**24 microservices interview questions and answers**

**1. What is the microservices architectural style?**

The microservices architectural style is commonly used to build enterprise applications and distributed systems. It’s a software development approach that develops a web application through the collective efforts of several independent standalone applications, or microservices, that can be run on different machines. These individual microservices are highly specialized, and upon execution, form a single application.

**2. What are some advantages of microservice architecture?**

Advantages of microservice architecture include:

* Supports system scalability, testability, and maintainability
* Developer teams can focus and specialize on specific microservices
* Enables the setup of [continuous delivery pipelines](https://www.educative.io/blog/what-is-ci-cd-devops)
* Microservices often communicate using HTTP, so software developer teams can use their preferred tech stacks or combination of languages on the backend of their respective microservice applications
* As it prevents single points of failure, the entire application won’t crash if one component crashes
* Possibility of containerization
* Microservices are separate applications, and as such they can be deployed and scaled separately

**3. What are some disadvantages of microservices?**

Disadvantages of microservice architecture include:

* **New complexities**: While microservices help reduce complexity in the codebase, the communication and networking between components in the application becomes more complex (e.g. more difficult to track paths of communication because of interaction with all the separate components).
* **Infrastructural overhead**: Many resources are needed to set up microservice architecture. While upfront costs may be high, long-term savings are a benefit.

**4. What is loose coupling and high cohesion?**

Loose coupling and high cohesion are core principles in microservices design. Loose coupling advocates for the separation of unrelated components, while high cohesion advocates for the grouping together of related components.

**5. What is the single responsibility principle?**

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

**6. What’s the relation between microservice architecture and DevOps?**

DevOps is a methodology that combines practices from both software development and IT operations. It strives to reduce the system development life cycle. Both DevOps and microservices architectural style emphasize the importance of continuous delivery and agile development. As such, **the methodology and software development approach are complementary and often used together**.

**7. What are the advantages of microservices over monolithic architecture?**

Compared to monolithic architecture, the advantages of microservice architecture include:

* **Easier to manage than one large complex codebase**: Breaking the entire application into individual microservice applications makes it easier to test, debug, and scale the entire system.
* **Easier for teams to work on**: Development teams can focus on one application at time, instead of one large complex codebase
* **Better flexibility**: When traffic increases, only a small portion of the app needs to be scaled in times of increased traffic (eg. server).
* **Shorter startup time**: Microservices have a faster service startup than monolithic applications
* **Less dependencies**: Loosely coupled architecture reduces dependencies and supports system scalability, maintainability, and flexibility

**8. What are the differences between microservice architecture and service-oriented architecture (SOA)?**

While microservice architectural style is a variation of SOA, there are various differences between SOA and microservices architecture.

**Microservices**:

* More scalable
* No dependencies between business units
* Prioritizes decoupling
* Lightweight communication protocols such as HTTP and REST

**SOA**:

* Less scalable
* Dependencies between business units
* Prioritizes reusability
* Supports various communication protocols

**9. What is a client certificate?**

A client certificate is a digital certificate that allows a client system to make authenticated requests, by verifying the requester’s identity. Without a client certificate, client systems can’t send requests to remote servers.

**10. What is Kubernetes?**

[Kubernetes is a container management platform](https://www.educative.io/blog/kubernetes-tutorial#what-is-kubernetes) and open-source tool. It’s used to deploy and manage containerized workloads.

**11. What is Eureka?**

Eureka is the Netflix Service Discovery Server.

**12. What is service discovery?**

Service discovery is the automated detection of services and devices on a network. Service discovery helps keep download distributed appropriately between microservices during deployment, where the number of active microservice instances are in constant flux.

**Get hands-on with microservices today.**

Try one of our 300+ courses and learning paths: **Scalability and System Design for Developers**.

[Start learning](https://www.educative.io/path/scalability-system-design)

**13. What is a distributed transaction?**

A distributed transaction is a database transaction that involves **more than one physical server** or network host.

**14. What is domain-driven design?**

Domain-driven design is a software development framework that models software in accordance with the model of the underlying business domain. **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.

**15. What is end-to-end microservices testing?**

End-to-end testing evaluates a microservice application from start to finish through an entire business transaction. End-to-end microservices testing can become very complex with microservices.

**16. What is a consumer-driven contract (CDC)?**

CDC is a **testing method** that 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.

**17. What is idempotence in microservice architecture?**

Idempotence in a microservice is achieved if the service 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.

**18. What communication styles are used in microservices architecture?**

Synchronous request-based communication such as HTTP and REST are great for deploying services outside your microservice cluster or containerization platform. For internal communication between services, binary format communication mechanisms such as WCF using TCP and binary format are a good choice, or asynchronous message-based comm mechanisms like AMQP. Message formats can also be JSON, XML, and binary formats.

**19. What is Docker?**

[Docker is a containerization platform](https://www.educative.io/blog/beginners-guide-to-docker). It’s an open source tool developers use to deploy microservice applications.

**20. What is Hystrix?**

Hystrix is a latency and fault tolerance library. Hystrix helps manage latency and failures by isolating the points of access between microservices.

**21. What are RESTful web services?**

Restful web services are web services that follow the [REST architectural style](https://www.educative.io/blog/what-are-rest-apis). RESTful web services communicate with each other using [CRUD operations](https://www.educative.io/blog/crud-operations). The REST architectural style emphasizes scalability and flexibility, which makes it compatible and frequently preferred for microservice architecture.

**22. What is OAuth?**

OAuth means open authorization protocol. It’s a security standard that’s used to secure endpoints.

**23. What is Spring Boot?**

Spring Boot is a Java open-source tool that enables microservice application development. You can use the [Spring Boot application to streamline microservice development](https://www.educative.io/courses/developing-microservices-with-spring-boot) with the [Spring Framework](https://www.educative.io/blog/40-spring-interview-questions).

**24. What are the benefits of using the Spring MVC framework?**

The [Spring MVC framework](https://www.educative.io/blog/spring-framework-tutorial) has many useful benefits, including:

* Data binding
* Security and testing capabilities
* Annotation system
* Enables separation of concerns
* Helpful for developing [REST APIs](https://www.educative.io/blog/rest-api-interview-questions) and web services