Docker

Содержание

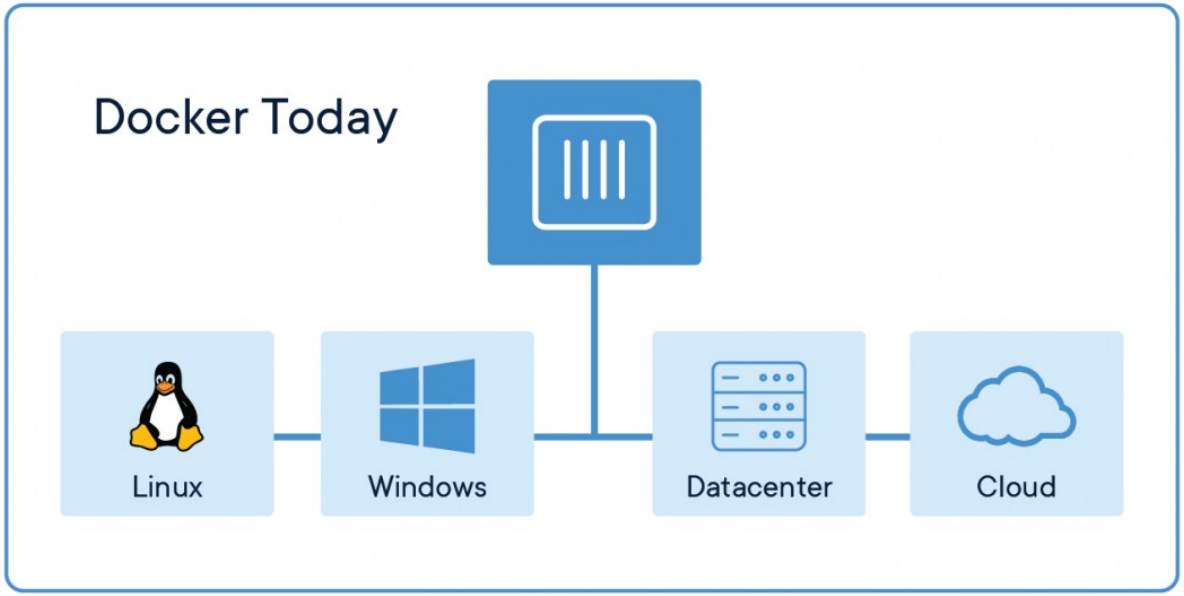
- Обзор
- Docker vs Vitual Machines
- Преимущества
- Установка
- Docker Hub
- Команды для работы с Docker
- Создание образа
- Dockerfile
- Запуск GUI приложения внутри контейнера

Docker

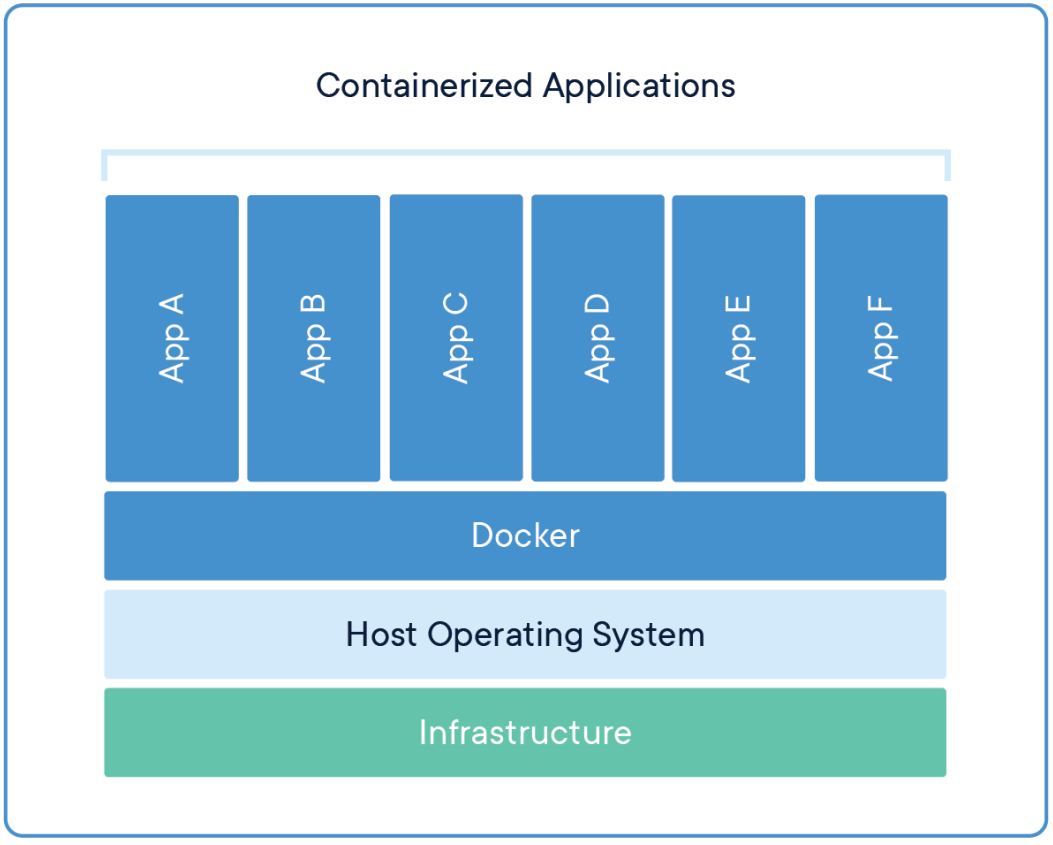
Docker — программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации.

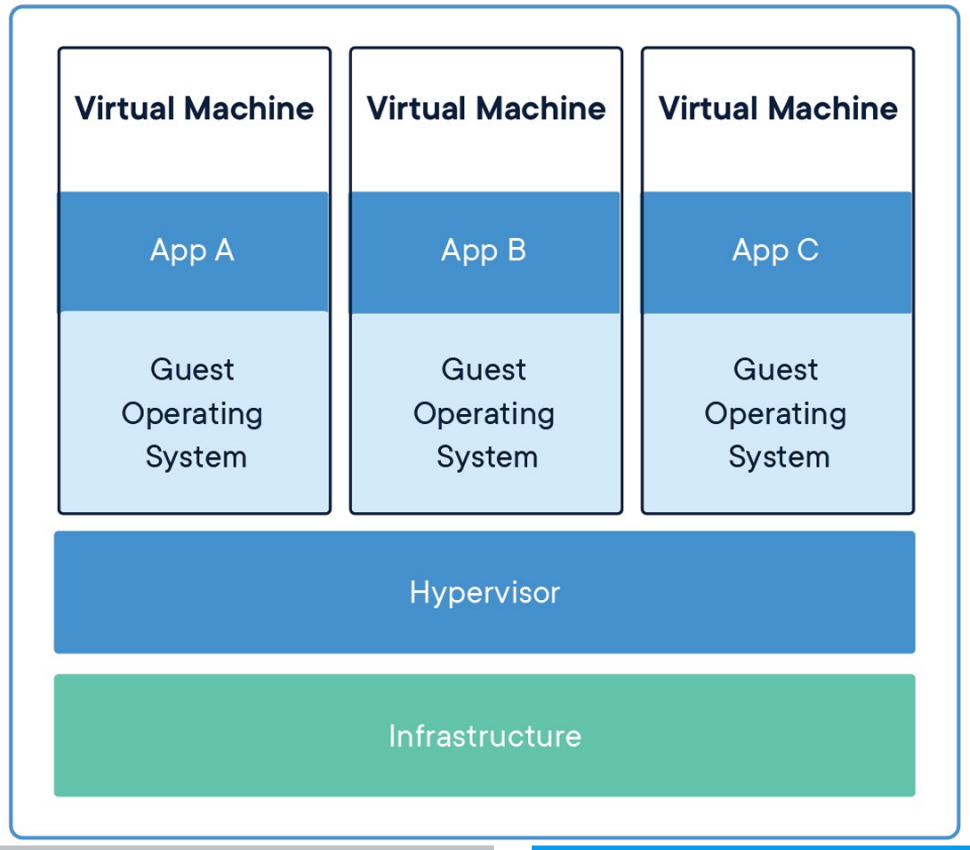
Написано на: Go

Дата выхода: 20 марта 2013 г



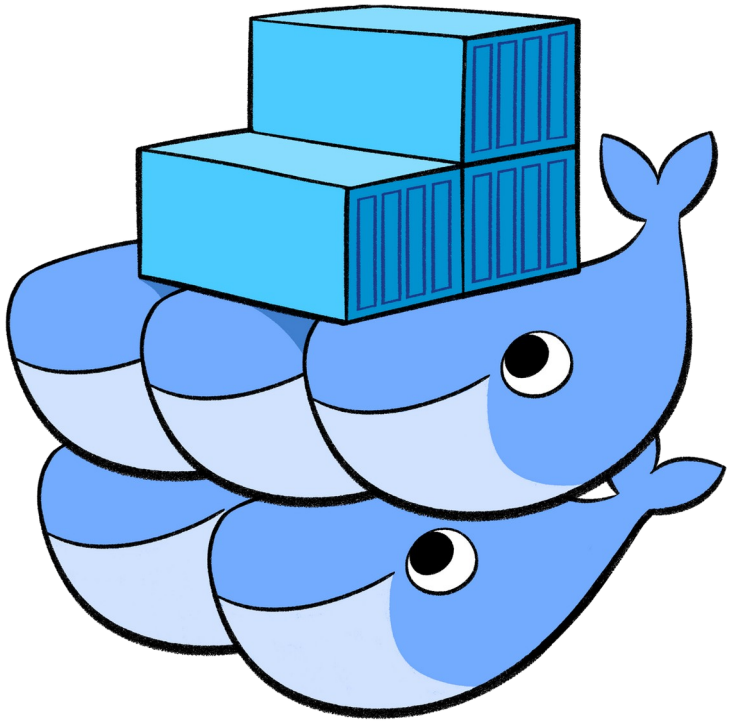
Docker vs Vitual Machines





Docker основные преимущества

- **Изоляция**
- **Модульность**
- **Маштабирование**
- **Скорость**
- **Версионность**

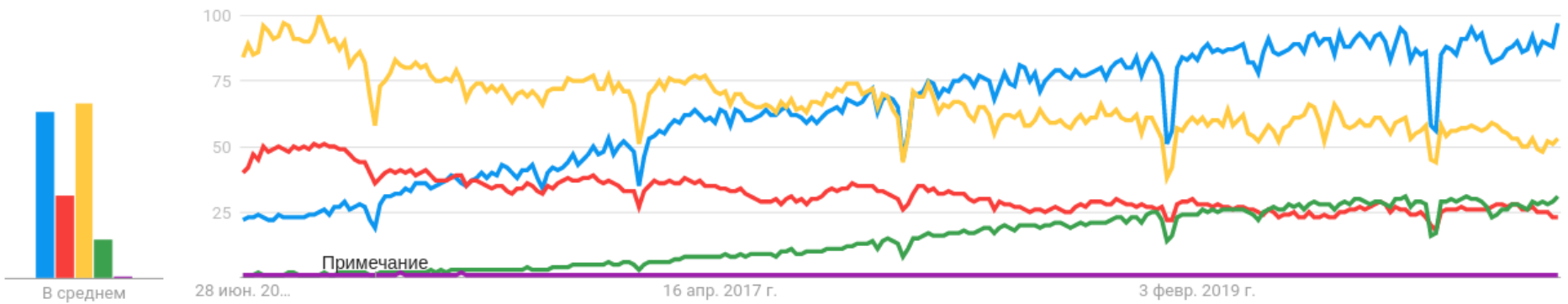


Динамика популярности

Docker	VirtualBox	VMware	Kubernetes	LXC
Поисковый запрос	Поисковый запрос	Поисковый запрос	Поисковый запрос	Поисковый запрос

По всему миру ▾ Последние 5 лет ▾ Все категории ▾ Веб-поиск ▾

Динамика популярности



Динамика популяности. Программное обеспечение

Docker
Поисковый запрос

VirtualBox
Поисковый запрос

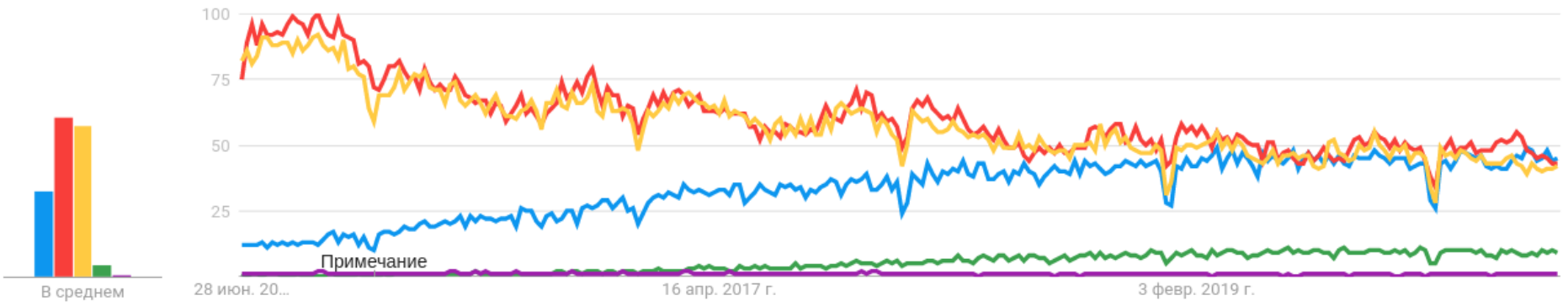
VMware
Поисковый запрос

Kubernetes
Поисковый запрос

LXC
Поисковый запрос

По всему миру | Последние 5 лет | Программное обеспечение | Веб-поиск

Динамика популярности



Установка Docker. Ubuntu

Используйте команду apt для установки пакета docker.io:

```
sudo apt install docker.io
```

Запустите docker:

```
sudo systemctl enable --now docker
```

[Опциоанльно] предоставьте пользователю права администратора для докера:

```
sudo usermod -aG docker USERNAME
```

Check docker version:

```
docker --version
```

Hello world

Для запуска контенера hello-world выполните команду:

sudo docker run hello-world

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:d58e752213a51785838f9eed2b7a498ffa1cb3aa7f946dda11af39286c3db9a9
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Docker Hub

Docker Hub - крупнейший в мире публичный репозиторий образов контейнернов

Сайт: https://hub.docker.com/

Для создания репозитория нам нужно зарегистрироваться в Docker Hub

Вы можете создать репозиторий, используя свой Docker ID

Команды для работы с Docker. Помощь

Список локальных оброзов:

docker image ls

Список контейнеров:

docker container ls --all

Узнать справочную информацию можно при помощи команды help. Примеры использования команды help:

docker –-help

docker container --help

docker container ls --help

docker run --help

Запуск ОС Ubuntu в Docker

Для запуска контейнера с Ubuntu выполнинте команду:

sudo docker run --interactive --tty ubuntu bash

(или) sudo docker run -it ubuntu bash

```
[sudo] password for evgeny:
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
a4a2a29f9ba4: Pull complete
127c9761dcba: Pull complete
d13bf203e905: Pull complete
4039240d2e0b: Pull complete
Digest: sha256:35c4a2c15539c6c1e4e5fa4e554dac323ad0107d8eb5c582d6ff386b383b7dce
Status: Downloaded newer image for ubuntu:latest
```

Выполните команды hostname и uname -a

```
root@370e10d4b26c:/# hostname
370e10d4b26c
root@370e10d4b26c:/# uname -a
Linux 370e10d4b26c 5.4.0-37-generic #41-Ubuntu SMP Wed Jun 3 18:57:02 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
root@370e10d4b26c:/#
```

Запуск web-сервера в Docker

Для запуска web-сервера выполните следующую команду:

sudo docker run --detach --publish 80:80 --name webserver nginx

```
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
8559a31e96f4: Pull complete
8d69e59170f7: Pull complete
3f9f1ec1d262: Pull complete
d1f5ff4f210d: Pull complete
1e22bfa8652e: Pull complete
Digest: sha256:21f32f6c08406306d822a0e6e8b7dc81f53f336570e852e25fbe1e3e3d0d0133
Status: Downloaded newer image for nginx:latest
823a6c5fd61e1383bf81101af94485946e0d59f7d325426d90febdb9c27e22bd
```

Остановка, повторныый запуск и удаление

Для остановки выполнить команду

sudo docker container stop webserver

Для повторного запуска:

sudo docker container start webserver

Для удаления контейнера:

sudo docker container rm webserver

Для удаления образа:

sudo docker rmi nginx

Взаимодействие с контейнерами

Docker предоставляет несколько инструментов, которые позволяют взаимодействовать с запущенными контейнерами

attach

sudo docker container attach <name>

exec

sudo docker container exec -it <name> /bin/bash

Журналы и обработка информации

Docker предоставляет несколько команд, позволяющих просматривать информацию о контейнерах без использования команд attach или exec

logs

sudo docker container logs --tail 5 <name>

sudo docker container logs -f <name>

top

sudo docker container top <name>

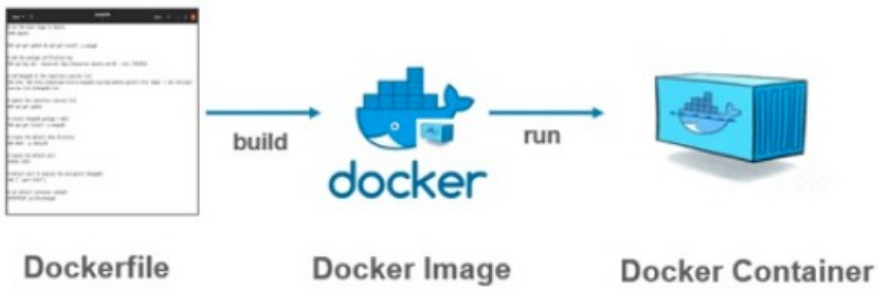
stats

sudo docker container stats <name>

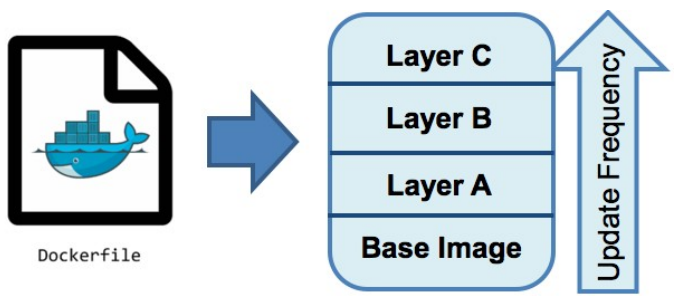
sudo docker container stats -a

Dockerfile

Dockerfile – это текстовый файл, который состоит из последовательности команд и аргументов, необходимых для создания образа.



Каждый Docker-образ состоит из слоёв (layers), каждый из которых описывает какую-то инструкцию.

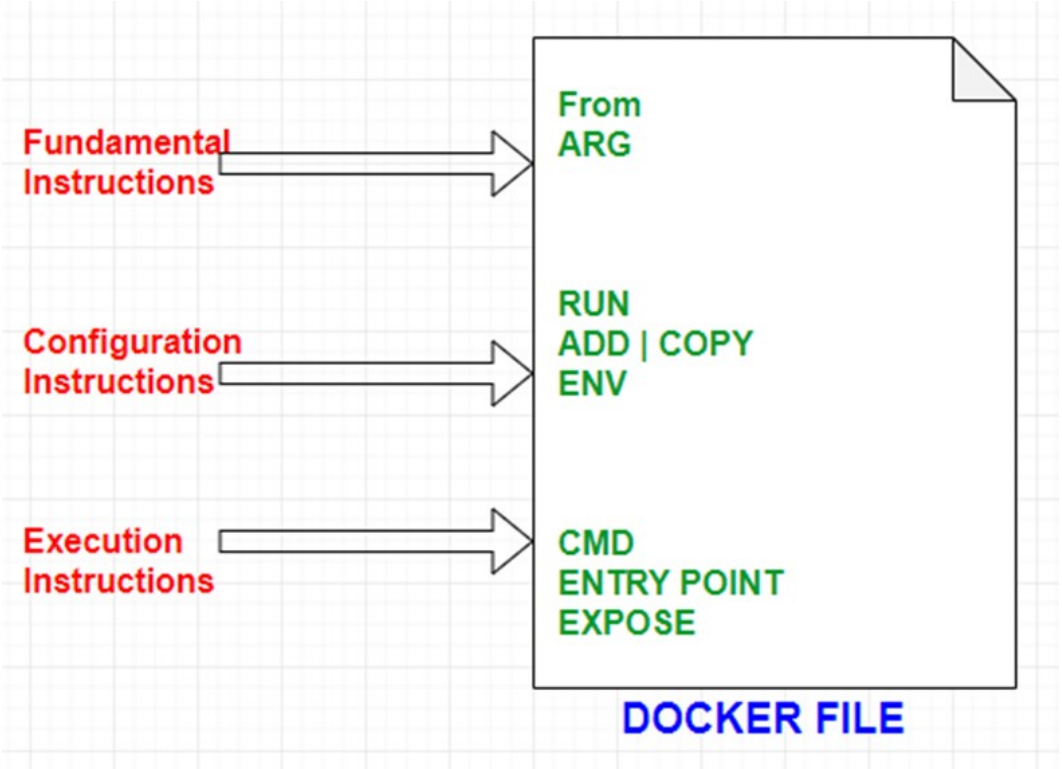


Dockerfile. Пример

```
FROM alpine:latest
LABEL maintainer="Russ McKendrick <russ@mckendrick.io>"
LABEL description="This example Dockerfile installs NGINX."
RUN apk add --update nginx && \
rm -rf /var/cache/apk/* && \
mkdir -p /tmp/nginx/
COPY files/nginx.conf /etc/nginx/nginx.conf
COPY files/default.conf /etc/nginx/conf.d/default.conf
ADD files/html.tar.gz /usr/share/nginx/
EXPOSE 80/tcp
ENTRYPOINT ["nginx"]
CMD ["-g", "daemon off;"]
```

Инструкции Dockerfil-a

- **FROM**
- **LABEL**
- **RUN**
- **COPY and ADD**
- **EXPOSE**
- **ENTRYPOINT and CMD**



Создание образа docker контейнера

sudo docker image build --help

```
Options:
      --add-host list           Add a custom host-to-IP mapping (host:ip)
      --build-arg list          Set build-time variables
      --cache-from strings      Images to consider as cache sources
      --cgroup-parent string    Optional parent cgroup for the container
      --compress                Compress the build context using gzip
      --cpu-period int          Limit the CPU CFS (Completely Fair
                                Scheduler) period
      --cpu-quota int           Limit the CPU CFS (Completely Fair
                                Scheduler) quota
  -c, --cpu-shares int          CPU shares (relative weight)
      --cpuset-cpus string      CPUs in which to allow execution (0-3, 0,1)
      --cpuset-mems string      MEMs in which to allow execution (0-3, 0,1)
      --disable-content-trust   Skip image verification (default true)
  -f, --file string             Name of the Dockerfile (Default is
                                'PATH/Dockerfile')
      --force-rm                Always remove intermediate containers
      --iidfile string          Write the image ID to the file
      --isolation string        Container isolation technology
      --label list              Set metadata for an image
  -m, --memory bytes            Memory limit
      --memory-swap bytes       Swap limit equal to memory plus swap:
                                '-1' to enable unlimited swap
      --network string          Set the networking mode for the RUN
                                instructions during build (default "default")
      --no-cache                Do not use cache when building the image
      --pull                    Always attempt to pull a newer version of
                                the image
  -q, --quiet                   Suppress the build output and print image
                                ID on success
      --rm                      Remove intermediate containers after a
                                successful build (default true)
      --security-opt strings    Security options
      --shm-size bytes          Size of /dev/shm
  -t, --tag list                Name and optionally a tag in the
                                'name:tag' format
      --target string           Set the target build stage to build.
      --ulimit ulimit           Ulimit options (default [])
```

Использование Dockerfile для создания образа контейнера. Практика

Общий вид команды для создания образа из Dockerfile:

sudo docker image build --file <path_to_Dockerfile> --tag <REPOSITORY>:<TAG> .

Шаг 1

Созададим образ, выполнив команду:

sudo docker image build --tag local:dockerfile-example .

Шаг 2

Убедимся доступен ли собранный образ:

sudo docker image ls

Использование Dockerfile для создания образа контейнера. Практика

Шаг 3

Запустим контейнер с только что созданным образом:

```
docker container run -d --name dockerfile-example -p 8080:80 local:dockerfile-example
```

Шаг 4

Проверим, что контенер запущен:

```
docker container ls
```

Шаг 5

Открыть броузер и перейти по адресу http://localhost:8080/

Использование Dockerfile для создания образа контейнера. Практика

Шаг 6

Отоброзим метки, которые мы встроили при сбокре образа:

```
sudo docker image inspect -f {{.Config.Labels}} local:dockerfile-example
```

Шаг 7

Остановим и удалим запущенный контейнер:

```
sudo docker container stop dockerfile-example
sudo docker container rm dockerfile-example
```

Использование существующего контенера для создания образа. Практика

Шаг 1

Загрузим образ, который хотим использовать в качестве базы:

```
sudo docker image pull alpine:latest
```

Шаг 2

Запустим контейнер, чтобы мы могли взаимодействовать с ним:

```
sudo docker container run -it --name alpine-test alpine /bin/sh
```

Использование существующего контенера для создания образа. Практика

Шаг 3

Как только контейнер запустится, добавим пакеты, используя команду apk:

```
apk update
apk upgrade
apk add --update nginx
rm -rf /var/cache/apk/*
mkdir -p /tmp/nginx/
exit
```

Шаг 4

Сохраним наш контенер:

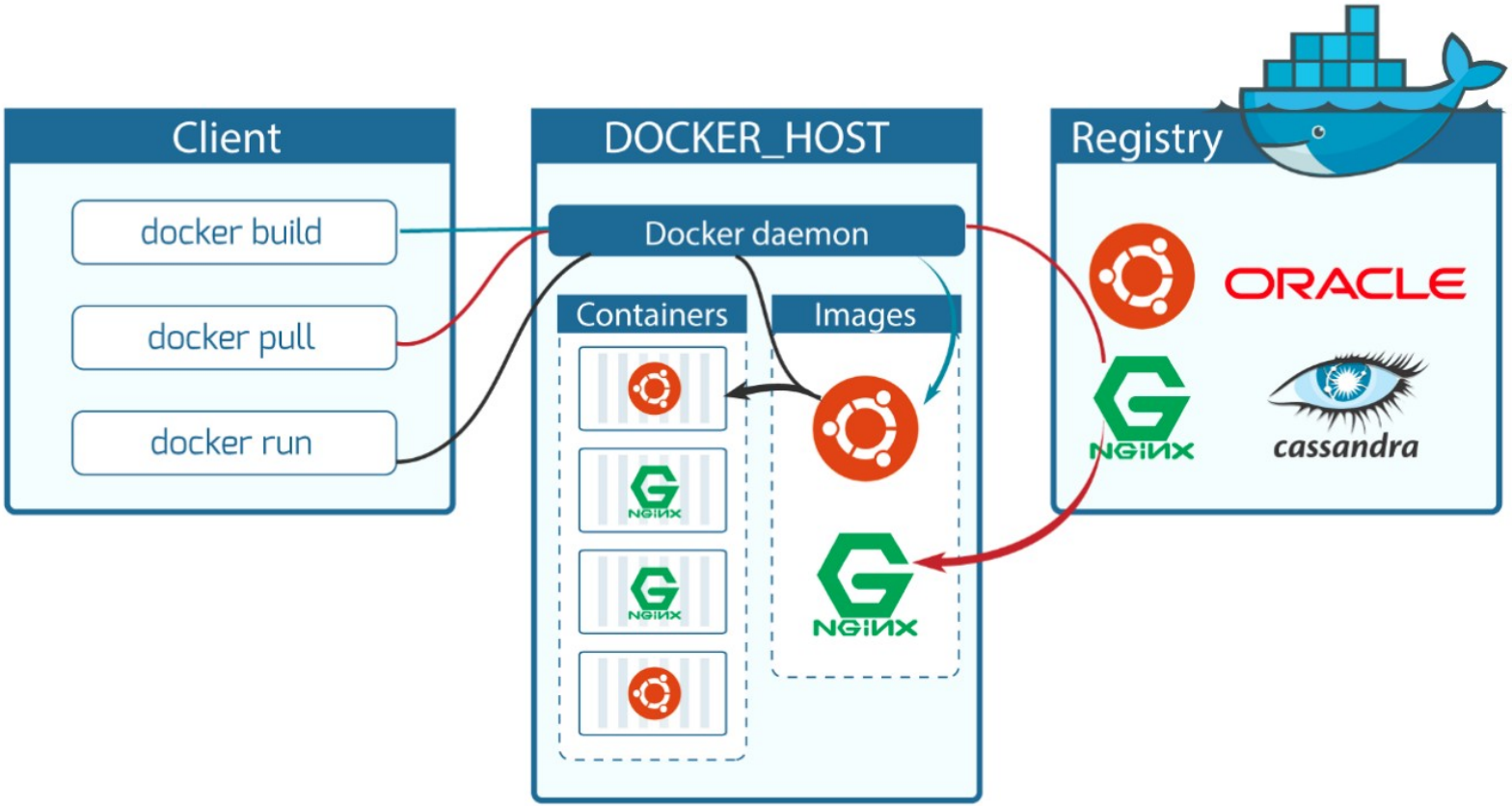
```
docker container commit alpine-test local:broken-container
```

Шаг 5

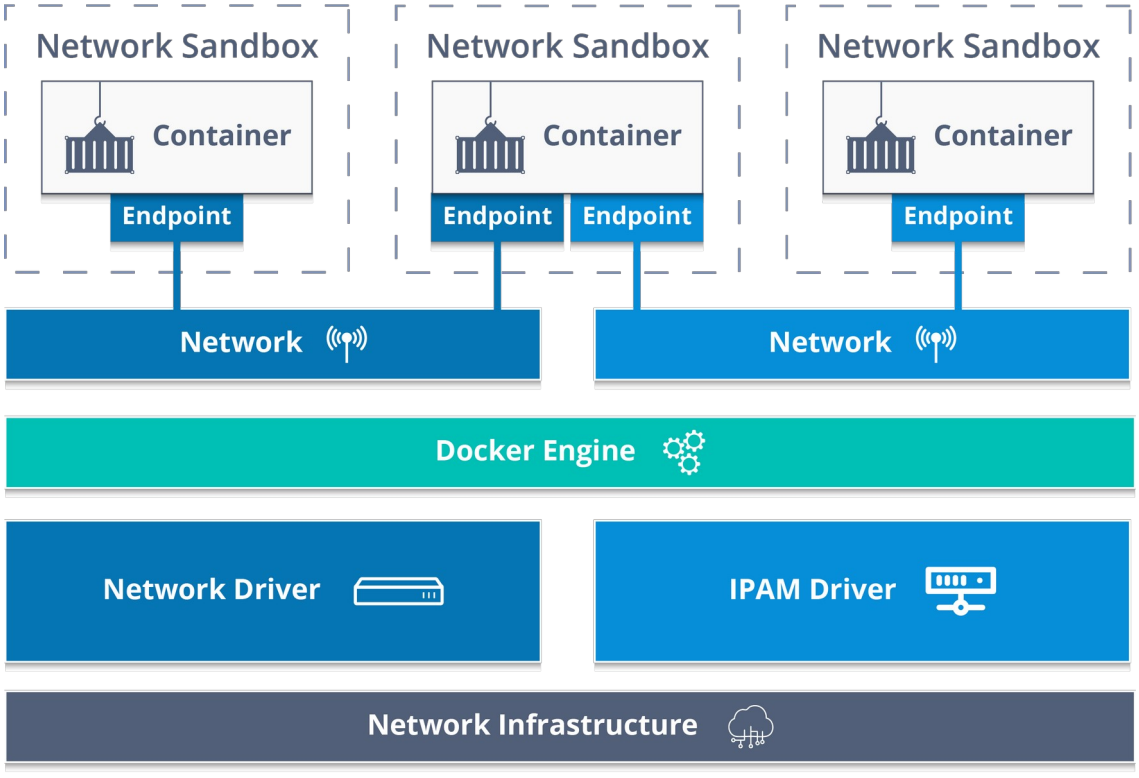
Сохраним файл образа:

```
docker image save -o broken-container.tar local:broken-container
```

Docker components



Docker networking



Architecture of Container Networking Model

Docker networking. Практика

Шаг 1

Загрузим образы контейнеров, которые мы будем использовать, а также создадим сеть:

sudo docker image pull redis:alpine

sudo docker image pull russmckendrick/moby-counter

sudo docker network create moby-counter

Шаг 2

Запускаем контейнер с Redis, затем контейнер moby-counter

sudo docker container run -d --name redis --network moby-counter redis:alpine

sudo docker container run -d --name moby-counter --network moby-counter -p 8080:80 russmckendrick/moby-counter

Шаг 3

Открыть броузер и перейти по адресу http://localhost:8080/

Docker networking. Практика

Шаг 4

Пропинговать Redis контейнер, через moby-counter:

sudo docker container exec moby-counter ping -c 3 redis

Шаг 5

Проверим файл /etc/hosts и /etc/resolve.conf:

sudo docker container exec moby-counter cat /etc/hosts

sudo docker container exec moby-counter cat /etc/resolv.conf

Шаг 6

Выполнитм DNS lookup, выполнив команду:

sudo docker container exec moby-counter nslookup redis 127.0.0.11

Docker networking. Практика

Шаг 7

Создадим еще одну сеть и запустим еще один moby-counter:

sudo docker network create moby-counter2

sudo docker run -itd --name moby-counter2 --network moby-counter2 -p 9090:80 russmckendrick/moby-counter

Шаг 8

Попробуем пропинговать контейнер redis из moby-ounter2 и проверим файл /etc/resolve.conf:

sudo docker container exec moby-counter2 ping -c 3 redis

sudo docker container exec moby-counter2 cat /etc/resolv.conf

Шаг 9

Выполним команду nslookup из контейнера moby-counter2:

sudo docker container exec moby-counter2 nslookup redis 127.0.0.11

Docker networking. Практика

Шаг 10

Запустим второй контейнер redis:

sudo docker container run -d --name redis2 --network moby-counter2 –network-alias redis redis:alpine

Шаг 11

Выполним команду nslookup из контейнера moby-counter2:

sudo docker container exec moby-counter2 nslookup redis 127.0.0.11

Шаг 12

Откроем броузер и убедимся, что оба приложения доступны по адресам http://localhost:8080/ и http://localhost:9090/

Docker networking. Практика

Шаг 13

Больше информации о конфигурации сети можно узнать при помощи docker network inspect:

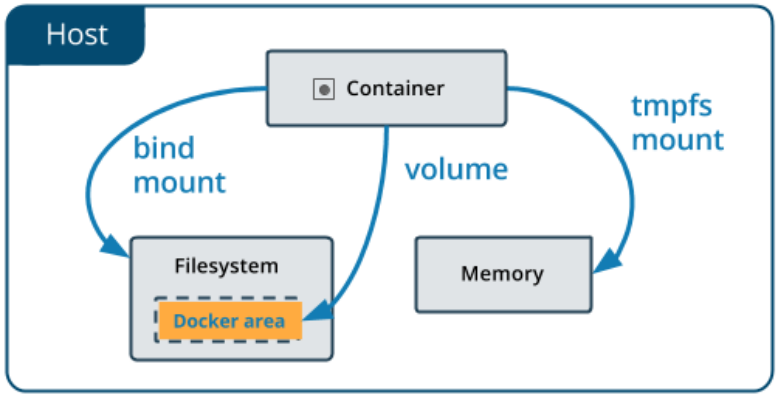
```
docker network inspect moby-counter
```

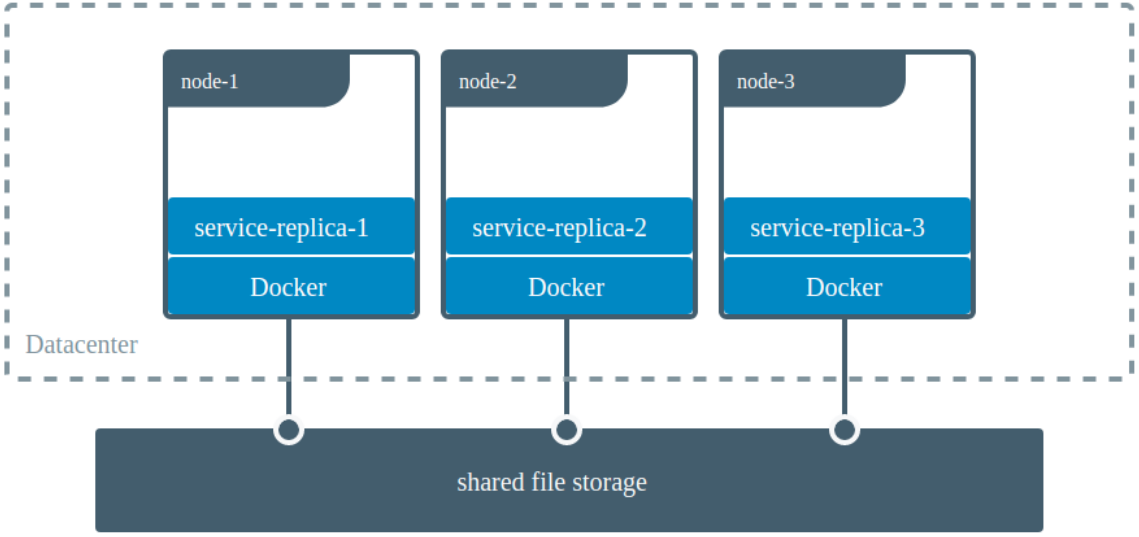
Шаг 14

Удалим один из контейнеров moby-counter, redis и сеть:

```
sudo docker container stop moby-counter2 redis2
sudo docker container prune
sudo docker network prune
```

Docker volumes





Docker volumes. Практика

Шаг 1

Остановим и удалим контейнер redis:

sudo docker container stop redis

sudo docker container rm redis

Шаг 2

Посмотрим какие тома созданы:

sudo docker volume ls

Шаг 3

Используем volume ID при следующем запуске контейнера Redis:

sudo docker container run -d --name redis -v <volume ID>:/data --network moby-counter redis:alpine

Docker volumes. Практика

Шаг 4

Посмотрим содержимое тома:

sudo docker container exec redis ls -lhat /data

Шаг 5

Переопределить volume своим собственным:

sudo docker volume create redis_data

Шаг 6

Удалим контейнер redis, и выполним следующую команду:

sudo docker container stop redis

sudo docker container rm redis

sudo docker container run -d --name redis -v redis_data:/data --network moby-counter redis:alpine

Docker volumes. Практика

Шаг 7

Посмотрим допоплнительную информацию о томе с помощью команды inspect:

```
sudo docker volume inspect redis_data
```

Шаг 8

Удалим контейнер moby-counter, redis и сеть:

```
sudo docker container stop redis moby-counter
sudo docker container prune
sudo docker network prune
```

Шаг 9

Удалим тома, выполнив следующую команду:

```
sudo docker volume prune
```

Запуск GUI приложения внутри контейнера

Разрешить доступ к X-серверу через xhost:

sudo xhost +local

Смонтировать X-Server Unix-domain сокета и передайте переменную окружения DISPLAY:

sudo docker run -e DISPLAY -v /tmp/.X11-unix:/tmp/.X11-unix <image>