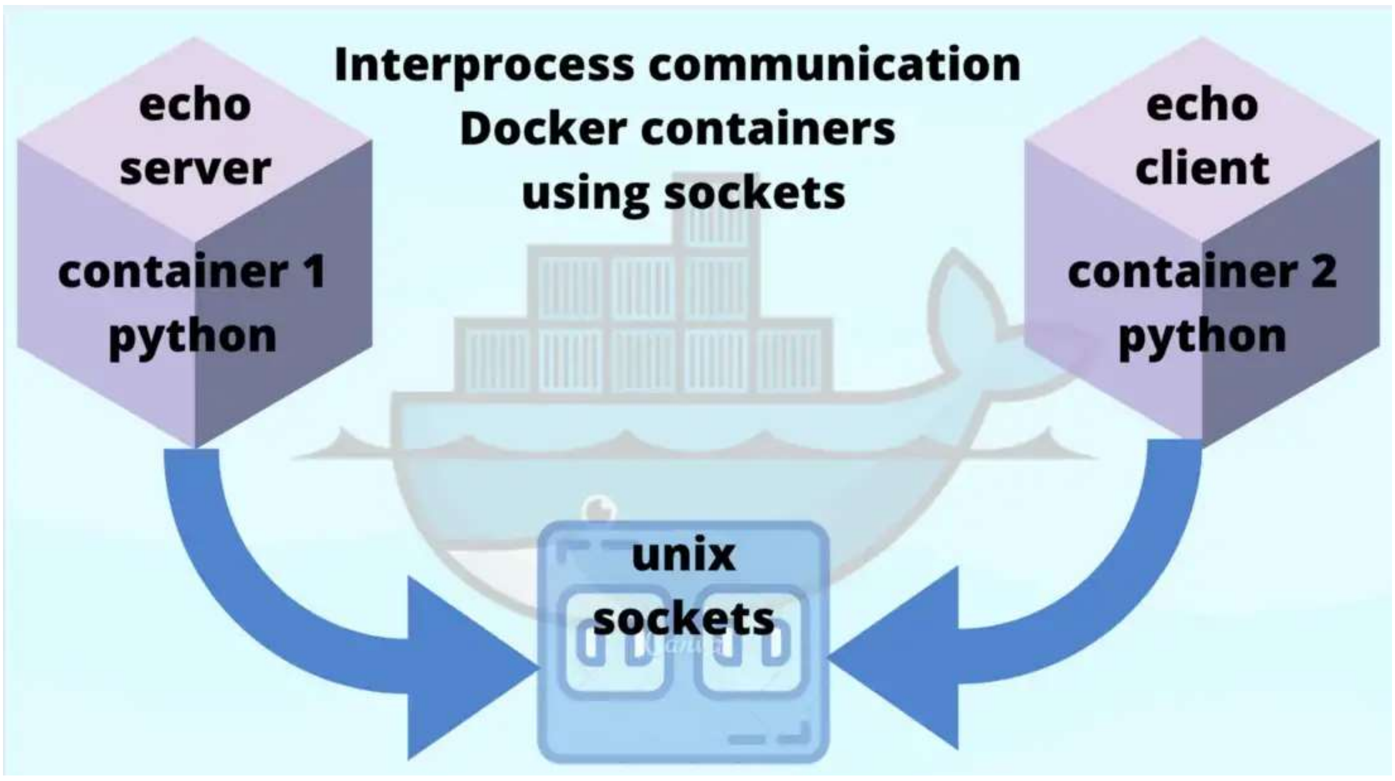
Published in Techanic

Aniket Pingley, Ph. D Follow
May 26, 2020 · 2 min read · Listen

Save



Docker Containers: IPC using Sockets — Part 1

Implementing an echo server and client using Python and deploying them in Docker containers.

In this post I demonstrate socket-based interprocess communication between docker containers. To this end, I will create two containers running python code for socket-based IPC. The server echoes the message back to the client. It must be noted that docker has built-in mechanism for IPC (link), which does not have a specific option for sockets. I will also use this blog to demo some basics of networking with Docker.

If you are conversant with Docker, you can skip to the second part by clicking HERE.

First and foremost, since containers use network namespaces loopback address 127.0.0.1 will not work inside containers for interprocess communication. This excellent blog explains the basics to this end.

Let's build-up step-by-step to the whole setup. First, we will create simple python scripts for server and client for sockets-based IPC.

```
#!/usr/bin/env python3
```

```
2   # ipc_server.py
```

```
8
9   with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
10      s.bind((HOST, PORT))
11      s.listen()
12      conn, addr = s.accept()
13
14      with conn:
15          print('Connected by', addr)
16          while True:
17              data = conn.recv(1024)
18              if not data:
19                  break
20              conn.sendall(data)
```

ipc_server.py hosted with ❤ by GitHub view raw

```
#!/usr/bin/env python3
# ipc_client.py

import socket

HOST = '127.0.0.1'  # The server's hostname or IP address
PORT = 9898        # The port used by the server

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    s.sendall(b'Hello, world. IPC success!')
    data = s.recv(1024)

print('Received', repr(data))
```

ipc_client.py hosted with ❤ by GitHub view raw

Test the scripts out using following commands in separate terminals. Run the server script first.

```
>> python3 ipc_server.py
>> python3 ipc_client.py
```

Let’s deploy the ipc_server.py inside a container. Remember, loopback addresses inside a container are not accessible outside of it due to isolation created by network namespaces. Thus, change line #6 in ipc_server.py to following:

```
HOST = '0.0.0.0'
# 0.0.0.0 is a non-routable meta-address that also specifies all IP #
addresses on all interfaces
```

I assume that you have installed Docker at this point. I will not cover basics of Docker in this blog. Create a file called 'Dockerfile' in the same directory as ipc_server.py.

```
# Get the latest base image for python
FROM python:latest
# Put files at the image '/server/' folder.
ADD ipc-server.py /server/
# '/server/' is base directory
WORKDIR /server/
# Expose port 9898 in the container
EXPOSE 9898
# execute the command
CMD [ "python3", "/server/ipc-server.py" ]
```

Dockerfile hosted with ❤ by GitHub view raw

To build a Docker image run the following commands, where 'my_ipc_server' is the name of image:

```
>> docker build -t my_ipc_server .
>> docker images
```

To create a container from 'my_ipc_server' image, run the following command:

```
>> docker run -rm -p 9898:9898 my_ipc_server
```

The flag '-p' will map internal port 9898 for the server-container to port 9898 that is visible outside. Ports internal to container can be mapped to any port outside, if it is not already in use. Again, the isolation created by network namespaces is responsible to this end. Without this mapping, ipc_client.py will not be able to connect to port 9898. The '-rm' flag will remove the container automatically after it exists.

Now run ipc_client.py from another terminal. You should see the message 'Hello, world. IPC success!' printed on the terminal of client. The message is simply echoed by the server.

In this blog, I have demonstrated IPC using sockets in Python between a container that is running the server code and a client script running on the host. In the second part of the blog, I deploy the client Python script inside another container and demonstrate IPC between containers. HERE is the link to Part 2.

Python Docker Sockets Containers