Spring Boot 3

Mar - Apr 2023

What is RESTful API?

- RESTful API is an interface that two computer systems use to exchange information securely over the internet.

- RESTful APIs support this information exchange because they follow secure, reliable, and efficient software communication standards.

What is an API?

- An application programming interface (API) defines the rules that you must follow to communicate with other software systems.

- Developers expose or create APIs so that other applications can communicate with their applications programmatically.

Web API

- You can think of a web API as a gateway between clients and resources on the web.

- Clients are users who want to access information from the web. The client can be a person or a software system that uses the API.

- Resources are the information that different applications provide to their clients.

  - Resources can be images, videos, text, numbers, or any type of data. The machine that gives the resource to the client is also called the server.

  - Organizations use APIs to share resources and provide web services while maintaining security, control, and authentication.

  - In addition, APIs help them to determine which clients get access to specific internal resources.

What is REST?

- Representational State Transfer (REST) is a software architecture that imposes conditions on how an API should work.

- REST was initially created as a guideline to manage communication on a complex network like the internet.

- You can use REST-based architecture to support high-performing and reliable communication at scale.

- APIs that follow the REST architectural style are called REST APIs.

- Web services that implement REST architecture are called RESTful web services.

- The term RESTful API generally refers to RESTful web APIs. However, you can use the terms REST API and RESTful API interchangeably.

Principles of the REST

- Uniform interface

- Statelessness

- Layered system

- Cache ability

Uniform interface

- The uniform interface is fundamental to the design of any RESTful webservice. It indicates that the server transfers information in a standard format.

- Uniform interface imposes four architectural constraints:

  1. Requests should identify resources. They do so by using a uniform resource identifier.

  2. Clients have enough information in the resource representation to modify or delete the resource if they want to. The server meets this condition by sending metadata that describes the resource further.

  3. Clients receive information about how to process the representation further. The server achieves this by sending self-descriptive messages that contain metadata about how the client can best use them.

  4. Clients receive information about all other related resources they need to complete a task. The server achieves this by sending hyperlinks in the representation so that clients can dynamically discover more resources.

Statelessness

- In REST architecture, statelessness refers to a communication method in which the server completes every client request independently of all previous requests.

- Clients can request resources in any order, and every request is stateless or isolated from other requests.

- This REST API design constraint implies that the server can completely understand and fulfill the request every time.

Layered system

- In a layered system architecture, the client can connect to other authorized intermediaries between the client and server, and it will still receive responses from the server.

- Servers can also pass on requests to other servers.

- You can design your RESTful web service to run on several servers with multiple layers such as security, application, and business logic, working together to fulfill client requests.

- These layers remain invisible to the client.

Cache ability

- RESTful web services support caching, which is the process of storing some responses on the client or on an intermediary to improve server response time.

- For example, suppose that you visit a website that has common header and footer images on every page. Every time you visit a new website page, the server must resend the same images. To avoid this, the client caches or stores these images after the first response and then uses the images directly from the cache.

- RESTful web services control caching by using API responses that define themselves as cacheable or noncacheable.

What are the benefits of RESTful APIs?

- Scalability

  - optimizes client-server interactions

  - Statelessness

- Flexibility

  - client-server separation

  - They simplify and decouple various server components so that each part can evolve independently.

  - Platform or technology changes at the server application do not affect the client application.

- Independence

  - Independent of the technology used.

  - Various programming languages without affecting the API design

  - Change the underlying technology, Without affecting the communication

How do RESTful APIs work?

C → Create → POST

R → Read → GET

U → Update → PUT

D → Delete → DELETE

HTTP Methods

edureka!
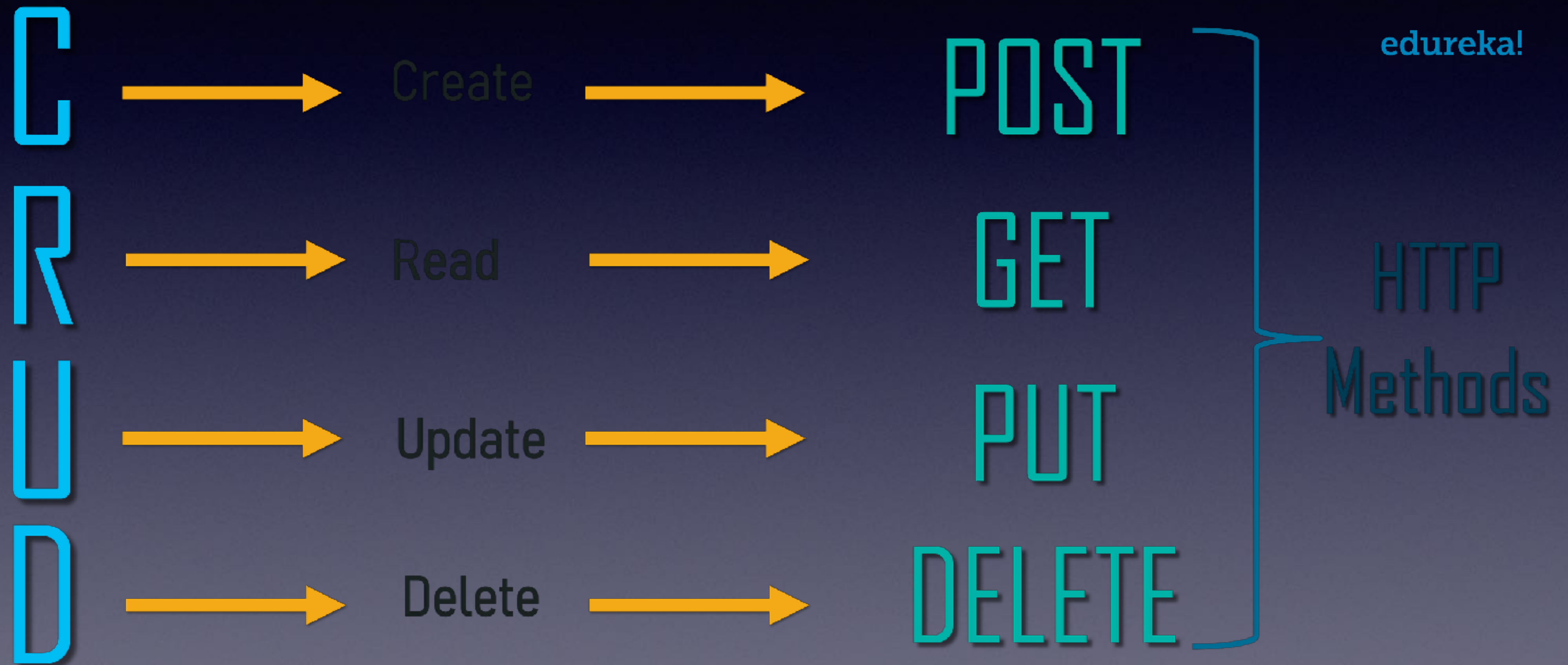
What does the RESTful API client request contain?

- Unique resource identifier

- Method

  - GET / POST / PUT / DELETE

- HTTP headers

- Data

  - Request body

- Parameters

  - Path parameters

  - Query parameters

  - Cookie parameters

What are RESTful API authentication methods?

- Basic authentication

- Bearer authentication

- API keys

- OAuth

What does the RESTful API server response contain?

- Status line

  - 2XX codes indicate success

  - 4XX and 5XX codes indicate errors.

  - 3XX codes indicate URL redirection

- Message body

- Headers

Interceptor

- authentication

- authorization

- logging

- customizing
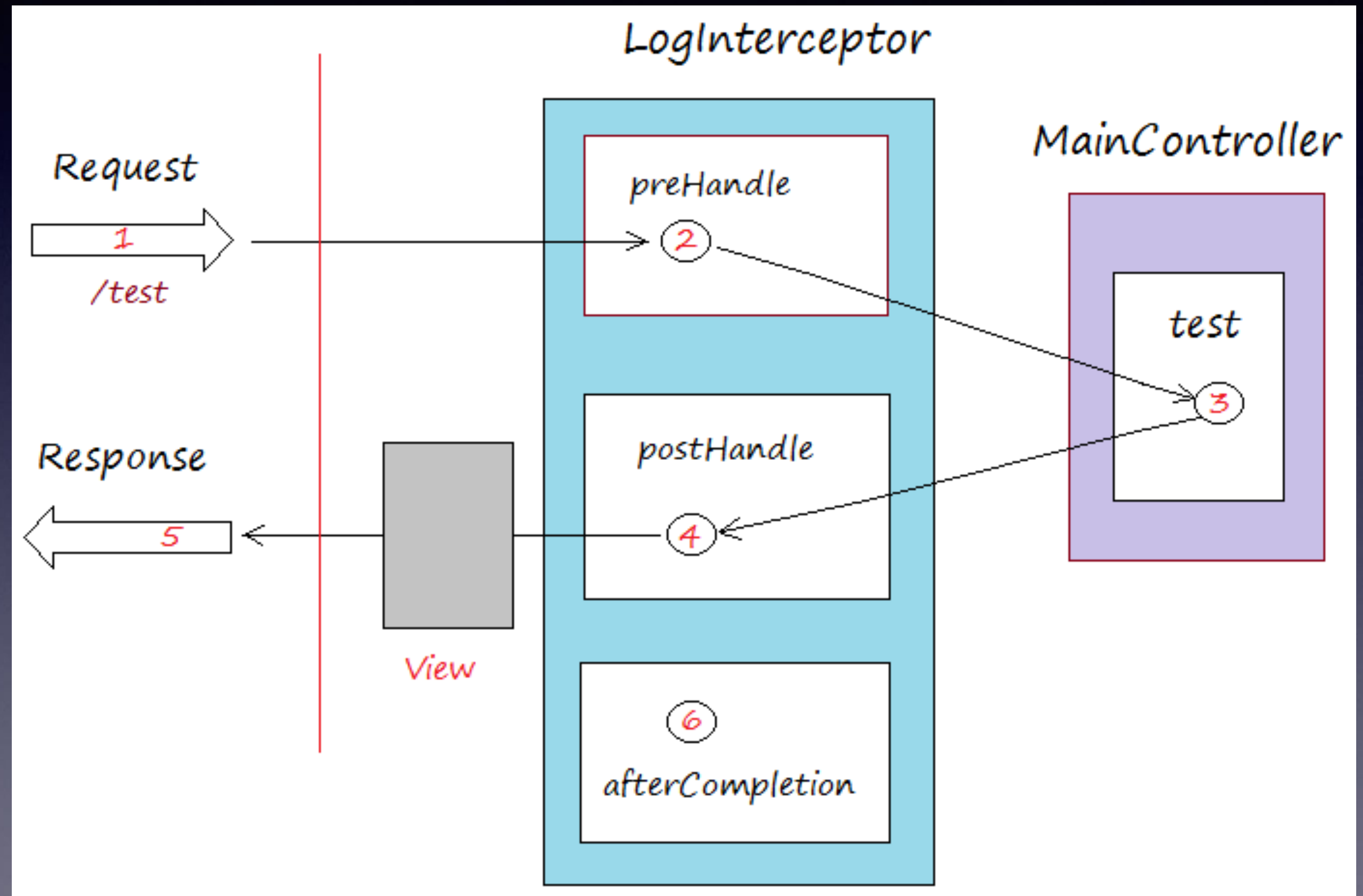
Web Filter
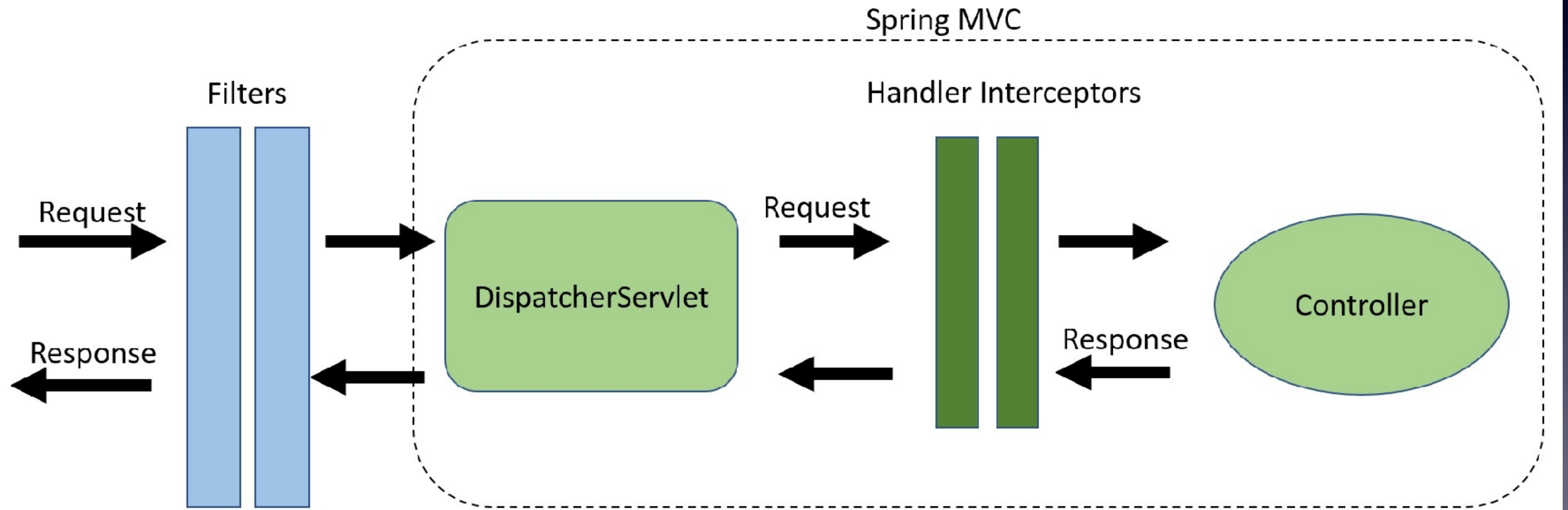
SpringDoc-OpenApi V2.0.2
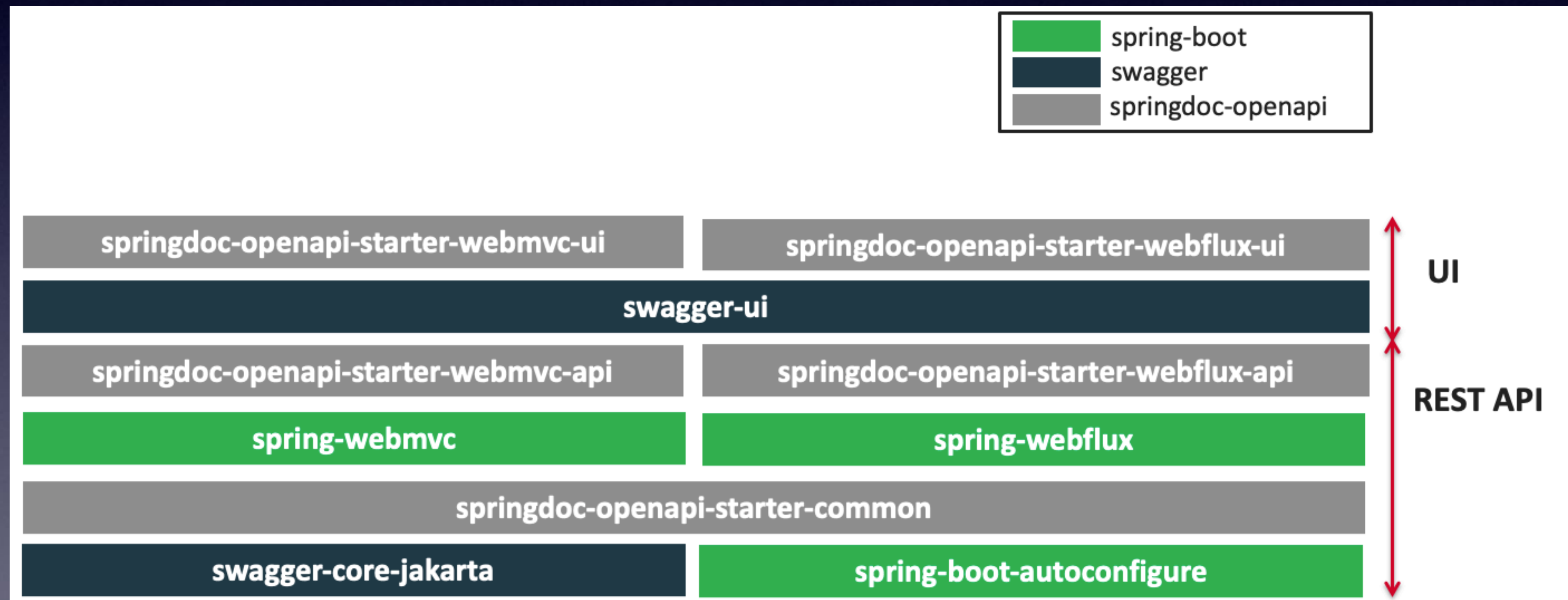
Springdoc-openapi Demos

GraphQL
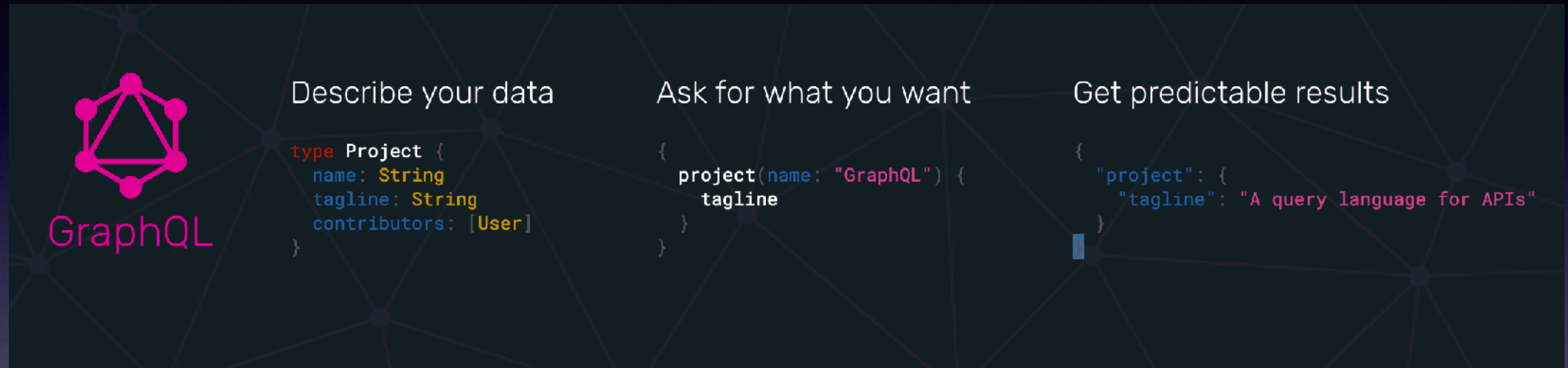
- open-source

- data query language

- data manipulation language for APIs

- and a query runtime engine.

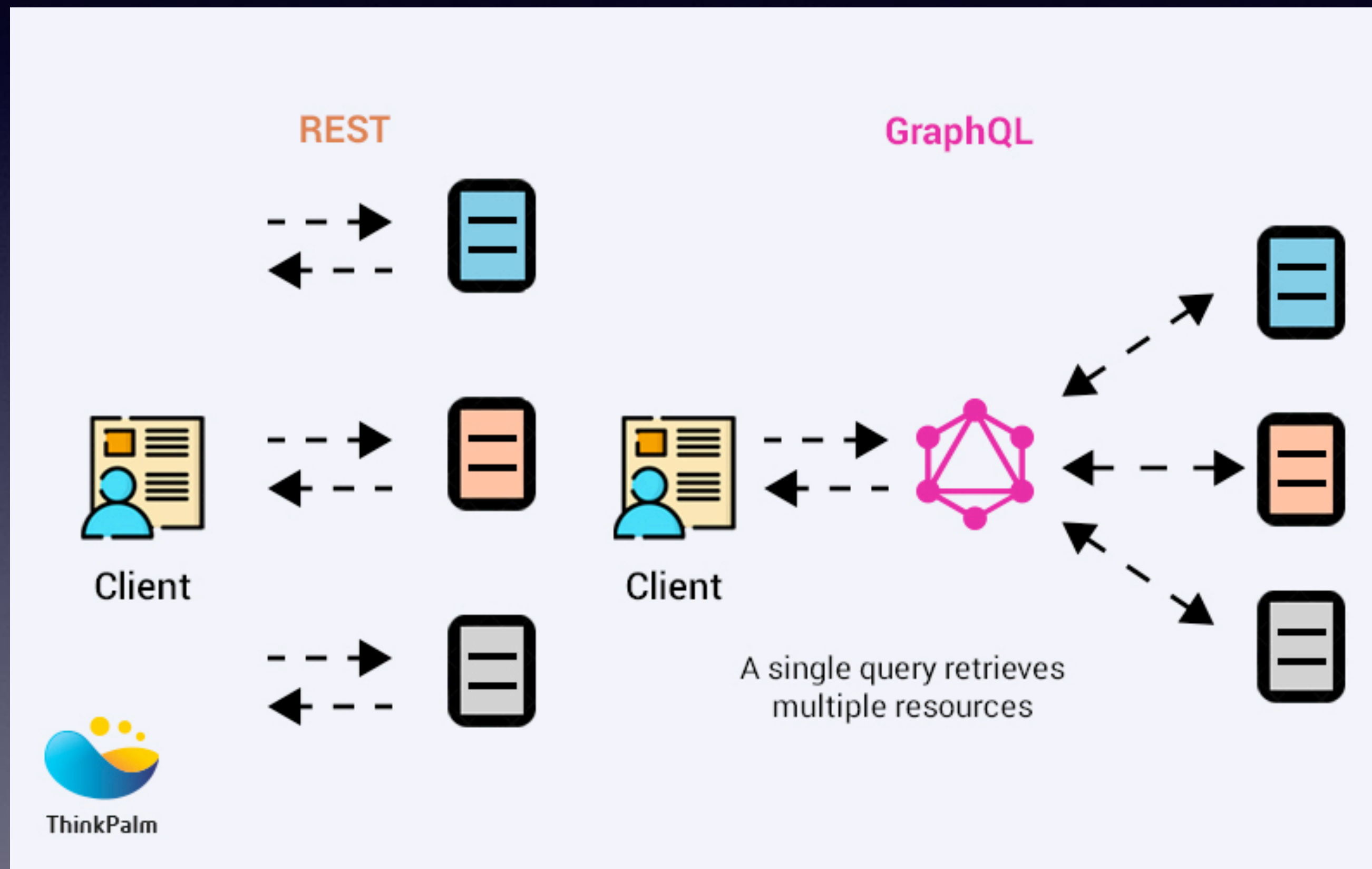# Ask for what you need, get exactly that

```
1 ▾ {
2 ▾     customer {
3           firstname
4           lastname
5           sample_attribute
6       }
7   }
8
```

```
{
    "data": {
        "customer": {
            "firstname": "test1",
            "lastname": "test1",
            "sample_attribute": "My sample attr value"
        }
    }
}
```

Get many resources
in a single request

Describe what's possible
with a type system

- GraphQL APIs are organized in terms of types and fields, not endpoints.

- Access the full capabilities of your data from a single endpoint.

- GraphQL uses types to ensure Apps only ask for what's possible and provide clear and helpful errors.

- Apps can use types to avoid writing manual parsing code.

Move faster with powerful developer tools

- Know exactly what data you can request from your API without leaving your editor, highlight potential issues before sending a query, and take advantage of improved code intelligence.

- GraphQL makes it easy to build powerful tools like **GraphiQL** by leveraging your API's type system.

Evolve your API
without versions

- Add new fields and types to your GraphQL API without impacting existing queries.

- Aging fields can be deprecated and hidden from tools.

- By using a single evolving version, GraphQL APIs give apps continuous access to new features and encourage cleaner, more maintainable server code.

Bring your own
data and code

- GraphQL creates a uniform API across your entire application without being limited by a specific storage engine.

- Write GraphQL APIs that leverage your existing data and code with GraphQL engines available in many languages.

- You provide functions for each field in the type system, and GraphQL calls them with optimal concurrency.

Interception

- Server transports allow intercepting requests before and after the GraphQL Java engine is called to process a request.

    - check HTTP request details

    - Customize graphql.ExecutionInput

    - Add HTTP response headers

    - Customize graphql.ExecutionResult

Error Handling

- GraphQL Response With Handled Exception

  - Extend a class from DataFetcherExceptionResolverAdapter

- @GraphQlExceptionHandler

- @ControllerAdvice

CAP theorem
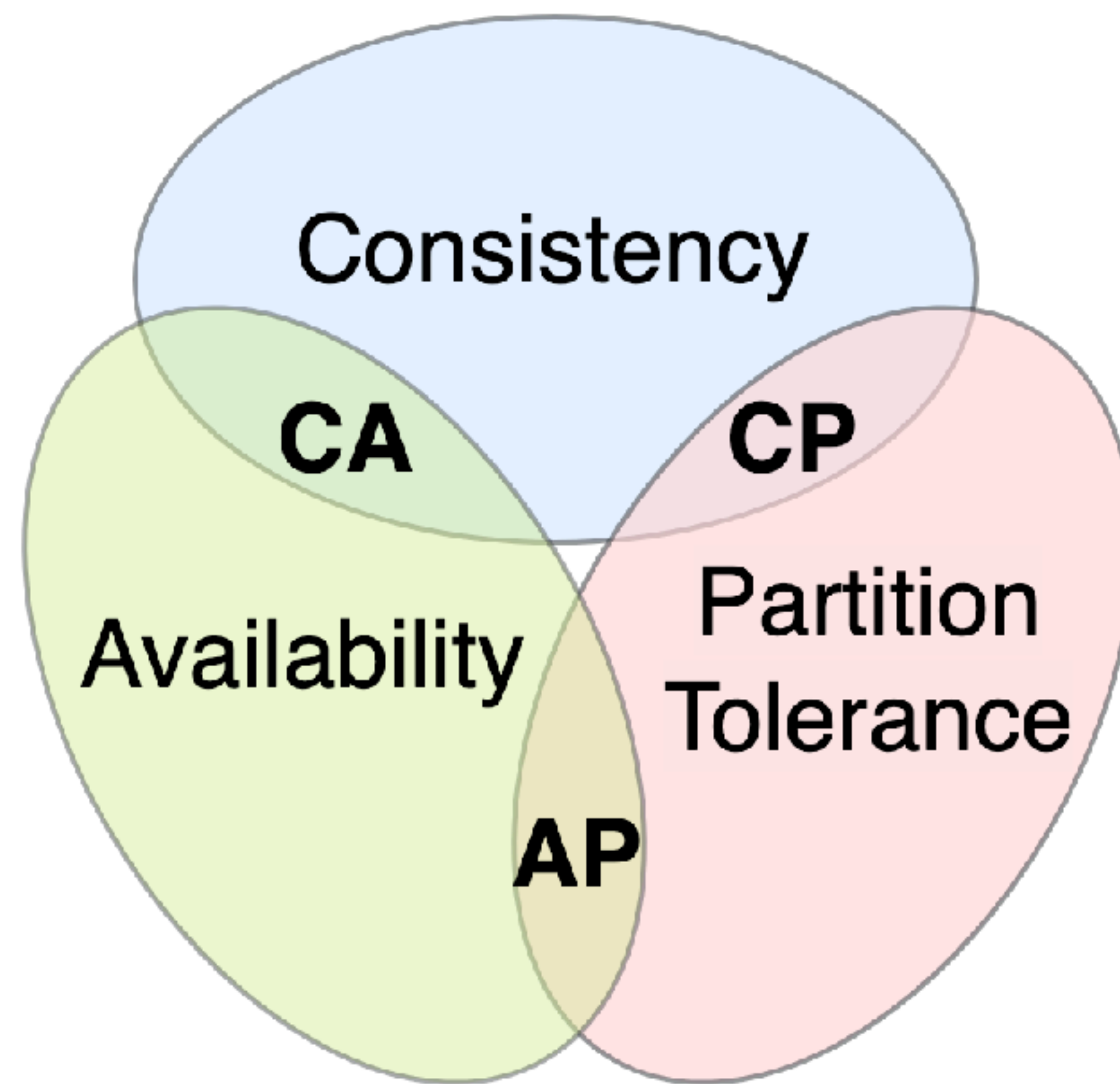
Service Registry & Discovery

- The focus on scriptable configuration allows for better container management.
  Eureka requires either external Configuration Server or multiple configuration files.

- The options for securing communications is more advanced.
  Eureka requires creating application with security settings desired. Default will allow HTTP only. Registration of end points assumes http but can be forced to https with code.

- Support for non-REST endpoints via DNS. This would allow database and other resource connections.
  Eureka presumably would do this through ZUUL and/or Sidecar.
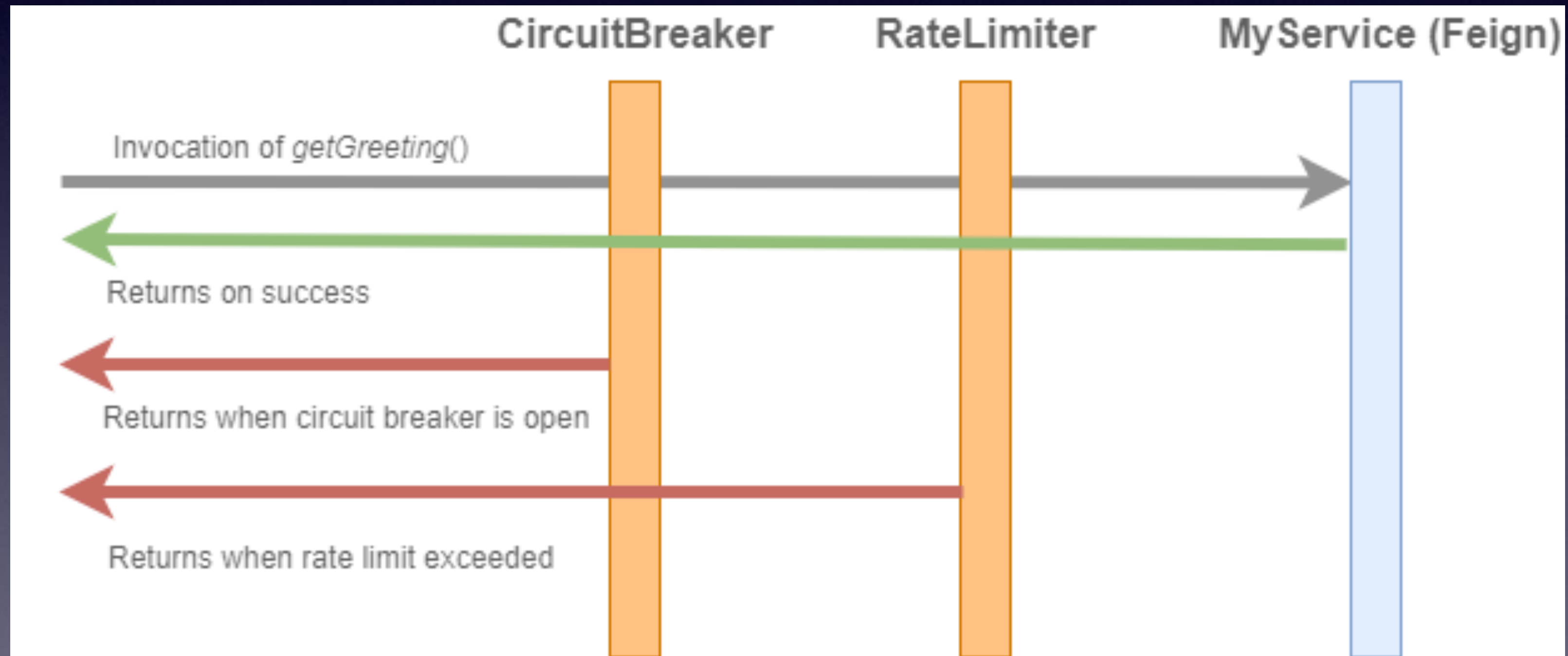
Routing Factories

- Spring Cloud Gateway **matches routes using the Spring WebFlux HandlerMapping infrastructure**. It also includes many built-in Route Predicate Factories. All these predicates match different attributes of the HTTP request. Multiple Route Predicate Factories can be combined via the logical "and"
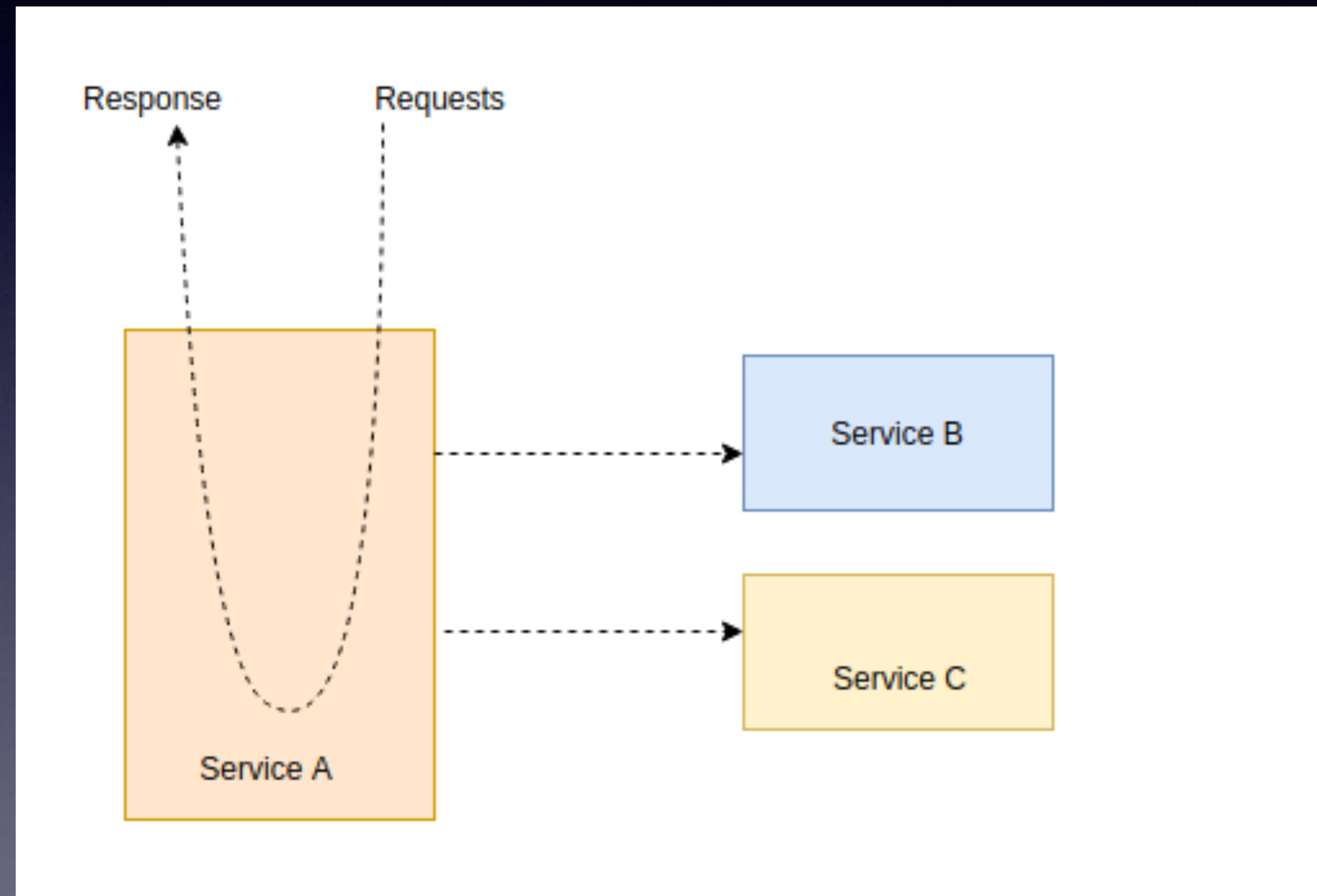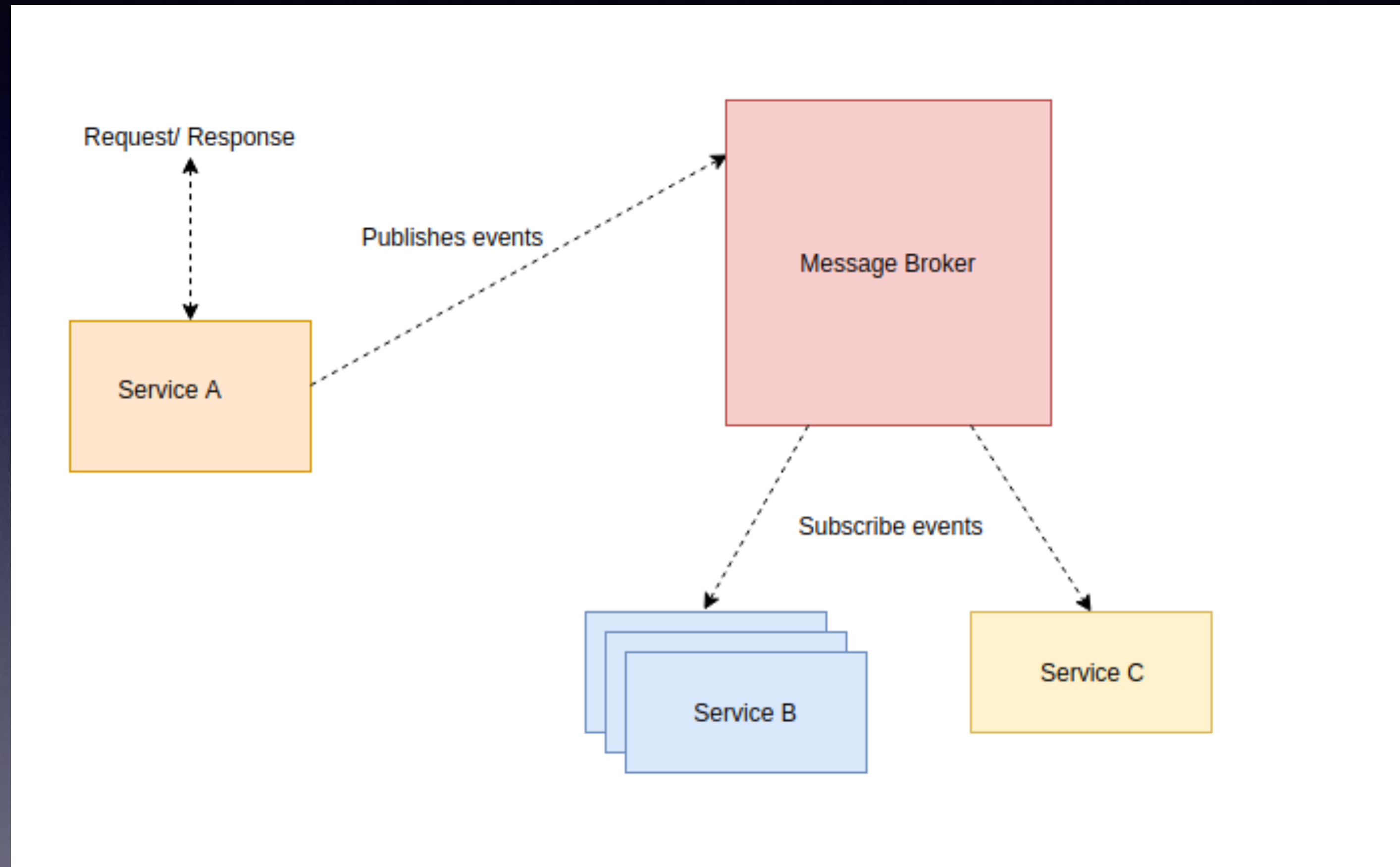
Feign Client
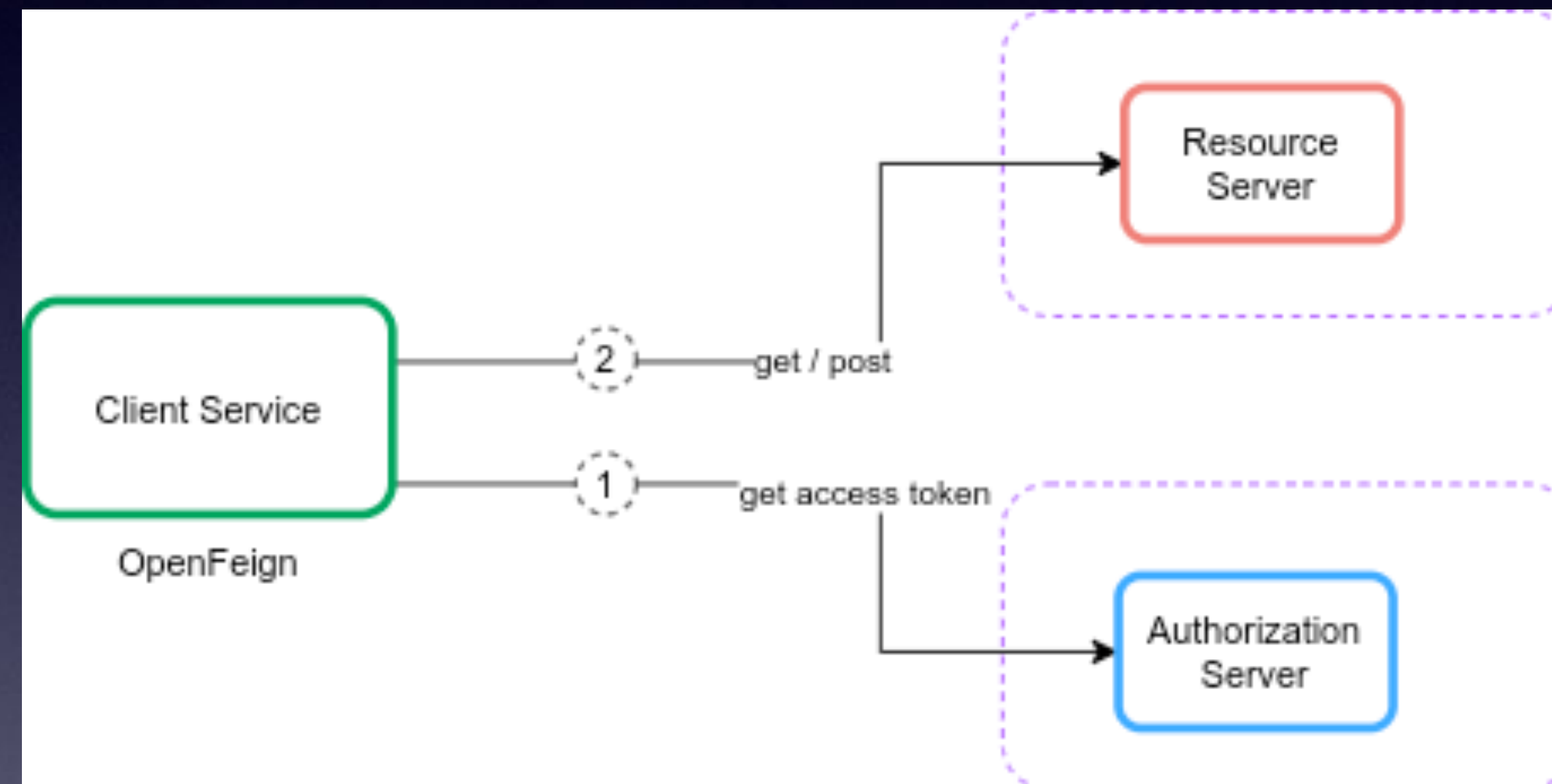
Circuit Breaker
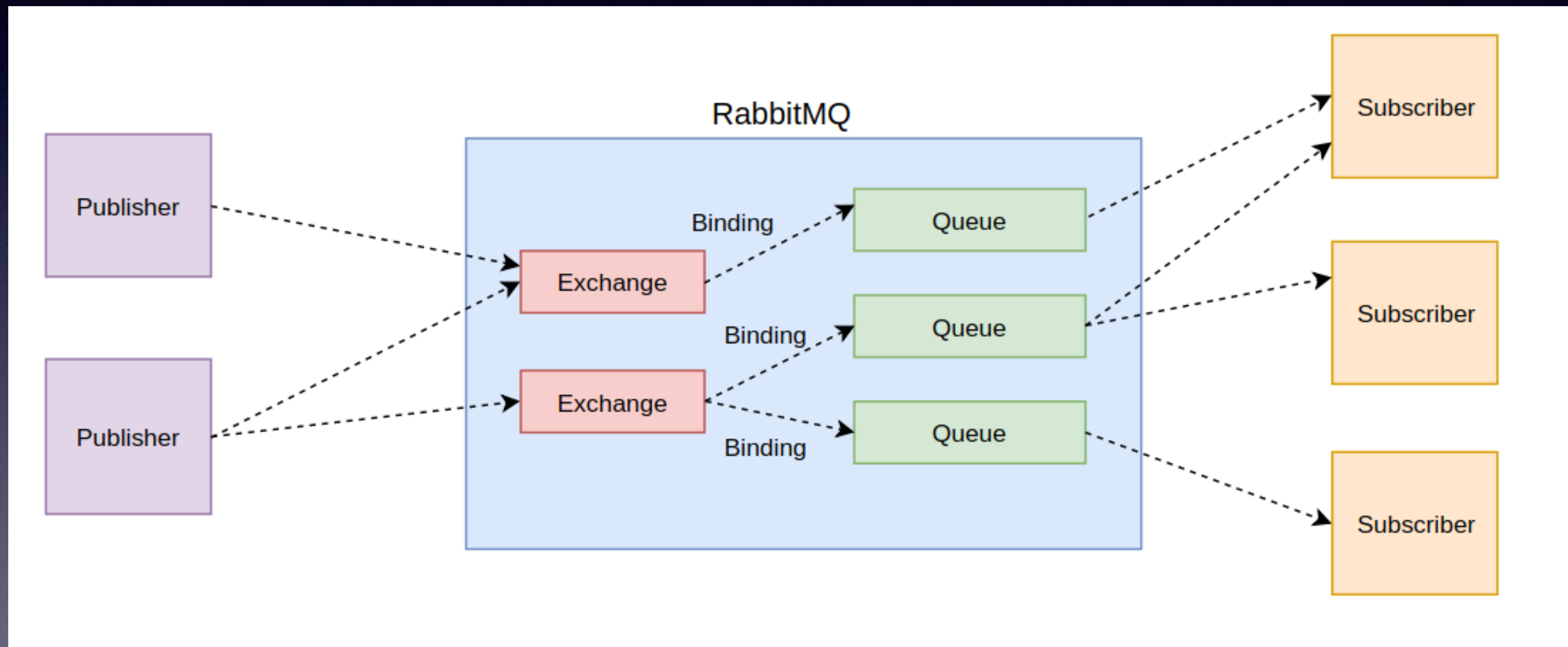
Microservices - Service Call

Microservices - Messaging

RabbitMQ Architectural Design

RabbitMQ Bindings