



London
Stock Exchange Group

Application Support Engineer
- Assignment -

Janith Gedarawatta

TABLE OF CONTENT

KEY POINTS	5
CONSIDERED SECURITY POINTS	5
LOGGING MECHANISMS USED	5
ENTERPRISE ARCHITECTURE DIAGRAM	6
Q1 - CREATING A NEW SERVER WITH PUBLIC ACCESS.....	7
CHECKING IF THE NEWLY CREATED EC2 INSTANCE CAN BE ACCESSED PUBLICLY (SSH USING PUTTY)	11
Q2 - INSTALLING APACHE SERVER ON EC2 INSTANCE	14
Q3 (I) - SSH TO EC2 INSTANCE AND CHECK FOR WEB SERVER STATUS	16
Q3 (II) – CHECK IF THE WEB SERVER IS SERVING THE CONTENT AND RETURN 200 CODE	19
Q3 (III) – SAVE THE SCRIPT RESULT IN A DATABASE.....	20
Q3 (IV) - SENDING EMAIL TO THE APP SUPPORT TEAM WITH SUITABLE ERROR MESSAGE ..	23
Q4 (I) – COLLECT LOG FILES OF THE WEB SERVER DAILY AND CREATE A COMPRESSED FILE.	24
Q4 (II) – MOVE THE COMPRESSED FILE TO THE LOCATION OF THE SCRIPT	24
Q4 (III) – UPLOAD THE COMPRESSED FILE TO S3 BUCKET	25
Q4 (IV) – DELETE THE COMPRESSED FILE IF SUCCESSFUL AND SEND THE EMAIL IF NOT SUCCESSFUL	27
COMPLETE SCRIPT FOR QUESTION 03	28
COMPLETE SCRIPT FOR QUESTION 04	29
SCRIPT FOR UPDATING SERVER AND INSTALLING HTTP SERVER	29
STEPS TO RUN THE SCRIPTS.....	30

AUTOMATING THE SETUP	31
KEY POINTS OF HIGH AVAILABILITY AND RELIABILITY OF THE PROPOSED ENTERPRISE ARCHITECTURE.....	32
TECHNOLOGIES USED	32

Table of Figures

Figure 1.....	6
Figure 2.....	7
Figure 3.....	7
Figure 4.....	8
Figure 5.....	8
Figure 6.....	9
Figure 7.....	9
Figure 8.....	10
Figure 9.....	10
Figure 10.....	11
Figure 11.....	12
Figure 12.....	13
Figure 13.....	13
Figure 14.....	14
Figure 15.....	14
Figure 16.....	15
Figure 17.....	16
Figure 18.....	17
Figure 19.....	18
Figure 20.....	18
Figure 21.....	18
Figure 22.....	19
Figure 23.....	19
Figure 24.....	20
Figure 25.....	20
Figure 26.....	21
Figure 27.....	21
Figure 28.....	21
Figure 29.....	22
Figure 30.....	22
Figure 31.....	23
Figure 32.....	23
Figure 33.....	24
Figure 34.....	24
Figure 35.....	25
Figure 36.....	25
Figure 37.....	26
Figure 38.....	26
Figure 39.....	26
Figure 40.....	27
Figure 41.....	27
Figure 42.....	27
Figure 43.....	28
Figure 44.....	28
Figure 45.....	29
Figure 46.....	29

Key points

- AWS EC2 instance is a **t2.micro RHEL** environment
- Status of the commands run in the script are echo'd to the terminal for easy reference of the user.
- AWS CLI has been configured on external machine using Access key ID, Secret Access Key and the Region to give access to IAM user from external machine. (screenshots attached below)
- SMTP server has been configured on the external machine to send emails. (screenshots attached below)
- Bash scripts were created part by part for different questions, later added them altogether to create 2 final scripts for Question 3 and Question 4.

Considered Security Points

- All the procedures are done using an **IAM** user. Root user is not used unless it's necessary
- A security group is configured with only access to 22 and 80 ports which can be accessed externally.
- Changed the access permission of the private key using **# chmod 400 key.pem**
- Private key less login method has been configured between the external machine and EC2 instance.
- Access key ID and Secret Access key are used to configure AWS CLI on external machine.

Logging Mechanisms used

- In script 1 all the log data along with a timestamp are uploaded into DynamoDB.
- For both script 1 and script 2 a log file is maintained inside the directory where script is running. The logs are stored in these text files for easy reference of the user.
- Every step is checked with an if statement and if any error pops while running the script, it is displayed in the terminal and it sends an email to application support team as well.

Enterprise Architecture Diagram

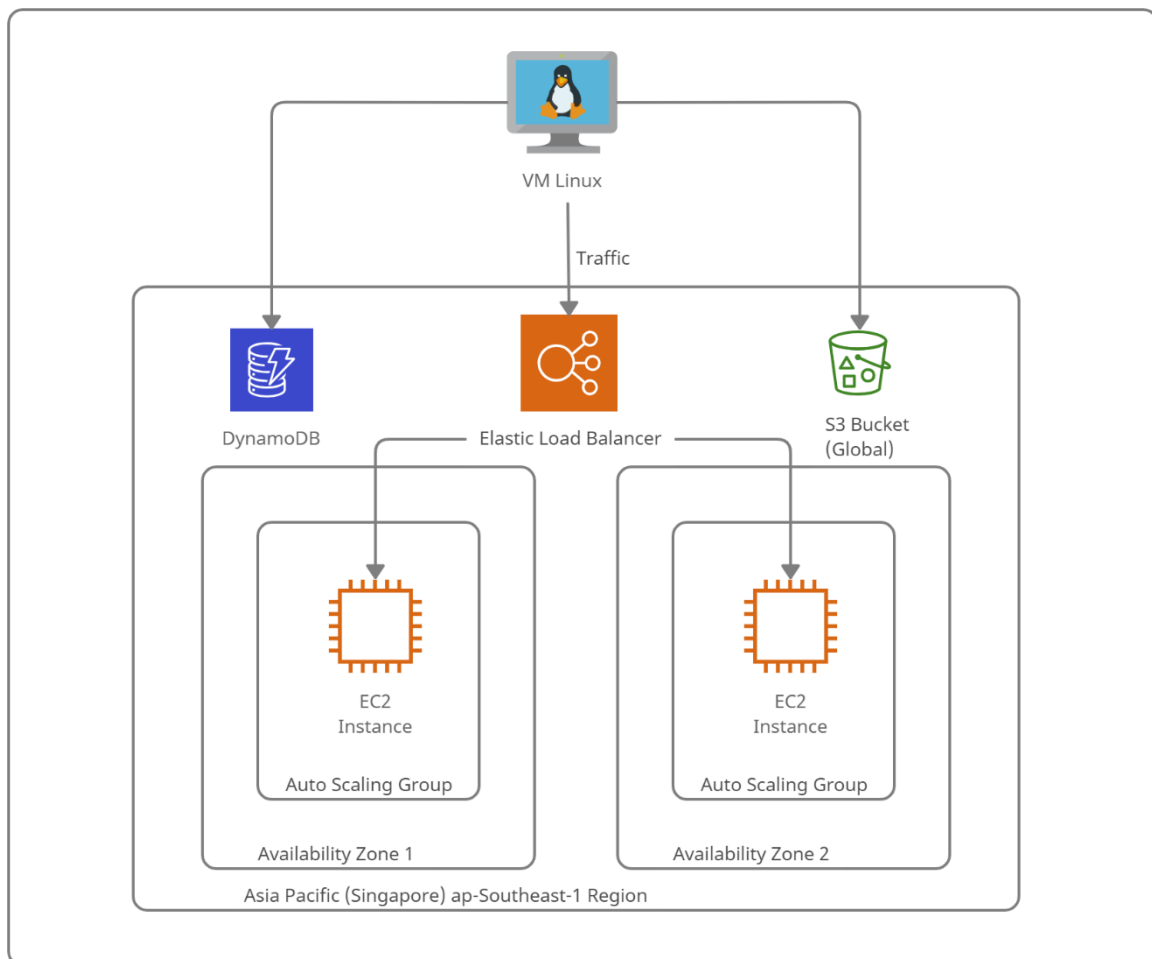


Figure 1

Q1 - Creating a new server with public access

1. Create a new free AWS root account if already not.
2. Using that root account, create an IAM account on AWS
3. Create a new EC2 instance
 - 3.1. Go to services and select EC2

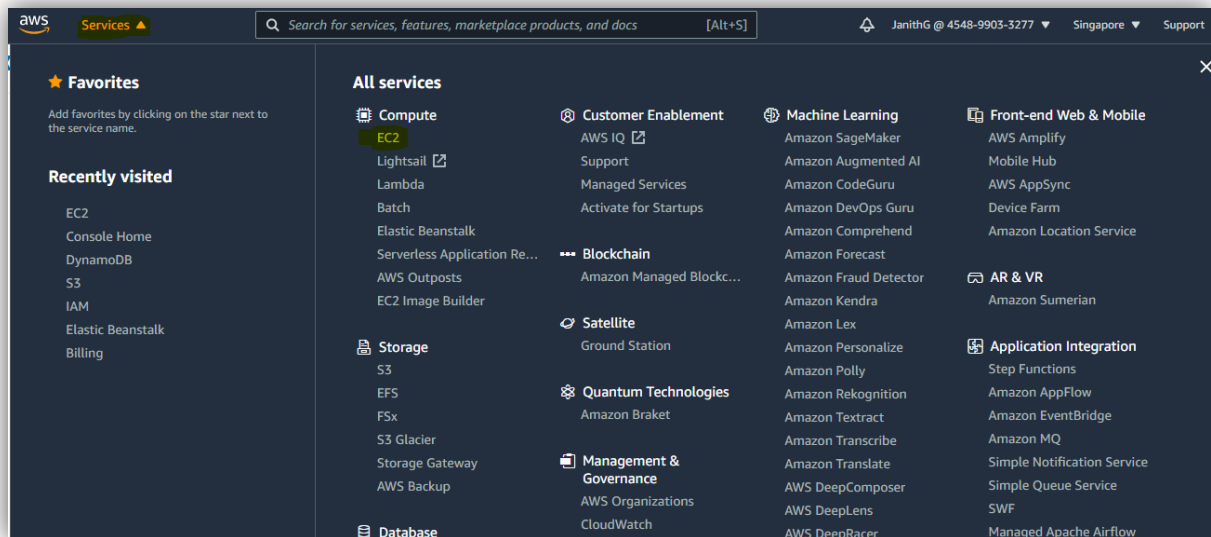


Figure 2

- 3.2. Select “Launch Instance”
- 3.3. Choose the preferred Amazon Machine Image (AMI) to install

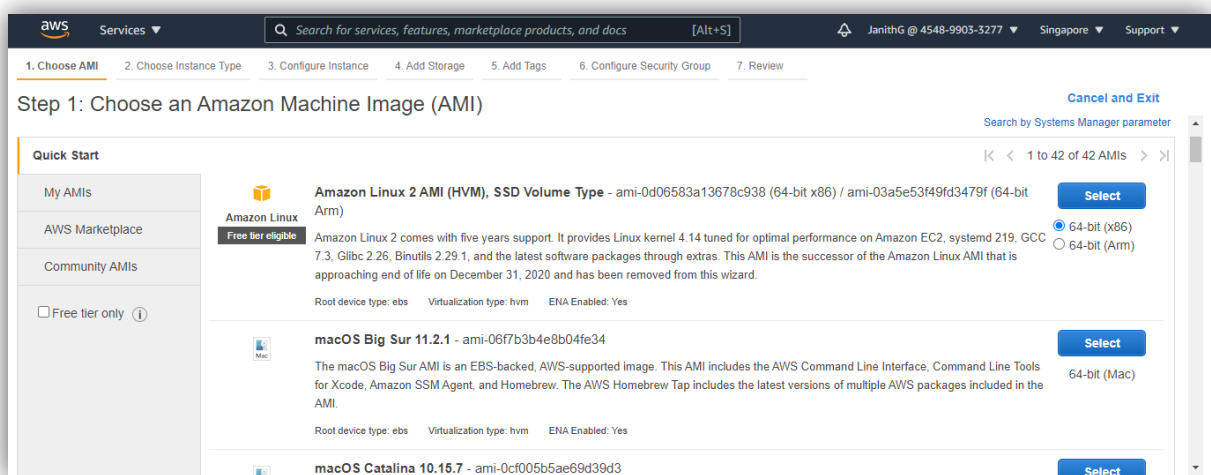


Figure 3

3.4. Choose an Instance Type

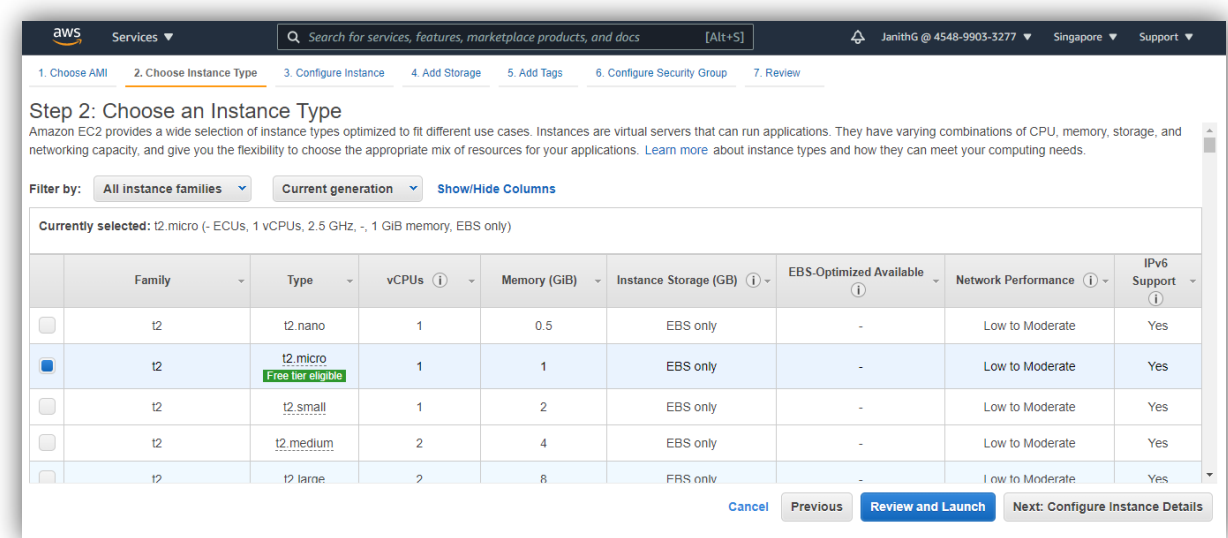


Figure 4

3.5. Configure the instance

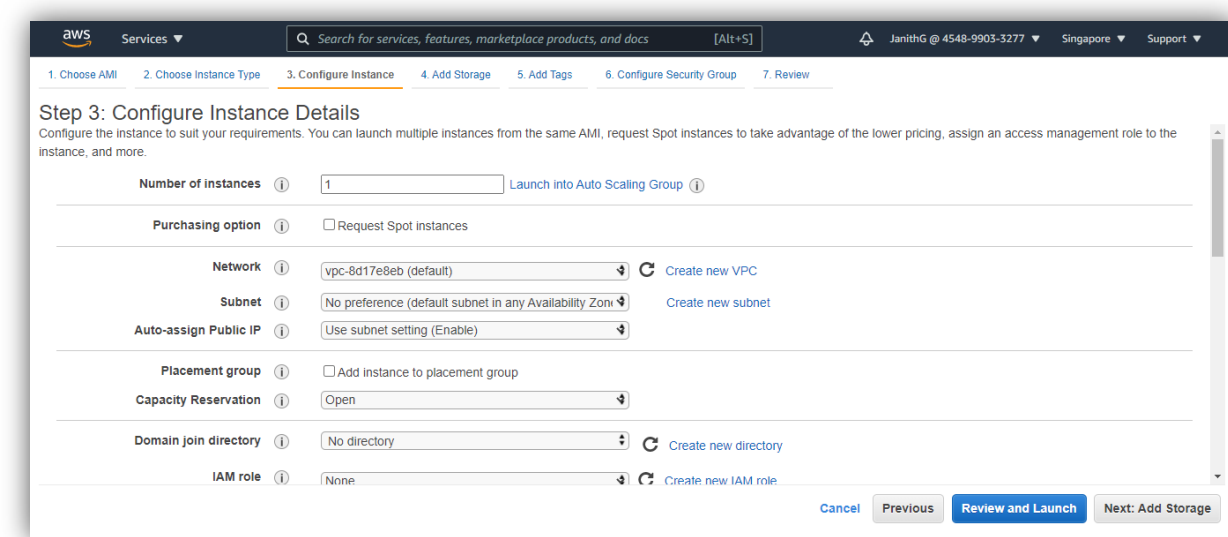


Figure 5

3.6. Add storage and tags

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 (MB/s)	Delete on Termination	Encryption
Root	/dev/xvda	snap-0bc2b223911166ef2	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypt

[Add New Volume](#)

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

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Add Tags](#)

Figure 6

3.7. Configure a Security Group

3.7.1. Allow HTTP, SSH and ICMP -IPv4 protocols from anywhere to access the server publicly

- HTTP – Access the web service
- SSH – Access the EC2 instance externally via SSH
- ICMP – Ipv4 – To activate ping

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 name:

Description:

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Anywhere	e.g. SSH for Admin Desktop
HTTP	TCP	80	Anywhere	e.g. SSH for Admin Desktop
All ICMP - IPv4	ICMP	0 - 65535	Anywhere	e.g. SSH for Admin Desktop

[Add Rule](#)

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Add Tags](#)

Figure 7

3.8. Create a new key pair (Public and Private key) and save it safe . This key is needed to access the sever from externally.

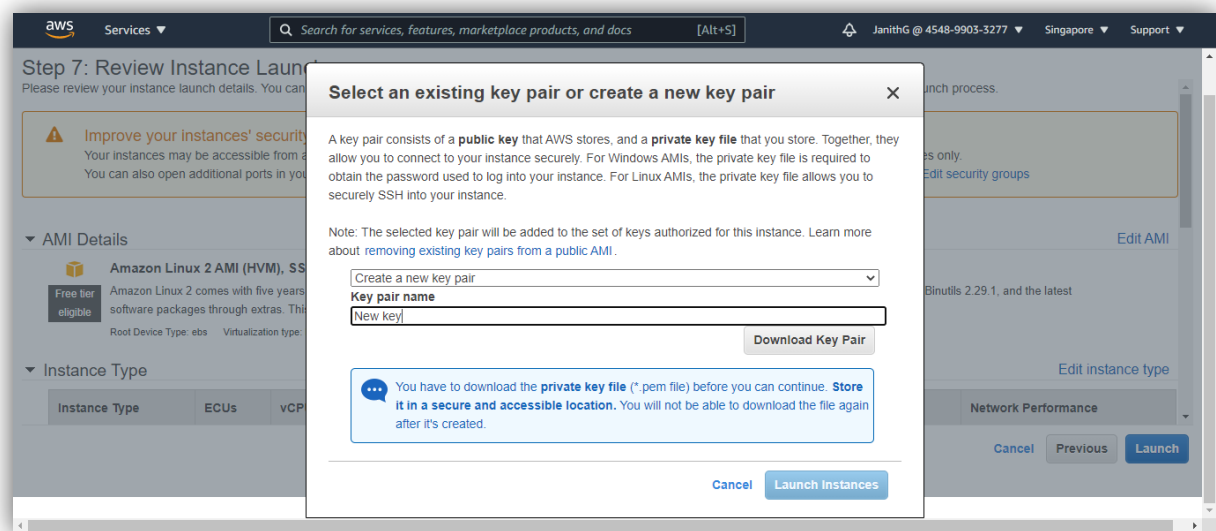


Figure 8

3.9. Review and launch the new Instance

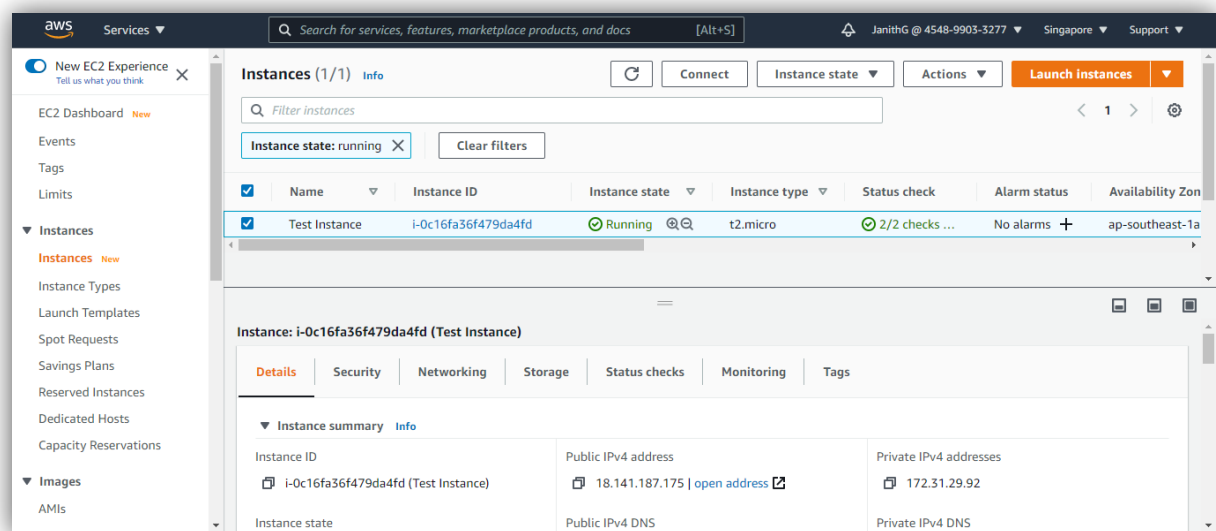


Figure 9

Checking if the newly created EC2 instance can be accessed publicly (SSH using putty)

1. Install PuTTY
2. Open PuTTYgen and load the .pem key which downloaded earlier

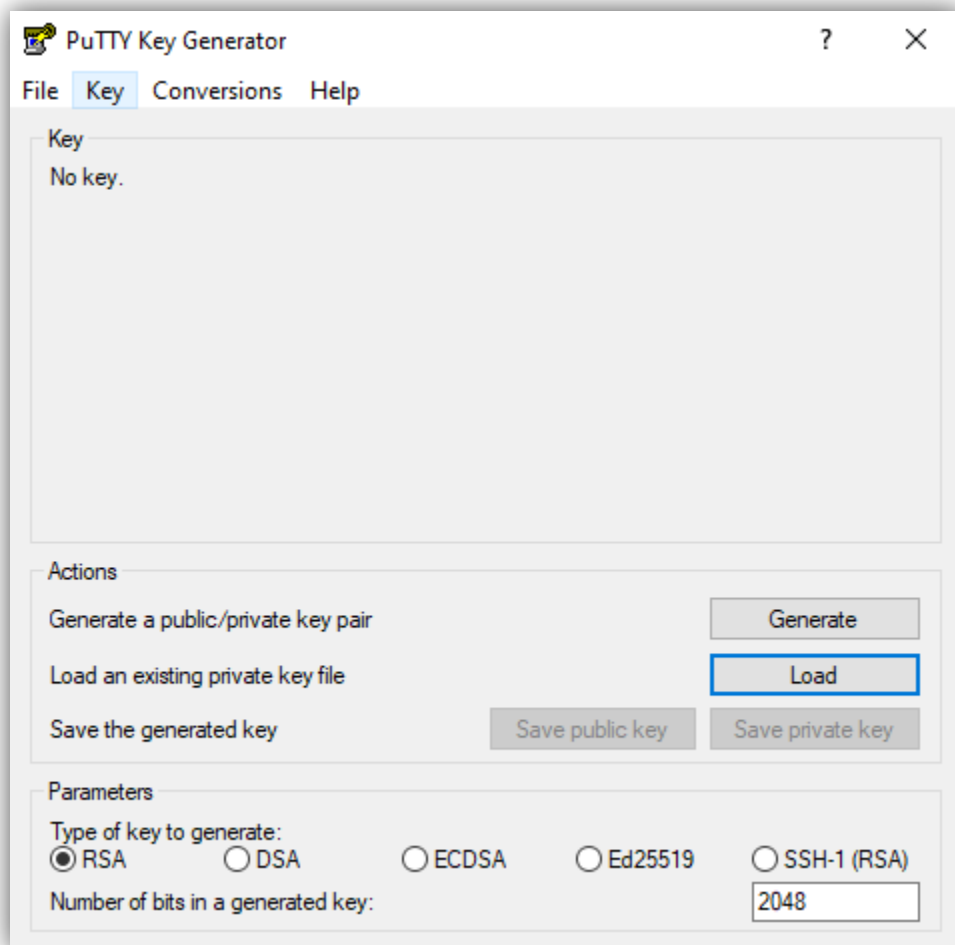


Figure 10

3. Click "Save private key" to save the converted .ppk file using the same file name

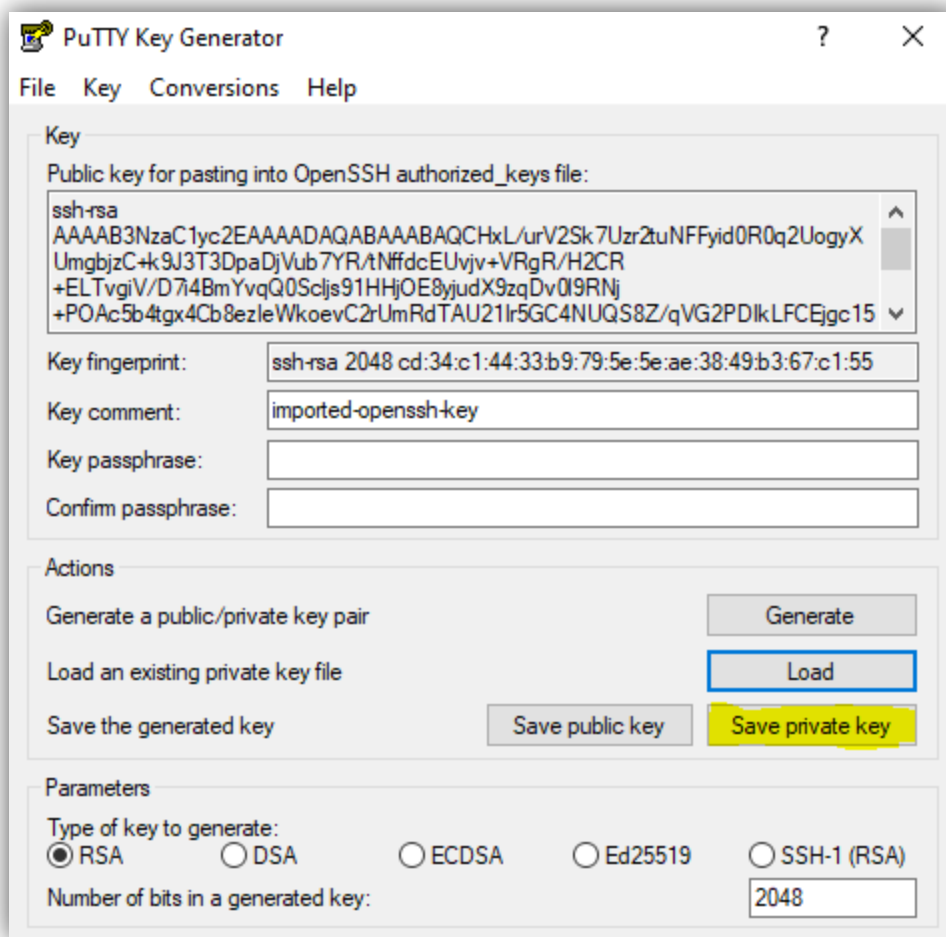


Figure 11

4. Open PuTTY and enter the following data then click "open"
 - Hostname – copy the public IPv4 DNS from EC2 instance
 - Connection type – SSH
 - Connection -> SSH -> Auth and select .ppk private key file
 - Connection -> Data -> Enter login username as "ec2-user"

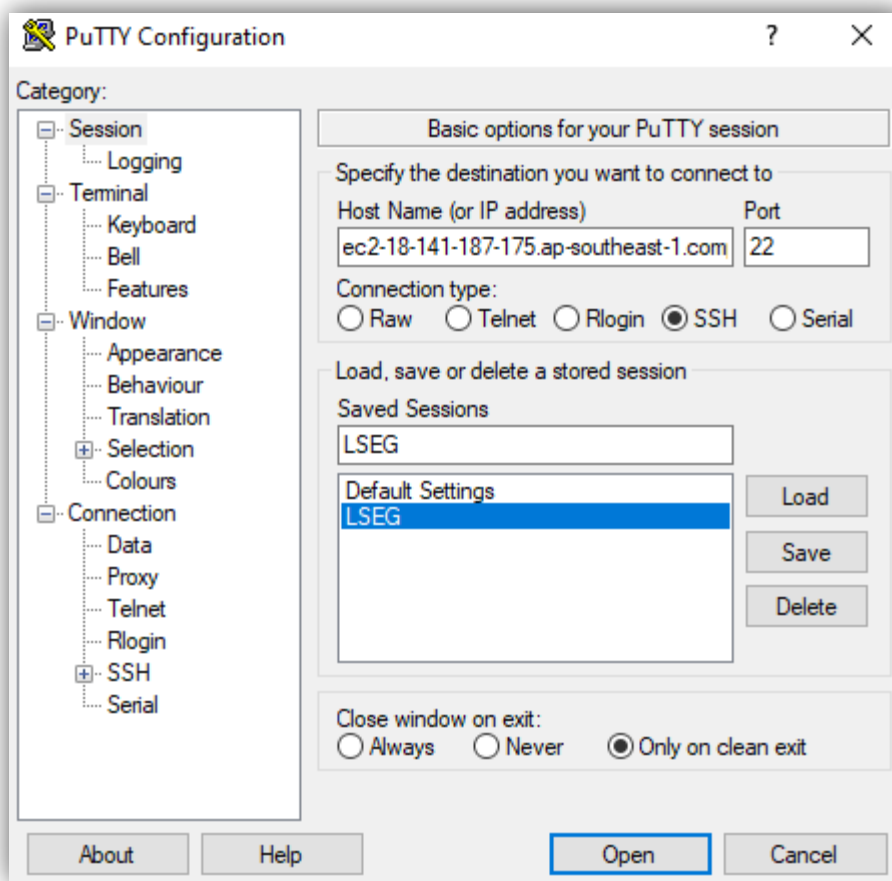


Figure 12

5. Successful connection will display the following window

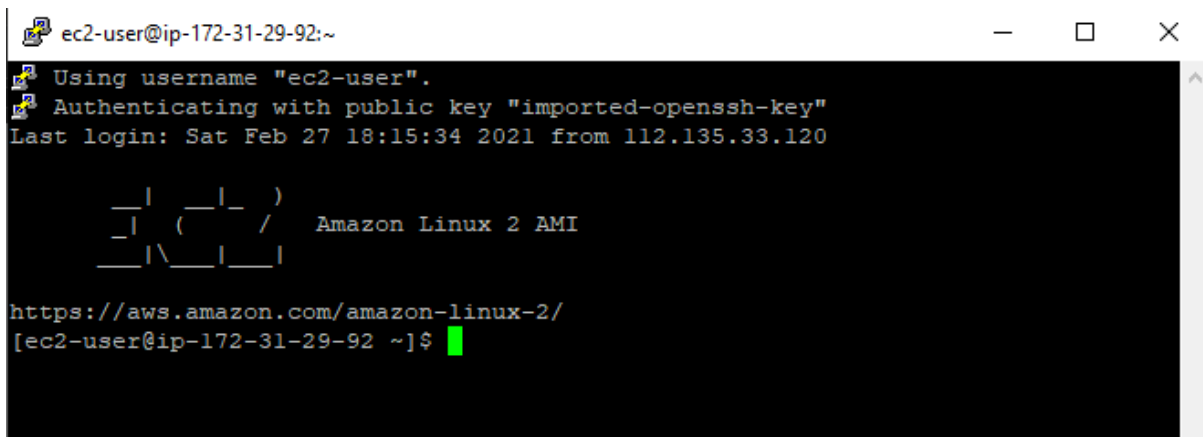


Figure 13

Q2 - Installing Apache server on EC2 instance

1. Update the server in the first launch

sudo yum update

```
$ sudo yum update -y
Redirecting to '/usr/bin/dnf update -y' (see 'man yum2dnf')

Last metadata expiration check: 2:50:07 ago on Tue Jul 11 07:14:53 2017.
Dependencies resolved.
Nothing to do.
Complete!
$ yum list installed
Redirecting to '/usr/bin/dnf list installed' (see 'man yum2dnf')

Last metadata expiration check: 6 days, 6:11:04 ago on Wed Jul 5 04:14:26 2017.
Installed Packages
GConf2.x86_64                               3.2.6-16.fc24
GeoIP.x86_64                               1.6.11-1.fc25
GeoIP-GeoLite-data.noarch                 2017.04-1.fc25
LibRaw.x86_64                             0.17.2-1.fc25
ModemManager.x86_64                      1.6.4-1.fc25
ModemManager-glib.x86_64                 1.6.4-1.fc25
NetworkManager.x86_64                   1:1.4.4-5.fc25
NetworkManager-ads1.x86_64              1:1.4.4-5.fc25
NetworkManager-bluetooth1.x86_64        1:1.4.4-5.fc25
NetworkManager-config-connectivity-fedora.x86_64 1:1.4.4-5.fc25
NetworkManager-glib.x86_64              1:1.4.4-5.fc25
NetworkManager-l2tp.x86_64              1.2.6-1.fc25
NetworkManager-libnm.x86_64             1:1.4.4-5.fc25
NetworkManager-libreswan.x86_64         1.2.4-1.fc25
```

Figure 14

2. Install the HTTP server

sudo yum -y install httpd

systemctl start httpd

systemctl enable httpd

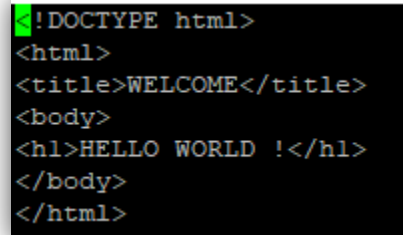
```
[root@host ~]# yum install httpd
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirror.1000mbps.com
 * extras: mirror.prolocation.net
 * updates: centos.mirror.triple-it.nl
Setting up Install Process
Resolving Dependencies
--> Running transaction check
--> Package httpd.i686 0:2.2.15-26.el6.centos will be installed
--> Processing Dependency: httpd-tools = 2.2.15-26.el6.centos for p
--> Processing Dependency: libaprutil-1.so.0 for package: httpd-2.2
--> Processing Dependency: libapr-1.so.0 for package: httpd-2.2.15-
--> Processing Dependency: apr-util-ldap for package: httpd-2.2.15-
--> Processing Dependency: /etc/mime.types for package: httpd-2.2.1
```

Figure 15

3. Check the status of the http service

systemctl status httpd

4. Go to **/var/www/html** and edit the **index.html** file

A terminal window with a black background and green text. The text is an HTML document structure:

```
!DOCTYPE html>
<html>
<title>WELCOME</title>
<body>
<h1>HELLO WORLD !</h1>
</body>
</html>
```

Figure 16

5. Open web browser and type the public IP and press Enter to display the message

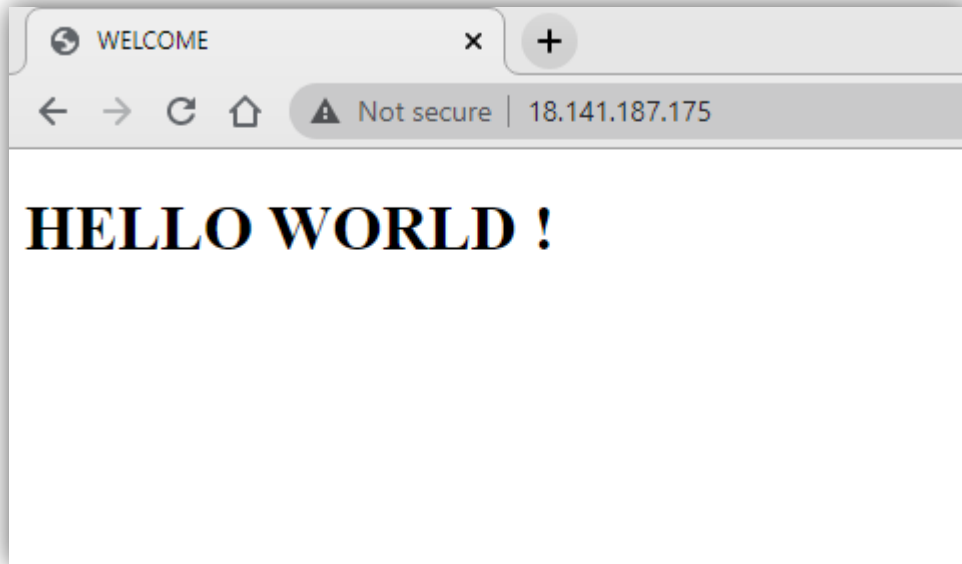


Figure 17

Q3 (i) - SSH to EC2 instance and check for web server status

1. Create access to EC2 instance from outside without private key
 - 1.1. Generate SSH key –
ssh-keygen -t rsa -m PEM
 - 1.2. Check for the key saved in ~/.ssh folder –
ls -lah ~/.ssh
 - 1.3. Copy the public key to EC2 instance –
cat ~/.ssh/id_rsa.pub | sudo ssh -i /home/janith/Downloads/key.pem ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "cat >> /home/ec2-user/.ssh/authorized_keys"
 - 1.4. Try SSH into server without key –
SSH ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com

Figure 18

- # ps -ef | grep -c httpd**
3. If the count is equal to 1, then the web services are down. If the count is greater than 2, then the web services are up and running.

4. If the web services are down, using the below command, services can be started.

5. Following script is created to process the above tasks at once

```
#!/bin/bash
count=$(ssh ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "ps -ef | grep -c httpd")
if [ "$count" -gt 2 ]
then
echo "HTTP Service is Running"
else
echo "HTTP Service is Not Running"
echo "HTTP Service is starting..."
ssh ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "sudo systemctl start httpd"
echo "HTTP Service has now started"
fi
```

Figure 19

6. Follow image shows the result of above script

```
root@J-ubuntu:/home/janith/Downloads# bash t.sh
HTTP Service is Running
root@J-ubuntu:/home/janith/Downloads#
```

Figure 20

```
root@J-ubuntu:/home/janith/Downloads# bash t.sh
HTTP Service is Not Running
HTTP Service is starting...
HTTP Service has now started
root@J-ubuntu:/home/janith/Downloads#
```

Figure 21

Q3 (ii) – Check if the web server is serving the content and return 200 code

1. Run the following command to get the details of the HTTP server

curl -I <http://localhost>

2. Below command will check the top row for the status code “200” and return the word count

curl -I <http://localhost> 2>/dev/null | head -n 1 | grep 200 | wc -l

3. Check the above result in an if condition. If the word count is equal to 1, then the web server is serving the content.

```
#!/bin/bash
STATUS=$(ssh ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "curl -I http://localhost 2>/dev/null | head -n 1 | grep 200 | wc -l")
if [ "$STATUS" -eq 1 ]
then
    echo $(ssh ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "curl -I http://localhost 2>/dev/null | head -n 1")
else
    echo "HTTP Error ! " $(ssh ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "curl -I http://localhost 2>/dev/null | head -n 1")
fi
```

Figure 22

4. Above script will return the below result

```
root@J-ubuntu:/home/janith/Downloads# bash t.sh
HTTP/1.1 200 OK
root@J-ubuntu:/home/janith/Downloads#
```

Figure 23

Q3 (iii) – Save the script result in a database

1. DynamoDB in AWS is used for this section
2. Create a new table with required fields and their respected data types

Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name*

Primary key* Partition key

☐ Add sort key

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

☒ Use default settings

- No secondary indexes.
- Auto Scaling capacity set to 70% target utilization, at minimum capacity of 5 reads and 5 writes.
- Encryption at Rest with DEFAULT encryption type.

Figure 24

script_log Close

Overview **Items** Metrics Alarms Capacity Indexes Global Tables Backups More

Create item Actions

Scan: [Table] script_log: time_stamp Viewing 1 to 1 Items

Scan [Table] script_log: time_stamp

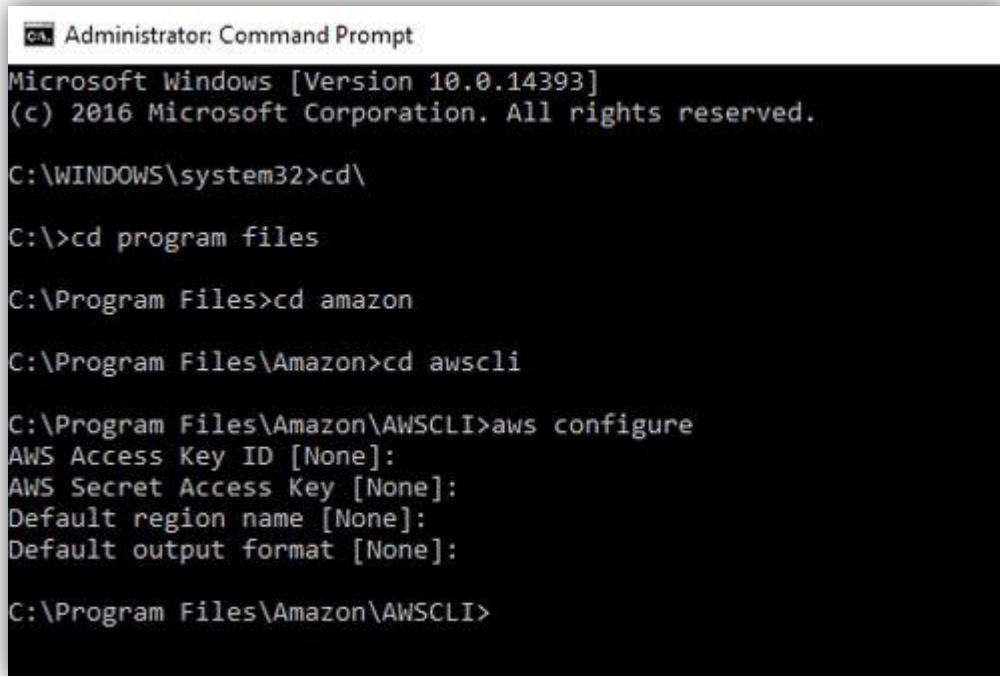
Add filter

Start search

time_stamp	result
2021-02-28 09:31:41	HTTP Service is Running

Figure 25

3. Configure AWS CLI in the server to perform rest of the commands
 - Enter the Key ID for the AWS user
 - Enter the Secret Access Key provided for the user
 - Enter the region name used for the AWS services and configure



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd\

C:\>cd program files

C:\Program Files>cd amazon

C:\Program Files\Amazon>cd awscli

C:\Program Files\Amazon\AWSCLI>aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]:
Default output format [None]:

C:\Program Files\Amazon\AWSCLI>
```

Figure 26

4. Data can be inserted into this DynamoDB using below command

```
aws dynamodb put-item \
--table-name script_log \
--item \
'{"time_stamp": {"S": \"\" + $dt + \"\"}, "result": {"S": "HTTP Service is Running"}}' \
--return-consumed-capacity TOTAL
~
~
```

Figure 27

```
root@J-ubuntu:/home/janith/Downloads# bash t.sh
{
  "ConsumedCapacity": {
    "TableName": "script_log",
    "CapacityUnits": 1.0
  }
}
root@J-ubuntu:/home/janith/Downloads#
```

Figure 28

```
#Check the return code of the http server
STATUS=$(ssh ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "curl -I http://localhost 2>/dev/null | head -n 1 | grep 200 | wc -l")

#Returns 1 as the word count if the server returns 200 code
if [ "$STATUS" -eq 1 ]
then
    cod1=$(ssh ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "curl -I http://localhost 2>/dev/null | head -n 1")
    echo $cod1

# Insert log into dynamodb
    aws dynamodb put-item \
        --table-name script_log \
        --item '{
            "time_stamp": {"S": "'$dt'"}, "result": {"S": "'$globalstatus'"}, "200_result": {"S": "HTTP/1.1 200 OK"}}'

#Send email to application support team if data insert is not successful
    if [ $? -ne 0 ]
    then
        echo -e "to: gedarawatta.j@gmail.com\nsubject:Log Upload failed\nLog files upload failed to DynamoDB" | ssmtp gedaraw
        attaj@gmail.com
        fi

else
    cod2="HTTP Error ! " $(ssh ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "curl -I http://localhost 2>/dev/null | head -n 1")

```

Figure 29

5. Pushed data is stored in the DynamoDB

Scan: [Table] script_log: time_stamp ^

Viewing 1 to 1 items

Scan

[Table] script_log: time_stamp

^

+ Add filter

Start search

<input type="checkbox"/>	time_stamp ⓘ ^	result ^	200_result ^	
<input type="checkbox"/>	2021-02-28 19:23:42	HTTP Service is Running	HTTP/1.1 200 OK	

Figure 30

Q3 (iv) - Sending email to the App Support Team with suitable error message

1. Configuring SMTP in the server – Install SMTP

```
# sudo apt-get install ssmtp
```

2. Go to `/etc/ssmtp/` and edit `ssmtp.conf` file with below configurations

```
mailhub=smtp.gmail.com:587
useSTARTTLS=YES
AuthUser=username-here
AuthPass=password-here
TLS_CA_File=/etc/pki/tls/certs/ca-bundle.crt
```

3. Use the below command to send emails to the App Support Team

```
# -e "to: gedarawatta.j@gmail.com\nsubject:Scrip Failed\nAn error has occurred while\nexecuting the script" | ssmtp gedarawatta.j@gmail.com
```

```
root@J-ubuntu:/home/janith/Downloads#
root@J-ubuntu:/home/janith/Downloads# bash sc.sh
HTTP Service is Running
HTTP/1.1 200 OK

An error occurred (ResourceNotFoundException) when calling the PutItem operation: Requested resource not found
root@J-ubuntu:/home/janith/Downloads#
```

Figure 31

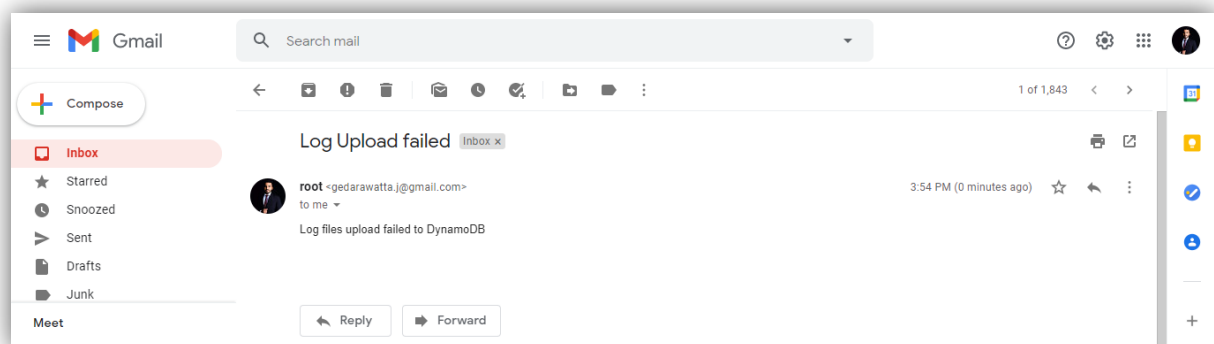


Figure 32

Q4 (i) – Collect log files of the web server daily and create a compressed file

1. Access the log files of HTTP server

```
# cd /var/log/httpd/
```

2. Create a compressed file of daily log files in location /tmp/ renaming it to current date

```
# sudo tar -czf /tmp/http_logs_`date +%F`.tar.gz /var/log/httpd/
```

```
[root@ip-172-31-29-92 tmp]# ls -ltr
total 12
drwx----- 3 root root 17 Feb 25 08:13 systemd-private-ab488126c6a24f42ac6b756d35809bed-chronyd.service-sxS2HR
drwx----- 3 root root 17 Feb 28 07:04 systemd-private-ab488126c6a24f42ac6b756d35809bed-httpd.service-2EBRIL
-rw-r--r-- 1 root root 8817 Feb 28 08:11 http_logs_2021-02-28.tar.gz
[root@ip-172-31-29-92 tmp]#
```

Figure 33

Q4 (ii) – Move the compressed file to the location of the script

1. Following command will copy the above created compressed file to the location where script is running

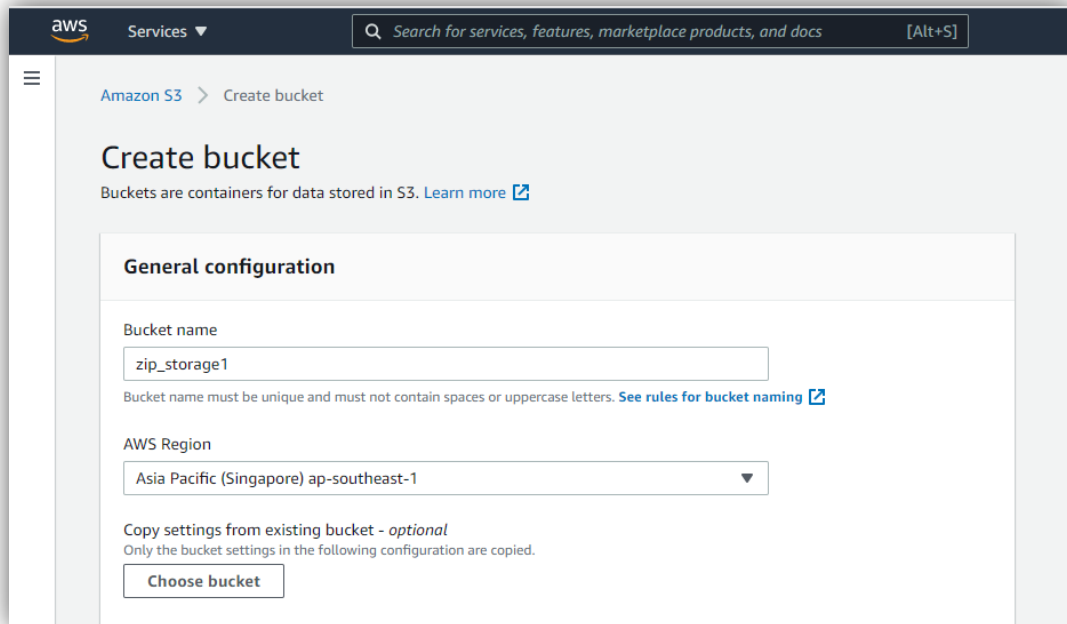
```
# ssh -tt ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "sudo tar -
czf /tmp/http_logs_`date +%F`.tar.gz /var/log/httpd/";
```

```
root@J-ubuntu:/home/janith/Downloads# bash t.sh
http_logs_2021-02-28.tar.gz 100% 8817 112.3KB/s 00:00
root@J-ubuntu:/home/janith/Downloads#
root@J-ubuntu:/home/janith/Downloads# ls
complete.sh http_logs_2021-02-28.tar.gz key.pem sc.sh t.sh
root@J-ubuntu:/home/janith/Downloads#
```

Figure 34

Q4 (iii) – Upload the compressed file to S3 bucket

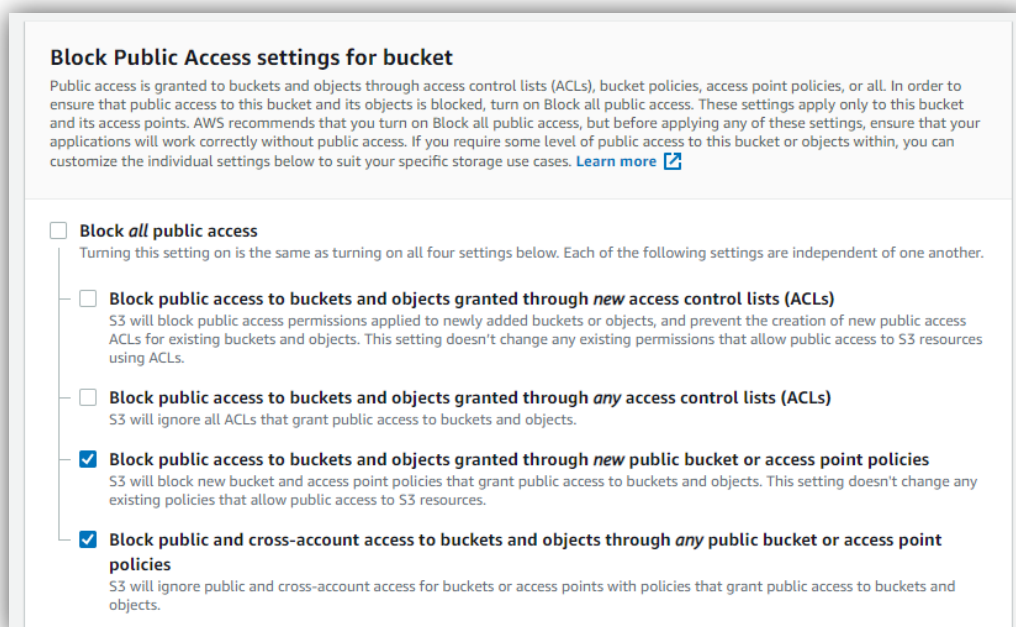
1. Create a S3 bucket in AWS



The screenshot shows the AWS 'Create bucket' page. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, and a search bar. Below the navigation bar, the breadcrumb 'Amazon S3 > Create bucket' is visible. The main heading is 'Create bucket', followed by a subtext: 'Buckets are containers for data stored in S3. [Learn more](#)'. The 'General configuration' section contains a 'Bucket name' input field with 'zip_storage1', a note about naming rules, an 'AWS Region' dropdown set to 'Asia Pacific (Singapore) ap-southeast-1', and a 'Copy settings from existing bucket - optional' section with a 'Choose bucket' button.

Figure 35

2. Apply below settings to access the S3 bucket from externally



The screenshot shows the 'Block Public Access settings for bucket' page. It starts with a heading and a paragraph explaining public access and the recommendation to block it. Below this, there are four settings, each with a checkbox and a description: 1. 'Block all public access' (unchecked), 2. 'Block public access to buckets and objects granted through new access control lists (ACLs)' (unchecked), 3. 'Block public access to buckets and objects granted through any access control lists (ACLs)' (unchecked), and 4. 'Block public access to buckets and objects granted through new public bucket or access point policies' (checked). The last setting is also checked for 'Block public and cross-account access to buckets and objects through any public bucket or access point policies'.

Figure 36

3. After creating the S3 bucket, files can be uploaded to is externally using the below command

```
# aws s3 cp http_logs_`date +%F`.tar.gz s3://zip-upload-storage/--acl public-read
```

4. Below script will upload the compressed file into the S3 bucket successfully

```
# Upload tar file to the s3 bucket

aws s3 cp http_logs_`date +%F`.tar.gz s3://zip-upload-storage/ --acl public-read

if [ "$?" -eq "0" ];
then
# Delete the tar file from the script location if the upload is successful
    echo "Successfully uploaded to S3 Bucket"
```

Figure 37

```
root@J-ubuntu:/home/janith/Downloads# bash t.sh
upload: ./http_logs_2021-02-28.tar.gz to s3://zip-upload-storage/http_logs_2021-02-28.tar.gz
UPLOAD SUCCESS
root@J-ubuntu:/home/janith/Downloads#
```

Figure 38

Objects (2)

Objects are the fundamental entities stored in Amazon S3. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

↻
Delete
Actions ▾
Create folder
Upload

< 1 >
⚙



<input type="checkbox"/>	Name ▲	Type ▾	Last modified ▾	Size ▾	Storage class ▾
<input type="checkbox"/>	 http_logs_2021-02-27.tar.gz	gz	February 27, 2021, 15:04:04 (UTC+05:30)	6.8 KB	Standard
<input type="checkbox"/>	 http_logs_2021-02-28.tar.gz	gz	February 28, 2021, 14:03:57 (UTC+05:30)	8.6 KB	Standard

Figure 39

Q4 (iv) – Delete the compressed file if successful and send the email if not successful

1. Below command will delete the compressed file from the location

```
# rm -rf http_logs_`date +%F`.tar.gz
```

2. Below command will send an email to Application Support Team if the upload was unsuccessful

```
# echo -e "to: gedarawatta.j@gmail.com\nsubject:Zip File Upload Failed\nhttp_logs_`date +%F`.tar.gz - file upload failed" | ssmtp gedarawatta.j@gmail.com
```

3. Below script will run the above commands

```
# Upload tar file to the s3 bucket
aws s3 cp http_logs_`date +%F`.tar.gz s3://zip-upload-storage/ --acl public-read
if [ $? -eq 0 ];
then
# Delete the tar file from the script location if the upload is successful
echo "Successfully uploaded to S3 Bucket"
rm -rf http_logs_`date +%F`.tar.gz
else
# Send email to application support team if upload failed
echo "Upload failed to S3 Bucket"
echo -e "to: gedarawatta.j@gmail.com\nsubject:Zip File Upload Failed\nhttp_logs_`date +%F`.tar.gz - file upload failed" | ssmtp gedarawatta.j@gmail.com
fi
```

Figure 40

```
root@J-ubuntu:/home/janith/Downloads# bash t.sh

The user-provided path http_logs_2021-02-28.tar.gz does not exist.
UPLOAD FAIL
root@J-ubuntu:/home/janith/Downloads#
```

Figure 41

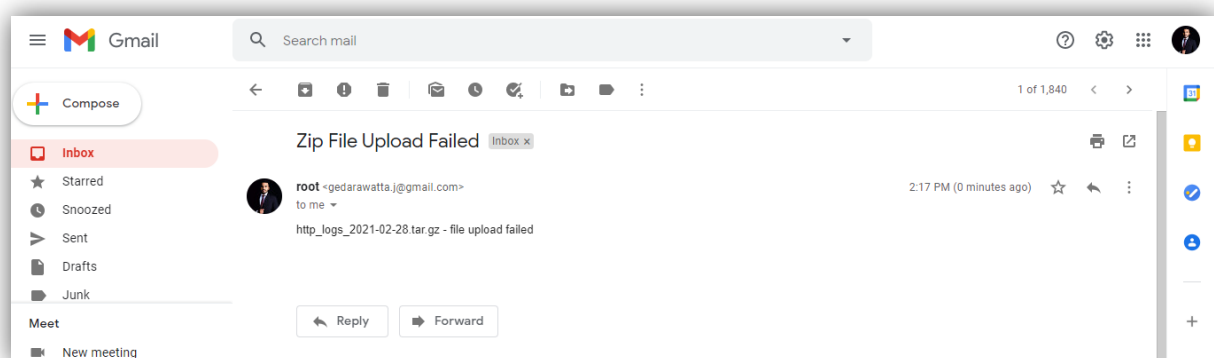


Figure 42

Complete Script for Question 03

```
#!/bin/bash

#SSH to the EC2 instance and check if the http server is running
count=$(ssh ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "ps -ef | grep -c httpd")
status1="HTTP Service is Running"
status2="HTTP Service is not Running"
dt=$(date +%F %T)

#Returns 2 if the server is down
if [ "$count" -gt 2 ]
then
    echo $status1
    globalstatus="HTTP Service is Running"
# Returns a non 0 value if an error occurred
    if [ $? -ne 0 ]
    then
        # Send an email to Application Support Team with the error
        echo -e "to: gedarawatta.j@gmail.com\nsubject:Log Upload failed\nLog files upload failed to DynamoDB" | ssmtp gedarawatta.j@gmail.com
    fi
else
    echo $status2
    globalstatus="HTTP Service is not Running"
    echo "HTTP Service is starting..."
#Start the web server if it is not running already
    ssh ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "sudo systemctl start httpd"
    echo "HTTP Service has now started"

    if [ $? -ne 0 ]
    then
        echo -e "to: gedarawatta.j@gmail.com\nsubject:Log Upload failed\nLog files upload failed to DynamoDB" | ssmtp gedarawatta.j@gmail.com
    fi
fi
```

Figure 43

```
#Check the return code of the http server
STATUS=$(ssh ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "curl -I http://localhost 2>/dev/null | head -n 1 | grep 200 | wc -l")

#Returns 1 as the word count if the server returns 200 code
if [ "$STATUS" -eq 1 ]
then
    cod1=$(ssh ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "curl -I http://localhost 2>/dev/null | head -n 1")
    echo $cod1

# Insert log into dynamodb
    aws dynamodb put-item \
        --table-name script_log \
        --item \
        '{"time_stamp": {"S": "'$dt'"}, "result": {"S": "'$globalstatus'"}, "200_result": {"S": "HTTP/1.1 200 OK"}}'

#Send email to application support team if data insert is not successful
    if [ $? -ne 0 ]
    then
        echo -e "to: gedarawatta.j@gmail.com\nsubject:Log Upload failed\nLog files upload failed to DynamoDB" | ssmtp gedarawatta.j@gmail.com
    fi
else
    cod2="HTTP Error ! " $(ssh ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "curl -I http://localhost 2>/dev/null | head -n 1")
    echo $cod2

# Insert log into dynamodb
    aws dynamodb put-item \
        --table-name script_log \
        --item \
        '{"time_stamp": {"S": "'$dt'"}, "result": {"S": "'$globalstatus'"}, "200_result": {"S": "HTTP Error"}}'

#Send email to application support team if data insert is not successful
    if [ $? -ne 0 ]
    then
        echo -e "to: gedarawatta.j@gmail.com\nsubject:Log Upload failed\nLog files upload failed to DynamoDB" | ssmtp gedarawatta.j@gmail.com
    fi
fi
```

Figure 44

Complete Script for Question 04

```
#!/bin/bash

# Check /var/log/httpd for log files and copy them into tar file
ssh -tt ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "sudo tar -czf /tmp/http_logs_`date +%F`.tar.gz /var/log/httpd/";

if [ $? -eq 0 ]
then
    echo "Successfully connected to remote host and created the compressed log folder"
else
    echo "Connection to remote host failed or compressing log folder failed"
fi

# Download the tar file to the location of the script
scp ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com:/tmp/http_logs_`date +%F`.tar.gz .

if [ $? -eq 0 ]
then
    echo "Successfully copied the log folder to current working directory"
else
    echo "Copying log folder to the current working directory failed"
fi

# Upload tar file to the s3 bucket
aws s3 cp http_logs_`date +%F`.tar.gz s3://zip-upload-storage/ --acl public-read

if [ "$?" -eq "0" ];
then
    # Delete the tar file from the script location if the upload is successful
    echo "Successfully uploaded to S3 Bucket"
    rm -rf http_logs_`date +%F`.tar.gz
else
    # Send email to application support team if upload failed
    echo "Upload failed to S3 Bucket"
    echo -e "to: gedarawatta.j@gmail.com\nsubject:Zip File Upload Failed\nhttp_logs_`date +%F`.tar.gz - file upload failed" | ssmtp gedarawatta.j@gmail.com
fi
```

Figure 45

Script for updating server and installing HTTP server

```
#!/bin/bash

# Update the server
ssh ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "sudo yum update -y";

# Install http server
ssh ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "sudo yum install httpd";

# Start http services
ssh ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "sudo systemctl start httpd";

# Enable http services
ssh ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "sudo systemctl enable httpd";

# Insert data into index.html
ssh ec2-user@ec2-18-141-187-175.ap-southeast-1.compute.amazonaws.com "echo "Hello World" > /var/www/html/index.html"
```

Figure 46

Steps to run the scripts

In order to perform the above scripts following setup should be configured

- AWS EC2 instance – Linux
- External machine – Linux/Ubuntu
- Open ports of 22 and 80 on EC2 instance
- HTTP services installed on EC2 instance
- S3 bucket
- DynamoDB table
- Keyless SSH to EC2 instance from external machine
- AWS CLI configured on external machine
- SMTP server configured on external machine

Order of running the scripts

1. Run terraform to provision

terraform plan

terraform apply

2. Run **httpconf.sh** to Update the newly created EC2 instance server and install HTTP service

bash httpconf.sh

3. Run **script1.sh** which contains the commands for Question 3

bash script1.sh

Note : #bash script1.sh > log_script1.txt will collect the log files to **log_script1.txt**

4. Run **script2.sh** which contains the commands for Question 4

bash script2.sh

Note : #bash script2.sh > log_script2.txt will collect the log files to **log_script2.txt**

Automating the setup

- Infrastructure provisioning is automated with terraform for EC2, Security groups, S3 Bucket , and DynamoDB instances.
- Installing HTTP related configurations in provisioned EC2 remote host can be done through running the script.
- Part one of the assignment can be fulfilled by running the first bash script.
- Part two of the assignment can be fulfilled by running the second bash script.
- In order to facilitate the Business requirement of make the website available 9*5, cronjobs are configured to run the scripts on daily, at the start of the day.

Key Points of high availability and reliability of the proposed enterprise architecture

- In order to ensure the availability of the architecture, with minimized downtime, the architecture can be deployed into multiple availability zones inside a single region, unless the multi-region deployment is necessary.
- Furthermore, in order to distribute the incoming traffic to web servers across multiple Availability Zones, internet facing load balancer can be used. Load Balancer is not only capable of distributing the load between instances, but also it will recognize and respond to unhealthy instances.
- Also to ensure that the architecture can handle changes in demand, the web server instances can be placed inside auto-scaling group. It will launch and terminate instances based on the specified conditions. The auto scaling groups also need to be registered with the load balancer.

Technologies Used

- Linux
- Ubuntu
- Amazon Web Services (AWS)
- Identify and Access Management (IAM)
- Amazon Machine Images (AMI)
- Amazon Simple Storage Service (S3)
- Amazon Elastic Compute Cloud (EC2)
- DynamoDB
- Terraform