Securing Access to Cloud Resources

SHARED RESPONSIBILITY · IAM USERS, GROUPS & ROLES · AUTHENTICATION · AUTHORIZATION · POLICIES · TEMPORARY CREDENTIALS

STUDY & REVIEW

01 MUST KNOW

if you remember five things, remember these

01 **Split the responsibility:** security and compliance are **shared**. AWS is responsible for security *of* the cloud — the hardware, software, facilities, and the global infrastructure (Regions, Availability Zones, edge locations). The customer is responsible for security *in* the cloud — guest OS and patches, application security, firewall and security group configuration, IAM, and their data.

02 **Authenticate, then authorize: authentication** establishes *who* is making the request (IAM calls a person or application a **principal**). **authorization** decides *what* they may do — IAM evaluates the relevant policies to allow or deny. A principal has **no access** to any resource until permissions are explicitly granted.

03 **Four IAM building blocks:** an **IAM user** represents a person or application and has long-term credentials; an **IAM group** collects users so they share one set of permissions; an **IAM role** has *no* long-term credentials and is assumed to receive temporary ones; an **IAM policy** is a JSON document that lists permissions. Best practice: attach policies to groups, then add users to the groups.

04 **Prefer roles and temporary credentials:** IAM roles issue **temporary security credentials** and are preferred for apps on Amazon EC2, mobile apps, cross-account access, and identity federation. **Never** store long-term access keys in code, on an AMI, or in a public place — launch the instance with a role instead, so there is nothing to store or rotate.

05 **Deny always wins; grant least privilege:** policy evaluation is **default deny** → an explicit *allow* overrides the default deny → an explicit *deny* overrides any allow. Evaluation order does not matter; on conflict the most restrictive rule applies. Follow the **principle of least privilege**: start with the minimum permissions the job needs and add more only as required.

02 KEY TERMS

TERM	DEFINITION
Shared responsibility model	The division of security duties: AWS secures *of* the cloud (infrastructure), the customer secures *in* the cloud (their data, OS, apps, and configuration)
Principal	A person or application whose identity IAM authenticates when it requests access to an AWS account and its resources
Authentication	Establishing the identity of the requester through credentials — confirming they are who they claim to be
Authorization	Determining what an authenticated principal is allowed to do, by checking the policies relevant to the request
IAM user	An entity created in AWS to represent a person or application; given a permanent set of credentials until a forced rotation
IAM group	A collection of IAM users, used to grant the same set of permissions to multiple users at once
IAM role	An AWS identity with attached permission policies but no long-term credentials; a principal assumes it to get temporary credentials
IAM policy	A JSON document that explicitly lists permissions (effect, actions, resources, optional conditions); attach to a user, group, role, or resource
Access key	The credential pair for programmatic access — an access key ID plus a secret access key — used by the CLI, SDKs, and APIs
MFA	Multi-factor authentication; an extra factor (hardware or virtual device code) on top of the user name and password

TERM	DEFINITION
AWS STS	AWS Security Token Service — the service that issues the temporary security credentials returned when a role is assumed
Principle of least privilege	Grant only the permissions needed to perform a task; start minimal and add access as needed rather than starting broad

03 SHARED RESPONSIBILITY MODEL

who secures what

AWS — SECURITY OF THE CLOUD

the infrastructure that runs all AWS services

- Hardware and the AWS Global Infrastructure — Regions, Availability Zones, edge locations
- Host OS and virtualization (hypervisor) layer
- Physical security of the facilities
- Software, compute, storage, databases, networking

CUSTOMER — SECURITY IN THE CLOUD

everything you build on and connect to AWS

- Customer data; platform, applications, and IAM
- Guest OS, network, and firewall configuration
- Security group configuration; account management
- Client-side and server-side encryption

04 HOW AN IAM ROLE IS ASSUMED

delegate access without long-term keys

01 A **principal** (IAM user, application, or AWS service) requests to assume a role → 02 **AWS STS** issues temporary security credentials for the role session → 03 The principal's prior permissions are temporarily set aside; it now has the role's permissions → 04 It uses the temporary credentials to make programmatic requests to AWS → 05 The credentials **expire** — nothing long-term to store or rotate

05 IDENTITY-BASED VS. RESOURCE-BASED POLICIES

the difference is where the policy is attached

ASPECT	IDENTITY-BASED POLICY	RESOURCE-BASED POLICY
Attached to	An IAM user, group, or role	A resource, such as an S3 bucket
Answers	What does this identity have access to?	Who has access to this resource?
Example use	Let a user access a DynamoDB table	Grant an S3 bucket cross-account access between trusted accounts

06 MANAGED VS. INLINE POLICIES

reusable and versioned, or embedded 1:1

ASPECT	MANAGED POLICY	INLINE POLICY
What it is	Standalone, identity-based; AWS-managed or customer-managed	Embedded inside a single principal (user, group, or role)
Relationship	Attach the same policy to many users, groups, and roles	Strict 1:1 relationship with its one entity
Management	Reusable · central change management · versioning and rollback · delegation	Each entity keeps its own copy; cannot be centrally managed

07 EXAM TRAPS

where the knowledge check gets you

- × “The customer just uses AWS, so AWS handles security.” — AWS secures *of* the cloud (hardware, facilities, global infrastructure); the **customer** is responsible *in* the cloud — guest OS and patches, application security, firewall and security group configuration, IAM, and their data.
- × “A brand-new IAM user can do things until you restrict it.” — Backwards. Users have **no access** to AWS resources until permissions are explicitly granted; anything not explicitly allowed is denied by default.
- × “An IAM role has its own password or access keys.” — A role has **no long-term credentials**. When a principal assumes it, AWS STS returns *temporary* security credentials for that session.
- × “An explicit allow can override an explicit deny.” — Never. An explicit **deny** always overrides any allow. Evaluation order has no effect; on conflict the most restrictive rule wins.
- × “Store the access key in a file on the AMI so the instances can reach DynamoDB.” — Insecure. Create an **IAM role** with permissions to the table and launch the instances with the role — no keys to embed or rotate.
- × “Inline policies are reusable and centrally managed.” — No. Inline policies are embedded 1:1 in a single entity and each keeps its own copy. **Managed** policies are the reusable, versioned, delegable kind.
- × “Identity-based and resource-based policies are the same.” — They differ by attachment point: identity-based attaches to a user, group, or role (what that identity can do); **resource-based** attaches to a resource (who can access it) and enables cross-account access.
- × “Grant broad permissions now and tighten later.” — Least privilege says the opposite: start with the **minimum** permissions the job role needs and grant additional access only as required.

08 SELF-QUIZ

answers are upside-down below

Q1 In the shared responsibility model, who is responsible for security OF the cloud, and who for security IN the cloud?

Q2 IAM uses one term for the person or application that requests access. What is it?

Q3 Authentication answers one question and authorization answers another. What does each answer?

Q4 Which IAM entity has no long-term credentials and provides temporary ones when it is assumed?

Q5 Which AWS service issues the temporary security credentials returned when a role is assumed?

Q6 What is the default outcome for any request that is not explicitly allowed by a policy?

Q7 When an allow statement and a deny statement conflict, which one is applied?

Q8 For long-term access, what is the recommended way to organize policies, users, and permissions?

Q9 Which policy type is attached to a resource and can grant cross-account access?

Q10 Instances built from a custom AMI need to read DynamoDB, and the access key is stored in a text file on the AMI. What is the more secure solution?

ANSWER KEY (rotate the page)

Q1 AWS is responsible OF the cloud; the customer is responsible IN the cloud **Q2** a principal **Q3** authentication = who is requesting access; authorization = what they are allowed to do **Q4** an IAM role **Q5** AWS Security Token Service (AWS STS) **Q6** it is denied (default deny) **Q7** the explicit deny — it overrides any allow **Q8** attach policies to IAM groups, then add users to those groups so they inherit the permissions **Q9** a resource-based policy **Q10** create an IAM role with permissions to the table and launch all instances with the role