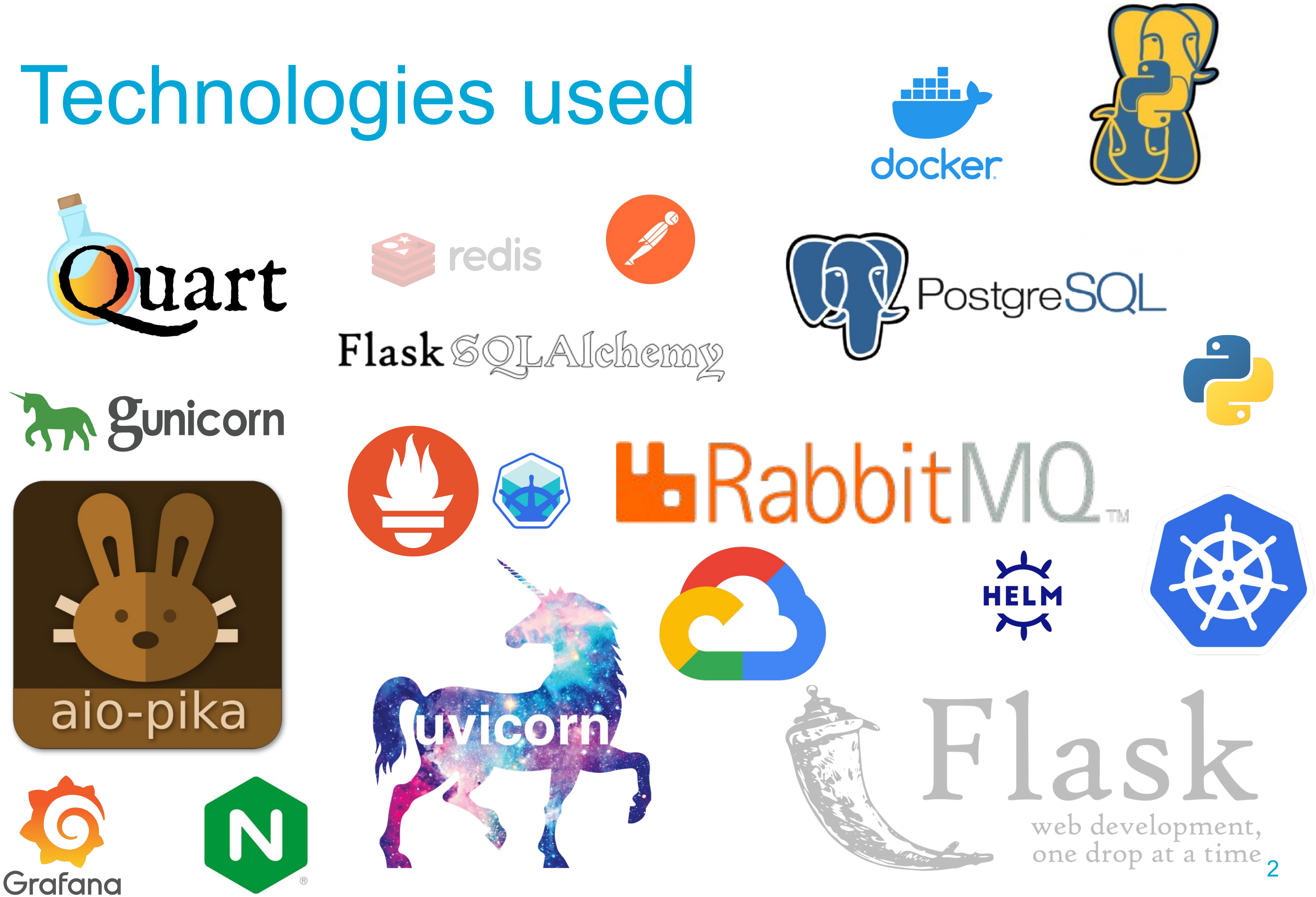
Web-scale Data Management

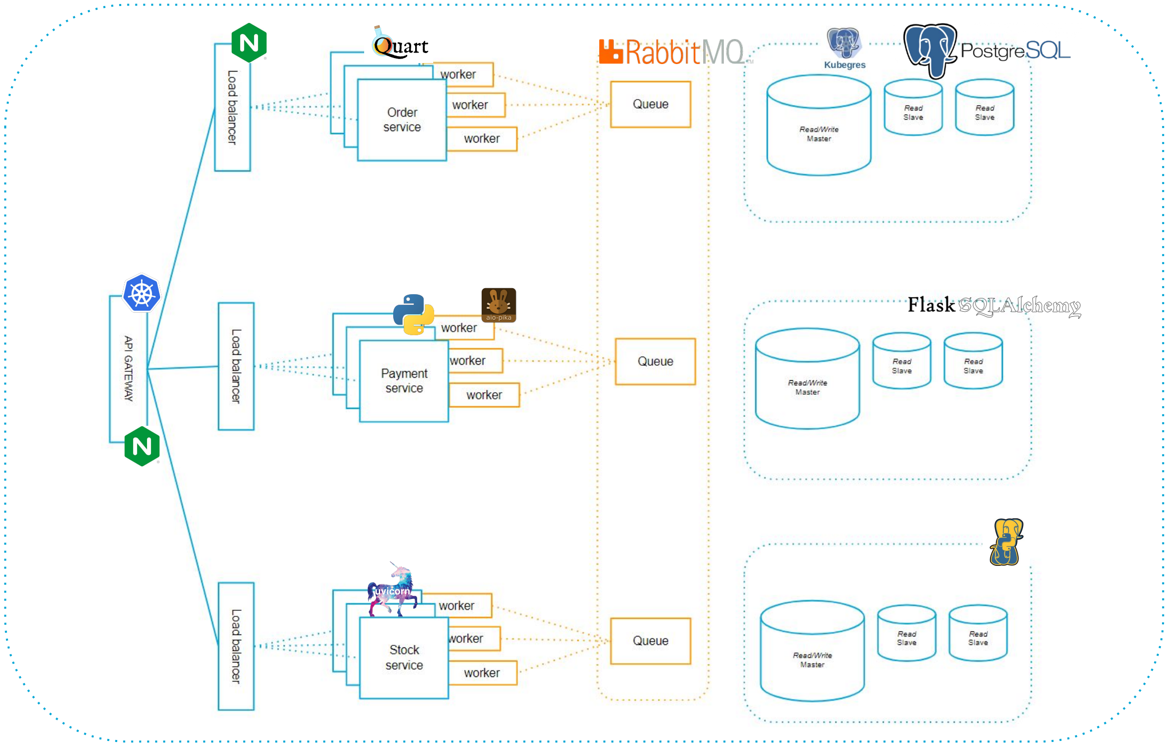
Group 2 - Pragmatic Project

TUDelft

Technologies used



Architecture Diagram

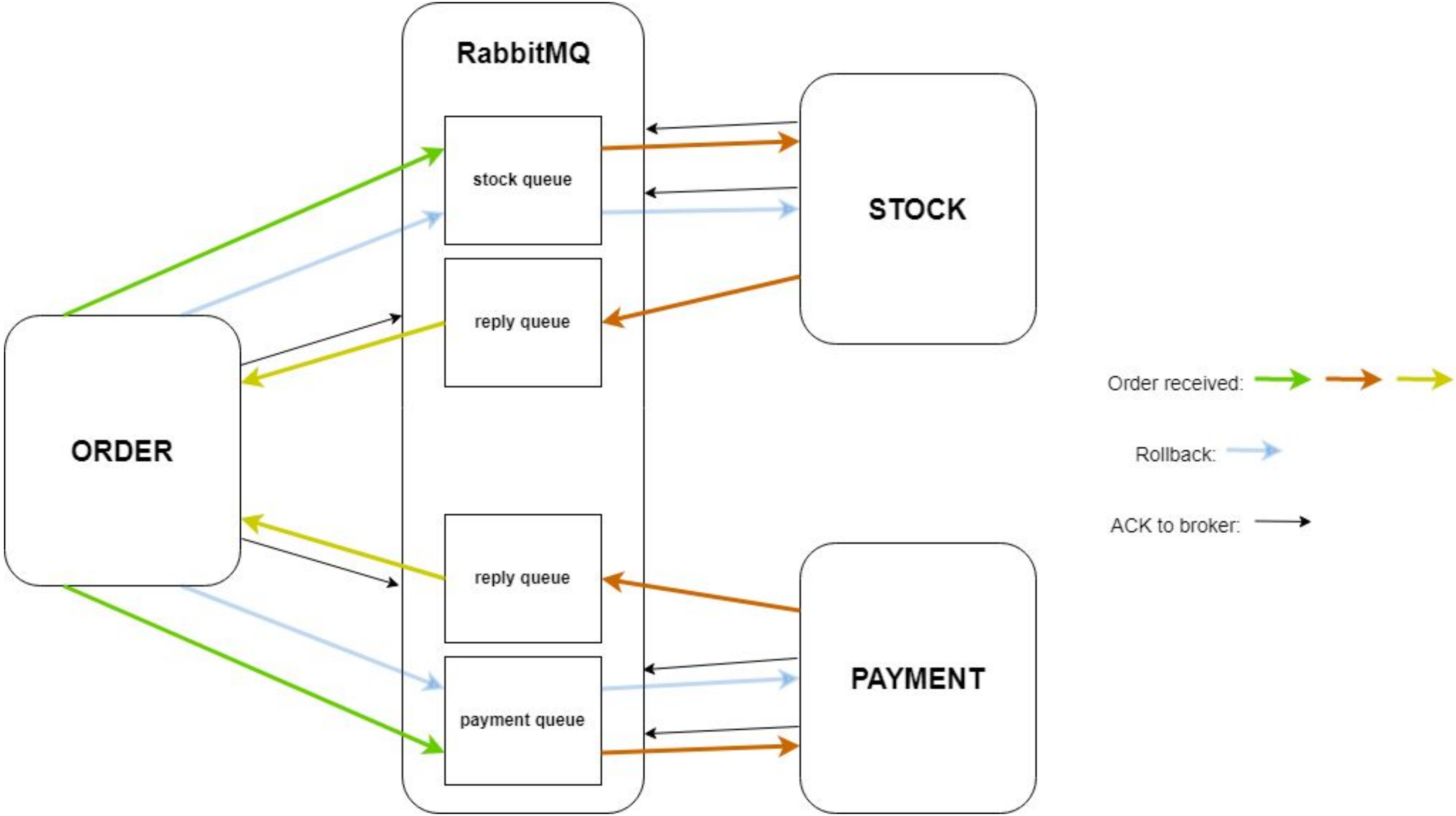


Deployment

docker
HELM

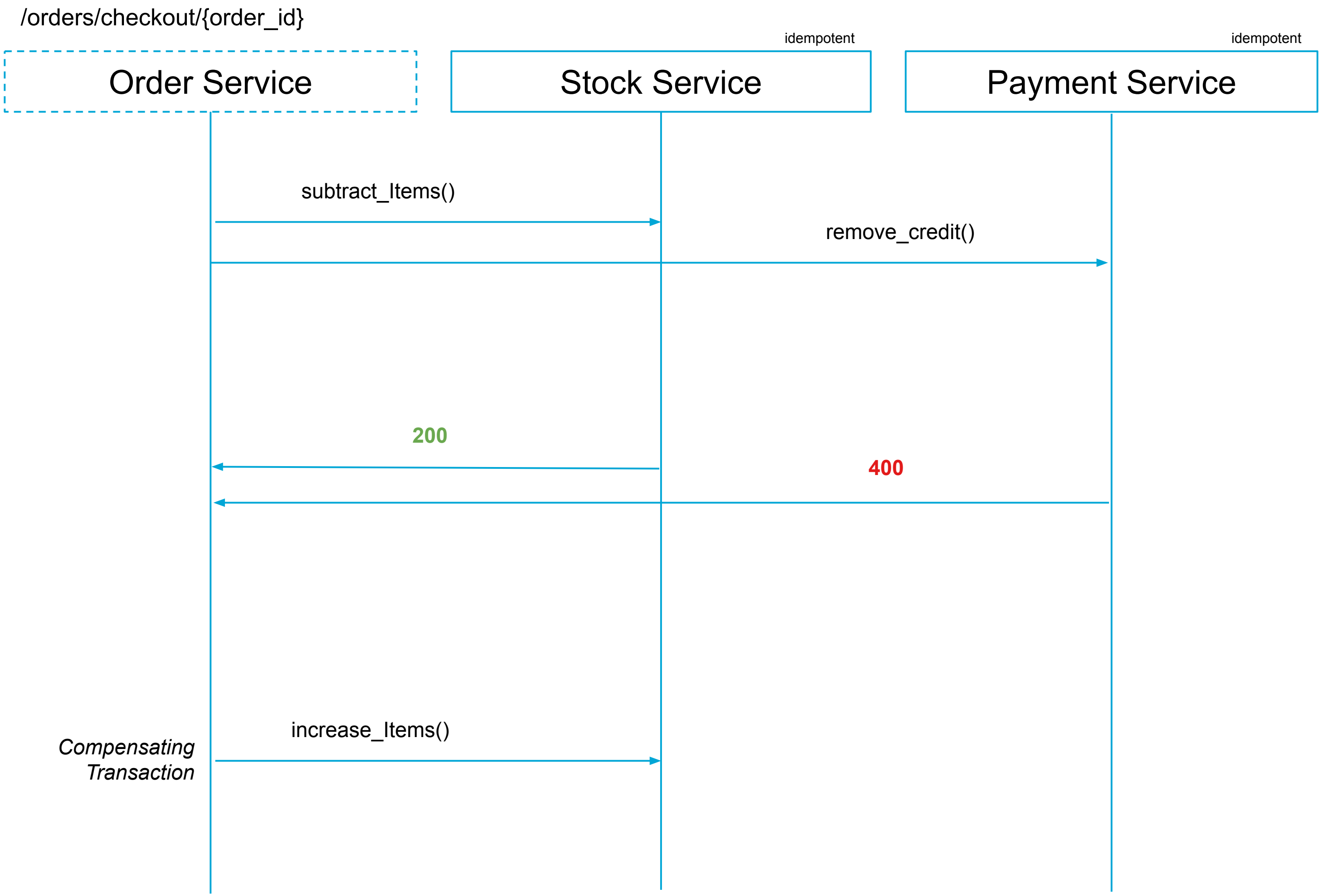
Monitoring & Observing

Messaging


RabbitMQ
stock queue
reply queue
reply queue
payment queue
ORDER
STOCK
PAYMENT
Order received:
Rollback:
ACK to broker:

Orchestration based SAGA

Method of transaction execution



TUDelft

SAGA Guarantees

ACID

Atomicity: ✅

Consistency: ✅

Isolation: ❌

Durability: ✅

BASE

Base **A**vailability: ✅

Soft state: ✅

Eventual Consistency: ✅

Source: https://www.infoq.com/articles/saga-orchestration-outbox/

Consistency

Eventual consistency

- Eventual consistency using SAGA [1], see SAGA guarantees
- BASE guarantees: Basic Availability, Soft State, Eventual Consistency [2]

PostgreSQL consistency

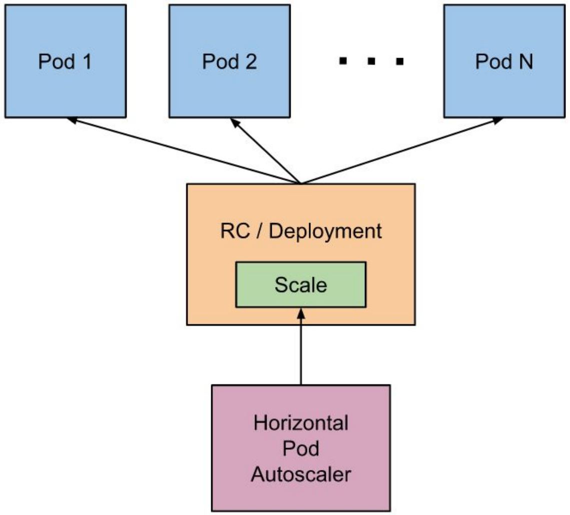
- Enforcing correct data using database constraints [3]
- “READ COMMITTED” transaction level [4]
  - Read sees snapshot of database, dirty read not possible
  - Write waits for concurrent transactions on target rows

Correctness of transactions

- Correctness through code checks
- Distributed Transactions possible through acknowledgements using RabbitMQ

Scalability

- Kubernetes Horizontal Pod Autoscaling
- Kubegres for PostgreSQL clusters + replicas
- Load balancing using Ingress
- Asynchronous messaging
- **No sharding and multiple masters**



Fault Tolerance

Database Fault Tolerance

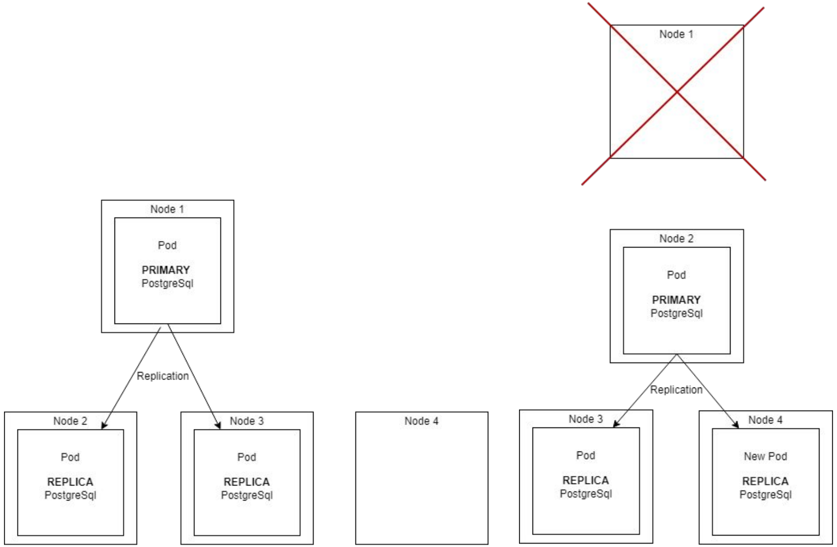
- Pod-anti affinity [5]
- Failover [5]

Microservice Fault Tolerance

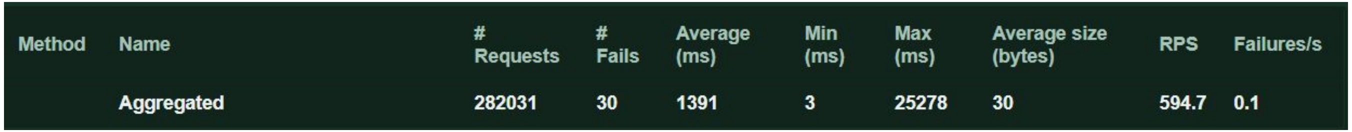
- Stateless and replicated
- Self-healing [6]

Communication Fault Tolerance

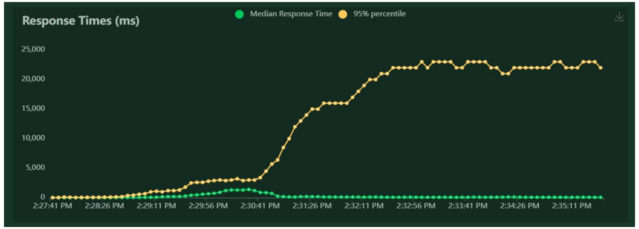
- Acknowledgements [7]
- Quorum Queues [8]
- RabbitMQ Cluster [9]

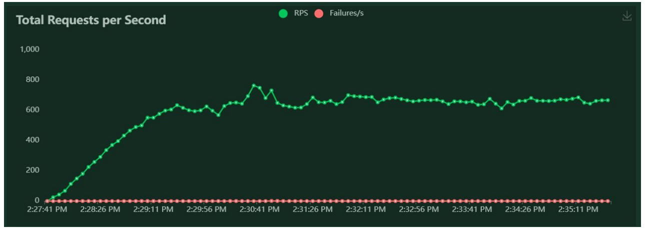


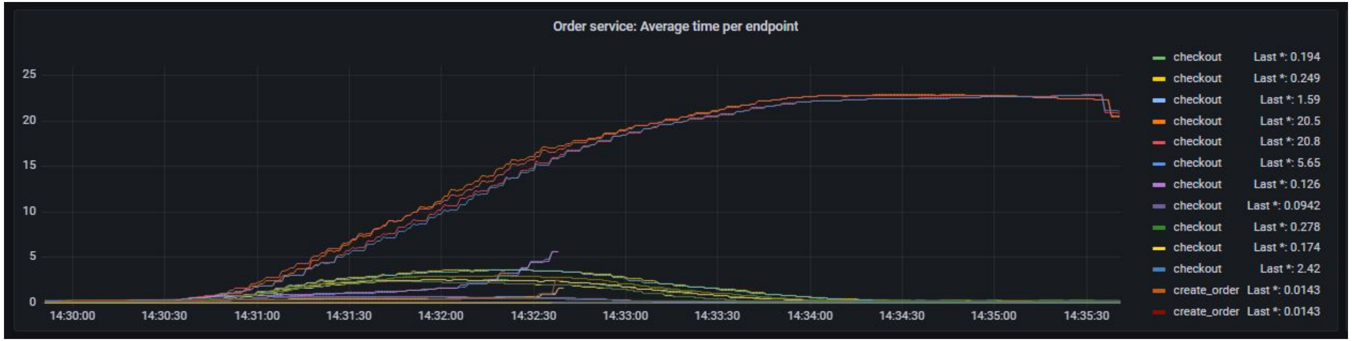
Results



Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
	Aggregated	282031	30	1391	3	25278	30	594.7	0.1







Limitations

- *Kubegres* does not allow sharding or multi-master replication. An example which does, is *BDR* [10].
- There is probably a bug somewhere, causing slow response times.

References

[1] https://medium.com/trendyol-tech/saga-pattern-briefly-5b6cf22dfabc

[2] https://www.scylladb.com/glossary/database-consistency/

[3] https://stackoverflow.com/questions/14225998/flask-sqlalchemy-column-constraint-for-positive-integer

[4] https://www.postgresql.org/docs/current/transaction-iso.html#XACT-READ-COMMITTED

[5] https://www.kubegres.io/doc/replication-and-failover.html

[6] https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

[7] https://www.rabbitmq.com/confirms.html

[8] https://www.rabbitmq.com/quorum-queues.html#usage

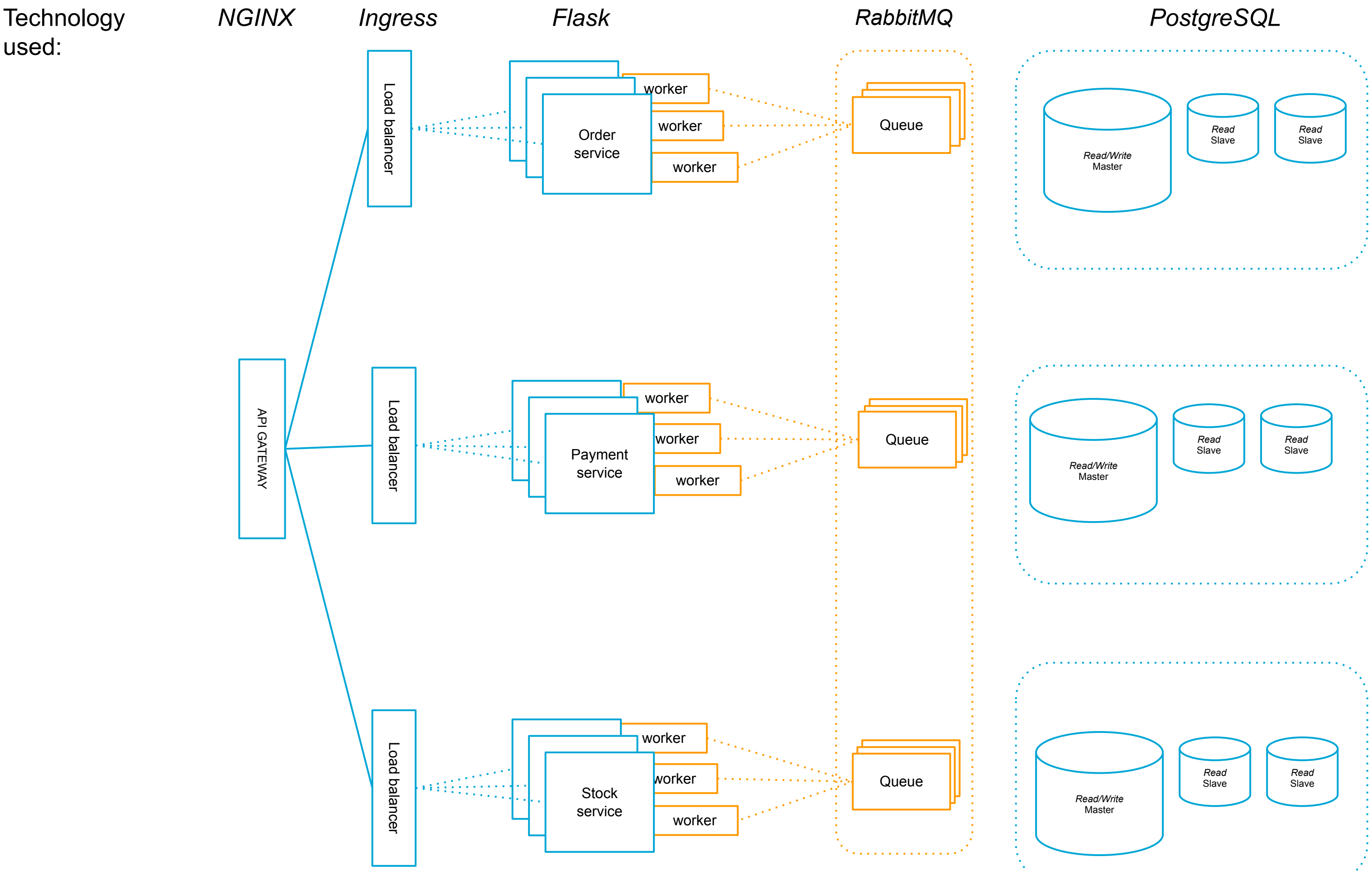
[9] https://www.rabbitmq.com/kubernetes/operator/operator-overview.html

[10] https://www.linkedin.com/pulse/multi-master-replication-relational-databases-scaling-ran-bechor/

Contents

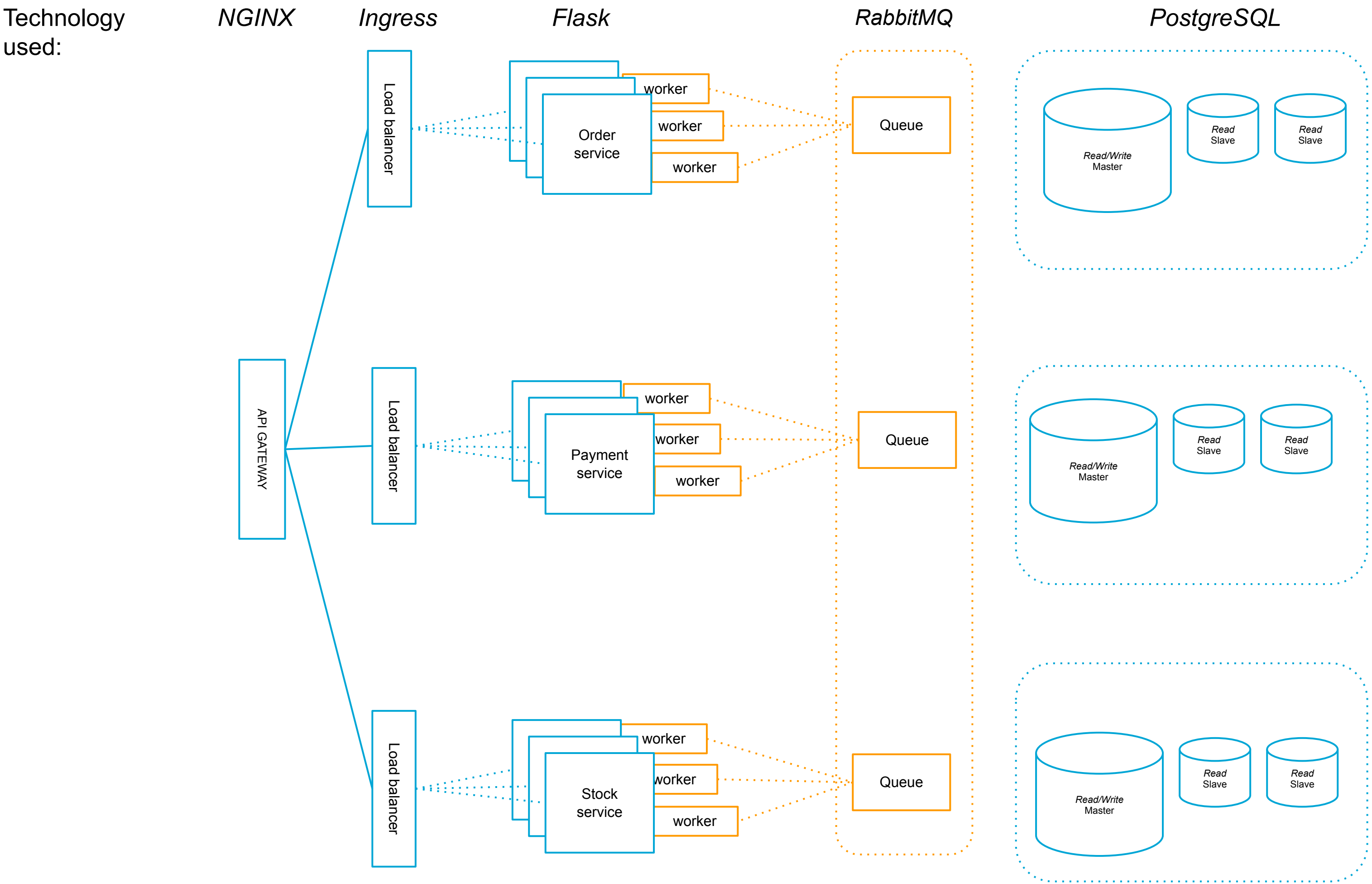
TU Delft

Overall Design Architecture Diagram



TUDelft

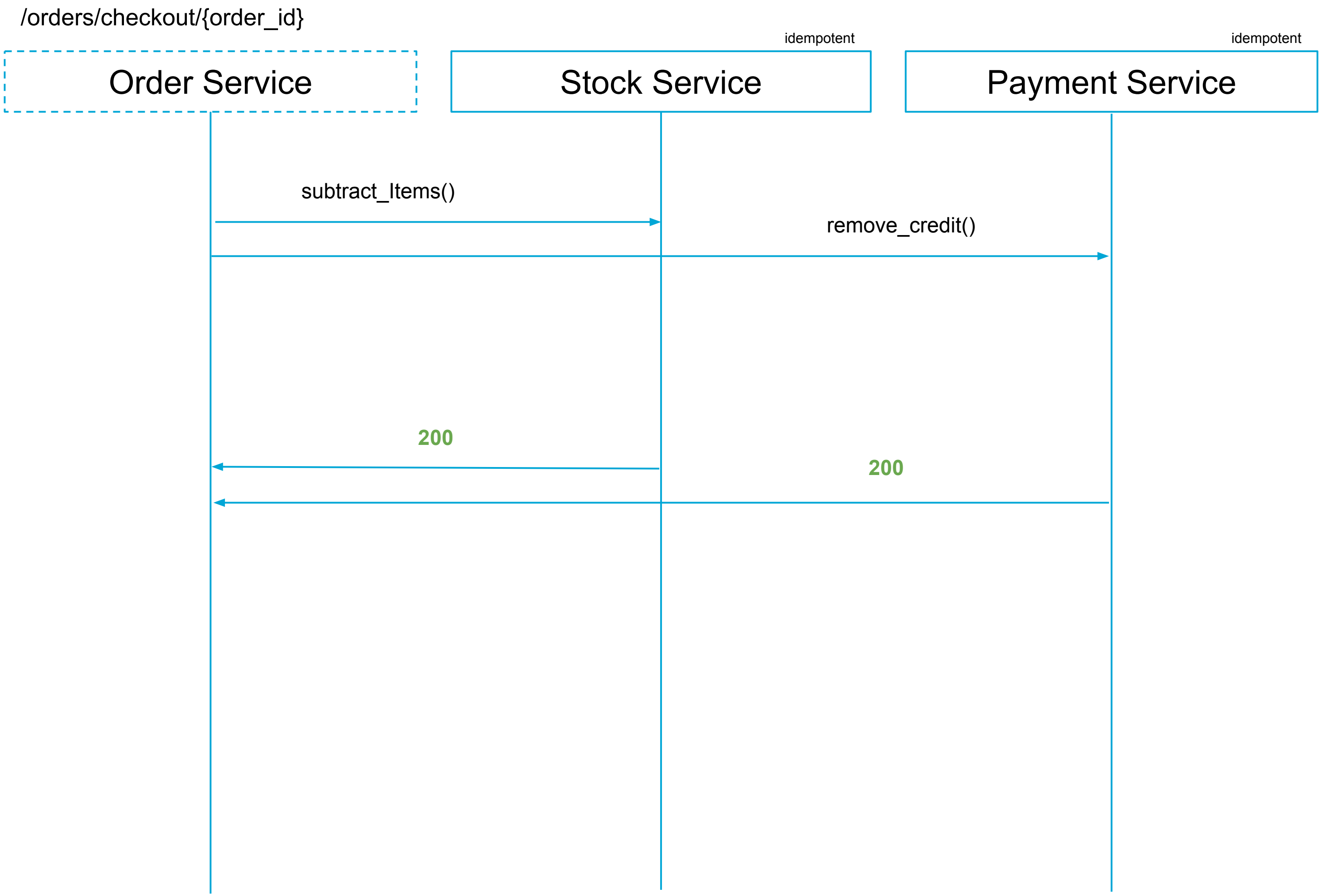
Overall Design Architecture Diagram



Transactions

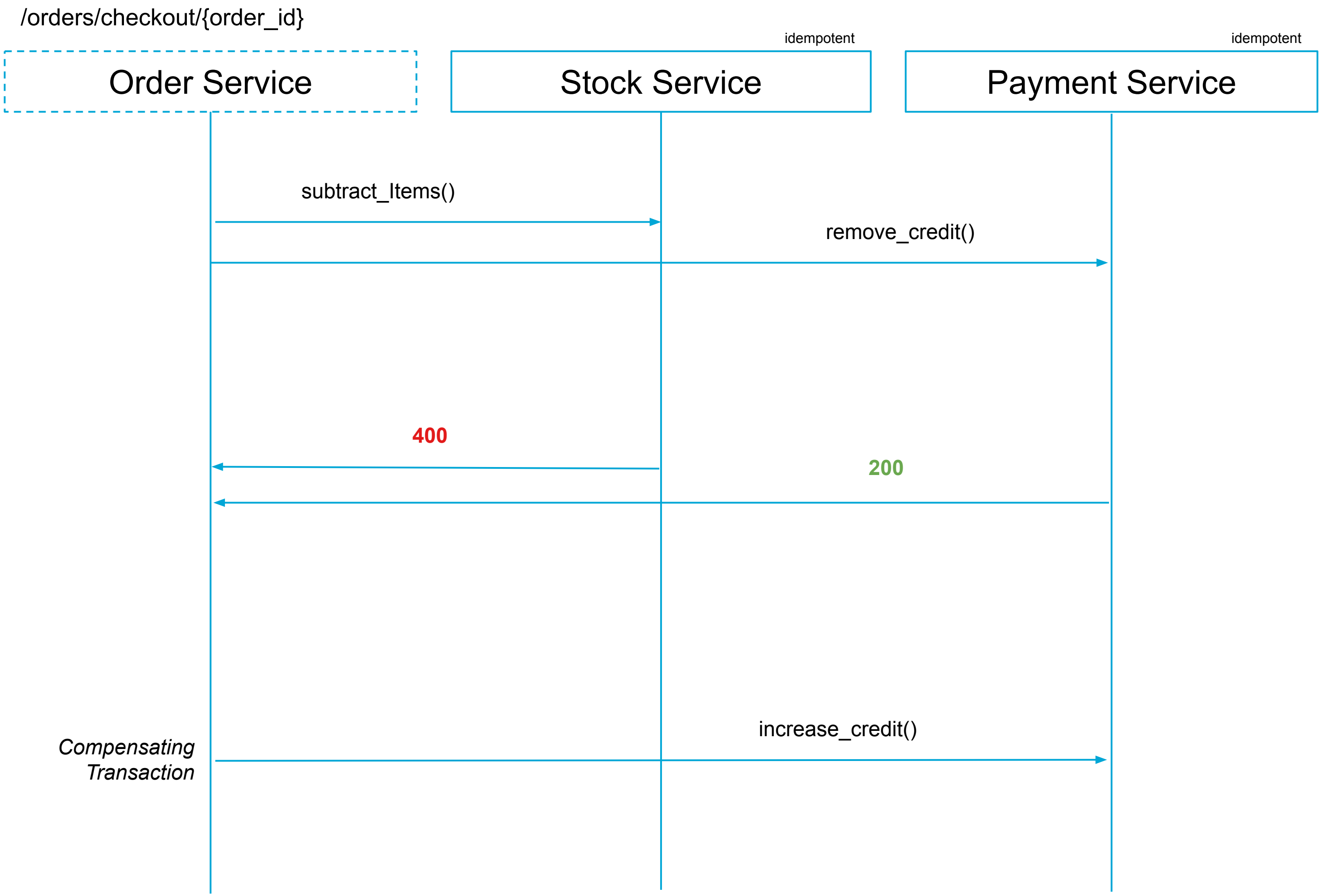
SAGA Pattern - Orchestration

Ideal scenario



Orchestration based SAGA

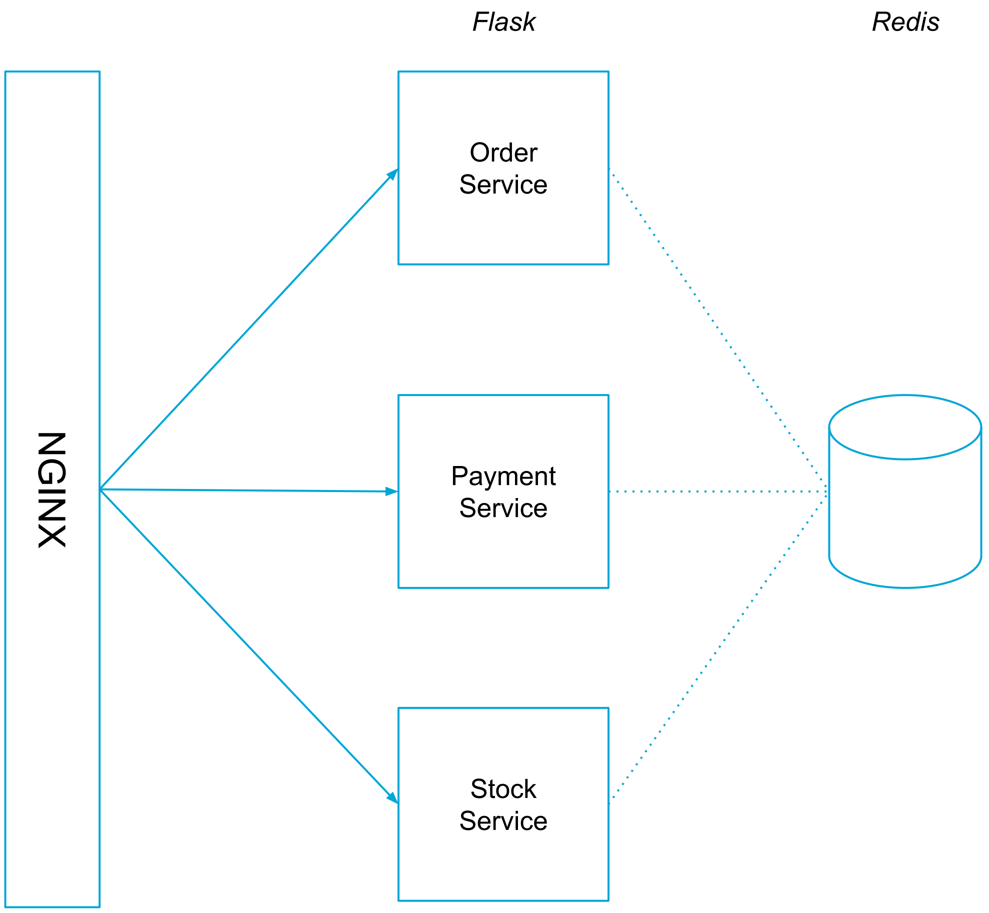
Method of transaction execution



Architectures

Technologies used

Current architecture

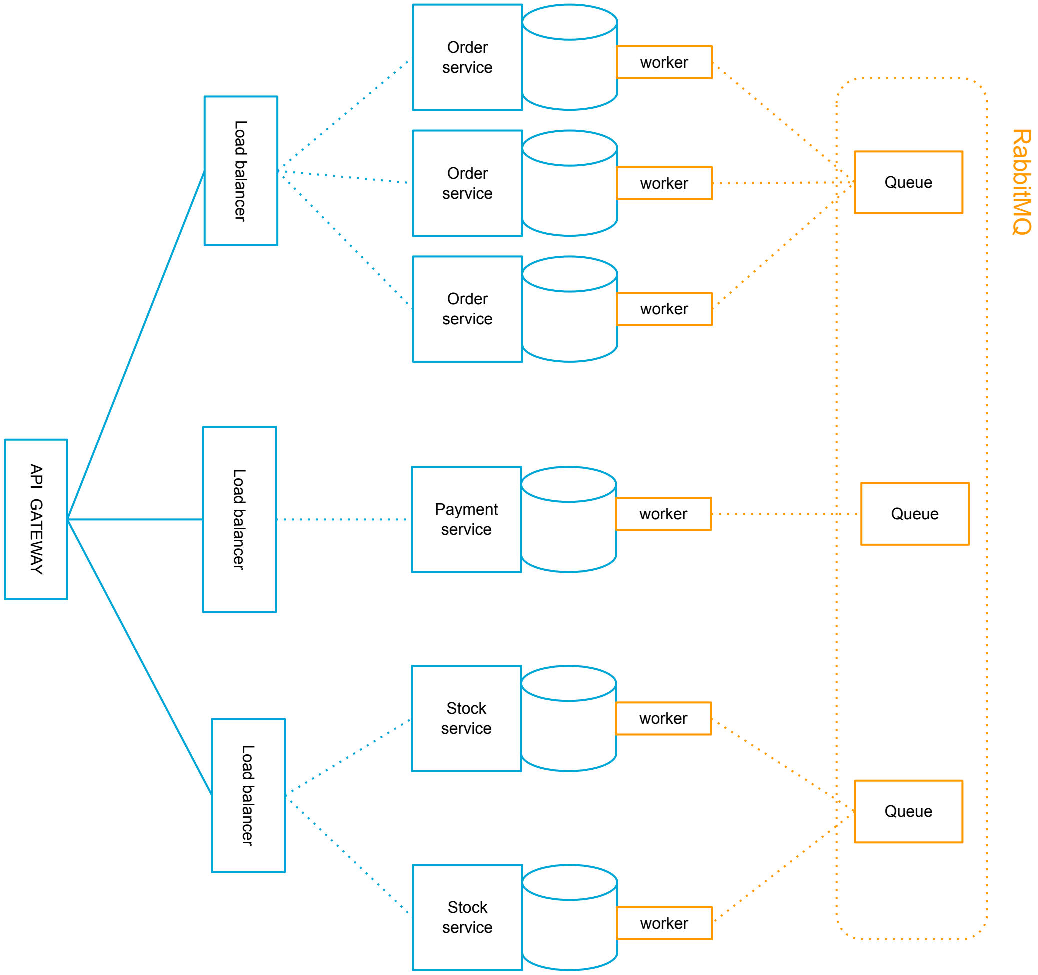


Prescriptive Architecture

Including RabbitMQ

Database per service

RabbitMQ
Kubernetes



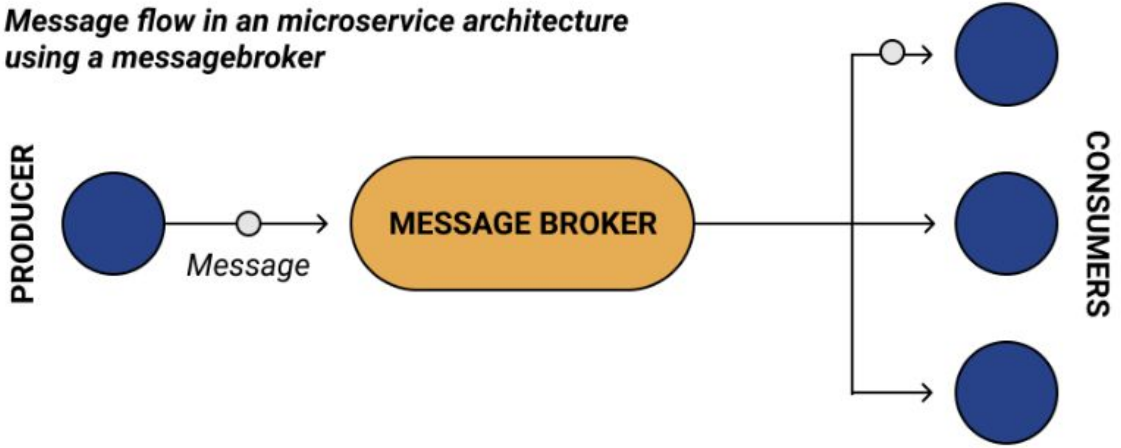
Patterns Chosen

Microservices architecture pattern → Database Per Service Pattern (Databases must sometimes be replicated and sharded in order to scale) → SAGA pattern

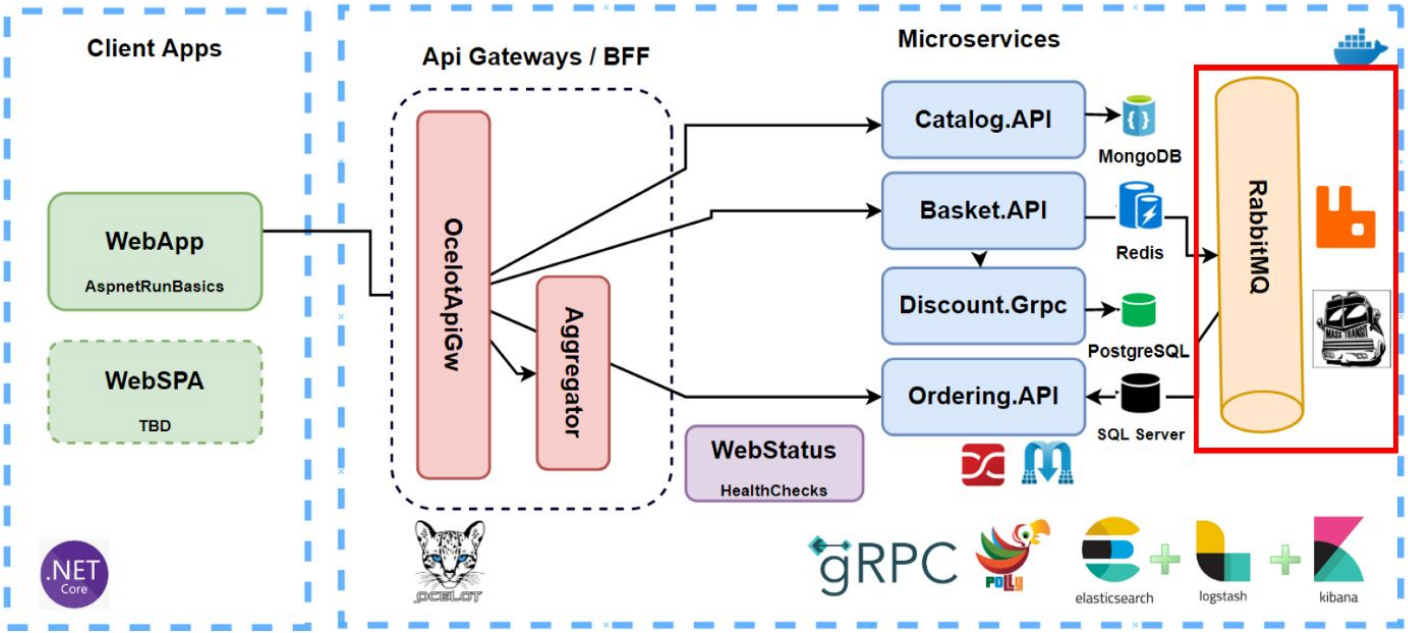
Messaging

RabbitMQ

- Implements message queueing, to provide message storage when a service is busy or disconnected


Message flow in an microservice architecture using a messagebroker

Microservices Event Driven Architecture with RabbitMQ



Source: Medium

Transactions

Transactions

SAGA
vs
2PC

Why we chose to work with SAGA

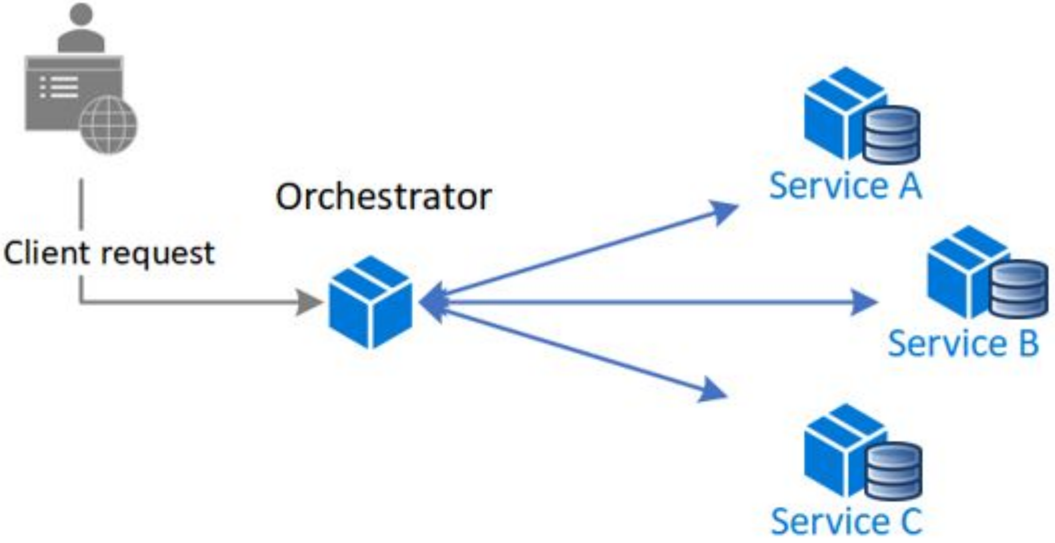
2 Phase Commit		SAGA	
Pros	Cons	Pros	Cons
Strong data consistency	Very complex process to maintain	No deadlocks	Eventual data consistency
Support ACID features	High latency & low throughput (blocking process)	No single point of failure	Complex design
	Possible deadlocks	Non-blocking operations	
	Transaction coordinator is a single point of failure		

[Source: Medium]

Transactions

SAGA

Orchestration pattern



Source: Microsoft

Example

Consistency

Concurrent requests