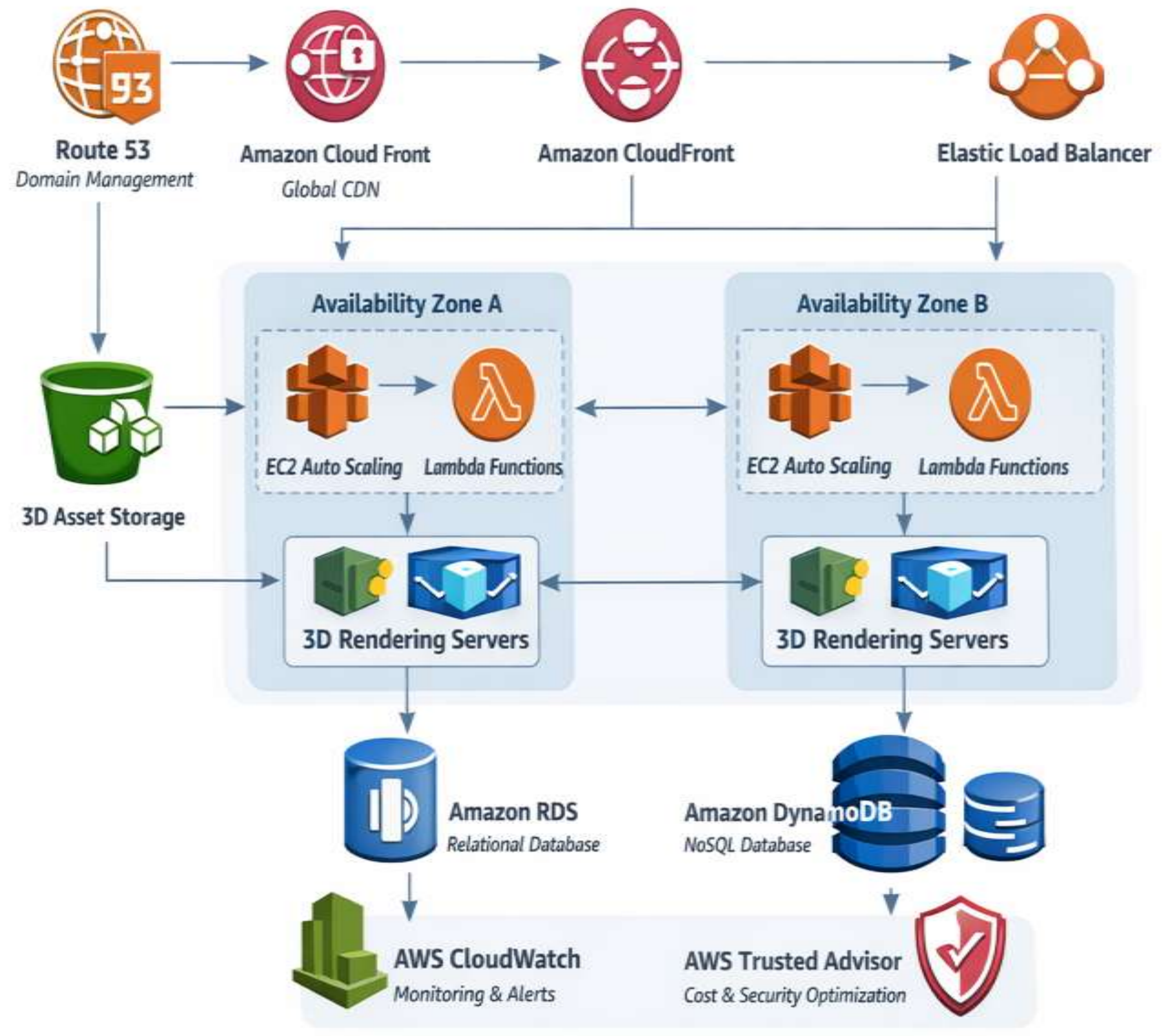
3D E-Commerce AWS Architecture

Scalable, Secure, High-Performance Platform



Summary

This architecture uses a globally distributed CDN, object storage for 3D assets, a mix of serverful and serverless compute for APIs and background jobs, managed databases for transactional and catalogue data, caching for low-latency interactions, and AWS security & monitoring services to meet the startup's needs for high availability, scalability, performance, security, and cost optimization.

Why each service was chosen?

- **Route 53** — Global DNS with health checks and routing policies (weighted / latency) to ensure reliable global reach and fast failover.
- **CloudFront** — Delivers static assets (3D models, textures, thumbnails, JS/CSS) from edge locations close to users to minimize latency and speed up 3D interactions.
- **Amazon S3** — Durable, cost-efficient storage for 3D assets (glTF, obj, textures). Use versioning, lifecycle policies, and S3 Intelligent-Tiering to keep costs down.
- **Elastic Load Balancer (ALB)** — Distributes API/websocket traffic across application servers running in multiple AZs for fault tolerance and sticky session support if needed.
- **EC2 Auto Scaling Group** — Hosts backend services that require longer-lived processes, WebSocket servers for real-time sessions, or GPU-enabled instances if server-side rendering is required. Auto Scaling adjusts capacity to demand.
- **AWS Lambda** — Serverless microservices for event-driven tasks (thumbnail generation, metadata processing, auth hooks) to reduce operational overhead and cost for spiky workloads.
- **Amazon RDS (Multi-AZ)** — Relational store for orders, payments, and ACID-consistent transactional data. Multi-AZ provides automatic failover.
- **Amazon DynamoDB (On-Demand/Auto Scaling)** — Fast, low-latency store for product catalogue reads, session data, and user preferences. On-demand mode reduces provisioning complexity for unpredictable traffic.
- **ElastiCache (Redis)** — Caches product data, sessions, and precomputed model metadata to reduce DB load and produce snappy UI responses.
- **SQS / Background Workers** — Decouple heavy jobs (asset processing, notifications) to improve responsiveness and allow horizontal scaling of workers.
- **AWS WAF & AWS Shield** — Protect the application and CDN endpoints from common web exploits and DDoS attacks.

- **IAM & KMS** — Centralized identity and encryption for least-privilege access and secure asset encryption at rest.
- **CloudWatch & Trusted Advisor** — Monitoring, alerts, and cost/operational guidance to keep SLA and optimize resource usage.

<u>How the architecture meets the 5 requirements?</u>

1. **High Availability**
    - Multi-AZ deployments for EC2 and RDS, ALB spreads traffic across AZs.
    - CloudFront caches at edge locations; Route 53 health checks with failover routing ensure resilient DNS.
    - Lambda + SQS provide retryable, durable background processing.
2. **Scalability**
    - Auto Scaling for EC2 adjusts horizontally; Lambda scales automatically for concurrent events.
    - DynamoDB on-demand or provisioned with auto-scaling handles sudden catalogue read traffic.
    - SQS decouples spikes into steady worker processing.
3. **Performance**
    - CloudFront serves 3D assets from edge locations, reducing cold load times.
    - ElastiCache stores frequently accessed metadata for sub-millisecond reads.
    - Client-side rendering (WebGL / glTF) keeps heavy rendering on user devices; server only handles preparation.
4. **Security**
    - Use WAF to block OWASP top 10 attacks; Shield for DDoS protection.
    - VPC, Security Groups, and Network ACLs isolate backend resources.
    - IAM least-privilege roles and KMS for keys; S3 bucket policies and signed URLs for secure asset access.
5. **Cost Optimization**
    - Use serverless (Lambda) for intermittent workloads and DynamoDB on-demand to avoid over-provisioning.
    - S3 lifecycle rules move cold assets to cheaper storage; CloudFront reduces origin egress costs.

    - Auto Scaling and spot instances (for non-critical workers) lower compute costs.
    - Trusted Advisor recommendations and CloudWatch cost-anomaly detection help control spend.

Design trade-offs and challenges

- **Client-side vs server-side rendering**: Rendering 3D models in the browser (WebGL) is far more cost-effective and scales to many users. However, it relies on user device capabilities. Server-side rendering (e.g., GPU instances) provides consistent visuals but is expensive and complex to scale.
- **Consistency vs scalability for data stores**: RDS gives strong ACID guarantees (needed for orders/payments) but is less horizontally scalable than DynamoDB. We use both: RDS for transactional data; DynamoDB for highly scalable read-heavy catalogue/session data.
- **Caching invalidation**: When assets or product metadata change, cache invalidation (CloudFront + Redis) must be carefully managed to avoid stale content while keeping latency benefits.
- **Large asset delivery**: Very large 3D assets and textures may still suffer from first-load latency. Techniques: progressive LOD assets, compression (DRACO for glTF), and background prefetching can mitigate this.

Deployment & operational notes

- Deploy infrastructure with IaC (AWS CloudFormation / Terraform). Use CI/CD pipelines (CodePipeline / GitHub Actions) to build, test, and deploy.
- Use CloudWatch logs + X-Ray for tracing slow API calls and optimizing hotspots.
- Regularly run Trusted Advisor checks and set budget alerts.