

Bachelor's thesis

Degree Programme in Information Technology

2019

Mala Shrestha

EXPLORING DOCKER IMPLEMENTATION WITH WORDPRESS



BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology

2019 | 47 | 7

Mala Shrestha

EXPLORING DOCKER IMPLEMENTATION WITH WORDPRESS

The aim of the thesis was to gain hands on experience on software containerization by exploring an open source software development platform, Docker and its implementation with a use case: WordPress.

The theoretical part of this thesis highlights the important concepts of containerization, some core components of Docker and the reason behind the usage of Docker as a popular platform. Similarly, a short overview of WordPress was given along with the explanation of WordPress image workflow with Docker.

The practical part of the thesis deals with the implementation processes. The testing was done in Windows operating system which included the installation of Docker in the beginning phase and implementation with WordPress in the later.

As a result, Docker was successfully installed. However, some errors were encountered during the installation process, which was solved with the help of available references and online materials. Thereafter, the implementation with WordPress was successfully done. The objective of learning about containerization with Docker was met.

The result was a useful experience of creating a container using one of the growing technologies: Docker, which is beneficial for both developers and system administrators to build, test and deploy applications quickly.

KEYWORDS:

Docker, containerization, wordpress, software, engineering, container images

CONTENT

LIST OF ABBREVIATIONS (OR) SYMBOLS	5
1 INTRODUCTION	6
2 DOCKER OVERVIEW	8
2.1 Docker Containers vs. Virtual Machines	8
2.2 Docker Engine	10
2.3 Docker Objects	12
3 IMPLEMENTATION OF DOCKER	14
3.1 Docker for Windows	14
3.2 Docker Toolbox Overview	15
3.3 Installation on Windows 10	17
4 IMPLEMENTATION WITH WORDPRESS	27
4.1 WordPress Overview	27
4.2 Creating a WordPress Container Image	27
4.3 Running WordPress with Docker Compose	31
5 CONCLUSION	36
REFERENCES	38

APPENDICES

Appendix 1. Configuring IP Address

Appendix 2. Output of Command 14 and 15 respectively

FIGURES

Figure 1. Docker Official Logo (DwGlogo - Stunning free images, 2017).	8
Figure 2. Layout of Docker Containers Vs. Virtual Machines (Image.slidesharecdn.com, 2016).	9
Figure 3. Structure of Docker Engine (Docker Documentation, 2019)	11
Figure 4. Home Screen of Kitematic.	15
Figure 5. Example of docker-compose.yml file (Docker Documentation, 2019).	16
Figure 6. Docker Toolbox for Windows Installation page (Docker Documentation, 2019).	17
Figure 7. Docker Toolbox Setup Wizard.	18
Figure 8. Completion of Setup process.	18
Figure 9. Docker Tools shortcut keys on desktop.	19
Figure 10. IP issue while launching Docker Terminal.	20
Figure 11. Successfully launched Docker Terminal.	21
Figure 12. Checking installation validity with Command 2.	22
Figure 13. Checking installation validity with Command 3.	22
Figure 14. Certification error while pulling docker image.	23
Figure 15. Troubleshooting Error.	23
Figure 16. Running <i>hello-world</i> container.	24
Figure 17. Downloading an image.	25
Figure 18. Getting Container ID.	25
Figure 19. Running <i>hello-world</i> container on the localhost.	26
Figure 20. Output of Command 12.	29
Figure 21. Output of Command 13.	29
Figure 22. Active Containers.	30
Figure 23. WordPress installation webpage.	30
Figure 24. docker-compose.yml file.	31
Figure 25. Output of Command 16.	32
Figure 26. WordPress installation webpage.	33
Figure 27. Docker Compose configuration for Production server.	34

LIST OF ABBREVIATIONS (OR) SYMBOLS

ADP	Automatic Data Processing
API	Application ProgrammingInterface
AWS	Amazon Web Services
CentOS	Community Enterprise Operating System
CLI	Command Line Interface
CMS	Content Management System
CPU	Central Processing Unit
GE	General Electric
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines
IP	Internet Protocol
IT	Information Technology
LXC	Linux Containers
MacOS	Macintosh Operating System
OS	Operating System
PC	Personal Computer
REST	Representational State Transfer
SLAT	Second Level Address Translation
URL	Uniform Resource Locator
VM	Virtual Machine
VMM	Virtual Machine Monitor
WAMP	Windows Apache MySQL PHP
XAMPP	Cross-Platform, Apache, MySQL, PHP and Perl
YAML	Yet Another Markup Language

1 INTRODUCTION

The main idea of virtualization is to run multiple independent systems potentially on a single piece of hardware. “The concept of virtualization is generally believed to have its origins in the mainframe days in the late 1960s and early 1970s, when IBM invested a lot of time and effort in developing robust time-sharing solutions” (Docs.oracle.com, 2012 a). The timeline of virtualization development can be overviewed from https://en.wikipedia.org/wiki/Timeline_of_virtualization_development?fbclid=IwAR18JN2kfZ1PYTGCB6jb6ZBS4_dd_8SrbMz2FiJdRpFtN1vYyPrN-6sfuY.

Hypervisor, also called as Virtual Machine Monitor (VMM) is an essential component of virtualization. It creates a virtual platform on the host computer and multiple operating systems are executed on top of it. This enables the operating systems to share the hardware resources offered by the host. (Docs.oracle.com, 2012 b.) Some of the hypervisors are Oracle VM, Microsoft Hyper-V, VMWare Server, VMWare ESX and Xen.

The key properties of virtualization are running multiple systems on a single machine and separating system resources between VMs, providing security isolation, encapsulation and hardware independence. Thus, virtualization has benefited all size businesses by reducing IT expenses while increasing efficiency, productivity and agility. (VMWare, 2019.)

Despite of the feasibility of virtualization, it requires a huge amount of storage and more resources of the server due to its isolated system whereas containers require far less resources of the server, as a container shares its OS kernel with other containers running in a single operating system.

Containerization is a lightweight alternative to virtualization to run an application more efficiently with its dependencies. However, containers cannot replace virtual machines fully, for the older programs which were designed to run on a virtual machine cannot run well in container technology (DeMuro, 2018). Since, containers consist of an application along with all its dependencies, libraries, binaries and configuration files set into a package, it has been beneficial for developers to run the applications reliably when moving from one computing environment to the other. For example: “from a developer's laptop to a test environment, from a staging environment into production, and perhaps

from a physical machine in a data center to a virtual machine in a private or public cloud” (Rubens, 2017).

Even though, container technology has been in the LXC format for over 10 years, it has gained popularity with Docker after the release of Docker 1.0 in June 2014. Docker, as an open source platform, enables developers to easily pack, ship and run any application as a lightweight, portable and a self-sufficient container that can also be deployed in a cloud (Vaughan-Nichols, 2018).

Docker runs on any Linux distribution running version 3.10 or later of the Linux kernel. In 2016, Microsoft introduced Docker containers designed for Windows, which can be managed from any Docker client or Microsoft’s PowerShell. Therefore, Docker can also be run on Windows Server 2016 and Windows 10. In addition to that, Docker also runs on popular cloud platforms like Amazon EC2, Google Compute Engine, Rackspace and Microsoft Azure. Containers are believed to be less secure than virtual machines as, they share the same host kernel. With this issue, Docker has included a signing infrastructure which allows administrators to sign container images for preventing untrusted containers from being deployed. Also, for more security Docker has provided container security scanning solutions. This helps administrators to detect vulnerabilities in container images that could be exploited. (Rubens, 2017.) Due to these benefits and services, Docker is “used by millions of developers and more than 650 Global 10K commercial customers including ADP, GE, MetLife, PayPal and Societe Generale” (Docker, 2019 a).

This thesis shows the important concepts of containerization with Docker and its implementation with WordPress. Chapter 1 introduces the background of container technology, its recent popularity with Docker and the chapters of the thesis. Chapter 2 gives the overview of Docker, its comparison with virtual machines and some of its important topics. Chapter 3 shows the installation process of Docker on Windows and Chapter 4 deals with the implementation of Docker with WordPress image container. Altogether, the thesis highlights the reason behind the trending use of Docker as well as gives the practical experience of the use case.

2 DOCKER OVERVIEW

Docker is an open source software platform, launched in March 2013 by Docker Inc. to build, ship and run applications in a container technology. Docker provides a unique technology by separating application dependencies from infrastructure which satisfies both system operators and developers. The official logo of Docker is shown in Figure 1.



Figure 1. Docker Official Logo (DwGlogo - Stunning free images, 2017).

Docker containers are lightweight as, they share the host OS kernel and therefore, multiple containers can run on the same machine in an isolated process. This requires less space than VMs and comparably, more portable and efficient as well. Docker containers are available for both Linux and Windows-based applications. Since, containers isolate software from its environment, it works uniformly during development and staging, regardless of the infrastructure (Docker, 2019 b). Therefore, Docker provides “freedom to build, manage and secure business-critical applications without the fear of technology or infrastructure lock-in” (Docker, 2019 c).

2.1 Docker Containers vs. Virtual Machines

Docker containers are lightweight, stand-alone, executable package of software with all necessary dependencies while virtual machines are virtual servers that emulate the hardware server. An entire operating system is required for virtual machines but in case of Docker containers, the parts of the OS and the kernel are shared. Therefore, compared to virtual machines, Docker containers work more efficiently with a lower overhead and a lot more containers can be packed onto a single server. Containers are

megabytes in size and take few seconds to start whereas, VMs are gigabytes in size and can take minutes to start.

Containers vs. VMs

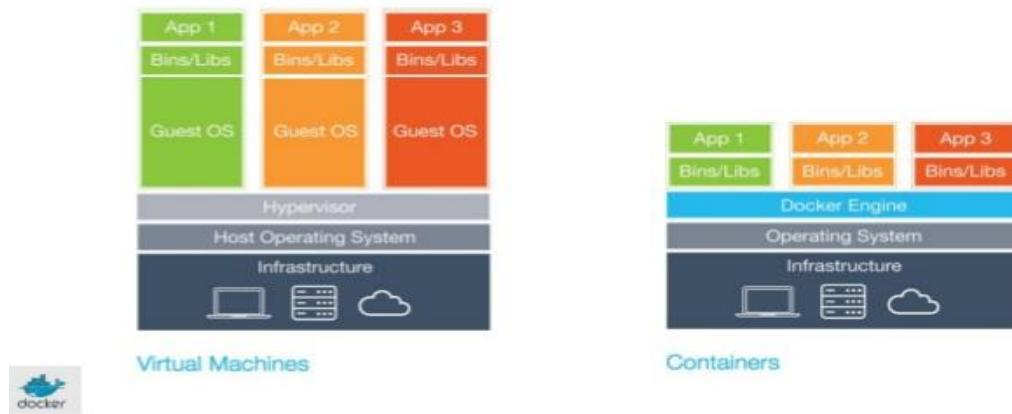


Figure 2. Layout of Docker Containers Vs. Virtual Machines (Image.slidesharecdn.com, 2016).

As shown in Figure 2, virtual machines need three separate operating systems to run three applications on a server in total isolation whereas, Docker containers share the common OS with the help of Docker engine. Hypervisor is used in case of virtual machines to control three guest operating systems on top of which various binaries and libraries are run. Each guest OS needs its own copy of dependencies and each needs its own CPU and memory resources as well. In case of Docker containers, all the necessary dependencies are built into special packages called Docker images which are run by Docker daemon. Each application is managed by Docker daemon and is still isolated. (Janetakis, 2017.) Therefore, Docker containers benefit more than VMs in the following ways:

- Docker containers are lightweight than virtual machines, therefore, more containers can run on a host machine.
- Docker containers can be created and destroyed quickly and easily while virtual machines need full installation and require more resources to execute (Access.redhat.com, 2019).
- With Docker containers, applications are portable across machines without compatibility issues.
- Docker containers can be shared using a remote repository.

- Docker containers also benefit with version control and component reuse (Access.redhat.com, 2019).

However, Docker containers cannot replace virtual machines fully for all use cases. A careful evaluation is needed before choosing the right one depending on the application requirements.

Docker also has few drawbacks as mentioned below:

- Docker has a complicated feature than a virtual machine due to its tooling ecosystem, which is managed by both Docker and third-party tools (Aquasec, 2018 a).
- When an application is run directly on a bare-metal server, a true bare-metal speed is achieved even without any use of containers or virtual machines. Docker containers have less overhead but not zero overhead, therefore; they do not run at bare-metal speed. (DataflairTeam, 2018 a.)
- Docker containers have an issue with cross-platform compatibility. For example: if an application is run in a Docker container on Linux, then it cannot run on Windows or vice versa.
- Since, there are frequent upgrades in new technologies, Docker cannot be an exception. So, there is a hassle of updating new versions frequently.

2.2 Docker Engine

Docker Engine is the core component of the Docker platform. It is a client-server application that allows to develop, bundle, ship and run applications with the help of its major components (as shown in Figure 3) like a daemon, REST API and CLI Client.

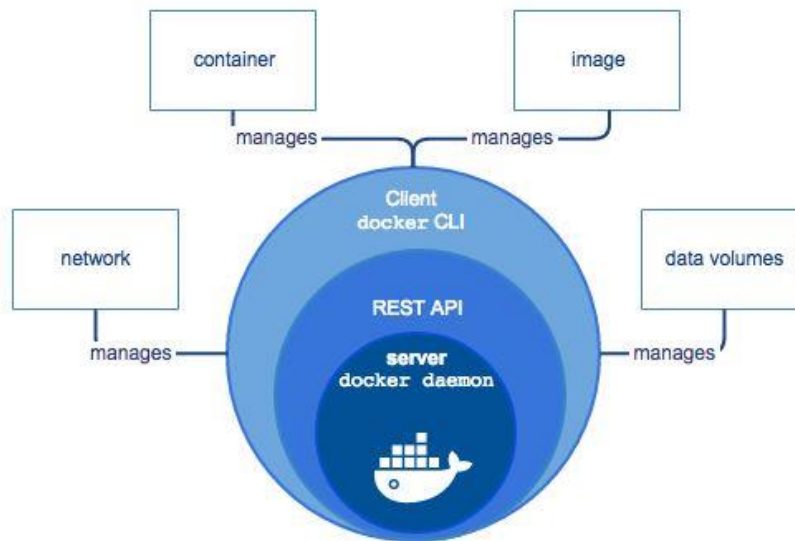


Figure 3. Structure of Docker Engine (Docker Documentation, 2019 a)

- Docker Daemon: It is a server; a type of long-running program called a daemon process (the `dockerd` command) which manages and creates Docker objects, like containers, images, volumes and network (DataflairTeam, 2018 b).
- REST API: It “specifies interfaces that programs can use to talk to the daemon and instruct it what to do” (Docker Documentation, 2019 a). The REST API can be accessed by an HTTP client.
- Docker CLI: It is a command line interface (CLI) client (the `docker` command) that uses the Docker REST API to interact with the Docker daemon through scripting or direct CLI commands (Docker Documentation, 2019 a).

Due to these features, container instances can be managed in a simple way, which makes Docker popular among developers.

2.3 Docker Objects

An application is assembled using various objects in a Docker container. Some of the main requisite Docker objects are as follows:

- Images: Images are a read-only template with instructions to build containers. They are used to store and ship applications. Usually, an image is based on another image that can be customized to add additional elements for the extension of configuration (Aquasec, 2018 a). For example, if an image is built based on the Ubuntu image, the Apache web server and the application along with the configuration details are needed to be installed to run the application (Docker Documentation, 2019 a).

Docker images can be created on own or can be used those created by others and published in a registry. In order to create own image, Dockerfile is created with a simple syntax for creating the image and run. Every step defined in a Dockerfile creates a layer in the image. When the Dockerfile is changed or the image is rebuilt, only those changed layers are rebuilt. Due to this, images are way more lightweight, fast and small in comparison to virtualization technologies (Docker Documentation, 2019 a).

- Containers: Containers are runnable instances of an image that can be created, started, stopped, moved or deleted by using the Docker API or CLI (Docker Documentation, 2019 a). Docker containers are isolated from other containers and its hos machine. They are “defined by the image and any additional configuration options provided on starting the container, including and not limited to the network connections and storage options” (Aquasec, 2018 b). Any changes made to a container without storing them are deleted along with the deletion of it.
- Storage: Data can be stored within the writable layer of a container in the form of persistent storage. In this case, Docker provides four options like data volumes, data volume container, storage plugins and directory mounts.

Data volumes provide the ability to list and rename volumes and also, list the container associated with the volume. In a data volume container, container hosts a volume and mount that to other containers. Since, the volume container is independent of the application container, it can be shared among the containers. Storage plugins connect the external storage platforms by mapping from the host to external sources like storage array or an appliance. Some examples of storage plugins are NetApp, HPE 3PAR and Google Compute Platform. (Aquasec, 2018 a.)

3 IMPLEMENTATION OF DOCKER

Docker can be implemented with a wide range of platforms as follows:

- Desktop: Windows 10, MacOS.
- Cloud: AWS, Microsoft Azure, IBM Cloud, Google Compute Platform and more.
- Server: Windows Server 2016 and various Linux distributions like CentOS, Debian, Ubuntu, Fedora and more.

This thesis is based on the implementation with Windows OS.

3.1 Docker for Windows

Microsoft Windows has introduced the Community Edition (CE) of Docker as 'Docker Desktop for Windows'. This can be downloaded from Docker Hub. However, there are some few things to be noted before the installation. Since, Microsoft Hyper-V is required to run for Docker Desktop for Windows, the Docker Desktop for Windows Installer can enable Hyper-V and can restart the machine. When Hyper-V is enabled, VirtualBox does not work, but any VirtualBox VM images remain. Also, VMs created with **docker-machine** no longer start and cannot be used alongside Docker Desktop for Windows. Instead, the remote VMs can be managed by using **docker-machine**. The installer includes Docker Engine, Docker Compose, Docker Machine, Docker CLI client and Kitematic. (Docker Documentation, 2019 c.)

The system requirements for Docker Desktop for Windows are mentioned below:

- Windows 10 64bit: Pro, Enterprise or Education (1607 Anniversary Update, Build 14393 or later) (Docker Documentation, 2019 c).
- RAM: 4GB minimum.
- CPU SLAT- capable feature.
- Virtualization enabled BIOS.

In case of the system not meeting these requirements or for previous versions, Docker Toolbox should be installed. Instead of Hyper-V, Docker Toolbox uses Oracle Virtual Box.

3.2 Docker Toolbox Overview

Docker Toolbox is a solution for older versions of Windows and Mac OS that do not meet the requirements for Docker Desktop for Windows and Docker Desktop for Mac. This installer provides the quick setup and launch of a Docker environment.

Docker Toolbox is a package of different Docker tools like Docker Engine, Docker Machine, Docker Compose, Kitematic (the Docker GUI), a shell preconfigured for a Docker command-line environment and Oracle VirtualBox (Docker Documentation, 2019 b). Some of the tools are explained in brief in the following ways:

Kitematic

It is an open source project to automate the installation and setup process of Docker. It provides GUI for running Docker containers. It helps to install the Docker Engine locally on the machine by integrating with Docker Machine to provision a VirtualBox VM.

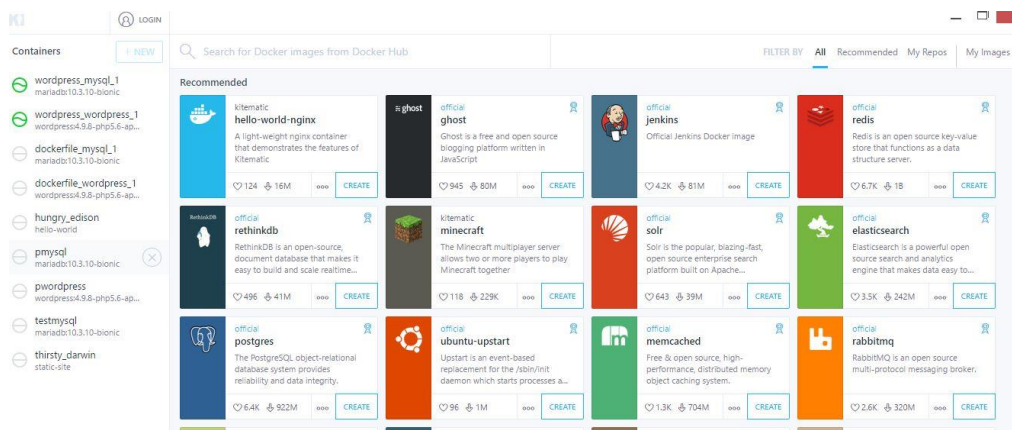


Figure 4. Home Screen of Kitematic.

After installing and launching the Kitematic GUI, number of public images can be seen on the home screen (as shown in Figure 4). Also, any images can be searched on Docker Hub from Kitematic. Kitematic GUI helps to create, run and manage containers. Some other uses of Kitematic are managing ports and configuring volumes, changing environment variables, streaming logs and single click terminal (Docker Documentation, 2019 e).

Docker Machine

It is used to install Docker Engine on virtual hosts and create Docker hosts on Mac or Windows, network, data center, or on cloud. It manages the hosts with `docker-machine` commands. Various operations like starting, stopping, inspecting and restarting a managed host can be done using `docker-machine` commands. The Docker client and daemon can also be upgraded configured using `docker-machine` commands.

For example: To point to a host called 'default', Command 1 is used.

```
$ docker-machine env default (1)
```

Docker Compose

It is a tool for defining and running container applications. The configuration of the application's services is done by creating a YAML file. A single compose command, then creates and starts all the services from that YAML file. For Docker Compose, first Dockerfile is defined for the application environment to reproduce anywhere. Then, `docker-compose.yml` file is created by defining all the services for the application. Finally, with `docker-compose up` command, the entire application is run. (Docker Documentation, 2019 f.)

An example of `docker-compose.yml` is shown in Figure 5.

```
version: '3'
services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - ./code
      - logvolume01:/var/log
    links:
      - redis
  redis:
    image: redis
volumes:
  logvolume01: {}
```

Figure 5. Example of `docker-compose.yml` file (Docker Documentation, 2019 f.).

Docker Compose has commands for starting, stopping and rebuilding services, viewing the status of running services, running a one-off command on a service and streaming the log output of running services (Docker Documentation, 2019 f).

3.3 Installation on Windows 10

Since, the installation for this thesis was done in the older version of Windows 10, Docker Toolbox was needed to be installed (as discussed in Chapter 3,3.1). The following steps were carried out in order to install Docker Toolbox:

1. First, Docker Toolbox for Windows installer was accessed from the site https://docs.docker.com/toolbox/toolbox_install_windows/ as shown in Figure 6.



Figure 6. Docker Toolbox for Windows Installation page (Docker Documentation, 2019 d).

2. When the installer was clicked, it launched the "Setup – Docker Toolbox" dialog as appeared in Figure 7.

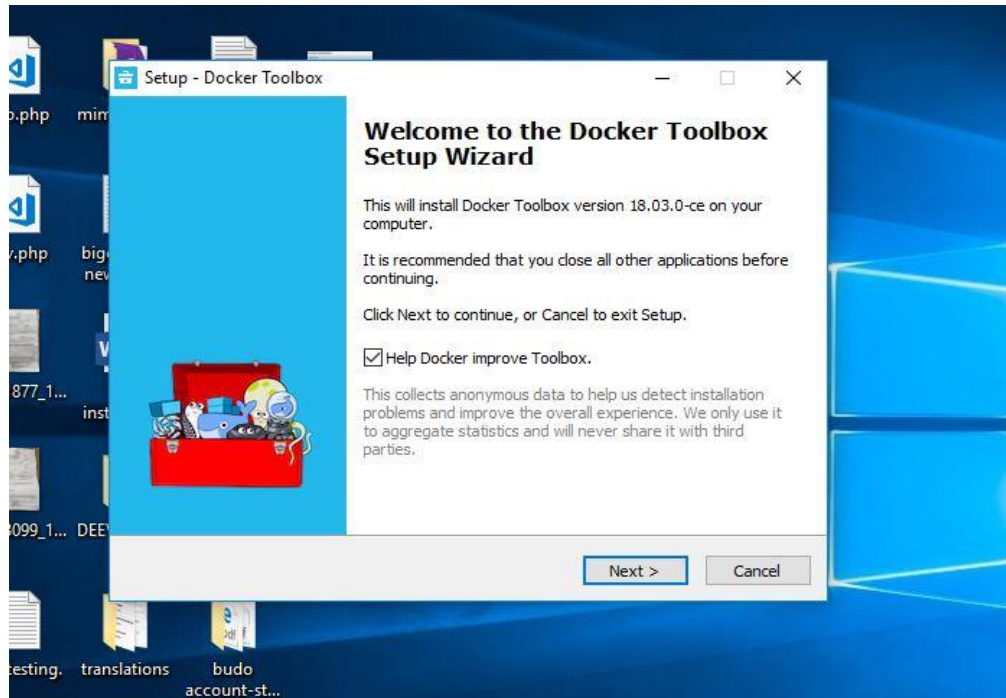


Figure 7. Docker Toolbox Setup Wizard.

3. By clicking the "Next" button and accepting all the installer defaults, the Setup wizard then, reported the successful setup process as displayed in Figure 8 and the button "Finish" was pressed.

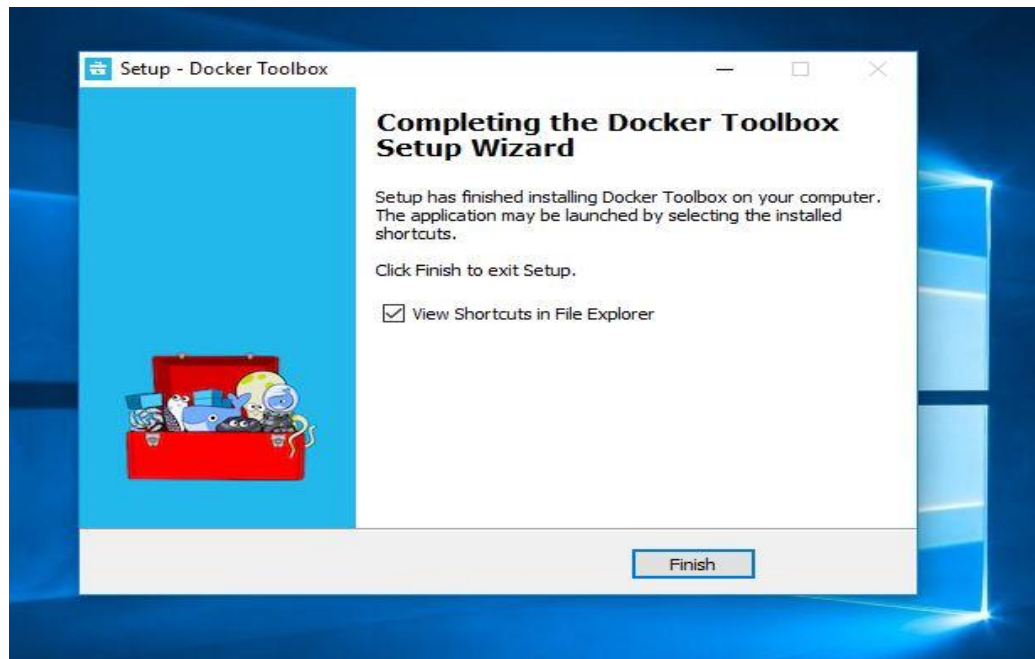


Figure 8. Completion of Setup process.

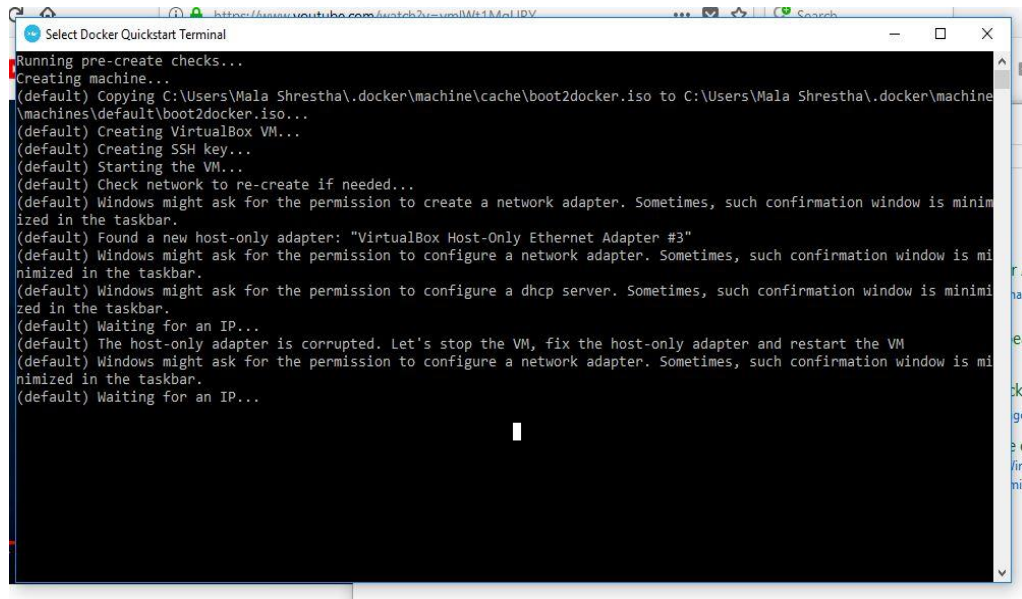
4. The installer added Docker Quickstart Terminal, VirtualBox and Kitematic to the folder along with the shortcut keys on the desktop as displayed in Figure 8.



Figure 9. Docker Tools shortcut keys on desktop.

Getting started with Docker Terminal:

5. Docker Terminal was tried to launch from the desktop shortcut 'Docker Quickstart Terminal'. The terminal did several things to set up Docker toolbox but it got stuck with an IP issue as shown in Figure 10.



```

Select Docker Quickstart Terminal
Running pre-create checks...
Creating machine...
(default) Copying C:\Users\Mala Shrestha\.docker\machine\cache\boot2docker.iso to C:\Users\Mala Shrestha\.docker\machine\machines\default\boot2docker.iso...
(default) Creating VirtualBox VM...
(default) Creating SSH key...
(default) Starting the VM...
(default) Check network to re-create if needed...
(default) Windows might ask for the permission to create a network adapter. Sometimes, such confirmation window is minimized in the taskbar.
(default) Found a new host-only adapter: "VirtualBox Host-Only Ethernet Adapter #3"
(default) Windows might ask for the permission to configure a network adapter. Sometimes, such confirmation window is minimized in the taskbar.
(default) Windows might ask for the permission to configure a dhcp server. Sometimes, such confirmation window is minimized in the taskbar.
(default) Waiting for an IP...
(default) The host-only adapter is corrupted. Let's stop the VM, fix the host-only adapter and restart the VM
(default) Windows might ask for the permission to configure a network adapter. Sometimes, such confirmation window is minimized in the taskbar.
(default) Waiting for an IP...

```

Figure 10. IP issue while launching Docker Terminal.

Configuring IP address:

In order to solve the IP address issue, PC terminal was first opened to check the IP address for the configuration which was **Network #3 : 192.168.99.1** for this test. The command entered was **ipconfig** and the output can be seen in Appendix 1. Then, by opening Windows Firewall Defender of PC, IP address was configured by following the steps as shown in Appendix 1.

6. After successful configuration of IP address, Docker Terminal was run again. This time, it was launched successfully that can be seen in Figure 11. In case of Docker, instead of the standard Windows command prompt, the terminal runs a bash environment with the prompt sign **\$** (a dollar sign). Commands are typed after the dollar sign.

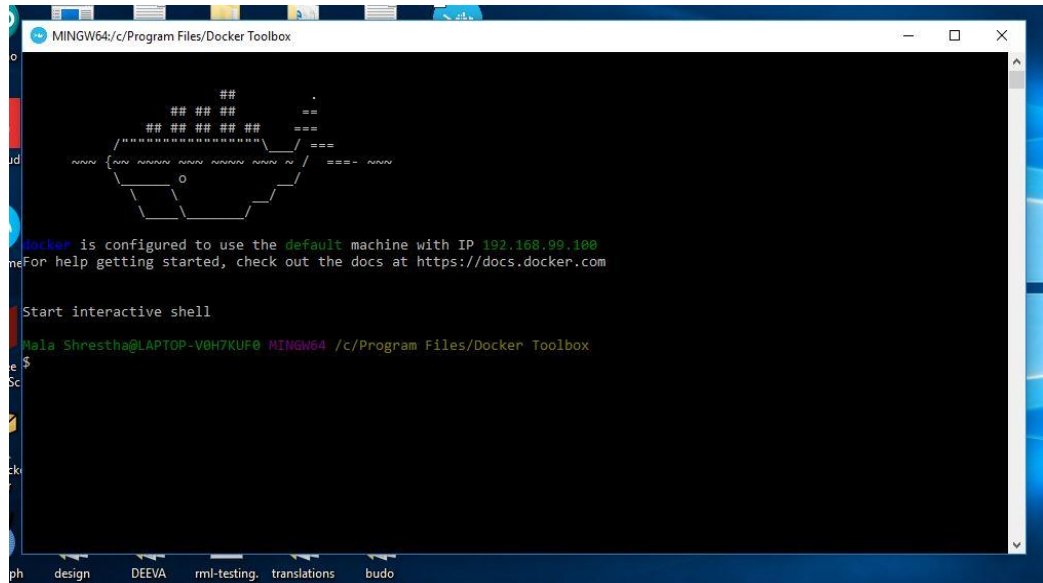


Figure 11. Successfully launched Docker Terminal.

Some random commands (Command 2 and Command 3) were passed to test if the installation was complete. The result was successful (as shown in Figure 12 and Figure 13) respectively.

`$ docker` – lists some management commands. (2)

`$ docker -version` – shows the Docker version information. (3)

```

Mala Shrestha@LAPTOP-V0H7KUF0 MINGW64 /c/Program Files/Docker Toolbox
$ docker

Usage:  docker COMMAND

A self-sufficient runtime for containers

Options:
  --config string          Location of client config files (default
                           "C:\\Users\\Mala Shrestha\\.docker")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level
                           ("debug"|"info"|"warn"|"error"|"fatal")
                           (default "info")
  --tls                    Use TLS; implied by --tlsverify
  --tlscacert string       Trust certs signed only by this CA (default
                           "C:\\Users\\Mala Shrestha\\.docker\\ca.pem")
  --tlscert string         Path to TLS certificate file (default
                           "C:\\Users\\Mala Shrestha\\.docker\\cert.pem")
  --tlskey string          Path to TLS key file (default "C:\\Users\\Mala
                           Shrestha\\.docker\\key.pem")
  --tlsverify              Use TLS and verify the remote
  -v, --version            Print version information and quit

Management Commands:
  checkpoint  Manage checkpoints
  config      Manage Docker configs
  container   Manage containers
  image       Manage images
  network     Manage networks
  node        Manage Swarm nodes
  plugin      Manage plugins
  secret      Manage Docker secrets
  service     Manage services
  swarm       Manage Swarm
  system      Manage Docker

```

Figure 12. Checking installation validity with Command 2.

```

push          Push an image or a repository to a registry
rename        Rename a container
restart       Restart one or more containers
rm            Remove one or more containers
rmi           Remove one or more images
run           Run a command in a new container
save          Save one or more images to a tar archive (streamed to STDOUT by default)
search        Search the Docker Hub for images
start         Start one or more stopped containers
stats         Display a live stream of container(s) resource usage statistics
stop          Stop one or more running containers
tag           Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
top           Display the running processes of a container
unpause       Unpause all processes within one or more containers
update        Update configuration of one or more containers
version       Show the Docker version information
wait          Block until one or more containers stop, then print their exit codes

Run 'docker COMMAND --help' for more information on a command.

Mala Shrestha@LAPTOP-V0H7KUF0 MINGW64 /c/Program Files/Docker Toolbox
$ docker --version
Docker version 18.03.0-ce, build 0520e24302

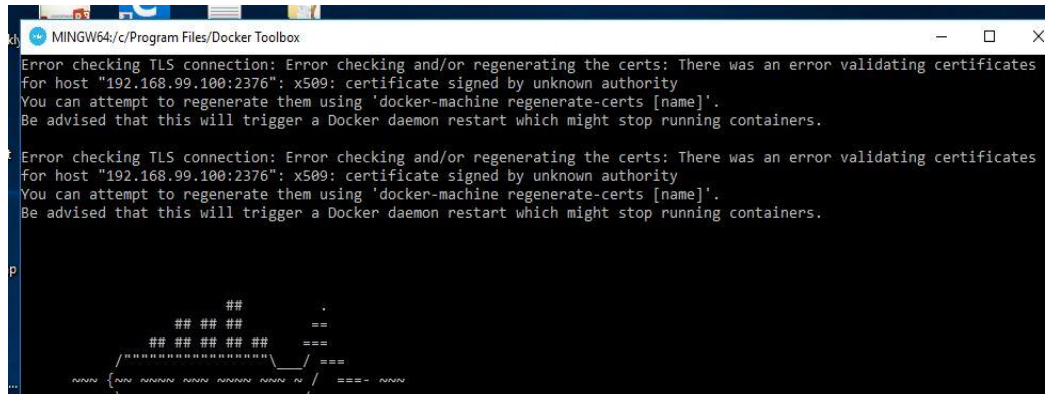
Mala Shrestha@LAPTOP-V0H7KUF0 MINGW64 /c/Program Files/Docker Toolbox
$

```

Figure 13. Checking installation validity with Command 3.

Running hello-world image

- In order to run the first container, "hello-world", it was tried to be pulled from Docker Hub but an error was encountered for certification validation as seen in Figure 14. However, the error was troubleshooted by executing Command 4 and 5 respectively (shown in Figure 15).



```

MINGW64/c/Program Files/Docker Toolbox
Error checking TLS connection: Error checking and/or regenerating the certs: There was an error validating certificates
for host "192.168.99.100:2376": x509: certificate signed by unknown authority
You can attempt to regenerate them using 'docker-machine regenerate-certs [name]'.
Be advised that this will trigger a Docker daemon restart which might stop running containers.

Error checking TLS connection: Error checking and/or regenerating the certs: There was an error validating certificates
for host "192.168.99.100:2376": x509: certificate signed by unknown authority
You can attempt to regenerate them using 'docker-machine regenerate-certs [name]'.
Be advised that this will trigger a Docker daemon restart which might stop running containers.

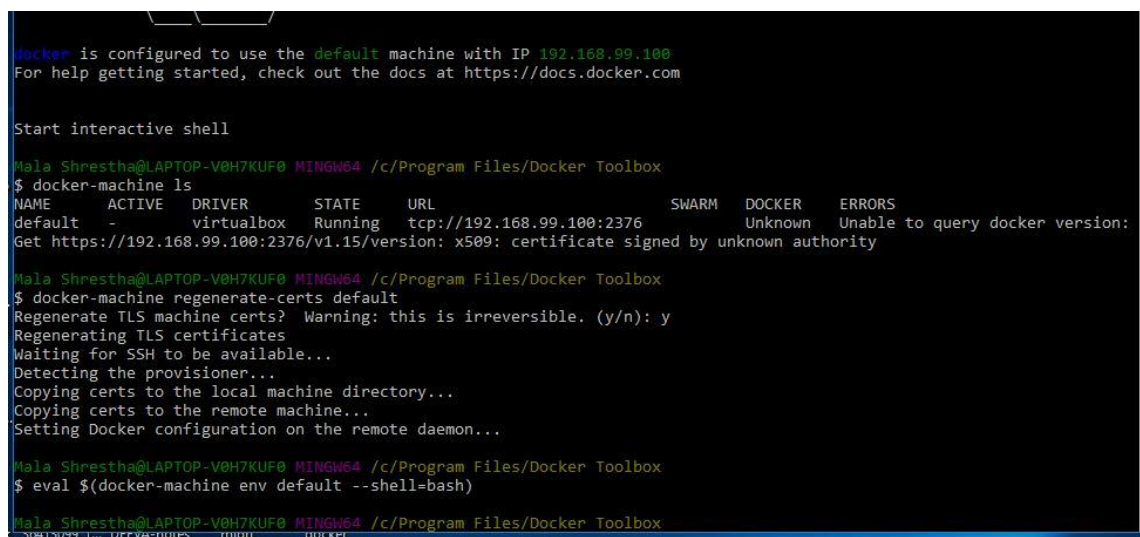
```

Figure 14. Certification error while pulling docker image.

Troubleshooting error:

`$ docker-machine ls` – lists the status of the virtual machine and ip address. (4)

`$ docker-machine regenerate-certs default` – regenerates the certificates. (5)



```

Docker is configured to use the default machine with IP 192.168.99.100
For help getting started, check out the docs at https://docs.docker.com

Start interactive shell
Mala Shrestha@LAPTOP-V0H7KUF0 MINGW64 /c/Program Files/Docker Toolbox
$ docker-machine ls
NAME      ACTIVE   DRIVER        STATE     URL                         SWARM   DOCKER  ERRORS
default   -        virtualbox    Running   tcp://192.168.99.100:2376   Unknown Unable to query docker version:
Get https://192.168.99.100:2376/v1.15/version: x509: certificate signed by unknown authority

Mala Shrestha@LAPTOP-V0H7KUF0 MINGW64 /c/Program Files/Docker Toolbox
$ docker-machine regenerate-certs default
Regenerate TLS machine certs? Warning: this is irreversible. (y/n): y
Regenerating TLS certificates
Waiting for SSH to be available...
Detecting the provisioner...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...

Mala Shrestha@LAPTOP-V0H7KUF0 MINGW64 /c/Program Files/Docker Toolbox
$ eval $(docker-machine env default --shell=bash)
Mala Shrestha@LAPTOP-V0H7KUF0 MINGW64 /c/Program Files/Docker Toolbox

```

Figure 15. Troubleshooting Error.

8. Finally, after solving the error, the "hello-world" container was run again by Command 6 and the container was created successfully as an output in Figure 16. Eventhough, the image was not available beforehand, it was pulled from the library automatically and downloaded in the repository which is the best part of working with Docker images. Then, the downloaded image properties was also checked by Command 7.

```
$ docker run hello-world (6)
```

```
$ docker images hello-world (7)
```

```
Mala Shrestha@LAPTOP-V0H7KUF0 MINGW64 /c/Program Files/Docker Toolbox
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
d1725b59e92d: Pull complete
Digest: sha256:0add3ace90ecb4adbf777e9aacf18357296e799f81cabc9fde470971e499788
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

Mala Shrestha@LAPTOP-V0H7KUF0 MINGW64 /c/Program Files/Docker Toolbox
$ docker images hello-world
REPOSITORY          TAG             IMAGE ID         CREATED          SIZE
hello-world         latest         4ab4c602aa5e   7 weeks ago     1.84kB
```

Figure 16. Running *hello-world* container.

9. The best way to check the running image was to run it in the site which could be done by downloading (as shown in Figure 17) sequence/static-site with Command 8.

```
$ docker run -d -P sequence/static-site (8)
```

```

Mala Shrestha@LAPTOP-V0H7KUF0 MINGW64 /c/Program Files/Docker Toolbox
$ docker run -d -P sequence/static-site
Unable to find image 'sequence/static-site:latest' locally
latest: Pulling from sequence/static-site
fdd5d7827f33: Pull complete
a3ed95caeb02: Pull complete
716f7a5f3082: Pull complete
7b10f03a0309: Pull complete
aff3ab7e9c39: Pull complete
Digest: sha256:41b286105f913fb7a5fbdce28d48bc80f1c77e3c4ce1b8280f28129ae0e94e9e
Status: Downloaded newer image for sequence/static-site:latest
b530b3c539355d88ee8dd940fec626dcaa9bf9f4fd2e9963b2ffdd2d720336c5

```

Figure 17. Downloading an image.

The next step was to get the port for running the image on the site. It was done with Command 9 and the output can be seen in Figure 18. The result was an **id :32769** which was the random id for the PC used in this test and it could be different in different PCs. Then, the same id was run on the localhost that displayed the *hello-world* container as shown in Figure 19.

`$ docker ps` (9)

```

Mala Shrestha@LAPTOP-V0H7KUF0 MINGW64 /c/Program Files/Docker Toolbox
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STA
TUS                PORTS
b530b3c53935      sequence/static-site  "/bin/sh -c 'cd /usr..."  5 minutes ago      Up
5 minutes         0.0.0.0:32769->80/tcp, 0.0.0.0:32768->443/tcp  thirsty_darwin
Mala Shrestha@LAPTOP-V0H7KUF0 MINGW64 /c/Program Files/Docker Toolbox

```

Figure 18. Getting Container ID.

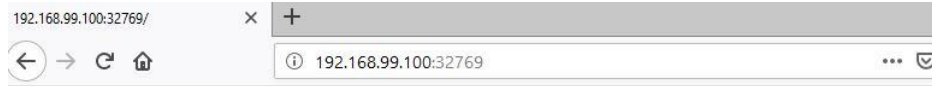


Figure 19. Running *hello-world* container on the localhost.

4 IMPLEMENTATION WITH WORDPRESS

Since, Docker is the trendiest technology to offer container-based development and services, CMS like WordPress is using Docker for application deployment as well as development processes. Normally, to run WordPress locally, additional configurations like Apache, MySQL server, XAMPP, WAMP and more are required whereas, Docker containers contain everything required to run an application. Similarly, Docker works the same everywhere irrespective of the server and OS.

This section of thesis covers the short overview of WordPress and the process of setting up a complete WordPress appliance using Docker.

4.1 WordPress Overview

WordPress is a free open source CMS based on PHP and MySQL to create a website, application or blog. It runs on a web hosting service. It was started in 2003 by Mike Little and Matt Mullenweg by creating a fork of b2/cafeblog (WordPress.org, 2019). It is the most popular blogging tool in use, at more than 60 million websites. The main features of WordPress include a plugin architecture and a template system. It is available in more than 70 languages among which the most popular languages are English, Spanish and Bahasa Indonesia (Hub.docker.com, 2019).

4.2 Creating a WordPress Container Image

Normally, Docker container is launched based on a Docker image which is built based on a Dockerfile. Similarly, to create a WordPress Container image for this thesis purpose, the first thing done was to define how the WordPress image looks like in a Dockerfile.

Dockerfile

A Dockerfile is simply a text file with instructions to build a Docker image. It contains the commands that are executed manually. The users use `docker build` command to create an automated build that executes several command-line instructions in succession (Docker Documentation, 2019 g). There are several commands used in a Dockerfile to define an image.

In case of the demonstration purpose for this thesis, a simple Dockerfile was created with three commands as listed below:

- FROM – A valid Docker file starts with this instruction. It initializes a new build stage and sets the base image for other instructions. An official WordPress image "FROM wordpress:php5.6-apache" from Docker Hub was used in this section of thesis.
- MAINTAINER – It identifies the author of the Dockerfile.
- COPY – This instruction helps to copy the codes to the defined directory of the image.

First, a directory was created with the name 'Dockerfile' in the terminal with Command 10.

```
$ mkdir Dockerfile (10)
```

Command 11 was executed to add a Dockerfile inside that directory.

```
$ touch Dockerfile (11)
```

Then, the file was edited using the editor 'notepad++' which can be viewed by Command `$ cat Dockerfile` in the terminal as shown in Figure 20. After editing the Dockerfile, the image was built in the terminal by Command 12.

```
$ docker build -t 'my-wp' . (12)
```

In Command 12, `-t` is for giving a repository name. The `.` after 'my-wp' tells the location of the Dockerfile to Docker. The output can be seen in Figure 20 as well.

```
Mala Shrestha@LAPTOP-V0H7KUF0 MINGW64 /c/NewDock/Dockerfile
$ cat Dockerfile
FROM wordpress:php5.6-apache
MAINTAINER mala shrestha
COPY . /var/www/html

Mala Shrestha@LAPTOP-V0H7KUF0 MINGW64 /c/NewDock/Dockerfile
$ docker build -t 'my-wp' .
Sending build context to Docker daemon 2.048kB
Step 1/3 : FROM wordpress:php5.6-apache
--> ed4f6f58673b
Step 2/3 : MAINTAINER mala shrestha
--> Using cache
--> 385e8e30d9e4
Step 3/3 : COPY . /var/www/html
--> f0289e76a322
Successfully built f0289e76a322
Successfully tagged my-wp:latest
```

Figure 20. Output of Command 12.

Command 13 was used to check the new built image as shown in Figure 21.

`$ docker images` (13)

```
Mala Shrestha@LAPTOP-V0H7KUF0 MINGW64 /c/NewDock/Dockerfile
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
my-wp               latest             f0289e76a322      39 seconds ago    397MB
```

Figure 21. Output of Command 13.

Then, the image was run with the Commands 14 and 15 respectively.

`$ docker run - -name mysql-p -e MYSQL_ROOT_PASSWORD=test -d mariadb:10.3.10-bionic` (14)

`$ docker run - -name wp-content - -link mysql-p:mysql -p 8085:80 -d my-wp` (15)

Command 14 was for starting a MySQL Docker container where, `mariadb:10.3.10-bionic` was used in this case and named `mysql-p`. It ran in the background (`-d` flag). The root password (`test`) for MySQL server was defined by passing `MYSQL_ROOT_PASSWORD` environmental variable.

Command 15 was for starting a container based on the image built in Figure 20 (`my-wp`). The container was named `wp-content` which was run in the background (`-d`) and port `8085` was mapped from the host to port `80`. This was done to open its URL in the web-

browser. In order to run WordPress with the database, WordPress container was linked to the MySQL container by **-link**.

In both cases, the images were not found locally. Hence, they were pulled from the library as shown as an output in Appendix 2.

Then, the container was checked if it was running or not by passing Command **docker ps**, which also displayed the port and the output can be seen in Figure 22.

```
Mala Shrestha@LAPTOP-V0H7KUF0 MINGW64 /c/NewDock/Dockerfile
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS            PORTS              NAMES
83ebe84da0f3      my-wp              "docker-entrypoint.s..." 6 seconds ago
Up 5 seconds      0.0.0.0:8085->80/tcp  wp-content
3bee14a627ad      mariadb:10.3.10-bionic "docker-entrypoint.s..." 2 minutes ago
Up 2 minutes      3306/tcp          mysql-p
```

Figure 22. Active Containers.

Finally, the WordPress installation page was successfully accessed through the containers IP **192.168.99.100:8085** (in case of the used PC) in the web-browser (as shown in Figure 23).

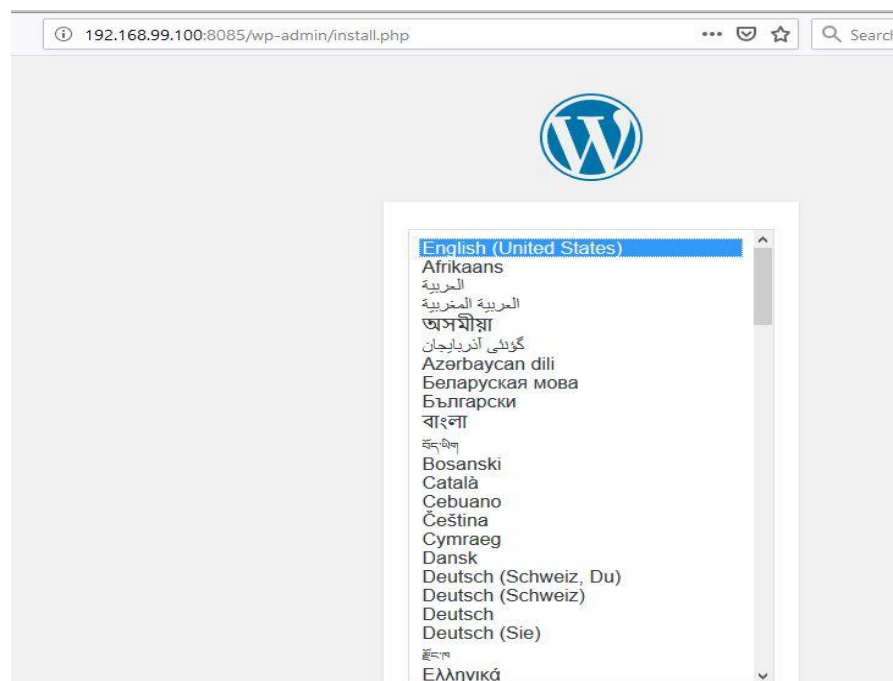


Figure 23. WordPress installation webpage.

4.3 Running WordPress with Docker Compose

The process of running WordPress container image can be simplified by using Docker Composer tool, which is for defining and running multiple containers (also explained in Chapter 3,3.2 section).

In order to use Docker Compose, `docker-compose.yml` file was added to the Dockerfile, defined in the previous steps (4.2 section). The content of the file can be seen in Figure 24.

```
version: '3.1'

services:
  wp-content:
    build: .
    ports:
      - "8085:80"
    environment:
      WORDPRESS_DB_PASSWORD: test
  mysql:
    image: "mariadb:10.3.10-bionic"
    environment:
      MYSQL_ROOT_PASSWORD: test
    volumes:
      - my-datavolume:/var/lib/mysql
volumes:
  my-datavolume:
```

Figure 24.docker-compose.yml file.

In Figure 24, two different services were defined in the same file that were linked with each other.

- wp-content – In this part, a Docker image was built based on the Dockerfile created in previous steps and port 8085 was mapped on the host to port 80 inside the container. Then, the MySQL password was passed via an environment variable.
- mysql – In this part, MySQL image was run from Docker Hub as `mariadb:10.3.10-bionic` with password `test` via an environment variable.

Here, the WordPress base image automatically handles the linked MySQL service and configures database access.

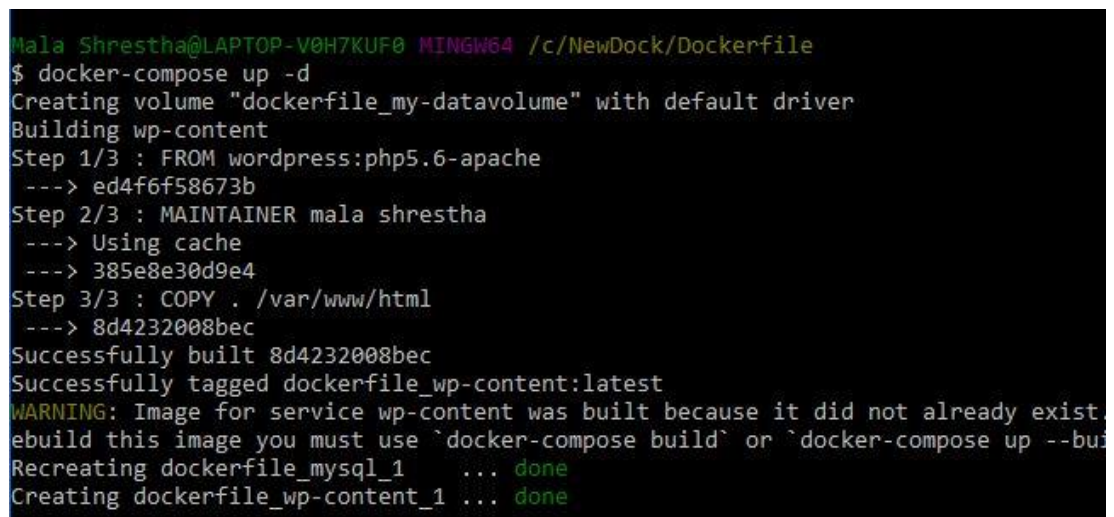
Volumes:

Since, data in the containers are not permanent and it disappears as soon as the container is stopped, WordPress is needed to be initiated every time. In order to avoid this problem, Docker volume was added to the compose file as seen in Figure 24. Docker created a volume in the `/var/lib/docker/volumes` folder. This volume can be stopped with command `docker-compose down -v`.

Finally, after creating Docker composer file, both containers was run with a single Command 16.

```
$ docker-compose up -d (16)
```

The output can be seen in Figure 25.



```
Mala Shrestha@LAPTOP-V0H7KUF0 MINGW64 /c/NewDock/Dockerfile
$ docker-compose up -d
Creating volume "dockerfile_my-datavolume" with default driver
Building wp-content
Step 1/3 : FROM wordpress:php5.6-apache
--> ed4f6f58673b
Step 2/3 : MAINTAINER mala shrestha
--> Using cache
--> 385e8e30d9e4
Step 3/3 : COPY . /var/www/html
--> 8d4232008bec
Successfully built 8d4232008bec
Successfully tagged dockerfile_wp-content:latest
WARNING: Image for service wp-content was built because it did not already exist.
To rebuild this image you must use `docker-compose build` or `docker-compose up --bu
Recreating dockerfile_mysql_1 ... done
Creating dockerfile_wp-content_1 ... done
```

Figure 25. Output of Command 16.

Once the containers were started, the URL was opened in the web-browser and WordPress installation page was successfully accessed (as shown in Figure 26).

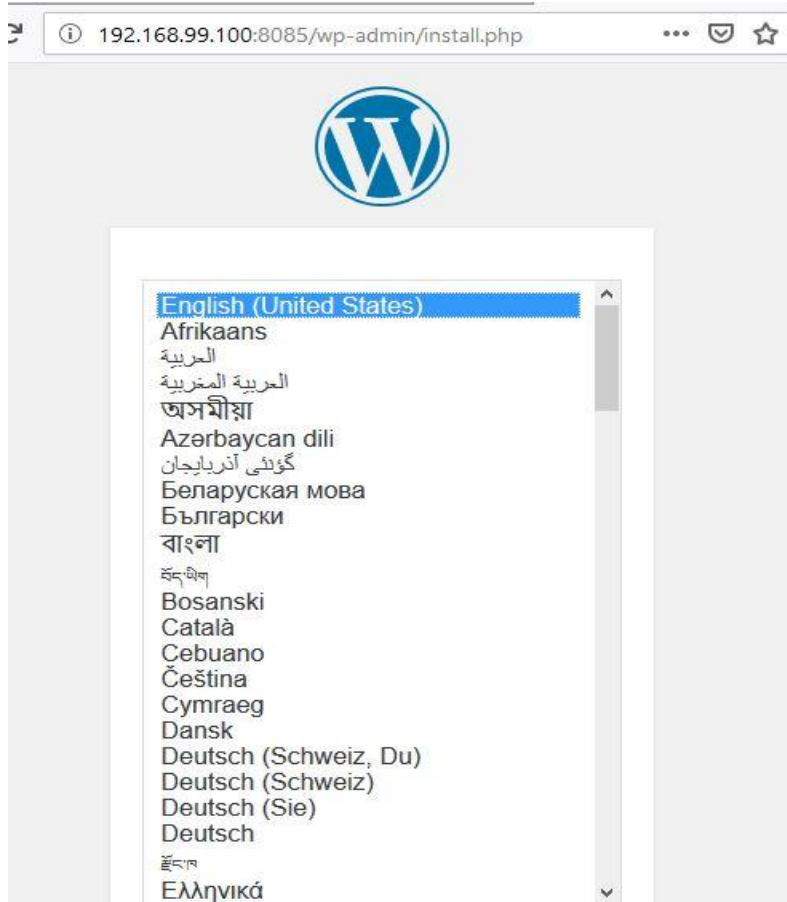


Figure 26. WordPress installation webpage.

Shipping a WordPress image to the Production server:

The image built locally cannot be done in the same way on the Production server. However, the images built on other servers/PC's can be run on the Production server. The image can be pushed to the Docker Hub. (Buddy, 2019.)

First, the image can be built using Docker Hub account's username in the place of image name as shown in Command 17.

```
$ docker build -t my-docker-hub-username/my-image . (17)
```

Then, the image can be pushed to Docker Hub with Command 18.

```
$ docker push my-docker-hub-username/my-image (18)
```

Once the image is uploaded to the registry, it can be pulled onto the production server. It can be done by creating the same `docker-compose` that is used locally. However, instead of using `build:`, the pushed image is referred and the application port is changed from 8085 to 80 as an example shown in Figure 27.

```
version: '3.1'

services:
  wp-content:
    image: "my-docker-hub-username/my-image"
    ports:
      - "80:80"
    environment:
      WORDPRESS_DB_PASSWORD: test

  mysql:
    image: "mariadb:10.3.10-bionic"
    environment:
      MYSQL_ROOT_PASSWORD: test
    volumes:
      - my-datavolume:/var/lib/mysql
volumes:
  my-datavolume:
```

Figure 27. Docker Compose configuration for Production server.

When the configuration is done, it can be run in the terminal with the same command as `docker-compose up -d`.

5 CONCLUSION

The primary goal of the thesis was to gain knowledge about software containerization with Docker and its practice. In order to meet this purpose, this thesis has highlighted the important concepts of Docker beginning with the background of containerization technology. It has shown the reasons behind the emerging use of Docker as a software container while comparing with virtual machines. Similarly, some of the core components of Docker has also been explained to some extent. On the other hand, this thesis has also demonstrated the practical use of Docker by showing the installation of Docker on Windows, as well as its implementation with one of the popular content management systems, WordPress.

The theoretical and practical evaluations were done referring to various sources and the outcome of the projects. Different aspects of Docker were analyzed and discussed. The requirements for the implementation processes were also discussed beforehand. Then, finally, the testing was done in a real environment following the instructions from different sources.

As a result, it was cleared that Docker has many advantages over the traditional virtualization technology and has been a leading software container platform. Even though, container technology cannot fully replace traditional hypervisor virtualization, Docker has been gaining popularity among the users as an alternative solution. The main concern regarding containers security has also been overcome by Docker Security Scanning available in docker cloud which has solved the problems of developers or system administrators. Despite of few drawbacks of having a complicated feature, an issue with cross-platform compatibility or frequent upgrades, Docker has guaranteed scalable, applications isolation, reliable, portable and secure runtime environment for daily workflows (G2 Crowd, 2019).

In case of the Docker installation process on Windows 10, Docker has provided the Docker Toolbox which simplified the installation process. Apart from some of the errors, the installation was only a few steps with quick and easy desktop shortcuts. Similarly, the large ecosystem of Docker holds a container image for any software which made the next implementation with WordPress easier and convenient. In the practical example of WordPress implementation, Docker Hub base images were used with the addition of only few steps in the Dockerfile. It reduced the needed work to create the images from

scratch and saved time. Similarly, while running with Docker Compose, a single file in plain text was only needed to represent the whole WordPress configurations and its connections. This way, Docker has simplified the workflows of packaging software stacks without the hassle of dependencies and other configurations.

Overall, this thesis has bundled the learning experience about software containerization technology with Docker and a proper guidance to implement it in a practical field.

REFERENCES

Access.redhat.com. (2019). 7.3. Comparison with Virtual Machines - Red Hat Customer Portal. [online] Available at: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/7.0_release_notes/sect-red_hat_enterprise_linux-7.0_release_notes-linux_containers_with_docker_format-comparison_with_virtual_machines [Accessed 20 Feb. 2019].

Aquasec. (2018 a). Docker Architecture. [online] Available at: <https://www.aquasec.com/wiki/display/containers/Docker+Architecture#DockerArchitecture-TheDockerEngine> [Accessed 24 Feb. 2019].

Aquasec. (2018 b). Docker Containers vs. Virtual Machines. [online] Available at: <https://www.aquasec.com/wiki/display/containers/Docker+Containers+vs.+Virtual+Machines> [Accessed 22 Feb. 2019].

Buddy. (2019). WordPress in Docker. Part 1: Dockerization | Buddy. [online] Available at: https://buddy.works/guides/wordpress-docker-kubernetes-part-1?fbclid=IwAR3lQath8_VKC6WbSWxD75K4sBEFCtOo-6piy1MdM6h1oLoOHlscCuvShl [Accessed 24 Mar. 2019].

DataflairTeam. (2018 a). Advantages and Disadvantages of Docker - Learn Docker - DataFlair. [online] DataFlair. Available at: <https://data-flair.training/blogs/advantages-and-disadvantages-of-docker/> [Accessed 22 Feb. 2019].

DataflairTeam. (2018 b). Docker Tutorial For Beginners | Learn Docker Architecture - DataFlair. [online] DataFlair. Available at: <https://data-flair.training/blogs/docker-tutorial/> [Accessed 24 Feb. 2019].

DeMuro, J. (2018). What is container technology?. [online] TechRadar. Available at: <https://www.techradar.com/news/what-is-container-technology?fbclid=IwAR2Lj9bnUeJLwWiZLcV0sozZPRI0cj2tAsoVDIKwbgSJD0JJNWuGFppK44> [Accessed 19 Jan. 2019].

Docker. (2019 a). About Docker - Management & History | Docker. [online] Available at: <https://www.docker.com/company> [Accessed 20 Jan. 2019].

Docker. (2019 b). What is a Container? | Docker. [online] Available at: <https://www.docker.com/resources/what-container> [Accessed 12 Feb. 2019].

Docker. (2019 c). Why Docker? | Docker. [online] Available at: <https://www.docker.com/why-docker> [Accessed 12 Feb. 2019].

Docker Documentation. (2019 a). Docker overview. [online] Available at: <https://docs.docker.com/engine/docker-overview/#docker-engine> [Accessed 23 Feb. 2019].

Docker Documentation. (2019 b). Docker Toolbox overview. [online] Available at: <https://docs.docker.com/toolbox/overview/> [Accessed 25 Feb. 2019].

Docker Documentation. (2019 c). Install Docker Desktop for Windows. [online] Available at: <https://docs.docker.com/docker-for-windows/install/> [Accessed 25 Feb. 2019].

Docker Documentation. (2019 d). Install Docker Toolbox on Windows. [online] Available at: https://docs.docker.com/toolbox/toolbox_install_windows/ [Accessed 25 Oct. 2019].

Docker Documentation. (2019 e). Kitematic user guide. [online] Available at: <https://docs.docker.com/kitematic/userguide/> [Accessed 25 Feb. 2019].

Docker Documentation. (2019 f). Overview of Docker Compose. [online] Available at: <https://docs.docker.com/compose/overview/> [Accessed 26 Feb. 2019].

Docker Documentation. (2019 g). [online] Available at: <https://docs.docker.com/engine/reference/builder/> [Accessed 11 Mar. 2019].

Docs.oracle.com. (2012 a). 1.1.1. Brief History of Virtualization. [online] Available at: https://docs.oracle.com/cd/E26996_01/E18549/html/VMUSG1010.html?fbclid=IwAR3CeVMMxj9syaqP5XAS0iW3BoGsbQ0ziVicJ_bzvcCt0pbA1r6YHHbwgoE [Accessed 19 Jan. 2019].

Docs.oracle.com. (2012 b). 1.1.2. Hypervisor. [online] Available at: https://docs.oracle.com/cd/E26996_01/E18549/html/VMUSG1011.html [Accessed 19 Jan. 2019].

DwGlogo - Stunning free images. (2017). Docker logo | Dwglogo. [online] Available at: <https://dwglogo.com/docker/> [Accessed 12 Feb. 2019].

G2 Crowd. (2019). Docker Reviews 2019: Details, Pricing, & Features | G2. [online] Available at: <https://www.g2.com/products/docker/reviews#survey-response-1426766> [Accessed 11 Apr. 2019].

Hub.docker.com. (2019). Docker Hub. [online] Available at: https://hub.docker.com/_/wordpress [Accessed 7 Mar. 2019].

Image.slidesharecdn.com. (2016). [online] Available at: <https://image.slidesharecdn.com/dockerbirthday3slides-overview-160322184101/95/docker-birthday-3-intro-to-docker-slides-18-638.jpg?cb=1458672346> [Accessed 12 Feb. 2019].

Janetakis, N. (2017). Comparing Virtual Machines vs Docker Containers. [online] Nick Janetakis. Available at: <https://nickjanetakis.com/blog/comparing-virtual-machines-vs-docker-containers> [Accessed 20 Feb. 2019].

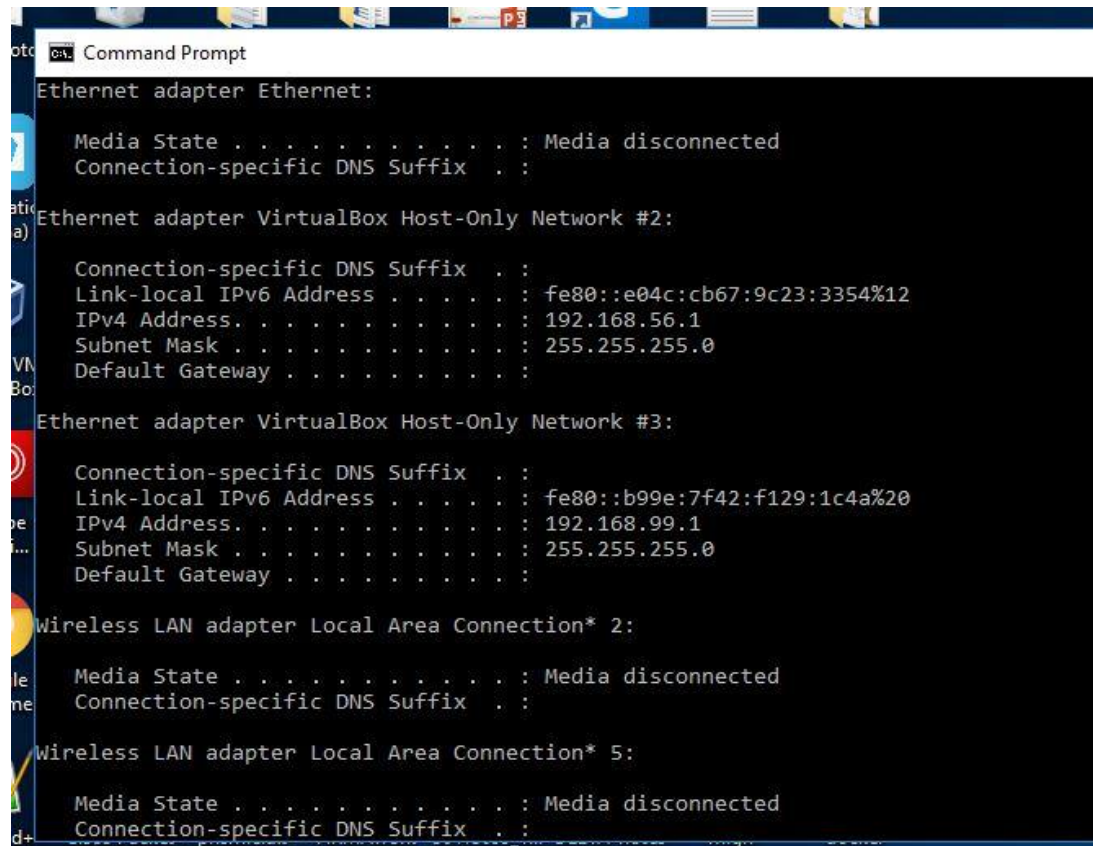
Rubens, P. (2017). What are containers and why do you need them?. [online] CIO. Available at: <https://www.cio.com/article/2924995/what-are-containers-and-why-do-you-need-them.html> [Accessed 19 Jan. 2019].

Vaughan-Nichols, S. (2018). What is Docker and why is it so darn popular? | ZDNet. [online] ZDNet. Available at: <https://www.zdnet.com/article/what-is-docker-and-why-is-it-so-darn-popular/> [Accessed 21 Jan. 2019].

VMWare. (2019). Virtualization Technology & Virtual Machine Software: What is Virtualization?. [online] Available at: <https://www.vmware.com/solutions/virtualization.html> [Accessed 19 Jan. 2019].

WordPress.org. (2019). Democratize Publishing. [online] Available at: <https://wordpress.org/about/> [Accessed 7 Mar. 2019].

Appendix 1 – Configuring IP Address



```
Command Prompt
Ethernet adapter Ethernet:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Ethernet adapter VirtualBox Host-Only Network #2:

    Connection-specific DNS Suffix  . :
    Link-local IPv6 Address . . . . . : fe80::e04c:cb67:9c23:3354%12
    IPv4 Address. . . . . : 192.168.56.1
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . :

Ethernet adapter VirtualBox Host-Only Network #3:

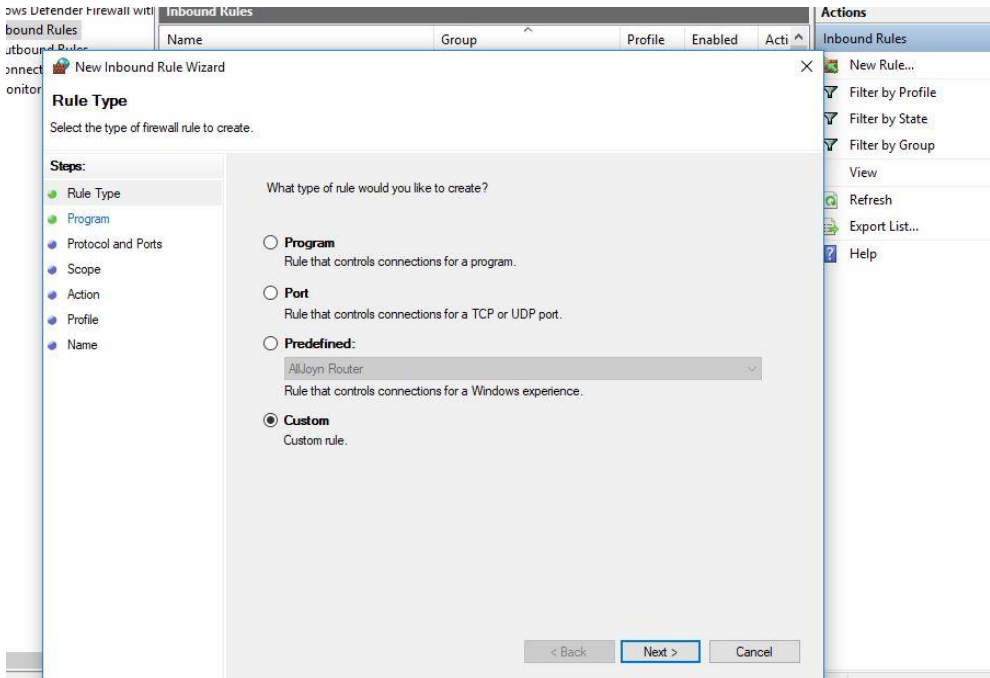
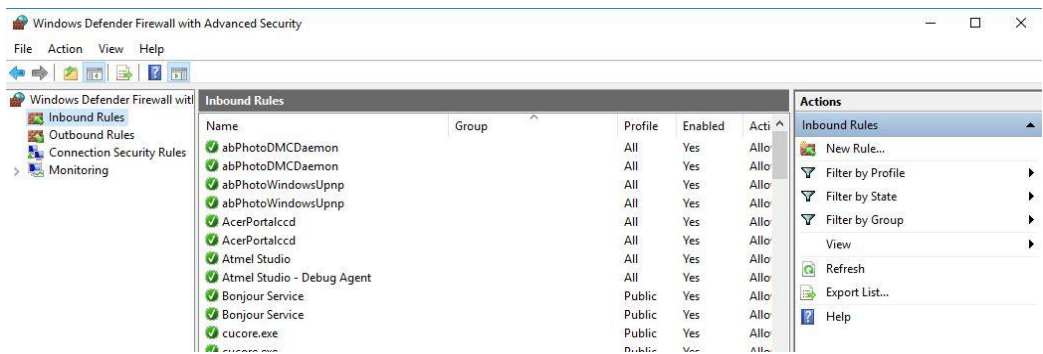
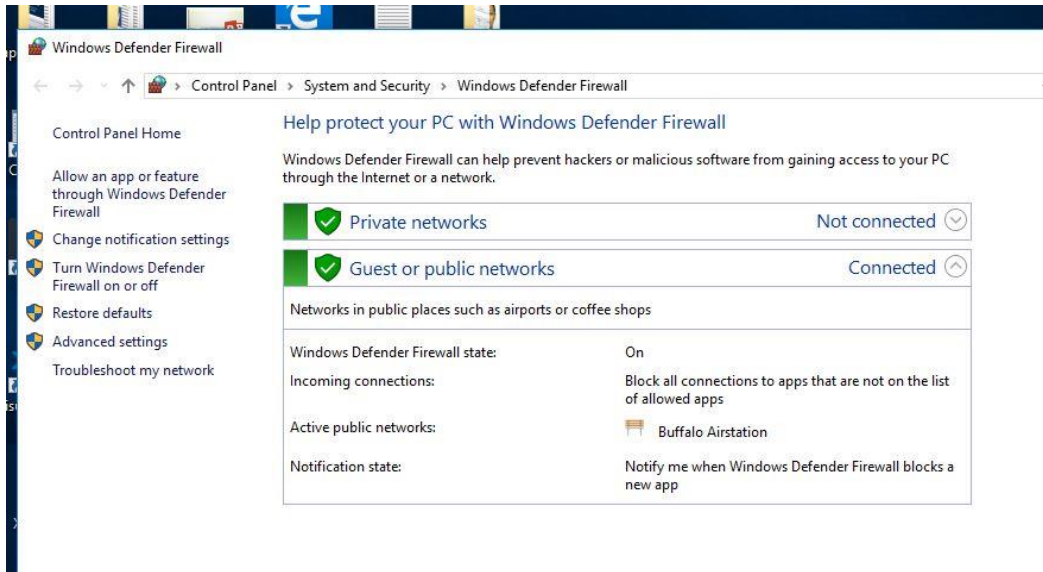
    Connection-specific DNS Suffix  . :
    Link-local IPv6 Address . . . . . : fe80::b99e:7f42:f129:1c4a%20
    IPv4 Address. . . . . : 192.168.99.1
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . :

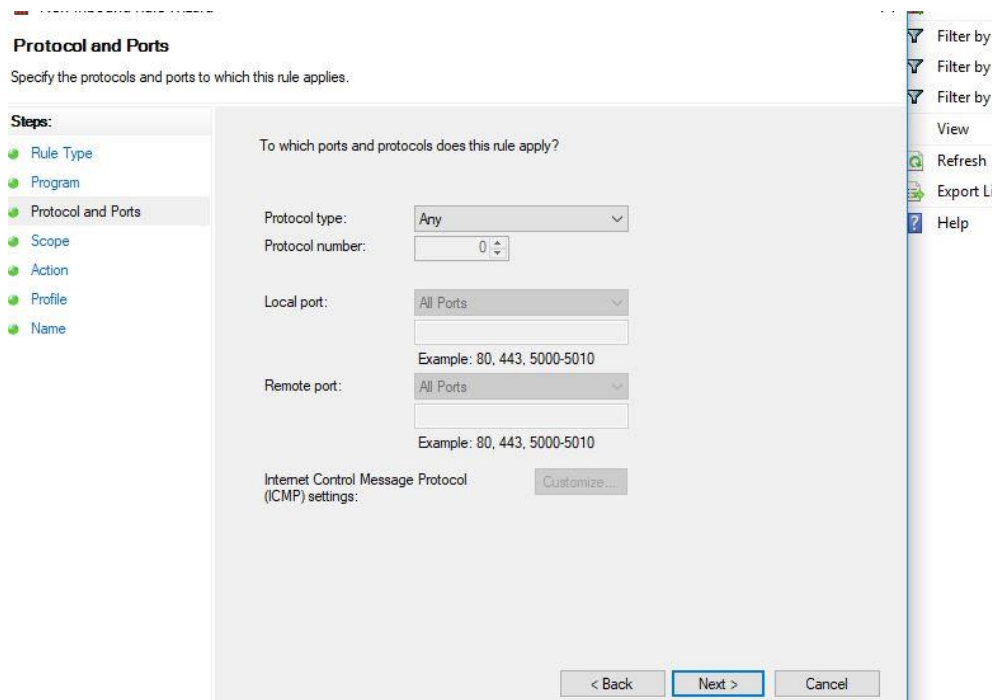
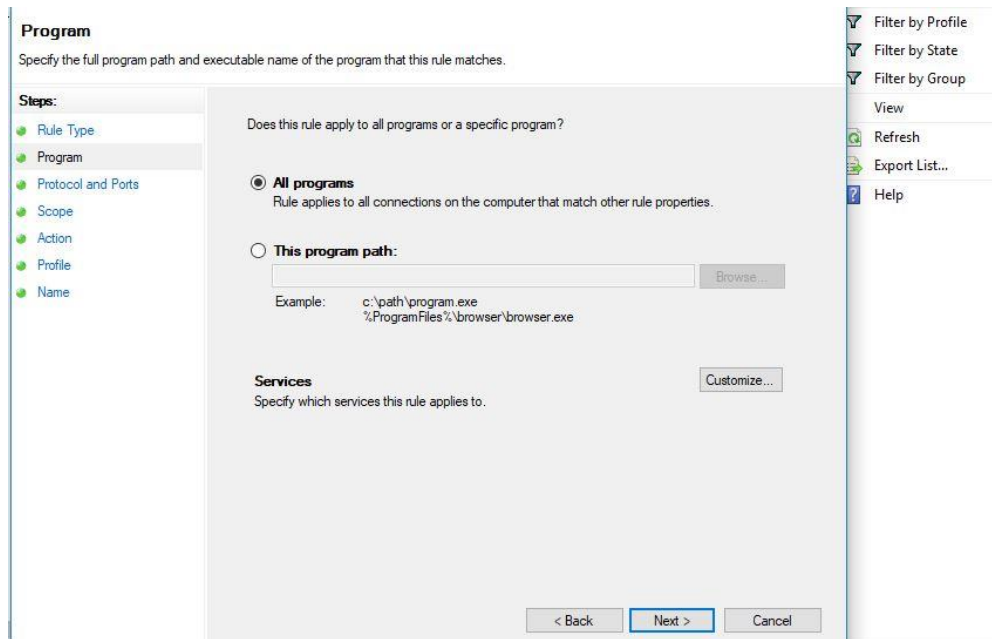
Wireless LAN adapter Local Area Connection* 2:

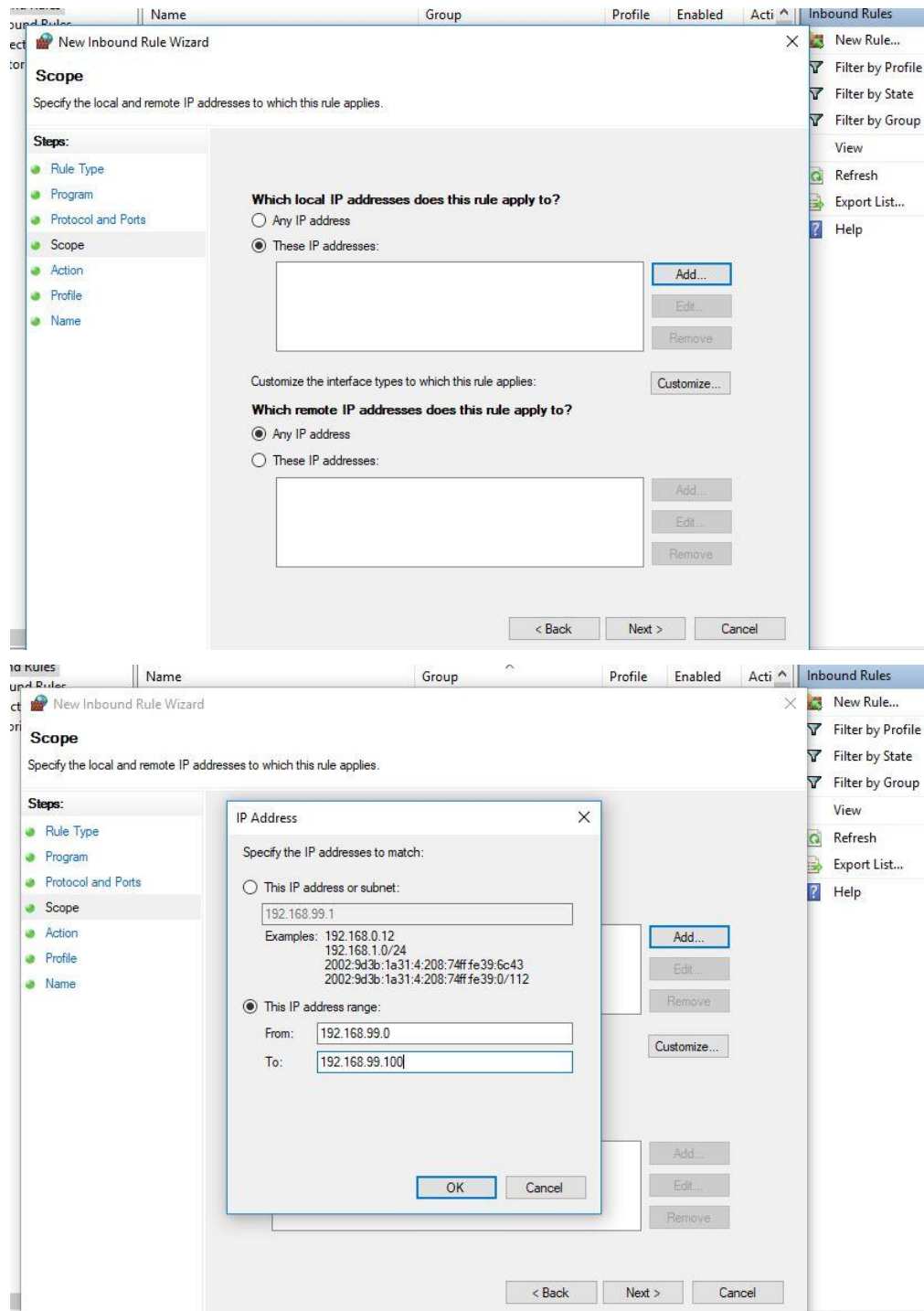
    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 5:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :
```







Action

Specify the action to be taken when a connection matches the conditions specified in the rule.

Steps:

- Rule Type
- Program
- Protocol and Ports
- Scope
- Action**
- Profile
- Name

What action should be taken when a connection matches the specified conditions?

Allow the connection
This includes connections that are protected with IPsec as well as those are not.

Allow the connection if it is secure
This includes only connections that have been authenticated by using IPsec. Connections will be secured using the settings in IPsec properties and rules in the Connection Security Rule node.
[Customize...](#)

Block the connection

< Back Next > Cancel

New Inbound Rule Wizard

Profile

Specify the profiles for which this rule applies.

Steps:

- Rule Type
- Program
- Protocol and Ports
- Scope
- Action
- Profile**
- Name

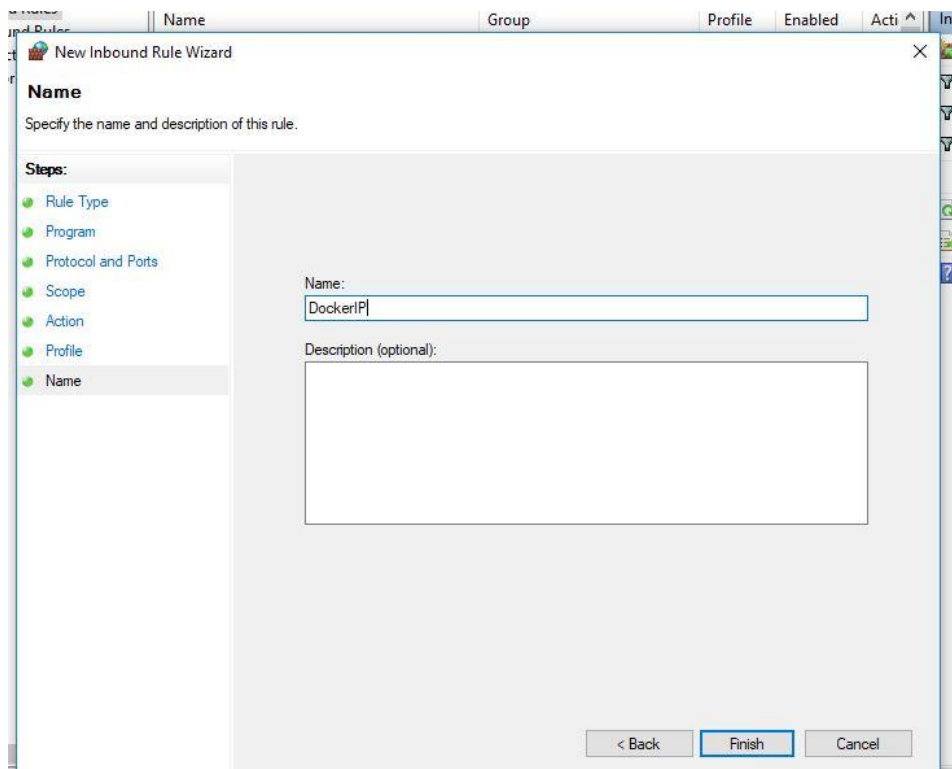
When does this rule apply?

Domain
Applies when a computer is connected to its corporate domain.

Private
Applies when a computer is connected to a private network location, such as a home or work place.

Public
Applies when a computer is connected to a public network location.

< Back Next > Cancel



Appendix 2- Output of Command 14 and 15 respectively

```
Unable to find image 'mariadb:10.3.10-bionic' locally
10.3.10-bionic: Pulling from library/mariadb
473ede7ed136: Pull complete
c46b5fa4d940: Pull complete
93ae3df89c92: Pull complete
6b1eed27cade: Pull complete
a7571426e9fe: Pull complete
dba27abd0643: Pull complete
3bfc1e8ac8a0: Pull complete
01fbc1aaebf: Pull complete
0c112e7b58a2: Pull complete
7a0418b24342: Pull complete
dea457cf24a8: Pull complete
21a8ebff5731: Pull complete
8171c3b81842: Pull complete
92c72b0d0b51: Pull complete
Digest: sha256:9d443337dfbb2a34583ed7c968cde6115ce1b10630530ff1f0f5c7f1e6f0a76b
Status: Downloaded newer image for mariadb:10.3.10-bionic
4270f31397ff196fc5b8999091aea28d6e9bb0bc7b68c4e7daf103296e4feafb
```

```
Unable to find image 'wordpress:4.9.8-php5.6-apache' locally
4.9.8-php5.6-apache: Pulling from library/wordpress
f17d81b4b692: Pull complete
376d99d019dc: Pull complete
80b3573727f0: Pull complete
2c492579cd1f: Pull complete
9127acfc108e: Pull complete
475593d953b6: Pull complete
52442c108349: Pull complete
34b7a8ed8171: Pull complete
57b93ed05069: Pull complete
54dfbd21ab2b: Pull complete
0242e4163f99: Pull complete
9a8826398acb: Pull complete
b6f2ba14da98: Pull complete
02a1f2ace267: Pull complete
d0ab6c1a30c0: Pull complete
027f4b1c7a64: Pull complete
220f2ab12d7f: Pull complete
09d5f6563cb2: Pull complete
194e7baff9aa: Pull complete
Digest: sha256:a2eb2069e5eefa2fbaeda30e87861be262e5ec5271c2a43af8059da717d4a21c
Status: Downloaded newer image for wordpress:4.9.8-php5.6-apache
c9e5536afd3f803cb1f67ac59aed44a926d524978458b52759ab6e7b203967d1
```