AWS IAM – Full Notes

🔹 What is AWS IAM?

AWS Identity and Access Management (IAM) is a **web service** that enables you to securely control **access to AWS resources**.

It allows you to:

- Manage **users, groups, roles, and policies**
- Define **who can do what** on which AWS resources
- Apply **fine-grained permissions** using policies

IAM is **global** — not region-specific.

🔹 Core IAM Concepts

Component	Description
User	An entity representing a person or application needing access to AWS
Group	A collection of IAM users (useful for assigning permissions in bulk)
Role	A set of permissions that can be assumed by users, services, or apps
Policy	A JSON document that defines permissions (allow/deny)
Principal	Any entity that can make an AWS request (user, role, federated user)
Authentication	Verifying identity (username, password, MFA)
Authorization	Granting or denying access based on policies

🔹 Types of IAM Policies

Policy Type	Applied To	Description
Managed Policy (AWS)	Users, Groups, Roles	Predefined by AWS
Customer Managed Policy	Users, Groups, Roles	Created by you for specific needs
Inline Policy	Embedded in a single user, group, or role	Use when specific to one identity
Permissions Boundary	Roles or users	Limit maximum permissions that can be granted

Policy Type	Applied To	Description
Service Control Policies (SCPs)	AWS Organizations	Restrict what accounts within an org can do

---

🔹 IAM Policy Structure

A policy is a JSON document with the following syntax:

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow" | "Deny",
   "Action": "s3:*",
   "Resource": "*"
  }
 ]
}
```

Fields:

- Effect: Allow or Deny
- Action: The specific API calls (e.g., s3:PutObject)
- Resource: The ARN of the resource
- Condition *(optional)*: Add logic (e.g., IP restriction, MFA required)

🔹 IAM Roles

IAM **roles** are used to delegate access to:

- AWS services (e.g., Lambda, EC2)
- Users from another AWS account
- Federated users (SSO, Google Workspace, etc.)
- Applications and containers

Roles use temporary security credentials provided by AWS STS.

✅ Example: An EC2 instance with a role to read/write to an S3 bucket.

🔹 IAM Best Practices

✅ **Enable MFA** on root and privileged accounts
✅ **Use IAM roles** for EC2, Lambda, and services (not access keys)
✅ **Grant least privilege** — only the permissions required
✅ Use **groups** to manage permissions at scale
✅ Rotate **access keys** regularly or avoid them altogether
✅ Use **IAM Access Analyzer** to detect unintended public access
✅ Use **policy conditions** to restrict access (IP, time, MFA)
✅ **Avoid using the root user** for daily tasks
✅ Use **service-linked roles** for trusted AWS services
✅ Monitor with **CloudTrail and IAM Access Advisor**

🔹 Multi-Factor Authentication (MFA)

Adds an extra layer of security.

- **Supported for**: IAM users, root account
- **Devices**: Virtual MFA apps (e.g., Google Authenticator), hardware MFA
- Use **MFA conditions** in policies to enforce secure access.

Example condition in a policy:

```
"Condition": {
  "Bool": {
    "aws:MultiFactorAuthPresent": "true"
  }
}
```

🔹 IAM Access Analyzer

- Analyzes IAM policies and access to resources (e.g., S3, IAM roles, KMS keys)
- Identifies:
    - Unintended public access
    - Cross-account access
- Use in **security audits** or continuous monitoring

🔹 Credential Types in IAM

Credential	Description
Username/Password	Used for AWS Management Console access
Access Keys (ID + Secret)	Used for AWS CLI/SDK
X.509 Certificates	For legacy use with AWS APIs
Session Tokens (STS)	Temporary credentials when assuming roles

---

🔹 **IAM Role Use Cases**

Use Case	IAM Role Behavior
EC2 instance to access S3	Attach IAM role to EC2 with s3:* policy
Lambda function access DynamoDB	Assign a role with dynamodb:* permissions
Cross-account access	Role in Account A assumed by Account B
Federated access (SSO)	External user assumes IAM role via SAML

🔹 **IAM and AWS Organizations**

- Use **Service Control Policies (SCPs)** to restrict what **accounts** in an org can do.
- SCPs apply **account-wide**, not to individual users/roles.
- They **don’t grant permissions** — only restrict.

🔹 **IAM Troubleshooting Tips**

Issue	Possible Cause
Access Denied error	Missing or incorrect permission in policy
Role cannot be assumed	Trust policy is misconfigured
MFA required but denied access	MFA condition missing or failed
Cross-account access fails	Role’s trust policy doesn't allow the other account
User can't see expected resource	Resource-level permissions missing

---

🔹 **Monitoring and Auditing**

Tool	Description
CloudTrail	Logs all IAM actions for auditing
IAM Access Advisor	Shows permissions last used by a user/role
AWS Config	Tracks changes to IAM resources
Access Analyzer	Helps detect unintended access

🔹 **IAM Limits (2024)**

Resource	Default Limit
Users per account	5,000
Groups per account	300
Roles per account	1,000
Inline policies	10 per identity
Managed policies per identity	10
Max policy size	6,144 characters

Limits can be increased by contacting AWS Support (in some cases).

🔹 **IAM Sample Policy Examples**

1. Allow full S3 access:

```
{
  "Effect": "Allow",
  "Action": "s3:*",
  "Resource": "*"
}
```

2. Deny S3 access unless MFA is used:

```
{
  "Effect": "Deny",
  "Action": "s3:*",
  "Resource": "*",
  "Condition": {
```

```
    "BoolIfExists": {
      "aws:MultiFactorAuthPresent": "false"
    }
  }
}
```

3. Allow only EC2 read-only:

```
{
  "Effect": "Allow",
  "Action": [
    "ec2:DescribeInstances",
    "ec2:DescribeVolumes"
  ],
  "Resource": "*"
}
```

🔹 **IAM vs Other Access Services**

Feature	IAM	Cognito	AWS SSO
Use case	AWS resource access	User identity for apps	Centralized login for AWS
Identity type	Users, roles	Federated or local	Federated enterprise users
Federation support	✅ Yes	✅ Yes	✅ Yes
UI access	Console/CLI/SDK	Web/mobile apps	AWS console/CLI

🔹 **IAM Summary**

Feature	IAM Supports
MFA	✅ Yes
Federation	✅ Yes
Temporary creds	✅ STS
Fine-grained perms	✅ Yes

Feature	IAM Supports
Audit & logging	✅ CloudTrail, Access Analyzer
Resource-based policies	✅ On some services (S3, Lambda, etc.)