Microservices Interview Questions

Description

1. What are Microservices

Microservices are small autonomous services that work together to achieve a business feature. Prior to microservices most of the applications were monolithic in nature, meaning the entire team used to work on a single and huge code base which was difficult to maintain.

Microservices ecosystem is a platform of services, and each service encapsulates a business capability/domain. Business capability means whatever a particular business does in that domain to fulfill its objectives and responsibilities. Microservices has its own independent life cycle. Developers can build, test and release each microservice independently. We can have small autonomous teams each responsible for one or multiple services. Contrary to general perception and â€˜microâ€™ in microservices, the size of each service matters least and may vary depending on the operational maturity of the organization. Read more here

2. What are advantages of Microservices architecture

Below are the advantages of microservices architecture.

1. **Technology Heterogeneity** – Number of microservices can vary from organization to organization. A single technology may or may not solve all the problem efficiently. So teams have freedom to use the technology of choice.
2. **Resilience** – Resilience means, if one component of the system fails, that failure does not cascade, and hence we can isolate the problems and the rest of the system can carry on working. In a monolithic service, if the service fails everything stops working, this can be avoided if we run the service on multiple machines to reduce the chance of failure. Deploying the application to multiple application servers or data centers.
3. **Scalability** – Scaling means increasing the number of instances of a microservice. This is beneficial when a particular feature/microservice has the maximum traffic.
4. **Ease of Deployment –** With microservices, we can make a change to a single service and deploy it independently without touching the rest of the services. This allows us to get our code

deployed faster. By chance If any problem occurs, it can be isolated quickly to an individual service, and rollback that application only.

For more extensive understanding Read more here

3. What are the disadvantages of Microservices architecture

1. **Increased Operational Efforts** – Due to more number of deployable microservice units, the efforts required for running and managing microservices is more compared to monolithic architecture. Appropriate monitoring is required as well as most of the operations must be automated to manage the microservices architecture.
2. **Difficult to change multiple microservices** – When there is a new feature to be implemented, and if it involves change in multiple microservices owned by multiple teams, the development and deployments efforts must be coordinated properly for the feature to work.
3. **Increased latency and failures** – Since Microservices communicate through the network, the latency can be higher sometimes or the communication between the services might fail.

4. What is loose coupling and high cohesion

Loose coupling and high cohesion are one of the core principles of microservices. Loose coupling promotes designing your system in such a way that each component of the system can be testable independently. The components are not tightly dependent with components in the system. If one component wants to use the services of other their dependency have to be injected wherever they are to be used.

High Cohesion means grouping related components in the system together. Lets take an example of Email class. It contains properties such as To, From, CC, BCC, Subject, Body, and may contain methods/features such as saveAsDraft(), send(), discardDraft(). But Login() feature should not be here, since there are a number of email protocol, and should be implemented separately.

5. What is single responsibility principle

In a microservices architecture, the single responsibility principle requires each microservice application to have just one focused responsibility.

6. What are client certificates

A client certificate is a digital certificate that enables a client system to perform authenticated requests, by verifying the requesters identity. Client systems cannot send requests to remote servers without a client certificate

7. What is service discovery in microservices

Service discovery is the process of calling the services by using the address provided by the Eureka server. All the services in the ecosystem register themselves with the Eureka server. When service A wants to call service B, it will get the address of any of the available instance from Eureka using the

service name registered i.e. A. Instead of hardcoding the dependent service direct URL in the microservice, we delegate the job of getting the service URL to Eureka service.

8. What is domain driven design in microservices

Domain Driven Design, also known as DDD is a framework to design software models by taking business domain models into consideration. **Because rules, definitions, and protocols vary across a large business domain model, domain-driven design suggests that we identify individual bounded contexts within our domain**, wherein rules, entities, and services are consistently applied. Bounded contexts can interact with each other, but they are separated by their own respective domain models that define their functionality.

9. What is consumer driven contract

Consumer driven contract, also known as CDC is a testing method which ensures the compatibility of services based on requirements defined by the consumers. The contract refers to an agreement between consumer and provider about the format of data transfer. CDC tests then are performed by both consumer and provider to ensure the contract is continually honored. PACT is an open-source tool that provides a CDC testing framework.

10. What is idempotence in microservices

Idempotence is when the services provides a consistent output in the case of duplicate messages. Idempotence is critical in ensuring consistent behavior from services, with no unintended side effects in the case that it receives duplicate requests.

11. Explain the working of microservice architecture

12. What is Spring Cloud

Spring Cloud provides tools for developers to quickly build some of the common patterns in distributed systems (e.g. configuration management, service discovery, circuit breakers, intelligent routing, micro-proxy, control bus, one-time tokens, global locks, leadership election, distributed sessions, cluster state). Coordination of distributed systems leads to boiler plate patterns, and using Spring Cloud developers can quickly stand up services and applications that implement those patterns. They will work well in any distributed environment, including the developerâ€™s own laptop, bare metal data centres, and managed platforms such as Cloud Foundry. Below are the features of Spring Cloud

- Distributed/versioned configuration
- Service registration and discovery
- Routing
- Service-to-service calls
- Load balancing
- Circuit Breakers
- Global locks
- Leadership election and cluster state

- Distributed messaging

13. What is the difference between Spring Cloud and Spring Boot

Spring Boot

- Using Spring boot we can develop standalone web applications, REST APIs and microservices quickly.
- It is an extension to Spring framework with added features like autoconfiguration, embedded servers like Tomcat, Jetty etc.
- Spring Boot addressed all the drawbacks of Spring framework.
- There is no need to use XML configuration
- The beans are initialized, configured and wired automatically.
- It provides a module called as Actuators, which we can provide the metrics and health of the application.

Spring Cloud

- Spring Cloud provides service discovery and registration, which allows itself to register into eureka server. and also lookup other registered services to access its features.
- It enables to maintain a centralized configuration server which contains the application configuration.
- It provides a dependency called as resilience4j , which enables our application to implement circuit breaker pattern.
- It provides load balancing features to efficiently use the available application instances.
- Spring Cloud Gateway is the API gateway designed for the Spring boot application.
- Spring Cloud Stream is another framework in Spring Cloud for building highly scalable event-driven microservices connected with shared messaging systems. This is used for stream processing

14. What is bounded context in domain driven design

Bounded contexts divide a system by domains. They do not have to be microservices. They can also be implemented as modules in a deployment monolith. If the bounded contexts are implemented as microservices, this results in modules that are independent at the domain and technical level. Therefore, it is sensible to combine the concepts of microservices and bounded contexts.

15. How do microservices communicate with each other?

We can use synchronous as well as asynchronous way of communication between microservices depending on the use case. REST HTTP, Grpc (Google remote procedure call) , WebClient can be used for synchronous communication and Kafka can be used for synchronous or asynchronous communication.

16. What is semantic monitoring in microservices architecture

Semantic monitoring in microservices architecture is a process of running automated tests in production environment frequently to check for performance issues, or to find any bug which impacts the overall business function. Semantic monitoring is done by either running automated tests or monitoring the application

17. What do you mean by distributed transaction

18. What is Eureka in microservice

19. How to implement service discovery in microservices

20. What are containers in microservices

Containers as the name suggests that it contains all the necessary dependencies and configurations required for a microservice to function, instead of depending on the server for its dependencies. Docker is one of the most popular way used to containerize a microservice.

21. What is circuit breaker pattern in microservices

When a particular microservice is calling another microservice which is down or facing outage, then it will return 500 internal server error. If the user is continuously calling the failed service then the resources are only getting wasted. Sometimes your service might be a victim of Denial of service (DoS) attack, where someone is calling bombarding your service purposefully so that it gets to a point where the service is out of memory and resources. This will cause your application to go down temporarily. Circuit breaker pattern can come to rescue if you implement it in your application. In order to prevent infinite calls to an application, we can restrict the number of calls by defining a request count per duration. For example, If the number call to a microservice reaches a threshold of 10 requests per minute (be it success or failure), then prevent the additional call for that time duration.

In order to implement this there are 3 states of an application.

- **CLOSED** – Closed state means circuit is closed and we are able to invoke the application successfully.
- **OPEN** – Open state means there is not live connectivity to the application and your call will not reach the application.
- **HALF OPEN** – If your application has been down for a while, the circuit will be in OPEN state. To test if the application has recovered from failure and if it is functioning properly, the circuit will be bought in HALF OPEN state to test the calls.

22. What is resilience4j in spring microservices

Resilience4j is an alternative to Hystrix which helps us to manage fault tolerance in microservice. Hystrix has been deprecated by the Spring. Since large part of the Spring community had implemented Hystrix in their application, the Spring team introduced Resilience4j. Apart from fault tolerance it offer much more features as listed below.

- **Rate Limiter –** Used to block too many frequent requests.
- **Time Limiter –** Set a time limit when calling remote operation.
- **Retry mechanism –** Automatically retry a failed remote operation.
- **Bulkhead –** Avoid too many concurrent request.
- **Cache –** To store results of remote operation which are costly.

Category

1. Interview

Date Created
March 15, 2021
Author
kk-ravi144gmail-com