AWS: A Simple, Visual Guide

With Plain Definitions, Diagrams, and Detailed Comparisons

Introduction

Amazon Web Services (AWS) provides on-demand infrastructure and services so you don't have to buy or maintain physical servers. You pay only for what you use, and you can scale up or down easily. This document explains *how AWS services interact*, *how data flows in multiple directions*, and *the differences/similarities between storage/database services like S3, RDS, DynamoDB, EBS*, and others. We'll keep definitions simple and include helpful diagrams.

1 Key Terms & Definitions (Quick Glossary)

Here are concise definitions for core AWS terms used throughout this guide. Check the diagrams to visualize how they interact.

Region

A geographical area (e.g., `us-east-1` or `eu-west-1`). Each region hosts multiple data centers.

Availability Zone (AZ)

One or more data centers in a region. Deploying resources across AZs can increase resilience.

EC2 (Elastic Compute Cloud)

Virtual machines in AWS. You choose the OS, apps, and scale as needed. AWS manages the physical servers.

Lambda (Serverless)

Upload code, and AWS runs it whenever an event triggers it (e.g., S3 file upload). No need to manage servers.

S3 (Simple Storage Service)

Object storage: store files, images, backups, or logs in *buckets*. Highly durable and globally accessible.

EBS (Elastic Block Store)

Block storage for **EC2** instances, like a virtual hard drive you attach to your EC2 VM.

RDS (Relational Database Service)

Managed SQL database (MySQL, PostgreSQL, etc.). AWS handles patches, backups, and scaling up to a point.

DynamoDB

A NoSQL key-value database. Can handle massive throughput and scales horizontally.

CloudFront

A Content Delivery Network (CDN) that caches or delivers content from edge locations near the user.

Load Balancer (ELB)

Distributes incoming traffic among multiple EC2 servers or containers so no single server is overwhelmed.

API Gateway

Front-end for your REST or WebSocket APIs. Often used with **Lambda** or **EC2** to route external calls.

SNS (Simple Notification Service)

Sends messages to subscribers (e.g., email, text, or AWS services) when certain events happen.

EventBridge

An advanced event bus system that can route events from multiple AWS sources (like S3, EC2) or external apps to your services.

Kinesis

A service for ingesting and processing large streams of data in real time (e.g., logs, sensor data).

Redshift

A data warehouse solution for large-scale analytical queries. Good for big data (SQL-based).

Athena

Run SQL queries on files stored in **S3** with no need to manage a database cluster.

SageMaker

AWS platform to train and deploy machine learning models without manually setting up servers.

ECS/EKS

Containers on AWS. **ECS** is Amazon's own container orchestration, **EKS** is AWS-managed Kubernetes.

Route 53

AWS's Domain Name System (DNS) service. It can route users to different Regions or services.

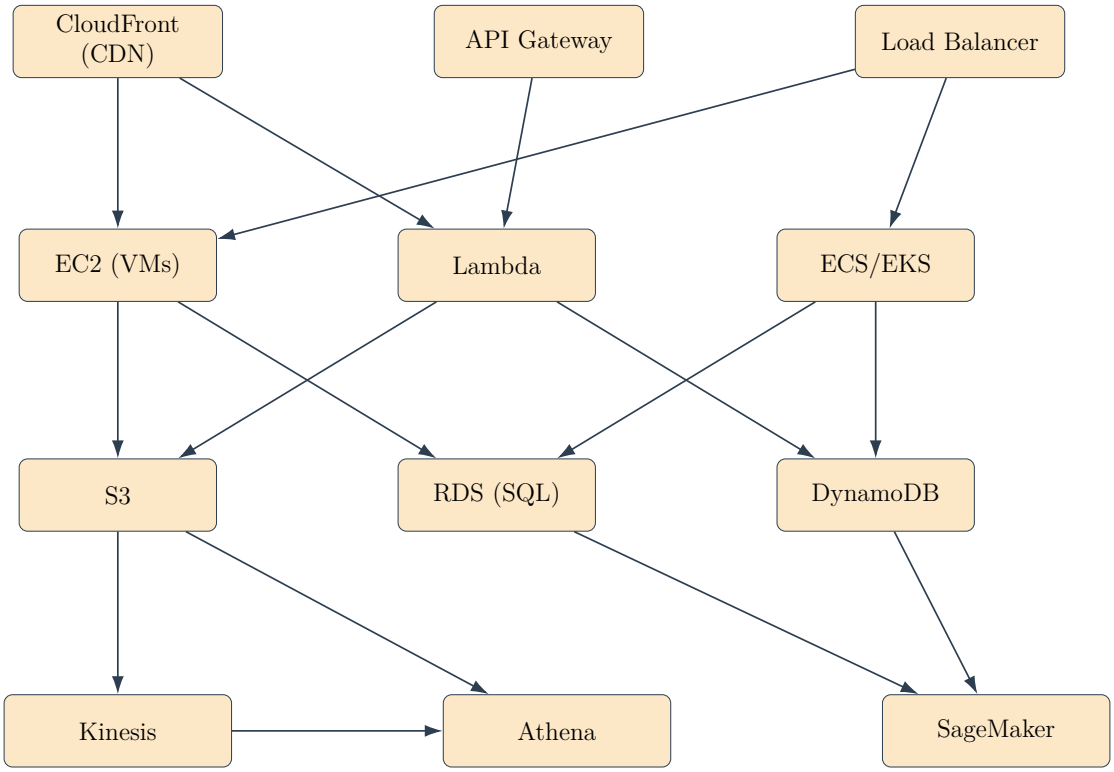
IoT

AWS IoT services let physical devices (sensors, microcontrollers) send/receive data in the cloud.

VPN / DirectConnect

Methods to link your on-premises data center to AWS, creating a **hybrid** environment.

2 How AWS Services Interact (High-Level)



Observations:

- Services like **CloudFront**, **API Gateway**, or **ELB** handle incoming traffic.
- **EC2**, **Lambda**, or **ECS/EKS** do the compute work.
- **S3**, **RDS**, and **DynamoDB** provide storage or database solutions.
- Tools like **Kinesis**, **Athena**, and **SageMaker** handle analytics or ML tasks.

3 AWS Storage & Database Comparisons

AWS has **multiple ways to store data**. Each one fits different use cases.

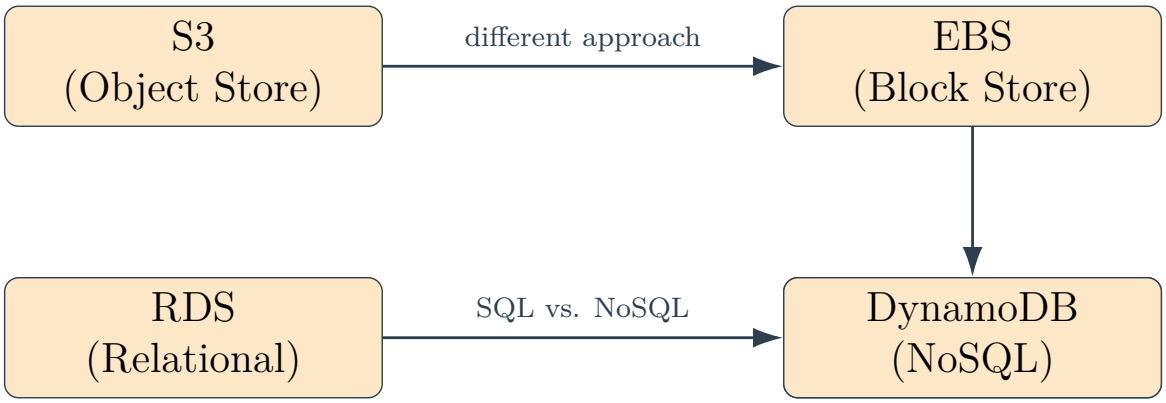
3.1. S3 vs. EBS vs. EFS (for completeness)

- **S3 (Object Storage)**:
  - Store files in buckets.
  - Access via HTTP/HTTPS.
  - Good for images, backups, static websites.
  - Not a traditional filesystem. You get objects, not directories in the same sense.
- **EBS (Block Storage)**:
  - Attaches to **one** EC2 instance like a hard drive.
  - You can format it with a filesystem (e.g., ext4) and mount it in your VM.
  - Good for running databases on a single server or storing OS-level data.
- **EFS (Elastic File System)**:
  - Network file system that can be shared by multiple EC2 instances at once.
  - If you need standard POSIX filesystem features across multiple servers.

3.2. RDS (SQL) vs. DynamoDB (NoSQL)

- **RDS (Relational Database Service)**:
  - SQL-based (MySQL, PostgreSQL, etc.).
  - Tables with columns and rows. Great for structured queries and relationships.
  - AWS manages backups, patches, scaling up to a limit. For read scaling, you can add read replicas.
- **DynamoDB (NoSQL)**:
  - Key-value or document-based.
  - Scales horizontally. You can handle extremely high request rates with auto-scaling.
  - Great for simple queries on large amounts of data.

– Doesnt support complex SQL joins or relationships well.

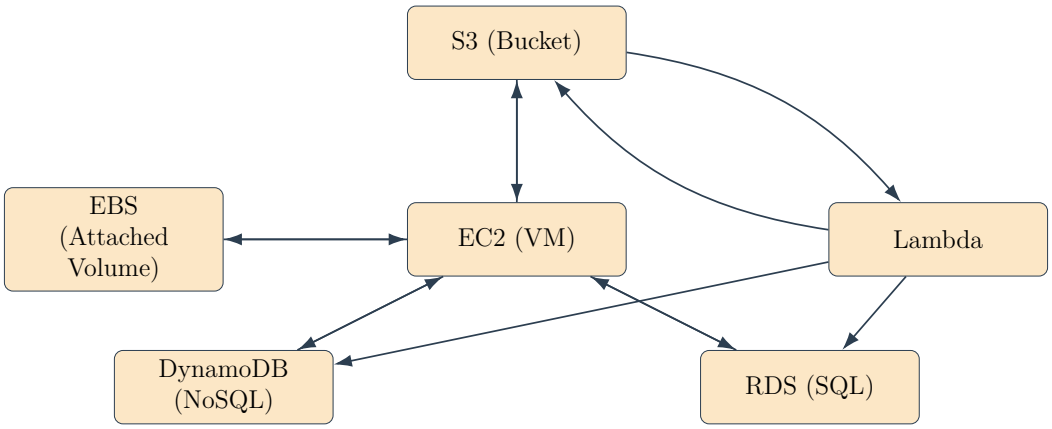


Basic rule of thumb:

- If you want to store *files* or *objects* (like images, PDFs), **S3** is best.
- If you need a **disk** attached to an EC2 instance, use **EBS**.
- If multiple servers need to share a network filesystem, use **EFS**.
- If you want a **SQL** database, choose **RDS**.
- If you want a **NoSQL** (key-value) database that scales big, pick **DynamoDB**.

4 Interactions Among S3, EBS, RDS, and DynamoDB

4.1 Architecture Diagram



Explanations:

- **EBS** is a virtual disk that the EC2 instance uses like a normal hard drive (storing OS files, logs, etc.).
- **S3** can store large objects or backups for the EC2 instance. The instance can upload or download files to S3.
- **RDS** might store relational (SQL) data for the application running on EC2.
- **DynamoDB** might store NoSQL data if the app needs a key-value store for certain high-scale operations.
- **Lambda** can be triggered by changes in S3 or read/writes to RDS or DynamoDB too, adding serverless workflows.

4.2 Differences in Data Access Patterns

- **EBS:** Accessed only by the EC2 instance it's attached to (unless you do specialized multi-attach). Speaks like a normal drive: ext4, NTFS, etc.
- **S3:** Accessed by `HTTP/HTTPS`, can be from EC2, Lambda, or even the public internet if you allow it.
- **RDS:** Accessed via SQL queries (e.g., MySQL protocol). Typically from your application.

- **DynamoDB:** Accessed via AWS SDK or APIs. You do read/write operations with a primary key.

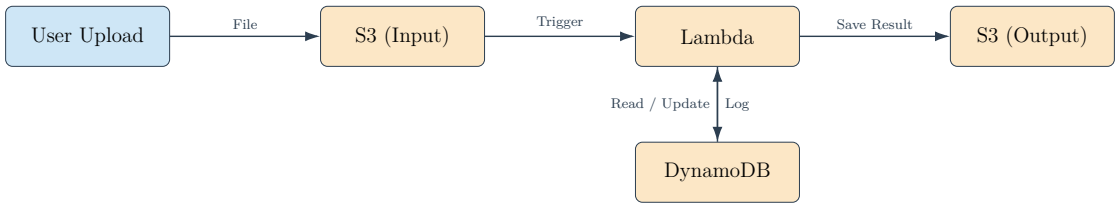
5 Sample Lambda Code (Recap)

Heres a simple AWS Lambda function in Python. Notice naming style: **PascalCase** for functions, **camelCase** for variables, and **MACRO_CASE** for constants.

```python
def ProcessFile(event, context):
    # event: Contains details about what triggered this function (e.g., S3 upload).
    # context: AWS environment info (e.g., memory, request ID).

    fileKey = event["Records"][0]["s3"]["object"]["key"]
    print("Processing file:", fileKey)

    MACRO_CONSTANT = 100
    processedResult = f"File {fileKey} processed with constant {MACRO_CONSTANT}"

    # Possibly store in S3, or update RDS / DynamoDB
    return {
        "statusCode": 200,
        "body": processedResult
    }
```

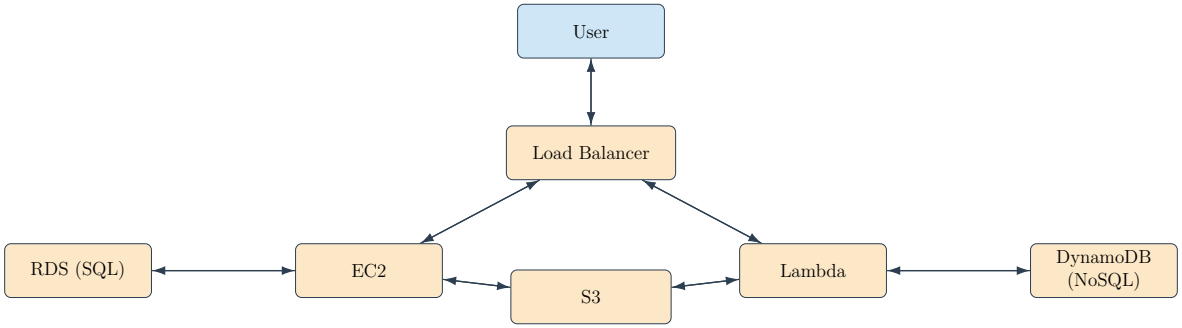
6 Use Cases (Multi-Directional Flows)

6.1 Serverless File Processing with DynamoDB Logging



Flow: 1. User uploads to S3. 2. Lambda processes the file. 3. Output is saved in another S3 bucket. 4. Lambda also logs metadata to DynamoDB (key-value store).

6.2 A Hybrid Database Approach



Flow: 1. Some data is stored in RDS for **transactional** or **relational** needs. 2. Some data is in DynamoDB for **high throughput** or **simple key-value** lookups. 3. S3 holds files or static assets. 4. Both **EC2** and **Lambda** can read from or write to any of these stores.

Conclusion

AWS provides many ways to store and manage data (**S3**, **EBS**, **RDS**, **DynamoDB**, etc.), each suited to different use cases. By mixing these services and letting data flow in multiple directions, you can build flexible, scalable architectures.

Key Points:

- **S3** is object storage for files, **EBS** is block storage for EC2 volumes, **RDS** handles SQL, and **DynamoDB** is NoSQL.
- Compute can be **EC2** (full server) or **Lambda** (serverless). Both can talk to the same storage or database.
- Frontend services like **CloudFront**, **API Gateway**, and **ELB** route user traffic.
- Multi-directional flows are common: a Lambda might read from S3, write to DynamoDB, then trigger an event for more processing.

Next Steps:

- Sign up for the **AWS Free Tier** to try building your own environment.
- Read the **AWS Documentation** for deeper tutorials on each service.
- Experiment with a small project (e.g., a static site on S3 + CloudFront, or a serverless file processor) to gain hands-on experience.