

Federated GraphQL

Tomáš Horáček - 10/2022

What is GraphQL?

REST API 🤔

✓ HTTP(s)

✓ JSON

✓ GET, POST, PUT, DELETE

✓ Multiple URLs

GraphQL 🧬

✓ HTTP(s)

✓ JSON

⚠️ Query, Mutation - POST

⚠️ Single URL, customizable queries

REST API 🤪

<https://swapi.dev/api/people/1/>

```
{
  "name": "Luke Skywalker",
  "height": "172",
  "starships": [
    "https://swapi.dev/api/starships/12/",
    "https://swapi.dev/api/starships/22/"
  ],
  // ...
}
```

REST API 🤪

<https://swapi.dev/api/starships/12/>

```
{
  "name": "X-wing",
  "model": "T-65 X-wing",
  "manufacturer": "Incom Corporation",
  "pilots": [
    "https://swapi.dev/api/people/1/",
    "https://swapi.dev/api/people/9/",
    "https://swapi.dev/api/people/18/",
    "https://swapi.dev/api/people/19/"
  ],
  // ...
}
```



<https://graphql.org/swapi-graphql>

GraphQL

<https://graphql.org/swapi-graphql>

GraphQL query

```
query PersonQuery {  
  person(id: "cGVvcGx1OjE=") {  
    id  
    name  
    height  
  }  
}
```

GraphQL

<https://graphql.org/swapi-graphql>

GraphQL query

```
query PersonQuery {  
  person(id: "cGVvcGx1OjE=") {  
    id  
    name  
    height  
  }  
}
```

JSON response

```
{  
  "data": {  
    "person": {  
      "id": "cGVvcGx1OjE=",  
      "name": "Luke Skywalker",  
      "height": 172  
    }  
  }  
}
```


GraphQL

```
query PersonQuery {  
  person(id: "cGVvcGx1OjE=") {  
    id  
    name  
    height  
    starshipConnection {  
      starships {  
        name  
      }  
    }  
  }  
}
```

GraphQL

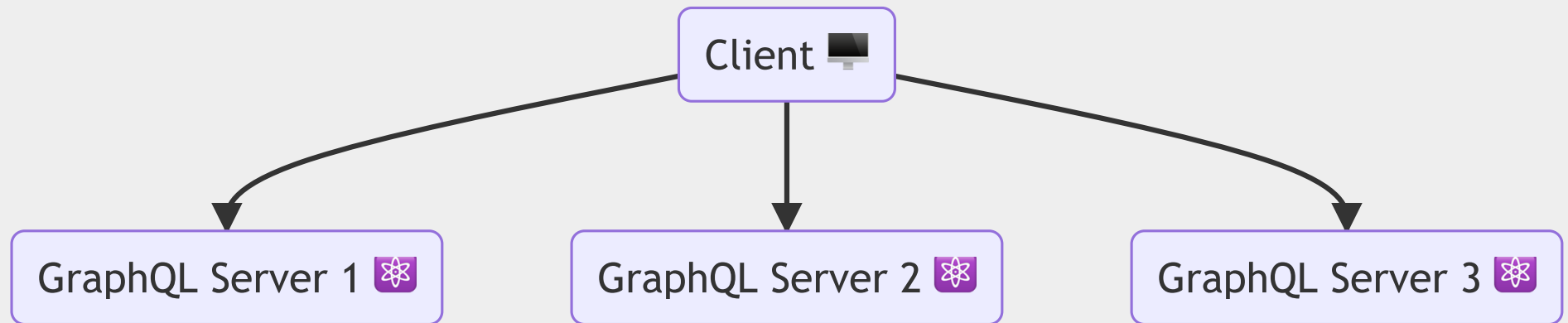
```
query PersonQuery {  
  person(id: "cGVvcGx1OjE=") {  
    id  
    name  
    height  
    starshipConnection {  
      starships {  
        name  
      }  
    }  
  }  
}
```

```
{  
  "data": {  
    "person": {  
      "id": "cGVvcGx1OjE=",  
      "name": "Luke Skywalker",  
      "height": 172,  
      "starshipConnection": {  
        "starships": [  
          { "name": "X-wing" },  
          { "name": "Imperial shuttle"  
        ]  
      }  
    }  
  }  
}
```

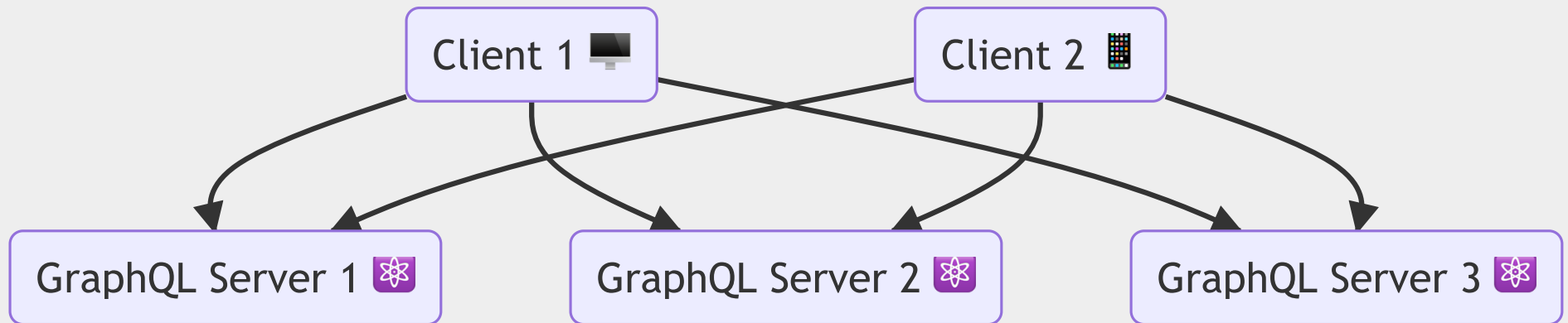
What is Federated GraphQL?



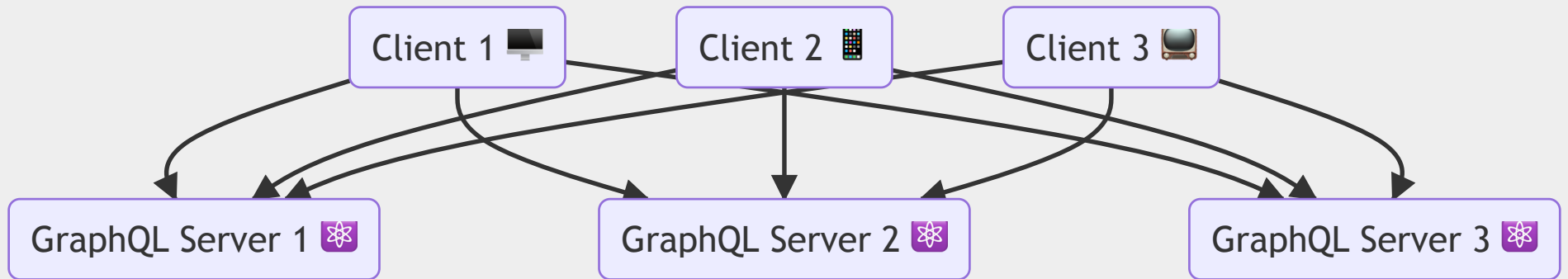
multiple GraphQL servers



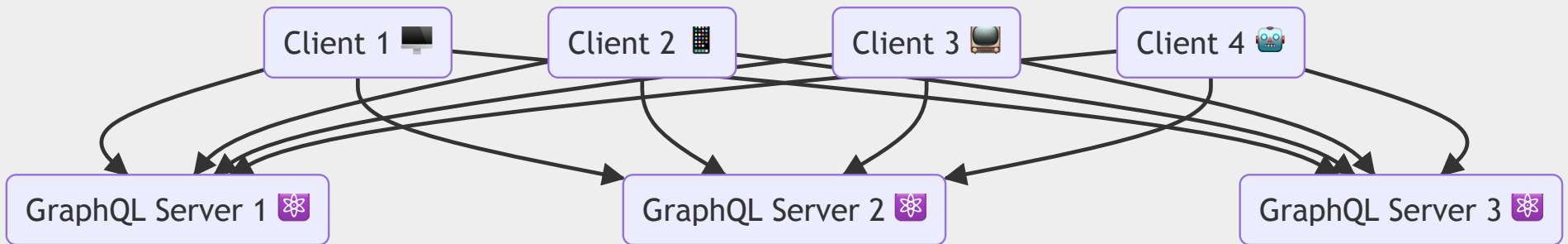
multiple GraphQL servers



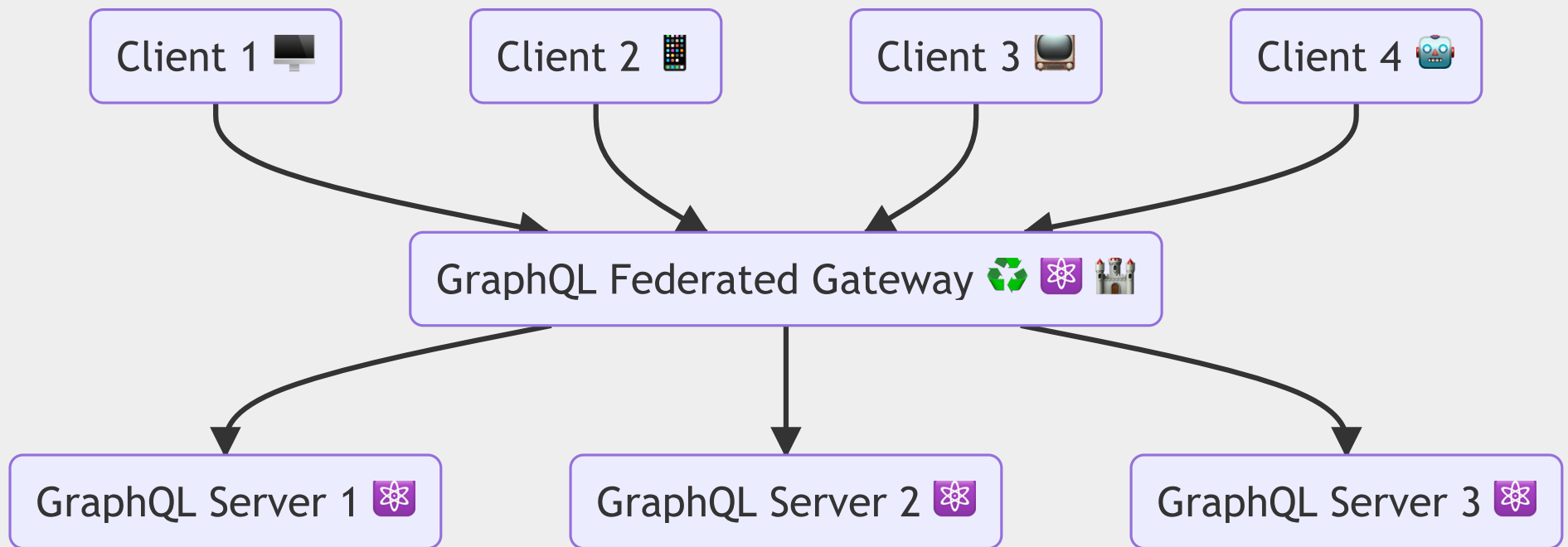
multiple GraphQL servers



multiple GraphQL servers




Federated GraphQL





Implementations



- Apollo Federation (v2)
- GraphQL Tools - Stitching (v7+) 
- Hasura GraphQL Joins
- ...

Implementations



- Apollo Federation (v2)
- GraphQL Tools - Stitching (v7+) 
- Hasura GraphQL Joins
- ...
-  Proposal: GraphQL Composite Schemas Working Group (23th May 2022)

GraphQL Tools - Stitching v8

GraphQL Tools - Stitching v8

- #1 combine
schemas

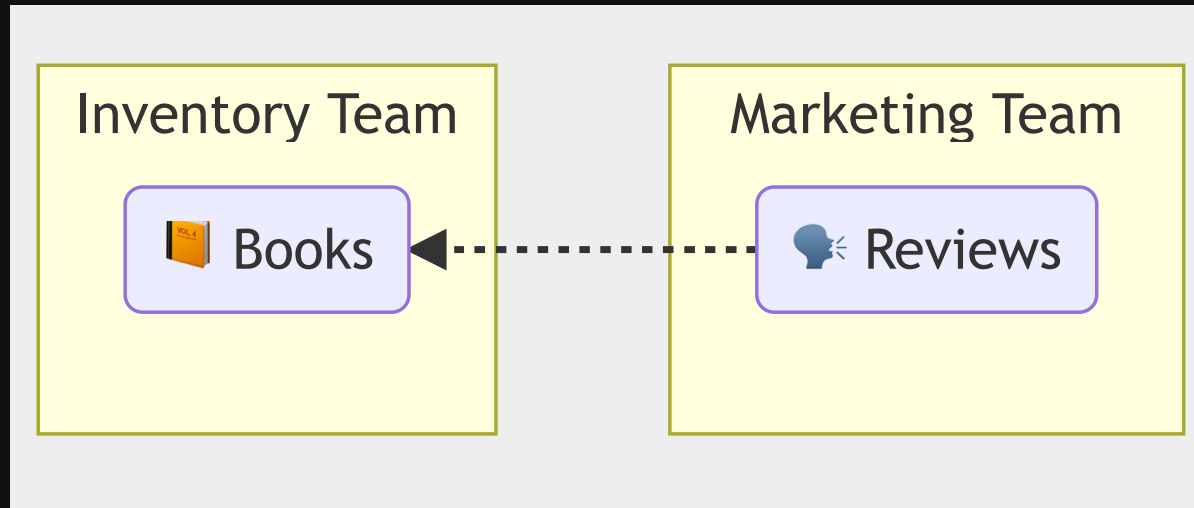
GraphQL Tools - Stitching v8

- #1 combine schemas
- #2 extend types

#1 Combine Schemas - Stitching

The Problem

books e-shop





Inventory GraphQL

Query

```
query AllBooks {  
  books {  
    id  
    title  
  }  
}
```




Inventory GraphQL

Query

```
query AllBooks {  
  books {  
    id  
    title  
  }  
}
```

Response

```
{  
  "data": {  
    "books": [  
      { "id": "1", "title": "The Fellowship of the Ring" },  
      { "id": "2", "title": "The Two Towers" },  
      { "id": "3", "title": "The Return of the King" }  
    ]  
  }  
}
```

Marketing GraphQL

Query

```
1 query ReviewsForBook {  
2   reviews(  
3     contentType: "book",  
4     objectId: "1"  
5   ) {  
6     rating  
7     text  
8   }  
9 }
```

Marketing GraphQL

Query

```
1 query ReviewsForBook {  
2   reviews(  
3     contentType: "book",  
4     objectId: "1"  
5   ) {  
6     rating  
7     text  
8   }  
9 }
```

Marketing GraphQL

Query

```
1 query ReviewsForBook {  
2   reviews(  
3     contentType: "book",  
4     objectId: "1"  
5   ) {  
6     rating  
7     text  
8   }  
9 }
```

Marketing GraphQL

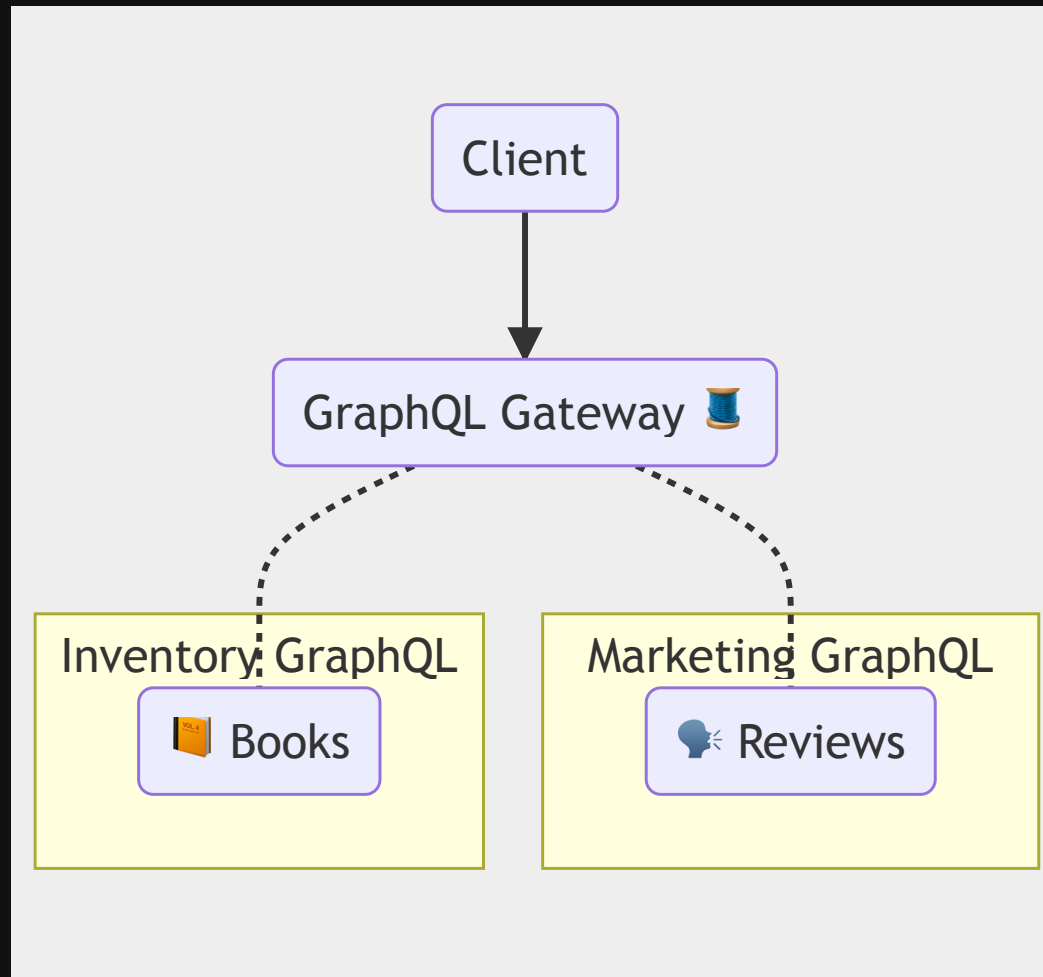
Query

```
1 query ReviewsForBook {
2   reviews(
3     contentType: "book",
4     objectId: "1"
5   ) {
6     rating
7     text
8   }
9 }
```

Response

```
{
  "data": {
    "reviews": [
      {
        "rating": 5,
        "text": "Fellowship is the bes
      },
      {
        "rating": 5,
        "text": "One does not simply..
      }
    ]
  }
}
```

#1 Combine Schemas - Stitching




#1 Combine Schemas - Stitching

#1 Combine Schemas - Stitching

 Inventory Schema

```
type Book {  
  id: String!  
  title: String!  
}  
  
type Query {  
  books: [Book]  
}
```


#1 Combine Schemas - Stitching


 Inventory Schema

```
type Book {  
  id: String!  
  title: String!  
}  
  
type Query {  
  books: [Book]  
}
```

 Marketing Schema

```
1 type Review {  
2   id: String!  
3  
4   contentType: String!  
5   objectId: String!  
6  
7   rating: Int  
8   text: String!  
9 }  
10  
11 type Query {  
12   reviews(  
13     contentType: String!,  
14     objectId: String!  
15   ): [Review]  
16 }
```

#1 Combine Schemas - Stitching


 Inventory Schema

```
type Book {  
  id: String!  
  title: String!  
}  
  
type Query {  
  books: [Book]  
}
```

 Marketing Schema

```
1 type Review {  
2   id: String!  
3  
4   contentType: String!  
5   objectId: String!  
6  
7   rating: Int  
8   text: String!  
9 }  
10  
11 type Query {  
12   reviews(  
13     contentType: String!,  
14     objectId: String!  
15   ): [Review]  
16 }
```

#1 Combine Schemas - Stitching


 Inventory Schema

```
type Book {  
  id: String!  
  title: String!  
}  
  
type Query {  
  books: [Book]  
}
```

 Marketing Schema


```
1 type Review {  
2   id: String!  
3  
4   contentType: String!  
5   objectId: String!  
6  
7   rating: Int  
8   text: String!  
9 }  
10  
11 type Query {  
12   reviews(  
13     contentType: String!,  
14     objectId: String!  
15   ): [Review]  
16 }
```

#1 Combine Schemas - Stitching

GraphQL Gateway 


```
1 import { createServer } from "@graphql-yoga/node";
2 import { stitchSchemas } from '@graphql-tools/stitch';
3
4 import { getSubschemas } from './subschemas'
5
6 const {
7   inventorySchema,
8   marketingSchema,
9 } = await getSubschemas();
10
11 export const gatewaySchema = stitchSchemas({
12   subschemas: [inventorySchema, marketingSchema]
13 });
14
15 const server = createServer({
16   schema: gatewaySchema
17 });
```

#1 Combine Schemas - Stitching

GraphQL Gateway 


```
1 import { createServer } from "@graphql-yoga/node";
2 import { stitchSchemas } from '@graphql-tools/stitch';
3
4 import { getSubschemas } from './subschemas'
5
6 const {
7   inventorySchema,
8   marketingSchema,
9 } = await getSubschemas();
10
11 export const gatewaySchema = stitchSchemas({
12   subschemas: [inventorySchema, marketingSchema]
13 });
14
15 const server = createServer({
16   schema: gatewaySchema
17 });
```

#1 Combine Schemas - Stitching

GraphQL Gateway 

```
1 import { createServer } from "@graphql-yoga/node";
2 import { stitchSchemas } from '@graphql-tools/stitch';
3
4 import { getSubschemas } from './subschemas'
5
6 const {
7   inventorySchema,
8   marketingSchema,
9 } = await getSubschemas();
10
11 export const gatewaySchema = stitchSchemas({
12   subschemas: [inventorySchema, marketingSchema]
13 });
14
15 const server = createServer({
16   schema: gatewaySchema
17 });
```

#1 Combine Schemas - Stitching

GraphQL Gateway 

```
1 import { createServer } from "@graphql-yoga/node";
2 import { stitchSchemas } from '@graphql-tools/stitch';
3
4 import { getSubschemas } from './subschemas'
5
6 const {
7   inventorySchema,
8   marketingSchema,
9 } = await getSubschemas();
10
11 export const gatewaySchema = stitchSchemas({
12   subschemas: [inventorySchema, marketingSchema]
13 });
14
15 const server = createServer({
16   schema: gatewaySchema
17 });
```

#1 Combine Schemas - Stitching

GraphQL Gateway 

```
1 import { createServer } from "@graphql-yoga/node";
2 import { stitchSchemas } from '@graphql-tools/stitch';
3
4 import { getSubschemas } from './subschemas'
5
6 const {
7   inventorySchema,
8   marketingSchema,
9 } = await getSubschemas();
10
11 export const gatewaySchema = stitchSchemas({
12   subschemas: [inventorySchema, marketingSchema]
13 });
14
15 const server = createServer({
16   schema: gatewaySchema
17 });
```


Gateway Schema 🏰

```
type Book {  
  id: String!  
  title: String!  
}  
  
type Review {  
  id: String!  
  
  contentType: String!  
  objectId: String!  
  
  rating: Int  
  text: String!  
}  
  
type Query {  
  books: [Book]  
  reviews(contentType: String!, objectId: String!): [Review]  
}
```

#2 Extend Types - Stitching

#2 Extend Types - Stitching

Inventory 

```
type Book {  
  id: String!  
  title: String!  
}
```

#2 Extend Types - Stitching

Inventory 

```
type Book {  
  id: String!  
  title: String!  
}
```

Gateway 

```
extend type Book {  
  reviews: [Review]  
}
```

#2 Extend Types - Stitching

GraphQL Gateway

```
1 import { delegateToSchema } from "@graphql-tools/delegate";
2
3 export const gatewaySchema = stitchSchemas({
4   subschemas: [inventorySchema, marketingSchema],
5   typeDefs: `
6     extend type Book {
7       reviews: [Review]
8     }
9   `,
10  resolvers: {
11    Book: {
12      reviews: {
13        selectionSet: `{ id }`, // Book.id
14        resolve(parent, args, context, info) {
15          ...
16        }
17      }
18    }
19  }
20 });
```

#2 Extend Types - Stitching

GraphQL Gateway

```
1 import { delegateToSchema } from "@graphql-tools/delegate";
2
3 export const gatewaySchema = stitchSchemas({
4   subschemas: [inventorySchema, marketingSchema],
5   typeDefs: `
6     extend type Book {
7       reviews: [Review]
8     }
9   `,
10  resolvers: {
11    Book: {
12      reviews: {
13        selectionSet: `{ id }`, // Book.id
14        resolve(parent, args, context, info) {
15          ...
16        }
17      }
18    }
19  }
20 });
```

#2 Extend Types - Stitching

GraphQL Gateway

```
1 import { delegateToSchema } from "@graphql-tools/delegate";
2
3 export const gatewaySchema = stitchSchemas({
4   subschemas: [inventorySchema, marketingSchema],
5   typeDefs: `
6     extend type Book {
7       reviews: [Review]
8     }
9   `,
10  resolvers: {
11    Book: {
12      reviews: {
13        selectionSet: `{ id }`, // Book.id
14        resolve(parent, args, context, info) {
15          ...
16        }
17      }
18    }
19  }
20 });
```

#2 Extend Types - Stitching

GraphQL Gateway

```
1 import { delegateToSchema } from "@graphql-tools/delegate";
2
3 export const gatewaySchema = stitchSchemas({
4   subschemas: [inventorySchema, marketingSchema],
5   typeDefs: `
6     extend type Book {
7       reviews: [Review]
8     }
9   `,
10  resolvers: {
11    Book: {
12      reviews: {
13        selectionSet: `{ id }`, // Book.id
14        resolve(parent, args, context, info) {
15          ...
16        }
17      }
18    }
19  }
20 });
```


#2 Extend Types - Stitching

GraphQL Gateway

```
1 import { delegateToSchema } from "@graphql-tools/delegate";
2
3 export const gatewaySchema = stitchSchemas({
4   subschemas: [inventorySchema, marketingSchema],
5   typeDefs: `
6     extend type Book {
7       reviews: [Review]
8     }
9   `,
10  resolvers: {
11    Book: {
12      reviews: {
13        selectionSet: `{ id }`, // Book.id
14        resolve(parent, args, context, info) {
15          ...
16        }
17      }
18    }
19  }
20 });
```

#2 Extend Types - Stitching

GraphQL Gateway 

```
1 selectionSet: `{ id }`, // Book.id
2 resolve(parent, args, context, info) {
3   const bookId = parent.id;
4
5   return delegateToSchema({
6     schema: marketingSchema,
7     operation: 'query',
8     fieldName: 'reviews',
9     args: {
10       contentType: 'book',
11       objectId: bookId,
12     },
13     context,
14     info,
15   });
16 }
```

Generated Query

```
query {
  reviews(
    contentType: "book",
    objectId: $bookId
  ) {
    ...
  }
}
```

#2 Extend Types - Stitching

GraphQL Gateway 

```
1 selectionSet: `{ id }`, // Book.id
2 resolve(parent, args, context, info) {
3   const bookId = parent.id;
4
5   return delegateToSchema({
6     schema: marketingSchema,
7     operation: 'query',
8     fieldName: 'reviews',
9     args: {
10       contentType: 'book',
11       objectId: bookId,
12     },
13     context,
14     info,
15   });
16 }
```

Generated Query

```
query {
  reviews(
    contentType: "book",
    objectId: $bookId
  ) {
    ...
  }
}
```

#2 Extend Types - Stitching

GraphQL Gateway 

```
1 selectionSet: `{ id }`, // Book.id
2 resolve(parent, args, context, info) {
3   const bookId = parent.id;
4
5   return delegateToSchema({
6     schema: marketingSchema,
7     operation: 'query',
8     fieldName: 'reviews',
9     args: {
10       contentType: 'book',
11       objectId: bookId,
12     },
13     context,
14     info,
15   });
16 }
```

Generated Query

```
query {
  reviews(
    contentType: "book",
    objectId: $bookId
  ) {
    ...
  }
}
```

#2 Extend Types - Stitching

GraphQL Gateway 

```
1 selectionSet: `{ id }`, // Book.id
2 resolve(parent, args, context, info) {
3   const bookId = parent.id;
4
5   return delegateToSchema({
6     schema: marketingSchema,
7     operation: 'query',
8     fieldName: 'reviews',
9     args: {
10       contentType: 'book',
11       objectId: bookId,
12     },
13     context,
14     info,
15   });
16 }
```

Generated Query

```
query {
  reviews(
    contentType: "book",
    objectId: $bookId
  ) {
    ...
  }
}
```

#2 Extend Types - Stitching

GraphQL Gateway 

```
1 selectionSet: `{ id }`, // Book.id
2 resolve(parent, args, context, info) {
3   const bookId = parent.id;
4
5   return delegateToSchema({
6     schema: marketingSchema,
7     operation: 'query',
8     fieldName: 'reviews',
9     args: {
10       contentType: 'book',
11       objectId: bookId,
12     },
13     context,
14     info,
15   });
16 }
```

Generated Query

```
query {
  reviews(
    contentType: "book",
    objectId: $bookId
  ) {
    ...
  }
}
```

#2 Extend Types - Stitching

GraphQL Gateway 

```
1 selectionSet: `{ id }`, // Book.id
2 resolve(parent, args, context, info) {
3   const bookId = parent.id;
4
5   return delegateToSchema({
6     schema: marketingSchema,
7     operation: 'query',
8     fieldName: 'reviews',
9     args: {
10       contentType: 'book',
11       objectId: bookId,
12     },
13     context,
14     info,
15   });
16 }
```

Generated Query

```
query {
  reviews(
    contentType: "book",
    objectId: $bookId
  ) {
    ...
  }
}
```

#2 Extend Types - Stitching

GraphQL Gateway 

```
1 selectionSet: `{ id }`, // Book.id
2 resolve(parent, args, context, info) {
3   const bookId = parent.id;
4
5   return delegateToSchema({
6     schema: marketingSchema,
7     operation: 'query',
8     fieldName: 'reviews',
9     args: {
10       contentType: 'book',
11       objectId: bookId,
12     },
13     context,
14     info,
15   });
16 }
```

Generated Query

```
query {
  reviews(
    contentType: "book",
    objectId: $bookId
  ) {
    ...
  }
}
```


#2 Extend Types - Stitching

GraphQL Gateway 

```
1 selectionSet: `{ id }`, // Book.id
2 resolve(parent, args, context, info) {
3   const bookId = parent.id;
4
5   return delegateToSchema({
6     schema: marketingSchema,
7     operation: 'query',
8     fieldName: 'reviews',
9     args: {
10       contentType: 'book',
11       objectId: bookId,
12     },
13     context,
14     info,
15   });
16 }
```

Generated Query

```
query {
  reviews(
    contentType: "book",
    objectId: $bookId
  ) {
    ...
  }
}
```

#2 Extend Types - Stitching

GraphQL Gateway 

```
1 selectionSet: `{ id }`, // Book.id
2 resolve(parent, args, context, info) {
3   const bookId = parent.id;
4
5   return delegateToSchema({
6     schema: marketingSchema,
7     operation: 'query',
8     fieldName: 'reviews',
9     args: {
10       contentType: 'book',
11       objectId: bookId,
12     },
13     context,
14     info,
15   });
16 }
```

Generated Query

```
query {
  reviews(
    contentType: "book",
    objectId: $bookId
  ) {
    ...
  }
}
```

#2 Extend Types - Stitching

GraphQL Gateway 

```
1 selectionSet: `{ id }`, // Book.id
2 resolve(parent, args, context, info) {
3   const bookId = parent.id;
4
5   return delegateToSchema({
6     schema: marketingSchema,
7     operation: 'query',
8     fieldName: 'reviews',
9     args: {
10       contentType: 'book',
11       objectId: bookId,
12     },
13     context,
14     info,
15   });
16 }
```

Generated Query

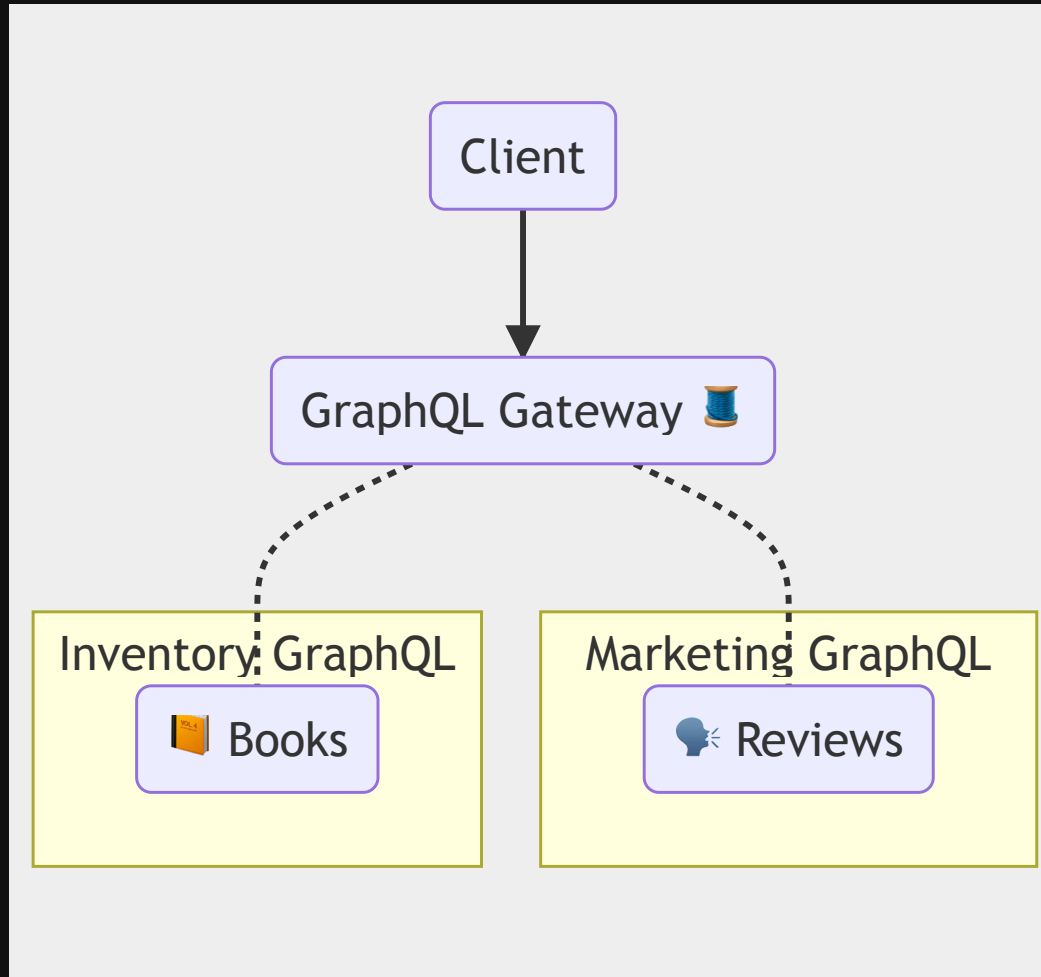
```
query {
  reviews(
    contentType: "book",
    objectId: $bookId
  ) {
    ...
  }
}
```

#2 Extend Types - Stitching

GraphQL Gateway 

```
type Book {  
  id: String!  
  title: String!  
  reviews: [Review]  
}
```

#2 Extend Types - Stitching



#2 Extend Types - Stitching

GraphQL Gateway 

```
query BooksWithReviews {  
  books {  
    title  
  
    reviews {  
      rating  
      text  
    }  
  }  
}
```

#2 Extend Types - Stitching

GraphQL Gateway

```
query BooksWithReviews {  
  books {  
    title  
  
    reviews {  
      rating  
      text  
    }  
  }  
}
```

```
{  
  "data": {  
    "books": [  
      {  
        "title": "The Fellowship of the Ring",  
        "reviews": [  
          { "rating": 5, "text": "Fellowship is the best!"  
          { "rating": 5, "text": "One does not simply..." }  
        ]  
      },  
      {  
        "title": "The Two Towers",  
        "reviews": [  
          { "rating": 4, "text": "Two books, one more to go  
        ]  
      },  
      {  
        "title": "The Return of the King",  
        "reviews": [  
          { "rating": 5, "text": "Long live the king!" }  
        ]  
      }  
    ]  
  }  
}
```

github.com/cngroupdk/nalejvarna-2022-federated-graphql

github.com/cngroupdk/nalejvarna-2022-federated-graphql



Questions?

Thank you for your attention! 🙏