**Basic Microservices Interview Questions:**

**1. What are Microservices?**

**Answer**:  
Microservices is an architectural style that structures an application as a collection of small, loosely coupled services, each responsible for a specific functionality, communicating over standard protocols like HTTP or messaging queues.

**2. What are the main benefits of using Microservices?**

**Answer**:

* Scalability
* Independent deployment
* Fault isolation
* Technology flexibility
* Better team ownership

**3. What are the challenges in Microservices?**

**Answer**:

* Distributed system complexity
* Data consistency
* Network latency
* Service coordination
* Monitoring and logging

**4. How do microservices communicate with each other?**

**Answer**:

* **Synchronous**: REST (HTTP), gRPC
* **Asynchronous**: Message Queues (RabbitMQ, Kafka)

**5. How do you implement service discovery in microservices?**

**Answer**:  
Using tools like **Eureka**, **Consul**, or **Zookeeper** to register and discover services dynamically.

**6. What is the role of an API Gateway?**

**Answer**:

* Single entry point
* Routing requests to backend services
* Load balancing
* Security (authentication/authorization)
* Rate limiting

Example: **Spring Cloud Gateway**, **Kong**, **NGINX**

**⚙️ Intermediate Questions**

**7. How do you handle centralized configuration in Microservices?**

**Answer**: Using **Spring Cloud Config Server** to manage properties of all services in one place (usually backed by Git).

**8. How do you handle fault tolerance?**

**Answer**: Using:

* **Circuit Breaker** (e.g., Resilience4j)
* **Retry**, **Fallback**
* **Timeouts**
* **Bulkhead Pattern**

**9. What is the difference between Monolithic and Microservices architecture?**

| **Monolithic** | **Microservices** |
| --- | --- |
| Single deployable unit | Independent deployable services |
| Tight coupling | Loose coupling |
| Hard to scale individual parts | Easier to scale independently |

**10. How do you secure microservices?**

**Answer**:

* **Authentication/Authorization** using **JWT** or **OAuth2**
* API Gateway to centralize security
* HTTPS for secure communication
* Role-based access control (RBAC)

**11. How do you implement inter-service communication with fault tolerance?**

**Answer**: Using **Feign Client** with **Resilience4j** or **Hystrix** for retries, timeouts, and fallback methods.

**12. How do you deploy microservices?**

**Answer**:

* Dockerize each service
* Use **Kubernetes** for orchestration
* CI/CD pipelines for automation

**13. What is a Sidecar Pattern?**

**Answer**:  
A helper container deployed alongside the main service container, providing supporting features like logging, monitoring, or service discovery (e.g., Istio proxy sidecar).

**🚀 Advanced Microservices Interview Questions**

**14. How do you handle transactions in Microservices?**

**Answer**: Use **Saga Pattern**:

* **Choreography**: Services publish and listen to events.
* **Orchestration**: A central orchestrator manages the flow.

**15. How do you handle versioning in Microservices?**

**Answer**:

* URI versioning: /api/v1/customers
* Header versioning
* Media type versioning

**16. Explain eventual consistency.**

**Answer**: Instead of strong consistency (like ACID), microservices prefer **eventual consistency** using **asynchronous events** or queues for data propagation.

**17. How do you monitor microservices?**

**Answer**:

* Use **Prometheus + Grafana**
* **ELK Stack** (Elasticsearch, Logstash, Kibana)
* **Zipkin** or **Jaeger** for distributed tracing
* **Spring Boot Actuator** for health endpoints

**18. Explain how CI/CD helps microservices.**

**Answer**: CI/CD automates the testing, building, and deployment of each microservice independently, enabling frequent, fast, and reliable delivery.

**19. What is service mesh in microservices?**

**Answer**: An infrastructure layer (like **Istio** or **Linkerd**) that handles service-to-service communication, security, routing, and observability outside the application code.

**20. How do you ensure backward compatibility in Microservices?**

**Answer**:

* Avoid breaking changes in APIs
* Maintain multiple versions
* Use contracts like **OpenAPI**/**Swagger**