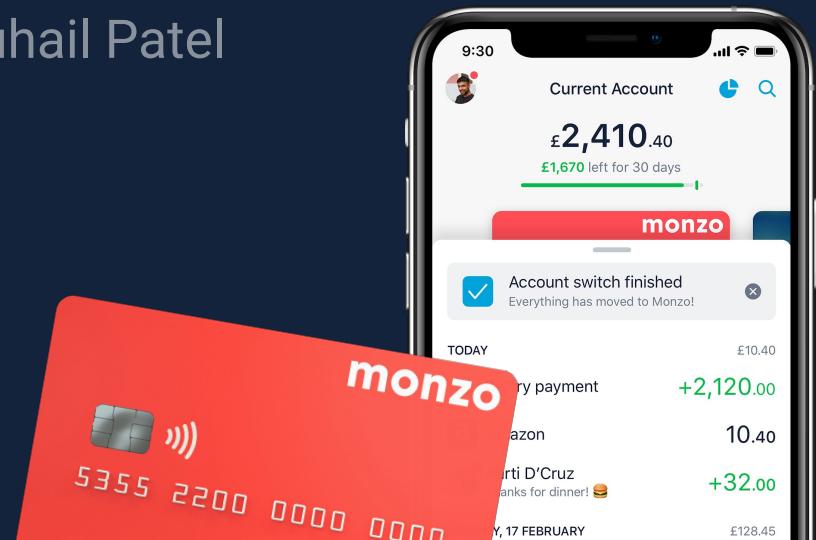
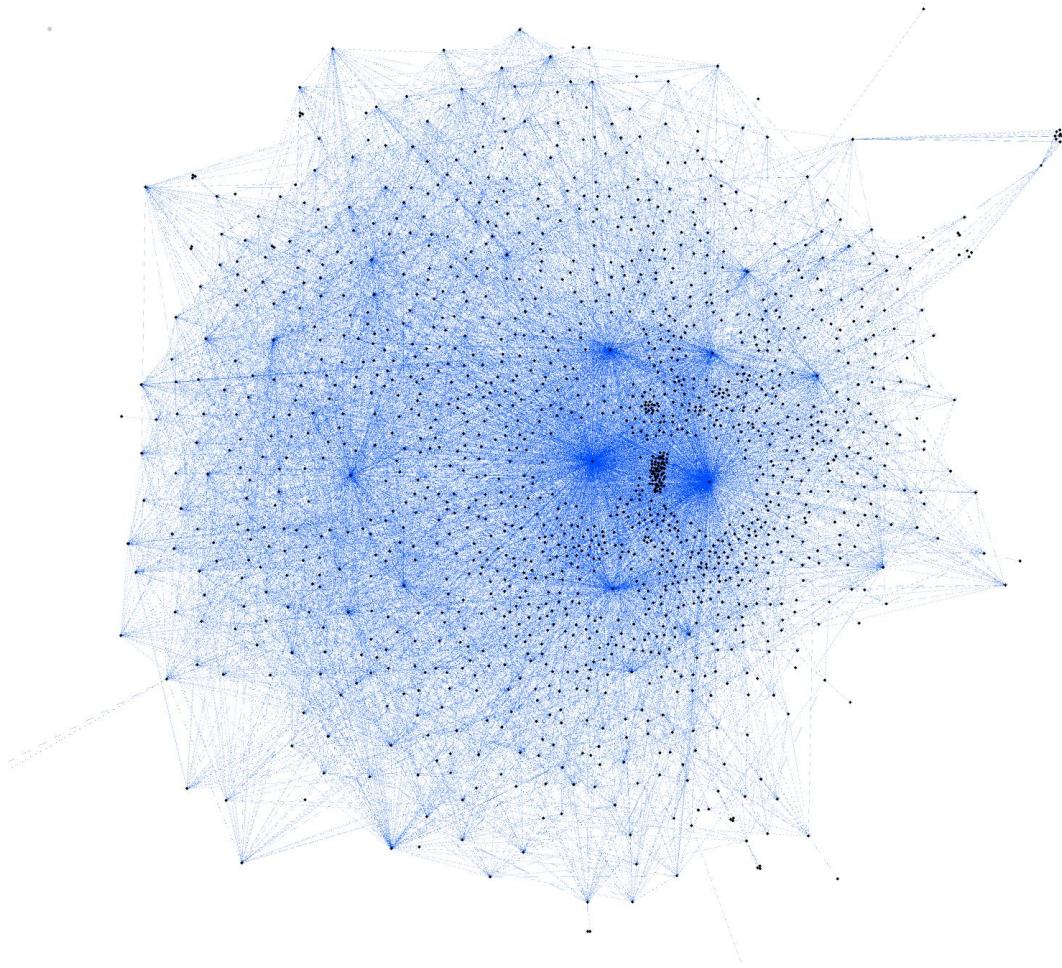


Modern Banking in 1500+ 1600 Microservices

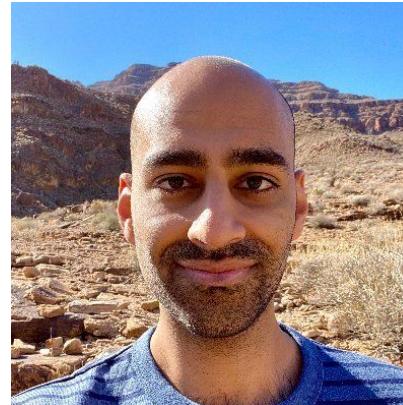
Matt Heath & Suhail Patel



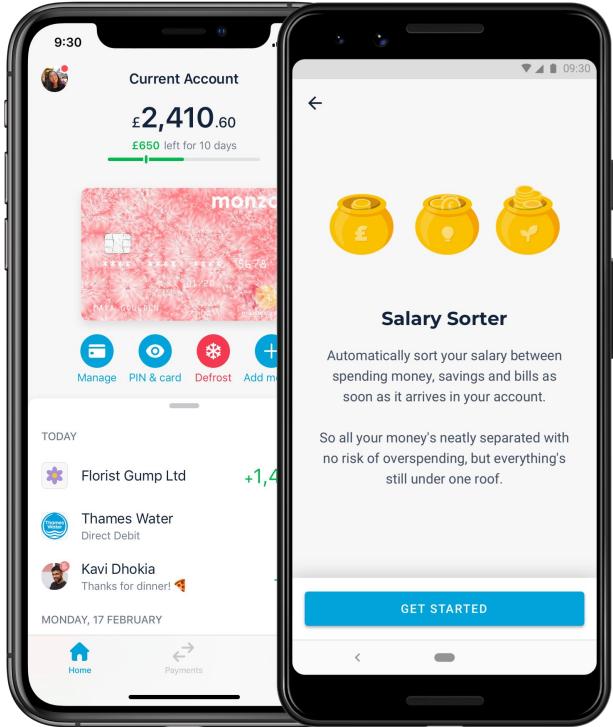
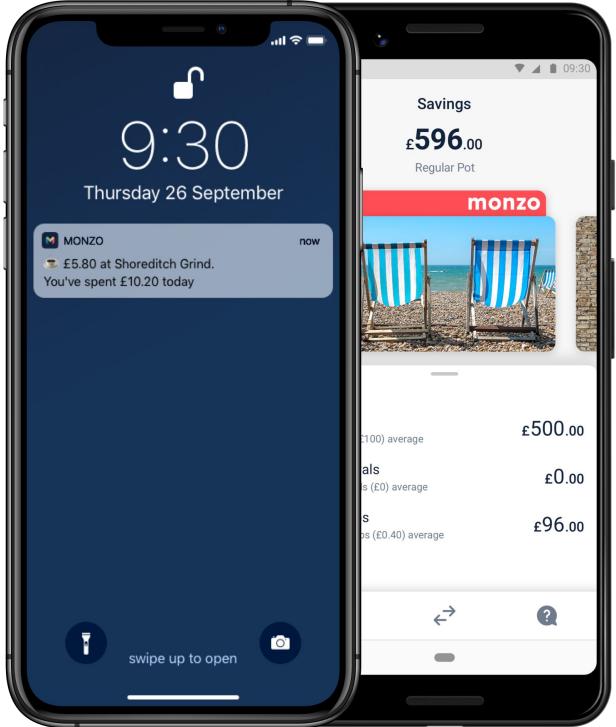




Matt Heath
Senior Staff Engineer at Monzo
[@mattheath](https://twitter.com/mattheath)



Suhail Patel
Senior Engineer at Monzo
[@suhailpatel](https://twitter.com/suhailpatel)





```
$ curl https://api.monzo.com/branches  
{"branches": [], "comment": "All our branches are on GitHub."}
```



**Join the 4,000,000 people with a
Monzo bank account**



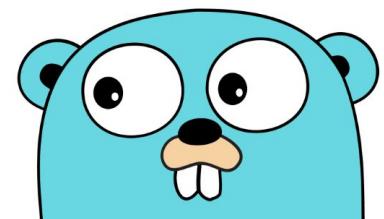




Project

untitled











oliver Oliver Beattie Monzo



oliver

Oct '17

Oct 2017

Hi everyone I'm Monzo's Head of Engineering, and as I [promised](#) on Friday I'd like to share some more information about what happened during this outage. Because the nature of the issue was technical, this post is also quite technical.

It's important to note that we had two major incidents last week that many of you will have experienced (sorry again.) The first incident lasted most of the week and affected only our prepaid product – ie. Monzo Alpha and Beta cards. The second outage affected both the prepaid product and our new current account for a period of around 1½ hours on Friday afternoon. This post is about the latter.

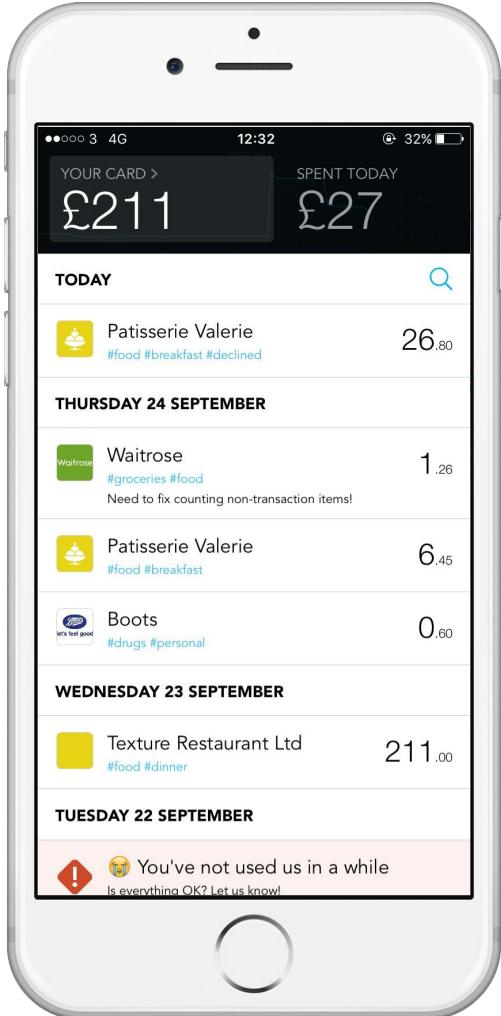
You can learn more about our overall backend architecture in [this blog post](#) 985 I published last year, but it's important to understand the role of a few components in our stack at a high level to understand this issue:

- [Kubernetes](#) 124 is a system which deploys and manages all of our infrastructure. Monzo's backend is written as several hundred microservices, packaged into Docker containers. Kubernetes manages these Docker containers and ensures they are running properly across our fleet of AWS nodes.
- [etcd](#) 152 is a distributed database used by Kubernetes to store information about which services are deployed, where they are running, and what state they're in. Kubernetes requires a stable connection to etcd in order to work properly, although if etcd does go down all of our services do continue running – they just can't be upgraded, or scaled up or down.
- [linkerd](#) 556 is a piece of software that we use to manage the communication between all of the services in our backend. In a system like ours, thousands of network calls are happening every second, and linkerd does the job of routing and load balancing all of these calls. In order to know where to route these calls, it relies on being able to receive updates about where services are located from Kubernetes.

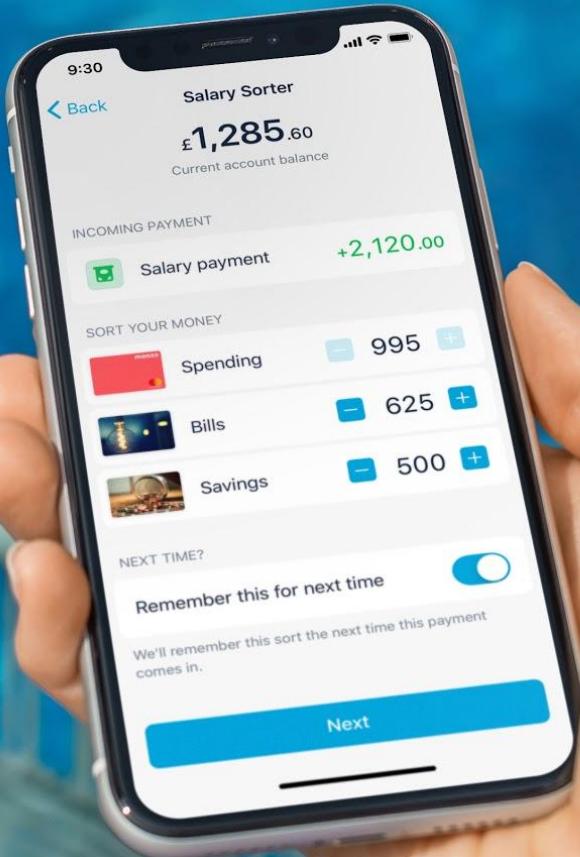
94 / 185

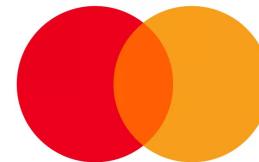
Oct 2017

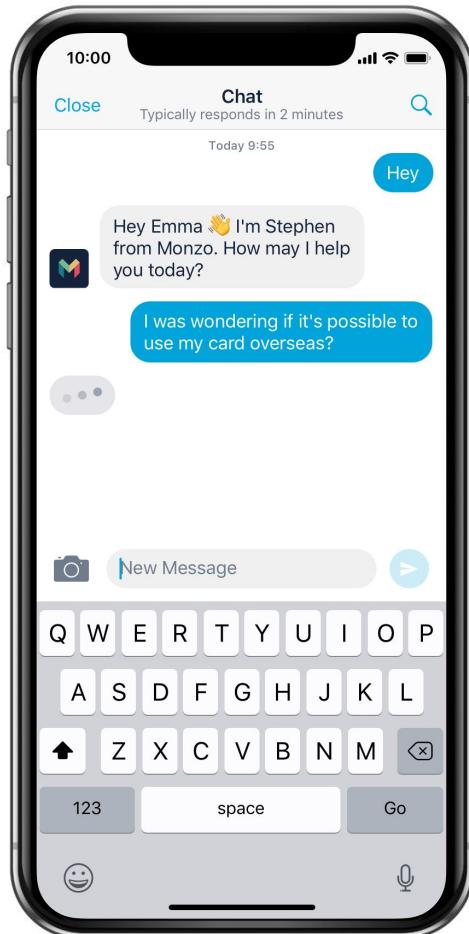
Dec 2017













My Hub

3/3



Chat with Groundskeeper Willie 9 minutes ago

Groundskeeper Willie: Doo do do do do dooooo 🌟

Chat with Randall de Smithburg a few seconds ago

Hank Scorpio: Is there anything else I can help you with?

Support Call (Incoming) 5 months ago

+44700000000

Tag t Escalate e Pause p Assigned a ...

Conversation auto-closed a day ago

It's gonna take a lot to drag me away from you
a day ago

Auto-assigned by Hank Scorpio a day ago

There's nothing that 100 men or more could ever do
a day ago • Read

I bless the clouds down in Africa
a day ago

Gonna take some time to do the things we never had
a day ago • Read

Doo do do do do dooooo 🌟 🌟

a day ago

Paused by Hank Scorpio a day ago

Reply Note

Type a message...

Send

Quick Actions



Current Account

Groundskeeper Willie

Date of birth	11 September 1994 (24 years old)
Account details	98793708 12-34-56
Current location	Location unknown
Device & app	ONEPLUS A5010 — Android 9.0.0, Monzo 2.54.0-master-65b8183ef
Last updated	Jun 21, 2019 4:12 PM
Account opened	Jun 17, 2019 10:53 AM
Account	£0.00
Overdraft limit	£0.00
Pots (0)	-
Last 4 card digits	7012

Customer

Date of birth	11 September 1994 (24 years old)
Legal name	Groundskeeper Willie
Preferred name	Groundskeeper Willie
Username	
User ID	user_00009jw3puspjwJL5hdrh
Account ID	acc_00009jw3qMy7Jtdv9U2nuz
Email	groundskeeperwillie@example.com
Phone	+44 000000000
Address	Flat 1, 230 City Road

<https://www.youtube.com/watch?v=HHsIPa1AVX8>

IFTTT

TransUnion®

flux

:experian™

pvc energy

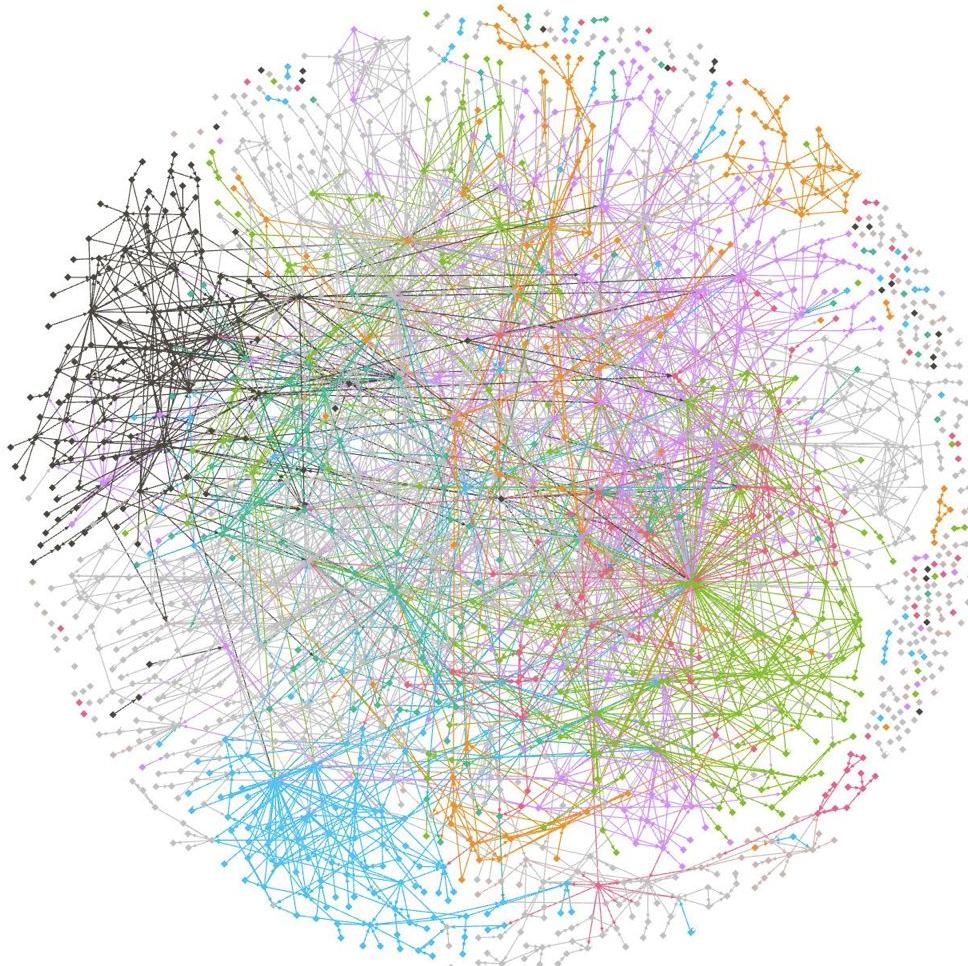


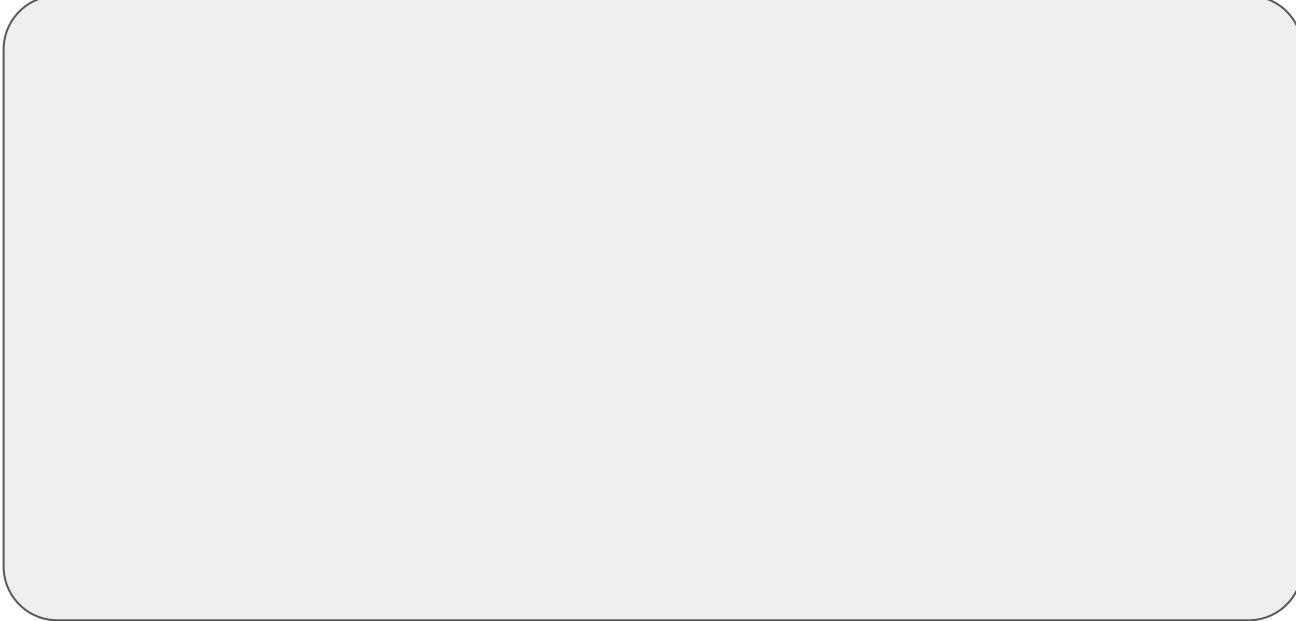


Feb
2015

monzo services

Mar
2020



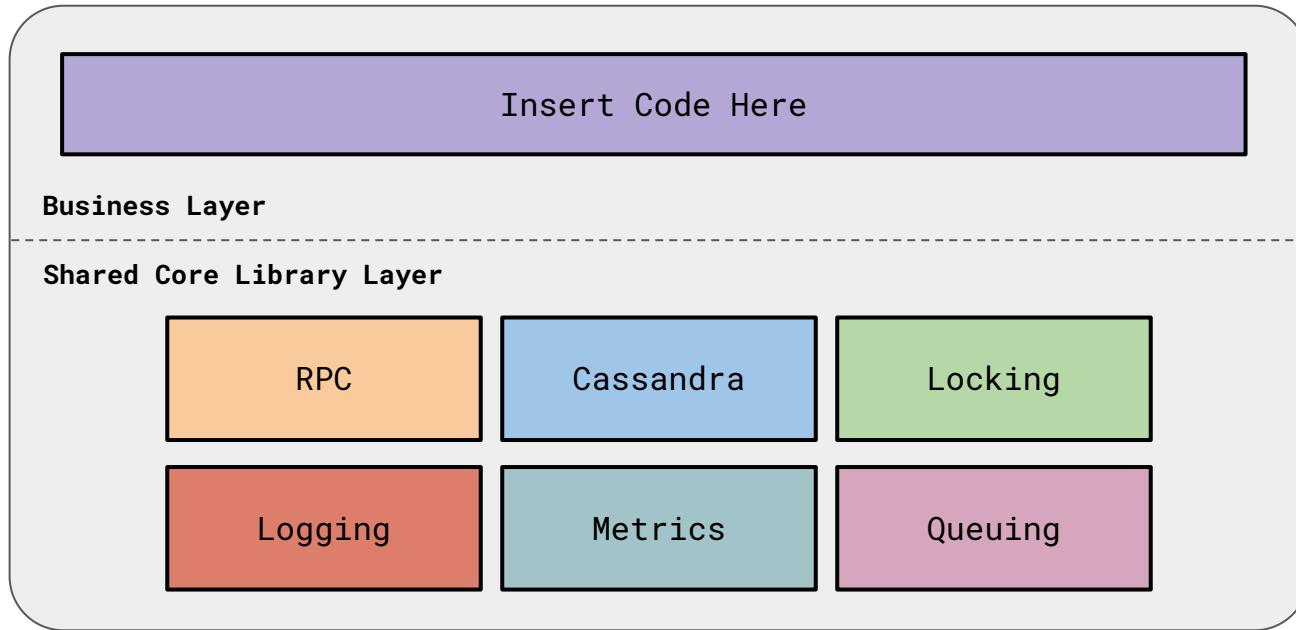


A microservice

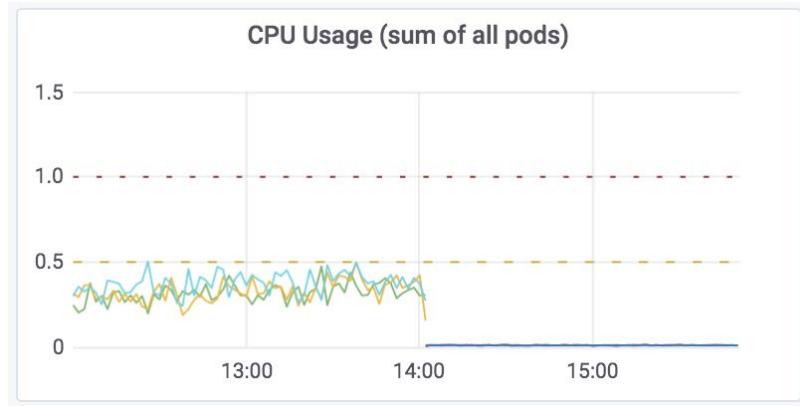
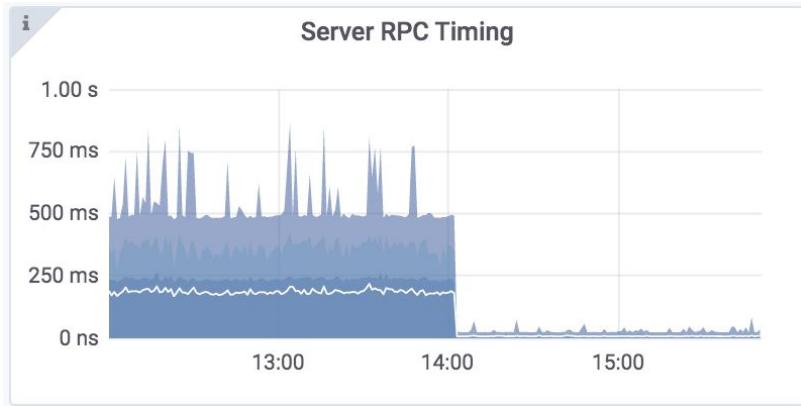
Insert Code Here

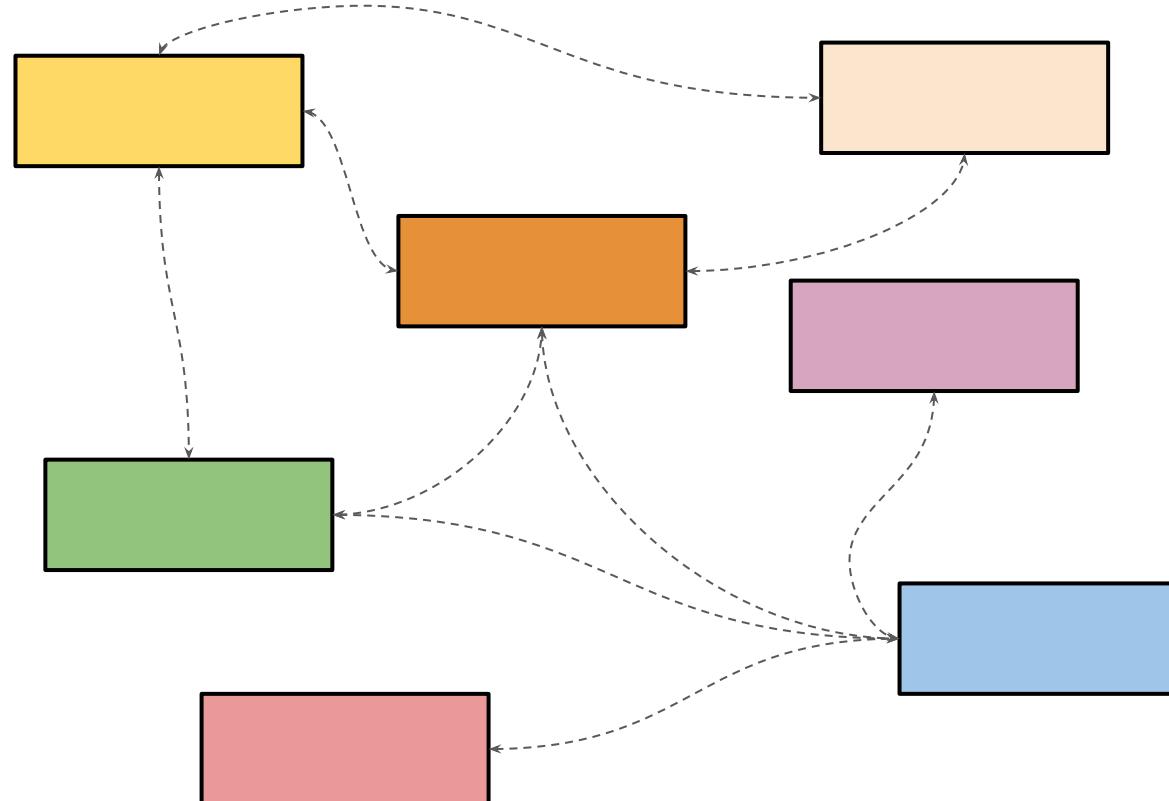
Business Layer

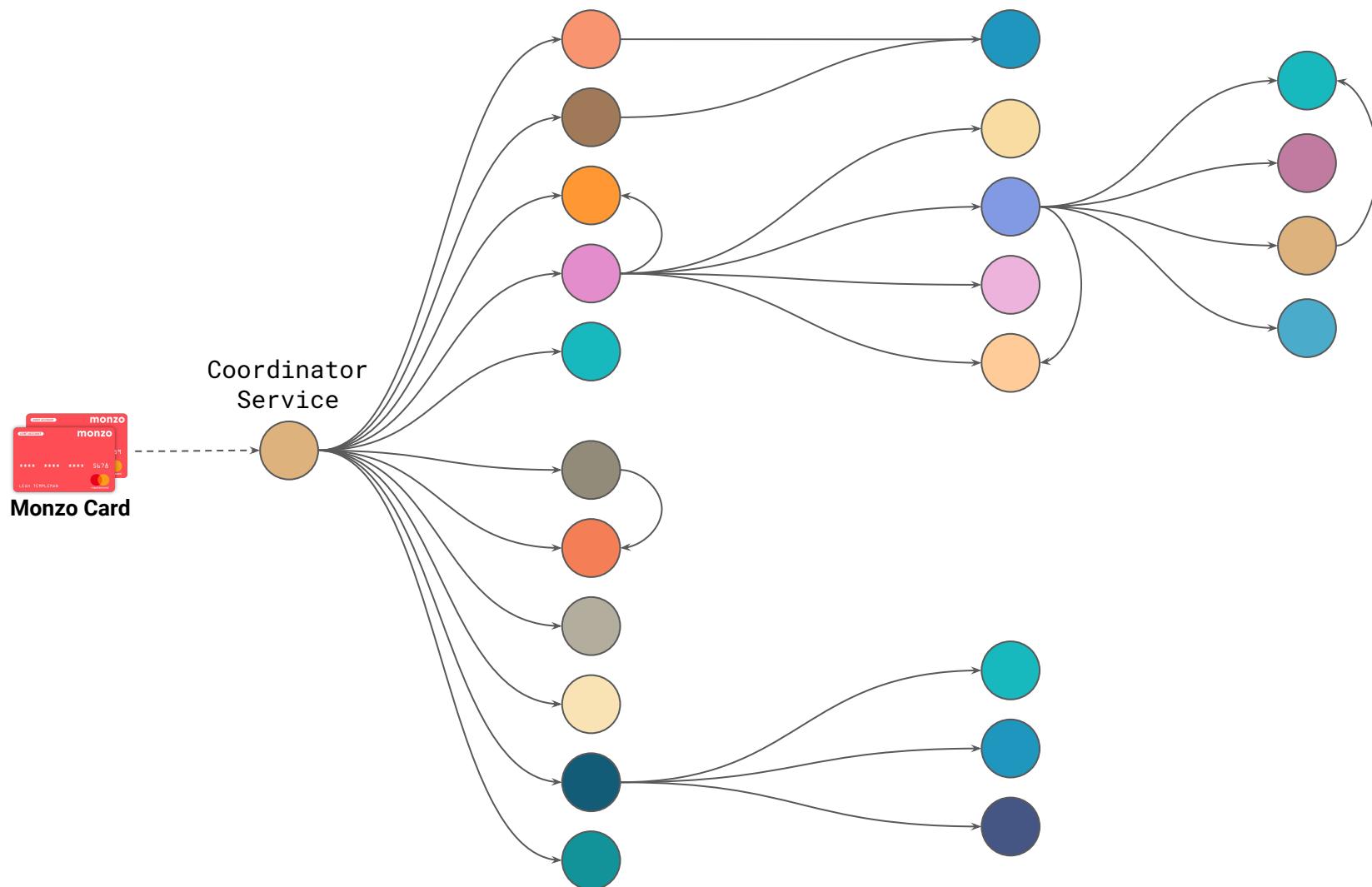
A **microservice**



A microservice









11

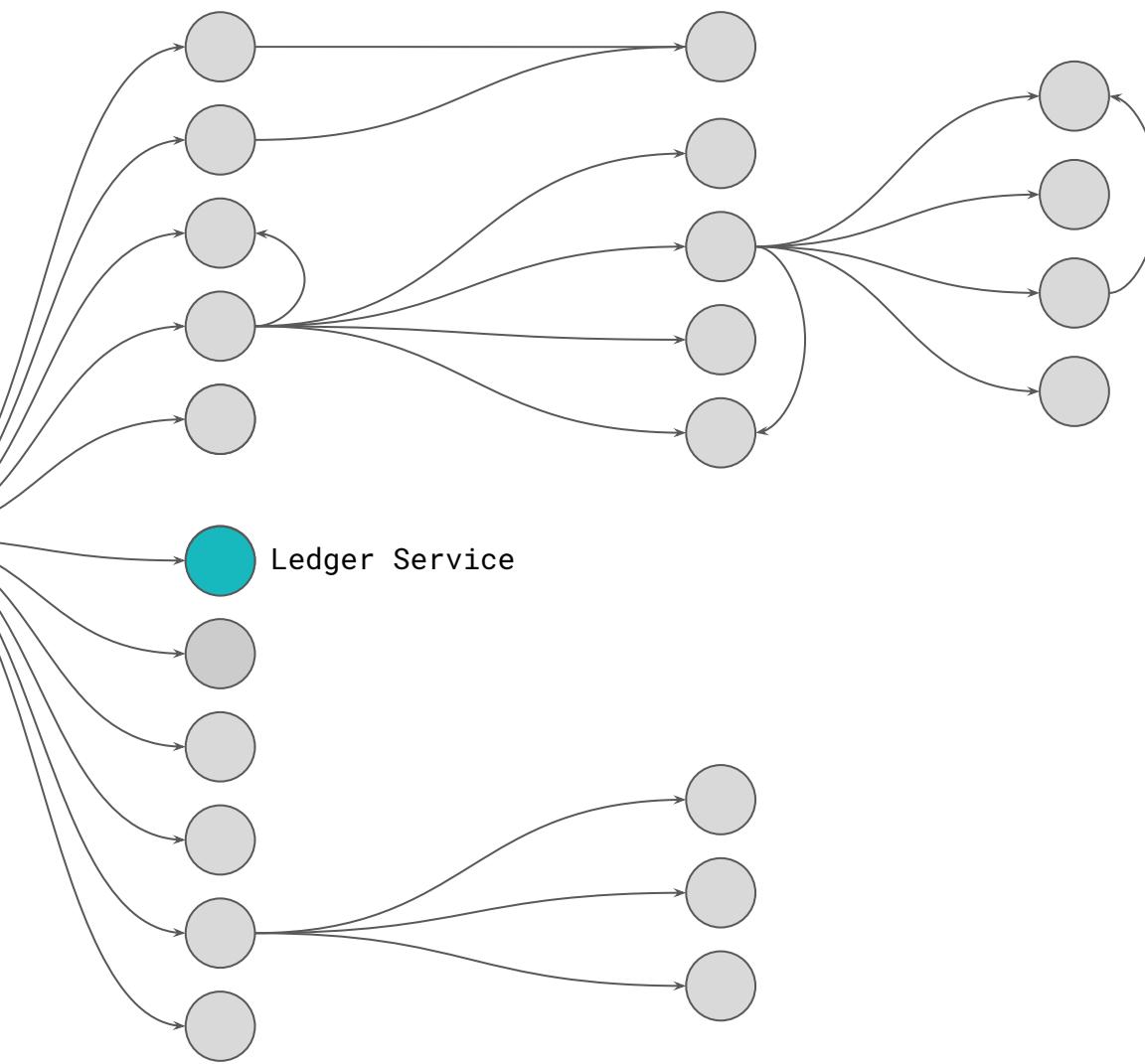
LEAH THOMPSON

5678

MasterCard

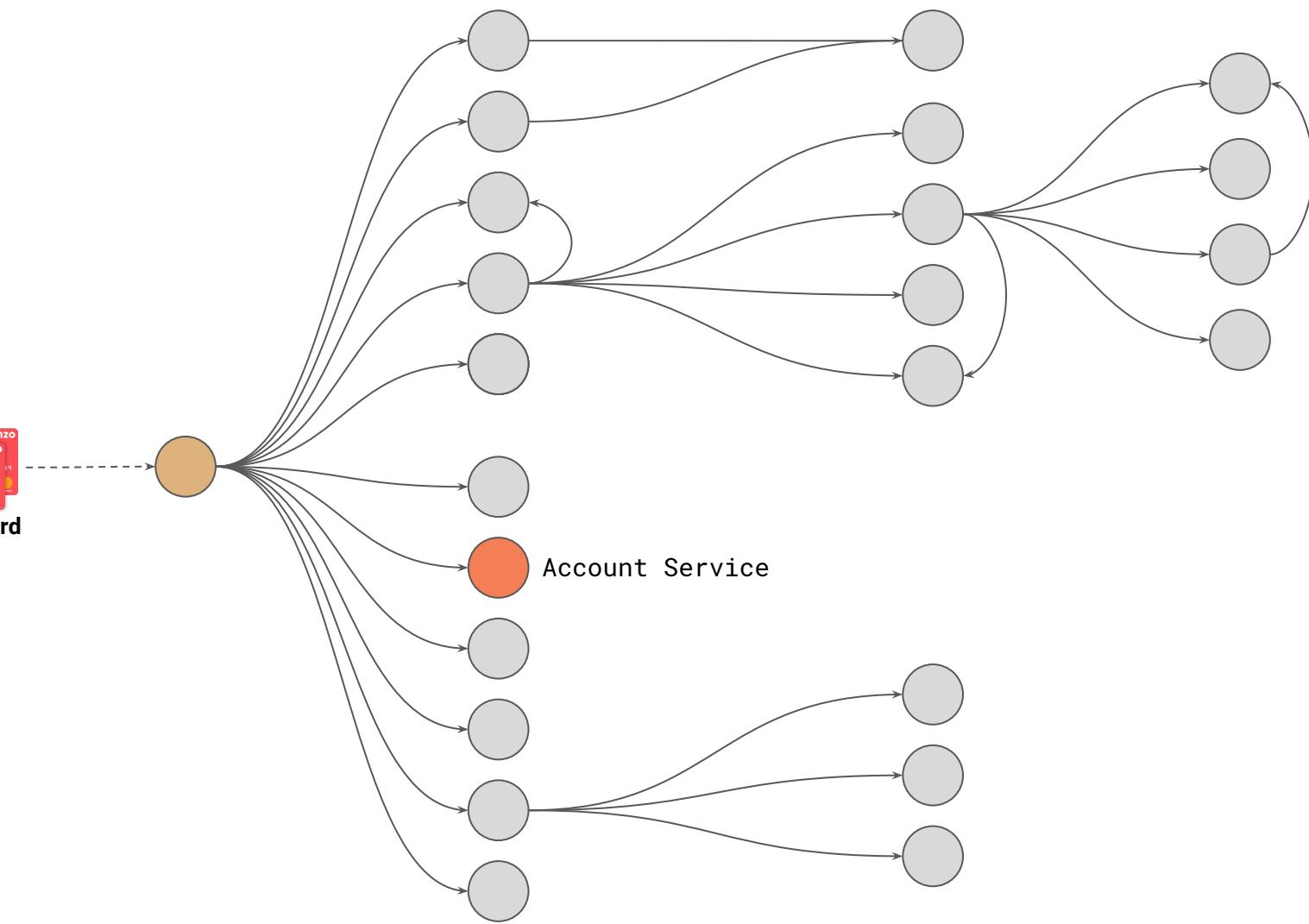


Ledger Service



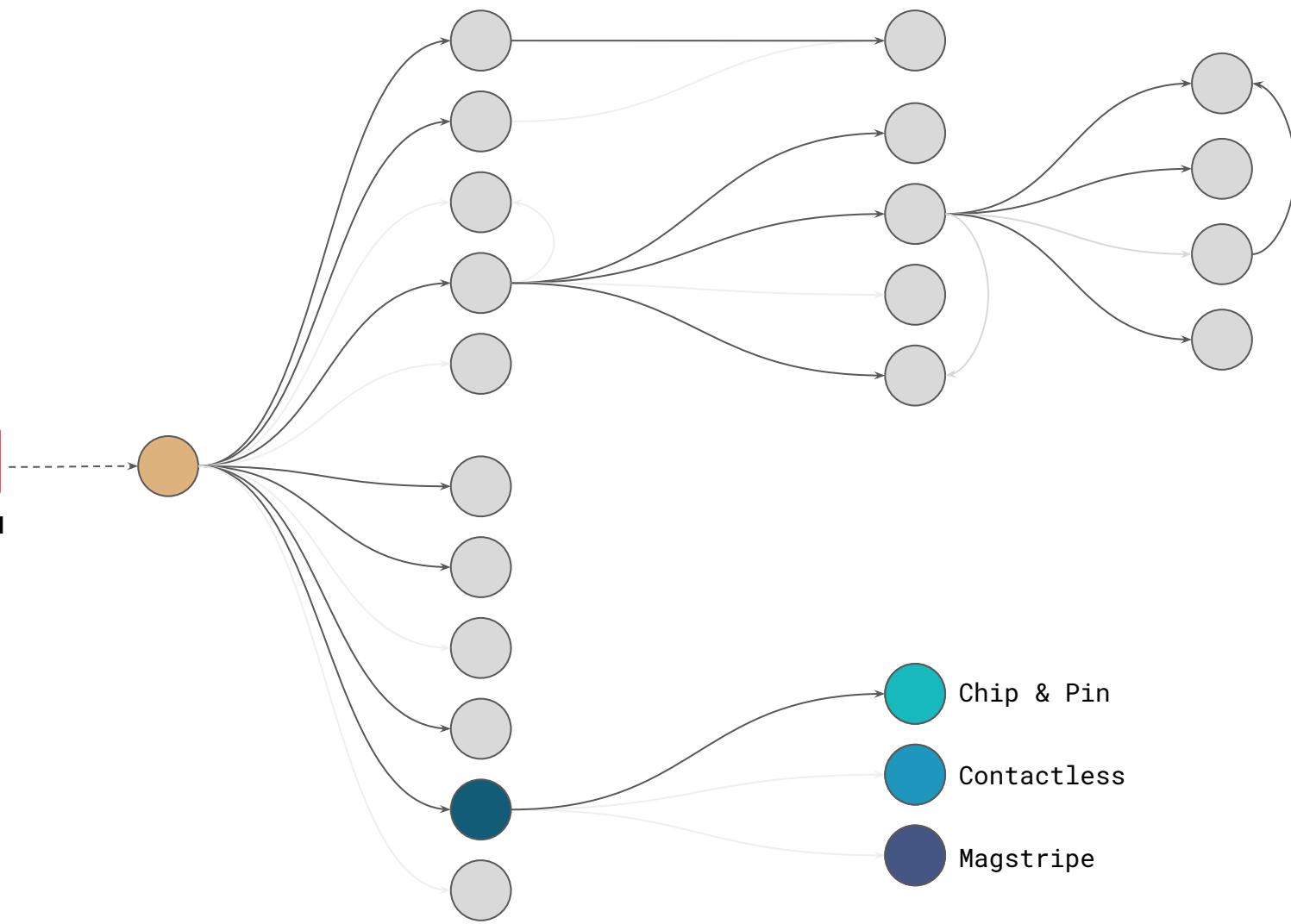


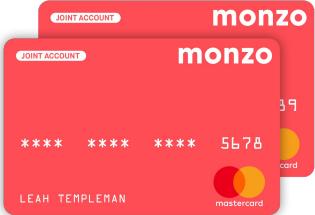
Monzo Card





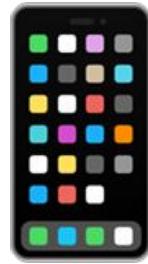
Monzo Card





Something You Know

Like a password or PIN number



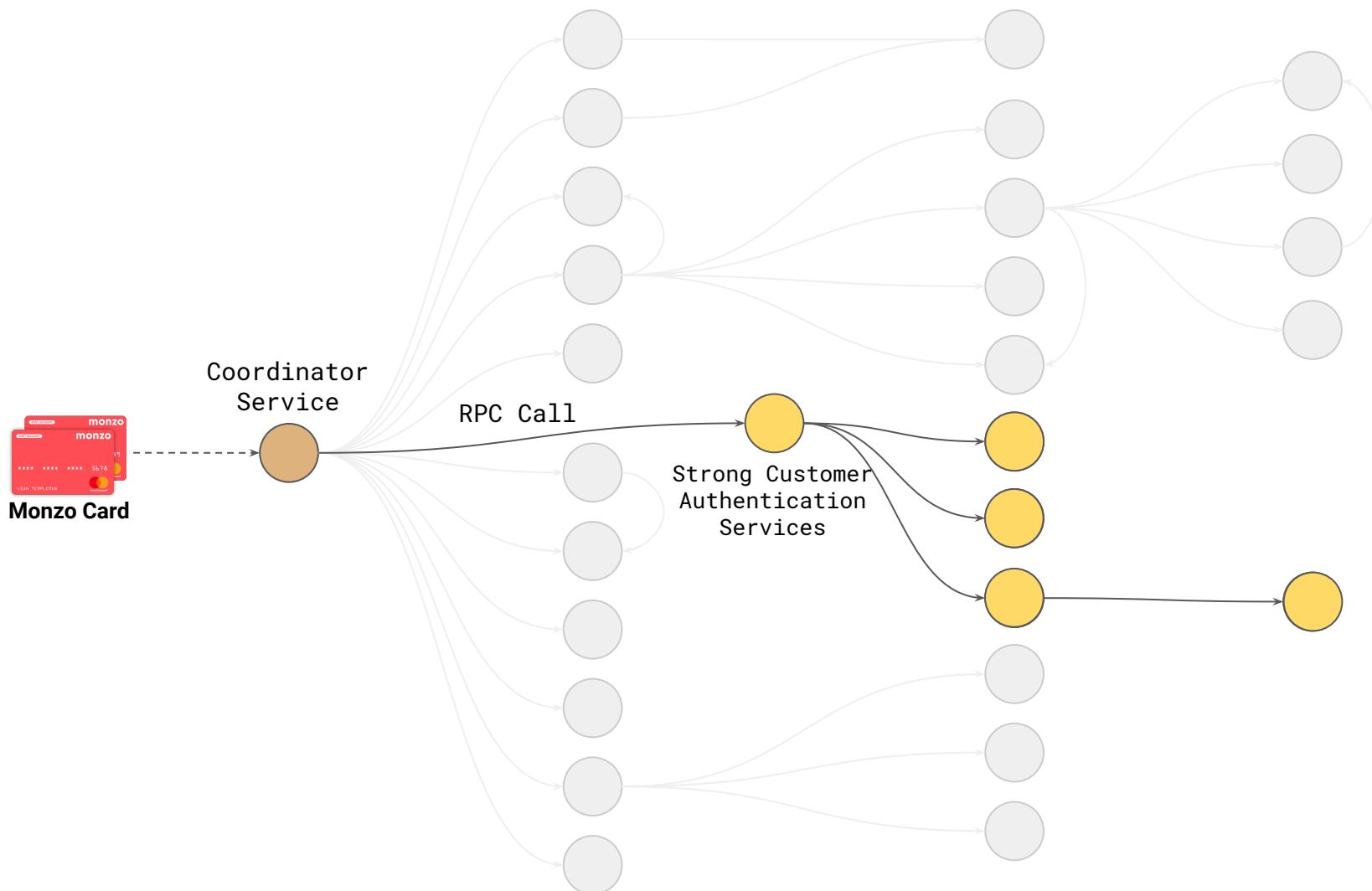
Something You Have

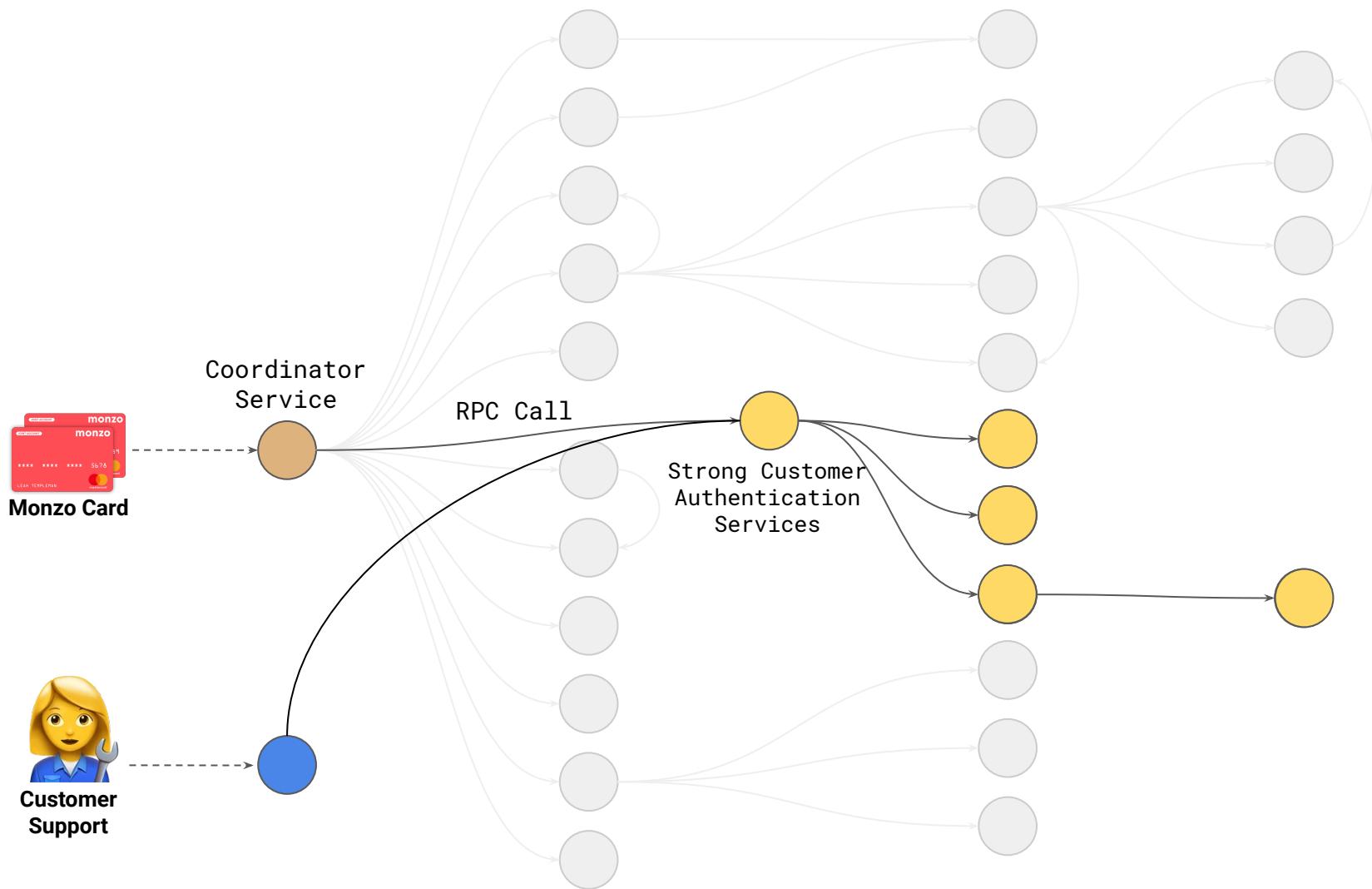
Like a mobile phone



Something You Are

Like a fingerprint or facial recognition







Security Team

Authorization flow
Mobile App Security
Decisioning
Monitoring



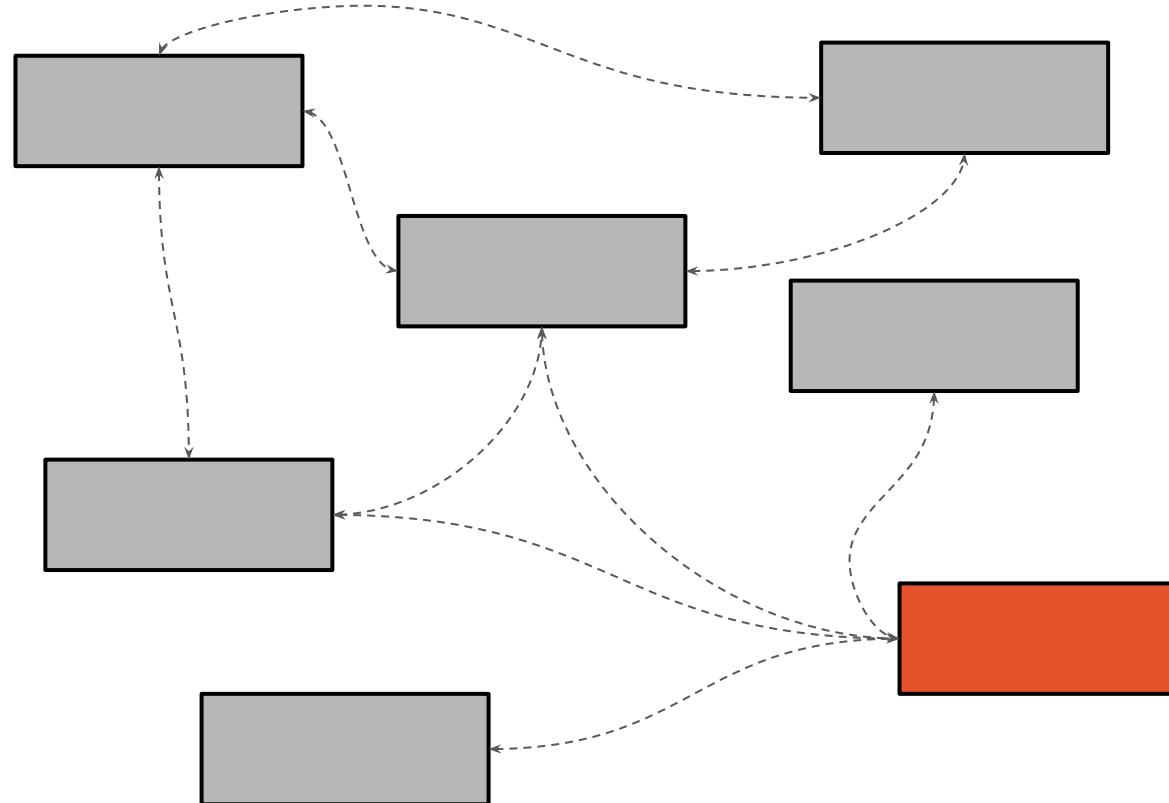
Product Team

App Experience
Notifications

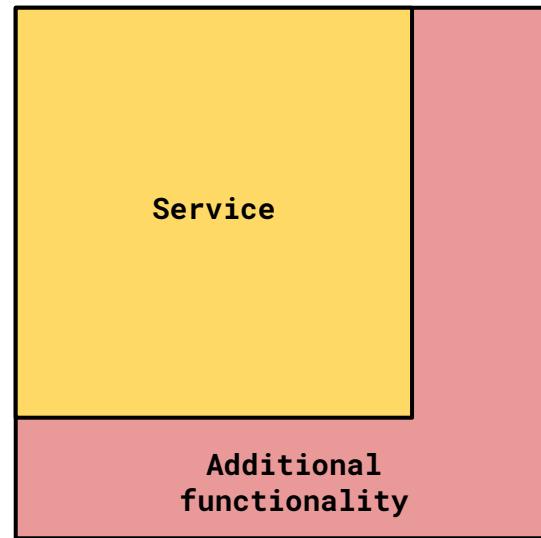


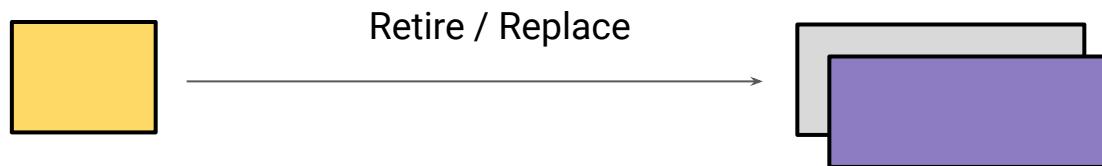
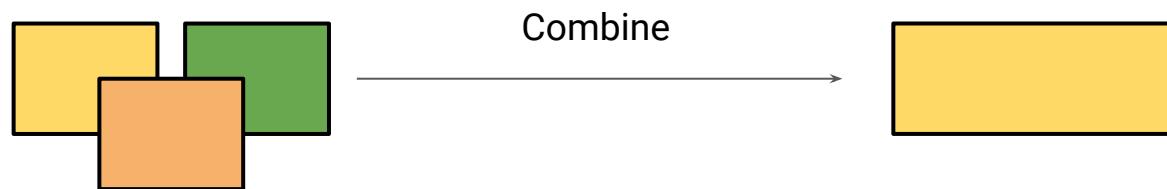
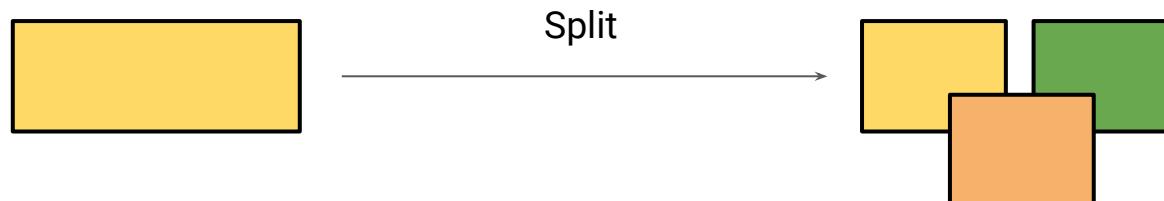
Payments Team

Mastercard
Faster Payments

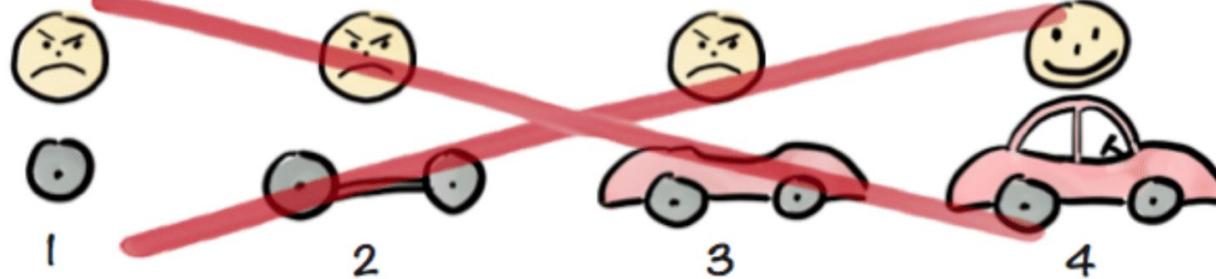


Service





Not like this....



Like this!

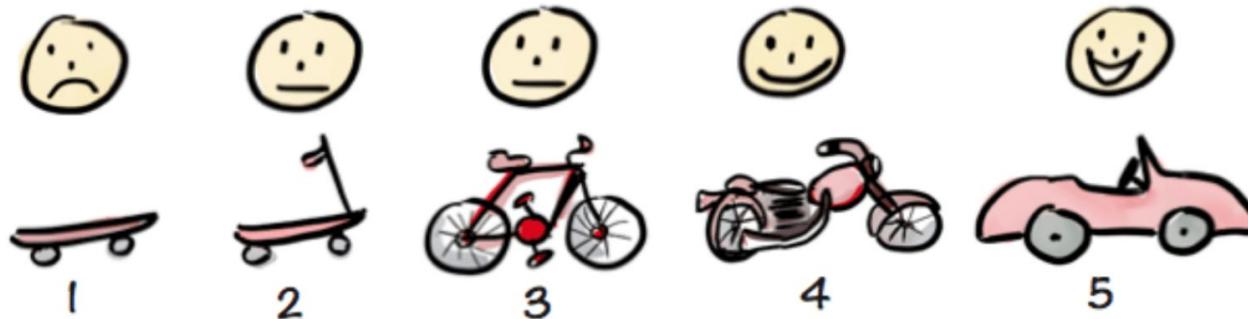
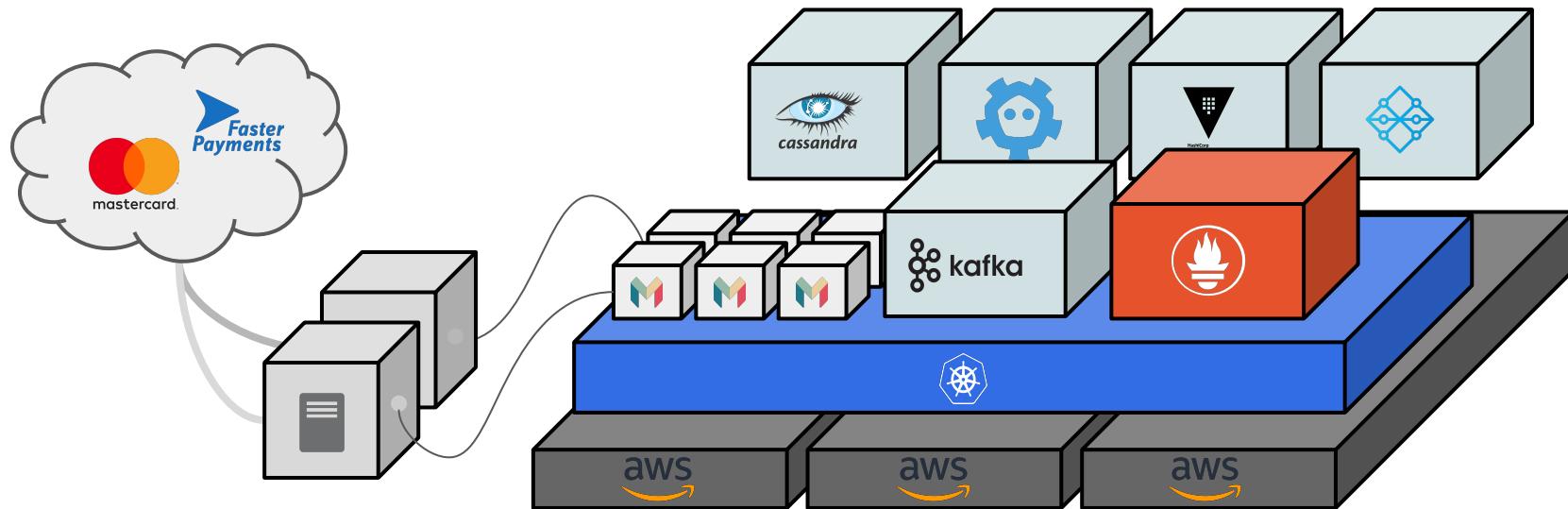


Image Credit: [Henrik Kniberg](#)



```
$ shipper deploy --prod 34720 service.aws-billing

─ Use wearedev repository
─ Looking up PR 34720
─ #34720: Expose billing data as Prometheus metrics
─ Merged by evnsio
─ aws-billing-metrics → master

─ Ready to deploy service.aws-billing b2ea82e557cc → cb4b9a47ef32

← To rollback, use:
shipper rollback --prod -r wearedev service.aws-billing

✓ Ready to deploy
  ↗ service.aws-billing
    └─ https://slog/?q=service%3Aservice.aws-billing
    └─ https://grafana/d/services?var-service=service.aws-billing

? deploy to prod?
```

```
syntax = "proto3";
package slackproto;

import "tools/protoc-gen-typgon/proto/typgon.proto";
import "tools/protoc-gen-validator/proto/validator.proto";

service slack {
    option (router).name = "service.slack";

    // Post a message to a channel
    rpc POSTMessage(PostRequest) returns (PostResponse) {
        option (handler).path = "/post";
    }

    // Returns message history for a given channel
    rpc ChannelHistory(ChannelHistoryRequest) returns (ChannelHistoryResponse) {
        option (handler).path = "/channel_history";
    }
}
```

```
$ ls -lah
lrwxr-xr-x@ 20 Jan 19:40 Makefile -> ../Makefile-service
-rw-r--r--@ 26 Nov 17:59 README.md
drwxr-xr-x@ 26 Nov 17:59 config
drwxr-xr-x@ 26 Nov 17:59 consumer
drwxr-xr-x@ 26 Nov 17:59 dao
drwxr-xr-x@ 26 Nov 17:59 domain
drwxr-xr-x@ 26 Nov 17:59 handler
-rw-r--r--@ 26 Nov 17:59 main.go
drwxr-xr-x@ 10 Feb 15:59 manifests
drwxr-xr-x@ 26 Nov 17:59 proto
```

```
$ svcquery
```

```
Learn things about services, based on your local git checkout.
```

Usage:

```
  svcquery [command]
```

Available Commands:

```
  analyze      Extract information from API and IAPI services.
```

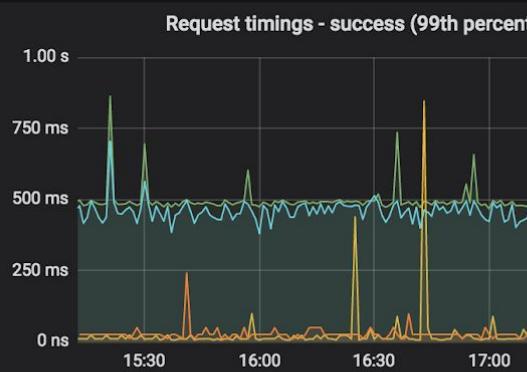
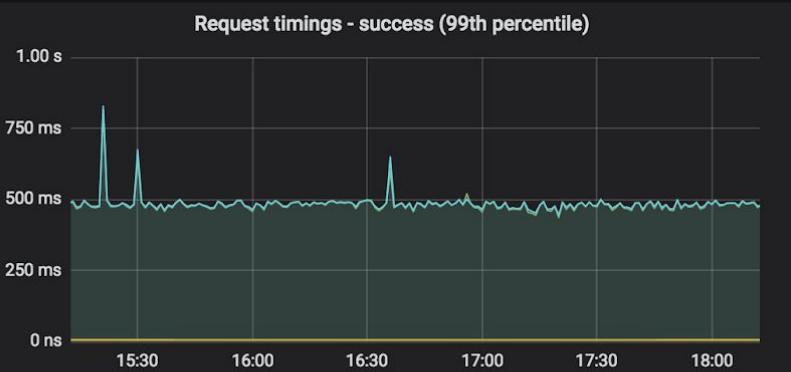
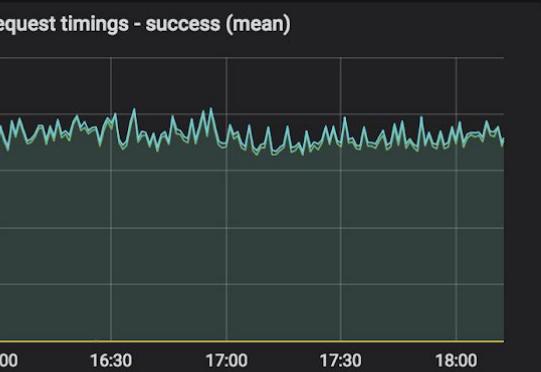
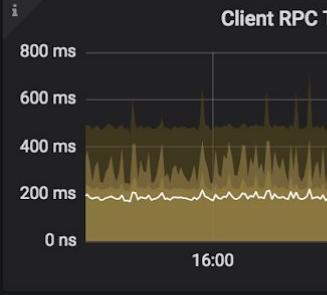
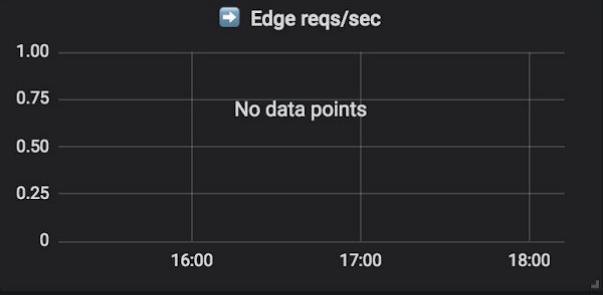
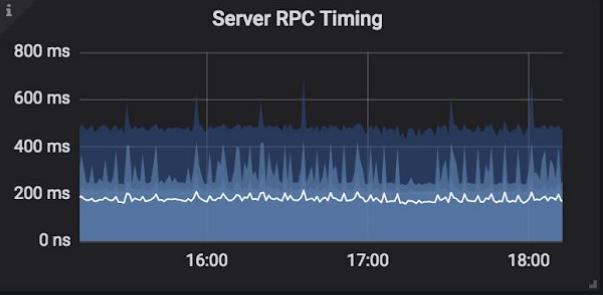
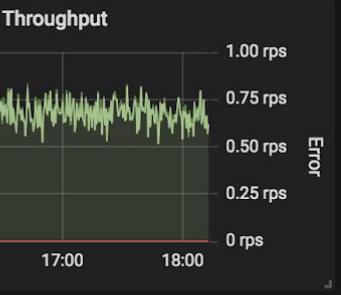
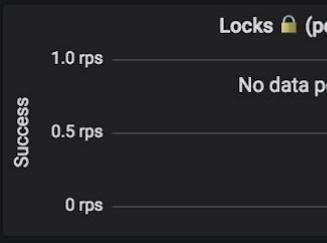
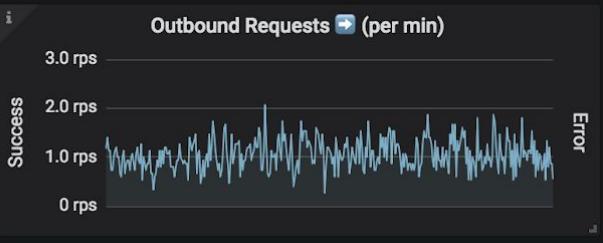
```
  endpoints    List the typhon endpoints of a service
```

```
  help         Help about any command
```

```
  history      Count the number of services and handlers at different points in time.
```

```
  owners       List the owners a service
```

Use "svcquery [command] --help" for more information about a command.



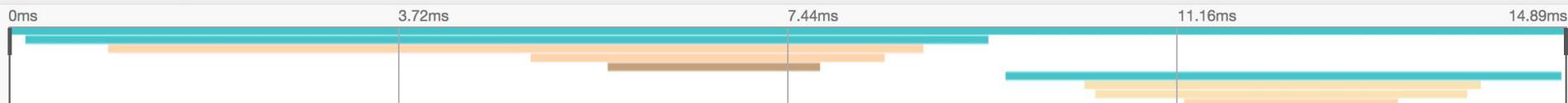


edge-proxy-external: GET edge-proxy-external/ping/whoami a6907c8

Find...



Trace Timeline ▾

Trace Start **February 23 2020, 20:49:35.255** Duration **14.89ms** Services **4** Depth **5** Total Spans **9**

Service & Operation	0ms	3.72ms	7.44ms	11.16ms	14.89ms
edge-proxy-external GET edge-proxy-e...					
edge-proxy-external POST service....					9.2ms
service.platform.authenticatio...					7.79ms
service.platform.authentic...					3.38ms
service.oauth2-client ...					2.03ms
edge-proxy-external GET service.a...					5.31ms
service.api.ping GET service.api....					3.79ms
service.api.ping POST serv...					3.55ms
service.platform.authe...					2.04ms



2020-02-28

09:31:37 ● DEBUG service.user-context list.go:24

[handler.handleGETList] Listing user contexts for user user_00009dcaydDRvjdsz01ULR

09:31:37 ● DEBUG service.user-context polyfill.go:22

[handler.polyfillUserContexts] Polyfilling contexts for user user_00009dcaydDRvjdsz01ULR, existing: [uc_00009oceGYGnrlldpUTYQHx]

09:31:37 ● ERROR service.account list_by_permissions.go:137

[handler.listAccountsByPermissions] Error reading additional accounts: internal_service: gocql: no response received from cassandra within timeout period

09:31:37 ● TRACE service.account errortracking.go:99

[errortracking.captureInSentry] Dropping event: internal_service: gocql: no response received from cassandra within timeout period

09:31:37 ● ERROR service.user-context list.go:36

[handler.handleGETList] Failed to polyfill missing contexts: internal_service: gocql: no response received from cassandra within timeout period

09:31:37 ● TRACE service.user-context errortracking.go:99

[errortracking.captureInSentry] Dropping event: internal_service: gocql: no response received from cassandra within timeout period

09:31:37 ● DEBUG service.user-context list.go:24

[handler.handleGETList] Listing user contexts for user user_00009dcaydDRvjdsz01ULR

09:31:37 ● DEBUG service.user-context polyfill.go:22

[handler.polyfillUserContexts] Polyfilling contexts for user user_00009dcaydDRvjdsz01ULR, existing: [uc_00009oceGYGnrlldpUTYQHx]

09:31:37 ● ERROR service.account list_by_permissions.go:137

[handler.listAccountsByPermissions] Error reading additional accounts: internal_service: gocql: no response received from cassandra within timeout period



elasticsearch

 **Grafana**



OpenTelemetry

The OpenTelemetry logo features a blue and orange microphone-like icon followed by the text "OpenTelemetry" in blue and orange.



Backend Engineering 101

⚠ It is important that you **first carefully complete all the backend onboarding steps**. This guide assumes this has been done. You will find yourself looking at obscure error messages if you don't.

Welcome to Backend Engineering 101 🎉 This tutorial will walk you through creating your first backend services at Monzo. It will teach you how to create a new service, implement RPC handlers, query Cassandra (our database), publish and consume messages from the Firehose (our pub/sub abstraction), write unit tests and deploy your code.

How to use this tutorial 🤔

Work through this tutorial at your own pace. As well as teaching you how to build backend services it will give you the opportunity to get comfortable with your development environment and the Go language (lots of us are new to Go when we join Monzo). Revisit the tutorial whenever you need to. The code you write here will serve as a useful example when you start picking up tasks in your team.

At the end of each chapter you'll find links to more in-depth resources. You can jump into these immediately if you'd like some more context, or you can revisit them later. You don't need to follow the links to complete the tutorial.

Don't forget to take regular breaks. Grab a coffee, go for a walk and chat to your fellow Monzonauts 😊



Getting help 🚧

If you've got any questions you can post them in the `#eng-onboarding` channel on Slack. To make it easy for others to help, start your message with one of the following emoji:

- ❓ I've got a question and it's blocking me from moving on in the tutorial
- ❔ I've got a question but it's not blocking me
- 💡 I've got a suggestion for improving the tutorial

You can also get help from your engineering buddy. They'll be happy to pair up with you at any point 🤝

```
INFO: ⏳ Waiting for config before serving...
INFO: 🚔 Admin server listening on 127.0.0.1:49545
INFO: ⏱ Config loaded (took 8.961748ms)
INFO: 📈 Enabling prometheus registry
INFO: 👋 service.api.teapot listening on 127.0.0.1:49548
```





Improved Organisational Flexibility

Microservices which are granular and well understood

Consistent code structure and tooling



Focus on the problem

Standardising enables engineers to focus on the business problem

Continuously improving tools and abstractions



Increase velocity whilst reducing risk

Make and deploy a series of small and iterative changes

Break down complexity and reduce risks

Thanks!

Matt Heath (@mattheath)

Suhail Patel (@suhailpatel)

