Assignment

Application Support / DevOps

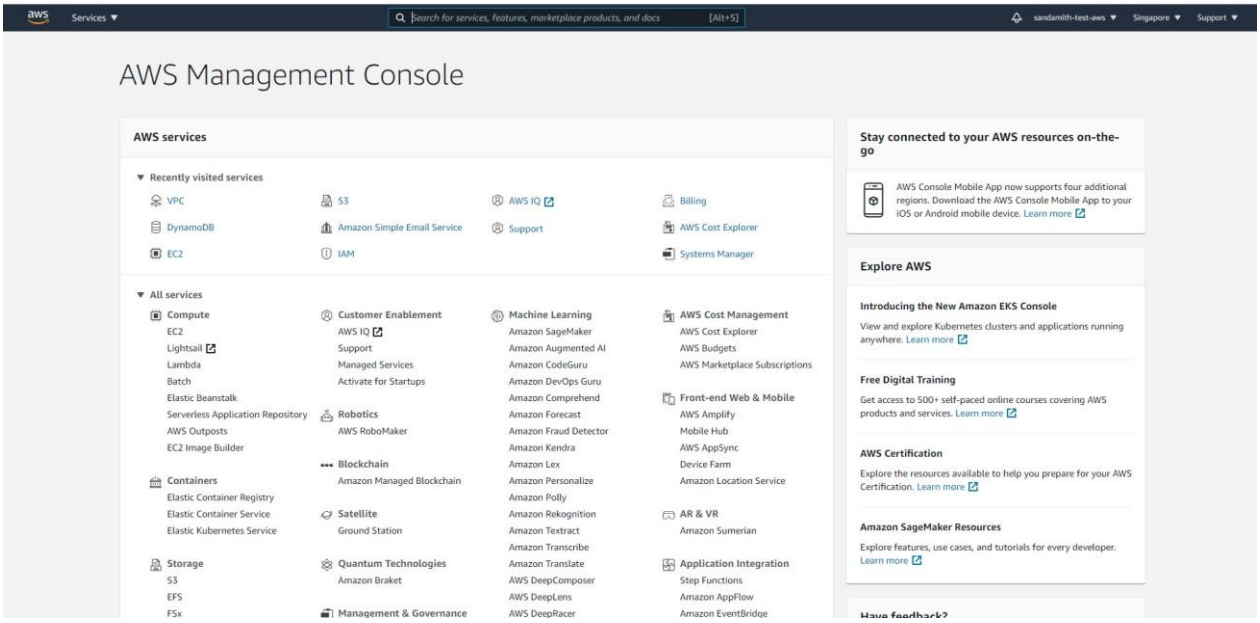
Sachinthya Sandamith Wanigasuriya

This document consists of following sections.

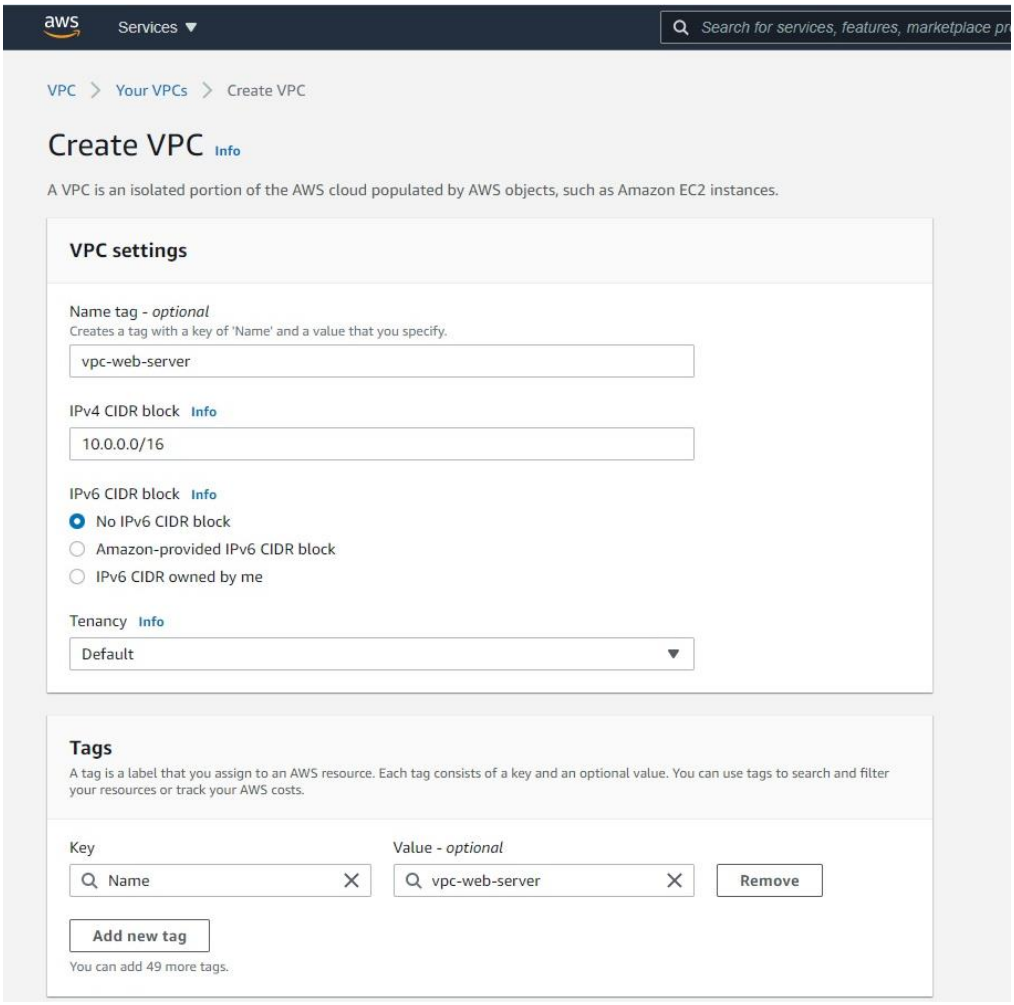
A) Implementing the Server
B) Implementing Apache Web Server
C) Scripts
D) Testing & Results

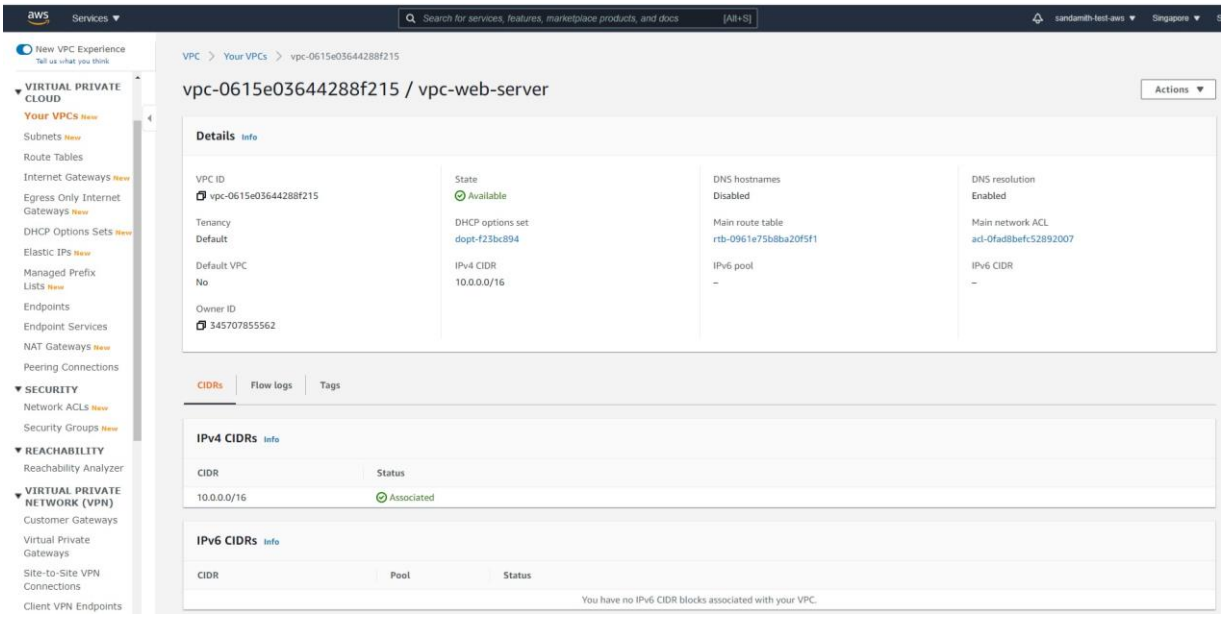
AWS services are used for the tasks of this assignment.

Section A: Implementing the Server

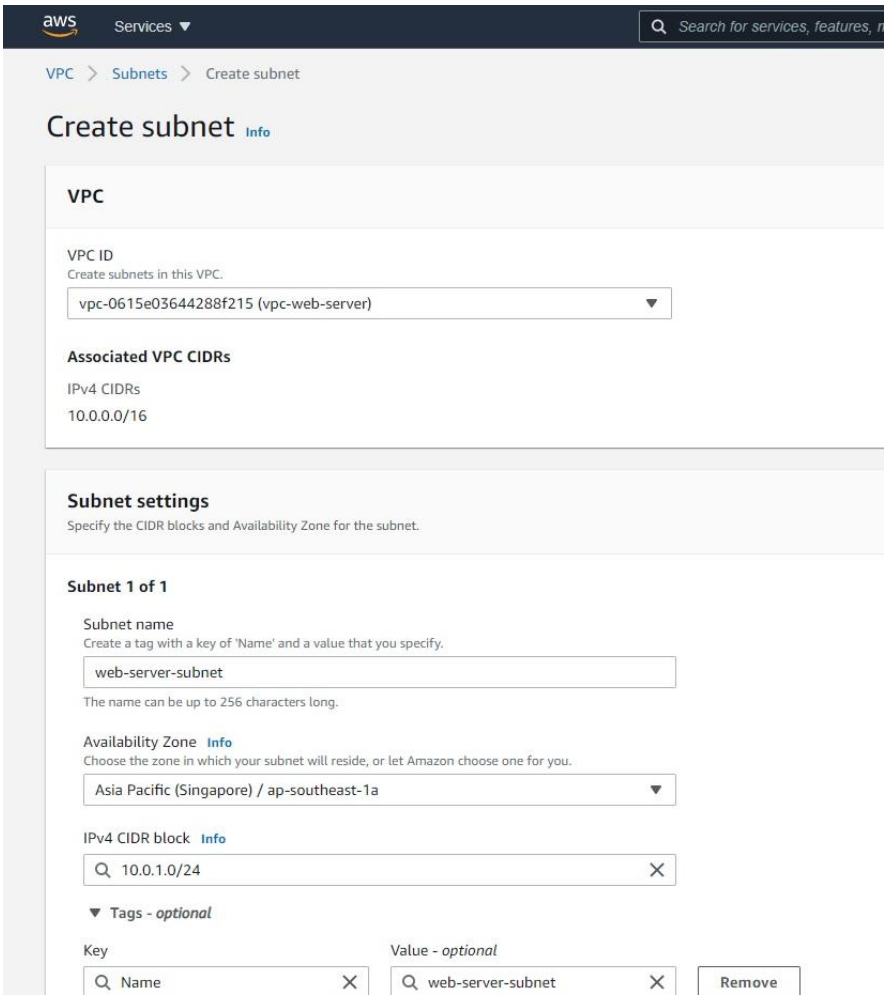


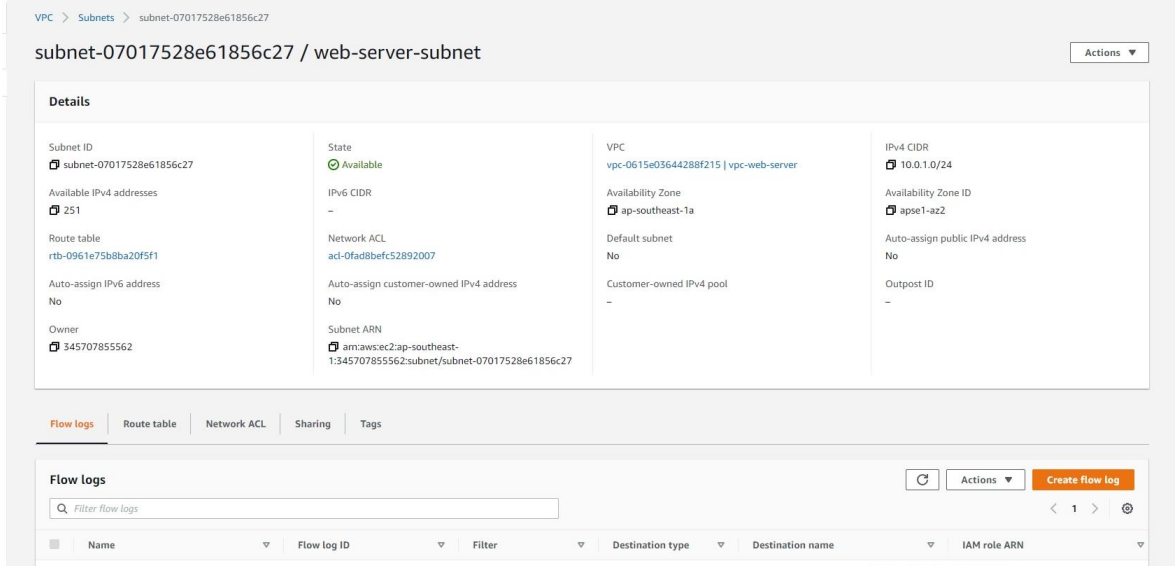
- First, a VPC was created in my AWS console to run the web server.


aws
Services
Search for services, features, marketplace pr
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
Name
Value - optional
vpc-web-server
Remove
Add new tag
You can add 49 more tags.

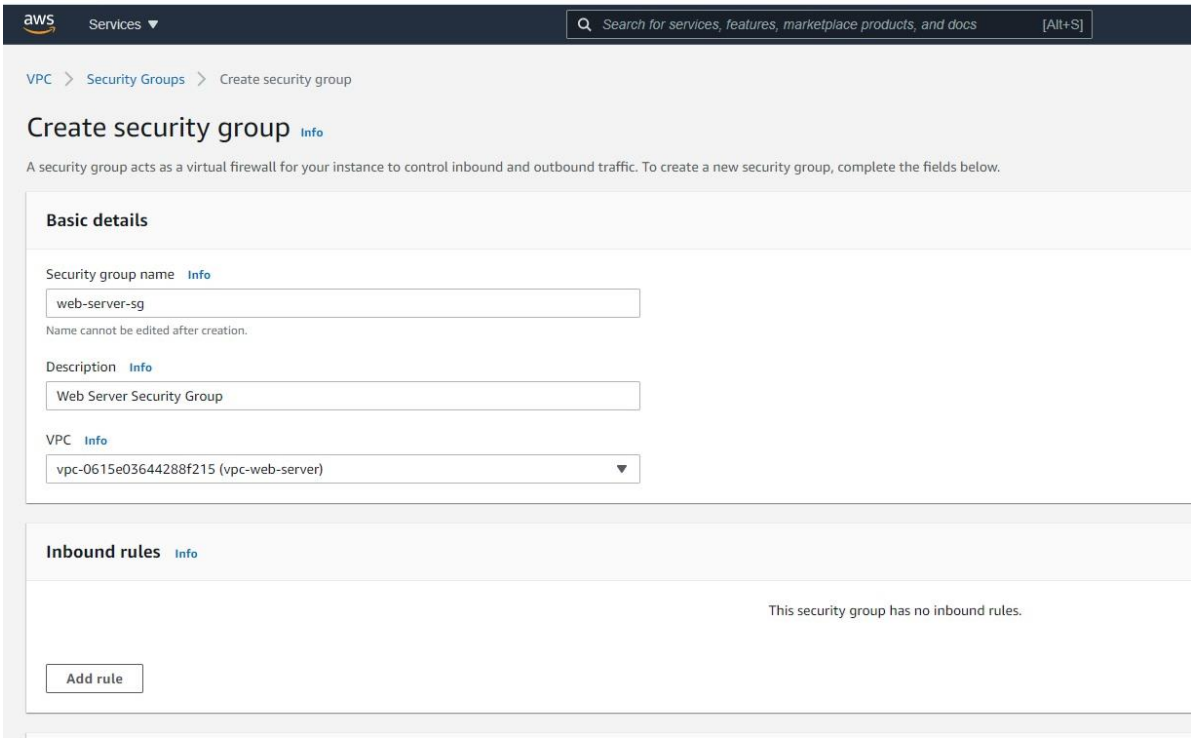

aws
Services
Search for services, features, marketplace products, and docs [Alt+S]
sandamith-test-aws
Singapore
New VPC Experience
Tell us what you think
VIRTUAL PRIVATE CLOUD
Your VPCs New
Subnets New
Route Tables
Internet Gateways New
Egress Only Internet Gateways New
DHCP Options Sets New
Elastic 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
vpc-0615e03644288f215
State
Available
DNS hostnames
Disabled
DNS resolution
Enabled
Tenancy
Default
DHCP options set
dopt-f23bc894
Main route table
rtb-0961e75b8ba20f5f1
Main network ACL
acl-0fad8befc52892007
Default VPC
No
IPv4 CIDR
10.0.0.0/16
IPv6 pool
-
IPv6 CIDR
-
Owner ID
345707855562
CIDRs
Flow logs
Tags
IPv4 CIDRs Info
CIDR
Status
10.0.0.0/16
Associated
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.

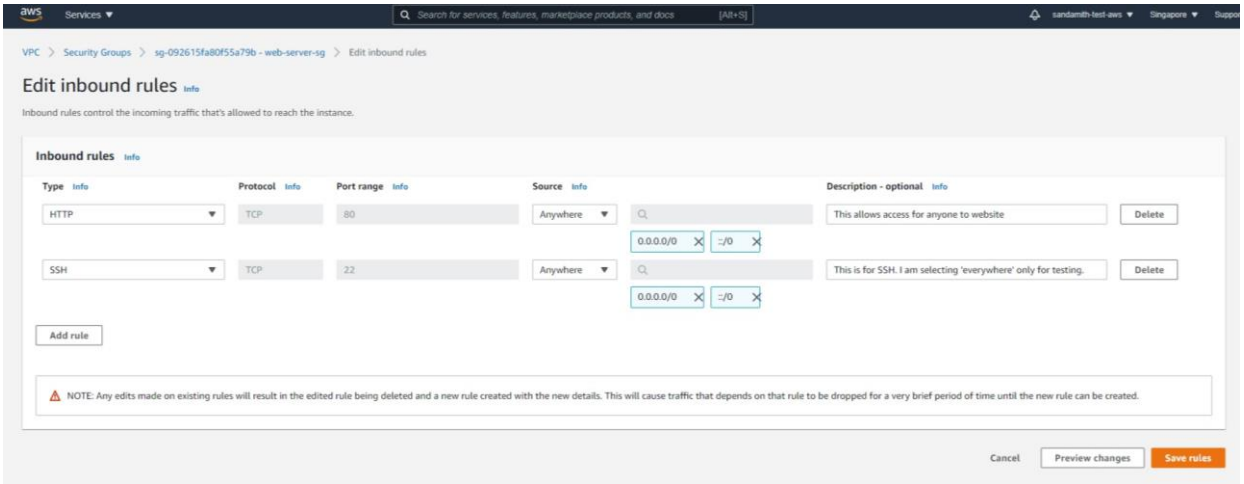




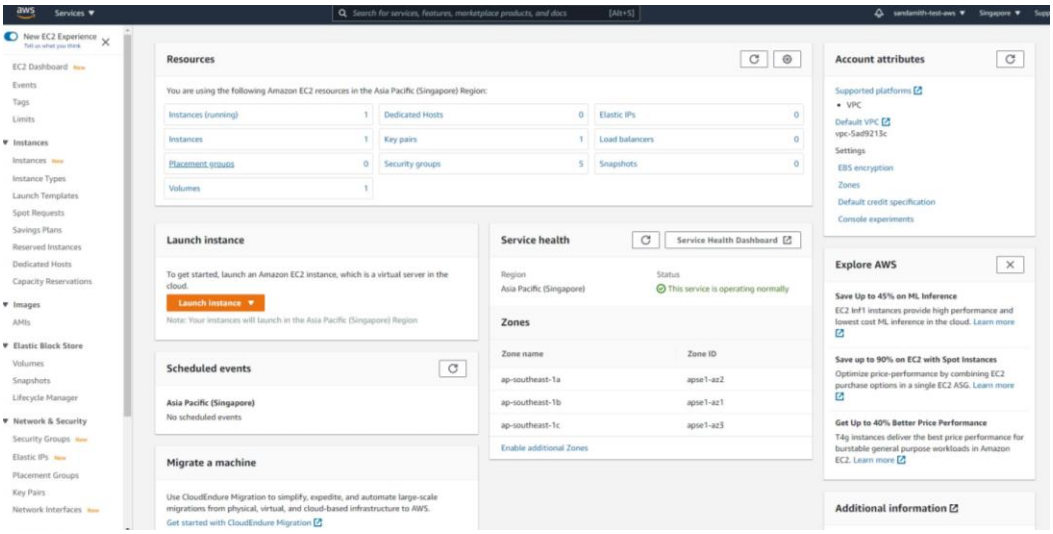
- A security group was created for my VPC to filter incoming traffic.



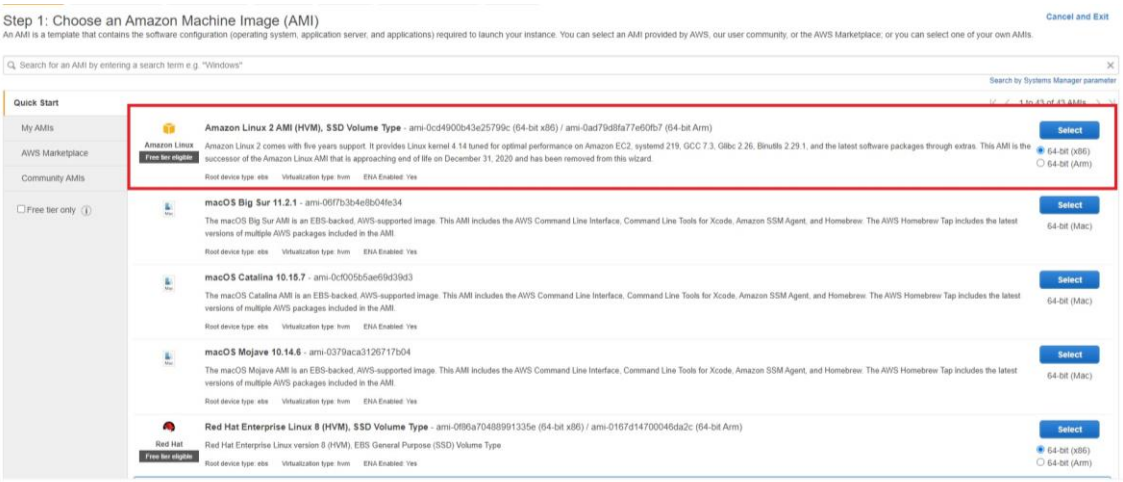
- Inbound rules were added.
- HTTP from anywhere and SSH from anywhere.
- Even though it’s not a good practice to allowing SSH from any IP, I configured it like this for only testing environments.

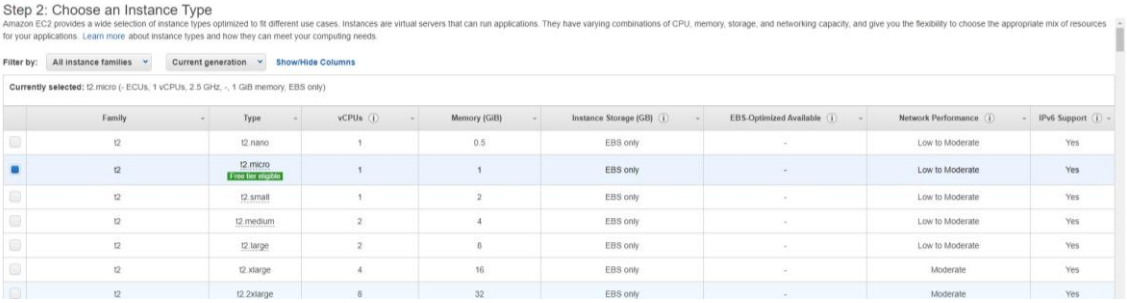


- An EC2 instance was created to launch the web server within it.

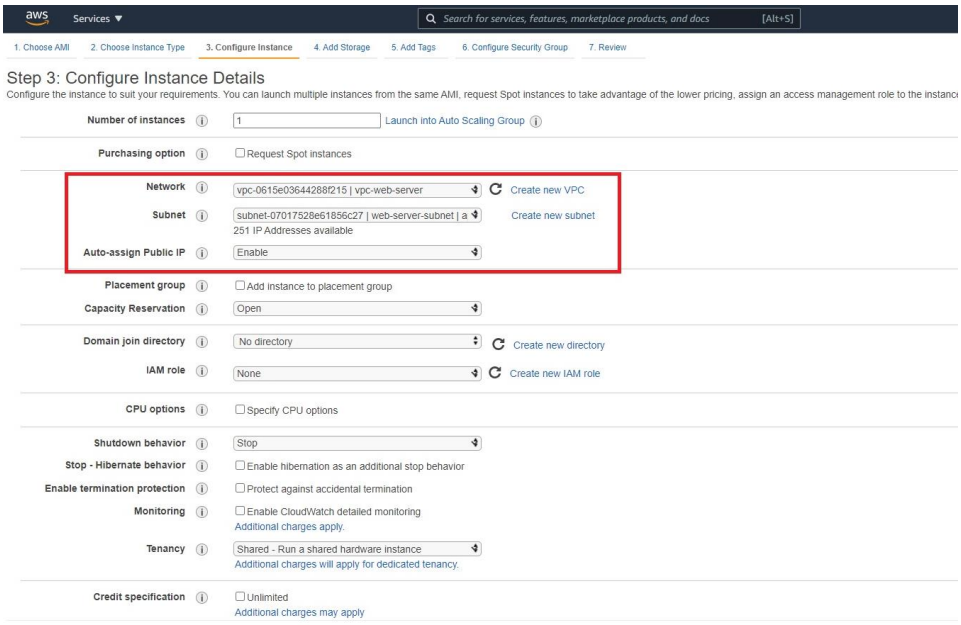


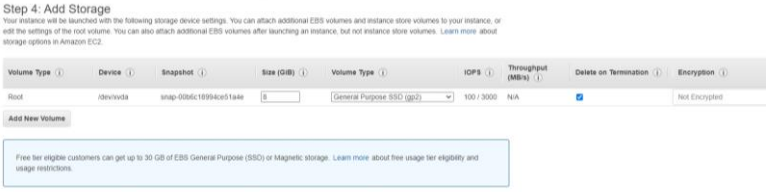
- I chose Amazon Linux 2 as the AMI for my EC2 instance.





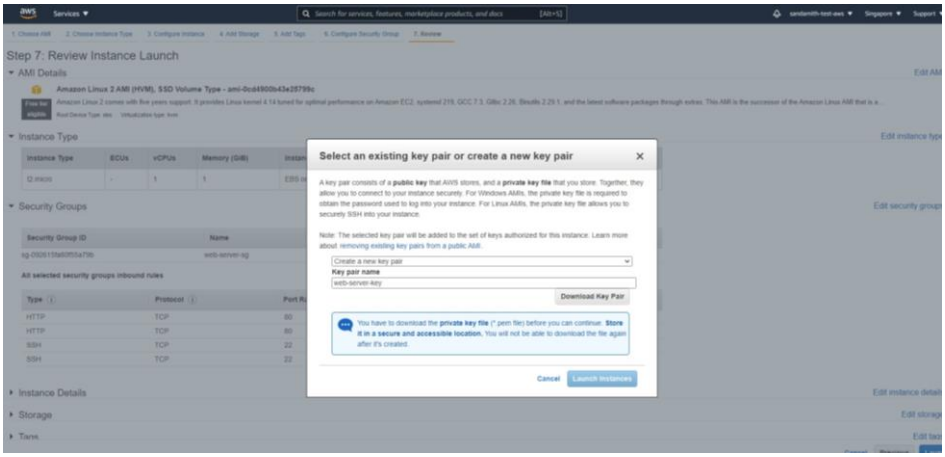
- Previously created VPC and subnet were assigned to the instance. Public IP auto-assigning was also enabled.

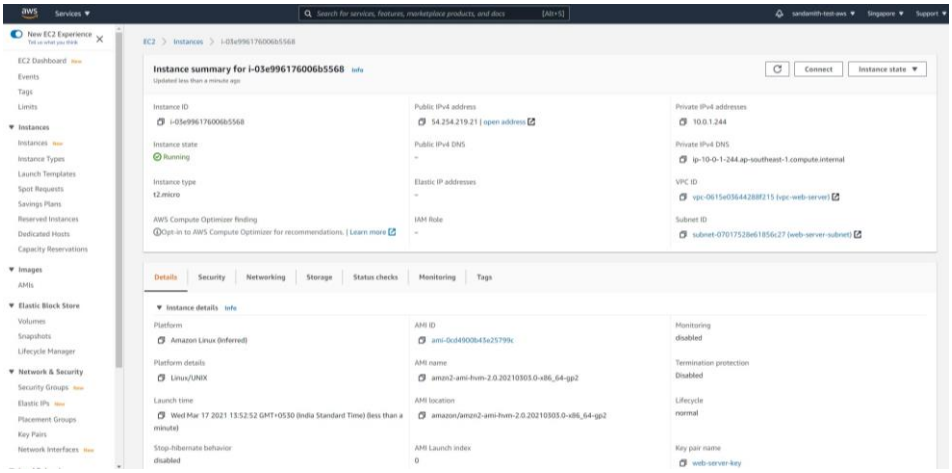




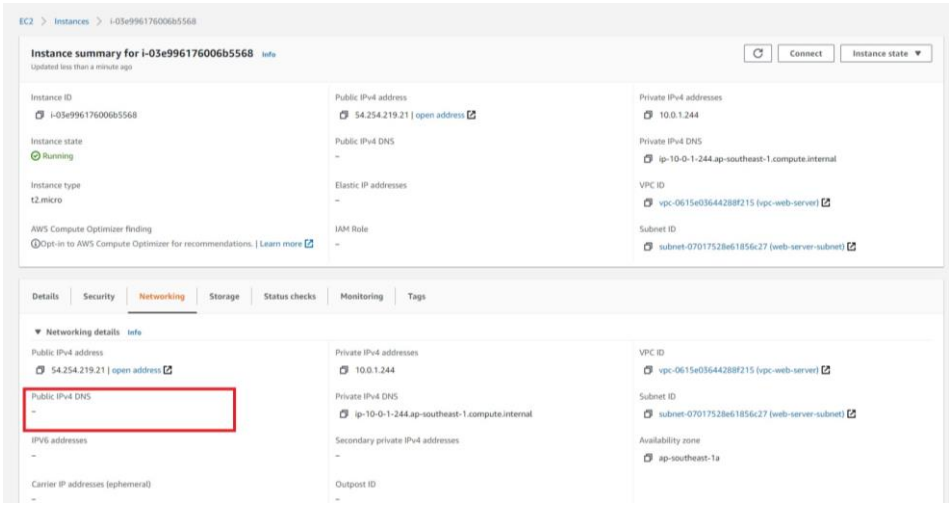
- Security group I created earlier was added to the instance.

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

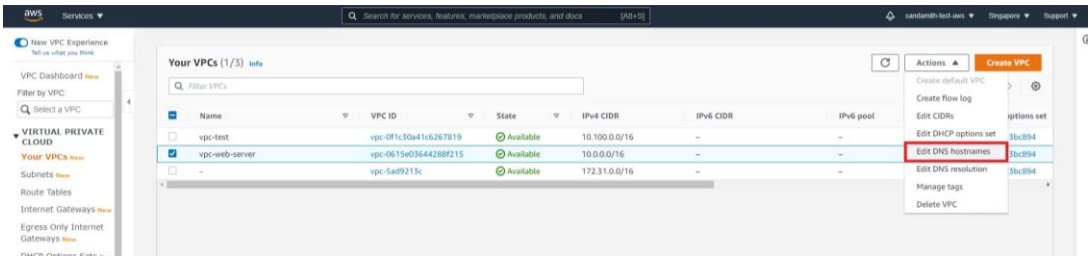


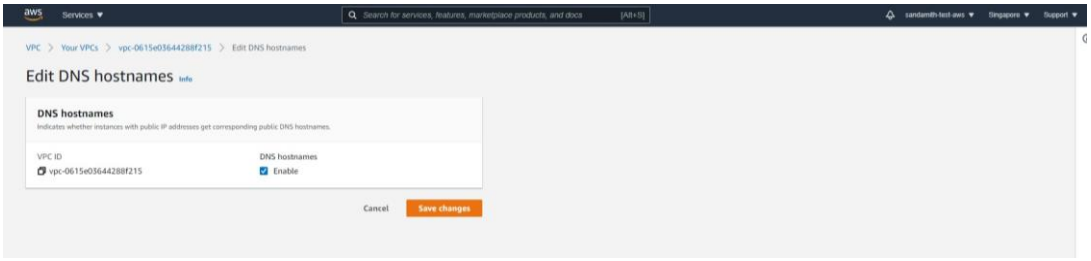


- When we look at our public DNS for our instance, it's not available.

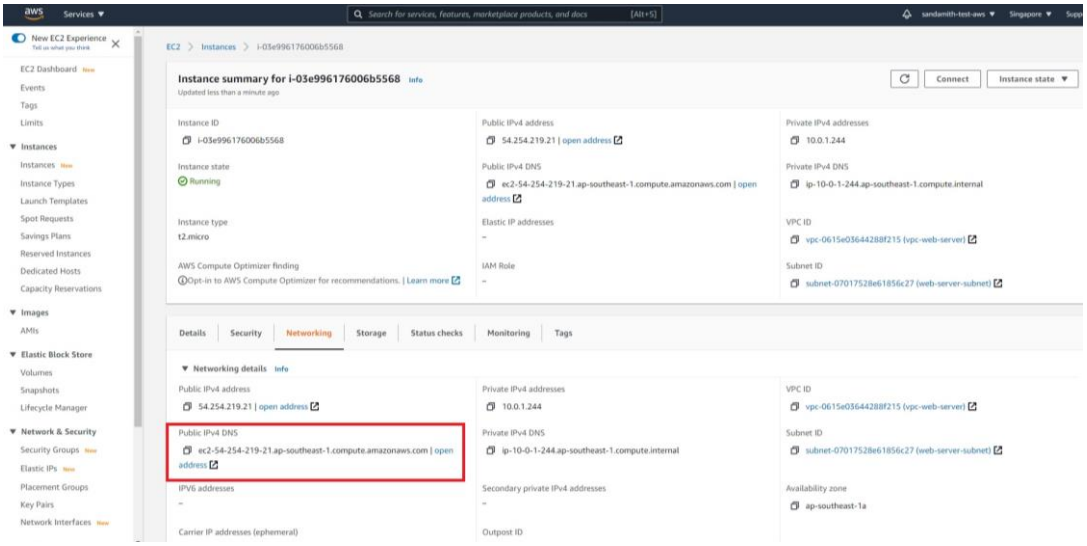


- To make this available, DNS hostnames were enabled in VPS settings.

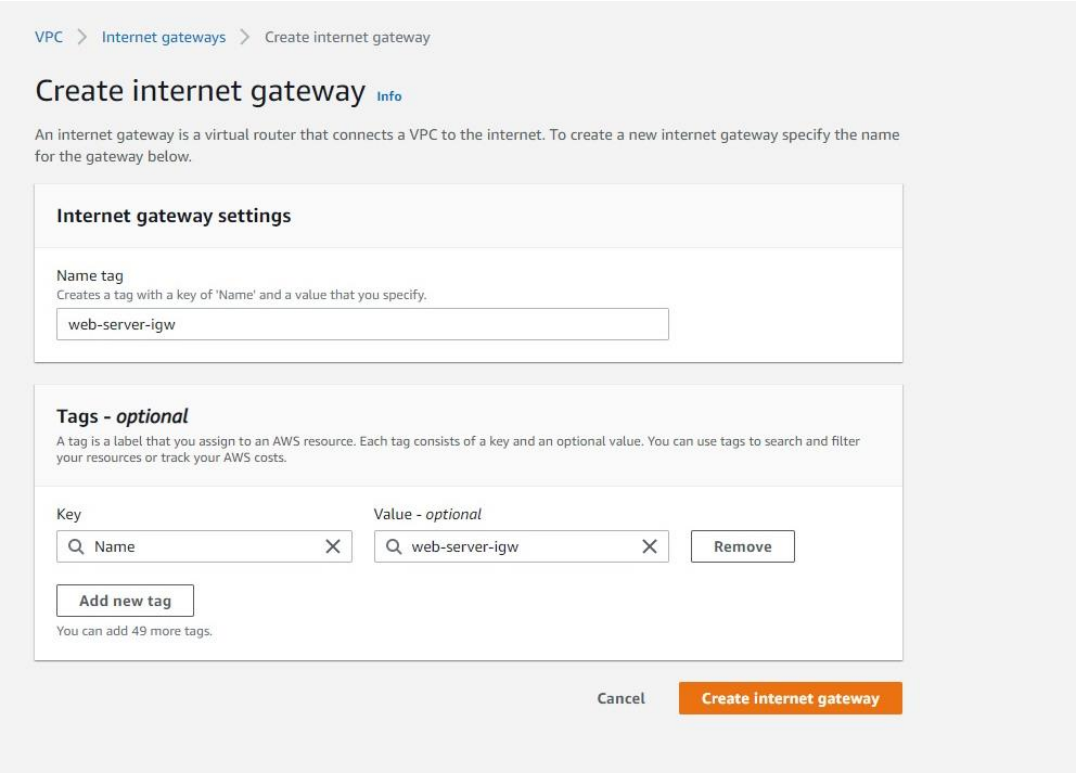


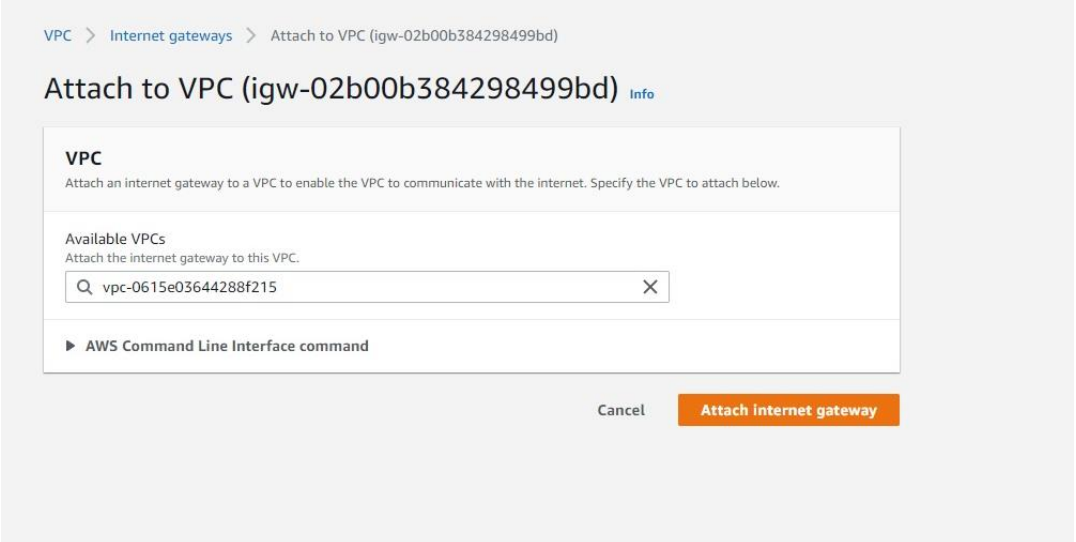


- Public DNS is available now.

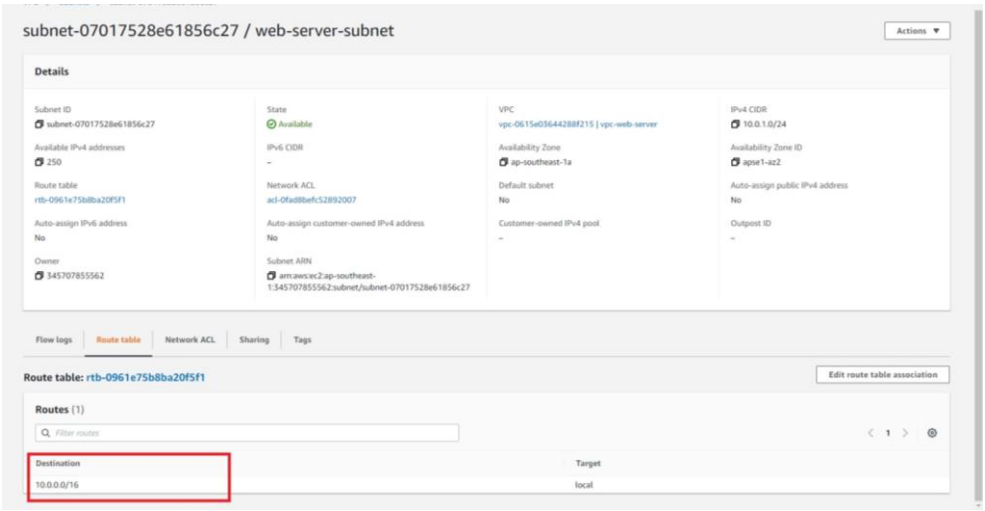


- Our VPC has no connectivity to internet. Therefore, an internet gateway was created and our VPC was attached to it.

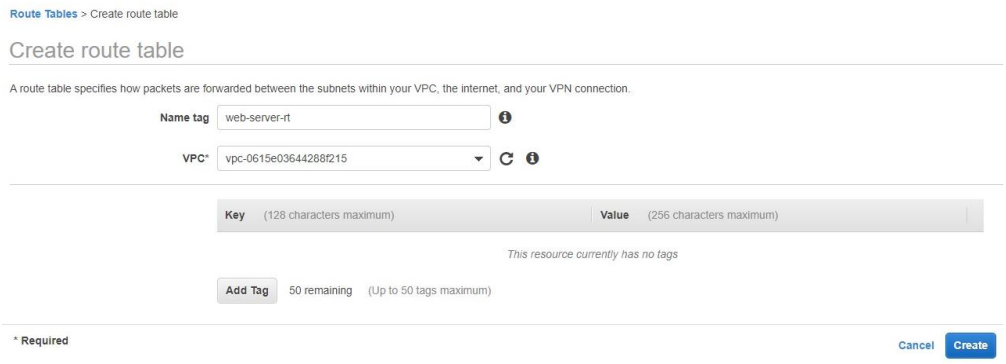




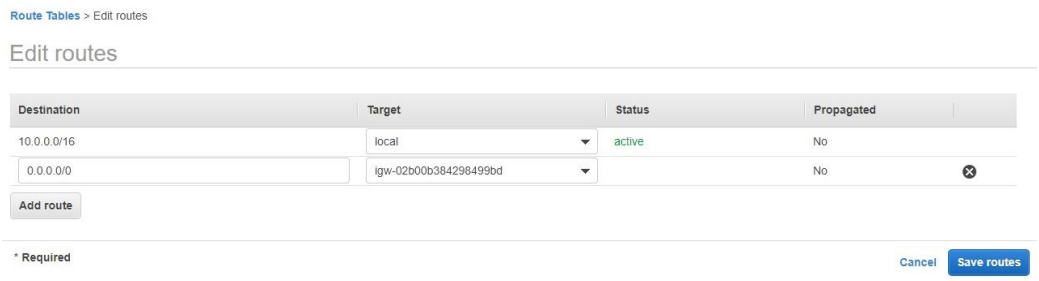
- When we look at our subnet route tables, there are no routes to connect with internet gateway.



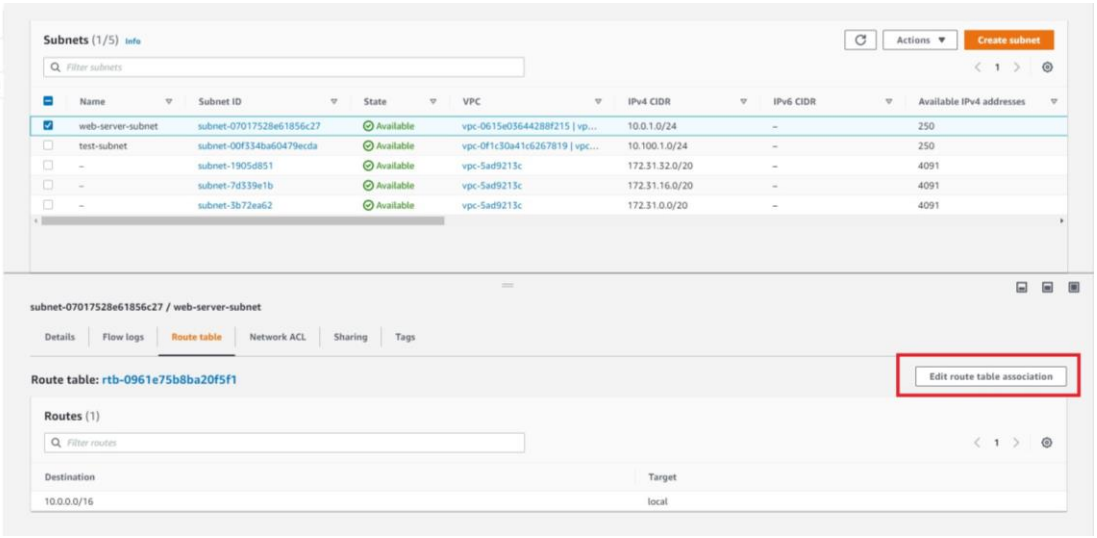
- So, a route table was created for our VPC to allocate routes.

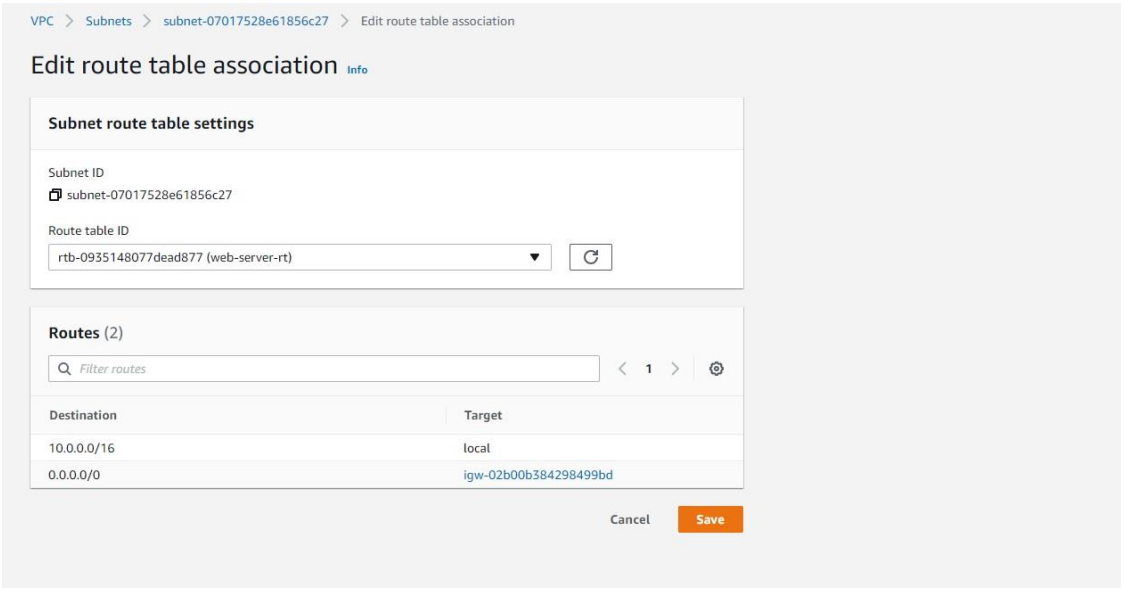


- And a route was added for the internet gateway.



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



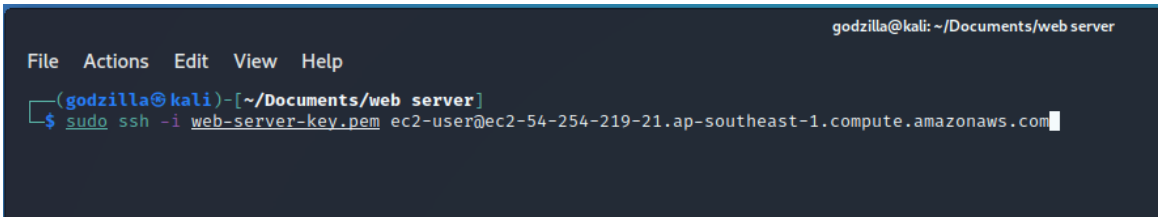


✓ Now everything within AWS console has been configured. Our server is up and running.

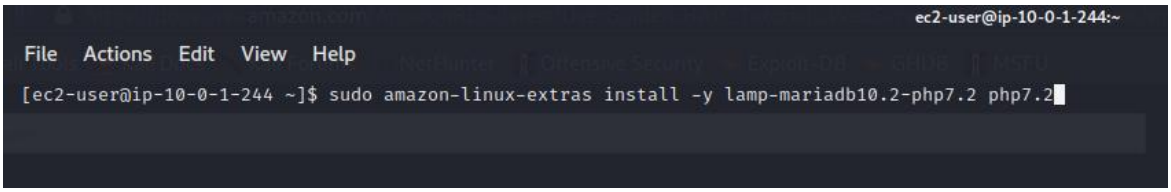
Section B: Implementing Apache Web Server

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

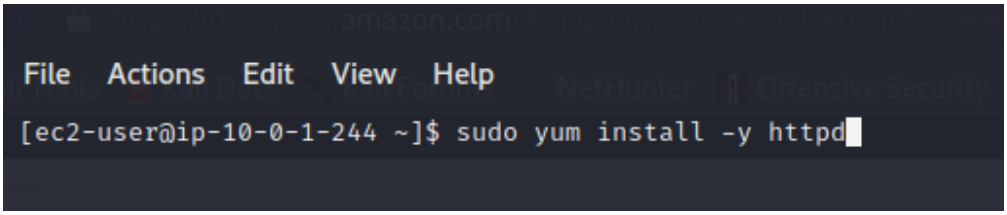
- First, I logged in to the server I created inside AWS console via SSH using the downloaded key-pair and public DNS.



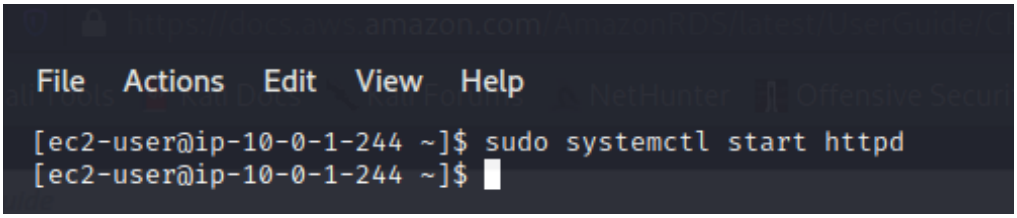
- PHP software was installed.



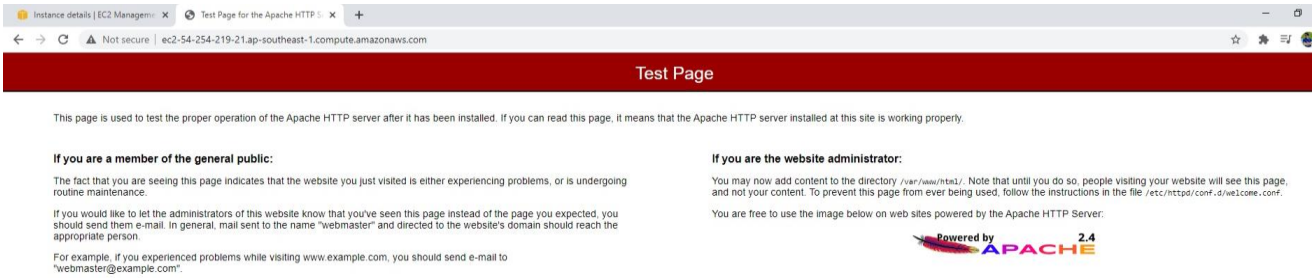
- Apache web server was installed on server after that.



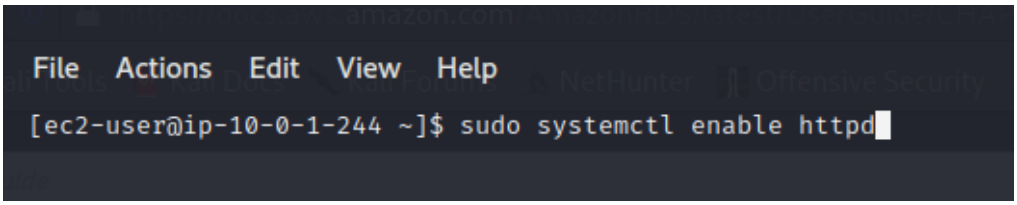
- Then started the apache service.



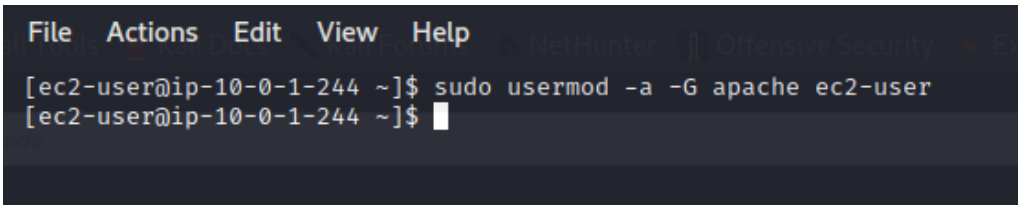
- Tested if the apache is properly installed and working.



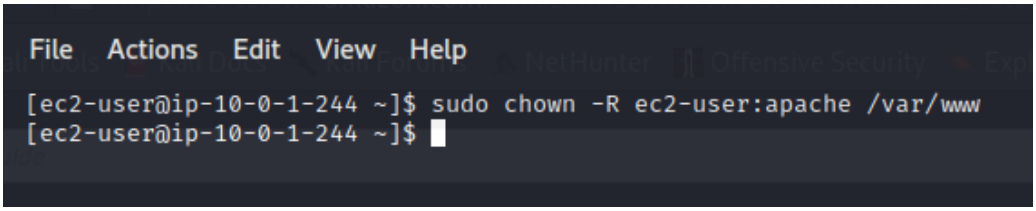
- Configured the web server to start with each system boot.



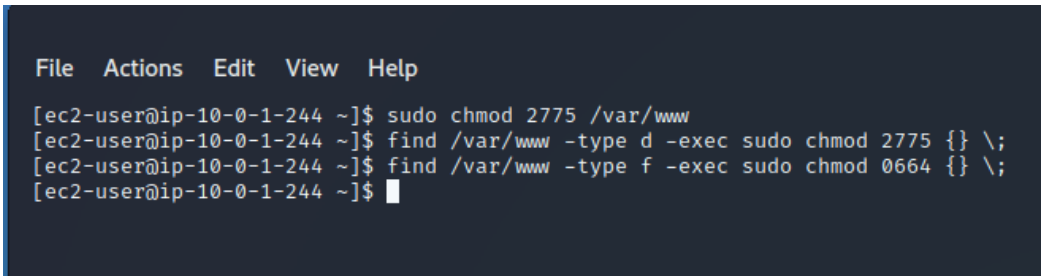
- Added ec2-user to apache group.



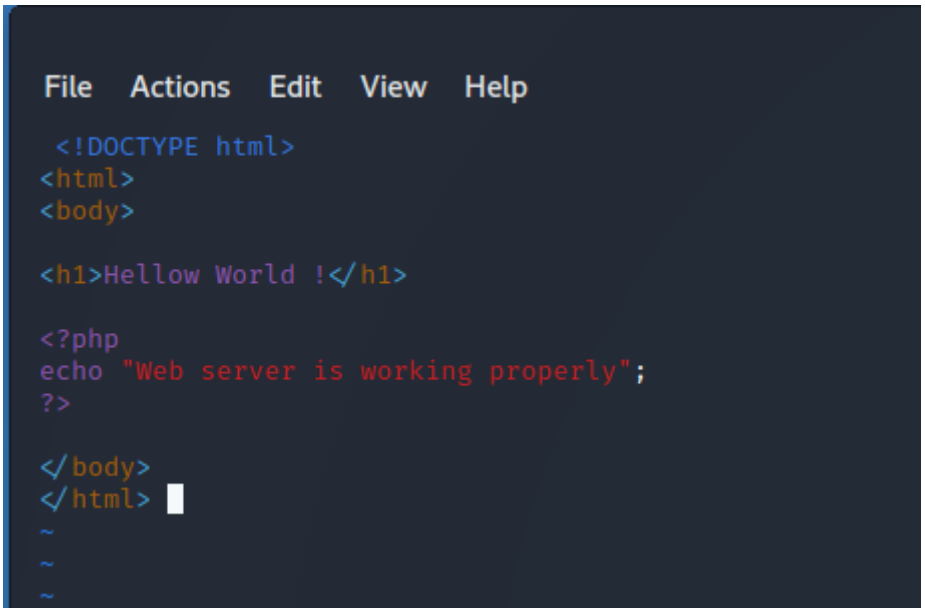
- Changed group ownership of /var/www directory and its contents to apache group.



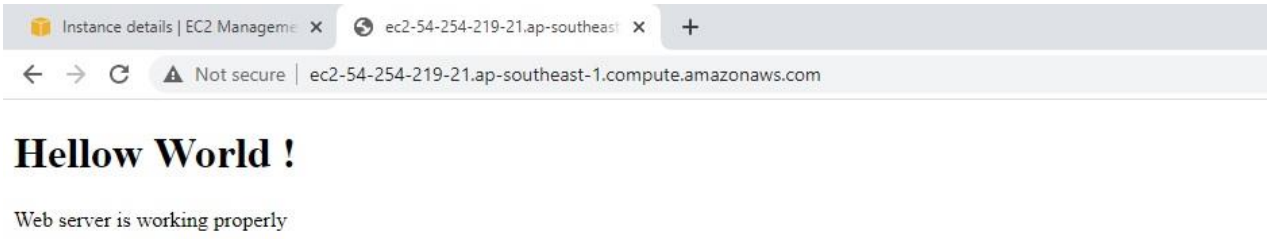
- Changed the directory and file permissions of /var/www and its subdirectories recursively to give permissions to future members of apache group.



- 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.



- Finally, I checked whether it’s working.



☺ Apache web server is running and serving content properly.

Section C: Scripts

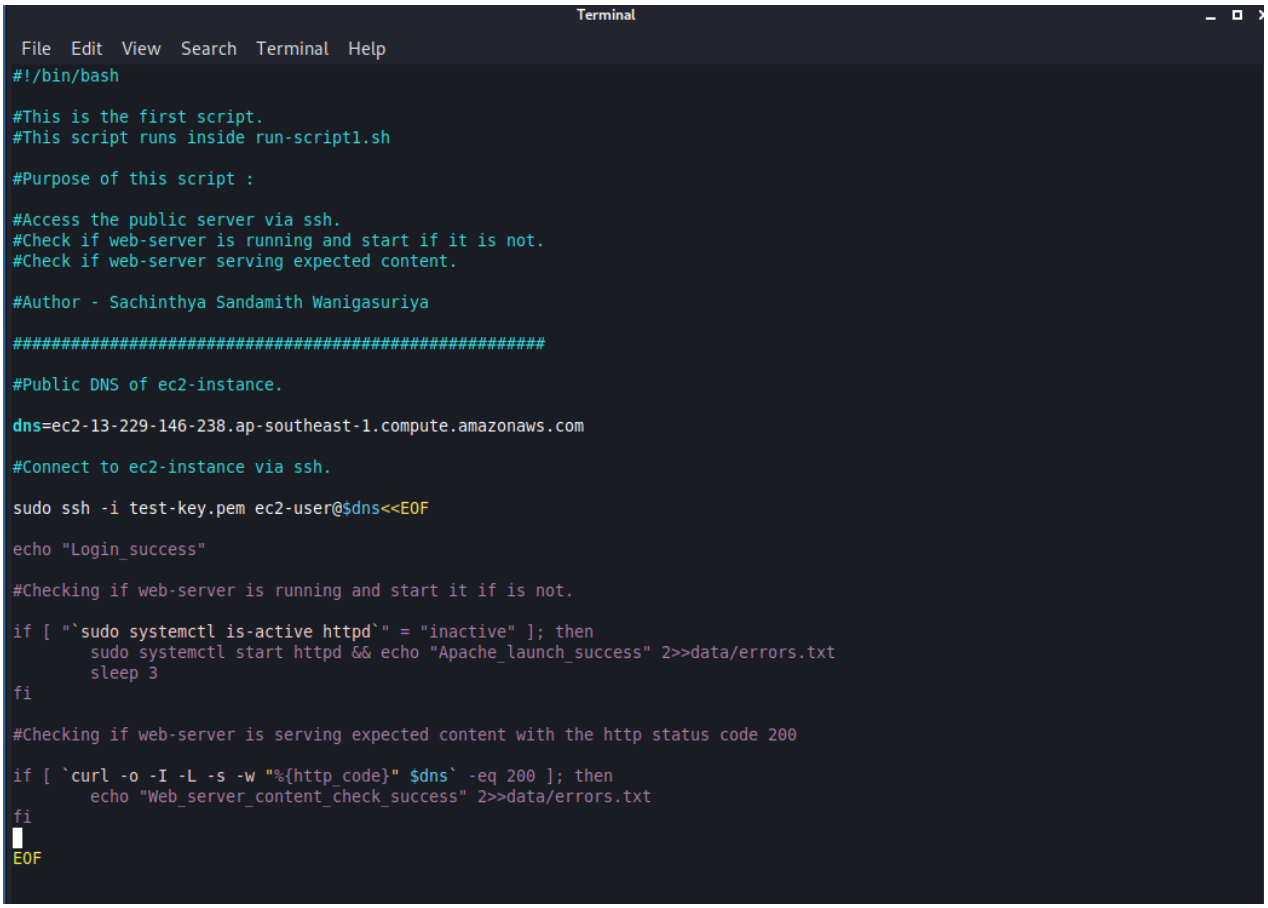
I created 2 main scripts to carry out the tasks of this assignment. There are 2 other scripts running inside these main scripts.

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

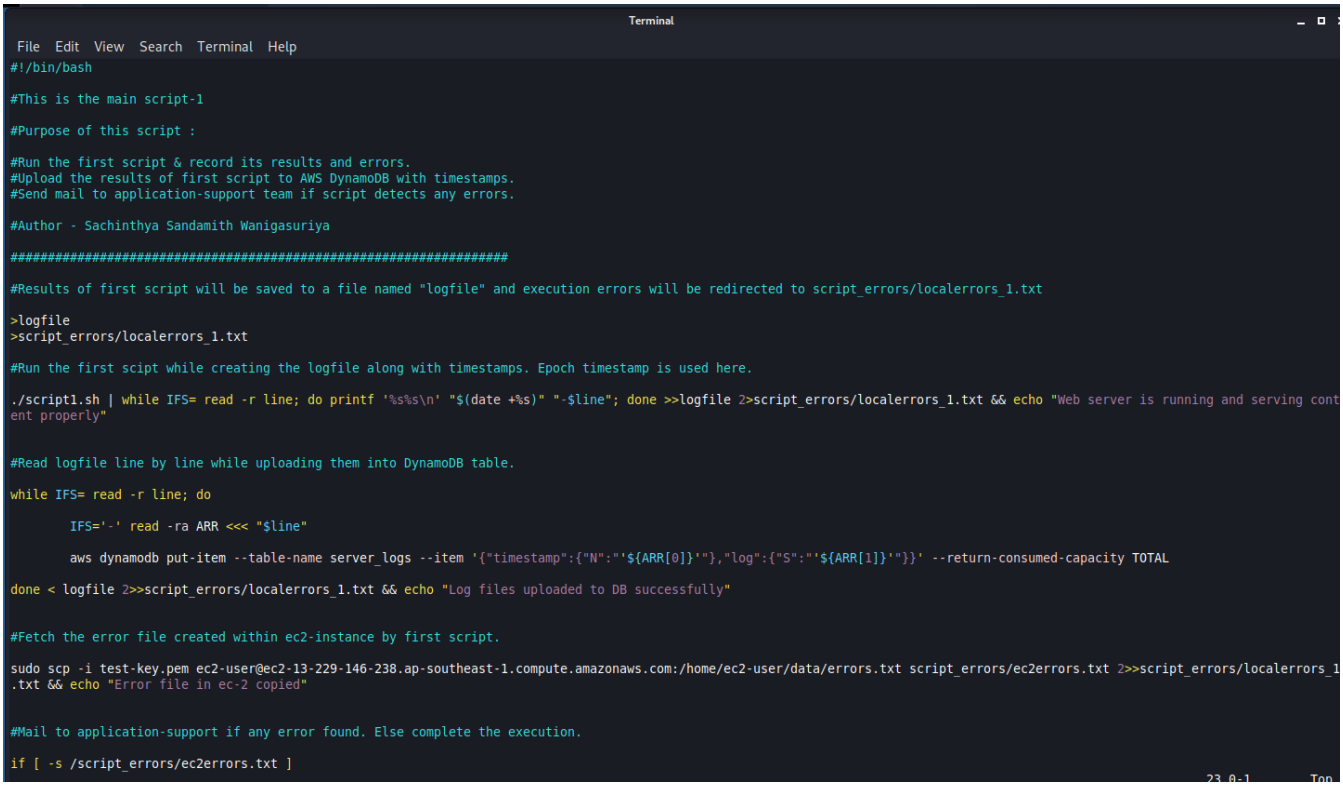
1) First Script (script1.sh)

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



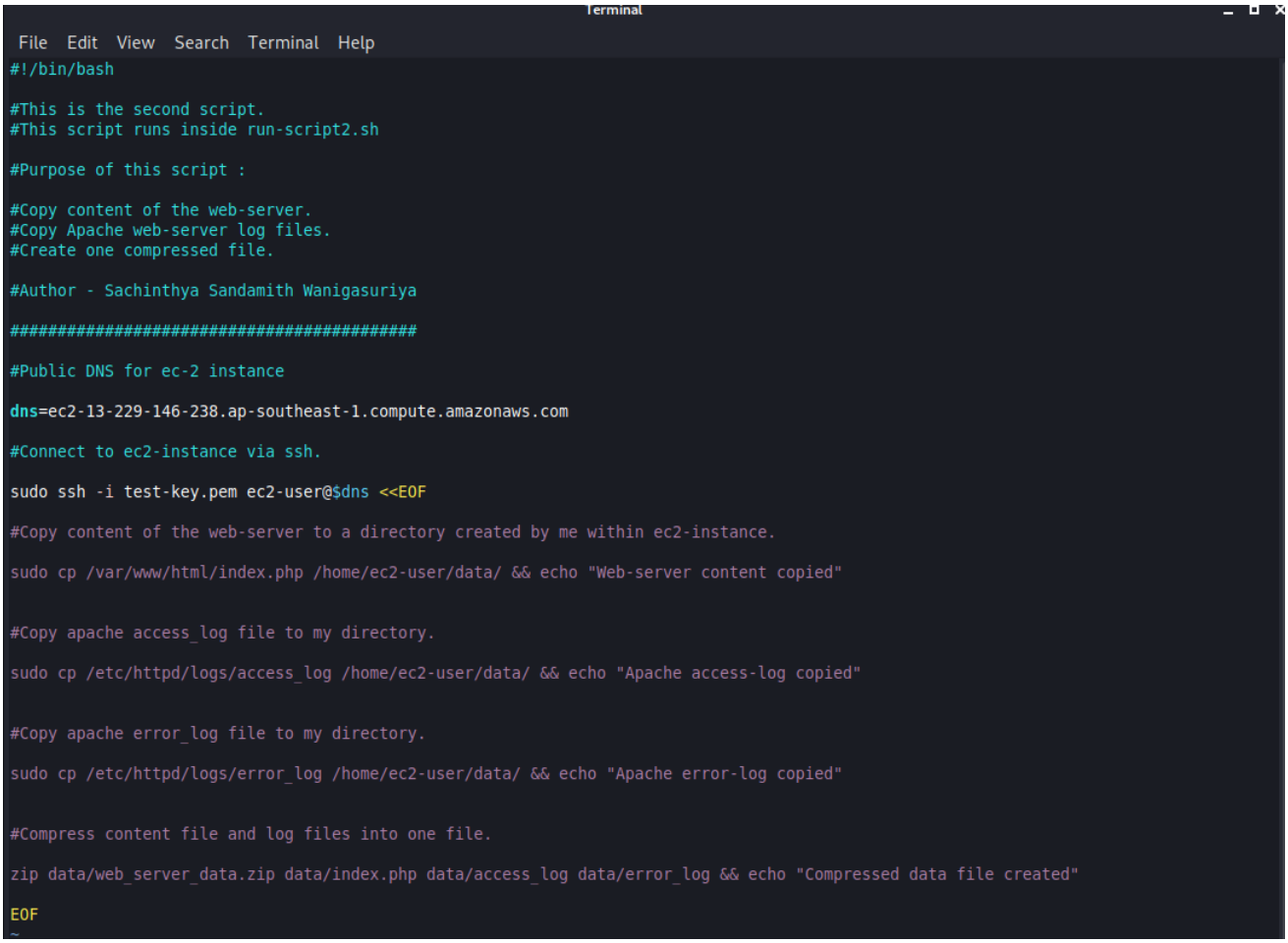
2) Main Script – 1 (run-script1.sh)

- 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.



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.



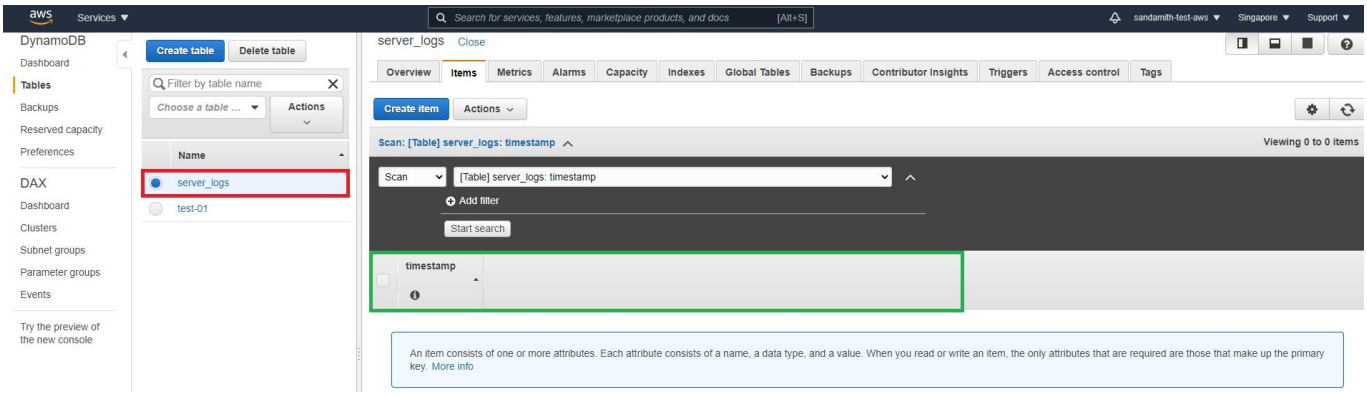
4) Main Script – 2 (run-script2.sh)

- 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 occured.

#Author - Sachinthya 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 downloade
d 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
```

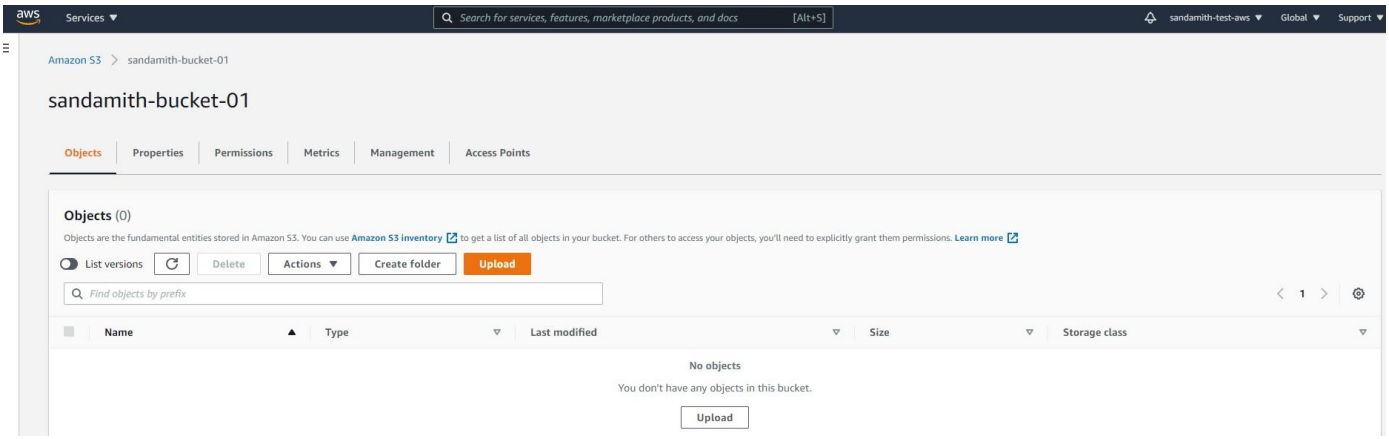
AWS DynamoDB Table (server_logs)

- This was created to save script results.



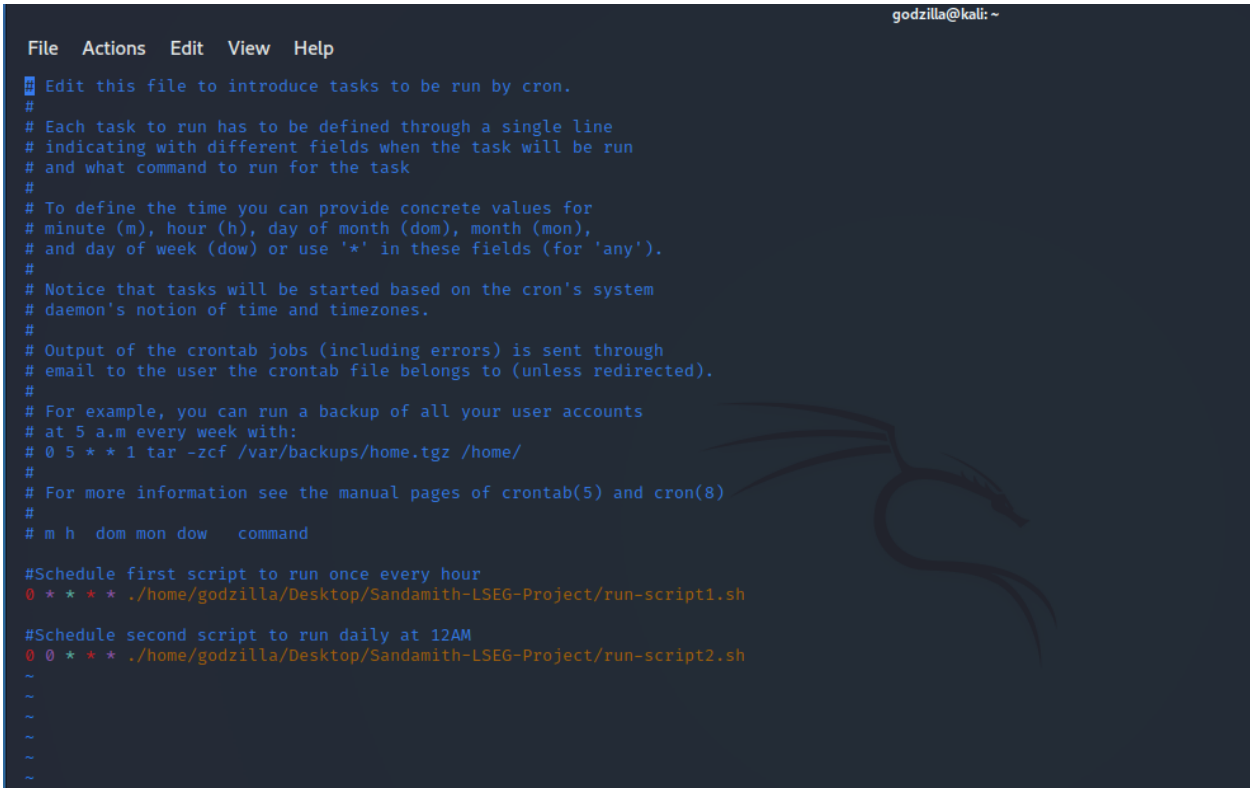
AWS S3 Bucket (sandamith-bucket-01)

- This was created to upload server data file.



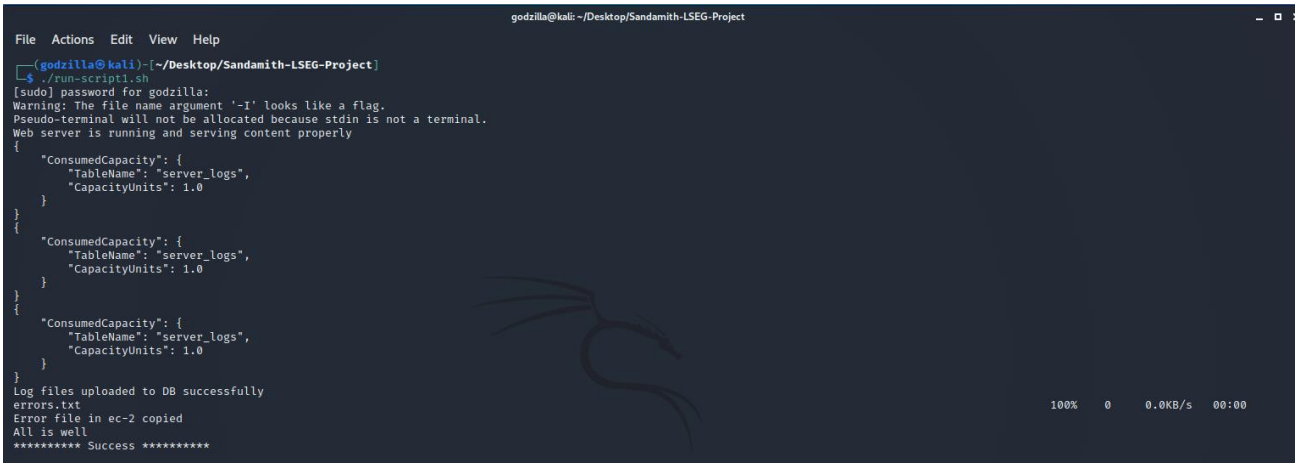
Scheduling scripts to run periodically

- 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.

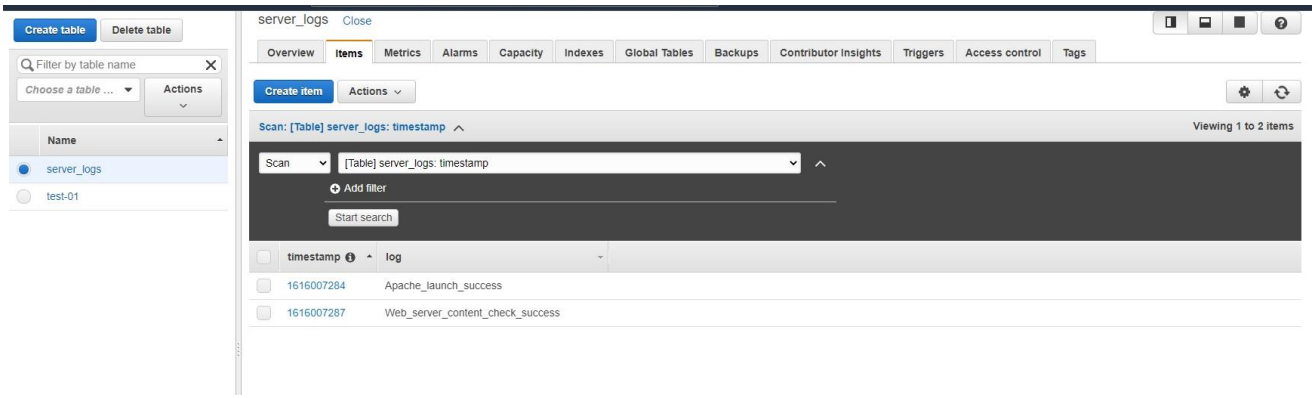


Section D: Testing & Results

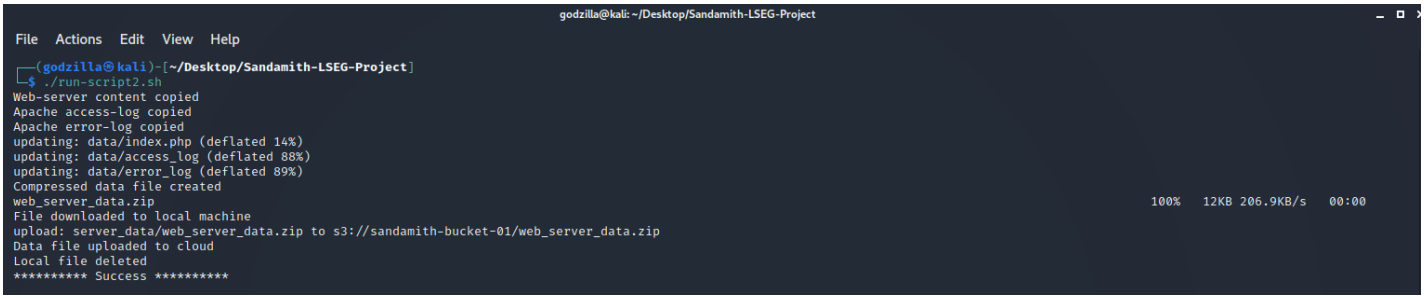
- Execution and results of Main Script –1 (run-script1.sh)



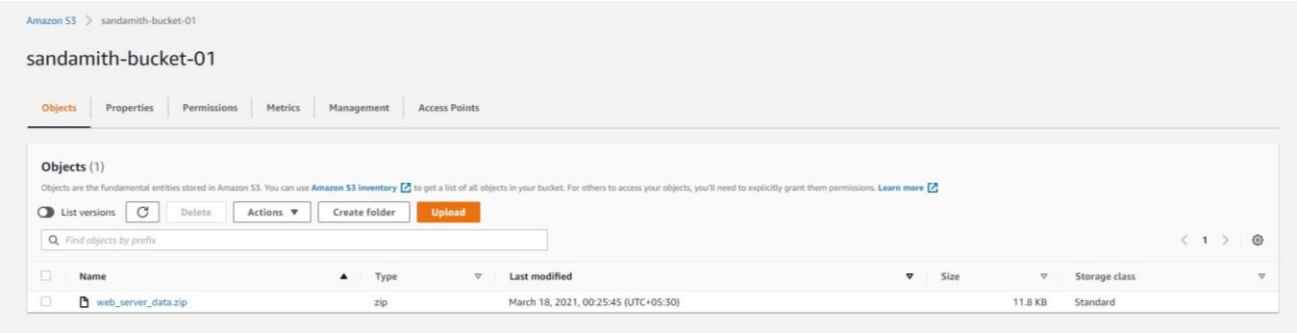
- Script results are successfully saved in DynamoDB table.



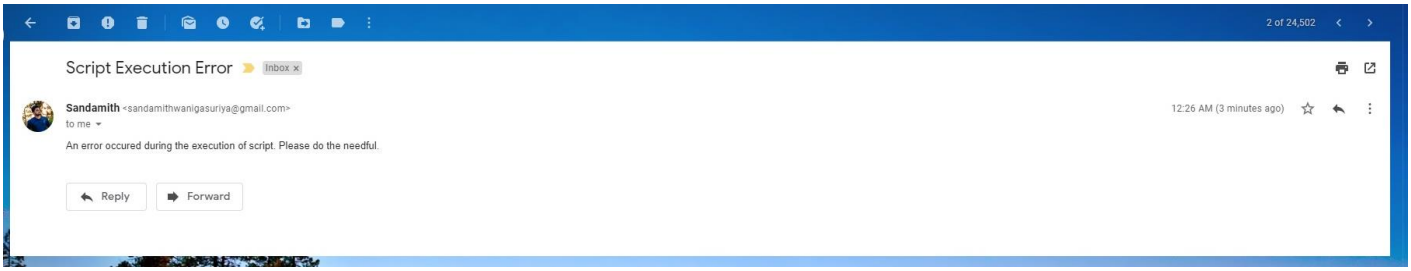
- Execution and results of Main Script –2 (run-script2.sh)



- 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 ***