

Ex No : 1 Study cloud account environment and its features

AIM:

To study various cloud account features and analyse the usage of the user application development

Procedure

When studying the cloud account environment across multiple providers such as **AWS**, **IBM Cloud**, **Google Cloud Platform (GCP)**, and others like **Microsoft Azure**, you'll encounter several similarities in structure, but each provider has its unique features and tools tailored for different use cases. Here's a cumulative study of the cloud account environments and their features across AWS, IBM Cloud, GCP, and other major platforms.

1. Account Structure

AWS, **IBM Cloud**, **GCP**, and **Azure** all use a similar foundational structure, but with some differences in terminology:

- **Account:** Centralized container for managing resources, users, and services.
 - **AWS:** Account is the fundamental unit. Everything is associated with a unique AWS account.
 - **IBM Cloud:** Cloud accounts can be linked to organizations and regions for better resource management.
 - **GCP:** Uses **Google Cloud Projects** as the base unit for organizing and managing resources.
 - **Azure:** **Azure Subscriptions** are the fundamental units to manage services, which are grouped under **Azure Active Directory**.
- **Billing:** All platforms offer centralized billing within the cloud account. For instance, AWS uses **AWS Billing and Cost Management**, IBM Cloud has **IBM Cloud Billing**, and GCP uses **Google Cloud Billing**.
- **Regions and Availability Zones:** All cloud providers have geographically distributed data centers.
 - **AWS:** Offers multiple regions, each with multiple Availability Zones.
 - **IBM Cloud:** Provides multiple data center locations globally.
 - **GCP:** Divided into regions and zones similar to AWS and Azure, with specific offerings like **Global Load Balancing**.
 - **Azure:** Regions and Availability Zones are used to organize resources, with services often optimized for high availability across these zones.

2. Identity and Access Management (IAM)

- **AWS IAM:** Users, groups, and roles, with policies defining permissions. IAM also integrates with **AWS Organizations** for managing multiple accounts.
- **IBM Cloud IAM:** Users, service IDs, and roles for access control. It integrates well with **IBM Cloud Identity and Access Management** for granular security controls.
- **GCP IAM:** Allows management of access control for Google Cloud resources with policies defining what actions can be performed by which identity.
- **Azure IAM:** Uses **Azure Active Directory (AAD)** for user management and access control, allowing identity federation and managing user roles and permissions for resources.

3. Compute Services

- **AWS EC2:** Offers scalable virtual servers with flexible instance types. AWS also offers **Lambda** for serverless compute and **Elastic Beanstalk** for PaaS.
- **IBM Cloud Virtual Servers:** Provides both public and private virtual servers for workloads, along with **IBM Cloud Functions** for serverless compute.
- **GCP Compute Engine:** Offers scalable virtual machines (VMs) and **Google Cloud Functions** for serverless workloads.
- **Azure Virtual Machines:** Similar to EC2, allows users to provision scalable VMs. **Azure Functions** offers serverless computing.

4. Storage Services

- **AWS S3 (Simple Storage Service):** Object storage with virtually unlimited scalability. Includes features like versioning and lifecycle policies.
- **IBM Cloud Object Storage:** Similar to AWS S3, providing highly scalable object storage with integrated encryption and lifecycle management.
- **GCP Cloud Storage:** Object storage with similar features to AWS S3, designed for scalability and security.
- **Azure Blob Storage:** Object storage for unstructured data, comparable to AWS S3 and GCP Cloud Storage, with a strong focus on integration into other Azure services.

5. Networking

- **AWS VPC (Virtual Private Cloud):** Allows you to create isolated networks in the cloud. AWS also provides **Elastic Load Balancer (ELB)** and **Route 53** for DNS management.
- **IBM Cloud VPC:** Offers a private, isolated network with integrated load balancing and VPN services. **IBM Cloud Direct Link** allows for private connectivity to on-premises infrastructure.
- **GCP VPC:** Provides global networking with internal and external IPs, global load balancing, and Cloud CDN for content distribution.
- **Azure Virtual Network (VNet):** Similar to AWS VPC, it allows private networks and integrates with **Azure Load Balancer** and **Azure Traffic Manager** for global distribution.

6. Database Services

- **AWS RDS (Relational Database Service):** Managed database service supporting multiple engines (MySQL, PostgreSQL, SQL Server, etc.), along with **Aurora** (a fully managed database).
- **IBM Cloud Databases:** Managed database services for multiple relational and NoSQL databases (including PostgreSQL, MongoDB, and MySQL).
- **GCP Cloud SQL:** Managed relational databases for MySQL, PostgreSQL, and SQL Server.
- **Azure SQL Database:** Managed SQL Server databases with scalable performance and high availability. Azure also offers **Cosmos DB** for NoSQL workloads.

7. AI and Machine Learning

- **AWS AI Services:** Services like **SageMaker** for ML model building and **Rekognition** for image recognition.
- **IBM Watson:** A suite of AI-powered tools, including NLP, visual recognition, and Watson Studio for building and deploying ML models.
- **GCP AI Platform:** Provides tools like **TensorFlow** for ML and various pre-built APIs for image, text, and video analysis (like **Vision API**).

- **Azure AI:** Offers machine learning services through **Azure Machine Learning**, as well as pre-built APIs for vision, speech, and text processing.

8. Monitoring and Logging

- **AWS CloudWatch:** Comprehensive monitoring service for AWS resources, including metrics, logs, and alarms.
- **IBM Cloud Monitoring:** Uses **Prometheus** and **Grafana** to monitor cloud resources and applications.
- **GCP Stackdriver:** Now rebranded as **Google Cloud Operations Suite**, it provides monitoring, logging, and diagnostics.
- **Azure Monitor:** A comprehensive monitoring service that collects, analyzes, and acts on telemetry data.

9. Security and Compliance

- **AWS Security Hub:** Centralized security management service that integrates with other AWS services.
- **IBM Cloud Security:** Includes services for encryption, firewall management, and identity access management, along with **IBM Cloud Security Advisor**.
- **GCP Security:** Uses services like **Cloud Security Command Center** for centralized security management and **Cloud Identity & Access Management (IAM)** for secure access control.
- **Azure Security Center:** Provides unified security management and threat protection across Azure resources.

10. Automation and Orchestration

- **AWS CloudFormation:** Infrastructure as code (IaC) service for automating resource provisioning using JSON/YAML templates.
- **IBM Cloud Schematics:** Infrastructure as code service based on **Terraform**, which allows you to define cloud resources using configuration files.
- **GCP Deployment Manager:** IaC service that allows you to manage Google Cloud resources through templates.
- **Azure Resource Manager (ARM):** Manages resources using templates in **JSON** format for provisioning and automation.

11. Container Orchestration and Serverless Computing

- **AWS ECS and EKS:** **Elastic Container Service (ECS)** and **Elastic Kubernetes Service (EKS)** are used for container orchestration.
- **IBM Cloud Kubernetes Service:** Managed Kubernetes offering for containerized applications.
- **GCP Kubernetes Engine:** Managed Kubernetes cluster for container orchestration.
- **Azure Kubernetes Service (AKS):** Managed Kubernetes service on Azure for containerized applications.

12. Cost Management

- **AWS Cost Explorer:** Tool for visualizing costs and identifying opportunities for savings.
- **IBM Cloud Cost Management:** Helps track and optimize cloud costs and usage.
- **GCP Cost Management:** **Google Cloud Pricing Calculator** and **Budgets & Billing** help you predict and optimize cloud expenses.

- **Azure Cost Management:** Provides insights into cloud cost data, helping with budgeting and resource optimization.

Summary of Key Differences:

- **Terminology:** Each cloud platform has its own set of names for core services (e.g., VPC in AWS vs. VNet in Azure).
- **Specialized Services:** IBM Cloud is well-known for its AI and cognitive services through Watson, while AWS, GCP, and Azure lead in containerization and serverless computing.
- **Integration with Other Services:** AWS, GCP, and Azure are heavily integrated with their respective ecosystems (e.g., AWS with EC2 and Lambda, GCP with BigQuery and TensorFlow, Azure with its enterprise solutions).
- **User Interface and Management Tools:** Each platform provides unique consoles or dashboards for managing resources, but all include rich CLI tools, APIs, and SDKs for automation.

Understanding the similarities and differences between these platforms will help you choose the right cloud environment for your needs based on specific services, scalability, compliance, and other criteria

Ex No : 2

Create cloud account

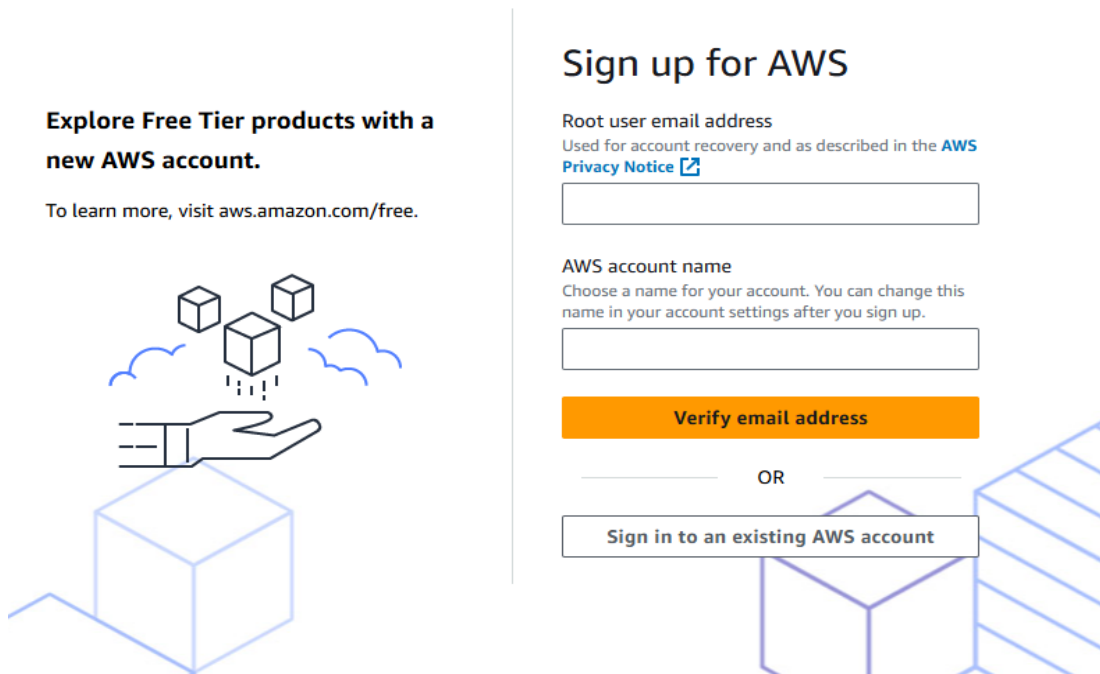
AIM:

To create free cloud accounts on both **AWS** and **IBM Cloud**, follow these step-by-step instructions:

Creating a Free AWS Account

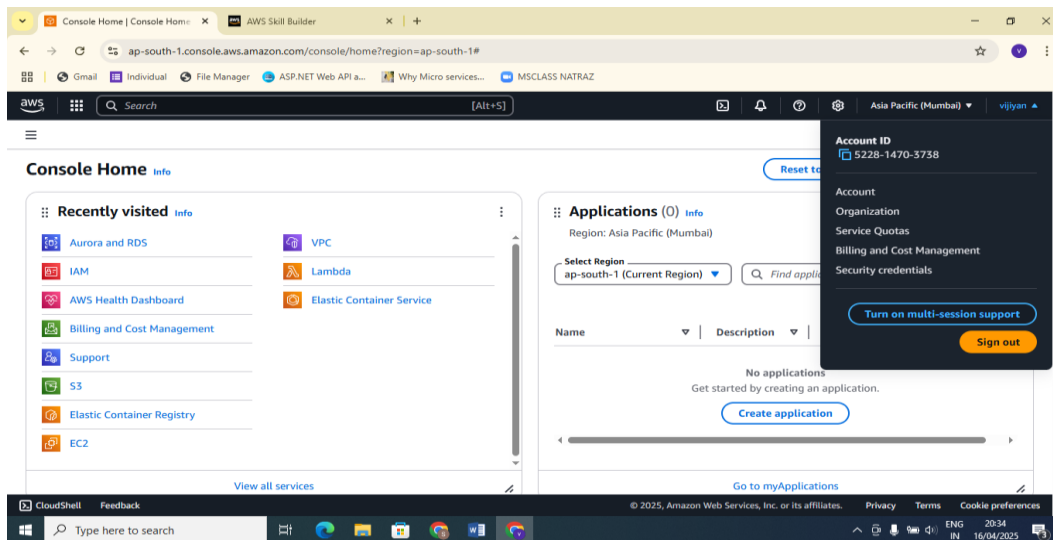
Step 1: Sign Up for AWS

1. **Visit AWS:** Go to the AWS Free Tier page and click on **Create a Free Account**.
2. **Enter your Email:** Provide your email address, create a password, and choose your AWS account name.



The image shows a screenshot of the AWS 'Sign up for AWS' page. On the left, there is a promotional banner for the 'Explore Free Tier products with a new AWS account' with a link to 'aws.amazon.com/free'. Below the banner is an illustration of a hand placing a cube on a larger cube, with other cubes floating above. The main form area on the right is titled 'Sign up for AWS'. It contains two input fields: 'Root user email address' (with a link to the AWS Privacy Notice) and 'AWS account name' (with a note that the name can be changed in settings). Below these fields is an orange 'Verify email address' button. Underneath the button is an 'OR' separator and a button labeled 'Sign in to an existing AWS account'.

3. **Provide Contact Information:** Fill out your contact details, such as your name, phone number, and address.
4. **Payment Information:** AWS requires you to provide a valid credit card or debit card to verify your identity, but don't worry—you won't be charged for using the Free Tier services as long as you stay within the free limits.
 - AWS Free Tier provides limited access to services (e.g., 750 hours of EC2, 5 GB of S3 storage, etc.).
 - Note that you might incur charges if you exceed these free limits.
5. **Identity Verification:** AWS will ask you to verify your phone number with a text message or a voice call.
6. **Select a Support Plan:** AWS offers various support plans. For the free account, you can select the **Basic Support Plan** (free).
7. **Complete Sign-Up:** Review your information, agree to the terms, and click **Complete Sign-Up**.



Step 2: Access the AWS Console

- After your account is created, you will be directed to the **AWS Management Console**, where you can start using various AWS services, including EC2, S3, and Lambda.
- You can also navigate to the **Billing and Cost Management Dashboard** to monitor your usage and stay within the free tier.

Creating a Free IBM Cloud Account

Step 1: Sign Up for IBM Cloud

1. **Visit IBM Cloud:** Go to the IBM Cloud Free Tier page and click **Create an IBM Cloud account**.
2. **Provide Your Email:** Enter your email address and create a password for your IBM Cloud account.
3. **Account Information:** You'll need to provide basic information such as your name and company details (optional).
4. **Payment Information:** IBM Cloud requires you to provide payment details (credit card or debit card) to verify your identity.
 - Like AWS, IBM Cloud will not charge you for using **free tier** services as long as you stay within the usage limits (e.g., 256MB of free Cloud Foundry memory, free Kubernetes cluster, and more).
5. **Agree to Terms:** Review and accept IBM's terms of service and privacy policy.
6. **Verify Your Email:** IBM Cloud will send a verification email to confirm your email address. Click the link in the email to activate your account.

Step 2: Access the IBM Cloud Console

Create an IBM Cloud account

Already have an IBM Cloud account? [Log in](#)

Account information

Email

Password

OR

Sign up with Google



Next ↓

Unlock the full IBM Cloud Catalog

Get started with everything you need to take your projects to the next level. Including always-free products like IBM Watson® APIs. They never expire and you can't be charged for them—ever.

200 USD Cloud Credit for free

Try any IBM Cloud product with \$200 in credit to use over the next 30 days.

Get started for free

We ask for your credit card to verify your identity. You will not be charged for any usage within the free tier. Learn more: www.ibm.com/cloud/free.

- After successfully signing up, you'll be directed to the **IBM Cloud Dashboard**, where you can begin utilizing free resources.
- IBM Cloud also provides access to various services, such as **IBM Cloud Functions**, **IBM Cloud Object Storage**, and **Kubernetes**. You can also monitor usage and set up alerts to ensure you stay within the free tier.

Summary of Free Tier Benefits

- **AWS Free Tier:**
 - 750 hours of EC2 (t2.micro instances), 5GB of S3 storage, 1 million requests per month for Lambda, and more.
 - 12-month free access to many services.
- **IBM Cloud Free Tier:**
 - 256MB of Cloud Foundry memory, 1GB of Object Storage, free Kubernetes cluster, and more.
 - Free access to many tools like **Watson** for AI services, **App Deployment** tools, and more.

Ex NO : 3

Host a Static Website on Cloud Storage on AWS

AIM

To host a static webpage for free on AWS, you can use **Amazon S3 (Simple Storage Service)** and check the url in various browser

Procedure:

1. Set Up an S3 Bucket

- **Log in to AWS:** Go to the [AWS Management Console](#).
- **Create an S3 Bucket:**
 - Open the **S3** service in the AWS Console.
 - Click **Create bucket**.
 - Choose a unique name for your bucket (the name must be globally unique across all of AWS).
 - Choose a region for your bucket (pick one closest to your target audience).
 - Leave the rest of the options as default, but ensure **Block all public access** is disabled (you can enable public access for your files).
 - Click **Create bucket**.

The screenshot shows the 'Create bucket' page in the AWS Management Console. At the top, there is a breadcrumb trail: 'Amazon S3 > Buckets > Create bucket'. Below this, the title 'Create bucket' is followed by an 'Info' link. A subtitle states: 'Buckets are containers for data stored in S3.' The main configuration area is titled 'General configuration'. It includes an 'AWS Region' dropdown set to 'Asia Pacific (Mumbai) ap-south-1'. Under 'Bucket type', there are two options: 'General purpose' (selected with a radio button) and 'Directory'. The 'General purpose' option has a description: 'Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.' The 'Directory' option is described as 'Recommended for low-latency use cases. These buckets use only the S3 Express On which provides faster processing of data within a single Availability Zone.' Below the bucket type selection, there is a 'Bucket name' field with an 'Info' link. The field contains the text 'myawsbucket'. A note below the field states: 'Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). [Le](#)'. At the bottom, there is a section for 'Copy settings from existing bucket - optional' with a note: 'Only the bucket settings in the following configuration are copied.' and a 'Choose bucket' button.

2. Upload Your Website Files

- Go to your newly created bucket in the S3 Console.
- Click on **Upload**, then **Add files** and select your HTML, CSS, JS, and image files.
- Once you've selected the files, click **Upload**.

3. Make Your Bucket Public

- After uploading your files, you need to configure the bucket to allow public access:
 - Select your bucket.
 - Go to the **Permissions** tab.
 - Under **Block public access**, click **Edit** and uncheck the "Block all public access" option. Confirm the change.
 - Next, go to the **Bucket Policy** section and add a policy to allow public access. Here's a basic policy example that you can copy-paste into the Bucket Policy editor:

Json Copy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::YOUR-BUCKET-NAME/*"
    }
  ]
}
```

Replace YOUR-BUCKET-NAME with the name of your S3 bucket.

4. Enable Static Website Hosting

- In your S3 bucket's settings, go to the **Properties** tab.
- Under the **Static website hosting** section, click **Edit**.
- Enable **Use this bucket to host a website**.
- Enter the **index document** (usually index.html), and optionally, an **error document** (usually error.html).
- Click **Save changes**.

You'll now see the **Endpoint** URL (something like <http://your-bucket-name.s3-website-us-east-1.amazonaws.com>), which is the URL for your website.

Ex NO : 4

Host a Static Website on Cloud Storage on IBM

AIM

Hosting a static website on IBM Cloud can be done using IBM Cloud Object Storage or IBM Cloud Foundry

Procedure:

Method 1: Using IBM Cloud Object Storage

IBM Cloud Object Storage allows you to host static websites easily. Here's how:

Step 1: Create an IBM Cloud Object Storage Instance

1. Log in to [IBM Cloud](#).
2. Navigate to **Catalog** → **Storage** → **Cloud Object Storage**.
3. Click **Create** and select a pricing plan (Lite is free with limited features).
4. Click **Create**.

Step 2: Create a Bucket

1. Open the **Object Storage** service.
2. Click **Buckets** → **Create Bucket**.
3. Choose a unique name and select **Standard** as the storage class.
4. Enable **Public Access** (if you want the site to be publicly accessible).
5. Click **Create**.

Step 3: Upload Your Website Files

1. Open the bucket you created.
2. Click **Upload Objects** and select your HTML, CSS, JavaScript, and other assets.
3. Once uploaded, set the main HTML file (e.g., `index.html`) as the root document.

Step 4: Enable Static Website Hosting

1. Inside the bucket, go to **Configuration** → **Static Website Hosting**.
2. Toggle **Enable Static Website Hosting**.
3. Set `index.html` as the index document.
4. Note the generated website URL.

Step 5: (Optional) Configure Custom Domain

- You can map a custom domain to your website using **IBM Cloud Internet Services (CIS)**.
- Update your domain's DNS settings to point to the IBM Cloud Object Storage endpoint.

Ex.NO 5A: Install CLI and configure IAM

Aim

To download the cli pack and install then configure the aws account

Procedure

To use AWS services from your terminal, you need to **configure the AWS CLI** with your **Access Key ID**, **Secret Access Key**, and **Region**. Here's how you can do it:

🔒 Step 1: Create AWS Access Key ID and Secret Access Key

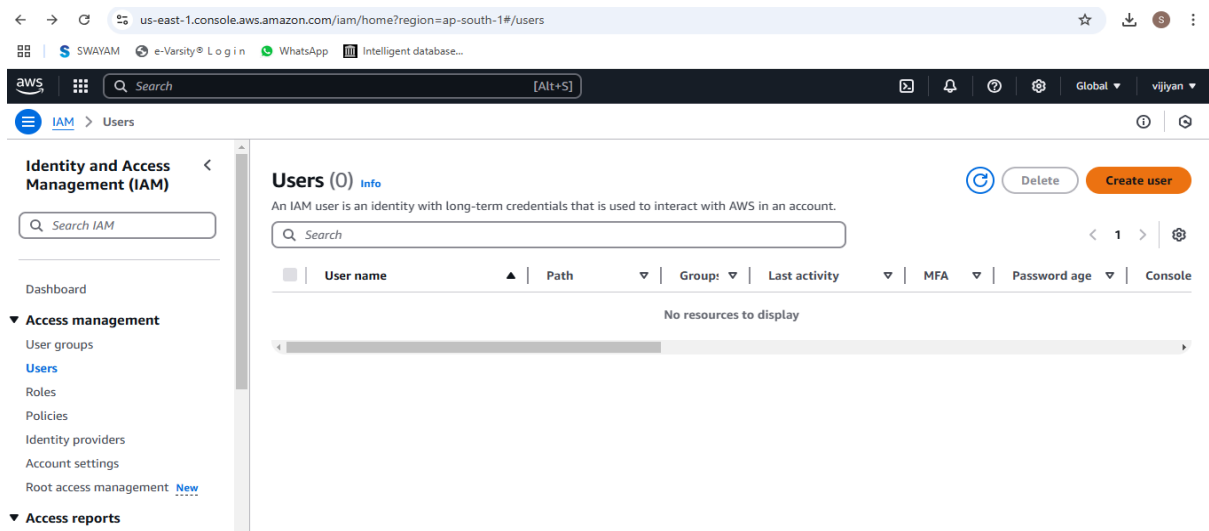
1. Sign in to AWS Management Console:

- Go to aws.amazon.com and sign in to your AWS account.

2. Go to IAM (Identity and Access Management):

- In the AWS Console, search for **IAM** in the search bar and select it.

3. Create a New User (If Needed):



- Click on **Users** (left menu) → **Add Users**.
- Set a username and choose **Programmatic access**.
- Attach the policy **AdministratorAccess** (for full access) or create a custom policy with limited permissions.

IAM > Users > Create user

123

Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

Step 4
Retrieve password

Specify user details

User details

User name

varadharajan

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and +, =, @, _ - (hyphen)

☒ Provide user access to the AWS Management Console - optional

If you're providing console access to a person, it's a best practice [to](#) manage their access in IAM Identity Center.

1 Are you providing console access to a person?

User type

☐ Specify a user in Identity Center - Recommended

We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage user access to their AWS accounts and cloud applications.

☒ I want to create an IAM user

We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keyspaces, or a backup credential for emergency account access.

Console password

☒ Autogenerated password

You can view the password after you create the user.

☐ Custom password

Enter a custom password for the user.

Must be at least 8 characters long

Click next

IAM > Users > Create user

123

Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

Step 4
Retrieve password

Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

User details

User name

varadharajan

Console password type

Autogenerated

Require password reset

Yes

Permissions summary

Name

IAMUserChangePassword

Type

AWS managed

Used as

Permissions policy

Tags - optional

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Cancel

Previous

Create user

Create access key

Permissions
Groups
Tags
Security credentials
Last Accessed

Console sign-in
Manage console access

Console sign-in link
<https://522814703738.signin.aws.amazon.com/console>

Console password
Updated 2 minutes ago (2025-02-19 03:36 GMT+5:30)

Last console sign-in
Never

Multi-factor authentication (MFA) (0)
Remove Resync Assign MFA device

Use MFA to increase the security of your AWS environment. Signing in with MFA requires an authentication code from an MFA device. Each user can have a maximum of 8 MFA devices assigned. [Learn more](#)

Type	Identifier	Certifications	Created on
No MFA devices. Assign an MFA device to improve the security of your AWS environment			

Assign MFA device

Access keys (0)
Create access key

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

No access keys. As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. [Learn more](#)

Create access key

4. Get Access Keys:

- Under **Security credentials**, click **Create access key**.
- AWS will show you the **Access Key ID** and **Secret Access Key**.
⚠ Important: Download the CSV file or save them immediately (you won't be able to view the secret key again).

Summary

ARN arn:aws:iam::522814703738:user/varadharajan	Console access Enabled without MFA	Access key 1 AKIAXTORPJB5MJSL6Q73 - Active Never used. Created today.
Created February 19, 2025, 03:36 (UTC+05:30)	Last console sign-in Never	Access key 2 Create access key

5. AWS CLI MSI

1.Download and run the AWS CLI MSI installer for Windows (64-bit):

<https://awscli.amazonaws.com/AWSCLIV2.msi>

2.To confirm the installation, open the **Start** menu, search for cmd to open a command prompt window, and at the command prompt use the `aws --version` command.

```
C:\Users\Dell>aws --version
aws-cli/2.24.7 Python/3.12.6 Windows/10 exe/AMD64
```

7. Configure aws

```
C:\Users\Dell>aws configure
AWS Access Key ID [None]: AKIAXTORPJB5MJS�6Q73
AWS Secret Access Key [None]: tmcteb/ToBYLUwDSJYGmqFs4ZWacAjdRZpnF8Rq
Default region name [None]: ap-south-1
Default output format [None]: json
```

8. Verify AWS CLI Configuration

Check if AWS CLI is properly configured:

```
aws sts get-caller-identity
```

You should see output with your **AWS Account ID** and **User ARN**, confirming the configuration is successful.

```
C:\Users\Dell>aws sts get-caller-identity
{
  "UserId": "AIDAXTORPJB5D5X5B64RV",
  "Account": "522814703738",
  "Arn": "arn:aws:iam::522814703738:user/varadharajan"
}
```

Step 1: Install AWS CLI and Docker (Local Machine)

First, ensure **Docker** and **AWS CLI** are installed on your local machine.

- **Install Docker:** Download here
- **Install AWS CLI:** Download here
- **Configure AWS CLI:**

```
aws configure
```

Enter your **AWS Access Key**, **Secret Key**, **Region**, and **Output format** (e.g., json).

Step 2: Create a Flask App and Dockerfile

1. Create a project folder:

```
mkdir flask-app && cd flask-app
```

2. Create a simple Flask app (app.py):

```
from flask import Flask
from urllib.parse import quote
app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello, World! This is a containerized app running on AWS ECS!"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=80)
```

3. Create a requirements.txt:

```
Flask==2.0.1
Werkzeug==2.0.3
```

4. Create a Dockerfile:

```
# Copy the requirements file
COPY requirements.txt .
# Install dependencies
RUN pip install --no-cache-dir -r requirements.txt
# Copy the application code
COPY . .
# Expose port 80
EXPOSE 80
# Run the application
CMD ["python", "app.py"]
```

Step 3: Build and Push Docker Image to Amazon Elastic Container Registry (ECR)

1. Authenticate Docker with AWS ECR:

Syntax:

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin <AWS_ACCOUNT_ID>.dkr.ecr.us-east-1.amazonaws.com
```

Example:

```
>aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 522814703738.dkr.ecr.ap-south-1.amazonaws.com/myapp1
```

2. Create an ECR repository:

```
aws ecr create-repository --repository-name myapp1
```

3. Build the Docker image:

```
docker build -t myapp .
```

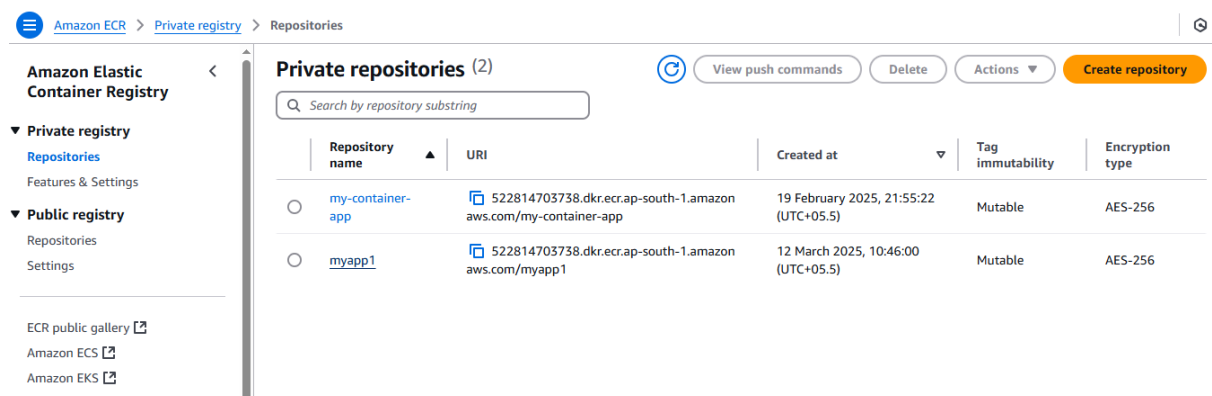
4. Tag the Docker image:

```
docker tag myapp1:latest 522814703738.dkr.ecr.ap-south-1.amazonaws.com/myapp1:latest
```

5. Push the image to AWS ECR:

```
docker push 522814703738.dkr.ecr.ap-south-1.amazonaws.com/myapp1:latest
```

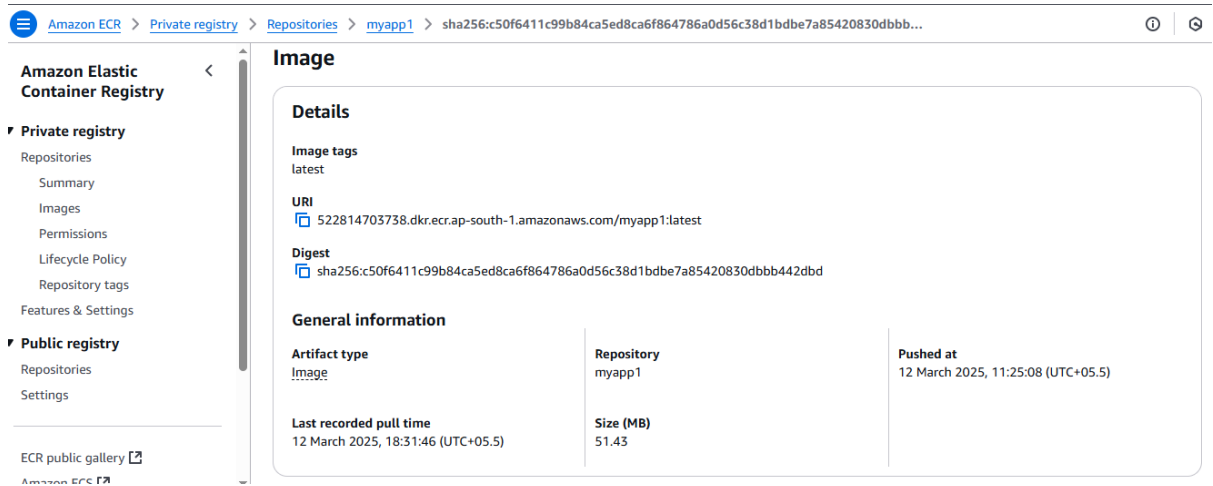
After check aws console,



The screenshot shows the Amazon Elastic Container Registry (ECR) console. The left sidebar contains navigation links for 'Amazon Elastic Container Registry', 'Private registry' (with sub-links for 'Repositories' and 'Features & Settings'), and 'Public registry' (with sub-links for 'Repositories' and 'Settings'). The main content area is titled 'Private repositories (2)' and includes a search bar and buttons for 'View push commands', 'Delete', 'Actions', and 'Create repository'. A table lists the repositories:

	Repository name	URI	Created at	Tag immutability	Encryption type
<input type="radio"/>	my-container-app	522814703738.dkr.ecr.ap-south-1.amazonaws.com/my-container-app	19 February 2025, 21:55:22 (UTC+05.5)	Mutable	AES-256
<input type="radio"/>	myapp1	522814703738.dkr.ecr.ap-south-1.amazonaws.com/myapp1	12 March 2025, 10:46:00 (UTC+05.5)	Mutable	AES-256

Docker image added in aws ecr



Step 4: Deploy to AWS ECS (Fargate)

1. Create a Cluster:

```
aws ecr create-repository --repository-name myapp1
```

2. Create a Task Definition (task-def.json):

```
{
  "family": "my-task",
  "networkMode": "awsvpc",
  "executionRoleArn":
"arn:aws:iam::522814703738:role/ecsTaskExecutionRole",
  "containerDefinitions": [
    {
      "name": "myapp1",
      "image": "522814703738.dkr.ecr.ap-south-
1.amazonaws.com/myapp1:latest",
      "memory": 512,
      "cpu": 256,
      "essential": true,
      "portMappings": [
        {
          "containerPort": 3000,
          "hostPort": 3000
        }
      ]
    }
  ],
  "requiresCompatibilities": ["FARGATE"],
  "cpu": "256",
  "memory": "512"
}
```

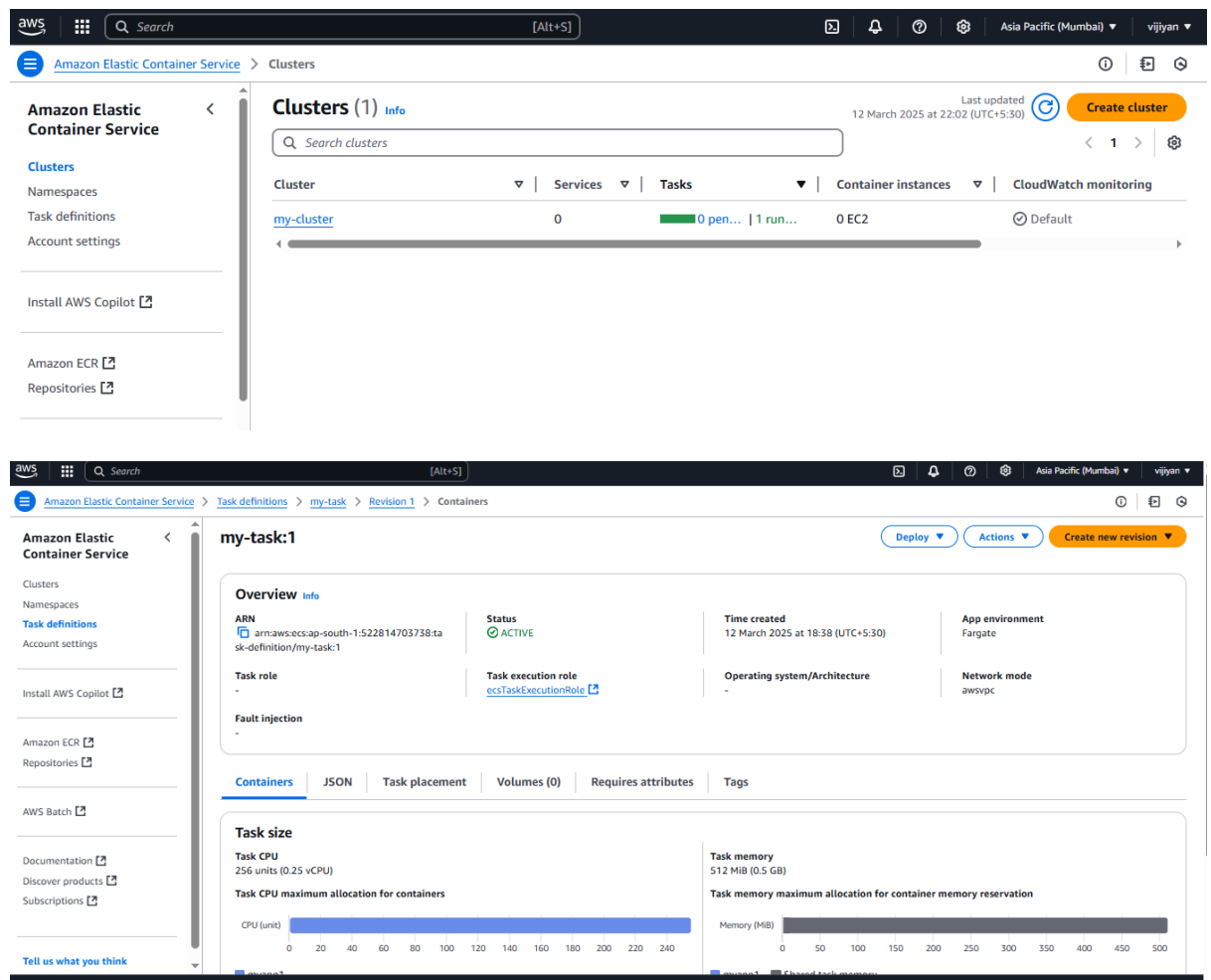
Register the task:

```
aws ecs register-task-definition --cli-input-json file://task-def.json
```

3. Create a Fargate Service:

```
aws ecs create-service --cluster flask-cluster --service-name flask-  
service \  
    --task-definition flask-task --desired-count 1 --launch-type  
FARGATE \  
    --network-configuration  
"awsVpcConfiguration={subnets=[SUBNET_ID],securityGroups=[SECURITY_GR  
OUP_ID],assignPublicIp=\"ENABLED\"}"
```

Otherwise you can create service at aws console



Task running

Amazon Elastic Container Service > Clusters > my-cluster > Tasks > 94860d8a2ab1484db0a4f10db24df860 > Configuration

Amazon Elastic Container Service

Clusters

Namespaces

Task definitions

Account settings

Install AWS Copilot

Amazon ECR

Repositories

AWS Batch

Documentation

Discover products

Subscriptions

Tell us what you think

Fargate ephemeral storage

Encryption	Info	Size (GiB)	20
Default AWS Fargate encryption			

Configuration

Operating system/Architecture	Capacity provider	ENI ID	Public IP
Linux/X86_64	-	eni-0506517d6546c5718	13.203.27.143 open address
CPU Memory	Launch type	Network mode	Private IP
25 vCPU 5 GB	FARGATE	awsvpc	172.31.24.210
Platform version	Container instance IDs:	Subnet ID	MAC address
1.4.0	-	subnet-0b6689053d61f08cd	06:5b:01:eb:56:b1

Container details for myapp1

Details | Log configuration | Restart policy | Network bindings | Docker labels and hosts | Environment variables and files | Volume configuration

Details

Image URI	Essential	Command
522814703738.dkr.ecr.ap-south-1.amazonaws.com/myapp1:latest	Yes	-

Step 5: Access the Application

- Get the **public IP** or **load balancer URL** from ECS and access the app in the browser:

`http://<PUBLIC_IP>:5000`

Ex No 6 : Deploy a simple containerized app on ibm cloud

Deploying a **containerized application** on **IBM Cloud** can be done using **IBM Cloud Kubernetes Service (IKS)** or **IBM Code Engine**. Below, I will walk you through deploying a **simple Flask app** using **IBM Code Engine**, which is a serverless platform that automatically manages infrastructure.

Step 1: Install IBM Cloud CLI & Plugins

1. **Install IBM Cloud CLI** from IBM Cloud CLI.
2. **Log in to IBM Cloud:**

```
ibmcloud login
```

If using an API key:

```
ibmcloud login --apikey YOUR_API_KEY
```

3. **Install the Code Engine plugin:**

```
ibmcloud plugin install code-engine
```

Step 2: Create a Simple Flask App

1. **Create a project folder:**

```
mkdir flask-app && cd flask-app
```

2. **Create a Flask app (app.py):**

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "Hello, IBM Cloud with Flask!"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)
```

3. **Create a requirements.txt:**

```
flask
gunicorn
```

4. **Create a Dockerfile:**

```
# Use Python base image
FROM python:3.9
```

```
# Set working directory
WORKDIR /app

# Copy files
COPY . /app

# Install dependencies
RUN pip install -r requirements.txt

# Expose port
EXPOSE 8080

# Start application
CMD ["gunicorn", "-b", "0.0.0.0:8080", "app:app"]
```

Step 3: Build and Push Docker Image to IBM Cloud Container Registry

1. **Target the IBM Cloud region:**

```
ibmcloud target -r us-south
```

2. **Enable IBM Cloud Container Registry (ICR):**

```
ibmcloud cr region-set us-south
ibmcloud cr namespace-add mynamespace
```

3. **Login to IBM Cloud Container Registry:**

```
ibmcloud cr login
```

4. **Build and tag Docker image:**

```
docker build -t us.icr.io/mynamespace/flask-app:latest .
```

5. **Push the image to IBM Cloud Container Registry:**

```
docker push us.icr.io/mynamespace/flask-app:latest
```

Step 4: Deploy Flask App on IBM Code Engine

1. **Create a Code Engine project:**

```
ibmcloud ce project create --name my-flask-project
ibmcloud ce project select --name my-flask-project
```

2. **Deploy the application:**

```
ibmcloud ce app create --name flask-app --image
us.icr.io/mynamespace/flask-app:latest --port 8080
```

3. **Check deployment status:**

```
ibmcloud ce app get --name flask-app
```

4. **Get the public URL:**

```
ibmcloud ce app get --name flask-app --output url
```

Example output:

```
https://flask-app.us-south.codeengine.appdomain.cloud
```

Ex No : 7 Set Up a Simple Database using AWS

AIM:

Setting up a simple database on AWS Free Tier involves a few key steps. Here's a step-by-step guide:

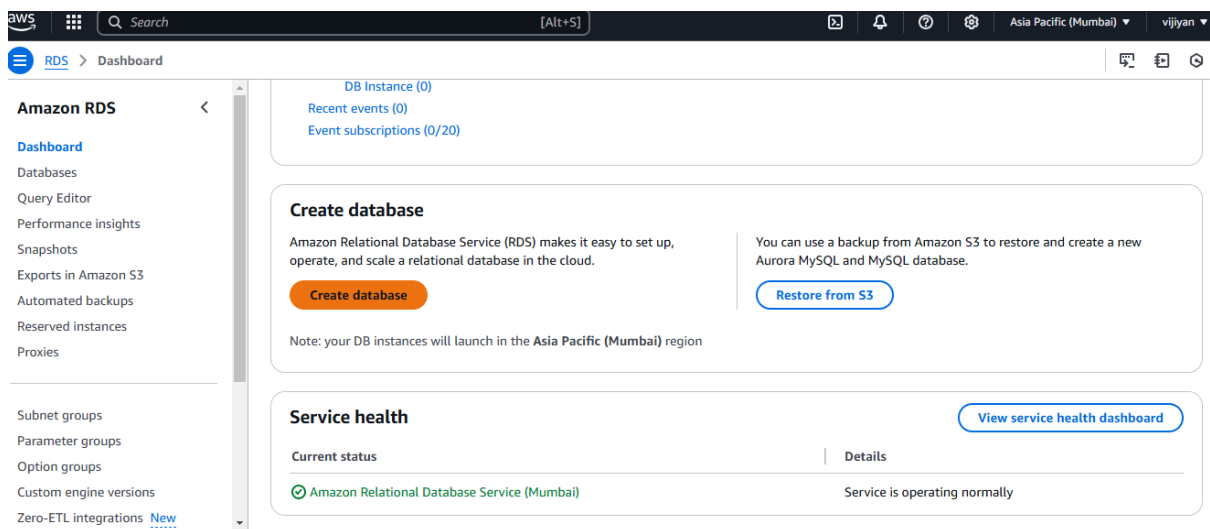
Procedure:

Step 1: Sign in to AWS

1. Go to aws.amazon.com and sign in to your account.
 - If you don't have an account, create one. The AWS Free Tier is free for the first 12 months.
-

Step 2: Navigate to RDS (Relational Database Service)

1. In the AWS Management Console, search for **RDS** in the search bar and select it.
2. Click **Create database**.



Step 3: Configure the Database

1. **Choose a database engine:** Select **MySQL**, **PostgreSQL**, or **MariaDB** (all have Free Tier options).
2. **Choose a database creation method:** Select **Standard create**.
3. **Templates:** Choose **Free tier**.

Deployment options Info

Choose the deployment option that provides the availability and durability needed for your use case. AWS is committed to a certain level of uptime depending on the deployment option you choose. Learn more in the [Amazon RDS service level agreement \(SLA\)](#).

Multi-AZ DB cluster deployment (3 instances)

Creates a primary DB instance with two readable standbys in separate Availability Zones. This setup provides:

- 99.9% uptime
- Redundancy across Availability Zones
- Increased read capacity
- Reduced write latency

Multi-AZ DB instance deployment (2 instances)

Creates a primary DB instance with a non-readable standby instance in a separate Availability Zone. This setup provides:

- 99.9% uptime
- Redundancy across Availability Zones

Single-AZ DB instance deployment (1 instance)

Creates a single DB instance without standby instances. This setup provides:

- 99.9% uptime
- No data redundancy

MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 64 TiB.
- Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
- Supports automated backup and point-in-time recovery.
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.

4. **DB instance identifier:** Enter a name for your database (e.g., mydb).
5. **Master username and password:** Set a username and a strong password.

Step 4: Instance and Storage Settings

1. **DB instance class:** Choose `db.t3.micro` (Free Tier eligible).
2. **Storage type:** Use **General Purpose (SSD)**.
3. **Allocated storage:** Set to 20 GiB (Free Tier limit).
4. **Storage autoscaling:** You can disable it to stay within Free Tier limits.

Step 5: Connectivity Settings

1. **VPC:** Use the default VPC.
2. **Public access:** Enable public access only if you need to connect from outside AWS (e.g., your laptop).
3. **VPC security group:** Create a new security group or use an existing one. Allow **inbound traffic on port 3306 (MySQL) or 5432 (PostgreSQL)**.

Configure Security Group (Allow Access)



1. **Go to EC2 > Security Groups**
2. Find the security group for your database.
3. Click **Inbound rules > Edit inbound rules**.
4. **Add Rule:**
 - **Type:** MySQL/Aurora
 - **Protocol:** TCP
 - **Port Range:** 3306
 - **Source:** Your IP (or **Anywhere** for public access)
5. **Save rules.**

Step 6: Additional Settings

1. **Database authentication:** Use password authentication.
2. **Backup:** Keep automatic backups enabled (7-day retention is default).
3. **Monitoring and Maintenance:** Disable enhanced monitoring to stay within Free Tier limits.

Step 7: Launch Database

1. Review your settings and click **Create database**.
2. Wait a few minutes for AWS to provision your database.

Connectivity & security		
Endpoint & port	Networking	Security
Endpoint  database1.c98mk2c8mqat.ap-south-1.rds.amazonaws.com	Availability Zone ap-south-1c	VPC security groups default (sg-0c7824db7a7340f4b)  Active
Port 3306	VPC vpc-08448aa85bf30e094	Publicly accessible Yes
	Subnet group default-vpc-08448aa85bf30e094	Certificate authority Info rds-ca-rsa2048-g1
	Subnets subnet-0b6689053d61f08cd subnet-06a7d1e3a9acbbaa5 subnet-02b0915b3835998ce	Certificate authority date May 20, 2061, 00:10 (UTC+05:30)
	Network type IPv4	DB instance certificate expiration date March 04, 2026, 20:24 (UTC+05:30)

Step 8: Connect to Your Database

- **Find Endpoint:** Go to **RDS > Databases**, select your database, and copy the **Endpoint**.
- **Connect via Client:** Use MySQL Workbench, pgAdmin, or the command line:

```
mysql -h your-endpoint.rds.amazonaws.com -u your-username -p
```

Setup New Connection

—□×

Connection Name:

Type a name for the connection

Connection Method:

Standard (TCP/IP)

Method to use to connect to the RDBMS

Parameters

SSL

Advanced

Hostname:

127.0.0.1

Port:

3306

Name or IP address of the server host - and TCP/IP port.

Username:

root

Name of the user to connect with.

Password:

Store in Vault ...

Clear

The user's password. Will be requested later if it's not set.

Default Schema:

The schema to use as default schema. Leave blank to select it later.

Configure Server Management...

Test Connection

Cancel

OK

Ex No :8 Set Up a Simple Database in IBM Cloud

Aim:

To **step-by-step guide to set up a simple database on IBM Cloud** using **IBM Cloud Databases for PostgreSQL**, which is a managed and secure option suitable for most use cases.

Procedure:

Step 1: Log in to IBM Cloud

1. Go to <https://cloud.ibm.com>
 2. Sign in with your IBM Cloud account (create one if you don't have it yet).
 3. Select or create a **Resource Group** if prompted.
-

✔ Step 2: Navigate to the Database Catalog

1. In the IBM Cloud console, click on "**Catalog**" in the top menu.
 2. Under the **Databases** section, select **Databases for PostgreSQL** (or your preferred database engine).
-

✔ Step 3: Configure Your Database

1. **Service name:** Give your database a recognizable name.
 2. **Location:** Choose a **region** close to your users (e.g., Dallas, Frankfurt, etc.).
 3. **Resource Group:** Select the resource group (default if unsure).
 4. **Pricing Plan:**
 - Choose the **Lite Plan** (free tier, limited resources) for testing or learning.
 - Or choose a **Standard Plan** for production (charges apply).
-

✔ Step 4: Create the Database

1. Click the "**Create**" button.
 2. Wait a few moments while IBM provisions the database instance.
 3. Once provisioned, you'll be taken to the **service dashboard**.
-

✔ Step 5: Get Your Database Credentials

1. On the database dashboard, go to the "**Connections**" tab.
2. Click on "**View Credentials**" or "**Connect**" to see:

- Hostname
- Port
- Username
- Password
- Database name
- SSL certificate link

You can **copy the connection string** for easy use in apps or clients.

✔ Step 6: Connect to Your Database

Use your favorite client, such as:

- **DBeaver**
- **pgAdmin**
- **psql CLI**
- Or from an app (Node.js, Python, etc.)

Example using `psql`:

```
psql "host=<hostname> port=31237 dbname=ibmcloudodb user=<username>
password=<password> sslmode=require"
```

Or using the full connection URI:

```
psql "<connection_string_from_dashboard>"
```

✔ Step 7: Start Using Your Database

Once connected, you can:

- Create tables:

```
CREATE TABLE users (
  id SERIAL PRIMARY KEY,
  name TEXT,
  email TEXT UNIQUE
);
```

- Insert data:

```
INSERT INTO users (name, email) VALUES ('Alice',
'alice@example.com');
```

- Query data:

```
SELECT * FROM users;
```

Enable High Availability & Backups

- In the service dashboard, explore settings like:
 - **Backup Schedule**
 - **Autoscaling**
 - **High Availability** (available in paid plans)

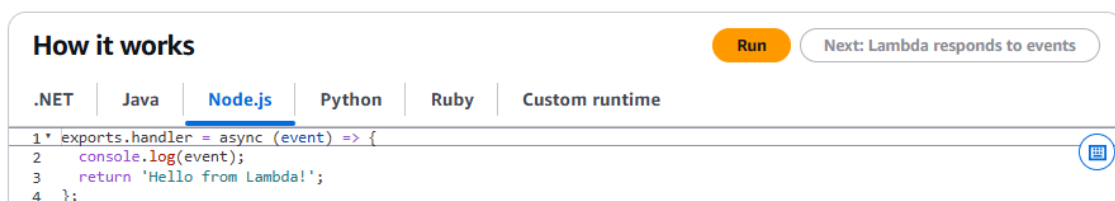
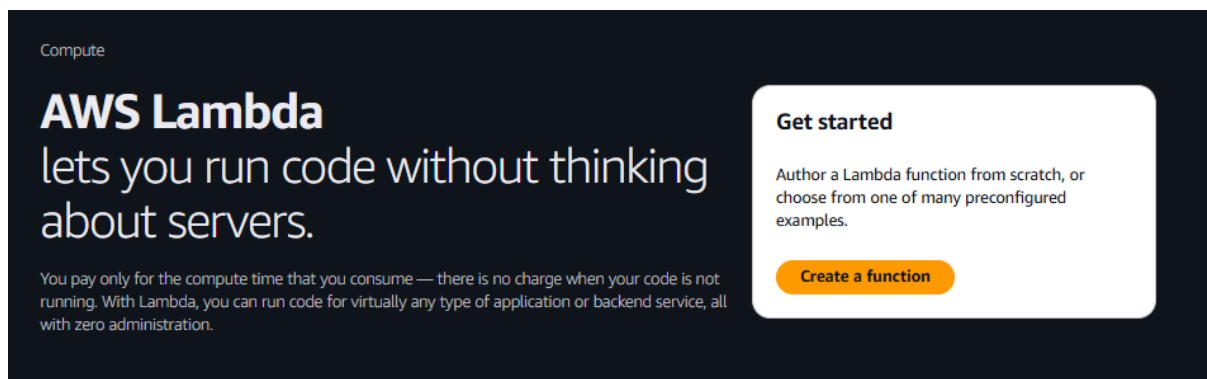
AIM

Building and using a **serverless function** in AWS typically involves **AWS Lambda**, which allows you to run code without provisioning or managing servers. Below is a step-by-step guide to **create, deploy, and invoke** an AWS Lambda function.

Procedure

Step 1: Create a Lambda Function

1. **Sign in to AWS Console**
 - Go to [AWS Lambda Console](#).
2. **Create a New Function**
 - Click **Create function**
 - Choose **Author from scratch**
 - Enter a function name (e.g., MyServerlessFunction)
 - Runtime: Select **Python 3.x, Node.js, or any preferred runtime**
 - Execution Role: Choose **Create a new role with basic Lambda permissions**
 - Click **Create function**



Step 2: Write the Lambda Function

Modify the default function code in the **Code** editor.

Example **Python** function (inside `lambda_function.py`):

```
import json
def lambda_handler(event, context):
    return {
```

```
    "statusCode": 200,  
    "body": json.dumps("Hello from AWS Lambda!")  
}
```

- The function accepts an `event` object and returns a JSON response.
- The `context` provides runtime information.

Click **Deploy** to save changes.

Step 3: Test the Lambda Function

1. Click **Test**
2. Configure a new test event (choose "Hello World" template)
3. Click **Test** again
4. You should see **Execution result: succeeded** with the response.

Step 4: Invoke the Function via AWS CLI

You can invoke the function using the AWS CLI.

1. Install and Configure AWS CLI

If not installed, [download AWS CLI](#) and configure it:

```
aws configure
```

Enter:

- AWS Access Key ID
- AWS Secret Access Key
- Default region (e.g., `us-east-1`)
- Output format (e.g., `json`)

2. Invoke the Lambda Function

Run the following command:

```
aws lambda invoke --function-name MyServerlessFunction response.json
```

This will execute the function and save the response in `response.json`.

Step 5: Deploy via API Gateway (Optional)

To expose the function as a REST API:

1. **Go to API Gateway** in AWS Console
2. Click **Create API > HTTP API**
3. **Create a new API**, choose **Lambda integration**
4. **Select your Lambda function** and click **Create**
5. Deploy the API and note the **Invoke URL**
6. Test it by calling the URL in a browser or using:

```
curl -X GET "https://your-api-id.execute-api.us-east-1.amazonaws.com/"
```

Conclusion

This successfully built and deployed an AWS Lambda function. You can integrate it with **S3**, **DynamoDB**, **API Gateway**, or **event-driven architectures** for various use cases.

Ex.No :10 Implement Basic Cloud Security in aws

Aim:

To check various security mechanism in aws and ensure secure our data

Procedure

🔒 1. Use IAM (Identity and Access Management) Properly

- **Create individual IAM users** — never use the root user for daily tasks.
- **Use IAM groups** with policies to manage permissions easily.
- **Apply the Principle of Least Privilege (PoLP)** — only give users the minimum access they need.

✓ *Quick Setup:*

```
aws iam create-user --user-name YourUser
```

```
aws iam add-user-to-group --user-name YourUser --group-name Developers
```

The screenshot shows the AWS IAM Dashboard. On the left is a navigation sidebar with 'Identity and Access Management (IAM)' selected. The main content area is titled 'IAM Dashboard' and includes two sections: 'Security recommendations' and 'IAM resources'.

Security recommendations (0 recommendations):

- ✓ Root user has MFA: Having multi-factor authentication (MFA) for the root user improves security for this account.
- ✓ Root user has no active access keys: Using access keys attached to an IAM user instead of the root user improves security.

IAM resources (Resources in this AWS Account):

User groups	Users	Roles	Policies	Identity providers
0	1	7	1	0

🔒 2. Enable Multi-Factor Authentication (MFA)

- Turn on **MFA for root** and all privileged IAM users.
- Use a virtual MFA app like Google Authenticator or Authy.

🔗 *To Enable:*

- Go to **IAM > Users > Security credentials > Assign MFA device**

➡ 3. Rotate Access Keys Regularly

- Avoid long-lived credentials.
 - Use roles and temporary credentials (e.g., **STS**).
 - Automatically rotate and disable old keys.
-

☁ 4. Use AWS Organizations & Service Control Policies (SCPs)

- Manage multiple accounts in a structured, secure way.
 - Apply guardrails using SCPs (e.g., prevent deleting logs or changing billing settings).
-

🔍 5. Enable Logging and Monitoring

- **CloudTrail**: Logs all API activity in your account.
- **Config**: Tracks configuration changes.
- **CloudWatch**: Monitors metrics and logs.

🔗 *Enable CloudTrail:*

```
aws cloudtrail create-trail --name myTrail --s3-bucket-name my-log-bucket
aws cloudtrail start-logging --name myTrail
```

🔒 6. Secure S3 Buckets

- **Block public access** unless absolutely necessary.
- Use **bucket policies** and **IAM policies** to control access.
- Enable **versioning** and **logging** for traceability.
- Turn on **encryption** (SSE-S3 or SSE-KMS).

🔗 *Block public access:*

```
aws s3api put-bucket-public-access-block \
--bucket your-bucket-name \
--public-access-block-configuration
BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictP
ublicBuckets=true
```

🔑 7. Use Security Groups and NACLs (Network Layer)

- Restrict access by IP and port.
 - Never leave ports like SSH (22) or RDP (3389) open to the world.
-

🔒 8. Encrypt Everything

- Use **KMS** (Key Management Service) to manage keys.
 - Encrypt:
 - EBS volumes
 - RDS databases
 - S3 data
 - Secrets in Secrets Manager
-

🛡️ 9. Set Up GuardDuty, Inspector, and Macie

- **GuardDuty**: Threat detection (malware, anomalies)
 - **Inspector**: Security assessment (vulnerability scanning)
 - **Macie**: Sensitive data discovery (like PII in S3)
-

☐ 10. Security Best Practices Checklist

- ✓ IAM best practices
- ✓ Regular patching of EC2 & services
- ✓ Network segmentation (use VPCs, subnets)
- ✓ Security assessments & audits
- ✓ Enable alerts for suspicious activity

Ex.No : 11 :

Implement basic security in IBM cloud

AIM

To establish a various security mechanism in IBM cloud account and ensure added protection in your data

Procedure

  Basic Cloud Security in IBM Cloud – Step-by-Step Guide

1. Set Up IBM Cloud IAM Properly

IBM Cloud uses **Identity and Access Management (IAM)** to control access to resources.

 *Actions:*

- **Create individual IAM users** (no sharing accounts!)
- Assign **roles** and **policies** using **principle of least privilege**
- Use **Access Groups** to bundle permissions for teams

```
ibmcloud iam user-create user@example.com
ibmcloud iam access-group-create DevTeam
ibmcloud iam access-group-policy-create DevTeam --roles Viewer,Editor --
service-name cloud-object-storage
```

2. Enable Multi-Factor Authentication (MFA)

Add an extra layer of protection for IBM Cloud account login.

 *Setup via:*

- IBM Cloud UI > Manage > Access (IAM) > Users > [Your user] > Enable MFA
-

3. Use API Keys & Service IDs Securely

- Avoid using personal API keys in production
- Use **Service IDs** with **least-privilege roles**
- Rotate API keys regularly
- Store them in **Secrets Manager**

```
ibmcloud iam service-id-create MyAppService
ibmcloud iam service-api-key-create MyAppKey MyAppService
```

4. Secure IBM Cloud Object Storage (COS)

- **Disable public access** to buckets
- Use **bucket policies** and **IAM controls**
- Enable **encryption (SSE or Key Protect)**

Example:

```
ibmcloud cos bucket-create --bucket secure-data-bucket --class standard --region us-south
```

□ 5. Use VPC for Network Isolation

Instead of Classic infrastructure, use **IBM Cloud VPC**:

- Segment networks using **subnets**
 - Use **security groups** for firewall rules
 - **No public IPs** unless required
-

🔑 6. Manage Secrets & Encryption

Use:

- **IBM Secrets Manager** for API keys, DB credentials
- **IBM Key Protect** for managing encryption keys

```
ibmcloud resource service-instance-create secrets-manager standard us-south
```

🔍 7. Enable Logging & Monitoring

- Use **Activity Tracker** to monitor user/API actions
- Enable **Cloud Monitoring with Sysdig** for infrastructure visibility
- Integrate with **IBM Log Analysis** for logs

□ 8. Set Up Resource Groups and Tags

- Organize resources into **resource groups**
 - Use **tags** to control access and track usage
 - Helps with **cost and security management**
-

👤 9. Use Trusted Profiles

- Use **trusted profiles** for access from services to services
- More secure than static API keys

10. Security Best Practices Checklist

- ✓ IAM and MFA configured
- ✓ VPC, not classic infrastructure
- ✓ No hardcoded credentials
- ✓ Secure object storage
- ✓ Monitor with Activity Tracker
- ✓ Encrypt data at rest & in transit
- ✓ Use trusted profiles & service IDs

[🔗 Optional Tools You Might Want to Explore:](#)

Tool	Purpose
IBM Cloud Security Advisor	Central dashboard for security recommendations
Cloud Pak for Security	Advanced threat management
IBM Cloud Activity Tracker	API access logs
Key Protect / HSM	Encryption key management
IBM Cloud Firewall	Network-level security control

Sample Viva Questions

AWS Interview – Top Frequently Asked Questions

◆ 1. What is AWS?

Answer:

AWS (Amazon Web Services) is a cloud computing platform offering infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). It provides services like computing, storage, databases, networking, machine learning, and more.

◆ 2. What are the main types of cloud computing?

Answer:

- **IaaS** – Infrastructure as a Service (e.g., EC2, S3)
 - **PaaS** – Platform as a Service (e.g., Elastic Beanstalk, Lambda)
 - **SaaS** – Software as a Service (e.g., Amazon WorkMail, Chime)
-

◆ 3. What is EC2?

Answer:

EC2 (Elastic Compute Cloud) is a virtual server that you can configure and run in the cloud. You can choose instance types, OS, storage, and networking.

◆ 4. What is S3?

Answer:

S3 (Simple Storage Service) is an object storage service that allows you to store and retrieve data from anywhere on the web.

◆ 5. What's the difference between S3 and EBS?

Feature	S3	EBS
Storage type	Object	Block
Use case	Static files, backups	Disk for EC2
Access	Over internet	Attached to EC2

◆ 6. What is IAM?

Answer:

IAM (Identity and Access Management) controls **who can access what** in AWS. It lets you manage users, groups, roles, and permissions securely.

◆ 7. What is a Security Group?

Answer:

A virtual firewall for EC2 instances that controls **inbound and outbound traffic**. Works at the **instance level**.

◆ 8. What is an AMI?

Answer:

AMI (Amazon Machine Image) is a pre-configured template for launching EC2 instances, containing the OS, app server, and applications.

◆ 9. What is Auto Scaling?

Answer:

Automatically increases or decreases the number of EC2 instances based on demand to ensure performance and cost-efficiency.

◆ 10. What is CloudWatch?

Answer:

A monitoring service for AWS resources and applications. You can set **alarms**, **view logs**, and create custom dashboards.

◆ 11. What is Elastic Load Balancer (ELB)?

Answer:

Distributes incoming traffic across multiple EC2 instances to improve fault tolerance and scalability.

◆ 12. What is Route 53?

Answer:

AWS's scalable **DNS service** used for domain registration, routing internet traffic, and health checking.

◆ 13. What is the difference between Public and Private Subnet?

- **Public Subnet:** Has a route to the internet (via IGW).
- **Private Subnet:** No direct internet access.

◆ 14. What is AWS Lambda?

Answer:

A **serverless** compute service that lets you run code without provisioning servers. Pay only for execution time.

◆ 15. What is the difference between Lambda and EC2?

Feature	Lambda	EC2
Serverless	✓	✗
Pricing	Per request	Per hour/second
Setup	Minimal	Manual
Use case	Short tasks, events	Full apps, control

◆ 16. What is an AWS VPC?

Answer:

Virtual Private Cloud – a **logically isolated** network within AWS where you can launch resources like EC2.

◆ 17. What is the use of AWS CloudFormation?

Answer:

It lets you **automate and manage** your infrastructure as code using YAML or JSON templates.

◆ 18. What is the Shared Responsibility Model in AWS?

Answer:

- **AWS:** Secures the cloud (hardware, infrastructure).
 - **Customer:** Secures what's *in* the cloud (data, access, apps).
-

◆ 19. What is AWS CLI?

Answer:

Command-line interface tool for managing AWS services via terminal or scripts.

◆ 20. How does AWS ensure security?

- IAM & roles
- MFA
- Logging (CloudTrail)
- Encryption (KMS, SSE)
- VPC & NACLs
- GuardDuty, Macie, Inspector

◆ 21. What is an Elastic IP in AWS?

Answer: A static IP address designed for dynamic cloud computing, which you can associate with EC2 instances.

◆ 22. What is AWS CloudTrail?

Answer: A logging service that records all API calls made in your AWS account for auditing and monitoring.

◆ 23. What is the use of AWS SNS (Simple Notification Service)?

Answer: Pub/sub messaging service used to send alerts or messages (email, SMS, HTTP endpoint).

◆ 24. What is the difference between AWS SNS and SQS?

Feature	SNS	SQS
Type	Push	Pull
Use case	Notifications	Message queue
Fan-out	Yes	No

◆ 25. What is AWS RDS?

Answer: Relational Database Service that lets you launch and manage databases like MySQL, PostgreSQL, SQL Server, and Aurora.

◆ 26. What is the difference between RDS and DynamoDB?

RDS	DynamoDB
Relational	NoSQL
Structured queries (SQL)	Key-value or document-based
Vertical scaling	Horizontal scaling

◆ 27. What is AWS CloudFront?

Answer: A **CDN** that delivers content to users globally with low latency using edge locations.

◆ 28. What are Edge Locations?

Answer: Data centers used by CloudFront to cache content closer to users.

◆ 29. What is AWS Elastic Beanstalk?

Answer: PaaS service to deploy and manage web applications without worrying about infrastructure (supports Java, Python, Node.js, etc.).

◆ 30. What is AWS Auto Scaling Group (ASG)?

Answer: A group of EC2 instances managed together for scaling based on traffic/load.

◆ 31. What is a NAT Gateway?

Answer: Allows EC2 instances in a **private subnet** to connect to the internet without exposing them directly.

◆ 32. What are the different storage classes in S3?

Answer:

- S3 Standard
 - S3 Intelligent-Tiering
 - S3 One Zone-IA
 - S3 Glacier
 - S3 Glacier Deep Archive
-

◆ 33. How does AWS handle high availability?

Answer: Through multiple **Availability Zones (AZs)** within a **region**, load balancing, and auto scaling.

◆ 34. What is AWS EFS?

Answer: Elastic File System – a **shared file storage system** for Linux instances.

◆ 35. What is AWS EBS Snapshot?

Answer: A backup (point-in-time) of an EBS volume stored in S3.

◆ 36. What is the difference between Vertical and Horizontal Scaling?

Vertical	Horizontal
Increase instance size	Add more instances
Limited by hardware	More scalable

◆ 37. What are AWS Availability Zones?

Answer: Isolated data centers in a region designed for fault tolerance and high availability.

◆ 38. What is AWS VPC Peering?

Answer: Allows you to connect two VPCs privately using AWS's backbone network.

◆ 39. What is Amazon Aurora?

Answer: A high-performance, MySQL/PostgreSQL-compatible database engine under RDS.

◆ 40. What is AWS Cost Explorer?

Answer: A tool to **visualize and manage AWS spending**, set budgets, and track usage.

IBM cloud viva Questions

◆ 1. What is IBM Cloud?

Answer:

IBM Cloud is a suite of cloud computing services from IBM that includes **IaaS, PaaS, and SaaS**. It offers virtual servers, Kubernetes, AI tools, databases, and more for building and running applications in a secure and scalable environment.

◆ 2. What are the key service models in IBM Cloud?

Answer:

- **IaaS** – Infrastructure as a Service
 - **PaaS** – Platform as a Service (e.g., Cloud Foundry, Kubernetes)
 - **SaaS** – Software as a Service (e.g., Watson Assistant)
-

◆ 3. What is IBM Cloud Foundry?

Answer:

A PaaS layer in IBM Cloud for deploying and managing applications without managing underlying infrastructure.

◆ 4. What is IBM Cloud CLI?

Answer:

A command-line tool to manage resources on IBM Cloud — create, deploy, and manage apps and services.

◆ 5. What is IBM Cloud Kubernetes Service?

Answer:

A fully managed Kubernetes container orchestration service on IBM Cloud, used to deploy, manage, and scale containerized apps.

◆ 6. What is a Resource Group in IBM Cloud?

Answer:

A way to organize cloud resources logically, making access and billing management easier.

◆ 7. What are IBM Cloud Regions and Zones?

Answer:

- **Region:** Geographic location (e.g., Dallas, London)
 - **Zone:** Data centers within a region, used for high availability
-

◆ 8. What is IBM Virtual Server?

Answer:

A virtual machine (VM) on IBM Cloud, used to host apps, services, or infrastructure.

◆ 9. What is VPC (Virtual Private Cloud) in IBM Cloud?

Answer:

A logically isolated network on IBM Cloud where you can launch and manage virtual server instances and resources.

◆ 10. What is IBM Cloud Identity and Access Management (IAM)?

Answer:

Manages user access and permissions. Supports roles like Viewer, Editor, Operator, and Administrator for different scopes.

◆ 11. What is the difference between Cloud Foundry and Kubernetes in IBM Cloud?

Feature	Cloud Foundry	Kubernetes
Type	PaaS	Container orchestration
App-focused	Yes	No
Flexibility	Less	More
Management	Fully managed	Requires config

◆ 12. What is IBM Cloud Object Storage?

Answer:

Highly scalable, secure, and durable object storage for storing unstructured data like images, videos, backups, and logs.

◆ 13. What is IBM Watson?

Answer:

A suite of AI services that includes language processing, machine learning, and conversational AI (e.g., Watson Assistant, NLU).

◆ 14. What is IBM Watson Assistant?

Answer:

A chatbot/virtual assistant service for building conversational AI interfaces.

◆ 15. What is IBM Code Engine?

Answer:

A fully managed, serverless platform to run containerized workloads, functions, and batch jobs.

◆ 16. What is IBM Cloud Monitoring?

Answer:

Provides observability and monitoring capabilities for cloud apps and infrastructure, using tools like Sysdig.

◆ 17. What is IBM Cloud DNS Services?

Answer:

Manages domain name system (DNS) for apps hosted on IBM Cloud, allowing domain routing.

◆ 18. What is IBM Cloud Secrets Manager?

Answer:

Securely stores and manages secrets like API keys, credentials, and certificates.

◆ 19. What is a Cloud API Key in IBM Cloud?

Answer:

An authentication key used to authorize programmatic access to IBM Cloud resources.

◆ 20. What is IBM Cloud Schematics?

Answer:

A tool for infrastructure-as-code using Terraform to automate provisioning and configuration of resources.

◆ 21. What is IBM Cloud Functions?

Answer:

A serverless compute platform (based on Apache OpenWhisk) to run functions in response to events.

◆ 22. What are IAM Policies in IBM Cloud?

Answer:

Access rules that define who can do what to which resource using IAM roles and actions.

◆ 23. How does billing work in IBM Cloud?

Answer:

Pay-as-you-go model with free-tier services and detailed cost tracking. Budgets and usage alerts are also available.

◆ 24. What is the difference between Classic and VPC infrastructure?

Feature	Classic	VPC
Older model	✓	✗
More control	✗	✓
Security	Basic	Enhanced

◆ 25. What is Ingress in IBM Cloud Kubernetes?

Answer:

Ingress manages external access (like HTTP/HTTPS) to Kubernetes services inside the cluster.

◆ 26. How does IBM Cloud ensure security?

Answer:

- IAM roles and access
 - Data encryption
 - Security groups and firewalls
 - Key Protect & Secrets Manager
 - Logging and monitoring
-

◆ 27. What is IBM Key Protect?

Answer:

A cloud-based key management service that allows you to create and control encryption keys used across IBM Cloud.

◆ 28. What is IBM Cloud CLI used for?

Answer:

To provision and manage resources via command line instead of UI (e.g., apps, services, containers, VPC, IAM).

◆ 29. Can you use Terraform with IBM Cloud?

Answer:

Yes. IBM Cloud supports **Terraform** for IaC (Infrastructure as Code) through **Schematics** or CLI.

◆ 30. What is Multi-Zone Region (MZR) in IBM Cloud?

Answer:

A region with 3 availability zones that ensures **high availability** and **disaster recovery** across zones.