**20 Microservice Interview Questions with Answers for Java Programmers**

Here is a list of some of the frequently asked Microservice questions from Java and Spring Boot interviews. You can use these questions to quickly review important Microservice concepts before interviews.

The list includes questions on Microservice architecture, the pros, and cons of Microservice architecture, Microservices vs Monolith architecture, and essential frameworks and tools required for Microservice development in Java,  like Spring Boot, Spring Cloud, Docker, and Kubernetes.

**1. What does "Microservices" mean? Explain the term Microservices?**  
Answer: Microservices is a Systems Development Life Cycle (SDLC) approach in which huge applications are built as a collection of small functional modules. These modules are deployed independently. They are scalable and can communicate with each other over standard protocols.

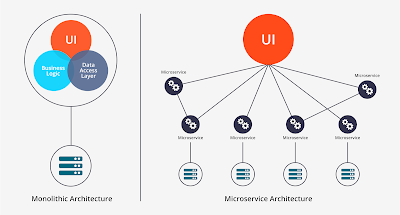
**2. What are the advantages of using Microservices?**  
Answer: If you have been doing software development then you know that Microservices is the future, the biggest advantage of Microservices is that it fits nicely in cloud infrastructure. By using containers like [Docker](https://javarevisited.blogspot.com/2019/05/top-5-courses-to-learn-docker-and-kubernetes-for-devops.html) it's easy to deploy and scale and it also makes the development easier.

Here are some key advantages of using Microservice architecture:

* They can be deployed independently.
* They are fault isolated.
* They are technologically diverse.
* Deployment time is less.

**3. Name the main components of Microservices?.**  
Answer: Containers, cloud infrastructure, API gateway, Service delivery, IaC, and Service bus.  
  
  
**4. How is Microservice architecture different from Monolithic architecture?**  
Answer: Monolithic architecture is tightly coupled (mostly) while microservice architecture is loosely coupled. Microservices focus on products while monolithic architecture focuses on the whole project. Moreover, service startups are faster in the microservice architecture.

If you want to learn more about Microservice architecture, I highly recommend[**the An Introduction to Microservice Principles and Concepts**](https://javarevisited.blogspot.com/2021/09/microservices-design-patterns-principles.html)course on Educative. This is a text-based interactive course that allows you to run code on a browser.

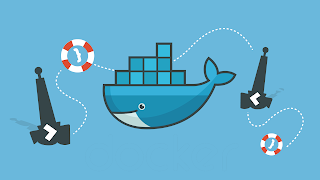
[](https://medium.com/javarevisited/7-free-microservices-courses-for-java-programmers-c9b2f3a2ea7d)

**5. What is the meaning of RESTful?**  
Answer: RESTful means Representational State Transfer web services. It's based on HTTP protocol and has been the backbone of modern web development which is highly based upon APIs.   
  
  
**6. What is the meaning of OAuth? And why is it used?**  
Answer: OAuth means open authorization protocol. OAuth is used to access the client applications on HTTP for third-party providers Facebook, GitHub, etc. On Java world, Spring Security supports OAuth 2.0 which you can use to secure your application. If you want to learn more about OAuth and Spring Security I suggest you join [**OAuth 2.0 in the Spring Boot Applications**course](https://medium.com/javarevisited/top-10-courses-to-learn-spring-security-and-oauth2-with-spring-boot-for-java-developers-8f0222d6066d) on Udemy. It's a great course to learn OAuth 2 in depth.   
  
  
**7. What are some challenges faced while using Microservices?**  
Answer: Here are some challenges faced on building applications on Microservices architecture:  
a) Being a distributed system, Microservice architecture is a heavily involved model.  
b) Microservices always need to communicate with each other because they always rely on each other.  
c) There are always operation overheads.

**8. What is the use of containers in Microservices?**  
Answer: Containers are used to manage microservice-based applications. They are easy and effective. Containers also help effectively in deploying and developing individually. The biggest advantages of containers are that they are easy to scale.

You can easily scale your Microservices using tools like [Kubernetes](https://javarevisited.blogspot.com/2020/06/top-5-courses-to-learn-kubernetes-for-devops-and-certification.html)which can manage containers at scale. Containers also make deployment uniform, for example, you can deploy a Microservice written in Java or any other programming language in the same way.   
  
  
**9. Explain end-to-end Microservices testing?**  
Answer: End-to-end microservices testing is a technique in which the entire flow of the application is tested using a business transaction. Such kind of testing covers the gaps left during other testing techniques such as unit and integration testing.  
  
  
**10. In what kind of application we should use microservices?**  
Answer: Microservices should be used in applications like web, desktop, mobile devices, Smart TVs, etc.

**11. What is Docker used for?**  
Answer: Docker provides a container environment that is used to host applications. It provides a static background for the app to run. Thus, preventing deployment issues. If you want to learn more about Docker and Kubernetes then I highly recommend you to join [**Docker & Kubernetes: The Practical Guide** course](https://medium.com/javarevisited/top-15-online-courses-to-learn-docker-kubernetes-and-aws-for-fullstack-developers-and-devops-d8cc4f16e773) by Maximillian Schwarzmuller on Udemy.

[](https://javarevisited.blogspot.com/2020/09/top-5-courses-to-crack-docker-certified-associate-certification.html)

**12. What is a "Client certificate"?**  
Answer: It is a digital certificate. It is used by client systems to make authenticated requests to any remote server. It is highly useful in mutual authentication designs as it provides strong assurances of a requester's identity.

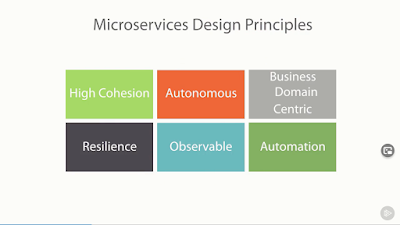
**13. How do independent Microservices communicate with each other?**  
Answer: Microservices can communicate with others through WebSockets for streaming, HTTP for request-response, or brokers.

**14. What are some common Microservices design principles?**

Microservices architecture is a better way of implementing Service-oriented architecture and following design principles are key for implementing Microservices applications

* High Cohesion
* Autonomous
* Business Domain Centric
* Resilience
* Observable
* Automation

If you want to learn more, I highly recommend you to check out the **[Microservices Architecture](https://medium.com/javarevisited/8-best-online-courses-to-learn-service-oriented-soa-and-microservices-architecture-94c01d6b94e6" \t "_blank)**[course](https://medium.com/javarevisited/8-best-online-courses-to-learn-service-oriented-soa-and-microservices-architecture-94c01d6b94e6" \t "_blank) by Rag Dhiman on Pluralsight.

[](https://javarevisited.blogspot.com/2021/09/microservices-design-patterns-principles.html)

**15. Give the major difference between Cohesion and Coupling?**  
Answer: Coupling is the relationship between two modules while cohesion is the relationship between two or more parts within a module.

**16. What are the disadvantages of using microservices?**  
Answer: While Microservices go hand-in-hand with modern Cloud infrastructure and they are easy to scale, it all comes with a cost of complexity.

Here are some of the notable disadvantages of Microservice architecture

* As a whole, microservices architecture is complicated.
* There is less control over third-party apps.
* Overall end-to-end testing is tough.
* Challenges while deployment.
* Accurate pre-planning is required.

Though the advantages offered by Microservices in a Cloud Computing environment outweigh these disadvantages, it's now becoming a standard way to develop a cloud-native software application.

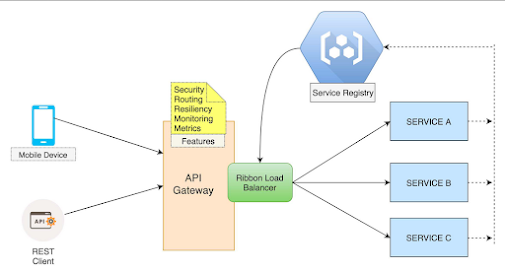
**17. What is difference between API Gateway and Load Balancer in Microservices? (**[**answer**](https://medium.com/javarevisited/difference-between-api-gateway-and-load-balancer-in-microservices-8c8b552a024)**)**

Load Balancer is an old concept which is used to distribute load or traffic across multiple instances. It can be implemented as a Hardware solution or as a Software Solution but distributing load is its main function and it can be used not just on Microservices architecture but also on Monolith and Service oriented architecture.

On the other hand, [API Gateway](https://javarevisited.blogspot.com/2023/08/how-to-implement-api-gateway-in.html) is a Microservice pattern which not only does load balancing but also can be used for lookup and can also simplify client code. Instead of remember 10s and 100s of Microservices, client can only remember API gateway host and port details.

You can also implement authentication, authorization, security, and other cross-cutting concern at API Gateway level instead of implementing them on each Microservice.

So, in short, API Gateway does a lot more than load balancer.

[](https://www.java67.com/2023/04/3-what-is-api-gateway-design-pattern-in.html)

**18. What are few ways to achieve synchronous communication between Microservices?**

There are many ways to implement synchronous communication in Microservices architecture like you can use REST API calls, gRPC or GraphQL API to retrieve data from server in synchronous manner and also to upload data to server.

**19. How will you achieve asynchronous communication between Microservices?**

Well, like synchronous communication there are multiple ways to achieve asynchronous communication in Microservices architecture like you can use Apache Kafka, RabbitMQ, and ActiveMQ. Message brokers allows senders to send and forget and receiver then can process when they are free giving both sender and receiver freedom to work on their speed.

**20. What is CQRS Pattern in Microservices? What problem does it solve? (**[**answer**](https://www.java67.com/2023/01/what-is-cqrs-command-query.html)**)**

This one is an interesting question as CQRS is an essential design pattern for Microservice architecture. CQRS stands for Command and Query System in which data operations are separated into two, command which writes data into database and Query system which reads data from database or data store. By splitting your application into two component you can better optimize your application depending upon whether its write heavy or read heavy application. You can read more about [CQRS Pattern](https://javarevisited.blogspot.com/2023/04/what-is-cqrs-design-pattern-in.html) here.