Intro to Amazon Web Services

Cloud Computing

Brenden west

Contents

Learning Outcomes

- What is Amazon Web Services (AWS)
- AWS shared responsibility model
- AWS Identity & Access (IAM)
- Core AWS services
- Ways to access AWS

Resources

- AWS Cloud Foundations - modules 1, 3, 4
- AWS Cloud Developing - module 2, 3, 4

What is AWS

- A secure cloud platform offering a broad set of services
- Provides on-demand access to virtualized IT resources and management tools
- Provides services on pay-as-you-go model
- Services designed to work together for composing complex systems
- Services are categorized (e.g. compute, storage, database, identity, etc)
- Services are supported by globally distributed & redundant physical infrastructure

AWS Global Infrastructure

- Scalable, fault-tolerant, highly available
- AWS infrastructure is provided in globally distributed **regions**
- Each region is a physical geographic location optimal for proximity to end users
- Regions are isolated from each other, but developers can replicate data across regions
- Each region consists of two or more **Availability Zones** connected by high-speed private networks
- Avallability zones each have own power infrastructure & are physically separated
- Developers can replicate data & resources between zones for resiliency
- Each availability zone consists of one or more physical data centers

AWS Shared Responsibility Model

AWS

- Operate physical infrastructure. Ensure reliability and physical security

Customer

- Choose regions, availability zones
- Replicate data as needed
- Configure appropriate user access

Selecting a Region

- Data governance & legal requirements - e.g. EU data
- Proximity to customers (low latency)
- Service availability
- Regional costs

Core AWS Services

AWS has a wide & sometimes confusing array of services. Some core services a new cloud developer will encounter:

- **Compute** - EC2, ECS, ELB, Lambda
- **Storage** - S3, EBS
- **Networking** - VPC, Route 53, CloudFront
- **Database** - RDS, DynamoDB
- **Identity** - IAM, Artifact, KMS
- **Management** - Config, Console, CloudWatch, CLI
- **Cost Management** - Cost & Usage Report, Cost Explorer, Budgets

Accessing AWS

Before using AWS:

- Create an AWS account (aka root user)
- Create an Identity & Access Management (IAM) user
- Set up IAM user permissions

Interacting with AWS

AWS supports 3 means for interacting with services through a REST-like API:

- Management Console
- Command Line Interface (CLI)
- Software Development Kits (SDKs)

AWS also provides several browser-based tools for using the CLI or SDKs:

- AWS CloudShell - a browser-based terminal for running CLI commands
- AWS Cloud9 - an integrated development environment (IDE)
-

AWS Management Console

- Browser-based UI for managing AWS resources
- Provides access to development tools (e.g. CloudShell, Cloud9)

AWS CloudShell

- Available in the AWS Console
- Inherits credentials from AWS Console
- Runs fully-managed Amazon Linux 2 instance with AWS CLI & SDKs pre-installed
- Has per-region persistent storage
- Pay only for AWS resources you create & run

AWS CLI

- AWS service access through terminal-based commands or scripts
- Can run on Windows, MacOS, Linux operating systems
- Uses AWS credentials stored on client computer

AWS SDKs

- Language or platform-specific libraries for accessing AWS services programmatically (e.g. Python, Java, Android, etc)
- SDK mediates request / response between a client application and AWS services

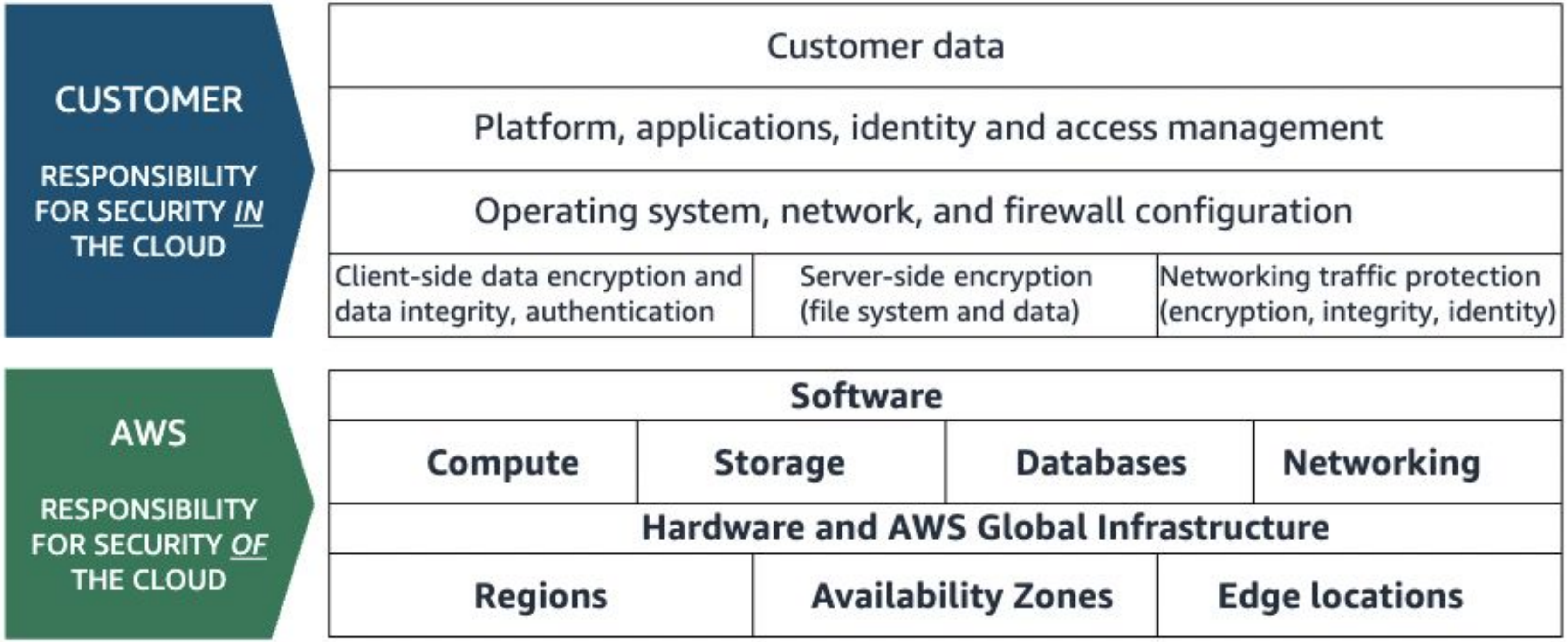
Identity Terminology

- **IAM -** Identity & Access Management
- **Authentication** - mechanism to verify the identity of a user
- **MFA** - multi-factor authentication uses two or more mechanisms to authenticate a user
- **Authorization** - mechanism to verify a user has permission to access a service or resource
-

AWS Shared Responsibility Model

- AWS & the customer share responsibility for security & compliance
- AWS responsible for security **of** the cloud (physical infrastructure)
- Customer responsible for security **in** the cloud
- Customers responsible for what is implemented using AWS products & services. E.g:
    - Managing their data
    - Configuring appropriate security using IAM & other security services
    - Guest operating systems on virtual machines
    - Firewalls & network configurations

AWS Shared Responsibility, cont.


CUSTOMER
RESPONSIBILITY FOR SECURITY *IN* THE CLOUD
Customer data
Platform, applications, identity and access management
Operating system, network, and firewall configuration
Client-side data encryption and data integrity, authentication
Server-side encryption (file system and data)
Networking traffic protection (encryption, integrity, identity)
AWS
RESPONSIBILITY FOR SECURITY *OF* THE CLOUD
Software
Compute
Storage
Databases
Networking
Hardware and AWS Global Infrastructure
Regions
Availability Zones
Edge locations

AWS Identity & Access Management (IAM)

Allows AWS customer to grant unique security credentials to users, roles, and groups.

- Securely controls who can access customer’s AWS resources, what resources they can use, and in what ways
- Integrates with other AWS services
- Supports granular permissions
- Supports federated identity management (via corporate identity providers)
- Supports multi-factor authentication (MFA)

IAM Overview

- **IAM user** - a person or application with permanent credentials to access the services & resources in an AWS account
- **IAM group** - a collection of IAM users with same permissions. Group members inherit permissions attached to the group.
- **IAM role** - an AWS identity with attached permission policies. Does not have long-term credentials
- **IAM policy** - a document that lists explicit permissions. Can be attached to an IAM user, IAM group, or IAM role

Best practice - attach IAM policies to IAM groups and then assign IAM users to these groups.

Authenticating with IAM

- Any interaction with AWS services, whether through management console, AWS CLI, or AWS SDK, requires authentication by providing credentials
- Management console authentication depends on **user name** and **password**
- CLI, SDKs, and APIs depend on **AWS access keys** (access key and secret key)
- Users or services that assume an IAM role are provided with temporary security credentials to use in accessing AWS resources

AWS Credentials File

The AWS CLI client depends on credentials stored in a local text file to interact with AWS accounts.

- File location is **~/.aws/credentials** (Unix, Linux, MacOS) or **c:\Users\<USERNAME>\.aws\credentials** (Windows)
- credentials file can be used by multiple projects
- credentials file can contain keys (**profiles**) for multiple AWS accounts or environments
- Credentials should not be stored in code or publicly accessible locations

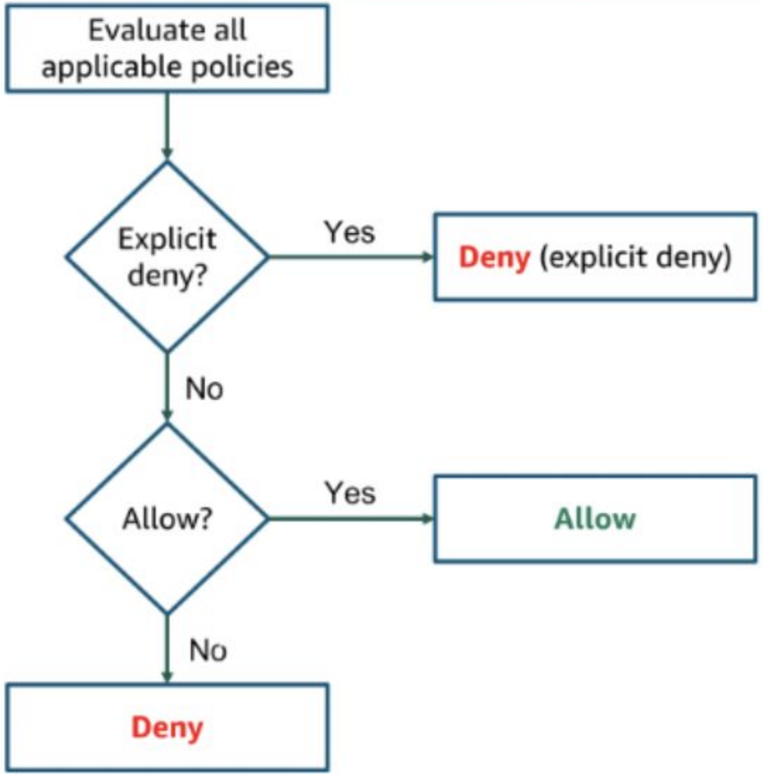
Authorizing with IAM

- By default, an authenticated user, group, or role has no access permissions
- Permissions to access AWS resources are controlled through IAM policies
- IAM policy is a JSON document that defines effect, action, resources, and optional conditions under which an entity can invoke API operations in an AWS account
- Any actions or resources not explicitly allowed are denied
- Actions may include wildcards (asterisks) to cover a set of related actions

Principle of Least Privilege

- Grant only permissions needed to perform a task
- Start with minimum set of permissions and grant additional permissions as needed
- Use account root user to create one or more IAM users
- Use IAM users for ongoing account access and management tasks

Evaluation logic for IAM policies



IAM Policy Types

Identity-based policy

- Attached to an IAM user, group, or role
- Specifies what an identity can do

Resource-based policy

- Attached to a resource
- Specifies what a user or group is permitted to do with the resource

IAM Policies

Managed policy

- Standalone, identity-based for attaching to multiple users, groups, and roles
- Provide reusability, central charge management, versioning, rollback, and ability to delegate permissions management

Inline policy

- Embedded in an entity
- If used for multiple entities, each has its own copy