6/14/2021

Nikolaos

Fontys ICt student

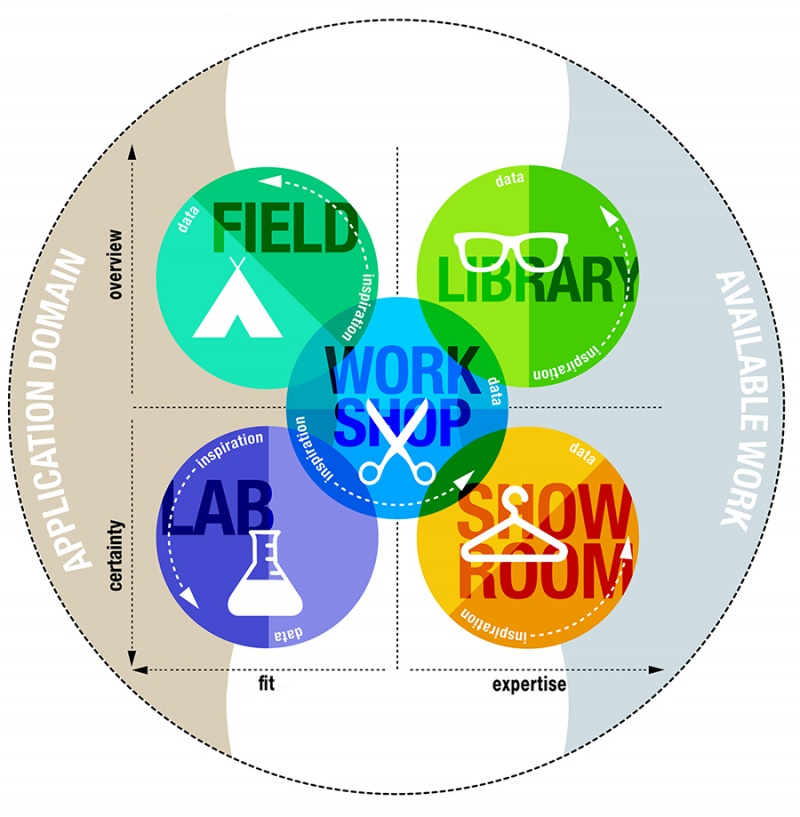
Docker Research

Introduction

My name is Nikolaos Tocila and I study at Fontys Hogeschool in the ICT department. As an IT student I have come across many difficulties and obstacles during my software development. One of my big concerns was using Docker. I have never used or interacted with Docker before so, I was a bit intimidated by including it in my project.  
This research is conducted for the purpose of answering three main questions: WHAT? WHY? HOW?  
The research is created following the DOT Framework methodology research. The DOT Framework methodology is basically covered by five parts:

1. Field
2. Library
3. Workshop
4. Lab
5. Showroom

And by combining some of the methods above we can answer the questions: What? Why? How?



What (The question of this Research)?

Overview

Let’s begin by explaining what Docker is. Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker’s methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production. Docker is available for operating systems like: Windows, Mac and Linux. However, it might be a bit confusing sometimes since some windows users can use the Linux version instead of the Windows one.  
The reason I chose Docker as a research subject is because during my individual and group project, I had to learn how to use it and integrate it into my application. Searching online for my solution was just a waste of time since every solution I found was different than the other ones. So, the real question for this research is: How did I deploy my Web API automatically to Docker Hub through GitHub Actions?

Why (The reasons for this research)

Why did I choose Docker as a subject of my research? The answer is simple, I just want to help other software developers to struggle less than me, while integrating Docker into their projects. To make it easier for readers of this research I have prepared some snippets in combination with a short description about the snippet itself.   
To begin with, let’s first answer the question Why Docker? Developing apps today requires so much more than writing code. Multiple languages, frameworks, architectures, and discontinuous interfaces between tools for each lifecycle stage creates enormous complexity. Docker simplifies and accelerates your workflow, while giving developers the freedom to innovate with their choice of tools, application stacks, and deployment environments for each project. Below are the advantages and disadvantages of using Docker:

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| Rapid Deployment | Data in a container |
| Security | Learning Curve |
| Easy CI/CD | Not all applications benefit from containers |

After coming across many different opinions about Docker, the most common opinions about Docker pros & cons are listed in the above table.

Let’s explain the advantages first:

**Rapid Deployment**

It can decrease deployment to seconds. It is because of the fact that it can create a container for every process and even does not boot an OS. So, even without worrying about the cost to bring it up again, it would be higher than what is affordable, Data can be created as well as destroyed.

**Security**

Docker makes sure that applications that are running on containers are completely segregated and isolated from each other, from a security point of view, by granting us complete control over traffic flow and management.

**Easy CI/CD**

With the help of a Docker, we can build a container image and can further use that same image over every step of the deployment process. The advantage of it is the ability to separate non-dependent steps and also run them in parallel. In addition, the duration of time it takes from build to production may speed up notably. While it comes to Continuous Integration, Docker works well as part of its pipelines along with tools such as Travis, Jenkins, and Wercker. These tools can save the new version as a Docker image, every time our source code is updated, just tag it with a version number and push to Docker Hub, then deploy it to production.

Continuing with the disadvantages:

**Data in a container**

There are times when a container goes down, so after that, it needs a backup and recovery strategy, although there are several solutions for that they are not automated or not very scalable yet.

**Learning curve**

Some developers find that switching to Docker containers can have quite a steep learning curve. Even those that are thoroughly familiar with VM infrastructure can find some of the Docker concepts challenging to get to grips with. That’s why working with a user-friendly container-based tool can be the key to making the most out of the Docker environment.

**Not all applications benefit from containers**

In common, the applications that are intended to work as a collection of thoughtful microservices attain to get the most from containers. Contrarily, Docker’s one real benefit is that it can interpret application delivery by giving an easy packaging mechanism.

With all the above been said, some of you might migrate to Docker and some not. However, as I mentioned before integrating to Docker might be intimidating from first hand. But it gets better and better once you approve and understand the concept of it.

To get a better understanding of this research I have split the main questions into smaller questions.  
Together with the questions I am going to be giving a snippet of the way I have done to solve the question.

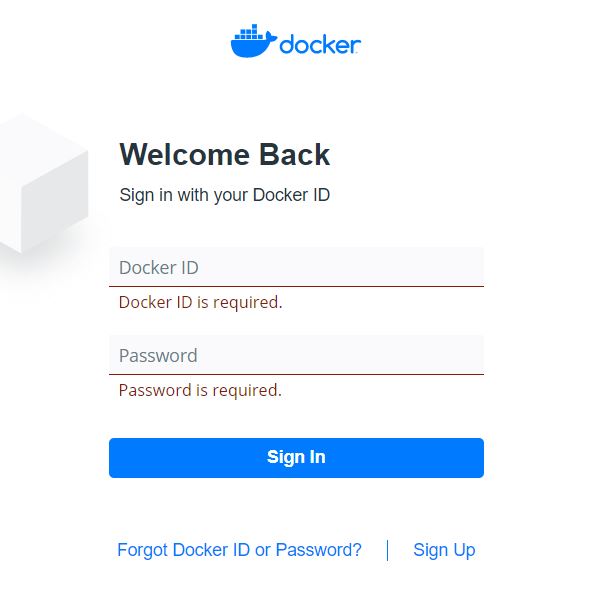
* How I create a Dockerfile?
* How I create a Docker-Hub Repository?
* How I create a Docker workflow in GitHub?
* How I deploy my application automatically to Docker-Hub Repository?

Creating a Dockerfile can sometimes be very confusing. A lot of projects have different usage and construction and that means that they need a different Dockerfile setup. The structure of a Dockerfile is more or less the same, the only thing that changes is the content of the project.

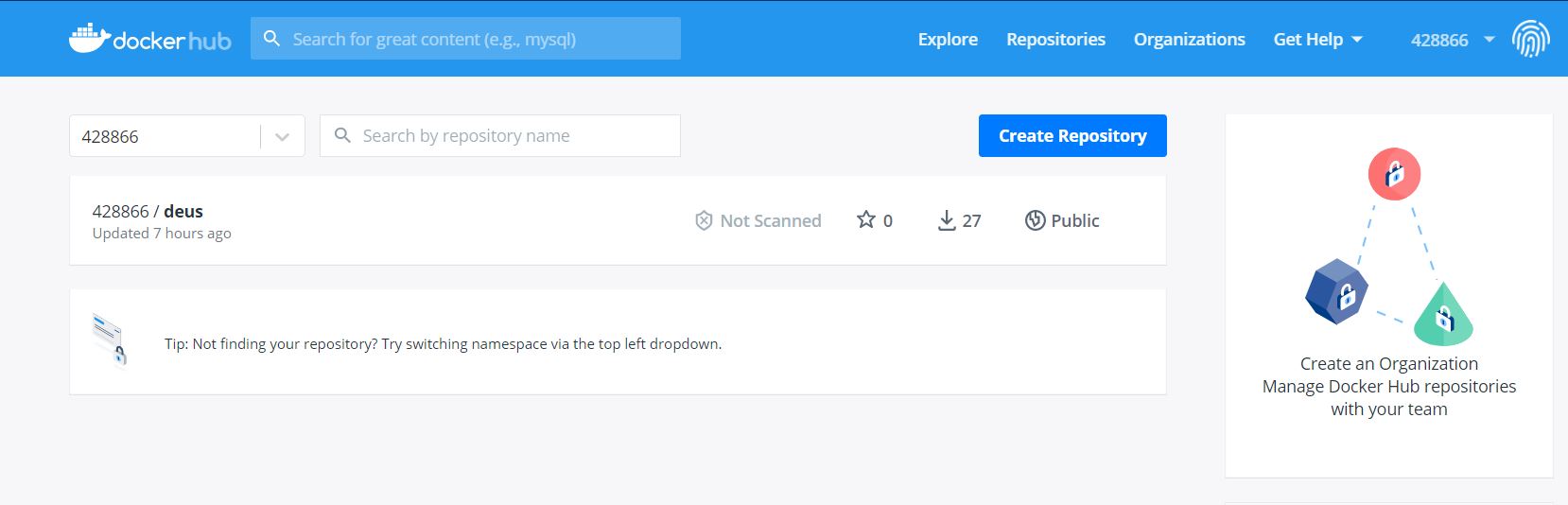


* FROM - defines the base image used to start the build process
* WORKDIR - sets the path where the command, defined with CMD, is to be executed
* COPY - copies the files from a source on the host into the container's own filesystem at the set destination
* RUN - central executing directive for Dockerfiles
* EXPOSE - associates a specific port to enable networking between the container and the outside world
* ENTRYPOINT - sets a default application to be used every time a container is created with the image

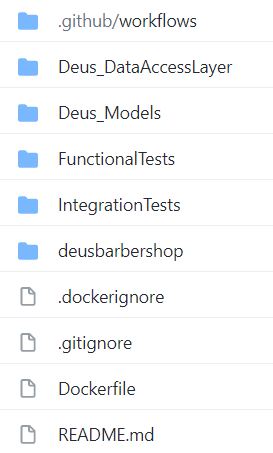
All the above commands are contributing into creating and building an image, which holds my application in it. After we have created the Dockerfile in the root directory of our project, I am now going to show you how to create a docker-hub repository. Before, we create a repository, first you have to login into your Docker-Hub account.



Once logged in, the home page will appear. As you can see I already have created my repository but I will show you exactly how I did it.



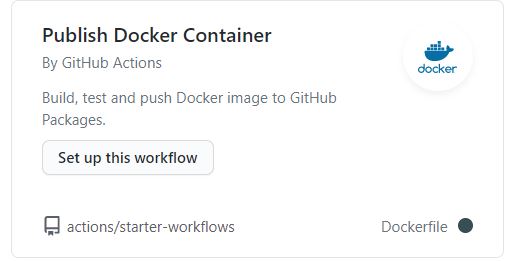
Pressing the blue button **Create Repository,** it will create a new repository with the name that we give it. Moreover, after we have created our repository it’s time to continue with the next question.   
In order to achieve an automated deployment, we have to own a GitHub account with the desired repository that we would like to deploy on Docker-Hub.



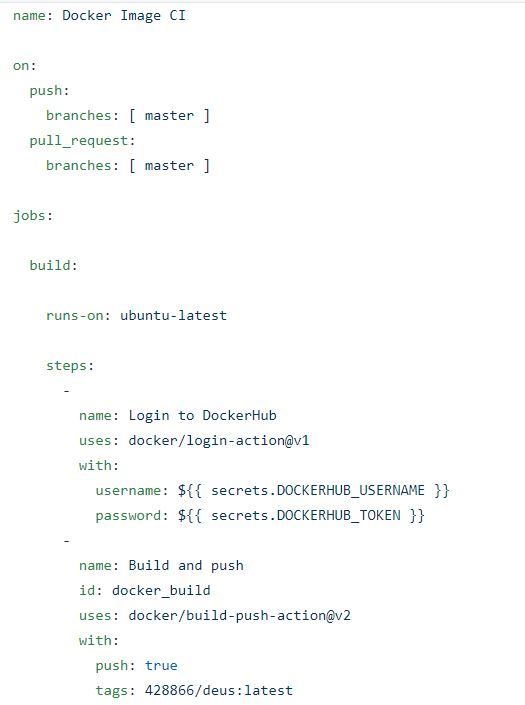
The snippet above is my own repository structure. On top of it there is a directory called workflows. In this section we will focus on how to create and manage a workflow for our project. But first let’s explain workflows. What is a workflow? A workflow is a configurable automated process made up of one or more jobs. You must create a YAML file to define your workflow configuration. So, basically workflows help automate the build and deployment of our project. How to add a workflow file?



In the nav bar of our repository we can see **Actions.** Navigating to Actions we can then create a new workflow of our own choice. The one that we need is this one:



At completion of this step, the workflow directory appears on top of our project with a YAML file that we have to configure in order to automate our deployment. In order for this process to succeed we have to configure our YAML file so that every time we push something new to our repository it will re-build it and push it to our Docker-Hub Repo automatically.



The snippet above shows how my project gets automatically pushed to Docker-Hub via GitHub.  
What this image says is:

* 1. Push & Pull requests will affect the master repository
  2. It should build the project based on the operating system it runs. (This case ubuntu-latest)
  3. Step 1: Login to Docker-Hub with the given credentials.
  4. Step 2: Build and push the project to the repository given on the **tags**:

In conclusion, the questions above are all problems I faced as a new user to Docker. The steep learning and the different uses of docker from other developers has made the problem level a bit more complicated to discover online (i.e., StackOverFlow).

How (Methods & Strategies)

|  |
| --- |
| I am using the DOT Framework methods and strategies in order to have a complete research. DOT Framework uses 5 different research strategies:  **Library** |
| Library research is done to explore what is already done and what guidelines and theories exist that could help you further your design. Since the advent of the internet library research is also called desk research. |

|  |
| --- |
| **Field** |
| Field research is done to explore the application context. You apply a field strategy to get to know your end users, their needs, desires and limitations as organizational and physical contexts in which they will use your product. |

|  |
| --- |
| **Lab** |
| Lab research is done to test parts or concepts of your product, of the final product. You use lab research to learn if things work out the way you intended them, or to test different scenarios. |

|  |
| --- |
| **Showroom** |
| Showroom research is done to test your ideas in relation to existing work. Showing your prototype to experts can be a form of showroom research or spelling out how your product is different from the competition. Also testing your product to general guidelines is a form of showroom research. |

|  |
| --- |
| **Workshop** |
| Workshop research is done to explore opportunities. Prototyping, designing and co-creation activities are all ways to gain insights in what is possible and how things could work. |

Methods & Strategies

In order to achieve a good research document, I have chosen to follow the DOT Framework strategies.  
The ones that I chose to conduct this research are:

+ [](https://ictresearchmethods.nl/File:Logo-field.png) + [](https://ictresearchmethods.nl/File:Logo-lab.png) + [](https://ictresearchmethods.nl/File:Logo-showroom.png)



**References**

* [**Docker**](https://www.docker.com/)
* [**Docker-Hub**](https://hub.docker.com/)
* [**Setup docker in GitHub Action**](https://docs.docker.com/ci-cd/github-actions/)
* [**build-push-action**](https://github.com/docker/build-push-action)
* [**creating-a-docker-container-action**](https://docs.github.com/en/actions/creating-actions/creating-a-docker-container-action)
* [**Containers Pros & Cons**](https://blog.iron.io/docker-containers-the-pros-and-cons-of-docker/)