



O T U S

ОНЛАЙН-ОБРАЗОВАНИЕ

# Меня хорошо видно и слышно?

Поставьте , если все хорошо  
Напишите в чат, если есть проблемы



# Docker: докерфайлы, команды, образы



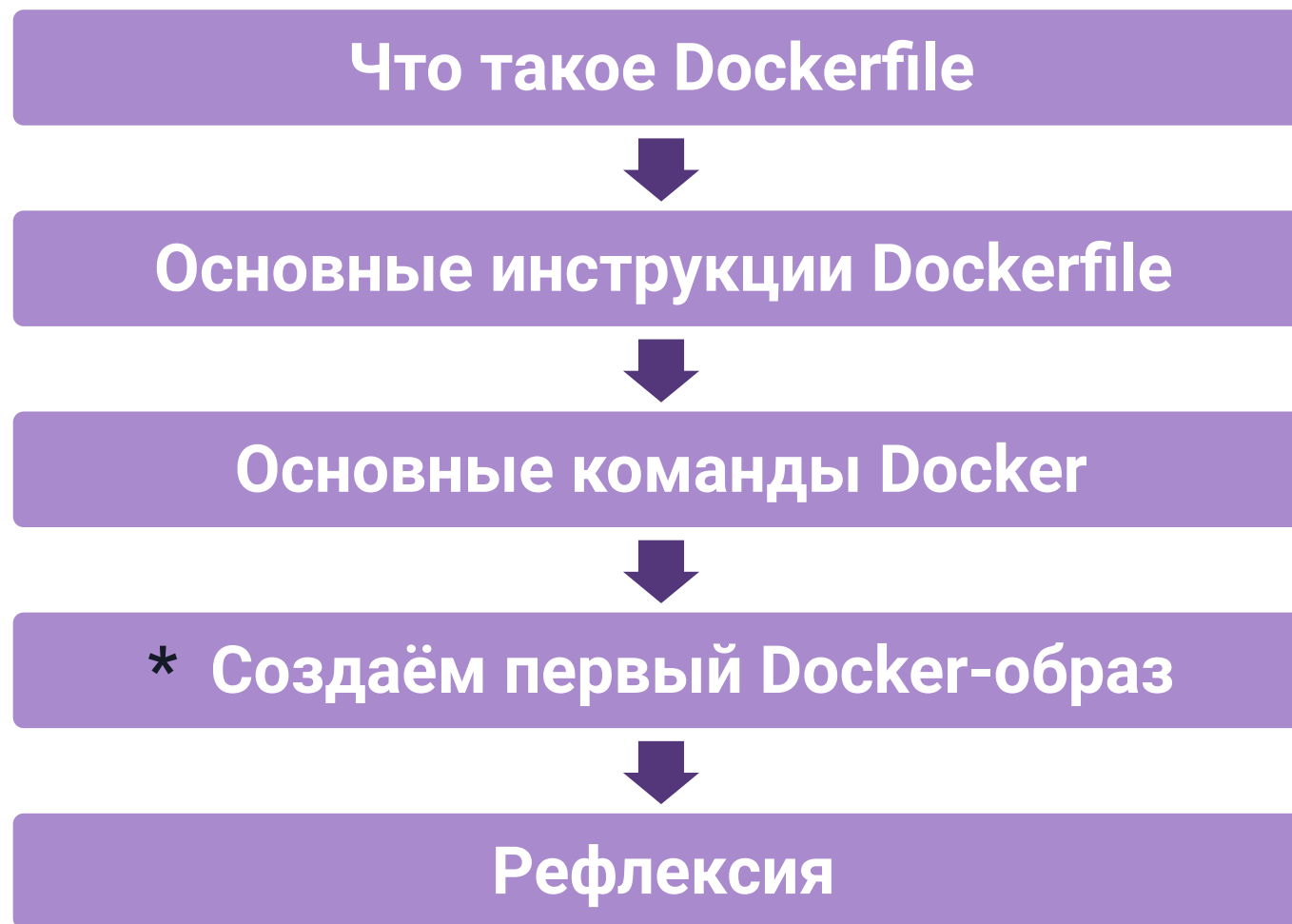
Константин Брюханов

 [vk.com/deusops](https://vk.com/deusops)

 [deus\\_ops](https://t.me/deus_ops)

 [deusops](https://www.instagram.com/deusops)

# Маршрут вебинара



# Зачем вам это уметь

1

Упаковка любого приложения в контейнер

2

Возможность быстрого тестирования на ранних стадиях

3

Возможность локальной разработки

4

Повышение скорости применения

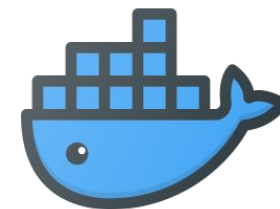


# Что такое Dockerfile

The background of the slide is a digital illustration of a futuristic city at night. The sky is a deep purple and blue, filled with numerous small white stars. Several bright red and blue vertical lines, resembling data streams or light beams, extend from the top of the frame down towards the city. The city itself is composed of various skyscrapers and buildings, some of which are illuminated with yellow and orange lights. In the foreground, there are glowing blue and yellow lines that create a sense of perspective, leading the eye towards the horizon. The overall aesthetic is high-tech and cybernetic.

# Что такое Dockerfile

## Dockerfile



- Текстовый файл с последовательным описанием инструкций для сборки образа
- Каждая инструкция создает промежуточный слой образа
- Инструкции кешируются в образах
- Сам сборку делает демон Docker, а не Docker CLI

# Основные инструкции Dockerfile

A futuristic cityscape at night with glowing skyscrapers and vertical red and blue light beams. The foreground features a dark floor with a glowing blue grid pattern and a single yellow line.



# Основные инструкции Dockerfile

## Инструкция: FROM

Инструкция **FROM** указывает базовый образ, на основе которого мы строим свою сборку:

```
FROM <image>[:<tag>]  
# <image> - имя базового образа  
# <tag> - опциональный атрибут указывающий на версию образа
```

Примеры:

```
FROM ubuntu:16.04  
FROM quay.io/vektorlab/ctop
```

# Основные инструкции Dockerfile

## Инструкция: LABEL

Инструкция **LABEL** задает метаданные для нашего образа:

```
LABEL <key>=<value> [<key>=<value> ...]  
# <key> - ключ  
# <value> - значение
```

Примеры:

```
LABEL maintainer="user@example.org" version="0.2.1-8d2095e3ce"
```

# Основные инструкции Dockerfile

## Инструкция: COPY

Инструкция **COPY** копирует файлы из контекста в образ:

```
COPY <src> [<src> ...] <dst>
```

```
# или
```

```
COPY [ "<src>", ... "<dest>" ]
```

```
# <src> - файл или директория внутри build контекста
```

```
# <dst> - файл или директория внутри контейнера
```

Примеры:

```
COPY start* /startup/
```

```
COPY httpd.conf magicfile /etc/httpd/conf/
```



# Основные инструкции Dockerfile

## Инструкция: ENV

Инструкция **ENV** задает переменные окружения при сборке:

```
ENV <key> <value>
```

*# <key> - имя переменной окружения*

*# <value> - присваиваемое значение*

Примеры:

```
ENV LOG_LEVEL debug
```

```
ENV DB_HOST 127.0.0.1:3389
```

# Основные инструкции Dockerfile

## Инструкция: WORKDIR

Инструкция **WORKDIR** задает рабочую директорию при сборке:

```
WORKDIR <path>
```

```
# <path> - путь внутри контейнера
```

Примеры:

```
WORKDIR /app
```

# Основные инструкции Dockerfile

## Инструкция: VOLUME

Инструкция **VOLUME** позволяет указать точки для монтирования томов внутри образа:

```
VOLUME <dst> [<dst> ...]
```

```
# <dst> - директория монтирования для volume'a
```

Примеры:

```
VOLUME /app /db /data
```

```
# или
```

```
VOLUME ["/var/www", "/var/log/apache2", "/etc/apache2"]
```



# Основные инструкции Dockerfile

## Инструкция: EXPOSE

Инструкция **EXPOSE** позволяет указать порты, которые слушает сервис в запущенном контейнере:

```
EXPOSE <port>[/<proto>] [<port>[/<proto>] ...]
```

*# <port> - порт сервиса внутри контейнера*

*# <proto> - tcp или udp*

Примеры:

```
EXPOSE 5000
```

```
EXPOSE 8080/tcp 3389/udp
```

# Основные инструкции Dockerfile

## Инструкция: RUN

Инструкция **RUN** задает команды, которые выполняются при **сборке** контейнера:

```
RUN <command>
```

*# <command> - команда которая будет выполнена при создании образа*

Примеры:

```
RUN apt-get update && apt-get install nginx  
RUN ["bash", "-c", "rm", "-rf", "/tmp/abc"]  
RUN ["myscript.py", "argument1", "argument2"]
```

# Основные инструкции Dockerfile

## Инструкция: CMD

Инструкция **CMD** задает команду, которая выполняется при **старте** контейнера:

```
CMD <command>
```

```
# <command> - команда которая будет выполнена при старте контейнера
```

Примеры:

```
CMD /start.sh
```

```
CMD [ "echo", "Dockerfile CMD demo" ]
```



# Основные инструкции Dockerfile

## Инструкция: ENTRYPOINT

Инструкция **ENTRYPOINT** задает команду, которая выполняется при **старте** контейнера:

```
ENTRYPOINT <command>
```

*# <command> – команда которая будет выполнена при старте контейнера*

Примеры:

```
ENTRYPOINT exec top -b  
ENTRYPOINT [ "/usr/sbin/apache2ctl", "-D", "FOREGROUND" ]  
ENTRYPOINT [ "/bin/sh", "/docker-entrypoint.sh" ]
```

# Основные инструкции Dockerfile

## Дополнительные инструкции

<b>ONBUILD</b> <cmd>	<i># Задаёт команду, которая запускается # при сборке образа на базе текущего</i>
<b>STOPSIGNAL</b> <sig>	<i># Указывает сигнал, который посылаётся # процессу при остановке контейнера</i>
<b>USER</b> <username>	<i># Имя (ID) пользователя, от которого # выполняются директивы RUN, CMD, ENTRYPOINT</i>
<b>ARG</b> <string>	<i># Почти как ENV, но задаёт параметры # только для docker build</i>
<b>HEALTHCHECK</b> <cmd>	<i># Указывает команду, которой можно # проверить состояние сервиса</i>

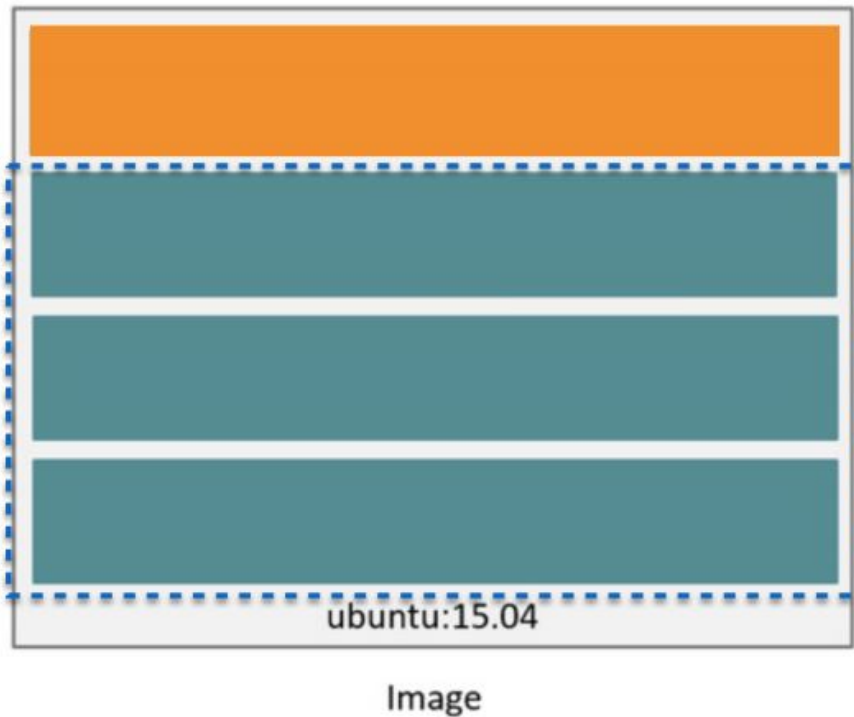
# Работа с Dockerfile

A futuristic cityscape at night with a grid floor and neon lines. The background features a city skyline with various skyscrapers, some of which are illuminated with blue and yellow lights. The foreground is dominated by a grid of glowing blue lines that recede into the distance, creating a sense of depth. A single yellow line runs parallel to the blue lines on the left side. The overall color palette is dark blue and purple, with bright neon accents.



# Что такое Dockerfile

## Структура слоёв



Итоговый слой, доступный для чтения и записи. Монтируется при запуске контейнера



Слои образа доступны только для чтения

# Что такое Dockerfile

## Структура слоёв

91e54dfb1179	0 B
d74508fb6632	1.895 KB
c22013c84729	194.5 KB
d3a1f33e8a5a	188.1 MB
ubuntu:15.04	

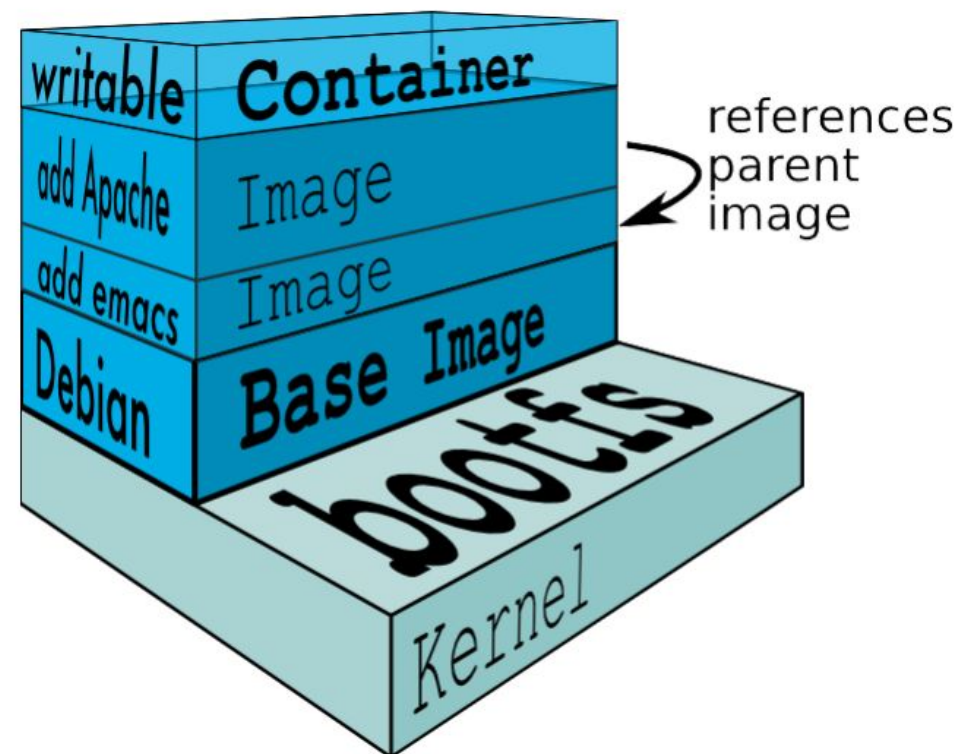
Image

CMD

RUN

RUN

ADD/COPY



# Что такое Dockerfile

## Кэширование

- Кеширование очень важно для реализации быстрых сборок
- **ADD, COPY** - файлы по умолчанию кэшируются (в случае изменений файлов, кэш сбрасывается)
- Для остальных инструкций, включая **RUN**, проверяется только изменение параметров самой инструкции
- **RUN apt-get -y update** не проверяет обновления постоянно, только в первый раз
- Пропустить и перестроить кэш можно командой

```
docker build --no-cache
```

# Что такое Dockerfile

## Кэширование

- Сначала **RUN**, потом **COPY**:

<code>ENV myvar false</code>	<code># Cache hit! ✓</code>
<code>RUN apt-get install -y nginx php-fpm &amp;&amp; imagemagick</code>	<code># Cache hit! ✓</code>
<code>COPY app /src/app</code>	<code># Cache miss! ✗</code>
<code>CMD ["/bin/cool-soft"]</code>	<code># Cache miss! ✗</code>

- Сначала **COPY**, потом **RUN**:

<code>ENV myvar false</code>	<code># Cache hit! ✓</code>
<code>COPY app /src/app</code>	<code># Cache miss! ✗</code>
<code>RUN apt-get install -y nginx php-fpm &amp;&amp; imagemagick</code>	<code># Cache miss! ✗</code>
<code>CMD ["/bin/cool-soft"]</code>	<code># Cache miss! ✗</code>



# Что такое Dockerfile

## Кэширование

- Кеширование очень важно для реализации быстрых сборок
- **ADD, COPY** - файлы по умолчанию кэшируются (в случае изменений файлов, кэш сбрасывается)
- Для остальных инструкций, включая **RUN**, проверяется только изменение параметров самой инструкции
- **RUN apt-get -y update** не проверяет обновления постоянно, только в первый раз
- Пропустить и перестроить кэш можно командой

```
docker build --no-cache
```

# Что такое Dockerfile

## Уменьшение размера образа

- Все просто: удаляйте за собой архивы и временные файлы, которые остались во время билда.

```
COPY <filename>.zip <copy_directory>      # New image! 🐳  
RUN unzip <filename>.zip                  # New image! 🐳
```

- В результирующем образе останется ZIP-архив

```
COPY <filename>.zip <copy_directory>      # New image! 🐳  
RUN unzip <filename>.zip                  # New image! 🐳  
RUN rm <filename>.zip                    # New image! 🐳
```

- Остаются наследуемые образы. И конечно же, с ZIP-архивом

# Что такое Dockerfile

## Уменьшение размера образа

- Имеем скомпилированное приложение, которое не имеет зависимостей:

```
FROM ubuntu:14.04

COPY ./hello-world .
EXPOSE 8080
CMD [ "./hello-world" ]
```

- Проверим размер образа созданного из такого Dockerfile:

	TAG	CREATED	SIZE
hello-app	1.0	15 seconds ago	194MB
ubuntu	14.04	8 days ago	188MB

- А нужен ли нам образ ОС? Он же в 30 раз больше самого приложения!

# Что такое Dockerfile

## Уменьшение размера образа

**scratch** - один из зарезервированных образов Docker, в котором ничего нет (пустой Dockerfile) В случае со **scratch**-образом, следующая инструкция создаст первый слой с файловой системой

```
FROM scratch

COPY ./hello-world .
EXPOSE 8080
CMD [ "./hello-world" ]
```

	TAG	CREATED	SIZE
hello-app	2.0	51 seconds ago	5.85MB
hello-app	1.0	15 minutes ago	194MB



# Что такое Dockerfile

## Общие рекомендации

- Избегайте установки лишних пакетов и упаковки лишних данных в образы при сборке (build context bloating)
- Используйте связанные команды для **RUN**-инструкций
- Следим за последовательностью описания **Dockerfile**, избегаем cache miss
- Уменьшайте количество слоев
- Один контейнер - одна задача
- Чистите за собой
- Используйте multi-stage сборки (для компилируемых языков)

# Что такое Dockerfile

## Секретный слайд!

- У **docker build** есть параметр **—squash**, который в финале собирает все слои в один (как будто инструкции **Dockerfile** были выполнены в одном слое), так что можно сильно не заморачиваться.



# Основные команды Docker

A futuristic cityscape at night with glowing skyscrapers and a grid of neon lines on the ground. The scene is dominated by blue and purple hues, with red and yellow light streaks adding a sense of motion and technology.

# Основные команды Docker

## Базовые команды docker-cli

- ↳ **docker build:** сборка из dockerfile
- ↳ **docker run:** запуск контейнера из образа
- ↳ **docker ps:** просмотр имеющихся контейнеров
- ↳ **docker images:** просмотр локально имеющихся образов
- ↳ **docker pull/push:** скачать или закатать docker-образ

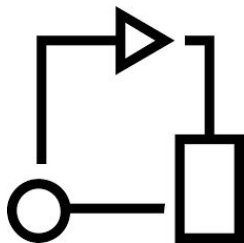




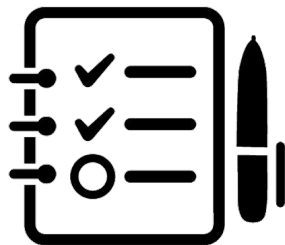


LIVE

# Рефлексия



С какими основными мыслями и инсайтами уходите с вебинара?



Достигли ли вы цели вебинара?

# Проверка достижения целей

1

Научились создавать докерфайлы

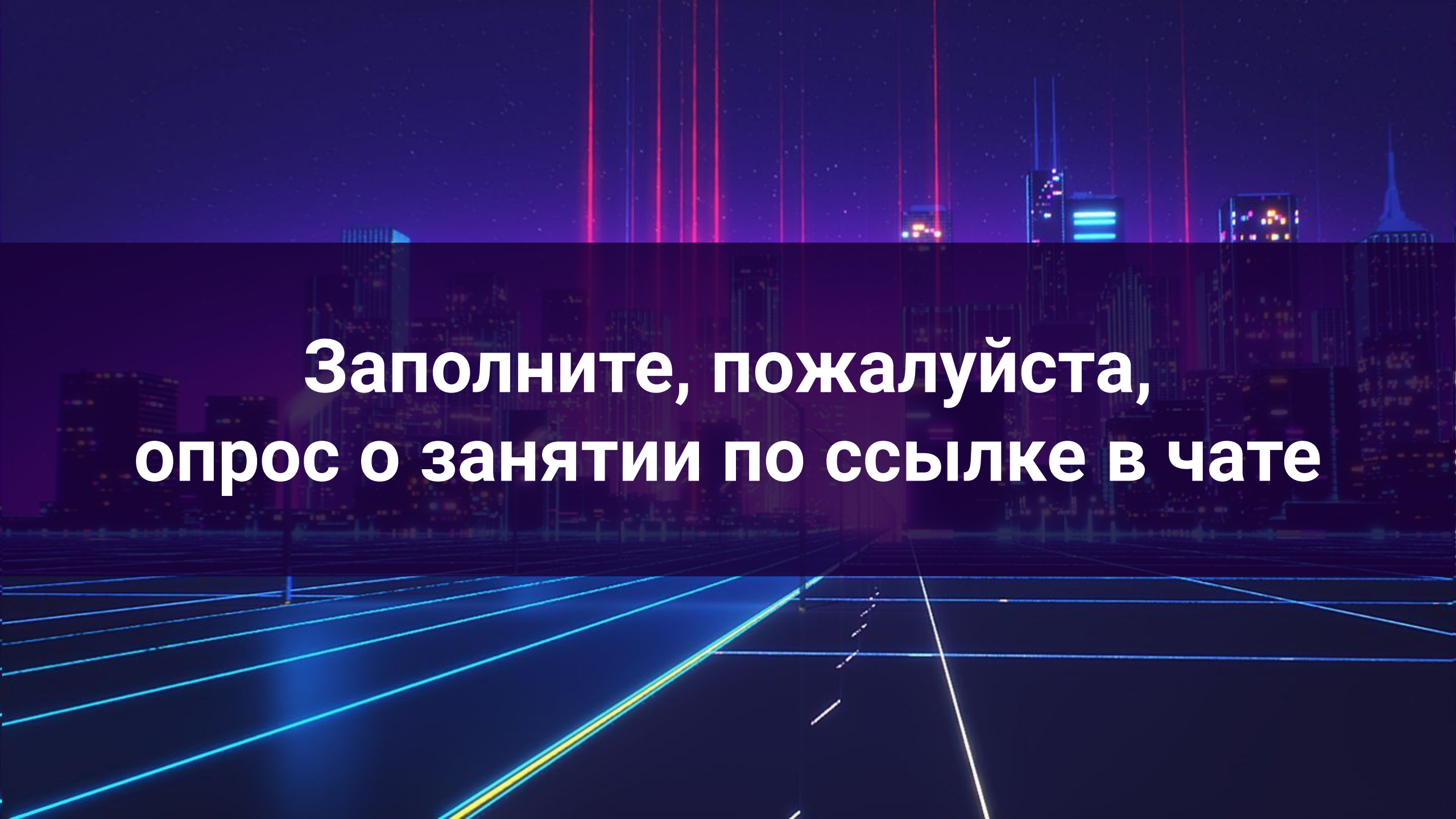
2

Собрали свой первый докер-образ

3

Встроили сборку докер-образа  
в свой пайплайн



A futuristic cityscape at night with a grid of glowing blue and yellow lines on the ground and vertical red and blue lines in the sky. The background shows a city skyline with lit-up buildings.

**Заполните, пожалуйста,  
опрос о занятии по ссылке в чате**



# Спасибо за внимание!



Константин Брюханов

 [vk.com/deusops](https://vk.com/deusops)

 [deus\\_ops](https://t.me/deus_ops)

 [deusops](https://www.instagram.com/deusops)