AWS Shared Responsibility Model
&
IAM (Identity Access Management)

1. What is the Shared Responsibility Model?

AWS and its customers share security and compliance responsibilities.

✅ AWS: 'Security of the cloud' (hardware, networking, facilities)

✅ Customer: 'Security in the cloud' (data, apps, OS, IAM)

2. Service Responsibility Spectrum

- IaaS (e.g., EC2): AWS manages infra; you manage OS, apps, data.
- Managed Services (e.g., RDS): AWS manages infra and platform; you secure your data & users.
- Serverless (e.g., S3, Lambda): AWS manages almost everything; you manage your logic & access.

3. Why It Matters

🎯 Understanding who secures what prevents misconfigurations.

⚠️ Misunderstandings can lead to:

- Data leaks

- Improper access control

- Compliance violations

4. Best Practices for Customers

- Encrypt data at rest and in transit
- Use IAM with least privilege
- Enable CloudTrail and AWS Config
- Regularly patch OS and apps
- Audit and monitor activity

AWS CloudTrail

Purpose:
Tracks *who* did *what* in your AWS account.

Key Features:

Records all **API calls** made in your AWS account (via Console, CLI, SDKs, etc.)

Helps in **security auditing**, **compliance monitoring**, and **operational troubleshooting**

Stores logs in **S3**, and optionally sends to **CloudWatch**

Example Use Case:
Detect if an IAM user deleted a resource or changed a security group.

AWS Config

Purpose:
Tracks *what* your AWS resources look like and how they've changed over time.

Key Features:

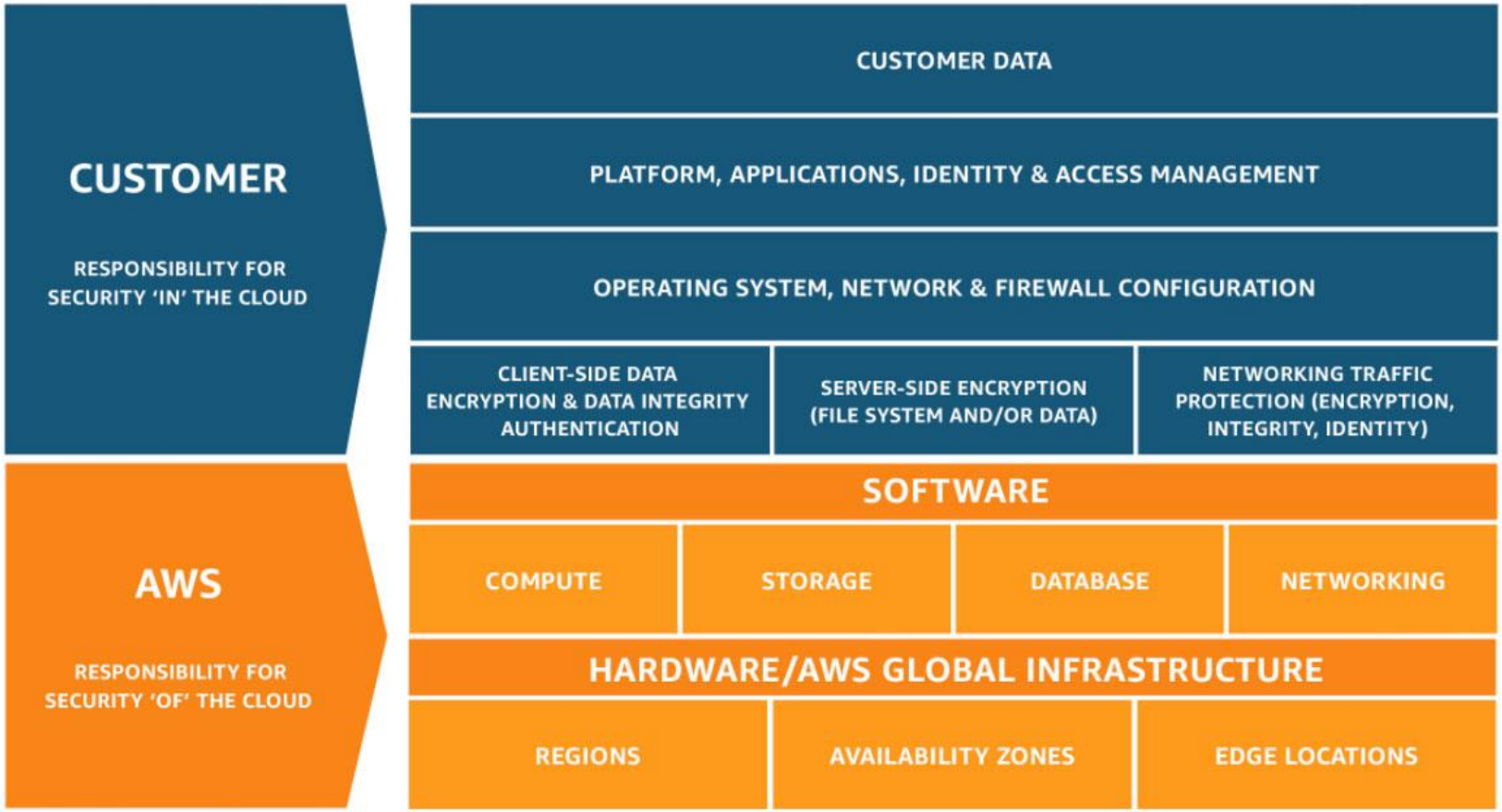
Continuously records **configuration changes** of AWS resources (like EC2, S3, IAM)

Evaluates resource compliance against **custom rules or best practices**

Helps in **audit** and **compliance reporting**

Example Use Case:
Check if all your EC2 instances are using encrypted EBS volumes.

5. AWS Shared Responsibility Model



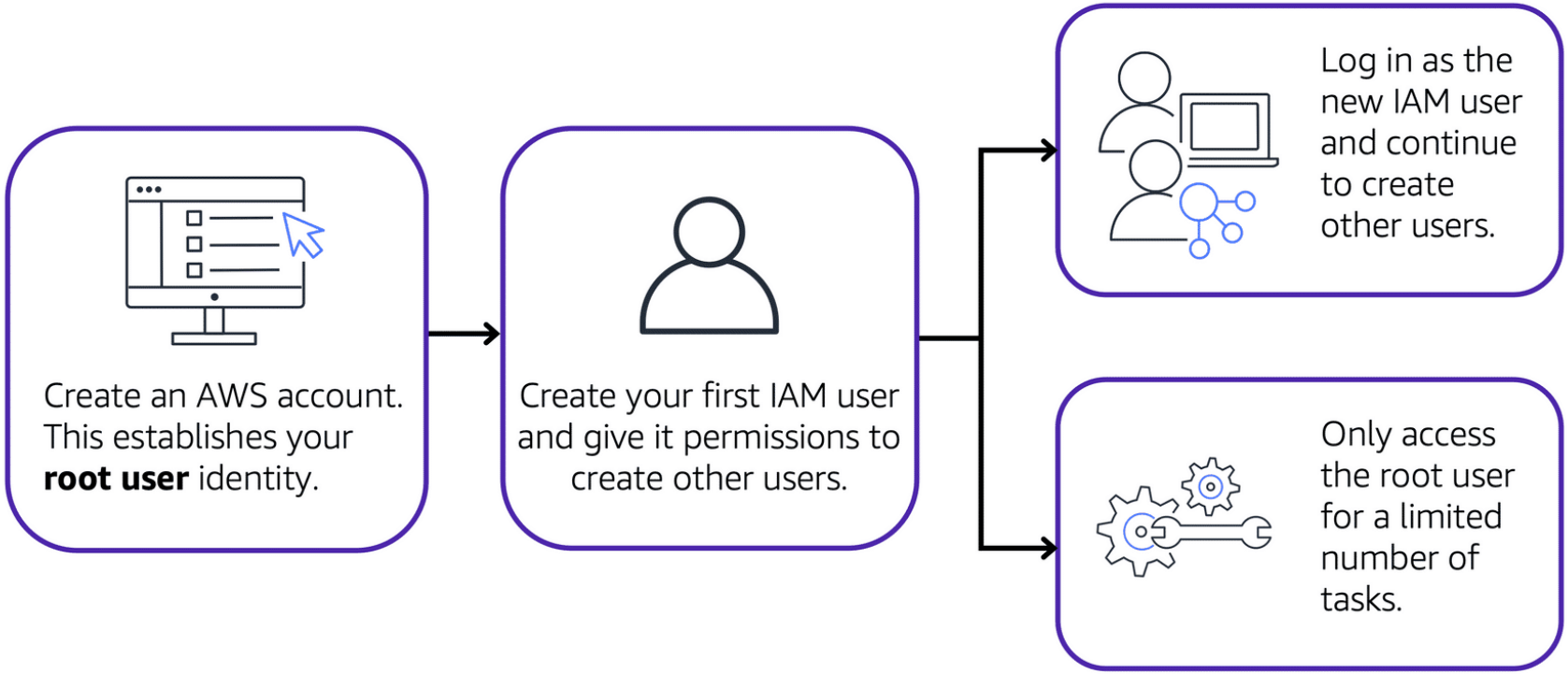
IAM and Its Components

Component	Purpose & Best Practice
Root User	Created with your AWS account; has full access. Best practice: **do not use** for daily tasks — use only for account setup.
IAM Users	Individual identities (people or apps) with unique credentials. Best practice: **create one per person/app** .
IAM Policies	JSON documents defining permissions (allow/deny). Attach to users, groups, or roles to grant precise access .
IAM Groups	Collections of users that share policies—makes permission management efficient .
IAM Roles	Like users but **temporary and shareable**—ideal for granting temporary access to AWS services or external entities .
MFA (Multi-Factor Authentication)	Adds extra security via a code (SMS/app) — highly recommended for root and privileged IAM users .

- IAM is Amazon's service for managing users, groups, roles, and permissions securely.
- Enables precise control over who can access what within your AWS environment

IAM Best Practices

- ✅ **Avoid using the root user** except for critical tasks.
- ✅ **Follow the principle of least privilege**—only grant necessary permissions.
- ✅ Use **groups** to simplify permission assignment.
- ✅ Use **roles** for temporary or cross-account access (e.g., EC2 instances assuming roles).
- ✅ Enable **MFA** for all sensitive accounts.
- ✅ Regularly **review policies and access logs** to detect risk.



Source: Amazon Web Services

IAM Users

- IAM user represents an entity (person or an application) that interacts with AWS resources and services.
- IAM user is made of credentials and a name.
- It is created without permissions by default.
- The root user can grant permissions to the IAM user.
- It is recommended that you create one IAM user for each individual.

✅ Who Can Create IAM Users in AWS?

1. **Root User**

- ✅ **Yes** – The **root user** has **unrestricted access** to the entire AWS account, including:
  - Creating IAM users
  - Deleting users
  - Managing policies, billing, and services

> **Best practice**: Use the root user only for initial setup. Then, lock it down with MFA and avoid using it regularly.

2. **IAM Users**

- ✅ **Can create other IAM users, only if** they have the **necessary permissions**.

> IAM users **do not** have full access by default. You must **explicitly grant permissions** using IAM **policies**.

🔐 Example Policy to Allow IAM User Creation

To allow an IAM user to create other IAM users, they need permissions like:

```
{
  "Effect": "Allow",
  "Action": [
    "iam:CreateUser",
    "iam:PutUserPolicy",
    "iam:AttachUserPolicy",
    "iam:CreateAccessKey"
  ],
  "Resource": "*"
}
```

🛡️ Best Practice

- Only **trusted IAM users** (e.g., cloud admins) should be allowed to create/manage other users.
- Always use **least privilege**—only grant what's necessary.
- Use **MFA** for IAM users with elevated permissions.

🛡️ What is an IAM Role?

An **IAM Role** is a set of **permissions** that define what actions are allowed or denied for an **AWS service**, **user**, or **application**.

Unlike an IAM **user**, a role **does not have long-term credentials** like a username or password.

Feature	IAM User	IAM Role
Long-term credentials	✅ Yes (password, access keys)	❌ No (uses temporary credentials)
Used by	Human users	AWS services, applications
Shared across accounts	❌ Typically no	✅ Yes (cross-account access)
MFA supported	✅ Yes	✅ Yes

🛡️ IAM Policy vs IAM Role

Feature	IAM Policy	IAM Role
Definition	A **document** (usually in JSON) that defines **permissions** – what actions are allowed or denied on specific AWS resources.	An **identity** that can be **assumed temporarily** by users, services, or other AWS accounts to gain permissions.
Purpose	Grants **permissions** to IAM identities (users, roles, groups).	Grants **temporary access** to resources by assuming a role with attached policies.
Who uses it?	IAM Users, Groups, and Roles	AWS Services (e.g., EC2, Lambda), IAM Users, External identities (SSO, cross-account)
Credentials	Does not provide credentials. It defines access **rules only**.	Provides **temporary credentials** when assumed.
Attachable to	Users, Groups, or Roles	Not attached to others; **roles have policies attached to them**.
Example Use	Allow user to access S3 bucket or start EC2 instances.	Allow EC2 instance to access S3 bucket or allow cross-account access.

🏦 Real-Time Scenario: IAM in a Bank

🔒 Background

A large commercial **bank** uses AWS cloud infrastructure to host its **core banking systems**, **loan processing applications**, **customer portals**, and **internal tools**. The bank needs to ensure that **only the right people can access the right resources**—nothing more, nothing less.

👤 Roles in the Bank

Each team or department needs specific access:

- **Bank Tellers**: Can access customer accounts, view balances, and update contact info.
- **Loan Officers**: Can only access loan application data—not customer accounts.
- **Auditors**: Need read-only access to logs, transactions, and historical data.
- **IT Admins**: Have full control of infrastructure (EC2, S3, RDS).
- **Developers**: Can deploy code but **cannot** access live customer data.
- **Branch Managers**: Can see regional performance dashboards.

✅ How IAM Helps

IAM Component	Usage in the Bank
IAM Users	Each employee is assigned a unique identity (e.g., john.doe@bank.com).
IAM Groups	Users are grouped by role (e.g., `Tellers`, `LoanOfficers`, `Admins`).
IAM Policies	JSON-based rules define what actions are allowed on which resources.
IAM Roles	Auditors and external vendors assume temporary roles with limited access.
MFA (Multi-Factor Authentication)	Required for Admins and Managers for higher security.

Thank You