# **--------------- DOCKER --------------**

# **Info About**

**docker --version** узнаем версию docker

**docker-compose --version** узнаем версию docker-compose

**docker --help** выводит всю документацию про docker

**docker my\_command --help** выводить всю информацию об указаной команде

**docker images** или **docker image ls** показываеть все сохраненыне образ(docker images) на етом пк

**docker ps** или **docker ps -a /** **docker container ls** или **docker container ls -а** показывает все запущение контейнеры в текущий момент

**docker logs my\_container\_id** возвращает данные про запущенный контейнер(на каком ip , на каком порту)

**docker ps -q** выводи все ID всех запущених контейнеров

**docker info** выдает всю информацию про docker и images и containers в нем

**docker logs name\_container** или **docker logs id\_container** выводит логи(информацию) внутри указаного одного контейнера

# **Build and Tag**

**docker build .** создает doker image из нашого dokerfile , . - означает что doker file находится в текущей директории.

**docker build ./name\_folder** создание докер образа, когда Dockerfile лежит в другой директории

**docker build .** **-t my\_image** или **docker build .** **--tag my\_image** тоже создаем docker image из Dockerfile и еще даем ему свое название

**docker build . --file Dockerfile** при создании docker-образа явно указываем названия файла в котором лежат инструкции docker

**docker tag old\_name new\_name** переименование docker образа

# **Run**

**docker run my\_image** или **docker run --name my\_name my\_image** создание нового контейнера из указаного docker image и сразу запуск (не нужно приписывать docker start) . Можно дополнительноу указывать имя для контейнера

**docker run -d** **my\_image** запус контейнера в фоновом режиме (daemon режим)

**docker run -p 80 : 80 my\_image** или **docker run -p 127.0.01:80:80** -p связиивание портов, первоче число - адрес\_порт машини которая будет подключаться к docker контенеру , второе число - порт взятой програмы внутри контейнера docker. Также можна указывать ip локальной машины

**Первый порт** для обращение от пользователей к контейнеру

**Второй порт** внутри контейнера , от которого идет ответ для пользователя от контейнера. Также второй порт служит для связывания текущего контейнера с другими контейнерами через docker-compose

**docker run -it my\_image** запус контейнера из образа в интерктивном режиме

**docker exec -it my\_container\_id**  **/bin/sh** заходум внутрь запущеного контейнера

# **Already Exist Container**

**docker start my\_container** запуск уже существующего контейнера

**docker stop my\_container** или **container\_id** остановка выполнения контейнера (удаляеться кеш данных)

**docker restart my\_container**  перезапуск контейнера

**docker pause my\_container** ставим контейнер на паузу (не удаляеться кеш данных)

**docker unpause my\_container** выходим из паузы контейнера

# **Deleting**

**docker rmi my\_image** или **docker image rm 0x31233** удаляет указаный образ по image id

удаление doker image проекта

**docker rmi -f my\_image** или **docker rmi -f** **0x31233** удалить image, если даже у него есть запущенные контейнеры

**docker kill my\_container** выход с контейнера с ошибкой (если контейнер не отвечает)

**docker container rm my\_container / docker rm 90a26aa89259** или **docker stop my\_container / docker stop 90a26aa89259**  удаление(остановка) запущеного указаного контейнера (образ не удаляеться)

**docker system prune -a** удаление все образов на етом пк

**docker stop $(docker ps -q -a)**  удаление(остановка) сразу всех запущеный контейнеров

**Остановка Контейнера -> Удаление всех контейнеров даного Image (образа) -> Удаление самого Image (образа)**

# **--------- DOCKER - COMPOSE ---------**

# **Info About**

**docker-compose ps** показывает запущеные контейнеры (сервисы)

# **Build and Tag**

**docker-compose build** создаем docker образы для наших сервисов

**docker-compose up** и собрать(пересобрать) и запустить все docker контейнеры

# **Run Containers**

**docker-compose up my\_service >** запускает только один указаный сервис по его названию

**docker-compose up -d** запуск контейнеров в фоновом режиме (не блокирует консоль)

**docker-compose up --build** одновременно запускает и пересобирает наши контейнеры

**docker-compose run my\_command** позволяет запустить конйетенры и указать дополнительную комманду для выполнения

# **Inside Containers**

**docker-compose run django python manage.py migrate** выполнение команд внутри запущеного контейнера, где лежит наш django-проект

# **Already Exist Containers**

**docker-compose down** остановка контейнера и следовательно удаление его данных из памяти

**docker-compose start** запускает остановленые контейнеры

**docker-compose restart** перезапустить все контейнеры

# **-------------- DOCKER-HUB --------------**

**docker login** авторизация пользователя в терминале через акаунт docker hub

**docker logout** выход из учетной записи пользователся в терминале

**docker push my\_image** отправляем свой docker image в репозиторий. При етом локальное название docker image и название репозитория должно иметь одно название

**docker push my\_image:latest** можна явно указывать версию образа, который отправляем

**docker pull my\_image** скачивание сторонего Docker Образа, который будет основой нашего приложения

**docker pull my\_image:latest** можна явно указывать версию образа, который скачиваем