



Evolving your APIs

A step-by-step approach

 [@nicolas_frankel](https://twitter.com/nicolas_frankel)

Me, myself and I



- Developer
- Developer advocate

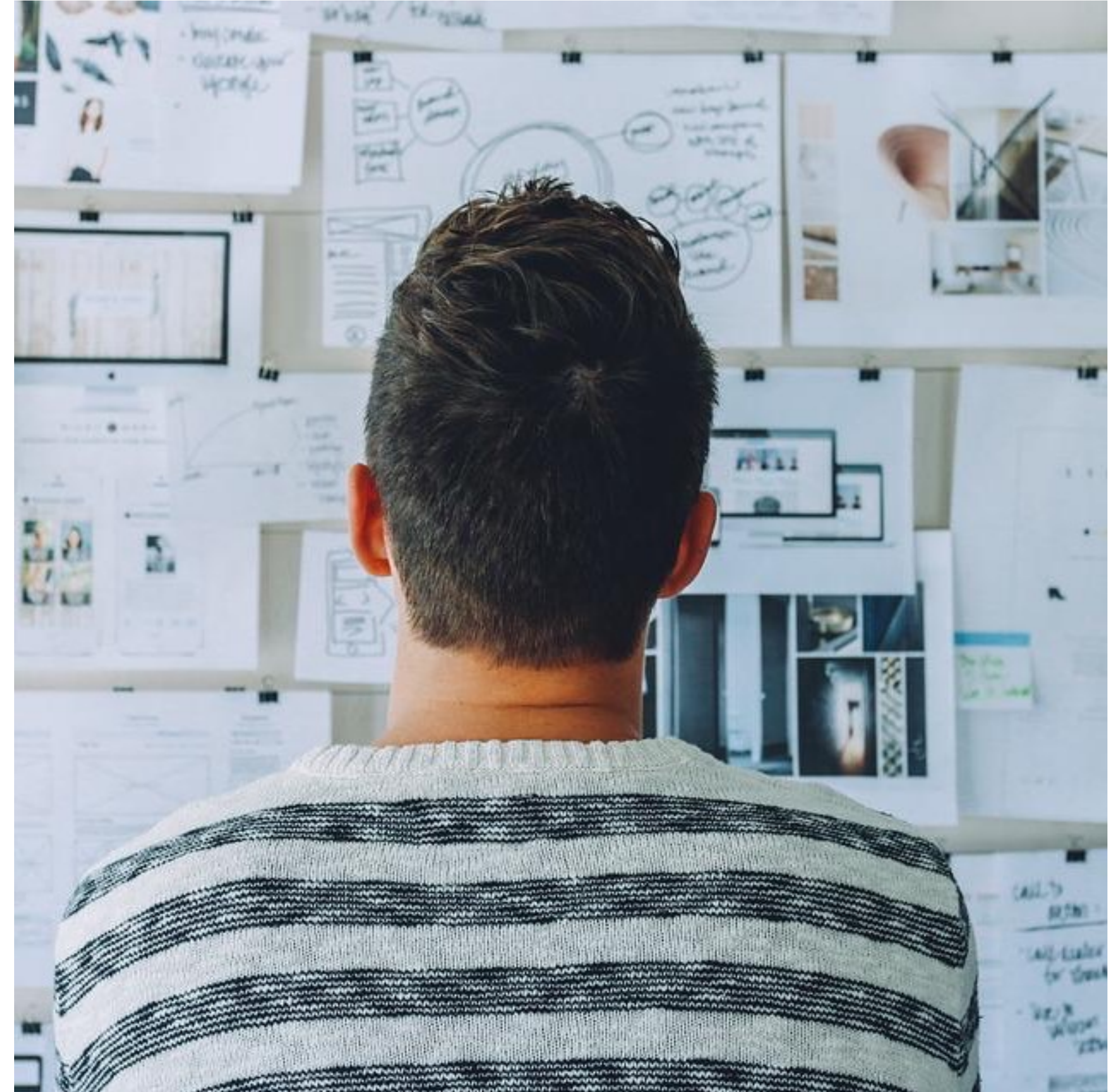


 @nicolas_frankel

Your first API



- Probably focused on REST(ful) semantics



We need to deploy v2!

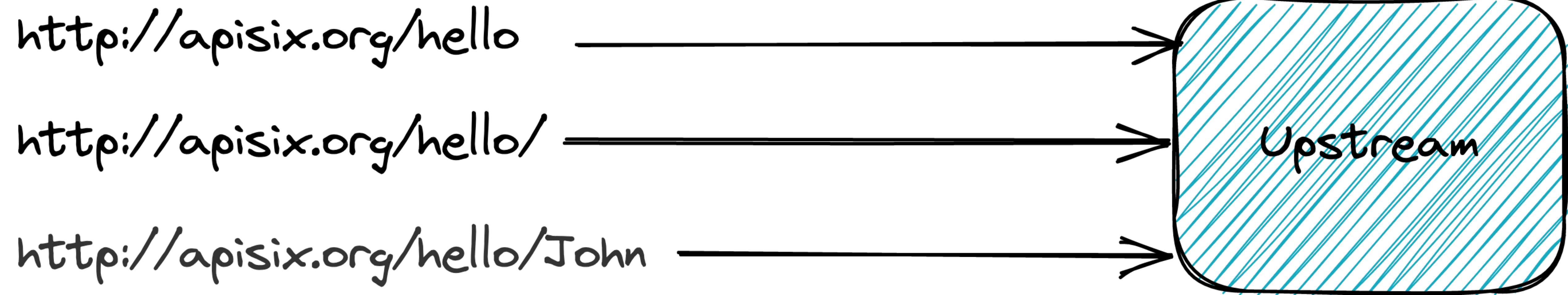


 @nicolas_frankel

Step 1 – The initial situation



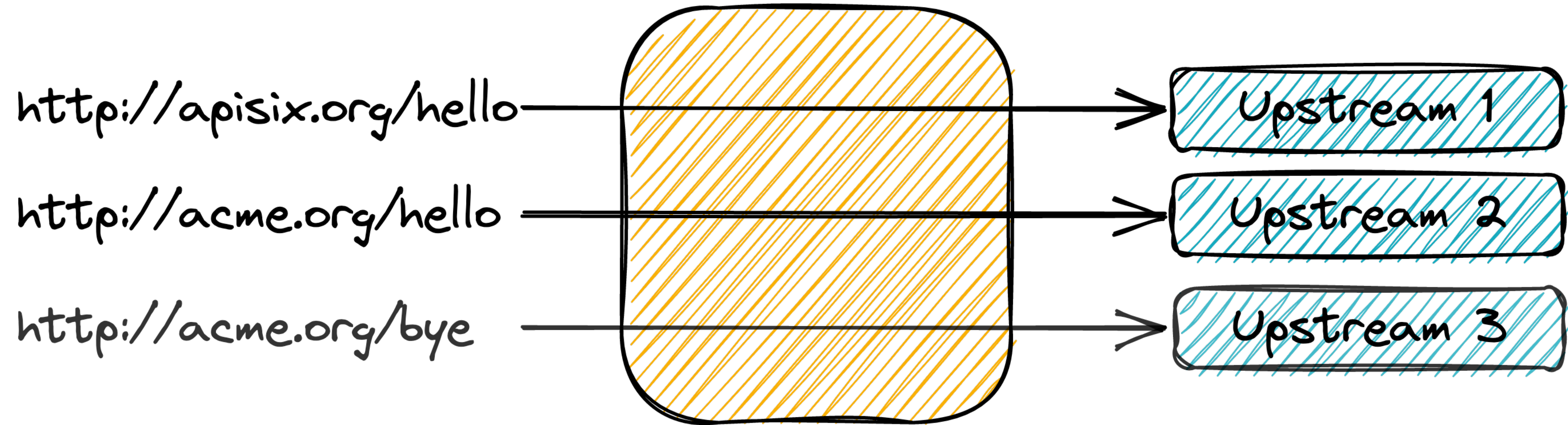
No Gateway



An API Gateway can help!



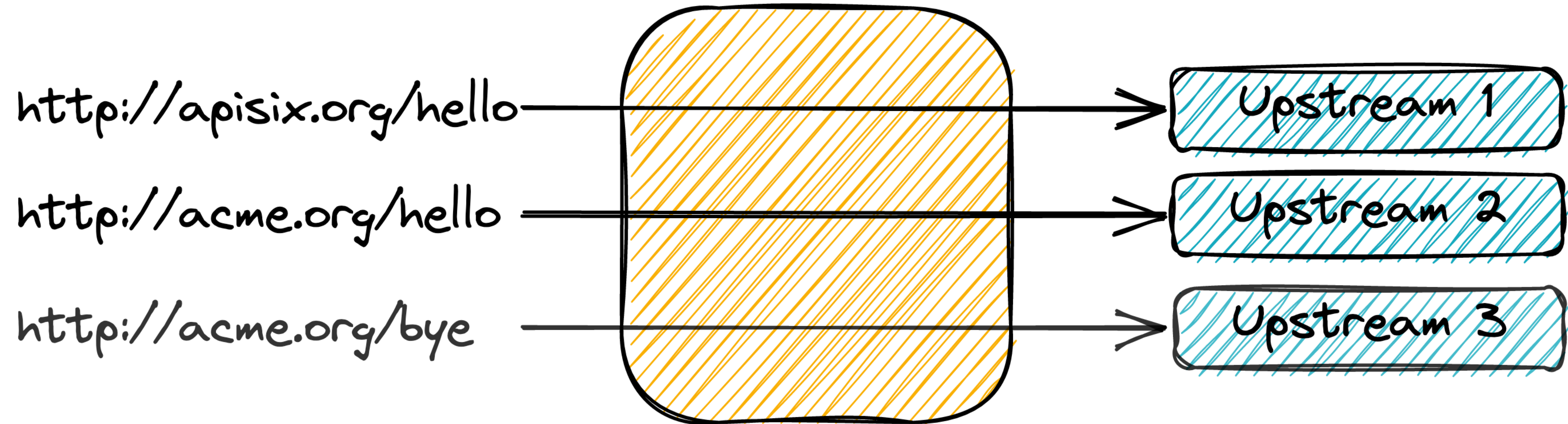
API Gateway



But we have a reverse proxy



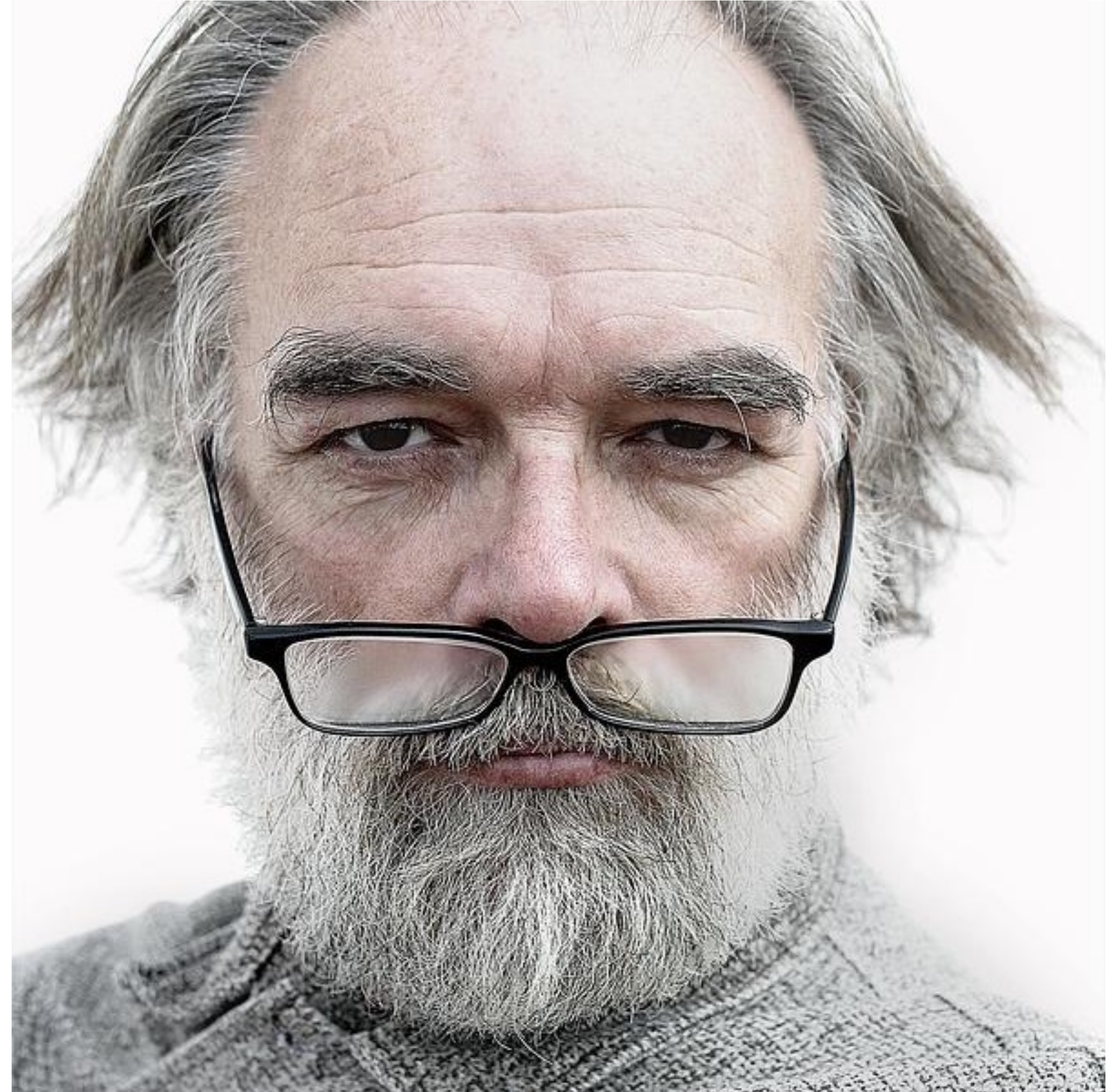
Reverse Proxy



Let's go back a bit



- I started to use the Internet a bit before 2000



Internet > World Wide Web

■ WWW

- Hypertext document management system accessed over the Internet

■ Internet

- Global system of interconnected computer networks that uses TCP/IP to communicate



My first website



■ HTML:

- With images
- With MIDI audio
- No CSS
- No JavaScript

```
HotDog Professional 7.03 - 30 Days left in trial version
File Edit Insert Format View Table Frame Form Tools External tools Window Buy HotDog
C:\Archivos de programa\Sausage\HotDog7\HTMLFiles\trial2.shtml
1 <html>
2   <head>
3     <meta name="Generator" content="Sausage Software
Professional 7">
4     <title>
5       Title goes here
6     </title>
7   <link rel="stylesheet" href="C:\ARCHIV~1\Sausage\HotDog
.css">
8
9 </head>
10  <body background="file:///c:/documents and settings/
documentos/mis imágenes/imágenes de muestra/colinas azu
bgcolor="#C0C0C0" text="#000000" alink="#FF0000" vlink=
link="#0000FF">
11
```


Web server



- Serve static content



Personal Home Page



- Apache Web Server
- People wanted more dynamic content
 1. CGI scripts
 2. PHP



The rise of WWW



- Companies started to use it as an official communication channel
 - Redundancy
 - Load balancing



WWW becomes ubiquitous



- More unrelated nodes
 - Routing



The evolution of the Web Server

1. Serve static content
2. Serve dynamic content
3. Load balancer
4. Routing



Reverse Proxy: Single Point of Entry

- Any cross-concern feature
 - Authentication
 - Authorization
 - Caching
 - IP Blocking



Interconnecting heterogeneous Information Systems



- File sharing with FTP
- Synchronous communication inside a tech stack
- Synchronous communication via HTTP



API Gateway



- Specialized Reverse Proxy
- Features dedicated to APIs
 - Billing
 - Complex rate-limiting
 - Etc.



API Gateways



- Apache APISIX
- Kong Gateway
- Tyk
- Gloo
- Ambassador
- Gravitee

 @nicolas_frankel



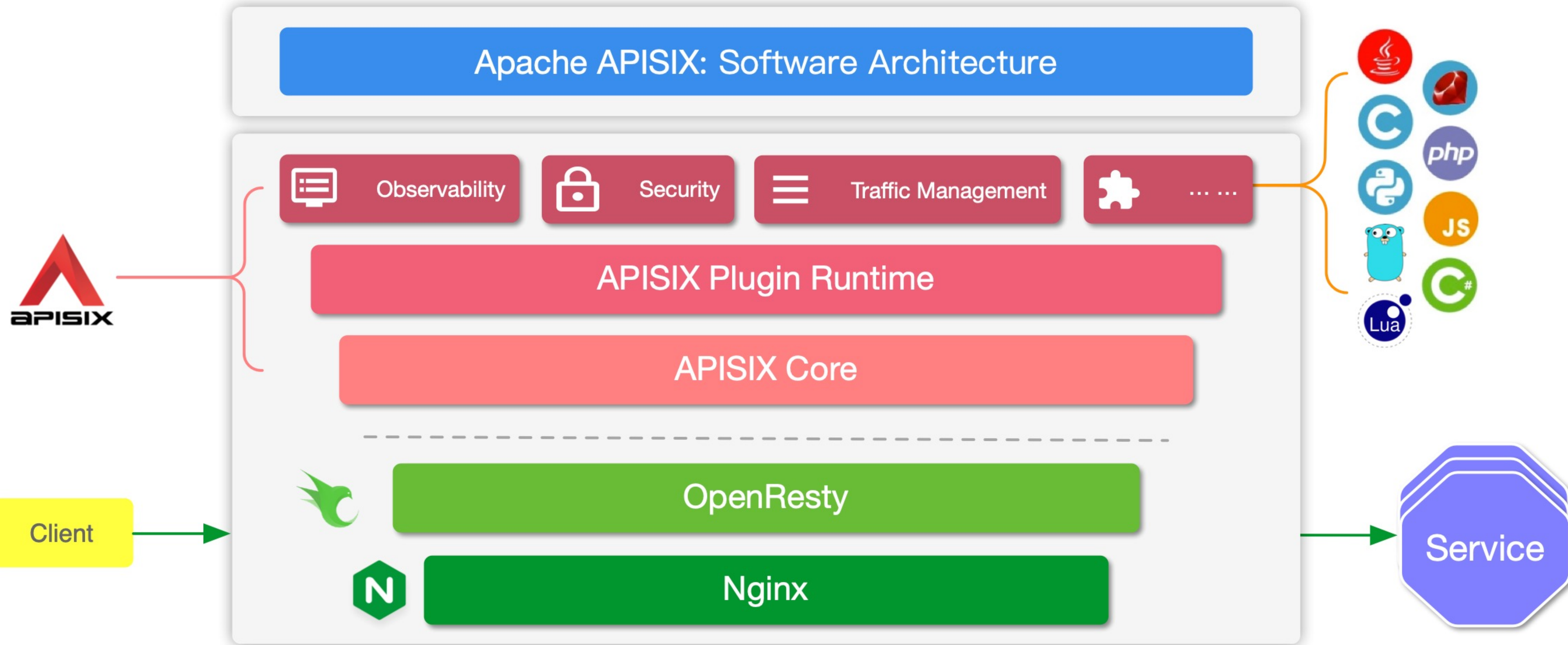
Apache APISIX



- Donated to the Apache Foundation in 2019
- Became a Top-Level Project in 2020



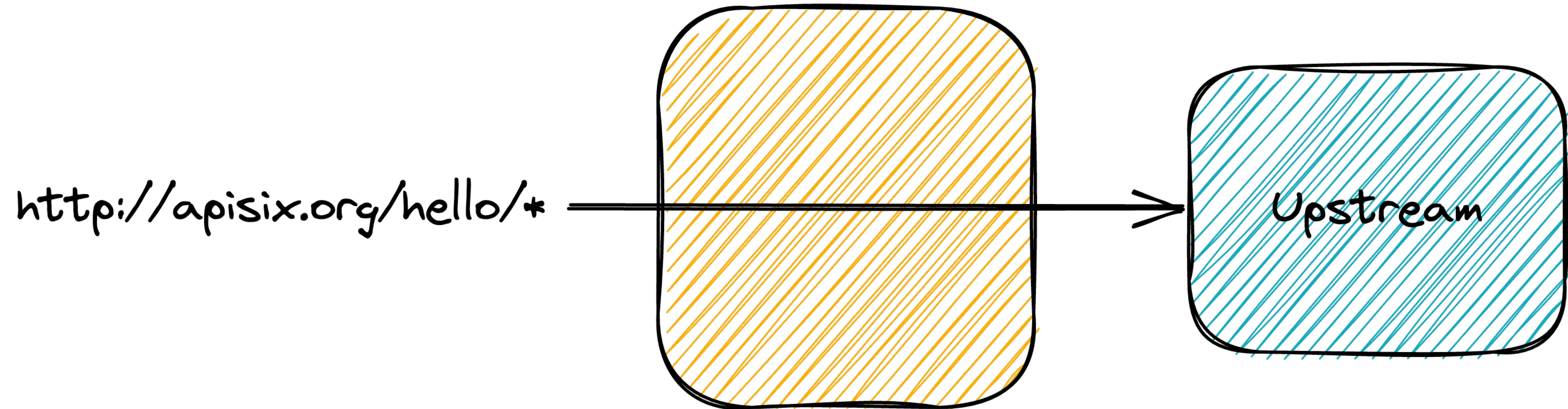
Apache APISIX, an API Gateway the Apache way



Step 2 – Introduce an API Gateway



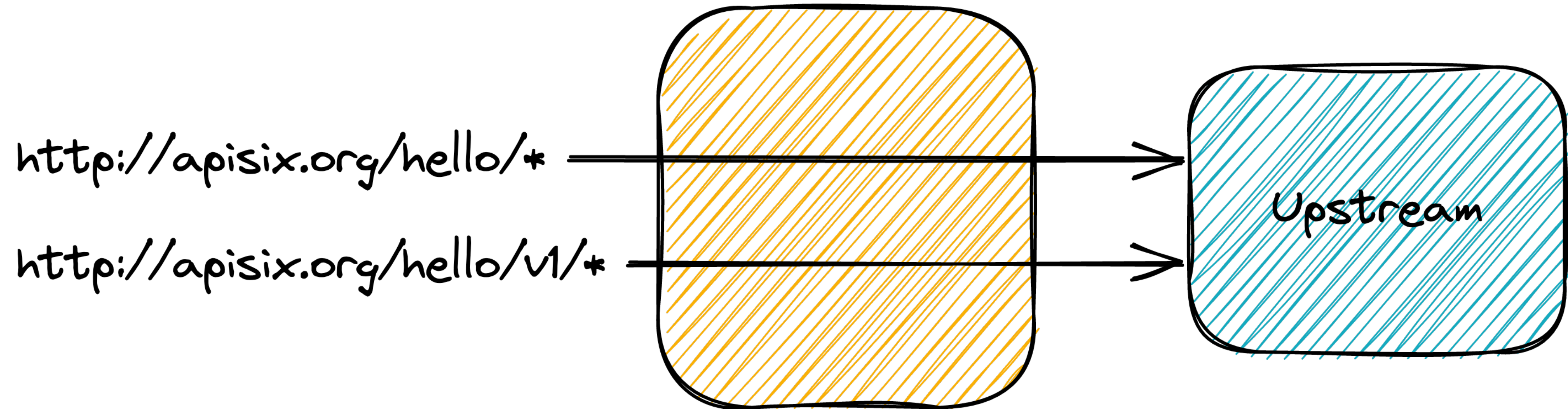
API Gateway



Step 3 – Introduce a versioned resource

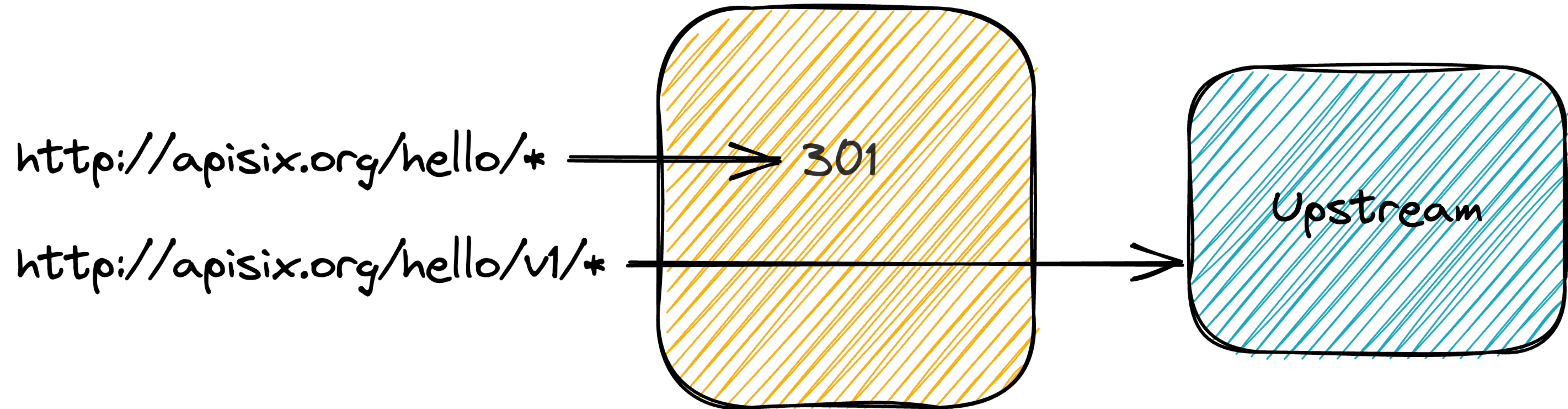


API Gateway



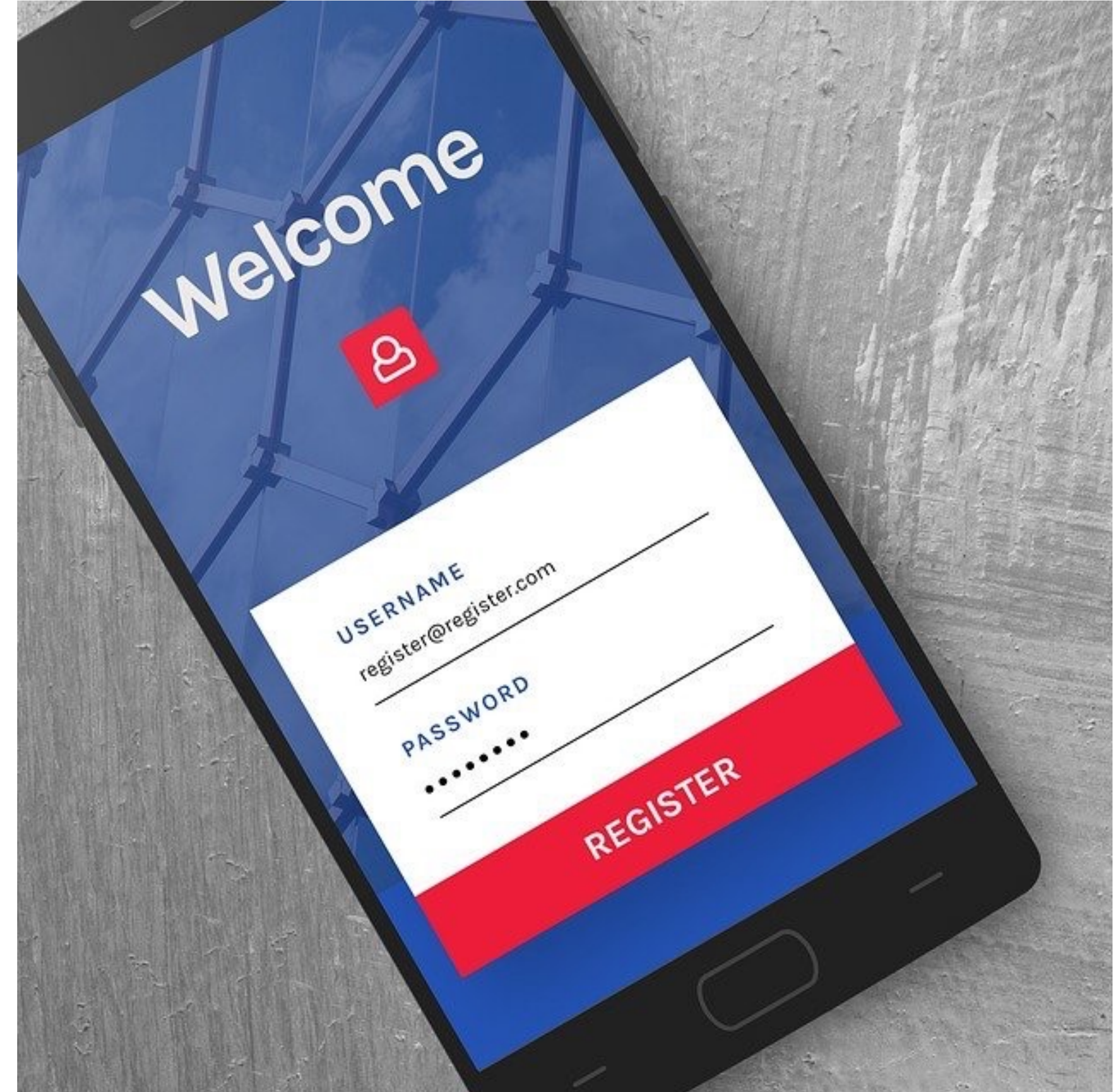
Step 4 – Move the unversioned resource permanently

API Gateway



Step 5 – Make users register

- Hard to change because we don't know our users
- But developers don't like to register
- Provide them an incentive
 - 429

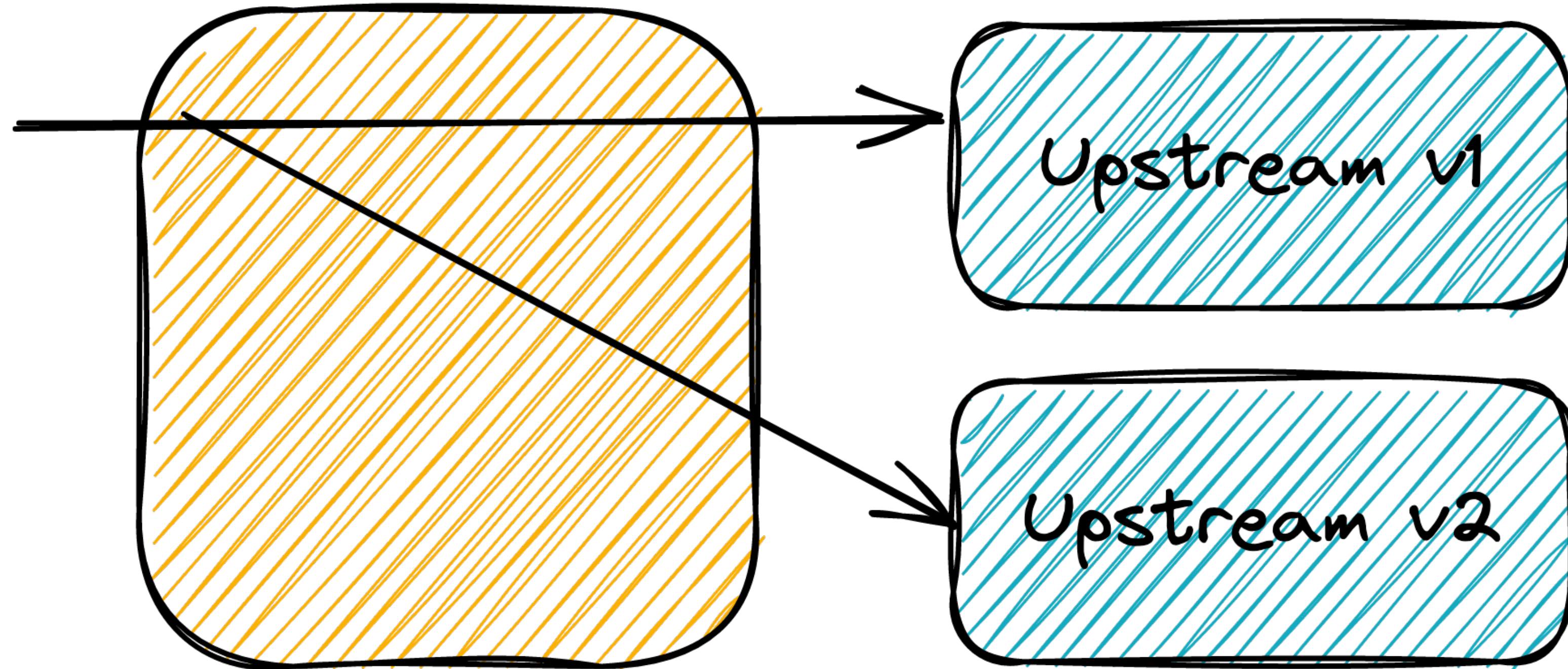


Step 6 – Test in production but be smart about it



API Gateway

`http://apisix.org/hello/v1/*`



Step 7 – Use proven deployment methods



- Canary release for the win



Step 8 – Deprecate your endpoints

- IETF Draft
- Deprecation header
 - Date or Boolean
- Link header to point to the new resource
- Sunset header



Step 9 and afterwards – Time for v3!



Thanks for your attention!



- <https://blog.frankel.ch/>
- @nicolas_frankel
- <https://bit.ly/evolve-apis>
- <https://apisix.apache.org/>

