**Лабораторная работа №9: Работа с Docker-образами через Windows**

**Цель:** Научиться работать с Docker образами.

**Задания:**

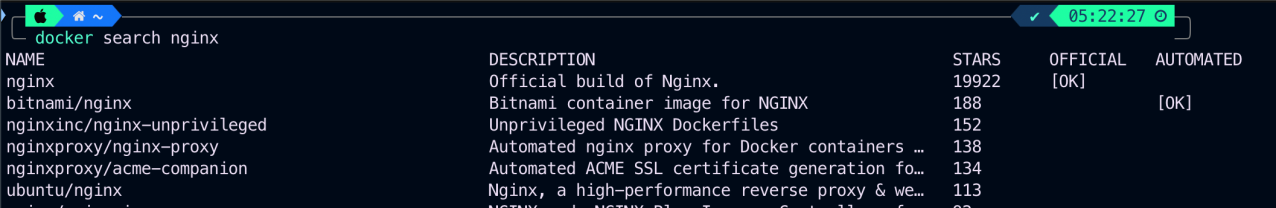
1. Поиск и загрузка Docker образа из Docker Hub.
2. Создание контейнера из образа.
3. Изучение Dockerfile.
4. Построение Docker образа из Dockerfile.
5. Запуск контейнера из созданного образа.

**Первое задание:** Поиск и загрузка Docker образа из Docker Hub.

**Поиск образа в Docker Hub**

* Откройте командную строку или терминал.
* Используйте команду `docker search` для поиска образов. Например, для поиска образа с Nginx:

docker search nginx



* Также, вы можете использовать веб версию для поиска – hub.docker.com

**Загрузка образа из Docker Hub**

* Используйте команду `docker pull` для загрузки образа. Например:

docker pull nginx

Также, вы можете указать специфическую версию, например:

docker pull nginx:latest

**Второе задание:** Создание контейнера из образа

**Запуск контейнера**

* Используйте команду `docker run` для создания и запуска контейнера. Например:

docker run -d --name my-nginx -p 8080:80 nginx

В команде указаны следующие опции:

- -d: запустить контейнер в фоновом режиме (detached mode).

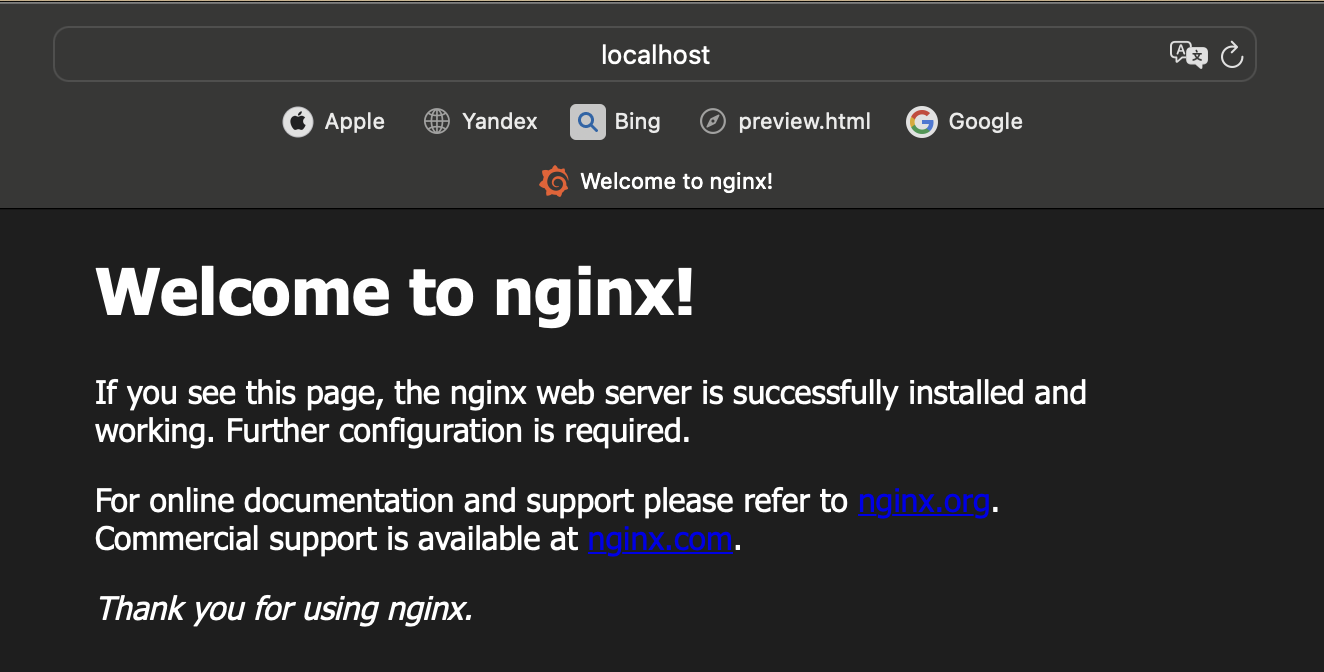
- --name my-nginx: задать имя контейнеру.

- -p 8080:80: проброс порта 8080 на хосте к порту 80 в контейнере.

В конце указывается название или ID докер образа, который мы запускаем

**Проверка работы контейнера**

* Откройте браузер и перейдите по адресу http://localhost:8080. Вы должны увидеть страницу по умолчанию Nginx.



**Третье задание:** Изучение Dockerfile

**Создание простого Dockerfile для Python приложения**

Создайте директорию для вашего проекта и перейдите в нее.

Напишем простое приложение на Python и положим его в файл app.py

from flask import Flask  
import os  
  
app = Flask(\_\_name\_\_)  
  
  
@app.route('/')  
def hello():  
 return os.getenv("GREETING", "Hello, Docker!")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app.run(host="0.0.0.0")

Далее мы создадим файл **requirements.txt** с зависимостями, требуемыми для этого приложения:

flask==3.0.3

Затем, чтобы собрать из этого приложения докер-образ. Создайте файл **Dockerfile** со следующим содержимым:

# Берём за основу уже существующий образ, содержащий установленный python

FROM python:3.12-slim  
  
# Установить рабочую директорию  
WORKDIR /app  
  
# Скопировать все файлы в рабочую директорию  
COPY . /app  
  
# Установить зависимости  
RUN pip install -r requirements.txt  
  
# Запустить приложение  
CMD ["python", "app.py"]

**Четвертое задание:** Собираем Docker образ, согласно инструкциям Dockerfile

**Билдим образ:**

* Используйте команду `docker build` для создания образа. Например:

docker build -t my-python-app .

В команде указаны следующие опции:

* + -t my-python-app: Задать тег для образа. Можно называть его как удобно, далее мы сможем взаимодействовать с образом по этому имени или ID.
  + «.»: Указать текущую директорию как контекст сборки. Условно, если ваше приложение лежит в папке /Users/katant/project, а все файлы проекта лежат в папке /Users/katant/project/app, то контект должен быть указан вместо точки «app».

**Проверка образа и запуск контейнера:**

* Используйте команду из прошлой лабораторной работы для просмотра созданного образа:

docker images

* Запуск контейнера производится снова же командой docker run:

docker run -d -p 5000:5000 my-python-app

Все опции вам уже известны, здесь мы запускаем контейнер в фоновом режиме на 5000 порту.

**Проверка работы контейнера**

* Откройте браузер и перейдите по адресу `http://localhost:5000`. Вы должны увидеть сообщение "Hello, Docker!".

**Дополнительное задание при сдаче работы:**

* Не меняя приложение Python запустить во вне приложение на 5001 порту
* Изучить флаг -e при использовании docker run, чтобы указать переменные окружения и заменить GREETING со стандартного значения на своё, не трогая исходный Docker-образ.
* Запустить контейнер не в фоновом режиме, чтобы увидеть логи при запуске приложения.
* Найдите флаг, который ограничит использование памяти контейнером и поставьте ограничение 128 мегабайт.