APOLLO

Empowering GraphQL at scale with Apollo open source tooling

Andy Roberts
Senior Manager - EMEA Customer Success

Apollo’s open source journey

And how it’s tooling and products empower organisations

- MeteorJS
- The data abstraction problem and GraphQL
- The birth of Apollo Client and Server
- Apollo Federation and interconnected graphs
- Commercial and open source working hand in hand

- What is GraphQL?
- Introduction to “The Graph”
- Overview of Federation

In the beginning…

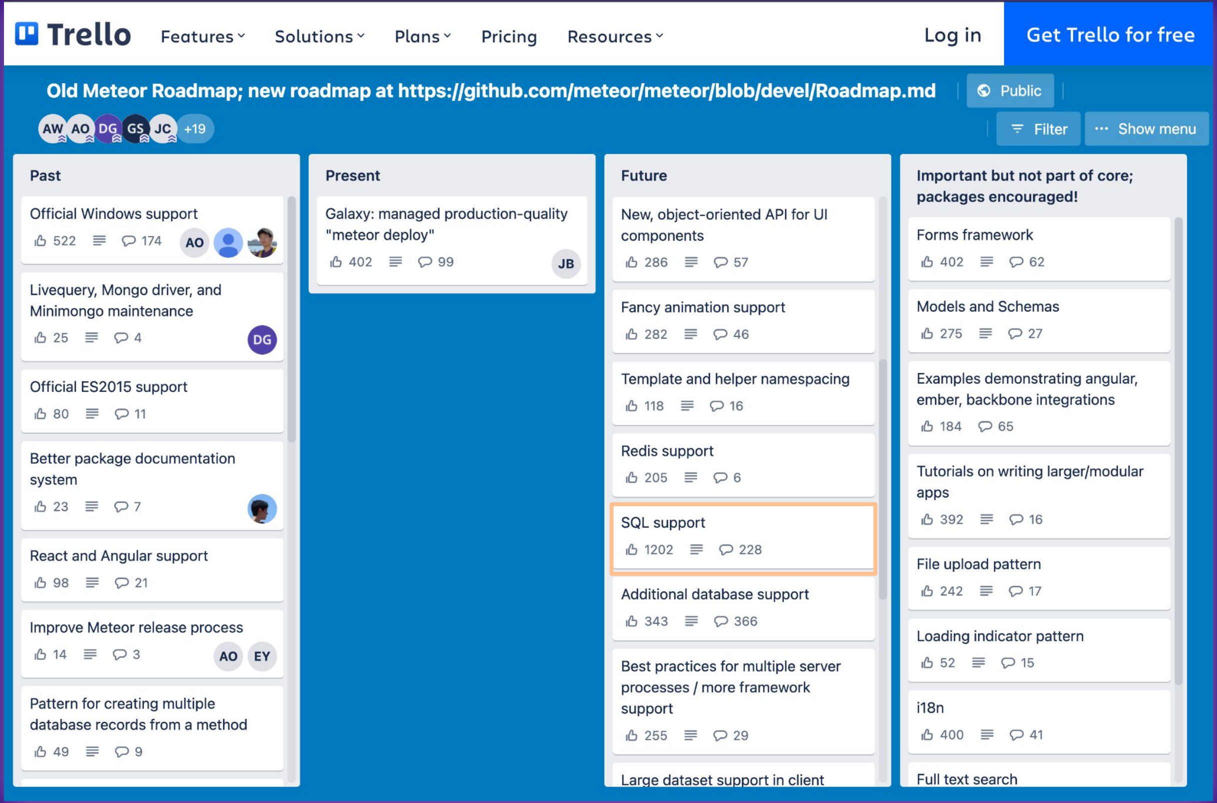
The start of our journey

The birth of Meteor

- Preview release in December 2011
- Fullstack JavaScript framework for building isomorphic apps
- Opinionated way to build fullstack apps
- Open source and community were at the heart of the project
- Meteor Galaxy - the best way to run your meteor apps
- But there was a problem…
- MongoDB

The start of our journey

The ticket that changed everything



Trello · Features · Solutions · Plans · Pricing · Resources · Log in · Get Trello for free

Old Meteor Roadmap; new roadmap at https://github.com/meteor/meteor/blob/devel/Roadmap.md

Public · Filter · Show menu

AW AO DG GS JC +19

Past

- Official Windows support — 522 · 174 — AO
- Livequery, Mongo driver, and Minimongo maintenance — 25 · 4 — DG
- Official ES2015 support — 80 · 11
- Better package documentation system — 23 · 7
- React and Angular support — 98 · 21
- Improve Meteor release process — 14 · 3 — AO EY
- Pattern for creating multiple database records from a method — 49 · 9

Present

- Galaxy: managed production-quality "meteor deploy" — 402 · 99 — JB

Future

- New, object-oriented API for UI components — 286 · 57
- Fancy animation support — 282 · 46
- Template and helper namespacing — 118 · 16
- Redis support — 205 · 6
- SQL support — 1202 · 228
- Additional database support — 343 · 366
- Best practices for multiple server processes / more framework support — 255 · 29
- Large dataset support in client

Important but not part of core; packages encouraged!

- Forms framework — 402 · 62
- Models and Schemas — 275 · 27
- Examples demonstrating angular, ember, backbone integrations — 184 · 65
- Tutorials on writing larger/modular apps — 392 · 16
- File upload pattern — 242 · 17
- Loading indicator pattern — 52 · 15
- i18n — 400 · 41
- Full text search

The start of our journey

A new star is born

- Desire to support SQL to kickstart wider adoption of Meteor
- Plan to create a DB agnostic layer that clients could talk to
- Meteor embraced the new open spec from Facebook: GraphQL
- Intention was to create tooling that would make Meteor incrementally adoptable
- Apollo Client and Server were born
- Initial excitement snowballed and both packages became runaway successes
- Before long the Apollo packages were more popular than Meteor itself

What is GraphQL?

What is GraphQL?

A query language for your APIs

- Invented at Facebook in 2012
- As a solution to too many service calls in their mobile app
- GraphQL replaced them all with a single request
- GraphQL is a query language for your API
- That helps you to build evolvable and client-focussed schemas

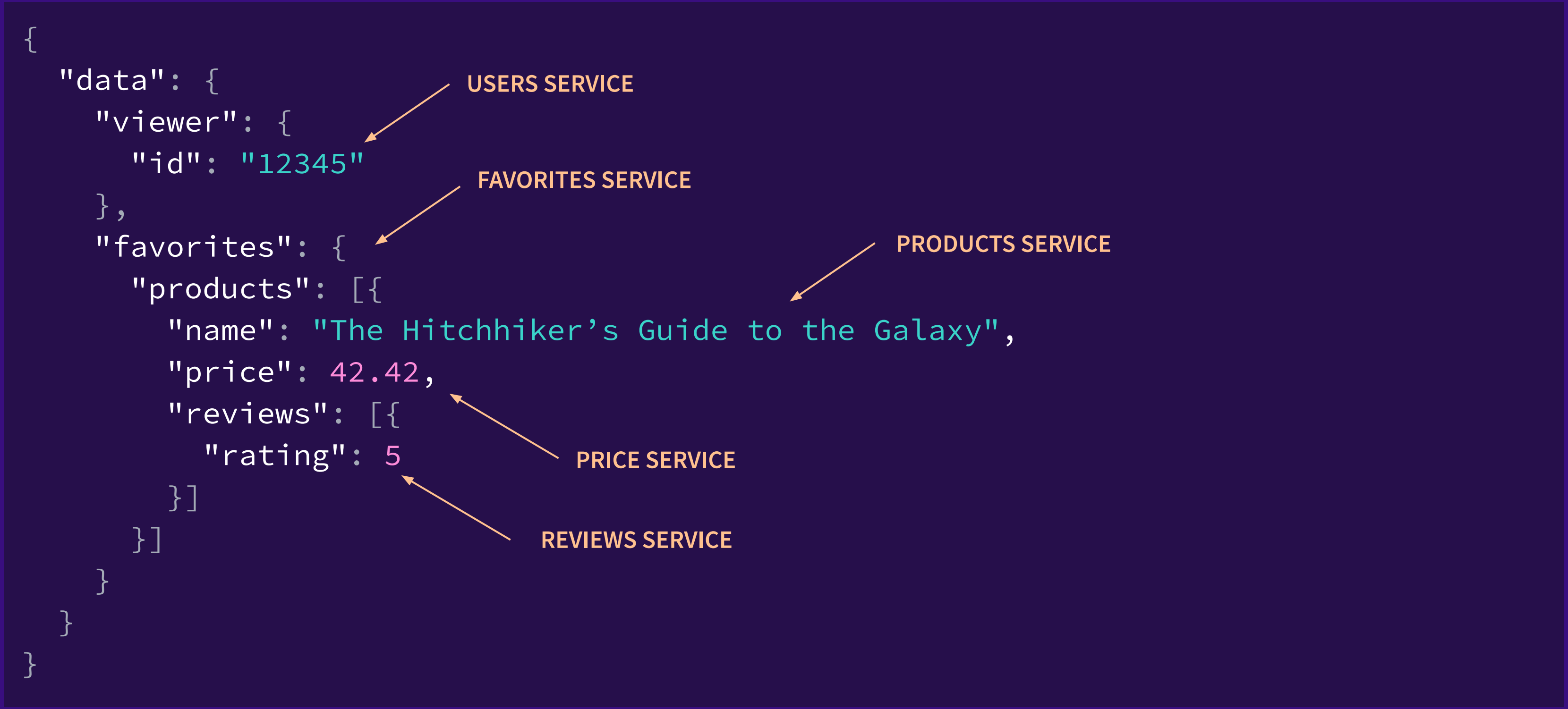
What is GraphQL?

An example query

```
query FavouriteProducts {
  viewer {
    id
  }
  favorites(orderBy: CREATEDAT_DESC) {
    products {
      name
      price
      reviews {
        rating
      }
    }
  }
}
```

What is GraphQL?

An example response



```
{
  "data": {
    "viewer": {
      "id": "12345"
    },
    "favorites": {
      "products": [{
        "name": "The Hitchhiker’s Guide to the Galaxy",
        "price": 42.42,
        "reviews": [{
          "rating": 5
        }]
      }]
    }
  }
}
```

What is GraphQL?

An example schema

```
type User {
  id: ID!
}

type Product {
  upc: String!
  reviews: [Review]
}

type Review {
  id: ID!
  rating: Int
  product: Product
}

type Query {
  favorites: [Product]
  viewer: User
}
```

What is GraphQL?

How resolvers work

```
const resolvers = {
  Product: {
    reviews(product, args, context, info) {
      return fetchReviewsForProduct(product.upc);
    }
  },
  Review: {
    product(review, args, context, info) {
      return fetchProductByUpc(review.productUpc);
    }
  },
  Query: {
    favorites(parent, args, context, info) {
      return fetchUserFavorites(context.userId);
    },
    viewer(parent, args, context, info) {
      return fetchUser(context.userId);
    }
  }
}
```

And that’s it!

The challenge of scale

The challenge of scale

Solving the challenge of creating a GraphQL API from many parts

- Explosion in open source offerings
- Initial work on tracing turned into a paid for SaaS: Apollo Optics
- Schema stitching was created as a way to combine GraphQL services
- Apollo Federation was launched in May 2019 as next generation solution to the problem
- Based on a model that more closely aligns with larger development teams

What do these companies have in common?

APOLLO

PayPal
globoplay
Zillow
wayfair
flexport.
usbank
RE/MAX
glassdoor
RS
priceline
Adobe
NETFLIX
Booking.com
RH
expedia group
okcupid
Levi's
World Fuel Services
American Airlines
experian
DOORDASH
PELOTON
HILTI
TechStyle FASHION GROUP
Atlassian
THE CLIMATE CORPORATION
EPICOR
Walmart
MLS
Queensland Government
Department of Housing and Public Works
Chegg
Charter COMMUNICATIONS
Square
The New York Times
Achievers
STITCH FIX
sky betting & gaming
pandora
intuit
COMCAST
Nutrien
Varo
HyVee
Audi

The Graph

All these companies face **unprecedented challenges** delivering great digital product fast

APOLLO

Omni-channel complexity

Cohesive experiences on all devices, all platforms

×

Service complexity

Cloud native & open source are accelerating service creation & evolution

×

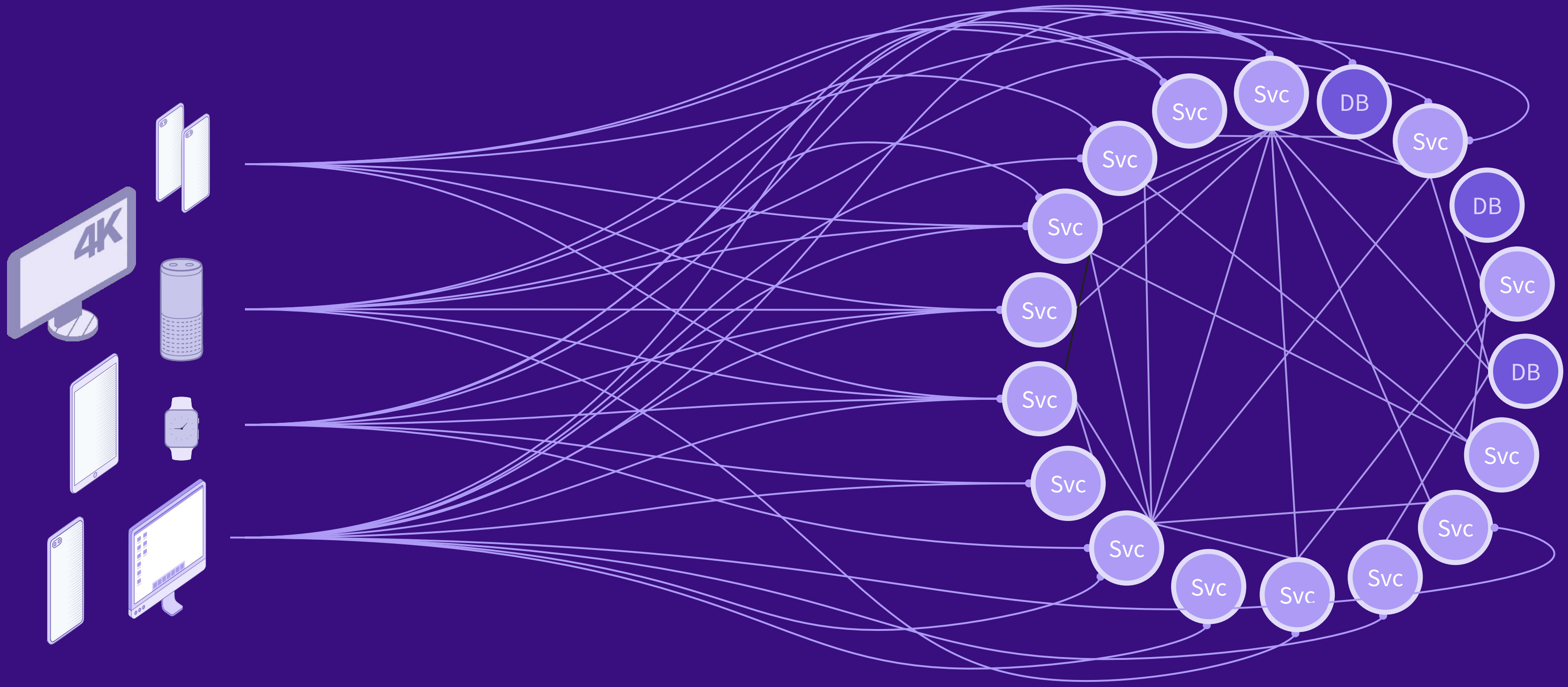
Competitive pressure

Moats are falling while delivery expectations rise

creating a

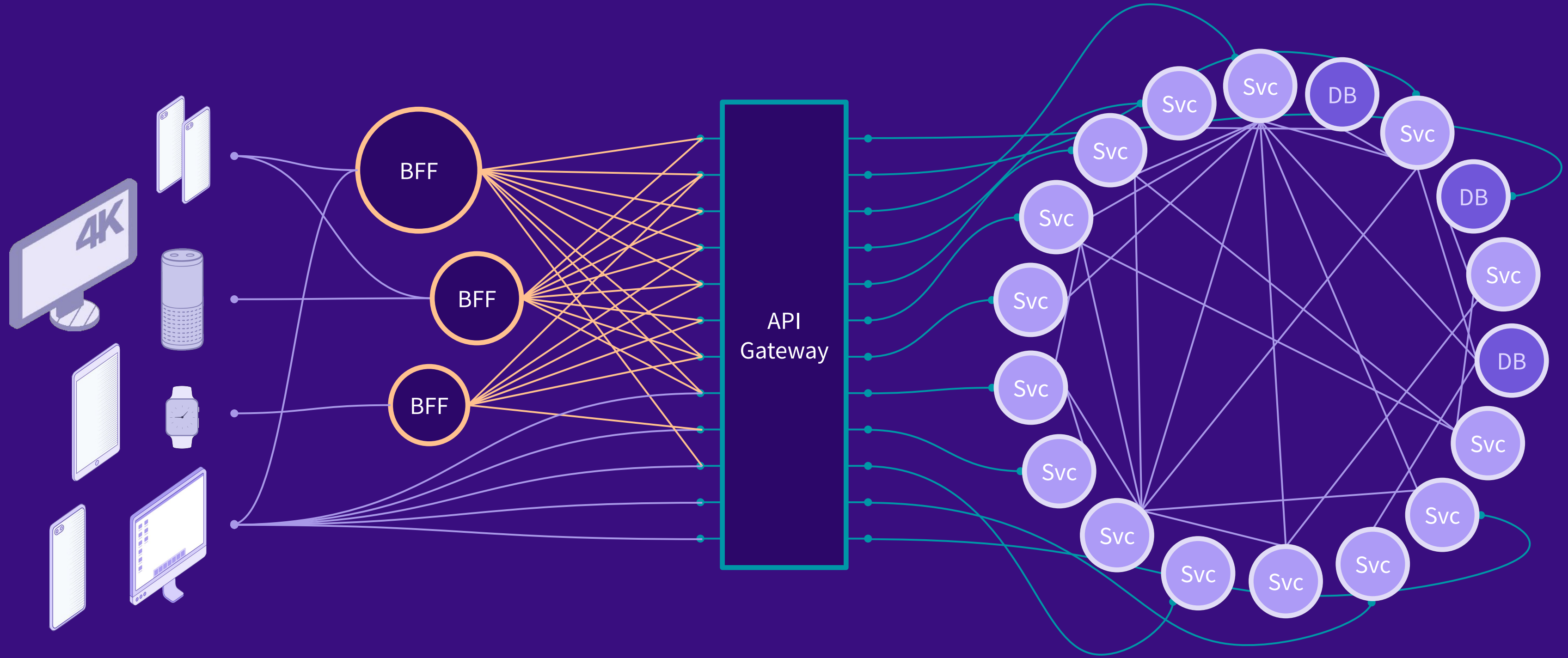
complexity bottleneck

Apps are exposed to full **end-to-end complexity**



App teams spend **⅔ of their time** on service integration
Customer experiences are inconsistent across channels

Past attempts to manage complexity have **failed**



API Gateways only address operational concerns. BFFs add duplication and complexity.
Teams are still tightly coupled, complexity still reigns.

Negative consequences of the complexity bottleneck

Productivity killer
Dev teams waste ⅔ of their time on API integrations, productivity declines *every* year, backlogs grow while deadlines are shorter

Fragmented customer experiences
Customer experiences become fragmented without expensive & time consuming alignment and duplication of effort

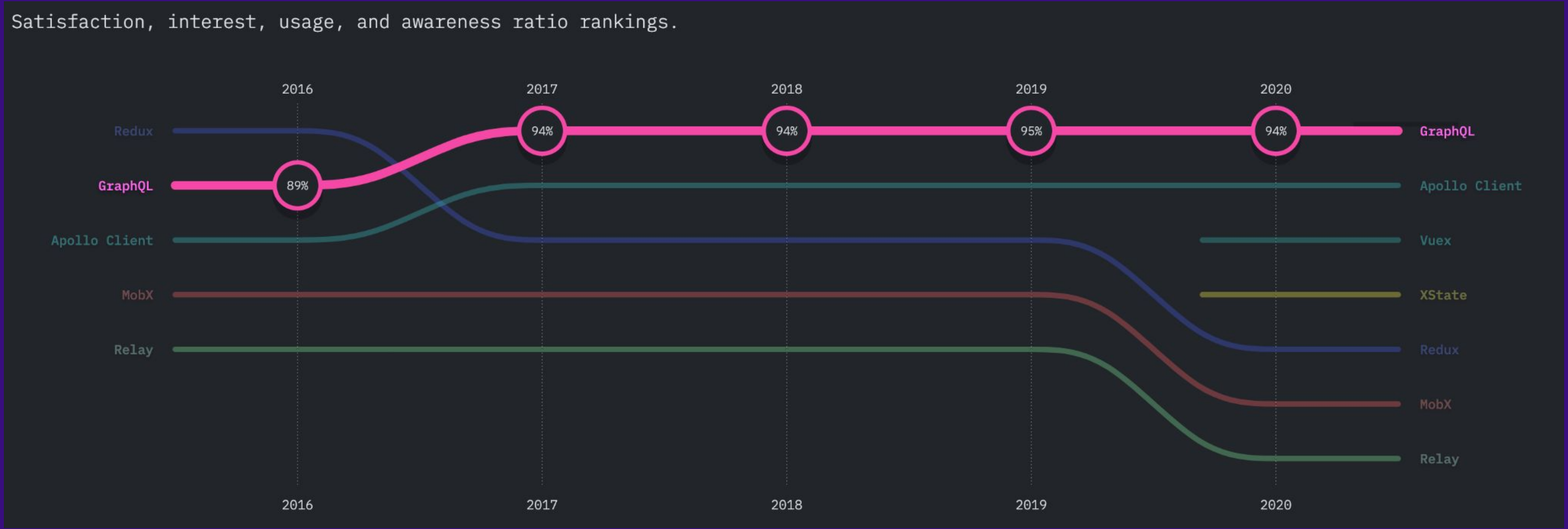
Tech debt burden
Code is written, re-written, thrown away; re-platforming is stifled. Sacrificing quality for speed

Stifled innovation & frustration
Managing complexity leaves no time to deliver new, differentiated experiences

The good news: a **solution exists**

A growing number of companies are adopting GraphQL to solve their complexity bottleneck

And they are doing it **strategically**

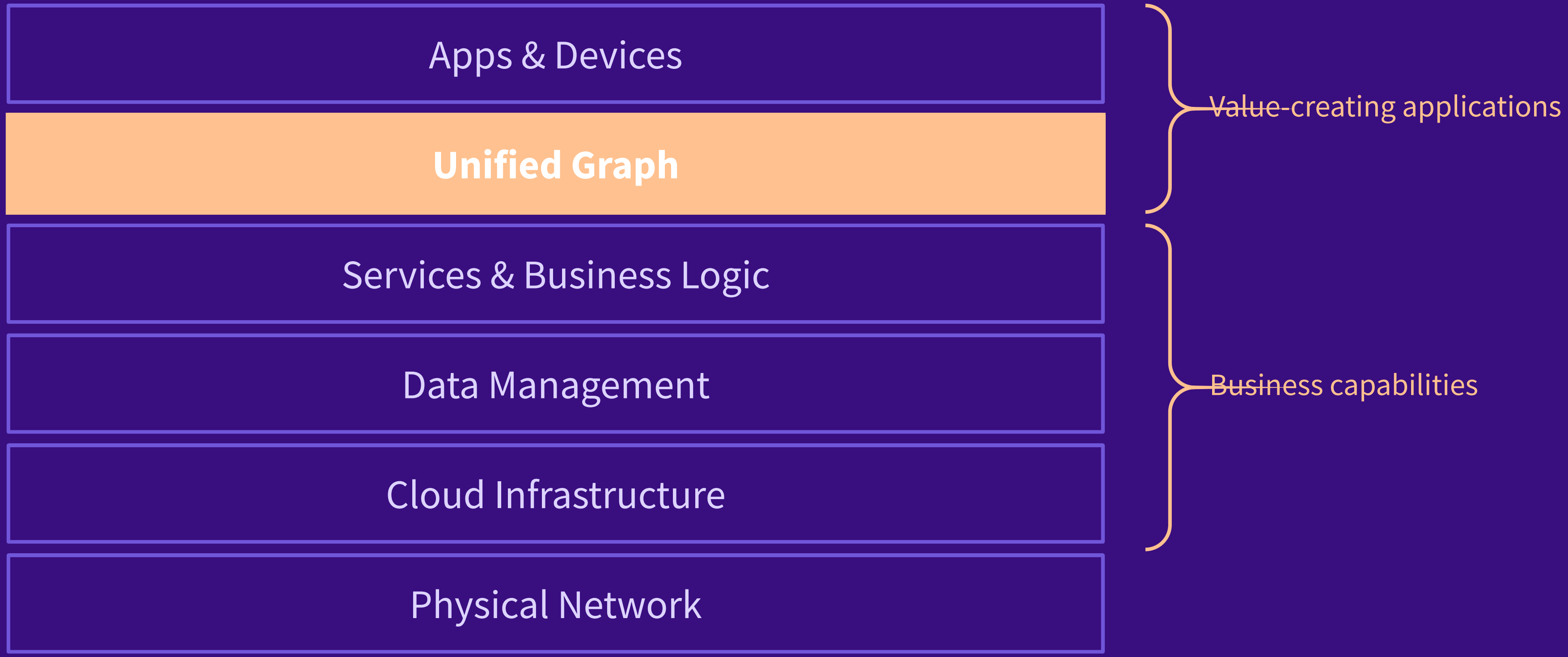


SOURCE: STATE OF JS 2020

A new and essential part of the **modern tech strategy**

APOLLO

The **Unified Graph** connects frontend and backend developers without tightly coupling them



The Graph's essential elements

An API built for building products

Freed from managing service endpoints and orchestration, app devs can focus on experiences not integrations

A query language tailored for use

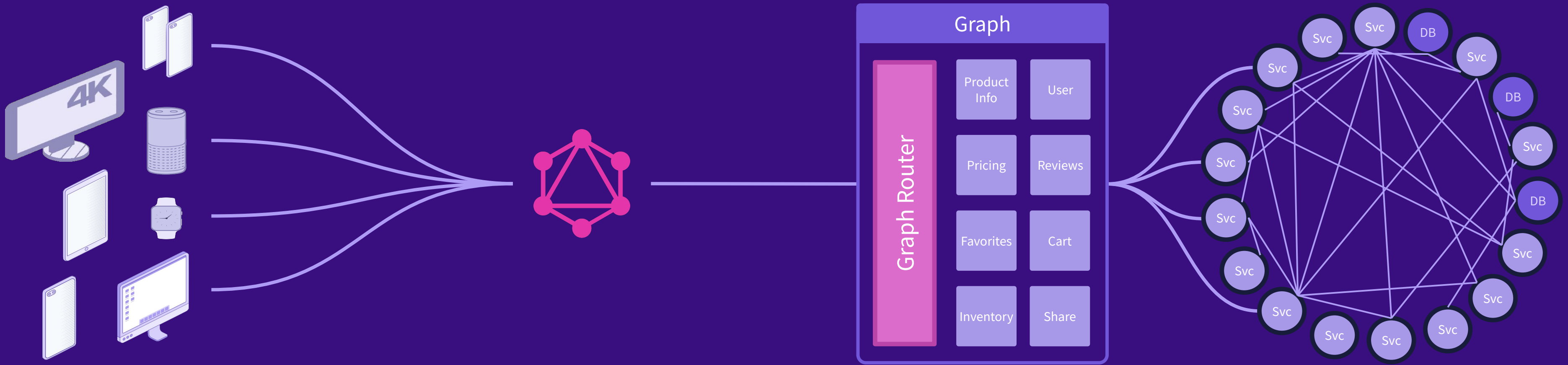
Apps pick what they need, from a shared common contract, optimizing performance and removing complexity from each app

Unified representation of your services, data & digital capabilities

Each capability is expressed as a declarative abstract contract via a schema

An insulating layer for service complexity

Decoupled from direct app requests, service teams can focus on optimizing capabilities and architecture w/out fear of breaking changes



The Graph's collaboration model

App teams bring their usage expertise

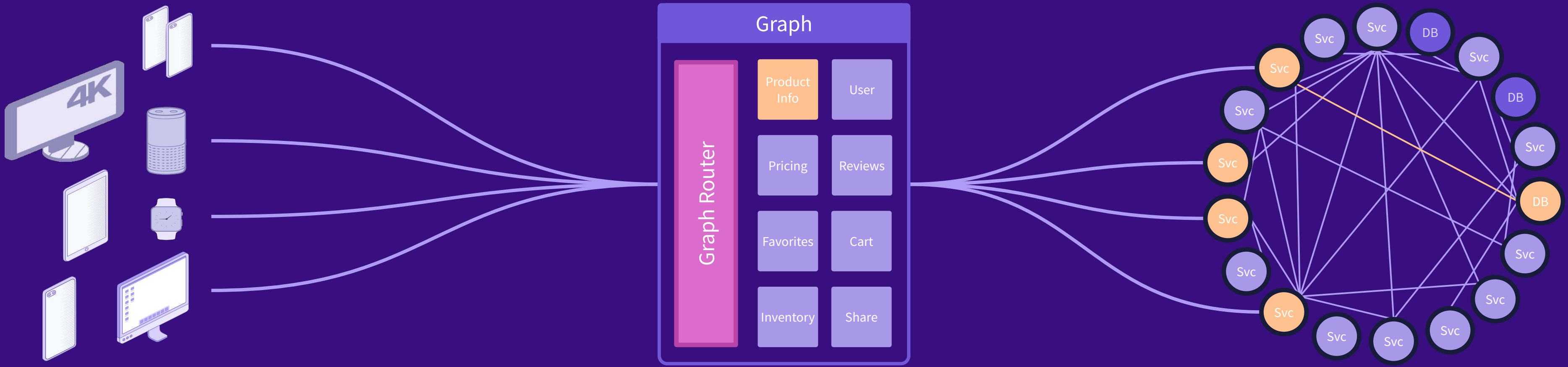
App teams know best the shape of an "ideal" API. Information Architects and Designers can be key

Schema collaboration yields the best abstraction "for now"

Teams propose and debate alternate ideas, using schema best practices to capture the best balance between app needs and service realities

Service teams bring their domain expertise

Service teams are deeply involved during initially schema design. Once defined they have a safer, faster way to address app needs w/out versioning and managing client migrations.



The Graph drives upfront collaboration on product centric contracts

Iteration replaces perfectionism and versioned complexity

The impact of Graph

Product velocity and a cohesive UX

Freed from service complexity, devs deliver rich omnichannel customer experiences in less time

A more streamlined and faster UX

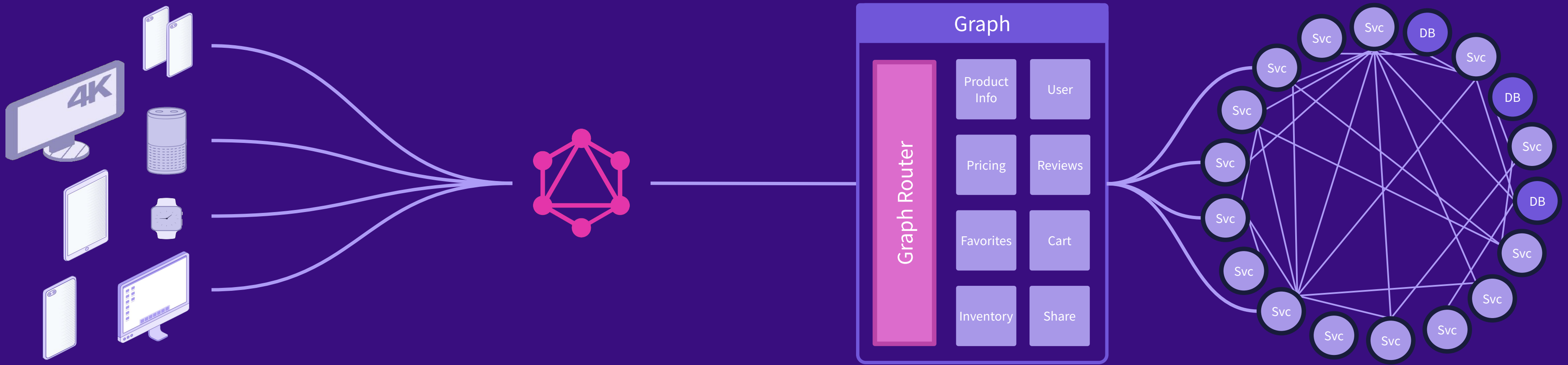
Apps ask for just what they need, with server-side orchestration optimizing their performance

A lasting home for all product capabilities

Each underlying capability adds to the whole, unlocking richer experiences, new use cases and business models

Service backlogs shorten, more service innovation

Decoupled from managing direct app requests, service teams move faster to evolve and replatform services without impact to clients

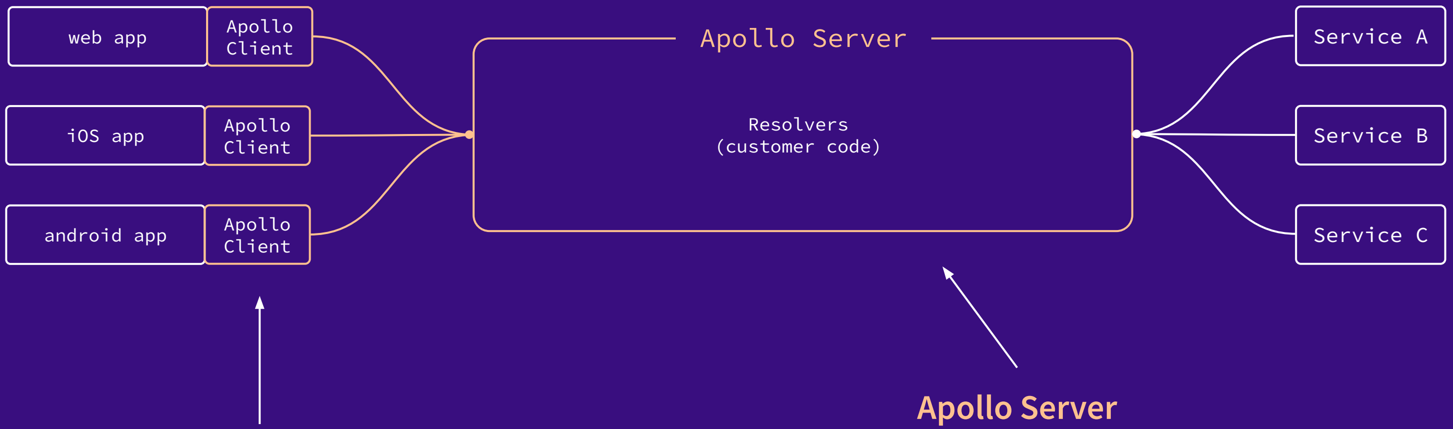


The Graph becomes the single source of truth and point of collaboration for all teams

Ultimately the Graph **becomes** your product

Apollo Federation

Before Federation: a single server



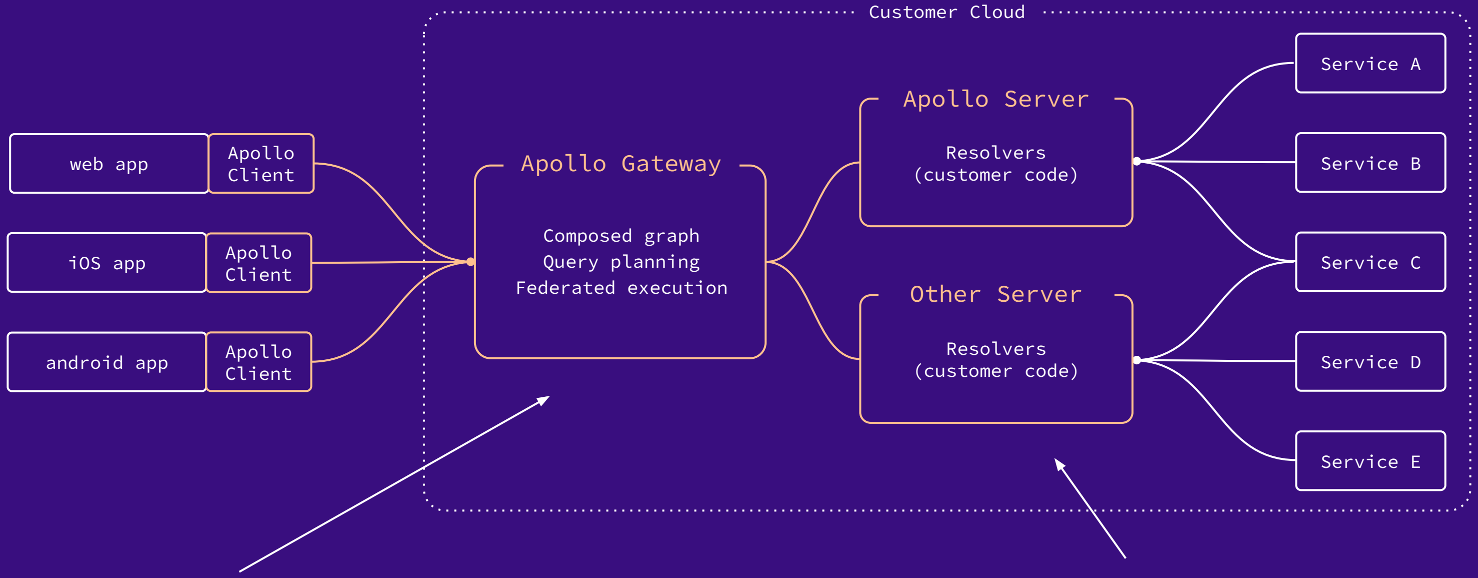
Apollo Client

A framework that lets you define queries (what you want) and connect them to UI components

Apollo Server

A framework that lets you define a schema (what you have) and connect it to underlying services

After Federation: single team → multiple teams



Apollo Gateway

Splits up a query and executes it against multiple subgraphs

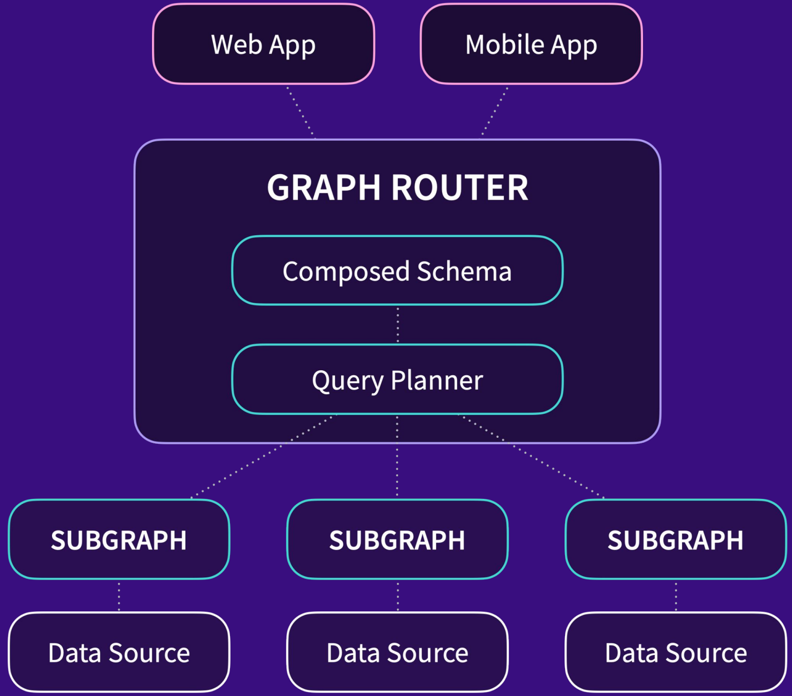
Subgraphs

Multiple services implemented in Apollo Server or similar frameworks that each implements one slice of a unified schema

Composability across the stack

APOLLO

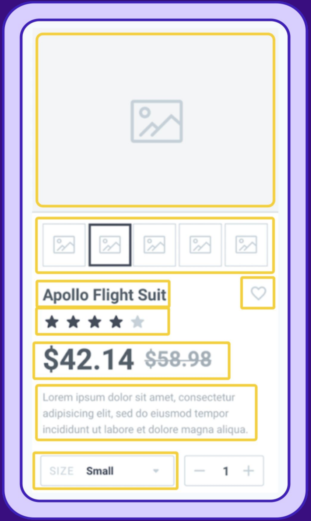
Apollo Federation introduces the concept of composability to your APIs



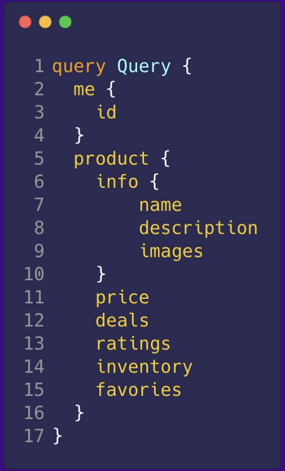
The Graph in action

The Graph in action

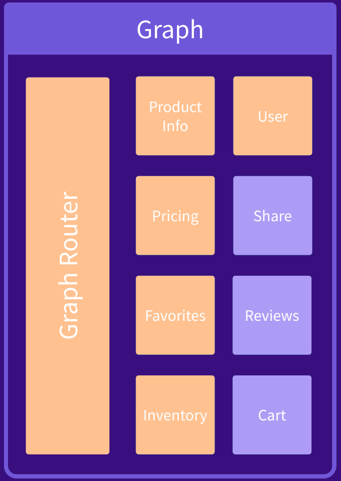
Native app product page example

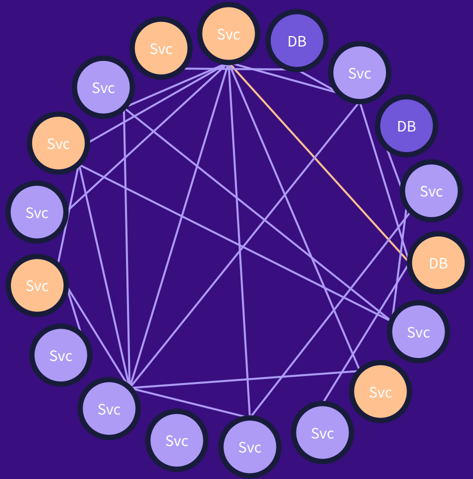


Apollo Flight Suit
$42.14 ~~$58.98~~
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
SIZE Small
1



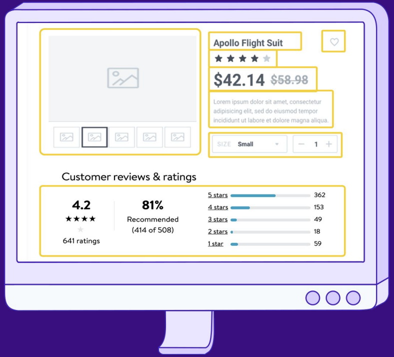
```
query Query {
  me {
    id
  }
  product {
    info {
        name
        description
        images
    }
    price
    deals
    ratings
    inventory
    favories
  }
}
```

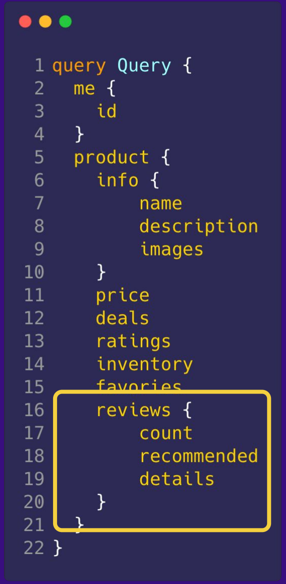




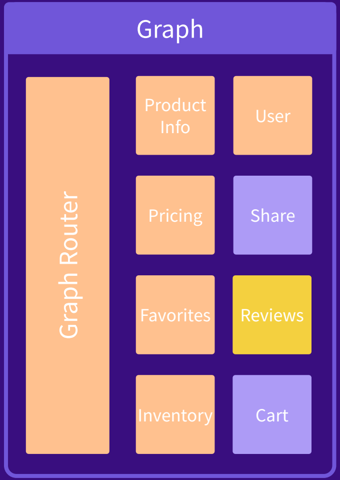
The Graph in action

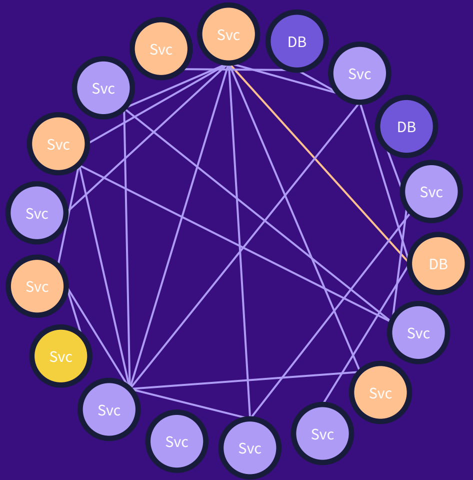
Desktop web product page example


Apollo Flight Suit
★★★★
$42.14 ~~$58.98~~
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
SIZE Small
− 1 +
Customer reviews & ratings
4.2
★★★★
641 ratings
81%
Recommended
(414 of 508)
5 stars 362
4 stars 153
3 stars 49
2 stars 18
1 star 59



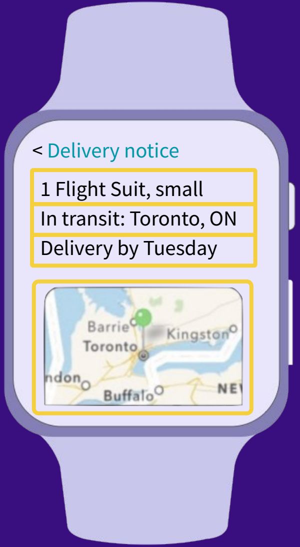
```
query Query {
  me {
    id
  }
  product {
    info {
        name
        description
        images
    }
    price
    deals
    ratings
    inventory
    favories
    reviews {
        count
        recommended
        details
    }
  }
}
```

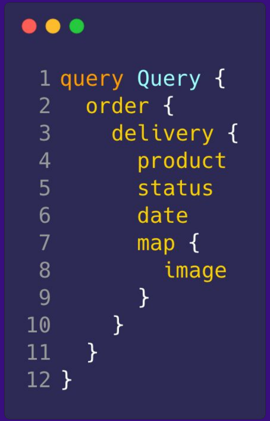




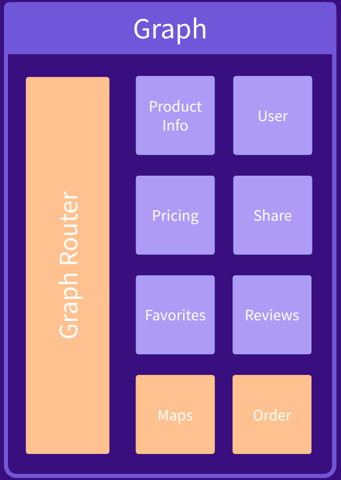
The Graph in action

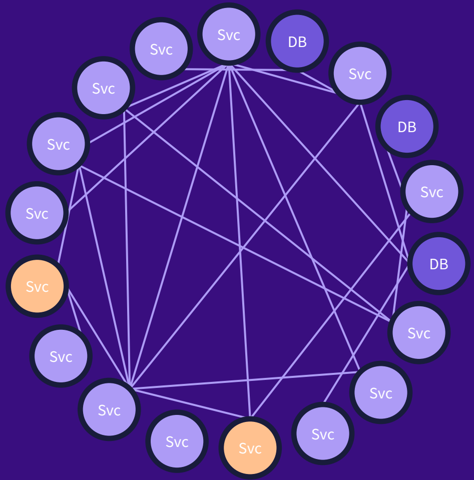
Wearable delivery confirmation app example





```
query Query {
  order {
    delivery {
      product
      status
      date
      map {
        image
      }
    }
  }
}
```





The future

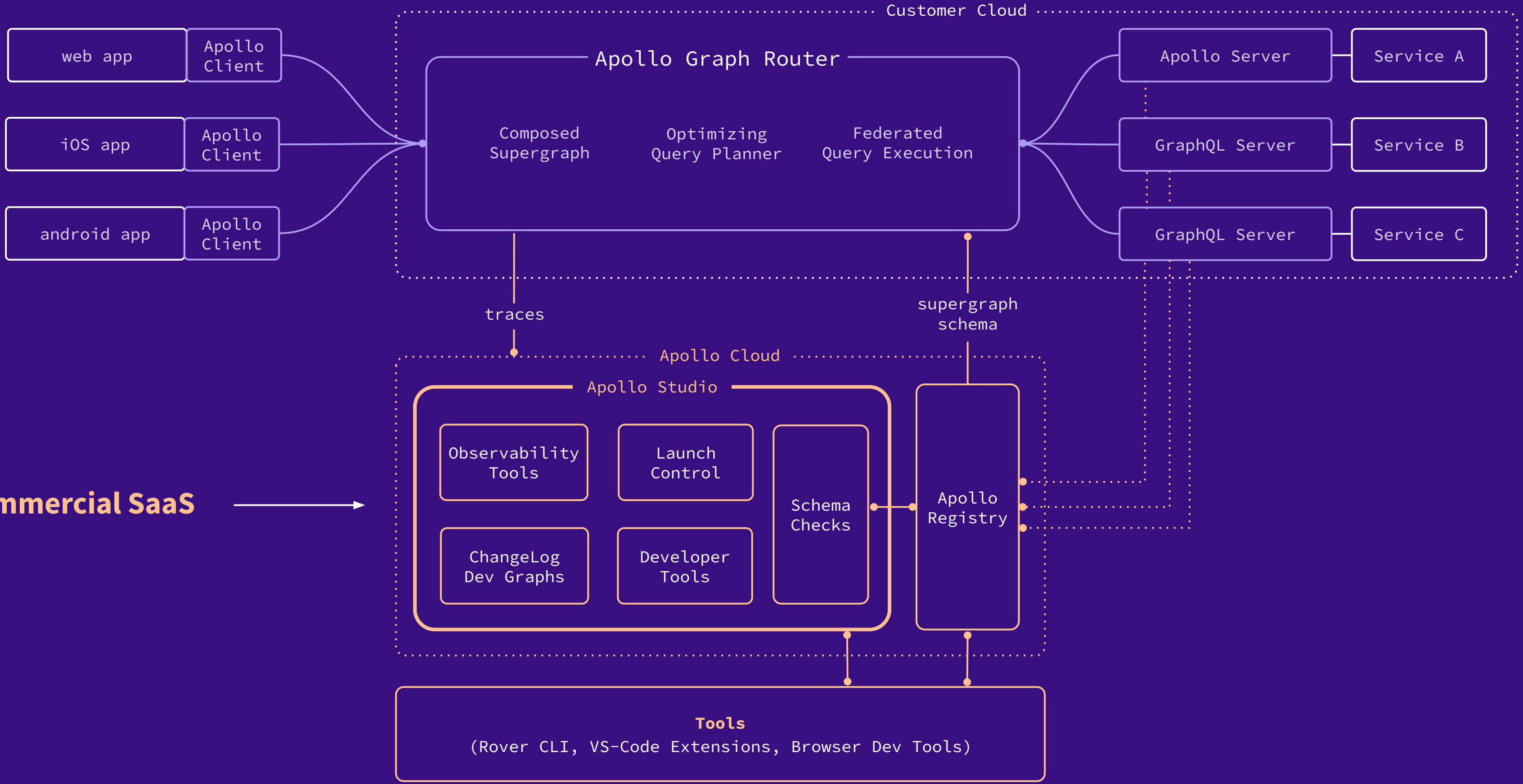
The future

The tools that enterprises need, the open source that developers need

- Apollo Federation 2 went GA in April 2022
- Contracts went GA in May 2022
- Graph Router went GA in May 2022
- Apollo Odyssey is looking to become the defaqto GraphQL learning resource
- Continued investment in our other open source offerings

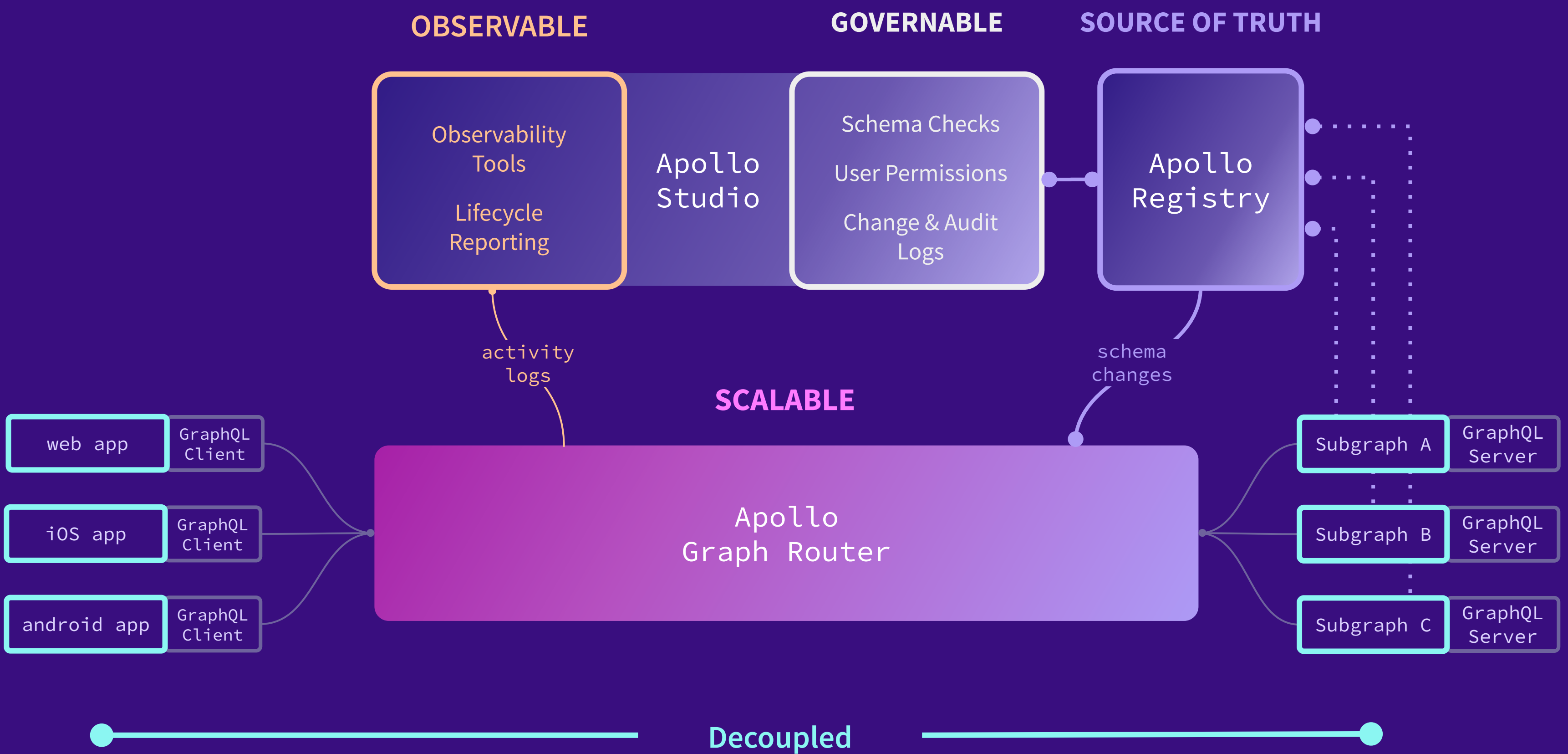
Apollo Platform

The Apollo Platform adds air-traffic control

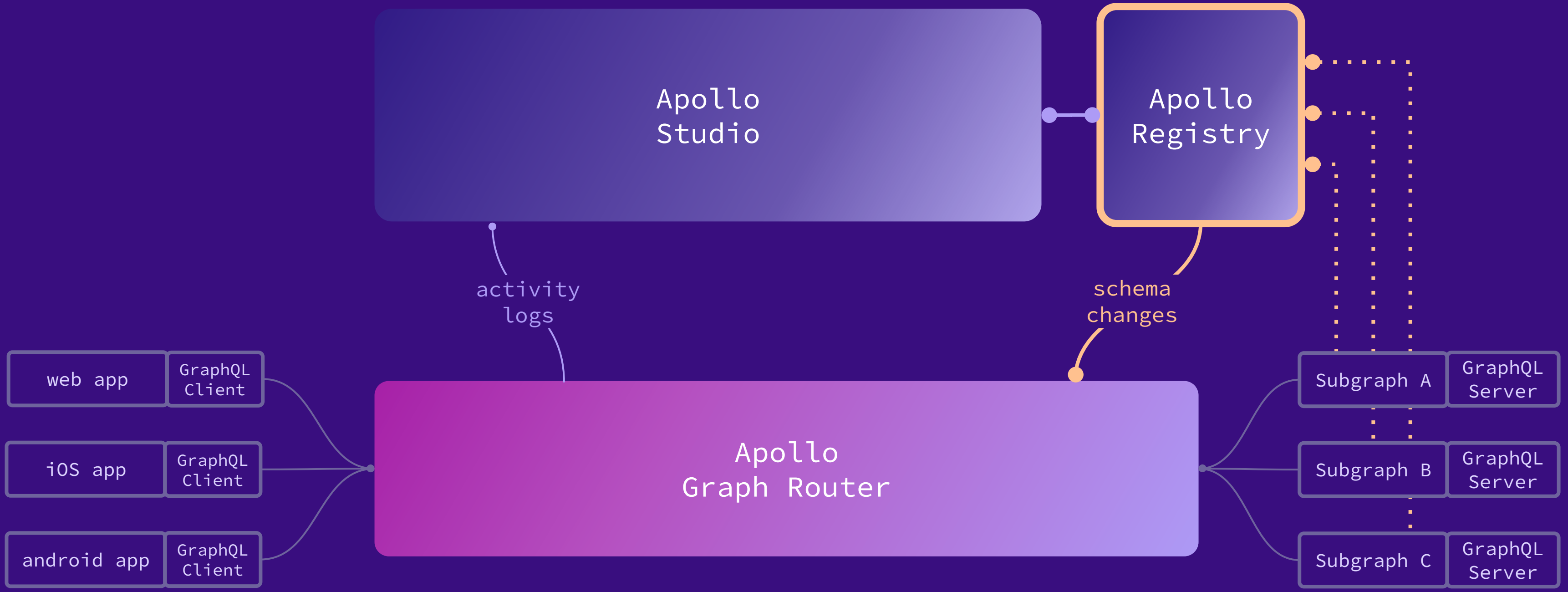


Commercial SaaS →

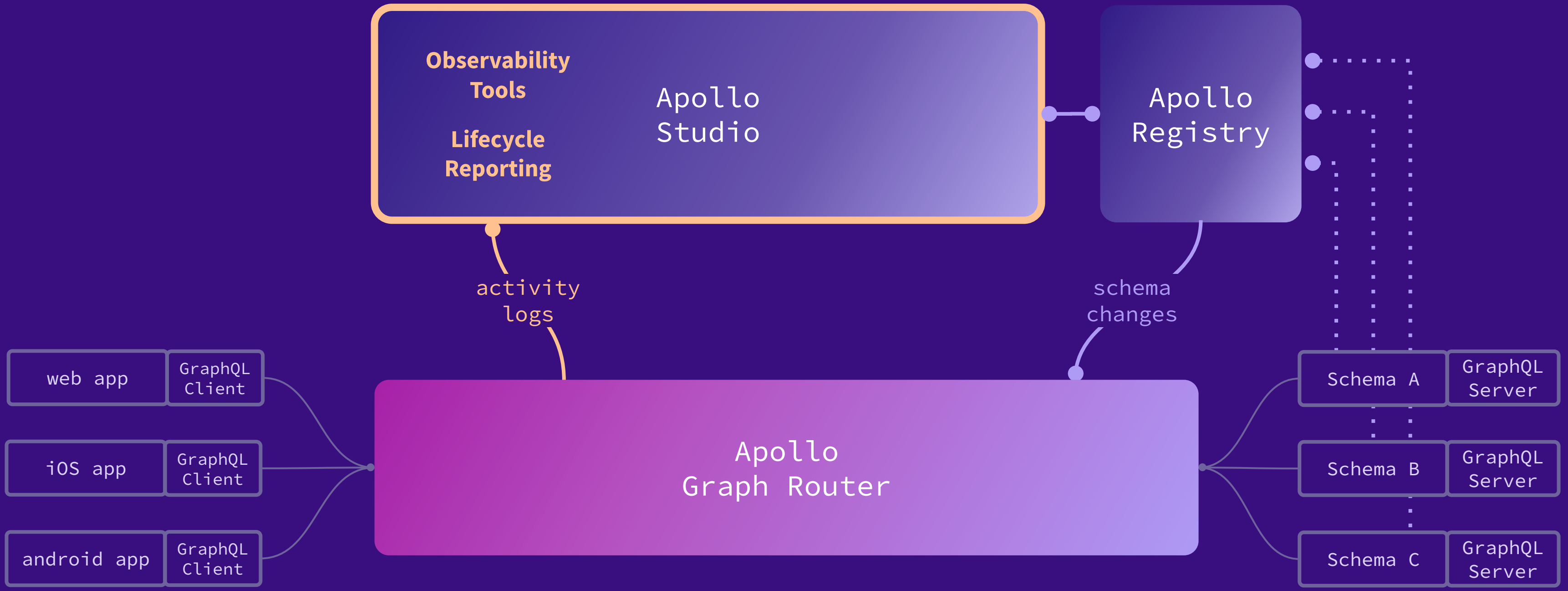
The Apollo Platform **enables** your unified graph strategy



The Apollo Registry is your single source of truth



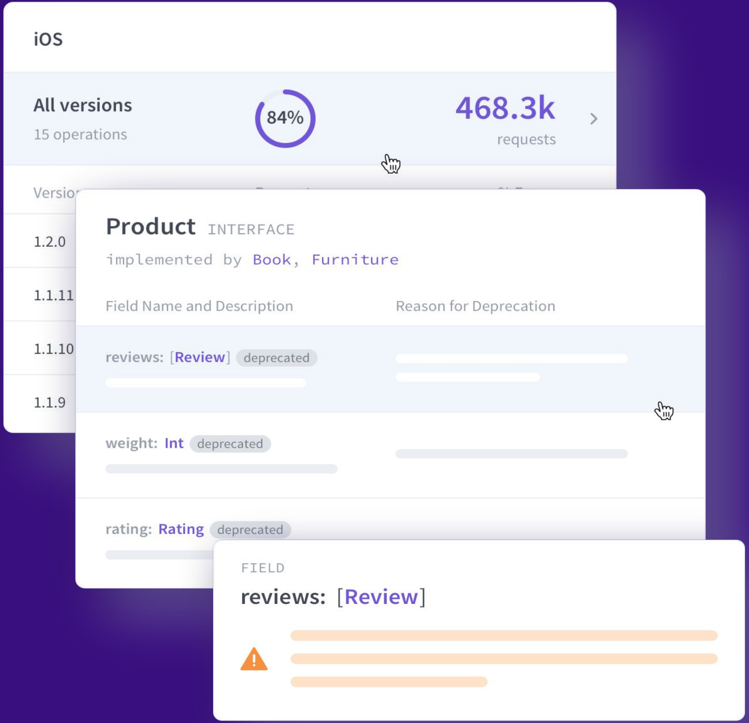
The Apollo Registry is your **single source of truth**



Observability yields insights

APOLLO

Understand how clients are consuming your graph



iOS

All versions
15 operations
84%
468.3k
requests

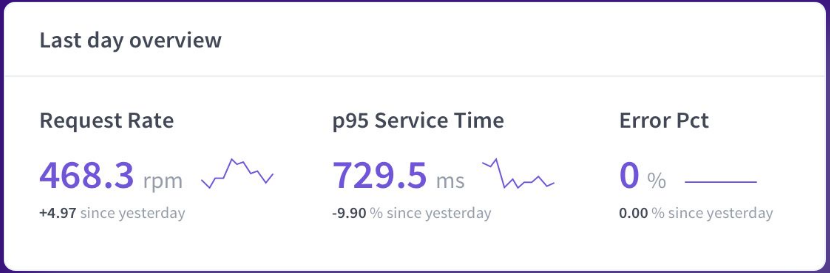
Version
1.2.0
1.1.11
1.1.10
1.1.9

Product INTERFACE
`implemented by Book, Furniture`

Field Name and Description	Reason for Deprecation
reviews: [Review] deprecated	
weight: Int deprecated	
rating: Rating deprecated	

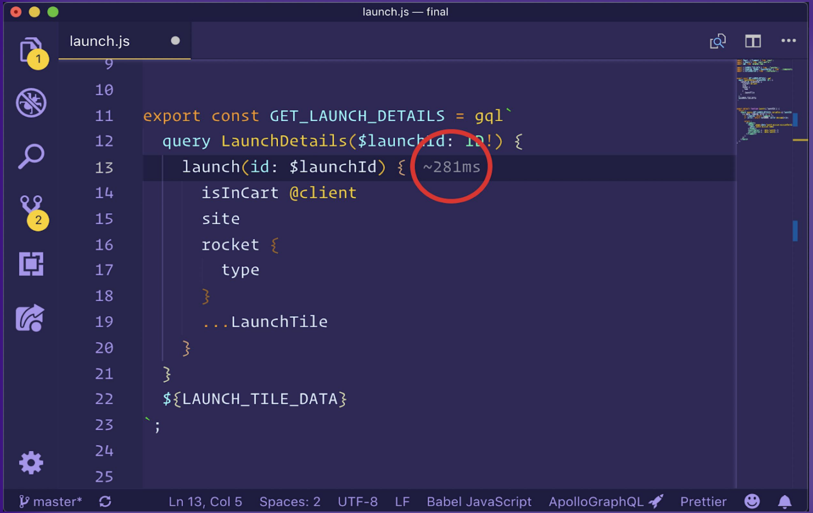
FIELD
reviews: [Review]

Design time performance data



Last day overview

Request Rate	p95 Service Time	Error Pct
468.3 rpm	729.5 ms	0 %
+4.97 since yesterday	-9.90 % since yesterday	0.00 % since yesterday



launch.js — final

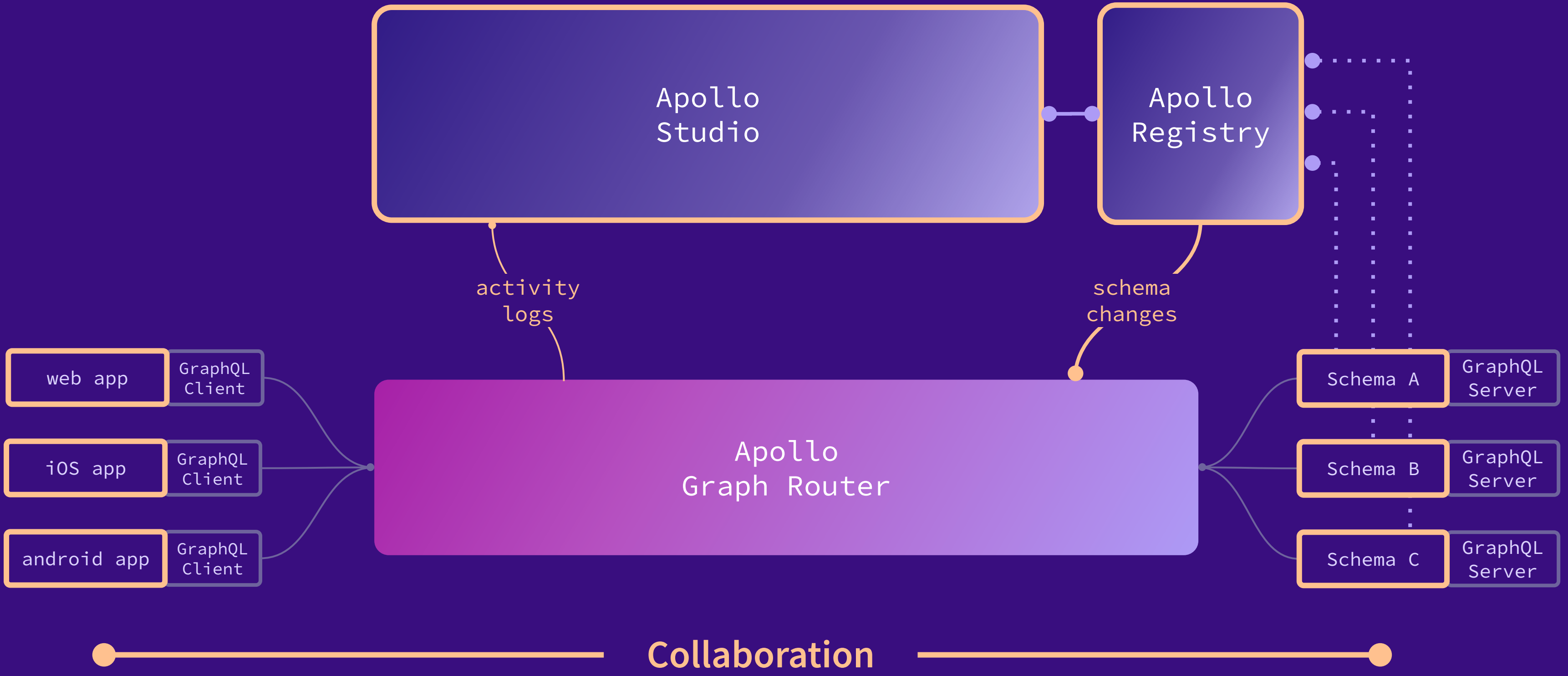
launch.js

```
export const GET_LAUNCH_DETAILS = gql`
  query LaunchDetails($launchId: ID!) {
    launch(id: $launchId) { ~281ms
      isInCart @client
      site
      rocket {
        type
      }
      ...LaunchTile
    }
  }
  ${LAUNCH_TILE_DATA}
`;
```

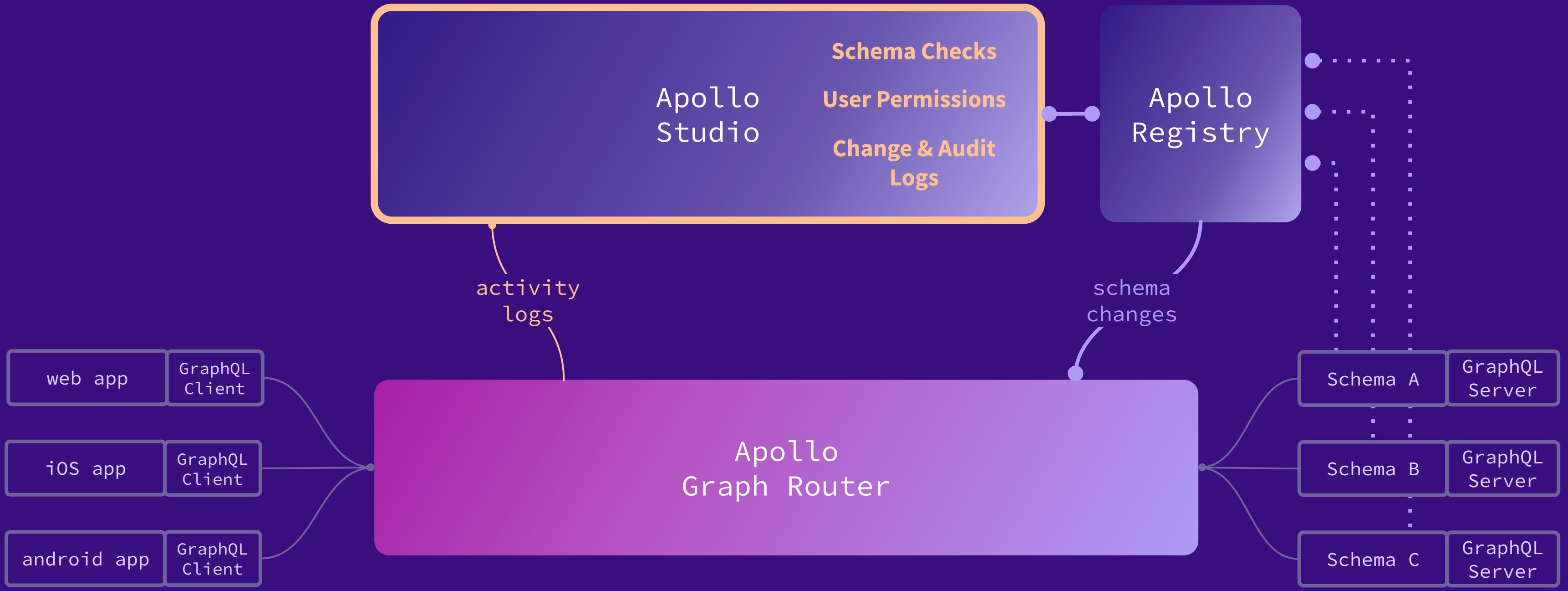
master* Ln 13, Col 5 Spaces: 2 UTF-8 LF Babel JavaScript ApolloGraphQL Prettier

Collaboration without coupling

Your teams work together on a **shared contract**, not a specific endpoint or version

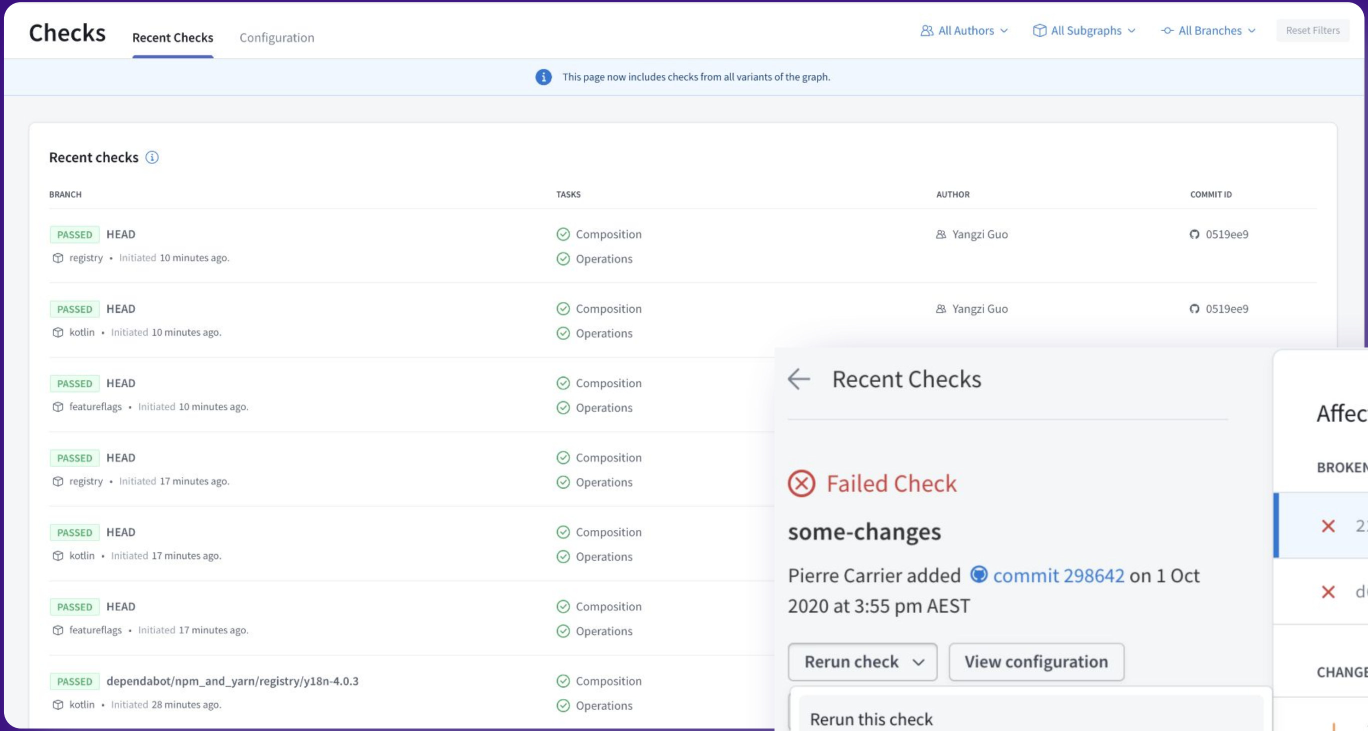


Governable means speed with safety



Governable means speed with safety

Schema checks ensure your teams find issues at development time



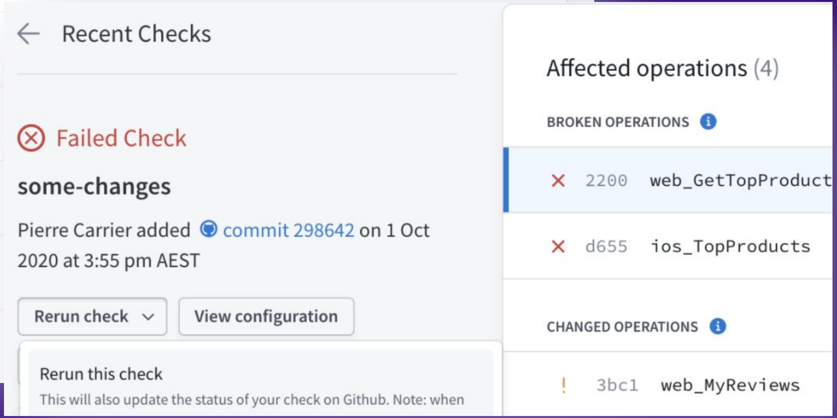
Checks | Recent Checks | Configuration

All Authors | All Subgraphs | All Branches | Reset Filters

This page now includes checks from all variants of the graph.

Recent checks

BRANCH	TASKS	AUTHOR	COMMIT ID
PASSED HEAD registry • Initiated 10 minutes ago.	Composition Operations	Yangzi Guo	0519ee9
PASSED HEAD kotlin • Initiated 10 minutes ago.	Composition Operations	Yangzi Guo	0519ee9
PASSED HEAD featureflags • Initiated 10 minutes ago.	Composition Operations		
PASSED HEAD registry • Initiated 17 minutes ago.	Composition Operations		
PASSED HEAD kotlin • Initiated 17 minutes ago.	Composition Operations		
PASSED HEAD featureflags • Initiated 17 minutes ago.	Composition Operations		
PASSED dependabot/npm_and_yarn/registry/y18n-4.0.3 kotlin • Initiated 28 minutes ago.	Composition Operations		



Recent Checks

Failed Check

some-changes

Pierre Carrier added commit 298642 on 1 Oct 2020 at 3:55 pm AEST

Rerun check | View configuration

Rerun this check

This will also update the status of your check on Github. Note: when

Affected operations (4)

BROKEN OPERATIONS

✗ 2200 web_GetTopProduct

✗ d655 ios_TopProducts

CHANGED OPERATIONS

! 3bc1 web_MyReviews

Thank You

Andy Roberts

Senior Manager - EMEA Customer Success

@andyroberts_io andyrobertsio andyroberts.io