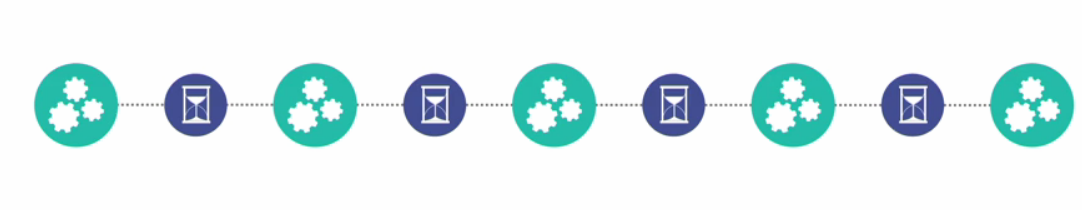
Micro Services

Problems

* May have high latency
* Since there are many components, one component going wrong can harm others as well

Service calling another service. There could be more delays



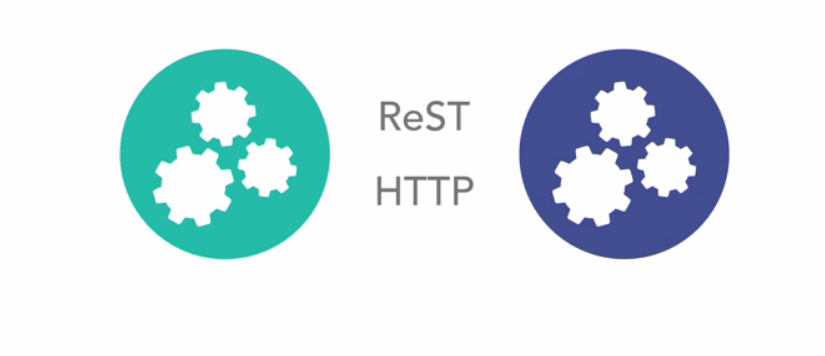
Circuit Breaker as Solution

Timeout logic when there is more delay

Circular Call may cause a problem



Communication in micro services is through HTTP



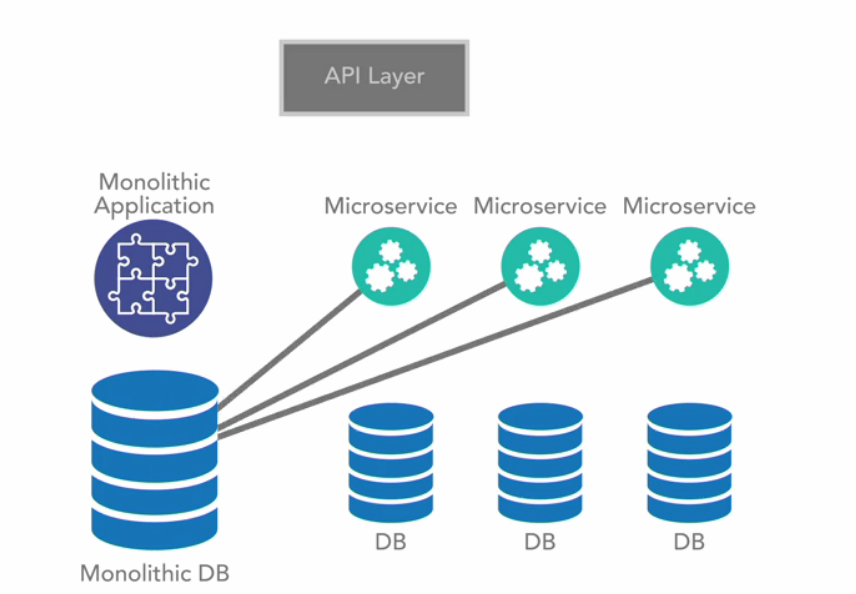
Based on the knowledge of the domain, contexts can be bounded

Example

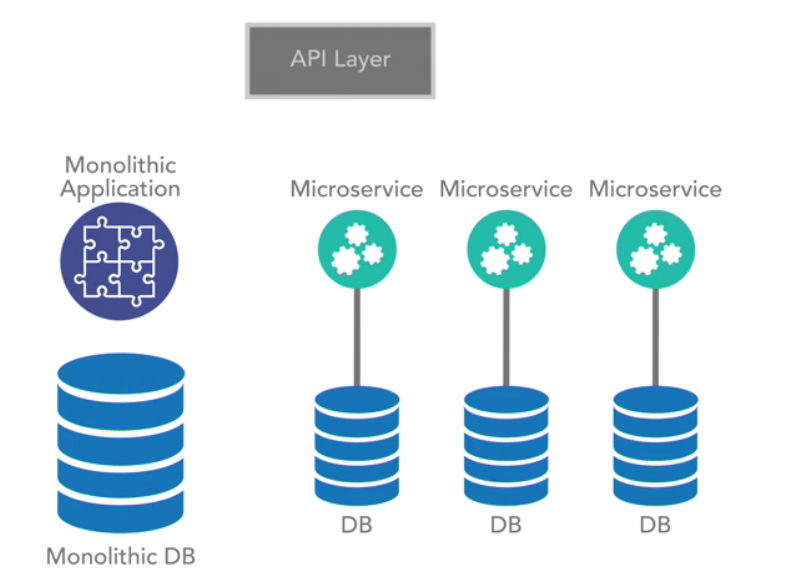
* User Domain
* Billing Domain
* Shopping Domain for Women etc.

If domains can be separated then stick with Monolith architecture

Bad Design



Good design



Each DB can be a separate instance

Micro Services uses BASE model – not ACID model

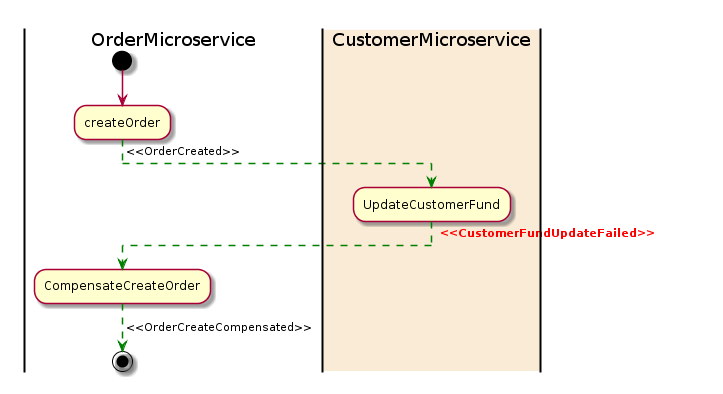
Basic availability, soft-state, and eventual consistency

Since one service can call another service and so on . It is difficult to main ACID properties. This kind of consistency can be maintained using Saga Pattern

Example:

Saga Pattern

In the event of failure, counter transaction will be executed and the previous transaction will be negated



**Design Considerations**

Asynchronous

Logging unification – Especially when there are many disparate teams working independently .

***Trace ID can be passed when send from one service to another Service. This can***

