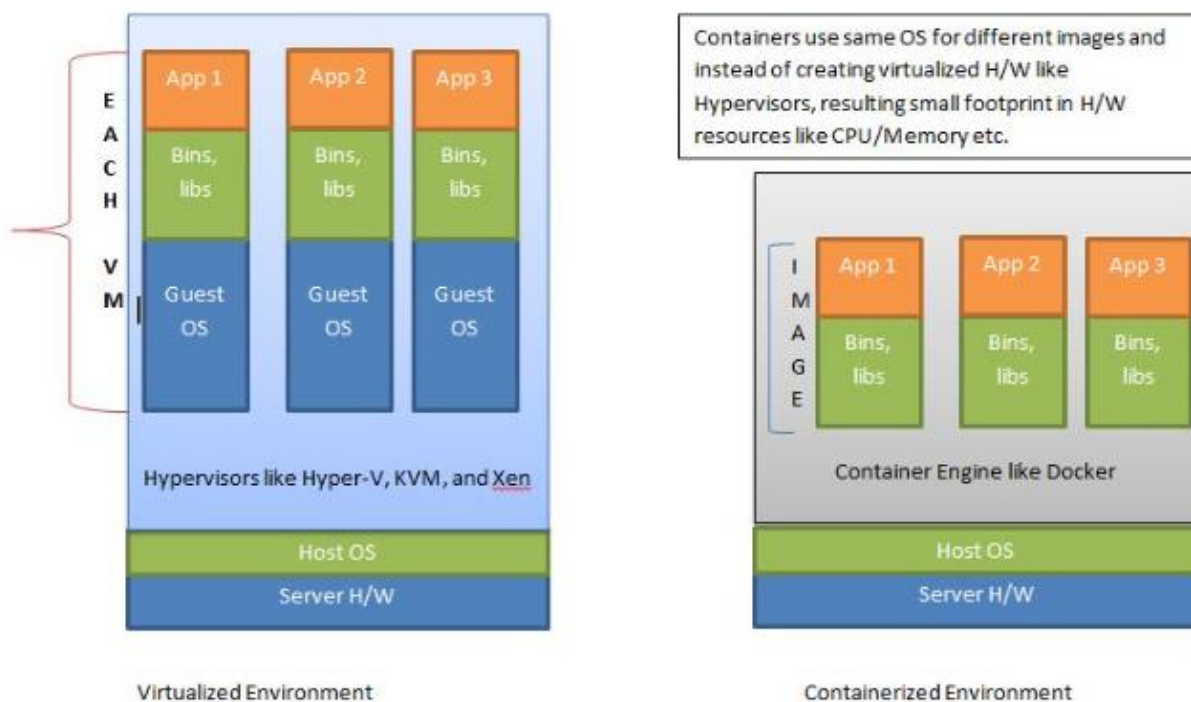


**Docker** is a developer tool to package applications along with their runtime environment, so anybody can deploy and run them on any other machine without facing runtime environment conflicts. It is very similar to the virtual machine concept ([virtualization](#)), where you can get a VM image and run it on any supporting hardware. All internal programs in VM will function as they were packaged originally.

**The difference between a VM and a docker image** is that the docker image does not package the whole virtual operating system. It uses the OS resources just like other processes in the developer's machine, only the application and its runtime-specific dependencies are packaged (**Containerization**).



Docker allows users to publish docker images and consume those published by others in repositories like [Docker Hub](#).

## 1. Installing Docker on Windows

To install docker on Windows machine, follow the below steps:

### 1.1. Choose the Appropriate Docker installer for your System

Before starting with the installation process we need to understand the exact Docker version that is suitable for the Windows that you are using. Docker has provided two versions of Windows distribution as bellow

- For Windows 10 we need to follow this link <https://docs.docker.com/docker-for-windows/>
- For Windows 7, 8 and older versions we need to use Docker Toolbox and here is the official link for that <https://docs.docker.com/toolbox/overview/>

### 1.2. Download Docker Installer

We need to first download the Docker toolbox distribution from [https://download.docker.com/win/static/stable/x86\\_64/](https://download.docker.com/win/static/stable/x86_64/) and we will follow the installation steps in local Workstation.

### 1.3. Enable Hardware Virtualization Technology

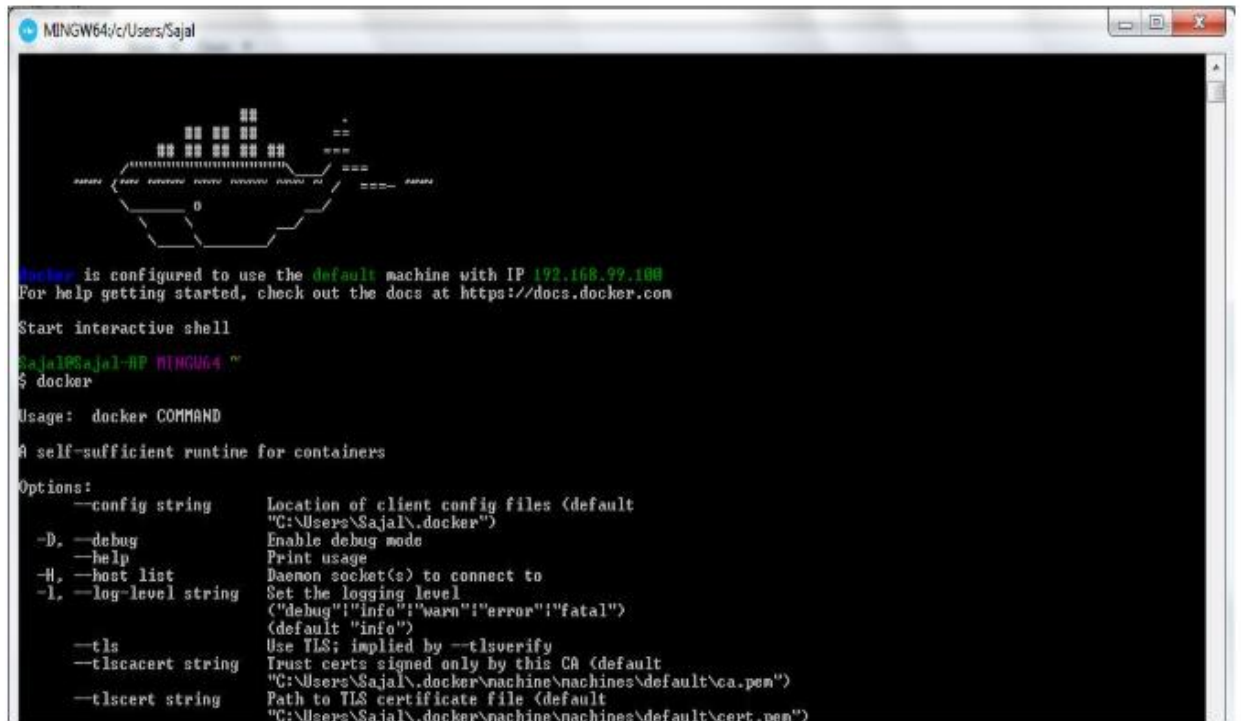
In order to Docker toolbox works properly we need to make sure your Windows system supports Hardware Virtualization Technology and that virtualization is enabled. Docker has provided detailed step on this here: [https://docs.docker.com/toolbox/toolbox\\_install\\_windows/#step-1-check-your-version](https://docs.docker.com/toolbox/toolbox_install_windows/#step-1-check-your-version). If we don't have this enabled then we need to go to BIOS option and enable Hardware Virtualization. The BIOS is bit different for different models of Computer, so please follow official guideline for enabling that.

#### **1.4. Run Docker Installer**

Once we have the Installer downloaded and we have enabled the Hardware Virtualization, we can start the installer. It's just like a simple another windows based installation process guided by a installation wizard.

#### **1.5. Verify the Installation**

To [verify Docker](#) installation, open *Docker Quickstart Terminal* shortcut from either Desktop or Start menu. Verify that Docker prompt is coming and then need to test few basic commands. The docker prompt and sample docker command will look like below.



```
MINGW64/c/Users/Sajal

Docker is configured to use the default machine with IP 192.168.99.100
For help getting started, check out the docs at https://docs.docker.com

Start interactive shell
Sajal@Sajal-HP MINGW64 ~
$ docker

Usage: docker COMMAND

A self-sufficient runtime for containers

Options:
  --config string      Location of client config files (default
                        "C:\Users\Sajal\.docker")
  -D, --debug           Enable debug mode
  --help               Print usage
  -H, --host list       Daemon socket(s) to connect to
  -l, --log-level string Set the logging level
                        ("debug"|"info"|"warn"|"error"|"fatal")
                        (default "info")
  --tls                Use TLS; implied by --tlsverify
  --tlscacert string    Trust certs signed only by this CA (default
                        "C:\Users\Sajal\.docker\machine\machines\default\ca.pem")
  --tlscert string      Path to TLS certificate file (default
                        "C:\Users\Sajal\.docker\machine\machines\default\cert.pem")
```

## 1.6. Note Down the Docker IP

We need to now note down the Docker IP assigned to this Container. We will access this IP to access the Applications installed inside Docker. To know the IP from the command prompt use command `docker-machine ip`. Here is the sample output of the command. Please note that this IP will be different for different M/Cs.

```
Sajal@Sajal-HP MINGW64 ~
$ docker-machine ip
192.168.99.100
```

## 2. Creating a Docker Image

We will first create a spring boot-based REST API, add docker-specific configuration and then we will create docker image.

## 2.1. Create a Spring Boot Application

Develop one simple hello world [Microservice](#) for testing. We have used [spring boot](#) and [Maven](#) and Eclipse as IDE. Add a [REST](#) endpoints so that once this application is deployed in to Docker, we can test this by accessing the rest endpoint.

```
import java.util.Date;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
public class HelloDockerApplication {

    public static void main(String[] args) {
        SpringApplication.run(HelloDockerApplication.class, args);
    }
}

@RestController
class HelloDockerRestController {
    @RequestMapping("/hello/{name}")
    public String helloDocker(@PathVariable(value = "name") String name) {
        String response = "Hello " + name + " Response received on : " + new Date()
        System.out.println(response);
        return response;
    }
}
```

Update resources/application.properties with server port information.

```
server.port = 9080
```

Now test this microservice by running the project as spring boot application.

## 2.2. Add Dockerfile

Now create a file named `Dockerfile` in the root directory and add the below lines as Docker configurations.

```
FROM openjdk:8-jdk-alpine
VOLUME /tmp
ADD target/hello-docker-0.0.1-SNAPSHOT.jar hello-docker-app.jar
ENV JAVA_OPTS=""
ENTRYPOINT [ "sh", "-c", "java $JAVA_OPTS -Djava.security.egd=file:/dev/./urand
```

This is used by Docker while creating the image. It is basically declaring the Java runtime information and target distributions

2.3. Optional. Not taken snapshot therefore

## 2.4. Create Docker Image

Now use maven command `mvn clean install dockerfile:build` to create docker image.

```
mvn clean install dockerfile:build
```

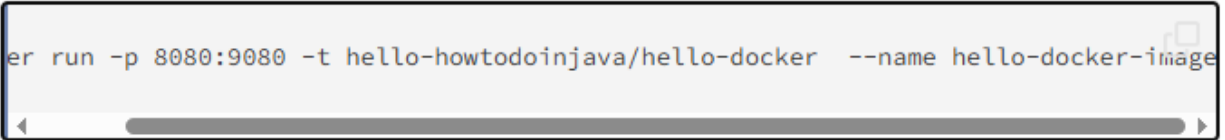
```
[INFO] Step 3/5 : ADD target/hello-docker-0.0.1-SNAPSHOT.jar hello-docker-app.jar
[INFO] ----> ce7491518508
[INFO] Removing intermediate container c74867501651
[INFO] Step 4/5 : ENV JAVA_OPTS ""
[INFO] ----> Running in f7cd27710bf3
[INFO] ----> 086226135205
[INFO] Removing intermediate container f7cd27710bf3
[INFO] Step 5/5 : ENTRYPOINT sh -c java $JAVA_OPTS -Djava.security.egd=file:/dev/
[INFO] ----> Running in 9ef14a442715
[INFO] ----> bf14919a32e2
[INFO] Removing intermediate container 9ef14a442715
[INFO] Successfully built bf14919a32e2
[INFO] Successfully tagged hello-howtodoinjava/hello-docker:latest
[INFO]
[INFO] Detected build of image with id bf14919a32e2
[INFO] Building jar: F:\Study\Technical Writings\docker\hello-docker\target\hel
[INFO] Successfully built hello-howtodoinjava/hello-docker:latest
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
```

### 3. Deploy and Run Docker Image

So we have created the Docker Image (i.e. hello-docker-0.0.1-SNAPSHOT-docker-info.jar). We also have a installed docker container running in our local machine.

Now, to run the docker image inside installed docker container, we will use below command.

```
docker run -p 8080:9080 -t hello-howtodoinjava/hello-docker --name hello-docker
```



```
er run -p 8080:9080 -t hello-howtodoinjava/hello-docker --name hello-docker-image
```

Here the option `-p 8080:9080` is important. It says that expose port 8080 for internal port 9080. Remember our application is running in port 9080 inside docker image and we will access that in port 8080 from outside Docker container.

Now access the application with URL <http://192.168.99.100:8080/hello/sajal>. Notice that the browser output is same as output of standalone REST API on localhost.



← → ↻ ⓘ localhost:9080/hello/sajal

Hello sajal Response received on : Tue Aug 15 21:54:14 IST 2017