1. High Availability (99.99%)

To achieve high availability, the architecture incorporates multiple layers of redundancy and failover mechanisms:

1. **Multi-AZ Deployment**:
    - All critical components are deployed across three Availability Zones (AZs) within the region, ensuring the system remains operational even if one AZ fails.
    - EC2 instances for the React and Svelte front-ends, the monolithic API, and media servers are distributed across AZs with **Auto Scaling Groups**, enabling dynamic scaling to handle traffic spikes and failures.
2. **Load Balancers**:
    - **Application Load Balancers (ALB)**: ALBs are used to evenly distribute incoming traffic to EC2 instances running front-ends and monolithic APIs, ensuring that only healthy instances receive traffic.
    - **Network Load Balancer (NLB)**: The NLB handles traffic for EKS-based microservices, providing low latency and fault tolerance for back-end services.
3. **Amazon RDS Multi-AZ**:
    - The RDS database is configured for Multi-AZ deployment, which automatically creates a standby instance in another AZ and promotes it to the primary role in case of failure.
4. **EKS Cluster**:
    - The EKS cluster spans three AZs, ensuring that microservices remain available even if an AZ becomes unavailable.
5. **Global Failover**:
    - Geographical redundancy is ensured by deploying the infrastructure in a **primary region** (Saudi Arabia) and a **failover region**. AWS services like **S3 Cross-Region Replication**, **RDS Read Replicas**, and **Route 53 DNS failover** ensure seamless transitions to the failover region.

These design choices collectively ensure minimal downtime and adherence to the high availability requirement of 99.99%.

---

2. Security

The design prioritizes end-to-end security for data in transit, at rest, and during processing:

1. **Data Encryption**:
    - All traffic is encrypted using **HTTPS** with certificates managed by **AWS Certificate Manager (ACM)**.

    - **S3 Buckets** are configured with **Server-Side Encryption (SSE)** using AES-256 or AWS KMS keys to secure static assets.
    - **RDS Encryption**: RDS uses KMS-managed encryption keys for data at rest, protecting sensitive information.
2. **Network Isolation**:
    - The entire architecture is deployed within a **VPC**. Subnets are segmented into private and public categories to ensure that back-end resources like databases and microservices are not exposed to the public internet.
    - Security groups are tightly configured to allow only the necessary traffic between resources (e.g., ALBs to EC2, EKS, and RDS).
3. **IAM Policies and Roles**:
    - IAM roles are used for EC2, EKS, and Lambda services, with least privilege access policies ensuring minimal permissions are granted.
    - **KMS keys** are managed with strict access controls, limiting their use to specific services.
4. **CloudFront Security**:
    - CloudFront, serving as the CDN for React and Svelte front-ends, enforces HTTPS and restricts direct S3 access, mitigating risks of unauthorized access.
    - Web Application Firewall (WAF) can be integrated with CloudFront to block common threats like SQL injection and DDoS attacks.

These measures ensure secure data transmission and storage, protecting media content and user data.

---

3. Performance

To deliver an optimal user experience, the architecture is designed to maximize performance and minimize latency:

1. **AWS CloudFront for Static Content**:
    - CloudFront provides a globally distributed caching layer for the React and Svelte front-ends, reducing latency by serving static assets from the nearest edge location to users.
    - Dynamic content is served through low-latency, highly available Application Load Balancers.
2. **Auto Scaling Groups**:
    - Auto Scaling Groups for EC2 instances (front-ends, APIs, and media servers) ensure resources scale dynamically based on demand. This prevents resource bottlenecks during traffic spikes.
3. **EKS for Microservices**:
    - Kubernetes on EKS ensures that microservices are containerized and can scale horizontally based on resource utilization.
    - Kubernetes' internal load balancing optimizes resource distribution among nodes.

4. **Multi-Region Redundancy**:
    - The primary region (Saudi Arabia) and the failover region ensure low latency for users by keeping resources geographically closer to them.
    - S3 bucket replication synchronizes static assets across regions, enabling faster failover with minimal impact.
5. **Database Performance**:
    - RDS Read Replicas improve database read performance by offloading read operations from the primary database.
    - Multi-AZ RDS ensures that database performance remains unaffected during maintenance or failover scenarios.
6. **Caching**:
    - Caching mechanisms in CloudFront and EKS improve content delivery times for static and dynamic resources.

How the Architecture Aligns with Requirements

Requirement	Feature in Architecture
High Availability	Multi-AZ deployment, Auto-Scaling Groups, Multi-AZ RDS, EKS cluster across AZs, Load Balancers, and health checks ensure 99.99% uptime.
Encrypted Video Traffic	HTTPS everywhere (ACM), secure media delivery, S3 bucket encryption (AES-256/KMS), and encrypted connections through TLS ensure data security.
Fast Static Content	AWS CloudFront with global edge caching ensures minimal latency for React and Svelte SPAs.
Geographical Redundancy	Failover region (e.g., AWS Bahrain), S3 cross-region replication, RDS read replicas, and Route 53 failover DNS routing.
Encapsulated Databases	Dedicated RDS databases or schemas for each client, encryption using KMS, and isolated VPC subnets provide secure and isolated database environments for special clients.

Cost Optimization	Auto-scaling, lifecycle policies for S3 storage, reserved instances, spot instances, and optimized resource allocation ensure the architecture fits the budget of $16k while maintaining high performance and redundancy.