Uber x Security

Tyler Julian, Security Engineer @Uber
Daniel Feldman, Software Engineer @Scytale

May 23, 2019

Uber

Agenda

01 Overview
02 Identity at Uber
03 SPIFFE
04 Case Study
05 Q&A

Identity at Uber

Tyler Julian

About Me

- Authentication
- Distributed Systems
- @Uber
  - Identity & Access Management
  - Trust & Safety
- @21 (acq. by Coinbase)
  - Cryptocurrency Protocol Implementation

Scale

xK+

Unique services.

xxxK

Running containers to support stateless services.

xxM

Batch workloads daily.

Infra

- Deployments in both cloud and on-prem data centers
- RPC with gRPC/HTTP and in-house protocols
- Routing/discovery built in-house
- Orchestration using Mesos, Hadoop, and in-house tools
- Services written in Go, Java, Python, Node.js, and more

Identity Requirements

- Compliance
  - General Data Protection Regulation (GDPR)
  - Sarbanes-Oxley (SOX)

- Trust and Security
  - Reduce assumptions on system behavior (zero trust)
  - Reduce risk of unauthorized access
  - Reduce risk of bad configuration

- Developer Experience
  - Easy to implement and use
  - Integrated with infrastructure

Identity Scope

```
function check(n)
  {  // check if the number n is a prime
   var factor; // if the checked number is not a prime, this
   var c;
   factor = 0;
   // try to divide the checked number by all numbers till it
   for (c=2 ; (c <= Math.sqrt(n)) ; c++)
     {
      if (n%c == 0)    // is n divisible by c ?
        {factor = c;  break}
     }
   return (factor);
  }  // end of check function

function communicate()
 {  // communicate with the user
   var i;       // i is the checked number
   var factor; // if the checked number is not a prime, this
   i = document.primetest.number.value;      // get the check
   // is it a valid input?
   if ((isNaN(i)) || (i <= 0) || (Math.floor(i) != i))
     {alert ("The checked object should be a whole positive n
   else
     {
      factor = check (i);
      if (factor == 0)
        {alert (i + " is a prime")} ;
      else
        {alert (i + " is not a prime, " + i + "=" + factor +
     }
  }    // end of communicate function
```

Humans

Riders, drivers, couriers, customer support representatives, managers, engineers, etc.

Machines

Addressable hosts that reside within “Uber” infrastructure.

Workloads

A process that runs application logic for some business purpose.

Workload Identity

- Goal:
  - Uniquely identify a particular program or application

- Control access to:
  - Database credentials
  - Third party API keys
  - Other internal services

- Protect data:
  - Encryption-in-transit (confidentiality and integrity)
  - Controlled access (authenticity and authorization)

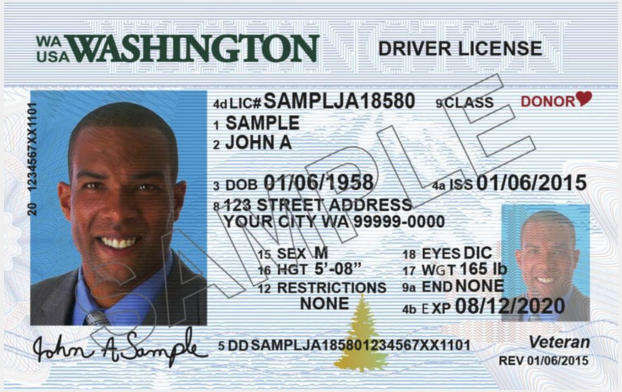
SPIFFE

Daniel Feldman, Scytale

About Me

- Worked on distributed filesystems @ Veritas for ~5 years
- Then 2 years working on service auth for all services in Veritas backup software
  - (Used at thousands of huge companies)
- Last year @ Scytale doing service auth in general

What makes a good ID?



Unique

Stable

Trusted

Verifiable

SPIFFE Verifiable ID Document

spiffe://acme.com/billing/payments

A SPIFFE ID

X.509-SVID describes exactly how to encode a SPIFFE ID in an X.509 certificate

JWT-SVID describes exactly how to encode a SPIFFE ID in an JWT bearer token

SPIFFE adoption

Launched in December 2017

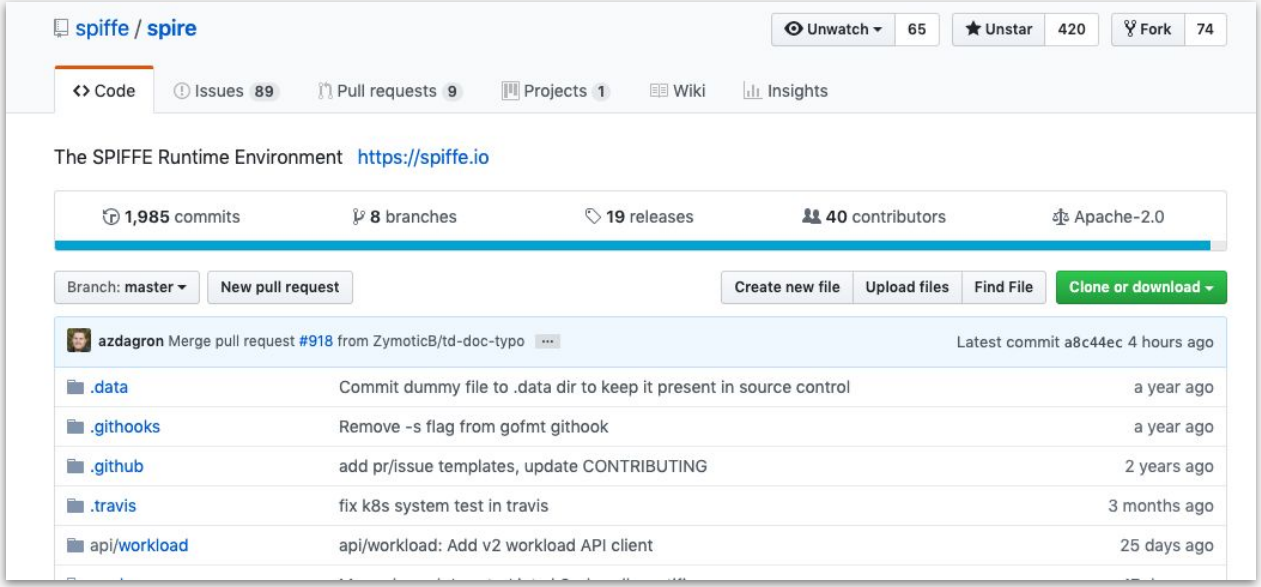
Accepted into the CNCF in March 2018

Integrated into multiple cloud-native open-source projects

Deployed by a growing number of enterprises

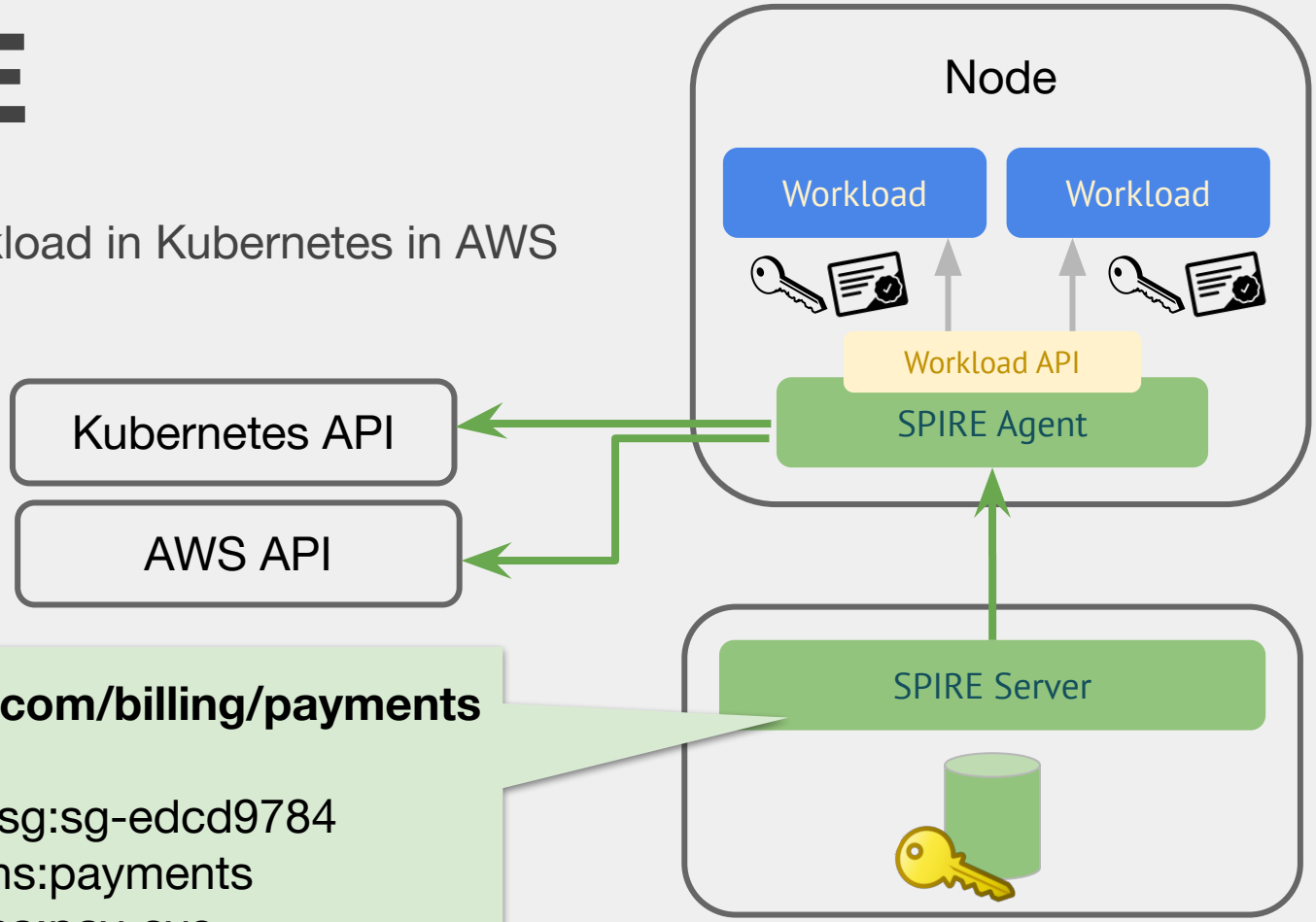
SPIRE

- **SPIRE is the reference implementation** of SPIFFE
- **Provides a simple API** for a workload to get its own SVID
- **Verifies the identity of workloads** using plugins that talk to infrastructure



SPIRE

Example: Workload in Kubernetes in AWS



spiffe://acme.com/billing/payments

selector: aws:sg:sg-edcd9784
selector: k8s:ns:payments
selector: k8s:sa:pay-svc

SPIRE Goals

SCYTALE

- **Fully automated and policy driven**. Existing identity (particularly PKI) infrastructure requires human trust. SPIRE is fully automated and minimizes manual key distribution.

- **Minimal Knowledge**. A compromised machine should only expose secrets for workloads running on that machine.

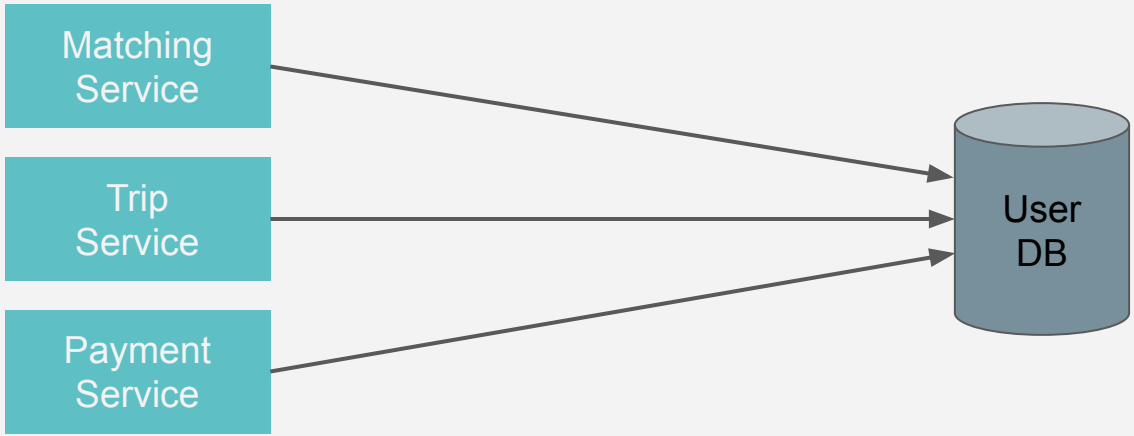
- **Reliable**. The single points of failure in the system should be minimized and the system should degrade gracefully when any SPOF is down.

- **Scoped trust roots**. There should be no hardcoded, global trust roots (unlike web PKI).

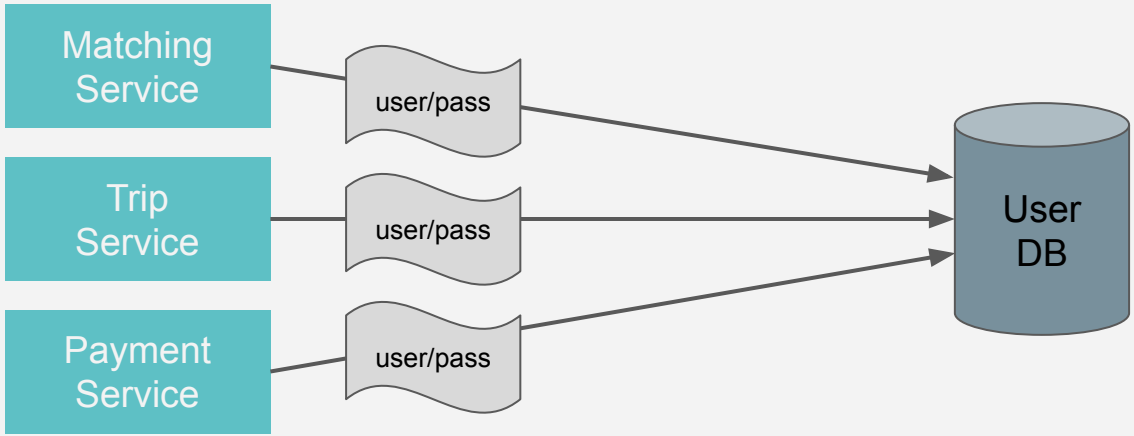
Case Study

Authentication in a Microservice Architecture

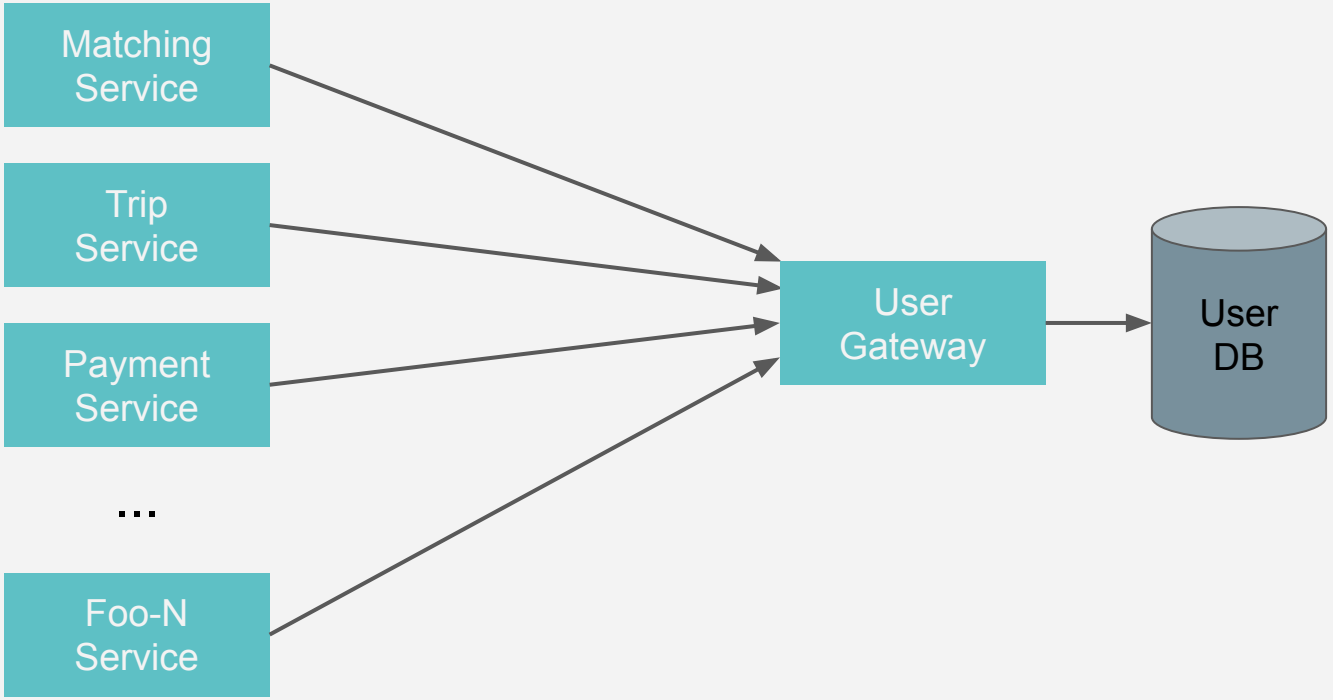
Early: Service to DB (Direct Data Access)



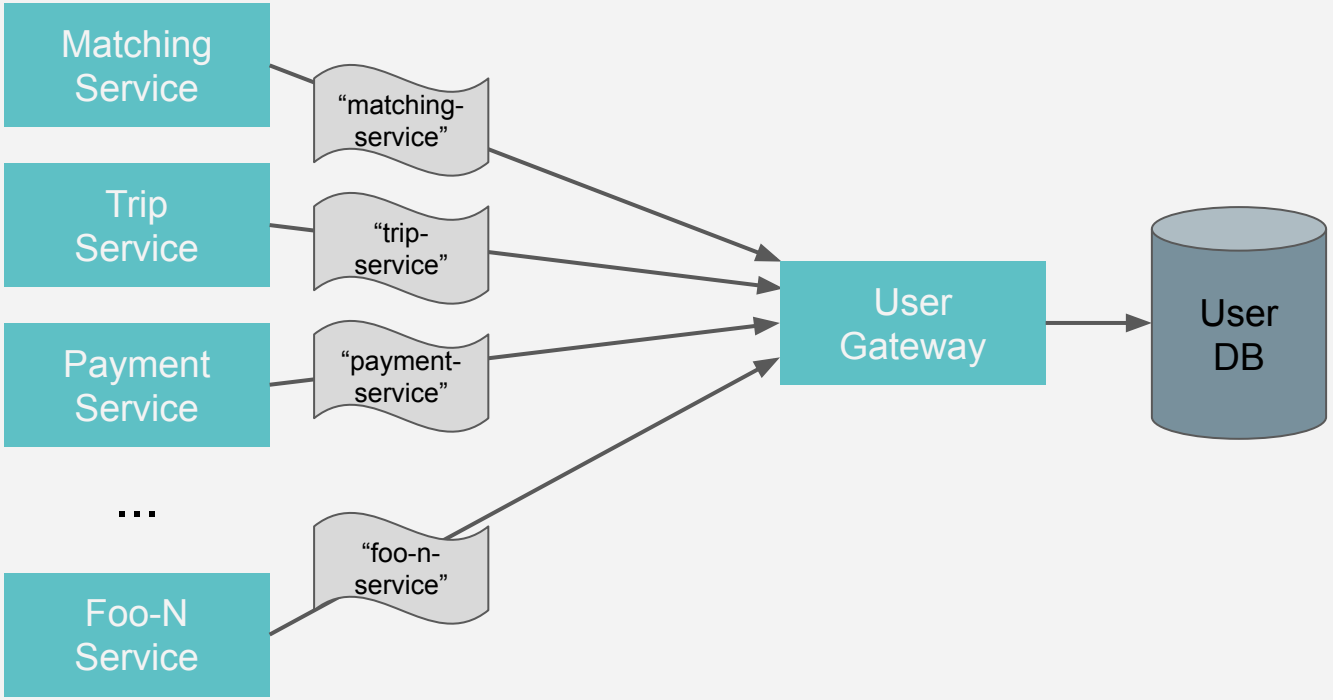
Early: Service to DB (Direct Data Access)



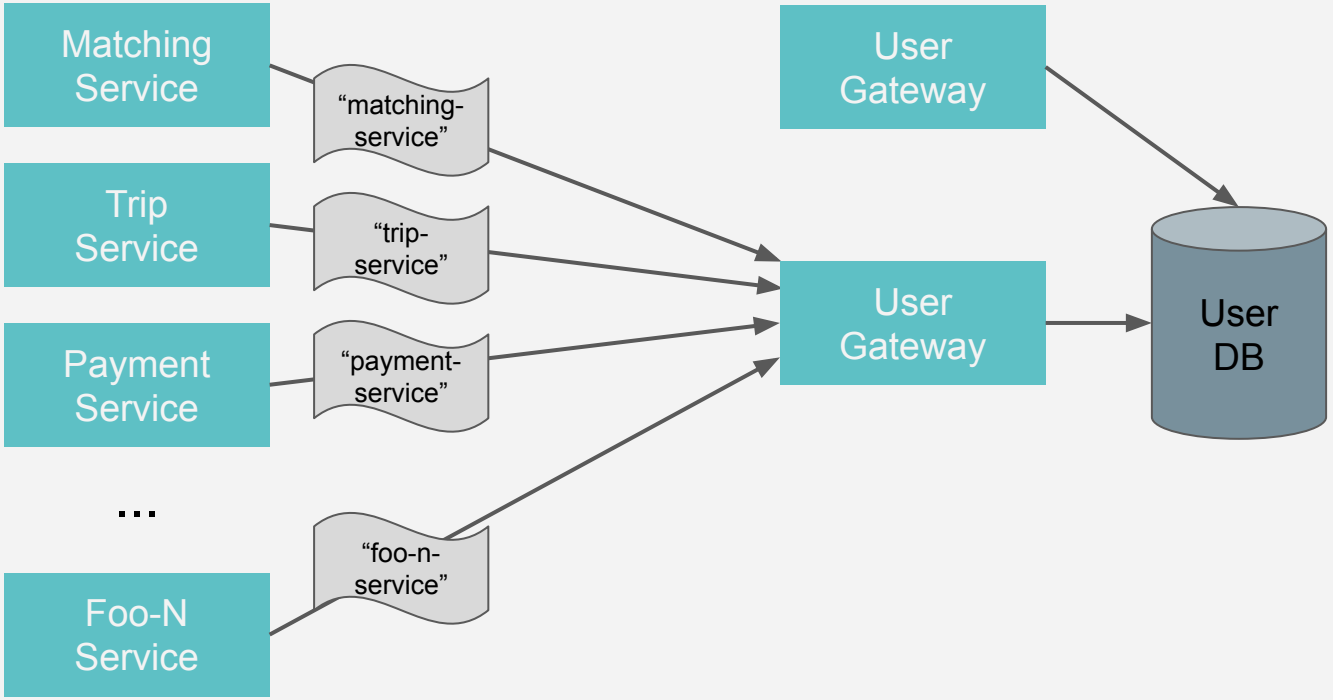
Growth: Service to Gateway (Proxied Data Access)



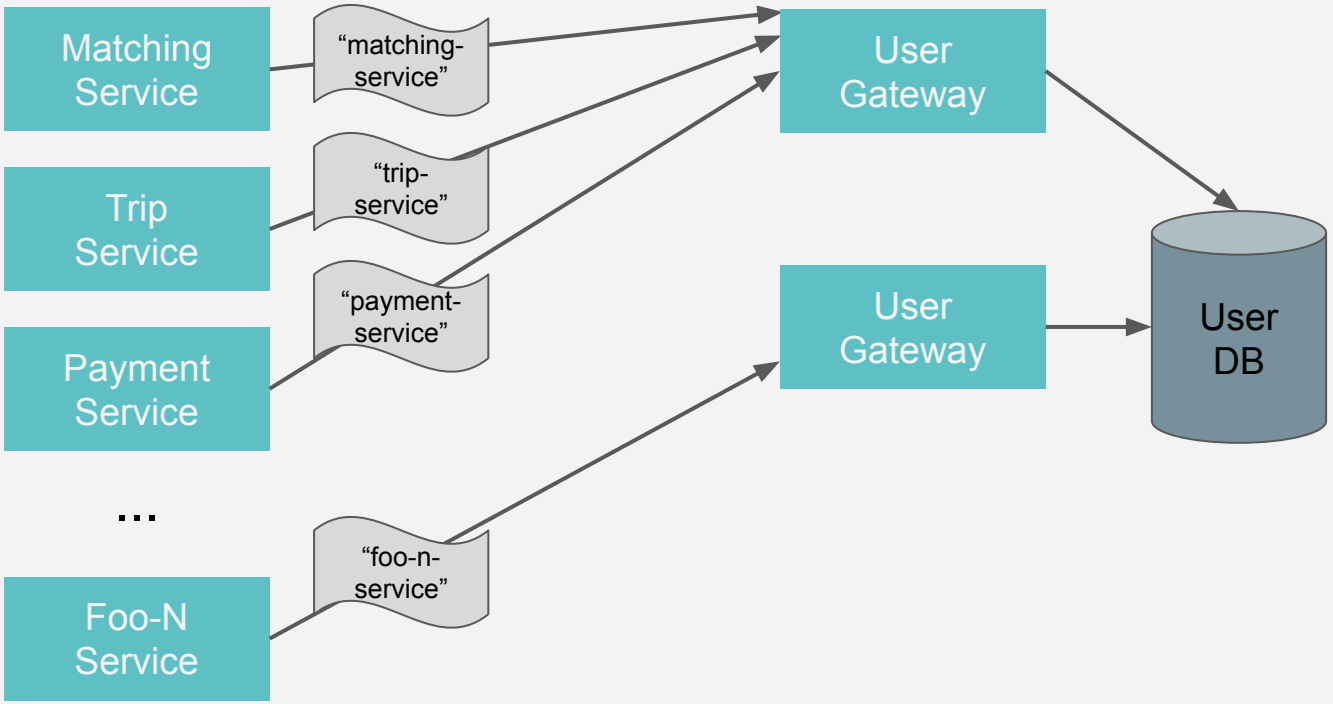
Growth: Service to Gateway (Proxied Data Access)



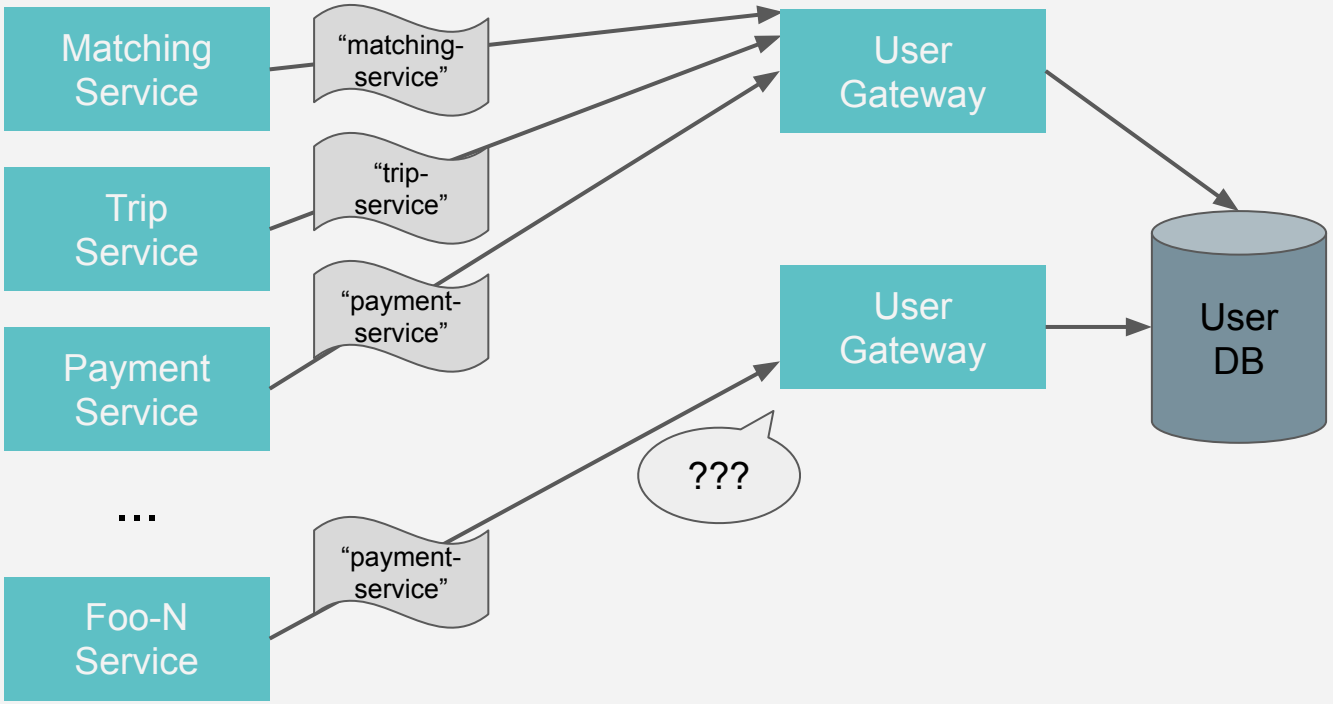
Late: Service to Gateway (Migration)



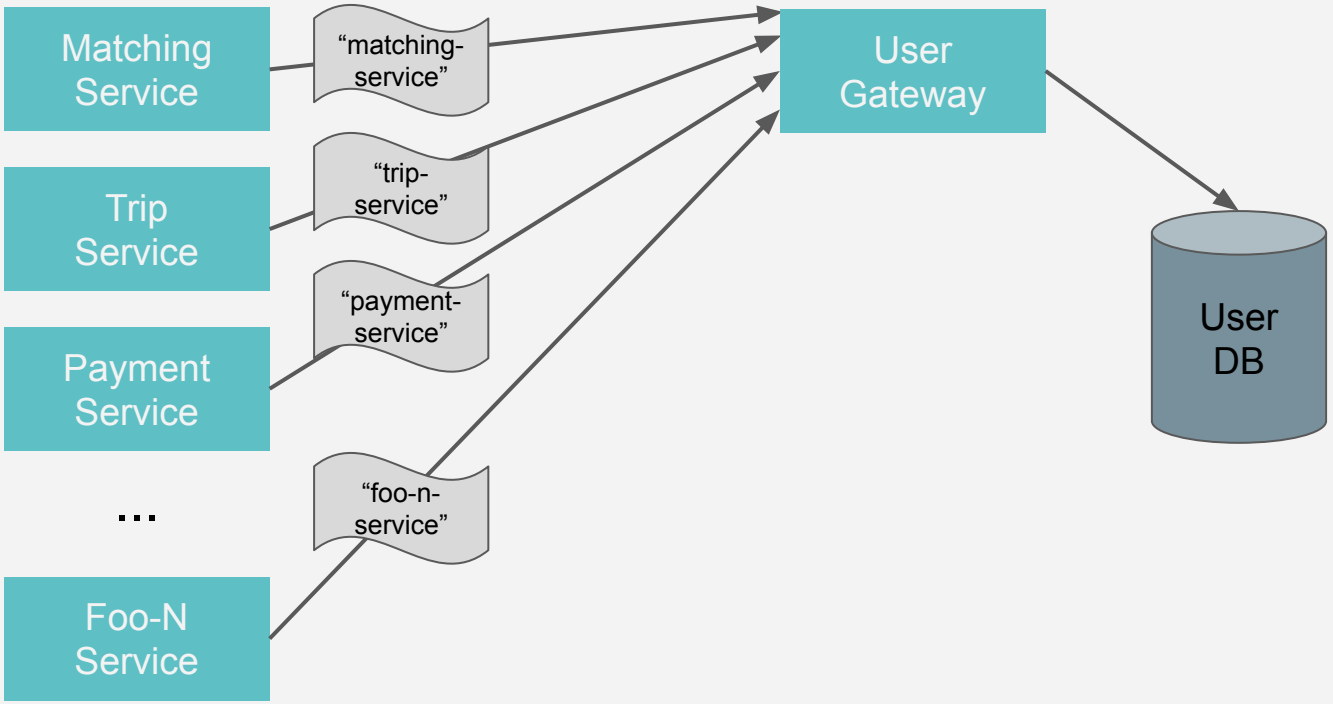
Late: Service to Gateway (Migration)



Late: Service to Gateway (Migration)



Late: Service to Gateway (Migration)



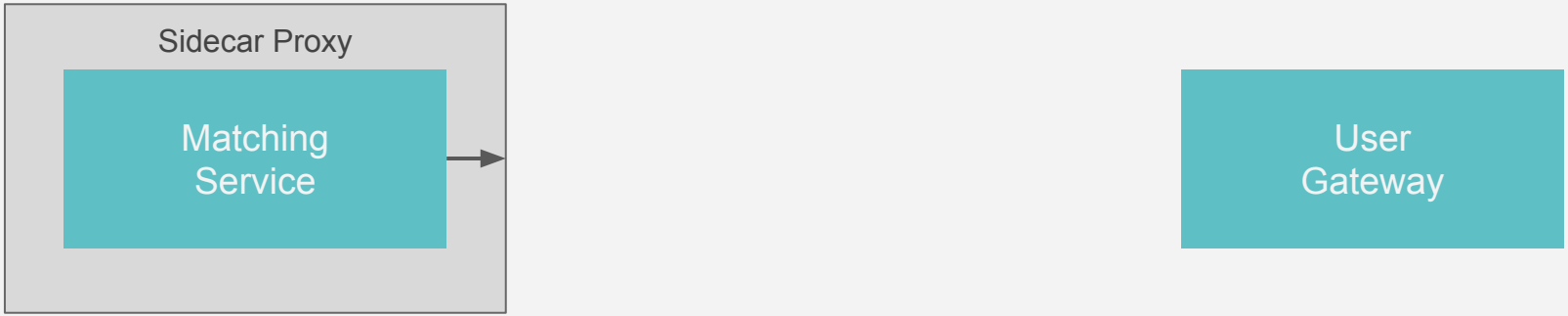
Implementation

- Libraries
  - Cross-language compatibility is hard
  - Breaking changes are nearly impossible
  - Maintenance burden

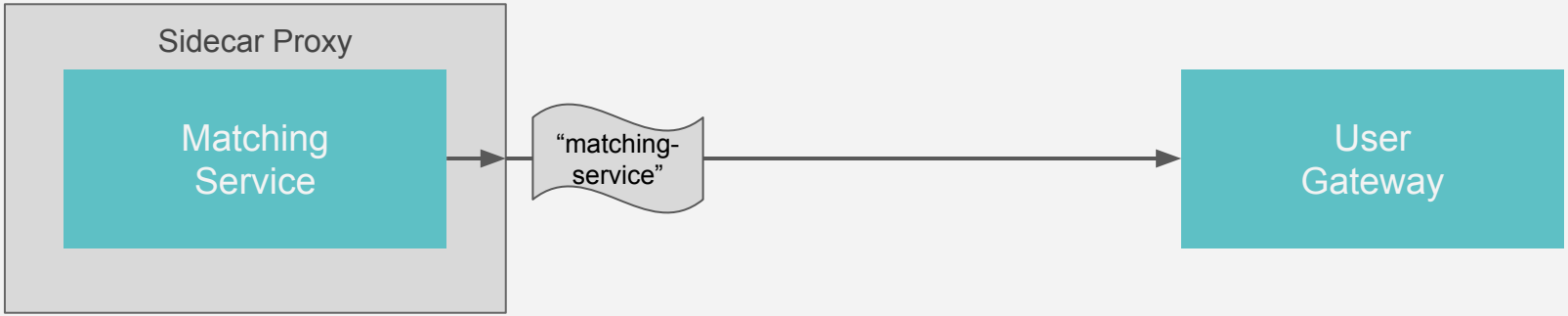
- Sidecar proxy
  - Language-agnostic
  - Encapsulated from application logic
  - Breaking changes are possible!

Sidecar Proxy

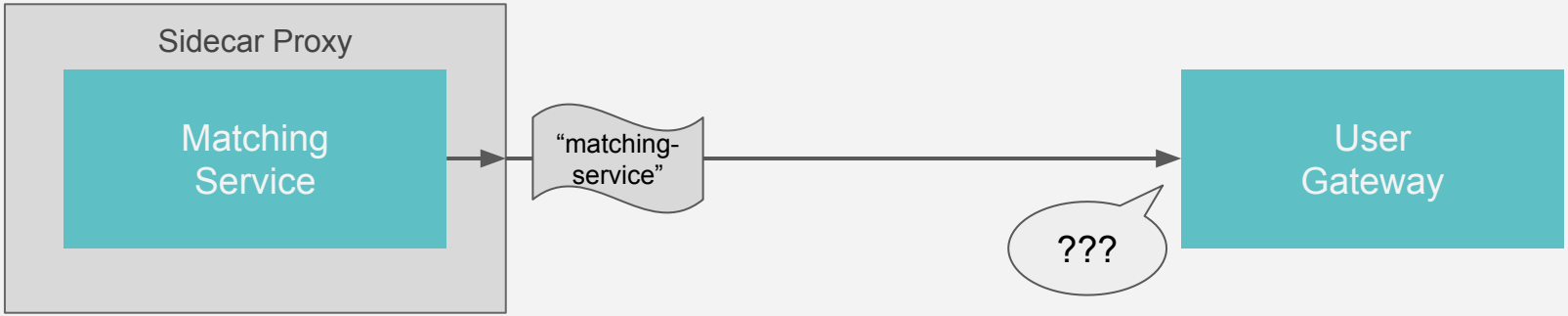
Sidecar Proxy



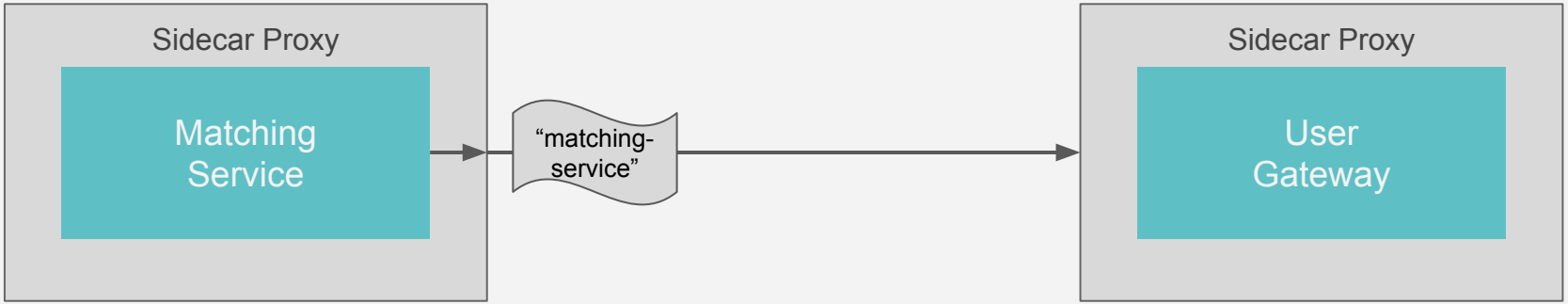
Sidecar Proxy



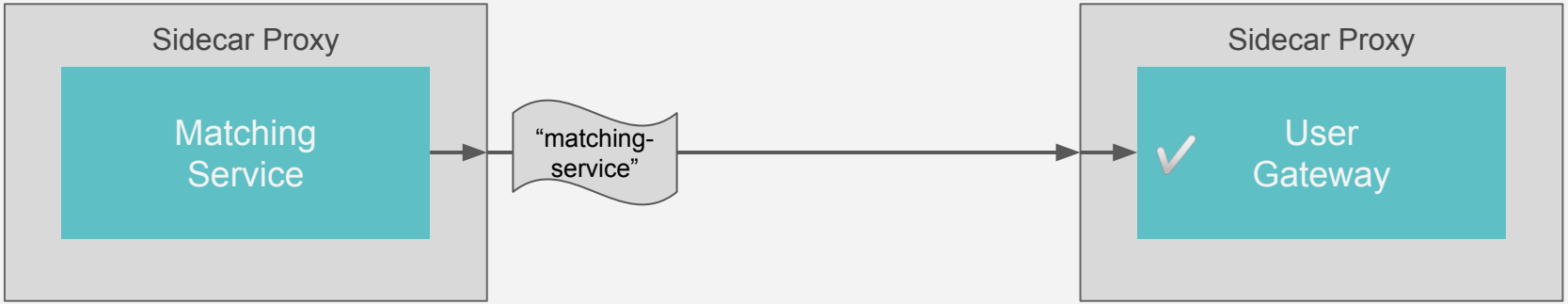
Sidecar Proxy



Sidecar Proxy



Sidecar Proxy



Q&A