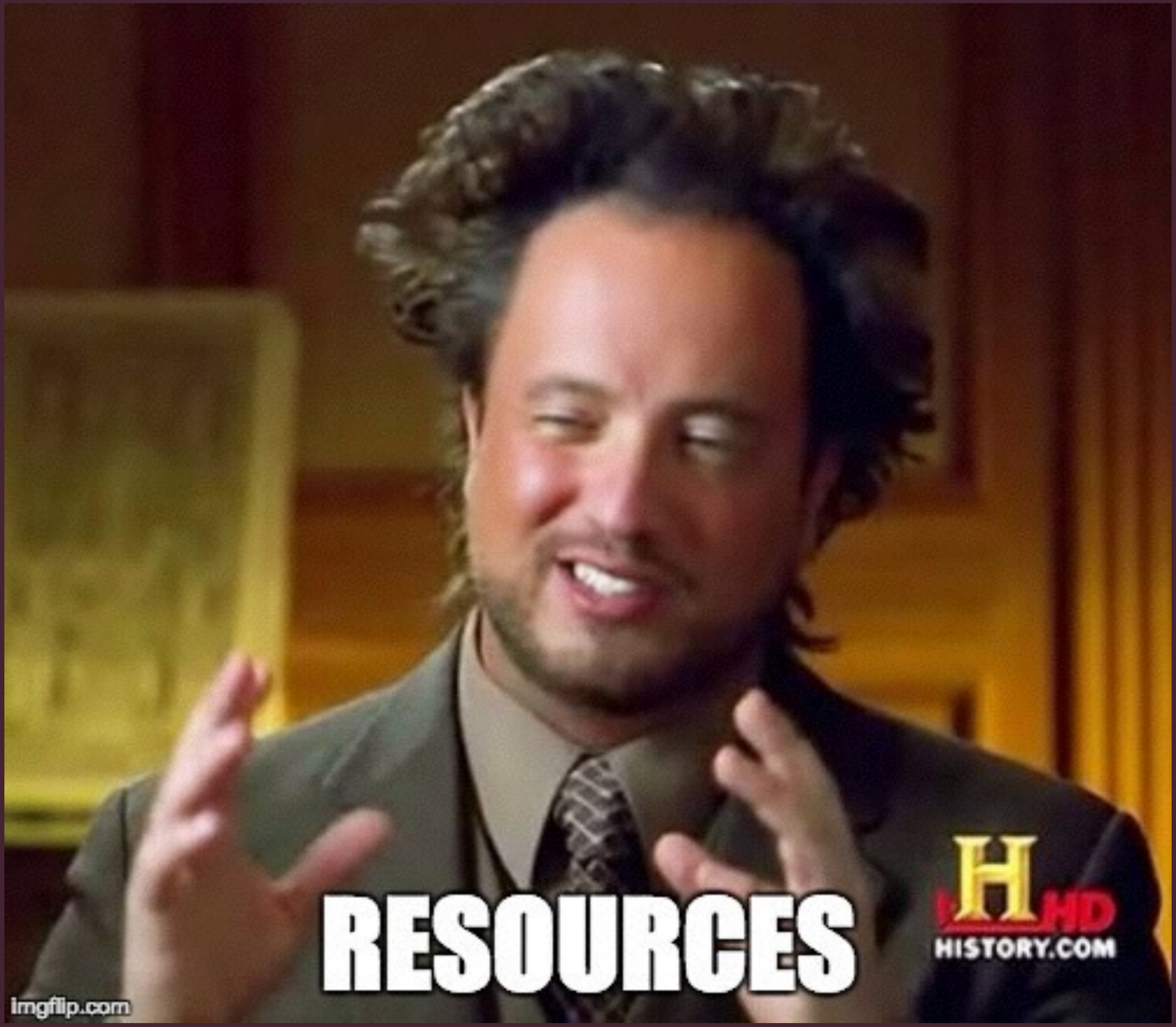
TÉCNICO LISBOA

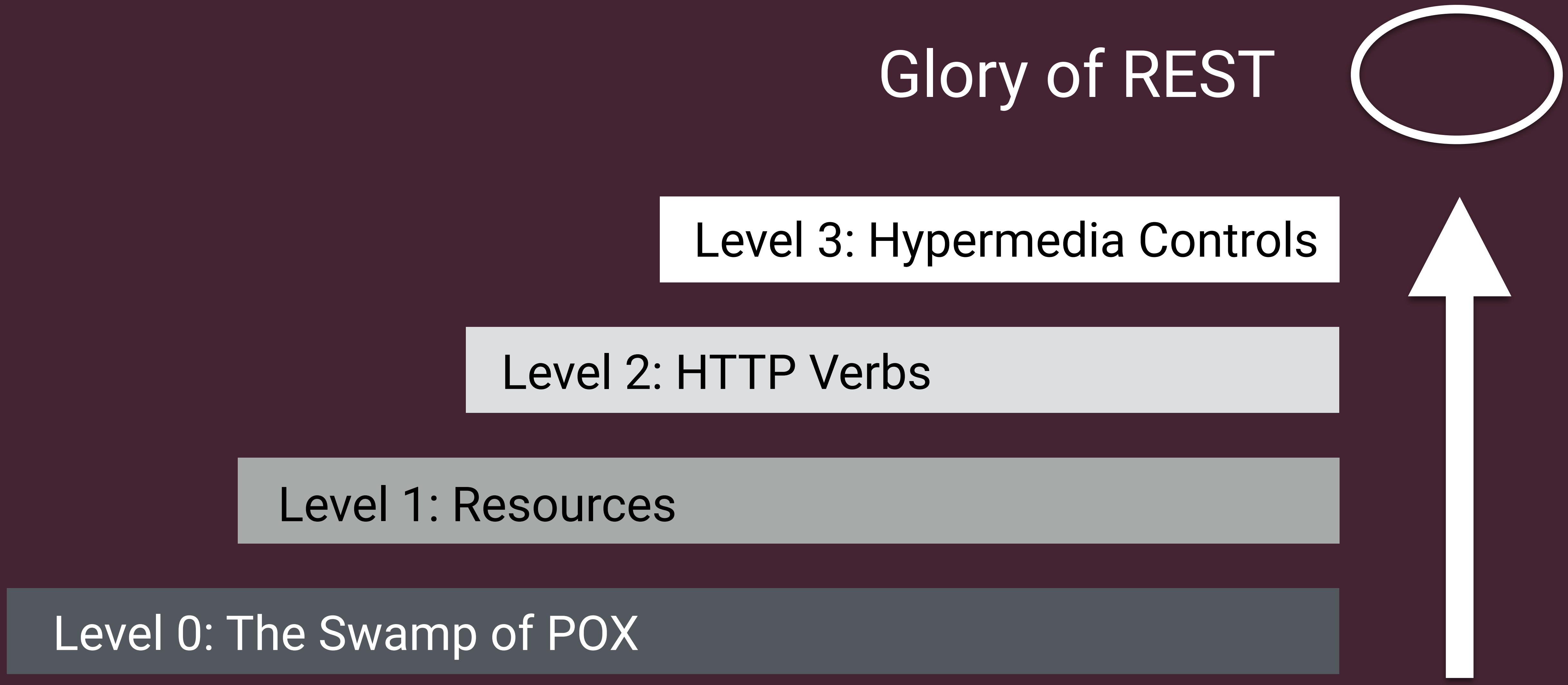
REST APIs & Microservices

DSI Dev Sessions

REpresentational **S**tate **T**ransfer



Richardson Maturity Model



Level 0: The Swamp of Plain Old XML

- HTTP is tunnel protocol
- POST to a single URL
- SOAP/XML-RPC

POST /movie-service

```
<service-invocation>
  <action>create-movie</action>
  <title>Star Wars: Episode IV - A New Hope</title>
</service-invocation>
```

Level 1: Resources

- Entities are resources
- /movies/start-wars instead of /movie-service
- /movies/start-wars/actors

POST /movies/start-wars

```
<service-invocation>
  <action>get-movie-info</action>
</service-invocation>
```

Level 2: HTTP Verbs

- POST or PUT for creations
- GET for retrievals
- PUT for updates
- DELETE for deletions
- PATCH for partial updates

Level 2: HTTP Verbs

POST /movies

```
{
  “title”: “Star Wars: Episode IV - A New Hope”,
  “year”: 1977
}
```

HTTP/1.1 201 Created
Location: /movies/1

```
{
  “id”: 1,
  “title”: “Star Wars: Episode IV - A New Hope”,
  “year”: 1977
}
```

Level 3: Hypermedia Controls

- HATEOAS (Hypertext as the engine of application state)
- Using links to detect your next states

GET /movies/1

```
{
  “id”: 1,
  “title”: “Star Wars: Episode IV - A New Hope”,
  “year”: 1977,
  “_links”: [{
   “href”: “/movies/1/characters”,
   “rel”: “characters”
  }]
}
```

Talking HTTP Codes

HTTP Status Codes

- 1xx: Informational
- 2xx: Successful
- 3xx: Redirection
- 4xx: Client Error
- 5xx: Server Error

How do I login into my API?

You don’t!

Authenticate/Authorize in every request

```
GET /movies
Authorization: Basic ZnJvZG9oOjIzNDIzNDIz

GET /movies/32
Authorization: Basic ZnJvZG9oOjIzNDIzNDIz

POST /movies
Authorization: Basic ZnJvZG9oOjIzNDIzNDIz
```

Transactions

Don’t make them cross multiple resources

POST /accounts/1/transactions
TransID: 556EA21

```
{
  “amount”: “-500”
}
```

POST /accounts/2/transactions
TransID: 556EA21

```
{
  “amount”: “500”
}
```

POST /commit
TransID: 556EA21

Make the transaction as a resource

POST /transfers

```
{
  “from_account”: 1,
  “to_account”: 2,
  “amount”: “500”
}
```

RECAP (1/2)

- Resources should always be nouns (always in plural)
  - /movies/:id/characters
- Actions is the HTTP Verb
  - All verbs except POST and PATCH are idempotent
  - But only GET is safe (read-only)

RECAP (2/2)

- Talk HTTP Status code for your API responses
  - POST 201 Created
  - GET 200 OK
  - PUT 200 Ok
  - DELETE 204 No Content
  - 4xx (Unauthorized, Forbidden, Not Found, …)
- Except for DELETE, all should return a body
  - POST /movies should return

```
{
  “id”: 1,
  “title”: “Star Wars: Episode IV - A New Hope”,
  “year”: 1977
}
```

Live Coding

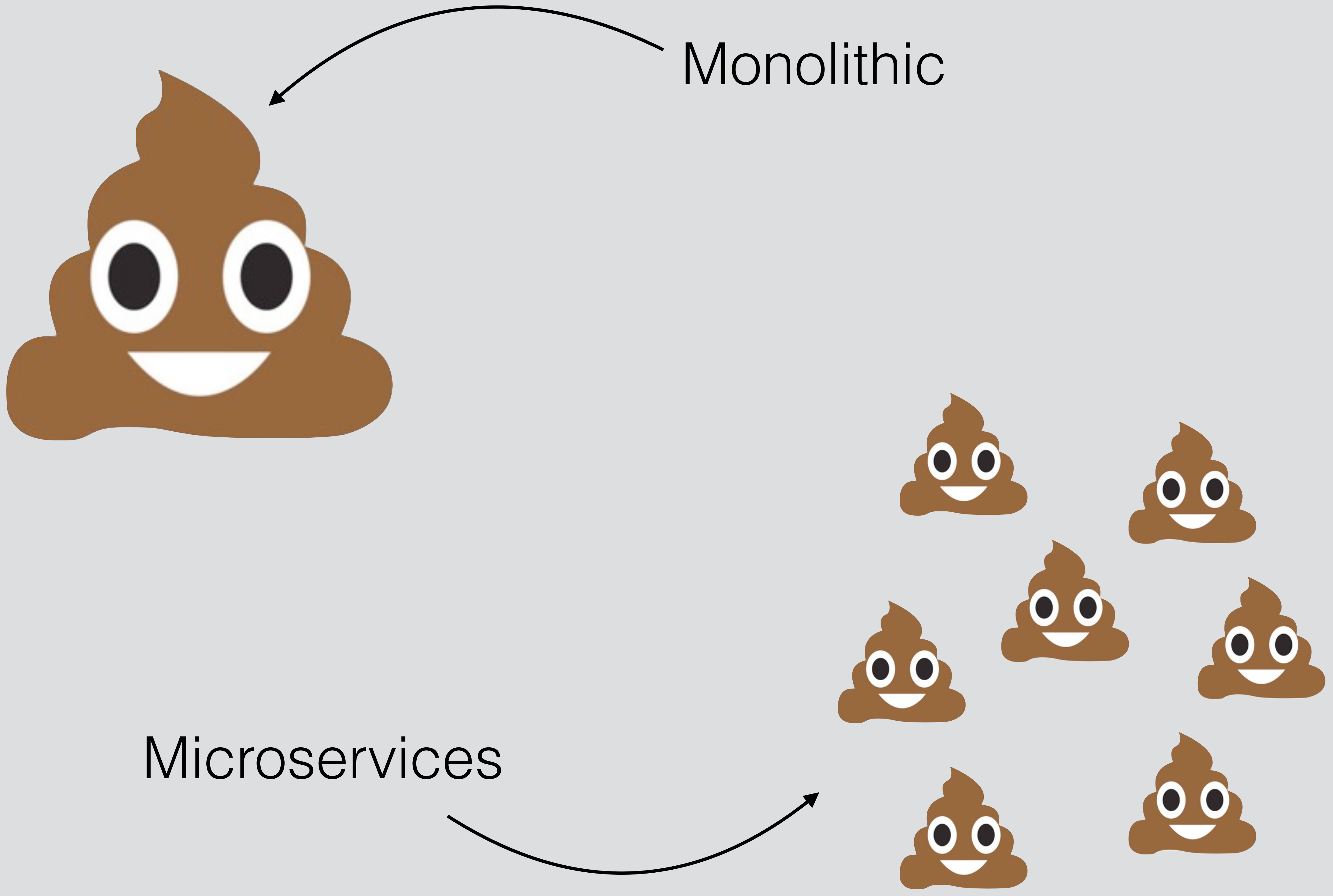
Microservices

Microservices Benefits

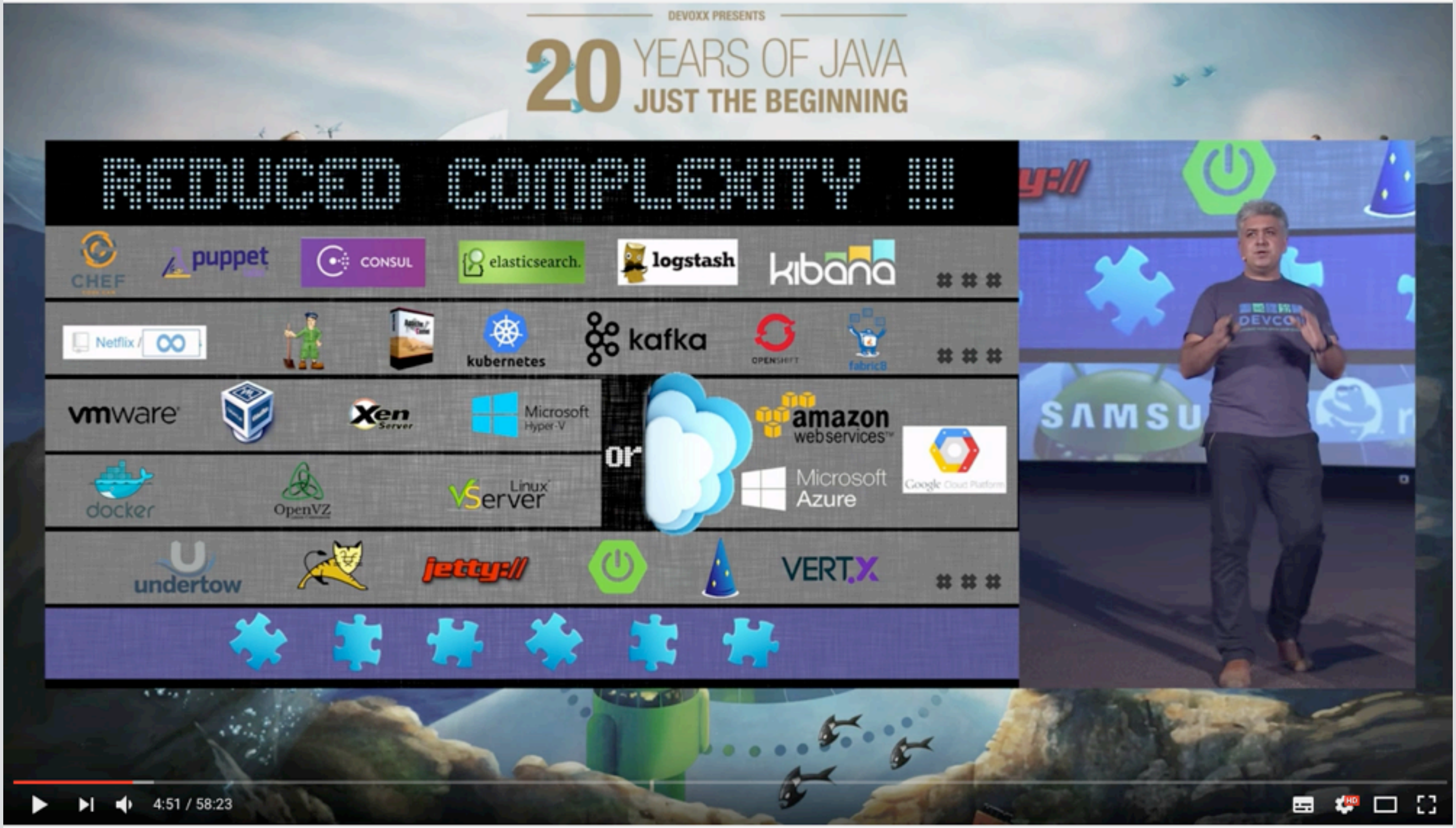
- Strong Module Boundaries
- Independent Deployment
- Technology Diversity

Microservices Costs

- Distribution
- Eventual Consistency
- Operational Complexity



https://twitter.com/alvaro_sanchez



Microservices and Modularity or the difference between treatment and cure! by Milen Dyankov

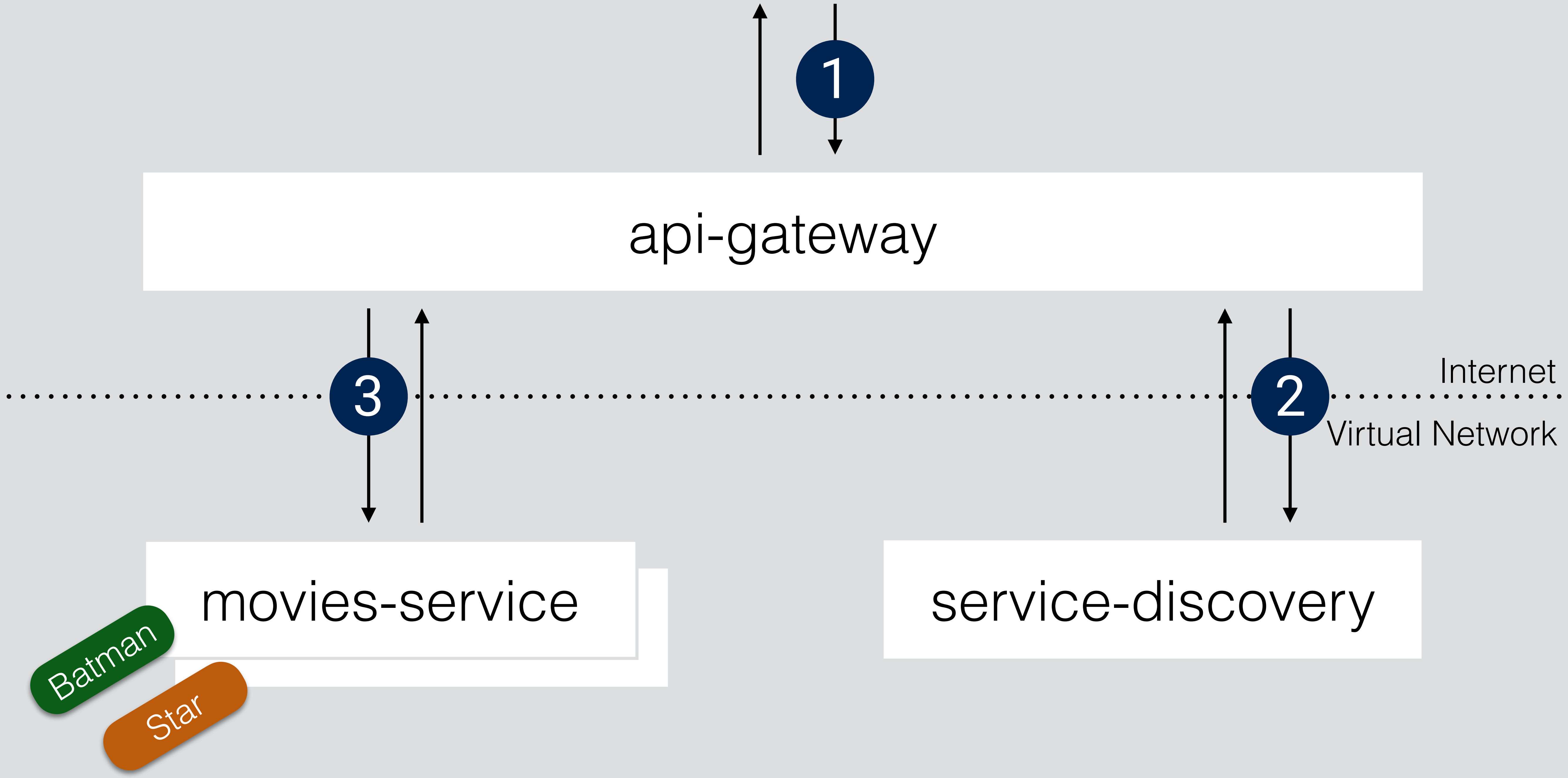
Devoxx

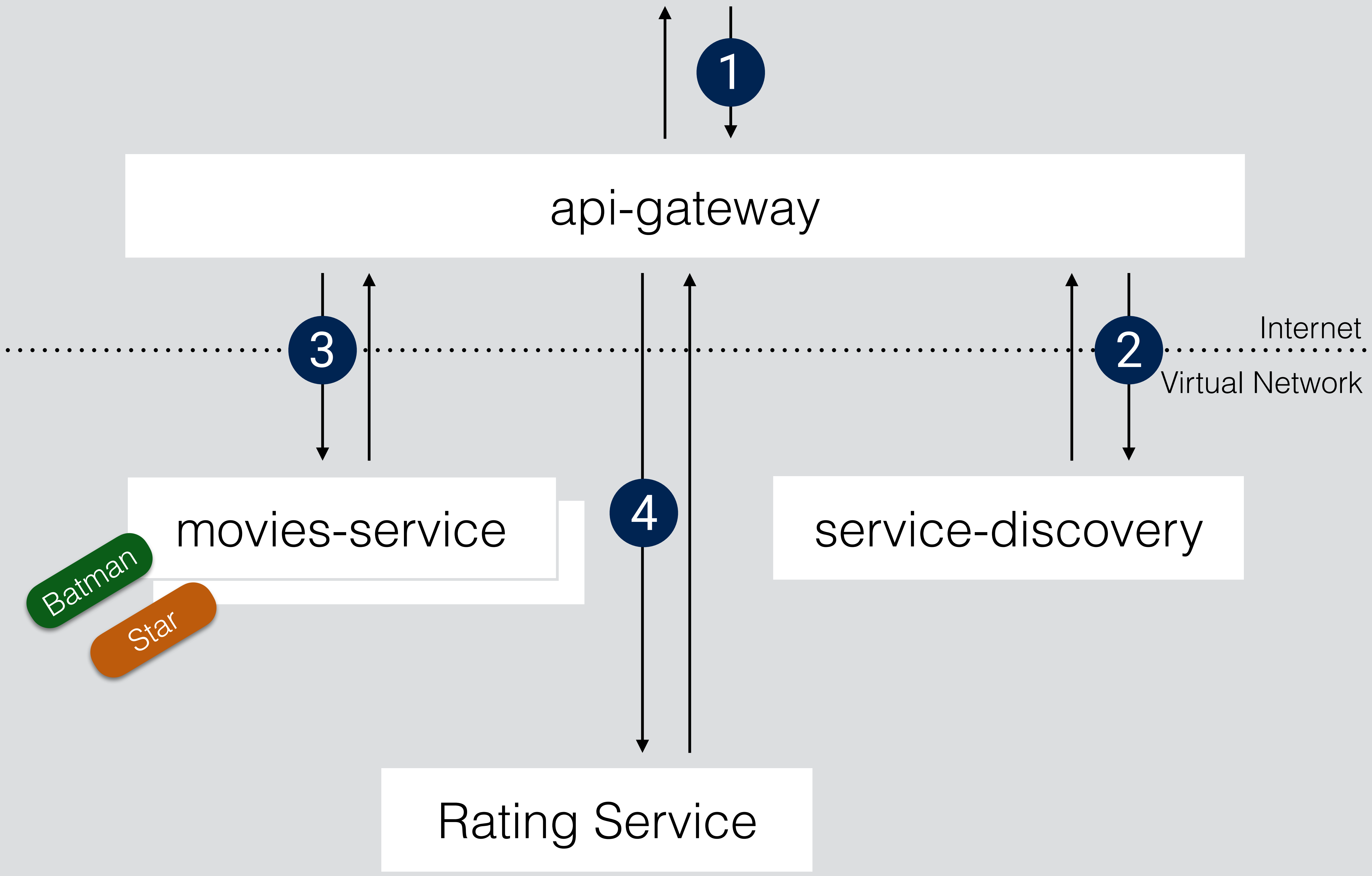
Subscrito 12 639

1 619 visualizações

Adicionar a Partilhar Mais 22 1

DEMO


1
api-gateway
3
2
Internet
Virtual Network
movies-service
Batman
Star
service-discovery


1
api-gateway
3
4
2
Internet
Virtual Network
movies-service
Batman
Star
service-discovery
Rating Service