



Scaling GraphQL subscriptions

Artjom Kurapov
Principal Software Engineer,
Engineering Platform

Artjom Kurapov



I develop in NodeJS / Go / PHP



Work on GraphQL for last 3 years



I focus on complexity, performance and scalability



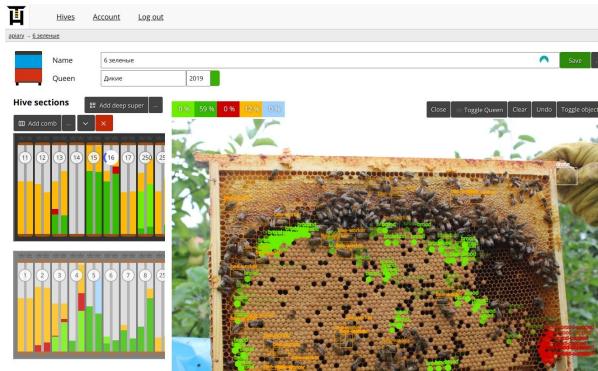
Open source maintainer - github.com/tot-ra



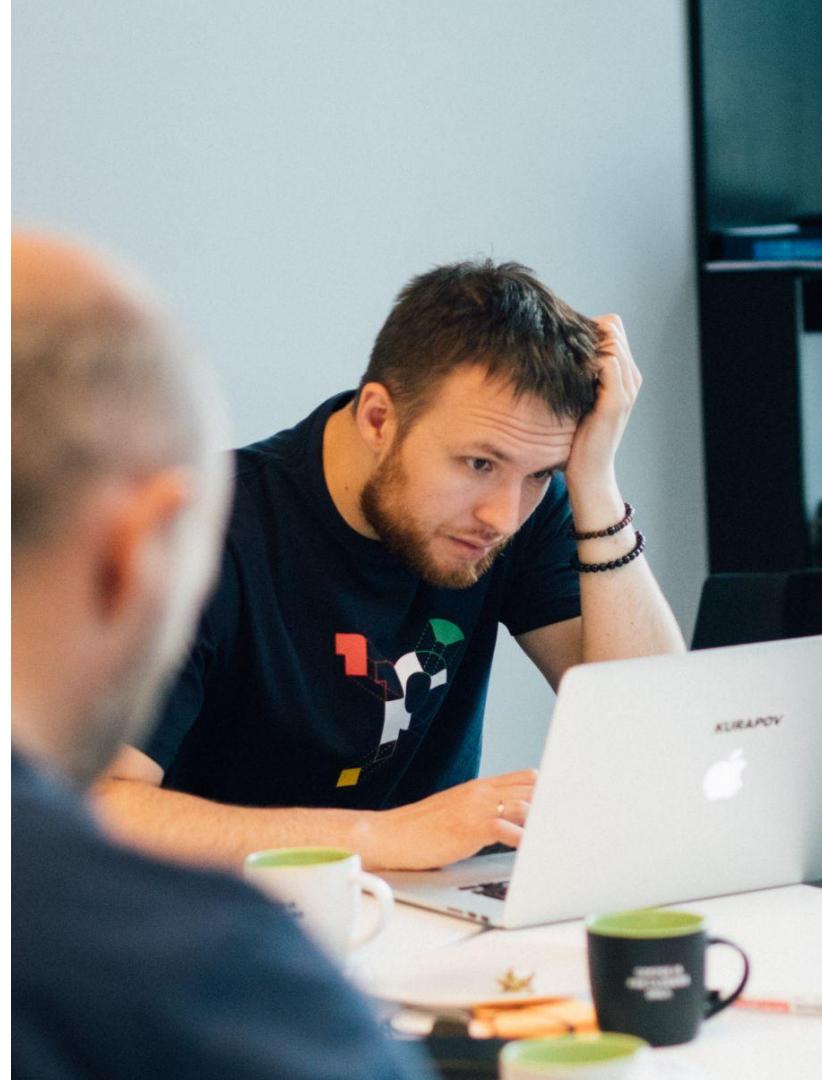
Interested in image recognition and



Beekeeping - github.com/Gratheon



pipedrive



Agenda

Questions

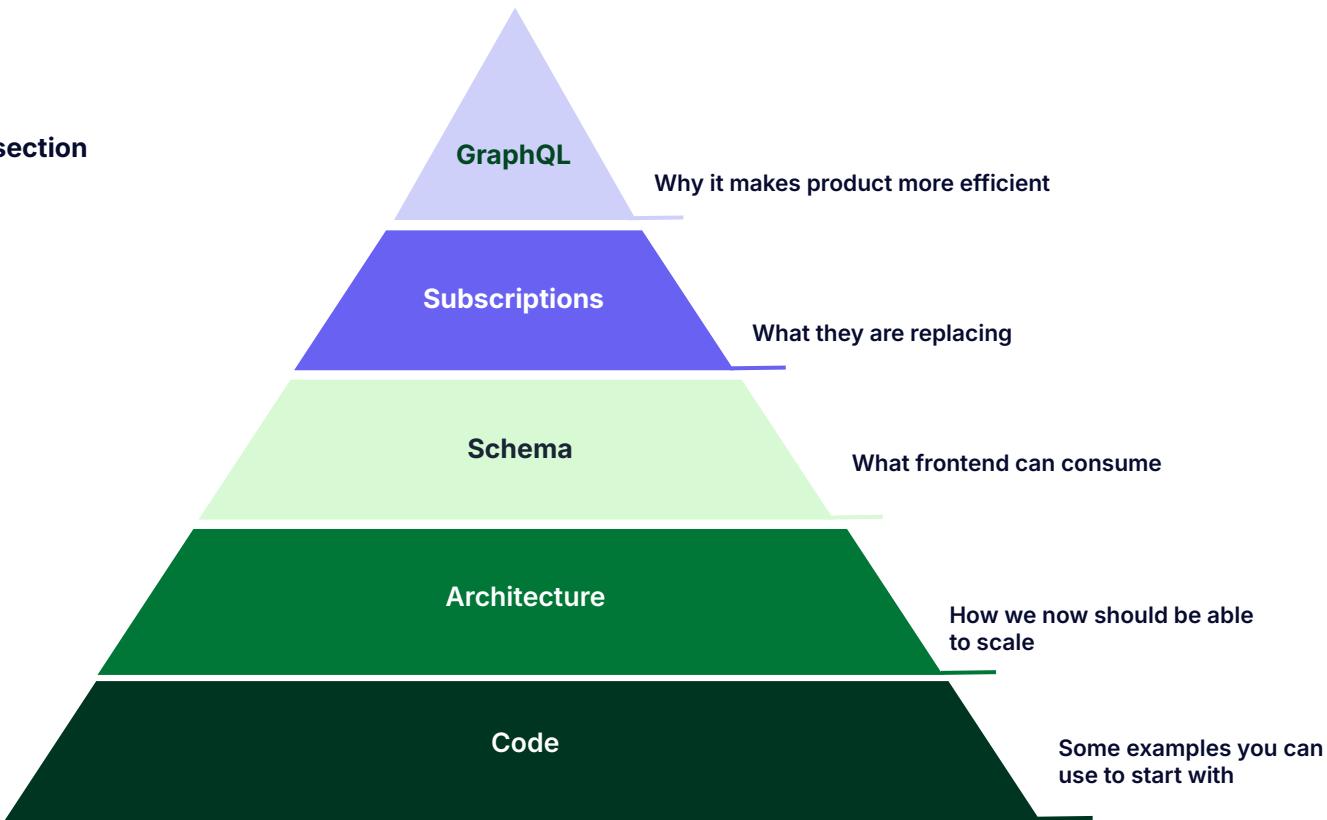
Ask at the end of every section

Time

ETA 50 min
~70 slides

Complexity

Expect it to grow with
every section





pipedrive

600

About Pipedrive



>100k
clients

400+
engineers

10
offices

30+
teams

440k
rpm at peak

730+
microservices

5
hosting regions

200+
libraries



The image displays the Pipedrive CRM system across two devices: a laptop and a smartphone.

Laptop Screen (Left):

- Header:** Deals, Search Pipedrive, Pipeline (dropdown), Everyone (dropdown).
- Left Sidebar:** Includes icons for deals, contacts, pipelines, tasks, calendar, messages, reports, and more. A deal titled "Silicon Links Inc deal" is highlighted.
- Deal List:** Shows various deals categorized by stage:
 - Qualified:** 10 000 \$ - 4 deals (e.g., Umbrella Corp deal, JMVD Inc deal, Ownerate LLP deal, Silicon Links Inc deal).
 - Contact Made:** 5 350 \$ - 3 deals (e.g., Principalspace Inc deal, Blue Marble LLP deal, ABC Inc deal).
 - Demo Scheduled:** 3 100 \$ - 2 deals (e.g., Moveer Limited deal, Wolfs Corp deal).
 - Proposal Made:** 2 700 \$ - 1 deal (e.g., Omnicorp deal).
 - Negotiations Started:** 4 200 \$ - 2 deals (e.g., Big Wheels Inc deal, Mindbend LLP deal).
- Bottom Panel:** A modal window titled "Pitch meeting preparation" shows a timeline from "Tomorrow" to "First pitch" (also tomorrow) for Phyllis Yang. It includes a "Schedule an activity" button.

Smartphone Screen (Right):

- Header:** Deal, Edit.
- Deal Details:** Big Wheels Inc deal, Phyllis Yang, Big Wheels Inc.
- Timeline:** Shows a green timeline from "Idea" to "Contact Made" to "Needs discovered".
- Value:** \$325,000.
- Status:** WON (green bar) and LOST (red bar).
- Sections:** Planned (Present the mood board, Allan Quatermain), Past (Hey @Esther Howard could you ple..., Re: Offer LM45945 de...), and Call Amanda.
- Bottom Navigation:** Deals, Activities, Contacts, Help.

Data synchronization / async event delivery

The image shows two side-by-side screenshots of the Pipedrive CRM interface, specifically the Deals module.

Left Screenshot:

- Header:** "Deals" with a search bar and a "Deal" button.
- Left Sidebar:** Includes icons for deals, prospects, contacts, and other modules, with a red notification badge showing "1".
- Prospect Card:** Shows "Prospect" with "10 000 € · 1 deal" and "Contact established" with "0 €".
- Deal Card:** Shows "Wise deal" with "10 000 €" and a yellow warning icon.

Right Screenshot:

- Header:** "Deals" with a search bar and a "Deal" button, and a total value of "10 000 €" displayed.
- Left Sidebar:** Includes icons for deals, prospects, contacts, and other modules, with a red notification badge showing "1".
- Prospect Card:** Shows "Prospect" with "10 000 € · 1 deal" and "Contact established" with "0 €".
- Deal Card:** Shows "Wise deal" with "10 000 €" and a yellow warning icon.

The two screenshots illustrate how data is synchronized across different instances of the CRM, showing identical deal and prospect information in both panels.

Why your business needs this?

Intuitive UX → **Simplicity** → Growth 

Interactivity → **Efficiency** → Retention 

Data consistency → **Trust** → Retention 

How? Pusher

The screenshot shows the Pusher website homepage. At the top, there's a navigation bar with the Pusher logo, a "Products" dropdown, a "Developers" dropdown, "User stories", "Blog", "Pricing" dropdown, and "Sign in" and "Sign up" buttons. The main content area features a hero section with the text "Powering realtime experiences for mobile and web" and a description of bi-directional hosted APIs. Below this, there's a "Get started today" call-to-action and two buttons: "Use Cases" and "Get your free account". To the right, there's a large sidebar titled "Pubsub" and "Notifications" with tabs for "Publish" and "Subscribe". Under "Publish", there are examples for PHP, Node.js, Ruby, ASP, Java, Python, and Go. The Node.js example is shown as:

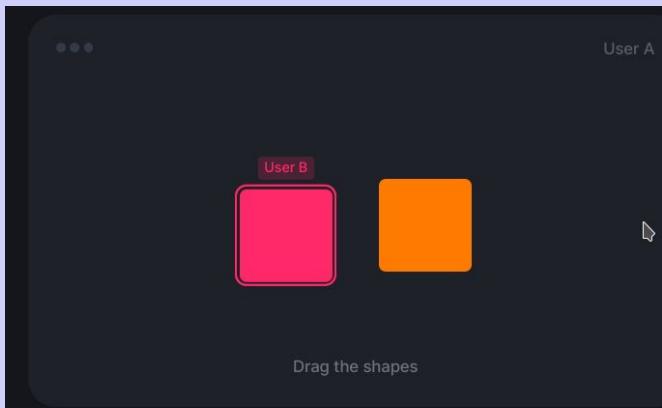
```
1 pusher.trigger('my-channel', 'my-event', {  
2   "message": "hello world"  
3 });
```

Under "Subscribe", there are examples for JS, Android, iOS (Swift), and iOS (Obj-C). The JS example is shown as:

```
2 channel.bind('my-event', function(data) {  
3   alert('Received my-event with message: ' + dat  
4 });
```

At the bottom of the sidebar is a "Learn more" button.

How? Liveblocks



React Redux Zustand

```
1 import { LiveList, LiveObject } from "@liveblocks/client";
2 import {
3   useStorage,
4   useMyPresence,
5   useOthers,
6 } from "./liveblocks.config";
7
8 // Pass these to RoomProvider
9 const initialPresence = { selectedShape: null };
10 const initialStorage = {
11   shapes: new LiveList([
12     new LiveObject({ x: -60, y: 0, bg: "#FF2868" }),
13     new LiveObject({ x: 60, y: 0, bg: "#FF7A00" }),
14   ]),
15 };
16
17 // Shape data updating in real-time
18 const shapes = useStorage(root => root.shapes);
19 // => [{ x: -60, y: 0, bg: "#FF2868" },
20 //       { x: 60, y: 0, bg: "#FF7A00" }]
21
22 // Share your presence as easily as using a useState
23 const [myPresence, setMyPresence] = useMyPresence();
24 setMyPresence({ selectedShape: "shapeId" });
25
26 // Get others people presence (cursors,
27 // avatar, name, etc) with a single line of code
28 const others = useOthers()
```

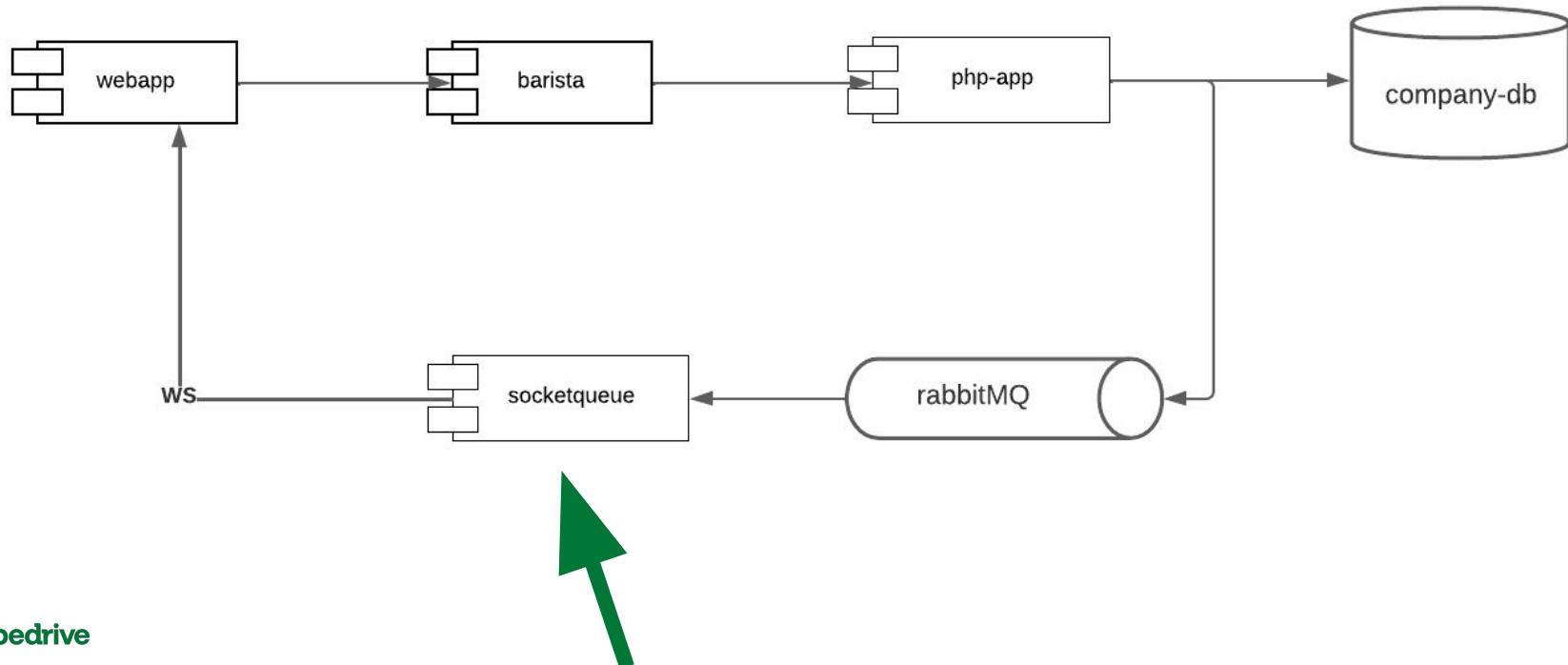
How? Firebase

The screenshot shows the Firebase documentation website. The top navigation bar includes links for Products, Solutions, Pricing, Docs (selected), Community, and Support, along with a search bar. On the left, a sidebar menu lists various services: Authentication, Realtime Database, Cloud Firestore (expanded), Introduction, Get started, Understand Cloud Firestore, Add and manage data, Read data (expanded), Get data, and Get real-time updates (highlighted with a blue background). Below these are sections for Perform simple and compound queries, Order and limit data, Count documents with aggregation queries, and a partially visible section starting with 'Project setup with'. The main content area displays code examples for different SDK versions. At the top of the content area, tabs for 'Web version 9 (modular)', 'Web version 8 (namespaced)', Swift, Objective-C, Java Android, Kotlin+KTX Android, Dart Flutter, Go, and More are shown. A callout for 'Web version 9 (modular)' encourages upgrading from version 8. The code example for 'Web version 9 (modular)' is as follows:

```
import { collection, query, where, onSnapshot } from "firebase/firestore";  
  
const q = query(collection(db, "cities"), where("state", "==", "CA"));  
const unsubscribe = onSnapshot(q, (snapshot) => {  
    snapshot.docChanges().forEach((change) => {  
        if (change.type === "added") {  
            console.log("New city: ", change.doc.data());  
        }  
        if (change.type === "modified") {  
            console.log("Modified city: ", change.doc.data());  
        }  
        if (change.type === "removed") {  
            console.log("Removed city: ", change.doc.data());  
        }  
    });  
});  
});
```

A mouse cursor is hovering over the 'listen_diffs.js' file name at the bottom right of the code block.

Pipedrive (oversimplified) architecture



Why GraphQL?



REST problems / schema documentation

P DEVELOPERS Docs & guides Explore Tools Marketplace go to pipedrive.com Log in Sign up

Currencies
Deals
DealFields
Files
Filters
GET Get all filters
GET Get all filter helpers
GET Get one filter
POST Add a new filter
PUT Update filter
DELETE Delete multiple filters in bulk
DELETE Delete a filter
Goals
ItemSearch
Leads
LeadLabels
LeadSources
LegacyTeams
Mailbox

Get one filter ↗
GET /v1/filters/{id}

Returns data about a specific filter. Note that this also returns the condition lines of the filter.

Path parameters

id INTEGER REQUIRED	The ID of the filter
---	----------------------

Response

200 OK

```
{  
    "success": true  
    - "data": (...)  
    - "id": 1  
    - "name": "All open deals"  
    - "active_flag": true  
    - "type": "deals"  
    - "temporary_flag": null  
    - "user_id": 927097  
    - "add_time": "2019-10-15 11:01:53"  
    - "update_time": "2019-10-15 11:01:53"  
    - "visible_to": 7  
    - "custom_view_id": 1  
}
```

Expand all Copy code

xplore Tools Marketplace go to pipedrive.com Log in Sign up

Add a new filter ↗
POST /v1/filters

Adds a new filter, returns the ID upon success. Note that in the conditions JSON object only one first-level condition group is supported, and it must be glued with 'AND', and only two second level condition groups are supported of which one must be glued with 'AND' and the second with 'OR'. Other combinations do not work (yet) but the syntax supports introducing them in future. For more information, see the tutorial for [adding a filter](#).

Body parameters

application/json

name STRING REQUIRED	The name of the filter
--	------------------------

conditions
OBJECT REQUIRED

The conditions of the filter as a JSON object. Please note that a maximum of 16 conditions is allowed per filter and date values must be supplied in the YYYY-MM-DD format. It requires a minimum structure as follows: ["glue":"and","conditions":[{"glue":"and","conditions": [{"CONDITION_OBJECTS}]}]}],["glue":"or","conditions": [{"CONDITION_OBJECTS}]}]}]. Replace CONDITION_OBJECTS with JSON objects of the following structure: {"object": "", "field_id": "", "operator": "", "value": "", "extra_value": ""} or leave the array empty. Depending on the object type you should use another API endpoint to get field_id. There are five types of objects you can choose from: "person", "deal", "organization", "product", "activity" and you can use these types of operators depending on what type of a field you have: "IS NULL", "IS NOT NULL", "<=", ">=", "<", ">", "<=", ">=", "LIKE '\$%'", "LIKE '%\$'", "NOT LIKE '\$%'", "NOT LIKE '%\$%'". To get a better understanding of how filters work try creating them directly from the Pipedrive application.

type STRING REQUIRED	The type of filter to create
--	------------------------------

VALUES deals leads org people products activity

Response

200 OK

```
{  
    "success": true  
    - "data": (...)  
}
```

Expand all Copy code



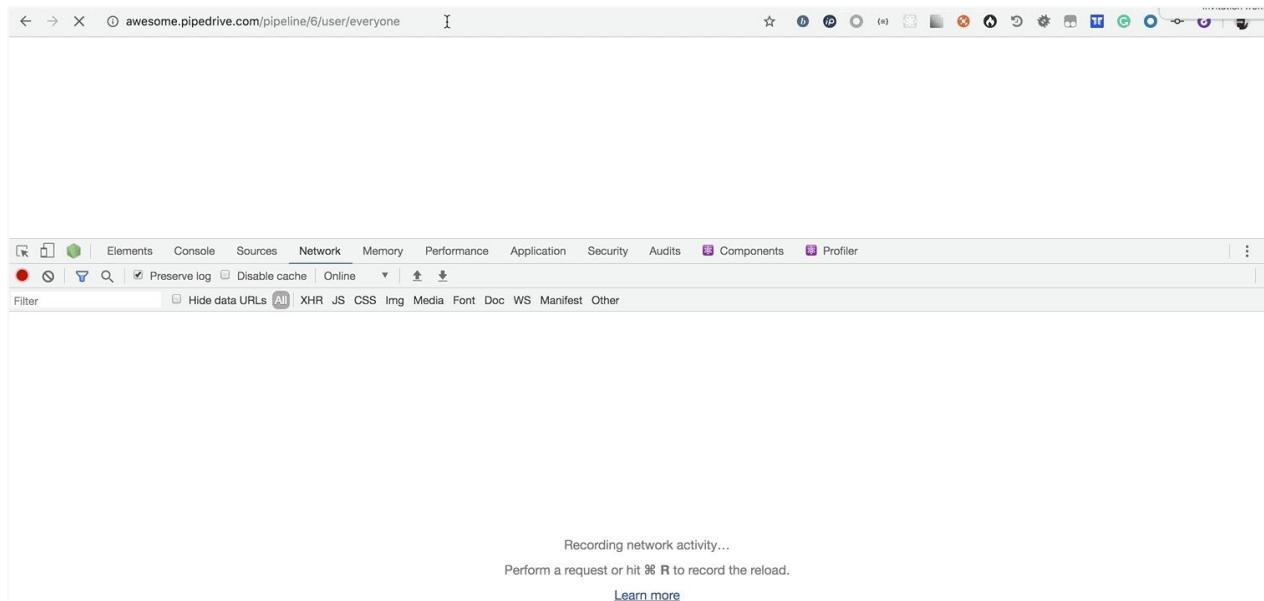
Overfetching

→ C https://00000.pipedrive.com/api/v1/deals?related_o

```
{  
  "success": true,  
  "data": [  
    {  
    },  
    {  
    },  
    {  
    }  
  ],  
  "additional_data": {  
    "pagination": {  
      "start": 0,  
      "limit": 90,  
      "more_items_in_collection": false  
    }  
  },  
  "related_objects": {  
    "user": {  
      "2113502": {  
        "id": 2113502,  
        "name": "Артём Курапов",  
        "email": "artjom.kurapov@pipedrive.com",  
        "has_pic": true,  
        "pic_hash": "e3be9b279e1e0e235f305567210a153d",  
        "active_flag": true  
      }  
    },  
    "person": {  
    }  
  }  
}
```



Underfetching + request dependencies



Manual API composition

- Synchronous and slow
- Denormalized, Not really REST
- Complex and hard to maintain
 - Timeouts
 - Error handling
 - Conditional API joins - custom GET params
 - Redundant internal requests
 - N+1 internal requests

```
1  + { "success": true,
2  +   "data": {
3  +     "id": 2113502,
4  +     "name": "Артём Курапов",
5  +     "default_currency": "EUR",
6  +     "locale": "en_US",
7  +     "lang": 1,
8  +     "email": "artjom.kurapov@pipedrive.com",
9  +     "phone": null,
10 +     "activated": true,
11 +     "2fa_enabled": false,
12 +     "created": "2017-01-24 08:52:00",
13 +     "modified": "2019-10-09 08:18:01",
14 +     "signup_flow_variation": "short_form",
15 +     "has_created_company": true,
16 +     "is_admin": 0,
17 +     "company_name": "Awesome Web Design Agency",
18 +     "company_status": "freeloader",
19 +     "company_add_time": "2012-05-23 10:28:12",
20 +     "company_id": 48069,
21 +     "original_company": 0,
22 +     "active_flag": true,
23 +     "timezone_name": "Europe/Helsinki",
24 +     "role_id": 1,
25 +     "icon_url": "https://d3myhnqlqwz314.cloudfront.net/profile_120x120_2113502_e3be9b279e1e0e235f305567210a153d",
26 +     "is_you": true,
27 +     "data_extra_field_limit": 50,
28 +     "counts": {},
29 +     "current_company_features": {},
30 +     "company_size": null,
31 +     "current_user_settings": {},
32 +     "companier": [],
33 +     "current_company_plan": {},
34 +     "current_company_mrr": 27423,
35 +     "connections": [],
36 +     "fields": {},
37 +     "stages": {},
38 +     "currencies": {},
39 +     "custom_views": {},
40 +     "activity_types": {},
41 +     "dialog_additional_fields": {},
42 +     "language": {},
43 +     "global_messages": [],
44 +     "cc_email": "",
45 +     "alternative_email": [],
46 +     "pipelines": [],
47 +     "started_paying_time": "2013-01-14 23:13:40",
48 +     "seconds_to_trial_end": 256745107,
49 +     "add_user_trial_extended_flag": false,
50 +     "intercom_user_hash": "",
51 +     "seats_quota": 277,
52 +     "user_count": 426,
53 +     "company_country": "EE",
54 +     "recurrly_coupons": "gold_signup_monthly",
55 +     "current_company_test_cases": [],
56 +     "timezone_offset": 10800,
57 +     "3rd_party_auth_links": {},
58 +     "mgw_active": true
59 +   },
60 +   "additional_data": {}
61 }
```

Multiple API composition endpoints

← → C ⌘ ⌘ https://app.pipedrive.com/menu-waitress/v2

```
{  
  - menus: {  
    + primary: [ ... ],  
    + detached: [ ... ],  
    + secondary: [ ... ],  
    + more: [ ... ]  
  },  
  rootUrl: "/",  
  + hiddenPaths: [ ... ],  
  - user: {  
    pic_url: "https://d3myhnlqw2314.cloudfront.net/profile_1"  
    name: "Artjom Kurapov",  
    language: "en-US",  
    email: "artjom.kurapov@pipedrive.com",  
    id: "354645",  
    + frontMenuState: [ ... ]  
  },  
  + company: { ... },  
  + services: [ ... ],  
  cacheVersion: "de7f35eb2b_2741",  
  currentVersion: "de7f35eb2b_2741",  
  + redirects: [ ... ],  
  + blacklist: { ... },  
  + viewSelects: [ ... ]  
}
```

pipedrive

← → C ⌘ ⌘ https://app.pipedrive.com/api/v1/flow/deal/436

```
{  
  success: true,  
  - data: [  
    - {  
      object: "activity",  
      timestamp: "2022-12-09 12:16:49",  
      + data: { ... }  
    },  
    - {  
      object: "note",  
      timestamp: "2022-12-09 12:16:57",  
      - data: {  
        id: 143,  
        user_id: 622732,  
        deal_id: 436,  
        person_id: 240,  
        org_id: 182,  
        lead_id: null,  
        content: "egregregr",  
        add_time: "2022-12-09 12:16:57",  
        update_time: "2022-12-09 12:16:57",  
        active_flag: true,  
        pinned_to_deal_flag: false,  
        pinned_to_person_flag: false,  
        pinned_to_organization_flag: false,  
        pinned_to_lead_flag: false,  
        last_update_user_id: null,  
        - organization: {  
          name: "MI6"  
        },  
        - person: {  
          name: "James Bond"  
        },  
        - deal: {  
          title: "beehives deal"  
        },  
        lead: null,  
        + user: { ... },  
        - comments: {  
          count: 1,  
          - user_ids: [  
            - user_id: 143  
          ]  
        }  
      }  
    }]  
  ]  
}
```

API composition mission



Make API efficient



Artjom Kurapov



Erik Schults

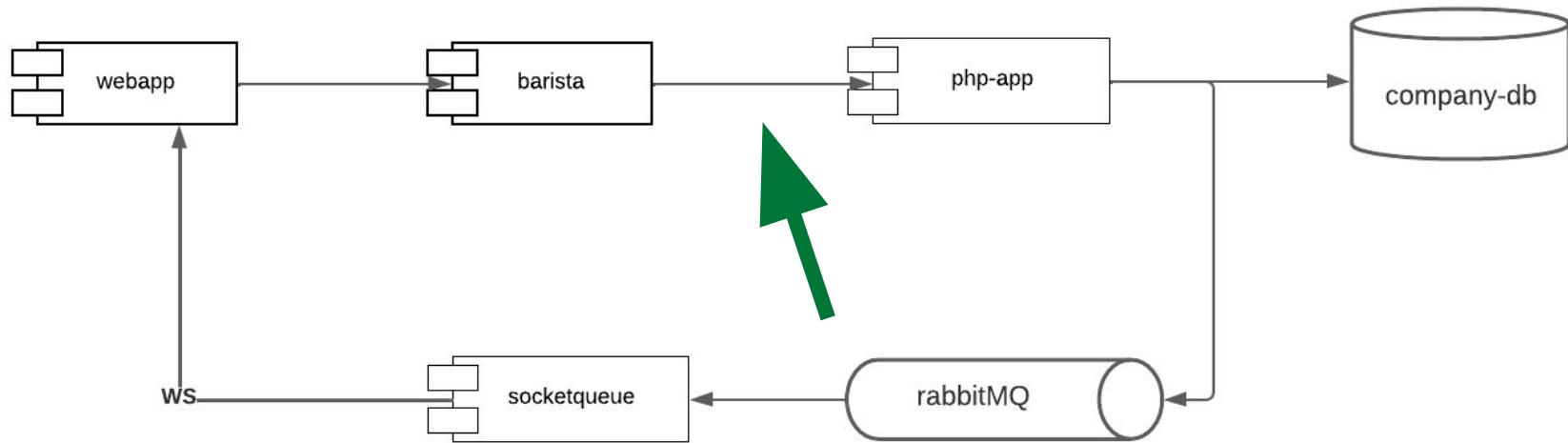


Oleksandr Shvechykov



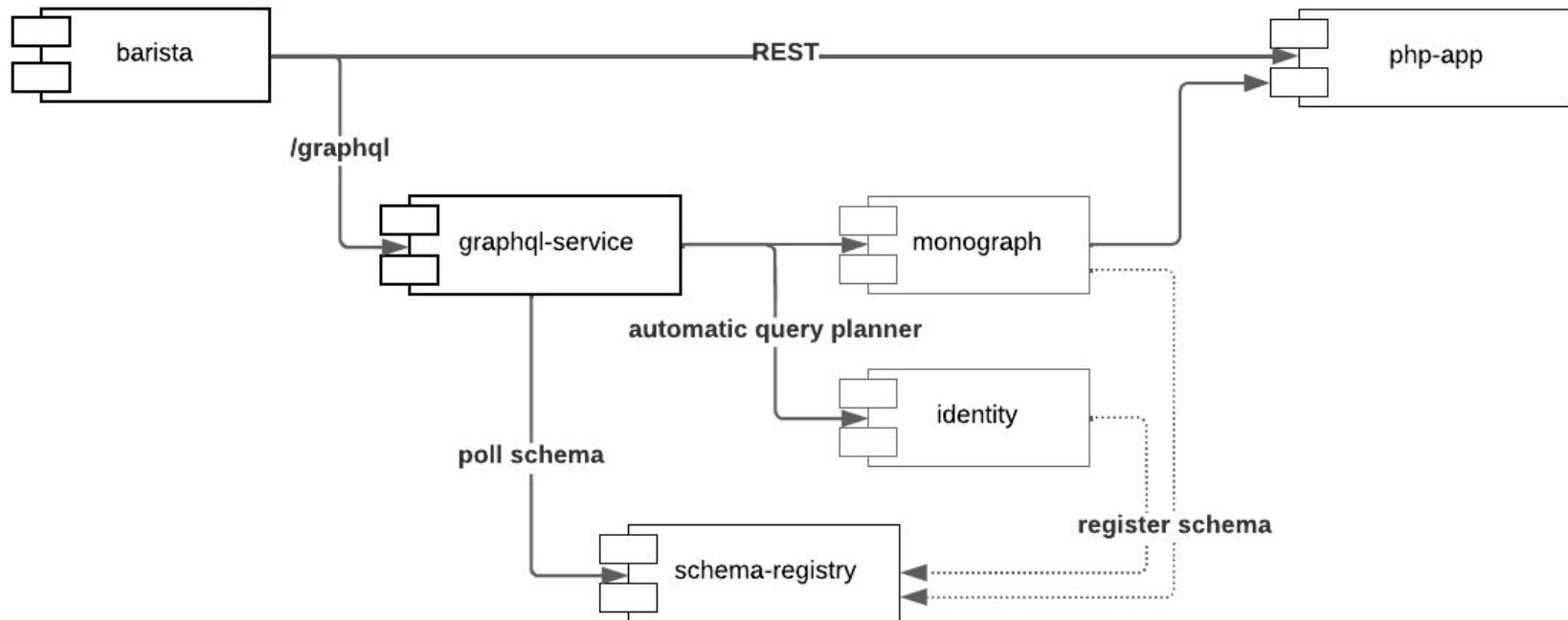
Aleksander Gasna

Add GraphQL here



A

Apollo federation



graphql-schema-registry service

SERVICES (2) PERSISTED QUERIES (0)

user-cycle > Search..

swarm-api > #3 +2539 added 9 months ago

Schema #3
Added 18:30, 25 October 2021 (GMT+3)
URL: 127.0.0.1:60002

DEACTIVATE

SDL	DIFF	USAGE	CONTAINERS (0)
1 scalar JSON	1 scalar JSON		
2	2		
3 type Query {	3 +scalar DateTime		
4 privet: String	4 +		
5 - account: JSON	5 + account: User		
6 }	6 }		
7	7		
8 type Mutation {	8 type Mutation {		
9 register(email: String, password: String): JSON	9 register(email: String, password: String): JSON		
10 }	10 }		
11	11 +		
12	12 +		
13	13 +		
14	14 +type User {		
15	15 + email: String		
16	16 + first_name: String		
17	17 + last_name: String		
18	18 + date_added: DateTime		
19	19 +}		

Entity Total hits

Query.hive	3
Query.apiry	2
Query.aparies	
Query.inspection	
Mutation.addApiary	
Mutation.updateApiary	
Mutation.addHive	
Mutation.updateHive	
Mutation.deactivateHive	
Mutation.addInspection	
Mutation.uploadFrameSide	
Apairy.id	2
Apairy.name	2
Apairy.hives	77
Apairy.location	
Hive.id	5
Hive.name	5

USAGE

CONTAINERS (8)

Client Hits Day

25	2022-07-14
50	2022-07-15
2	2022-07-16

SERVICES (2) PERSISTED QUERIES (0)

user-cycle > Search..

swarm-api > #15 +28 added 4 months ago

#14 +320 added 4 months ago

#13 +112 added 5 months ago

#11 +17 added 5 months ago

#2 +113 added 9 months ago

Schema #13
Added 23:38, 19 February 2022 (GMT+2)
URL: 127.0.0.1:4000

DEACTIVATE

SDL	DIFF	USAGE	CONTAINERS (0)
1 scalar JSON	1 scalar JSON		
2	2		
3 type Query {	3 +scalar DateTime		
4 privet: String	4 +		
5 - account: JSON	5 + account: User		
6 }	6 }		
7	7		
8 type Mutation {	8 type Mutation {		
9 register(email: String, password: String): JSON	9 register(email: String, password: String): JSON		
10 }	10 }		
11	11 +		
12	12 +		
13	13 +		
14	14 +type User {		
15	15 + email: String		
16	16 + first_name: String		
17	17 + last_name: String		
18	18 + date_added: DateTime		
19	19 +}		

graphql-query-cost library

```
type Query {  
    field: String @cost(complexity: 3)  
    default: String  
}
```



```
type Query {  
    parents(limit: Int): [Parent] @cost(complexity: 2, multipliers: ["limit"])  
}
```

```
type Parent {  
    name: String @cost(complexity: 8, useMultipliers: false)  
}
```



Mission results

- ✓ **13% decrease of initial pipeline page load**
(5.4s → 4.7s)
- ✓ **25% decrease of cached request pipeline page load** (5.4s → 4s)

Queries (Prettify) Merge Copy History

```

1 query{
2   user{
3     name
4   }
5   company{
6     country
7   }
8 }
```

id
name
status
domain
country
region
migration_status
public_traffic_lock_flag
ownerUser

data:
 user:
 name: "Artjom Kurapov"
 company:
 country: "EE"

Activities Connection

Activities Connection

node(id: ID!): Node

Fetch a node (any type of object) by ID.
Use fragments to specify fields to fetch, depending on object type

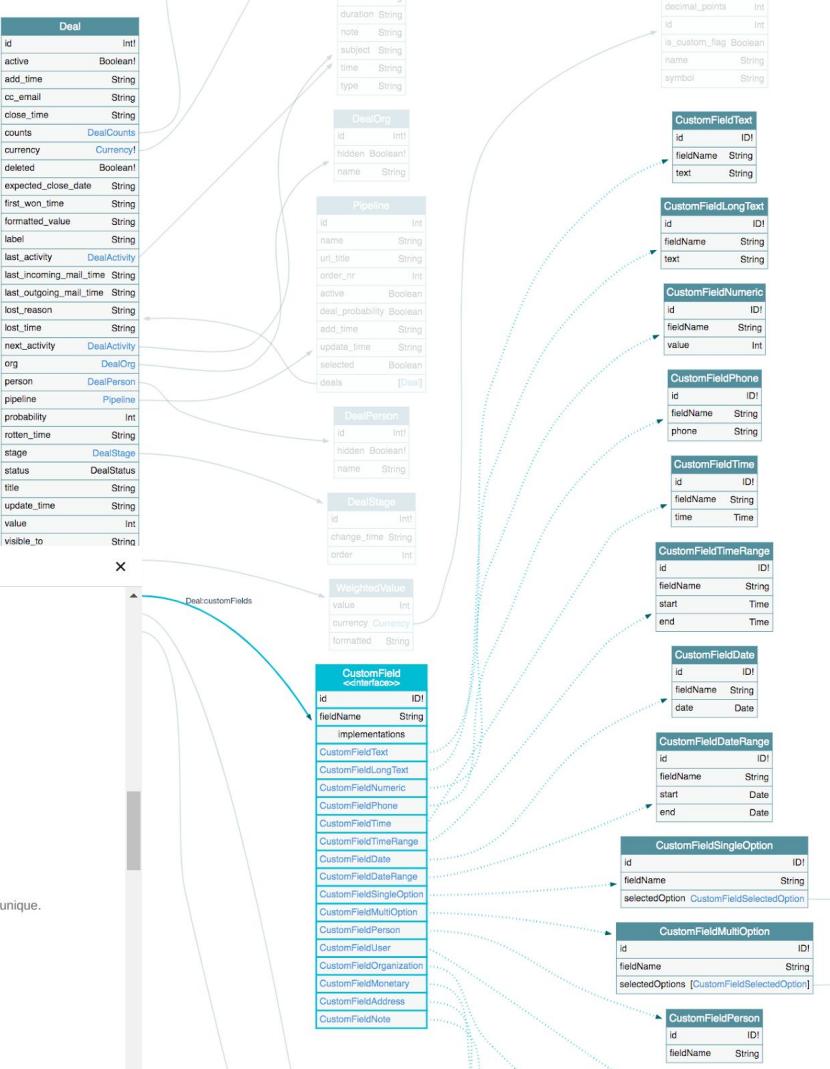
id - globally unique ID of the object to fetch

Note that **Node.id** here will ignore opaque param and will always be globally unique.

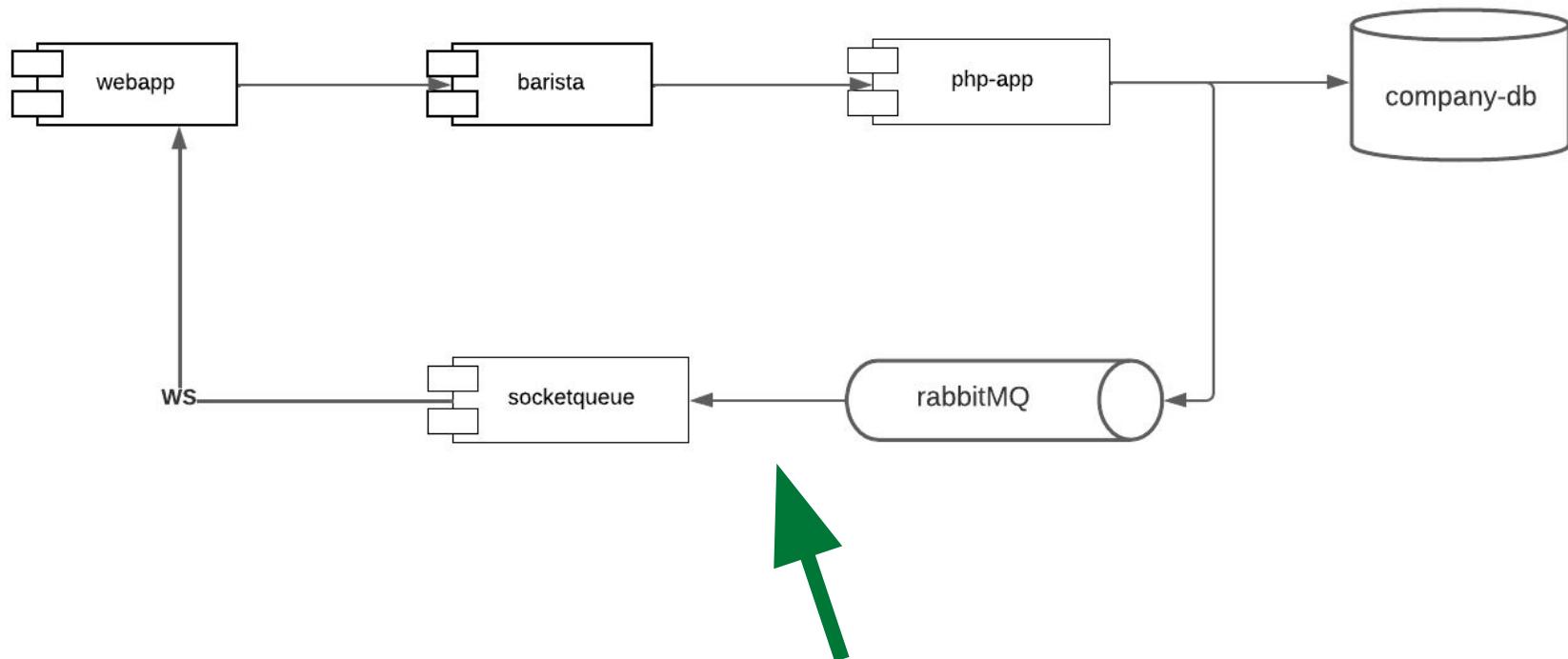
Teams Connection

Changelog Connection

Returns user changelogs by user id (opaque string)



What about this part?



Why subscriptions?



“

**Why is Pipedrive is consuming
80GB traffic per day?***

*Before socketqueue traffic was compressed

Problems - denormalized undocumented schema

The screenshot shows the Pipedrive web application interface. At the top, there's a navigation bar with a search bar and user profile information. Below it, a deals list is displayed with various columns like Leads In, Needs Discov..., Idea, Contact Made, New Stage, Demonstration, Proposal Pre..., and In Negotia. A green button labeled '+ Deal' is visible. The main content area has a header 'Leads In' and a list of deals.

At the bottom, the browser's developer tools Network tab is open, specifically the 'Messages' section under the 'websocket' tab. This tab lists numerous messages from a socket connection, showing a log of interactions between the client and the server. The messages are timestamped and show various actions like 'updated' and 'person' objects being exchanged.

Name	Length	Time
5-8ODghhzU4QYQS9L2-pKwJwf-O8oe4ZvyxFkX...	16931	14:00:37.780
websocket	17343	14:00:37.782
	17238	14:00:37.798
	16768	14:00:37.861
	17349	14:00:37.921
	17362	14:00:37.990
	17721	14:00:38.006
	17239	14:00:38.245
	16913	14:00:38.264
	16769	14:00:38.303
	17363	14:00:38.325
	17350	14:00:38.334
	17741	14:00:38.344

At the very bottom of the page, there's a footer with the Pipedrive logo.

Problems - scalability

Name

x Headers Messages Initiator Timing

websocket

info?t=1601493940673

▼ General

Request URL: `wss://channel9.us-east-1.pipedrive.com/sockjs/344/ieg5thu3/websocket`

Request Method: GET

Status Code: 101 Switching Protocols

► Response Headers (13)

▼ Request Headers view source

Accept-Encoding: gzip, deflate, br

Accept-Language: en-GB,en;q=0.9,en-US;q=0.8,ru;q=0.7

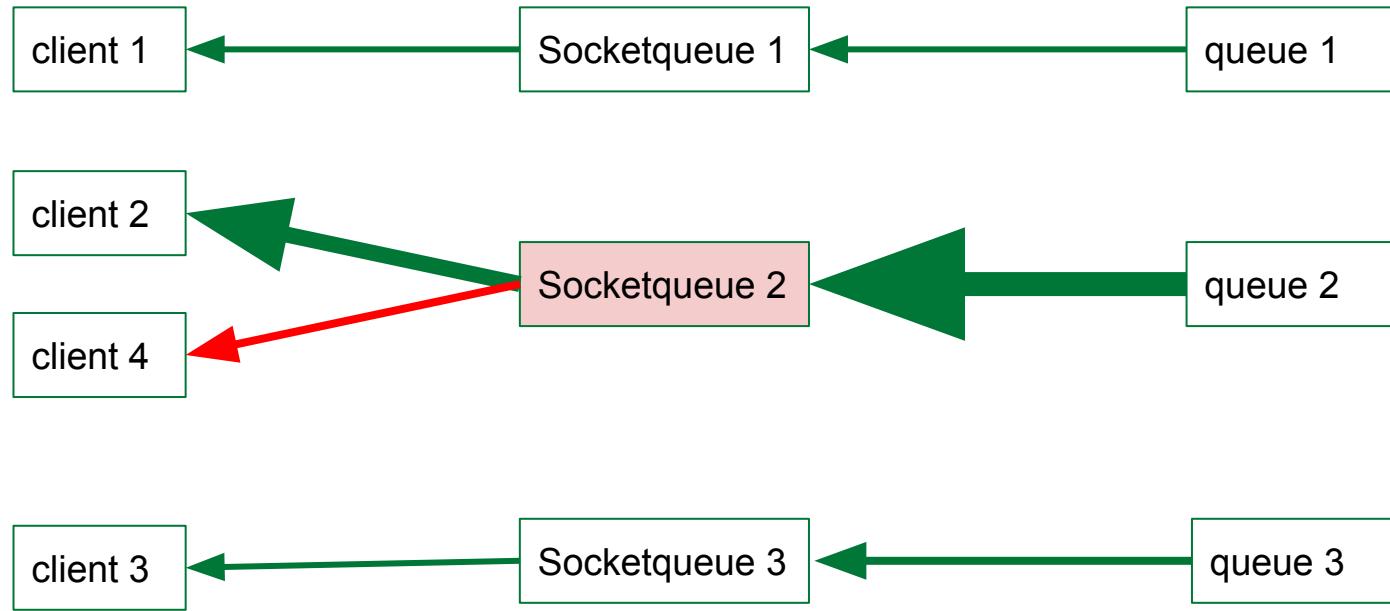
Cache-Control: no-cache

Connection: Upgrade

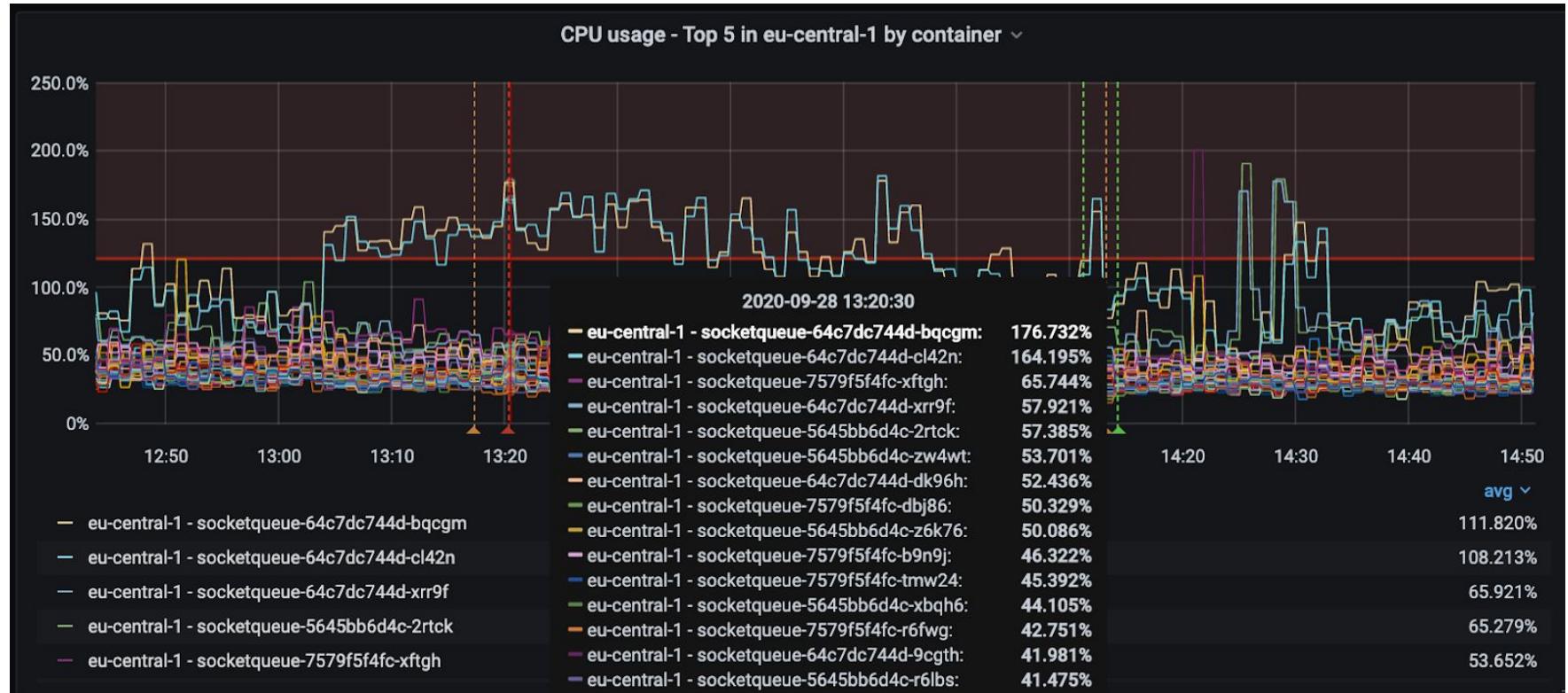
Host: channel9.us-east-1.pipedrive.com

Origin: https://awesome.pipedrive.com

Problems - scalability with fixed queues

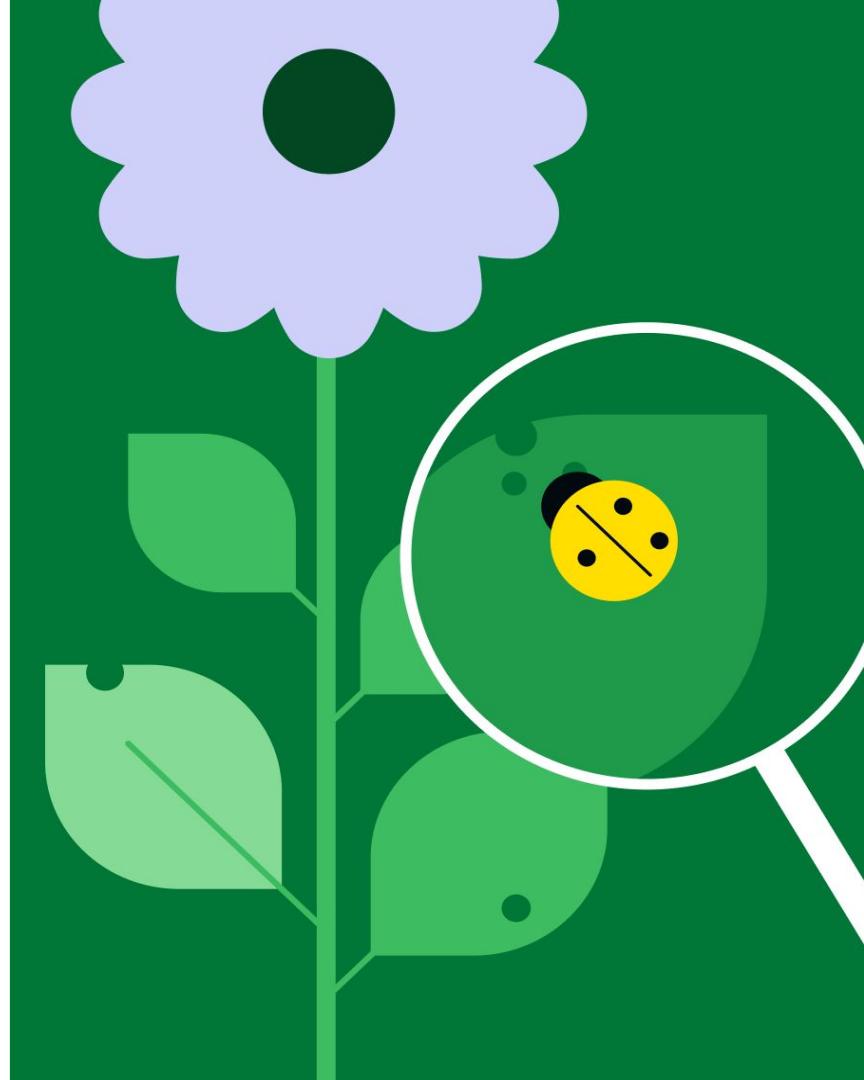


Problems - noisy neighbour → CPU → infra waste



Problems - reliability

- ✗ Loss of events if RabbitMQ is down
- ✗ Loss of frontend state if user connection is down



Proof of concept

- Websocket transport
- Apollo server / subscriptions-transport-ws



A screenshot of a GraphQL subscription testing interface. The URL is `https://graphql-subscriptions.pipedrive.dev/graphql`. The query is:

```
1 subscription{  
2   deal  
3 }
```

The response is a JSON object representing a deal:

```
{  
  "data": {  
    "deal": {  
      "creator_user_id": 10380622,  
      "expected_close_date": null,  
      "first_won_time": null,  
      "owner_id": 10380622,  
      "stage_id": 38,  
      "probability": null,  
      "custom_fields": {},  
      "close_time": null,  
      "title": "erg deal",  
      "lost_reason": null,  
      "update_time": "2021-05-04T14:00:53Z",  
      "won_time": null,  
      "visible_to": "3",  
      "org_id": 81,  
      "...  
    }  
  }  
}  
Listening
```

The interface includes tabs for PRETTIFY, HISTORY, COPY CURL, SCHEMA, and DOCS. A red button labeled "SUBSCRIBE" is visible on the right side of the response pane. The pipedrive logo is at the bottom left.

GraphQL subscriptions mission



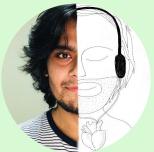
Make events efficient



Artjom Kurapov



Pavel Nikolajev



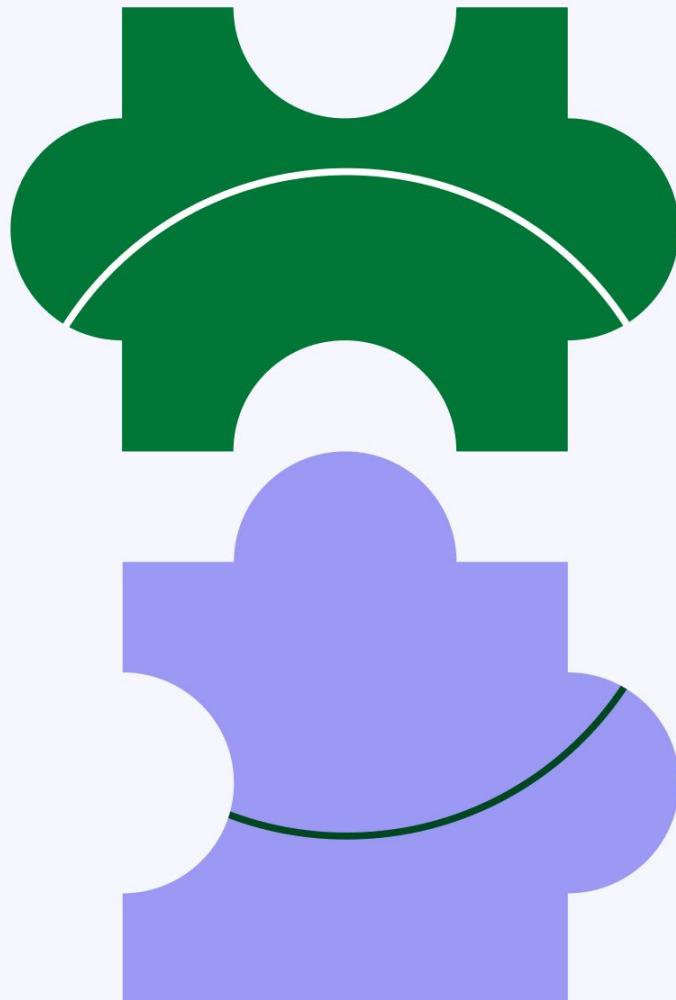
Abhishek Goswami



Kristjan Luik

Scope

- Authentication
- **Filtering** + Limiting stream to specific user
- Routing
- Enrichment with timeouts
- Handle state on connection loss
- Test multiple subscriptions per view
- Testing scalability
- Testing performance. CPU, memory



Mission progress



- Tried [graph-gophers/graphql-go](#)
- Tried [99designs/gqlgen](#)



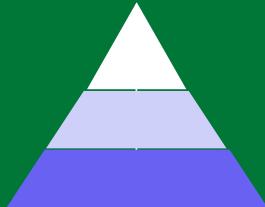
- Ended up still with nodejs:
[apollosolutions/federation-subscription-tools](#)
[enisdenjo/graphql-ws](#)



Dynamic federated schema merging



Schema



Mini demo

Subscriptions Prettify Merge Copy History

```
1subscription{
2  dealChanged(ids:[540]){
3    delta
4    raw{
5      title
6    }
7    enriched{
8      pipeline{
9        name
10     }
11    creator{
12      name
13    }
14  }
15  meta{
16    time
17  }
18}
19}
```

The image shows a screenshot of the Pipedrive CRM interface. On the left, there's a code editor window titled "Subscriptions" displaying GraphQL query code. Above the code editor are buttons for "play", "Prettify", "Merge", "Copy", and "History". To the right of the code editor is a dark sidebar with various icons: a magnifying glass, a dollar sign, a checkmark, a megaphone, an envelope, a calendar, and a graph. The main area of the interface is titled "Deals" and shows a list of deals. One deal is highlighted with a yellow warning icon and labeled "Wise deal" with a value of "10 000 €". Other deals visible include "Prospect", "Contact establis...", and "Product demo". A search bar at the top right says "Search Pipedrive".

Raw events

- Delivers data from kafka
- camelCase remapping
- Permission check
- Fast & most reliable



```
type Subscription {
    dealAdded: DealEvent
}

type DealEvent {
    raw: DealDomainEvent
}

type DealDomainEvent {
    id: Int!
    ownerId: Int
    stageId: Int
    orgId: Int
    personId: Int
    pipelineId: Int
    creatorUserId: Int

    title: String
    status: String
    value: Float
    probability: Float
    currency: String
    visibleTo: Int

    addTime: Time
    updateTime: Time
    lostTime: Time
    wonTime: Time
    firstWonTime: Time
    expectedCloseDate: Time
    stageChangeTime: Time
    closeTime: Time
}
```

Delta field

- Subscribe & deliver only changed fields (diff)
- Relies on original event to have this data
- JSON type - no property **usage** tracking
- Frontend likely needs to **query** original entity first
- Useful if frontend has **storage layer**

```
type Subscription {  
    dealAdded: DealEvent  
}  
  
type DealEvent {  
    delta: JSON  
}
```

GraphQL  Prettify Merge Copy History

```
1 ▼ subscription{  
2     dealChanged(ids:[31]){  
3         delta  
4     }  
5 }
```



```
{  
    "data": {  
        "dealChanged": {  
            "delta": {  
                "updateTime": "2023-09-01T12:00:00Z",  
                "wonTime": "2023-09-01T12:00:00Z",  
                "closeTime": "2023-09-01T12:00:00Z",  
                "status": "Open"  
            }  
        }  
    }  
}
```

Event types - Universal vs Domain Driven

- Both have filtering by IDs
- Universal - for FE storage sync
 - ORDER of events is important
- Custom - flexibility for specific views

```
type Subscription {  
    # universal events  
    dealEvent(filter: DealEventInput): DealEvent  
  
    # action-specific events  
    dealAdded: DealEvent  
    dealChanged(ids: [ID]!, onlySubscribedFieldChanges: Boolean): DealEvent  
    dealDeleted(ids: [ID]!): DealEvent  
}
```

Enriched events

- Allows deep enrichment
- Queries federated gateway
- Query is hard-coded
- Entity ID is taken from kafka event
- Useful if frontend has no unified storage layer (**react views**)

```
type Subscription {  
    # universal events  
    dealEvent(filter: DealEventInput): DealEvent  
  
    # action-specific events  
    dealAdded: DealEvent  
    dealChanged(ids: [ID]!): DealEvent  
    dealDeleted(ids: [ID]!): DealEvent  
}  
  
input DealEventInput {  
    # mandatory, because creation type doesn't need ids  
    eventTypes: [EventType]!  
  
    # optional  
    ids: [ID]  
}  
  
type DealEvent {  
    enriched: Deal  
    meta: EventMeta  
}
```



Live queries

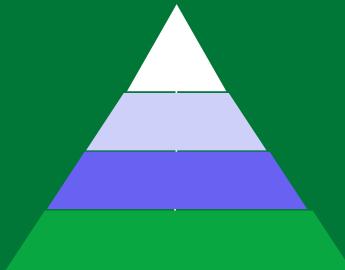
- Query + Subscribe
- Async iterator magic

```
1 ▶ subscription{
2 ▶   dealChanged(ids:["436"], liveQuery:true){
3 ▶     enriched{
4       title
5       creator{
6         name
7       }
8     }
9   }
10 }
```

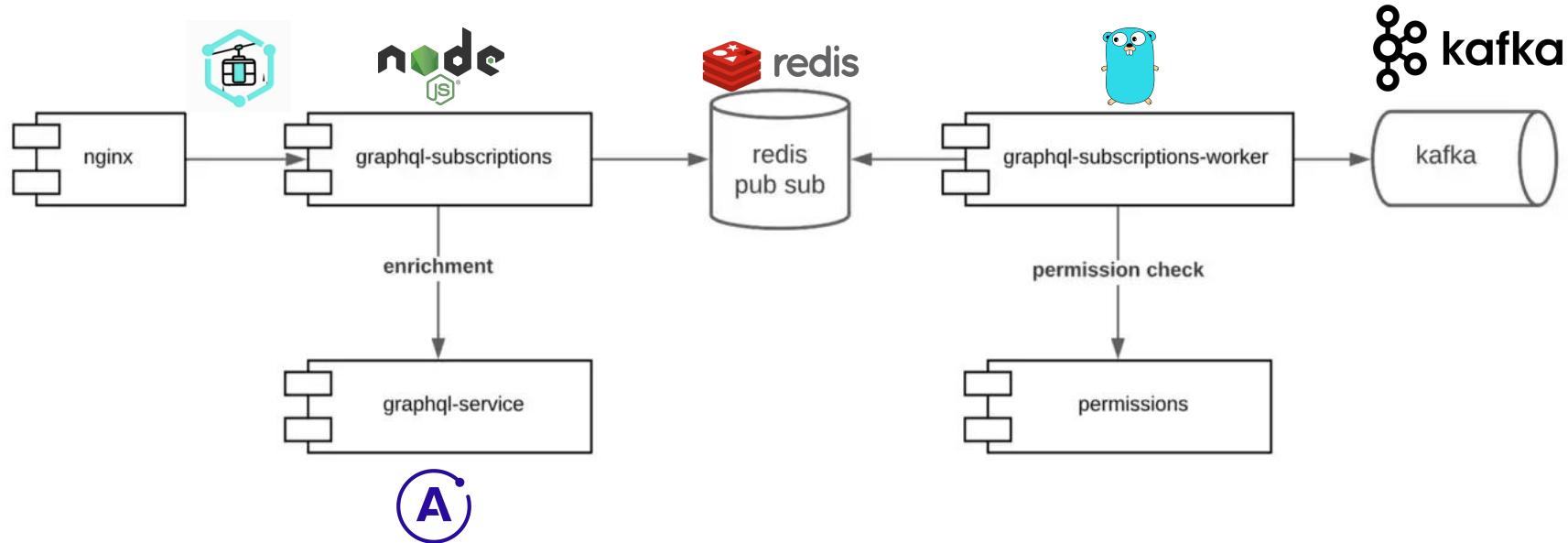


I

Architecture



Services

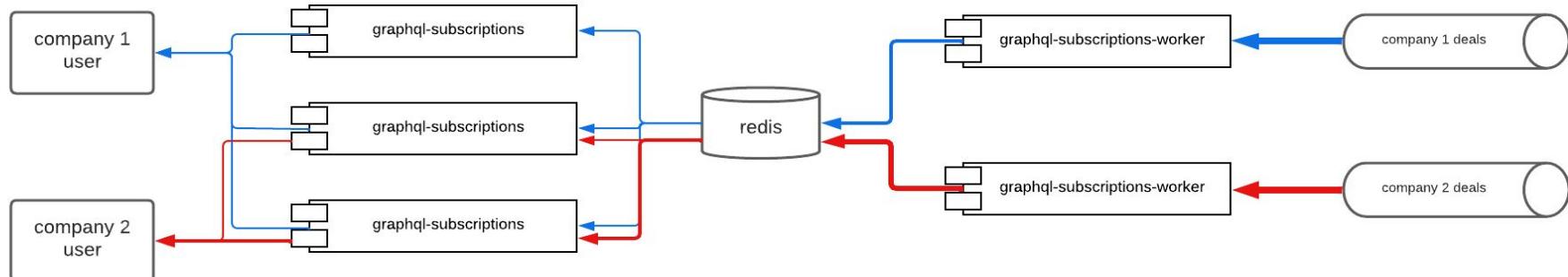


Scalability

- Workers scale to max amount of kafka partitions
- WS connections scale horizontally
- Redis CPU is the weakest spot
- For more efficiency, in the future, subscription state (users, ids) could be passed to **workers**

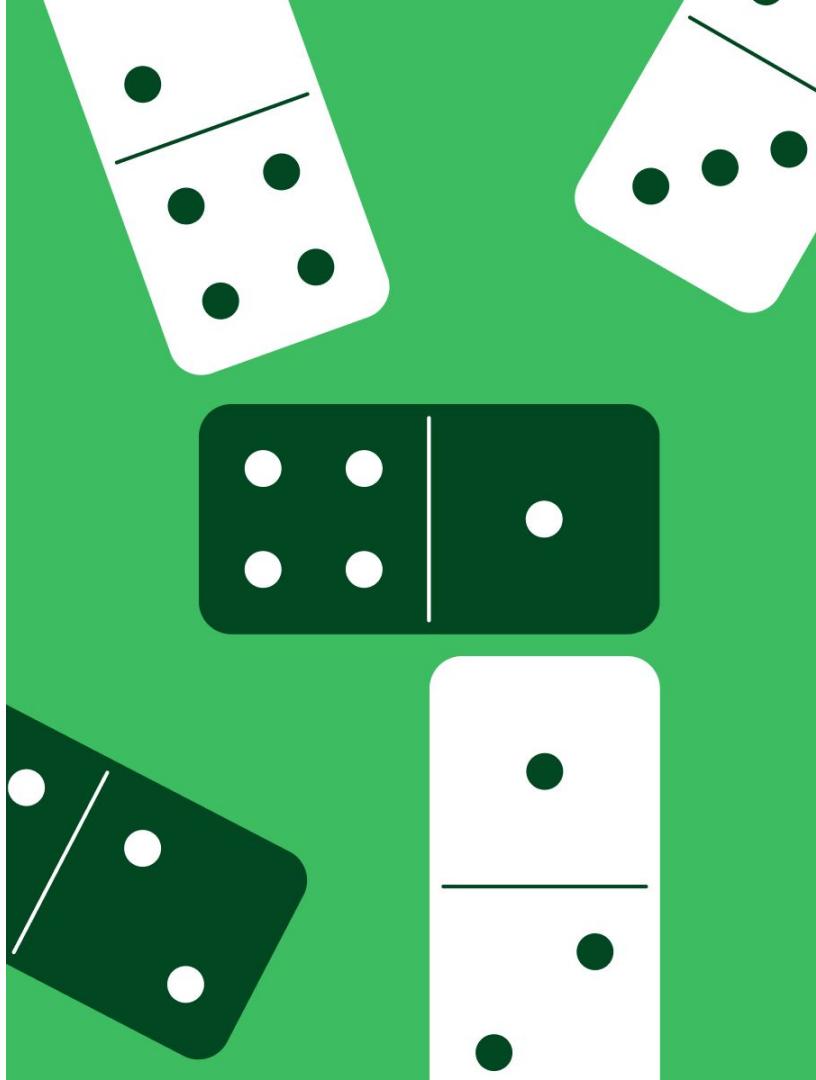
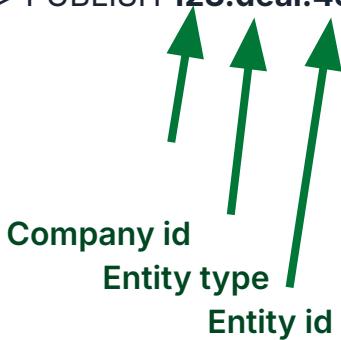
Pods

Name	Namespace	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created
graphql-subscriptions-7b85f4d65f-mxsdx	graphql	Running	0	60.00m	481.01Mi	3 days ago
graphql-subscriptions-worker-d4bfc7b6d-tb2ps	graphql	Running	0	41.00m	59.03Mi	3 days ago



Redis pub-sub channels

- We use high-availability redis sentinel
- Routing idea is similar to RabbitMQ,
except there are no in-memory queues
- redis-cli -h localhost -p 6379
- > PUBLISH **123.deal.456** some-json-here



Performance testing

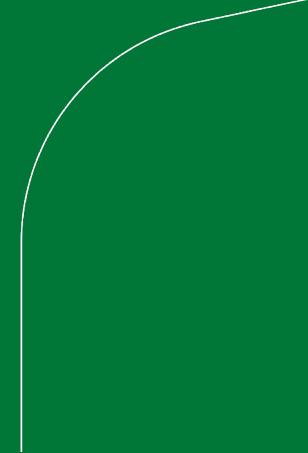
- One entity type ~ 200 events / sec
- Rolled out to 100% of customers
- Stress tested in live with 50 subscriptions per connection



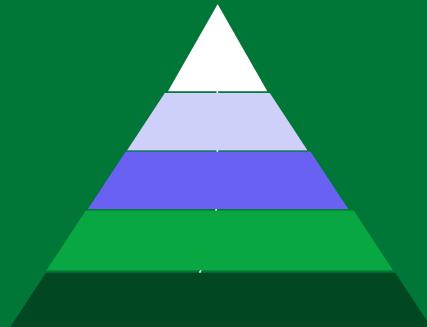
Limitations

- Max amount of IDs per subscription
 - Redis channels per subscription
- Connection per user per pod
 - Subscriptions per connection
- Subscriptions per pod
- Enrichment request timeout
- Connection time to live
- Disconnect users
 - On logout
 - On permission change
- Automatic reconnect (by graphql-ws)

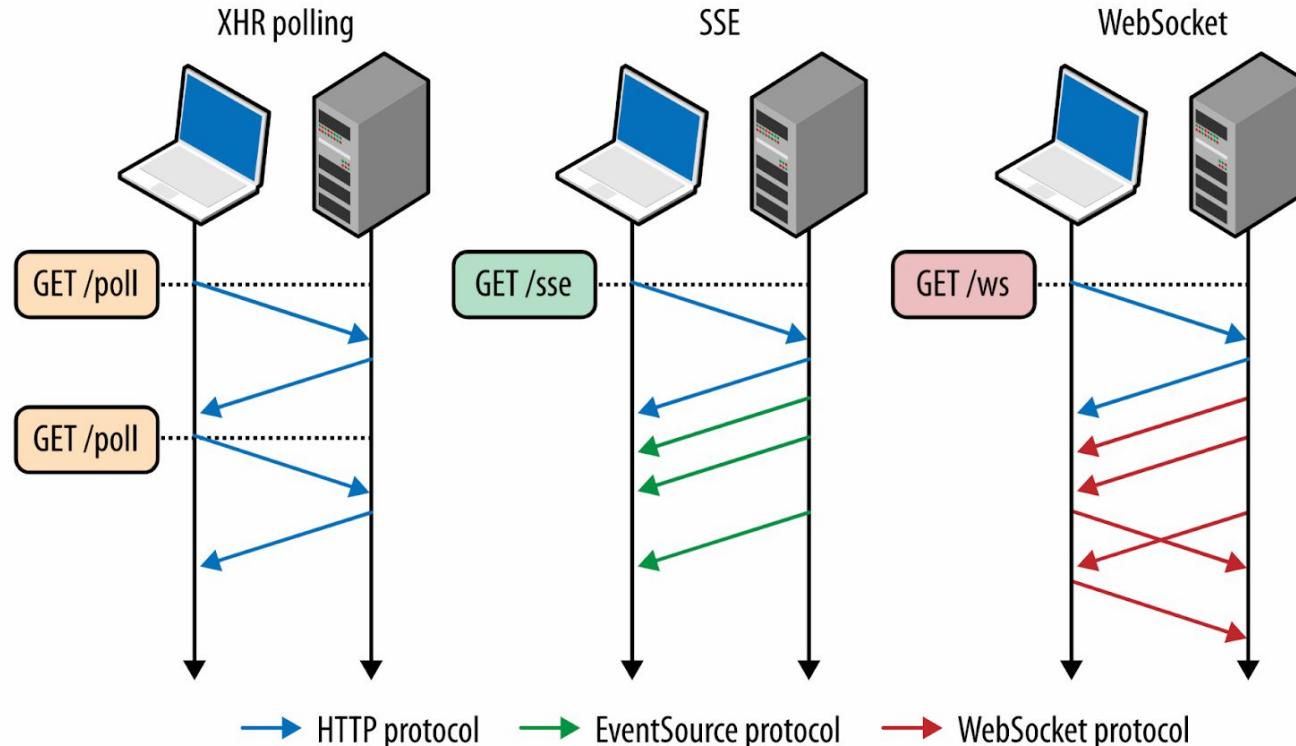




Code



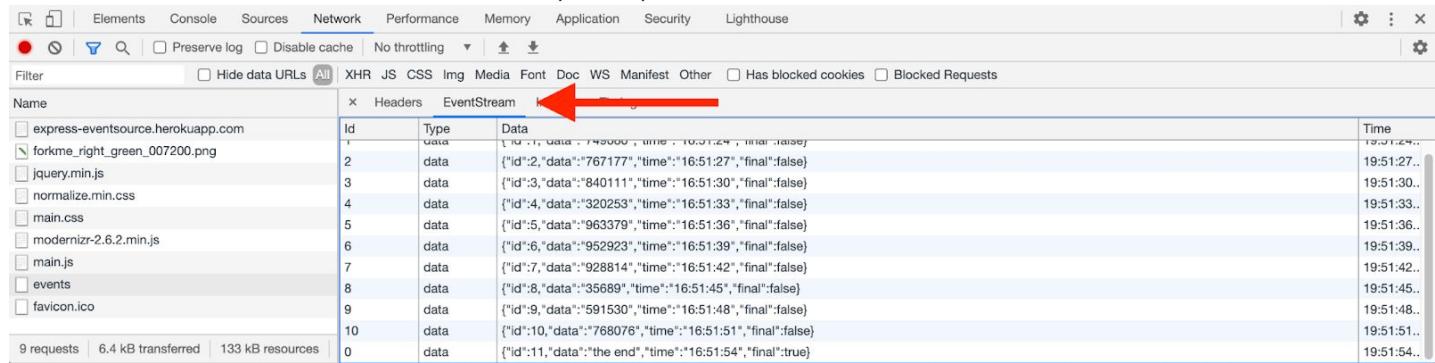
Transport



Server Sent Events over HTTP2 transport

- Only unidirectional data flow
- Basically a HTTP long polling on steroids - no HTTP connection closing when individual data item is sent, no need to explicitly re-establish connection
- Very simple to set up both client & BE
 - Built in support for re-connection and event id
- A text/event-stream with JSON payload separated by new line characters
- Adds 5 bytes per msg overhead
- Only UTF :(
- Over HTTP1, limited to 6 connections :(
- enisdenjo/graphql-sse lib could now be used - first commit Jun 22, 2021, after mission

```
if (!!window.EventSource) {  
  var source = new EventSource('stream.php');  
} else {  
  // Result to xhr polling :(  
}  
  
source.addEventListener('message', function(e) {  
  console.log(e.data);  
}, false);  
  
source.addEventListener('open', function(e) {  
  // Connection was opened.  
}, false);  
  
source.addEventListener('error', function(e) {  
  if (e.readyState == EventSource.CLOSED) {  
    // Connection was closed.  
  }  
}, false);
```



The screenshot shows the Chrome DevTools Network tab. A red arrow points to the 'EventStream' entry in the list of requests. The table below shows the details of the event stream requests.

Name	Id	Type	Data	Time
express-eventsource.herokuapp.com	1	data	{"id":1,"data":"749000","time":16:51:24,"final":false}	19:51:24..
forkme_right_green_007200.png	2	data	{"id":2,"data":"767177","time":16:51:27,"final":false}	19:51:27..
jquery.min.js	3	data	{"id":3,"data":"840111","time":16:51:30,"final":false}	19:51:30..
normalize.min.css	4	data	{"id":4,"data":"320253","time":16:51:33,"final":false}	19:51:33..
main.css	5	data	{"id":5,"data":"963379","time":16:51:36,"final":false}	19:51:36..
modernizr-2.6.2.min.js	6	data	{"id":6,"data":"952923","time":16:51:39,"final":false}	19:51:39..
main.js	7	data	{"id":7,"data":"928814","time":16:51:42,"final":false}	19:51:42..
events	8	data	{"id":8,"data":"35689","time":16:51:45,"final":false}	19:51:45..
favicon.ico	9	data	{"id":9,"data":"591530","time":16:51:48,"final":false}	19:51:48..
	10	data	{"id":10,"data":"768076","time":16:51:51,"final":false}	19:51:51..
	11	data	{"id":11,"data":"the end","time":16:51:54,"final":true}	19:51:54..

Websocket transport

- Bi-directional full duplex (send at any time)
- HTTP 1.x **upgrade header**
 - Some old firewalls may deny this
- Sub-protocols & versions
- Binary (Blob) or UTF8 (ArrayBuffer)
- Low-level, has many implementation libraries (we use sockjs in socketqueue)
- HTTPS pages require WSS
- Stateful connection
- Nginx needs long-lived connections, otherwise it dies after default 1 min

```
1 let socket = new WebSocket("wss://javascript.info/article/websocket");
2
3 socket.onopen = function(e) {
4   alert("[open] Connection established");
5   alert("Sending to server");
6   socket.send("My name is John");
7 };
8
9 socket.onmessage = function(event) {
10   alert(`[message] Data received from server: ${event.data}`);
11 };
12
13 socket.onclose = function(event) {
14   if (event.wasClean) {
15     alert(`[close] Connection closed cleanly, code=${event.code} reason=${
16   }`);
17   } else {
18     // e.g. server process killed or network down
19     // event.code is usually 1006 in this case
20     alert(`[close] Connection died`);
21 }
22 }
```

Bit	+0..7	+8..15		+16..23	+24..31
0	FIN	Opcode	Mask	Length	Extended length (0..8 bytes) ...
32	...				
64	...				
96	...				
...	...				

pipedrive

Name	x	Headers	Messages	Initiator	Timing
5-F804eUjRop8QrrxeWRtLcRMMeKMjkVq1hw_n-6...	<input type="checkbox"/>	All	<input type="text"/> Enter regex, for example: (web)?socket		
websocket	<input checked="" type="checkbox"/>				
Data					Length Time
↓ o					1 20:36:32.098
↓ a["\\"messageText\"\":\"Connection opened. socketqueue-d854cff97-9fsqd; 24c924fa6d26bfc27cc5e27bed40461c\"]"]					107 20:36:32.102
↓ a["\\"messageText\"\":\"subscribed\"]"]					37 20:36:32.102
↑ a["\\"meta\"\":{\\"company_id\"\":48069,\\"user_id\"\":2113502,\\"user_name\"\":\"Артём Курнов\",\\\"host\"\":\"awesome.pipedrive.com\",\\\"timestamp\"\":1620322592...}					371 20:36:32.467
↓ a["\\"rabbitStateChange\"\":\"open\",\\\"messageText\"\":\"Connection to RabbitMQ opened\"]"]					87 20:36:32.613
↓ a["\\"messageText\"\":\"Started listening for company 48069; my user 2113502!\",\\\"routingKey\"\":\"auth\",\\\"auth\":true]"]					117 20:36:32.613
↓ a["\\"meta\"\":{\\"v\"\":1,\\"action\"\":\"updated\",\\\"object\"\":\"userCounts\",\\\"id\"\":2113502,\\"company_id\"\":48069,\\"user_id\"\":2113502,\\"host\"\":\"awesome.pipedri...]					1059 20:36:32.810
↓ a["\\"meta\"\":{\\"v\"\":1,\\"action\"\":\"updated\",\\\"object\"\":\"userSetting\",\\\"id\"\":\"deals_view_model\",\\\"company_id\"\":48069,\\"user_id\"\":2113502,\\"host\"\":\"aw...]					899 20:36:33.812
↓ a["\\"meta\"\":{\\"v\"\":1,\\"action\"\":\"added\",\\\"object\"\":\"proactive_card\",\\\"id\"\":28496532,\\"company_id\"\":48069,\\"user_id\"\":2113502,\\"host\"\":\"app.pipedrive....]					1836 20:36:40.282

subscriptions-transport-ws

- Originally written by Apollo
- 📈 2016-2018

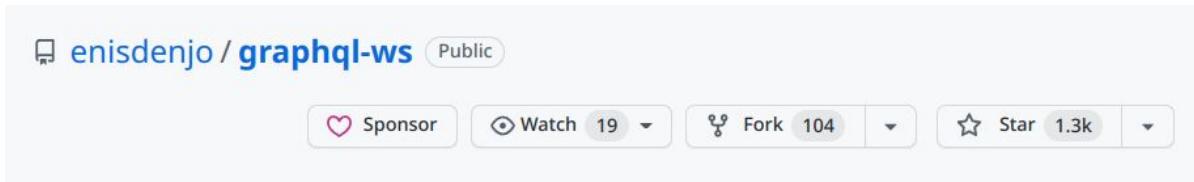
This repository has been archived by the owner before Nov 9, 2022. It is now read-only.

 apollographql / **subscriptions-transport-ws** Public archive

 Watch 72 ▾  Fork 352 ▾  Star 1.5k ▾

graphql-ws

- Easy adoption, has **both frontend and backend** examples
- Security
 - ws payload must be compliant to the protocol, otherwise connection is dropped
 - Connection/auth initialization is complete
- Allows to even send **queries & mutations over ws**
 - Less auth overhead with long-running connection
- Automatic reconnect, exponential back-off
 - Connection termination removed → ws equivalent
 - Keep-alive ping-pongs can be customized



Frontend

```
type Subscription {  
  commentAdded(postID: ID!): Comment  
}
```

```
const COMMENTS_SUBSCRIPTION = gql`  
subscription OnCommentAdded($postID: ID!) {  
  commentAdded(postID: $postID) {  
    id  
    content  
  }  
}  
`;  
  
function LatestComment({ postID }) {  
  const { data: { commentAdded }, loading } = useSubscription(  
    COMMENTS_SUBSCRIPTION,  
    { variables: { postID } }  
  );  
  return <h4>New comment: {!loading && commentAdded.content}</h4>;  
}
```



```
const app = express();
app.use("/graphql", graphqlHTTP({ schema }));
app.use("/", express.static("public"));
app.get('/health', (_ , res) => {
  return res.status(200).send("ok");
});

import { execute, subscribe } from "graphql";
import express from "express";
import { graphqlHTTP } from "express-graphql";
import { useServer } from "graphql-ws/lib/use/ws";
import { WebSocketServer } from "ws";
import { makeExecutableSchema } from "@graphql-tools/schema";
import gql from "gql-tag";
```

```
app.listen(8300, () => {
  const wsServer = new WebSocketServer({
    port: 8350,
    path: "/graphql",
  });
}
```

```
useServer(
  {
    schema,
    execute,
    subscribe,
```

```
>   onConnect: (ctx) => { ... },
>   onSubscribe: (ctx, msg) => { ... },
>   onNext: (ctx, msg, args, result) => { ... },
>   onError: (ctx, msg, errors) => { ... },
>   onComplete: (ctx, msg) => { ... },
  },
),
wsServer
);
});
```



Subscription (with generator function)

A screenshot of a GraphQL playground interface. On the left, there is a code editor window containing the following GraphQL subscription:

```
1▼ subscription{
2  hi
3 }
```

Next to the code editor are several icons: a play button, a copy icon, and a refresh/circular arrow icon. To the right of the code editor is a large, empty white area where the results of the query would be displayed. In the top right corner of this area, there is a small "GraphiQL" label with a plus sign next to it.

```
// schema and resolvers
const schema = makeExecutableSchema({
  typeDefs: gql`  
    type Subscription {  
      hi: String  
    }  
  `,  
  resolvers: {  
    Subscription: {  
      hi: {  
        subscribe: async function* sayHi() {  
          for (const hi of [  
            "Hi",  
            "Привет",  
            "Bonjour",  
            "Hola",  
            "Ciao",  
            "Zdravo",  
          ]) {  
            yield new Promise((resolve) => setTimeout(resolve, 2000));  
            yield { hi };  
          }  
        },  
      },  
    },  
  },
});
```



Subscription (with async iterator + promises)

The image shows a screenshot of a GraphQL playground interface. On the left, there is a code editor window containing the following GraphQL subscription query:

```
1subscription{  
2  numberIncremented  
3}
```

To the right of the code editor is a large, empty white area representing the results of the query. On the far right edge of this results area, there is a small blue button with a plus sign and the text "GraphQL".

On the far left side of the playground, there are some navigation icons: a file icon, a refresh/circular arrow icon, and a back/forward icon.

```
const schema = makeExecutableSchema({
  typeDefs: gql`  
    type Subscription {  
      numberIncremented: Int  
    }  
  
    resolvers: {  
      Subscription: {  
        numberIncremented: {  
          subscribe: () => ({  
            [Symbol.asyncIterator]() {  
              let value = 0;  
              return {  
                async next() {  
                  return new Promise((resolve, reject) => {  
                    setTimeout(() => {  
                      value++;  
                      resolve({  
                        value: { numberIncremented: value },  
                        done: value > 100,  
                      });  
                    }, 1000);  
                  });  
                },  
                async return() {  
                  return { value: { numberIncremented: value }, done: true };  
                },  
                async throw(error) {  
                  return { value: { numberIncremented: value }, done: true };  
                },  
              };  
            }  
          }  
        }  
      }  
    }  
  }`  
});
```



Subscription (with in-memory pub-sub)



```
1▼ subscription{  
2  timePubSub  
3 }
```

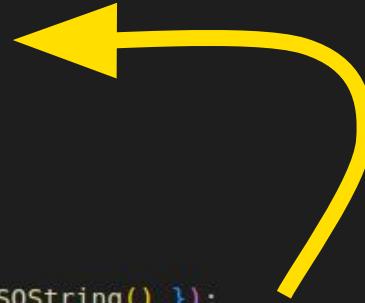


```
▼ {  
  ▼ "data": {  
    "timePubSub": "2022-12-18T20:29:47.253Z"  
  }  
}
```

```
import { PubSub } from "graphql-subscriptions";

const pubsub = new PubSub();

const schema = makeExecutableSchema({
  typeDefs: gql`  
    type Subscription {  
      timePubSub: String  
    }  
  `,  
  resolvers: {  
    Subscription: {  
      timePubSub: {  
        subscribe: ()=> pubsub.asyncIterator("TIME"),  
      }  
    },  
  },  
});  
  
function publishTime() {  
  pubsub.publish("TIME", { timePubSub: (new Date()).toISOString() });  
  setTimeout(publishTime, 1000);  
}  
publishTime();
```



Subscription (with redis pub-sub)

```
import ws from 'ws';
import { RedisPubSub } from 'graphql-redis-subscriptions';

const wsServer = new ws.Server({
  server: fastify.server,
  path: '/graphql',
});

const redisSub = new RedisPubSub({
  publisher: redisClient, // ioRedis instance
  subscriber: redisClient,
});
```

Subscription with redis pub-sub

```
import { useServer } from 'graphql-ws/lib/use/ws';

useServer({
  execute, // from 'graphql'
  subscribe, // from 'graphql'
  context: (ctx) => contextHandler(ctx),
  onConnect: (ctx) => connectHandler(ctx, fastify, redisSub),
  onDisconnect: (ctx: any) => {},
  onClose: (ctx: any) => {},
  onSubscribe: (ctx, msg) => subscribeHandler(ctx, fastify, msg),
  onError: (ctx, message, errors) => {},
},
wsServer,
);
```

Setting datasources &
context from connection to
resolvers

Connection limits, set
redis to ctx

Subscription with redis pub-sub

```
Subscription: {  
    dealAdded: {  
        subscribe: withFilter(  
            () => ctx.redis.asyncIterator('123.deal.456'), // bind to redis channel  
            (payload, variables) => {  
                return true; // check permissions  
            },  
            ),  
            ),  
            async resolve(rawPayload, _, ctx, info) => {}, // enrich  
        },  
    },  
},
```

Pipedrive engineering blog

The screenshot shows a blog post header with the author's profile picture, name, date (Sep 9, 2020), and a 13-minute read time. Below the header is the title 'Journey to a Federated GraphQL'. The main content area features a large, stylized image of a red molecular or GraphQL graph structure against a dark, mountainous background. A caption below the image reads 'Image taken from dgraph.io.' The text discusses the company's five-year journey with GraphQL, mentioning its use in production for API composition and its performance at 110 requests per second with 100ms latency.

Engineers tend to love good stories, so hopefully our 5-year journey of moving towards API composition with [GraphQL](#) now in production (serving at peak 110 requests per second at 100ms latency) provides a good story.

[If you're in a hurry, scroll down to *Lessons learned* and check out the open-sourced [graphql-schema-registry](#).]

A screenshot of the Pipedrive GraphQL schema editor. It shows a sidebar with 'SCHEMA' and 'PERSISTED QUERIES (0)' sections. Under 'SCHEMA', there are entries for 'pets' and 'users'. The 'pets' entry has a timestamp 'added 2 days ago'. The 'users' entry has a timestamp 'added 11 days ago'. In the center, there is a schema definition for 'Schema #4' with a timestamp 'Added 17/11, 28 August 2020 (GMT+3)'. Below it, there are buttons for 'DIFF WITH PREVIOUS', 'DEFINITION', and 'CONTAINERS (0)'. At the bottom, there is a note: 'The query type, represents all of the entry'.

pipedrive

The screenshot shows a blog post header with the author's profile picture, name, date (Nov 23), and a 11-minute read time. Below the header is the title 'A dream of scalable and enriched GraphQL subscriptions'. The main content area features a large, stylized image of a red molecular or GraphQL graph structure superimposed on a photograph of a waterfall cascading over rocks. A caption below the image reads 'A stylised photo of Jagala juga (Estonia), original photo by Aleksandr Abrosimov, Wikimedia Commons'. The text discusses the company's ten-year journey of delivering websocket events to the frontend, building upon the foundation established in the previous article.

In my last article, I wrote about our five-year journey with GraphQL at Pipedrive. Now, I'd like to tell you about a ten-year journey of delivering websocket events to the frontend. Hopefully, it'll be of some help to you, too.

A screenshot of the Pipedrive deals management interface. It shows two side-by-side deal cards. The left card is for a 'Prospect' named 'Wise deal' with a value of '10 000 € - 1 deal'. The right card is for a 'Contact established' deal with a value of '10 000 € - 1 deal', a 'Product demo' status, and a 'B' priority level. Both cards have a 'Wise deal' status indicator. The interface includes a search bar at the top right and various navigation and filtering tools on the left.

Thank you!

Any questions? Contact me!

Some code is available in my pet project:

- 🐝 github.com/Gratheon/event-stream-filter
- 🐝 github.com/Gratheon/web-app



Artjom Kurapov
artkurapov@gmail.com