**Scalable, Global AWS Architecture for API Handling with Background Jobs**

**Overview**

This architecture is designed to handle **global traffic**, support **read/write APIs**, manage **background jobs**, and **integrate external APIs**. It ensures **high scalability, fault tolerance**, and **cost efficiency** using AWS-native services.

**Business Requirements**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Requirement** | **Solution Provided** | | Handle different types of read and write APIs | Separated Read API (Redis) and Write API (Kafka) components | | Handle background job processing for large data | Batch jobs implemented via AWS Batch / K8s Jobs | | Integration with external systems for product lists | Through API Gateway (Kong) routing to external API systems | | Manage global traffic with peak-hour scaling | CloudFront + Auto Scaling Group for scaling based on demand | | Ensure cost-effective and scalable infrastructure | Use of EC2 Auto Scaling, Serverless options, DynamoDB, Aurora, and WAF/Shield for optimized cost and protection | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Architecture Components**

**Global User Entry Point**

* **Route 53 + CloudFront**: Distributes user traffic globally with low latency.
* **AWS Shield & WAF**: Protects from DDoS and filters malicious requests using custom rules.

**Load Balancing & Web Layer**

* **Application Load Balancer (ALB)**: Routes incoming requests intelligently.
* **Auto Scaling Group (ASG)**: Hosts **NGINX + App Logic** servers that scale based on traffic (EC2).

**API Gateway Layer**

* **API Gateway**: Central API entry point.
  + Routes requests to internal APIs (read/write).
  + Interfaces with external APIs (e.g., product list services).

**Read/Write APIs**

* **Redis (Read API)**: Fast in-memory reads for frequently accessed data.
* **Kafka (Write API)**: Manages high-throughput write streams with durability.

**Background Jobs**

* **Batch Jobs** (AWS Batch or Kubernetes Jobs):
  + Processes large-scale data in the background.
  + Can be triggered via Kafka events or periodic schedules.

**External API System**

* Fetches external data (e.g., product catalogs) and routes back to API gateway.

**Database Layer**

* **Aurora**: Handles transactional workloads (e.g., order, user, payment info).
* **DynamoDB**: NoSQL for high-throughput operations and global tables.
* **Redshift (optional)**: For analytical workloads and BI queries.

**Scalability & High Availability**

* **Auto Scaling** ensures EC2/K8s pods grow or shrink with load.
* **CloudFront + ALB** distribute traffic evenly across regions/zones.
* **DynamoDB + Aurora Read Replicas** support scale-out read patterns.

**Cost Optimization**

* Use **Spot Instances** in ASG for batch jobs.
* Choose **on-demand vs reserved** based on app tier.
* **WAF filtering rules** reduce unwanted traffic to save backend resources.
* Use **CloudWatch** and **Auto-scaling policies** to prevent overprovisioning.

**Observability & Monitoring**

* Integrated with **Amazon CloudWatch**, **Prometheus**, and **Grafana** for:
  + App health metrics
  + Auto-scaling triggers
  + API latency and errors

**Conclusion**

This architecture is designed for **real-time APIs**, **global scale**, and **cost-effective operations**, making it an ideal setup for modern applications handling unpredictable traffic, heavy data operations, and external system dependencies.