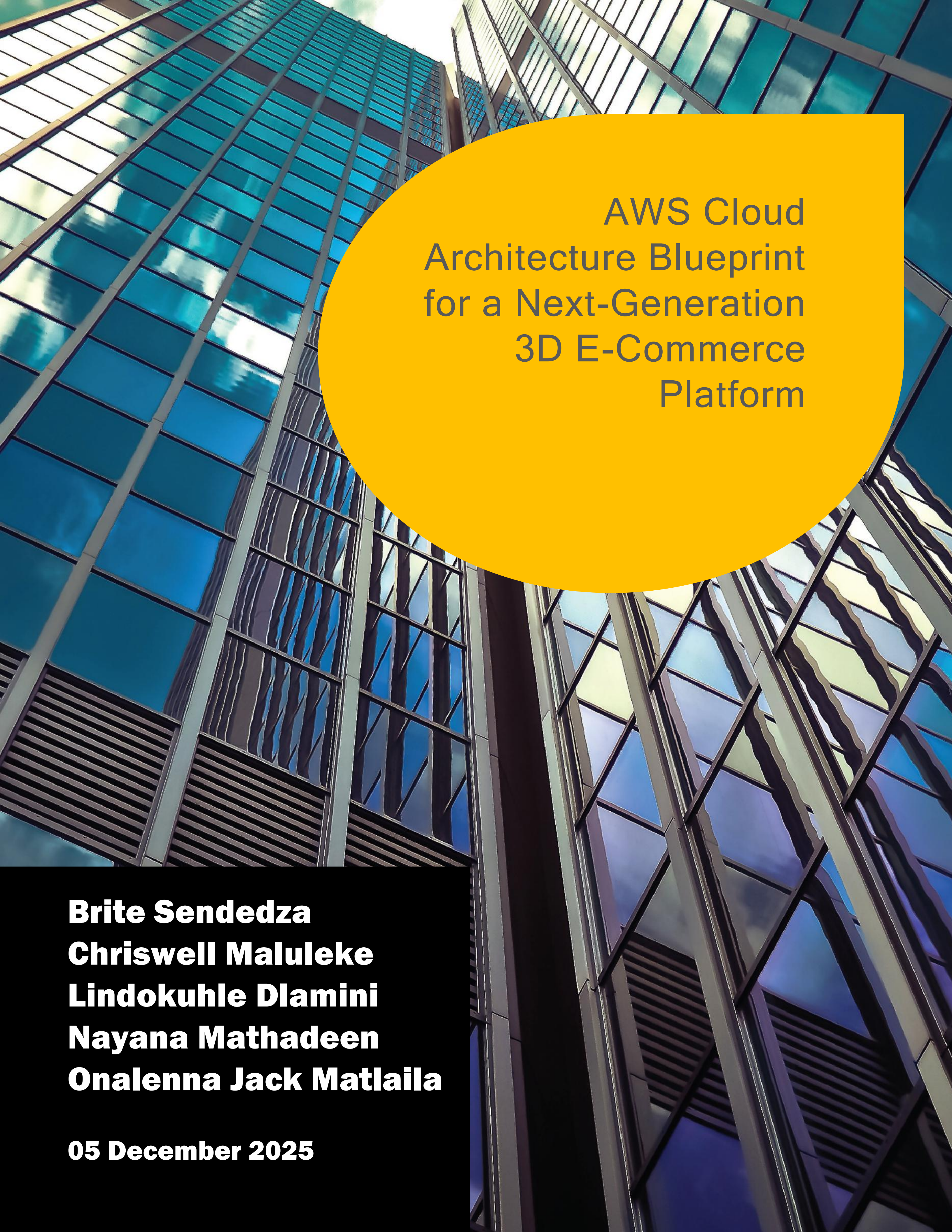
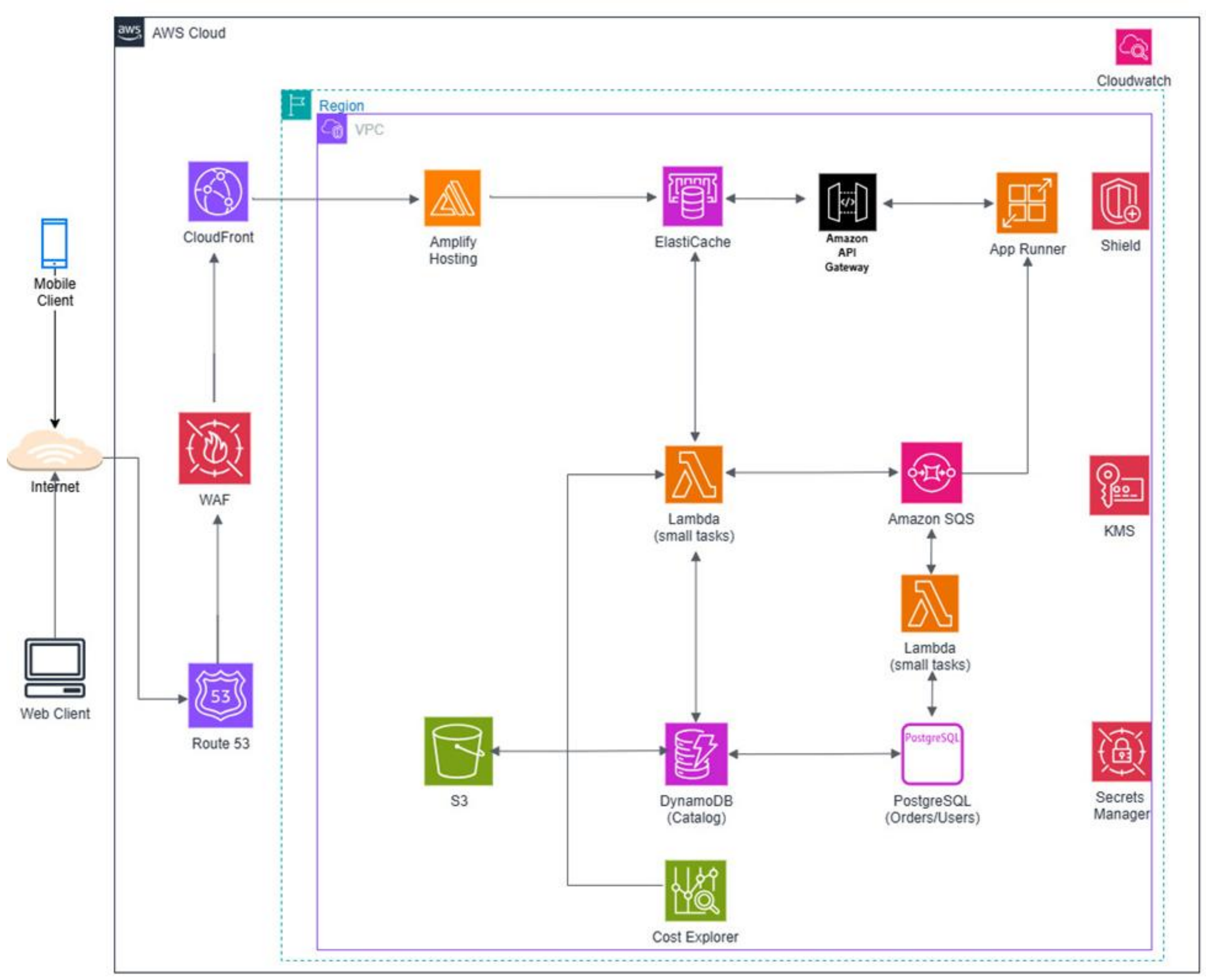


AWS Cloud Architecture Blueprint for a Next-Generation 3D E-Commerce Platform

Brite Sendedza
Chriswell Maluleke
Lindokuhle Dlamini
Nayana Mathadeen
Onalenna Jack Matlaila

05 December 2025

The AWS architecture:



Why we chose each AWS service:

Frontend

The frontend refers to the client-side of the application, focusing on user experience and interface. It's what users interact with directly more like the face of the app.

Route 53: Domain name system (DNS) management for routing users to the application, providing high availability and scalability. Think of it like a phonebook of internet. It helps users find your app by translating domain names into IP addresses.

CloudFront: Content delivery network (CDN) for distributing static assets, reducing latency, and improving user experience. Imagine a network of caching servers around the world, storing copies of your website's static files, so users get faster access no matter the location.

Amplify Hosting: Managed hosting for web applications, providing automated deployment, SSL/TLS, and integration with other AWS services. It's like having a hosting service that takes care of all the technical details, so you can focus on building your app.

Backend

The backend refers to the server-side of the application, handling business logic, database interactions, and API integrations. Basically this is where the magic happens.

App Runner: Containerized application hosting, simplifying deployment and scaling of containerized applications. It's more of like having a managed platform for running your containerized apps, without worrying about the underlying infrastructure.

Lambda: Serverless compute for handling small, event-driven tasks, reducing operational overhead. Think of it like a service that runs your code only when needed, without requiring server management.

Storage

Storage refers to the services used to store and manage data.

S3 (3D assets + static files): Object storage for hosting static files, providing durability, scalability, and high availability. It's like having a massive, secure hard drive in the cloud, where you can store and serve files.

Databases

Databases are specialized storage services for structured data.

DynamoDB (catalog): NoSQL database for handling high-traffic catalog data, providing low latency and scalability. It's designed for fast, efficient data retrieval and storage.

RDS PostgreSQL (orders / users): Relational database for managing orders and user data, providing ACID compliance and SQL querying capabilities. It's a managed database service, handling the underlying infrastructure, so you can focus on your data.

Performance

Performance services focus on optimizing application speed and efficiency.

ElastiCache Redis: In-memory caching for improving application performance, reducing database load, and enhancing user experience. It's like having a super-fast cache layer, storing frequently accessed data.

SQS + Lambda workers: Message queuing and processing for handling asynchronous tasks, decoupling application components, and improving scalability. It's like having a message broker, handling task queues and processing, so your app can focus on other things.

Security

Security services protect the application and its data from threats and vulnerabilities.

WAF: Web application firewall for protecting against common web exploits and vulnerabilities. It's like having a security guard, monitoring and blocking malicious traffic.

Shield: DDoS protection for safeguarding against distributed denial-of-service attacks. It's like having a shield, protecting your app from overwhelming traffic.

KMS: Key management service for encrypting sensitive data and managing encryption keys. It's like having a secure key vault, protecting your encryption keys.

Secrets Manage: Secrets management for securely storing and retrieving sensitive information, such as database credentials. It's like having a secure password manager, storing and retrieving secrets.

Monitoring

Monitoring services track application performance, errors, and security issues.

CloudWatch: Monitoring and logging service for tracking application performance, errors, and security issues. It's like having a dashboard, showing real-time insights into your app's health.

Cost Explorer: Cost management tool for analyzing and optimizing AWS costs, ensuring efficient resource utilization. It's like having a financial advisor, helping you optimize your AWS spending.

How our architecture meets the five requirements?

AWS services used:

Frontend: Route 53, CloudFront, Amplify Hosting

Backend: App Runner, Lambda

Storage: Amazon S3 (3D Assets + statis files)

Databases: DynamoDB, RDS PostgreSQL

Performance: ElastiCache Redis, SQS +Lambda

Security: WAF, Shield, KMS, Secrets Manager

Monitoring: CloudWatch, Cost Explorer

Our five requirements:

- High Availability
- Scalability
- Performance
- Security
- Cost Optimization

How our services meet our requirement:

High availability is the ability of a website/ services to remain operating for majority of the time, meaning the website has minimal downtime. To meet the requirement of high availability, services such as Elastic Load Balancer should be used to direct incoming traffic to healthy endpoints. ELB can be configured for Lambda, CloudWatch and IP Addresses. CloudFront is a Content Delivery Network service provided by AWS that speeds up delivery of web content, API's and applications. CloudFront will cache information to an edge location, making the content easily accessible for a customer and it can also distribute traffic to a global network of servers. Amazon S3 has a multi-AZ deployment feature which allows storage of assets and static files in more than one availability zone. Should one zone fail, the second AZ will take over as the main storage point. Auto scaling can allow for high availability, by increasing or decreasing compute services according to the demand at the time. Databases can have multi-AZ deployment and have a failover policy, where should one availability zone have an issue, another database instance in another AZ will take over as the main database. ElastiCache also allows for high availability, by storing frequently accessed information/files closed to the customer, making it easier for them to access their information.

With time, demand will increase and the website will need to accommodate more users and a bigger inventory. To cope with this, our architecture was built with scalability in mind using services such as Amazon S3, CloudFront, Lambda and Auto-Scaling. Amazon S3 has virtually unlimited storage, so as inventory demand increases more static files and 3D Assets can be added without any storage limits. CloudFront directs traffic through a global network of servers to increase content delivery to customers, and

Lambda is a serverless service where AWS manages scaling to meet the demand. Auto-scaling allows the increase or decrease of compute resources according to the demand.

In terms of performance CloudFront, Elastic Load balancer and CloudWatch helps us achieve the best performance for our needs. CloudFront caches content closer to its users reducing latency, Elastic Load Balancer will distribute traffic to healthy endpoints, re-routing traffic where needed and CloudWatch monitoring can help identify bottlenecks in the system.

To protect our beautiful architecture we used Amazon WAF, Shield, KMS and Secrets Manager. Web Application Firewall (WAF) can filter web traffic with allows and block rules and protect your website from common exploits such as SQL-injection and cross-site-scripting. Traffic filtering can be based on IP address, request headers and request rate. Shield is a service that protects against DDOS attacks and identifying network security configuration issues. Key Management Service is a key management service that enables easy management of cryptogenic keys for encryption and decryption. Secrets Manager protects access to the applications, services and its resources by storing, encrypting and retrieval of secrets such as API keys, database credentials and passwords. They aren't required to stay within the application but can be managed by Secrets Manager.

To achieve a high level of cost optimization AWS Cost Explorer was used in our architecture. AWS Cost Explorer allows the visualization, understanding and management of AWS cost and usage. This allows us to optimize our usage of AWS services whilst optimizing costs.

Design Trade-Offs and Challenges

1. Using App Runner vs. ECS or Lambda for Compute

App Runner provides simplified deployment, autoscaling, and HTTPS, reducing operational burden. The trade-off is reduced infrastructure control and higher costs for long-running workloads. It prioritizes developer speed over deep customization.

2. DynamoDB + RDS Combination

Using DynamoDB for catalog data and RDS PostgreSQL for transactional workloads improves performance and reliability. The trade-off is increased operational complexity, duplicated data models, and the need for careful consistency management.

3. CloudFront + Amplify Hosting

Amplify Hosting simplifies frontend deployment, while CloudFront provides global caching. The challenge is managing cache invalidation so users do not see outdated content. This requires deliberate cache behaviors and versioning strategies.

4. High Availability vs. Cost

Multi-AZ RDS, autoscaling compute, global CDN distribution, and caching layers improve uptime and performance, but they significantly increase cost. Teams must balance reliability with budget constraints.

5. Using Redis (ElastiCache) for Performance

ElastiCache provides very fast response times for sessions, carts, and hot data. Challenges include failover complexity, in-memory volatility, and the need for fallback strategies if Redis becomes unavailable.

6. Introducing SQS and Lambda Workers

SQS and Lambda provide reliable asynchronous processing. The trade-off is additional moving parts (queues, DLQs, retry logic). Teams must ensure idempotency and message integrity to avoid duplicate order processing.

7. Security Layers (WAF, IAM, Shield, KMS)

Security services strengthen the platform but increase configuration overhead. IAM misconfigurations, WAF rule tuning, and KMS API costs require continuous monitoring and expertise.

8. Complexity vs. Simplicity

This architecture is highly scalable and robust but introduces many interconnected services. Without strong logging, monitoring, and documentation, troubleshooting becomes difficult for small teams.