ASP.NET Core & Docker

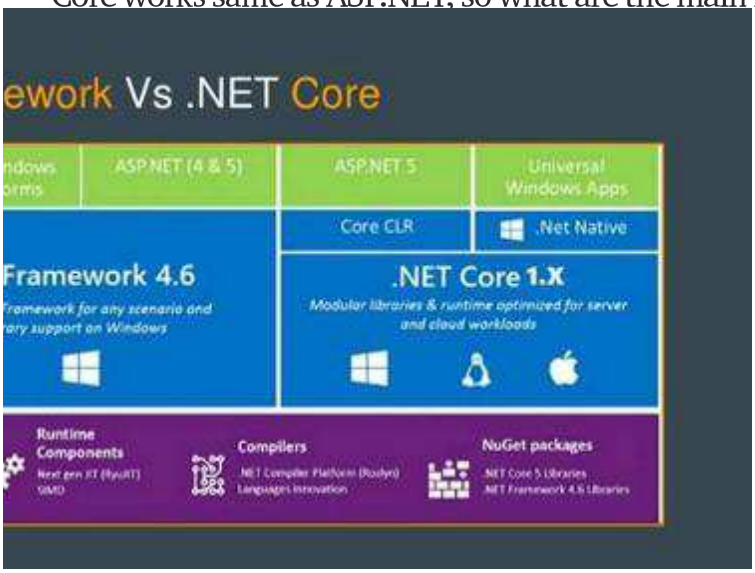
Lucas Palacios Follow
Jun 16, 2018 · 7 min read

In this guide, I will help you create your first app with ASP.NET Core and Docker. I believe that if you are here you are familiar with ASP.NET, but what is ASP.NET Core? What is Docker? How will they change my work?

. . .

ASP.NET Core

ASP .NET Core is a new technology, separated from the already known .NET Framework. Besides, Core works same as ASP.NET, so what are the main reasons for working with it?



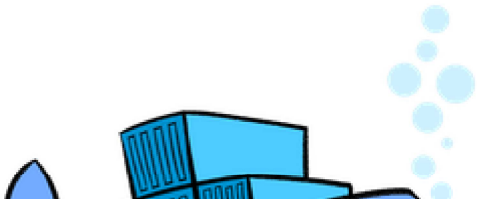
- Multi-Platform: The apps we create will work on Windows, Linux and Mac.
- Cloud oriented: .NET Core was designed to work combined with cloud and with this comes a higher performance than the 'traditional' ASP.NET.

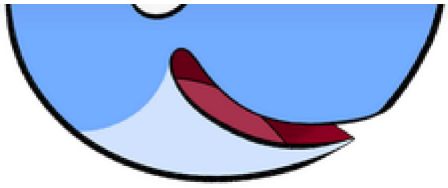
For installation click here, download the package and install it in a few minutes.
So we can say that this is an improvement from the ASP.NET Freamework where we can work with the cloud and have less problems with other platforms. Now, how can we take more advantage of this? Here is where we meet **Docker.**

. . .

Docker

So, what is Docker and how is this related with the cloud service?
Docker is the company driving the container movement and the only container platform provider to address every application across the hybrid cloud.
Besides of the official web page description, we can say Docker is a software platform that holds everything as a container. That's why we see these containers on the top of the whale. So what are this containers?





. . .

Containers

Containers are the essential part of Docker, they contain:

- The Operating System
- The app that we are building
- An environment variable

So with this, we can name and work with them. Also, we can freely download these containers on Docker's official page. We can start running in the environment that we are working on and start working in our app in no time.

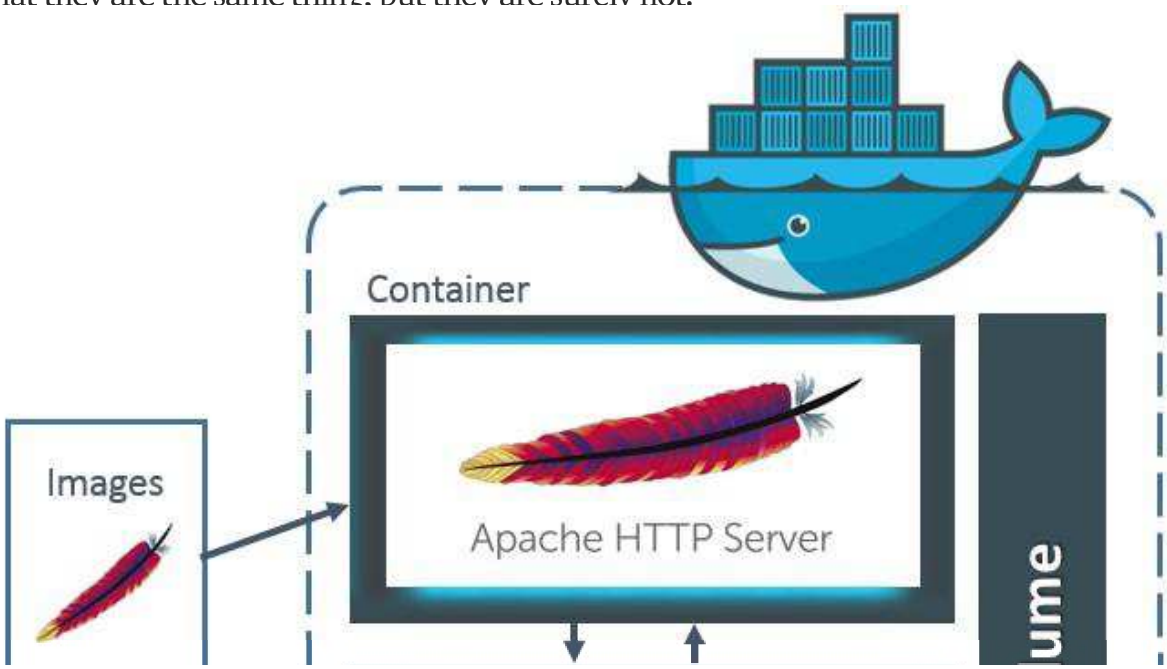
In another words, we can think of them as a box with the essential things that we need to start a project and we will have unlimited amount of them so we can start our work with the same base project many times.

For more detailed guide on containers, click here

. . .

Images

While we talk about containers there's a word that comes with it: Images. So, What are images? Images are the base of the containers which hold files that will run the container. The images can't turn on or off as well as they can not connect with each other like containers . There are many images on Docker's official page that you can download. So, you will read, listen and see many times that they are the same thing, but they are surely not.

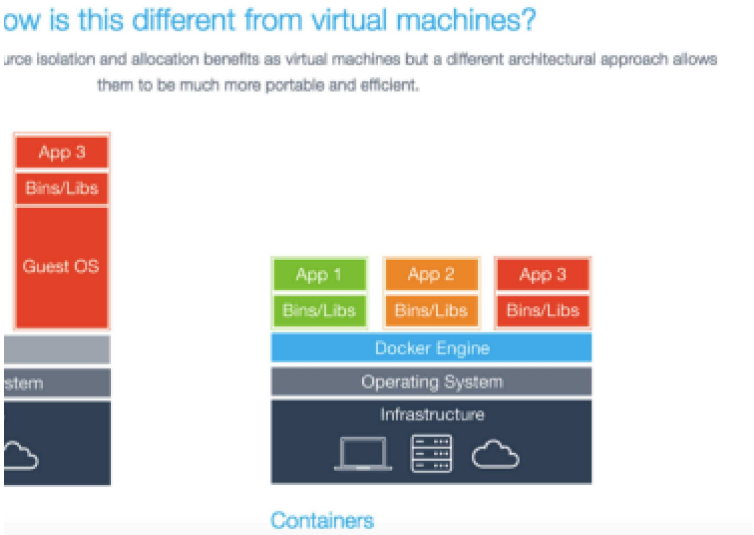


Here I leave you the official Docker where you can find a more detailed information.
Images
Containers

. . .

Let's go deeper in Docker

So this containers are Virtual Machine images? NO

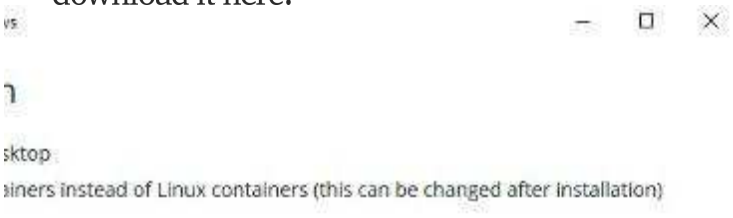


The virtual machines, having an infrastructure, will have a host Operating System and on top of it comes the Hypervisor which can be the virtual machine or the Hyper-V, then they have to work with the Guest Operating System to deliver our app. Also, all of this makes a huge amount of GBs to be working with.
The Containers include the application and all of its dependencies, running as isolated processes in user's space on the host Operating System.
With Docker we forget about working for different platforms and the Docker Engine makes all the magic for us to forget about the guest OS.
This is getting interesting, isn't it? Now, if we work with this plus ASP. Net Core there will be a free environment. Let's try, shall we?

. . .

Get ready your Docker

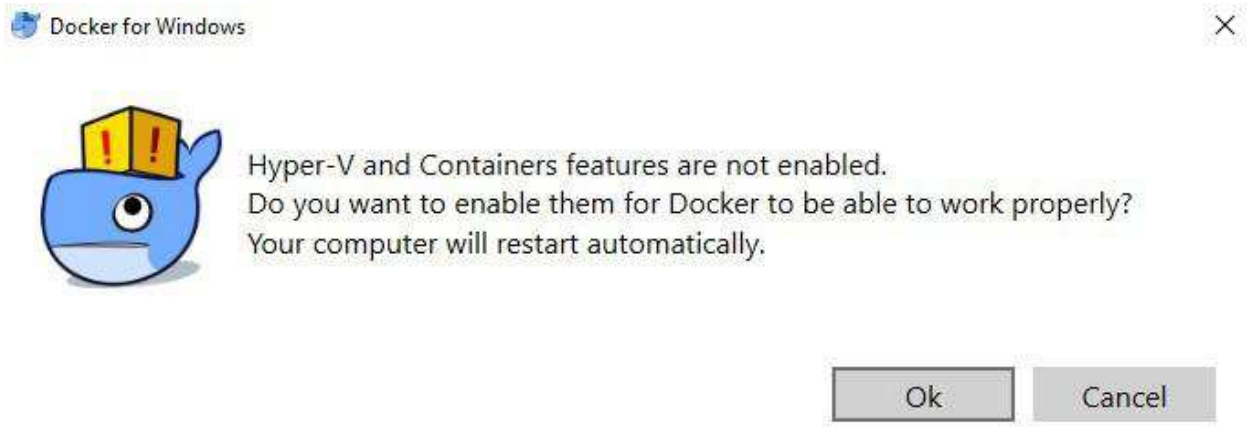
Unlike ASP.NET Core, Docker is a little bit more complicated to install. First of all, we have to download it here.



Now, personally I don't recommend using the Windows containers just because they didn't work fine in my computer and by not choosing this option in the settings I could start working in my app. But this

Windows containers.

And if you are on **Windows** please **don't** forget to activate the **Hyper-V.** Docker will ask you to activate it but if you are distracted remember this.

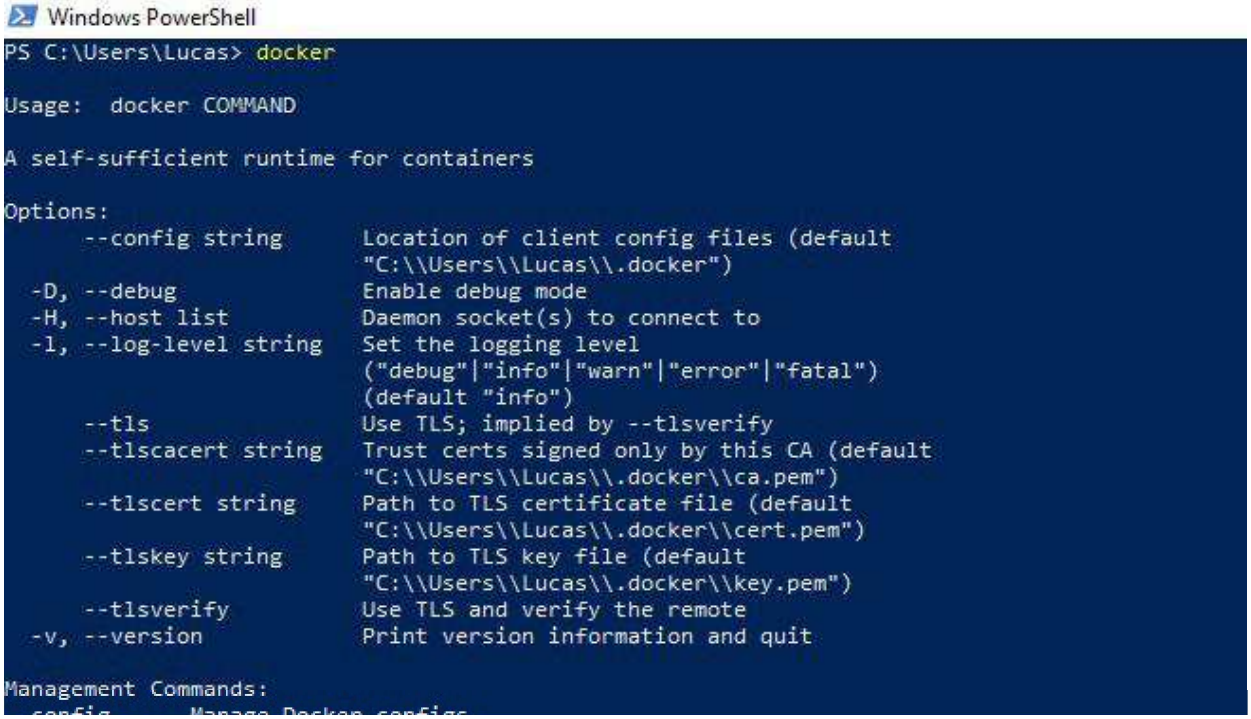


Now that we have Docker installed we need to get into it by learning some of the basic commands that you may use before starting our project.

. . .

Docker Commands

First of all you can use the Command Console or PowerShell if you are in Windows. If you have the possibility to choose use PowerShell as it will be easier to write with than the Command Console. Let's have a quick view of the commands, you can write *docker* in the console and you will get the full list of the commands with their description.



```
  plugin      Manage plugins
  secret      Manage Docker secrets
  service     Manage services
  swarm       Manage Swarm
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes

Commands:
  attach      Attach local standard input, output, and error streams to a running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  ps          List containers
  pull        Pull an image or a repository from a registry
  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
```

Remember that for cleaning the console you just have to enter *cls*.

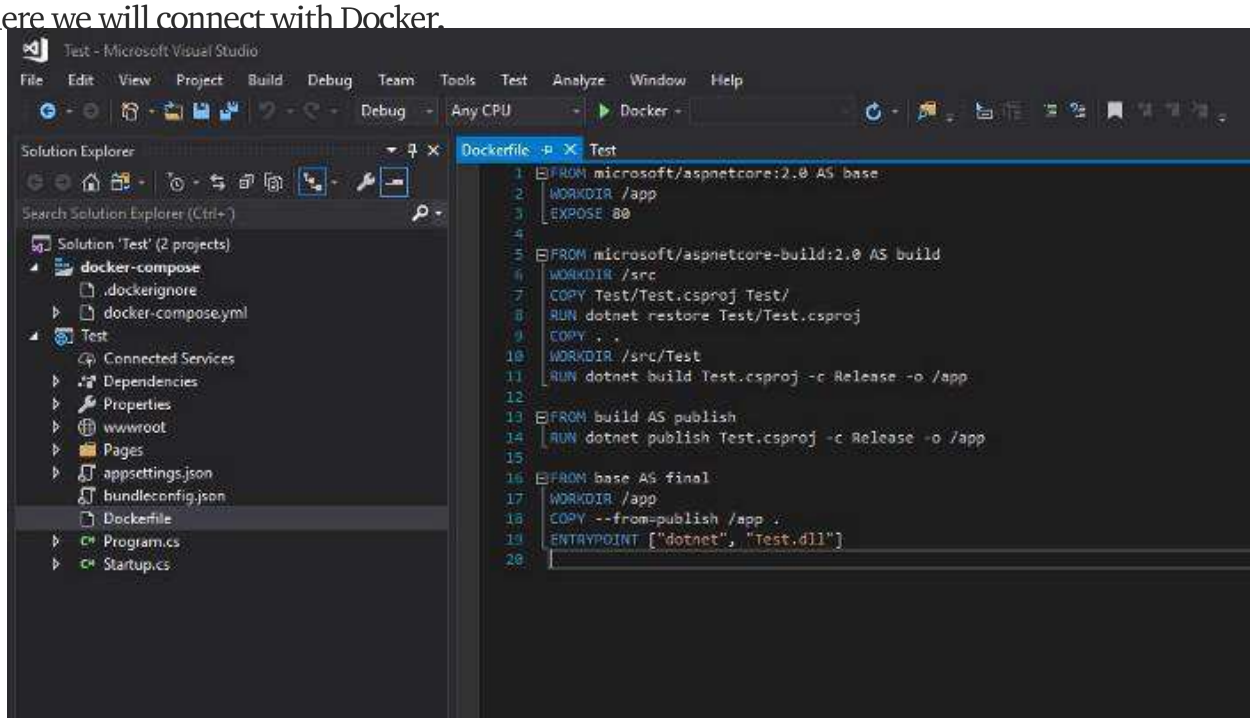
. . .

First Page with ASP.NET Core & Docker

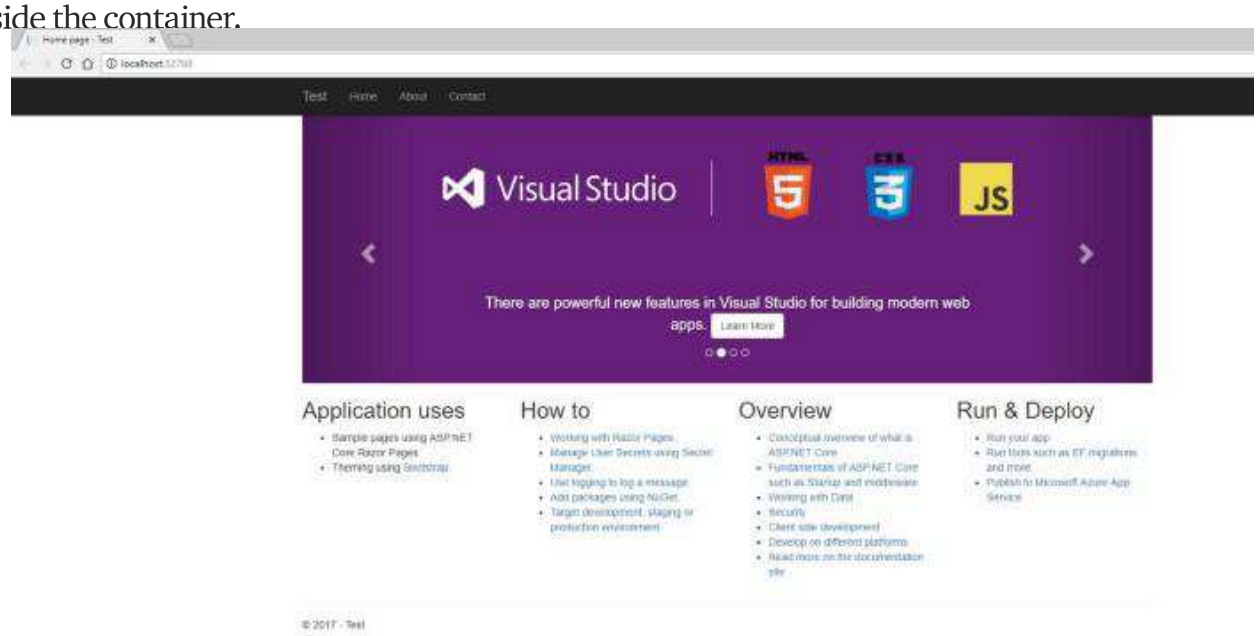
Now lets make a simple example.

Open Visual Studio and start a new project ASP.NET Core Web Application. You will see two options: One to Enable Docker Support and another where you can select what container you want to use. I will make the same suggestion than before, use Linux.

Having created the project, you will see a new document called Dockerfile that, as you imagine, is where we will connect with Docker.



Now the start button says Docker. If we click, the button will automatically build an image using our application,creating a container using that same image, and our application will start running inside the container.

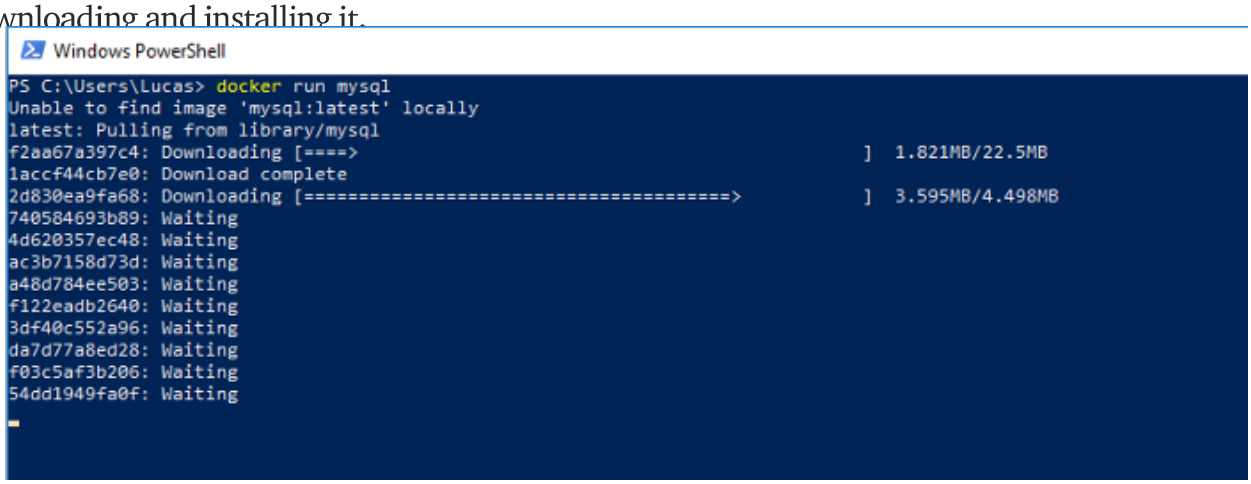


We can check all of this with the command *ps* in Power Shell where we will see the Containers ID and the command *images* to see the images of our project.

• • •

Let's Make Something More Difficult

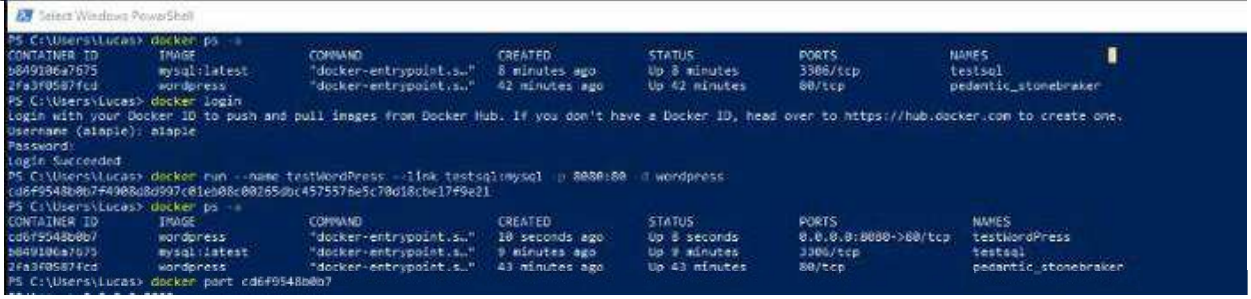
Now we will create two containers and make them connect, all by commands with Docker.
First of all, let's download the images that we will use.
One option is to go to Docker's page and download them right there, if not, it can be done from the commands. If you know what image you want to use for your container you have to type *docker run imageName,* by doing this, docker will see that we don't have the image so it will start downloading and installing it.

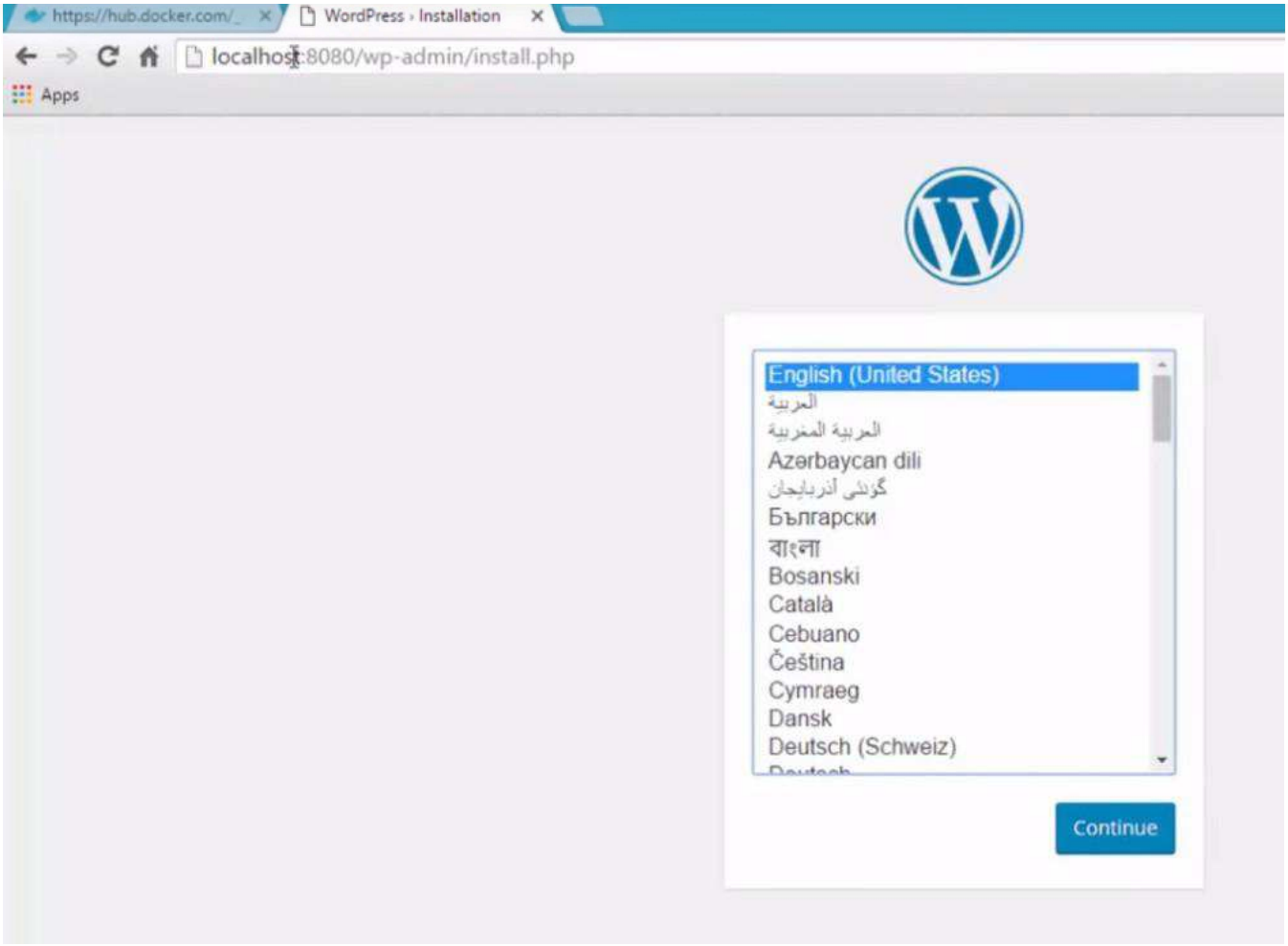


We will download MySQL and WordPress. Another command to download the images is the command *Docker pull imageName*.
Next step is running the MySQL servers with the command:
docker run — name mysqlname **-e MYSQL_ROOT_PASSWORD=**abc123 **-d mysql:latest**

Still didn't run the WordPress Image

Also, you can see in this example where I remove a container with the command *rm* and for that I must use the Container ID. At the end, I was checking the images and the containers were all right to continue.
Finally, we will connect the two containers to run the WordPress, first login our account and then with the command:
docker run — name mylocalPage **— link** mysqlname**:mysql -p** port **-d wordpress**





Now, we can install WordPress in our container and also there is a connection between the container and the mysql container Data Base.

. . .

Personal Opinion

If you followed me from the beginning you can see now the potential of Docker with ASP.NET Core. I think this is a way to have less problems while working on multi-platforms and only thinking about our app. It's also easy to understand. While doing this I had a ton of problems connecting and downloading, but all of this was solved by different forums that had the same problems as I did with many solutions. Docker is a power full tool and with Core you can make great apps, so don't give up here.

. . .

Finally, the links that helped me making this guide

My job here was to make an introduction, but if you want to know more you can click this links to understand how I worked here. Thanks for reading and keep improving!

- Introduction to ASP.NET Core
- Get started with ASP.NET Cores
- Windows Containers