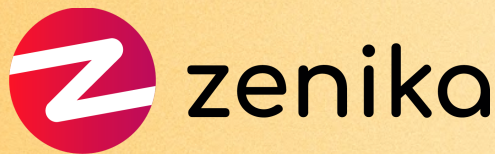


Comment j'ai largué Apollo Server pour GraphQL Yoga



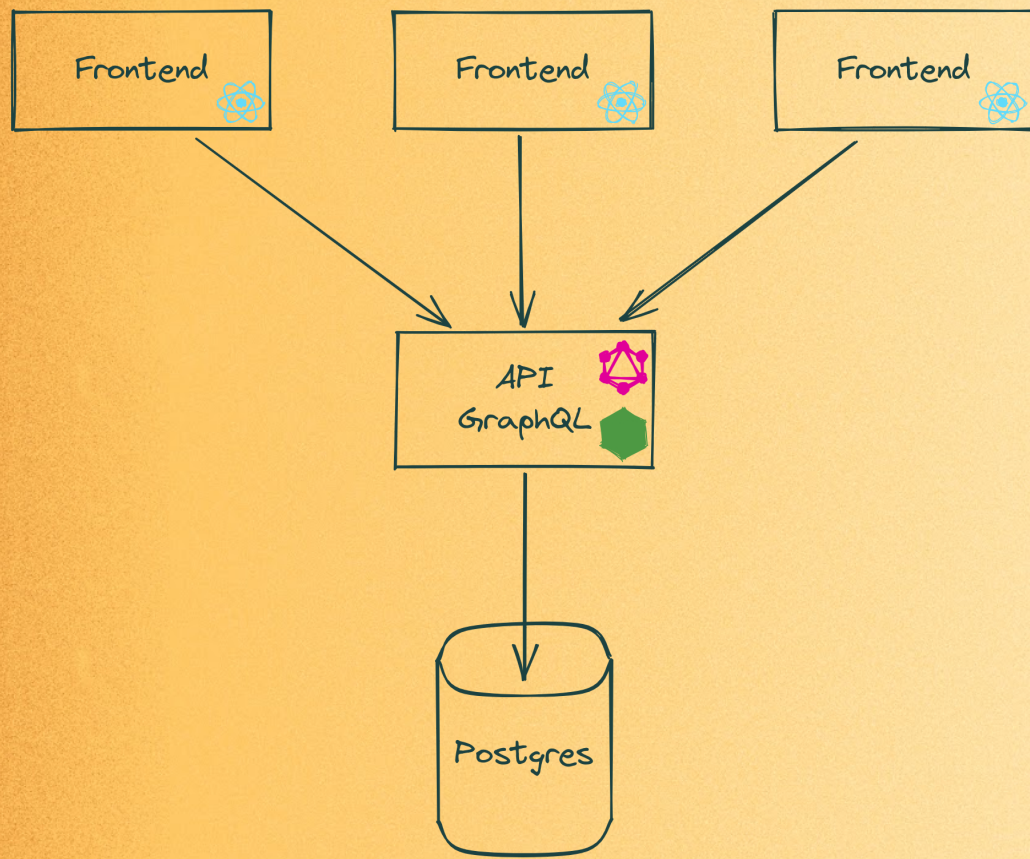
Nicolas Lepage - [@njblepage](https://twitter.com/njblepage)



Le projet

- Développé par Zenika Nantes
- Pour l'Institut Catholique de Vendée
- Application de gestion

Le projet





GraphQL

```
query getStudents {  
  students {  
    id, firstName, lastName  
    grades {  
      id, grade  
      subject {  
        id, name  
      }  
    }  
  }  
}
```

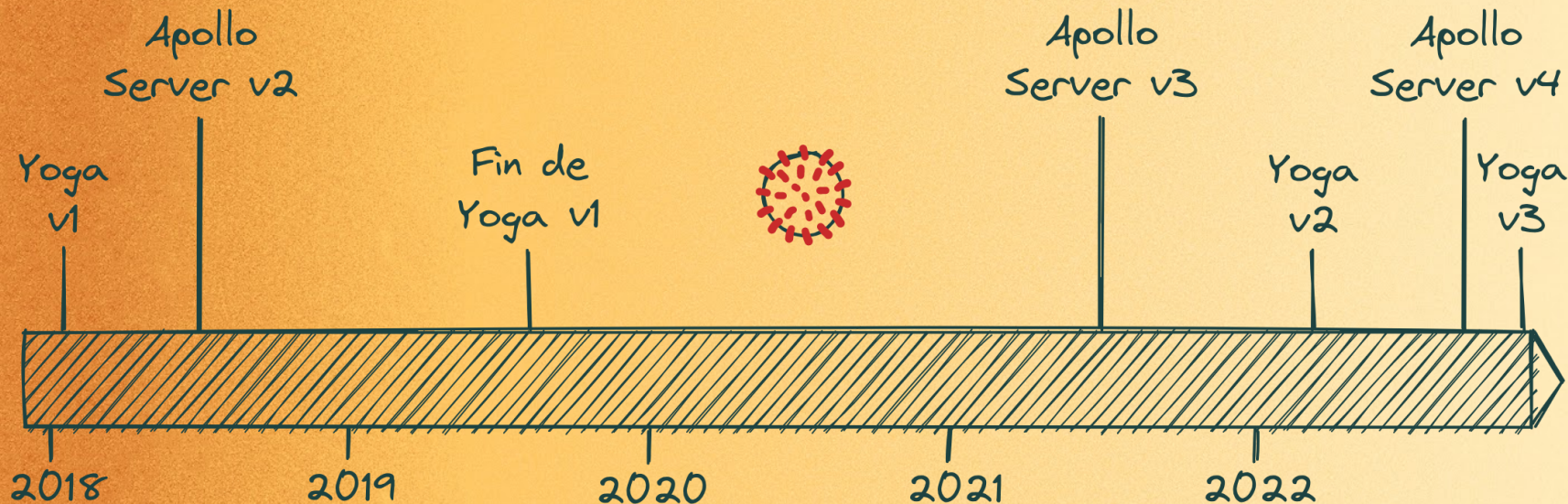
```
type Query {  
  students: [Students]!  
}  
  
type Student {  
  id: ID!  
  firstName: String!  
  lastName: String!  
  grades: [Grade!]!  
}  
  
# ...
```

- Langage de requêtage + Schéma
- Environnement d'exécution
- Spécification opensource

L'API GraphQL du projet

- +100 queries, +100 mutations, +250 types
- Quelques subscriptions
- Uploads de fichiers
- D'abord Apollo Server v1 puis v2

Historique de l'écosystème



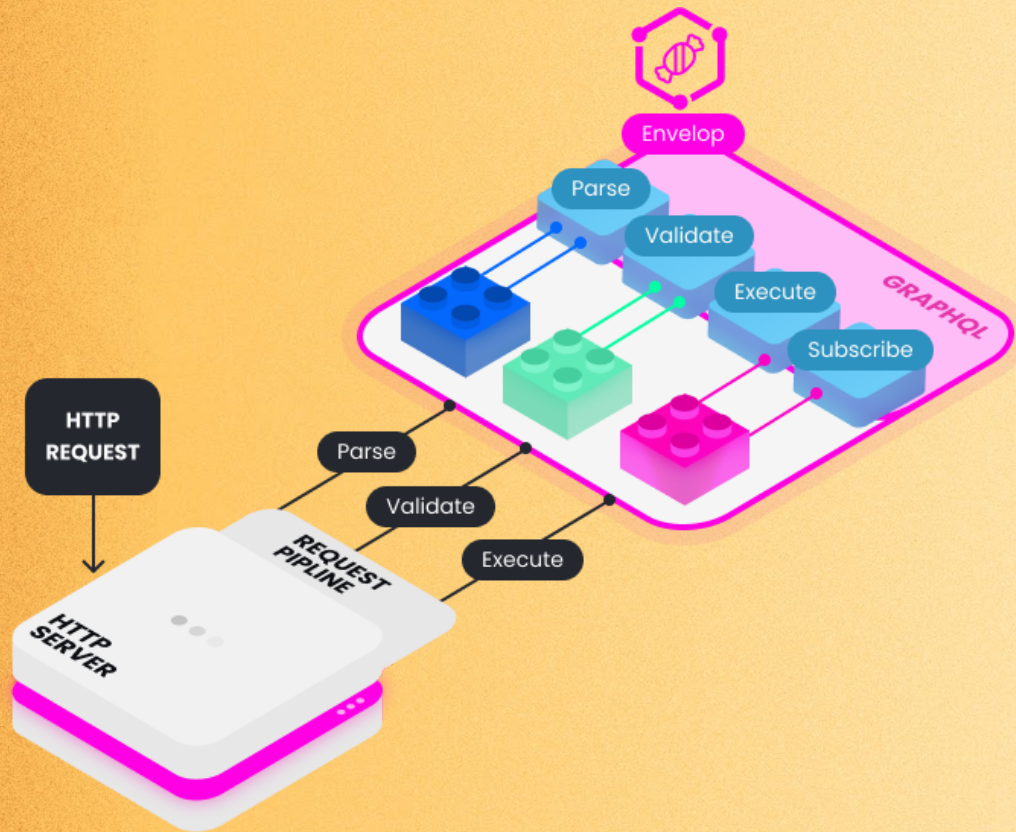


GraphQL Envelop

- Démarré début 2021
- Système de plugins pour GraphQL
- *Framework agnostic*



GraphQL Envelop









GraphQL Envelop - Plugins

Explore Plugins


tracing metrics core errors security utilities performance caching devtool authentication authorization schema subscription

Find plugins...

Trending

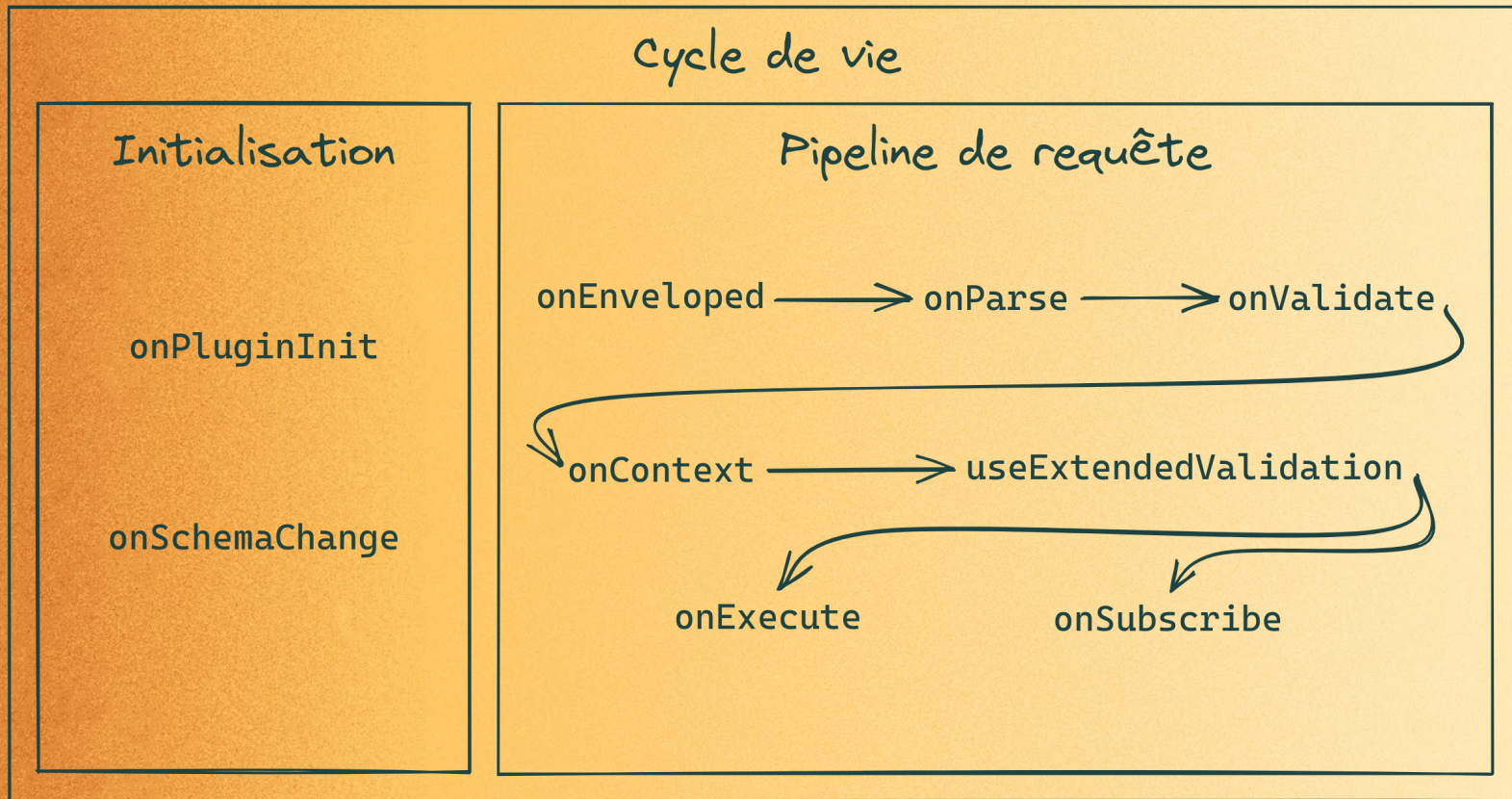
NAME	LAST UPDATE
 useSchema This is the core package for Envelop. You can find a complete documentation here: https://github.com/n1ru4l/envelop core schema	Oct 14, 2022 >
 useSchemaByContext This is the core package for Envelop. You can find a complete documentation here: https://github.com/n1ru4l/envelop core schema	Oct 14, 2022 >
 useErrorHandler This is the core package for Envelop. You can find a complete documentation here: https://github.com/n1ru4l/envelop core errors	Oct 14, 2022 >
 useMaskedErrors This is the core package for Envelop. You can find a complete documentation here: https://github.com/n1ru4l/envelop	Oct 14, 2022 >

Recently Updated

NAME	LAST UPDATE
 useStatsD This plugin tracks the complete execution flow, and reports metrics using StatsD (based on 'hot-shots'). metrics errors	Oct 14, 2022 >
 useValidationCache This plugin adds simple LRU caching to your 'validate', to improve performance by caching the validation result. performance caching	Oct 14, 2022 >
 useSentry This plugin collects errors and performance tracing for your execution flow, and reports it to [Sentry](https://sentry.io/). tracing metrics errors	Oct 14, 2022 >
 useResponseCache - Skip the execution phase and reduce server load by caching execution results in memory. - Customize cache entry time to	Oct 14, 2022 >



GraphQL Envelop - Plugins





GraphQL Envelop - Construire un serveur

```
import * as GraphQLJS from 'graphql'
import { envelop, useEngine, useSchema } from '@envelop/core'
import { useParserCache } from '@envelop/parser-cache'
import { useValidationCache } from '@envelop/validation-cache'
import { schema } from './schema'

export const getEnveloped = envelop({
  plugins: [
    useEngine(GraphQLJS),
    useSchema(schema),
    useParserCache(),
    useValidationCache()
  ]
})
```




GraphQL Envelop - Construire un serveur

```
import { createServer } from 'node:http'
import { GraphQLError } from 'graphql'
import { getEnveloped } from './envelop'

const httpServer = createServer(async (req, res) => {
  const initialContext = { req }

  const {
    parse,
    validate,
    contextFactory,
    execute,
    schema,
  } = getEnveloped(initialContext)

  // ...
})
```




GraphQL Envelop - Construire un serveur

```
const { query, variables } = JSON.parse(req.body)
const document = parse(query)
const validationErrors = validate(schema, document)

if (validationErrors.length > 0) {
  return res.end(JSON.stringify({ errors: validationErrors }))
}

const contextValue = await contextFactory()
const result = await execute({
  document,
  schema,
  variableValues: variables,
  contextValue,
})

res.end(JSON.stringify(result))
```




GraphQL Yoga

- Démarré début 2022
- Serveur GraphQL *fully featured*
- Extensible
- Facile à mettre en place



GraphQL Yoga

```
import { createServer } from 'node:http'
import { createYoga } from 'graphql-yoga'
import { useGraphQLJit } from '@envelop/graphql-jit'
import { schema } from './schema'

const yoga = createYoga({
  schema,
  plugins: [useGraphQLJit()],
})

const server = createServer(yoga)

server.listen(4000, () => {
  console.info('Server is running on http://localhost:4000/graphql')
})
```




**Je largue
Apollo Server !**

La migration (sur le papier)

```
- import { ApolloServer } from 'apollo-server'
+ import { createServer } from 'node:http'
+ import { createYoga } from 'graphql-yoga'
+ import { useApolloServerErrors } from '@envelop/apollo-server-errors'
  import { schema } from './schema'

- const server = new ApolloServer({
+ const yoga = createYoga({
    schema,
+   plugins: [useApolloServerErrors()],
  })

+ const server = createServer(yoga)

  server.listen(4000)
```


La vraie migration

- Directives customs ?
- Subscriptions ?
- Batching HTTP ?
- Resolvers génératrices ?!
- **Directives customs ?**
- Subscriptions ?
- Batching HTTP ?
- Resolvers génératrices ?!

Directives customs - Authentication

```
type Query @permission(permissions: "base") {  
  #...  
  users(filter: UserFilter): [User!]! @permission(permissions: "admin")  
  version: String! @public  
  #...  
}  
  
directive @permission(  
  permissions: [String!]!  
) on OBJECT | FIELD_DEFINITION  
  
directive @public on FIELD_DEFINITION
```


Directives customs - Authentication

Implémentées grâce à mapSchema (anciennement SchemaDirectiveVisitor) de GraphQL Tools

```
mapSchema(schema, {
  [MapperKind.OBJECT_TYPE]: (type) => {
    const directive = getDirective(schema, type, 'permission')?.[0]
    // ...
  },
  [MapperKind.FIELD]: (field, fieldName) => {
    const directive = getDirective(schema, field, 'permissions')?.[0]
    if (!directive) return undefined

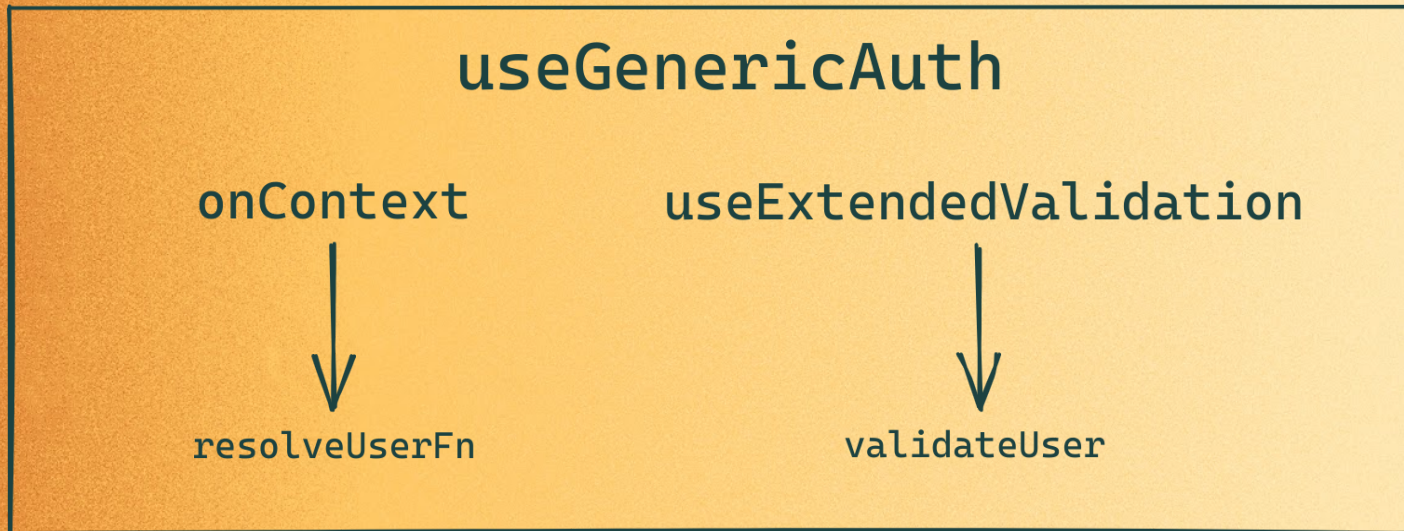
    const { permissions } = directive

    return { ...field, resolve: (parent, args, context, info) => {
      const user = context.user

      // Vérification des permissions...

      return field.resolve(parent, args, context, info)
    } }
  },
})
```

Plugin envelop useGenericAuth



- `resolveUserFn` : Fonction de résolution de l'utilisateur
- `validateUser` : Fonction de validation *custom*

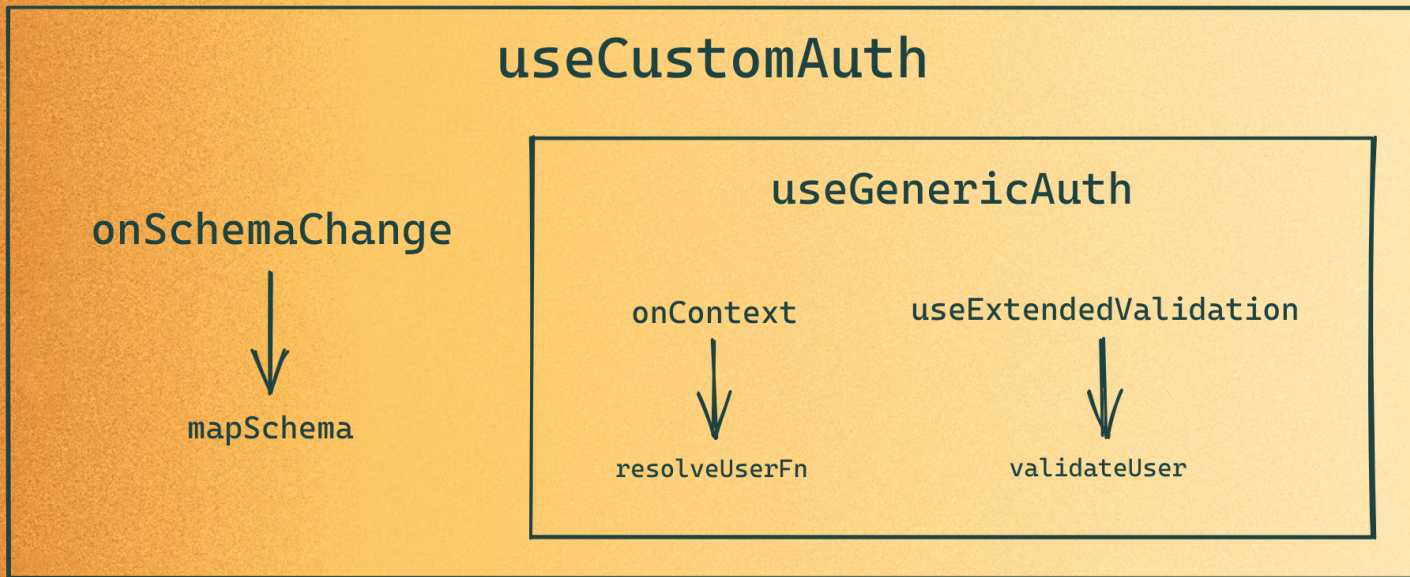
Plugin envelop useGenericAuth

```
export function addAuthExtension(schema) {
  return mapSchema(schema, {
    [MapperKind.OBJECT_TYPE]: (type) => {
      // ...
    },
    [MapperKind.OBJECT_FIELD]: (field, fieldName, typeName) => {
      const directive = getDirective(schema, field, 'permission')?.[0]
      const publicDirective = getDirective(schema, field, 'public')?.[0]
      if (!directive && !publicDirective) return undefined

      if (publicDirective) {
        const { permissions, ...extensions } = field.extensions
        return { ...field, extensions }
      }

      return {
        ...field,
        extensions: { ...field.extensions, permissions: directive.permissions },
      }
    },
  })
}
```

Plugin envelop useGenericAuth



La vraie migration

- Directives customs ?
- Subscriptions ?
- Batching HTTP ?
- Resolvers génératrices ?!

Directives customs - CRUD

```
type Query {  
  # ...  
  pathway(id: Int!): Pathway @crud(name: "pathways", operation: "get", args: ":id")  
  #...  
}  
  
type Pathway {  
  # ...  
  units: [Unit!]! @crud(name: "units", operation: "find", args: "{ pathwayId: ':parent.id' }")  
  #...  
}  
  
directive @crud(  
  name: String!  
  operation: String!  
  args: [String!]  
) on FIELD_DEFINITION
```


Directives customs - CRUD

```
export function addCrudResolvers(schema) {  
  return mapSchema(schema, {  
    [MapperKind.OBJECT_FIELD]: (field, fieldName) => {  
      const directive = getDirective(schema, field, 'crud')?.[0]  
      if (!directive) return undefined  
  
      const { name, operation, args: crudArgs } = directive  
      return { ...field, resolve: (parent, args, context) => {  
        // implémentation...  
      } }  
    },  
  })  
}
```

```
const yoga = createYoga({  
  schema: addCrudResolvers(addAuthExtension(schema)),  
  plugins: [  
    useCustomAuth(),  
  ],  
})
```

La vraie migration

- **Directives customs**
- Subscriptions ?
- Batching HTTP ?
- Resolvers génératrices ?!
- Directives customs
- **Subscriptions ?**
- Batching HTTP ?
- Resolvers génératrices ?!

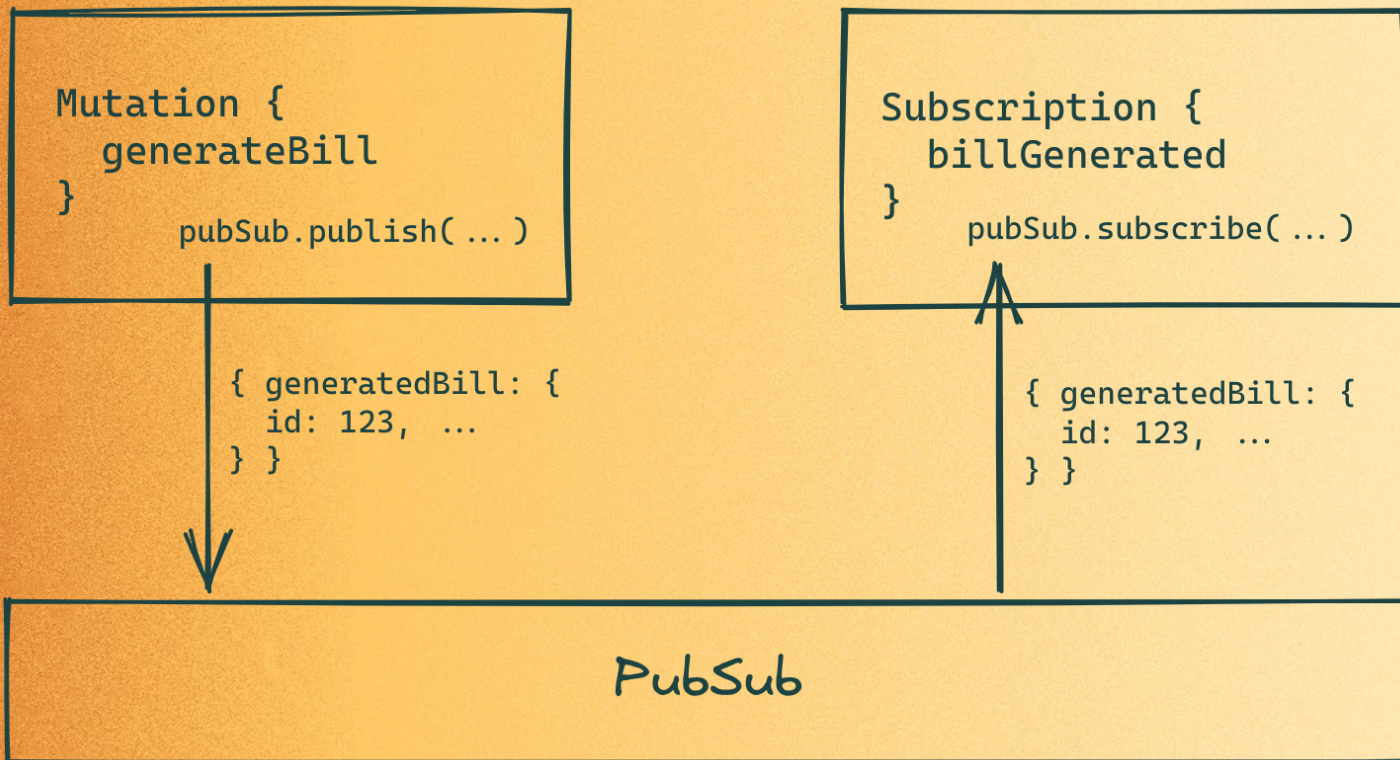
Subscriptions - Quel protocole ?

- WebSockets
- Server-Sent Events (SSE)

Subscriptions - Yoga

- Uniquement SSE par défaut (http/2 recommandé)
- Web Sockets possible avec graphql-ws

Subscriptions - PubSub



Subscriptions - PubSub

```
- import { PubSub } from 'apollo-server'  
+ import { createPubSub } from 'graphql-yoga'  
  
- export const pubSub = new PubSub()  
+ export const pubSub = createPubSub()
```

```
const Mutation = {  
  //...  
  async generateBill(parent, { id }, context) {  
    // Ça prend du temps...  
  
    pubSub.publish('GENERATED_BILL', { generatedBill: { id } })  
  },  
  //...  
}
```


Subscriptions - PubSub

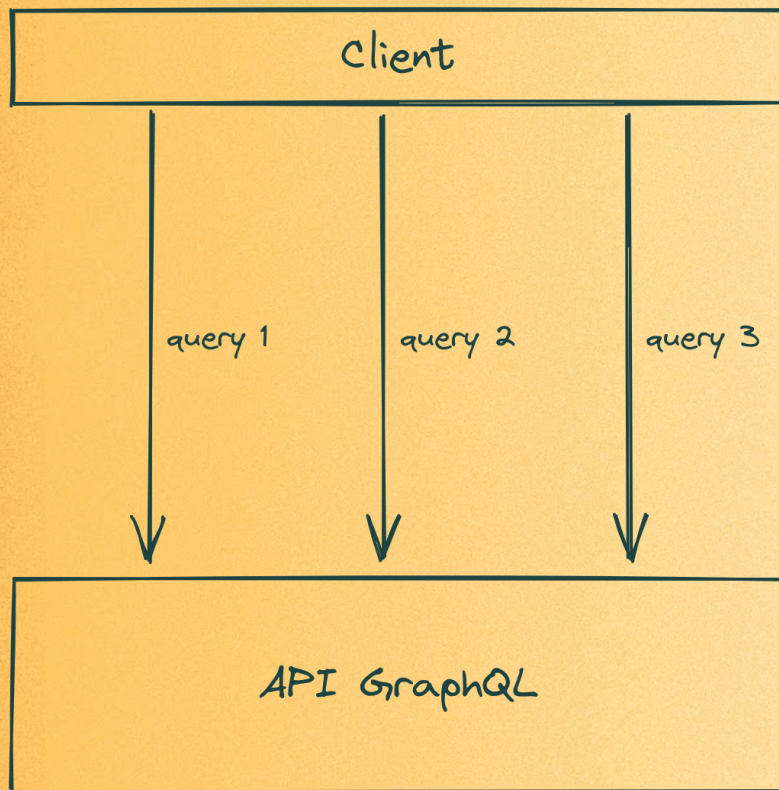
```
- import { withFilter } from 'apollo-server'
+ import { pipe, filter } from 'graphql-yoga'

const Subscription = {
  generatedBill: {
    -   subscribe: withFilter(
    -     () => pubSub.asyncIterator('GENERATED_BILL'),
    -     (payload, variables) => payload.generatedBill.id === variables.id,
    -   ),
    +   subscribe: pipe(
    +     pubSub.subscribe('GENERATED_BILL'),
    +     filter(
    +       (payload, variables) => payload.generatedBill.id === variables.id,
    +     ),
    +   ),
  },
}
```

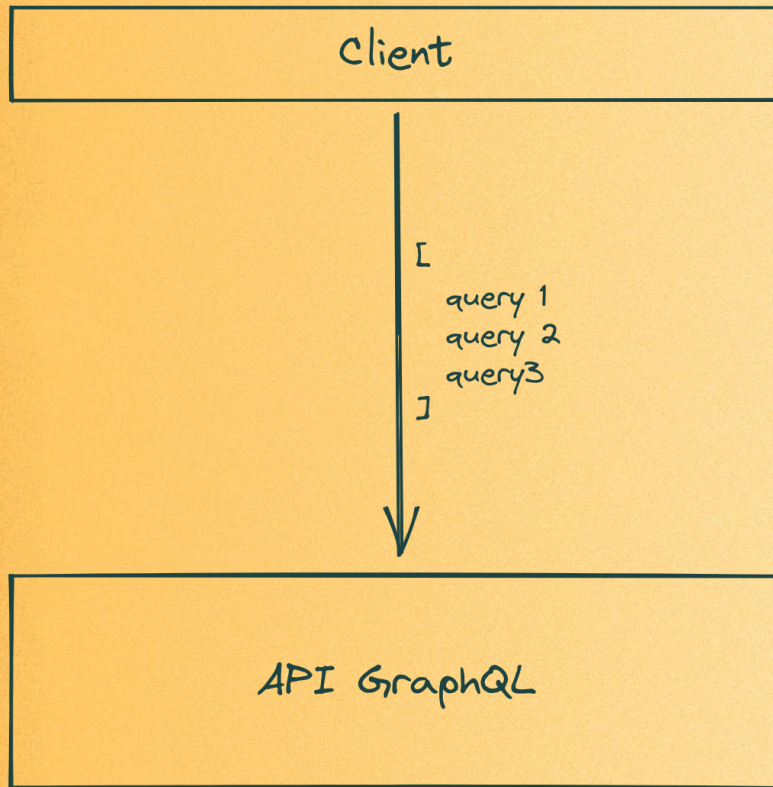
La vraie migration

- Directives customs
 - **Subscriptions**
 - Batching HTTP ?
 - Resolvers génératrices ?!
-
- Directives customs
 - Subscriptions
 - **Batching HTTP ?**
 - Resolvers génératrices ?!

Batching HTTP



Batching HTTP



Batching HTTP Yoga

- À partir de Yoga v3
- Compatible avec le Batching HTTP Link de Apollo Client

```
const yoga = createYoga({  
  schema: addCrudResolvers(addAuthExtension(schema)),  
  plugins: [  
    useCustomAuth(),  
  ],  
  batching: true,  
})
```

La vraie migration

- Directives customs
 - Subscriptions
 - **Batching HTTP**
 - Resolvers génératrices ?!
-
- Directives customs
 - Subscriptions
 - Batching HTTP
 - **Resolvers génératrices ?!**

Resolvers génératrices ?!

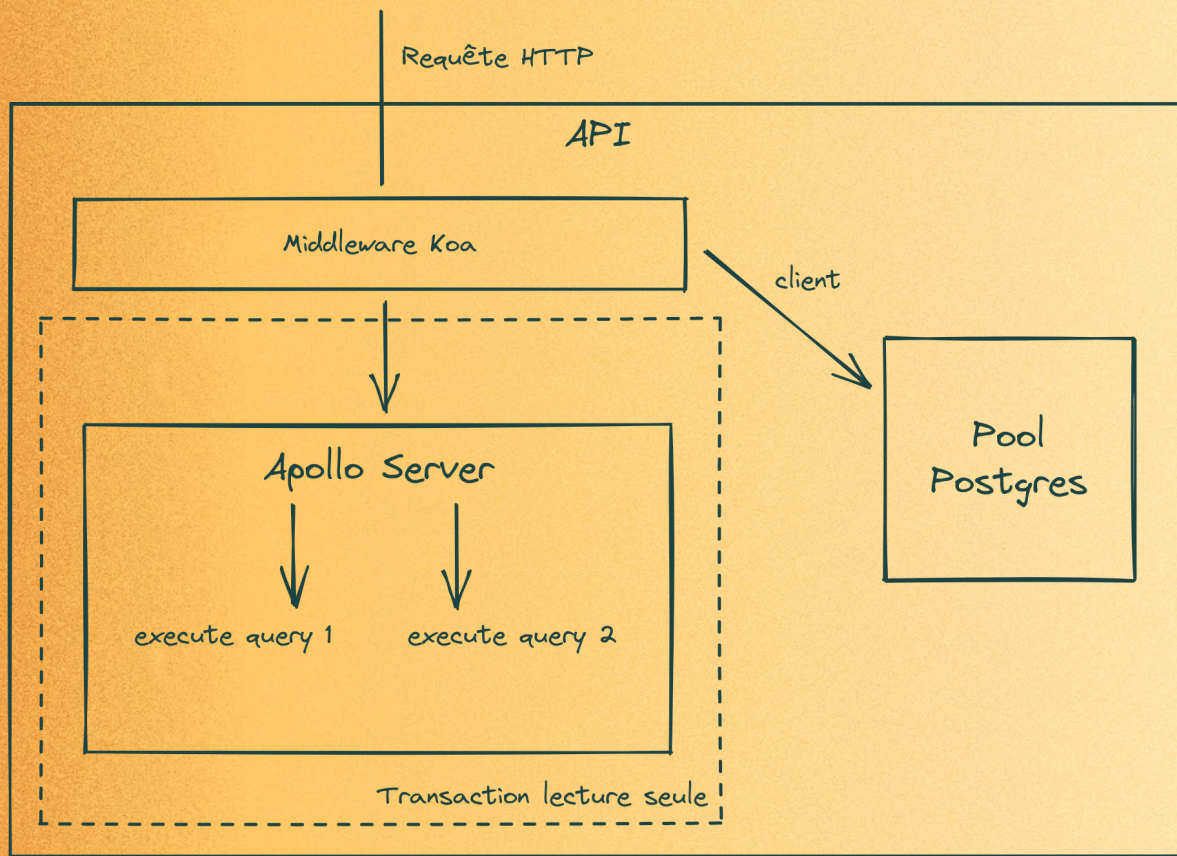
```
const Query = {  
  // ...  
  * student(_, { id }) {  
    return yield crud.students.get(id)  
  }  
  // ...  
}  
  
const Mutation = {  
  // ...  
  * updateStudent(_, { studentInput }) {  
    return yield crud.students.update(studentInput)  
  }  
  // ...  
}
```

Resolvers génératrices ?!

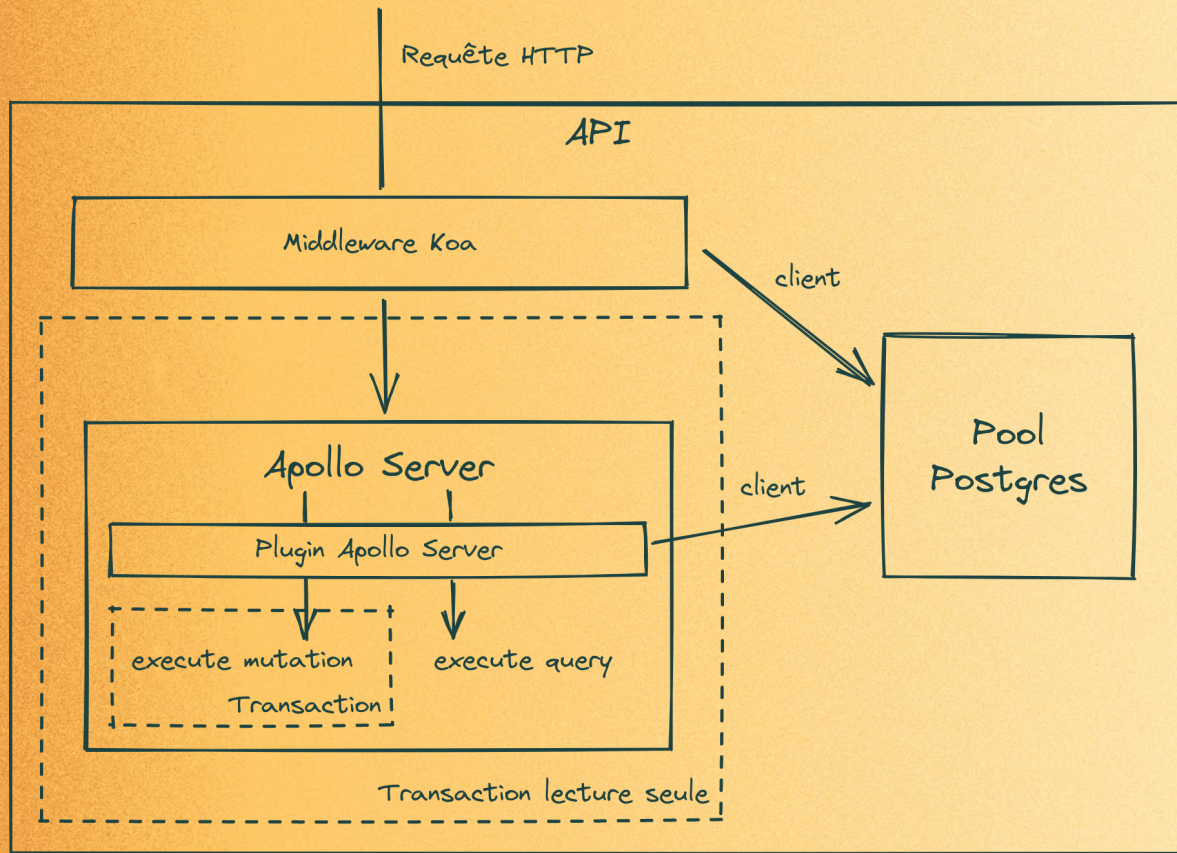
- Déduplication
- Batching
- Gestion du pool et des transactions Postgres

```
const yoga = createYoga({  
  schema: wrapGeneratorResolvers(addCrudResolvers(addAuthExtension(schema))),  
  plugins: [  
    useCustomAuth(),  
  ],  
  batching: true,  
})
```


Gestion du pool et des transactions Postgres



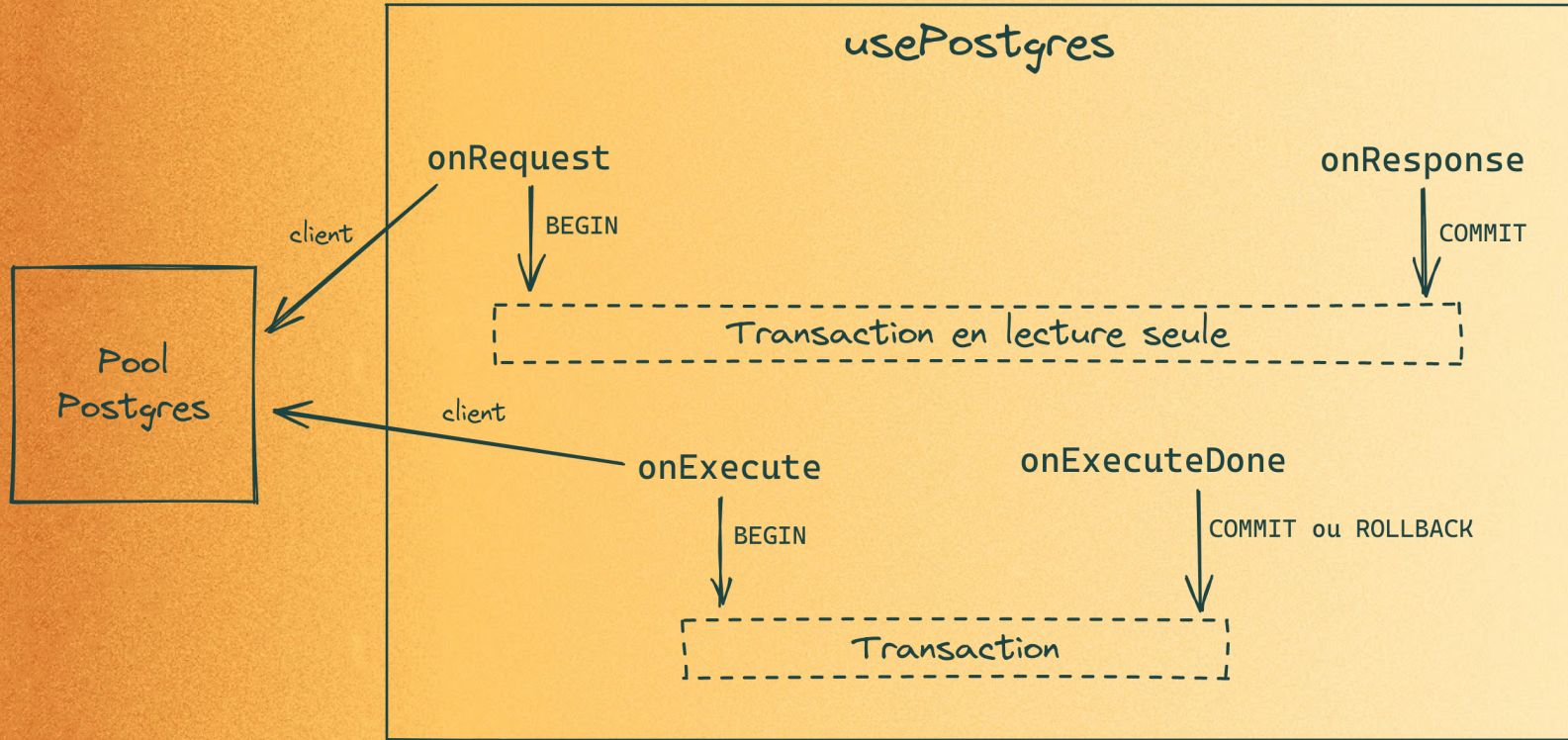
Gestion du pool et des transactions Postgres



Plugins Yoga

```
export type Plugin = EnvelopPlugin & {  
  /**  
   * Use this hook with your own risk. It is still experimental and may change in the future.  
   * @internal  
   */  
  onRequest?: OnRequestHook<TServerContext>  
  
  /**  
   * Use this hook with your own risk. It is still experimental and may change in the future.  
   * @internal  
   */  
  onResponse?: OnResponseHook<TServerContext>  
}
```

Gestion du pool et des transactions Postgres



Gestion du pool et des transactions Postgres

```
const yoga = createYoga({  
  schema: wrapGeneratorResolvers(addCrudResolvers(addAuthExtension(schema))),  
  plugins: [  
    useCustomAuth(),  
    usePostgres(),  
  ],  
  batching: true,  
})
```


Migration réussie ?

What's next ?

- Support de @defer et @stream
- Support officiel de Bun !

The Guild

The Ecosystem

Our advanced, modular solutions can be adopted gradually as individual open source libraries or as a complete unified API platform. Explore our suite of sustainable, open source API tools that covers everything you need to scale your API infrastructure:

[Learn more about The Guild →](#)



Hive



Yoga



Envelop



Inspector



Code Generator



Mesh



Tools



Modules



ESLint



Config



Scalars



Helix



Shield



Swift



CLI



SOFA



Stencil



Angular



WhatsApp



KitQL

Merci !

