National College of Ireland

Automated Dockerization of Web Applications Developed in Various Languages

MSc Research Project
Cloud Computing

Shubham Sanjay Tendulkar
Student ID: x20224753

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Shubham Sanjay Tendulkar
Student ID:	x20224753
Programme:	Cloud Computing
Year:	2022
Module:	MSc Research Project
Supervisor:	Vikas Sahni
Submission Due Date:	15/08/2022
Project Title:	Automated Dockerization of Web Applications Developed in Various Languages
Word Count:	5100
Page Count:	18

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	19th September 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Automated Dockerization of Web Applications Developed in Various Languages

Shubham Sanjay Tendulkar
x20224753

Abstract

The competition for computing is at an all-time high as the demand for computing power is increasing, and cloud computing is a great solution. For a long time, one of the main issues while developing has been that the developers develop the application in one place and then deploy the application in another. Then this becomes an issue for DevOps.One of the best strategies is to use Docker to resolve the application's deployment, but dockerizing an appliction is still a difficult process. Therefore even though Docker is useful, it is not widely used. The purpose of this research is to reduce its complexity. Therefore, this research, proposes a solution to reduce the complexity of dockerization. The automated dockerization application simplifies the process of Dockerization and hides the complexity of dockerizing a web application. The following research discusses virtualization, containerization, and dockerization. This research also discusses the benefits of dockerization. The automated dockerization can dockerize web applications written in ReactJS and ExpressJS it can also create WordPress instance on docker. The Dockerized application can be readily uploaded to any environment that supports Docker, notably cloud-based environments like Amazon AWS EC2. The application for automated dockerization appeals to a bigger audience because it supports popular languages like ReactJS(Frontend) and ExpressJS(Backend), this will encourage developers to utilize Dockers more.

1 Introduction

Recently, there has been much technological development in serverless computing, IoT, etc., which has created a great scope for the cloud Buyya (2010). In this context, the most popular techniques utilized in today's world are containers and virtual machines to improve the scalability and portability of any web application as well as to facilitate cross-platform use. While each has its own benefits, it is sometimes crucial to pick the best strategy for the circumstances. Optimizing server capacity is the basic goal of virtualization. However, given the operating system's existence and the requirement for a hypervisor to enable virtualization, this cannot be referred to as lightweight.

1.1 Docker

According to the accepted definition, Docker is "an open-source project Bernstein (2014) that automates OS-level virtualization on Linux and adds an additional layer of abstraction Teigland and Mauelshagen (2001) to the deployment of software applications inside

containers." Because of its open-source platform, Docker is more preferred over virtualization as technology advances. Docker enables both development and deployment processes. Docker's ability to separate the infrastructure from the application is one of its most appealing features Merkel et al. (2014). Docker's increased use has made it simpler to concentrate on developing applications rather than worrying about how they will be deployed Naik (2016). The management of infrastructure is, therefore, more enticing than ever. The ability to create Docker images and deploy them reduces the amount of time that passes between the development of new code and its deployment, preventing developers from getting sidetracked by other tedious chores. The main advantage here is the use of containers, which in this scenario is compared to virtualization. Containers have attracted many developers due to their lightweight and lack of an operating system, as opposed to virtual machines.

1.2 ReactJS

In its most basic form, React is a web framework that was developed with the primary intention of resolving performance concerns that were present in web applications Javeed (2019). Within the Model-View-Controller architectural pattern, React is referred to as the View (V) (MVC). As a result of its abstraction of the Document Object Model (DOM), React provides an approach to application development that is straightforward, performant, and reliable. The majority of the rendering for React is done on the server side with NodeJS, while support for native mobile app development is provided via React Native. As a result of implementing unidirectional data flow, which reduces the amount of boilerplate code required, React is far simpler to use than traditional data binding Aggarwal (2018). Facebook's internal development team came up with the UI library known as React in order to make it easier to create interactive, stateful, and reusable UI components. It is put to use in the production process at Facebook. Rendering sophisticated user interfaces while maintaining a high level of efficiency is best accomplished with ReactJS Kumar and Singh (2016). With a utilization rate of 35.9% and over 7.4 million live websites currently using the library in 2022, reactJS was the second most utilized web development framework worldwide in 2021 Tushev (2022). Therefore the automated docerization application from this research supports dockerization of ReactJS application.

1.3 WordPress

Support for creating WordPress instance in a docker is provided in the automated docerization application. The reason why this application suports this functionality is because the most widely used content management system (CMS) in today is WordPress, which is used to power over 29 percent of all websites Cabot (2018). WordPress is used to power approximately 56 percent of all websites that are driven by content management systems (CMSs), which is a significant percentage. This is an astounding figure that speaks not only to the growing rate of adoption of WordPress by users but also to its long-term viability as a content management system Jones and Alida-Farrington (2011). The PHP programming language and either MySQL or the MariaDB Relational Database Management System are the foundations upon which the WordPress web application is built (RDBMS). As of right now, it is the open-source website Content Management System (CMS) that is utilized by the most websites Tomiša et al. (2019). Mostly even people with

less technical skills develop and create WordPress websites, therefore for these users the automated docerization application hides the complexity of creating WordPress instance on docker.

1.4 ExpressJS

ExpressJS is a framework that is developed on the Node platform. It does this by offering a more user-friendly application programming interface (API) for several of Node's key features. It is possible to think of it as an abstraction layer that sits on top of the HTTP module that is part of Node's core API. ExpressJS offers a significant amount of additional functionality in comparison to the HTTP module. This functionality eliminates the necessity to rewrite typical tasks, such as handling requests, setting routes, or rendering static assets, from start Peters (2017). Overall ExpressJS is a good framework for developing backend. Therefor the automated docerization application supports dockerzing applications written in EpressJS.

1.5 Research Question

Will using an intuitive user interface and masking the complexities of dockerizing ReactJS, WordPress, and ExpressJS web applications enhance the adoption rates and the number of dockerized applications?

The remainder of the study is dedicated to a discussion of the research that are associated with Docker, during which the pros of utilizing Docker are subjected to an in-depth evaluation. After that, the author of the paper explains how the process of dockerization should be simplified in order to boost the application of docker. In addition to this, the study delves into the technique that may be utilized to make the dockerization process more manageable.

2 Related Work

The core of this research is to promote the use of docker. Therefore this research focuses on work done on docker. Docker was initially released in 2013 Combe et al. (2016). After releasing in 2013 a lot of research has been done on docker Rad et al. (2017). This research extends the work done on docker and tries to promote the use of docker.

2.1 Dockerizing or Containerizing an application

External dependencies, such as frameworks, modules, libraries, and so on, are common in web applications. There may be instances of multiple applications that are built on the same framework but run on different versions that are either compatible or incompatible with one another Ijtihadie et al. (2017). Nowadays, everyone strives to reduce deployment time and to perform multiple deployments in very less time. Therefore DevOps gains a lot of popularity at the same time containers pack dependencies into docker file. Docker files are secure and isolated, allowing cross-platform and version issues to be resolved. Similar deployment steps can be followed in the development, staging, and deployment processes Bin et al. (2019). There are so many middleware methods to dockerizing applications for cloud deployment. The ontology approach does not allow for

the creation of ontologies and their implementation Pahl and Carle (2013). The gap between the developer community and cloud infrastructure community has been bridged by Paas. However, being able to design an application regardless of the target platform gives every developer an edge in terms of increasing development productivity. This is made feasible by a devops ecosystem that includes Google App Engine, Docker, and Kubernetes, among other things.

2.2 Microservices

Many businesses have evolved into Microservices architecture as technology and DevOps have advanced. Specifically, e-commerce systems make it easier to control and reduce overall application downtime. This change has been supported by DevOps Ci/CD using RunDeck, Docker, and Kubernetes. The Docker images were submitted to Docker Hub. These images can be pulled from Docker Hub, but the essential goal is to keep the flow between the microservices intact. A single yml file was introduced that runs alongside the docker-compose, keeping the containers connected. The emphasis remained on the development of the yml file. Manual configuration is not required in the production environment of Docker-compose Rajavaram et al. (2019). Kubernetes supports calling multiple pods, but our approach sets up the portability in very less time. Automation and virtualization have also been applied in situations like as connecting mobile terminals with cloud services, which is then referred to as Testing as a Service (TaaS) Tao et al. (2015). The apps can also be tested for vulnerability here. This makes it easier to create flexible testing environments for various scenarios. This strategy can be used with our suggestion to increase adaptability even further, resulting in speedier containerization and, over time, safe environments on the hosted cloud for the applications.

2.3 Dockeriztion of pyhton web apps

Kedambadi Shreekar (2020) has previously worked on automated dockerization of Python-based web apps, and this research extends the work on that. This research proposes to automate the dockerization of web applications in ReactJS, WordPress, and ExpressJS, which will simplify the process of dockerization.

In conclusion, while prior methods have addressed concerns about the security of cloud-hosted apps and created FlowVisors, the majority don't handle the complexity of this move, and the method that does address the complexity doesn't support many languages. For web programs that need to be transferred and to reduce the overhead of configuration setup, containerization is the most suitable method when compared to virtual machines. This research wants to make the process simpler by automating web app dockerization to encourage adoption. How to make Docker adoption simpler and easier? What can be done, specifically in the phases from no container to container engine stage, to enable teams who are hesitant to begin using Docker to recognize the benefits and simplicity of doing so? This is what the research paper are aiming towards.

3 Methodology

The main logic is developed such that it may be readily extended to handle non-web applications and different types of programming languages in the future. Eventually, a comprehensive web application that can carry out these activities instantly will be

created, keeping the essential logic the same. For the proposal's effective implementation, our method of implementation incorporates a variety of concepts and frameworks, as shown below.

- The capability of utilizing the command line interface or shell to run Docker as a client.
- The capability of downloading all the required dependencies which are used by ReactJS, ExpressJS and Wordpress. The ability to run commands for command line to reduce the complexity. A simple to understand user interface to take inputs from user and perform its task.

This method calls shell commands and the relevant stacks surrounding it in order to take advantage of Python's programming strengths. The methodology utilized was agile in that features were introduced as they were discovered and new features were used as needed. This was mostly caused by the project's research-based approach, which resulted in agile mode. It was possible to install and manage packages created in Python using the pip package management system. Despite the fact that Python offers a number of graphical user interface frameworks, Tkinter framework a built-in framewoed was more suitable for the automated docerization application. One very significant factor is that this research proposal was strongly supported by the cross-platform nature of Tkinter, which functions equally well with any Operating System. Light weighted in nature, it was more persuasive for the cause.

Additional module In different operating systems, OS module in python provides functions for interacting with the operating system. The application's foundational module is this one. Using this module makes it feasible to execute different docker-related commands and procedures to produce docker artifacts. Another great feature is its ability to run in the background.

By selecting ReactJS and ExpressJS based web applications and Wordpress from open source code repositories like GitHub, compiling them using our setup to produce a Docker file, and then being able to execute the image on our target platform, the experiments were conducted and tested.

- Write the ReactJS(Frontend)/ ExpressJS(Backend) application code. Or input wordpress details to deploy wordpress on docker.
- Create a Docker file using containers.
- Deploy to the chosen target (local system or cloud).

The Environment requirements are as stated below: For setting up our implementation we had the following setup done:

- Installed Python 3.x. The most recent version of Python 3.x should always be used.
- Tkinter and numpy libraries must be installed.
- Docker Desktop has been set up. To execute commands using Docker, this is necessary. The issue with the aforementioned is that it only works with Windows Professional and later and cannot be utilized with Windows Home.
- If installing Docker Desktop is not possible, we can use Docker clients for older operating systems.

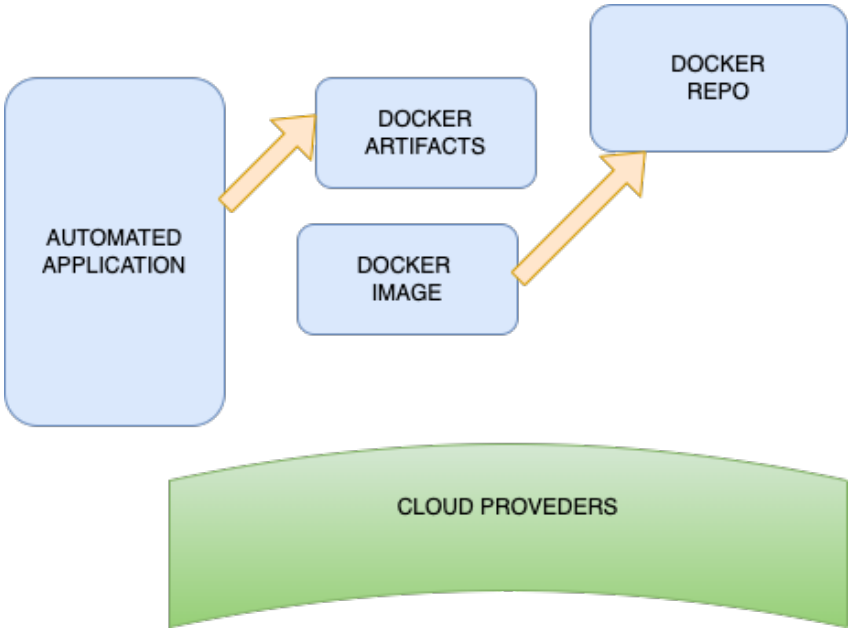

Figure 1: Flow of automated application .

4 Design Specification

The purpose of this research was to create an application that simplifies the process of dockerizing any web application written in popular languages like:

- ReactJS
- ExpressJS

This application can also dockerize and deploy Wordpress. With the help of this application, developers can easily develop an application then containerize/build docker file and then they can deploy to target of their choice like local system or cloud. Also with the help of this application users can also Dockerize and deploy wordpress with ease.

4.1 Input Specification:

The input specification is divided into three sections. In the first section, information regarding ReactJS input is provided, the second section provides information regarding WordPress inputs, and in the last, third section, information regarding ExpressJS is provided.

4.1.1 ReactJS Input Specifications

The first section of the input specification is about ReactJS Dockerization.

- Project directory :- In order to Dockerize a ReactJS application, users must first specify the directory in which the application resides. This can be done by clicking browse button in the application.
- Project name :- The second step is to input the project name for ReactJS application.

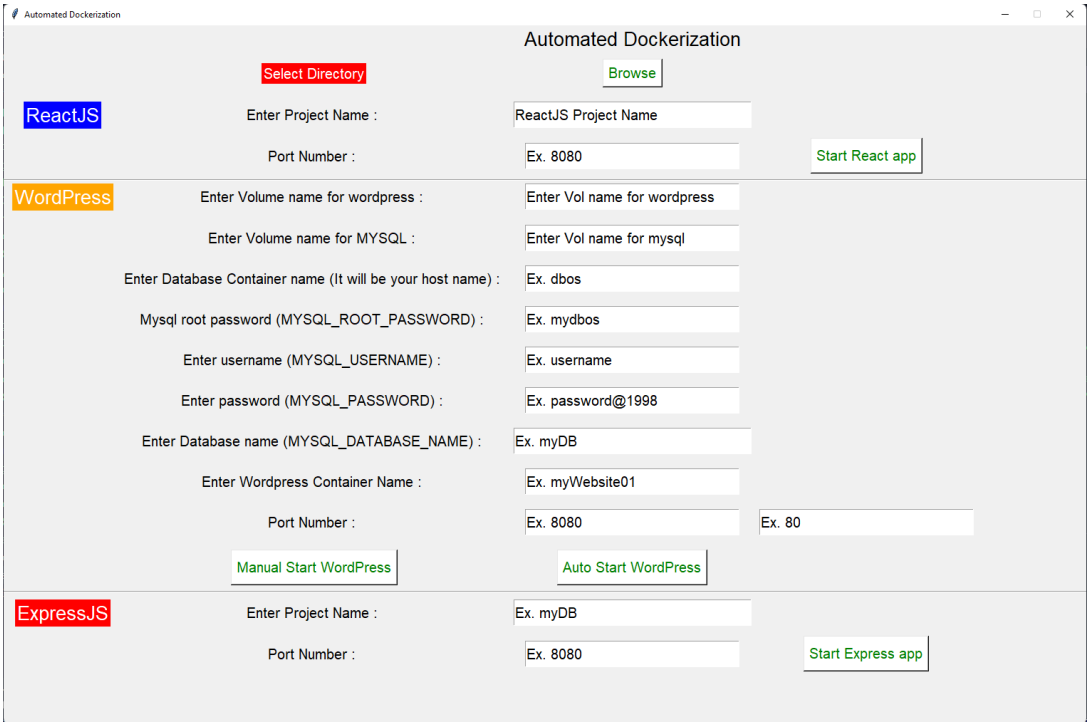

Figure 2: UI / Input Tab of the applicaiton .

- Port Number :- The last thing to do is to enter the port number which needs to be exposed by the application from docker.

4.1.2 WordPress Input Specification

This section describes the WordPress Input Specification, which assists users with dockerizing and deploying WordPress. There are two methods by which users can dockerize WordPress either automatically or manually. In the manual method, every detail needs to be provided by the user and configure the WordPress manually, whereas in the automatic method, the user can dockerize WordPress automatically with the click of just one button.

- Volume name for WordPress:- The user identifies the WordPress volume by giving its name in this field.
- Volume name for MYSQL:- The user is responsible for providing the volume name for MYSQL in this field.
- Database Container Name:- The user is responsible for providing the container name for Database in this field. This name will also serve as the host name.
- MYSQL root password :- This is the field in which the user enters the root password for MYSQL.
- MYSQL username :- This is the field in which the user supplies their username for usage with MYSQL.

- MYSQL password :- The user's password for accessing the MYSQL database is entered into this field.
- MYSQL database name :- In this area, the user enters the name of the database that will be used by MYSQL.
- WordPress container name :- The user is providing a name for the WordPress container in this area.
- Port number :- The port number for both the host and the container should be entered into this field by the user.

4.1.3 ExpressJS Input Specification

This section discusses about the ExpressJS Input Specification.

- Project directory :- In order to Dockerize a ExpressJS application, users must first specify the directory in which the application resides. This can be done by clicking browse button in the application.
- Project name :- The second step is to input the project name for ExpressJS application.
- Port Number :- The final step is to specify the port number that must be exposed by the docker application.

4.2 Output Specification

This application has the capability of producing a very wide variety of different kinds of output at any given time. The build artifacts that are produced by the docker build command are the ones that are encountered most frequently. This represents the application's output at a lower level than the previous one. The ability to run the Docker application and the capability to access it using web browsers constitutes the second level. Additionally, the capability to execute WordPress in a dockerized environment has been implemented. Even if it is not currently possible, it will eventually have the capacity to push the docker artifacts to the central repository. The objects that are uploaded in this manner are able to be uploaded without much difficulty on any public cloud network or Kubernetes platform.

5 Implementation

Python was the programming language that was decided to utilize for the purpose of implementation. This allowed us to construct user interfaces and logic processing. Tkinter, which is the graphical user interface component of Python, and OS module, which deals with launching processes in multiple operating systems from within Python, are the two key modules that made up the application. Tkinter was utilised because it is compatible with a variety of operating systems, including Windows, Linux, and macOS, which is the primary benefit of using this tool. Tkinter contains a broad variety of widgets from which to choose. However, for the purpose of this project, the input box was used, which is also referred to as the entry box, the file dialog box, and the button. In addition, there were

two distinct forms of the file dialog that were utilized. The first is the conventional file dialog, which enables file selection. The second is the directory file dialog, which enabled directory selection. This is necessary due to the fact that the input specification indicates that both files and directories were required. The event action pattern was achieved through the use of callbacks. That made it possible to regulate certain behaviors that are essential for either capturing the names of files or directories or dealing with the validation of input parameters that are supplied by the user.

- Dockerizing ReactJS Application
  If the user has already developed an application with ReactJS, then the GUI should be used to go to the directory of ReactJS project. After the directory has been selected, the user is required to specify a name for the ReactJS application. Following that, the user will be required to provide the port number that should be exposed for the ReactJS application.
  In the backend the Tkinter application will store the directory path provided by the user through GUI. This is performed by the function filedialog.askdirectory(). The application then fetches images from Docker, and if the image already exists, it replaces the existing or duplicate image with the latest image. The application then examines whether the specified directory path is legitimate, and if the directory already exists, it deletes and recreates it. The directory's contents are then cloned into the react-app directory. The application will then move its directory to the react-app directory. The program then generates a docker file for the react application. This is followed by the execution of the docker build command in the background, which builds the docker image. And finally, the application executes the container and exposes the port number supplied by the user.

- Code for Dockerizing ReactJS Application

```
    def createReactDockerFile(appName, portNo, dir):
# print (dir)
app_folder = 'react-app'
images = 'docker images | grep {}:latest'.format(appName)
rm_image = 'docker rmi {}:latest'.format(appName)
if os.system(images):
    os.system(rm_image)
dirpath = Path(os.getcwd()) / app_folder
if dirpath.exists() and dirpath.is_dir():
    shutil.rmtree(app_folder)
os.mkdir(app_folder)
copy_tree(dir, app_folder)
os.chdir(app_folder)

with open("Dockerfile",'w',encoding = 'utf-8') as f:
    f.write('FROM node:18-alpine3.15\n')
    f.write('WORKDIR /app\n')
    f.write('ENV PATH /app/node_modules/.bin:$PATH\n')
    f.write('COPY . .\n')
    f.write('RUN npm install\n')
    f.write('RUN npm install react-scripts@5.0.1 -g\n')
```

```
        f.write('CMD ["npm", "start"]')

    os.system('docker build -t {}:latest .'.format(appName))

    os.system('docker run -d -p {}:3000 {}:latest'.format(portNo, appName))
```

- Dockerizing WordPress
  There are two ways for users to dockerize and run a WordPress instance through the application. The application offers two options for dockerizing WordPress. The first type is automatically dockerizing WordPress. With this method, the user doesn't need to enter any information. They can simply click a button to dockerize and run WordPress. The application's backend runs the docker-compose.yml file when the user clicks the "Auto Start Wordpress" button. In this file default values has been set which will instantiate WordPress 6.0.1 and MYSQL version 8.0.3 on the docker. The user will see a browser open after clicking the button, with WordPress installed and running locally on port 8080:80. The user can then setup WordPress and begin creating websites.
  In the second way, the user must specify the WordPress volume name. The user is then needed to specify the volume name for MYSQL. The user must next specify the Database container name. This will also be the host's name. The user is then requested to input the MYSQL root password. The user must next provide a username for MYSQL. Then, the user must set a password for MYSQL database access. The user must next provide the database name that will be utilized by MYSQL. The user must then specify a container name for WordPress. The user must then specify the port number for both the host and the container. After the user gives all required details, a docker compose file containing those details is generated. The application's backend then executes the docker-compose file when the "Manual Start Wordpress" button is clicked. This file will instantiate WordPress 6.0.1 and MYSQL 8.0.3 on a Docker container. After pressing the button, a browser will appear with WordPress installed and running locally on the user-specified port. The user can then configure WordPress and initiate website creation.

- Dockerizing ExpressJS Application
  The GUI should be used to access the expressJS project directory if the user has already created an application with expressJS. The user must next enter a name for the ReactJS application after choosing a directory. The user will next need to enter the port number that should be exposed for the expressJS application.
  The Tkinter program will save the directory path provided by the user via the GUI in the backend. This is performed by the function filedialog.askdirectory(). The application then retrieves images from Docker, and if the image already exists, it replaces the existing or duplicate image with the most recent image. The application then examines whether the specified directory path is legitimate, and if the directory already exists, it deletes and recreates it. The directory's contents are then copied into the expressJS-app directory. The program will then move its directory to the expressJS-app directory. The application then generates a docker file for the expressJS application. This is followed by the execution of the docker build command in the background, which builds the docker image. And finally, the

application executes the container and exposes the port number supplied by the user.

- Code for Dockerizing ExpressJS Application

```
    def createExpressDockerFile(appName, portNo, dir):
app_folder = 'express-app'
images = 'docker images | grep {}:latest'.format(appName)
rm_image = 'docker rmi {}:latest'.format(appName)
if os.system(images):
    os.system(rm_image)
dirpath = Path(os.getcwd()) / app_folder
if dirpath.exists() and dirpath.is_dir():
    shutil.rmtree(app_folder)
os.mkdir(app_folder)
copy_tree(dir, app_folder)
os.chdir(app_folder)

with open("Dockerfile",'w',encoding = 'utf-8') as f:
    f.write('FROM node:18-alpine3.15\n')
    f.write('WORKDIR /app\n')
    f.write('ENV PATH /app/node_modules/.bin:$PATH\n')
    f.write('COPY . .\n')
    f.write('RUN npm install\n')
    f.write('RUN npm install react-scripts@5.0.1 -g\n')
    f.write('CMD ["npm", "start"]')

os.system('docker build -t {}:latest .'.format(appName))

os.system('docker run -d -p {}:3000 {}:latest'.format(portNo, appName))
```

6 Evaluation

The application's execution resulted in three distinct results. In the first output, the program dockerized a user-provided ReactJS application. This application was running on the user provided port number. In the second output, a WordPress instance was launched on the user's docker and was running on the port number specified by the user, in other output, WordPress was running on the default port number of 8080:80. In the third output, the program containerized an existing ExpressJS project that the user provided. The ExpressJS application was executed on the user-specified port. When users submitted valid inputs, the application functioned effectively. // Following were the few errors that occurred when testing the application.

- Error during connect :- This error occurred when the user attempted to start a dockerizing a project without first starting Docker desktop. This issue could be resolved by installing and launching docker desktop on the user's machine.

- Directory attribute error :- This issue occurred when the user attempted to start dockerizing program without selecting the directory where the project is located. This results in an empty directory variable. This error can be resolved by selecting the directory in which the project is saved using the "browse" button on the GUI and then clicking the "start" button.

- Tkinter/numpy error :- This issue occurs while running the command python/python3 main.py to launch the application. If Tkinter or numpy packages are not installed, this error occurs. To address this issue, the user must correctly install the Tkinter and numpy packages on their system.

- Empty fields error :- This error happens when the user attempts to begin the dockerization process without providing the necessary details for the project, or when the user provides inaccurate information. The solution to this problem is to deliver accurate information without error.

If the application is executed properly, then it functions properly and delivers the desired results. Therefore, by using this application, the complexity of dockerizing web apps and WordPress is hidden. The following experiments showcase the output as well as the different functionalities of the application.

6.1 Experiment 1

In the first experiment, the ReactJS application was containerized. It was decided to containerize the react-boilerplate application for this experiment. In order to execute this test, we initially prepared the system's environment. This was accomplished by installing docker desktop and the Tkinter and numpy package libraries. Then, to obtain the react-boilerplate application following command is executed.

```
npx create-react-app new-app
```

Executing this command creates a react-boilerplate application in new-app directory. The Automated Docker application is then launched by executing python main.py. Next, in the graphical user interface, we picked the location where the react-boilerplate application is saved, gave the project a name, and entered port 6060. After entering our information, we pressed the "Start React app" button. After hitting the button, the react-boilerplate application was dockerized, and the react application could be viewed in a web browser at "http://localhost//6060", as shown in the figure3.

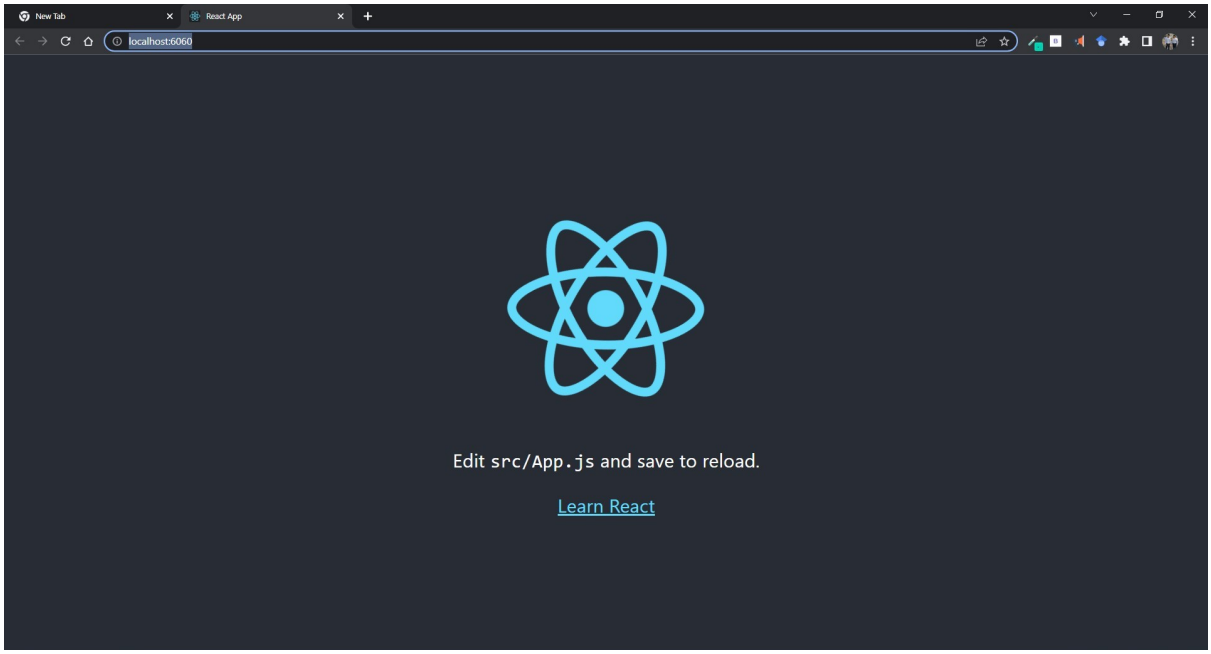

Figure 3: ReactJS app running on port 6060

6.2 Experiment 2

In the second experiment, a WordPress instance was manually configured on the system's docker. Since we had already prepared the system environment for the initial experiment, additional preparation was not necessary. Therefore, we executed the application by directly invoking the python main.py command. This initiated the GUI of the application. In this GUI, we entered the information shown in figure4.

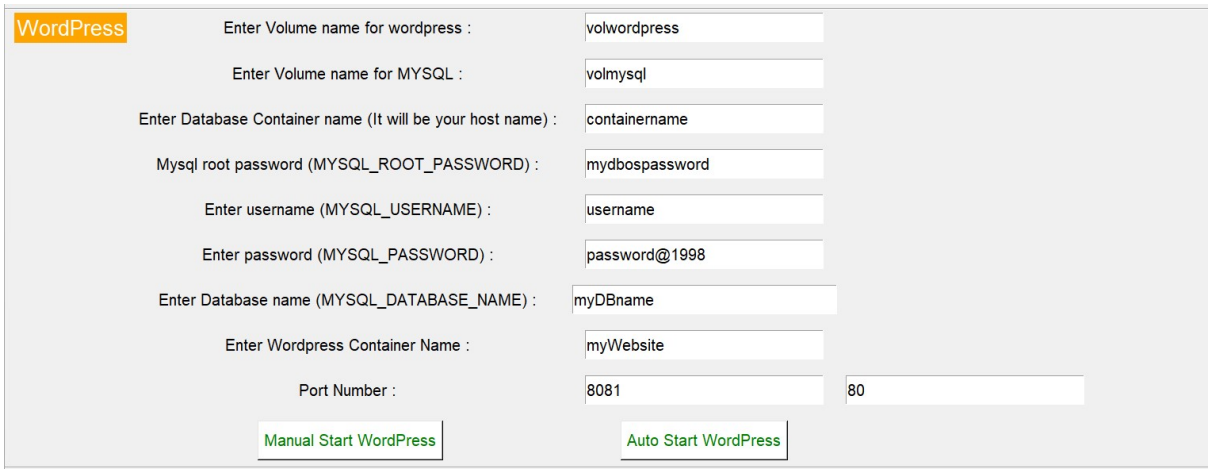

Figure 4: Manual Wordpress dockerization inputs

Then we clicked the "Manual Start WordPress" button. After hitting this button, the intended results were obtained. Wordpress 6.0.1 was installed and operating on the specified port, as shown in the figure5.

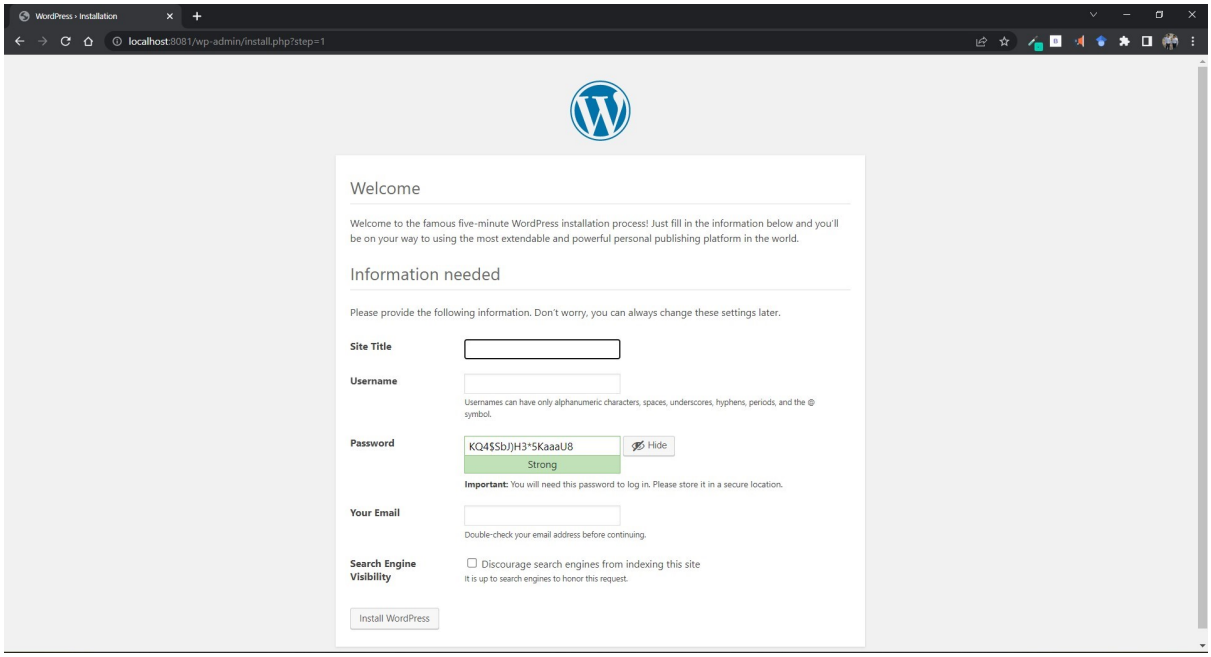

Figure 5: Wordpress instance running on manually configured port :8081

6.3 Experiment 3

In the third experiment, the system's docker was automatically configured with a WordPress instance. Since we had already set the system environment for the first and second experiments, no further preparation was required. Therefore, the application was executed by immediately executing the python main.py command. This launched the application's GUI. In this GUI, it was unnecessary to enter any information because default settings were utilised. We then clicked the "Automatically Start WordPress" button. After clicking this button, the desired outcome was accomplished. As indicated in the illustration, Wordpress 6.0.1 was installed and running and could be viewed in a web browser at "http://localhost//8080", as depicted in the figure6.

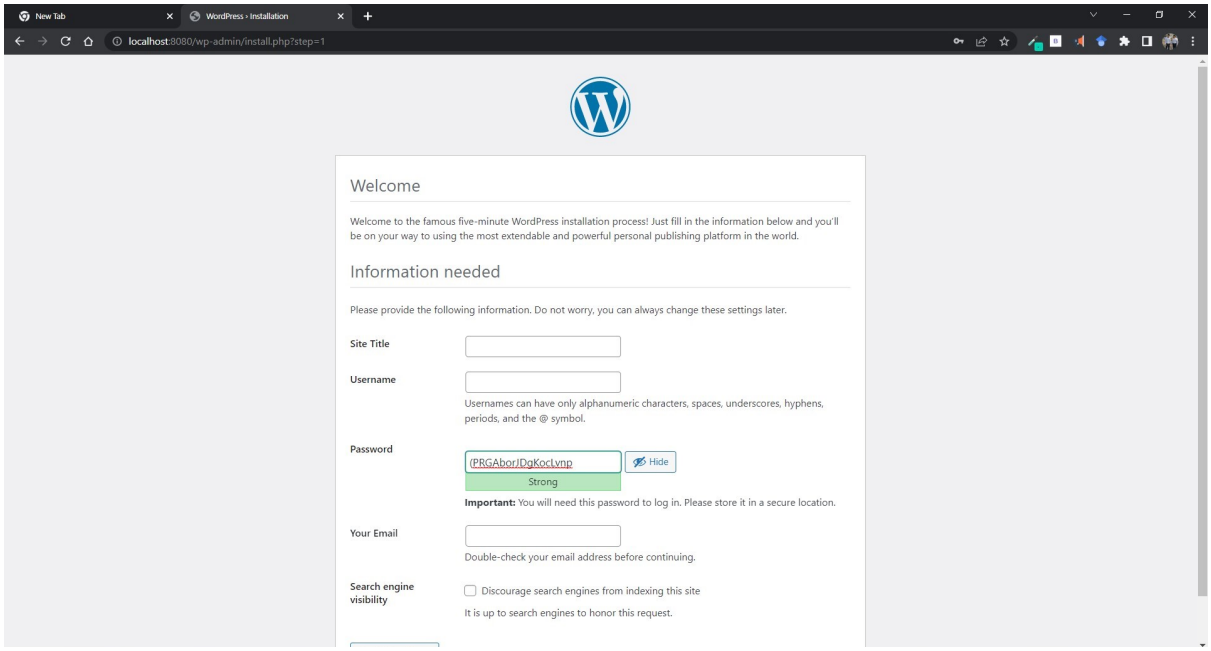

Figure 6: Wordpress instance running on manually configured port :8080

6.4 Experiment 4

The fourth experiment included containerizing the ExpressJS application. Due to the fact that we had already established the system environment for the first and second experiment, additional preparation was unnecessary. We cloned a basic ExpressJS application from GitHub in order to test it. The Automated Docker application is then launched by executing python main.py. Next, in the graphical user interface, we picked the directory where the ExpressJS application is placed, named the project, and entered port 5050. After entering our information, we hit "Start Express app." After hitting the button, the ExpressJS application was dockerized and could be viewed in a web browser at "http://localhost//5050", as depicted in the figure7.

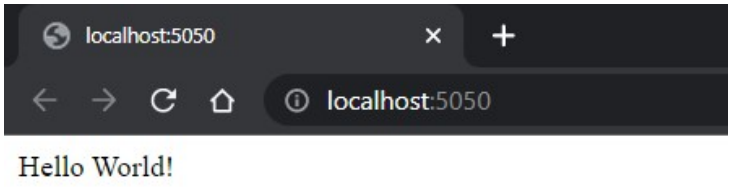

Figure 7: ExpressJS app running on port 5050

6.5 Discussion

The preceding experiments indicate that the automated dockerization application facilitates the dockerization of ReactJS and ExpressJS applications. It also demonstrates

that a user with no prior understanding of dockerization may construct a WordPress instance on their docker using the application's graphical user interface. This will assist the user in developing and testing their WordPress site locally on their docker. All of the aforementioned studies lead us to the conclusion that the automated dockerization application performs well and it can been seen in figure8 that containers were created as per the experiments.It also states that with the help of the Tkinter GUI, complexity of the dockerization process is hided away.

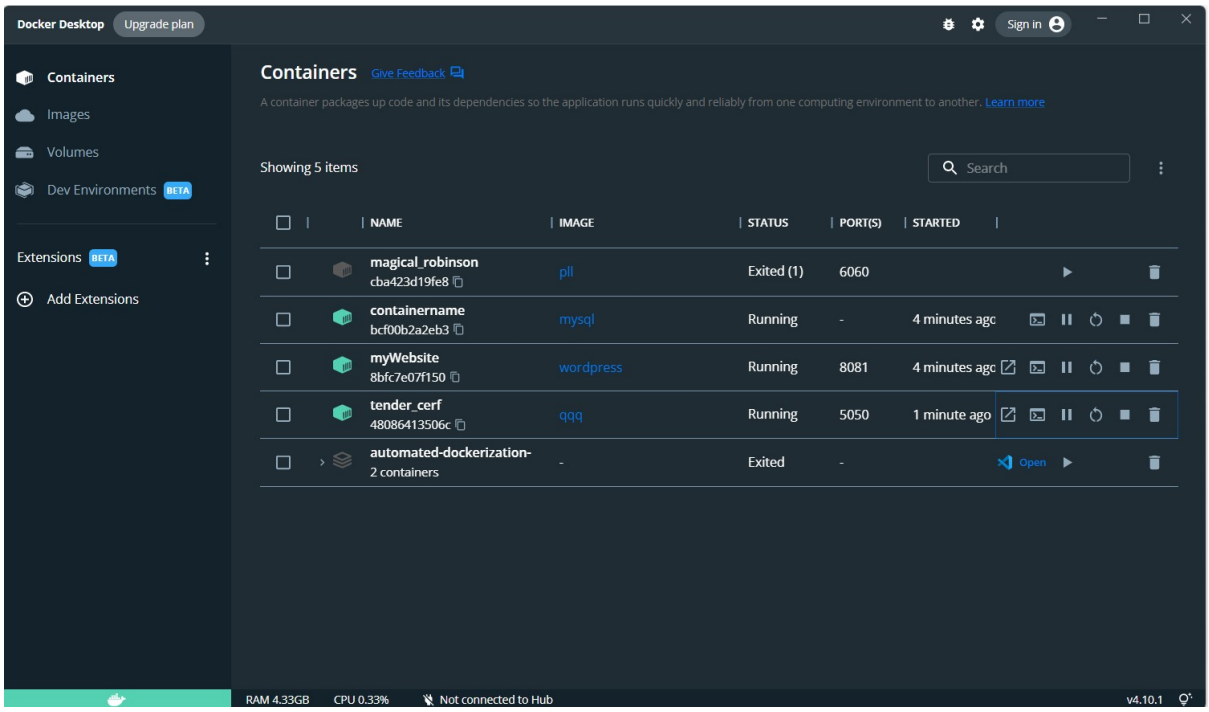

Figure 8: Docker desktop app showcasing the created containers

7 Conclusion and Future Work

The research question that this research paper proposed was "Will using an intuitive user interface and masking the complexities of dockerizing ReactJS, WordPress, and ExpresSJS web applications enhance the adoption rates and the number of dockerized applications?" This research has successfully showed how a simple Python application can be used to put a wrapper around a sophisticated dockerization process in order to generate cloud-deployable artifacts. The automated docerization application is geared mostly toward novice users and developers who are interested in utilizing Docker but have limited knowledge of the topic and are under time pressure to get started as soon as possible. The usage of the approach described above not only enhances usability, but it also boosts the productivity of end developers and users by simplifying deployment using Docker. What this research presented was a little drop in the bucket. This project's ultimate objective is to develop a standard user interface that allows users to deploy any type of application written in any programming language, with the option to switch versions as needed, and with deployment completed in minutes. The following aspects of future work can be highlighted: Possibility to dockerize non-web apps written in other languages, potential to extend the number of supported programming languages, capability to evaluate

Docker, deploy to the container registry, convert the standalone automation application to a web application. The implementation can be enhanced to handle more programming languages and databases. Based on our research, we can conclude that containers have an advantage due to their superior performance and quicker start-up and deployment times. While there are numerous and diverse implementations of containers, each has its own pros and downsides. With Docker, we can also add layers to Linux Containers, which makes them ideally suited for Cloud PaaS services. We envision a bright future for containers in PaaS if abstraction is improved further.

References

Aggarwal, S. (2018). Modern web-development using reactjs, *International Journal of Recent Research Aspects* **5**(1): 133–137.

Bernstein, D. (2014). Containers and cloud: From lxc to docker to kubernetes, *IEEE cloud computing* **1**(3): 81–84.

Bin, N., Zhihua, B., Dejian, L., Sheng, L. and LiXin, Y. (2019). Containerization of intelligent terminal application in ubiquitous power internet of things, *2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)*, IEEE, pp. 1821–1826.

Buyya, R. (2010). Cloud computing: The next revolution in information technology, *2010 First International Conference On Parallel, Distributed and Grid Computing (PDGC 2010)*, IEEE, pp. 2–3.

Cabot, J. (2018). Wordpress: A content management system to democratize publishing, *IEEE Software* **35**(3): 89–92.

Combe, T., Martin, A. and Di Pietro, R. (2016). To docker or not to docker: A security perspective, *IEEE Cloud Computing* **3**(5): 54–62.

Ijtihadie, R. M., Santoso, B. J., Fablius, D. and Nusawan, I. D. P. A. (2017). Multi criteria decision system for distribution provisioning and resource optimization using analytical hierarchy process, *2017 11th International Conference on Information & Communication Technology and System (ICTS)*, IEEE, pp. 197–202.

Javeed, A. (2019). Performance optimization techniques for reactjs, *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, IEEE, pp. 1–5.

Jones, K. M. and Alida-Farrington, P. (2011). Getting started with wordpress, *Library Technology Reports* **47**(3): 8–15.

Kedambadi Shreekar, S. (2020). *Automated Dockerization of Python based Web Apps*, PhD thesis, Dublin, National College of Ireland.

Kumar, A. and Singh, R. K. (2016). Comparative analysis of angularjs and reactjs, *International Journal of Latest Trends in Engineering and Technology* **7**(4): 225–227.

Merkel, D. et al. (2014). Docker: lightweight linux containers for consistent development and deployment, *Linux j* **239**(2): 2.

Naik, N. (2016). Building a virtual system of systems using docker swarm in multiple clouds, *2016 IEEE International Symposium on Systems Engineering (ISSE)*, IEEE, pp. 1–3.

Pahl, M.-O. and Carle, G. (2013). The missing layer—virtualizing smart spaces, *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, IEEE, pp. 139–144.

Peters, C. (2017). Building rich internet applications with node. js and express. js, *Rich Internet Applications w/HTML and Javascript* p. 15.

Rad, B. B., Bhatti, H. J. and Ahmadi, M. (2017). An introduction to docker and analysis of its performance, *International Journal of Computer Science and Network Security (IJCSNS)* **17**(3): 228. JCR Impact Factor 2017: 1.5 Impact Factor.

Rajavaram, H., Rajula, V. and Thangaraju, B. (2019). Automation of microservices application deployment made easy by rundeck and kubernetes, *2019 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, IEEE, pp. 1–3.

Tao, D., Lin, Z. and Lu, C. (2015). Cloud platform based automated security testing system for mobile internet, *Tsinghua Science and Technology* **20**(6): 537–544.

Teigland, D. and Mauelshagen, H. (2001). Volume managers in linux., *USENIX Annual Technical Conference, FREENIX Track*, pp. 185–197.

Tomiša, M., Milković, M. and Čačić, M. (2019). Performance evaluation of dynamic and static wordpress-based websites, *2019 23rd International Computer Science and Engineering Conference (ICSEC)*, pp. 321–324.

Tushev, V. (2022). The Best React Websites Examples That Ever Built [2022] | ProCoders. [Online; accessed 2022-08-15].