**Microservices**

Services that do one single thing very well and interacts with other services using a protocol of communication.

5 Key Concepts

* High cohesion and low coupling
* Independent deploy ability
* Business domain modelling
* Observability
* Team organization

Advantages of Microservices:

* Autonomy of tech stack
* Robustness
* Ease of development and deployment
* Reusability
* Team performance

Disadvantages of Microservices

* Cost
* Tech overloaded
* Data consistency

Communication between microservices:

* Synchronous blocking communication
  + A microservice makes a call to another microservice and blocks operation while waiting for the response.
    - Rest over Http
    - RPC
* Asynchronous non-blocking communication
  + A microservice emitting a call can carry on processing whether the call is received.
    - Event Driven
      * Topic Based messaging system
      * Queue based brokers or RPC
    - Shared Database
      * Database file system

Transactions in Microservices:

* If one microservice has the database updated and the other is not, then it is important to either carry out the transaction in the other one, or to revert it back from the first one.
* Better approach is to use a two-faced commit.
  + Downside is that it is a long living commit problem, as it risky to hold an application for a transaction.
* Hence it best to use the SAGA approach.
  + Backward recovery
  + Forward recovery
* Orchestration
  + Command and control
    - One commander (service) controls the transactions all the time
* Choreography
  + Like event-based architecture. All services are emitting an event and other services kind of react to that event.
    - Con: Debugging is tough, if something goes wrong.
* Correlation-Id is used to debug the failure case and find out the culprit.

<https://lucid.app/lucidchart/73a0c130-5392-44ba-bb3f-2df9702b660b/edit?invitationId=inv_2d8e6c79-7f79-4ed8-97cc-8665202308e7&page=0_0#>

Event Based Communication

* Two microservice communicate with each other with the help of message broker.
* Event is the payload that is sent over the message on the message broker.
* Each event can be consumed by one or more services.
* But there can be a scenario where an event can have data which is not required by one of the microservice but required by the other one.
  + Here there are two options, first one is to split the data into two different events and send the relevant events to respective microservices.
  + The second option is to make the event with the data that is available to both and send it as a message. And if a service requires some more data, it will ask via another API request.
* Message Broker
  + Topic Based
  + Queue Based
* Implementation issues
  + Lost messaged
  + Order of messages
  + Consumer at fault
* Solution includes
  + Retrials, Replays
  + Retention of messages
  + Sending same messages twice