Microservices with Docker

# Docker

Docker is an open platform for developing, shipping, and running applications. It enables you to separate your applications from your infrastructure so you can deliver software quickly.

In Docker, we create images of environments with application programs, so that one code can run everywhere without any installations. Portable images are called a *Dockerfile*. *Dockerfile* will define what goes on in the environment inside your container.

## Terminologies

* Dockerfile – This file is used to create images. We define out application configurations here.
* Images - The file system and configuration of our application which are used to create containers.
* Containers - Running instances of Docker images, it runs the actual applications.
* Docker daemon – It runs in background on host and executes commands given by client.
* Docker client – It helps user interact with the daemon.
* Docker Store – Registry of Docker images, you can pull or push images from here.

## Guidelines and Example to deploy our application on Docker

In this document, it is assumed that your services are successfully build and running without errors.

### Install Docker

Download and install Docker on your machine from this link: <https://docs.docker.com/install/>

After installation check version using following command.



It should return Docker version installed on your machine.

### Try Hello-World program

Test your installation by running a sample Hello-World Docker image.



As you can see here, it first checks locally and fails to find image. It will then pull the image from Docker repository and run it.

### Some important commands

Before we move further, here is list of some commands that we will be using in this document.

* List all images



* List all containers



* Run a container



* Build image from Dockerfile



Here *“.”* represents current directory.

* Stop a container



* Delete a container



* Delete locally stored image



### Create Dockerfile

First step in deploying our application in Docker is creating *Dockerfile*. It contains base image, configurations, and execution commands.



*Dockerfile* always starts with *FROM* and then base image.

Here we are fetching *openjdk* from Docker repository which is java image and required to run our spring application. We are setting current working directory as root for container. *ADD* command copies and rename our application jar file. Since we are running our spring application on *port no 9091*, we need to expose it outside using *EXPOSE* command. At last we run command for executing application.

### Create Image from Dockerfile

After creating *Dockerfile*, we create image from that to start out container.



We are setting our image name as *eureka-server* and *“.”* represents current directory.

### Run the container

Once we are done with building image for application, next we run the image in a container.



We have used some arguments here, below is detailed explanation on same.

* ***-d***

It is used to run in detach mode. Using this argument will get you back to terminal and container will keep running in background.

* ***-p {host-port}:{docker-port}***

This is used to bind *docker-port* with host-port, so that we can access our application using *host-port*. Here our application is running on port *7072* in container and we are binding it to port *7072* of host, so we can access our application using this link http://{host}:7072.

* ***--net=host***

We use this argument to bind container directly to our host network interface. We have used this here to get services registered on eureka with host’s IP address so that other applications/containers can access it.