**1 ---------------------------------------------------------------------------**

# Создаём и запускаем nginx контейнер

**docker container run nginx**

**2 ---------------------------------------------------------------------------**

# Создаём и запускаем nginx контейнер

# 1. в фоновом режиме

**docker container run -d nginx**

# OR

# **docker container run --detached nginx**

**3 ----------------------------------------------------------------------------**

# Создаём и запускаем nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

**docker container run -d -p 80:80 nginx**

# OR

# docker container run --detached --publish 80:80 nginx

**4 ----------------------------------------------------------------------------**

# Создаём и запускаем nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

**docker container run -d -p 80:80 --name proxy nginx**

**5 ---------------------------------------------------------------------------**

# Создаём и запускаем nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

**docker container run -d -p 80:80 --name proxy nginx**

# Вернёт ошибку так как имя контейнера должно быть уникальным

**docker container run -d -p 8080:80 --name proxy nginx**

**6 ---------------------------------------------------------------------------**

# Создаём и запускаем nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

**docker container run -d -p 80:80 --name proxy nginx**

# Порт host машины и имя контейнера должны быть уникальны

**docker container run -d -p 8080:80 --name webserver nginx**

**7---------------------------------------------------------------------------**

# Создаём и запускаем nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

# 4. Определяем healthcheck

**docker container run -d -p 80:80 --name proxy --health-cmd 'curl http://localhost:80/' --health-retries 3 --health-interval '1s' nginx**

# Для того что бы исправить статус на healthy необходимо установить в контейнере curl.

# **apt-get update && apt-get install curl –y**

**8 ---------------------------------------------------------------------------**

# Создаём и запускаем nginx контейнер

# 1. В интерактивном режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

**docker container run -p 80:80 --name proxy -i nginx**

#OR

# **docker container run -d -p 80:80 --name proxy --interactive nginx**

**9 ---------------------------------------------------------------------------**

# Создаём и запускаем nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

# 4. Устанавливаем хард лимит занимаемой памяти

**docker container run -d -p 80:80 --name proxy -m 10485760 nginx**

#OR

# **docker container run -d -p 80:80 --name proxy --memroy 10485760 nginx**

#NB limit in bytes. 10485760 = 10Mb (10 \* 1024 \* 1024)

**10 ---------------------------------------------------------------------------**

# Создаём и запускаем nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

# 4. Устанавливаем хард и софт лимит занимаемой памяти

**docker container run -d -p 80:80 --name proxy -m 10485760 --memory-reservation 5242880 nginx**

# 10485760 = 10Mb

# 5242880 = 5Mb

# **--memory-reservation = baseline**

**11 ---------------------------------------------------------------------------**

# Создаём и запускаем nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

# 4. Устанавливаем хард лимит и лимит swap который может использовать контейнер

**docker container run -d -p 80:80 --name proxy -m 5242880 --memory-swap 10485760 nginx**

**12 ---------------------------------------------------------------------------**

# Создаём и запускаем nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

# 4. Устанавливаем политику рестарта контейнера

**docker container run -d -p 80:80 --name proxy --restart always nginx**

# no — default

# on-failure — перезагрузить в случае ошибки

# always — всегда перезагружать

# unless-stopped — пока контейнер не будет остановлен

**13 ---------------------------------------------------------------------------**

# Создаём и запускаем nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

# 4. Удаляем контейнер после его остановки

**docker container run --rm -d -p 80:80 --name proxy nginx**

**14 ---------------------------------------------------------------------------**

# Создаём и запускаем nginx контейнер

# 1. В интерактивном режиме

# 2. Устанавливаем имя контейнера

# 3. Подключаем псевдо TTY

**docker container run -i -t --name proxy ubuntu bash**

#OR

# **docker container run -it --name proxy ubuntu bash**

**15 ---------------------------------------------------------------------------**

# Создаём nginx контейнер

# 1. Мапим порт к host машине

# 2. Устанавливаем имя контейнера

**docker container create -p 80:80 --name proxy nginx** 09b5b111388f12b06ddead99ae648718fe0086acd331c28e7ccf14bdabab7ab9

# Показывает только запущенные контейнеры

**docker container ps**

**16 ---------------------------------------------------------------------------**

# Создаём nginx контейнер

# 1. Мапим порт к host машине

# 2. Устанавливаем имя контейнера

**docker container create -p 80:80 --name proxy nginx**

# Показывает все контейнеры

**docker container ps -a**

**17 ---------------------------------------------------------------------------**

# Создаём nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

**docker container run -d -p 80:80 --name proxy nginx**

# Присоединяет stdout, stderror к контейнеру

# **docker container attach proxy**

**18 ---------------------------------------------------------------------------**

# Создаём nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

**CONTAINER\_ID=$(docker container run -d -p 80:80 --name proxy nginx)**

# Создать новый образ на основании контейнера

**docker container commit --author "Andrey Prisniak aprisniak@lectrum.io" --message "Add curl" "$CONTAINER\_ID" sotream/nginx-curl:0.0.1**

**19 ---------------------------------------------------------------------------**

# Создаём nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

**docker container run -d -p 80:80 --name proxy nginx**

# Получить подробную информацию о контейнере

**docker container inspect proxy**

**20 ---------------------------------------------------------------------------**

# Создаём nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

**docker container run -d -p 80:80 --name proxy nginx**

# Получить отформатированную информацию о контейнере

**docker container inspect proxy --format "IP: {{ .NetworkSettings.IPAddress }} | Gateway: {{.NetworkSettings.Networks.bridge.Gateway}}"**

**21 ---------------------------------------------------------------------------**

# Создаём nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

**docker container run -d -p 80:80 --name proxy1 nginx**

**docker container run -d -p 8080:80 --name proxy2 nginx**

# Получить подробную информацию о контейнере

**docker container rename proxy1 proxy**

**docker container rename proxy2 webhost**

**22 ---------------------------------------------------------------------------**

# Создаём nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

**docker container run -d -p 80:80 --name proxy nginx**

# Удалить контейнер

# Вызовет ошибку если контейнер запущен

**docker container rm proxy**

**23 ---------------------------------------------------------------------------**

# Создаём nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

**docker container run -d -p 80:80 --name proxy nginx**

# получить real time информацию о ресурсах контейнера

**docker container stats**

**24 ---------------------------------------------------------------------------**

# Создаём nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

**for VARIABLE in 1 2 3**

**do**

**docker container create --name proxy-$VARIABLE nginx**

**done**

# удалить все остановленные контейнеры

**docker container prune**

**25 ---------------------------------------------------------------------------**

# Создаём nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

**docker container run -d -p 80:80 --name proxy nginx**

# получить информацию о портах контейнера

**docker container port proxy**

**26 ---------------------------------------------------------------------------**

# Создаём nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

**docker container run -d -p 27017:27017 --name db mongo**

# получить логи контейнера

**docker container logs db**

**27 ---------------------------------------------------------------------------**

# Создаём nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

**docker container run -d -p 80:80 --name proxy nginx**

# выполнить команду внутри контейнера

**docker container exec -it proxy bash**

**28 ---------------------------------------------------------------------------**

# Создаём nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

**docker container run -d -p 80:80 --name proxy nginx**

# скопировать файлы и папки внутрь контейнера

**docker container cp ./data/\*.html proxy:/usr/share/nginx/html**

**docker container cp ./data/css proxy:/usr/share/nginx/html/css**

**29 ---------------------------------------------------------------------------**

# Создаём nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

# 3. Устанавливаем имя контейнера

**docker container run -d -p 80:80 --name proxy nginx**

# скопировать файлы и папки внутрь контейнера

**docker container cp ./data/\*.html proxy:/usr/share/nginx/html**

**docker container cp ./data/css proxy:/usr/share/nginx/html/css**

# получить изменения в контейнере

**docker container diff proxy**

# A файл или директория были добавлены

# D файл или директория были удалены

# C файл или директория были изменены

**30 ---------------------------------------------------------------------------**

# Создаём nginx контейнер

# 1. В фоновом режиме

# 2. Мапим порт к host машине

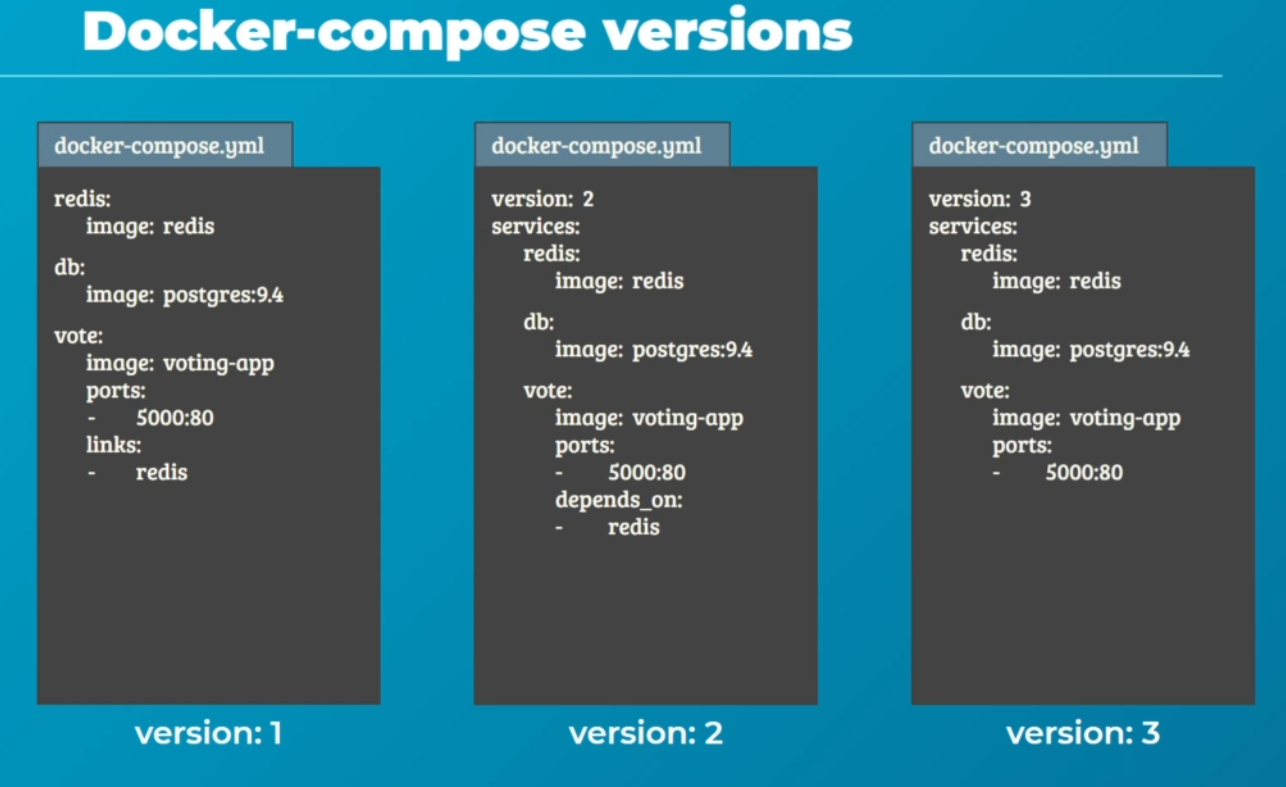
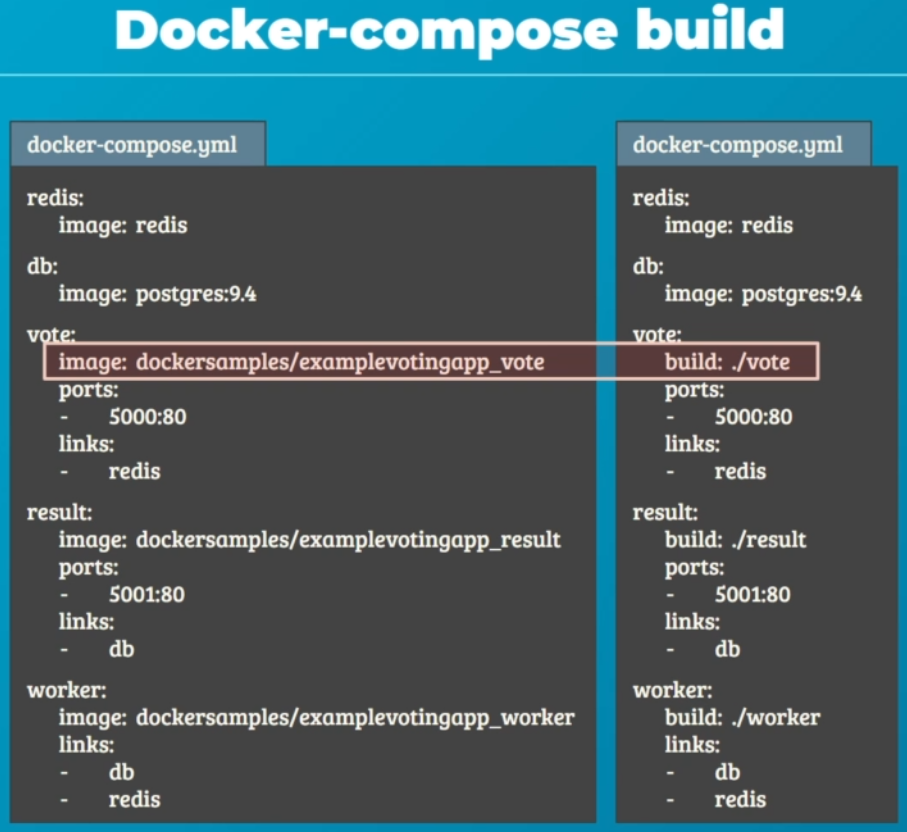
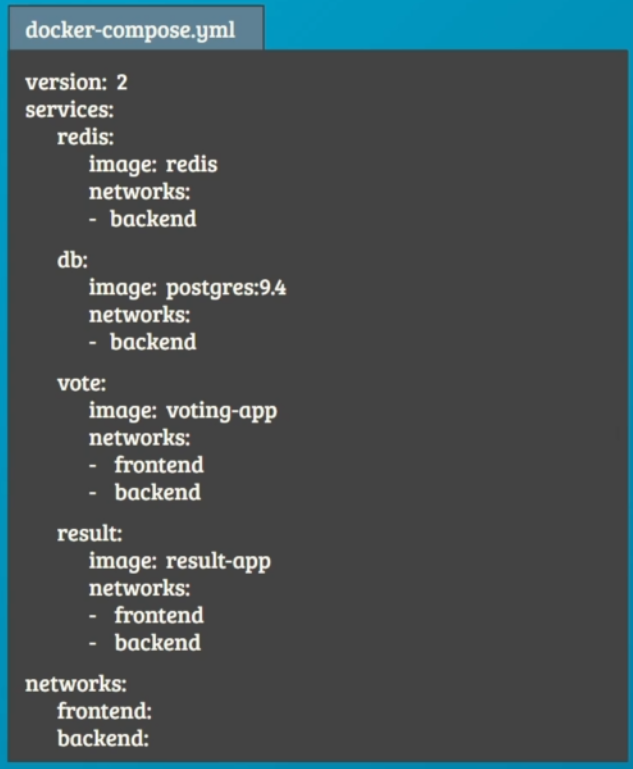
# 3. Устанавливаем имя контейнера

**docker container run -d -p 80:80 --name proxy nginx**

# рестарт контейнера через 5с

**sleep 5**

**docker container restart proxy**

version: "3"

services:

webserver:

image: nginx:alpine

container\_name: webserver

ports:

- 8080:80

volumes:

- ./website:/usr/share/nginx/html

depends\_on:

- db

networks:

- db\_net

- backend\_net

db:

image: postgres

container\_name: database

volumes:

- ./website/db:/data/db

networks:

- db\_net

networks:

backend\_net:

driver: bridge

db\_net:

driver: bridge

* FROM — задаёт родительский (главный) образ;
* LABEL — добавляет метаданные для образа. Хорошее место для размещения информации об авторе;
* ENV — создаёт переменную окружения;
* RUN — запускает команды, создаёт слой образа. Используется для установки пакетов и библиотек внутри контейнера;
* 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.