Microservices Questions & Answers

1. **What are Microservices?**

More than one Spring Boot module/service integration ant they will communicate with each other.

Microservices or more appropriately Microservices Architecture is an SDLC approach based on which large applications are built as a collection of small functional modules. These functional modules are independently deployable, scalable, target specific business goals, and communicate with each other over standard protocols. Such modules can also be implemented using different programming languages, have their databases, and deployed on different software environments. Each module here is minimal and complete.

2. **What are the benefits and drawbacks of Microservices?**

Benefits:

Self-contained, and independent deployment module.

Independently managed services.

In order to improve performance, the demand service can be deployed on multiple servers.

It is easier to test and has fewer dependencies.

A greater degree of scalability and agility.

Simplicity in debugging & maintenance.

Better communication between developers and business users.

Development teams of a smaller size.

Drawbacks:

Due to the complexity of the architecture, testing and monitoring are more difficult.

Lacks the proper corporate culture for it to work.

Pre-planning is essential.

Complex development.

Requires a cultural shift.

Expensive compared to monoliths.

Security implications.

Maintaining the network is more difficult.

3. **Difference between Monolithic, SOA and Microservices Architecture.**

Monolithic Architecture: It is "like a big container" where all the software components of an application are bundled together tightly. It is usually built as one large system and is one code-base.

SOA (Service-Oriented Architecture): It is a group of services interacting or communicating with each other. Depending on the nature of the communication, it can be simple data exchange or it could involve several services coordinating some activity.

Microservice Architecture: It involves structuring an application in the form of a cluster of small, autonomous services modeled around a business domain. The functional modules can be deployed independently, are scalable, are aimed at achieving specific business goals, and communicate with each other over standard protocols.

4. **What is Actuator in spring boot?**

A spring boot actuator is a project that provides restful web services to access the current state of an application that is running in production. In addition, you can monitor and manage application usage in a production environment without having to code or configure any of the applications.

5. What is Semantic Monitoring?

The semantic monitoring method, also called synthetic monitoring, uses automated tests and monitoring of the application to identify errors in business processes. This technology provides a deeper look into the transaction performance, service availability, and overall application performance to identify performance issues of microservices, catch bugs in transactions and provide an overall higher level of performance.

6. **What is OAuth.**

Generally speaking, OAuth (Open Authorization Protocol) enables users to authenticate themselves with third-party service providers. With this protocol, you can access client applications on HTTP for

third-party providers such as Gamil, Linkedin, GitHub, Facebook, etc. Using it, you can also share resources on one site with another site without requiring their credentials

7. **Whats is Eureka in Microservices.**

Eureka Server, also referred to as Netflix Service Discovery Server, is an application that keeps track of all client-service applications. As every Microservice registers to Eureka Server, Eureka Server knows all the client applications running on the different ports and IP addresses. It generally uses Spring Cloud and is not heavy on the application development process.

8. **What is Circuit Breaker in Microservices?**

In a microservice based application, Circuit Breaker is a technique, where we stop executing an erroneous method and redirect every request to a custom method (Fallback method). Generally, we stop execution of a particular method if it is continuously throwing an exception. When we break the circuit, we also avoid any cascading failures to the downstream services. Its basic function is to interrupt current flow after a fault is detected. A circuit breaker can be reset to resume normal operation either manually or automatically.

9. **What is Hystrix Dashboard In Microservices Application?**

Netflix has provided a bunch of libraries to form an ecosystem in microservices. Like Eureka, Hystrix is also an open source library provided by Netflix in the Microservices space. Hystrix implements the Circuit Breaker pattern. You don't have to write the network or thread programming to handle fault tolerance in the Microservices. You need to use Hystrix library just by giving parameters and that's it. Hystrix is going to do all the work for you internally. The best part of it is that it works amazingly with Spring Boot. If you pick any Microservice based project, there are pretty good chances that they are using Hystrix.

10. **What are the states of Circuit Breaker?**

Depending on the state, Circuit Breaker changes its behavior.

A]Closed

If Client request is sent to the actual service method only, then it is called as CLOSED CIRCUIT. Hence, this state represents that the service is running properly and providing the expected functionality.

B]Open

If Client request is redirected to Fallback method, then such case is an OPEN CIRCUIT. Hence, this state represents that service is unavailable or faulty and error rate is beyond the threshold.

C]Half-Open

Once the state becomes OPEN, we wait for some time in the OPEN state. After a certain period of time, the state becomes HALF_OPEN.

During this period, we do send some requests to Service to check if we still get the proper response. If the failure rate is below the threshold, the state would become CLOSED. If the failure rate is above the threshold, then the state becomes OPEN once again. This cycle continues till the service becomes stable.

11. **What is Domain Driven Design?**

Domain-Driven Design is an architectural style based on Object-Oriented Analysis Design concepts and principles. It helps in developing a complex system by connecting the related components of the software system into a continuously evolving system. Domain-Driven Design is based on three core principles:

Focus on the core domain and domain logic.

Base complex designs on models of the domain.

Regularly collaborate with the domain experts to improve the application model and resolve any emerging domain-related issues.

12. **Zipkin-**

Zipkin is very efficient tool for distributed tracing in microservices ecosystem. Distributed tracing, in general, is latency measurement of each component in a distributed transaction where multiple microservices are invoked to serve a single business usecase.

Zipkin was originally developed at Twitter, based on a concept of a Google paper that described Google's internally-built distributed app debugger – dapper. It manages both the collection and lookup of this data. To use Zipkin, applications are instrumented to report timing data to it.

If you are troubleshooting latency problems or errors in ecosystem, you can filter or sort all traces based on the application, length of trace, annotation, or timestamp. By analyzing these traces, you can decide which components are not performing as per expectations, and you can fix them.

13. **Sleuth-**

Sleuth introduces unique IDs to your logging which are consistent between microservice calls which makes it possible to find how a single request travels from one microservice to the next.

Spring Cloud Sleuth adds two types of IDs to your logging, one called a trace ID and the other called a span ID. The span ID represents a basic unit of work, for example sending an HTTP request. The trace ID contains a set of span IDs, forming a tree-like structure. The trace ID will remain the same as one microservice calls the next.

Sleuth is a tool from Spring cloud family. It is used to generate the trace id, span id and add these information to the service calls in the headers and MDC, so that It can be used by tools like Zipkin and ELK etc. to store, index and process log files. As it is from spring cloud family, once added to the CLASSPATH, it automatically integrated to the common communication channels like –

requests made with the RestTemplate etc.

requests that pass through a Netflix Zuul microproxy

HTTP headers received at Spring MVC controllers

requests over messaging technologies like Apache Kafka or RabbitMQ etc.

14. **The Twelve Factors in Microservices**

I. Codebase

One codebase tracked in revision control, many deploys

II. Dependencies

Explicitly declare and isolate dependencies

III. Config

Store config in the environment

IV. Backing services

Treat backing services as attached resources

V. Build, release, run

Strictly separate build and run stages

VI. Processes

Execute the app as one or more stateless processes

VII. Port binding

Export services via port binding

VIII. Concurrency

Scale out via the process model

IX. Disposability

Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity

Keep development, staging, and production as similar as possible

XI. Logs

Treat logs as event streams

XII. Admin processes

Run admin/management tasks as one-off processes