

Labs 1-5

Student ID: 23796349

Student Name: Adharsh Sundaram Soudakar

Lab 1

AWS Account and Log in

[1] Log into an IAM user account created for you on AWS

- Navigate to the AWS login page [AWS Console](#).
- Sign in with the provided username ('student_ID@student.uwa.edu.au) and password.
- Once signed in, a **change your password** prompt will appear.
- On successfully changing the password, you will redirected to the AWS dashboard.

[2] Search and open Identity Access Management

- Click on your user profile and click on **Security Credentials**.
- Scroll down to the **Access Keys** section and click on **Create access key** button.
- Choose the **Other** option and click **Next**.
- Set a description tag(optional) and click on **Create access key**.
- Make note of the **Access ID key** and **Secret Access key** or download the **csv** file that contains these information.

Set up recent Linux OSes

- I already have Ubuntu OS 22.04.4 LTS set up on Oracle VM VirtualBox.



```
adharsh@adharsh:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 22.04.4 LTS
Release: 22.04
Codename: jammy
adharsh@adharsh:~$
```

- This was done when I was enrolled in the units **CITS1003 - Introduction to Cybersecurity** and **CITS4407 - Open Source Tools and Scripting**.

Install Linux packages

[1] Install Python 3.8.x

- I already have the latest version of Python installed.



```
adharsh@adharsh:~$ python3 -V
Python 3.10.12
adharsh@adharsh:~$
```

- Again, this was done when I was enrolled in the unit **CITS1003 - Introduction to Cybersecurity**.

[2] Install awscli

- Open a terminal and enter the following commands:

```
sudo apt install awscli
```

```

Activities Terminal Jul 29 17:16
adharsh@adharsh:~$ sudo apt install awscli
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libawscli-common libcurl4-gnutls-dev libimage Magick libimage Magick-6-common
  libimage Magick-6-common libcurl4-openssl-dev libjxr-tools libjxr0 liblqr-1-0
  libmagickcore-6.q16-6 libmagickcore-6.q16-6-extra libmagickwand-6.q16-6
  libnetpbm10 netpbm psutils python3-botocore python3-docutils
  python3-jmespath python3-pyasn1 python3-pymgments python3-roman python3-rsa
  python3-s3transfer
Suggested packages:
  imagemagick-doc autotrace enscript ffpmpeg gnuplot grads hp2xx html2ps
  libwmf-bin mplayer povray radience texlive-base-bin transfig uraw-batch
  liblfrw3-bin liblfrw3-devinkscape docutils-doc fonts-linuxlibertine
  ttf-linux-libertine texlive-lang-french texlive-latex-base
  texlive-latex-recommended texlive-latex-recommended ttf-bitstream-vera
The following NEW packages will be installed:
  awscli docutils-common groff gsfonts imagemagick imagemagick-6-common
  imagemagick-6.q16 liblfrw3-double3 libjxr-tools libjxr0 liblqr-1-0
  libmagickcore-6.q16-6 libmagickcore-6.q16-6-extra libmagickwand-6.q16-6
  libnetpbm10 netpbm psutils python3-botocore python3-docutils
  python3-jmespath python3-pyasn1 python3-pymgments python3-roman python3-rsa
  python3-s3transfer
0 to upgrade, 25 to newly install, 0 to remove and 47 not to upgrade.
Need to get 18.9 MB of archives.
After this operation, 117 MB of additional disk space will be used.
Do you want to continue? [Y/n] y

```

Followed by,

```
pip3 install awscli --upgrade` or `sudo snap install aws-cli --classic
```

```

adharsh@adharsh:~$ pip3 install awscli --upgrade
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: awscli in /usr/lib/python3/dist-packages (1.22.34)
Collecting awscli
  Downloading awscli-1.33.31-py3-none-any.whl (4.5 MB)
    4.5/4.5 MB 6.6 MB/s eta 0:00:00
Requirement already satisfied: PyYAML<6.1,>=3.10 in /usr/lib/python3/dist-packages (from awscli) (5.4.1)
Requirement already satisfied: colorama<0.4.7,>=0.2.5 in /usr/lib/python3/dist-packages (from awscli) (0.4.4)
Collecting botocore==1.34.149
  Downloading botocore-1.34.149-py3-none-any.whl (12.4 MB)
    12.4/12.4 MB 6.6 MB/s eta 0:00:00
Collecting s3transfer<0.11.0,>=0.10.0
  Downloading s3transfer-0.10.2-py3-none-any.whl (82 kB)
    82.7/82.7 KB 2.7 MB/s eta 0:00:00
Collecting docutils<0.17,>=0.10
  Downloading docutils-0.16-py2.py3-none-any.whl (548 kB)
    548.2/548.2 KB 6.8 MB/s eta 0:00:00
Collecting rsa<4.8,>=3.1.2
  Downloading rsa-4.7.2-py3-none-any.whl (34 kB)
Requirement already satisfied: jmespath>=2.0.0,<2.1 in /usr/lib/python3/dist-packages (from botocore==1.34.149->awscli) (2.10.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3/dist-packages (from botocore==1.34.149->awscli) (2.8.1)
Requirement already satisfied: urllib3!=2.0.0,<1.27 in /usr/lib/python3/dist-packages (from botocore==1.34.149->awscli) (1.26.5)
Requirement already satisfied: pyasn1<0.1.3 in /usr/lib/python3/dist-packages (from rsa>4.8,>=3.1.2->awscli) (0.4.8)
Successfully installed awscli-1.33.31 botocore-1.34.149 docutils-0.16 rsa-4.7.2 s3transfer-0.10.2
adharsh@adharsh:~$ 

```

- To ensure correct installation, try checking the version using the following command:

```
aws --version
```

[3] Configure AWS

- To configure AWS, enter the following command:

```
aws configure
```

- When prompted, enter your **AWS Access Key ID** and **AWS Secret Access Key**.
- Enter your **Default region name**, in my case it is **ap-northeast-2**.
- Set the **Default output format** to **json**.

```

successfully installed awscli-1.33.31 botocore-1.34.149 docutils-0.16 rsa-4.7.2 s3transfer-0.10.2
adharsh@adharsh:~$ aws configure
AWS Access Key ID [None]: AKIAXD4PISLYJYIH57F
AWS Secret Access Key [None]: hzkfK8+7VP7nV4pIvlelm/446a3Kwxc+XXcKhZl
Default region name [None]: ap-northeast-2
Default output format [None]: json
adharsh@adharsh:~$ 

```

[4] Install boto3

- To install the **Boto3** library, enter the following command:

```
pip3 install boto3
```

```
adharsh@adharsh:~$ pip3 install boto3
Defaulting to user installation because normal site-packages is not writeable
Collecting boto3
  Downloading boto3-1.34.149-py3-none-any.whl (139 kB)
    139.2/139.2 KB 3.5 MB/s eta 0:00:00
Requirement already satisfied: s3transfer<0.11.0,>=0.10.0 in ./local/lib/python3.10/site-packages (from boto3) (0.10.2)
Requirement already satisfied: botocore<1.35.0,>=1.34.149 in ./local/lib/python3.10/site-packages (from boto3) (1.34.149)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3/dist-packages (from boto3) (0.10.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3/dist-packages (from botocore<1.35.0,>=1.34.149->boto3) (2.8.1)
Requirement already satisfied: urllib3!=2.2.0,<3,>=1.25.4 in /usr/lib/python3/dist-packages (from botocore<1.35.0,>=1.34.149->boto3) (1.26.5)
Installing collected packages: boto3
Successfully installed boto3-1.34.149
adharsh@adharsh:~$
```

Test the installed environment

[1] Test the AWS environment

- To test the AWS environment, for example, to display the available regions with their respective endpoint names , enter the following command:

```
aws ec2 describe-regions --output table
```

```
adharsh@adharsh:~$ aws ec2 describe-regions --output table
|-----+-----+-----+
|       | DescribeRegions |           |
|-----+-----+-----+
|       | Regions          |           |
|-----+-----+-----+
|   Endpoint      | OptInStatus | RegionName |
|-----+-----+-----+
| ec2.ap-south-1.amazonaws.com | opt-in-not-required | ap-south-1 |
| ec2.eu-north-1.amazonaws.com | opt-in-not-required | eu-north-1 |
| ec2.eu-west-3.amazonaws.com | opt-in-not-required | eu-west-3 |
| ec2.eu-west-2.amazonaws.com | opt-in-not-required | eu-west-2 |
| ec2.eu-west-1.amazonaws.com | opt-in-not-required | eu-west-1 |
| ec2.ap-northeast-3.amazonaws.com | opt-in-not-required | ap-northeast-3 |
| ec2.ap-northeast-2.amazonaws.com | opt-in-not-required | ap-northeast-2 |
| ec2.ap-northeast-1.amazonaws.com | opt-in-not-required | ap-northeast-1 |
| ec2.ca-central-1.amazonaws.com | opt-in-not-required | ca-central-1 |
| ec2.sa-east-1.amazonaws.com | opt-in-not-required | sa-east-1 |
| ec2.ap-southeast-1.amazonaws.com | opt-in-not-required | ap-southeast-1 |
| ec2.ap-southeast-2.amazonaws.com | opt-in-not-required | ap-southeast-2 |
| ec2.eu-central-1.amazonaws.com | opt-in-not-required | eu-central-1 |
| ec2.us-east-1.amazonaws.com | opt-in-not-required | us-east-1 |
| ec2.us-east-2.amazonaws.com | opt-in-not-required | us-east-2 |
| ec2.us-west-1.amazonaws.com | opt-in-not-required | us-west-1 |
| ec2.us-west-2.amazonaws.com | opt-in-not-required | us-west-2 |
adharsh@adharsh:~$
```

[2] Test the Python environment

- To test the python environment, first we need to open the python interpreter:

```
python3
```

- Again to display the available regions with their respective endpoint names (un-tabulated), enter the following python code:

```
>>> import boto3
>>> ec2 = boto3.client('ec2')
>>> response = ec2.describe_regions()
>>> print(response)
```

```

viharsh@viharsh:~$ python3
Python 3.10.12 (main, Mar 22 2024, 16:50:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import boto3
>>> ec2 = boto3.client('ec2')
>>> response = ec2.describe_regions()
>>> print(response)
{'Regions': [{"Endpoint": "ec2.ap-south-1.amazonaws.com", "RegionName": "ap-south-1", "OptInStatus": "opt-in-not-required"}, {"Endpoint": "ec2.eu-north-1.amazonaws.com", "RegionName": "eu-north-1", "OptInStatus": "opt-in-not-required"}, {"Endpoint": "ec2.eu-west-2.amazonaws.com", "RegionName": "eu-west-2", "OptInStatus": "opt-in-not-required"}, {"Endpoint": "ec2.eu-west-3.amazonaws.com", "RegionName": "eu-west-3", "OptInStatus": "opt-in-not-required"}, {"Endpoint": "ec2.ap-northwest-1.amazonaws.com", "RegionName": "ap-northwest-1", "OptInStatus": "opt-in-not-required"}, {"Endpoint": "ec2.ap-northeast-1.amazonaws.com", "RegionName": "ap-northeast-1", "OptInStatus": "opt-in-not-required"}, {"Endpoint": "ec2.ap-northeast-2.amazonaws.com", "RegionName": "ap-northeast-2", "OptInStatus": "opt-in-not-required"}, {"Endpoint": "ec2.ap-northeast-3.amazonaws.com", "RegionName": "ap-northeast-3", "OptInStatus": "opt-in-not-required"}, {"Endpoint": "ec2.ap-north-1.amazonaws.com", "RegionName": "ap-north-1", "OptInStatus": "opt-in-not-required"}, {"Endpoint": "ec2.ca-central-1.amazonaws.com", "RegionName": "ca-central-1", "OptInStatus": "opt-in-not-required"}, {"Endpoint": "ec2.sa-east-1.amazonaws.com", "RegionName": "sa-east-1", "OptInStatus": "opt-in-not-required"}, {"Endpoint": "ec2.ap-southeast-1.amazonaws.com", "RegionName": "ap-southeast-1", "OptInStatus": "opt-in-not-required"}, {"Endpoint": "ec2.ap-southeast-2.amazonaws.com", "RegionName": "ap-southeast-2", "OptInStatus": "opt-in-not-required"}, {"Endpoint": "ec2.us-east-1.amazonaws.com", "RegionName": "us-east-1", "OptInStatus": "opt-in-not-required"}, {"Endpoint": "ec2.us-west-1.amazonaws.com", "RegionName": "us-west-1", "OptInStatus": "opt-in-not-required"}, {"Endpoint": "ec2.us-west-2.amazonaws.com", "RegionName": "us-west-2", "OptInStatus": "opt-in-not-required"}, {"Endpoint": "ec2.us-east-2.amazonaws.com", "RegionName": "us-east-2", "OptInStatus": "opt-in-not-required"}, {"Endpoint": "ec2.us-west-2.amazonaws.com", "RegionName": "us-west-2", "OptInStatus": "opt-in-not-required"}], "ResponseMetadata": {"RequestId": "c870faa2-3562-40ae-93ac-5c061753e375", "CacheControl": "no-cache, no-store", "Strict-Transport-Security": "max-age=31536000; includeSubDomains", "Vary": "accept-encoding", "Content-Type": "text/xml; charset=UTF-8", "Content-Length": "289"}, "Date": "Mon, 29 Jul 2024 09:40:34 GMT", "Server": "AmazonEC2", "RetryAttempts": 0}}
>>>

```

[3] Write a Python script

- To get a tabulated response of the above output with only **Endpoint** and **RegionName** columns, follow the steps below:
- Create a Python file eg: **lab1_script.py** and open it with a text editor (Vim) or an IDE (VScode).
- Type the following code and save the file:

```

import boto3
ec2 = boto3.client('ec2')
response = ec2.describe_regions()
print("-----+-----+")
print("|          Endpoint          |    RegionName   |")
print("-----+-----+")
for item in response["Regions"]:
    print("| {:<32} | {:<14} | ".format(item["Endpoint"],item["RegionName"]))
print("-----+-----+")

```

- Run the script using the command:

```
python3 lab1_script.py
```

```

viharsh@viharsh:~$ python3 lab1_script.py
-----+-----+
|          Endpoint          |    RegionName   |
|-----+-----+
| ec2.ap-south-1.amazonaws.com | ap-south-1      |
| ec2.eu-north-1.amazonaws.com | eu-north-1      |
| ec2.eu-west-3.amazonaws.com | eu-west-3       |
| ec2.eu-west-2.amazonaws.com | eu-west-2       |
| ec2.eu-west-1.amazonaws.com | eu-west-1       |
| ec2.ap-northeast-3.amazonaws.com | ap-northeast-3 |
| ec2.ap-northeast-2.amazonaws.com | ap-northeast-2 |
| ec2.ap-northeast-1.amazonaws.com | ap-northeast-1 |
| ec2.ca-central-1.amazonaws.com | ca-central-1    |
| ec2.sa-east-1.amazonaws.com | sa-east-1       |
| ec2.ap-southeast-1.amazonaws.com | ap-southeast-1 |
| ec2.ap-southeast-2.amazonaws.com | ap-southeast-2 |
| ec2.eu-central-1.amazonaws.com | eu-central-1    |
| ec2.us-east-1.amazonaws.com | us-east-1       |
| ec2.us-east-2.amazonaws.com | us-east-2       |
| ec2.us-west-1.amazonaws.com | us-west-1       |
| ec2.us-west-2.amazonaws.com | us-west-2       |
-----+-----+

```

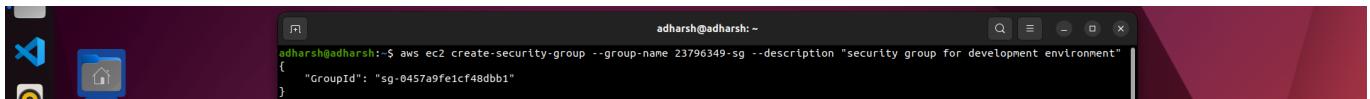
Lab 2

Create an EC2 instance using awscli

[1] Create a security group

- To create a **security group**, open a terminal and type the following command, use your student ID followed by sg:

```
aws ec2 create-security-group --group-name 23796349-sg --description "security group for development environment"
```

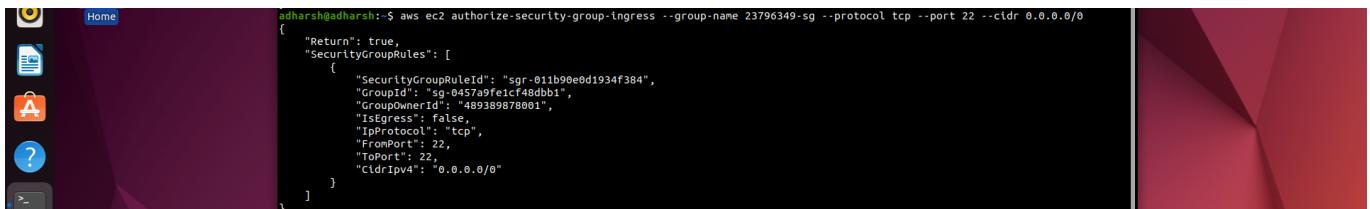


- Take a note of the security group id that is created.

[2] Authorise inbound traffic for ssh

- Type the following command to set **ssh authorisation**:

```
aws ec2 authorize-security-group-ingress --group-name 23796349-sg --protocol tcp - -port 22 --cidr 0.0.0.0/0
```



[3] Create a key pair

- Type the following command to create a **key pair** and save it in a file:

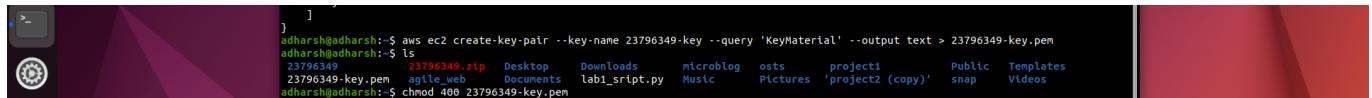
```
aws ec2 create-key-pair --key-name 23796349-key --query 'KeyMaterial' --output text > 23796349-key.pem
```

- To use this key on Linux, copy the file to a directory `~/.ssh` using the following command:

```
mv 23796349-key.pem ~/.ssh
```

and change its permissions to **read-only** using the following command:

```
chmod 400 23796349-key.pem
```



```
[1]
adharsh@adharsh:~$ aws ec2 create-key-pair --key-name 23796349-key --query 'KeyMaterial' --output text > 23796349-key.pem
adharsh@adharsh:~$ ls
23796349-key.pem  23796349.zip  Desktop  Downloads  microblog  osfs  project1  'project2 (copy)'  Public  Templates
adharsh@adharsh:~$ chmod 400 23796349-key.pem
```

[4] Create the instance

- Enter the following command to create an instance and get its ID.

NOTE: Use your allocated image-id, in my case it is **ami-056a29f2eddc40520**.

```
aws ec2 run-instances --image-id ami-056a29f2eddc40520 --security-group-ids
23796349-sg --count 1 --instance-type t2.micro --key-name 23796349-key --query
'Instances[0].InstanceId'
```

[5] Add a tag to your Instance

- To easily identify the instance, add a tag to it using the following command with the instance ID obtained from the previous step:

```
aws ec2 create-tags --resources i-073e156e4154b59db --tags
Key=Name,Value=23796349-vm
```

[6] Get the public IP address

- To get the public IP address of the instance, use the following command with the instance ID obtained from [Step 4](#):

```
aws ec2 describe-instances --instance-ids i-073e156e4154b59db --query
'Reservations[0].Instances[0].PublicIpAddress'
```

- Screenshots below show [Steps 4,5,6](#):

adharsh@adharsh:~\$ aws ec2 run-instances --image-id ami-056a29f2eddc40520 --security-group-ids 23796349-sg --count 1 --instance-type t2.micro --key-name 23796349-key --query 'Instances[0].InstanceId' "i-073e156e4154b59db"
adharsh@adharsh:~\$ aws ec2 create-tags --resources i-073e156e4154b59db --tags KeyName,Value=23796349-vm
adharsh@adharsh:~\$ aws ec2 describe-instances --instance-ids i-073e156e4154b59db --query 'Reservations[0].Instances[0].PublicIpAddress'
\$ '54.180.241.75'

EC2 > Instances > i-073e156e4154b59db

Instance summary for i-073e156e4154b59db (23796349-vm) Info

Updated less than a minute ago

Instance ID i-073e156e4154b59db (23796349-vm)	Public IPv4 address 54.180.241.75 [open address]	Private IPv4 addresses 172.31.5.159
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-54-180-241-75.ap-northeast-2.compute.amazonaws.com open address
Hostname type IP name: ip-172-31-5-159.ap-northeast-2.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-5-159.ap-northeast-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 54.180.241.75 [Public IP]	VPC ID vpc-01f84220d00070f97	Subnet ID subnet-09f996d1ad81767a9
IAM Role -	Instance ARN arn:aws:ec2:ap-northeast-2:489389878001:instance/i-073e156e4154b59db	Auto Scaling Group name -
IMDSv2 Optional EC2 recommends setting IMDSv2 to required Learn more		

Details Status and alarms Monitoring Security Networking Storage Tags

Instance details Info

Platform Ubuntu (Inferred)	AMI ID ami-056a29f2eddc40520	Monitoring disabled
Platform details Linux/UNIX	AMI name ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20240701	Termination protection Disabled

[7] Connect to the instance via ssh

- To connect to the instance from the terminal, use the following command with the public IP obtained from the previous step:

```
ssh -i 23796349-key.pem ubuntu@54.180.241.75
```

Activities Terminal Aug 11 13:11

ubuntu@ip-172-31-5-159:~

```
adharsh@adharsh:~$ ssh -i 23796349-key.pem ubuntu@54.180.241.75
The authenticity of host '54.180.241.75 (54.180.241.75)' can't be established.
ED25519 key fingerprint is SHA256:c9gRpVf41TyScIjr7gedFZBnQ5xZhWtscBE+4dpFm8A.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.180.241.75' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1022-aws x86_64)

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

System information as of Sun Aug 11 05:11:13 UTC 2024

System load: 0.12      Processes:          105
Usage of /: 20.7% of 7.57GB   Users logged in:     0
Memory usage: 21%        IPv4 address for eth0: 172.31.5.159
Swap usage:  0%

Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

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.

ubuntu@ip-172-31-5-159:~$
```

[8] List the created instance using the AWS console

Create an EC2 instance with Python Boto3

- To create an EC2 instance using python script, use the Boto3 library, refer the code below:

```
import boto3

# Start a session
ec2 = boto3.resource('ec2')
ec2_client = boto3.client('ec2')

# Naming the EC2 instance
name = input("Enter a name for your EC2 instance:")

#Check if security group exists else creating one
try:
    sg = ec2_client.describe_security_groups(GroupName=[f'{name}-sg'])
    security_group_id = sg['SecurityGroups'][0]['GroupId']
except:
    security_group = ec2.create_security_group(GroupName=f'{name}-sg',
                                                Description='Security group for development environment using pythonScript')
    security_group.authorize_ingress(
        IpProtocol='tcp',
        FromPort=22,
        ToPort=22,
        CidrIp='0.0.0.0/0'
    )
    security_group_id = security_group.id

#Check if key pair exists else creating one
try:
    kp = ec2_client.describe_key_pairs(KeyName=[f'{name}-key'])
except:
    key_pair = ec2.create_key_pair(KeyName=f'{name}-key')
    with open(f'{name}-key.pem', 'w') as file:
        file.write(key_pair.key_material)

# Create EC2 instance
instance = ec2.create_instances(
    ImageId='ami-056a29f2eddc40520',
    MinCount=1,
    MaxCount=1,
    InstanceType='t2.micro',
    KeyName=f'{name}-key',
    SecurityGroupIds=[security_group_id],
    TagSpecifications=[
        {
            'ResourceType': 'instance',
            'Tags': [
                {
                    'Key': 'Name',
                    'Value': name
                },
            ]
        }
    ]
)
```

```

        },
    ]
)[0]

# Wait and load instance details
instance.wait_until_running()
instance.load()

#Printing instance details
print(f'Instance ID: {instance.id}')
print(f'Security Group ID: {security_group_id}')
print(f'Public IP Address: {instance.public_ip_address}')

```

NOTE: Code referenced from [here](#).

```

adharsh@adharsh:~$ vi lab2_script.py
adharsh@adharsh:~$ python3 lab2_script.py
Enter a name for your EC2 Instance:23796349_pyscript
Instance ID: i-0dec21429e24fa6f0
Security Group ID: sg-09f35d0214895e774
Public IP Address: 13.124.252.122
adharsh@adharsh:~$ 

```

The screenshot shows the AWS CloudShell interface. On the left is a sidebar with navigation links like EC2 Dashboard, Instances, and Network & Security. The main area displays the 'Instance summary for i-0dec21429e24fa6f0 (23796349_pyscript)' page. Key details shown include:

- Instance ID: i-0dec21429e24fa6f0 (23796349_pyscript)
- Public IPv4 address: 13.124.252.122
- Instance state: Running
- Private IP DNS Name: ip-172-31-12-131.ap-northeast-2.compute.internal
- Instance type: t2.micro
- VPC ID: vpc-01fb42220d0070f97
- Subnet ID: subnet-09f996d1ad81767a9
- Instance ARN: arn:aws:ec2:ap-northeast-2:489389878001:instance/i-0dec21429e24fa6f0
- AMI ID: ami-056a29f2eddc40520
- Platform: Ubuntu (Inferred)
- AMI name: ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-20240701
- Monitoring: disabled
- Termination protection: Disabled

Use Docker inside a Linux OS

[1] Install Docker

- I have already set up the latest version of docker on my system.

```

Activities Terminal Aug 11 14:06
adharsh@adharsh:~$ docker --version
Docker version 27.1.1, build 6312585
adharsh@adharsh:~$ 

```

- This was done when I was enrolled in **CITS1003 - Introduction to Cybersecurity** and **CITS4407 - Open Source Tools and Scripting**.

[2] Build and run an httpd container

- Create a directory called **html** using the following command:

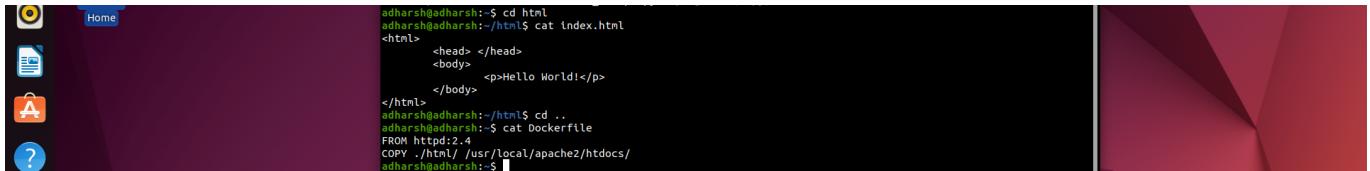
```
mkdir html
```

- Create a file **index.html** inside the html directory (use commands **vim** or **nano** or **touch** to create the file) and add the following content:

```
<html>
  <head> </head>
  <body>
    <p>Hello World!</p>
  </body>
</html>
```

- Create a file called **Dockerfile** outside the html directory with the following content:

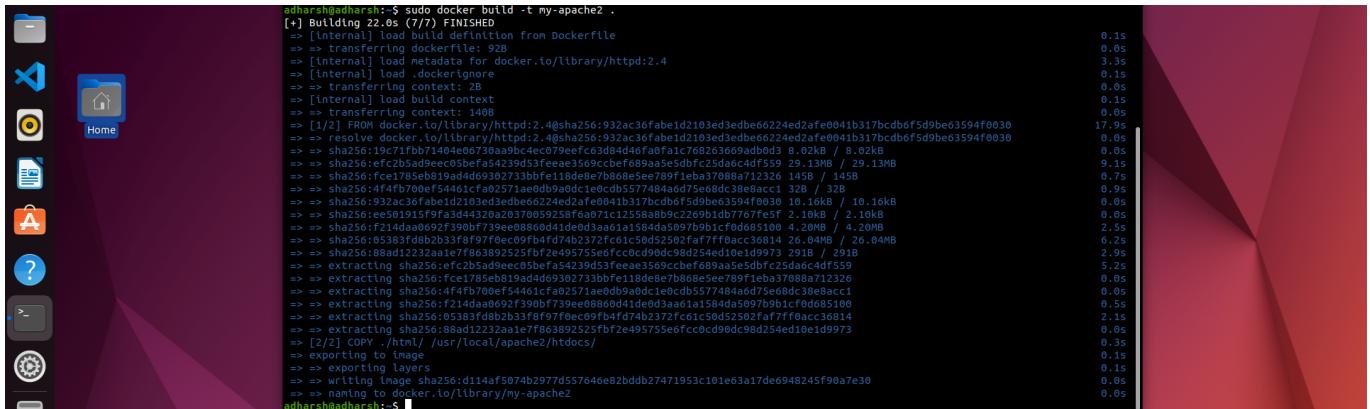
```
FROM httpd:2.4
COPY ./html/ /usr/local/apache2/htdocs/
```



```
adharsh@adharsh:~$ cd html
adharsh@adharsh:~/html$ cat index.html
<html>
  <head> </head>
  <body>
    <p>Hello World!</p>
  </body>
</html>
adharsh@adharsh:~/html$ cd ..
adharsh@adharsh:~$ cat Dockerfile
FROM httpd:2.4
COPY ./html/ /usr/local/apache2/htdocs/
adharsh@adharsh:~$
```

- Use the following command to build a docker image:

```
docker build -t my-apache2 .
```



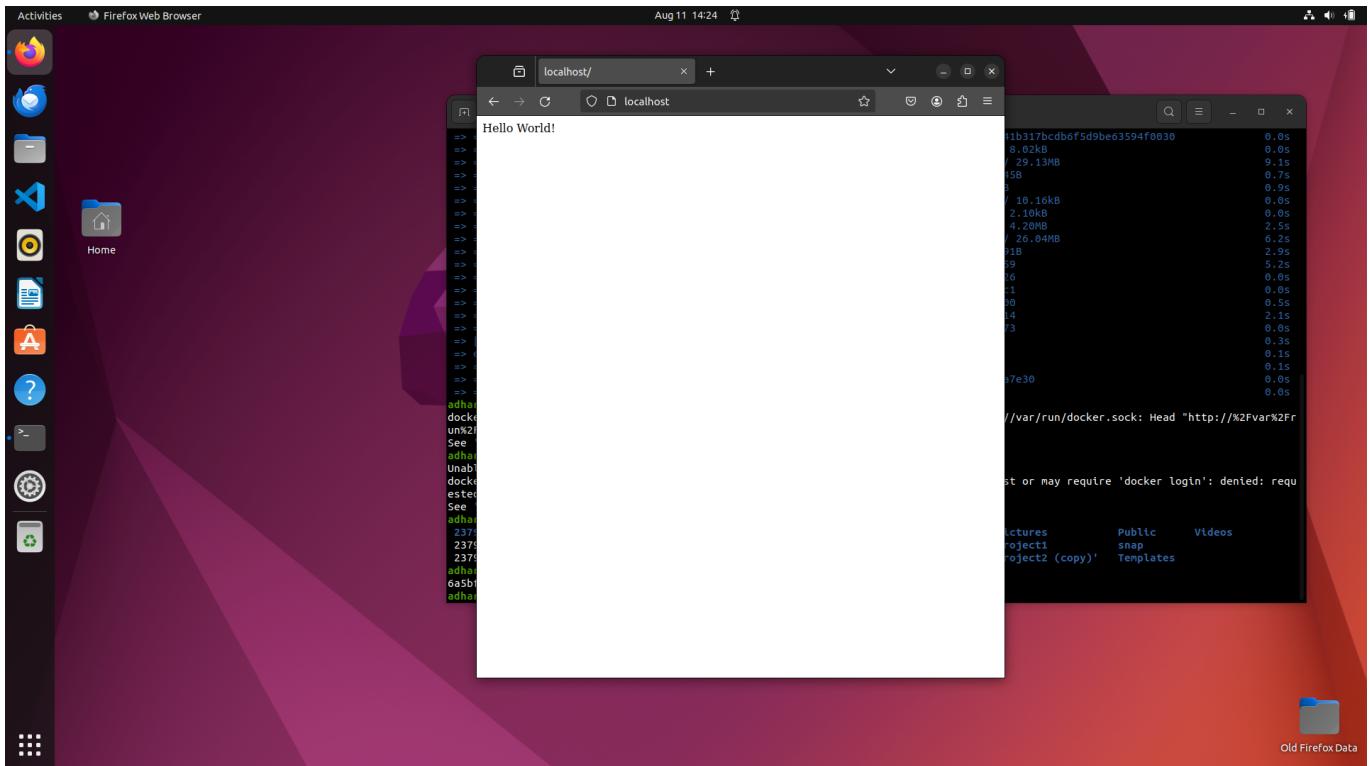
```
adharsh@adharsh:~$ sudo docker build -t my-apache2 .
[+] Building 22.0s (7/7) FINISHED
--> [internal] load build definition from Dockerfile
--> [internal] load metadata for docker.io/library/httpd:2.4
--> [internal] load .dockerignore
--> => transferring context: 2B
--> [internal] load build context
--> [internal] transferring context: 140B
--> [1/2] FROM docker.io/library/httpd:2.4@sha256:932ac36fabe1d2103ed3edb66224ed2afe0041b317bcd0f59be63594f0030
--> => sha256:19c71fb071484e06730aa9b9c4ec07eefc63d8d46fa0fa1c768263669adb0d3 8.02kB / 8.02kB
--> => sha256:efc2b58d9eec05be784239d3f7eeae3599c0bc0899a5e5dbfc25d86c40f539 29.13MB / 29.13MB
--> => sha256:0e01915f0ad344320a203178059258f6a071c12559aabb9c269b1d7767fe5f 17.95MB / 17.95MB
--> => sha256:c1f4fb05b7546cf4a2571a00d004dc1ec0cb57404ad75093780838eac1 1458 / 1458
--> => sha256:932ac36fabe1d2103ed3ed6724ed2afe0041b317bcd0f59be63594f0030 10.16kB / 10.16kB
--> => sha256:e01915f0ad344320a203178059258f6a071c12559aabb9c269b1d7767fe5f 2.10kB / 2.10kB
--> => sha256:f214daa06927390bf739ee08860d41de0d3aa1a1584da0997b9b1cf0d685100 4.20MB / 4.20MB
--> => sha256:05383fd0bb2b3f8f977fe09bf4bd74b237fc61c59d52502faf7fffa0acc36814 26.04MB / 26.04MB
--> => sha256:88ad12232aa1e7f863892525fb2e495755e6fcc0cd9ddc98d254ed10e1d9973 2918 / 2918
--> => extracting sha256:05383fd0bb2b3f8f977fe09bf4bd74b237fc61c59d52502faf7fffa0acc36814
--> => extracting sha256:efc2b58d9eec05be784239d3f7eeae3599c0bc0899a5e5dbfc25d86c40f539
--> => extracting sha256:ce1785eb019ad469303273bfe118deeerb0885ee789f1eba37088a712326
--> => extracting sha256:a4fb700ef54461cf0a02571ae0db9a0dc1eb0cb57484ad75e08d38eac1
--> => extracting sha256:214daa06927390bf739ee08860d41de0d3aa1a1584da0997b9b1cf0d685100
--> => extracting sha256:05383fd0bb2b3f8f977fe09bf4bd74b237fc61c59d52502faf7fffa0acc36814
--> => extracting sha256:088ad12232aa1e7f863892525fb2e495755e6fcc0cd9ddc98d254ed10e1d9973
--> [2/2] COPY ./html/ /usr/local/apache2/htdocs/
--> => exporting to image
--> => exporting layers
--> => writing image sha256:d114af074b2977d55746e8b2bdd27471953c101e6a17de6948245f90a7e30
--> => naming to docker.io/library/my-apache2
adharsh@adharsh:~$
```

- To run the docker image created before, use the following command:

```
docker run -p 80:80 -dit --name my-app my-apache2
```

```
adharsh@adharsh:~$ ls
23796349      23796349.zip  Dockerfile  html      microblog  Pictures   Public    Videos
23796349_key.pem  Desktop    Documents  lab1_script.py  Music     project1  snap     Templates
23796349_pyScript-key.pem  Downloads  lab2_script.py  osts      'project2 (copy)'  Templates
adharsh@adharsh:~$ sudo docker run -p 80:80 -dlt --name my-app my-apache2
6a5bf1ffff764b6db89910e6489f464377ed6b64452de418108e2f05de99f743c6
adharsh@adharsh:~$
```

and proper execution of the previous steps can be verified by opening a browser and accessing address:
<http://localhost> or <http://127.0.0.1>.



[3] Other docker commands

- To check the containers that are running, use the following command:

```
docker ps -a
```

```
adharsh@adharsh:~$ sudo docker ps -a
CONTAINER ID IMAGE NAMES COMMAND CREATED STATUS PORTS
6a5bf1ffff764 my-apache2 "httpd-foreground" About a minute ago Up About a minute 0.0.0.0:80->80/tcp
p ::::80::80/tcp my-apache2
bc36c4ac0b9c njw203/osts_2023:v1 "bash" 15 months ago Exited (129) 15 months ago
1f9dd62c81aa njw203/osts_2023:v1 "bash" 15 months ago Exited (130) 15 months ago
stolic_jones
d9e9aaafc0df njw203/osts_2023:v1 "bash" 16 months ago Exited (0) 16 months ago
silly_mayer
e8661f885a2b njw203/osts_2023:v1 "bash" 16 months ago Exited (255) 16 months ago
fervent_wilbur
ff51b83af1b2 njw203/osts_2023:v1 "bash" 16 months ago Exited (255) 16 months ago
stolic_cohen
5cca643e9d71 njw203/osts_2023:v1 "bash" 17 months ago Exited (129) 17 months ago
alito_waderson
7e29355b75ad njw203/osts_2023:v1 "bash" 17 months ago Exited (0) 17 months ago
heuristic_jemison
d5f639ce00d8 uwacyber/cits1003-labs: bash "/bin/bash" 17 months ago Exited (130) 17 months ago
boring_elgama
6bdd77e362aa uwacyber/cits1003-labs: bash "/bin/bash" 17 months ago Exited (0) 17 months ago
fervent_gates
adharsh@adharsh:~$
```

- To stop and remove any of the running containers, use the following command:

NOTE: Here the container name is **my-app**.

```
docker stop my-app
docker rm my-app
```



```
adharsh@adharsh:~$ docker stop my-app
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post "http://%2Fvar%2Frun%2Fdocker.sock/v1.46/containers/my-app/stop": dial unix /var/run/docker.sock: connect: permission denied
adharsh@adharsh:~$ sudo docker stop my-app
my-app
adharsh@adharsh:~$ sudo docker rm my-app
my-app
adharsh@adharsh:~$
```

- The ec2 instances that were created for this lab were deleted through the **AWS Console** after completion.
-

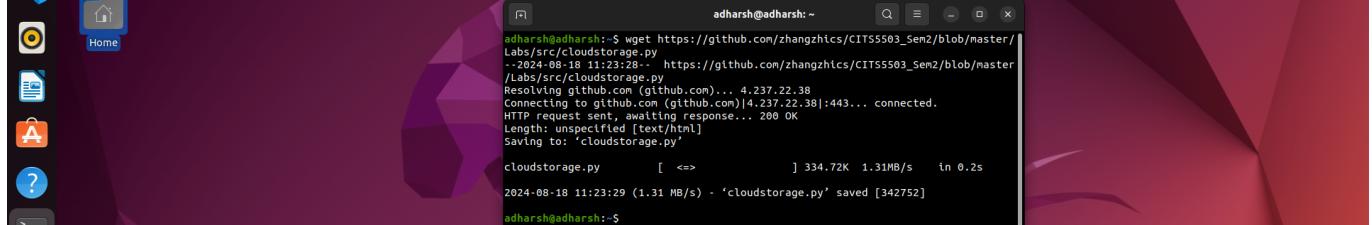
Lab 3

Program

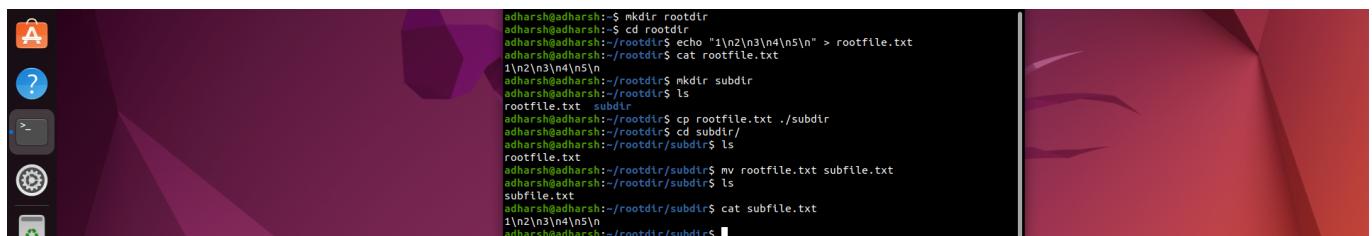
[1] Preparation

- To download the **cloudstorage.py** from the [src](#), use the command

```
wget  
https://github.com/zhangzhics/CITS5503_Sem2/blob/master/Labs/src/cloudstorage.py
```



- The necessary directories and files were created as shown in the screenshot below:



[2] Save to S3 by updating **cloudstorage.py**

- Modified code referenced from [here](#) :

```
import os  
import boto3  
import base64  
  
# -----  
# CITS5503  
#  
# cloudstorage.py  
#  
# skeleton application to copy local files to S3  
#  
# Given a root local directory, will return files in each level and  
# copy to same path on S3  
#  
# -----  
  
ROOT_DIR = '.'
```

```

ROOT_S3_DIR = '23796349-cloudstorage'

s3 = boto3.client("s3")

bucket_config = {'LocationConstraint': 'ap-northeast-2'} #Replace the region with
your allocated region name.

def upload_file(bucket,folder_name, file, file_name):
    try:
        path = f"{folder_name}{file_name}"
        print("Uploading %s" % file)
        s3.upload_file(file,bucket,path)
        print(f"Uploaded{file} into {bucket}/{path}")
    except Exception as error:
        pass

# Main program
# Insert code to create bucket if not there

try:
    s3.create_bucket(Bucket=ROOT_S3_DIR,CreateBucketConfiguration=bucket_config)
#create_bucket function is idempotent.
    response = s3.head_bucket(Bucket=ROOT_S3_DIR)
    print(f"Bucket '{ROOT_S3_DIR}' created and exists.")
    print(response)
except Exception as error:
    pass

# parse directory and upload files

for dir_name, subdir_list, file_list in os.walk(ROOT_DIR, topdown=True):
    if dir_name != ROOT_DIR:
        for fname in file_list:
            upload_file(ROOT_S3_DIR,"%s/" % dir_name[2:], "%s/%s" % (dir_name,
            fname), fname)

print("done")

```

The screenshot shows a terminal window on a Linux desktop. The terminal output is as follows:

```

adharsh@adharsh:~/ctts5503$ mkdir lab3
adharsh@adharsh:~/ctts5503$ cd ..
adharsh@adharsh:~$ mv cloudstorage.py ./ctts5503/lab3
adharsh@adharsh:~$ mv rootdir ./ctts5503/lab3
adharsh@adharsh:~$ cd ctts5503/
adharsh@adharsh:~/ctts5503$ cd lab3
adharsh@adharsh:~/ctts5503/lab3$ ls
cloudstorage.py rootdir
adharsh@adharsh:~/ctts5503/lab3$ python3 cloudstorage.py --i
Uploading ./rootdir/rootfile.txt
Uploaded ./rootdir/rootfile.txt into 23796349-cloudstorage/rootdir/rootfile.txt
Uploading ./rootdir/subdir/subfile.txt
Uploaded ./rootdir/subdir/subfile.txt into 23796349-cloudstorage/rootdir/subdir/subfile.txt
done
adharsh@adharsh:~/ctts5503/lab3$ 

```

The screenshot shows the AWS S3 console interface. The left sidebar includes options like Buckets, Access Grants, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, IAM Access Analyzer for S3, Storage Lens, Dashboards, Storage Lens groups, and AWS Organizations settings. The main content area displays the '23796349-cloudstorage' bucket. The 'Objects' tab is selected, showing one object named 'rootdir/'. A search bar at the top says 'Find objects by prefix'. Below it is a table with columns: Name, Type, Last modified, Size, and Storage class. The single entry is 'rootdir/' (Folder).

This screenshot shows the contents of the 'rootdir/' folder within the '23796349-cloudstorage' bucket. The 'Objects' tab is selected. It lists two objects: 'rootfile.txt' (Type: txt, Last modified: August 18, 2024, 12:36:17 (UTC+08:00), Size: 16.0 B, Storage class: Standard) and 'subdir/' (Type: Folder). A 'Copy S3 URI' button is visible on the right.

This screenshot shows the contents of the 'subdir/' folder within the 'rootdir/' folder. The 'Objects' tab is selected. It lists one object: 'subfile.txt' (Type: txt, Last modified: August 18, 2024, 12:36:17 (UTC+08:00), Size: 16.0 B, Storage class: Standard). A 'Copy S3 URI' button is visible on the right.

[3] Restore from S3

- Code to retrieve data from the same S3 bucket (Reference : [link](#)):

```
import os
import boto3

def download_file(bucket, s3_path, local_path):
```

```

try:
    s3.download_file(bucket, s3_path, local_path)
    print(f"Downloaded {s3_path} to {local_path}")
except Exception as e:
    print(f"An error occurred while downloading {s3_path}: {e}")

# Constants
ROOT_DIR = './rootdirRestored' # The directory where files will be restored
s3 = boto3.client("s3")
bucket = '23796349-cloudstorage'

# Create a "rootdir" for restored data
if not os.path.exists(ROOT_DIR):
    os.makedirs(ROOT_DIR)

# Fetching list of all objects from the bucket
try:
    response = s3.list_objects(Bucket=bucket)
except Exception as e:
    print(f"An error occurred: {e}")
    exit(1)

# Download each file to the appropriate directory
for item in response['Contents']:
    s3_path = item['Key']
    local_path = os.path.join(ROOT_DIR, s3_path)

    # Create directory if it doesn't exist
    local_dir = os.path.dirname(local_path)
    if not os.path.exists(local_dir):
        os.makedirs(local_dir)

    # Download the file
    download_file(bucket, s3_path, local_path)

print("Restore completed.")

```

```

adharsh@adharsh:~/cits5503/lab3$ touch restoreFromCloudStorage.py
adharsh@adharsh:~/cits5503/lab3$ ls
cloudstorage.py restoreFromCloudStorage.py rootdir
adharsh@adharsh:~/cits5503/lab3$ python3 restoreFromCloudStorage.py
Downloaded rootdir/rootfile.txt to ./rootdirRestored/rootdir/rootfile.txt
Downloaded rootdir/subdir/subfile.txt to ./rootdirRestored/rootdir/subdir/subfile.txt
Restore completed.

```

- Screenshot below shows the file structure of the restored data from s3:



[4] Write information about files to DynamoDB

- The required directory was created and the current directory was changed using the following commands:

```
mkdir dynamodb  
cd dynamodb
```

- The screenshot below shows that I have the updated version of jre already installed:

```
adharsh@adharsh:~/cits5503/lab3/dynamodb$ sudo apt-get install default-jre  
[sudo] password for adharsh:  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
default-jre is already the newest version (2:1.11-72build2).  
0 to upgrade, 0 to newly install, 0 to remove and 9 not to upgrade.
```

- The required file was downloaded and its contents were extracted using the following commands:

```
wget https://s3-ap-northeast-1.amazonaws.com/dynamodb-local-tokyo/dynamodb_local_latest.tar.gz  
tar -zvxf dynamodb_local_latest.tar.gz
```

```
adharsh@adharsh:~/cits5503/lab3/dynamodb$ wget https://s3-ap-northeast-1.amazonaws.com/dynamodb-local-tokyo/dynamodb_local_latest.tar.gz  
--2024-08-18 13:03:18 -- https://s3-ap-northeast-1.amazonaws.com/dynamodb-local-tokyo/dynamodb_local_latest.tar.gz  
Resolving s3-ap-northeast-1.amazonaws.com (s3-ap-northeast-1.amazonaws.com)... 52.219.68.194, 52.219.8.50, 52.219.162.244, ...  
Connecting to s3-ap-northeast-1.amazonaws.com (s3-ap-northeast-1.amazonaws.com)|52.219.68.194|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 54892500 (52M) [application/x-tar]  
Saving to: 'dynamodb_local_latest.tar.gz'  
  
dynamodb_local_latest.t 100%[=====]> 52.35M 6.56MB/s in 9.9s  
2024-08-18 13:03:29 (5.27 MB/s) - 'dynamodb_local_latest.tar.gz' saved [54892500/54892500]  
adharsh@adharsh:~/cits5503/lab3/dynamodb$ tar -zvxf dynamodb_local_latest.tar.gz  
DynamoDBLocal.jar  
DynamoDBLocalLib.jar  
DynamoDBLocalLib/apache-commons-HttpClient-4.5.x.jar
```

- Local instance of DynamoDB can be started using the following commands:

```
java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar -sharedDb
```

or

```
docker run -p 8000:8000 amazon/dynamodb-local -jar DynamoDBLocal.jar -inMemory -sharedDb
```

```
adharsh@adharsh:~/cits5503/lab3/dynamodb$ java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar -sharedDb  
Initializing DynamoDB Local with the following configuration:  
Port: 8000  
InMemory: false  
Version: 1.25.0  
DBPath: null  
SharedDB: true  
shouldDelayTransientStatuses: false  
CorsParams: null
```

NOTE: All the steps below were done on a new terminal, keeping the old one open.

- Python code to create a DynamoDB table named **CloudFiles** and write metadata of each file that was uploaded to s3 (References: 1, 2, 3):

```

import boto3
from datetime import datetime

# Initialize DynamoDB resource pointing to local instance
dynamodb = boto3.resource('dynamodb', endpoint_url='http://localhost:8000')
table_name = 'CloudFiles'

# Check if table already exists
try:
    table = dynamodb.create_table(
        TableName=table_name,
        KeySchema=[
            {
                'AttributeName': 'userId',
                'KeyType': 'HASH' # Primary key
            }
        ],
        AttributeDefinitions=[
            {
                'AttributeName': 'userId',
                'AttributeType': 'S' # String
            }
        ],
        ProvisionedThroughput={
            'ReadCapacityUnits': 1,
            'WriteCapacityUnits': 1
        }
    )
    print(f"Creating table {table_name}.")
    table.meta.client.get_waiter('table_exists').wait.TableName=table_name
    print(f"Table {table_name} created.")
except dynamodb.meta.client.exceptions.ResourceInUseException:
    # Table already exists, retrieve it
    table = dynamodb.Table(table_name)
    print(f"Table {table_name} already exists.")

# Initialize S3 client
s3 = boto3.client('s3')
bucket_name = '23796349-cloudstorage'

try:
    # Get list of all objects from the bucket
    objects = s3.list_objects_v2(Bucket=bucket_name)[ 'Contents' ]
    for obj in objects:
        file_name = obj[ 'Key' ]
        last_modified = obj[ 'LastModified' ].strftime("%Y-%m-%d %H:%M:%S")
        # Skip the folder itself
        if file_name.endswith('/'):
            continue
        # Get Object ACL
        acl = s3.get_object_acl(Bucket=bucket_name, Key=file_name)
        #print(acl)
        owner = acl[ 'Owner' ][ 'ID' ]
        permissions = [grant[ 'Permission' ] for grant in acl[ 'Grants' ]]
        # Insert metadata into DynamoDB

```

```

        table.put_item(
            Item={
                'userId': '23796349', #Since I am the user of this bucket
                'fileName': file_name,
                'path': f"s3://{bucket_name}/{file_name}",
                'lastUpdated': last_modified,
                'owner': owner,
                'permissions': ','.join(permissions)
            }
        )
        print(f"Added metadata for {file_name} to DynamoDB.")
except Exception as e:
    print(f"An error occurred: {e}")

```

- **NOTE:** I have used **Owner's ID**, as my region is not within the listed regions.

```

}
}
adharsh@adharsh:~/cits5503/lab3/dynamodb$ python3 dynamodb.py
Creating table CloudFiles.
Table CloudFiles created.
Added metadata for rootdir/rootfile.txt to DynamoDB.
Added metadata for rootdir/subdir/subfile.txt to DynamoDB.
dynamodb$ aws dynamodb scan --table-name CloudFiles --endpoint-url http://localhost:8000

```

[5] Scan the table

- The following command can be used to read the contents of the **CloudFiles** table:

```
aws dynamodb scan --table-name CloudFiles --endpoint-url http://localhost:8000
```

```

adharsh@adharsh:~/cits5503/lab3/dynamodb$ aws dynamodb scan --table-name CloudFiles --endpoint-url http://localhost:8000
{
    "Items": [
        {
            "owner": {
                "S": "2a5fac7aada1ad2caa48c9ab08cc4e2428d4eb596108daa3b59f1204ae96482e"
            },
            "path": {
                "S": "s3://23796349-cloudstorage/rootdir/subdir/subfile.txt"
            },
            "lastUpdated": {
                "S": "2024-08-18 04:36:17"
            },
            "fileName": {
                "S": "rootdir/subdir/subfile.txt"
            },
            "userId": {
                "S": "23796349"
            },
            "permissions": {
                "S": "FULL_CONTROL"
            }
        }
    ],
    "Count": 1,
    "ScannedCount": 1,
    "ConsumedCapacity": null
}
adharsh@adharsh:~/cits5503/lab3/dynamodb$ 

```

[6] Delete the table

- The following command can be used to delete the **CloudFiles** table:

```
aws dynamodb delete-table --table-name CloudFiles --endpoint-url
http://localhost:8000
```

```
adharsh@adharsh: ~/cits5503/lab3/dynamodb          adharsh@adharsh: ~/cits5503/lab3/dynamodb
adharsh@adharsh:~/cits5503/lab3/dynamodb$ aws dynamodb delete-table --table-name CloudFiles --endpoint-url http://localhost:8000
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "userId",
                "AttributeType": "S"
            }
        ],
        "TableName": "CloudFiles",
        "KeySchema": [
            {
                "AttributeName": "userId",
                "KeyType": "HASH"
            }
        ],
        "TableStatus": "ACTIVE",
        "CreationDateTime": 1723959331.08,
        "ProvisionedThroughput": {
            "LastIncreaseDateTime": 0.0,
            "LastDecreaseDateTime": 0.0,
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 1,
            "WriteCapacityUnits": 1
        },
        "TableSizeBytes": 227,
        "ItemCount": 1,
        "TableArn": "arn:aws:dynamodb:ddblocal:000000000000:table/cloudFiles",
        "DeletionProtectionEnabled": false
    }
}
adharsh@adharsh:~/cits5503/lab3/dynamodb$
```

- All the resources created for this lab were deleted via the AWS Console after completion.

Lab 4

Apply a policy to restrict permissions on bucket

[1] Write a Python script

- Before enforcing a policy, the appropriate resource needs to be created.
- Although the lab instructions ask us to use the S3 bucket created in **lab3**, the instructions in the same lab ask us to **delete** the resource after completion.
- Hence a new S3 bucket with the same name and objects was created using the AWS console.
- The following script applies the specified policy to the specified S3 bucket (Reference: [link](#)):

```
import boto3
import json

# Initialize a session using Amazon S3
s3 = boto3.client('s3')

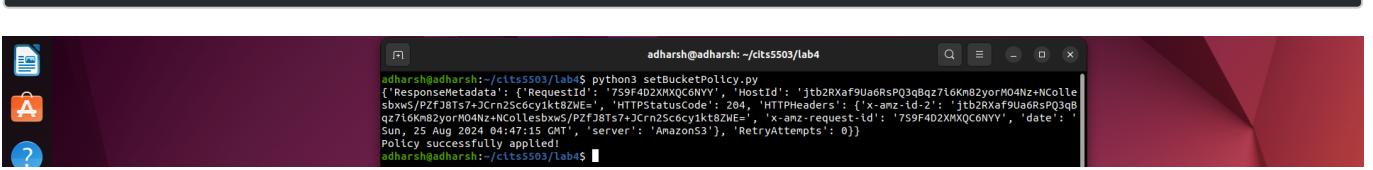
# Variables
bucket_name = '23796349-cloudstorage'
student_number = '23796349'

# Defining policy
policy = {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllS3ActionsInUserFolderForUserOnly",
            "Effect": "DENY",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": f"arn:aws:s3:::{bucket_name}/rootdir/subdir/*",
            "Condition": {
                "StringNotLike": {
                    "aws:username": f"{student_number}@student.uwa.edu.au"
                }
            }
        }
    ]
}

# Convert the policy to a JSON string
policy_string = json.dumps(policy)

# Apply the bucket policy
response = s3.put_bucket_policy(
    Bucket=bucket_name,
    Policy=policy_string
)

# Printing response
print(response)
print("Policy successfully applied!")
```

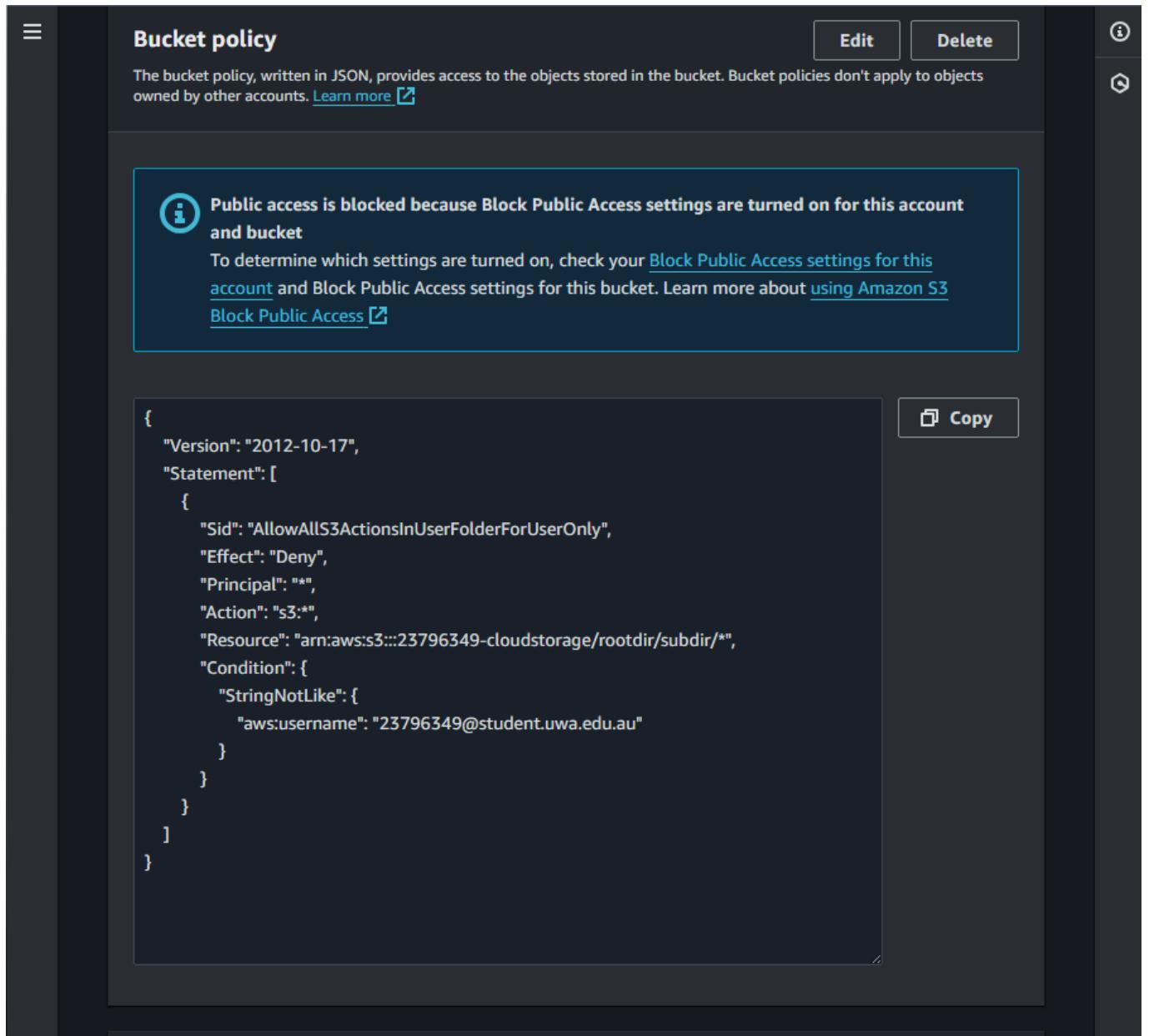


```
adharsh@adharsh:~/ctts5503/lab4$ python3 setBucketPolicy.py
{'ResponseMetadata': {'RequestId': '759F4D2XMXQG6NYY', 'HostId': 'jtb2Rxf9Ua6RsPQ3Bqz7l6Km82yrmO4Nz+NColle
sbwxs/PzfJ8t57+JcrnZsc6cyktBzWE', 'HTTPStatusCode': 204, 'HTTPHeaders': {'x-amz-id-2': 'jtb2Rxf9Ua6RsPQ3B
qz7l6Km82yrmO4Nz+NCollesbxws/PzfJ8t57+JcrnZsc6cyktBzWE', 'x-amz-request-id': '759F4D2XMXQG6NYY', 'date': 'Sun, 25 Aug 2024 04:47:15 GMT', 'server': 'AmazonS3'}, 'RetryAttempts': 0}}
Policy successfully applied!
```

[2] Check if the script works

- The screenshots below prove that the script works:

The policy on the console



Bucket policy

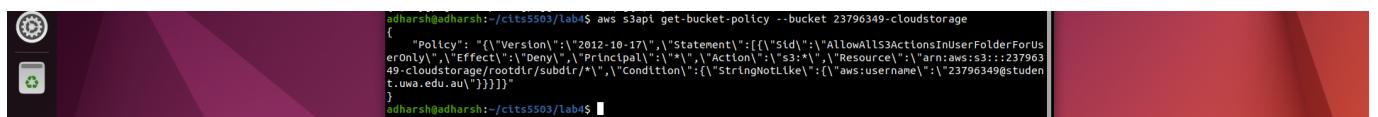
The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

Public access is blocked because Block Public Access settings are turned on for this account and bucket

To determine which settings are turned on, check your [Block Public Access settings for this account](#) and [Block Public Access settings for this bucket](#). Learn more about [using Amazon S3 Block Public Access](#)

```
{ "Version": "2012-10-17", "Statement": [ { "Sid": "AllowAllS3ActionsInUserFolderForUserOnly", "Effect": "Deny", "Principal": "*", "Action": "s3:*", "Resource": "arn:aws:s3:::23796349-cloudstorage/rootdir/subdir/*", "Condition": { "StringNotLike": { "aws:username": "23796349@student.uwa.edu.au" } } } ] }
```

The policy using aws cli

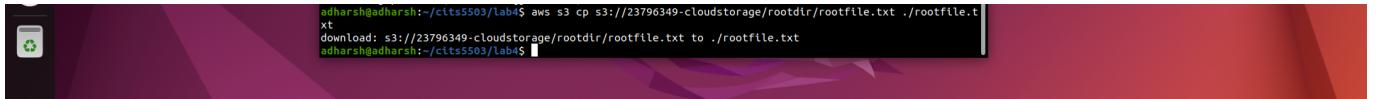


```
adharsh@adharsh:~/ctts5503/lab4$ aws s3api get-bucket-policy --bucket 23796349-cloudstorage
{
    "Policy": "{\"Version\":\"2012-10-17\"},\"Statement\":[{\"Sid\":\"AllowAllS3ActionsInUserFolderForUserOnly\", \"Effect\":\"Deny\", \"Principal\":\"*\", \"Action\":\"s3:*\", \"Resource\":\"arn:aws:s3:::23796349-cloudstorage/rootdir/subdir/*\", \"Condition\":{\"StringNotLike\":{\"aws:username\":\"23796349@student.uwa.edu.au\"}}}]"
}
adharsh@adharsh:~/ctts5503/lab4$
```

Trying to download the rootfile.txt using an username without access

```
adharsh@adharsh:~/cits5503/lab4$ aws s3 cp s3://23796349-cloudstorage/rootdir/rootfile.txt ./rootfile.txt --profile adharsh136@gmail.com
fatal error: An error occurred (403) when calling the HeadObject operation: Forbidden
```

Trying to download the rootfile.txt using the username with access



```
adharsh@adharsh:~/cits5503/lab4$ aws s3 cp s3://23796349-cloudstorage/rootdir/rootfile.txt ./rootfile.txt
download: s3://23796349-cloudstorage/rootdir/rootfile.txt to ./rootfile.txt
adharsh@adharsh:~/cits5503/lab4$
```

AES Encryption using KMS

[1] Create a KMS key

- The code below creates a KMS key with my student number as its alias (Reference: [link](#)):

```
import boto3
import json

# Initialize a session using Amazon KMS
session = boto3.Session(region_name='ap-northeast-2')
kms_client = session.client('kms')

# Create the KMS key
try:
    response = kms_client.create_key(
        Description='KMS key for lab4',
        KeyUsage='ENCRYPT_DECRYPT',
        Origin='AWS_KMS',
        BypassPolicyLockoutSafetyCheck=False
    )
    key_id = response[ 'KeyMetadata'][ 'KeyId']
    print(f"Created KMS key with id: {key_id}")
except Exception as e:
    print(f"Failed to create key: {e}")

key_alias = "alias/23796349_KMS"
# Add the alias
try:
    kms_client.create_alias(
        AliasName=key_alias,
        TargetKeyId=key_id
    )
    print(f"Successfully attached alias {key_alias} to KMS key.")
except kms_client.exceptions.AlreadyExistsException:
    print(f"Alias {key_alias} already exists. Deleting existing alias.")

# Delete the existing alias
try:
    kms_client.delete_alias(
        AliasName=key_alias
    )
    print(f"Successfully deleted existing alias {key_alias}.")
```

```

# Create the new alias
    kms_client.create_alias(
        AliasName=key_alias,
        TargetKeyId=key_id
    )

    print(f"Successfully attached new alias {key_alias} to KMS key.")
except Exception as e:
    print(f"Failed to replace alias: {e}")
except Exception as e:
    print(f"Failed to attach alias: {e}")

```

[2] Attach a policy to the created KMS key

- The code in the previous step is updated to define a KMS key policy and attach the same to the KMS key (Reference: [link](#)):

```

import boto3
import json

# Initialize a session using Amazon KMS
session = boto3.Session(region_name='ap-northeast-2')
kms_client = session.client('kms')

# Create the KMS key
try:
    response = kms_client.create_key(
        Description='KMS key for lab4',
        KeyUsage='ENCRYPT_DECRYPT',
        Origin='AWS_KMS',
        BypassPolicyLockoutSafetyCheck=False
    )
    key_id = response['KeyMetadata']['KeyId']
    print(f"Created KMS key with id: {key_id}")
except Exception as e:
    print(f"Failed to create key: {e}")

key_alias = "alias/23796349_KMS"
# Add the alias
try:
    kms_client.create_alias(
        AliasName=key_alias,
        TargetKeyId=key_id
    )
    print(f"Successfully attached alias {key_alias} to KMS key.")
except kms_client.exceptions.AlreadyExistsException:
    print(f"Alias {key_alias} already exists. Deleting existing alias.")

# Delete the existing alias
try:

```

```

kms_client.delete_alias(
    AliasName=key_alias
)
print(f"Successfully deleted existing alias {key_alias}.")

# Create the new alias
kms_client.create_alias(
    AliasName=key_alias,
    TargetKeyId=key_id
)

print(f"Successfully attached new alias {key_alias} to KMS key.")
except Exception as e:
    print(f"Failed to replace alias: {e}")
except Exception as e:
    print(f"Failed to attach alias: {e}")

#-----UPDATED SECTION-----
#Defining key policy

key_policy={

    "Version": "2012-10-17",
    "Id": "key-consolepolicy-3",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::489389878001:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allow access for Key Administrators",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::489389878001:user/23796349@student.uwa.edu.au"
            },
            "Action": [
                "kms>Create*",
                "kms>Describe*",
                "kms>Enable*",
                "kms>List*",
                "kms>Put*",
                "kms>Update*",
                "kms>Revoke*",
                "kms>Disable*",
                "kms>Get*",
                "kms>Delete*",
                "kms>TagResource",
                "kms>UntagResource",
                "kms>ScheduleKeyDeletion",
                "kms>CancelKeyDeletion"
            ],
        }
    ]
}

```

```

    "Resource": "*"
},
{
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::489389878001:user/23796349@student.uwa.edu.au"
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
},
{
    "Sid": "Allow attachment of persistent resources",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::489389878001:user/23796349@student.uwa.edu.au"
    },
    "Action": [
        "kms>CreateGrant",
        "kms>ListGrants",
        "kms:RevokeGrant"
    ],
    "Resource": "*",
    "Condition": {
        "Bool": {
            "kms:GrantIsForAWSResource": "true"
        }
    }
}
]
}
# Put key policy
try:
    kms_client.put_key_policy(
        KeyId=key_id,
        PolicyName='default',
        Policy=json.dumps(key_policy, indent=4),
        BypassPolicyLockoutSafetyCheck=False
    )
    print(f"The key ID is: {key_id}")
    print("Successfully attached the policy to the KMS key.")
except Exception as e:
    print(f"Failed to attach policy: {e}")

```

[3] Check whether the script works

- Screenshots below show the policy attached to the KMS key seen in the AWS Console:

adharsh@adharsh:~/ctts5503/lab4\$ touch createKMS.py
adharsh@adharsh:~/ctts5503/lab4\$ python3 createKMS.py
Created KMS key with id: 98105fc3-8aa8-46b6-a305-07492577b81e
Successfully attached alias alias/23796349_KMS to KMS key.
The key ID is: 98105fc3-8aa8-46b6-a305-07492577b81e
Successfully attached the policy to the KMS key.
adharsh@adharsh:~/ctts5503/lab4\$

```

{
    "Version": "2012-10-17",
    "Id": "key-consolepolicy-3",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::489389878001:root"
            }
        }
    ]
}

```

CloudShell Feedback Privacy Terms Cookie preferences

NOTE: Here for both the key administrator and the key user, the ARN is the same, `arn:aws:iam::489389878001:user/23796349@student.uwa.edu.au`

[4] Use the created KMS key for encryption/decryption

- The code below uses the created KMS key to encrypt/decrypt every file in the bucket (Reference: 1, 2)
- I have used argument parser to have the choice between encrypting and decrypting files (options `-e` to encrypt and `-d` to decrypt files).

```

import os
import boto3
import argparse
from botocore.exceptions import NoCredentialsError

```

```

# Initialize KMS client
kms = boto3.client('kms')

# Function to encrypt data using KMS key
def encrypt_file(file_path, kms_key_id):
    with open(file_path, 'rb') as f:
        plaintext = f.read()
    response = kms.encrypt(KeyId=kms_key_id, Plaintext=plaintext)
    ciphertext = response['CiphertextBlob']
    return ciphertext

# Function to decrypt data using KMS key
def decrypt_file(ciphertext, kms_key_id):
    response = kms.decrypt(KeyId=kms_key_id, CiphertextBlob=ciphertext)
    plaintext = response['Plaintext']
    return plaintext

# Argument Parsing
parser = argparse.ArgumentParser(description='KMS_Encrypt_Decrypt')
parser.add_argument('--encrypt', '-e', help='KMS key ID for encrypting files')
parser.add_argument('--decrypt', '-d', help='KMS key ID for decrypting files')
args = parser.parse_args()

# Constants and Initialization
ROOT_DIR = './rootdir'
DECRYPT_DIR = './rootdir_de'
s3 = boto3.client("s3")
bucket_name = '23796349-cloudstorage'

# Function to upload files
def upload_file(bucket_name, folder_name, file_data, file_name):
    s3_path = f"{folder_name}/enc_{file_name}"
    s3.put_object(Body=file_data, Bucket=bucket_name, Key=s3_path)
    print(f"Uploaded encrypted {file_name} to {bucket_name}/{s3_path}")

# Function to download and decrypt files
def download_and_decrypt_file(bucket_name, s3_path, local_path, kms_key_id):
    response = s3.get_object(Bucket=bucket_name, Key=s3_path)
    encrypted_data = response['Body'].read()
    decrypted_data = decrypt_file(encrypted_data, kms_key_id)
    with open(local_path, 'wb') as f:
        f.write(decrypted_data)
    print(f"Downloaded and decrypted {s3_path} to {local_path}")

# Walk through the directory and upload files
if args.encrypt:
    for dir_name, _, file_list in os.walk(ROOT_DIR):
        relative_folder = os.path.relpath(dir_name, ROOT_DIR)
        if relative_folder == '.':
            relative_folder = 'rootdir'
        else:
            relative_folder = 'rootdir/' + relative_folder
        for fname in file_list:
            full_path = os.path.join(dir_name, fname)
            encrypted_data = encrypt_file(full_path, args.encrypt)
            upload_file(bucket_name, relative_folder, encrypted_data, fname)

```

```
# Download and decrypt files
if args.decrypt:
    os.makedirs(DECRYPT_DIR, exist_ok=True)
    s3_resource = boto3.resource('s3')
    my_bucket = s3_resource.Bucket(bucket_name)
    for obj in my_bucket.objects.all():
        s3_path = obj.key
        local_path = os.path.join(DECRYPT_DIR, '/'.join(s3_path.split('/')[1:]))
        local_dir = os.path.dirname(local_path)
        os.makedirs(local_dir, exist_ok=True)
        print(s3_path)
        if 'enc_' in s3_path:
            download_and_decrypt_file(bucket_name, s3_path,
local_path.replace('enc_','dec_'), args.decrypt)
print("Done")
```

- Screenshots below show the encryption and decryption operations:

Encrypt

```
adharsh@adharsh:~/cits5503/lab4$ python3 encryptDecryptKMS.py --encrypt 98105fc3  
-8aa8-46b6-a305-07492577b81e  
Uploaded encrypted rootfile.txt to 23796349-cloudstorage/rootdir/enc_rootfile.tx  
t  
Uploaded encrypted subfile.txt to 23796349-cloudstorage/rootdir/subdir/enc_subfi  
le.txt  
Done
```

The screenshot shows the AWS S3 console interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, search icon, filter icon, refresh icon, help icon, and a dropdown for 'Seoul'. To the right, it shows the email '23796349@student.uwa.edu.au @ 4893-8987-8001' and a 'Copy S3 URI' button.

The main area displays the 'rootdir/' folder under the '23796349-cloudstorage' bucket. The 'Objects' tab is selected. On the left, there are buttons for 'Actions' (dropdown), 'Create folder', and 'Upload' (highlighted in orange). Below these are buttons for 'Copy S3 URI', 'Copy URL', 'Download', 'Open', and 'Delete'.

A search bar at the top of the object list contains the placeholder 'Find objects by prefix'. To its right are navigation icons for back, forward, and a refresh symbol.

The object list table has columns: Name, Type, Last modified, Size, and Storage class. The table shows three objects:

Name	Type	Last modified	Size	Storage class
enc_rootfile.txt	txt	August 25, 2024, 18:44:04 (UTC+08:00)	168.0 B	Standard
rootfile.txt	txt	August 25, 2024, 12:31:22 (UTC+08:00)	16.0 B	Standard
subdir/	Folder	-	-	-

At the bottom of the page, there are links for 'CloudShell', 'Feedback', 'Privacy', 'Terms', and 'Cookie preferences'.

Amazon S3 > Buckets > 23796349-cloudstorage > rootdir/ > subdir/

subdir/

Objects **Properties**

Objects (2) Info

Actions **Create folder** **Upload**

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	enc_subfile.txt	txt	August 25, 2024, 18:44:05 (UTC+08:00)	168.0 B	Standard
<input type="checkbox"/>	subfile.txt	txt	August 25, 2024, 12:31:24 (UTC+08:00)	16.0 B	Standard

CloudShell Feedback Privacy Terms Cookie preferences

Decrypt

```

A adharsh@adharsh:~/cits5503/lab4$ python3 encryptDecryptKMS.py --decrypt 98105fc3-8aa8-46b6-a305-07492577b81e
rootdir/enc_rootfile.txt
Downloaded and decrypted rootdir/enc_rootfile.txt to ./rootdir_de/dec_rootfile.txt
rootdir/rootfile.txt
rootdir/subdir/enc_subfile.txt
Downloaded and decrypted rootdir/subdir/enc_subfile.txt to ./rootdir_de/subdir/dec_subfile.txt
rootdir/subdir/subfile.txt
Done
adharsh@adharsh:~/cits5503/lab4$ cd rootdir_de
adharsh@adharsh:~/cits5503/lab4/rootdir_de$ tree
.
+- dec_rootfile.txt
  +- subdir
    +- dec_subfile.txt
1 directory, 2 files

```

[5] Apply `pycryptodome` for encryption/decryption

- The code below uses the `pycryptodome` library to encrypt/decrypt every file in the bucket (Adopted from: [link](#)).

- Again, I have used argument parser to choice between encryption and decryption of files (options **-e** to encrypt and **-d** to decrypt files).
- Additionally, as the encryption/decryption function requires a passphrase, option **-p** was introduced to facilitate this.

```

import os, struct, hashlib, argparse
import boto3
from botocore.exceptions import NoCredentialsError
from Crypto.Cipher import AES
from Crypto import Random

CHUNK_SIZE = 64 * 1024

# Argument Parsing
parser = argparse.ArgumentParser(description='Encrypt_decrypt_pycryptodome')
parser.add_argument('--encrypt', '-e', action='store_true', help='Encrypt files')
parser.add_argument('--decrypt', '-d', action='store_true', help='Decrypt files')
parser.add_argument('--passphrase', '-p', required=True, help='Passphrase for encryption/decryption')
args = parser.parse_args()

# Constants and Initialization
ROOT_DIR = './rootdir'
DECRYPT_DIR = './rootdir_de2'
s3 = boto3.client("s3")
bucket_name = '23796349-cloudstorage'

# Function to encrypt data using passphrase
def encrypt_file(file_path, passphrase):
    key = hashlib.sha256(passphrase.encode("utf-8")).digest()
    iv = Random.new().read(AES.block_size)
    encryptor = AES.new(key, AES.MODE_CBC, iv)
    filesize = os.path.getsize(file_path)
    with open(file_path, 'rb') as infile:
        encrypted_data = struct.pack('<Q', filesize) + iv
        while True:
            chunk = infile.read(CHUNK_SIZE)
            if len(chunk) == 0:
                break
            elif len(chunk) % 16 != 0:
                chunk += ' '.encode("utf-8") * (16 - len(chunk) % 16)
            encrypted_data += encryptor.encrypt(chunk)
    return encrypted_data

# Function to decrypt data using passphrase
def decrypt_file(encrypted_data, passphrase):
    key = hashlib.sha256(passphrase.encode("utf-8")).digest()
    origsize = struct.unpack('<Q', encrypted_data[:8])[0]
    iv = encrypted_data[8:24]
    decryptor = AES.new(key, AES.MODE_CBC, iv)
    ciphertext = encrypted_data[24:]

    plaintext = b''

```

```

    for i in range(0, len(ciphertext), CHUNK_SIZE):
        chunk = ciphertext[i:i + CHUNK_SIZE]
        plaintext += decryptor.decrypt(chunk)
    return plaintext[:origsize]

# Function to upload files
def upload_file(bucket_name, folder_name, file_data, file_name):
    s3_path = f"{folder_name}/enc1_{file_name}"
    s3.put_object(Body=file_data, Bucket=bucket_name, Key=s3_path)
    print(f"Uploaded encrypted {file_name} to {bucket_name}/{s3_path}")

# Function to download and decrypt files
def download_and_decrypt_file(bucket_name, s3_path, local_path, passphrase):
    response = s3.get_object(Bucket=bucket_name, Key=s3_path)
    encrypted_data = response['Body'].read()
    decrypted_data = decrypt_file(encrypted_data, passphrase)
    with open(local_path, 'wb') as f:
        f.write(decrypted_data)
    print(f"Downloaded and decrypted {s3_path} to {local_path}")

# Walk through the directory and upload files
if args.encrypt:
    for dir_name, _, file_list in os.walk(ROOT_DIR):
        relative_folder = os.path.relpath(dir_name, ROOT_DIR)
        if relative_folder == '.':
            relative_folder = 'rootdir'
        else:
            relative_folder = 'rootdir/' + relative_folder
        for fname in file_list:
            full_path = os.path.join(dir_name, fname)
            encrypted_data = encrypt_file(full_path, args.passphrase)
            upload_file(bucket_name, relative_folder, encrypted_data, fname)

# Download and decrypt files
if args.decrypt:
    os.makedirs(DECRYPT_DIR, exist_ok=True)
    s3_resource = boto3.resource('s3')
    my_bucket = s3_resource.Bucket(bucket_name)
    for obj in my_bucket.objects.all():
        s3_path = obj.key
        local_path = os.path.join(DECRYPT_DIR, '/'.join(s3_path.split('/')[1:]))
        local_dir = os.path.dirname(local_path)
        os.makedirs(local_dir, exist_ok=True)
        if 'enc1_' in s3_path:
            download_and_decrypt_file(bucket_name, s3_path,
                                      local_path.replace('enc1_', 'dec1_'), args.passphrase)
    print("Done")

```

- Screenshots below show the encryption and decryption operations:

Encrypt

adharsh@adharsh:~/cts5503/lab4\$ touch encryptDecryptLib.py
Command 'touch' not found, did you mean:
 command 'touch' from deb coreutils (8.32-4.1ubuntu1.2)
Try: sudo apt install <deb name>
adharsh@adharsh:~/cts5503/lab4\$ python3 encryptDecryptLib.py --encrypt --passphrase cloud
Uploaded rootfile.txt to 23796349-cloudstorage/rootdir/enc1_rootfile.txt
Uploaded encrypted subfile.txt to 23796349-cloudstorage/rootdir/subdir/enc1_subfile.txt
Done

aws Services Search Bucket Notifications Help Seoul 23796349@student.uwa.edu.au @ 4893-8987-8001 ▾

Amazon S3 > Buckets > 23796349-cloudstorage > rootdir/

rootdir/ Copy S3 URI

Objects Properties

Objects (4) Info

C Copy S3 URI Copy URL Download Open Delete

Actions Create folder Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified
<input type="checkbox"/>	enc_rootfile.txt	txt	Aug 10
<input type="checkbox"/>	enc1_rootfile.txt	txt	Aug 10
<input type="checkbox"/>	rootfile.txt	txt	Aug 10
<input type="checkbox"/>	subdir/	Folder	-

CloudShell Feedback Privacy Terms Cookie preferences

Amazon S3 > Buckets > 23796349-cloudstorage > rootdir/ > subdir/

subdir/

Objects Properties

Objects (3) Info

C Copy S3 URI Copy URL Download Open Delete

Actions Create folder Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

Name	Type	Last Modified
enc_subfile.txt	txt	Aug 10
enc1_subfile.txt	txt	Aug 10
subfile.txt	txt	Aug 10

CloudShell Feedback Privacy Terms Cookie preferences

Decrypt

```
adharsh@adharsh:~/cits5503/lab4$ python3 encryptDecryptLib.py --decrypt --passphrase cloud
Downloaded and decrypted rootdir/enc1_rootfile.txt to ./rootdir_de2/dec1_rootfile.txt
Downloaded and decrypted rootdir/subdir/enc1_subfile.txt to ./rootdir_de2/subdir/dec1_subfile.txt
Done
adharsh@adharsh:~/cits5503/lab4$ cd rootdir_de2
adharsh@adharsh:~/cits5503/lab4/rootdir_de2$ tree
.
└── dec1_rootfile.txt
    └── subdir
        └── dec1_subfile.txt
1 directory, 2 files
adharsh@adharsh:~/cits5503/lab4/rootdir_de2$
```

Answer the following question

What is the performance difference between using KMS and using the custom solution?

AWS KMS:

- Since KMS is a managed service by AWS, it's easier to implement and scale to all resources.
- The Encryption/Decryption operations are offloaded to KMS, so overhead on the user's or application's resources is reduced.
- Additionally, features such as key rotation, key versioning, etc. are available.

Custom Solution:

- Custom solution are important if there is specific or unique encryption/decryption need.
- The performance can vary depending on various factors such as algorithm, code quality, implementation technique, etc and there's a greater control over the cryptographic operations.
- There could be a performance hit if the encryption/decryption operation is performed directly by the user's or application's resources.

At the end of the day, its over ease of use, cost-effectiveness and seamless integration or greater control, complexity and specific need.

All resources that were created for this lab were deleted using AWS Console after completion.

Lab 5

Application Load Balancer

[1] Create 2 EC2 instances

- Initially, I had to find the **VPC ID** of my allocated region and also the respective **subnet IDs**.
- The following AWS CLI command was used to find the VPC ID:

```
aws ec2 describe-vpcs --filters "Name=isDefault,Values=true"
```

- To get the subnets IDs specific to a VPC, the following AWS CLI command:

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=vpca-01f842220d0070f97"
```

NOTE: I have used the VPC ID in my allocated region in the **Values** field.

- The code below creates **two EC2** instances and a security group that authorise inbound traffic for **HTTP** and **SSH** (Reference: [Lab 2](#)).

NOTE: The **key-pair** created in [Lab 2](#) was used here. Also, I chose availability zones A and C since I am creating instances of the type **t2.micro**.

```
import boto3
import time

# Constants

#Using the same key-pair created for Lab2
KEY_NAME = '23796349-key'

#The AMI ID for my region
AMI_ID = 'ami-056a29f2eddc40520'

#The default subnets a and c IDs respectively
SUBNETS = [ 'subnet-09f996d1ad81767a9', 'subnet-0b3601189181ee48e' ]

#The default VPC ID
VPC_ID = 'vpc-01f842220d0070f97'

#The two availability zones in the region
AVAILABILITY_ZONES = [ 'ap-northeast-2a', 'ap-northeast-2c' ]

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

# Create a security group and 2 EC2 instances in two different availability zones
```

```

try:
    sg = ec2.describe_security_groups(GroupNames='23796349-sg')
    sec_grp_id = sg['SecurityGroups'][0]['GroupId']
except:
    security_group = ec2.create_security_group(GroupName='23796349-sg',
Description='Security group for ssh and http inbound traffic')
    security_group.authorize_ingress(
        IpPermissions=[
            {
                'IpProtocol': 'tcp',
                'FromPort': 22,
                'ToPort': 22,
                'IpRanges':[{'CidrIp': '0.0.0.0/0'}]
            },
            {
                'IpProtocol': 'tcp',
                'FromPort': 80,
                'ToPort': 80,
                'IpRanges':[{'CidrIp': '0.0.0.0/0'}]
            }
        ]
    )
    sec_grp_id = security_group.id

instances = []
for i, zone in enumerate(AVAILABILITY_ZONES):
    try:
        instance = ec2.create_instances(
            ImageId=AMI_ID,
            MinCount=1,
            MaxCount=1,
            KeyName=KEY_NAME,
            InstanceType='t2.micro',
            SecurityGroupIds=[sec_grp_id],
            Placement={
                'AvailabilityZone': zone
            },
            SubnetId=SUBNETS[i],
            TagSpecifications=[
                {
                    'ResourceType': 'instance',
                    'Tags': [
                        {
                            'Key': 'Name',
                            'Value': '23796349-vm' + str(i+1)
                        },
                    ]
                },
            ],
        )[0]
        instances.append(instance)
    except Exception as e:
        print(f"Error creating instance in {zone}: {e}")

# Wait for instances to be in running state

```

```

waiter = ec2.meta.client.get_waiter('instance_running')
waiter.wait(InstanceIds=[instance.id for instance in instances])

# Fetch and print the public IP addresses of the instances
for instance in instances:
    instance.reload() # Refresh instance details
    print(f"Instance {instance.id} Public IP: {instance.public_ip_address}")

```

[2] Create an Application Load Balancer

- The code above was updated to create an application load balancer and target group with the two instances previously created as the targets and finally a listener. (Reference: [link](#))
- The implementation uses the ELB V2 interface.

```

import boto3
import time

# Constants

#Using the same key-pair created for Lab2
KEY_NAME = '23796349-key'

#The AMI ID for my region
AMI_ID = 'ami-056a29f2eddc40520'

#The default subnets a and c IDs respectively
SUBNETS = [ 'subnet-09f996d1ad81767a9', 'subnet-0b3601189181ee48e' ]

#The default VPC ID
VPC_ID = 'vpc-01f842220d0070f97'

#The two availability zones in the region
AVAILABILITY_ZONES = [ 'ap-northeast-2a', 'ap-northeast-2c' ]

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

# Create a security group and 2 EC2 instances in two different availability zones
try:
    sg = ec2.describe_security_groups(GroupNames='23796349-sg')
    sec_grp_id = sg['SecurityGroups'][0]['GroupId']
except:
    security_group = ec2.create_security_group(GroupName='23796349-sg',
                                                Description='Security group for ssh and http inbound traffic')
    security_group.authorize_ingress(
        IpPermissions=[
            {
                'IpProtocol': 'tcp',
                'FromPort': 22,
                'ToPort': 22,
                'IpRanges':[{'CidrIp': '0.0.0.0/0'}]
            }
        ]
    )

```

```

        },
        {
            'IpProtocol': 'tcp',
            'FromPort': 80,
            'ToPort': 80,
            'IpRanges':[{'CidrIp': '0.0.0.0/0'}]
        }
    ]
)
sec_grp_id = security_group.id

instances = []
for i, zone in enumerate(AVAILABILITY_ZONES):
    try:
        instance = ec2.create_instances(
            ImageId=AMI_ID,
            MinCount=1,
            MaxCount=1,
            KeyName=KEY_NAME,
            InstanceType='t2.micro',
            SecurityGroupIds=[sec_grp_id],
            Placement={
                'AvailabilityZone': zone
            },
            SubnetId=SUBNETS[i],
            TagSpecifications=[
                {
                    'ResourceType': 'instance',
                    'Tags': [
                        {
                            'Key': 'Name',
                            'Value': '23796349-vm' + str(i+1)
                        },
                    ]
                },
            ],
        )[0]
        instances.append(instance)
    except Exception as e:
        print(f"Error creating instance in {zone}: {e}")

# Wait for instances to be in running state
waiter = ec2.meta.client.get_waiter('instance_running')
waiter.wait(InstanceIds=[instance.id for instance in instances])

# Fetch and print the public IP addresses of the instances
for instance in instances:
    instance.reload() # Refresh instance details
    print(f"Instance {instance.id} Public IP: {instance.public_ip_address}")

# Initialize the ELBV2 client
client = boto3.client('elbv2')

# Create the Application Load Balancer
response = client.create_load_balancer(

```

```

        Name='23796349-ALB',
        Subnets=SUBNETS[:2],
        SecurityGroups=[sec_grp_id],
        Scheme='internet-facing',
        Tags=[
            {
                'Key': 'Name',
                'Value': '23796349-ALB'
            },
        ],
    )
lb_arn = response['LoadBalancers'][0]['LoadBalancerArn']
print(f"Load Balancer ARN: {lb_arn}")

# Create a target group
response = client.create_target_group(
    Name='23796349-TG',
    Protocol='HTTP',
    Port=80,
    VpcId=VPC_ID,
    Tags=[
        {
            'Key': 'Name',
            'Value': '23796349-TG'
        },
    ],
)
tg_arn = response['TargetGroups'][0]['TargetGroupArn']
print(f"Target Group ARN: {tg_arn}")

# Register targets in the target group
client.register_targets(
    TargetGroupArn=tg_arn,
    Targets=[
        {'Id': instances[0].id},
        {'Id': instances[1].id}
    ]
)

# Create a listener
client.create_listener(
    LoadBalancerArn=lb_arn,
    Protocol='HTTP',
    Port=80,
    DefaultActions=[
        {
            'Type': 'forward',
            'TargetGroupArn': tg_arn
        },
    ],
)
print("2 EC2 instances and an Application Load Balancer successfully created!")

```

- The screenshots below show everything created using the above code:

Code output

```
Activities Terminal Sep 4 19:54
adharsh@adharsh:~$ cd cits5503/labs/
adharsh@adharsh:~/cits5503/labs$ ls
createalb.py
adharsh@adharsh:~/cits5503/labs$ python3 createalb.py
Instance i-0746e1cd215ad7fe1 Public IP: 43.203.124.68
Instance i-0c1a17b68538c9473 Public IP: 54.180.103.72
Load Balancer ARN: arn:aws:elasticloadbalancing:ap-northeast-2:489389878001:loadbalancer/app/23796349-ALB/e3b27d64a5bcb62b
Target Group ARN: arn:aws:elasticloadbalancing:ap-northeast-2:489389878001:targetgroup/23796349-TG/523c7a95f66116b4
2 EC2 Instances and an Application Load Balancer successfully created!
```

Security group

Details

Security group name	Security group ID	Description	VPC ID
23796349-sg	sg-02fe53250980e5f29	Security group for ssh and http inbound traffic	vpc-01f842220d0070f97

Inbound rules (2)

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
-	sgr-04bbd65545db08...	IPv4	SSH	TCP	22	0.0.0.0/0	-
-	sgr-0ce90ff0f0852b90	IPv4	HTTP	TCP	80	0.0.0.0/0	-

Two EC2 instances

Instances (2) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic I...
23796349-vm2	i-0c1a17b68538c9473	Running	t2.micro	2/2 checks passed	User: arn:aws:	ap-northeast-2c	ec2-54-180-103-72.ap...	54.180.103.72	-
23796349-vm1	i-0746e1cd215ad7fe1	Running	t2.micro	2/2 checks passed	User: arn:aws:	ap-northeast-2a	ec2-43-203-124-68.ap...	43.203.124.68	-

Application Load Balancer

The screenshot shows the AWS EC2 Load Balancers console. On the left, a sidebar navigation includes: EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations (New), Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), and Load Balancing (Load Balancers, Target Groups, Trust Stores). The main content area displays the details of the load balancer '23796349-ALB'. It shows the load balancer type is Application, status is Active, VPC is vpc-01f842220d0070f97, and it's internet-facing. The hosted zone is ZWKZPGTH4BKDX. Availability zones include subnet-0b360189181ee48e (ap-northeast-2c) and subnet-09f996d1ad81767a9 (ap-northeast-2a). The load balancer ARN is arn:aws:elasticloadbalancing:ap-northeast-2:489389878001:loadbalancer/app/23796349-ALB/e3b27d64a5 bcb62b. The DNS name is 23796349-ALB-1909200667.ap-northeast-2.elb.amazonaws.com (A Record). Below this, the 'Listeners and rules' tab is selected, showing one listener rule for port 80 forwarding to target group 23796349-TG.

Target Group

The screenshot shows the AWS Target Groups console. The sidebar navigation is identical to the previous screenshot. The main content area displays the details of the target group '23796349-TG'. It shows the target type is Instance, protocol is HTTP: 80, and protocol version is HTTP1. The VPC is vpc-01f842220d0070f97. There are 2 total targets: 0 healthy, 2 unhealthy, 0 unused, 0 initial, and 0 draining. The distribution of targets by Availability Zone (AZ) is shown in a table. Below this, the 'Registered targets' section lists two instances: i-0c1a17b68538c9473 and i-0746e1cd215ad7fe1, both marked as unhealthy due to failed health checks.

[3] Test the Application Load Balancer

- The load balancer initially wasn't working and the two EC2 instances were showing up as **Unhealthy** targets in the Target Group dashboard.
- The reason being, target group health verification happens over port 80 but the two instances weren't ready to use **port 80 (HTTP)**.
- This problem got sorted a few minutes after **Apache 2** was installed on both instances.
- To connect to the instances using ssh, the following command was used:

```
ssh -i <path to key-pair file> ubuntu@<instance public IP>
```

```
#instance 1
ssh -i /home/adharsh/cits5503/23796349-key.pem ubuntu@43.203.124.68
```

```
adharsh@adharsh:~/cits5503/labs$ ssh -i /home/adharsh/cits5503/23796349-key.pem ubuntu@43.203.124.68
The authenticity of host '43.203.124.68' (43.203.124.68) can't be established.
ED25519 key fingerprint is SHA256:OkXWLzI9nssVf08l/P1QuhnF5TA6GcpjW3JRTx08KLC.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '43.203.124.68' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1022-aws x86_64)

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

System information as of Wed Sep 4 11:32:05 UTC 2024

System load: 0.0      Processes:          99
Usage of /:   20.7% of 7.57GB   Users logged in:    0
Memory usage: 20%           IPv4 address for eth0: 172.31.8.190
Swap usage:   0%
```

and

```
#instance 2
ssh -i /home/adharsh/cits5503/23796349-key.pem ubuntu@54.180.103.72
```

```
Activities Terminal Sep 4 20:03 ⓘ
ubuntu@ip-172-31-46-44: ~
adharsh@adharsh:~/cits5503/labs$ ssh -i /home/adharsh/cits5503/23796349-key.pem ubuntu@54.180.103.72
The authenticity of host '54.180.103.72' ('54.180.103.72') can't be established.
ED25519 key fingerprint is SHA256:9tNyRFwxxuGfvh10CEYAY9k9m0Wx91lop9vVOgmzTBw.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.180.103.72' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1022-aws x86_64)

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

System information as of Wed Sep 4 11:59:06 UTC 2024

System load: 0.0      Processes:          98
Usage of /:   20.7% of 7.57GB   Users logged in:    0
Memory usage: 20%           IPv4 address for eth0: 172.31.46.44
Swap usage:   0%
```

NOTE: This **key-pair** file was created in [Lab 2](#). I used the same file by copying it to this working directory.

- Each instance was updated using the command:

```
sudo apt-get update
```

Instance 1

```
Activities Terminal Sep 4 19:54 ⓘ
ubuntu@ip-172-31-8-190: ~
ubuntu@ip-172-31-8-190: ~
Get:1 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy InRelease [128 kB]
Get:2 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:6 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:9 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:10 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:11 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1988 kB]
```

Instance 2

```
Activities Terminal Sep 4 19:54 ⓘ
ubuntu@ip-172-31-46-44: ~
ubuntu@ip-172-31-46-44: ~
Hit:1 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:5 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:9 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:10 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:11 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1988 kB]
```

- The following command was used to install Apache 2:

```
sudo apt install apache2
```

Instance 1

```
Activities Terminal Sep 4 19:55 ubuntu@ip-172-31-8-190:~  
ubuntu@ip-172-31-8-190:~$ sudo apt install apache2  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  apache2-bin apache2-data apache2-utils bzip2 libapr1 libaprutil1  
  libaprutil1-dbd-sqlite3 libaprutil1-ldap liblbuas5.3-0 mailcap mime-support  
  ssl-cert  
Suggested packages:  
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser  
  bzip2-dbg  
The following NEW packages will be installed:  
  apache2 apache2-bin apache2-data apache2-utils bzip2 libapr1 libaprutil1  
  libaprutil1-dbd-sqlite3 libaprutil1-ldap liblbuas5.3-0 mailcap mime-support  
  ssl-cert  
0 upgraded, 13 newly installed, 0 to remove and 51 not upgraded.  
Need to get 2141 kB of archives.  
After this operation, 8524 kB of additional disk space will be used.  
Do you want to continue? [Y/n] y  
Get:1 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libapr1 amd64 1.7.0-8ubuntu0.22.04.1 [108 kB]  
Get:2 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libaprutil1 amd64 1.6.1-Subuntu4.22.04.2 [92.8 kB]  
Get:3 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libaprutil1-dbd-sqlite3 amd64 1.6.1-Subuntu4.22.04.2 [111.3 kB]  
Get:4 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libaprutil1-ldap amd64 1.6.1-Subuntu4.22.04.2 [9176 B]  
Get:5 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 liblbuas5.3-0 amd64 5.3.6-1build1 [146 kB]  
Get:6 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 apache2-bin amd64 2.4.52-1ubuntu4.12 [1348 kB]  
Get:7 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 apache2-data all 2.4.52-1ubuntu4.12 [165 kB]  
Get:8 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 mailcap all 3.70+mutubuntu1 [23.8 kB]  
Get:9 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 apache2-utils amd64 2.4.52-1ubuntu4.12 [89.1 kB]  
Get:10 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 apache2 amd64 2.4.52-1ubuntu4.12 [97.9 kB]  
Get:11 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 apache2 and64 2.4.52-1ubuntu4.12 [34.8 kB]  
Get:12 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 bzip2 amd64 1.0.8-5build1 [34.8 kB]  
Get:13 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 ssl-cert all 1.1.2 [17.4 kB]  
Fetched 2141 kB in 1s (1542 kB/s)  
Selecting previously unselected package libapr1:amd64.  
(Reading database... 65320 files and directories currently installed.)  
Preparing to unpack .../00-libapr1_1.7.0-8ubuntu0.22.04.1_amd64.deb ...  
Unpacking libapr1:amd64 (1.7.0-8ubuntu0.22.04.1) ...  
Selecting previously unselected package libaprutil1:amd64.  
Preparing to unpack .../01-libaprutil1_1.6.1-Subuntu4.22.04.2_amd64.deb ...  
Unpacking libaprutil1:amd64 (1.6.1-Subuntu4.22.04.2) ...  
Selecting previously unselected package libaprutil1-dbd-sqlite3:amd64.  
Preparing to unpack .../02-libaprutil1-dbd-sqlite3_1.6.1-Subuntu4.22.04.2_amd64.deb ...  
Unpacking libaprutil1-dbd-sqlite3:amd64 (1.6.1-Subuntu4.22.04.2) ...  
Selecting previously unselected package libaprutil1-ldap:amd64.  
Preparing to unpack .../03-libaprutil1-ldap_1.6.1-Subuntu4.22.04.2_amd64.deb ...  
Unpacking libaprutil1-ldap:amd64 (1.6.1-Subuntu4.22.04.2) ...  
Selecting previously unselected package liblbuas5.3-0:amd64.  
Preparing to unpack .../04-liblbuas5.3-0_5.3.6-1build1_amd64.deb ...  
Unpacking liblbuas5.3-0:amd64 (5.3.6-1build1) ...  
Selecting previously unselected package apache2-bin.  
Preparing to unpack .../05-apache2-bin_2.4.52-1ubuntu4.12_amd64.deb ...  
Unpacking apache2-bin (2.4.52-1ubuntu4.12) ...  
Selecting previously unselected package apache2-data.  
Preparing to unpack .../06-apache2-data_2.4.52-1ubuntu4.12_all.deb ...
```

Instance 2

```
Activities Terminal Sep 4 20:03 ubuntu@ip-172-31-46-44:~  
ubuntu@ip-172-31-46-44:~$ sudo apt install apache2  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  apache2-bin apache2-data apache2-utils bzip2 libapr1 libaprutil1  
  libaprutil1-dbd-sqlite3 libaprutil1-ldap liblbuas5.3-0 mailcap mime-support  
  ssl-cert  
Suggested packages:  
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser  
  bzip2-dbg  
The following NEW packages will be installed:  
  apache2 apache2-bin apache2-data apache2-utils bzip2 libapr1 libaprutil1  
  libaprutil1-dbd-sqlite3 libaprutil1-ldap liblbuas5.3-0 mailcap mime-support  
  ssl-cert  
0 upgraded, 13 newly installed, 0 to remove and 51 not upgraded.  
Need to get 2141 kB of archives.  
After this operation, 8524 kB of additional disk space will be used.  
Do you want to continue? [Y/n] y  
Get:1 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libapr1 amd64 1.7.0-8ubuntu0.22.04.1 [108 kB]  
Get:2 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libaprutil1 amd64 1.6.1-Subuntu4.22.04.2 [92.8 kB]  
Get:3 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libaprutil1-dbd-sqlite3 amd64 1.6.1-Subuntu4.22.04.2 [111.3 kB]  
Get:4 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libaprutil1-ldap amd64 1.6.1-Subuntu4.22.04.2 [9176 B]  
Get:5 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 liblbuas5.3-0 amd64 5.3.6-1build1 [146 kB]  
Get:6 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 apache2-bin amd64 2.4.52-1ubuntu4.12 [1348 kB]  
Get:7 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 apache2-data all 2.4.52-1ubuntu4.12 [165 kB]  
Get:8 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 mailcap all 3.70+mutubuntu1 [23.8 kB]  
Get:9 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 apache2-utils amd64 2.4.52-1ubuntu4.12 [89.1 kB]  
Get:10 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 apache2 and64 2.4.52-1ubuntu4.12 [97.9 kB]  
Get:11 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 apache2 and64 2.4.52-1ubuntu4.12 [34.8 kB]  
Get:12 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 bzip2 amd64 1.0.8-5build1 [34.8 kB]  
Get:13 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 ssl-cert all 1.1.2 [17.4 kB]  
Fetched 2141 kB in 1s (2527 kB/s)  
Selecting previously unselected package libapr1:amd64.  
(Reading database... 65320 files and directories currently installed.)  
Preparing to unpack .../00-libapr1_1.7.0-8ubuntu0.22.04.1_amd64.deb ...  
Unpacking libapr1:amd64 (1.7.0-8ubuntu0.22.04.1) ...  
Selecting previously unselected package libaprutil1:amd64.  
Preparing to unpack .../01-libaprutil1_1.6.1-Subuntu4.22.04.2_amd64.deb ...  
Unpacking libaprutil1:amd64 (1.6.1-Subuntu4.22.04.2) ...  
Selecting previously unselected package libaprutil1-dbd-sqlite3:amd64.  
Preparing to unpack .../02-libaprutil1-dbd-sqlite3_1.6.1-Subuntu4.22.04.2_amd64.deb ...  
Unpacking libaprutil1-dbd-sqlite3:amd64 (1.6.1-Subuntu4.22.04.2) ...  
Selecting previously unselected package libaprutil1-ldap:amd64.  
Preparing to unpack .../03-libaprutil1-ldap_1.6.1-Subuntu4.22.04.2_amd64.deb ...  
Unpacking libaprutil1-ldap:amd64 (1.6.1-Subuntu4.22.04.2) ...  
Selecting previously unselected package liblbuas5.3-0:amd64.  
Preparing to unpack .../04-liblbuas5.3-0_5.3.6-1build1_amd64.deb ...  
Unpacking liblbuas5.3-0:amd64 (5.3.6-1build1) ...  
Selecting previously unselected package apache2-bin.  
Preparing to unpack .../05-apache2-bin_2.4.52-1ubuntu4.12_amd64.deb ...
```

- Now that Apache 2 was installed in both instances, both the instances show up as healthy in the Target Group Dashboard:

The screenshot shows the AWS EC2 Dashboard with the target group '23796349-TG' selected. The 'Details' section shows the target type as 'Instance', protocol as 'HTTP: 80', and the load balancer as '23796349-ALB'. There are 2 total targets, both of which are healthy. The VPC is listed as 'vpc-01f842220d0070f97'. Below this, a table shows the distribution of targets by Availability Zone (AZ). The 'Targets' tab is selected, showing two registered targets: instance i-0c1a17b68538c9473 in zone ap-northeast-2c and instance i-0746e1cd215ad7fe1 in zone ap-northeast-2a, both marked as healthy.

- To change the content in the **title** tag, use the following command:

NOTE: I learnt **sed** operations in the unit **CITS4407 - Open Source Tools and Scripting**

```
#instance 1
sudo sed -i "s|<title>.*</title>|<title>23796349-vm1</title>|g"
/var/www/html/index.html
```

The terminal window shows the command being run: `sudo sed -i "s|<title>.*</title>|<title>23796349-vm1</title>|g" /var/www/html/index.html`. The output shows the file being modified, with the title tag content changing from '23796349-vm1' to '23796349-vm2'.

```
Activities Terminal Sep 4 19:57
ubuntu@ip-172-31-8-190:~$ sudo sed -i "s|<title>.*</title>|<title>23796349-vm1</title>|g" /var/www/html/index.html
ubuntu@ip-172-31-8-190:~$ cat /var/www/html/index.html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--
  Modified from the Debian original for Ubuntu
  Last updated: 2022-03-22
  See: https://launchpad.net/bugs/1906004
-->
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>23796349-vm1</title>
  <style type="text/css" media="screen">
    {
      margin: 0px 0px 0px;
      padding: 0px 0px 0px 0px;
    }
    body, html {
      padding: 3px 3px 3px 3px;
      background-color: #D0DBE2;
      font-family: Ubuntu, Verdana, sans-serif;
      font-size: 11pt;
      text-align: center;
    }
    div.main_page {
      position: relative;
      display: table;
      width: 800px;
      margin-bottom: 3px;
      margin-left: auto;
      margin-right: auto;
      padding: 0px 0px 0px 0px;
      border-width: 2px;
      border-color: #212738;
      border-style: solid;
      background-color: #FFFFFF;
      text-align: center;
    }
    div.page_header {
      height: 180px;
      width: 100%;
      background-color: #F5F6F7;
    }
  </style>
</head>
<body>
  <div>
    <h1>Hello World</h1>
    <p>This is a test page</p>
  </div>
</body>
</html>
```

and

```
#instance 2
sudo sed -i "s|<title>.*</title>|<title>23796349-vm2</title>|g"
```

/var/www/html/index.html

```
Activities Terminal Sep 4 20:04
ubuntu@ip-172-31-8-190: ~
ubuntu@ip-172-31-46-44: ~
ubuntu@ip-172-31-46-44: ~
ubuntu@ip-172-31-46-44: $ sudo sed -i "s|<title>.*</title>|<title>23796349-vm2</title>|g" /var/www/html/index.html
ubuntu@ip-172-31-46-44: $ cat /var/www/html/index.html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--
Modified from the Debian original for Ubuntu
Last updated: 2022-03-22
See: https://launchpad.net/bugs/1966004
-->
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>23796349-vm2</title>
<style type="text/css" media="screen">
* {
margin: 0px 0px 0px 0px;
padding: 0px 0px 0px 0px;
}
body, html {
padding: 3px 3px 3px 3px;
background-color: #D8DBE2;
font-family: Ubuntu, Verdana, sans-serif;
font-size: 11pt;
text-align: center;
}
div.main_page {
position: relative;
display: table;
width: 800px;
margin-bottom: 3px;
margin-left: auto;
margin-right: auto;
padding: 0px 0px 0px 0px;
border-width: 2px;
border-color: #212738;
border-style: solid;
background-color: #FFFFFF;
text-align: center;
}
div.page_header {
height: 180px;
width: 100%;
background-color: #F5F6F7;
}

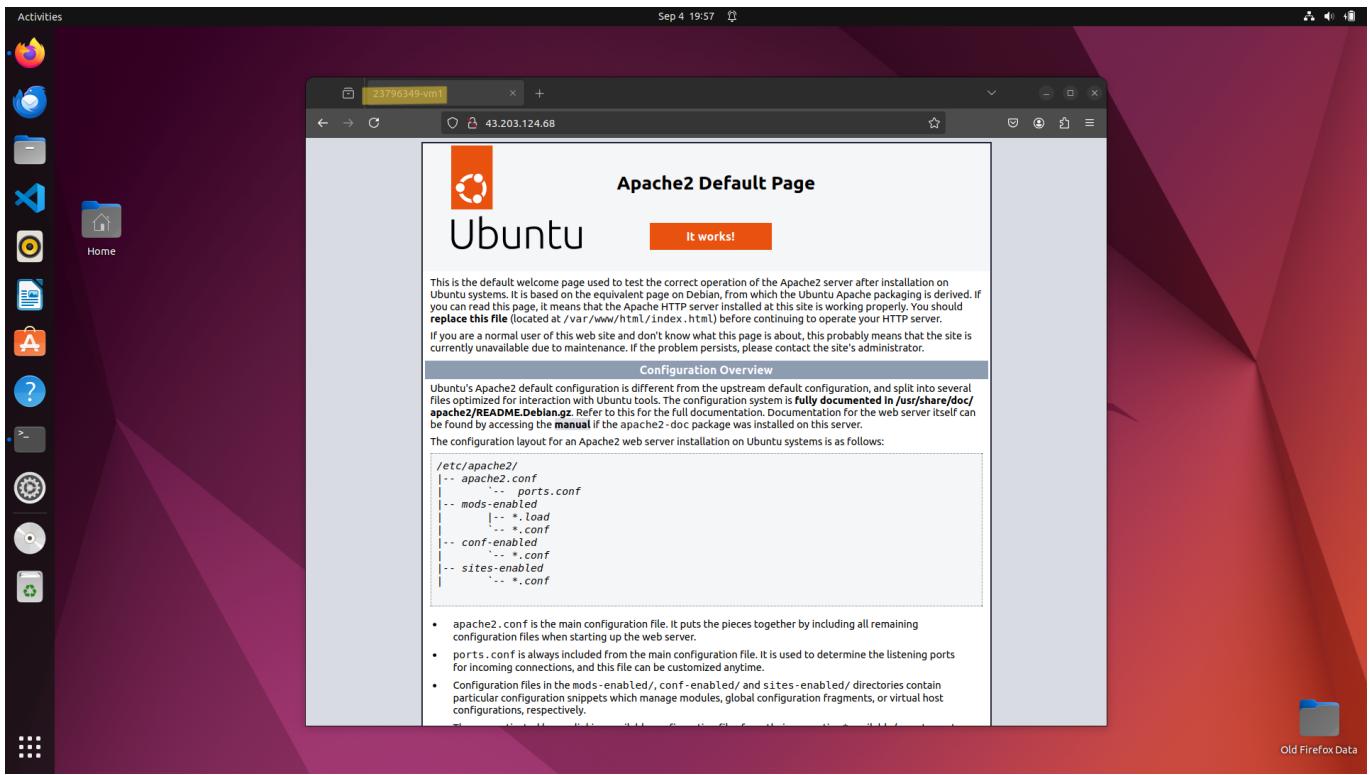
```

NOTE: To see the changes, the following command can be used:

```
cat /var/www/html/index.html
```

- Alternatively, visiting the public IP of each instance through browser can also be used to verify the changes (screenshots below):

Instance 1

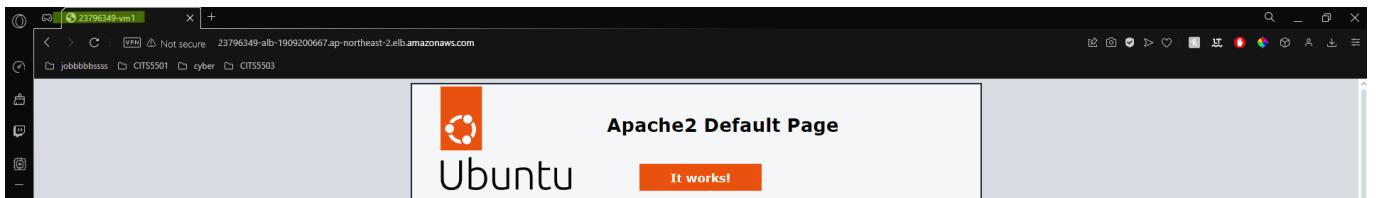


Instance 2

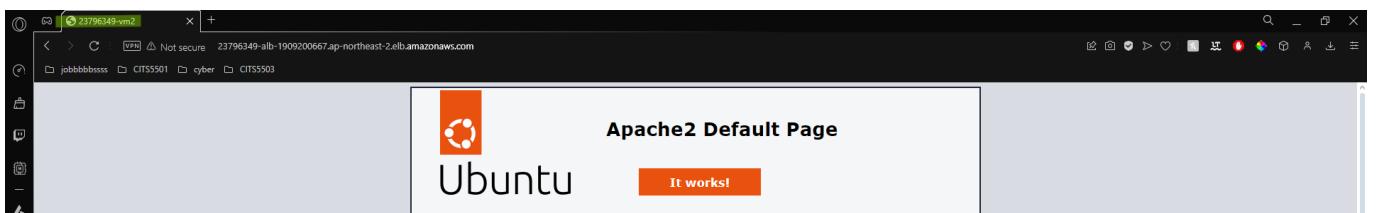


- To check if the load balancer works, copy the **DNS Name** of your load balancer and use it in the browser.
- Each time the page is reloaded, the index.html file of instance 1 and instance 2 are loaded equally, one for each reload (screenshots below):

When the page is loaded



After page reload



- All resources created for this lab were deleted using AWS Console after the completion.