

Best kept secrets ⚡ **2019**

GraphQL

+ Prisma

Gist



Current state of JS



Why GraphQL



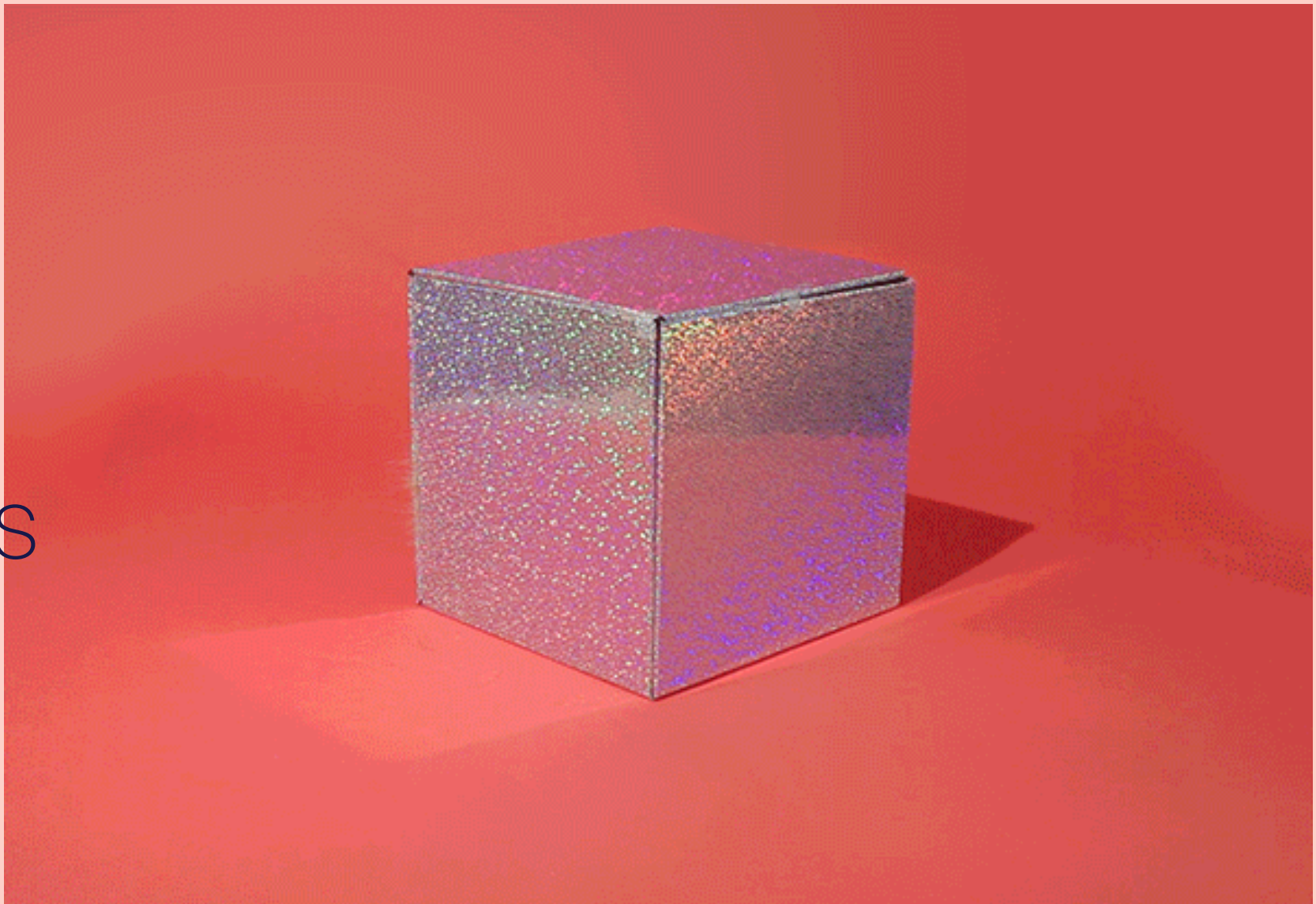
Why Prisma



Demo



Domas Bitvinskas
domasbitvinskas.com



Startup fanatic, co-founder @ Honestive
Freelance developer with React / GraphQL / Prisma

Preface

**This is just my
subjective opinion.**

#2018

JavaScript

Ecosystem



ES6 is the
new default
{...props}

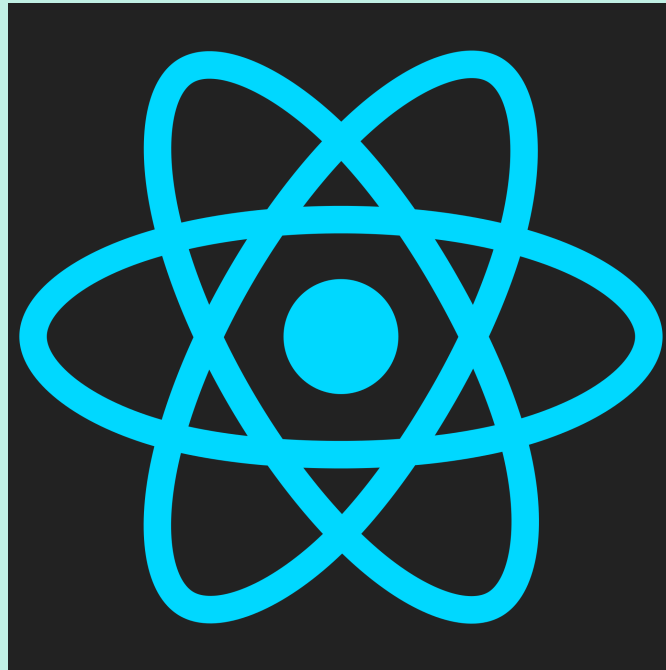


Angular
kinda 🙄



Vue

kinda 🔥



React

still 

Tensorflow.js



Machine Learning in browser

TypeScript & Flow

getting popular



what: string

Promises dead

`async / await`
everywhere



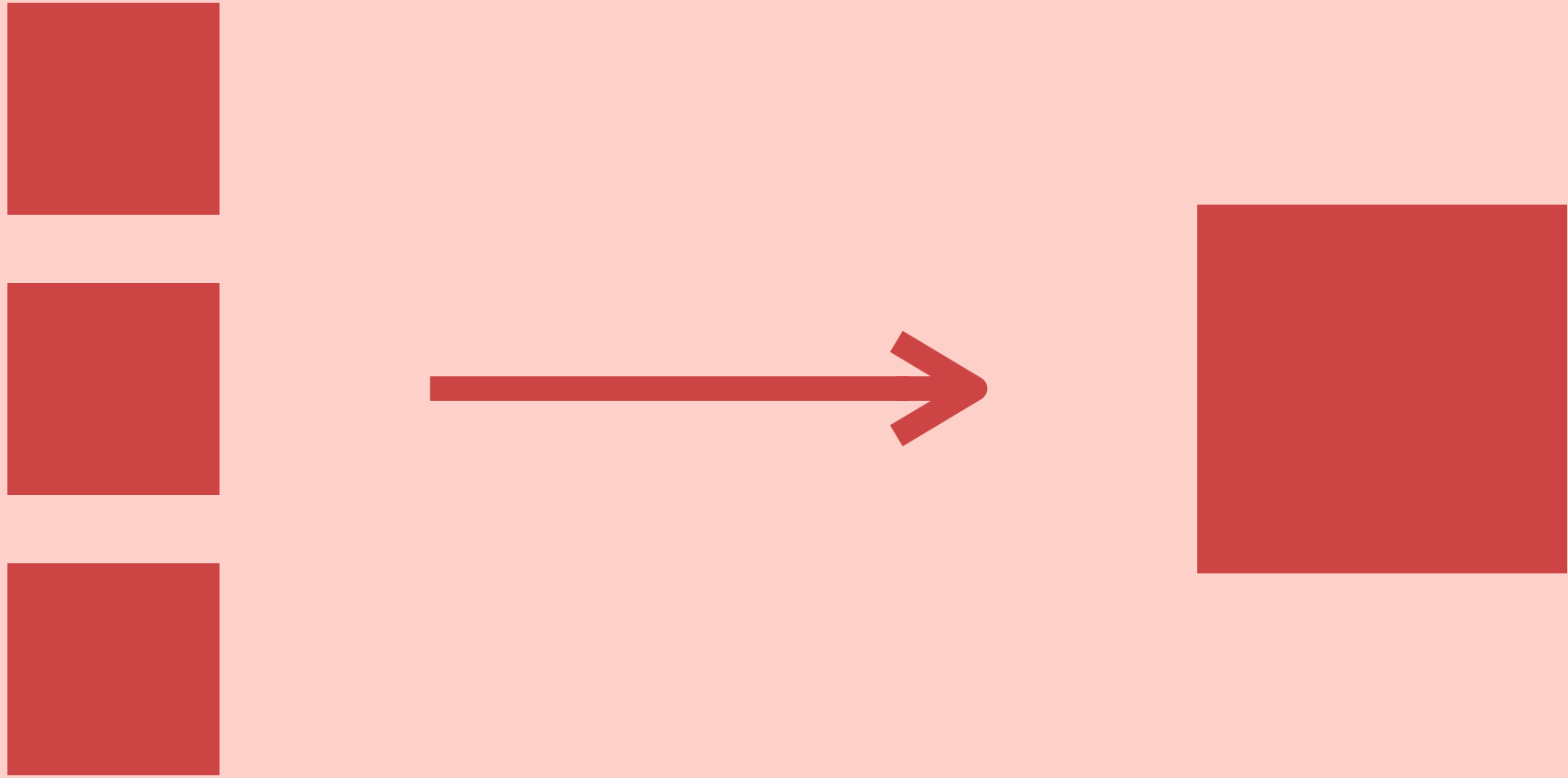
GraphQL

GraphQL

is cool

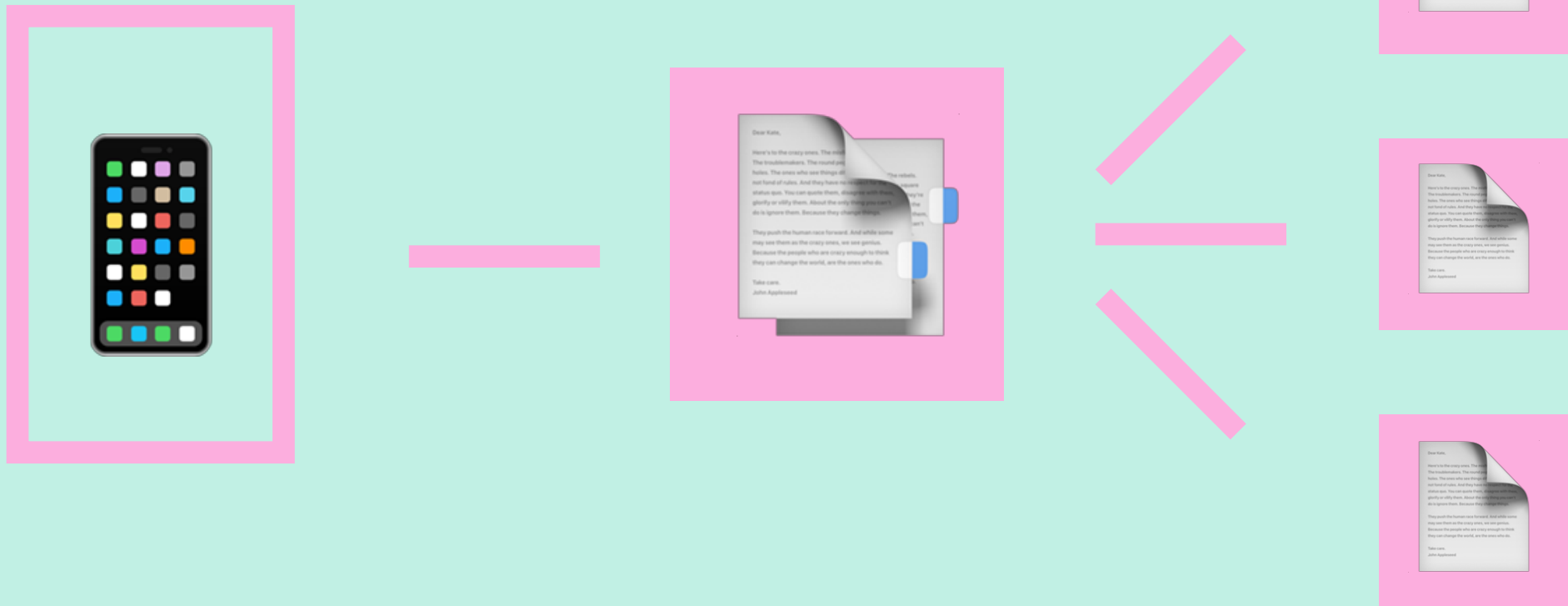
Fetch what you need only

```
posts {  
  title  
  author {  
    avatarUrl  
  }  
}
```



Batch queries





Schema stitching

Way better DX

than REST



Typical GraphQL server structure



Schema / TypeDefs

```
type Query {  
  posts: [Post!]!  
  post(id: ID!): Post!  
}
```

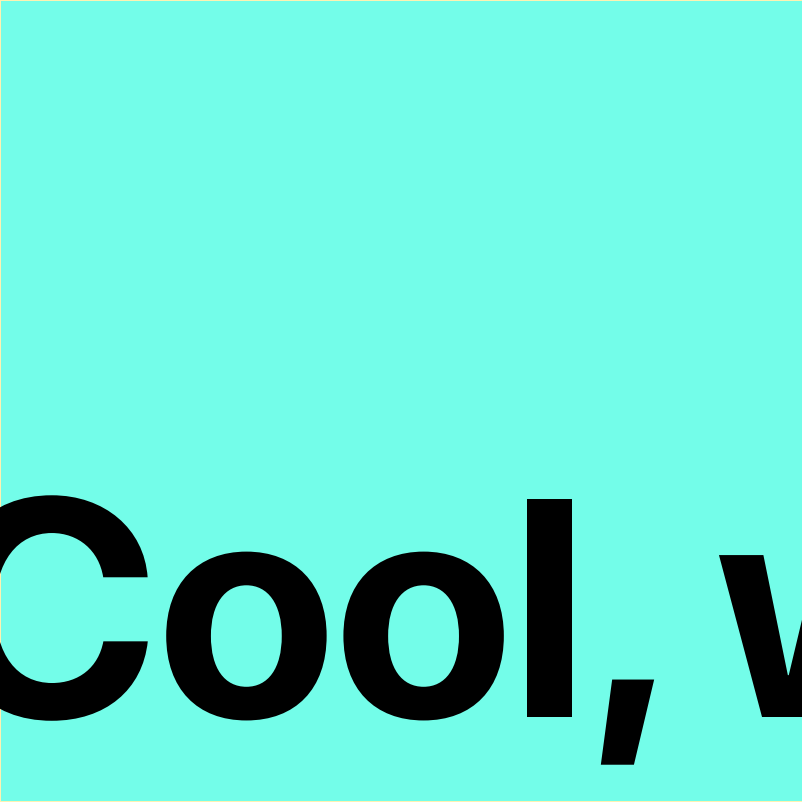
```
type Post {  
  id: ID!  
  title: String  
}
```

Typical GraphQL server structure



Resolvers

```
{
  Query: {
    posts: (_, args) => Post.findAll(),
    post: (_, { id }) => Post.findOne({
      where: {
        id
      }
    })
  }
}
```



**Cool, where's the
problem?**

No real standard
of how messy
your resolvers
should be

```
module.exports = {
  Query: {
    services: (_, { orderBy }) => Service.findAll({
      order: [[orderBy || 'createdAt', 'DESC']]
    }),
    service: (_, { where }) => Service.findOne({
      where,
    }),
    statusReports: (_, { orderBy, where, last }) => StatusReport.findAll({
      where: transformWhere(where),
      order: [[orderBy || 'createdAt', 'DESC']],
      limit: last,
      include: transformInclude(where),
    }),
    statusReport: (_, { where }) => StatusReport.findOne({
      where,
    }),
    loggedIn: (_, _args, ctx) => isLoggedIn(ctx),
  },
  Mutation: {
    createService: (_, { data }, ctx) => ensureLoggedIn(ctx) && Service.create
```



Prisma



Prisma

A data layer that replaces traditional ORMs in your application architecture.

Open-source:

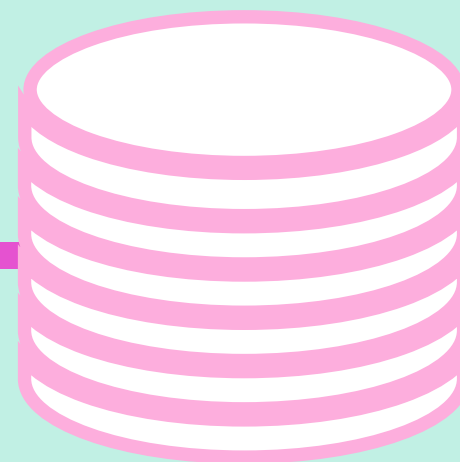
github.com/prisma/prisma ★ 11.6k



Client

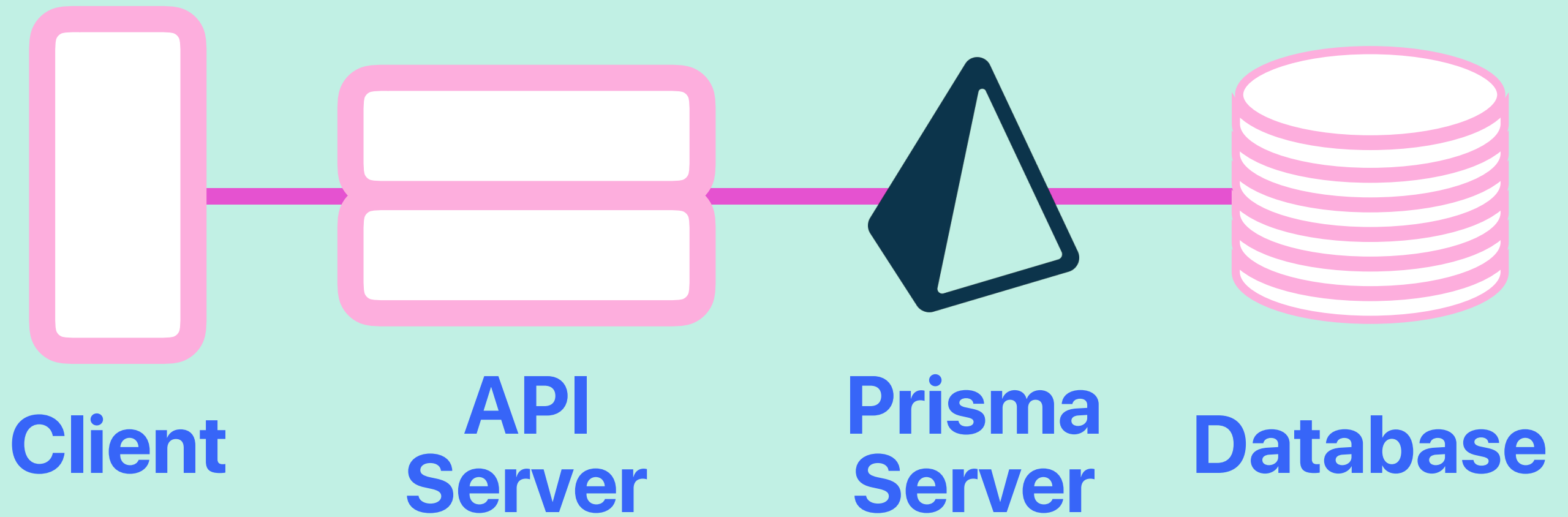


**API
Server**



Database





**Interact with your
database the same
way you communicate
with a GraphQL server**



Define database
tables as
GraphQL schema

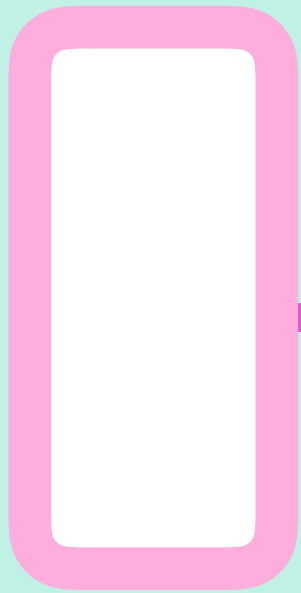


Free CRUD



Demo

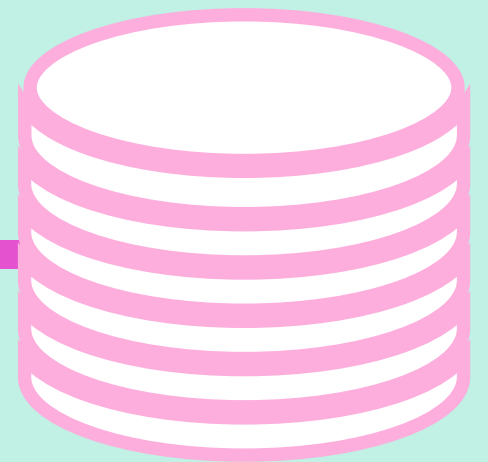




Client

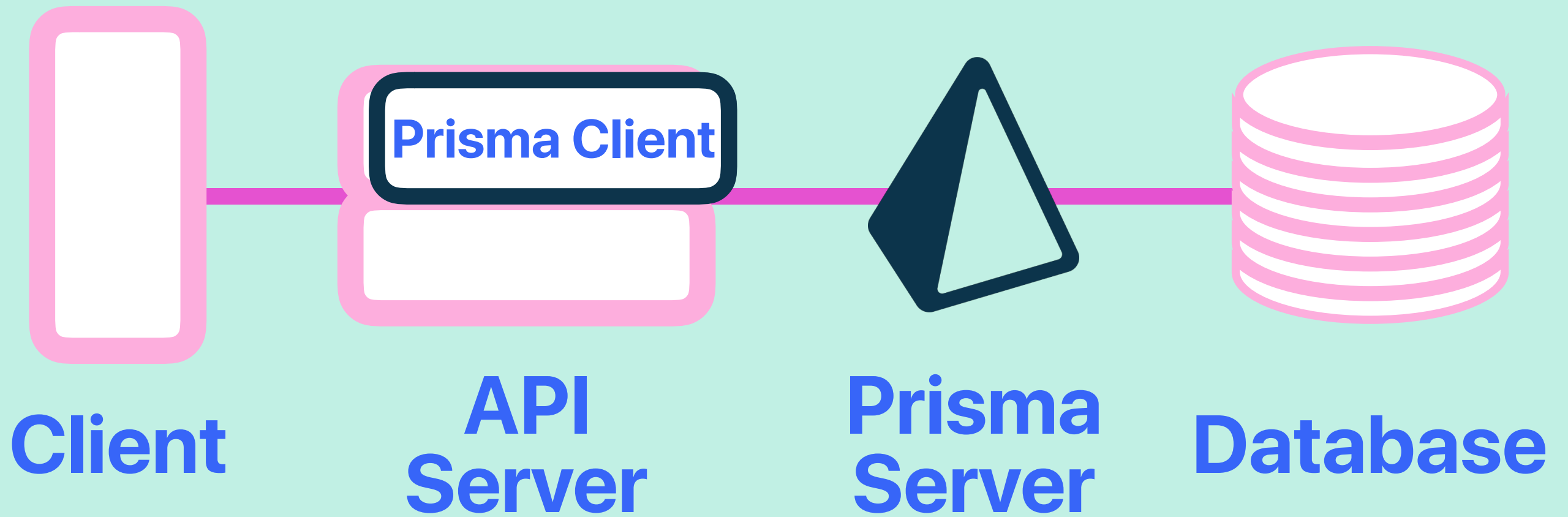


**Prisma
Server**



Database





```
yarn global add prisma
```

```
domas@Macbook-D2:~/Developer/vilniusjs/prisma-test3$ prisma init
? Set up a new Prisma server or deploy to an existing server? Demo server
? Choose the region of your demo server domas-bitvinskas/demo-eu1
? Choose a name for your service prisma-test3
? Choose a name for your stage dev
? Select the programming language for the generated Prisma client Prisma JavaScript Client
```

Created 2 new files:

<code>prisma.yml</code>	Prisma service definition
<code>datamodel.prisma</code>	GraphQL SDL-based datamodel (foundation for database)

Next steps:

1. Deploy your Prisma service: `prisma deploy`
2. Read more about deploying services:
<http://bit.ly/prisma-deploy-services>

Generating schema... 28ms

Saving Prisma Client (JavaScript) at /Users/domas/Developer/vilniusjs/prisma-test3/generated/prisma-client/

```
domas@Macbook-D2:~/Developer/vilniusjs/prisma-test2$ prisma deploy
```

```
? Set up a new Prisma server or deploy to an existing server?
```

Set up a new Prisma server for local development (based on docker-compose):

Use existing database

Connect to existing database

Create new database

Set up a local database using Docker

Or deploy to an existing Prisma server:

domas-bitvinskas/el-backo-heroku4

Production Prisma cluster

domas-bitvinskas/candour-honestive

Production Prisma cluster

riviera4media-dashboard/riviera4media

Production Prisma cluster

› Demo server

Hosted demo environment incl. database (requires login)

Use other server

Manually provide endpoint of a running Prisma server

```
domas@Macbook-D2:~/Developer/vilniusjs/prisma-test2$ prisma deploy
Deploying service `prisma-test2` to stage `dev` to server `prisma-eu1` 12.0s
```

Changes:

User (Type)

- + Created type `User`
- + Created field `id` of type `GraphQLID!`
- + Created field `name` of type `String!`
- + Created field `updatedAt` of type `DateTime!`
- + Created field `createdAt` of type `DateTime!`

Applying changes 15.5s

Your Prisma GraphQL database endpoint is live:

HTTP: <https://eu1.prisma.sh/domas-bitvinskas/prisma-test2/dev>
WS: <wss://eu1.prisma.sh/domas-bitvinskas/prisma-test2/dev>

```
const resolvers = {  
  Query: {  
    user: (_, args) => prisma.user({  
      id: args.id,  
    }),  
  },  
}
```

Prisma

supported databases

Postgres

MySQL

Mongo

Prisma clients

JavaScript

TypeScript

Flow

Go

Deployment



Heroku

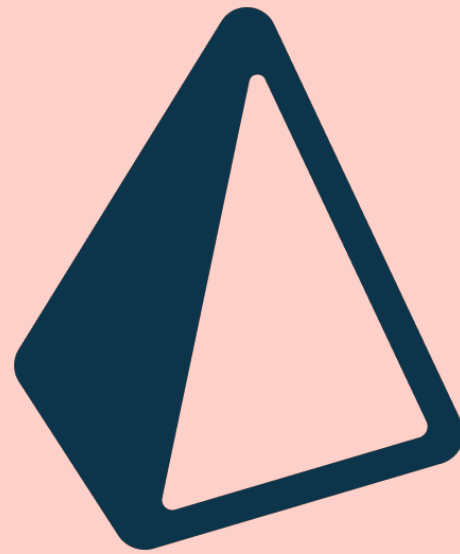


Prisma Cloud



...where you

deploy Docker



Short version

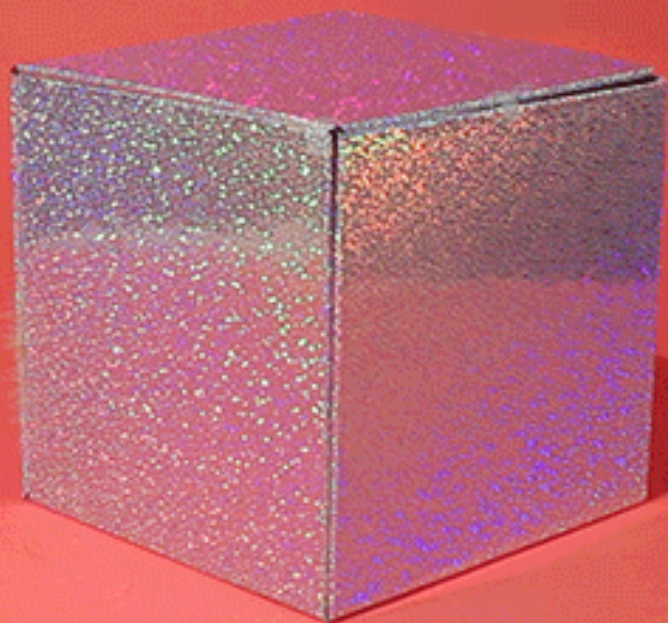
**Prisma makes it easy to
develop great GraphQL API's**

2019

- GraphQL adoption will continue grow
- Prisma will grow rapidly as a remedy for GraphQL mess



Thanks!



/Domas

domasbitvinskas.com