Лабораторная работа №7.1.

Установка системы Docker на операционную систему Windows 10

Цель работы: ознакомиться с особенностями установки и использования системы виртуали- зации на основе технологии Docker при использовании операционной системы Windows.

Методические указания.

Система Docker разработана для операционной системы (ОС) Linux, поэтому при использо- вании виртуализации Docker на Windows необходимо использовать виртуализацию. Если исполь- зуется ОС Windows 10, то для виртуализации можно использовать Hyper-V. Если используется ОС Windows 8 или ниже, то для виртуализации надо использовать Oracle Virtual Box или VMware Workstation. Обычно используют Virtual Box, т. к. это бесплатная программа. В данной лаборатор- ной работе рассмотрим использование Docker в Windows 10 с применением Hyper-V.

По результатам выполнения лабораторной работы надо сделать отчёт, который должен со- держать скриншоты, описывающие процедуру выполнения лабораторной работы. Также в отчёте должны быть приведены используемые команды и результат их выполнения.

Указание по выполнению лабораторной работы.

1. Установите Hyper-V на Windows 10. Инструкция по установке была приведена в лаборатор- ной работе №6. Учтите, что после установки Hyper-V скорее всего перестанет работать VMware Workstation. После удаления Hyper-V гипервизор VMware Workstation снова ста- нет работоспособным.
2. Скачайте установщик Docker по ссылке <https://docs.docker.com/docker-for-windows/install> или с сайта курса.
3. Установите Docker для Windows.
4. Запустите приложение. В системном трее должен появится значок c изображением кита. Щелкнув по нему можно управлять различными настройками Docker или посмотреть ин- формацию об установленной версии.
5. Проверим работоспособность программы Docker. Для этого в командной строке надо вве- сти следующую команду.

docker –version

Изображение выглядит как текст

Автоматически созданное описание

1. Запустим контейнер с Linux Ubuntu и выведем стандартную фразу «Hello World!» Контей- нер, который мы будем запускать содержит все необходимые файлы для запуска на выпол- нения команды вывода на экран фразы «Hello World!»
   1. Команда запуска на выполнение контейнера

docker run hello-world

где run – команда запуска контейнера на выполнение; hello-world – имя контейнера.

Изображение выглядит как текст

Автоматически созданное описание

* 1. Приведите в отчёте результат выполнения этой команды. Объясните по-русски, какие опе- рации были сделаны при выполнении этой команды.

1. Наш клиент докера соединился с daemon, который управляет сервером
2. Daemon «скачал» образ hello-world из docker-hub
3. Daemon создал новый контейнер из этого образа и выдал результат, который мы видим на экране
4. Daemon оттранслировал результат выполнения в наш терминал
   1. Справку по этой и другим командам docker можно получить, используя ключ --help, например

docker run –help

Изображение выглядит как текст

Автоматически созданное описание

docker container run –help

Изображение выглядит как текст

Автоматически созданное описание

1. Запустим контейнер с ОС Ubuntu. Этот контейнер представляет работоспособный вариант ОС без графической подсистемы.
   1. Задайте команду

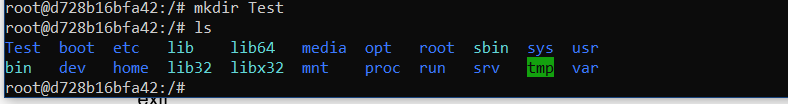
docker container run -it ubuntu bash

Изображение выглядит как текст

Автоматически созданное описание

где ubuntu – имя контейнера. -i (--interactive) – ключ оставляет открытым стандартный поток ввода. Таким образом можно передавать команды в контейнер. -t (--tty) – запуск эмулятора терми- нала, чтобы обеспечить возможность управления контейнером. bash – имя командного интерпре- татора.

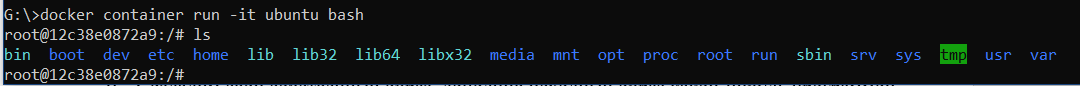
* 1. Проверьте работоспособность ОС Ubuntu какими-нибудь командами. 7.3.Создайте папку Test.



* 1. Выйдите из контейнера, используя команду

exit

* 1. Запустите контейнер ubuntu ещё раз и проверьте, сохранилась ли папка Test? Объясните результат.



Папки Test нет, так как контейнер это индивидуальный объект, который независим от образа, при изменении в контейнере не произойдет изменений в образе

* 1. Выйдите из контейнера.

1. Посмотрим существующие в системе образы. Для этого используется команда

docker image ls

Изображение выглядит как текст

Автоматически созданное описание

1. Создадим свой собственный образ, дополнив исходный образ какой-нибудь программой. Для примера установим приложение figlet в контейнере ubuntu. figlet – это программа, ко- торая выводит на экран какой-нибудь текст крупными буквами с использованием псевдо- графики. Сам текст передаётся в качестве параметра к программе.
   1. Запустите контейнер ubuntu.
   2. Установим приложение figlet в этом контейнере

apt-get update

(Получает список обновлений пакетов)

apt-get install -y figlet (установка пакета figlet (-y – без доп. вопросов))

Объясните, что делают вышеприведённые команды.

* 1. Напишем Hello Docker! при помощи псевдографики

figlet "Hello Docker!"

Изображение выглядит как текст

Автоматически созданное описание

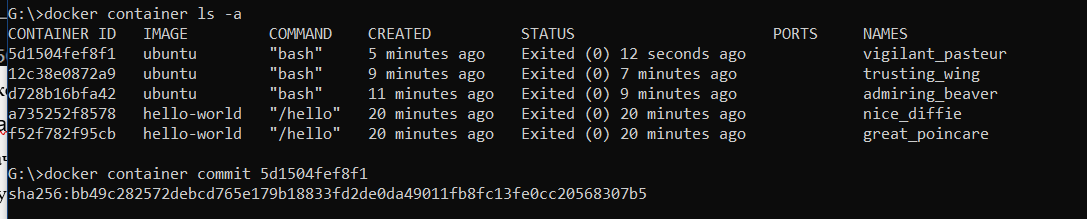
* 1. Выйдем из контейнера и выполним команду просмотра контейнеров

docker container ls -a

ключ -a означает, что должны быть выведены все контейнеры: открытые и закрытые.

* 1. Нас интересует контейнер, который был закрыт недавно. В этом контейнере надо сохра- нить произведённые нами изменения

docker container commit <CONTAINER\_ID>



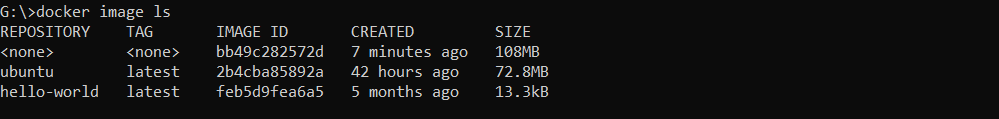
где container commit – команда сохранения контейнера в виде образа; CONTAINER\_ID – номер контейнера. Весь номер контейнера вводить не надо, достаточно ввести первые 3 цифры.

После введения команды контейнер сохранится в виде образа. Поясните, что означает ин- формация, появившаяся после ввода команды? -уникальный хэш-идентификатор коммита

* 1. Посмотрим список образов

docker image ls

Появился образ без имени.



* 1. Дадим этому образу имя myfiglet docker image tag <IMAGE\_ID> myfiglet

где image tag – команда задания имени образа. IMAGE\_ID – номер образа. Можно вводить только первые цифры. myfiglet – имя образа.

* 1. Посмотрите список образов. Обратите внимание, что новый образ имеет больший размер по сравнению с исходным.

Изображение выглядит как текст

Автоматически созданное описание

* 1. Запустим контейнер, основанный на нашем образе сразу с нужными параметрами

docker container run myfiglet figlet ‘Good bye my love’

Таким способом можно создавать новый образ, содержащий требуемую программу, на ос- нове существующего образа.

Изображение выглядит как текст

Автоматически созданное описание

1. Создадим образ, используя дополнительный файл. Напишем программу на JavaScript. Для преобразования в исполняемый код используем платформу **Node.js**.
   1. Создадим папку **dockerapp**. В этой папке создадим текстовый файл **index.js** следующего содержания:

var os = require("os");

var hostname = os.hostname();

console.log("hello from " + hostname)

Данный код извлекает hostname и отображает его на экран с приветственным сообщением.

* 1. В качестве базовой OS будем использовать образ alpine. Это легковесная версия Linux, нетребовательная к ресурсам. Alpine Linux в основном используется во встраиваемых систе- мах, также хорошо подходит для создания Docker-контейнеров. По умолчанию, Alpine Linux во время запуска полностью загружается в оперативную память.

В полученный контейнер установим Node.js, а также скопируем туда исходный код нашего приложения в каталог /app. Кроме того, укажем команду, которая должна выполняться при запуске контейнера.

* 1. В папке создадим также файл **hello.v0.1.txt** со следующим содержимым

FROM alpine — задаёт базовый (родительский) образ.

RUN apk update && apk add nodejs — показывает обновления пакетов и устанавливает nodejs.

COPY **.** /app -нужно взять файлы и папки из локального контекста и добавить их в текущую рабочую директорию образа. Если целевая директория не существует, эта инструкция её создаст.

WORKDIR /app -определение рабочей директории

CMD ["node","index.js"] -описывает команду с аргументами, которую нужно выполнить когда контейнер будет запущен

Объясните, что делает вышеприведённая последовательность команд?

* 1. В результате в папке **dockerapp** должно находиться 2 файла **index.js** и **hello.v0.1.txt**. Созда- дим новый образ. Следующую команду надо выполнять в каталоге **dockerapp**. В качестве ар- гумента указываем текущий каталог «.», в качестве докер файла флагом -f указываем создан- ный докер-файл **hello.v0.1.txt**.

Docker build -f <FILE\_NAME> **.**

Изображение выглядит как текст

Автоматически созданное описание

* 1. Назовём созданный образ hello и присвоим ему тег v0.1. docker image tag <IMAGE\_ID> hello:v0.1
  2. Убедитесь, что образ получил новое имя. Обратите внимание на размер созданного образа

hello и сравните с размером образа myfiglet.

Изображение выглядит как текст

Автоматически созданное описание

* 1. Запустим контейнер на основе нашего образа

docker container run hello:v0.1

Изображение выглядит как текст

Автоматически созданное описание

1. Создайте свой, более сложный, вариант программы, написанной на JavaScript или Python, и упакуйте её в контейнер. Результат покажите преподавателю.