Introduction to containers and Docker

Summer School on Effective HPC for Climate and Weather
Alberto Madonna, CSCS
August 27, 2020
Table of Contents

1. Introduction to containers and Docker (30 min)
2. Tutorial / live demo (60 min)
   - Break -
3. Next part: containers on HPC with Sarus

- Slides and code available at [https://github.com/eth-cscs/containers-hands-on](https://github.com/eth-cscs/containers-hands-on)
- Lab tutorial video: [https://youtu.be/dv74sFb3cVc](https://youtu.be/dv74sFb3cVc)

Disclaimer: This material reflects only the author’s view and the EU-Commission is not responsible for any use that may be made of the information it contains
Containers

- Isolated environments to run applications/services
- Images include all software dependencies
- Prescriptive, portable, easy to build, quick to deploy
Linux containers ecosystem

- Linux containers rely on abstraction features (*namespaces*\(^1\)) provided by the kernel
- Different design decisions and use cases gave rise to several solutions:

\(^1\) “Namespaces in operation, part 1: namespaces overview” at [https://lwn.net/Articles/531114/](https://lwn.net/Articles/531114/)
Docker

- Extremely popular container implementation

- Easy to use authoring tools
  - Container images are created from recipe-like files
  - Images can be named, tagged and built on top of other images

- Cloud-based image distribution strategy
  - Several remote registries available (e.g. Docker Hub)
  - Client includes facilities to authenticate, push and pull images
Docker workflow

1. An image is created locally from a Dockerfile
2. Push (i.e. upload) the image to a remote registry
   DockerHub is the public registry maintained by the Docker company
3. Pull (i.e. download) the image on a target machine and run the container

2. Push to Docker Hub

1. Create Docker image

3. Pull image and run container
**Key terms**

- **Image**: standalone, executable package that includes everything needed to run a piece of software (code, runtime libraries, configuration files). Provides the filesystem and metadata (e.g. environment variables, initial working directory) for a container.

- **Container**: a process isolated from the rest of the system through abstractions created by the kernel. The level of isolation can be controlled, allowing access to host resources. Its filesystem content comes from an image.
  - Can be thought as the runtime *instance* of an image: what the image becomes in memory when actually executed.
So... how are containers useful?

- Containers give the possibility to create (scientific) applications that are:

  1. Portable
  2. Prescriptive
  3. Easy to deploy
  4. Easy to test
Live demo!
Using NVIDIA GPUs in Docker

**GPU-accelerated application**

- Included/built in the image, along with its runtime dependencies

- NVIDIA provides base images for CUDA, featuring compilers, runtime and accelerated libraries: [https://hub.docker.com/r/nvidia/cuda](https://hub.docker.com/r/nvidia/cuda)

- Quickest way to get a Dockerfile going: 
  ```
  FROM nvidia/cuda
  ```

**GPU driver**

- It is tied to the hardware: cannot be part of a portable image!

- Has to be imported upon container creation

- NVIDIA Container Toolkit to the rescue! 
  [https://github.com/NVIDIA/nvidia-docker](https://github.com/NVIDIA/nvidia-docker)

- Docker >= 19.03 has native support: 
  ```
  docker run --gpus all nvidia/cuda nvidia-smi
  ```
Docker cheatsheet

```bash
docker pull <user/image:tag>
docker run <image:tag> <command>
docker run -it <image:tag> bash
docker run <image:tag> mpiexec -n 2
docker images
docker build -t <user/image:tag> .
docker login
docker push <user/image:tag>
```
Further material & contact

- Slides and Lab material: [https://github.com/eth-cscs/containers-hands-on](https://github.com/eth-cscs/containers-hands-on)

- Lab intro video: [https://youtu.be/dv74sFb3cVc](https://youtu.be/dv74sFb3cVc)

- Official Docker documentation: [https://docs.docker.com/](https://docs.docker.com/)

- Best practices for writing Dockerfiles: [https://docs.docker.com/develop/develop-images/dockerfile_best-practices/](https://docs.docker.com/develop/develop-images/dockerfile_best-practices/)

- Contact: alberto.madonna@cscs.ch
The ESiWACE1/2 projects have received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 675191 and No 823988

Disclaimer: This material reflects only the author’s view and the EU-Commission is not responsible for any use that may be made of the information it contains
Thank you for your attention.