

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 --description "security group for development environment"
{
  "GroupId": "sg-0a1b2c3d4e5f6g7h"
}
```

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": "sg-rule-0a1b2c3d4e5f6g7h",
      "GroupId": "sg-0a1b2c3d4e5f6g7h",
      "GroupOwnerId": "i-0a1b2c3d4e5f6g7h",
      "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=22489437
```

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 'Reservations[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

Instances (1) [info](#)

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring	Security group name	Key name	Launch time
<input type="checkbox"/>	-	i-0e757a5193269f9da	Running	t2.micro	2/2 checks passed	No alarms	ap-southeast-2b	ec2-3-27-149-106.ap-s...	3.27.149.106	-	-	disabled	22489437-sg	22489437-key	2023/08/04 16:54 GMT+8

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_call
    return self._make_api_call(operation_name, kwargs)
  File "/home/jookai/.local/lib/python3.10/site-packages/botocore/client.py", line 980, in _make_api_call
    raise error_class(parsed_response, operation_name)
botocore.exceptions.ClientError: An error occurred (InvalidKeyPair.Duplicate) when calling the CreateKeyPair operation: The keypair already exists

```

Name	Type	Created	Fingerprint	ID
test	rsa	2023/08/01 11:07 GMT+8	d1:1f:35:e0:02:17:fd:6e:d0:49:14:0d:5f:...	key-08d6770f0bed...
00108973-key	rsa	2023/08/02 13:18 GMT+8	66:aa:27:a2:3c:5f:cb:68:2b:5e:17:4d:d0:...	key-0240af7f5247f496
Demo	rsa	2023/08/03 08:45 GMT+8	88:45:d1:f2:a0:45:15:5a:b8:08:52:df:7e:...	key-073f5952068db6347
demo	rsa	2023/08/03 09:07 GMT+8	cf:4f:e7:21:65:9b:dd:f3:21:c8:32:e6:2c:...	key-0c308ea85f2063a0f
demo_keypair	rsa	2023/08/03 09:59 GMT+8	13:0e:90:76:63:ea:97:30:55:bd:25:c2:21:...	key-0885db06b5eda051
wesley_3	rsa	2023/08/04 09:03 GMT+8	a2:cc:c8:7c:b5:db:2b:6e:3e:93:3e:c7:58:e:...	key-041859118560a6366
23495103-key	rsa	2023/08/04 13:36 GMT+8	47:e8:5cd7:2ef1:84:1a:c0:74:5c:9d:f0:8:...	key-068299067eec856a7
22489437-key	rsa	2023/08/04 16:49 GMT+8	12:67:02:37:a8:93:ee:4e:b8:ca:b7:5b:e1:...	key-0659cb5ff3be3e481
demo_keypair01	rsa	2023/08/05 14:05 GMT+8	41:f5:72:81:60:3c:d6:db:5c:84:c2:4f:86:...	key-092b1cff6d3fe053
23424251-key	rsa	2023/08/05 15:04 GMT+8	a0:74:72:b5:57:5f:4a:de:e3:d4:0a:b9:d4:...	key-024d142b8616c9987
22714755-key	rsa	2023/08/05 20:32 GMT+8	dd:2b:67:c8:5b:07:b3:b2:a9:14:b1:e7:c8:...	key-0c73b70aae134f574
227003003-key	rsa	2023/08/06 13:53 GMT+8	c4:6b:d7:54:c8:39:3f:b0:aa:6a:a6:16:f3:...	key-060ed558a8d6180c1
22703003-key	rsa	2023/08/06 14:09 GMT+8	ed:2e:bcc:d2:9f:63:9d:78:70:ca:a0:39:...	key-09075198fb92dc86b
book1	rsa	2023/08/06 15:18 GMT+8	5e:9b:ed:b6:e9:d8:5f:2e:04:4a:8e:fb:af:5:...	key-0784fe3dd618c03e9
00108973_NEW_key	rsa	2023/08/06 16:56 GMT+8	21:b6:e2:a9:bb:89:29:46:0f:b9:7c:d3:6b:...	key-0ff54f6dbb3599c9f
23415578-key	rsa	2023/08/07 11:28 GMT+8	05:26:63:fe:1f:3e:89:fe:08:8a:f8:2d:97:7:...	key-0ea01a34edfcd7637
22701889-key	rsa	2023/08/07 11:27 GMT+8	36:2cc:8:2f:6b:b1:be:76:b5:cb:b4:ae:fa:7:...	key-07808df1b9c90936b

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_call
    return self._make_api_call(operation_name, kwargs)
  File "/home/jookai/.local/lib/python3.10/site-packages/botocore/client.py", line 980, in _make_api_call
    raise error_class(parsed_response, operation_name)
botocore.exceptions.ClientError: An error occurred (InvalidGroup.Duplicate) when calling the CreateSecurityGroup operation: The security group '22489437-sg' already exists for VPC 'vpc-02e0a31e970c8534d'
```

Security Groups (1/17) Info					Actions	Export security groups to CSV	Create security group
Filter security groups					View details		
					Edit inbound rules		
					Edit outbound rules		
					Manage tags		
					Manage stale rules		
					Copy to new security group		
					Delete security groups		
Name	Security group ID	Security group name	VPC ID	Owner	Inbound rules count		
-	sg-0b60b8046d1dc721d	launch-wizard-2	vpc-02e0a31e970c8534d	489389878001	3 Permission entries		
-	sg-07dabd3a34ef172e0	launch-wizard-1	vpc-02e0a31e970c8534d	489389878001	3 Permission entries		
-	sg-004ec2ce904232c60	22714755-sg	vpc-02e0a31e970c8534d	489389878001	1 Permission entry		
-	sg-04bc8a76b4900354f	19014181-sg	vpc-02e0a31e970c8534d	489389878001	1 Permission entry		
-	sg-0fdc0e2d85286c813	abdullah-alelyani-sg	vpc-02e0a31e970c8534d	489389878001	1 Permission entry		
-	sg-0dff1fc3933950afd	demo-1	vpc-02e0a31e970c8534d	489389878001	1 Permission entry		
-	sg-092e41e3cdcc68874	22701889-sg	vpc-02e0a31e970c8534d	489389878001	1 Permission entry		
-	sg-0931d6fe3362524c7	launch-wizard-3	vpc-02e0a31e970c8534d	489389878001	2 Permission entries		
-	sg-08807f2d8f4a3cee8	my_uwa_group	vpc-02e0a31e970c8534d	489389878001	1 Permission entry		
-	sg-0461c34948c8aa484	23495103-sg	vpc-02e0a31e970c8534d	489389878001	1 Permission entry		
-	sg-041b2e0f5edcd92de	22489437-sg	vpc-02e0a31e970c8534d	489389878001	1 Permission entry		
-	sg-072cc979edd55622b	00108973_NEW	vpc-02e0a31e970c8534d	489389878001	1 Permission entry		
-	sg-048c78a26df803a84	23415578-sg	vpc-02e0a31e970c8534d	489389878001	1 Permission entry		
-	sg-0514ad8bd3c81133c	00108973-sg	vpc-02e0a31e970c8534d	489389878001	1 Permission entry		
-	sg-0747b963db1489712	default	vpc-02e0a31e970c8534d	489389878001	1 Permission entry		
-	sg-0ae453e9214cb6b0f	23424251-sg	vpc-02e0a31e970c8534d	489389878001	1 Permission entry		
-	sg-0e5767196a2396525	22703003-sg	vpc-02e0a31e970c8534d	489389878001	0 Permission entries		

4. The script ran successfully and returned the 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:

Instance summary for i-03fcfc9196993b582 Info			Connect	Instance state	Actions
Updated less than a minute ago					
Instance ID i-03fcfc9196993b582	Public IPv4 address 52.64.159.255 open address	Private IPv4 addresses 172.31.11.166			
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-52-64-159-255.ap-southeast-2.compute.amazonaws.com open address			
Hostname type IP name: ip-172-31-11-166.ap-southeast-2.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-11-166.ap-southeast-2.compute.internal	Elastic IP addresses -			
Answer private resource DNS name -	Instance type t2.micro	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more			
Auto-assigned IP address 52.64.159.255 [Public IP]	VPC ID vpc-02e0a31e970c8534d	Auto Scaling Group name -			
IAM Role -	Subnet ID subnet-0e041fe28d449628c				
IMDSv2 Optional					

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

Instances (1/3) Info							Connect	Instance state ▲	Actions ▼	Launch instances ▼
Find instance by attribute or tag (case-sensitive)								Stop instance		< 1 > ⓘ
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status		Start instance	Public IPv4 DNS	Public IPv4 ...
<input type="checkbox"/>	-	i-0e99905e44e758915	Terminated ⓘ ⓘ	t2.micro	-	No alarms +		Reboot instance	-	-
<input type="checkbox"/>	-	i-0b23f6bc82748530f	Terminated ⓘ ⓘ	t2.micro	-	No alarms +		Hibernate instance	-	54.252.167.98
<input checked="" type="checkbox"/>	-	i-03fcfc9196993b582	Running ⓘ ⓘ	t2.micro	2/2 checks passed	No alarms +		Terminate instance	ec2-52-64-159-255.ap-...	52.64.159.255

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/
```

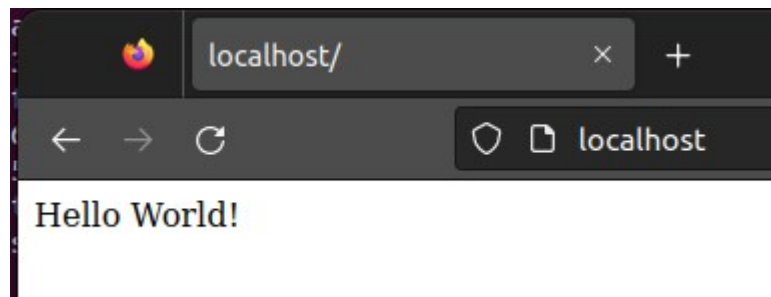
5. Building the docker image from the dockerfile and index.html


```
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-app 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	NAMES	CREATED	STATUS
962f91e96c31	my-apache2	"httpd-foreground"	my-app	56 seconds ago	Up 56 seconds
0.0.0.0:80->80/tcp, :::80->80/tcp					

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
```