## Docker образ: прослойка данных и кеширование

Каждый раз, когда вы собираете образ, он кешируется в отдельный слой. Ввиду того, что образы являются неизменяемыми, их ядро никогда не модифицируются, потому применяется система кеширования, которая нужна для увеличения скорости билдинга.

*Каждая команда в Dockerfile сохраняется как отельный слой образа.*

Рассмотрим это на примере нашего прошлого Dockerfile-а:

FROM php:7.2-apache

# Копирует код ядра

COPY . /var/www/html

WORKDIR /var/www/html

EXPOSE 80

Когда вы пишите свой Dockerfile, вы добавляете слои поверх существующего основного образа (указанного в FROM), и создаёте свой собственный образ (Image).

***FROM***: говорит Докеру взять за основу этот существующий образ. А все новые команды будут добавлены слоями поверх этого основного образа.  
***COPY***: копирует файлы с основной ОС в образ  
***WORKDIR***: устанавливает текущую папку образа в /var/www/html

Docker начинает кешировать с "того места, где остановился" во время билдинга Dockerfile. Если в Докерфайле не было никаких изменений с момента последнего билдинга, то образ будет взят полностью из кеша. Если же вы измените какую-то строку в Dockerfile - кеш будет взят только тех слоёв команд, которые находятся выше изменённой команды.

Для иллюстрации этого, добавим новые строки в Dockerfile:

FROM php:7.2-apache

WORKDIR /var/www/html

# Copy the app code

COPY . /var/www/html

RUN apt-get update && apt-get upgrade -y && apt-get install -y curl php7.2-mbstring php7.2-zip php7.2-intl php7.2-xml php7.2-json php7.2-curl

RUN echo "Hello, Docker Tutorial"

EXPOSE 80

После чего, пересоберём образ:

docker build . --tag own\_php\_apache

*Когда вы используете команду*COPY*, она копирует указанную директорию в контейнер. И, в случае изменения содержимого любого из файлов этой директории, кеш команды*COPY*будет сброшен. Docker сверяет изменения во время билдинга в каждом из файлов. Если они были изменены, кеш будет сброшен, как и для всех последующих слоёв.*

Какие выводы из этого можно сделать:

1. Команды, которые вероятнее всего не будут меняться в будущем, нужно помещать как можно выше в Dockerfile.
2. Команды копирования данных нужно помещать ниже, потому что файлы при разработке изменяются довольно часто.
3. Команды, которые требуют много времени на билдинг, нужно помещать выше.

В заключение, так же хочу сказать, ***как можно уменьшить размер слоёв Docker образов***.  
В Dockerfile вы можете иметь несколько команд (RUN) на выполнение:

RUN apt-get update

RUN apt-get install -y wget

RUN apt-get install -y curl

В результате выполнения этой команды, будет создано 3 разных слоя в образе. Вместо этого, все команды стараются объединить в одну строку:

RUN apt-get update && apt-get install -y wget curl

Если команда становится длинной, и нечитаемой, то для переноса на следующую строку делаем так:

RUN apt-get update && apt-get install -y wget curl && \

&& apt-get clean -y \

&& docker-php-ext-install soap mcrypt pdo\_mysql zip bcmath

*Если же команда становится слишком большой, и неудобной для чтения, то можно создать новый shell скрипт, в который поместить длинную команду, и запускать этот скрипт одной простой командой RUN.*

*Технически, только команды****ADD****,****COPY****, и****RUN****создают новый слой в Docker образе, остальные команды кешируются*[*по-другому*](https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#minimize-the-number-of-layers)