**Docker**

docker build . -t docker\_account\_username/name:tag\_name

mvn spring-boot:build-image

docker run -d -p 8081:8080 reposityory\_name:tag\_name

docker ps

docker images

docker image push docker.io/reposityory\_name/tag\_name

docker compose up -d

docker compose down

docker run -p 3306:3306 –name db\_name -e MYSQL\_ROOT\_PASSWORD=root -e MYSQL\_DATABASE=accountsdb -d mysql

mvn compile jib:dockerBuild -> it will generate new docker image

**spring.sql.init.mode=always** => it is used to execute the script automatically by using schema.sql file

What are microservices?

Microservices is an architecture in which the application performs as a loosely coupled service that can be developed, deployed, and maintained independently. Each service in this architecture is called a Microservice.

Advantages:

* Easy to develop, test, and deploy.
* Increase agility.
* Ability to scale horizontally.
* Parallel development
* Modeled around a business domain.

Dis-Advantages:

* Complexity
* Infrastructure overhead
* Security concerns

**Monolithic V/s SOA V/s Microservices**

What is monolithic?

Back a decade, all the applications used to be deployed as a single unit where all functionality deployed together inside a single server. We call this architecture approach as Monolithic.

Advantages:

* Simpler development and deployment for smaller teams and applications
* Fewer cross-cutting concerns
* Better performance due to no network latency

Dis-Advantages:

* Difficult to adopt new technologies.
* Limited agility
* Single code base and difficult to maintain.
* Not Fault tolerance.
* Tiny update and feature development always need a full development.

**The SOA (Service-Oriented Architecture):**

|  |  |  |  |
| --- | --- | --- | --- |
| **FEATURES** | **MONOLITHIC** | **SOA** | **MICROSERVICES** |
| Parallel development | Low | Medium | High |
| Agility | Low | Medium | High |
| Scalability | Low | Medium | High |
| Usability | Low | Medium | High |
| Complexity & Operational Overhead | Low | Medium | High |
| Security Concerns & Performance | Low | Medium | High |

**Microservices Architecture Questions**

1. **What are Microservices?**
   * Define microservices and explain how they differ from monolithic architectures.
   * What are the advantages and disadvantages of microservices?
2. **What is Service Discovery in Microservices?**
   * How does **Eureka**, **Consul**, or **Zookeeper** help in service discovery?
   * How do microservices find each other at runtime?
3. **Explain the concept of API Gateway in Microservices.**
   * What is the role of an **API Gateway** in a microservices architecture?
   * Can you explain how **Zuul** or **Spring Cloud Gateway** works?
4. **What is Circuit Breaker and how does it work in Microservices?**
   * What is the role of a **circuit breaker** in microservices?
   * How does **Hystrix** or **Resilience4j** handle failure in microservices?
5. **What is Event-Driven Architecture in Microservices?**
   * Explain **event-driven microservices**.
   * How would you implement **event-driven communication** between microservices using tools like **Kafka** or **RabbitMQ**?
6. **Explain Database per Service in a Microservices Architecture.**
   * How would you manage data consistency across multiple microservices?
   * What are the strategies for handling distributed transactions in microservices (e.g., **SAGA**)?
7. **What are the common ways to manage distributed transactions in microservices?**
   * How does **Two-Phase Commit** work in distributed systems?
   * What is the **SAGA Pattern** and how do you implement it?
8. **How would you handle asynchronous communication between microservices?**
   * What message queues (like **RabbitMQ**, **Kafka**) have you used?
   * How would you manage message processing reliability?

**Spring Boot and Microservices Development Questions**

1. **What is Spring Boot and how does it simplify microservices development?**
   * How does **Spring Boot** provide auto-configuration, embedded servers, and reduced boilerplate code?
   * What are some of the common **starter dependencies** in Spring Boot?
2. **What are Spring Cloud features and how do they help in developing microservices?**
   * Explain **Spring Cloud Config**, **Spring Cloud Eureka**, **Spring Cloud Zuul**, and **Spring Cloud Gateway**.
   * How would you implement service discovery and API gateway using **Spring Cloud**?
3. **How do you implement security in a Spring Boot-based microservices architecture?**
   * How do you secure microservices using **JWT** (JSON Web Token)?
   * How would you implement **OAuth2** authentication and authorization in Spring Boot?
4. **Explain Spring Security and its role in microservices security.**
   * How do you handle **user authentication** and **authorization** in a microservices-based environment?
   * Explain how **OAuth2** works in Spring Security.
5. **What is Spring Data JPA and how does it fit in microservices architecture?**
   * How would you handle multiple databases in a microservices environment?
   * How do you implement **JPA/Hibernate** for persistence in Spring Boot?
6. **What are the best practices for Exception Handling in microservices?**
   * How would you implement **global exception handling** using **@ControllerAdvice** in Spring Boot?
   * How would you return a standardized error response across multiple services?
7. **How does Spring Boot handle service discovery and load balancing?**
   * What role does **Spring Cloud Netflix Eureka** play in microservices?
   * How does **Spring Cloud Ribbon** help in client-side load balancing?

**Microservices Design and Best Practices Questions**

1. **How do you manage inter-service communication in microservices?**
   * When would you use **REST** vs. **gRPC** vs. **message queues** for inter-service communication?
   * How do you manage data serialization/deserialization between microservices?
2. **What is idempotence in microservices, and why is it important?**
   * What would you do to make a service **idempotent**?
3. **How would you implement caching in a microservices-based application?**
   * Explain how you would use **Redis** or **EhCache** in a distributed cache across microservices.
   * What is the difference between **local cache** and **distributed cache**?
4. **What are polyglot persistence and when would you use it in microservices?**
   * How do you decide between **SQL** and **NoSQL** databases for microservices?
5. **What is CQRS (Command Query Responsibility Segregation) and how is it used in microservices?**
   * Explain how **CQRS** helps in decoupling reads and writes.
6. **What are the key differences between monolithic and microservices architectures?**
   * What are the advantages and challenges of moving from a monolithic architecture to microservices?
7. **What strategies do you use for scaling microservices?**
   * How do you handle **horizontal scaling** and **vertical scaling** in a microservices architecture?

**Microservices Deployment and DevOps Questions**

1. **How do you deploy microservices using Docker and Kubernetes?**
   * How would you containerize a Spring Boot microservice using **Docker**?
   * Explain how to deploy and manage Spring Boot microservices on **Kubernetes**.
2. **What is the role of CI/CD pipelines in microservices?**
   * How would you set up a **Jenkins** or **GitLab CI/CD pipeline** for a microservices-based Java application?
   * What are some challenges in **automating deployment** in microservices?
3. **How would you implement service scaling in a Kubernetes cluster?**
   * What tools would you use to **automate scaling** in Kubernetes, and how would you handle **auto-scaling**?
4. **How do you monitor and log microservices in production?**
   * What tools would you use for **centralized logging** (e.g., **ELK Stack**, **Graylog**)?
   * How would you use **Prometheus** and **Grafana** for **monitoring microservices**?
5. **How would you ensure resilience and fault tolerance in microservices?**
   * What is the role of **Hystrix** and **Resilience4j** in ensuring microservices' reliability?
   * How do you handle **timeouts** and **circuit breaking** in microservices?
6. **Explain the concept of blue-green deployment or canary releases in microservices.**
   * How would you manage **rolling updates** and versioning of microservices in production?

**System Design Questions**

1. **Design a URL shortening service using microservices.**
   * Explain how to handle scaling, consistency, and database design.
2. **Design a Notification Service for a large-scale microservices-based system.**
   * How would you decouple components, manage scalability, and implement retry mechanisms?
3. **How would you design a payment gateway using microservices?**
   * What components would you break the service into, and how would you handle high concurrency and transactions?
4. **Design a multi-tenant system using microservices.**
   * How would you manage data isolation, security, and performance for multiple tenants?

These questions cover a broad range of topics. To prepare:

* Review **hands-on coding** and practice deploying microservices using **Spring Boot**, **Docker**, **Kubernetes**, etc.
* Understand **design patterns**, **best practices**, and **scalability** for microservices.
* Do mock interviews with peers or platforms like **Pramp** or **Interviewing.io**.