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

**Step 1: Log in to AWS Console**

1. Go to the [AWS Management Console](https://aws.amazon.com/console/).
2. Log in with your AWS credentials.

**Step 2: Open the EC2 Dashboard**

1. On the AWS Management Console homepage, navigate to **Services** > **Compute** > **EC2**.

**Step 3: Launch an Instance**

1. In the EC2 Dashboard, click **Launch Instance**.
2. Enter a name for your instance.

**Step 4: Choose an Amazon Machine Image (AMI)**

1. Choose an AMI, which serves as the base operating system for your EC2 instance. You can select from:
   * **Quick Start AMIs** (e.g., Amazon Linux, Ubuntu, Windows Server)
   * **My AMIs** (previously created or customized AMIs)
   * **AWS Marketplace** (commercial software images)
2. Select the AMI you want to use.

**Step 5: Select an Instance Type**

1. Choose an instance type that provides the required CPU, memory, and networking capacity. For general purposes, **t2.micro** is often selected, as it’s eligible for the AWS Free Tier.
2. Click **Next** to proceed.

**Step 6: Configure Instance Details**

1. Set the **number of instances** (usually set to 1 for a single instance).
2. Specify the **network settings**, such as the VPC, subnet, auto-assign IP, etc.
3. Choose additional options such as IAM roles, monitoring, and shutdown behaviour if needed.
4. Once configured, click **Next**.

**Step 7: Add Storage**

1. Configure your storage volume for the instance. You can modify the default size or add additional volumes if needed.
2. Once storage is configured, click **Next**.

**Step 8: Add Tags**

1. Optionally, you can add tags in the format of **Key** and **Value** (e.g., Name: MyEC2Instance) to help identify and organize your instances.
2. Click **Next** to proceed.

**Step 9: Configure Security Group**

1. Create or select an existing Security Group to control inbound and outbound traffic.
   * **Inbound Rules**: Add rules to allow necessary traffic (e.g., SSH on port 22 for Linux or RDP on port 3389 for Windows).
   * **Outbound Rules**: Configure based on specific needs or leave default.
2. Click **Review and Launch** once done.

**Step 10: Review and Launch**

1. Review all the configurations for accuracy.
2. Click **Launch** when ready.

**Step 11: Select or Create a Key Pair**

1. In the pop-up, choose an existing key pair or create a new one to securely connect to your instance.
2. Download the key pair if it’s new (ensure you store it securely, as you won’t be able to download it again).
3. Confirm the acknowledgment box and click **Launch Instances**.

**Step 12: Access Your Instance**

1. After the instance initializes, go to the **Instances** page in the EC2 dashboard.
2. Select your instance and copy its **Public DNS** or **IP address**.
3. Use SSH (for Linux) or RDP (for Windows) to connect to the instance using the key pair created.

**Additional Tips**

* **Elastic IP**: For a persistent IP address, consider assigning an Elastic IP.
* **Monitoring**: Use AWS CloudWatch for monitoring and setting alarms.

**Write steps for creation of IAM Groups.**

**Step 1: Log in to the AWS Console**

1. Go to the [AWS Management Console](https://aws.amazon.com/console/).
2. Log in with your AWS credentials.

**Step 2: Open the IAM Console**

1. From the AWS Management Console homepage, navigate to **Services** > **Security, Identity, & Compliance** > **IAM**.

**Step 3: Navigate to User Groups**

1. In the left sidebar, click **User Groups** under **Access Management**.
2. On the User Groups page, click **Create Group**.

**Step 4: Name the Group**

1. Enter a name for your new group in the **Group name** field (e.g., “Developers” or “Admins”).
2. Click **Next** to proceed.

**Step 5: Attach Policies to the Group**

1. In this step, you can attach one or more policies to define the permissions for the group.
2. Select relevant policies by checking the boxes next to them (e.g., **Administrator Access**, **AmazonS3FullAccess**, etc.). You can also use the search bar to find specific policies.
3. After selecting the necessary policies, click **Next**.

**Step 6: Review and Create the Group**

1. Review the group name and attached policies.
2. Once everything looks correct, click **Create Group**.

**Step 7: Add Users to the Group (Optional)**

1. After creating the group, you can add users by selecting the group and clicking **Add users to group**.
2. Select existing users to add to the group, or create new users if needed.
3. Click **Add users** to finalize.

**Additional Tips**

* **Custom Policies**: You can create custom policies under **Policies** in the IAM console if none of the AWS-managed policies meet your needs.
* **Group Management**: You can view, modify, or delete groups in the **User Groups** section of the IAM console.
* **Access Monitoring**: Use AWS CloudTrail to monitor access and actions taken by users within groups.

**Explain these topics: Load Balancing , Auto scaling , Lambda**

**1. Load Balancing**

**Load Balancing** is a method used to distribute incoming application traffic across multiple targets (like EC2 instances, containers, or IP addresses) to ensure no single server is overwhelmed. In AWS, **Elastic Load Balancing (ELB)** provides this capability, improving application availability, scalability, and fault tolerance.

**Types of AWS Load Balancers:**

* **Application Load Balancer (ALB)**: Best suited for web applications, it operates at the application layer (Layer 7) and can route requests based on URLs, host headers, and more.
* **Network Load Balancer (NLB)**: Works at the network layer (Layer 4) and is designed for handling millions of requests per second, ideal for low-latency connections.
* **Gateway Load Balancer (GWLB)**: Primarily for network appliance use cases, like firewalls or monitoring, and operates at Layer 3.

**Benefits of Load Balancing:**

* **Improves reliability and availability** by distributing requests across multiple resources.
* **Enhances fault tolerance** by automatically routing traffic to healthy targets.
* **Scales automatically** with traffic demands.

**2. Auto Scaling**

**Auto Scaling** enables automatic adjustments to the number of compute resources based on application demand. In AWS, **Auto Scaling Groups** (ASGs) work with EC2 instances to help applications handle changing traffic levels, allowing resources to scale up when demand increases and scale down when demand decreases.

**Key Components of AWS Auto Scaling:**

* **Launch Configuration/Template**: Defines instance specifications like instance type, AMI, security groups, etc., that new instances will use.
* **Auto Scaling Group (ASG)**: A logical group that manages the collection of EC2 instances.
* **Scaling Policies**: Rules that define when and how the ASG should add or remove instances based on conditions such as CPU usage, network traffic, or custom CloudWatch metrics.

**Benefits of Auto Scaling:**

* **Cost efficiency**: Helps you pay only for the resources you need by scaling down during low-demand periods.
* **Improved performance**: Ensures your application is always responsive by scaling up during peak traffic times.
* **Fault tolerance**: Replaces unhealthy instances automatically to maintain application performance.

**3. Lambda**

**AWS Lambda** is a serverless compute service that allows you to run code without provisioning or managing servers. With Lambda, you upload your code, and AWS automatically runs and scales it in response to events (like HTTP requests or updates in an S3 bucket).

**How AWS Lambda Works:**

* **Event-Driven Execution**: AWS Lambda executes code in response to triggers like API Gateway calls, database events, file uploads to S3, and more.
* **Function as a Service (Favas)**: Code is organized as individual "functions" which are run independently as needed.
* **Pay-as-you-go**: Billing is based on the number of requests and the time it takes to run your code, rather than server uptime.

**Benefits of Lambda:**

* **Cost-effective**: Only pay for the time and resources consumed during code execution.
* **Automatic Scaling**: Lambda automatically scales to meet the demand, handling thousands of requests simultaneously.
* **Focus on Code**: Allows developers to focus solely on the application logic, as Lambda manages infrastructure, security patches, and scaling.

Each of these services helps build scalable, efficient, and resilient applications on AWS: Load Balancing improves request handling, Auto Scaling optimizes resource usage, and Lambda enables flexible, event-driven computing. Together, they support high-performance applications that can scale and respond effectively to varying workloads.