Devops - Assignment - 04

RollNo: 21111036
Class: MSc. C.S. B-21

Problem Statement: Architect WhatsApp like application for University premise.

Please consider below conditions:

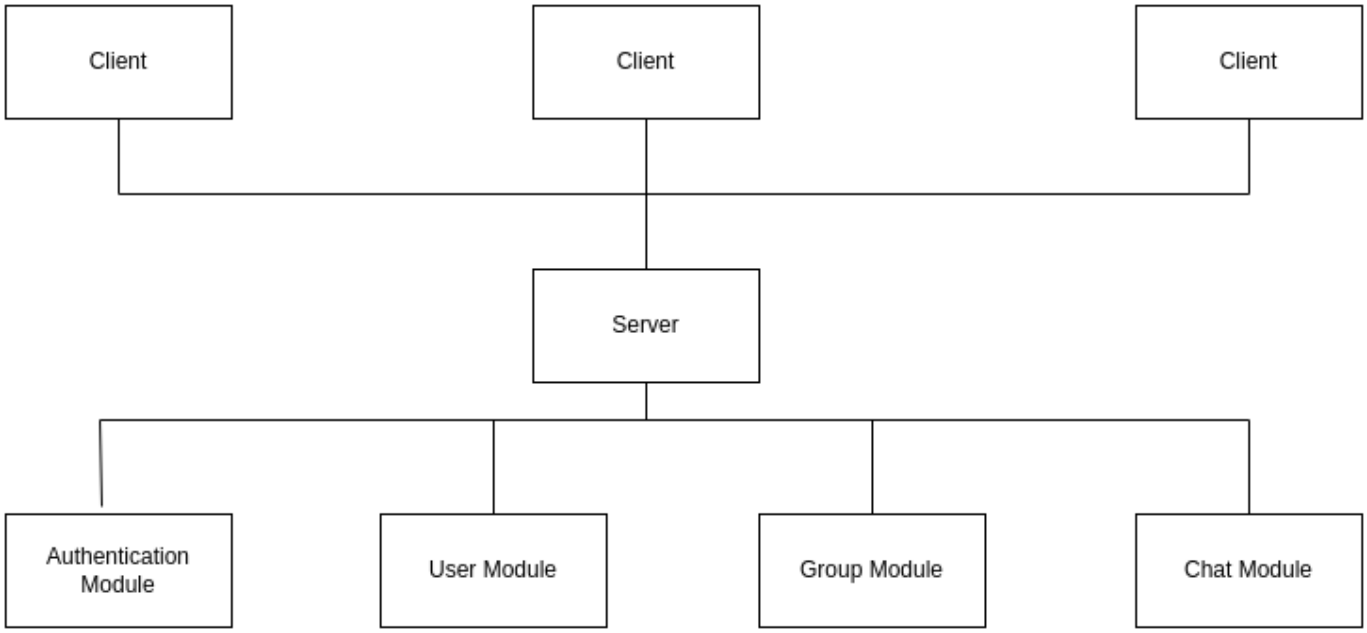
1. User limit 1000 users
2. Audio and Video call available
3. File Upload Limit : 100 MB
4. Group Limit: 150
5. Private and Public Groups Facility

Provide details in the below order:

1. Approach towards problem
2. Architecture Diagram
3. VM resources details i.e RAM, CPU & HDD used
4. How auto-scaling will be managed ?

   Ans.
   1. Approach towards problem:
   To architect a WhatsApp-like application as mentioned, we need to follow a microservices-based architecture approach.
   The application can be divided into different microservices such as user management, chat, audio/video call, file storage, group management, etc. By breaking down the application into smaller services, we can achieve better scalability, fault-tolerance, and resilience.
   For audio/video call functionality, we can leverage WebRTC technology that allows peer-to-peer real-time communication between clients. For file storage, we can use object storage services like Amazon S3, Google Cloud Storage, or Azure Blob Storage.

2. Architecture Diagram:



The architecture comprises of the following components:
- Clients: Users can access the application using client application on their devices.
- Server: A web server that can be used for routing requests to the appropriate microservices.
- Auth Module: Responsible for user authentication and authorization.
- User Module: Manages user accounts, profiles, and settings.
- Group Module: Manages group creation, membership, and settings.
- Chat Module: Manages chat messages between users and groups.

3. VM resources details i.e RAM, CPU & HDD used:
For all modules we can consider using the following requirements per module: 4 cores CPU, 8 GB RAM, 30 GB HDD. We can scale later according to the requirements.

4. How auto-scaling will be managed ?
We can leverage cloud-based container orchestration platforms like Kubernetes or Docker Swarm. These platforms provide built-in support for auto-scaling and can automatically scale up or down the number of containers running our microservices based on various metrics such as CPU usage, memory usage, or network traffic.

We need to set up a container orchestration platform like Kubernetes or Docker Swarm on a cloud provider like AWS, Azure, or Google Cloud.

We need to define scaling policies for each microservice that specify the conditions under which the number of containers should be scaled up or down. For example, we can define a policy that increases the number of containers for the Chat Service when the CPU usage of the existing containers goes above a certain threshold. When a scaling policy is triggered, the container orchestration platform will automatically spin up or shut down containers to adjust the resource usage of the microservice.

We need to monitor various metrics such as CPU usage, memory usage, and network traffic to determine when to scale up or down.

By leveraging a container orchestration platform and defining scaling policies, we can ensure that our WhatsApp-like application for the university premise can dynamically adjust its resource usage based on demand, ensuring optimal performance and scalability.