Microservices: Brownfield: Transactions

acarlstein.com/

Posted by Alejandro G. Carlstein Ramos Mejia on January 16, 2017 January 17, 2017 About Programming / Architecture / Microservices

Microservices: Brownfield: Migration: Database | Microservices: Brownfield: Reporting

When moving from a monolithic system to a micro-service architectured system, we need a different approach when dealing with transactions.

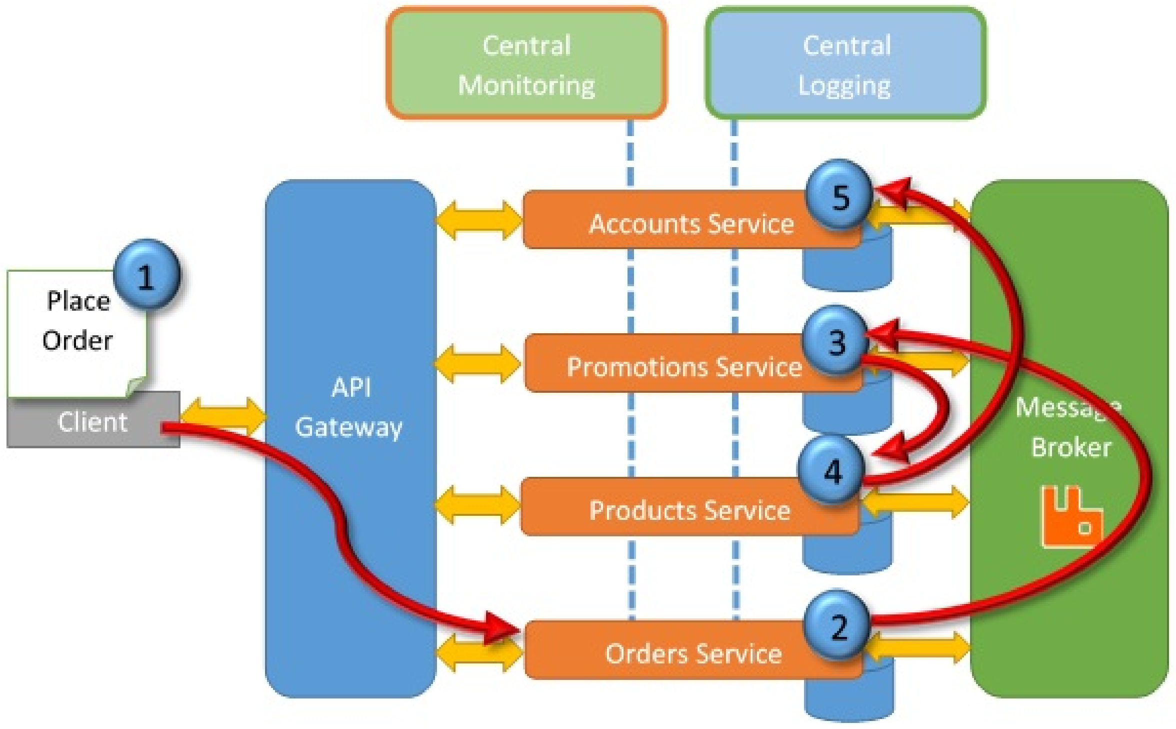
Transactions are useful:

- They ensure data integrity.
- They allow us to updates several records as part of one transaction.
- If one or more updates (and/or creates) fails, we can roll the entire transaction back.

In monolithic transactions are simple. We can have one process which is updating and creating records. These records are part of the transaction; therefore, the same process can either commit the transaction or roll it back if there are any issues.

In micro-services, transactions spanning are complex because there are several processes. This means that several micro-services are involved in complete one transaction. Since our transaction is distributed along multiple micro-services, it becomes a complex procedure to observe and solve problems; therefore, it becomes complex to roll back.

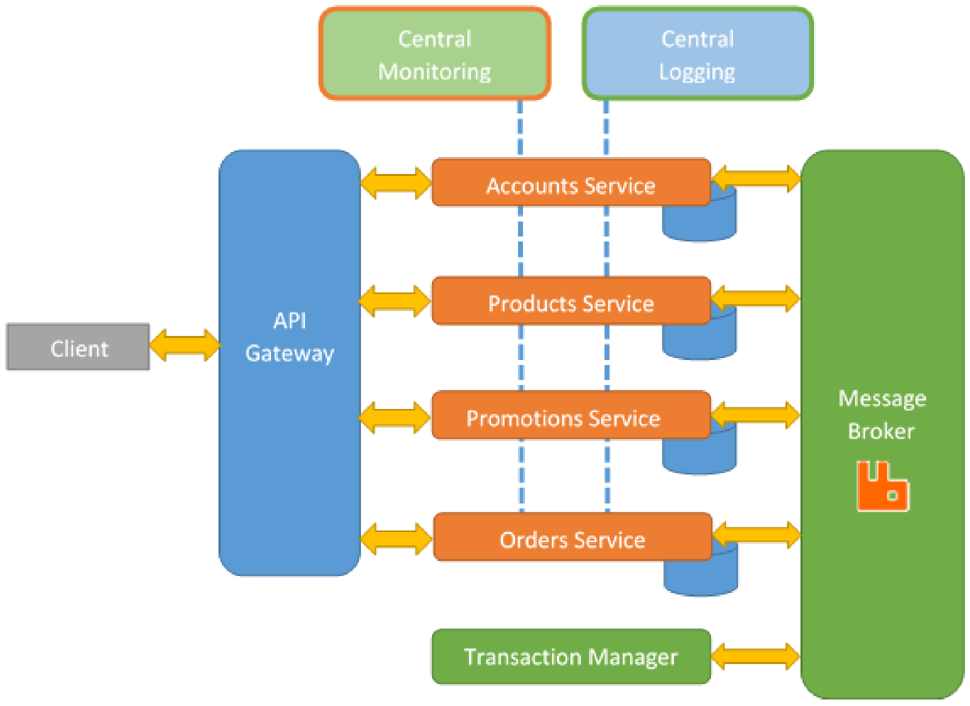
For example, we can have a order being place. This process will take several micro-services working together.



If one of these micro-services fails when trying to create or update a record, we will need to rollback the entire transaction.

How to handle fail transactions:

- Option 1: Try again later.
  The part of the transaction that failed is put into a queue so another service can pick it up and process.
  - Transaction will eventually be completed.
  - It relies on other instances not failing with the same part of transaction.
- Option 2: Abort the entire transaction.
  We detect our transaction has failed, then we issue an Undo transaction to all the micro-services involved so they undo any creates or updates
  - Problems:
    - Who issue the undo transaction?
    - What happens when the undo transaction fails itself.
  - One way to overcome this problem is to use a transaction manager software.



- This software use a two phase commit.
- Phase 1: All micro-services involved indicates to the transaction manager if they are fine to commit to their part of the transaction.
- Phase 2: If they are fine to commit, then the transaction manager tells all participating micro-services to commit the transaction.
- If any of the micro-services doesn't respond or responds with a "no to committing" then the transaction manager tells to all the participating micro-services to rollback the transaction.
- Problem using transaction manager?
    - We are heavily dependent of it.
    - It delays the processing of our transactions. Potential bottleneck.
    - Complex to implement.
    - More complex when we need to communicate with a monolithic system.

        This can be accomplish with the message queue.