Cloud Computing - Container

Environment

Operating System	Ubuntu 16.04
docker version	1.12.6
runc version	1.0.0-rc2-dev

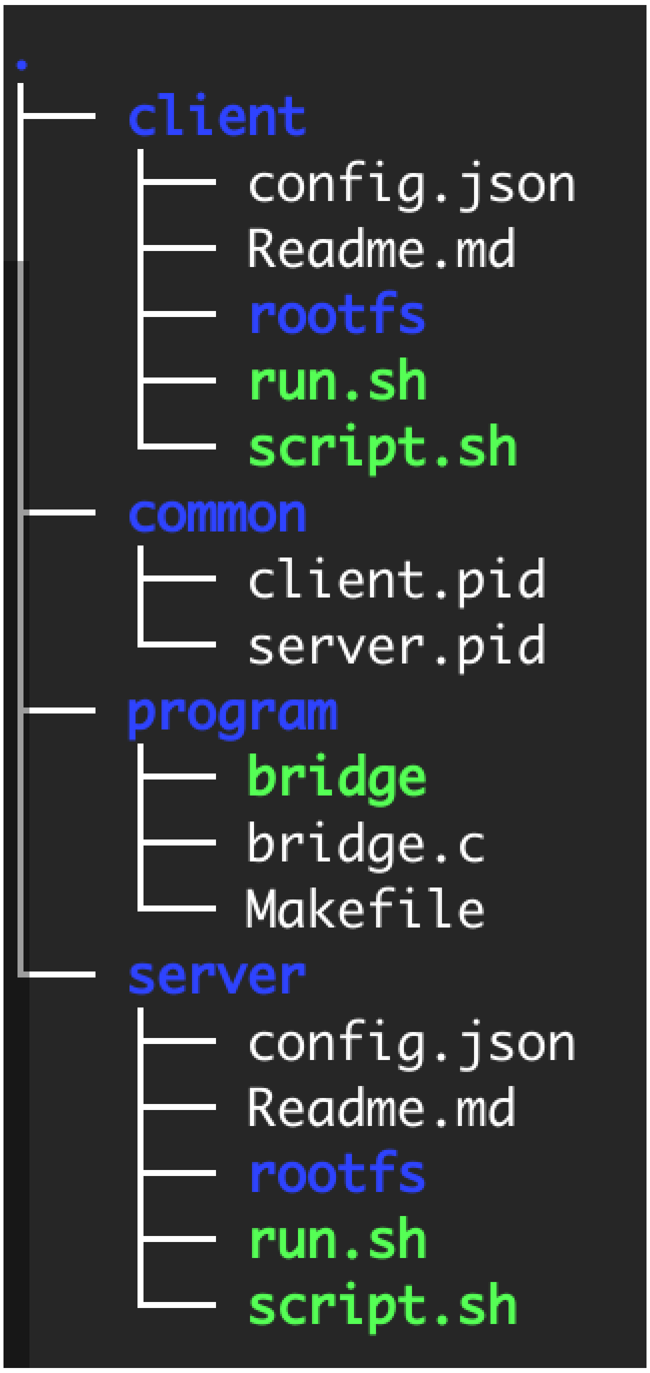
Structure

Figure 1 shows the structure of this project, there are four directory

- server
- client
- common
- program

server and **client** directory contain the `config.json`, `rootfs` which are the necessary info for `runc`, `run.sh` is the script to run the specific container, using `sh run.sh` to run.

program directory contain a `bridge.c` which can set namespace same as container and the `Makefile`, **common** directory store the PID of server and client container.



```
.
├── client
│   ├── config.json
│   ├── Readme.md
│   ├── rootfs
│   ├── run.sh
│   └── script.sh
├── common
│   ├── client.pid
│   └── server.pid
├── program
│   ├── bridge
│   ├── bridge.c
│   └── Makefile
└── server
    ├── config.json
    ├── Readme.md
    ├── rootfs
    ├── run.sh
    └── script.sh
```

Figure 1.

Bridge

In this project, we want to build an echo server with two container, which will

1. Start container server and run the server program, which will block for an imcoming message.
2. Start container client and run the client program, which will block for user inputs.
3. Run the bridge program on host, which will be the bridge of two container.
4. The bridge program will set its `IPC` same as client container, and set its `MNT` same as server container.

5. After user type something, client will communcate with bridge through IPC protocal, and bridge will write file `message` to communcate with server.

Implementaion

When we start the container, we used `--pid-file` to record the pid and store on **common** directory, then the **bridge** program will get the server, client PID from `pid-file`, then use `setns()` to set the namespace to achieve the functionality of bridge.

```
int server_pid;
int client_pid;

GetContainerPid(&server_pid, &client_pid);

std::string client_path = "/proc/" + std::to_string(client_pid) + "/ns/ipc";
std::string server_path = "/proc/" + std::to_string(server_pid) + "/ns/mnt";

setns(open(client_path.c_str(), O_RDONLY), CLONE_NEWIPC))
setns(open(server_path.c_str(), O_RDONLY), CLONE_NEWNS))
```

Usage

Open a session for server

```
 $ cd server
 $ ./run.sh
/# ./server
```

then, open another session for client

```
 $ cd client
 $ ./run.sh
/# ./client <ipc_num>
```

finally, open a session for bridge

```
$ cd bridge
$ make
$ ./bridge <ipc_num> # need privilege
```