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?

What?

What do we get out of this?

How?

How do we get there?

Spotify's Infrastructure

```
~2500 services

~1000 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 NIH to OSS

Why gRPC?

CLOUD NATIVE
imgflip.com

gRPC

1. HTTP/2 based
2. Binary protocol
3. Strongly typed service and message definition (Protobuf)
4. Encryption

The gRPC Advantage

GRPC

The Proto

Code Generation

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

Protobuf

```
syntax = "proto3";

package spotify.metadata.v1;

option java_package = "com.spotify.metadata.v1"
option java_multiple_files = true;
option java_outer_classname = "MetadataProto";

// Interface exported by the server.
service Metadata {
 rpc GetMetadata(SongId) returns (SongMetadata) {}
}

message SongId {
 int32 id = 1;
}

message SongMetadata {
 int32 id = 1;
 string name = 2;
 string artist = 3;
 string album = 4;
}
```

Server Logic (Java)

```java
public class MetadataService extends MetadataGrpc.MetadataImplBase {

    // [...]

    @Override
    public void getMetadata(SongId songId,
                                StreamObserver<SongMetadata> response) {
        LOG.info("Received getMetadata request");
        response.onNext(store.searchMetadata(songId)
                .orElse(EMPTY_METADATA));
        response.onCompleted();
    }

}
```

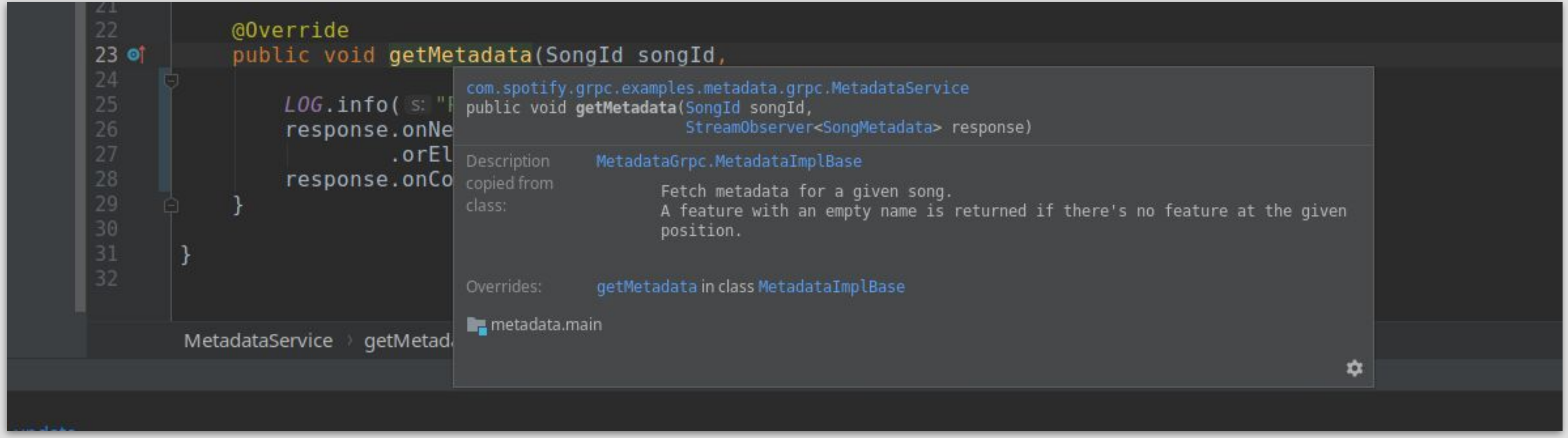
Polyglot client impl.

```go
func main() {
   ctx, cancel := context.WithTimeout(
       context.Background(), time.Second)
   defer cancel()
    )

    conn, err := grpc.Dial("nls://metadata")
    if err != nil {
        log.Fatalf("couldn't connect to grpc server: %v", err)

    }
    defer conn.Close()

    client := pb.NewMetadataServiceClient(conn)
    r, err := client.Metdata(ctx,
        &pb.SongId{
            Id: "42"
        },
    )
    if err != nil {
        log.Printf("metadata request failed: %v\n", err)
        return
    }
    fmt.Println(r.Response)
}
```



```
    @Override
    public void getMetadata(SongId songId,

        LOG.info( s: "F
        response.onNe
                .orEl
        response.onCo
    }

}
```

MetadataService › getMetad

```
com.spotify.grpc.examples.metadata.grpc.MetadataService
public void getMetadata(SongId songId,
                        StreamObserver<SongMetadata> response)
```

Description copied from class: MetadataGrpc.MetadataImplBase

```
Fetch metadata for a given song.
A feature with an empty name is returned if there's no feature at the given
position.
```

Overrides: getMetadata in class MetadataImplBase

metadata.main

Schema Management

1. Embrace the **proto**

2. Shared repo for all **protos**

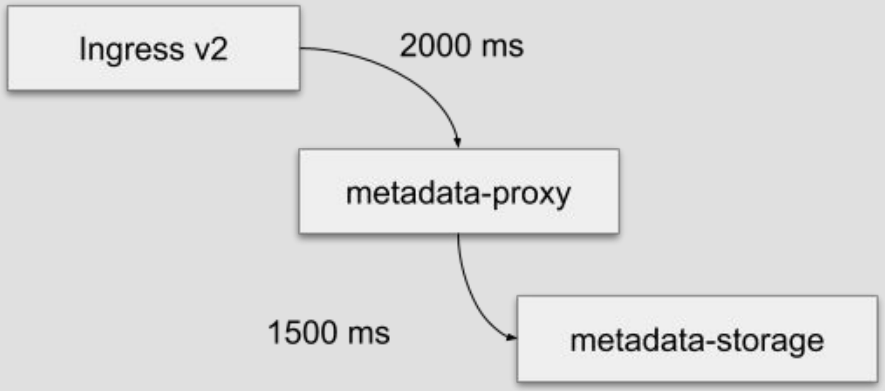
3. Version on the **proto** level

Schema Management

github.com/uber/prototool

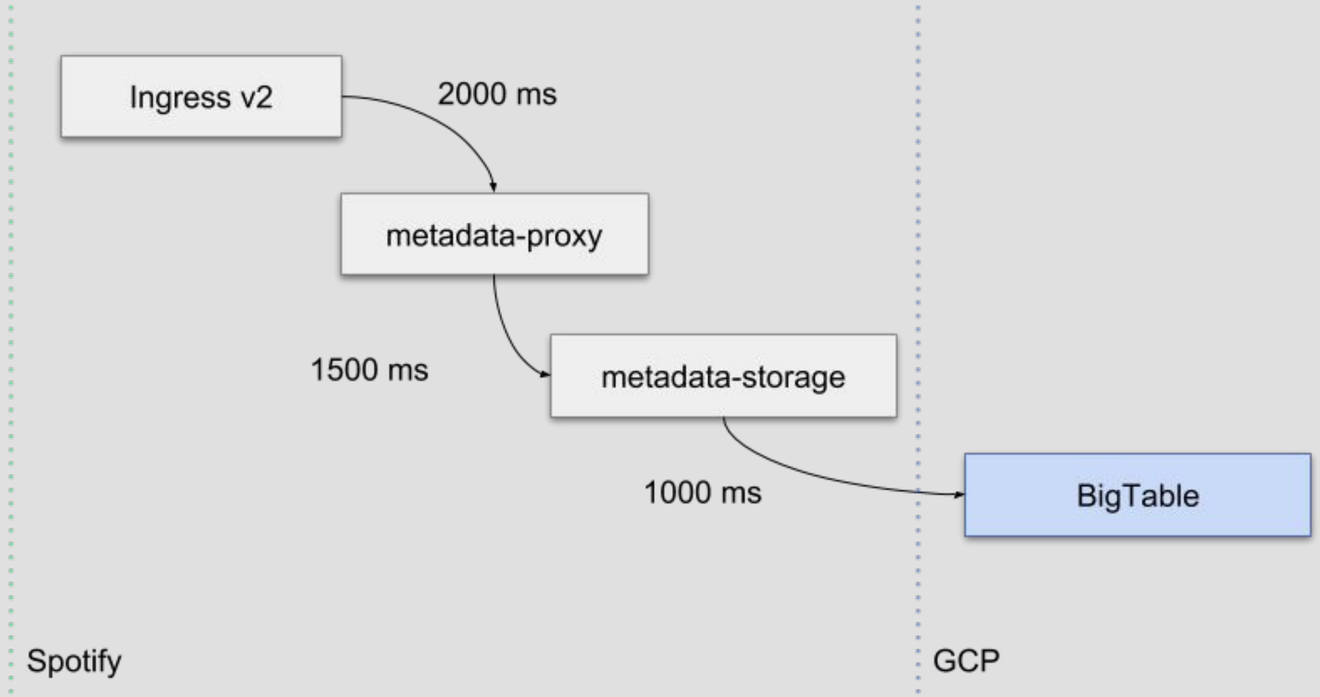
Resiliency

Deadlines



Spotify

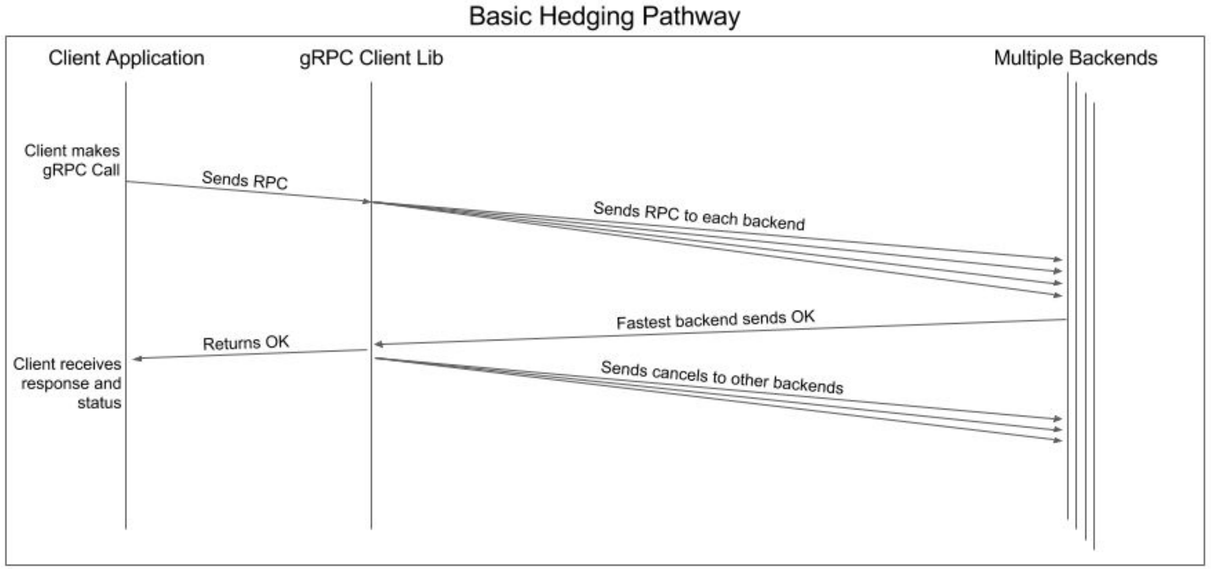
A common RPC


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

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

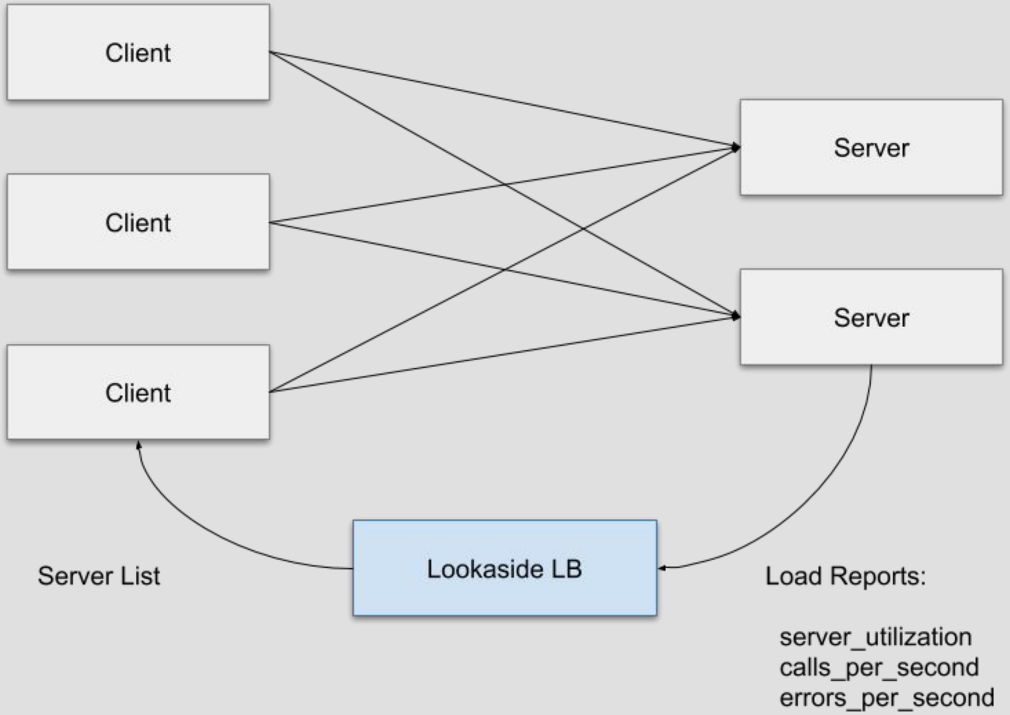
Load Balancing & Routing

Load Balancing

- Client-side
- Proxy
- Lookaside

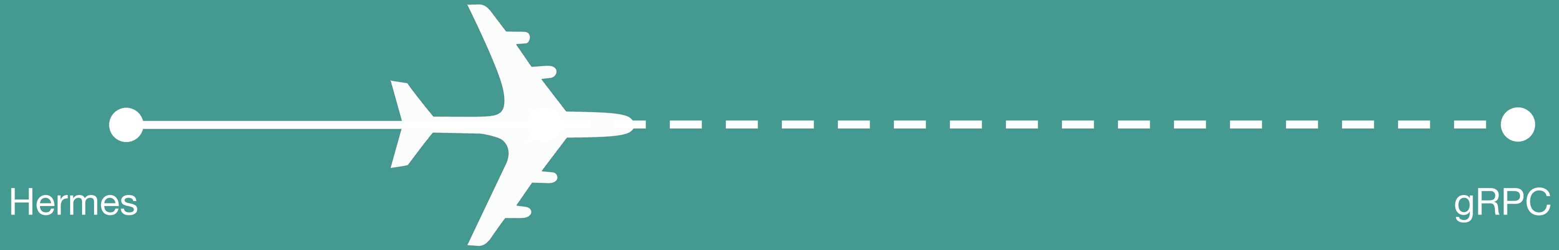
Load Balancing

lookaside



How to Migrate?

Our Journey



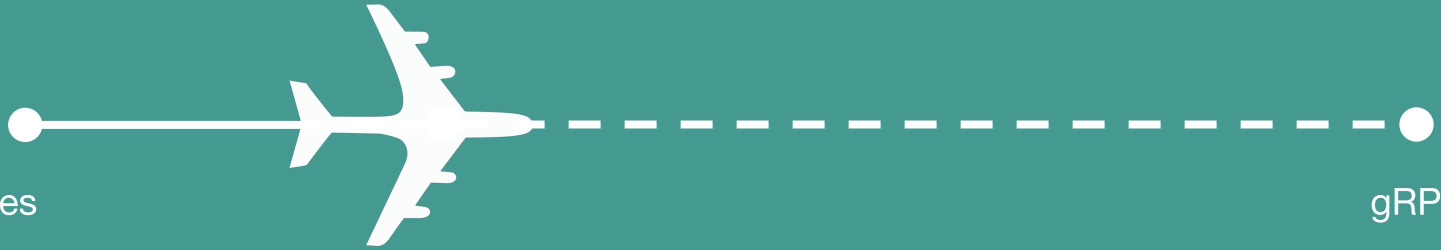
Stats

2874	Services
1341	HTTP
983	Hermes
76	gRPC

Our Journey

80 services
Distance Travelled

900 services
Remaining Distance to Destination



Challenge #1

Change is hard

1. Don’t force it!

1. Don’t force it!

`code generation`

1. Don’t force it!

Resiliency
patterns

code generation

1. Don’t force it!

Tracing

Resiliency
patterns

code generation

1. Don’t force it!

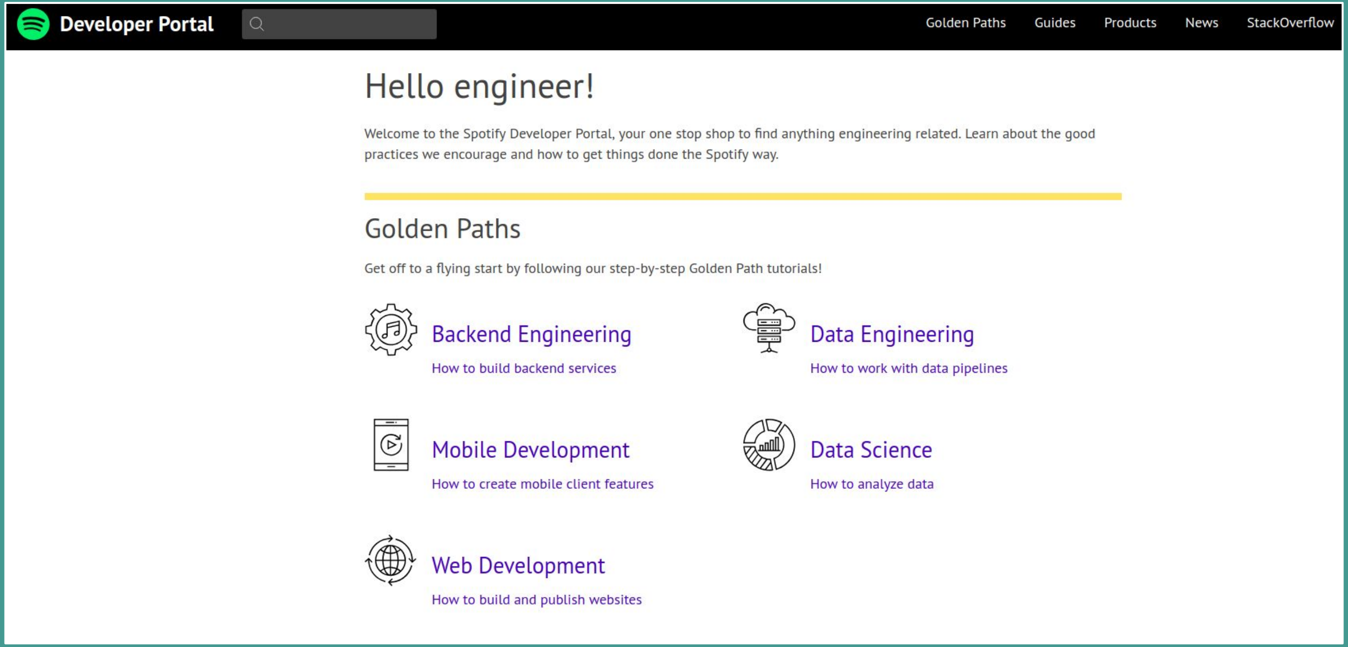
Tracing

Resiliency
patterns

Istio

code generation

2. Make the Right Choice the Easy Choice



Hello engineer!

Welcome to the Spotify Developer Portal, your one stop shop to find anything engineering related. Learn about the good practices we encourage and how to get things done the Spotify way.

Golden Paths

Get off to a flying start by following our step-by-step Golden Path tutorials!

Backend Engineering
How to build backend services

Data Engineering
How to work with data pipelines

Mobile Development
How to create mobile client features

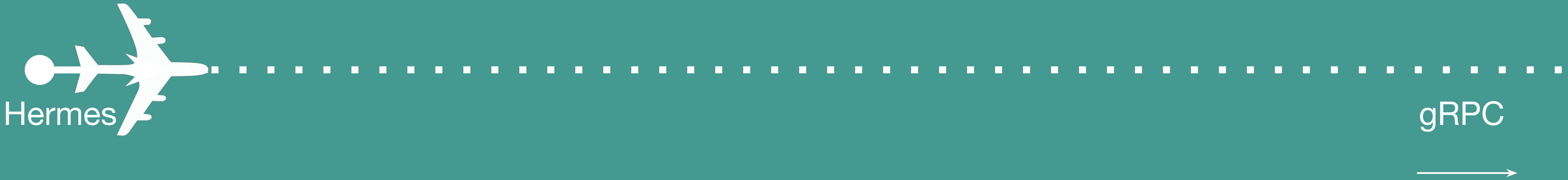
Data Science
How to analyze data

Web Development
How to build and publish websites

Challenge #2

Yet another protocol

Never-ending migration



Step by step

1. Add a new gRPC API
2. Move clients to new API
3. Remove old API

Step by step

1. Add a new gRPC API
2. Move clients to new API ← this is hard!
3. Remove old API

Challenge #3

Developer experience

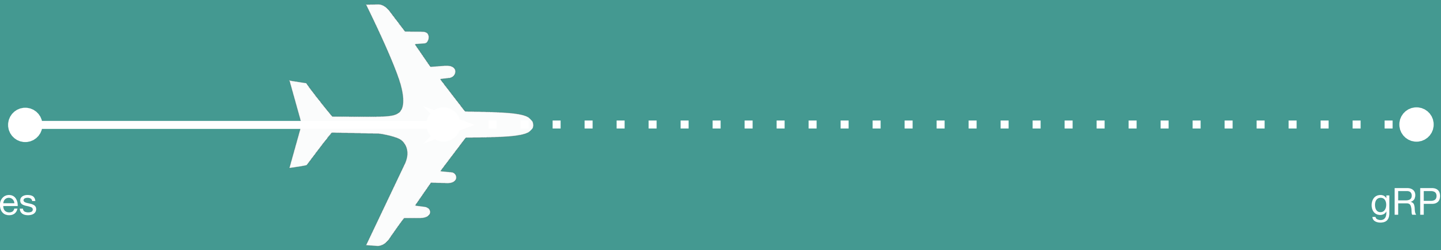
Our Journey

80 services

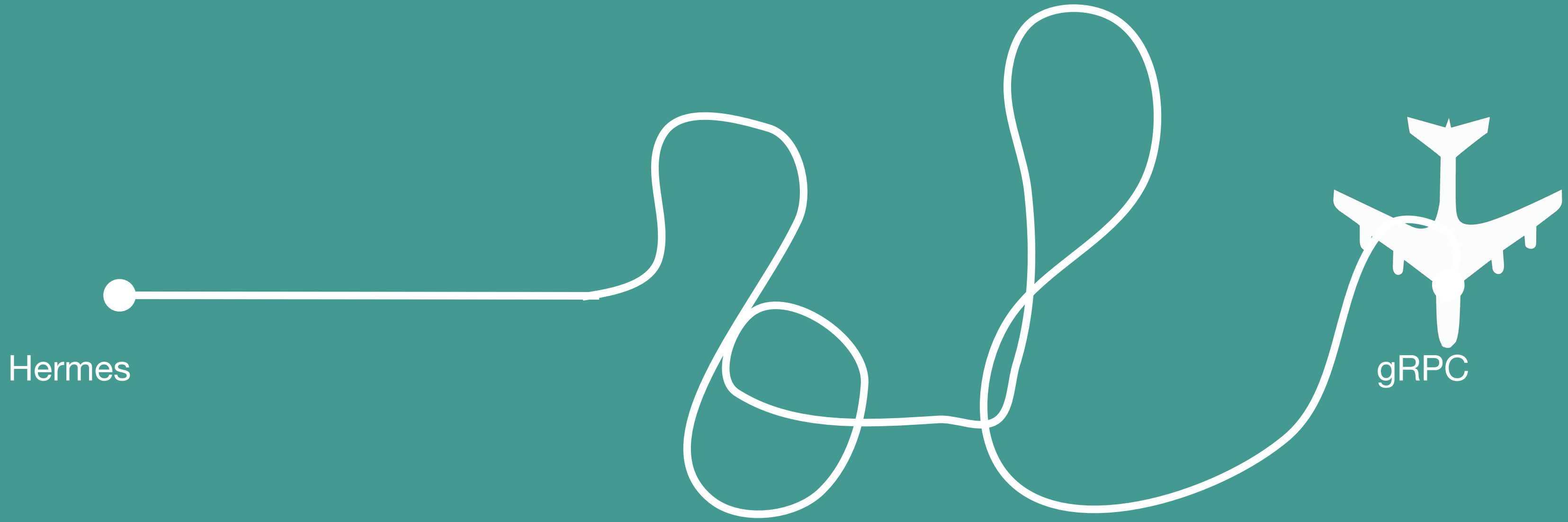
Distance Travelled

900 services

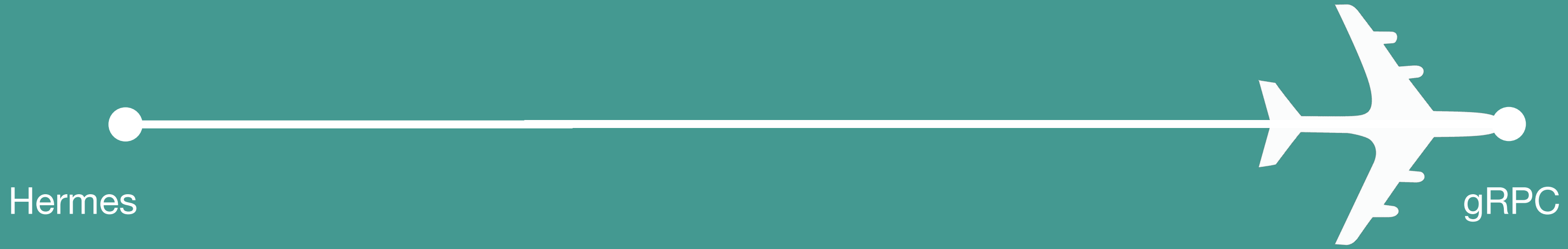
Remaining Distance to Destination



Our Journey



Our Journey



“Developers **don’t care** about new RPC technologies”

-- someone at KubeCon 2018

Developers don't **have** care about new RPC technologies

Thank You