Building & trusting a cloud bank

@gmorpheme #qconlondon

STARLING BANK

18th June 2012

19th June 2012

10th July 2012

we can rebuild the bank in an hour

we can rebuild the bank in an hour

job done

we can rebuild the bank in an hour

job done(*)

* terms and conditions apply

we could rebuild in an hour but…

- …the problem might not be us
- …there might be some phone calls
- …only in AWS
- …from recent backups
- *…only if we make the decision to do it*

Starling Bank today

Full UK Current Account

iOS & Android Apps

Debit Card

Faster Payments & DDs

Overdrafts

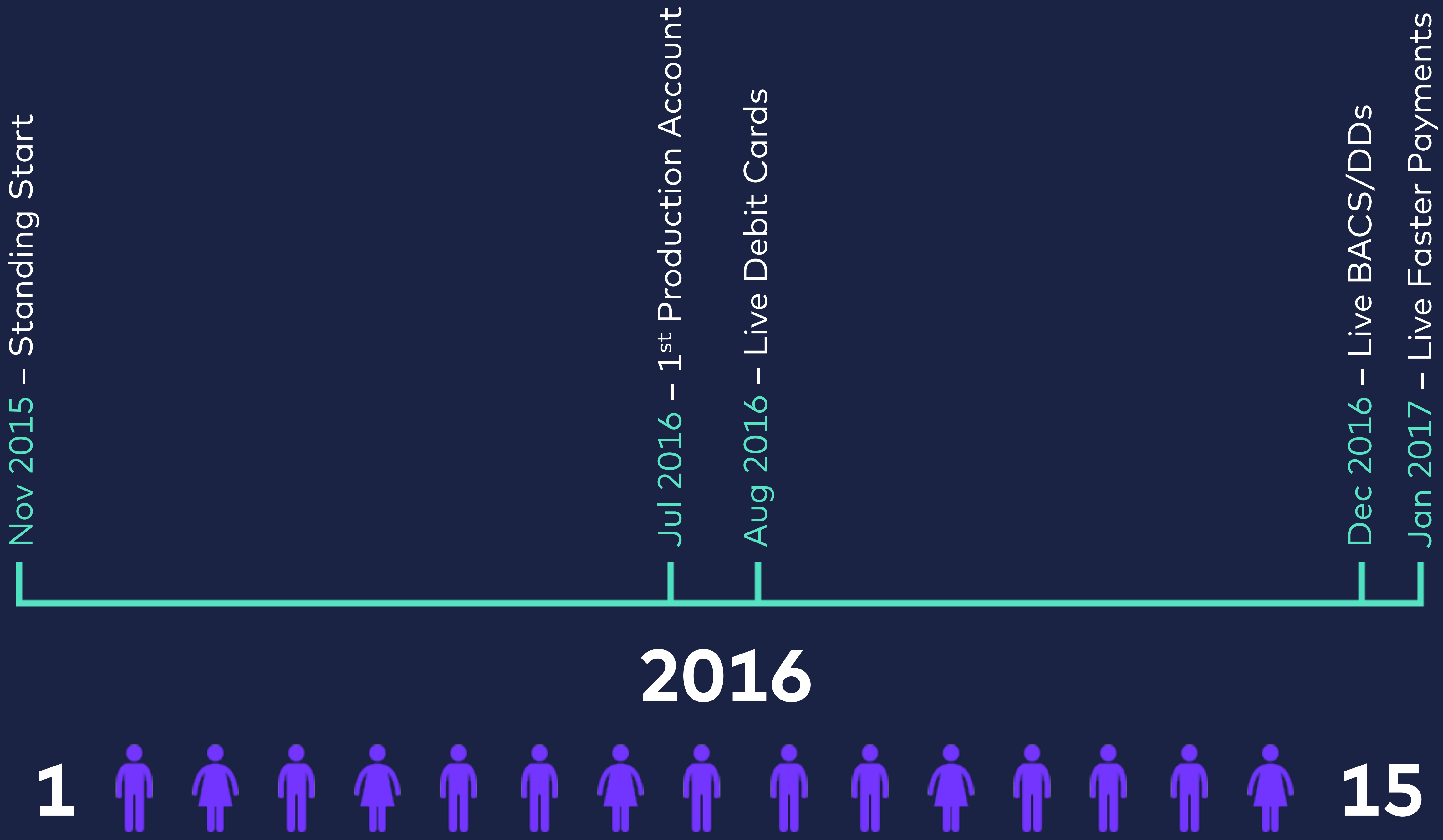
In-app Support

Open API

Closed Beta (100s Accs)

Core Ledger

Credit, AML, KYC, Fraud Processes


Nov 2015 – Standing Start
Jul 2016 – 1st Production Account
Aug 2016 – Live Debit Cards
Dec 2016 – Live BACS/DDs
Jan 2017 – Live Faster Payments
2016
1
15

security > resilience > scale

velocity > economy

architect for change

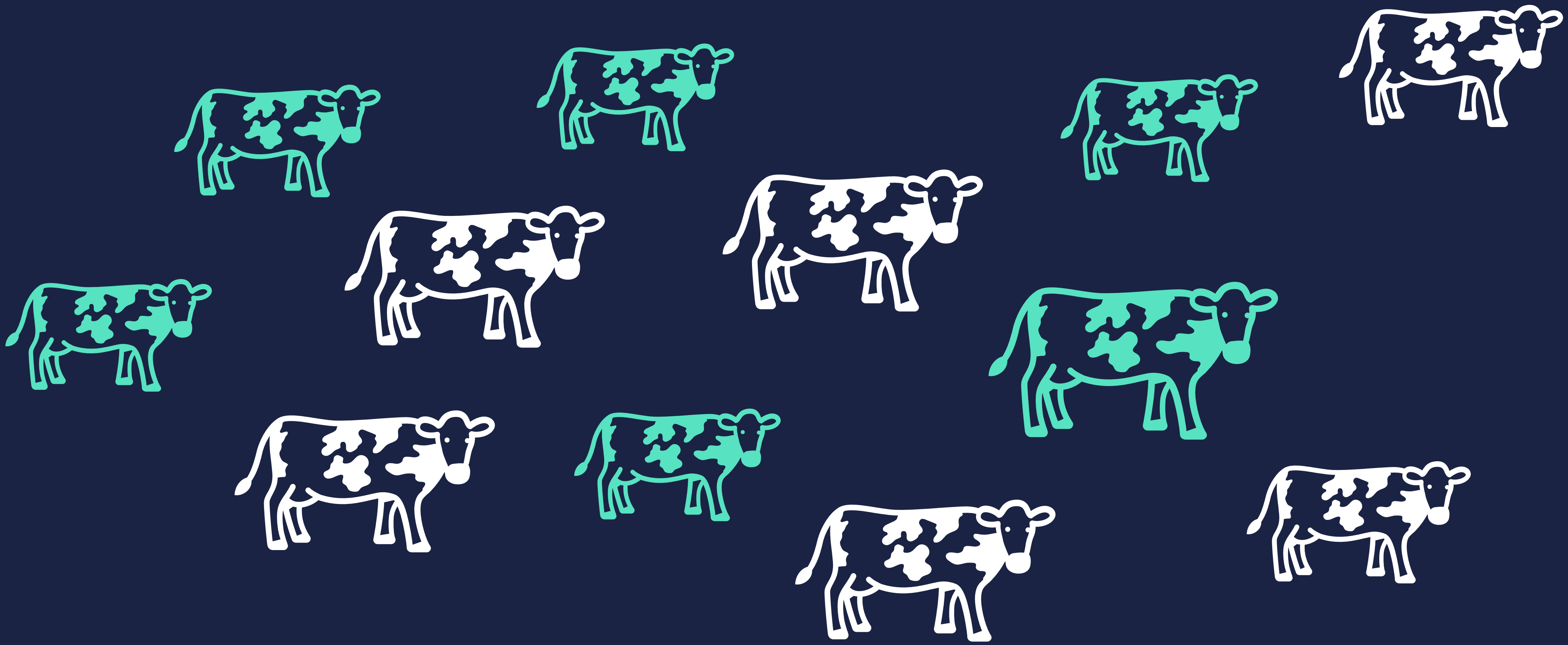
embrace cloud

three key categories of failure

- errors that correlate by infrastructure
- errors that correlate by function
- errors that emerge in complex systems under load

- we'll talk about the first two

immutable infrastructure



instance termination is safe

- single stateless service per instance
- if ever a server is in doubtful state, kill it
  - pen testing?
  - chaos experiments?
  - suspicious activity?
- chat-ops slack bot
  - starbot **kill**
- rolling deployments by termination (not quick but safe)
  - starbot **recycle**
  - starbot **reboot**

…everywhere

- *everything* in our core infrastructure is either
  - immutable service in EC2
  - data in a managed service
- no large infrastructure pet
  - no "clusters"
  - no state in EC2
  - no EBS volumes to manage
  - no shared caches
  - no external queues
  - no orchestration engines
  - (yet!)

a Starling service

- simple AWS approach
- ELB / ASG / RDS across 3 AZs in eu-west-1
- “service discovery” is just DNS
- service is docker as systemd unit on CoreOS
- all specified in CloudFormation (!)
- with config and versions in S3

impact of instance outage

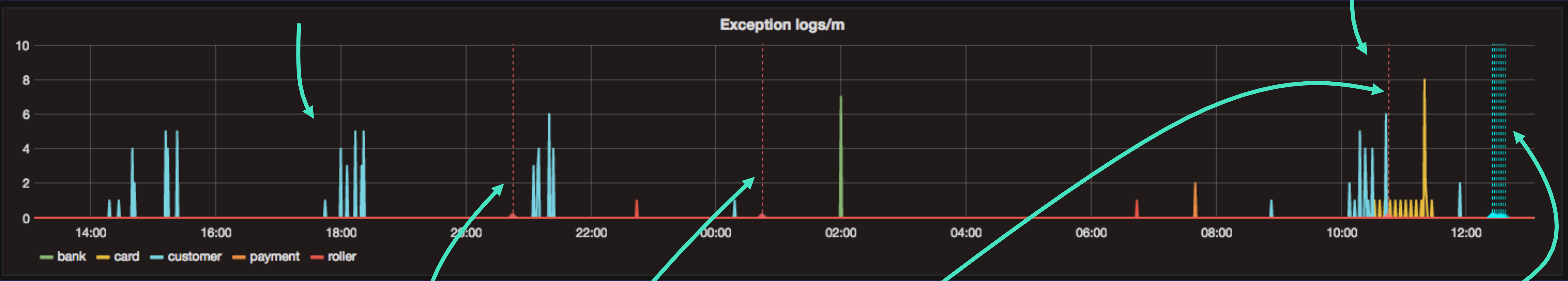
- 2x (5s interval + 2s timeout) = max **14s** to drop out of ELB
- some 504s then 5m of reduced capacity

- 14s when 1 of our ~10 services is partially degraded at least one bank has an iOS app that takes ~14s to start

production chaos

- we know we're resilient because we kill servers all the time

importance of noise-free steady state



errors

errors

(NO CUSTOMER IMPACT)

chaos

rolling release

artificial load in production

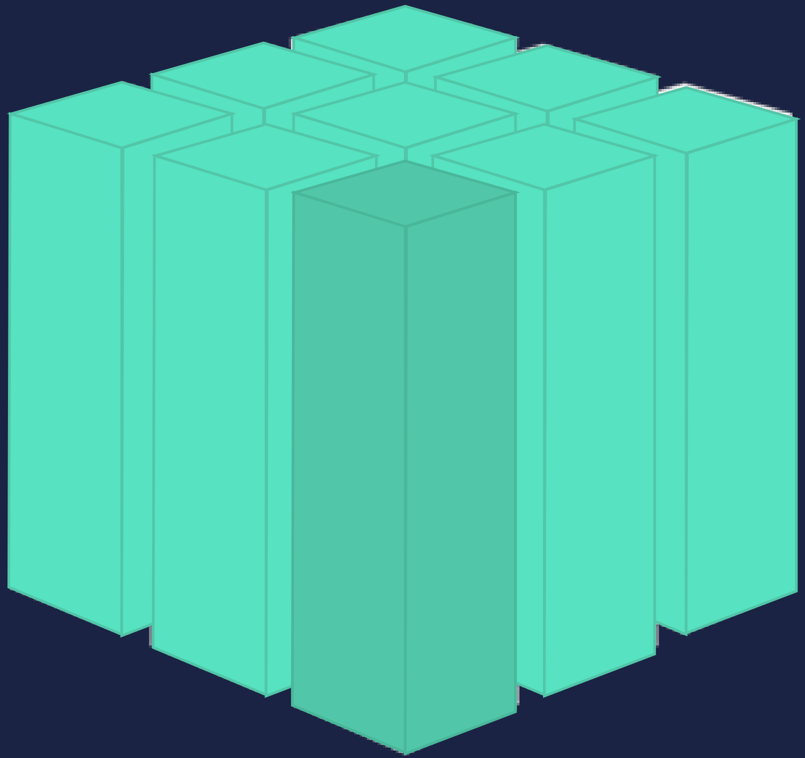
- monitoring and control are difficult without volume
- we deploy a "simulator" service in production
- generates synthetic transactions
- e.g. 160,000 card authorisations a day
- continual assurance on available headroom
- interruptions are obvious
- all servers are naturally warmed up
- synthetic transactions *are* difficult

impact of AZ / region outage

- AZ loss => ASGs and ELBs rebalance
- region loss (EC2/RDS) => rebuild

- S3 outage
  - lose some message archiving
  - new instances fail retrieving config (easy fix)

self-contained systems



http://scs-architecture.org

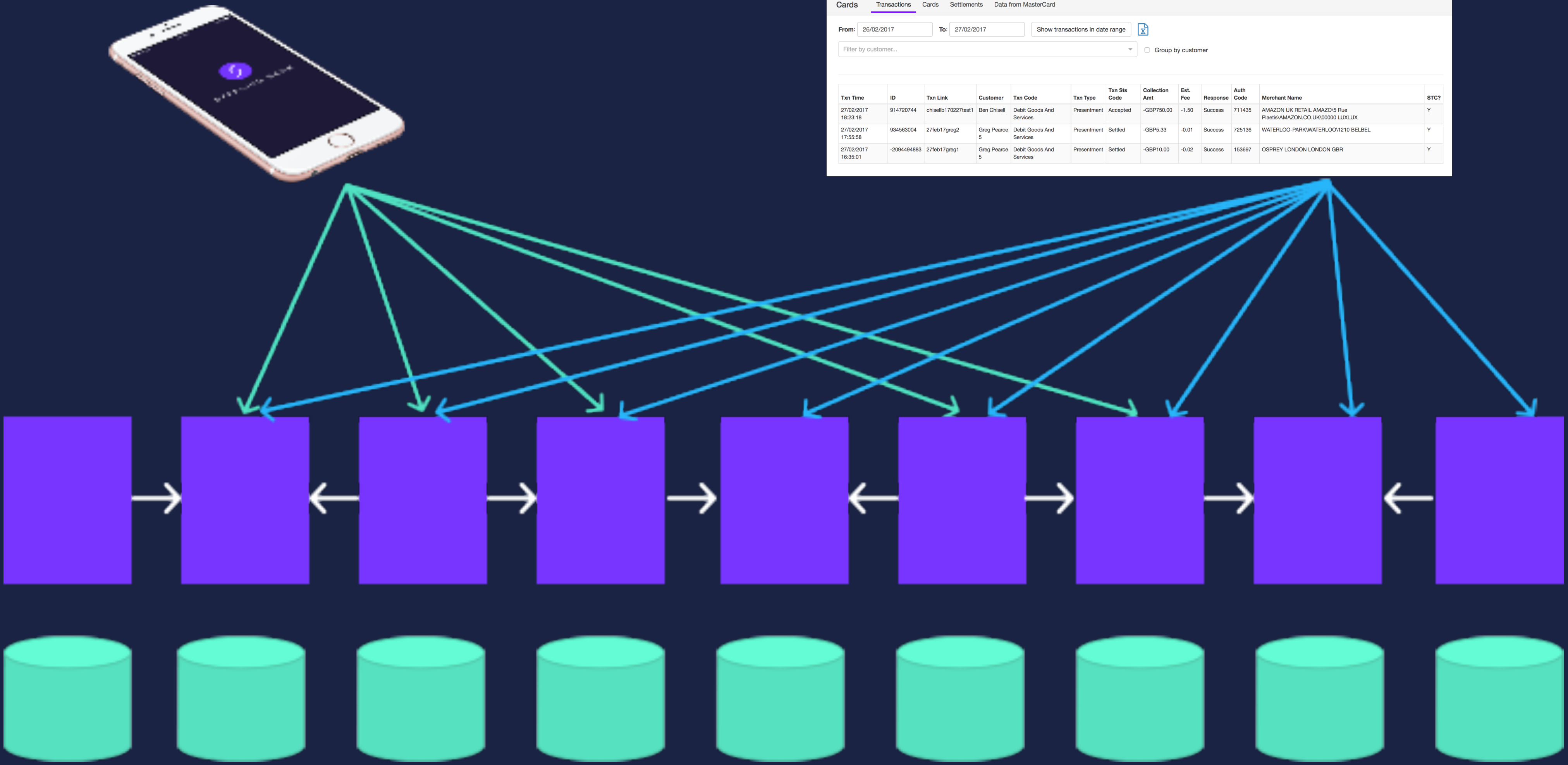
Starling as self-contained systems

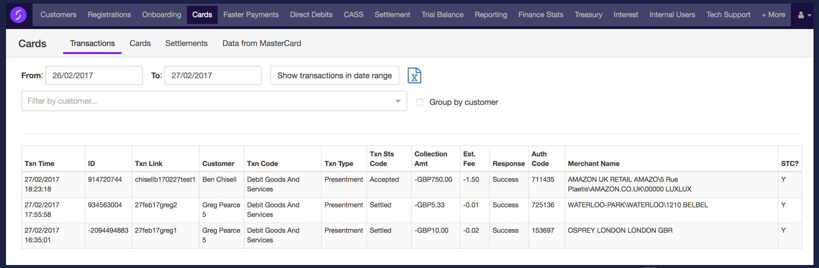
- all services have their own RDS instance
- inter-service comms is generally async
- mobile layer integrates data from different services
- no start-up order dependencies

not pure SCS

- we're mobile-first (and API-first!) – web is secondary
- services not owned by single team
- our services have REST APIs but no internal web UI
    - internal (inter-service)
    - external (mobile)
    - management (web console)
    - operations (health check etc.)
- one key area with sync interaction (balance allocation)

each service exposes distinct APIs





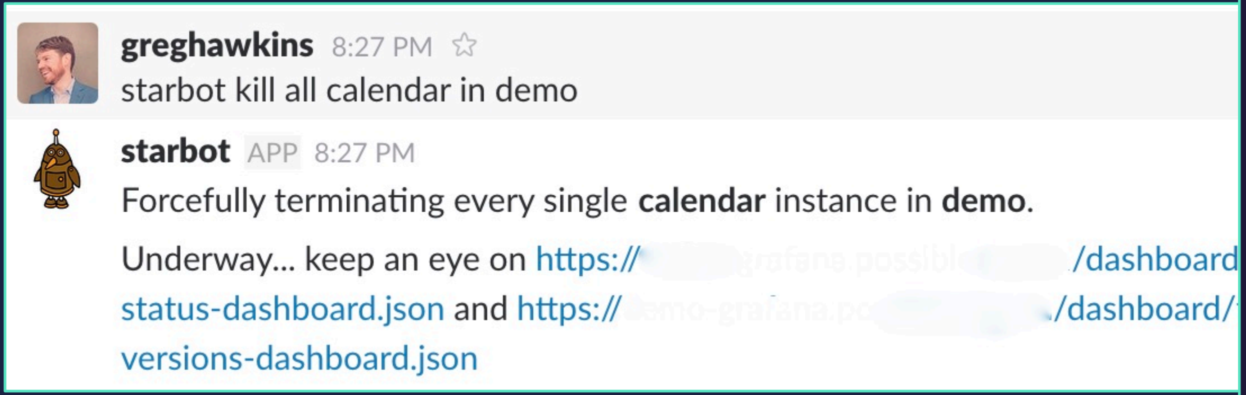
From: 26/02/2017 To: 27/02/2017

☐ Group by customer

Txn Time	ID	Txn Link	Customer	Txn Code	Txn Type	Txn Sts Code	Collection Amt	Est. Fee	Response	Auth Code	Merchant Name	STC?
27/02/2017 18:23:18	914720744	chisellb170227test1	Ben Chisell	Debit Goods And Services	Presentment	Accepted	-GBP750.00	-1.50	Success	711435	AMAZON UK RETAIL AMAZO\5 Rue Plaetis\AMAZON.CO.UK\00000 LUXLUX	Y
27/02/2017 17:55:58	934563004	27feb17greg2	Greg Pearce 5	Debit Goods And Services	Presentment	Settled	-GBP5.33	-0.01	Success	725136	WATERLOO-PARK\WATERLOO\1210 BELBEL	Y
27/02/2017 16:35:01	-2094494883	27feb17greg1	Greg Pearce 5	Debit Goods And Services	Presentment	Settled	-GBP10.00	-0.02	Success	153697	OSPREY LONDON LONDON GBR	Y

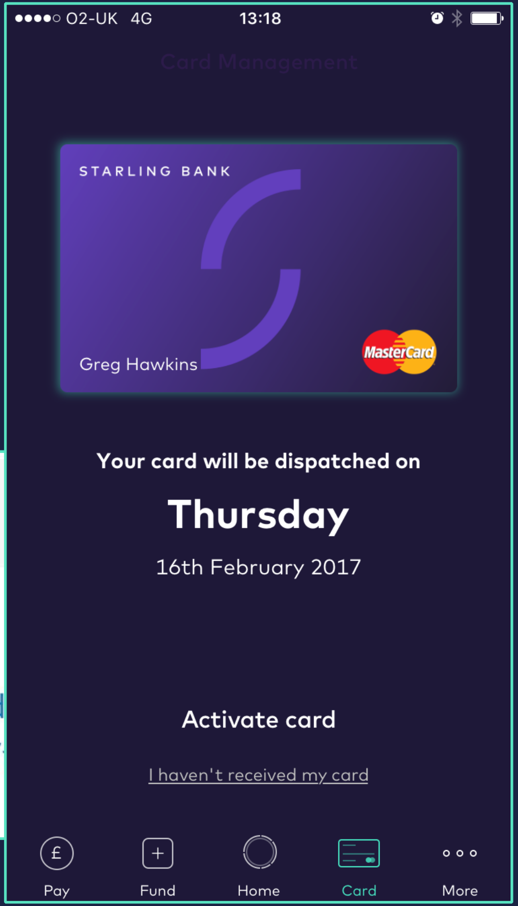
testing service loss

- starbot chat-ops exposes
  - starbot **kill**
  - starbot **kill all**
- available to all developers



greghawkins 8:27 PM
starbot kill all calendar in demo

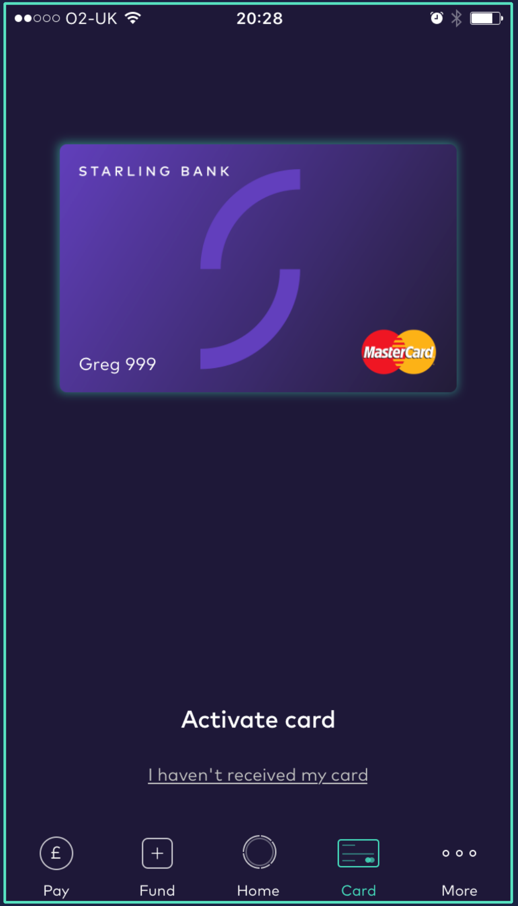
starbot APP 8:27 PM
Forcefully terminating every single **calendar** instance in **demo**.
Underway... keep an eye on https://[redacted]/dashboard status-dashboard.json and https://[redacted]/dashboard/ versions-dashboard.json



●●●●○ O2-UK 4G 13:18

Your card will be dispatched on
Thursday
16th February 2017

Activate card
I haven't received my card



●●○○○ O2-UK 20:28

Activate card
I haven't received my card

L.O.A.S.C.T.T.D.I.T.T.E.O.

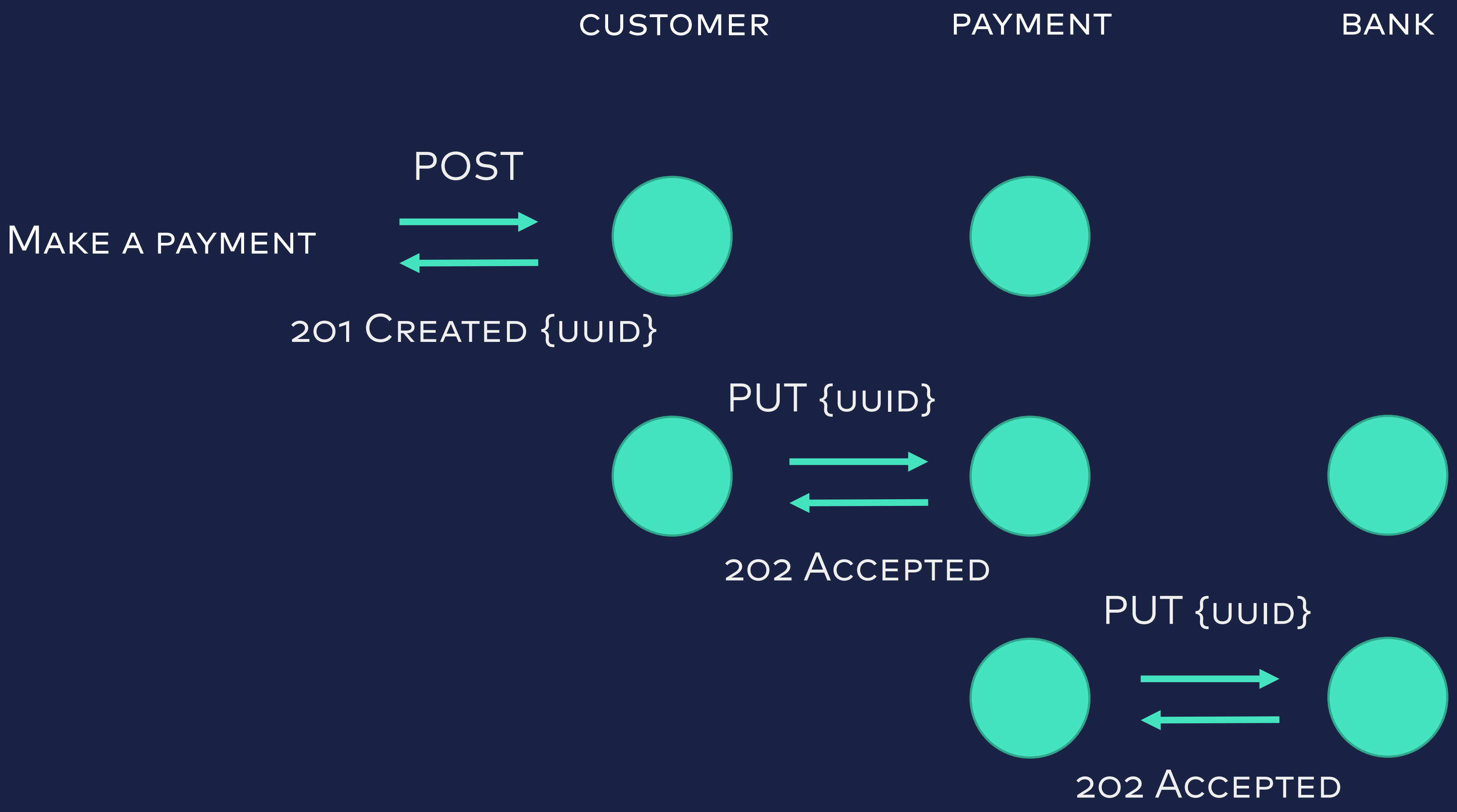
(LOTS OF AUTONOMOUS SERVICES CONTINUALLY TRYING TO DO IDEMPOTENT THINGS TO EACH OTHER)

DITTO architecture

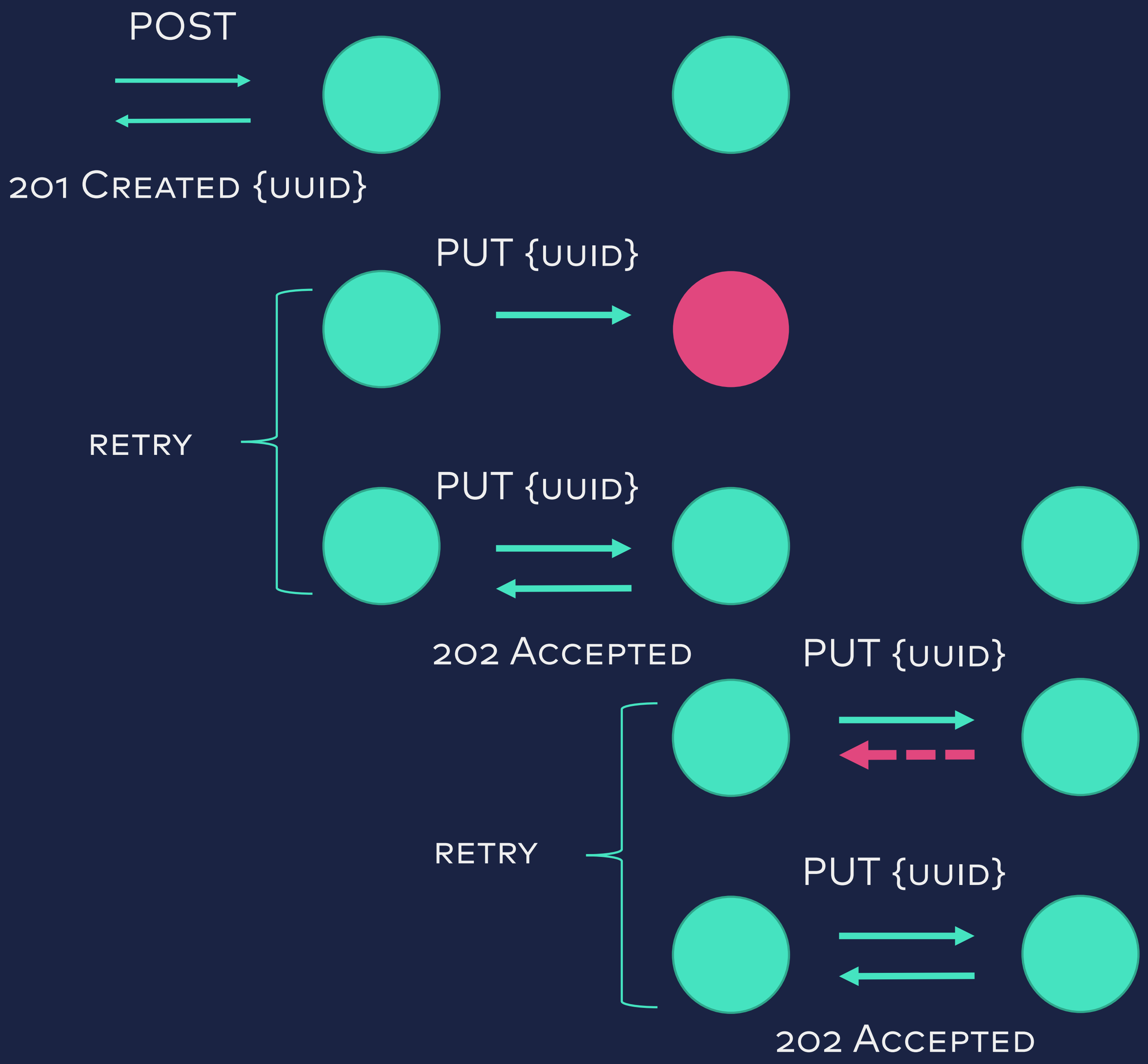
(DO IDEMPOTENT THINGS TO OTHERS)

DITTO architecture

- async + idempotence + retry
  - async: 202 Accepted (once written to store)
  - idempotence: create with PUT
  - retry: accept and store (or 400) then work from database
- each service constantly working towards correctness
- often achieve idempotence by immutability
  - subsequent requests match previous or fail
  - reflects append-only approach to data
- no distributed transactions


CUSTOMER
PAYMENT
BANK
Make a payment
POST
201 Created {uuid}
PUT {uuid}
202 Accepted
PUT {uuid}
202 Accepted

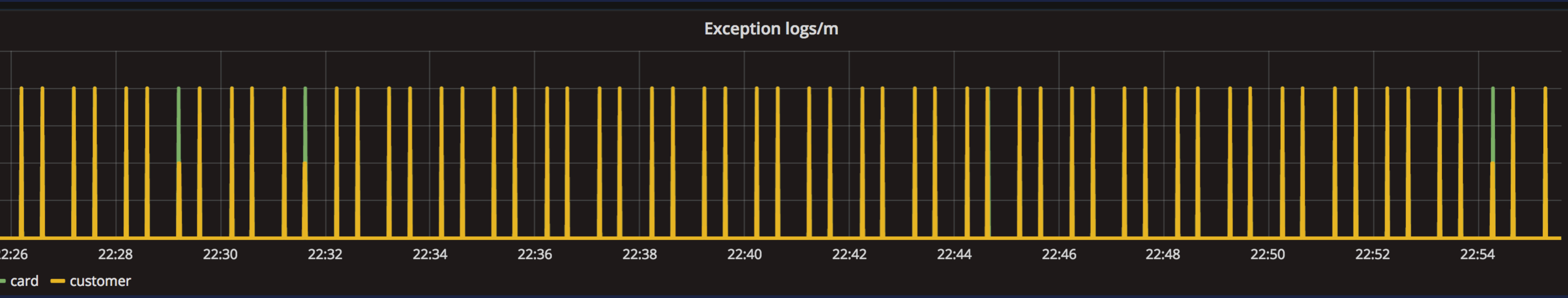
Retry provides "at least once"



Idempotence = "at most once"

234,000

CATCH-UP PROCESSORS WITHOUT BACK-OFF

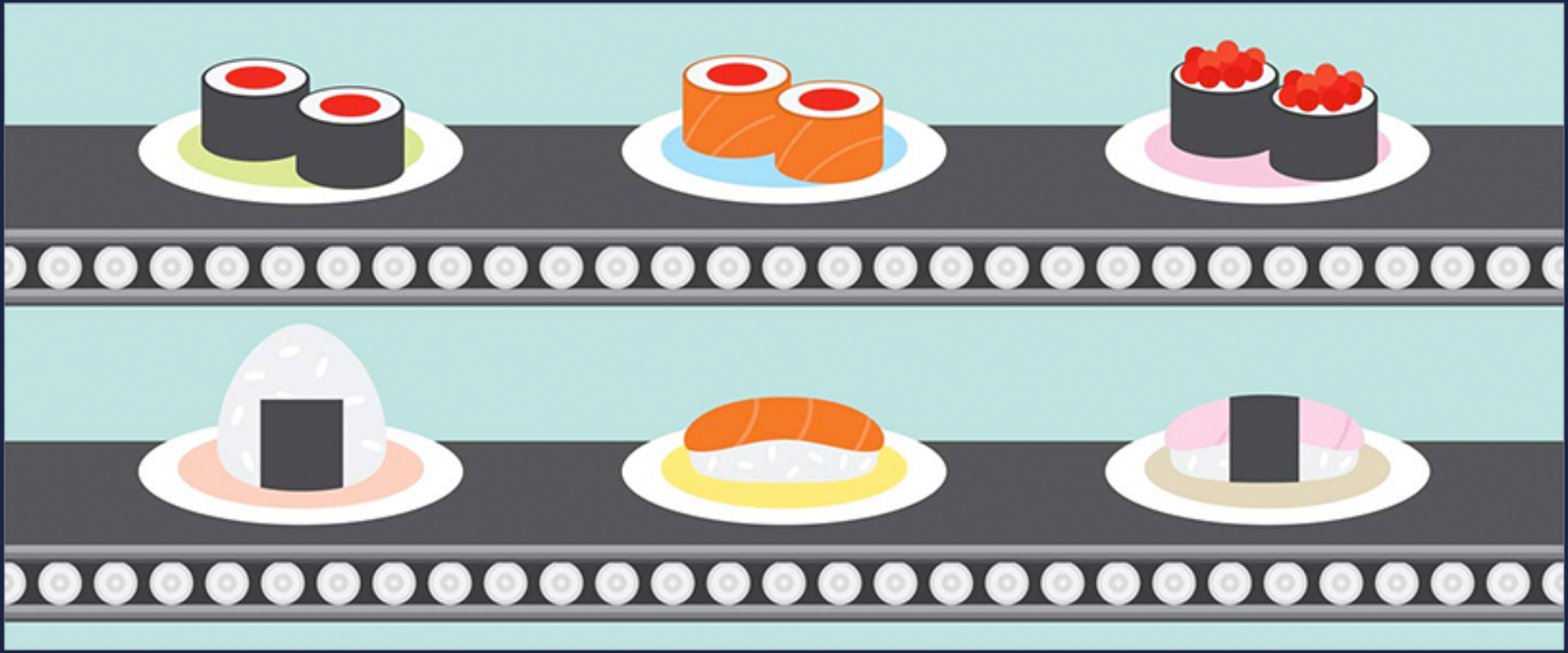


cherish your bad data

impact of service outage

- UI degrades gracefully
- back-end work delayed
  - payments
  - card creation
  - ledger postings
  - interest accrual
- but real disruption: card auth & ATM usage

continuous delivery



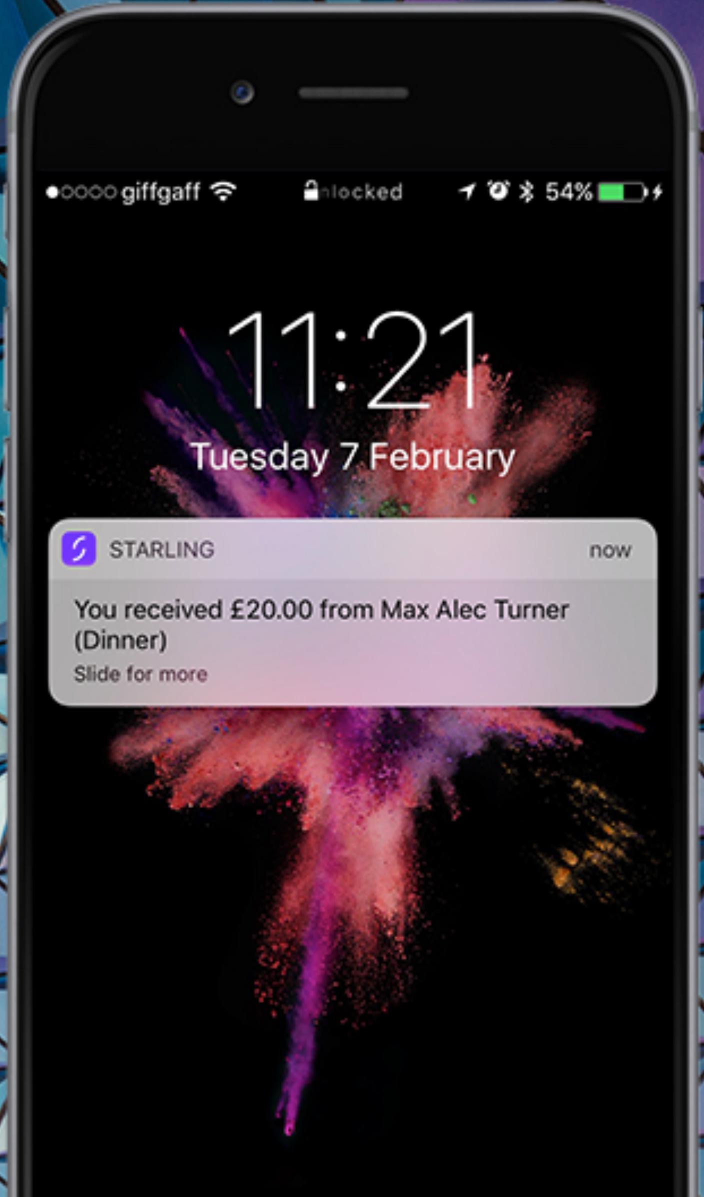
you can do anything you can undo

continuous delivery of back-end

- continual deployment to non-prod, sign-off into prod
- auto build, dockerise, test, scan, deploy < 30m
- in first **221** days of production environment
  - **134** releases of software (~ **1 per weekday**)
  - **70** releases of infrastructure (~**1 per 2 weekdays**)

summary

- SCS + immutable infra + CD
- infrastructure failure absorbed
- failure of function isolated and tolerated
  - UI degrades gracefully
  - items “buffered” and retried
  - fixed safely and swiftly
- this year
  - ++services, scale!, k8s, ML/data


●○○○○ giffgaff
Unlocked
54%
11:21
Tuesday 7 February
STARLING
now
You received £20.00 from Max Alec Turner (Dinner)
Slide for more

starlingbank.com/signup

Careers: starlingbank.com/careers
Hackathon: starlingbank.com/hackathon

Sushi image credit: www.vecteezy.com/vector-art/92795-sushi-platter-vectors