Data Residency

@nicolas_frankel

Me, myself and I

- Developer
- Developer advocate

@nicolas_frankel

Data Residency

"Data localization or data residency law requires data about a nation's citizens or residents to be collected, processed, and/or stored inside the country, often before being transferred internationally. Such data is usually transferred only after meeting local privacy or data protection laws, such as giving the user notice of how the information will be used and obtaining their consent."

-- https://en.wikipedia.org/wiki/Data_localization

Data Residency Flavors

- Legal Requirements
- Jurisdictional challenges

Legal Requirements in 2023

Country	Scope
Australia	Health records
Canada	Public service providers: all personal data
China	Personal, business, and financial data
Germany	Telecommunications metadata
India	Payment System Data
Indonesia	Public services companies must maintain data centers in country
Kazakhstan	Servers running on the country domain (.kz)
Nigeria	All government data
Russia	All personal data

Jurisdictional challenges: Patriot Act & USA Cloud Act

"[…] primarily amends the Stored Communications Act (SCA) of 1986 to allow federal law enforcement to compel **U.S.-based technology companies** via warrant or subpoena to provide requested data stored on servers regardless of whether the data are stored in the U.S. **or on foreign soil**"

-- https://en.wikipedia.org/wiki/CLOUD_Act

Cloud Providers and Data Sovereignty

- Google:
  - No data center in Malaysia
- AWS:
  - No data center in Malaysia
- Azure:
  - No data center in Malaysia
  - “Coming soon” (June 2023)

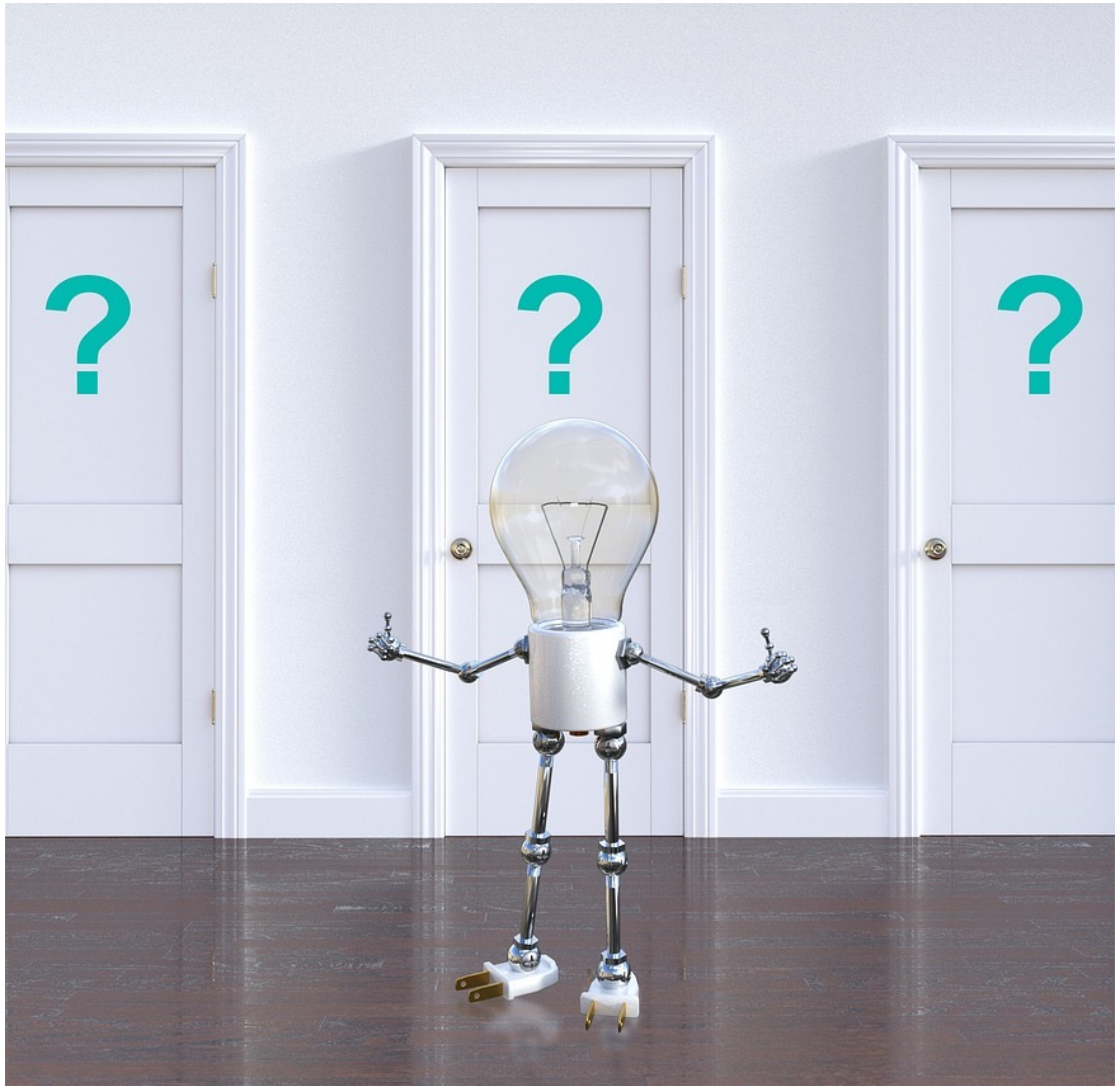
Data Partitioning

"A partition is a division of a logical database or its constituent elements into distinct independent parts. Database partitioning is normally done for **manageability**, **performance** or **availability** reasons, or for load balancing. It is popular in distributed database management systems, where **each partition may be spread over multiple nodes**"

-- https://en.wikipedia.org/wiki/Partition_(database)

Partitioning criteria

- Round-robin partitioning
- (Consistent) Hash partitioning
- Range partitioning
- List partitioning
- Composite partitioning



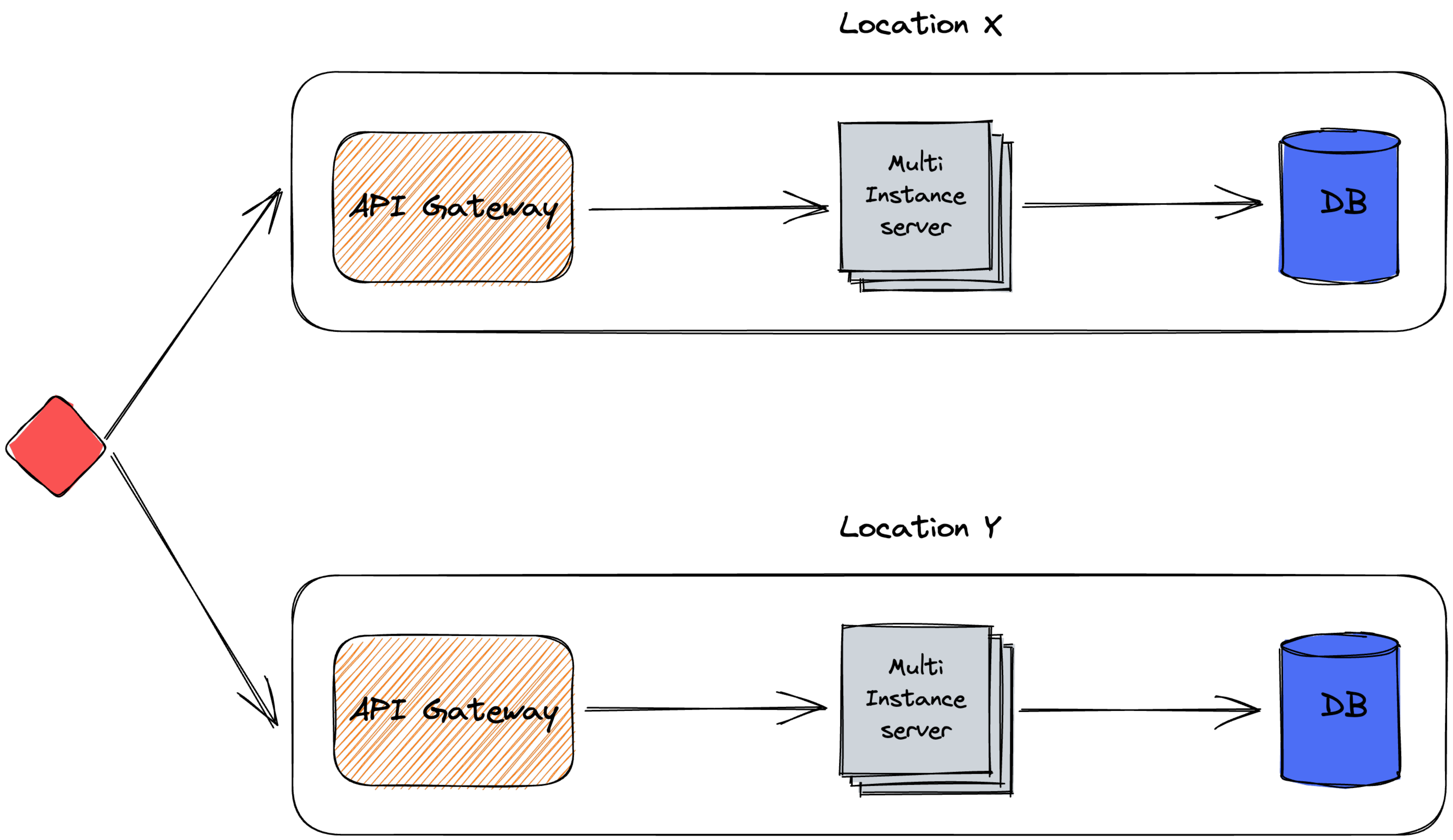
Data Partitioning in the wild

- Most of data partinioning is for horizontal scaling
  - Consistent hashing

Partitioning criteria

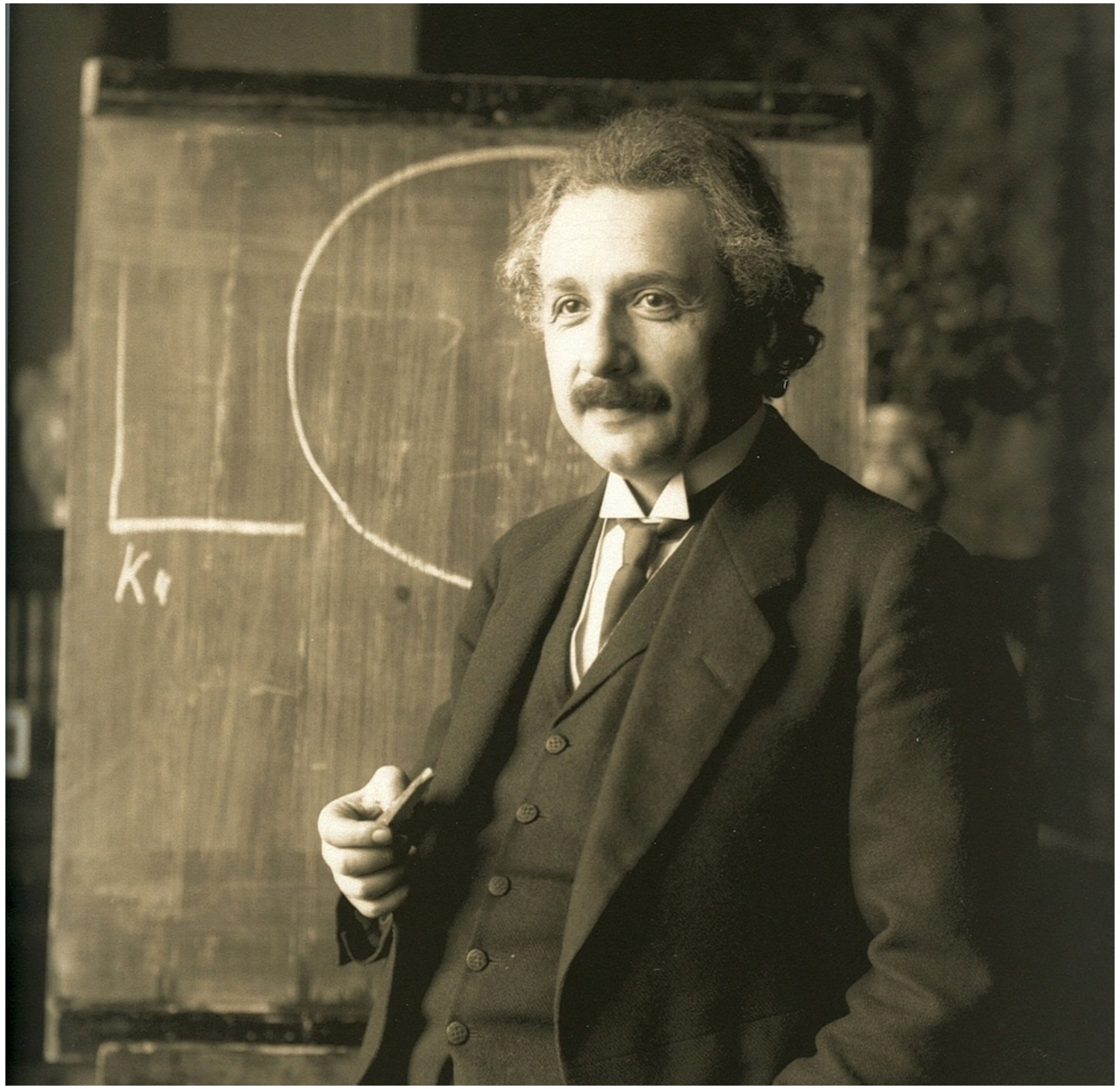
- ~~Round-robin partitioning~~
- ~~(Consistent) Hash partitioning~~
- ~~Range partitioning~~
- List partitioning
- Composite partitioning?

A simple two-locations architecture



Our very complex algorithm

```
IF country = 'us' THEN
  location = 'X'
ELSE IF country = 'fr' THEN
  location = 'Y'
```



Where to put the algorithm

- In the client?
- In the app?
  - In a lib/framework?
- In a proxy?
- In the API Gateway?

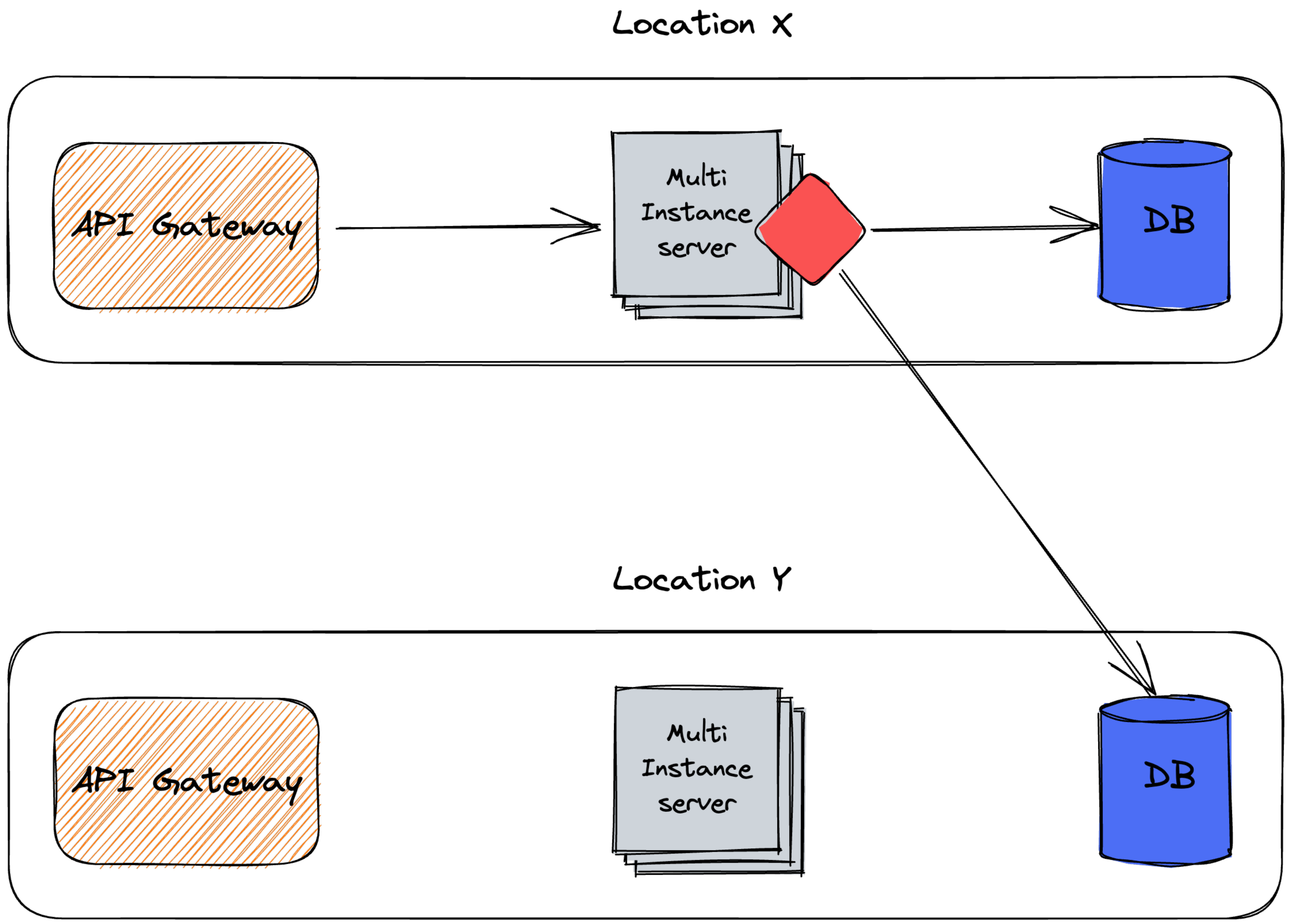
Client-side algorithm

- "Smart" client
- Need to know the topology
- Doesn't work on the first request

@nicolas_frankel

Server-side algorithm

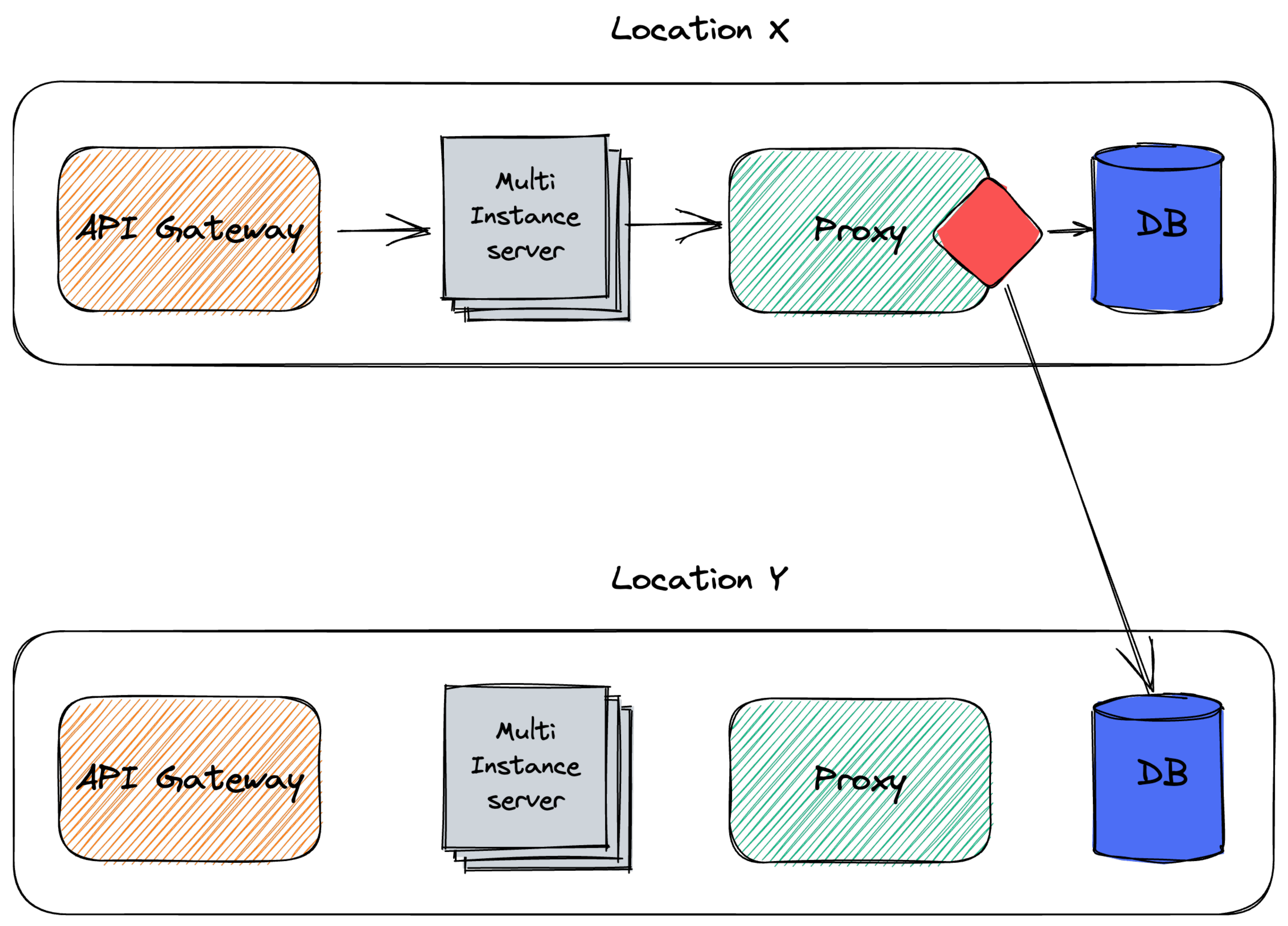
- Most flexible approach
  - Allows fetching additional data to compute location
- Requires code
  - Most error prone
- Performance cost of crossing location



@nicolas_frankel

Proxy-based algorithm

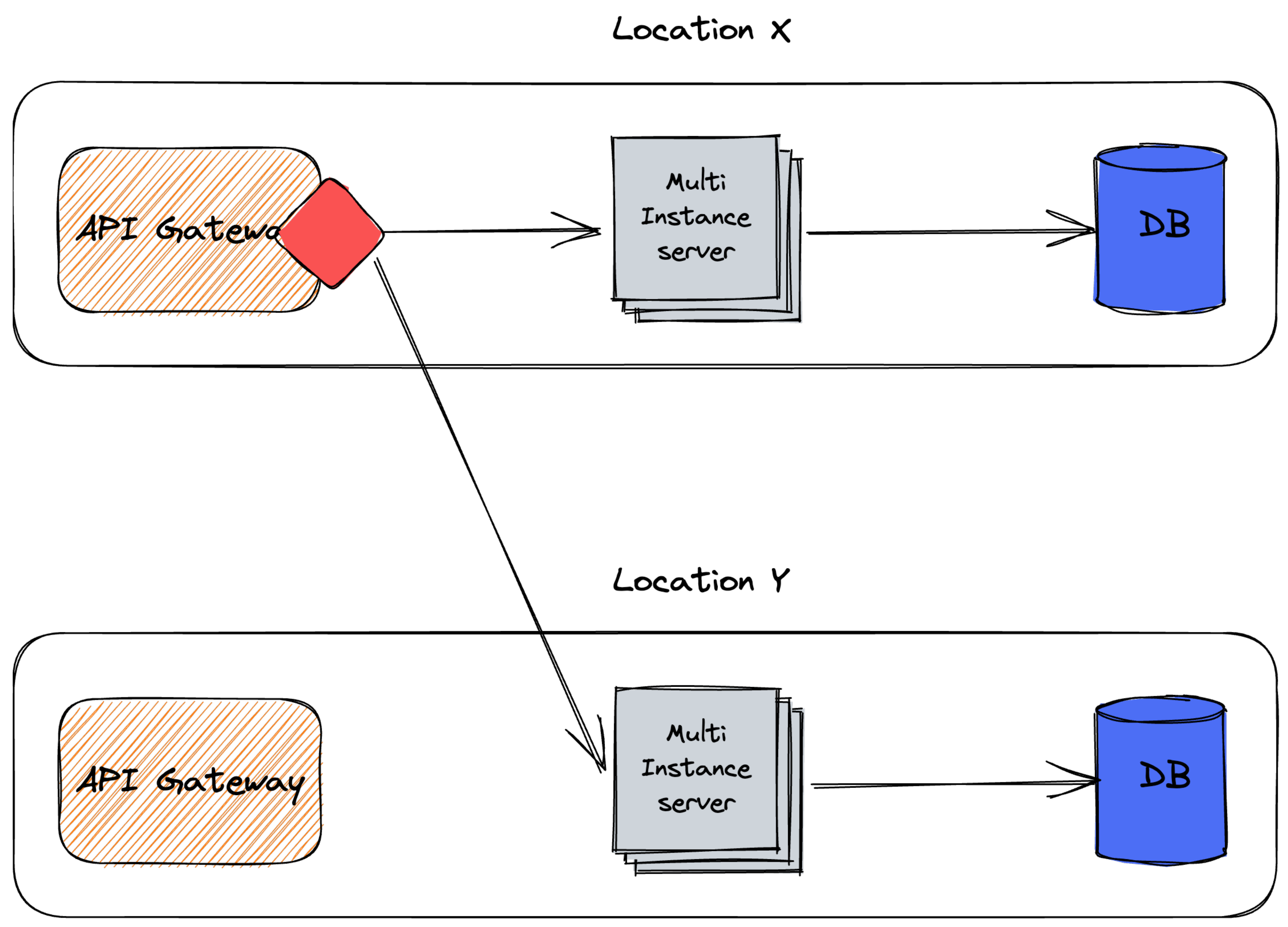
- More complex architecture
  - One more moving piece
- More decoupled
- Same performance cost of crossing location



@nicolas_frankel

API Gateway-based algorithm

- Early decision
- Gateway already knows the upstreams
- No access to the data



A tentative proposal: double computation

1. The first request hits any of the two endpoints
2. The response sends additional metadata to store client-side
3. The second request queries the correct Gateway
4. The Gateway computes the location based on the metadata and forwards it to the correct app
5. The app computes the location again
    1. If it's correct, continue to the database
    2. If it's not, forward to the correct database

Similar to validation implemented client- AND server-side

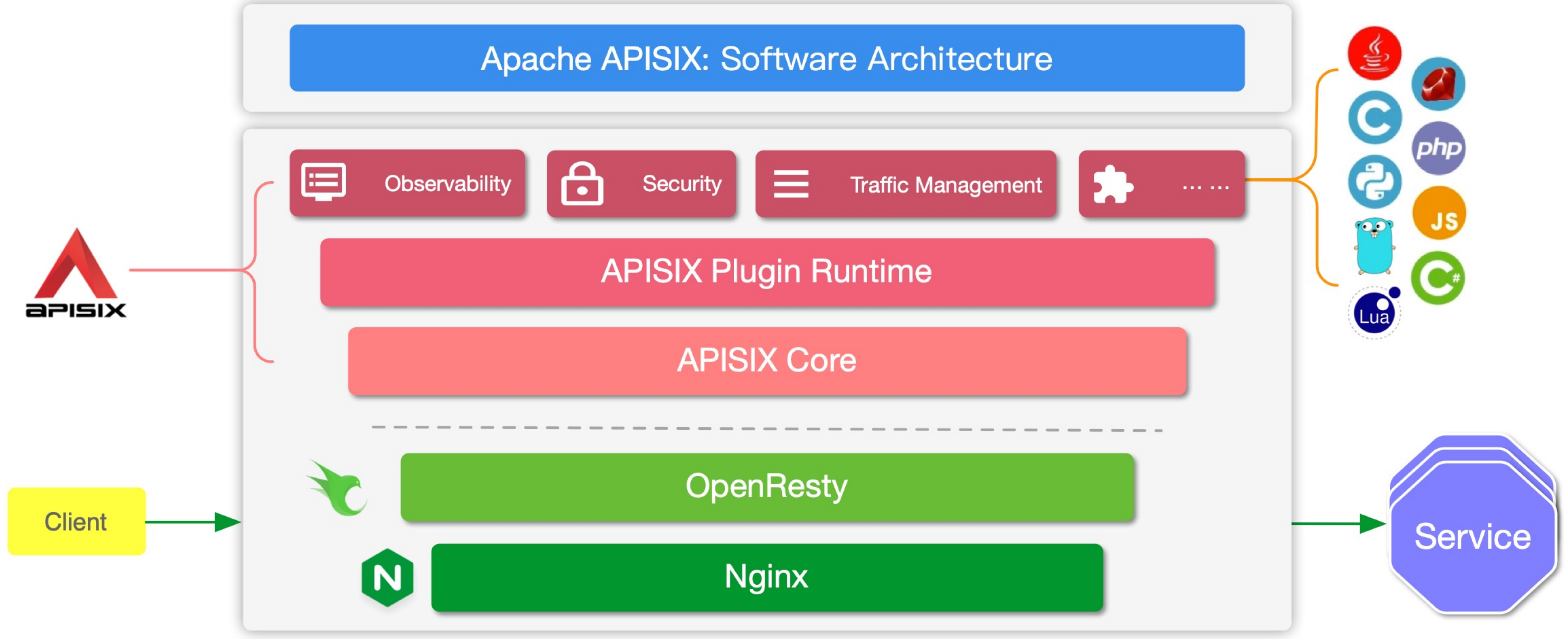
One possible implementation with the Apache Stack

apisix

ShardingSphere

@nicolas_frankel

Apache APISIX, an API Gateway the Apache way



Apache Shardingsphere

“Apache ShardingSphere is an ecosystem to transform any database into a distributed database system, and enhance it with sharding, elastic scaling, encryption features & more.”

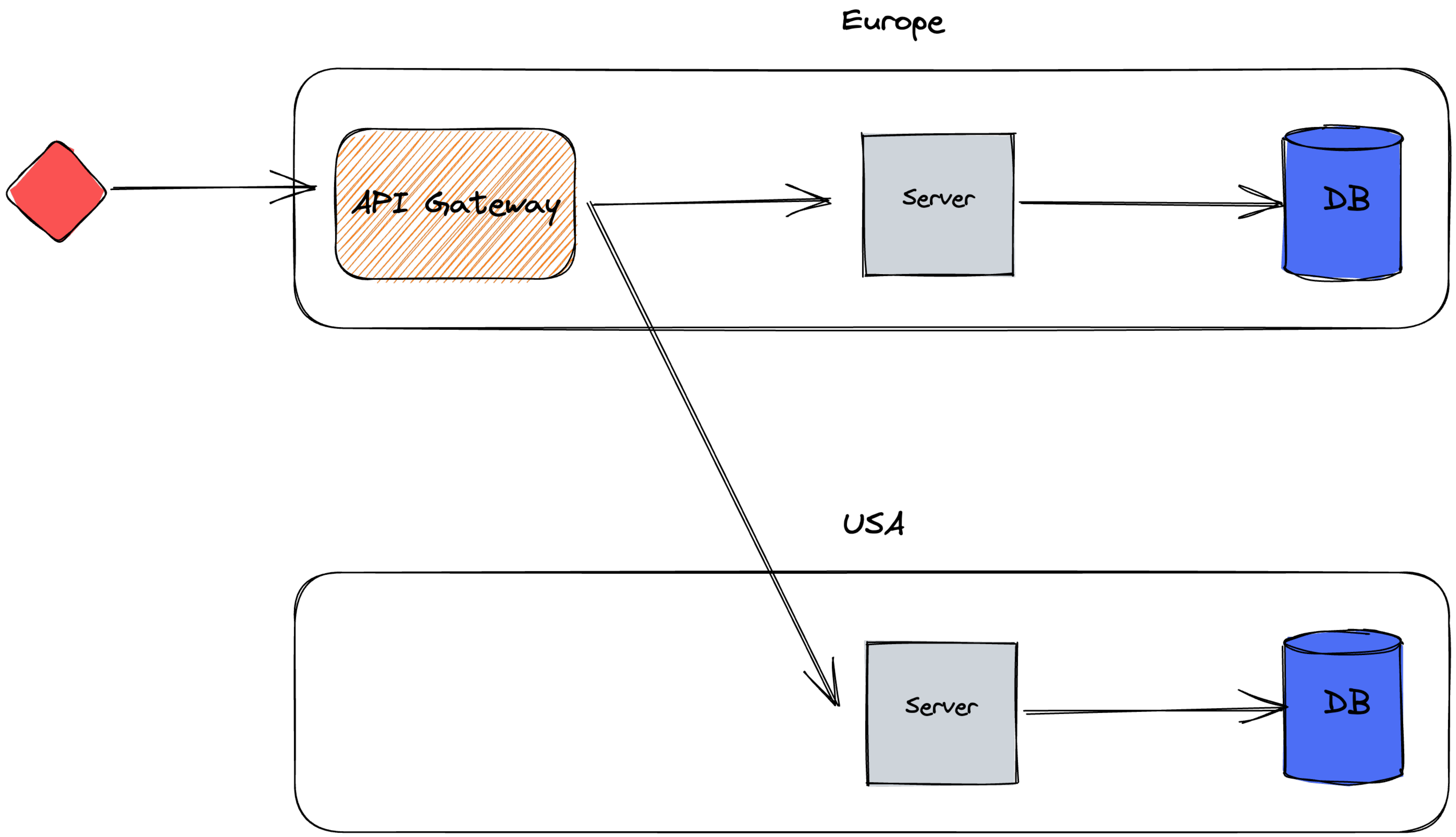
-- https://shardingsphere.apache.org/document/current/en/overview/

Time for DEMO

Architecture, limitations and choices

- No client storage for metadata
  - Single API Gateway
  - Two apps
  - Two databases
- JVM-based app
  - Kotlin
  - Spring Boot
- Shardingsphere via library

Demo set up



Thanks for your attention!

- @nicolas_frankel
- @nico@frankel.ch
- https://bit.ly/dataresid
- https://apisix.apache.org/