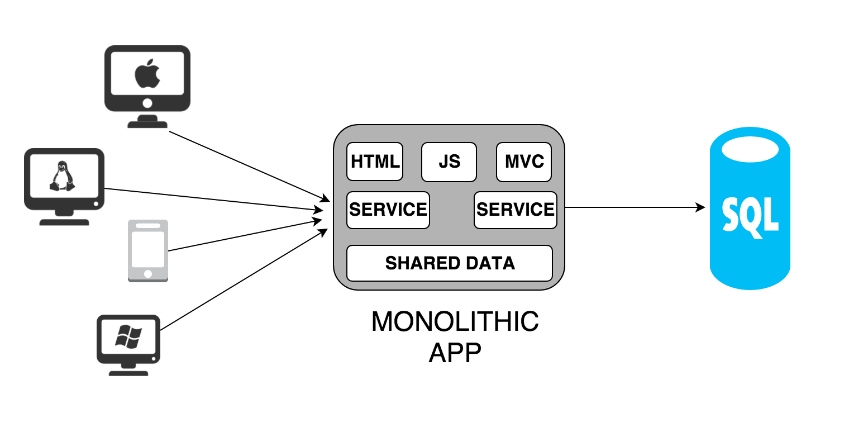
**Microservices with springboot**

Micro-services is the way of developing a complex and large applications by dividing it into smaller modules. Micro-services are loosely coupled, deployed independently and suitably for small teams. With Micro-services developers, we can decide how to design, language to choose, deploy, scaling etc.

A monolithic architecture is traditional combine model for develop and design a software application. Monolithic, means collected all in one piece. Monolithic software is designed as self-contained components of program are interrelated and independent rather than loosely coupled as is the case with modular software programs. In a tightly-coupled architecture, each component and its associated components must be present in order for code to be executed or compiled.

Drawbacks of Monolithic Architecture

* This approach is simple and has a size and complexity limitation.
* Application is large and complex.
* Application size is large so it can slow down start time.
* On each update must redeploy the entire application.
* Continuous deployment is difficult.
* Difficult to scale
* Another problem is reliability. Bug in any one of the module can possibly get down the complete process.



**Microservices Architecture**

Microservices

Service

Service

Service

Service

API Gateway

Identity Provider

Client

Remote Service

Ser

Services Discovery

Management

Static Content

CDN

**About Micro-services**

* Micro-services can use variable programming languages for development.
* We can easily be found out the errors in any module or micro-services consequently saves time.
* Easy to manage smaller modules or micro-services.
* When update required, it can be directly pushed on to the particular module or micro-services, or else, the entire application requires to be updated.
* Based on the client need, we can manage scale up and scale down for the particular module or micro-service without disturbing the other module or micro-services.
* It also increase the productivity.
* If any one of the micro-services or module goes down, the application will remains unaffected.

**Docker and Container**

Docker is a containerization platform which packages your application and all its dependencies together in the form of containers so as to ensure that your application works seamlessly in any environment be it development or test or production.

Docker containers, wrap a piece of software in a complete file system that contains everything needed to run: code, runtime, system tools, system libraries etc. anything that can be installed on a server. This guarantees that the software will always run the same, regardless of its environment.

What is Docker Image?

Docker image is the source of Docker container. In other words, Docker images are used to create containers. Images are created with the build command, and they’ll produce a container when started with run. Images are stored in a Docker registry such as registry.hub.docker.com because they can become quite large, images are designed to be composed of layers of other images, allowing a minimal amount of data to be sent when transferring images over the network.

What is Docker file used for?

Docker can build images automatically by reading the instructions from a Docker file.  
Now I will suggest you to give a small definition of Docker file.

A Docker file is a text document that contains all the commands a user could call on the command line to assemble an image. Using docker build users can create an automated build that executes several command-line instructions in succession.

**How to create Docker container?**

We can use Docker image to create Docker container by using the below command:

|  |  |
| --- | --- |
|  | docker run -t -i command name |

This command will create and start a container.

You should also add, If you want to check the list of all running container with the status on a host use the below command:

|  |  |
| --- | --- |
|  | docker ps –a |

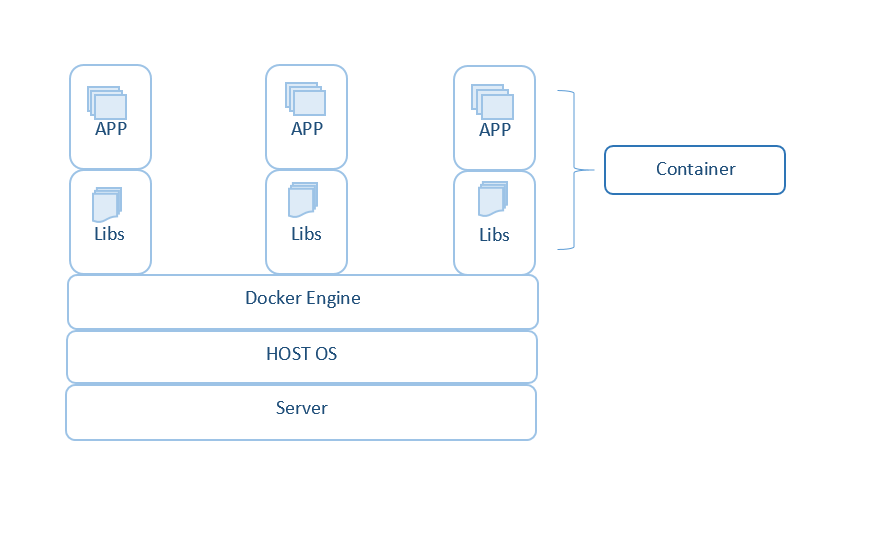
**How to stop and restart the Docker container?**

In order to stop the Docker container you can use the below command:

|  |  |
| --- | --- |
|  | docker stop container ID |

Now to restart the Docker container you can use:

|  |  |
| --- | --- |
|  | docker restart container ID |

Creating a virtualization environment to run several applications or operating system without interrupting each other is called container. With the help of container, we can deploy our application quickly, reliably and consistently. Because containers have its own CPU, memory, network assets. Containers are lightweight because they don’t requires the extra load of a hypervisor, they are capable of direct run within the host machine.

Earlier we were facing the problem that code easily run on developer machine or environment but while implementing it on the production environment, dependency issue occurs to overcome this virtual machines are came, but they are also heavyweight that still includes to wastage of resources like RAM, the processor, etc. If we have more than 50 micro-services to run then, VMs are not the best choice.

Docker is light weighted Container that consists inbuilt images and it takes very less storage comparatively. For running a Docker we require a Linux or Ubuntu as a base machine.

**Terms in Docker are**

**Docker Hub**— it’s cloud hosted service provided by Docker. Here we can pull the images in public repository and we can upload our own image.

**Docker Registry —**Storage or space component for Docker images also we can store in public repository or in private repository. We can use this to integrate image storage with the development workflow and also we can control where images are stored.

**Docker images**— Template read only that is used to create the container stored on Docker hub or local registry built by Docker user.

**Docker Containers**— it’s the runtime instance of Docker image, built from one or more images. Hence Docker helps to achieve application issues, Isolation, and faster development.

**Containers Advantages**

* Consumption of resources like RAM, Processor, Disc space are maintained as present there is no requirement to pre locate these resources and are chanced according to the application requirements.
* Sharing a container is easy.
* Docker delivers a platform to manage the containers lifecycle.
* Containers offer consistent environment computation.
* Containers can run separate applications within a single shared operating system.

**8. Container Orchestration**

Automated, Arrangement, Coordination, and Management of containers and the resources they consume during deployment of a multi-container packed application is called Container Orchestration.

**Various features of Container Orchestration includes**

* **Cluster Management**— Management of all containers is done by Orchestration and Developer’s task is limited to launch a cluster of container instances and specify the tasks which are needed to run.
* **Task Definitions**— many tasks can be launched through single task definition. It allows the developer to define task where they have to specify the number of containers required for the task and their dependencies.
* **Programmatic Control**— to launch and stop Docker containers simple API calls one can register and deregister tasks.
* **Scheduling**— Container scheduling deals with placing the containers from the cluster according to the resources they need and the availability of requirements.
* **Load Balancing**—distributing traffic across the containers/deployment.
* **Monitoring**— one can monitor CPU and memory utilization of running tasks and also gets alerted if scaling is needed by containers.

**Tools used for Container Orchestration**

For Container orchestration different tools are used, some of are open source tools like Kubernetes, and Docker Swarm which can be used privately, also some paid tools are there like Google Containers, AWS ECS from Amazon, and Microsoft Containers.

* **Amazon ECS —**Amazon ECS is one of the important product from Amazon Web Services that provides us runtime environment for Docker Containers and also provides orchestration. It gives running Dockerized applications on top of the Amazon Infra.
* **Azure Containers Service —**Azure Container Service product is by [Microsoft](https://azure.microsoft.com/en-in/) allows almost same functionalities. It consists very good support form .NET ecosystem.
* **Docker Swarm —**it is open source tool, with this tool, we can run multiple Docker engines as a single virtual Docker. These Dockers own containers orchestration Tool. It consists of the worker and manager nodes that run different services for orchestration. Worker node run containers that assigned by managers and Managers that distributes tasks across the cluster.
* **Google Container Engine**— Google Container Engine permit us to run Docker containers on the [Google Cloud Platform](https://cloud.google.com/). It schedules the containers into cluster and manages them as per requirements given. It’s going to build on the top of Kubernetes.
* **Kubernetes**— Kubernetes is most mature orchestration systems for the Docker containers. It’s also an open source system using for automating the deployment and management of containerised

Example:

* 1. Clone from the GitHub: <https://github.com/anand-prodevans/Springboot.git>
  2. Open terminal and go to the project folder path and check the docker and maven installation with

$docker –version



$mvn –v

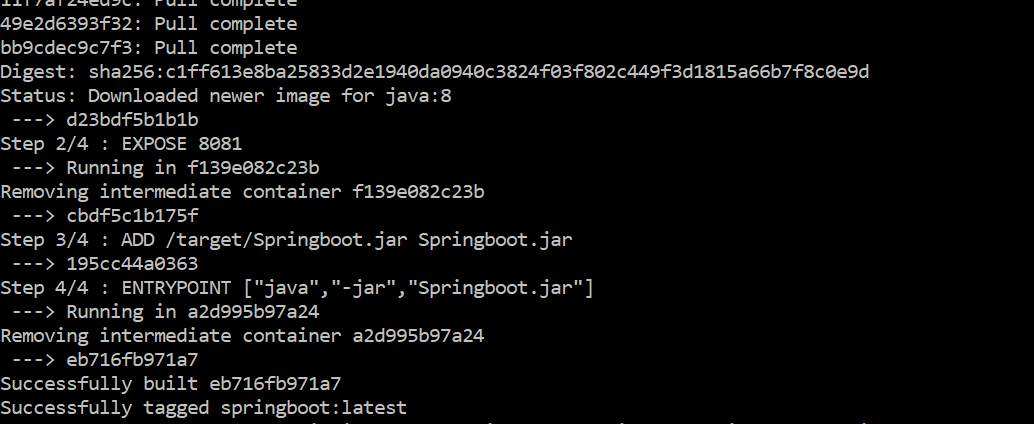
* 1. Clean and build the project

$mvn clean install



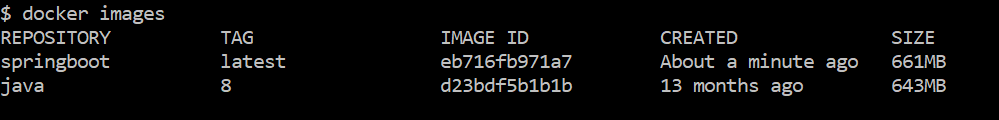
* 1. Build the Dockerfile

$docker build -f Dockerfile -t springboot .



And check the docker image

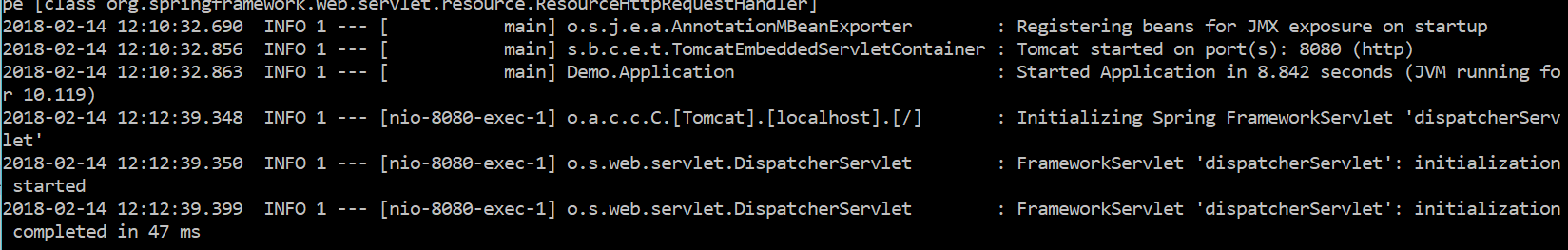
$docker images



* 1. Build the Docker file

$ docker run -p 8080:8080 springboot





* 1. Open browser and enter the url

http://<ipaddress>:8080

