Creating a container that only provides a runtime environment

The simplest scenario for using a Docker container is to provide only the runtime environment in it. The set of files necessary to run the application should be on the local (host) system.

Building an image

Create an image:

`docker build -t bind_app1_app2:v1 .`

Run the image without connecting the local volume:

`docker run -ti bind_app1_app2:v1`.

Note that:

1. when you start the container, it automatically starts the python environment
2. pandas package is installed in this environment: the command `import pandas as pd` does not generate an error.

Copying files from host to container using the CLI

To be able to use such a configured environment in practice, you need to transfer files to it. To do this, you can use the `docker cp` command.

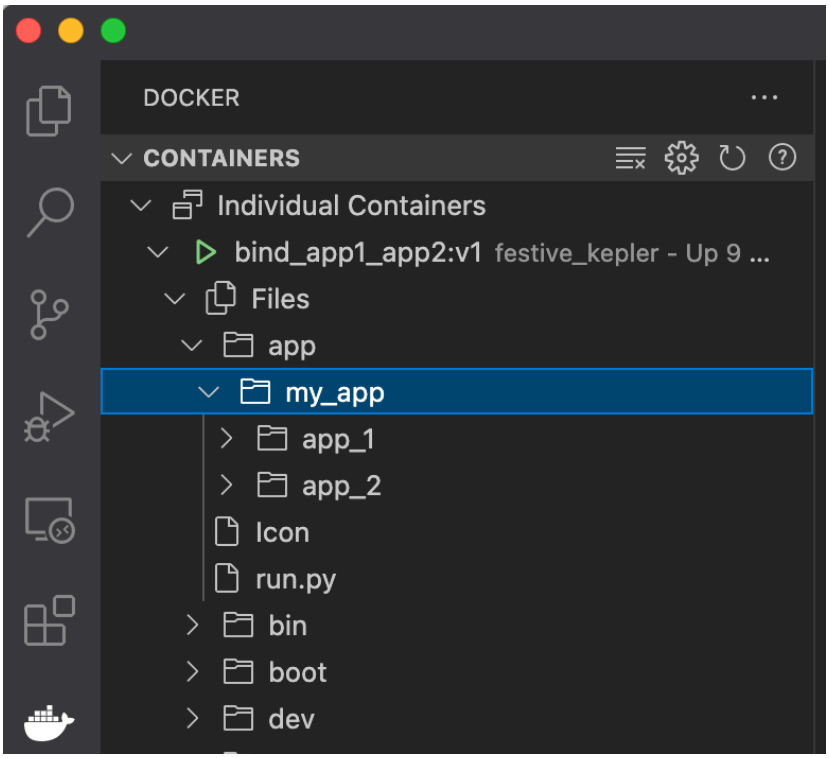
`docker cp ...` allows you to copy files between the running container and the local system. It accepts only one argument on the `source` side, so it is a good practice to gather all files and folders in one directory (in our case *my_app*), and then copy it all to the container. The second parameter is the destination: the ID of the container and its destination folder.

The complete procedure using the command line is as follows:

1. In a new terminal, run the container: `docker run -ti bind_app1_app2:v1 bash`.
2. check its ID: `docker ps`.
3. navigate to the local folder where the folder you want to copy is located. In my case: the folder where *my_app/* folder is located.
4. run another terminal in this folder and type the command in it: `docker cp my_app 111ac631710c:/app`. NOTE: replace *111ac631710c* with the ID of your own container 5 In the terminal with the container running:
    1. check with the `ls` command that your folder has copied correctly.
    2. enter it and run the script: `python3 run.py`.

Copying host-container files using the Docker plugin in an IDE (such as MS Visual Studio Code).

In practice, it is often more convenient to use the IDE environment for file operations on active containers. In particular, MS Visual Studio Code provides inspection and management capabilities for images, containers and volumes (Docker plug-in).



Use cases

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

Organization

I want to quickly introduce a new employee to the project.

A new employee is joining our team. We want to quickly prepare a machine for him (PC, server, notebook).

Instead of installing a set of packages: we run a Docker container.

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

I often change location: travel, go 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 quickly test the performance of a new machine learning library.

However, I don't want to create a dedicated virtual environment, install the appropriate packages, etc.

I acquire an official application image (containing a properly configured environment) and run the corresponding component.

I want to create a new machine learning service by combining ready-made components.

I use https://hub.docker.com/ to acquire the relevant images (a bit like an application "store"). I link them together in a pipeline using docker compose.

Deployment

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 am running a Docker container with an image of the end device.

I want to run my ML project in a scalable environment, such as Kubernetes, or at a service provider (GCP, MS Azure or Amazon EC2).

I create a separate image for each component of my machine learning process. I use the Kubeflow system to manage the full process in Kubernetes.