a. Simply explain your code. (10 points)

1. run.sh

```
#!/bin/bash
inotifywait -m /usr/src/app/src -e close_write |
    while read path action file; do
        echo "The file '$file' appeared in directory '$path' via '$action'"
        #upload to server
        ext="${file##*.}"
        filename="${file%.*}"
        case "${ext}" in
          c)
           yes | scp ./src/$file root@172.28.1.2:/usr/src/app/src
           ;;
          cpp)
           yes | scp ./src/$file root@172.28.1.2:/usr/src/app/src
           ;;
          java)
           yes | scp ./src/$file root@172.28.1.2:/usr/src/app/src
           ;;
        esac
done
```

過濾 c c++ java 當有 close_write event 便傳送到 server:/usr/src/app/src

```
#!/bin/bash
inotifywait -m /usr/src/app/src -e create |
    while read path action file; do
        echo "The file '$file' appeared in directory '$path' via '$action'"
        #upload to server
        ext="${file##*.}"
        filename="${file%.*}"
        case "${ext}" in
          c)
           gcc ./src/$file -o ./src/${filename}"_out"
           yes | scp ./src/${filename}"_out" root@172.28.1.1:/usr/src/app/result
           ;;
          cpp)
           g++ ./src/$file -o ./src/${filename}"_out"
           yes | scp ./src/${filename}"_out" root@172.28.1.1:/usr/src/app/result
           ;;
          java)
           javac ./src/$file
           ;;
          class)
           yes | scp ./src/$file root@172.28.1.1:/usr/src/app/result
           ;;
        esac
done
```

根據副檔名 compile 並回傳到 client: /usr/src/app/result

2. Docker-compose

```yaml
version: '3'
services:
  client:
    image: loststar1991/demo_client
    container_name: client1
    tty: true
    stdin_open: true
    command: '/usr/src/app/run.sh'
    networks:
      mybridge:
        ipv4_address: 172.28.1.1
    volumes:
      - /Users/Zack/Documents/DockerFile/client/src:/usr/src/app/src
      - /Users/Zack/Documents/DockerFile/client/result:/usr/src/app/result
  server:
#    image: loststar1991/demo_server
    build:
      context: ./server
      dockerfile: Dockerfile
    container_name: server1
    tty: true
    stdin_open: true
    command: '/usr/src/app/run.sh'
    networks:
      mybridge:
        ipv4_address: 172.28.1.2
    volumes:
      - /Users/Zack/Documents/DockerFile/server/src:/usr/src/app/src
```

使用現有的 image loststar1991/demo_client

Container 名稱 client1

開啟後執行 run.sh

使用 mybridge 固定 ip 172.28.1.1

Bind volume src

使用 ./server/Dockerfile 建立 image

```yaml
networks:
  mybridge:
    driver: bridge
    ipam:
      driver: default
      config:
      - subnet: 172.28.0.0/16
```

建立 bridge 名稱 mybridge 可用 172.28.0.1~172.28.255.255

b. What is the main difference between container and VM? (10 points)

非常輕量化 指安裝所跑的 app 安裝所需程式包裝成 image 可以再任意 host 上執行 不必擔心環境設定的問題

c. What are the three method for container to store data in host machine, what is

the difference between them? (10 points)

1.Volumes 存在 host filesystem 中內容只能由 container 裡的應用修改

2.Bind mounts 存在 host filesystem 中與 host 資料夾同步 可同時由 host 與 container 修改內容

3.tmpfs mounts 只存在 host 記憶體不會寫進 host filesystem 中