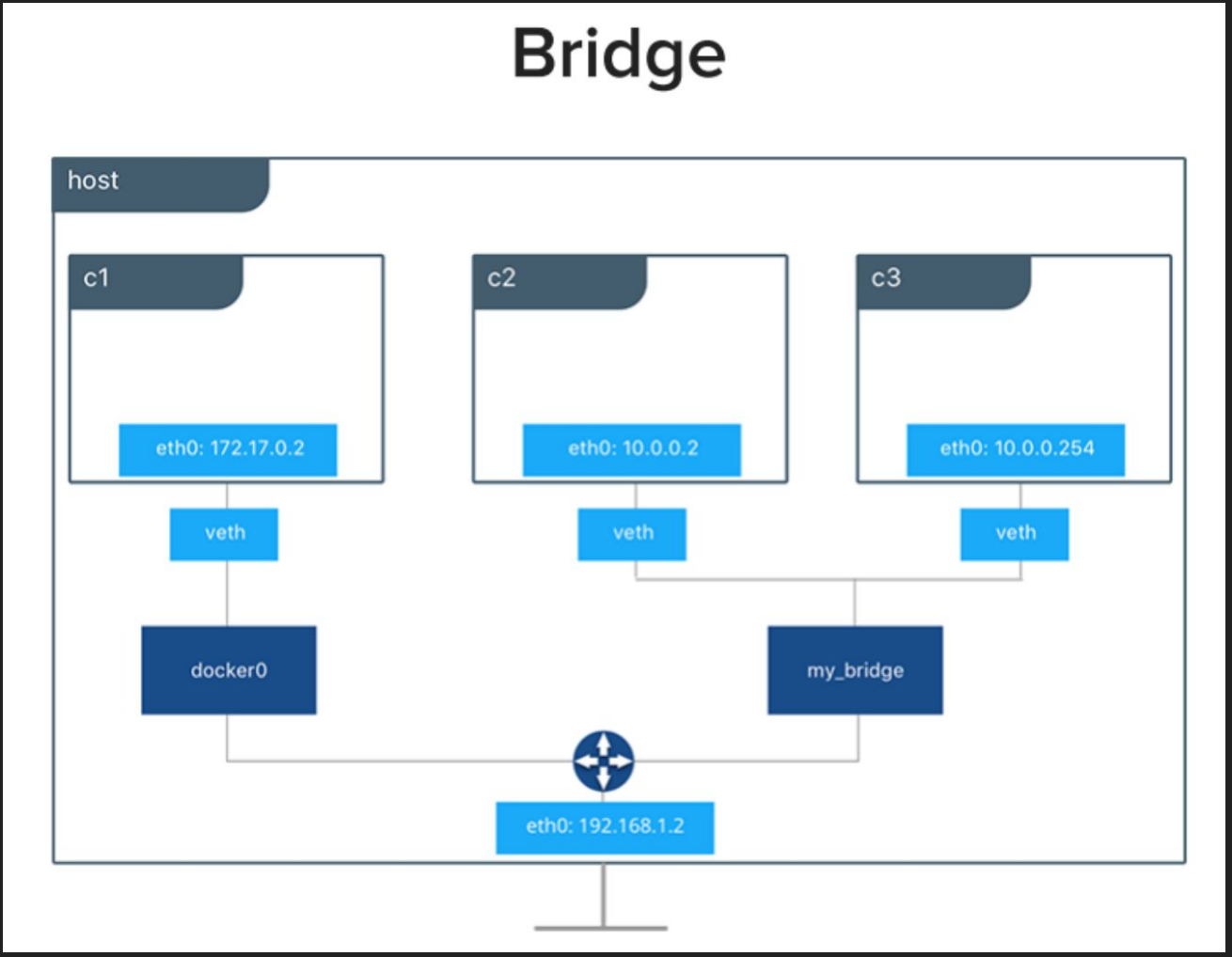
Пример номер два

Ссылка тут

Docker сеть Bridge network

Если нужно отделить контейнер или группу контейнеров
Контейнер может быть подключен к нескольким Bridge сетям (без рестарта)
Работает Service Discovery
Произвольные диапазоны IP-адресов

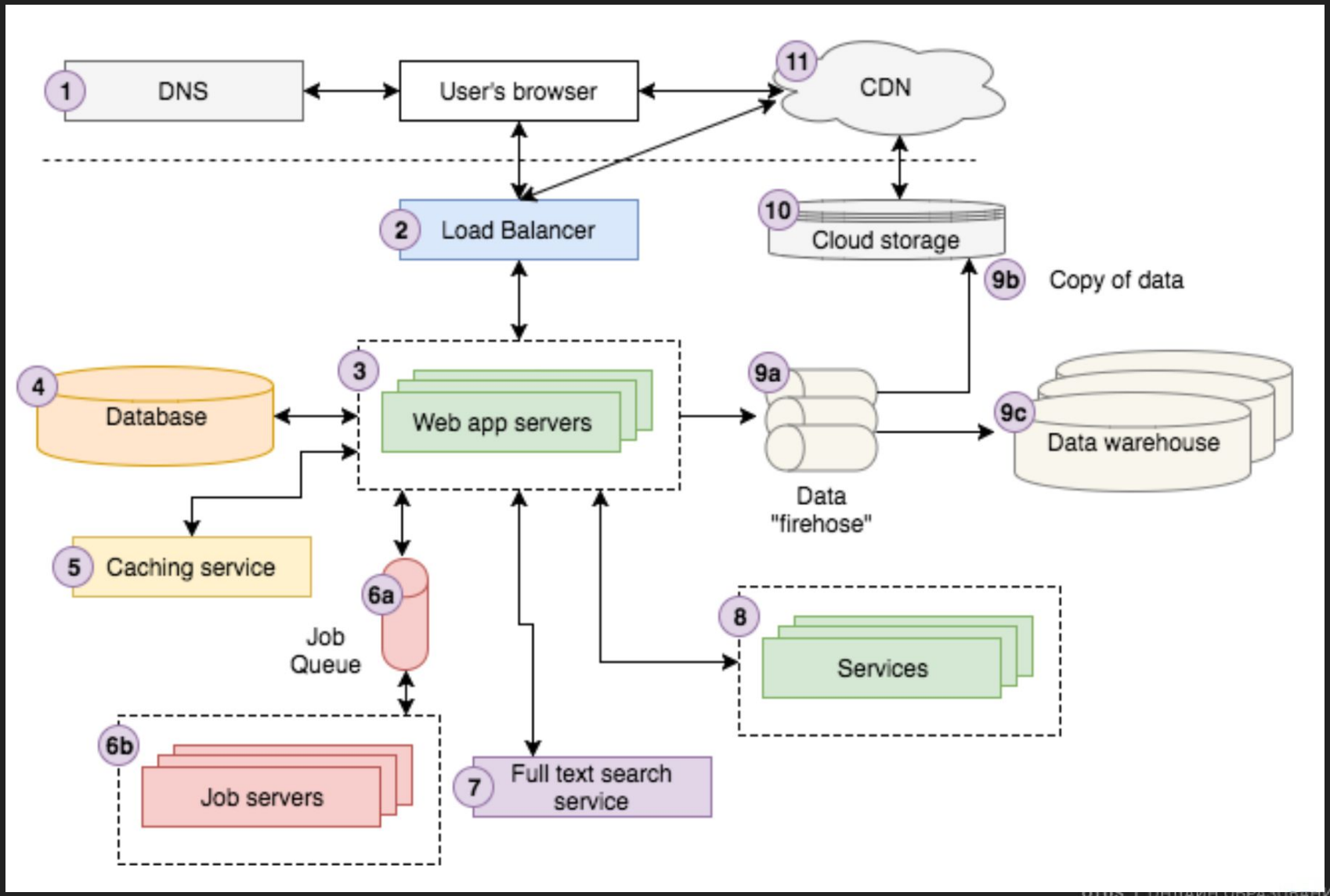
Docker сеть Bridge network



Docker полезности

1. Смотрим офф документация
2. Не стесняемся пользоваться hadolint
3. Multistage наше все
4. Еще есть Dive
5. Поставим автокомплит для докера
6. Активно используем docker-compose
7. Смотрим что и как добавляем в образ
8. Не творим ДИЧЬ!!!!

Modern Web-App Architecture



Docker Compose

Позволяет настраивать и разворачивать несколько контейнеров одновременно.

```
version: 'версия'
networks:
  сети
volumes:
  хранилища
services:
  контейнеры
```

Docker Compose

```
version: "3.9"
services:
 grafana:
   image: grafana/grafana:8.0.6-ubuntu
   ports:
     - "3000:3000"
   volumes:
     - grafana-data:/var/lib/grafana
     - grafana-configs:/etc/grafana
 prometheus:
   image: prom/prometheus:v2.28.1
   ports:
     - "9090:9090"
   volumes:
     - prom-data:/prometheus
     - prom-configs:/etc/prometheus
```

```
node-exporter:
   image: prom/node-exporter:v1.2.0
   ports:
     - "9100:9100"
   volumes:
     - /proc:/host/proc:ro
     - /sys:/host/sys:ro
     - /:/rootfs:ro
   command:
     - '--path.procfs=/host/proc'
     - '--path.sysfs=/host/sys'
     -
'--collector.filesystem.mount-points-exclude'
     -
'^/(sys|proc|dev|host|etc|rootfs/var/lib/docker/con
tainers|rootfs/var/lib/docker/overlay2|rootfs/run/d
ocker/netns|rootfs/var/lib/docker/aufs)($$|/)'
volumes:
 grafana-data:
 grafana-configs:
 prom-data:
 prom-configs:
```

Список материалов для изучения

1. Docker for beginners
2. Build your java image
3. Справочник по Dockerfile
4. Видео: Docker и Java
5. Видео: Механизмы контейнеризации Linux

6. Web Architecture 101
7. Видео: Архитектура Web-приложения