Рассмотрим пример скачивания нашего первого образа.
Для этого, существует команда:

`docker pull <IMAGE_NAME>`, где `<IMAGE_NAME>` - имя скачиваемого образа

Зная эту команду, скачаем образ `Ubuntu 18.10`:

```
docker pull ubuntu:18.10
```

Чтобы посмотреть список всех загруженных образов, нужно выполнить:

```
docker images
```

Для запуска контейнера существует команда:

```
docker run <image> <опциональная команды, которая выполнится внутри контейнера>
```

Давайте запустим наш первый контейнер Ubuntu:

```
docker run ubuntu:18.10 echo 'hello from ubuntu'
```

Теперь выполним команду для проверки списка запущенных контейнеров:

```
docker ps
```

`docker ps` показывает только список контейнеров, которые запущены в данный момент (наш же контейнер выполнил одну команду `echo 'hello from ubuntu'` и завершил свою работу).

А для того, чтобы посмотреть список всех контейнеров без исключения, нужно добавить флаг `-a`, выполним:

```
docker ps -a
```

Выполнение неограниченное количество команда внутри контейнера

Мы можем подключиться к консоли виртуальной ОС (`Ubuntu 18.10`), и выполнять любое количество команд без завершения работы контейнера, для этого, запустим команду:

```
docker run -it ubuntu:18.10 /bin/bash
```

Опция `-it` вместе с `/bin/bash` даёт доступ к выполнению команд в терминале внутри контейнера *Ubuntu*.

Теперь, внутри этого контейнера можно выполнять любые команды, применимые к `Ubuntu`. Вы же можете представлять это как мини виртуальную машину, условно, к консоли которой мы подключились по *SSH*.

Узнаём ID контейнера

Иногда является очень полезным узнать ID контейнера, с которым мы работаем. И как раз-таки, при выполнении команды `docker run -it <IMAGE> /bin/bash`, мы окажемся в терминале, где все команды будут выполняться от имени пользователя `root@<containerid>`.

Теперь откройте **новое окно терминала** (не закрывая и не отключаясь от текущего), и выполните команду `docker ps`

Теперь вернёмся назад к первому окну терминала (который находится внутри контейнера), и выполним:

```
mkdir /truedir   #создаст папку truedir
exit #выйдет из контейнера, и вернётся в основную ОС
```

Выполнив команду `exit`, контейнер будет остановлен (чтобы убедиться, можете проверить командой `docker ps`). Теперь, вы так же знаете, **как выйти из Docker контейнера**.

Теперь, попробуем ещё раз просмотреть список всех контейнеров, и убедимся, что новый контейнер был создан `docker ps -a`

Так же, для того, чтобы запустить ранее созданный контейнер, можно выполнить команду `docker start <CONTAINER_ID>`,
где **CONTAINER_ID** - id контейнера, который можно посмотреть, выполнив команду `docker ps -a` (и увидеть в столбце CONTAINER_ID)

Запустим контейнер командой:

```
docker start <Введи номер контейнера>      #ваш CONTAINER_ID
docker ps
docker exec -it <Введи номер контейнера> /bin/bash   #ваш CONTAINER_ID
```

И теперь, если внутри контейнера выполнить команду `ls`, то можно увидеть, что ранее созданная папка truedir *существует в этом контейнере*

Для выхода, как обычно, выполним `exit`.
Теперь остановим и **удалим Docker контейнеры** командами:
`docker stop <CONTAINER_ID>`
`docker rm <CONTAINER_ID>`

```
docker ps a    # просмотрим список активных контейнеров
docker stop <Введи номер контейнера> # остановим активный контейнер
docker rm <Введи номер контейнера> # удалим контейнер
docker rm <Введи номер контейнера> # удалим второй контейнер
```

В основном, нам не нужно, чтобы в системе плодилось большое количество контейнеров. Потому, команду `docker run` очень часто запускают с дополнительным флагом --rm, который удаляет запущенный контейнер после работы:
`docker run -it --rm ubuntu:18.10 /bin/bash`