Cloud Computing (**LAB**)

HANDS ON  docker

Giuseppe Bisicchia
giuseppe.bisicchia@phd.unipi.it

Dipartimento di Informatica, Università di Pisa

Cosa faremo oggi?

- Esplorare le possibilità offerte da *Docker* utilizzando la sua CLI e containerizzando una Web App.
- **TO-DO** seguire la checklist disponibile su moodle:
    1. Rispondere ad alcune domande sui principali comandi di *Docker.*
    2. Scrivere un `Dockerfile` per eseguire una Web App.
    3. Provare l'applicazione sia in locale che tramite Docker.
- **Suggerimento:** sfruttate questo laboratorio per **sperimentare** il più possibile! Prendete la **checklist** come **linea guida** ma sentitevi liberi di provare ulteriori comandi o approfondire quelli già utilizzati.

Dockerfile cheat sheet

command	description
FROM *image*	base image for the build
COPY *path dst*	copy *path* from the context into the container at location *dst*
ADD *src dst*	same as COPY but accepts archives and urls as *src*
RUN *args…*	run an arbitrary command inside the container
CMD *args…*	set the default command
USER *name*	set the default username
WORKDIR *path*	set the default working directory
ENV *name value*	set an environment variable
EXPOSE *port(s)*	allow the container to listens on the network *port(s)*
ENTRYPOINT *exec args…*	configure a container that will run as an executable

Il Dockerfile è un file contenente (alcuni) di questi comandi che vengono eseguiti ordinatamente. Fate attenzione: il nome 'Dockerfile' è con la D maiuscola e senza estensione.

Docker cheat sheet

IMAGES

Docker images are a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

Build an Image from a Dockerfile
```
docker build -t <image_name> .
```

Build an Image from a Dockerfile without the cache
```
docker build -t <image_name> . –no-cache
```

List local images
```
docker images
```

Delete an Image
```
docker rmi <image_name>
```

Remove all unused images
```
docker image prune
```

CONTAINERS

A container is a runtime instance of a docker image. A container will always run the same, regardless of the infrastructure. Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging.

Create and run a container from an image, with a custom name:
```
docker run --name <container_name> <image_name>
```

Run a container with and publish a container's port(s) to the host.
```
docker run -p <host_port>:<container_port> <image_name>
```

Run a container in the background
```
docker run -d <image_name>
```

Start or stop an existing container:
```
docker start|stop <container_name> (or <container-id>)
```

Remove a stopped container:
```
docker rm <container_name>
```

Open a shell inside a running container:
```
docker exec -it <container_name> sh
```

Fetch and follow the logs of a container:
```
docker logs -f <container_name>
```

To inspect a running container:
```
docker inspect <container_name> (or <container_id>)
```

To list currently running containers:
```
docker ps
```

List all docker containers (running and stopped):
```
docker ps --all
```

View resource usage stats
```
docker container stats
```

from https://docs.docker.com/get-started/docker_cheatsheet.pdf

Documentazione

- https://docs.docker.com/get-started/
- https://docs.docker.com/engine/reference/builder/
- https://docs.docker.com/develop/develop-images/dockerfile_best-practices/
- https://docs.docker.com/engine/reference/commandline/docker/