Software Application Development Using Container Technology

Dražen Marinković, Velimir Kojić, Zoran Ž. Avramović
PanEuropean University APEIRON Banjaluka, Republic of Srpska, B&H
{drazen.m.marinkovic, velimir.d.kojic, zoran.z.avramovic}@apeiron-edu.eu

Critical Review
https://doi.org/10.7251/JIT2101054M UDC: 656.615.073:621.869.88

Abstract: The paper will give an example and an overview of how we can set up and maintain web applications using a Docker. We will define what Docker is, what containers are and how to use them. Developers often find themselves in a situation where their program works properly on a computer in the laboratory environment in which the application was developed, but after installing the program on the production server, the program does not work as expected. In such circumstances, it is difficult for a programmer to determine why a program is not working. Docker solves this problem by placing applications and virtual containers that run on the same operating system.

Keywords: Docker, Devops,Virtual machine, Container technology.

Introduction – Problems of operation development

Since the beginnings of programming and software application development, developers have created programs and applications in one place and installed them to run in another place. Preparing the work environment has often been a problem in implementation.

Developers often find themselves in a situation where their program works properly on a computer in the laboratory environment in which the application was developed, but after installing the program on the production server, the program does not work as expected. In such circumstances, it is difficult for a programmer to determine why a program is not working. There can be several reasons for that and it is very difficult to detect these malfunctions.

This type of problem is costly, frustrating, and often disrupts interpersonal relationships between system administrators and developers.

Today, there are several technologies that are trying to solve this problem. The traditional way of solving is based on the use of virtual machines, using cloned environments and "snapshots". A new and improved way is to use containerization technologies (in our case Docker containers).

Docker – a different approach to virtual machines

One of Docker's common misunderstandings is its comparison to a virtual machine, which seems to do something similar, but is, in fact, significantly different. [1]

Virtual machines solve the same problem, but in a different way. Virtual machines are virtual instances of a complete operating system, Windows or Linux. If we are developing a web application on a Linux virtual instance, we can run that application anywhere to run that same instance of Linux.

This approach has been successful for years. However, there is one problem - if we want to service and deploy a single web application, we need to run a complete virtual operating system. It is quite expensive, especially if we have multiple applications running.

Docker solves this problem by placing applications and virtual containers that run on the same operating system. Each container contains the com-

plete code, all the libraries and all the dependencies that are necessary for the application to run. [2]

This approach saves us large amounts of disk space, time, processing power and significantly reduces complexity.

Why Docker?

Many reputable companies use Docker for their web applications.

The main advantages of using Docker are the following:

- Docker allows us to have many more individual applications on the same number of servers than with other technologies.
- It makes application development encapsulated, ready to use. Everything is delivered in containers.
- Also, it makes the whole process of managing and installing applications on servers much easier and more secure. We can modify the application, send a new image to the repository, and instantly launch that application from any location.

When we put all this together, we can have a very attractive product that will make maintenance work easier for both developers and system administrators. [3]

Placing applications in Docker using Docker containers

The basic steps of developing and placing a web application in a container using Docker are as follows:

- Installing Docker on the physical or virtual environment we will be using
- Setting up the registry on the Docker Hub
- Creating your own Docker image
- Setting up "docked" environments
- Setting up the application

Docker installation

Docker allows us to easily run our applications anywhere, but the places where those applications run must have Docker installed.

There are three main Docker releases:

- Docker Enterprise Edition (Docker EE)
- Docker Comunity Edition (Docker CE)
- Docker Cloud

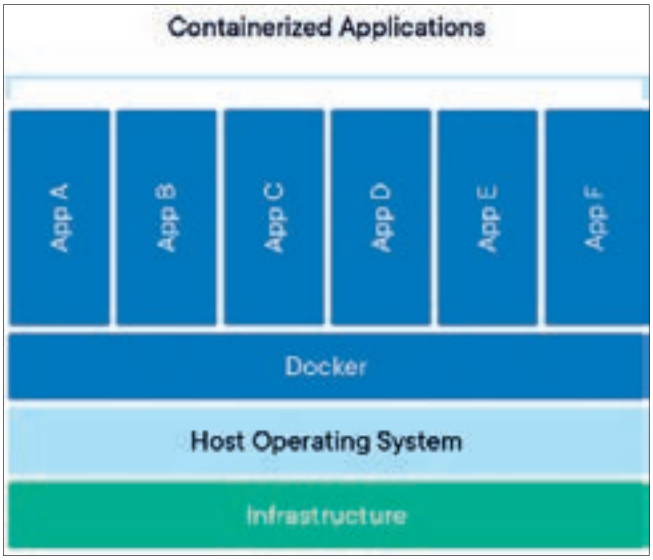

Figure 1: *Illustrative presentation of container technology*
**https://www.docker.com/resources/what-container*

At the moment, Docker can be installed on Windows, Linux and MacOS environments, as well as on cloud service providers such as Amazon AWS and Microsoft Azure.

In our example, we will use Linux CentOS 8.

Before starting the installation, it is necessary to remove all old versions of Docker called docker or docker-engine. [4]

We will perform the installation by adding an official repository, so that we have the option of automatically upgrading and patching Docker.

If it is a server that doesn't already have a Docker repository, we need to add them.

Preparing the Docker repository:

```
$ sudo dnf install -y dnf-utils
$ sudo dnf config-manager \
 --add-repo \
   https://download.docker.com/linux/
centos/docker-ce.repo
```

Docker Engine installation:

```
$ sudo dnf install docker-ce docker-
ce-cli containerd.io
```

If we have not used Docker repositories before, the installation will ask us to accept the GPG - Gnu Privacy Guard key, of the repository, which we must do if we want to continue the installation.

After a successful installation, we start Docker as follows:

```
$ systemctl start docker.service
```

Setting up the registry on the Docker Hub

In order to easily distribute applications, we need to have a place to share them.

The Docker Hub is like a GitHub Docker container. On it we can find thousands of applications and examples of how to use them.

To set up our own private or public repository and to start setting up our applications, we need to go to the Docker Hub website.

We can use the Docker Hub to set up and download changes to our application in different locations. There are also tools we can use to automate this.

To access the Docker Hub panel, we must log in using our user account. If we don't have it, we need to create an account on the Docker Hub. After creating an account, we log in to the Docker Hub panel, from where we can create a public or private repository.

Once we have created the repository, we can proceed with the next steps.

It is important to note that there is a paid Docker Hub option, which provides more functionality and resources than the free one, which we will use in this example. For these purposes, the free option is more than enough. [5]

SETTING UP DOCKED ENVIRONMENTS

The next step is to prepare the Docker environment to execute the application. We will do this using the command line and several different tools.

These environments are responsible for setting up and running the container. There are several other, interesting things we can do, such as running multiple instances of our application and using a load balancer to send traffic to different versions.

Making Docker images and containers

Docker containers are created from a Docker figure. In order for our application to be distributed to different servers, we need to create a Docker image. We can do this using the Docker file.

The Docker file (we will use the official title Dockerfile below), usually contains 3 to 30 lines of text that represent the commands necessary to create Docker figures.

There are many ways to create a Dockerfile, but before that we need to get acquainted with the best practice, before we make it.

For example, we will take a simple application, node-bulletin-board, which is used as an example of how to create and run Docker figures. [5]

The first step is to download the necessary files:

```
$ curl -LO https://github.com/dock-
ersamples/node-bulletin-board/ar-
chive/master.zip
```

We unpack the compressed archive:

```
$ unzip master.zip
```

We change the current working directory to the directory created by unpacking the archive:

```
$ cd node-bulletin-board-master/bul-
letin-board-app
```

We can look at the contents of the new directory:

```
[root@localhost docker]# cd node-bulletin-board-master/bulletin-board-app
[root@localhost bulletin-board-app]# ls -la
total 36
drwxr-xr-x. 4 root root  187 Oct 27 16:53 .
drwxr-xr-x. 3 root root   65 Oct 27 16:53 ..
-rw-r--r--. 1 root root 1239 Oct 27 16:53 app.js
drwxr-xr-x. 2 root root   53 Oct 27 16:53 backend
-rw-r--r--. 1 root root  127 Oct 27 16:53 Dockerfile
drwxr-xr-x. 3 root root   23 Oct 27 16:53 fonts
-rw-r--r--. 1 root root   22 Oct 27 16:53 .gitignore
-rw-r--r--. 1 root root 1826 Oct 27 16:53 index.html
-rw-r--r--. 1 root root 1131 Oct 27 16:53 LICENSE
-rw-r--r--. 1 root root  572 Oct 27 16:53 package.json
-rw-r--r--. 1 root root  888 Oct 27 16:53 readme.md
-rw-r--r--. 1 root root 1071 Oct 27 16:53 server.js
-rw-r--r--. 1 root root 1227 Oct 27 16:53 site.css
[root@localhost bulletin-board-app]#
```

Figure 2: *Application directory content*

We can see a Dockerfile that contains parameters to create. When we open Dockerfile, we see its contents in the image below:

Figure 3: *Dockerfile content*

In order, we see lines that give instructions on which command to execute:

- FROM: which figures will be used as the basis
- WORKDIR: define the working directory, all commands will be executed in it
- COPY: copy local files to the container
- RUN: run the command during container creation
- CMD: run the command after starting the container
- EXPOSE: publish container ports to be available outside the Docker network

We will continue to create containers. The command with which we create and test the image is:

```
$ docker build --tag bulletin-board:1.0 .
```

For the command to be successful, we must be in the directory where the Dockerfile is located or manually specify the path.

Figure 4: *Creating a Docker image*

By command `$ docker images` we can view the containers available in the local environment:

```
[root@localhost bulletin-board-app]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
bulletinboard       1.0                 0451f4b23265        4 minutes ago       177MB
node                current-slim        9ac9e9f30b2c        2 days ago          160MB
[root@localhost bulletin-board-app]#
```

Figure 5: *Display of available figures on a Docker host*

Once we have the image created, we can proceed to create the container with the command:

```
$ docker run --publish 8000:8080 --detach --name bb bulletinboard:1.0
```

We can access the application at the address of the server and port that we entered in EXPOSE conf

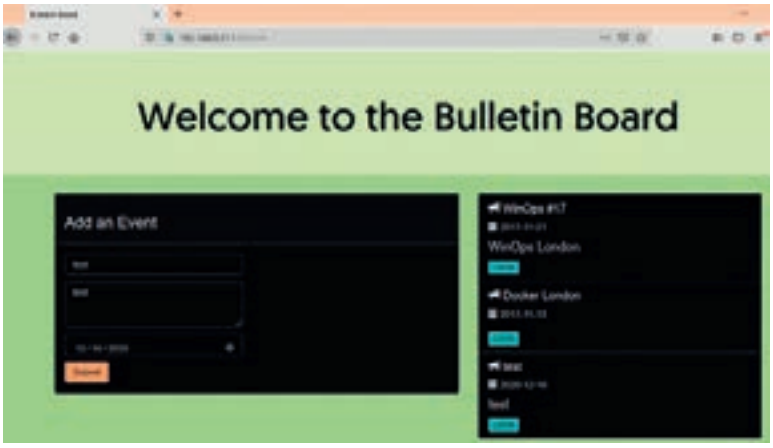

Figure 6: *Application launched from Docker container*

Image publishing on Docker Hub

We create a bulletinboard repository on our Docker hub.

By command

```
$ docker tag bulletinboard:1.0
kojo1984/bulletinboard:1.0
```

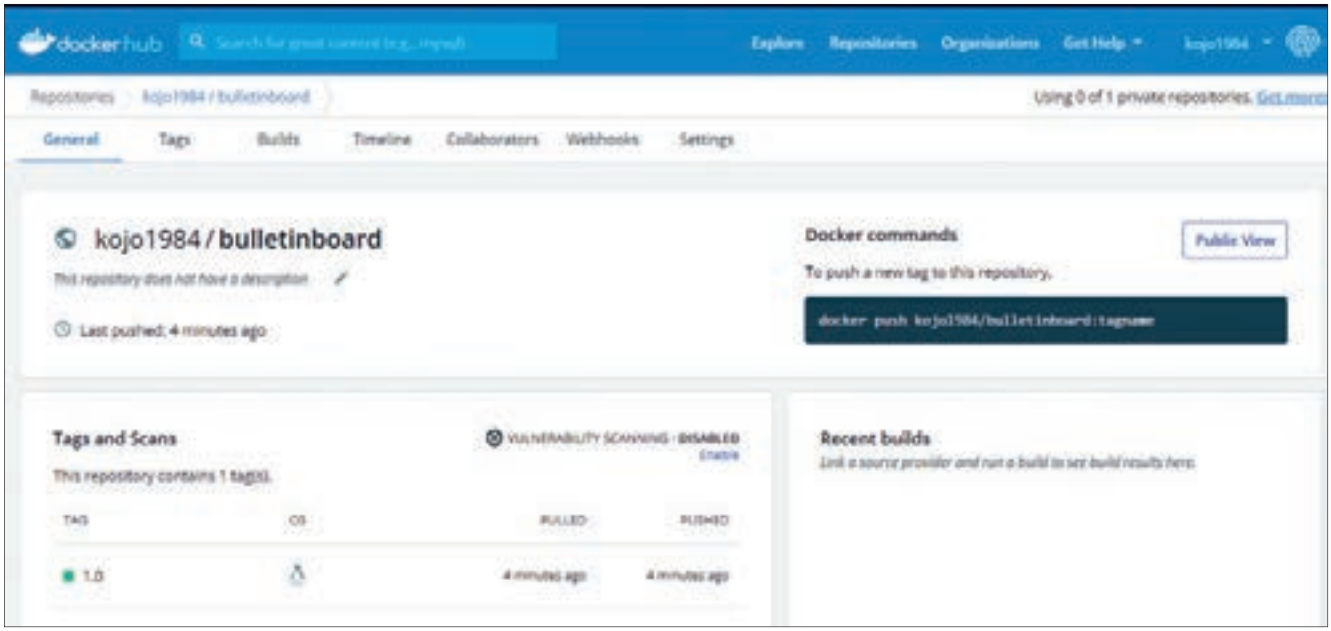

Figure 7: *Image tagging and copying on Docker Hub*

We tag the image, preparing it for copying to the Docker hub.

By command

```
$ docker push kojo1984/bulletinboard:1.0
```

We copy the image to the repository

Figure 8: *Image display on Docker Hub - we can see the published image on Docker Hub*

Once we have published and made the image available on the Docker Hub, we can run it anywhere. If we try to run the image on a system that does not have an image, Docker will automatically try to download it from the Docker Hub. By using the figure in this way, we no longer need to install additional software and libraries on the system on which the application will run, it is enough to have Docker installed.

It is a good practice to stick Dockerfile together with the source code of the application, so that the changes and versions in the file and the application are at the same level. [6]

Conclusion

Docker is a very popular technology nowadays, and with good reason. In this paper, we have seen only a fraction of what can be done with container technology (in our case with Docker).

With the proper use of container technology, in addition to saving us money, it will save us time, resources, make work easier and make applications more accessible.

We have presented the basic process of how to run a containerized application on a server. The same procedure is applied anywhere, from local server, to virtual machine and cloud environment.

References

[1] [1] A. Avram, „Docker: Automated and Consistent Software Deployments," [Na mreži]. Available: http://www.infoq.com/news/2013/03/Docker. [Poslednji pristup 21 4 2021].

[2] [2] F. John, „Docker: a Software as a Service, Operating System-Level Virtualization Framework," *Code4Lib Journal,* br. 25, 2014.

[3] [3] B. Thanh, „Analysis of Docker Security," *arXiv: Cryptography and Security,* 2015.

[4] [4] S. R. Meadusani, „Virtualization Using Docker Containers: For Reproducible Environments and Containerized Applications," , 2018. [Na mreži]. Available: https://repository.stcloudstate.edu/msia_etds/50. [Poslednji pristup 21 4 2021].

[5] [5] L. W. I. Z. L. Bass, „Devops: A Software Architect's Perspective," 2015. [Na mreži]. Available: https://amazon.com/devops-software-architects-perspective-engineering/dp/0134049845. [Poslednji pristup 21 4 2021].

[6] [6] B. Carl, „Dear DockerHub users: please configure your repository links," 2014. [Na mreži]. Available: https://carlboettiger.info/2014/11/07/dear-docker-hub-users.html. [Poslednji pristup 21 4 2021].

[7] [7] F. John, „Docker: a Software as a Service, Operating System-Level Virtualization Framework," *Code4Lib Journal,* br. 25, 2014.

Submitted: May 8, 2021
Accepted: May 31, 2021

ABOUT THE AUTHORS

Dražen M. Marinković was born in Banja Luka, Republika Srpska - Bosnia and Herzegovina. He finished elementary school in Prnjavor. He finished secondary school of Electrical Engineering in Prnjavor. He graduated at the Faculty of Information Technologies at Pan-European University "APEIRON" in Banja Luka. Master studies completed in 2015. Doctoral studies completed 2020. He had PHD in Computer Science. He lives and works in Prnjavor.

Velimir D. Kojić, born 1984. in Sisak, Croatia. Back in 2018., earned Bachelor of Engineer in informatics and computing degree at Pan-European University "Apeiron" – Faculty of Information Technologies. Currently attending master studies at the same university.
For a past 10 years, working as a System Engineer and Security Officer."

Zoran Ž. Avramović born in Serbia. He graduated from the University of Belgrade. At this University he received a degree of a master, and then a PhD. He is academician of the Russian Transport Academy, academician of the Russian Academy of Natural Sciences, academician of the Engineering Academy of Serbia, academician of the Academy of Electrotechnical Sciences of the Russian Federation and Scientific Secretary of the Electrical Engineering Department of the Engineering Academy of Serbia. He is professor on the University of Belgrade, professor on the Pan-European University APEIRON Banjaluka, Republic Srpska and professor on the University Adriatic, Montenegro.

FOR CITATION

Dražen Marinković, Velimir Kojić, Zoran Ž. Avramović, Software Application Development Using Container Technology, *JITA – Journal of Information Technology and Applications Banja Luka*, PanEuropien University APEIRON, Banja Luka, Republika Srpska, Bosna i Hercegovina, JITA 11(2021) 1:54-60, (UDC: 656.615.073:621.869.88), (DOI: 10.7251/JIT2101054M), Volume 11, Number 1, Banja Luka, June 2021 (1-68), ISSN 2232-9625 (print), ISSN 2233-0194 (online), UDC 004