Microservice architecture

SWA, FEL CVUT

Ing. Marek Schmidt
Quality Engineer, Red Hat Middleware

2017-05-11

MICROSERVICES

Introduction

Concepts

Small & autonomous, loose coupling & tight cohesion, Conway's law, resilience & scaling

Testing

Consumer-driven contracts, Canary releases, Semantic monitoring

Monitoring

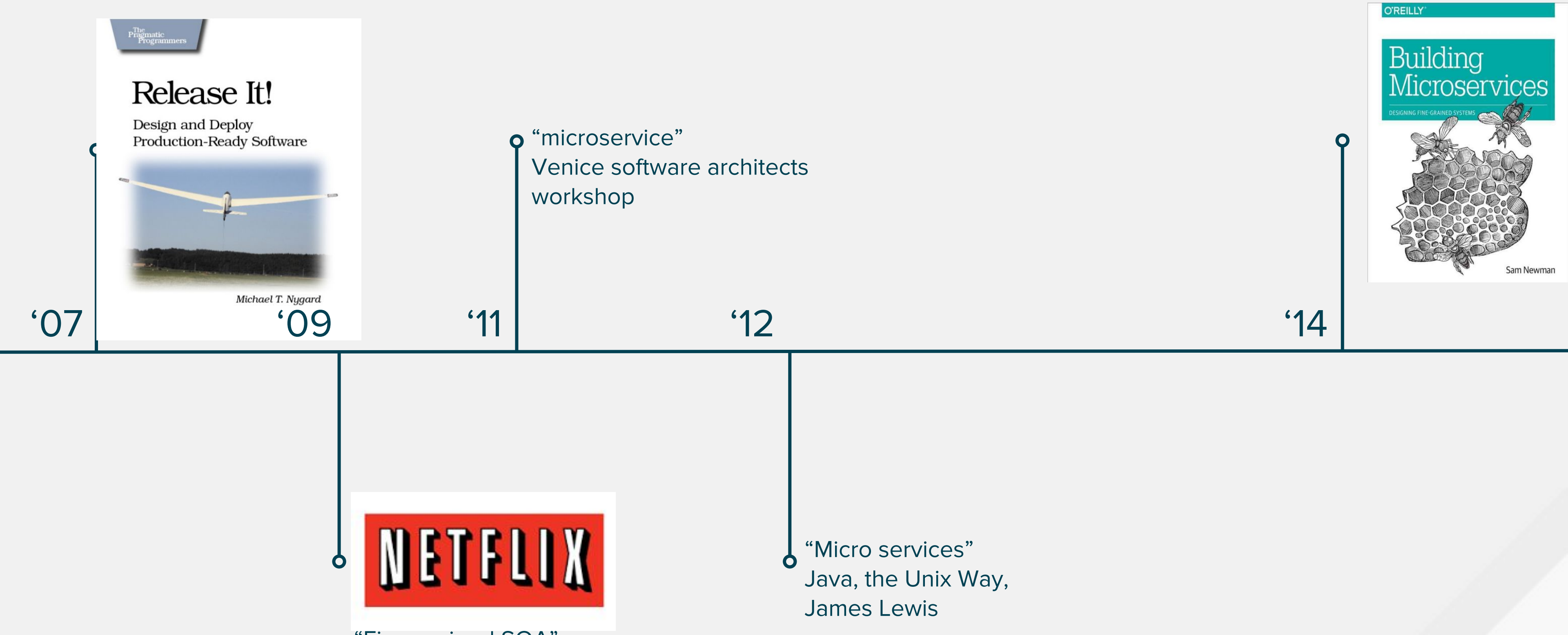
Distributed logging, Correlation IDs, Distributed Tracing

Patterns

Timeouts, Circuit breakers, Bulkheads, Idempotency, Fail fast, Decoupling Middleware, Event sourcing

Introduction

MICROSERVICES, THE TERM


The Pragmatic Programmers
Release It!
Design and Deploy Production-Ready Software
Michael T. Nygard
NETFLIX
O'REILLY
Building Microservices
DESIGNING FINE-GRAINED SYSTEMS
Sam Newman

‘07

‘09

“Fine grained SOA”
Adrian Cockcroft @
Netflix

‘11

“microservice”
Venice software architects
workshop

‘12

“Micro services”
Java, the Unix Way,
James Lewis

33rd Degree,
Kraków

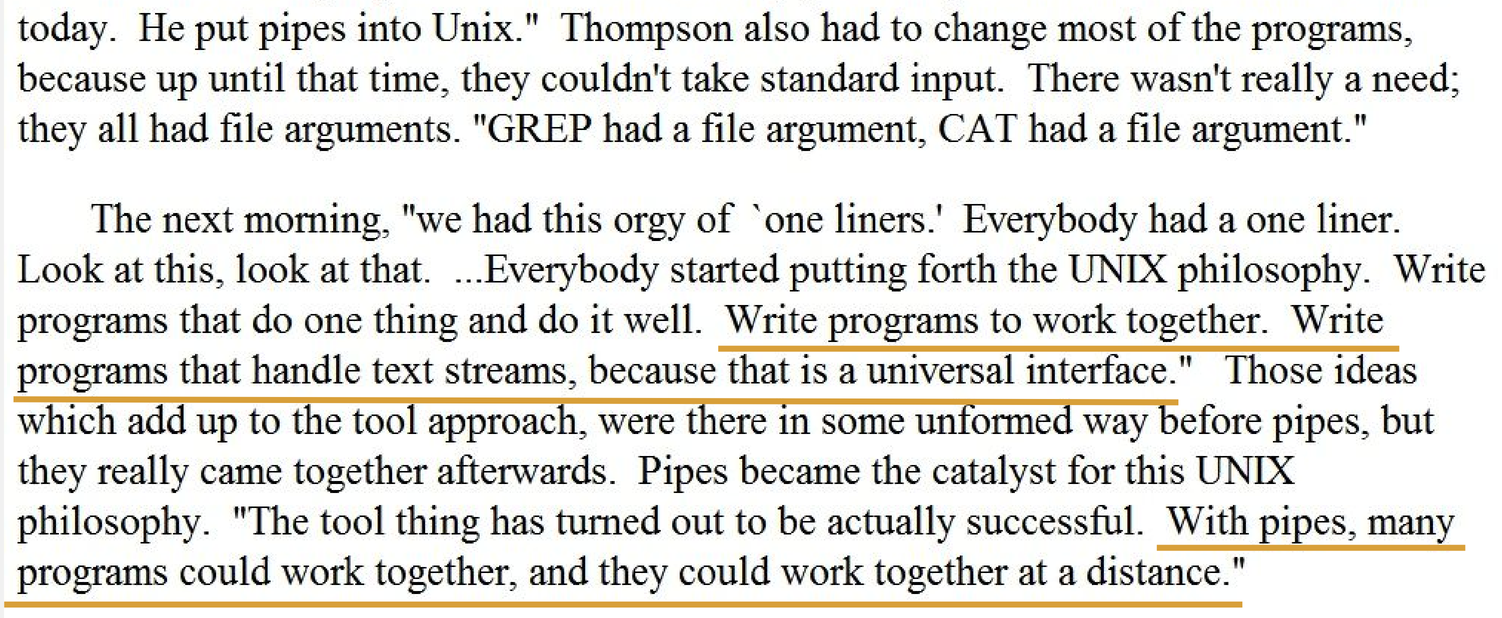
‘14

NETFLIX STORY

- …the very first piece of Netflix that was running in the cloud was the search auto-complete service. … That ran as a service, there was no graphics around it. All of the website that was supporting that was still running in the datacenter. It’s just that as you type that word in, it was sent off to a search index in the cloud…. .
- ..It’s a trivial piece of technology, but it taught us everything about pushing production systems to the cloud, hooking them up to a load balancer and the tooling we needed to do it. Two or three engineers, I think, worked on getting that built in a month or so maybe. It was a very small piece of work, plus the tooling, but it proved certain things worked. Then, we got the first bits and pieces up and running in the cloud one piece at a time…

-- Adrian Cockcroft

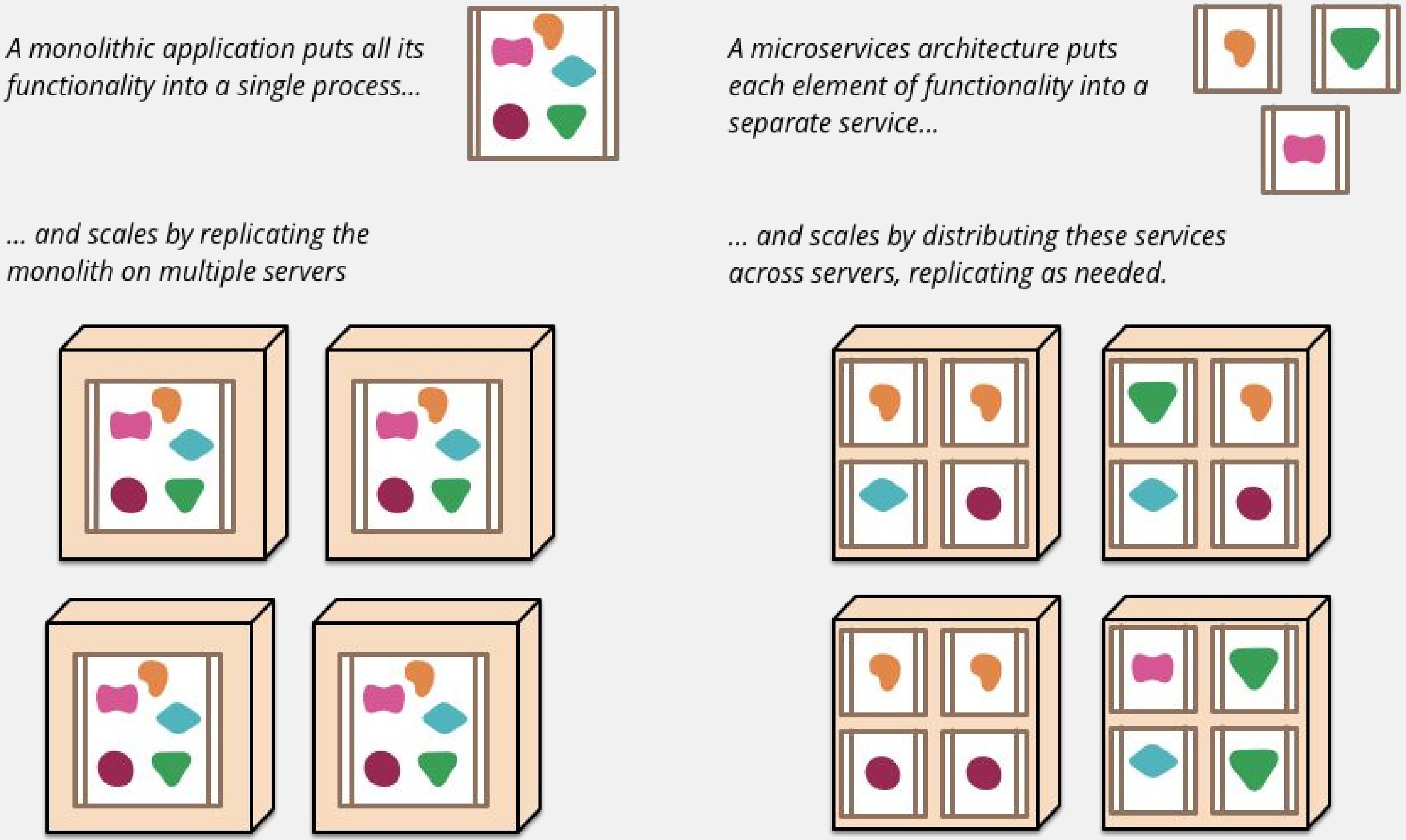
JAVA, THE UNIX WAY



today. He put pipes into Unix." Thompson also had to change most of the programs, because up until that time, they couldn't take standard input. There wasn't really a need; they all had file arguments. "GREP had a file argument, CAT had a file argument."

The next morning, "we had this orgy of `one liners.' Everybody had a one liner. Look at this, look at that. ...Everybody started putting forth the UNIX philosophy. Write programs that do one thing and do it well. Write programs to work together. Write programs that handle text streams, because that is a universal interface." Those ideas which add up to the tool approach, were there in some unformed way before pipes, but they really came together afterwards. Pipes became the catalyst for this UNIX philosophy. "The tool thing has turned out to be actually successful. With pipes, many programs could work together, and they could work together at a distance."

-- Lions commentary on UNIX 2nd Edition



A monolithic application puts all its functionality into a single process...

... and scales by replicating the monolith on multiple servers

A microservices architecture puts each element of functionality into a separate service...

... and scales by distributing these services across servers, replicating as needed.

Martin Fowler - **Microservices**
a definition of this new architectural term
https://martinfowler.com/articles/microservices.html

Concepts

SMALL AND AUTONOMOUS

Small

- Each application only does one thing
- Small enough to fit in your head

Packaging and Deployment

- single “fat jar” file
- independently testable
- installable as any system service
- containers
    - fat jar vs. shared layers

SMALL AND AUTONOMOUS

e.g. as a SystemD service

```
[Unit]
Description=Store Catalogue
After=network.service

[Service]
Environment=SPRING_DATASOURCE_URI=jbdc://foo/bar
ExecStart=/bin/sh -c 'java -jar /opt/store/catalogue.jar'
User=catalogue

[Install]
WantedBy=multi-user.target
```

LOOSE COUPLING

…and tight cohesion

- The Single Responsibility Principle
  - Gather together those things that change for the same reason, and separate those things that change for different reasons.
- Services separated via network calls
  - APIs
    - JSON/XML over HTTP (REST), Messaging
    - versioning

CONWAY’S LAW

Organizations which design systems are constrained to produce designs whose structure are copies of the communication structures of these organizations -- Melvin Conway 1968

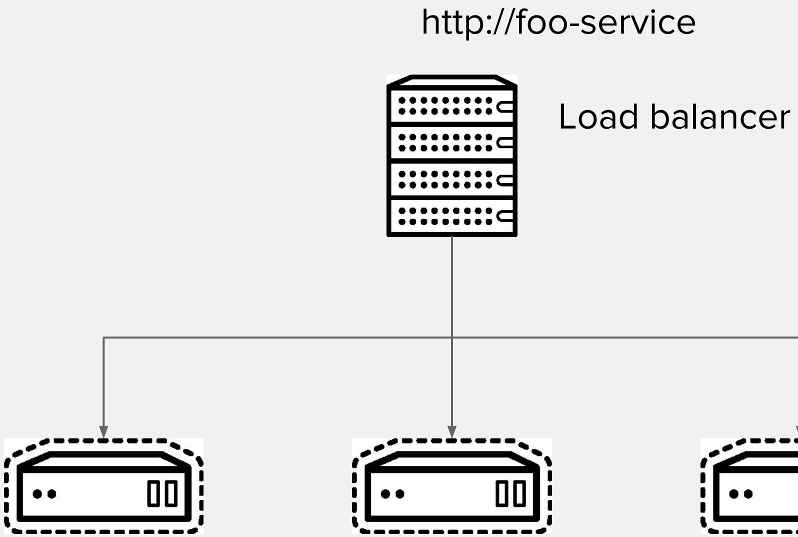
- Each application in separate source repository
  - Treat common code as any other shared library
- Mythical Man-Month
- 2 pizza rule

RESILIENCE AND SCALING

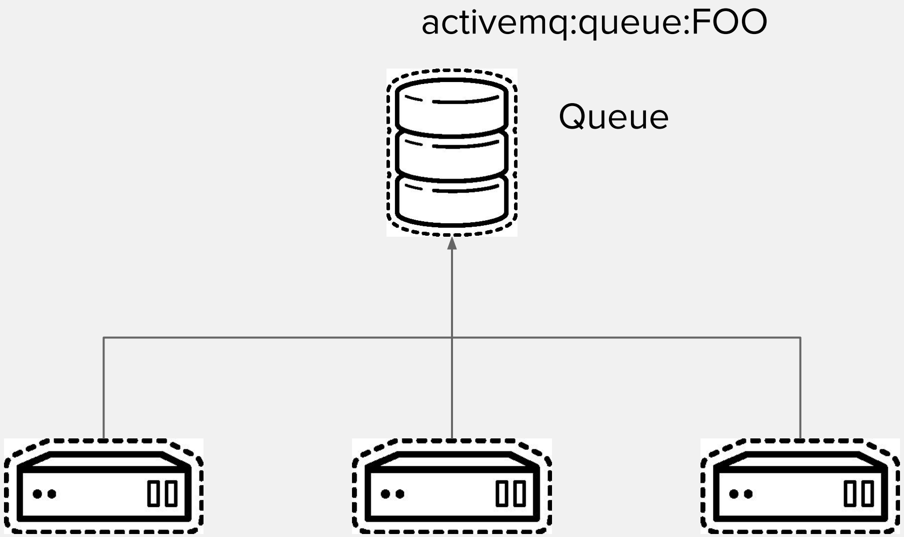
Design for failure

- Resilience
  - A resilient system keeps processing transactions, even when there are transient impulses, persistent stresses, or component failures disrupting normal processing.
  - Ability to contain a failure to the failing component.
- Scaling
  - horizontal scaling
    - fault tolerance via redundancy

LOAD BALANCING

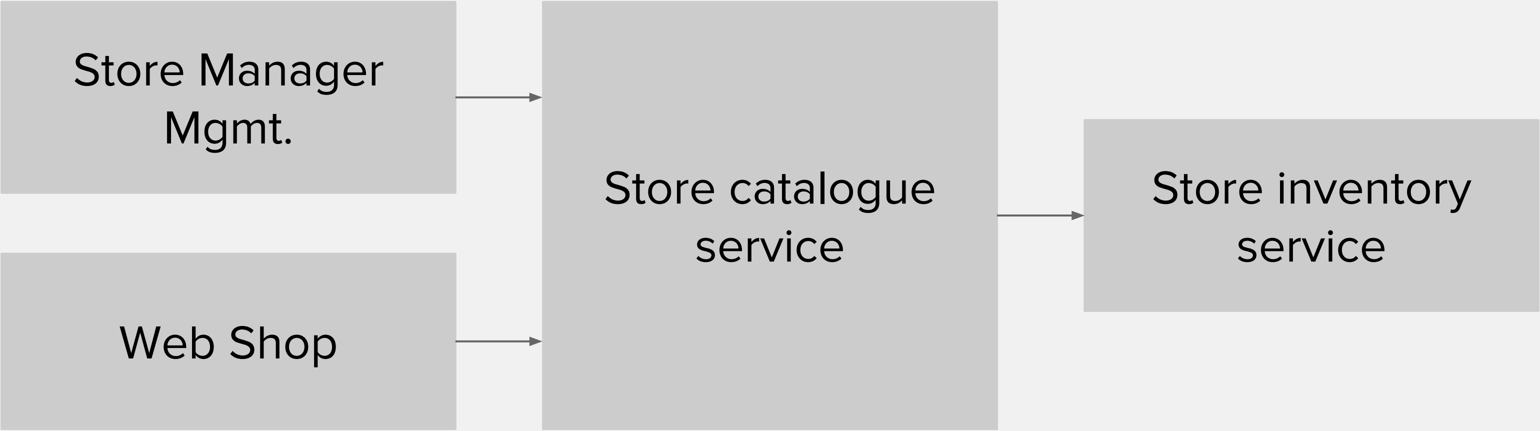


WORKER BASED SCALING



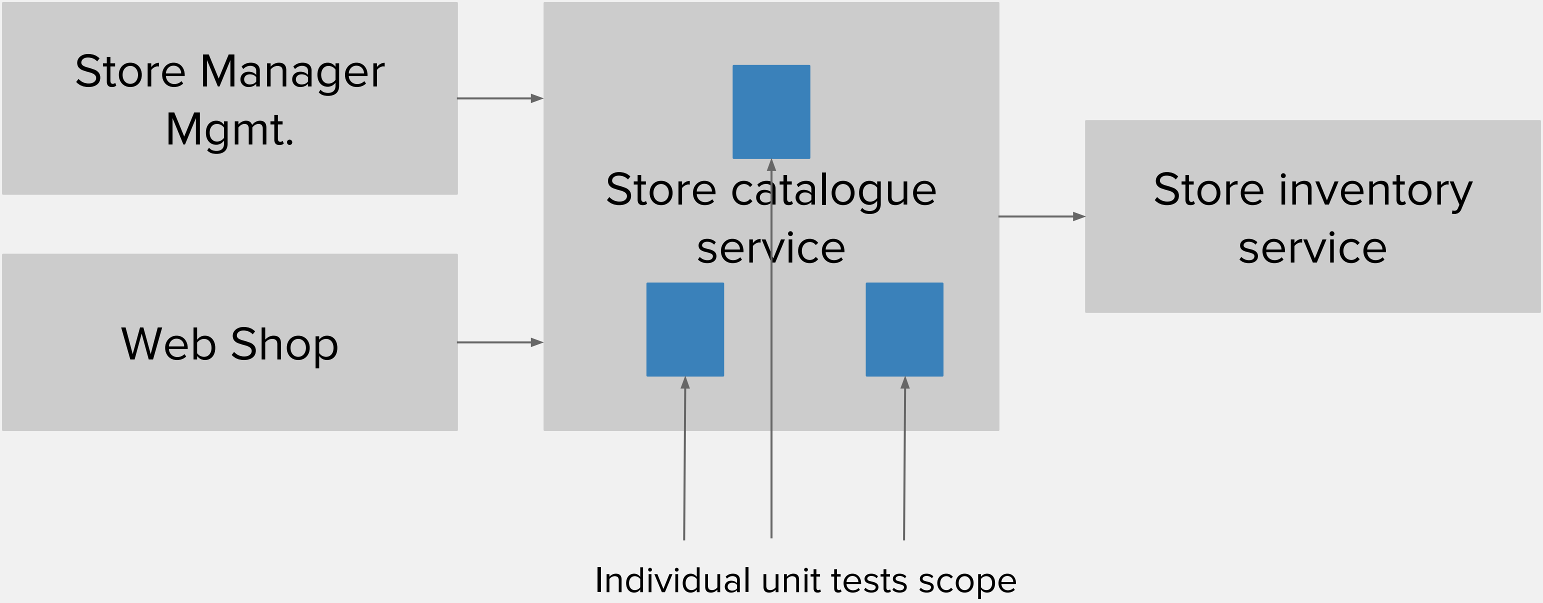
Testing

TEST CATEGORIES



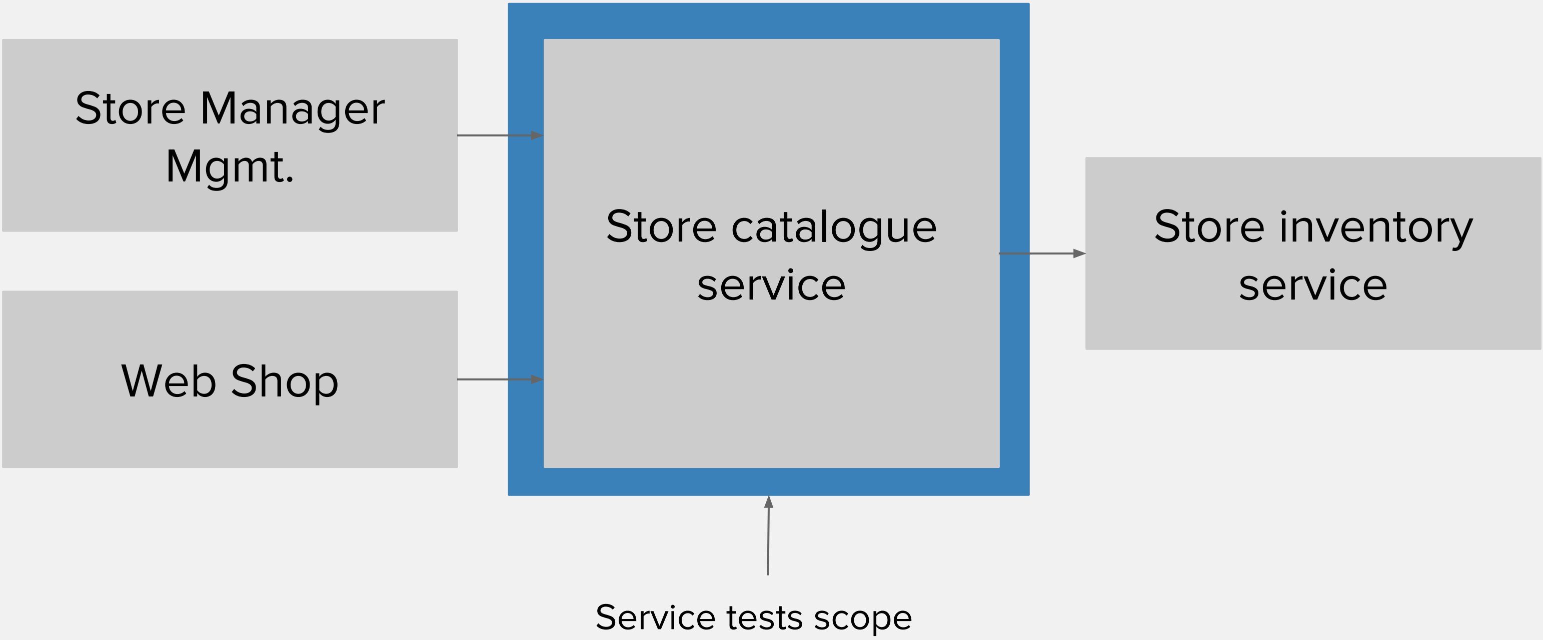
TEST CATEGORIES

Unit tests



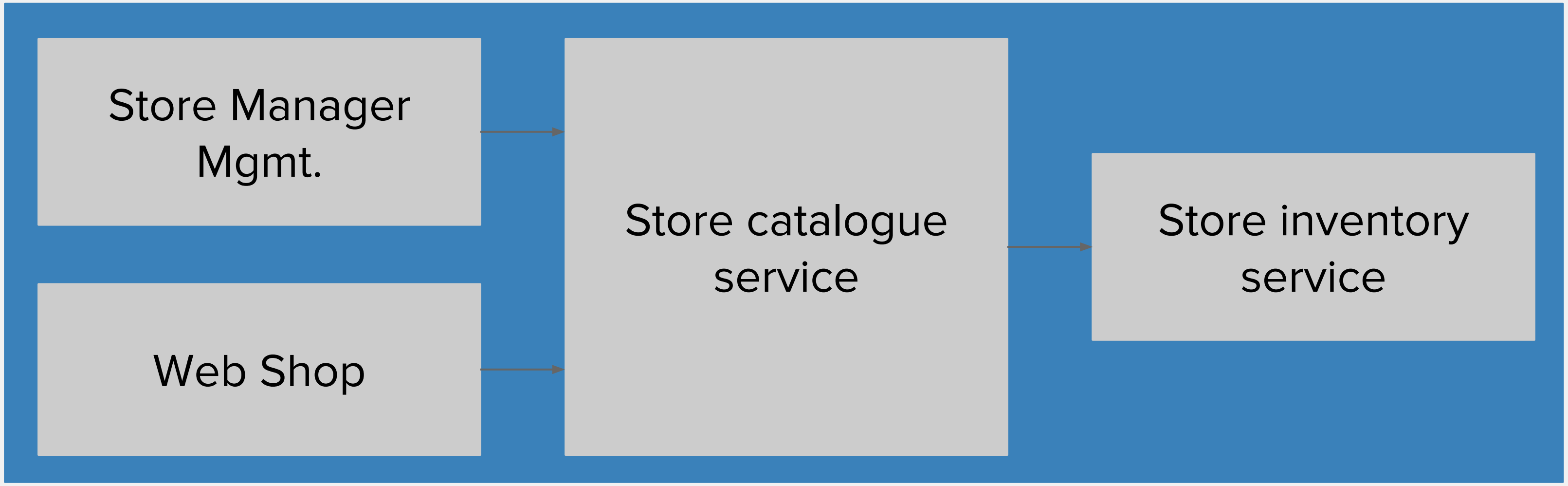
TEST CATEGORIES

Service tests



TEST CATEGORIES

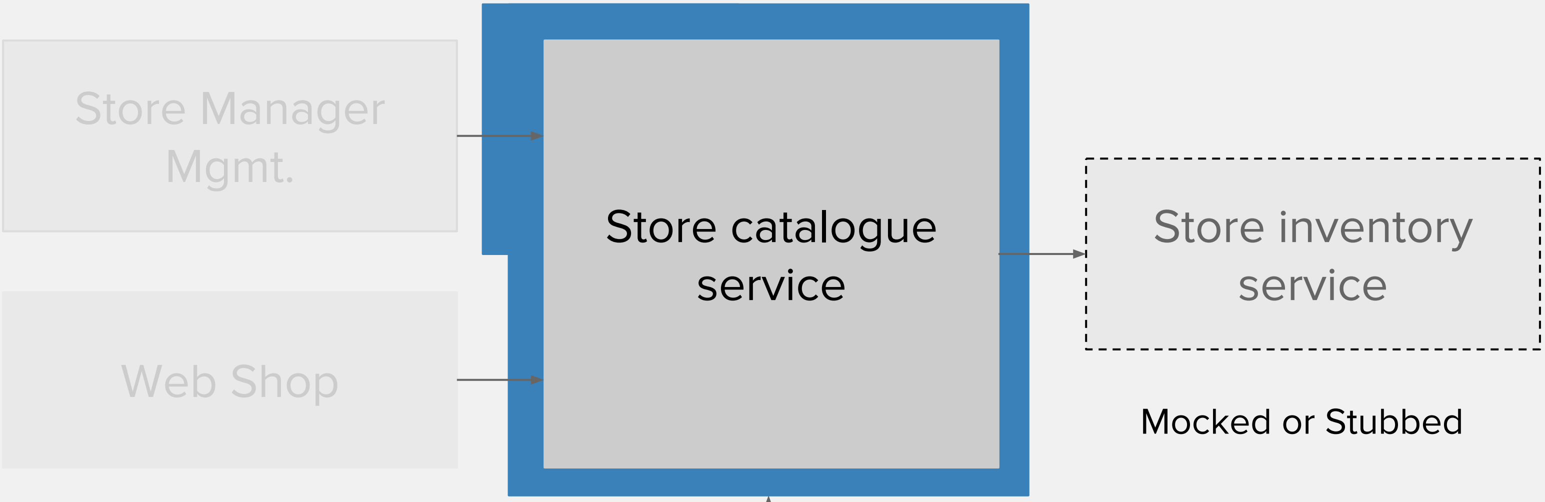
End-to-end tests



End-to-end tests scope

TEST CATEGORIES

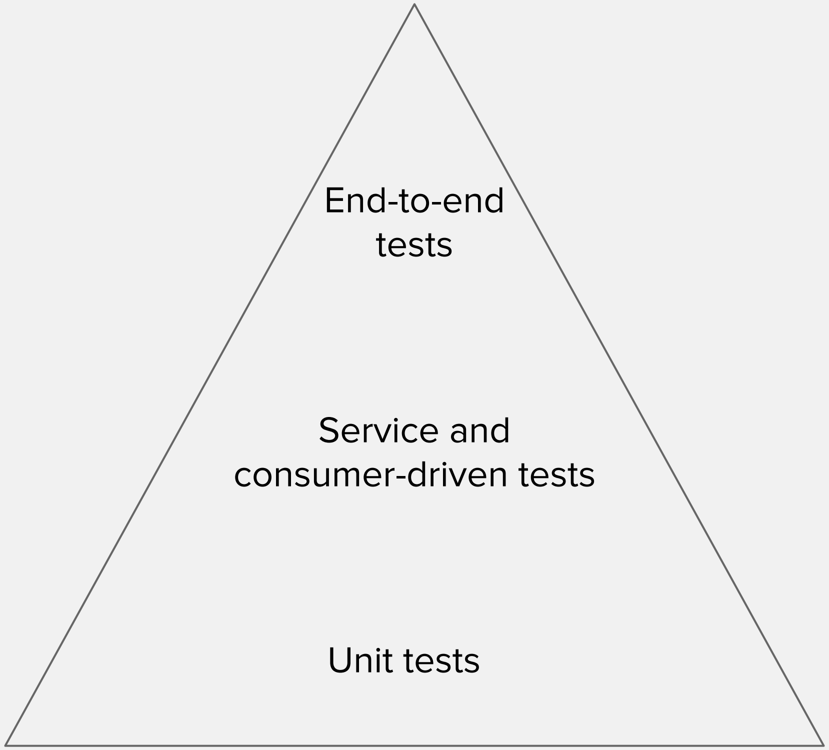
Consumer-driven contracts



Mocked or Stubbed

Store Manager Mgmt. CDC Test scope

TEST CATEGORIES



TESTING IN PRODUCTION

Testing environment is never identical to production

- Blue/green deployment
- Canary release
- Semantic monitoring
- Chaos monkey

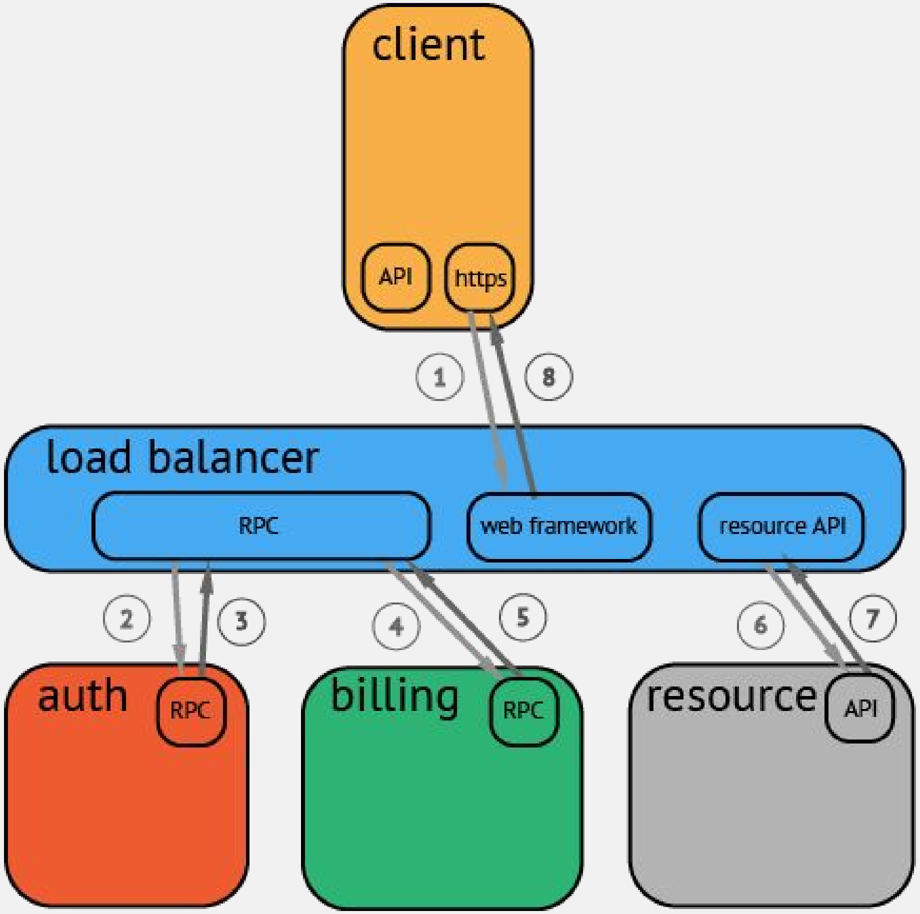
Monitoring

LOGGING

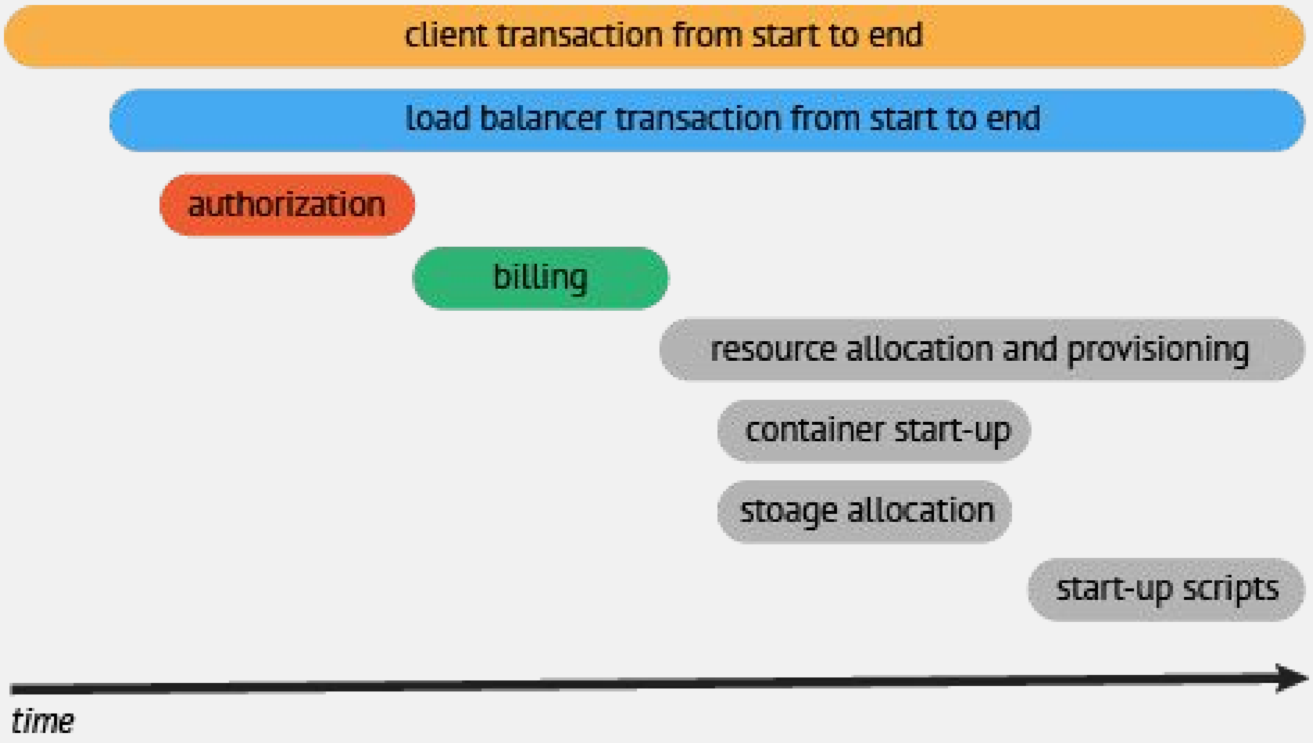
aggregation

- ELK / EFK
- Correlation ID
- Distributed Tracing

DISTRIBUTED TRACING



DISTRIBUTED TRACING

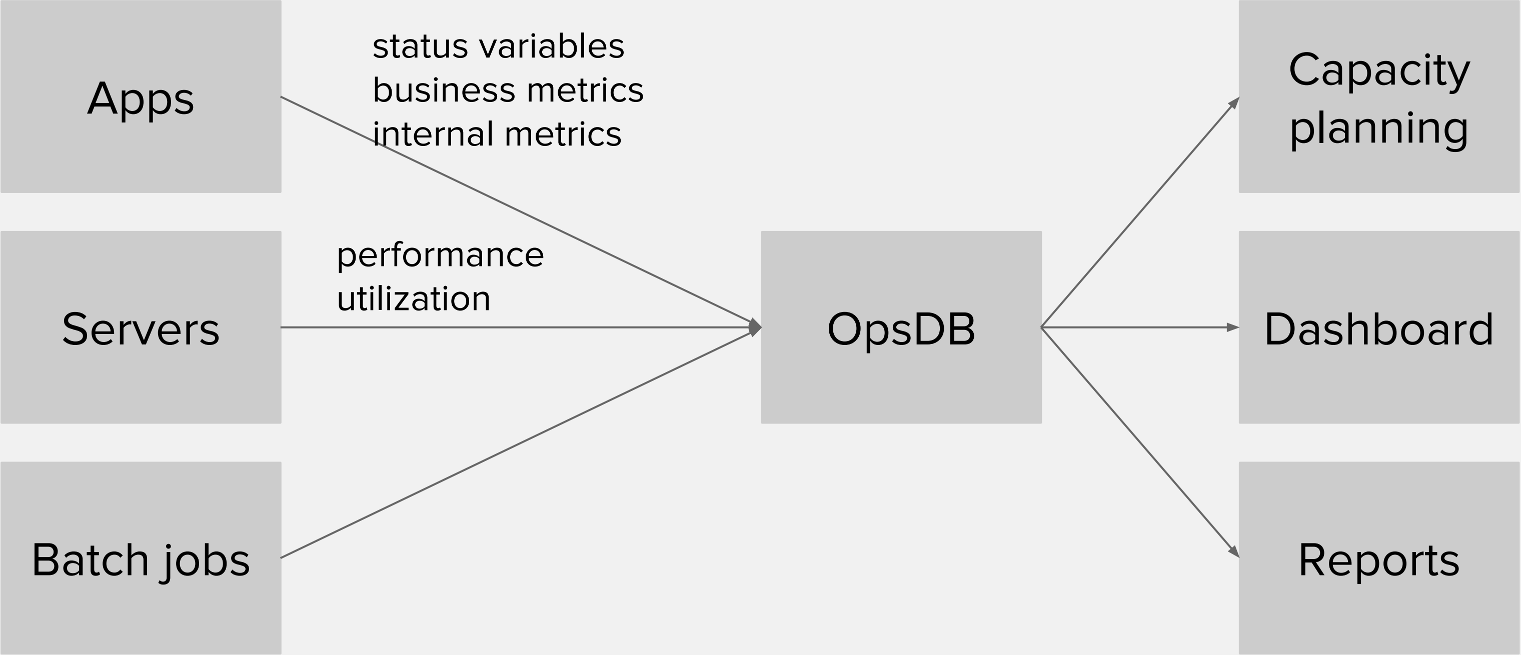

client transaction from start to end
load balancer transaction from start to end
authorization
billing
resource allocation and provisioning
container start-up
stoage allocation
start-up scripts
time

MONITORING

individual services

- Mechanisms
  - JMX / Jolokia
- Data
  - Traffic indicators
  - Resource pool health
  - DB connections health
  - Integration point health
  - Cache health

OPERATIONS DATABASE



Apps

Servers

Batch jobs

status variables
business metrics
internal metrics

performance
utilization

OpsDB

Capacity planning

Dashboard

Reports

Patterns

TIMEOUTS

networks are fallible

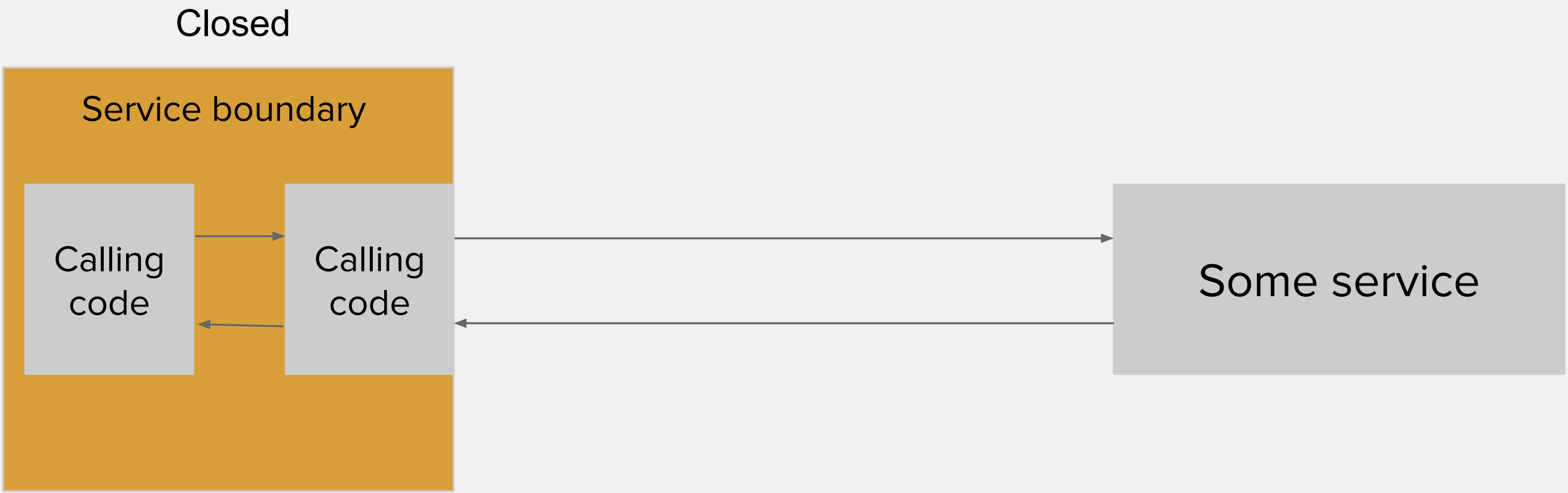
- prevents calls to integration points from causing blocked threads
    - cascading failures
- thread / connection pools
- consider delayed retries / queueing
    - Beware of Acts of self-denial

CIRCUIT BREAKER

“don’t do it if it hurts”

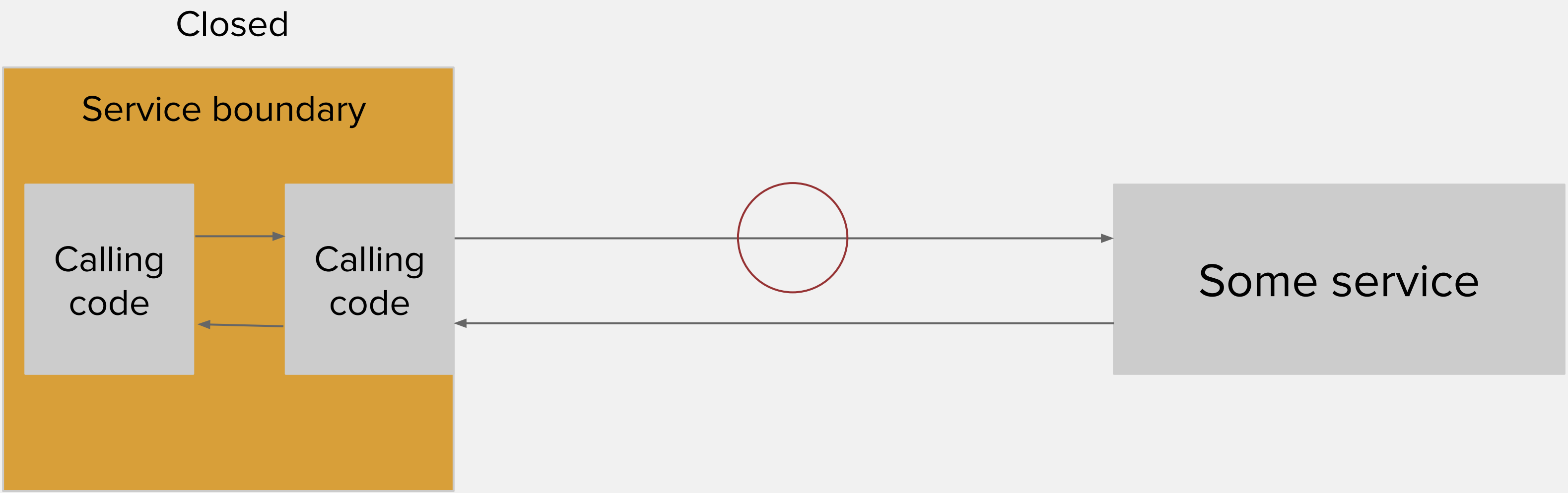
CIRCUIT BREAKER

Service functions normally



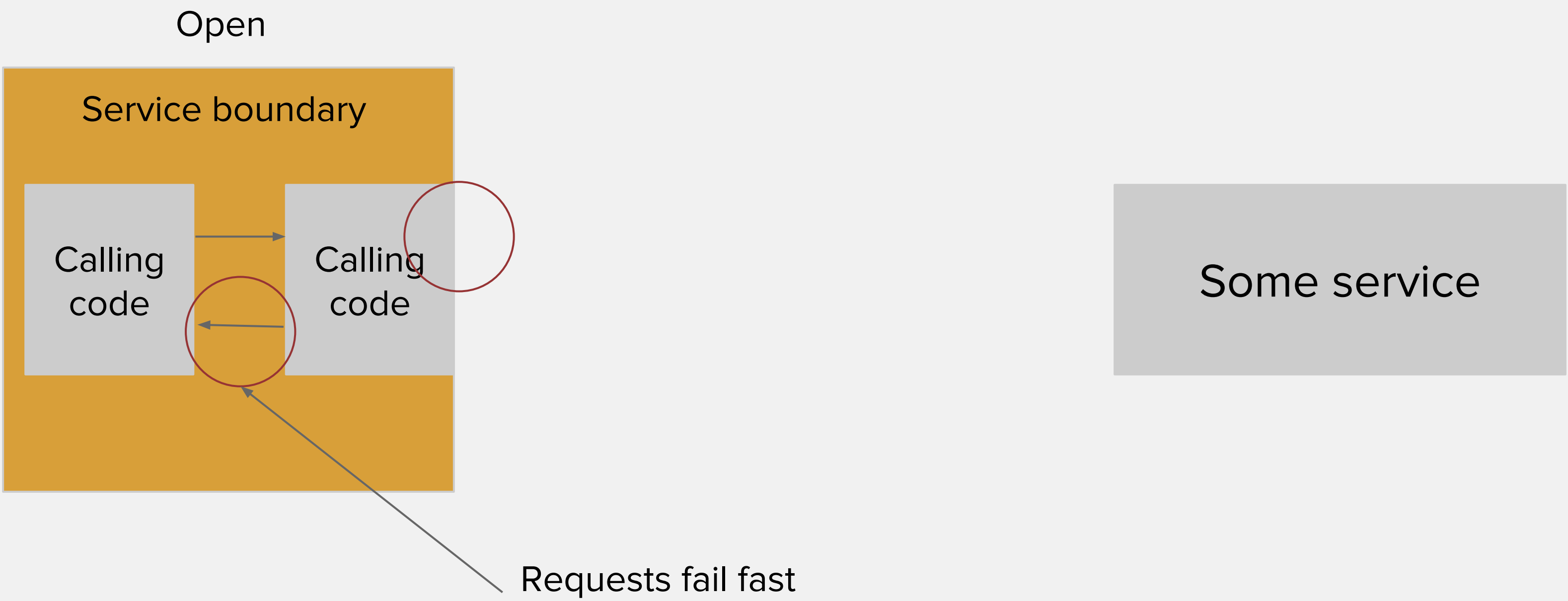
CIRCUIT BREAKER

Calls starting to timeout or returning errors



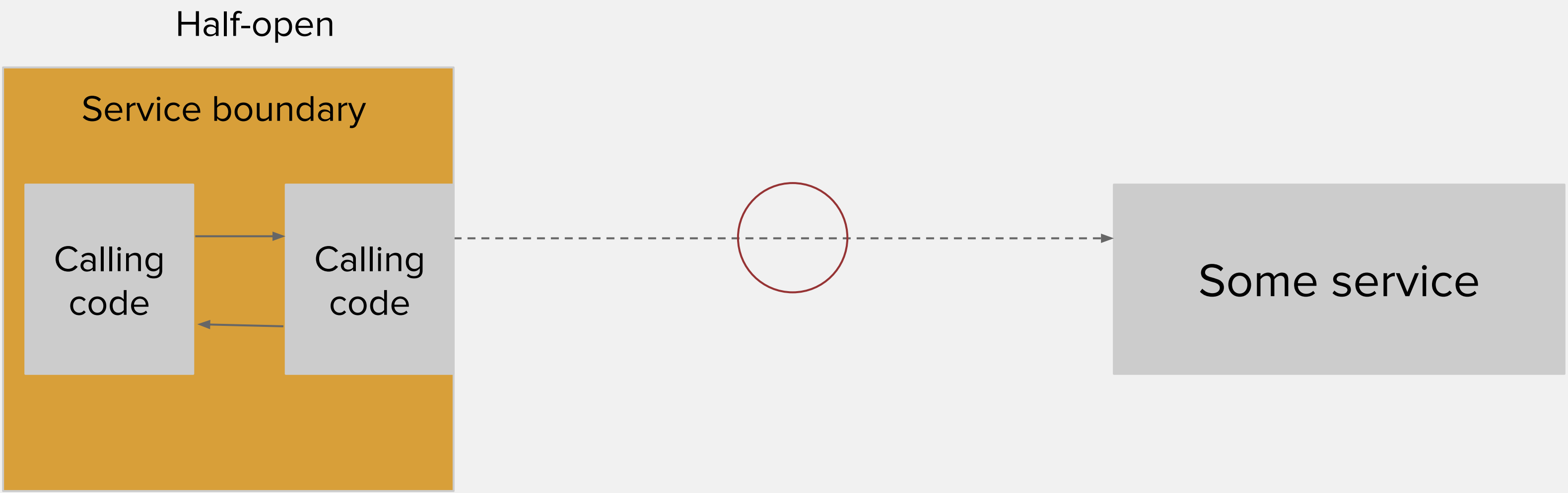
CIRCUIT BREAKER

Connection stopped when threshold reached



CIRCUIT BREAKER

Retry after grace period, or occasional health check



Half-open

Service boundary

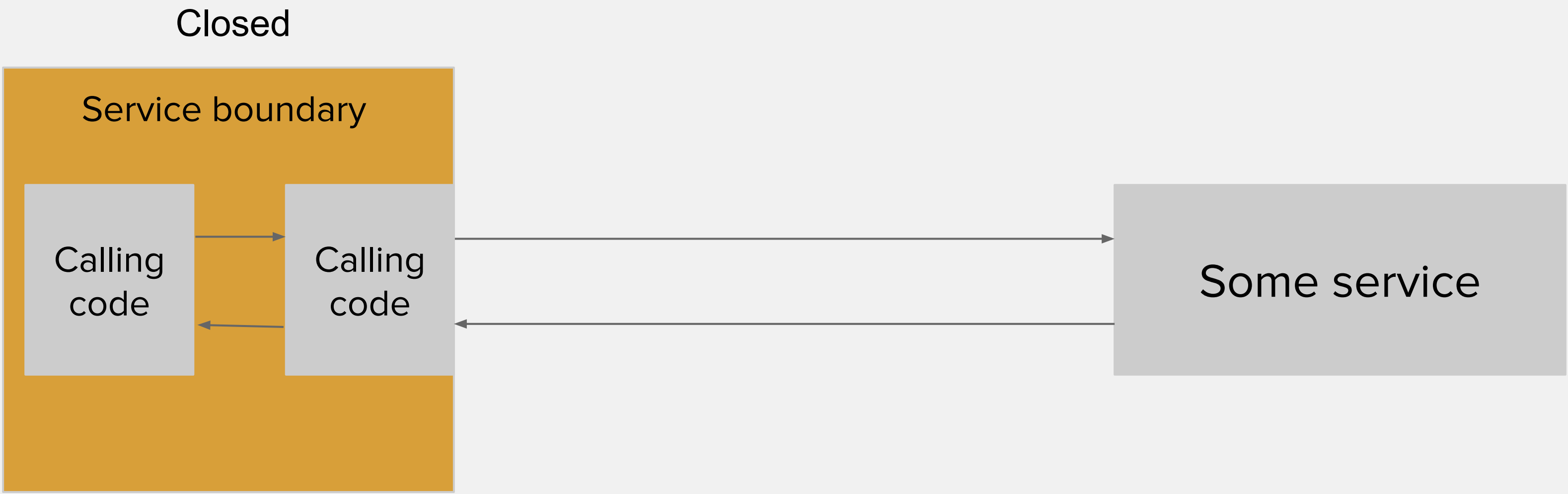
Calling code

Calling code

Some service

CIRCUIT BREAKER

Connection reset when healthy threshold reached


Closed

Service boundary

Calling code

Calling code

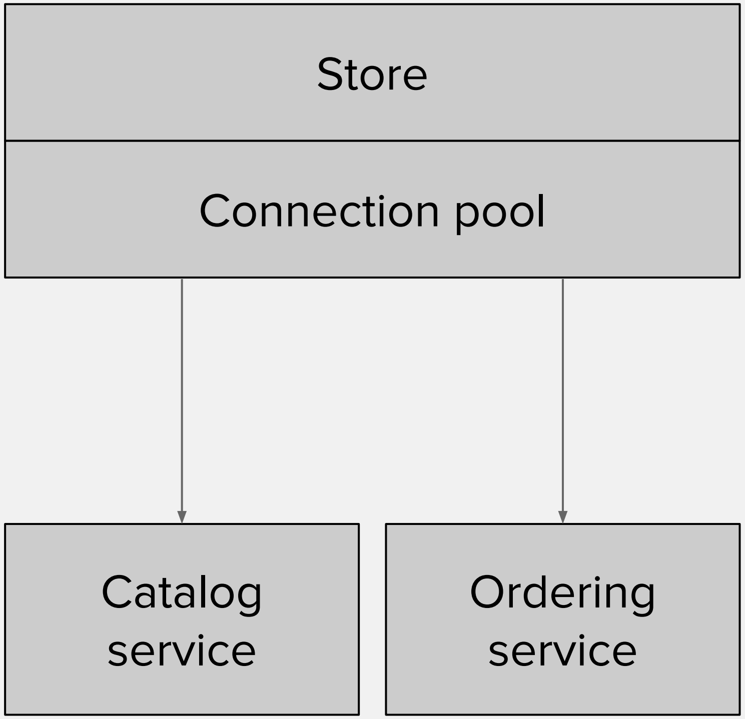
Some service

BULKHEADS

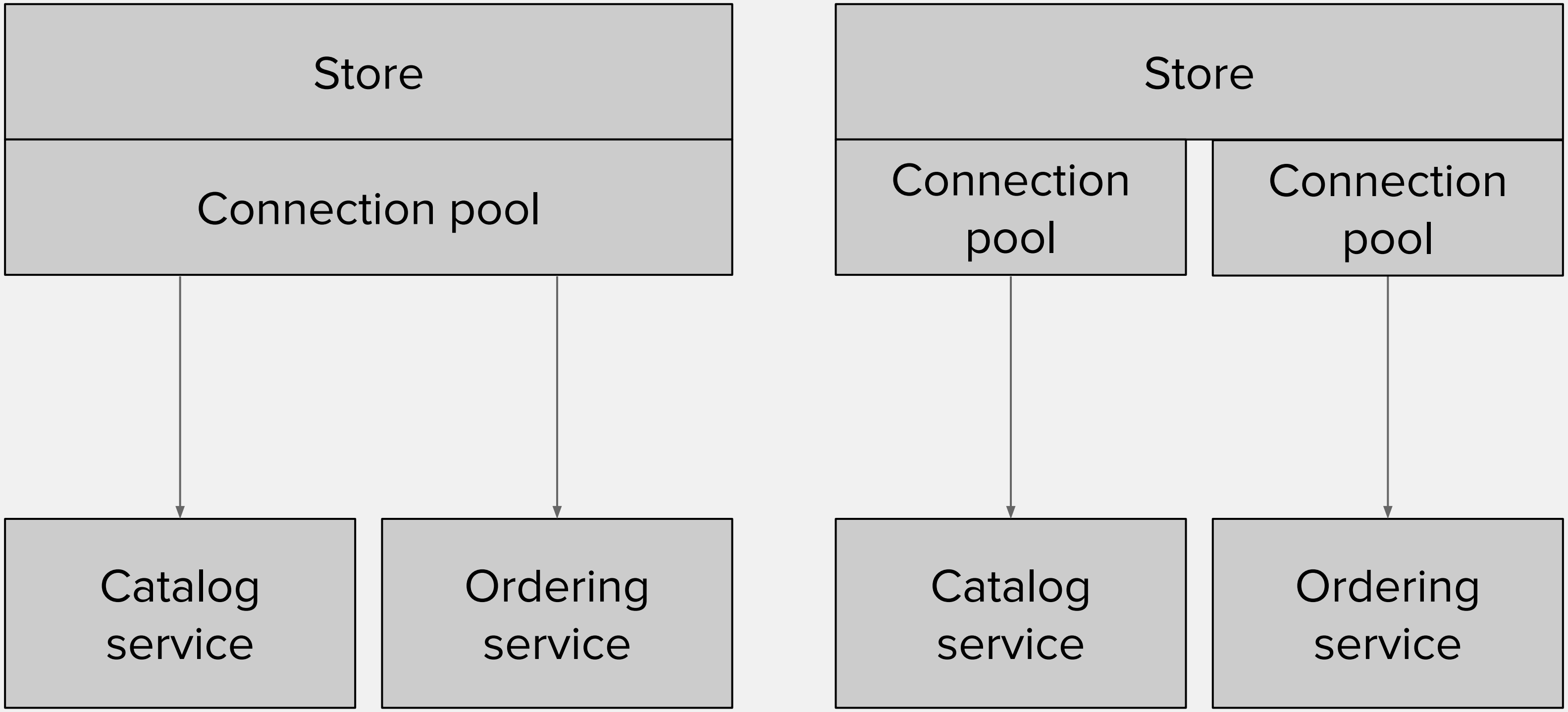
isolating from failure

- Separation of concerns (separate service)
- Separating thread/connection pools

BULKHEADS

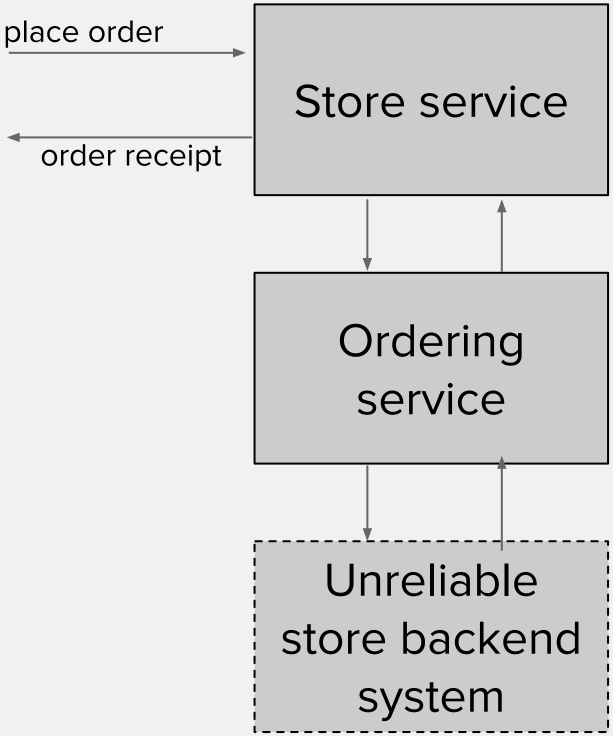


BULKHEADS


Store
Connection pool
Catalog service
Ordering service
Store
Connection pool
Connection pool
Catalog service
Ordering service

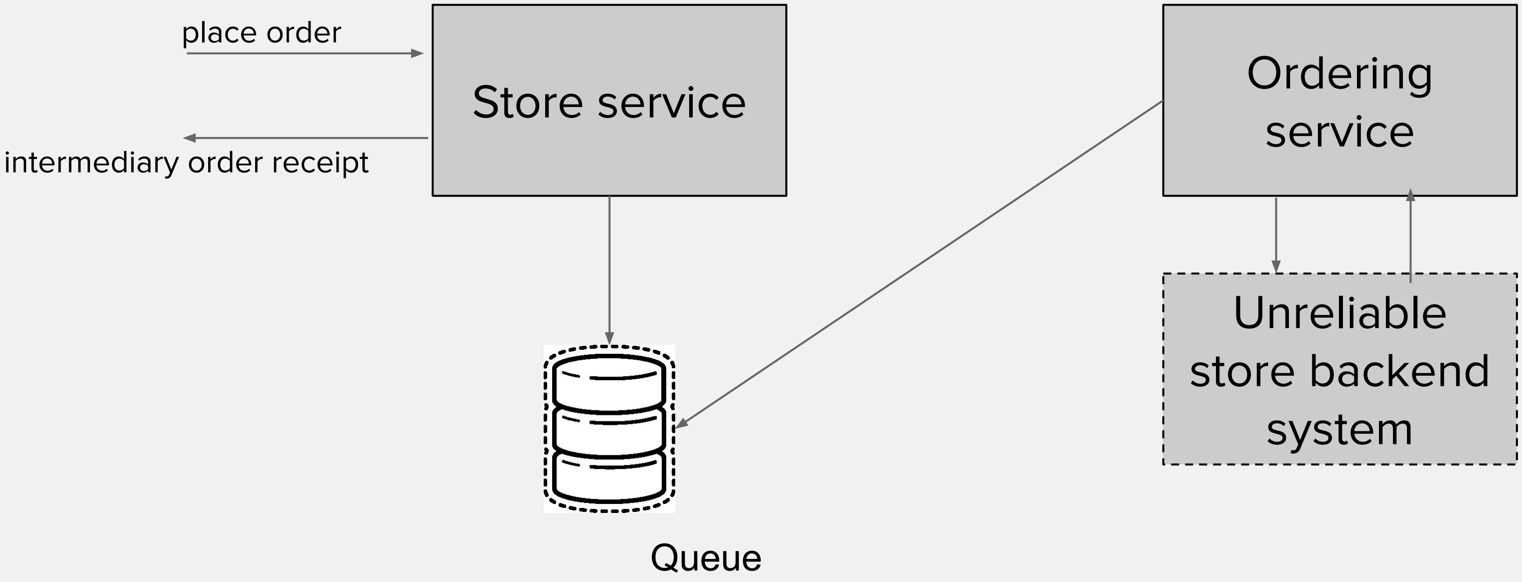
DECOUPLING MIDDLEWARE

enable message processing in different place and time

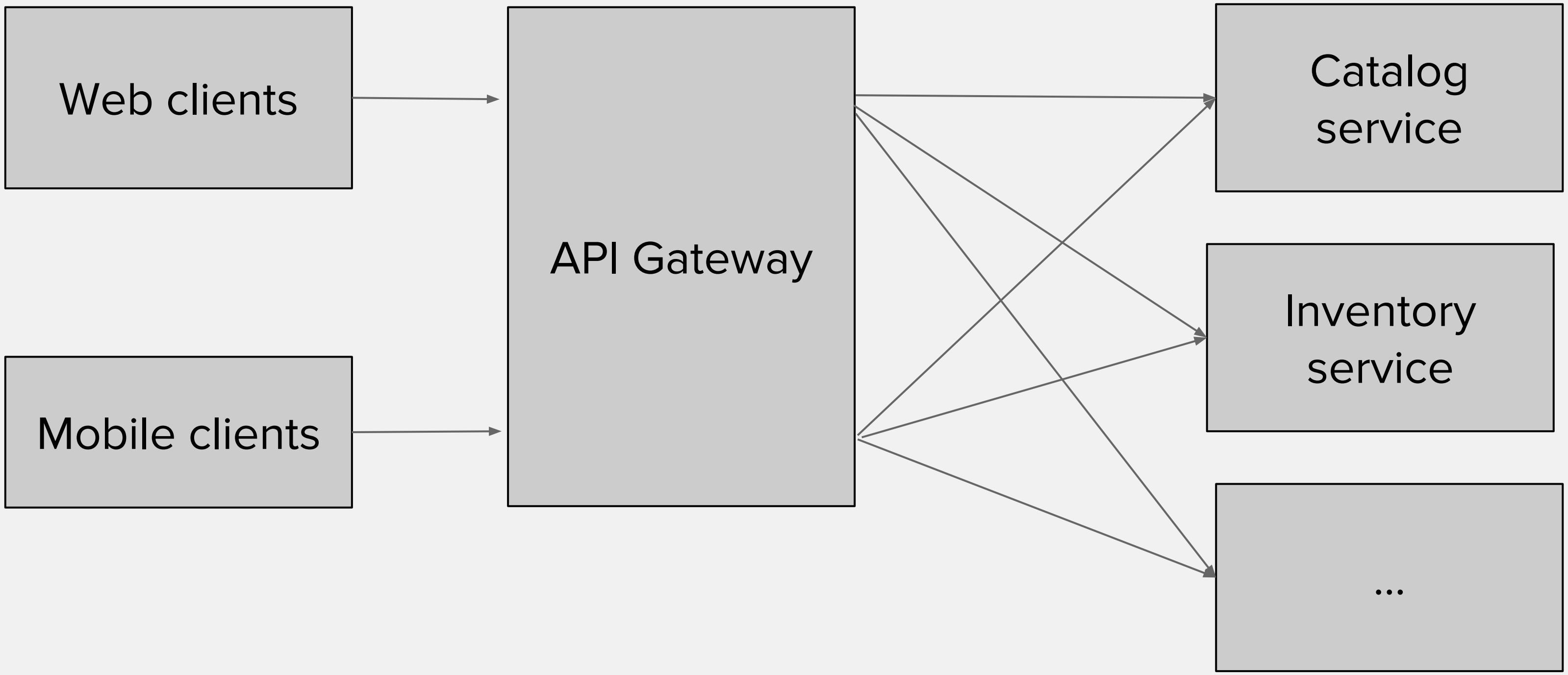


DECOUPLING MIDDLEWARE

enable message processing in different place and time



API GATEWAY


Web clients
Mobile clients
API Gateway
Catalog service
Inventory service
...

API GATEWAY

is not just a proxy

- merge multiple calls (saving network overhead)
- different APIs for different clients
- API management

IDEMPOTENCY

and eventual consistency

IDEMPOTENCY

PUT http://order-service/order

```
{
  “customer”: {
    “name”: “John Doe”,
    ...
  },
  “items” = [
    {“storeId”: 1000, “itemPrice”: 99.90, “amount”: 2}
  ]
}
```

IDEMPOTENCY

PUT http://order-service/order

```
{
  “uuid”: “12de-adbe-ef42”,
  “customer”: {
    “name”: “John Doe”,
    ...
  },
  “items” = [
    {“storeId”: 1000, “itemPrice”: 99.90, “amount”: 2}
  ]
}
```

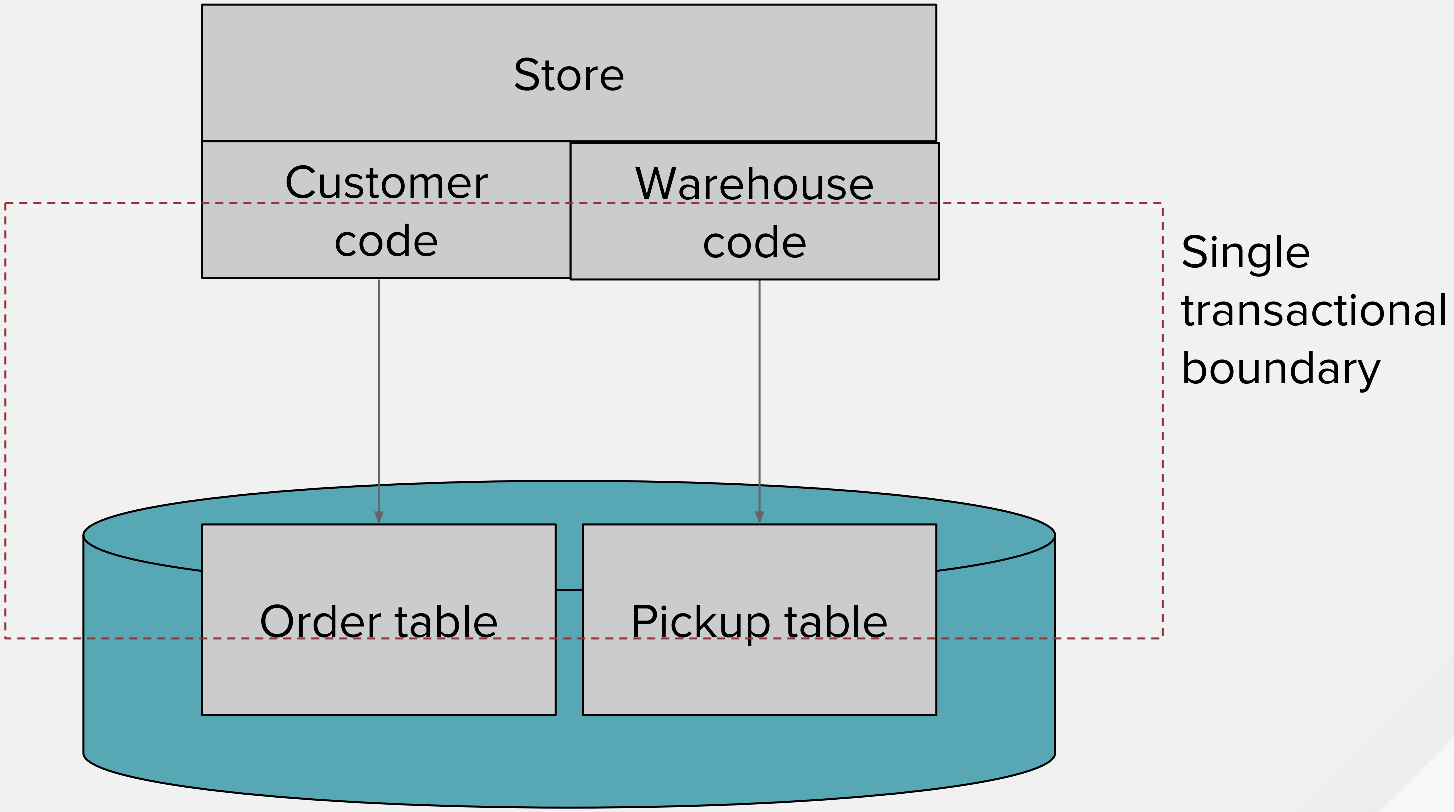
CAP THEOREM

Two out of three ain’t bad

- Consistency
- Availability
- Partition Tolerance

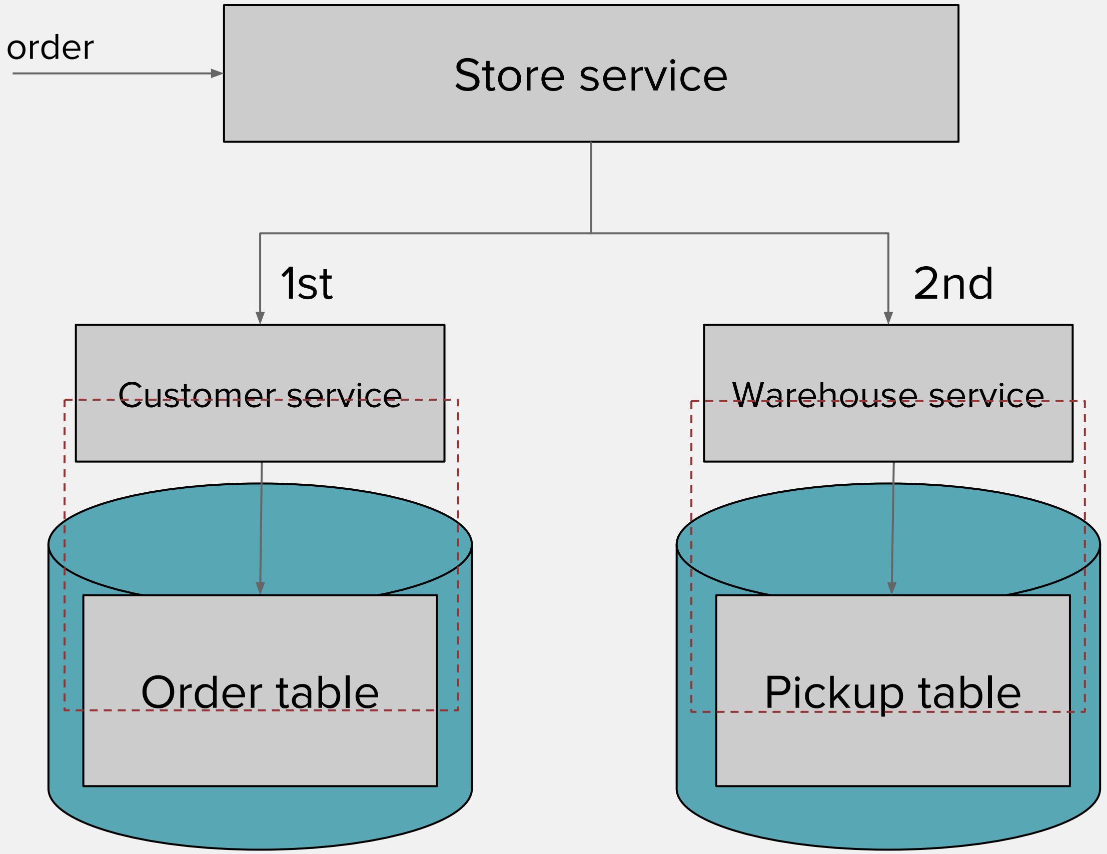
NON-DISTRIBUTED

Monolithic service



INCONSISTENT APPROACH

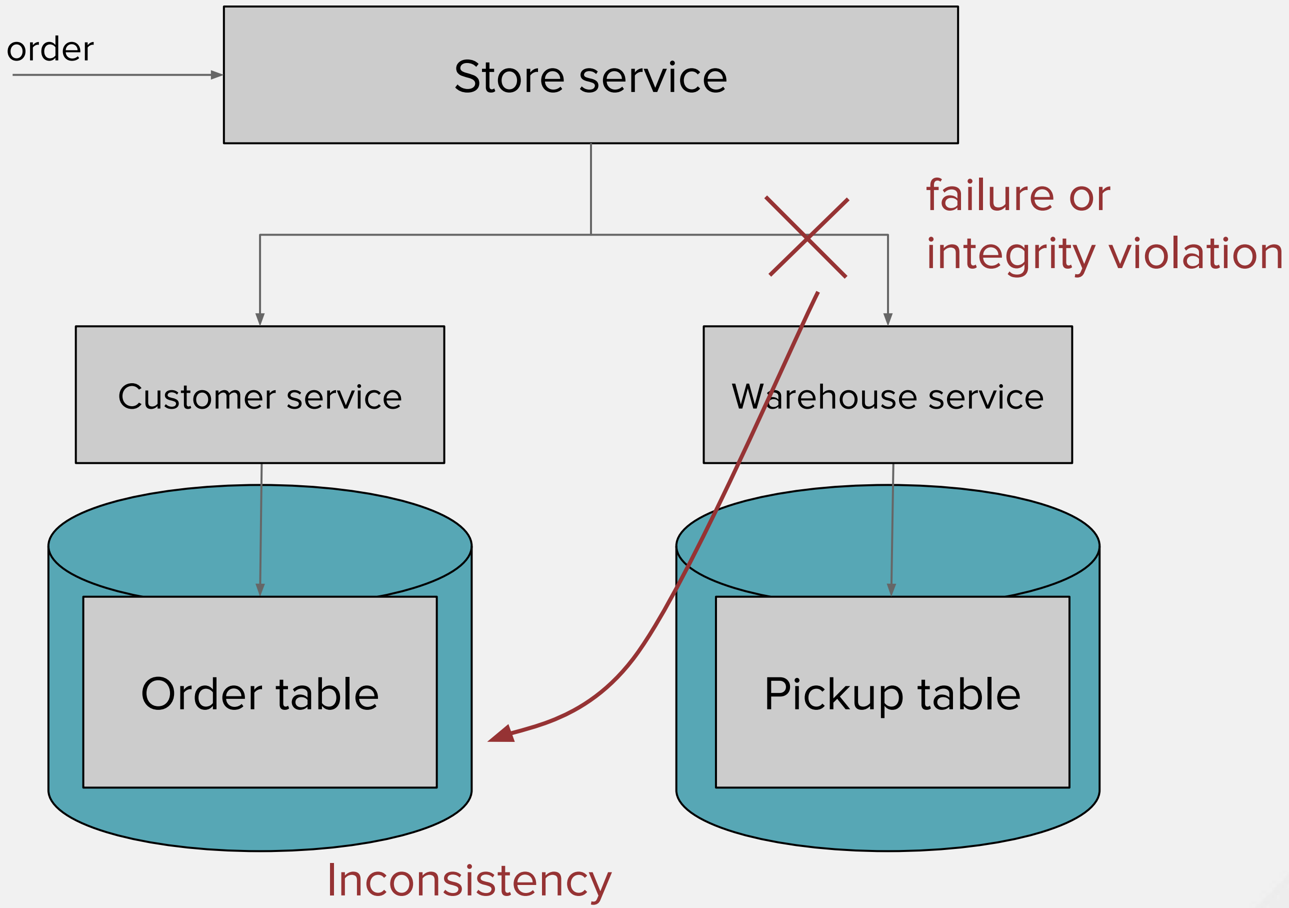
Separate services



Separate transactional boundaries

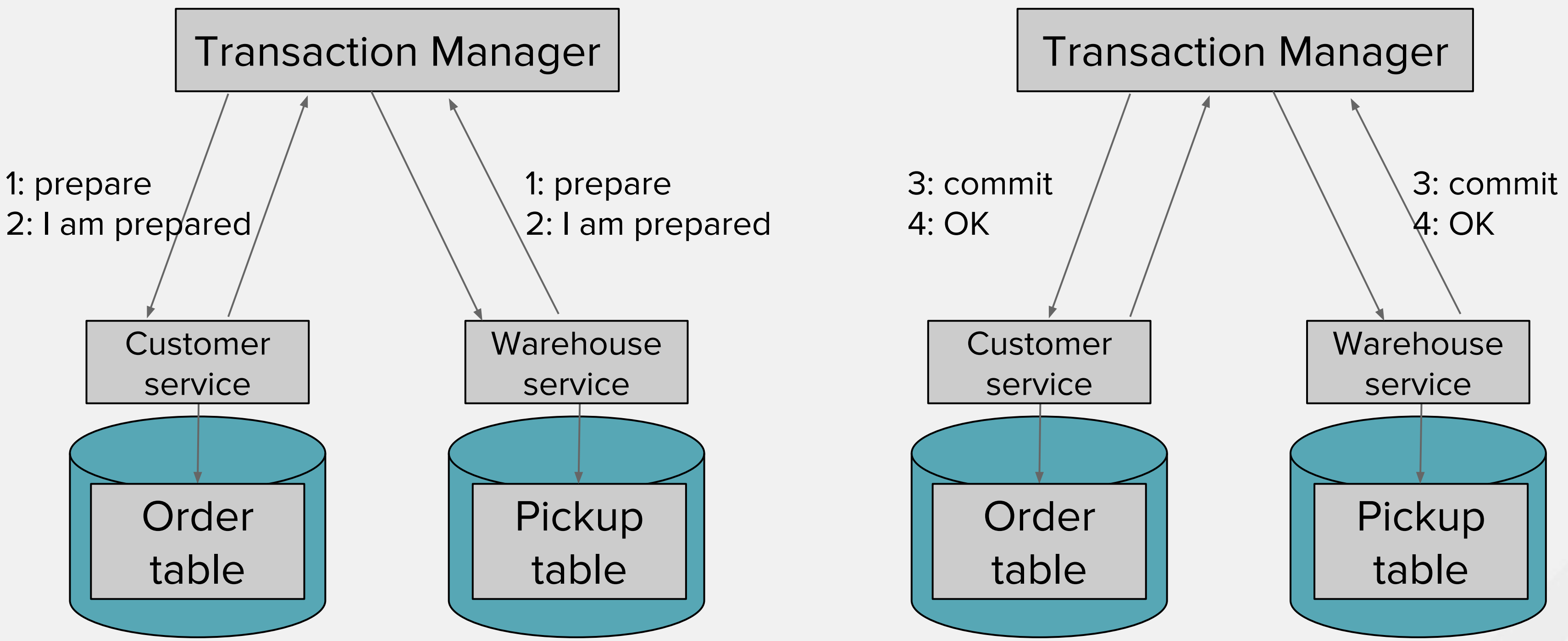
INCONSISTENT APPROACH

Separate services



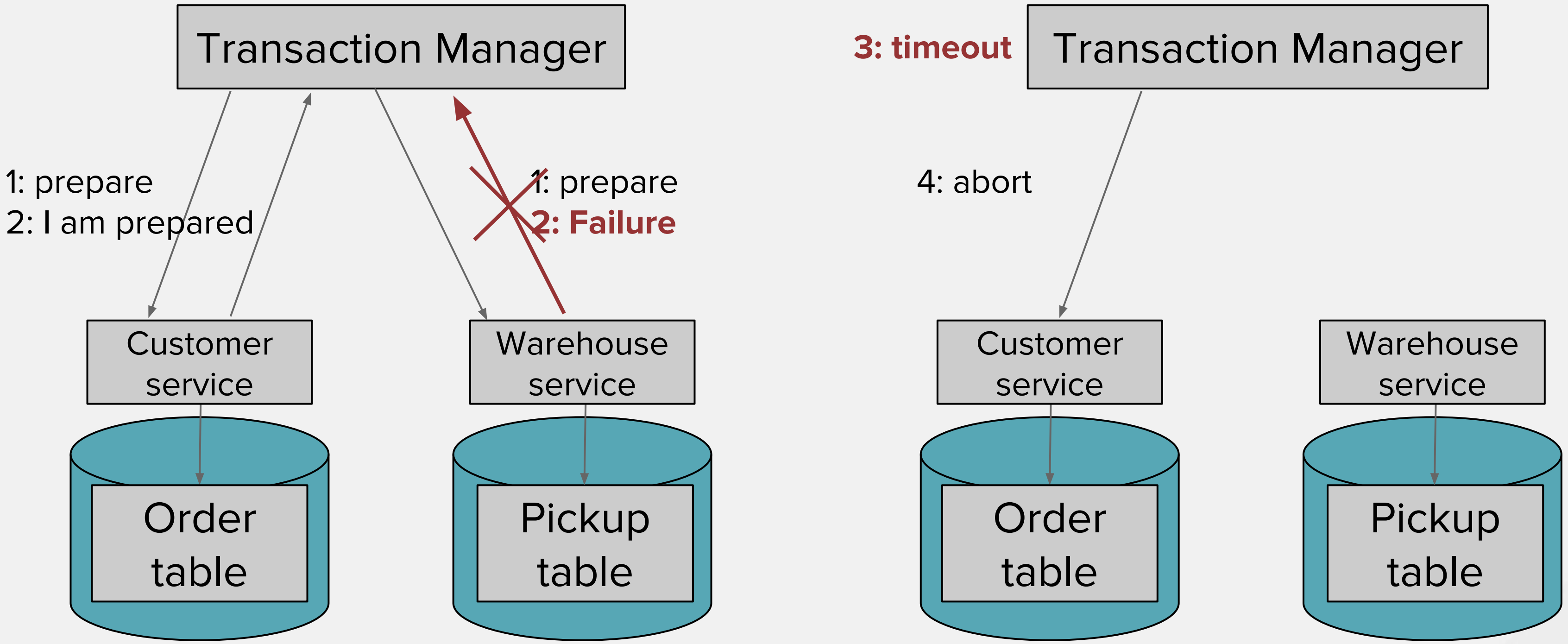
(almost) CONSISTENT APPROACH

Block writes until all services are ready to commit (two-phased commit)



(almost) CONSISTENT APPROACH

Block writes until all services are ready to commit (two-phased commit)



Transaction Manager

1: prepare
2: I am prepared

1: prepare
2: Failure

Customer service

Order table

Warehouse service

Pickup table

3: timeout

Transaction Manager

4: abort

Customer service

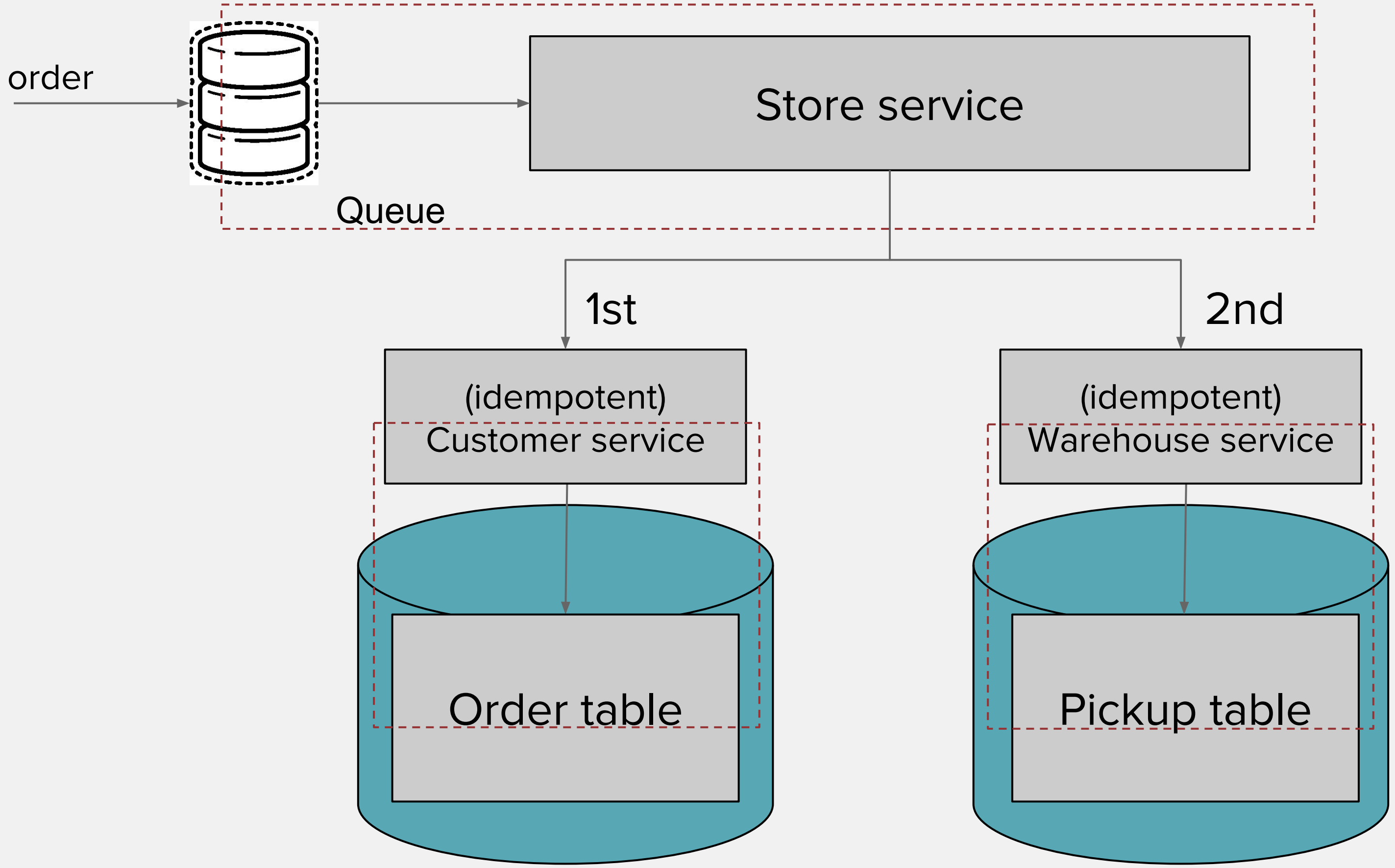
Order table

Warehouse service

Pickup table

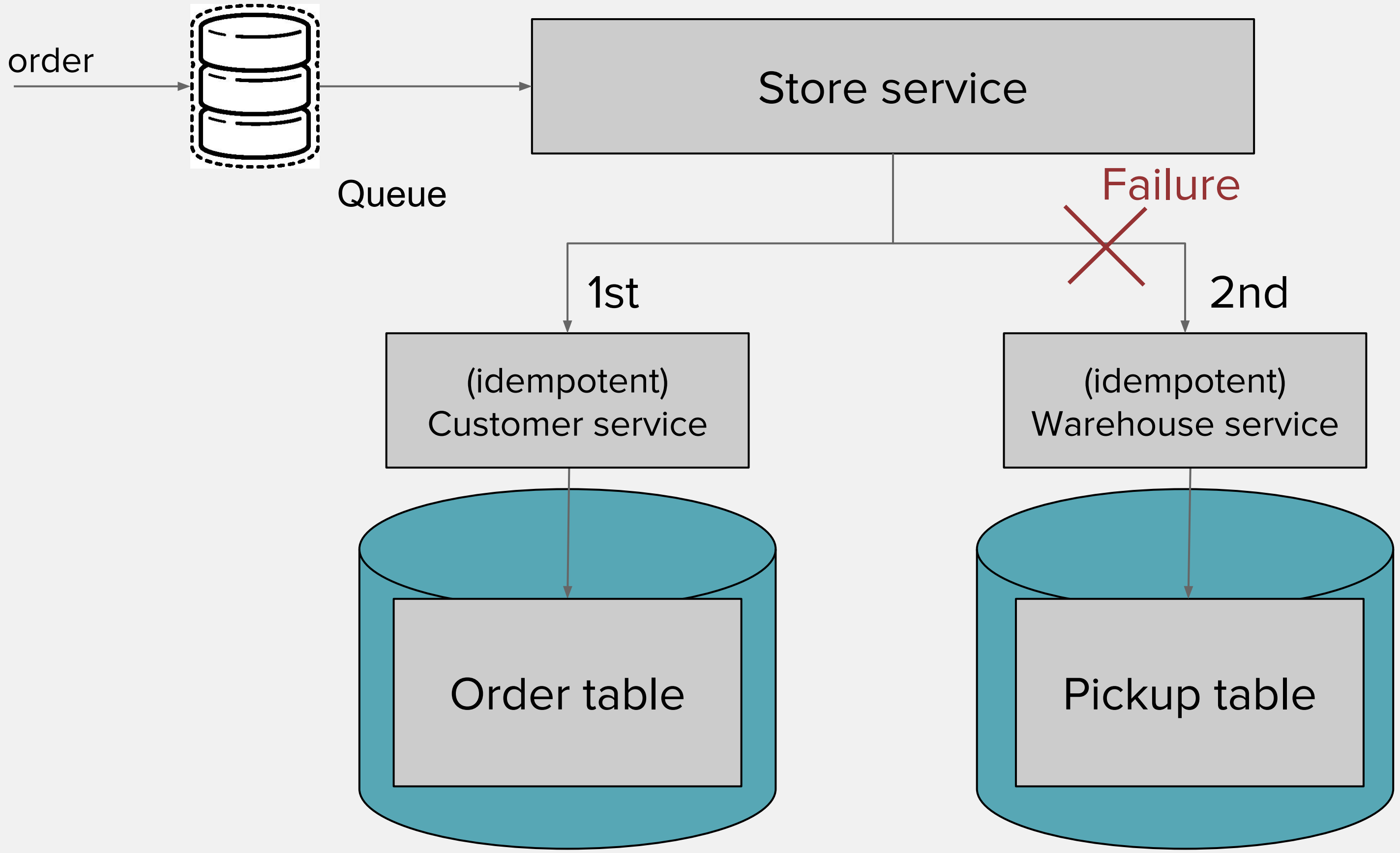
EVENTUALLY CONSISTENT APPROACH

Compensating Transactions



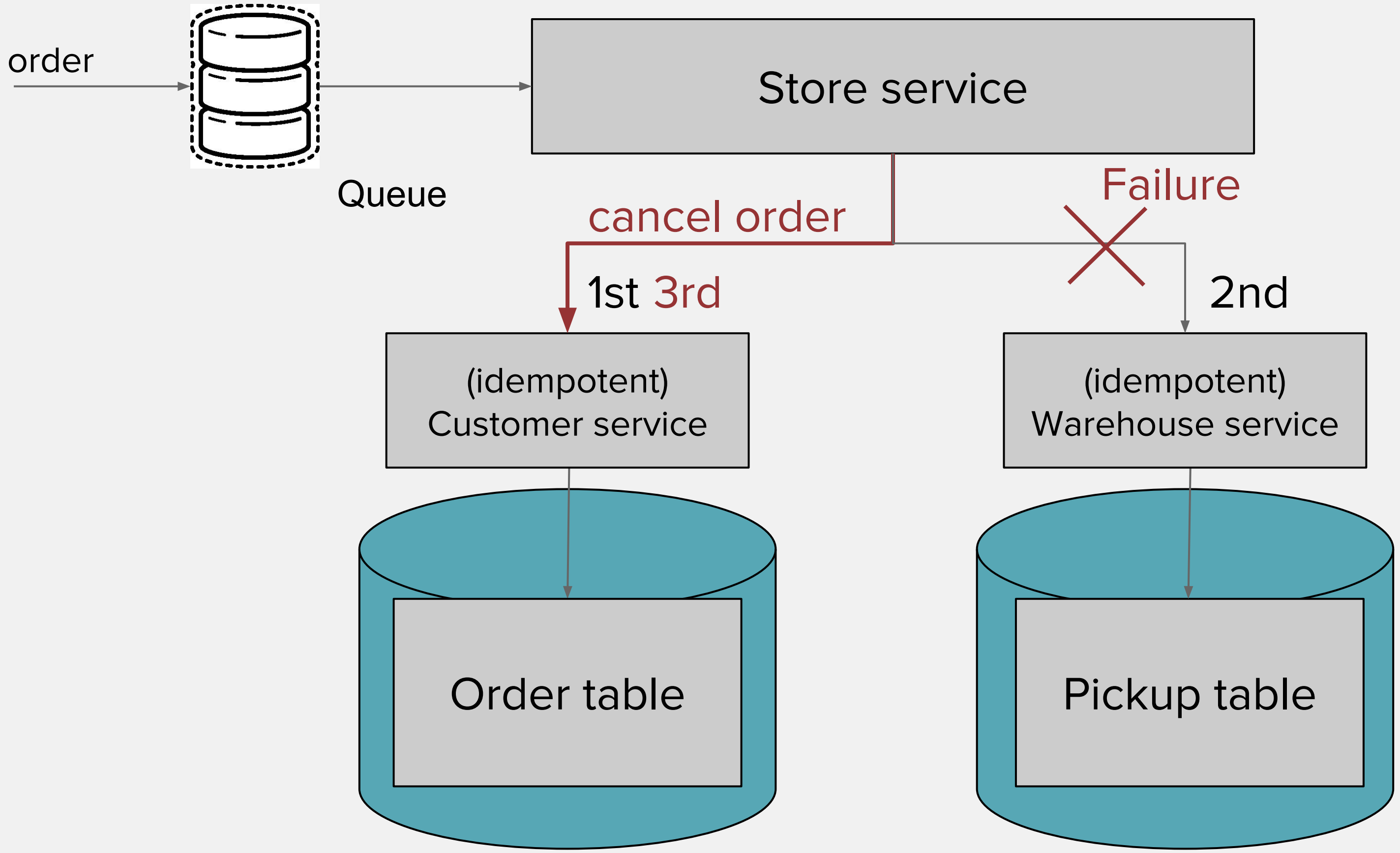
COMPENSATING TRANSACTIONS

Compensating Transactions

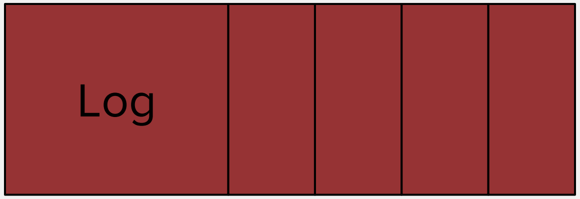


COMPENSATING TRANSACTIONS

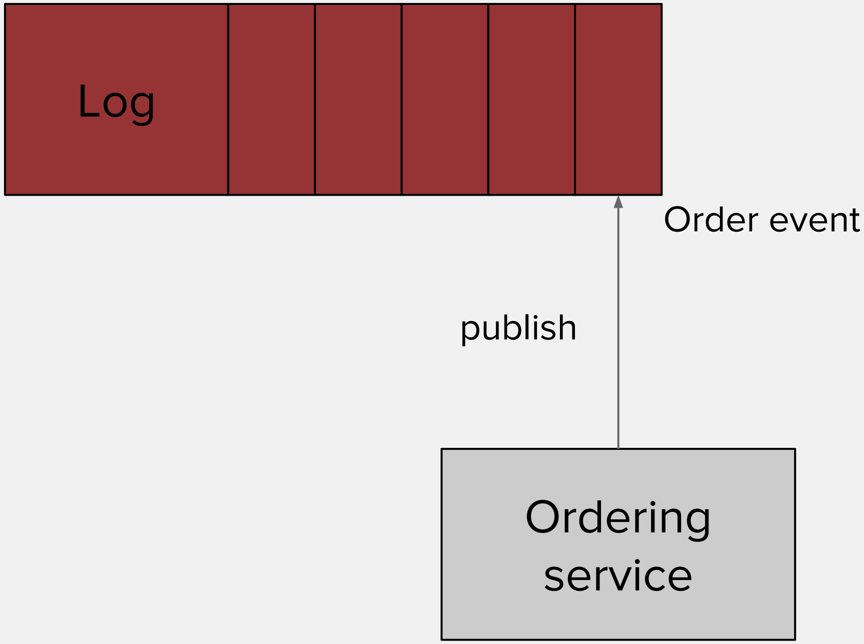
Compensating Transactions



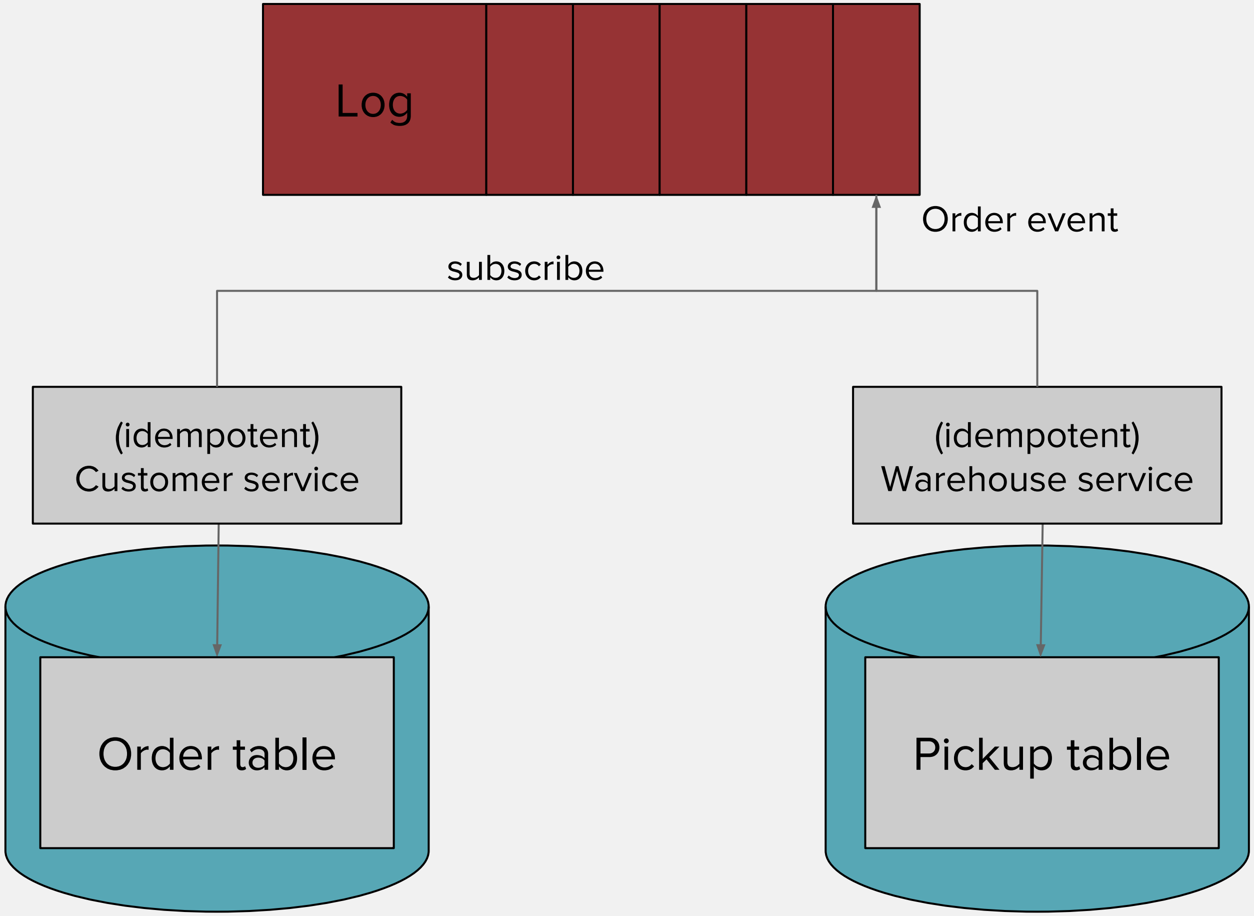
EVENT SOURCING



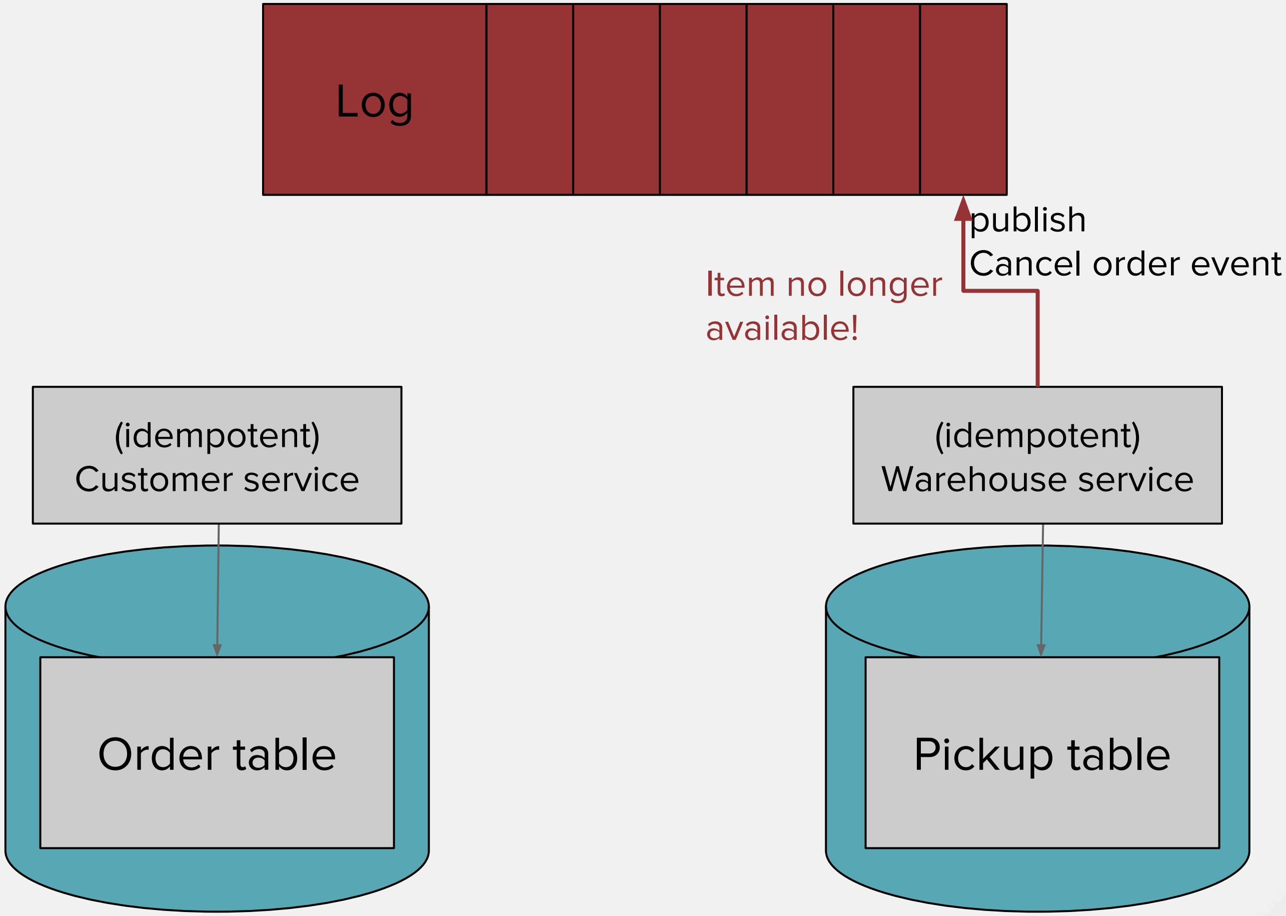
EVENT SOURCING



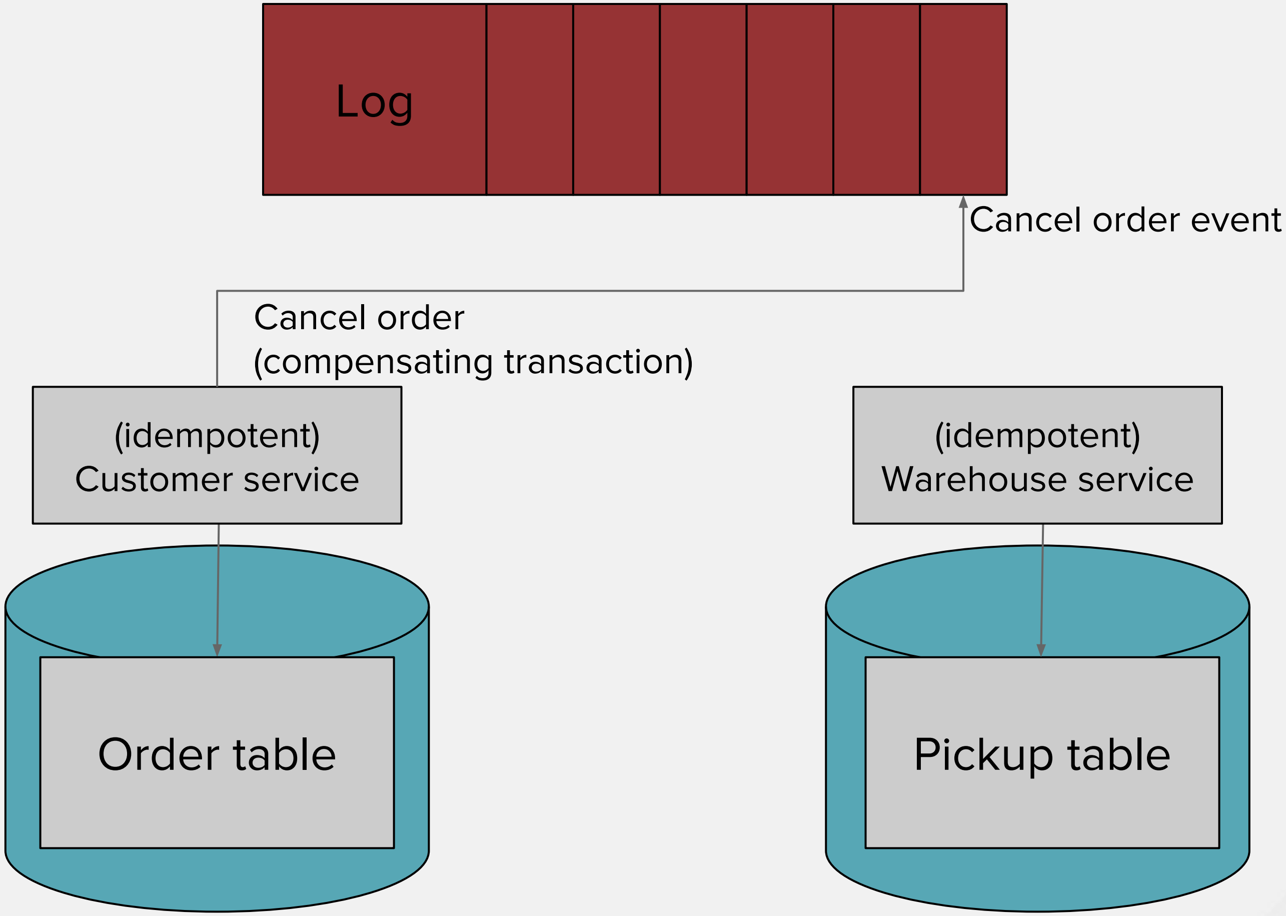
EVENT SOURCING



EVENT SOURCING



EVENT SOURCING



EVENT SOURCING

One Log to Rule them All

- Log is the source of truth
    - audit trail
- CQRS
- Debugging
- Alternate histories

- Drawbacks
    - external systems
    - identifiers
    - event schema

Summary

MICROSERVICE BENEFITS

INDEPENDENT COMPONENT SCALING

CONTINUOUS AND DECOUPLED DEPLOYMENTS

SMALL AND AGILE DEV TEAMS

MICROSERVICE TRADE-OFFS

DISTRIBUTED SYSTEM

EVENTUAL CONSISTENCY

OPERATIONAL COMPLEXITY

MICROSERVICE TRADE-OFFS

DISTRIBUTED SYSTEM

Design for failure

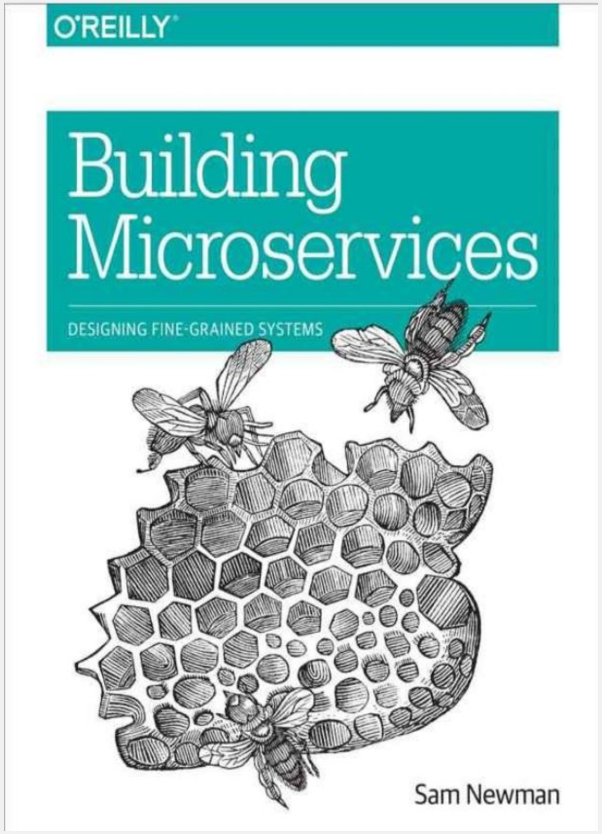
EVENTUAL CONSISTENCY

Carefully consider consistency requirements

OPERATIONAL COMPLEXITY

Embrace immutable infrastructure, automate

Further reading

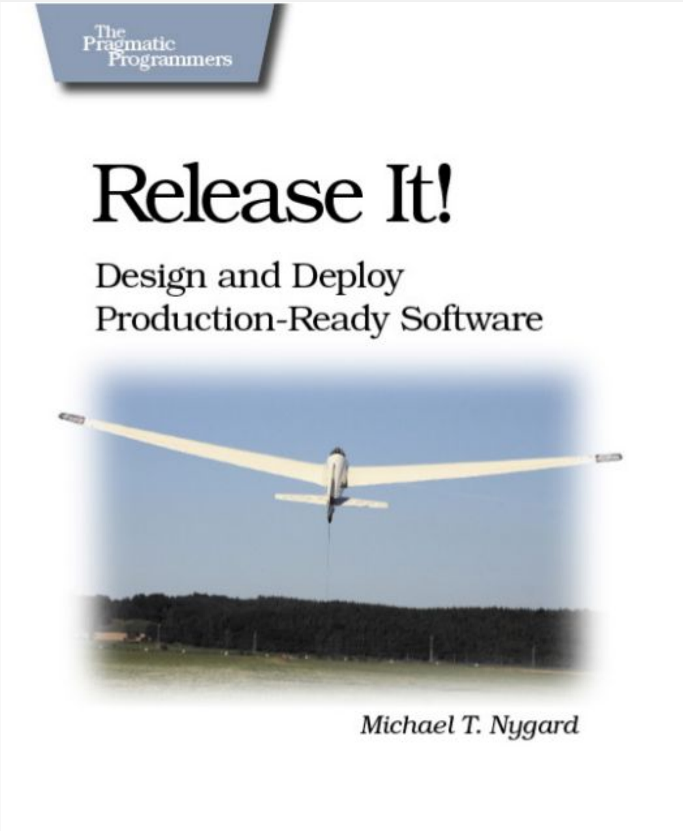


O'REILLY

Building Microservices

DESIGNING FINE-GRAINED SYSTEMS

Sam Newman



The Pragmatic Programmers

Release It!

Design and Deploy Production-Ready Software

Michael T. Nygard

redhat.

THANK YOU

- plus.google.com/+RedHat
- facebook.com/redhatinc
- linkedin.com/company/red-hat
- twitter.com/RedHatNews
- youtube.com/user/RedHatVideos