Yes — this diagram already reflects an AWS architecture that is **highly available**, **high-performance**, **secure**, and **cost-optimized**.  
Here’s how each part works, step by step, so you can see why it meets all four goals.

**1. High Availability**

**Goal:** Your application should remain available even if one server or an entire Availability Zone (AZ) fails.

* **Users → Route 53 (DNS + Health Checks)**
  + Route 53 monitors health of endpoints (ALBs) and routes users to healthy ones.
  + It can be set up with **multi-region failover** or **latency-based routing**.
* **Route 53 → CloudFront + WAF + Shield**
  + CloudFront is a globally distributed CDN, serving cached content from edge locations closest to the user.
  + If one AWS region fails, you can configure CloudFront origin failover to route to another.
* **CloudFront → ALB (Multi-AZ)**
  + The ALB spans at least two Availability Zones, so even if one AZ fails, traffic still flows to the other.
  + Health checks ensure traffic goes only to healthy instances.
* **ALB → App Tier (Auto Scaling EC2/ECS)**
  + EC2 or ECS tasks run in **multiple AZs** with auto scaling to handle varying traffic loads.
  + If one AZ’s compute fails, the other AZ’s compute keeps running.
* **Database Layer: RDS/Aurora Multi-AZ**
  + RDS synchronously replicates to a standby instance in another AZ for instant failover.

**2. Performance Excellence**

**Goal:** Low latency and fast response time, even under heavy traffic.

* **CloudFront CDN**
  + Serves static content (images, CSS, JS, API cache) from edge nodes, reducing round trips to origin.
* **ElastiCache (Redis)**
  + Caches frequent queries and session data to reduce DB load and speed up requests.
* **SQS (Async Jobs)**
  + Heavy or long-running tasks are queued, processed by worker nodes, keeping API responses fast.
* **Auto Scaling**
  + Scales up during peak traffic and scales down during low load to maintain performance.
* **S3 (Static Assets)**
  + Stores and delivers static files efficiently, offloading traffic from application servers.

**3. Security**

**Goal:** Defense-in-depth, protecting at network, application, and data layers.

* **WAF (Web Application Firewall)**
  + Filters malicious traffic (SQL injection, XSS, bots) before it reaches your ALB.
* **AWS Shield**
  + DDoS protection at network and transport layers.
* **Secrets Manager + KMS**
  + Stores credentials and keys securely, encrypted with KMS-managed keys.
* **Private Subnets**
  + App servers, databases, and caches are not publicly exposed. Only ALB and CloudFront are public-facing.
* **CloudTrail + GuardDuty**
  + CloudTrail records all API calls for auditing.
  + GuardDuty detects threats like compromised credentials or unusual API calls.

**4. Cost Optimization**

**Goal:** Achieve above while minimizing unnecessary spend.

* **CloudFront + S3** reduce expensive origin egress from ALB/EC2.
* **Auto Scaling** avoids over-provisioning by adding capacity only when needed.
* **Aurora Serverless v2 (optional)** scales DB capacity automatically.
* **Spot Instances** for stateless workers reduce compute cost by up to 90%.
* **VPC Endpoints** for S3 and CloudWatch reduce NAT Gateway data processing charges.
* **S3 Lifecycle Policies** move old data to cheaper storage tiers like Glacier Deep Archive.

✅ **Result:**  
Your application can survive failures, deliver fast responses globally, defend against attacks, and do it all while keeping AWS costs under control.