**Session 1: IAM (Identity & Access Management)**

**1. Compare the use of IAM roles vs. IAM users in a multi-account AWS Organization setup. Provide an example scenario where roles are preferred over users.**

IAM **users** are tied to long-term credentials and are suitable for individuals who need consistent access. IAM **roles**, however, provide temporary credentials and are ideal for cross-account access or applications.\ **Example:** In a multi-account setup, a developer in the central account needs access to resources in a production account. Instead of creating a user in each account, you create a role in the production account and allow the developer to assume it via AWS STS. This approach improves security, reduces credential sprawl, and aligns with best practices for least privilege.

**2. Design an IAM policy that enforces read-only access to S3 for a specific user group, but restricts access to a single bucket. Justify how this aligns with the principle of least privilege.**

**Policy Example:**

{

  "Version": "2012-10-17",

  "Statement": [

    {

      "Effect": "Allow",

      "Action": ["s3:GetObject", "s3:ListBucket"],

      "Resource": [

        "arn:aws:s3:::finance-reports",

        "arn:aws:s3:::finance-reports/\*"

      ]

    }

  ]

}

This policy grants only read permissions for the finance-reports bucket, ensuring users cannot modify or delete data. It aligns with **least privilege** by limiting access to necessary actions and resources only.

**3. Explain the security risks of using IAM root credentials. Propose a mitigation strategy using IAM policies, Service Control Policies (SCPs), and MFA.**

Root credentials have unrestricted access, making them a prime target for attackers. Risks include accidental deletion of resources and compliance violations.\ **Mitigation:**

* Enable **MFA** on the root account.
* Create **IAM users and roles** for daily tasks.
* Apply **SCPs** in AWS Organizations to restrict high-risk actions.
* Store root credentials securely and use them only for account-level tasks like billing.

**Session 2: Compute Service (EC2)**

**4. Your company runs a high-traffic web app. Compare using EC2 Auto Scaling Groups vs. Serverless (Lambda + API Gateway) for cost, scalability, and operational management.**

* **EC2 Auto Scaling:** Offers predictable performance and control over OS and environment. Ideal for stateful apps but requires patching and scaling configuration. Costs can rise during peak traffic.
* **Serverless:** Scales automatically, charges only for execution time, and eliminates server management. Best for event-driven workloads. However, cold starts and execution limits may affect performance. **Decision:** For unpredictable traffic and microservices, serverless is cost-efficient. For legacy or compute-heavy apps, EC2 Auto Scaling is better.

**5. An EC2 instance is hosting a financial reporting application. Explain how you would configure IAM instance profiles to allow access to S3 securely without storing credentials on the server.**

Attach an **IAM role** with an S3 access policy to the EC2 instance via an **instance profile**. This way, the application uses temporary credentials provided by AWS STS automatically. No hardcoding or manual credential storage is needed, reducing security risks and ensuring compliance.