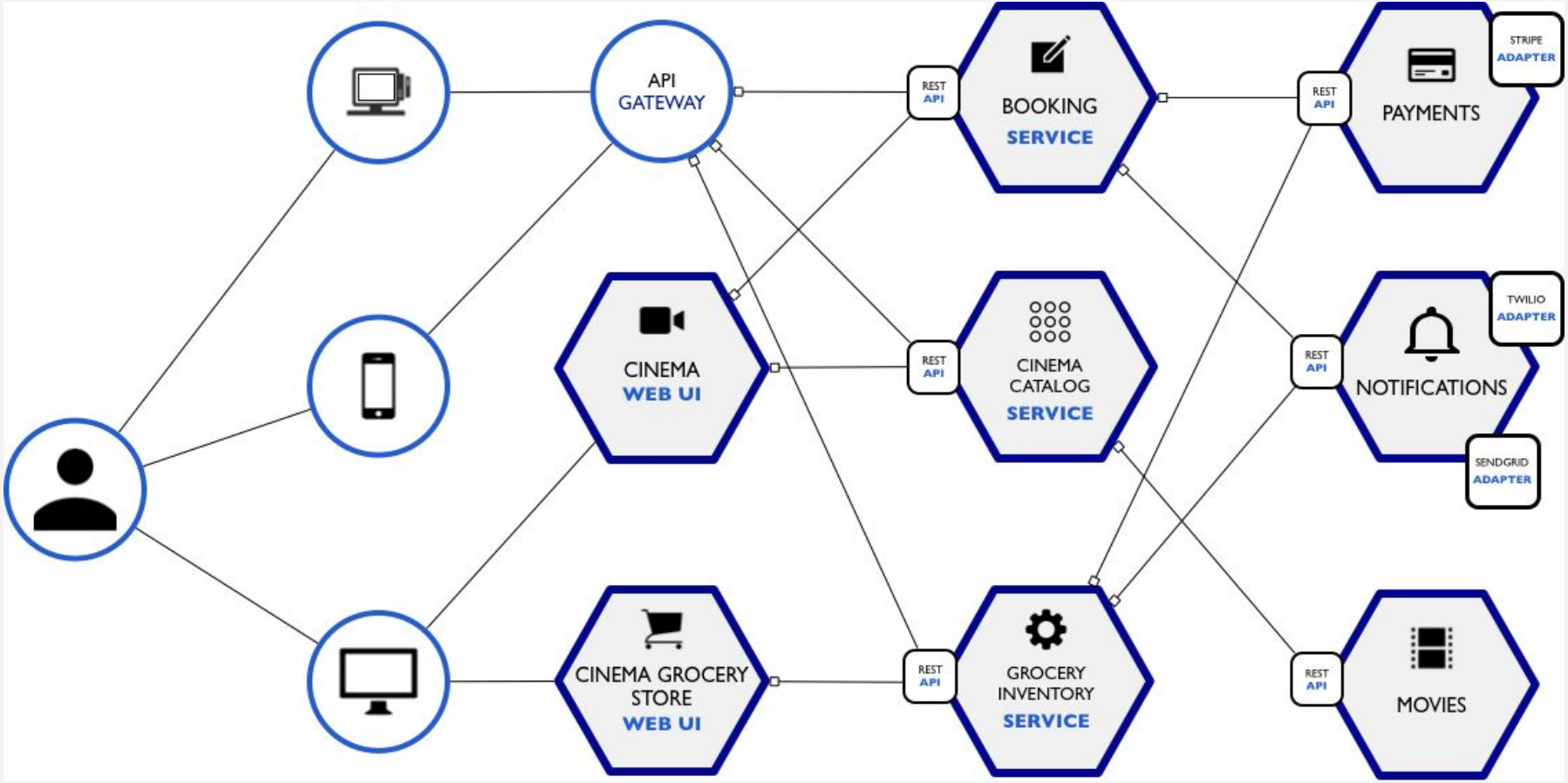
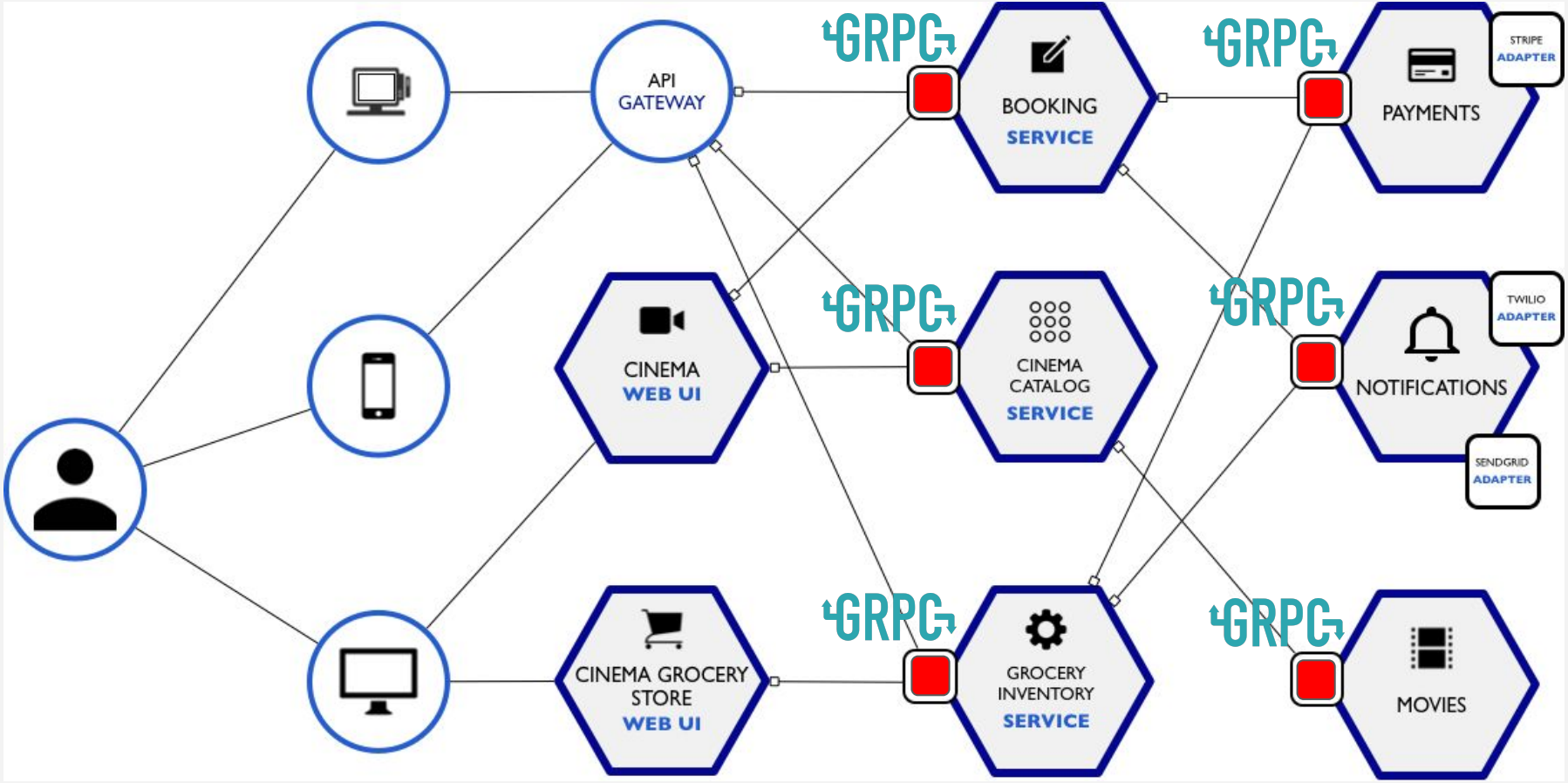
gRPC - Go Micro

Imre Nagi


API
GATEWAY
REST
API
BOOKING
SERVICE
REST
API
PAYMENTS
STRIPE
ADAPTER
CINEMA
WEB UI
REST
API
CINEMA
CATALOG
SERVICE
REST
API
NOTIFICATIONS
TWILIO
ADAPTER
SENDGRID
ADAPTER
CINEMA GROCERY
STORE
WEB UI
REST
API
GROCERY
INVENTORY
SERVICE
REST
API
MOVIES

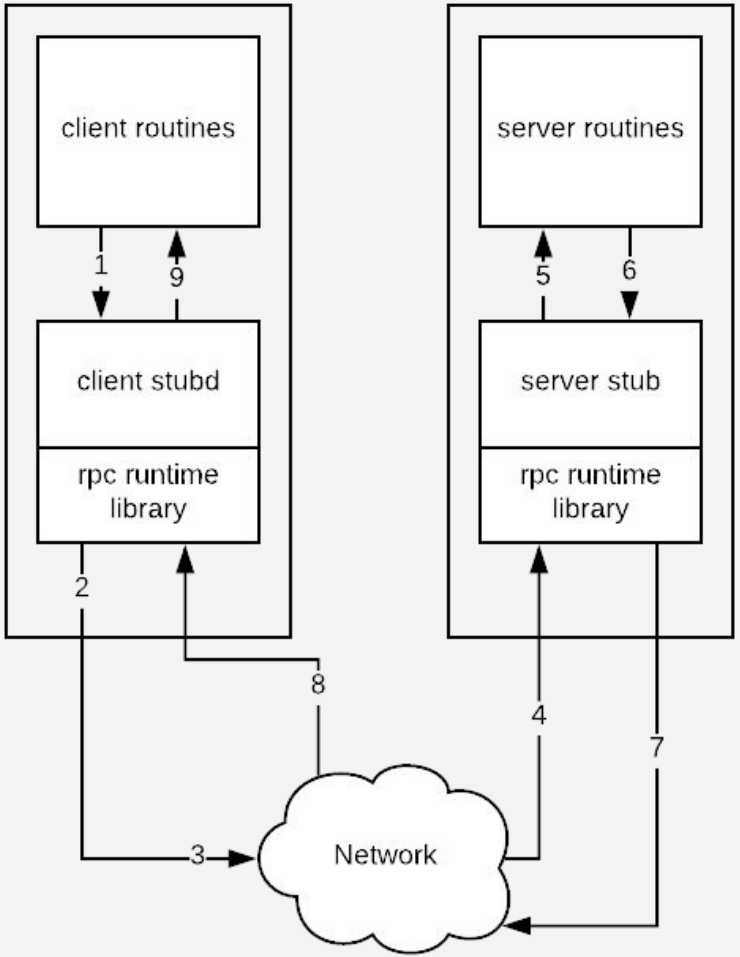

API
GATEWAY
GRPC
BOOKING
SERVICE
GRPC
PAYMENTS
STRIPE
ADAPTER
CINEMA
WEB UI
GRPC
CINEMA
CATALOG
SERVICE
GRPC
NOTIFICATIONS
TWILIO
ADAPTER
SENDGRID
ADAPTER
CINEMA GROCERY
STORE
WEB UI
GRPC
GROCERY
INVENTORY
SERVICE
GRPC
MOVIES

gRPC stands for **g**RPC **R**emote **P**rocedure **C**all

A high performance, general purpose, feature rich RPC framework

HTTP/2.0 and mobile first

Open source!



gRPC Over Rest

- Make streaming highly possible
- Machine Readable (not meant for browser)
- Lesser Boilerplate code (Code generator for 8 languages)
- Supports types and Validations: Unlike json , we can specify field types and add validations for the same in the .proto file.

micro / go-micro

A microservice framework https://micro.mu

go micro rpc distributed-systems microservices go-micro microservice cloud-native

787 commits | **3** branches | **16** releases | **30** contributors | Apache-2.0

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

asim Merge pull request #309 from fireyang/master ... Latest commit 4cee1f1 on Sep 20

broker	set broker address on Init	2 months ago
client	fix rpc client call WARNING: DATA RACE	a month ago
cmd	fix bug with go-config panic	a month ago
codec	strip doc.go files	2 years ago
errors	errors: Added 405 Method Not Allowed helper function	a month ago
metadata	switch to stdlib context	8 months ago
registry	Add Init to all things, use init in cmd package to initialise	3 months ago
selector	switch to stdlib context	8 months ago
server	Change waitgroup processing	5 months ago
transport	Add Init to all things, use init in cmd package to initialise	3 months ago
.travis.yml	bump travis	2 months ago
LICENSE	Add apache license	4 years ago
README.md	update readme	6 months ago
function.go	Move publication to message	6 months ago
function_test.go	restructure test	5 months ago
go-micro.go	switch to stdlib context	8 months ago
go-micro.png	update image	3 years ago
options.go	Fix broker registry issue	6 months ago

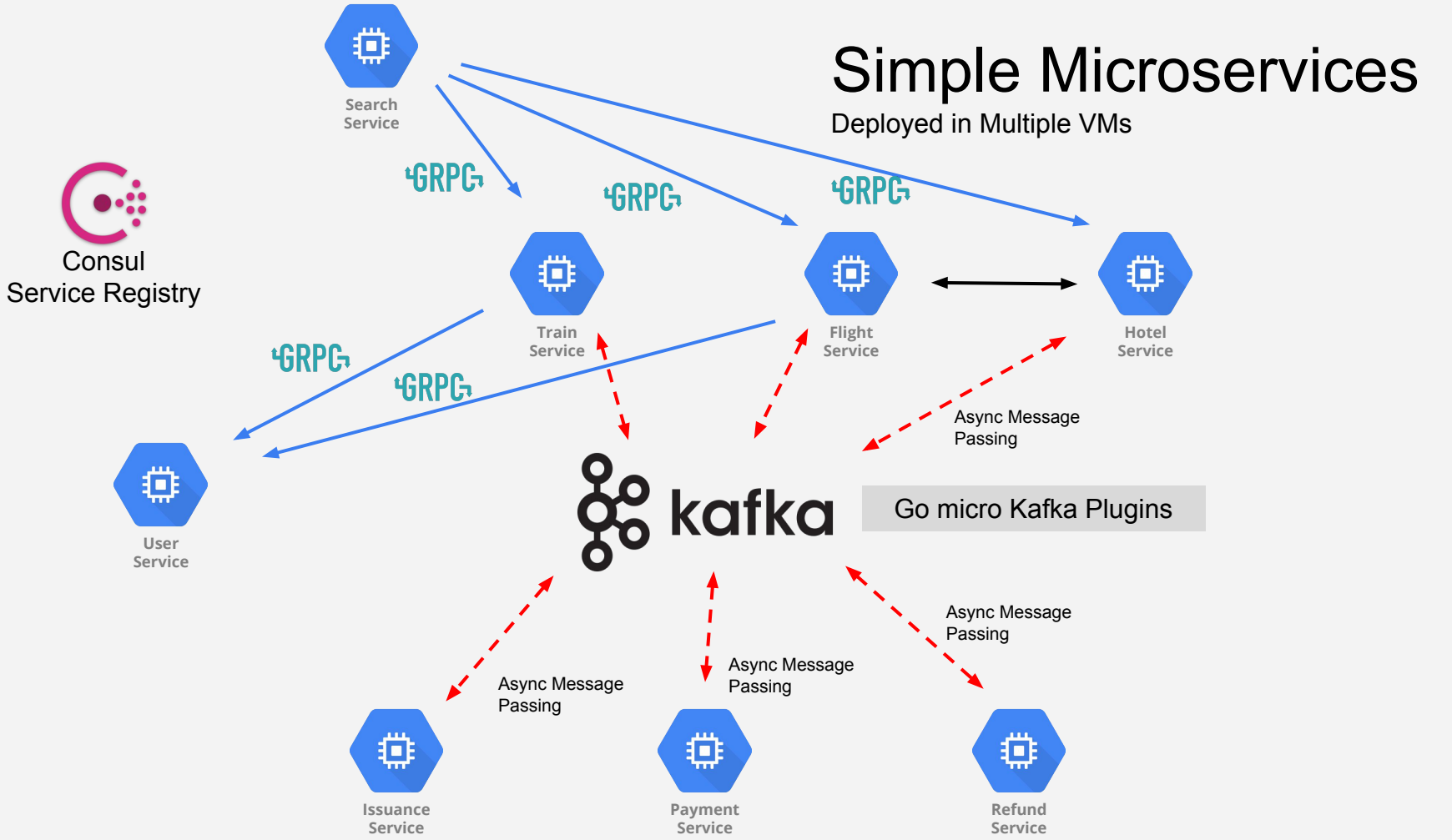
Go Micro

Go Micro abstracts away the details of distributed systems. Here are the main features.

- **Service Discovery - Automatic service registration and name resolution**
- **Load Balancing - Client side load balancing built on discovery**
- Message Encoding - Dynamic encoding based on content-type with protobuf and json support
- Sync Streaming - RPC based communication with support for bidirectional streaming
- **Async Messaging - Native PubSub messaging built in for event driven architectures**

Simple Microservices

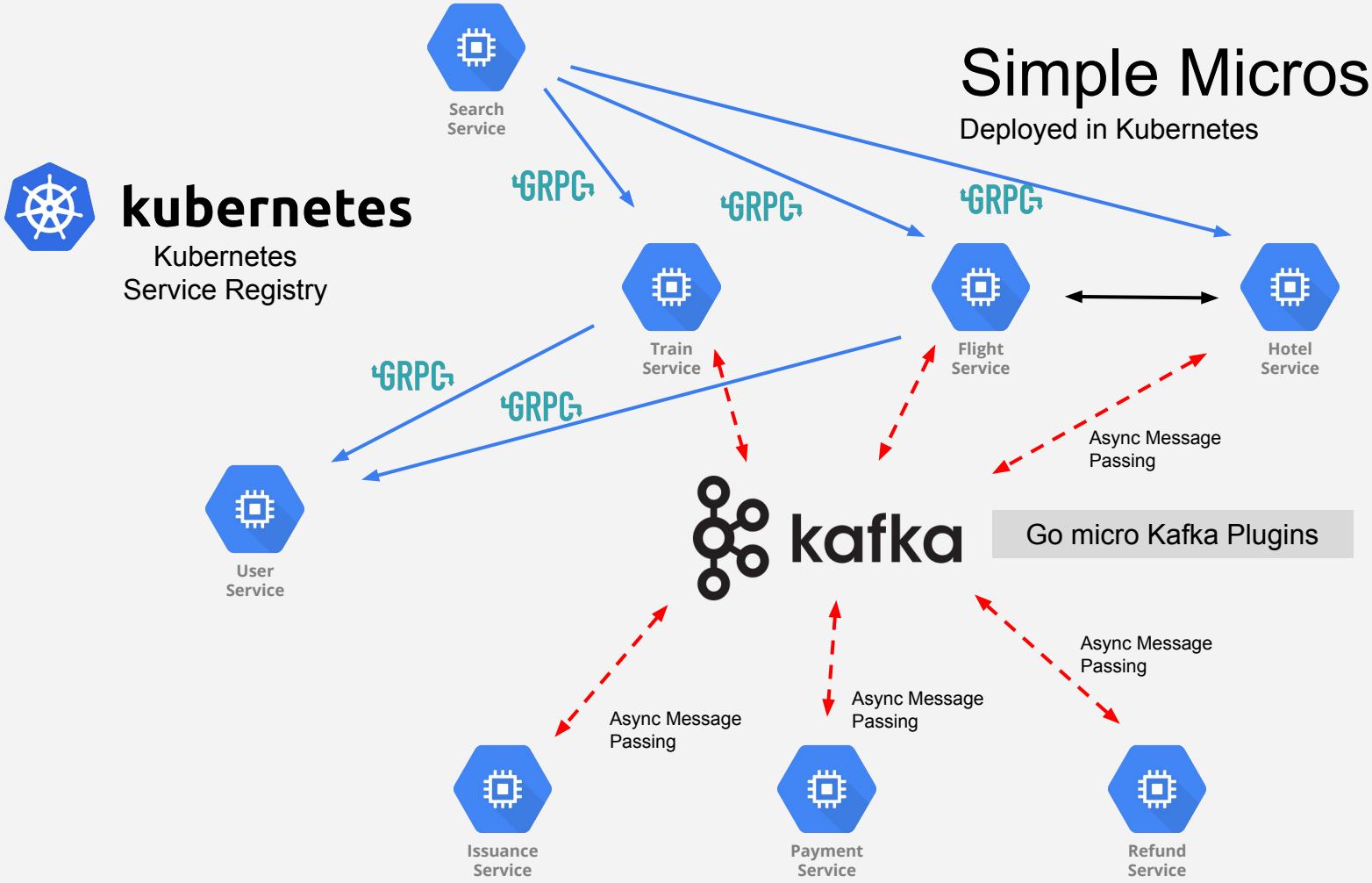
Deployed in Multiple VMs



*VM icon taken from GCP Icon

Simple Microservices

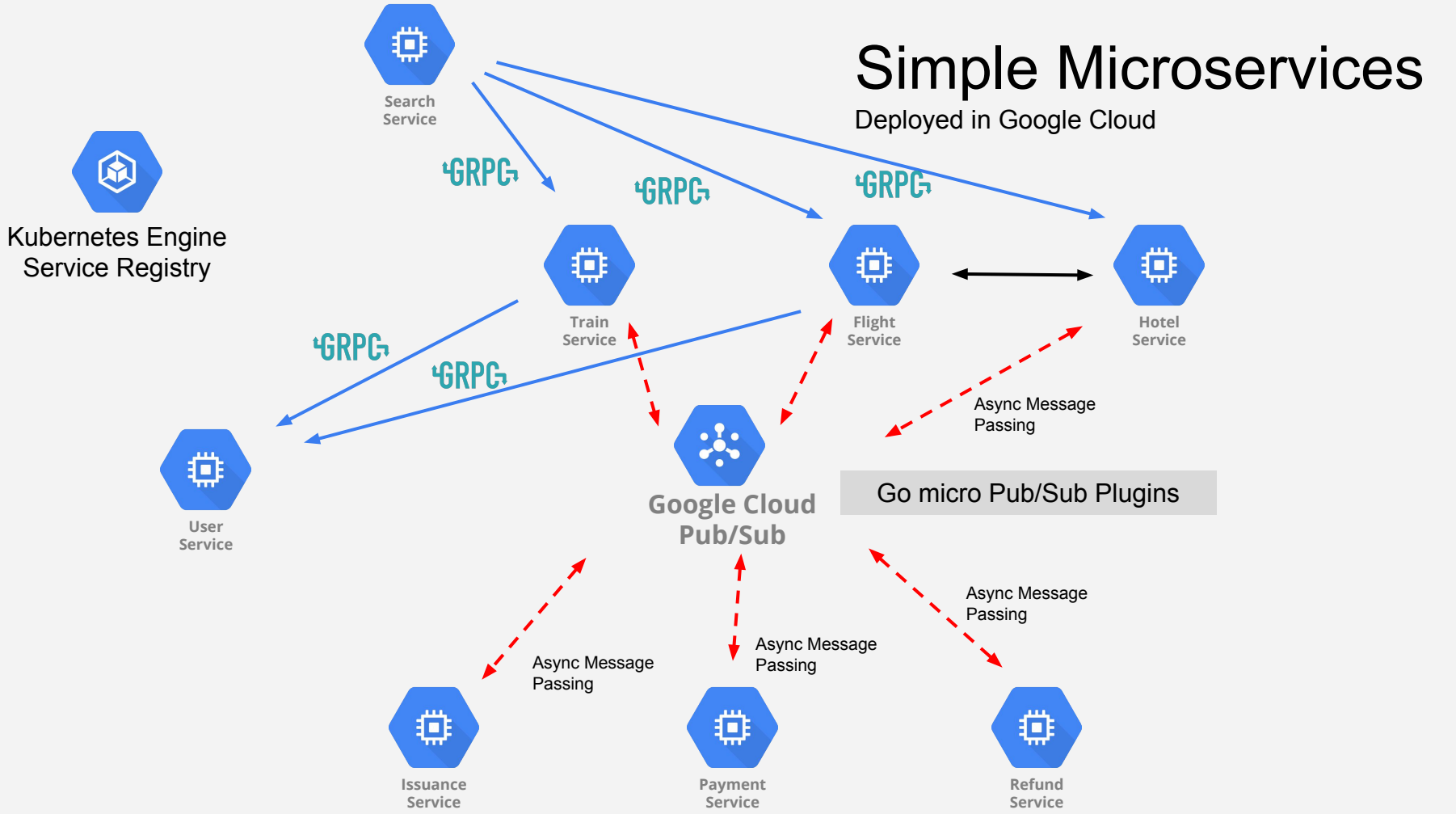
Deployed in Kubernetes



*VM icon taken from GCP Icon

Simple Microservices

Deployed in Google Cloud



*VM icon taken from GCP Icon

```
syntax = "proto3";

service Greeter {
        rpc Hello(HelloRequest) returns (HelloResponse) {}
}

message HelloRequest {
        string name = 1;
}

message HelloResponse {
        string greeting = 2;
}
```

```go
package main

import (
        "context"
        "fmt"

        micro "github.com/micro/go-micro"
        proto "github.com/micro/examples/service/proto"
)

type Greeter struct{}

func (g *Greeter) Hello(ctx context.Context, req *proto.HelloRequest, rsp *proto.HelloResponse) error {
        rsp.Greeting = "Hello " + req.Name
        return nil
}
```

```go
func main() {
	// Create a new service. Optionally include some options here.
	service := micro.NewService(
		micro.Name("greeter"),
	)

	// Init will parse the command line flags.
	service.Init()

	// Register handler
	proto.RegisterGreeterHandler(service.Server(), new(Greeter))

	// Run the server
	if err := service.Run(); err != nil {
		fmt.Println(err)
	}
}
```

```go
package main

import (
        "context"
        "fmt"

        micro "github.com/micro/go-micro"
        proto "github.com/micro/examples/service/proto"
)


func main() {
        // Create a new service. Optionally include some options here.
        service := micro.NewService(micro.Name("greeter.client"))
        service.Init()

        // Create new greeter client
        greeter := proto.NewGreeterService("greeter", service.Client())

        // Call the greeter
        rsp, err := greeter.Hello(context.TODO(), &proto.HelloRequest{Name: "John"})
        if err != nil {
                fmt.Println(err)
        }

        // Print response
        fmt.Println(rsp.Greeting)
}
```