Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

**(ПНИПУ)**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

**о научно-исследовательской работе на тему**

Применение Docker для автоматизации развертывания веб-приложений

Выполнил: студент группы РИС-17-1б

Ширинов Рустам Рамазанович

Проверил: доцент кафедры ИТАС

Курушин Даниил Сергеевич

**г. Пермь, 2021**

**РЕФЕРАТ**

Отчет 15 с., 2 источн.

РАЗВЕРТЫВАНИЕ, DOCKER, КОНТЕЙНЕРИЗАЦИЯ, КОНТЕЙНЕР, ВЕБ-ПРИЛОЖЕНИЕ, ВЕБ-СЕРВЕР.

Объектом исследования является применение Docker для автоматизации развертывания веб-приложений.

Целью данной работы является исследование применения Docker для автоматизации развертывания веб-приложений.

В процессе работы проводилось исследование применения Docker для автоматизации развертывания веб-приложений.

В результате работы было выявлено, что с помощью Docker можно автоматизировать процесс развертывания веб-приложения, сократив количество выполняемых консольных команд при развертывании до одной.

**СОДЕРЖАНИЕ**

[ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 4](#_Toc69930567)

[ВВЕДЕНИЕ 5](#_Toc69930568)

[1 Исследование применения Docker для развертывания веб-приложений 7](#_Toc69930569)

[2 Реализация тестового веб-приложения 8](#_Toc69930570)

[3 Развертывание тестового веб-приложения без дополнительных инструментов 10](#_Toc69930571)

[4 Развертывание тестового веб-приложения с помощью Docker 12](#_Toc69930572)

[ЗАКЛЮЧЕНИЕ 14](#_Toc69930573)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 15](#_Toc69930574)

# ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящем отчете о НИР применяют следующие термины с соответствующими определениями:

Docker – программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений.

Dockerfile – файл, содержащий набор инструкций, следуя которым Docker будет собирать образ контейнера.

JavaScript – мультипарадигменный язык программирования. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам

Node.js – программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения и даже программировать микроконтроллеры.

Фреймворк – программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

HTTP (англ. HyperText Transfer Protocol – «протокол передачи гипертекста») – протокол прикладного уровня передачи данных, изначально – в виде гипертекстовых документов в формате HTML, в настоящее время используется для передачи произвольных данных.

cURL (Client URL) – кроссплатформенная служебная программа командной строки, позволяющая взаимодействовать с множеством различных серверов по множеству различных протоколов с синтаксисом URL.

NPM (англ. Node Package Manager) – менеджер пакетов, входящий в состав Node.js.

# ВВЕДЕНИЕ

Одним из важнейших этапов жизненного цикла программного обеспечения является развертывание – это действия, которые делают разработанную программную систему доступной для использования конечным пользователем. Если говорить о веб-приложении, то в данном случае развертывание – это процесс установки, настройки и запуска данного веб-приложения на боевом сервере, то есть на хост машине.

Развертывание веб-приложения является трудоемкой задачей и подразумевает постоянное повторение одних и тех же действий. Именно поэтому, в современной веб-разработке принято максимально автоматизировать процесс развертывания продукта. Если в маленьких продуктах ещё, может быть, разработчики позволяют себе разворачивать приложения вручную, то в более крупных продуктах такое уже невозможно, поскольку одна ошибка при развертывании приложения может стоить огромных денег для компании, потому что потенциальные пользователи не смогут воспользоваться продуктом. Помимо этого, очевидно, что время, потраченное разработчиком на развертывание веб-приложения вручную – это деньги работодателя, которые можно было бы сэкономить, автоматизировав данный процесс всего один раз [1]. Преимущества, которые предоставляет автоматизация развертывания ПО, отлично подчеркивает пример компании «Etsy», в которой когда-то процесс ручного развертывания мог занимать до четырех часов, но после автоматизации данного процесса, на его выполнение стало тратиться около 10 минут в среднем и теперь инженерный персонал компании «Etsy» выполняет развертывание до 60 раз в день [2].

Естественно, что со временем для упрощения и автоматизации процесса развертывания были созданы инструменты, которые позволяют это сделать достаточно гибко и просто. И одним из таких инструментов является Docker. Docker был создан чтобы обеспечить возможность развертывания приложений в изолированной среде, при этом автоматизируя сам процесс запуска приложения [1]. Таким образом, поскольку одной из причин создания Docker была проблема автоматизированного развертывания ПО, то исследование его применения для автоматизации развертывания веб-приложений является как никогда актуальным.

Объектом исследования является применение Docker для автоматизации развертывания веб-приложений.

Целью данной работы является исследование применения Docker для автоматизации развертывания веб-приложений.

Задачи:

* исследовать применение Docker для развертывания веб-приложений;
* реализовать тестовое веб-приложение;
* развернуть тестовое веб-приложение без использования дополнительных инструментов развертывания;
* развернуть тестовое веб-приложение с помощью Docker.

1. Исследование применения Docker для развертывания веб-приложений

Docker – это программное обеспечение с открытым исходным кодом для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации. Из предшествующего определения, становится ясно, что основной технологией, которая лежит в основе Docker, является контейнеризация – метод виртуализации, при котором ядро операционной системы поддерживает несколько изолированных экземпляров пространства пользователя вместо одного [1]. Эти экземпляры (обычно называемые контейнерами или зонами) с точки зрения пользователя полностью идентичны отдельному экземпляру операционной системы.

Таким образом, Docker предоставляет возможность упаковывать и запускать приложение в слабо изолированной среде, называемой контейнером. Данная изоляция позволяет запускать множество контейнеров одновременно на одном хосте. Контейнеры потребляют в разы меньше ресурсов, чем виртуальная машина, но при этом они содержат все необходимое для запуска приложения, благодаря этому разработку не нужно каждый раз конфигурировать новую хост машину для запуска на нем веб-приложения, и он может не волноваться о том, что в настоящее время установлено на хосте. Помимо этого, благодаря все тем же преимуществам, описанным выше, Docker позволят легко делиться образами контейнеров между разработчиками и при этом у всех он будет работать одинаково [3].

В итоге, воспользовавшись Docker можно значительно сократить задержку между написанием кода и его запуском в производственной среде [3].

1. Реализация тестового веб-приложения

Под тестовым веб-приложением, подразумевается приложение, на примере которого будет продемонстрировано развертывание вручную и с помощью Docker.

В качестве тестового веб-приложения будет реализовано приложение, написанное на языке JavaScript, с использованием платформы «Node.js» и фреймворка Express. Данное веб-приложение будет просто отвечать сообщением «Hello, world!» на любой HTTP запрос.

Для начала, чтобы создать описанное выше веб-приложение, необходимо инициализировать проект с помощью команды «npm init -y» (листинг 1).

Листинг 1 – Инициализация проекта

$ npm init -y

Wrote to /home/rustam/dev/hello-world/package.json:

{

"name": "hello-world",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"keywords": [],

"author": "",

"license": "ISC"

}

Следующим шагом необходимо установить фреймворк Express (листинг 2).

Листинг 2 – Установка фреймворка Express

$ npm i express --save

npm notice created a lockfile as package-lock.json. You should commit this file.

npm WARN hello-world@1.0.0 No description

npm WARN hello-world@1.0.0 No repository field.

+ express@4.17.1

added 50 packages from 37 contributors and audited 50 packages in 2.633s

found 0 vulnerabilities

Далее, необходимо создать файл с именем «server.js» и, с помощью функционала, предоставляемого фреймворком Express, реализовать веб-сервер, который будет слушать порт 5000 и отвечать на все запросы «Hello, world!». Реализация данного веб-сервера представлена листинге 3.

Листинг 3 – Реализация HTTP сервера с помощью фреймворка Express

const express = require('express');

const app = express();

const port = 5000;

app.all('\*', (req, res) => {

res.end('Hello, world!');

});

app.listen(port, () => console.log(`Server has been started on port ${port}`));

Теперь можно запустить реализованный веб-сервер. Запуск веб-сервера приведен на листинге 4.

Листинг 4 – Запуск веб-сервера

$ node server.js

Server has been started on port 5000

Проверим работу запущенного веб-сервера с помощью утилиты cURL (листинг 5).

Листинг 5 – Проверка работы запущенного веб-сервера с помощью утилиты cURL

$ curl localhost:5000

Hello, world!

$ curl -X POST localhost:5000

Hello, world!

Таким образом, было реализовано тестовое веб-приложение, которое отвечает сообщением «Hello, world!» на любой HTTP-запрос.

1. Развертывание тестового веб-приложения без дополнительных инструментов

Для того, чтобы показать пример развертывания тестового веб-приложения, будет использоваться хост машина с ОС Ubuntu 20.04. Подразумевается, что папка с проектом уже лежит на хост машине и на хост машине уже установлено всё необходимое, кроме непосредственно зависимостей тестового веб-приложения.

У реализованного тестового веб-приложения есть следующие зависимости:

* Node.js v12 и NPM (поставляются вместе);
* Express.js.

Таким образом, для развертывания тестового веб-приложения для начала необходимо установить Node.js и NPM. Команды для установки нужной версии Node.js и NPM представлены на листинге 6.

Листинг 6 – Команды для установки Node.js v12 и NPM

curl -fsSL https://deb.nodesource.com/setup\_12.x | sudo -E bash -

sudo apt-get install -y nodejs

После этого необходимо установить все NPM зависимости проекта (в данном случае это только фреймворк Express) с помощью одноименного пакетного менеджера (листинг 7).

Листинг 7 – Установка NPM зависимостей проекта

$ npm i

npm WARN hello-world@1.0.0 No description

npm WARN hello-world@1.0.0 No repository field.

added 50 packages from 37 contributors and audited 50 packages in 0.619s

found 0 vulnerabilities

Теперь, когда все зависимости были установлены, можно запустить тестовое веб-приложение (листинг 8).

Листинг 8 – Запуск развернутого тестового веб-приложения

$ node server.js

Server has been started on port 5000

Проверим работу запущенного веб-приложения с помощью утилиты cURL (листинг 9).

Листинг 9 – Проверка работы развернутого веб-приложения

$ curl localhost:5000

Hello, world!

$ curl -X POST localhost:5000

Hello, world!

Таким образом, для развертывания реализованного тестового приложения без дополнительных инструментов необходимо было осуществить три действия: установить Node.js и NPM, установить NPM зависимости, запустить веб-приложение.

1. Развертывание тестового веб-приложения с помощью Docker

Для развертывания тестового веб-приложения требуется установленный на хост машине Docker. Помимо этого, само веб-приложение должно быть упаковано в Docker-образ, с помощью которого будет создаваться контейнер. Для того, чтобы создать Docker-образ тестового веб-приложения, необходимо создать Dockerfile, содержащий набор инструкций, следуя которым Docker будет собирать образ контейнера [3]. Необходимый для создания Docker-образа тестового веб-приложения Dockerfile приведен на листинге 10.

Листинг 10 – Dockerfile для создания Docker-образа тестового веб-приложения

FROM node:12

WORKDIR /app

COPY . .

RUN npm i

CMD ["node", "server.js"]

EXPOSE 5000

Далее необходимо создать Docker-образ тестового веб-приложения. Команда для создания Docker-образа тестового веб-приложения с названием «hello-world» приведена на листинге 11.

Листинг 11 – Создание Docker-образа тестового веб-приложения

$ docker image build -t hello-world .

[+] Building 51.9s (10/10) FINISHED

Все, теперь все подготовительные работы для развертывания тестового веб-приложения выполнены. Приведенные выше действия не относятся непосредственно к развертыванию, поскольку собрать образ можно один раз и далее развертывать его на бесконечное количество машин.

Теперь, имея те же исходные данные, что и в случае развертывания без использования Docker, развернем наше тестовое веб-приложение на новой хост машине. Стоит отметить, что в данном случае не учитывается способ получения Docker-образа на новую хост машину, предполагается, что он уже на ней есть. В стандартном случае, скорее всего, Docker-образ будет лежать в удаленном Docker-реестре и при запуске контейнера если его нет на хост машине, то он автоматически будет загружен из реестра.

Развертывание тестового веб-приложения из Docker-образа приведено на листинге 12.

Листинге 12 – Развертывание тестового веб-приложения из Docker-образа

$ docker run -d -p 5000:5000 hello-world

ecdba73130e623e4cc19167e6aed0876c833f2ab8c9e0cbd146de1315924dea0

В итоге, будет запущен контейнер с тестовым веб-приложением. Проверим работу запущенного веб-приложения с помощью утилиты cURL (листинг 13).

Листинг 13 – Проверка работы развернутого с помощью Docker веб-приложения

$ curl localhost:5000

Hello, world!

$ curl -X POST localhost:5000

Hello, world!

Таким образом, для развертывания реализованного тестового приложения с помощью Docker необходимо было выполнить лишь одну команду.

# ЗАКЛЮЧЕНИЕ

В ходе работы было проведено исследование применения Docker для автоматизации развертывания веб-приложений. В результате проделанной работы были выполнены следующие задачи:

* исследовано применение Docker для развертывания веб-приложений;
* реализовано тестовое веб-приложение;
* развернуто тестовое веб-приложение без использования дополнительных инструментов развертывания;
* развернуто тестовое веб-приложение с помощью Docker.

Таким образом, было выявлено, что с помощью Docker можно автоматизировать процесс развертывания веб-приложения, сократив количество выполняемых консольных команд при развертывании до одной.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Моуэт Э. Использование Docker. Разработка и внедрение программного обеспечения при помощи технологии контейнеров: руководство. – ДМК-Пресс, 2017. – 300 с.
2. Дэвис Д., Дэниелс К. Философия DevOps. Искусство управления IT. – СПб.: Питер, 2017. – 416 с.
3. Docker overview | Docker Documentation. – URL: https://docs.docker.com/get-started/overview/ (дата обращения 13.04.2021).