Mobimeo

Changing the way cities move

Microservices pitfalls

Addressing the most frequent pitfalls when transitioning to Microservices

Mobimeo – Changing the way cities move

Easy access to daily mobility

Our technology empowers mobility providers to orchestrate existing and new modes of public transport.

Together we create an effortless transport experience to make mobility service attractive to millions of users.

More mobility. Less traffic.

We know what drives the mobility sector - today and tomorrow

Founded in Founded in 2018 as subsidiary company of Deutsche Bahn AG and merged with parts of moovel Group GmbH in 2020

Offices in Berlin and Hamburg

170 Mobimeos from over 39 nations

Magnus Kulke

Engineering Manager

github.com/mkulke

lnkd.in/magnuskulke

Lothar Schulz

Engineering Manager

lotharschulz.info

github.com/lotharschulz

lnkd.in/lotharschulz

Contracts

Lawyer up!
Ambiguities and Unmet Expectations

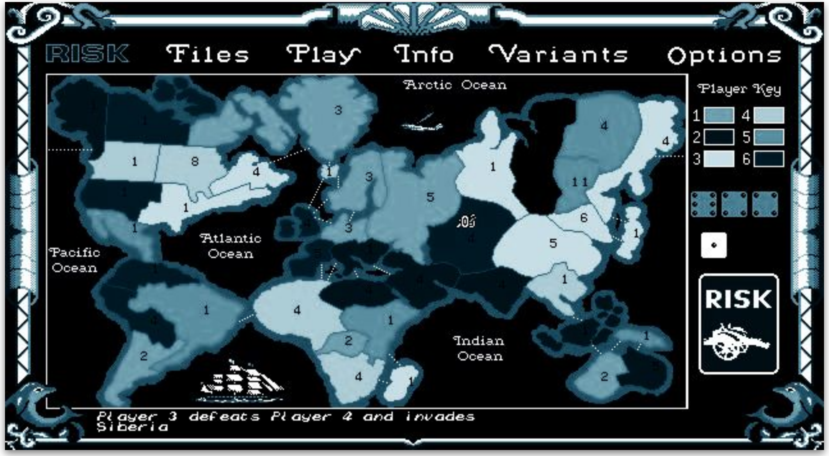
Microservices are (also/primarily?) a social tool

- There is a relation between architecture and team setup

- **“Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization’s communication structure.”**

  Conway’s Law

- Enables teams to make autonomous decisions



Player 3 defeats Player 4 and invades Siberia

Service Boundaries are Defined by Contracts

- Codify expectations towards an API from the consumer’s perspective
  - Behaviour: does not change unexpectedly
  - Availability: when can we retire an API?

- How to express such a contract?
  - Machine readable: Swagger/OpenAPI, JSON Schema, GraphQL
  - API Versions

- Abstain from breaking changes
  - Additional properties?
  - Extending enums?

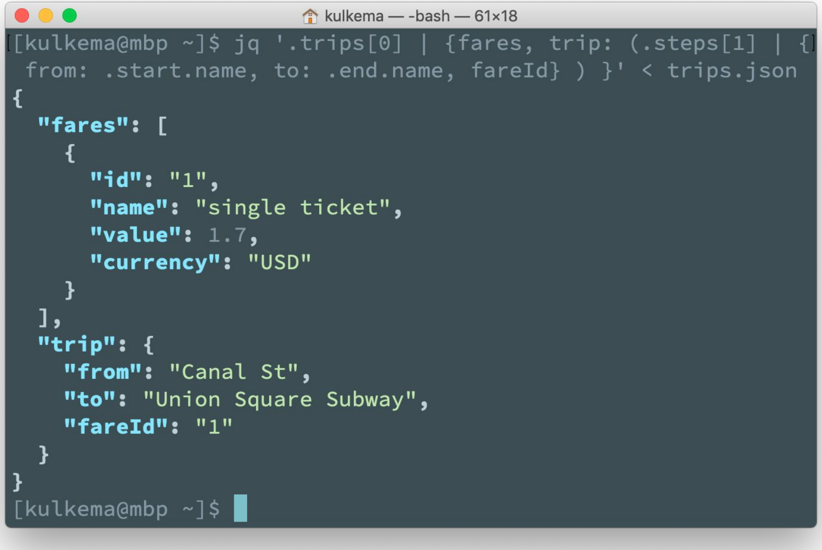
- Make everything optional: Protobuf3

Problem: A Schema might not be expressive enough

- Documents can be formally correct

- But semantics have changed
  - References in a document
  - Content: New ID for entity

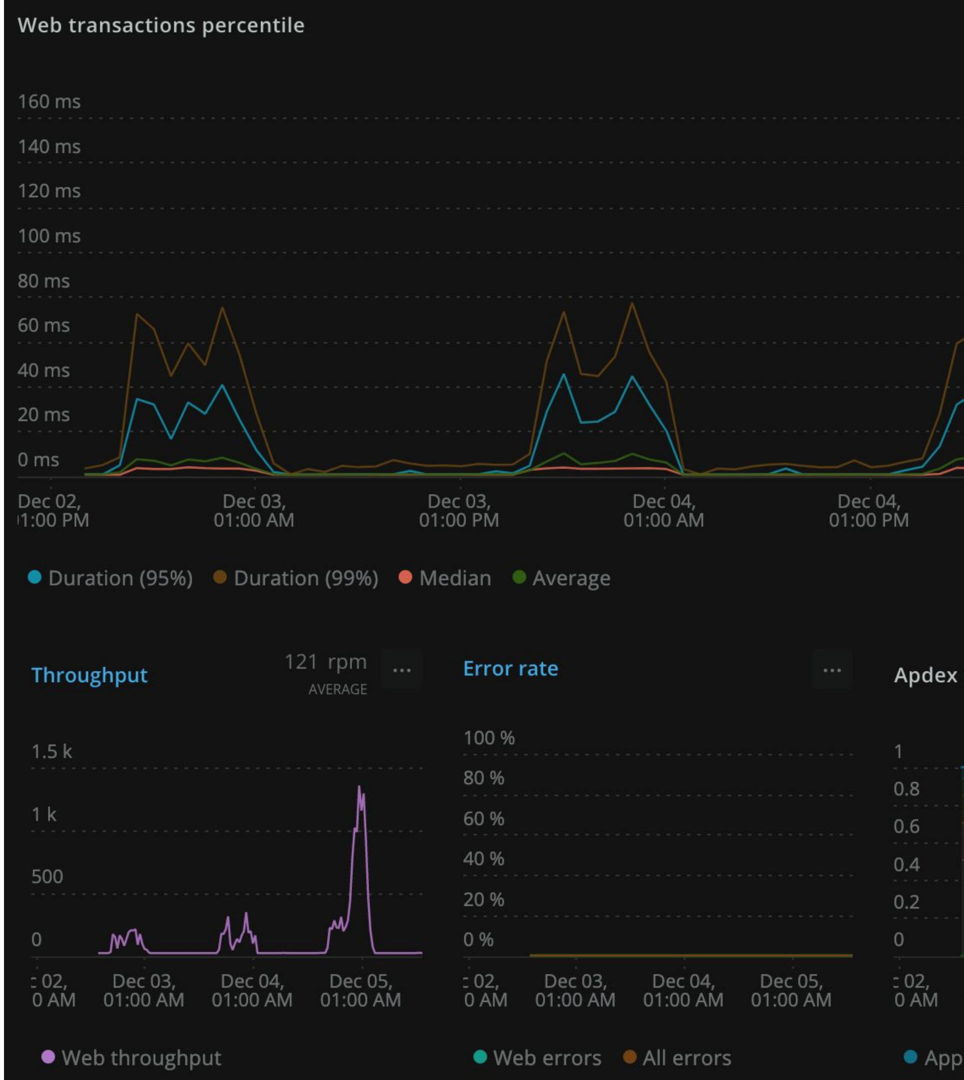
- Pragmatic solution: Contract tests



```
[kulkema@mbp ~]$ jq '.trips[0] | {fares, trip: (.steps[1] | {
 from: .start.name, to: .end.name, fareId} ) }' < trips.json
{
  "fares": [
    {
      "id": "1",
      "name": "single ticket",
      "value": 1.7,
      "currency": "USD"
    }
  ],
  "trip": {
    "from": "Canal St",
    "to": "Union Square Subway",
    "fareId": "1"
  }
}
[kulkema@mbp ~]$
```

Performance Characteristics

- Service level objectives
- Rate limits
- Request budgets



The Other Side: Protection from Harmful Workloads

- Unforeseen (ab)use patterns
- How to attribute incoming traffic?
    - Correlation Ids
    - Callers need to tag their requests
- Manage access
    - Service Accounts
    - Declarative: Service Mesh

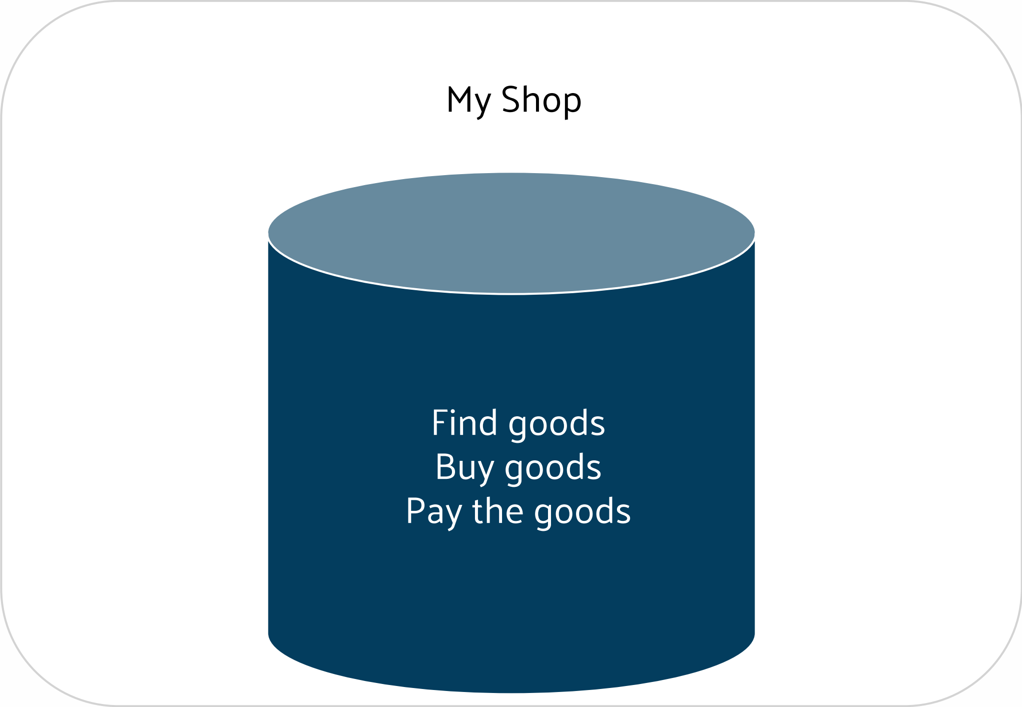
Domains

None of your concern!

Slicing microservices properly

Database as Microservice

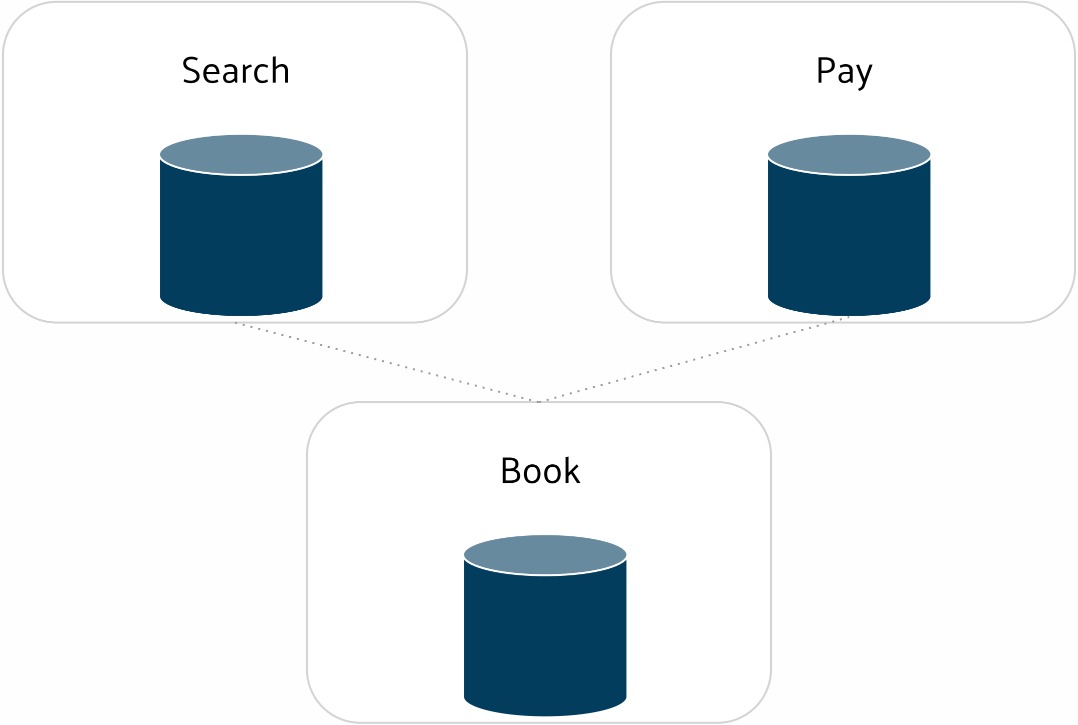
Monolith



My Shop

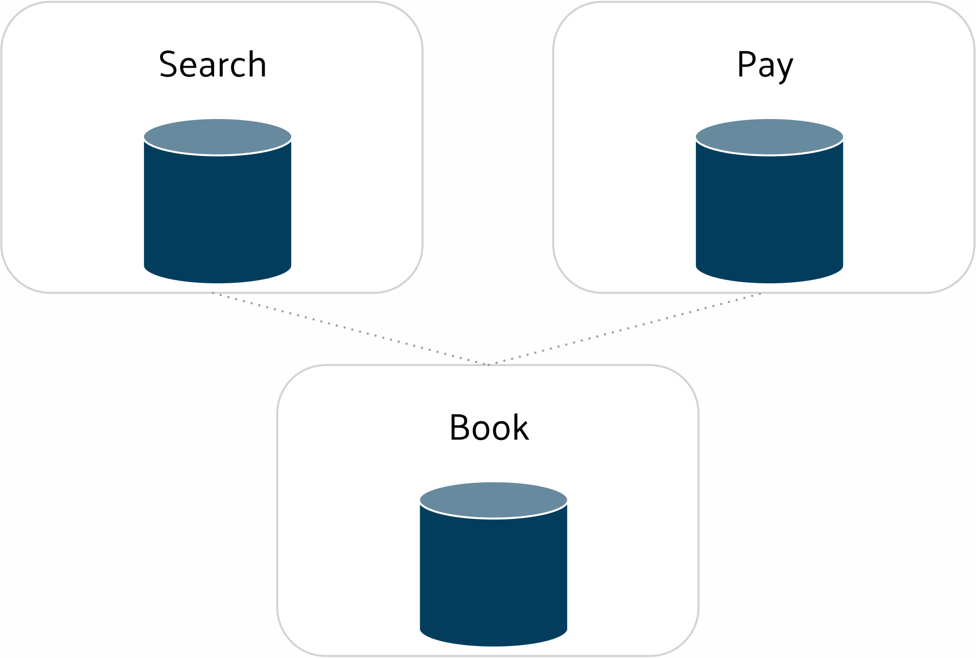
Find goods
Buy goods
Pay the goods

Domains



Domains

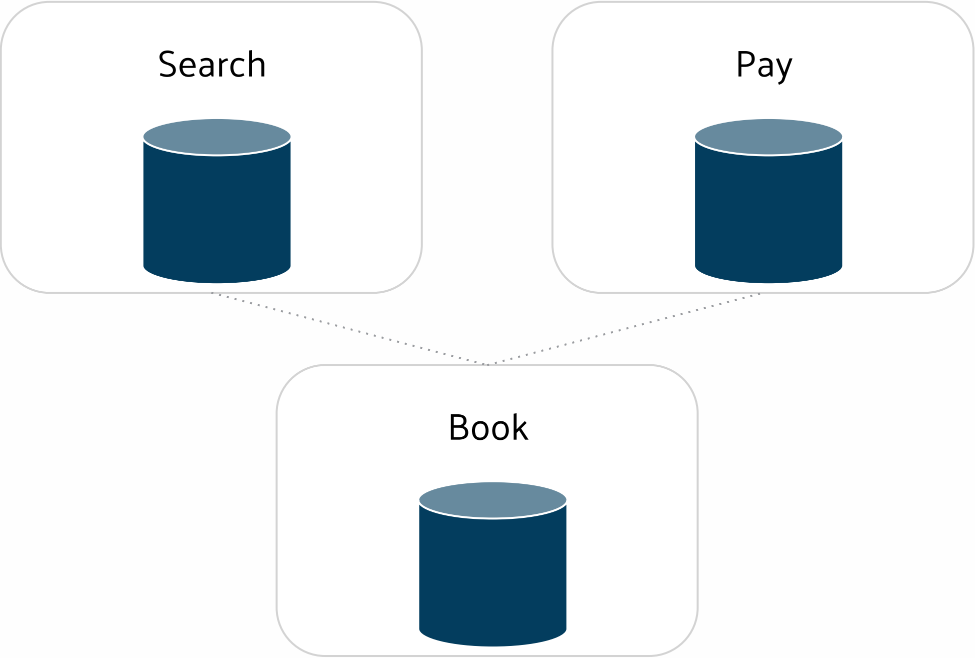
Scaling



Domains

Scaling

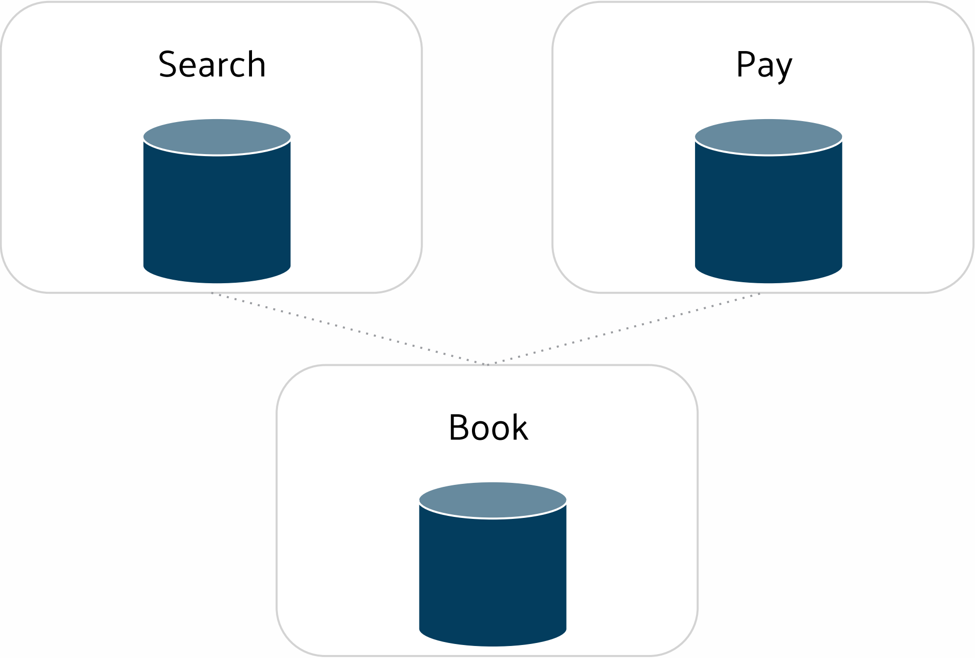
- Vertical



Domains

Scaling

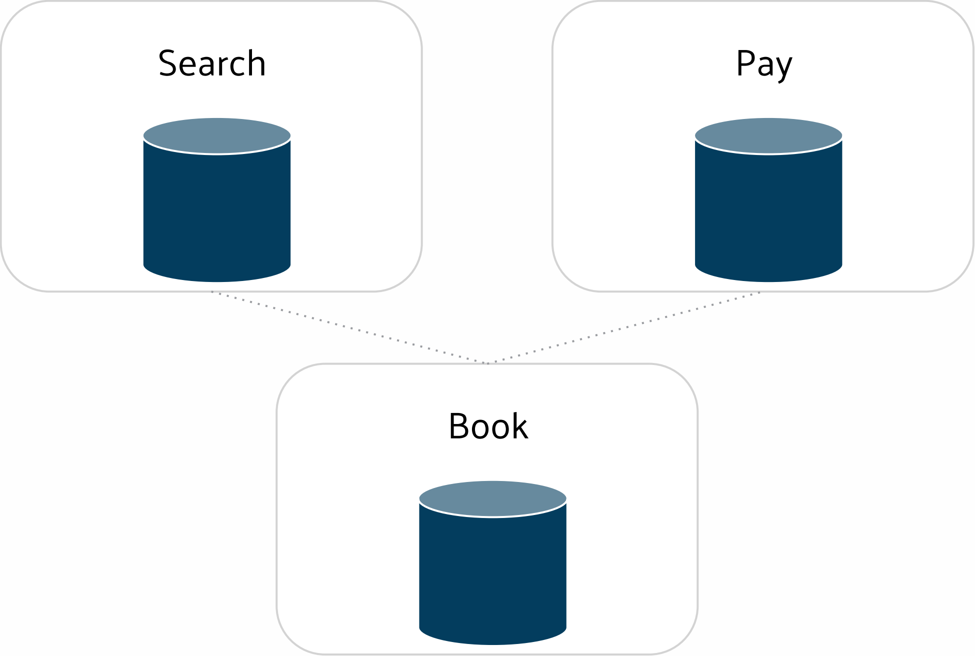
- Vertical
- Horizontal



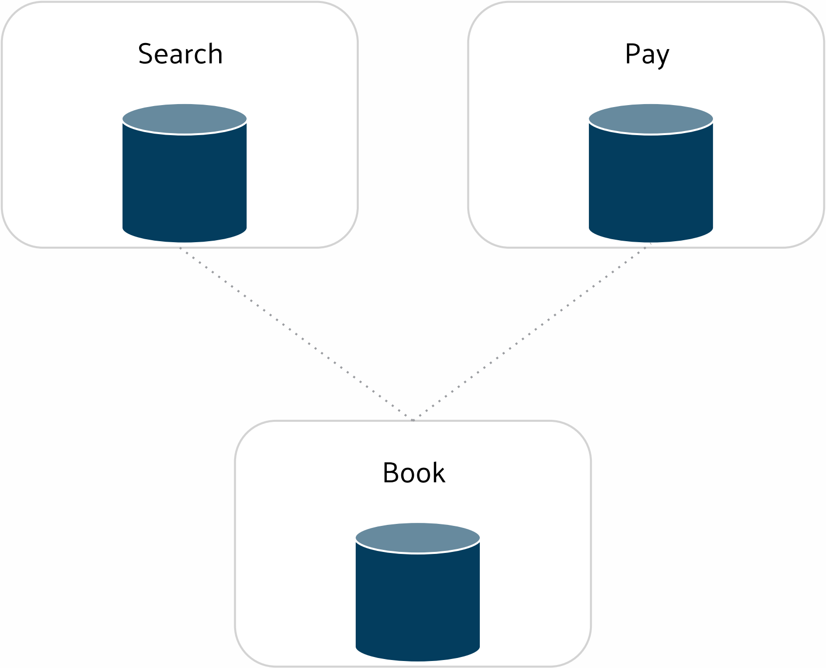
Domains

Scaling

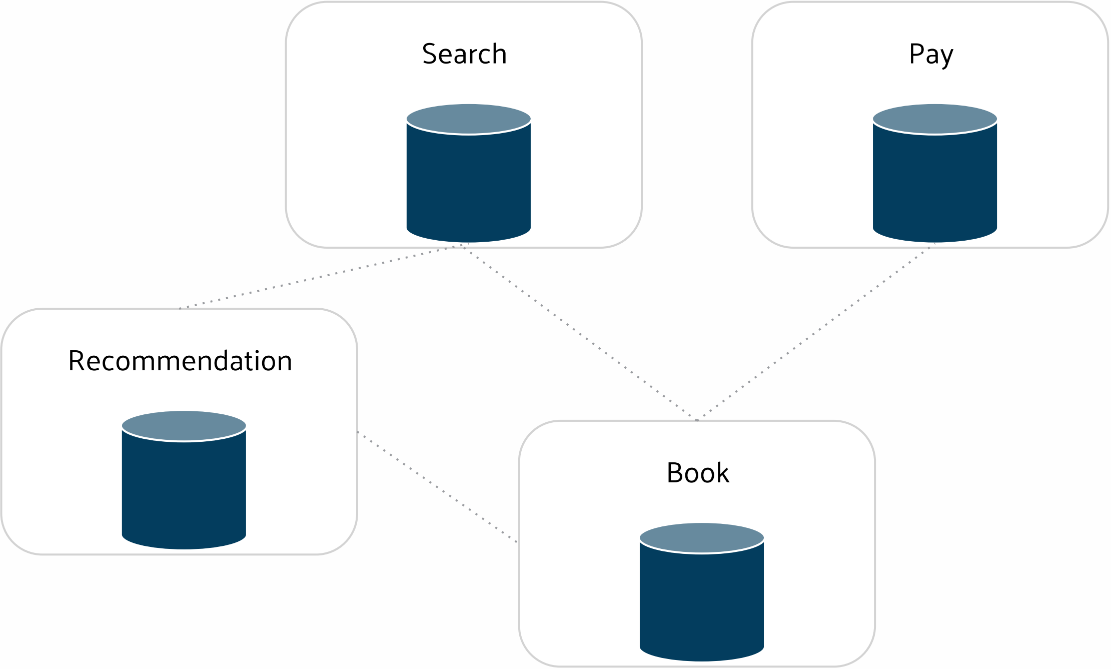
- Vertical
- Horizontal
- Sharding



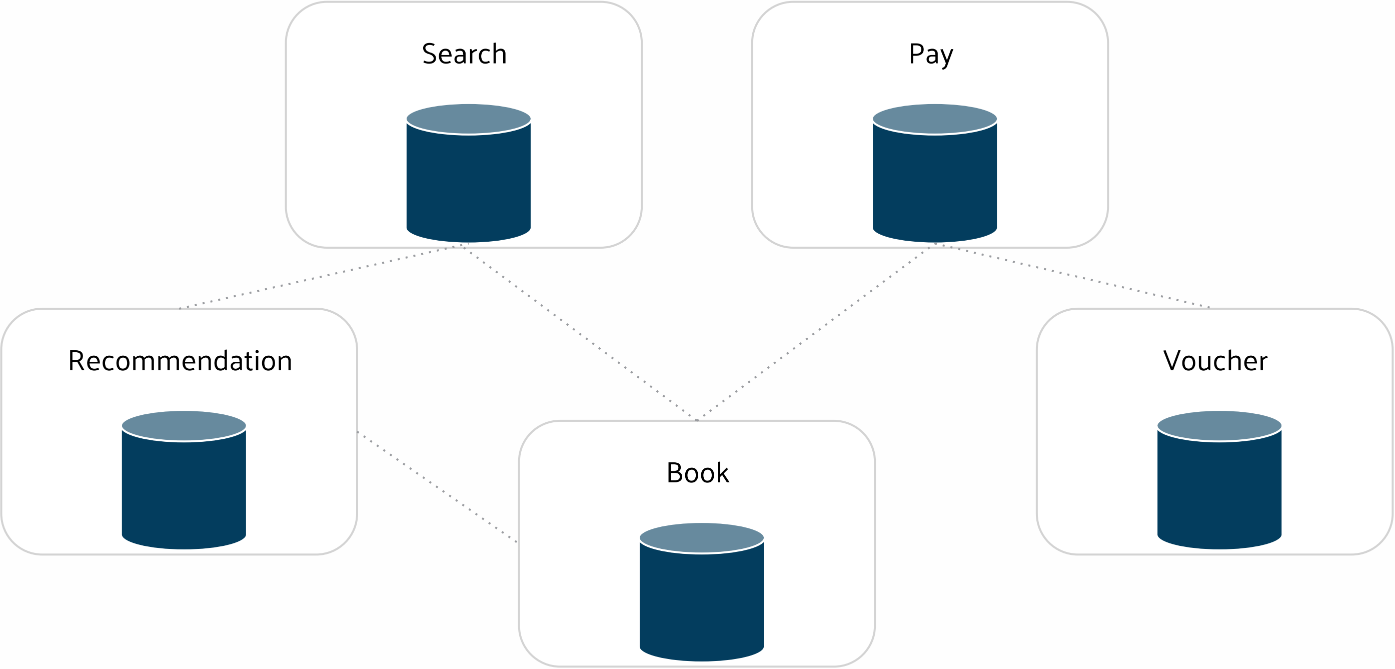
Domains



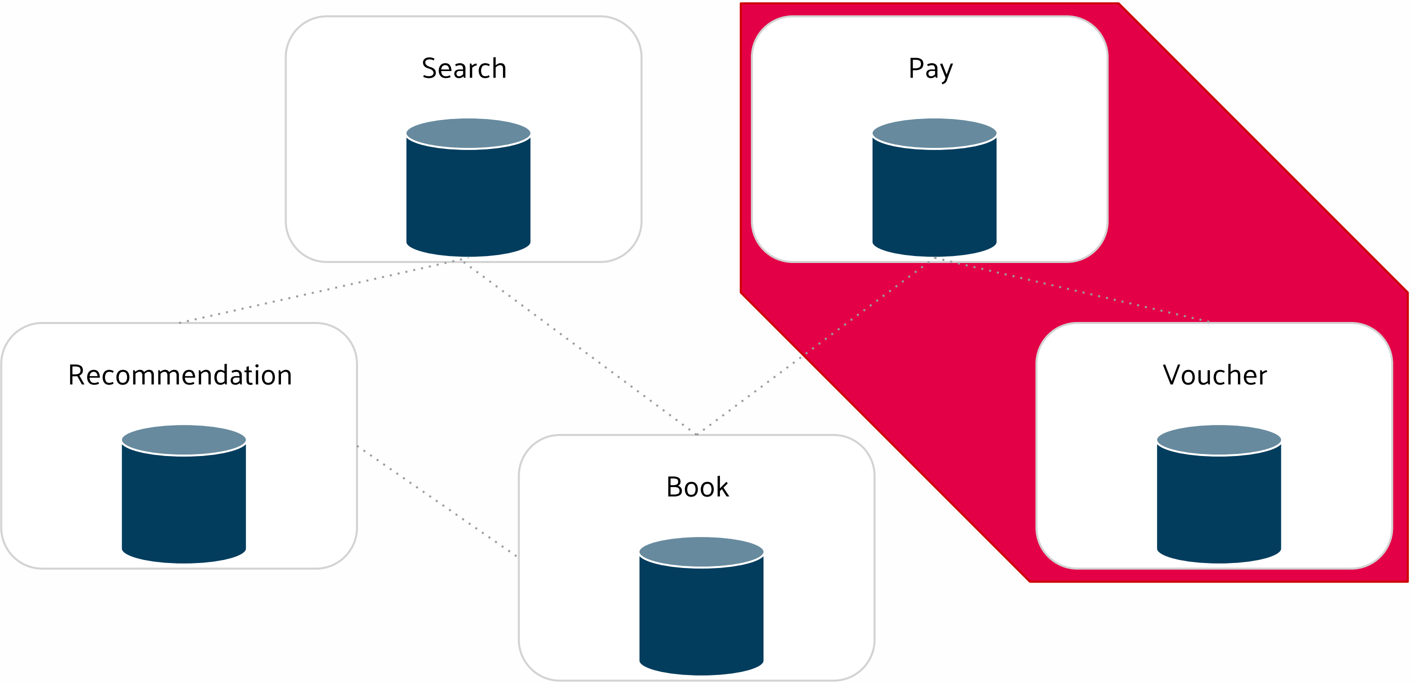
Domains



Domains



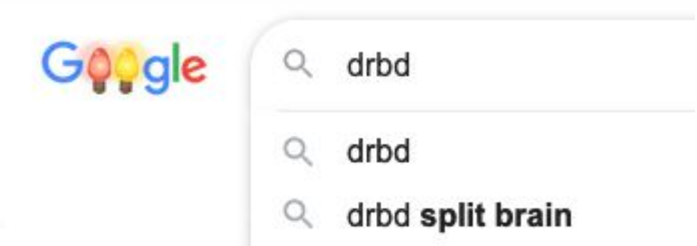
Domains - Bounded Contexts



Distributed Systems

Your Consensus is a House of Cards

Consensus Systems are Great

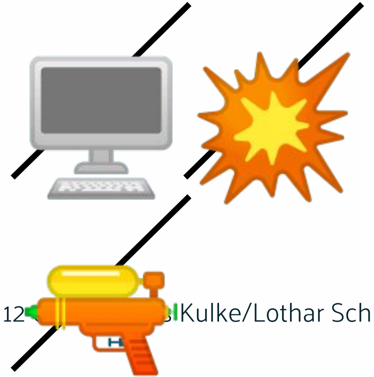

drbd

drbd

drbd **split brain**

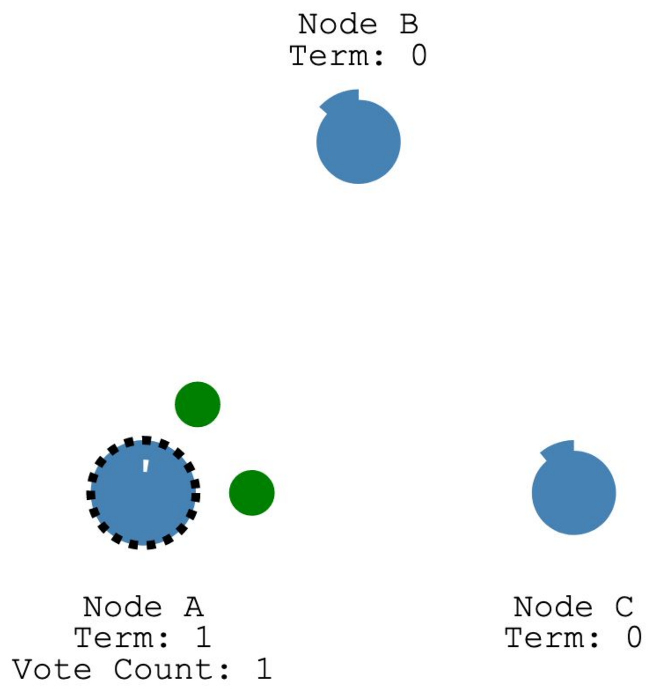
- HA/Clustering prior to consensus systems
  - Heartbeats with serial cable
  - DRBD/GFS
  - STONITH Hardware

- Complex HA machinery was often the cause of outages



Safe Coordination in Distributed systems

- Systems need to agree on a single truth
- Consensus Protocols
- L. Lamport: *The Part-Time Parliament*, 1998
- Simple example: Raft (consul, etcd)



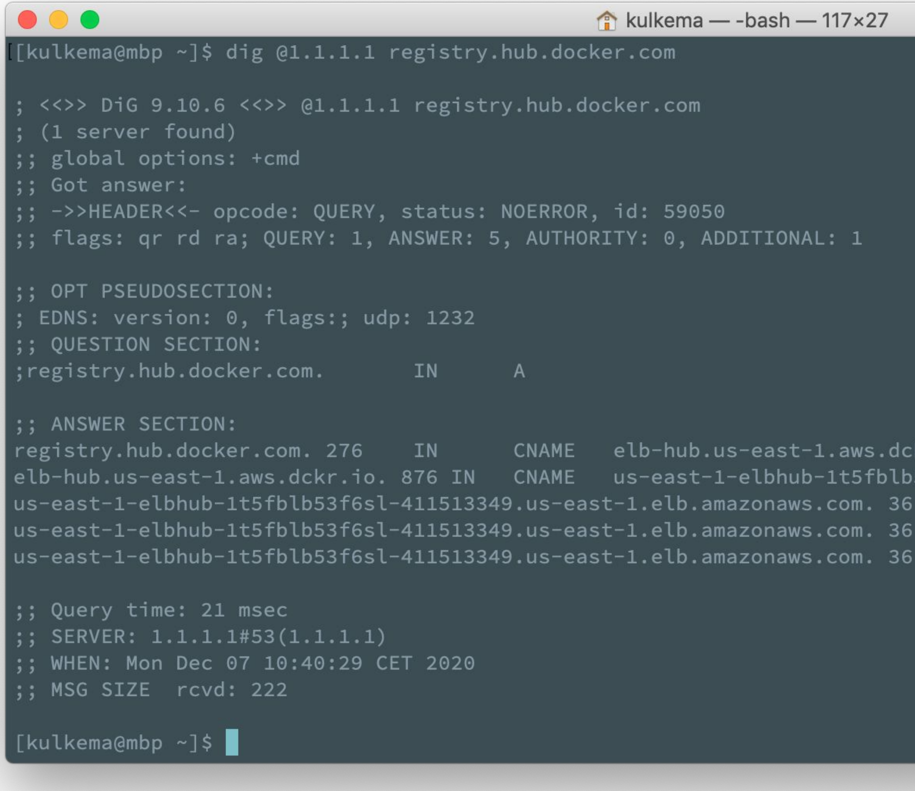
However: Murphy's Law

”Anything that can go wrong will [eventually] go wrong”

We take a lot of things for granted + there are unknown unknowns.

Scenario 1: DockerHub

- Recently introduced rate limits
    - Urgent rollback, 3am
    - Node cannot pull *redis:latest* 🙀
- DNS Load Balancing
- DNS transport is UDP
- UDP Packages are limited in size
- Per Spec DNS allows <= 512 bytes



```
[kulkema@mbp ~]$ dig @1.1.1.1 registry.hub.docker.com

; <<>> DiG 9.10.6 <<>> @1.1.1.1 registry.hub.docker.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59050
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;registry.hub.docker.com.       IN      A

;; ANSWER SECTION:
registry.hub.docker.com. 276    IN      CNAME   elb-hub.us-east-1.aws.dc
elb-hub.us-east-1.aws.dckr.io. 876 IN   CNAME   us-east-1-elbhub-1t5fblb
us-east-1-elbhub-1t5fblb53f6sl-411513349.us-east-1.elb.amazonaws.com. 36
us-east-1-elbhub-1t5fblb53f6sl-411513349.us-east-1.elb.amazonaws.com. 36
us-east-1-elbhub-1t5fblb53f6sl-411513349.us-east-1.elb.amazonaws.com. 36

;; Query time: 21 msec
;; SERVER: 1.1.1.1#53(1.1.1.1)
;; WHEN: Mon Dec 07 10:40:29 CET 2020
;; MSG SIZE  rcvd: 222

[kulkema@mbp ~]$
```

Scenario 1: DockerHub, cont.

- DNS responses > 512 bytes fall back to TCP
    - Your sysadmin might not know this
    - Security Group blocks tcp/53

- Not all resolvers are alike / agree on the spec
    - Glibc “salvages” truncated DNS messages
    - Golang DNS resolver (Docker) does not
    - Quick fix: `CGO_ENABLED=1`

Scenario 2: DNS, again (it’s always DNS)

- Our J2EE service is stuck in an exception loop
  - Logs a lot of large stack traces (lots of lines)
- Engineers integrate cool .io SaaS for tailing logs in Logstash
  - Every line a request to cool .io data sink
  - Every line a hostname is resolved
- Cloud Providers disapproves, starts rate-limiting DNS the service’s node
- K8S api-server/node comm. is affected.
  - Node is marked as broken
  - Scheduler moved ever-crashing service to fresh, healthy node
- Repeat

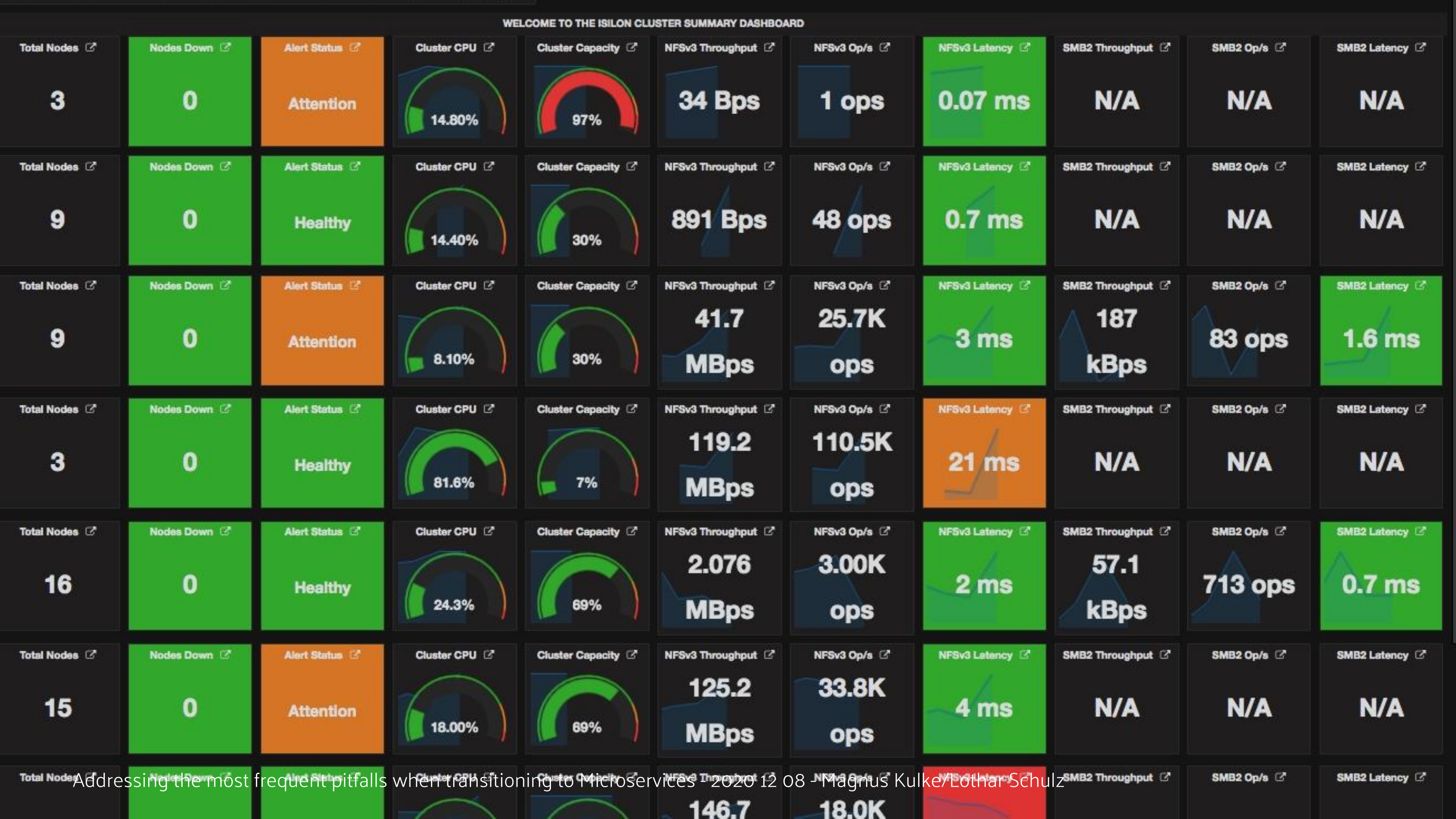
Scenario 3: Seemingly unlimited resources

- Nov. 25th Kinesis outage
  - every node connects with every other node
  - After scaling exceeded threads-max

- File Handles
  - Some workloads do not properly close TCP/IP connections
  - Intermediate proxies have to arbitrarily terminate
  - (Old) user-land kube-proxy leaked goroutines & file handles

Observability

How to X-Ray a hairball



WELCOME TO THE ISILON CLUSTER SUMMARY DASHBOARD

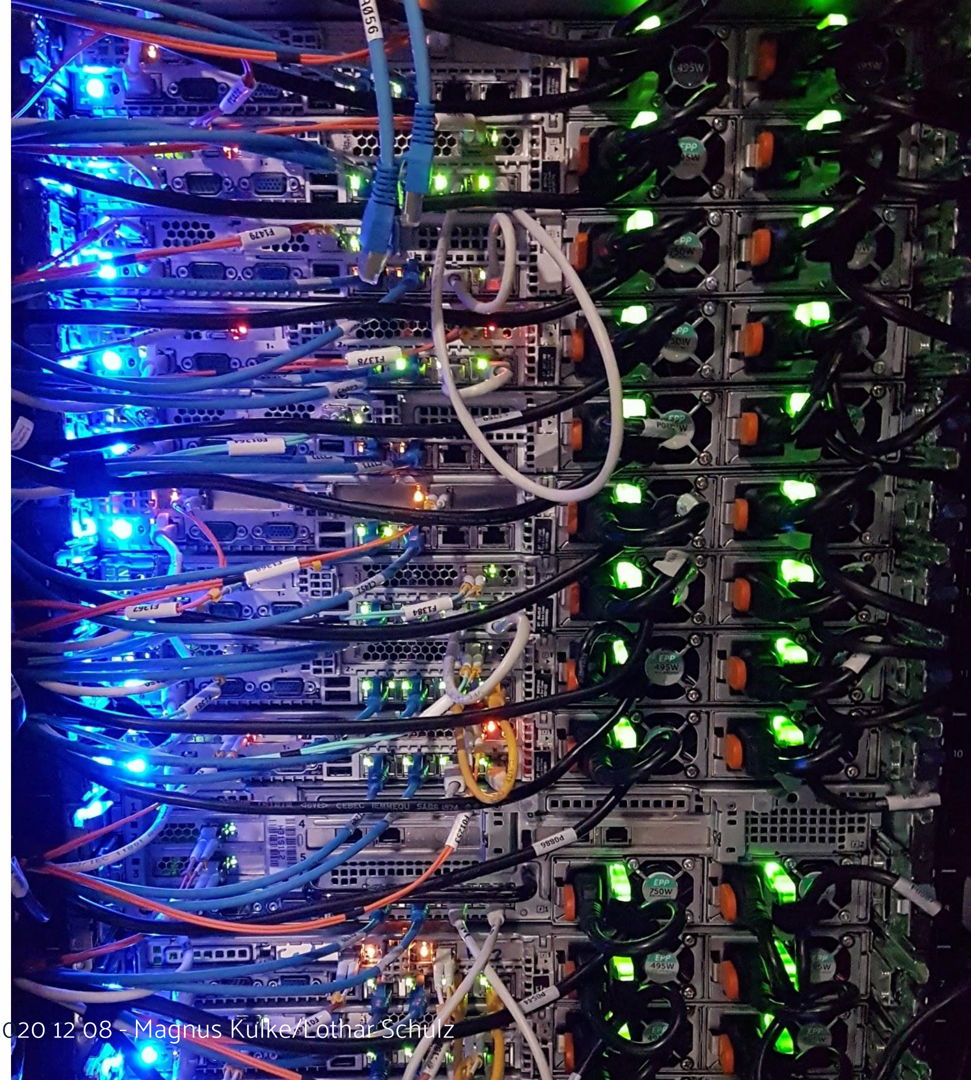
Total Nodes	Nodes Down	Alert Status	Cluster CPU	Cluster Capacity	NFSv3 Throughput	NFSv3 Op/s	NFSv3 Latency	SMB2 Throughput	SMB2 Op/s	SMB2 Latency
3	0	Attention	14.80%	97%	34 Bps	1 ops	0.07 ms	N/A	N/A	N/A
9	0	Healthy	14.40%	30%	891 Bps	48 ops	0.7 ms	N/A	N/A	N/A
9	0	Attention	8.10%	30%	41.7 MBps	25.7K ops	3 ms	187 kBps	83 ops	1.6 ms
3	0	Healthy	81.6%	7%	119.2 MBps	110.5K ops	21 ms	N/A	N/A	N/A
16	0	Healthy	24.3%	69%	2.076 MBps	3.00K ops	2 ms	57.1 kBps	713 ops	0.7 ms
15	0	Attention	18.00%	69%	125.2 MBps	33.8K ops	4 ms	N/A	N/A	N/A
					146.7	18.0K				

Tailor towards audience

Example:

- 24x7
- the engineering teams

- Management
- End customers



Service Level Objectives

Intuition, **experience**, and an **understanding** of what engineers know about the services they serve is used to define

- service level indicators (SLIs),
- objectives (SLOs),
- and agreements (SLAs).

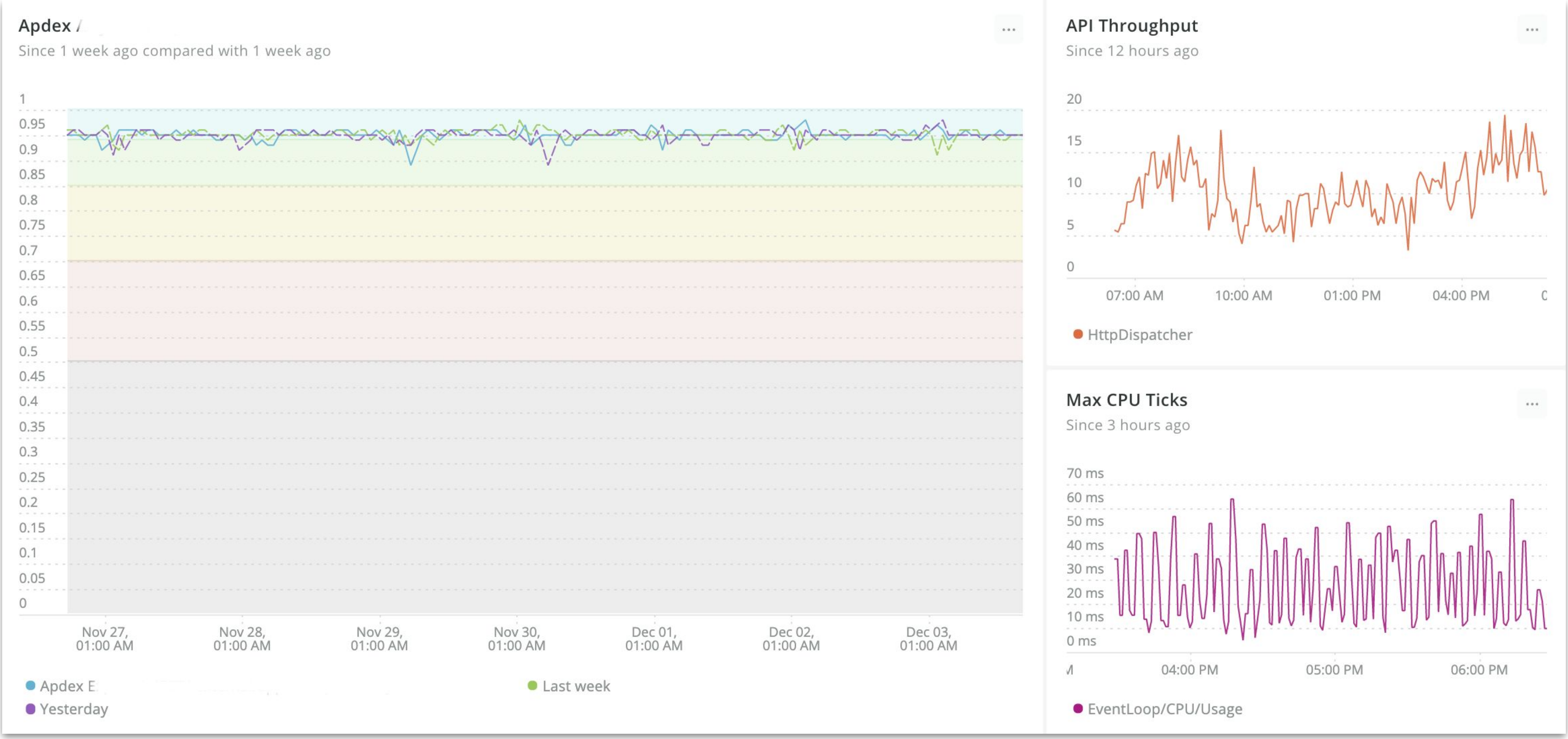
SRE Book - Service Level Objectives

Guidance - The Four Golden Signals

- **request latency** - request response time and/or timeout rate
- **error rate** - proportion of service errors
- **traffic / system throughput** - typically measured in requests per second
- **availability** - what's the uptime of a service
- **saturation** - measures the system fraction, emphasizing the resources that are most constrained (e.g., in a memory-constrained system, show memory; in an I/O-constrained system, show I/O). I experienced system degrading service levels before being saturated, e.g. 90% CPU utilization triggered a service degradation already.

SRE Book - The Four Golden Signals

Results



Apdex /

Since 1 week ago compared with 1 week ago

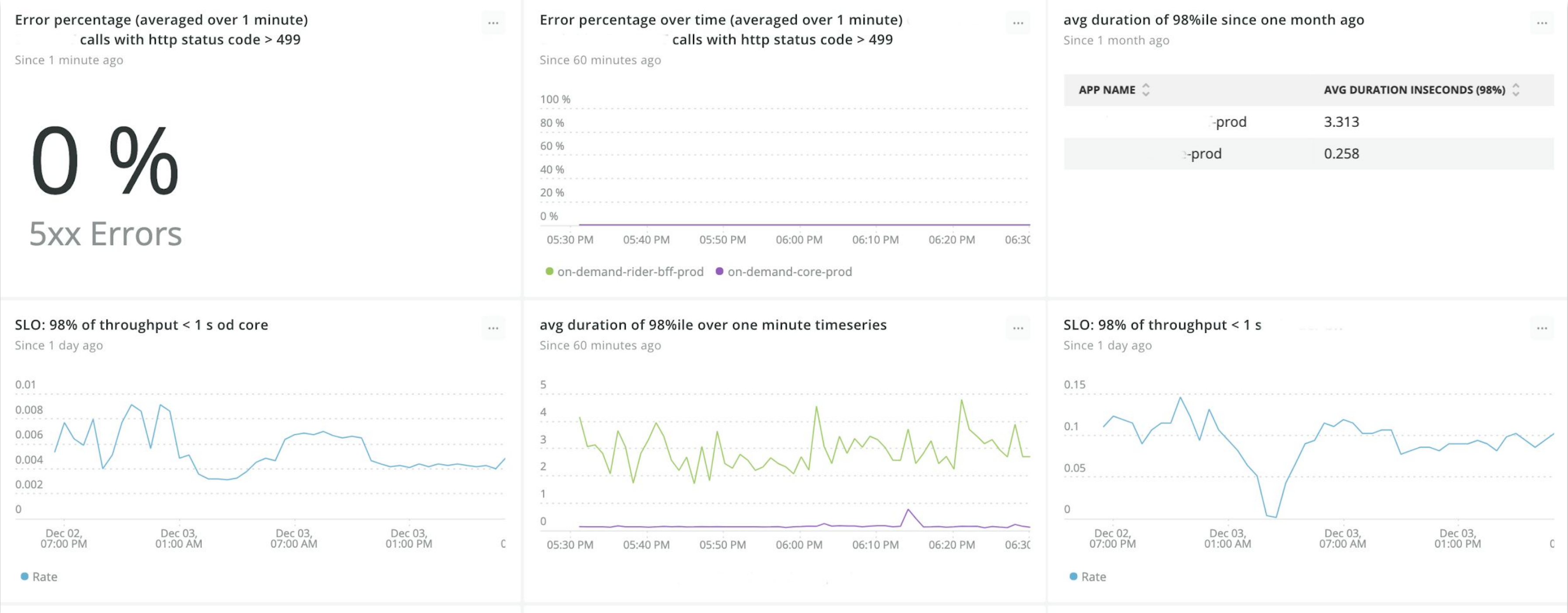
API Throughput

Since 12 hours ago

Max CPU Ticks

Since 3 hours ago

Results



Error percentage (averaged over 1 minute)
calls with http status code > 499
Since 1 minute ago

0 %

5xx Errors

Error percentage over time (averaged over 1 minute)
calls with http status code > 499
Since 60 minutes ago

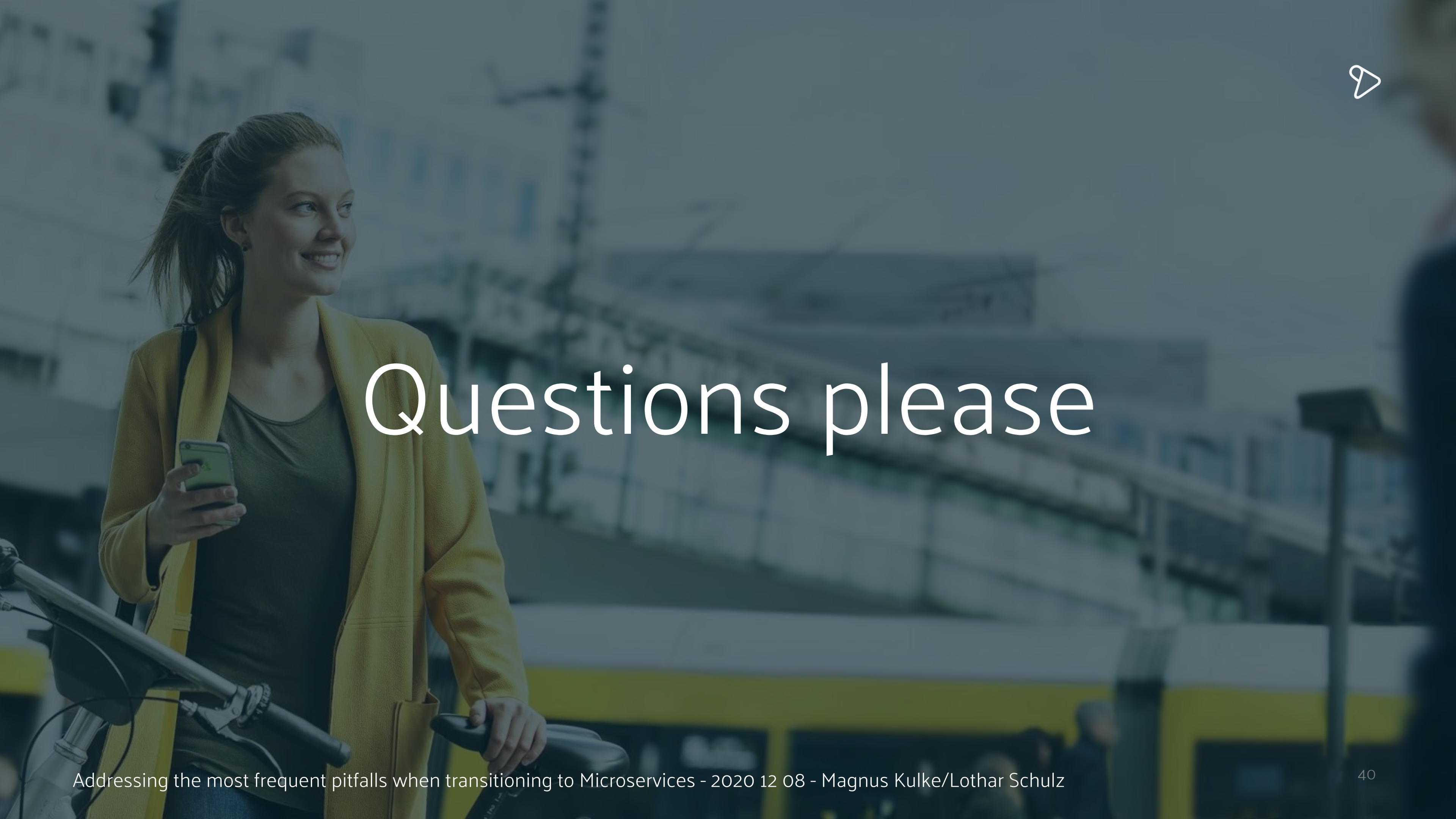
avg duration of 98%ile since one month ago
Since 1 month ago

APP NAME	AVG DURATION INSECONDS (98%)
-prod	3.313
-prod	0.258

SLO: 98% of throughput < 1 s od core
Since 1 day ago

avg duration of 98%ile over one minute timeseries
Since 60 minutes ago

SLO: 98% of throughput < 1 s
Since 1 day ago



Questions please

Mobimeo