FAF.PAD16.1 Autumn 2021
Lab 2: Logically Linked DBs

Handed out:	Friday, October 22, 2021
Due:	Midterm 2

Introduction

You should continue working on your first lab. This time, your system will have 2 logically bound microservices. You will work on implementing distributed DBs and transactions. A few spoken requirements that are now written:

- Use at least 2 programming languages;
- Use Docker containers for each service and docker compose (or something else) for orchestrating them.

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:

■ Long-running saga transactions;

■ Database redundancy/replication + failover. Service instances connect to different DB replicas;

■ ELK stack or (Prometheus + Grafana for logging).

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:

■ Service high availability (if a request to a service fails or the service is otherwise unavailable, route the request to a different one);

■ Trip circuit breaker if multiple re-routes happen;

■ ELK stack or (Prometheus + Grafana for logging);

★ Cache high availability;

★ Microservice-based 2 phase commits including distributed transactions.

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:

- ★ Cache replication;
- ★ Distributed cache using consistency hashing.

Reporting

To ensure that no surprises happen at the presentation day, besides the final presentation / repo, you are required to do weekly status reports on your progress. Every week you will need to provide a status report in which you discuss your *tangible progress* on the lab. Tangible things include, but are not limited to:

- Commits to your code repo;
- Written research notes;
- Diagrams / drawings.

Make sure that all research materials end up in the project's readme file or in a folder called "docs". Don't forget to upload a link to your public repo of the project on Else, in the "Submit Lab" activity.

Grading

To ensure a better workflow during the whole "lab development" period, this lab will have 3 checkpoints, a presentation and a test based on the knowledge gained during the lab. These checkpoints will roughly follow the waterfall model of software development, but are open to discussion, so do discuss with your teacher if you would want to apply another approach. *You can pass at most one checkpoint per day*, so plan accordingly.

Passing the *first checkpoint* involves designing your system. You need to document the services, their API's and the technologies you plan to use during development (i.e. languages, protocols, frameworks etc.). A system's architecture diagram is also required.

Getting the *second checkpoint* implies that you have created an MVP that can receive tasks from clients and route them to service nodes via a gateway node, can process tasks on said service nodes, and can cache the responses on a cache node.

Lastly, passing the *third checkpoint* means that you've developed most of the features that you plan to present at the presentation.

To receive a mark for this laboratory work you need to receive a passed mark on all three checkpoints as well as successfully present your work during the presentation, as well as successfully pass the test.

Working in Groups

Working in groups is allowed but optional. You can work in groups of 1, 2 or 3 students. The bigger the group, the more features you are expected to implement. *The exact workload should be negotiated with your teacher*, before the first checkpoint.

Presentation

To be admitted to the lab presentation, you need to:

- Submit your repo link on Else;
- Have passed at least 3 checkpoints.

The presentation will follow a strict schedule. During the presentation, you are required to:

- State the mark you are aiming for;
- State the features you have implemented;
- Demonstrate the implemented functionality;
- Pass the individual code review;
- Discuss (if needed) Midterm 2 test and final mark.

Each team will have at their disposal 15/22.5/30 minutes to present their work (for teams of 1/2/3 members respectively). If by the end of the specified time slot you didn't finish presenting, the presentation is considered finished and I mark you (or not) based on what I have. Obviously, the individual code review has the most influence on passing / not passing.

There will be no extensions after Midterm 2.

Midterm 2 is a hard deadline for both labs.

Readings

[1] Keyang Xiang. "Patterns for distributed transactions within a microservices architecture". https://developers.redhat.com/blog/2018/10/01/patterns-for-distributed-transactions-within-a-microservices-architecture.

[2] Ben Lutkevich. "database replication". https://searchdatamanagement.techtarget.com/definition/database-replication.

[3] Elasticsearch. "What is the ELK Stack?". https://www.elastic.co/what-is/elk-stack.

[4] Cloud Native Computing Foundation. "Prometheus". https://prometheus.io.

[5] Grafana Labs. "Grafana". https://grafana.com.

[6] Redis. "Replication". https://redis.io/topics/replication.

[7] Juan Pablo Carzolio. “The Ultimate Guide to Consistent Hashing”. https://www.toptal.com/big-data/consistent-hashing.

[8] l3a0. “Distributing a Cache”. https://blog.baowebdev.com/2019/04/distributing-a-cache.

Good Luck!