DevOps

Л08. Docker basics

Виктор Моисеев
+7-902-83-145-30
t.me/v_paranoid
victorparanoid@gmail.com

Л08. Docker basics

Способы запуска кода

Docker intro

Как запускать контейнеры

Как собирать образы

Как презентовать ресурсы контейнерам

Способы запуска приложений (варианты виртуализации)

App / OS / Physical Server (baremetal) [App-app isolation - memmaps]

Baremetal hw virtualization

App / JavaVM(runtime) etc / OS / baremetal

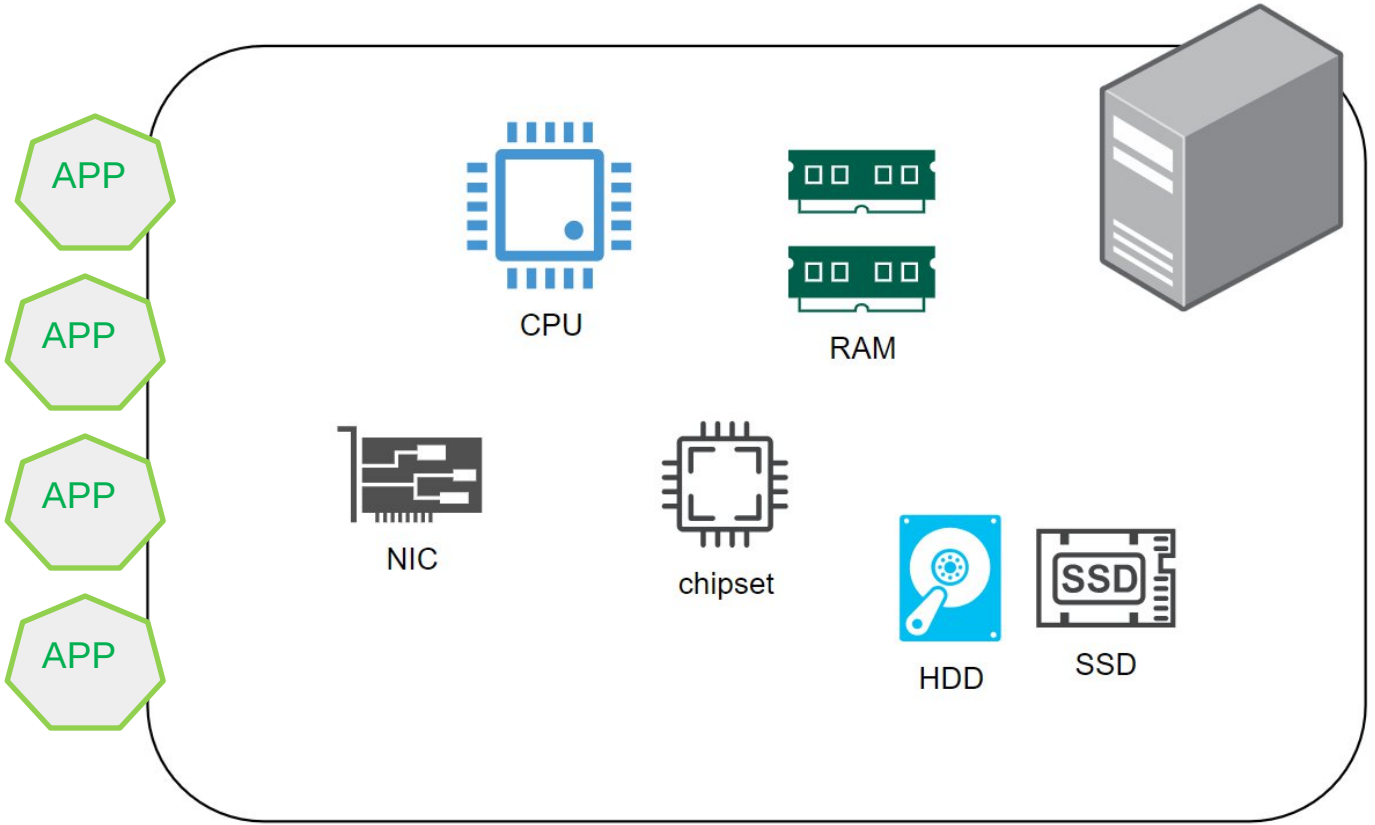
App / VM / hypervisor App / OS / baremetal

App / VM / hypervisor OS / baremetal [cpu virt support]

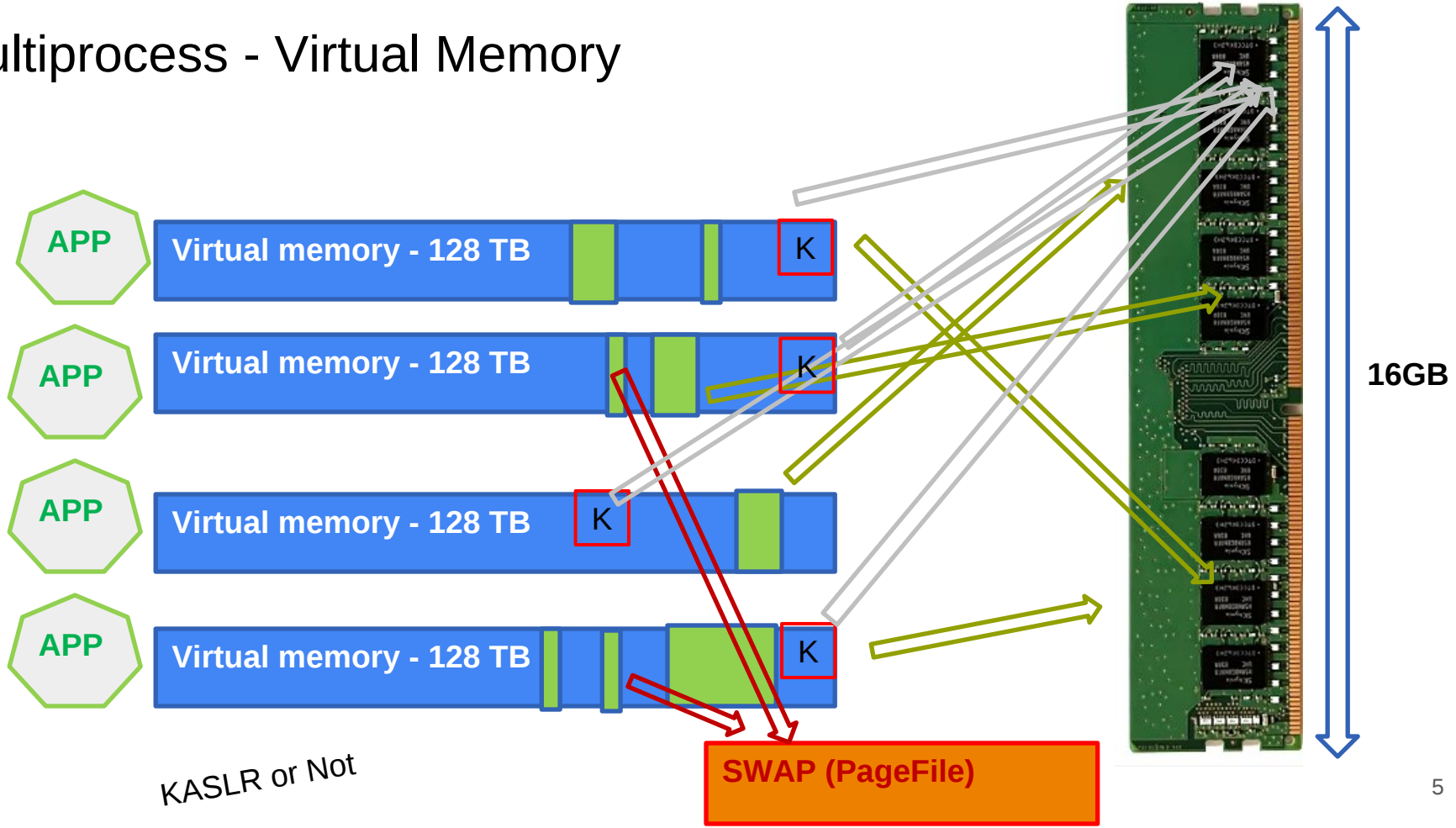
App / container / OS / baremetal

“Serverless” FaaS (AWS Lambda etc)

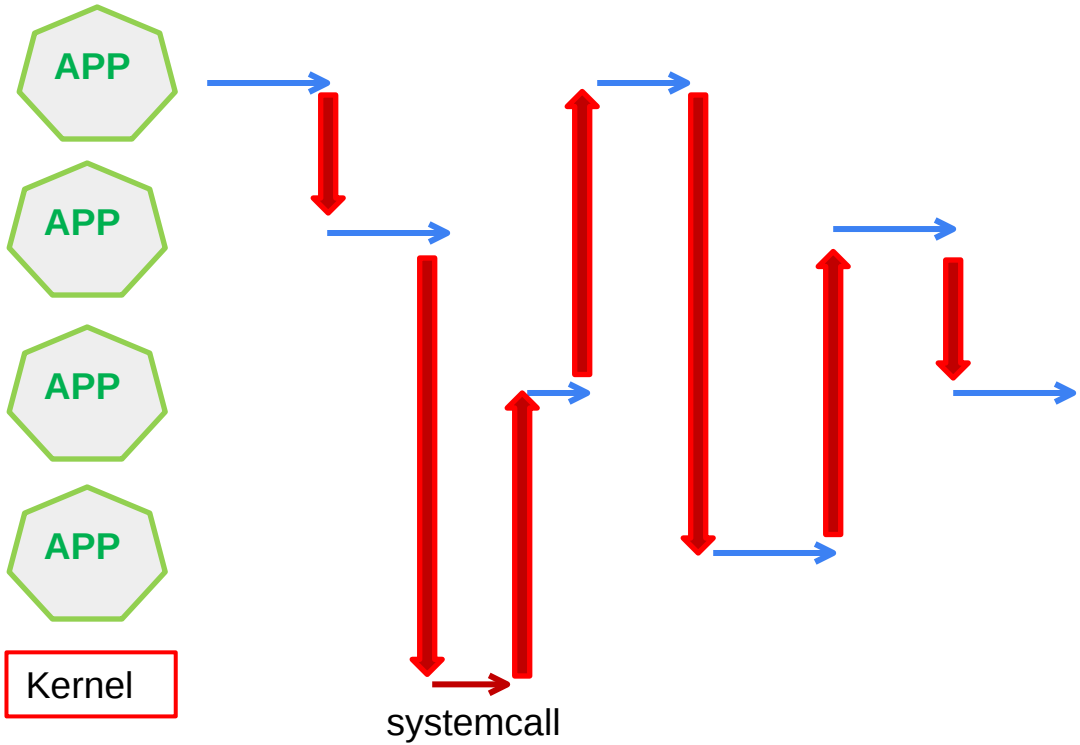
Способы запуска приложений (варианты виртуализации)



Multiprocess - Virtual Memory



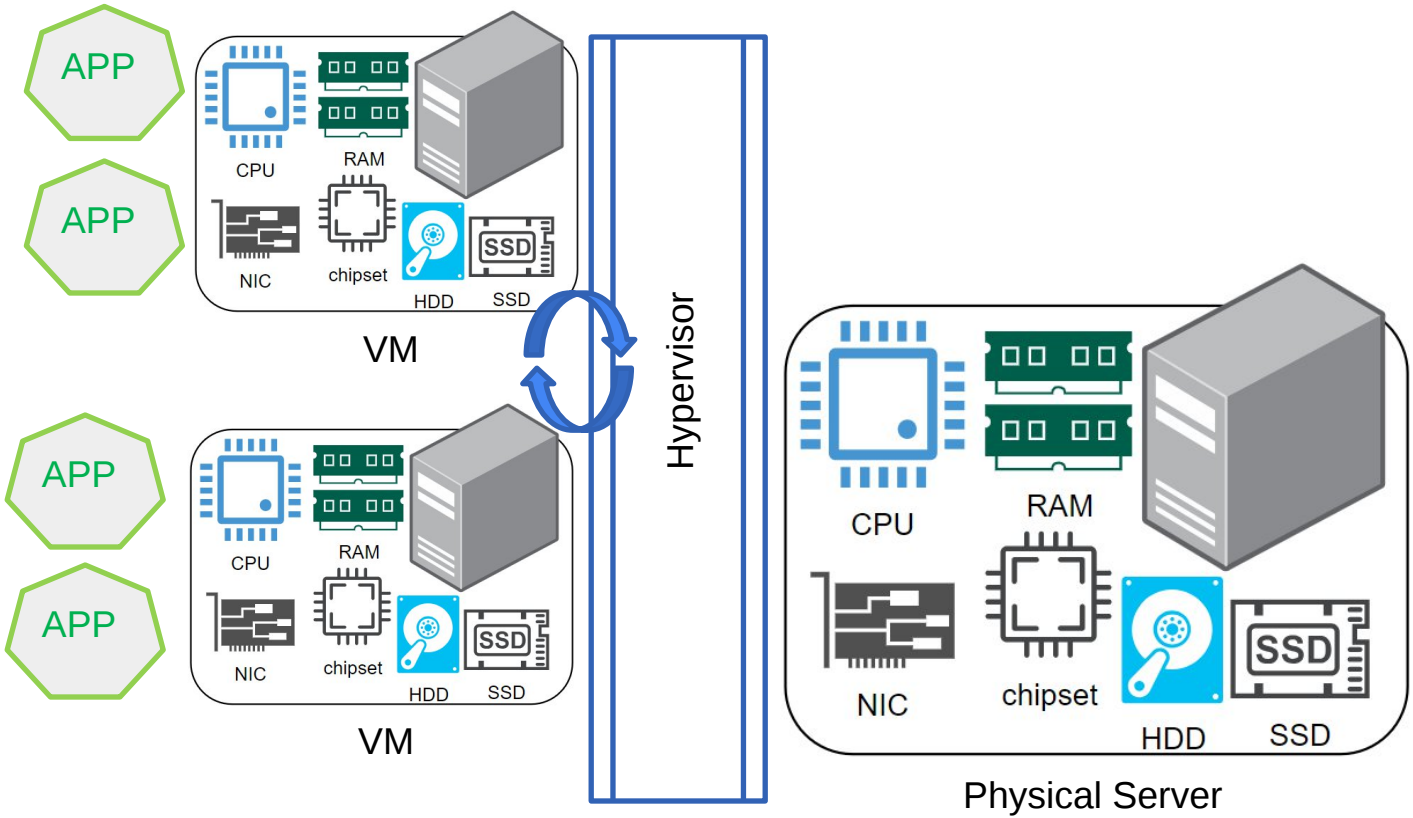
Multiprocess – CPU (context switching)



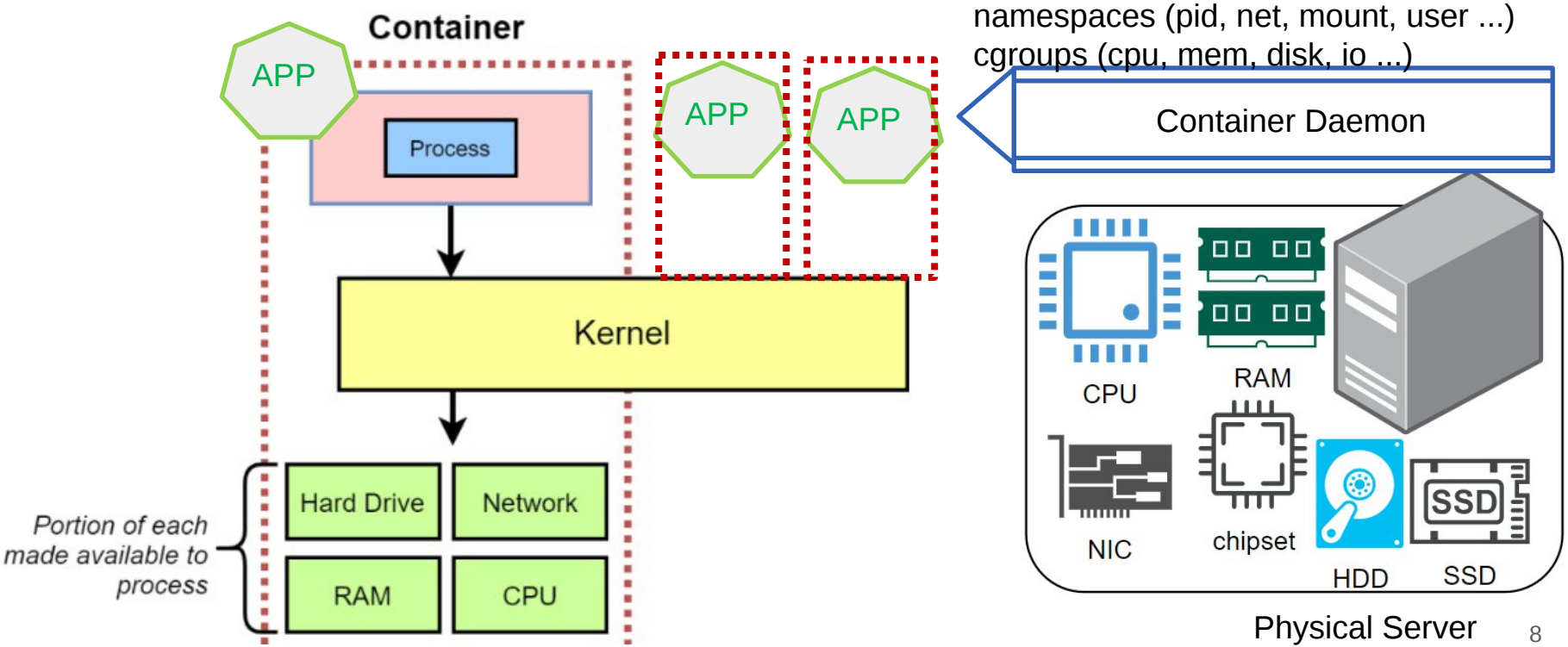
context storage

HW/SW interrupts

Hardware virtualization



Способы запуска приложений (варианты виртуализации)

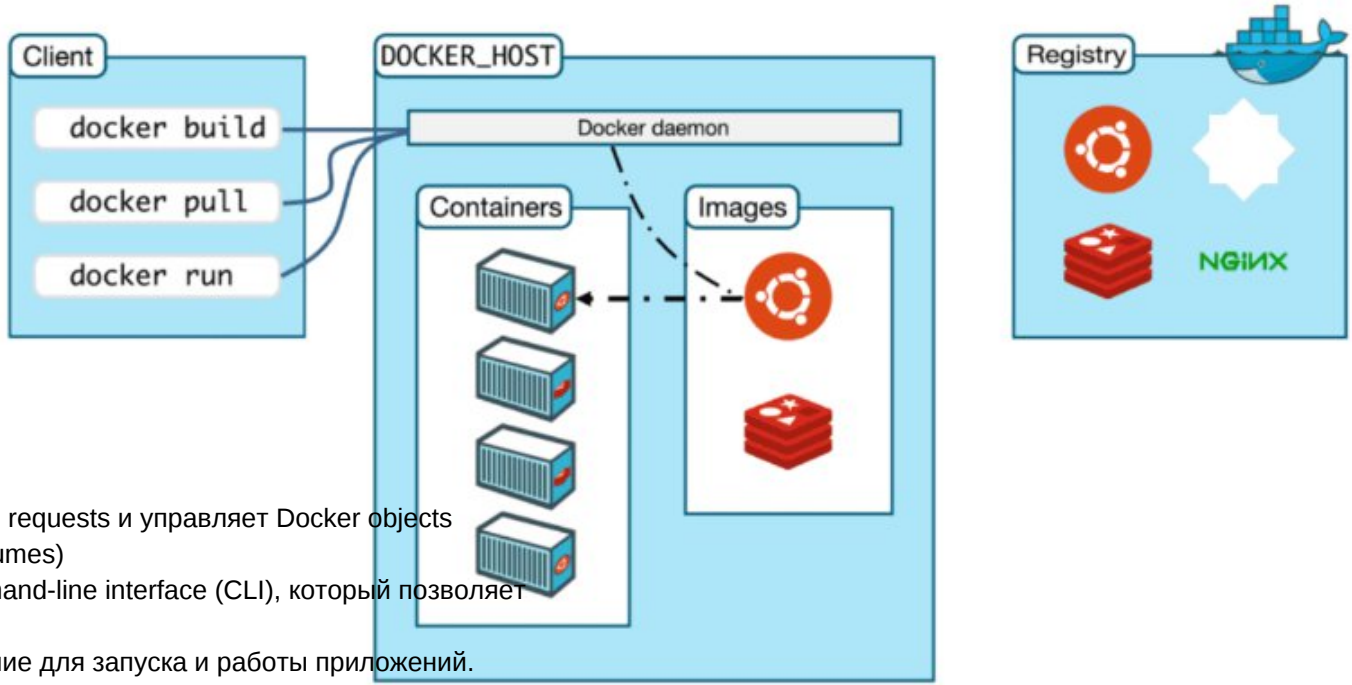


Docker intro

Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called *containers*.[5] The service has both free and premium tiers. The software that hosts the containers is called **Docker Engine**.[6] It was first released in 2013 and is developed by Docker, Inc.[7]

Docker is a tool that is used to automate the deployment of applications in lightweight containers so that applications can work efficiently in different environments in isolation.

Компоненты



Docker Daemon слушает Docker API requests и управляет Docker objects (images, containers, networks and volumes)
Docker Clients предоставляет command-line interface (CLI), который позволяет отправлять команды Docker Daemon
Docker Host предоставляет окружение для запуска и работы приложений. Состоит из Docker daemon, Images, Containers, Networks, and Storage.
Docker Registry хранит образы. Docker Hub это публичное registry где каждый может хранить свой образ и использовать чужие образы.
Docker Images это read-only темплейты которые вы собираете из инструкций, написанных в Dockerfile.

Как запустить контейнер из созданного кем-то образа

```
victor@ubuntu:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:d211f485f2dd1dee407a80973c8f129f00d54604d2c90732e8e320e5038a0348
Status: Downloaded newer image for hello-world:latest

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

Как собрать свой образ на базе другого образа

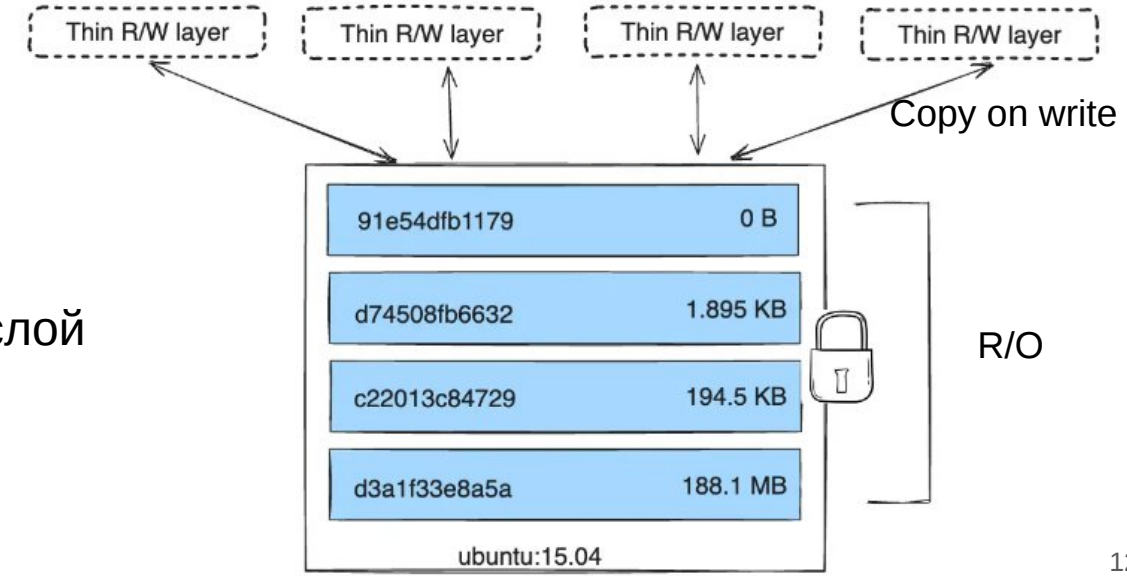
- Сборка образа:

docker build -t <tag_name> **.**

- без кэша:

--no-cache

Dockerfile - каждая строка = слой



Dockerfile – список команд для builder

Builder main commands

command	description
`FROM` *image*|`scratch`	base image for the build
`MAINTAINER` *email*	name of the mainainer (metadata)
`COPY` *path dst*	copy *path* from the context into the container at location *dst*
`ADD` *src dst*	same as `COPY` but untar archives and accepts http urls
`RUN` *args...*	run an arbitrary command inside the container
`USER` *name*	set the default username
`WORKDIR` *path*	set the default working directory
`CMD` *args...*	set the default command
`ENV` *name value*	set an environment variable

dockerfile

```
  GNU nano 7.2                                          nginx/dockerfile
FROM nginx

ADD ./index.html /var/www/html/index.html
ADD ./nginx-conf/nginx.conf /etc/nginx/conf.d/default.conf

RUN /usr/sbin/nginx -t

STOPSIGNAL SIGTERM

CMD ["nginx", "-g", "daemon off;"]
```

Можно (но не нужно) указывать volumes, expose etc

см. также FROM scratch (пустой образ, ядро)

```
victor@ubuntu:~/devops6docker$ sudo docker build -t devops/nginx-server ./nginx
DEPRECATED: The legacy builder is deprecated and will be removed in a future release
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  4.608kB
Step 1/6 : FROM nginx
latest: Pulling from library/nginx
a480a496ba95: Pull complete
f3ace1b8ce45: Pull complete
11d6fdd0e8a7: Pull complete
f1091da6fd5c: Pull complete
40eea07b53d8: Pull complete
6476794e50f4: Pull complete
70850b3ec6b2: Pull complete
Digest: sha256:28402db69fec7c17e179ea87
Status: Downloaded newer image for ngir
 ---> 3b25b682ea82
Step 2/6 : ADD ./index.html /var/www/ht
Step 4/6 : RUN /usr/sbin/nginx -t
 ---> Running in 724c39e1154d
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
Removing intermediate container 724c39e1154d
 ---> 73da11a5f0fa
Step 5/6 : STOPSIGNAL SIGTERM
 ---> Running in 5731a9c7c086
Removing intermediate container 5731a9c7c086
 ---> a59c7d5ab3b4
Step 6/6 : CMD ["nginx", "-g", "daemon off;"]
 ---> Running in 437d8929f59f
Removing intermediate container 437d8929f59f
 ---> 494e5fa8a9a2
Successfully built 494e5fa8a9a2
Successfully tagged devops/nginx-server:latest
victor@ubuntu:~/devops6docker$
```

Как запустить контейнер из созданного нами образа

sudo docker run -d --name <CONTAINER-NAME> <IMAGE-NAME>

-d = detached

```
victor@ubuntu:~/devops6docker$ sudo docker run -d --name nginx-cont devops/nginx-server
39fef2469ebde8ebd7cf6c940839374c472e9a68a5f3ffb3b654afb8b4f55c7c
victor@ubuntu:~/devops6docker$ sudo docker ps -a
CONTAINER ID   IMAGE                 COMMAND                  CREATED          STATUS                      PORTS     NAMES
39fef2469ebd   devops/nginx-server   "/docker-entrypoint.…"   7 seconds ago    Up 6 seconds                80/tcp    nginx-cont
```

Где хранить образы

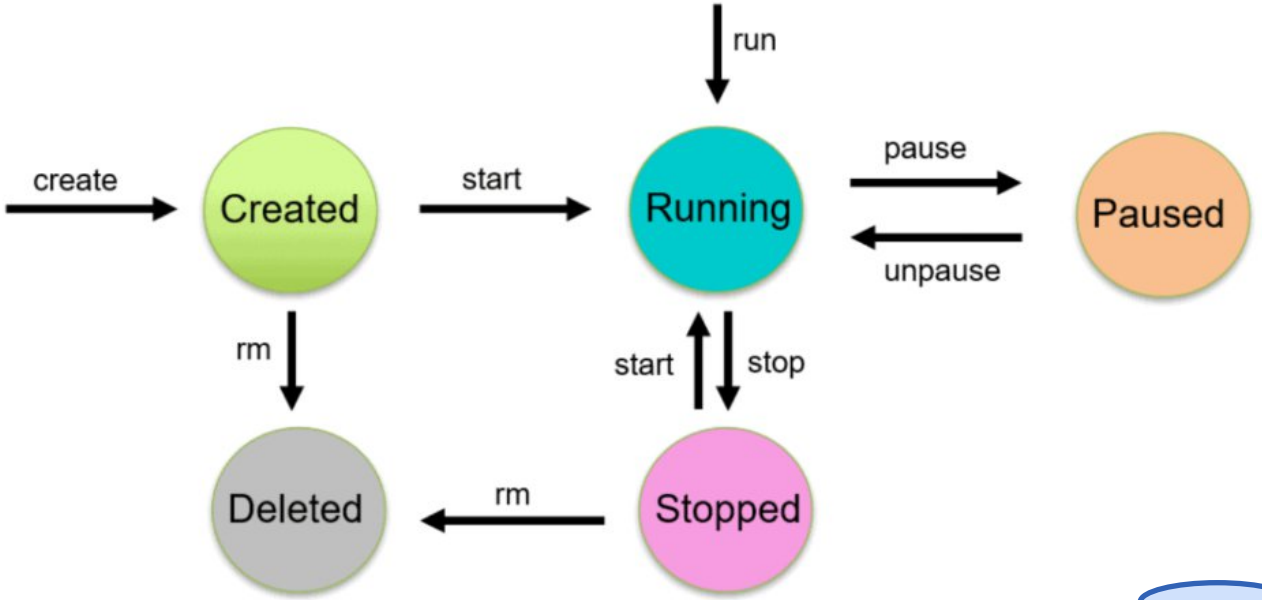
Private registry:

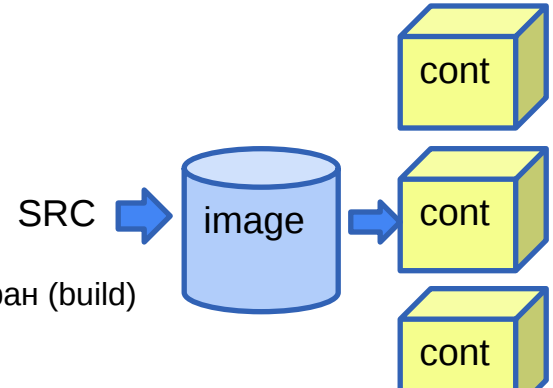
- local
- private registry server

Public registry

- docker hub

Жизненный цикл контейнера



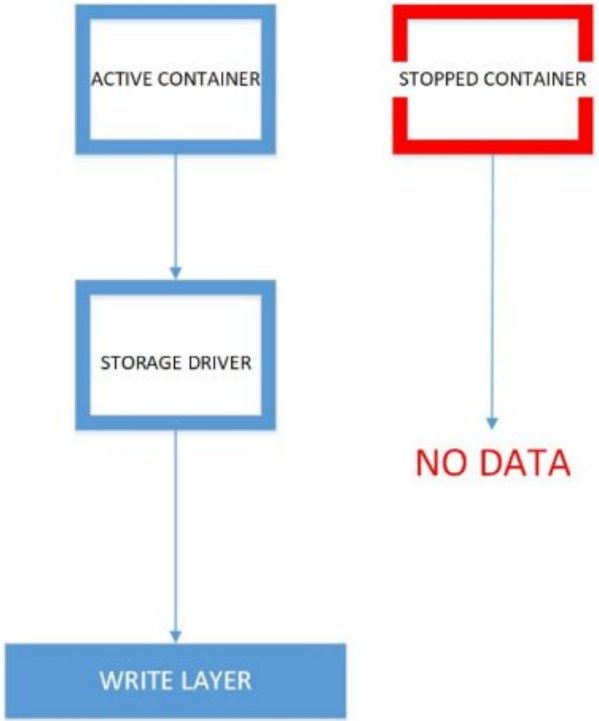


Контейнер создаётся из образа (image), который изначально должен быть собран (build)

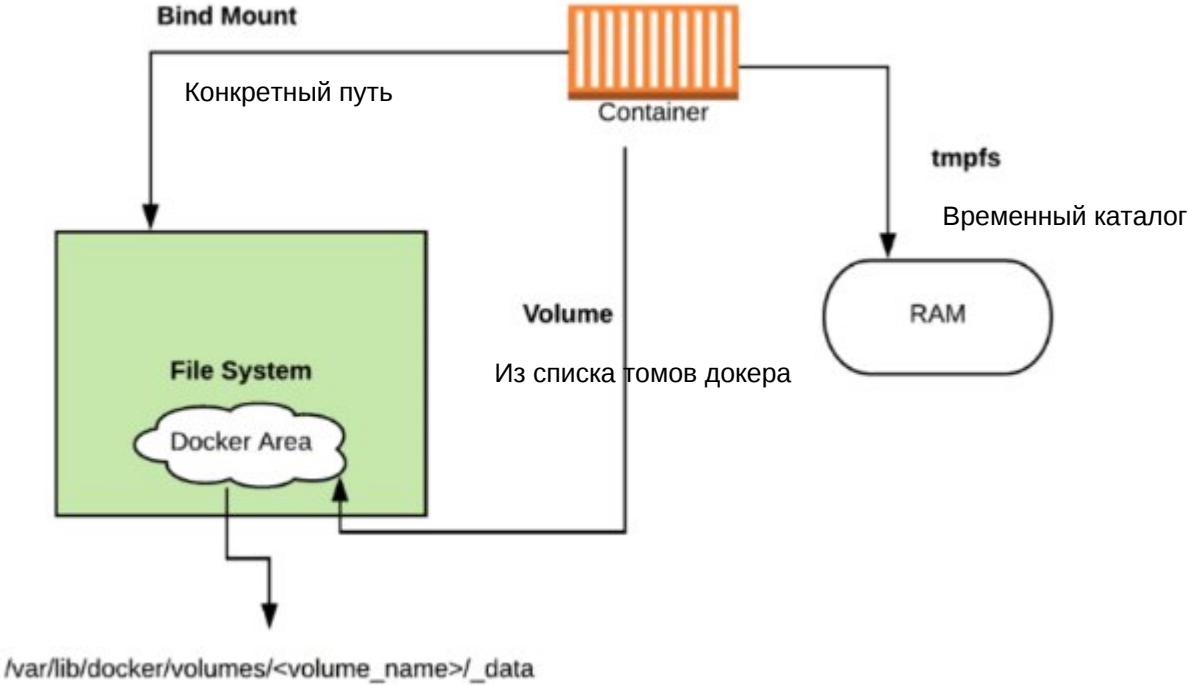
Жизненный цикл контейнера

$ docker create --name <container name> <image name>
$ docker start/stop <container name>
$ docker run -it --name <container name> <image name>
$ docker pause/unpause <container name>
$ docker rm <container name>

Ресурсы контейнера (не образа!)
Storage



Ресурсы контейнера (не образа!)
Storage



Storage (volumes)

Volume mount (создать каталог хостовой системы доступный для монтирования внутрь контейнеров)

- permanent storage

Commands:

- docker volume create - Create a volume
- docker volume inspect - Display detailed information on one or more volumes
- docker volume ls - List volumes
- docker volume prune - Remove all unused local volumes
- docker volume rm - Remove one or more volumes

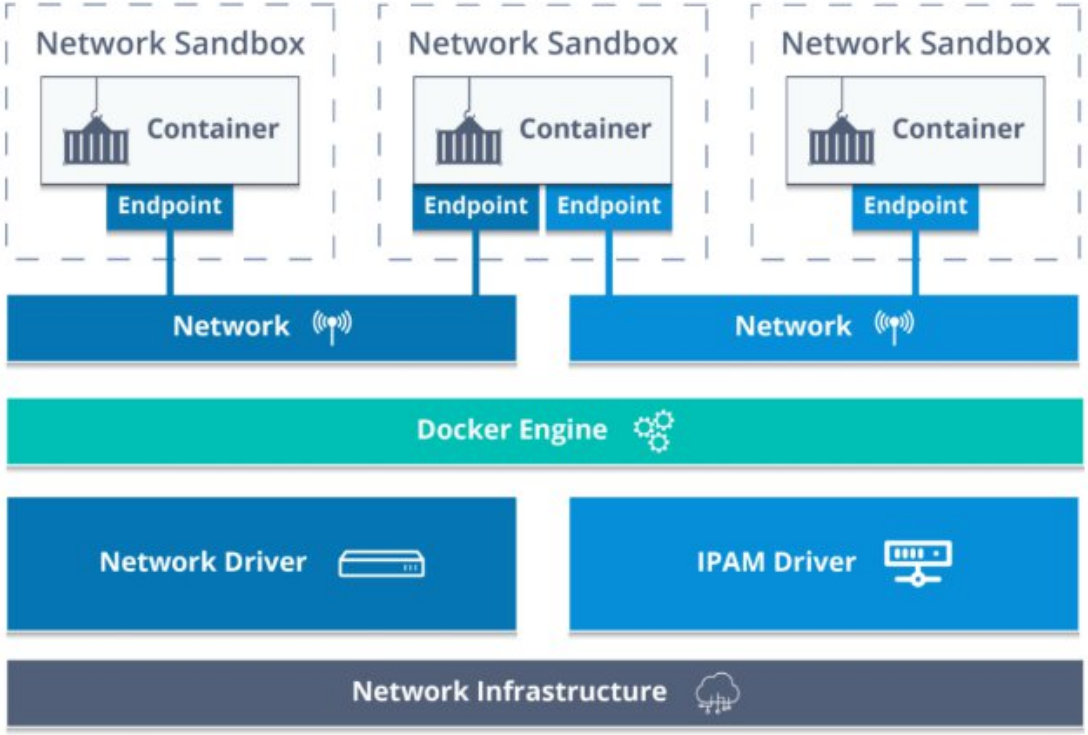
Bind mount (расшарить каталог с хостовой системы внутрь контейнера)

- docker container run -v /host-path:/container-path image-name

Как запустить контейнер с подключением volume

```
sudo docker run -d \
  --name <CONTAINER_NAME> \
  --restart unless-stopped \
  -v <SHARED_VOL_NAME>:<INTERNAL_CONTAINER_PATH> \
  -v <EXTERNAL_HOST_PATH>:<INTERNAL_CONTAINER_PATH> \
 <IMAGE_NAME>
```

Ресурсы контейнера (не образа!)
Network



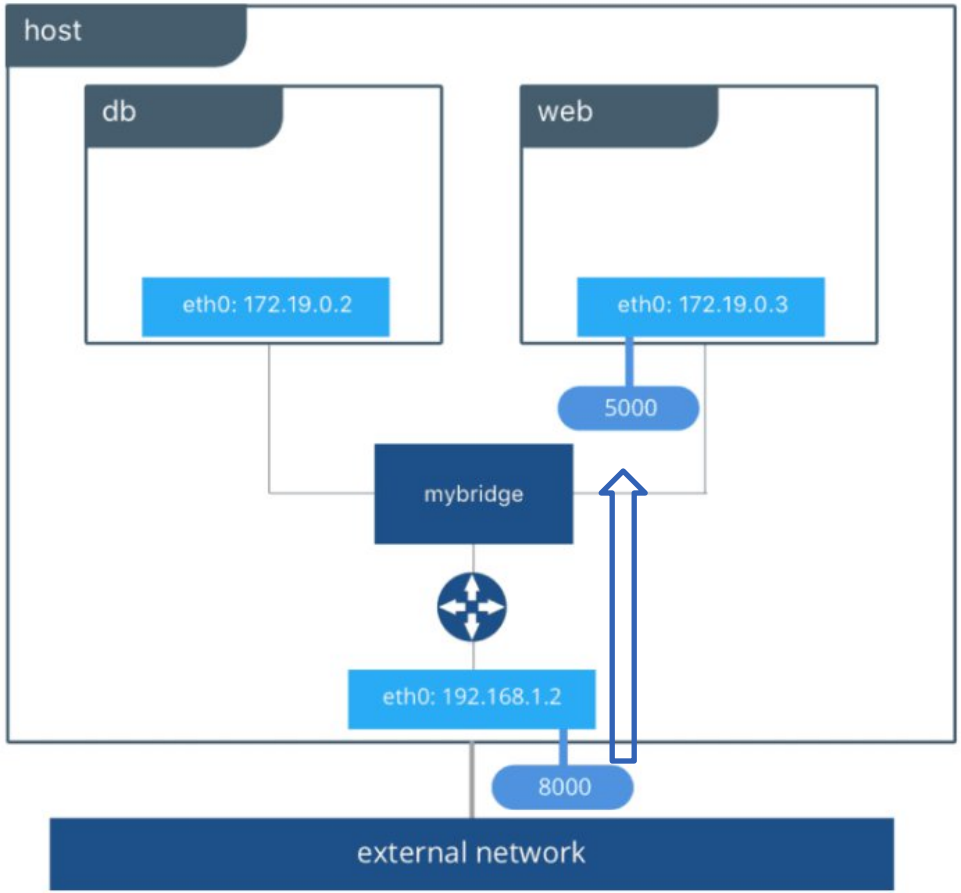
Driver	Description
`bridge`	The default network driver.
`host`	Remove network isolation between the container and the Docker host.
`none`	Completely isolate a container from the host and other containers.
`overlay`	Overlay networks connect multiple Docker daemons together.
`ipvlan`	IPvlan networks provide full control over both IPv4 and IPv6 addressing.
`macvlan`	Assign a MAC address to a container.

Network - bridge

Publish/Expose (Проброс)

Default bridge

User-defined bridges



Как запустить контейнер с публикацией порта

Flag value	Description
`-p 8080:80`	Map port `8080` on the Docker host to TCP port `80` in the container.
`-p 192.168.1.100:8080:80`	Map port `8080` on the Docker host IP `192.168.1.100` to TCP port `80` in the container.
`-p 8080:80/udp`	Map port `8080` on the Docker host to UDP port `80` in the container.
`-p 8080:80/tcp -p 8080:80/udp`	Map TCP port `8080` on the Docker host to TCP port `80` in the container, and map UDP port `8080` on the Docker host to UDP port `80` in the container.

```
docker run -d --name nginx-do -p 54321:80 devops/nginx-server
```

Как посмотреть логи контейнера

sudo docker logs <CONTAINER>

```
victor@ubuntu:~/devops6docker$ sudo docker logs nginx-cont -f -n 10 -t
2024-10-20T20:37:16.828860555Z 2024/10/20 20:37:16 [notice] 1#1: nginx/
2024-10-20T20:37:16.828865442Z 2024/10/20 20:37:16 [notice] 1#1: built
2024-10-20T20:37:16.828868632Z 2024/10/20 20:37:16 [notice] 1#1: OS: Li
2024-10-20T20:37:16.828872192Z 2024/10/20 20:37:16 [notice] 1#1: getrli
2024-10-20T20:37:16.828875286Z 2024/10/20 20:37:16 [notice] 1#1: start
2024-10-20T20:37:16.828879188Z 2024/10/20 20:37:16 [notice] 1#1: start
2024-10-20T20:37:16.829739625Z 2024/10/20 20:37:16 [notice] 1#1: start
2024-10-20T20:37:49.350526368Z 172.17.0.1 - - [20/Oct/2024:20:37:49 +00
2024-10-20T20:37:57.952854477Z 172.17.0.1 - - [20/Oct/2024:20:37:57 +00
2024-10-20T20:37:58.951255626Z 172.17.0.1 - - [20/Oct/2024:20:37:58 +00
```

Как смотреть список контейнеров и образов

sudo docker ps -a

```
victor@ubuntu:~/devops6docker$ sudo docker ps -a
CONTAINER ID   IMAGE                  COMMAND                  CREATED          STATUS
4aa18684951e   devops/nginx-server    "/docker-entrypoint.…"   16 minutes ago   Up 5 minutes
1cb060477804   hello-world            "/hello"                 2 hours ago      Exited (0) 2 h
victor@ubuntu:~/devops6docker$
```

```
victor@ubuntu:~/devops6docker$ sudo docker image ls
REPOSITORY             TAG       IMAGE ID       CREATED          SIZE
devops/nginx-server    latest    494e5fa8a9a2   2 hours ago      192MB
nginx                  latest    3b25b682ea82   2 weeks ago      192MB
hello-world            latest    d2c94e258dcb   17 months ago    13.3kB
```

Как подключиться к контейнеру интерактивно

docker exec vs docker attach

sudo docker exec –it <CONTAINER> <CMD>

```
victor@ubuntu:~/devops6docker$ sudo docker exec -it nginx-cont /bin/bash
root@4aa18684951e:/#
root@4aa18684951e:/#
```

П08. Docker basics - практика

Создать образ и запустить контейнер:

- внутри которого будет работать веб-сервер Nginx,

- отдающий статическую html страницу с приветствием с порта,

- для доступа снаружи к nginx по сети пробросить в контейнер порт 54321

- команду запуска контейнера оформить шелл-скриптом

Установка docker (.io)

sudo apt install docker.io

```
victor@ubuntu:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; pres
     Active: active (running) since Sun 2024-10-20 18:35:58 UTC; 2min 8s a
TriggeredBy: ● docker.socket
```

```
victor@ubuntu:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:d211f485f2dd1dee407a80973c8f129f00d54604d2c90732e8e320e5038a0348
Status: Downloaded newer image for hello-world:latest

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

Структура каталога

```
~/devops6docker$ tree
.
├── deploy.sh
├── nginx
│   ├── dockerfile
│   ├── index.html
│   └── nginx-conf
│       └── nginx.conf
```

Готовим индексную страницу

```
  GNU nano 7.2                                  nginx/index.html
<!DOCTYPE html>
<html>
<head>
<title>DevOps Course 2024</title>
</head>
<body>
DevOps Course 2024!
</body>
</html>
```

Готовим конфиг nginx

Можно скопировать из /etc/nginx, если есть

```
  GNU nano 7.2                    nginx/nginx-conf/nginx.conf

# Default server configuration
#
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        root /var/www/html;

        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                try_files $uri $uri/ =404;
        }

}
```

Готовим dockerfile

```
  GNU nano 7.2                                  nginx/dockerfile
FROM nginx

ADD ./index.html /var/www/html/index.html
ADD ./nginx-conf/nginx.conf /etc/nginx/conf.d/default.conf

RUN /usr/sbin/nginx -t

STOPSIGNAL SIGTERM

CMD ["nginx", "-g", "daemon off;"]
```

Готовим скрипт запуска

```
  GNU nano 7.2                                deploy.sh
#!/bin/bash -x

# Удаляем старый контейнер, если есть
docker stop nginx-cont
docker rm   nginx-cont

# Собираем образ
docker build -t devops/nginx-server ./nginx

# Создаем и запускаем контейнер
docker run -d --name nginx-cont -p 54321:80 \
  --restart unless-stopped \
  devops/nginx-server

# Проверяем
docker ps -a
sleep 5
curl 127.0.0.1:54321
docker logs -n 10 nginx-cont
```

Проверка

```
victor@ubuntu:~/devops6docker$ sudo ./deploy.sh
+ docker stop nginx-cont
nginx-cont
+ docker rm nginx-cont
nginx-cont
+ docker build -t devops/nginx-server ./nginx
```

...

```
<title>DevOps Course 2024</title>
</head>
<body>
DevOps Course 2024!
</body>
</html>
+ docker logs -n 10 nginx-cont
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/10/20 20:59:36 [notice] 1#1: using the "epoll" event method
2024/10/20 20:59:36 [notice] 1#1: nginx/1.27.2
2024/10/20 20:59:36 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/10/20 20:59:36 [notice] 1#1: OS: Linux 6.8.0-45-generic
2024/10/20 20:59:36 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/10/20 20:59:36 [notice] 1#1: start worker processes
2024/10/20 20:59:36 [notice] 1#1: start worker process 28
2024/10/20 20:59:36 [notice] 1#1: start worker process 29
172.17.0.1 - - [20/Oct/2024:20:59:41 +0000] "GET / HTTP/1.1" 200 115 "-" "cu
```