**AWS IAM Roles - Detailed Notes**

**Table of Contents**

1. **Introduction to IAM Roles**
   * Purpose of Roles
   * Difference Between Users & Roles
2. **Key Concepts**
   * Temporary vs. Permanent Credentials
   * Trust Relationships
   * AssumeRole Mechanism
3. **Practical Implementation**
   * **Task 1:** Granting Access to a Specific S3 Bucket
   * **Task 2:** Assigning a Role to a User (AssumeRole)
4. **Real-World Use Cases**
5. **Best Practices & Interview Tips**
6. **Conclusion**

**1. Introduction to IAM Roles**

**Purpose of IAM Roles**

* **Roles** define **temporary permissions** for AWS services/users without long-term credentials.
* Unlike **users**, roles **do not have passwords or access keys** (only temporary security tokens).
* Used for:
  + **AWS services** (e.g., EC2 accessing S3).
  + **Cross-account access**.
  + **Federated users** (e.g., SSO via Google/Azure AD).

**Difference Between Users & Roles**

| **Feature** | **IAM User** | **IAM Role** |
| --- | --- | --- |
| **Credentials** | Permanent (username/password, keys) | Temporary (STS tokens) |
| **Usage** | Humans/Applications | AWS Services/Federated Identities |
| **Attach Policy** | Directly or via Groups | Via Trust Policy |

**2. Key Concepts**

**A. Temporary vs. Permanent Credentials**

* **Roles use STS (Security Token Service)** to generate **short-lived credentials**.
* **Metadata URL**:

bash

curl http://169.254.169.254/latest/meta-data/iam/security-credentials/ROLE\_NAME

* + Fetches temporary keys for the attached role.

**B. Trust Relationships**

* Defines **who can assume the role** (e.g., EC2, Lambda, specific users).
* **Example Trust Policy**:

json

{

"Version": "2012-10-17",

"Statement": [{

"Effect": "Allow",

"Principal": {“Service": "ec2.amazonaws.com" },

"Action": "sts:AssumeRole"

}]

}

**C. AssumeRole Mechanism**

* Allows a **user/service to "borrow" permissions** from a role.
* **Workflow**:
  1. User calls sts:AssumeRole.
  2. AWS returns **temporary credentials**.
  3. Credentials expire after 1 hour (default).

**3. Practical Implementation**

**Task 1: Grant Access to a Specific S3 Bucket**

**Scenario:** Allow an EC2 instance to access **only one S3 bucket**.

**Steps:**

1. **Create an IAM Role**:
   * Go to **IAM Console → Roles → Create Role**.
   * Select **AWS Service → EC2**.
   * Name: s3-bucket-access-role.
2. **Attach an Inline Policy**:

json

Copy

Download

{

"Version": "2012-10-17",

"Statement": [{

"Effect": "Allow",

"Action": "s3:\*",

"Resource": [

"arn:aws:s3:::flow-logs-236",

"arn:aws:s3:::flow-logs-236/\*"

]

}]

}

* + Replace flow-logs-236 with your bucket name.

1. **Attach Role to EC2**:
   * Go to **EC2 → Instance → Actions → Security → Modify IAM Role**.
   * Select s3-bucket-access-role.
2. **Test Access**:

bash

Copy

Download

*# Success (allowed bucket)*

aws s3 cp test.txt s3://flow-logs-236/

*# Failure (other bucket)*

aws s3 cp test.txt s3://another-bucket/

**Task 2: Assign a Role to a User (AssumeRole)**

**Scenario:** Let a user **temporarily assume a role** with elevated permissions.

**Steps:**

1. **Create a User**:
   * Name: temp-user (no policies attached).
2. **Create a Policy for AssumeRole**:

json

{

"Version": "2012-10-17",

"Statement": [{

"Effect": "Allow",

"Action": "sts:AssumeRole",

"Resource": "arn:aws:iam::ACCOUNT\_ID:role/s3-bucket-access-role"

}]

}

* + Attach to temp-user.

1. **Update Role’s Trust Policy**:

json

{

"Version": "2012-10-17",

"Statement": [{

"Effect": "Allow",

"Principal": { "AWS": "arn:aws:iam::ACCOUNT\_ID:user/temp-user" },

"Action": "sts:AssumeRole"

}]

}

1. **Assume the Role**:

bash

*# Fetch temporary credentials*

aws sts assume-role --role-arn arn:aws:iam::ACCOUNT\_ID:role/s3-bucket-access-role --role-session-name test-session

*# Export credentials*

export AWS\_ACCESS\_KEY\_ID="TEMP\_ACCESS\_KEY"

export AWS\_SECRET\_ACCESS\_KEY="TEMP\_SECRET\_KEY"

export AWS\_SESSION\_TOKEN="TEMP\_SESSION\_TOKEN"

*# Verify access*

aws s3 ls s3://flow-logs-236

**4. Real-World Use Cases**

1. **DevOps Temporary Access**
   * Developers assume a **read-only role** for debugging.
   * DevOps engineers assume **admin roles** for deployments.
2. **Cross-Account Access**
   * Role in **Account A** grants **Account B** access to its S3 buckets.
3. **Lambda Permissions**
   * Lambda function assumes a role to interact with DynamoDB.

**5. Best Practices & Interview Tips**

* **Least Privilege:** Grant minimal permissions in roles.
* **Use Temporary Credentials:** Avoid long-term keys for roles.
* **Monitor with CloudTrail:** Track AssumeRole calls.
* **Interview Question:**
  + *"How do you restrict a role to a specific S3 bucket?"*
  + **Answer:** Use a policy with the bucket ARN in Resource.

**6. Conclusion**

* **Roles** are secure, temporary, and scalable for AWS access.
* **AssumeRole** enables flexible permission management.
* Always test in a sandbox before production!

**Next Session:** Cross-account IAM roles.