|  |
| --- |
|  |
| Explore Docker |
| Basics :: Installation: POC |
|  |
| **Vibranarayanan** |
| **5/16/2018** |

|  |
| --- |
|  |

Table of Contents

[Change Log 2](#_Toc514354048)

[Docker Terms 2](#_Toc514354049)

[Docker work flow 3](#_Toc514354050)

[Virtualization Vs Containerization 4](#_Toc514354051)

[Containers on VM 4](#_Toc514354052)

[Docker Client-server architecture 5](#_Toc514354053)

[Docker Benefits 5](#_Toc514354054)

[Portability 5](#_Toc514354055)

[Version Control 6](#_Toc514354056)

[Installing Docker Toolbox in Windows 6](#_Toc514354057)

[Pre requisites 6](#_Toc514354058)

[VM virtual box with Linux 674 bit 6](#_Toc514354059)

[Running Docker Linux Vs Window 6](#_Toc514354060)

[Current system Configuration 7](#_Toc514354061)

[Hyperv detection Tool 7](#_Toc514354062)

[Download & Install Docker Tool 7](#_Toc514354063)

[Install Docker Tool 7](#_Toc514354064)

[Step by Step Installation screen 8](#_Toc514354065)

[Verify Docker commands after installation 10](#_Toc514354066)

[Docker Hello-world 10](#_Toc514354067)

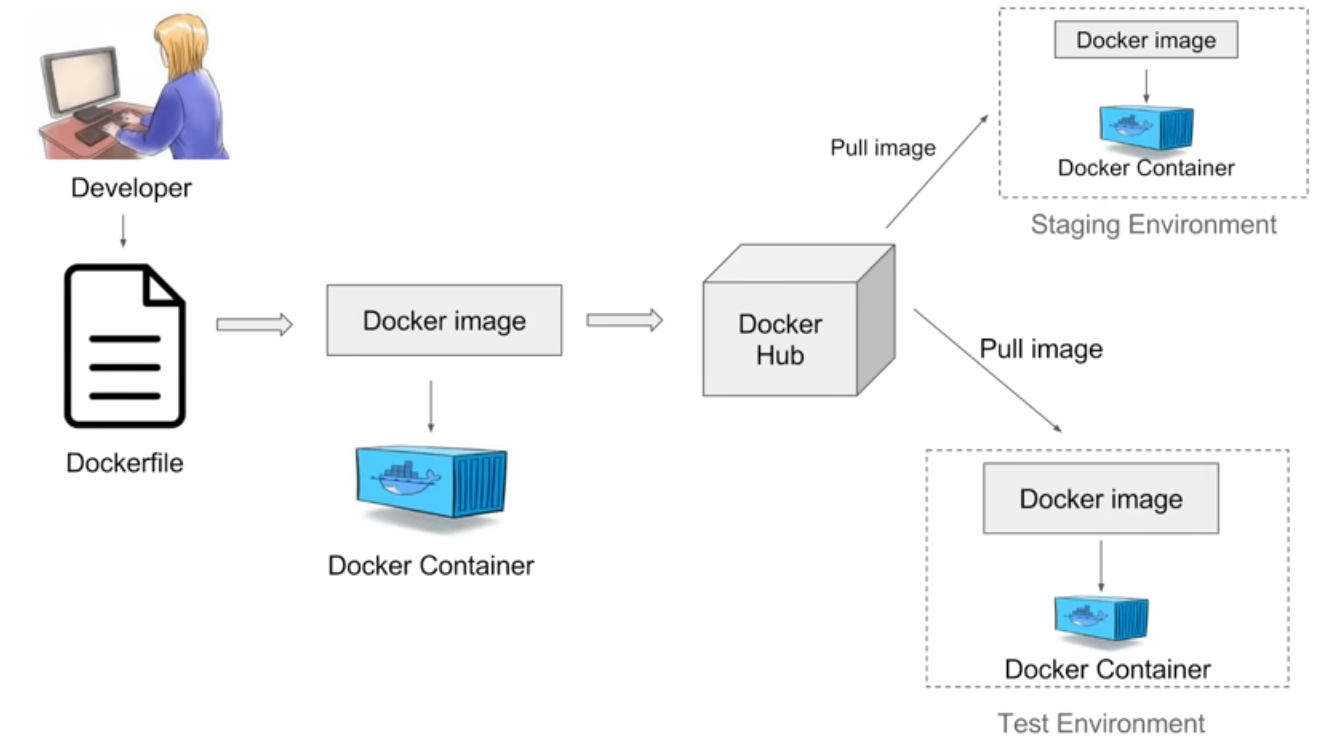
## Change Log

|  |  |
| --- | --- |
| Date | Details |
| 05/16/2018 | * Docker terms * Docker work flow * Virtualization vs Containerization * Client server architecture * Benefits |
| 05/17/2018 | * Docker Tool installation on windows * verify Docker commands after installation * Running hello world Docker image * Learn how to create Docker file * Running Docker file with existing repo image.   Pages (6 to 10) |

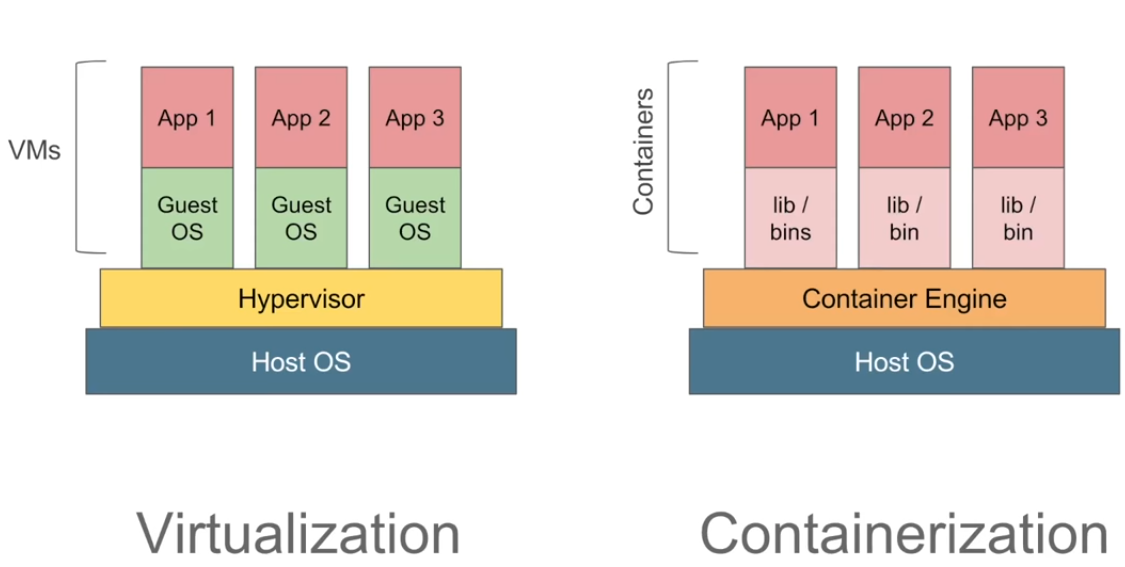
## Docker Terms

|  |  |
| --- | --- |
| Term | Explanation |
| Docker file | * Describes steps to create Docker image |
| Docker image | * Created using Docker file * An **image** is an executable package that includes everything needed to run an application--the code, a runtime, libraries, environment variables, and configuration files. dependencies |
| Docker container | * Created by running Docker image. * Run time instances of Docker image. * A **container** runs natively on Linux and shares the kernel of the host machine with other containers. It runs a discrete process, taking no more memory than any other executable, making it lightweight. |
| Docker Hub | * Online repository, images can be stored in this repository |
| Docker Client | * Command line inter face to interact with Docker server. |
| Docker Server/Daemon | * Will have all the containers |
| Docker Engine | * Combination of Docker Client and server component. |

## Docker work flow



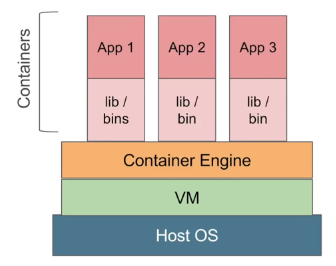
## Virtualization Vs Containerization



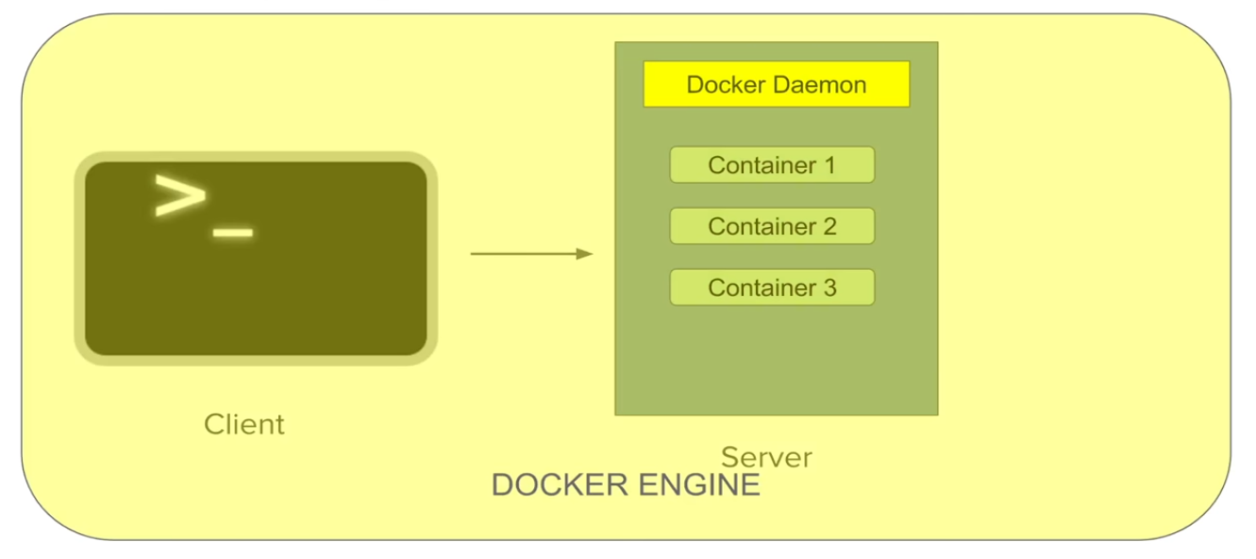
|  |  |
| --- | --- |
| **Virtualization** | **Containerization** |
| * Create Multiple VMs using Hypervisor in Host OS * VMs have their own OS and does not use host OS. * Overhead on Host platform * Each VM fixed memory need to be allocated. this leads to wastage of memory. | * Containers are light weight * Container engine will manage/use Host OS configurations/memory based on applications in containers. * No memory/space overhead. |

## Containers on VM

There may be a scenarios like container need to be run on VM. In that case Containers can be managed in below stacks.



## Docker Client-server architecture



* Docker server/Docker Daemon receives command from client in this form of CLI or rest API.
* All the Docker Client and server together form Docker engine.
* Docker client or Docker Daemon can be present in same host or in different host.

## Docker Benefits

* An Application inside a container runs on any system that Docker installed.
* Build application only once and no need to configure multiple times.
* Docker images can be maintained in any repository , later images can be pulled for usage.
* Test your application inside your container and ship it inside the container, This means Environment in which you test is identical with the app in production.
* *Isolation* is the Key, with Docker every application works in isolation in its own container. Does not interferes with other application running on the same system.
* Removal of an application is easy by deleting a container.
* Developer can package software with all its dependency and Docker will take care if running those application in different platform.
* *Productivity* is an another key. Docker allows faster and efficient deployment without worrying about, application running on different planform.

### Portability



* Docker containers can run on any platform. This can run on local system and Amazon ec2 or Google Cloud, etc..
* Container running on AWS can easily be ported to Virtual machine.

### Version Control

* Like Git Docker has in-build version control.
* Docker containers work just like a GIT repo, allow you to commit your changes into Docker images and version control them.
* Docker images can be tracked in other version control system like GIT.

## Installing Docker Toolbox in Windows

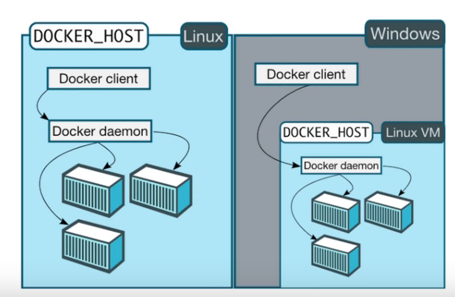
### Pre requisites

review system requirement and install in your windows system. your OS should be in 64 bit architecture. get installation information from this [Link](https://docs.docker.com/toolbox/toolbox_install_windows/#step-1-check-your-version)

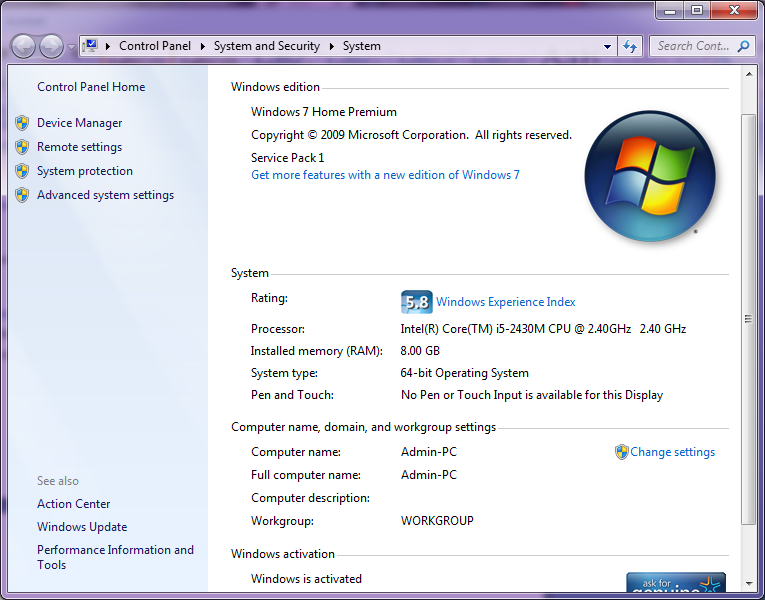
### VM virtual box with Linux 674 bit

from higher version of windows 7, by enabling Virtualization at BIOS level and disabling Hyper-vT in windows feature. you can start installing 64 bit version linux flavour. for more details refer this [link](http://www.fixedbyvonnie.com/2014/11/virtualbox-showing-32-bit-guest-versions-64-bit-host-os/#.Wv0vLsIQDIU)

### Running Docker Linux Vs Window

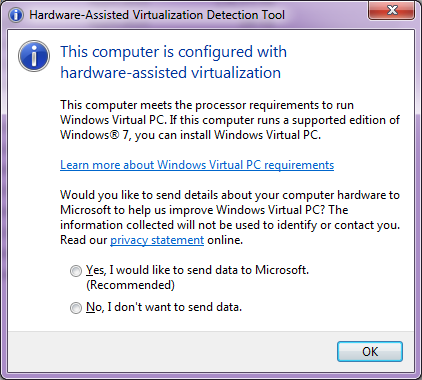


### Current system Configuration



### Hyperv detection Tool

As mention in this [link](https://docs.docker.com/toolbox/toolbox_install_windows/#step-2-install-docker-toolbox) under Window section downloaded havdetection tool and verified, i am good with run windows virtual PC. [Download URL](http://www.microsoft.com/en-us/download/details.aspx?id=592)



### Download & Install Docker Tool

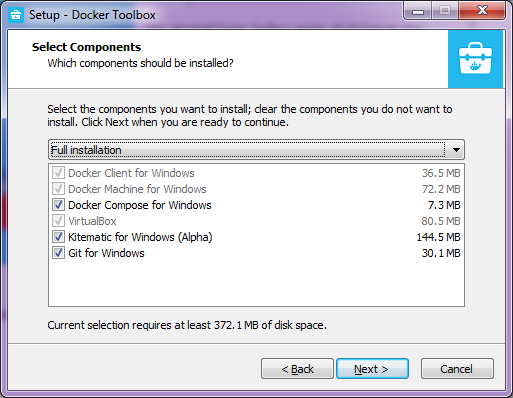
Down load Docker tool from this [URL](https://docs.docker.com/toolbox/toolbox_install_windows/)

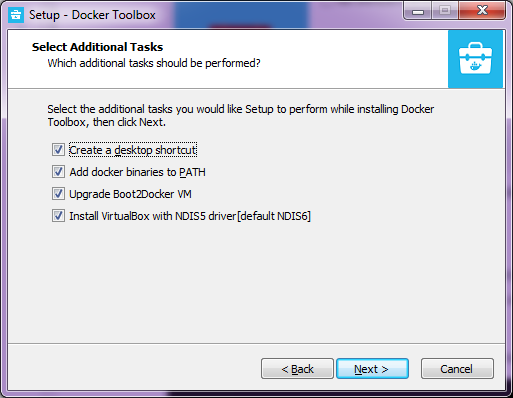
### Install Docker Tool

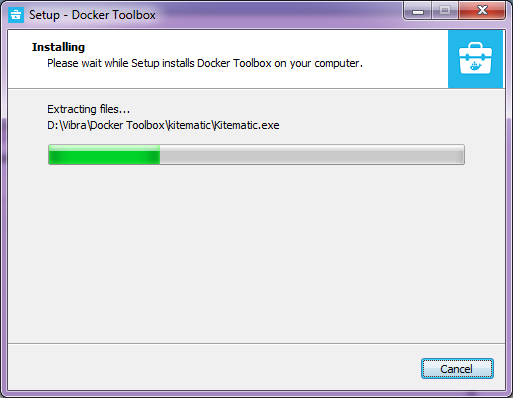
Install Down loaded docker tool.

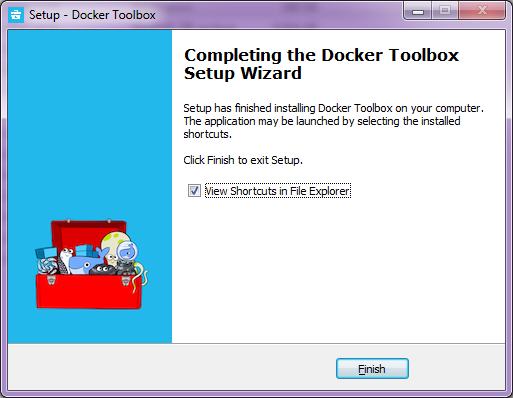
### Step by Step Installation screen

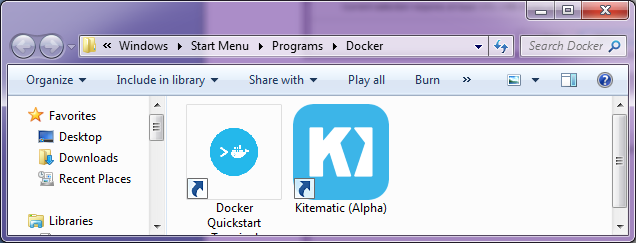




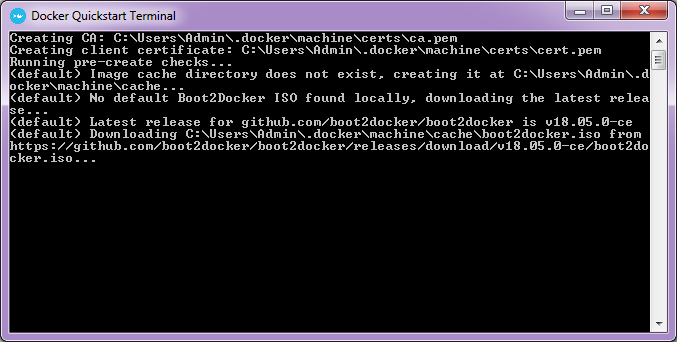




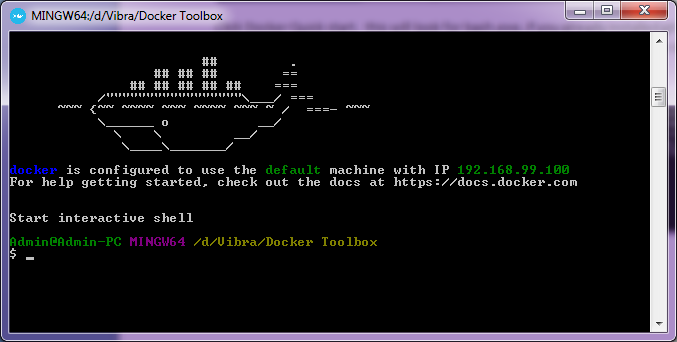




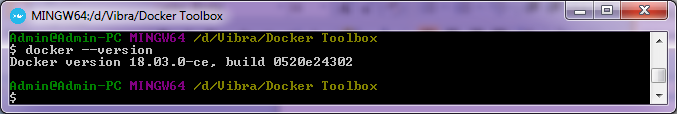
Click Docker Quick start. this will look for bash.exe, if you already installed git this will try to find that bash.exe location. if you sure about location you can point that location by browsing option provided.



#### Final screen after installation

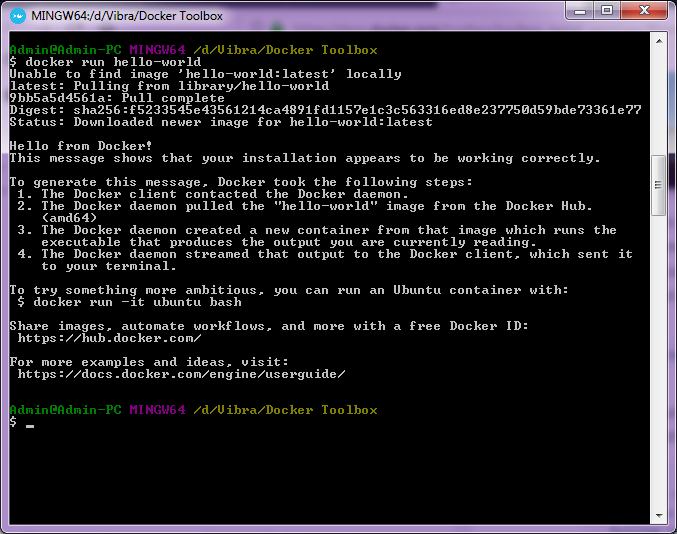


### Verify Docker commands after installation



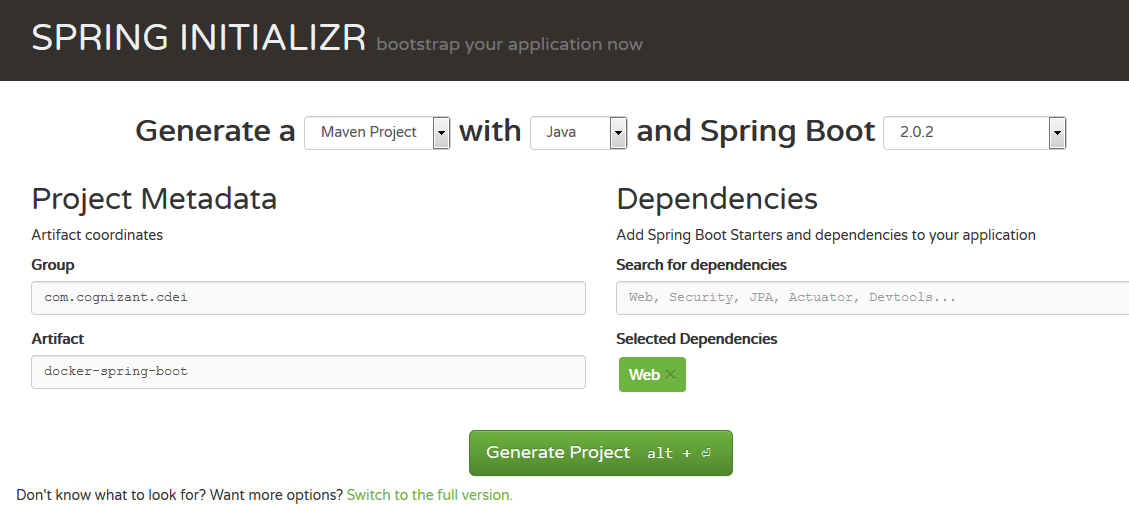
### Docker Hello-world

In command line type below to check the installation   
$docker run hellow-world  
This will look for local image, and if it doesn't find it will get it from common repro

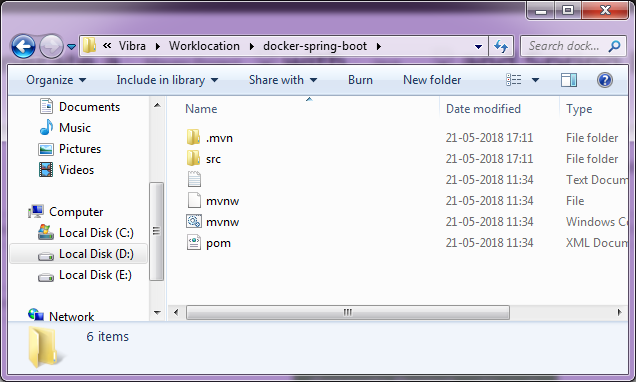


## Launching Simple Spring boot application in Docker

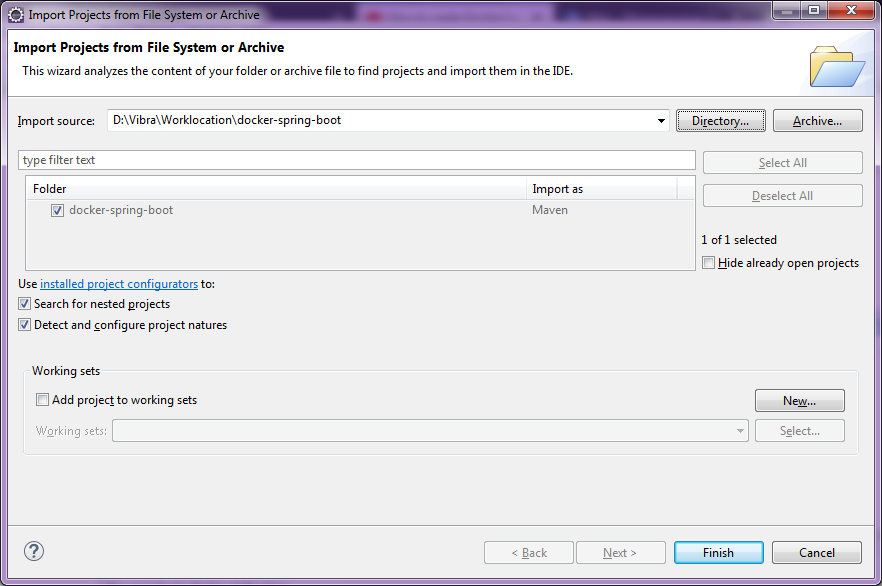
### Create Spring project structure

For easiest way go to http://start.spring.io/ , provide required details and click generate project. 

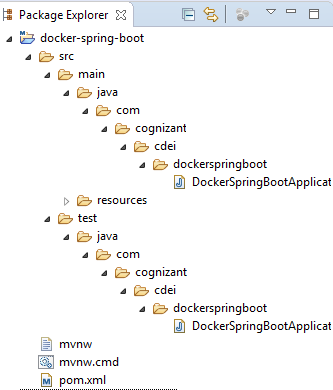
### Project structure in downloaded folder



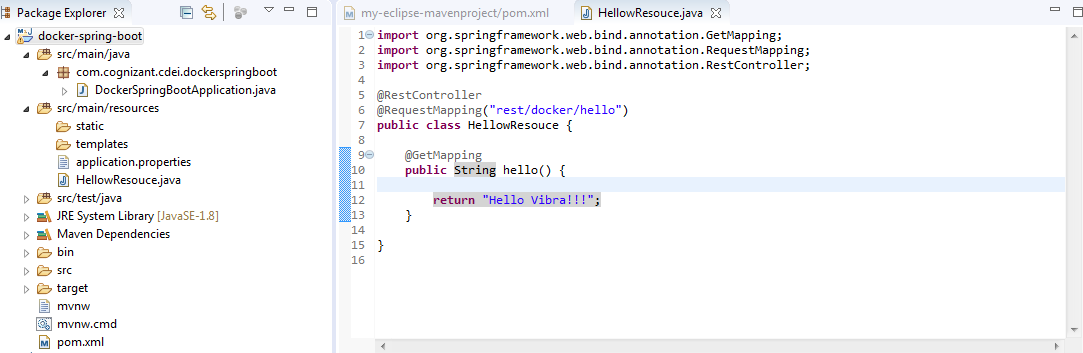
### Open this project in eclipse



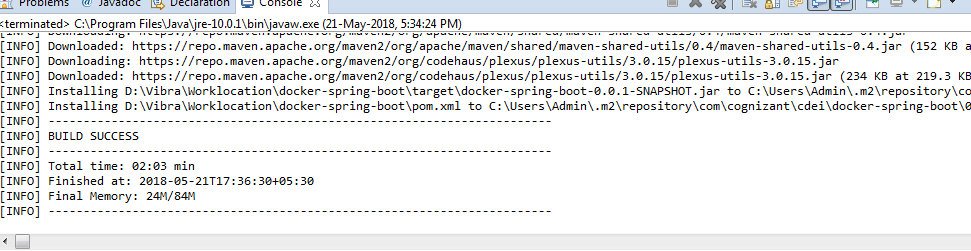
### Project Explorer



### Write a simple REST end point resource

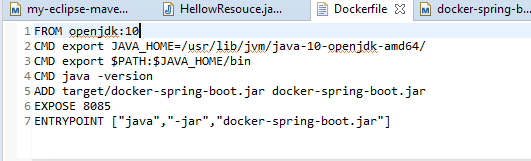
Create below class fine under resources package

### Build the project



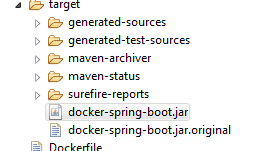
### Create a Docker File

Create a docker file with below commands to run created application in docker container.



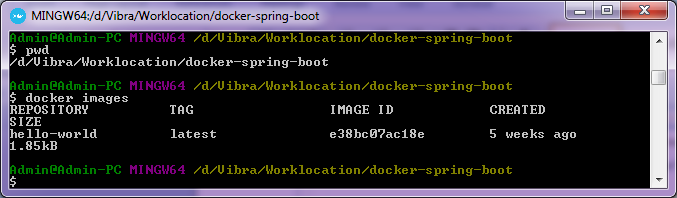
### Docker File Explained

* FROM openjdk:10 : This line will pull open JDK version 10 from docker hub. since my local environment created in using JSK 10.
* ADD : this line copy created jar file from "/target/docker-spring-boot.jar" to containers root directory with same name.

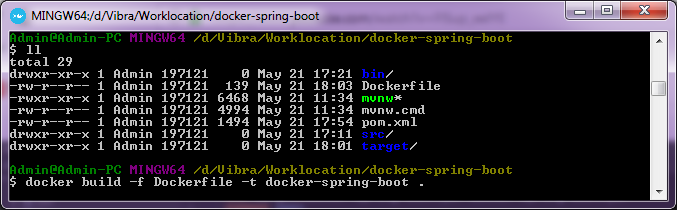


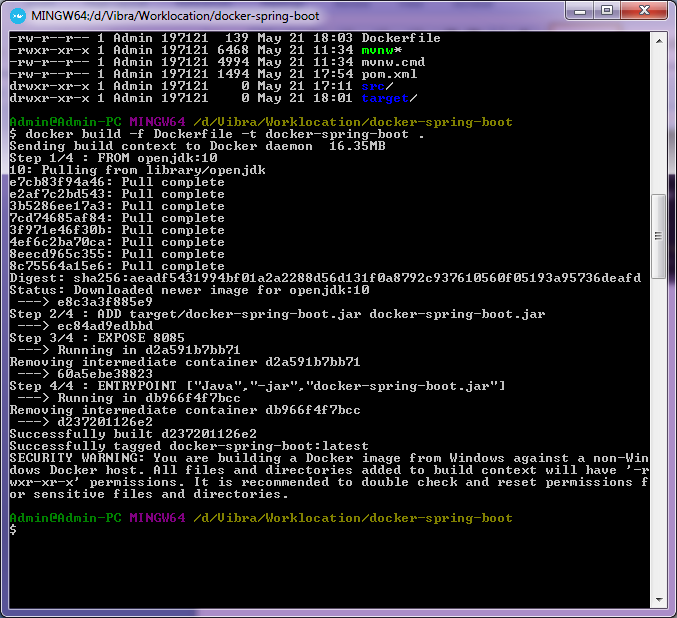
* EXPOSE: this line will expose port 8085to use.
* ENTEYPOINT: This the final command to execute our created jar file inside the container.

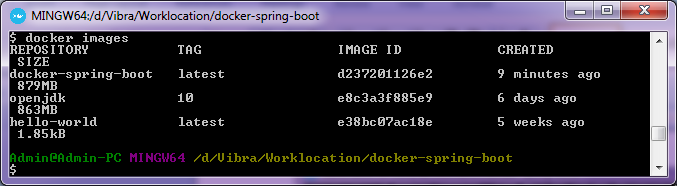
### Build docker image

Navigate to a directory where we developed our spring boot application. and start build the image.  
above screen will list the current directory and currently running images.

Execute build command: docker build -f Dockerfile -t docker-spring-boot .

Docker build command will take -f argument "Dockerfile" for creating docker image. -t will be the tag option for this image. image will be created in this name. "." will be the current location to build the images.

Final screen after building image from docker image.In above screen, we can see each step we mention in Dockerfile executed one by one.

Openjdk and docker-spring-boot image available after build our image from Dockerfile

### Run created image

execute command "docker run -p 8085:8085 docker-spring-boot" this will run created image in port 8085. -p option will publish this application in 8085. first one implies local applications port and 2nd one mapping this to containers 8085 port.

