SWA Final exam

[10 minutes]

Select all statements that are correct.

- ☐ A. Zipkin makes use of a circuit breaker
- ☐ B. For calls between Microservice A and Microservice B we use client side load balancing.
- ☐ C. The config service makes use of a circuit breaker
- ☐ D. Logstash makes use of a circuit breaker
- ☐ E. Zipkin makes use of a load balancer
- ☐ F. The API gateway makes use of a load balancer
- ☐ G. The config service makes use of a load balancer
- ☐ H. Logstash makes use of a load balancer
- ☐ I. In a stream based architecture you should always use CQRS.
- ☐ J. With the saga pattern we can implement strict consistency between microservices

[10 minutes]

Give **3 different and valid** reasons when it is a good idea to apply the CQRS pattern. (Be careful, only give valid reasons. If you give one or more reasons that are not valid you will lose some points)

- When you need fast read performance
- When your screens start to look different from your tables
- When you have different scalability requirements between queries and commands

[15 minutes]

a. Suppose one Kafka broker can handle 300.000 messages for a certain topic. The broker has simple not enough resources to handle more than 300.000 messages. But we need to handle 1.200.000 messages for this certain topic. It is important that all 1.200.000 messages go to the same topic. What technique can be used by Kafka so that Kafka can handle all 1.200.000 messages? You can mention only 1 technique. If you mention more than 1 technique you will lose points.

Partitioning

b. We can add as many consumers to Kafka topic as we want. Some consumes are fast, others are slow. Also it is possible that a consumer is not always online all the time. Explain clearly how Kafka makes it possible that we can add as many consumers to a Kafka topic as we want and all consumers get all messages? For this question make sure you only mention the techniques used by Kafka so that all consumers (fast consumers, slow consumers, temporarily offline consumers) get all messages. If you mention other techniques used by Kafka that have nothing to do with solving this problem you will loose points. Make sure you mention all techniques that are necessary.

Event sourcing and using an offset

c. In Kafka it is important that we never loose messages if a broker goes down. Explain clearly what technique can be used by Kafka so that we never loose messages if a broker goes down. For this question make sure you only mention the techniques used by Kafka to solve this problem. If you mention other techniques used by Kafka that have nothing to do with solving this problem you will loose points.

Replication

[20 minutes]

We learned that a monolith application can become very complex and that we always try to avoid complexity in our architectures.

We also learned that in a microservice we also run into a lot of complexity: communication complexity, transactions complexity, security complexity, resilience complexity, etc.

Suppose you need to design a large application with the following main requirements:

- The application is large and we need 4 scrum teams that will work on this application
- The application is mainly transactional, meaning that many use cases are transactional.
- Performance is crucial for this application. Almost all user actions are request-response type of actions and should be very fast.
- The system does not need to be scalable, there are only 400 users.
- We do not need fast deployment for this system.
- We need to avoid architectural complexity as much as possible

Describe clearly the architecture style(s) you use for this system based on the given requirements. Describe all architecture relevant choices you make to support the given requirements.

The number of points you get depends on how well you explain the architecture relevant choices you make and how these choices help to achieve the required requirements.

We learned that in a microservices architecture, we introduce a lot of complexity. Also with a SOA we have to deal with a lot of complexity. Both are distributed architectures and they are complex. To make things simple, we would use a monolith system. But then we have 4 different teams working on this monolith. To solve this problem we split the logic in the monolith into 4 separate components.

So the architecture to use is a component based monolith.

If your answer was a SOA or a microservice architecture, then you introduce too much complexity that is not needed.

[10 minutes]

a. Explain clearly the difference between tracing and logging.

b. Explain how tracing and logging in a monolith is different than tracing and logging in a microservice architecture.

Tracing: shows the larger overview of who is calling who and the corresponding details of these calls. Tracing is not application or domain specific.
Logging: Every application or service logs their own specific data. Logging is application or domain specific.

In a monolith you have no, or at least minimal tracing. Logging in a monolith is simple, you just log to one place (file) for the whole monolith.
In a microservice architecture you need to log and trace centrally in one place so that you can combine and relate the logging and tracing from different microservices together to get a good overview of what happens in the distributed microservices.

[15 minutes]

Suppose we have a microservice that has grown in size over time and we are considering breaking up this one microservice into two microservices.

a. Give 4 valid reasons when it is a good idea to break up this one microservice into two microservices. (Be careful, only give valid reasons. If you give one or more reasons that are not valid you will lose some points)

1. The one service is too large for one team
2. The one service is getting very complex
3. Different scaling requirements for different parts of the microservice
4. You want to use a different technology stack for different parts of the microservice
5. You have different architectural requirements for different parts of the microservice (security, transactions, type of database, etc)

b. Give 4 valid reasons when it is **NOT** a good idea to break up this one microservice into two microservices. (Be careful, only give valid reasons. If you give one or more reasons that are not valid you will lose some points)

All data access in this microservice needs strict consistent transactions

1. The one service can be maintained by one team
2. The one service is not very complex
3. There are no scaling requirements for different parts of the microservice
4. All parts of the microservice uses the same technology stack
5. The whole microservice has the same architectural requirements (security, transactions, type of database, etc)
6. You need strict consistence for all functionality of this service

[10 minutes]

Suppose your team is responsible for one microservice and your team decides to apply **event sourcing** together with **CQRS** in this microservice.

Explain clearly in which service you apply event sourcing. In all microservices? If you apply event sourcing only to one of the services, in which one of the services do you apply event sourcing?

Explain clearly why you made this choice.

When we apply event sourcing it becomes harder to get the current state. We need to replay all events in order to get the current state. This is a big disadvantage of event sourcing.

We can solve this problem with applying CQRS together with event sourcing. We use event sourcing at the command side, and then we store the current state at the query side. This way we have the advantages of event sourcing at the command side, and we also have fast performance at the query side.

[10 minutes]

Suppose we need to implement security in our microservice architecture

a. Explain clearly what information related to security needs to be specified in the central authorization service

Username, password and role(s) for every user

b. Explain clearly what information related to security needs to be specified in the individual microservices

Which roles can access which endpoints

c. Explain clearly what information needs to be stored in a JWT token in order to make OAuth2 + JWT work.

The role of the particular user (BTW, password is never placed inside the token)

[10 minutes]

Describe how we can relate a **microservice architecture** to one or more principles of SCI. Your answer should be about 2 to 3 paragraphs. The number of points you get for this questions depends how well you explain the relationship between a **microservice architecture** and one or more principles of SCI.