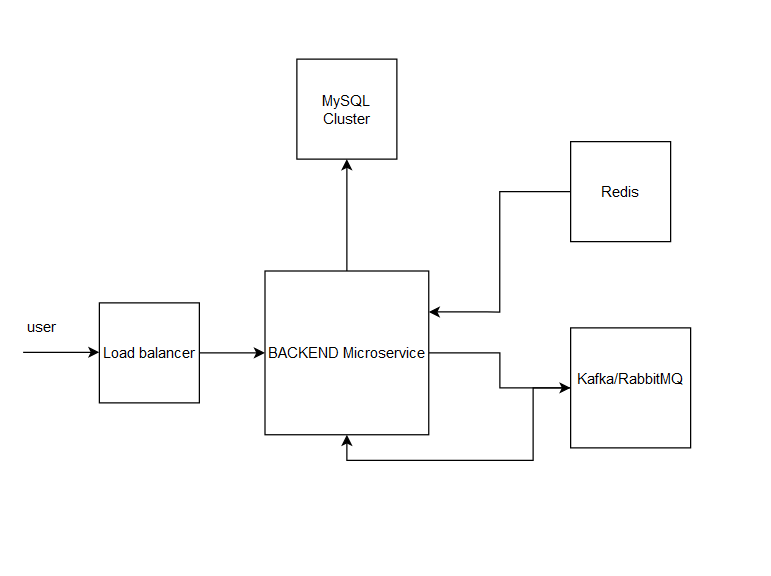
**Twitter Architecture**

Twitter is a read heavy website, so we should focus on the making the read operation as fast as possible. For that we’ll have to use distributed database (**MYSQL Cluster**) with proper **indexing**, we’ll have to use cache (**Redis**) for faster retrieval of the frequently asked data. Also, for live chat between users, we can use **web sockets** along with **RabbitMQ** or **Kafka** for channel subscription and identification.



User will hit the twitter endpoint to fetch details for home page (tweets from user’s followers), then the request will go to the load balancer (**kubernates on ECS Cluster**) , then the backend api will fetch all tweets from distributed database using date index for recent tweets.

If user want to chat with other user, then message sent by a user will pushed to message broker (**Kafka/RabbitMQ**), which will be then consumer to publish the message to the destination user channel.

Redis can be used here to save all the users subscription to the message channel so that they can be retrieved very fast to live chat. In this process we can also save the message in the database with the timestamp.

Whenever the user post any tweet, we’ll save that tweet in the database with some indexed tag, so that we anyone searches for tweets for particular topic, we can use that index to fetch the tweet faster.