Dockerfile

Цель: Ознакомиться с принципами работы в Docker, научится собирать Docker-образы на основе dockerfile.

# Теория:

Для сборки контейнеров из образа используйте каманду (bash):

docker run -d -p 80:80 docker/getting-started

Данная команда соберет образ getting-started. Опция -d запускает докер в фоновом режиме, -p открывает 80 порт

Для сборки образа может быть использован dockerfile.

Dockerfile — это конфигурационный файл, в котором описаны инструкции, которые будут применены при сборке Docker-образа и запуске контейнера. Dockerfile создается в корневой директории проекта и не имеет расширения.

Синтаксис Dockerfile близок к синтаксису конфигурационных файлов .ini. На каждую строчку приходится одна инструкция. Инструкции пишутся капсом, а их значения отделяются пробелом.

Dockerfile имеет следующую логику заполнения:

1. Первой инструкцией всегда идѐт FROM с указанием родительского образа. Например,

FROM python:latest

1. Инструкция RUN может принимать конвейер команд Linux, чтобы не создавать лишние слои. Например,

RUN apt-get update && apt-get install python3-pip -y && pip install --upgrade pip && pip install pipenv

1. Инструкция WORKDIR устанавливает рабочий каталог контейнера. Последующие

команды RUN, CMD, ENTRYPOINT наследуют привязку WORKDIR

WORKDIR /usr/src/app/

1. Завершающей инструкцией всегда идѐт CMD. CMD наследует привязку к WORKDIR, поэтому web\_interface.py будет запущен из папки /usr/src/app/.

CMD ["python", "web\_interface.py"]

Ниже приведен пример несложного Dockerfile.

#Указываем основной Docker-образ, в данном случае репозиторий python последняя версия образа

FROM python:latest

#Запускаем update, устанавливаем pip, обновляем pip, из pip устанавливаем pipenv

RUN apt-get update && apt-get install python3-pip -y && pip install -- upgrade pip && pip install pipenv

#Создаем папку app и переходим в нее RUN mkdir -p /usr/src/app/

WORKDIR /usr/src/app/

#Копируем содержимое текущего каталога в app COPY . /usr/src/app/

#Указываем какие порты надо открыть

EXPOSE 5000

#Указываем, что при запуске контейнера необходимо выполнить python- скрипт hello.py, который должен находится в папке app

CMD ["python", "hello.py"]

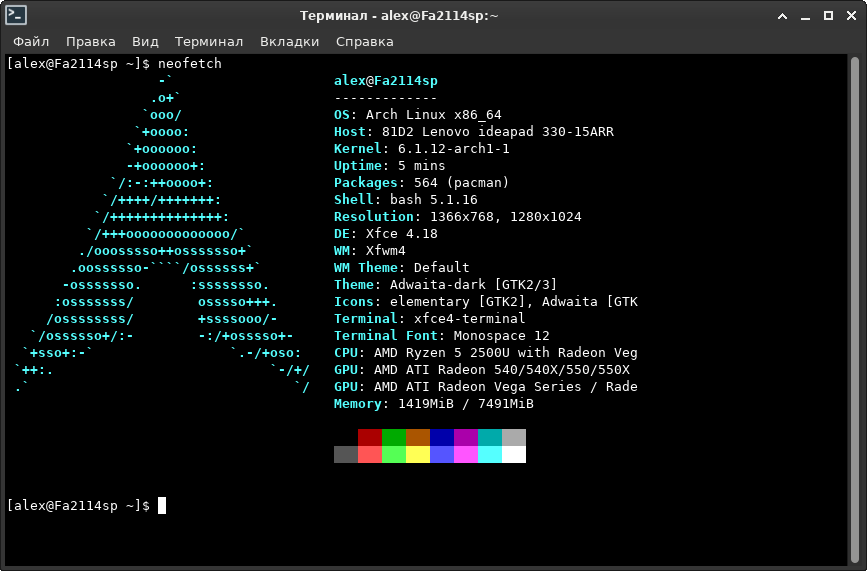
Для поиска Docker-образов можно воспользоваться ресурсом hub.docker.com, либо можно использовать команду *docker search <имя требуемого образа>.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Инструкция** | **Описание** | **Пример использования** | **Комментарий** |
| FROM | Задает базовый образ. Все последующие инструкции создают слои поверх родительского образа. | FROM python:latest FROM debian:wheezy | Быстрее всего можно найти образ с нужным тегом на [Docker](https://hub.docker.com/) [hub](https://hub.docker.com/). |
| RUN | Выполняет команду внутри контейнера и сохраняет результат. | RUN mkdir /usr/src/app/  RUN apt-get update && apt-get install python3-pip -y | RUN может исполнять конвейер команд с логическими  операторами && и ||. |
| COPY | Копирует файлы и папки из текущей директории, где находится пользователь в указанную директорию в контейнере | COPY . /usr/src/app/ | COPY считывает позицию  пользователя на хосте, поэтому первым аргументом идет «.». |
| ADD | Копирует файлы и папки из текущей позиции  пользователя, скачивает файлы по URL и работает с tar-архивами. | ADD  https://1cloud.ru/archive/api\_config.ini  /usr/src/app/ | Официальная документация не рекомендует применять ADD. Для скачивания по URL можно использовать RUN с CURL или WGET, а для копирования — COPY. |
| CMD | Выполняет команду с  указанными аргументами во время запуска контейнера. | CMD ["python", "web\_interface.py"] | CMD должна быть одна в конце Dockerfile. CMD может вызывать исполняемый файл — .sh  Аргументы docker run  переопределяют CMD. Если в Dockerfile нет CMD, обязательно должна быть инструкция ENTRYPOINT. |
| ENTRYPOINT | Похожа на CMD, но при запуске контейнера не  переопределяется в отличие от CMD. | ENTRYPOINT ["python", "web\_interface.py"] | ENTRYPOINT может  использоваться совместно с CMD. |
| ENV | Задает переменные среды | ENV ADMIN="ivan" | ENV часто применяется для |

|  |  |  |  |
| --- | --- | --- | --- |
| **Инструкция** | **Описание** | **Пример использования** | **Комментарий** |
|  | внутри образа, на которые могут ссылаться другие инструкции. |  | передачи информации в контейнеризированное  приложение через переменные среды. |
| ARG | Задает переменные, значение которых  передается докером во время сборки образа. | ARG maintainer=ivan | В отличие от ENV-переменных, ARG-переменные недоступны во время выполнения контейнера. |
| WORKDIR | Устанавливает рабочую директорию контейнера. | WORKDIR /usr/src/app/ | Последующие инструкции CMD,  RUN, ENTRYPOINT наследуют привязку к директории  установленной WORKDIR. |
| VOLUME | Создает и подключает  постоянный том хранения данных. | VOLUME /data\_cont\_1 | Просмотреть существующие тома можно командой docker volume ls. К контейнеру можно  подключить существующий том, для этого достаточно указать  уже существующий том. |
| EXPOSE | Указывает планируемый рабочий порт у контейнера. Инструкция сама по себе не открывает порт. Чтобы  использовался указанный в EXPOSE порт — нужно  указать docker run -P при запуске контейнера. | EXPOSE 5000 | Если требуется пробросить и сопоставить разные порты внутри и снаружи контейнера используется docker run -p  внутренний порт:внешний порт |
| LABEL | Добавляет метаданные в образ. | LABEL maintainer="katkov\_ivan" | Обычно LABEL содержит информацию об авторе образа. |

# Задание:

1. Установить Docker Desktop на машину с Linux ОС. Можно воспользоваться инструкцией с официального сайта: <https://docs.docker.com/desktop/>



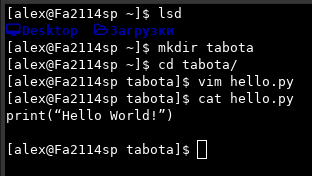
Работу решил делать в arch linux так как fedora отказалась проботать после не правального выключения

Команды для установки докера:

* sudo pacman -Syu
* sudo pacman -S docker
* sudo pacman -S docker-compose
* sudo systemctl start docker.service
* sudo systemctl enable docker.service
* sudo docker version
* sudo docker info
* sudo usermod -aG docker $USER
* sudo reboot

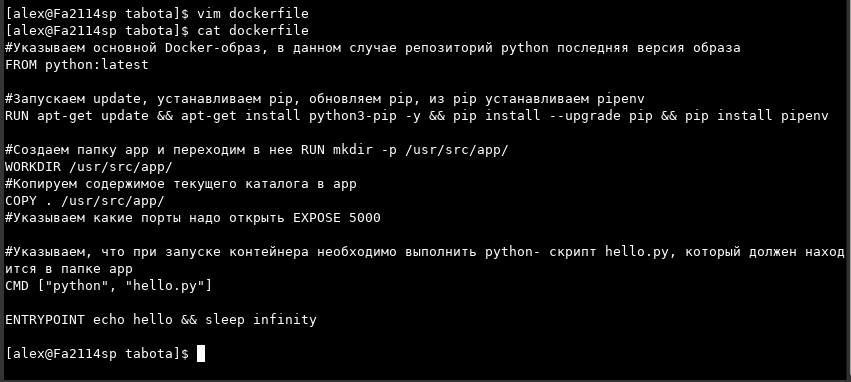
1. В домашнем каталоге пользователя Linux создать каталог (в качестве имени задать вашу фамилию). В каталоге создать файл hello.py со следующим содержимым:

print(“Hello World!”)



Создание каталога tabota и файла с расширением py

3. В этом же каталоге создать файл dockerfile и скопировать в него пример из теории

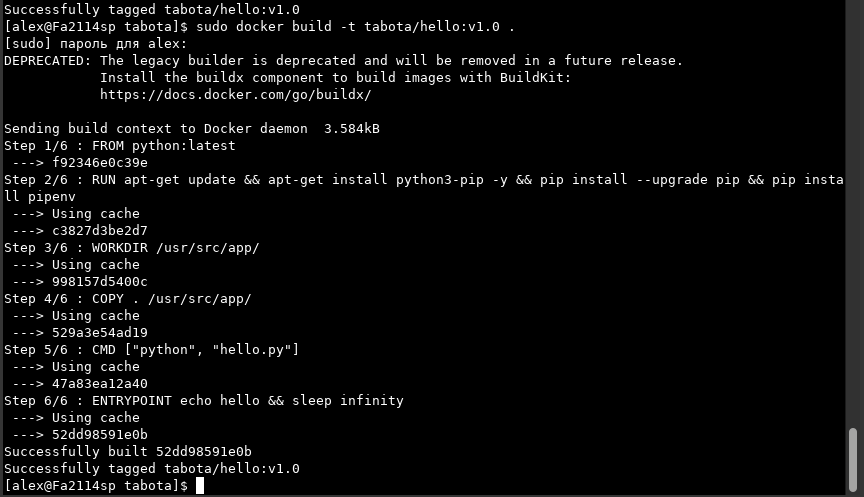


Создание dockerfile из примера, но с маленьким изменением в конце

ENTRYPOINT echo hello && sleep infinity //я добавил немного упоминания о bash что бы в будущем подключится к контейнеру

4. Соберем образ из dockerfile:

docker build -t my\_name/image\_name:my\_tag



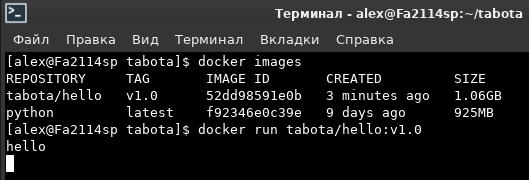
Контейнер собран

5. Теперь можем запустить наш образ:

#просмотр списка образов docker image

#создание и запуск контейнера на основе образа в фоновом режиме

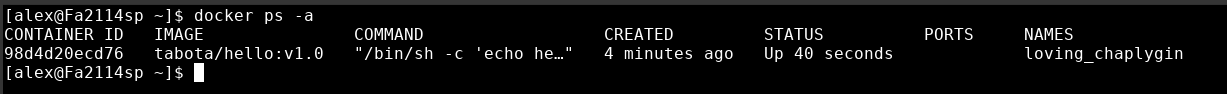
docker run -d my\_name/image\_name:my\_tag



Сейчас мы запустили контейнер в реальном времени и видим что он ждёт таким образом строчка которую мы добавили в 3 пункте отрабатывает теперь остановим данный контейнер и запустим его в фоне с помощью префикса -d

6.Для просмотра списка запущенных контейнеров можно воспользоваться:

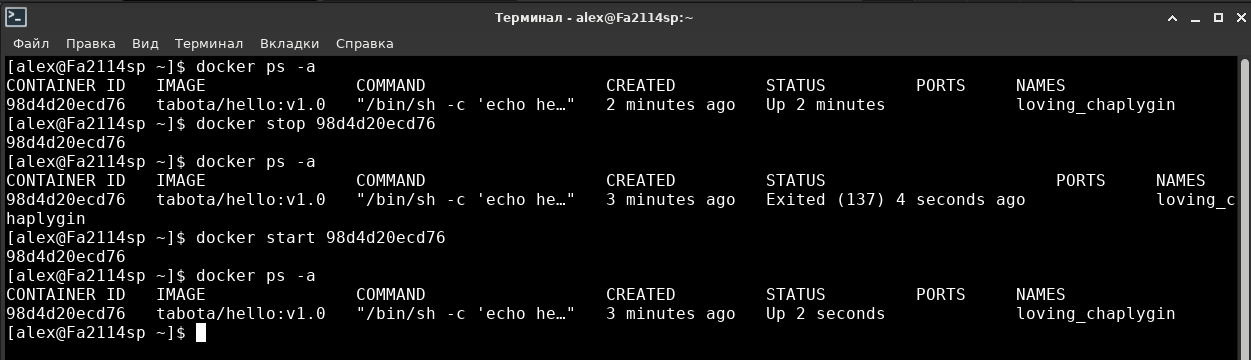
docker ps -a



Команда работает

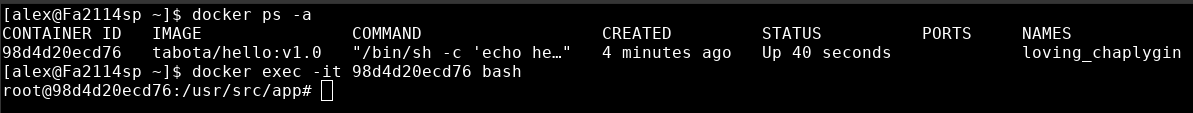
7. Для запуска и остановки созданных контейнеров:

docker stop <container\_id> docker start <container\_id>



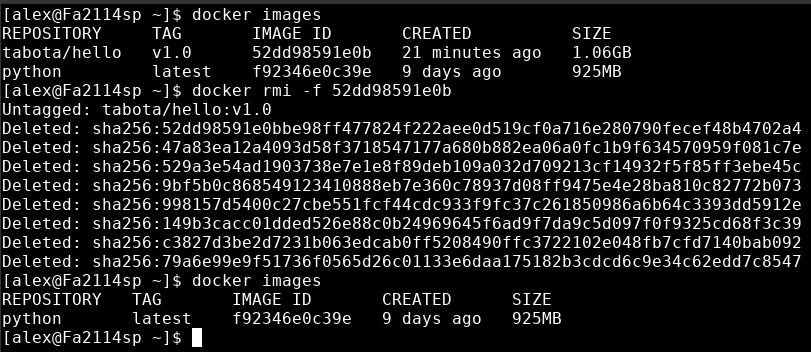
Тут тоже работает

8. Для доступа к терминалу контейнера необходимо использовать: docker exec -it <container\_id> bash



И вот тут срабатывает так самая строчка из начала работы

9. Удалить контейнер, образ и dockerfile



10. Собрать образ на основе нового dockerfile. Образ должен быть построен на основе library/httpd и выполнять роль HTTP сервера. Необходимо, что бы основная страница находиласть на хостовой машине (не внутри контейнера) и была доступна по адресу http://loclalhost:8080. На странице разместить свою фамилию и группу.

Контейнер я поместил на [github](https://github.com/Yang2114/webapache2)

Обратная связь)

Работы очень интересные местами нужно немного думать и мне это нравится)

Надеюсь дальше тоже будут тоже интересные работы

Так же хотелось ещё работ по докеру)