



**SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY**  
(An Autonomous Institution. Affiliated to Anna University, Chennai)  
**Kuniamuthur, Coimbatore - 641 008**



## **DEPARTMENT OF INFORMATION TECHNOLOGY**

### **22AD401 - CLOUD COMPUTING LABORATORY**

#### **Continuous Assessment Record**

Submitted by

Name :.....

Register No :.....

Degree & Branch :.....

Class & Semester :.....

Academic Year :.....



**SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY**  
(An Autonomous Institution. Affiliated to Anna University, Chennai)  
**Kuniamuthur, Coimbatore - 641 008**



## **DEPARTMENT OF INFORMATION TECHNOLOGY**

### **22AD401 - CLOUD COMPUTING LABORATORY**

#### **Continuous Assessment Record**

#### **Submitted by**

**Name :** ..... **Register No.** : .....

**Class/Semester :**..... **Degree & Branch :**.....

**Faculty In-charge**

**Head of the Department**

#### **BONAFIDE CERTIFICATE**

This is to certify that this record is the bonafide record of work done by Mr./Ms.

---

During the academic year 2023–2024.

Submitted for the practical examination held on \_\_\_\_\_

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**



## DEPARTMENT OF INFORMATION TECHNOLOGY

### 22AD401-CLOUD COMPUTING LABORATORY

#### Record of laboratory

EVEN SEMESTER: 2023-2024

Name of the Faculty	Ms.A.Raihana
---------------------	--------------

#### CONTINUOUS EVALUATION SHEET

#### REFERENCES RUBRICS TABLE

Criteria	Range of Marks			
	Excellent	Good	Average	Below Average
Aim (10 marks)	9-10	7-8	5-6	0-4
Procedure (30 Marks)	27-30	21-26	15-20	0-14
Demo (40 Marks)	37-40	31-36	25-30	0-24
Result (10 Marks)	9-10	7-8	5-6	0-4
Viva (10 Marks)	9-10	7-8	5-6	0-4
Overall Marks	90-100	70-89	50-69	0-49

## **INDEX**

<b>Ex.No</b>	<b>Name of the Experiment</b>	<b>Page No.</b>
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		



# **DEPARTMENT OF INFORMATION TECHNOLOGY**

**22AD401-CLOUD COMPUTING LABORATORY**

**EVEN SEMESTER : 2023-2024**

## **EX:NO -1**

**Create an S3 bucket with the following requirements**

**Create an S3 bucket in the Asia location of 'ap-south-1'(Mumbai).**

**Create a folder in the name of 'myfolder' in the created bucket.**

**Upload a file named 'mybiodata.pdf' under the folder 'myfolder'.**

**Make the object 'mybiodata.pdf' file accessible to everyone(publicly).**

### **Aim:**

To create an s3 bucket in ap-south-1 region and to create folder and upload a file to the bucket which can be accessible to everyone.

### **Description:**

#### **Bucket Creation:**

1. Login to the AWS console.
2. Search for S3 service in search bar.
3. Click 'Create bucket' in S3 service.
4. Select ap-south-1 (Mumbai) AWS Region.
5. Give unique name to the bucket.
6. Enable ACL and change object ownership to Object Writer.
7. Allow public access to the bucket objects.
8. Finally Click 'Create Bucket' and your Bucket is created.

#### **Folder Creation:**

1. Select your bucket from the bucket list.
2. Click Create folder and create a folder with name 'myfolder'.

#### **File Upload:**

1. Select the folder 'myfolder' in your bucket and Click Upload to upload a file.
2. Now upload a pdf file 'mybiodata' from your local storage to the folder inside the created S3 bucket.
3. Click on the file inside the folder and go to its properties.
4. In properties edit the accessibility of the object with public access.

## Output:

The screenshot shows the AWS S3 console interface. At the top, there's a header bar with the URL [s3.console.aws.amazon.com/s3/home?region=ap-south-1&bucketType=general](https://s3.console.aws.amazon.com/s3/home?region=ap-south-1&bucketType=general). Below the header, the main content area is titled "Account snapshot". It includes a "View Storage Lens dashboard" button. Under "General purpose buckets", there is a sub-section titled "General purpose buckets (1) [Info](#)". A note says "Buckets are containers for data stored in S3." Below this is a search bar labeled "Find buckets by name". A table lists one bucket: "apsouth1bucketnew" located in "Asia Pacific (Mumbai) ap-south-1". The table columns are Name, AWS Region, Access, and Creation date. The bucket details show "Objects can be public" and was created on "March 3, 2024, 14:23:21 (UTC+05:30)".

This screenshot shows the AWS S3 console for the specific bucket "apsouth1bucketnew". The URL is [s3.console.aws.amazon.com/s3/buckets/apsouth1bucketnew?region=ap-south-1&tab=objects](https://s3.console.aws.amazon.com/s3/buckets/apsouth1bucketnew?region=ap-south-1&tab=objects). The page title is "apsouth1bucketnew [Info](#)". The navigation bar shows "Amazon S3 > Buckets > apsouth1bucketnew". Below the title, there are tabs for "Objects", "Properties", "Permissions", "Metrics", "Management", and "Access Points". The "Objects" tab is selected. The main content area shows a table with one object: "myfolder/" which is a "Folder". There are buttons for "Upload" and "Actions" (with options like Copy 53 URI, Copy URL, Download, Open, Delete). A note at the top says "Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions.".

The screenshot shows the AWS S3 console interface. At the top, the URL is <https://s3.console.aws.amazon.com/s3/buckets/apsouth1bucketnew?region=ap-south-1&bucketType=general&prefix=myfolder/>. The page displays a single object, 'mybiodata.pdf', which is a PDF file. The file's properties are as follows:

Name	Type	Last modified	Size	Storage class
mybiodata.pdf	pdf	March 3, 2024, 14:25:45 (UTC+05:30)	418.6 KB	Standard

Below the object list, the permissions tab is selected for the file 'mybiodata.pdf'. The access control list (ACL) shows the following grants:

Grantee	Object	Object ACL
Object owner (your AWS account) Canonical ID: 22cd19fd4553f6eba9f8d5087b935a82eb42a89f3b47155a443191e5d832978	Read	Read, Write
Everyone (public access) Group: <a href="http://acs.amazonaws.com/groups/global/AllUsers">http://acs.amazonaws.com/groups/global/AllUsers</a>	Read	-
Authenticated users group (anyone with an AWS account) Group: <a href="http://acs.amazonaws.com/groups/global/AuthenticatedUsers">http://acs.amazonaws.com/groups/global/AuthenticatedUsers</a>	-	-

## Result:

Thus a S3 bucket is created in Mumbai region with a folder having a file with public access.

## **EX: NO- 2**

**Create an EC2 Instance in the us-east-1 region with the following requirements.**

**Give the Name tag of both EC2 instance & keypair as "ec2usecase1"(Name).**

**EC2 instance AMI should be "Amazon Linux 2".**

**Allow SSH traffic for taking putty remote connection.**

**Allow HTTP traffic from the internet for reaching website requests.**

### **Aim:**

To create an EC2 instance of name 'ec2usecase1' in the us-east-1 region with a key pair name of 'ec2usecase1'. EC2 instance AMI should by 'Amazon Linux' and the instance should allow SSH and HTTP traffics.

### **Description:**

1. Login to your AWS console.
2. Change your AWS region to us-east-1 (N. Virginia).
3. Search for EC2 service in Search bar.
4. Click 'Launch Instance' there.
5. Give a name for your EC2 instance. As given in question, the name of the EC2 instance should be 'ec2usecase1'.
6. Select 'Amazon Linux 2023' as AMI for your instance.
7. Select Instance type. (For free tier account, select t2.micro).
8. Create a new key pair in the name of 'ec2usecase1'.
9. Allow SSH and HTTP traffic to the instance.
10. If we want to change our root volume for the instance, we can configure it in Configure storage.

## Output:

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with sections like EC2 Dashboard, Services, and Instances (which is currently selected). The main content area displays a table of instances. There is one instance listed:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DM
ec2usecase1	i-0bee3daa595726498	Running	t2.micro	2/2 checks passed	View alarms	us-east-1d	ec2-54-173-10

A modal window titled "Select an instance" is open at the bottom of the screen.

## Result:

Thus, an EC2 instance of AMI 'Amazon Linux' and name and keypair as 'ec2usecase1' which allows SSH and HTTP traffics across the internet has been successfully created.

### **EX: NO -3**

**Create an IAM group called 'Network-L1-Team' with 'AmazonVPCReadOnlyAccess' and 'AWSNetworkManagerReadOnlyAccess' policies, then add an IAM user called 'Network-L1-User1' to the group.**

**The name of the IAM group should be 'Network-L1-Team'.**

**The name of the IAM user should be 'Network-L1-User1'.**

**The 'AmazonVPCReadOnlyAccess' policy should be attached.**

**The 'AWSNetworkManagerReadOnlyAccess' policy should be attached.**

#### **Aim:**

To create an IAM user group named as 'Network-L1-Team' with 'AmazonVPCReadOnlyAccess' and 'AWSNetworkManagerReadOnlyAccess' policies, then add an IAM user called 'Network-L1-User1' to the group.

#### **Description:**

1. Login to your AWS console.
2. Search for 'IAM' service in search box.
3. Click on 'User groups' in side bar, to create an IAM user group.
4. Click 'Create group'.
5. Give 'Network-L1-Team' as name to the user group.
6. Attach following two policies to the user group.
  - 'AmazonVPCReadOnlyAccess'
  - 'AWSNetworkManagerReadOnlyAccess'
7. Click Create Group and your group is created.
8. Click on 'Users' in side bar, to create an IAM user.
9. Give 'Network-L1-User1' as user name for IAM user, and click next.
10. In permissions options, select 'Add user to the group' and select the user group we already created to add our newly created IAM user to that group.
11. Click 'Create user' and thus a new IAM user has been created.

## Output:

The screenshot shows the AWS IAM User Groups page for the 'Network-L1-Team' group. The 'Summary' section displays the group name 'Network-L1-Team', creation time 'March 04, 2024, 18:21 (UTC+05:30)', and ARN 'arn:aws:iam::654654552681:group/Network-L1-Team'. The 'Users' tab shows one user named 'Network-L1-User1'.

The screenshot shows the AWS IAM User Groups page for the 'Network-L1-Team' group. The 'Permissions' tab is selected, displaying two attached policies: 'AmazonVPCReadOnlyAccess' and 'AWSNetworkManagerReadOnlyAccess', both of which are AWS managed policies.

## Result:

Thus, a user group with given name and permissions, and a user with given name has been created and the created user has been added to the user group successfully.

## **EX-NO: 4**

**Create a VPC with two public subnets as mentioned below, and also create an Internet Gateway and attach it with the created VPC and also add internet route in the Route Table**

**The VPC 'Name' tag must be "public-vpc" in the region "N. Virginia"**

**The VPC CIDR range must be "10.10.0.0/16".**

**The VPC "public-vpc" should have a public subnet with the CIDR "10.10.1.0/24" in**

**the "us-east-1a" zone**

**The VPC "public-vpc" should have another public subnet with the CIDR "10.10.2.0/24" in the "us-east-1b" zone**

**Internet Gateway should be attached to the VPC "public-vpc"**

**The "public-vpc" route table should have Internet route entry**

### **Aim:**

To create a VPC named "public-vpc" in the region "N. Virginia" with given CIDR range and other given requirements.

### **Description:**

#### **VPC Creation And its Connection:**

- a. Login to the AWS console.
- b. Search for VPC service in search bar.
- c. Click 'Create VPC' in create it with the name "public-vpc"
- d. Select us-east-1 (N.Virginia) AWS Region.
- e. Choose the CIDR Range for VPC as 10.10.0.0/16.
- f. Create Public-subnet1 with cidr "10.10.1.0/24" in the "us-east-1a" and Public-subnet2 with cidr "10.10.2.0/24" in the "us-east-1b".
- g. Make sure that route table is created for each subnet
- h. Choose the number of availability zones to have NAT Gateways
- i. Create a VPC endpoints and Enable DNS option.
- j. Finally click "Create VPC" and VPC is created
- k. Attach Internet Gateway is connected with created VPC using route table

## Output :

The screenshot shows the AWS Subnets page. On the left, there's a navigation sidebar with options like VPC dashboard, EC2 Global View, Filter by VPC (with a dropdown for 'Select a VPC'), Virtual private cloud (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints, Endpoint services, NAT gateways, Peering connections), and Security (Network ACLs, Security groups). The main area is titled 'Subnets (2) info' and contains a table with two rows:

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR
public-subnet2	subnet-0f4a12a1934cf10bb	Available	vpc-079b039f9a3dd053c   publ...	10.10.2.0/24	-
public-subnet1	subnet-0754b25ce0272cc08	Available	vpc-079b039f9a3dd053c   publ...	10.10.1.0/24	-

The screenshot shows the AWS VPCs page. The left sidebar is identical to the previous screenshot. The main area is titled 'Your VPCs (1/1) info' and shows a table with one row:

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP option set
public-vpc	vpc-079b039f9a3dd053c	Available	10.10.0.0/16	-	dopt-000322e722654f...

Below this, under 'vpc-079b039f9a3dd053c / public-vpc', is a 'Resource map' section. It shows a network diagram with four components: 'VPC' (public-vpc), 'Subnets (2)' (public-subnet1, public-subnet2), 'Route tables (2)' (public-route), and 'Network connections (1)' (public-internet-gateway). Arrows indicate connections between public-subnet1 and public-route, and between public-subnet2 and public-route. The public-internet-gateway is also connected to public-route.

## Result :

Thus VPC named "public-vpc" is created in the region "N. Virginia" with given CIDR range and other given requirements.

## **EX:NO -5**

**Configure AWS CloudFront to deliver your website content from an S3 bucket in the ap-northeast-1 region to users in All edge Locations**

**Create a new S3 bucket in the region of "Tokyo".**

**Upload a file in the name of 'homepage.html'.**

**Make the object 'index.html' file accessible to everyone(publicly)**

**Create a CloudFront distribution with the created S3 Bucket as its Origin Domain.**

### **Aim:**

To Configure AWS CloudFront to deliver your website content from an S3 bucket in the ap-northeast-1 region to users in All edge Locations.

### **Description:**

1. Login to your AWS console and search for S3 service in search bar.
2. Click 'Create bucket' and select 'ap-northeast-1' AWS region.
3. Give a unique name to your bucket.
4. Enable ACL in object ownership and select object writer and object owner.
5. Enable public access to the bucket.
6. Enable Bucket Versioning, if you want.
7. Finally, Click 'Create Bucket' to create your S3 bucket.
8. Select the bucket you created and upload a HTML file named 'homepage' to the bucket.
9. Before uploading the file, grant public read access to the file.
10. Search for 'CloudFront' int search bar and click 'Create distribution'.
11. Choose your S3 bucket as origin domain.
12. In WAF, do not enable security protections.
13. Finally, click 'Create Distribution' to create cloud distribution.

## Output:

The screenshot shows the AWS S3 console interface. On the left, the navigation pane is visible with options like Buckets, Access Grants, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, IAM Access Analyzer for S3, Block Public Access settings for this account, Storage Lens, Dashboards, Storage Lens groups, AWS Organizations settings, Feature spotlight, and AWS Marketplace for S3. The main content area displays an 'Account snapshot' with a link to 'View Storage Lens dashboard'. Below it, there are tabs for 'General purpose buckets' and 'Directory buckets', with 'General purpose buckets (1)' selected. A table lists one bucket: Name (exercixe5bucket), AWS Region (Asia Pacific (Tokyo) ap-northeast-1), Access (Objects can be public), and Creation date (March 4, 2024, 18:39:26 (UTC+05:30)). Action buttons for Copy ARN, Empty, Delete, and Create bucket are available at the top of the table.

The screenshot shows the AWS CloudFront console interface. The navigation pane includes CloudFront, Distributions, and other options. The main content area shows the 'Distributions (1)' section with an 'Info' button. A table lists one distribution: ID (E15AULHZPRXFCK), Description (-), Type (Production), Domain (dn1ubgos...), Alternative ( - ), Origins (exercixe5bucket), Status (Enabled), and Last modified (Enabled). Action buttons for Enable, Disable, Delete, and Create distribution are present at the top of the table.

## Result:

Thus, a file is uploaded to a newly created S3 bucket in Tokyo region with public access and a cloud distribution has been created with the created S3 bucket as its origin domain.

## **EX:NO - 6**

**Create a public ECR with the name 'docker' in the region 'us-east-1' and push the docker image to the repository.**

**Create an EC2 instance with the name tag "dockerinstance1" in the "us-east-1" region**

**EC2 instance AMI should be "Amazon Linux 2023".**

**Allow HTTP traffic from the instance security group**

**Install docker in the "dockerinstance1"**

**Write a dockerfile with httpd as its base image and build the file**

**Create public repository in ECR with the name "docker"**

**Push the image to the created repository**

**The image should have 'latest' as its tag**

### **Aim:**

To create a Ec2 instance with Amazon Linux AMI and to enable HTTP traffic and to install docker in the instance and to write a dockerfile with httpd as a base image and to create a public repo in ECR named as Docker and to push the image to it and to view the image with latest tag.

### **Description:**

#### **Instance Creation:**

1. Login to the AWS console.
2. Search for EC2 service in search bar and click on “Launch instance”.
3. Select eu-east-1 (Paris) as AWS Region.
4. Give instance name and key pair name as “dockerinstance1”.
5. Choose Amazon Linux as AMI.
6. Enable SSH and HTTP requests in Networking and security groups.
7. Finally Click ‘Launch Instance’ and your Instance is created.

#### **Docker Installation with ECR:**

1. Install docker in terminal with basic linux commands.
2. Write a dockerfile with httpd as its base image.
3. Create a public repository in ECR with the name “docker”
4. Push the created docker image in the “docker” repository
5. Make sure the Image have a tag called “latest” in it.

## Output :

The screenshot shows a terminal window in AWS CloudShell. The user has run the command `yum install docker`. The output details the installation of Docker and its dependencies from the Amazon Linux repository. It includes a transaction summary showing 10 packages installed, totaling 309M, and asks if the user wants to proceed.

```
[root@ip-172-31-30-71 ec2-user]# yum install docker
Last metadata expiration check: 0:00:29 ago on Sat Mar  2 13:52:09 2024.
Dependencies resolved.

Package          Architecture Version      Repository  Size
Installing:
  docker          x86_64    24.0.5-1.amzn2023.0.3   amazonlinux 42 M
Installing dependencies:
  containerd      x86_64    1.7.11-1.amzn2023.0.1   amazonlinux 35 M
  iptables-libs   x86_64    1.8.8-3.amzn2023.0.2   amazonlinux 401 k
  iptables-nft    x86_64    1.8.8-3.amzn2023.0.2   amazonlinux 183 k
  libcgroup       x86_64    3.0.1.amzn2023.0.1    amazonlinux 75 k
  libnetfilter_conntrack x86_64  1.0.8-2.amzn2023.0.2   amazonlinux 58 k
  libnftnl        x86_64    1.0.1-19.amzn2023.0.2  amazonlinux 30 k
  libxft          x86_64    2.5.1.amzn2023.0.3   amazonlinux 84 k
  pigz            x86_64    1.1.11-1.amzn2023.0.1  amazonlinux 83 k
  runc            x86_64    1.0.1-19.amzn2023.0.1  amazonlinux 3.0 M

Transaction Summary

Install 10 Packages

Total download size: 81 M
Installed size: 309 M
Is this ok [y/N]: y
Downloading Packages:

i-06b5975408ee43d37 (dockerinstance1)
PublicIPs: 54.144.217.8 PrivateIPs: 172.31.30.71
```

The screenshot shows the Amazon ECR console. A green success message indicates that a public repository named `docker` has been successfully created in the public registry. The main interface displays the `Public repositories` section, which lists the single repository `docker` with its URI and creation date.

Created public repository  
docker has been successfully created in public registry

Amazon ECR > Public Registry > Repositories

Public repositories

Repository name	URI	Created at
docker	public.ecr.aws/u5h5s5l7/docker	March 02, 2024, 19:27:44 (UTC+05:5)

The screenshot shows the AWS ECR interface. The left sidebar has sections for Private registry (Repositories, Settings) and Public registry (Repositories, Images, Gallery detail, Permissions, Repository tags, Settings). Under Public registry, 'Images' is selected. The main area shows a repository named 'docker'. A table lists one image entry:

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
latest	Image	March 02, 2024, 19:33:16 (UTC+05:5)	64.53	<a href="#">Copy URI</a>	<a href="#">sha256:524db63363a02f...</a>

## Result :

Thus, EC2 instance with Amazon Linux AMI is created and HTTP traffic is enabled and docker is installed and written with httpd as a base image and the image with latest tag is pushed into a public repository called “docker”.

## **EX:NO - 7**

**You are working on a project that involves developing and deploying a web application using containers on a cloud-based platform. You have been asked to create and manage the containers and container images for this project. You are using Docker as your containerization technology.**

**Evaluate the process for creating and removing containers and container images in Docker.**

### **Aim :**

To evaluate the process for creating and removing containers and container images in docker.

### **Description :**

Docker is an open-source project that uses several resource-isolation features of the Linux kernel to sandbox an application, its dependencies, configuration files, and interfaces inside of an atomic unit called a container. This allows a container to run on any host with the appropriate kernel components, while shielding the application from behavioral inconsistencies due to variances in software installed on the host

There are many ways in which using containers on AWS can benefit your organization. Docker has been widely employed in use cases such as distributed applications, batch jobs, continuous deployment pipelines, and etc. The use cases for Docker continue to grow in areas like distributed data processing, machine learning, streaming media delivery and genomics

## Process :

### 1. Creating a EC2 instance with Ubuntu AMI named as “Dockerinstance”

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links like EC2 Dashboard, EC2 Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, and Elastic Block Store. The main area displays the 'Instance summary for i-08ebf87dc1ec12c83 (Docker instance)'. It includes fields for Instance ID, Public IPv4 address, Private IPv4 addresses, IPv6 address, Instance state, Hostname type, Private IP DNS name (IPv4 only), Instance type, Auto-assigned IP address, VPC ID, and AWS Compute Optimizer finding. Below this is a 'Details' tab with sub-sections for Instance details, Platform, AMIs, and Launch time.

### 2. Logging in and installing Docker in the instance

```
root@ip-172-31-1-69:/home/ubuntu# apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 34 not upgraded.
Need to get 69.8 MB of archives.
After this operation, 267 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-lubuntu3 [34.4 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 runc amd64 1.1.7~Ubuntu1~22.04.2 [4267 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 containerd amd64 1.7.2~Ubuntu1~22.04.1 [36.0 MB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dns-root-data all 2023112702~ubuntu22.04.1 [5136 B]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dnsmasq-base amd64 2.90~Ubuntu22.04.1 [374 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 docker.io amd64 24.0.5~Ubuntu1~22.04.1 [28.9 MB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 ubuntu-fan all 0.12.16 [35.2 kB]
Fetched 69.8 MB in 2s (38.6 MB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
i-08ebf87dc1ec12c83 (Docker instance)
PublicIPs: 3.110.108.235 PrivateIPs: 172.31.1.69
```

### 3. Building and Running the image in docker

```
running kernel seems to be up-to-date.
No services need to be restarted.
No containers need to be restarted.
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-4-54:~$ sudo systemctl start docker
ubuntu@ip-172-31-4-54:~$ sudo systemctl enable docker
ubuntu@ip-172-31-4-54:~$ sudo docker pull httpd:latest
Latest: Pulling from library/httpd
e1caa4ceb9d2: Pull complete
97b0fe460fd9: Pull complete
f4fb700ef54: Pull complete
9c6bd3e3b523: Pull complete
e9304da947c5: Pull complete
b60d4b6b6b268: Pull complete
Digest: sha256:104f07de17ee186c8f37b9f561e04fbe4cf080d78c6e5f3802fd08fd118c3da
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
ubuntu@ip-172-31-4-54:~$ sudo docker run -d -p 80:80 --name Apache-server httpd:latest
10a5e3e6ac474ee283cd58a88348501f5680blcc06829ceb7ceb8e699d29fc62
ubuntu@ip-172-31-4-54:~$ [ ]
```

i-0314f6877404001db (Docker instance)  
Public IPs: 65.2.83.26 Private IPs: 172.31.4.54

### 4. Stopping and Removing the Running Container

```
Latest: Pulling from library/httpd
e1caa4ceb9d2: Pull complete
97b0fe460fd9: Pull complete
f4fb700ef54: Pull complete
9c6bd3e3b523: Pull complete
e9304da947c5: Pull complete
b60d4b6b6b268: Pull complete
Digest: sha256:104f07de17ee186c8f37b9f561e04fbe4cf080d78c6e5f3802fd08fd118c3da
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
ubuntu@ip-172-31-4-54:~$ sudo docker run -d -p 80:80 --name Apache-server httpd:latest
10a5e3e6ac474ee283cd58a88348501f5680blcc06829ceb7ceb8e699d29fc62
ubuntu@ip-172-31-4-54:~$ sudo docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
10a5e3e6ac47 httpd:latest "httpd-foreground" 2 minutes ago Up 2 minutes 0.0.0.0:80->80/tcp, ::80->80/tcp Apache-server
ubuntu@ip-172-31-4-54:~$ sudo docker rm 10a5e3e6ac47
Error response from daemon: You cannot remove a running container 10a5e3e6ac474ee283cd58a88348501f5680blcc06829ceb7ceb8e699d29fc62. Stop the container before attempting removal or force remove
ubuntu@ip-172-31-4-54:~$ sudo docker stop 10a5e3e6ac47
10a5e3e6ac47
ubuntu@ip-172-31-4-54:~$ sudo docker rm 10a5e3e6ac47
10a5e3e6ac47
ubuntu@ip-172-31-4-54:~$ sudo docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
ubuntu@ip-172-31-4-54:~$ [ ]
```

i-0314f6877404001db (Docker instance)  
Public IPs: 65.2.83.26 Private IPs: 172.31.4.54

## Result :

Thus, the process for creating and removing containers and container images in docker is evaluated.

## **EX: NO -8**

**Create a three VPC in the region ca-central-1 and enable peer to peer connections with the following requirements.**

**Requirements:**

**Name should be vpc-1, vpc-2 and vpc-3 respectively.**

**CIDR ranges should be 10.0.0.0/16 ,11.0.0.0/16 and 12.0.0.0/16 for the above created VPCs.**

**Create a one public subnet in each.**

**Connect vpc-1, vpc-2 as first connection.**

**Connect vpc-2, vpc-3 as second connection.**

**Connect vpc-3, vpc-1 as third connection.**

**Aim:**

To Create a three VPC in the region ca-central-1 and enable peer to peer connections with the given requirements

**Description:**

**VPC Creation:**

1. Login to the AWS console.
2. Click ‘Create VPC’ in VPC service for three specific VPC’s.
3. Select ca-central-1 as AWS Region.
4. Give its name as “vpc-1”, “vpc-2”, “vpc-3” respectively.
5. Choose the CIDR Range as 10.0.0.0/16 ,11.0.0.0/16 and 12.0.0.0/16 respectively.
6. Create 1 subnet for each VPC.
7. Finally click “Create VPC” and VPC is created.

**VPC Peering Connection :**

1. Create a VPC peering connection
2. Connect vpc-1, vpc-2 as first connection.
3. Connect vpc-2, vpc-3 as second connection.
4. Connect vpc-3, vpc-1 as third connection.
5. Finally Create and establish the peering between VPC’s.

## Output :

Your VPCs (4) <a href="#">Info</a>						
<input type="checkbox"/>	Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCI
<input type="checkbox"/>	-	<a href="#">vpc-0117a7a95b57b31f8</a>	<span>Available</span>	172.31.0.0/16	-	<a href="#">dopt-</a>
<input type="checkbox"/>	vpc-1	<a href="#">vpc-0e6649d245c73ce8b</a>	<span>Available</span>	10.0.0.0/16	-	<a href="#">dopt-</a>
<input type="checkbox"/>	vpc-2	<a href="#">vpc-0d53ed073716cab03</a>	<span>Available</span>	11.0.0.0/16	-	<a href="#">dopt-</a>
<input type="checkbox"/>	vpe-3	<a href="#">vpc-0cb0d5070989d3278</a>	<span>Available</span>	12.0.0.0/16	-	<a href="#">dopt-</a>

Peering connections (3) <a href="#">Info</a>				
<input type="checkbox"/> Find resources by attribute or tag				
Name	Peering connection ID	Status	Requester VPC	Acceptor VPC
peer vpc 3 to vpc 1	<a href="#">pcx-09dd6af2192385e70</a>	<span>Active</span>	<a href="#">vpc-0cb0d5070989d3278 / vpe-3</a>	<a href="#">vpc-0e6649d245c73ce8b / vpc-1</a>
peer vpc1 to vpc2	<a href="#">pcx-0c3bc6baf14670c80</a>	<span>Active</span>	<a href="#">vpc-0e6649d245c73ce8b / vpc-1</a>	<a href="#">vpc-0d53ed073716cab03 / vpc-2</a>
peer vpc2 to vpc3	<a href="#">pcx-04ed5681a6b414243</a>	<span>Active</span>	<a href="#">vpc-0d53ed073716cab03 / vpc-2</a>	<a href="#">vpc-0cb0d5070989d3278 / vpe-3</a>

## Result :

Three VPC in the region ca-central-1 is created and enabled for peer to peer connections with the given requirements

## **EX:NO - 9**

**Create a docker container with the following requirements.**

**Requirements:**

**Give tag name and key name as “ubuntu-docker”.**

**EC2 instance AMI should be “ubuntu”.**

**Allow SSH for SSH client for instance connection.**

**Allow HTTP traffic from the internet for reaching website requests.**

**Install docker.**

**Pull docker Apache image from docker hub.**

**Docker container name should be “Apache-server” and run docker in 8080 port, forward it to 80;**

**Aim:**

To create a Docker container with the given requirements.

**Description:**

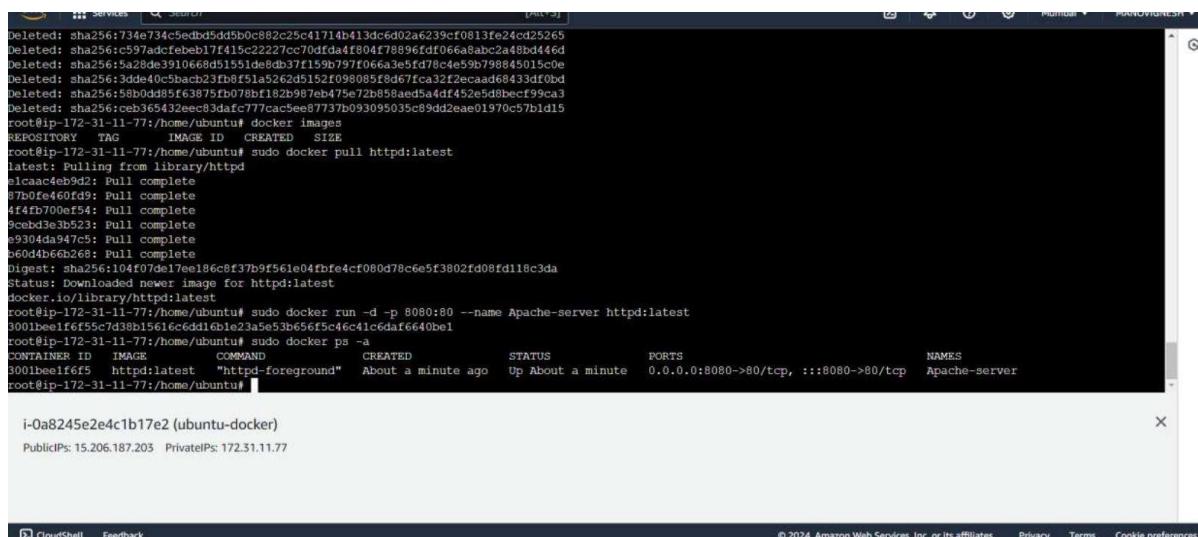
**Instance Creation:**

1. Login to the AWS console.
2. Search for EC2 service in search bar and click on “Launch instance”.
3. Select eu-east-1 (Paris) as AWS Region.
4. Give instance name and key pair name as “ubuntudocker”.
5. Choose ubuntu as AMI.
6. Enable SSH and HTTP requests in Networking and security groups.
7. Finally Click ‘Launch Instance’ and your Instance is created.

**Docker Installation:**

1. Install docker in terminal with basic linux commands.
2. Write a dockerfile with apache as its base image.
3. Pull the Apache server from dockerhub
4. Docker container name should be “Apache-server” and run docker in 8080 port and forward it to 80.

## Output :



```
Deleted: sha256:734e734c5edb5dd5b0c882c25c41714b413dc6d02a6239cf0813fe24cd25265
Deleted: sha256:c597adfebeb17f415c2227cc70dfda4fb0478896fd066a8ab2a4fb0d446d
Deleted: sha256:5a28de3910668d5151de8db37f159b797f066a3e5fd78ce4e59b798845015c0e
Deleted: sha256:3dd40c5bach23fb8f51a5262d5152f098085f8d67fca32f2ecaad66433df0bd
Deleted: sha256:58bd0d85f63875fb78bf182b987eb475e72b858aed5a4d4f452e5d8becf99ca3
Deleted: sha256:ceb365432ee83dacf77cac5ee87737b093095035cb9dd2ea01970c57b1d15
root@ip-172-31-11-77:/home/ubuntu# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
root@ip-172-31-11-77:/home/ubuntu# sudo docker pull httpd:latest
latest: Pulling from library/httpd
elcaac4eb9d2: Pull complete
97b0fe460fd9: Pull complete
4f4fb700ef54: Pull complete
9cebd3e3b523: Pull complete
e9304da947c5: Pull complete
b60d4b6b268: Pull complete
Digest: sha256:104f07de17ee186c8f37b9f561e04fbfe4cf080d78c6e5f3802fd08fd118c3da
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
root@ip-172-31-11-77:/home/ubuntu# sudo docker run -d -p 8080:80 --name Apache-server httpd:latest
3001beef1f6f55c7d38b15616c6d16b1e23a5e53b656f5c46c41c6daf6640be1
root@ip-172-31-11-77:/home/ubuntu# sudo docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
3001beef1f6f5 httpd:latest "httpd-foreground" About a minute ago Up About a minute 0.0.0.0:8080->80/tcp, :::8080->80/tcp Apache-server
root@ip-172-31-11-77:/home/ubuntu#
```

i-0a8245e2e4c1b17e2 (ubuntu-docker)  
PublicIPs: 15.206.187.203 PrivateIPs: 172.31.11.77

## Result :

Thus, the docker is created with the given requirements.

## **EX:NO - 10**

**Create a docker container with the following requirements.**

**Requirements:**

**Tag name and key name: "ubuntu-docker".**

**EC2 instance AMI: Ubuntu.**

**Allow SSH for SSH client for instance connection.**

**Allow HTTPS traffic from the internet for secure connections.**

**Install Docker.**

**Build Docker image for Apache2 server with index.html using Dockerfile.**

**Run Docker container from the built image.**

**Aim:**

To create a Docker container with the given requirements.

**Description:**

**Instance Creation:**

1. Login to the AWS console.
2. Search for EC2 service in search bar and click on “Launch instance”.
3. Select eu-east-1 (Paris) as AWS Region.
4. Give instance name and key pair name as “ubuntu-docker”.
5. Choose ubuntu as AMI.
6. Enable SSH and HTTP requests in Networking and security groups.
7. Finally Click ‘Launch Instance’ and your Instance is created.

**Docker Installation:**

1. Install docker in terminal with basic linux commands.
2. Write a dockerfile with apache as its base image.
3. Build docker image for Apache server using Dockerfile.
4. Run the docker container with built image.

## Output :

The screenshot shows the AWS EC2 Instances page. The instance summary for i-02814ca74c13c06eb (ubuntu-docker) is displayed. Key details include:

- Instance ID: i-02814ca74c13c06eb (ubuntu-docker)
- Public IPv4 address: 43.204.214.156
- Private IP4 address: ip-172-31-7-61.ap-south-1.compute.internal
- Instance state: Running
- VPC ID: vpc-09121e5255a573338
- Subnet ID: subnet-0322dd6acc45e858d
- Instance type: t2.micro
- AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations.

```
No user sessions are running outdated binaries.  
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
root@ip-172-31-7-61:/opt/dockerimage# sudo systemctl start apache2  
root@ip-172-31-7-61:/opt/dockerimage# sudo systemctl enable apache2  
synchronizing state of apache2.service with SysV service script with /lib/systemd/systemd-sysv-install.  
executing: /lib/systemd/systemd-sysv-install enable apache2  
root@ip-172-31-7-61:/opt/dockerimage# sudo systemctl status apache2  
● apache2.service - The Apache HTTP Server  
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)  
   Active: active (running) since Sat 2024-03-02 18:14:54 UTC; 28s ago  
     Docs: https://httpd.apache.org/docs/2.4/  
     Main PID: 7161 (apache2)  
        Tasks: 55 (limit: 1121)  
       Memory: 5.2M  
          CPU: 32ms  
        CGroup: /system.slice/apache2.service  
                └─7161 /usr/sbin/apache2 -k start  
                  ├─7163 /usr/sbin/apache2 -k start  
                  └─7164 /usr/sbin/apache2 -k start  
  
Mar 02 18:14:54 ip-172-31-7-61 systemd[1]: Starting The Apache HTTP Server...  
Mar 02 18:14:54 ip-172-31-7-61 systemd[1]: Started The Apache HTTP Server.  
root@ip-172-31-7-61:/opt/dockerimage# sudo docker run -d -p 443:443 --name apache2-container apache2-server:  
  
i-02814ca74c13c06eb (ubuntu-docker)  
PublicIPs: 43.204.214.156 PrivateIPs: 172.31.7.61
```

## Result :

Thus, the docker is created with the given requirements.

## **EX: NO -11**

**Create a EC2 instance (ap-south-1) region, attach the volume and take snapshot (EBS) with the following requirements.**

**Give the name tag of both ec2 instance and keypair as “ap-ec2”.**

**Ec2 instance AMI should be Amazon Linux 2.**

**Create a EBS volume of 5 Gb and attach the of the ap-ec2.**

**Take snapshot of the EBS volume attached.**

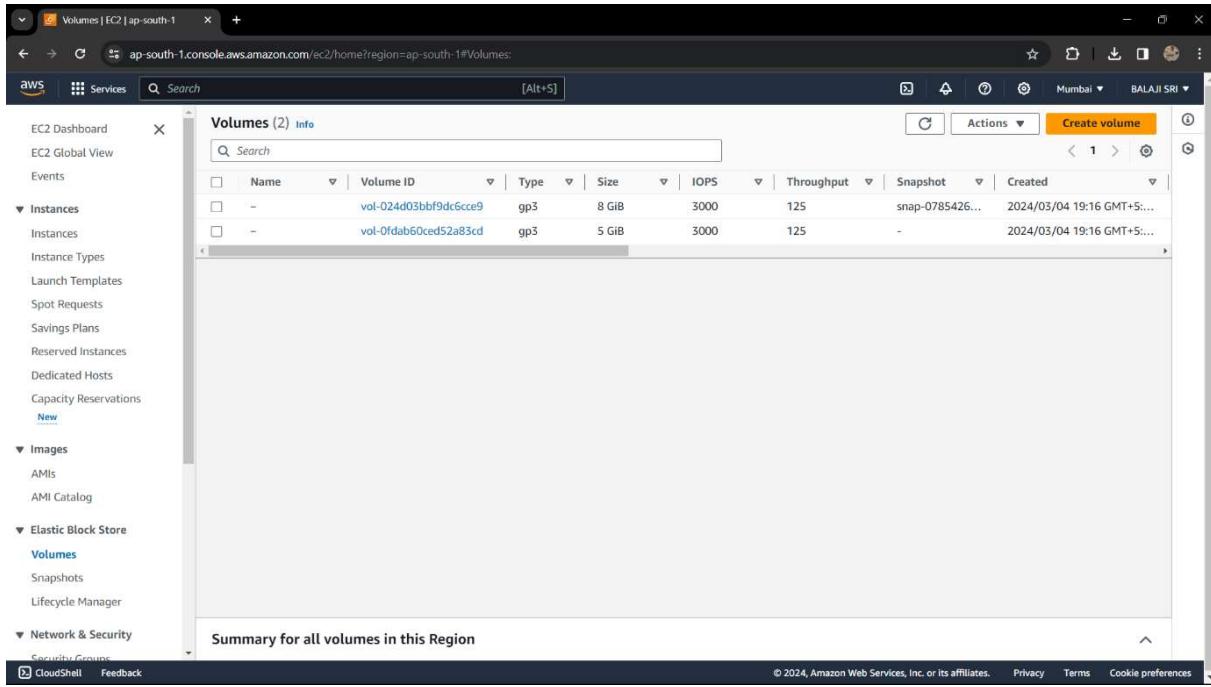
### **Aim:**

To create an EC2 instance int ap-south-1 region and to attach the volume and take snapshot with the given requirements.

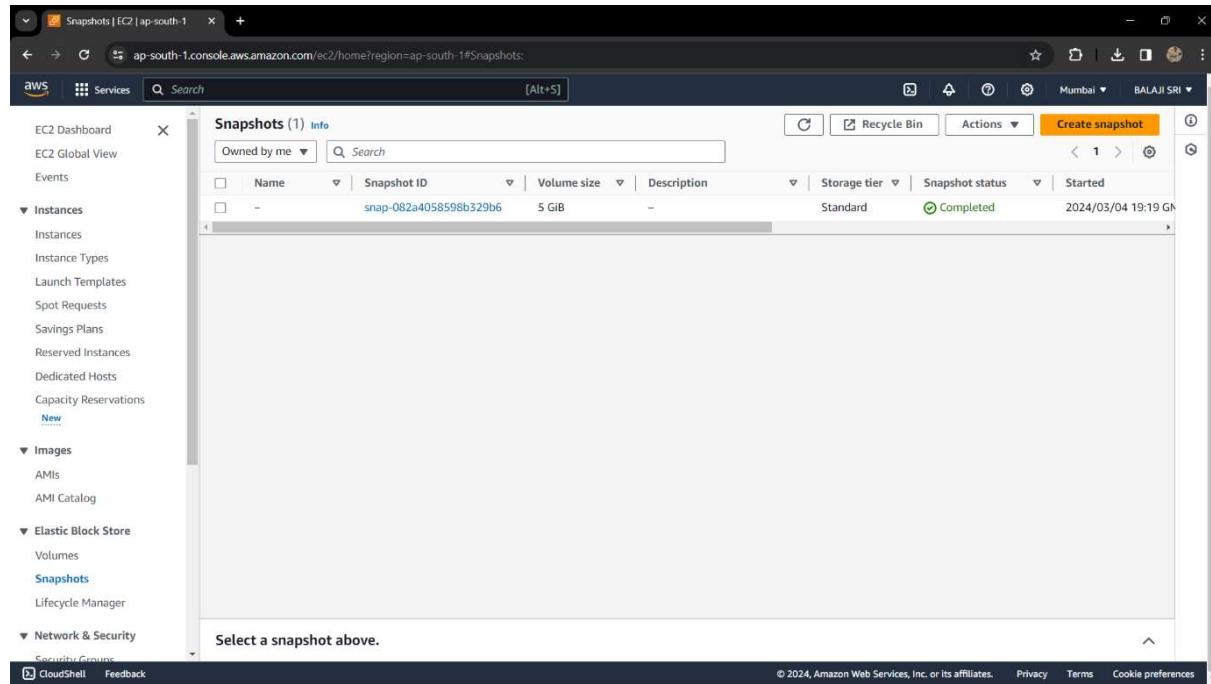
### **Description:**

1. Login to your AWS console and Search for EC2 service in search bar.
2. Change your AWS region to ap-south-1 (Mumbai).
3. Click ‘Launch Instances’ to create a new instance.
4. Give name of EC2 instance as ‘ap-ec2’.
5. Select ‘Amazon Linux 2023’ as AMI for your EC2 instance.
6. Select ‘t2.micro’ as your instance type. (only instance type available for free tier account).
7. Create a keypair with the name of ‘ap-ec2’.
8. Allow SSH, HTTP traffic in network to your EC2 instance.
9. In configure storage, Add new EBS volume of size 5 GB.
10. Finally, Click ‘Launch Instance’ to create your instance.
11. In side bar, you can find ‘Snapshots’. Click on that and Create Snapshot.
12. Select resource type as ‘Volume’.
13. Select your EBS volume ID and thus finally create snapshot.

## Output:



The screenshot shows the AWS EC2 Volumes page. The left sidebar navigation includes EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, New, Images (AMIs, AMI Catalog), Elastic Block Store (Volumes selected, Snapshots, Lifecycle Manager), Network & Security (Security Groups), CloudShell, and Feedback. The main content area displays a table titled "Volumes (2) Info" with columns: Name, Volume ID, Type, Size, IOPS, Throughput, Snapshot, and Created. Two volumes are listed: "vol-024d03bbf9dc6cce9" (gp3, 8 GiB, 3000 IOPS, 125 throughput, snap-0785426..., 2024/03/04 19:16 GMT+5...) and "vol-0fdab60ced52a83cd" (gp3, 5 GiB, 3000 IOPS, 125 throughput, -). A summary bar at the bottom states "Summary for all volumes in this Region".



The screenshot shows the AWS EC2 Snapshots page. The left sidebar navigation is identical to the previous screenshot. The main content area displays a table titled "Snapshots (1) Info" with columns: Name, Snapshot ID, Volume size, Description, Storage tier, Snapshot status, and Started. One snapshot is listed: "snap-082a4058598b329b6" (5 GiB, -, Standard, Completed, 2024/03/04 19:19 GMT+5). A message at the bottom says "Select a snapshot above."

## Result:

Thus an EC2 instance has been created in Mumbai region with an Elastic Block Store of size 5GB and the snapshot for that EBS volume has been successfully taken.

## **EX:NO -12**

**Create EC2 windows instance in eu-west-3 region with given requirements.**

**Requirements:**

**Give the name tag of both ec2 instance and key pair as “windows-server”.**

**Enable http and RDP in security group.**

**Login into window server using RDP.**

**Enable IIS and show the IIS page output with the help of public IP.**

**Aim:**

To create an ec2 instance in eu-west-3 region and enabling IIS to show the IIS page output with help of public IP.

**Description:**

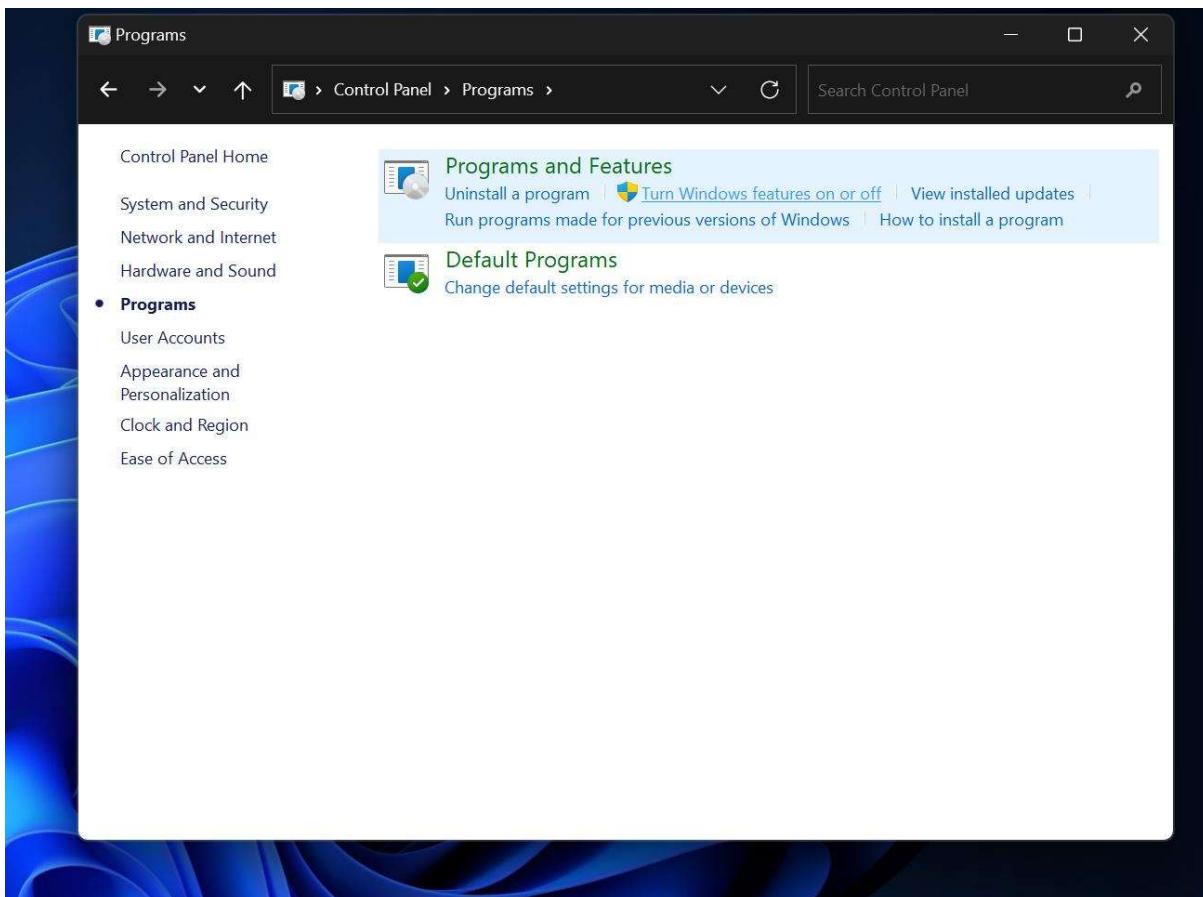
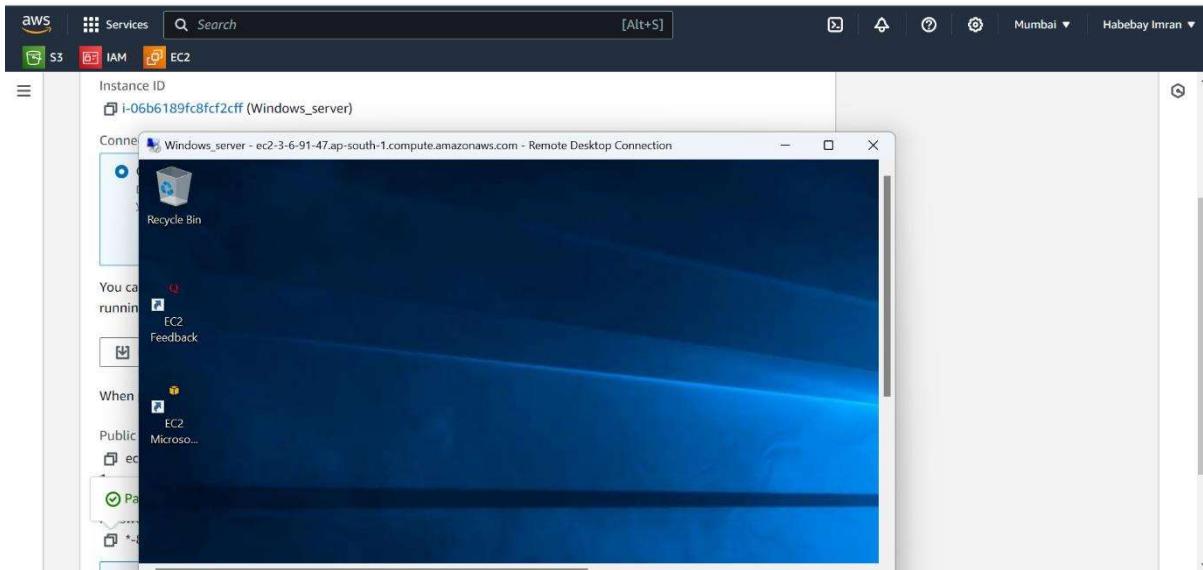
**Instance Creation:**

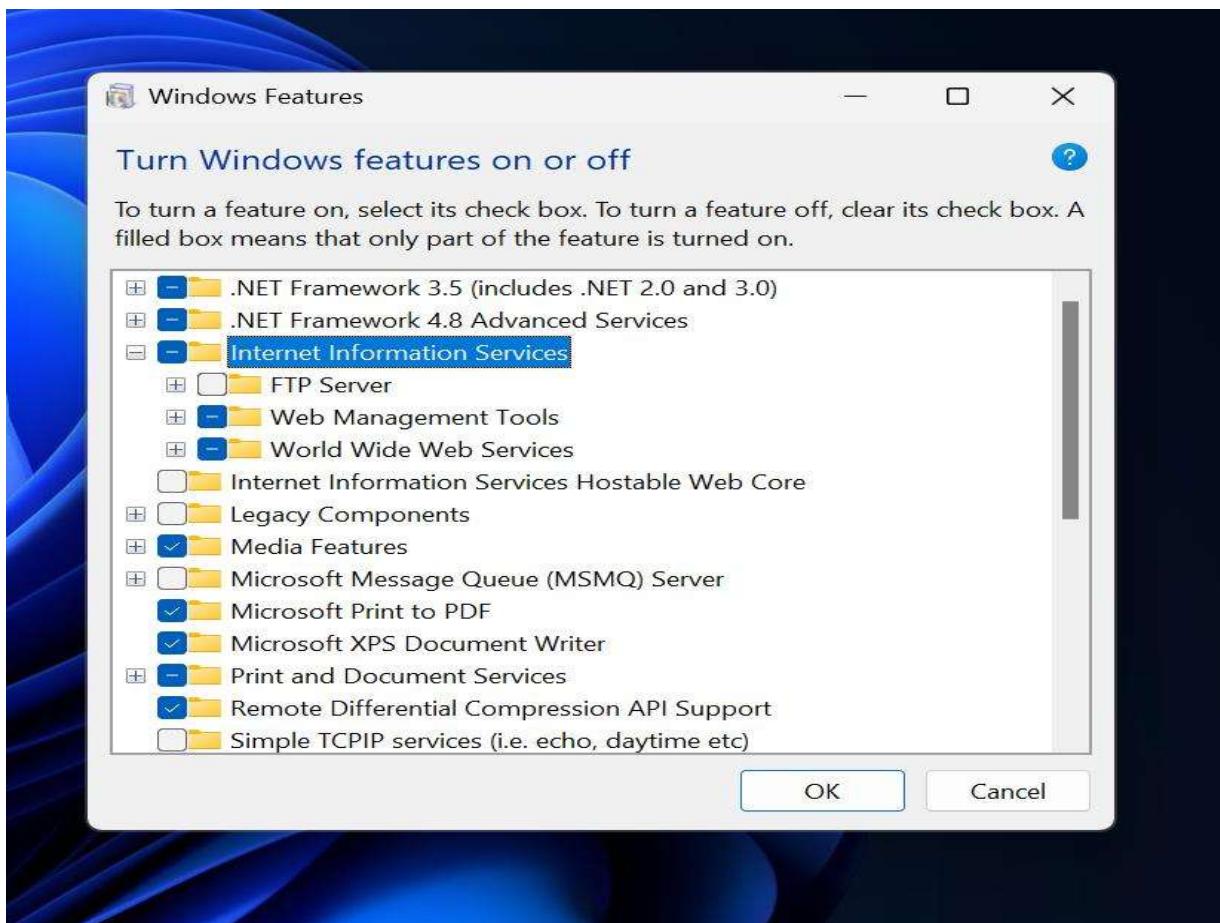
1. Login to the AWS console.
2. Search for EC2 service in search bar.
3. In Instance section, Click ‘Launch Instance’.
4. Select eu-west-3 (Paris) as AWS Region.
5. Give instance name as “windows-server”.
6. Choose Windows as AMI.
7. Choose t2-micro or t3-micro as instance type.
8. Enable RDP and HTTP requests in Networking and security groups.
9. Finally Click ‘Launch Instance’ and your Instance is created.

**Instance Login and IIS Access:**

1. Connect the instance using RDP client by downloading file and decrypting the password
2. Enter the password to open instance window
3. In Instance Window, Go to Programs In Control Panel.
4. Select “Turn Windows Features On or Off”
5. Search For IIM and Enable it.

## Output:





## Result:

Thus an EC2 instance is created in eu-west-3 region and logged in using instance and IIS is enabled.

## **EX:NO -13**

**Create an EC2 Amazon Linux instance in (ap-northeast-1) Tokyo and launch another instance using the backup AMI of Japanese server with the following requirements:**

**Requirements:**

**Give the name tag of both ec2 instance and keypair as “japanese-server”.**

**Enable ssh and http in security group.**

**Create an image backup from running server with the name (server-backup).**

**Launch another server using the created AMI backup.**

**Aim:**

To create an ec2 instance in ap-northeast-1 region and launching another instance using the backup AMI of first instance.

**Description:**

**Instance Creation With Amazon Linux:**

1. Login to the AWS console.
2. Search for EC2 service in search bar.
3. In Instance section, Click ‘Launch Instance’.
4. Select ap-northeast-1 (Tokyo) as AWS Region.
5. Give instance name as “japanese-server”.
6. Choose Amazon as AMI.
7. Choose t2-micro or t3-micro as instance type.
8. Create a key pair in the same name as instance
9. Enable SSH and HTTP requests in Networking and security groups.
10. Finally Click ‘Launch Instance’ and your Instance is created.

**Instance Creation With AMI Backup:**

1. Click on Actions in the created instance and create a image in the name “server-backup”
2. Launch new instance with same name.
3. Choose the Backup AMI for AMI
4. Apply the same properties as first instance.
5. Finally Click ‘Launch Instance’ and your Instance is created.

## Output:

The screenshot shows the AWS EC2 Instances page. The left sidebar navigation includes: AWS logo, Services (S3, IAM, EC2, VPC, CloudFront, Elastic Container Registry), Search bar, and Mumbai/Habebay Imran account dropdown. Under the 'Instances' section, there are links for Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs, AMI Catalog), and Elastic Block Store (Volumes, Snapshots, Lifecycle Manager). The main content area displays 'Instances (1/1) Info'. A table lists one instance: 'japanese-server' (i-02d3802a8c09ef653). The instance is 'Running' (t2.micro), has 2/2 checks passed, and is in 'ap-south-1a' availability zone with a public IPv4 DNS of 'ec2-13-201-38-70.a'. Below the table, the 'Instance: i-02d3802a8c09ef653 (japanese-server)' details page is shown with tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. The 'Details' tab shows various instance attributes like Public IPv4 address (13.201.38.70), Instance state (Running), and Instance type (t2.micro).

The screenshot shows the AWS AMIs page. The left sidebar navigation is identical to the previous screenshot. The main content area displays 'Amazon Machine Images (AMIs) (1/1) Info'. A table lists one AMI: 'server-backup' (ami-0b5ce43ac84365393). The AMI is 'Available' and is associated with the user account 992382625408. Below the table, the 'AMI ID: ami-0b5ce43ac84365393' details page is shown with tabs for Details, Permissions, Storage, and Tags. The 'Details' tab shows AMI attributes like AMI ID (ami-0b5ce43ac84365393), Owner account ID (992382625408), and Status (Available).

The screenshot shows the AWS EC2 Instances page. The left sidebar includes links for EC2 Dashboard, EC2 Global View, Events, Instances (selected), Instances Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, and Elastic Block Store. The main content area displays a table of instances. The first instance, with ID i-02d3802a8c09ef653, is listed as 'Running' with status check '2/2 checks passed'. The second instance, with ID i-069e56ff6ad359238, is also 'Running' with the same status check. Both instances are of type t2.micro and are located in ap-south-1a. Their public IPv4 DNS names are ec2-13-201-38-70.a and ec2-15-206-146-164 respectively. The second instance is currently selected, indicated by a checked checkbox in the header and highlighted rows.

## Result:

Thus an EC2 instance is created in ap-northeast-1 region and an another EC2 instance is created with the backup AMI of first instance

## **EX-NO: 14**

**Create a VPC infrastructure with the following requirements.**

**Requirements:**

**Choose “ap-south-1” as VPC region and VPC name should be “vpc-south”**

**Your VPC must have 65,536 IP with the starting IP of 10.0.0.0.**

**VPC should contain at least one private subnet in each zone.**

**Create routing table, each private subnet should have separate routing table.**

**And connect it with the internet using NAT gateway.**

**Aim:**

To create an VPC in ap-south-1 region with the given requirements

**Description:**

**VPC Creation And NAT Gateway Connection:**

- a. Login to the AWS console.
- b. Search for VPC service in search bar.
- c. Click ‘Create VPC’ in VPC service.
- d. Select ap-south-1 (Mumbai) AWS Region.
- e. Give “vpc-south” as its name.
- f. Choose the CIDR Range as 10.0.0.0/16 with 65,536 IP’s.
- g. Choose the number of availability zone to provision subnets.
- h. Choose the number of public and private subnets.
- i. Make sure that route table is created for each private subnet
- j. Choose the number of availability zones to have NAT Gateways
- k. Create a VPC endpoints and Enable DNS option.
- l. Finally click “Create VPC” and VPC is created
- m. Make sure that NAT Gateway is connected with created VPC.

## Output:

VPC ID: vpc-0e5e6a47c819034fb

State: Available

Subnets (6): ap-south-1a, vpc-south-subnet-public1-ap-south-1a, vpc-south-subnet-private1-ap-south-1a, vpc-south-subnet-private3-ap-south-1a, vpc-south-rb-public, rtb-05ba91210985ce6a4

Route tables (6): vpc-south-rb-private1-ap-south-1a, vpc-south-rb-private4-ap-south-1b, vpc-south-rb-private2-ap-south-1b, vpc-south-rb-public, rtb-05ba91210985ce6a4

Network connections (3): vpc-south-qw, vpc-south-nat-public1-ap-south-1a, vpc-south-vpce-s3

Name	NAT gateway ID	Connectivity...	State	State message	Primary public I...	Primary private I...	Primary network...	VPC
vpc-south-nat-public1...	nat-027e8aa9d977a5fe4	Public	Available	-	43.205.163.77	10.0.1.151	eni-082aff20793cd7...	vpc-0e5e6a47c819034fb

## Result:

Thus VPC is created in ap-south-1 region with a specified CIDR range and having specific route table for private subnets and connected to internet using NAT Gateway.

## **EX-NO: 15**

**Create a VPC in the region ap-northeast-1 with the following requirements.**

**Requirements:**

**Choose “ap-northeast-1” as VPC region and VPC name should be “vpc-northeast”.**

**Your VPC must have 12.0.0.0/14 CIDR range.**

**Your VPC must have 1 public subnets and 2 private subnets with the proper IP CIDR range.**

**Aim:**

To create an VPC in ap-northeast-1 region with the given requirements

**Description:**

- a. Login to the AWS console.
- b. Search for VPC service in search bar.
- c. Click ‘Create VPC’ in VPC service.
- d. Select ap-northeast-1 (Tokyo) AWS Region.
- e. Give “vpc-northeast” as its name.
- f. Choose the CIDR Range as 12.0.0.0/14.
- g. Choose the number of availability zone as 1 to provision subnets.
- h. Choose the number of public subnets as 1.
- i. Choose the number of private subnets as 2.
- j. Choose the number of availability zones to have NAT Gateways
- k. Create a VPC endpoints
- l. Enable DNS options
- m. Finally click “Create VPC” and VPC is created

## Output:

The screenshot shows the AWS VPC Details page for a VPC named 'vpc-northeast-vpc'. The 'Details' tab is selected, displaying information such as VPC ID (vpc-0037fe4786e648da6), State (Available), and CIDR (10.0.0.0/16). The 'Resource map' tab is also visible, showing the VPC structure with Subnets (3), Route tables (4), and Network connections (2).

VPC ID: vpc-0037fe4786e648da6

State: Available

Tenancy: Default

Default VPC: No

Network Address Usage metrics: Disabled

DNS resolution: Enabled

Main route table: rtb-0ea25582bfd68bd43

IPv6 pool: -

Owner ID: 992382625408

Subnets (3): ap-northeast-1a, ap-northeast-1b, ap-northeast-1c

Route tables (4): rtb-0ea25582bfd68bd43, rtb-0ea25582bfd68bd43, rtb-0ea25582bfd68bd43, rtb-0ea25582bfd68bd43

Network connections (2): vpc-northeast-igw, vpc-northeast-vpc-s5

## Result:

Thus VPC is created in ap-northeast-1 region with a specified CIDR range and having 1 public and 2 private subnets.