# Введение в Docker

*Docker* – это программное обеспечение для автоматизации развёртывания и управления приложениями со всем их окружением в среде виртуализации уровня операционной системы (т.н. контейнере). Таким образом система позволяет *docker* «упаковать» приложение со всем его окружением и зависимостями в контейнер, а также предоставляет среду по управлению контейнерами.

Команды платформы docker обладают своим синтаксисом, в котором за названием следуют разные опции и аргументы:

*docker <option> <command> <arguments>,*

например, команда,

*docker run hello-world.*

запускает тестовый контейнер *hello-world*.

Для запуска контейнеров *docker* есть две похожие команды: команда *run* — взять образ *X* и создать контейнер *Z* с процессом *A*. Команда *exec* — взять контейнер *Z* и запустить в нем процесс *B*, при этом процесс *A* будет работать, как и прежде.

Контейнеры *docker* могут быть частными или публичными. Некоторые разработчики программного обеспечения выпускают официальные контейнеры для своих продуктов. Существуют публичные и приватные хранилища образов *docker*. Они называются *docker registry*. Самый популярный из них *Docker Hub*. Каждый пользователь может вносить свои образы в данный реестр. Многие дистрибутивы *Linux*, системы управления базами данных и приложения создают собственные образы в *Docker Hub*. Скачать образ из репозитория можно командой:

*docker pull <image_name>*.

Загрузить образ в репозиторий:

*docker push example.com:5000/my_image.*

Запустить свой *docker registry* можно, например, командой:

*docker run -d -p 5000:5000 --restart=always --name registry registry:2.*

Следует отметить, что в терминологии *docker* загружаются образы контейнеров, т.н. *images*. Образы контейнеров — это замороженные неизменяемые снимки контейнеров. Сам контейнер — это запущенные (или остановленные) экземпляры некоторого образа.

**Задание 1.**

1. Установите docker, используя официальную документацию:

   *https://docs.docker.com/engine/install/.*

2. Добавьте вашего пользователя в группу docker, чтобы не вводить sudo каждый раз.

   *sudo usermod -aG docker <имя пользователя>.*

3. Необходимо убедиться, что запуск демона прошел гладко и никаких предупреждений об ошибках не обнаружено:

   *sudo systemctl status docker.*

4. Инструкцию по доступным командам и их описанию можно запустить через короткую команду:

   *docker.*

5. Проверьте, может ли пользователь доставать и вносить образы в реестр командой:

   *docker run hello-world.*

6. Посмотрите какие образы загружены командой:

   *docker images.*

7. Посмотрите список контейнеров:

   *docker ps,*

ключ *"-a"* покажет в том числе остановленные контейнеры, а ключ *"-s"* —размер контейнера. т.е. сколько фактически, сейчас, в рантайме, этот контейнер занимает места на диске.

8. Посмотрите подробную информацию о контейнере командой:

   *docker inspect <friendly-name|container-id>.*

9. Удалить образ можно командой

   *docker image rm <image_name>.*

**Задание 2.**

1. Найдите образ *Debian stretch* на *Docker Hub* и загрузите его:

   *docker search debian:stretch;*

   *docker pull debian:stretch.*

**Примечание:** для каждого образа могут быть различные версии – они задаются метками через знак: например, последний образ (последняя версия) может быть загружена при помощи запроса с меткой: *lastes,* заданной через знак «:». В данном случае загружена *stretch* версия *debian.*

2. Запустите bash консоль контейнера при помощи запроса вида:

   *docker run -ti <образ>:<метка> bin/bash.*

3. Проверьте под каким пользователем вы зашли, проверьте какие пользователи зарегистрированы в системе образа.
4. Установите в системе утилиту проверки процессов *ps:*

   *apt-get update && apt-get -y install procps.*
5. Проверьте что утилита *ps* работает.
6. Зайдите во второй терминал и проверьте запущенные *docker* контейнеры (*docker ps*); проверьте все *docker* контейнеры на вашем пк (*ps* с ключом *"-a"*).

   Примечание, также заущенные контейнеры могут быть проверены командой

   *docker container ls.*
7. Проверьте логи запущенного контейнера из вторго терминала комадной

   *docker logs <container_ID>.*
8. В первом терминале выйдите из *docker* (*exit* или *ctrl+D*) и зайдите в него снова.
9. Проверьте во втором терминале какие контейнеры на вашем пк, отметьте изменения.
10. В открытом *docker* контейнере попробуйте запустить процесс *ps,* прокомментируйте результат.
11. Скопируйте *container id* того контейнера в котором была установлена утилита *ps,* запустите его командой

    *docker start <docker_id>.*
12. Проверьте запущенные контейнеры.
13. Зайдите в *bash* консоль запущенного контейнера, командой *exec*

    *docker exec -ti <docker_id> bin/bash.*
14. Проверьте запущенные процессы, прокомментируйте результаты. Также посмотрите все запущенны процессы (*ps aux*) сколько консолей запущены? Прокомментируйте результат.
15. Добавьте коммит (изменения) в новый контейнер командой

    *docker commit [CONTAINER_ID] [new_image_name]*
16. Проверьте образы.
17. Удалите все неиспользуемые контейнеры и вспомогательные данные командой

    *docker system prune*

    и образы

    *docker image prune*

    Примечание использование ключа *"-a"* к команде *prune* позволяет удалить все образы и сопутствующую информацию.

**Задание 3**

1. Создайте свой *docker* образ на основе официального образа *ubuntu* последней версии.
2. Создайте директорию нового образа в домашней директории.
3. В новой директории создайте файл с названием *Dockerfile.*

   *Dockerfile* файл - это файл с набором инструкций, которые будут совершены в чистом контейнере указанного вами образа, а на выходе вы получите свой готовый образ. Таким образом, *Dockerfile* позволяет сконфигурировать систему на языке разметки. Основными полями в *Dockerfile* являются следующие:

   *FROM <SOURCE>*— какой образ взять за основу.

   *MAINTAINER <NAME>* — автор данной разработки.

   *RUN <COMMANDS>* — команды (набор команд на языке bash, записанных в строчку или через, переносы «\» и команда «&&») исполняемые внутри контейнера, на этапе сборки образа.

   CMD <CMD>— команда, которая будет выполняться при запуске контейнера.
4. Заполните *Dockerfile* так, чтобы при запуске в нем запускалась программа *cowsay*:

   *FROM ubuntu:latest*

   *RUN apt-get update && apt-install -y cowsay*

   *CMD [“/usr/games/cowsay”, “DOK”]*
5. Запустите сборку нового контейнера. Для этого надо использовать команду

   *docker build -t <USER_NAME>/<PROJECT>:<VERSION> .*

   обратите внимание на точку в кноце строки.
6. Проверьте список образов.
7. Запустите образ с ключем *“-ti”* – что вы получите.

**Задание 4**

1. Создайте новую директорию.
2. В директории создайте *Dockerfile* с источником (*FROM*) *python* версия *2*.
3. Также добавьте скрипт:

   *WORKDIR /code*

   *COPY requirements.txt /code/*

   *RUN pip install -r requirements.txt*

   *COPY . /code/*
4. Прокомментируйте, что данный скрипт делает.
5. Создайте файл *requirements.txt* в котором укажите необходимость установки *numpy* и *matplotlib.*
6. Создайте докер образ из директории с *Dockerfile* и запустите его.

7. Проверьте, что нужные библиотеки, указанные в файле *requirements.txt* присутствуют, например при помощи команды import numpy
8. Удалите образ.

**Задание 5**

1. Создайте новую директорию.
2. в директории создайте файл *index.html* со следующим описанием:

```
<html>
<h1>Hello World!</h1>
  <p>This is a simple static website!</p>
  <p>Visit us @ <a href="https://urfu.ru">Ural Federal University</a></p>
</html>
```

3. В директории создайте *Dockerfile*, в котором укажите в качестве источника *nginx:alpine* и укажите требование скопировать все из текущей директории в следующую директорию контейнера *usr/share/nginx/html*.
4. Соберите образ.
5. Запустите образ в фоновом режиме (ключ *"-d"*) и с поротом «наружу» 8080 с прослушкой 80 порта (ключ*"-p" 8080:80).*
6. Запустите в браузере нужный порт (*localhost:8080*), убедитесь, что контейнер работает правильно.

**Задание 6**

1. Установите *docker-compose* используя, например, следующий скрипт:

```
sudo yum install epel-release
sudo yum install -y python3-pip
sudo yum upgrade python*
sudo pip3 install -U docker-compose
docker-compose version
```

Утилита *docker-compose* – представляет собой утилиты сборки много контейнерных приложений.

2. Создайте папку нового проекта. В папке должен быть файл *docker-compose.yml*, папки *server* и *client*.
3. Создайте в папке *server* файлы *Dockerfile, server.py* и *index.html.*
4. Создайте в папке *client* файлы *Dockerfile, client.py.*
5. Добавьте *в server.py* следующий код:

```
#!/usr/bin/env python3
import http.server
```

*import socketserver*

*handler = http.server.SimpleHTTPRequestHandler*

*with socketserver.TCPServer(("", 1234), handler) as httpd:*

*httpd.serve_forever()*

6. Добавьте в *client*.py следующий код:

*#!/usr/bin/env python3*

*import urllib.request*

*fp = urllib.request.urlopen("http://localhost:1234/")*

*encodedContent = fp.read() decodedContent = encodedContent.decode("utf8")*

*print(decodedContent)*

*fp.close()*

7. Добавьте в *server/Dockerfile* следующий код:

*FROM python:latest*

*ADD server.py /server/*

*ADD index.html /server/*

*WORKDIR /server/*

8. Добавьте в *client/Dockerfile* следующий код:

*FROM python:latest*

*ADD client.py /client/*

*WORKDIR /client/*

9. Добавьте в *server/index.html* приветствие, например:

*Hello-world!*

10. Добавьте в *docker-compose* следующий код:

*version: "3"*

*services:*

*server*

*build: server/*

*command: python ./server.py*

*ports:*

*- 1234:1234*

*client:*

*build: client/*

*command: python ./client.py*

*network_mode: host*

*depends_on:*

`- server`

11. Соберите проект *docker-compose* при помощи команды *build*:
    *docker-compose build*
12. запустите *docker-compose* при помощи команды *up.*
13. Проверьте, что *docker-compose* в браузере.
14. Проверьте запущенные контейнеры *docker*.
15. Проверьте запущенные контейнеры *docker-compose* (в режиме супер-пользовтаеля), сравните резлуьтаты.
16. Проверьте образы *docker-compose* (в режиме супер-пользователя).
17. Поменяйте порт выхода *docker*-контейнера сервера на *8080.*Убедитесь, что изменения прошли успешно.
18. Прокомментируйте *Dockerfile* для клиентской и серверной частей.
19. Для осатновки *docker-compose* используйте команду *down.*
20. Проверьте образы *docker-compose* (в режиме супер-пользователя) после выходы из процесса.
21.