



# Assignment

Application Support / DevOps

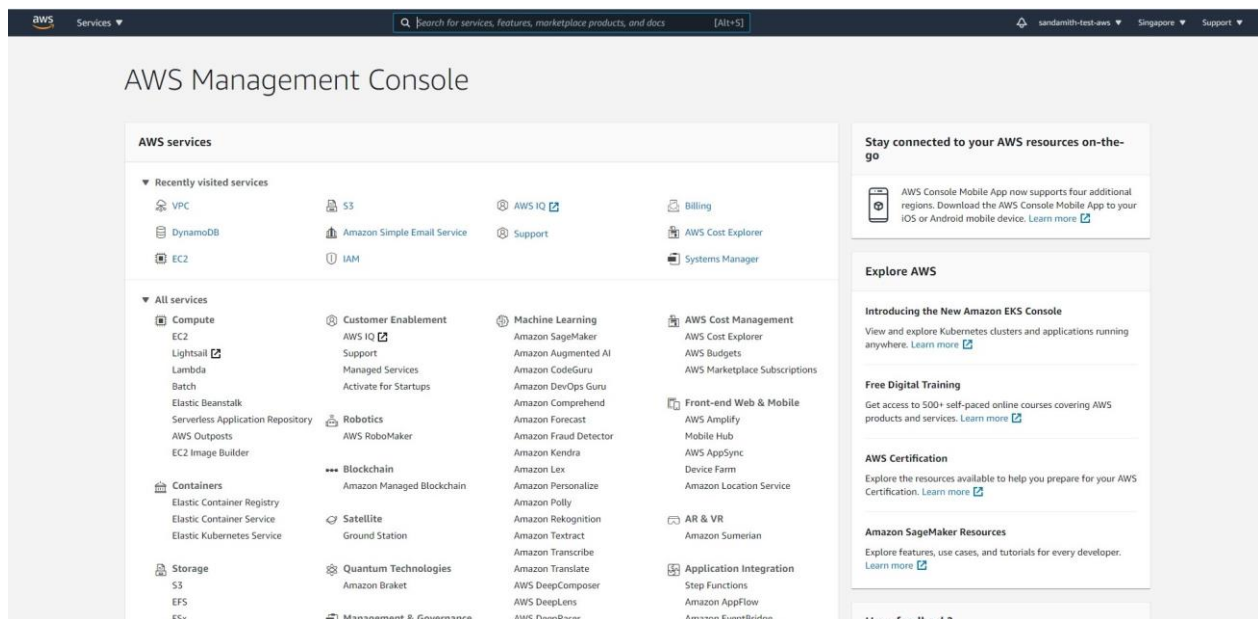
Sachintha Sandamith Wanigasuriya

This document consists of following sections.

- A) Implementing the Server
- B) Implementing Apache Web Server
- C) Scripts
- D) Testing & Results

*AWS services are used for the tasks of this assignment.*

## Section A: Implementing the Server



- First, a VPC was created in my AWS console to run the web server.

aws

Services

Search for services, features, marketplace products, and docs

VPC

>

Your VPCs

>

Create VPC

## Create VPC [Info](#)

A VPC is an isolated portion of the AWS cloud populated by AWS objects, such as Amazon EC2 instances.

### VPC settings

**Name tag - optional**  
Creates a tag with a key of 'Name' and a value that you specify.

vpc-web-server

**IPv4 CIDR block** [Info](#)

10.0.0.0/16

**IPv6 CIDR block** [Info](#)

☒ No IPv6 CIDR block

☐ Amazon-provided IPv6 CIDR block

☐ IPv6 CIDR owned by me

**Tenancy** [Info](#)

Default

### Tags

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key

Value - optional

Q Name X

Q vpc-web-server X

Remove

Add new tag

You can add 49 more tags.

aws

Services

Search for services, features, marketplace products, and docs

[Alt+S]

sandamh-test-aws

Singapore

New VPC Experience  
Tell us what you think

VIRTUAL PRIVATE CLOUD

Your VPCs [New](#)

Subnets [New](#)

Route Tables

Internet Gateways [New](#)

Egress Only Internet Gateways [New](#)

DHCP Options Sets [New](#)

Elastic IPs [New](#)

Managed Prefix Lists [New](#)

Endpoints

Endpoint Services

NAT Gateways [New](#)

Peering Connections

SECURITY

Network ACLs [New](#)

Security Groups [New](#)

REACHABILITY

Reachability Analyzer

VIRTUAL PRIVATE NETWORK (VPN)

Customer Gateways

Virtual Private Gateways

Site-to-Site VPN Connections

Client VPN Endpoints

VPC

>

Your VPCs

>

vpc-0615e03644288f215

## vpc-0615e03644288f215 / vpc-web-server [Actions](#)

### Details [Info](#)

|                                 |                                   |   |   |
|---------------------------------|-----------------------------------|---|---|
| VPC ID<br>vpc-0615e03644288f215 | State<br><span>Available</span>   | DNS hostnames<br>Disabled                 | DNS resolution<br>Enabled                 |
| Tenancy<br>Default              | DHCP options set<br>dopt-f23bc894 | Main route table<br>rtb-0961e75b8ba20f5f1 | Main network ACL<br>acl-0fad8befc52892007 |
| Default VPC<br>No               | IPv4 CIDR<br>10.0.0.0/16          | IPv6 pool<br>-                            | IPv6 CIDR<br>-                            |
| Owner ID<br>345707855562        |                                   |   |   |

CIDRs

Flow logs

Tags

### IPv4 CIDRs [Info](#)

| CIDR        | Status                  |
|-------------|-------------------------|
| 10.0.0.0/16 | <span>Associated</span> |

### IPv6 CIDRs [Info](#)

| CIDR | Pool | Status |
|------|------|--------|
|      |      |        |

You have no IPv6 CIDR blocks associated with your VPC.

- Then a subnet was created for the VPC. I am using a Singapore region AZ.

aws

Services

Search for services, features, r

VPC > Subnets > Create subnet

## Create subnet [Info](#)

### VPC

VPC ID  
Create subnets in this VPC.

vpc-0615e03644288f215 (vpc-web-server)

Associated VPC CIDRs

IPv4 CIDRs

10.0.0.0/16

### Subnet settings

Specify the CIDR blocks and Availability Zone for the subnet.

#### Subnet 1 of 1

Subnet name  
Create a tag with a key of 'Name' and a value that you specify.

web-server-subnet

The name can be up to 256 characters long.

Availability Zone [Info](#)  
Choose the zone in which your subnet will reside, or let Amazon choose one for you.

Asia Pacific (Singapore) / ap-southeast-1a

IPv4 CIDR block [Info](#)

10.0.1.0/24

Tags - optional

| Key  | Value - optional  |        |
|------|-------------------|--------|
| Name | web-server-subnet | Remove |

VPC > Subnets > subnet-07017528e61856c27

subnet-07017528e61856c27 / web-server-subnet Actions

### Details

|                                       |   |   |                                       |
|---------------------------------------|---|---|---------------------------------------|
| Subnet ID<br>subnet-07017528e61856c27 | State<br>Available  | VPC<br>vpc-0615e03644288f215   vpc-web-server | IPv4 CIDR<br>10.0.1.0/24              |
| Available IPv4 addresses<br>251       | IPv6 CIDR<br>-  | Availability Zone<br>ap-southeast-1a          | Availability Zone ID<br>apse1-az2     |
| Route table<br>rtb-0961e75b8ba20f5f1  | Network ACL<br>acl-0fad8betc52892007  | Default subnet<br>No                          | Auto-assign public IPv4 address<br>No |
| Auto-assign IPv6 address<br>No        | Auto-assign customer-owned IPv4 address<br>No   | Customer-owned IPv4 pool<br>-                 | Outpost ID<br>-                       |
| Owner<br>345707855562                 | Subnet ARN<br>arn:aws:ec2:ap-southeast-1:345707855562:subnet/subnet-07017528e61856c27 |   |                                       |

Flow logs

Route table

Network ACL

Sharing

Tags

### Flow logs

Filter flow logs

1

| Name | Flow log ID | Filter | Destination type | Destination name | IAM role ARN |
|------|-------------|--------|------------------|------------------|--------------|
|------|-------------|--------|------------------|------------------|--------------|

- A security group was created for my VPC to filter incoming traffic.

The screenshot shows the 'Create security group' page in the AWS Management Console. The breadcrumb navigation is 'VPC > Security Groups > Create security group'. The page title is 'Create security group' with an 'Info' link. A subtitle states: 'A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.' The 'Basic details' section contains three fields: 'Security group name' (value: 'web-server-sg', with a note 'Name cannot be edited after creation.'), 'Description' (value: 'Web Server Security Group'), and 'VPC' (value: 'vpc-0615e03644288f215 (vpc-web-server)'). The 'Inbound rules' section shows a message: 'This security group has no inbound rules.' and an 'Add rule' button.

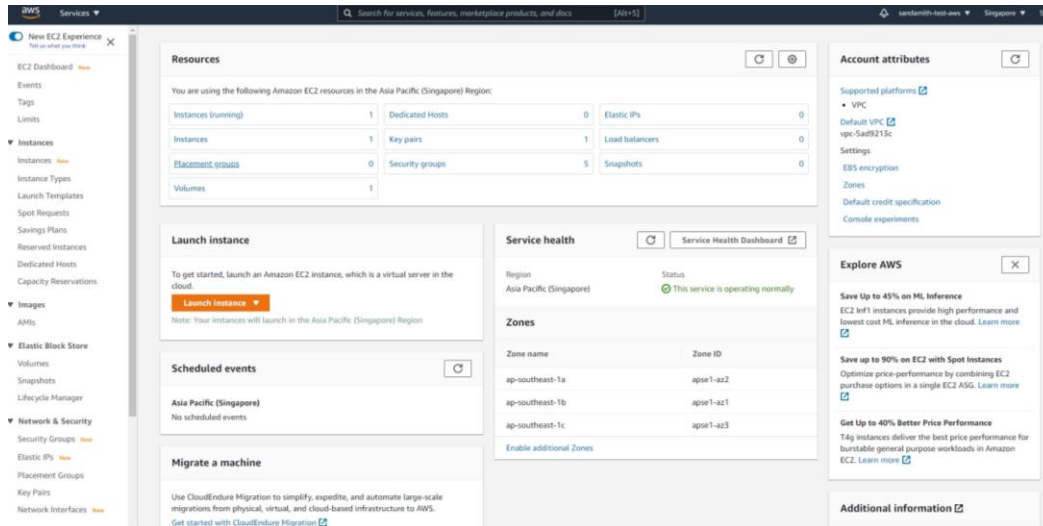
- Inbound rules were added.
- HTTP from anywhere and SSH from anywhere.
- Even though it's not a good practice to allowing SSH from any IP, I configured it like this for only testing environments.

The screenshot shows the 'Edit inbound rules' page in the AWS Management Console. The breadcrumb navigation is 'VPC > Security Groups > sg-092615fa80f55a79b - web-server-sg > Edit inbound rules'. The page title is 'Edit inbound rules' with an 'Info' link. A subtitle states: 'Inbound rules control the incoming traffic that's allowed to reach the instance.' The 'Inbound rules' section displays a table with two rules:

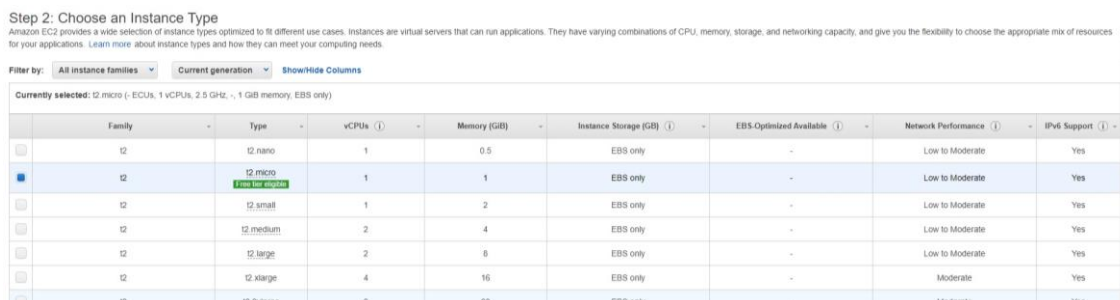
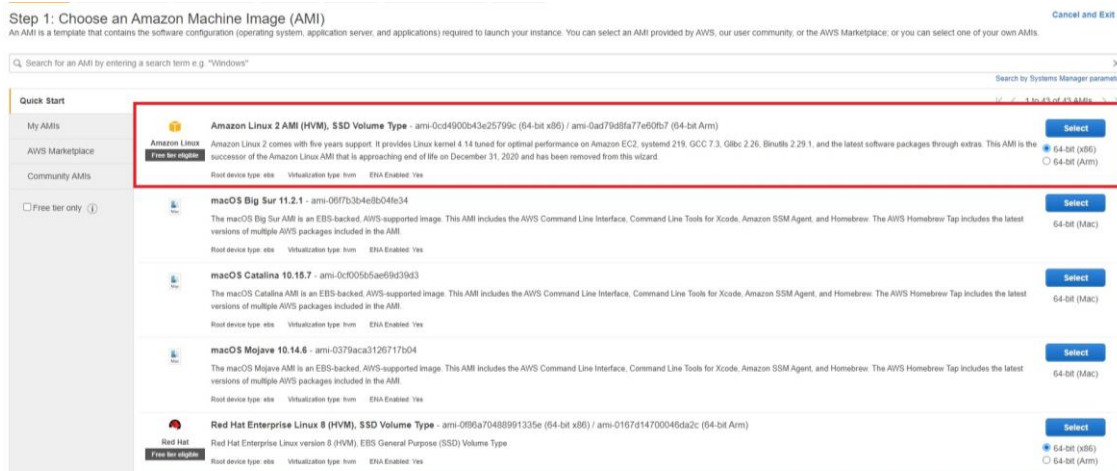
| Type | Protocol | Port range | Source   | Description - optional   |        |
|------|----------|------------|----------|--|--------|
| HTTP | TCP      | 80         | Anywhere | This allows access for anyone to website                       | Delete |
| SSH  | TCP      | 22         | Anywhere | This is for SSH. I am selecting 'everywhere' only for testing. | Delete |

Below the table is an 'Add rule' button. A note at the bottom states: 'NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.' At the bottom right are 'Cancel', 'Preview changes', and 'Save rules' buttons.

- An EC2 instance was created to launch the web server within it.



- I chose Amazon Linux 2 as the AMI for my EC2 instance.



- Previously created VPC and subnet were assigned to the instance. Public IP auto-assigning was also enabled.

**aws** Services ▼  [Alt+S]

1. Choose AMI 2. Choose Instance Type 3. **Configure Instance** 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

### Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance

**Number of instances**  [Launch into Auto Scaling Group](#)

**Purchasing option** ☐ Request Spot instances

**Network**  [Create new VPC](#)

**Subnet**  [Create new subnet](#)

251 IP Addresses available

**Auto-assign Public IP**

**Placement group** ☐ Add instance to placement group

**Capacity Reservation**

**Domain join directory**  [Create new directory](#)

**IAM role**  [Create new IAM role](#)

**CPU options** ☐ Specify CPU options

**Shutdown behavior**

**Stop - Hibernate behavior** ☐ Enable hibernation as an additional stop behavior

**Enable termination protection** ☐ Protect against accidental termination

**Monitoring** ☐ Enable CloudWatch detailed monitoring  
Additional charges apply.

**Tenancy**    
Additional charges will apply for dedicated tenancy.

**Credit specification** ☐ Unlimited  
Additional charges may apply

### Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more about storage options in Amazon EC2.](#)

| Volume Type | Device    | Snapshot                | Size (GiB) | Volume Type               | IOPS       | Throughput (MiB/s) | Delete on Termination               | Encryption    |
|-------------|-----------|-------------------------|------------|---------------------------|------------|--------------------|-------------------------------------|---------------|
| Root        | /dev/xvda | snapp-006dc16994cd31ade | 8          | General Purpose SSD (gp2) | 100 / 3000 | N/A                | <input checked="" type="checkbox"/> | Not Encrypted |

[Add New Volume](#)

Free tier eligible customers can get up to 30 GiB of EBS General Purpose (SSD) or Magnetic storage. [Learn more about free usage tier eligibility and usage restrictions.](#)

- Security group I created earlier was added to the instance.

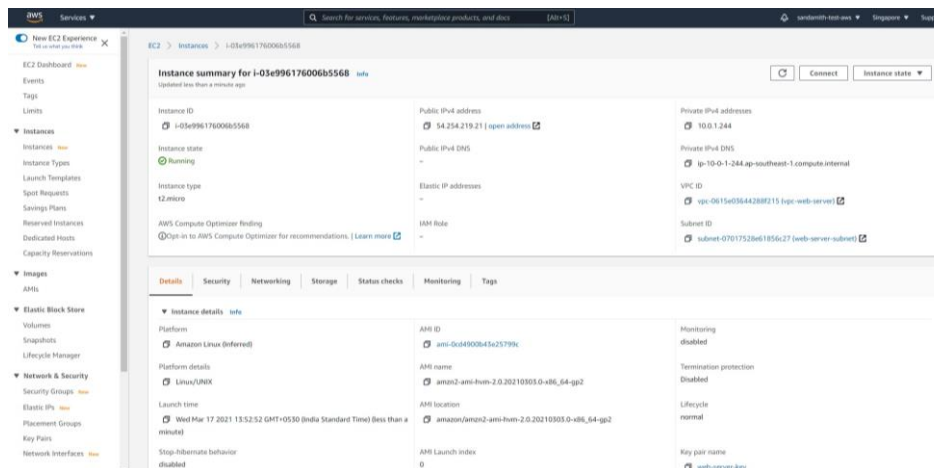
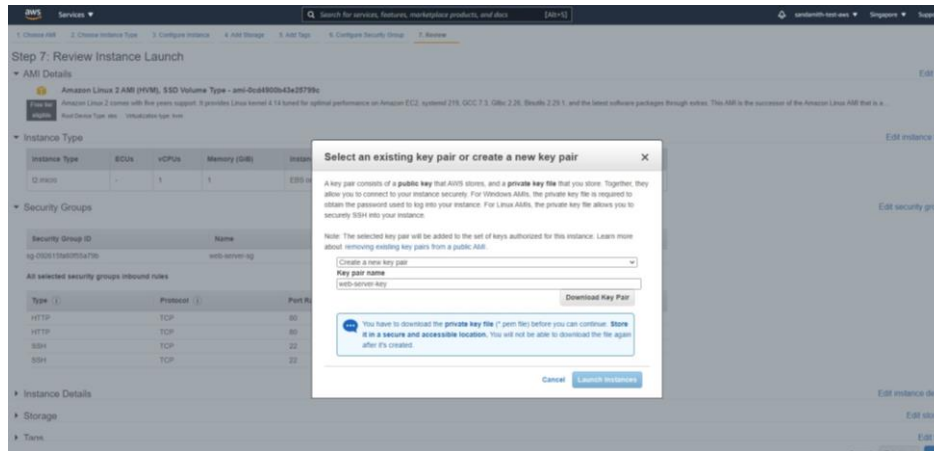
### Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more about Amazon EC2 security groups.](#)

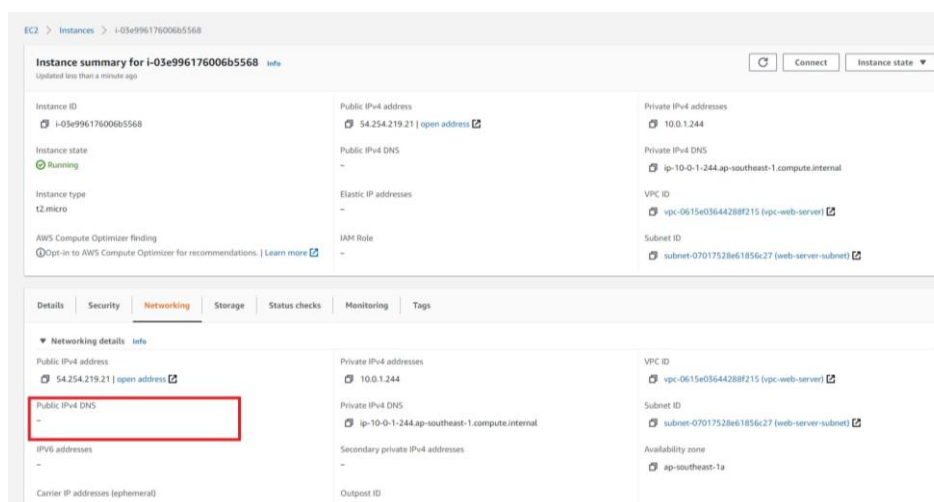
**Assign a security group:** ☐ Create a new security group ☒ Select an existing security group

| Security Group ID  | Name          | Description                | Actions                     |
|--|---------------|----------------------------|-----------------------------|
| <input type="checkbox"/> sg-03d91b7d7df11bcf2            | default       | default VPC security group | <a href="#">Copy to new</a> |
| <input checked="" type="checkbox"/> sg-092615fa80f55a79b | web-server-sg | Web Server Security Group  | <a href="#">Copy to new</a> |

- EC2 instance was launched after creating a new key and downloading it.

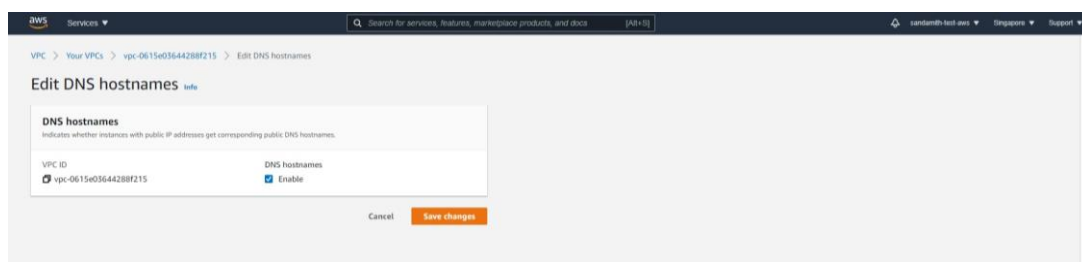
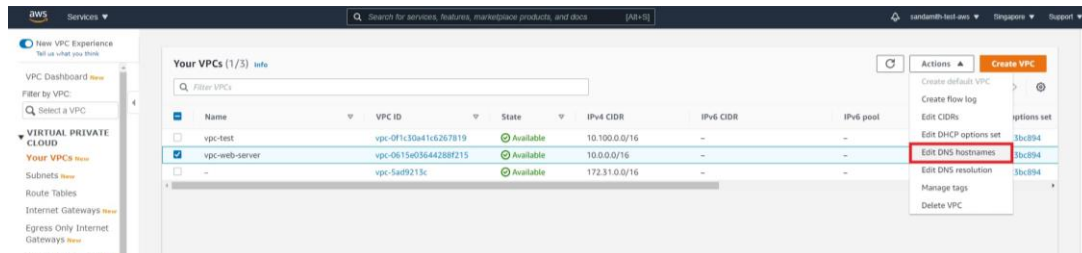


- When we look at our public DNS for our instance, it's not available.

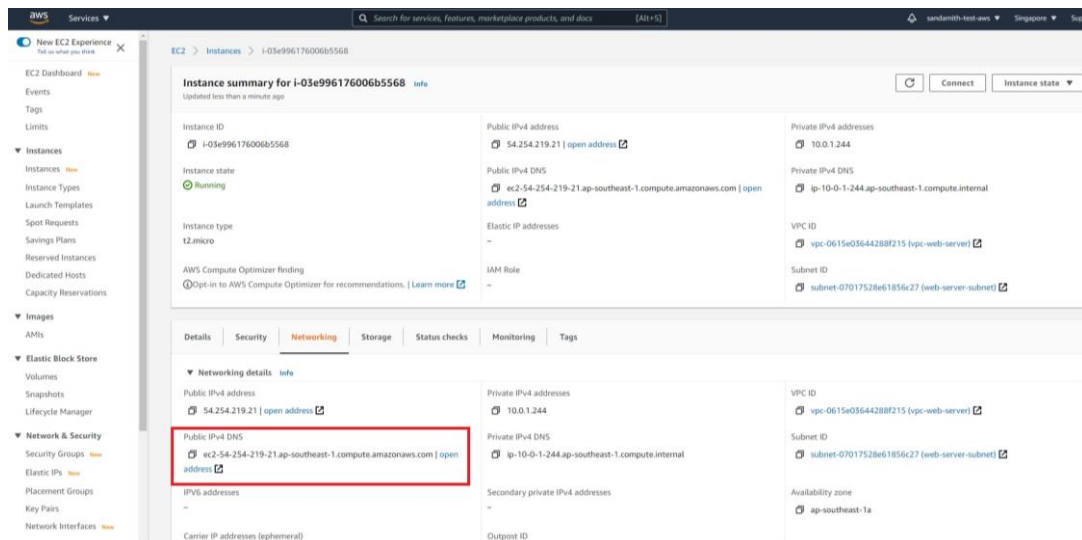




- To make this available, DNS hostnames were enabled in VPS settings.



- Public DNS is available now.



- Our VPC has no connectivity to internet. Therefore, an internet gateway was created and our VPC was attached to it.

VPC > Internet gateways > Create internet gateway

## Create internet gateway [Info](#)

An internet gateway is a virtual router that connects a VPC to the internet. To create a new internet gateway specify the name for the gateway below.

### Internet gateway settings

**Name tag**  
Creates a tag with a key of 'Name' and a value that you specify.

### Tags - optional

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

| Key                               | Value - optional                            |                                       |
|-----------------------------------|---|---------------------------------------|
| <input type="text" value="Name"/> | <input type="text" value="web-server-igw"/> | <input type="button" value="Remove"/> |

You can add 49 more tags.

VPC > Internet gateways > Attach to VPC (igw-02b00b384298499bd)

## Attach to VPC (igw-02b00b384298499bd) [Info](#)

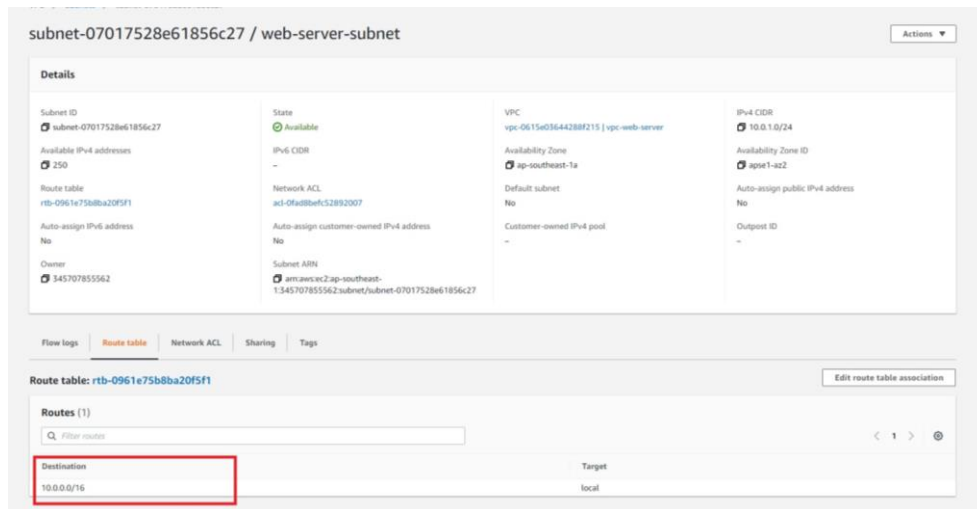
### VPC

Attach an internet gateway to a VPC to enable the VPC to communicate with the internet. Specify the VPC to attach below.

**Available VPCs**  
Attach the internet gateway to this VPC.

► AWS Command Line Interface command

- When we look at our subnet route tables, there are no routes to connect with internet gateway.



- So, a route table was created for our VPC to allocate routes.

Route Tables > Create route table

### Create route table

A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.

Name tag web-server-rt

VPC\* vpc-0615e03644288f215

Key (128 characters maximum)

Value (256 characters maximum)

This resource currently has no tags

Add Tag 50 remaining (Up to 50 tags maximum)

\* Required

Cancel Create

- And a route was added for the internet gateway.

Route Tables > Edit routes

### Edit routes

| Destination | Target                | Status | Propagated |
|-------------|-----------------------|--------|------------|
| 10.0.0.0/16 | local                 | active | No         |
| 0.0.0.0/0   | igw-02b00b384298499bd |        | No         |

Add route

\* Required

Cancel Save routes

- The route table we created was attached to our subnet from route table association.

The screenshot shows the AWS Management Console interface. At the top, there's a 'Subnets (1/5)' header with a search bar and a 'Create subnet' button. Below this is a table listing subnets. The first subnet, 'web-server-subnet' (subnet-07017528e61856c27), is selected. Below the table, the 'Route table' tab is active for the selected subnet. It shows a route table 'rtb-0961e75b8ba20f5f1' with a single route for destination '10.0.0.0/16' pointing to 'local'. A red box highlights the 'Edit route table association' button in the top right corner of the route table configuration area.

| Name              | Subnet ID                | State     | VPC                            | IPv4 CIDR      | IPv6 CIDR | Available IPv4 addresses |
|-------------------|--------------------------|-----------|--------------------------------|----------------|-----------|--------------------------|
| web-server-subnet | subnet-07017528e61856c27 | Available | vpc-0615e03644288f215   vp...  | 10.0.1.0/24    | -         | 250                      |
| test-subnet       | subnet-00f334ba60479ecda | Available | vpc-0f1c30a41c6267819   vpc... | 10.100.1.0/24  | -         | 250                      |
| -                 | subnet-1905d851          | Available | vpc-5ad9213c                   | 172.31.32.0/20 | -         | 4091                     |
| -                 | subnet-7d339e1b          | Available | vpc-5ad9213c                   | 172.31.16.0/20 | -         | 4091                     |
| -                 | subnet-3b72ea62          | Available | vpc-5ad9213c                   | 172.31.0.0/20  | -         | 4091                     |

subnets-07017528e61856c27 / web-server-subnet

Details | Flow logs | **Route table** | Network ACL | Sharing | Tags

Route table: **rtb-0961e75b8ba20f5f1**

Routes (1)

| Destination | Target |
|-------------|--------|
| 10.0.0.0/16 | local  |

The screenshot shows the 'Edit route table association' page. It has a breadcrumb trail: 'VPC > Subnets > subnet-07017528e61856c27 > Edit route table association'. The page title is 'Edit route table association'. Under 'Subnet route table settings', the 'Subnet ID' is 'subnet-07017528e61856c27' and the 'Route table ID' is 'rtb-0935148077dead877 (web-server-rt)'. Below this, under 'Routes (2)', there are two routes: one for destination '10.0.0.0/16' pointing to 'local', and another for destination '0.0.0.0/0' pointing to 'igw-02b00b384298499bd'. At the bottom right, there are 'Cancel' and 'Save' buttons.

VPC > Subnets > subnet-07017528e61856c27 > Edit route table association

### Edit route table association

**Subnet route table settings**

Subnet ID  
subnet-07017528e61856c27

Route table ID  
rtb-0935148077dead877 (web-server-rt)

**Routes (2)**

| Destination | Target                |
|-------------|-----------------------|
| 10.0.0.0/16 | local                 |
| 0.0.0.0/0   | igw-02b00b384298499bd |

Cancel Save

- ✓ Now everything within AWS console has been configured. Our server is up and running.

## Section B: Implementing Apache Web Server

*I am using Kali-Linux environment on my local machine for further configurations and executions of the server and scripts.*

- First, I logged in to the server I created inside AWS console via SSH using the downloaded key-pair and public DNS.

```
godzilla@kali: ~/Documents/web server
File Actions Edit View Help
(godzilla@kali)~[~/Documents/web server]
$ sudo ssh -i web-server-key.pem ec2-user@ec2-54-254-219-21.ap-southeast-1.compute.amazonaws.com

godzilla@kali: ~/Documents/web server
File Actions Edit View Help
(godzilla@kali)~[~/Documents/web server]
$ sudo ssh -i web-server-key.pem ec2-user@ec2-54-254-219-21.ap-southeast-1.compute.amazonaws.com
[sudo] password for godzilla:
The authenticity of host 'ec2-54-254-219-21.ap-southeast-1.compute.amazonaws.com (54.254.219.21)' can't be established.
ECDSA key fingerprint is SHA256:XQeCrwKWE/K7Wk//BsAY5pvcH4G7DMXLwJRfCgh+8Ls.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-254-219-21.ap-southeast-1.compute.amazonaws.com,54.254.219.21' (ECDSA) to the list of known hosts.

  _ | ( _ | )
  _||_ / _||_
  _||_ / _||_

Amazon Linux 2 AMI

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-1-244 ~]$
```

- PHP software was installed.

```
File Actions Edit View Help
[ec2-user@ip-10-0-1-244 ~]$ sudo amazon-linux-extras install -y lamp-mariadb10.2-php7.2 php7.2
```

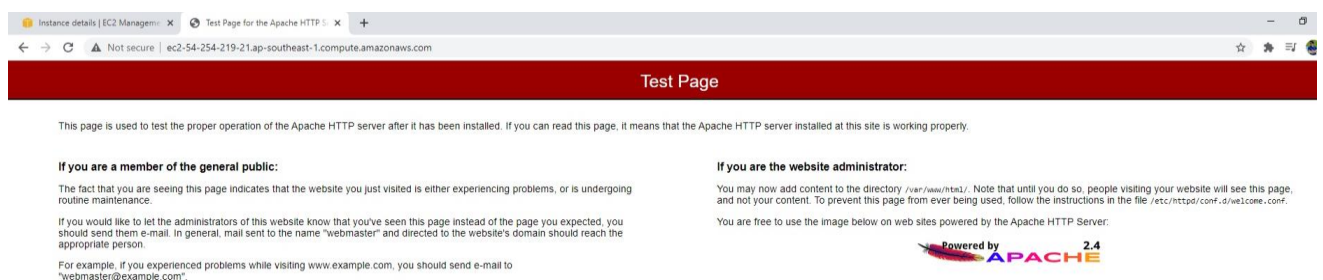
- Apache web server was installed on server after that.

```
File Actions Edit View Help
[ec2-user@ip-10-0-1-244 ~]$ sudo yum install -y httpd
```

- Then started the apache service.

```
File Actions Edit View Help
[ec2-user@ip-10-0-1-244 ~]$ sudo systemctl start httpd
[ec2-user@ip-10-0-1-244 ~]$
```

- Tested if the apache is properly installed and working.



- Configured the web server to start with each system boot.

```
File Actions Edit View Help
[ec2-user@ip-10-0-1-244 ~]$ sudo systemctl enable httpd
```

- Added ec2-user to apache group.

```
File Actions Edit View Help
[ec2-user@ip-10-0-1-244 ~]$ sudo usermod -a -G apache ec2-user
[ec2-user@ip-10-0-1-244 ~]$
```

- Changed group ownership of /var/www directory and its contents to apache group.

```
File Actions Edit View Help
[ec2-user@ip-10-0-1-244 ~]$ sudo chown -R ec2-user:apache /var/www
[ec2-user@ip-10-0-1-244 ~]$
```

- Changed the directory and file permissions of /var/www and its subdirectories recursively to give permissions to future members of apache group.

```
File Actions Edit View Help
[ec2-user@ip-10-0-1-244 ~]$ sudo chmod 2775 /var/www
[ec2-user@ip-10-0-1-244 ~]$ find /var/www -type d -exec sudo chmod 2775 {} \;
[ec2-user@ip-10-0-1-244 ~]$ find /var/www -type f -exec sudo chmod 0664 {} \;
[ec2-user@ip-10-0-1-244 ~]$
```

- I created a php file inside /var/www/html to deploy content to the web server. It's a simple program to show hello world.

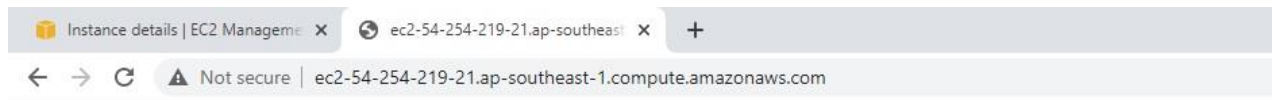
```
File Actions Edit View Help
<!DOCTYPE html>
<html>
<body>

<h1>Hellow World !</h1>

<?php
echo "Web server is working properly";
?>

</body>
</html>
~
~
~
```

- Finally, I checked whether it's working.



- ☺ Apache web server is running and serving content properly.



## Section C: Scripts

*I created 2 main scripts to carry out the tasks of this assignment. There are 2 other scripts running inside these main scripts.*

*AWS DynamoDB NoSQL database is used to save the results of the script. AWS S3 storage service is used to upload the server datafile.*

### Snapshots & tasks of the scripts

#### 1) First Script (script1.sh)

- Accessing public server via SSH.
- Checking whether web-server is running and starting it if it's not.
- Checking whether web-server is serving the expected content (With HTTP status code 200).

```
Terminal
File Edit View Search Terminal Help
#!/bin/bash

#This is the first script.
#This script runs inside run-script1.sh

#Purpose of this script :

#Access the public server via ssh.
#Check if web-server is running and start if it is not.
#Check if web-server serving expected content.

#Author - Sachintha Sandamith Wanigasuriya

#####

#Public DNS of ec2-instance.

dns=ec2-13-229-146-238.ap-southeast-1.compute.amazonaws.com

#Connect to ec2-instance via ssh.

sudo ssh -i test-key.pem ec2-user@$dns<<EOF

echo "Login_success"

#Checking if web-server is running and start it if is not.

if [ "`sudo systemctl is-active httpd`" = "inactive" ]; then
    sudo systemctl start httpd && echo "Apache_launch_success" 2>>data/errors.txt
    sleep 3
fi

#Checking if web-server is serving expected content with the http status code 200

if [ `curl -o -I -L -s -w "%{http_code}" $dns` -eq 200 ]; then
    echo "Web_server_content_check_success" 2>>data/errors.txt
fi

EOF
```

## 2) Main Script – 1 (run-script1.sh)

- Executing the **first script** and creating a logfile and an error file.
- Uploading first script results to DynamoDB with timestamps. (Epoch timestamp is used).
- Sending mail to Application-Support team if script detects errors.

```
Terminal
File Edit View Search Terminal Help
#!/bin/bash

#This is the main script-1
#Purpose of this script :
#Run the first script & record its results and errors.
#Upload the results of first script to AWS DynamoDB with timestamps.
#Send mail to application-support team if script detects any errors.
#Author - Sachintha Sandamith Wanigasuriya
#####
#Results of first script will be saved to a file named "logfile" and execution errors will be redirected to script_errors/localerrors_1.txt
>logfile
>script_errors/localerrors_1.txt
#Run the first script while creating the logfile along with timestamps. Epoch timestamp is used here.
./script1.sh | while IFS= read -r line; do printf '%s%s\n' "$(date +%s)" "$line"; done >>logfile 2>script_errors/localerrors_1.txt && echo "Web server is running and serving content properly"
#Read logfile line by line while uploading them into DynamoDB table.
while IFS= read -r line; do
    IFS='-' read -ra ARR <<< "$line"
    aws dynamodb put-item --table-name server_logs --item '{"timestamp":{"N":"'${ARR[0]}'"},"log":{"S":"'${ARR[1]}'"},"' --return-consumed-capacity TOTAL
done < logfile 2>>script_errors/localerrors_1.txt && echo "Log files uploaded to DB successfully"
#Fetch the error file created within ec2-instance by first script.
sudo scp -i test-key.pem ec2-user@ec2-13-229-146-238.ap-southeast-1.compute.amazonaws.com:/home/ec2-user/data/errors.txt script_errors/ec2errors.txt 2>>script_errors/localerrors_1.txt && echo "Error file in ec-2 copied"
#Mail to application-support if any error found. Else complete the execution.
if [ -s /script_errors/ec2errors.txt ]
```

## 3) Second Script (script2.sh)

- Copying content of the web-server.
- Copying Apache web-server log files. (Access log & Error log)
- Creating one compressed file from the files copied.

```
Terminal
File Edit View Search Terminal Help
#!/bin/bash

#This is the second script.
#This script runs inside run-script2.sh

#Purpose of this script :

#Copy content of the web-server.
#Copy Apache web-server log files.
#Create one compressed file.

#Author - Sachintha Sandamith Wanigasuriya
#####

#Public DNS for ec-2 instance

dns=ec2-13-229-146-238.ap-southeast-1.compute.amazonaws.com

#Connect to ec2-instance via ssh.

sudo ssh -i test-key.pem ec2-user@$dns <<EOF

#Copy content of the web-server to a directory created by me within ec2-instance.

sudo cp /var/www/html/index.php /home/ec2-user/data/ && echo "Web-server content copied"

#Copy apache access_log file to my directory.

sudo cp /etc/httpd/logs/access_log /home/ec2-user/data/ && echo "Apache access-log copied"

#Copy apache error_log file to my directory.

sudo cp /etc/httpd/logs/error_log /home/ec2-user/data/ && echo "Apache error-log copied"

#Compress content file and log files into one file.

zip data/web_server_data.zip data/index.php data/access_log data/error_log && echo "Compressed data file created"

EOF
~
```

#### 4) Main Script – 2 (run-script2.sh)

- Executing the **second script**.
- Fetching the compressed file created by second script to the script location.
- Uploading compressed file to the AWS S3 bucket.
- Sending mail to Application-Support team if any error occurred.

```
Terminal
File Edit View Search Terminal Help
#!/bin/bash

#This is the main script-2

#Purpose of this script :

#Run the second script.
#Fetch the second script output; compressed web-server data file to the script location.
#Upload the data file to AWS s3 bucket.
#Send mail to the application-support team if any error occurred.

#Author - Sachintha Sandamith Wanigasuriya

#####

#Execution errors will be redirected to /script_errors/localerrors_2.txt
script_errors/localerrors_2.txt

#Run the second script.

./script2.sh 2>script_errors/localerrors_2.txt

#Fetch the compressed web-server data file from ec2-instance to the script location via scp.

sudo scp -i test-key.pem ec2-user@ec2-13-229-146-238.ap-southeast-1.compute.amazonaws.com:/home/ec2-user/data/web_server_data.zip server_data/ && echo "File downloaded to local machine" 2>>script_errors/localerrors_2.txt

#Upload the fetched file to aws s3 bucket.

aws s3 cp server_data/web_server_data.zip s3://sandamith-bucket-01 && uploadsuccess=true && echo "Data file uploaded to cloud"

#Deleting file upon upload success.

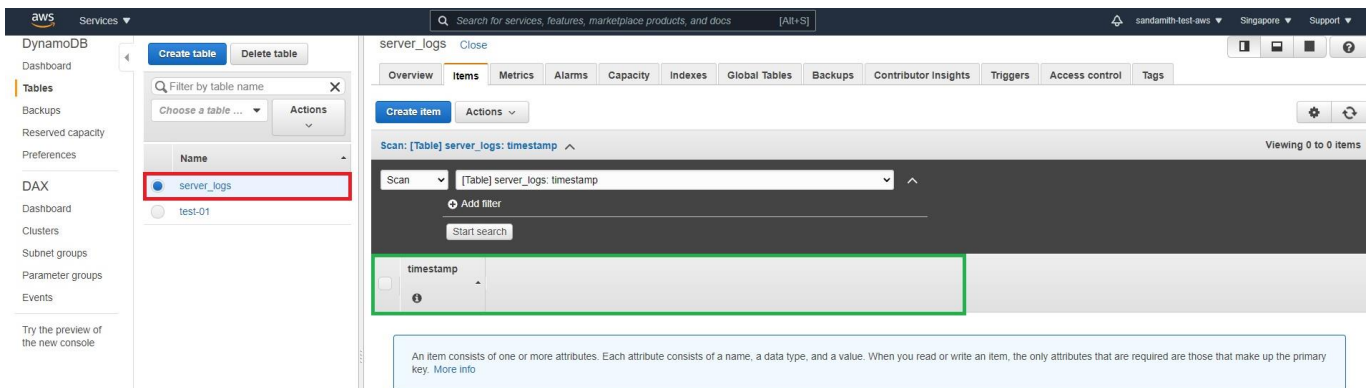
if [ $uploadsuccess = true ]; then
    sudo rm server_data/web_server_data.zip && echo "Local file deleted"
    echo "***** Success *****"
fi

#Sending mail to app-support team upon upload failure.

if [ $uploadsuccess != true ]; then
    php email/error2_send_mail.php && echo "Upload Failed" >>localerrors_2.txt
fi
```

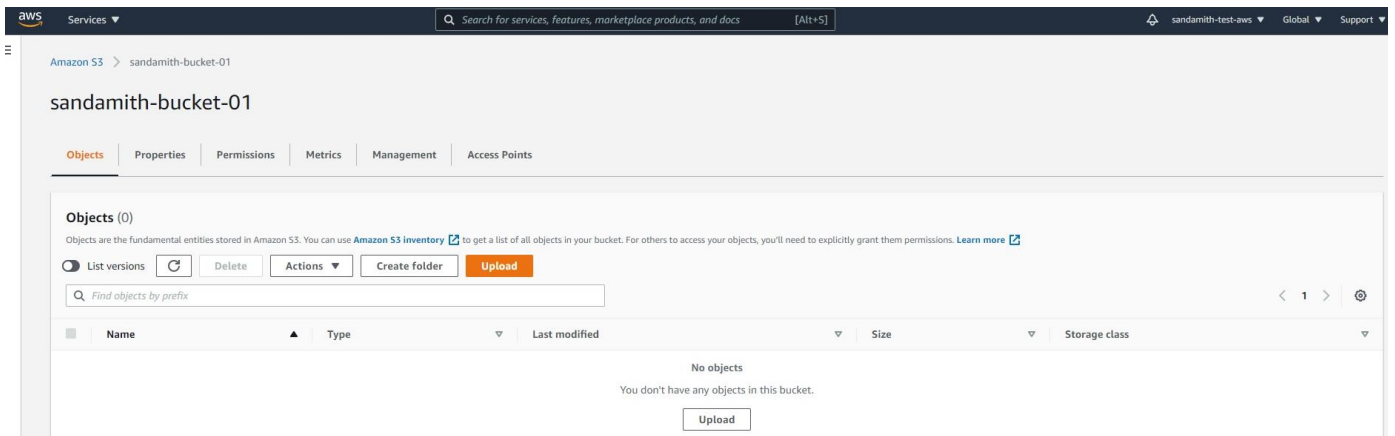
## AWS DynamoDB Table (server\_logs)

- This was created to save script results.



## AWS S3 Bucket (sandamith-bucket-01)

- This was created to upload server data file.



## Scheduling scripts to run periodically

- Two main scripts are scheduled to run as Cron Jobs as root.
- Main Script - 1 is scheduled to run once at every hour.
- Main Script - 2 is scheduled to run at 12AM daily.

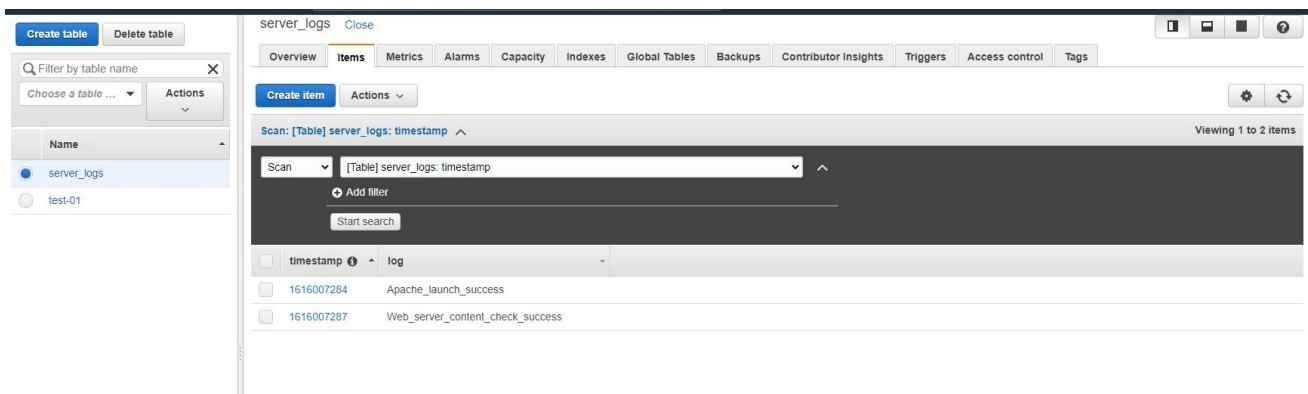
```
godzilla@kali: ~  
File Actions Edit View Help  
# Edit this file to introduce tasks to be run by cron.  
#  
# Each task to run has to be defined through a single line  
# indicating with different fields when the task will be run  
# and what command to run for the task  
#  
# To define the time you can provide concrete values for  
# minute (m), hour (h), day of month (dom), month (mon),  
# and day of week (dow) or use '*' in these fields (for 'any').  
#  
# Notice that tasks will be started based on the cron's system  
# daemon's notion of time and timezones.  
#  
# Output of the crontab jobs (including errors) is sent through  
# email to the user the crontab file belongs to (unless redirected).  
#  
# For example, you can run a backup of all your user accounts  
# at 5 a.m every week with:  
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/  
#  
# For more information see the manual pages of crontab(5) and cron(8)  
#  
# m h dom mon dow command  
  
#Schedule first script to run once every hour  
0 * * * * ./home/godzilla/Desktop/Sandamith-LSEG-Project/run-script1.sh  
  
#Schedule second script to run daily at 12AM  
0 0 * * * ./home/godzilla/Desktop/Sandamith-LSEG-Project/run-script2.sh  
~  
~  
~  
~  
~
```

## Section D: Testing & Results

- Execution and results of Main Script –1 (run-script1.sh)

```
godzilla@kali: ~/Desktop/Sandamith-LSEG-Project
File Actions Edit View Help
(godzilla@kali)~/Desktop/Sandamith-LSEG-Project
$ ./run-script1.sh
[sudo] password for godzilla:
Warning: The file name argument '-i' looks like a flag.
Pseudo-terminal will not be allocated because stdin is not a terminal.
Web server is running and serving content properly
{
  "ConsumedCapacity": {
    "TableName": "server_logs",
    "CapacityUnits": 1.0
  }
}
{
  "ConsumedCapacity": {
    "TableName": "server_logs",
    "CapacityUnits": 1.0
  }
}
{
  "ConsumedCapacity": {
    "TableName": "server_logs",
    "CapacityUnits": 1.0
  }
}
}
Log files uploaded to DB successfully
errors.txt
Error file in ec-2 copied
All is well
***** Success *****
```

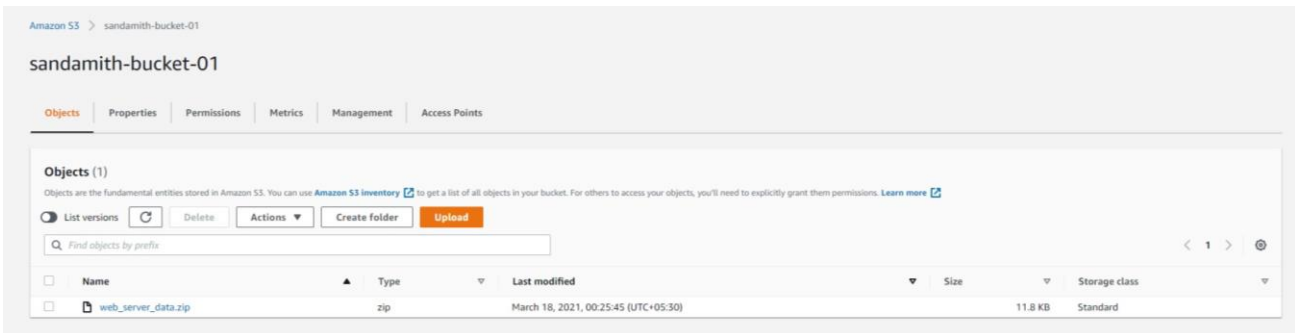
- Script results are successfully saved in DynamoDB table.



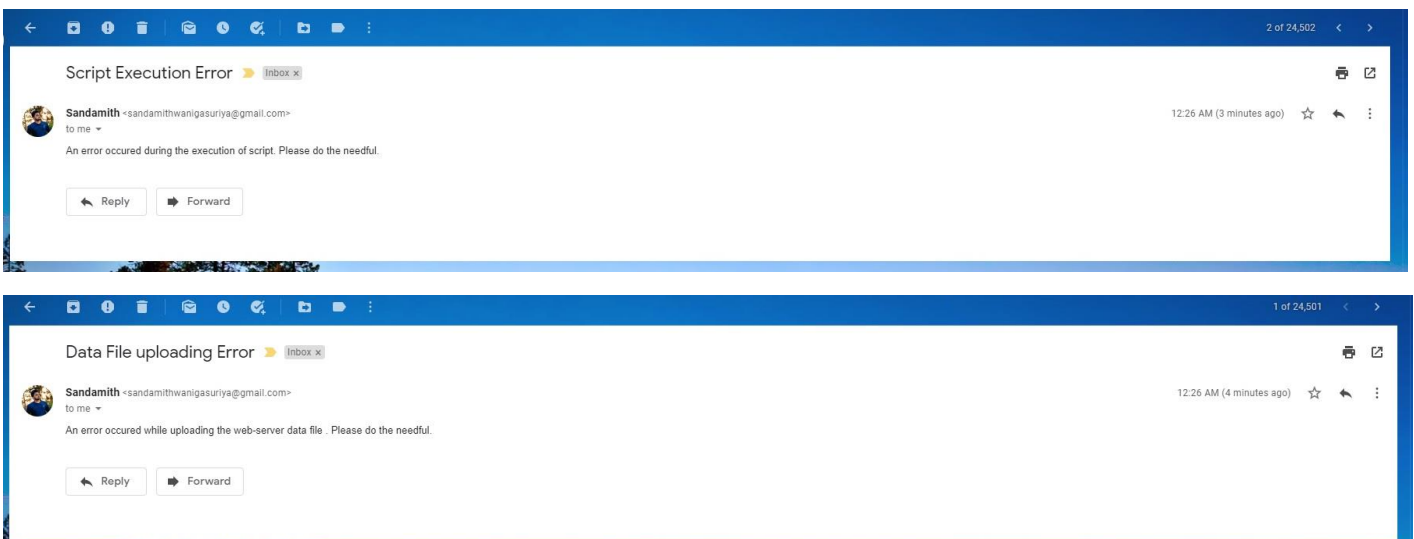
- Execution and results of Main Script –2 (run-script2.sh)

```
godzilla@kali: ~/Desktop/Sandamith-LSEG-Project
File Actions Edit View Help
(godzilla@kali)~/Desktop/Sandamith-LSEG-Project
$ ./run-script2.sh
Web-server content copied
Apache access-log copied
Apache error-log copied
updating: data/index.php (deflated 14%)
updating: data/access_log (deflated 88%)
updating: data/error_log (deflated 89%)
Compressed data file created
web_server_data.zip
File downloaded to local machine
upload: server_data/web_server_data.zip to s3://sandamith-bucket-01/web_server_data.zip
Data file uploaded to cloud
Local file deleted
***** Success *****
```

- Compressed datafile is successfully uploaded to S3 bucket.



- For mailing, I have used PHPMailer using Gmail SMTP server. These are the test e-mails I sent for Application-Support team for both scripts errors.



- Thank you for reading.

\*\*\* End of Assignment \*\*\*