Docker и все все все!!!

Что такое Docker и зачем он нужен?

Докер — это открытая платформа для разработки, доставки и эксплуатации приложений. С помощью docker вы можете отделить ваше приложение от вашей инфраструктуры и обращаться с инфраструктурой как управляемым приложением.

Докер написан на Go и использует некоторые возможности ядра Linux(cgroups, namespaces).

1. Что такое Docker и зачем он нужен?
2. Отличие VM(virtual machine) от контейнера.
3. Union filesystem.
4. Что такое образ?
5. Что такое контейнер?
6. DockerHub.
7. Базовые команды.
8. Практикум.

Небольшой пример. Процесс локального тестирования приложения.

Самой большой проблемой в данном случае является установка и развертывание требуемой версии вашего приложения, могут возникать дополнительные трудности из-за соблюдения критически важных особенностей (Глобальные переменные, определенные версии библиотек, специфические ОС и т.д). Как решить данную проблему:

1. Установочный скрипт.
2. Облачное решение.
3. Виртуальная машина.
4. Контейнеризация.

Отличие VM(virtual machine) от контейнера.

Важно отметить, что эти технологии решают один и тот же набор задач, просто их подходы слегка отличаются. В случае с VM виртуализация происходит на аппаратном уровне, когда как у контейнера на уровне ОС. Т.к это часть ОС то это открывает огромные возможности по переиспользованию имеющихся ресурсов. Речь идет о файловой системе, оперативной памяти и процессорных мощностях.

P.S Некоторые VM тоже так умеют, но это требует очень тонкой настройки.

Union Filesystem.

Union File Sysem или UnionFS — это файловая система, которая работает создавая уровни, делая ее очень легковесной и быстрой. Docker использует UnionFS для создания блоков, из которых строится контейнер.

Что такое образ?

Основной элемент, из которого создаются контейнеры. Образ создаётся из Dockerfile, добавленного в проект и представляет собой набор файловых систем (слоёв) наслоённых друг на друга и сгруппированных вместе, доступных только для чтения; максимальное число слоёв равно 127.

В основе каждого образа находится базовый образ, который указывается командой FROM — входная точка при формировании образа Dockerfile. Каждый слой является readonly-слоем и представлен одной командой, модифицирующей файловую систему, записанной в Dockerfile. Данный подход позволяют разным файлам и директориям из разных файловых слоёв прозрачно накладываться, создавая каскадно-объединённую файловую систему. Слои содержат метаданные, позволяющие сохранять сопутствующую информацию о каждом слое во время выполнения и сборки. Каждый слой содержит ссылку на следующий слой, если слой не имеет ссылки, значит это самый верхний слой в образе.

Что такое контейнер?

Абстракция на уровне приложения, объединяющая код и зависимости. Контейнеры всегда создаются из образов, добавляя доступный для записи верхний слой и инициализирует различные параметры. Т.к. контейнер имеет свой собственный слой для записи и все изменения сохраняются в этом слое, несколько контейнеров могут совместно использовать доступ к одному и тому же образу.

DockerHub

Docker Hub предоставляет публичные и приватные хранилища образов. Поиск и скачивание образов из публичных хранилищ доступно для всех. Содержимое приватных хранилищ не попадает в результат поиска. И только вы и ваши пользователи могут получать эти образы и создавать из них контейнеры.

https://hub.docker.com

Базовые команды

1. docker pull {imageId} - Получение образа из некого источника(by default - DockerHub).
2. docker images - Просмотр доступных локально образов.
3. docker ps -a - Просмотр всех локальных контейнеров. Флаг -a = all
4. docker history {imageId} - Просмотр образа по слоям.
5. docker run {imageId} - Запуск контейнера на основании образа.
6. docker stop {containerId} - Остановка контейнера.
7. docker kill {containerId} - Остановка контейнера, более жесткая.
8. docker logs {containerId} - Просмотр логов контейнера.
9. docker rm - Удаление контейнера.
10. docker rmi - Удаление образа.
11. docker exec {containerId} {command} - Выполнение какой-либо команды на контейнере.

Практикум

Выкачаем всем известный и любимый образ ubuntu и посмотрим из каких слоев он состоит.

a. docker pull ubuntu - скачиваем нужный нам образ.
b. docker images - Давайте посмотрим какие образы у нас уже есть.
c. Docker history ubuntu - Давайте посмотрим из каких слоев он состоит.

Запустим образ ubuntu, посмотрим на логи

a. docker run --rm -it ubuntu (В случае если у вас еще нет этого образа начнется автоматический процесс скачивания.)
b. Откроем новый терминал, в Mac OS это можно сделать сочетанием клавиш ctrl + T.
c. docker ps -a - Посмотрим какие контейнеры у нас подняты(-a == All, в противном случае выведет только живые).
d. docker logs {containerId} - Посмотрим логи нашего контейнера.
e. Завершим сеанс работы с оболочкой.
f. docker ps -a - Посмотрим какие контейнеры у нас подняты(-a == All, в противном случае выведет только живые).

Остановим работающий контейнер и попытаемся понять разницу между stop и pause.

a. docker -it run ubuntu
b. Откроем новый терминал, в Mac OS это можно сделать сочетанием клавиш ctrl + T.
c. docker stop {containerId} - Остановим контейнер.
d. docker logs {containerId} - Попробуем прочитать логи с остановленного контейнера.
e. docker start {containerId} - Восстановим его работоспособность.
f. docker pause {containerId} - Поставим контейнер на паузу.
g. docker stop {containerId} - Остановим контейнер.
h. docker rm {containerId} - Удалим контейнер.

Заключительная часть, а что если мы хотим немного больше.

Давайте попробуем поставить перед собой более менее серьезную задачу и выполнить ее, например: Поднимем какой-либо веб сервер и развернем на нем веб страничку.

Всего лишь одна команда!!

```
docker run --name some-nginx -p 8080:80 -v
~/IdeaProjects/TimeTrackingHelper/src/main/resources/html/:/usr/share/nginx/html -d
nginx
```