Conception d'une application serverless avec GraphQL et Kotlin.

Cabane.io

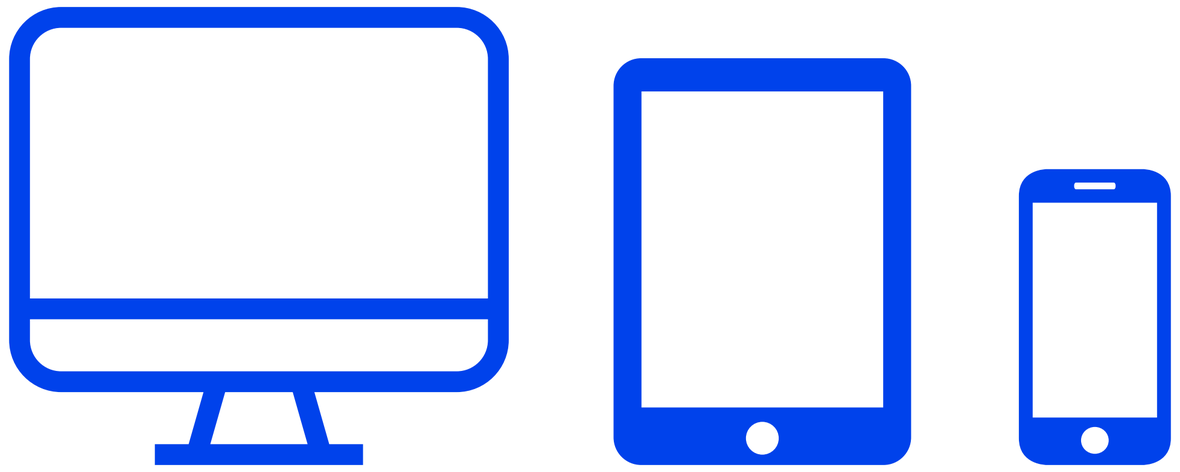
Qui sommes-nous?

William Garneau

Dominique Richard

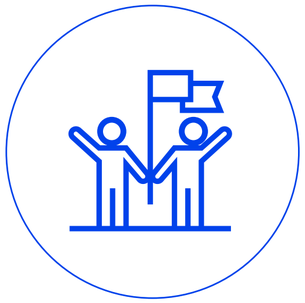
NEXAPP

TALENTS EN APPLICATION



Quelle techno prendre?

Ce qu'on demande.



Évolutif ?

Peu coûteux ?

Prise en main ?

Comment faire ?

Décision #1

GraphQL

Appel vers l'API Rest Heroes

/heroes/123}

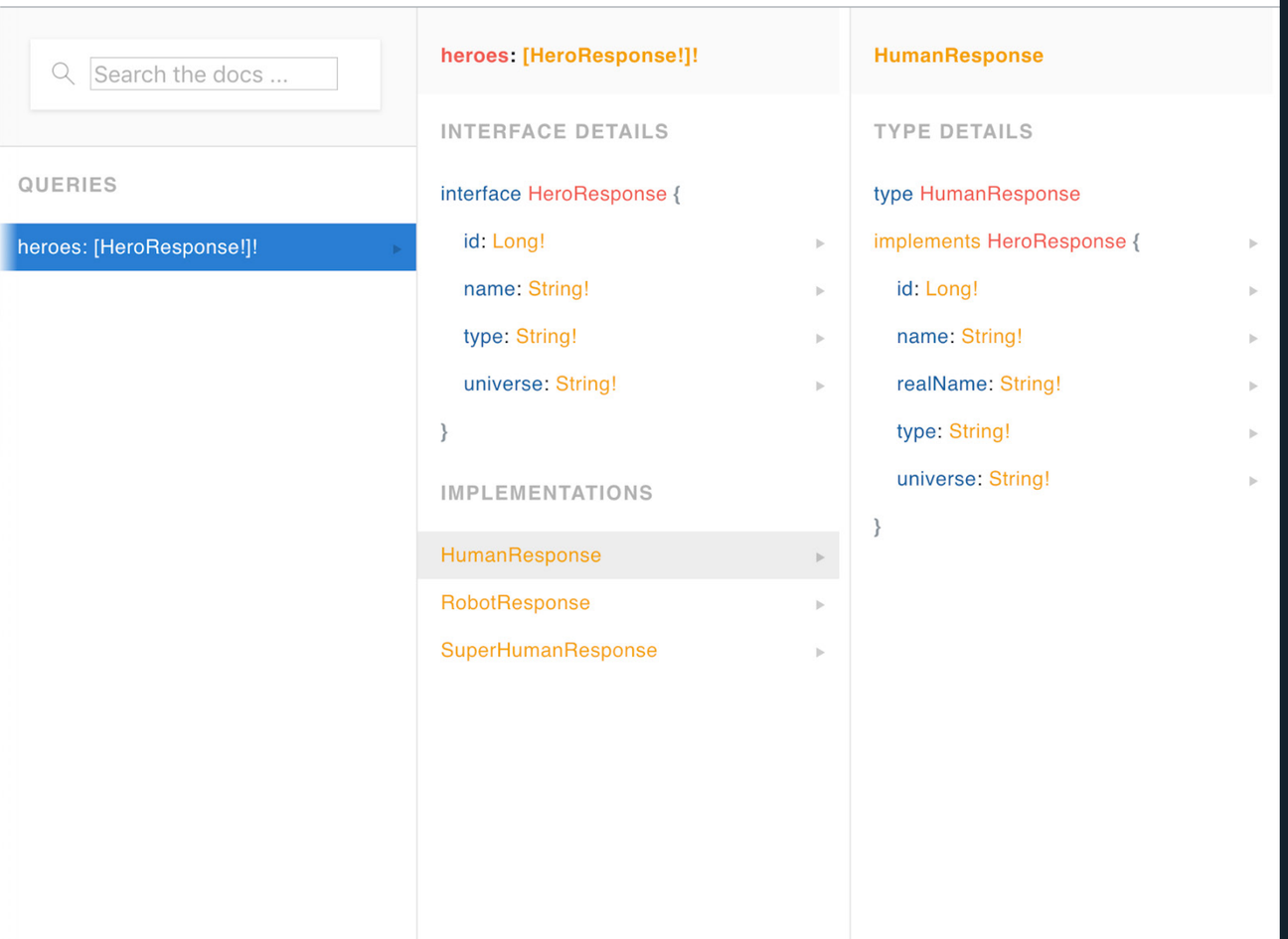

```
{
  "hero": {
    "id": "123",
    "name": "Invincible",
    "universe": "DARK_HORSE",
    "type": "SuperHuman",
    "superPower": "Super strength, super speed",
    "realName": "Mark"
  }
}
```


Appel vers l'API GraphQL Heroes

```
query getHero {
    hero(id: 123) {
        name
        universe
    }
}
```


```
{
  data:{
    "hero": {
        "name": "Invincible",
        "universe": "DARK_HORSE",
    }
  }
}
```

Shéma




```
interface HeroResponse {
  id: Long!
  name: String!
  type: String!
  universe: String!
}

type HumanResponse implements HeroResponse {
  id: Long!
  name: String!
  realName: String!
  type: String!
  universe: String!
}
```

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


```
"name": "HeroResponse",
"fields": [
  {
    "description": null,
    "name": "id"
  },
  {
    "description": "Name of the hero",
    "name": "name"
  },
  {
    "description": null,
    "name": "type"
  },
  {
    "description": "Name of the publisher owning the rights to this character",
    "name": "universe"
  }
```

Mutation

```
mutation addHero($heroInput: HeroInput!) {
  addHeroMutation(heroInput: $heroInput) {
    name
    universe
  }
}
```

```
{
  "heroInput": {
    "name": "Invincible",
    "universe": "DARK_HORSE",
    "realName": "Mark",
    "type": "SuperHuman",
    "superPower": "Super strength, super speed"
  }
}
```

Mutation

```
{
  "heroInput": {
    "name": "Invincible",
    "universe": "Marv",
    "realName": "Mark",
    "type": "SuperHuman",
    "superPower": "Super strength, super speed"
  }
}
```

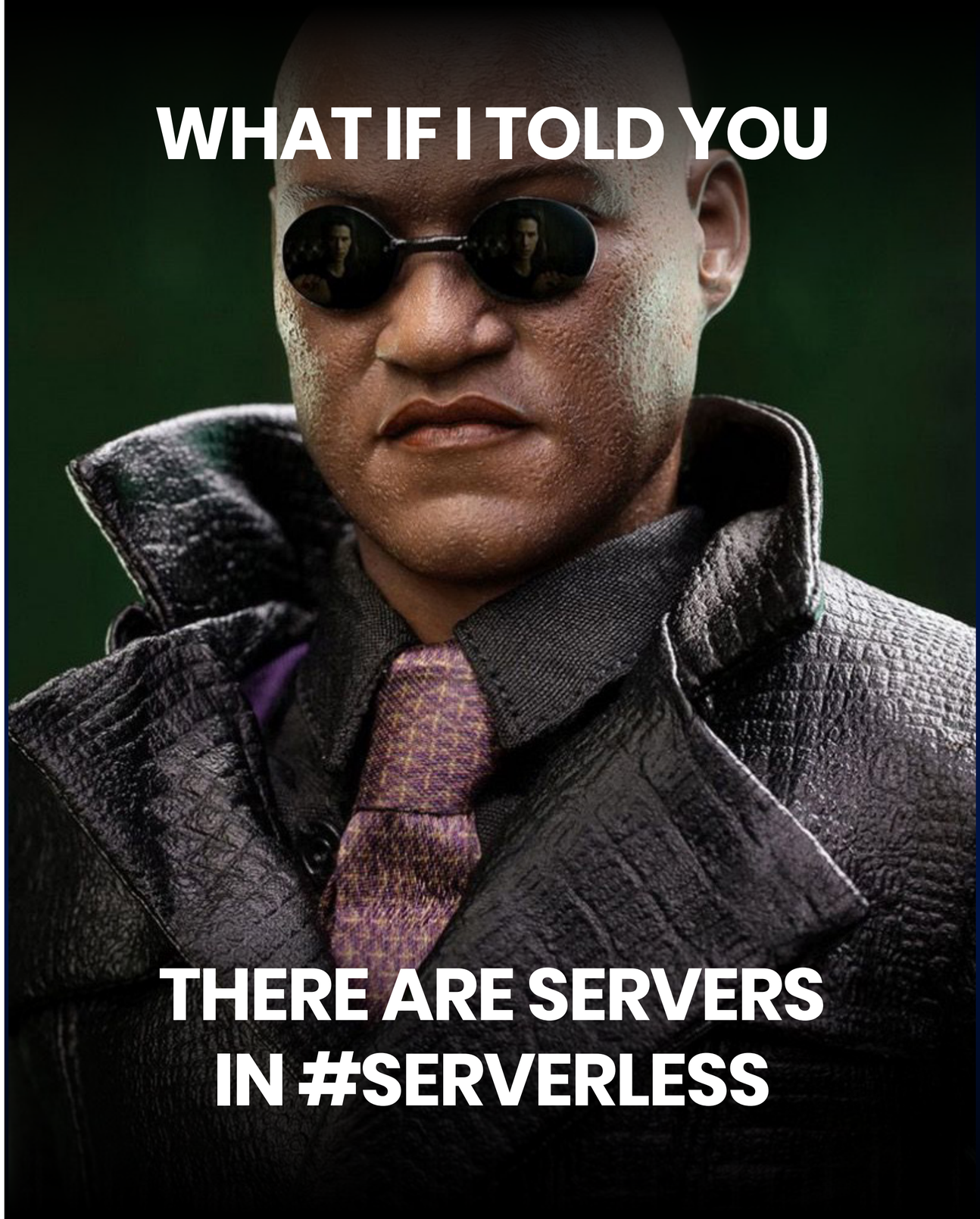

```
{
  "errors": [
    {
      "message": "Variable 'universe' has an invalid value. Invalid input for Enum 'ComicUniverse'. No value found for name 'Marv'",
      "locations": [
        {
          "line": 1,
          "column": 18,
          "sourceName": null
        }
      ],
      "errorType": "ValidationError",
      "path": null,
      "extensions": null
    }
  ]
}
```

Décision # 2

Function-as-a-service

Détrompez-vous...

On a encore des serveurs !



Function-as-a-service

- **Container**
- **Géré par le cloud provider**
- **Déployé au besoin**
- **Chargé au besoin ($)**

Serverless

- **Infrastructure as code**
- **Ça supporte plusieurs fournisseurs de cloud différents?**
- **Déploiement facile et rapide** (Serverless deploy)

Serverless.yml

```yaml
service: serverless-backend

custom:
  allowedHeaders:
    - Content-Type
    - X-Amz-Date
    - Authorization
    - X-Api-Key
    - X-Amz-Security-Token
    - X-Amz-User-Agent

provider:
  name: aws
  runtime: java8
  region: ${env:AWS_DEFAULT_REGION}
  stage: dev
  timeout: 10

package:
  artifact: build/libs/backend-dev.jar

functions:
  graphql:
    handler: com.serverlessorder.backend.api.serverless.GraphQLHandler
    events:
      - http:
          path: graphql
          method: post
          cors:
            origin: '*'
            headers: ${self:custom.allowedHeaders}
```

Décision # 3

Kotlin

Handler

```kotlin
package com.heroesAndVillains.backend.api.serverless

import com.amazonaws.services.lambda.runtime.Context
import com.amazonaws.services.lambda.runtime.RequestHandler
import com.fasterxml.jackson.databind.ObjectMapper
import com.fasterxml.jackson.module.kotlin.readValue
import com.heroesAndVillains.backend.api.graphql.ApiGatewayResponse
import com.heroesAndVillains.backend.api.graphql.GraphQLEndpoint
import com.heroesAndVillains.backend.api.graphql.GraphQLRequest


class GraphQLHandler : RequestHandler<Map<String, Any>, ApiGatewayResponse> {

    override fun handleRequest(input: Map<String, Any>, context: Context): ApiGatewayResponse {
        val objectMapper = ObjectMapper()
        val request: GraphQLRequest = objectMapper.readValue(input["body"] as String)

        val result = GraphQLEndpoint().execute(request)

        return ApiGatewayResponse.build {
            statusCode = 200
            objectBody = result
            headers = mapOf("Access-Control-Allow-Origin" to "*")
        }
    }
}
```

GraphQL + Kotlin + Lambda

Merci.

William Garneau & Dominique Richard

NEXAPP
TALENTS EN APPLICATION