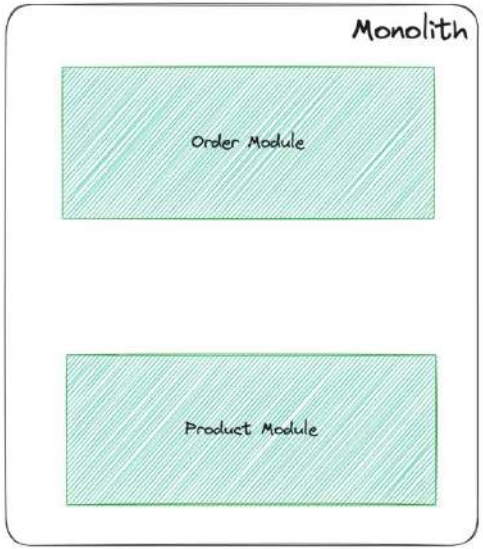
Welcome to

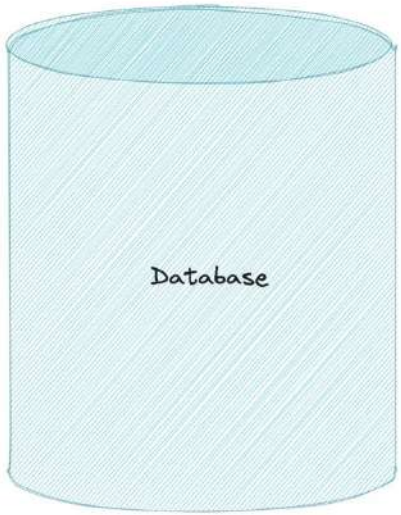
Stay updated!

Building gRPC Microservices Effectively with Go

Chris Shepherd - Senior Systems Engineer @ Cloudflare

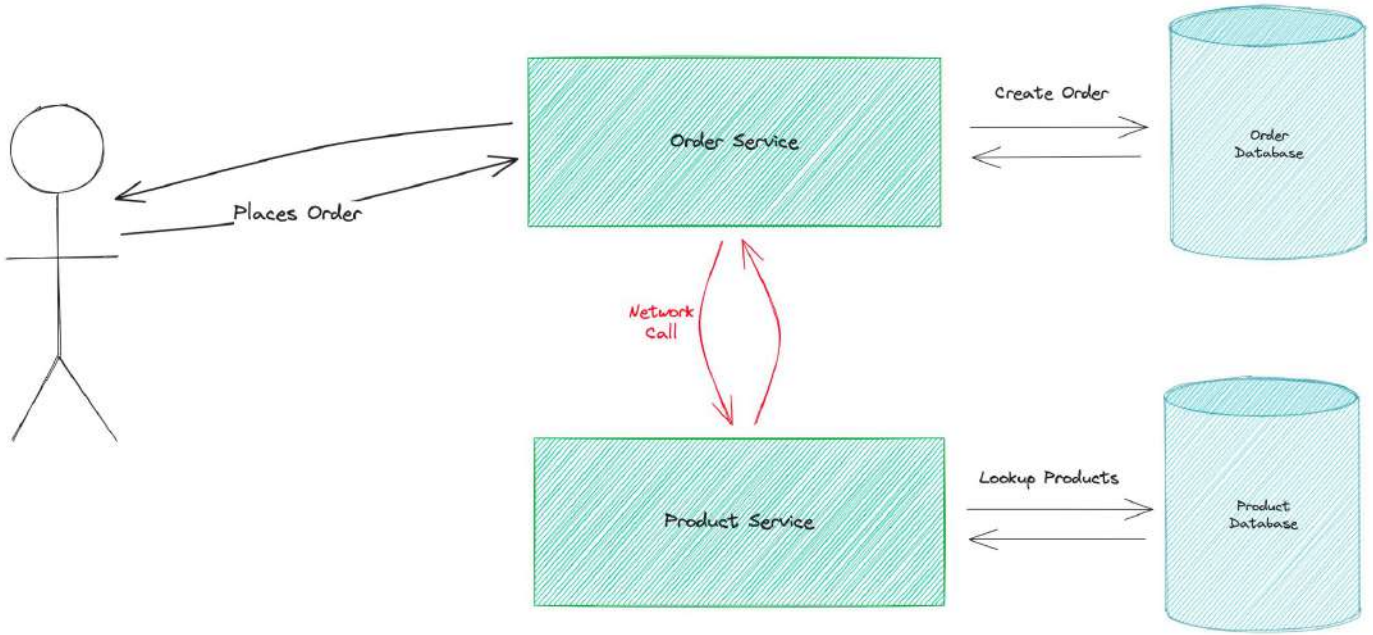
Communication in a Monolith





Data Boundaries in Microservices

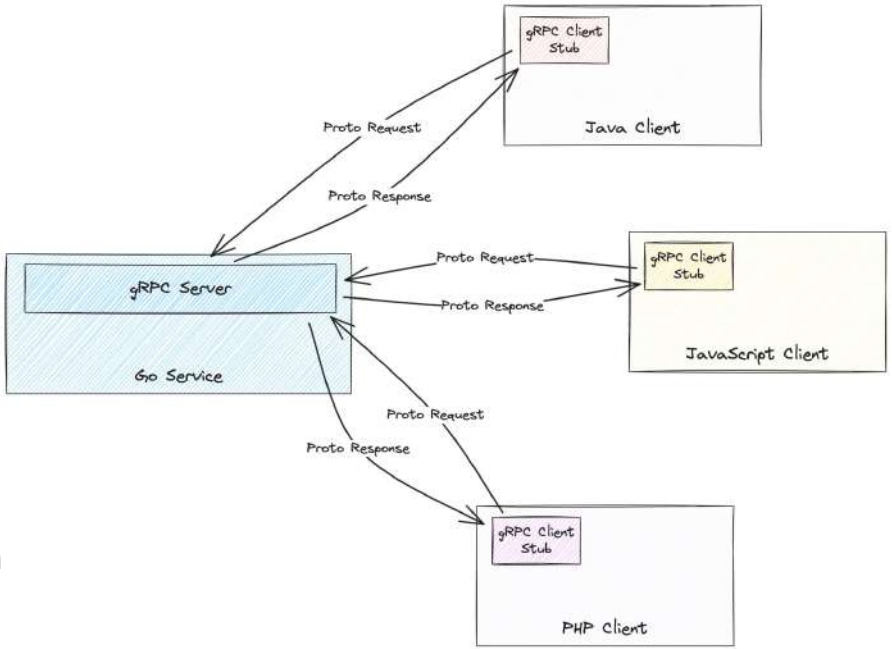
Communication Between Microservices



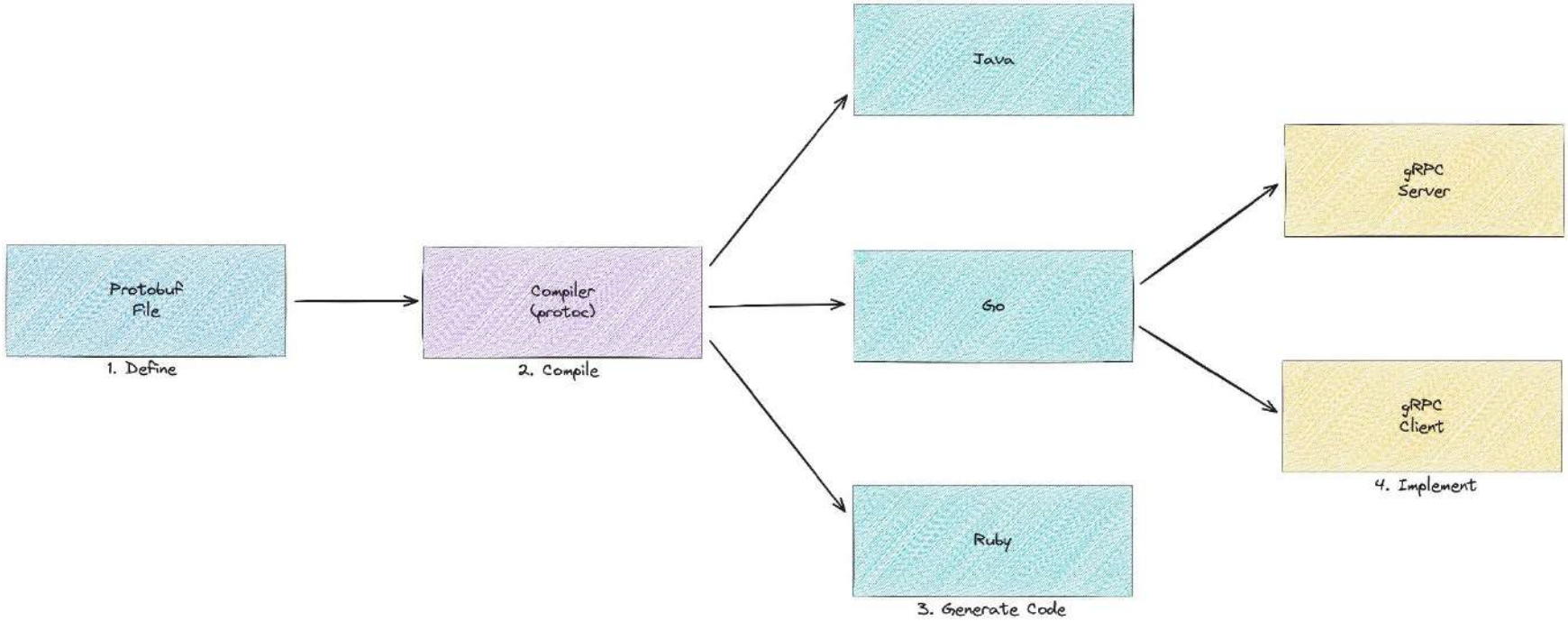
What is gRPC?

gRPC is a modern open source high performance Remote Procedure Call (RPC) framework

- It uses HTTP/2 for transport & Protocol Buffers as the interface description language
- Efficiently connects polyglot microservices
- Generates idiomatic client libraries in all the major programming languages
- Auth, tracing, load balancing and health checking provided out of the box



gRPC Workflow



Defining a gRPC Service

```
syntax = "proto3";

package hello;

service HelloService {
 rpc SayHello(SayHelloRequest) returns (SayHelloResponse);
}

message SayHelloRequest {
 string name = 1;
}

message SayHelloResponse {
 string message = 1;
}
```

Generated Code

```
type SayHelloRequest struct {
   ...
   Name string `protobuf:"bytes,1,opt,name=name,proto3" json:"name,omitempty"`
}
type SayHelloResponse struct {
   ...
   Message string `protobuf:"bytes,1,opt,name=message,proto3" json:"message,omitempty"`
}

...

type HelloServiceServer interface {
   SayHello(context.Context, *SayHelloRequest) (*SayHelloResponse, error)
}

...


type helloServiceClient struct {
   cc grpc.ClientConnInterface
}

func NewHelloServiceClient(cc grpc.ClientConnInterface) HelloServiceClient {
   return &helloServiceClient{cc}
}

func (c *helloServiceClient) SayHello(ctx context.Context, in *SayHelloRequest, opts ...grpc.CallOption) (*SayHelloResponse, error) {
   out := new(SayHelloResponse)
   err := c.cc.Invoke(ctx, "/hello.HelloService/SayHello", in, out, opts...)
   if err != nil {
      return nil, err
   }
   return out, nil
}
```

Why use gRPC?

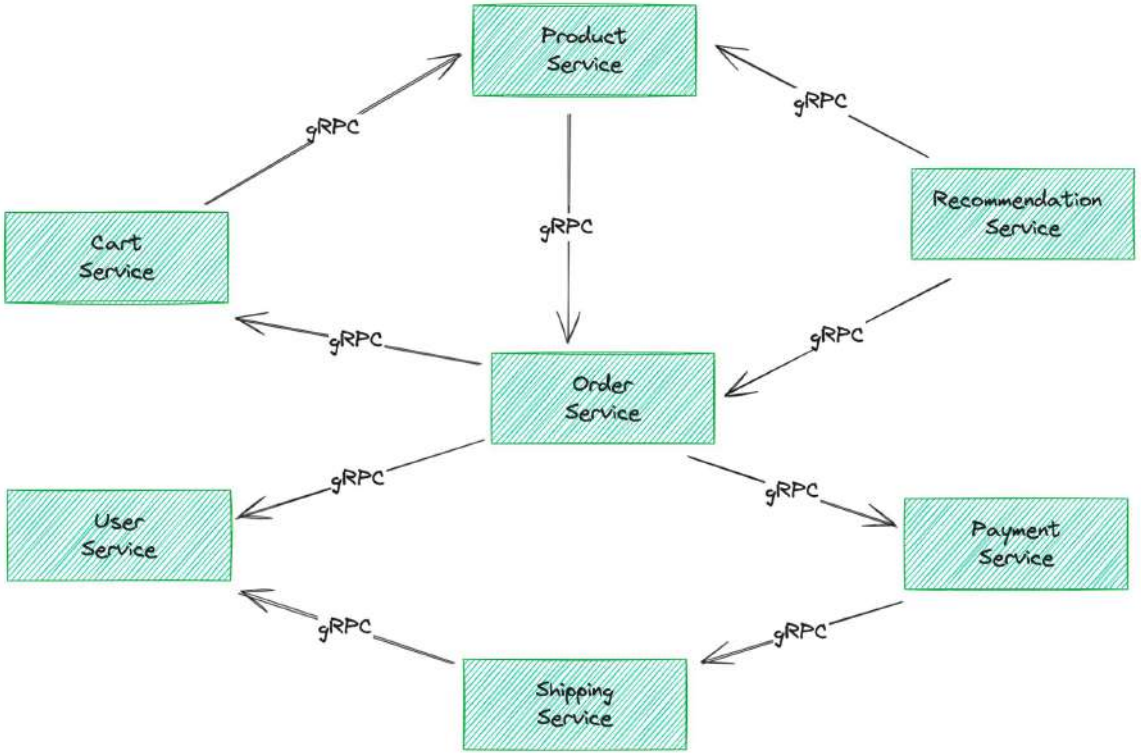
Why use gRPC?

Strong Message Structure

- Contracts are defined upfront.
- Helps ensure compatibility between versions.
- Promotes standardisation.
- Built-in documentation.

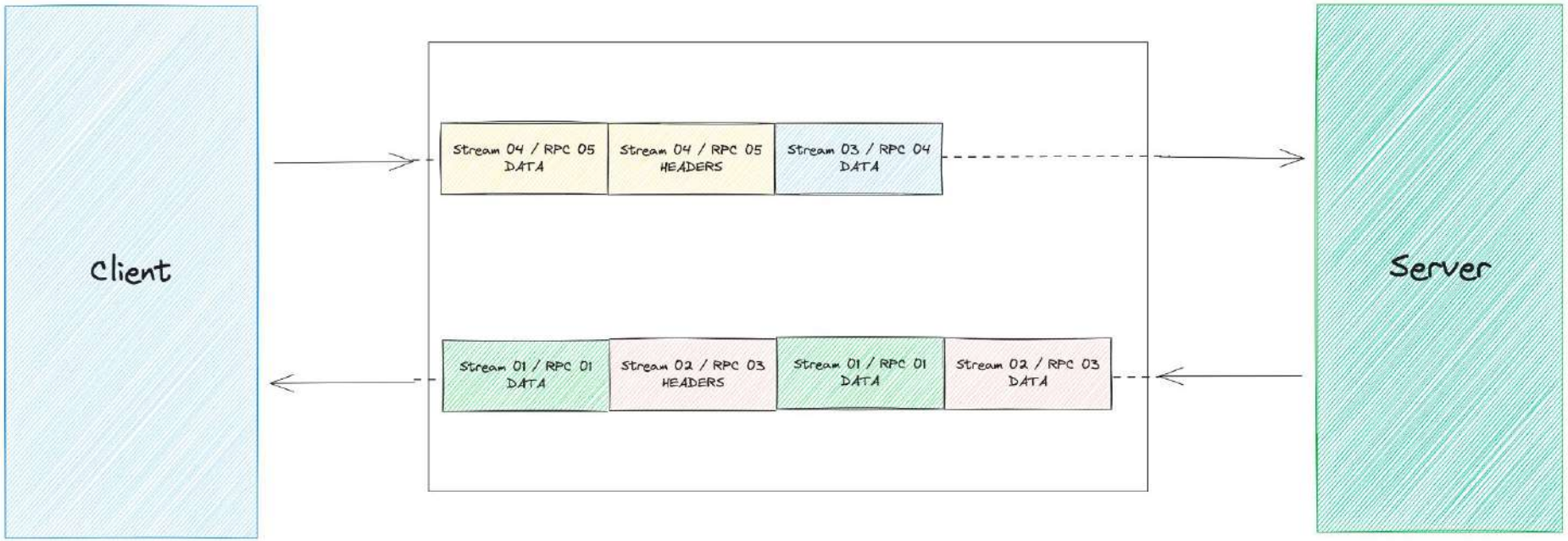
Why use gRPC?

Strong Message Structure



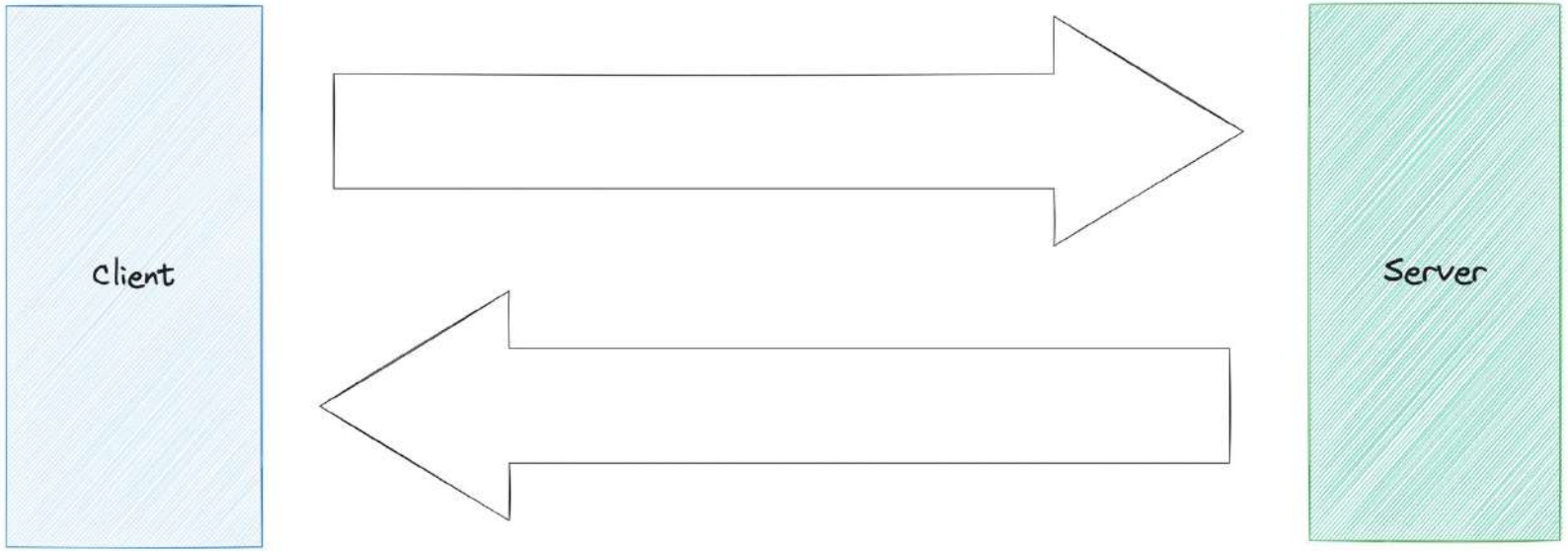
Why use gRPC?

Built on HTTP/2



Why use gRPC?

Built on HTTP/2



Why use gRPC?

Supports Different Languages

```php
require dirname(__FILE__).'/vendor/autoload.php';

function greet($hostname, $name)
{
    $client = new Helloworld\GreeterClient($hostname, [
        'credentials' => Grpc\ChannelCredentials::createInsecure(),
    ]);
    $request = new Helloworld\HelloRequest();
    $request->setName($name);
    list($response, $status) = $client->SayHello($request)->wait();
    if ($status->code !== Grpc\STATUS_OK) {
        echo "ERROR: " . $status->code . ", " . $status->details . PHP_EOL;
        exit(1);
    }
    echo $response->getMessage() . PHP_EOL;
}

$name = !empty($argv[1]) ? $argv[1] : 'world';
$hostname = !empty($argv[2]) ? $argv[2] : 'localhost:50051';
greet($hostname, $name);
```

Client

SayHello(HelloRequest)

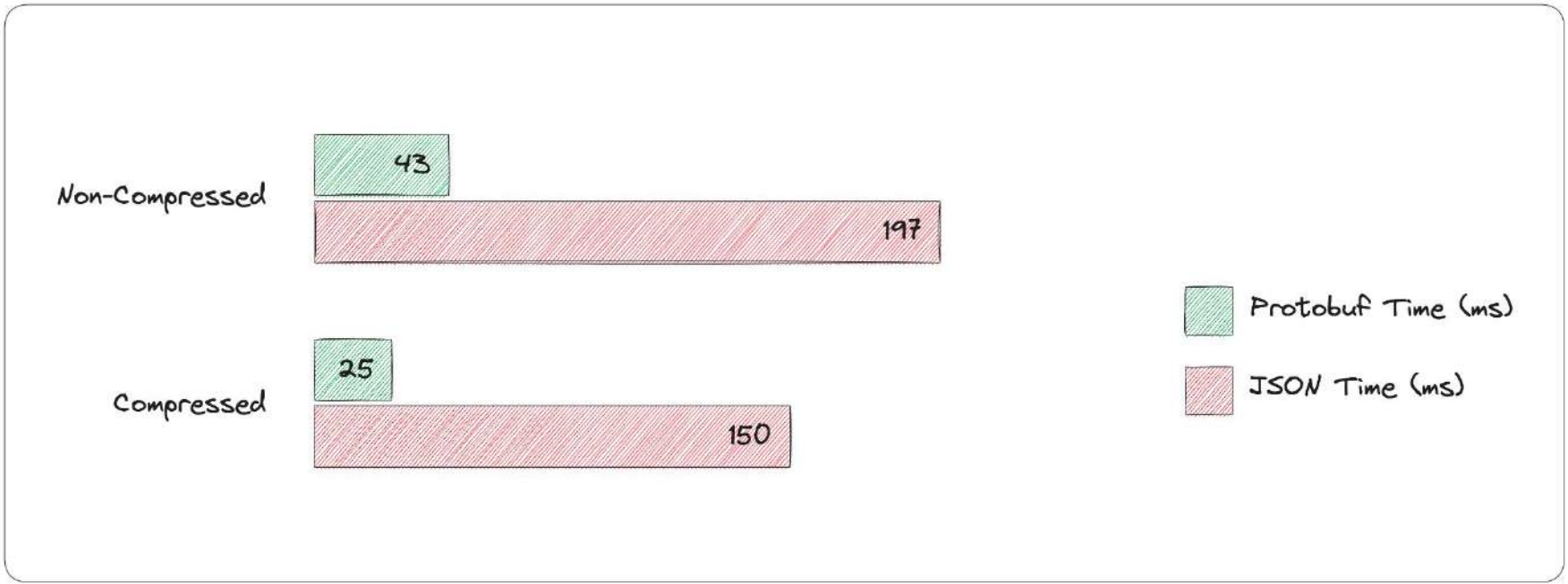
HelloReply{}

```go
package main

import (
        "context"
        "flag"
        "fmt"
        "log"
        "net"

        "google.golang.org/grpc"
        pb "google.golang.org/grpc/examples/helloworld/helloworld"
)

var (
        port = flag.Int("port", 50051, "The server port")
)

// server is used to implement helloworld.GreeterServer.
type server struct {
        pb.UnimplementedGreeterServer
}

// SayHello implements helloworld.GreeterServer
func (s *server) SayHello(ctx context.Context, in *pb.HelloRequest) (*pb.HelloReply, error) {
        log.Printf("Received: %v", in.GetName())
        return &pb.HelloReply{Message: "Hello " + in.GetName()}, nil
}

func main() {
        flag.Parse()
        lis, err := net.Listen("tcp", fmt.Sprintf(":%d", *port))
        if err != nil {
                log.Fatalf("failed to listen: %v", err)
        }
        s := grpc.NewServer()
        pb.RegisterGreeterServer(s, &server{})
        log.Printf("server listening at %v", lis.Addr())
        if err := s.Serve(lis); err != nil {
                log.Fatalf("failed to serve: %v", err)
        }
}
```

Server

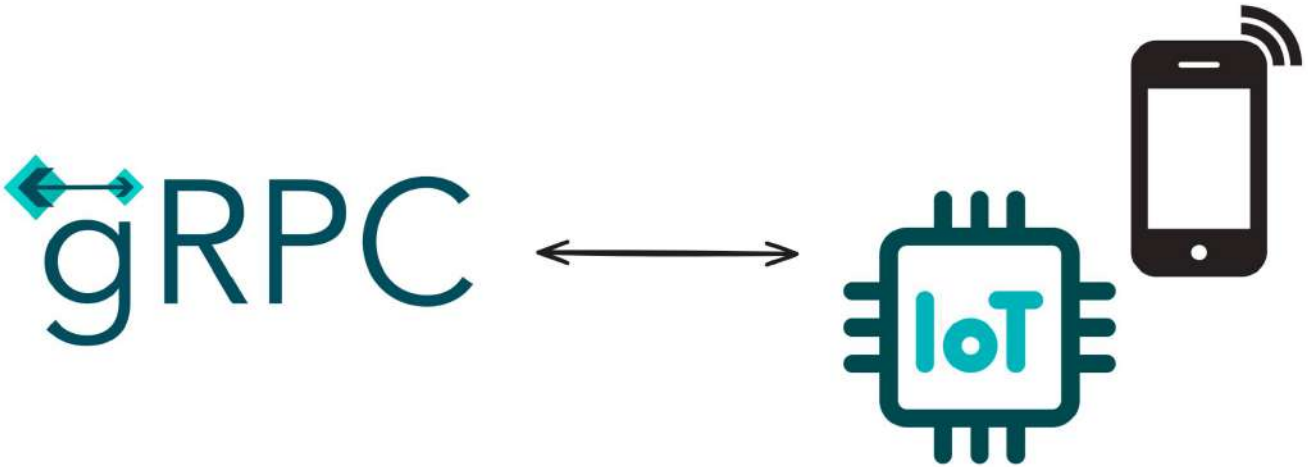
Why use gRPC?

Faster Message Transmission



Why use gRPC?

Faster Message Transmission



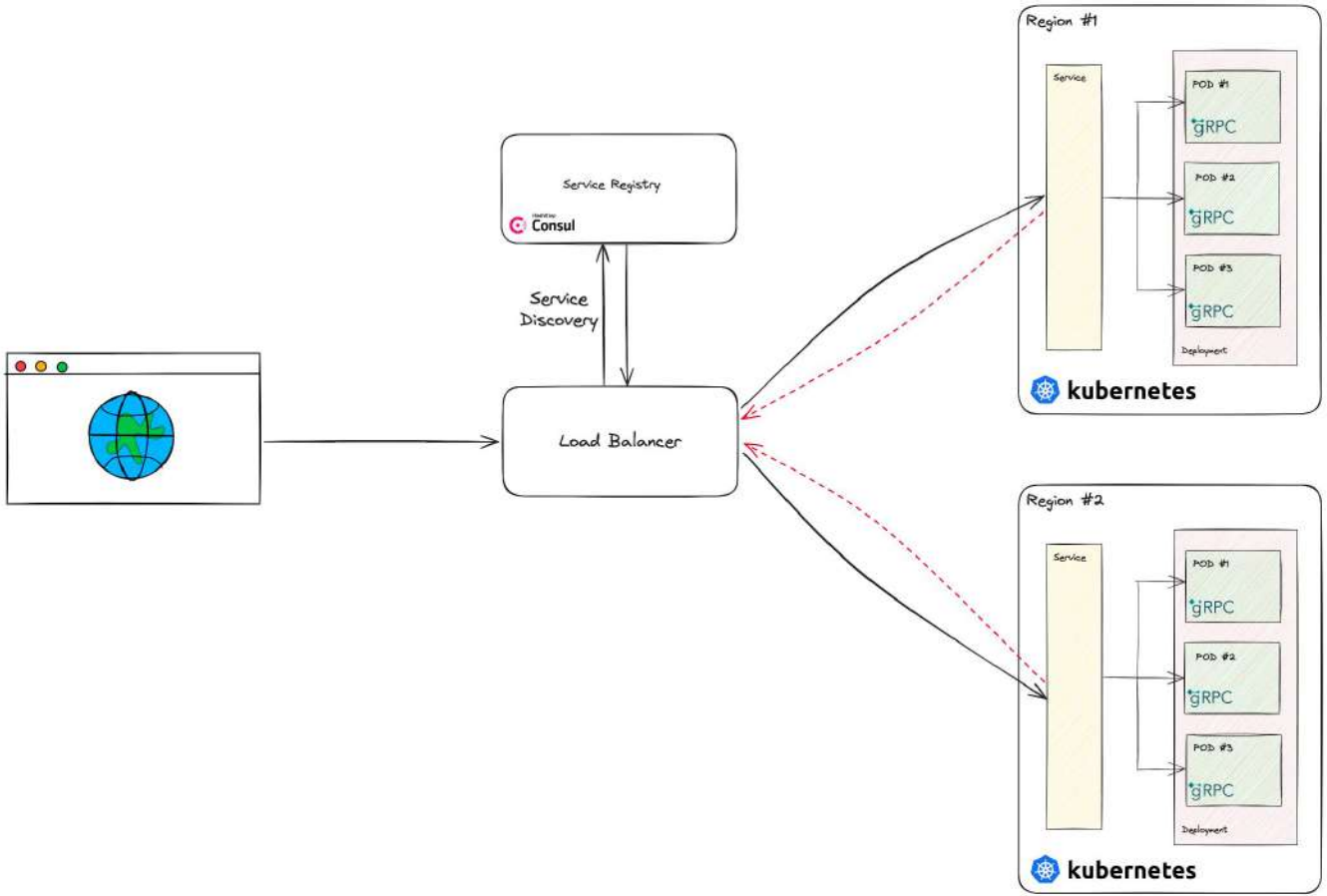
Let’s See Some Code!

Using gRPC in the Real World

gRPC @ Cloudflare

- Event driven architecture with over **1000** microservices.
- Synchronous APIs communicating via **gRPC**.
- **Highly available** services running in multiple data centers around the world.
- Over **70** engineering teams making multiple deployments per day.
- gRPC clients and services communicating in **7** different programming languages.

gRPC @ Cloudflare



Maintaining Protocol Buffers at Scale

Standardised Protobuf Build Pipeline

- Centralised repository for all gRPC protobuf definitions.
- Standardised build pipeline for protobuf.
- Integrates with **Buf** to handle linting, building & breaking change detection.
- Acts as documentation for the provided gRPC APIs.

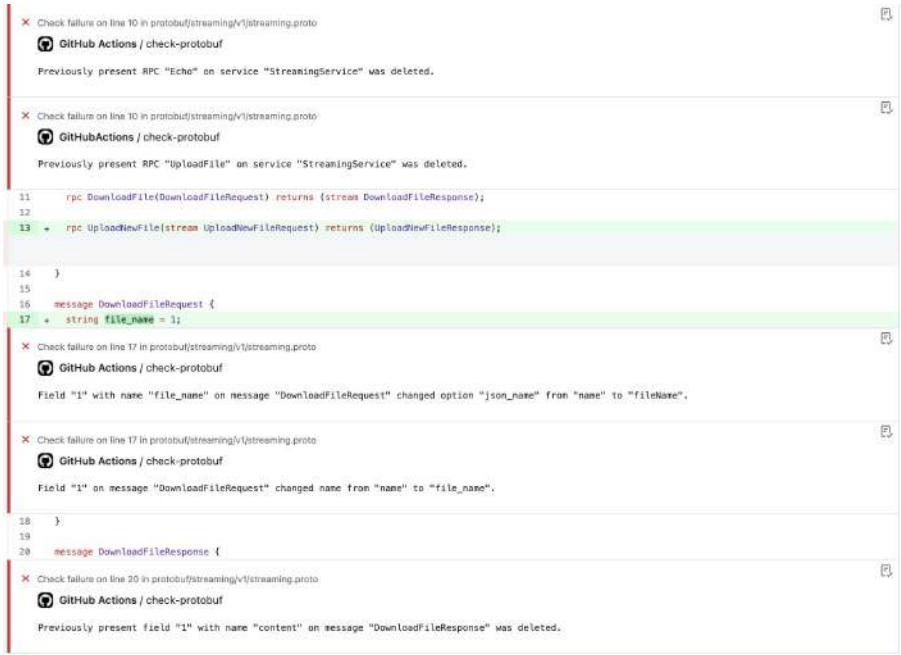
Simple, Repeatable Code Generation

```
$ protoc \
 -I protobuf \
 -I vendor/protoc-gen-validate \
 --go_out=build/go \
 --go_opt=paths=source_relative \
 --go-grpc_out=build/go \
 --go-grpc_opt=paths=source_relative \
 --php_out=build/php \
 --grpc_out=build/php \
 --plugin=protoc-gen-grpc=bins/opt/grpc_php_plugin \
 --python_out=build/python \
 --python-grpc_out=build/python \
 ...

 $(find protobuf -name '*.proto')
```

```
$ buf generate protobuf
```

Breaking Changes

- Allows for rapid development without the risk of breaking existing clients/servers.
- Ensures all APIs are forward and backward compatible.
- Examples:
  - Deleted RPCs/fields
  - Renamed RPCs/fields
  - Changes to RPC signatures
  - Changes to data types
  - Incompatible modifications to generated code (e.g. change to package name or class name)
- Can be integrated into a CI pipeline.



The Good 

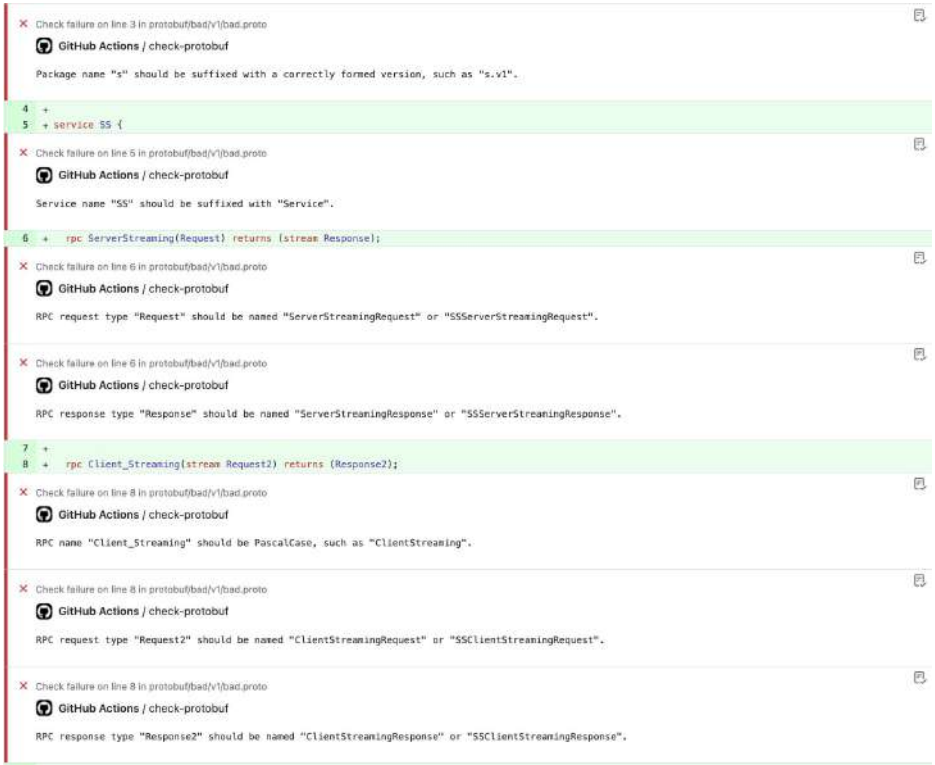
```proto
package streaming.v1;

option go_package = "github.com/cshep4/proto-repository/gen/go/streaming/v1";
option java_multiple_files = true;
option java_package = "com.cshep4.proto.streaming.v1";
option php_namespace = "Streaming\\V1";

service StreamingService {
 rpc ServerStreaming(ServerStreamingRequest) returns (stream ServerStreamingResponse);

 rpc ClientStreaming(stream ClientStreamingRequest) returns (ClientStreamingResponse);

 rpc BiDirectionalStreaming(stream BiDirectionalStreamingRequest) returns (stream BiDirectionalStreamingResponse);
}

message ServerStreamingRequest {
 string message = 1;
}

message ServerStreamingResponse {
 string message = 1;
}

message ClientStreamingRequest {
 string name = 1;
}

message ClientStreamingResponse {
 repeated string received_messages = 1;
}

message BiDirectionalStreamingRequest {
 string name = 1;
}

message BiDirectionalStreamingResponse {
 string message = 1;
}
```

The Bad 😬

```
package s;

service SS {
 rpc ServerStreaming(Request) returns (stream Response);

 rpc Client_Streaming(stream Request2) returns (Response2);

 rpc BiDirectionalStreaming(stream Request3) returns (stream Response3);
}

message Request {
 string m = 1;
}

message Response {
 string m = 1;
}

message Request2 {
 string name = 1;
}

message Response2 {
 repeated string recMess = 1;
}

message Request3 {
 string a = 1;
 string b = 3;
}

message Response3 {
 string message = 3;
}
```

Linting

- Helps maintain the quality of protobuf.
- Catch errors early & make code easier to understand.
- Reduce the amount of manual code review required.
- Lint rules are configurable.



Additional Plugins

Buf Schema Registry

Protovalidate - Easy message validation using Protobuf annotations.

```
message User {
 // User's name, must be at least 1 character long.
 string name = 1 [(buf.validate.field).string.min_len = 1];

 int32 age = 1 [(buf.validate.field).cel = {
   id: "user.age",
   expression: "this < 18 ? 'User must be at least 18 years old': ''"
 }];

 // User's creation date must be in the past.
 Timestamp created_at = 1 [(buf.validate.field).timestamp.lt_now = true];
}
```

Connect - gRPC compatible APIs, with less boilerplate & more idiomatic generated code. Working over HTTP/1.1, HTTP/2, or HTTP/3

Thank you!

Chris Shepherd - Senior Systems Engineer

Check out the code examples used in this talk -

chris-shepherd1993

@cshep_4

Thank you!

bit summit

Stay updated!