Docker is a container management service. The keywords of Docker are **develop, ship** and **run** anywhere. The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.

The initial release of Docker was in March 2013 and since then, it has become the buzzword for modern world development, especially in the face of Agile-based projects.

Features of Docker

* Docker has the ability to reduce the size of development by providing a smaller footprint of the operating system via containers.
* With containers, it becomes easier for teams across different units, such as development, QA and Operations to work seamlessly across applications.
* You can deploy Docker containers anywhere, on any physical and virtual machines and even on the cloud.
* Since Docker containers are pretty lightweight, they are very easily scalable.

## Components of Docker

Docker has the following components

* **Docker for Mac** − It allows one to run Docker containers on the Mac OS.
* **Docker for Linux** − It allows one to run Docker containers on the Linux OS.
* **Docker for Windows** − It allows one to run Docker containers on the Windows OS.
* **Docker Engine** − It is used for building Docker images and creating Docker containers.
* **Docker Hub** − This is the registry which is used to host various Docker images.
* **Docker Compose** − This is used to define applications using multiple Docker containers.

**DOCKER HUB**

Docker Hub is a registry service on the cloud that allows you to download Docker images that are built by other communities. You can also upload your own Docker built images to Docker hub.

In Docker, everything is based on Images. An image is a combination of a file system and parameters. Let’s take an example of the following command in Docker.

docker run hello-world

* The Docker command is specific and tells the Docker program on the Operating System that something needs to be done.
* The **run** command is used to mention that we want to create an instance of an image, which is then called a **container**.
* Finally, "hello-world" represents the image from which the container is made.
* docker images
* This command is used to display all the images currently installed on the system.

## Removing Docker Images

The Docker images on the system can be removed via the **docker rmi** command. Let’s look at this command in more detail.

docker rmi

This command is used to remove Docker images.

### **Syntax**

docker rmi ImageID

### **Options**

* **ImageID** − This is the ID of the image which needs to be removed.

### **Return Value**

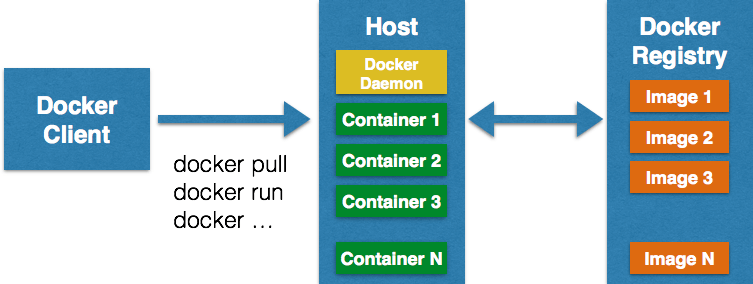
The output will provide the Image ID of the deleted Image.

docker ps (It returns the currently running cnontainer )

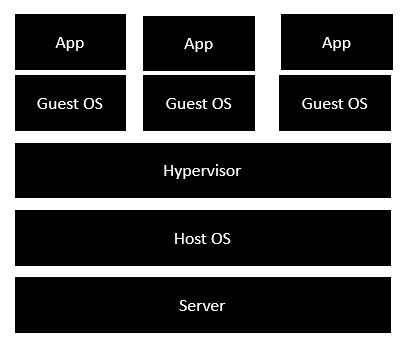
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

Docker ps –a (Lists all the available container on the system..)

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMESb6ef2ec5c2da fce289e99eb9 "/hello" 9 minutes ago Exited (0) 9 minutes ago priceless\_golick

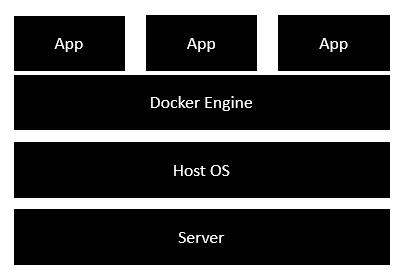
73b973cee383 fce289e99eb9 "/hello" 9 minutes ago Exited (0) 9 minutes ago wonderful\_wilbur

The following image shows the standard and traditional architecture of **virtualization**.



* The server is the physical server that is used to host multiple virtual machines.
* The Host OS is the base machine such as Linux or Windows.
* The Hypervisor is either VMWare or Windows Hyper V that is used to host virtual machines.
* You would then install multiple operating systems as virtual machines on top of the existing hypervisor as Guest OS.
* You would then host your applications on top of each Guest OS.

The following image shows the new generation of virtualization that is enabled via Dockers. Let’s have a look at the various layers.



* The server is the physical server that is used to host multiple virtual machines. So this layer remains the same.
* The Host OS is the base machine such as Linux or Windows. So this layer remains the same.
* Now comes the new generation which is the Docker engine. This is used to run the operating system which earlier used to be virtual machines as Docker containers.
* All of the Apps now run as Docker containers.

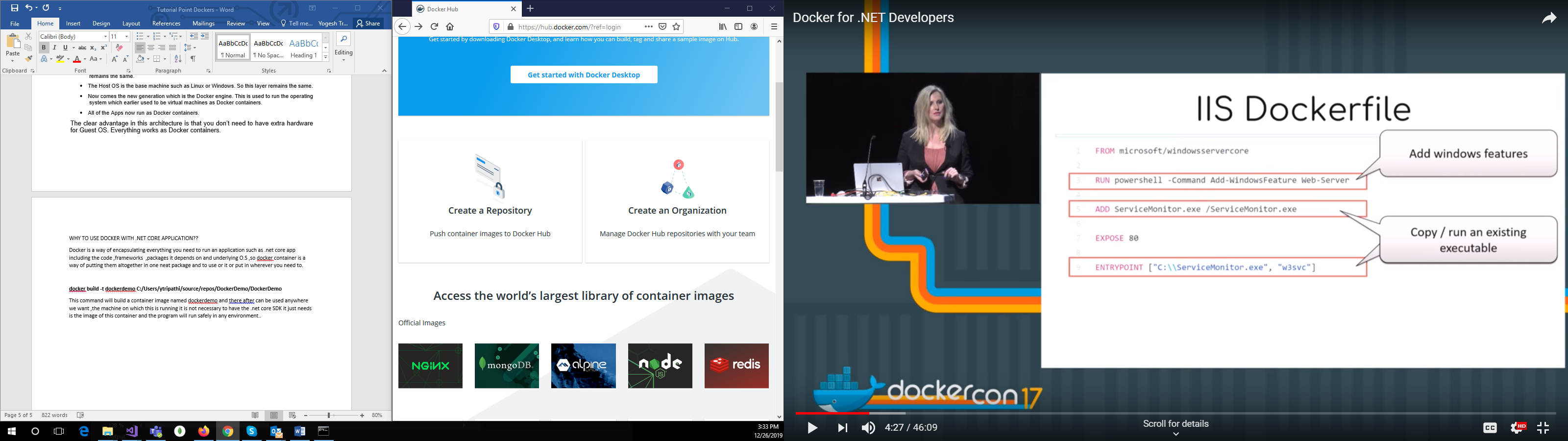
The clear advantage in this architecture is that you don’t need to have extra hardware for Guest OS. Everything works as Docker containers.

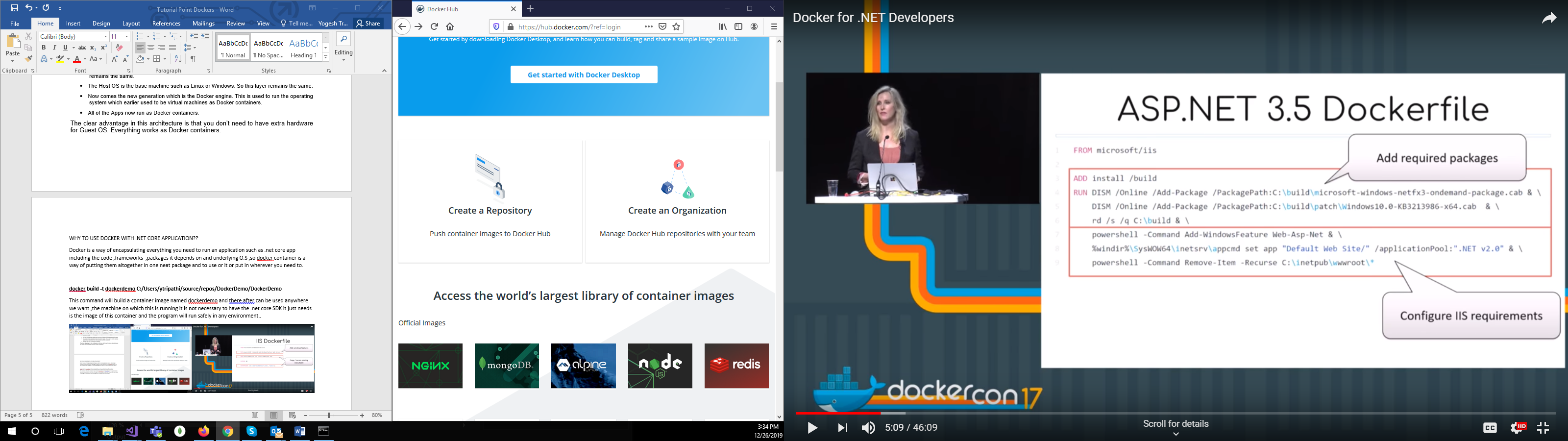
WHY TO USE DOCKER WITH .NET CORE APPLICATION??

Docker is a way of encapsulating everything you need to run an application such as .net core app including the code ,frameworks ,packages it depends on and underlying O.S ,so docker container is a way of putting them altogether in one neat package and to use or it or put in wherever you need to.

**docker build -t dockerdemo C:/Users/ytripathi/source/repos/DockerDemo/DockerDemo**

This command will build a container image named dockerdemo and there after can be used anywhere we want ,the machine on which this is running it is not necessary to have the .net core SDK it just needs is the image of this container and the program will run safely in any environment..





**DOTNET CONTAINER….**

D:\Dockers\ScramNet.Gps.AnalyticsApi\ScramNet.Gps.AnalyticsApi>docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

scramnetgpsanalyticsapi dev f1de8590141e 25 minutes ago 493MB

<none> <none> 868cd67b5394 38 minutes ago 1.76GB

<none> <none> e5619371734a 48 minutes ago 493MB

microsoft/dotnet 2.2-sdk-nanoserver-1709 cc5bae1219e2 8 months ago 1.76GB

microsoft/dotnet 2.2-aspnetcore-runtime-nanoserver-1709 09d6f69bf265 8 months ago 493MB

D:\Dockers\ScramNet.Gps.AnalyticsApi\ScramNet.Gps.AnalyticsApi>docker container ls

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

2833720b7e29 scramnetgpsanalyticsapi:dev "C:\\remote\_debugger\\…" 21 minutes ago Up 20 minutes 0.0.0.0:22108->80/tcp festive\_shaw