[**1. Introduction to Docker**](https://www.vogella.com/tutorials/Docker/article.html#introduction-to-docker)

[**1.1. What is Docker**](https://www.vogella.com/tutorials/Docker/article.html#what-is-docker)

Docker is a light weight container, allowing to run pre-configured system images on another operating system. Is is developed as an open source project, released under the Apache License, version 2.

You can package a applications, libraries or operating systems in a container, for example: \* OS, \* JVM, \* App server \* Application with its configuration

This makes the virtual machine portable across environments. Developers can use the same setup as you can use in your production environment.

Docker requires a very small amount of resources, as does not require an hypervisor for each container nor does it simulate a complete computer.

Docker addresses the following use cases:

* You can to use a specific version of a operating system for testing
* You want to run your software tests in an controlled environment
* You want to configure a system and share the configuration

[**1.2. Docker components**](https://www.vogella.com/tutorials/Docker/article.html#docker-components)

Docker consists of the following components:

* Images
* Containers
* Daemon
* Clients
* Registries

[**1.3. Dockerfile**](https://www.vogella.com/tutorials/Docker/article.html#dockerfile)

A dockerfile allows to define images.

[**1.4. Images**](https://www.vogella.com/tutorials/Docker/article.html#images)

Images are read-only templates which provide functionality for running an instance of this image. For example, the latest release of Ubuntu might be provided as image. Images are defined as layers, for example, you can add Java to the Ubuntu image and get another image based on this.

The [Docker hub](https://hub.docker.com/) provides pre-configured images. You can modify existing images and save these modifications as new image.

[**1.5. Containers**](https://www.vogella.com/tutorials/Docker/article.html#containers)

Container are the started components based on images. They contain the actual application and dependencies but share the same kernel. They can be started, stopped, paused, deleted. Containers are immutable and disposable.

[**1.6. Docker Daemon**](https://www.vogella.com/tutorials/Docker/article.html#docker-daemon)

Is used to manage the container. It runs natively on Linux and inside a VM on Windows and Mac OS X. To start it use the docker command.

[**1.7. Docker clients**](https://www.vogella.com/tutorials/Docker/article.html#docker-clients)

Clients (CLI, IDE) run on host VM. They provide the tools to interact with container, i.e., to start them.

[**1.8. Docker registry**](https://www.vogella.com/tutorials/Docker/article.html#docker-registry)

Images are saved in a registry and have an ID with consists of a repository and a tag. For example, *fedora:22*, is an image which contains the Fedora 22 OS from the fedora repository.

To use an image you have to pull it from a registry, to share an image with others you have to push it to one. The default Docker registry is the Docker Hub. You can upload your personal images to Github, in this case you add your user name as prefix to the image, e.g., *vogella/fedore:22*

[**1.9. Docker Compose**](https://www.vogella.com/tutorials/Docker/article.html#docker-compose)

Compose is a tool for defining and running applications in multiple Docker container. You use a YAML file to configure your applications services.

[**1.10. Managing data in Docker**](https://www.vogella.com/tutorials/Docker/article.html#managing-data-in-docker)

By default, Docker stores all data inside the container, which makes is harder to persist this data and to reuse the data in another container.

Docker offers two storage options for hosting data on the host machine:

* volumes - stored on the host file system managed by Docker, preferred way of persisting data in Docker containers
* bind mounts - old way of managing file includes, prefer using volumes

Docker volumes which are not explicitly created, are created the first time they are mounted to a container. If the container is stopped the volume still persists.

[**2. Docker installation and setup**](https://www.vogella.com/tutorials/Docker/article.html#docker-installation-and-setup)

Install Docker Toolbox from <https://www.docker.com/>. The installation is well described on the getting started page. For example <https://docs.docker.com/linux/step_one/> describes the installation for Linux

Afterwards you can test you installation as described on the webpage:

sudo docker run hello-world

If you issue the command the first time, it downloads the hello-world image and start it.

To allow your user to run docker commands without the sudo prefix, configure a new group and add you user to it.

sudo usermod -aG docker ubuntu

Afterwards you need to logout and login again. Validate that you can run docker commands without sudo.

docker run hello-world

This is only relevant if you want to install docker locally on your PC and play around with it. Otherwise, we have dedicated docker servers on-prem.

[**3. Docker commands**](https://www.vogella.com/tutorials/Docker/article.html#docker-commands)

[**3.1. List all available images**](https://www.vogella.com/tutorials/Docker/article.html#list-all-available-images)

docker images

[**3.2. List all available containers**](https://www.vogella.com/tutorials/Docker/article.html#list-all-available-containers)

Via the -a flag you list all contains. Without -a you only list the running containers.

docker ps -a

[**3.3. Create a new container**](https://www.vogella.com/tutorials/Docker/article.html#create-a-new-container)

docker run image

[**3.4. Delete a container and an image**](https://www.vogella.com/tutorials/Docker/article.html#delete-a-container-and-an-image)

Use the following command to remove all your containers.

docker rm $(docker ps -a -q)

Use the following command to remove all your images.

docker rmi $(docker images -q)

[**3.5. Start a terminal session on a running container**](https://www.vogella.com/tutorials/Docker/article.html#start-a-terminal-session-on-a-running-container)

Use this command to start a terminal session on a running container:

docker exec -it <container id> /bin/sh

|  |  |
| --- | --- |
|  | If your image is using bash you have to replace /bin/sh with /bin/bash. |

[**3.6. Start a terminal session on an image**](https://www.vogella.com/tutorials/Docker/article.html#start-a-terminal-session-on-an-image)

Use this command to start a terminal session on an image:

docker run -i -t --entrypoint /bin/sh <image id>

[**3.7. Inspect a Docker container**](https://www.vogella.com/tutorials/Docker/article.html#inspect-a-docker-container)

docker inspect <image id>

[**4. Building and Deploying an Image**](Docker%20Guide.docx)

cd /[path of your code]

sudo docker build -t [image name]:[tag] .

sudo docker login [openshift path] -p [login token]

sudo docker push [image name]:[tag]

For push to work you need the image name to start with [openshift path]/[namespace]