**Порты контейнеров**

Docker позволяет нам получить доступ к какому-то из портов контейнера, пробросив его наружу (в основную операционную систему). По умолчанию, мы не можем достучаться к каким-либо из портов контейнера. Однако, в *Dockerfile* опция EXPOSE позволяет нам объявить, к какому из портов мы можем обратиться из основной ОС.

Для этого запустим Docker-образ php-apache, который работает на 80 порту.

Для начала, создадим новую папку apache (перейдём в неё cd apache), в которой создадим файл index.php, на основе которого мы и поймём, что всё работает.

**<?php**

echo 'Hello from apache. We have PHP version = ' . phpversion() . PHP\_EOL;

А так же, в этой папке создадим файл Dockerfile:

FROM php:7.2-apache

# Указываем рабочую папку

WORKDIR /var/www/html

# Копируем все файлы проекта в контейнер

COPY . /var/www/html

EXPOSE 80

Пробежимся по командам:  
**FROM**: это вам уже знакомо, это образ с уже установленным php и apache  
**WORKDIR**: создаст папку если она не создана, и перейдёт в неё. Аналогично выполнению команд mkdir /var/www/html && cd /var/www/html  
**EXPOSE**: Apache по-умолчанию запускается на 80 порту, этот же порт мы пробрасываем на хост

Для работы с сетью в Docker, нужно проделать 2 шага:

* Прокинуть системный порт (*Expose*).
* Привязать порт основной ОС к порту контейнера (выполнить соответствие).
* Выполним первый шаг сделаем контейнер:
* docker build . --tag own\_php\_apache
* И после этого, запустим контейнер:
* docker run own\_php\_apache
* После чего, попробуем перейти по адресу [localhost:80](http://localhost/)
* Но, ***это не сработало***, потому что мы ещё не выполнили 2 шаг по маппингу портов.
* Выйдите из контейнера, нажав CTRL+C.

*Если у вас проблемы с остановкой контейнера, в новом окне откройте терминал, выполните docker ps, найдите ID контейнера, который сейчас запущен, и выполните docker stop {CONTAINER\_ID} (указав ваш ID контейнера)*

Теперь, осталось сообщить нашему компьютеру, какой порт контейнера ему нужно слушать, и для этого формат записи будет такой:

docker run -p <HOST\_PORT>:<CONTAINER\_PORT>

И мы можем указать любое соответствие портов, но сейчас просто укажем, что порт системы 80 будет слушать 80 порт контейнера:

docker run -p 80:80 own\_php\_apache

Здесь, вы уже наверное заметили, что добавился новый параметр -p 80:80, который говорит Docker-у: я хочу, чтобы порт 80 из apache был привязан к моему локальному порту 80.

И теперь, если перейти по адресу [localhost:80](http://localhost/), то должны увидеть успешный ответ.

ейчас, можем попробовать выполнить запуск на разных портах:

docker run -p 8080:80 own\_php\_apache

Теперь, немного подчистим за собой: нужно остановить и удалить контейнеры, которые в даный момент мы запустили:

docker ps

docker stop <CONTAINER\_ID> ...

docker rm <CONTAINER\_ID> ...

*Для \*nix пользователей есть небольшой хак, который позволит остановить и удалить все контейнеры Docker:*

docker stop $(docker ps -a -q) # Остановит все контейнеры

docker rm $(docker ps -a -q) # Удалит все остановленные контейнеры

Запомните, что любой, кто будет запускать этот код на своём компьютере, не обязан иметь установленный PHP, всё что ему нужно - один только Docker.