Microservices Architecture Design Patterns

Haim Michael

January 3rd, 2023

www.lifemichael.com

life michael

Introduction

Design Pattern?

“Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice”

Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King and Shlomo Angel. *A Pattern Language*. Oxford University Press, New York, 1977.

Software Design Pattern?

"Software patterns are reusable solutions to recurring problems that we encounter during software development."

Mark Grand, *Patterns in Java*. John Wiley & Sons, 2002.

History

- The idea of using patterns evolves from the architecture field.

  Christopher Alexander. *A Pattern Language: Towns, Buildings, Construction* (Oxford University Press, 1977)

- The first GUI software patterns were set in 1987.

  Ward Cunningham and Kent Beck. *Using Pattern Languages for Object-Oriented Programs*. OOPSLA-87.

- The Gang of Four (AKA GOF) publish their "Design Patterns" book in 1994.

  Erich Gamma, Richard Helm, John Vlissides, and Ralph Johnson. *Design Patterns.* Addison Wesley, 1994.

The Pattern Elements

- Each Pattern has the following elements:

  Pattern Name

  The official well known name we use to refer each one of the patterns.

  Problem Description

  A description of the problem the pattern comes to solve.

  Solution Description

  A description of the design solution. This element can include UML diagram and a code sample.

  Consequences

  The benefits & costs of using a specific design pattern. This element is required in order to evaluate one pattern comparing with others.

Continuous Evolvement

- Apart of the classic design patterns described by the Gang of Four book, during the last 20 years we can identify many more design patterns that evolved in specific domains (e.g. Server Side), specific programming languages paradigms (e.g. Functional Programming), specific architectures (e.g. Microservices Architecture), and specific programming languages (e.g. JavaScript).

Microservices

The Microservices Architecture

- The microservices architecture is a modern variation of the Service-Oriented Architecture (SOA). Instead of developing one monolithic application we develop multiple separated small fine grained applications that provide their service using a lightweight protocol.

Loosely Coupling Services

- Each and every micro-service is independent of the others. This loosely coupling architecture reduced the dependencies between the services.

RESTful Web Services

- Representational state transfer (REST) is a software architectural style that dominates the World Wide Web.
- REST emphasizes the scalability of interactions between components, uniform interfaces, independent deployment of components, and the creation of a layered architecture to facilitate caching components to reduce user-perceived latency, enforce security, and encapsulate legacy systems.

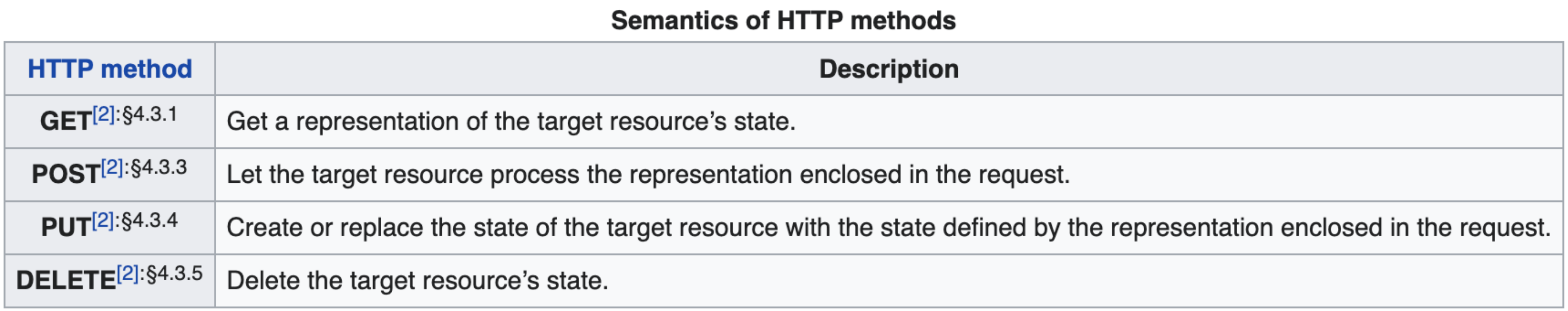
RESTful Web Services

- REST has been employed throughout the software industry and is a widely accepted set of guidelines for creating stateless, reliable web APIs.
- Web APIs that obey the REST constraints is informally described as RESTful.

RESTful Web Services

- RESTful web APIs are typically loosely based on HTTP methods to access resources via URL-encoded parameters and the use of JSON or XML to transmit data.

RESTful Web Services



Semantics of HTTP methods

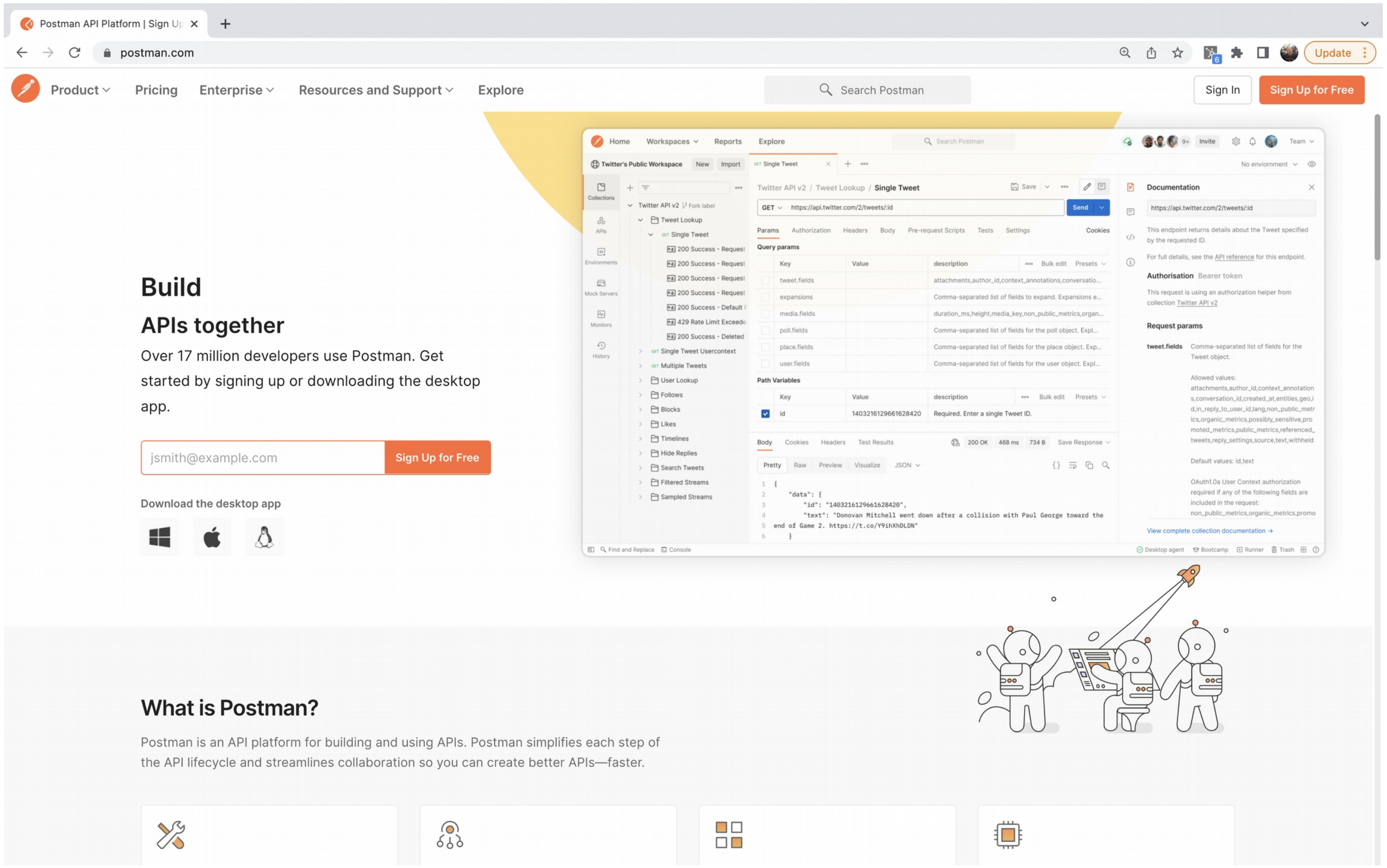
HTTP method	Description
GET[2]:§4.3.1	Get a representation of the target resource's state.
POST[2]:§4.3.3	Let the target resource process the representation enclosed in the request.
PUT[2]:§4.3.4	Create or replace the state of the target resource with the state defined by the representation enclosed in the request.
DELETE[2]:§4.3.5	Delete the target resource's state.

This image was taken from https://en.wikipedia.org/wiki/Representational_state_transfer

The postman Platform

- Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration.
- We can download the Postman application and use it for checking web APIs we want to use.
  https://postman.com

The postman Platform


Postman API Platform | Sign Up

postman.com

Product Pricing Enterprise Resources and Support Explore

Search Postman

Sign In Sign Up for Free

Build

APIs together

Over 17 million developers use Postman. Get started by signing up or downloading the desktop app.

jsmith@example.com Sign Up for Free

Download the desktop app

What is Postman?

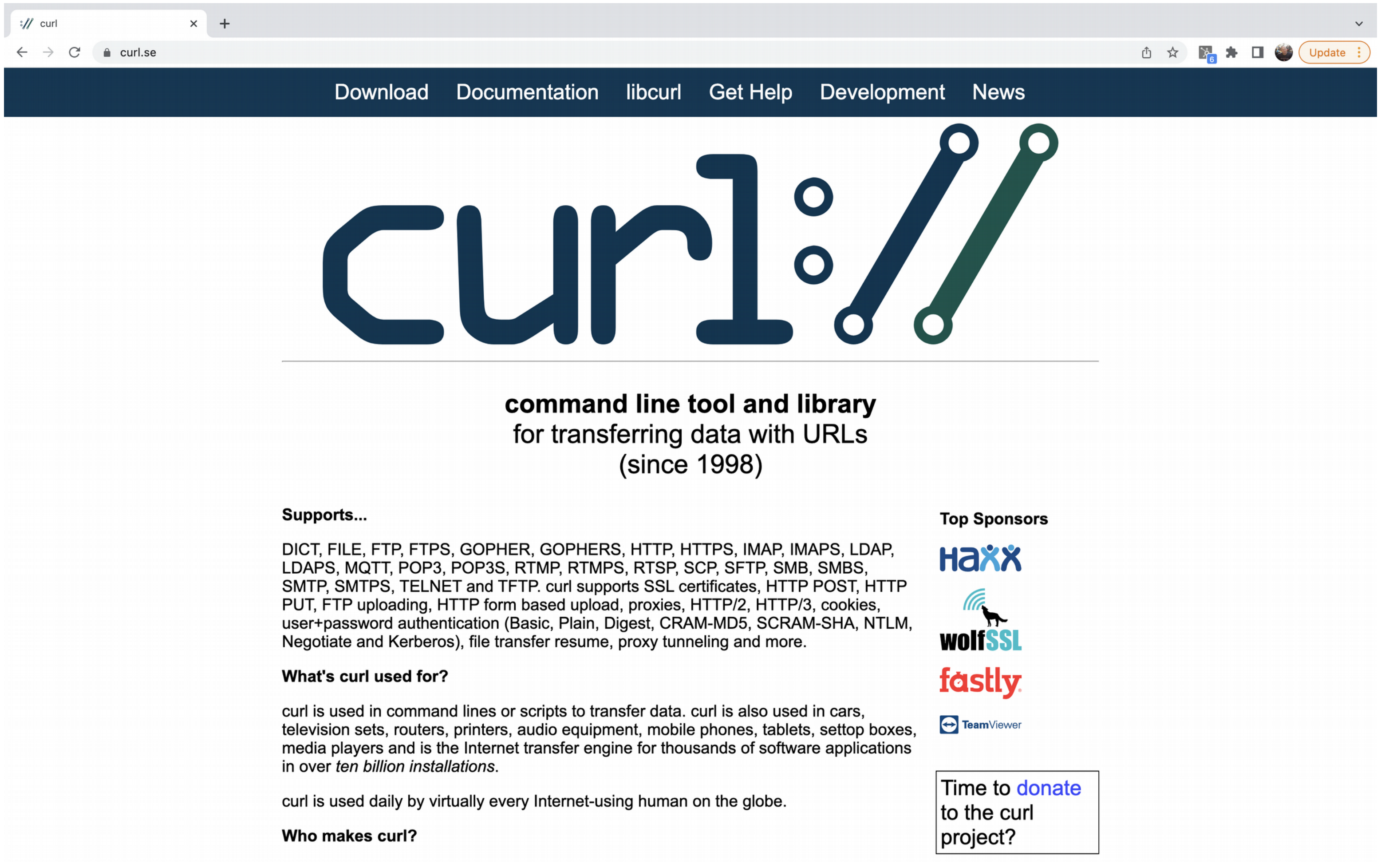
Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster.

The curl Utility

- The curl utility is a command line tool and library for transferring data with URLs.

  https://curl.se

The curl Utility



Download Documentation libcurl Get Help Development News

curl://

command line tool and library
for transferring data with URLs
(since 1998)

Supports...

DICT, FILE, FTP, FTPS, GOPHER, GOPHERS, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, MQTT, POP3, POP3S, RTMP, RTMPS, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET and TFTP. curl supports SSL certificates, HTTP POST, HTTP PUT, FTP uploading, HTTP form based upload, proxies, HTTP/2, HTTP/3, cookies, user+password authentication (Basic, Plain, Digest, CRAM-MD5, SCRAM-SHA, NTLM, Negotiate and Kerberos), file transfer resume, proxy tunneling and more.

What's curl used for?

curl is used in command lines or scripts to transfer data. curl is also used in cars, television sets, routers, printers, audio equipment, mobile phones, tablets, settop boxes, media players and is the Internet transfer engine for thousands of software applications in over *ten billion installations*.

curl is used daily by virtually every Internet-using human on the globe.

Who makes curl?

Top Sponsors

HAXX
wolfSSL
fastly
TeamViewer

Time to donate to the curl project?

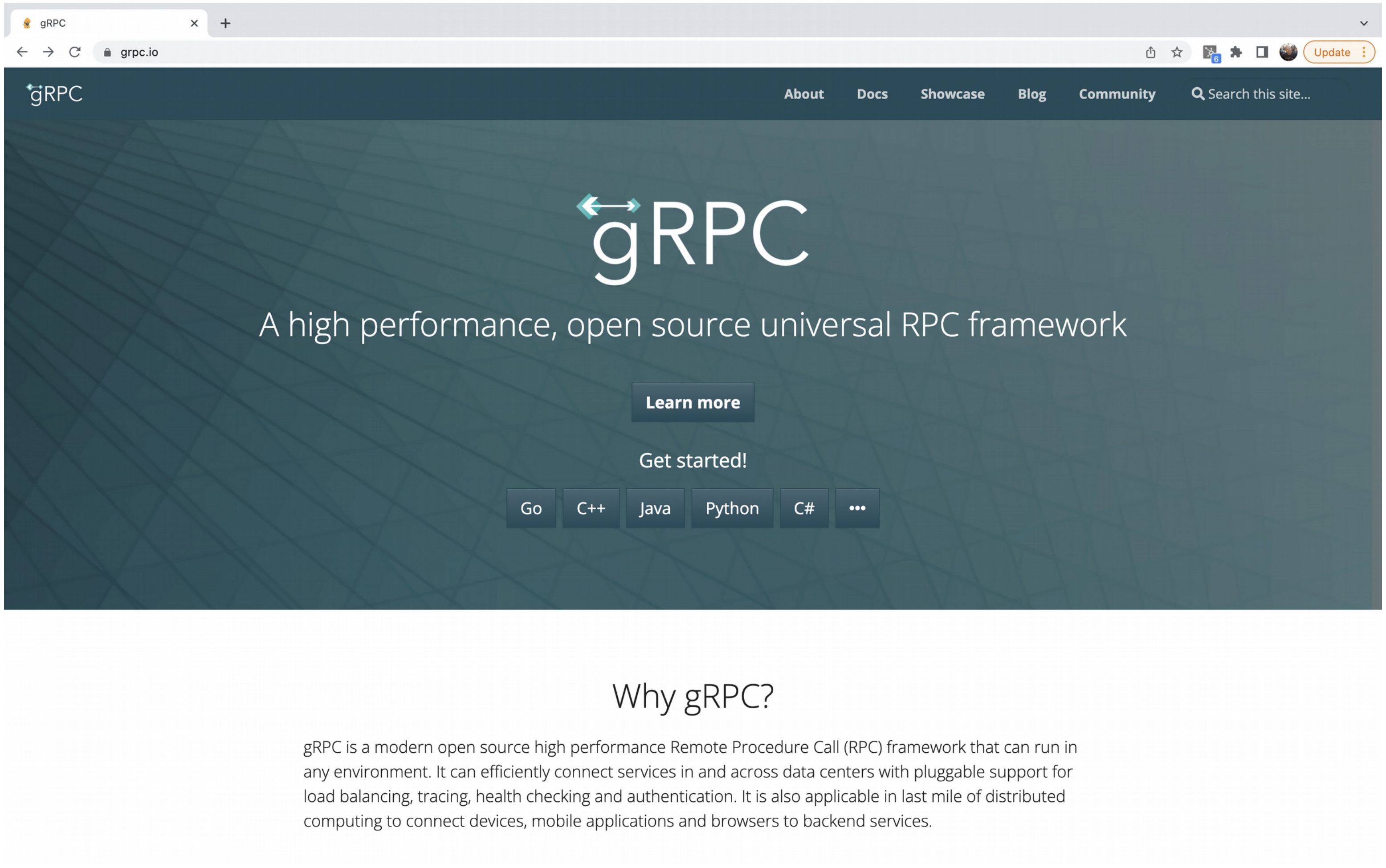
The grpc Framework

- gRPC is a modern open source high performance Remote Procedure Call (RPC) framework.
- This framework allows us to connect services in and across data centers with pluggable support for load balancing, tracing, health checking and authentication.

The grpc Framework

- This framework is also relevant in last mile of distributed computing. We can use gRPC for connecting devices, mobile applications and browsers to backend services.
  https://grpc.io

The grpc Framework



A high performance, open source universal RPC framework

Why gRPC?

gRPC is a modern open source high performance Remote Procedure Call (RPC) framework that can run in any environment. It can efficiently connect services in and across data centers with pluggable support for load balancing, tracing, health checking and authentication. It is also applicable in last mile of distributed computing to connect devices, mobile applications and browsers to backend services.

Data Management

Database Per Service

- Most services need to persist data, and they usually use a database for doing so (e.g. invoices service stores information about invoices we issue, customers service stores information about customers, etc.).
- The Services must be loosely coupled in order to allow us to develop, deploy and scale them independently.

$\downarrow$

The Problem

Database Per Service

- There are many cases in which there is a need to query data managed by multiple micro services.
- Different micro services might have different requirements, which might lead to the use of relational databases in some of the cases, and the use of no-sql databases in others.

$\downarrow$

The Problem

Database Per Service

- The solution for these problems might be having a separated persistent data for each and every micro service, and have a separated API (for each and every micro service), that uses that persist data only.
- We can allocate private tables per service, we can allocate schema per service, and we can allocate a separated database.

$\downarrow$

The Solution

Database Per Service

- Implementing this pattern ensures that the services are loosely coupled. Changes to one service's database won't cause impact any other services.
- Each and every service can use the type of database that is best suited to its needs.

Consequences

Shared Database

- Microservices might need to implement ACID (Atomicity, Consistency, Isolation, and Durability) transactions.

$\downarrow$

The Problem

Shared Database

- We can share a single database between different micro services in order to allow those micro services to implement ACID transactions.

$\downarrow$

The Solution

Shared Database

- The developers can easily implement ACID transactions when working with a single database. The single database is simpler to handle.
- The development of the entire server side might be slower due to the need to coordinate between different developers that work on different micro services every schema change. The coupling between the micro services increases.

V

Consequences

Shared Database

- When the server side is up and running the increased coupling between the micro services might damage the performance of our server side.
- Single database for multiple microservices might be not the optimal one for some of the microservices.

V

Consequences

Communication

Message Queue Pattern

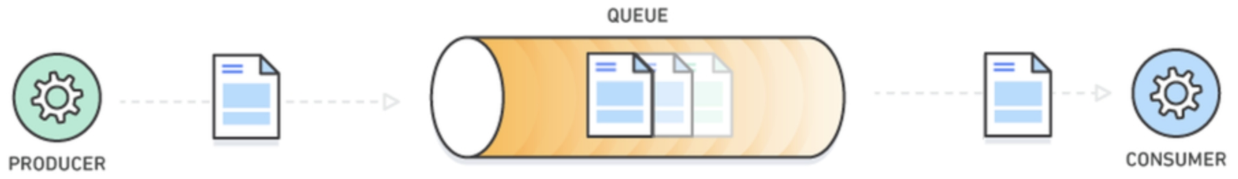
- When implementing the micro services architecture, we might have micro services that need to communicate with each other. This communication needs to be asynchronous in order to keep the micro services loosely coupled.

$\downarrow$

The Problem

Message Queue Pattern

- The solution would be to implement a message queue component that allows communication between processes or between threads in the same process.



The Solution

Image Credit to AWS

Message Queue Pattern

- The message queues provide an asynchronous communication protocol in which the sender and receiver don't need to interact at the same time. The messages are held in a queue until the recipient retrieves them.

↓

The Solution

Message Queue Pattern

- When implementing the micro services architecture, this pattern allows multiple micro services to communicate with each other asynchronously.

↓

The Consequences