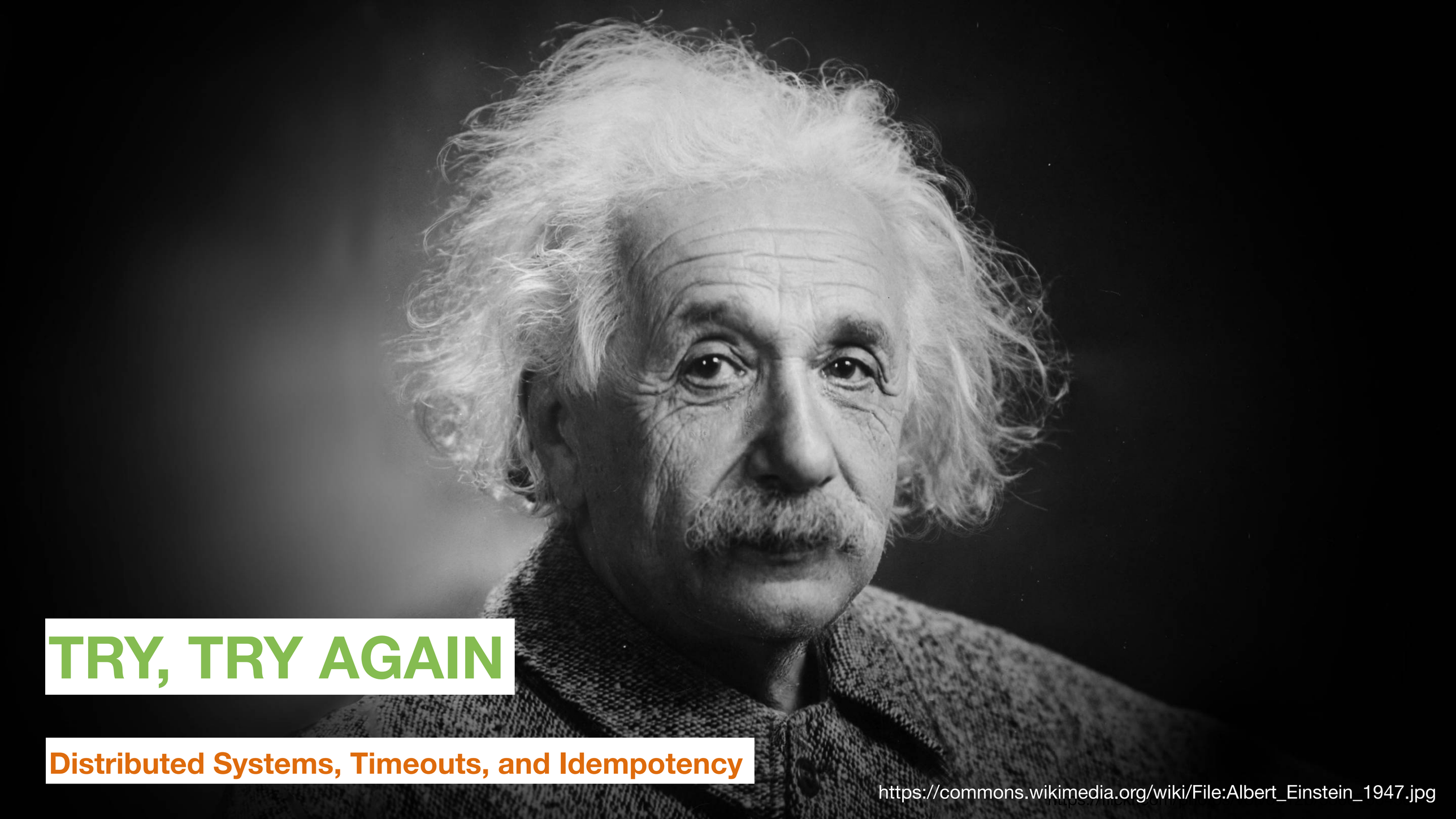
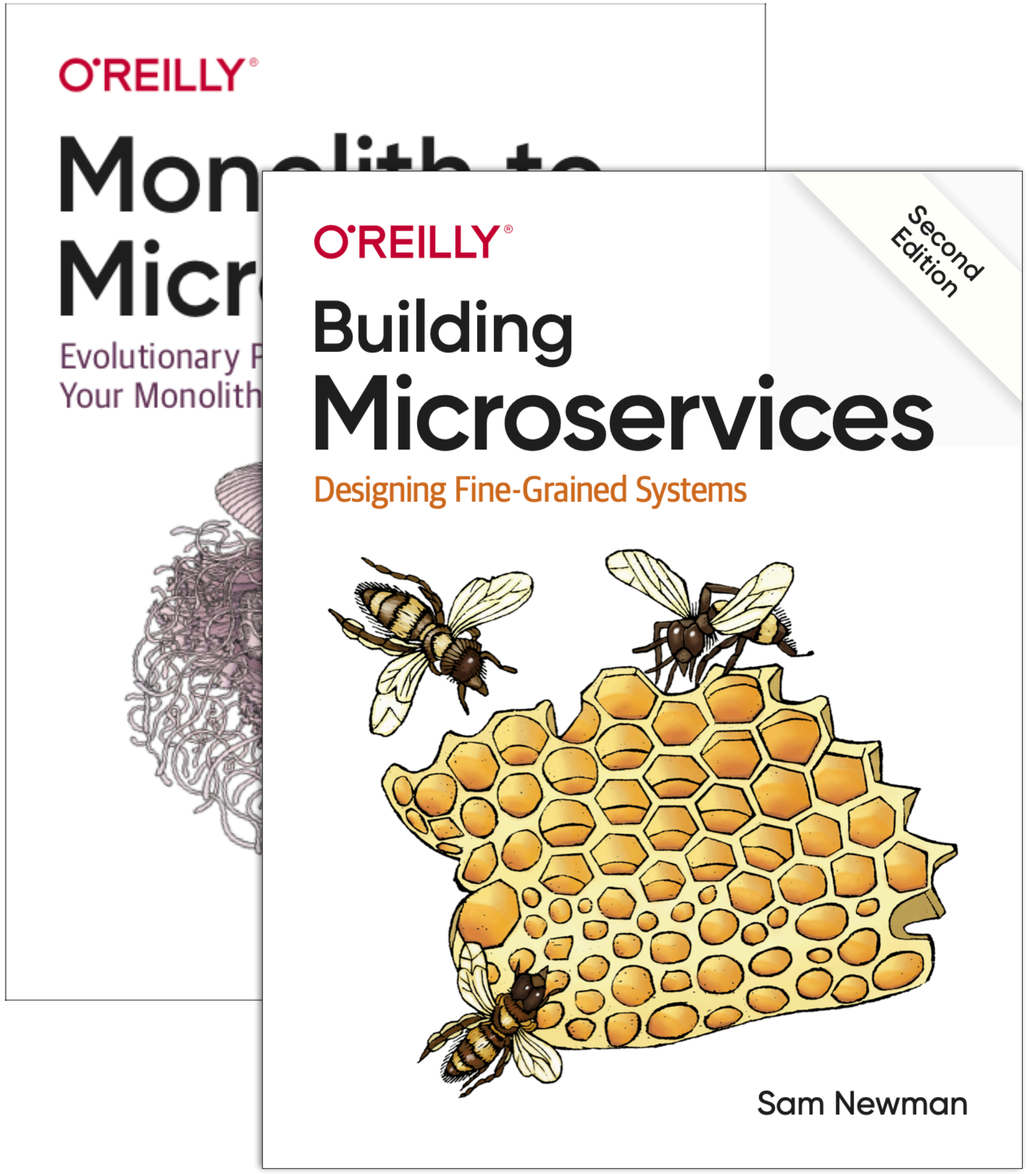
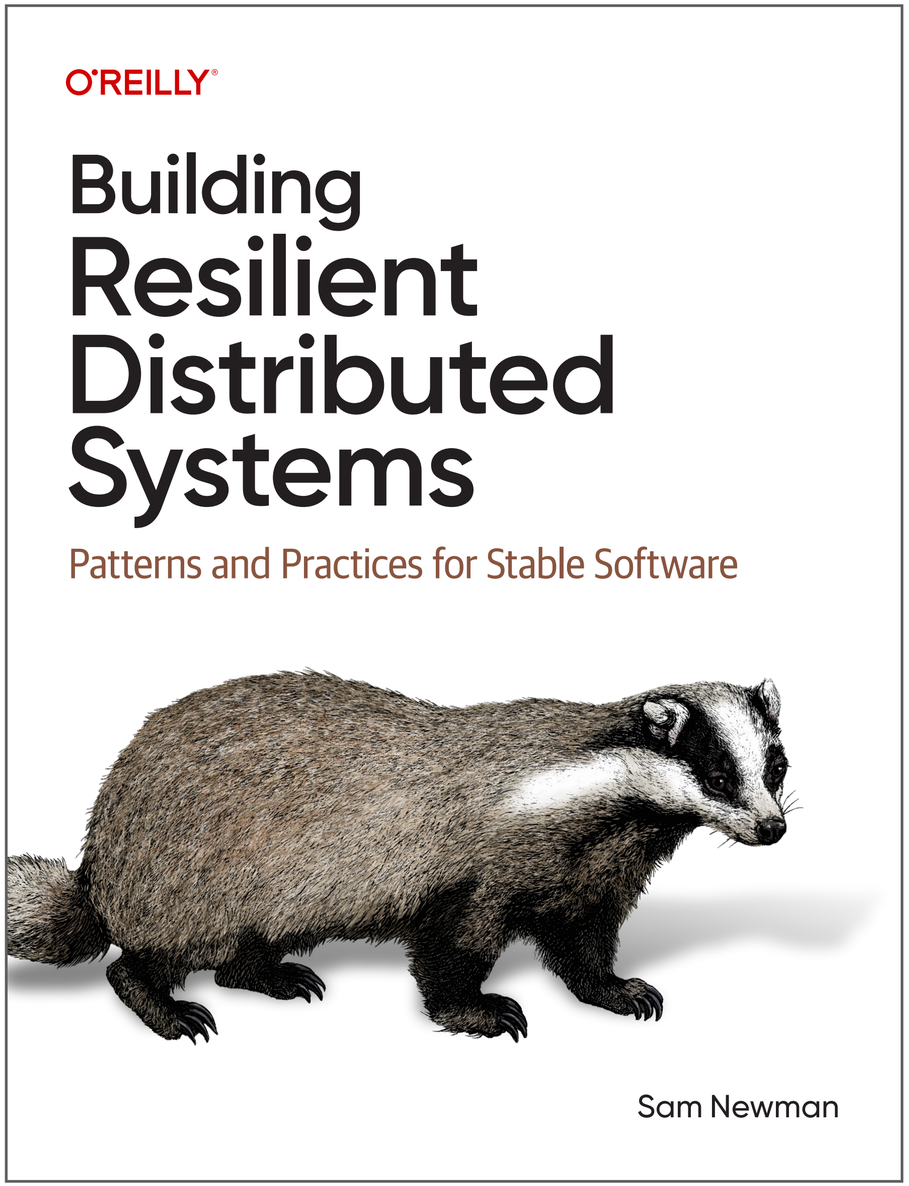


TRY, TRY AGAIN

Distributed Systems, Timeouts, and Idempotency

https://commons.wikimedia.org/wiki/File:Albert_Einstein_1947.jpg


O'REILLY
Mon
Micr
Evolutionary
Your Monolith
O'REILLY
Second Edition
Building
Microservices
Designing Fine-Grained Systems
Sam Newman


O'REILLY
Building
Resilient
Distributed
Systems
Patterns and Practices for Stable Software
Sam Newman

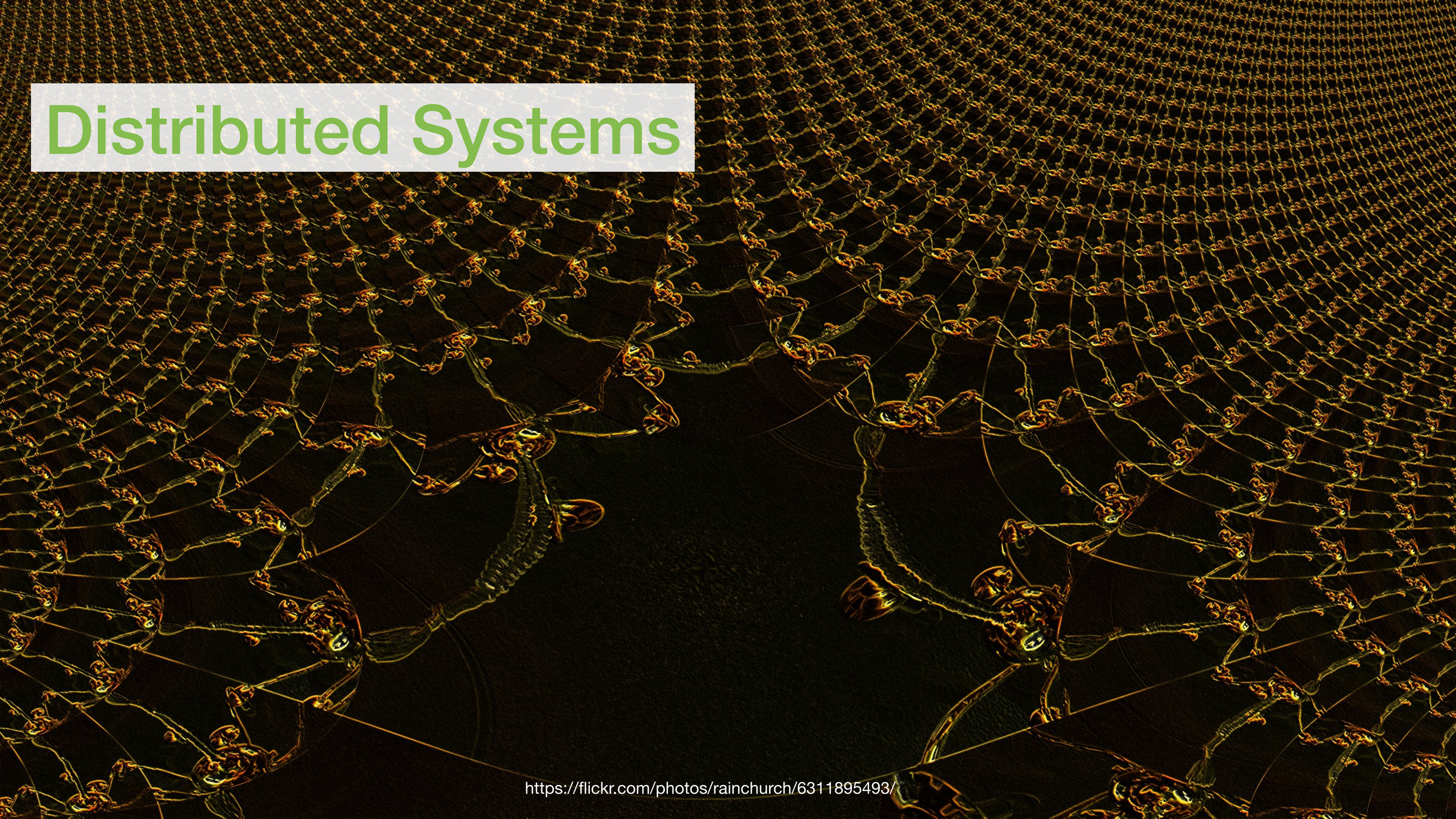
Retries

https://flickr.com/photos/mn1705/14907413228/

Idempotency

https://flickr.com/photos/skuds/3781599677/

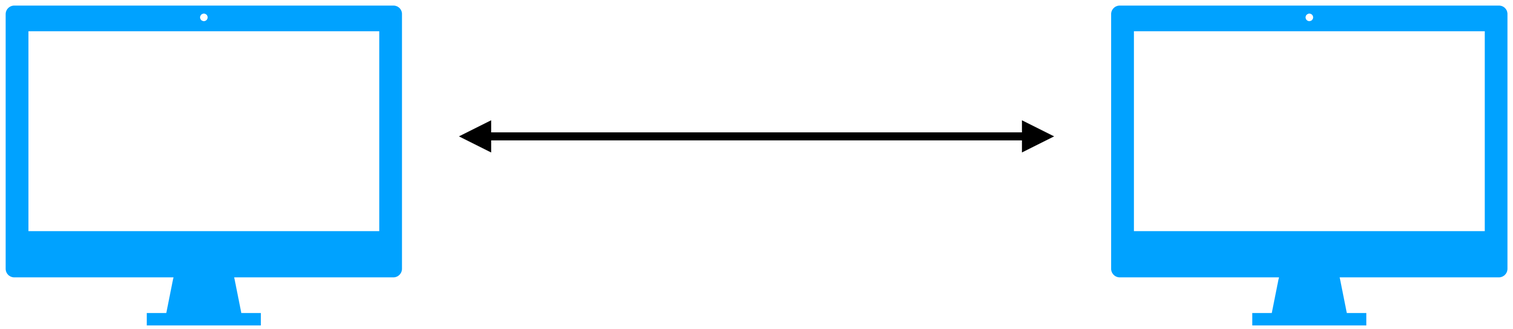
Distributed Systems



https://flickr.com/photos/rainchurch/6311895493/

Distributed System

A system which consists of two or more computers connected by networks




https://flickr.com/photos/facing-my-life/3038310109/i

“A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.”

- Leslie Lamport

Amazon Web Services' US-EAST-1 r trouble again, with EC2 and contain impacted

Internal dependencies again prove problematic

Simon Sharwood

Wed 29 Oct 2025 / 01:16 UT

UPDATED Amazon Web Services' US- disruption to online services, is having a prove problematic.

At 3:36 PM PDT on October 28 (10:36P that "Earlier today some EC2 launches experienced increased latencies for EC

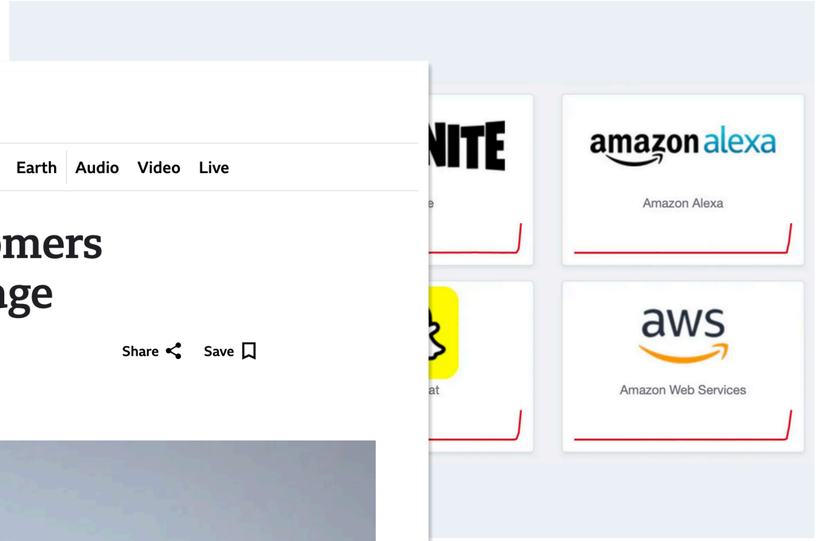
Amazon throttled some requests for EC resolve the issue.

Another impact of the incident created " Service] ECS tasks for both EC2 and F US-EAST-1 Region."

NEWS TECH AMAZON

Major AWS outage took down Fortnite, Alexa, Snapchat, and more

/ Amazon says th AWS outage was



the outage. Image: Downdetector

by Jess Weatherbed

Updated Oct 20, 2025, 6:41 PM GMT+2

130 Comments (All

d on creative industries, computing, and internet culture. Jess ews and hardware reviews.

BBC

Home News Sport Business Innovation Culture Arts Travel Earth Audio Video Live

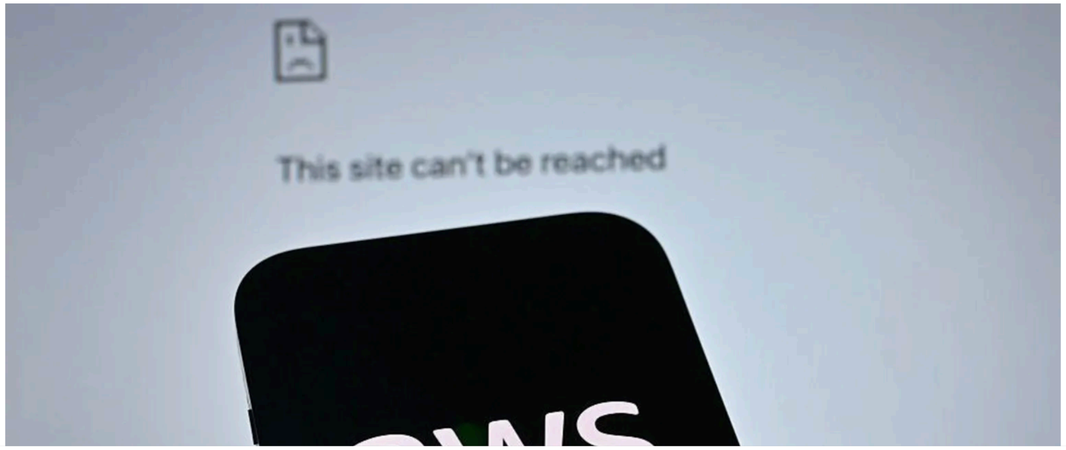
Amazon apologises to customers impacted by huge AWS outage

23 October 2025

Share Save

Liv McMahon

Technology reporter



NEWS TECH AMAZON

Eight Sleep adds ‘outage mode’ to smart beds after AWS problems left them frozen

Eight Sleep Pod mattress toppers can now be controlled offline in the event of future outages. Image: Eight Sleep

/ Thousands of E
owners were una
temperatures or
while while Amaz
were down.

by Jess Weatherbed
Oct 22, 2025, 5:28 PM GMT+2

48 Comments (All

https://www.theverge.com/news/804289/eight-sleep-smart-bed-aws-outage-overheating-offline

“A distributed system is one in which the failure of a computer you didn't even know existed can render your own ~~computer~~ mattress unusable.”

- Leslie Lamport

https://flickr.com/photos/spilt-milk/357016637/

THE THREE GOLDEN RULES OF DISTRIBUTED COMPUTING

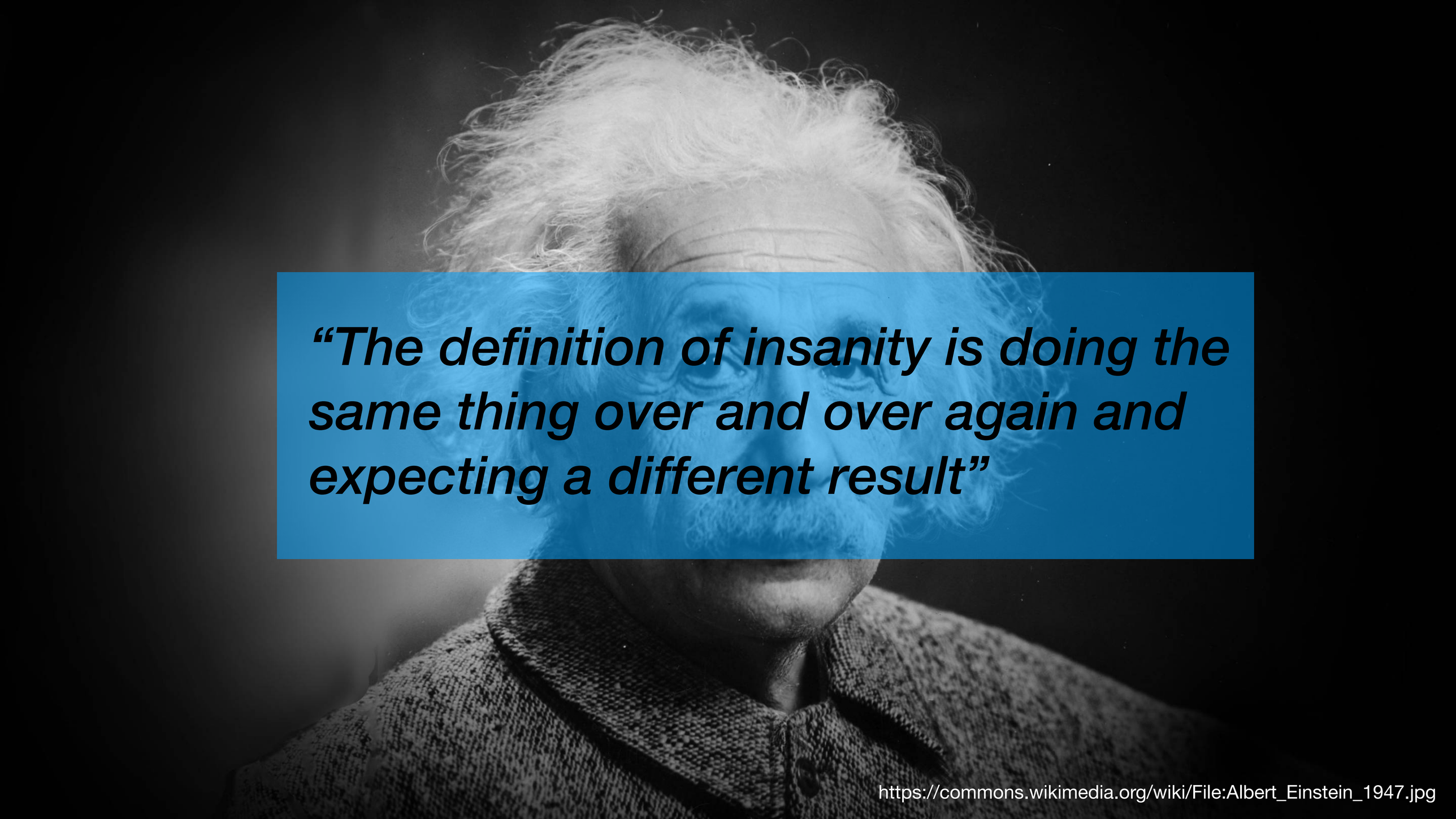
1. You can’t beam information between two points instantly

2. Sometimes, you can’t reach the thing you want to talk to

3. Resources are not infinite

Retries

https://flickr.com/photos/mn1705/14907413228/



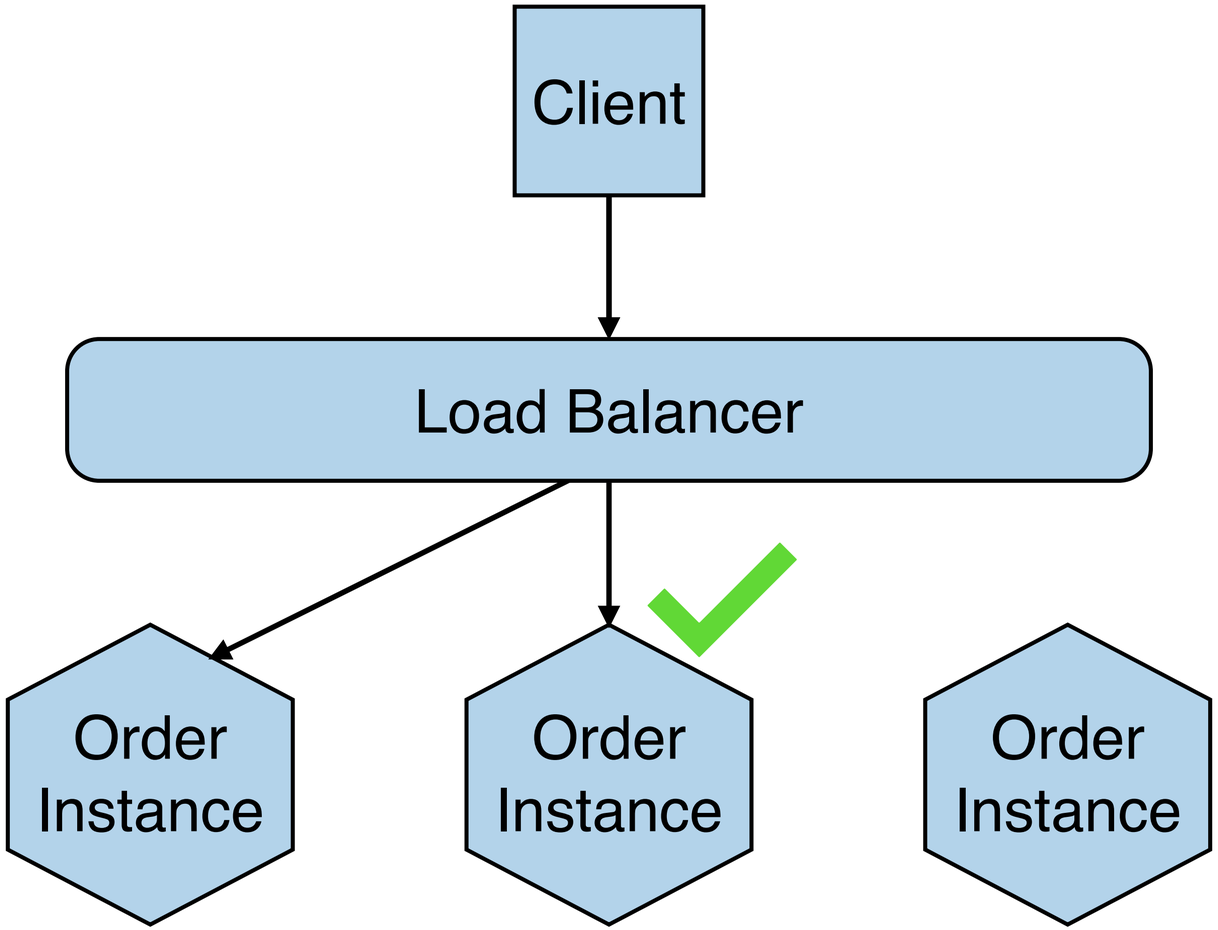
https://commons.wikimedia.org/wiki/File:Albert_Einstein_1947.jpg

THE THREE GOLDEN RULES OF DISTRIBUTED COMPUTING

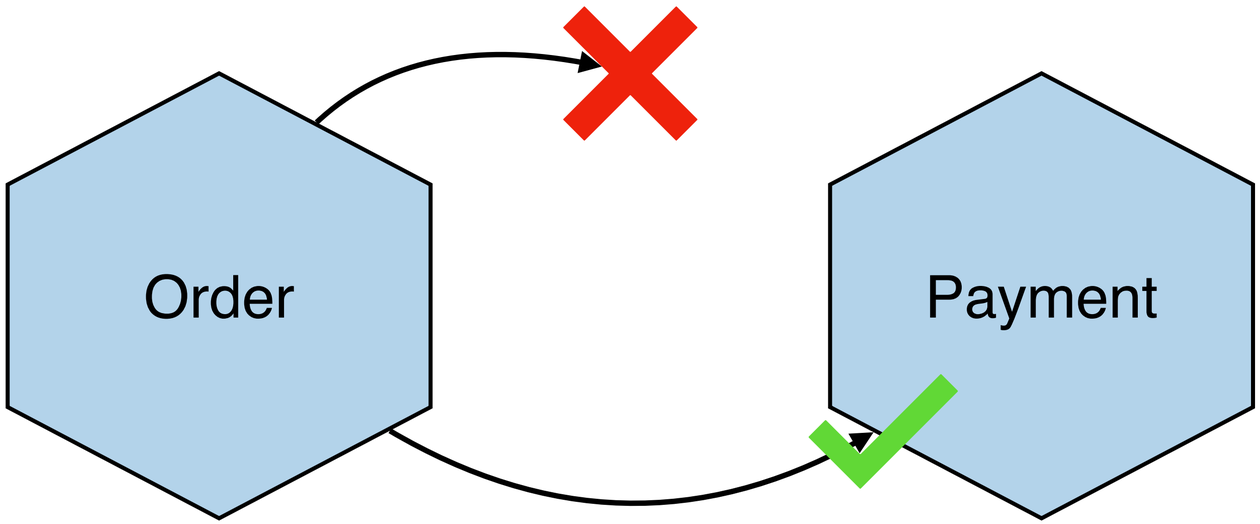
1. You can’t beam information between two points instantly

2. Sometimes, you can’t reach the thing you want to talk to

3. Resources are not infinite


Client
Load Balancer
Order Instance
Order Instance
Order Instance

So if the first attempt fails…



It might be worth trying again

How many retries?

THE THREE GOLDEN RULES OF DISTRIBUTED COMPUTING

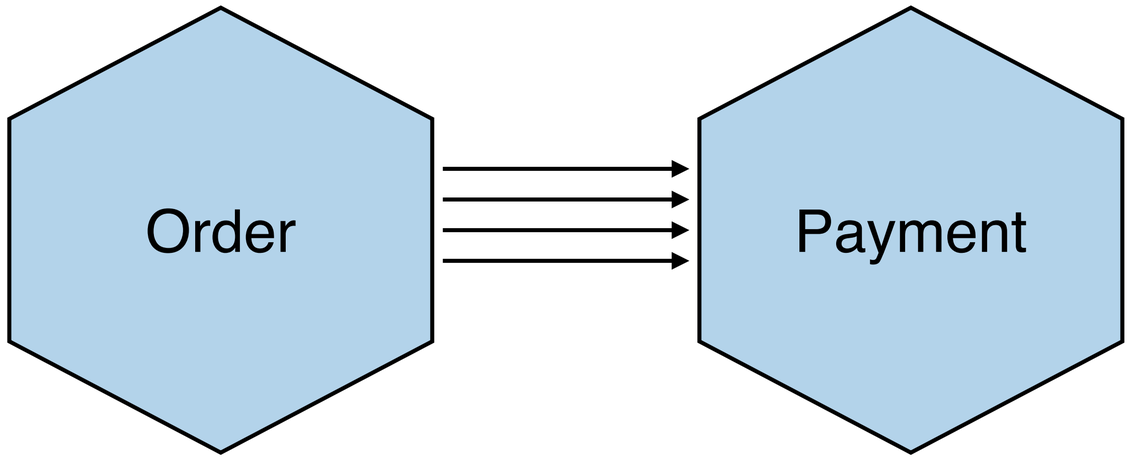
1. You can’t beam information between two points instantly

2. Sometimes, you can’t reach the thing you want to talk to

3. Resources are not infinite

Retrying too often can overload a server

Setting a max number of retries makes sense

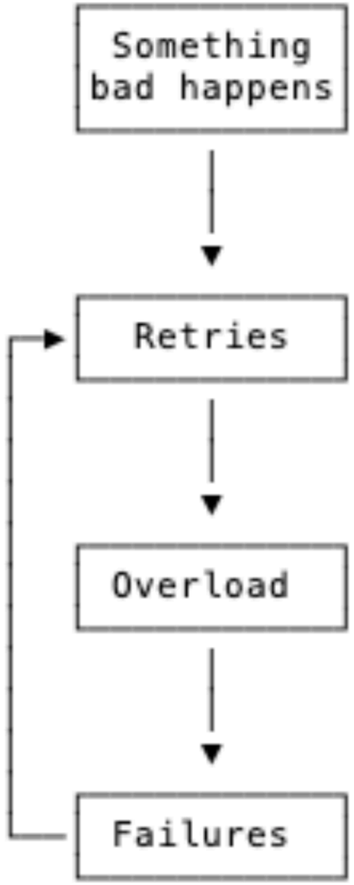


And load shedding is a sensible defence here

Will circuit breakers solve my problems?

Maybe, but you need to know what problem you're trying to solve first.

A couple of weeks ago, I started a tiny storm on Twitter by posting this image, and claiming that retries (mostly) make things worse in real-world distributed systems.

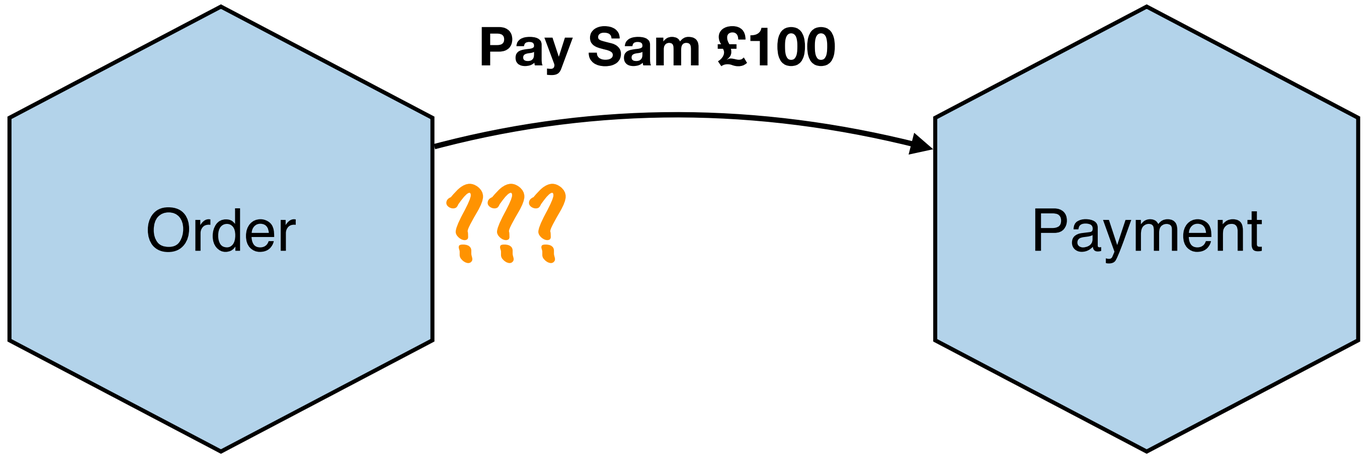


https://brooker.co.za/blog/2022/02/16/circuit-breakers.html

https://medium.com/square-corner-blog/incident-summary-2017-03-16-2f65be39297

Is it safe to retry?

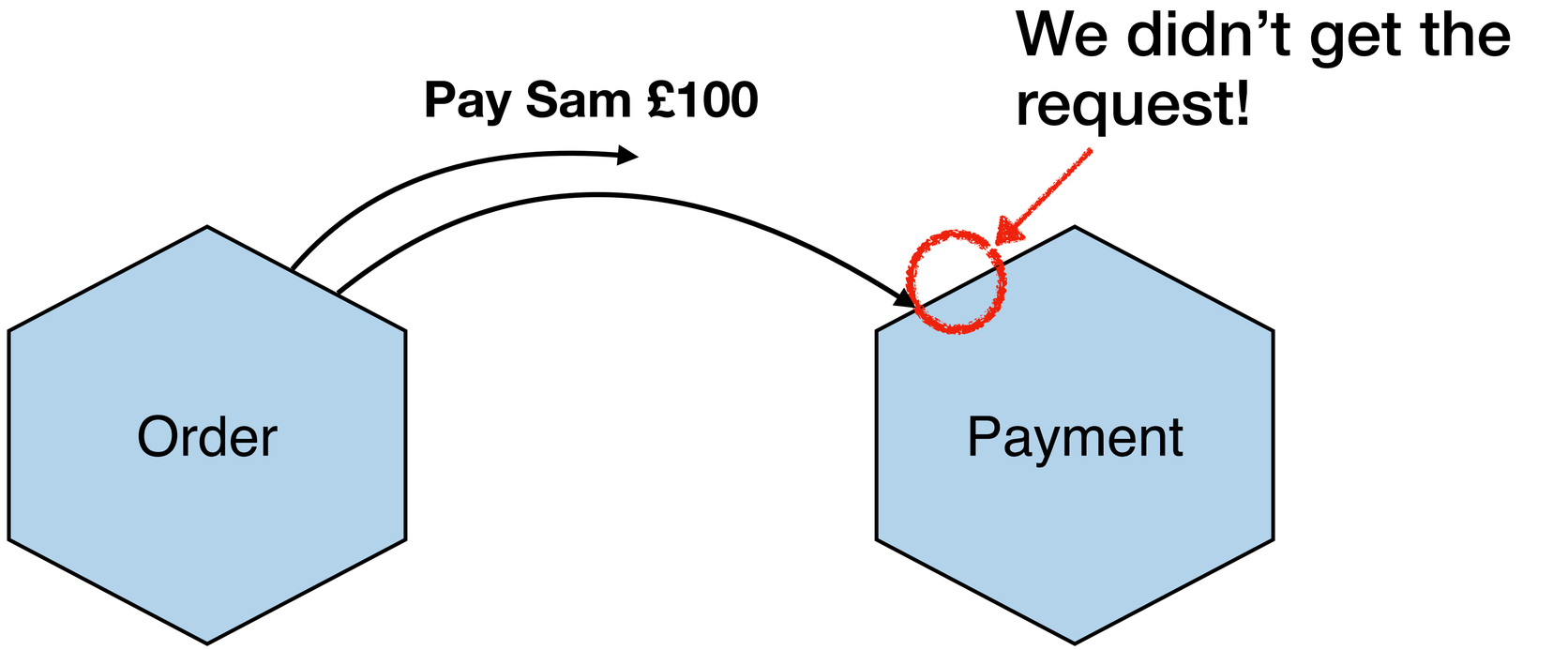
What happens if we don’t get a response?



Two possibilities…

First Possibility

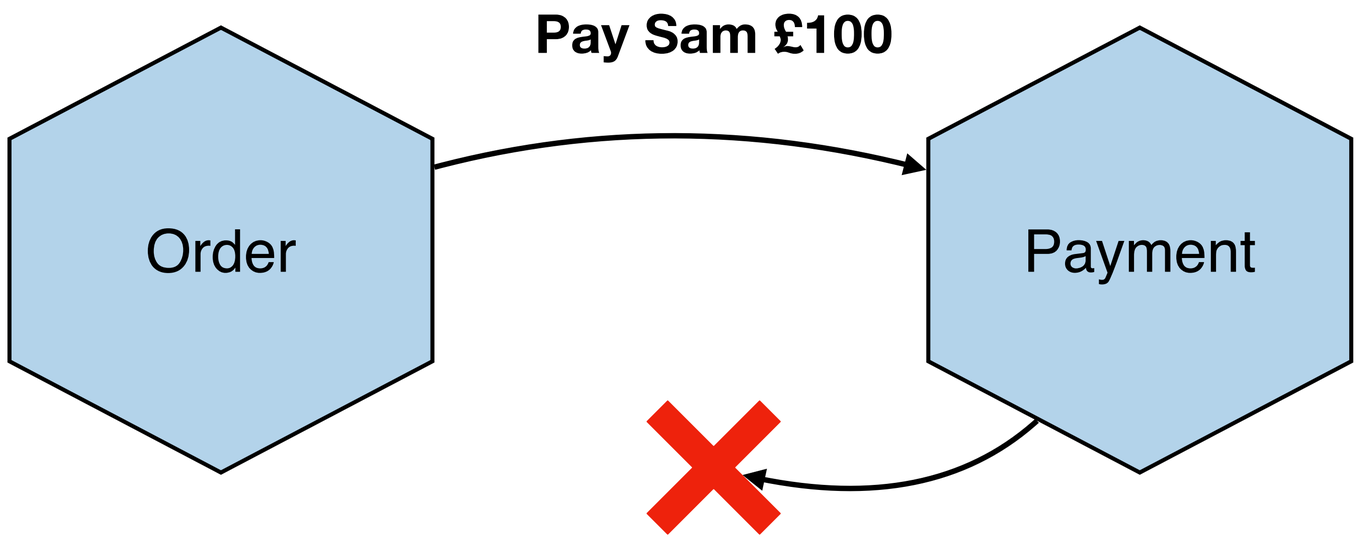
The payment wasn't carried out



Or we did, but crashed during processing…

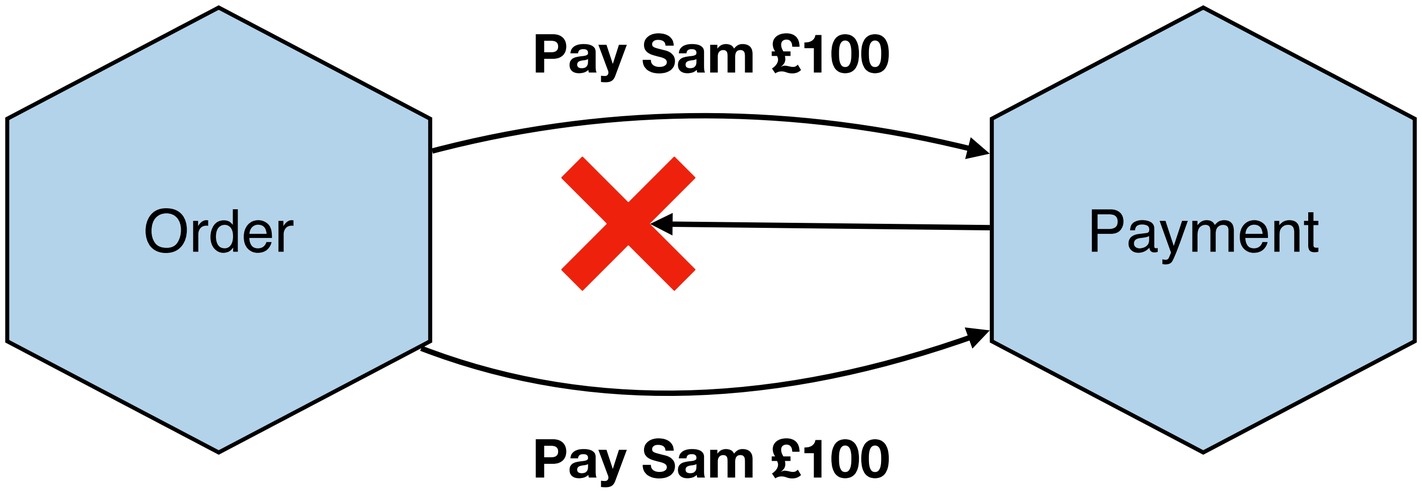
Second Possibility

The payment was performed, but the response wasn't processed




https://flickr.com/photos/ftzdomino/6304930642/

The new problem? What if that means Sam gets paid twice?



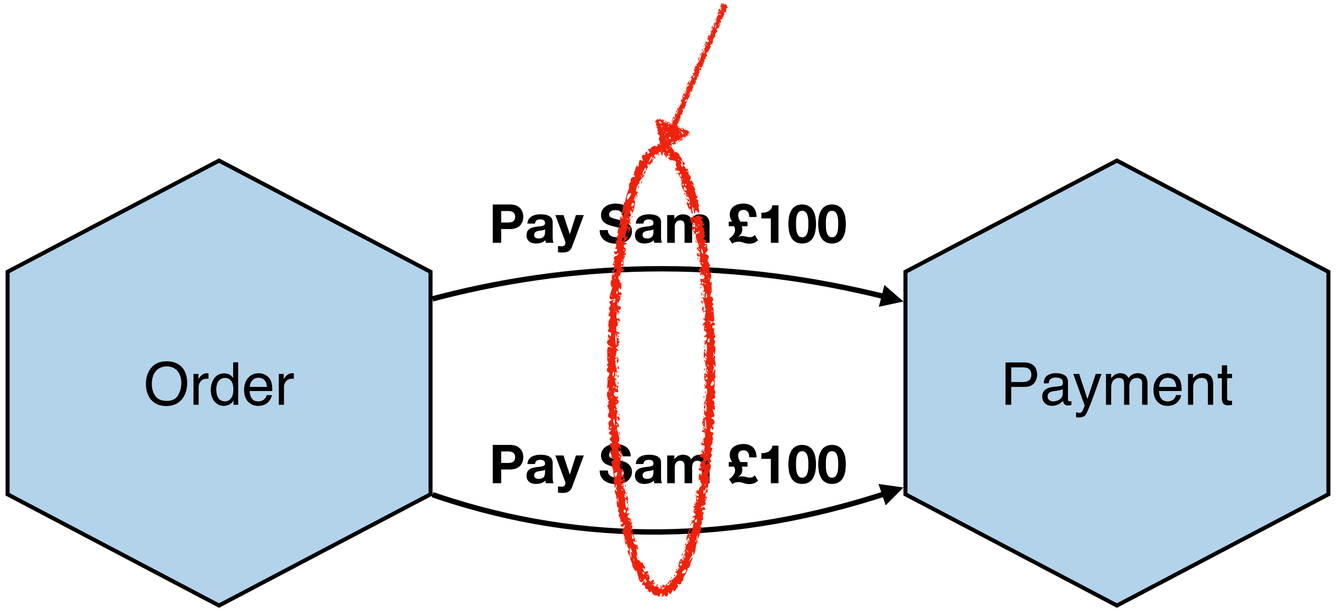
Idempotency

https://flickr.com/photos/skuds/3781599677/

Idempotent Operation

An operation that can be applied multiple times without changing the result

This operation is NOT idempotent - performing the same operation twice does not give the same result



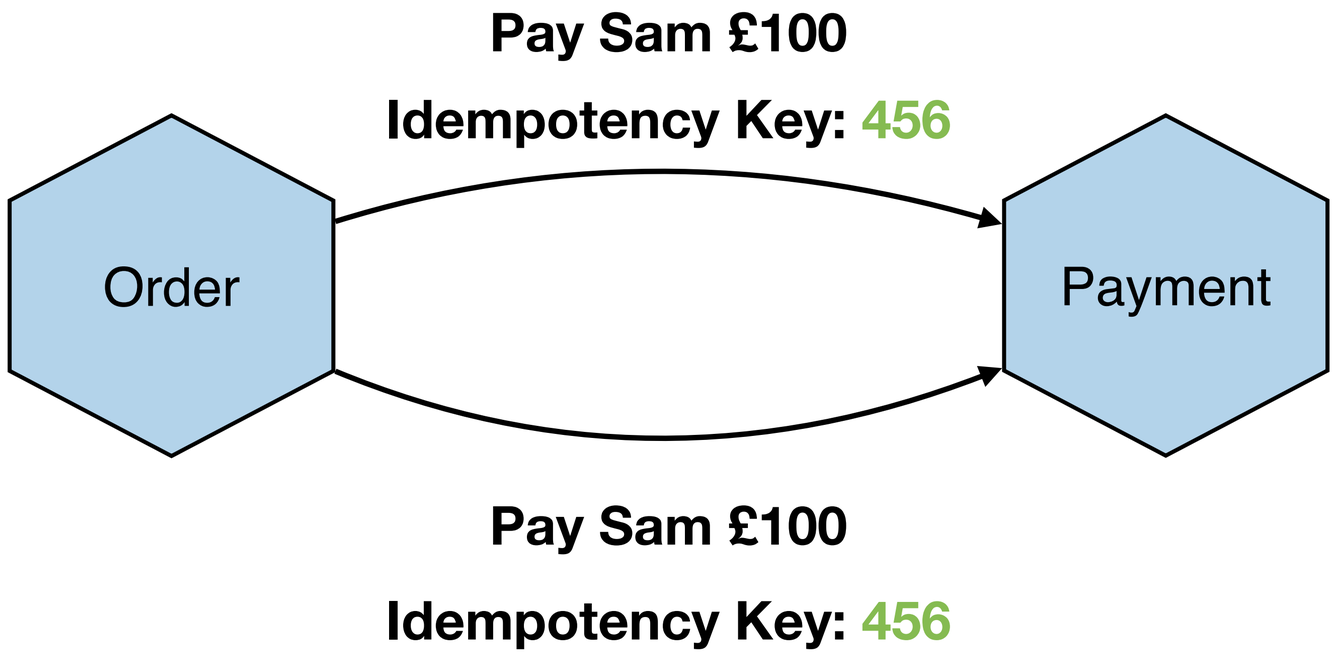
Idempotency Keys

Server Side Request Fingerprinting

Idempotency Key

A unique ID used to identify a specific operation

Payment can now see this payment has already been made

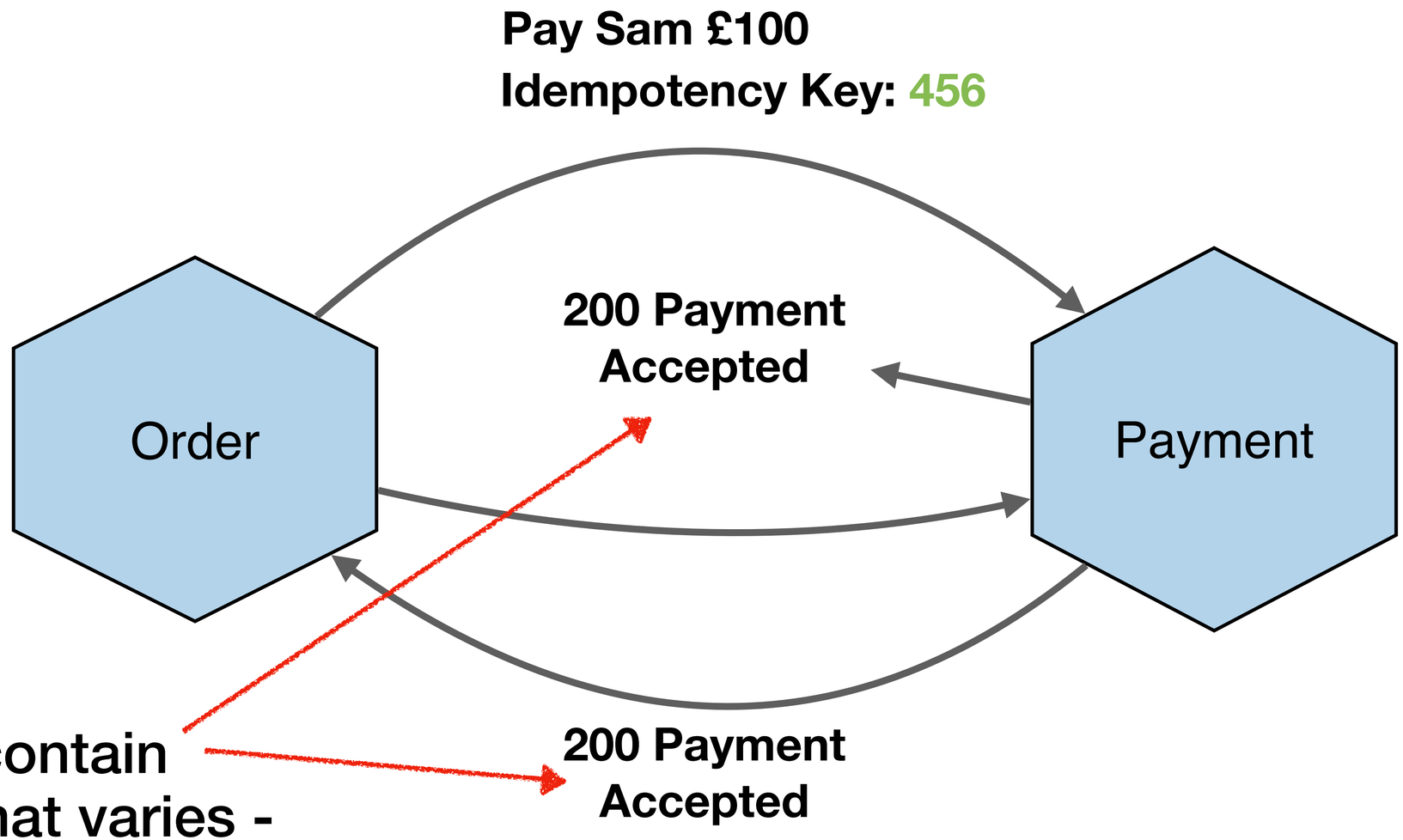


Processed Payments

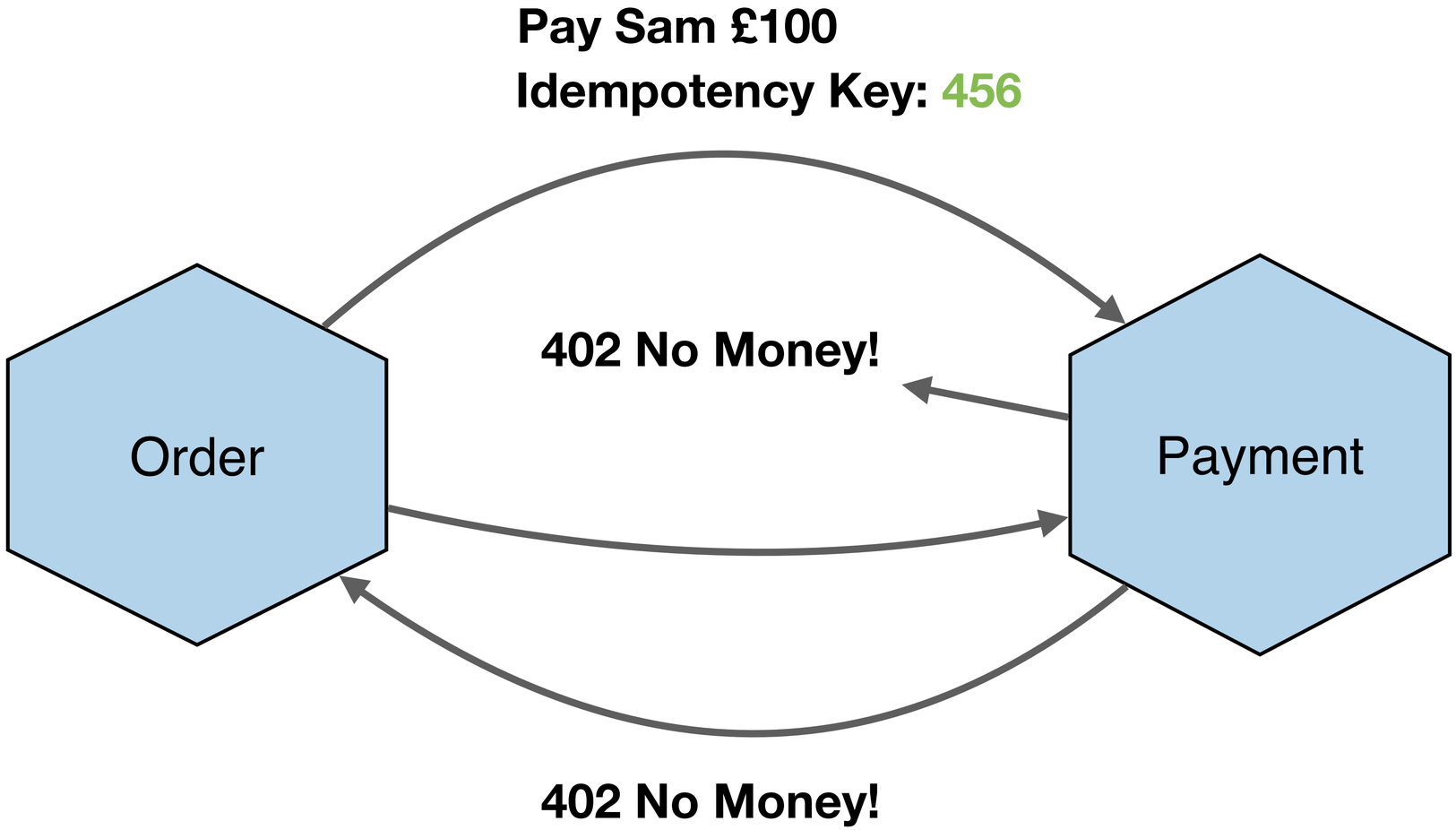
Key	Result
456	200 Success

What response should be sent when ignoring the retried operation?

Send the same (semantic) response as the original request would have been sent



Could still contain metadata that varies - ideally kept in headers


Pay Sam £100
Idempotency Key: 456
Order
402 No Money!
Payment
402 No Money!

Idempotency Keys IDs require a change in the client server protocol. What if that can’t easily be changed?

Server Side Request Fingerprinting

Generating a fingerprint of the operation on the server side to spot duplicate requests

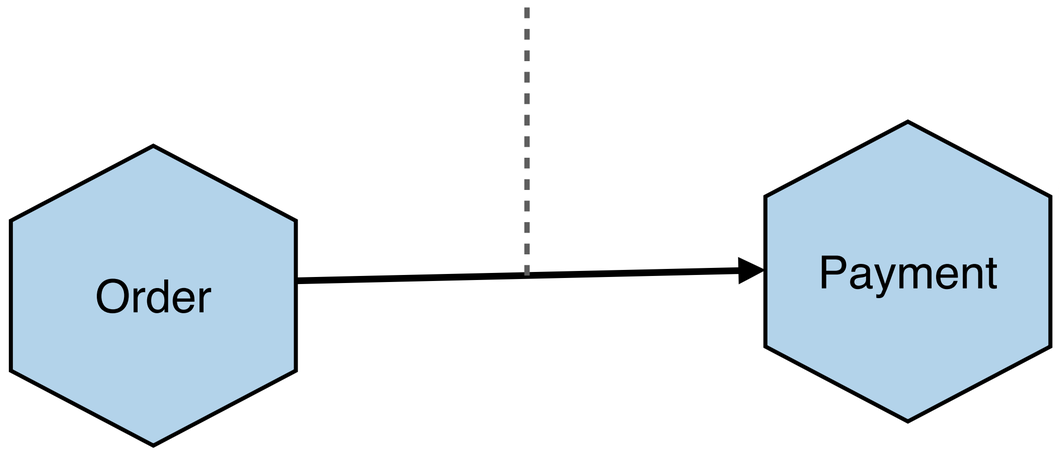
Header

```
PUT /payment HTTP/1.1
Host: payment.mycompany.com
…
```

Body

```
customer-id: 123
payment-amount: 100
current: GBP
```

= **MD5 Hash** 92a22cca1d3b2…



Processed Payments

Fingerprint	Result
040cfd437…	402 Failed
92a22cca…	200 Success

```
PUT /payment HTTP/1.1
Host:
payment.mycompany.com

customer-id: 123
payment-amount: 100
current: GBP
timestamp: 1728468810
```

MD5 Hash
040cfd437dea56d…

```
PUT /payment HTTP/1.1
Host:
payment.mycompany.com

customer-id: 123
payment-amount: 100
current: GBP
timestamp: 1728468853
```

MD5 Hash
997d0bb02e23f2a…

Option 1

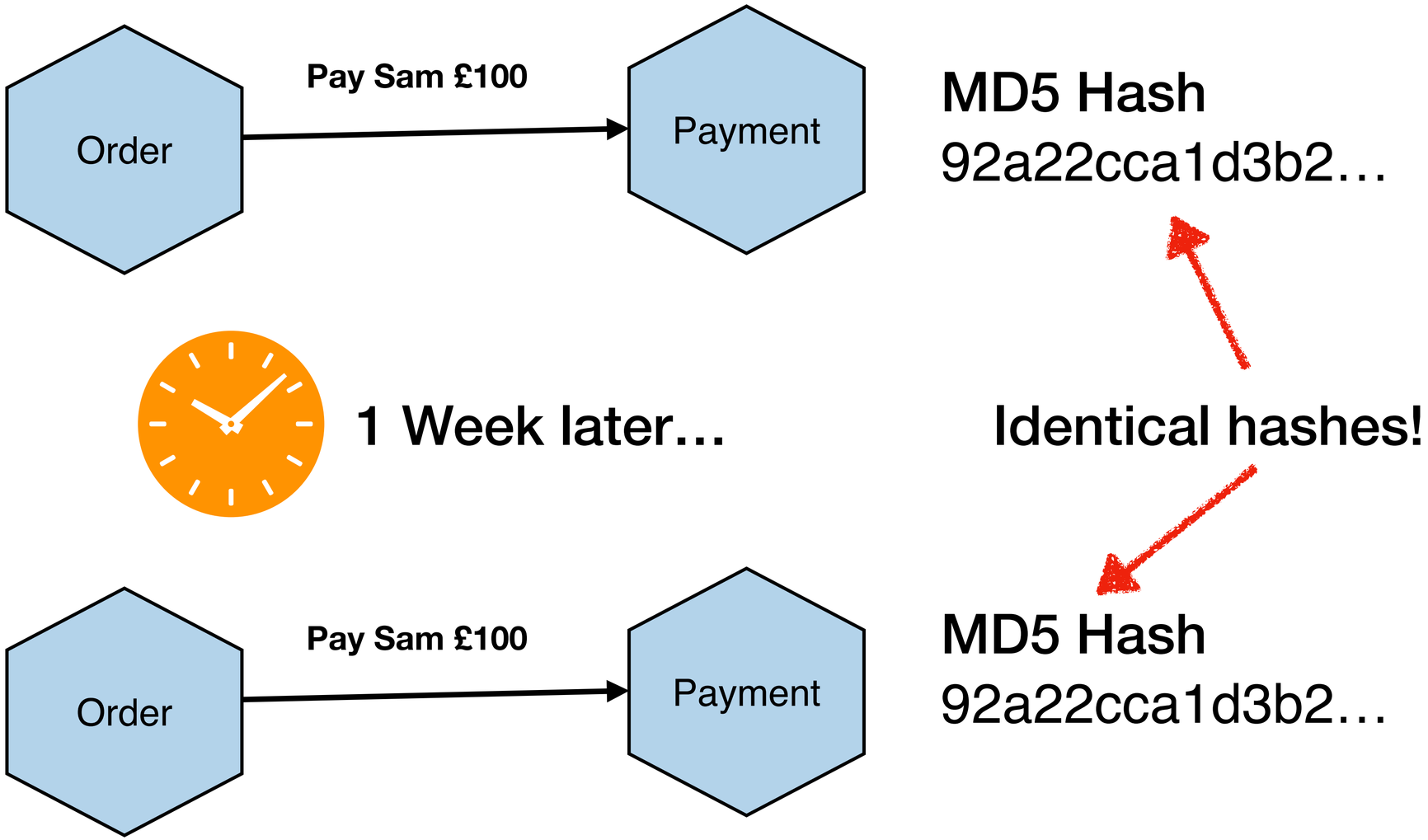
Only use part of the request body for fingerprinting

```
PUT /payment HTTP/1.1
Host: payment.mycompany.com

customer-id: 123
payment-amount: 100
current: GBP

timestamp: 1728468810
```

Option 2

Move fields to the header

```
PUT /payment HTTP/1.1
Host: payment.mycompany.com
timestamp: 1728468810

customer-id: 123
payment-amount: 100
current: GBP
timestamp: 1728468810
```


Order
Pay Sam £100
Payment
MD5 Hash
92a22cca1d3b2…
1 Week later…
Identical hashes!
Order
Pay Sam £100
Payment
MD5 Hash
92a22cca1d3b2…

Fingerprint	Result	Expiry
92a22cca1…	Success	12:01:03
997d0bb0…	Failed	12:01:17
040cfd437…	Success	12:03:45

Getting retention right when using fingerprints is very important!

https://flickr.com/photos/yusamoilov/13639280834/

Can still be a problem if you want to do the same thing multiple times in a short space of time

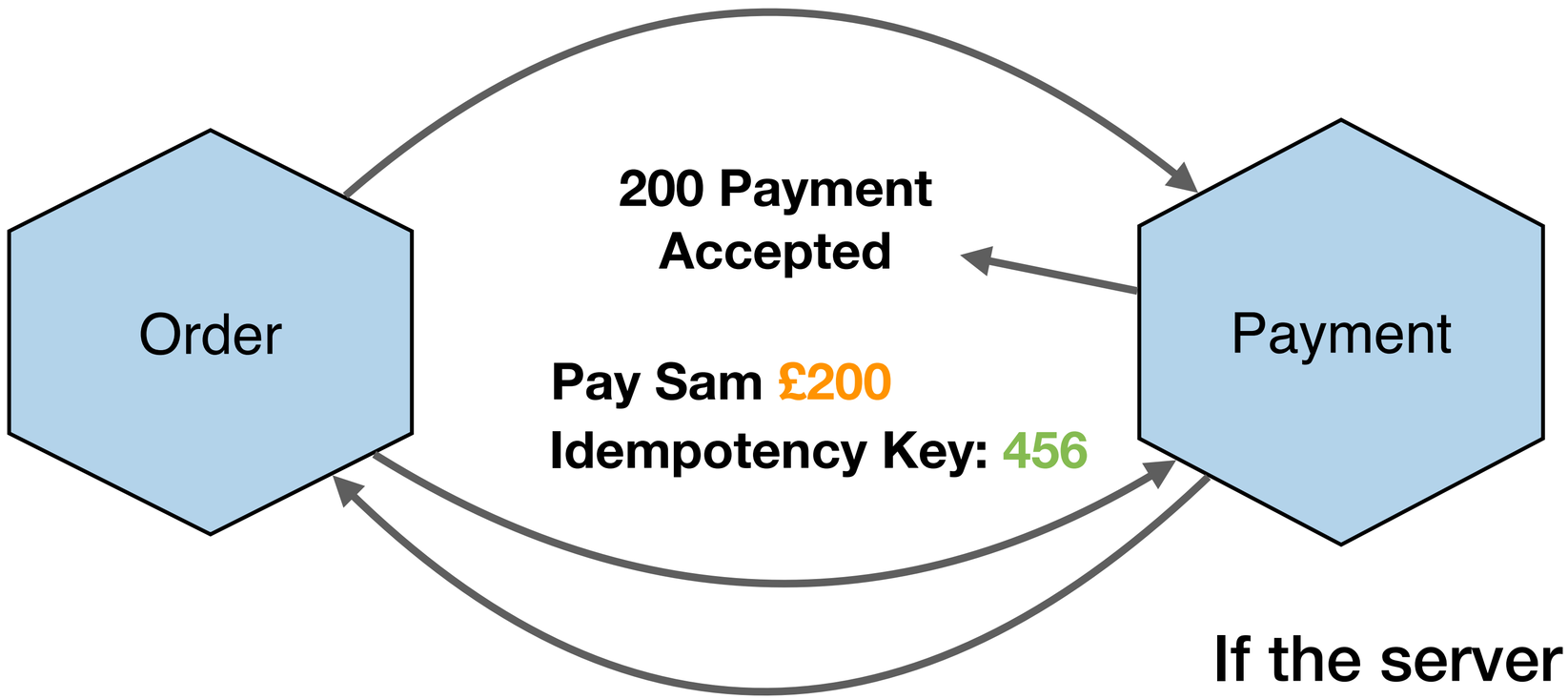
https://www.flickr.com/photos/ibm_media/33838065805/

If you can change the client-server protocol, then idempotency keys are the easiest way to make an operation idempotent

Otherwise consider server side request fingerprinting, but be aware of the downsides

Using *idempotency keys* and *server side request fingerprinting* together?

Can be useful to catch client errors



Resending “Payment accepted” here could confuse the client!

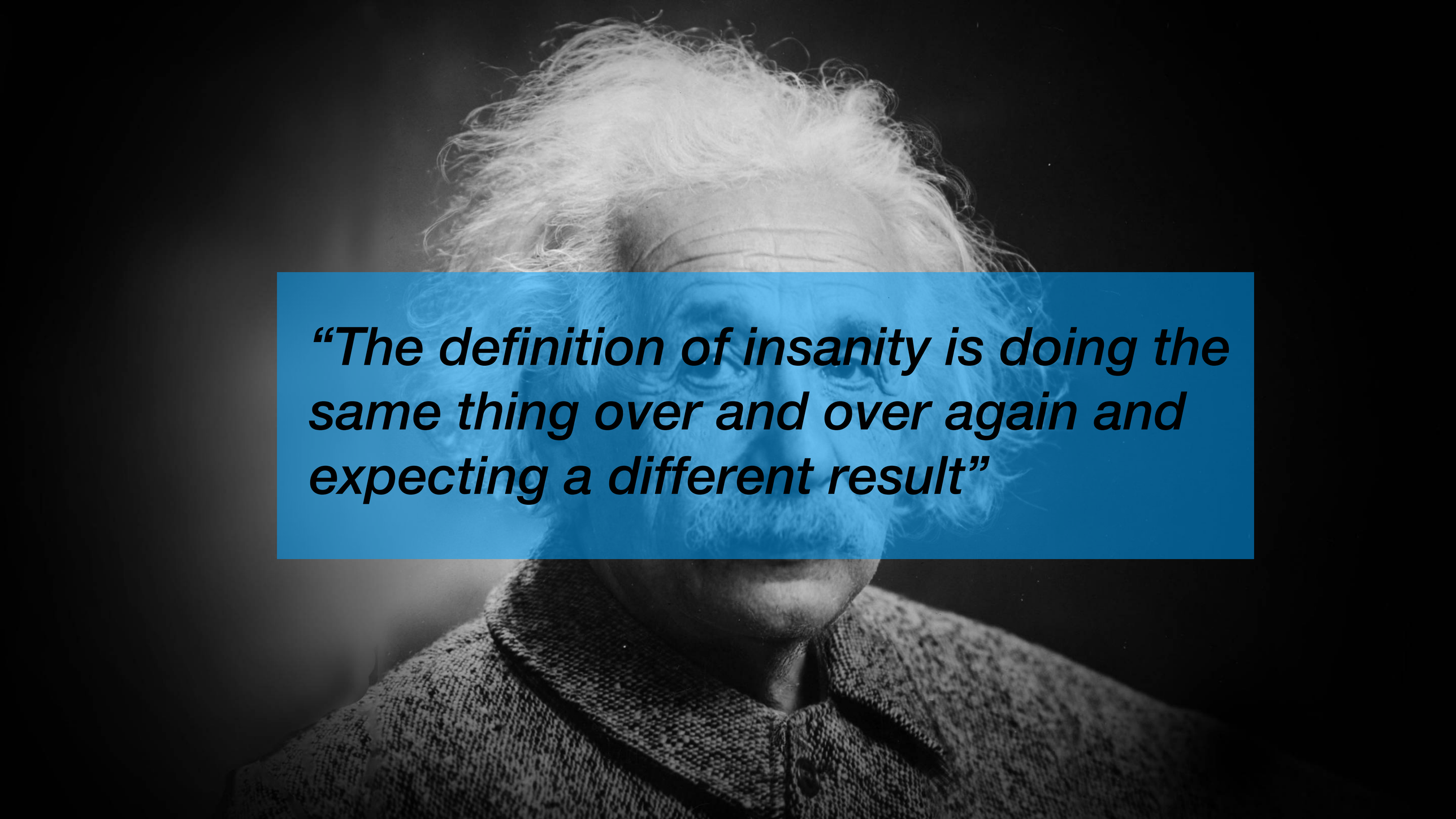
If the server sees a retry with changing values, sending an error can be more clear

Key	Fingerprint	Result	Expiry
456	92a22cca1…	Success	12:01:03
789	997d0bb0…	Failed	12:01:17
123	040cfd437…	Success	12:03:45

Just store the key and the fingerprint together

So doing the same thing over and over again might be sensible…

…but only if you can do so safely


“The definition of insanity is doing the same thing over and over again and expecting a different result”

Einstein didn’t say this quote

THE THREE GOLDEN RULES OF DISTRIBUTED COMPUTING

1. You can’t beam information between two points instantly

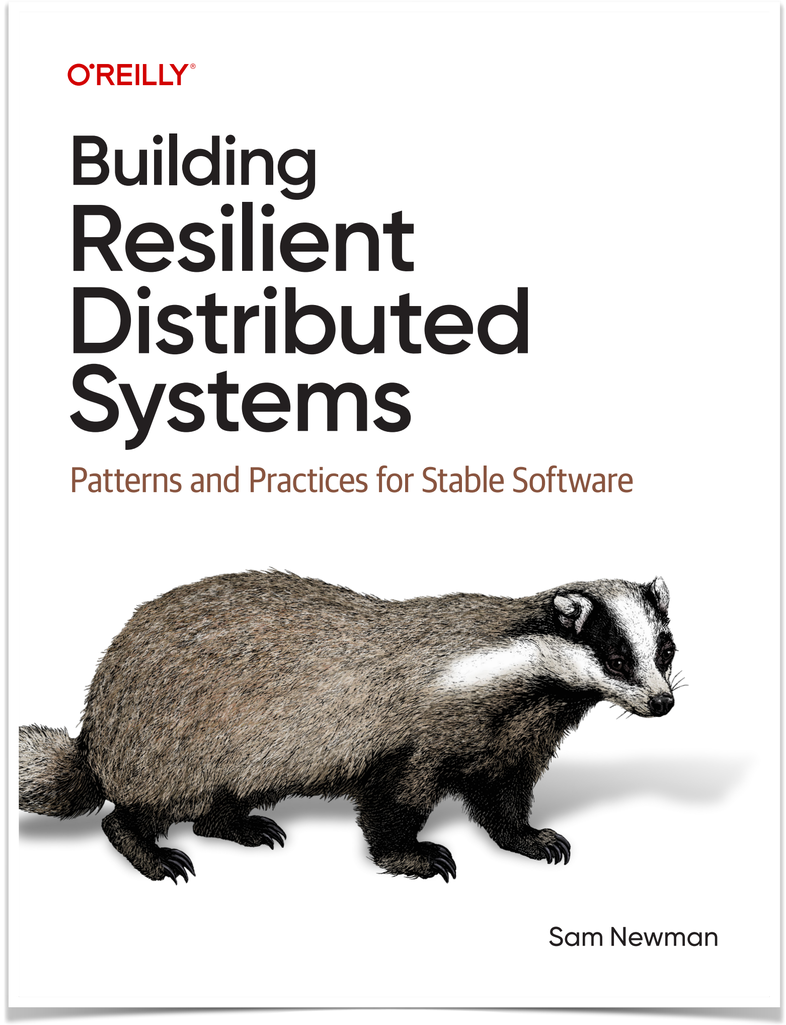
2. Sometimes, you can’t reach the thing you want to talk to

3. Resources are not infinite

If at first you don’t succeed, retrying can make sense…to a point!

If you do want to retry, making your operations idempotent is vital

NEW BOOK IN EARLY ACCESS!



Book details and slides

https://samnewman.io/books/building-resilient-distributed-systems/