

Labs 1-5

Student ID: 23905652

Student Name: Zhe Han

Lab 1 Intro Set up

AWS Account and Log in

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

Navigate in the browser to <https://489389878001.signin.aws.amazon.com/console>, enter the **account root user ID 489389878001**, as well as the **IAM user name** and **password** provided in student email, and click login.

Sign in as IAM user

Account ID (12 digits) or account alias

489389878001

IAM user name

23905652@student.uwa.edu.au

Password

●●●●●●●●●●●●

Remember this account

Sign in

[Sign in using root user email](#)

[Forgot password?](#)

After logging in, change the password to a personal one.

AWS account 489389878001

IAM user name 23905652@student.uwa.edu.au

Old password [REDACTED]

New password [REDACTED]

Retype new password [REDACTED]

Confirm password change

[Sign in using root user email](#)

[2] Search and open Identity Access Management

After entering the home page, click the dropdown menu of **the account name** in the upper-right corner, and select "**Security credentials**" to navigate to the Security Credentials page. Here, we can see some detailed information about the user.

The screenshot shows the AWS Console Home page. In the top navigation bar, the account name "23905652@student.uwa.edu.au @ 4893-8987-8001" is visible. Below it, the "Services" dropdown is open, showing various AWS services like IAM, Systems Manager, Security Lake, IAM Identity Center, EC2, S3, and Key Management Service. The "IAM" service is selected. On the right side, there's a sidebar with account information (Account ID: 4893-8987-8001, IAM user: 23905652@student.uwa.edu.au) and a dropdown menu. The "Security credentials" option in this menu is highlighted with a red box.

My security credentials Info

Manage credentials for your currently authenticated IAM user. To learn more about the types of AWS credentials and how they're used, see [AWS Security Credentials](#)



You don't have MFA assigned

As a security best practice, we recommend you assign MFA.

[Assign MFA](#)

Account details

User name
23905652@student.uwa.edu.au

AWS account ID
[489389878001](#)

User ARN
[arn:aws:iam::489389878001:user/23905652@student.uwa.edu.au](#)

Canonical user ID
[2a5fac7aada1ad2caa48c9ab08cc4e2428d4eb596108daa3b59f1204ae96482e](#)

[AWS IAM credentials](#)

[AWS CodeCommit credentials](#)

[Amazon Keyspaces credentials](#)

Console sign-in

[Update console password](#)

Console sign-in link
<https://489389878001.signin.aws.amazon.com/console>

Console password
Updated 18 minutes ago (2024-08-13 07:25 GMT+8)

Last console sign-in

Scroll down the page, and in the "**Access keys**" section, click "create an access key". And for the use case, select "**Command Line Interface (CLI)**" to allow command-line control.

Access key best practices & alternatives Info

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

Command Line Interface (CLI)

You plan to use this access key to enable the AWS CLI to access your AWS account.

Local code

You plan to use this access key to enable application code in a local development environment to access your AWS account.

Application running on an AWS compute service

You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

Third-party service

You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

Application running outside AWS

You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.

Other

Your use case is not listed here.

Create a tag for the access key, which can include the purpose of the access key to differentiate between different access keys. After entering the details, click "Create access key."

Set description tag - *optional* Info

The description for this access key will be attached to this user as a tag and shown alongside the access key.

Description tag value

Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

CITS5503_V1

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: _ . : / = + - @

[Cancel](#)

[Previous](#)

[Create access key](#)

After successfully creating the key, click "Download .csv file" to save the Access Key ID and Secret Access Key for secure storage. If the Access Key ID and Secret Access Key are compromised, these details can allow someone to create extensive resources and misuse the user account, so **we have to handle them with caution**.

Retrieve access keys Info

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIAXD4P15LYTD4N4RI4	***** Show

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [best practices for managing AWS access keys](#).

[Download .csv file](#)

[Done](#)

Set up recent Linux OSes

For the labs in this unit, WSL on Windows will be used. Install WSL from the command line by running:

```
> wsl --install -d Ubuntu-20.04
```

This command will enable the features required to run WSL and install the recommended version of Ubuntu 20.04 from the lab sheet. Once installed, enter a username and password to complete the setup:

```
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: mayhan
New password:
Retype new password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.153.1-microsoft-standard-WSL2 x86_64)

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

 System information as of Tue Aug 13 13:30:23 CST 2024

System load: 0.0          Processes:           74
Usage of /: 0.1% of 1006.85GB   Users logged in:      0
Memory usage: 3%           IPv4 address for eth0: 172.27.66.180
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

This message is shown once a day. To disable it please create the
/home/mayhan/.hushlogin file.
mayhan@DESKTOP-KEJ3P3J:~$ -
```

Install Linux packages

[1] Install Python 3.8.x

Enter the following command in the terminal to check the Python version:

```
python3 -v
```

Enter the following command to install pip3, which is a tool that will allow us to install and manage Python libraries:

```
sudo apt install -y python3-pip
```

```

mayhan@DESKTOP-KEJ3P3J:~$ python3 -V
Python 3.8.10
mayhan@DESKTOP-KEJ3P3J:~$ sudo apt install -y python3-pip
[sudo] password for mayhan:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential cpp cpp-9 dpkg-dev fakeroot g++ g++-9 gcc gcc-9
  gcc-9-base libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan5 libatomic1 libbinutils
  libc-dev-bin libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0 libctf0 libdpkg-perl libexpat1-dev libfakeroot
  libfile-fcntllock-perl libgcc-9-dev libgomp1 libis122 libitm1 liblsan0 libmpc3 libpython3-dev libpython3.8-dev
  libquadmath0 libstdc++-9-dev libtsan0 libubsan1 linux-libc-dev make manpages-dev python-pip-whl python3-dev
  python3-wheel python3.8-dev zlib1g-dev
Suggested packages:
  binutils-doc cpp-doc gcc-9-locales debian-keyring g++-multilib g++-9-multilib gcc-9-doc gcc-multilib autoconf
  automake libtool flex bison gdb gcc-doc gcc-9-multilib glibc-doc bzr libstdc++-9-doc make-doc
The following NEW packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential cpp cpp-9 dpkg-dev fakeroot g++ g++-9 gcc gcc-9
  gcc-9-base libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan5 libatomic1 libbinutils
  libc-dev-bin libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0 libctf0 libdpkg-perl libexpat1-dev libfakeroot
  libfile-fcntllock-perl libgcc-9-dev libgomp1 libis122 libitm1 liblsan0 libmpc3 libpython3-dev libpython3.8-dev
  libquadmath0 libstdc++-9-dev libtsan0 libubsan1 linux-libc-dev make manpages-dev python-pip-whl python3-dev
  python3-pip python3-wheel python3.8-dev zlib1g-dev
0 upgraded, 50 newly installed, 0 to remove and 0 not upgraded.
Need to get 52.3 MB of archives.

```

From the output, we have successfully installed Python 3.8.10 and pip3 in WSL.

[2] Install awscli

Enter the following command to install awscli:

```
sudo apt install awscli
```

```

mayhan@DESKTOP-KEJ3P3J: $ sudo apt install awscli
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  docutils-common fontconfig fonts-droid-fallback fonts-noto-mono fonts-urw-base35 ghostscript groff gsfonts
  hicolor-icon-theme imagemagick imagemagick-6-common imagemagick-6.q16 libavahi-client3 libavahi-common-data
  libavahi-common3 libcairo2 libcurl2 libdatriel1 libdjvuibre-text libdjvuibre21 libfftw3-double3 libgraphite2-3
  libgs9 libgs9-common libharfbuzz0b libidn11 libjs-0.35 libilmbase24 libimagequant0 libjbig0 libjbig2dec0
  libjpeg-turbo8 libjpeg8 liblcms2-2 liblqr-1-0 libmagickcore-6.q16-6 libmagickcore-6.q16-6-extra
  libmagickwand-6.q16-6 libnetpbm10 libopenexr24 libopenjp2-7 libpango-1.0-0 libpangocairo-1.0-0 libpangoft2-1.0-0
  libpaper-utils libpaper1 libpixman-1-0 libthai-data libthai0 libtiff5 libwebp6 libwebpdmux2 libwebpdmux3 libwmf0.2-7
  libxcb-render0 netpbm poppler-data psutils python3-botocore python3-dateutil python3-docutils python3-jmespath
  python3-olefile python3-pil python3-pygments python3-roman python3-rsa python3-s3transfer sgml-base xml-core
Suggested packages:
  fonts-noto fonts-freefont-otf | fonts-freefont-ttf fonts-texgyre ghostscript-x imagemagick-doc autotrace cups-bsd
  | lpr | lprng enscript ffmpeg gimp gnuplot grads graphviz hp2xx html2ps libwmf-bin mplayer povray radience
  sane-utils texlive-base-bin transfig ufraw-batch xdg-utils cups-common libfftw3-bin libfftw3-dev liblcms2-utils
  inkscape libjxr-tools libwmf0.2-7-gtk poppler-utils fonts-japanese-mincho | fonts-ipafont-mincho
  fonts-japanese-gothic | fonts-ipafont-gothic fonts-archic-ukai fonts-archic-uming fonts-nanum docutils-doc
  fonts-linuxlibertine | ttf-linux-libertine texlive-lang-french texlive-latex-base texlive-latex-recommended
  python3-pil-doc python3-pil-dbg python3-pygments-doc ttf-bitstream-vera sgml-base-doc debhelper

```

Then:

```
pip3 install awscli --upgrade
```

```
mayhan@DESKTOP-KEJ3P3J: $ pip3 install awscli --upgrade
Collecting awscli
  Downloading awscli-1.33.41-py3-none-any.whl (4.5 MB)
    |██████████| 4.5 MB 584 kB/s
Requirement already satisfied, skipping upgrade: docutils<0.17,>=0.10 in /usr/lib/python3/dist-packages (from awscli) (0.16)
Collecting botocore==1.34.159
  Downloading botocore-1.34.159-py3-none-any.whl (12.5 MB)
    |██████████| 12.5 MB 8.9 MB/s
Requirement already satisfied, skipping upgrade: colorama<0.4.7,>=0.2.5 in /usr/lib/python3/dist-packages (from awscli) (0.4.3)
Requirement already satisfied, skipping upgrade: PyYAML<6.1,>=3.10 in /usr/lib/python3/dist-packages (from awscli) (5.3.1)
Requirement already satisfied, skipping upgrade: rsa<4.8,>=3.1.2 in /usr/lib/python3/dist-packages (from awscli) (4.0)
Collecting s3transfer<0.11.0,>=0.10.0
  Downloading s3transfer-0.10.2-py3-none-any.whl (82 kB)
    |██████████| 82 kB 785 kB/s
Requirement already satisfied, skipping upgrade: urllib3<1.27,>=1.25.4; python_version < "3.10" in /usr/lib/python3/dist-packages (from botocore==1.34.159->awscli) (1.25.8)
Requirement already satisfied, skipping upgrade: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3/dist-packages (from botocore==1.34.159->awscli) (0.9.4)
Requirement already satisfied, skipping upgrade: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3/dist-packages (from botocore==1.34.159->awscli) (2.7.3)
Installing collected packages: botocore, s3transfer, awscli
Successfully installed awscli-1.33.41 botocore-1.34.159 s3transfer-0.10.2
```

From the output, we can see that the installation was successful.

[3] Configure AWS

Enter the following command to start configuring AWS:

```
aws configure
```

Enter the access key ID and secret access key generated in the first step. Set the region name to **ap-southeast-1** based on the student number:

```
mayhan@DESKTOP-KEJ3P3J: $ aws configure
AWS Access Key ID [None]: AKIAJD4PI5LYTD4N4RI4
AWS Secret Access Key [None]: [REDACTED]
Default region name [None]: ap-southeast-1
Default output format [None]: json
```

[4] Install boto3

Enter the following command to install boto3:

```
pip3 install boto3
```

```
mayhan@DESKTOP-KEJ3P3J: $ pip3 install boto3
Collecting boto3
  Downloading boto3-1.34.159-py3-none-any.whl (139 kB)
    |██████████| 139 kB 567 kB/s
Requirement already satisfied: s3transfer<0.11.0,>=0.10.0 in ./local/lib/python3.8/site-packages (from boto3) (0.10.2)
Requirement already satisfied: botocore<1.35.0,>=1.34.159 in ./local/lib/python3.8/site-packages (from boto3) (1.34.159)
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/lib/python3/dist-packages (from boto3) (0.9.4)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/lib/python3/dist-packages (from botocore<1.35.0,>=1.34.159->boto3) (2.7.3)
Requirement already satisfied: urllib3<1.27,>=1.25.4; python_version < "3.10" in /usr/lib/python3/dist-packages (from botocore<1.35.0,>=1.34.159->boto3) (1.25.8)
Installing collected packages: boto3
Successfully installed boto3-1.34.159
```

Test the installed environment

[1] Test the AWS environment

Enter the following command to test the AWS environment:

```
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

This command outputs a table with information about the available AWS regions for the EC2 service. And we can obtain the output of this table indicates that **our AWS environment has been successfully configured.**

[2] Test the Python environment

We can test the Python environment by entering the following commands:

```
python3
>>> import boto3
>>> ec2 = boto3.client('ec2')
>>> response = ec2.describe_regions()
>>> print(response)
```

```

>>> 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-3.amazonaws.com', 'RegionName': 'eu-west-3', '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-1.amazonaws.com', 'RegionName': 'eu-west-1', 'OptInStatus': 'opt-in-not-required'},
        {'Endpoint': 'ec2.ap-northeast-3.amazonaws.com', 'RegionName': 'ap-northeast-3', '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-1.amazonaws.com', 'RegionName': 'ap-northeast-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.eu-central-1.amazonaws.com', 'RegionName': 'eu-central-1', 'OptInStatus': 'opt-in-not-required'},
        {'Endpoint': 'ec2.us-east-1.amazonaws.com', 'RegionName': 'us-east-1', '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-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'}
    ],
    'ResponseMetadata': {
        'RequestId': 'fc535be6-65b9-49fb-81c5-2b7a705ee145',
        'HTTPStatusCode': 200,
        'HTTPHeaders': {
            'x-amzn-requestid': 'fc535be6-65b9-49fb-81c5-2b7a705ee145',
            'cache-control': 'no-cache, no-store',
            'strict-transport-security': 'max-age=31536000; includeSubDomains',
            'vary': 'accept-encoding',
            'content-type': 'text/xml;charset=UTF-8',
            'content-length': '2890',
            'date': 'Tue, 13 Aug 2024 06:54:19 GMT',
            'server': 'AmazonEC2'
        },
        'RetryAttempts': 0
    }
}

```

This command will create an un-tabulated response, also retrieving information about the available AWS regions for the EC2 service.

[3] Write a Python script

Next, we will write a Python script to format the un-tabulated response into a table with only two columns: Endpoint and RegionName.

```

import boto3

# Create an EC2 client
ec2 = boto3.client('ec2')

# Get the region descriptions
response = ec2.describe_regions()

# Extract the needed data
regions = response['Regions']

# Print the header
# 'Endpoint':<40 means left-align and occupy a width of 40 characters.
# The second line prints 50 equal signs as a separator
print(f"{'Endpoint':<40} {'RegionName'}")
print("=".*50)

# Print each region's Endpoint and RegionName
for region in regions:
    endpoint = region['Endpoint']
    region_name = region['RegionName']
    print(f"{endpoint:<40} {region_name}")

```

Save the script in a file called '5503_lab01.py' and **run the script using the following command:**

```
python3 5503_lab01.py
```

The output will be as shown:

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 EC2 & Docker

Create an EC2 instance using awscli

[1] Create a security group

Security groups in AWS act as virtual firewalls for EC2 instances and other resources, controlling inbound and outbound traffic at the instance level to enhance network security. Enter the following command in the terminal to create a security group named "23905652-sg", named after the student number:

```
aws ec2 create-security-group --group-name 23905652-sg --description "security group for development environment"
```

Output:

```
● mayhan@DESKTOP-KEJ3P3J:~/lab$ aws ec2 create-security-group --group-name 23905652-sg --description "security group for development environment"
{
    "GroupId": "sg-0098114ed24ad0122"
}
```

Record the security group ID, "sg-0098114ed24ad0122", for later use.

[2] Authorise inbound traffic for ssh

Enter the following command to authorize inbound traffic for SSH. This command **adds a rule to the security group named "23905652-sg" that allows incoming TCP traffic on port 22 (SSH) from any IP address:**

```
aws ec2 authorize-security-group-ingress --group-name 23905652-sg --protocol tcp --port 22
--cidr 0.0.0.0/0
```

Where:

- `aws ec2 authorize-security-group-ingress`: This is the AWS CLI command used to add an inbound rule to the security group.
- `--group-name 23905652-sg`: Specifies the name of the security group where the rule is to be added.
- `--protocol tcp`: Sets the protocol for the rule to TCP.
- `--port 22`: Specifies that the rule applies to port 22, typically used for SSH connections.
- `--cidr 0.0.0.0/0`: Defines the allowed source IP range. 0.0.0.0/0 means "from any IP address."

```
bash: student: No such file or directory
● mayhan@DESKTOP-KEJ3P3J:~/lab$ aws ec2 authorize-security-group-ingress --group-name 23905652-sg --protocol tcp --port 22 --cidr 0.0.0.0/0
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0a44a2ce7f6236910",
            "GroupId": "sg-0098114ed24ad0122",
            "GroupOwnerId": "489389878001",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIpv4": "0.0.0.0/0"
        }
    ]
}
● mayhan@DESKTOP-KEJ3P3J:~/lab$
```

[3] Create a key pair

In AWS, **key pairs** provide secure access to EC2 instances, allowing for encrypted SSH connections to Linux instances or RDP to Windows instances. The following command is used to create a key pair named "23905652-key":

```
aws ec2 create-key-pair --key-name 23905652-key --query 'KeyMaterial' --output text > 23905652-key.pem
```

After running this command, it retrieves the private key material and saves it to a file named "23905652-key.pem" in the current directory.

To use this file to connect to the EC2 instance, we also need to set permissions for the file using the following command:

```
chmod 400 23905652-key.pem
```

This command sets read-only permissions for the file owner, restricting access for others.

[4] Create the instance

Use the following command to create an instance:

```
aws ec2 run-instances --image-id ami-0497a974f8d5dcef8 --security-group-ids 23905652-sg --count 1 --instance-type t2.micro --key-name 23905652-key --query 'Instances[0].InstanceId'
```

Where:

- **aws ec2 run-instances**: Used to create an EC2 instance.
- **--image-id ami-0497a974f8d5dcef8**: Specifies the Amazon Machine Image (AMI) for the instance. In this case, it corresponds to the AMI for the "ap-southeast-1" region.
- **--security-group-ids 23905652-sg**: Assigns the previously created security group to the instance.
- **--count 1**: Launches one instance.
- **--instance-type t2.micro**: Sets the instance type to t2.micro, a small general-purpose instance.
- **--key-name 23905652-key**: Associates the key pair just created with the instance to allow SSH access.

- `--query 'Instances[0].InstanceId'`: Filters the output to return only the ID of the created instance.

After executing this command, we will receive the ID of the created instance:

```
"i-06dc94e32950071fd"
```

[5] Add a tag to your Instance

Use the following command to add a tag to the instance, to help distinguish between different instances:

```
aws ec2 create-tags --resources i-06dc94e32950071fd --tags Key=lab02,Value=23905652
```

[6] Get the public IP address

Enter the following command to retrieve the public IP address of the instance:

```
aws ec2 describe-instances --instance-ids i-06dc94e32950071fd --query 'Reservations[0].Instances[0].PublicIpAddress'
```

The output will contain the public IP address:

```
"54.254.165.9"
```

[7] Connect to the instance via ssh

We use the key pair and the public IP address obtained in the previous step to connect to the instance via SSH:

```
ssh -i 23905652-key.pem ubuntu@54.254.165.9
```

Output:

```
mayhan@DESKTOP-KEJ3P3J:~/lab$ ssh -i 23905652-key.pem ubuntu@54.254.165.9
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 Sat Sep 14 09:50:23 UTC 2024

System load: 0.0          Processes:         98
Usage of /:   20.7% of 7.57GB  Users logged in:    0
Memory usage: 21%          IPv4 address for eth0: 172.31.23.233
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.

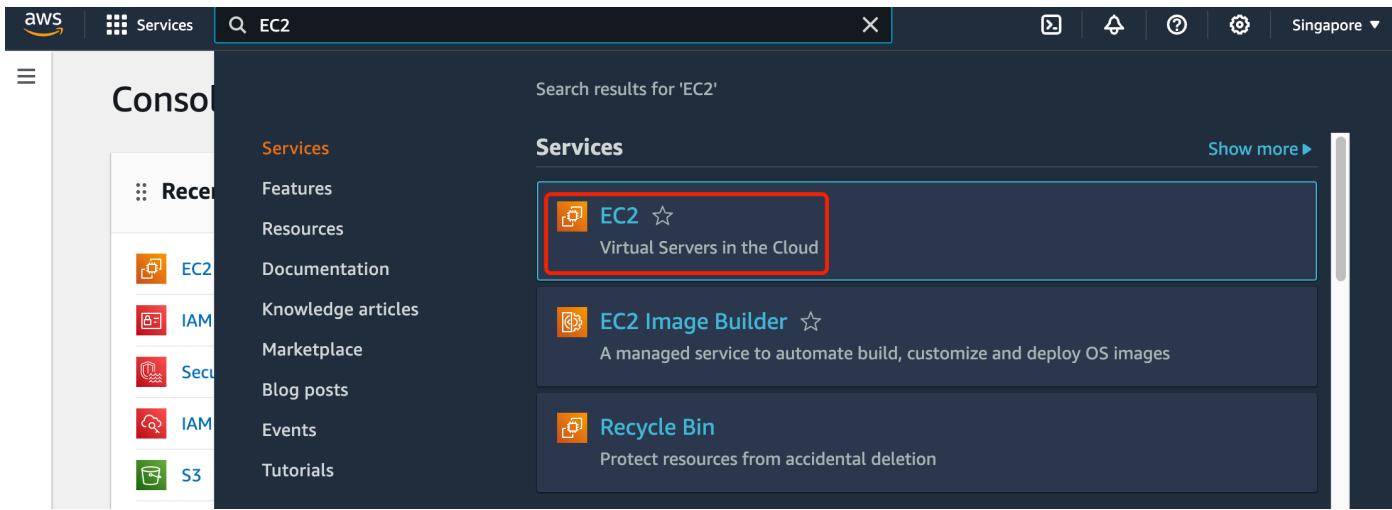
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-23-233:~$
```

Connection successful.

[8] List the created instance using the AWS console

In the AWS home page, search for "EC2" in the search bar to navigate to the EC2 service page.



On the left-hand menu of the EC2 service page, select "Instances" under the "Instances" section:

The screenshot shows the AWS EC2 Dashboard. On the left, there's a sidebar with sections like EC2 Dashboard, EC2 Global View, Events, Instances (with 'Instances' selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity, Reservations (marked as New), and Images (with AMIs). The main area is titled 'Resources' and shows various EC2 metrics: Instances (running) 0, Auto Scaling Groups 0, Capacity Reservations 0, Dedicated Hosts 0, Elastic IPs 4, Instances 13, Key pairs 54, Load balancers 0, Placement groups 0, Security groups 53, Snapshots 0, and Volumes 13. Below this is a 'Launch instance' button and a 'Service health' section.

We will be able to see the instance we created in the AWS console, name it "23905652-vm":

Instances (1/5) Info										
Last updated less than a minute ago C Connect Instance state ▾ Actions ▾ Launch instances ▾										
<input type="text" value="Find Instance by attribute or tag (case-sensitive)"/> All states ▾										
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elas...	Elas...
23909219-vm	i-011a8ff6b56a83a66	Stopped	t2.micro	-	User: arn:aw...	ap-southeast-1c	-	-	-	-
	i-082cc0f265d27efd7	Running	t2.micro	2/2 checks passed	User: arn:aw...	ap-southeast-1c	ec2-18-142-140-237.ap...	18.142.140.237	-	-
23905652-vm1	i-0f97f02b65d2532eb	Terminated	t2.micro	-	User: arn:aw...	ap-southeast-1a	-	-	-	-
23905652-vm2	i-08f04a80fbe65da9e	Terminated	t2.micro	-	User: arn:aw...	ap-southeast-1b	-	-	-	-
<input checked="" type="checkbox"/> 23905652-vm	i-06dc94e32950071fd	Running	t2.micro	2/2 checks passed	User: arn:aw...	ap-southeast-1b	ec2-54-254-165-9.ap.s...	54.254.165.9	-	-

Create an EC2 instance with Python Boto3

1. Next, we'll implement steps 1-6 using a Python script, step-by-step explanations are written in the code comments.:)

```
# Import necessary libraries. boto3 is AWS's Python SDK, os is used for file and directory operations
```

```
import boto3
import os

student_number = "23905652"
region = "ap-southeast-1"

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

# Step [1]
# Create a security group and get its ID
response = ec2.create_security_group(
    GroupName=f'{student_number}-sg',
    Description="security group for development environment"
)
security_group_id = response['GroupId']

# Step [2]
# Authorize inbound SSH traffic for the security group
ec2.authorize_security_group_ingress(
    GroupId=security_group_id,
    IpProtocol="tcp",
    FromPort=22,
    ToPort=22,
    CidrIp="0.0.0.0/0"
)

# Step [3]
# Create a key pair and save the private key to a file
response = ec2.create_key_pair(KeyName=f'{student_number}-key')
private_key = response['KeyMaterial']

private_key_file = f'{student_number}-key.pem'

# Allow writing to the private key file
with open(private_key_file, 'w') as key_file:
    key_file.write(private_key)

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

# Print message to confirm key file creation
print(f"Private key saved to {private_key_file}")

# Step [4] and [5]
# Launch an EC2 instance
response = ec2.run_instances(
    ImageId="ami-0497a974f8d5dcef8",
    SecurityGroupIds=[security_group_id],
    MinCount=1,
    MaxCount=1,
    InstanceType="t2.micro",
```

```

KeyName=f" {student_number}-key"
)

print("Instance created successfully")

instance_id = response[ 'Instances' ][0][ 'InstanceId' ]

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

# Step [6]
# Describe the instance to get its public IP address
response = ec2.describe_instances(InstanceIds=[instance_id])
public_ip_address = response[ 'Reservations' ][0][ 'Instances' ][0][ 'PublicIpAddress' ]
print(f"Instance created successfully with Public IP: {public_ip_address}")

```

2. Save the Python script as "lab2.py", and **run the script with the following command:**

```
python3 lab2.py
```

We will **get the IP address of the created instance:**

- mayhan@DESKTOP-KEJ3P3J:~/lab\$ python3 lab2.py
Private key saved to 23905652-key.pem

Instance created successfully with Public IP: 122.248.228.126

3. Use the same method to view the created instances in the AWS console, name it "23905652-vm2" this time to distinguish with the instance we created before:

Instances (1/3) Info										
Last updated less than a minute ago C Connect Instance state ▾ Actions ▾ Launch instances										
<input type="text" value="Find Instance by attribute or tag (case-sensitive)"/> All states ▾										
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...		
23905652-vm	i-01e86fbc2fb1b6f59	Terminated	t2.micro	-	User: arn:aws:	ap-southeast-1b	-	-		
23929804	i-01f3af3d68f304473	Running	t2.micro	2/2 checks passed	User: arn:aws:	ap-southeast-1b	ec2-13-215-48-37.ap-s...	13.215.48.37		
<input checked="" type="checkbox"/> 23905652-vm2	i-06bdbbe3245b2d47a9	Running	t2.micro	Initializing	User: arn:aws:	ap-southeast-1b	ec2-122-248-228-126.a...	122.248.228.126		

Below are the details, we can see that the public IP address matches the one output in Terminal earlier:

Instance summary for i-06bdbe3245b2d47a9 (23905652-vm2) Info	
Updated less than a minute ago	
Instance ID i-06bdbe3245b2d47a9 (23905652-vm2)	Public IPv4 address 122.248.228.126 open address
IPv6 address -	Instance state Running
Hostname type IP name: ip-172-31-23-168.ap-southeast-1.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-23-168.ap-southeast-1.compute.internal
Answer private resource DNS name -	Instance type t2.micro
Auto-assigned IP address 122.248.228.126 [Public IP]	VPC ID vpc-0ad7c05df6174aa82
IAM Role -	Subnet ID subnet-056a5c6c3bd465883
IMDSv2 Optional EC2 recommends setting IMDSv2 to required Learn more	Instance ARN arn:aws:ec2:ap-southeast-1:489389878001:instance/i-06bdbe3245b2d47a9
Elastic IP addresses -	
AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more	
Auto Scaling Group name -	

4. Finally, terminate the created instance in the console:

Instances (1/3) Info						
Last updated less than a minute ago						
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Available
23905652-vm	i-01e86fbc2fb1b6f59	Terminated	t2.micro	-	User: arn:aws:iam::137154837001:root	ap-southeast-1b
23929804	i-01f3af3d68f304473	Running	t2.micro	2/2 checks passed	User: arn:aws:iam::137154837001:root	ap-southeast-1b
<input checked="" type="checkbox"/> 23905652-vm2	i-06bdbe3245b2d47a9	Running	t2.micro	2/2 checks passed	User: arn:aws:iam::137154837001:root	ap-southeast-1b

Use Docker inside a Linux OS

[1] Install Docker

Use the following command to install Docker:

```
sudo apt install docker.io -y
```

Enter the device password to complete the installation.

```
mayhan@DESKTOP-KEJ3P3J: $ sudo apt install docker.io -y
[sudo] password for mayhan:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 0 not upgraded.
Need to get 69.0 MB of archives.
After this operation, 284 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/universe amd64 pigz amd64 2.4-1 [57.4 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal/main amd64 bridge-utils amd64 1.6-2ubuntu1 [30.5 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 runc amd64 1.1.12-0ubuntu2~20.04.1 [8066 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 containerd amd64 1.7.12-0ubuntu2~20.04.1 [34.1 MB]
Get:5 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 dns-root-data all 2023112702~ubuntu0.20.04.1 [5308 B]
Get:6 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 dnsmasq-base amd64 2.90-0ubuntu0.20.04.1 [350 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 docker.io amd64 24.0.7-0ubuntu2~20.04.1 [26.3 MB]
Get:8 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 ubuntu-fan all 0.12.13ubuntu0.1 [34.4 kB]
Fetched 69.0 MB in 16s (4314 kB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 48810 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.4-1_amd64.deb ...
Unpacking pigz (2.4-1) ...
Selecting previously unselected package bridge-utils.
```

[2] Start Docker

Use the following command to start Docker:

```
sudo systemctl start docker
```

[3] Enable Docker

Use the following command to enable Docker:

```
sudo systemctl enable docker
```

[4] Check the version

Use the following command to check the Docker version:

```
docker --version
```

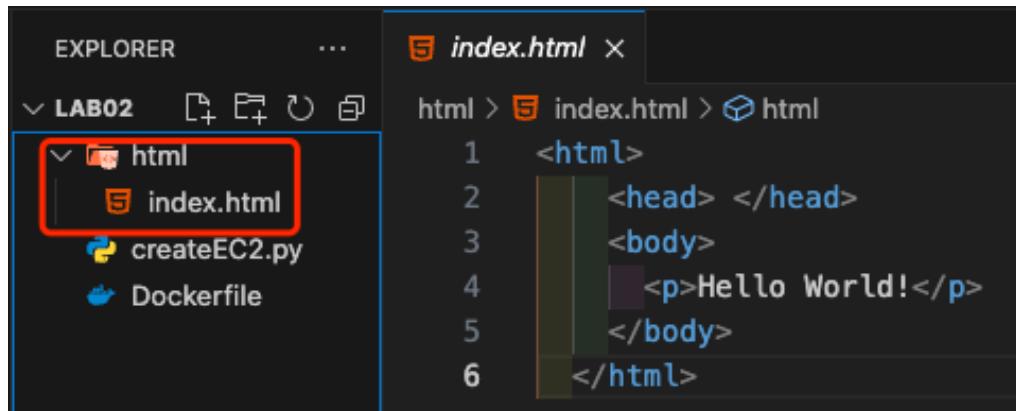
Output:

```
mayhan@DESKTOP-KEJ3P3J: $ sudo systemctl start docker
mayhan@DESKTOP-KEJ3P3J: $ sudo systemctl enable docker
mayhan@DESKTOP-KEJ3P3J: $ docker --version
Docker version 24.0.7, build 24.0.7-Ubuntu2~20.04.1
mayhan@DESKTOP-KEJ3P3J: $
```

[5] Build and run an httpd container

1. Create a directory named `html`, and create an `index.html` file inside it with the following content:

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

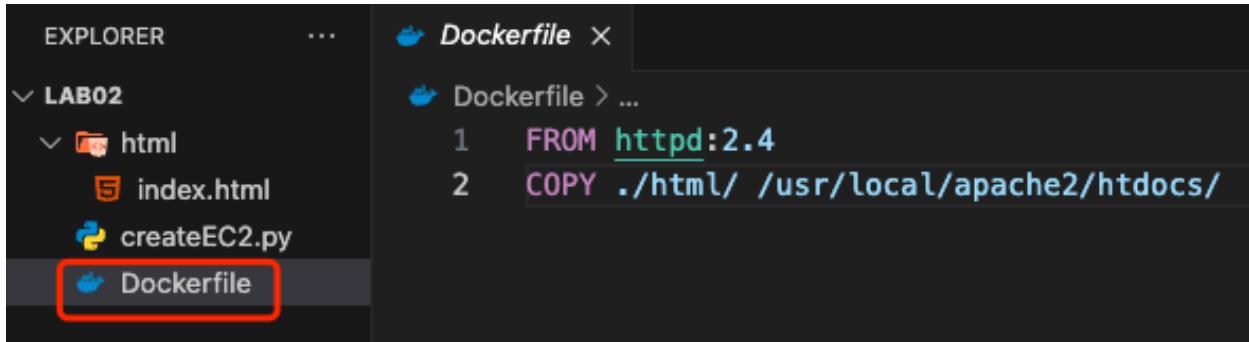


2. Create a `Dockerfile` with the following content:

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

This Dockerfile creates an image that:

- Uses **Apache HTTP Server 2.4** as its base.
- Copies custom HTML files into the appropriate directory for Apache to serve them.



3. Use the following command to build the Docker image:

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

Where:

- **docker build**: Used to build a new Docker image from a Dockerfile
- **-t my-apache2**: Tags the new image
- **.**: Looks for the Dockerfile in the current directory

Output:

```
● mayhan@DESKTOP-KEJ3P3J:~/lab$ docker build -t my-apache2 .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
              Install the buildx component to build images with BuildKit:
              https://docs.docker.com/go/buildx/
Sending build context to Docker daemon 9.216kB
Step 1/2 : FROM httpd:2.4
2.4: Pulling from library/httpd
e4fff0779e6d: Pull complete
1d0292c3dcd2: Pull complete
4f4fb700ef54: Pull complete
1316399d8fbf: Pull complete
b4cc6570db82: Pull complete
fd1a778092db: Pull complete
Digest: sha256:3f71777bcfac3df3aff5888a2d78c4104501516300b2e7ecb91ce8de2e3debc7
Status: Downloaded newer image for httpd:2.4
    --> a49fd2c04c02
Step 2/2 : COPY ./html/ /usr/local/apache2/htdocs/
    --> e66b01b90095
Successfully built e66b01b90095
Successfully tagged my-apache2:latest
● mayhan@DESKTOP-KEJ3P3J:~/lab$
```

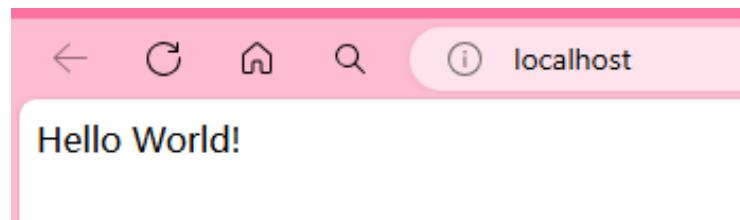
4. Use the following command to run the image and return the container ID:

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

Output:

```
● mayhan@DESKTOP-KEJ3P3J:~/lab$ docker run -p 80:80 -dit --name my-app my-apache2
b92713d25d3ead217dfc08d3d765177dd054ef27c29f4e5b21219475448f25d6
● mayhan@DESKTOP-KEJ3P3J:~/lab$
```

5. Open a browser and visit: <http://localhost>, check that it outputs "Hello World!":



[6] Other docker commands

Use the following command to check running contents:

```
docker ps -a
```

```
● mayhan@DESKTOP-KEJ3P3J:~/lab$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
b92713d25d3e        my-apache2        "httpd-foreground"   47 seconds ago    Up 46 seconds     0.0.0.0:80->80/tcp, :::80->80/tcp   my-app
● mayhan@DESKTOP-KEJ3P3J:~/lab$
```

Stop and remove the container:

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

```
● mayhan@DESKTOP-KEJ3P3J:~/lab$ docker stop my-app
my-app
● mayhan@DESKTOP-KEJ3P3J:~/lab$ docker rm my-app
my-app
```

Lab 3 S3 & DynamoDB

Program

[1] Preparation

Download the `cloudstorage.py` file from the lab sheet and save it in the “lab03” folder:

```
● mayhan@DESKTOP-KEJ3P3J:~/lab/lab03$ ls
  cloudstorage.py
○ mayhan@DESKTOP-KEJ3P3J:~/lab/lab03$ █
```

Next, we will prepare the files to be stored in S3. Create a directory `rootdir`, create a file named `rootfile.txt` inside it, and write `1\n2\n3\n4\n5\n` in it:

```
● mayhan@DESKTOP-KEJ3P3J:~/lab/lab03$ cd rootdir
● mayhan@DESKTOP-KEJ3P3J:~/lab/lab03/rootdir$ ls
  rootfile.txt  subdir
● mayhan@DESKTOP-KEJ3P3J:~/lab/lab03/rootdir$ cat rootfile.txt
○ 1\n2\n3\n4\n5\nmayhan@DESKTOP-KEJ3P3J:~/lab/lab03/rootdir$ █
```

Create a `subdir` folder inside `rootdir`, and create a file `subfile.txt` in it with the same content `1\n2\n3\n4\n5\n`:

```
● mayhan@DESKTOP-KEJ3P3J:~/lab/lab03/rootdir$ cd subdir
● mayhan@DESKTOP-KEJ3P3J:~/lab/lab03/rootdir/subdir$ ls
  subfile.txt
● mayhan@DESKTOP-KEJ3P3J:~/lab/lab03/rootdir/subdir$ cat subfile.txt
○ 1\n2\n3\n4\n5\nmayhan@DESKTOP-KEJ3P3J:~/lab/lab03/rootdir/subdir$ █
```

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

Next, we modify the downloaded file `cloudstorage.py` to create an S3 bucket named `23905652-cloudstorage`. The script is as follows, **with detailed code explanations written in the comments:**

```
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 = '.' # Set the local root directory to the current directory
ROOT_S3_DIR = '23905652-cloudstorage' # Set the S3 bucket name

s3 = boto3.client("s3") # Create an S3 client

bucket_config = {'LocationConstraint': 'ap-southeast-1'} # Set the bucket region
# configuration, replaced the region according to the student number in the table

def upload_file(folder_name, file, file_name):
    print(f"Uploading {file}")
    s3_key = os.path.join(folder_name, file_name) # Construct the S3 file key (path)
    s3.upload_file(file, ROOT_S3_DIR, s3_key) # Upload the file to S3

# Main program

# Attempt to create bucket if not there
try:
    s3.create_bucket(Bucket=ROOT_S3_DIR, CreateBucketConfiguration=bucket_config)
    print(f"Bucket {ROOT_S3_DIR} created successfully")
except Exception as error:
    print(f"Error creating bucket or bucket already exists: {error}")

# Parse directory and upload files

for dir_name, subdir_list, file_list in os.walk(ROOT_DIR, topdown=True):
    if dir_name != ROOT_DIR: # If not the root directory
        relative_dir = os.path.relpath(dir_name, ROOT_DIR) # Calculate relative path
        for fname in file_list: # Iterate through all files in the directory
            file_path = os.path.join(dir_name, fname) # Construct full file path
            upload_file(relative_dir, file_path, fname) # Call upload function

print("done")

```

Run the modified script:

```

● mayhan@DESKTOP-KEJ3P3J:~/lab/lab03$ /bin/python3 /home/mayhan/lab/lab03/cloudstorage.py
Bucket 23905652-cloudstorage created successfully
Uploading ./rootdir/rootfile.txt
Uploading ./rootdir/subdir/subfile.txt
done

```

The print result shows success.

We can go to the AWS Console, search for "S3" in the search bar, find the S3 service. On the S3 service homepage, search for "23905652" (my student number) to find the newly created S3 bucket.

Amazon S3						
▶ Account snapshot - updated every 24 hours All AWS Regions			View Storage Lens dashboard			
Storage lens provides visibility into storage usage and activity trends. Learn more						
General purpose buckets		Directory buckets				
General purpose buckets (158) Info All AWS Regions				 Copy ARN		

Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer	Creation date
12345-bucket	Asia Pacific (Tokyo) ap-northeast-1	View analyzer for ap-northeast-1	August 17, 2024, 16:55:30 (UTC+08:00)
21211711-cloudstorage	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 19, 2024, 16:33:14 (UTC+08:00)
21978612-cloudstorage	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 17, 2024, 18:34:38 (UTC+08:00)
22448064-cloudstorage	Asia Pacific (Sydney) ap-southeast-2	View analyzer for ap-southeast-2	September 13, 2024, 00:11:41 (UTC+08:00)
22595879-cloudstorage	US East (N. Virginia) us-east-1	View analyzer for us-east-1	September 13, 2024, 03:13:40 (UTC+08:00)

Find the created S3 bucket and click its name to enter the details page. Confirm that the files in the S3 bucket have the same structure as shown in [1] Preparation :

23905652-cloudstorage Info				
Objects Properties Permissions Metrics Management Access Points				
Objects (1) Info				
	 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				
<input type="text"/> Find objects by prefix				
Name	Type	Last modified	Size	Storage class
rootdir/	Folder	-	-	-
rootdir/ 				
Objects Properties				
Objects (2) Info				
	 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				
<input type="text"/> Find objects by prefix				
Name	Type	Last modified	Size	Storage class
rootfile.txt	txt	August 31, 2024, 15:28:07 (UTC+08:00)	15.0 B	Standard
subdir/	Folder	-	-	-

subdir/

Copy S3 URI

Objects Properties

Objects (1) [Info](#)



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

< 1 >

Name

▲

Type

▼

Last modified

▼

Size

▼

Storage class

▼

[subfile.txt](#)

txt

August 31, 2024, 15:28:08
(UTC+08:00)

15.0 B Standard

[3] Restore from S3

Create a new script to read from the S3 bucket and download its contents, **with detailed code explanations provided in the comments:**

```
import os
import boto3

# Description: This script downloads all files from an S3 bucket to the local directory.

# Set the local and S3 root directories
ROOT_DIR = '.' # Set the local root directory to the current directory
ROOT_S3_DIR = '23905652-cloudstorage'
REGION = 'ap-southeast-1'

# Configure the S3 bucket location
bucket_config = {'LocationConstraint': REGION}

# Initialize the S3 client
s3 = boto3.client("s3", region_name=REGION)

# Define a function to download a file from S3
def download_file(s3_key, local_path):
    # Create the local directory structure if it doesn't exist
    os.makedirs(os.path.dirname(local_path), exist_ok=True)

    # Download the file from S3 to the local path
    s3.download_file(ROOT_S3_DIR, s3_key, local_path)

    print(f"Downloaded: {s3_key}")

# List objects in the S3 bucket and download them
response = s3.list_objects(Bucket=ROOT_S3_DIR) # Get a list of all objects in the bucket

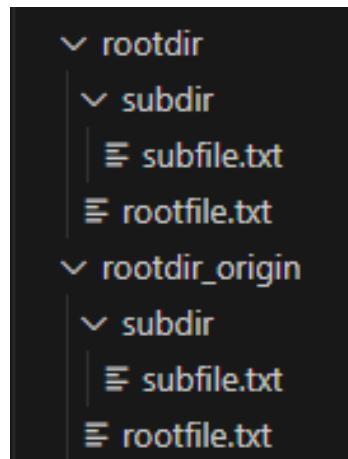
# Iterate through each object in the bucket
for obj in response.get('Contents', []):
    s3_key = obj['Key'] # Get the S3 key (path) of the object
    local_path = os.path.join(ROOT_DIR, s3_key) # Construct the local path for the file
    download_file(s3_key, local_path) # Call the download function for each file
```

```
print("Download complete")
```

Save the script with the name `restorefromcloud.py`. To check if the download was successful, we can rename the original `rootdir` to `rootdir_origin`, then **run the code and obtain**:

```
done
● mayhan@DESKTOP-KEJ3P3J:~/lab/lab03$ /bin/python3 /home/mayhan/lab/lab
03/restorefromcloud.py
Downloaded: rootdir/rootfile.txt
Downloaded: rootdir/subdir/subfile.txt
Download complete
```

We can then see the newly downloaded files **in local file system**, and the structure of the downloaded files matches the original:



[4] Write information about files to DynamoDB

1. Run the following commands to install DynamoDB and JRE:

```
mkdir dynamodb
cd dynamodb
sudo apt-get install default-jre
wget https://s3-ap-northeast-1.amazonaws.com/dynamodb-local-
tokyo/dynamodb_local_latest.tar.gz
```

Output:

```
● mayhan@DESKTOP-KEJ3P3J:~$ mkdir dynamodb
● mayhan@DESKTOP-KEJ3P3J:~$ cd dynamodb
● mayhan@DESKTOP-KEJ3P3J:~/dynamodb$ sudo apt-get install default-jre
[sudo] password for mayhan:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
at-spi2-core ca-certificates-java default-jre-headless
fonts-dejavu-extra java-common libatk-bridge2.0-0
libatk-wrapper-java libatk-wrapper-java-ini libatk1.0-0
```

```
mayhan@DESKTOP-KEJ3P3J:~/dynamodb$ wget https://s3-ap-northeast-1.amazonaws.com/dynamodb-local-tokyo/dynamodb_local_latest.tar.gz
--2024-08-31 16:00:08-- 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)... 104.244.46.186, 2a03:2880:f10a:83:face:b00c:0:25de
Connecting to s3-ap-northeast-1.amazonaws.com (s3-ap-northeast-1.amazonaws.com)|104.244.46.186|:443... failed: Connection timed out.
Connecting to s3-ap-northeast-1.amazonaws.com (s3-ap-northeast-1.amazonaws.com)|2a03:2880:f10a:83:face:b00c:0:25de|:443... failed: Network is unreachable.
```

It was found that there was an issue with the "ap-northeast-1" network connection, which prevented the successful download of the DynamoDB local version's compressed file. Thus, we tried downloading from the "us-west-2" region and succeeded:

```
aws.com)|173.250.182.137|:443...
● mayhan@DESKTOP-KEJ3P3J:~/dynamodb$ wget https://s3.us-west-2.amazonaws.com/dynamodb-local/dynamodb_local_latest.tar.gz
--2024-08-31 16:14:08-- https://s3.us-west-2.amazonaws.com/dynamodb-local/dynamodb_local_latest.tar.gz
Resolving s3.us-west-2.amazonaws.com (s3.us-west-2.amazonaws.com)... 52.92.204.208, 52.218.177.56, 52.92.152.136, ...
Connecting to s3.us-west-2.amazonaws.com (s3.us-west-2.amazonaws.com)|52.92.204.208|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 54892500 (52M) [application/x-tar]
Saving to: 'dynamodb_local_latest.tar.gz'

dynamodb_local_la 100%[=====>] 52.35M 4.50MB/s in 13s

2024-08-31 16:14:23 (4.03 MB/s) - 'dynamodb_local_latest.tar.gz' saved [54892500/54892500]
```

2. Use the following command to unzip the compressed file:

```
tar -zxvf dynamodb_local_latest.tar.gz
```

Output:

```
● mayhan@DESKTOP-KEJ3P3J:~/dynamodb$ tar -zxvf dynamodb_local_latest.tar.gz
DynamoDBLocal_lib/
DynamoDBLocal.jar
DynamoDBLocal_lib/Apache-HttpComponents-HttpClient-4.5.x.jar
DynamoDBLocal_lib/Apache-HttpComponents-HttpCore-4.4.x.jar
DynamoDBLocal_lib/AwsFlowJava-1.0.jar
DynamoDBLocal_lib/AwsJavaSdk-CognitoIdentity-2.0.jar
DynamoDBLocal_lib/AwsJavaSdk-CognitoIdentityProvider-2.0.jar
DynamoDBLocal_lib/AwsJavaSdk-Core-2.0.jar
DynamoDBLocal_lib/AwsJavaSdk-Core-Annotations-2.0.jar
DynamoDBLocal_lib/AwsJavaSdk-Core-Auth-2.0.jar
DynamoDBLocal_lib/AwsJavaSdk-Core-AwsCore-2.0.jar
```

3. Use the following command to start the local version of DynamoDB:

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

Output:

```
mayhan@DESKTOP-KEJ3P3J:~/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: false
shouldDelayTransientStatuses: false
CorsParams: null

[]
```

4. Next, we have to use the AWS CLI command to create a new table `CloudFiles`, with the table attributes:

```
CloudFiles = {
    'userId',
    'fileName',
    'path',
    'lastUpdated',
    'owner',
    'permissions'
}
)
```

The `userId` is the partition key and `fileName` is the sort key.

Here is the AWS CLI command used:

```
○ mayhan@DESKTOP-KEJ3P3J:~/lab/lab03$ aws dynamodb create-table \
>   --table-name CloudFiles \
>   --attribute-definitions \
>     AttributeName=userId,AttributeType=S \
>     AttributeName=fileName,AttributeType=S \
>   --key-schema \
>    AttributeName=userId,KeyType=HASH \
>    AttributeName=fileName,KeyType=RANGE \
>   --provisioned-throughput \
>     ReadCapacityUnits=5,WriteCapacityUnits=5 \
>   --endpoint-url http://localhost:8000 \
>   --region ap-southeast-1
```

The response received after running the code is as follows:

```

{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "userId",
                "AttributeType": "S"
            },
            {
                "AttributeName": "fileName",
                "AttributeType": "S"
            }
        ],
        "TableName": "CloudFiles",
        "KeySchema": [
            {
                "AttributeName": "userId",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "fileName",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "ACTIVE",
        "CreationDateTime": 1725094086.435,
        "ProvisionedThroughput": {
            "LastIncreaseDateTime": 0.0,
            "LastDecreaseDateTime": 0.0,
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 5,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 0,
        "ItemCount": 0,
        "TableArn": "arn:aws:dynamodb:ddblocal:000000000000:table/CloudFiles",
        "DeletionProtectionEnabled": false
    }
}

```

The returned result displays detailed information about the created table.

- To write the attributes of all files in S3 to the created table, we created the following script, **with detailed code explanations provided in the comments:**

```

import boto3
from botocore.exceptions import ClientError

# Constants
S3_BUCKET_NAME = '23905652-cloudstorage' # Name of the S3 bucket to process
DYNAMODB_TABLE_NAME = 'CloudFiles' # Name of the DynamoDB table to write to
REGION = 'ap-southeast-1' # AWS region for the services

# Initialize clients
s3 = boto3.client('s3', region_name=REGION)
# Create a DynamoDB resource for interacting with the DynamoDB service
# Using a local DynamoDB instance for development/testing

```

```

dynamodb = boto3.resource('dynamodb', endpoint_url='http://localhost:8000',
region_name=REGION)

def get_file_attributes(bucket, key):
    """
    Retrieve attributes of a file from S3.

    Args:
        bucket (str): The name of the S3 bucket.
        key (str): The key (path) of the file in the bucket.

    Returns:
        dict: A dictionary containing file attributes, or None if an error occurs.
    """
    try:
        # Get the metadata of the object from S3
        response = s3.head_object(Bucket=bucket, Key=key)
        return {
            'fileName': key,
            'path': f"s3://{bucket}/{key}",
            'lastUpdated': response['LastModified'].isoformat()
        }
    except ClientError as e:
        print(f"Error getting attributes for {key}: {e}")
        return None

def write_to_dynamodb(table, user_id, owner, file_info):
    """
    Write file information to a DynamoDB table.

    Args:
        table (boto3.resources.factory.dynamodb.Table): The DynamoDB table resource.
        user_id (str): The ID of the user who owns the file.
        owner (str): The display name of the file owner.
        file_info (dict): Information about the file to be written.
    """
    try:
        # Prepare the item to be written to DynamoDB
        item = {
            'userId': user_id,
            'owner': owner,
            'permissions': s3.get_bucket_acl(Bucket=S3_BUCKET_NAME)[ 'Grants' ][0]
[ 'Permission' ],
            **file_info
        }
        # Write the item to the DynamoDB table
        table.put_item(Item=item)
        print(f"Added to DynamoDB: {file_info['fileName']} ")
    except ClientError as e:
        print(f"Error writing to DynamoDB: {e}")

def main():

```

```

"""
Main function to process S3 objects and write their information to DynamoDB.
"""

# Get the DynamoDB table resource
table = dynamodb.Table(DYNAMODB_TABLE_NAME)

try:
    # List objects in the S3 bucket
    response = s3.list_objects(Bucket=S3_BUCKET_NAME)

    # Extract user ID and owner name from the first object
    user_id = str(response['Contents'][0]['Owner']['ID'])
    owner = response['Contents'][0]['Owner']['DisplayName']

    # Process each object in the bucket
    for obj in response.get('Contents', []):
        # Get file attributes
        file_attributes = get_file_attributes(S3_BUCKET_NAME, obj['Key'])
        if file_attributes:
            # Write file information to DynamoDB
            write_to_dynamodb(table, user_id, owner, file_attributes)

    print("Process completed. All files have been processed.")

except ClientError as e:
    print(f"Error listing S3 objects: {e}")

if __name__ == "__main__":
    main()

```

Run the code and obtained:

```

}
● mayhan@DESKTOP-KEJ3P3J:~/lab/lab03$ python3 addDataToDB.py
Added to DynamoDB: rootdir/rootfile.txt
Added to DynamoDB: rootdir/subdir/subfile.txt
Process completed. All files have been processed.

```

[5] Scan the table

Use the following AWS CLI command to scan the created DynamoDB table and check if the table contains information about the S3 files:

```

Process completed. All files have been processed.
● mayhan@DESKTOP-KEJ3P3J:~/lab/lab03$ aws dynamodb scan \
>   --table-name CloudFiles \
>   --endpoint-url http://localhost:8000

```

Output result:

```
● mayhan@DESKTOP-KEJ3P3J:~/lab/lab03$ aws dynamodb scan \
>     --table-name CloudFiles \
>     --endpoint-url http://localhost:8000
{
    "Items": [
        {
            "owner": {
                "S": "zhi.zhang"
            },
            "path": {
                "S": "s3://23905652-cloudstorage/rootdir/rootfile.txt"
            },
            "lastUpdated": {
                "S": "2024-08-31T07:28:07+00:00"
            },
            "fileName": {
                "S": "rootdir/rootfile.txt"
            },
            "userId": {
                "S": "2a5fac7aada1ad2caa48c9ab08cc4e2428d4eb596108daa3b59f1204ae96482e"
            },
            "permissions": {
                "S": "FULL_CONTROL"
            }
        },
        {
            "owner": {
                "S": "zhi.zhang"
            },
            "path": {
                "S": "s3://23905652-cloudstorage/rootdir/subdir/subfile.txt"
            },
            "lastUpdated": {
                "S": "2024-08-31T07:28:08+00:00"
            },
            "fileName": {
                "S": "rootdir/subdir/subfile.txt"
            },
            "userId": {
                "S": "2a5fac7aada1ad2caa48c9ab08cc4e2428d4eb596108daa3b59f1204ae96482e"
            },
            "permissions": {
                "S": "FULL_CONTROL"
            }
        }
    ],
    "Count": 2,
    "ScannedCount": 2,
    "ConsumedCapacity": null
}
```

We can see that the information for the two files has been added to the table.

[6] Delete the table

1. Use the following AWS CLI command to delete the table:

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

Output:

```
● mayhan@DESKTOP-KEJ3P3J:~/lab/lab03$ aws dynamodb delete-table \
>   --table-name CloudFiles \
>   --endpoint-url http://localhost:8000
{
    "TableDescription": {
        "AttributeDefinitions": [
            {
                "AttributeName": "userId",
                "AttributeType": "S"
            },
            {
                "AttributeName": "fileName",
                "AttributeType": "S"
            }
        ],
        "TableName": "CloudFiles",
        "KeySchema": [
            {
                "AttributeName": "userId",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "fileName",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "ACTIVE",
        "CreationDateTime": 1725094086.435,
        "ProvisionedThroughput": {
            "LastIncreaseDateTime": 0.0,
            "LastDecreaseDateTime": 0.0,
            "NumberOfDecreasesToday": 0,
            "ReadCapacityUnits": 5,
            "WriteCapacityUnits": 5
        },
        "TableSizeBytes": 456,
        "ItemCount": 2,
        "TableArn": "arn:aws:dynamodb:ddblocal:000000000000:table/CloudFiles",
        "DeletionProtectionEnabled": false
    }
}
```

To confirm that the table has been deleted, we can run the following command:

```
● mayhan@DESKTOP-KEJ3P3J:~/lab/lab03$ aws dynamodb list-tables --endpoint-url http://localhost:8000
{
    "TableNames": []
}
○ mayhan@DESKTOP-KEJ3P3J:~/lab/lab03$
```

The table list has been cleared, indicating that the created table has been successfully deleted.

- Empty and delete the created S3 bucket from the AWS console:

Empty bucket Info



- Emptying the bucket deletes all objects in the bucket and cannot be undone.
- Objects added to the bucket while the empty bucket action is in progress might be deleted.
- To prevent new objects from being added to this bucket while the empty bucket action is in progress, you might need to update your bucket policy to stop objects from being added to the bucket.

[Learn more](#)



If your bucket contains a large number of objects, creating a lifecycle rule to delete all objects in the bucket might be a more efficient way of emptying your bucket. [Learn more](#)

[Go to lifecycle rule configuration](#)

Permanently delete all objects in bucket "23905652-cloudstorage"?

To confirm deletion, type *permanently delete* in the text input field.

permanently delete

[Cancel](#)

[Empty](#)

Delete bucket Info



- Deleting a bucket cannot be undone.
- Bucket names are unique. If you delete a bucket, another AWS user can use the name.
- If this bucket is used with a Multi-Region Access Point in an external account, initiate failover before deleting the bucket.
- If this bucket is used with an access point in an external account, the requests made through those access points will fail after you delete this bucket.

[Learn more](#)

Delete bucket "23905652-cloudstorage"?

To confirm deletion, enter the name of the bucket in the text input field.

[Cancel](#)

[Delete bucket](#)

Lab 4 Encryption

Apply a policy to restrict permissions on bucket

[1] Write a Python script

To apply the following policy to the S3 bucket to **only allow access to my username**, modifications were made based on the code from Lab 3 that created the S3 bucket.

Policy:

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Sid": "AllowAllS3ActionsInUserFolderForUserOnly",  
        "Effect": "DENY",  
        "Principal": "*",  
        "Action": "s3:*",  
        "Resource": "arn:aws:s3:::23905652-cloudstorage/folder1/folder2/*",  
        "Condition": {  
            "StringNotLike": {  
                "aws:username": "23905652@student.uwa.edu.au"  
            }  
        }  
    }]  
}
```

Add the `apply_bucket_policy()` function to apply the policy:

```
ROOT_S3_DIR = '23905652-cloudstorage'  
  
def apply_bucket_policy():  
    """  
        Apply a bucket policy to restrict access to the S3 bucket.  
        This policy denies all S3 actions to everyone except '23905652@student.uwa.edu.au'.  
    """  
  
    # Define the bucket policy  
    policy = {  
        "Version": "2012-10-17", # Policy language version  
        "Statement": [{  
            "Sid": "AllowAllS3ActionsInUserFolderForUserOnly", # Statement identifier  
            "Effect": "DENY", # This policy will deny access  
            "Principal": "*", # Applies to all AWS users  
            "Action": "s3:*", # Applies to all S3 actions  
            "Resource": [ # The AWS resources this policy applies to  
                f"arn:aws:s3:::{ROOT_S3_DIR}/folder1/folder2/*"  
            ],  
            "Condition": { # Conditions for when this policy is in effect  
                "StringNotLike": { # Applies when the following is not true  
                    "aws:username": "23905652@student.uwa.edu.au"  
                }  
            }  
        }]  
    }  
    return policy
```

```

        "aws:username": "23905652@student.uwa.edu.au" # Specific user exempt
from this policy
    }
}
}

# Convert the policy dictionary to a JSON-formatted string
policy_json = json.dumps(policy)

try:
    # Attempt to apply the policy to the S3 bucket
    s3.put_bucket_policy(Bucket=ROOT_S3_DIR, Policy=policy_json)
    print(f"Policy successfully applied to bucket {ROOT_S3_DIR}")
except Exception as e:
    # If an error occurs, print the error message
    print(f"Error applying policy: {str(e)}")

```

Keep the other code for creating S3 and uploading files same. After uploading the files, call the `apply_bucket_policy()` function:

```

...
# Same as before, parse directory and upload files
for dir_name, subdir_list, file_list in os.walk(ROOT_DIR, topdown=True):
    if dir_name != ROOT_DIR:
        relative_dir = os.path.relpath(dir_name, ROOT_DIR)
        for fname in file_list:
            file_path = os.path.join(relative_dir, fname)
            upload_file(relative_dir, file_path, fname)

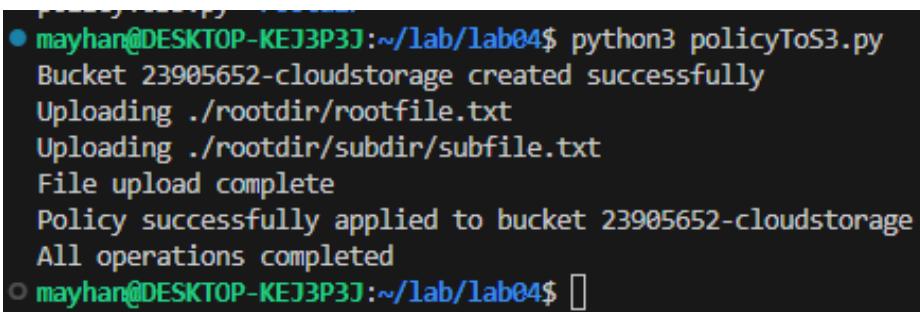
print("File upload complete")

# Apply bucket policy
apply_bucket_policy()

print("All operations completed")

```

Save the new script as 'policyToS3.py' and **run the script**:



```

● mayhan@DESKTOP-KEJ3P3J:~/lab/lab04$ python3 policyToS3.py
Bucket 23905652-cloudstorage created successfully
Uploading ./rootdir/rootfile.txt
Uploading ./rootdir/subdir/subfile.txt
File upload complete
Policy successfully applied to bucket 23905652-cloudstorage
All operations completed
○ mayhan@DESKTOP-KEJ3P3J:~/lab/lab04$ []

```

[2] Check whether the script works

1. Use the AWS CLI command to display the policy content:

```
aws s3api get-bucket-policy --bucket 23905652-cloudstorage
```

```
All operations completed
● mayhan@DESKTOP-KFJ3P3:~/lab/lab04$ aws s3api get-bucket-policy --bucket 23905652-cloudstorage
{
  "Policy": "{\"Version\":\"2012-10-17\"},\"Statement\":[{\"Sid\":\"AllowAllS3ActionsInUserFolderForUserOnly\",\"Effect\":\"Deny\",\"Principal\":\"*\",\"Action\":\"s3:*\",\"Resource\":\"arn:aws:s3:::23905652-cloudstorage/folder1/folder2/*\"},\"Condition\":{\"StringNotLike\":{\"aws:username\":\"23905652@student.uwa.edu.au\"}}]}"
}
● mayhan@DESKTOP-KFJ3P3:~/lab/lab04$
```

The response is as follows, **matching the expected policy/script policy**:

```
{
  "Policy": "{\"Version\":\"2012-10-17\"},\"Statement\":[
    {
      "Sid": "AllowAllS3ActionsInUserFolderForUserOnly",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::23905652-cloudstorage/folder1/folder2/*",
      "Condition": {
        "StringNotLike": {
          "aws:username": "23905652@student.uwa.edu.au"
        }
      }
    }
  ]
}
```

2. Open the AWS S3 console to view the policy. In the S3 details page, select "**Permissions**":

[Amazon S3](#) > [Buckets](#) > 23905652-cloudstorage

23905652-cloudstorage [Info](#)

[Objects](#) | [Properties](#) | [Permissions](#) | [Metrics](#) | [Management](#) | [Access Points](#)

Scroll down to find "**Bucket policy**":

Bucket policy

[Edit](#)[Delete](#)

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:::23905652-cloudstorage/folder1/folder2/*",  
      "Condition": {  
        "StringNotLike": {  
          "aws:username": "23905652@student.uwa.edu.au"  
        }  
      }  
    }  
  ]  
}
```

[Copy](#)

The policy has been successfully applied to the bucket.

3. To test the effectiveness of the policy in AWS Console, we extended the "**Resource**" covered by the policy to the entire bucket and set the "**Condition**" to a strict "**StringNotEquals**:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "AllowAllS3ActionsInUserFolderForUserOnly",  
      "Effect": "Deny",  
      "Principal": "*",  
      "Action": "s3:",  
      "Resource": [  
        "arn:aws:s3:::23905652-cloudstorage/*",  
        "arn:aws:s3:::23905652-cloudstorage"  
      ],  
      "Condition": {  
        "StringNotEquals": {  
          "aws:username": "23905652@student.uwa.edu.au"  
        }  
      }  
    }  
  ]  
}
```

Attempting to access the bucket with another username (23254894@student.uwa.edu.au) shows the permission restrictions:

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with 'Amazon S3' selected. Under 'Buckets', it lists 'Access Grants', 'Access Points', 'Object Lambda Access Points', 'Multi-Region Access Points', 'Batch Operations', and 'IAM Access Analyzer for S3'. Below that is a section for 'Block Public Access settings for this account'. Under 'Storage Lens', it lists 'Dashboards', 'Storage Lens groups', and 'AWS Organizations settings'. There's also a 'Feature spotlight' section with a '7' badge. At the bottom of the sidebar, it says '▶ AWS Marketplace for S3'. The main content area shows the '23905652-cloudstorage' bucket. It has tabs for 'Objects', 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'. Under 'Objects', there are buttons for 'Actions' (with a dropdown), 'Create folder', and 'Upload'. A search bar says 'Find objects by prefix'. Below that is a 'Show versions' toggle. A table header includes columns for 'Name', 'Type', 'Last modified', 'Size', and 'Storage class'. A prominent red box highlights a warning message: 'Insufficient permissions to list objects. After you or your AWS administrator has updated your permissions to allow the s3>ListBucket action, refresh the page. Learn more about Identity and access management in Amazon S3'.

AES Encryption using KMS

[1] Create a KMS key

To create a KMS key, a new Python script was written, where the alias for the key is my student number (23905652). The script is attached in the next step combined with the process of attaching the policy .

[2] Attach a policy to the created KMS key

The Python script provided includes the creation of a KMS key and attaching a specified policy to the key. It includes two functions: `create_kms_key()` and `attach_policy_to_key()`. **Detailed explanations are included in the comments of the code:**

```

import boto3
import json

# Initialize the KMS client, this creates a client for interacting with AWS Key Management
# Service in the specified region
kms_client = boto3.client('kms', region_name='ap-southeast-1')

# Student number will be used to create an alias for the KMS key
STUDENT_NUMBER = '23905652_2'

# This is the AWS Identity and Access Management username that will be granted permissions
# to the key

```

```

IAM_USERNAME = '23905652@student.uwa.edu.au'

def create_kms_key():
    """
    Creates a new KMS key and an alias for it.

    Returns:
        str: The ID of the created KMS key
    """

    # Create a new KMS key
    response = kms_client.create_key(
        Description=f'KMS key for student {STUDENT_NUMBER}', # Description of the key
        KeyUsage='ENCRYPT_DECRYPT', # Specifies that this key will be used for encryption
        and decryption
        Origin='AWS_KMS' # Indicates that AWS KMS creates and manages the key material
    )

    # Extract the key ID from the response
    key_id = response['KeyMetadata']['KeyId']

    # Create an alias for the key, an alias is a user-friendly name for the key
    kms_client.create_alias(
        AliasName=f'alias/{STUDENT_NUMBER}', # The alias name, prefixed with 'alias/'
        TargetKeyId=key_id # The ID of the key to which this alias refers
    )

    print(f"KMS key created with ID: {key_id}")
    return key_id

def attach_policy_to_key(key_id):
    """
    Attaches a policy to the specified KMS key.

    Args:
        key_id (str): The ID of the KMS key to which the policy will be attached
    """

    # Define the policy, this policy defines who can administer and use the key
    policy = {
        "Version": "2012-10-17",
        "Id": "key-consolepolicy-3",
        "Statement": [
            {
                # Allows the root user full access to the key
                "Sid": "Enable IAM User Permissions",
                "Effect": "Allow",
                "Principal": {
                    "AWS": "arn:aws:iam::489389878001:root"
                },
                "Action": "kms:*",
                "Resource": "*"
            },
        ]
    }

```

```
{
    # Allows the specified IAM user to administer the key
    "Sid": "Allow access for Key Administrators",
    "Effect": "Allow",
    "Principal": {
        "AWS": f"arn:aws:iam::489389878001:user/{IAM_USERNAME}"
    },
    "Action": [
        # List of administrative actions allowed
        "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": "*"
},
{
    # Allows the specified IAM user to use the key for cryptographic
operations
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": f"arn:aws:iam::489389878001:user/{IAM_USERNAME}"
    },
    "Action": [
        # List of cryptographic operations allowed
        "kms>Encrypt", "kms>Decrypt", "kms>ReEncrypt*",
        "kms>GenerateDataKey*", "kms>DescribeKey"
    ],
    "Resource": "*"
},
{
    # Allows the specified IAM user to manage grants on the key
    "Sid": "Allow attachment of persistent resources",
    "Effect": "Allow",
    "Principal": {
        "AWS": f"arn:aws:iam::489389878001:user/{IAM_USERNAME}"
    },
    "Action": [
        "kms>CreateGrant", "kms>ListGrants", "kms>RevokeGrant"
    ],
    "Resource": "*",
    "Condition": {
        "Bool": {
            "kms>GrantIsForAWSResource": "true"
        }
    }
}
]
```

```

# Attach the policy to the key
kms_client.put_key_policy(
    KeyId=key_id,
    PolicyName='default', # The name of the policy
    Policy=json.dumps(policy) # The policy document, converted to a JSON string
)

print(f"Policy attached to key {key_id}")

def main():
    """
    Main function to create a KMS key and attach a policy to it.
    """
    key_id = create_kms_key()
    attach_policy_to_key(key_id)
    print("KMS key creation and policy attachment completed successfully.")

if __name__ == "__main__":
    main()

```

Save the script as "createKMS.py" and run the code:

```

● mayhan@DESKTOP-KEJ3P3J:~/lab/lab04$ python3 createKMS.py
KMS key created with ID: d7e6d2e8-140d-4529-b8cf-4277e5c1a567
Policy attached to key d7e6d2e8-140d-4529-b8cf-4277e5c1a567
KMS key creation and policy attachment completed successfully.
○ mayhan@DESKTOP-KEJ3P3J:~/lab/lab04$ █

```

[3] Check whether the script works

Open the AWS KMS Console and check the policy of the created KMS:

Key administrator:

d7e6d2e8-140d-4529-b8cf-4277e5c1a567

Key actions ▾ Edit

General configuration

Alias 23905652_2	Status Enabled	Creation date Sep 13, 2024 21:06 GMT+8
ARN <code>arn:aws:kms:ap-southeast-1:489389878001:key/d7e6d2e8-140d-4529-b8cf-4277e5c1a567</code>	Description KMS key for student 23905652_2	Regionality Single Region

Key policy Cryptographic configuration Tags Key rotation Aliases

Key policy

Edit

```
"AWS": "arn:aws:iam::489389878001:root"
},
"Action": "kms:*",
"Resource": "*"
},
{
"Sid": "Allow access for Key Administrators",
"Effect": "Allow",
"Principal": {
    "AWS": "arn:aws:iam::489389878001:user/23905652@student.uwa.edu.au"
},
"Action": [
    "kms>Create*",
    "kms<*>*<*>*"
]
```

It can be seen that **the Key Administrator is myself**.

Key user:

d7e6d2e8-140d-4529-b8cf-4277e5c1a567

Key actions ▾ Edit

General configuration

Alias 23905652_2	Status Enabled	Creation date Sep 13, 2024 21:06 GMT+8
ARN <code>arn:aws:kms:ap-southeast-1:489389878001:key/d7e6d2e8-140d-4529-b8cf-4277e5c1a567</code>	Description KMS key for student 23905652_2	Regionality Single Region

Key policy Cryptographic configuration Tags Key rotation Aliases

Key policy

Edit

```
Resource : ...
},
{
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::489389878001:user/23905652@student.uwa.edu.au"
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ]
}
```

It can be seen that **the Key User is myself only**, too.

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

In this step, we handle encryption and decryption separately, starting with encryption.

1. Encryption Script:

The following script uses the KMS key to encrypt S3 files, defining two functions: `get_kms_key_id()` and `encrypt_file()`:

```
import boto3
import os

# Initialize the S3 and KMS clients
# These clients allow interaction with AWS S3 (Simple Storage Service) and KMS (Key Management Service)
s3 = boto3.client("s3")
kms = boto3.client('kms')

# S3 bucket name and KMS key alias
# These constants define the S3 bucket to work with and the alias of the KMS key to use
# for encryption
BUCKET_NAME = '23905652-cloudstorage'
KEY_ALIAS = 'alias/23905652_2'

def get_kms_key_id():
    """
    Retrieve the KMS key ID associated with the specified alias.

    Returns:
        str: The ID of the KMS key
    Raises:
        Exception: If the KMS key with the specified alias is not found
    """
    response = kms.list_aliases()
    for alias in response['Aliases']:
        if alias['AliasName'] == KEY_ALIAS:
            return alias['TargetKeyId']
    raise Exception(f"KMS key with alias {KEY_ALIAS} not found")

def encrypt_file(file_key):
    """
    Download a file from S3, encrypt it using the KMS key, and upload the encrypted
    version back to S3.

    Args:
        file_key (str): The key (path) of the file in the S3 bucket
    """
    # Download the file from S3 to the local filesystem
    local_file = os.path.basename(file_key)
    s3.download_file(BUCKET_NAME, file_key, local_file)

    # Read the contents of the downloaded file
    with open(local_file, 'rb') as file:
        file_contents = file.read()

    # Encrypt the file contents using the KMS key
    response = kms.encrypt(KeyId=key_id, Plaintext=file_contents)
    encrypted_contents = response['CiphertextBlob']
```

```

# Write the encrypted contents to a new local file
encrypted_file = f"{local_file}.encrypted"
with open(encrypted_file, 'wb') as file:
    file.write(encrypted_contents)

# Upload the encrypted file back to S3 with a '.encrypted' suffix
s3.upload_file(encrypted_file, BUCKET_NAME, f"{file_key}.encrypted")

print(f"File {file_key} encrypted successfully")

def main():
    """
    Main function to encrypt all non-encrypted files in the specified S3 bucket.
    """
    global key_id
    # Retrieve the KMS key ID
    key_id = get_kms_key_id()

    # List all objects in the S3 bucket
    response = s3.list_objects_v2(Bucket=BUCKET_NAME)

    # Iterate through each object in the bucket
    for obj in response.get('Contents', []):
        file_key = obj['Key']
        # Check if the file is not already encrypted (doesn't end with '.encrypted')
        if not file_key.endswith('.encrypted'):
            encrypt_file(file_key)

if __name__ == "__main__":
    main()

```

Save the script as "encryptByKMS.py" and run the code:

```

● mayhan@DESKTOP-KEJ3P3J:~/lab/lab04$ python3 encryptByKMS.py
  File rootdir/rootfile.txt encrypted successfully
  File rootdir/subdir/subfile.txt encrypted successfully
○ mayhan@DESKTOP-KEJ3P3J:~/lab/lab04$ █

```

Returning to the **AWS S3 console**, we can see the encrypted file:

rootdir/

[Copy S3 URI](#)[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

< 1 >

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	rootfile.txt	txt	September 13, 2024, 19:42:50 (UTC+08:00)	15.0 B	Standard
<input type="checkbox"/>	rootfile.txt.encrypted	encrypted	September 13, 2024, 21:42:38 (UTC+08:00)	167.0 B	Standard
<input type="checkbox"/>	subdir/	Folder	-	-	-

subdir/

[Copy S3 URI](#)[Objects](#) [Properties](#)**Objects (2) 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

< 1 >

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	subfile.txt	txt	September 13, 2024, 19:42:50 (UTC+08:00)	15.0 B	Standard
<input type="checkbox"/>	subfile.txt.encrypted	encrypted	September 13, 2024, 21:42:39 (UTC+08:00)	167.0 B	Standard

Downloading the encrypted file and opening it reveals the ciphertext:

```
≡ rootfile.txt.encrypted ×
lab04 > ≡ rootfile.txt.encrypted
1 SOH$STXNULX^Q Bs ?C?yQ` ?CAN? ?jN?" ???
2 3?h?7m)? ?S? ?S? ?%Y0? ?h/C NULNULNULm0ACK *?H??
3 SOHBELACK?^0\STXSOHNUL0WACK *?H??
4 SOHBELSOH0 RS ACK `?H$OHETEXEOTSOH .0CLEFT FF ? ?SOH5? ?@S$? ETB$T$SOHOLE?*(f'!C5Id\?>FS ?`?2??ETX$RS ? ?DC3 VT ETB HSYN-?J?6z#??

```

```
≡ subfile.txt.encrypted ×
lab04 > ≡ subfile.txt.encrypted
1 SOH$STXNULX^Q Bs ?C?yQ` ?CAN? ?jN?" ???
2 3?h?7m)? ?S? ?S? ?I SO /| GS DDEA?#? NULNULNULm0ACK *?H??
3 SOHBELACK?^0\STXSOHNUL0WACK *?H??
4 SOHBELSOH0 RS ACK `?H$OHETEXEOTSOH .0CLEFT FF N ? ?i: ? ?kSTXSOHOLE?*?9? ?@ ,W? -@N?>?)Z? ?us { M$TXV? CS EOTA? EM? X%? #H FF %? CS ? SI
```

2. Decryption Script:

For the decryption process, we add two more functions `decrypt_file(file_key)` and `compare_files()`:

```
def decrypt_file(file_key):
    """
        Download an encrypted file from S3, decrypt it using the KMS key, and upload the
        decrypted version back to S3.
    
```

```

Args:
    file_key (str): The key (path) of the encrypted file in the S3 bucket
"""

# Download the encrypted file from S3 to the local filesystem
local_file = os.path.basename(file_key)
s3.download_file(BUCKET_NAME, file_key, local_file)

# Read the contents of the downloaded encrypted file
with open(local_file, 'rb') as file:
    encrypted_contents = file.read()

# Decrypt the file contents using the KMS key
response = kms.decrypt(CiphertextBlob=encrypted_contents, KeyId=key_id)
decrypted_contents = response['Plaintext']

# Write the decrypted contents to a new local file
# The new filename replaces '.encrypted' with '.decrypted'
decrypted_file = local_file.replace('.encrypted', '.decrypted')
with open(decrypted_file, 'wb') as file:
    file.write(decrypted_contents)

# Upload the decrypted file back to S3
# The S3 key (path) also replaces '.encrypted' with '.decrypted'
s3.upload_file(decrypted_file, BUCKET_NAME, file_key.replace('.encrypted',
'.decrypted'))

print(f"File {file_key} decrypted successfully")

def compare_files(original_key, decrypted_key):
"""

Compare the contents of the original file and its decrypted version in S3.

Args:
    original_key (str): The key (path) of the original file in S3
    decrypted_key (str): The key (path) of the decrypted file in S3
"""

# Create unique local filenames for the original and decrypted files
original_file = 'original_' + os.path.basename(original_key)
decrypted_file = 'decrypted_' + os.path.basename(decrypted_key)

# Download both the original and decrypted files from S3
s3.download_file(BUCKET_NAME, original_key, original_file)
s3.download_file(BUCKET_NAME, decrypted_key, decrypted_file)

# Open and read both files, then compare their contents
with open(original_file, 'rb') as f1, open(decrypted_file, 'rb') as f2:
    if f1.read() == f2.read():
        # If the contents match, the decryption was successful
        print(f"File {original_key} was successfully decrypted and matches the
original!")
    else:

```

```

# If the contents don't match, there was an error in the
# encryption/decryption process
print(f"Error: {original_key} and its decrypted version do not match.")

```

Modify the `main()` function to perform the decryption process:

```

def main():
    """
    Main function to decrypt all encrypted files in the specified S3 bucket
    and compare them with their original versions.
    """

    global key_id
    key_id = get_kms_key_id()

    response = s3.list_objects_v2(Bucket=BUCKET_NAME)

    for obj in response.get('Contents', []):
        file_key = obj['Key']

        # Check if the file is an encrypted one (ends with '.encrypted')
        if file_key.endswith('.encrypted'):
            # Decrypt the file
            decrypt_file(file_key)

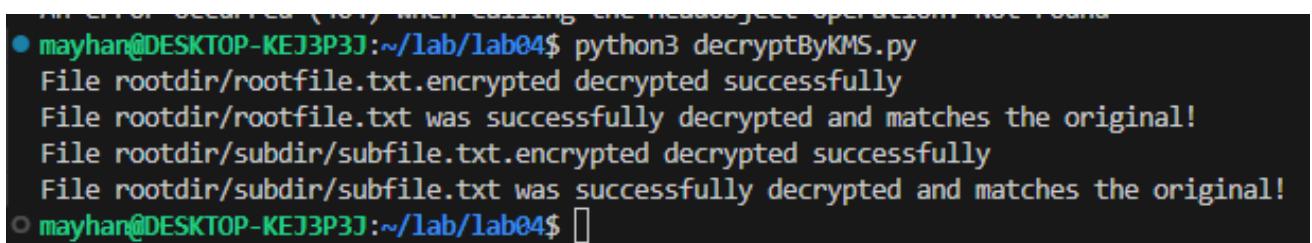
            # Construct the key for the original (unencrypted) file
            # by removing the '.encrypted' suffix
            original_key = file_key.replace('.encrypted', '')

            # Construct the key for the decrypted file
            # by replacing '.encrypted' with '.decrypted'
            decrypted_key = file_key.replace('.encrypted', '.decrypted')

            # Compare the original file with its decrypted version
            compare_files(original_key, decrypted_key)

```

Save the modified script as "decryptByKMS.py" and run the code:



```

● mayhan@DESKTOP-KEJ3P3J:~/lab/lab04$ python3 decryptByKMS.py
File rootdir/rootfile.txt.encrypted decrypted successfully
File rootdir/rootfile.txt was successfully decrypted and matches the original!
File rootdir/subdir/subfile.txt.encrypted decrypted successfully
File rootdir/subdir/subfile.txt was successfully decrypted and matches the original!
● mayhan@DESKTOP-KEJ3P3J:~/lab/lab04$ 

```

The result shows that **the decrypted file matches the original file**.

Opening the **AWS S3 Console** reveals the new decrypted files:

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](#)< [1](#) > [⚙️](#)

<input type="checkbox"/>	Name	Type	Last modified	Size
<input type="checkbox"/>	rootfile.txt	txt	September 13, 2024, 19:42:50 (UTC+08:00)	15.0 E
<input type="checkbox"/>	rootfile.txt.decrypted	decrypted	September 13, 2024, 21:56:59 (UTC+08:00)	15.0 E
<input type="checkbox"/>	rootfile.txt.encrypted	encrypted	September 13, 2024, 21:42:38 (UTC+08:00)	167.0 E
<input type="checkbox"/>	subdir/	Folder	-	

subdir/

[Copy S3 URI](#)[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](#)

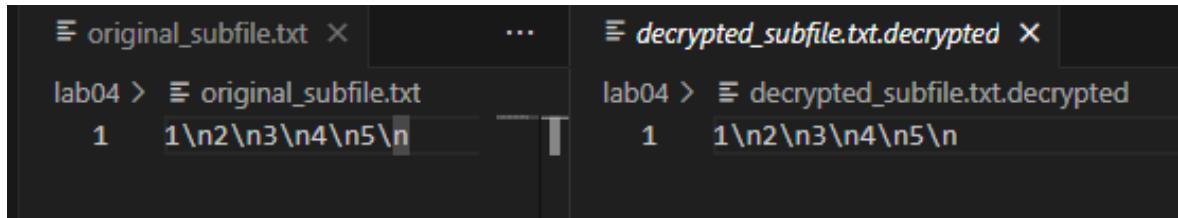
 [Find objects by prefix](#)< [1](#) > [⚙️](#)

<input type="checkbox"/>	Name	Type	Last modified	Size
<input type="checkbox"/>	subfile.txt	txt	September 13, 2024, 19:42:50 (UTC+08:00)	15.0 E
<input type="checkbox"/>	subfile.txt.decrypted	decrypted	September 13, 2024, 21:56:59 (UTC+08:00)	15.0 E
<input type="checkbox"/>	subfile.txt.encrypted	encrypted	September 13, 2024, 21:42:39 (UTC+08:00)	167.0 E

Comparing the downloaded decrypted file with the original, matching the original file :

```
original_rootfile.txt  ...
lab04 > original_rootfile.txt
1 1\n2\n3\n4\n5\n

decrypted_rootfile.txt.decrypted  ...
lab04 > decrypted_rootfile.txt.decrypted
1 1\n2\n3\n4\n5\n
```



```
original_subfile.txt  ...
lab04 > original_subfile.txt
1 1\n2\n3\n4\n5\n

decrypted_subfile.txt.decrypted  ...
lab04 > decrypted_subfile.txt.decrypted
1 1\n2\n3\n4\n5\n
```

[5] Apply `pycryptodome` for encryption/decryption

Write another Python script using the `pycryptodome` library to encrypt and decrypt each file in the S3 bucket. The general process is similar to the previous step using KMS key:

```
import os, random, struct
from Crypto.Cipher import AES
from Crypto import Random
import boto3
import base64
import hashlib

s3 = boto3.client('s3', region_name='ap-southeast-1')
BUCKET_NAME = '23905652-cloudstorage'

# Constants for encryption
BLOCK_SIZE = 16 # AES block size
CHUNK_SIZE = 64 * 1024 # Size of chunks to read/write at a time (64KB)

def encrypt_file(password, in_filename, out_filename):
    # Generate a 256-bit key from the password using SHA-256
    key = hashlib.sha256(password.encode("utf-8")).digest()

    # Generate a random 16-byte initialization vector
    iv = Random.new().read(AES.block_size)

    # Create an AES cipher object in CBC mode
    encryptor = AES.new(key, AES.MODE_CBC, iv)

    # Get the size of the input file
    filesize = os.path.getsize(in_filename)

    with open(in_filename, 'rb') as infile:
        with open(out_filename, 'wb') as outfile:
            # Write the original filesize and IV to the output file
            outfile.write(struct.pack('<Q', filesize))
            outfile.write(iv)

            while True:
                # Read the file in chunks
                chunk = infile.read(CHUNK_SIZE)
                if len(chunk) == 0:
                    break
                elif len(chunk) % 16 != 0:
                    # Pad the chunk if it's not a multiple of 16 bytes
                    chunk += b'\0' * (16 - len(chunk) % 16)

                encryptor.update(chunk)
                outfile.write(encryptor.encrypt(chunk))
```

```

        chunk += ' '.encode("utf-8") * (16 - len(chunk) % 16)

    # Encrypt and write the chunk
    outfile.write(encryptor.encrypt(chunk))

def decrypt_file(password, in_filename, out_filename):
    # Generate the same 256-bit key from the password
    key = hashlib.sha256(password.encode("utf-8")).digest()

    with open(in_filename, 'rb') as infile:
        # Read the original filesize and IV
        origsize = struct.unpack('<Q', infile.read(struct.calcsize('Q')))[0]
        iv = infile.read(16)

        # Create an AES cipher object for decryption
        decryptor = AES.new(key, AES.MODE_CBC, iv)

        with open(out_filename, 'wb') as outfile:
            while True:
                # Read and decrypt chunks
                chunk = infile.read(CHUNK_SIZE)
                if len(chunk) == 0:
                    break
                outfile.write(decryptor.decrypt(chunk))

            # Truncate the output file to the original size (removes padding)
            outfile.truncate(origsize)

def process_s3_files(password):
    response = s3.list_objects_v2(Bucket=BUCKET_NAME)

    for obj in response.get('Contents', []):
        file_key = obj['Key']
        if not file_key.endswith('.enc', '.dec'):
            # Download the file from S3
            local_file = os.path.basename(file_key)
            s3.download_file(BUCKET_NAME, file_key, local_file)

            # Encrypt the file
            encrypted_file = local_file + ".enc"
            encrypt_file(password, local_file, encrypted_file)
            # Upload the encrypted file back to S3
            s3.upload_file(encrypted_file, BUCKET_NAME, f"{file_key}.enc")
            print(f"File {file_key} encrypted successfully")

            # Decrypt the file
            decrypted_file = local_file + ".dec"
            decrypt_file(password, encrypted_file, decrypted_file)
            # Upload the decrypted file back to S3
            s3.upload_file(decrypted_file, BUCKET_NAME, f"{file_key}.dec")
            print(f"File {file_key} decrypted successfully")

```

```

# Compare original and decrypted files to verify integrity
with open(local_file, 'rb') as f1, open(decrypted_file, 'rb') as f2:
    if f1.read() == f2.read():
        print(f"File {file_key} was successfully encrypted and decrypted!")
    else:
        print(f"Error: {file_key} and its decrypted version do not match.")

if __name__ == "__main__":
    password = 'kitty and the kat' # Password for encryption/decryption
    process_s3_files(password)

```

Save the script as "pycryptodome.py" and run the code:

```

Installing collected packages: pycryptodome
Successfully installed pycryptodome-3.20.0
mayhan@DESKTOP-KEJ3P3J:~/lab/lab04$ python3 pycryptodome.py
File rootdir/rootfile.txt encrypted successfully
File rootdir/rootfile.txt decrypted successfully
File rootdir/rootfile.txt was successfully encrypted and decrypted!
File rootdir/rootfile.txt.decrypted encrypted successfully
File rootdir/rootfile.txt.decrypted decrypted successfully
File rootdir/rootfile.txt.decrypted was successfully encrypted and decrypted!
File rootdir/rootfile.txt.encrypted encrypted successfully
File rootdir/rootfile.txt.encrypted decrypted successfully
File rootdir/rootfile.txt.encrypted was successfully encrypted and decrypted!
File rootdir/subdir/subfile.txt encrypted successfully
File rootdir/subdir/subfile.txt decrypted successfully
File rootdir/subdir/subfile.txt was successfully encrypted and decrypted!
File rootdir/subdir/subfile.txt.decrypted encrypted successfully
File rootdir/subdir/subfile.txt.decrypted decrypted successfully
File rootdir/subdir/subfile.txt.decrypted was successfully encrypted and decrypted!
File rootdir/subdir/subfile.txt.encrypted encrypted successfully
File rootdir/subdir/subfile.txt.encrypted decrypted successfully
File rootdir/subdir/subfile.txt.encrypted was successfully encrypted and decrypted!

```

After running the code, we can find that the program performs encryption and decryption operations on the '.encrypted' and '.decrypted' files from the previous step as well. For effective checking, **we focus only on the original files.**

In the **AWS S3 console**, we can see the newly added encrypted and decrypted files:

[Objects](#) [Properties](#)**Objects (10) [Info](#)** [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 [1](#)

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	 rootfile.txt	txt	September 13, 2024, 19:42:50 (UTC+08:00)	15.0 B	Standard
<input type="checkbox"/>	 rootfile.txt.dec	dec	September 13, 2024, 22:16:06 (UTC+08:00)	15.0 B	Standard
<input type="checkbox"/>	 rootfile.txt.decrypted	decrypted	September 13, 2024, 22:00:36 (UTC+08:00)	15.0 B	Standard
<input type="checkbox"/>	 rootfile.txt.decrypted.dec	dec	September 13, 2024, 22:16:07 (UTC+08:00)	15.0 B	Standard
<input type="checkbox"/>	 rootfile.txt.decrypted.enc	enc	September 13, 2024, 22:16:07 (UTC+08:00)	40.0 B	Standard
<input type="checkbox"/>	 rootfile.txt.enc	enc	September 13, 2024, 22:16:06 (UTC+08:00)	40.0 B	Standard
<input type="checkbox"/>	 rootfile.txt.encrypted	encrypted	September 13, 2024, 21:42:38 (UTC+08:00)	167.0 B	Standard
<input type="checkbox"/>	 rootfile.txt.encrypted.dec	dec	September 13, 2024, 22:16:07 (UTC+08:00)	167.0 B	Standard
<input type="checkbox"/>	 rootfile.txt.encrypted.enc	enc	September 13, 2024, 22:16:07 (UTC+08:00)	200.0 B	Standard
<input type="checkbox"/>	 subdir/	Folder	-	-	-

Amazon S3 > Buckets > 23905652-cloudstorage > rootdir/ > subdir/

subdir/

[Copy S3 URI](#)

[Objects](#) [Properties](#)

Objects (9) [Info](#)

[C](#) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#) [Upload](#)

Objects are 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](#)

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	subfile.txt	txt	September 13, 2024, 19:42:50 (UTC+08:00)	15.0 B	Standard
<input type="checkbox"/>	subfile.txt.dec	dec	September 13, 2024, 22:16:08 (UTC+08:00)	15.0 B	Standard
<input type="checkbox"/>	subfile.txt.decrypted	decrypted	September 13, 2024, 22:00:37 (UTC+08:00)	15.0 B	Standard
<input type="checkbox"/>	subfile.txt.decrypted.dec	dec	September 13, 2024, 22:16:08 (UTC+08:00)	15.0 B	Standard
<input type="checkbox"/>	subfile.txt.decrypted.enc	enc	September 13, 2024, 22:16:08 (UTC+08:00)	40.0 B	Standard
<input type="checkbox"/>	subfile.txt.enc	enc	September 13, 2024, 22:16:08 (UTC+08:00)	40.0 B	Standard
<input type="checkbox"/>	subfile.txt.encrypted	encrypted	September 13, 2024, 21:42:39 (UTC+08:00)	167.0 B	Standard
<input type="checkbox"/>	subfile.txt.encrypted.dec	dec	September 13, 2024, 22:16:09 (UTC+08:00)	167.0 B	Standard
<input type="checkbox"/>	subfile.txt.encrypted.enc	enc	September 13, 2024, 22:16:08 (UTC+08:00)	200.0 B	Standard

Download and compare the original file/encrypted file/decrypted file. The decrypted file matches the original one, and the encrypted file shows the ciphertext:

Answer the following question

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

- Scalability:
 - KMS: Capable of handling a large number of concurrent requests, suitable for high-throughput scenarios.

- Custom Solution: Limited by local resources, may not perform as well when handling many concurrent requests.
- Network Latency:
 - KMS: Each encryption or decryption operation requires communication with AWS servers, which may introduce network latency.
 - Custom Solution: Executed locally, no additional network latency.
- Large File Handling:
 - KMS: Has a 4KB data size limit, requiring chunking for large files, which may impact performance.
 - Custom Solution: Can handle large files more flexibly, typically more efficient for large file processing.
- Key Management:
 - KMS: Provides automatic key rotation and versioning, but may add some overhead.
 - Custom Solution: Key management is your responsibility, potentially faster but less secure.
- Cost:
 - KMS: Charged based on usage, high volume may result in significant costs.
 - Custom Solution: Main cost is computational resources, potentially more economical for high-volume operations.

Lab 5 Networking

Application Load Balancer

[1] Create 2 EC2 instances

Similar to the script created for EC2 in Lab 2, we will write a Python Boto3 script to create a security group to allow inbound HTTP and SSH traffic, and create 2 EC2 instances in two different availability zones, named '-vm1' and '-vm2'. The repeated part of the code from before has not been commented.

```
import boto3
import os

student_number = "23905652"
region = "ap-southeast-1"

ec2 = boto3.client('ec2', region_name=region)

def create_security_group():
    # Create a new security group with a name based on the student number
    response = ec2.create_security_group(
       GroupName=f"{student_number}-sg",
        Description="Security group for ALB lab"
    )
    security_group_id = response['GroupId']

    # Add inbound rules to the security group
    # Allow incoming traffic on port 80 (HTTP) and port 22 (SSH) from any IP address
    ec2.authorize_security_group_ingress(
        GroupId=security_group_id,
        IpPermissions=[
            {
                'IpProtocol': 'tcp',
                'FromPort': 80,
                'ToPort': 80,
                'IpRanges': [{"CidrIp": '0.0.0.0/0'}]
            },
            {
                'IpProtocol': 'tcp',
                'FromPort': 22,
                'ToPort': 22,
                'IpRanges': [{"CidrIp": '0.0.0.0/0'}]
            }
        ]
    )
    print(f"Created Security Group: {security_group_id}")
    return security_group_id

def create_key_pair():
```

```

key_name = f"{student_number}-key"
response = ec2.create_key_pair(KeyName=key_name)
private_key = response[ 'KeyMaterial' ]

private_key_file = f"{key_name}.pem"

with open(private_key_file, 'w') as key_file:
    key_file.write(private_key)

os.chmod(private_key_file, 0o400)

print(f"Private key saved to {private_key_file}")
return key_name

def get_availability_zones():
    # Retrieves the first two availability zones in the specified region

    # Make an API call to describe availability zones with a filter for "ap-southeast-1"
    response = ec2.describe_availability_zones(
        Filters=[
            {
                'Name': 'region-name', # Filter by region name
                'Values': [region]      # Use the global 'region' variable defined earlier
            },
        ]
    )
    # Return the 'ZoneName' of the first two zones available zones
    return [az['ZoneName'] for az in response['AvailabilityZones'][:2]]

def create_ec2_instance(security_group_id, key_name, availability_zone, instance_number):
    # Launch a new EC2 instance with specified parameters
    response = ec2.run_instances(
        ImageId="ami-0497a974f8d5dcef8",
        InstanceType="t2.micro",
        KeyName=key_name,
        MaxCount=1,
        MinCount=1,
        SecurityGroupIds=[security_group_id],
        Placement={
            'AvailabilityZone': availability_zone
        },
        TagSpecifications=[
            {
                'ResourceType': 'instance',
                'Tags': [
                    {'Key': 'Name', 'Value': f'{student_number}-vm{instance_number}'}
                ]
            }
        ]
    )
    instance_id = response['Instances'][0]['InstanceId']
    print(f"Created EC2 instance: {instance_id} in {availability_zone}")

```

```

    return instance_id

def main():
    # Create security group
    security_group_id = create_security_group()

    # Create key pair
    key_name = create_key_pair()

    # Get availability zones
    availability_zones = get_availability_zones()

    # Create EC2 instances
    instance_ids = []
    for i, az in enumerate(availability_zones, start=1):
        # Create an EC2 instance in each availability zone
        instance_id = create_ec2_instance(security_group_id, key_name, az, i)
        # Add the new instance ID to our list
        instance_ids.append(instance_id)

    # Wait for instances to be running
    print("Waiting for instances to enter 'running' state...")
    waiter = ec2.get_waiter('instance_running')
    waiter.wait(InstanceIds=instance_ids)
    print("Instances are now running.")

    # Describe instances to get public IP addresses
    for i, instance_id in enumerate(instance_ids, start=1):
        # Retrieve detailed information about each instance
        response = ec2.describe_instances(InstanceIds=[instance_id])
        # Extract the instance details from the response
        instance = response['Reservations'][0]['Instances'][0]
        # Get the public IP address, or 'N/A' if it doesn't exist
        public_ip = instance.get('PublicIpAddress', 'N/A')
        # Get the availability zone where the instance is running
        az = instance['Placement']['AvailabilityZone']
        # Print out the details of each instance
        print(f"Instance {i} (ID: {instance_id}) created in {az} with Public IP:
{public_ip}")

if __name__ == "__main__":
    main()

```

Save the script as "create2EC2.py" and run the code:

```

● mayhan@DESKTOP-KEJ3P3J:~/lab/lab05$ python3 create2EC2.py
Created Security Group: sg-02e38b48c9c59b8f4
Private key saved to 23905652-key.pem
Created EC2 instance: i-0f97f02b65d2532eb in ap-southeast-1a
Created EC2 instance: i-08f04a80fbe65da9e in ap-southeast-1b
Waiting for instances to enter 'running' state...
Instances are now running.
Instance 1 (ID: i-0f97f02b65d2532eb) created in ap-southeast-1a with Public IP: 54.255.225.54
Instance 2 (ID: i-08f04a80fbe65da9e) created in ap-southeast-1b with Public IP: 13.250.104.237
● mayhan@DESKTOP-KEJ3P3J:~/lab/lab05$ 

```

We can check the EC2 instances in the **AWS Console**, named them "23905652-vm1" and "23905652-vm2":

Instances (3) Info									
Find Instance by attribute or tag (case-sensitive) Last updated 1 minute ago									
All states ▾ Connect Instance state ▾ Actions ▾ Launch instances ▾									
	Name ▾	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
<input type="checkbox"/>	23909219-vm	i-011a8ff6b56a83a66	Stopped ⓘ ⓘ	t2.micro	-	User: arm:aw ⓘ	ap-southeast-1c	-	-
<input type="checkbox"/>	23905652-vm1	i-0f97f02b65d2532eb	Running ⓘ ⓘ	t2.micro	Initializing ⓘ	User: arm:aw ⓘ	ap-southeast-1a	ec2-54-255-225-54.ap... 54.255.225.54	-
<input type="checkbox"/>	23905652-vm2	i-08f04a80fbe65da9e	Running ⓘ ⓘ	t2.micro	Initializing ⓘ	User: arm:aw ⓘ	ap-southeast-1b	ec2-13-250-104-237.ap... 13.250.104.237	-

It shows that **the Instance ID matches the output in Terminal**.

i-0f97f02b65d2532eb (23905652-vm1)

[Details](#) | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Instance summary [Info](#)

Instance ID	i-0f97f02b65d2532eb (23905652-vm1)	Public IPv4 address	54.255.225.54 open address
IPv6 address	-	Instance state	Running
Hostname type	IP name: ip-172-31-33-111.ap-southeast-1.compute.internal	Private IP DNS name (IPv4 only)	ip-172-31-33-111.ap-southeast-1.compute.internal
Answer private resource DNS name	-	Instance type	t2.micro
Auto-assigned IP address	54.255.225.54 [Public IP]	VPC ID	vpc-0ad7c05df6174aa82
IAM Role	-	Subnet ID	subnet-0859ff38bfce967b8 (23925353)
IMDSv2	Optional	Instance ARN	arn:aws:ec2:ap-southeast-1:489389878001:instance/i-0f97f02b65d2532eb

i-08f04a80fbe65da9e (23905652-vm2)

[Details](#) | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Instance summary [Info](#)

Instance ID	i-08f04a80fbe65da9e (23905652-vm2)	Public IPv4 address	13.250.104.237 open address
IPv6 address	-	Instance state	Running
Hostname type	IP name: ip-172-31-24-68.ap-southeast-1.compute.internal	Private IP DNS name (IPv4 only)	ip-172-31-24-68.ap-southeast-1.compute.internal
Answer private resource DNS name	-	Instance type	t2.micro
Auto-assigned IP address	13.250.104.237 [Public IP]	VPC ID	vpc-0ad7c05df6174aa82
IAM Role	-	Subnet ID	subnet-056a5c6c3bd465883 (23925353)
IMDSv2	Optional	Instance ARN	arn:aws:ec2:ap-southeast-1:489389878001:instance/i-08f04a80fbe65da9e

Click to view the details of the two instances, and we can find that **the public IP address also matches the output**.

[2] Create an Application Load Balancer

Update the above script to create an application load balancer and load balance HTTP requests to the 2 instances created, using the v2 of the ELB interface. The updates to the script include:

- Create a load balancer, specifying the two subnets and security groups created in the previous step.
- Create a target group using the same VPC used for creating instances.
- Register targets in the target group.
- Create a listener with the default rule protocol: HTTP and port 80 forwarding to the target group.

1. To achieve this, we added four functions to the previous script, `create_load_balancer()`, `create_target_group()`, `register_targets()`, `create_listener()`:

```
import boto3
import botocore
import os

student_number = "23905652"
region = "ap-southeast-1"

# Initialize boto3 clients for EC2 and Elastic Load Balancing v2 services
ec2 = boto3.client('ec2', region_name=region)
elbv2 = boto3.client('elbv2', region_name=region)

def create_load_balancer(security_group_id, subnet_ids):
    # Create an Application Load Balancer (ALB)
    response = elbv2.create_load_balancer(
        Name=f'{student_number}-alb', # Name the ALB using the student number
        Subnets=subnet_ids, # Specify subnets where the ALB will be created
        SecurityGroups=[security_group_id], # Attach a security group to the ALB
        Scheme='internet-facing', # Make the ALB accessible from the internet
        Type='application', # Specify that this is an Application Load Balancer
        Tags=[
            {
                'Key': 'Name',
                'Value': f'{student_number}-alb'
            },
        ] # Tag the ALB for easier identification
    )
    # Return the Amazon Resource Name (ARN) of the created load balancer
    return response['LoadBalancers'][0]['LoadBalancerArn']

def create_target_group(vpc_id):
    # Create a target group for the ALB
    response = elbv2.create_target_group(
        Name=f'{student_number}-tg', # Name the target group using the student number
        Protocol='HTTP', # Use HTTP protocol for the target group
        Port=80, # Set the port to 80 for HTTP traffic
        VpcId=vpc_id, # Specify the VPC where the target group will be created
        HealthCheckProtocol='HTTP', # Use HTTP for health checks
    )
```

```

        HealthCheckPath='/', # Set the health check path to the root
        TargetType='instance' # Specify that the targets will be EC2 instances
    )
# Return the ARN of the created target group
return response['TargetGroups'][0]['TargetGroupArn']

def register_targets(target_group_arn, instance_ids):
    # Register EC2 instances as targets in the target group
    targets = [{ 'Id': instance_id} for instance_id in instance_ids] # Create a list of
target dictionaries

    elbv2.register_targets(
        TargetGroupArn=target_group_arn, # Specify the target group
        Targets=targets # Register the EC2 instances as targets
    )

def create_listener(load_balancer_arn, target_group_arn):
    # Create a listener for the ALB
    response = elbv2.create_listener(
        LoadBalancerArn=load_balancer_arn, # Specify the ALB
        Protocol='HTTP', # Use HTTP protocol for the listener
        Port=80, # Set the port to 80 for HTTP traffic
        DefaultActions=[
            {
                'Type': 'forward', # Set the default action to forward traffic
                'TargetGroupArn': target_group_arn # Specify the target group to forward
traffic to
            }
        ]
    )
    # Return the ARN of the created listener
    return response['Listeners'][0]['ListenerArn']

def main():
    # Main function to process the creation of ALB and related resources

    # Hardcoded resource IDs
    vpc_id = 'vpc-0ad7c05df6174aa82'
    subnet_ids = [ 'subnet-0859ff38bfce967b8', 'subnet-056a5c6c3bd465883' ]
    security_group_id = 'sg-02e38b48c9c59b8f4'
    instance_ids = [ 'i-0f97f02b65d2532eb', 'i-08f04a80fbe65da9e' ]

    print("Creating Application Load Balancer...")
    load_balancer_arn = create_load_balancer(security_group_id, subnet_ids)
    print(f"Load Balancer created: {load_balancer_arn}")

    print("Creating Target Group...")
    target_group_arn = create_target_group(vpc_id)
    print(f"Target Group created: {target_group_arn}")

    print("Creating listener...")
    listener_arn = create_listener(load_balancer_arn, target_group_arn)

```

```

print(f"Listener created: {listener_arn}")

print("Registering targets...")
register_targets(target_group_arn, instance_ids)
print("Targets registered")

print("Setup complete!")

if __name__ == "__main__":
    main()

```

2. In the `main()` function, the `security_group_id` and `instance_ids` are saved from the previous output, while the `vpc_id` and `subnet_ids` are found in the details of each instance in the AWS Console and then hardcoded into the program to operate on specific objects:

i-0f97f02b65d2532eb (23905652-vm1)

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Instance summary [Info](#)

Instance ID	Public IPv4 address
i-0f97f02b65d2532eb (23905652-vm1)	54.255.225.54 open address
IPv6 address	Instance state
-	Running
Hostname type	Private IP DNS name (IPv4 only)
IP name: ip-172-31-33-111.ap-southeast-1.compute.internal	ip-172-31-33-111.ap-southeast-1.compute.internal
Answer private resource DNS name	Instance type
-	t2.micro
Auto-assigned IP address	VPC ID
54.255.225.54 [Public IP]	vpc-0ad7c05df6174aa82
IAM Role	Subnet ID
-	subnet-0859ff38bfce967b8 (23925353)

i-08f04a80fbe65da9e (23905652-vm2)

Details Status and alarms Monitoring Security Networking Storage Tags

Instance summary [Info](#)

Instance ID

[i-08f04a80fbe65da9e \(23905652-vm2\)](#)

Public IPv4 address

[13.250.104.237 | open address](#)

IPv6 address

Instance state

[Running](#)

Hostname type

IP name: ip-172-31-24-68.ap-southeast-1.compute.internal

Private IP DNS name (IPv4 only)

[ip-172-31-24-68.ap-southeast-1.compute.internal](#)

Answer private resource DNS name

Instance type

t2.micro

Auto-assigned IP address

[13.250.104.237 \[Public IP\]](#)

VPC ID

[vpc-0ad7c05df6174aa82](#)

IAM Role

Subnet ID

[subnet-056a5c6c3bd465883 \(23925353\)](#)

3. Save the script as "createALB.py" and run the code:

```
● mayhan@DESKTOP-KEJ3P3J:~/lab/lab05$ python3 createALB.py
Creating Application Load Balancer...
Load Balancer created: arn:aws:elasticloadbalancing:ap-southeast-1:489389878001:loadbalancer/app/23905652-alb/ed59254d0b721306
Creating Target Group...
Target Group created: arn:aws:elasticloadbalancing:ap-southeast-1:489389878001:targetgroup/23905652-tg/a34d3644c87dce17
Creating listener...
Listener created: arn:aws:elasticloadbalancing:ap-southeast-1:489389878001:listener/app/23905652-alb/ed59254d0b721306/cdb20af71e4a5170
Registering targets...
Targets registered
Setup complete!
```

Based on the output, the setup can be considered successful.

4. In the AWS Console, we can see the created Load Balancer and Target Group information:

- **Load balancer:**

Load balancers (1/1)						
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.						
<input type="button" value="C"/> Actions ▾ <input type="button" value="Create load balancer"/> <input type="button" value="▼"/>						
<input type="text" value="Filter load balancers"/>						
Name	DNS name	State	VPC ID	Availability Zones	Type	Date created
23905652-alb	23905652-alb-121094141...	Provisioning...	vpc-0ad7c05df6174aa82	2 Availability Zones	application	September 14, 2024, 16:38 (UTC+0...)

Load balancer: 23905652-alb

Details	Listeners and rules	Network mapping	Resource map - new	Security	Monitoring	Integrations	Attributes	Tags
Details								
Load balancer type Application	Status Provisioning	VPC vpc-0ad7c05df6174aa82	Load balancer IP address type IPv4					
Scheme Internet-facing	Hosted zone Z1LMS91P8CMLES	Availability Zones subnet-0859ff38bfce967b8 ap-southeast-1a (apse1-az2) subnet-056a5c6c3bd465883 ap-southeast-1b (apse1-az1)	Date created September 14, 2024, 16:38 (UTC+0:00)					
Load balancer ARN arn:aws:elasticloadbalancing:ap-southeast-1:489389878001:loadbalancer/app/23905652-alb/ed59254d0b721306	DNS name Info 23905652-alb-1210941414.ap-southeast-1.elb.amazonaws.com (A Record)							

Load balancer: 23905652-alb

Details	Listeners and rules	Network mapping	Resource map - new	Security	Monitoring	Integrations	Attributes	Tags
Listeners and rules (1) Info								
A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.								
<input type="text" value="Filter listeners"/>	<input type="button" value="C"/> Manage rules ▾ <input type="button" value="Manage listener"/> <input type="button" value="Add listener"/>							
<input type="checkbox"/> Protocol:Port <input type="checkbox"/> HTTP:80	<input type="checkbox"/> Default action <input type="checkbox"/> Forward to target group <ul style="list-style-type: none"> 23905652-tg: 1 (100%) Target group stickiness: Off 	<input type="checkbox"/> Rules 1 rule	<input type="checkbox"/> ARN ARN	<input type="checkbox"/> Security policy Not applicable	<input type="checkbox"/> Default SSL/TLS certificate Not applicable	<input type="checkbox"/> mTLS Not applicable		

Load balancer: 23905652-alb

Details	Listeners and rules	Network mapping	Resource map - new	Security	Monitoring	Integrations	Attributes	Tags						
Network mapping Info														
Targets in the listed zones and subnets are available for traffic from the load balancer using the IP addresses shown.														
<table border="1"> <tr> <td>VPC vpc-0ad7c05df6174aa82</td><td>Load balancer IP address type IPv4</td></tr> <tr> <td>IPv4 VPC CIDR: 172.31.0.0/16</td><td></td></tr> <tr> <td>IPv6 : -</td><td></td></tr> </table>									VPC vpc-0ad7c05df6174aa82	Load balancer IP address type IPv4	IPv4 VPC CIDR: 172.31.0.0/16		IPv6 : -	
VPC vpc-0ad7c05df6174aa82	Load balancer IP address type IPv4													
IPv4 VPC CIDR: 172.31.0.0/16														
IPv6 : -														
Mappings														
Including two or more Availability Zones, and corresponding subnets, increases the fault tolerance of your applications.														
Zone	Subnet	IPv4 address	Private IPv4 address	IPv6 address										
ap-southeast-1a (apse1-az2)	subnet-0859ff38bfce967b8	Assigned by AWS	Assigned from CIDR 172.31.32.0/20	Not applicable										
ap-southeast-1b (apse1-az1)	subnet-056a5c6c3bd465883	Assigned by AWS	Assigned from CIDR 172.31.16.0/20	Not applicable										

- Target group:

Target group: 23905652-tg

Details Targets Monitoring Health checks Attributes Tags

Registered targets (2) [Info](#)

Anomaly mitigation: Not applicable [Edit](#) [Deregister](#) [Register targets](#)

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Filter targets [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) [12](#) [13](#) [14](#) [15](#) [16](#) [17](#) [18](#) [19](#) [20](#) [21](#) [22](#) [23](#) [24](#) [25](#) [26](#) [27](#) [28](#) [29](#) [30](#) [31](#) [32](#) [33](#) [34](#) [35](#) [36](#) [37](#) [38](#) [39](#) [40](#) [41](#) [42](#) [43](#) [44](#) [45](#) [46](#) [47](#) [48](#) [49](#) [50](#) [51](#) [52](#) [53](#) [54](#) [55](#) [56](#) [57](#) [58](#) [59](#) [60](#) [61](#) [62](#) [63](#) [64](#) [65](#) [66](#) [67](#) [68](#) [69](#) [70](#) [71](#) [72](#) [73](#) [74](#) [75](#) [76](#) [77](#) [78](#) [79](#) [80](#) [81](#) [82](#) [83](#) [84](#) [85](#) [86](#) [87](#) [88](#) [89](#) [90](#) [91](#) [92](#) [93](#) [94](#) [95](#) [96](#) [97](#) [98](#) [99](#) [100](#) [101](#) [102](#) [103](#) [104](#) [105](#) [106](#) [107](#) [108](#) [109](#) [110](#) [111](#) [112](#) [113](#) [114](#) [115](#) [116](#) [117](#) [118](#) [119](#) [120](#) [121](#) [122](#) [123](#) [124](#) [125](#) [126](#) [127](#) [128](#) [129](#) [130](#) [131](#) [132](#) [133](#) [134](#) [135](#) [136](#) [137](#) [138](#) [139](#) [140](#) [141](#) [142](#) [143](#) [144](#) [145](#) [146](#) [147](#) [148](#) [149](#) [150](#) [151](#) [152](#) [153](#) [154](#) [155](#) [156](#) [157](#) [158](#) [159](#) [160](#) [161](#) [162](#) [163](#) [164](#) [165](#) [166](#) [167](#) [168](#) [169](#) [170](#) [171](#) [172](#) [173](#) [174](#) [175](#) [176](#) [177](#) [178](#) [179](#) [180](#) [181](#) [182](#) [183](#) [184](#) [185](#) [186](#) [187](#) [188](#) [189](#) [190](#) [191](#) [192](#) [193](#) [194](#) [195](#) [196](#) [197](#) [198](#) [199](#) [200](#) [201](#) [202](#) [203](#) [204](#) [205](#) [206](#) [207](#) [208](#) [209](#) [210](#) [211](#) [212](#) [213](#) [214](#) [215](#) [216](#) [217](#) [218](#) [219](#) [220](#) [221](#) [222](#) [223](#) [224](#) [225](#) [226](#) [227](#) [228](#) [229](#) [230](#) [231](#) [232](#) [233](#) [234](#) [235](#) [236](#) [237](#) [238](#) [239](#) [240](#) [241](#) [242](#) [243](#) [244](#) [245](#) [246](#) [247](#) [248](#) [249](#) [250](#) [251](#) [252](#) [253](#) [254](#) [255](#) [256](#) [257](#) [258](#) [259](#) [260](#) [261](#) [262](#) [263](#) [264](#) [265](#) [266](#) [267](#) [268](#) [269](#) [270](#) [271](#) [272](#) [273](#) [274](#) [275](#) [276](#) [277](#) [278](#) [279](#) [280](#) [281](#) [282](#) [283](#) [284](#) [285](#) [286](#) [287](#) [288](#) [289](#) [290](#) [291](#) [292](#) [293](#) [294](#) [295](#) [296](#) [297](#) [298](#) [299](#) [300](#) [301](#) [302](#) [303](#) [304](#) [305](#) [306](#) [307](#) [308](#) [309](#) [310](#) [311](#) [312](#) [313](#) [314](#) [315](#) [316](#) [317](#) [318](#) [319](#) [320](#) [321](#) [322](#) [323](#) [324](#) [325](#) [326](#) [327](#) [328](#) [329](#) [330](#) [331](#) [332](#) [333](#) [334](#) [335](#) [336](#) [337](#) [338](#) [339](#) [340](#) [341](#) [342](#) [343](#) [344](#) [345](#) [346](#) [347](#) [348](#) [349](#) [350](#) [351](#) [352](#) [353](#) [354](#) [355](#) [356](#) [357](#) [358](#) [359](#) [360](#) [361](#) [362](#) [363](#) [364](#) [365](#) [366](#) [367](#) [368](#) [369](#) [370](#) [371](#) [372](#) [373](#) [374](#) [375](#) [376](#) [377](#) [378](#) [379](#) [380](#) [381](#) [382](#) [383](#) [384](#) [385](#) [386](#) [387](#) [388](#) [389](#) [390](#) [391](#) [392](#) [393](#) [394](#) [395](#) [396](#) [397](#) [398](#) [399](#) [400](#) [401](#) [402](#) [403](#) [404](#) [405](#) [406](#) [407](#) [408](#) [409](#) [410](#) [411](#) [412](#) [413](#) [414](#) [415](#) [416](#) [417](#) [418](#) [419](#) [420](#) [421](#) [422](#) [423](#) [424](#) [425](#) [426](#) [427](#) [428](#) [429](#) [430](#) [431](#) [432](#) [433](#) [434](#) [435](#) [436](#) [437](#) [438](#) [439](#) [440](#) [441](#) [442](#) [443](#) [444](#) [445](#) [446](#) [447](#) [448](#) [449](#) [450](#) [451](#) [452](#) [453](#) [454](#) [455](#) [456](#) [457](#) [458](#) [459](#) [460](#) [461](#) [462](#) [463](#) [464](#) [465](#) [466](#) [467](#) [468](#) [469](#) [470](#) [471](#) [472](#) [473](#) [474](#) [475](#) [476](#) [477](#) [478](#) [479](#) [480](#) [481](#) [482](#) [483](#) [484](#) [485](#) [486](#) [487](#) [488](#) [489](#) [490](#) [491](#) [492](#) [493](#) [494](#) [495](#) [496](#) [497](#) [498](#) [499](#) [500](#) [501](#) [502](#) [503](#) [504](#) [505](#) [506](#) [507](#) [508](#) [509](#) [510](#) [511](#) [512](#) [513](#) [514](#) [515](#) [516](#) [517](#) [518](#) [519](#) [520](#) [521](#) [522](#) [523](#) [524](#) [525](#) [526](#) [527](#) [528](#) [529](#) [530](#) [531](#) [532](#) [533](#) [534](#) [535](#) [536](#) [537](#) [538](#) [539](#) [540](#) [541](#) [542](#) [543](#) [544](#) [545](#) [546](#) [547](#) [548](#) [549](#) [550](#) [551](#) [552](#) [553](#) [554](#) [555](#) [556](#) [557](#) [558](#) [559](#) [560](#) [561](#) [562](#) [563](#) [564](#) [565](#) [566](#) [567](#) [568](#) [569](#) [570](#) [571](#) [572](#) [573](#) [574](#) [575](#) [576](#) [577](#) [578](#) [579](#) [580](#) [581](#) [582](#) [583](#) [584](#) [585](#) [586](#) [587](#) [588](#) [589](#) [590](#) [591](#) [592](#) [593](#) [594](#) [595](#) [596](#) [597](#) [598](#) [599](#) [600](#) [601](#) [602](#) [603](#) [604](#) [605](#) [606](#) [607](#) [608](#) [609](#) [610](#) [611](#) [612](#) [613](#) [614](#) [615](#) [616](#) [617](#) [618](#) [619](#) [620](#) [621](#) [622](#) [623](#) [624](#) [625](#) [626](#) [627](#) [628](#) [629](#) [630](#) [631](#) [632](#) [633](#) [634](#) [635](#) [636](#) [637](#) [638](#) [639](#) [640](#) [641](#) [642](#) [643](#) [644](#) [645](#) [646](#) [647](#) [648](#) [649](#) [650](#) [651](#) [652](#) [653](#) [654](#) [655](#) [656](#) [657](#) [658](#) [659](#) [660](#) [661](#) [662](#) [663](#) [664](#) [665](#) [666](#) [667](#) [668](#) [669](#) [670](#) [671](#) [672](#) [673](#) [674](#) [675](#) [676](#) [677](#) [678](#) [679](#) [680](#) [681](#) [682](#) [683](#) [684](#) [685](#) [686](#) [687](#) [688](#) [689](#) [690](#) [691](#) [692](#) [693](#) [694](#) [695](#) [696](#) [697](#) [698](#) [699](#) [700](#) [701](#) [702](#) [703](#) [704](#) [705](#) [706](#) [707](#) [708](#) [709](#) [710](#) [711](#) [712](#) [713](#) [714](#) [715](#) [716](#) [717](#) [718](#) [719](#) [720](#) [721](#) [722](#) [723](#) [724](#) [725](#) [726](#) [727](#) [728](#) [729](#) [730](#) [731](#) [732](#) [733](#) [734](#) [735](#) [736](#) [737](#) [738](#) [739](#) [740](#) [741](#) [742](#) [743](#) [744](#) [745](#) [746](#) [747](#) [748](#) [749](#) [750](#) [751](#) [752](#) [753](#) [754](#) [755](#) [756](#) [757](#) [758](#) [759](#) [760](#) [761](#) [762](#) [763](#) [764](#) [765](#) [766](#) [767](#) [768](#) [769](#) [770](#) [771](#) [772](#) [773](#) [774](#) [775](#) [776](#) [777](#) [778](#) [779](#) [780](#) [781](#) [782](#) [783](#) [784](#) [785](#) [786](#) [787](#) [788](#) [789](#) [790](#) [791](#) [792](#) [793](#) [794](#) [795](#) [796](#) [797](#) [798](#) [799](#) [800](#) [801](#) [802](#) [803](#) [804](#) [805](#) [806](#) [807](#) [808](#) [809](#) [810](#) [811](#) [812](#) [813](#) [814](#) [815](#) [816](#) [817](#) [818](#) [819](#) [820](#) [821](#) [822](#) [823](#) [824](#) [825](#) [826](#) [827](#) [828](#) [829](#) [830](#) [831](#) [832](#) [833](#) [834](#) [835](#) [836](#) [837](#) [838](#) [839](#) [840](#) [841](#) [842](#) [843](#) [844](#) [845](#) [846](#) [847](#) [848](#) [849](#) [850](#) [851](#) [852](#) [853](#) [854](#) [855](#) [856](#) [857](#) [858](#) [859](#) [860](#) [861](#) [862](#) [863](#) [864](#) [865](#) [866](#) [867](#) [868](#) [869](#) [870](#) [871](#) [872](#) [873](#) [874](#) [875](#) [876](#) [877](#) [878](#) [879](#) [880](#) [881](#) [882](#) [883](#) [884](#) [885](#) [886](#) [887](#) [888](#) [889](#) [890](#) [891](#) [892](#) [893](#) [894](#) [895](#) [896](#) [897](#) [898](#) [899](#) [900](#) [901](#) [902](#) [903](#) [904](#) [905](#) [906](#) [907](#) [908](#) [909](#) [910](#) [911](#) [912](#) [913](#) [914](#) [915](#) [916](#) [917](#) [918](#) [919](#) [920](#) [921](#) [922](#) [923](#) [924](#) [925](#) [926](#) [927](#) [928](#) [929](#) [930](#) [931](#) [932](#) [933](#) [934](#) [935](#) [936](#) [937](#) [938](#) [939](#) [940](#) [941](#) [942](#) [943](#) [944](#) [945](#) [946](#) [947](#) [948](#) [949](#) [950](#) [951](#) [952](#) [953](#) [954](#) [955](#) [956](#) [957](#) [958](#) [959](#) [960](#) [961](#) [962](#) [963](#) [964](#) [965](#) [966](#) [967](#) [968](#) [969](#) [970](#) [971](#) [972](#) [973](#) [974](#) [975](#) [976](#) [977](#) [978](#) [979](#) [980](#) [981](#) [982](#) [983](#) [984](#) [985](#) [986](#) [987](#) [988](#) [989](#) [990](#) [991](#) [992](#) [993](#) [994](#) [995](#) [996](#) [997](#) [998](#) [999](#) [1000](#)

Target group: 23905652-tg

Details Targets Monitoring Health checks Attributes Tags

Details

arn:aws:elasticloadbalancing:ap-southeast-1:489389878001:targetgroup/23905652-tg/a34d3644c87dce17

Target type Instance	Protocol : Port HTTP: 80	Protocol version HTTP1	VPC vpc-0ad7c05df6174aa82
IP address type IPv4	Load balancer 23905652-alb		
2 Total targets	0 Healthy	2 Unhealthy	0 Unused
	0 Anomalous		0 Initial
			0 Draining

Distribution of targets by Availability Zone (AZ)
Select values in this table to see corresponding filters applied to the Registered targets table below.

[3] Test the Application Load Balancer

To test the Application Load Balancer, we will try to access each EC2 instance using its public IP address in a browser.

1. To use ALB, we need to install **Apache 2** on the instances, a popular open-source web server software widely used for hosting websites and web applications across various operating systems.

First, connect to the two instances via SSH (using one of the instances as an example, the other follows the same steps):

```

23905652-key.pem createec2.py createdeb.py
mayhan@DESKTOP-KEJ3P3J:~/lab/lab05$ ssh -i 23905652-key.pem ubuntu@13.250.104.237
The authenticity of host '13.250.104.237 (13.250.104.237)' can't be established.
ECDSA key fingerprint is SHA256:68/giouYCCrYrHaIs1nNrHwNfp+eih+K4uAVfqIKkBg.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.250.104.237' (ECDSA) 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 Sat Sep 14 08:54:39 UTC 2024

System load: 0.0          Processes: 97
Usage of /: 20.7% of 7.57GB Users logged in: 0
Memory usage: 21%         IPv4 address for eth0: 172.31.24.68
Swap usage: 0%

```

Update the instance:

```

ubuntu@ip-172-31-24-68:~$ sudo apt-get update
Hit:1 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 http://ap-southeast-1.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-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:9 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:10 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:11 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2023 kB]

```

Install apache2:

```

ubuntu@ip-172-31-24-68:~$ 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 liblua5.3-0 mailcap mime-support ssl-cert
Suggested packages:
apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser bzip2-doc
The following NEW packages will be installed:
apache2 apache2-bin apache2-data apache2-utils bzip2 libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.3-0 mailcap mime-support ssl-cert
0 upgraded, 13 newly installed, 0 to remove and 69 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

```

2. Use the following command to edit the <title> tags in the /var/www/html/index.html file to display the instance name:

```

sudo nano /var/www/html/index.html

```

```

GNU nano 6.2
/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>Instance: 23905652-vm2</title>
<style type="text/css" media="screen">
* {
    margin: 0px 0px 0px 0px;
    padding: 0px 0px 0px 0px;
}

```

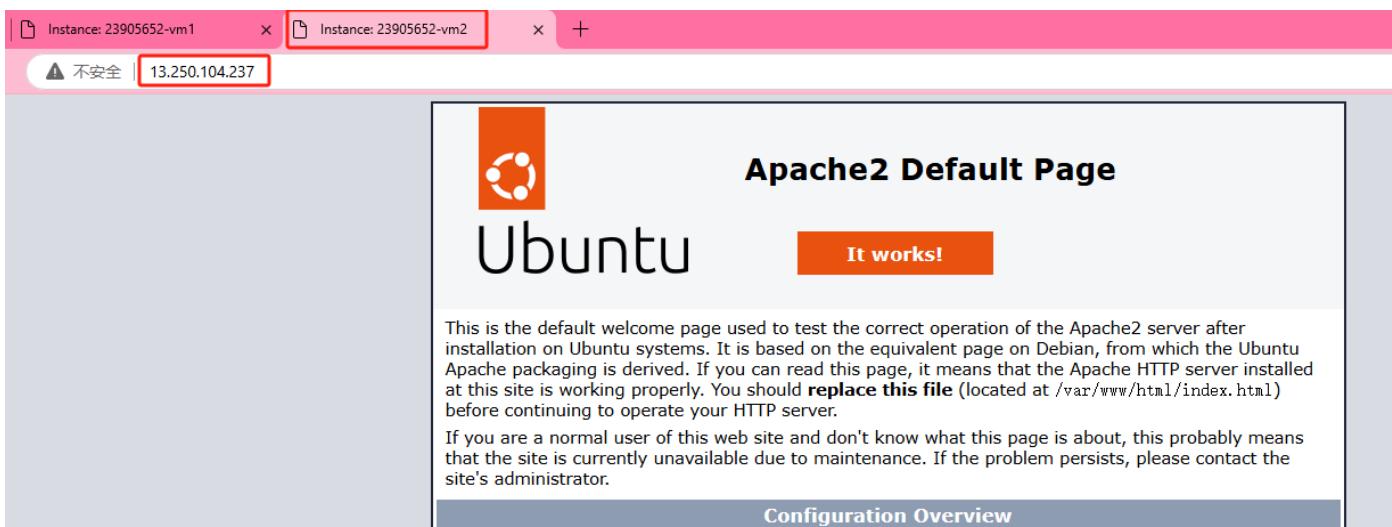
Save and exit according to the prompt. For the other instance, follow the same steps and change the title to "23905652-vm1".

3. Open a browser and access each instance using their respective **IP addresses** to see the default Apache webpage with their respective titles:

[http://54.255.225.54/ \(23905652-vm1\)](http://54.255.225.54/) :



[http://13.250.104.237/ \(23905652-vm2\)](http://13.250.104.237/) :



4. To verify the ALB, we access its **DNS name** <http://23905652-alb-1210941414.ap-southeast-1.elb.amazonaws.com/> in a browser and perform **several refreshes to test accessing different instances**:

05652-vm1 x | Instance: 23905652-vm2 x | Instance: 23905652-vm1 x | +

| 23905652-alb-1210941414.ap-southeast-1.elb.amazonaws.com



Apache2 Default Page

Ubuntu

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at /var/www/html/index.html) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

05652-vm1 x | Instance: 23905652-vm2 x | Instance: 23905652-vm1 x | +

| 23905652-alb-1210941414.ap-southeast-1.elb.amazonaws.com



Apache2 Default Page

Ubuntu

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at /var/www/html/index.html) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.