Podman-compose

Isac Pasianotto

1. **Podman-compose... Why?**
2. Main concepts
3. Practical session

Step by step towards the real world...

- At this point we’ve seen how to distribute software using containers.
- The examples shown so far were simple, but things don't always go that way:
    - Single process application
    - One-purpose image
    - The software does not need to interact with others applications

When this is a limitation?

- When you need to handle several tasks concurrently.
  - E.g., Reading I/O, while performing computation, while storing results in a database.
- The application design requires it. E.g., ui+server+storage+rev.proxy

Proposal:

1. Let's make a huge Dockerfile and manage everything as we already know how.

2. Several Dockerfiles, and several contaiers managed as we already know.

Proposal:

1. Let's make a huge Dockerfile and manage everything as we already know how

2. Several Dockerfiles, and several contaiers managed as we already know.

NOT APPROVED

But why?

Why?

- Technically speaking, if you are skilled enough it will work anyway.
- The fact that you can do something, does not necessarily mean you should do it!
- Never heard about the KISS?

Why? (cont’d)

- Managing them individually is cumbersome and ***error-prone.***
- In the case of manually manage several container, you have to take care of all of them, runs in the correct order and ensure connectivity among those.
- Since it is a very common problem, a better, more canonical solution exists!

1. Podman-compose... Why?
2. **Main concepts**
3. Practical session

Podman compose... What is that?

- A (python-based) tool that interprets a special kind of files (we will see)
- Has the same interface of the `podman` command.
- It supports:
  - Building and running multi-container apps
  - Networking between containers
  - Environment variable injection
  - Persistent volumes
  - ...
- Compatible with its counterpart Docker compose

Compose.yaml file

- A.K.A. `docker-compose.yaml` or `[docker-]compose.yml`
- Written in yaml language.
  - It is human-readable!
  - It has a declarative approach
- Can be concatenated with other files and been overridden (eg. base & dev/prod)

Compose.yaml file cheat-sheet

version	Refers to the Compose version (usually do not edit this please!)
services	Defines the services that need to run.
app	A custom name for one of your containers.
image	The image to pull.
container_name	The name for each container.
restart	Starts or restarts a service container.
port	Defines the custom port to run the container.
working_dir	The current working directory for the service container.
environment	Defines environment variables (e.g., DB credentials).
command	The command to run the service.

1. Podman-compose... Why?
2. Main concepts
3. **Practical session**