# Problem statement

Messaging Queues are widely use in asynchronous systems. Message processing in an asynchronous fashion allows the client to relieve itself from waiting for a task to complete and, hence, can do other jobs during that time. It also allows a server to process it's jobs in the order it wants to.

Messaging Queues provide useful features such as persistence, routing and task management. We will be discussing the benefits of a message queue in future videos.

A system having a message queue can move to higher level requirements while abstracting implementation details of message delivery and event handling to the messaging queue.

The 'queue' is just a name for this data structure. In practice, it could be storing messages using any policy. Some examples of message queues are Kafka and RabbitMQ. They are widely used for various purposes such as command query request segregation (CQRS) and event sourcing.

# Publisher Subscriber model.

Microservices benefit from loose data coupling, which is provided by a publish-subscribe model. In this model, events are produced by a publishing service and consumed by downstream services.

Designing the microservice interactions involves event handling and consistency checks. We look into a pub-sub architecture to evaluate its advantages and disadvantages compared to a request-response architecture.

This type of architecture relies on message queues to ensure event passing. An example would be rabbitMQ or Kafka.

The architecture is common in real-life scenarios and interviews. If there is no strong consistency guarantee to be made for transactions, an event model is good to use in microservices.

Here are the main advantages:

Decouples a system's services.

Easily add subscribers and publishers without informing the other.

Converts multiple points of failure to a single point of failure.

Interaction logic can be moved to services/ message broker.

Disadvantages:

An extra layer of interaction slows services

Cannot be used in systems requiring strong consistency of data

Additional cost to the team for redesigning, learning, and maintaining the message queues.

This model provides the basis for event-driven systems.