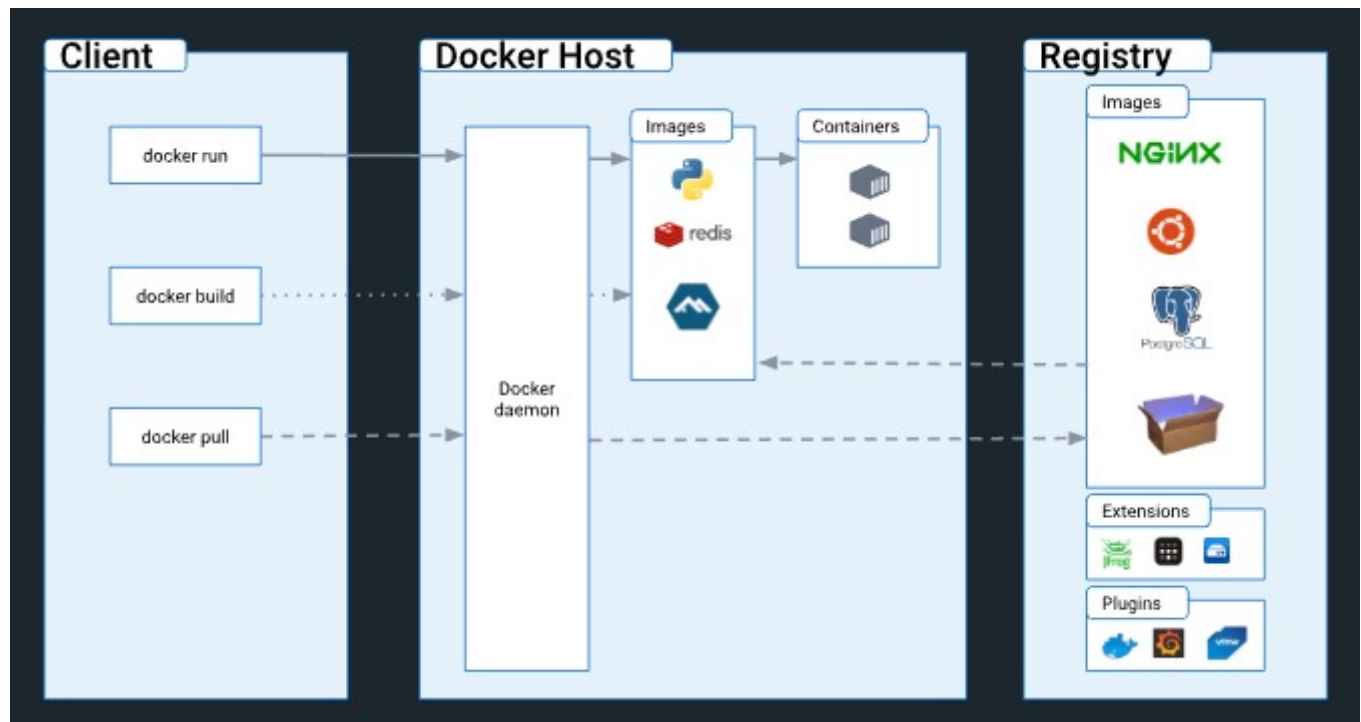




LEARNING DOCKER



[Explore the docs »](#)

[Main Page](#) - [Code Page](#) - [Report Bug](#) - [Request Feature](#)

Summary

► [TABLE OF CONTENT](#)

About Project

This project aims to help students or professionals to learn the main concepts of Docker

[\(back to top\)](#)

Getting Started

To get a local copy up and running follow these simple example steps.



Prerequisites

List things you need to use the software and how to install them.

```
git
Virtual Box and extension
Vagrant
```

Installation

Clone the repo

```
git clone https://github.com/marcossilvestrini/learning-docker.git
```

Usage

Use this repository for get learning about Docker exam

[\(back to top\)](#)

Roadmap

- ☒ Create repository
- ☒ Create github action for automation tasks
- ☒ Create examples about docker containers
- ☒ Create examples about docker images

[\(back to roadmap\)](#)

[\(back to top\)](#)

Docker Engine work with namespaces(PID,NET,IPC,MNT,UTS) and cgroups.

```
# Get a version of docker
docker --version
```

Docker Containers

```
# list containers
docker container ls
```

Light



```
docker ps
# list containers id
docker container ls -aq
docker ps -aq

# list containers virtual size
docker container ls -s

# create containers
docker container create -it ubuntu

# run container
docker run hello-world

# run container iterative
docker run -it <image_name> bash

# execute command in container
docker exec -it <container_id_or_name> <command>

# connect to docker container
docker container attach <CONTAINER ID>

# run container with name
docker run -it --name ubuntu01 ubuntu bash

# create container with specified network
docker run -it --name ubuntu01 --network skynet ubuntu bash

# create container with network host
docker run -it --name ubuntu01 --network host ubuntu bash

# start containers
docker container start ubuntu

# stop pause containers
docker stop <container_id_or_name>
docker stop -t=0 <container_id_or_name>

# Stop all containers
docker stop $(docker container ls -q)

# Pause\Unpause containers
docker pause <container_id_or_name>
docker unpause <container_id_or_name>

# delete container
docker rm <container_id_or_name> --force

# delete all containers
docker container rm $(docker container ls -aq) --force

# forwarding port
docker run -d -P <container_id_or_name>
```

□ **Light** `docker run -d -p 8080:80 <container_id_or_name>`



`# show map ports`

`docker port <container_id_or_name>`

`# inspect container`

`docker inspect <container_id_or_name>`

`# show container resources usage information`

`docker container stat`

`docker container stats <container_id_or_name>`

`# show process in execution in container`

`docker container top <container_id_or_name>`

`# show container logs`

`docker container logs <container_id_or_name>`

`docker container logs -f <container_id_or_name>`

`# set limit of memory for container`

`docker container run -it -m 512M --name testmemory debian`

`docker container run -it --name testmemory2 --memory 1G debian`

`# set limit of cpu for container`

`docker container run -it --cpus=0.5 --name testcpu nginx`

`# update ram|cpu resource in container`

`docker container update -m 2048 testmemory`

`docker container update --cpus=3 testcpu`

`# get infos memory and cpu`

`docker inspect <container_id_or_name> | grep -i cpu`

`docker inspect <container_id_or_name> | grep -i mem`

[\(back to docker containers\)](#)

[\(back to top\)](#)

Docker Images

`# pull image`

`docker pull <image_name>`

`# show local images images`

`docker images`

`# show details of images`

`docker inspect <image_id>`

`# show details of images layers`

`docker history <image_id>`

Light

```
# remove docker images
docker rmi <image_id> --force

# remove all docker images
docker rmi $(docker images -aq) --force

# commit changes in container
## install and customize your container after...then:
docker commit -m "my container" CONTAINERID
docker tag IMAGEID marcosilvestrini/apache_2:1.0
```



Docker Build

Build a docker image

```
# first, create your Dockerfile with your app

# Example Dockerfile
FROM debian
RUN /bin/echo "HELLO DOCKER"

# then create a docker image.
cd <path_of_your_dockerfile>
docker build -t <dockerhub_username/image_name:tag>

# publish your image in docker hub
docker push <dockerhub_username/image_name:tag>
```

[\(back to docker images\)](#)

[\(back to top\)](#)

Docker Volumes

```
# list docker volumes
docker volume ls

# inspect docker volumes
docker volume inspect <volume_name>

# find docker volumes
docker volume inspect --format '{{ .Mountpoint }}' <volume_name>

# create docker volume
docker volume create <volume_name>
```

Light



```

# delete docker volume
docker volume rm <volume_name>

# delete all docker volume if not in user]
docker volume prune

# create container with docker bind mounts
docker run -d --mount type=bind,source=/myfolder-volume,target=/app <image_name_or_id>

# mount volumes
docker run -it -d -v <path_local_for_data>:<path_container_for_data> <image_name_or_id>
docker run -d -v <volume_name>:/app <image_name_or_id>

# mount file
docker container run -ti --mount type=bind,src=<path_local_for_data/file>,dst=<path_container_for_data/file> <image_name_or_id>

# share container volumes

## create container volume
docker container create -v /data --name dbdata centos

## create containers
docker run -d -p 5432:5432 --name pgsq11 --volumes-from dbdata \
-e POSTGRES_USER=docker -e POSTGRES_PASS=docker \
-e POSTGRES_DB=docker kamui/postgresql

docker run -d -p 5433:5432 --name pgsq12 --volumes-from dbdata \
-e POSTGRES_USER=docker -e POSTGRES_PASS=docker \
-e POSTGRES_DB=docker kamui/postgresql

## find volume
docker inspect dbdata | grep -i Source

## list containers data of shared volume
sudo ls /var/lib/docker/volumes/<volume_id>/_data

# create backups of shared volume
docker run -ti --volumes-from dbdata -v $(pwd):/backup debian tar -cvf /backup/backups.tar.gz /backup

```

[\(back to docker volumes\)](#)

[\(back to top\)](#)

Docker Network

```

# list networks
docker network list

# inspect docker network
docker network inspect <network_name>

# create docker network bridge

```

Light `docker network create --driver bridge <network_name>`



```
# delete docker network
docker network rm <network_name>
```

[\(back to docker network\)](#)

[\(back to top\)](#)

Docker Compose

```
# list containers|services
docker-compose ps
docker-compose -f configs/docker/apps/app-silvestrini/docker-compose.yml ps

# create containers|services
docker-compose up
docker-compose up -d
docker-compose -f configs/docker/apps/app-silvestrini/docker-compose.yml up
```

[\(back to docker composed\)](#)

[\(back to top\)](#)

Docker Stack

```
# create stack|services
docker stack deploy -c docker-compose.yml first

# list stacks|services
docker stack ls

# view stack services
docker stack services first

# view detailed stack|services | all containers
docker stack ps first

# remove stack|services
docker stack rm <stack_service_name>
```

[\(back to docker stack\)](#)

[\(back to top\)](#)

Docker Swarm

Light



```
# init docker swarm with manager
docker swarm init --advertise-addr 192.168.0.140:2377

# get swarm token
MANAGER_TOKEN=$(docker swarm join-token manager -q)
WORKER_TOKEN=$(docker swarm join-token worker -q)

# join server in docker swarm
docker swarm join --token $MANAGER_TOKEN 192.168.0.140:2377

# promote node worker to manager
docker node promote debian-server02

# list swarm nodes
docker nodes ls

# inspect nodes
docker node inspect <node_name>

# leave node
docker swarm leave --force

# delete node
docker node rm debian-server01

# list services
docker service ps
docker service ps webserver

# create service
docker service create --name webserver --replicas 5 -p 8080:80 nginx
docker service create --name webserver --replicas 5 -p 8080:80 --mount type=volume,s

# inspect service
docker service inspect webserver

# scale service
docker service scale webserver=10
```

[\(back to docker secrets\)](#)

[\(back to top\)](#)

Docker Secrets

```
# create secret
echo 'mypassword' | docker secret create my_password -
docker secret create my_secret2 my_secret.txt

# list secrets
docker secret ls
```


□ Light

```
# inspect secrets
docker secret inspect my_secret2

# delete secrets
docker secret rm my_secret2

# create service with secrets
docker service create --name web --detach=false --secret my_password nginx
```



(back to docker secrets)

(back to top)

Contributing

Contributions are what make the open source community such an amazing place to learn, inspire, and create. Any contributions you make are **greatly appreciated**.

If you have a suggestion that would make this better, please fork the repo and create a pull request. You can also simply open an issue with the tag "enhancement". Don't forget to give the project a star! Thanks again!

1. Fork the Project
2. Create your Feature Branch (`git checkout -b feature/AmazingFeature`)
3. Commit your Changes (`git commit -m 'Add some AmazingFeature'`)
4. Push to the Branch (`git push origin feature/AmazingFeature`)
5. Open a Pull Request

License

- This project is licensed under the MIT License * see the LICENSE.md file for details

Contact

Marcos Silvestrini - marcos.silvestrini@gmail.com



Project Link: <https://github.com/marcoossilvestrini/learning-docker>

(back to top)

Acknowledgments

☐ Light

- Docker Website
- Docker Overview
- Convert Command in Dockerfile
- Deploy Docker Register
- Record Terminal



(back to top)