網路程式設計 HW5-report 0410024 余東儒

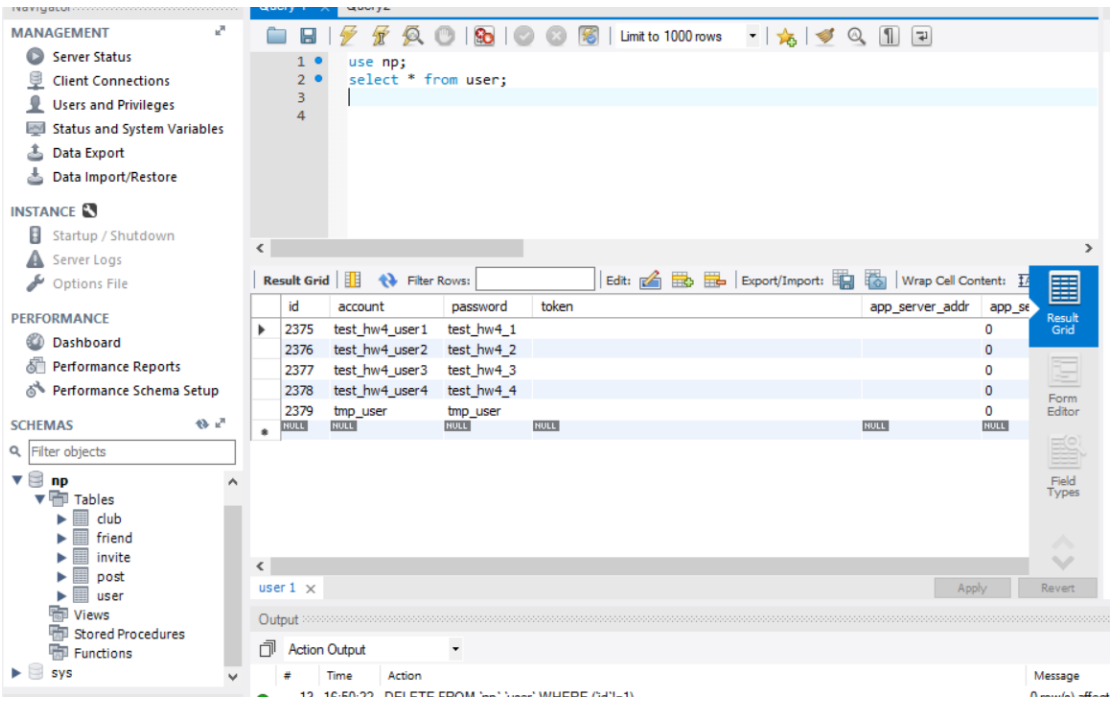
一、環境設定

首先先辦一個 AWS 的帳號，然後創建一組 access key，再來就是在先開一個 ec2 的 instance，我這邊是用 ubuntu 的 OS，然後需要設定的地方為 security group，在 inbound 跟 outbound 我設定為 All traffic 都可以 access。創立好 instance 之後，用 ssh 連上 instance，下載我需要的套件，我用 anaconda 作為套件管理，下載 peewee、pymysql、Stomp 和 boto3，另外再載 activemq 的安裝程式並安裝，另外也有下載 AWS 的 CLI，用-aws configure 這個指令將一開始下載的 access key 輸入，這樣才能連到我的 aws account。

```
conda install -c conda-forge peewee
conda install -c conda-forge pymysql
conda install -c conda-forge stomp.py
conda install -c conda-forge boto3
```

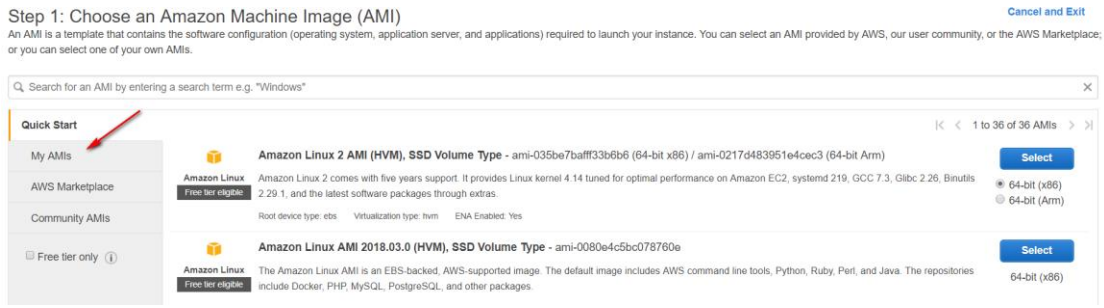
接下來將我的 login_server 和 app_server 分別丟到兩個 instance，再來將兩個 instance 分別做成 AMI，以供之後程式創立 instance 用，

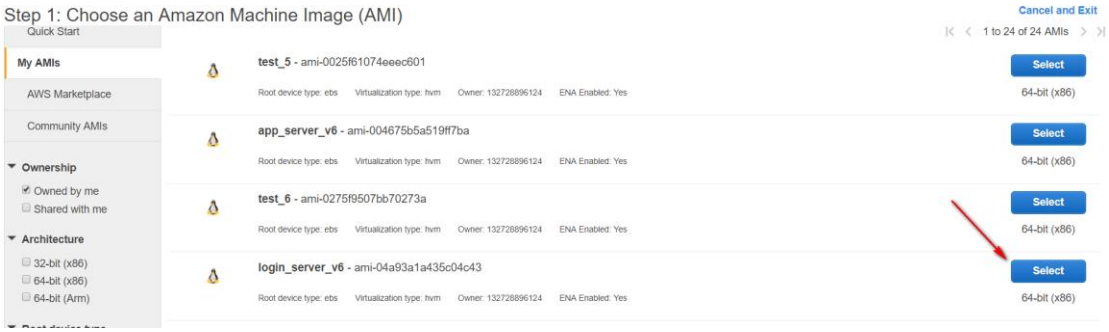
我的 database 也是放在 AWS 上，我在 RDS 上 create 一個 MySQL 的 database，然後用 workbench 連線到 AWS 的 MySQL，進行 database 的建立。



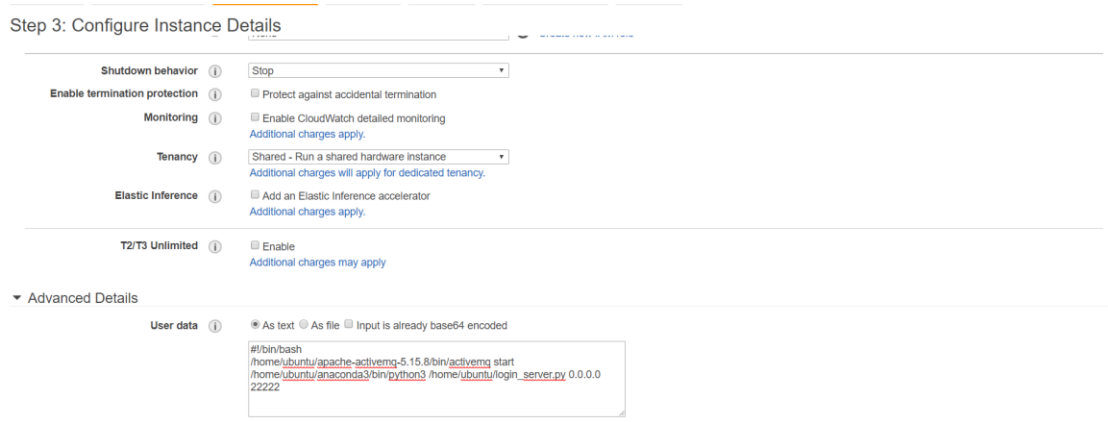
二、程式執行方式

Login_server 的部分我直接在 aws 的 console create，選擇 MY AMIS，選擇先前創立好給 login_server 的 AMI，

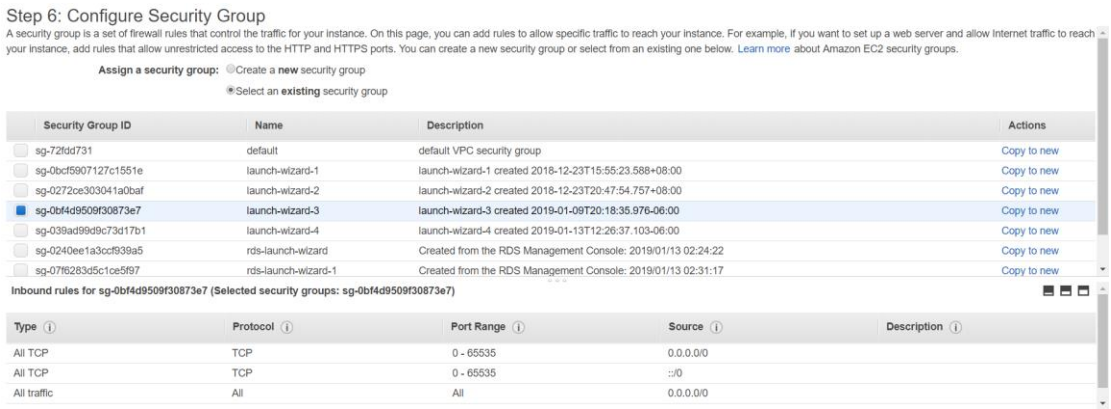




再來在 step 3 這裡的 User data 輸入我們在開機時要的指令，這邊第一行指令是將 activemq 開啟，第二行則是執行我的 login_server python 檔，ip 為 0.0.0.0 是因為要開放給全部人連線。

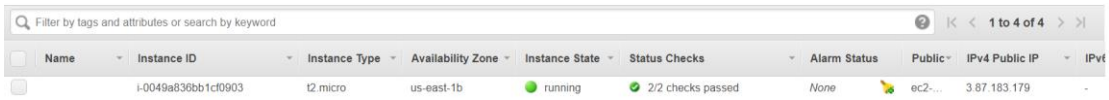


在 Step 6 這邊則要選擇剛剛創立好的 security group，這樣才能讓其他人連到這台 server。



最後創建好的這個 instance 就是我的 login_server，記下 public IP 以供我的 client

能夠連線。

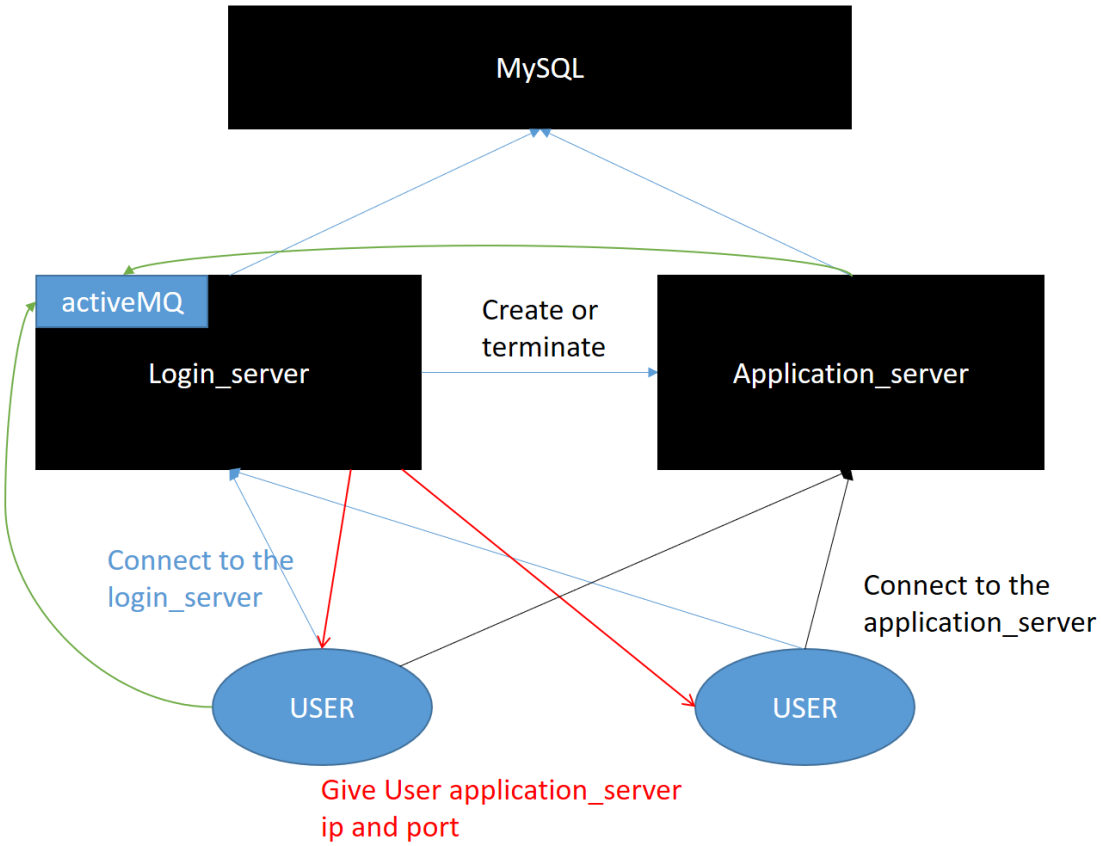


只要在 cmd 上直接執行 client 程式就好 後面的 arg 為 login_server 的連線 ip 和 port。

```
tommyyu@tommyyu-VirtualBox:~$ python3 client_hw5.py 3.87.183.179 22222 <./test_hw5/1.txt >1.txt
tommyyu@tommyyu-VirtualBox:~$ vim 1.txt
tommyyu@tommyyu-VirtualBox:~$ python3 client_hw5.py 3.87.183.179 22222 <./test_hw5/2.txt >2.txt
tommyyu@tommyyu-VirtualBox:~$ python3 client_hw5.py 3.87.183.179 22222 <./test_hw5/3.txt >3.txt
tommyyu@tommyyu-VirtualBox:~$ python3 client_hw5.py 3.87.183.179 22222 <./test_hw5/4.txt >4.txt
tommyyu@tommyyu-VirtualBox:~$ python3 client_hw5.py 3.87.183.179 22222 <./test_hw5/5.txt >5.txt
tommyyu@tommyyu-VirtualBox:~$ python3 client_hw5.py 3.87.183.179 22222 <./test_hw5/6.txt >6.txt
tommyyu@tommyyu-VirtualBox:~$ vimdiff 1.txt ./answer_hw5/1.txt
```

三、程式流程圖

三個黑色的區塊是架在 AWS，application_server 是 login_server 由 boto3 控制建立或是中止，login_server 同時也會記錄 applicatoin_server 的 ip，當 user 向 login_server 要求 login 時，login_server 同時也會分配一個 ip 給他，而所有的資料。ActiveMQ 則是架在 login_server，供 User、application_server 和 login_server 使用。



四、程式碼說明

這邊主要說明為了將之前作業放到 AWS 所增加或改變的 code

1. Client 端在 login 的時候，會用一個 dictionary 將 user 和對應的 application_ip 和 port 存下來，每一個 cmd 連結到 application server 或是 login server。

```
elif(cmd=="login"):
    tmp=command.find(" ")

    user=command[tmp+1:command.find(" ",tmp+1)]
    status,msg,token,app_conn_addr,app_conn_port=login(command)
    app_HOST=app_conn_addr
    app_PORT=app_conn_port
    if(status==0):
        login_dict.update({user:token})
        app_conn_dict.update({user:(app_HOST,app_PORT)})
    #print(status,msg,token)
    output_msg=msg+'\n'
    #OUTPUT.writelines(output_msg)
    print(msg)
```

2. login_server.py 函式 app_server_assign 是當 login_server 發現沒有 app server 存在或是沒空位的時候，就會互叫這個函式，用 boto3 新建 instance 當作 app server，一樣用 user data 讓 server 一開始就 run command。

```
def app_server_assign():
    ec2 = boto3.resource('ec2')
    th=ec2.Instance(login_iid)
    this_addr=th.public_ip_address

    user_data="""#!/bin/bash
    /home/ubuntu/anaconda3/bin/python3 /home/ubuntu/app_server.py 0.0.0.0 22222 """+this_addr
    #user_data+=
    ec2 = boto3.resource('ec2', region_name = 'us-east-1')
    a=ec2.create_instances(
        BlockDeviceMappings=[
          {
              'DeviceName':"/dev/sda1",
              'Ebs':{
                  'DeleteOnTermination':True,
                #'SnapshotId':'snap-07a5307a919c1486b'
                }
          }
        ],
        ImageId='ami-07077b75d639ddc57',
        NetworkInterfaces=[
            {
                'AssociatePublicIpAddress': True,
                'DeviceIndex': 0,
            'Groups': ["sg-0bf4d9509f30873e7"],

                'SubnetId': 'subnet-d9690ff7'
            }
        ],
        MinCount=1,
        MaxCount=1,
        KeyName='ec2_key',

        InstanceType="t2.micro",

        UserData=user_data
    )


    iid=a[0].instance_id
    #print(iid)
    time.sleep(3)
    b=ec2.Instance(iid)
    time.sleep(20)

    return iid,b.public_ip_address
try:
```

3. 這個部分則是當 user login 時，login server 判斷是否有 app server 空位，login_server.py 用 instances_list 和 instances_dict 紀錄曾創建過的 app server 和其連線數，也會將 user 和其連線到的 ip 記錄到 database 中，以防 client 中止前沒有將 user logout。

```
elif row["token"]=="":
    token=uuid.uuid4().hex
    print(token)
    User.update(**{"token": token}).where(User.account==user,User.password==pw).execute()
    #######Instance job#########
    app_conn_addr=""
    app_conn_port=22222
    app_iid=""
    if len(instances_list)==0:
        app_iid,app_conn_addr=app_server_assign()
        #app_conn_port=app_conn_init_port

        instances_list.append((app_iid,app_conn_addr))
        instances_dict.setdefault(app_conn_addr,1)
    else:
        flag=0 #all full or not
        for inst in instances_list:
            if instances_dict[inst[1]]<10:
                app_conn_addr=inst[1]
                #app_conn_port=app_conn_init_port+instances_dict[inst]
                instances_dict[inst[1]]+=1
                flag=1
                break
            else:
                flag=0
            #elif instances_dict[inst]
        if flag==0:
            app_iid,app_conn_addr=app_server_assign()
            #app_conn_port=app_conn_init_port
            instances_list.append((app_iid,app_conn_addr))
            instances_dict.setdefault(app_conn_addr,1)
    print('app:'+app_conn_addr,app_conn_port)
    User.update(**{"app_server_addr": app_conn_addr}).where(User.account==user,User.password==pw).execute()
    User.update(**{"app_server_port": app_conn_port}).where(User.account==user,User.password==pw).execute()
```

而 logout 的時候也會判斷 instance 是否已經 idle，若 idle 則直接中止，delete 同理。

```
    instances_dict[row["app_server_addr"]]-=1
    if instances_dict[row["app_server_addr"]]==0:

        del instances_dict[row["app_server_addr"]]




        for i,a in enumerate(instances_list):
            if a[1]==row["app_server_addr"]:
                client = boto3.client('ec2')
                client.terminate_instances(InstanceIds=[a[0]])
                instances_list.pop(i)
                print(a[1]+" shut down")
    User.update(**{"token": ""}).where(User.account==row["account"]).execute()
    User.update(**{"app_server_addr": ""}).where(User.account==row["account"]).execute()
    User.update(**{"app_server_port": 0}).where(User.account==row["account"]).execute()



    #print("Bye")
    msg="Bye!"
    msg_status=0
```

4. Application_server 的部分則沒太多變動，就只是把先前的 server code 切到另一個程式去。