Lab 2: EC2 and Docker

Section 1: Creating an EC2 instance using awscli

1. Using the command `aws ec2 create-security-group --group-name <student number>-sg --description "security group for development environment"`, a security group is created. The security group ID is noted for use in future steps.

```
jookai@jookai:~$ aws ec2 create-security-group --group-name 22489437-sg --descri
ption "security group for development environment"
{
    "GroupId": [redacted]
}
```

2. The following command is used to authorise inbound traffic for ssh: `aws ec2 authorize-security-group-ingress --group-name <student number>-sg --protocol tcp --port 22 --cidr 0.0.0.0/0`. Some things to note about this command: a. The TCP protocol is specified using `--protocol tcp --port 22` and the default port 22 is selected. b. The command `--cidr 0.0.0.0/0` is used as a wildcard. This allows all IP addresses to connect to SSH port that was opened. This is considered a major security risk as it allows anyone on the internet to attempt SSH connections to instances associated with this security group.

```
jookai@jookai:~$ aws ec2 authorize-security-group-ingress --group-name 22489437-
sg --protocol tcp --port 22 --cidr 0.0.0.0/0
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "[redacted]",
            "GroupId": "[redacted]",
            "GroupOwnerId": [redacted],
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIpv4": "0.0.0.0/0"
        }
    ]
}
```

3. The command `aws ec2 create-key-pair --key-name <student number>-key --query 'KeyMaterial' --output text > <student number>-key.pem` creates a key pair that will allow a user to ssh to the EC2 instance. The private key is stored in the output file `22489437-key.pem`

```
jookai@jookai:~$ aws ec2 create-key-pair --key-name 22489437-key --query 'KeyMaterial' -
-output text > 22489437-key.pem
```

4. The private key file is moved into the SSH directory and its permissions are changed to be read only for the owner of the file, and no permissions for others.

```
jookai@jookai:~/Desktop$ mv 22489437-key.pem ~/.ssh
jookai@jookai:~/Desktop$ cd ~/.ssh
jookai@jookai:~/.ssh$ ls
22489437-key.pem
jookai@jookai:~/.ssh$ chmod 400 22489437-key.pem
```

5. The command ` aws ec2 run-instances --image-id ami-d38a4ab1 --security-group-ids -sg --count 1 --instance-type t2.micro --key-name -key --query 'Instances[0].InstanceId' is used to create the EC2 instance. The command returns the instance ID of the created instance.

```
jookai@jookai:~$ aws ec2 run-instances --image-id ami-d38a4ab1 --security-group-ids 22489437-sg
--count 1 --instance-type t2.micro --key-name 22489437-key --query 'Instances[0].InstanceId'
"i-0e757a5193269f9da"
```

6. Using the instance ID from step 5, a tag is added to the instance.

```
jookai@jookai:~$ aws ec2 create-tags --resources i-0e757a5193269f9da --tags Key=lab2,Value=22489
437
```

7. The public IP address of this instance is determined using the command `aws ec2 describe-instances --instance-ids i-<instance id from above> --query 'Reservations[0].Instances[0].PublicIpAddress'`

```
jookai@jookai:~$ aws ec2 describe-instances --instance-ids i-0e757a5193269f9da --query 'Reservat
ions[0].Instances[0].PublicIpAddress'
"3.27.149.106"
```

8. Using the private key file from step 3 and the IP address from step 7, the user is able to SSH into the EC2 instance.

```
jookai@jookai:~/.ssh$ ssh -i 22489437-key.pem ubuntu@3.27.149.106
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-1052-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

9. The created instance can be viewed from the AWS console

10. The instance can be terminated from the awscli using the following command: `aws ec2 terminate-instances --instance-ids i-<your instance id>`

```
jookai@jookai:~$ aws ec2 terminate-instances --instance-ids i-0e757a5193269f9da
{
    "TerminatingInstances": [
        {
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "InstanceId": "i-0e757a5193269f9da",
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

Section 2: Creating an AWS EC2 Instance with Python Boto Script

1. The following code is used to replicate steps 1-7 from section 1:

```
import boto3
import os
import shutil

student_number = "22489437"
region = "ap-southeast-2"

# Initialize the EC2 client
ec2 = boto3.client('ec2', region)

# Create a security group
response = ec2.create_security_group(
    GroupName=f"{student_number}-sg",
    Description="security group for development environment"
)
security_group_id = response['GroupId']

# Authorize inbound SSH traffic for the security group
ec2.authorize_security_group_ingress(
    GroupId=security_group_id,
    IpProtocol="tcp",
    FromPort=22,
    ToPort=22,
    CidrIp="0.0.0.0/0"
)

# Create a key pair and save the private key to a file
response = ec2.create_key_pair(KeyName=f"{student_number}-key")
private_key = response['KeyMaterial']

private_key_file = f"{student_number}-key.pem"
# Allow writing to the private key file
os.chmod(private_key_file, 0o666)
with open(private_key_file, 'w') as key_file:
    key_file.write(private_key)

# Set the correct permissions for the private key file
os.chmod(private_key_file, 0o400)

# Copy the private key file to ~/.ssh directory
ssh_directory = os.path.expanduser("~/.ssh")
if not os.path.exists(ssh_directory):
    os.makedirs(ssh_directory)
shutil.copy(private_key_file, ssh_directory)

# Launch an EC2 instance
response = ec2.run_instances(
    ImageId="ami-d38a4ab1",
    SecurityGroupIds=[security_group_id],
    MinCount=1,
    MaxCount=1,
```

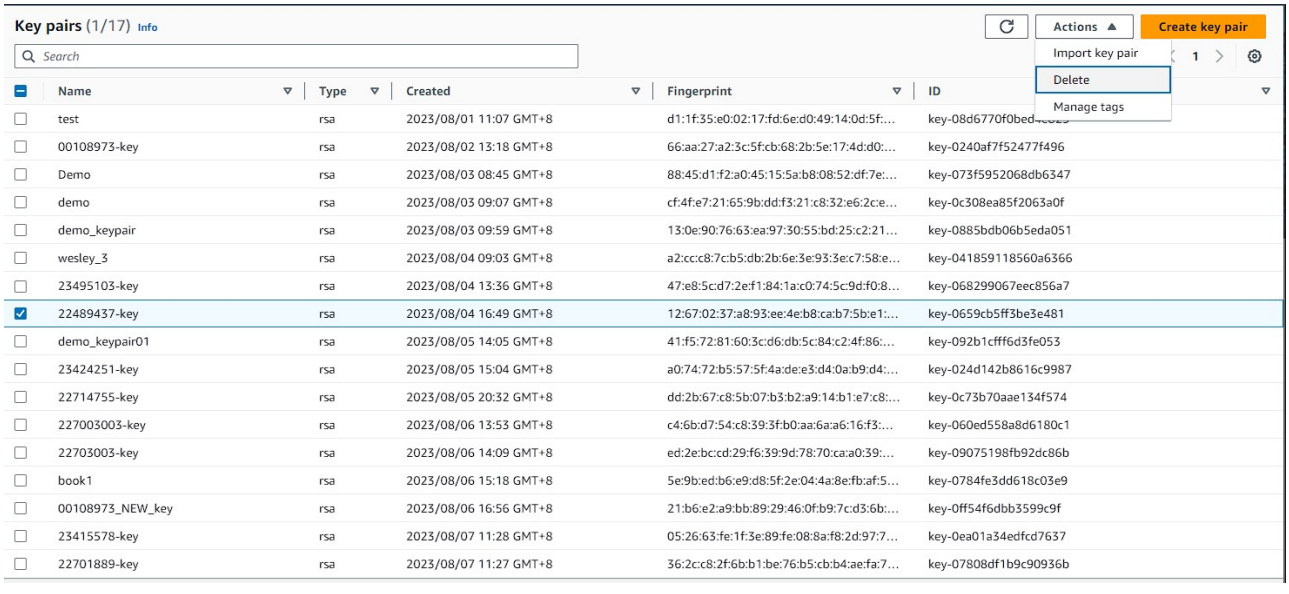
```
        InstanceType="t2.micro",
        KeyName=f"{student_number}-key"
    )
    instance_id = response['Instances'][0]['InstanceId']

    # Wait for the instance to be up and running
    ec2.get_waiter('instance_running').wait(InstanceIds=[instance_id])

    # Describe the instance to get its public IP address
    response = ec2.describe_instances(InstanceIds=[instance_id])
    public_ip_address = response['Reservations'][0]['Instances'][0]
['PublicIpAddress']
    print(f"Instance created successfully with Public IP: {public_ip_address}")
```

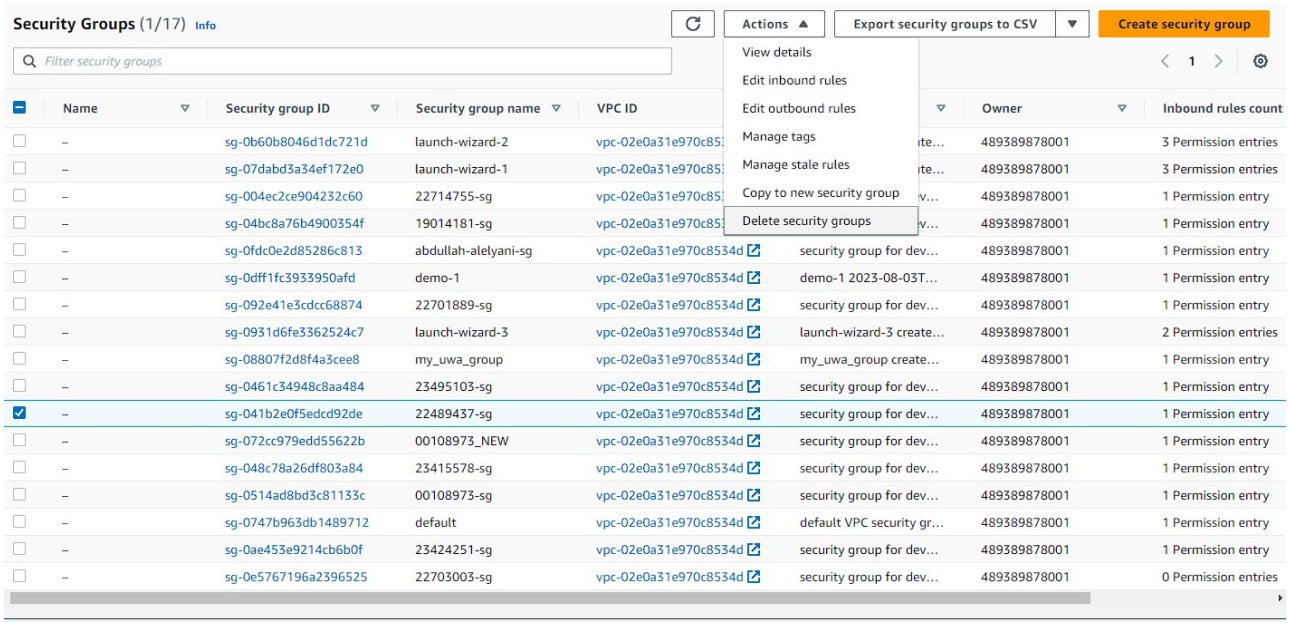
2. An error was shown when running the script. This was due to the existing key pair that was created in step 1. In order to resolve this error, we head into the AWS console and delete the previously created key pair.

```
jookai@jookai:~/Desktop$ python3 lab2.py
Traceback (most recent call last):
  File "/home/jookai/Desktop/lab2.py", line 27, in <module>
    response = ec2.create_key_pair(KeyName=f"{student_number}-key")
  File "/home/jookai/.local/lib/python3.10/site-packages/botocore/client.py", line 535, in _api_cal
l
    return self._make_api_call(operation_name, kwargs)
  File "/home/jookai/.local/lib/python3.10/site-packages/botocore/client.py", line 980, in _make_ap
i_call
    raise error_class(parsed_response, operation_name)
botocore.exceptions.ClientError: An error occurred (InvalidKeyPair.Duplicate) when calling the Crea
teKeyPair operation: The keypair already exists
```



3. A second error was shown when running the script. This was due to the existing security group that was created in step 1. In order to resolve this error, we head into the AWS console and delete the previously created security group.

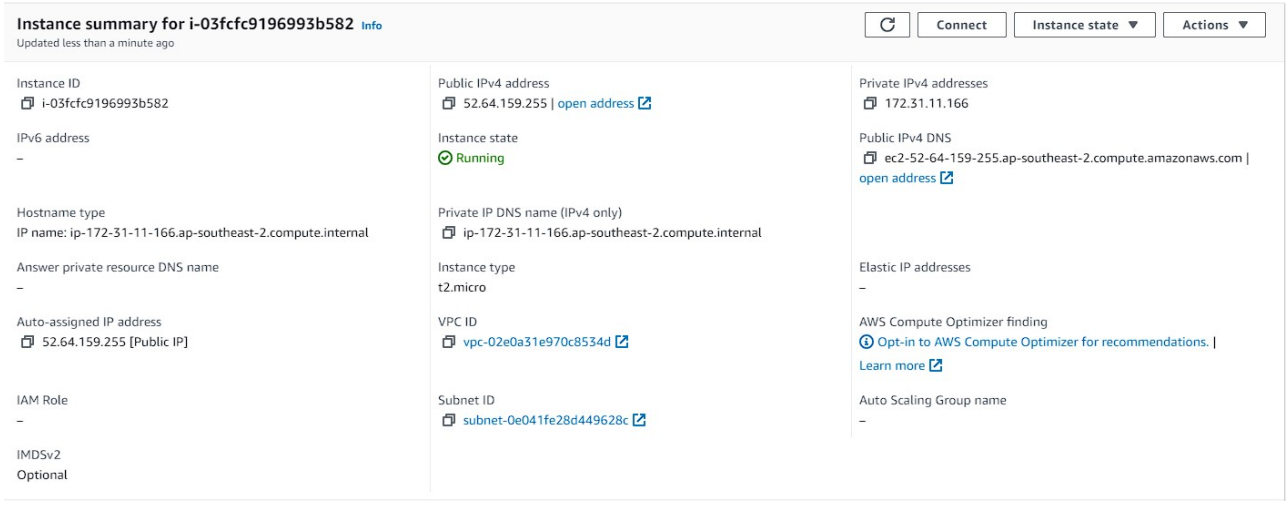
```
jookai@jookai:~/Desktop$ python3 lab2.py
Traceback (most recent call last):
  File "/home/jookai/Desktop/lab2.py", line 11, in <module>
    response = ec2.create_security_group(
  File "/home/jookai/.local/lib/python3.10/site-packages/botocore/client.py", line 535, in _api_cal
l
    return self._make_api_call(operation_name, kwargs)
  File "/home/jookai/.local/lib/python3.10/site-packages/botocore/client.py", line 980, in _make_ap
i_call
    raise error_class(parsed_response, operation_name)
botocore.exceptions.ClientError: An error occurred (InvalidGroup.Duplicate) when calling the Create
SecurityGroup operation: The security group '22489437-sg' already exists for VPC 'vpc-02e0a31e970c8
534d'
```



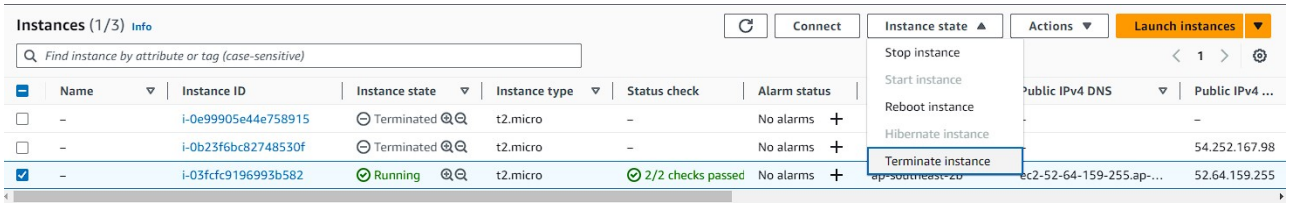
4. The script ran successfully and returned he public IP of the created EC2 instance:

```
jookai@jookai:~/Desktop$ python3 lab2.py
Instance created successfully with Public IP: 52.64.159.255
```

5. The details of the EC2 instance on the AWS console:



6. This instance was terminated using the AWS console instead of the AWSCLI, demonstrating the second way to terminate an EC2 instance:



Section 3: Using Docker

1. Docker was installed using the command `sudo apt install docker.io -y`

```
jookai@jookai:~/Desktop$ sudo apt install docker.io -y
```

2. Enabling docker and checking the version installed:

```
jookai@jookai:~/Desktop$ sudo systemctl start docker
jookai@jookai:~/Desktop$ sudo systemctl enable docker
jookai@jookai:~/Desktop$ docker --version
Docker version 20.10.25, build 20.10.25-0ubuntu1~22.04.1
```

3. Creating index.html:

```
jookai@jookai:~/Desktop/cits5503/lab2/html$ cat index.html
  <html>
    <head> </head>
    <body>
      <p>Hello World!</p>
    </body>
  </html>
```

4. Creating the Dockerfile

```
jookai@jookai:~/Desktop/cits5503/lab2$ cat Dockerfile
FROM httpd:2.4
COPY ./html/ /usr/local/apache2/htdocs/
```

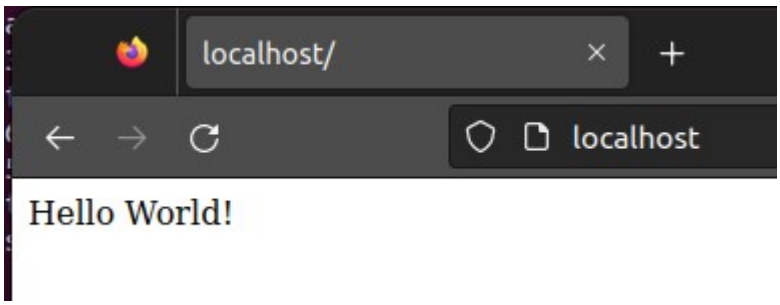
5. Building the docker image from the dockerfile and index.html

```
jookai@jookai:~/Desktop/cits5503/lab2$ sudo docker build -t my-apache2 .
[sudo] password for jookai:
Sending build context to Docker daemon  10.24kB
Step 1/2 : FROM httpd:2.4
2.4: Pulling from library/httpd
648e0aadf75a: Pull complete
c76ba39af630: Pull complete
b9819ffb14ec: Pull complete
37baa60548e6: Pull complete
6dbce5de7542: Pull complete
Digest: sha256:d7262c0f29a26349d6af45199b2770d499c74d45cee5c47995a1ebb336093088
Status: Downloaded newer image for httpd:2.4
 ---> 96a2d0570deb
Step 2/2 : COPY ./html/ /usr/local/apache2/htdocs/
 ---> b0b174fe899f
Successfully built b0b174fe899f
Successfully tagged my-apache2:latest
```

6. Running the docker container:

```
jookai@jookai:~/Desktop/cits5503/lab2$ sudo docker run -p 80:80 -dit --name my-a
pp my-apache2
962f91e96c313a1bdf3a31875836116e5d353d3b1bd8a3a0571fb6b58b4c3777
```

7. Confirming the "Hello World!" output:



8. Viewing what is running:

```
jookai@jookai:~/Desktop/cits5503/lab2$ sudo docker ps -a
CONTAINER ID   IMAGE        COMMAND              CREATED          STATUS
  PORTS                                NAMES
962f91e96c31   my-apache2   "httpd-foreground"   56 seconds ago   Up 56 seconds
  0.0.0.0:80->80/tcp, :::80->80/tcp   my-app
```

9. Terminating the container:

```
jookai@jookai:~/Desktop/cits5503/lab2$ sudo docker stop my-app
my-app
```

```
jookai@jookai:~/Desktop/cits5503/lab2$ sudo docker rm my-app
my-app
```