Modern Architecture and key points in AWS

Contents

01 Introduction

02 Gravity

03 Evolution of Architecture

04 Modern Architecture Principles

05 AWS Services for Modern Architecture

06 Case Studies

07 Choosing AWS Services

08 Common Mistakes

09 What’s Next in Cloud Dev

10 Key Takeaways

Speaker

SWE @ JBC

BRIDGEBOOKS

Shad Reza

Introduction

Architecture is the **foundation** of every **system** — how components are structured, connected, and scaled

Cloud architecture shifts the base foundation from data centers to **dynamic**, **virtualized** and **scalable**, on-demand **resources**

AWS (Amazon Web Services) is the world's leading **cloud platform** — offering a **vast** toolbox of **services** that make it faster and easier to build modern, resilient applications.

Gravity

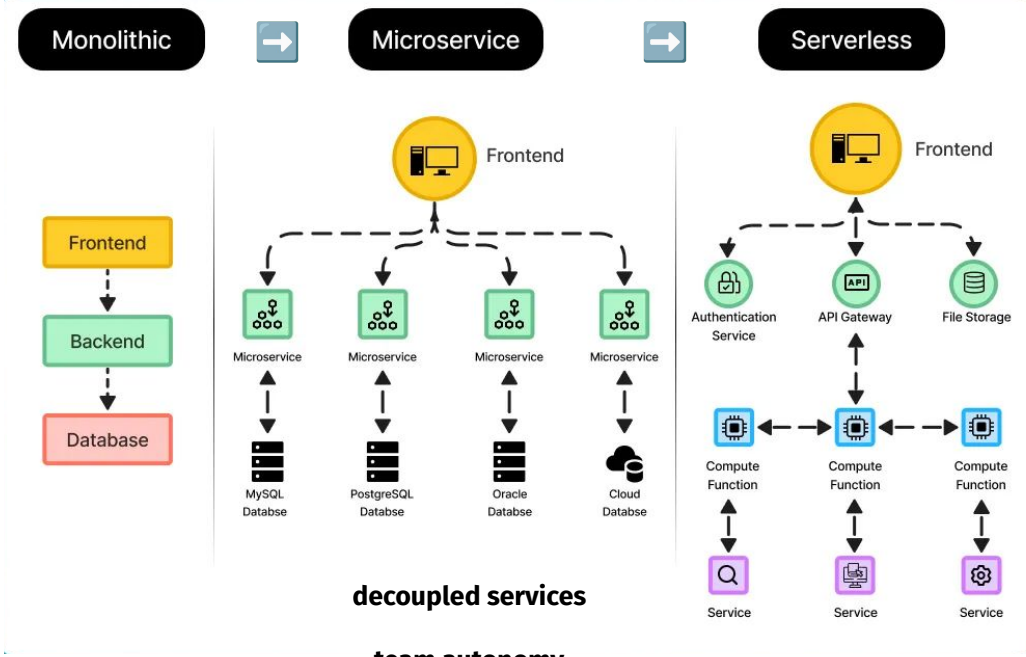
- Cloud development has **changed** dramatically in the last decade
- Businesses demand **faster** delivery, **greater** scale, and **smarter** cost control
- Modern architecture solves these challenges by combining patterns like **microservices**, **serverless**, and **automation** — with **AWS** as the backbone

Evolution of Architecture

tightly coupled

hard to scale

single point of failure



decoupled services

team autonomy

independent scaling

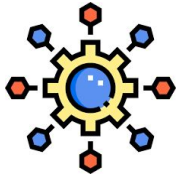
pay-per-use

scale to zero

async flows

higher agility

Modern Architecture Principles



Micro Services

Small and Independent units of business logic

Infrastructure as Code [IaC]

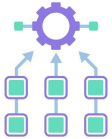
Repeatable, auditable infra via tools like Terraform, AWS CDK

Serverless

No servers to manage
Pay per execution

Observability

Metrics
Tracing

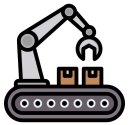


Event Driven

Async
Decoupled Communication

Automation

Altering
CI/CD

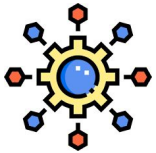


AWS Services for Modern Architecture

200+ services

Cloud Computing Essentials	Simple Storage Service (S3)		Elastic Compute Cloud (EC2)	
Cloud First Steps	Management Console	IAM	CloudWatch	
Computing Solutions	Elastic Compute Cloud (EC2)	Lambda	Fargate	ECS
Cloud Economics	Cost Explorer	Budgets	Cost & Usage Report	
Networking Concepts	Virtual Private Cloud (VPC)	Route 53	API Gateway	
Connecting VPCs	VPC Peering	Transit Gateway	Direct Connect	VPN Connection
First NoSQL Database	DynamoDB			
Storage and File Systems	Simple Storage Service (S3)	EBS	EFS	FSx
Auto-healing and Scaling Applications	EC2 Auto Scaling	ELB	CloudFront	
Serverless Architectures	Simple Storage Service (S3)	Lambda	API Gateway	Step Functions
DevOps and Continuous Delivery	Code Commit	Code Build	Code Deploy	Code Pipeline
Relational Database Solutions	RDS	Aurora		
Monitoring and Observability	CloudWatch	CloudTrail	Config	X-Ray
Messaging and Queuing	SNS	SQS	MQ	
Security and Compliance	IAM	WAF	Shield	GuardDuty
	Secrets Manager	KMS	NACL	Security Groups
Migration to the Cloud	Application Migration Service	DMS	Snowball	DataSync
Machine Learning and AI	Comprehend	SageMaker	Lex	Rekognition

Micro Services

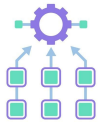


AWS ECS · Amazon EKS · AWS Lambda · Amazon API Gateway

Serverless

AWS Lambda · DynamoDB · Amazon S3 · Amazon EventBridge

Event Driven



Amazon EventBridge · Amazon SQS · amazon SNS · AWS Step Functions

IaC

CloudFormation · aws CDK · HashiCorp Terraform

Observability

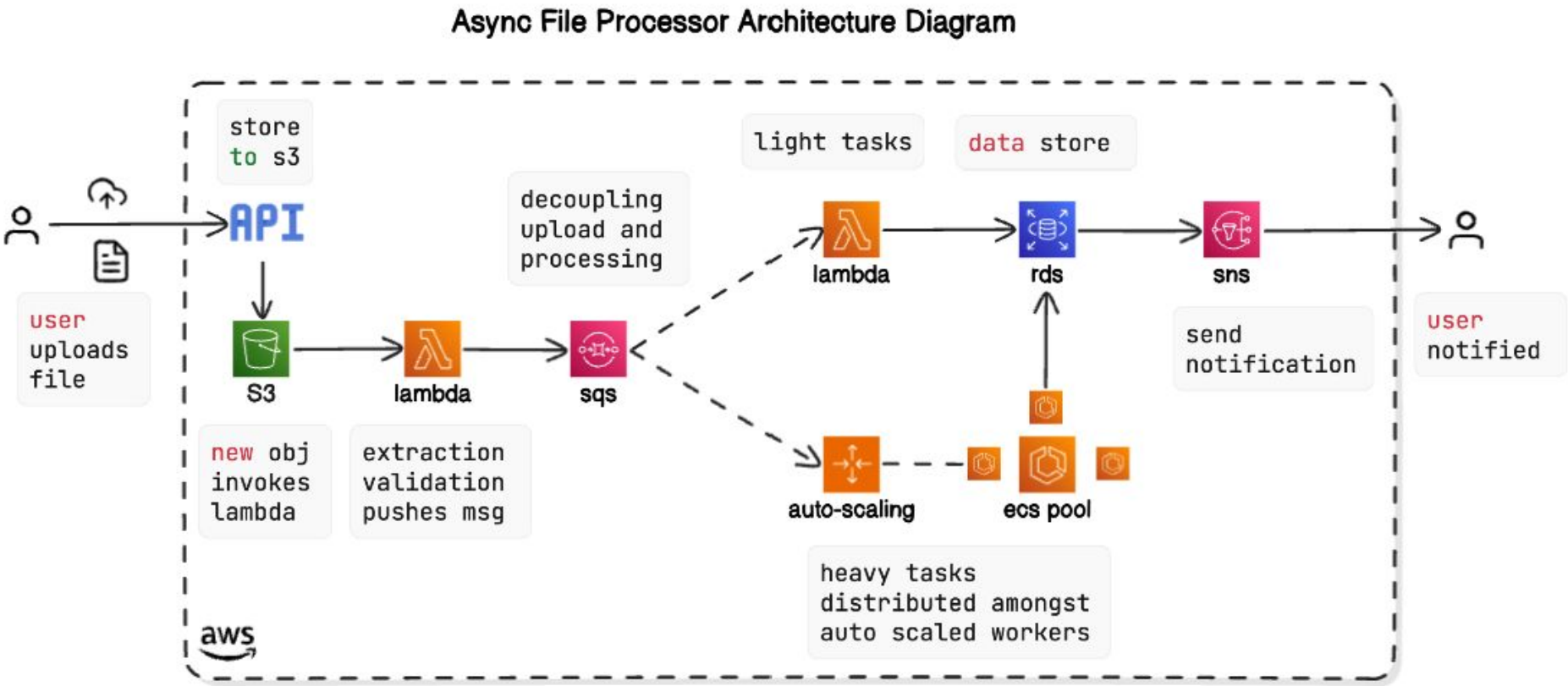
Amazon CloudWatch · AWS X-Ray · AWS Distro for OpenTelemetry

Case Study 01 - Async File Processor

- Users upload **files** that **need processing**
  - **Image resizing**
  - **PDF parsing**
  - **Metadata extraction**
- Processing may **take time** — it **shouldn't block** the user experience
- Solution must handle **bursty traffic** and **scale automatically**

- **Separate upload from processing**
- Allow file **processors** to **scale independently**
- **Notify** users when **processing is done**

Case Study 01 - Async File Processor Architecture Diagram



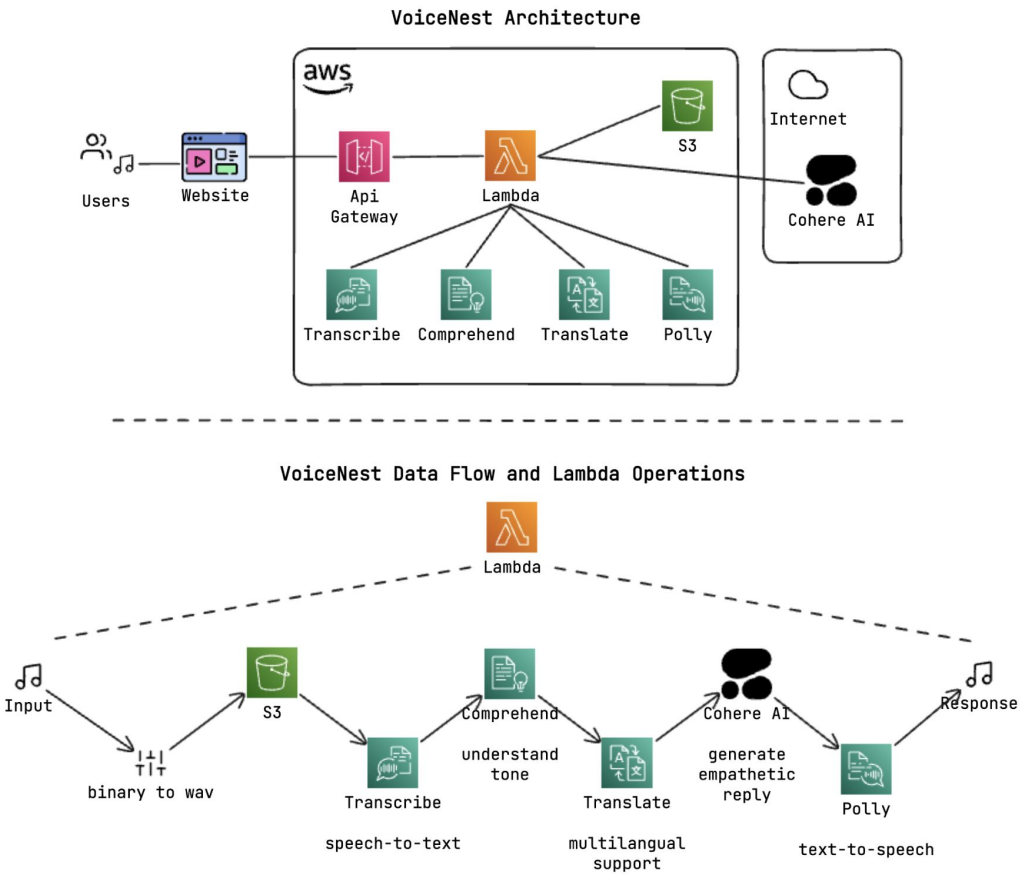
Case Study 02 - Serverless AI Companion App

- **Millions** of **elderly individuals** suffer from **digital loneliness**
- Most tech is **complex**: apps, screens, buttons — **barriers** for the elderly
- There's a **growing need** for **emotional companionship**, delivered **simply**

- **Build** an **empathetic, multilingual voice assistant**
- One that works **only by speaking** — no screens, no typing
- Entire **backend** must be **serverless**, **scalable**, and **affordable** [considering Modern Architecture]

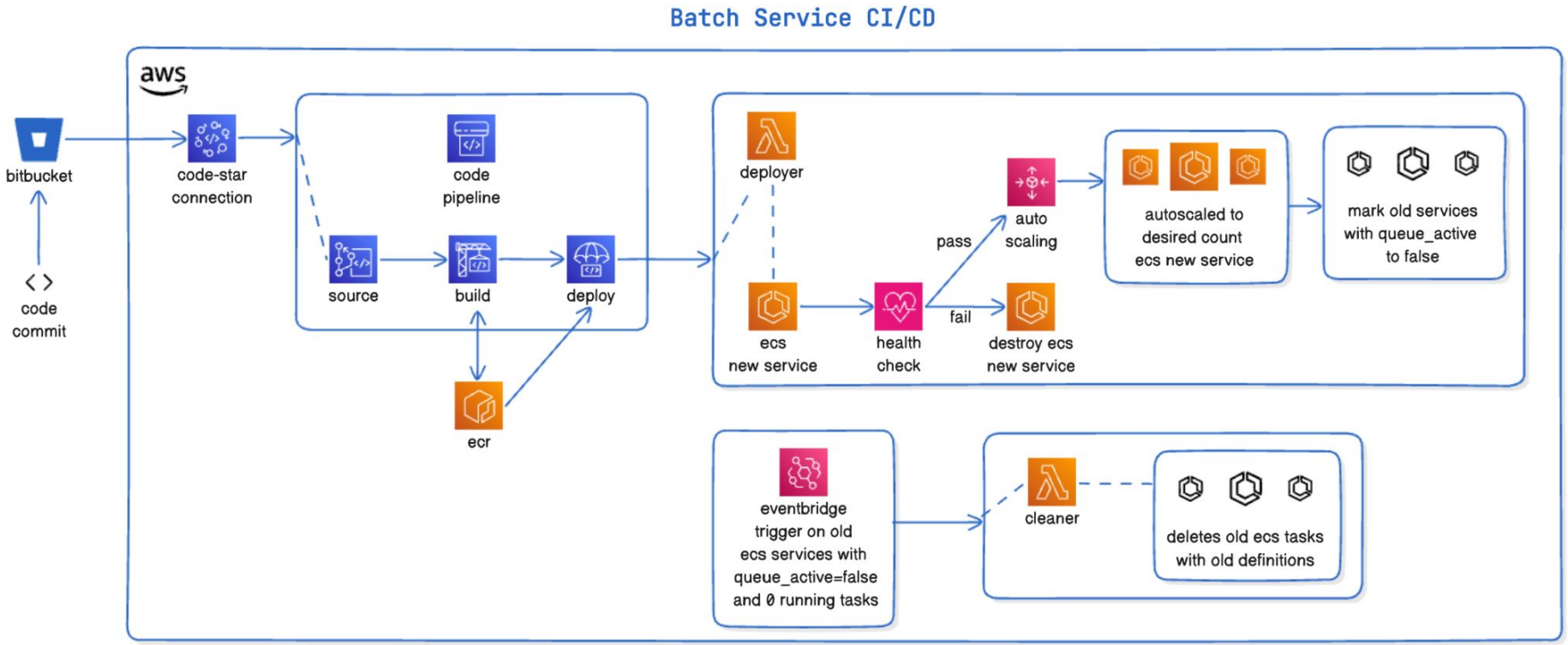
- **Serverless backend** that **understands**, **responds**, and **speaks in natural voice**
- Runs on **AWS free Tier** — from voice input to emotional AI response

Case Study 02 - VoiceNest Architecture Diagram



Speak Your Heart

Case Study 03 - CI/CD Pipeline



Choosing AWS Services

💼 Use Case	🟢 Service A	🔵 Service B
🧠 **Compute**	**AWS Lambda** • **Serverless**, auto-scales • Ideal for **short, event-based tasks**	**ECS / Fargate** • **Container-based**, custom runtimes • Best for **long-running workloads**
💾 **Storage**	**Amazon S3** • **Object storage**, highly durable • Great for **media, backups, static assets**	**Amazon EFS** • **File system**, POSIX-compliant • For **shared storage** across apps
🔔 **Messaging**	**Amazon SQS** • **Queue-based** decoupling • Built-in **retries & DLQ**	**Amazon EventBridge** • **Event routing** with filters • Connects AWS + **SaaS services**
🗃️ **Database**	**DynamoDB** • **Serverless NoSQL**, auto-scale • High throughput, **low latency**	**RDS** • **Relational DB**, managed • Great for **joins & transactions**
🚀 **Deployment**	**CodePipeline** • **AWS-native CI/CD** • Simple, integrated pipelines	**CDK / Terraform + GitHub Actions** • **IaC + external CI** • More **flexibility & control**

* Portable Operating System Interface [POSIX]

Common Mistakes

Cold Starts Ignored

➤ Use *provisioned concurrency* for latency-sensitive functions

IAM Over-permissioned

➤ Enforce *least privilege*, never use wildcards (*)

Synchronous-Only Thinking

➤ Prefer *asynchronous workflows* for scale and resilience

No Retries or DLQs

➤ Add *retry logic* and *dead letter queues* by default

Cost Blindness

➤ Set up *budgets, alarms, and usage reports* from day one

Zero Observability

➤ Use *CloudWatch, X-Ray, structured logs* — always

What’s Next in Cloud Development

AI-Native Architectures
➢ LLMs, vector DBs, and serverless orchestration

Event-Driven Everything
➢ EventBridge as the new glue — decouple, scale, reroute easily

Graviton & Arm-based Efficiency
➢ Lower cost and power use → better sustainability

Edge + Hybrid Apps
➢ AWS Lambda@Edge, CloudFront Functions, Local Zones

Sustainability by Design
➢ Optimize resources, reduce idle time, turn off what you don’t use

Amazon Braket
Get started with quantum computing

Key Takeaways

1. Modern architecture ≠ complexity — it's about **smart choices**
2. **Serverless, event-driven, IaC, and AI** are cornerstones
3. AWS offers **modular services** — pick what fits your use case
4. Prioritize **observability, security, and automation**
5. Think like an **architect** — design for scale, change, and failure

```
╔═══════════════════════════════════════════════════════════╗
║  "Modern architecture isn't about using everything –      ╒
║   it's about using the right things,                      ╒
║   for the right reasons,                                  ╒
║   with the future in mind."                               ╒
║                                                           ╒
║                                  – Shad Reza              ╒
╚═══════════════════════════════════════════════════════════╝
```

thanks!