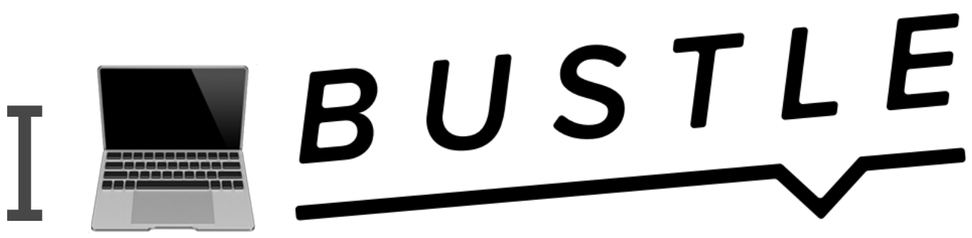
I 👋 Francis

reconbot


BUSTLE

Hey Francis, How did everyone at Bustle make the website so fast?

— People

We put everything in Redis.

— Francis

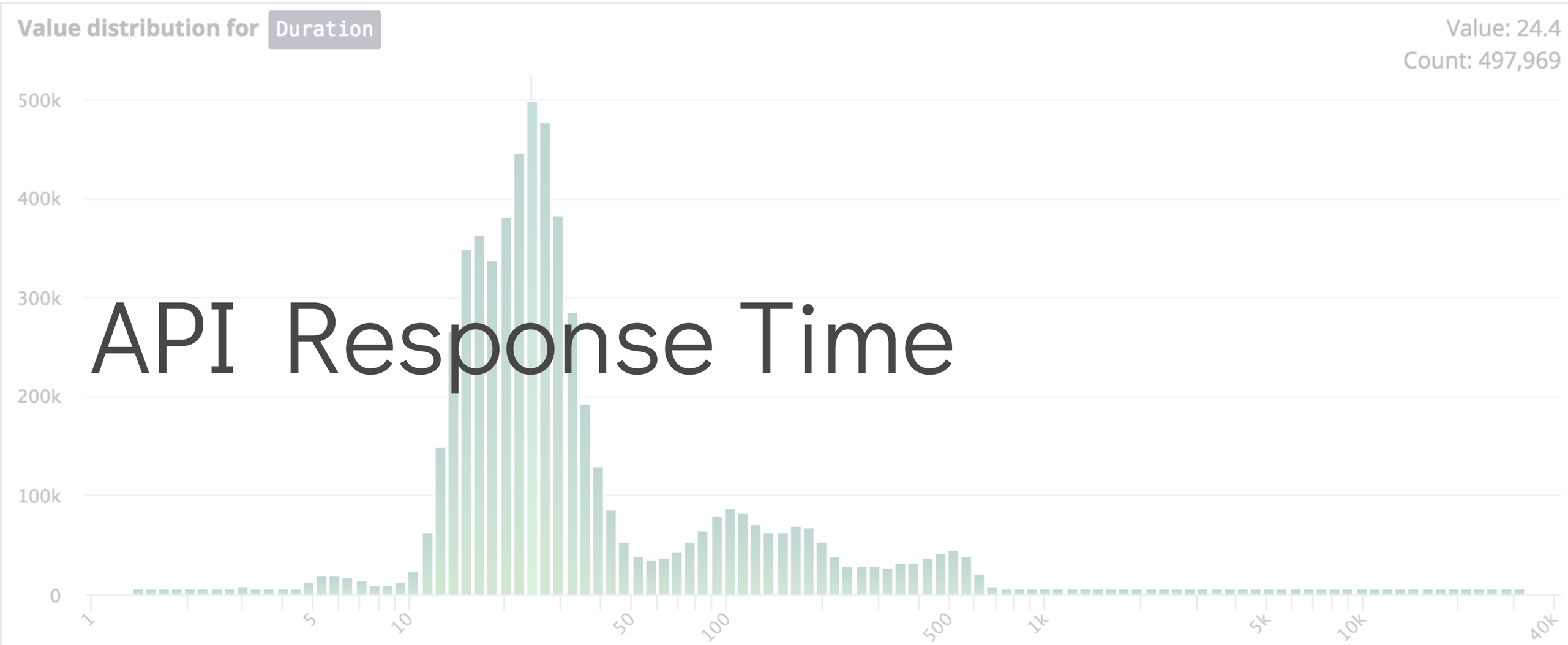
this didn't cut it so I made a talk

We live in Memory

a nice way of saying this is

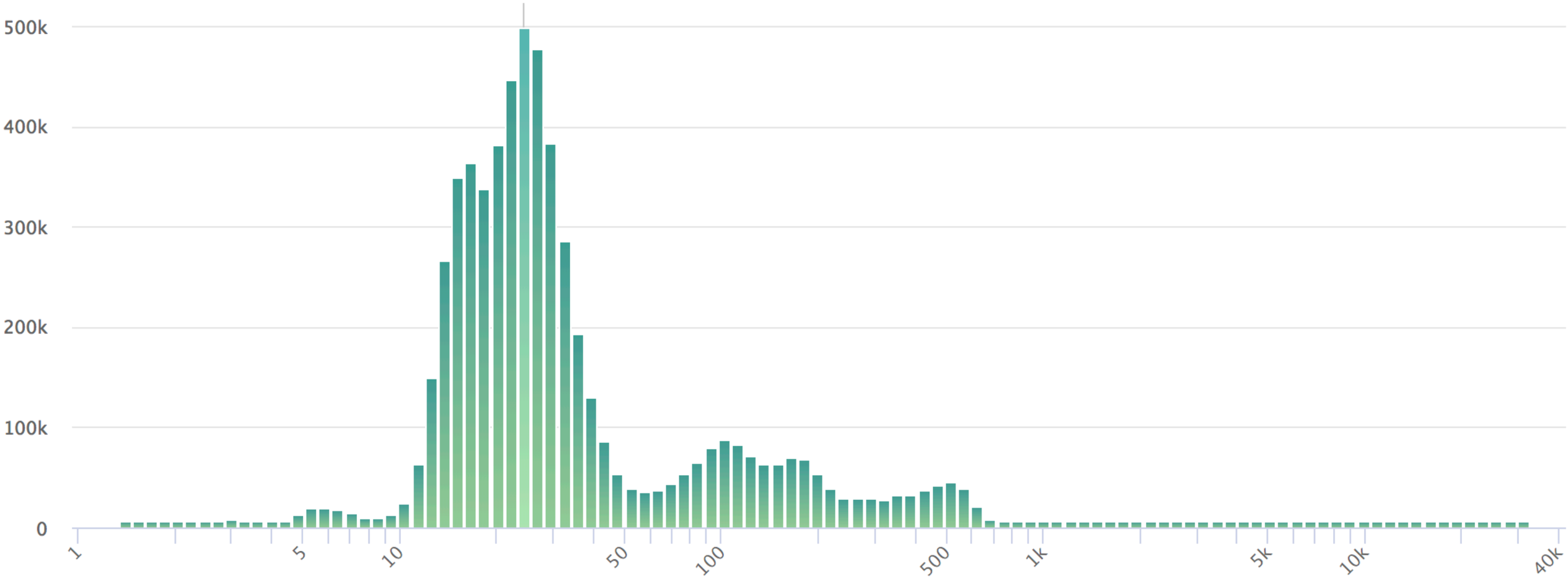
λ, Graphql, and Redis In Under 70ms

https://github.com/reconbot/we-live-in-memory



API Response Time

Value distribution for `Duration`



The Story

1. What is bustle doing
2. How is bustle doing it
3. How you can do it

BUSTLE DIGITAL GROUP

Welcome To The New Authority

BUSTLE DIGITAL GROUP

BDG is the largest reaching publisher for millennial women

Bustle itself has more visitors than washingtonpost, theguardian.com, stackexchange.com, imdb.com, even webmd.com (https://www.quantcast.com/top-sites/US) and we're now 5 sites

bustle acquires

bustle acquires **elite daily**
bustle acquires **zoe report**
bustle acquires **gawker**

Google Search I'm Feeling Lucky

Report inappropriate predictions

Platform Goals

- Best reader experience possible
- Best features for our writers and designers

Just like you 👍

Platform Strategies

- Fastest page load
- Reduce the cost of change

make it easy to build and test new products and easy to remove old ones, and keep them all really fast

What is AWS Lambda (λ)?

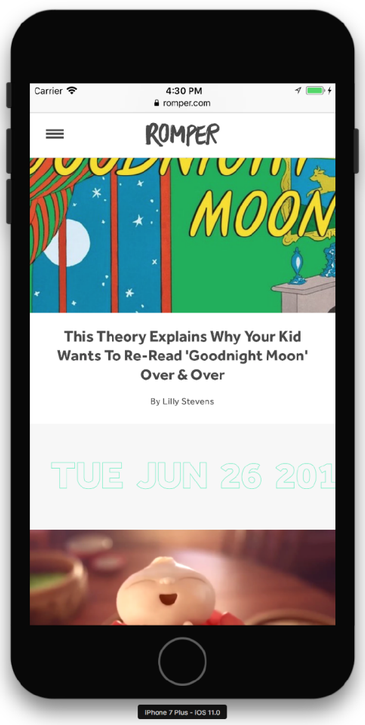
ls functions as a service, you get a remote api to call your function, in a consistent environment regardless of concurrency. 1 or 100k requests/s and they're guaranteed to have the same cpu and memory.

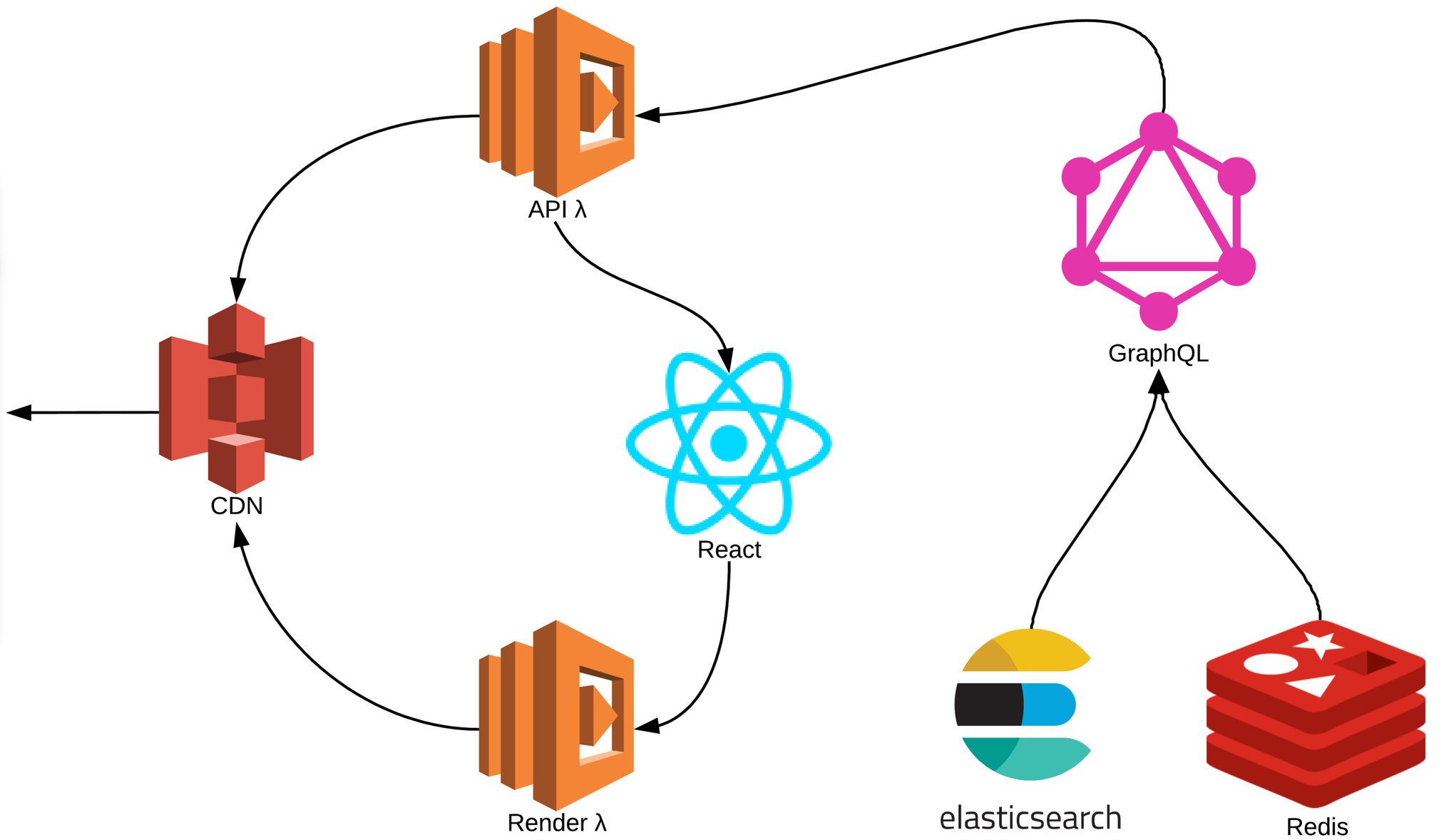
Why are we using lambda?

We lowered our monthly spend form 30k a month to 3k a month, and we can now handle unpredictable spikes in traffic that miss our CDN, we've grown 10x without having to worry about it

Ok Francis, but how does it work?

— Get to it already





CDN, react λ , graphql λ ,
Redis/ES

Layers

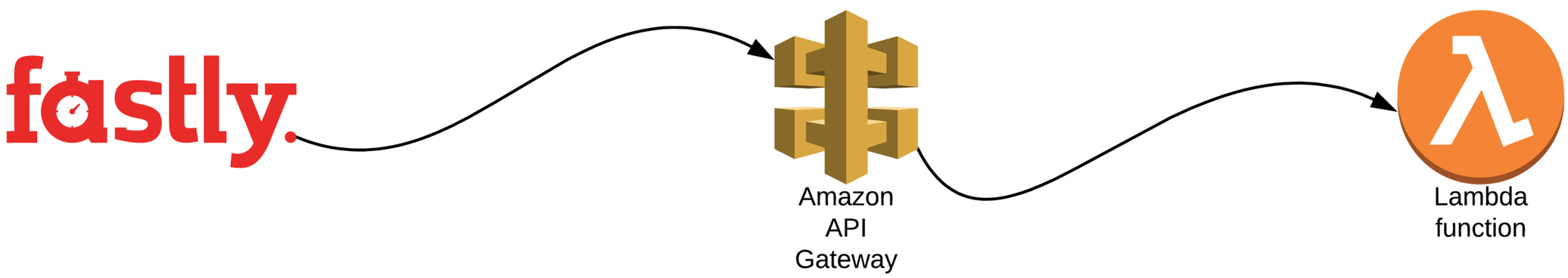
1. CDN
2. Rendering
3. API
4. Database

CDN

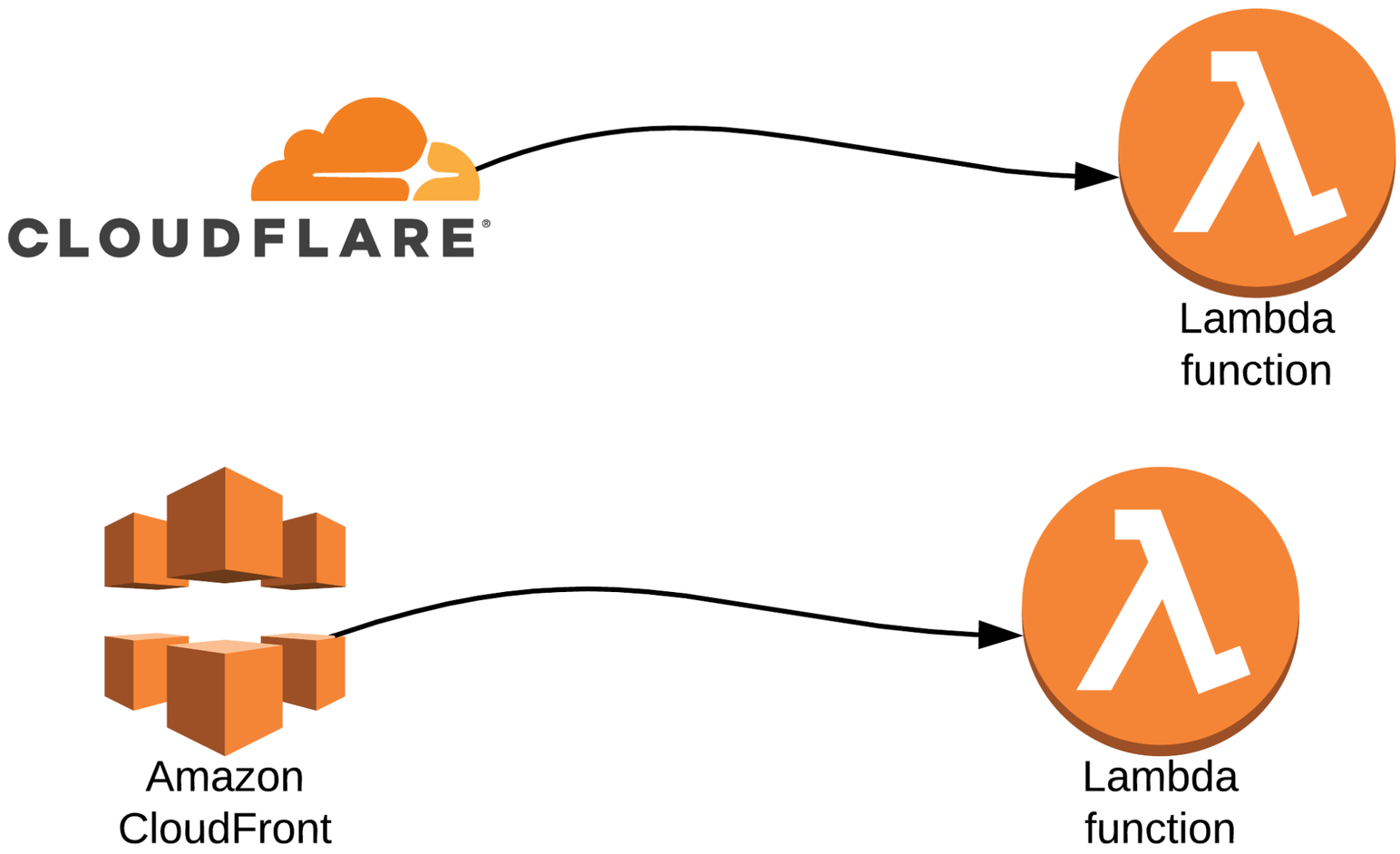
Takes an HTTP request and try to give a response out of cache, try really hard.

- API Gateway / Cloudfront
- Cloudflare
- Fastly

The CDN Should Execute my Functions



we have a special need, quick invalidation



I'm pretty sure we'll be able to drop API gateway and get a faster execution

Rendering

Take an HTTP request, fetch data, and render out { status, body, headers }

- Server Side Render a page
- Client Side Render a page
- Be smart about Loading Stylesheets and Components

Rendering

API

- Take a query and return some data
- Take some input and change state, then return some data
- Be strict about what types we return

API

API

we used to have a dozen microservices, each slightly different. We now have 1 way to do everything

Database

- Store the data safely
- Retrieve the data fast

Database

- Store the data safely
- Retrieve the data fast, in a consistent time

Redis is our Primary Data Store

Redis is an in-memory data structure server. It supports, hashes, lists, sets, sorted sets with range queries.

— Redis.io (kinda)

people usually use this as a disposable cache, cache it here and throw it away (eg sessions)

Hey Francis,

Isn't that really dangerous?

— 50/50 chance you'll say this

if you know redis...

No.

— 100% chance I'll say this

I don't believe you.

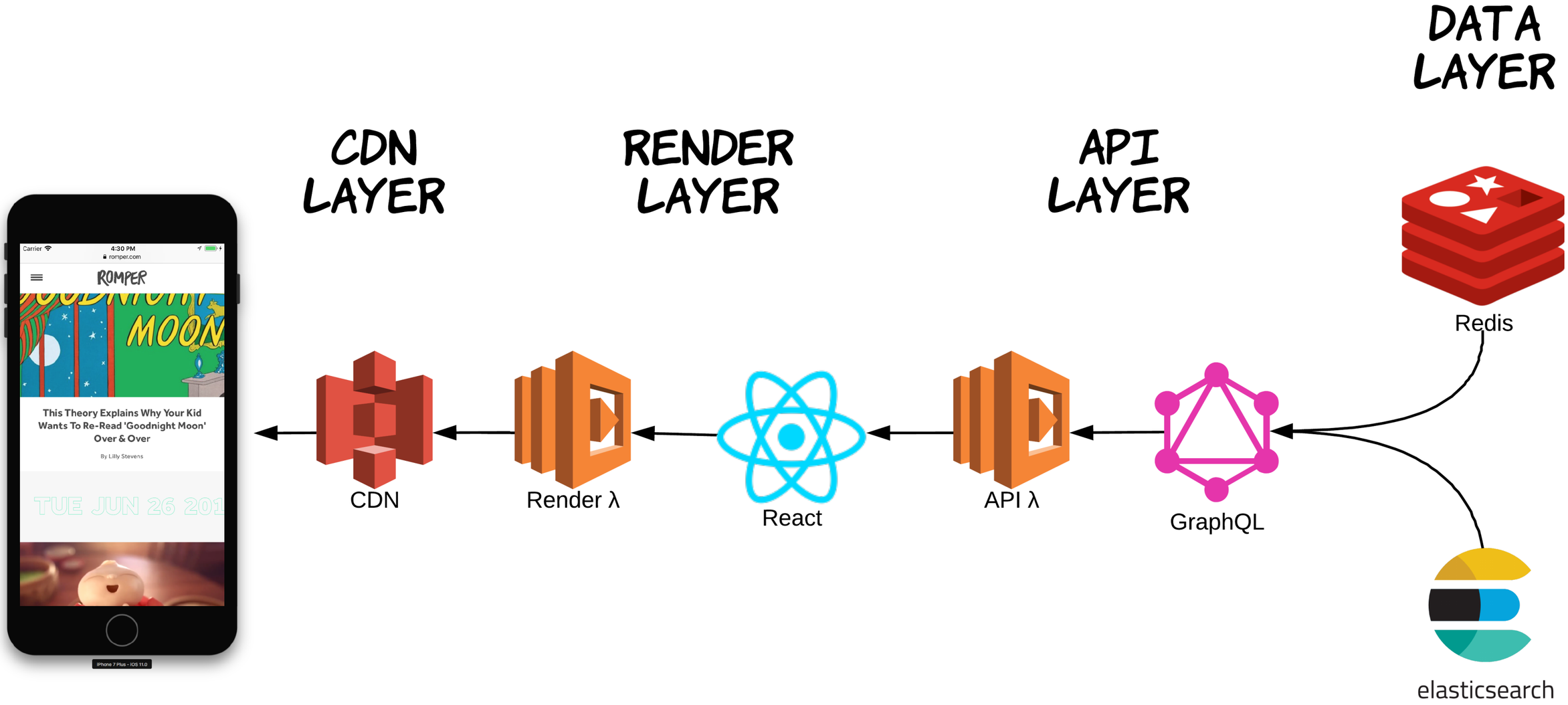
– 99% chance you're thinking this

Redis Persistance

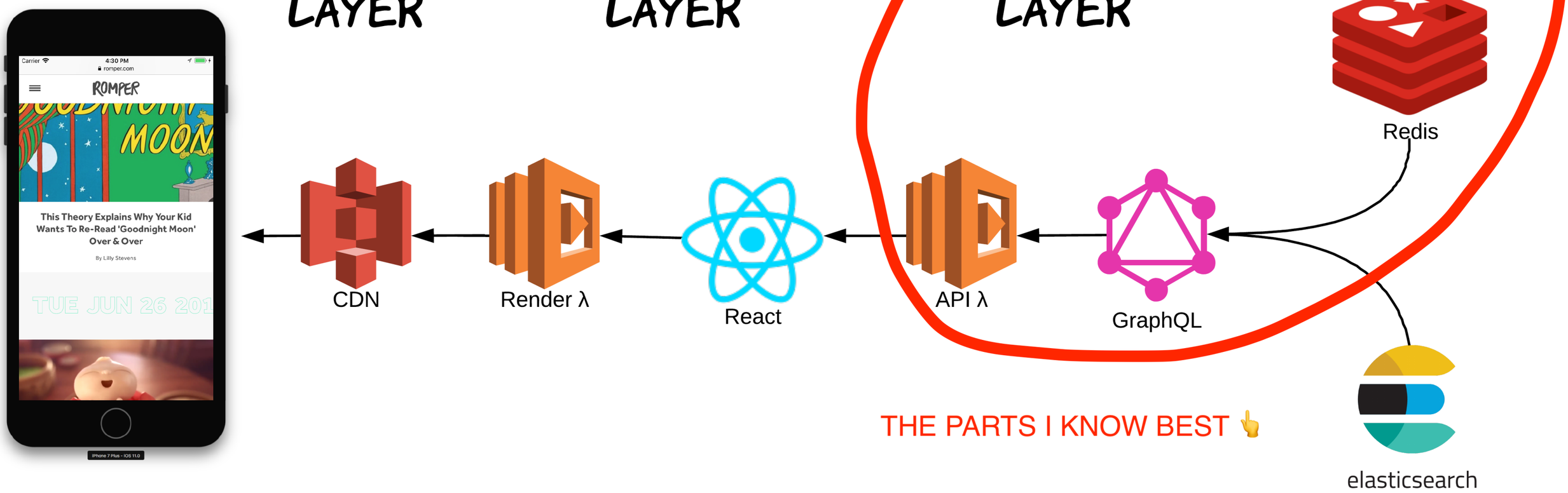
- 1s fsync of Append Only File (AOF)
- 1 hour snapshot of the Redis Database File (RDB) backed up to S3
- Read replicas ready to take over really fast
- Perfect for our read heavy load

This could be any database

Speed is important to us, we have ads to load



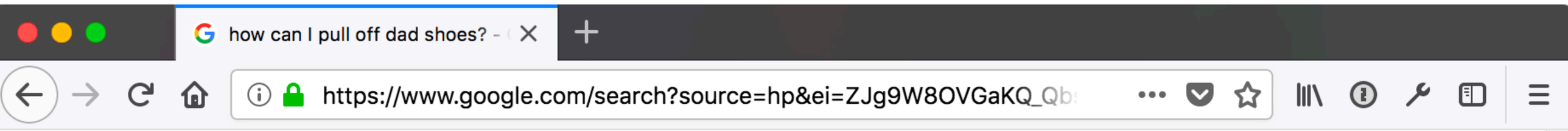
describe everything, note that elasticsearch is "You Know, for Search"



lets start with an example

how can I pull off dad shoes?

Google Search I'm Feeling Lucky



I Wore 'Dad Shoes' For A Week & They Were SO Much Cooler Than I ...
https://www.bustle.com/.../i-wore-dad-shoes-for-a-week-they-were-so-much-cooler-th...
Mar 15, 2018 - My two reference points for the perfect **dad shoes** were a stylish Ryan ... I ended up **pulling** a blazer from the back **of** my closet and wearing a ...

'Dad' trainers are everywhere, but would you wear them?
https://www.harpersbazaar.com/uk/fashion/.../chunky-dad-grandpa-trainers-trend/
Feb 7, 2018 - Spongy gym **shoes** are fast becoming the fashion trainer **of** choice. Here's how to **pull** them **off**.

8 "Dad Style" Moves That Are Actually Really Cool | GQ
https://www.gq.com/story/dad-style-is-actually-cool
Jun 15, 2016 - These days, add "**dad**" to anything—**dad** jeans, **dad** bod, **dad** style—and you're implying that it's not hip, hopelessly **out of** touch. Then again ...

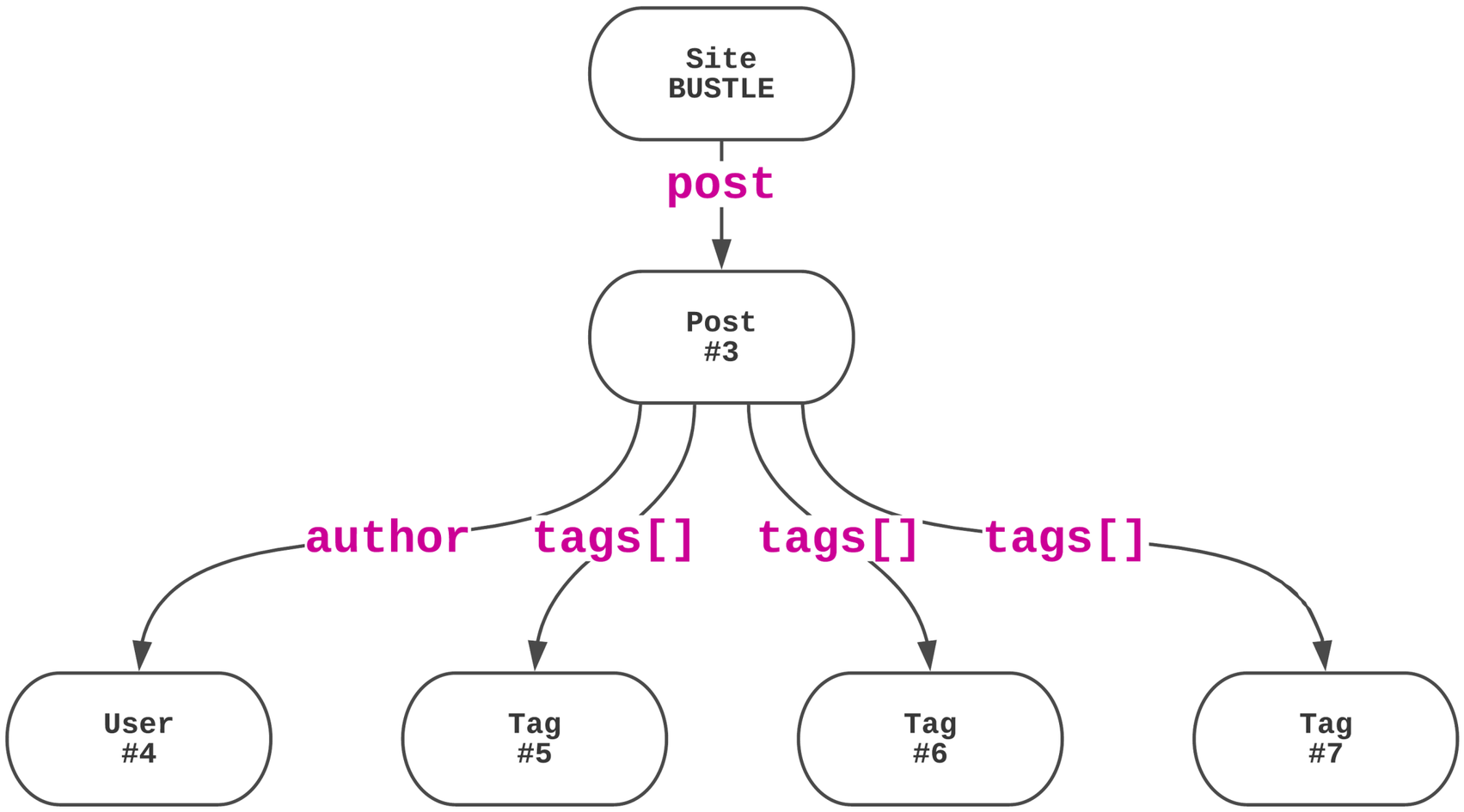
I Wore 'Dad Shoes' For A Week & They Were SO Much Cooler Than I Thought They'd Be

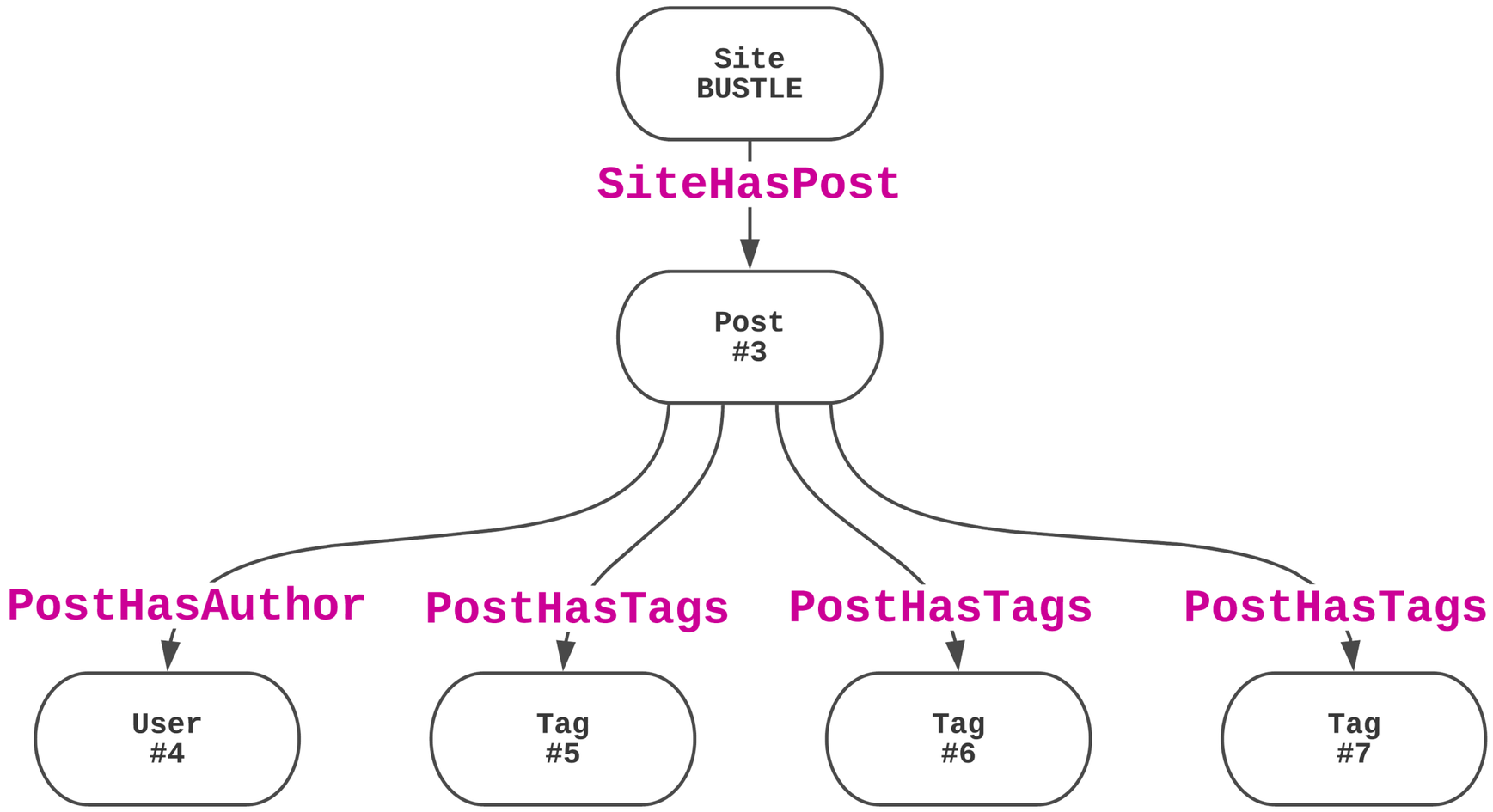
By DALE ARDEN CHONG | Mar 15 2018

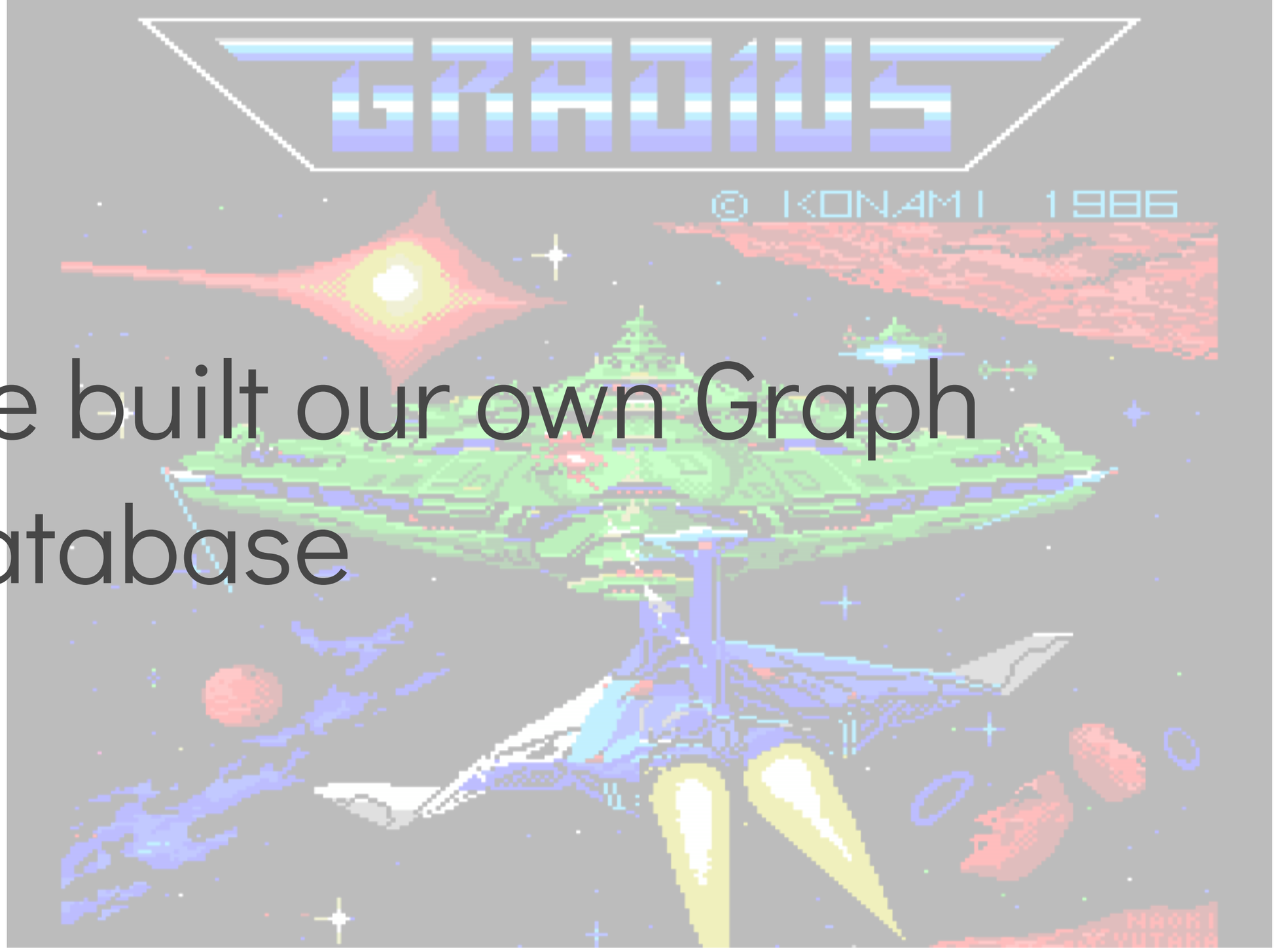
```
{
  "site": {
    "name": "BUSTLE",
    "__typename": "Site",
    "post": {
      "id": "3",
      "__typename": "Post",
      "title": "I Wore 'Dad Shoes' For A Week & They Were SO Much Cooler Than I Thought",
      "path": "/p/i-wore-dad-shoes-for-a-week-they-were-so-much-cooler-3",
      "body": "99% of the JSON you'd be looking at, HI JSCONF!",
      "author": {
        "id": "4",
        "__typename": "User",
        "name": "Dale Arden Chong"
      },
      "tags": [
        { "id": "5", "__typename": "Tag", "name": "homepage" },
        { "id": "6", "__typename": "Tag", "name": "fashion" },
        { "id": "7", "__typename": "Tag", "name": "freelancer" }
      ]
    }
  }
}
```

```
query postByPath {
  site(name: BUSTLE) {
    name
    post(path: "/p/i-wore-dad-shoes-for-a-week-they-were-so-much-cooler-3") {
      id
      __typename
      title
      path
      body
      author {
        id
        __typename
        name
      }
      tags {
        id
        __typename
        name
      }
    }
  }
}
```

```
{
  "site": {
    "name": "BUSTLE",
    "__typename": "Site",
    "post": {
      "id": "3",
      "__typename": "Post",
      "author": {
        "id": "4",
        "__typename": "User"
      },
      "tags": [
        { "id": "5", "__typename": "Tag" },
        { "id": "6", "__typename": "Tag" },
        { "id": "7", "__typename": "Tag" }
      ]
    }
  }
}
```


Site
BUSTLE
post
Post
#3
author
tags[]
tags[]
tags[]
User
#4
Tag
#5
Tag
#6
Tag
#7


Site
BUSTLE
SiteHasPost
Post
#3
PostHasAuthor
PostHasTags
PostHasTags
PostHasTags
User
#4
Tag
#5
Tag
#6
Tag
#7



We built our own Graph Database

what if we had a database that let us save and access data like this?

^ we called it gradius

Oh you mean, like Neo4j right?

— 99% of you

Sure, but it's faster and doesn't do any of the same things.

— Francis

Trains aren't slow they have one speed, people get on & off, the people are slow. Databases aren't slow they have one speed, data goes in and out, the queries are slow.

— Ikai (a DBA, who I guess never took the subway)

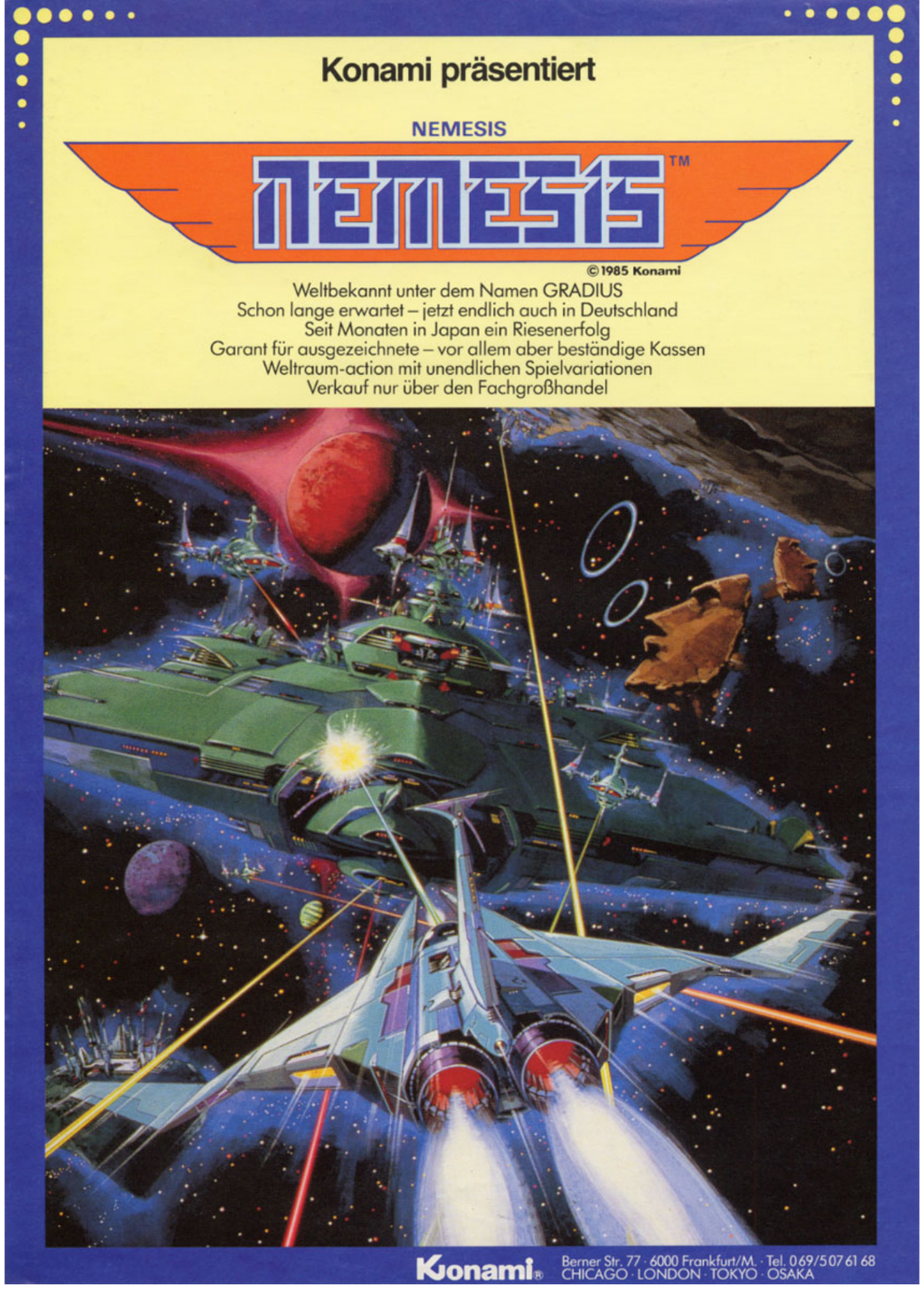
yes but also you can build a faster train

It's all about tradeoffs

Bustle Traded Query Flexibility for Speed

and that's why we replicate from graphDB to BigQuery and elasticsearch

Bustle loves to share & fund Open Source



install

```
> npm i nemesis-db
```

- Really Fast Reads (0-1ms)
- Node and Edge Schemas with Types and Interfaces
- Weighted and Labeled Edges with Scanning
- Compression
- Still porting, not done yet!

The US arcade port of Gradius

Lets Make a GraphQL Server

1. The tools
2. The schema
3. The resolvers
4. The loaders

Tools

```
{
  "dependencies": {
    "apollo-server": "^2.0.5",
    "apollo-server-lambda": "^2.0.4",
    "graphql": "^0.13.2",
    "nemesis-db": "^1.3.0-0"
  }
}
```

apollo-server

```
const { ApolloServer } = require('apollo-server')
const { readFileSync } = require('fs')
const resolvers = require('./resolvers')
const typeDefs = readFileSync('./lib/types.graphql', 'UTF8')

const server = new ApolloServer({ typeDefs, resolvers })

server.listen().then(({ url }) => {
  console.log(`🚀 🌘 Server ready at ${url}`)
});
```

good for developing in

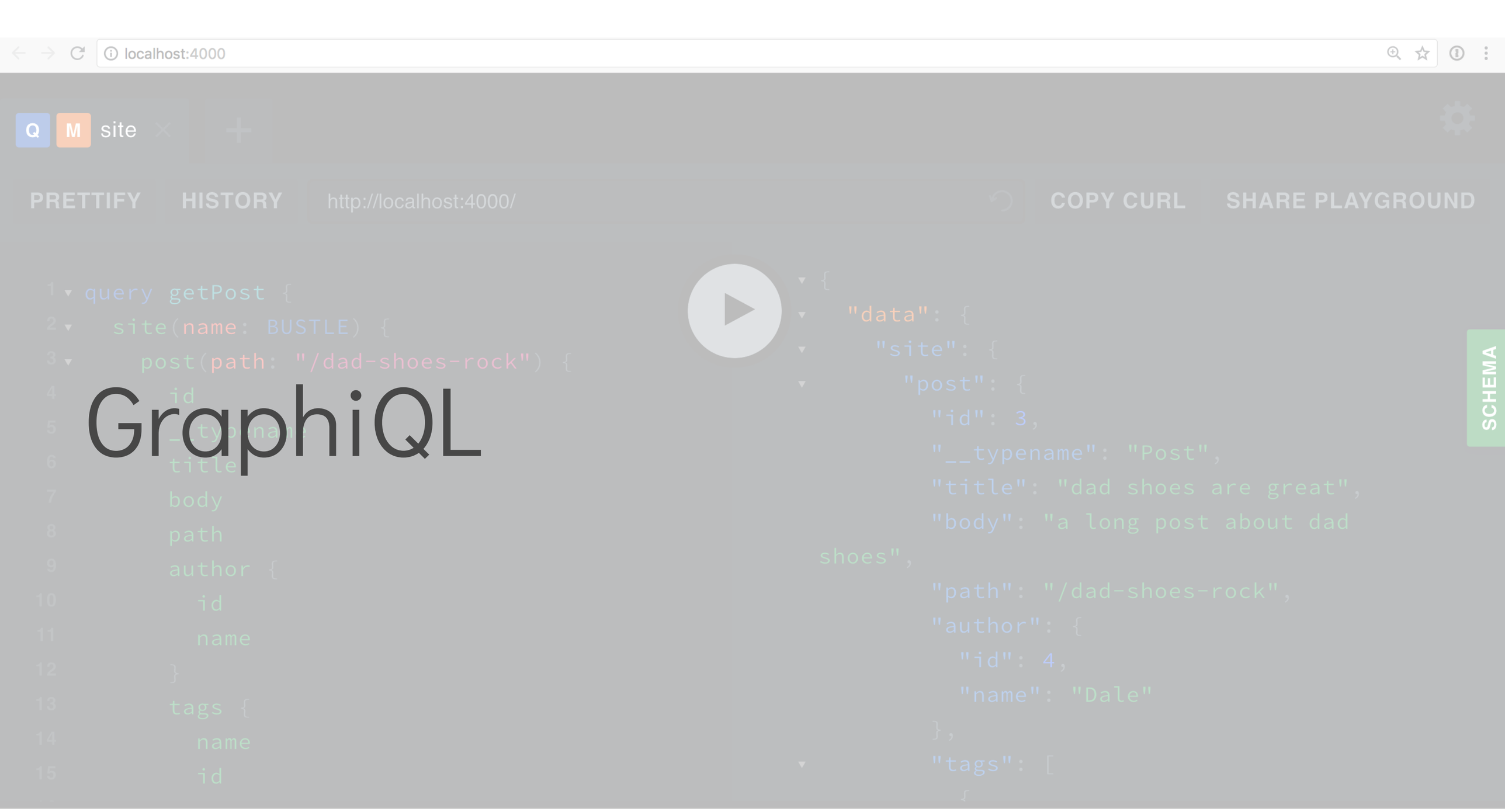
apollo-server-lambda

```
const { ApolloServer } = require('apollo-server-lambda')
const { readFileSync } = require('fs')
const resolvers = require('./resolvers')
const typeDefs = readFileSync('./lib/types.graphql', 'UTF8')

const server = new ApolloServer({ typeDefs, resolvers })

exports.handler = server.createHandler() // 🚀 🏀
```

good for production


localhost:4000
Q
M
site
PRETTIFY
HISTORY
http://localhost:4000/
COPY CURL
SHARE PLAYGROUND
query getPost {
site(name: BUSTLE) {
post(path: "/dad-shoes-rock") {
id
__typename
title
body
path
author {
id
name
}
tags {
name
id
{
"data": {
"site": {
"post": {
"id": 3,
"__typename": "Post",
"title": "dad shoes are great",
"body": "a long post about dad
shoes",
"path": "/dad-shoes-rock",
"author": {
"id": 4,
"name": "Dale"
},
"tags": [
SCHEMA

GraphiQL

localhost:4000

Q M site

PRETTIFY HISTORY http://localhost:4000/ COPY CURL SHARE PLAYGROUND

```
query getPost {
  site(name: BUSTLE) {
    post(path: "/dad-shoes-rock") {
      id
      __typename
      title
      body
      path
      author {
      id
      name
    }
    tags {
      name
      id
```

```
{
  "data": {
    "site": {
      "post": {
        "id": 3,
        "__typename": "Post",
        "title": "dad shoes are great",
        "body": "a long post about dad
shoes",
        "path": "/dad-shoes-rock",
        "author": {
          "id": 4,
          "name": "Dale"
        },
        "tags": [
          {
```

SCHEMA

Search the schema ...

QUERIES

site(...): Site!

MUTATIONS

resetDevData: String!

site(
  name: SITE_NAME!
): Site!

TYPE DETAILS

type Site {
  id: Int!
  name: String!
  post(...): Post
}

ARGUMENTS

name: SITE_NAME!

post(
  path: String!
): Post

TYPE DETAILS

type Post {
  id: Int!
  path: String!
  title: String!
  body: String!
  author: User!
  tags: [Tag!]!
}

ARGUMENTS

path: String!

author: User!

TYPE DETAILS

type User {
  id: Int!
  name: String!
}

The Schema

- Defines what the data looks like
- Doesn't care how it behaves

The Schema

```
type Query {
  site(name: SITE_NAME!): Site!
}

enum SITE_NAME {
  BUSTLE
  ROMPER
}

type Site {
  id: Int!
  name: String!
  post(path: String!): Post
}
```

id and name are DATA, post is a lookup

The Schema

```
type Post {
  id: Int!
  path: String!
  title: String!
  body: String!
  author: User!
  tags: [Tag!]!
}
```

id, path, title, body are all data,
author and tags are a lookup

The Schema

```
type User {
  id: Int!
  name: String!
}

type Tag {
  id: Int!
  name: String!
}
```

The Resolvers

- Only needed for lookups
- Return a full object

used to return only the necessary fields, with our db it was not worth it, not even a little

The Resolvers

```
module.exports = {
  Query: {
    site: (root, { name }) => { /*...*/ }
  },
  Site: {
    post: async (site, { path }) => { /*...*/ }
  },
  Post: {
    author: async post => { /*...*/ },
    tags: async post => { /*...*/ }
  }
}
```

The Site Resolver

```
{
  Site: {
    post: async (site, { path }) => {
      const edge = await graph.findLabeledEdge({
        subject: site.id,
        predicate: 'path',
        label: path
      })
      if (!edge) { return null }
      const { object: postId } = edge
      return graph.findNode(postId)
    }
  }
}
```

The Post Resolver

```
{
    Post: {
    author: async ({ id }) => {
      const [{ object: userId }] = await graph.findEdges({
        subject: id,
        predicate: 'PostHasAuthor'
      })
      return graph.findNode(userId)
    },
    tags: async ({ id }) => {
      const edges = await graph.findEdges({
        subject: id,
        predicate: 'PostHasTags'
      })
      return Promise.all(edges.map(({ object }) => graph.findNode(object)))
    }
  }
}
```

The N+1 problem

we solve this with command batching and redis pipelining

Command Batching

dataloader - lets you batch and dedupe all queries in a single tick

```
const loader = new DataLoader(ids => db.getBunchOfIds(ids))
await Promise.all([
  loader.load(1),
  loader.load(2),
  loader.load(2),
  loader.load(3)
])
// db.getBunchOfIds([1,2,3])
```

Command Batching

`redis-loader` - lets you pipeline all queries in a single tick.

```
const posts = await Promise.all([
  graph.findNode(1),
  graph.findNode(2),
  graph.findNode(2),
  graph.findNode(3)
])
```

```
// redis
1535039766.77294 [2 127.0.0.1:55031] "hmget" "node:1" "c" "data"
1535039766.77397 [2 127.0.0.1:55031] "hmget" "node:2" "c" "data"
1535039766.77554 [2 127.0.0.1:55031] "hmget" "node:2" "c" "data"
1535039766.77725 [2 127.0.0.1:55031] "hmget" "node:3" "c" "data"
// 434 µs

// redis-loader
1535039810.951880 [13 127.0.0.1:55036] "multi"
1535039810.951898 [13 127.0.0.1:55036] "hmget" "node:1" "c" "data"
1535039810.951921 [13 127.0.0.1:55036] "hmget" "node:2" "c" "data"
1535039810.951940 [13 127.0.0.1:55036] "hmget" "node:3" "c" "data"
1535039810.951956 [13 127.0.0.1:55036] "exec"
// 76 µs
```

super unscientific test on my latptop

Lambda Tooling

- `npm/sammie` - "Serverless Application Model Made Infinitely Easier"
- Architect - https://arc.codes
- `npm/shep`

You're ready to rock and roll 🚀 

Nemesis isn't done yet, you can help

Open source

- `bluestream` Streams for Async functions
- `mobiledoc-kit` A toolkit for building WYSIWYG editors with Mobiledoc
- `nemesis-db` A fast redis graph database
- `redis-loader` An ioredis-like object that batches commands via dataloader
- `sammie` Serverless Application Model Made Infinitely Easier
- `streaming-iterables` Replace your streams with async iterators

Thank you 🙏

- I'm Francis / reconbot
- For slides and a short story
  `https://github.com/reconbot/we-live-in-memory`
- For a great place to work
  `https://bustle.company`