[CQRS](https://medium.com/javarevisited/difference-between-saga-and-cqrs-design-patterns-in-microservices-acd1729a6b02)(Command Query Responsibility Segregation)

*It is a design pattern that suggests separating read and write operations into separate models to improve scalability, performance, and maintainability of the system. In a microservices architecture, CQRS can be applied to separate the command side (write operations) from the query side (read operations) of the system.*

*An example of using CQRS could be an e-commerce application. Let’s say that the application has a feature where users can search for products. In this case, the Command side would handle the creation, updating, and deletion of products. This would involve services that allow users to add products to the system, update product details, and delete products from the system.*

*On the other hand, the Query side would be responsible for handling user requests for product information. This would involve services that allow users to search for products based on various criteria, such as name, category, price range, etc.*

*Command side:*

*On the Command side, we can have a Spring Boot microservice that handles the creation, updating, and cancellation of orders. We can use Spring Data JPA to interact with the database and Apache Kafka to publish events.*

***Pros of CQRS:***

1. ***Scalability -*** *CQRS allows you to scale read and write operations independently. This is particularly useful in systems where read and write loads are significantly different.*
2. ***Flexibility*** *- Since commands and queries are separated, you can optimize data storage and retrieval strategies independently.*
3. ***Improved Maintainability*** *- CQRS simplifies the codebase by segregating concerns. This separation results in cleaner, more maintainable code, as commands and queries don’t interfere with each other.*
4. ***Enhance Security*** *- You to apply different security mechanisms to read and write operations. You can have stricter security controls on commands, ensuring that only authorized users can make changes.*

***Cons of CQRS****:*

*1. Increased Complexity - Implementing CQRS can introduce additional complexity to your system. You need to manage the flow of data between command and query models, potentially duplicating data for different models.*

*2. Eventual Consistency - CQRS can lead to eventual consistency issues, where query models may not reflect the most recent changes made by commands immediately. Dealing with this inconsistency requires careful handling and synchronization.*