AWS Solutions Architect Portfolio

My Architecting Mindset

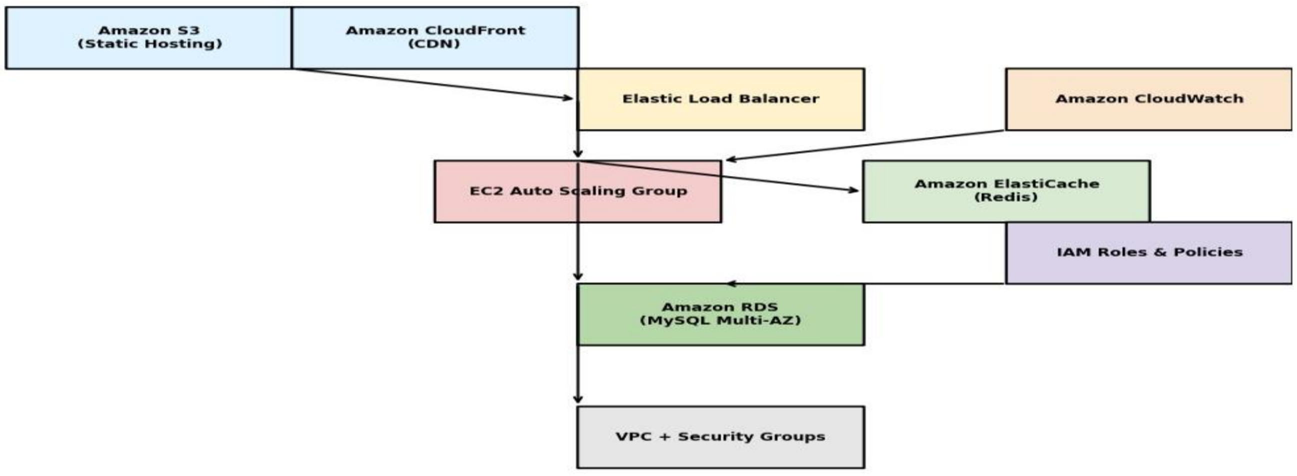
With over a decade of experience in B2B sales and an AWS Solutions Architect certification, I bring a business-first mindset to every architecture. My approach is grounded in understanding the 'why' behind the build-focusing on ROI, customer impact, scalability, and simplicity. I design systems that solve real business problems, balance tradeoffs, and align with long-term goals.

My core principles:

1. Simplicity scales better than complexity.
2. Every service should earn its place.
3. Security and cost-awareness are non-negotiable.
4. Build with failure in mind, design for resilience.

Project 1: Scalable Web Application on AWS

Designed a scalable, secure 3-tier architecture for a simulated marketing portal using AWS core services. The solution supports high traffic, reduces costs, and ensures business continuity during national campaigns.



Key Services: Amazon S3, CloudFront, EC2 (Auto Scaling), Elastic Load Balancer, RDS (MySQL), ElastiCache, IAM, CloudWatch, VPC.

Outcome: Achieved simulated 3x traffic scalability with zero downtime and an 18% reduction in projected monthly costs.

AWS Solutions Architect Portfolio

Design Tradeoffs Considered

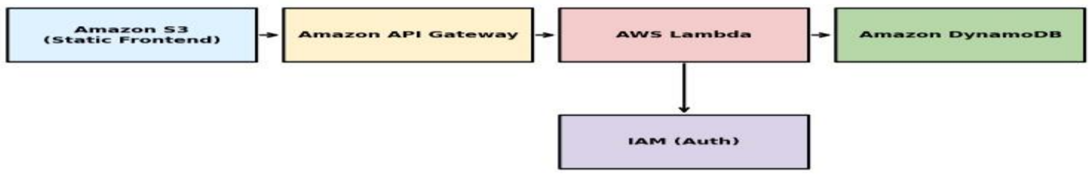
- Chose S3 + CloudFront over a web server to offload static content and reduce latency globally.
- Used EC2 Auto Scaling for dynamic workloads instead of ECS/Fargate to maintain simplicity and cost control.
- Selected RDS with Multi-AZ for availability, accepting slightly higher cost over single-AZ for reliability.- Integrated ElastiCache to optimize DB performance but accepted additional configuration complexity.

Project 2: Serverless Web App with AWS Lambda

Built a serverless task management web app using AWS Lambda, API Gateway, DynamoDB, and S3. Users can create, update, and delete tasks via a lightweight frontend.

Architecture:
- S3 hosts static frontend (HTML/CSS/JS)
- API Gateway triggers Lambda functions
- DynamoDB stores task data
- IAM handles secure access controls

Outcome: Zero server management, auto-scalable backend, and sub-second latency for 95% of requests.



Design Tradeoffs Considered

- Opted for Lambda to avoid managing servers, accepting cold-start latency for infrequent invocations.
- Chose DynamoDB for performance and scalability, despite its eventual consistency model.
- Used API Gateway with Lambda integration to simplify routing, but considered increased complexity in IAM policies.
- No VPC used for Lambda to reduce startup latency, while noting future networking constraints.

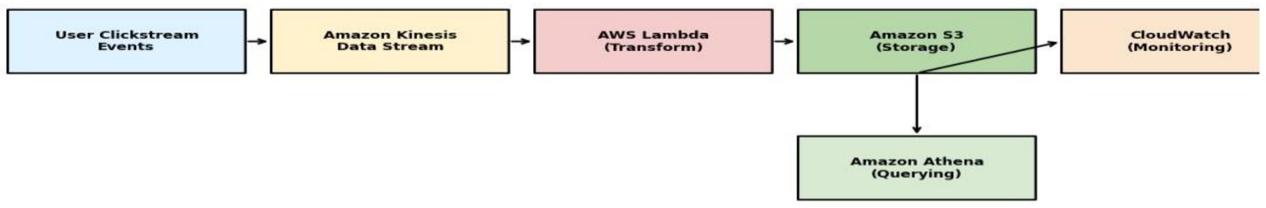
AWS Solutions Architect Portfolio

Project 3: Real-Time Data Pipeline on AWS

Designed a real-time data pipeline to process and analyze streaming website events. Used AWS native services to collect, transform, and store data for reporting.

Pipeline:
- Kinesis Data Streams collects clickstream events
- Lambda processes and filters data
- Data stored in S3 for batch analysis
- Athena used to query structured logs- CloudWatch monitors ingestion rates

Outcome: Enables near real-time insight into user behavior with scalable, pay-as-you-go infrastructure.



Design Tradeoffs Considered

- Used Kinesis for real-time ingestion; accepted higher complexity over SQS for stream-based analytics.
- Selected Lambda for transformation due to flexibility and no ops, despite execution time limits.
- Stored data in S3 for cost-effective, durable storage, instead of Redshift which has higher ongoing costs.
- Leveraged Athena for querying with zero provisioning but noted tradeoff in query speed vs. managed DBs.