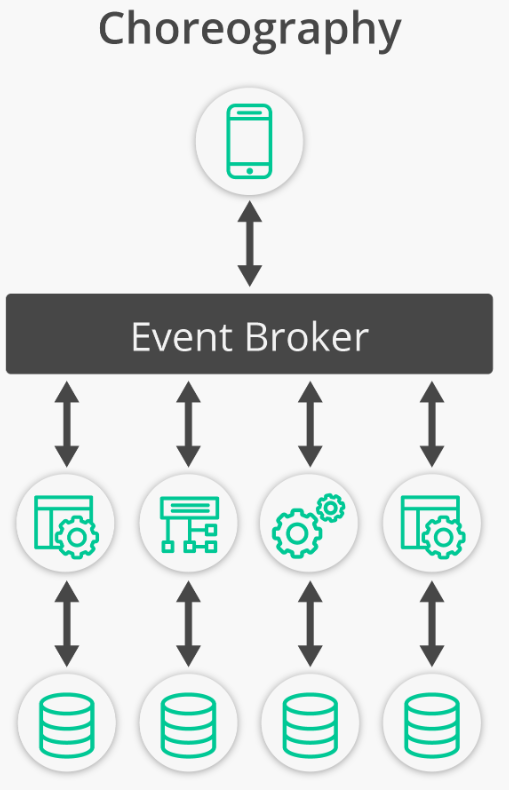
**Saga pattern**

Saga pattern is in a distributed system where 2 microservices need to communicate with each other. Managing transactions across multiple microservices is difficult and that’s where saga pattern comes into existence.

In saga pattern, we keep track of each transaction and if one transaction fails the the whole transaction fails.

There are 2 types of saga pattern implementations. The first one is chereography.

In chereography, each microservice communicates with each other through events. There is no centrelized coordinator.

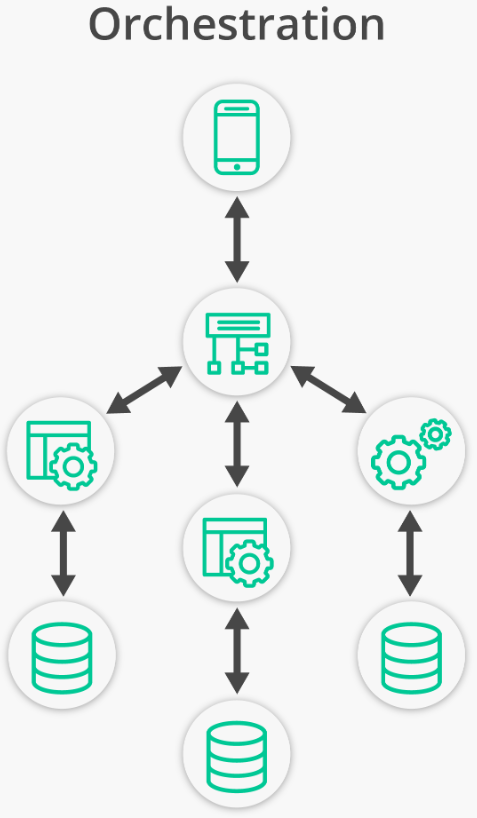


We can use message brokers, for instance RabbitMQ.

One of the pros of this approach over orchestration is that there is no single point of failure. Also it is easy to start with since we don’t need to set up a coordinator.

However, it gets complex in case to manage large number of microserivces.

The second implementation of saga pattern is orchestration.



The orchestrator tells each microservice which operation to perform. And in case of an issue, it also sends messages to different microservices for rollback of the transaction. It manages state of each task through a state machine.