Теория

<https://cloud.yandex.ru/blog/posts/2022/03/docker-containers>

<https://1cloud.ru/help/docker/docker_image_work>

<https://1cloud.ru/help/docker/docker_container>

<https://1cloud.ru/blog/docker_start>

<https://1cloud.ru/help/linux/instruktsiya-docker-na-centos7>

Docker – это средство или система упаковки, доставки и запуска приложений. Он позволяет запускать приложения, написанные на разных языках программирования благодаря унифицированному подходу к упаковке, доставке и запуску приложений.

Какие преимущества Docker приносит компаниям и разработчикам?

* Гибкость и адаптивность. Благодаря Docker можно легко запускать контейнер в облачной инфраструктуре и на любом локальном устройстве. Можно создать базовые шаблоны контейнеров и использовать повторно бесконечное число раз. Бесшовная переносимость и простота развертывания — важные преимущества этой технологии.
* Меньше ошибок и несовпадений окружений. В контейнерах Docker содержится всё, что требуется для запуска приложения, поэтому перенос приложений из одной среды в другую не вызывает затруднений. Исчезает проблема, когда у разработчиков всё функционирует как надо, а на боевом сервере — нет. Пример того, как это помогает в бизнесе, от компании «Бухта».
* Скорость развертывания. Так как настраивать окружение для разработки, тестирования и боевого режима больше не нужно, время развертывания сокращается в несколько раз.
* Рост универсальности. Docker позволяет использовать любые языки программирования и стек технологий на сервере, избавляя от проблемы несовместимости разных библиотек и технологий.
* Комьюнити и поддержка. Существует огромная библиотека контейнеров с открытым исходным кодом. Можно скачать нужный образ для конкретной задачи или обратиться за помощью к большому комьюнити разработчиков, которые используют Docker.
* Непрерывность работы. С учетом инструментов управления трафиком можно построить процесс обновления приложения так, чтобы обновление одних контейнеров не влияло на работоспособность системы и оказание услуг пользователям.
* Упрощение администрирования. С помощью Docker легче перенести контейнер с одного хоста на другой, запустить сразу несколько образов, обновить группы контейнеров и откатиться к старой версии.
* Повышение уровня безопасности. Контейнеры в Docker частично изолированы друг от друга на уровне процессов и ОС, поэтому запуск большого количества контейнеров на одной машине не несет рисков.
* Экономическая эффективность. Контейнеры легковесны и производительны, а благодаря использованию Docker можно эффективнее управлять имеющимися ресурсами и сократить расходы компании.
* Современный подход. Отказ от монолитной архитектуры в пользу микросервисной позволяет более гибко развивать продукт, добавлять в него новые функции. Эффективность такого подхода показывает

Основные компоненты Docker:

* Dockerfile. Текстовый файл с последовательно расположенными инструкциями для создания образа Docker. Файл создаётся по принципу «одна строка — одна команда».
* Daemon. Фоновая служба на хосте, которая отвечает за создание, запуск и уничтожение контейнеров.
* Image. Неизменяемый файл (образ), из которого можно неограниченное количество раз развернуть контейнер.
* Client. Утилита командной строки в Docker для управления демоном. Любое взаимодействие с контейнером проходит через Daemon.
* Container. Запущенное приложение, которое развернули из образа.
* Registry. Служба в Docker, выполняющая функции репозитория (хранилища). Позволяет следить за версиями образов, создавать приватные репозитории.
* Docker Hub. Популярный публичный репозиторий, используемый по умолчанию в Docker. Обеспечивает интеграцию с GitHub и BitBucket.
* Docker Desktop. Приложение, позволяющее локально собирать, выполнять и тестировать контейнеры. Работает на Windows и macOS.
* Docker volumes. Тома для постоянного хранения информации. По умолчанию в Docker папки хранилищ создаются на хост-машине, но предусмотрена и возможность подключения удаленных хранилищ. Использование томов позволяет лучшим образом настроить хранение данных.

**Практика**

*Установка под Centos 7*

Обновление списка пакетов:

yum check-update

#проверка наличия последних обновлений для установленных пакетов

Добавление репозитория и установка стабильной версии Docker:

curl -fsSL https://get.docker.com/ | sh

#Ключи curl:

#-s – бесшумный режим, не показывает прогресс и ошибки;

#-S – показывает ошибки, вместе с ключом -s также будет работать;

#-f – быстрый сбой без вывода сообщений при ошибках сервера;

#-L – автоматический переход по другому адресу, если ресурс перемещён;

#pipe sh – вызов командного интерпретатора.

Добавление пользователя для Docker, ибо из-под root не комильфо:

useradd udocker

passwd udocker

usermod -aG docker udocker

#useradd добавляет пользователя udocker без пароля с домашней директорией

#в /home/udocker;

#утилита passwd задаёт пароль для нового пользователя udocker;

#Утилита usermod управляет записями пользователей, ключи:

#-a – добавляет пользователя в одну или несколько дополнительных групп,

#работает только в связке с ключом -G;

#-G – указать список дополнительных групп, в которые должен водить

#пользователь.

Запуск демона Docker с последующей проверкой состояния:

systemctl start docker && systemctl status docker

*Работа с образами Docker*

База используемого синтаксиса:

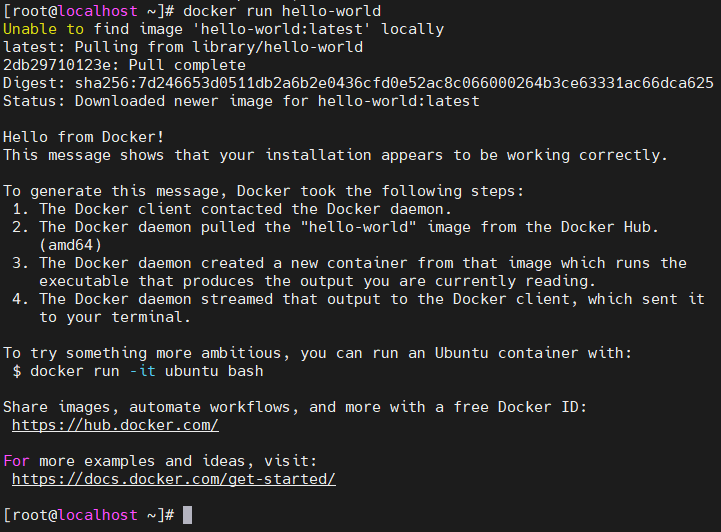
docker <option> <command> <arguments>

Вывод на экран всех доступных команд с краткой информацией о них:

docker

Проверка получения и загрузки образов на машину:

docker run hello-world



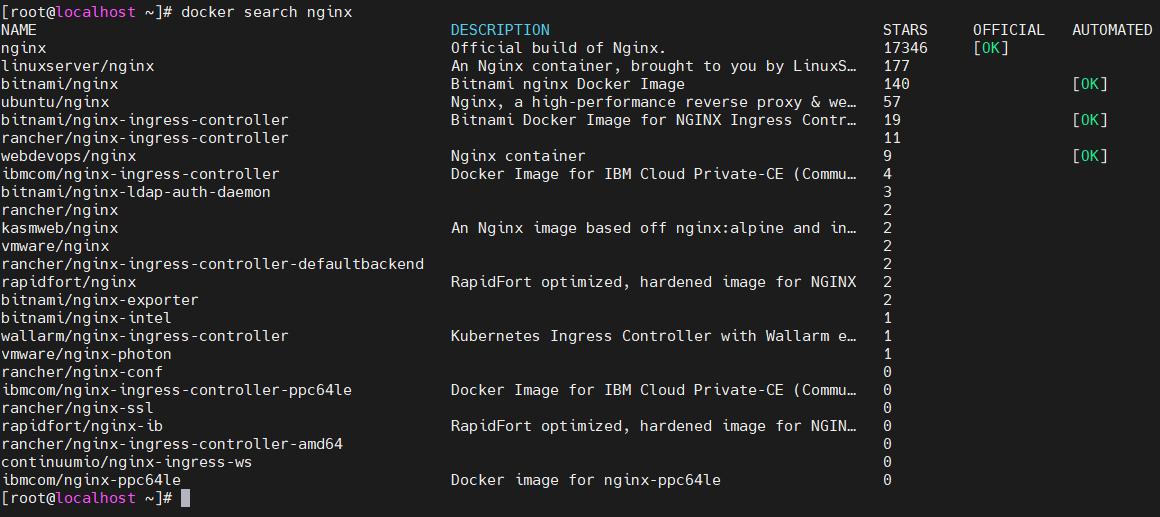
#docker run позволяет запустить контейнер из образа hello-world, если образ

#отсутствует на локальной машине, то он будет загружен из реестра.

Поиск необходимого образа:

docker search <name>

docker search nginx

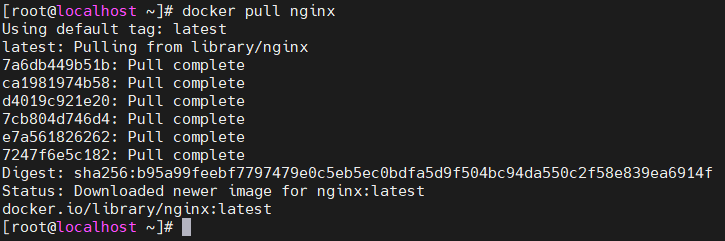


#Поиск образов в Docker Hub

Загрузка необходимого образа:

docker pull <name>

docker pull nginx



После загрузки образа, его также можно запустить:

docker run -d -p 80:80 nginx



#-d – запуск контейнера в фоновом режиме;

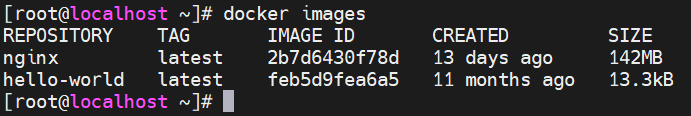
#-p – проброс портов, конструкция 80:80 говорит о том, что номер порта слева

#будет портом, который смотрит во внешний мир, а номер порта справа

#будет являться портом контейнера

Просмотр всех загруженных контейнеров:

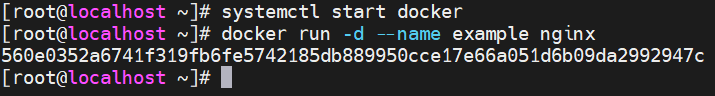
docker images



Задание имени контейнеру:

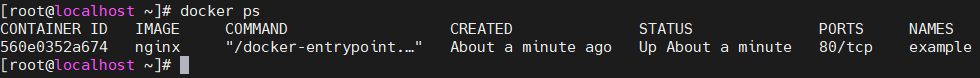
docker run -d --name <name> <image>

docker run -d --name example nginx



Просмотр запущенных контейнеров:

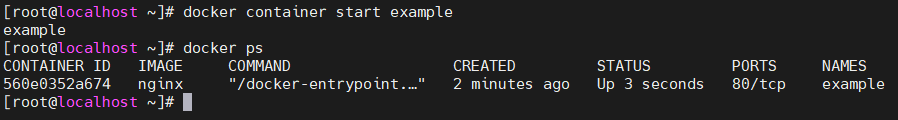
docker ps



Запуск докер контейнера в фоновом режиме:

docker container start <имя\_или\_id>

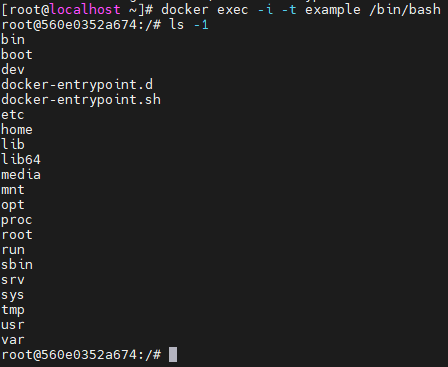
docker container start example



Зайти внутрь контейнера:

docker exec -i -t <имя\_или\_id> /bin/bash

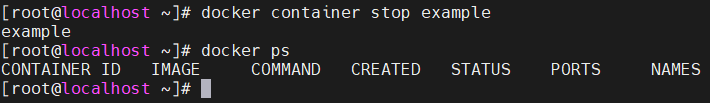
docker exec -i -t example /bin/bash



Остановить работу контейнера:

docker container stop <имя контейнера>

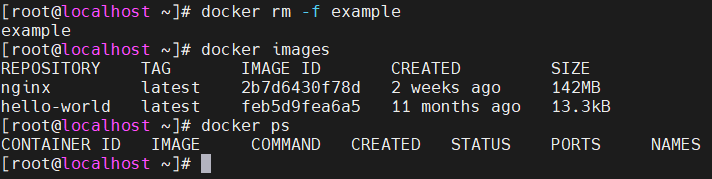
docker container stop example



Удаление контейнера:

docker rm -f <имя\_или\_id>

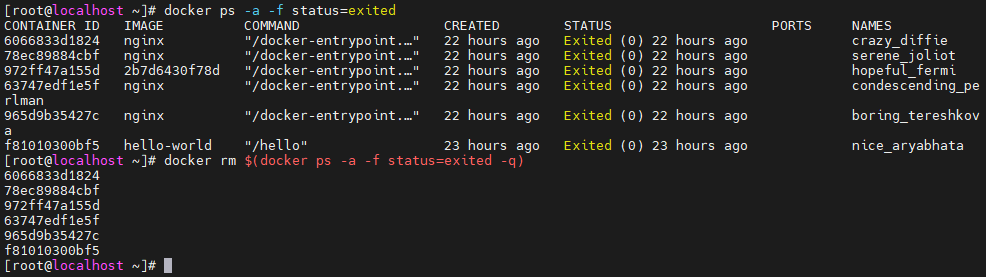
docker rm -f example



Найти все контейнеры в различном состоянии с последующим полным удалением:

docker ps -a -f status=exited

docker rm $(docker ps -a -f status=exited -q)



#-a – вывести список всех остановленных контейнеров

#Затем удаление всех остановленных контейнеров

Удаление образов:

docker images -a

docker rmi <name>

