Insecure Transit

Microservice Security

Sam Newman - QCON London 2018

HTTPS Everywhere!

HTTP + TLS

HTTP + TLS

Server guarantees!

HTTP + TLS

Server guarantees!

Payload not manipulated

HTTP + TLS

Server guarantees!

Payload not manipulated

Client guarantees?

HTTP + TLS

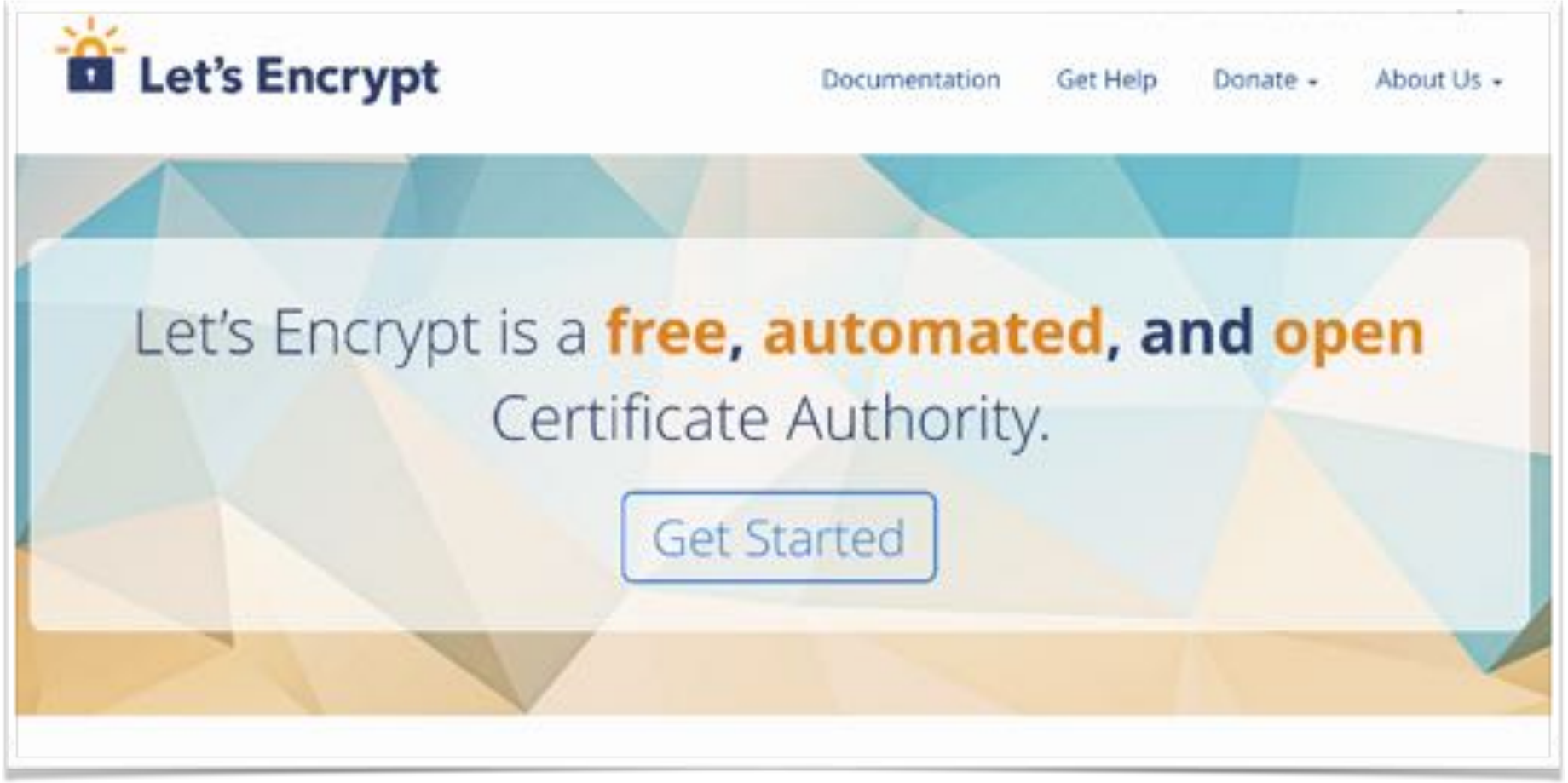
Server guarantees!

Payload not manipulated

Client guarantees?

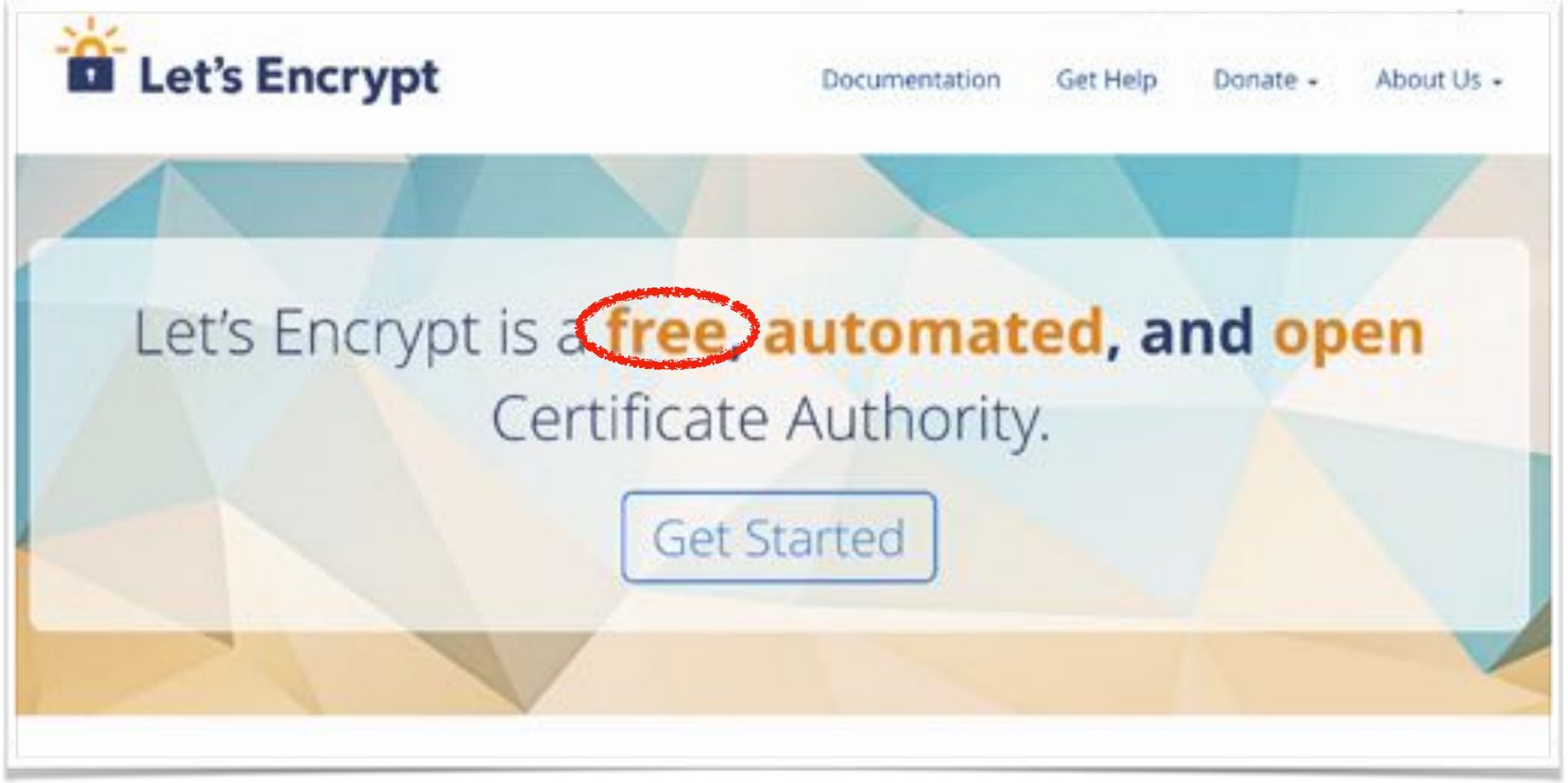
Certificate management can be painful

LET'S ENCRYPT



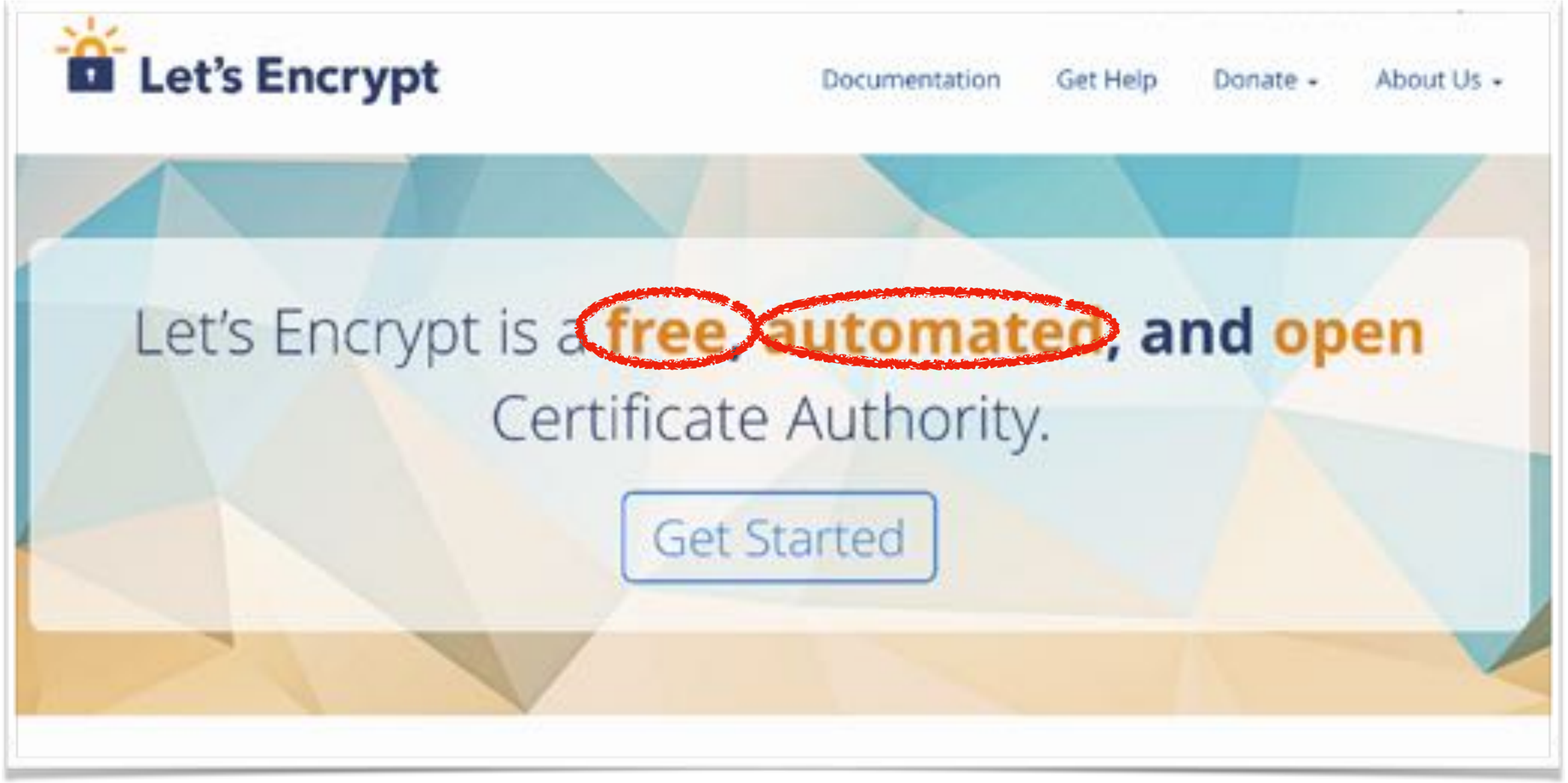
https://letsencrypt.org/

LET'S ENCRYPT



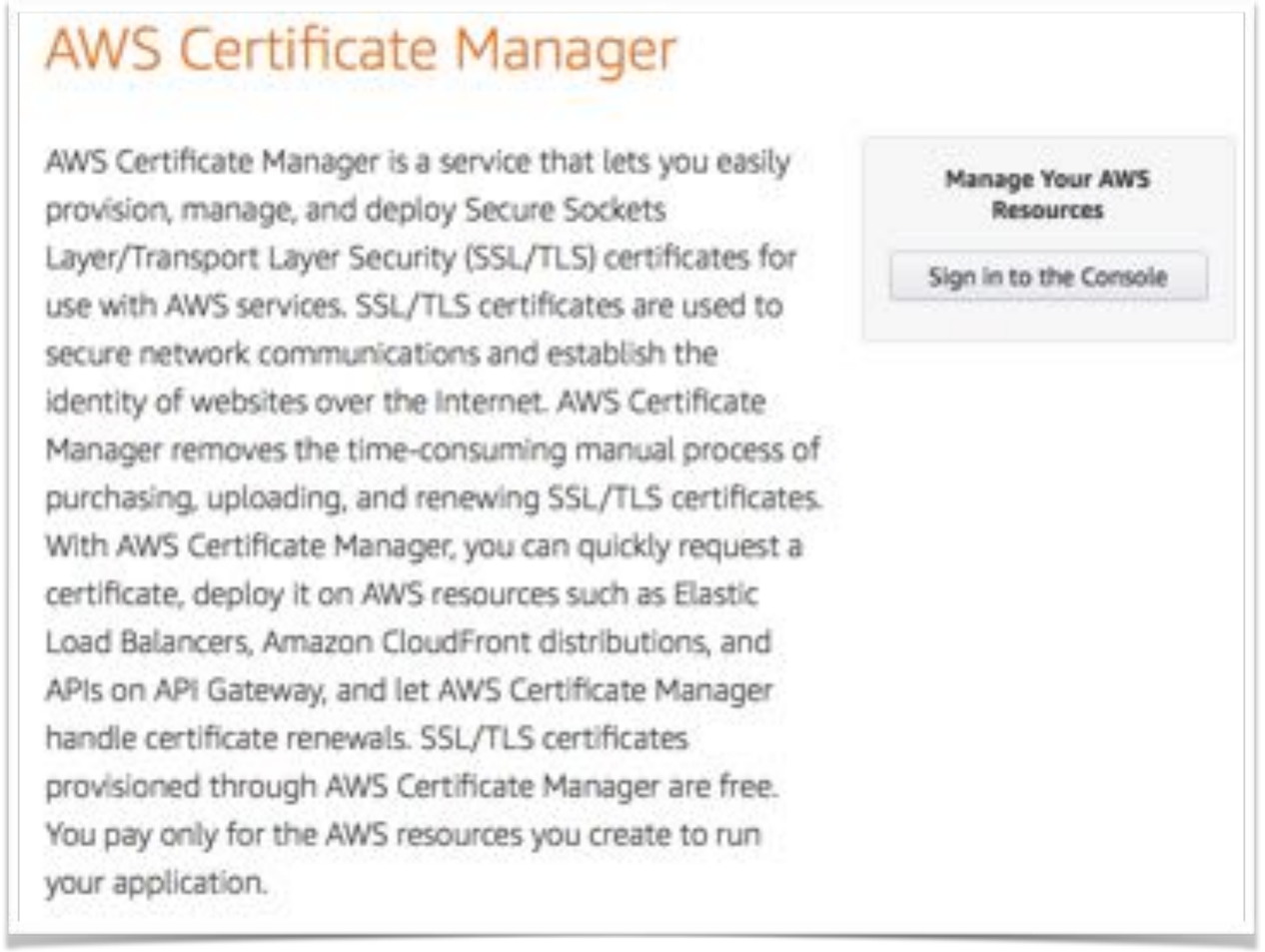
https://letsencrypt.org/

LET'S ENCRYPT



https://letsencrypt.org/

AWS CERTIFICATE MANAGER



https://aws.amazon.com/certificate-manager/

HOW DOES THIS STACK UP?

Server guarantees!

Payload not manipulated

Client guarantees?

Certificate management can be painful

HOW DOES THIS STACK UP?

Server guarantees!

Payload not manipulated

Client guarantees?

Certificate management can be painful

Observation of data

Manipulation of data

Restricting access to endpoints

Impersonation of endpoints

HOW DOES THIS STACK UP?

Server guarantees!

Payload not manipulated

Client guarantees?

Certificate management can be painful

Observation of data

Manipulation of data

Restricting access to endpoints

Impersonation of endpoints

HOW DOES THIS STACK UP?

Server guarantees!

Payload not manipulated

Client guarantees?

Certificate management can be painful


Observation of data


Manipulation of data

Restricting access to endpoints

Impersonation of endpoints

HOW DOES THIS STACK UP?

Server guarantees!

Payload not manipulated

Client guarantees?

Certificate management can be painful

✓ Observation of data

✓ Manipulation of data

? Restricting access to endpoints

Impersonation of endpoints

HOW DOES THIS STACK UP?

Server guarantees!

Payload not manipulated

Client guarantees?

Certificate management can be painful

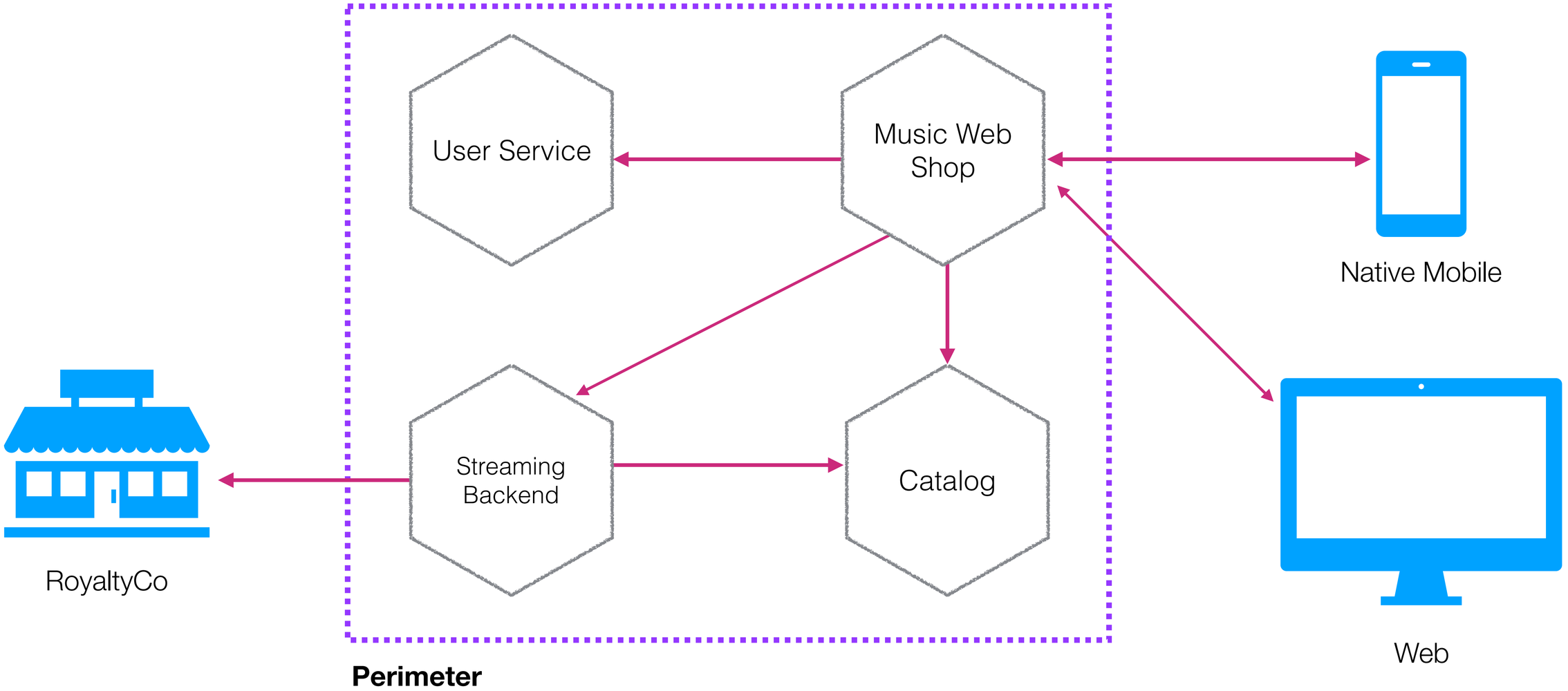
✓ Observation of data

✓ Manipulation of data

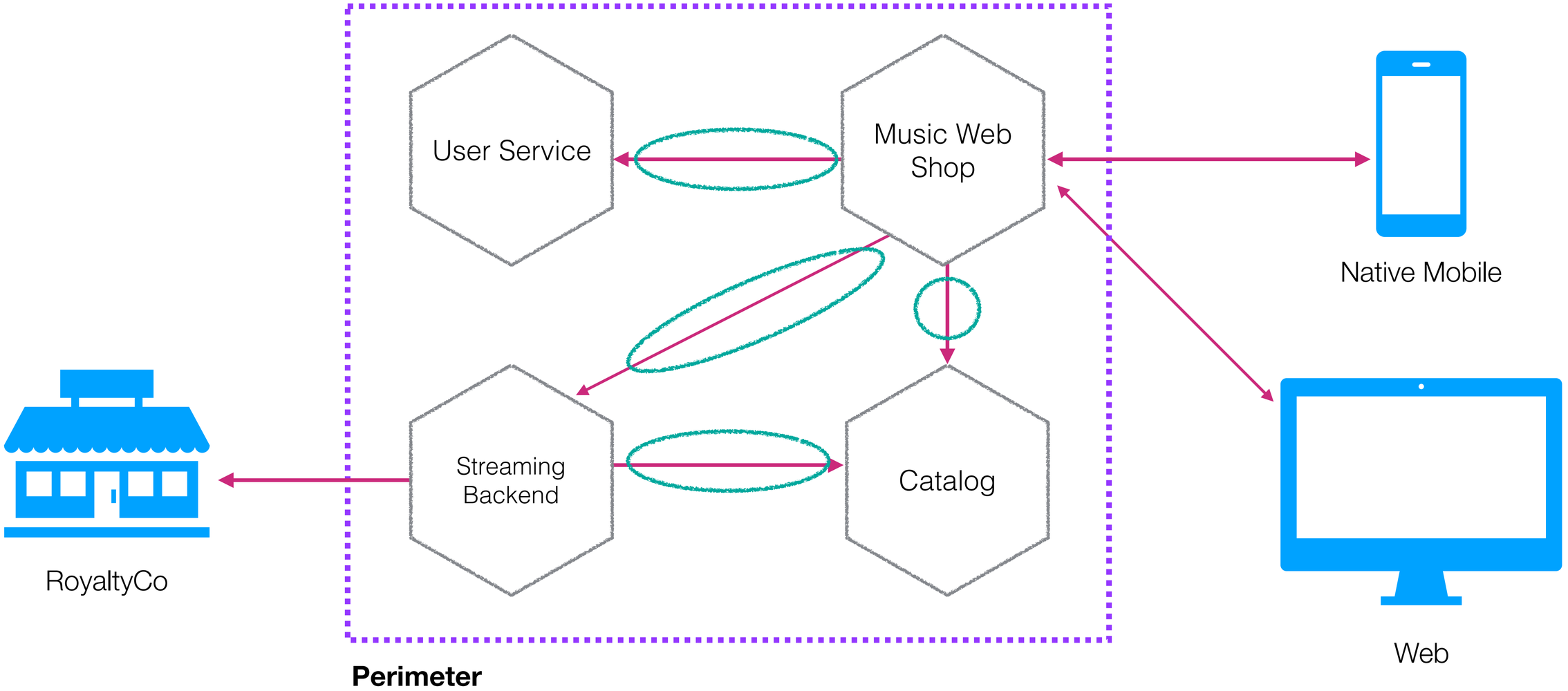
? Restricting access to endpoints

✓ Impersonation of endpoints

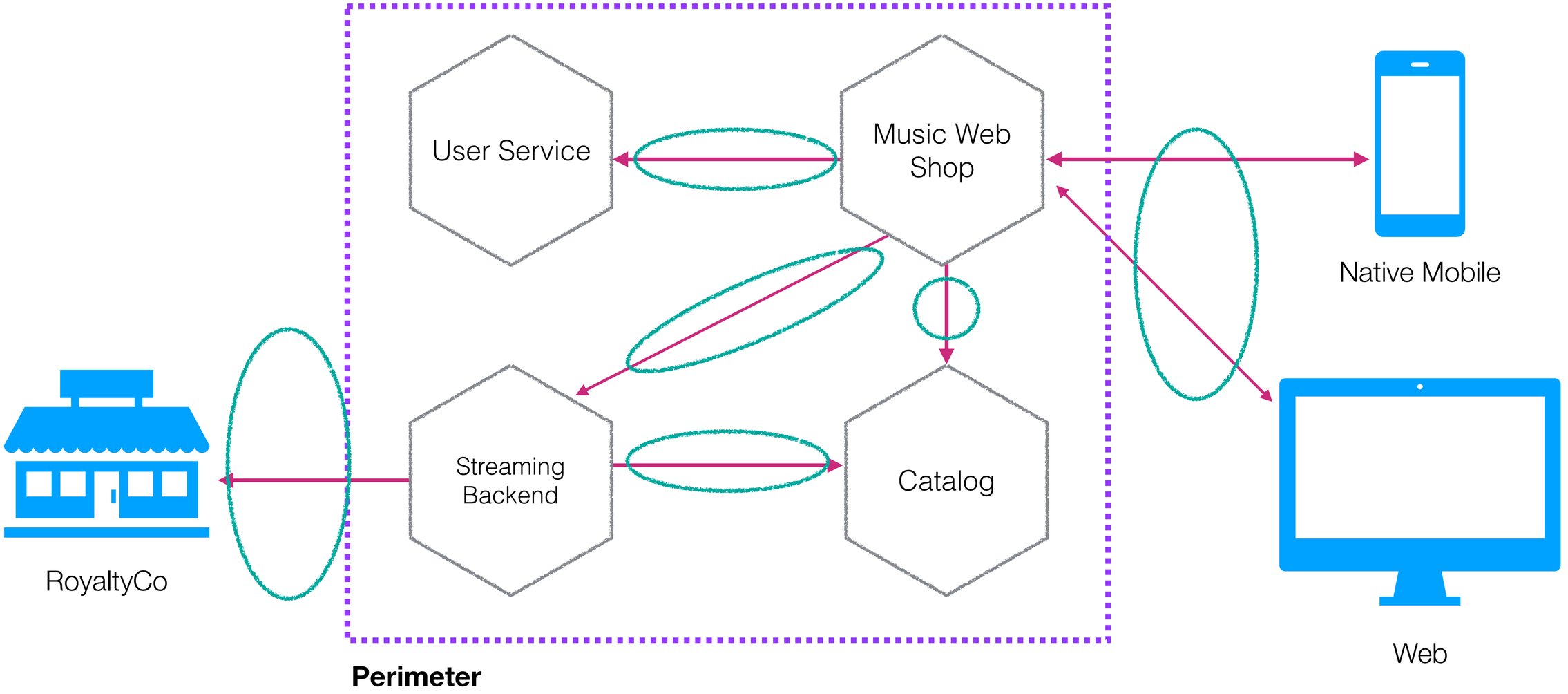
HTTPS EVERYWHERE!



HTTPS EVERYWHERE!

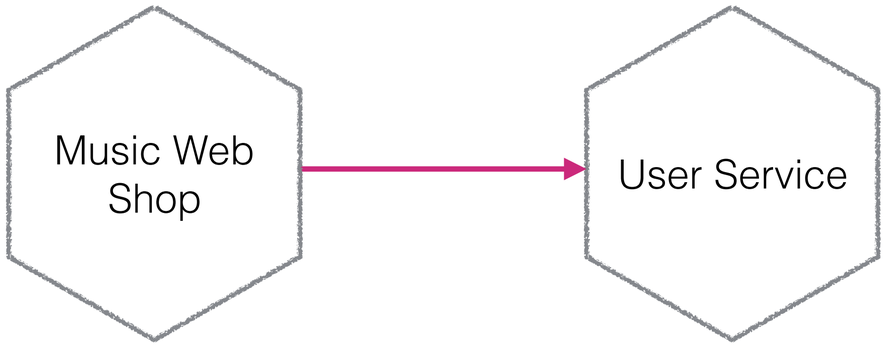


HTTPS EVERYWHERE!

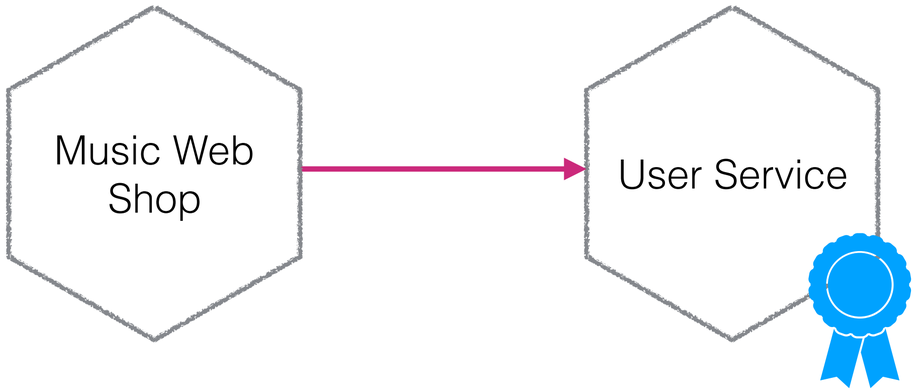


Mutual TLS

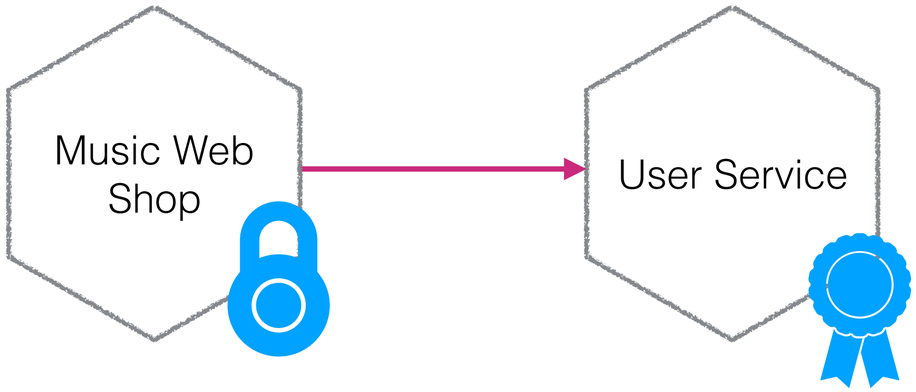
MUTUAL TLS



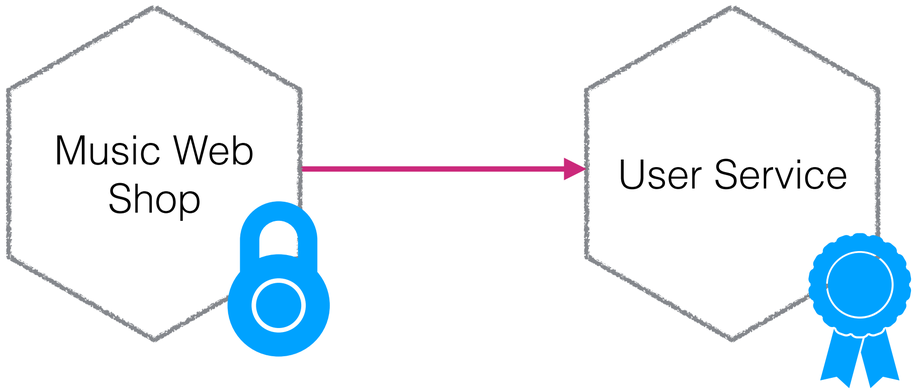
MUTUAL TLS



MUTUAL TLS

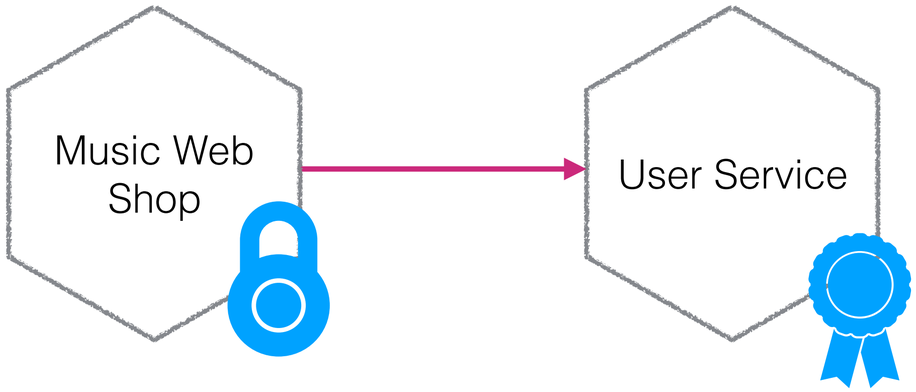


MUTUAL TLS



Client and server guarantees!

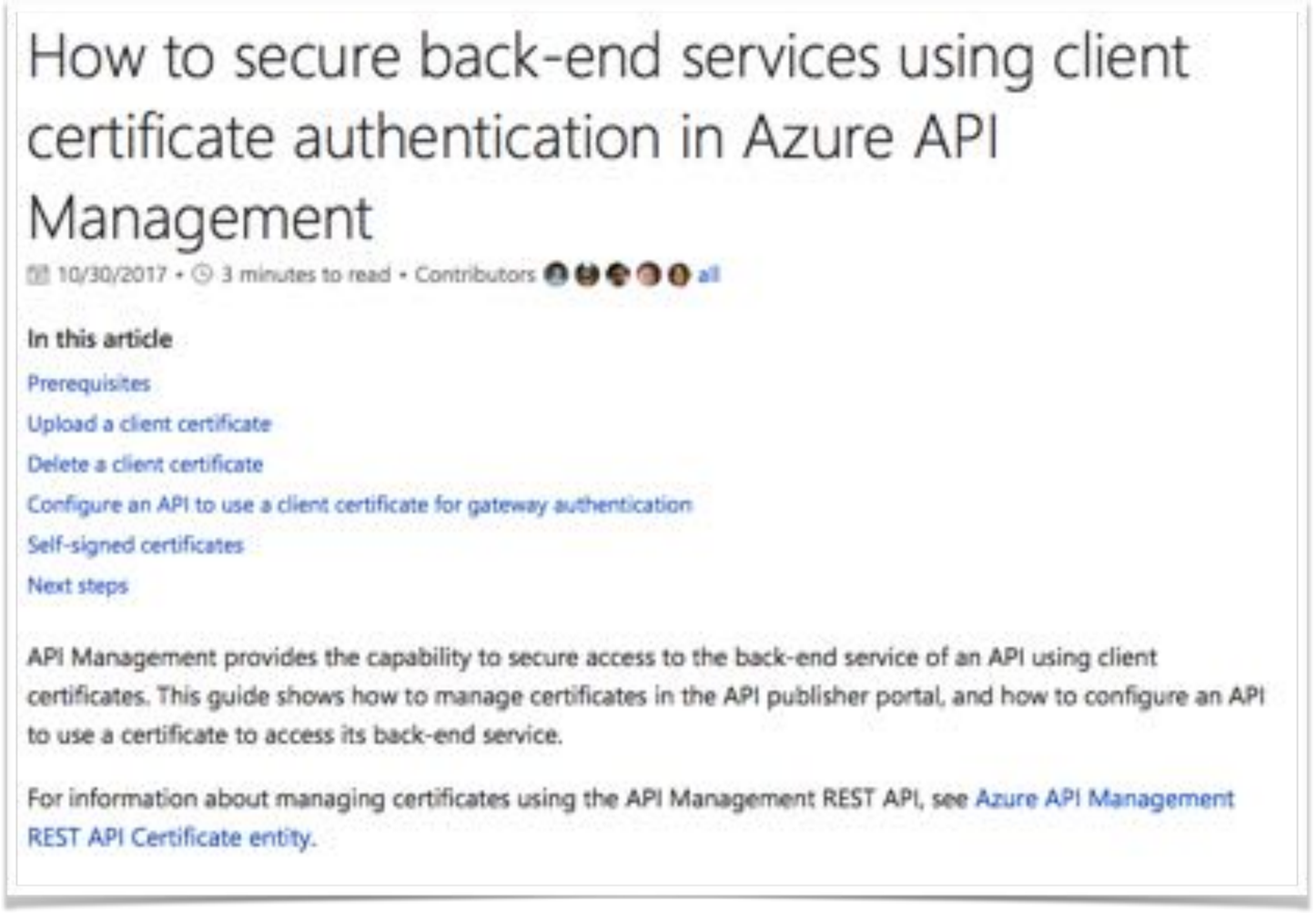
MUTUAL TLS



Client and server guarantees!

Certificate management is REALLY painful

AZURE - CLIENT-SIDE CERTIFICATE MANAGEMENT



How to secure back-end services using client certificate authentication in Azure API Management

10/30/2017 • 3 minutes to read • Contributors all

In this article

Prerequisites

Upload a client certificate

Delete a client certificate

Configure an API to use a client certificate for gateway authentication

Self-signed certificates

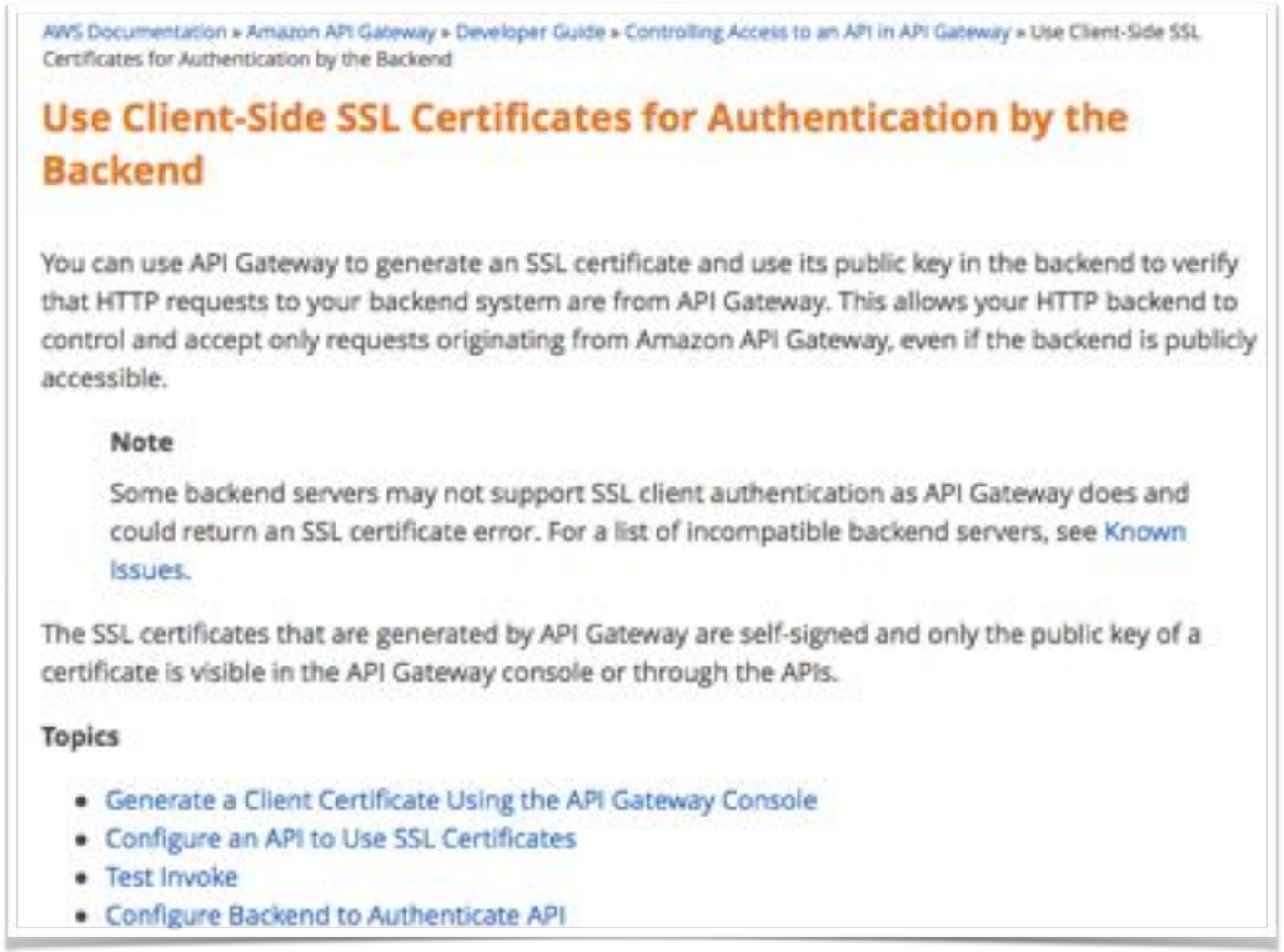
Next steps

API Management provides the capability to secure access to the back-end service of an API using client certificates. This guide shows how to manage certificates in the API publisher portal, and how to configure an API to use a certificate to access its back-end service.

For information about managing certificates using the API Management REST API, see Azure API Management REST API Certificate entity.

https://docs.microsoft.com/en-us/azure/api-management/api-management-howto-mutual-certificates

AWS - CLIENT-SIDE CERTIFICATE MANAGEMENT



AWS Documentation » Amazon API Gateway » Developer Guide » Controlling Access to an API in API Gateway » Use Client-Side SSL Certificates for Authentication by the Backend

Use Client-Side SSL Certificates for Authentication by the Backend

You can use API Gateway to generate an SSL certificate and use its public key in the backend to verify that HTTP requests to your backend system are from API Gateway. This allows your HTTP backend to control and accept only requests originating from Amazon API Gateway, even if the backend is publicly accessible.

> **Note**
>
> Some backend servers may not support SSL client authentication as API Gateway does and could return an SSL certificate error. For a list of incompatible backend servers, see Known Issues.

The SSL certificates that are generated by API Gateway are self-signed and only the public key of a certificate is visible in the API Gateway console or through the APIs.

Topics

- Generate a Client Certificate Using the API Gateway Console
- Configure an API to Use SSL Certificates
- Test Invoke
- Configure Backend to Authenticate API

https://docs.aws.amazon.com/apigateway/latest/developerguide/getting-started-client-side-ssl-authentication.html

MUTUAL TLS

Observation of data

Manipulation of data

Restricting access to endpoints

Impersonation of endpoints

MUTUAL TLS

✔ Observation of data

Manipulation of data

Restricting access to endpoints

Impersonation of endpoints

MUTUAL TLS

✓ Observation of data

✓ Manipulation of data

Restricting access to endpoints

Impersonation of endpoints

MUTUAL TLS

✓ Observation of data

✓ Manipulation of data

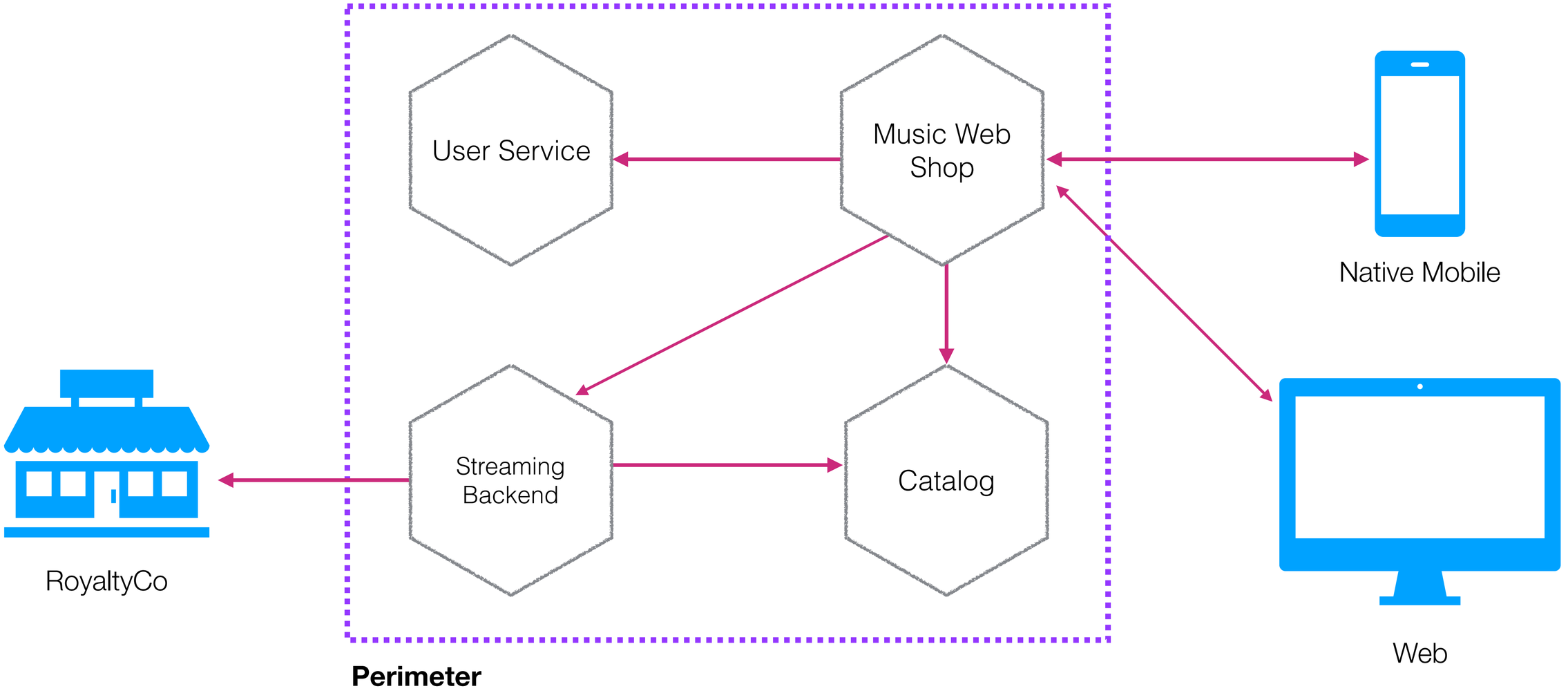
✓ Restricting access to endpoints

Impersonation of endpoints

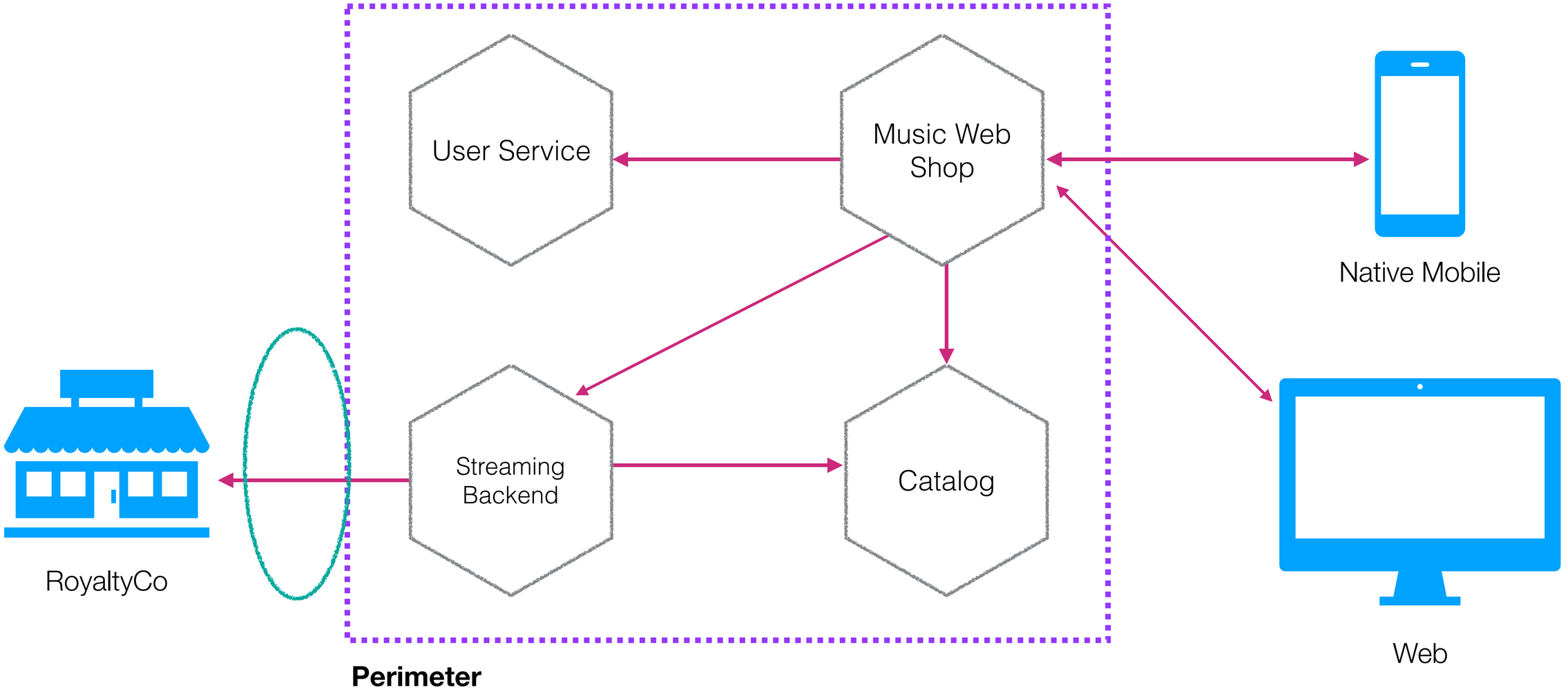
MUTUAL TLS

- ✓ Observation of data
- ✓ Manipulation of data
- ✓ Restricting access to endpoints
- ✓ Impersonation of endpoints

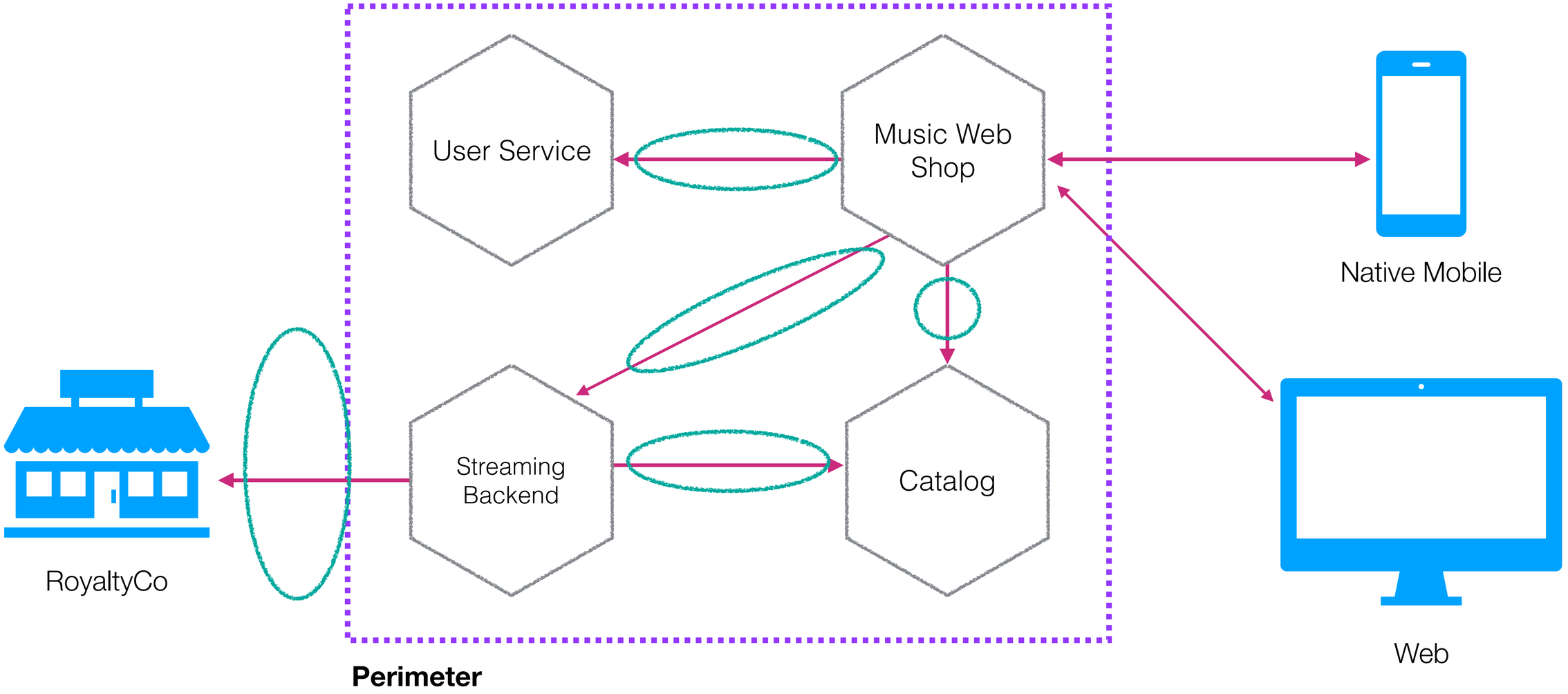
MUTUAL TLS


User Service
Music Web Shop
Native Mobile
Streaming Backend
Catalog
RoyaltyCo
Web
Perimeter

MUTUAL TLS


User Service
Music Web Shop
Native Mobile
Streaming Backend
Catalog
RoyaltyCo
Web
Perimeter

MUTUAL TLS


User Service
Music Web Shop
Native Mobile
Streaming Backend
Catalog
RoyaltyCo
Web
Perimeter

OTHER PROTOCOLS?

OTHER PROTOCOLS?

Synchronous, via HTTP, RPC or other

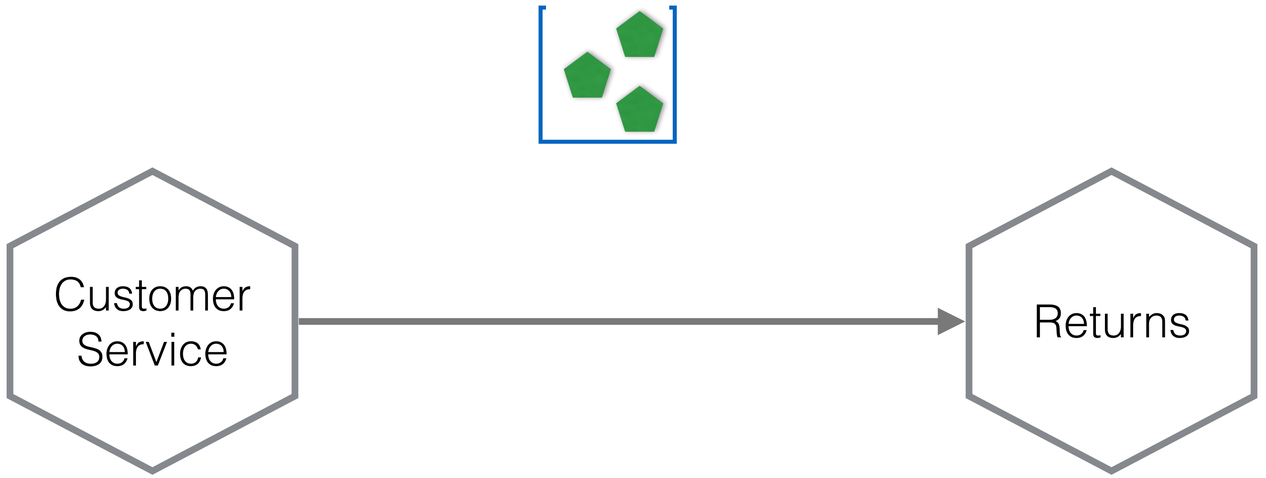
OTHER PROTOCOLS?

Async, via e.g. via message broker

Synchronous, via HTTP, RPC or other

OTHER PROTOCOLS?

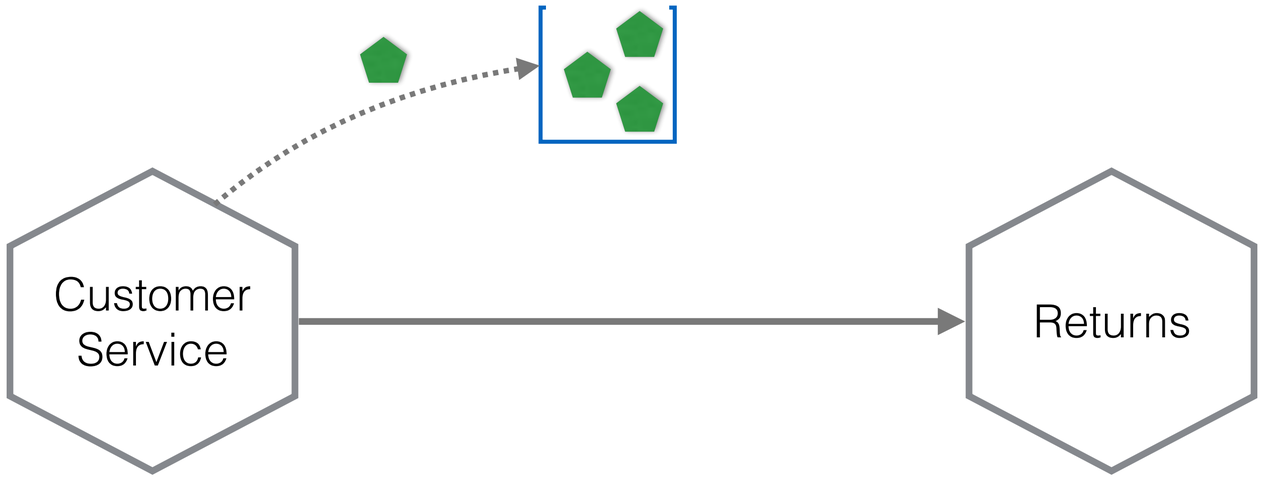
Async, via e.g. via message broker



Synchronous, via HTTP, RPC or other

OTHER PROTOCOLS?

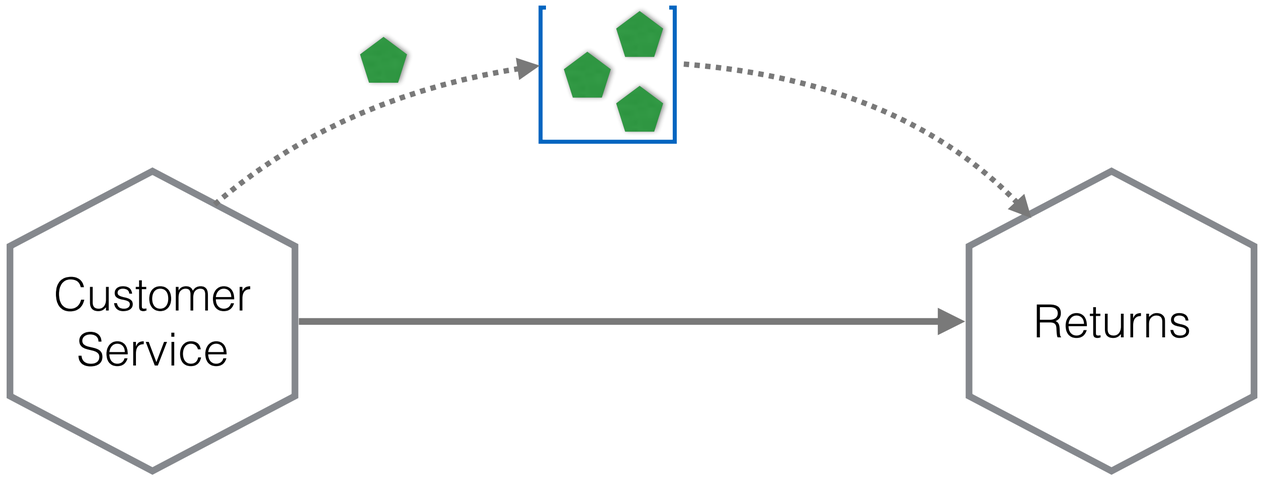
Async, via e.g. via message broker



Synchronous, via HTTP, RPC or other

OTHER PROTOCOLS?

Async, via e.g. via message broker



Synchronous, via HTTP, RPC or other

RABBITMQ

TLS Support

Intro

RabbitMQ has inbuilt support for TLS. This includes client connections and popular plugins, where applicable, such as Federation links. It is also possible to use TLS to encrypt inter-node connections in clusters.

This guide covers various topics related to TLS in RabbitMQ:

- Enabling TLS listeners in RabbitMQ
- How to generate self-signed certificates for development and QA environments
- TLS configuration in Java and .NET clients
- Known vulnerabilities and their migration
- TLS version and cipher suite configuration
- Certificate chain validation depth
- Tools that can be used to evaluate a TLS setup

and more. It is not, however, a primer on TLS, encryption, Public Key Infrastructure and related topics, so the concepts are covered very briefly. A number of beginner-oriented primers are available elsewhere on the Web: one two, three, four.

RABBITMQ

TLS Support

Intro

RabbitMQ has inbuilt support for TLS. This includes client connections and popular plugins, where applicable, such as Federation links. It is also possible to use TLS to encrypt inter-node connections in clusters.

This guide covers various topics related to TLS in RabbitMQ:

- Enabling TLS listeners in RabbitMQ
- How to generate self-signed certificates for development and QA environments
- TLS configuration in Java and .NET clients
- Known vulnerabilities and their migration
- TLS version and cipher suite configuration
- Certificate chain validation depth
- Tools th

and mor
topics, s
available

Erlang/OTP Requirements for TLS Support

In order to support TLS connections, RabbitMQ needs TLS and crypto-related modules to be available in the Erlang/OTP installation. The recommended Erlang/OTP version to use with TLS is the most recent supported Erlang release. Earlier versions, even if they are supported, may work for most certificates but have known limitations (see below).

RABBITMQ

TLS Support

Intro

RabbitMQ has inbuilt support for TLS. This includes client connections and popular plugins, where applicable, such as Federation links. It is also possible to use TLS to encrypt inter-node connections in clusters.

This guide covers various topics related to TLS in RabbitMQ:

- Enabling TLS listeners in RabbitMQ
- How to generate self-signed certificates for development and QA environments
- TLS configuration in Java and .NET clients
- Known vulnerabilities and their migration
- TLS version and cipher suite configuration
- Certificate chain validation depth
- Tools th

and mor
topics, s
available

Erlang/OTP Requirements for TLS Support

In order to support TLS connections, RabbitMQ needs TLS and crypto-related modules to be available in the Erlang/OTP installation. The recommended Erlang/OTP version to use with TLS is the most recent supported Erlang release. Earlier versions, even if they are supported, may work for most certificates but have known limitations (see below).

Access Control (Authentication, Authorisation) in RabbitMQ

This document describes authentication and authorisation machinery that implements access control. Authentication backends should not be confused with authentication mechanisms in AMQP 0-9-1.

A separate guide covers multiple topics around passwords. It is only applicable to the internal authentication backend.

Terminology and Definitions

Authentication and authorisation are often confused or used interchangeably. That's wrong and in RabbitMQ, the two are separated. For the sake of simplicity, we'll define authentication as "identifying who the user is" and authorisation as "determining what the user is and isn't allowed to do."

Default Virtual Host and User

When the server first starts running, and detects that its database is uninitialised or has been deleted, it initialises a fresh database with the following resources:

KAFKA

TLS, Kerberos, SASL, and Authorizer in Apache Kafka 0.9 – Enabling New Encryption, Authorization, and Authentication Features

Apache Kafka is frequently used to store critical data making it one of the most important components of a company's data infrastructure. Our goal is to make it possible to run Kafka as a central platform for streaming data, supporting anything from a single app to a whole company. Multi-tenancy is an essential requirement in achieving this vision and, in turn, security features are crucial for multi-tenancy.

Previous to 0.9, Kafka had no built-in security features. One could lock down access at the network level but this is not viable for a big shared multi-tenant cluster being used across a large company. Consequently securing Kafka has been one of the most requested features. Security is of particular importance in today's world where cyber-attacks are a common occurrence and the threat of data breaches is a reality for businesses of all sizes, and at all levels from individual users to whole government entities.

Four key security features were added in Apache Kafka 0.9, which is included in the Confluent Platform 2.0:

KAFKA

TLS, Kerberos, SASL, and Authorizer in Apache Kafka 0.9 – Enabling New Encryption, Authorization, and Authentication Features

Apache Kafka is frequently used to store
company's data infrastructure. Our goal i
supporting anything from a single app to
this vision and, in turn, security features a

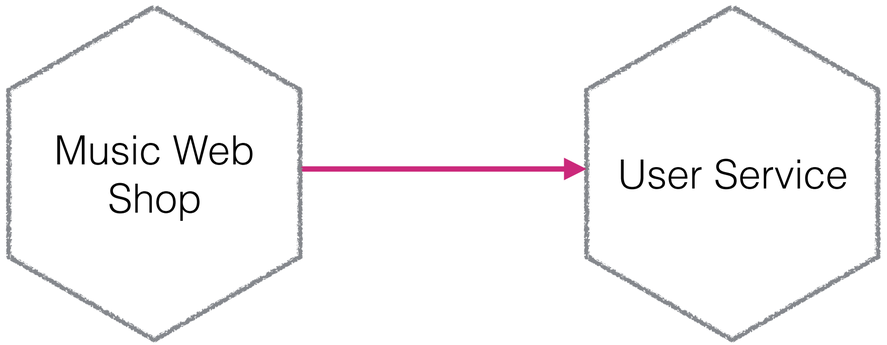
Previous to 0.9, Kafka had no built-in secu
not viable for a big shared multi-tenant cl
has been one of the most requested featu
attacks are a common occurrence and th
levels from individual users to whole government entities.

Four key security features were added in Apache Kafka 0.9, which is included in the Confluent Platform 2.0:

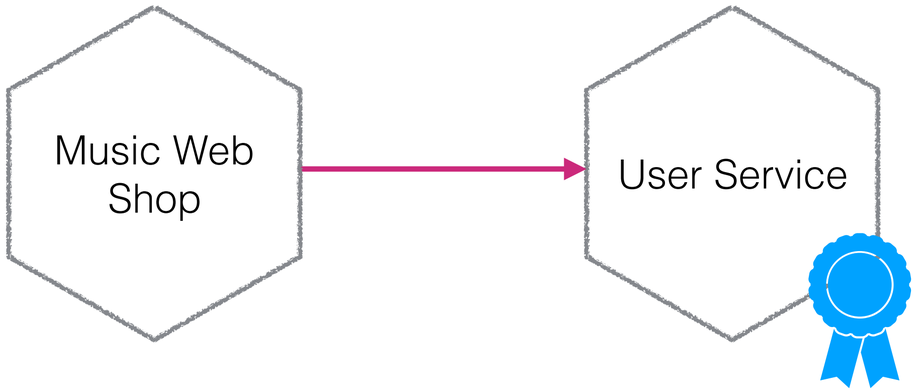
Four key security features were added in Apache Kafka 0.9, which is included in the Confluent Platform 2.0:

1. Administrators can require client authentication using either Kerberos or Transport Layer Security (TLS) client certificates, so that Kafka brokers know who is making each request
2. A Unix-like permissions system can be used to control which users can access which data.
3. Network communication can be encrypted, allowing messages to be securely sent across untrusted networks.
4. Administrators can require authentication for communication between Kafka brokers and ZooKeeper.

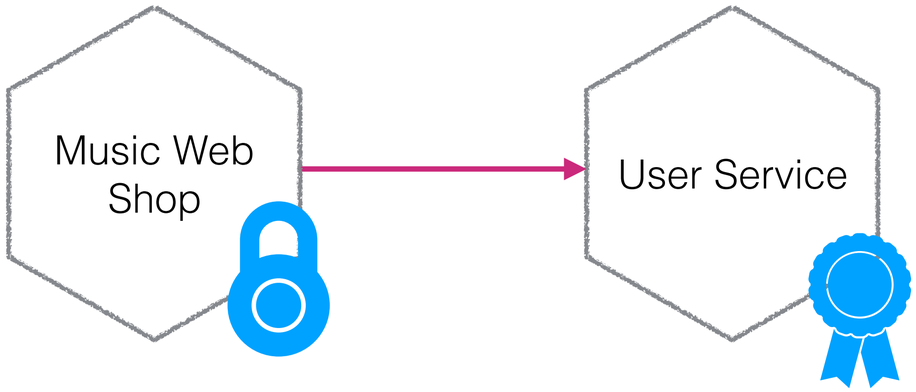
TRANSPORT AUTHENTICATION



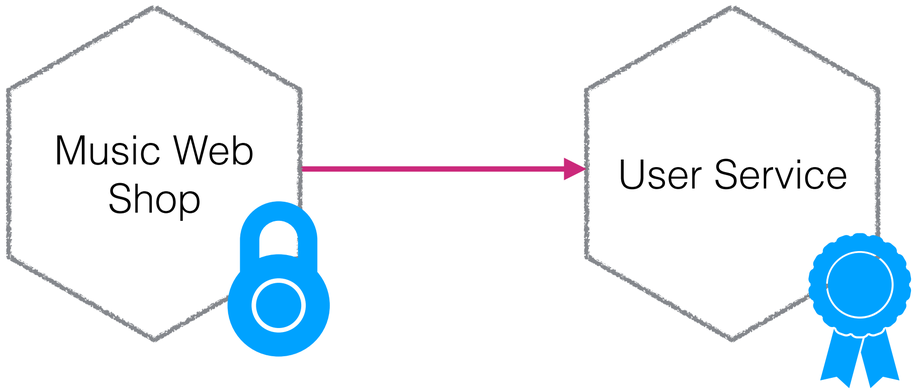
TRANSPORT AUTHENTICATION



TRANSPORT AUTHENTICATION

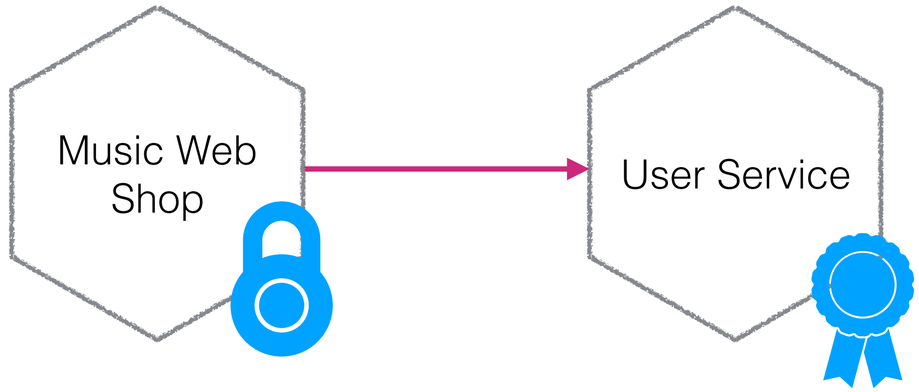


TRANSPORT AUTHENTICATION



Server-side identity

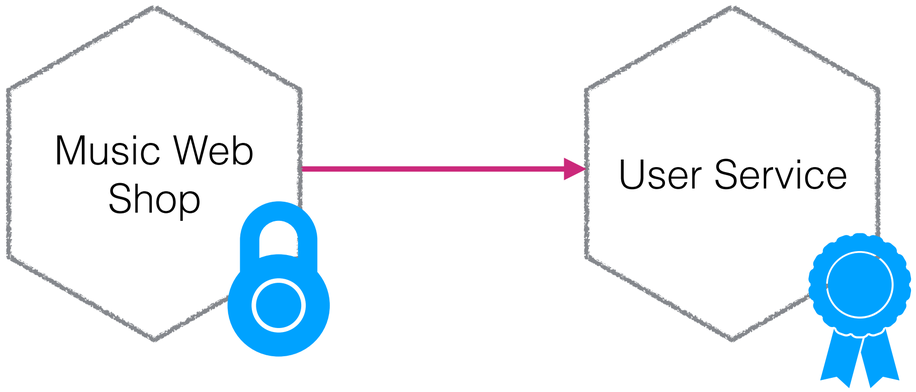
TRANSPORT AUTHENTICATION



Client-side identity

Server-side identity

TRANSPORT AUTHENTICATION



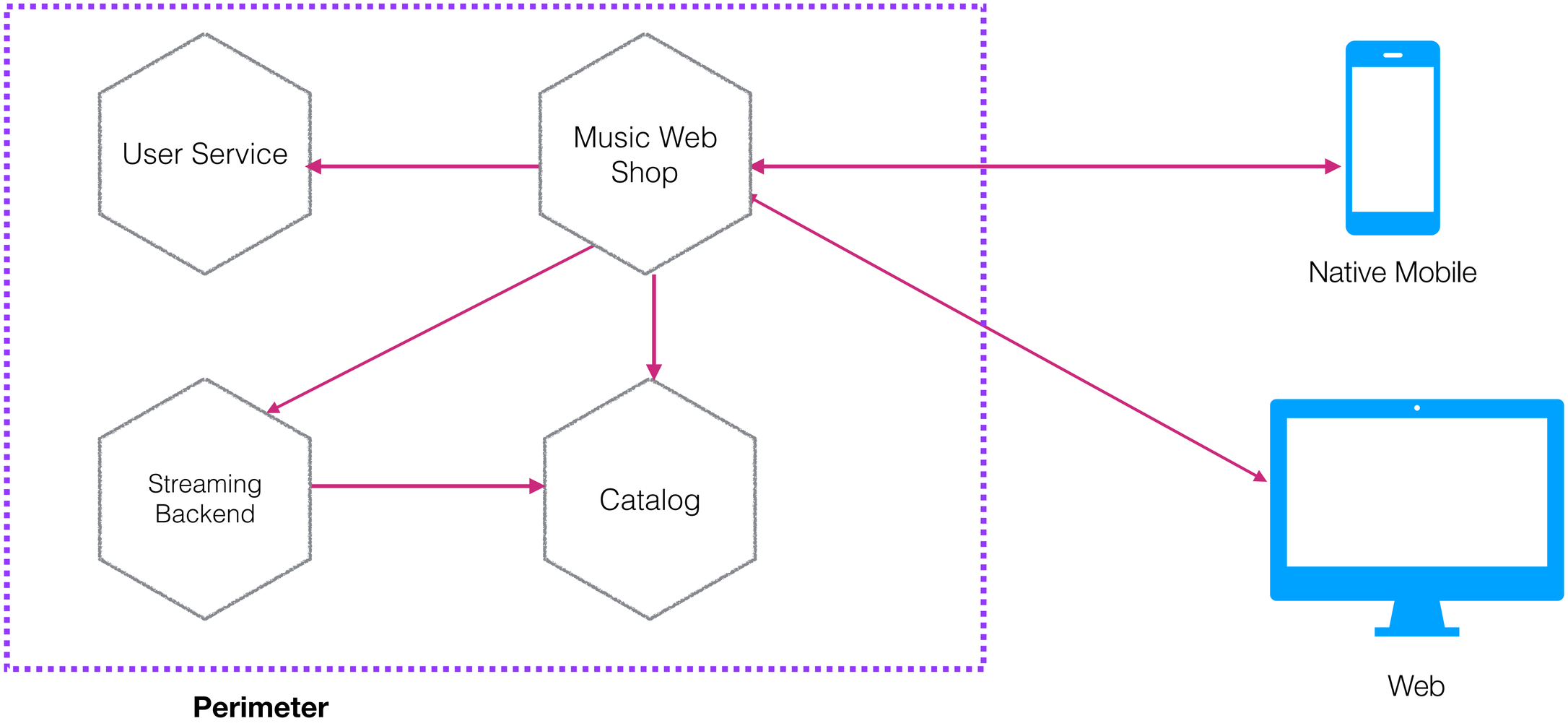
Client-side identity

Server-side identity

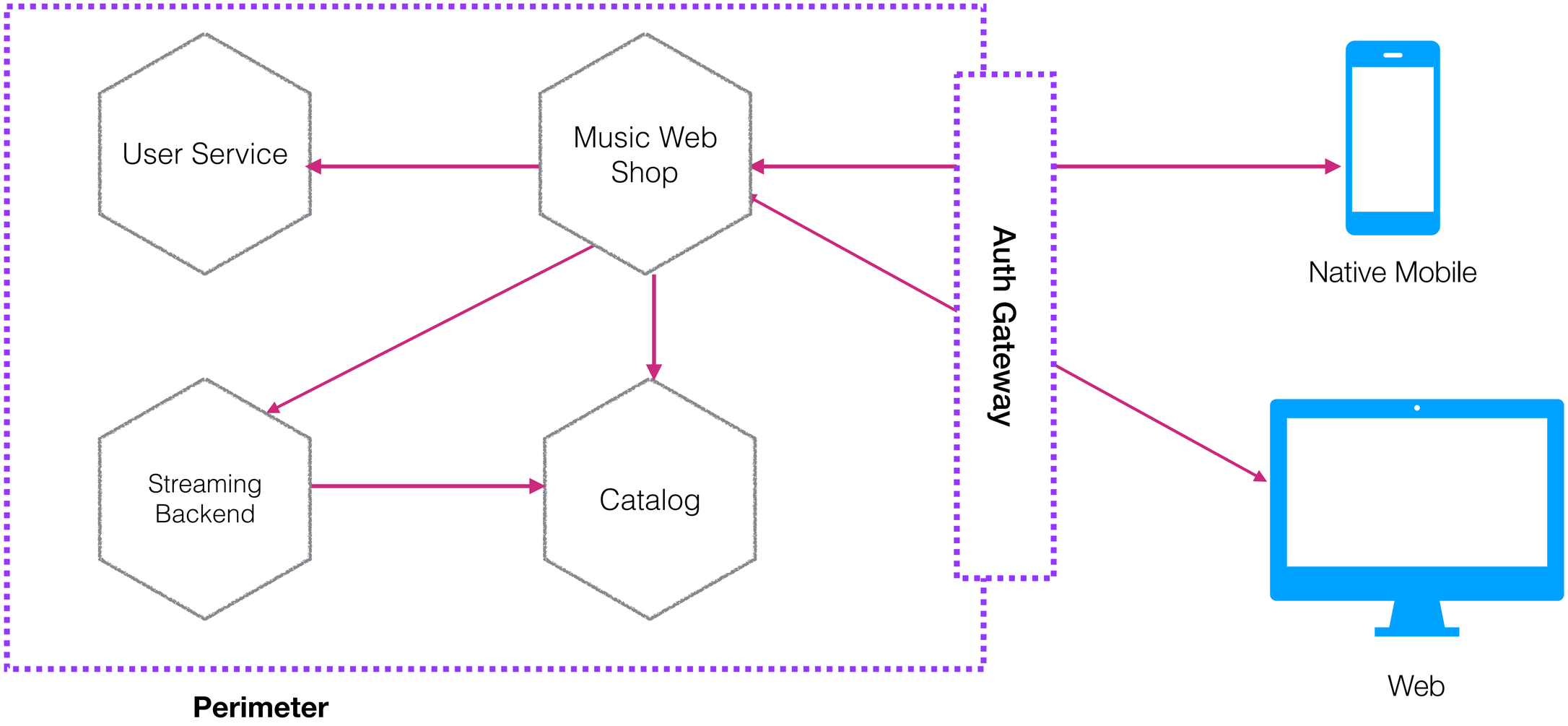
= service-to-service authentication

https://www.flickr.com/photos/madfishdigital/35652755551/

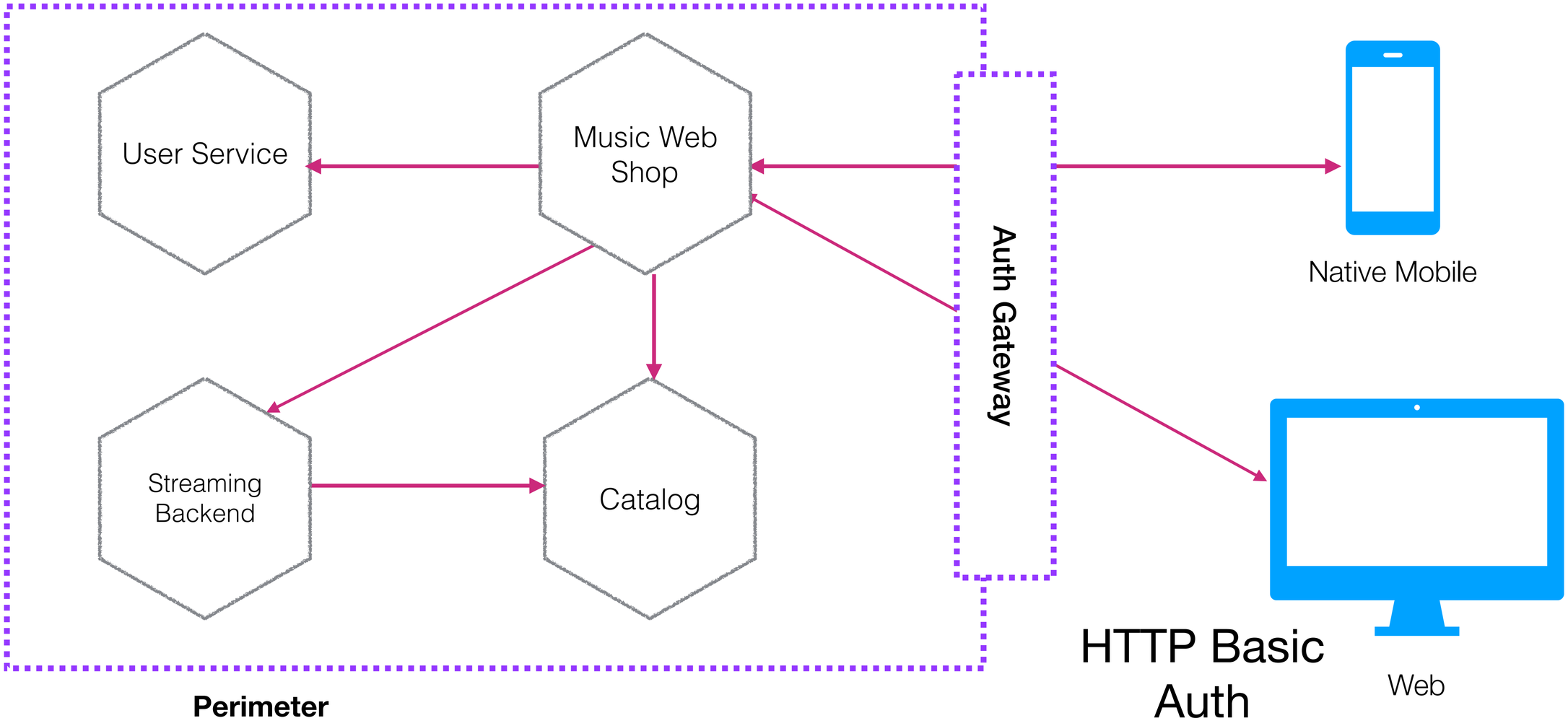
USER AUTHENTICATION - PROXY-BASED



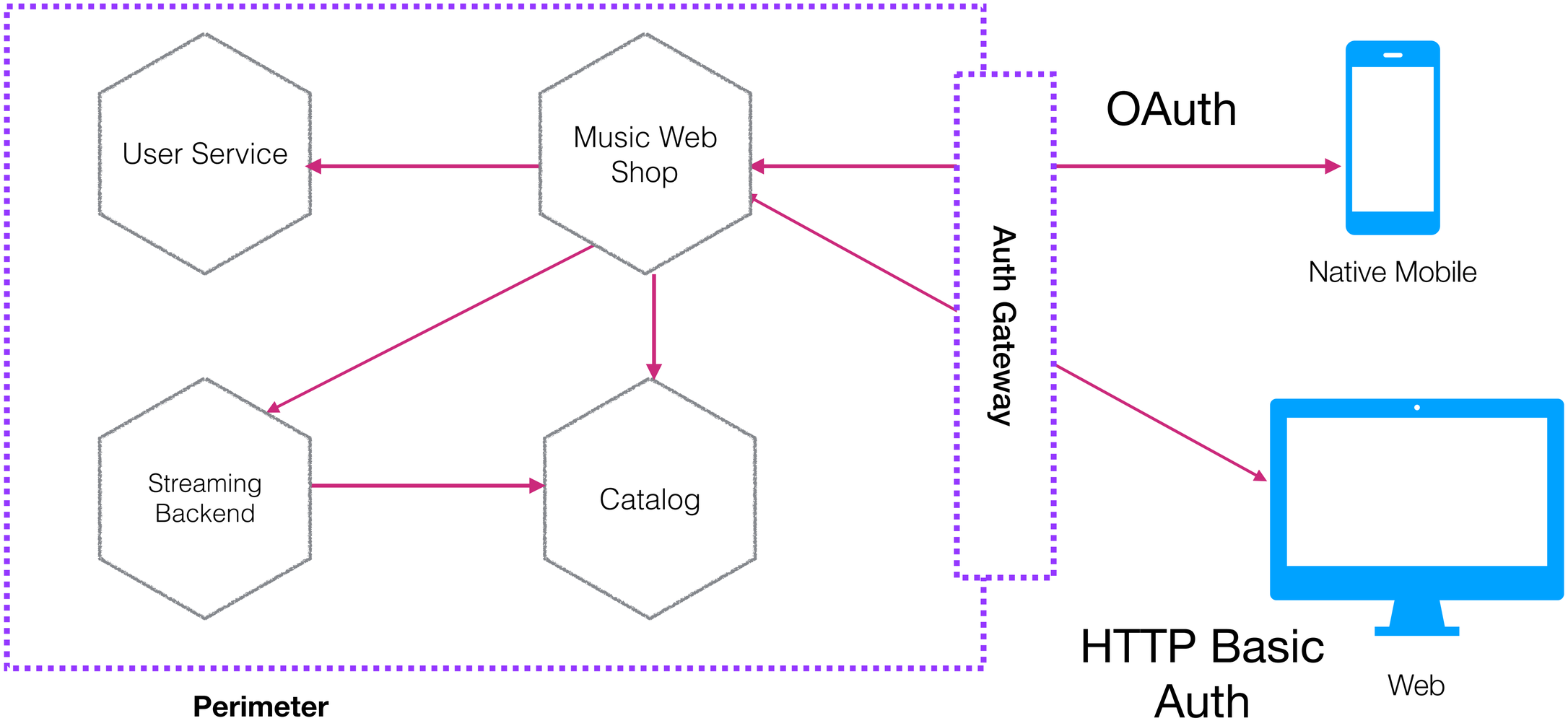
USER AUTHENTICATION - PROXY-BASED



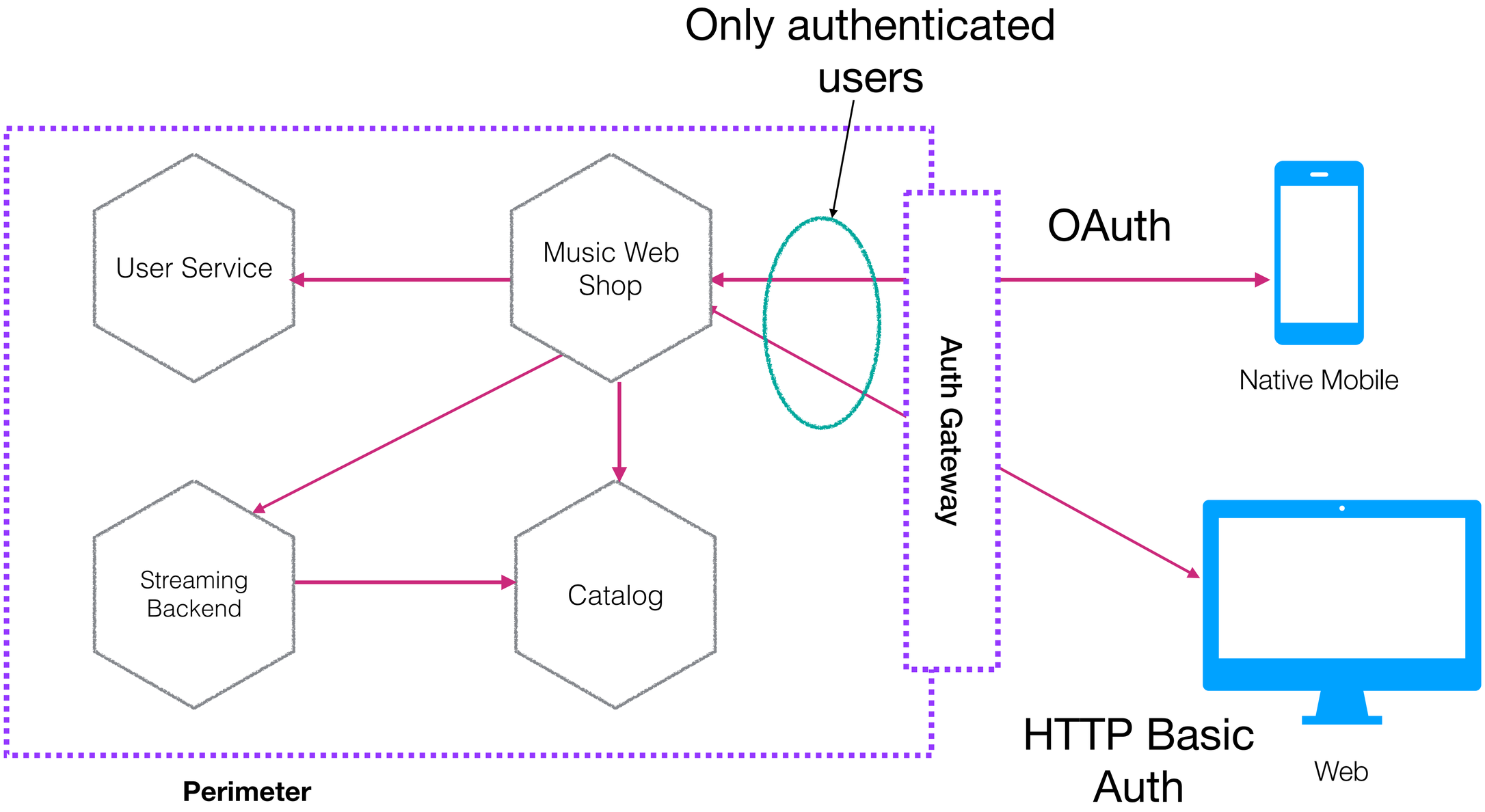
USER AUTHENTICATION - PROXY-BASED



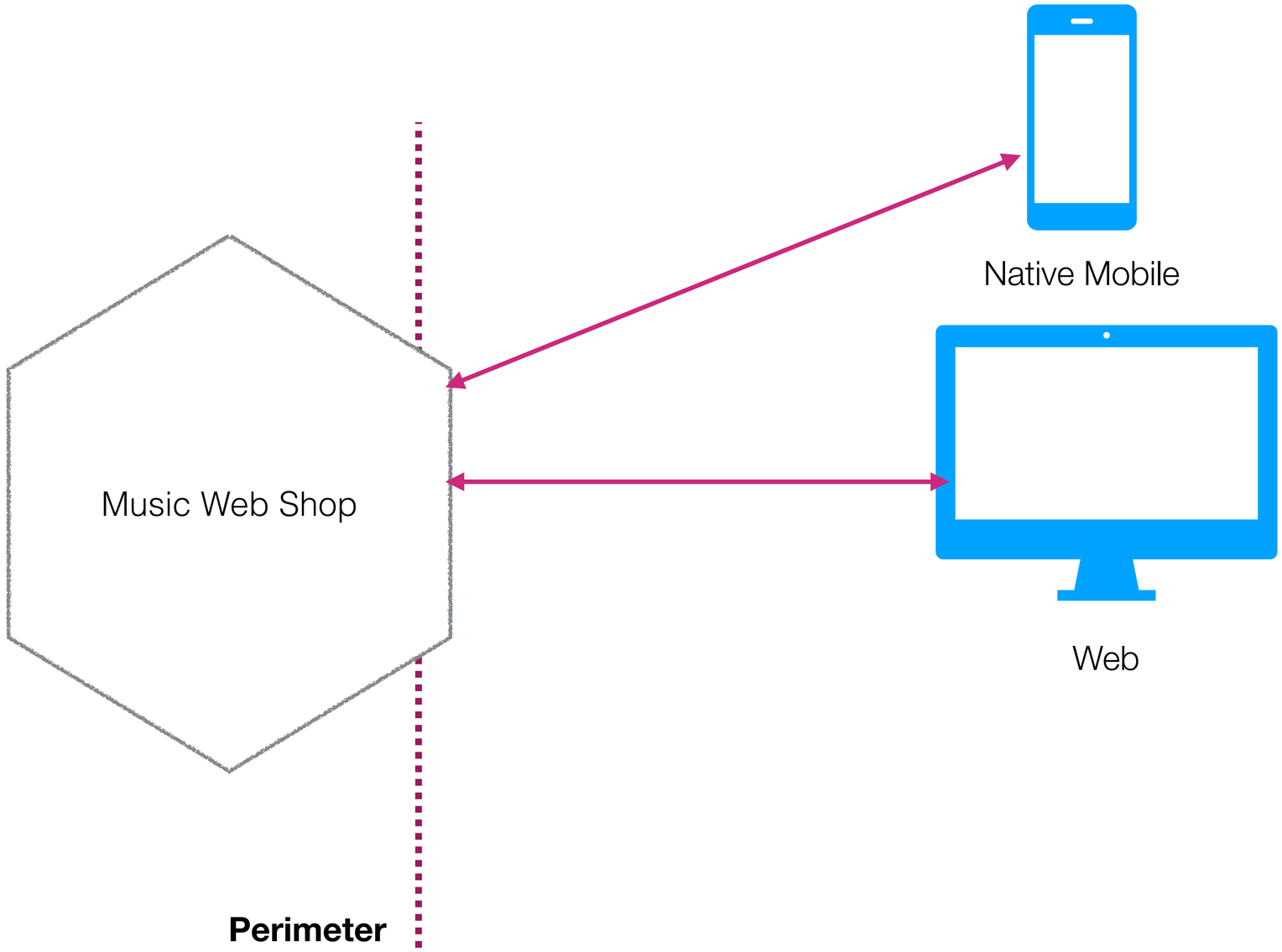
USER AUTHENTICATION - PROXY-BASED



USER AUTHENTICATION - PROXY-BASED

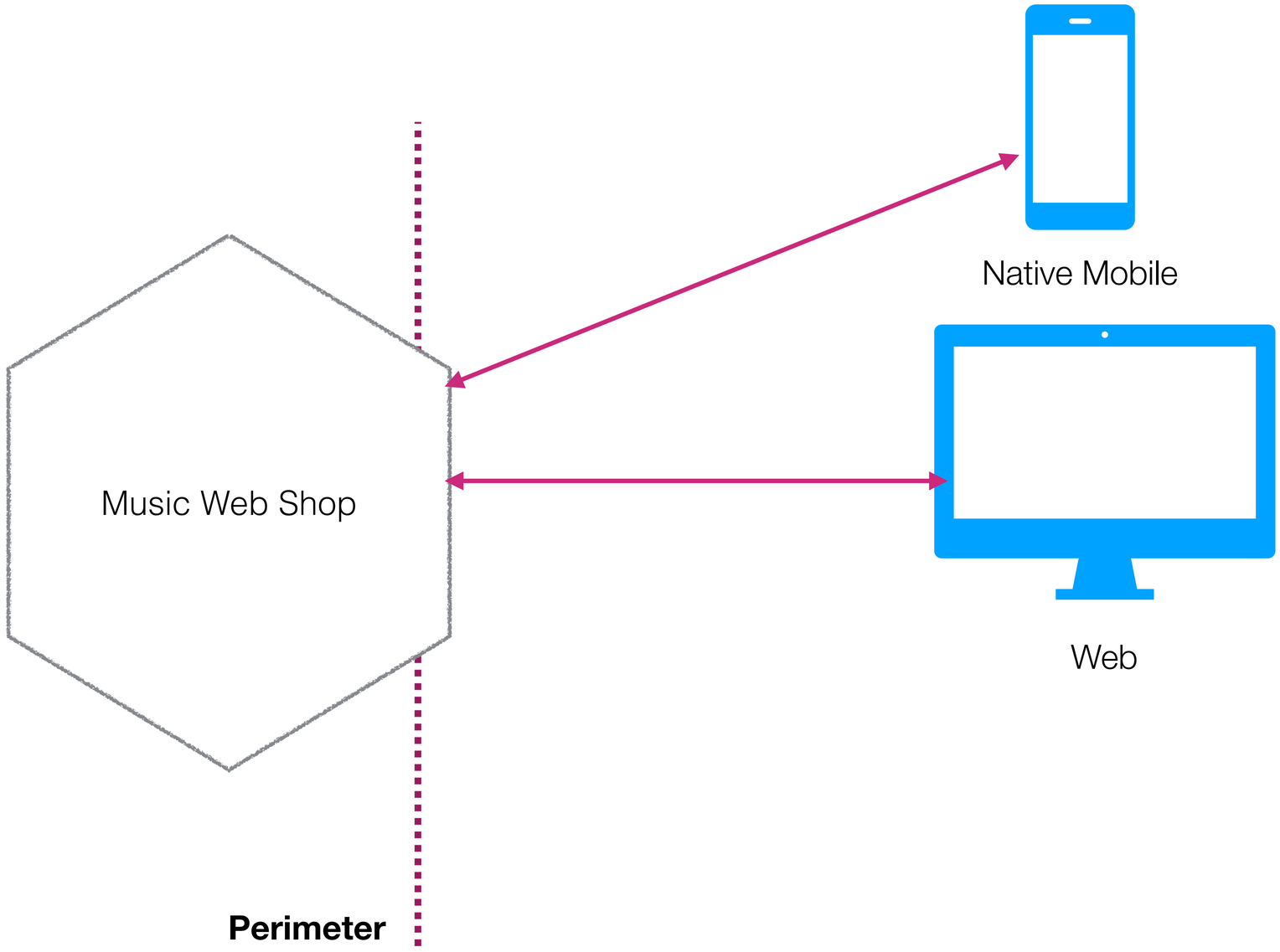


USER AUTHENTICATION - HANDLED IN SERVICE



USER AUTHENTICATION - HANDLED IN SERVICE

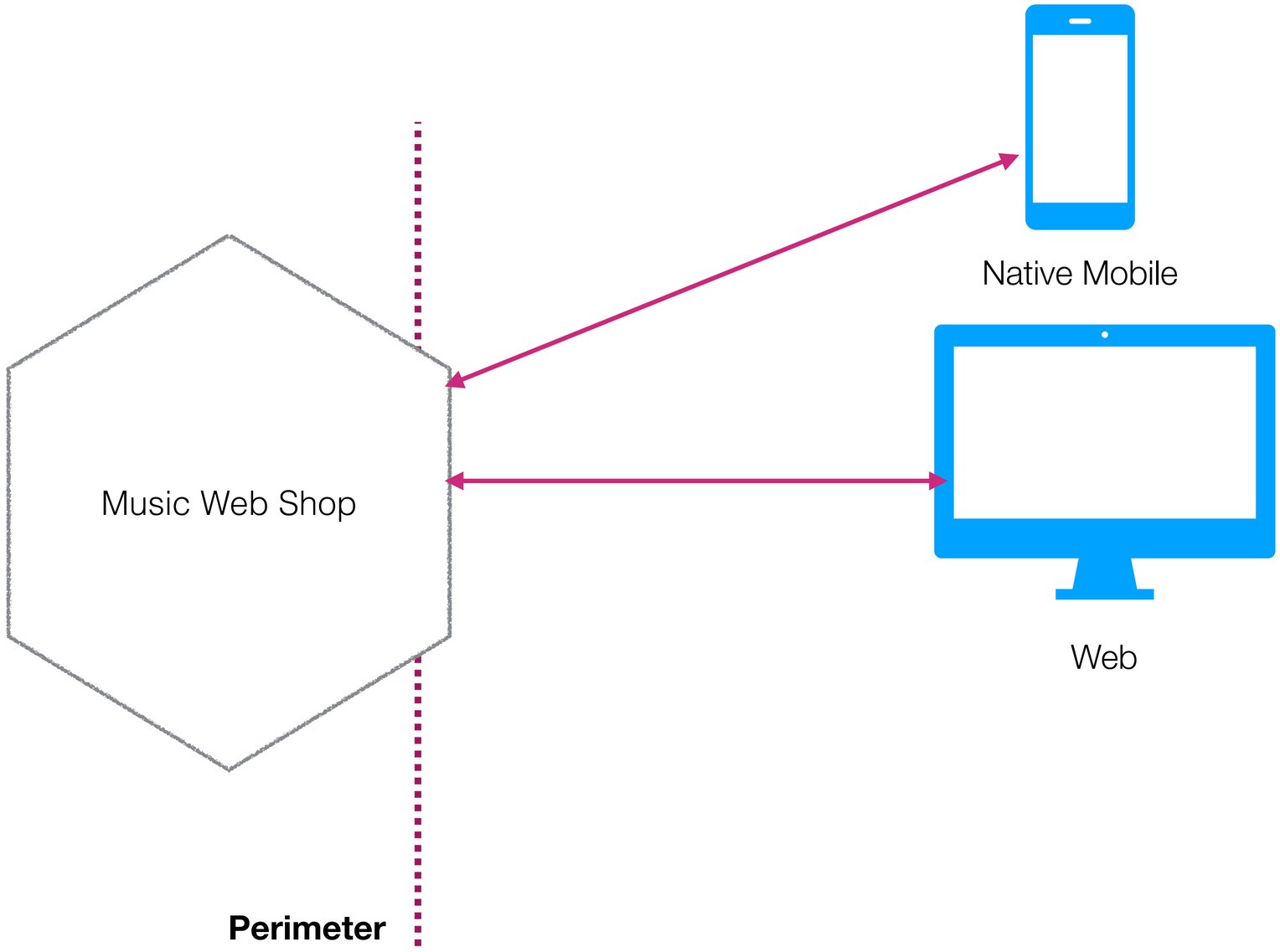
Can reduce latency



USER AUTHENTICATION - HANDLED IN SERVICE

Can reduce latency

Service potentially exposed to public internet

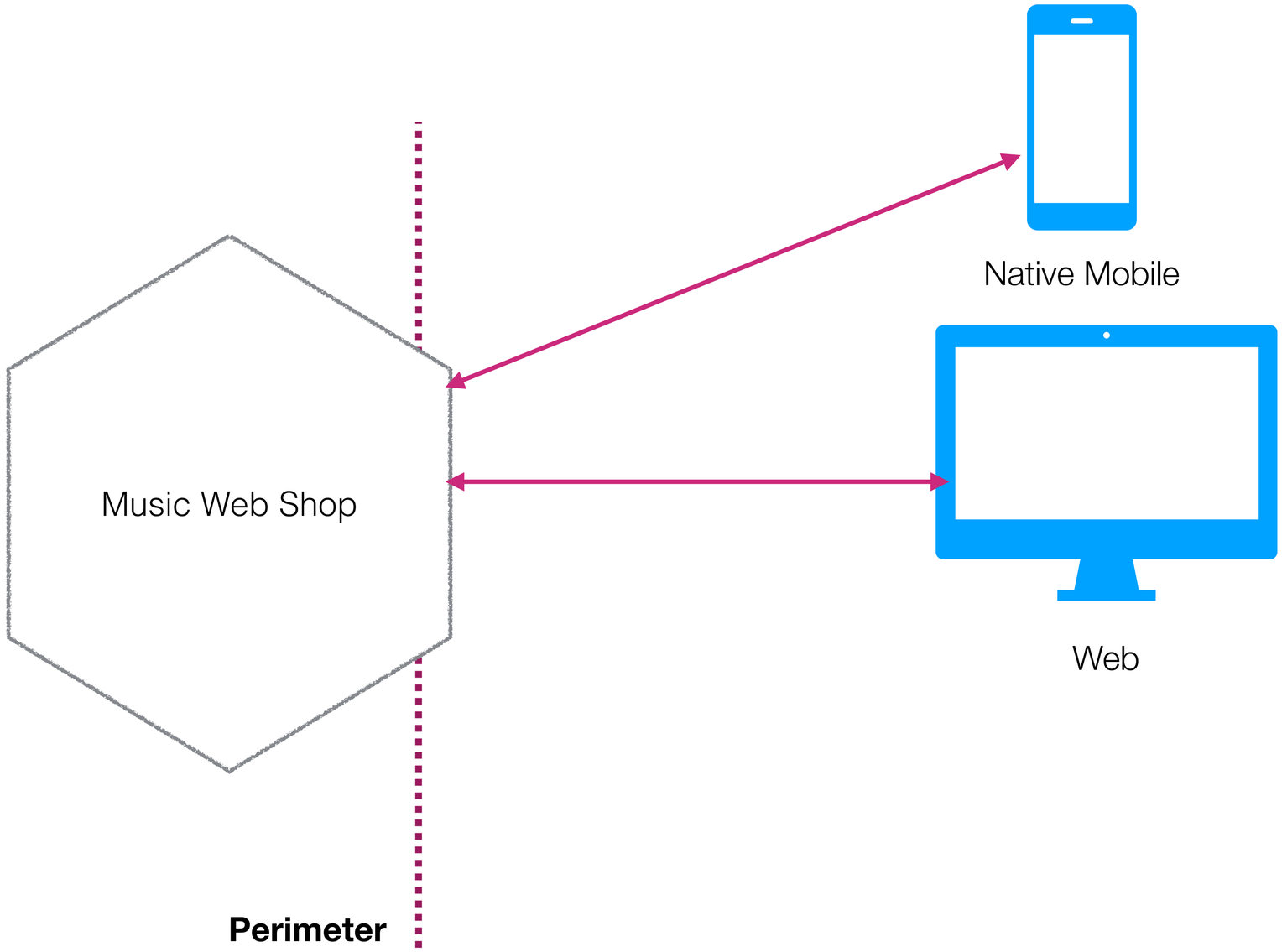


USER AUTHENTICATION - HANDLED IN SERVICE

Can reduce latency

Service potentially exposed to public internet

Self-contained



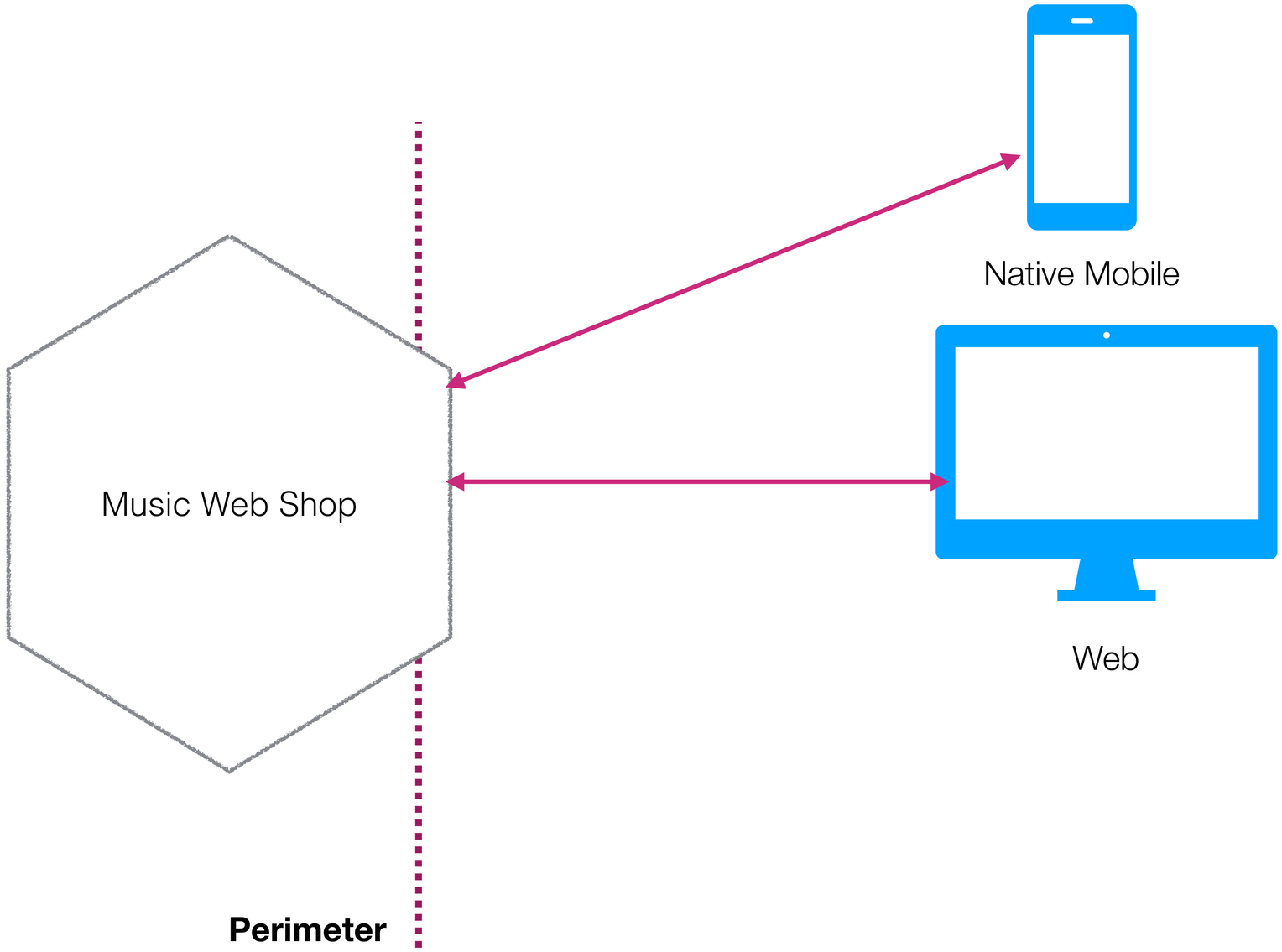
USER AUTHENTICATION - HANDLED IN SERVICE

Can reduce latency

Service potentially exposed to public internet

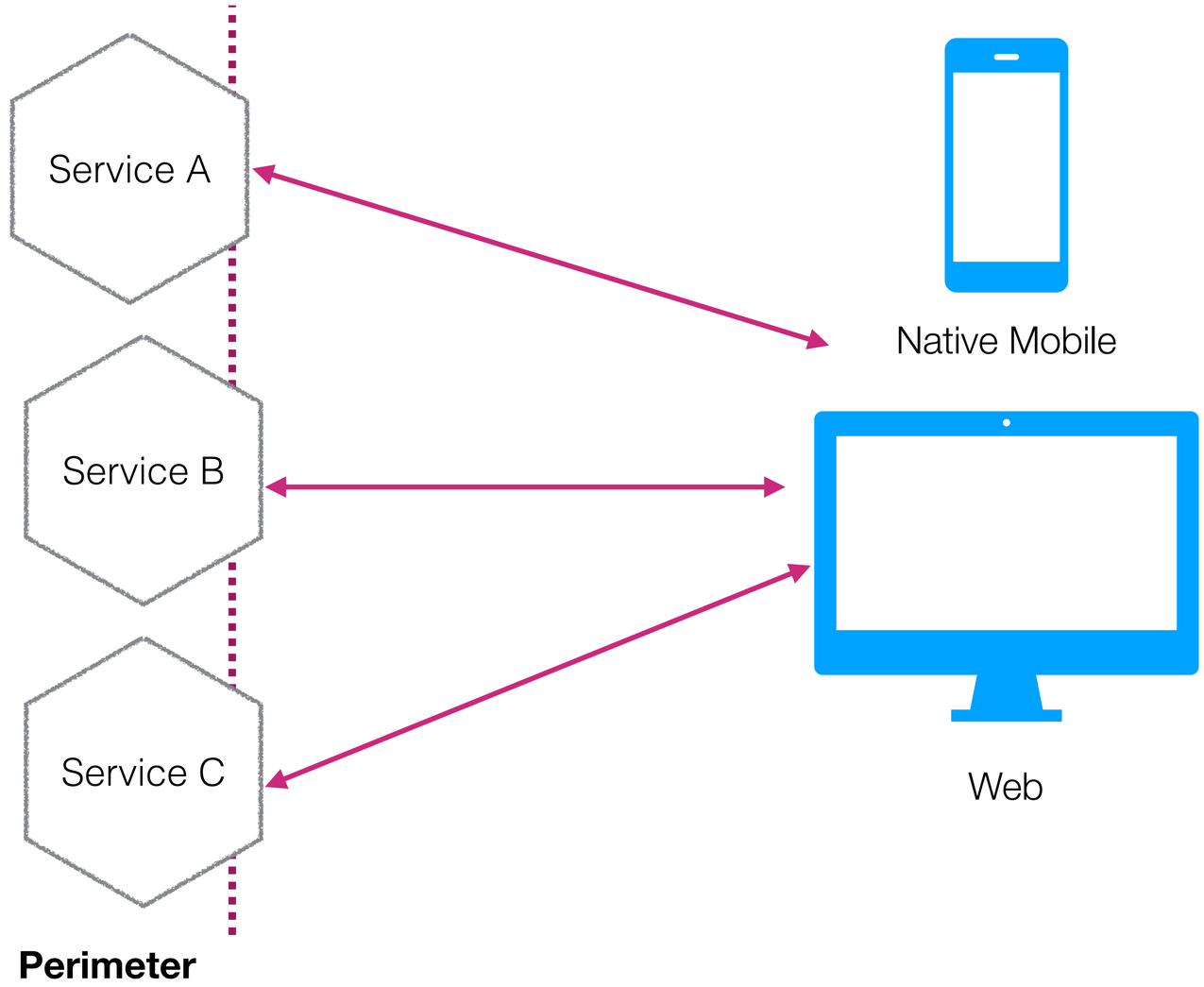
Self-contained

Code reuse?



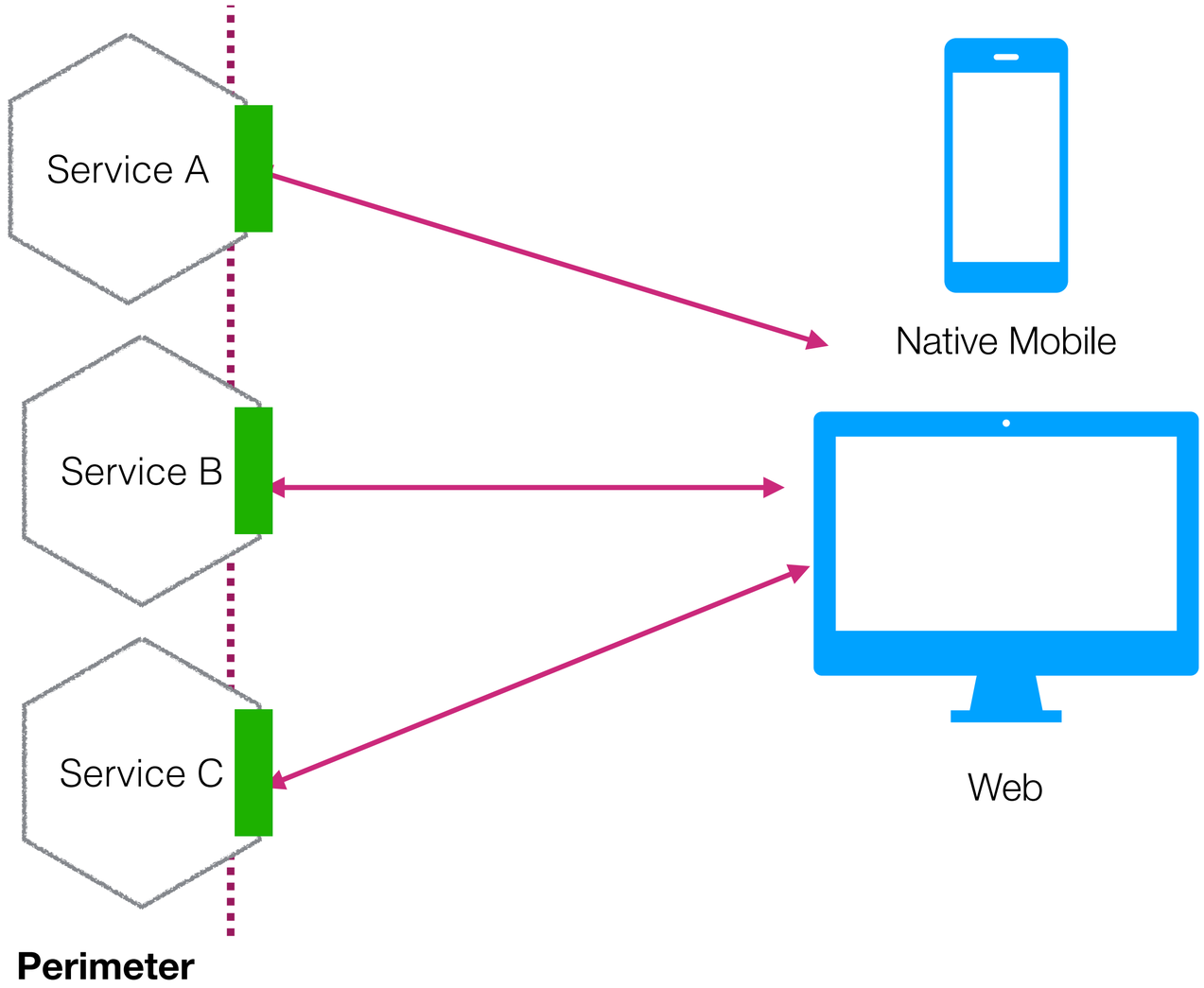
USER AUTHENTICATION - LIBRARY-BASED

Re-use authentication flow code via library



USER AUTHENTICATION - LIBRARY-BASED

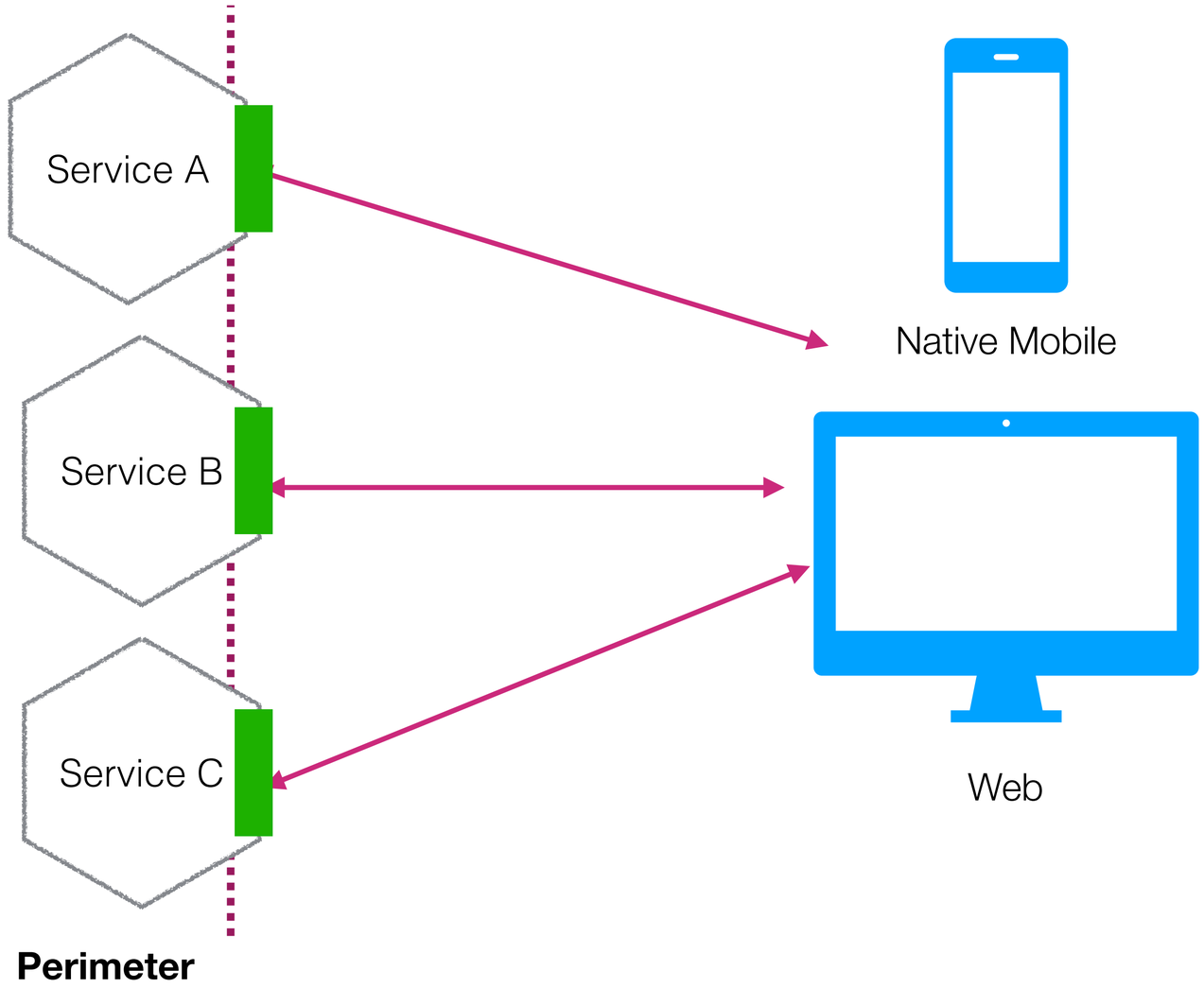
Re-use authentication flow code via library



USER AUTHENTICATION - LIBRARY-BASED

Re-use authentication flow code via library

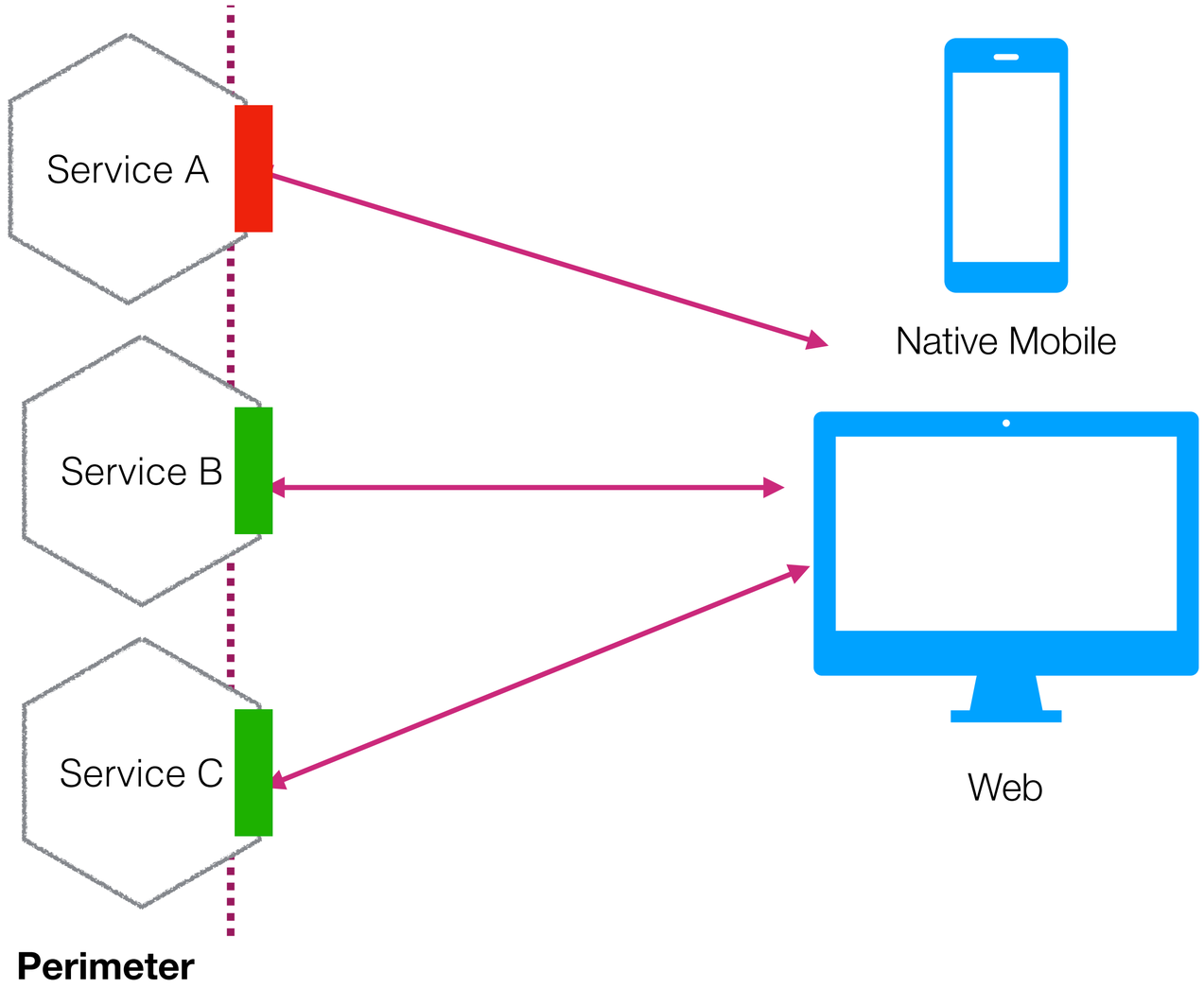
Version drift?



USER AUTHENTICATION - LIBRARY-BASED

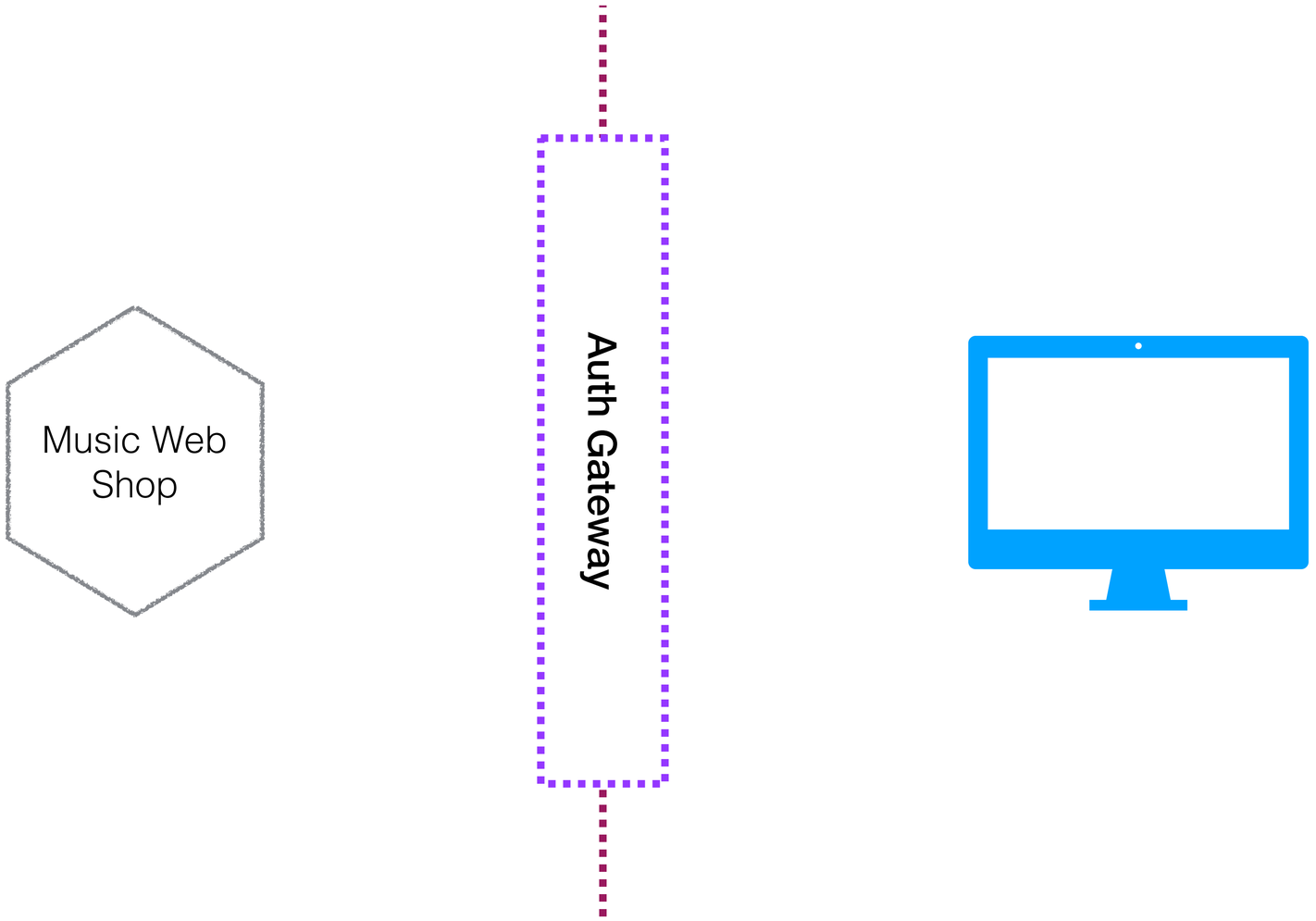
Re-use authentication flow code via library

Version drift?

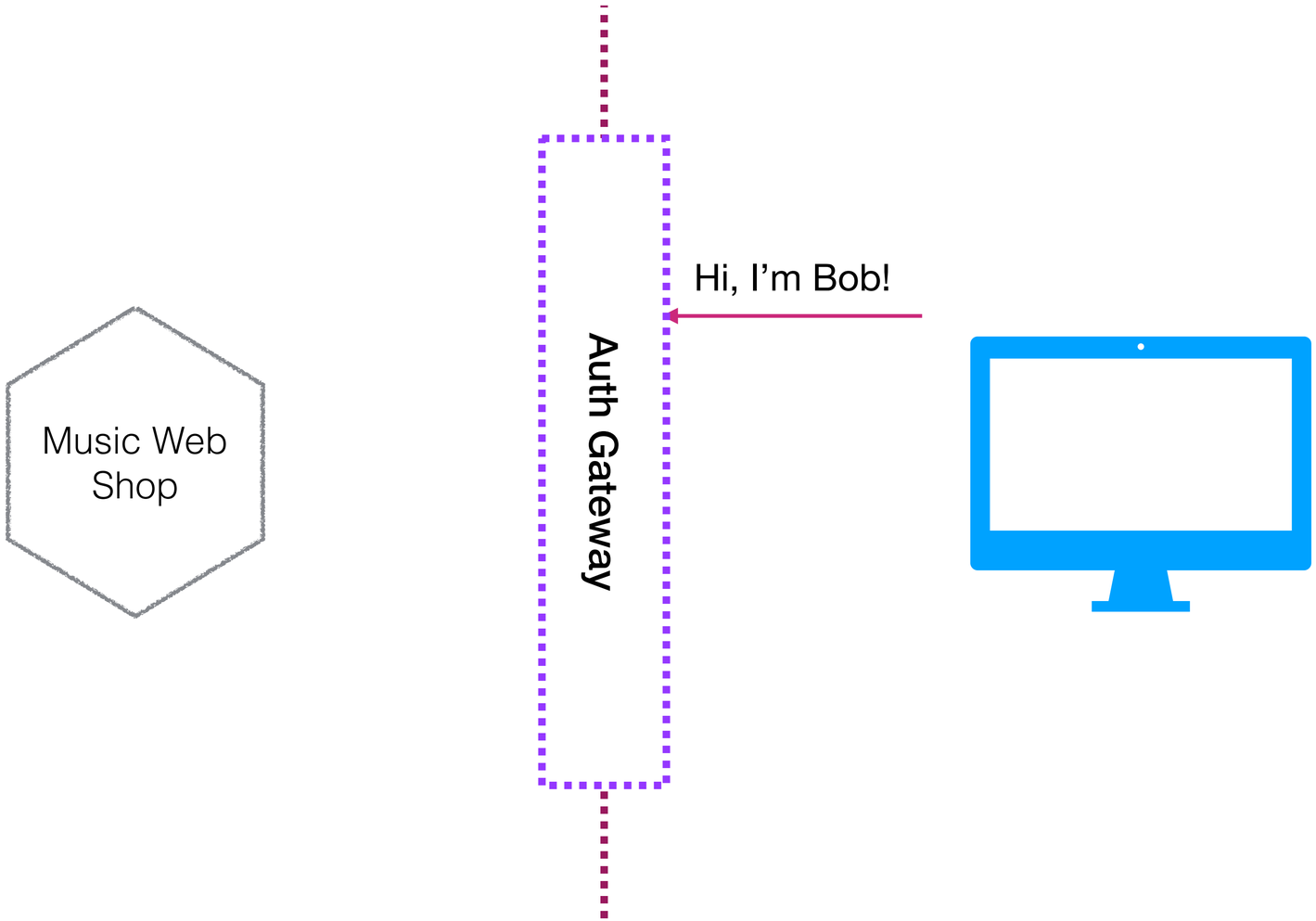


What about authorisation?

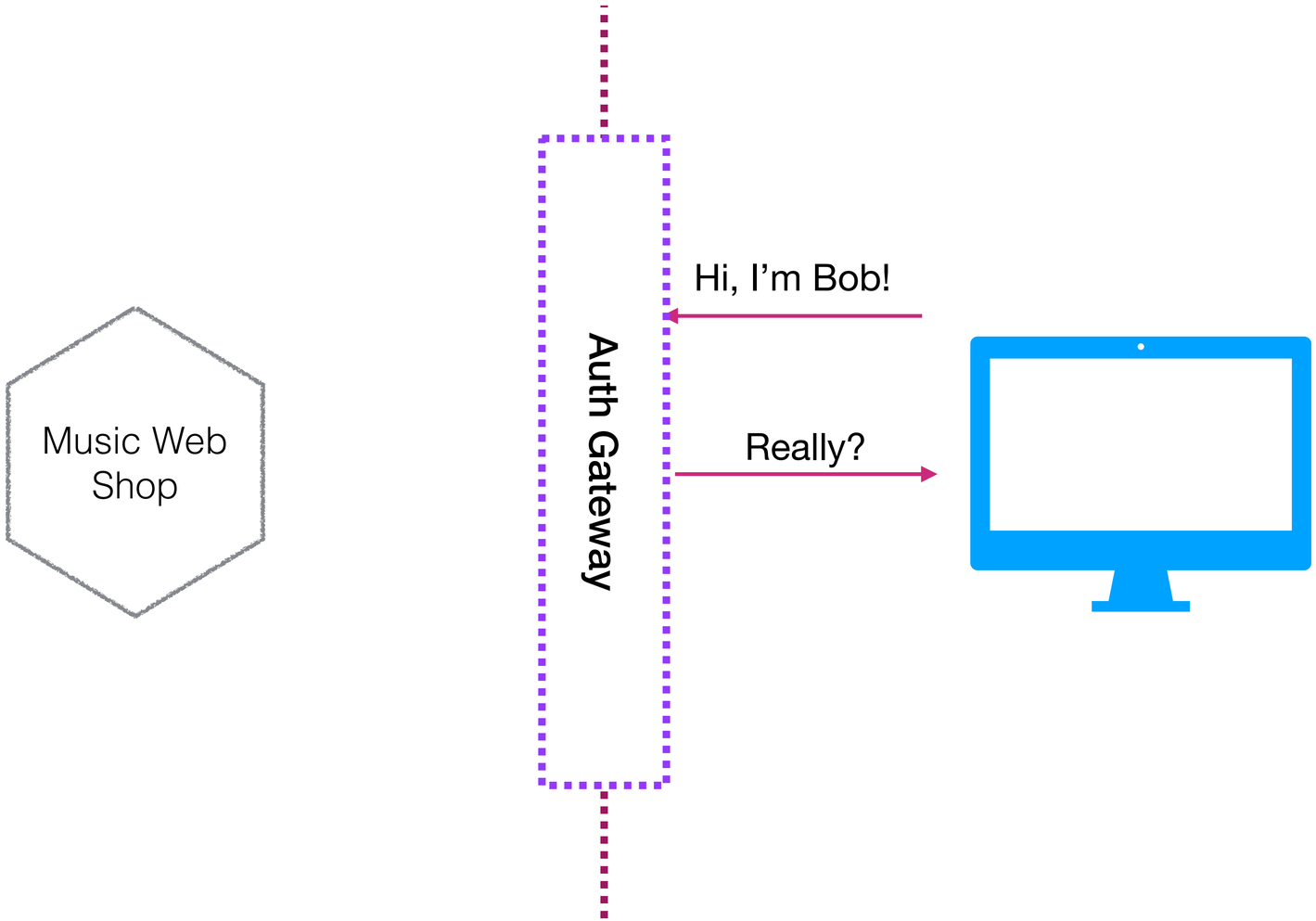
DO YOU EVEN AUTH?



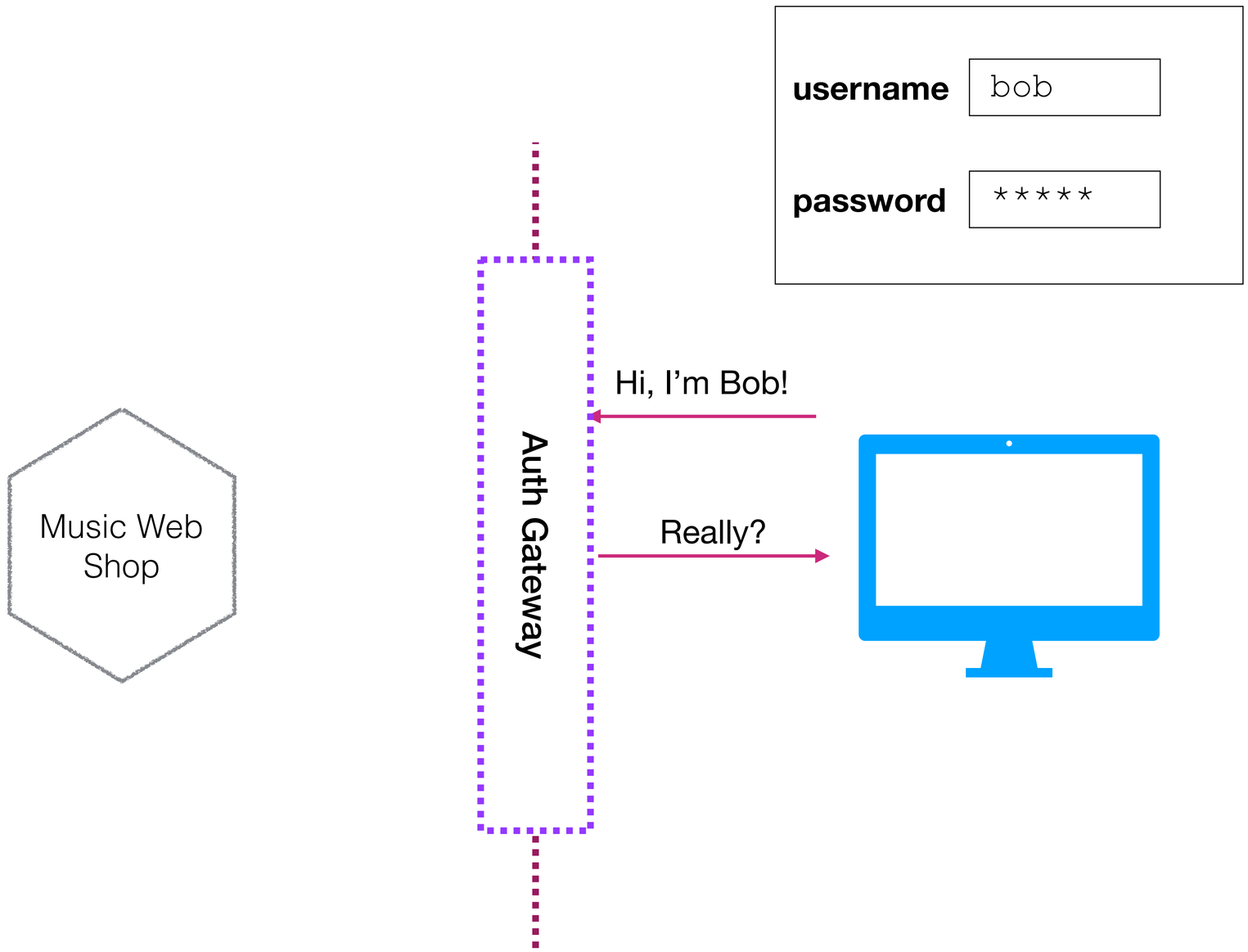
DO YOU EVEN AUTH?



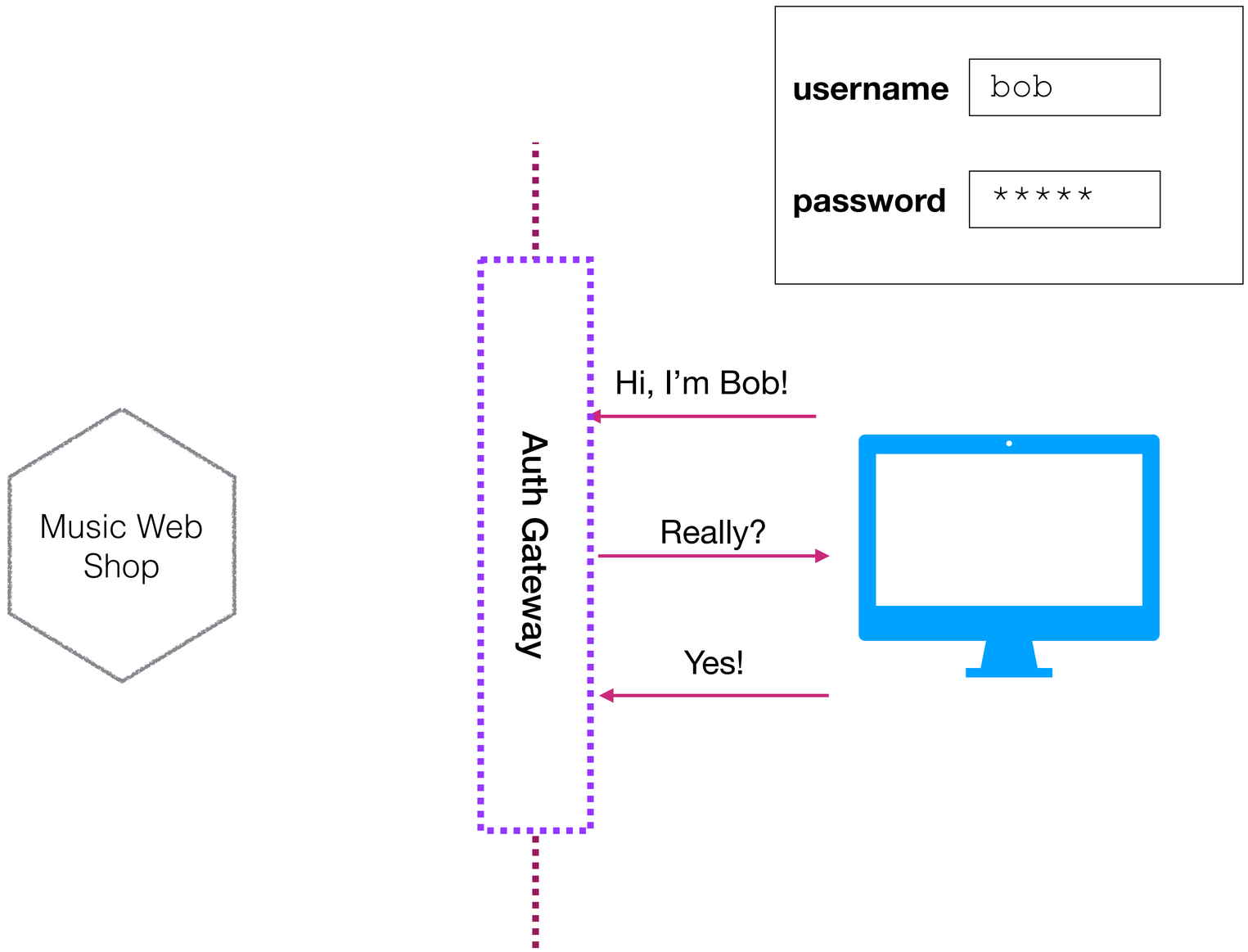
DO YOU EVEN AUTH?



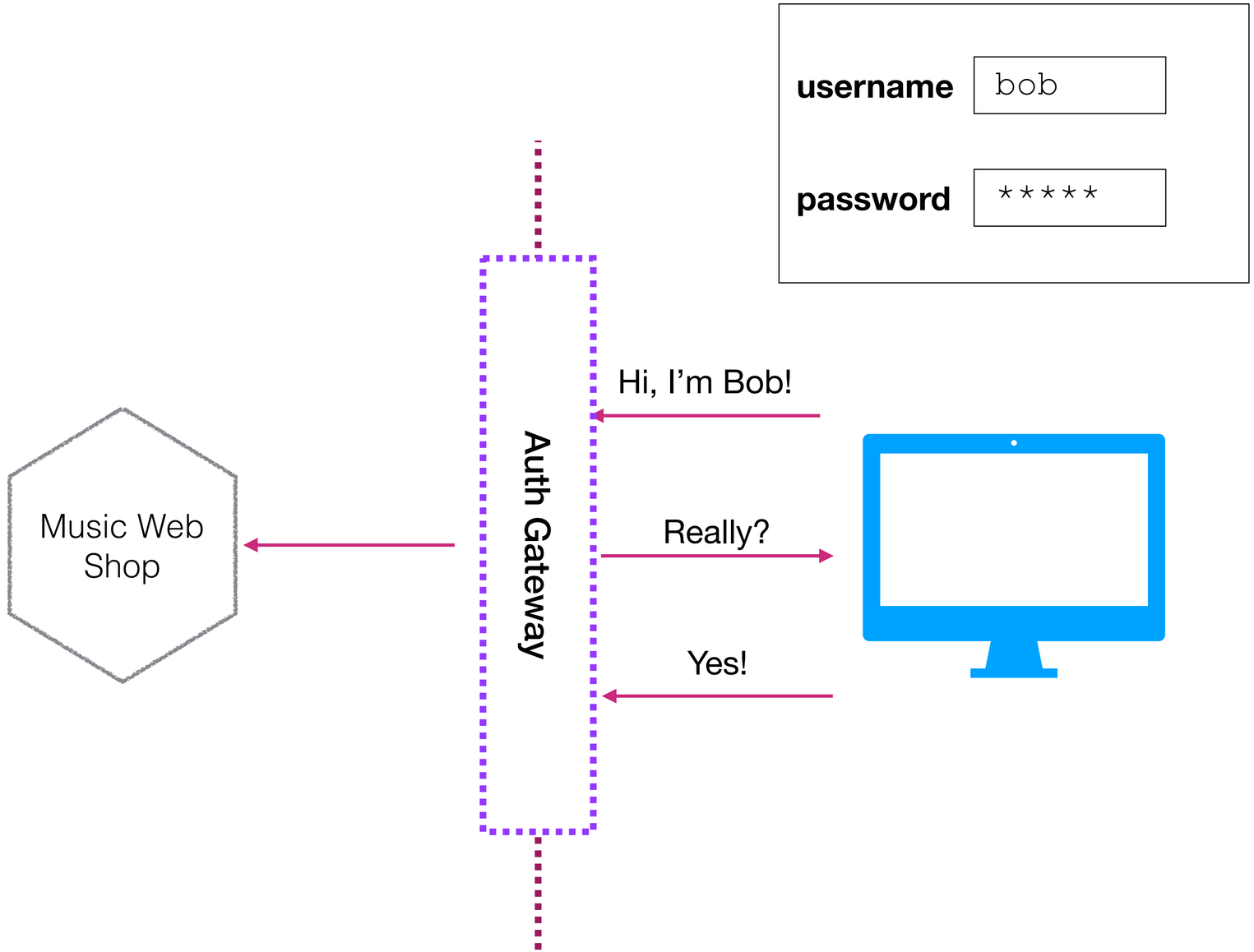
DO YOU EVEN AUTH?



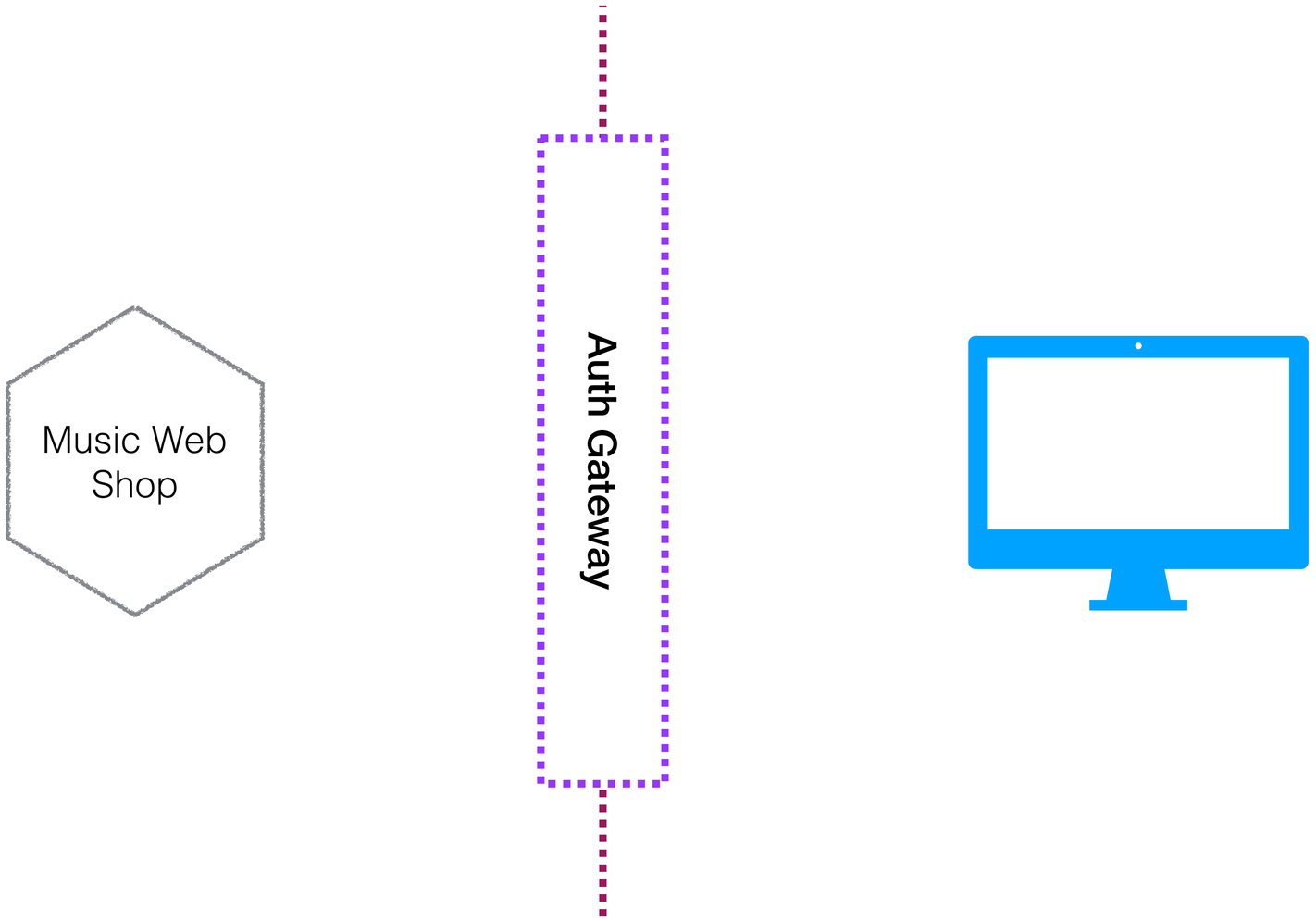
DO YOU EVEN AUTH?



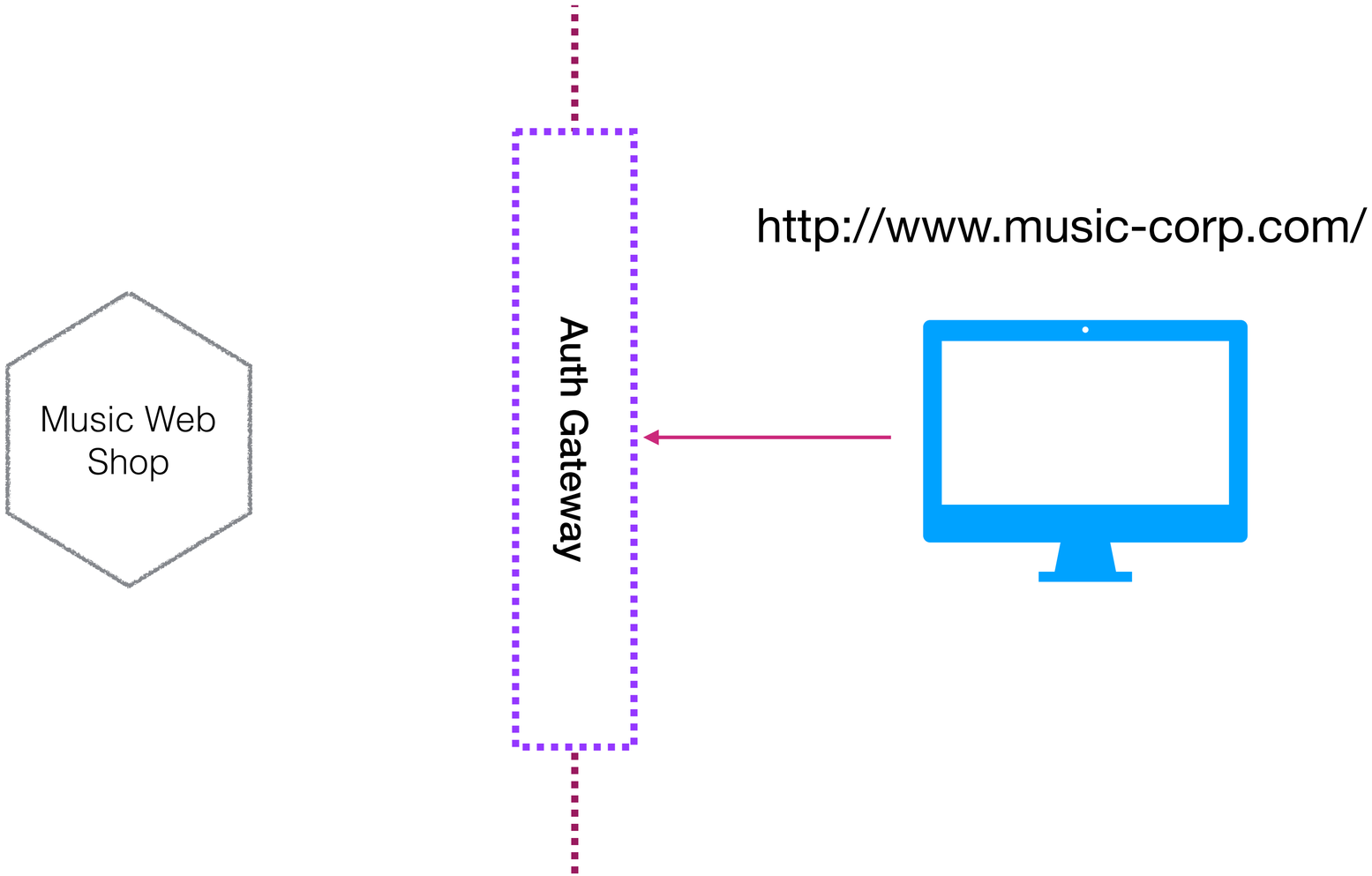
DO YOU EVEN AUTH?



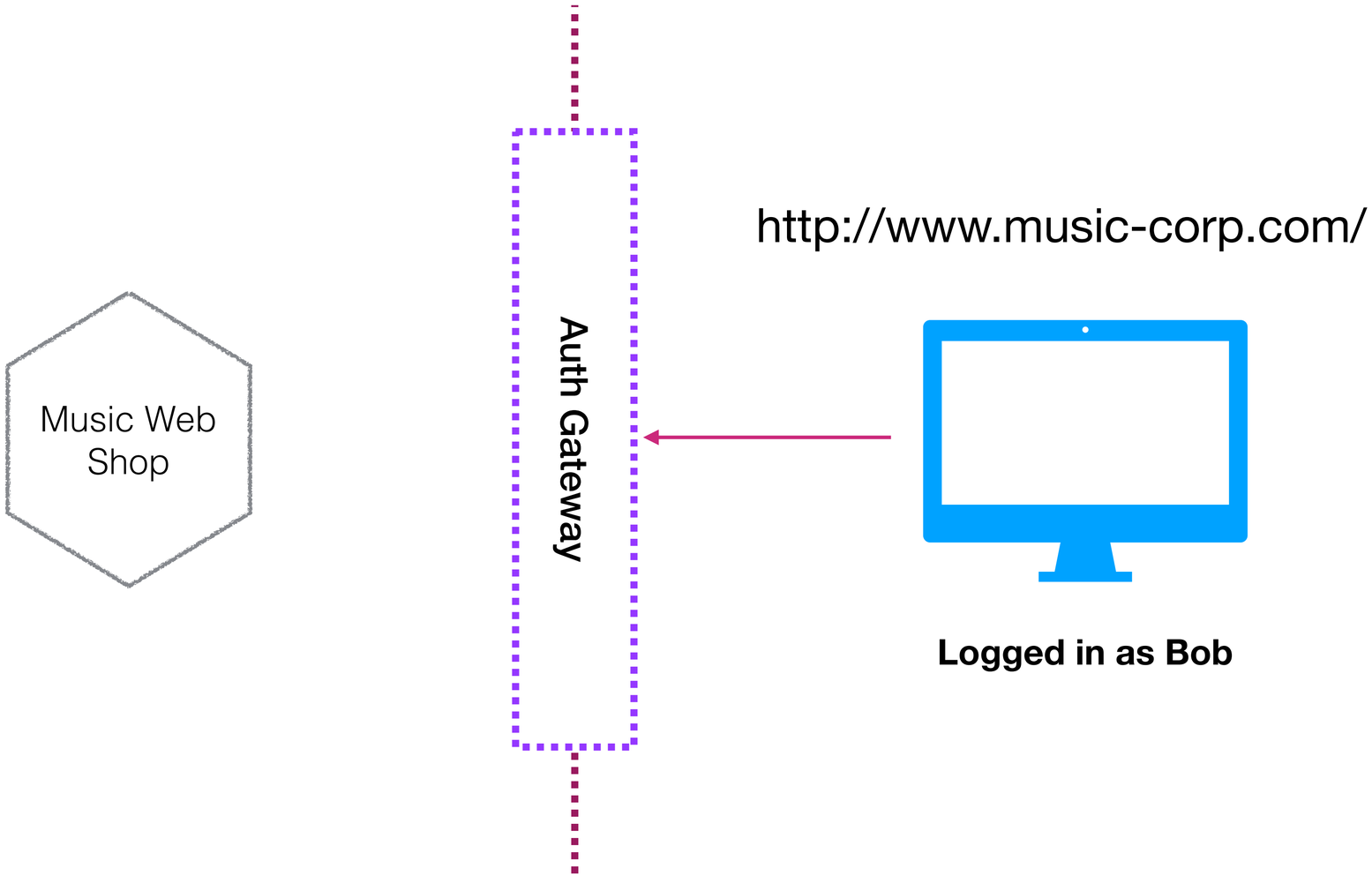
DO YOU EVEN AUTH?



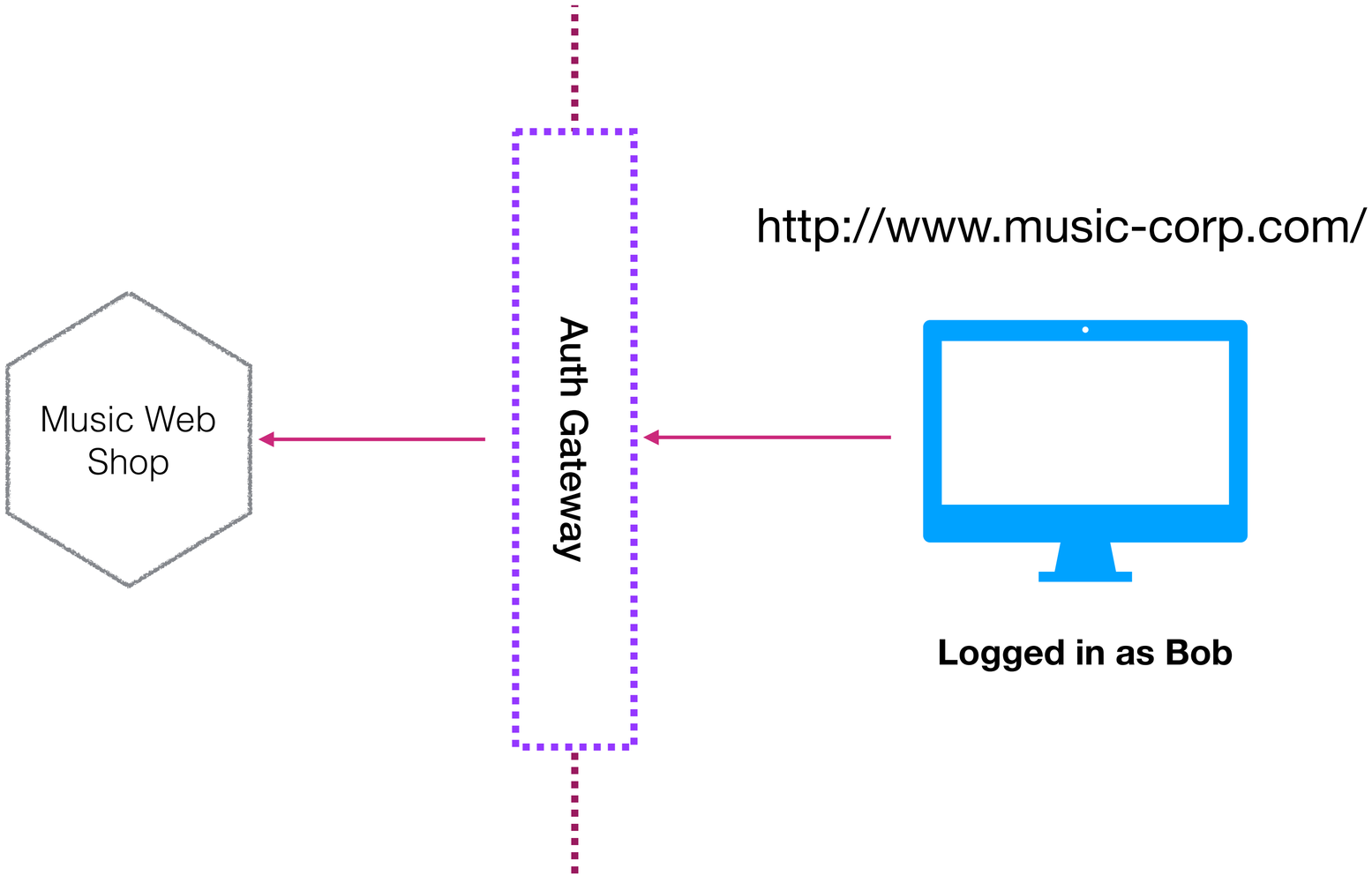
DO YOU EVEN AUTH?



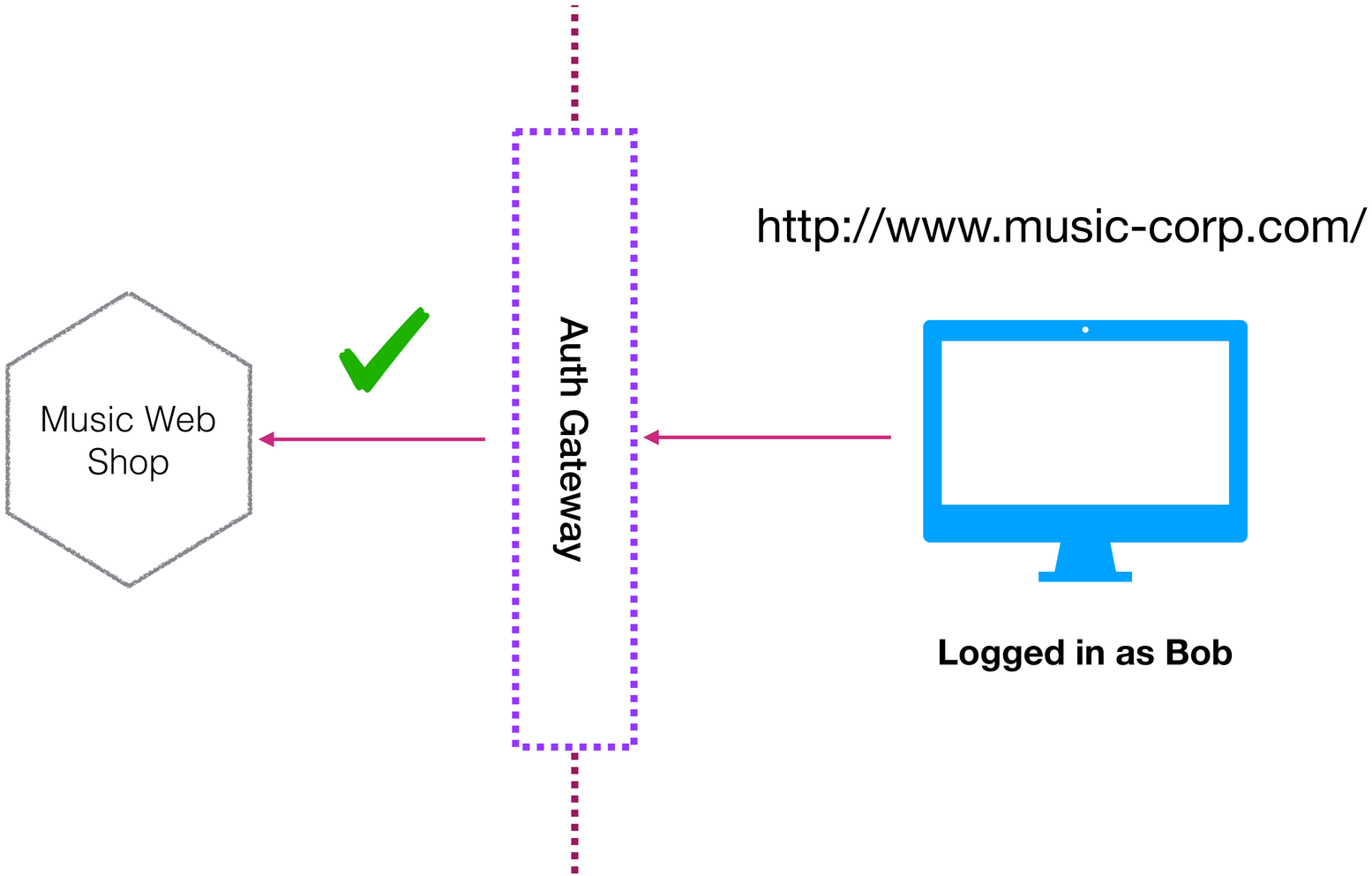
DO YOU EVEN AUTH?



DO YOU EVEN AUTH?

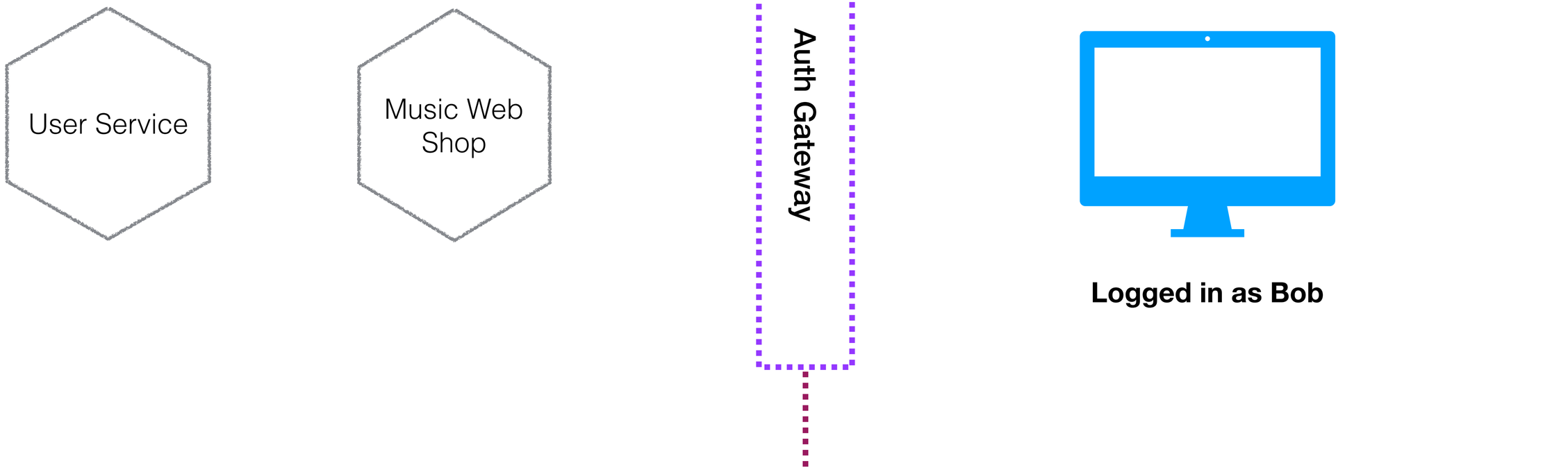


DO YOU EVEN AUTH?

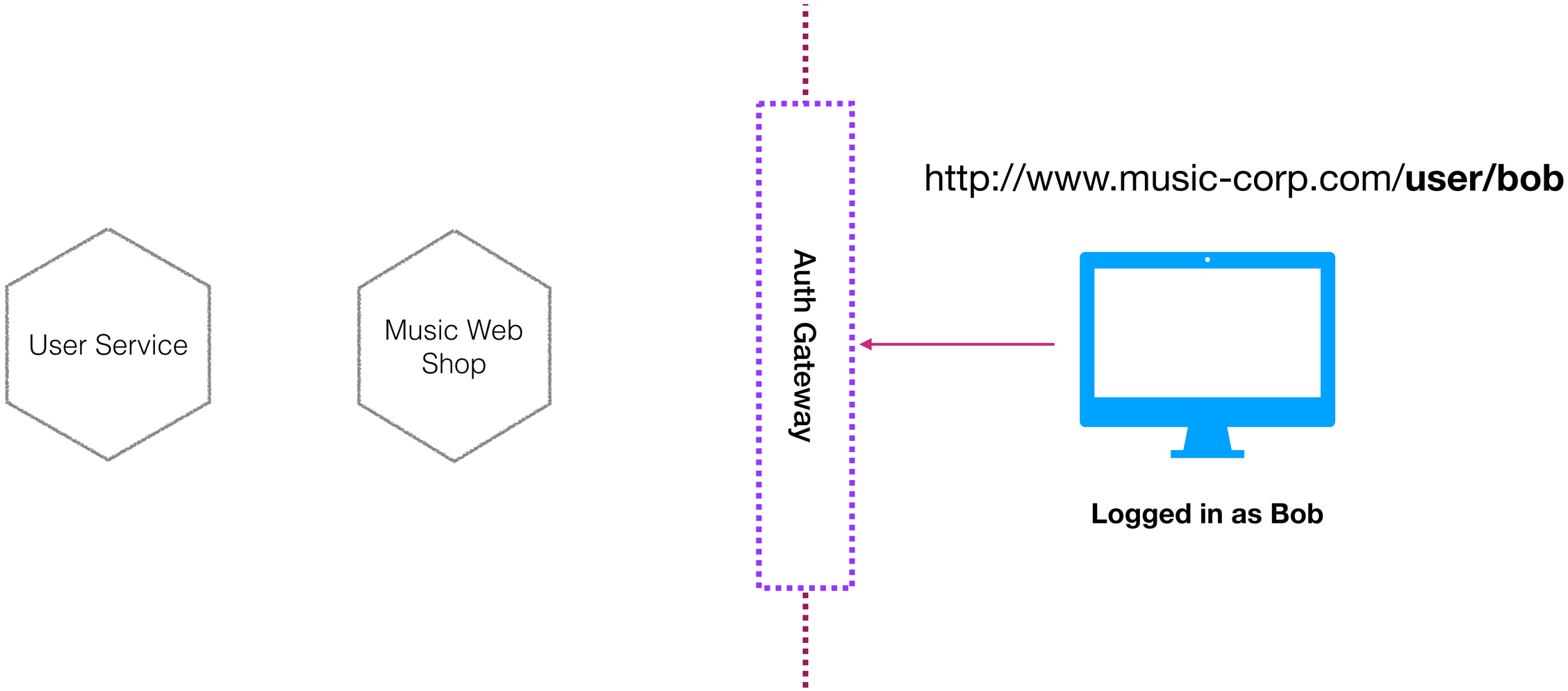


DOWNSTREAM AUTH - IMPLICIT TRUST?

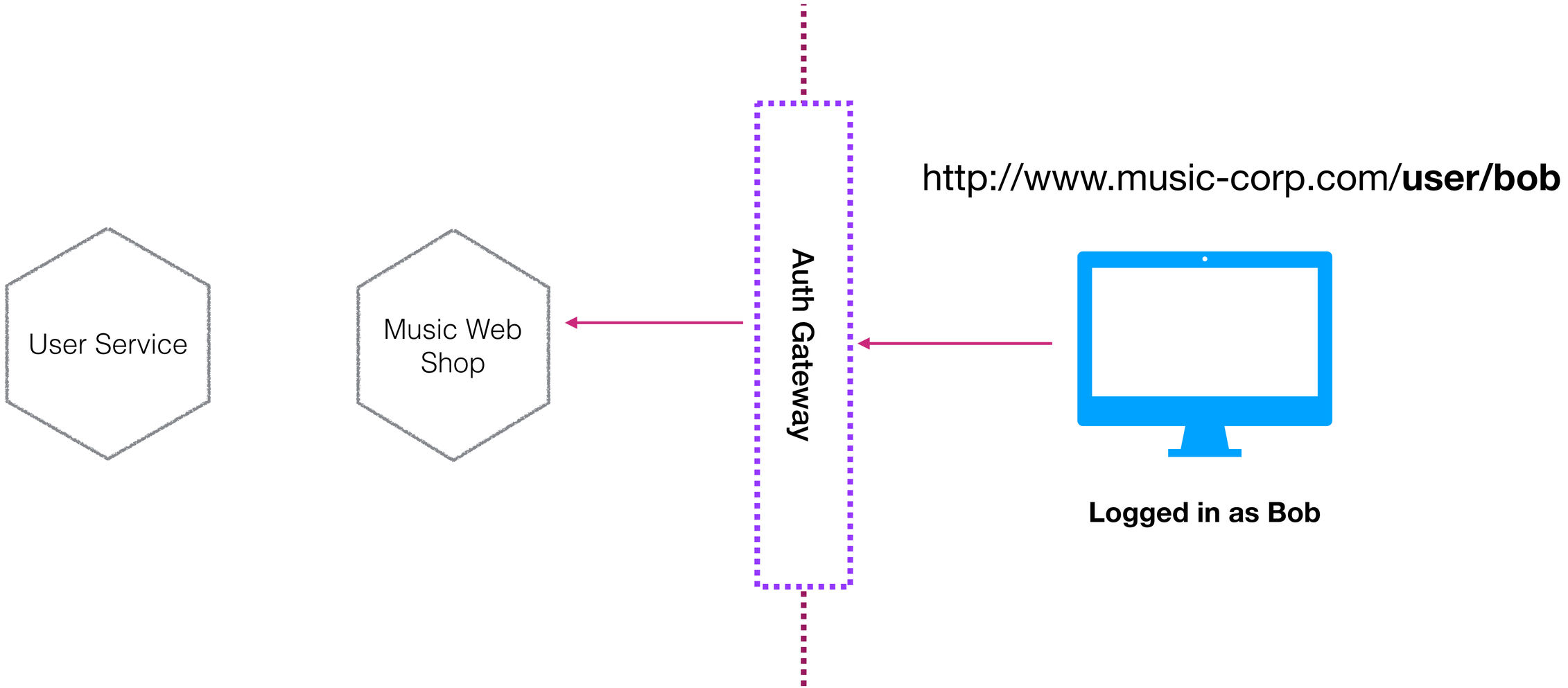
DOWNSTREAM AUTH - IMPLICIT TRUST?



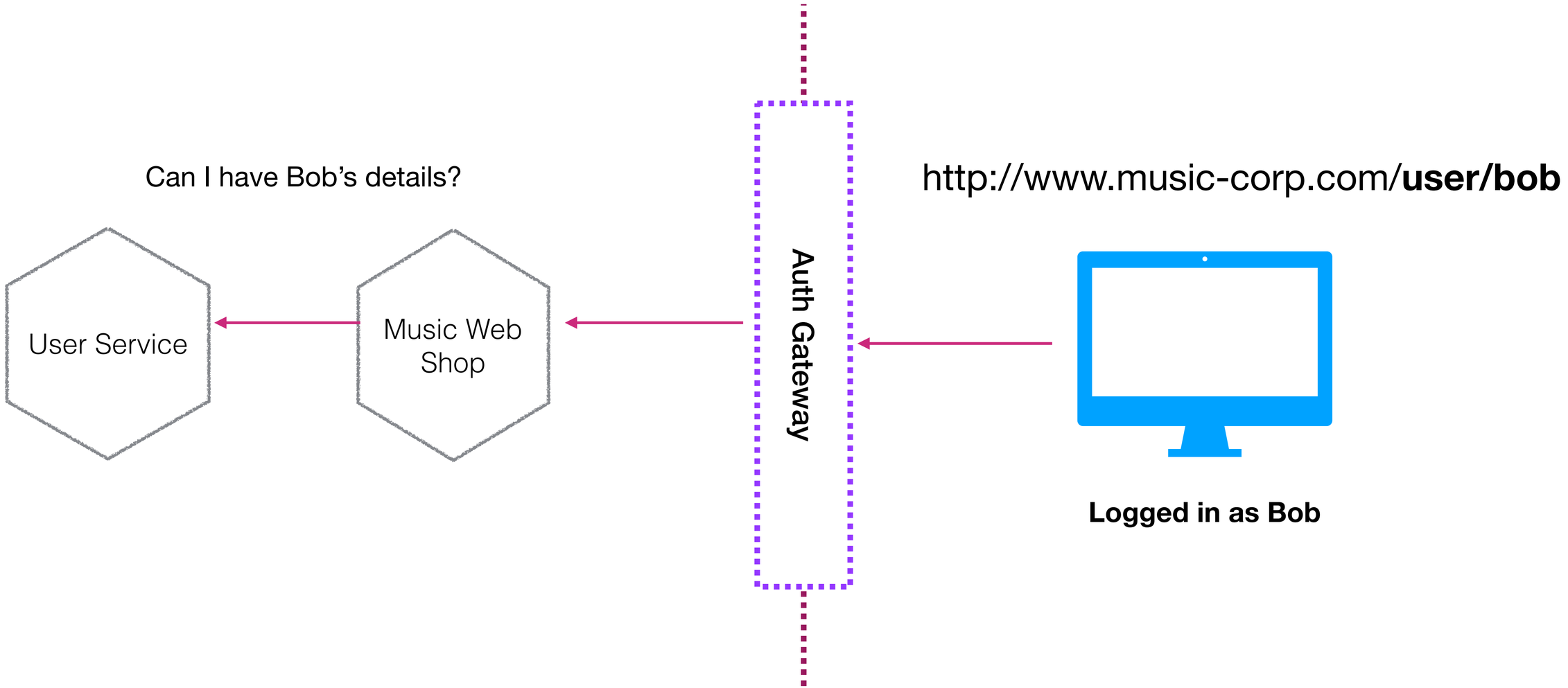
DOWNSTREAM AUTH - IMPLICIT TRUST?



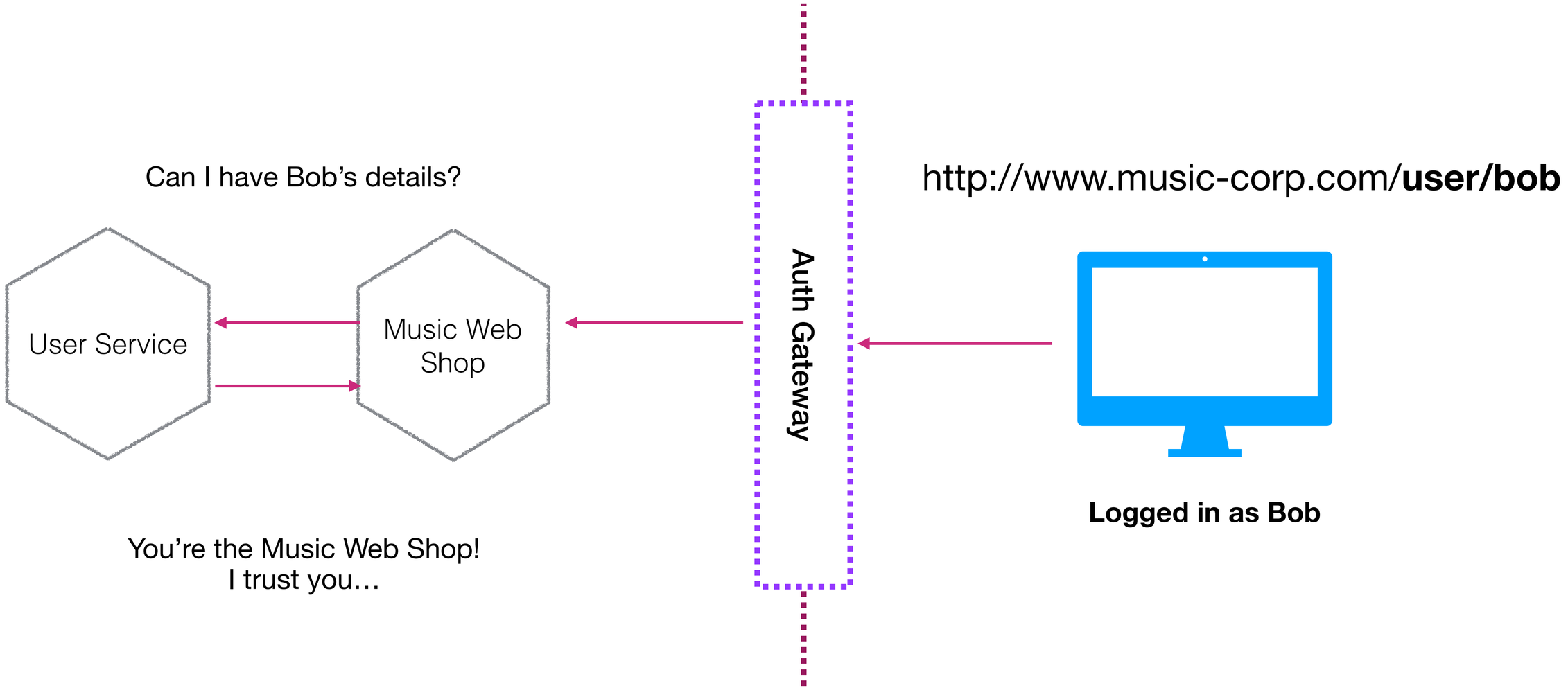
DOWNSTREAM AUTH - IMPLICIT TRUST?



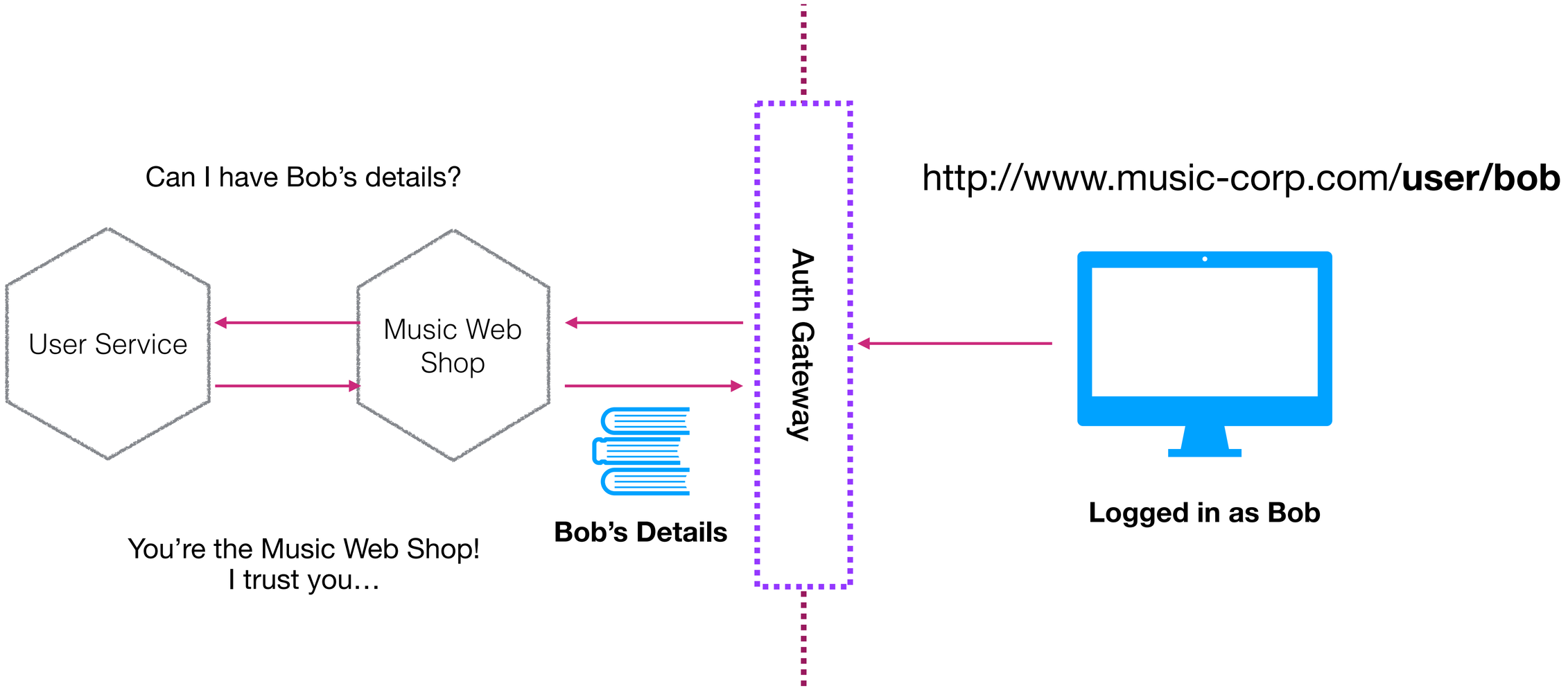
DOWNSTREAM AUTH - IMPLICIT TRUST?



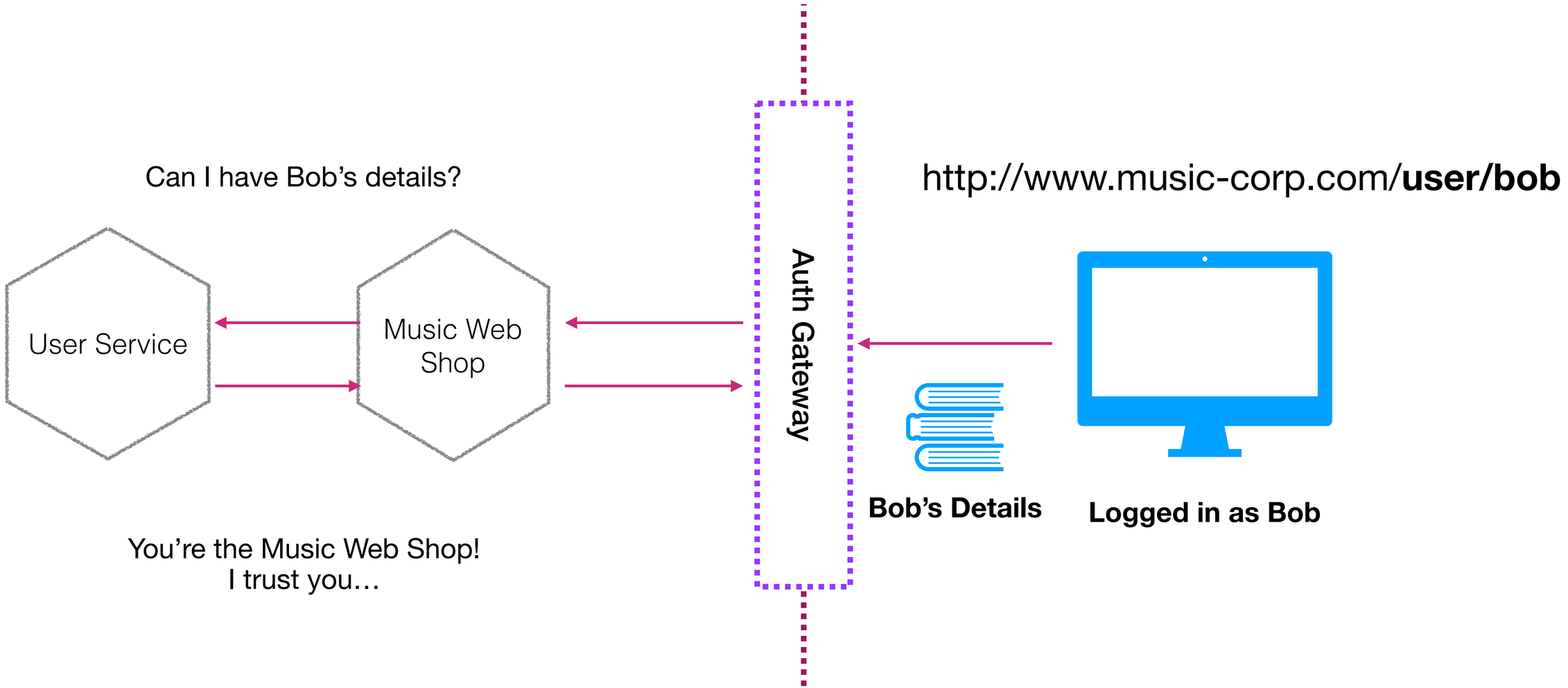
DOWNSTREAM AUTH - IMPLICIT TRUST?



DOWNSTREAM AUTH - IMPLICIT TRUST?



DOWNSTREAM AUTH - IMPLICIT TRUST?



DOWNSTREAM AUTH - IMPLICIT TRUST?

DOWNSTREAM AUTH - IMPLICIT TRUST?

User Service

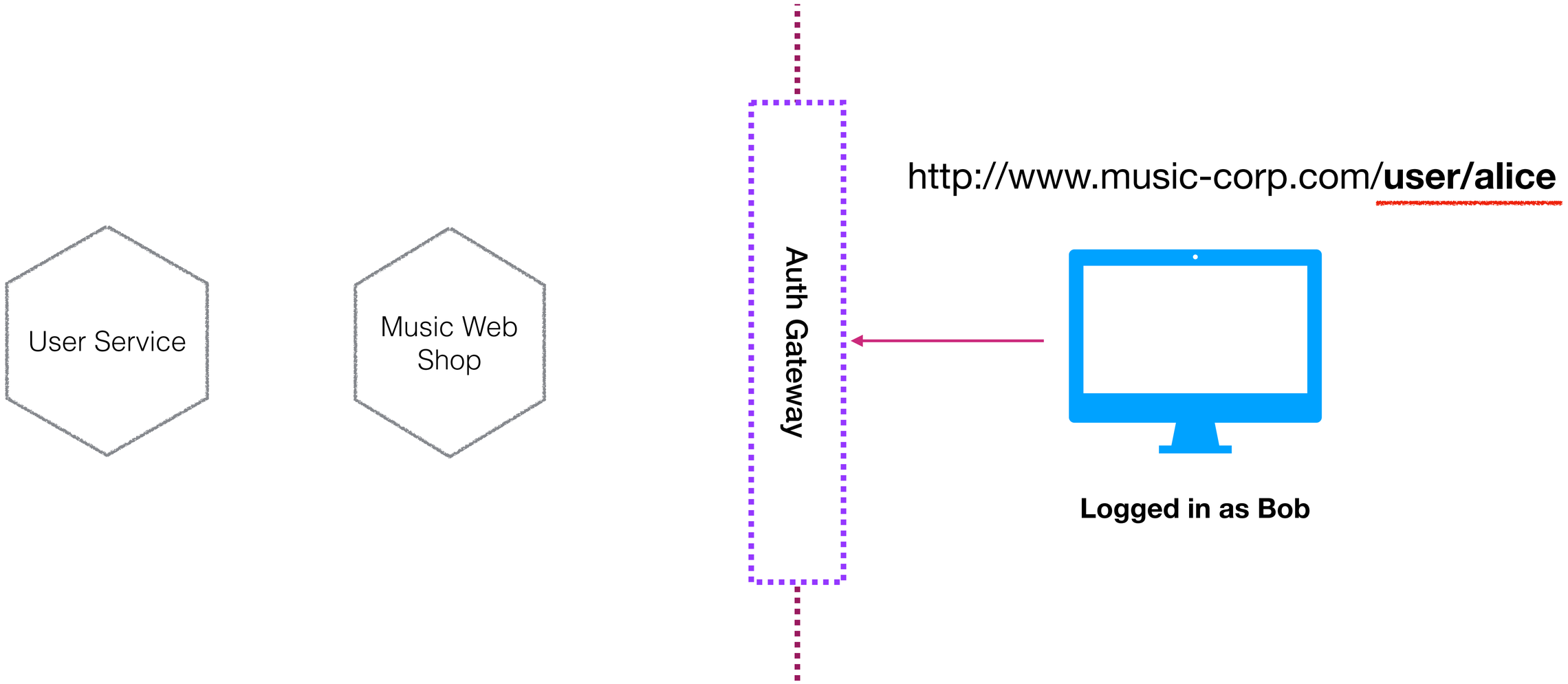
Music Web Shop

Auth Gateway

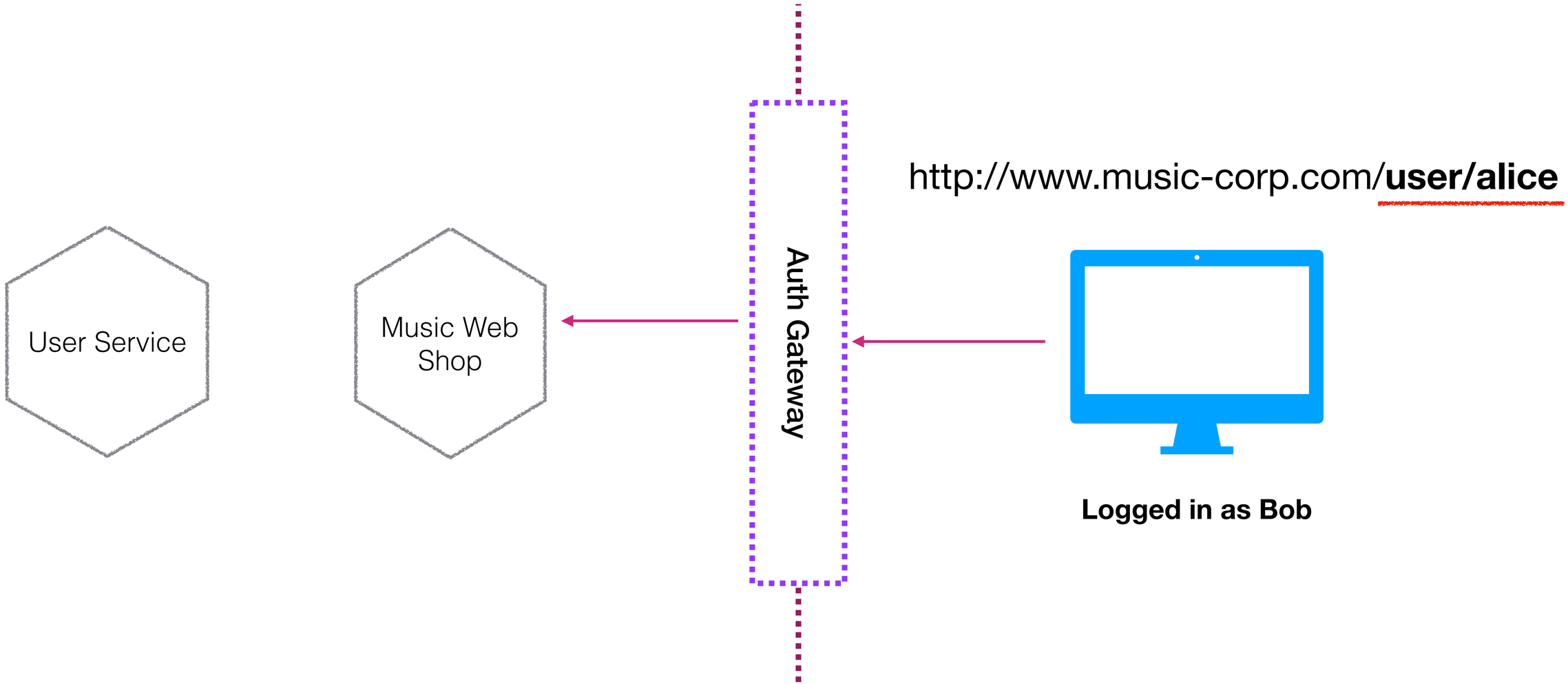
http://www.music-corp.com/**user/alice**

Logged in as Bob

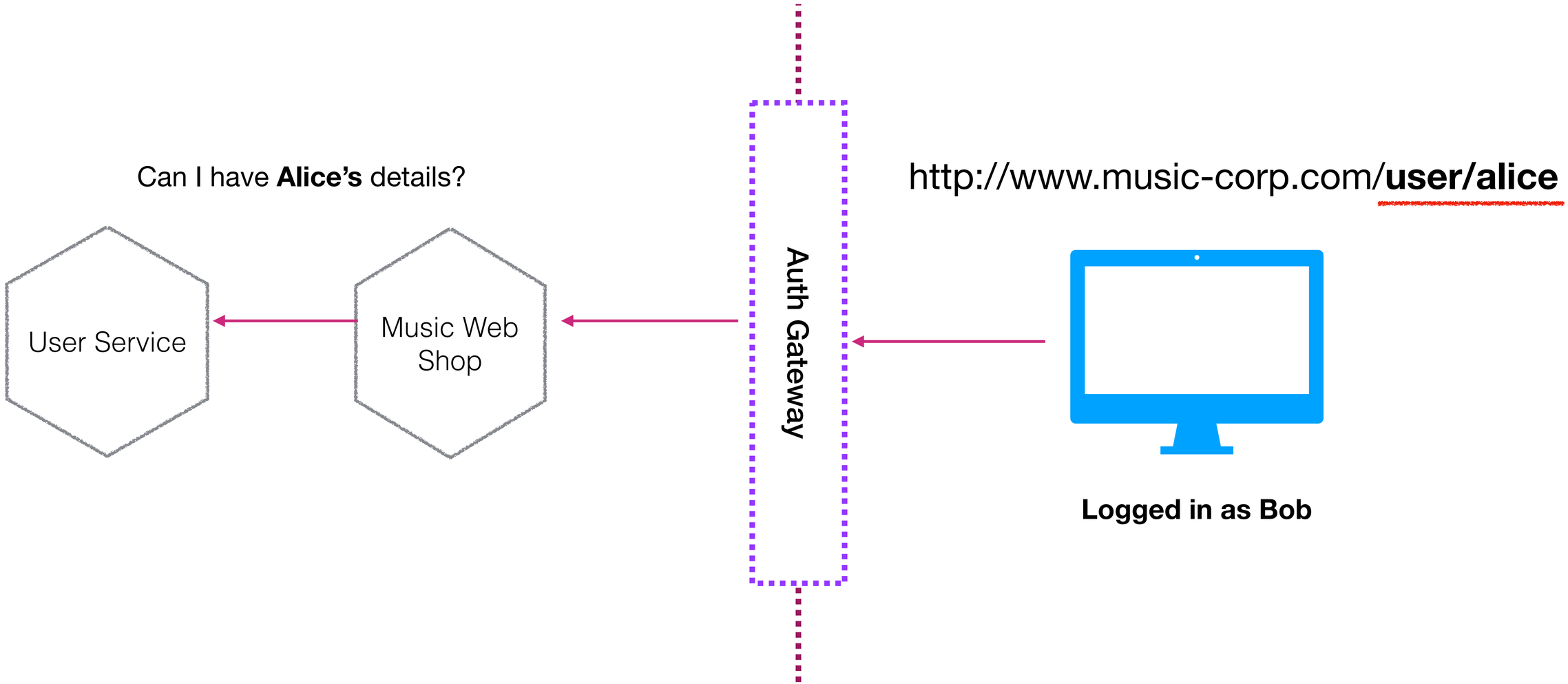
DOWNSTREAM AUTH - IMPLICIT TRUST?



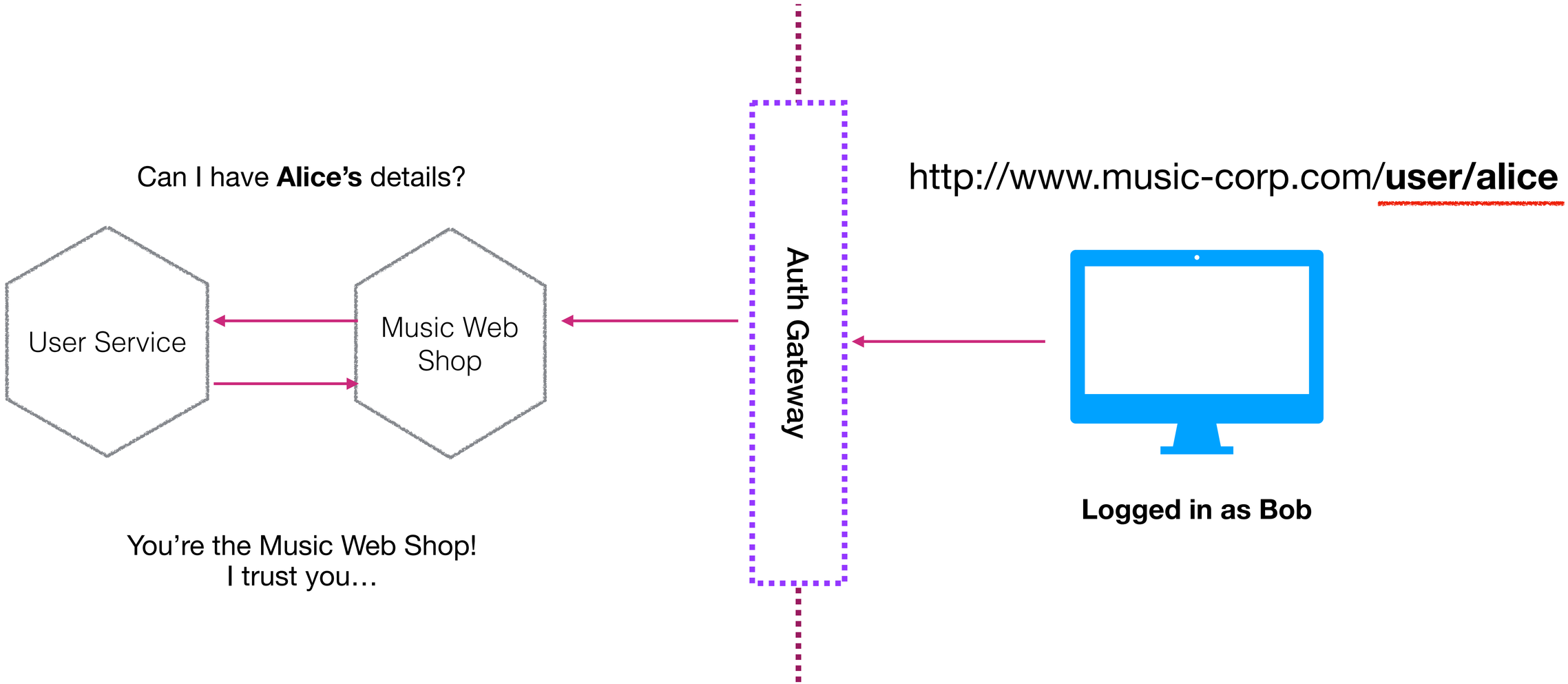
DOWNSTREAM AUTH - IMPLICIT TRUST?



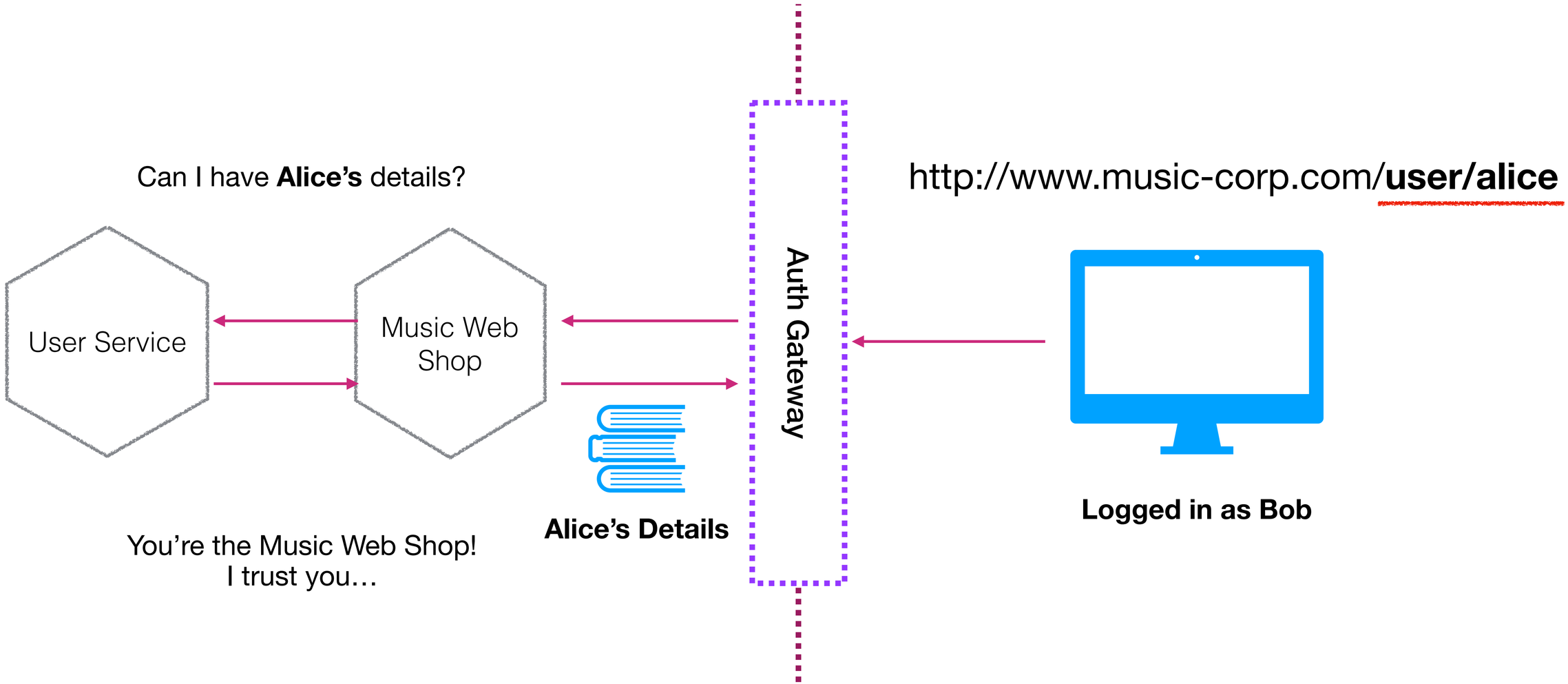
DOWNSTREAM AUTH - IMPLICIT TRUST?



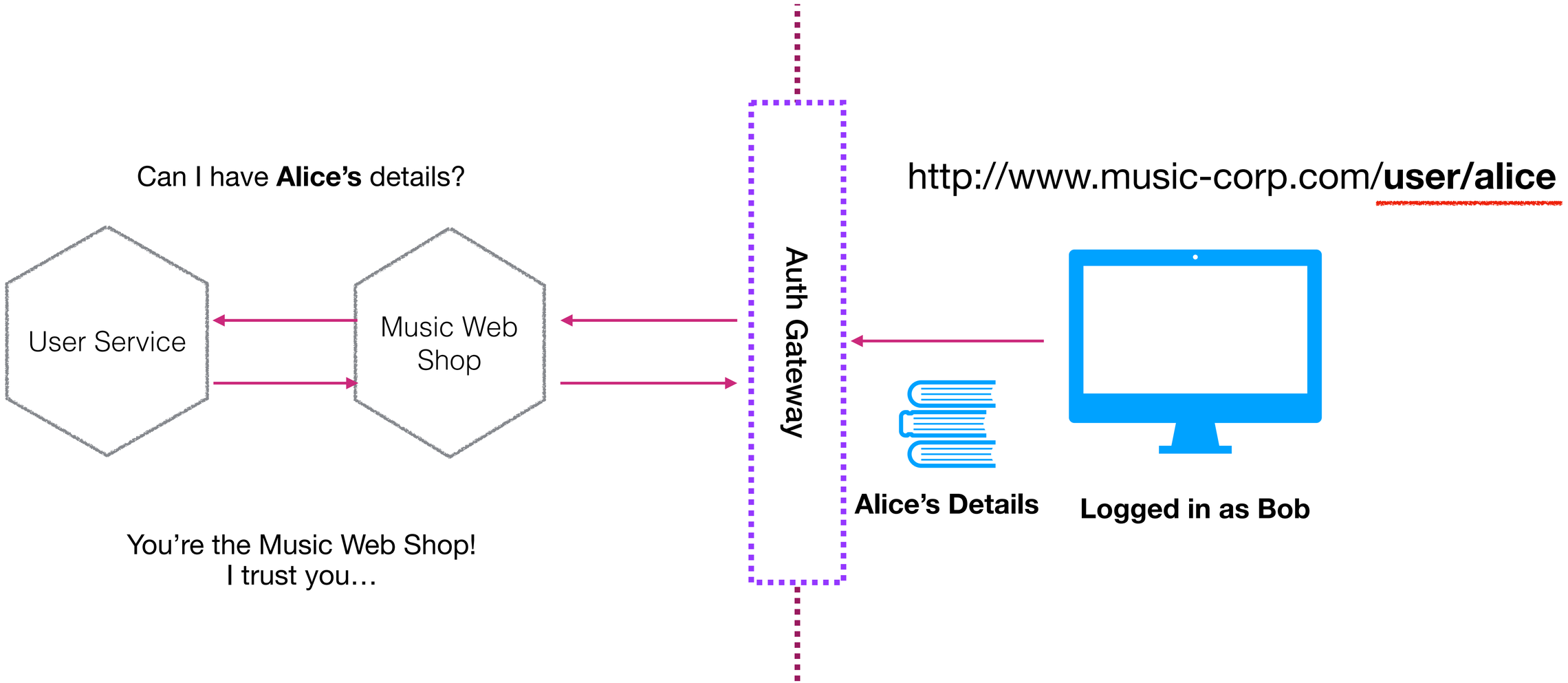
DOWNSTREAM AUTH - IMPLICIT TRUST?



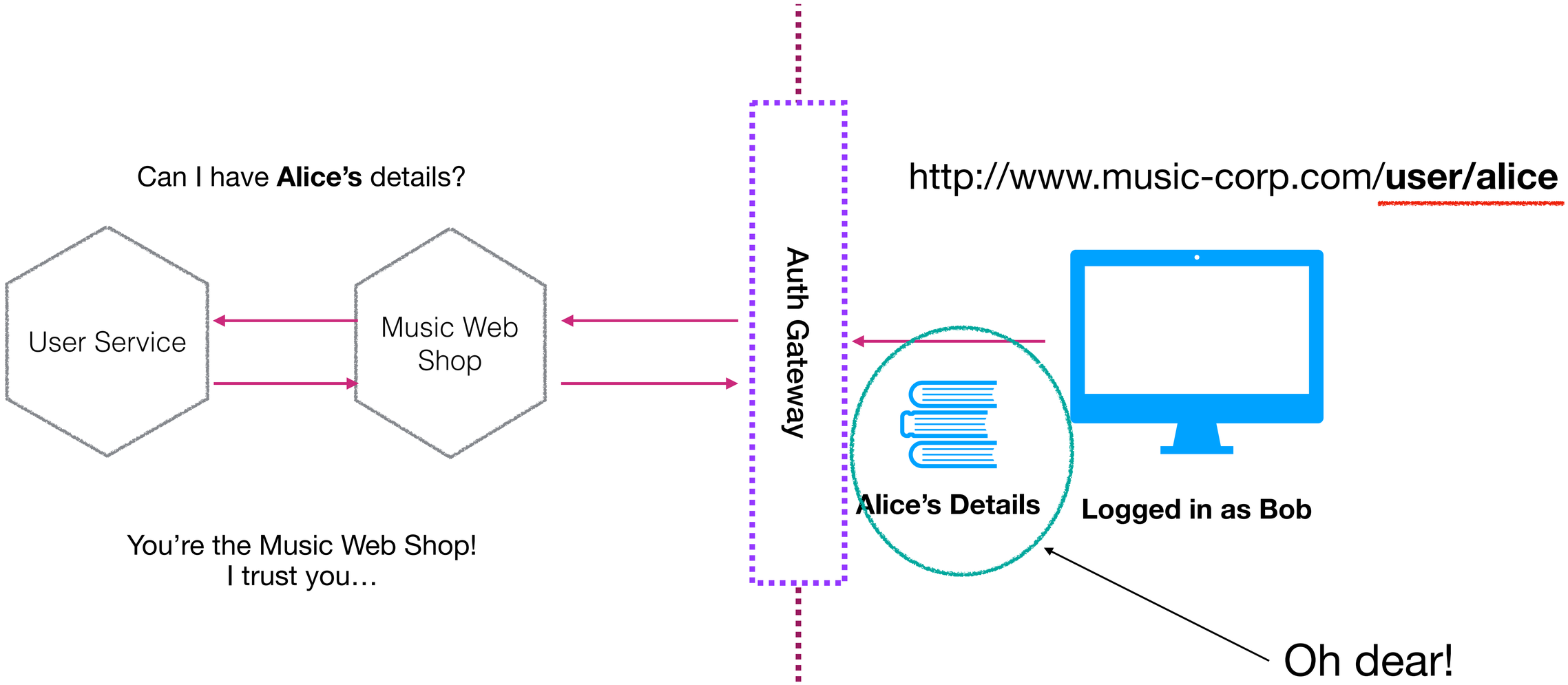
DOWNSTREAM AUTH - IMPLICIT TRUST?



DOWNSTREAM AUTH - IMPLICIT TRUST?



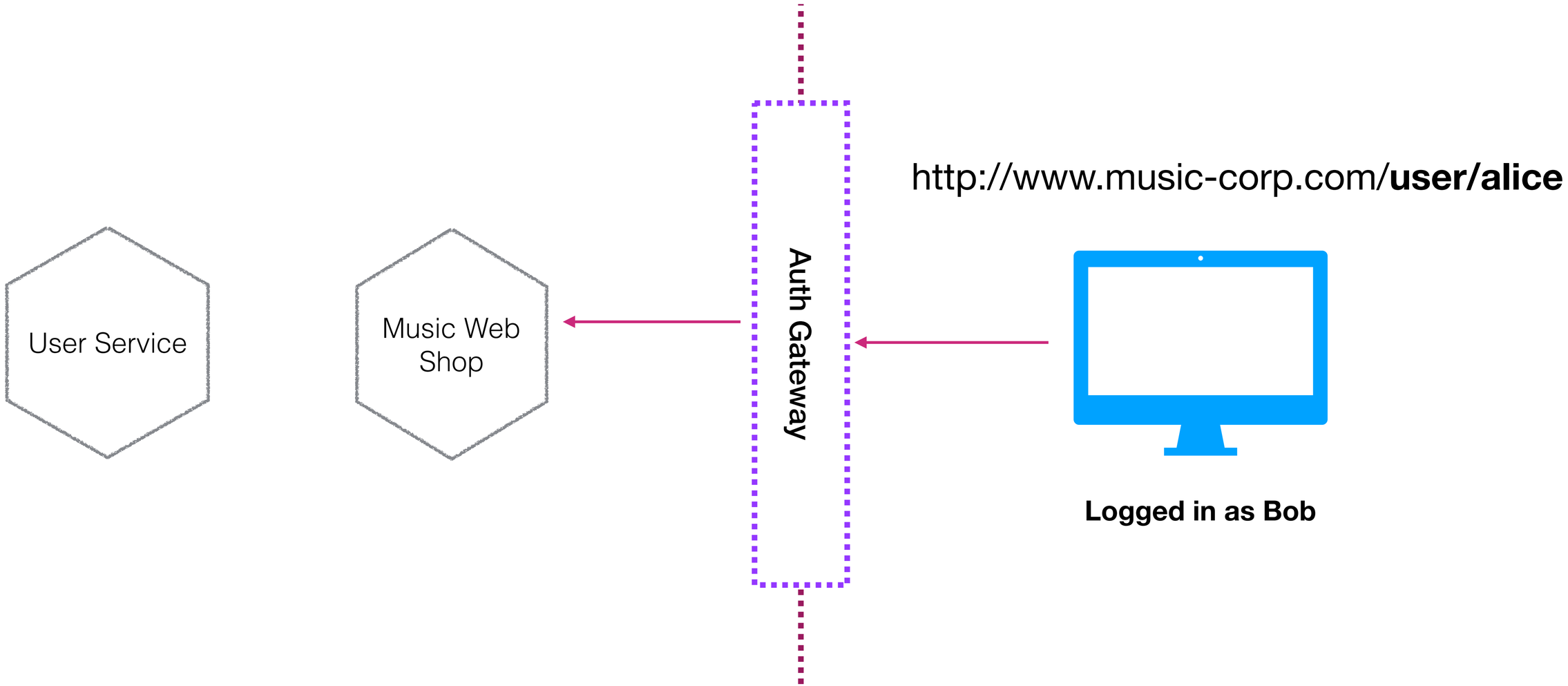
DOWNSTREAM AUTH - IMPLICIT TRUST?



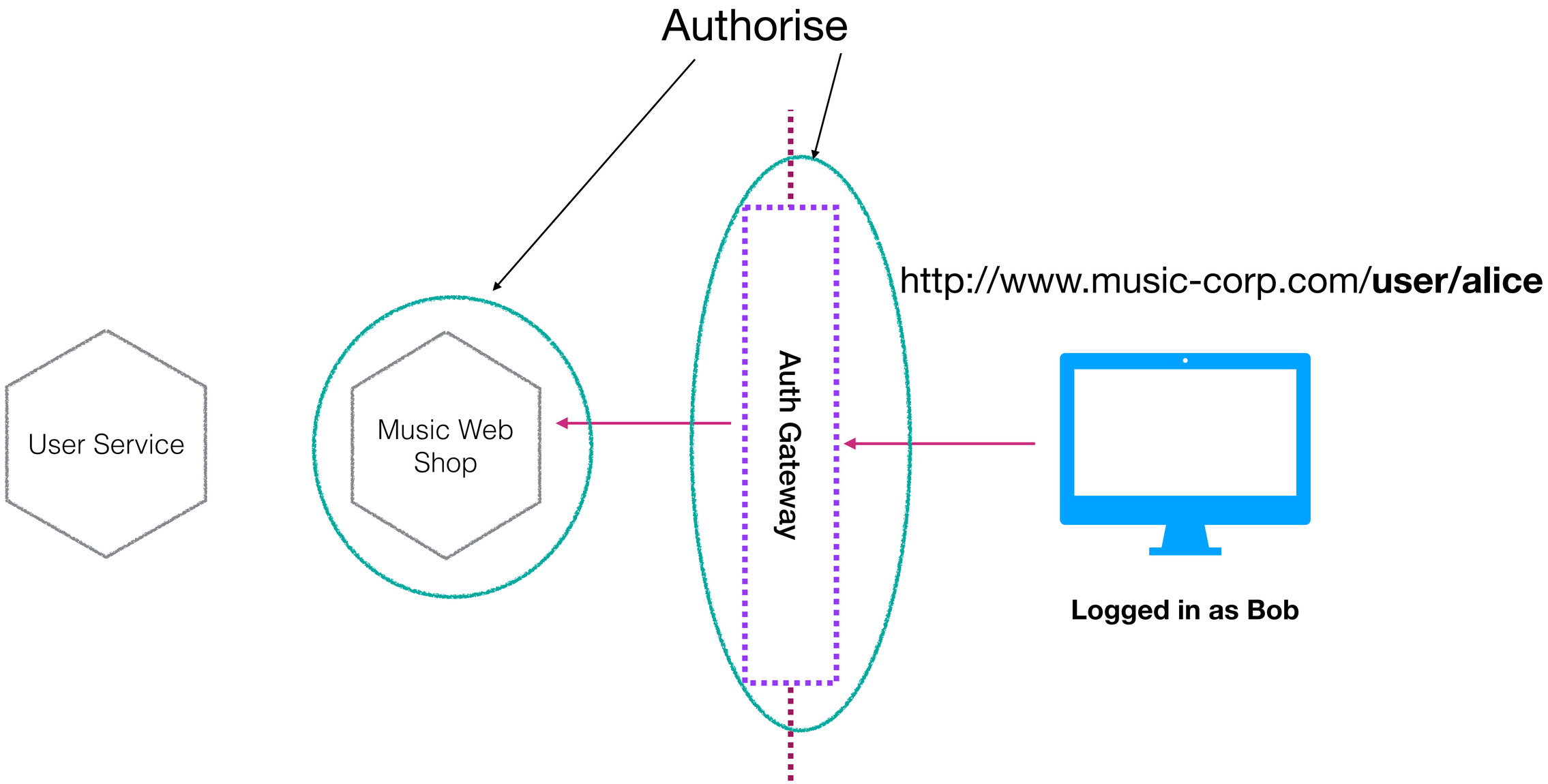
Confused Deputy Problem!

https://www.flickr.com/photos/lundyd/14481829564

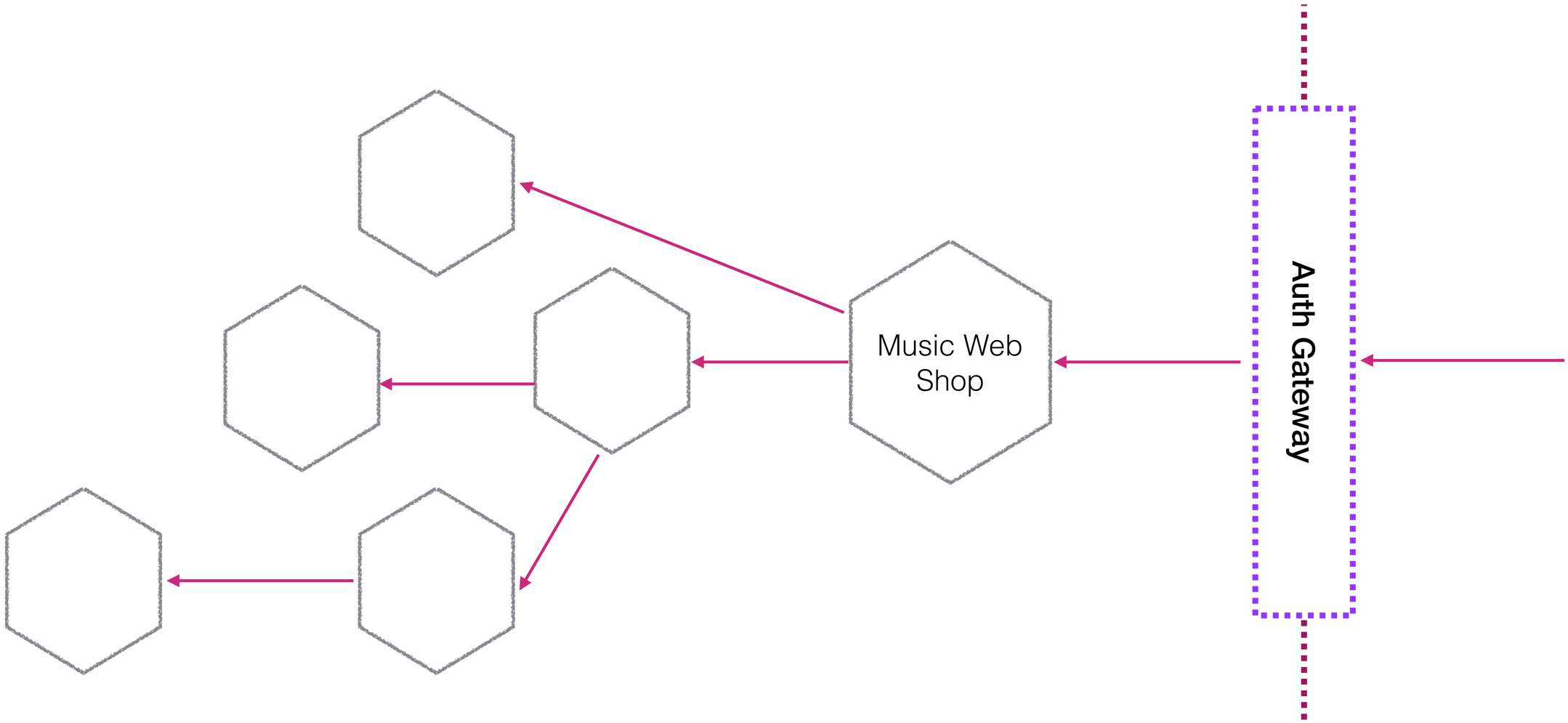
AUTHORISE UPSTREAM?



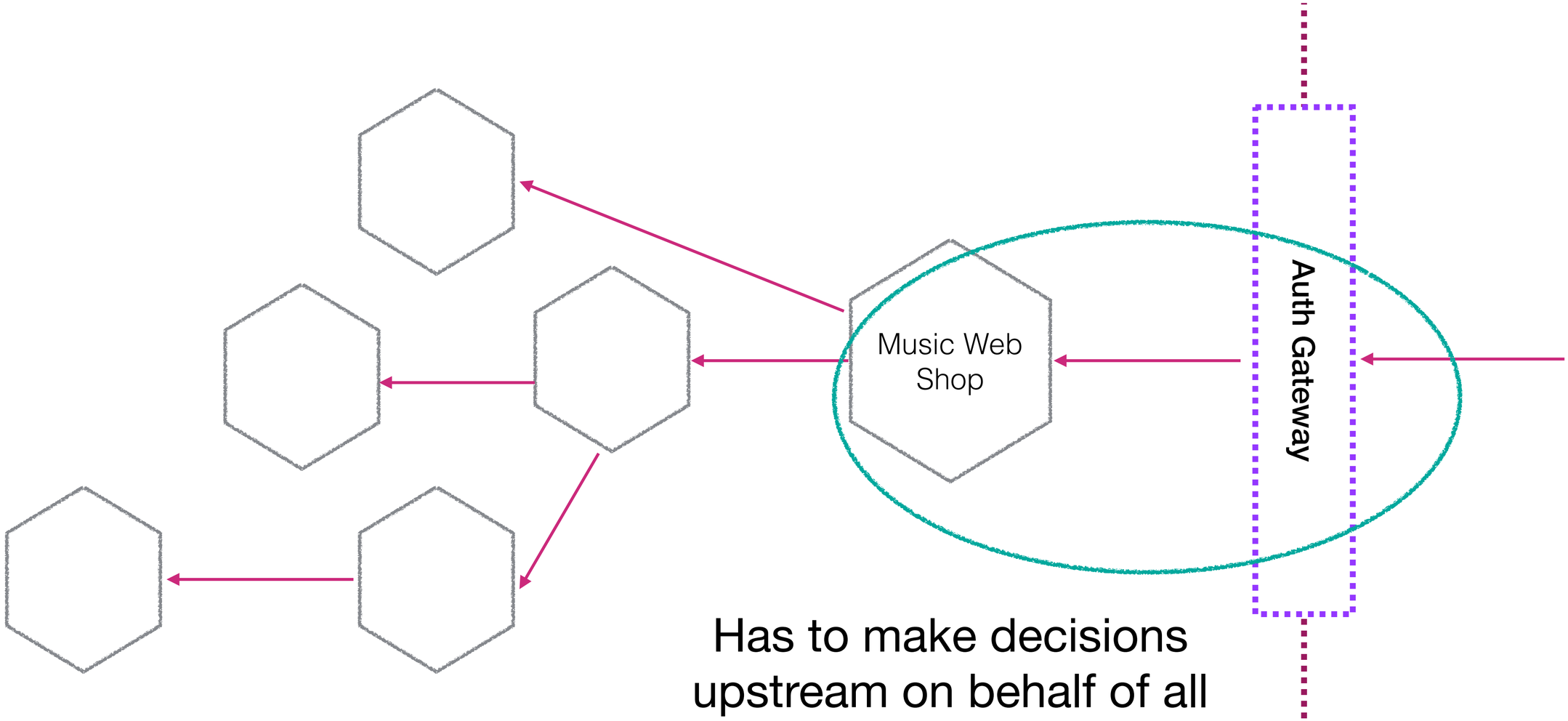
AUTHORISE UPSTREAM?



WHERE DO THE SMARTS LIVE?

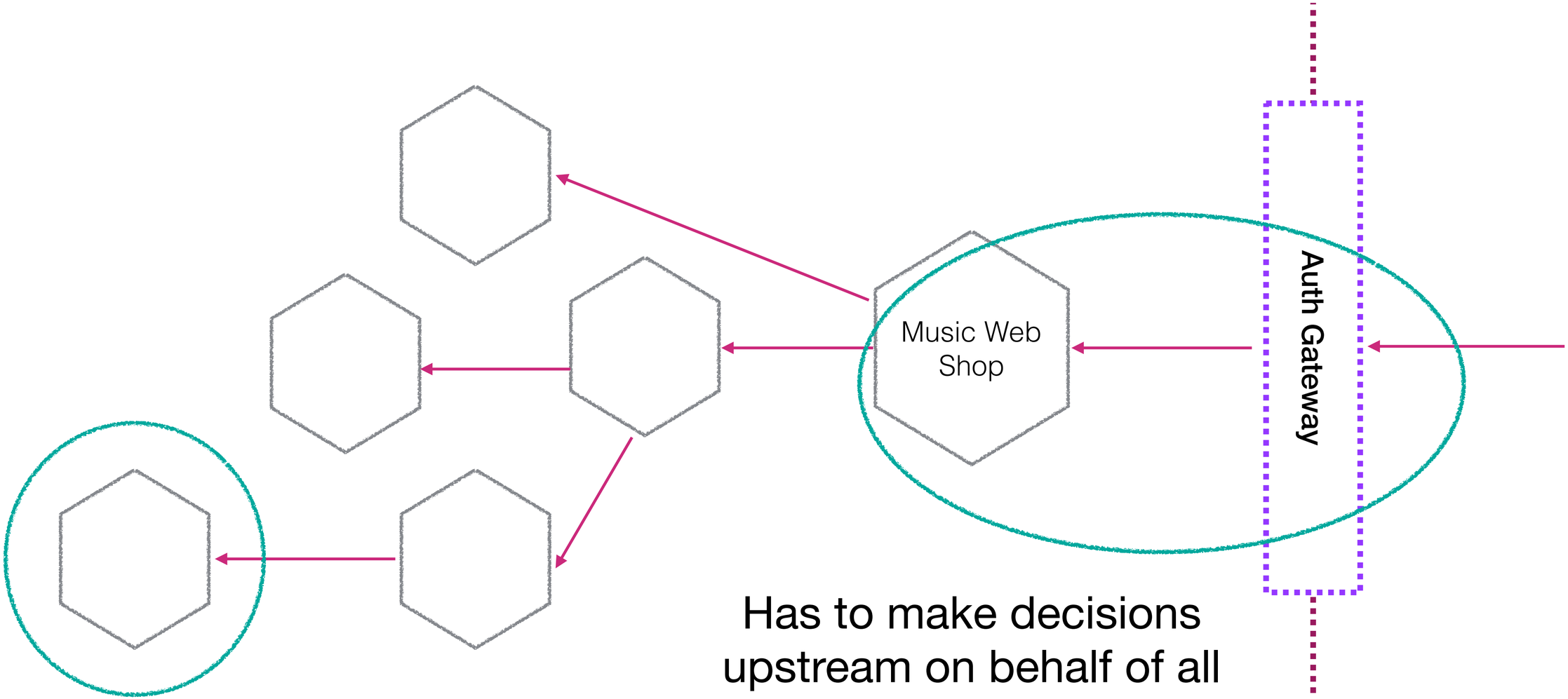


WHERE DO THE SMARTS LIVE?



Has to make decisions upstream on behalf of all downstream services….

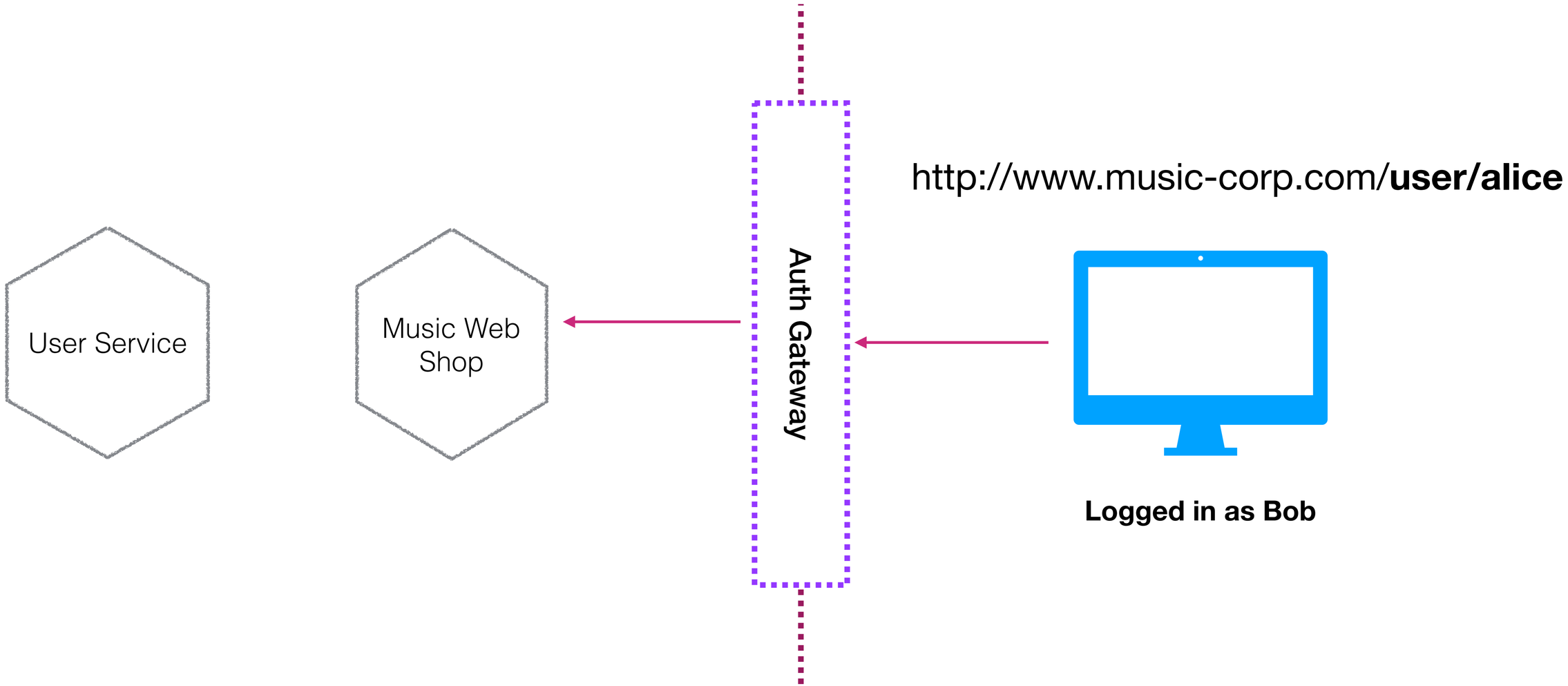
WHERE DO THE SMARTS LIVE?



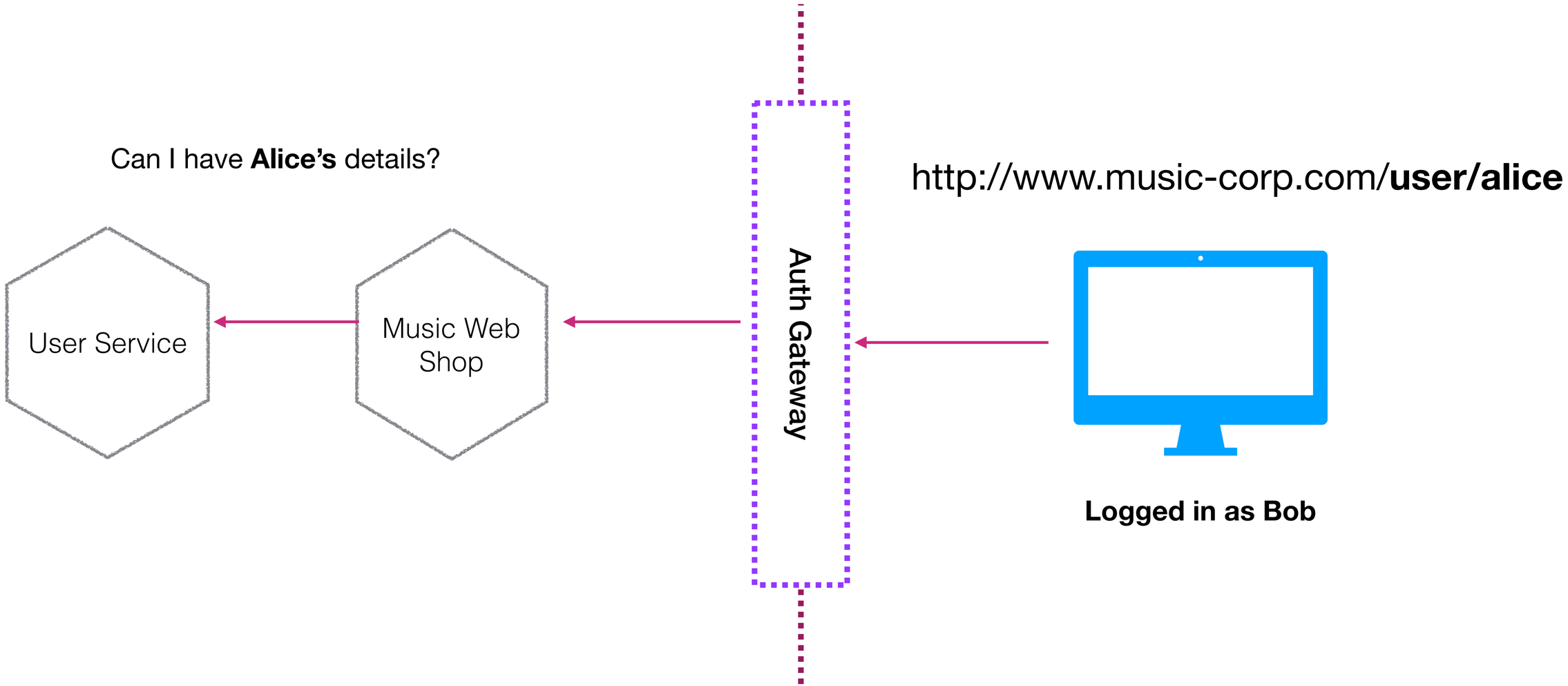
But it can be preferable to push this logic to the service itself

Has to make decisions upstream on behalf of all downstream services….

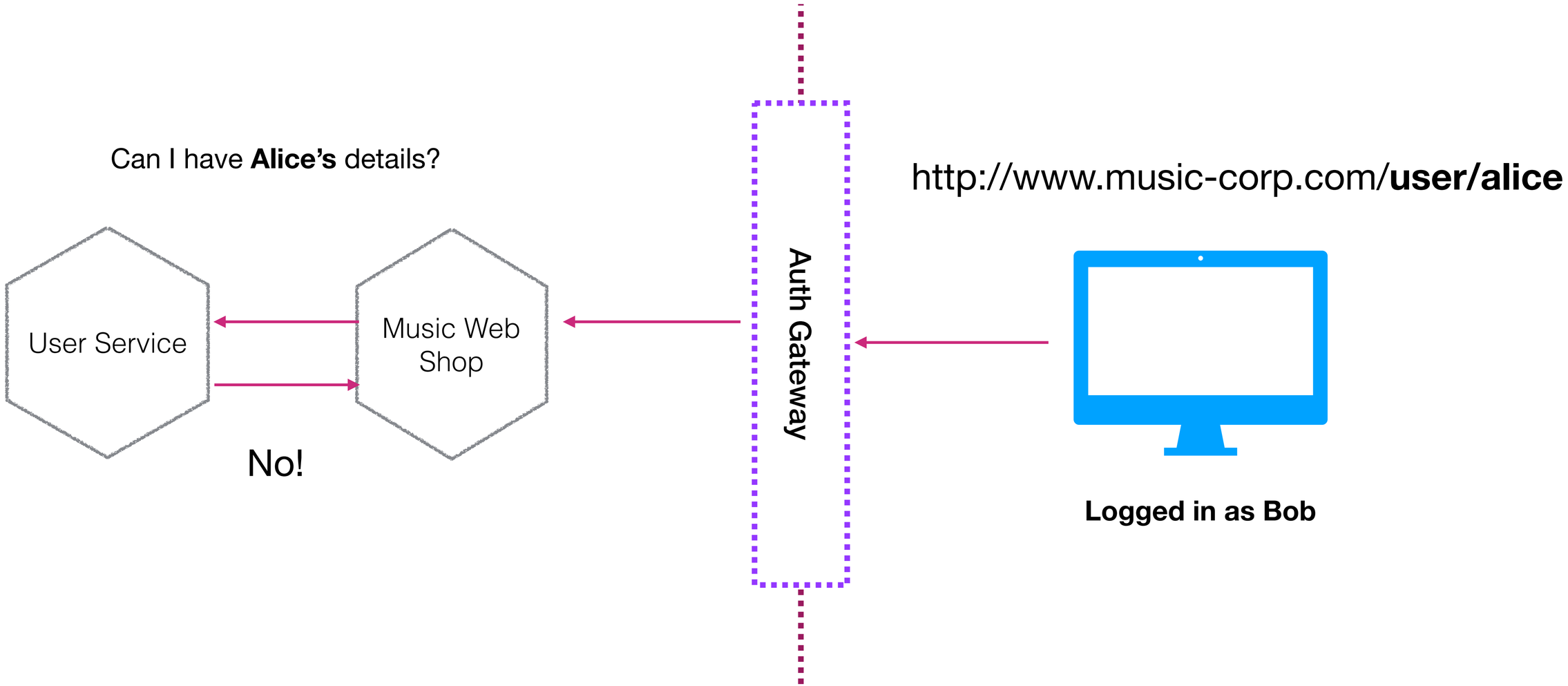
AUTHORISE DOWNSTREAM



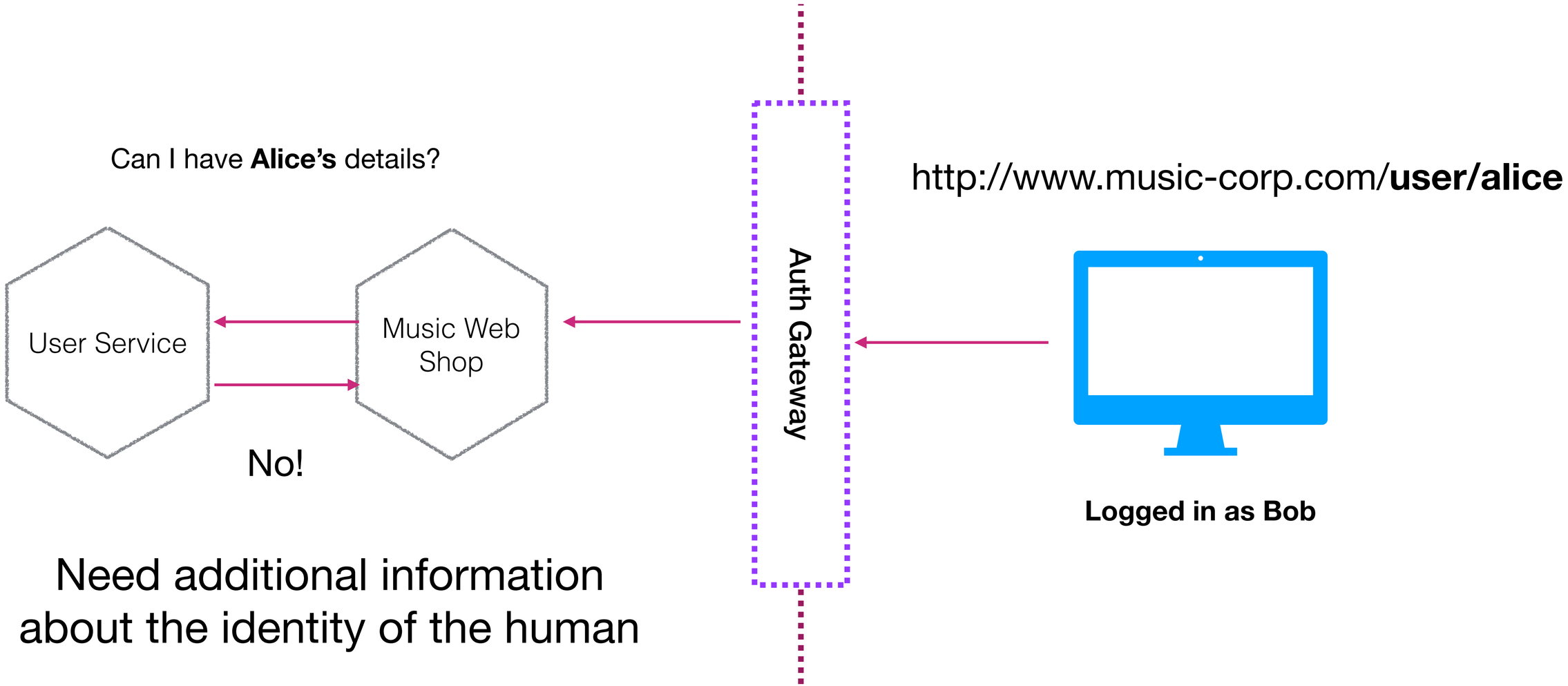
AUTHORISE DOWNSTREAM



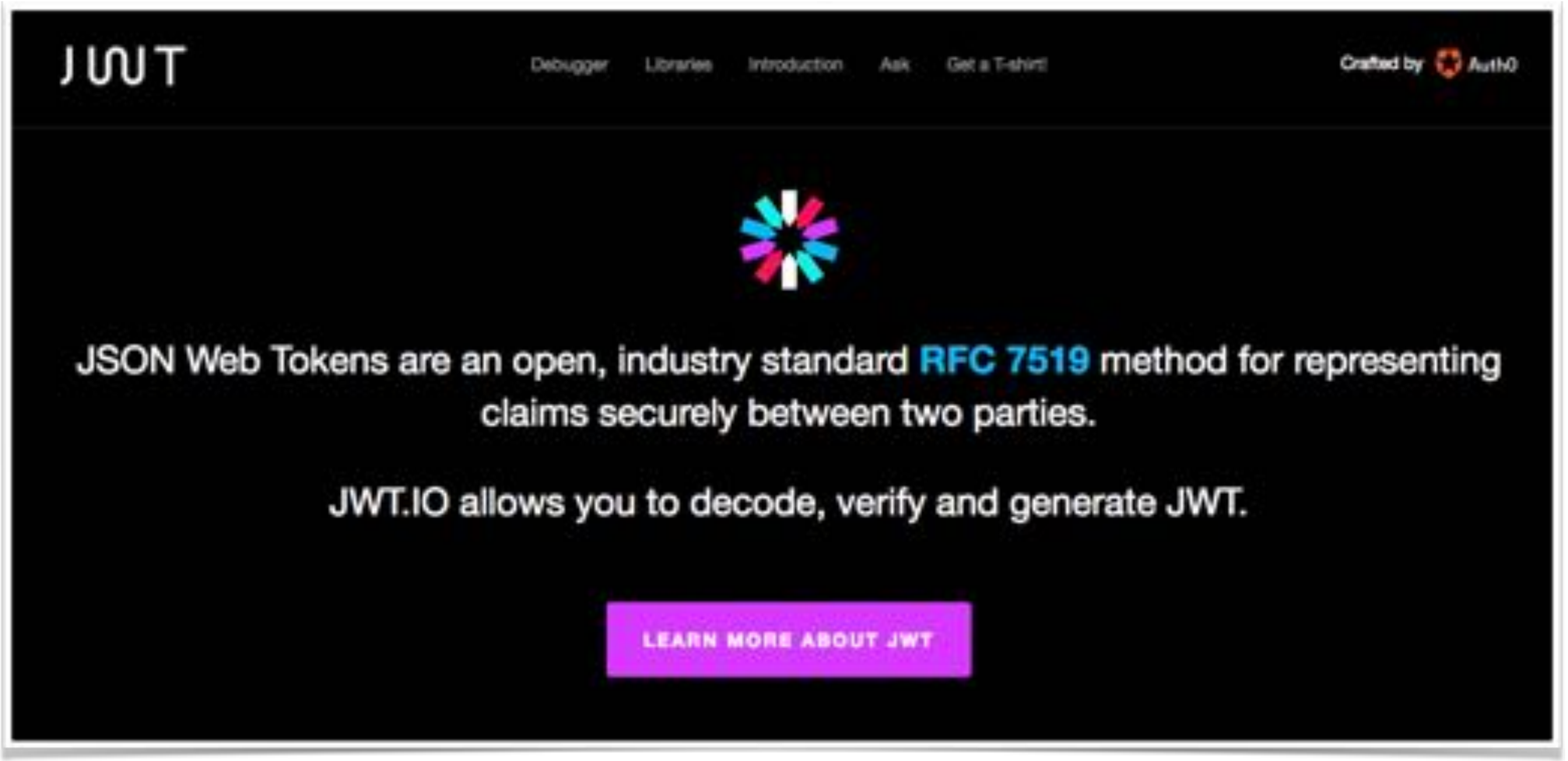
AUTHORISE DOWNSTREAM



AUTHORISE DOWNSTREAM



Need additional information about the identity of the human


JWT
Debugger
Libraries
Introduction
Ask
Get a T-shirt!
Crafted by Auth0
JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties.
JWT.IO allows you to decode, verify and generate JWT.
LEARN MORE ABOUT JWT

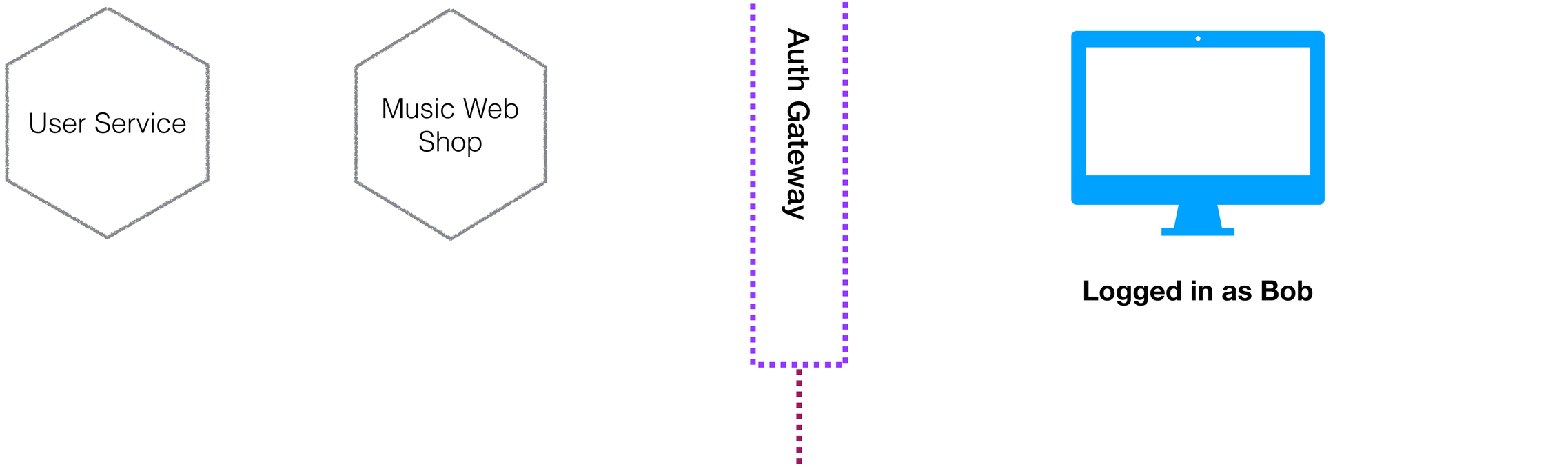
https://jwt.io/

```
{
   "id": "402ndj39",
   "name": “Alice Alison"
}
```

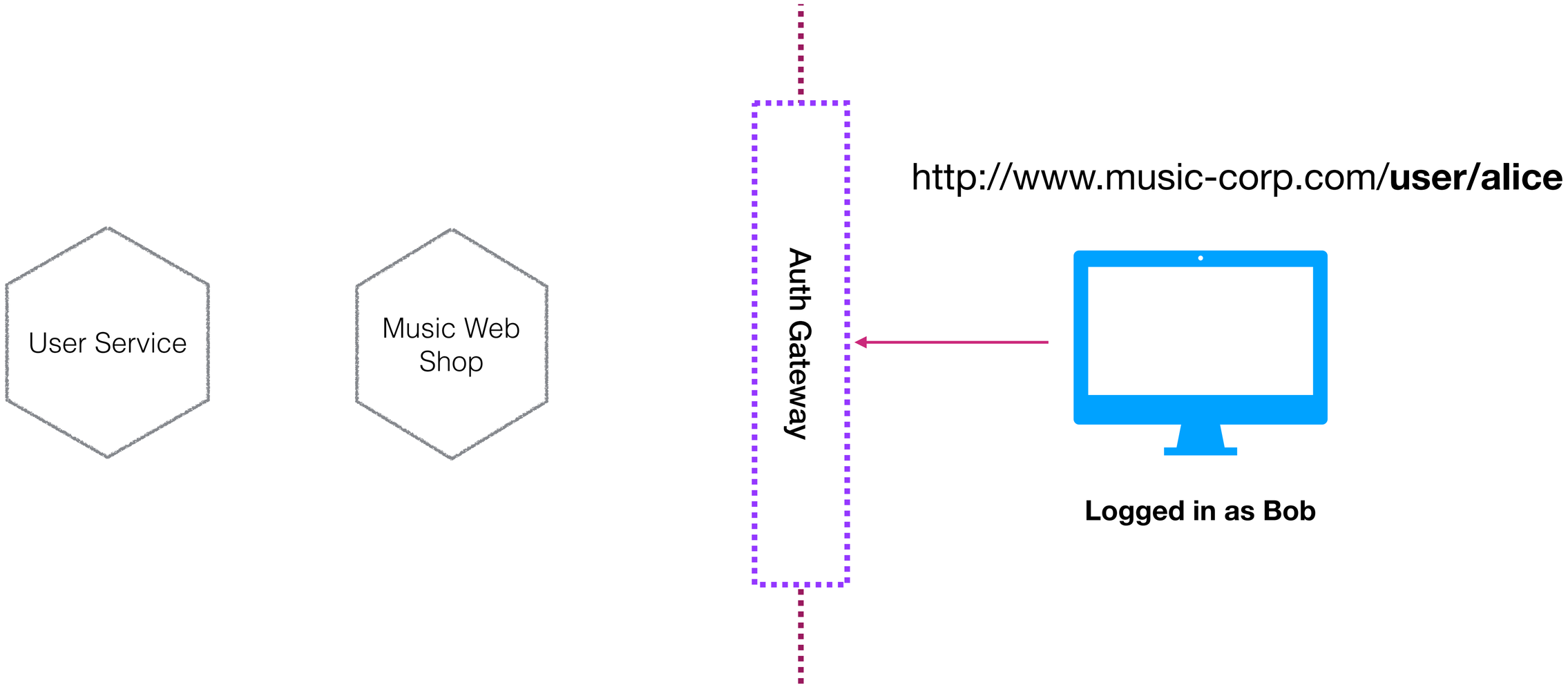
```
{
  "id": "402ndj39",
  "name": “Alice Alison"
}
```

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

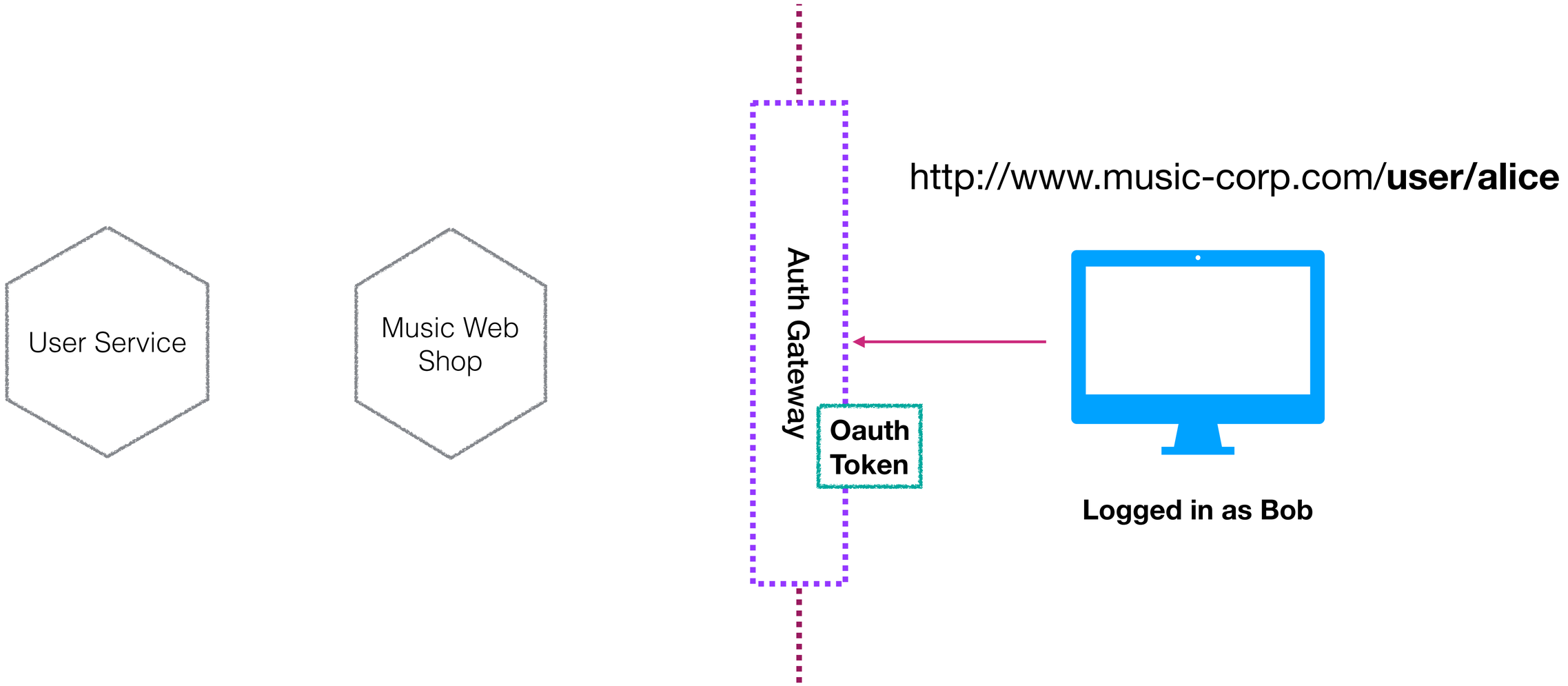
USING JWT TOKENS



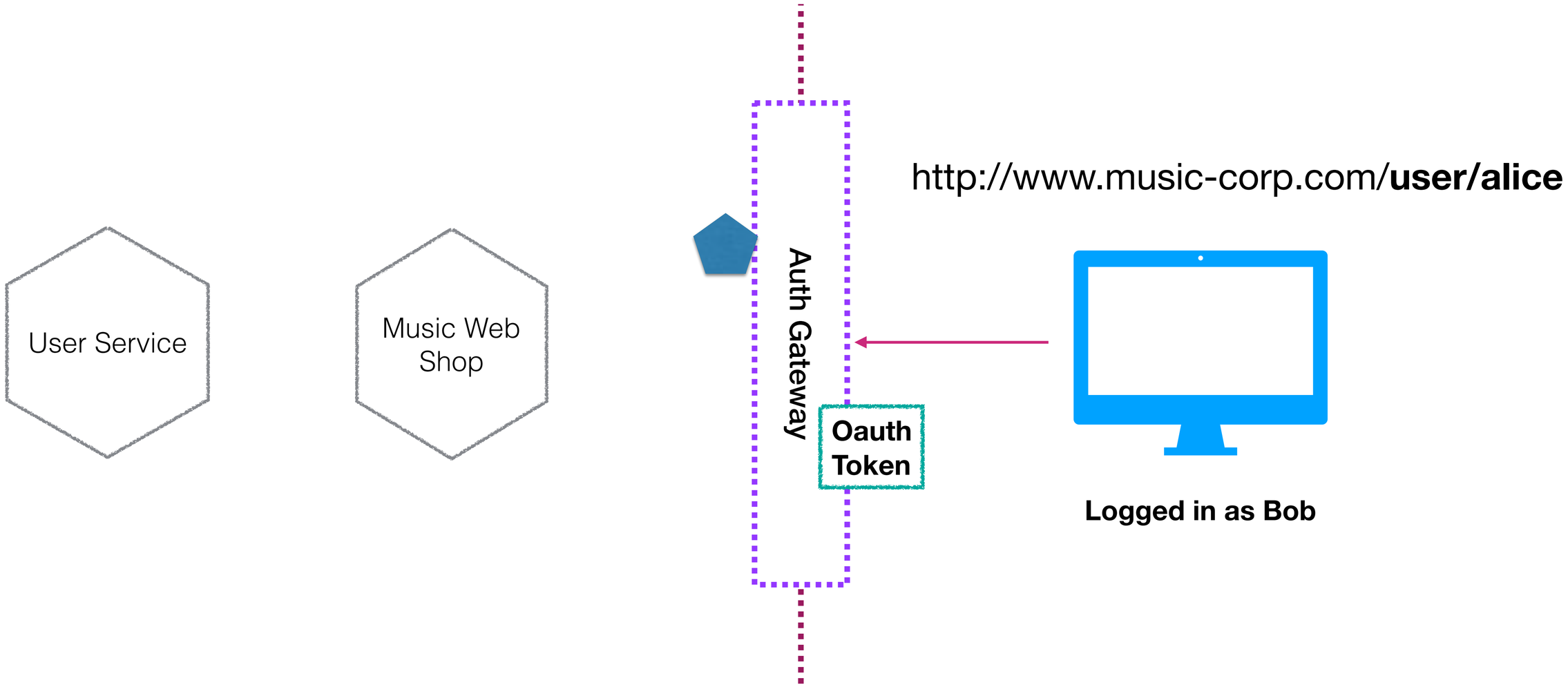
USING JWT TOKENS



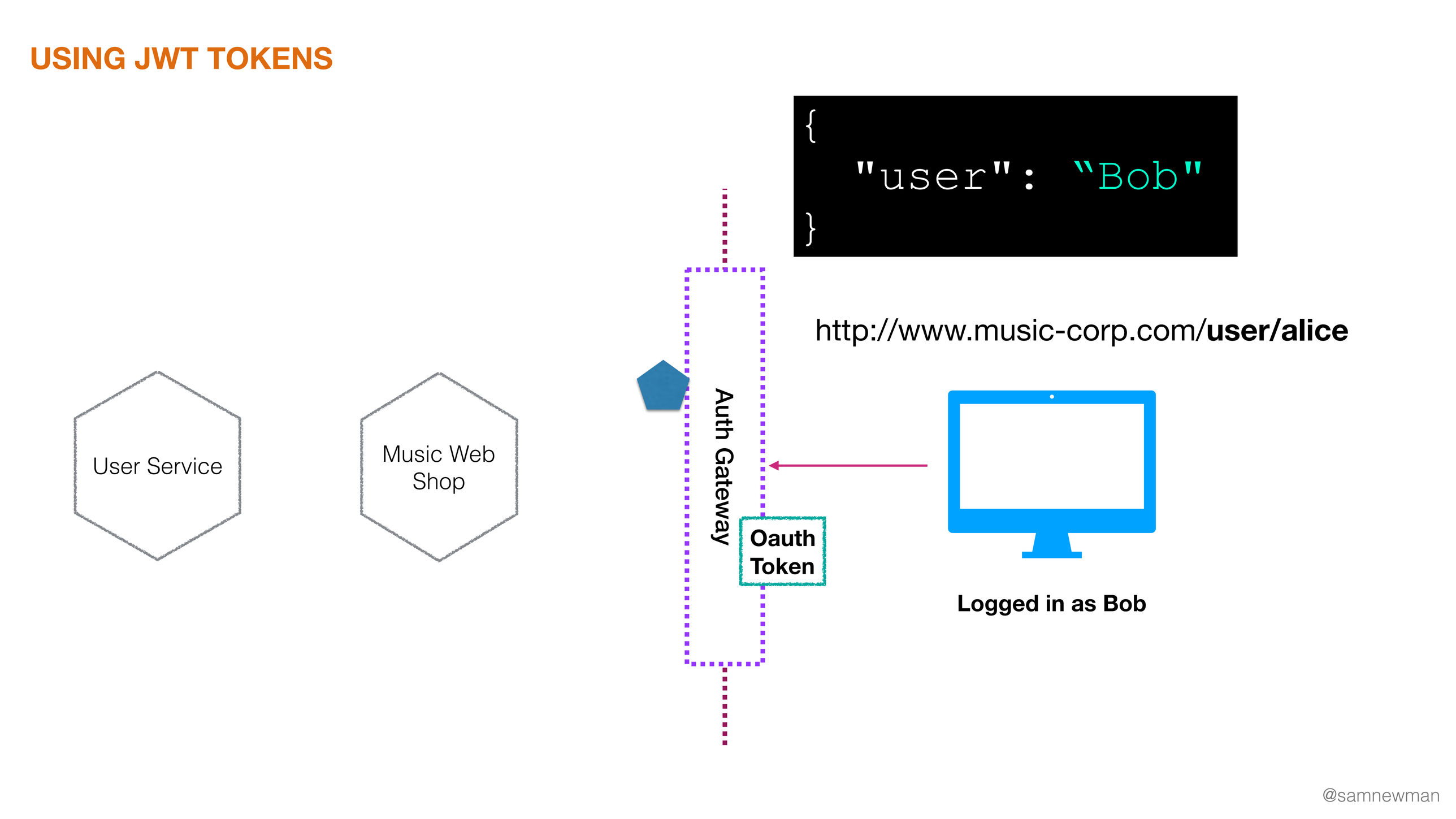
USING JWT TOKENS



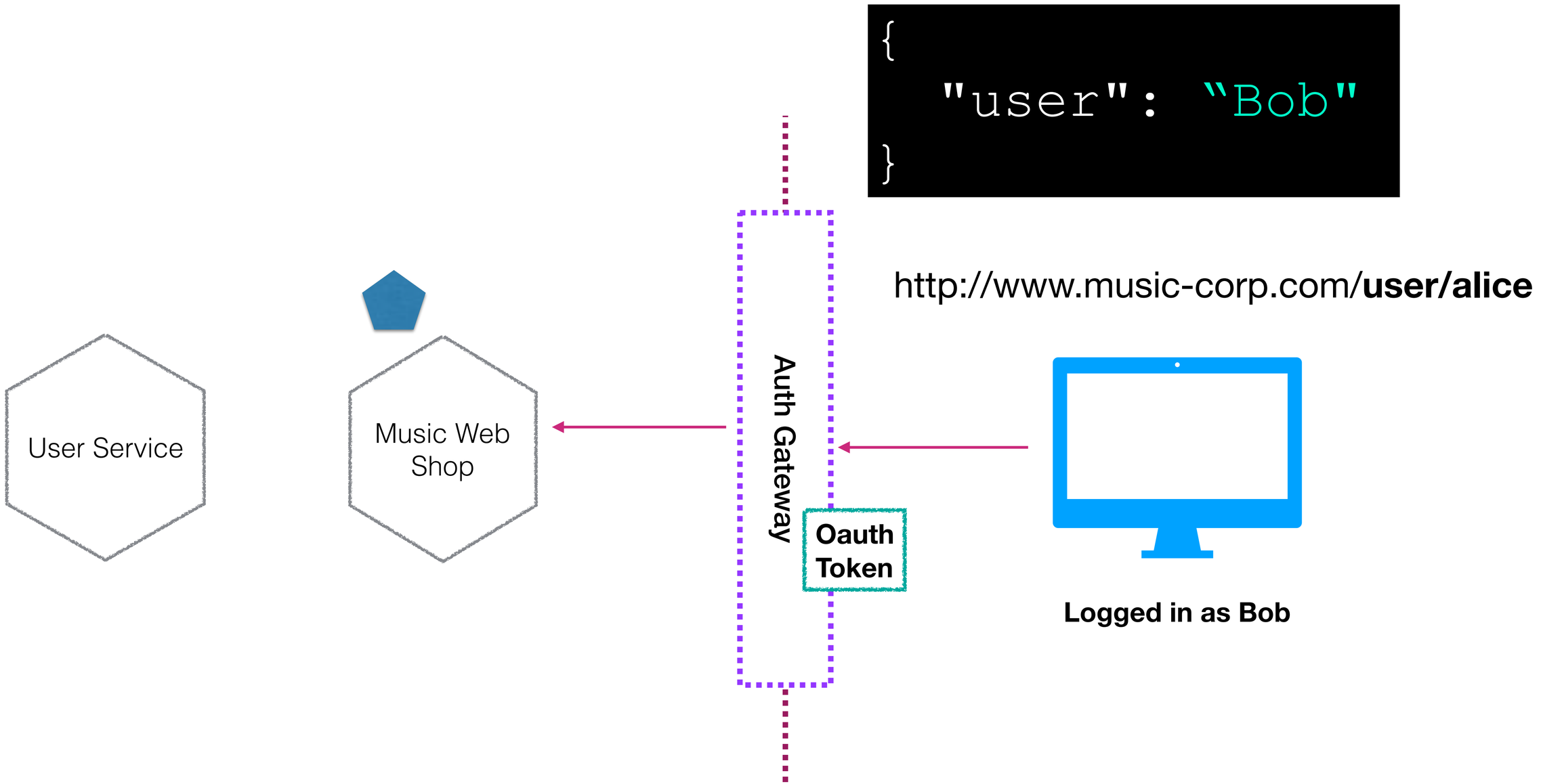
USING JWT TOKENS



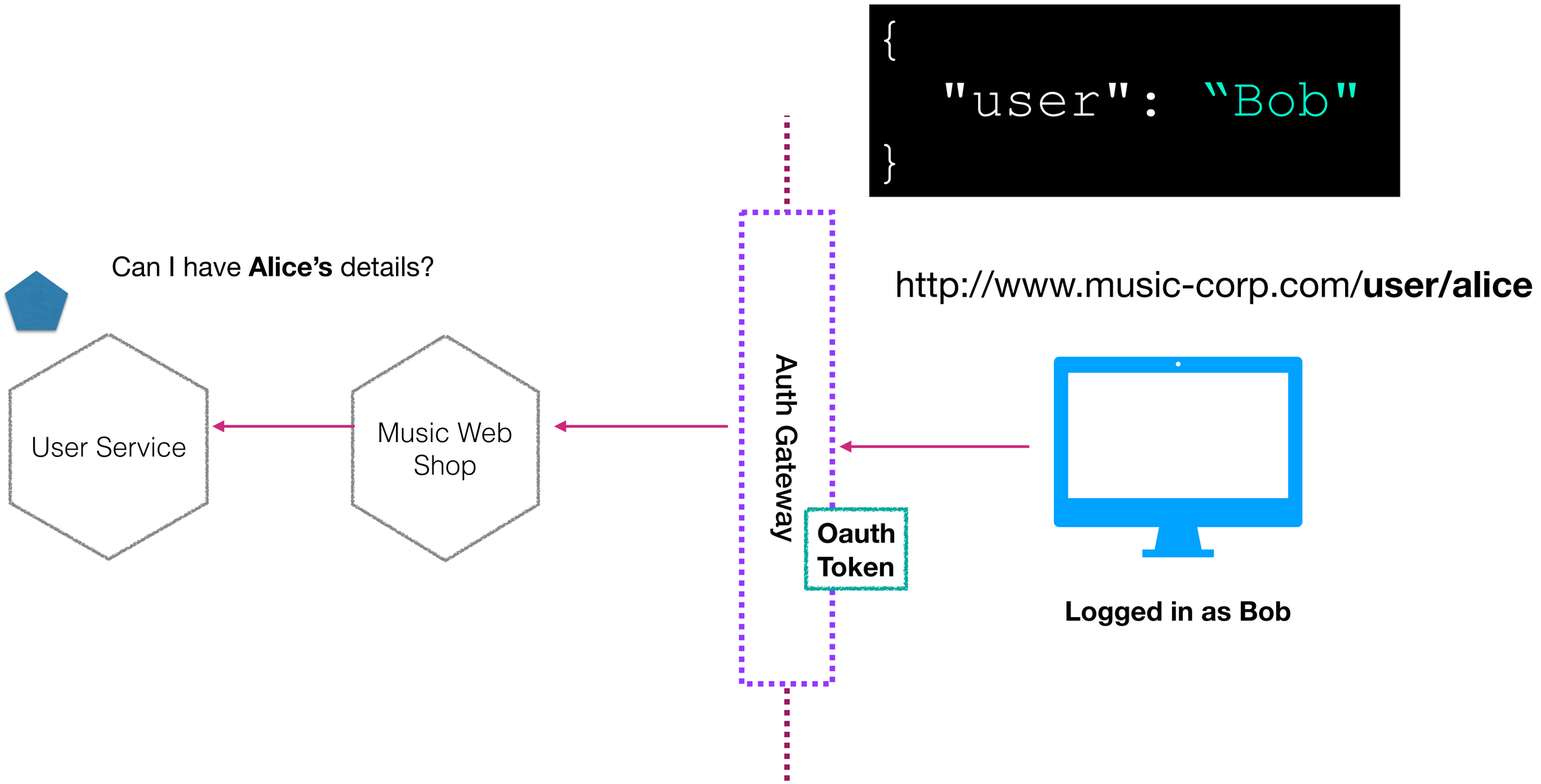
USING JWT TOKENS



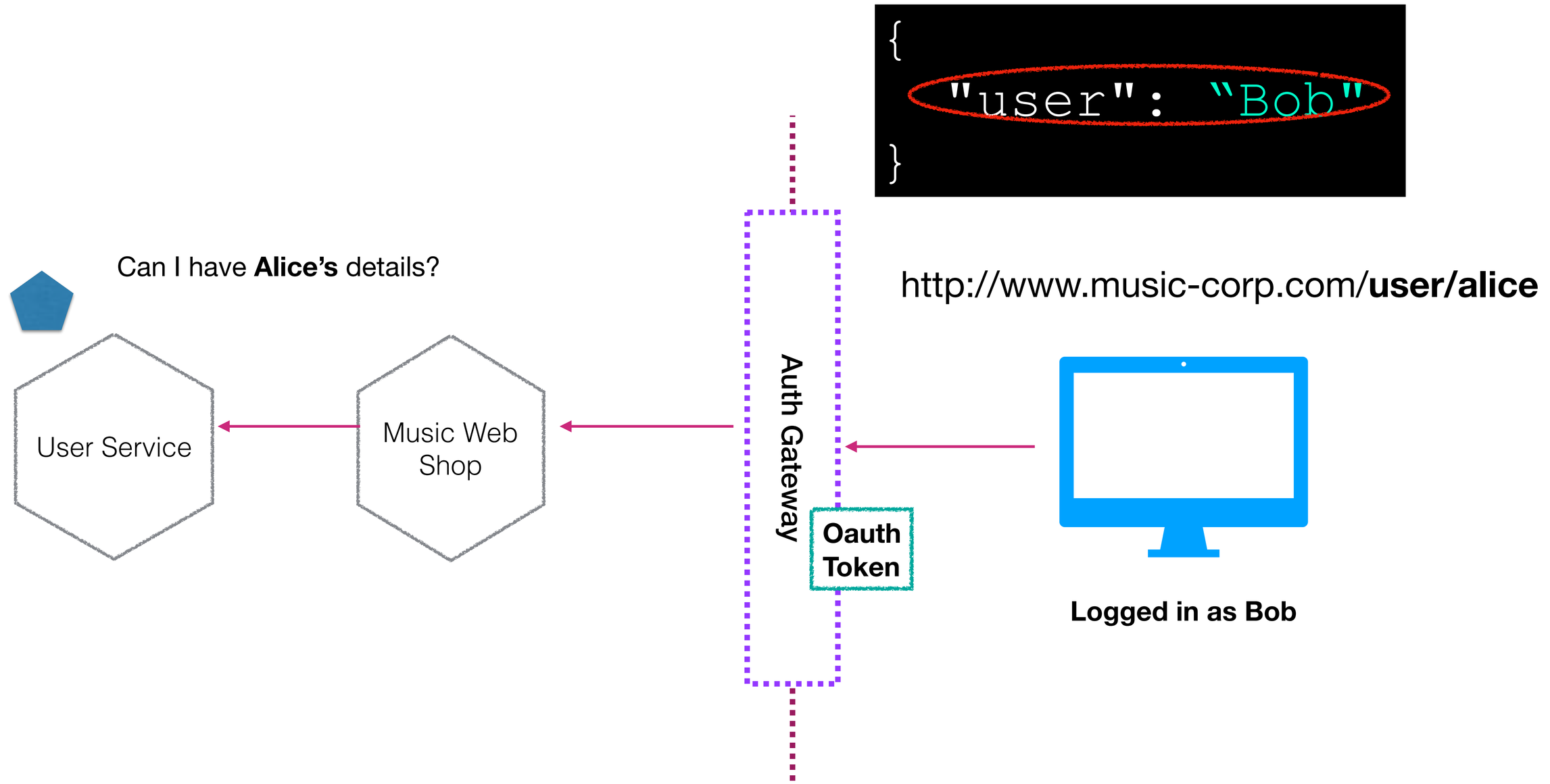
USING JWT TOKENS



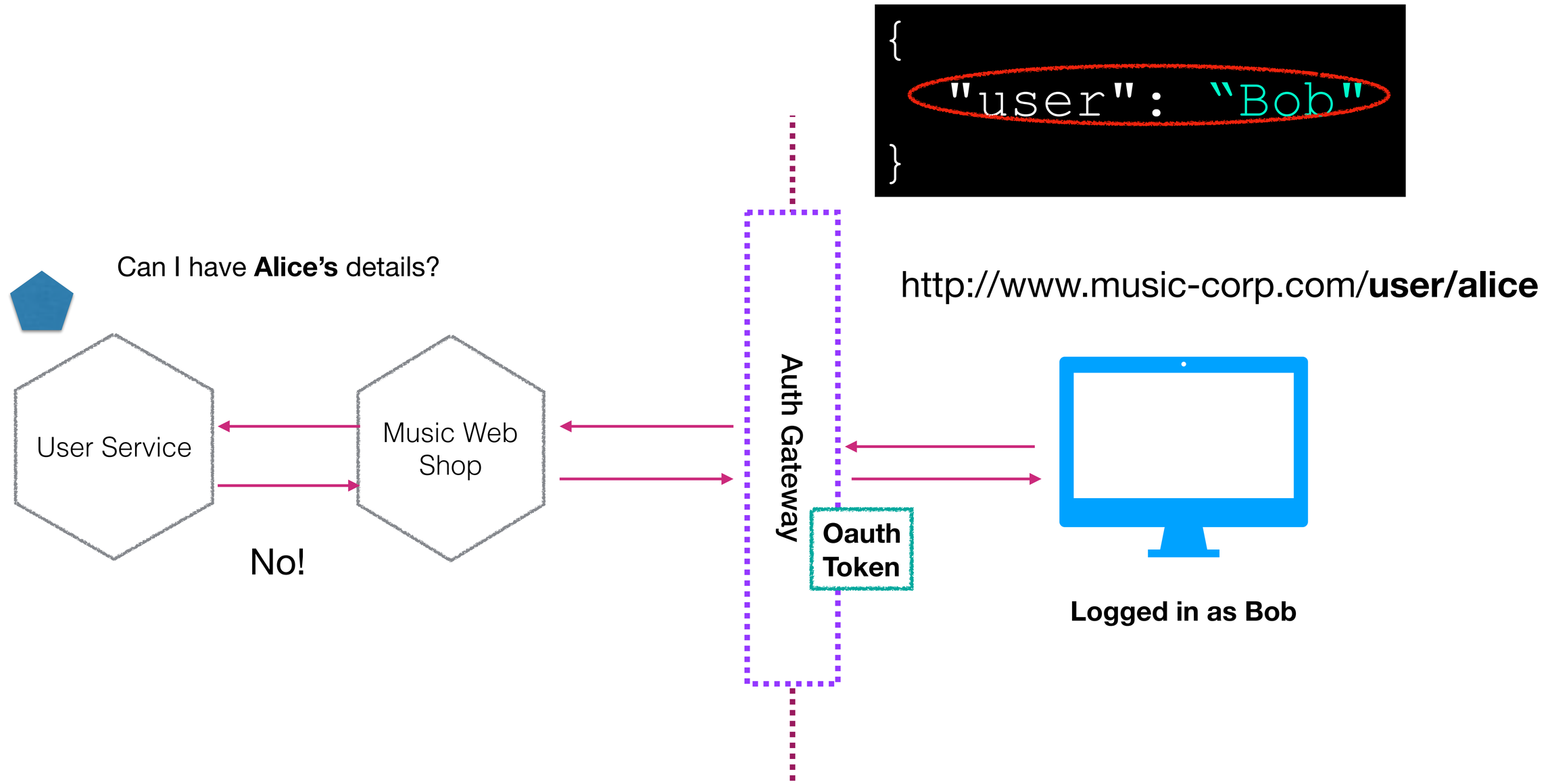
USING JWT TOKENS



USING JWT TOKENS

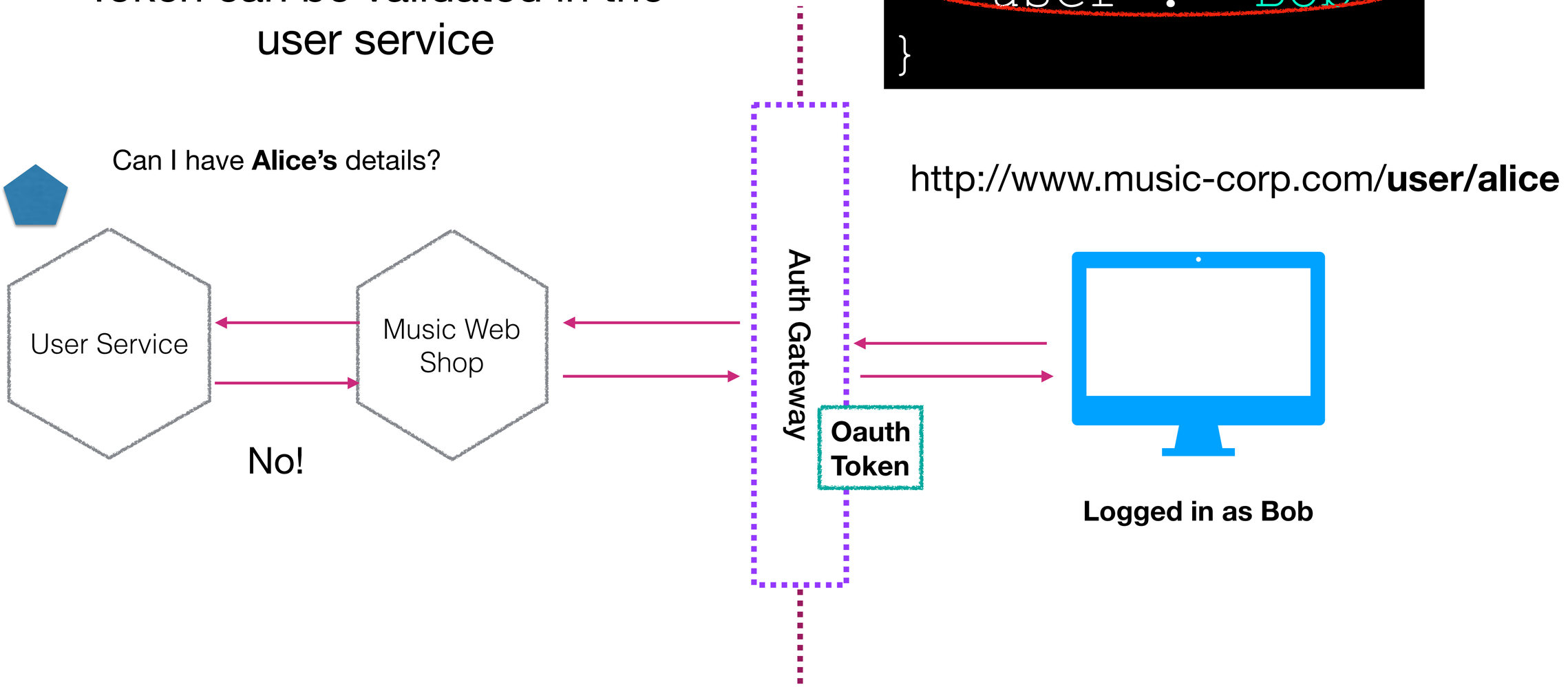


USING JWT TOKENS



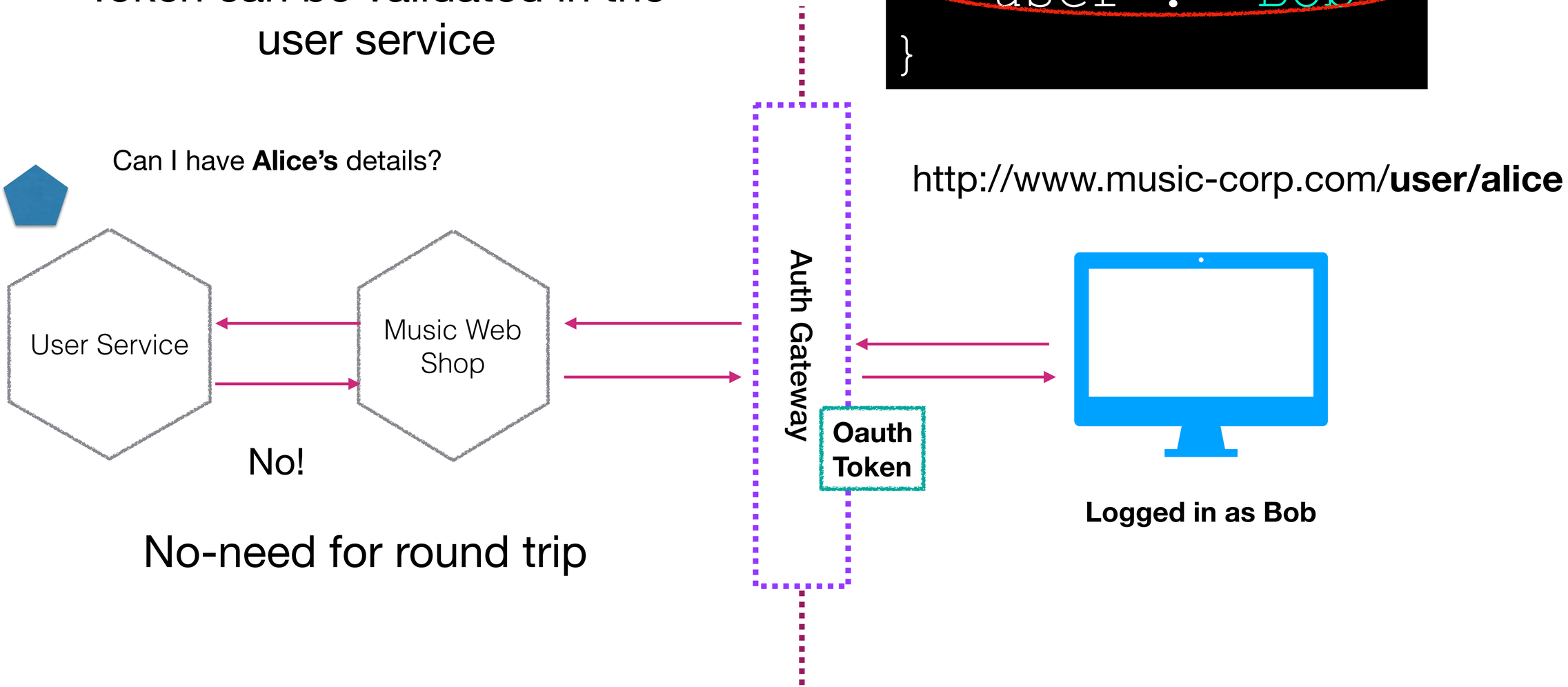
USING JWT TOKENS

Token can be validated in the user service



USING JWT TOKENS

Token can be validated in the user service



No-need for round trip

SERVICE MESHES

Linkerd

https://linkerd.io

SERVICE MESHES

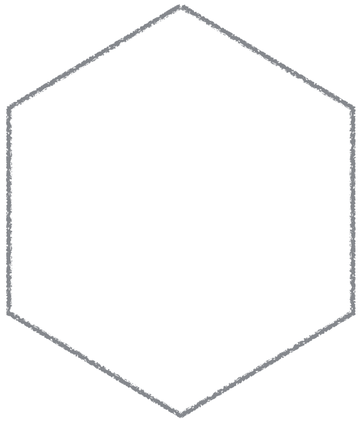
Linkerd

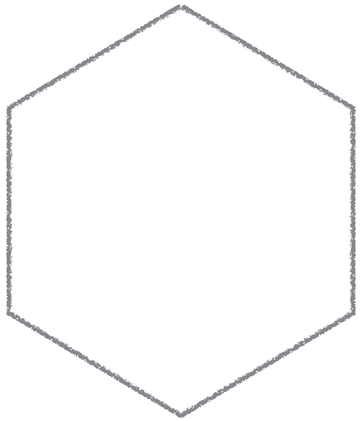
https://linkerd.io

Istio

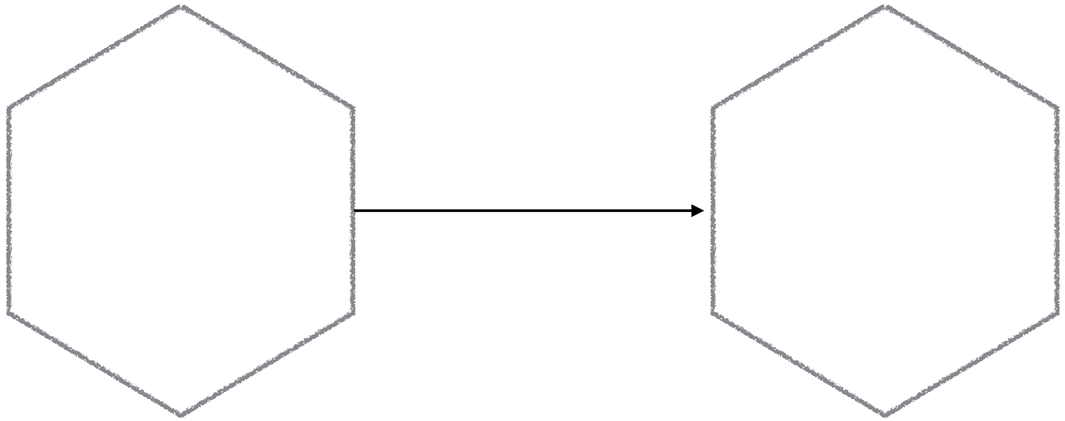
https://istio.io

COMMON CONNECTION CONCERNS

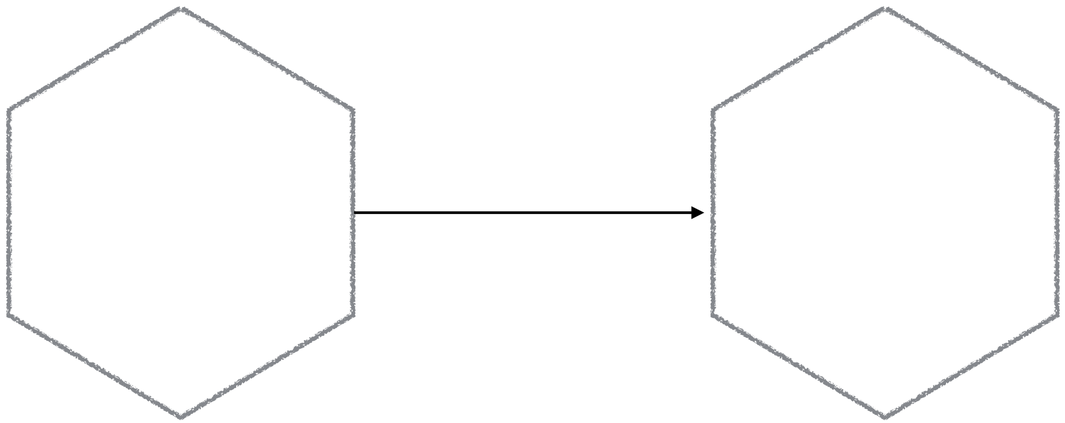




COMMON CONNECTION CONCERNS

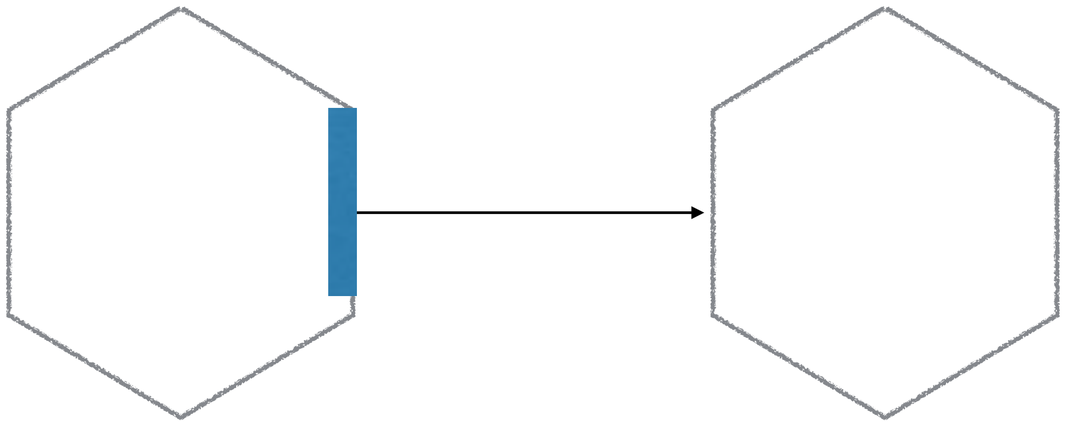


COMMON CONNECTION CONCERNS



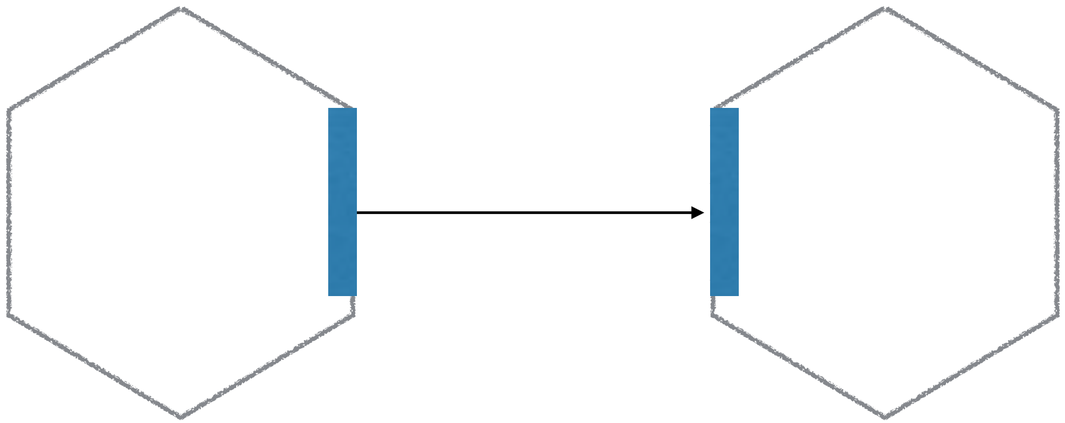
Tracing

COMMON CONNECTION CONCERNS



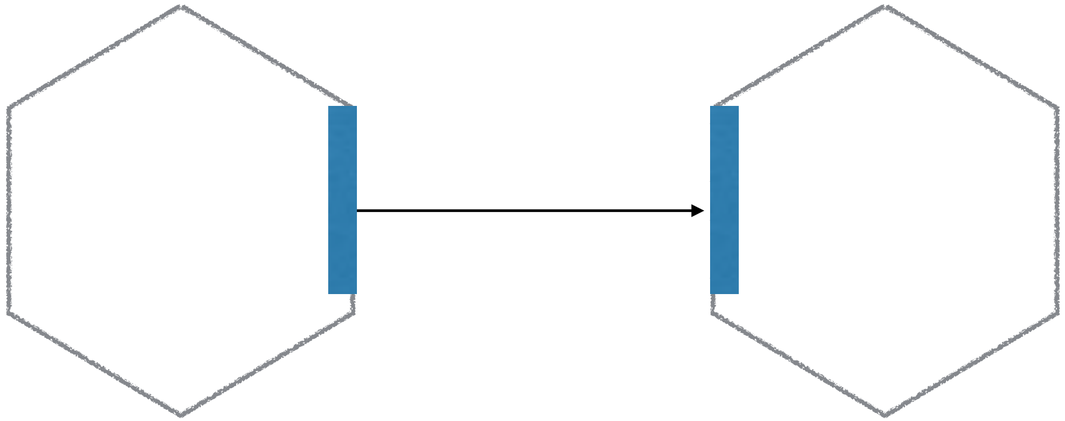
Tracing

COMMON CONNECTION CONCERNS



Tracing

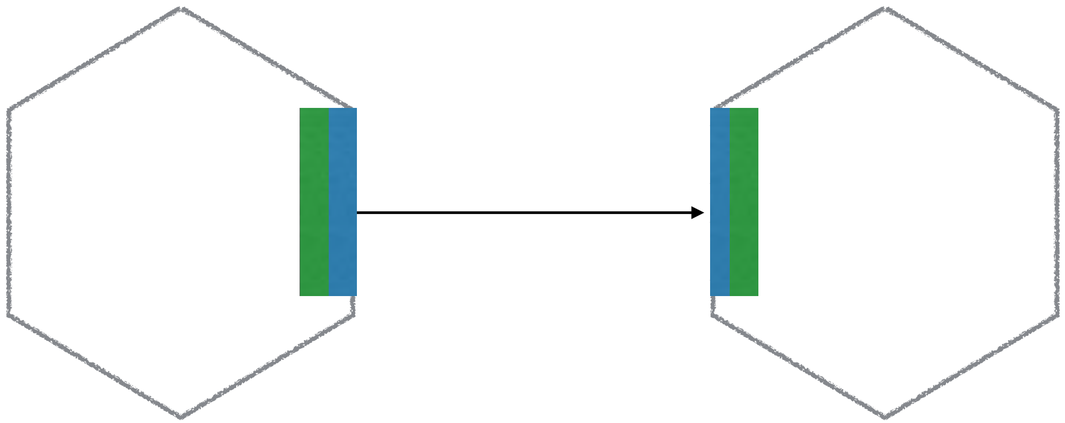
COMMON CONNECTION CONCERNS



Tracing

Load Balancing & Service Discovery

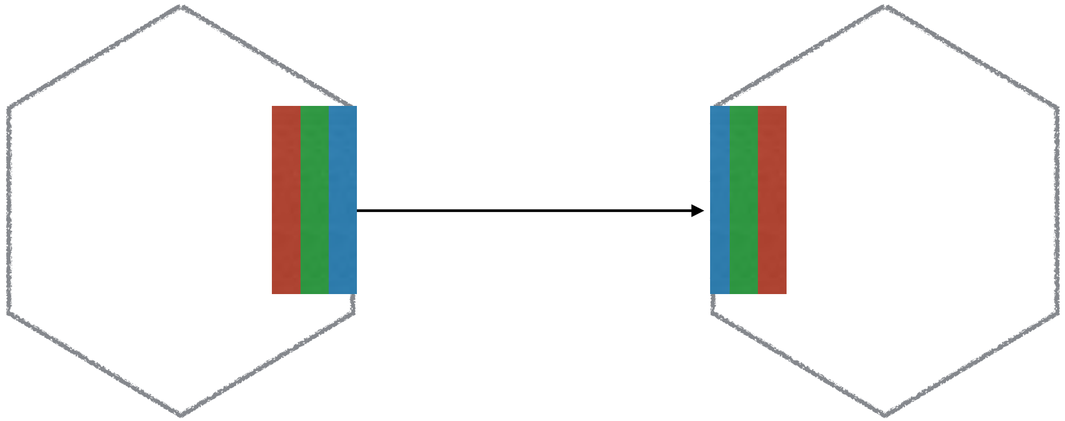
COMMON CONNECTION CONCERNS



Tracing

Load Balancing & Service Discovery

COMMON CONNECTION CONCERNS

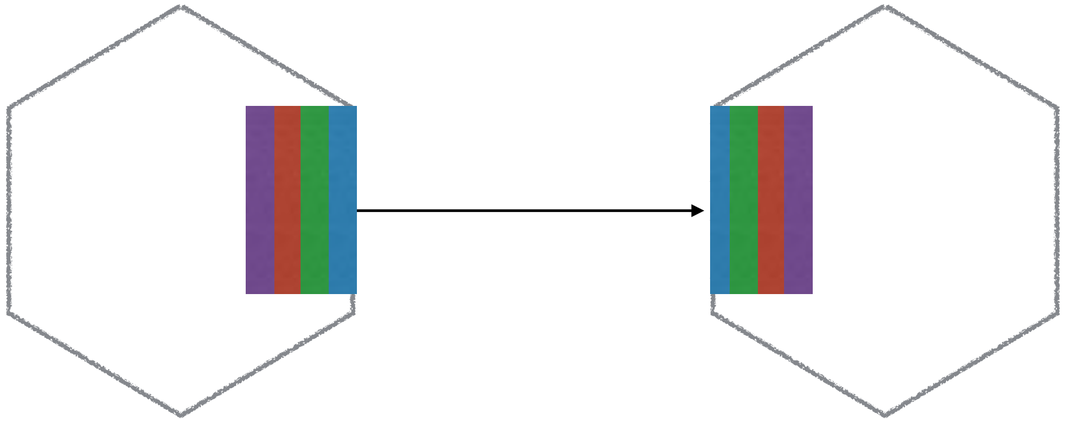


Tracing

Load Balancing & Service Discovery

Authorisation & Authentication

COMMON CONNECTION CONCERNS



Tracing

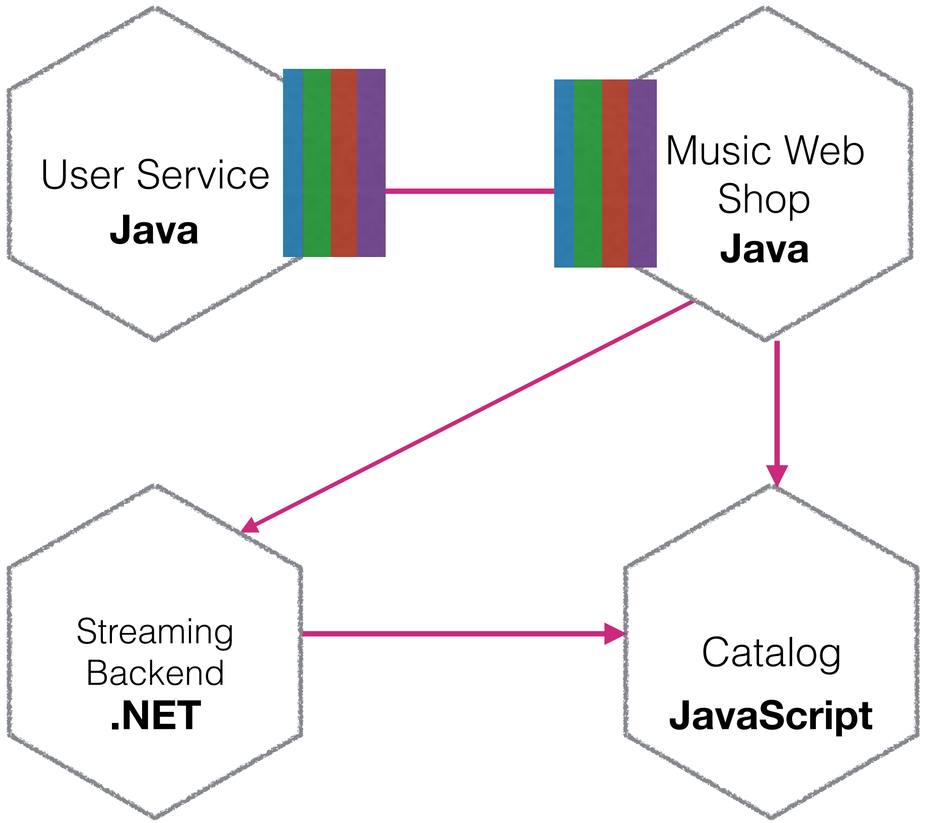
Load Balancing & Service Discovery

Authorisation & Authentication

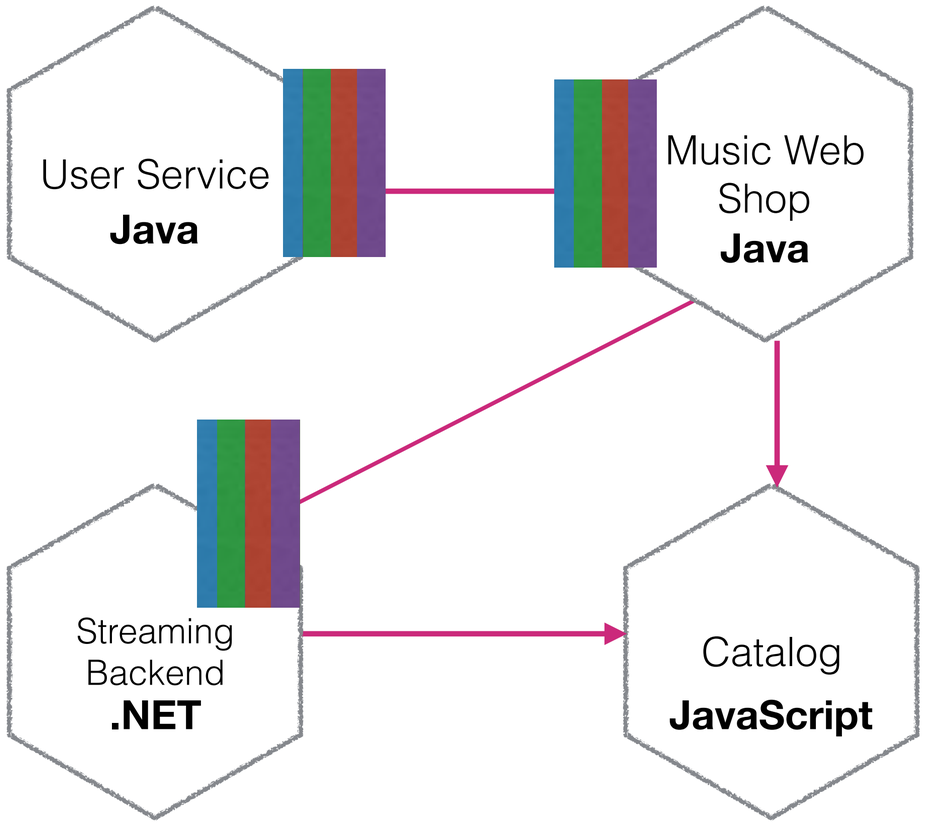
Connection Resilience & Retry

COMMON MICROSERVICE FRAMEWORKS

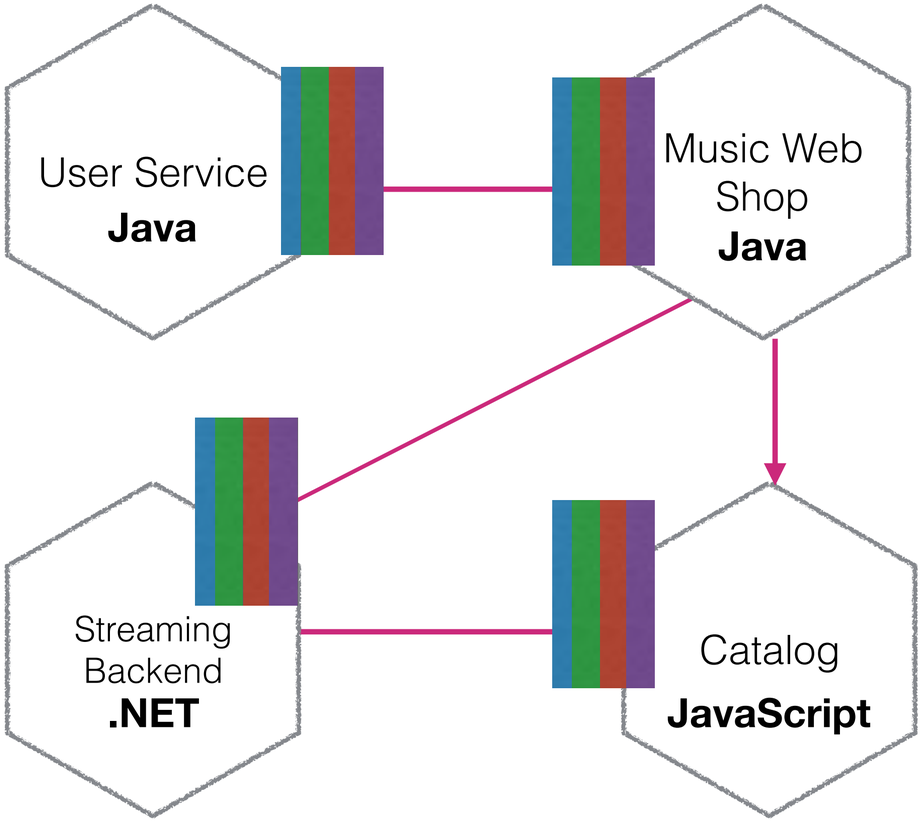
POLYGLOT?


User Service
Java
Music Web Shop
Java
Streaming Backend
.NET
Catalog
JavaScript

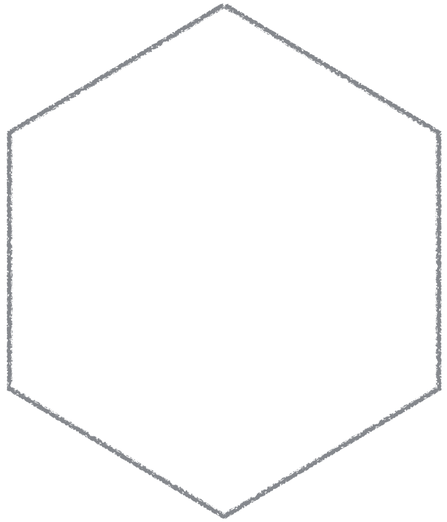
POLYGLOT?

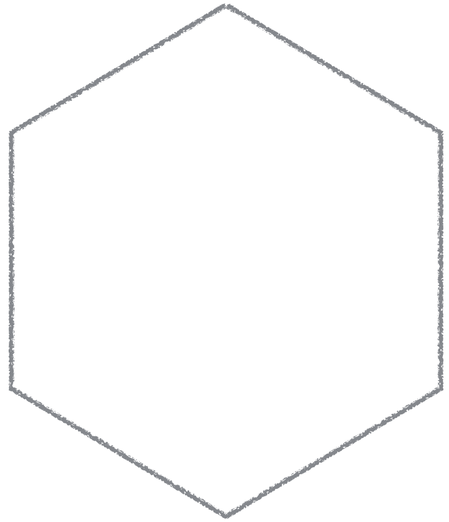

User Service
Java
Music Web Shop
Java
Streaming Backend
.NET
Catalog
JavaScript

POLYGLOT?


User Service
Java
Music Web
Shop
Java
Streaming
Backend
.NET
Catalog
JavaScript

VERSION DRIFT

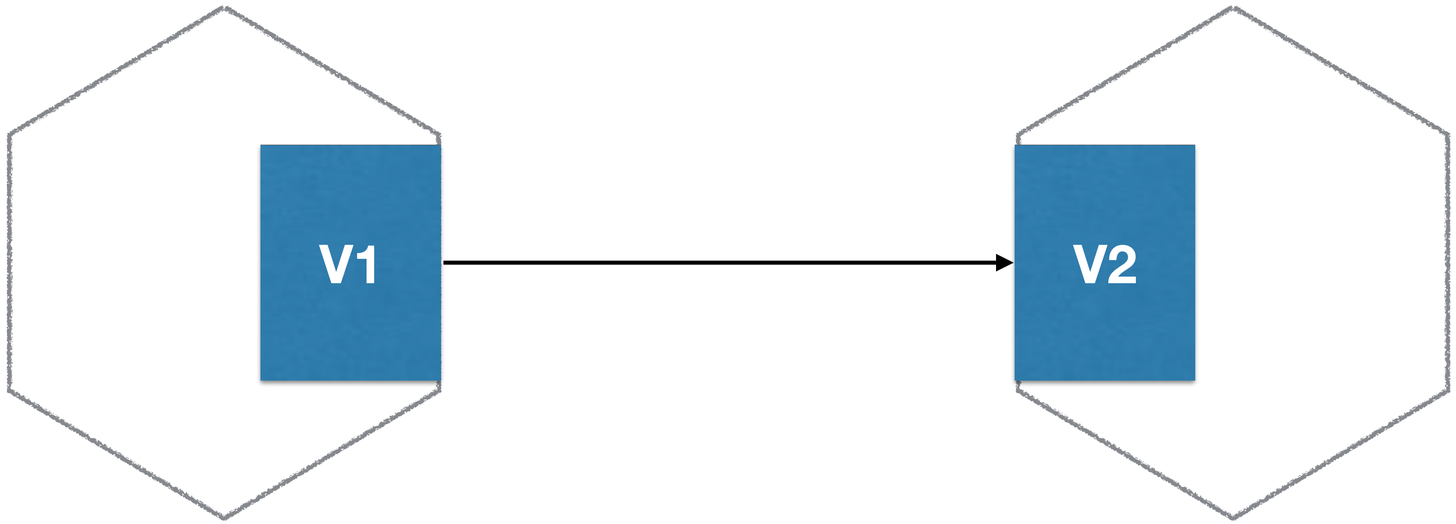




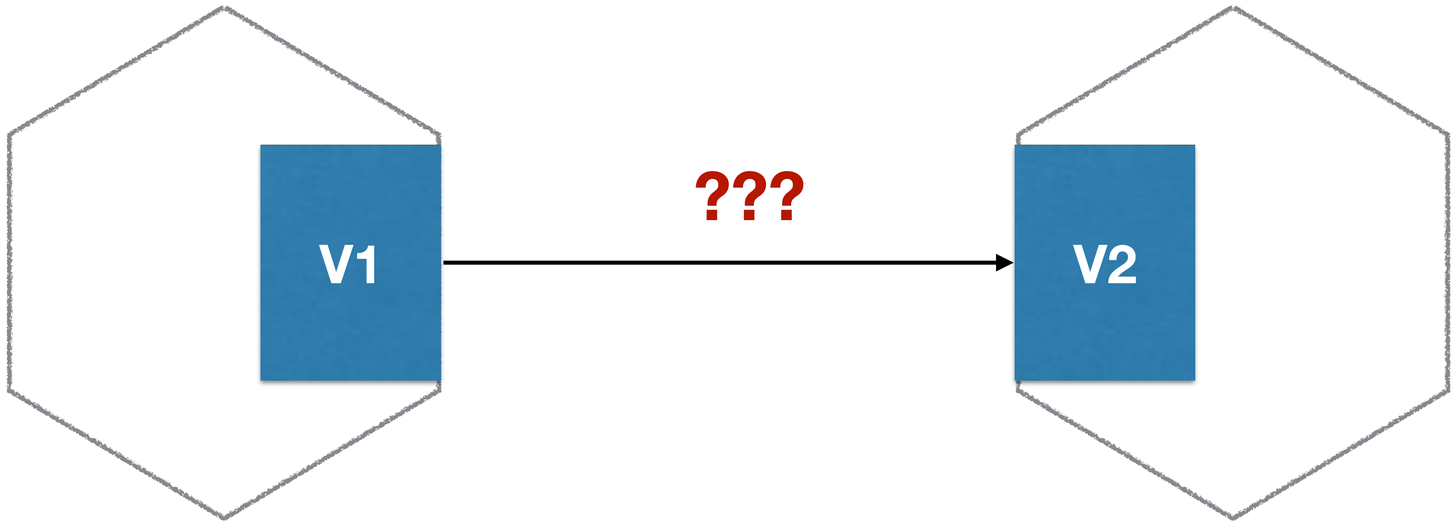
VERSION DRIFT

VERSION DRIFT

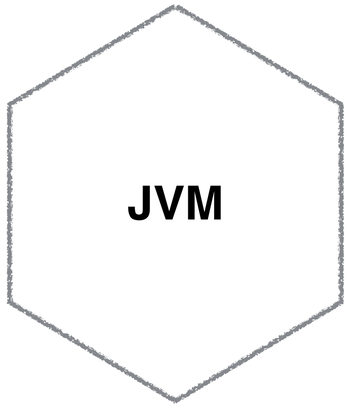
VERSION DRIFT

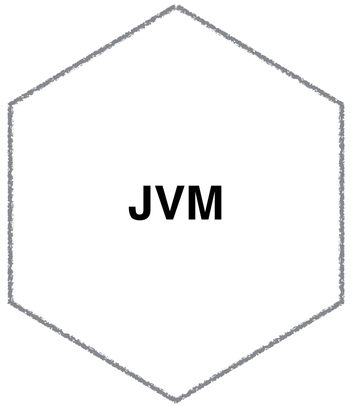


VERSION DRIFT



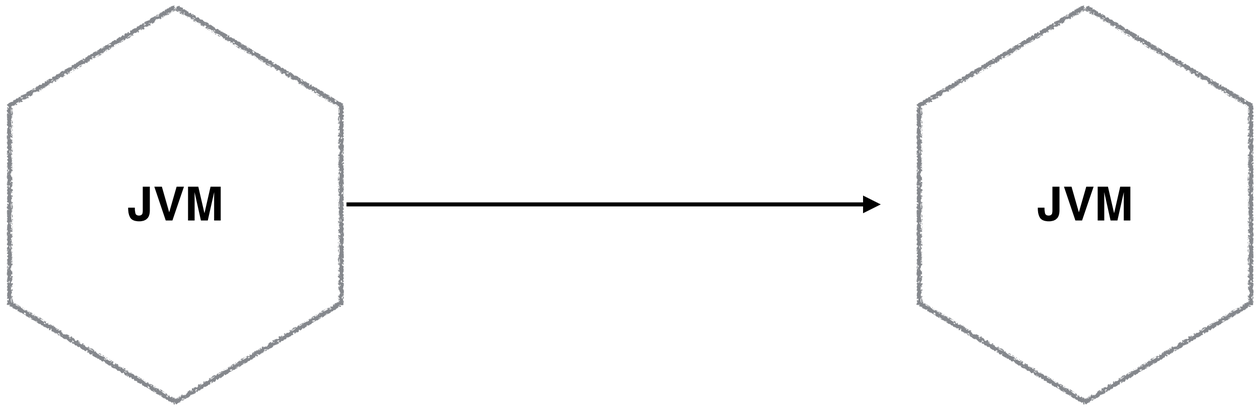
NETFLIX - ENFORCEMENT OF REUSE





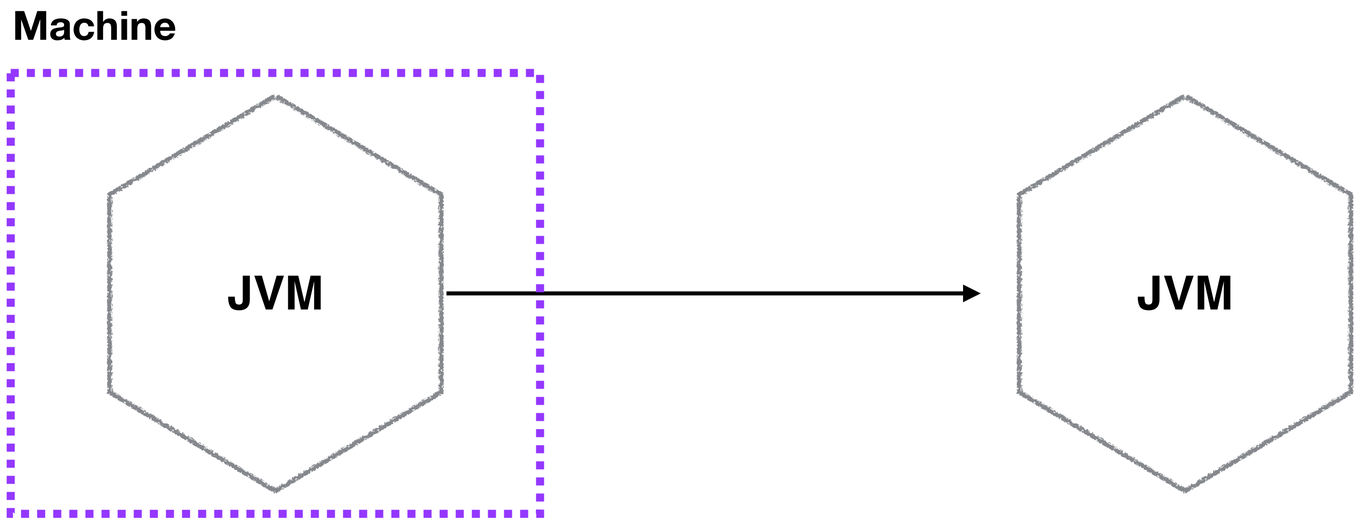
NETFLIX

NETFLIX - ENFORCEMENT OF REUSE



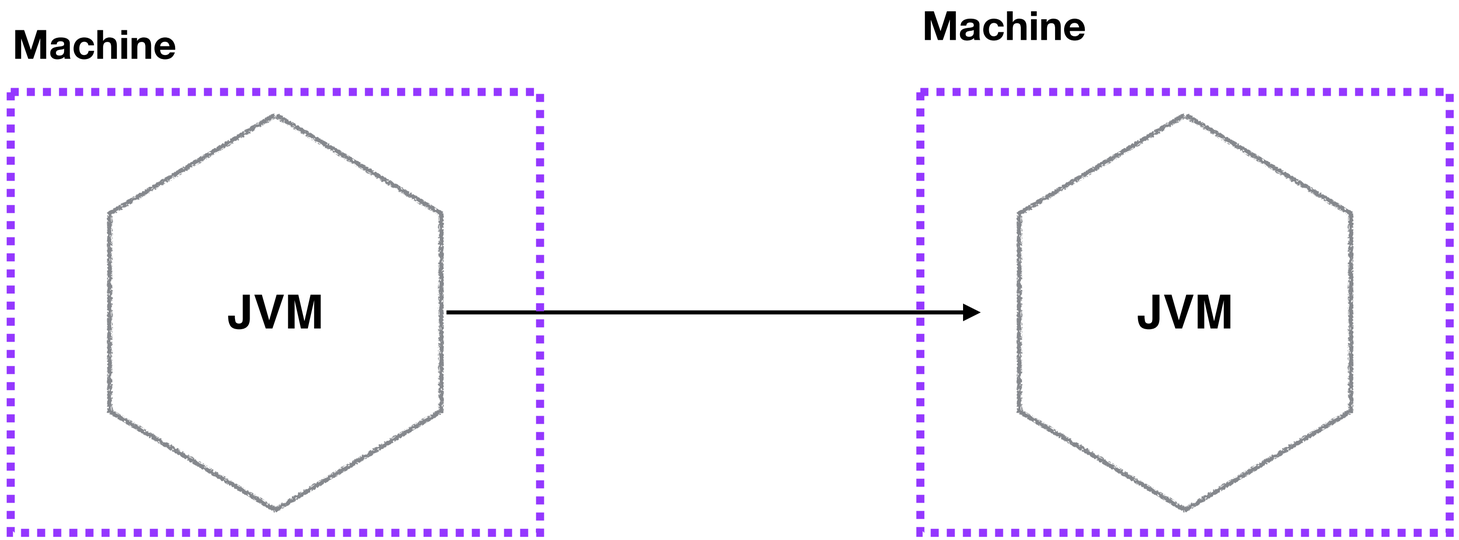
NETFLIX

NETFLIX - ENFORCEMENT OF REUSE



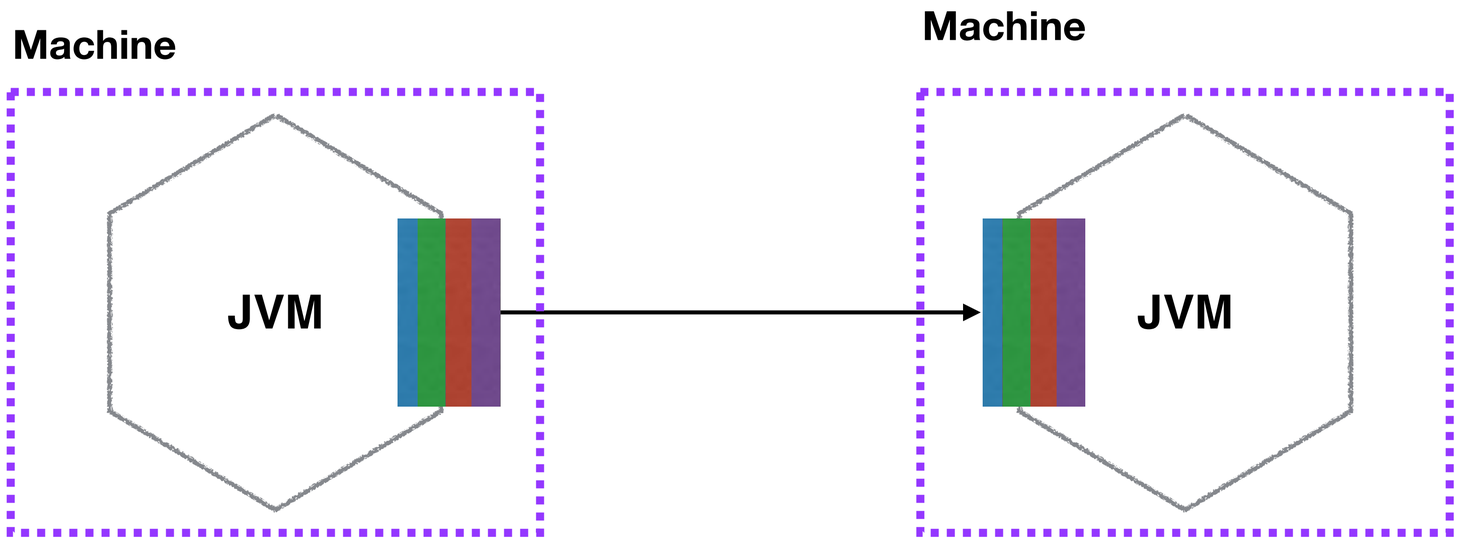
NETFLIX

NETFLIX - ENFORCEMENT OF REUSE



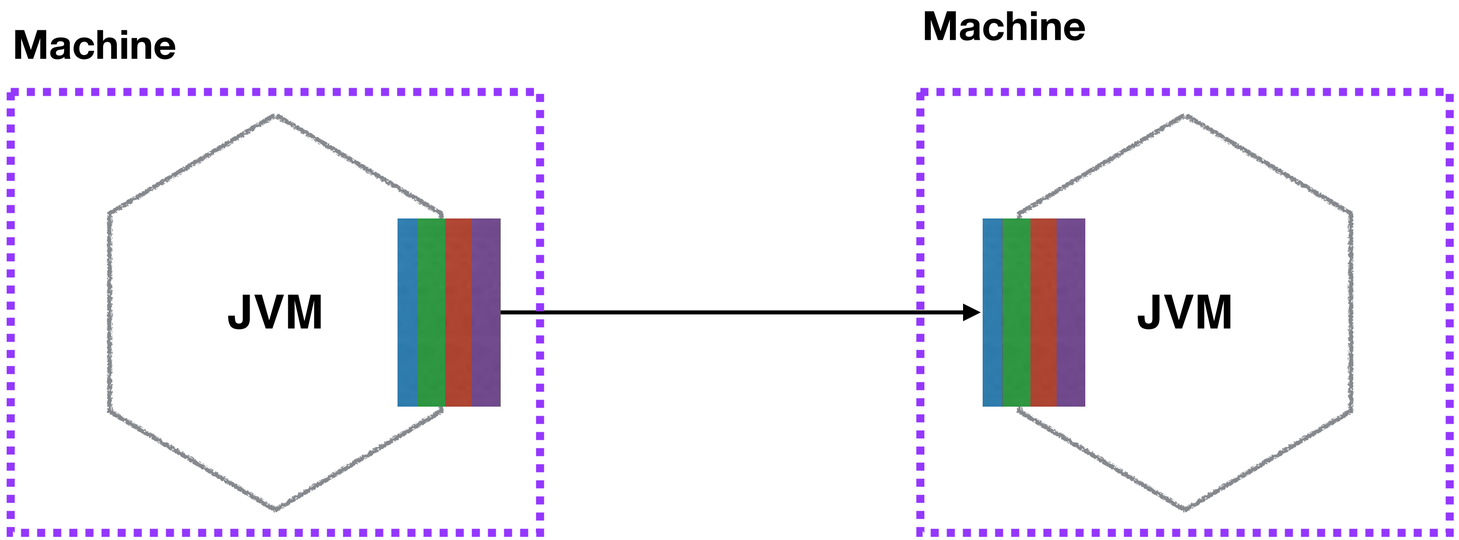
NETFLIX

NETFLIX - ENFORCEMENT OF REUSE



NETFLIX

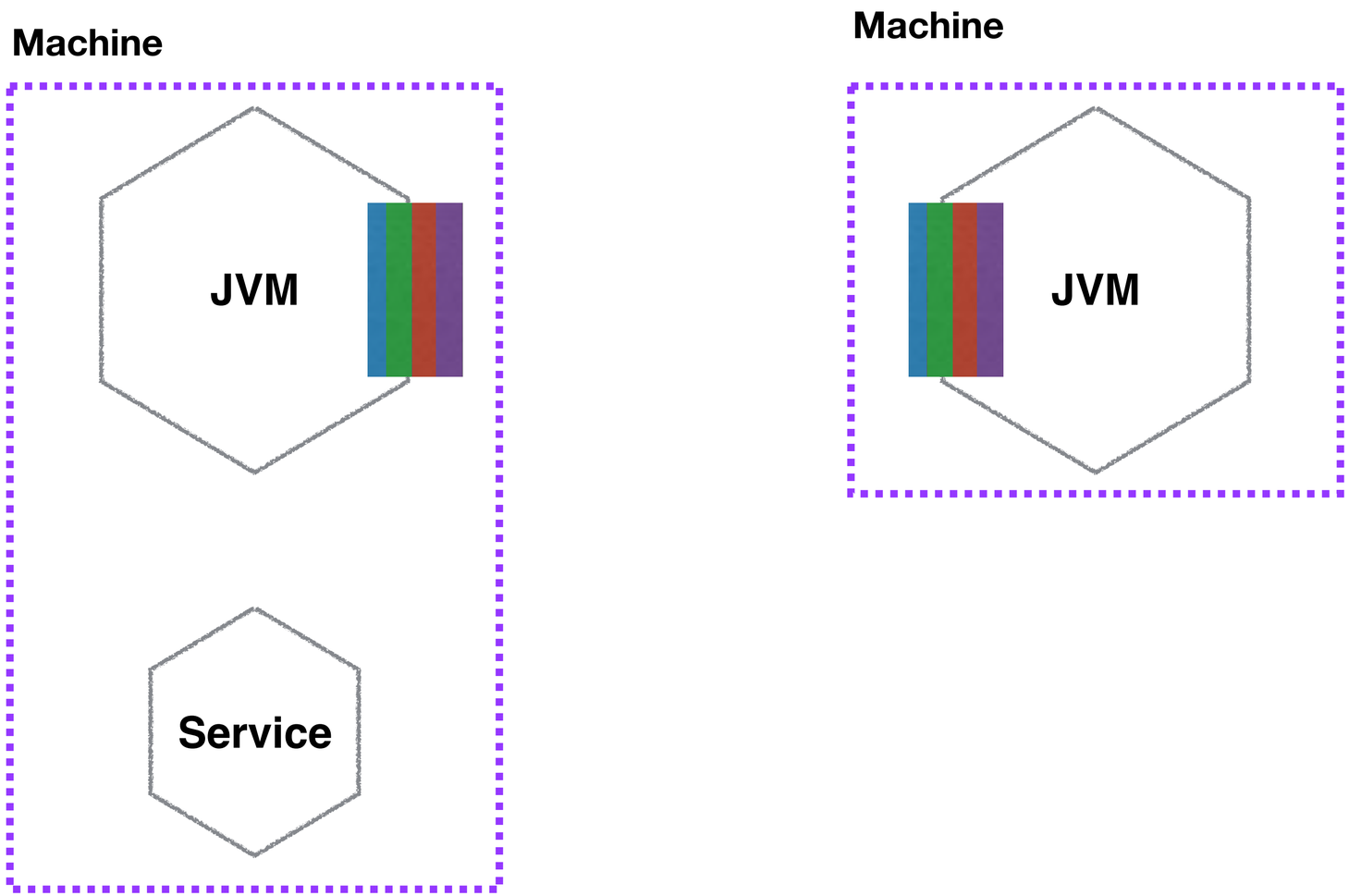
NETFLIX - ENFORCEMENT OF REUSE



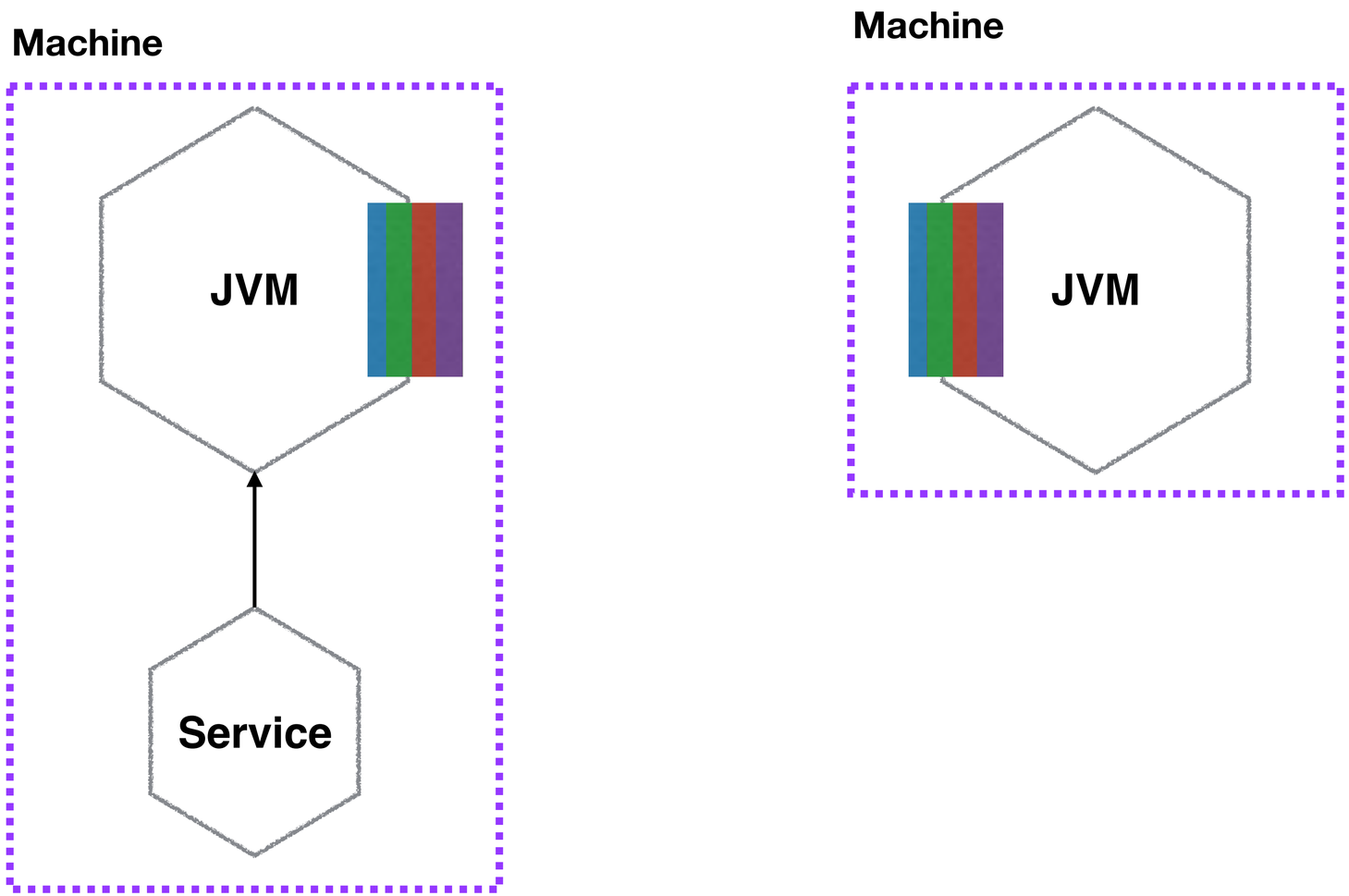
NETFLIX

What about non-JVM languages?

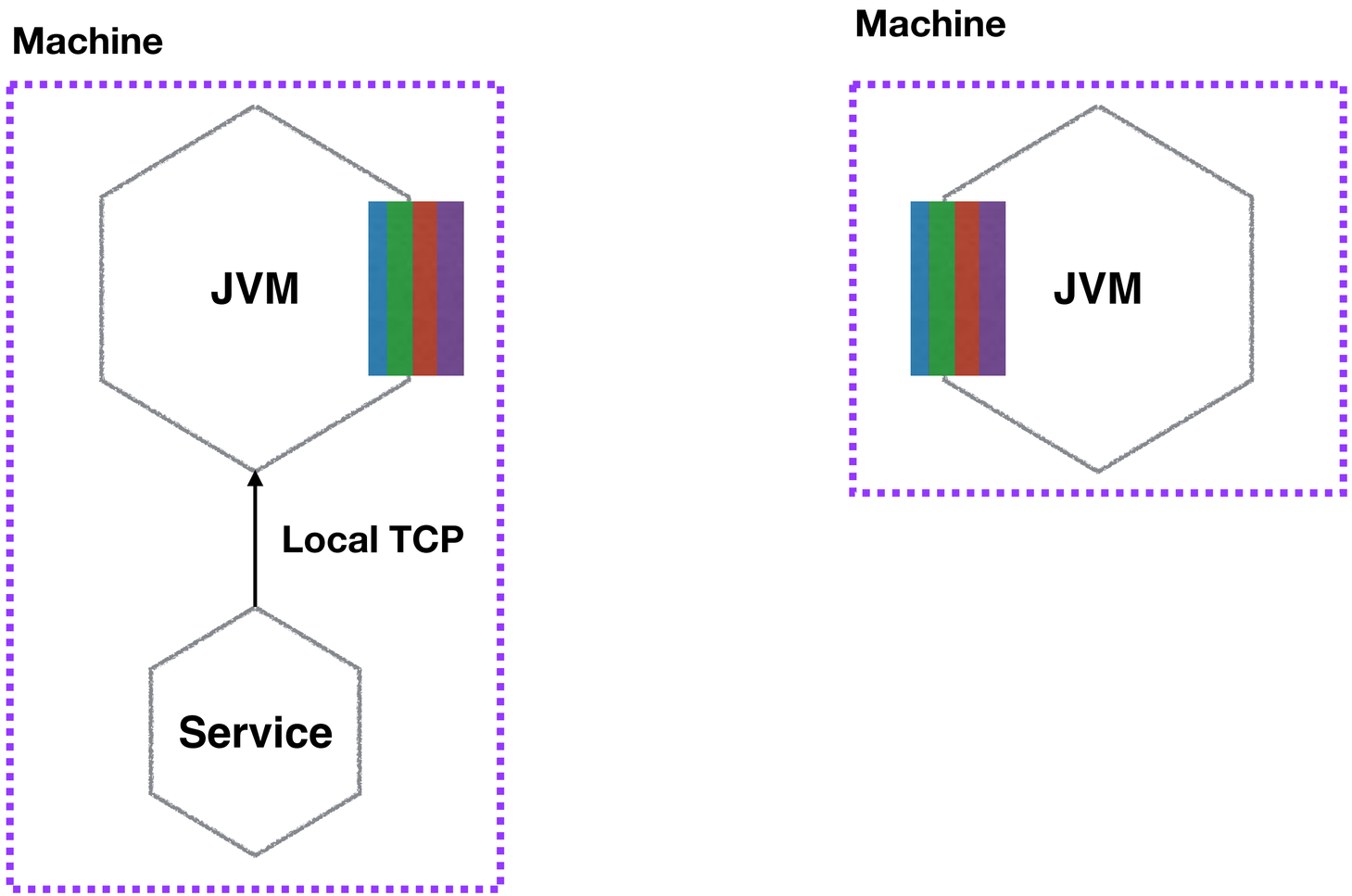
NETFLIX - SIDECAR PATTERN



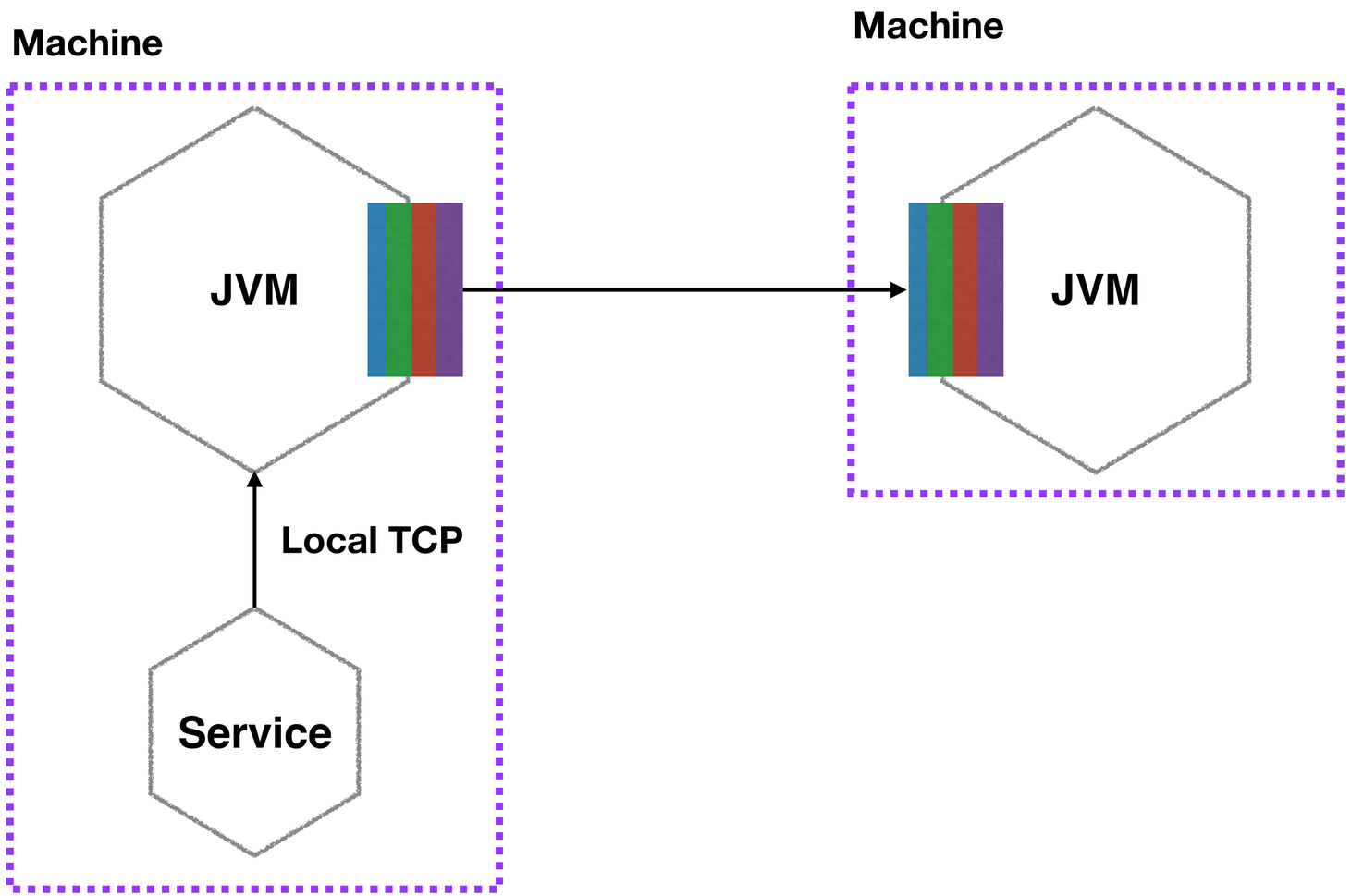
NETFLIX - SIDECAR PATTERN



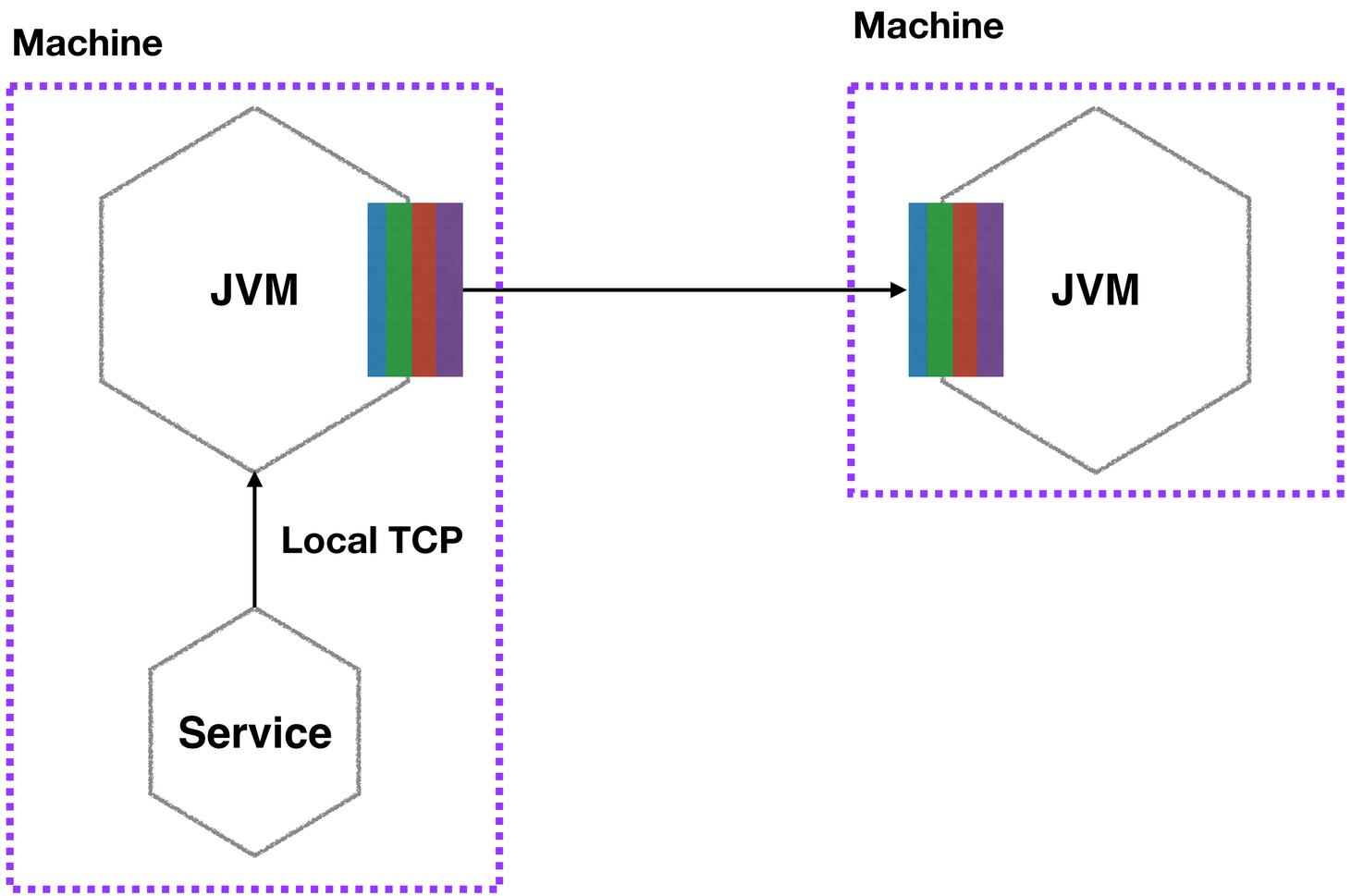
NETFLIX - SIDECAR PATTERN



NETFLIX - SIDECAR PATTERN

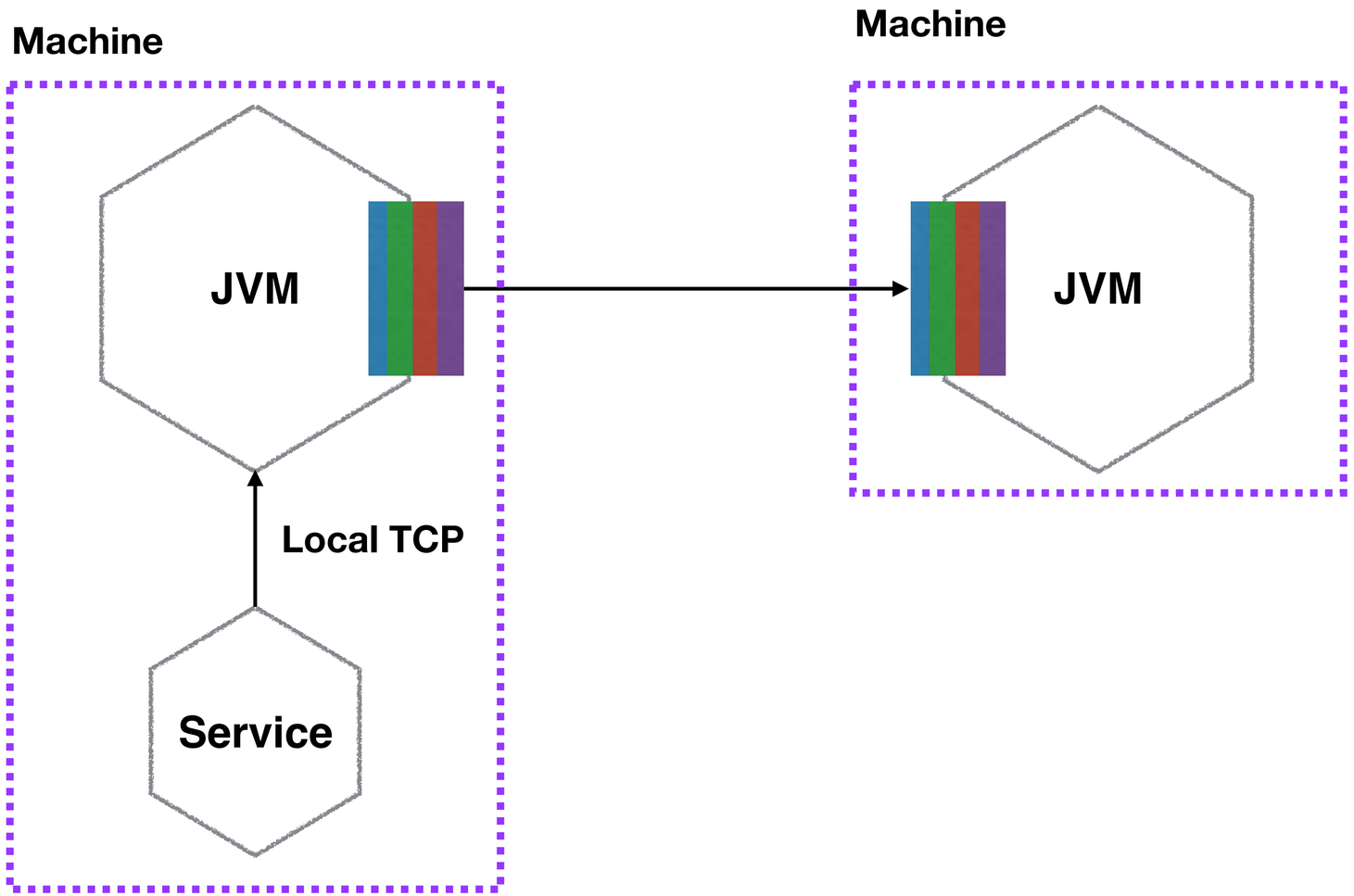


NETFLIX - SIDECAR PATTERN



Re-use code across tech stacks

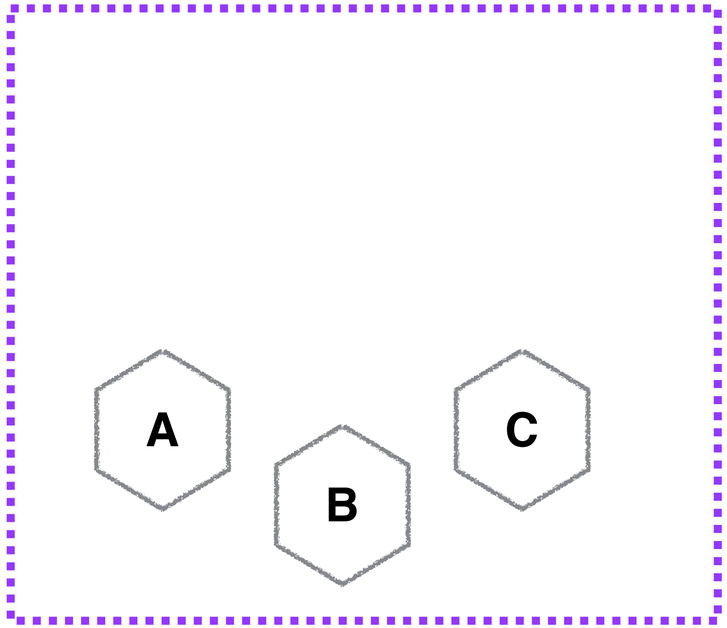
NETFLIX - SIDECAR PATTERN



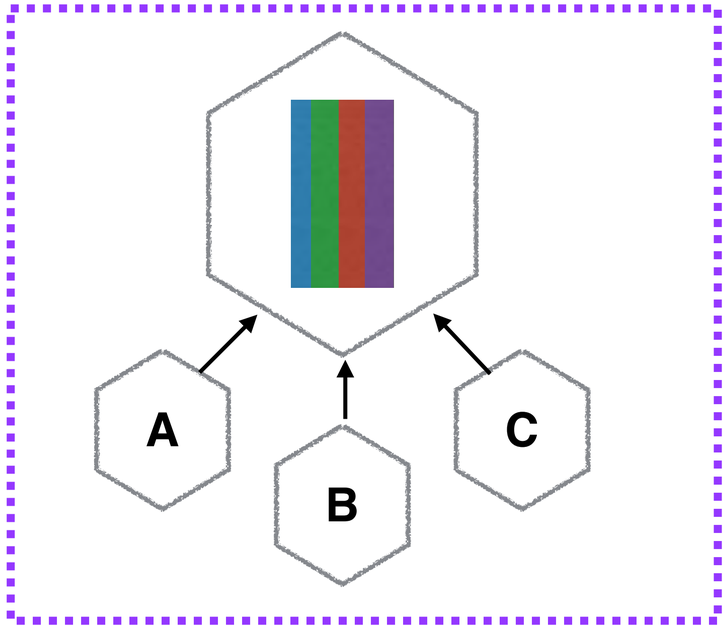
Re-use code across tech stacks

Reduce impact of version drift

FROM PROXIES TO SERVICE MESHES

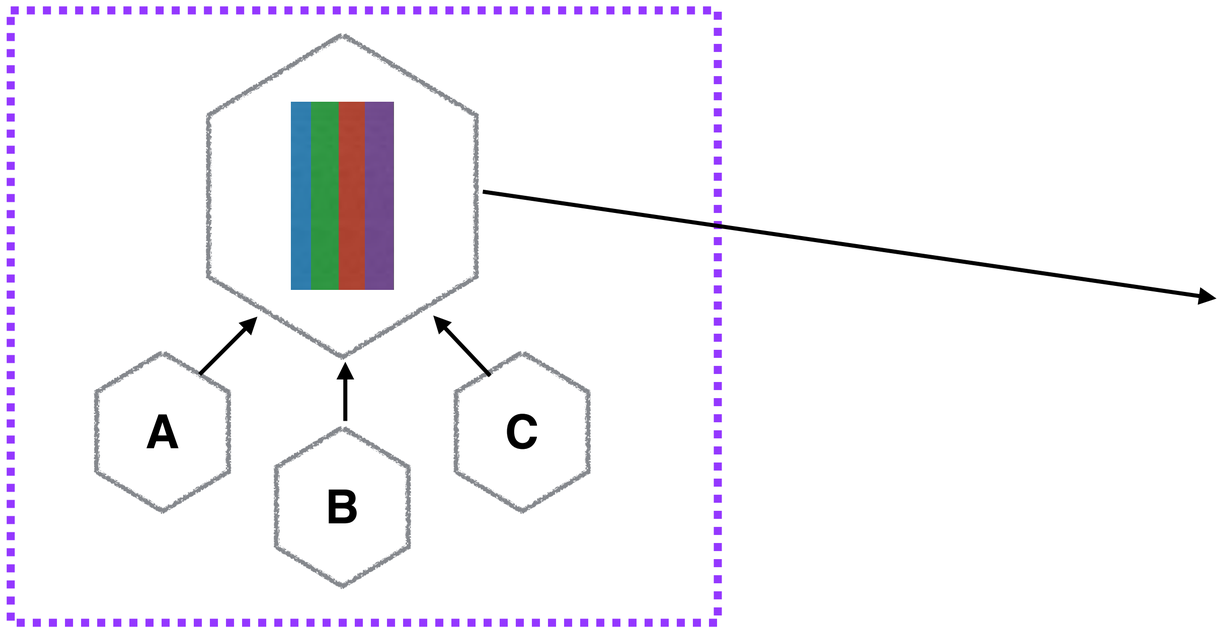


FROM PROXIES TO SERVICE MESHES

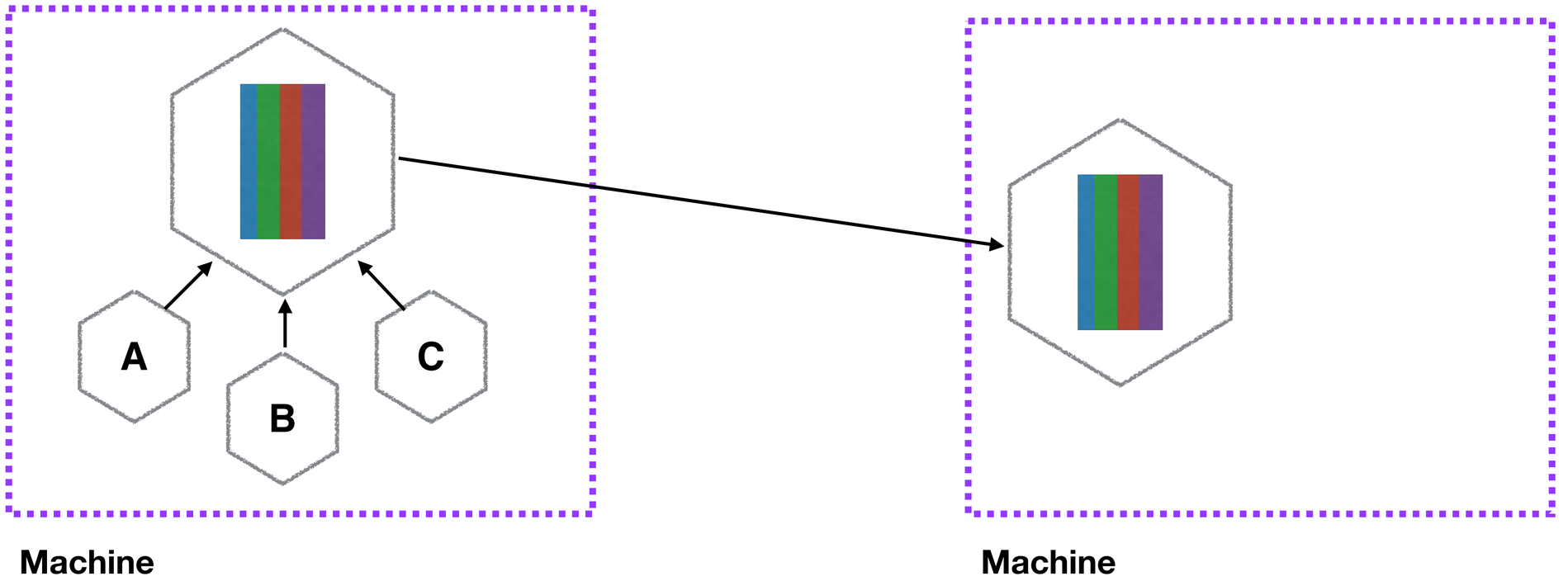


Machine

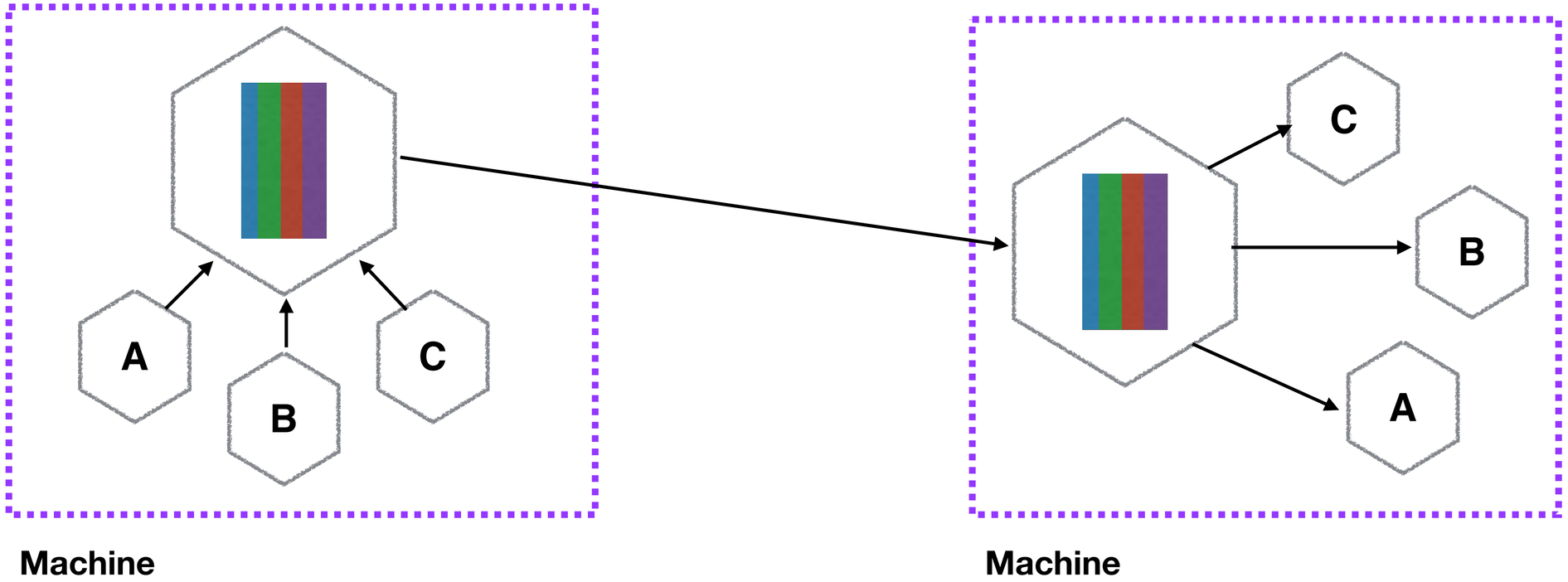
FROM PROXIES TO SERVICE MESHES



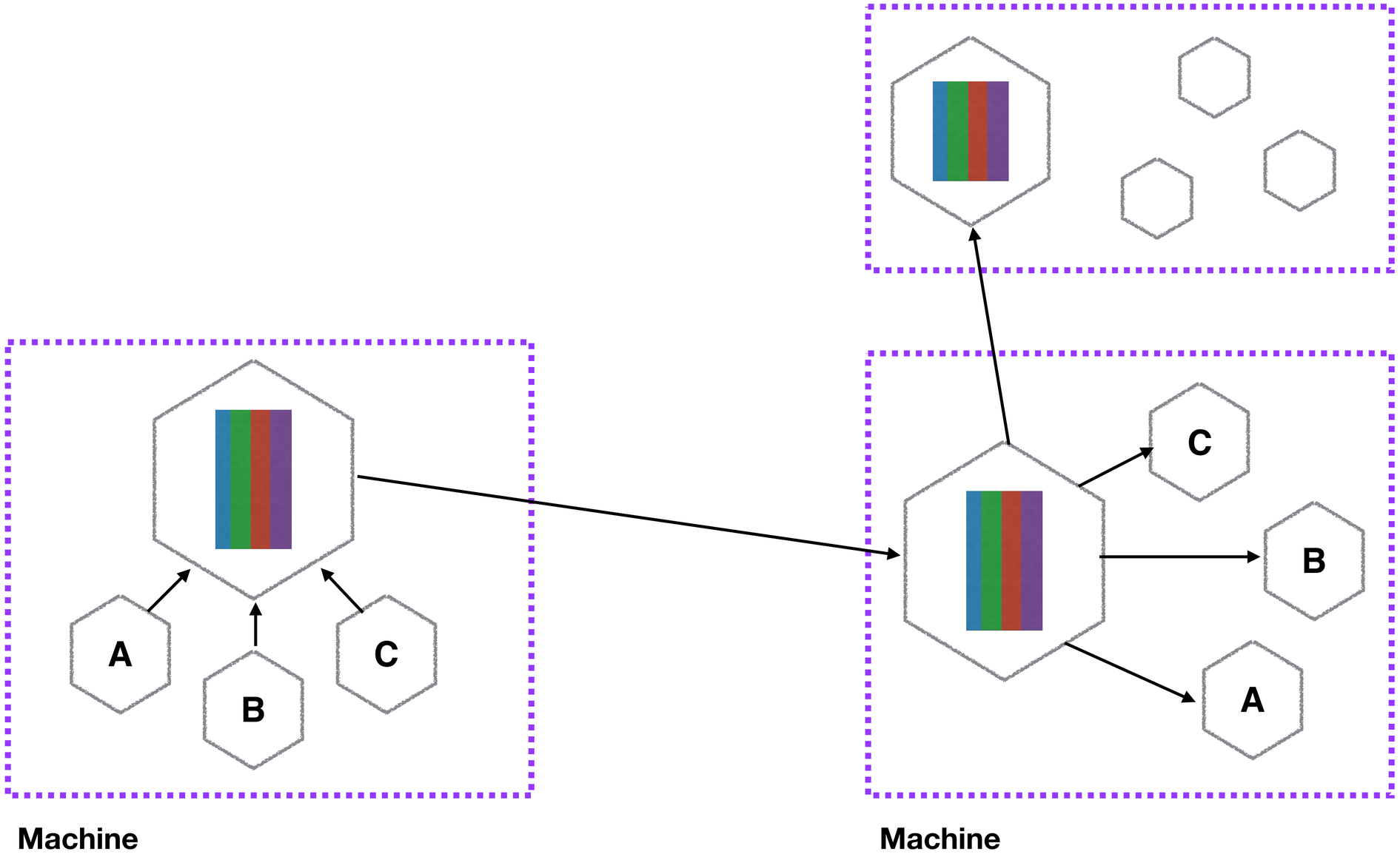
FROM PROXIES TO SERVICE MESHES



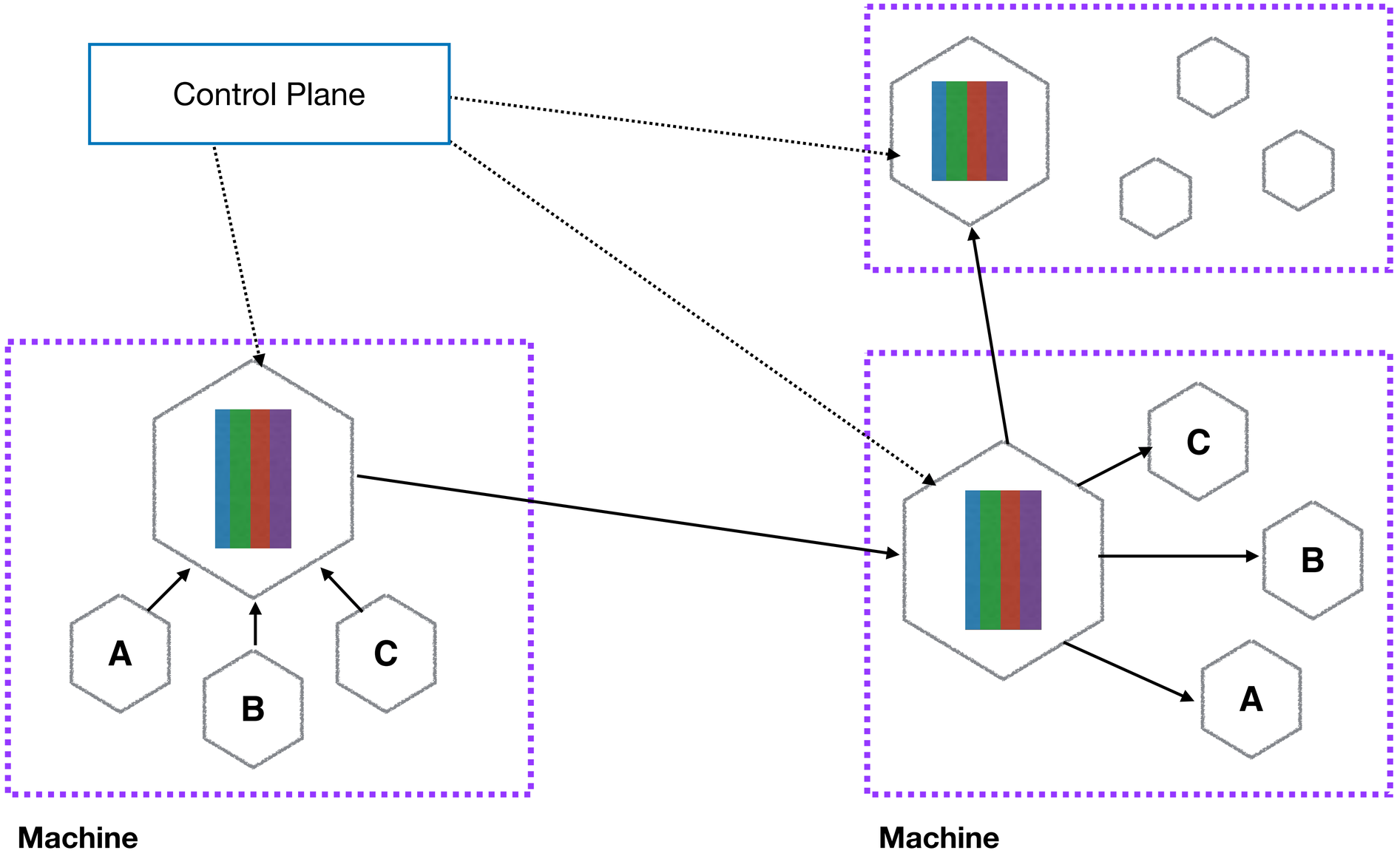
FROM PROXIES TO SERVICE MESHES



FROM PROXIES TO SERVICE MESHES



FROM PROXIES TO SERVICE MESHES



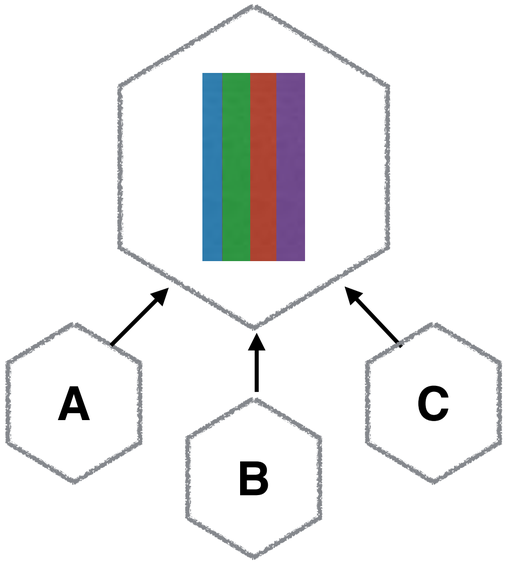
SIDECARS VS PROXIES

Local Proxy

Sidecar

SIDECARS VS PROXIES

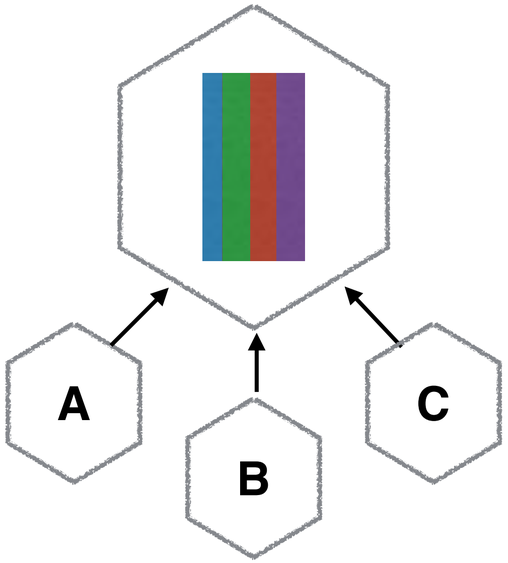
Local Proxy



Sidecar

SIDECARS VS PROXIES

Local Proxy

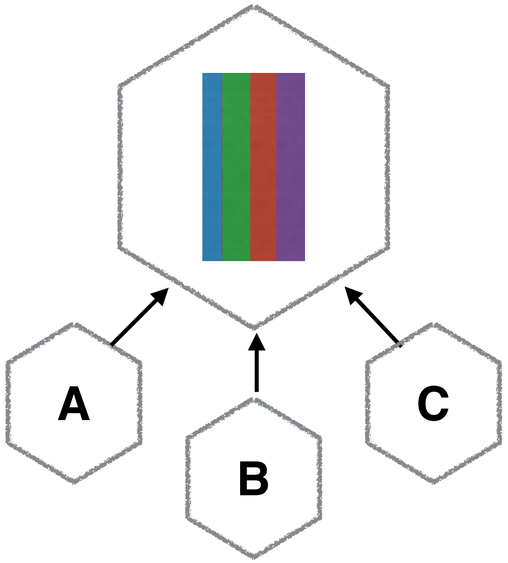


Linkerd

Sidecar

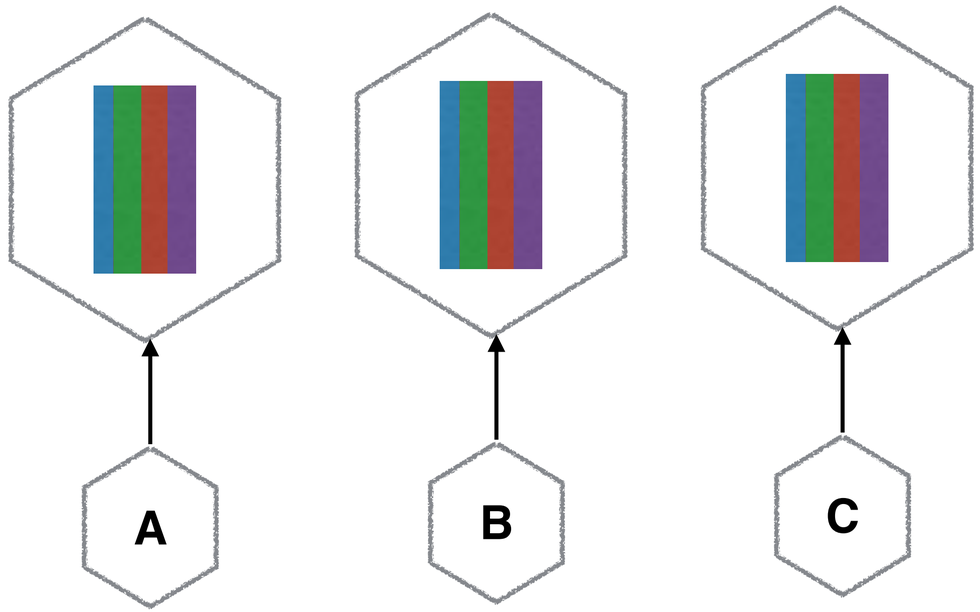
SIDECARS VS PROXIES

Local Proxy



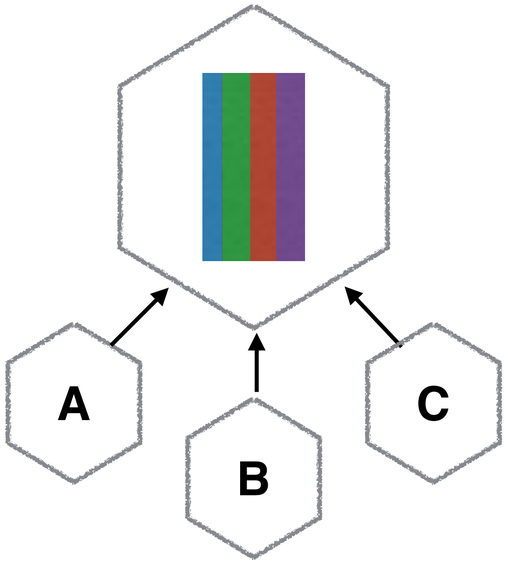
Linkerd

Sidecar



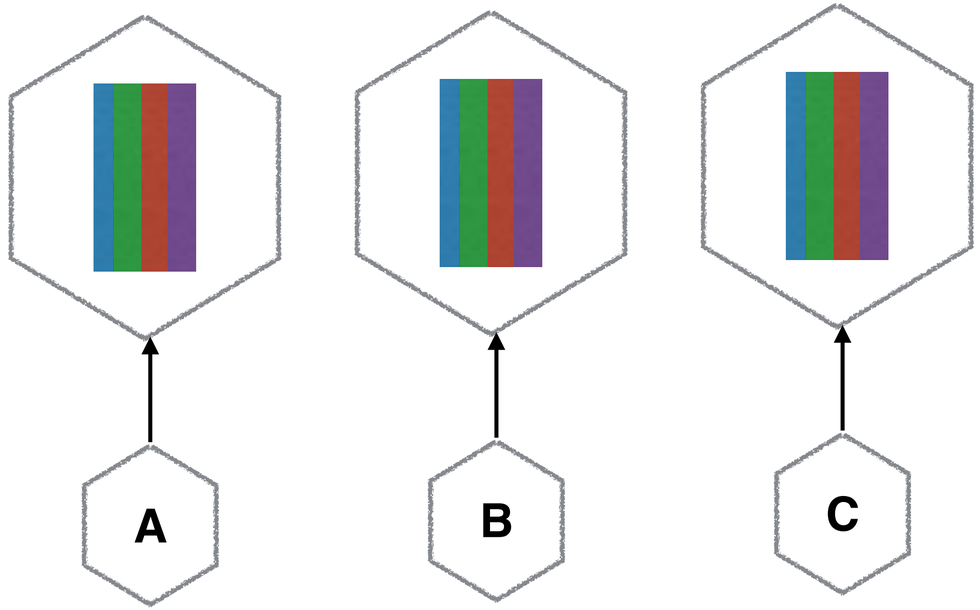
SIDECARS VS PROXIES

Local Proxy



Linkerd

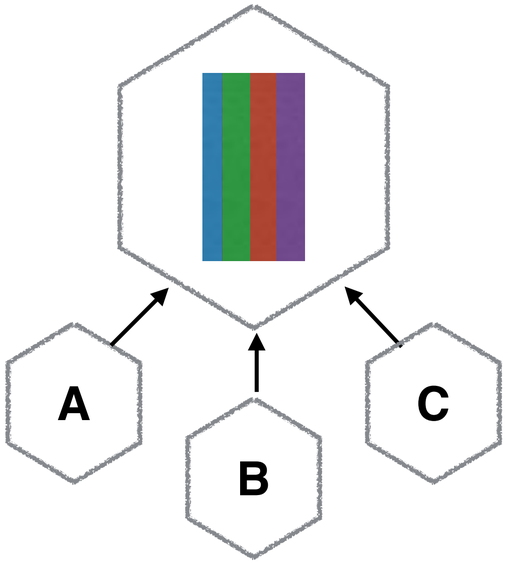
Sidecar



Istio

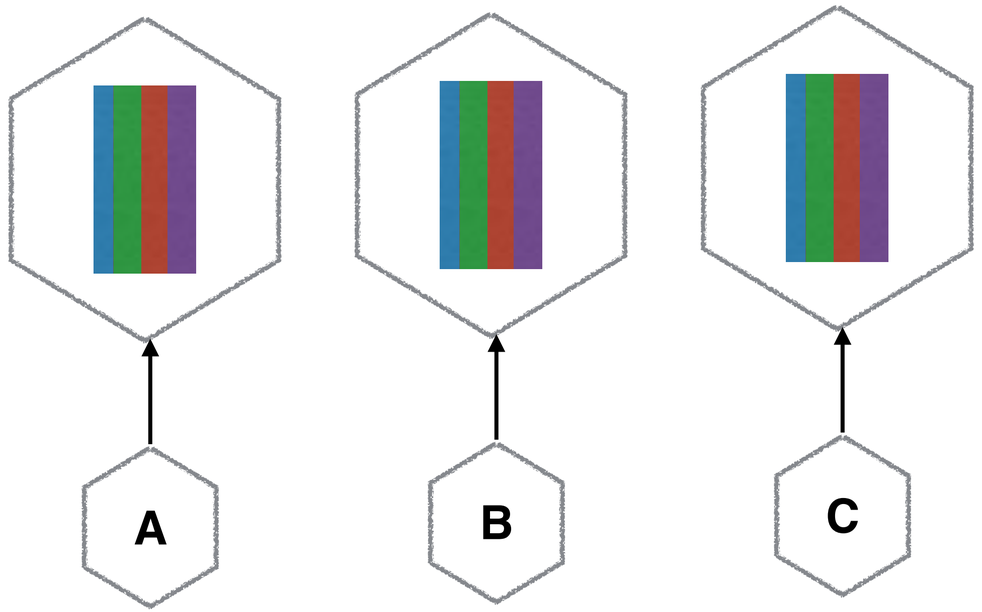
SIDECARS VS PROXIES

Local Proxy



Linkerd

Sidecar



Istio

CONDUIT

SERVICE MESH CAPABILITIES

SERVICE MESH CAPABILITIES

Load balancing

SERVICE MESH CAPABILITIES

Load balancing

Traffic Routing (blue/green deploys, canaries)

SERVICE MESH CAPABILITIES

Load balancing

Traffic Routing (blue/green deploys, canaries)

Service discovery

SERVICE MESH CAPABILITIES

Load balancing

Traffic Routing (blue/green deploys, canaries)

Service discovery

Tracing

SERVICE MESH CAPABILITIES

Load balancing

Traffic Routing (blue/green deploys, canaries)

Service discovery

Tracing

Security!

MUTUAL TLS

Mutual TLS Authentication

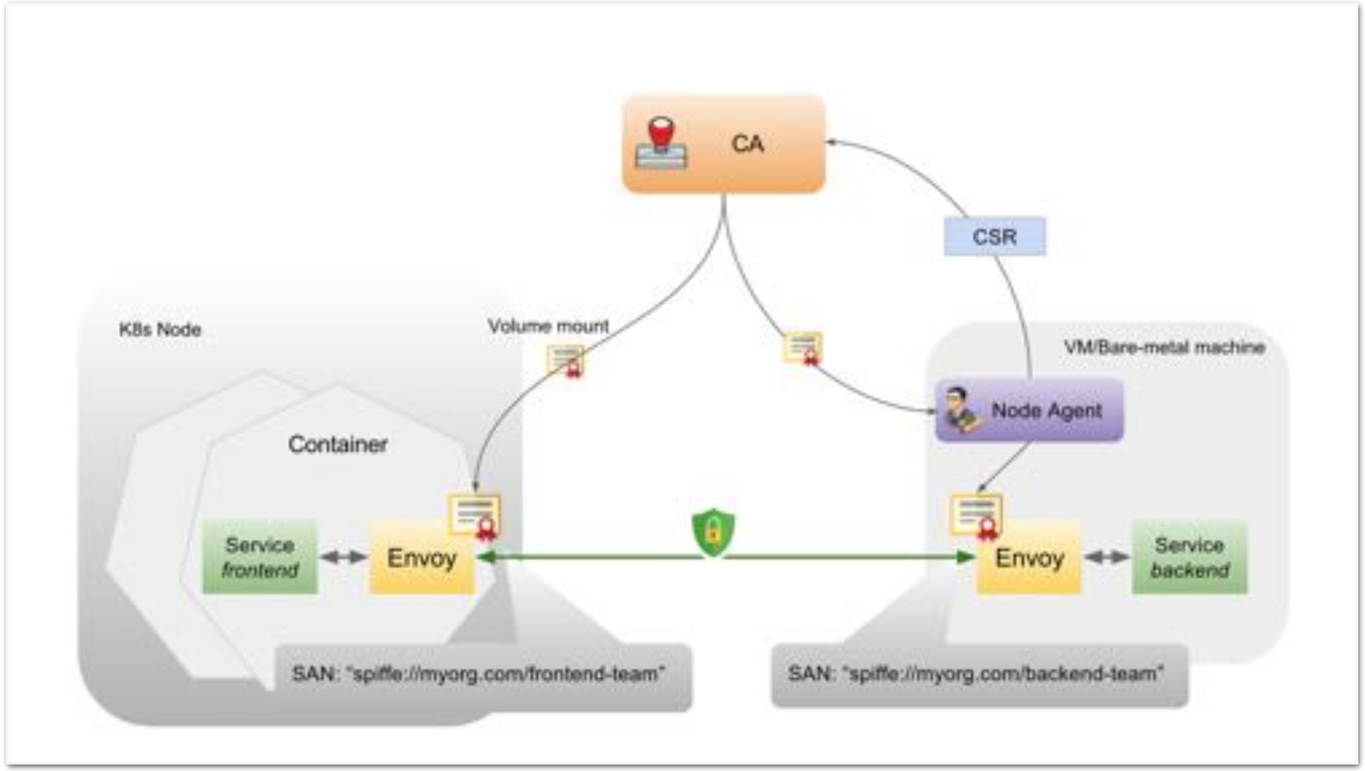
Overview

Istio Auth's aim is to enhance the security of microservices and their communication without requiring service code changes. It is responsible for:

- Providing each service with a strong identity that represents its role to enable interoperability across clusters and clouds
- Securing service to service communication and end-user to service communication
- Providing a key management system to automate key and certificate generation, distribution, rotation, and revocation

Architecture

The diagram below shows Istio Auth's architecture, which includes three primary components: identity, key management, and communication security. This diagram describes how Istio Auth is used to secure the service-to-service communication between service 'frontend' running as the service account 'frontend-team' and service 'backend' running as the service account 'backend-team'. Istio supports services running on both Kubernetes containers and VM/bare-metal machines.



https://istio.io/docs/concepts/security/mutual-tls.html

Caution warranted?

SUMMARY

SUMMARY

Patching & Passwords

SUMMARY

Patching & Passwords

Storing Secrets

SUMMARY

Patching & Passwords

Storing Secrets

Transport Security

SUMMARY

Patching & Passwords

Storing Secrets

Transport Security

Authorisation

SUMMARY

Patching & Passwords

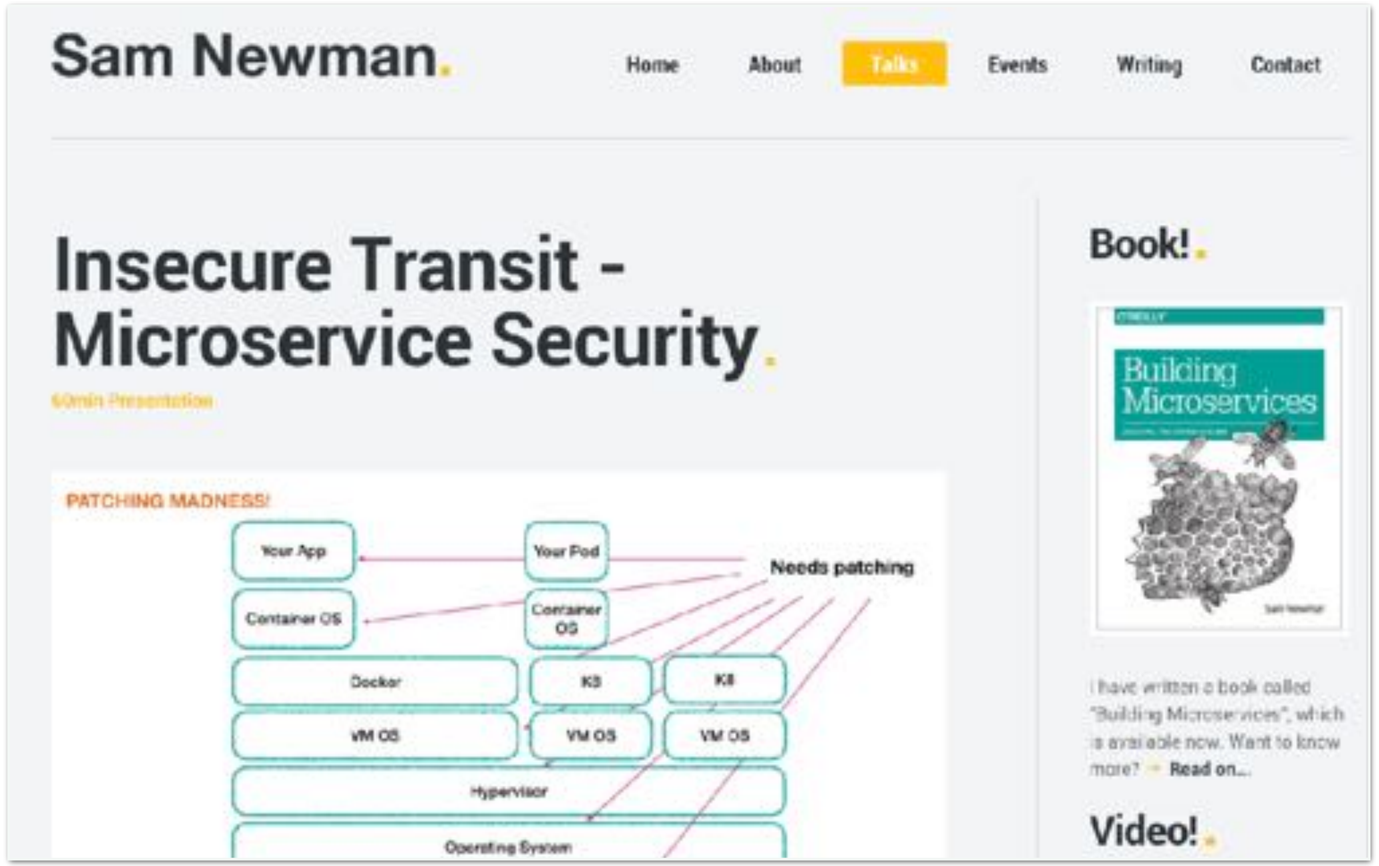
Storing Secrets

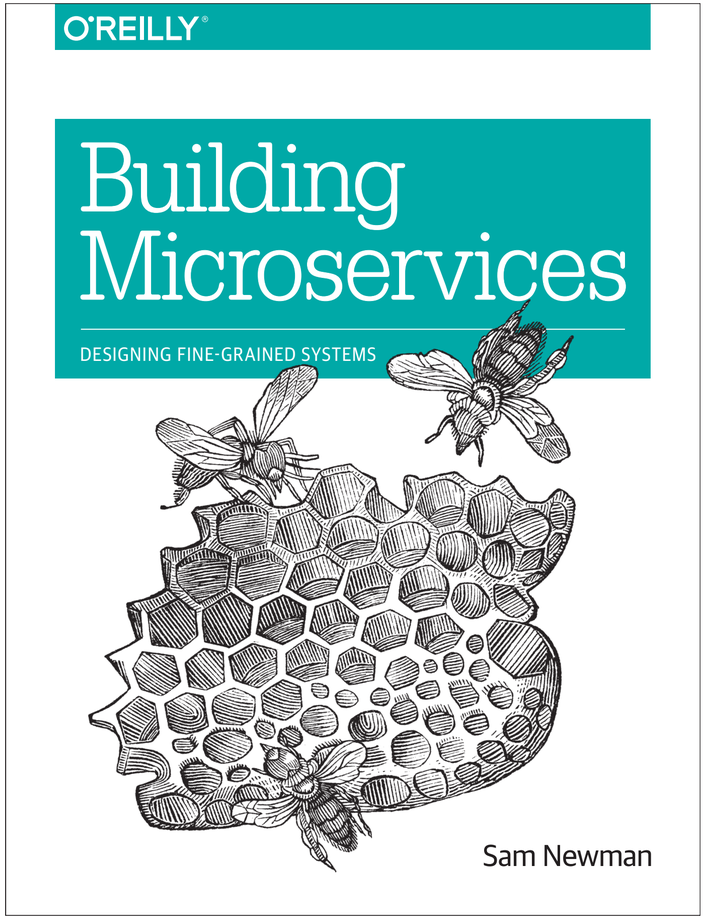
Transport Security

Authorisation

Service Meshes

THANKS!





http://samnewman.io/

@samnewman