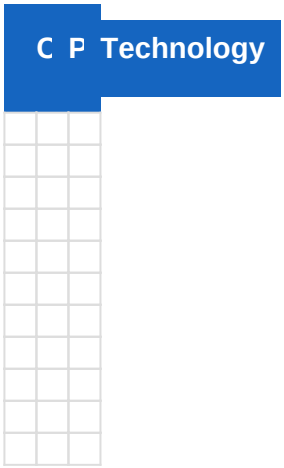
ATLAS System Design

Overview

A real-time chat application designed to support 100,000 concurrent users, enabling instant one-to-one and group messaging, user presence indicators, and message history persistence.

Components



Scalability

- **Horizontal Scaling of WebSocket Servers**: Add more WebSocket server instances behind the Load Balancer. Servers are stateless, making horizontal scaling straightforward to handle increased concurrent connections.
- **Distributed Message Queue**: Utilize a clustered message queue (e.g., Kafka) that can scale horizontally by adding more brokers and partitions to handle extremely high message throughput.
- **Database Sharding and Replication**: Implement sharding for the Chat Database (e.g., by conversation ID or user ID) to distribute data and read/write load across multiple database nodes. Replication ensures high availability and read scalability.
- **Caching Layer**: Implement a distributed cache cluster (e.g., Redis Cluster) to store frequently accessed data like user presence and recent messages, significantly reducing database load.
- **Microservices Architecture**: Each component (e.g., Presence Service, Notification Service, Authentication Service) can be developed and scaled independently based on its specific load requirements.
- **Load Balancing**: Employ multiple layers of load balancers (e.g., DNS-based, L4/L7) to efficiently distribute traffic to the API Gateway and WebSocket servers.
- **CDN for Media**: Use a Content Delivery Network (CDN) for serving user-uploaded media files, reducing latency and offloading traffic from the main application servers.