✅ AWS IAM — Complete Session Guide (Basics → Advanced)

"IAM is the heart of AWS security. Every request, every service, every user → checked by IAM."

1️⃣ What is IAM?

IAM (Identity and Access Management) is a global AWS security service used to **identify who can access what** in your AWS account.

Simple Definition

IAM controls **WHO** (Identity) can do **WHAT** (Permissions) on **WHICH resource** (EC2, S3, RDS, etc.).

2️⃣ Why IAM is Used?

Purpose	Explanation
Security	Protect AWS resources from unauthorized access
Least Privilege	Give only required permissions, nothing more
Centralized Access Control	Manage all users/apps access from one place
Auditing & Compliance	Tracks who did what (CloudTrail)
Cross-Account Access	Share resources with other AWS accounts securely
Temporary Access	For developers, applications, or services

3️⃣ How IAM Works (Internals Explained)

IAM verifies access using:

✔️ **Identity** → User / Group / Role
✔️ **Policy** → JSON document with allow/deny

✔ **Resource** → AWS service object (S3 bucket, EC2 instance)
✔ **Request** → Action you want to perform

IAM Decision Process:

```
1. Request comes to AWS → "Can User do Action on Resource?"
2. Check Default Rule → Deny
3. Check Explicit Deny → Deny
4. Check Allow → If allowed, access granted
5. Else → Deny
```

4⃣ IAM Components (Most Important Topic)

✅ 1. IAM User

Human identity (developer, admin, tester).
User logs in with **username + password** or **Access keys** (CLI).

Best Practice: Never give admin access to users.

✅ 2. IAM Group

Collection of users.
Permissions applied to group propagate to all users.

Examples:

- Developers
- Admin
- Finance
- DevOps

✅ 3. IAM Role (Interview-favorite topic)

A role is a **temporary identity** for AWS services or external users.

Used by:

- EC2 instances (to access S3, DynamoDB)
- Lambda functions
- Applications
- AWS Cross-Account Access
- Federated logins (SSO, Google/Microsoft)

Difference between User vs Role:

User	Role
Long-term credentials	Temporary credentials
For humans	For services/apps
Password & Access keys	STS Tokens

✅ 4. IAM Policies

JSON permission documents that define **Allow/Deny**.

Example Policy: Allow read to S3 bucket

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::mybucket/*"
  }]
}
```

Types of policies:

✓ **Managed (AWS created)**

✓ **Customer Managed (You create)**

✓ **Inline Policies (Attached directly to user/role)**

---

5 □ IAM Security Best Practices

Practice	Why
Enable MFA	Protect login
Never use Root account	Too much power, unsafe
Use Roles not Access Keys	Best for automation

Practice	Why
Rotate Access/Secret keys	Security compliance
Use Groups to assign permissions	Easier management
Use Least Privilege	Minimize attack

6 □IAM in Cloud & DevOps (Real-Time Usage)

1. EC2 → S3 Access using IAM Role

DevOps uses IAM Role for EC2 so apps inside EC2 can:

- Upload logs to S3
- Pull code from CodeCommit
- Access DynamoDB

---

2. Jenkins → AWS Deployment using IAM

Jenkins uses IAM user with limited access to deploy:

- ECS tasks
- Lambda functions
- EKS deployments
- CloudFormation stacks

---

3. Kubernetes (EKS) IAM Integration

IAM roles mapped to Kubernetes service accounts → secure pod access.

---

4. Terraform → AWS

Terraform needs IAM user access key to:

- create VPC
- launch EC2
- create S3 bucket

5. Serverless + Lambda

Lambda always requires IAM execution role.

7 □IAM Step-by-Step Practical Tasks (Today's Session Lab)

⭐ Task 1: Create IAM User

Steps:

1. Open IAM Console
2. Users → Create User
3. Give name `developer-user`
4. Don't give password unless needed
5. Add to group → DeveloperGroup
6. Attach policy → `AmazonS3ReadOnlyAccess`

⭐ Task 2: Create IAM Group

1. IAM → Groups → Create
2. Create `DevOpsGroup`
3. Attach:
   - `AmazonEC2FullAccess`
   - `IAMReadOnlyAccess`
4. Add users

⭐ Task 3: Create IAM Role for EC2

1. IAM → Roles → Create role
2. Choose **EC2**
3. Attach policy: `AmazonS3FullAccess`
4. Launch an EC2 instance → attach role
5. Test inside EC2:

```
aws s3 ls
```

⭐ Task 4: Create Custom IAM Policy

Use this sample policy:

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Action": [
        "ec2:StartInstances",
        "ec2:StopInstances"
      ],
      "Resource": "*"
  }]
}
```

Attach to user.

---

8 □IAM Architecture Diagram (Text Version)

```
               +----------------------------+
               |      IAM Console           |
               +----------------------------+
                          |
        ------------------------------------------------
        |                |                 |           |
     Users             Groups            Roles        Policies
        |                |                 |           |
        +----------------+-----------------+-----------+
                          |
                AWS Resources (EC2, S3, RDS, Lambda)
```

---

9 □IAM Interview Questions (Frequently Asked)

⭐ What is IAM?

IAM is a global service to manage access to AWS resources.

⭐ What is Difference Between User & Role?

- User → Human, long-term credentials
- Role → Services, temporary credentials

⭐ What is Least Privilege?

Giving only required permissions.

⭐ What is IAM Policy?

JSON-based allow/deny document.

⭐ What is IAM STS?

AWS Security Token Service → gives temporary credentials.

⭐ What is Access Key?

Used for AWS CLI/SDK.

⭐ When to use IAM Role?

When EC2/Lambda/EKS need AWS access without keys.

🔟 Advanced IAM Topics (For Senior DevOps)

🔷 IAM Permission Boundaries

Restrict maximum permissions for users/roles.

🔷 SCP (Service Control Policies) in AWS Organizations

Control whole account.

🔷 IAM Conditions

Time-based access
IP-based access
Tag-based access

🔷 Identity Federation

Login using:

- Google
- Microsoft AD
- SAML
- AWS SSO

---

1 1⃣ ⃣ Summary (What You Learned Today)

✅ What IAM is
✅ Why IAM is used
✅ How IAM works
✅ Users, Groups, Roles, Policies
✅ Real-time DevOps use cases
✅ Step-by-step labs
✅ Interview questions
✅ Advanced IAM topics