

---

**ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ТЕЛЕКОММУНИКАЦИИ**

---

УДК 004.67

**Сергеева О.О., Белозерова А.Р.**

*Димитровградский инженерно-технологический институт – филиал федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский ядерный университет «МИФИ», Россия, г. Димитровград*

**Sergeeva O.O., Belozerova A.R.**

*Dimitrovgrad Engineering and Technological Institute of the National Research Nuclear University MEPhI, Russia, Dimitrovgrad  
e-mail: alyancevich99@mail.ru, asabel2009@yandex.ru*

## **KUBERNETES C DOCKER НА ЛОКАЛЬНОЙ МАШИНЕ**

## **KUBERNETES WITH DOCKER ON A LOCAL MACHINE**

**Аннотация:** целью статьи является актуальная подборка потребительских сведений об аппаратной виртуализации на основе технологии Kubernetes с Docker на локальной машине. Статья представляет собой обзорный стриминг с разбором основных понятий из концепции аппаратной виртуализации и погружением в изучение кластеризации по способу контейнеризации веб-приложения на локальном хостинге.

**Abstract:** the purpose of this article is an up-to-date collection of consumer information about hardware virtualization based on Kubernetes technology with Docker on a local machine. This article is the overview of streaming with an analysis of the main concepts from the concept of hardware virtualization and a dive into the study of clustering by the method of containerization of a web application on local hosting.

**Ключевые слова:** аппаратная виртуализация, виртуальная машина (ВМ), контейнер, Docker, Kubernetes.

**Keywords:** hardware virtualization, virtual machine (VM), container, Docker, Kubernetes.

Актуальность выбранной темы продиктована современными тенденциями по контейнеризации приложений, аппаратной виртуализации, которая предоставляет целый ряд преимуществ, таких как масштабируемость, безопасность, изоляция использования гипервизоров для совместного использования физического оборудования с виртуальными машинами.

В данном обзоре авторы описывают практику использования Kubernetes под ОС Windows, которая имеет свои отличия от существующих, описанных в большом количестве статей и постов на сегодняшний день по Kubernetes. Поскольку статья представляет обзорный стриминг, поэтому все последующие определения даются исходя из общего представления авторов.

Виртуальная машина (ВМ) – это виртуальный компьютер со всеми виртуальными устройствами и виртуальным жёстким диском, на который и устанавливается новая независимая ОС (гостевая ОС), вместе с виртуальными драйверами устройств, управлением памятью и другими компонентами. Таким образом, происходит получение абстракции физического оборудования, позволяющей запускать на одном компьютере множество виртуальных компьютеров. Виртуальное оборудование отображается в свойствах системы, а установленные приложения взаимодействуют с ним как с настоящим. При этом сама виртуальная машина полностью изолирована от реального компьютера, хотя и может иметь доступ к его диску и периферийным устройствам.

Развитие технологий виртуализации происходят благодаря увеличению мощностей аппаратного обеспечения, как для серверных систем, так и для настольных компьютеров. Технологии виртуализации позволяют запускать на одном физическом компьютере (хосте) несколько виртуальных экземпляров операционных систем (гостевых ОС) в целях обеспечения их независимости от аппаратной платформы и сосредоточения нескольких виртуальных машин на одной физической. Виртуализация предоставляет такие преимущества, как существенная экономия на аппаратном обеспечении, обслуживании, повышение гибкости ИТ-инфраструктуры, упрощение процедуры резервного копирования и восстановления после сбоев. Виртуальные машины, являясь независимыми от конкретного оборудования единицами, могут быть запущены на любой аппаратной платформе поддерживаемой архитектуры.

Установленная ВМ может по-разному занимать место на диске компьютера:

- фиксированное место на жёстком диске, что позволяет осуществлять более быстрый доступ к виртуальному жёсткому диску и позволяет избежать фрагментации файла;

- динамическое выделение памяти. При установке дополнительных приложений память будет динамически выделяться под них, пока не достигнет максимального объема, отведенного ей [1].

При использовании ВМ появляются дополнительные расходы на эмуляцию виртуального оборудования и запуск гостевой ОС, поддержка и администрирование необходимого окружения для работы приложения. Также при разворачивании большого количества виртуальных машин на сервере, объем занимаемого ими места на жёстком диске будет только расти, в связи с тем, что для каждой ВМ требуется место, как минимум, для гостевой ОС и драйверов для виртуальных устройств.

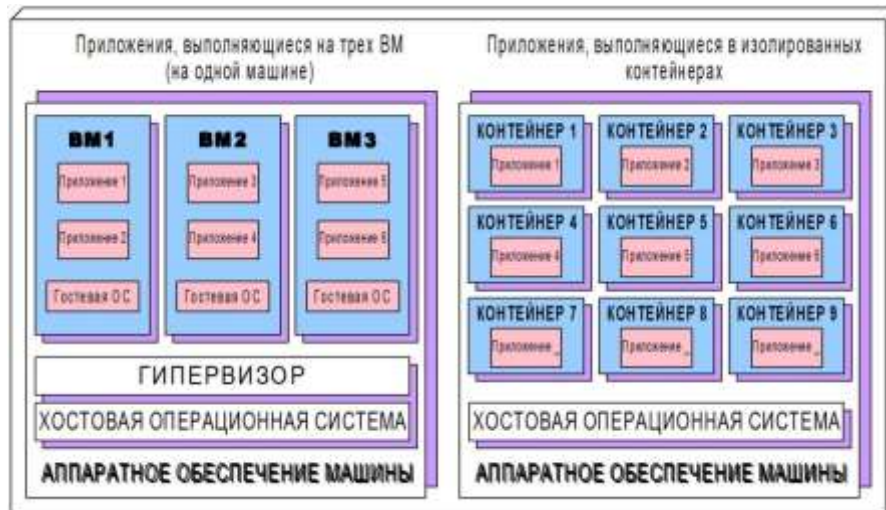
Виртуальные машины и контейнеры решают одну задачу, однако осуществляют это по-разному. Контейнеры занимают меньше места, так как переиспользуют большее количество общих ресурсов хост-системы чем ВМ, в отличие от ВМ, обеспечивает виртуализацию на уровне ОС, а не

аппаратного обеспечения. Такой подход обеспечивает меньший объем занимаемого места на жёстком диске, быстрое развертывание и более простое масштабирование.

Контейнер – это абстракция на уровне приложения, объединяющая код и зависимости. Это не облегченные виртуальные машины, а стандартизированные исполняемые пакеты, используемые для доставки приложений, включая приложения, разработанные с использованием архитектуры программного обеспечения на основе микросервиса, и включающие все компоненты, необходимые для запуска приложений.

Контейнер – это не что иное, как отдельный изолированный процесс, выполняющийся в центральной ОС, потребляющий только те ресурсы, которые приложение потребляет, без накладных расходов в виде дополнительных процессов.

Из-за накладных расходов виртуальных машин в конечном счете приходится группировать несколько приложений в каждую отдельную виртуальную машину, поскольку недостаточно ресурсов для выделения всей виртуальной машины каждому приложению. При использовании контейнеров, как показано на рисунке 1, можно иметь по одному контейнеру для каждого приложения. Конечным результатом является то, что на одном и том же аппаратном обеспечении умещать еще больше приложений [2].



**Рисунок 1 – Использование виртуальных машин для изоляции групп приложений по сравнению с изоляцией отдельных приложений с помощью контейнеров**

Когда запускается три виртуальных машины на хосте, имеется три совершенно отделенные друг от друга операционные системы, работающие на одном и том же аппаратном обеспечении. Под этими виртуальными машинами находятся хостовая ОС и гипервизор, который разделяет физические аппаратные ресурсы на меньшие наборы виртуальных ресурсов, которые могут использоваться операционной системой внутри каждой вир-

туальной машины. Приложения, запущенные на этих виртуальных машинах, выполняют системные вызовы ядра гостевой ОС в виртуальной машине, а затем ядро выполняет инструкции x86 на физическом ЦП хоста через гипервизор.

Контейнеры же выполняют системные вызовы на одном и том же ядре, работающем в хостовой ОС. Это единственное ядро, выполняющее инструкции x86 на процессоре хоста. ЦП не нужно делать какой-либо виртуализации, как он делает с виртуальными машинами, представлено на рисунках 2 и 3.



Рисунок 2 - Приложения, выполняющиеся на множестве VM



Рисунок 3 - Приложения, выполняющиеся в изолированных контейнерах

Главное преимущество виртуальных машин заключается в их полной изоляции, поскольку каждая виртуальная машина работает под управлением собственного ядра ОС Windows, в то время как все контейнеры обращаются к одному ядру, что может явно представлять угрозу безопасности. При ограниченном количестве аппаратных ресурсов виртуальные машины могут использоваться только в том случае, если требуется изолировать небольшое количество процессов.

Для выполнения большого количества изолированных процессов на одной машине контейнеры являются гораздо лучшим выбором из-за их низкой нагрузки. Однако, каждая виртуальная машина выполняет свой собственный набор системных служб, в то время как контейнеры этого не делают, потому что все они выполняются в одной ОС. Это также означает, что для запуска контейнера начальная загрузка компонентов не требуется,

как в случае с виртуальными машинами. Процесс, выполняющийся в контейнере, запускается немедленно.

На современном этапе развития, контейнерные технологии актуальны для перехода от классической инфраструктуры к виртуализированной, вне зависимости от параметров аппаратного обеспечения хоста. Контейнерные технологии стали более широко известны с появлением контейнерной платформы Docker. Docker была первой контейнерной системой, которая сделала контейнеры легко переносимыми на разные машины.

Docker – это платформа для упаковки, распространения и выполнения приложений. Она позволяет упаковывать приложение вместе со всей его средой. Это могут быть либо несколько библиотек, которые требуются приложению, либо все файлы, которые обычно доступны в файловой системе установленной операционной системы. Docker позволяет переносить этот пакет в центральный репозиторий, из которого он затем может быть перенесен на любой компьютер, на котором работает Docker, и выполнен там [4]. Этот сценарий состоит из трех главных понятий в Docker:

- образы (Images). Образ контейнера на основе Docker – это то, во что упаковывается приложение и его среда. Он содержит файловую систему, которая будет доступна приложению;

- хранилища (Registry). Хранилище Docker – это репозиторий, в котором хранятся образы Docker и который упрощает обмен этими образами между различными людьми и компьютерами. Когда создается образ, можно либо запустить его на компьютере, на котором он создан, либо отправить (закачать) образ в хранилище, а затем извлечь(скачать) его на другом компьютере и запустить его там;

- контейнеры. Контейнер на основе Docker – это обычный контейнер ОС Windows, созданный из образа контейнера на основе Docker. Выполняемый контейнер – это процесс, запущенный на хосте, на котором работает Docker, но он полностью изолирован как от хоста, так и от всех других процессов, запущенных на нем [2].

На рисунке 4 показаны все три главных понятия в Docker и их взаимосвязь.

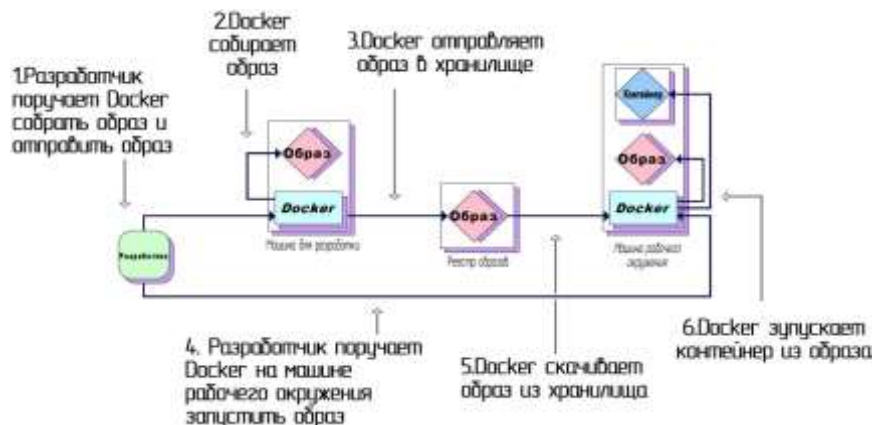


Рисунок 4 - Образы, реестры и контейнеры Docker



Kubernetes – это программная система, которая позволяет легко разворачивать контейнеризированные приложения и управлять ими. Она использует возможности контейнеров ОС Windows для запуска разнородных приложений без необходимости знать какие-либо внутренние детали этих приложений и без необходимости вручную разворачивать эти приложения на каждом хосте. Процедура разворачивания приложений через Kubernetes всегда одинаковая, независимо от того, содержит ли кластер всего несколько узлов или тысячи [2].

На аппаратном уровне кластер Kubernetes состоит из множества узлов, которые можно разделить на два типа:

- ведущий узел (мастер), на котором размещена плоскость управления (Control Plane) Kubernetes, контролирующая и управляющая всей системой Kubernetes;
- рабочие узлы, на которых выполняются разворачиваемые приложения.

На рисунке 5 показаны компоненты, работающие на этих двух наборах узлов.

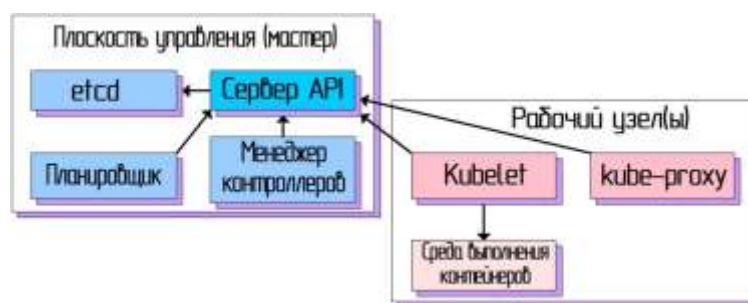


Рисунок 5 - Компоненты, составляющие кластер Kubernetes

Плоскость управления – это то, что управляет кластером и заставляет его функционировать. Она состоит из нескольких компонентов, которые могут работать на одном ведущем узле либо быть распределены по нескольким узлам и реплицированы для обеспечения высокой доступности. Эти компоненты следующие:

- сервер Kubernetes API, с которым взаимодействуете вы и другие компоненты плоскости управления;
- планировщик, который распределяет приложения (назначает рабочий узел каждому разворачиваемому компоненту приложения);
- менеджер контроллеров, выполняющий функции кластерного уровня, такие как репликация компонентов, отслеживание рабочих узлов, обработка аварийных сбоев узлов и т. д.;
- etcd, надежное распределенное хранилище данных, которое непрерывно сохраняет конфигурацию кластера.

Компоненты плоскости управления содержат и управляют состоянием кластера, но не выполняют приложения. Это делается (рабочими) узлами.

Рабочие узлы – это машины, на которых выполняются контейнеризированные приложения. Задача выполнения, мониторинга и предоставления служб приложениям выполняется следующими компонентами:

- - Docker, rkt или другая среда выполнения контейнеров, в которой выполняются контейнеры;
- - Kubelet, агент, который обменивается с сервером API и управляет контейнерами на своем узле;
- - служебный прокси Kubernetes (kube-proxy), который балансирует нагрузку сетевого трафика между компонентами приложения.

Перед установкой Docker в Windows потребуется операционная система Windows 10 версии Корпоративная, Профессиональная или Образовательная. Также должны быть подключены компоненты Hyper-V и Windows Containers. Для работы с этими компонентами система должна выполнять следующие требования:

- 64-разрядный процессор с поддержкой преобразования адресов второго уровня (SLAT от англ. «Second Level Address Translation»);
- не менее 4 ГБ оперативной памяти;
- поддержка аппаратной виртуализации на уровне BIOS.

Существует два варианта установки Docker для Windows:

1. Приложение «Docker Toolbox for Windows»
2. Приложение «Docker for Windows».

Docker Toolbox для Windows является устаревшим решением, которым можно воспользоваться, если компьютер не удовлетворяет требованиям, которые нужны для Docker для Windows.

Для того чтобы установить Docker, необходимо следовать инструкциям на странице <http://docs.docker.com/engine/installation/> для операционной системы Windows. Так как мы используем Windows 10, то при установке Docker в соответствии с инструкциями, Docker настроит для нас виртуальную машину и запустит демон Docker внутри этой виртуальной машины. Исполняемый файл клиента Docker будет доступен в хостовой ОС и будет взаимодействовать с демоном внутри виртуальной машины.

Когда установка завершится, Docker запустится автоматически. Kubernetes в области уведомлений указывает, что Docker запущен и доступен из терминала, рисунок 6.

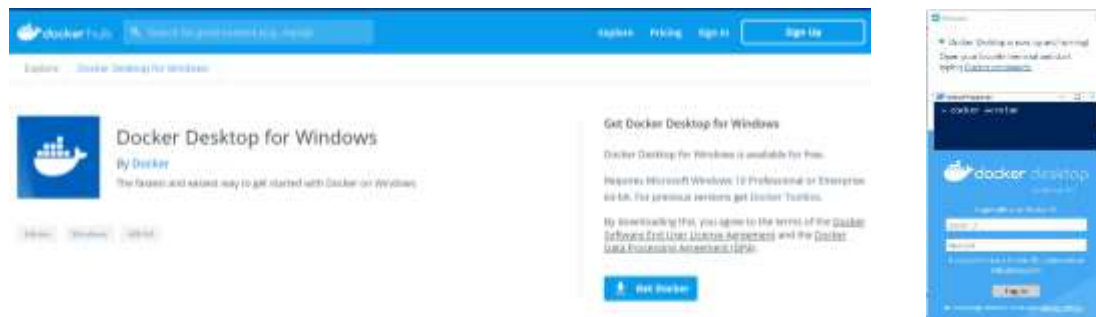
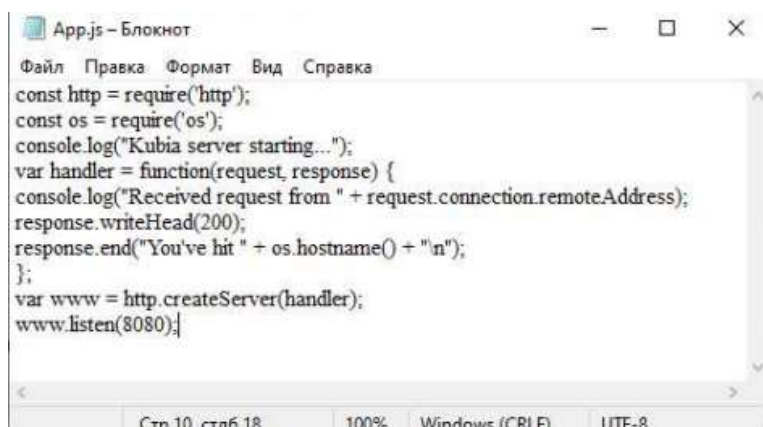


Рисунок 6 – Экранные формы для установки Docker

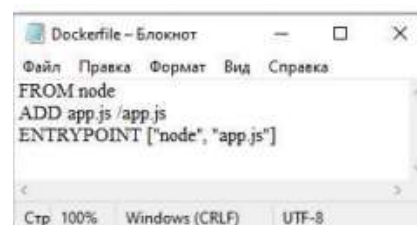
Docker предназначен для серверных приложений – веб-сайтов, API, решений для обмена сообщениями и других компонентов, работающих в фоновом режиме. Невозможен запуск настольных приложений в Docker, так как нет интеграции пользовательского интерфейса между платформой Docker и узлом Windows. Это исключает запуск приложений Windows Forms или Windows Presentation Foundation (WPF) в контейнерах (хотя возможно использование Docker для упаковки и распространения этих настольных приложений). Однако, Windows Communication Foundation (WCF), консольные приложения .NET и все разновидности ASP.NET это отличные кандидаты.

Выполним построение веб-приложения Node.js и упакуем его в образ контейнера. Приложение будет принимать HTTP-запросы и отвечать хостнеймом машины, на которой оно выполняется. Таким образом, приложение, запущенное внутри контейнера, видит собственный хостнейм, а не имя хостовой машины, даже если оно работает на хосте, как и любой другой процесс. Приложение будет состоять из одного файла под именем `app.js` с содержимым, показанным на рис.7 (а). Для упаковки приложения в образ создали файл инструкций `Dockerfile`, показанный в следующем ниже листинге, рис.7 (б).



```
const http = require('http');
const os = require('os');
console.log("Kubia server starting...");
var handler = function(request, response) {
  console.log("Received request from " + request.connection.remoteAddress);
  response.writeHead(200);
  response.end("You've hit " + os.hostname() + "\n");
};
var www = http.createServer(handler);
www.listen(8080);
```

(а)



```
FROM node
ADD app.js /app.js
ENTRYPOINT ["node", "app.js"]
```

(б)

**Рисунок7 – Приложение Node.js: `app.js` – (а) и его образ контейнера на основе инструкций в `Dockerfile` – (б)**

Сборка образа выполняется по команде Docker:  
`$ docker build -t kubia.`

Docker будет искать файл `Dockerfile` в каталоге и строить образ, рис.8.

Во время процесса сборки сначала Docker извлекает базовый образ (`node`) из общедоступного хранилища образов (Docker Hub), если только образ уже не был извлечен и не сохранен на локальной машине.



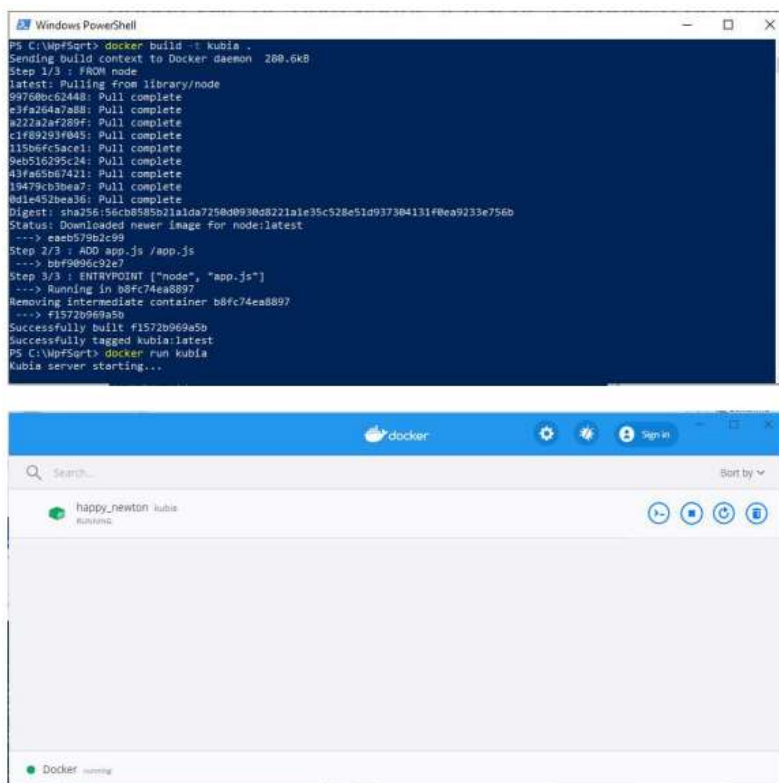


Рисунок 8 – Создание нового образа контейнера из Dockerfile

Образ можно запустить с помощью команды:

```
$ docker run --name kubia-container -p 8080:8080 -d kubia.
```

Она поручает платформе Docker запустить новый контейнер с именем `kubia-container` из образа `kubia`. Контейнер будет отсоединен от консоли (флаг `-d`), имея в виду, что он будет работать в фоновом режиме. Порт 8080 на локальной машине будет увязан с портом 8080 внутри контейнера (параметр `-p 8080:8080`), можно получить доступ к приложению через `http://localhost:8080`.

## ЗАКЛЮЧЕНИЕ

В представленной читателю статье авторы в обзорном виде раскрыли основные понятия в концепции аппаратной виртуализации на основе контейнеризации приложений на платформе Docker под ОС Windows 7, 10.

Для контейнеризации приложений и немедленного запуска их в ОС Windows авторы рассмотрели использование открытой платформы Docker, и открытого программного обеспечения Kubernetes от Google для автоматизации развёртывания, масштабирования контейнеризированных приложений и управления всеми процессами Docker-а, запущенными в host системе на локальной машине.

Без развития технологии виртуализации не был бы возможен переход к «облакам» и контейнеризации приложений в облачном хостинге.

## БИБЛИОГРАФИЧЕСКИЙ СПИСОК:

1. Антич, А. Виртуализация - эволюционная революция [Текст] / А. Антич // Системный администратор. – 2010. – № 03(88). – март. – С. 5–7.
2. Марко Лукша Kubernetes в действии [Текст] / Марко Лукша ; пер. с англ. А. В. Логунов. – М. : ДМК Пресс, 2019. – 672 с.
3. Моуэт, Э. Использование Docker [Текст] / Э. Моуэт ; пер. с англ. А. В. Снастина ; науч. ред. А. А. Маркелов. – М. : ДМК Пресс, 2017. – 354 с.
4. Docker и рост контейнеризации [Электронный ресурс]. – Режим доступа: <https://azure.microsoft.com/ru-ru/topic/kubernetes-vs-docker/> (дата обращения: 07.02.2020).