Designing a 3D E-Commerce Platform Architecture on AWS

Scenario

As part of a startup team launching a next-generation 3D e-commerce web application, the platform will allow users to interact with 3D models of products (e.g., furniture, gadgets, and fashion items) before making a purchase. Millions of users are expected globally, and the experience must be fast, highly available, secure, and cost-efficient.

As a Cloud Practitioner, my role was to design the cloud architecture using AWS services to support the 3D application while meeting key business and technical requirements.

Services Used in the Architecture and Justification

Amazon Route 53

I chose Route 53 for its reliability and efficiency in routing users to the web application through DNS (Domain Name System). Route 53 is designed to facilitate high availability and reliability. When users access the web application, their requests are directed to the nearest CloudFront edge location, ensuring the fastest possible experience.

Amazon CloudFront and AWS WAF (Web Application Firewall)

CloudFront is a content delivery network (CDN) designed to serve content as close to users as possible. It leverages edge locations as part of the worldwide AWS global edge network, which significantly reduces latency. AWS WAF helps protect web applications from exploits that could compromise availability or security. By integrating CloudFront with WAF, the service can inspect and manage web requests based on specific criteria, strengthening the overall security posture.

Amazon S3 Bucket

I selected S3 buckets for storing static web files and serving as a repository for 3D models due to its high availability. Storing 3D assets in S3 decouples storage from compute, providing flexibility and efficiency in scaling both storage and compute resources independently.

Application Load Balancer (ALB)

The ALB distributes incoming traffic across compute instances to prevent application downtime during traffic spikes. During peak seasons or sales events, the load balancer automatically handles increased traffic. It performs HTTP health checks and routes requests based on URL paths.

EC2 Auto Scaling

For backend compute hosting APIs and rendering logic, I used EC2 instances with Auto Scaling. Given the unpredictable nature of e-commerce traffic especially during sales and festive seasons, Auto Scaling dynamically adjusts capacity to handle growth and scales down during low-traffic periods to optimize costs.

Amazon RDS (Multi-AZ)

For the database layer, I chose a fully managed relational database service. RDS simplifies setup, operation, and scaling of relational databases in the cloud. Its relational nature supports product and order data with ACID transactions, ensuring accurate storage and processing of user payments and orders. Multi-AZ deployment creates a synchronously replicated standby instance, enabling automatic failover if the primary database fails.

Amazon ElasticCache

For in-memory caching, I used ElasticCache (Redis). Since 3D product metadata and user sessions are frequently accessed, caching this data reduces read load on RDS and enables faster retrieval, improving overall application performance.

Amazon CloudWatch and AWS Trusted Advisor

These services were implemented for monitoring and cost optimization. CloudWatch monitors EC2 CPU utilization and provides logs for debugging. Trusted Advisor identifies idle load balancers or underutilized RDS instances, helping to reduce unnecessary costs.

How the Architecture Meets Requirements

1. High Availability (24/7 with Fault Tolerance)

- EC2 instances are deployed across three Availability Zones (AZs)

- RDS database is configured in Multi-AZ mode for automatic failover

- Elastic Load Balancer distributes traffic evenly across healthy instances

- CloudFront edge locations reduce page rendering time by serving content locally

2.Scalability

- CloudFront handles millions of user requests at the edge, preventing application slowdowns

- Auto Scaling automates resource adjustment based on real-time demand

3.Performance

- CloudFront with WAF enables fast traffic inspection at the edge

- 3D models are cached at CloudFront edge locations close to users

- ElasticCache reduces latency for API calls by serving cached query results

4.Security

- CloudFront and WAF act as the first line of defense against internet exploits before traffic reaches the AWS backbone

- S3 buckets are configured to be accessible only through CloudFront, preventing direct public access

- If attackers attempt to access S3 buckets via direct URLs, WAF handles traffic and denies unauthorized access

- EC2 and RDS instances are placed in private subnets within the VPC with no public internet access

5.Cost Optimization

- CloudFront reduces the load on the Application Load Balancer and EC2 instances, allowing fewer instances to handle the same traffic

- S3's serverless nature ensures the team only pays for actual storage used, avoiding costs for unused compute or storage resources

Design Trade-offs and Challenges

Challenge: CORS Configuration

Since the static web application is hosted using two buckets, one for HTML5 and CSS files, and another for storing 3D model assets, the browser's security model would likely block the 3D models from loading unless CORS headers are correctly configured on the S3 bucket containing the 3D assets.

Trade-offs: To resolve this, we would configure the S3 bucket storing 3D models to allow requests only from the web application's domain, ensuring secure cross-origin access while maintaining browser security policies.

3D E-Commerce Platform Architecture