AWS

Some summaries and notes

Eduardo Cunha

Julho 2025

Introduction

This document aims to synthesize the key concepts learned about Amazon Web Services. Its primary purpose is to clarify, summarize, and reinforce the knowledge acquired, with a particular emphasis on theoretical applications. The structure of this paper reflects the order in which the topics were studied, rather than following a strict sequence.

AWS

Cloud computing is a technology that allows users to access computing resources (such as servers, storage, and databases) over the internet, eliminating the need for physical infrastructure.

Why AWS?

Amazon Web Services (AWS) is the most widely used cloud computing platform, offering a broad range of services and a global infrastructure. Its popularity is driven by its reliability, scalability, and service offerings.

S3 Bucket

An S3 Bucket is a storage service provided by AWS designed to store and retrieve any amount of data, such as documents, backups, and digital assets.

Pros:

- Cost-effective for storing large volumes of data
- Highly scalable and reliable

Cons:

- Can experience latency for frequent access
- Not ideal as a primary database due to higher access costs

EC2

EC2 (Elastic Compute Cloud) provides virtual server instances (machines) that can be used to run applications, host servers, or perform computing tasks. It is a foundational service within AWS, functioning as the backbone for many cloud-based solutions.

Pros:

- Highly versatile, supporting a wide range of use cases
- Essential core service for building infrastructure on AWS

Cons:

- Billing is based on uptime rather than actual resource usage, which may lead to higher costs if not managed carefully

ECS (Elastic Container Service)

ECS is a container orchestration service that manages clusters of EC2 instances and automates the deployment, scaling, and management of containerized applications.

pros: simplify scale, on demand scale, cons: more complex for configure, lockin in aws

Pros:

- Simplifies scaling of containerized workloads
- Enables on-demand scaling and management

Cons:

- Configuration can be complex
- Creates some dependency on AWS ecosystem (vendor lock-in)

ALB (Application Load Balancer)

The Application Load Balancer distributes incoming traffic across multiple EC2 instances, ECS clusters, or Lambda functions. It supports advanced routing based on HTTP requests, paths, or hostnames.

Pros:

- Advanced routing capabilities
- Supports WebSockets and modern web protocols

Cons:
- Higher cost compared to basic load balancers
- Can be challenging to configure for complex scenarios

Fargate
Fargate is a serverless compute engine for containers. It removes the need to provision, configure, or manage clusters. You simply define and deploy your containers, and AWS handles the rest.

Pros:
- Reduces operational complexity
- Pricing is based on actual compute and memory resources consumed

Cons:
- Can be more expensive than managing your own EC2 clusters

Elastic Beanstalk
Elastic Beanstalk is a Platform as a Service (PaaS) that allows you to deploy applications without managing the underlying infrastructure. It automates provisioning, load balancing, scaling, and monitoring.

Pros:
- Simple setup for web applications
- Competitive pricing for managed services

Cons:
- Vendor lock-in and limited customization
- Can feel like a “black box,” making troubleshooting more difficult

Lambdas
AWS Lambda is a serverless compute service where you pay only for the compute time you consume. It is typically used for handling individual requests, endpoints, or event-driven tasks.

Pros:
- Cost-effective for low-traffic or unpredictable workloads
- Infinitely scalable to handle any number of requests

Cons:
- Infinite scalability can lead to unexpectedly high costs if not properly managed (e.g., under attack)
- Limited execution time (timeout), lower memory, and potential cold-start latency (first request is slower)
- You are billed for total execution time, including when waiting on external APIs, which may be less efficient than using a server for such tasks

API-GATEWAY
API Gateway acts as the entry point for APIs, receiving client requests and routing them to the appropriate backend service. While similar to a load balancer, its primary function is API management rather than traffic distribution.

Pros:
- Seamless integration with other AWS services
- Supports authentication, throttling, and monitoring

Cons:
- Can introduce additional latency
- Vendor lock-in

RDS (Relational Database Service)

RDS is a managed SQL database service that supports several database engines like MySQL, PostgreSQL, and SQL Server. It offers automated backups, scaling, and maintenance.

Pros:
- Simplifies database management
- Enables easy scalability and high availability

Dynamo

DynamoDB is a fully managed NoSQL database service designed for key-value and document data structures.

Pros:
- Extremely low latency for high-performance applications

Cons:
- Can be more expensive at scale compared to traditional databases

SQS (Simple Queue Service) & SNS (Simple Notification Service)

SQS: A fully managed message queue service for decoupling and buffering messages between distributed systems.

SNS: A pub/sub messaging service for sending notifications to multiple subscribers or endpoints.

Secrets manager

AWS Secrets Manager is used to securely store, manage, and retrieve sensitive information such as API keys, passwords, and other credentials.

Cloudfront

CloudFront is AWS's Content Delivery Network (CDN) that distributes content globally with low latency by caching it closer to users.

EKS (Elastic Kubernetes Service)

Amazon EKS is a fully managed Kubernetes service that simplifies running Kubernetes clusters on AWS and on-premises environments. EKS handles the Kubernetes control plane, automating tasks such as cluster management, scaling, updates, and security, allowing teams to focus on deploying and managing containerized applications rather than infrastructure.

ECK (Elastic Cloud on Kubernetes)

Elastic Cloud on Kubernetes (ECK) is an operator-based solution that simplifies the deployment, management, and scaling of Elastic Stack components (such as Elasticsearch, Kibana, APM Server, Beats, Elastic Agent, Elastic Maps Server, and Logstash) on Kubernetes clusters.

Clarifying some things

Kubernetes and AWS

Do you need Kubernetes when using AWS? Does ALB do the same things?
Kubernetes is a powerful, open-source container orchestration platform. On AWS, you can use Kubernetes via Amazon EKS (Elastic Kubernetes Service), which manages clusters of EC2 instances running containers.

ALB (Application Load Balancer) is a load balancing service that distributes incoming traffic to targets such as EC2 instances, ECS services (including Fargate), or Lambda functions. It handles routing and balancing HTTP(S) requests but does not orchestrate containers or manage their lifecycle.

You do not need Kubernetes to use AWS. For many use cases, AWS services like ECS (with or without Fargate) and ALB provide sufficient orchestration and load balancing.

Kubernetes is used for advanced container orchestration inside EC2/EKS clusters, while ALB is used for routing traffic to services (ECS, EKS, Lambda, etc.)

Fargate vs Elastic Beanstalk vs Lambda

They seem similar, what are the main differences?

Feature	**Fargate**	**Elastic Beanstalk**	**Lambda**
Type	Serverless containers	Platform as a Service (PaaS)	Serverless functions
Deployment	Container images (ECS/EKS)	Code or containers (multi-language)	Individual functions (code snippets)
Scaling	By number of containers/tasks	By EC2 instances (auto-scaling)	By number of function invocations
Use case	Microservices, long-running, batch jobs	Web apps, APIs, background jobs	Event-driven, short-lived tasks
Control	High (over container/runtime)	Low (abstracts infra details)	Very low (fully abstracted)
Cost model	Pay for running containers	Pay for underlying resources	Pay per request and execution time
Management	No server management, some config needed	Minimal config, AWS manages infra	No server/container management
Limitations	Needs container knowledge	Less flexibility, slower scaling	Max 15 min per execution, cold starts

Summary:

- Fargate: Best for running containers without managing servers. Good for microservices, APIs, or jobs that need custom runtimes or longer execution times.
- Elastic Beanstalk: Easiest for deploying web applications with minimal infrastructure management. Supports multiple languages and frameworks. Good for traditional web apps and APIs.
- Lambda: Best for event-driven, short-lived workloads (e.g., API endpoints, file processing, automation). Scales instantly but has execution time and memory limits.