Topic: Google Drive system(cloud storage/file sharing)

The Service
Service nodes are the ones that do the work a client is interested in. They receive tasks, process them and send back responses. Processing requests is usually costly so we imply the help of gateways and caches. Each service has a database. For the services, you'll need to implement the following features:

- ☐ Concurent tasks limit
- ☐ Status Endpoint
- ☐ Docker container
- ★ Task timeouts
- ★ priority system

The Gateway
The gateway is the node that receives and forwards user tasks to the service nodes. Based on some logic, the gateway caches responses and balances the load of service nodes. Finally, it has a service registry and chooses from registered services when load balancing. For the gateway service, you'll need to implement the following features:

- ☐ Load Balancing: Round Robin;
- ☐ Service Discovery
- ★ Circuit Breaker; Remove service if threshold is reached;

The Cache
A cache allows your system to temporary store responses given by your services and serve them without bothering the service nodes. Using caches makes your system more responsive. Usually caches use in-memory storage. For the cache service, you'll need to implement the following features:

- ☐ Keep-Alive Connection;
- ☐ Multiple Simultaneous Connections;

Languages: for the gateway Java. For the remaining components- Python.
Tools: For microservice db will be Postgresql, maybe will add mongoDB for cache part. IDEs Pycharm and Intellij. Git, Docker.

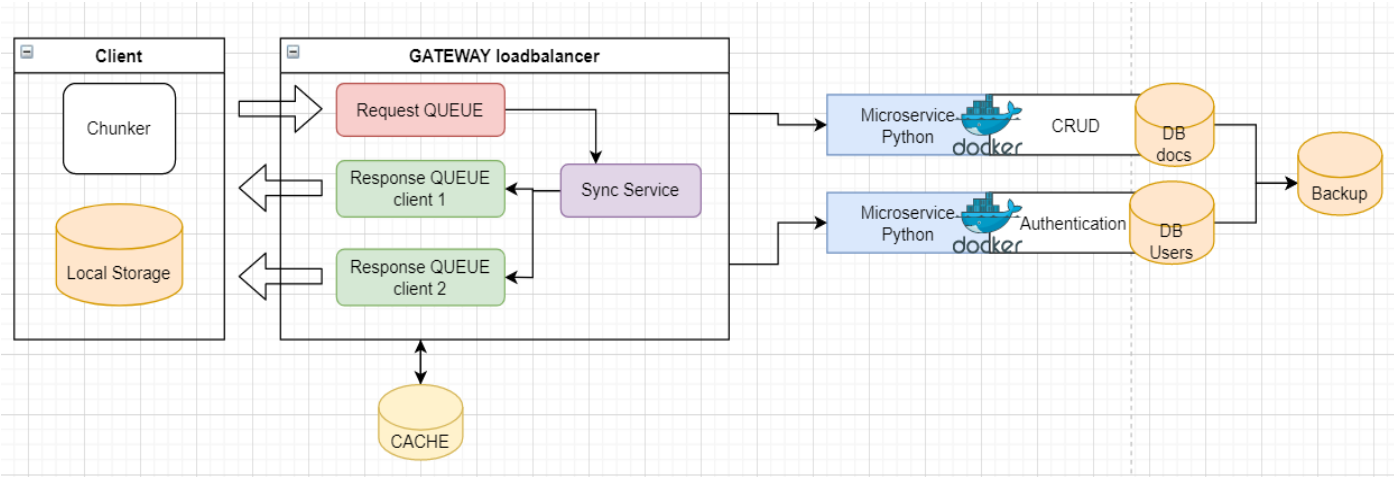
There will be 4 servers, by round robin algorithm on each new client connection this algorithm will iterate for choosing the next server. Ex: 1st client - 1st server, 2nd client - 2nd server … 5th client - 1st server, etc. Server's capacity can be set according to need, for the beginning it will be 100 clients/server. After the client connects to the server the communication with microservices begins through API requests. Outbound calls will be in form of commands for the user but in background they will be requests for connection with the gateway which will redirect to servers.

These requests will be such as getting a document, publishing one, deleting or updating the document/file (CRUD operations). also to add a new user, remove, update. The authentication ability, so a user will be able to “crud” only his/her files but not foreign ones.
Cache will serve as temporary storage for responses. Everything regarding simultaneous connections will be using threads/workers, priority will be achieved by synchronized queues.

For improvement if will have time remaining the “Chunking” can be done. It’s the division of a file/doc in multiple smaller parts which allows to send/receive file faster and doesn't become a bottleneck for the entire system. It also speeds up the response the client and server get. These data chunks are separated when sent and are re-created on arrival. Also the collaboration feature could be added for users to modify the same file but it’s more like google-docs. Also a good thing here will be addition of a backup storage which will contain all the data in case some service or main db fails

Disclaimer: I tend to modify/improve/add things while coding a system so the things described here may be significantly improved and described later more in depth. The solution arises not instantly, it takes time to understand and to code everything as it should be, with the possibility to argument why I choose this approach over the other, or why I decided to refuse initial idea.

Diagram:
https://drive.google.com/file/d/14MEdREjR0h82VZ_XT5c8jhzVcOSyh366/view?usp=sharing



Link to github (each part of the lab is on separate branch):
https://github.com/DivineBee/microservices-cloud