The Story of why we migrate to gRPC and how we go about it

KubeCon

CloudNativeCon

Europe 2019

@mattgruter

2019-05-22

“Developers don’t care about new RPC technologies”

-- someone at KubeCon 2018

The Story of why we migrate to gRPC and how we go about it

KubeCon

CloudNativeCon

Europe 2019

@mattgruter

2019-05-22

Why?

Why are we doing this?

How?

How do we get there?

Why?

Spotify's Infrastructure

~2.5k services

~1k developers

~250 teams

Java, Python, ...

Spotify's Infrastructure

~2.5k services

~1k "developers"

~250 teams

Java, Python, ...

Hermes

Not this Hermes!

Hermes

But this

Hermes

Written in 2012

Based on ZeroMQ

JSON or protobuf payload

Not a RPC framework

Hermes works!

Why Change?

Hermes Ecosystem

404 Not Found

From DIY to FOSS

Why gRPC?

Why gRPC?

CLOUD NATIVE
imgflip.com

The gRPC Advantage

The Proto

Code Generation

Java, Golang, Python, Ruby,
Dart, PHP, Node.js,
Objective-C, C#, C++

Schema Management

1. Embrace the **proto**

2. Shared repo for all **protos**

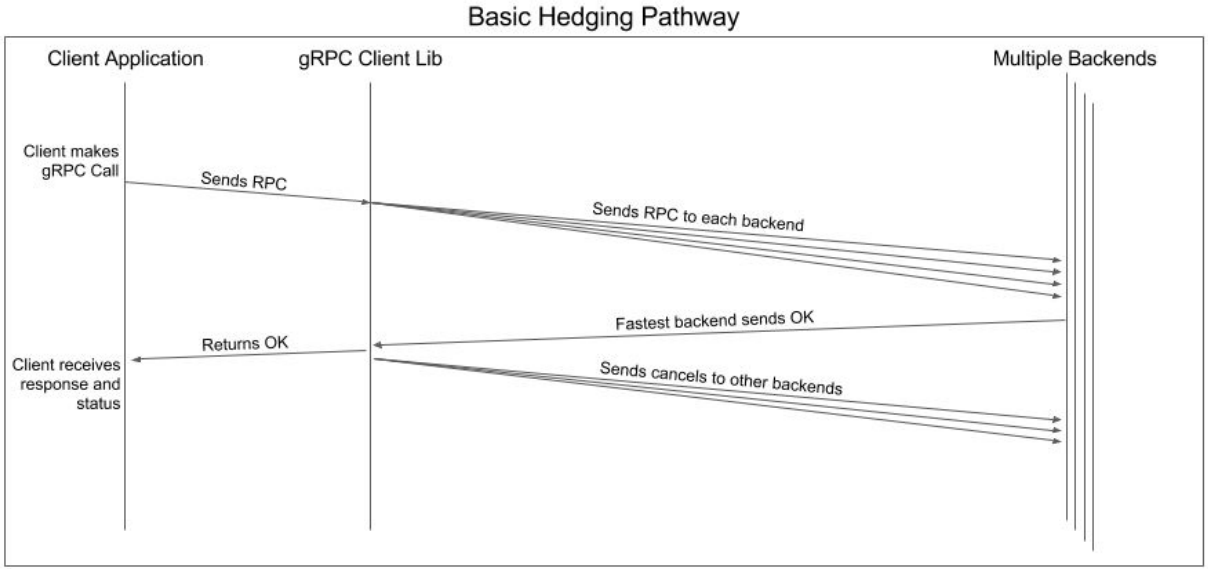
3. Version on the **proto** level

Resiliency

Retries

- Transparent
- Configurable

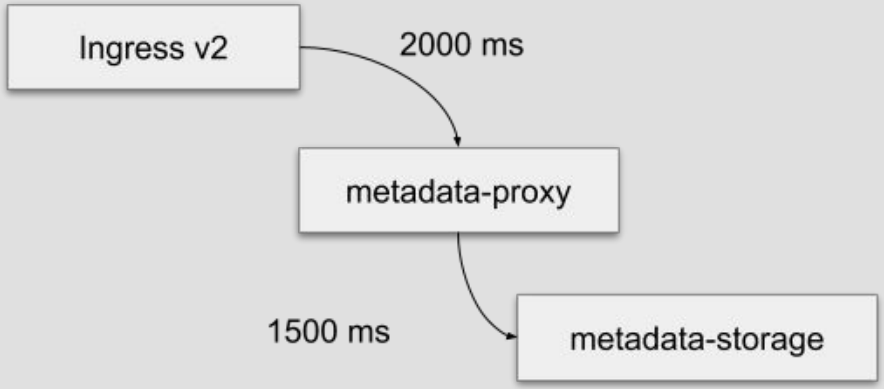
Hedging


Basic Hedging Pathway
Client Application
gRPC Client Lib
Multiple Backends
Client makes gRPC Call
Sends RPC
Sends RPC to each backend
Fastest backend sends OK
Returns OK
Client receives response and status
Sends cancels to other backends

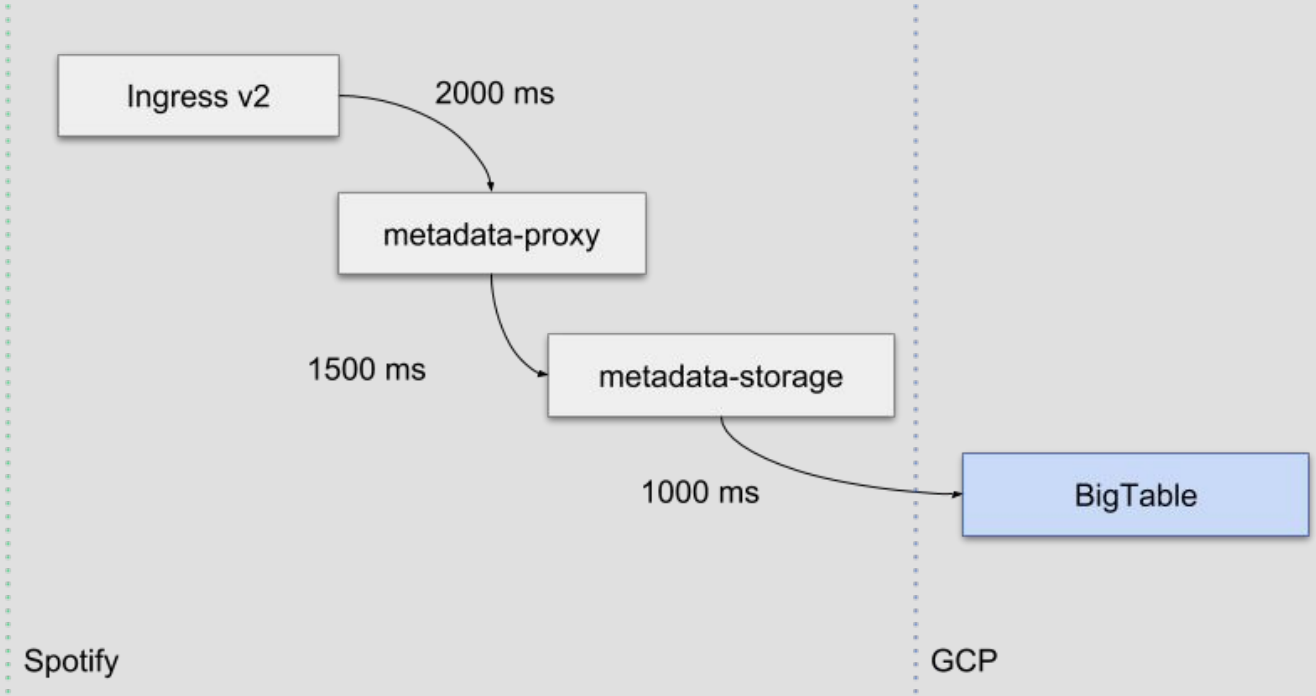
Thundering Herd

Retry throttling

Deadlines



A common RPC


Ingress v2
2000 ms
metadata-proxy
1500 ms
metadata-storage
1000 ms
BigTable
Spotify
GCP

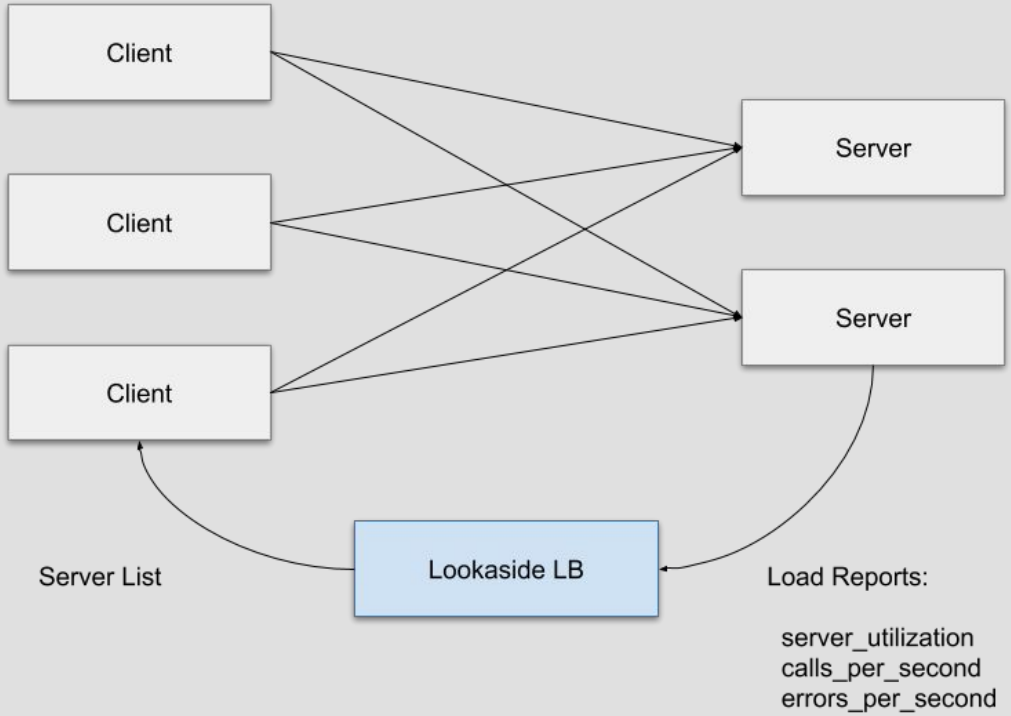
Load Balancing & Routing

Load Balancing

- Client-side
- Proxy
- Lookaside

Load Balancing

lookaside



The gRPC Advantage

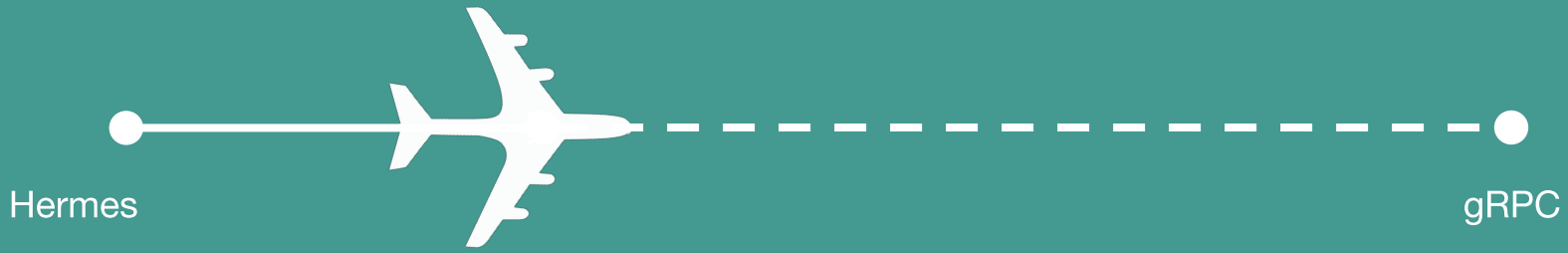
gRPC

TO PROTOC & SERVE

Spotify
gRPC

How?

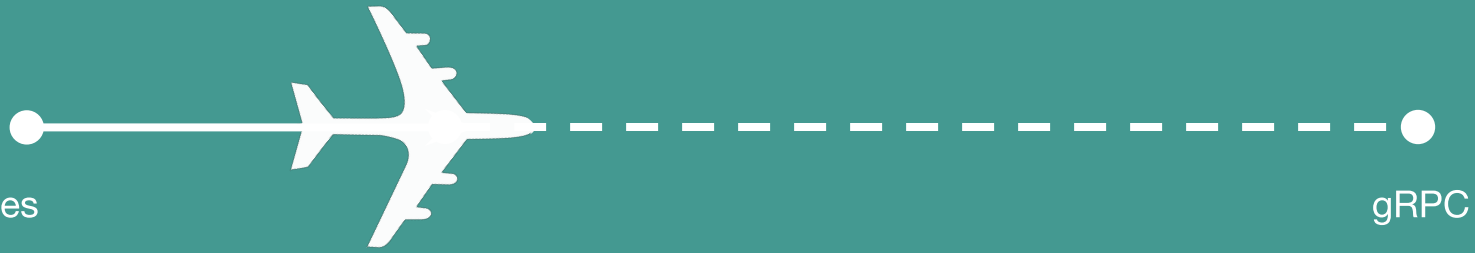
Our Journey



Our Journey

10%
Distance Travelled

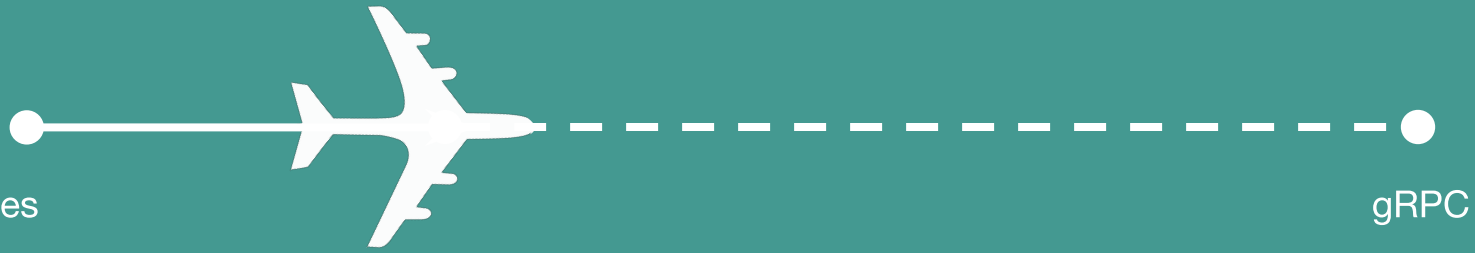
90%
Remaining Distance to Destination



Our Journey

10%
Distance Travelled

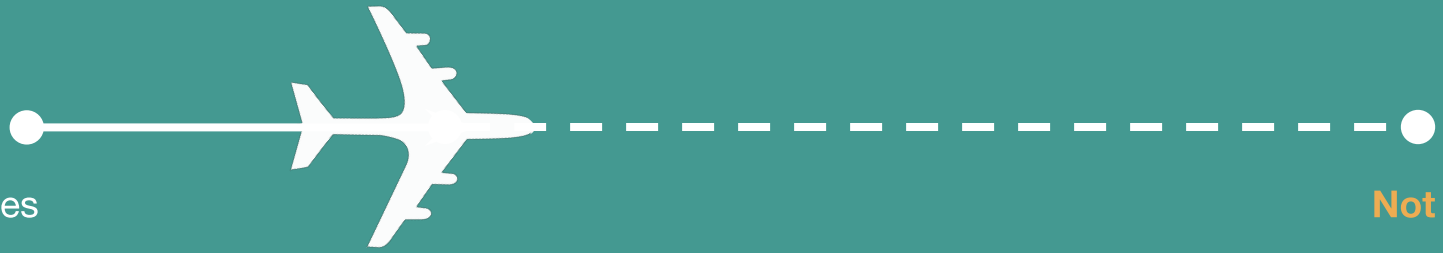
90%
Remaining Distance to Destination



Our Journey

10%
Distance Travelled

90%
Remaining Distance to Destination



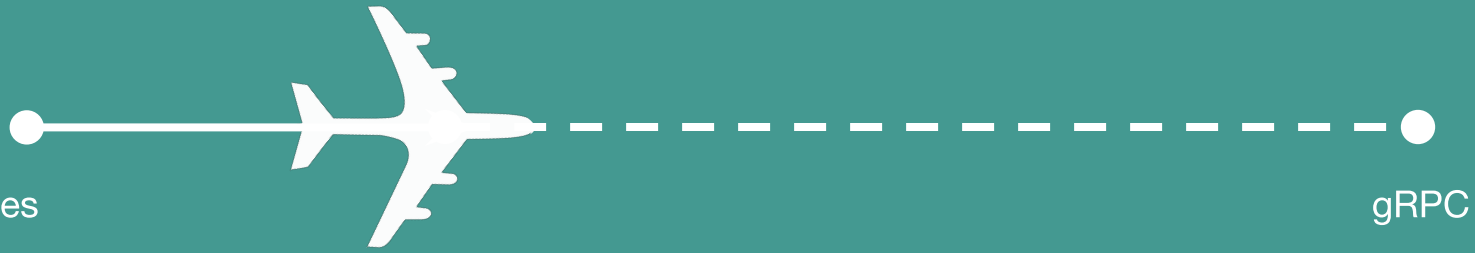
Hermes

Not Hermes

Our Journey

10%
Distance Travelled

90%
Remaining Distance to Destination



Main Challenge

Change is hard

“This is not the protocol you’re looking for”

Main Challenge

Change is hard

Trust

Aligned Autonomy

Aligned Autonomy

Autonomy != Freedom

Alignment != Consensus

Aligned Autonomy

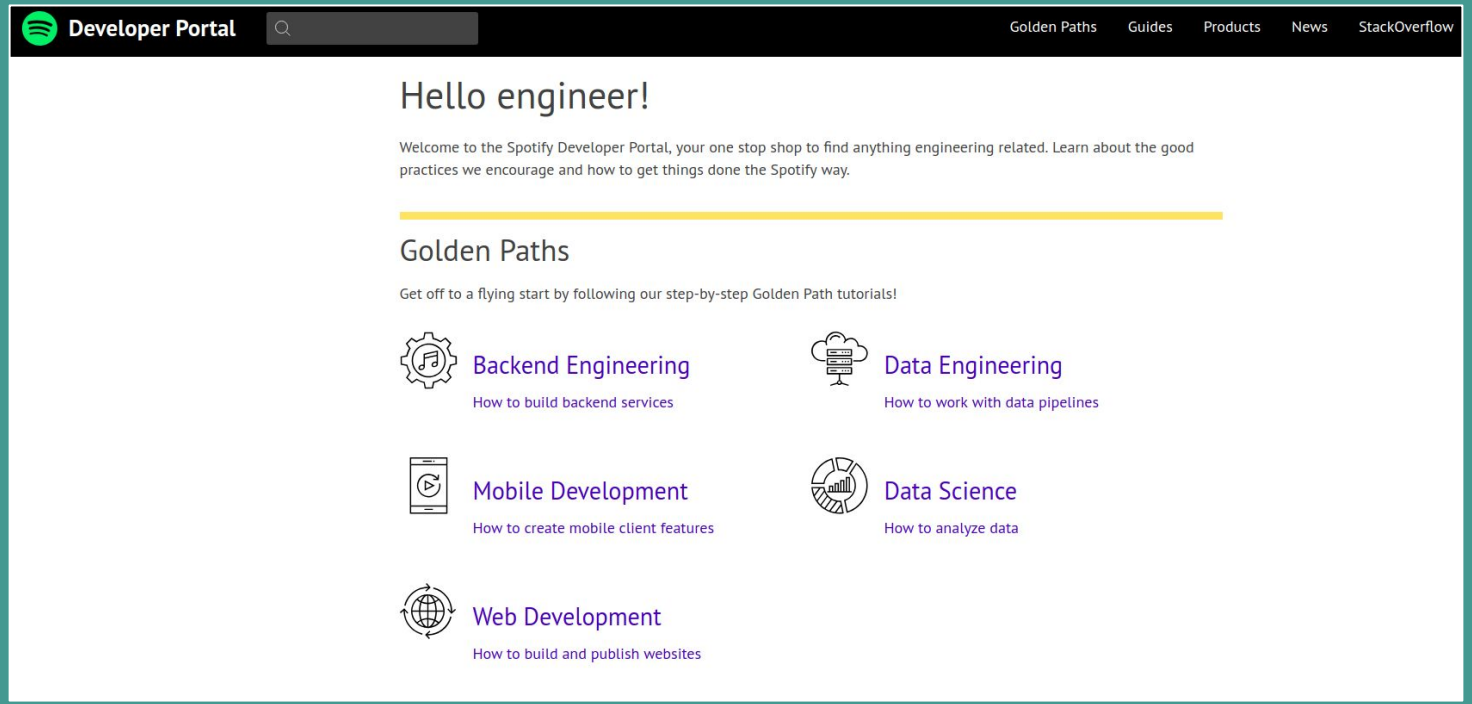
Autonomy != Freedom

Align by making

the **right** choice

the **easy** choice.

1. Right Choice = Easy Choice



2. Don’t force it!

2. Don’t force it!

`code generation`

2. Don’t force it!

Resiliency patterns

code generation

2. Don’t force it!

Tracing

Resiliency patterns

code generation

2. Don’t force it!

Tracing

Resiliency patterns

Istio

code generation

In case of emergency

aka. ever-ending migration
aka. yet another protocol

gRPC

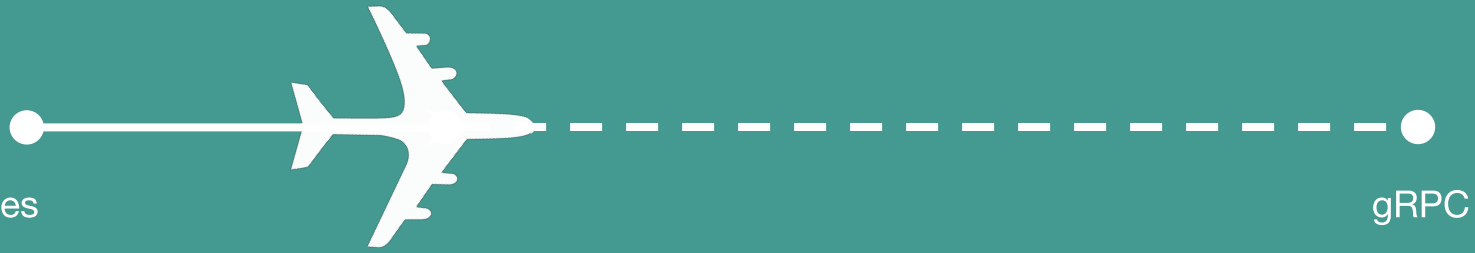
3. Force it!

$ sudo migrate

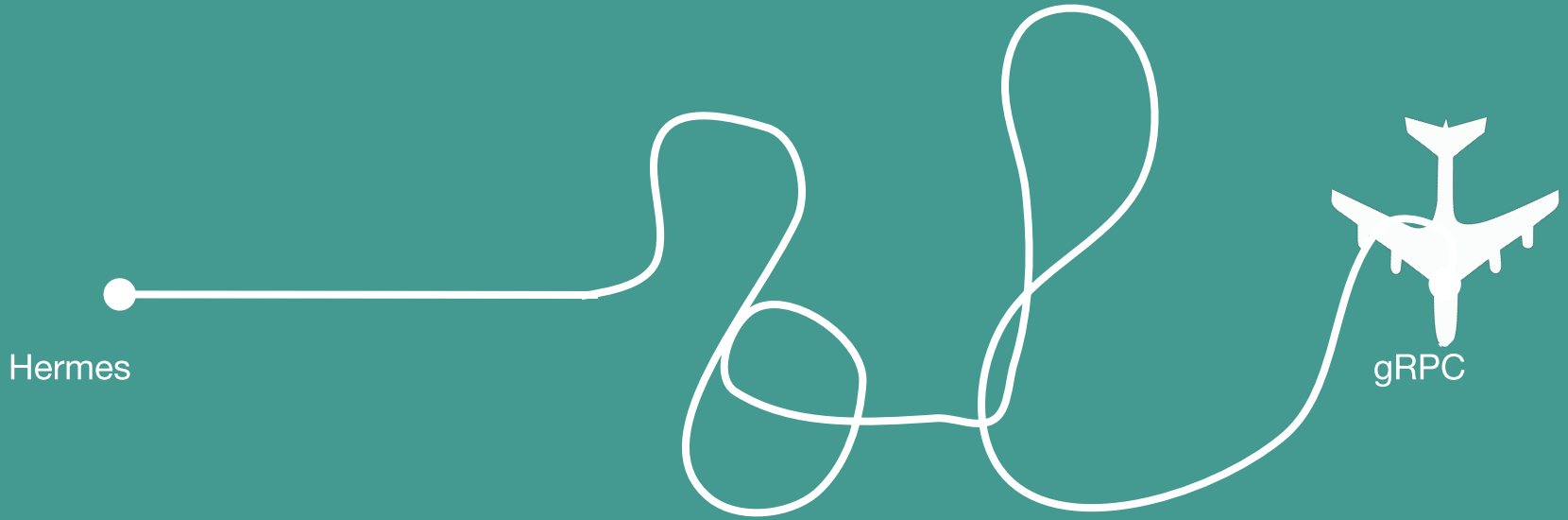
Our Journey

10%
Distance Travelled

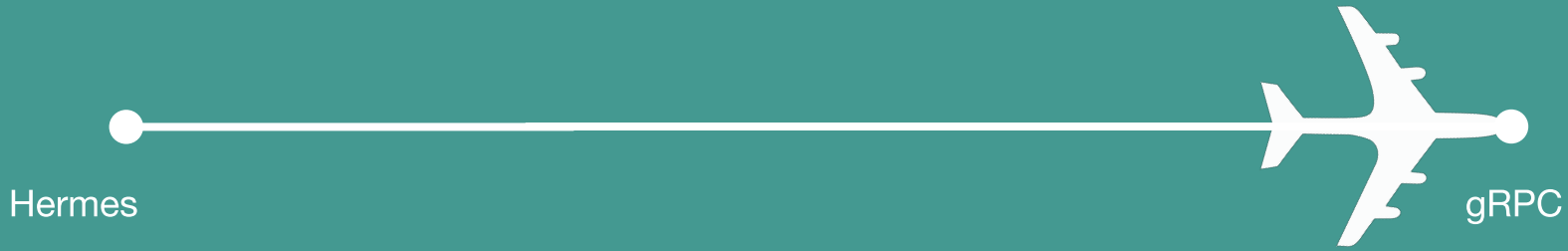
90%
Remaining Distance to Destination



Our Journey



Our Journey



“Developers don’t care about new RPC technologies”

-- overheard at KubeCon 2018 Seattle

“Developers don’t have to care about new RPC technologies”

-- “overheard” at KubeCon 2019 Barcelona

Thank You

Matthias Grüter
@mattgruter