Docker - платформа контейнеризации с открытым исходным кодом, которая позволяет разработчикам упаковывать, доставлять и запускать приложения в контейнерах.

Что дает:

* Приложения в контейнерах не будут конкурировать за ресурсы (цп, оператива)
* Изолирование. Процессы в контейнере имеют ограниченное представление о системе. Они не видят процессов других контейнеров и хоста, так же не видят файлы, относящиеся к другим контейнерам.
* Так же изолируется и сетевой стек. Можно запускать приложения, работающие на одних и тех же портах.
* Нет конфликтов, когда приложения используют разные версии одних и тех же библиотек, либо других зависимостей (например одно приложение работает на 8 джаве, второе на 11). Каждый образ контейнера несет в себе все зависимости, необходимые приложению.
* Облегчается доставка приложений, и их переносимость. Чтобы запустить приложение достаточно чтобы на компьютере был докер. Отсутствуют ошибки, связанные с различием в окружении (у разработчика работает на компе, у тестировщика нет).
* Контейнеры гораздо легче виртуалок, т.к. контейнеры это просто процессы, запускаемые в операционной системе. Благодаря разбиению фаловой системы на уровни, несколько контейнеров могут использовать общие уровни (например 2 контейнера с разными джава приложениями, будут отличаться только парой верхних слоев, отвечающих за само приложений. Нижние слои будут общими).
* Позволяет легко масштабировать приложения, запуская дополнительные контейнеры.

**Осноные компоненты Docker**

* **Docker daemon (**демон Docker**) –** ответственный за создание, запуск и контроль работы контейнеров, и за создание и хранение образов. Также управляет сетями и хранилищами. Запускается командой **docker daemon**, обычно его запускает операционная система. Демон может связываться с другими демонами для управления сервисами Docker.
* **Клиент Docker –** пользовательский интерфейс, используется для взаимодействия с демоном по протоколу HTTP. Можно без затруднений организовать соединение с удаленными демонами Docker. Клиент Docker может взаимодействовать с несколькими демонами.
* Docker-образ – файл, на основе которого создаются контейнеры. Включает в себя зависимости и конфигурацию приложения. Образы описываются в Dockerfile.

Основные команды:

* **docker run image –** запуск контейнера
* **docker ps –** просмотр списка запущенных контейнеров.
* **docker logs <container> -** логи указанного контейнера.
* **docker stop <container> -** остановить контейнер
* **docker exec –** выполнить команду внутри контейнера.
*  **-** запустить командную оболочку в контейнере. Сможем выполнять команды непосредственно в контейнере.
* **docker build –** собрать образ, на основе Dockerfile

**Dockerfile –** файл с инструкциями, которые необходимо выполнить для создания образа. Основные инструкции:

**FROM образ –** базовый образ, на основе которой создается текущий.

**RUN –** запустить команду при сборке контейнера. Обычно используется для установки зависимостей, библиотек.

**ENTRY-POINT –** точка входа в программу. Исполняемый файл, либо команда, которая будет выполнена при запуске контейнера.

**CMD –** аргументы для **ENTRY-POINT.** Если отсутствует ENTRY-POINT, то при запуске контейнера выполняется команда из CMD.

**COPY –** скопировать файл с хоста в контейнер.

**ENV –** переменные окружения. На них можно ссылаться как в Dockerfile, так и во время работы контейнера.

**EXPOSE –** указание, что приложение в контейнере будет слушать такие-то порты. Эти порты нужно прокидывать в контейнер при запуске, либо докер использует их при автоматическом пробросе портов.

**VOLUME –** путь к фалам, который монтируются на хост.

**WORKDIR –** рабочий каталог. В нем будут выполнятся все последующие команды в Dockerfile, а также команды запускаемые в *exec*.

**ТОМА (VOLUMES) –** папки из контейнера, которые хранятся непосредственно на хосте. Используется в основном для сохранения данных между перезапусками контейнера (например, контейнера с БД). В редких случаях могут использоваться для обмена данными между контейнерами.

**NETWORKS –** технология, позволяющая объединить несколько контейнеров в одну сеть. Контейнеры, добавленные в сеть смогут взаимодействовать друг с другом. В качестве имени хоста в таком случае используется имя контейнера.

**Docker-compose –** инструмент для запуска приложений, состоящих из нескольких файлов.

Каждый контейнер описывается как сервис в файле docker-compose.yml. Мы можем выставлять зависимости между сервисами, пробрасывать порты, настраивать переменные окружения, объединять контейнеры в сеть, добавлять контейнерам volumes, настраивать ограничения ресурсов и др.

Все описанные сервисы затем можно запустить одной командой: **docker-compose up.**