
# A tale of REST and GraphQL

September 25th, 2019

Section 1

# **Introduction**

# Who are we?

Managing Director @ SheTech

@ChiaraBrughera

chiarabrughera

# Who are we?

CTO SheTech

Senior Software Engineer in Credimi

@pamelagotti84

pamelagotti

# Who are we?

**Tech**

**Inclusione**

**Impatto**

*www.shetechitaly.org*

*hello@shetechitaly.org*

Section 2

# **Theory time!**

# REST

**REpresentational State Transfer**

## Principles

Architectural constraints

- Uniform interface
- Client-server
- Stateless
- Cacheable
- Layered system
- Code on demand

## Resource

Key abstraction of information

- URI
- Representation
- Media type
- Singleton or collection
- May contain sub collections

## Methods

Performing transactions

- Uniform interface
- Not necessary POST/PUT/GET

# Beware!

## Not database related

Despite the name, GraphQL is not an ORM and it is not a replacement of SQL

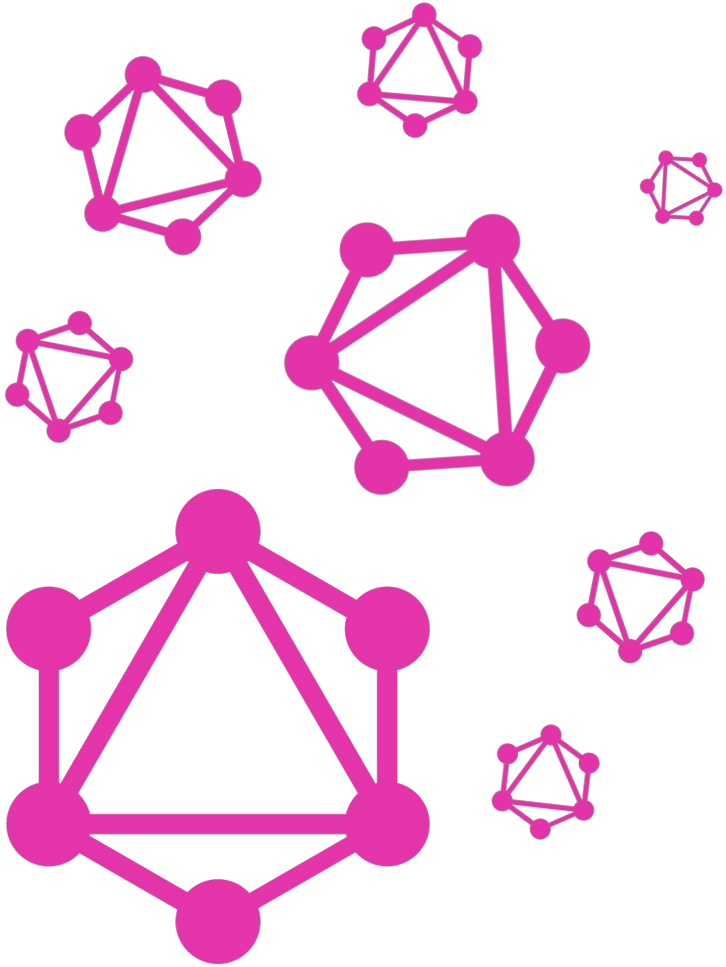
## Not a REST replacement

More an alternative approach

# GraphQL

## Modern approach to data access

- Resolvers
- No more over or under fetching
- Get many resources in a single fetch
- Typed system
- No need for API versioning
- Push data from server to client
- Great ecosystem



# Comparison

## REST

- Shape and size of resource determined by the server
- Multiple calls to fetch related resources
- Clear errors management
- Easy monitoring
- Easy caching at network layer

- Predictible complexity

## GraphQL

- Description of a resource not coupled to the way you fetch it
- Multiple resources retrieved traversing the graph
- Errors return 200
- Single endpoint requires more elaborate monitoring
- Up to the developer to ensure that caching is correctly implemented
- n+1 problem

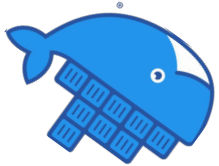
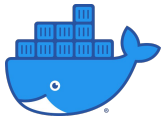
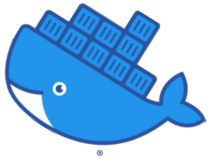
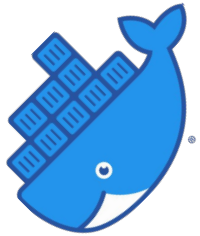

ENOUGH BLAH BLAH
SHOW ME SOME CODE!
makeameme.org

Section 2
# **Hands on!**

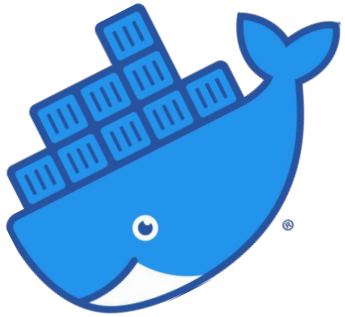
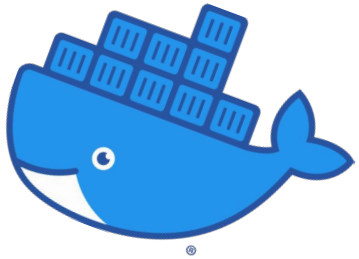
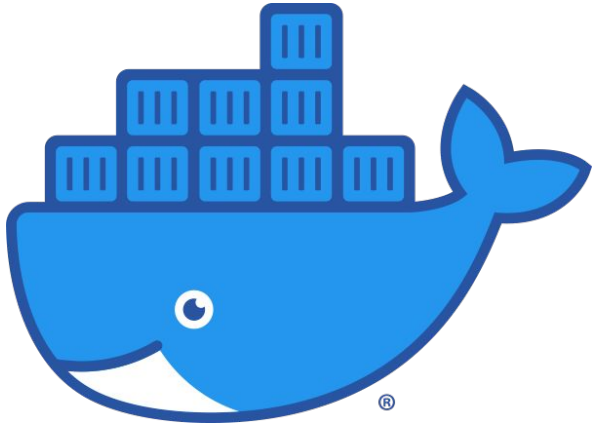
# Show me the code!

https://github.com/shetechitaly/graphql-workshop

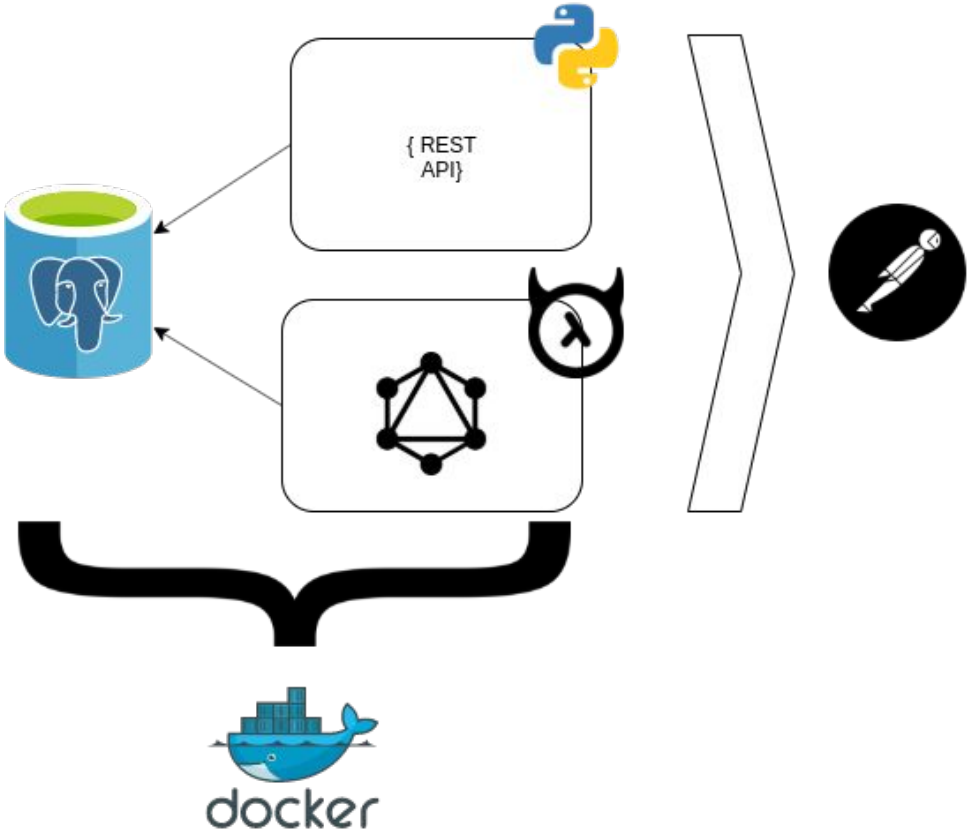
# Show me the code!

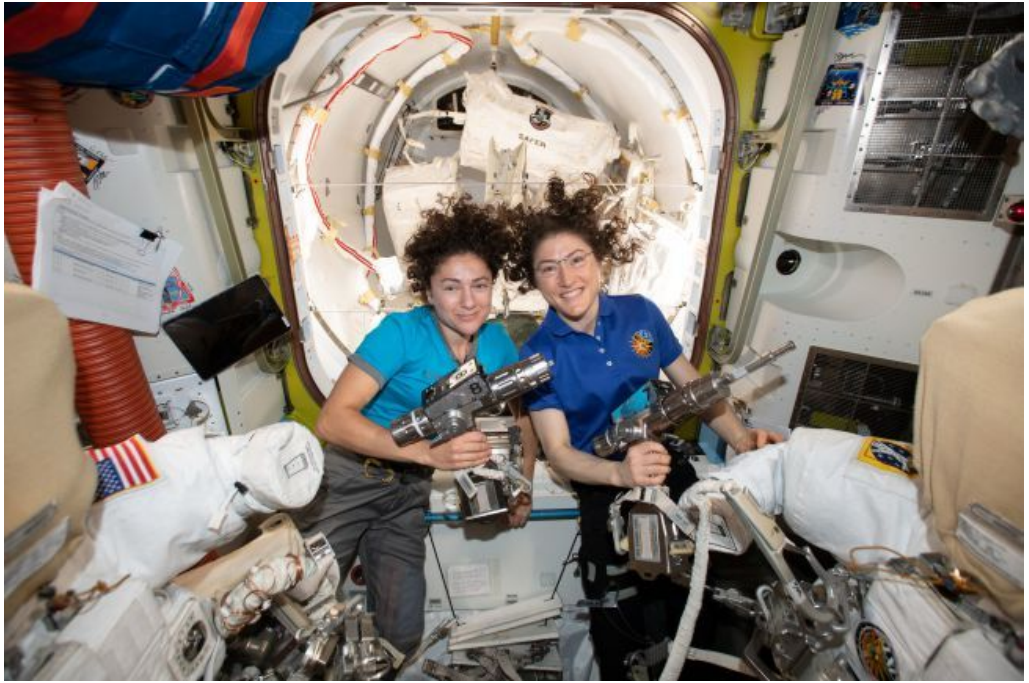
docker-compose up

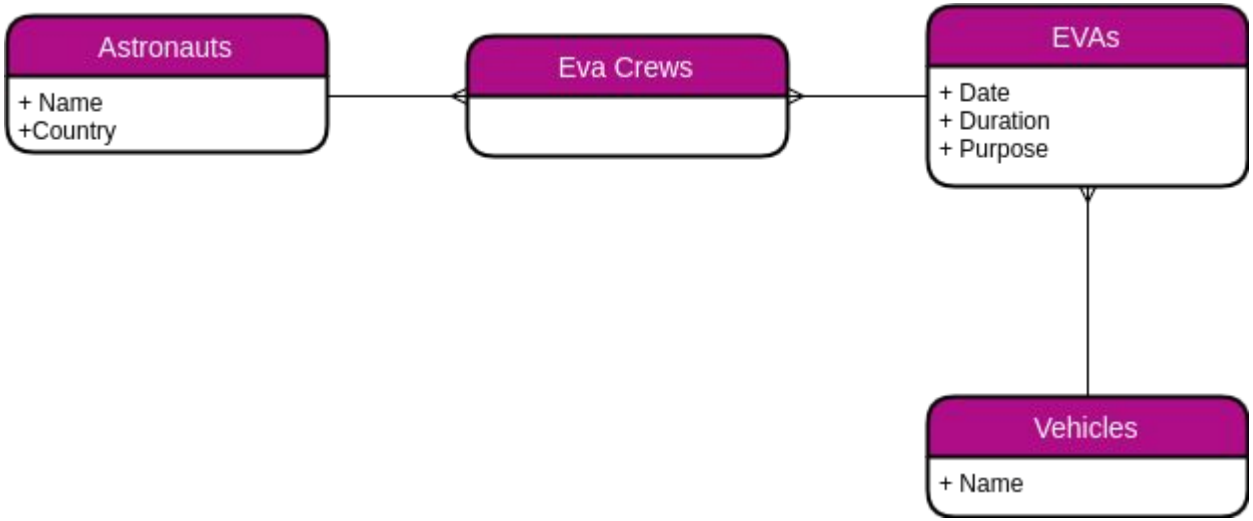
# Setup



# EVAs!



# Dataset




https://data.nasa.gov/Raw-Data/Extra-vehicular-Activity-EVA-US-and-Russia/9kcy-zwvn

# REST approach

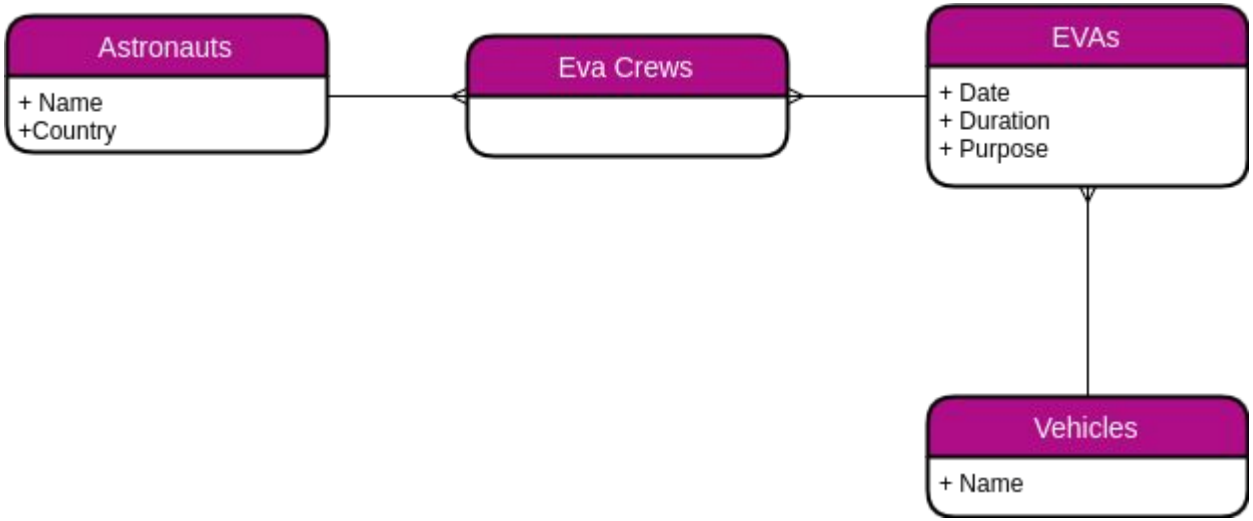
## Resources

- astronauts
- vehicles
- evas

## Endpoints

- GET/astronauts
- GET/astronauts/{id}
- GET/astronauts/{id}/evas
- GET/vehicles
- GET/vehicles/{id}
- GET/vehicles/{id}/evas
- GET/evas
- ...

# Resources graph



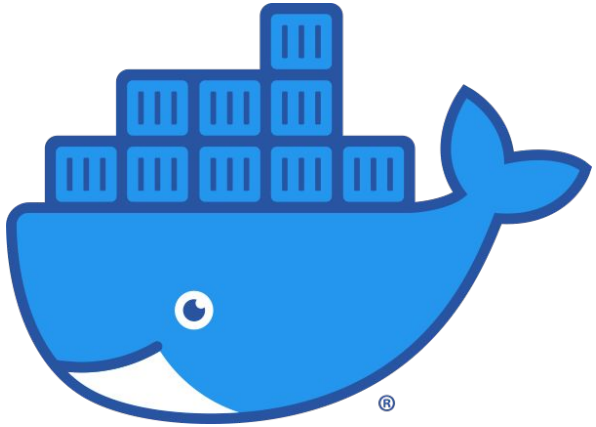
# Test time!

Rest

http://localhost:5000/

GraphQL

http://localhost:8080/console



# Query - overfetching

Return the names of all astronauts

## Rest

GET/ http://localhost:5000/api/astronauts

## GraphQL

POST http://localhost:8080/v1/graphql

```
{
  astronauts {
    name
  }
}
```

# Query - underfetching

Return the names of the vehicles where Buzz Aldrin was on

## Rest

GET/ http://localhost:5000/api/astronauts?q={"filters":[{"name":"name","op":"eq","val":"Buzz Aldrin"}]}

GET/ http://localhost:5000/api/vehicles/1

GET/http://localhost:5000/api/vehicles/13

## GraphQL

POST http://localhost:8080/v1/graphql

```
{
  astronauts(where: {name: {_eq: "Buzz Aldrin"}}) {
    evas_crews {
      eva {
        vehicle {
          name
        }
      }
    }
  }
}
```

# Query - API versioning

## Expose a new column country on astronauts

### Rest

GET/ http://localhost:5000/api/astronauts
GET/ http://localhost:5000/api/astronauts2

### GraphQL

POST http://localhost:8080/v1/graphql

```
{
  astronauts {
    name
    country
  }
}
```

# Query - errors management

## Select a vehicle that does not exist

### Rest

GET/ http://localhost:5000/api/vehicles/abc

### GraphQL

POST http://localhost:8080/v1/graphql

```
query {
  vehicles(where: {id: {_eq: "abc"}}) {
     name
  }
}
```

# Mutation

## Change the purpose of the first EVA

### Rest

```
PATCH/ http://localhost:5000/api/evas/1
{
        "purpose": "First EVA"
}
```

### GraphQL

```
POST http://localhost:8080/v1/graphql
mutation {
  update_evas(_set: {purpose: "First eva"}, where: {id: {_eq: 1}}) {
    affected_rows
  }
}
```

# Subscription

Get updates when Svetlana Savitskaya goes spacewalking again

## Rest

GET
http://localhost:5000/api/astronauts?q={"filters":[{"name":"name","op":"eq","val":"Svetlana Savitskaya"}]}

aaaaand polling!

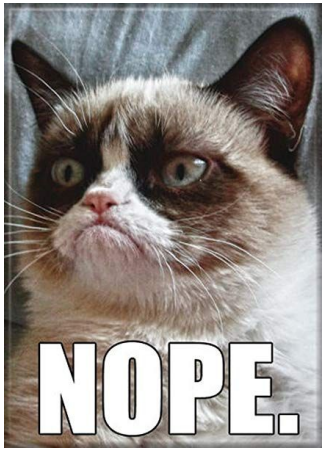
## GraphQL

```
subscription {
  astronauts(where: {name: {_eq: "Svetlana Savitskaya"}}) {
    evas_crews {
      eva {
        eva_date
        duration
        purpose
      }
    }
  }
}
```

# Query - type introspection

## Query GraphQL schema

### Rest



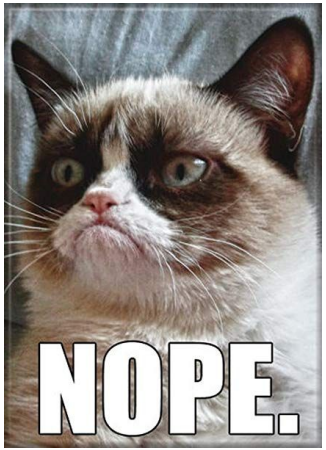

### GraphQL

```
query {
  __schema {
    types {
      kind
      name
      fields {
        name
        description
        type {
          name
        }
      } } } }
```

# Authorization

Allow column level authorization

Rest




GraphQL (Hasura)

# That’s all folks!
# Questions?

Section 3
# **Resources**

# Resources

- Rest APIs: https://restfulapi.net/
- Rest dissertation by Roy Fielding https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
- GraphQL specifications by Facebook: https://graphql.github.io/graphql-spec/June2018/
- Fullstack tutorial: https://www.howtographql.com/
- GraphQL resources: https://github.com/chentsulin/awesome-graphql
- Resolvers: https://medium.com/paypal-engineering/graphql-resolvers-best-practices-cd36fdbcef55
- N+1 problem:
  https://engineering.shopify.com/blogs/engineering/solving-the-n-1-problem-for-graphql-through-batching