第 14 週 –
Socket & Docker

大 綱
Outlines

01 Socket Implementation (python)

02 Docker container

01

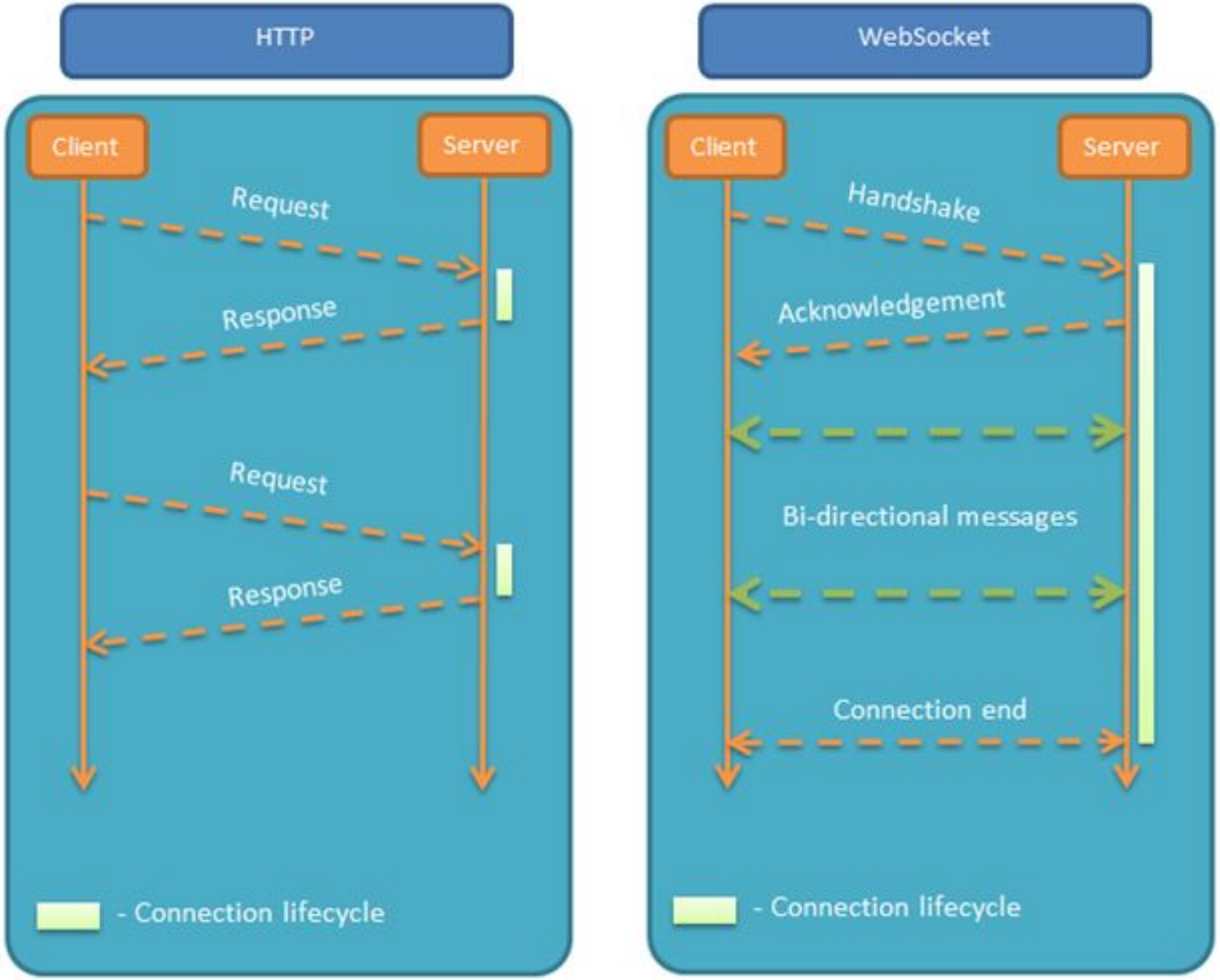
Socket

Socket

- a mechanism allowing communication between processes over the network
- Unix socket v.s. Network socket

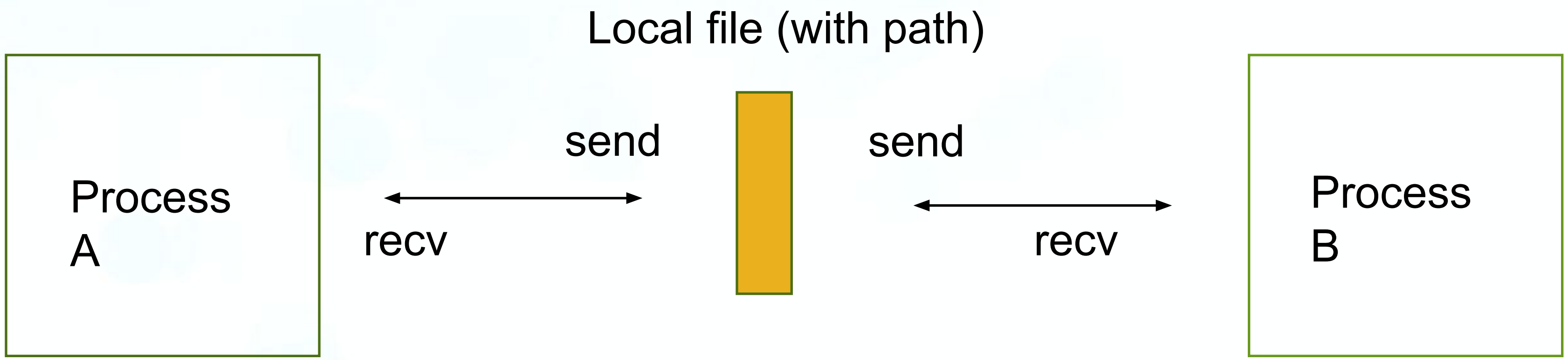
Socket v.s Http request

- Socket
  - Stateful
  - Faster
  - Full duplex
- HTTP
  - Stateless
    - header
  - Slower
  - Half duplex
    - One-way direction



Unix Socket

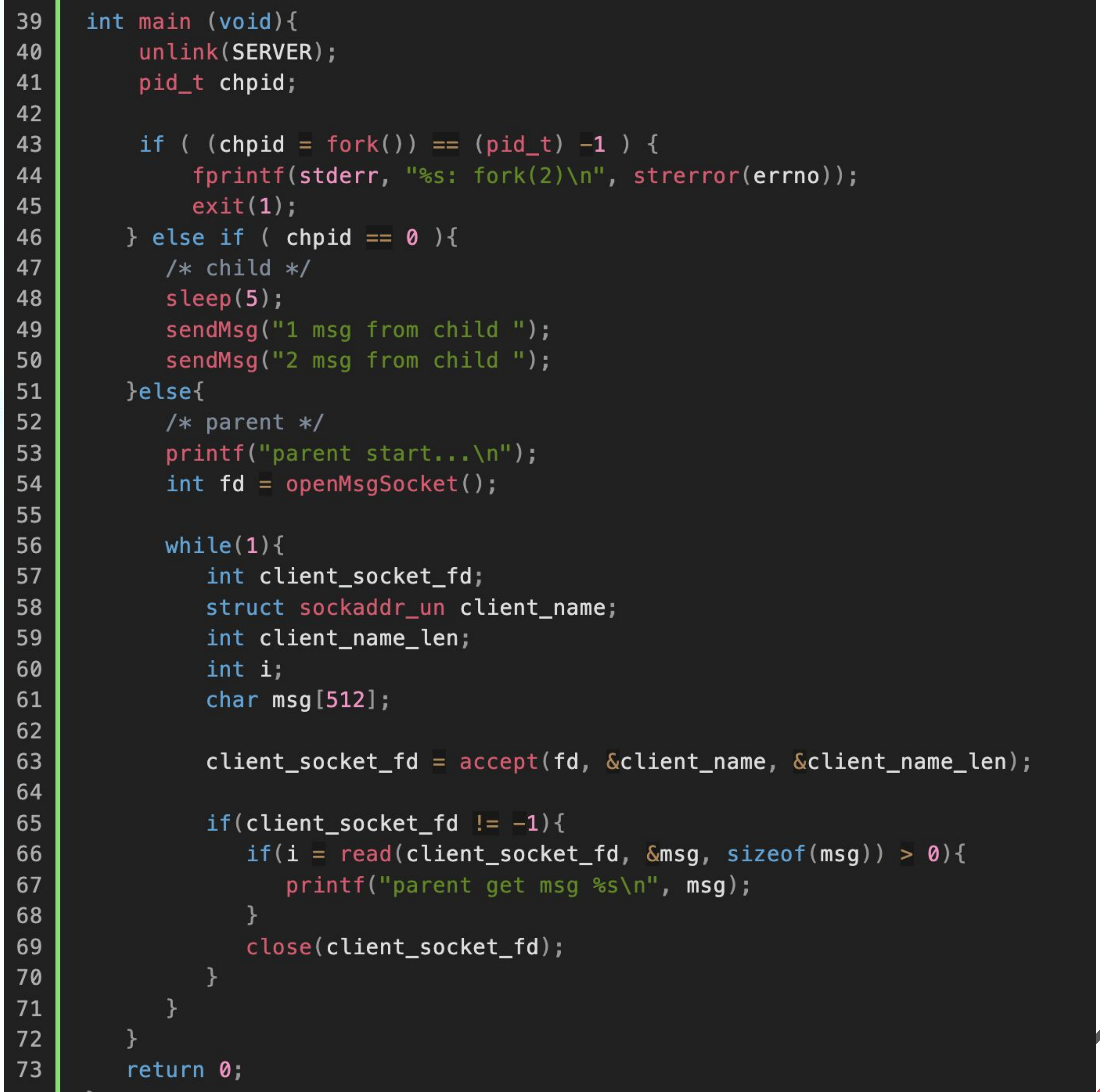
- A UNIX socket is an **inter-process** communication mechanism
- allow bidirectional data exchange between processes running on the same machine.



Unix Socket

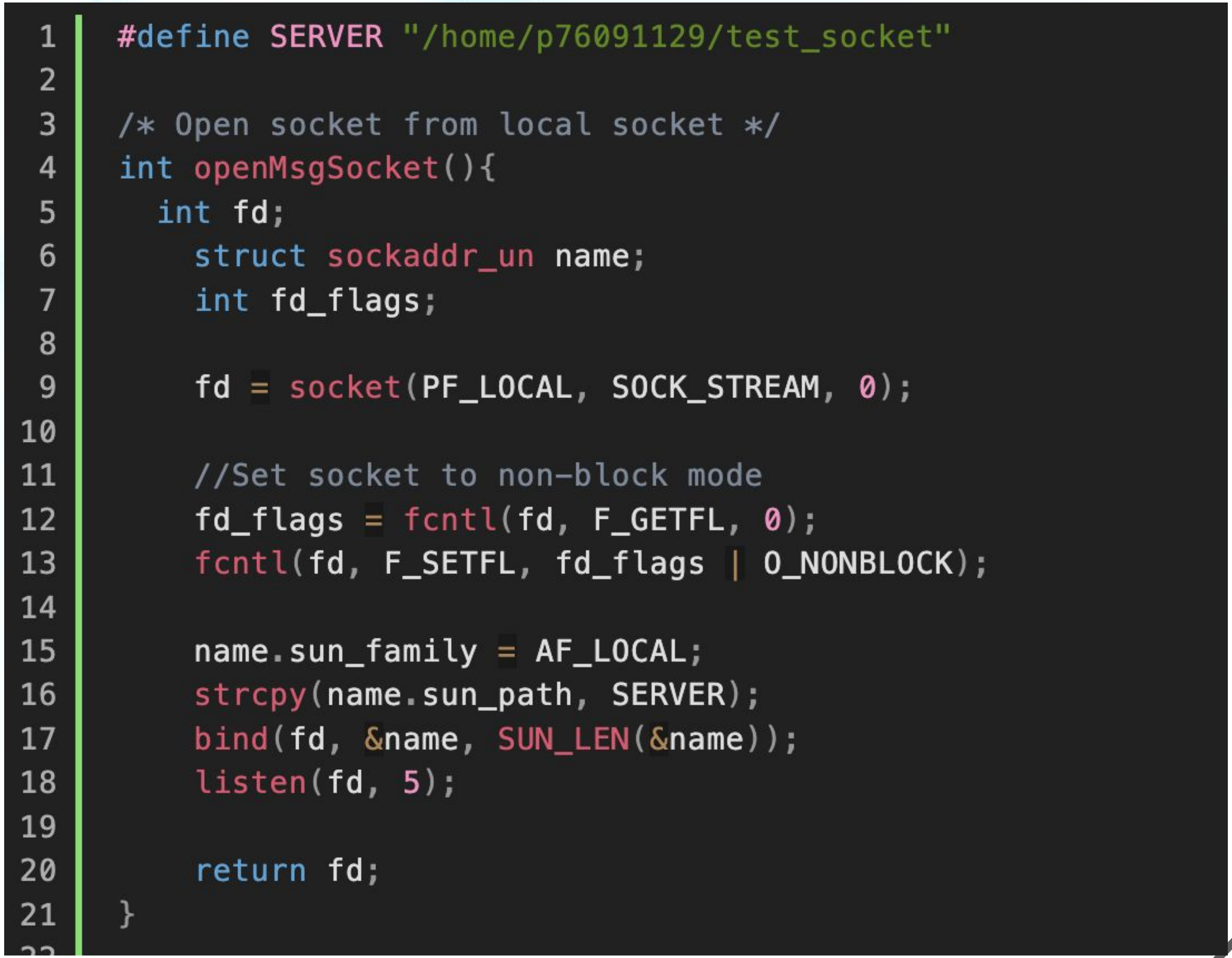
Child (client)

Parent (server)



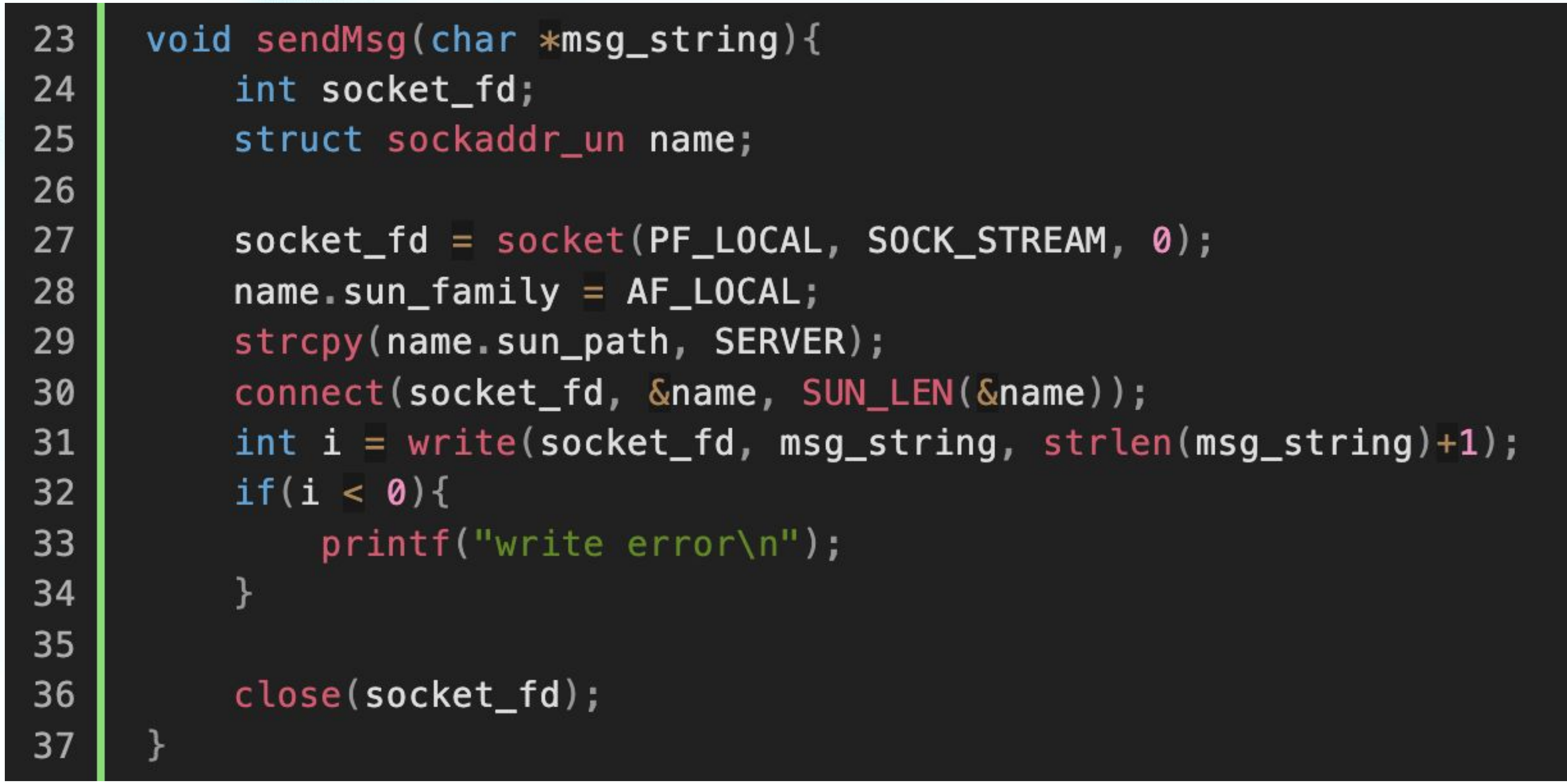
```c
int main (void){
    unlink(SERVER);
    pid_t chpid;

    if ( (chpid = fork()) == (pid_t) -1 ) {
        fprintf(stderr, "%s: fork(2)\n", strerror(errno));
        exit(1);
   } else if ( chpid == 0 ){
      /* child */
      sleep(5);
      sendMsg("1 msg from child ");
      sendMsg("2 msg from child ");
   }else{
      /* parent */
      printf("parent start...\n");
      int fd = openMsgSocket();

      while(1){
         int client_socket_fd;
         struct sockaddr_un client_name;
         int client_name_len;
         int i;
         char msg[512];

         client_socket_fd = accept(fd, &client_name, &client_name_len);

         if(client_socket_fd != -1){
            if(i = read(client_socket_fd, &msg, sizeof(msg)) > 0){
               printf("parent get msg %s\n", msg);
            }
            close(client_socket_fd);
         }
      }
   }
   return 0;
```

Unix Socket



```c
#define SERVER "/home/p76091129/test_socket"

/* Open socket from local socket */
int openMsgSocket(){
  int fd;
    struct sockaddr_un name;
    int fd_flags;

    fd = socket(PF_LOCAL, SOCK_STREAM, 0);

    //Set socket to non-block mode
    fd_flags = fcntl(fd, F_GETFL, 0);
    fcntl(fd, F_SETFL, fd_flags | O_NONBLOCK);

    name.sun_family = AF_LOCAL;
    strcpy(name.sun_path, SERVER);
    bind(fd, &name, SUN_LEN(&name));
    listen(fd, 5);

    return fd;
}
```

Unix Socket



```c
void sendMsg(char *msg_string){
    int socket_fd;
    struct sockaddr_un name;

    socket_fd = socket(PF_LOCAL, SOCK_STREAM, 0);
    name.sun_family = AF_LOCAL;
    strcpy(name.sun_path, SERVER);
    connect(socket_fd, &name, SUN_LEN(&name));
    int i = write(socket_fd, msg_string, strlen(msg_string)+1);
    if(i < 0){
        printf("write error\n");
    }

    close(socket_fd);
}
```

TCP Socket

- IP sockets (especially TCP/IP sockets) are a mechanism allowing communication between processes over the network.
- Use TCP/IP sockets to talk with processes running on the same computer (by using the loopback interface).

TCP Socket Coding

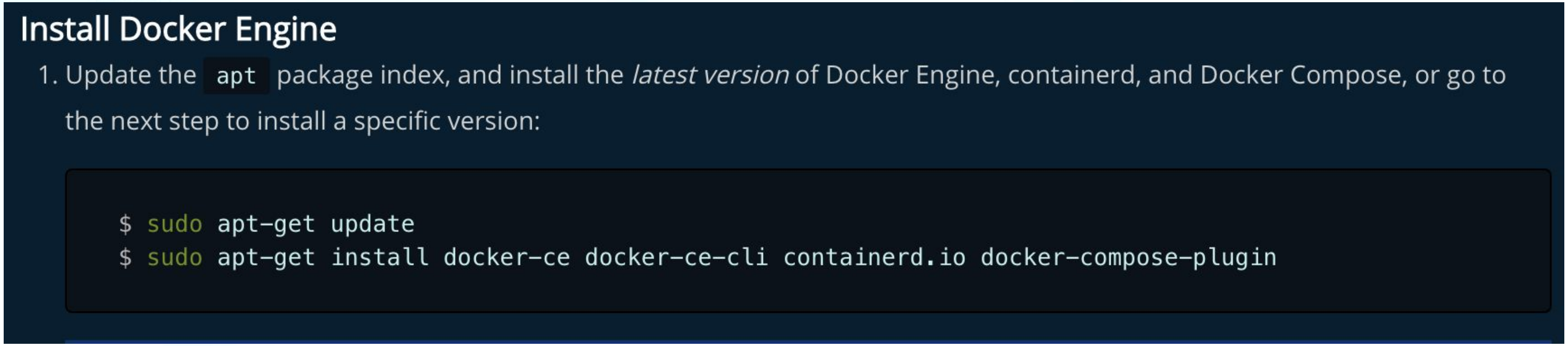
- Code reading
  - git clone https://github.com/jonec76/w14_socket.git
- tmux
  - ctrl+b -> “ 分割 new terminal
  - ctrl+b -> 方向鍵
- Implement the “broadcast” function
  - ps.
    A socket can only transfer bytes and therefore it needs to get bytes.
    A string is not a sequence of bytes but a sequence of characters.
- Implement the “ls” function
  - list all members in chat room

02

Docker

Docker

- Install docker
- Use **container** to manage to the process
- Build the **image** for executing the specific program easily



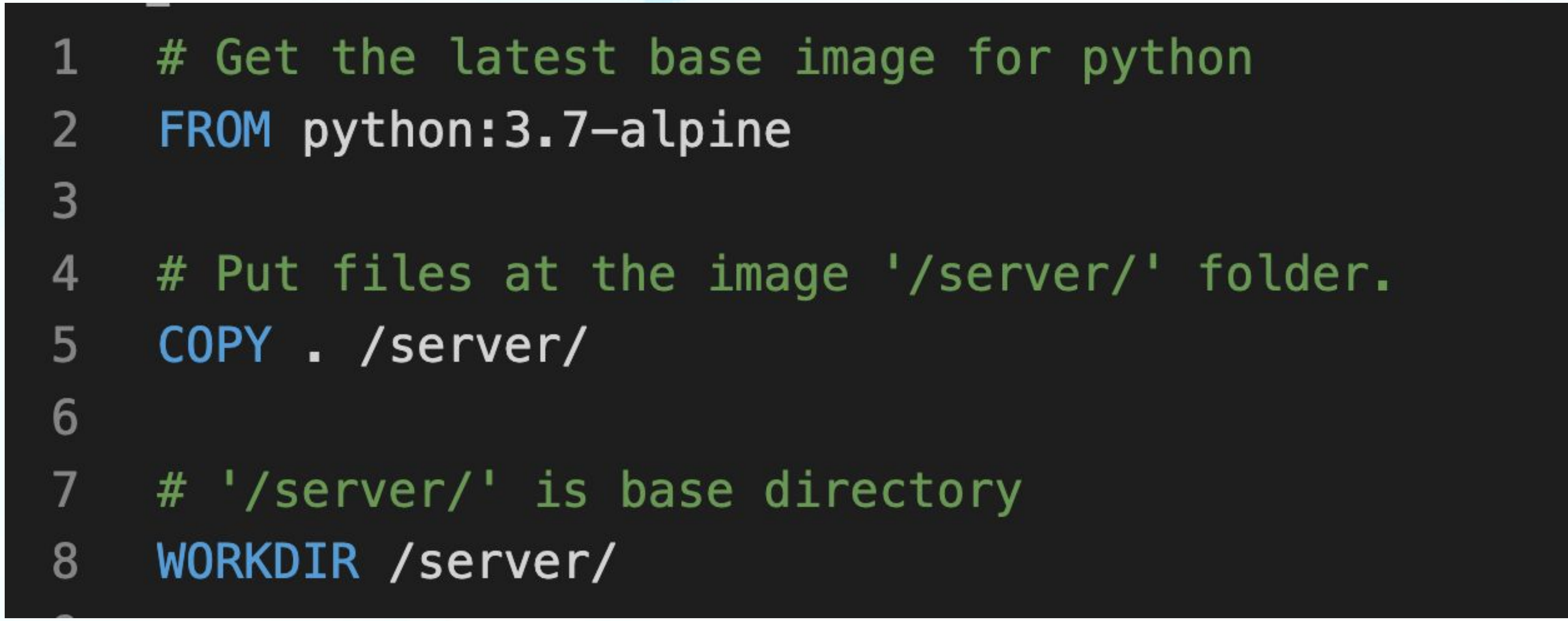
Install Docker Engine

1. Update the `apt` package index, and install the *latest version* of Docker Engine, containerd, and Docker Compose, or go to the next step to install a specific version:

```
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

Docker image (1)

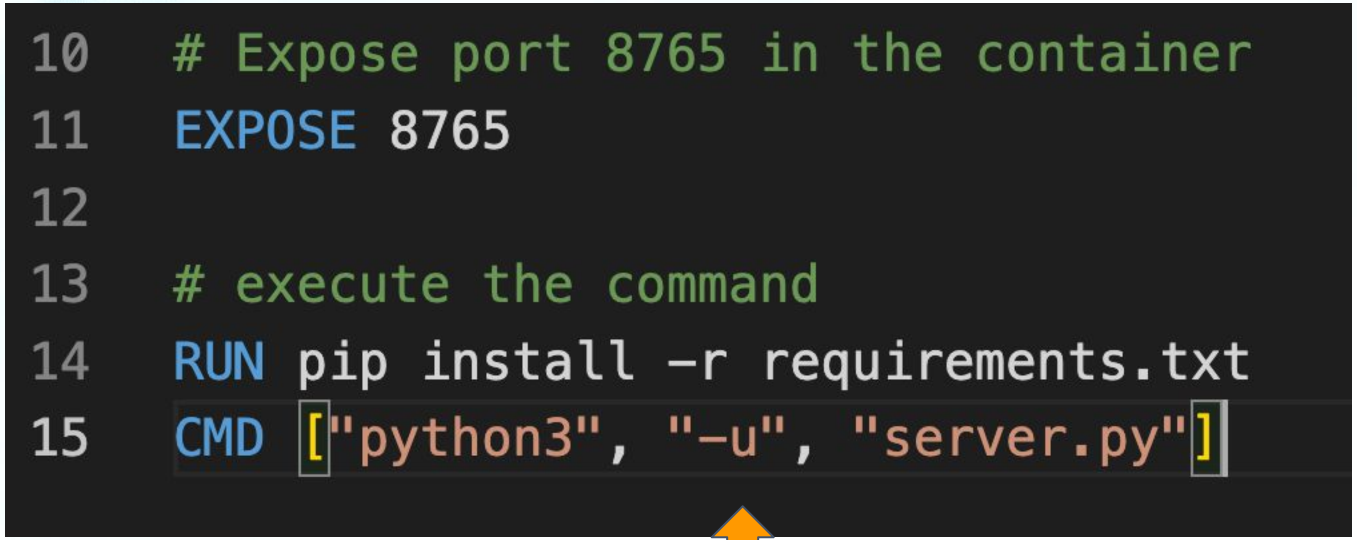
- Dockerfile



```
# Get the latest base image for python
FROM python:3.7-alpine

# Put files at the image '/server/' folder.
COPY . /server/

# '/server/' is base directory
WORKDIR /server/
```

Docker image (2)

- Dockerfile



```
# Expose port 8765 in the container
EXPOSE 8765

# execute the command
RUN pip install -r requirements.txt
CMD ["python3", "-u", "server.py"]
```

docker build -f server_docker -t w14_server .

docker run -d -p 8765:8765 w14_server