**[ 15 minutes]**

Explain clearly how the ELK stack works. Clearly describe the responsibility of the individual parts of the ELK stack.

Logstash: collect all the log messages from the individual services
Elasticsearch: NoSQL database that is optimized for fast searching to store the individual log messages
Kibana: Tool to visualize the log messages on a dashboard.

**[ 10 minutes]**

By default Spring Sleuth only sends 10% of requests to Zipkin. We can increase this value by setting the **sleuth.sampler.probability** property in the configuration file. Explain why Spring Sleuth by default only sends 10% of requests to Zipkin, and not all requests.

If we would send 100% of all tracing information to zipkin, we would use a lot of bandwidth only for tracing information and the system might get overwhelmed by the volume of additional rest calls.

**[ 10 minutes]**

Give 3 important advantages of client side load balancing compared to server side load balancing.

- No single point of failure
- Simplifies service management
- Only one hop (better performance)
- Every client can use its own load balancing algorithm
- Unlimited scalable

**[10 minutes]**

Suppose we have so many clients that talk to the API gateway that we need to apply load balancing to the calls between the clients and the API gateway. What kind of load balancing would be appropriate, client side load balancing or server side load balancing?

Server side load balancing. The clients only know the address of the API gateway. The client is often a JavaScript application running in the browser or a mobile app, and these applications don't know about the registry, and they don't know about a framework like ribbon.

**[10 minutes]**

a. Give 2 advantages of reactive systems.
   No blocking threads
   Performance

b. Give 2 disadvantages (or issues) of reactive systems.

The whole stack needs to be reactive
Complex to debug

**[ 20 minutes]**

We learned that kafka supports partitions. Explain clearly what problem(s) can be solved by using partitions in kafka. Make sure your answer makes it very clear for what kind of problem(s) you should use partitions in kafka.

Partitions help for load balancing the messages. The publisher can divide the messages over the different partitions, and then we can have different subscriber instances where each instance handles the messages from 1 partition.

**[15 minutes]**

We learned that we can have multiple consumers on the same kafka topic. Some consumers are faster than other consumers. Explain clearly why having multiple consumers on the same topic works perfectly fine in kafka. Explain the techniques used that takes care that every consumer gets all messages from a certain kafka topic.

For this to work correctly we need 2 techniques:

1. Event sourcing: messages will not be updated or deleted
2. Offset: each subscriber has its own offset.

**[15 minutes]**

Suppose your company has a large monolith application that contains 3 independent components. These 3 components are completely encapsulated and every component has its own database. The components talk to each other on their interface. There are 3 scrum teams, and every scrum team can work independently on their own component.

Your manager just followed a presentation about  microservice architecture, and your manager asks you if it would be a good idea to split up the large monolith application into 3 separate microservices.

Give **2 valid reasons** that would justify the choice to break up the monolith into 3 separate microservices.

Because we have already 3 independent components, we have already the following advantages

- Each scrum team can work on its own component
- We have already high cohesion/low coupling between the components

We learned that microservices have several disadvantages, so we need good reasons to justify a microservice architecture

Valid reasons are:

- If we have different scaling options for each component/microservice
- If we need each component to have its own build/test/release pipeline
- If we want to use different technologies in these different components

**[15 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.