zalando

ARE WE ALL ON THE SAME PAGE?

LET'S FIX THAT

Luis Mineiro @voidmaze

SRE @ Zalando

Coding Serbia, 17.05.2019

ZALANDO AT A GLANCE

~ 5.4 billion EUR

revenue 2018

> 300 million visits per month

> 27 million active customers

> 400,000 product choices

> 15,500 employees in Europe

> 80% of visits via mobile devices

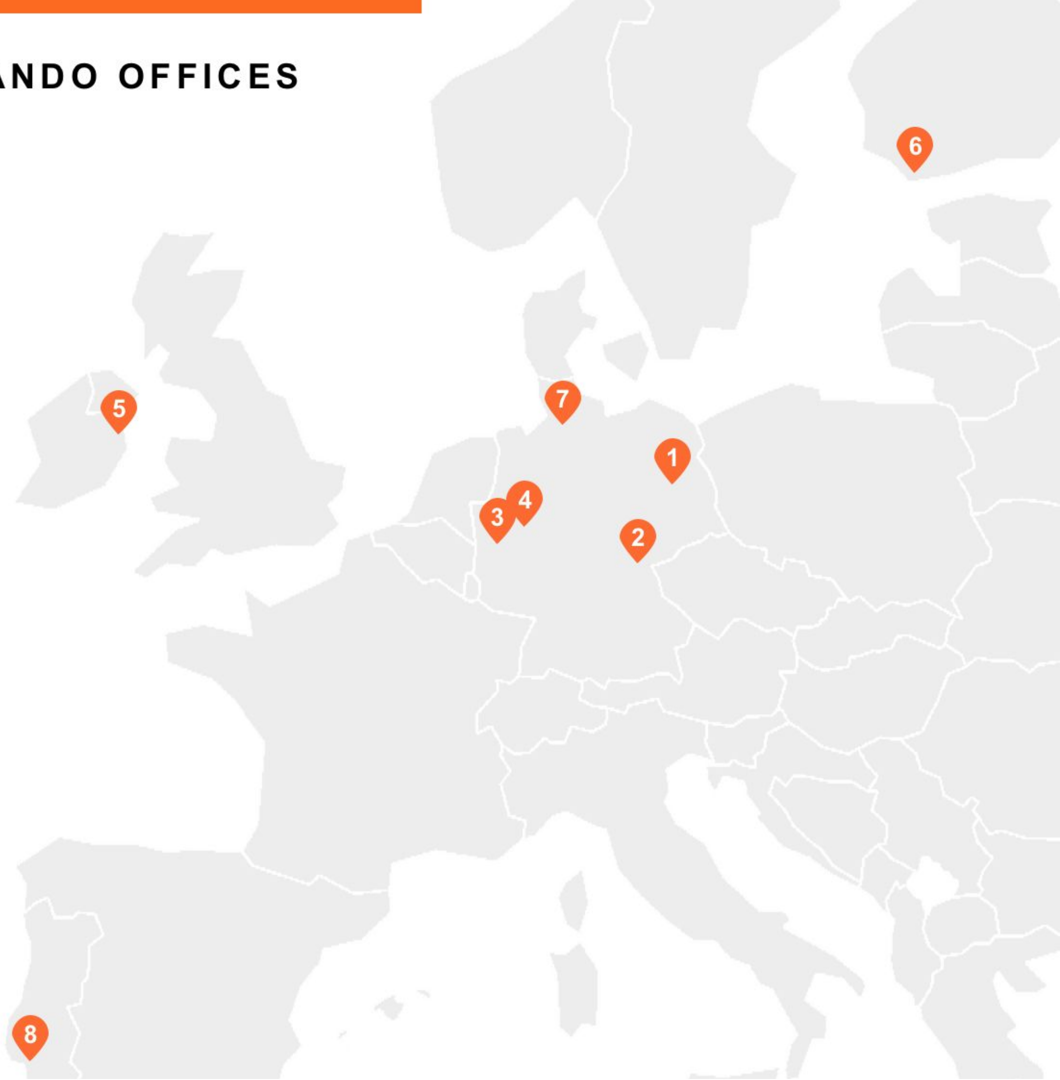
~ 2,000 brands

17 countries

as of March 2019

zalando

ZALANDO OFFICES



1	BERLIN **HEADQUARTERS**
2	ERFURT **TECH OFFICE**
3	MÖNCHENGLADBACH **TECH OFFICE**
4	DORTMUND **TECH HUB**
5	DUBLIN **TECH HUB**
6	HELSINKI **TECH HUB**
7	HAMBURG **ADTECH LAB**
8	LISBON **TECH HUB**

as of March 2019

WE ARE CONSTANTLY INNOVATING TECHNOLOGY

HOME-BREWED, CUTTING-EDGE & SCALABLE
technology solutions

help our brand to
WIN ONLINE

8 international tech locations

HQs
in Berlin

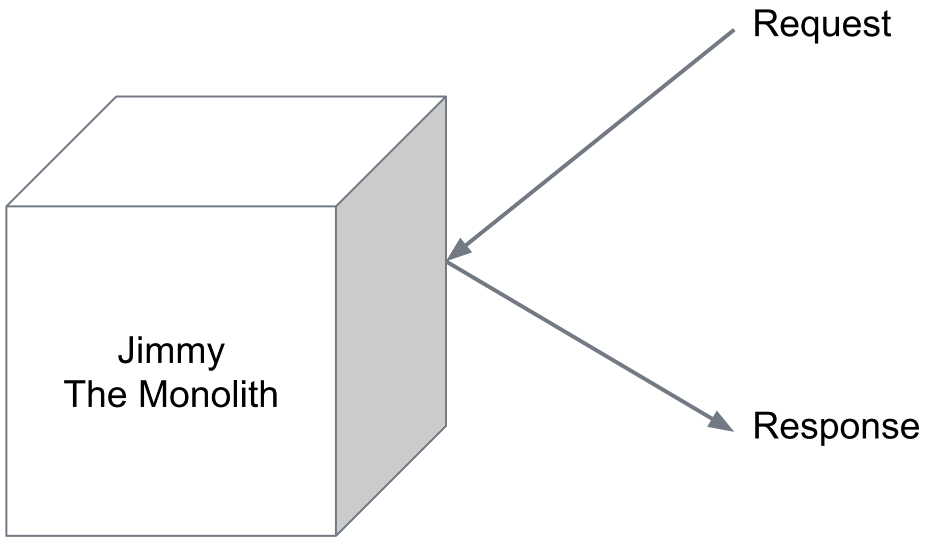
> 2,000
employees at

zalando

Photo by Dawn Armfield on Unsplash

THE AGE OF THE MONOLITH

Single, large boxes
that did everything



MONITORING THE MONOLITH

Ops Monitoring

- Is the box alive?
- Is the monolith process up?

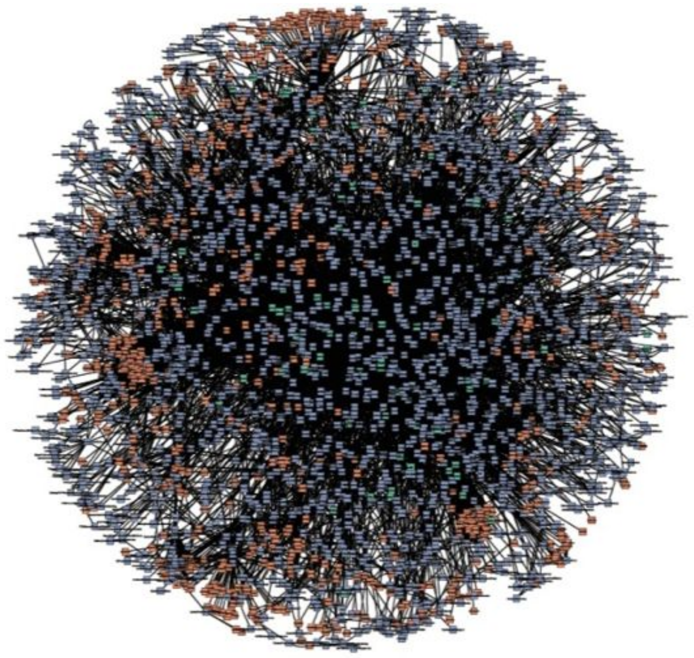
Devs Monitoring

- Are requests returning errors?
- Are requests reasonably fast?

Photo by Deneen LT on Pexels

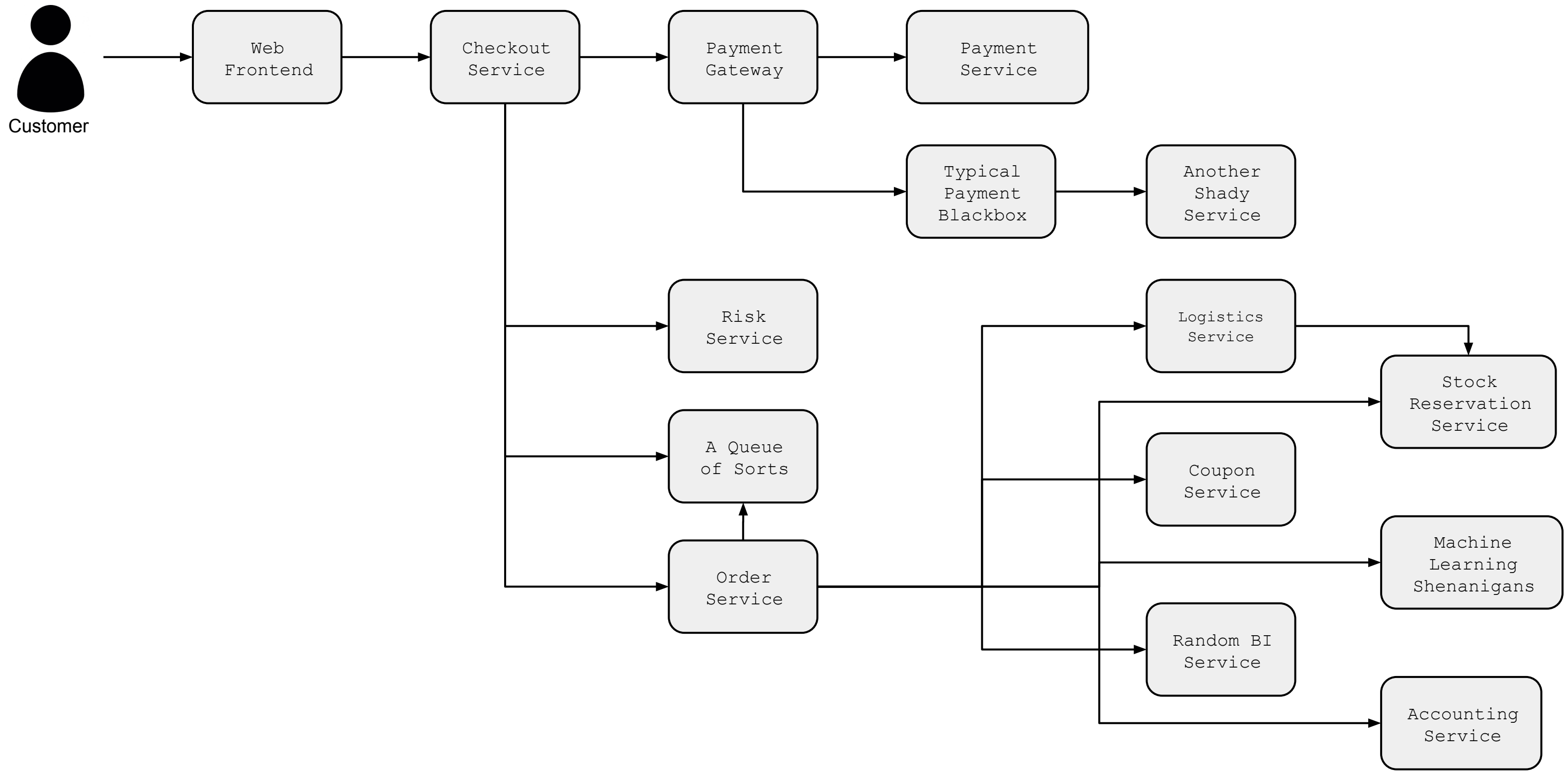
zalando

MODERN MICROSERVICES ARCHITECTURES



Amazon internal service dependency visualization

EXAMPLE - PLACING AN ORDER



MONITORING MICROSERVICES

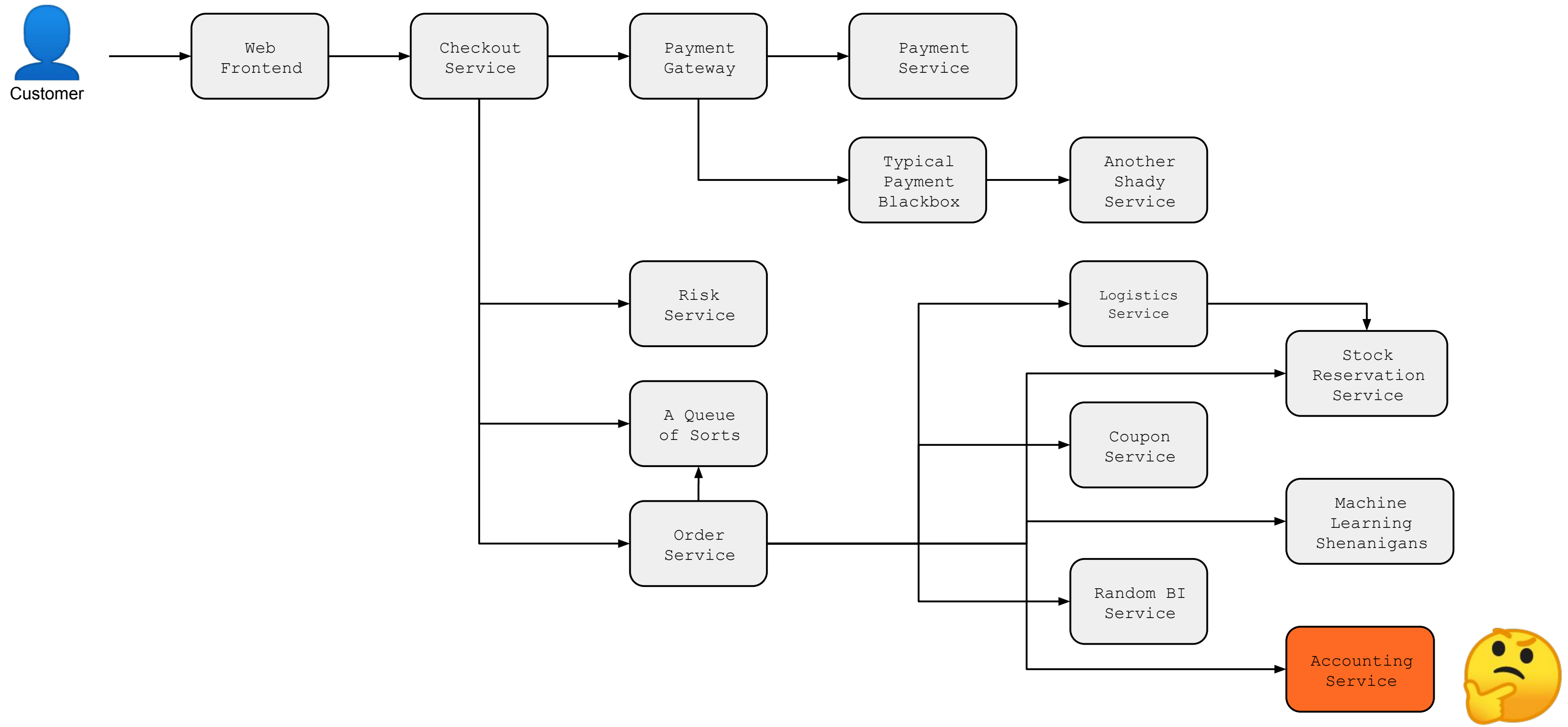
"DevOps" Monitoring

- Is the box alive?
- Is the micro-service process up?
- Are requests returning errors?
- Are requests reasonably fast?

Photo by Antoine Plüss on Unsplash

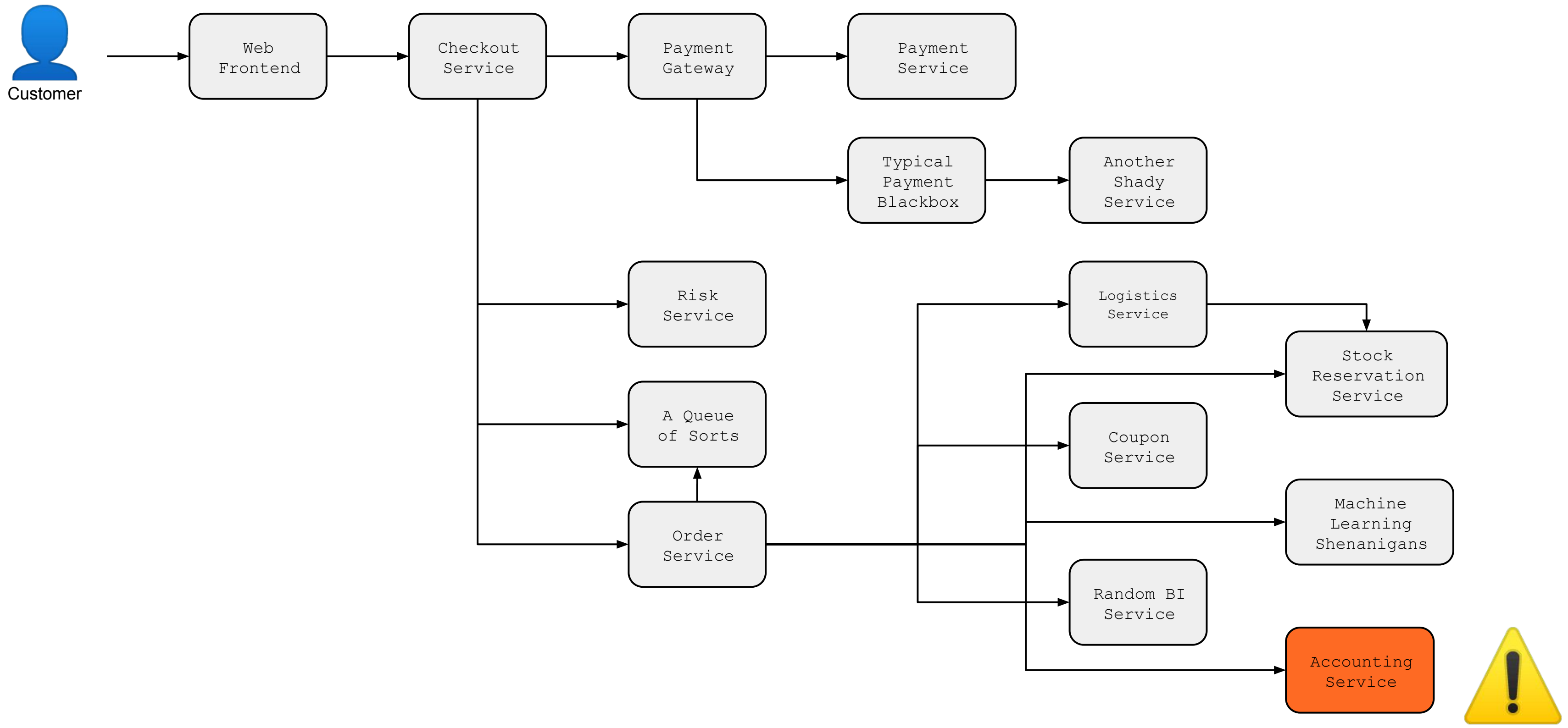
zalando

FAILURE PLACING AN ORDER



zalando

ALERTS ON FAILURE PLACING AN ORDER



zalando

ALERTS ON FAILURE PLACING AN ORDER

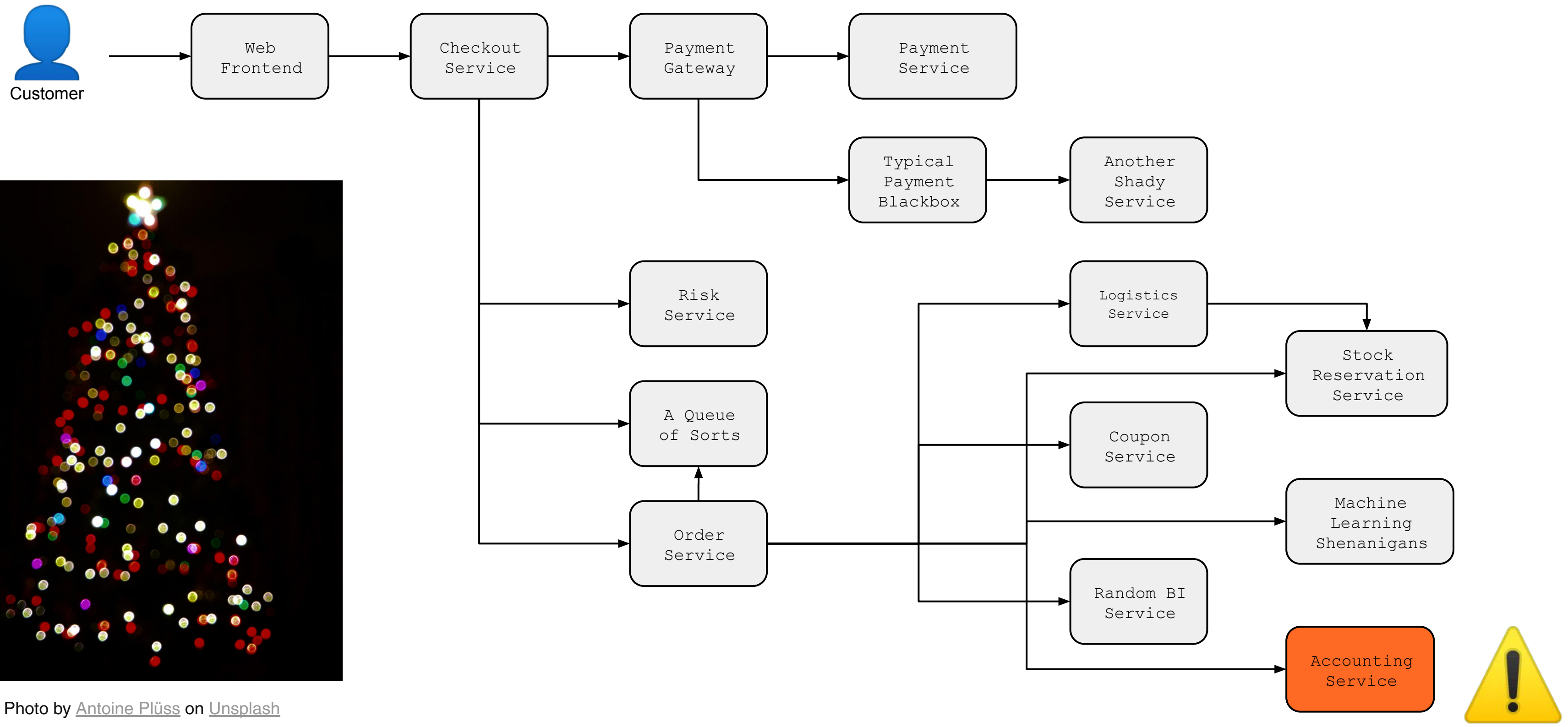
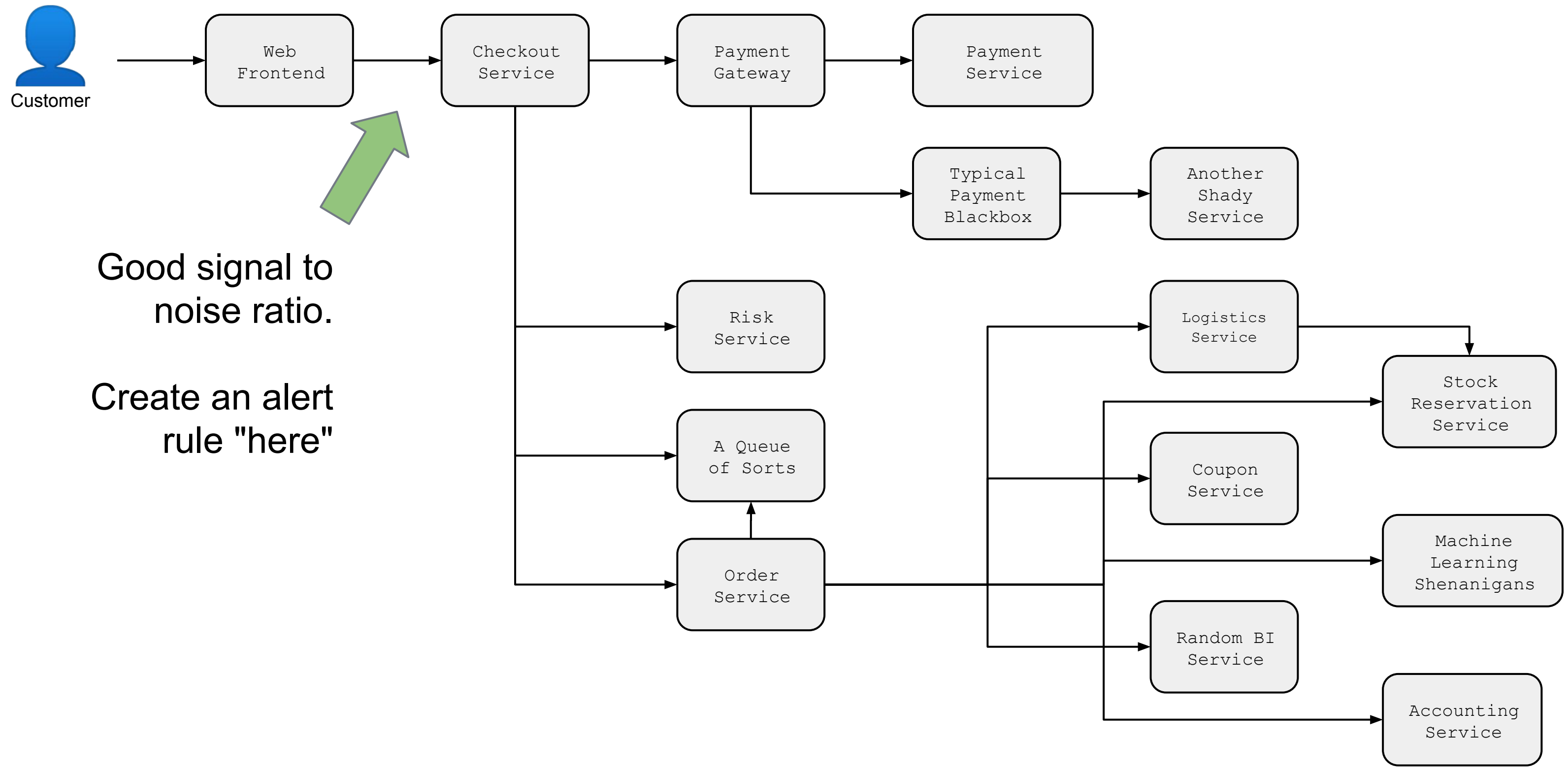


Photo by Antoine Plüss on Unsplash

zalando

SYMPTOM BASED ALERTING RULE

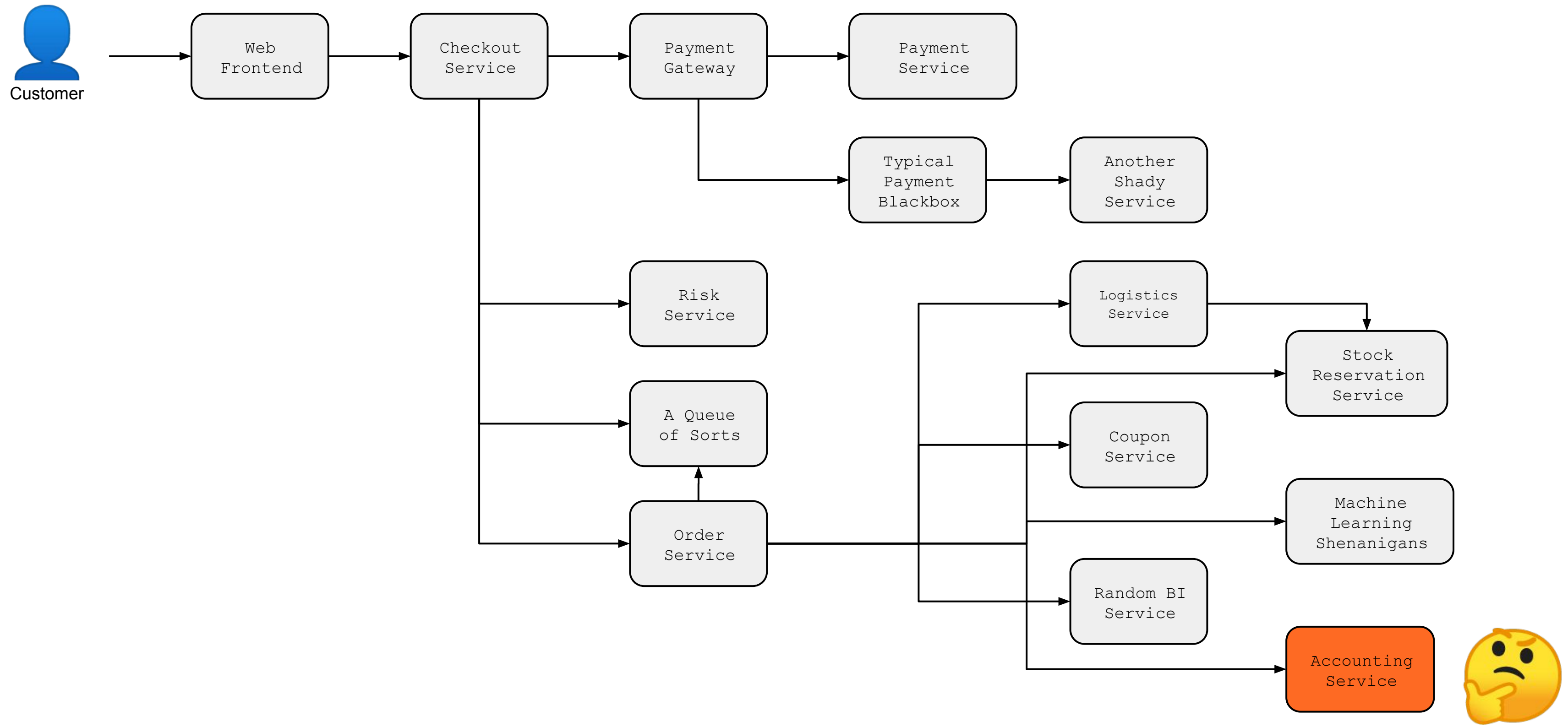


Good signal to noise ratio.

Create an alert rule "here"

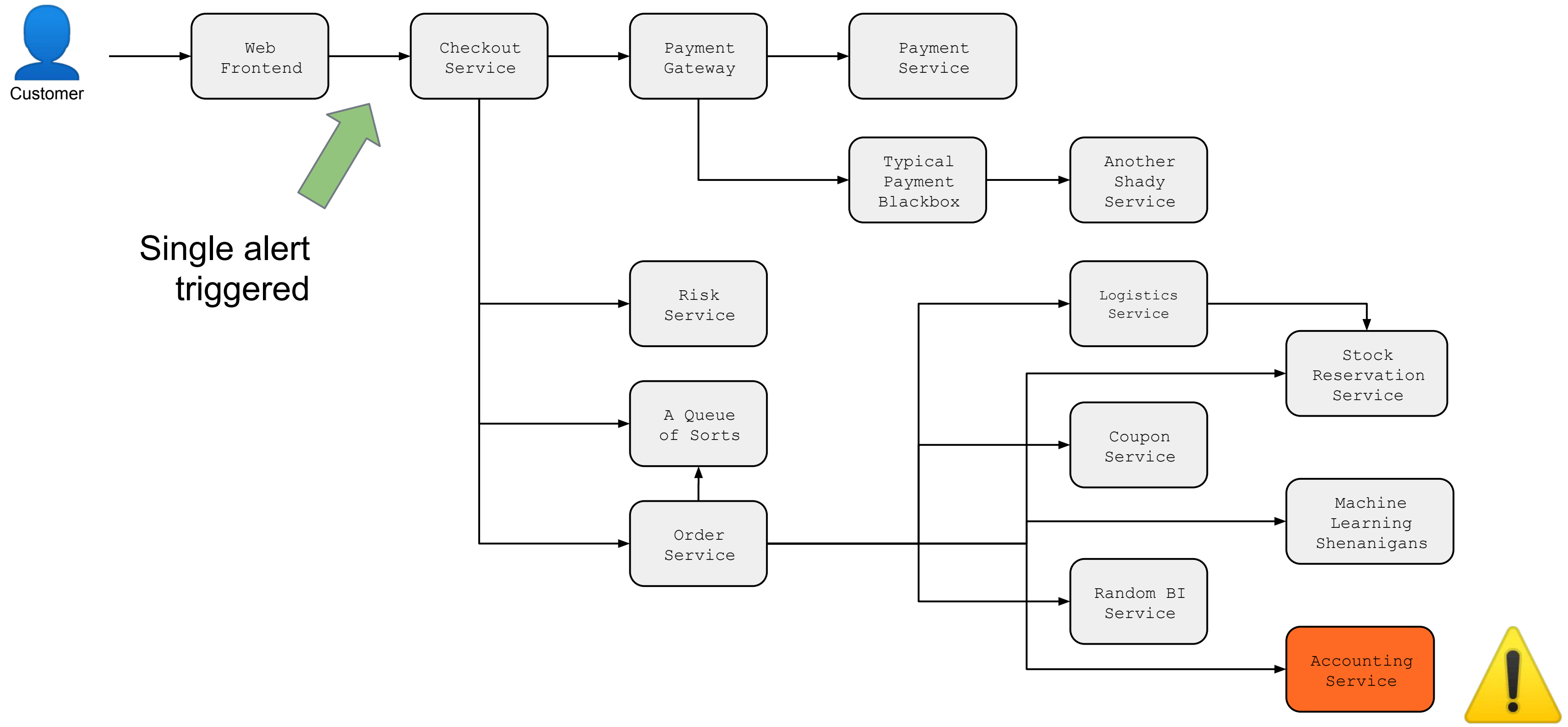
zalando

ALERT ON THE SYMPTOM



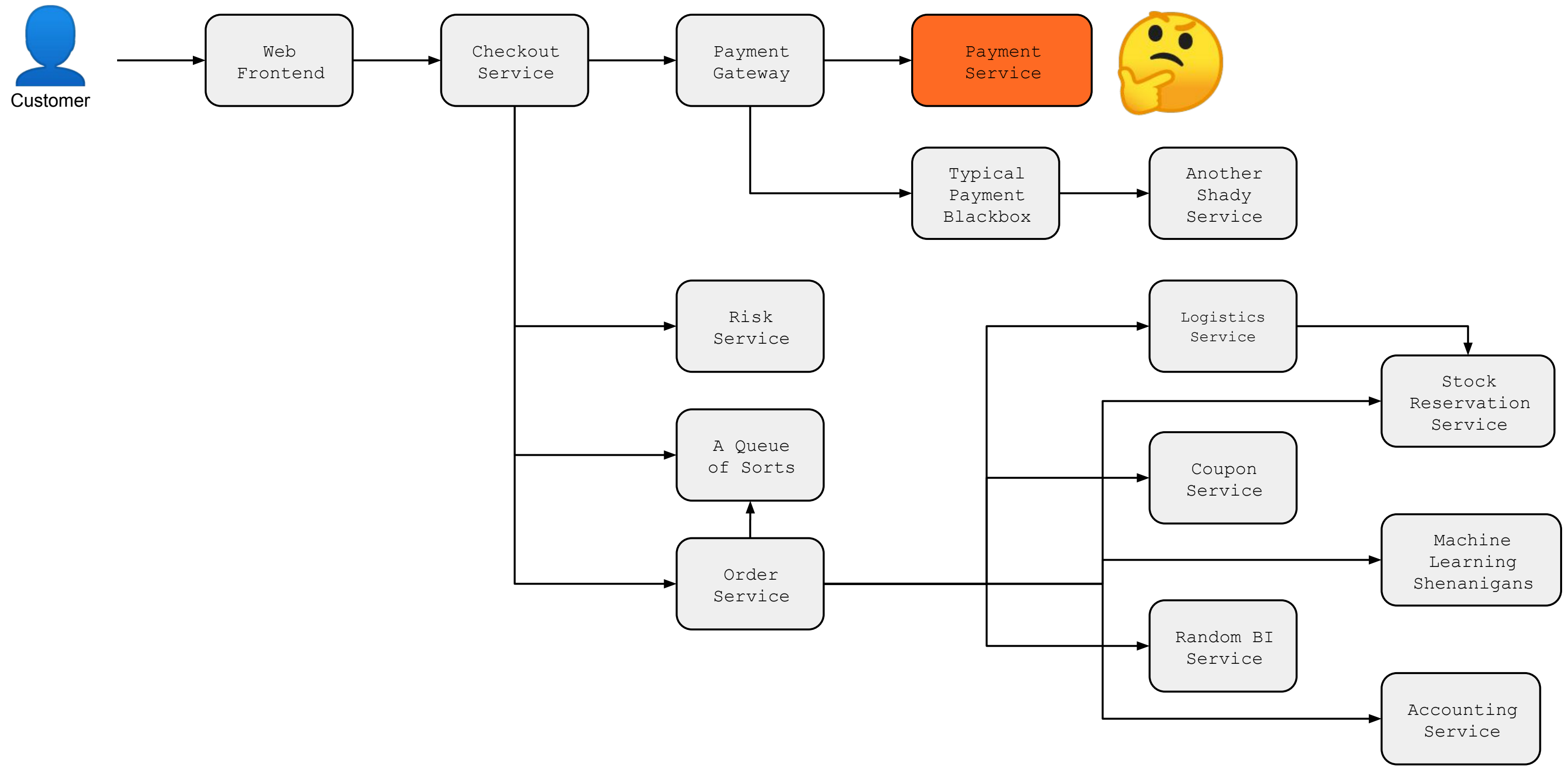
zalando

ALERT ON THE SYMPTOM



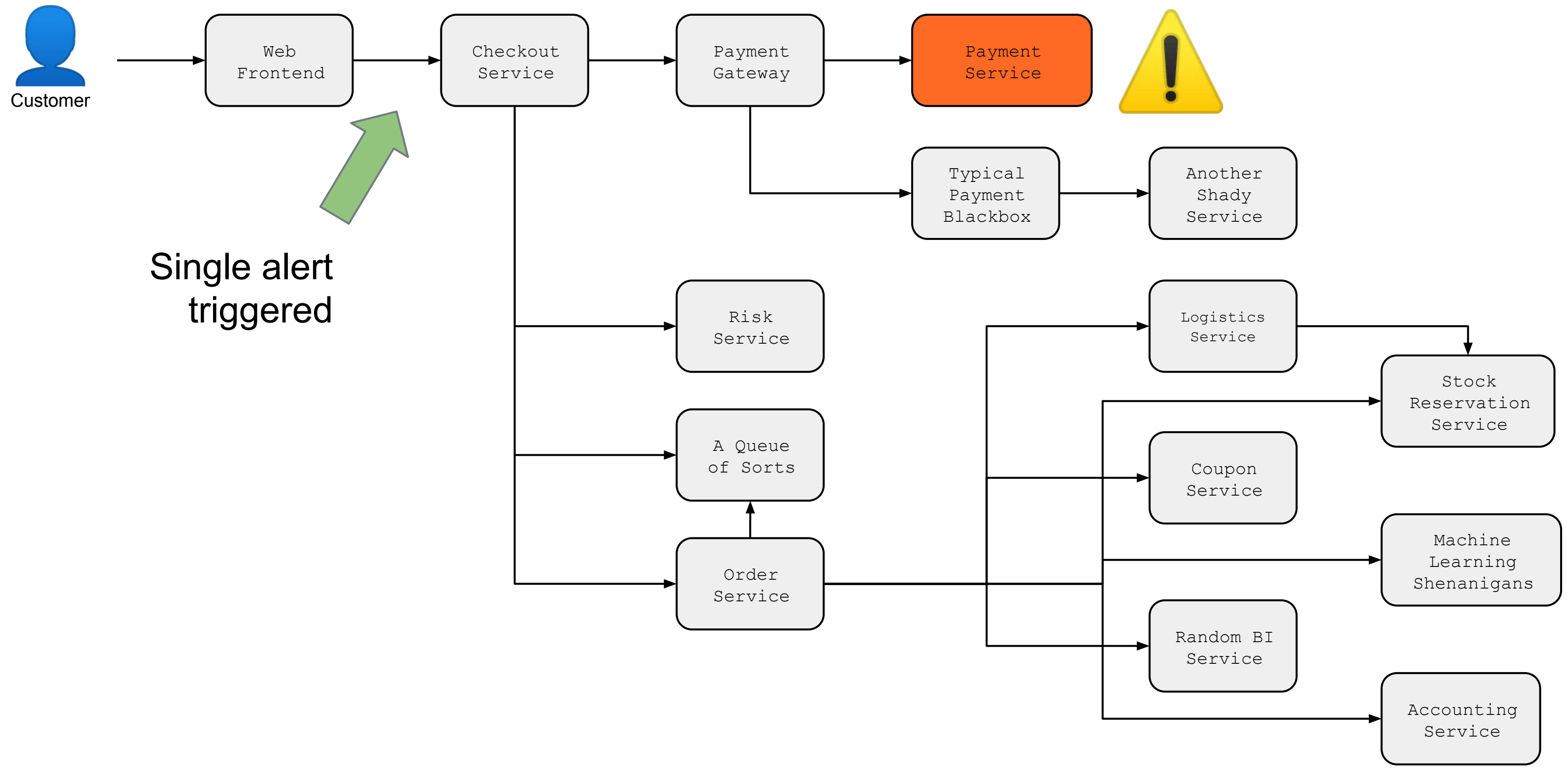
zalando

ALERT ON THE SYMPTOM - DIFFERENT ISSUE



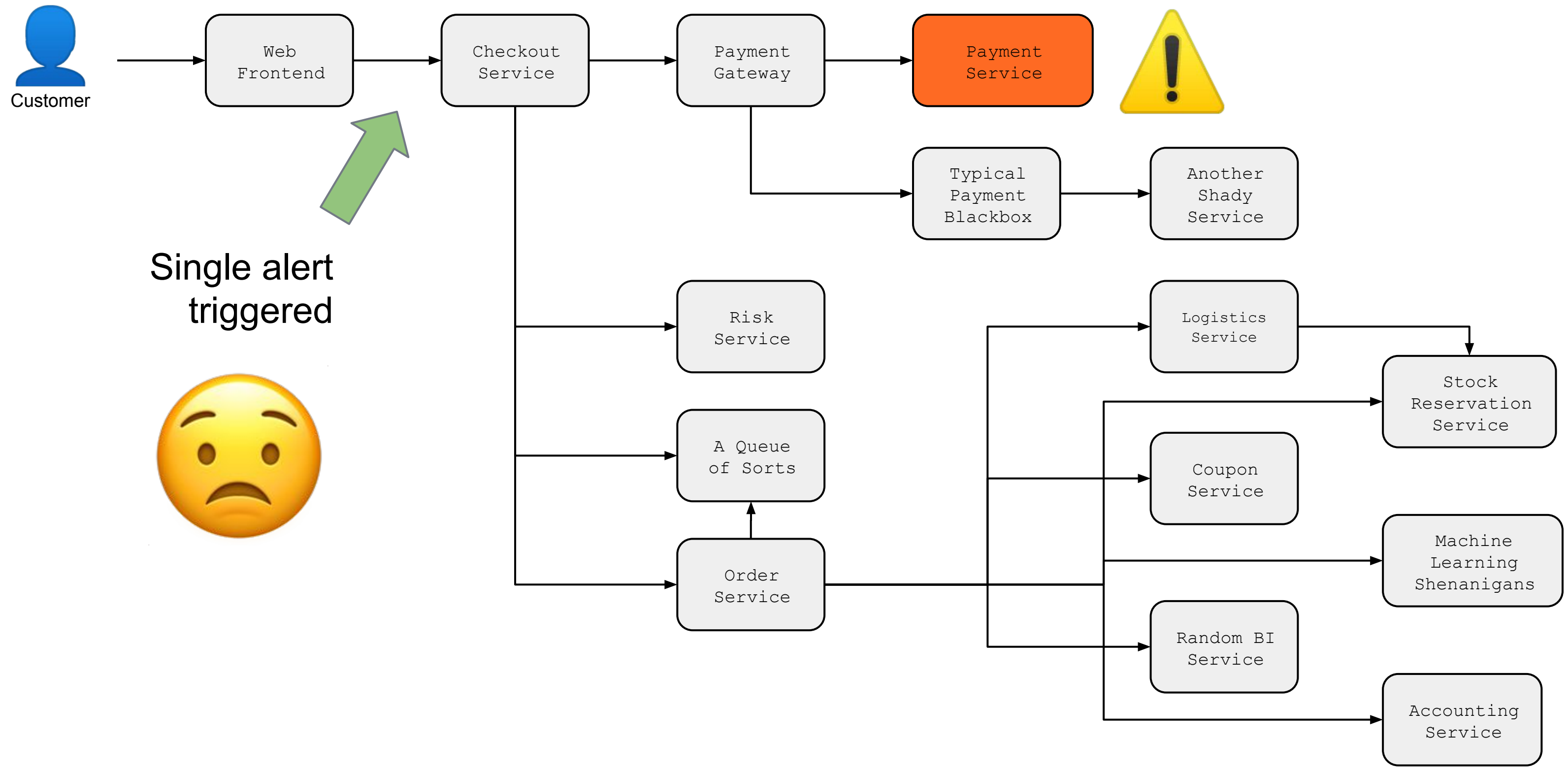
zalando

ALERT ON THE SYMPTOM - DIFFERENT ISSUE



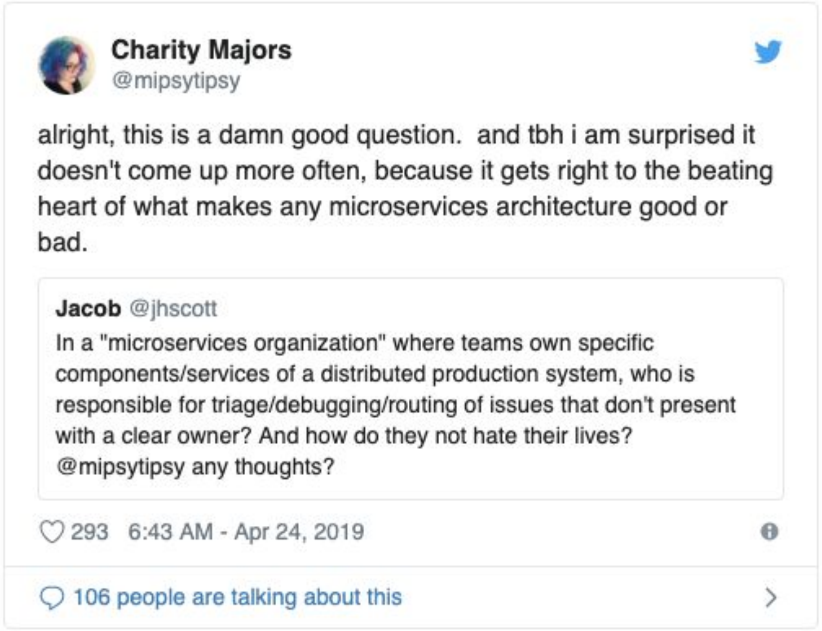
zalando

PLACING AN ORDER - ALERT BOMBING



zalando

ALERTING FOR MICROSERVICES



Charity Majors
@mipsytipsy

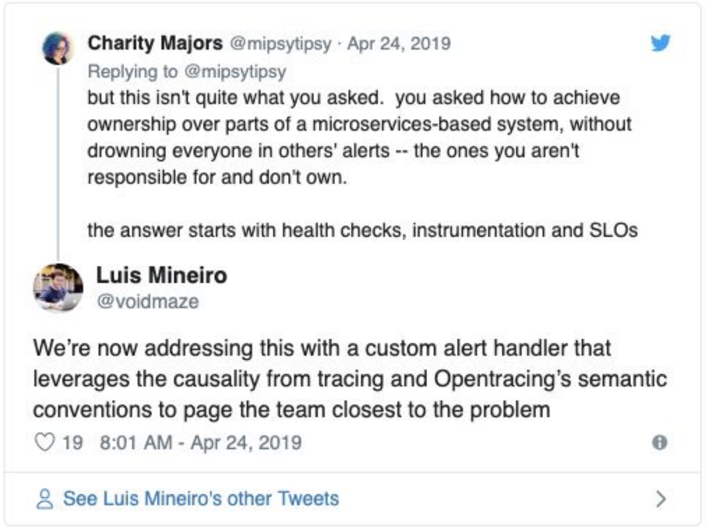
alright, this is a damn good question. and tbh i am surprised it doesn't come up more often, because it gets right to the beating heart of what makes any microservices architecture good or bad.

Jacob @jhscott
In a "microservices organization" where teams own specific components/services of a distributed production system, who is responsible for triage/debugging/routing of issues that don't present with a clear owner? And how do they not hate their lives? @mipsytipsy any thoughts?

293 6:43 AM - Apr 24, 2019

106 people are talking about this

ADAPTIVE PAGING



Charity Majors @mipsytipsy · Apr 24, 2019

Replying to @mipsytipsy

but this isn't quite what you asked. you asked how to achieve ownership over parts of a microservices-based system, without drowning everyone in others' alerts -- the ones you aren't responsible for and don't own.

the answer starts with health checks, instrumentation and SLOs

Luis Mineiro
@voidmaze

We're now addressing this with a custom alert handler that leverages the causality from tracing and Opentracing's semantic conventions to page the team closest to the problem

19 8:01 AM - Apr 24, 2019

See Luis Mineiro's other Tweets

Adaptive Paging is an **alert handler**

that leverages the **causality from tracing**

and **OpenTracing's semantic conventions**

to page the team **closest the problem**.

zalando

DISTRIBUTED TRACING AND OPENTRACING

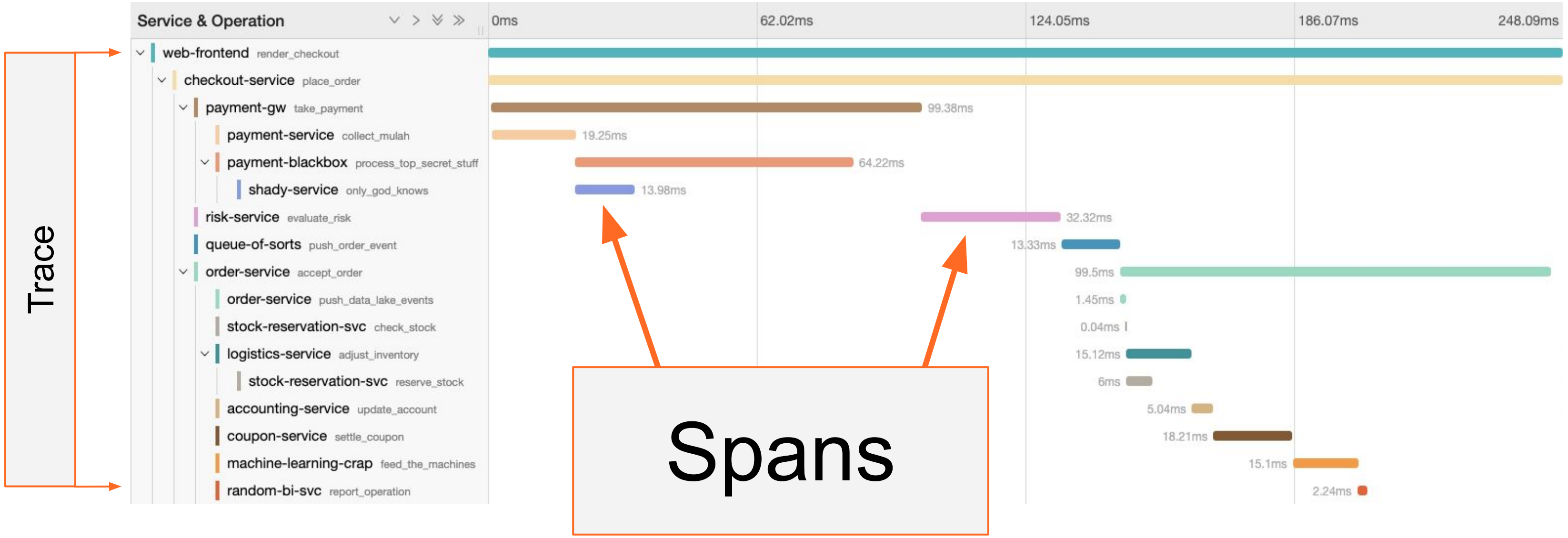
- A trace tells the **story of a transaction or workflow as it propagates** through a distributed system.
- It's basically directed acyclic graph (DAG), with a **clear start** and a **clear end** - no loops.
- A trace is made up of **spans** representing contiguous segments of work in that trace.
- Opentracing is a set of **vendor-neutral APIs** and code instrumentation **standard for distributed tracing**

OPENTRACING

zalando

OPENTRACING CONCEPTS

Span: a named operation which records the **duration**, usually a remote procedure call, with optional **Tags** and Logs.

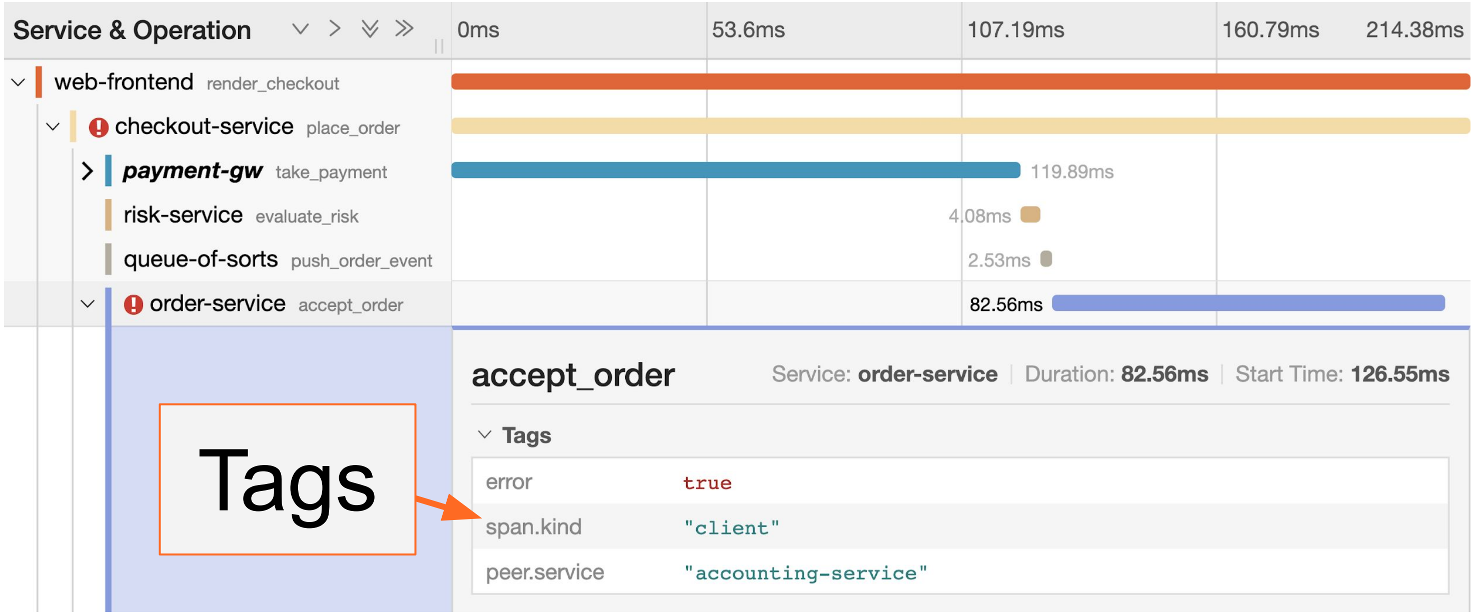


Service & Operation	
web-frontend render_checkout	
checkout-service place_order	
payment-gw take_payment	99.38ms
payment-service collect_mulah	19.25ms
payment-blackbox process_top_secret_stuff	64.22ms
shady-service only_god_knows	13.98ms
risk-service evaluate_risk	32.32ms
queue-of-sorts push_order_event	13.33ms
order-service accept_order	99.5ms
order-service push_data_lake_events	1.45ms
stock-reservation-svc check_stock	0.04ms
logistics-service adjust_inventory	15.12ms
stock-reservation-svc reserve_stock	6ms
accounting-service update_account	5.04ms
coupon-service settle_coupon	18.21ms
machine-learning-crap feed_the_machines	15.1ms
random-bi-svc report_operation	2.24ms

zalando

OPENTRACING CONCEPTS

Tag: A "mostly" arbitrary **Key:Value pair** (value can be a string, number or bool)


Service & Operation
0ms
53.6ms
107.19ms
160.79ms
214.38ms
web-frontend render_checkout
checkout-service place_order
payment-gw take_payment
119.89ms
risk-service evaluate_risk
4.08ms
queue-of-sorts push_order_event
2.53ms
order-service accept_order
82.56ms
Tags

accept_order — Service: **order-service** | Duration: **82.56ms** | Start Time: **126.55ms**

Tags

error	true
span.kind	"client"
peer.service	"accounting-service"

zalando

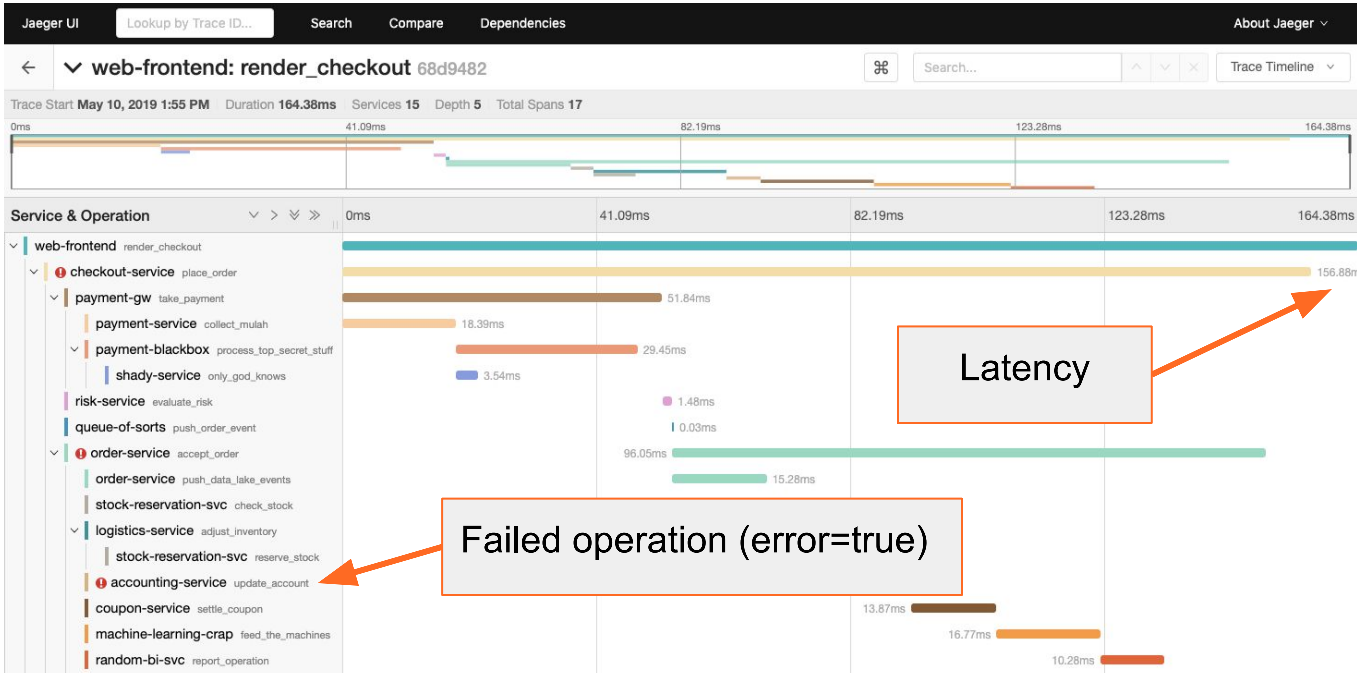
OPENTRACING SEMANTIC CONVENTIONS

Span tag name	Type	Notes and examples
component	string	The **software package**, framework, library, or module that generated the associated Span. E.g., "checkout-service".
error	bool	**true** if and only if the application considers the operation represented by the Span to have failed
peer.service	string	**Remote service name** (for some unspecified definition of "service"). E.g., "accounting-service"
span.kind	string	Either "client" or "server" for the appropriate roles in an RPC.
… and more		

Opentracing semantic conventions

zalando

OPENTRACING MONITORING SIGNALS



web-frontend: render_checkout 68d9482

Trace Start **May 10, 2019 1:55 PM** Duration **164.38ms** Services **15** Depth **5** Total Spans **17**

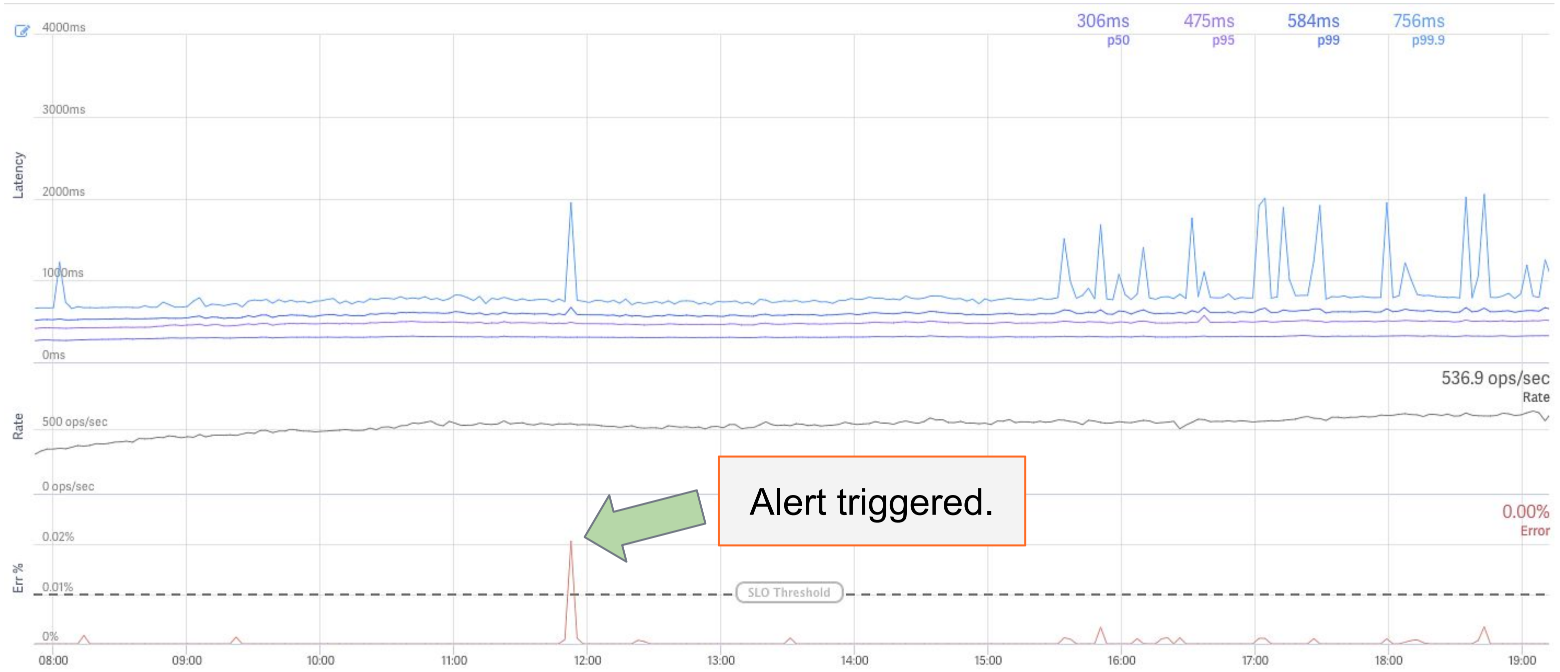
0ms 41.09ms 82.19ms 123.28ms 164.38ms

```
Service & Operation                                   0ms   41.09ms   82.19ms   123.28ms   164.38ms
web-frontend render_checkout
  checkout-service place_order                         156.88m
    payment-gw take_payment                            51.84ms
      payment-service collect_mulah                    18.39ms
      payment-blackbox process_top_secret_stuff        29.45ms
        shady-service only_god_knows                   3.54ms
    risk-service evaluate_risk                         1.48ms
    queue-of-sorts push_order_event                    0.03ms
    order-service accept_order                         96.05ms
      order-service push_data_lake_events              15.28ms
      stock-reservation-svc check_stock
      logistics-service adjust_inventory
        stock-reservation-svc reserve_stock
      accounting-service update_account
      coupon-service settle_coupon                     13.87ms
      machine-learning-crap feed_the_machines          16.77ms
      random-bi-svc report_operation                   10.28ms
```

The Four Golden Signals
SRE Book, Chapter 6: Monitoring Distributed Systems

zalando

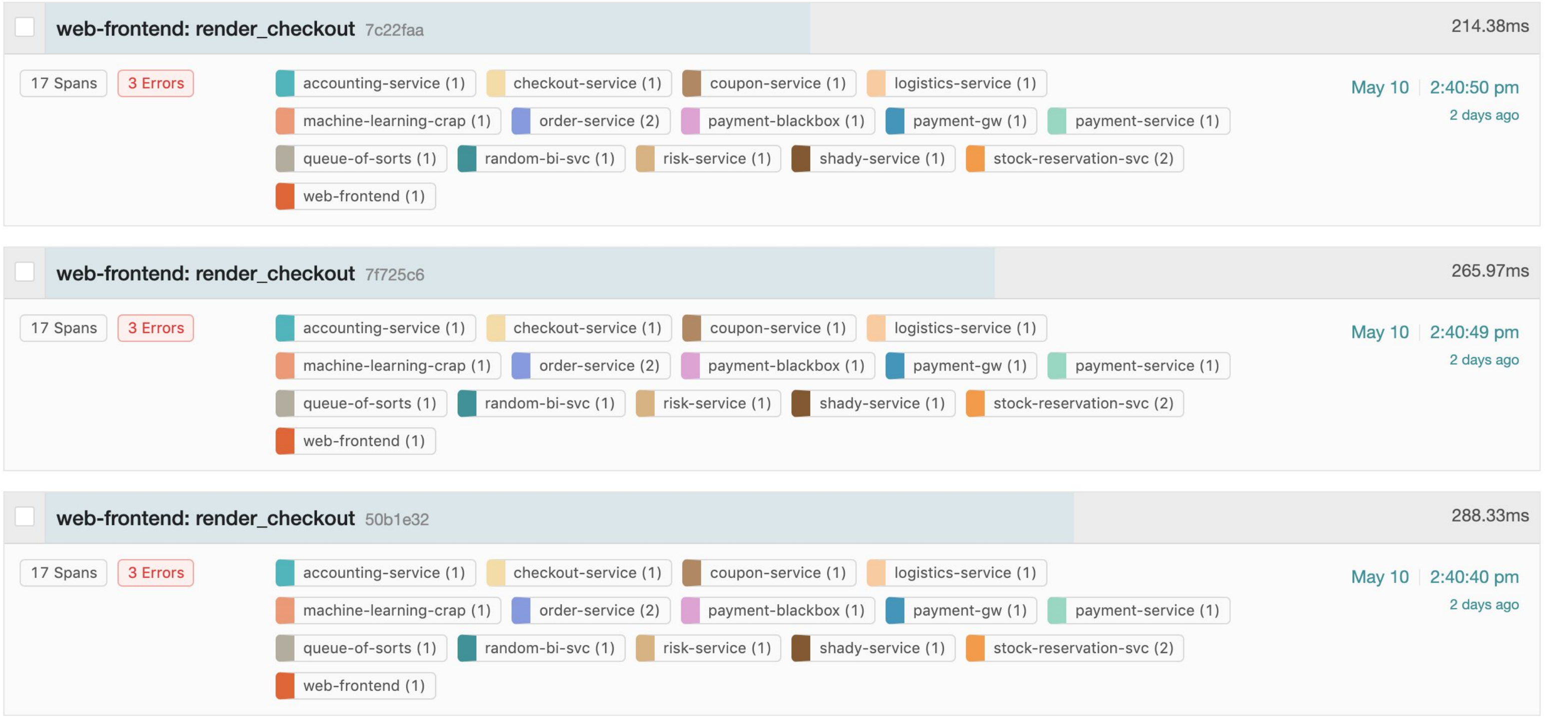
ERROR RATE ALERTING RULE



component: **checkout_service** && operation: **place_order** && error: **true**

zalando

ALERT PAYLOAD



web-frontend: render_checkout 7c22faa 214.38ms

17 Spans 3 Errors

accounting-service (1) checkout-service (1) coupon-service (1) logistics-service (1) machine-learning-crap (1) order-service (2) payment-blackbox (1) payment-gw (1) payment-service (1) queue-of-sorts (1) random-bi-svc (1) risk-service (1) shady-service (1) stock-reservation-svc (2) web-frontend (1)

May 10 | 2:40:50 pm
2 days ago

web-frontend: render_checkout 7f725c6 265.97ms

17 Spans 3 Errors

accounting-service (1) checkout-service (1) coupon-service (1) logistics-service (1) machine-learning-crap (1) order-service (2) payment-blackbox (1) payment-gw (1) payment-service (1) queue-of-sorts (1) random-bi-svc (1) risk-service (1) shady-service (1) stock-reservation-svc (2) web-frontend (1)

May 10 | 2:40:49 pm
2 days ago

web-frontend: render_checkout 50b1e32 288.33ms

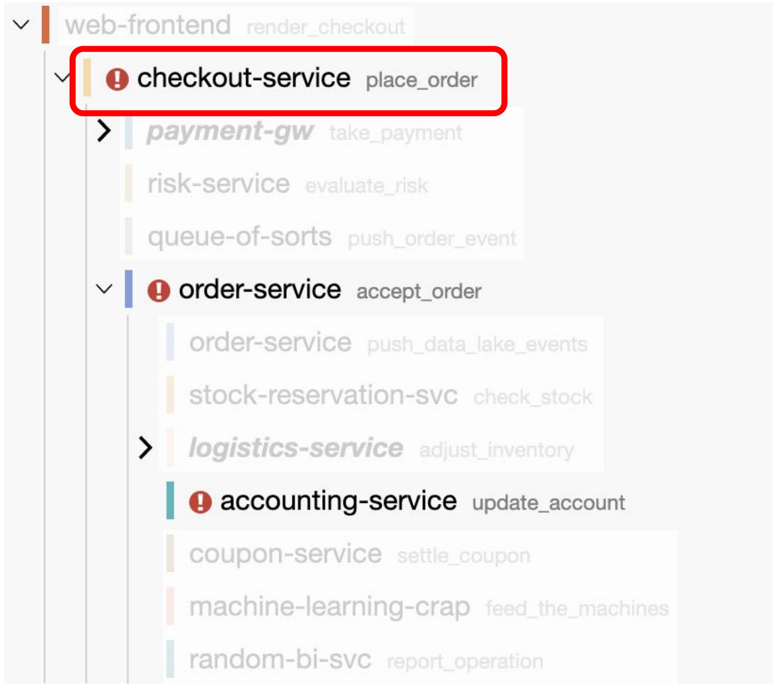
17 Spans 3 Errors

accounting-service (1) checkout-service (1) coupon-service (1) logistics-service (1) machine-learning-crap (1) order-service (2) payment-blackbox (1) payment-gw (1) payment-service (1) queue-of-sorts (1) random-bi-svc (1) risk-service (1) shady-service (1) stock-reservation-svc (2) web-frontend (1)

May 10 | 2:40:40 pm
2 days ago

zalando

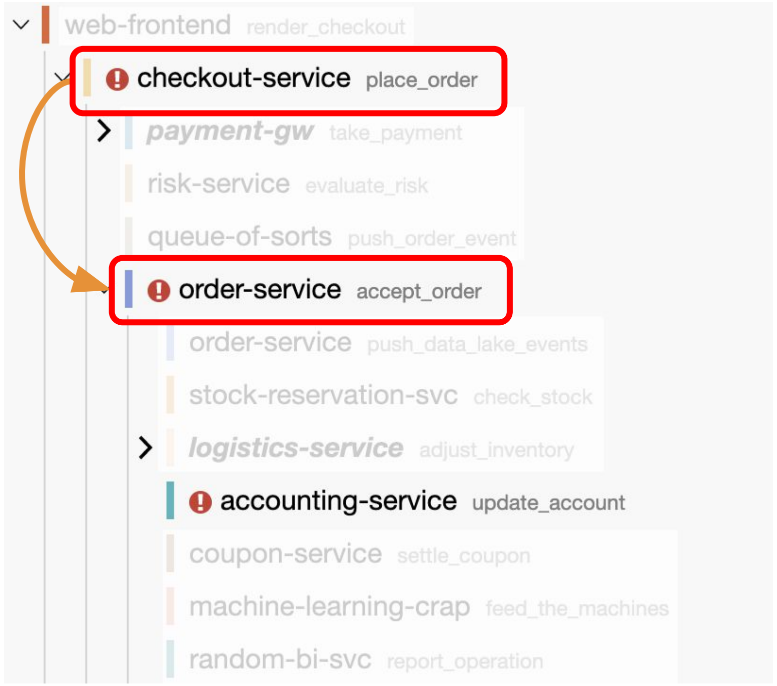
WALKING THROUGH A TRACE


web-frontend render_checkout
checkout-service place_order
payment-gw take_payment
risk-service evaluate_risk
queue-of-sorts push_order_event
order-service accept_order
order-service push_data_lake_events
stock-reservation-svc check_stock
logistics-service adjust_inventory
accounting-service update_account
coupon-service settle_coupon
machine-learning-crap feed_the_machines
random-bi-svc report_operation

1. Starting at the span which was defined as the signal - **place_order**

zalando

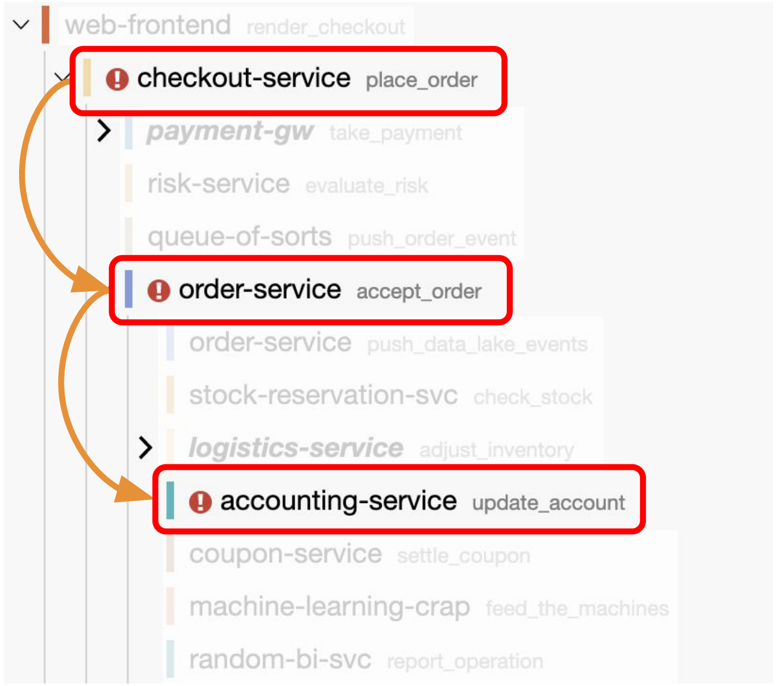
WALKING THROUGH A TRACE


web-frontend render_checkout
checkout-service place_order
payment-gw take_payment
risk-service evaluate_risk
queue-of-sorts push_order_event
order-service accept_order
order-service push_data_lake_events
stock-reservation-svc check_stock
logistics-service adjust_inventory
accounting-service update_account
coupon-service settle_coupon
machine-learning-crap feed_the_machines
random-bi-svc report_operation

1. Starting at the span which was defined as the signal - **place_order**
2. Inspect every child span's tags
3. Follow path with **error=true**

zalando

WALKING THROUGH A TRACE

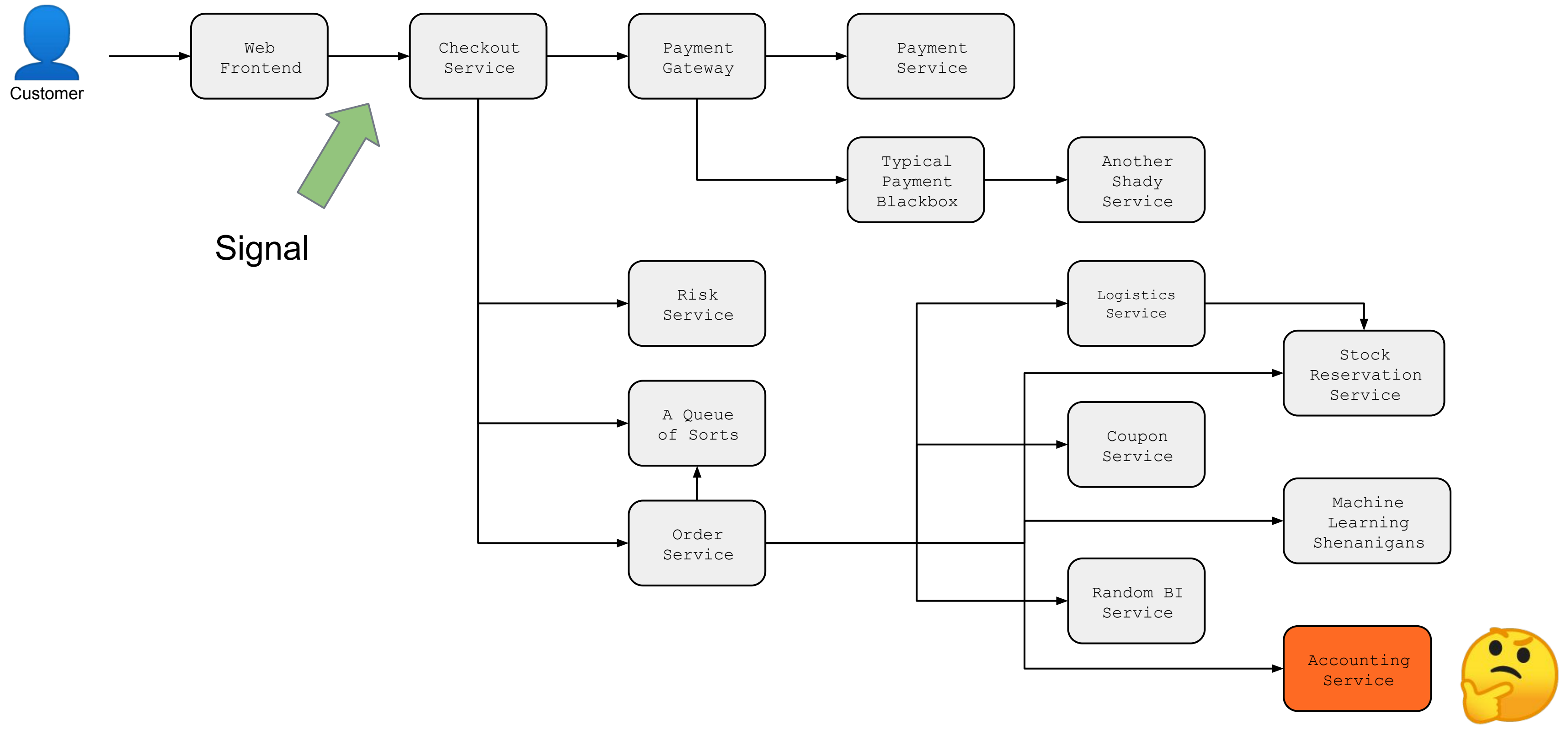


web-frontend render_checkout
checkout-service place_order
payment-gw take_payment
risk-service evaluate_risk
queue-of-sorts push_order_event
order-service accept_order
order-service push_data_lake_events
stock-reservation-svc check_stock
logistics-service adjust_inventory
accounting-service update_account
coupon-service settle_coupon
machine-learning-crap feed_the_machines
random-bi-svc report_operation

1. Starting at the span which was defined as the signal - **place_order**
2. Inspect every child span's tags
3. Follow path with **error=true**
4. Rinse and repeat until no more children

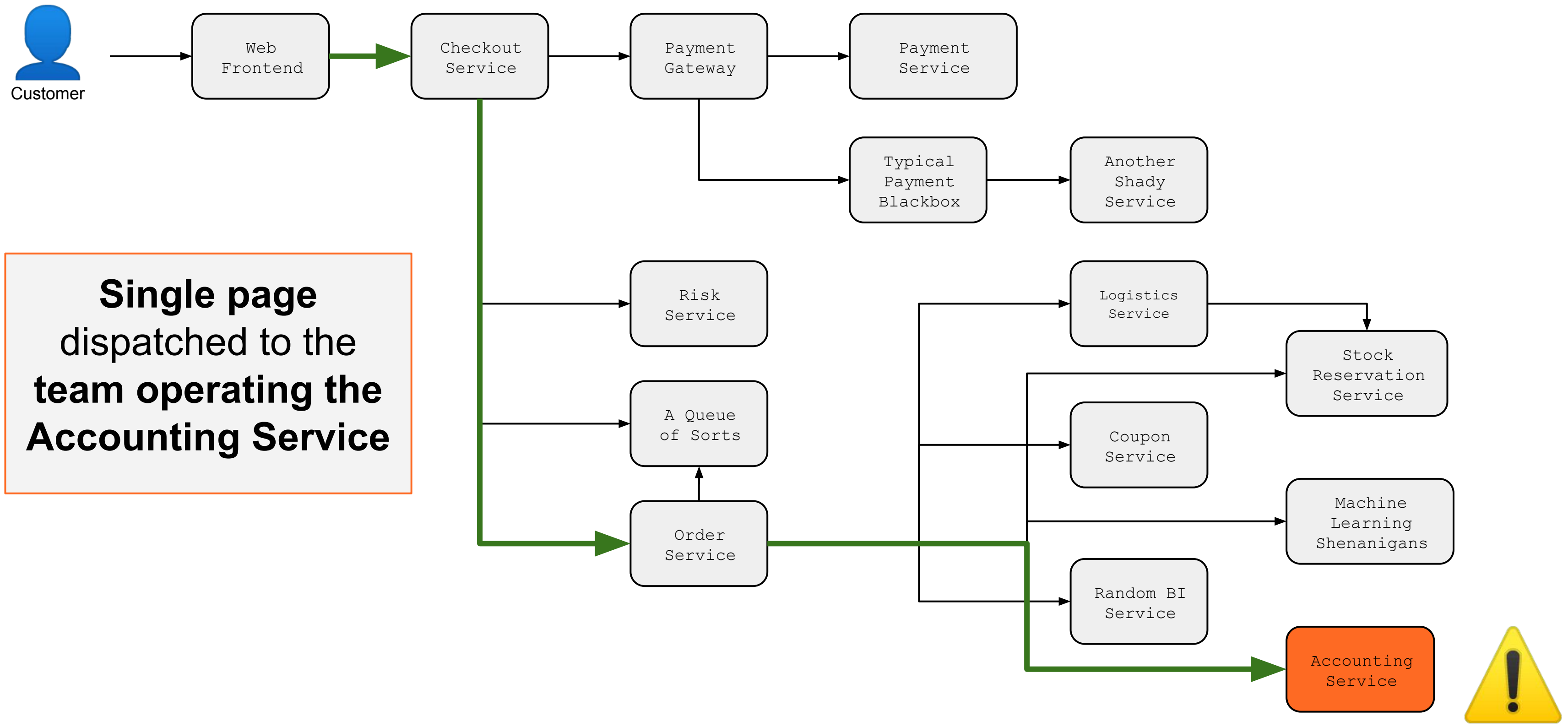
zalando

ALERT ON THE SYMPTOM



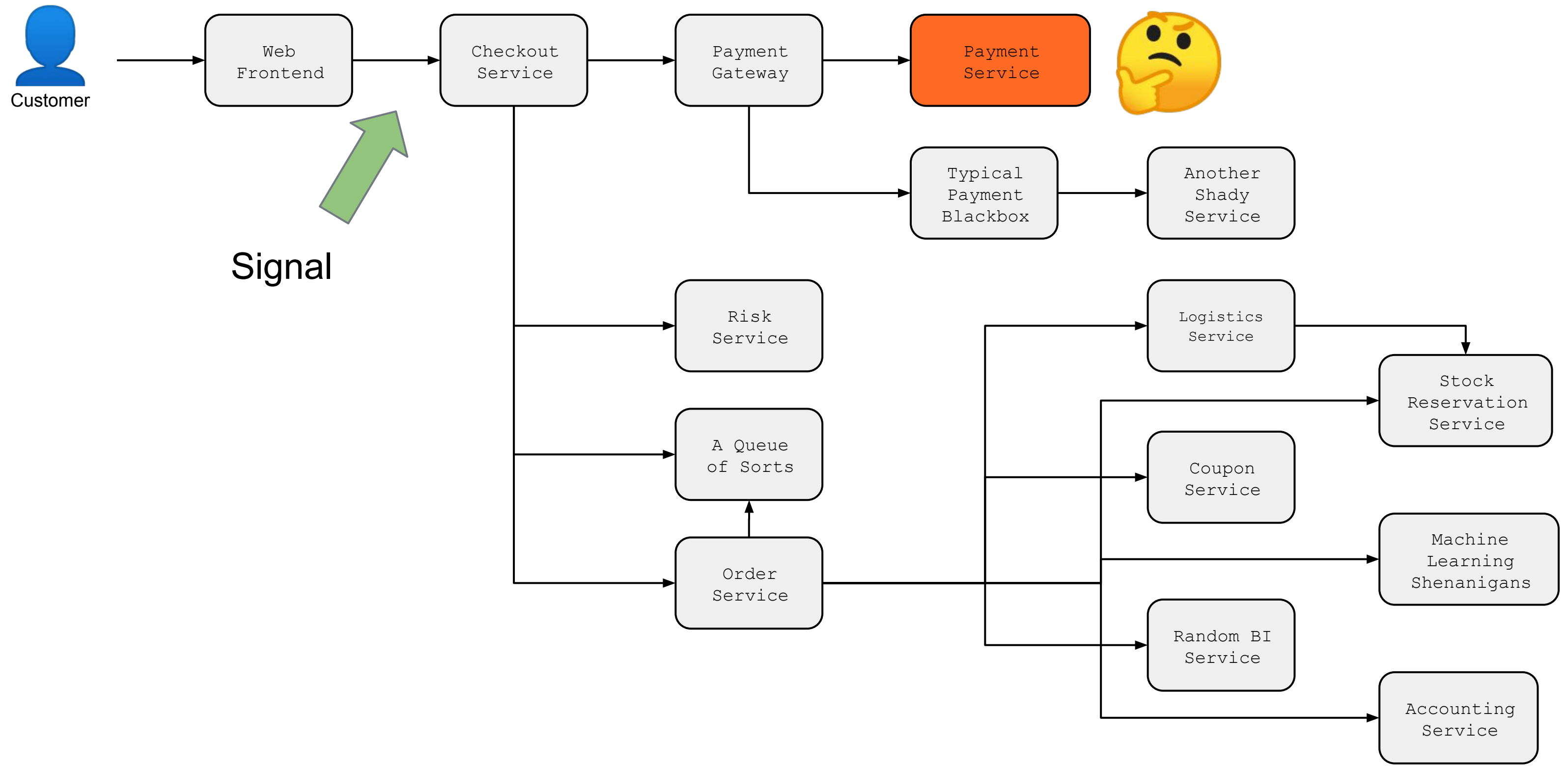
zalando

ALERT ON THE SYMPTOM



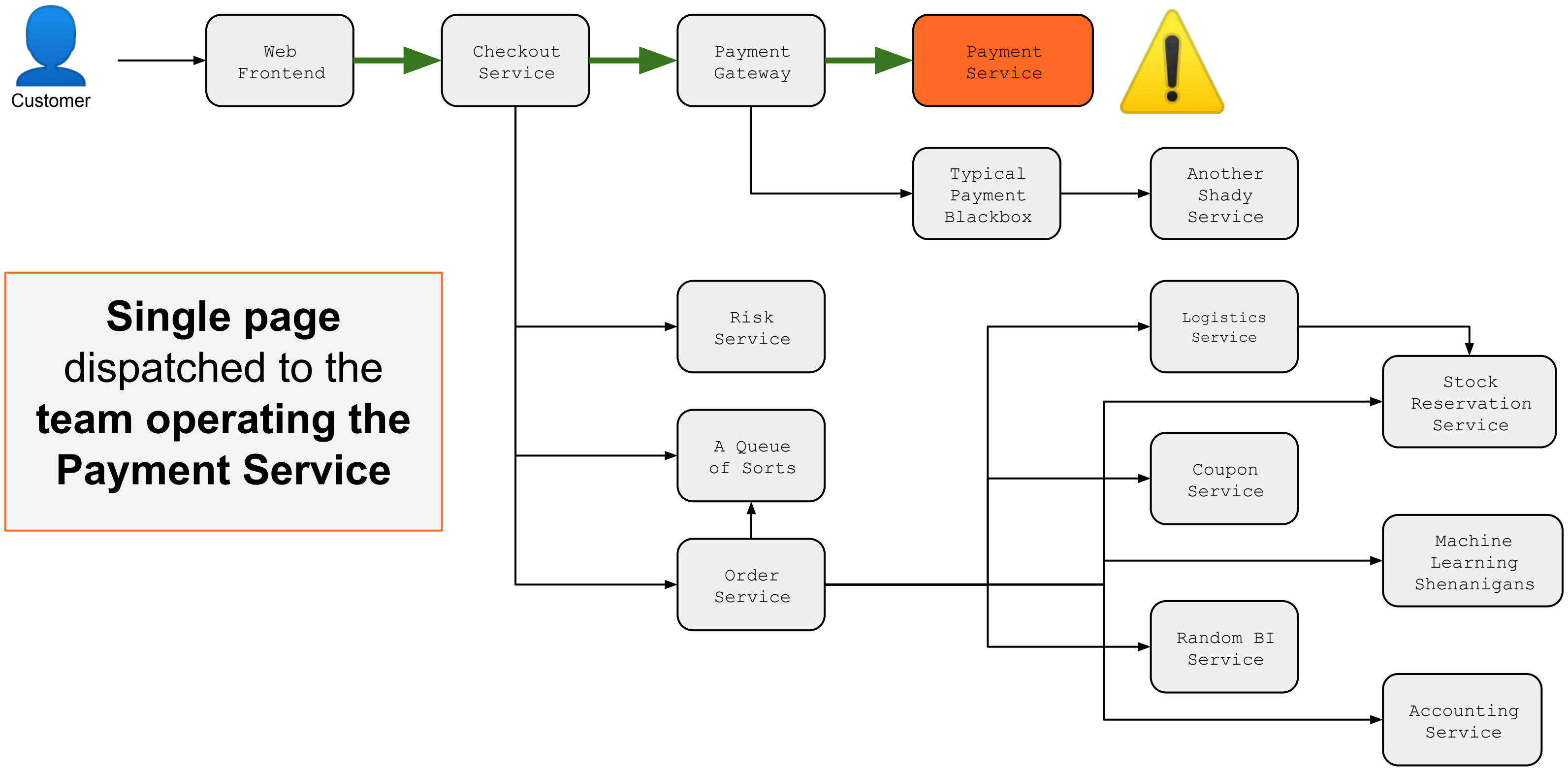
Single page dispatched to the **team operating the Accounting Service**

ALERT ON THE SYMPTOM - DIFFERENT ISSUE



zalando

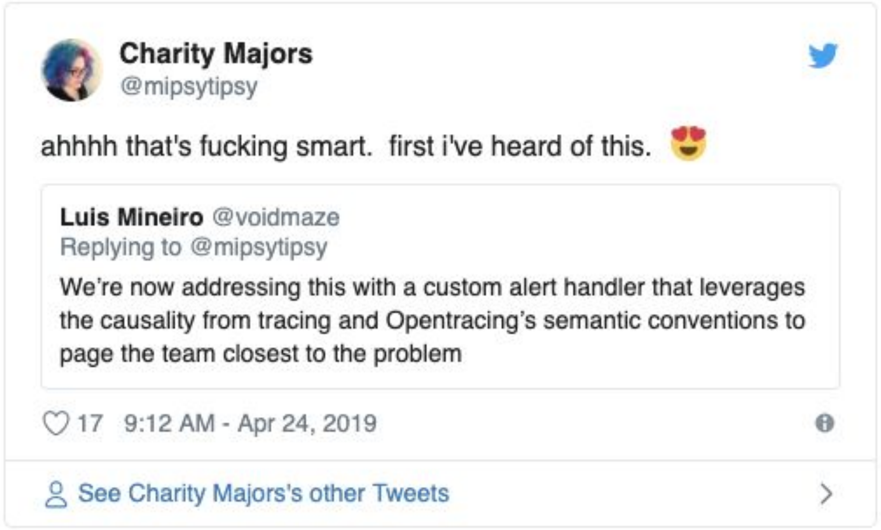
ALERT ON THE SYMPTOM - DIFFERENT ISSUE



Single page dispatched to the **team operating the Payment Service**

zalando

ADAPTIVE PAGING



Charity Majors
@mipsytipsy

ahhhh that's fucking smart. first i've heard of this. 😍

Luis Mineiro @voidmaze
Replying to @mipsytipsy
We're now addressing this with a custom alert handler that leverages the causality from tracing and Opentracing's semantic conventions to page the team closest to the problem

♡ 17   9:12 AM - Apr 24, 2019

See Charity Majors's other Tweets

zalando

CHALLENGES

- Multiple child spans with error=true:
  - Follow each path, attribute the probable cause a score
  - Analyze more exemplars and adjust the scores
  - Worse case scenario, page both probable causes
- Missing instrumentation or circuit breaker open
  - Use the **peer.service** and **span.kind=client** tag to suggest which dependency would be the target
- Mapping services to escalation
  - Owning team may not have their own on-call escalation.

Photo by Patrick Tomasso on Unsplash

QUESTIONS?

Luis Mineiro @voidmaze

We're Hiring!
https://jobs.zalando.com

zalando