

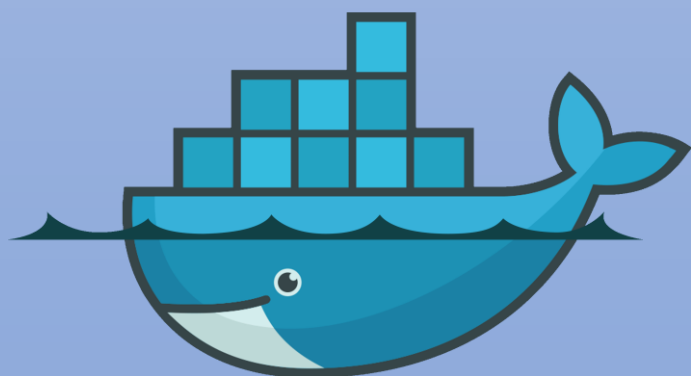
“The last thing one knows in constructing a work is what to put first”. -- Blaise Pascal

«Лишь в конце работы мы обычно узнаем, с чего ее нужно было начать». – Блез Паскаль

Установка и знакомство с технологией Docker

Лабораторная работа #1

Сергей Станкевич



ЛАБОРАТОРНАЯ РАБОТА #1

Установка и знакомство с технологией Docker

Цель работы

Ознакомиться с платформой Docker. Установить Docker Desktop на персональный компьютер и освоить принципы работы с Docker через терминал (интерфейс командной строки). Изучить основные команды при работе с Docker.

Норма времени выполнения: 2 академических часа.

Оценка работы: 3 зачетных единицы + 1 бонус.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ.

Платформа Dockerfile

Docker – один из самых известных инструментов по работе с *контейнерами*.

Контейнеры – хорошая альтернатива аппаратной виртуализации. Они позволяют запускать приложения в изолированном окружении, но при этом потребляют намного меньше ресурсов.

Несколько слов о технологии *контейнеризации*. **Контейнеризация** – это способ стандартизации развертки приложения и отделения его от общей инфраструктуры. Экземпляр приложения запускается в изолированной среде, не влияющей на основную операционную систему.

Разработчикам не нужно задумываться, в каком окружении будет работать их приложение, будут ли там нужные настройки и зависимости. Они просто создают приложение, упаковывают все зависимости и настройки в некоторый единый образ. Затем этот образ можно запускать на других системах, не беспокоясь, что приложение не запустится.

Docker – это *платформа* для разработки, доставки и запуска контейнерных приложений. Docker позволяет создавать контейнеры, автоматизировать их запуск и развертывание, управляет жизненным циклом. Он позволяет запускать множество контейнеров на одной хост-машине.

Установка дистрибутива Docker на персональный компьютер

На сегодняшний момент разработчики технологии Docker предоставляют готовое решение проблемы установки дистрибутива на персональный компьютер. Все что нужно для установки представлено на сайте www.docker.com. Здесь в разделе [Developers - Docker](#) получите возможность начать работу с Docker.

Docker Desktop – это приложение для компьютеров *MacOS*, *Windows* и *Linux*, предназначенное для создания и совместного использования контейнерных приложений.

Docker Hub – ведущий в мире сервис для поиска образов контейнеров и обмена ими с вашей командой и сообществом Docker. Для тех, кто экспериментирует с Docker, это отправная точка для изучения контейнеров Docker.

Выбор необходимого *пакета установки* зависит от среды, в которой Вы работаете, то есть от *операционной системы*, которая установлена на Вашем компьютере.

В настоящий момент три лидера операционных систем для настольных компьютеров, это: *Windows*, *MacOS* и *дистрибутив Linux*.

При установке платформы Docker на OS *Windows* или *MacOS* *потребуется дополнительная виртуализация*.

Операционные системы семейства *Linux* для Докера являются родной средой. Здесь не нужна дополнительная *виртуализация*. Все будет работать быстро и прекрасно.

Виртуализация

Виртуализация – это предоставление набора вычислительных ресурсов или их логического объединения, абстрагированное от аппаратной реализации, и обеспечивающее при этом логическую изоляцию друг от друга вычислительных процессов, выполняемых на одном физическом ресурсе.

Примером виртуализации является возможность запуска и одновременной работы нескольких операционных систем на одном компьютере.

При том каждый из экземпляров таких гостевых операционных систем работает со своим набором логических ресурсов (процессорных, оперативной памяти, устройств хранения), предоставлением которых из общего пула, доступного на уровне оборудования, управляет *хостовая* операционная система – *гипервизор*.

Иными словами, виртуализация, это сокрытие конкретной реализации за универсальным стандартизованным методом обращения к ресурсам. Или еще проще, это создание абстракции над аппаратным обеспечением.

Основные виды виртуализации

- Аппаратная виртуализация.
- Виртуализация рабочих столов.
- Виртуализация на уровне ОС (контейнеризация).

Аппаратная виртуализация позволяет создавать независимые и изолированные друг от друга виртуальные компьютеры с помощью программной имитации ресурсов (процессора, памяти, сети, диска и др.) физического сервера.

Физический сервер называют *хостовой машиной* (хостом).

Виртуальные компьютеры – *виртуальными машинами*, ВМ (иногда называют гостями).

Программное обеспечение, которое создает виртуальные машины и управляет ими, называют *гипервизором* (а также виртуальным монитором или контрольной программой – например, CP-40).

Виртуализация на уровне операционной системы, контейнеризация, позволяет запускать программное обеспечение в изолированных на уровне операционной системы пространствах.

Наиболее распространенной формой виртуализации на уровне ОС являются *контейнеры* (например, Docker).

Контейнеры более легковесны, чем виртуальные машины, так как они опираются на функционал ядра ОС и им не требуется взаимодействовать с аппаратным обеспечением.

Контейнеры представляют из себя изолированную среду для запуска любого приложения со всеми его зависимостями и настройками.

Платформа WSL

Для тех, кто работает с Docker в ОС Windows, а также кто хочет расширить свои познания в Linux, я рекомендую установить и освоить специальную платформу WSL (Windows Subsystem for Linux).

WSL позволяет разработчикам запускать среду GNU/Linux с большинством программ командной строки, служебных программ и приложений непосред-

ственно в Windows без каких-либо изменений и необходимости использовать традиционную виртуальную машину или двойную загрузку.

В рамках сотрудничества компаний Майкрософт и Canonical стало возможным использовать оригинальный образ ОС Ubuntu 14.04 для непосредственного запуска поверх WSL множества инструментов и утилит из этой ОС без какой-либо виртуализации.

WSL предоставляет интерфейсы, во многом совместимые с интерфейсами ядра Linux. Однако подсистема WSL полностью разработана корпорацией Майкрософт и *не содержит* в себе каких-либо *исходных кодов ядра Linux*.

WSL запускает многие немодифицированные приложения, работающие в пространстве пользователя, в частности, оболочку bash, утилиты sed, awk, интерпретаторы языков программирования Ruby, Python, и т. д.

Существуют две версии WSL, версии 1 и 2. Версия WSL2 предназначена для Windows 10 с 2017 года и моложе. Версия WSL1 используется в старых версиях ОС Windows.

WSL 2 использует последнюю и самую новую технологию виртуализации для запуска ядра Linux внутри упрощенной служебной виртуальной машины. Однако WSL 2 не является традиционным функционалом виртуальной машины.

Основные причины, чтобы обновить WSL 1 до WSL 2:

- повышение производительности файловой системы;
- поддержка полной совместимости системных вызовов.

Получение справочной информации по технологии Docker

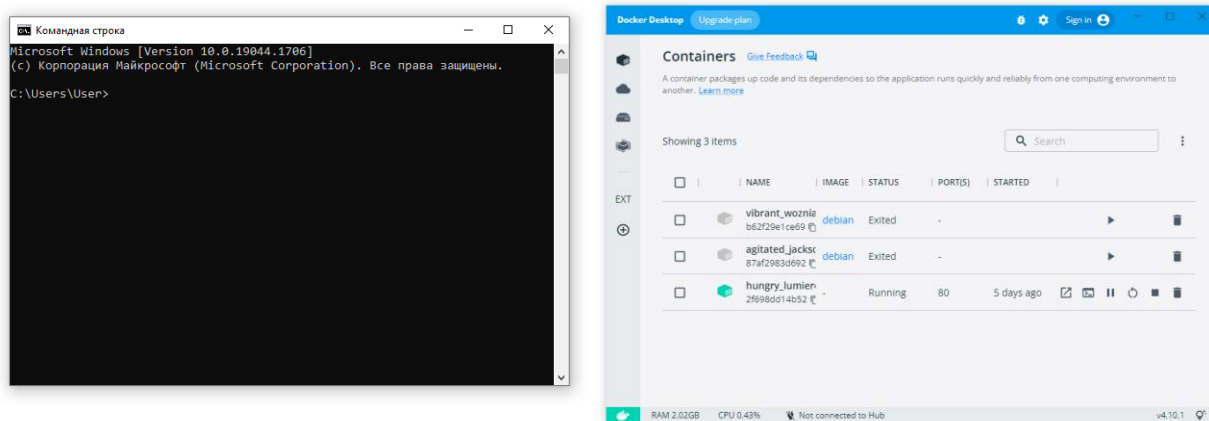
Вся или почти вся необходимая информация представлена на сайте www.docker.com. Однако перейдя по контекстному меню [Developers/Docs](#) вы попадете на другой сайт www.docs.docker.com. Основная информация здесь представлена в шести разделах. Для получения информации об установке дистрибутива загляните в раздел **“Get started”**. Для получения нужного вам дистрибутива используйте раздел **“Download and install”**. Хотя если вы воспользуетесь этими двумя разделами вас все равно выбросит в раздел **“Guides”**.

Установив дистрибутив, вы начнете углубляться в изучение Docker, и здесь вам поможет раздел **“Reference”**. В этом разделе представлена достаточная информация для освоения Docker. Здесь вы будете частым, и скорее, постоянным гостем.

Информацию непосредственную об установленной на Вашем компьютере платформы Docker, вы можете получить с помощью интерфейса командной строки (Command Line Interface, CLI).

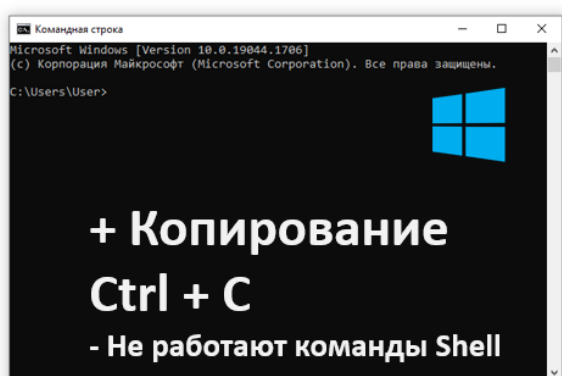
Инструменты для дальнейшей работы

Для дальнейшей работы нам нужны будут два средства, терминал командной строки (cmd) и Docker Desktop.

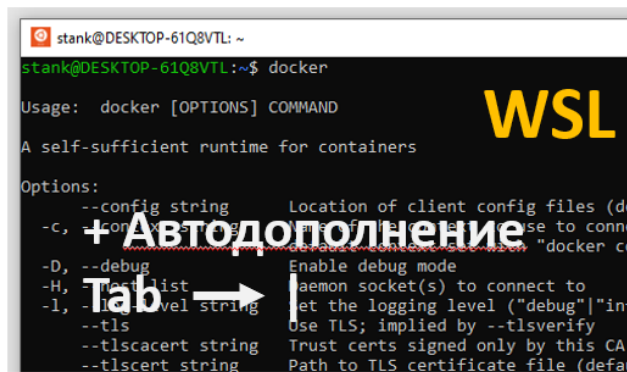


Однако терминалы сильно отличаются в зависимости от среды, в которой они работают. Можно определить два типа терминалов: Командный терминал ОС Windows (cmd); и терминал командной оболочки Shell операционных систем семейства UNIX/Linux (как правило в таких дистрибутивах предоставлен командный интерпретатор **bash**).

Командный терминал Windows (cmd)



Командная оболочка Shell (bash - предустановлен)



В терминалах командной строки Windows хорошо работает технология «сорупаст» (горячие клавиши Ctrl + C, Ctrl + V). Однако в этой среде не работают многие команды оболочки Shell (UNIX). Это представляет большие неудобства при работе с Docker.

Консольные терминалы среды UNIX/Linux поддерживают все команды оболочки Shell, а также удобное средство «Автодополнение» команд (клавиша **Tab**). Оболочка Shell, это очень удобной средство для работы с Docker. Одна-

ко эти терминалы плохо поддерживают технологию «сорупаст» при работе в среде Windows.

Предлагаю крутой «лайфхак» для запуска текстового редактора **notepad.exe** из среды WSL. Это значительно упростит вам работу.

```
stank@DESKTOP-61Q8VTL:~/cowsay$ notepad.exe Dockerfile
```

Здесь мы в среде WSL открыли **Dockerfile** в текстовом редакторе **notepad.exe**. можно свободно использовать технологию «сорупаст».

Текстовые редакторы

Немного расскажу о текстовых редакторах, которые мы будем использовать в дальнейшей работе. Это:

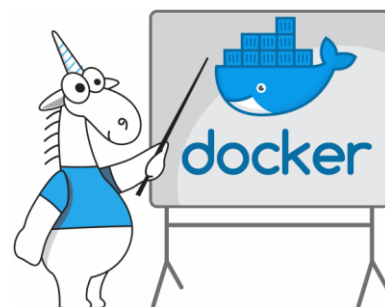
- nano
- vi (vim)
- notepad.txt

nano простой, не универсальный редактор, управляется клавиатурой.

vi (vim) - мощный редактор, требующий долгого изучения. Управляется клавиатурой. POSIX, стандарт программной совместимости систем Unix, требует наличия vi. Предназначен для профессиональной работы.

Редактор **notepad** очень удобен для работы в среде Windows.

Best of LUCK with it, and remember to HAVE FUN
while you're learning :)



УПРАЖНЕНИЯ

Упражнение 1

Установка дистрибутива Docker на персональный компьютер. Получение справочной информации.

В браузере перейдите по ссылке [Home – Docker](#), и ознакомьтесь с содержанием сайта. Определитесь в какой операционной системе вы будете работать, скачайте с сайта соответствующий дистрибутив платформы Docker, и установите его на свой персональный компьютер.

Получите доступ к **Docker Desktop** и следуйте *пошаговым инструкциям*, чтобы создать свое первое контейнерное приложение за считанные минуты.

После установки Docker **Desktop** вы можете сразу начать работу с Docker. Изучите графический интерфейс установленного приложения Docker Desktop.

Запустите терминал командной строки (CLI) вашей операционной системы.

Чтобы убедиться в правильности установки и работоспособности программной среды Docker, выполните команду **docker version**.

```
$ docker version
```

Получите общую информацию о командах Docker и их опциях:

```
$ docker
```

В UNIX-подобных системах попробуйте сочетание:

```
$ docker
```

 + двойное нажатие клавиши TAB.

Семантика выполнения любой *docker-команды* имеет следующий вид:

```
docker [OPTIONS] COMMAND
```

Получить справку о какой ни будь *docker-команды* можно с помощью следующего синтаксиса:

```
docker COMMAND --help
```

например, изучим наиболее распространенную команду **run**.

```
$ docker run --help
```

Получим информацию о состоянии Docker:

```
$ docker info
```

Полученную информацию подтвердите скриншотами в отчете.

Упражнение 2

Первые шаги Docker. Запуск первого образа. Основные команды.

Для выполнения упражнения необходимо подключение к интернету.

Проверим правильность установки программной системы Docker выполнив команду:

```
$ docker run debian echo "My first Docr"
```

Подождем, и получим примерно такой результат:

```
Unable to find image 'debian' locally
debian:latest: The image you are pulling has
been verified
511136ea3c5a: Pull complete
638fd9704285: Pull complete
61f7f4f722fb: Pull complete
Status: Downloaded newer image for
debian:latest
My first Docr
```

Что это значит?

The diagram shows the output of the Docker command with callouts explaining each line:

- Unable to find image 'debian' locally**: Невозможно найти образ 'debian' на локальной системе
- debian:latest: The image you are pulling has been verified**: debian:latest: Загружаемый образ проверен
- 511136ea3c5a: Pull complete**: 511136ea3c5a: Загружен полностью
- Status: Downloaded newer image for debian:latest**: Состояние: Загружен обновленный образ для debian:latest

My first Docker

Разберем команду подробнее:

\$ docker run debian echo "My first Docker"

- docker**: Инициализирует запуск контейнеров.
- run**: Аргумент, это имя образа.
- debian**: Упрощенная версия ОС Debian Linux.
- echo**: Сообщает об отсутствии локальной копии образа Debian.
- "My first Docker"**: Docker в онлайн-режиме проверяет Docker Hub и загружает самую новую версию образа Debian. Собирает компоненты ОС Debian.

Unable to find image 'debian' locally

debian:latest: The image you are pulling has been verified

511136ea3c5a: Pull complete

638fd9704285: Pull complete

61f7f4f722fb: Pull complete

Status: Downloaded newer image for debian:latest

My first Docker

Docker помещает образ в работающий контейнер и выполняет заданную команду shell: 'echo'.

Если выполнить команду еще раз, то контейнер запустится немедленно, без предварительной загрузки образа. Это займет *менее 1 секунды*. Тоже самое в обычной виртуальной машине время выполнения *от n секунд до n минут*. Вот вам доказательство преимущества контейнеризации.



Docker выполнил следующую работу:

1. подготовил и запустил контейнер;
2. выполнил команду 'echo';
3. остановил контейнер.

Основные команды Docker

Исследуем Docker, запуская контейнер и наблюдая в нем результаты выполнения различных команд и действий.

Инициализируем новый контейнер, и зададим для него имя хоста с помощью флага -h:

\$ docker run -h CONTAINER -it debian /bin/bash

```

Командная строка - docker run -h CONTAINER -it debian /bin/bash
Microsoft Windows [Version 10.0.19044.1706]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\User>docker run -h CONTAINER -it debian /bin/bash
root@CONTAINER:/#

```

Здесь обратите внимание на решетку (# – шарп). Это значит, что вы запустили в контейнере оболочку (**shell**) с правами суперпользователя.

Получите справку о флагах (опциях): **-h, -i, -t**.

Продолжим работу. Откройте новый терминал (не закрывая работу первого контейнера). Выполните команду **docker ps** прямо с *хоста*.

```

Командная строка
Microsoft Windows [Version 10.0.19044.1706]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\User>docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                    NAMES
432649c8b3ee   debian    "/bin/bash"             5 minutes ago Up 5 minutes   0.0.0.0:80->80/tcp      hopeful_matsumoto
2f698dd14b52   docker/getting-started "/docker-entrypoint..." 9 days ago    Up 9 days (Paused) 0.0.0.0:80->80/tcp      hungry_lumiere
C:\Users\User>

```

Команда ‘ps’ – показывает список запущенных контейнеров. Это позволяет узнать некоторые подробности обо всех контейнерах, работающих в текущий момент. ‘ps’ это сокращенное “process”, то есть, работающий контейнер — это отдельно работающая программа.

Обратите внимание на поле “NAMES”, это имена контейнеров. Вы сами можете назначить контейнеру любое имя с помощью флага **–name**, например:

\$ docker run --name sergeyst debian echo "Boo"

NAME	IMAGE	STATUS	PORT(S)
sergeyst 2884b491e929	debian	Exited	
hopeful_matsumoto 432649c8b3ee	debian	Running	
update_rubin ca62135de59d	debian	Exited	

Однако, если вы этого не сделали, то Docker сам сделает это за вас. Docker генерирует имена в виде случайного прилагательного, за которым следует имя известного ученого, инженера или хакера.

Больше информации о конкретном контейнере можно получить с помощью команды **'docker inspect'**. Например:

```
$ docker inspect sergeyst
```

При этом вы получите очень много информации, практически всё об одном контейнере. Нужно ли вам это. Воспользуйтесь регулярными выражениями для выделения нужной нам информации. Можно использовать утилиту **grep** или аргумент **--format** (который принимает шаблон регулярных выражений языка Go). Например так:

```
C:\Users\User>docker inspect sergeyst | grep IPAddress
"grep" не является внутренней или внешней
командой, исполняемой программой или пакетным файлом.
```

или так

```
C:\Users\User>docker inspect --format {{.NetworkSettings.IPAddress}} hopeful_matsumoto
172.17.0.3
```

Обе команды выдают IP-адрес работающего контейнера.

Все что происходит с контейнерами Docker все записывает. Получить журналы контейнера можно командой **'docker logs'**.

```
$ docker logs hungry_lumiere
```

Вы узнаете большую историю об одном контейнере.

И наконец, когда вам надоело работать с контейнерами Docker, выполните команду **'exit'**. Эта команда прекратит работу всех контейнеров, в этом легко убедиться, набрав команду **'docker ps'**. Вы не увидите ни одного работающего контейнера. Однако команда **'exit'** не является командой Docker, это команда консольного терминала. Она прекращает работу консольного терминала (закрывает процесс терминала, и все его дочерние процессы, в нашем случае это работающие контейнеры).

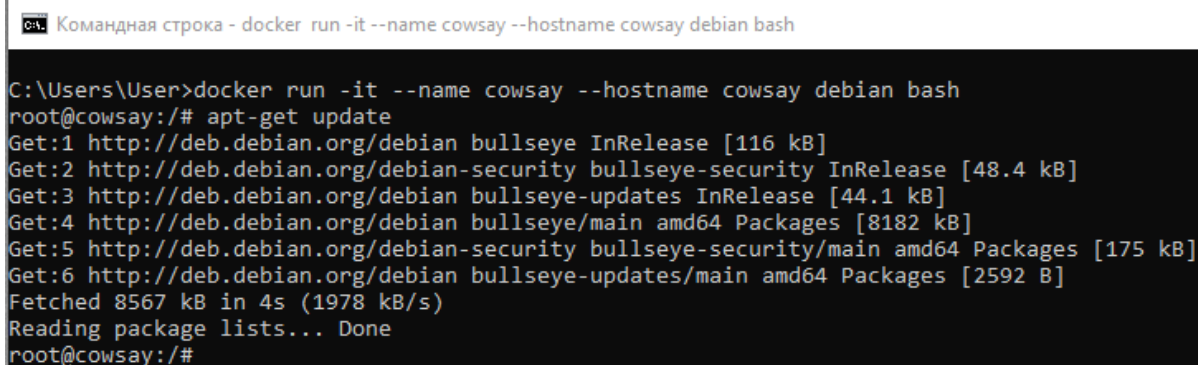
Упражнение 3

Создание приложения, работающего внутри Docker.

Создадим приложение «cowsay», работающее внутри Docker. Начнем с запуска контейнера и установки нескольких пакетов. Необходимо подключение к Интернету:

```
$ docker run -it --name cowsay --hostname cowsay debian bash
```

```
root@cowsay:/# apt-get update
```



Командная строка - docker run -it --name cowsay --hostname cowsay debian bash

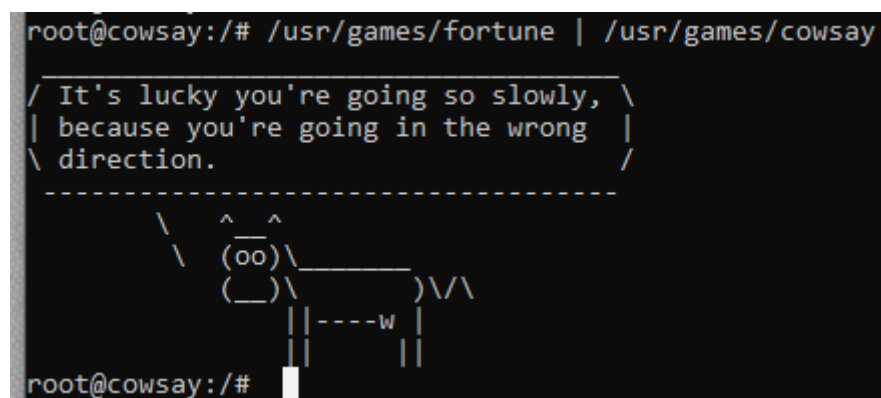
```
C:\Users\User>docker run -it --name cowsay --hostname cowsay debian bash
root@cowsay:/# apt-get update
Get:1 http://deb.debian.org/debian bullseye InRelease [116 kB]
Get:2 http://deb.debian.org/debian-security bullseye-security InRelease [48.4 kB]
Get:3 http://deb.debian.org/debian bullseye-updates InRelease [44.1 kB]
Get:4 http://deb.debian.org/debian bullseye/main amd64 Packages [8182 kB]
Get:5 http://deb.debian.org/debian-security bullseye-security/main amd64 Packages [175 kB]
Get:6 http://deb.debian.org/debian bullseye-updates/main amd64 Packages [2592 B]
Fetched 8567 kB in 4s (1978 kB/s)
Reading package lists... Done
root@cowsay:/#
```

```
root@cowsay:/# apt-get install -y cowsay fortune
```

```
...
Setting up perl (5.32.1-4+deb11u2) ...
Setting up cowsay (3.03+dfsg2-8) ...
Processing triggers for libc-bin (2.31-13+deb11u3) ...
root@cowsay:/#
```

Запускаем приложения:

```
root@cowsay:/# /usr/games/fortune | /usr/games/cowsay
```



```
root@cowsay:/# /usr/games/fortune | /usr/games/cowsay

/ It's lucky you're going so slowly, \
| because you're going in the wrong  |
\ direction.                          /
-----
      \  ^  ^
       (oo)\_____)
        (__)|       )\/\
           ||----w |
           ||     ||

root@cowsay:/#
```

Сохраним этот контейнер. Для превращения контейнера в образ нужно всего лишь выполнить команду **docker commit**. При этом не имеет значения, работает контейнер или он остановлен. Необходимо передать в команду имя кон-

тейнера («cowsay»), имя для создаваемого образа («cowsayimage»), а также имя репозитория, в котором образ будет сохранен («test»):

```
root@cowsay:/# exit
exit
$ docker commit cowsay test/cowsayimage
```

Возвращаемое командой значение представляет собой уникальный идентификатор созданного образа.

```
C:\Users\User>docker commit cowsay test/cowsayimage
sha256:ce096e47dc3a95f6a6f7850b86e773321a909e1b54ee0e520dedf40dbac2aadb
```

Теперь у нас есть образ с предустановленным приложением cowsay, который мы можем запускать в любое время:

```
C:\Users\User>docker run test/cowsayimage /usr/games/cowsay "Moooo"
```

```
C:\Users\User>docker run test/cowsayimage /usr/games/cowsay "Moooo"
< Moooo >
-----
      \
       ^ ^
      (oo)\
      ( _)\
           )\ \
           || ---w ||
           ||       ||

C:\Users\User>
```

We hope you enjoy working with Docker!



ЗАДАНИЯ

Задание 1

Определитесь со способом работы с Docker и необходимыми для этого инструментами. Выполните упражнение №1.

Создайте первый образ и проработайте следующие команды docker: run, inspect, diff, ps, logs, rm. Получите справку по этим командам.

Результат подтвердите скриншотом.

Задание 2

Выполните упражнение №2. Создайте несколько контейнеров не менее трёх, один из них назовите своим именем. Имена остальных контейнеров должны генерироваться средой Docker. Узнайте в именах каких личностей были названы эти контейнеры. Информацию о личностях и благодаря чему они стали известны укажите в отчете.

Результат подтвердите скриншотом.

Задание 3

Выполните упражнение №3. Сгенерируйте не менее 3-х высказываний коровы. Переведите их на белорусский или русский язык. Фразы на английском и перевод с него представьте в отчете.

Результат подтвердите скриншотом.

Бонусом в этом задании может быть говорящий кит «whalesay».

<https://dker.ru/docs/docker-engine/get-started-with-docker/build-your-own-image/>

«Easy things should be easy and hard things should be possible»
«Простые вещи должны быть простыми, а сложные вещи должны быть
ВОЗМОЖНЫМИ»



Контрольные вопросы:

- 1) Что такое **виртуализация**? Какие типы виртуализации Вы знаете?
- 2) Что такое **виртуальная машина**?
- 3) Что такое **контейниризация**?
- 4) Что такое командная строка, что представляет интерфейс командной строки (**CLI**)? Что такое консоль и терминал?
- 5) При установке дистрибутива **Docker** были ли у Вас проблемы с виртуализацией, и если «Да», то как Вы ее решили?
- 6) В какой операционной среде Вы установили **Docker** и почему?
- 7) Что такое платформа **Docker** и для чего она нужна?
- 8) Достаточно ли востребована технология **Docker** на рынке разработка ПО?
- 9) Насколько востребованы специалисты **Docker** знанием дд, есть ли вакансии для таких специалистов в настоящий момент на рунке труда?
- 10) Что такое Linux, и как Linux связан с **Docker**?

ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ

Литература

Моуэт Э. **Использование Docker** / пер. с англ. А.В. Снастина; науч. ред. А.А. Маркелов. – М.: ДМК Пресс, 2017. – 354 с.: ил. []

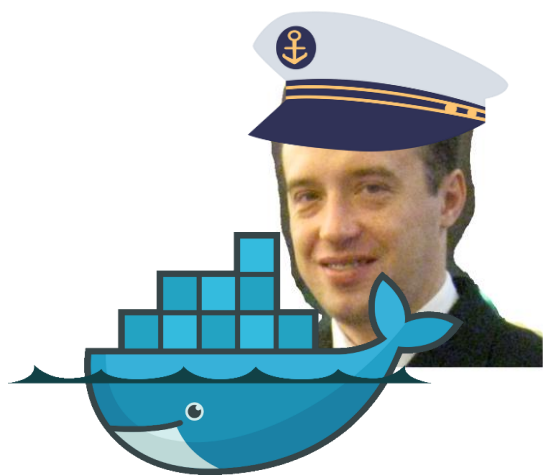
Иан Милл, Эйдан Хобсон Сейерс **Docker на практике** / пер. с англ. Д.А. Беликов. – М.: ДМК Пресс, 2020. – 516 с.: ил. []

Шоттс У. Ш80 **Командная строка Linux. Полное руководство.** — СПб.: Питер, 2017. — 480 с.: ил. — (Серия «Для профессионалов»). []

Интернет источники

[Основы виртуализации \(обзор\) / Хабр \(habr.com\)](#) Основы виртуализации (обзор). Simust 28 марта в 09:09.

[Что такое Docker: для чего он нужен и где используется - Блог компании Селектел \(selectel.ru\)](#) Что такое Docker: для чего он нужен и где используется. Главная/Блог/Технические обзоры



Счастливого плавания!