Enterprise Application Development

Emre Savcı
Senior Software Engineer @trendyol

Who Am I

Social

Go Türkiye

github.com/mstrYoda

twitter.com/mstrYoda_

medium.com/@mstrYoda

Interests

Microservices, scalability, kubernetes, istio, cncf, golang, software architecture

Contributions

RestSharp, OpenFeign, kustomize, Istio, Dapr, ksniff, gocb

Agenda

Monolith Applications

- Scaling monolith applications
- Limitations of monolith applications
- Microservices

Microservice Applications

- Scaling microservices
- Advantages of microservices
- Communication between microservices
- Message brokers
- Event driven communication
- Service discovery

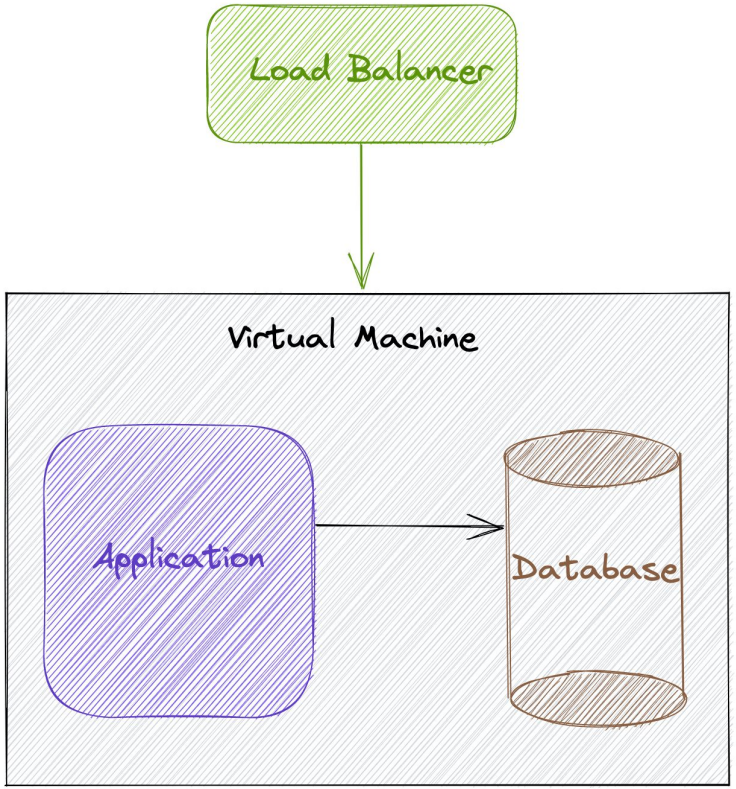
Performance & Platform

- CQRS
- Caching
- Kubernetes

Monolith Applications

- Applications are run as a single process

- Applications are scaled up/down as a whole



Monolith Applications

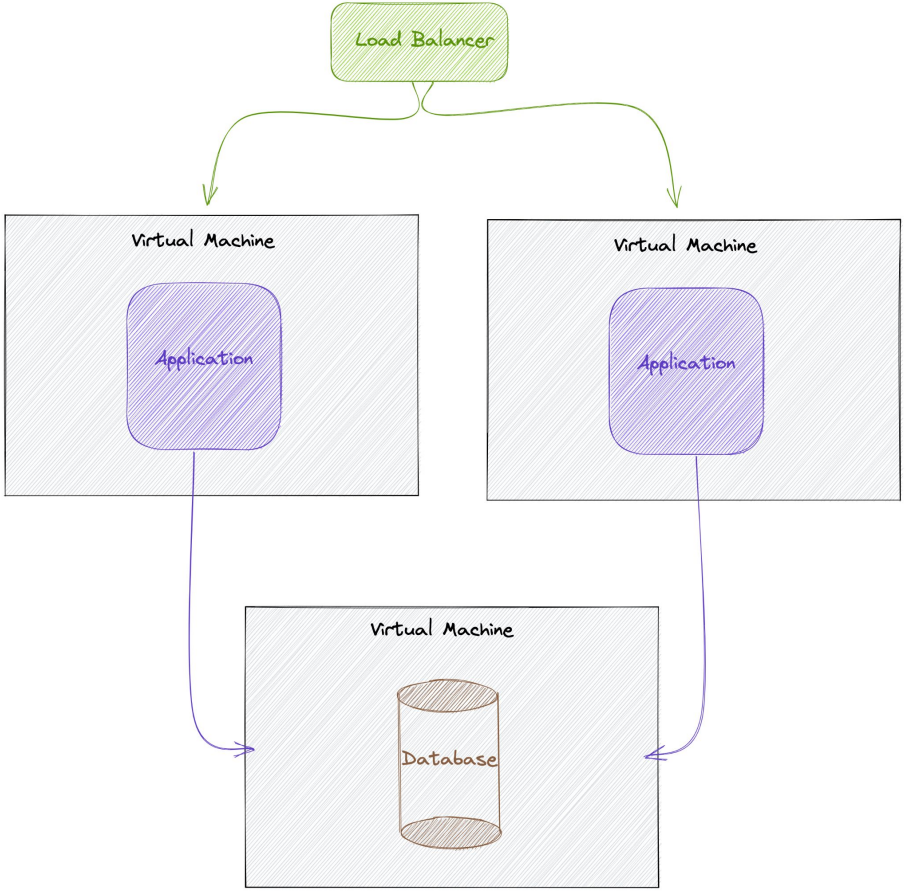
Scaling Monolith Applications

Vertical Scaling

- Add more resource (limited)

Horizontal Scaling

- Add more application/machine



Monolith Applications

Limitations for Monoliths?

Monolith Applications

Limitations for Monoliths

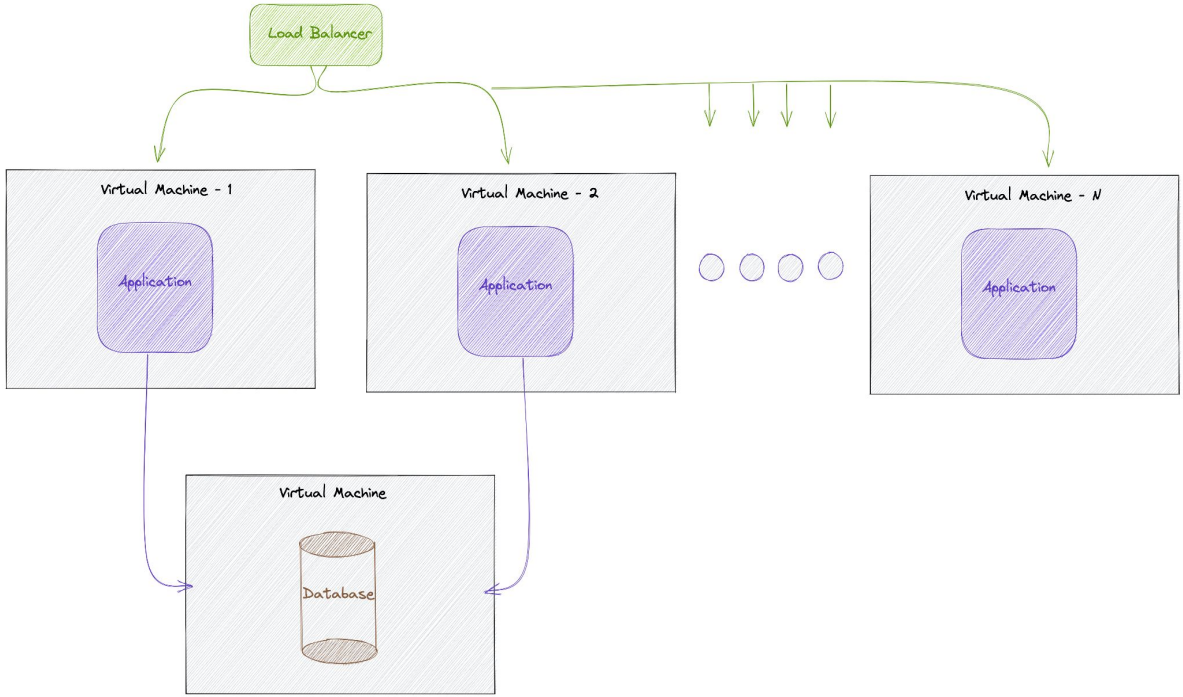
- Scaling bottlenecks separately is not possible

- Tightly coupling

- Resource consumption might grow up unnecessarily

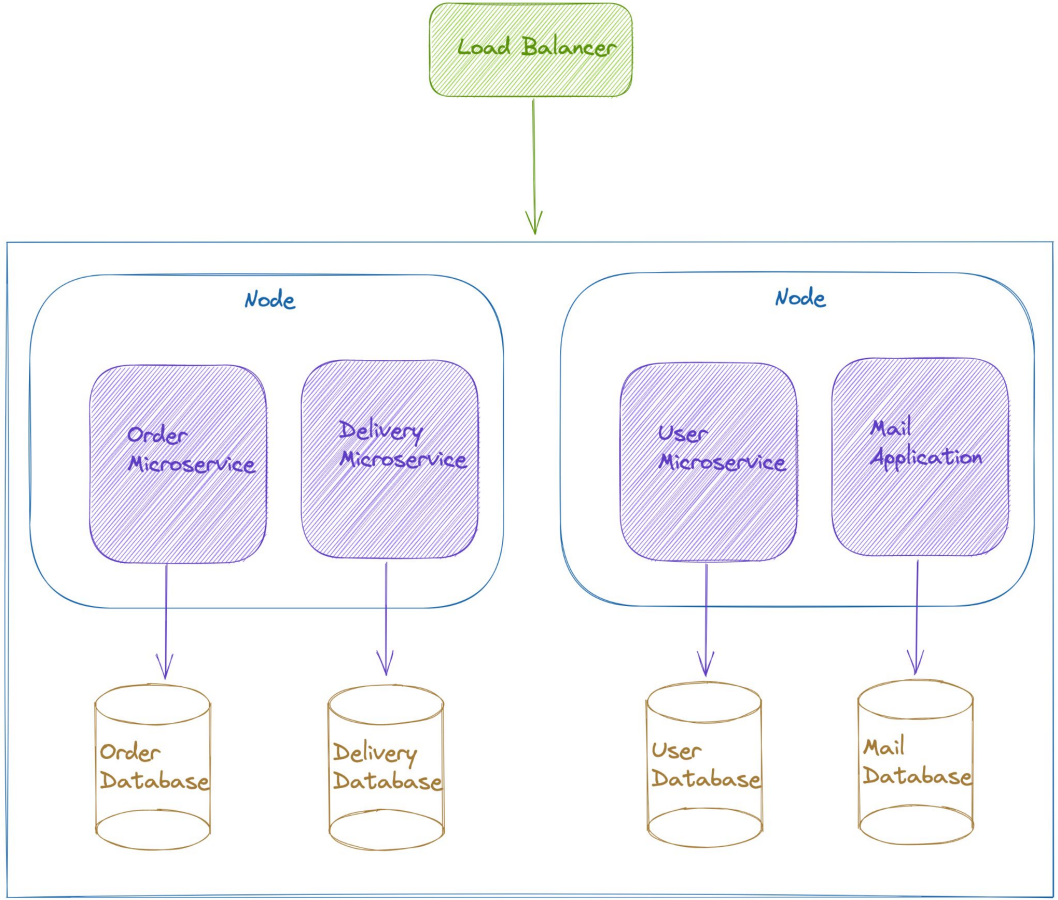
- Teams might conflict while making concurrent development

- Hard deployment process



Microservice Applications

- Manages its own data
- Independent scale option for any component
- Dedicated resource for different applications
- Concurrent development
- Fast deployment time
- Fault isolation



Microservice Applications

Communication

Depends on your requirements

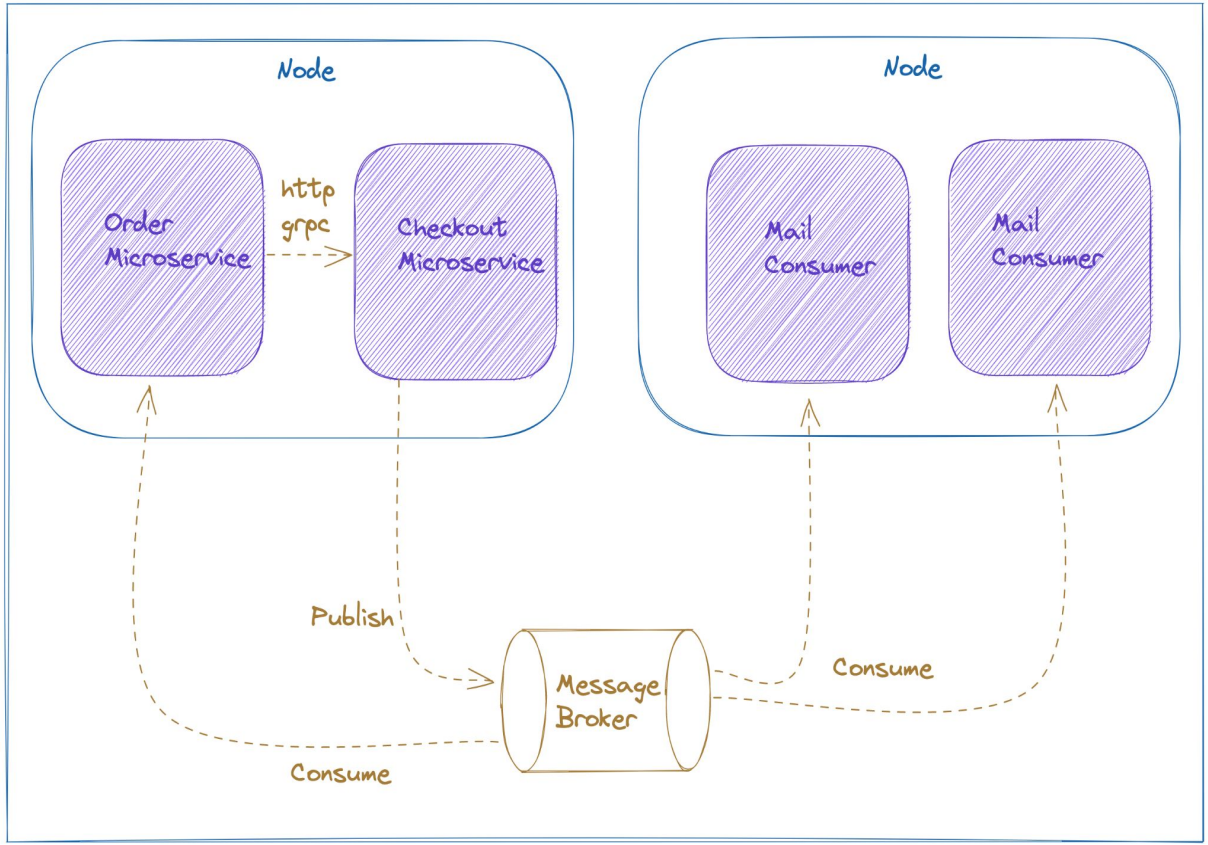
Sync. Communication

- Http

- Grpc

Async. Communication

- Messaging (Message brokers: RabbitMQ, ActiveMQ, MSMQ, Kafka, SQS, Nats etc..)



Microservice Applications

Disadvantages?

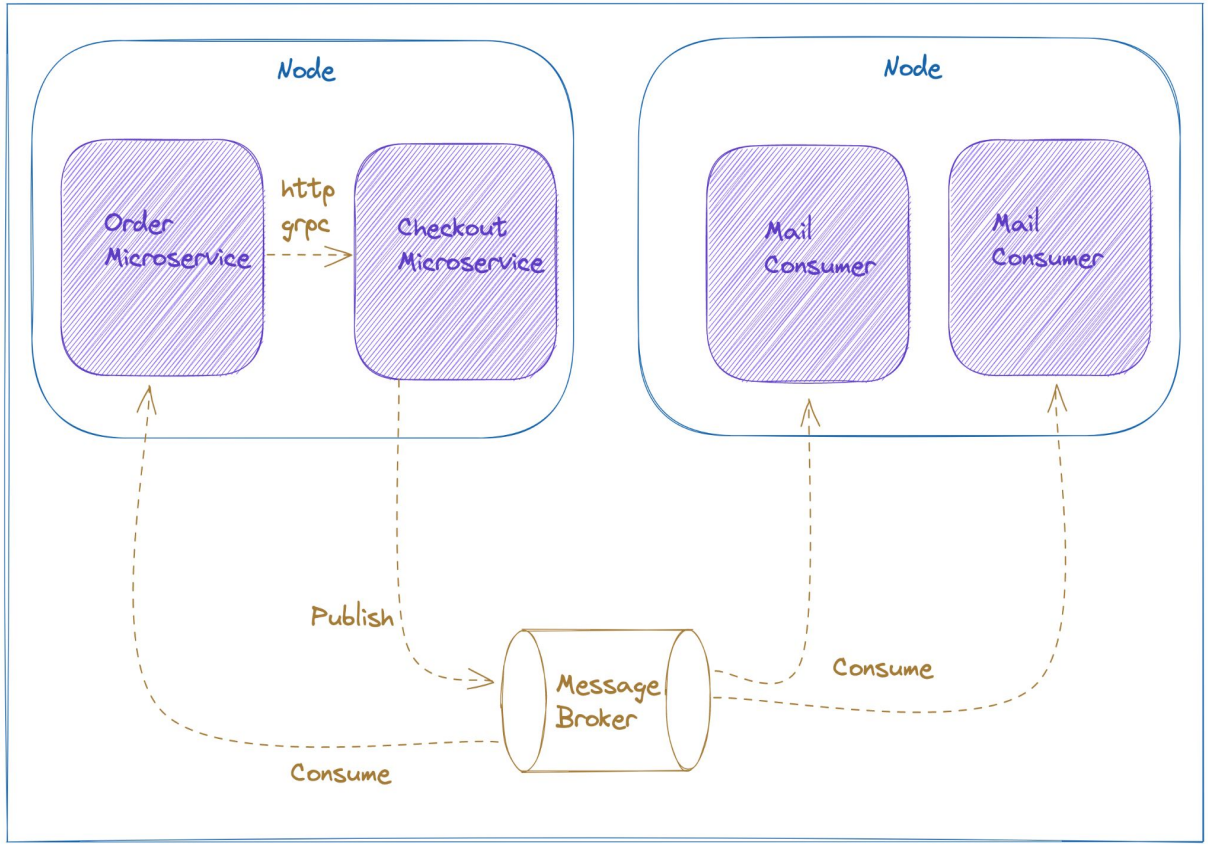
Microservice Applications

Disadvantages?

- Hard to test

- More complex structure

- Overall domain knowledge



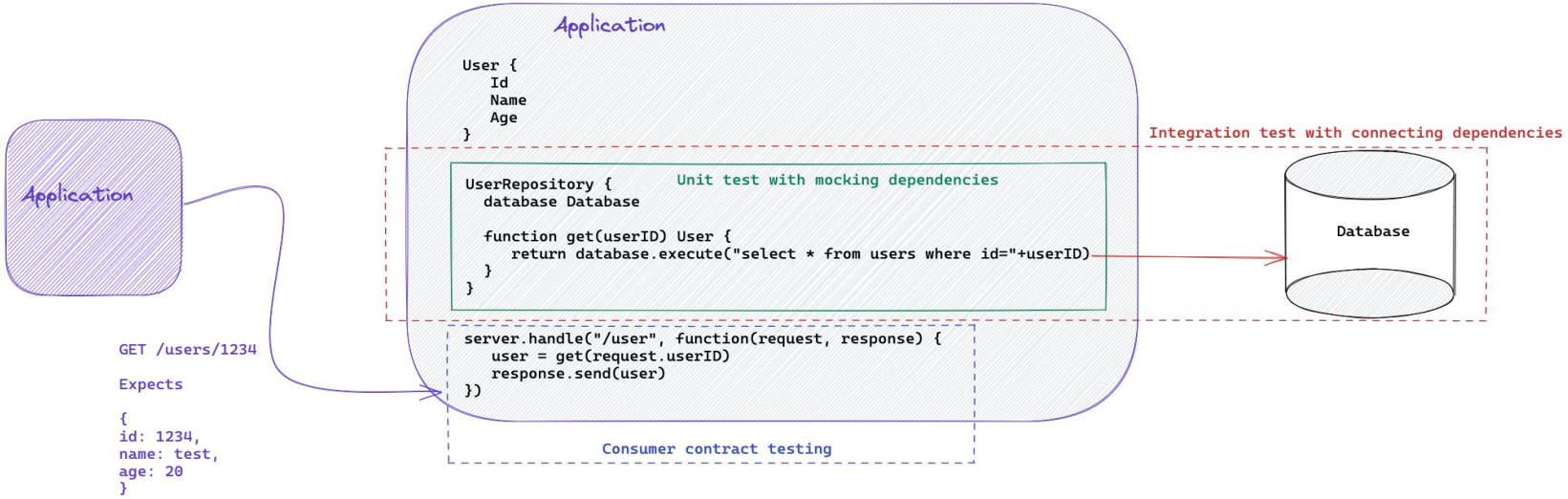
Microservice Applications

How to Test?

- Unit Testing

- Integration Testing

- Consumer Driven Contract Testing



Scaling

- How to improve data access?

- How to distribute load across different applications?

- How to eliminate read latency?

- How to automatically scale applications?

Scaling

- CQRS (Command Query Responsibility Segregation)

- Read scale

- Write scale

- CAP - Eventually consistency

- Caching

Scaling

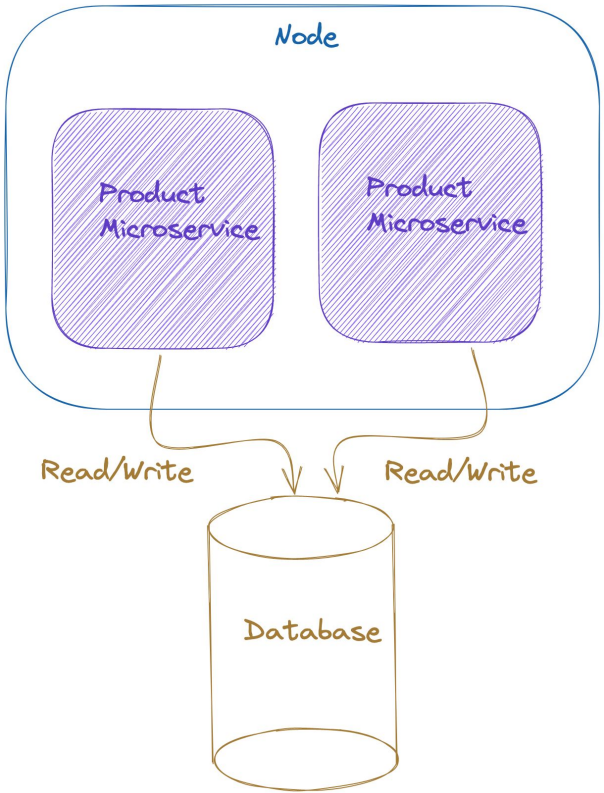
CQRS & Read/Write Scale

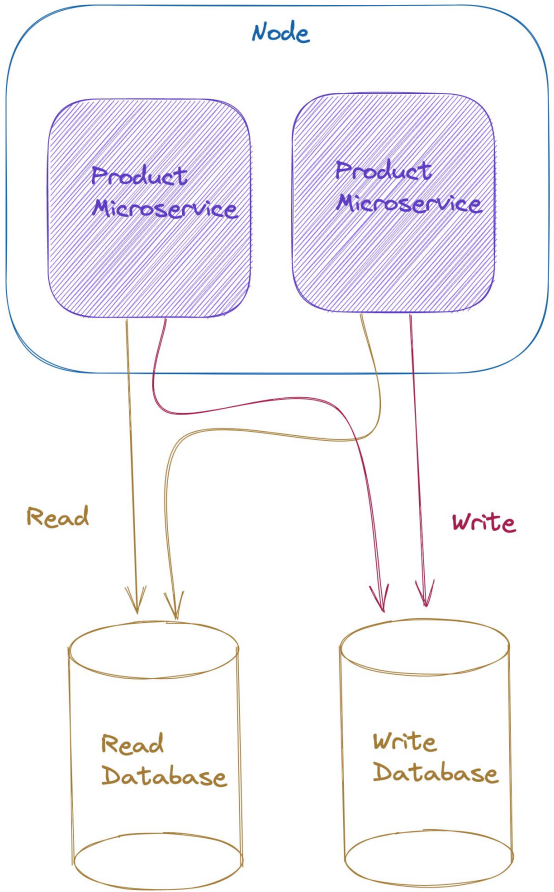
- Separating read and write

- It can be applied to both source code and infrastructure level

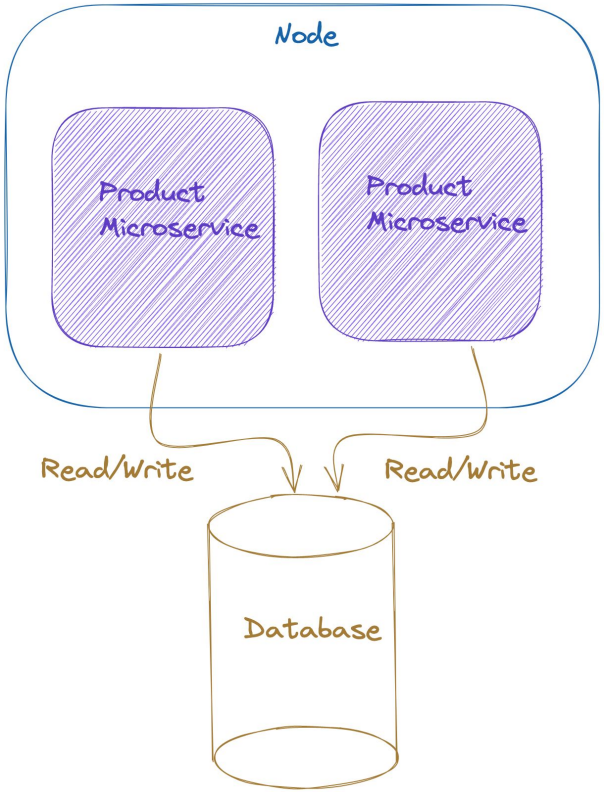
- We can create different applications for read/write operations

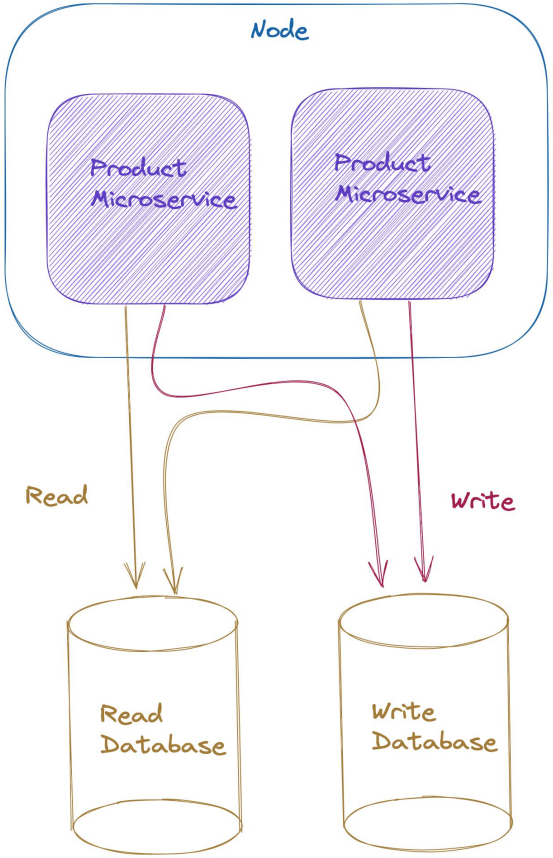
- We can use different databases for reading and writing resources

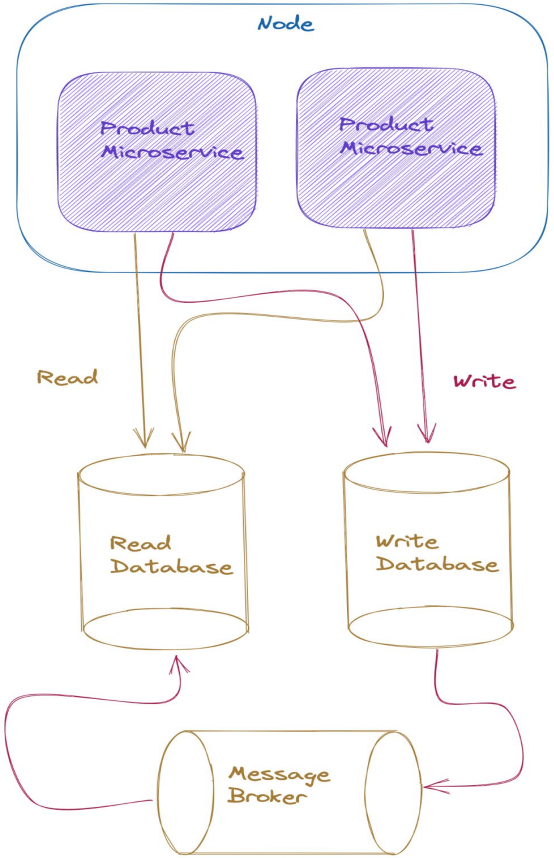




Scaling

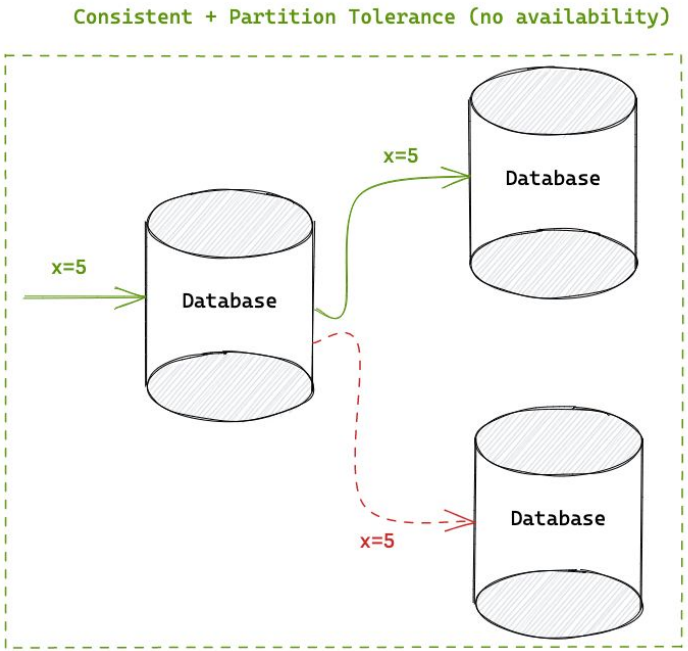

Node
Product Microservice
Product Microservice
Read/Write
Read/Write
Database

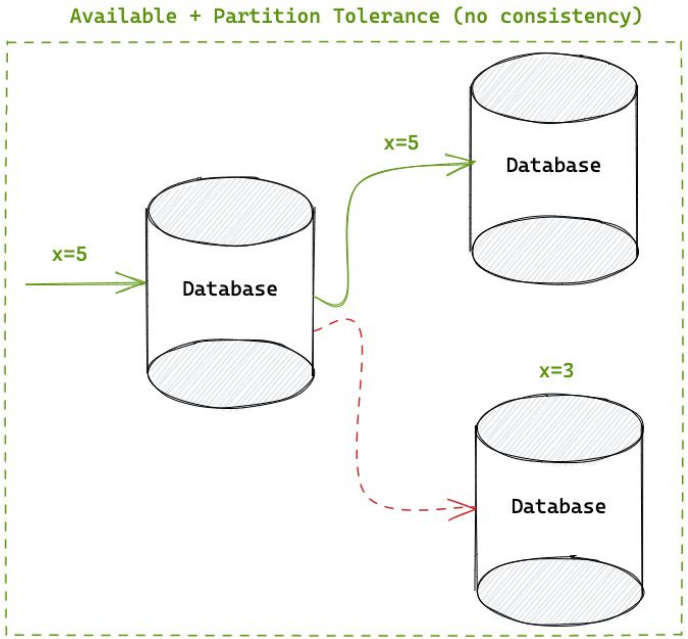

Node
Product Microservice
Product Microservice
Read
Write
Read Database
Write Database


Node
Product Microservice
Product Microservice
Read
Write
Read Database
Write Database
Message Broker

Scaling

CAP Theorem (Consistency, Availability, Partition Tolerance)



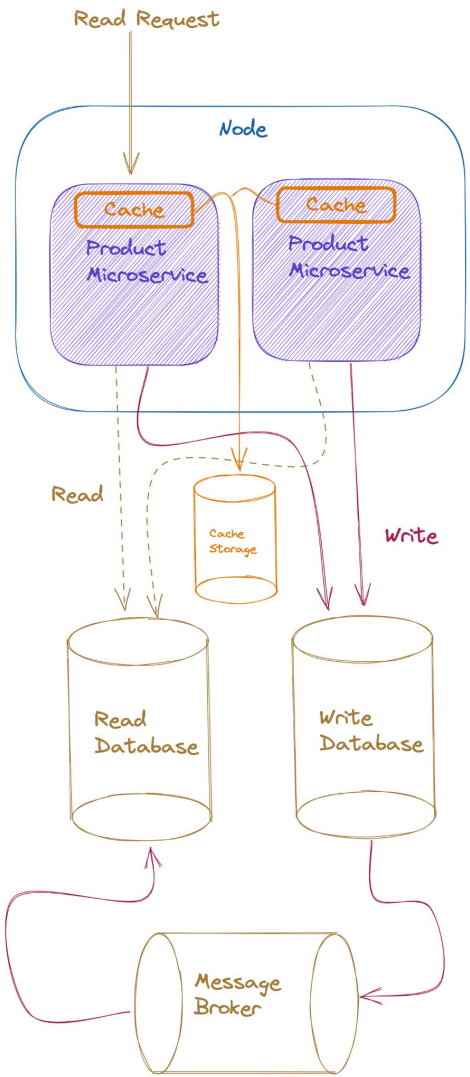


Scaling

Caching

- Faster reads

- Separate cache storage



Scaling

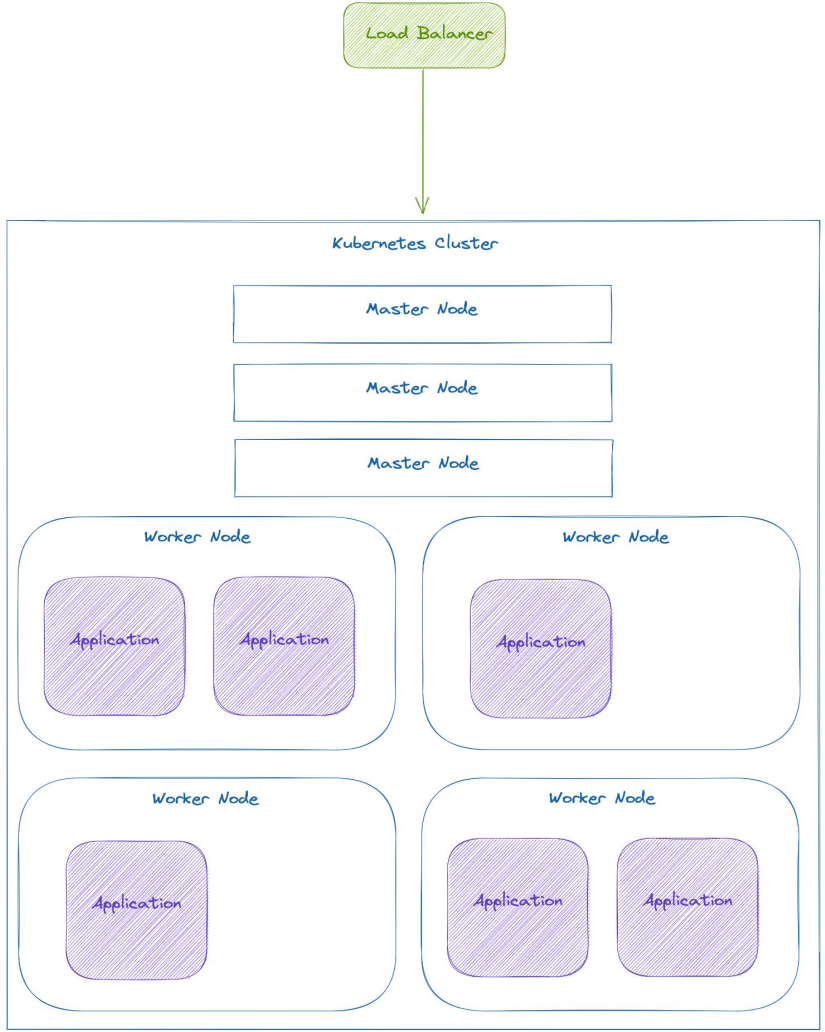
Kubernetes

- Automatically scaling

- Service discovery

- Health checking

- Rolling update / rollback



Q/A

keywords: **#**monolith **#**microservice **#**CI/CD **#**cache **#**horizontal **#**vertical **#**scaling **#**service-discovery **#**cap **#**message-broker **#**testing **#**CQRS