Host-container sync

Often we need to synchronize the local file system with the container so that the source data can be changed without rebuilding the image. For example, we want to be able to modify data files, change python scripts, or preview the results of container runs on the local system.

The above scenarios can be implemented using the **bind mount** mechanism.

Synchronizing the host folder with the container

To create the image, in the folder with the Dockerfile, run the command in the terminal:

`docker build -t bind_app1_app2:v3 .`

After building the image, run the corresponding container by connecting the local volume:

`docker run -ti -v"$(pwd):/app" bind_app1_app2:v3 python3 run.py`.

Note that:

1. this time the files generated within the container synchronize with your local folder. In particular, new files have appeared in the `app_1/input` and `app_1/output` folders
2. however, it is necessary to run the container in a folder that contains the necessary files. The `-v"$(pwd):/app"` option removes files from the `app` folder of the container.
3. if you want to run the container in a place where the necessary files do not exist, run it without the bind mount option: `docker run -ti bind_app1_app2:v3 python3 run.py`.

Modify the file in the local folder

Now let's check if this synchronization works both ways. To do this, modify the `run.py` script by adding to it the print `first old...`.

As you can see, the modification of the local file is taken into account by the container without rebuilding the image.

Usage scenarios

In particular, you can use the mechanism mentioned above in the following situations:

Setup

I want to continue working on my project on another computer.

I frequently change location: traveling, returning home, etc. I would like to be able to perform calculations, train models or test the application on the computers I have access to. They differ not only in operating systems, but also in the environments installed on them and the hardware configuration. In particular, some of them cannot use gas pedals such as graphics cards.

Instead of installing dedicated environments on each of them, I run the docker container with different parameters (e.g. with or without GPU acceleration).

Development

I want to demonstrate an application using machine learning to other team members.

...but so that they can experiment with the relevant components themselves if necessary. (e.g., change the data, training algorithms, final application, etc.).

I provide them with a docker image with the appropriate environment and a set of source codes.

Sharing

I want to conduct a training on Data Science.

I want to show someone a solution in jupyter notebook, using various libraries (e.g. PyCaret). The users don't have any environments installed.

I ask them to install Docker and then run my container.

Deployment

I want to pass host-specific data to the container (e.g. as runtime parameters).

I use docker bind mount for this purpose.

I want to simulate a production environment.

Our image analysis system is running on an NVidia Jetson Nano endpoint device. On my workstation, I want to simulate how the latest version of my model will behave on this device.

To do this, I run a Docker container with an image of the end device.

Useful sources

A very good presentation of the *bind mount* and *volume mount* methods can be found in the official docker documentation available here.