**DOCKER**

Docker is a developer tool to package applications along with their runtime environment, so anybody can deploy and run them in any other machine without facing runtime environment conflicts.

Docker is a container management service. The keywords of Docker are Develop, ship and run.

Docker is an open-source containerization platform by which you can pack your application and all its dependencies into a standardized unit called a container.

Containers are light in weight which makes them portable and they are isolated from the underlying infrastructure and from each other container. We can run the docker image as a docker container in any machine where docker is installed without depending on the operating system.

Objective of Docker is to develop the applications easily, ship them into containers which can then be deployed anywhere.

We can deploy containers anywhere (Physical machine, VM Cloud)

Docker containers are light weight and scalable.

In Docker, everything is based on images. Images is combination of a file system and parameters.

**Docker Components**

Dockerfile: Dockerfile uses DSL (Domain specific language) and contains instructions for generating docker image. Dockerfile ill define the process to quickly produce an image.

Docker Image: It is a executable package of software that includes everything that is needed to run the application. The image informs how a container should be instantiated, determine which software components will run and how.

Docker Container: Is a run time instance of an image. Container is a virtual environment that bundles application code and all its dependencies that are required to run the application.

Docker Hub: Is a cloud based repository service where we can push to docker images and pull them as needed.

Docker Compose: Is a tool that we can use to run the application on multiple containers using single source. Ex: If we use MySQL and Java to build a project, we can create single image that starts both the containers as a service. To achieve this, we need to define docker-compose.yml

Docker Engine: The software that hosts the container is called Docker Engine. It is a client-server based application. Docker engine has 3 main components:

1. Server: It is responsible for creating and managing docker images, containers, networks and volumes on the docker. It is referred to as a daemon process.
2. REST API: It specifies how the applications can interact with the server and instructs it what to do.
3. Client: Client is a Docker command line interface (CLI) that allows users to interact with docker using docker commands.

**Docker Architecture**

Container 2

Container 1

App2 lib

App1 lib

Docker Engine

**Docker Workflow**

Virtual Machine

DockerHub

Docker Image

Dockerfile

DEV

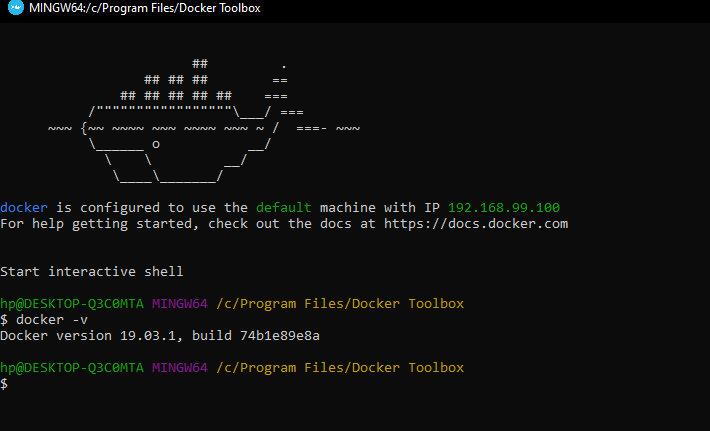
PROD

UAT

Docker Container

Docker Installation: <https://www.tutorialspoint.com/docker/docker_installation.htm>

To check if docker is installed or not, Open “Docker Quickstart Terminal” and run docker -v which will display docker version.



**Build a Spring Boot App and Deploy it in Docker as image**

1. Build a basic spring Boot Application.
2. Create a REST controller and expose an API
3. Create “Dockerfile” in project root directory with FROM, EXPOSE, ADD and ENTRYPOINT
4. Build the Spring project.
5. Open Docker terminal
6. Go to root directory of the project to create docker image (from Dockerfile), then run the image)
7. docker build -t <<Jarfilename.jar>> .
8. docker run -p 9090:8080 <<Jarfilename.jar>>
9. Open browser and call the API from docker image (http://192.168.99.100:9090/hello)

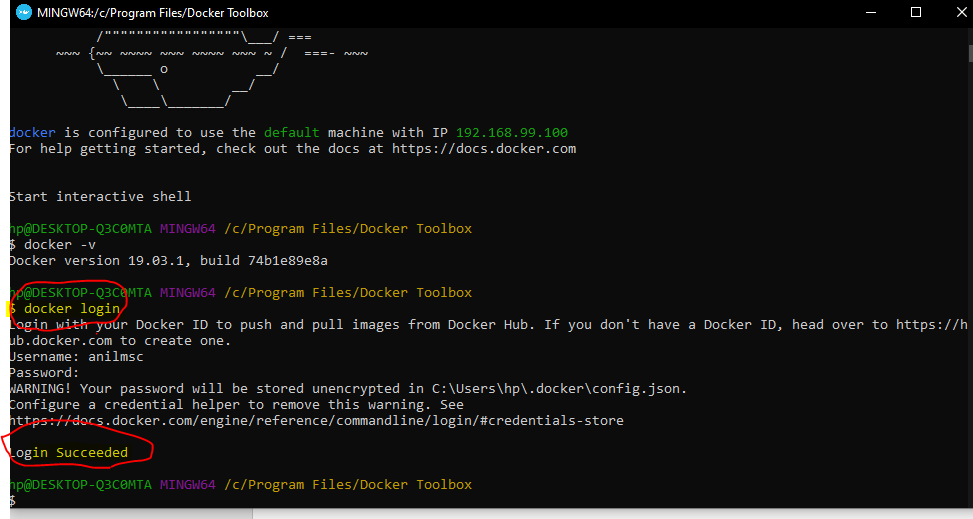


Dockerfile:

**Push & Pull**

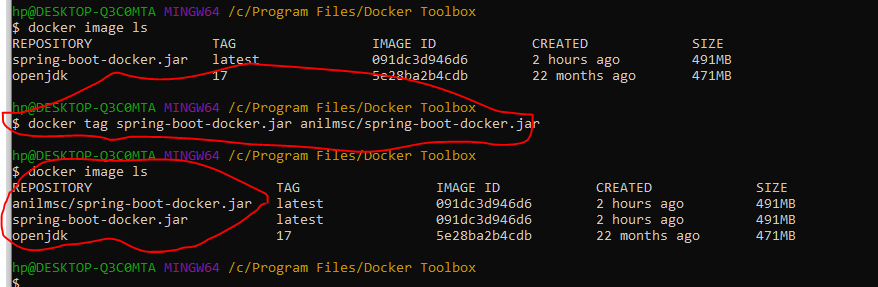
To push and pull images to and from Docker hub repository, we should have account in Docker Hub.

1. Sign up (<https://hub.docker.com/>)
2. Sign in
3. Open Docker terminal and login with credentials.

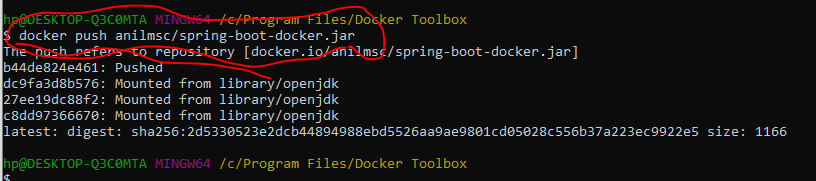


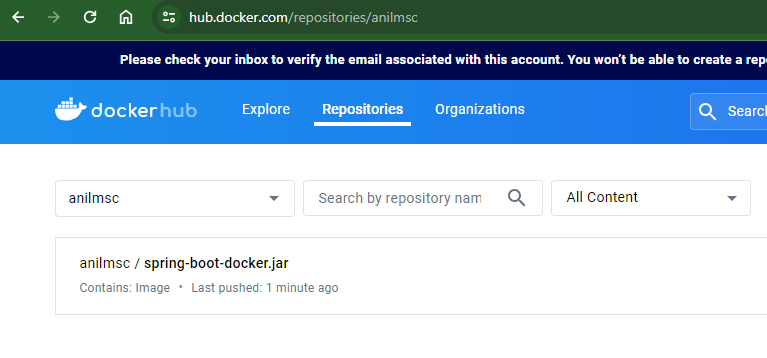
1. Run “docker image ls” to see list of images.
2. As we have our image in public repository, we need to tag our images to identify them.

docker tag <<jar name.jar>> userid/jarname.jar



1. push the image (docker push anilmsc/jarname.jar (It uploads the image to docker hub)
2. Login to docker hub and check the repo





1. To pull the image from Docker Repo, run (docker pull anilmsc/spring-boot-docker.jar) and the docker image will be available in local.
2. To run the pulled image (docker -p run 9090:8080 anilmsc/spring-boot-docker.jar) and then hit the REST endpoint in browser (<http://192.168.99.100:9090/hello>)

**Dockerize Spring App using Maven plugin**

To build and push the application to docker hub, add the below plugin in pom.xml and build the project. After build is successful, image will be uploaded to the docker hub with tag.

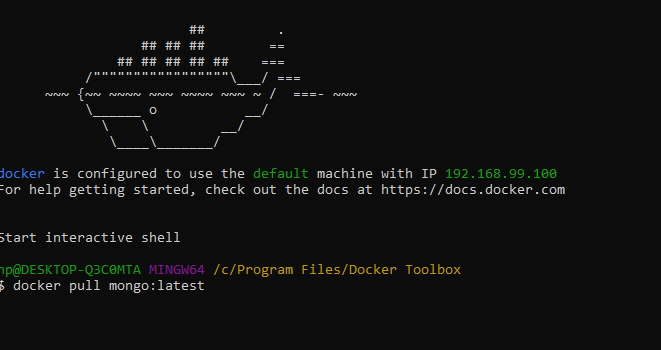
|  |
| --- |
| <plugin>  <groupId>com.spotify</groupId>  <artifactId>dockerfile-maven-plugin</artifactId>  <version>1.4.0</version>  <configuration>  <repository>anilmsc/${project.artifactId}</repository>  <tag>${project.version}</tag>  <buildArgs>  <JAR\_FILE>target/${project.build.finalName}.jar</JAR\_FILE>  </buildArgs>  </configuration>  <executions>  <execution>  <id>default</id>  <phase>install</phase>  <goals>  <goal>build</goal>  <goal>push</goal>  </goals>  </execution>  </executions>  </plugin> |



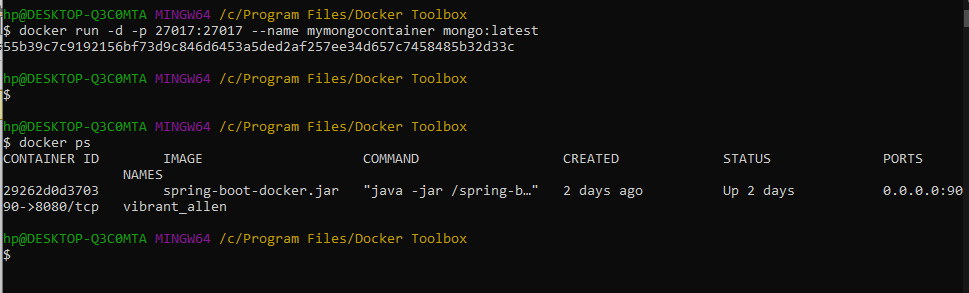
**Spring App and Mongo DB as Containers using Docker-Compose**

In this example, we will run spring boot application in one container and mongo db in another container. To integrate these two, we will use Docker-compose utility.

Instead of downloading mongo and setup it, we will download mongo image from docker.



* docker pull mongo:latest 🡪 will download mongo image
* docker images 🡪 To see list of images
* docker run -d -p 27017:27017 –name <<nameofthecontainer>> imagename:latest 🡪 To run mongo image in container



* docker ps 🡪 this will show the images running.

1. Create a Spring boot application for CRUD operations
2. Build the image (Go to the project directory and run below command)

**docker build -t spring-boot-mongodb:1.0 .**

