#### Cloud Technology Assignment\_1

Submitted by: Prinsh kumar raj

#### Write steps for creation of AWS EC2 instance.

#### **EC2 (Elastic Compute Cloud)**

* **Definition**: A web service that provides **scalable virtual servers** (instances) in the cloud. It allows you to run applications with flexible computing capacity, paying only for what you use.

### **Steps to Create an AWS EC2 Instance**

1. **Log in** to the AWS Management Console.
2. Go to **EC2 Dashboard** under the "Services" menu.
3. Click on **"Launch Instance"**.
4. **Choose an AMI** (Amazon Machine Image) like Amazon Linux, Ubuntu, etc.
5. **Select Instance Type** (e.g., t2.micro for free tier).
6. **Configure Instance Details** (e.g., network, IAM role, etc.).
7. **Add Storage** (default is 8 GB for free tier).
8. **Add Tags** (optional, e.g., Name = MyInstance).
9. **Configure Security Group** (e.g., allow SSH, HTTP).
10. **Review and Launch** the instance.
11. **Select or Create Key Pair** for SSH access.
12. Click **"Launch"**.
13. **Write steps for creation of IAM Groups.**

#### **IAM (Identity and Access Management)**

* **Definition**: A service for **securely managing access** to AWS resources. It allows you to create users, groups, and roles with specific permissions, ensuring **fine-grained access control**.

### Steps to Create AWS IAM Groups

1. **Log in to the AWS Management Console.**
2. **Go to IAM Dashboard under "Services".**
3. **Click on "User groups" on the left sidebar.**
4. **Click "Create group".**
5. **Enter a Group Name (e.g., AdminGroup).**
6. **Attach Policies (e.g., AdministratorAccess).**
7. **Click "Create group".**
8. **Explain these topics:**
9. **Load Balancing**
10. **Auto scaling**
11. **Lambda**

### **Load Balancing**

* **Distributes incoming traffic** across multiple servers (EC2 instances) to ensure **high availability** and reliability.
* Supports **Elastic Load Balancer (ELB)** types: Application, Network, and Classic Load Balancer.

### **II) Auto Scaling**

* Automatically adjusts the number of EC2 instances based on **demand** (scaling up or down).
* Helps optimize **costs** and maintain **performance** by ensuring enough resources are running.

### **III) AWS Lambda**

* **Serverless compute service** that lets you run code in response to events (like HTTP requests) without managing servers.
* Supports multiple languages (Python, Node.js, Java, etc.) and charges based on **compute time** used.