Building a Reliable Cloud Bank in Java

March 2018

@jasonmaude

STARLING BANK

18th June 2012

19th June 2012

20th June 2012

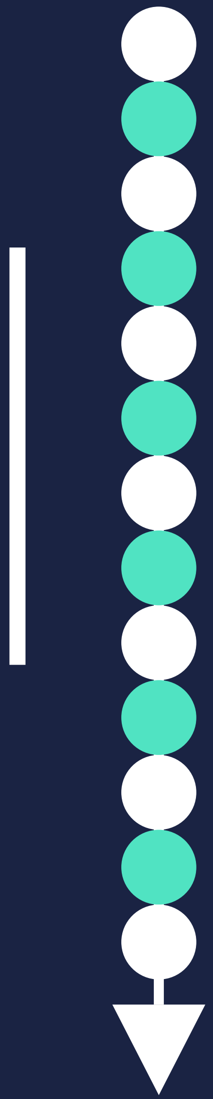
10th July 2012

How did this happen?

The people accepted the possibility of failure

The software didn’t

We built a bank in a year



- 2014 Founded by Anne Boden
- June 2014 Kick-off with Regulators
- Sept 2015 Technical prototypes
- **Jan 2016** Raise $70m – start build
- **July 2016** Banking licence & first account in production
- **October 2016** Mastercard debit cards
- **November 2016** Alpha testing mobile app
- **December 2016** Direct debits live
- **January 2017** Faster payments live
- February 2017 Launched beta testing program
- May 2017 Public App Store Launch
- July 2017 ApplePay
- September 2017 AndroidPay

Starling Bank today

Tech start-up with a banking licence

100% cloud-based, mobile-only

Mastercard debit card

DDs and faster payments

Location-enriched transaction feed

ApplePay, GooglePay, FitBitPay...

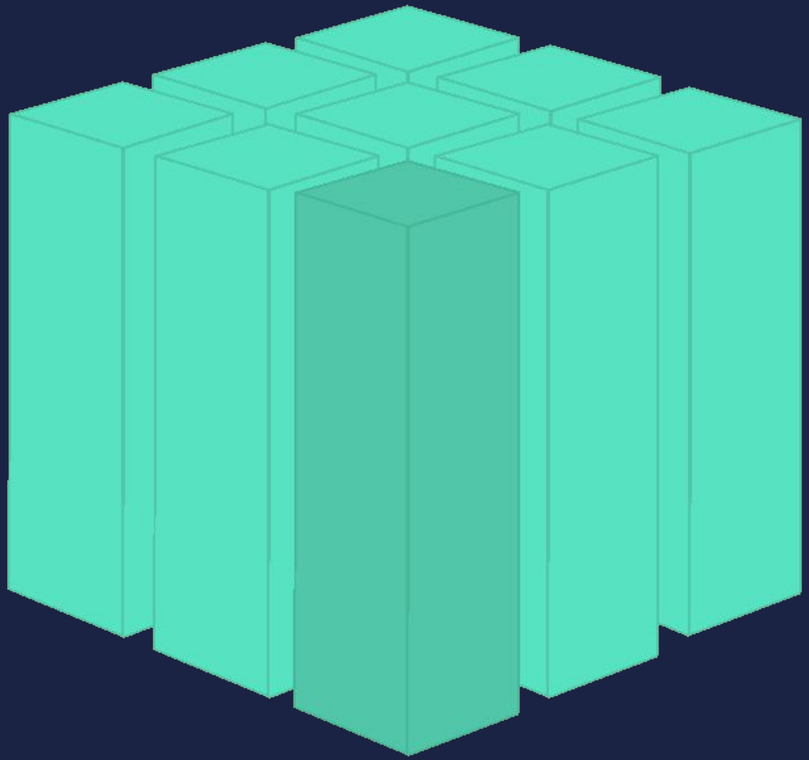
Spending insights

Granular card control

Open APIs & developer platform

Is Java cutting edge?

Self-contained systems



http://scs-architecture.org

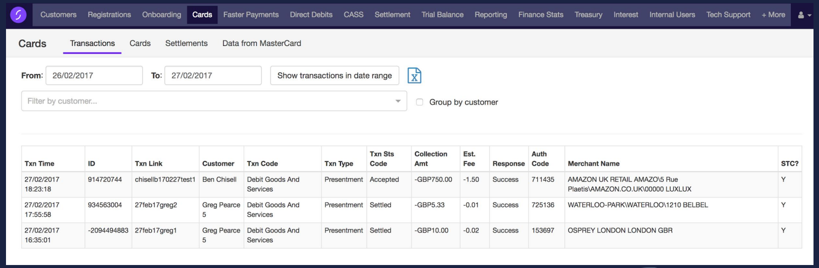
Starling as self-contained systems

- 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
- one key area with sync interaction (balance allocation)

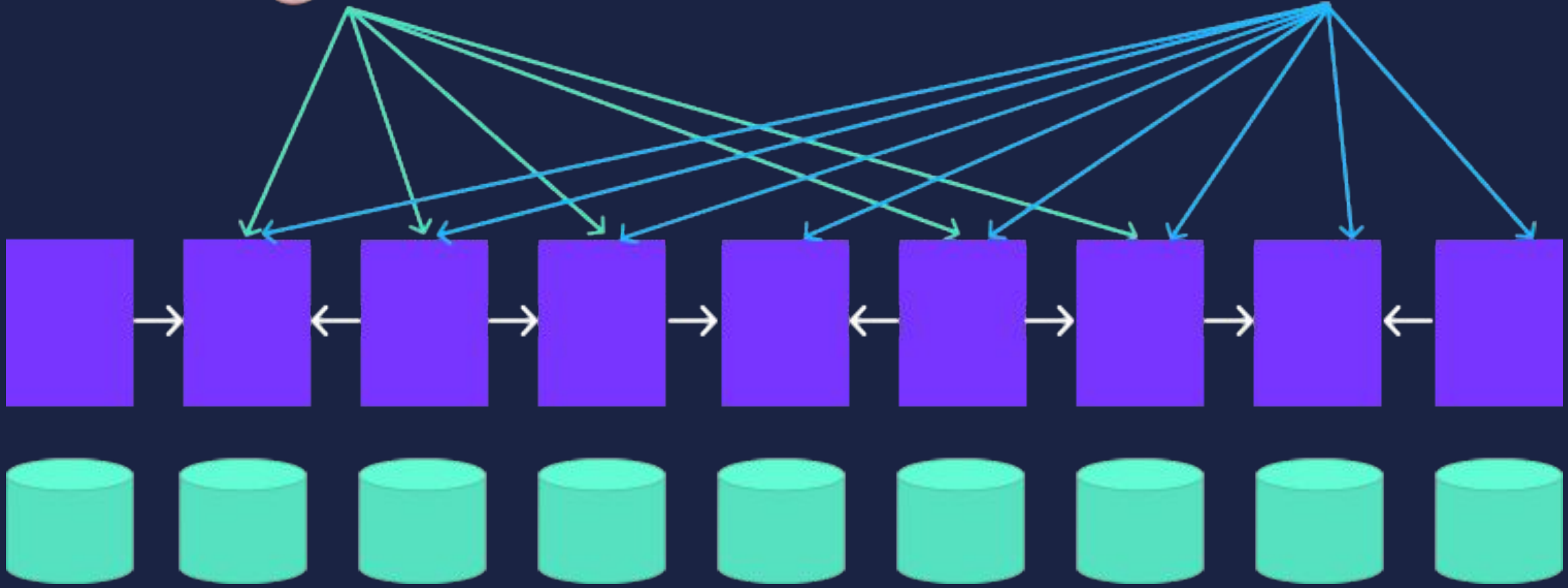
Self-Contained Systems



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



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

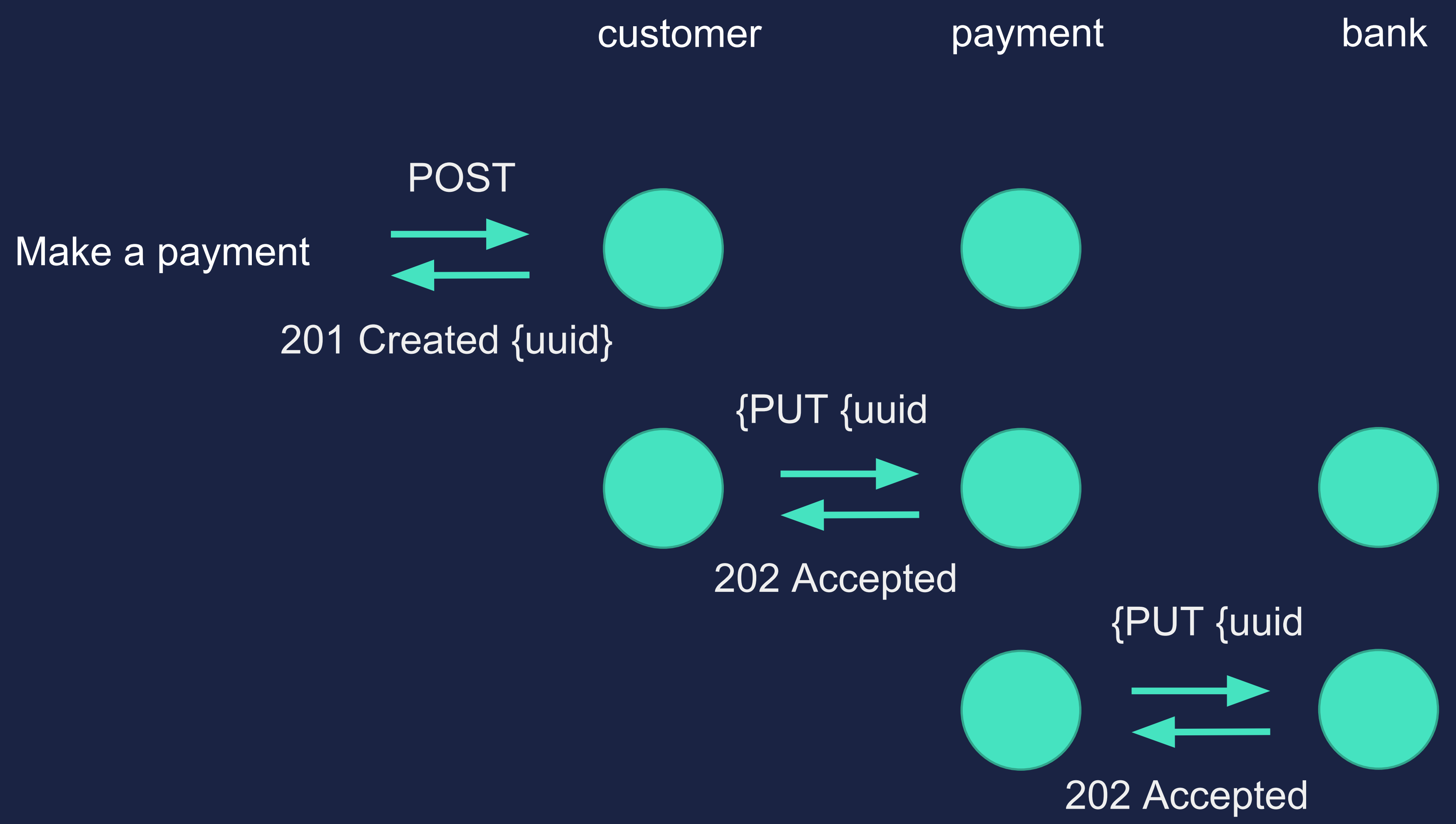
(lots of autonomous services continually trying to do idempotent things to each other)

DITTO architecture

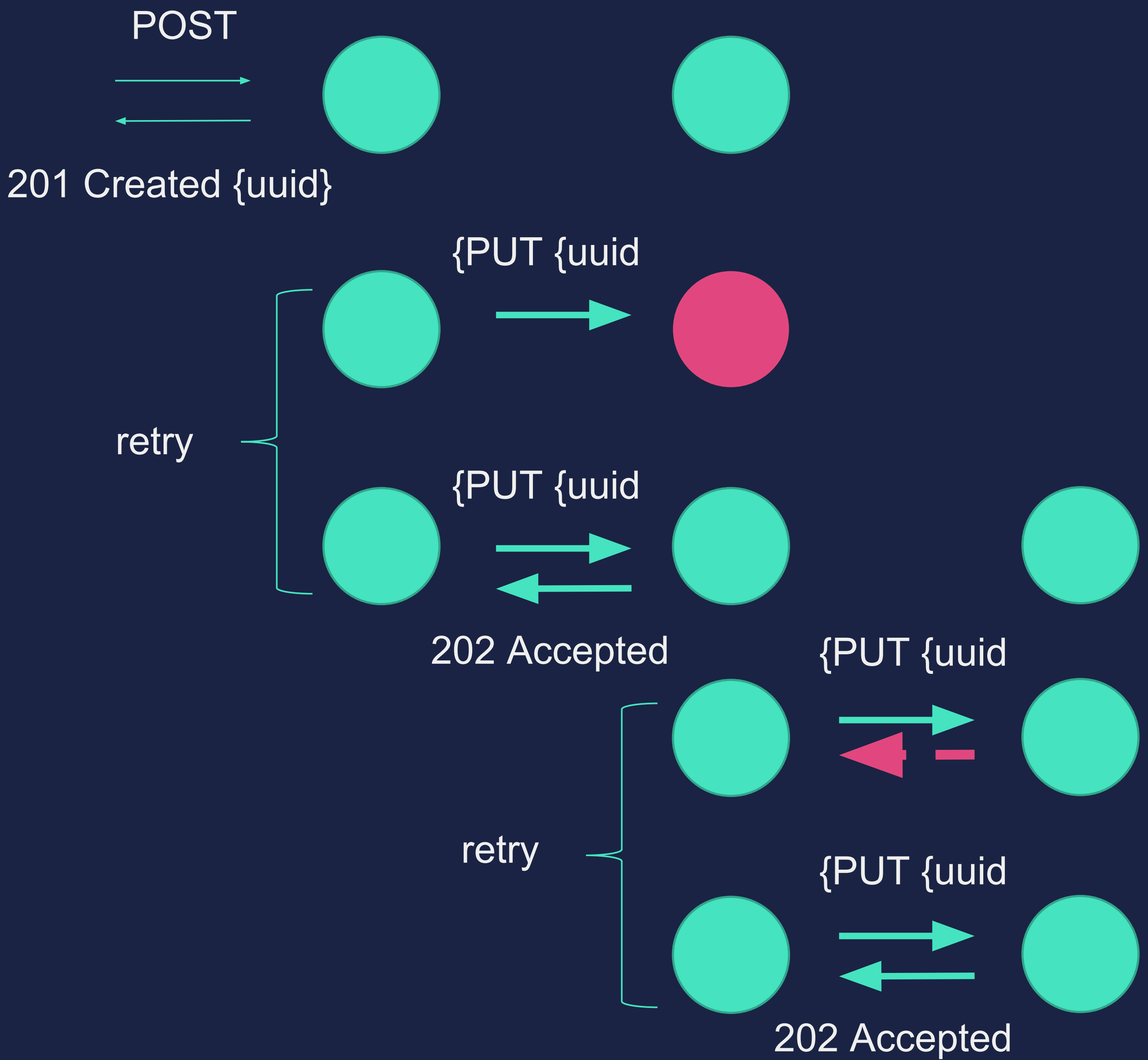
(do idempotent things to others)

DITTO architecture

- do everything at least once and at most once
- async + idempotence + retry
- each service constantly working towards correctness
- often achieve idempotence by immutability
- no distributed transactions
- don’t trust other services


customer
payment
bank
POST
Make a payment
201 Created {uuid}
{PUT {uuid
202 Accepted
{PUT {uuid
202 Accepted

”Retry provides “at least once

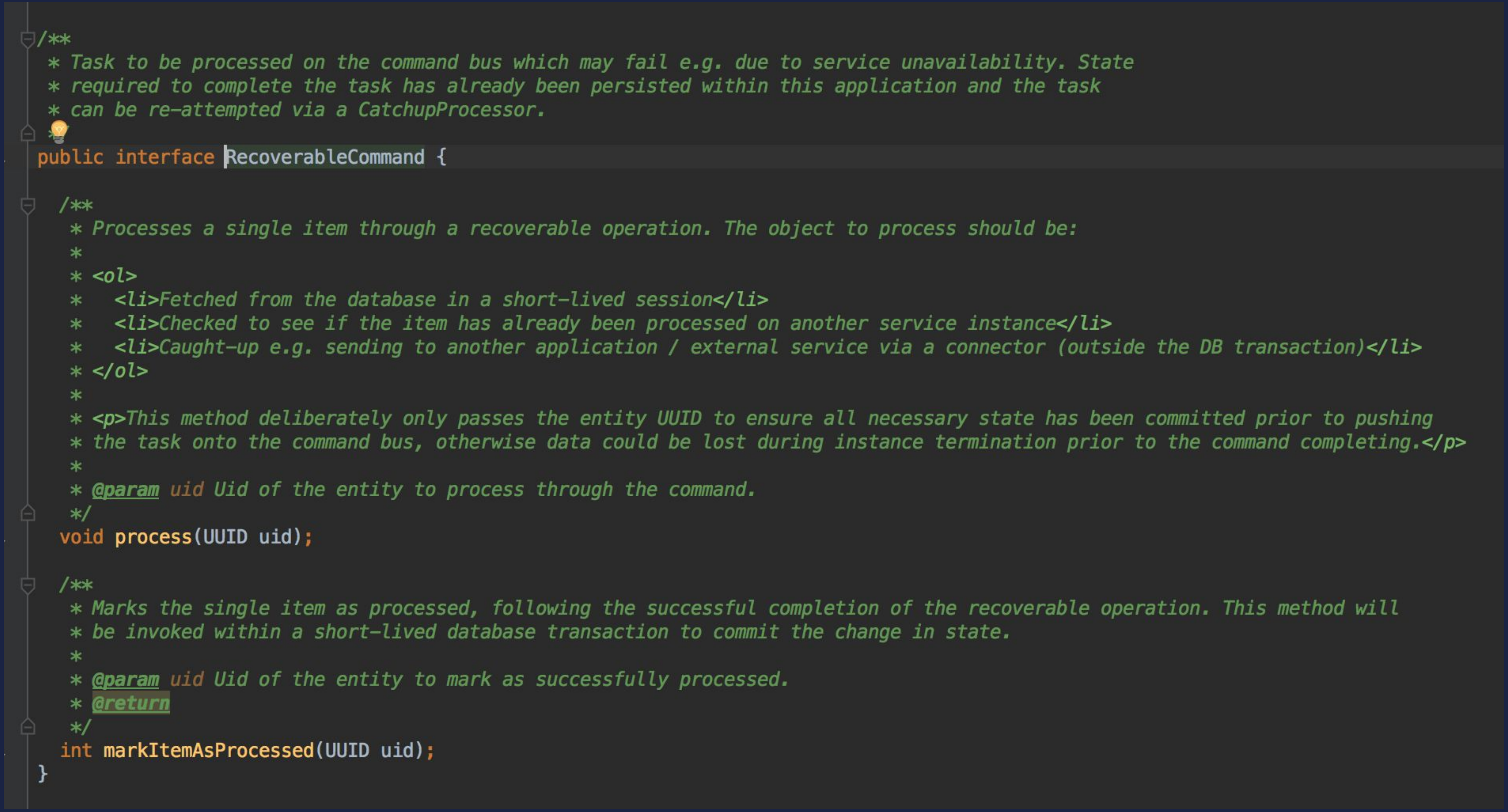


”Idempotence = “at most once

Recoverable Command

- What do I need to do?
- How do I record that I’ve done it?

Recoverable Command

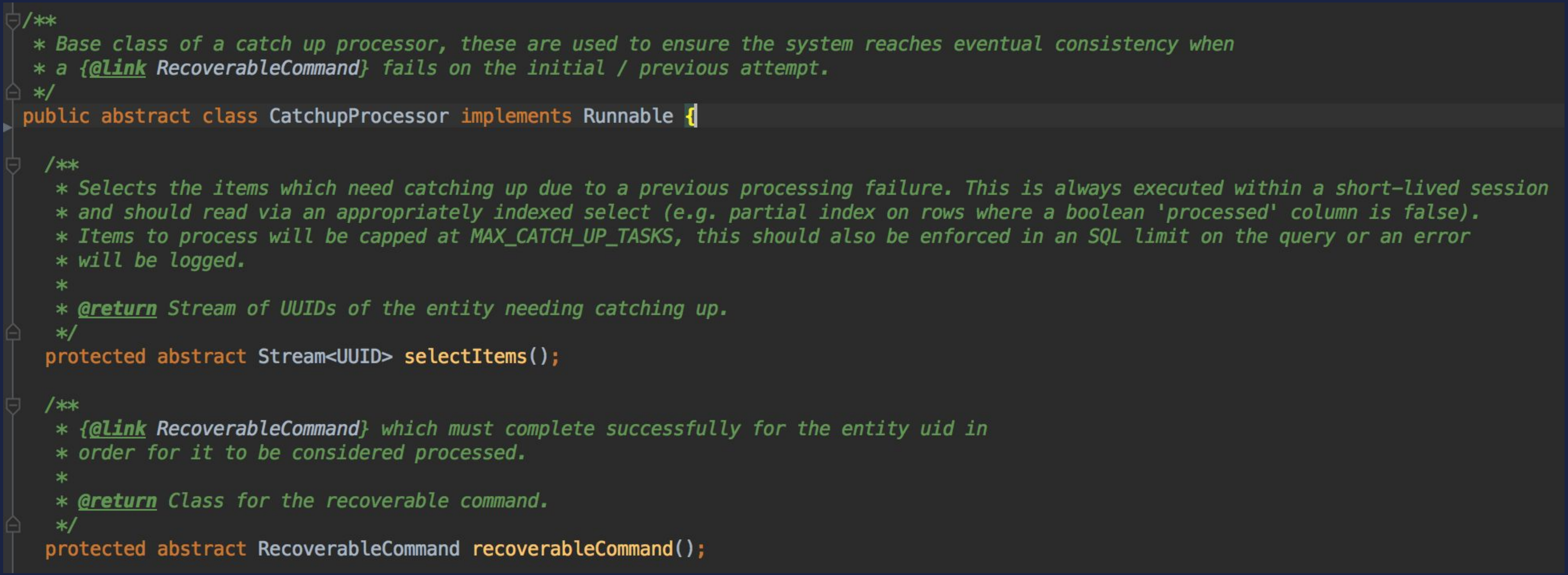


```java
/**
 * Task to be processed on the command bus which may fail e.g. due to service unavailability. State
 * required to complete the task has already been persisted within this application and the task
 * can be re-attempted via a CatchupProcessor.
 */
public interface RecoverableCommand {

  /**
   * Processes a single item through a recoverable operation. The object to process should be:
   *
   * <ol>
   *   <li>Fetched from the database in a short-lived session</li>
   *   <li>Checked to see if the item has already been processed on another service instance</li>
   *   <li>Caught-up e.g. sending to another application / external service via a connector (outside the DB transaction)</li>
   * </ol>
   *
   * <p>This method deliberately only passes the entity UUID to ensure all necessary state has been committed prior to pushing
   * the task onto the command bus, otherwise data could be lost during instance termination prior to the command completing.</p>
   *
   * @param uid Uid of the entity to process through the command.
   */
  void process(UUID uid);

  /**
   * Marks the single item as processed, following the successful completion of the recoverable operation. This method will
   * be invoked within a short-lived database transaction to commit the change in state.
   *
   * @param uid Uid of the entity to mark as successfully processed.
   * @return
   */
  int markItemAsProcessed(UUID uid);
}
```

Catch-up Processor

- Which data items should I attempt to re-process?
- What command should I use to re-process them?

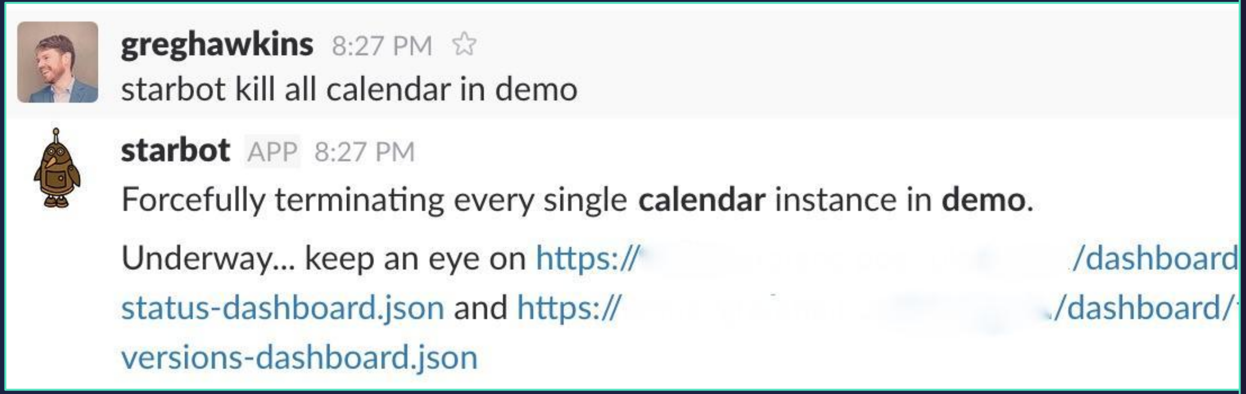
Catch-Up Processor



```java
/**
 * Base class of a catch up processor, these are used to ensure the system reaches eventual consistency when
 * a {@link RecoverableCommand} fails on the initial / previous attempt.
 */
public abstract class CatchupProcessor implements Runnable {

    /**
     * Selects the items which need catching up due to a previous processing failure. This is always executed within a short-lived session
     * and should read via an appropriately indexed select (e.g. partial index on rows where a boolean 'processed' column is false).
     * Items to process will be capped at MAX_CATCH_UP_TASKS, this should also be enforced in an SQL limit on the query or an error
     * will be logged.
     *
     * @return Stream of UUIDs of the entity needing catching up.
     */
    protected abstract Stream<UUID> selectItems();

    /**
     * {@link RecoverableCommand} which must complete successfully for the entity uid in
     * order for it to be considered processed.
     *
     * @return Class for the recoverable command.
     */
    protected abstract RecoverableCommand recoverableCommand();
```

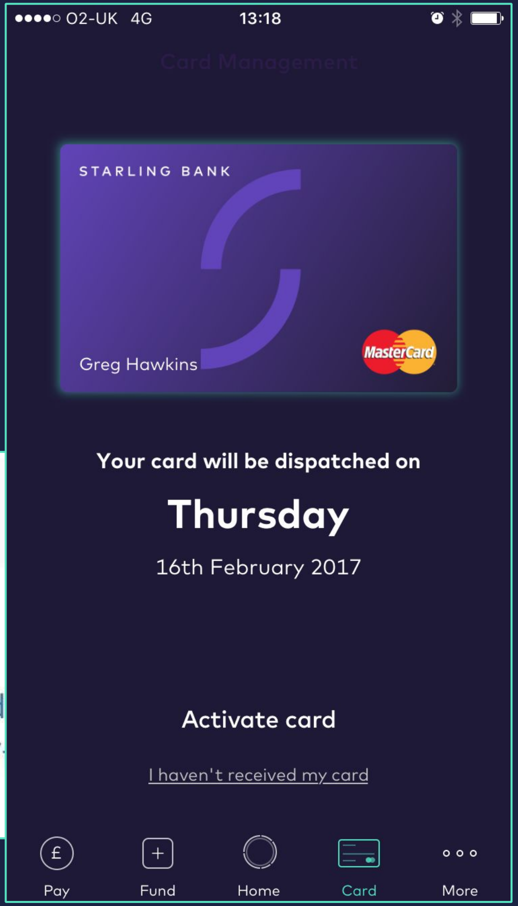
Testing

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



greghawkins 8:27 PM
starbot kill all calendar in demo

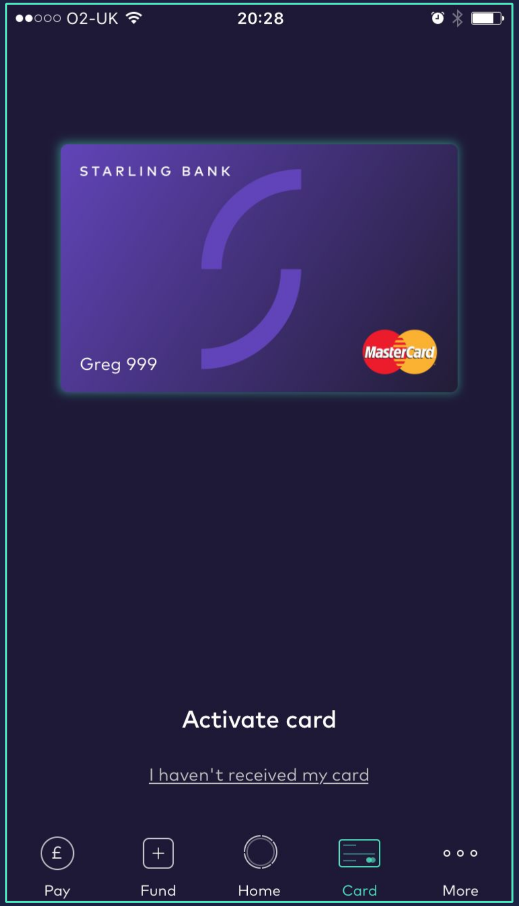
starbot APP 8:27 PM
Forcefully terminating every single **calendar** instance in **demo**.
Underway... keep an eye on https://[redacted]/dashboard[redacted]status-dashboard.json and https://[redacted]/dashboard/[redacted]versions-dashboard.json



Your card will be dispatched on

Thursday

16th February 2017



Instance termination is safe

- single stateless service per instance
- if ever a server is in doubtful state, kill it
- chat-ops slack bot
- rolling deployments by termination (not quick but safe)

Continual delivery of back-end

- continual deployment to non-prod, sign-off into prod
- auto build, dockerise, test, scan, deploy < 1h
- code released to production up to 5 times a day

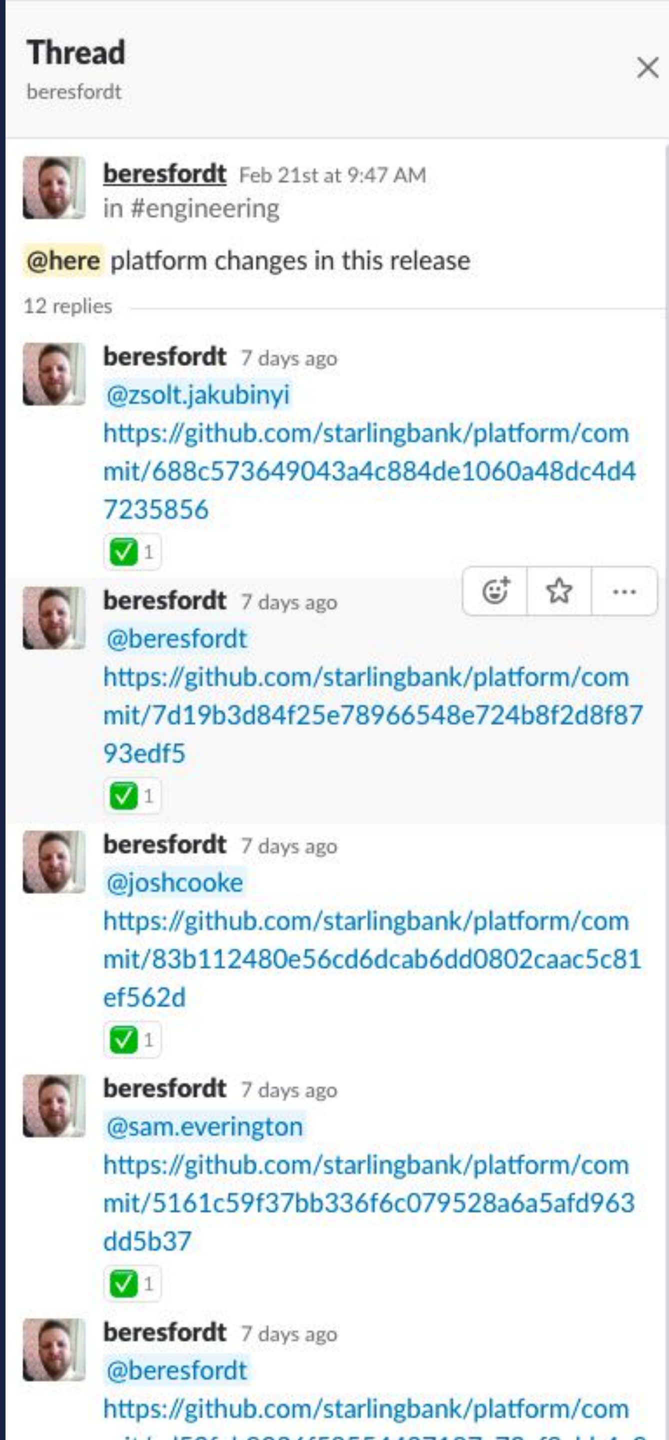
We have turned 2-speed IT on its head

- traditional banks operate:
  - legacy backends that move at glacial pace
  - and try to iterate the customer experience faster

- we release the backend at **10x** the rate of the mobile apps
  - 1-5 backend software releases per day
  - 1-2 infrastructure releases per day
  - mobile apps released weekly or fortnightly

A “take ownership” ceremony

- all engineers explicitly bless their commits in slack
- everyone knows the release is imminent
- everyone knows when their changes go out
- everyone gets a last ditch “OMG” opportunity
- everyone asserts their change is “good for prod”



beresfordt Feb 21st at 9:47 AM
in #engineering

@here platform changes in this release

12 replies

beresfordt 7 days ago
@zsolt.jakubinyi
https://github.com/starlingbank/platform/commit/688c573649043a4c884de1060a48dc4d47235856
✅ 1

beresfordt 7 days ago
@beresfordt
https://github.com/starlingbank/platform/commit/7d19b3d84f25e78966548e724b8f2d8f8793edf5
✅ 1

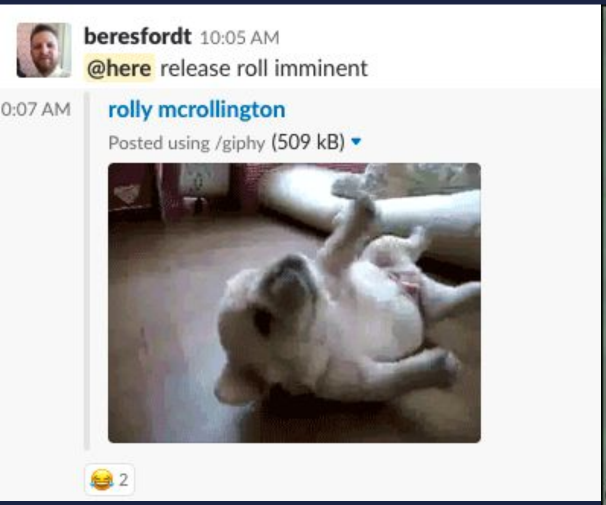
beresfordt 7 days ago
@joshcooke
https://github.com/starlingbank/platform/commit/83b112480e56cd6dcab6dd0802caac5c81ef562d
✅ 1

beresfordt 7 days ago
@sam.everington
https://github.com/starlingbank/platform/commit/5161c59f37bb336f6c079528a6a5afd963dd5b37
✅ 1

beresfordt 7 days ago
@beresfordt
https://github.com/starlingbank/platform/com

The “rolling” giphy

- our auditors *loved* this one
- yes it’s in our release documentation
- clear signal in engineering channel that is release in progress

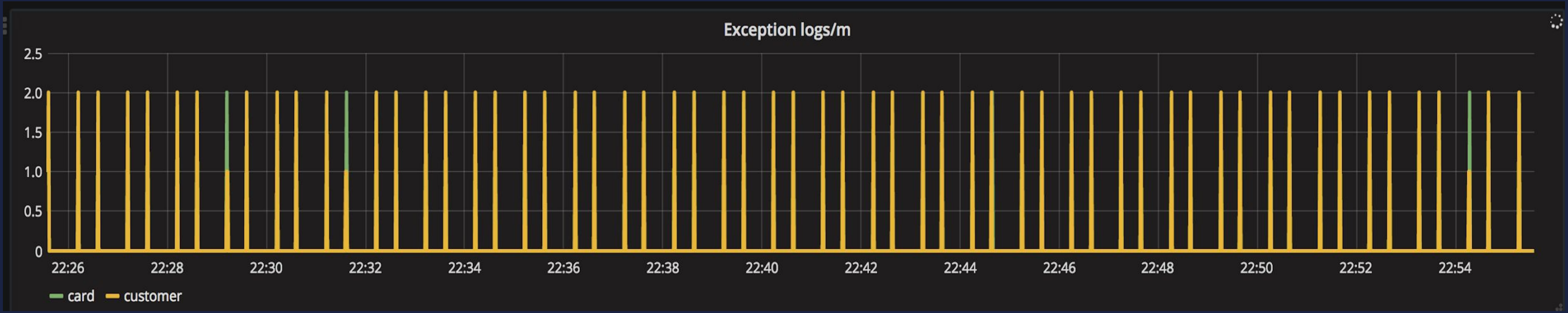


beresfordt 10:05 AM
@here release roll imminent

0:07 AM **rolly mcrollington**
Posted using /giphy (509 kB)

😂 2

… and if something goes wrong...



Case Study

- a failed db upgrade locked the db in notification service
- customer service kept trying to send requests to notification
- the queue in customer filled up, meaning that other requests were denied
- problem was located, instances of customer could be regularly recycled until the problem was fixed
- once the problem was fixed all the work due in notification was performed as required

… but why Java?

- exceptions are noisy and difficult to ignore
- integrations with legacy third parties (SOAP etc)
- lightweight (if you cut down on your dependencies)
- reliable ecosystem (user base, job market, etc)

… and finally: some important takeaways

Give EVERYTHING a UUID

It’s not just the hardware that can fail

Cherish your bad data

You can do anything you can undo

For more of Starling Bank see Yann and Teresa on Tuesday - 17:25 (Next Gen Bank track)

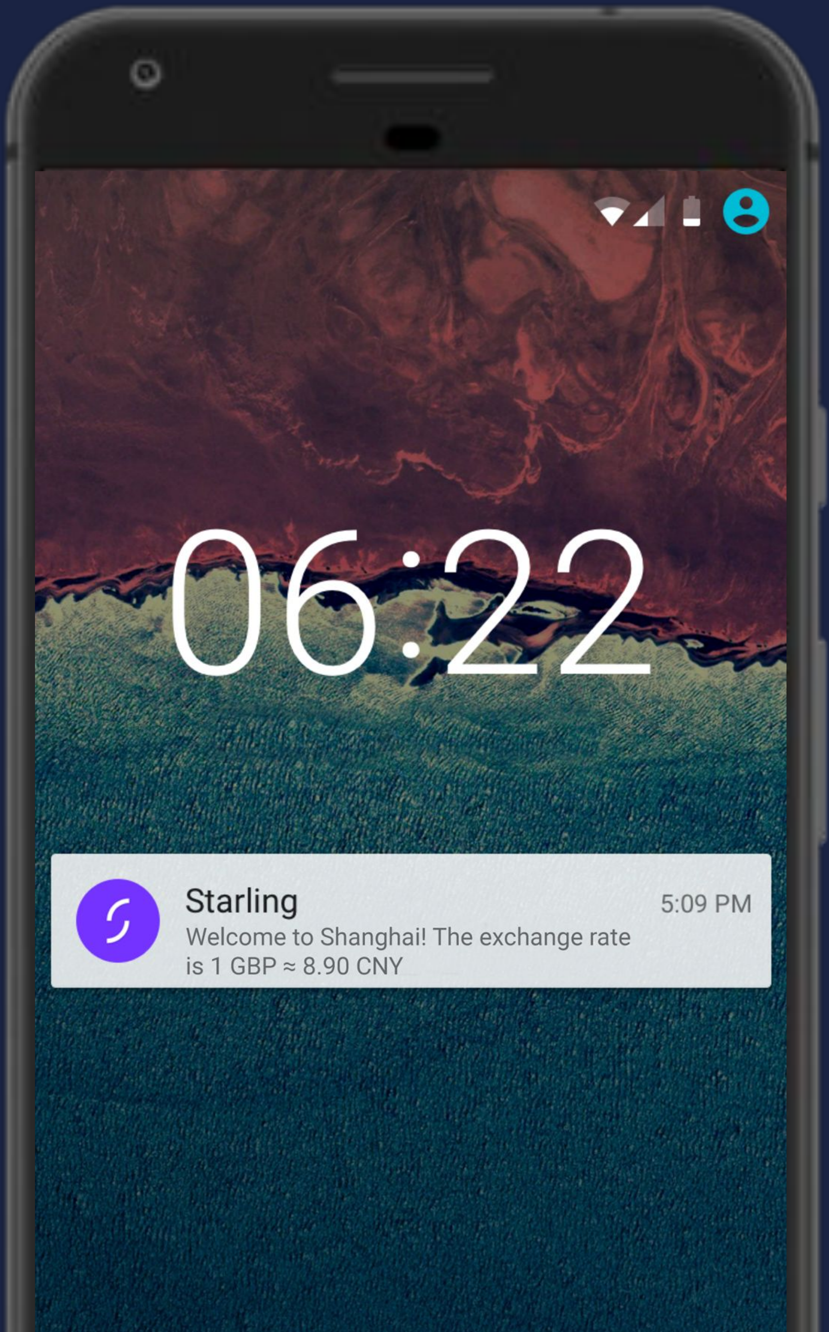
Thank You!

https://developer.starlingbank.com

Check out the Starling Developer Podcast!

Download on the App Store

GET IT ON Google Play



Starling 5:09 PM
Welcome to Shanghai! The exchange rate is 1 GBP ≈ 8.90 CNY