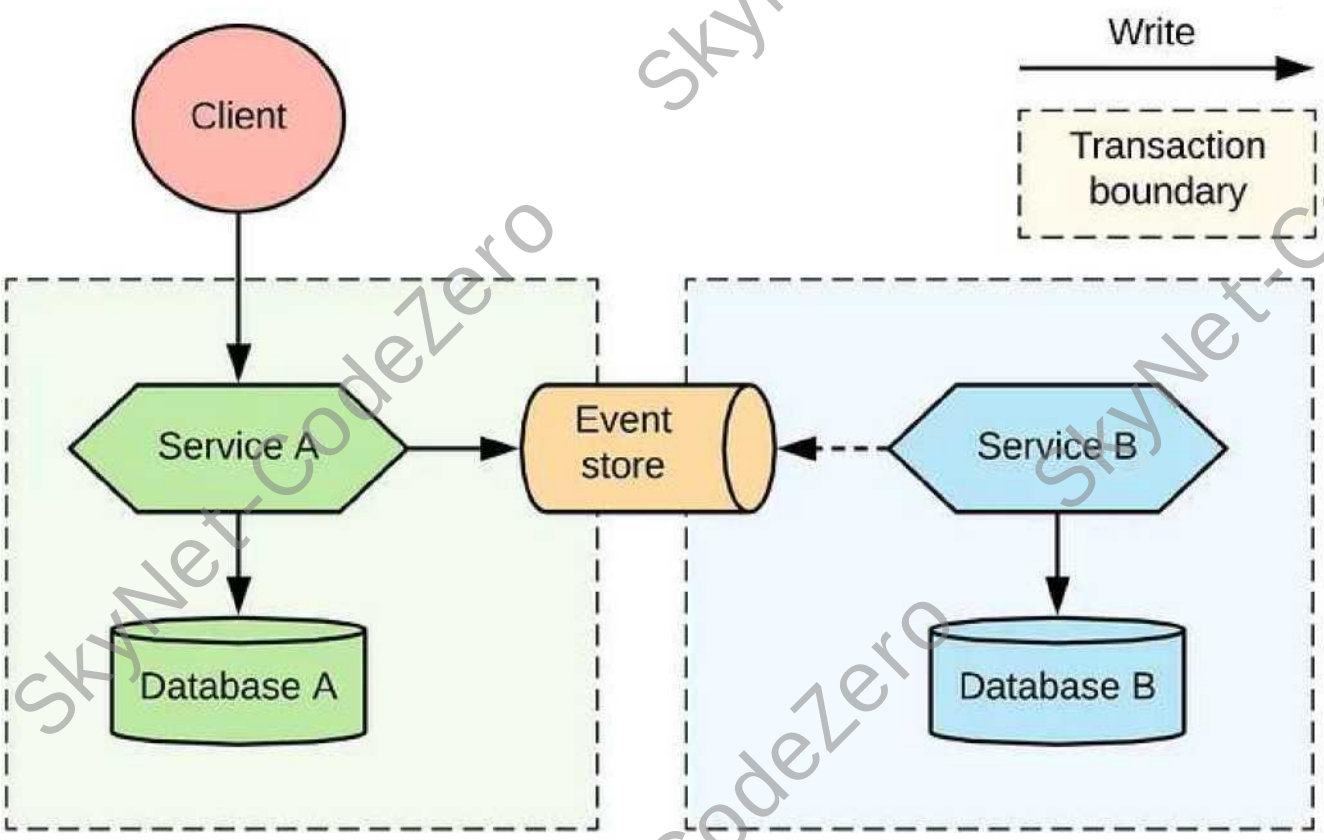
Top 10 Microservices Problem Solving Questions



Hey guys, as Microservice architecture becomes increasingly popular in this era of cloud computing and distributed systems, experienced developers need to have a deep understanding of how to tackle common challenges that arise in Microservice architecture.

Since we have all worked in Monolith and modules, things can not be done in the same way when you switch to Microservice architecture. Everything changes from development to debugging and from deployment to monitoring.

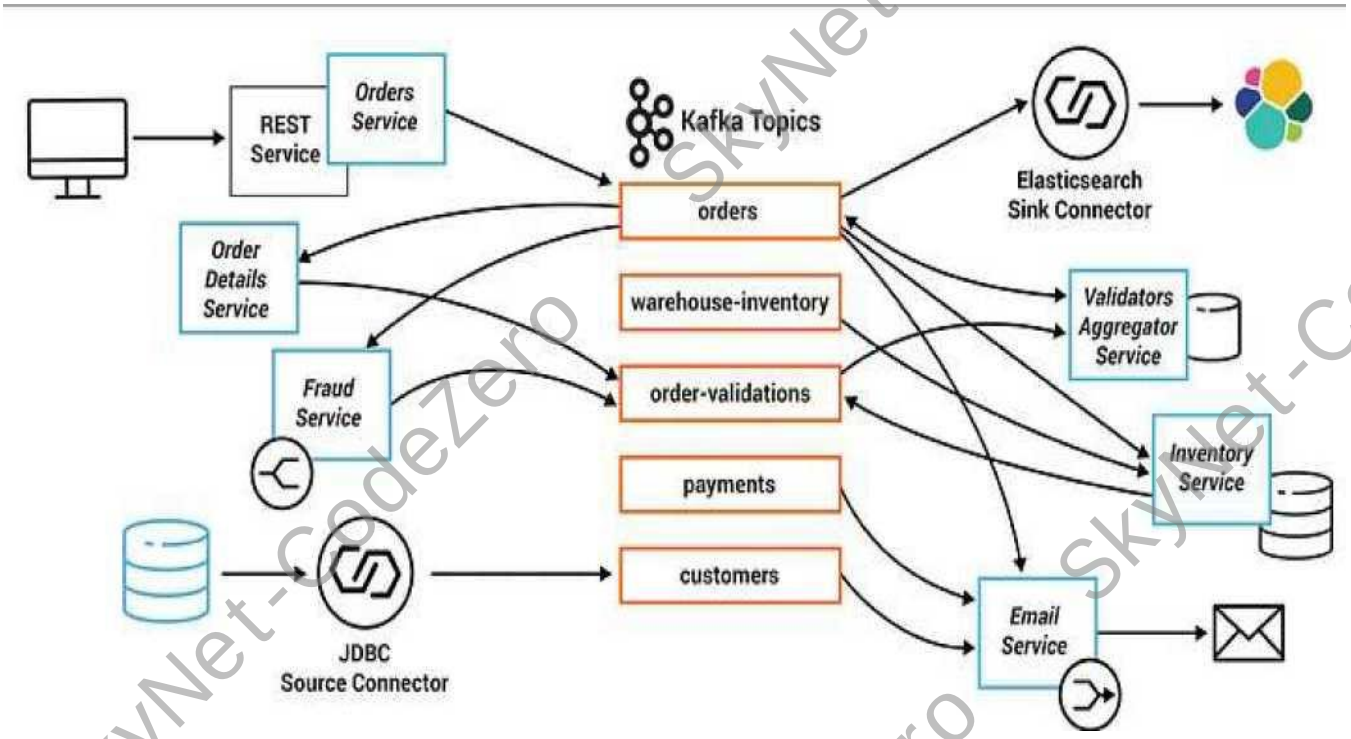
From performance issues to communication problems, there are a variety of complex situations that can arise when building and maintaining Microservices and those are also the thing which are being tested or asked during interviews.

1. Imagine you are working on a Microservice that's responsible for processing orders. However, due to some issues, it's currently down. What would you do ?

Answer: between the order creation service and the order processing service.Orders could be stored in the queue until the processing service is back up, at which point they could be picked up and processed. For message queue, you can either use RabbitMQ or Apache Kafka, both allows you to create asynchronous Microservices which are not hard dependent

on each other.

Here is how a Microservice architecture with Apache Kafka look like:



2. Suppose you have a microservice that's responsible for user authentication. How would you ensure that the service can handle a large number of requests and is highly available?

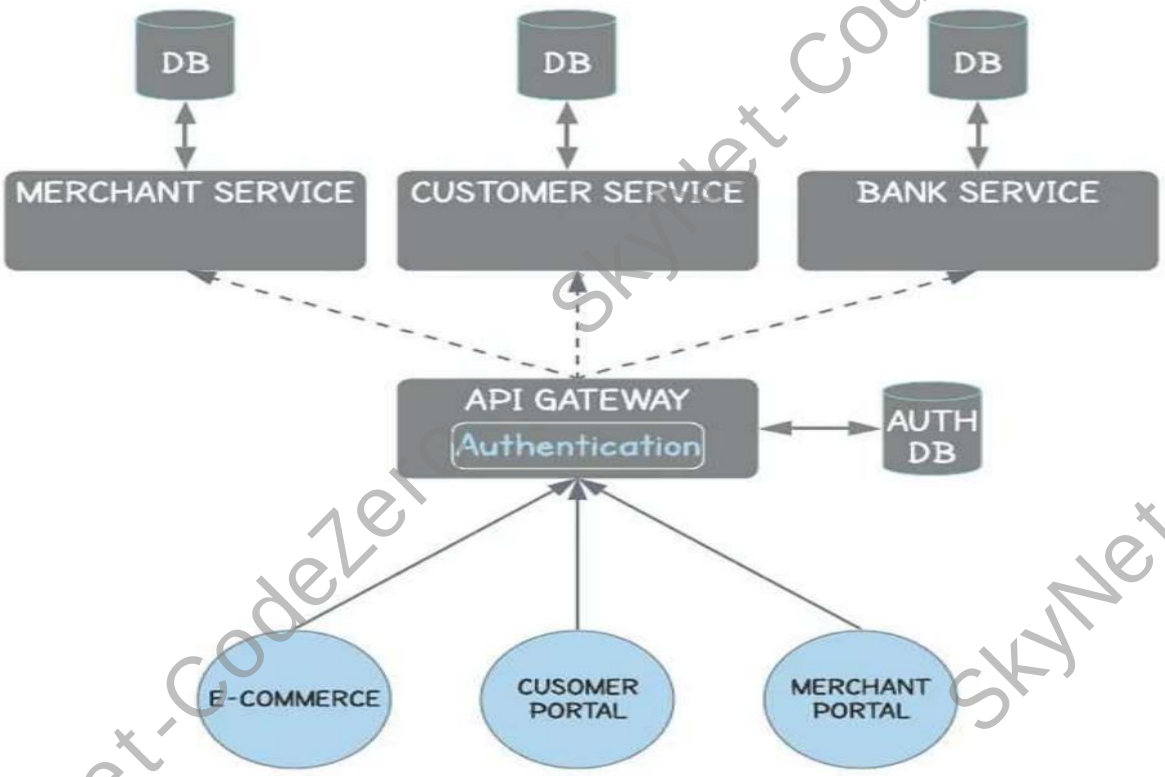
Answer: This question test your design skill for designing scalable and robust system which can handle millions of request. One possible solution would be to **use load balancing and clustering.**

The service could be deployed on multiple servers, with a load balancer distributing incoming requests among them.

Additionally, the service could **less,** meaning that each request can be handled independently without requiring access to a shared resource.

You can also use **API Gateway design pattern** to implement user authentication in Microservices architecture.

This is how a **API Gateway design pattern looks like:**



3. Imagine you are working on a microservice that's responsible for generating reports. How would you ensure that the reports are generated correctly and efficiently, while minimizing the impact on other services in the system?

Answer: One possible solution would be to **use caching and batching.** The service could cache previously generated reports and reuse them when possible, reducing the need to generate new reports from scratch.

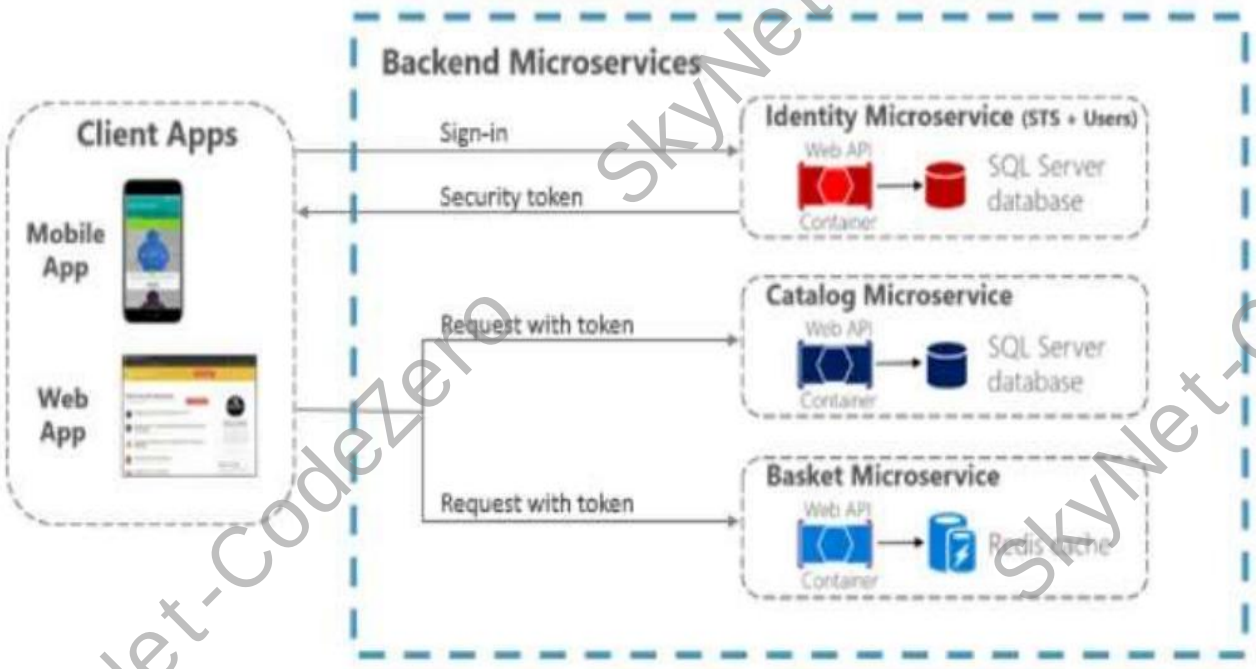
Additionally, the service could use batch processing to generate reports in batches, rather than on-demand, further reducing the load on the system.

4. Suppose you have a microservice thafs responsible for handling payments. How would you ensure that the service is secure and that sensitive payment information is protected?

Answer: If you have done any microservice interviews then you may know that Payment processing is interviewers favorite topic as this requires transaction management, security and you can't afford to lose data.

One possible solution of this problem would be to **use encryption and tokenization.** Payment information could be **encrypted** before being transmitted to the service,

ensuring that it's protected in transit.



Additionally, the service could use tokenization to store payment information in a secure manner, replacing sensitive information with non-sensitive tokens that can be safely stored and transmitted.

5. Imagine you have a Microservice that's responsible for handling user feedback. How would you ensure that the feedback is processed quickly and accurately while also minimizing the risk of spam and abuse?

Answer: This question test your skill about how do you protect your system from abuse. One possible solution would be to use a combination of automated and manual moderation.

Automated moderation could be used to filter out obvious spam and abusive content, while more complex cases could be flagged for manual review.

Additionally, the service could implement **rate limiting** to prevent users from submitting large volumes of feedback in a short amount of time.

6. Your team has built a new microservice that communicates with several other services. However, you notice that the performance of your microservice is slow. What could be the potential reasons for this, and what would be your approach to troubleshoot and resolve the issue?

This is another popular Microservice question as it touches on finding performance issues and solving them. There could be several potential reasons for slow performance in a microservice, such as:

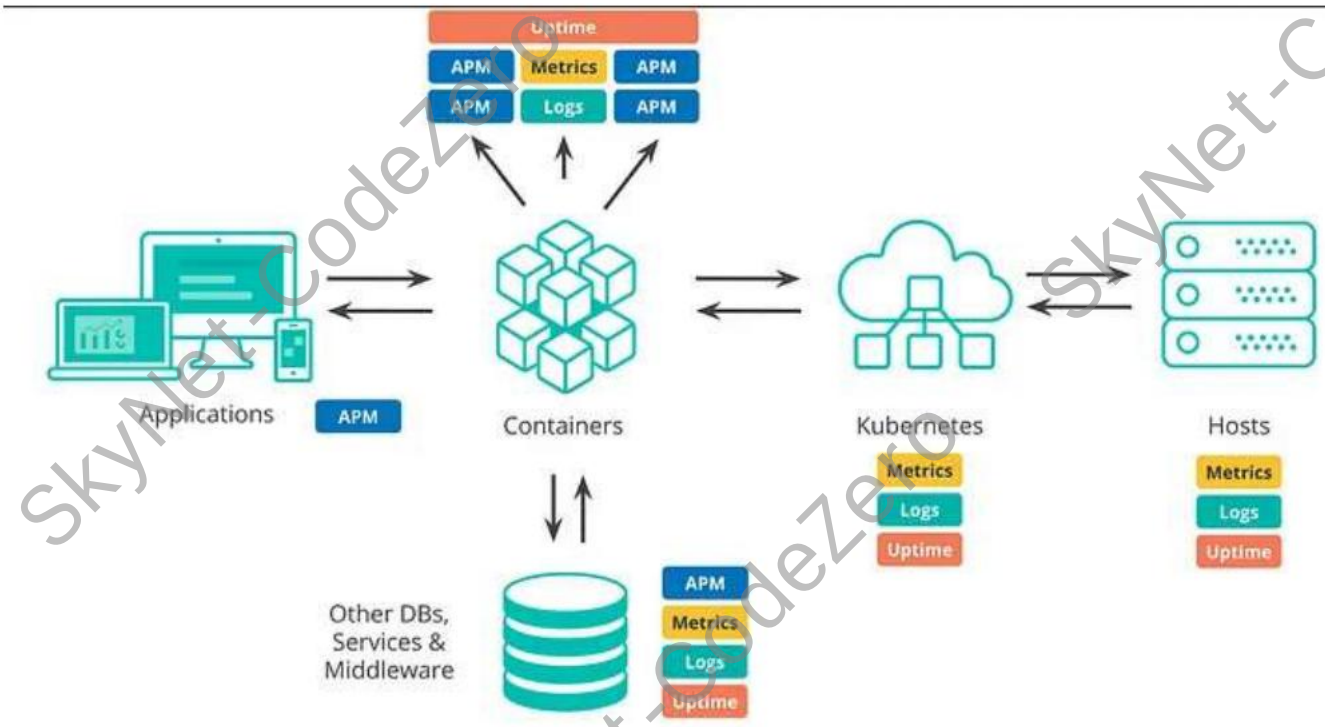
1. Network latency or poor network connectivity between services
2. Bottlenecks in the microservice code or database queries
3. Insufficient resources allocated to the microservice or its dependencies
4. **Inefficient communication** protocols or poor service design

To troubleshoot and resolve the issue, I would take the following approach:

1. Perform a thorough analysis of the microservice architecture, including the dependencies and communication protocols used.
2. Use appropriate monitoring tools to identify the bottleneck in the code, database queries,or network connectivity.
3. Use distributed tracing to identify latency or issues between the services.
4. Check the resource allocation and adjust them accordingly.

5. Optimize code and queries, if required.

Once the issue is identified, *we can fix it by modifying the microservice architecture, optimizing the code, database queries, and adjusting the resource allocation.* We may also consider redesigning the communication protocols or revisiting the service design if required. (Some open source APM(Application Performance Monitoring) tools :Apache SkyWalking,Apache HTrace.)



7. Imaging, you are working on a distributed system where a message is published to a message queue, and multiple services consume it. However,one of the services failed to consume the message. What would be your approach to identify the root cause of the issue and resolve it?

This question is also quite common on real world but don't answer in a way that our Support team handles this or we run a script to fix the problem etc. Interviewer is more interested in your technical solution.

As per my experience, you can take following approach to identify the root cause of the issue and resolve it:

1. **Check the logs** of the service that failed to consume the message to see `if` there are any

error messages or exceptions thrown.

2. **Check the configuration of the service** to ensure that it is properly configured to consume messages from the queue.

3. **Check the connection between the service and the message queue** to ensure that it is established and working properly.

4. **Check the message queue** to see if the message was correctly published and if it is still available for consumption.

5. If the issue is still not resolved, check if there are **any network or firewall issues** that could be preventing the service from consuming the message.

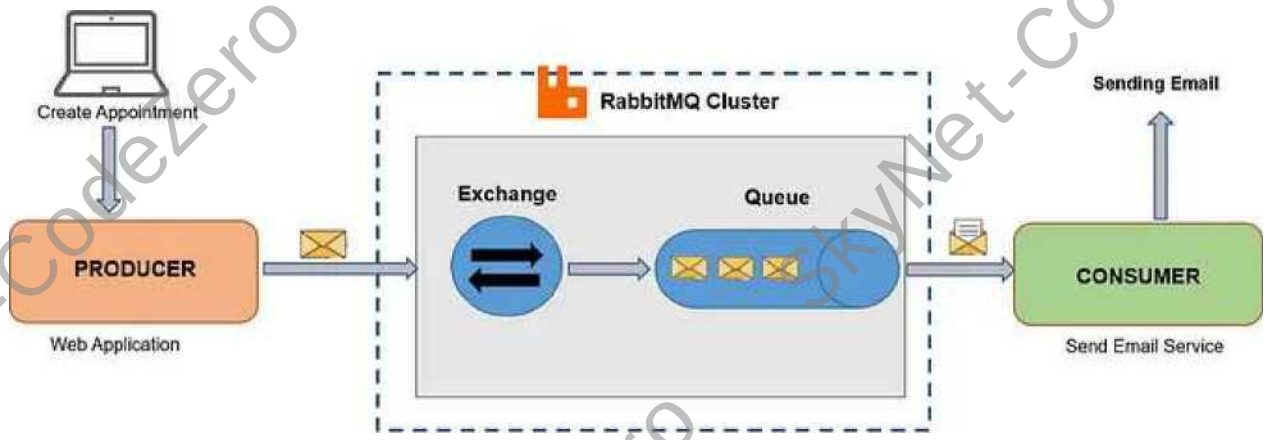
To resolve the issue, the following steps can be taken:

1. If the issue is due to a configuration or code issue, it can be fixed by updating the configuration or code of the service.

2. If the issue is due to a network or firewall issue, the relevant team should be notified to resolve the issue.

3. If the issue is due to an issue with the message queue, the queue should be investigated and any issues resolved.

4. If the message cannot be consumed, it should be moved to an error queue for further analysis and handling.



8. Consider a scenario where your team is developing a new Microservice that requires data (Static data or Market data) from another service. However, the other service has a different data schema, and you cannot change it. What would

be your approach to handle this situation?

This is also a common real-world problem which you will face when you work on any project. One approach to handle this situation is to use an adapter or transformation layer between the two services. This adapter can convert the data from the format used by the source service to the format expected by the target service.

Another approach is to **use a data integration platform such as Apache Kafka or Apache Nifi,** which can help to transform the data on the fly and make it available to the target service.

Alternatively, you could create a copy of the data in the target service's data schema and synchronize it with the source service on a regular basis using a data replication tool such as *Apache Flink* or *Apache Spark.*

It is important to ensure that the chosen approach **does not introduce any data inconsistencies or data quality issues.** Proper testing and validation should be conducted to ensure that the data is accurately transformed and integrated between the services.

9. Think of scenario where you are working on a Microservice architecture where multiple services are dependent on a shared database. However you noticed that one of the services is causing a deadlock in the database. What would be your approach to handle this situation and prevent it from happening again?

If one of the services is causing a deadlock in the shared database, the following approach could be taken to handle this situation and prevent it from happening again:

1. **Identify the cause of the deadlock**
   The first step would be to identify which service is causing the deadlock and

what queries are causing the issue. This can be done by analyzing the database logs and monitoring the database activity.

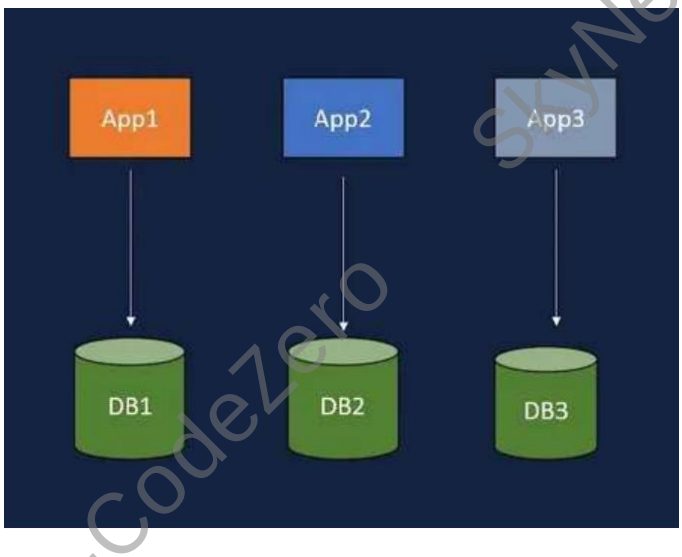
2. **Fix the deadlock**
   Once the cause of the deadlock has been identified, the next step is to fix it. This can be done by changing the queries, adding indexes, or optimizing the database schema. In some cases, it may be necessary to modify the code of the service causing the deadlock.

3. **Implement locking and concurrency controls**
   To prevent deadlocks from happening in the future, it is important to implement proper locking and concurrency controls. This can be done by using database-level locking mechanisms such as row-level locks, table-level locks, or database-level locks.

4. **Use distributed transactions**
   Another way to prevent deadlocks is by using distributed transactions. This allows multiple services to work together in a transactional manner^ ensuring that data consistency is maintained across all services.

5. **Perform load testing**
   Finally, it is important to perform load testing to ensure that the system can handle high loads and that the database can handle concurrent requests without causing deadlocks. This can help identify any performance bottlenecks and allow for preemptive measures to be taken.

To avoid these kind of issues, its also recommended that every service use their own database. There is also a pattern called **Database Per Microservices** which is worth knowing for any Java developer.



10. Suppose, your team is developing a new Microservice that communicates with several other services using RESTful APIs. However, the APIs are not well- documented, and you do not have access to the source code of the other services. What would be your approach to handle this situation?

Answer: This is also a common scenario which you will see when you work in real world application.

One possible approach to handle this situation would be to **use a tool like Swagger to document the APIs automatically.** Swagger can analyze the APIs and generate API documentation that can be shared with other **developers**. Additionally, **tools like Postman can be used to test the APIs** and get a better understanding of their behavior.

Another approach would be to *try to communicate with the owners of the other services and request for API documentation* or access to the source code. If that is not possible, then reverse engineering the APIs can be done by analyzing the network traffic and the response payloads.

This approach can help in understanding the behavior of the APIs, but it may not be very reliable or scalable.In any case, it is important to handle such situations with caution, as undocumented APIs can be prone to unexpected behavior or changes, which can affect the performance and reliability of the microservice architecture.

11. You are working on a microservice architecture where multiple services are communicating with each other using synchronous RESTful APIs. However you notice that the response time of the services is high, and the system is not scalable. What would be your approach to improve the system's performance and scalability?

There are several approaches that can be taken to improve the performance and scalability of a microservice architecture using synchronous RESTful APIs.

Here are some possible solutions:

1. **Implement caching**
   Caching can significantly improve response times and reduce the load on the system. By caching frequently accessed data or responses, you can avoid expensive round-trips to the services and reduce the response time.

2. **Use asynchronous messaging**
   Asynchronous messaging can improve scalability and reduce response times by allowing services to communicate without waiting for a response. By decoupling services using messaging, you can process requests in parallel and reduce the response time.

3. **Implement load balancing**
   Load balancing can distribute the load across multiple instances of a service and improve scalability. By using load balancers； you can ensure that requests are evenly distributed and avoid overloading any one service.

4. **Optimize the database**
   Database optimizations such as indexing, sharding, and partitioning can improve performance and scalability. By optimizing the database, you can reduce the response time and improve the scalability of the system.

5. **Use API gateways**
   API gateways can provide a central entry point for the services and can help with load balancing, caching, and rate limiting. By using API gateways, you can improve the scalability and reliability of the system.

6. **Consider microservice redesign**
   If the above solutions are not sufficient, it may be necessary to redesign the microservices to reduce dependencies and improve scalability. By breaking down monolithic services into smaller, more independent services,you can improve the scalability and performance of the

system.

Why Practice Scenario based Microservices Interview Questions?

As I said, Microservices have become an integral part of modern software development, especially for cloud native development, and experienced developers are expected to have a good understanding of the common issues and challenges that come with it and these scenario based questions gives you a chance to experience it, even if you have not worked in Microservice architecture extensively.

To excel in microservices development, developers should have a good understanding of distributed systems； design patterns, and architecture principles. They should also be familiar with various tools and technologies that can be used to build and manage Microservices.

One important tip for experienced developers working with microservices is to stay up to date with the **latest trends and best practices in microservices architecture.** They should also continuously improve their skills through training, certifications, and participation in community events.

By having a solid foundation in microservices and continuously learning and adapting to new challenges, experienced developers can successfully build robust, scalable, and reliable Microservice-based systems.

These scenarios and problem solving questions test a developer's problem-solving skills and understanding of microservice architecture principles, such as fault tolerance, scalability, and resilience.

Conclusion:

Thats all about the **common scenarios based Microservices Interview questions and answers** The scenario-based and problem solving Microservices questions discussed in this article will give you a glimpse of the real-world challenges you may face when working with microservices.

I have tried to cover common pitfalls and challenges like performance, security, integration as well as offered solution using caching, asynchronous messaging, message queues, design patterns and tools like Swagger and Postman.

WeChat:Artists_PinkFloyd.add me to your friends to get more skills.