



Scaling GraphQL subscriptions

Artjom Kurapov

Principal Software Engineer,
Engineering Platform

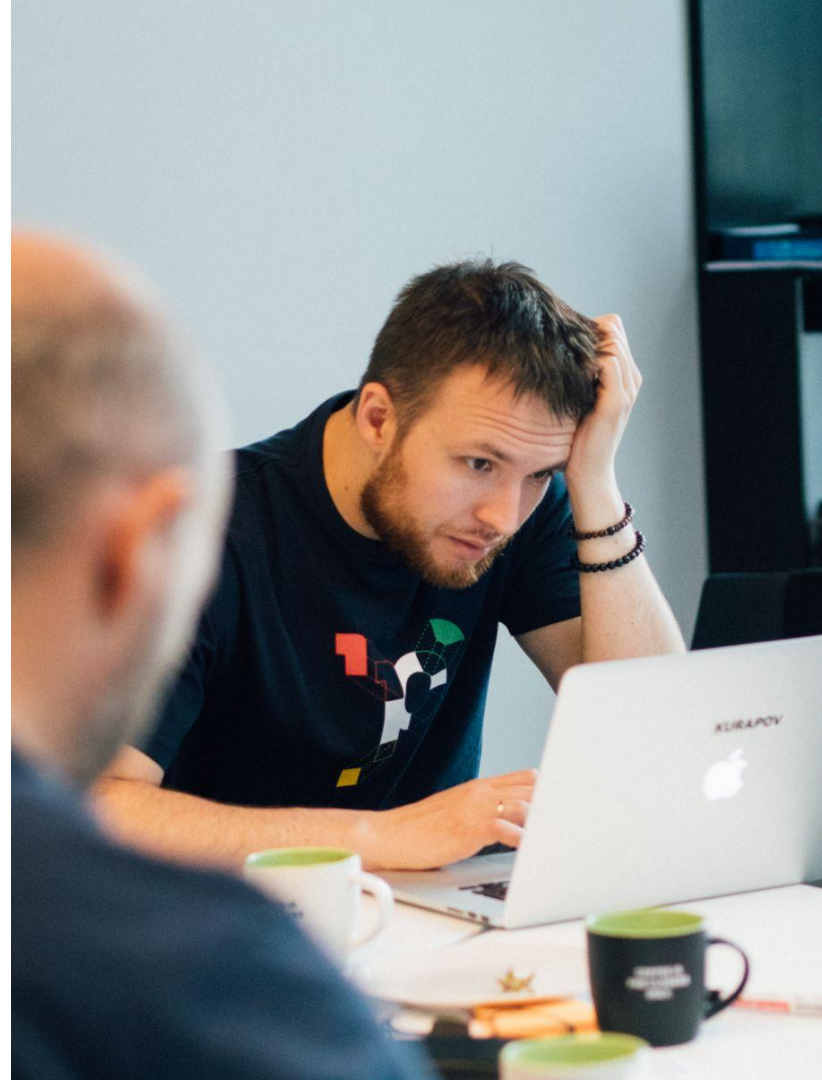
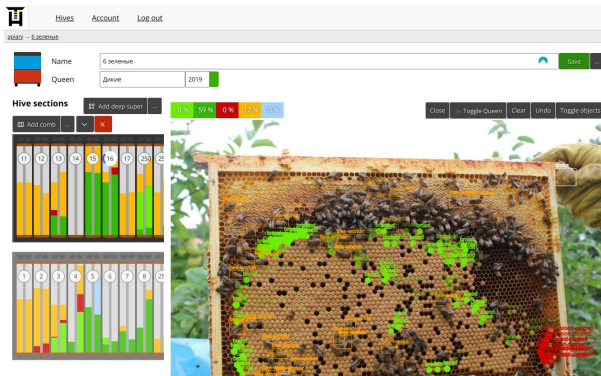
Artjom Kurapov

- 🔧 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



Agenda

Questions

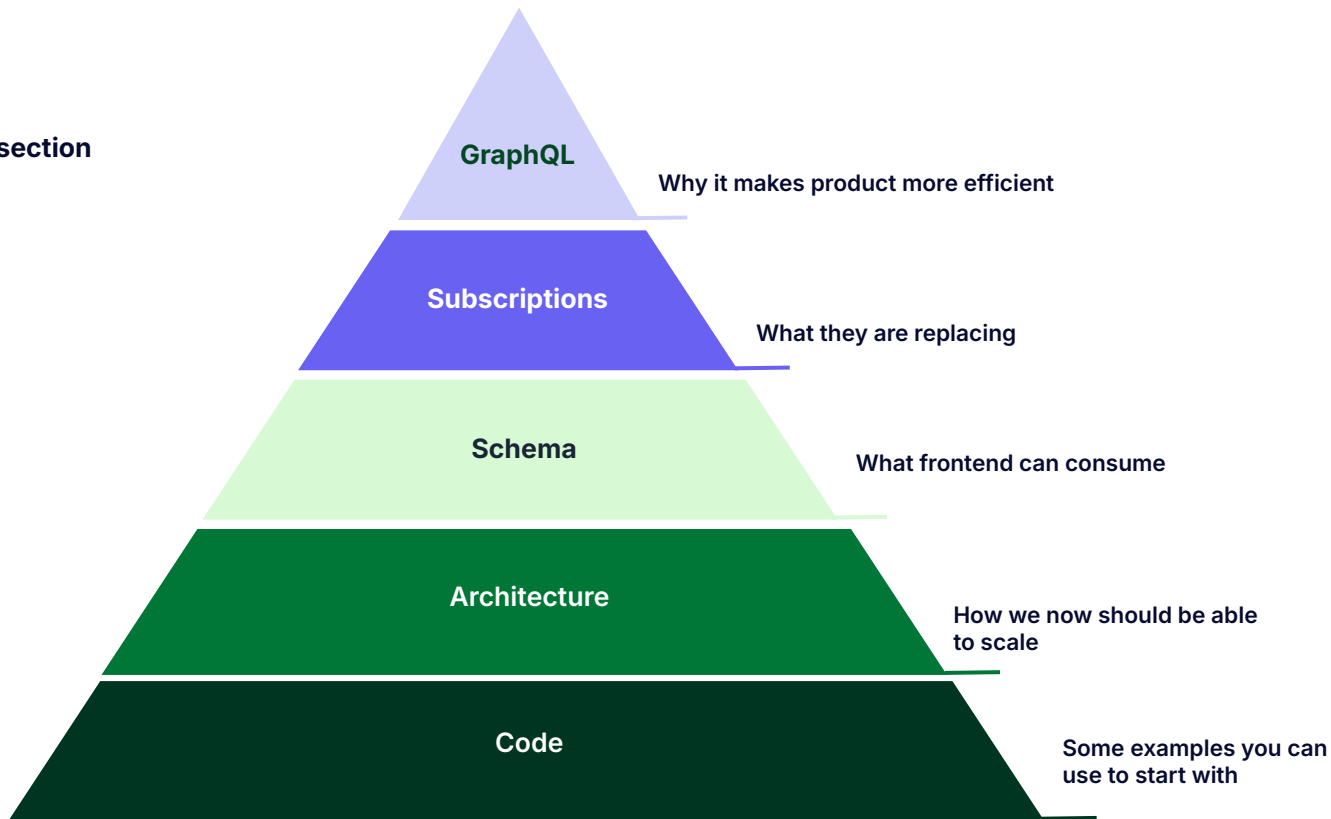
Ask at the **end of every section**

Time

ETA 50 min
~70 slides

Complexity

Expect it to grow with every section





pipedrive

600

600

FAKON
ODDANEN

SUPER
HERO

About Pipedrive

>100k

clients

400+

engineers

10

offices

30+

teams

440k

rpm at peak

730+

microservices

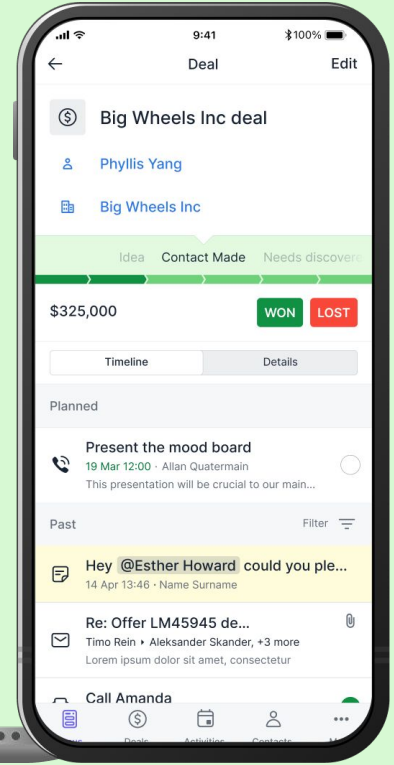
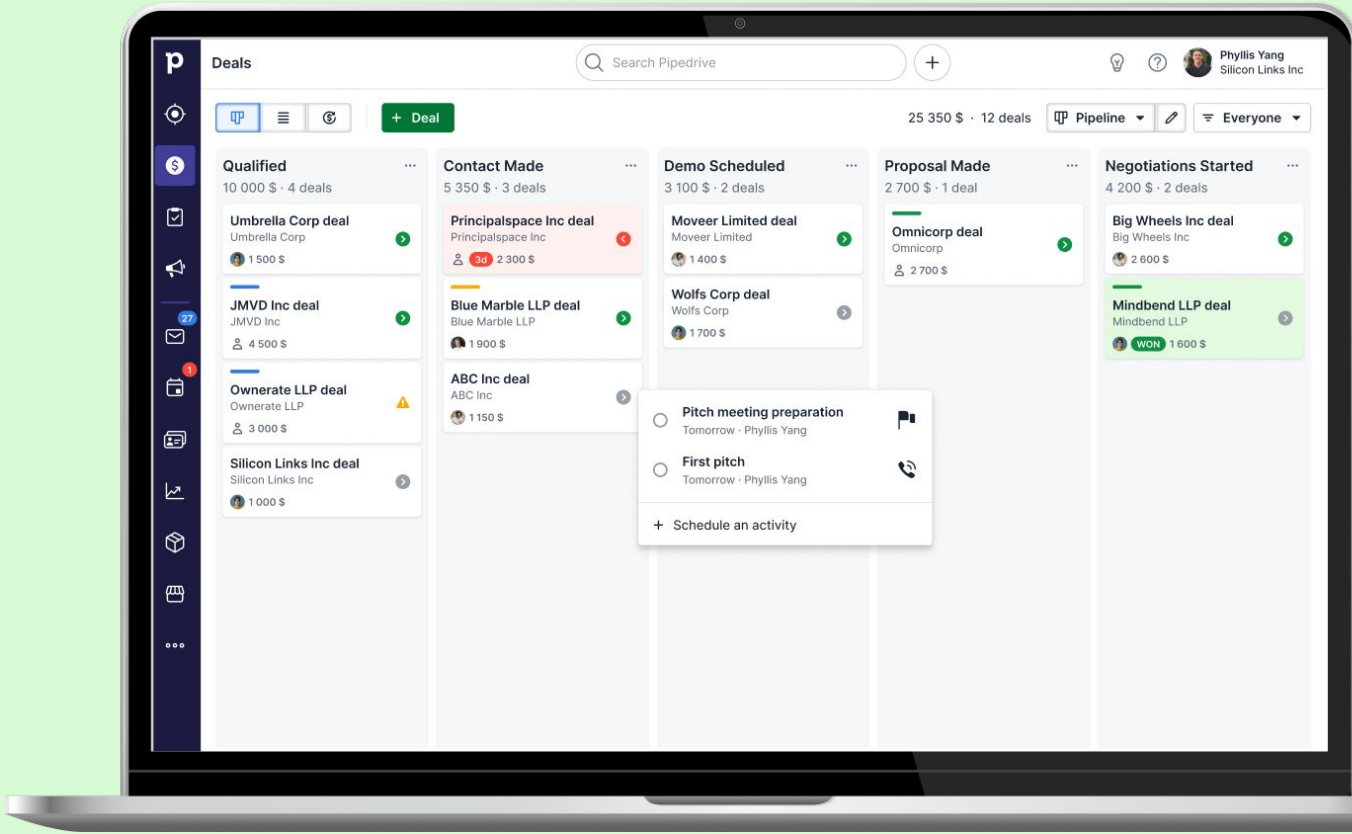
5

hosting regions

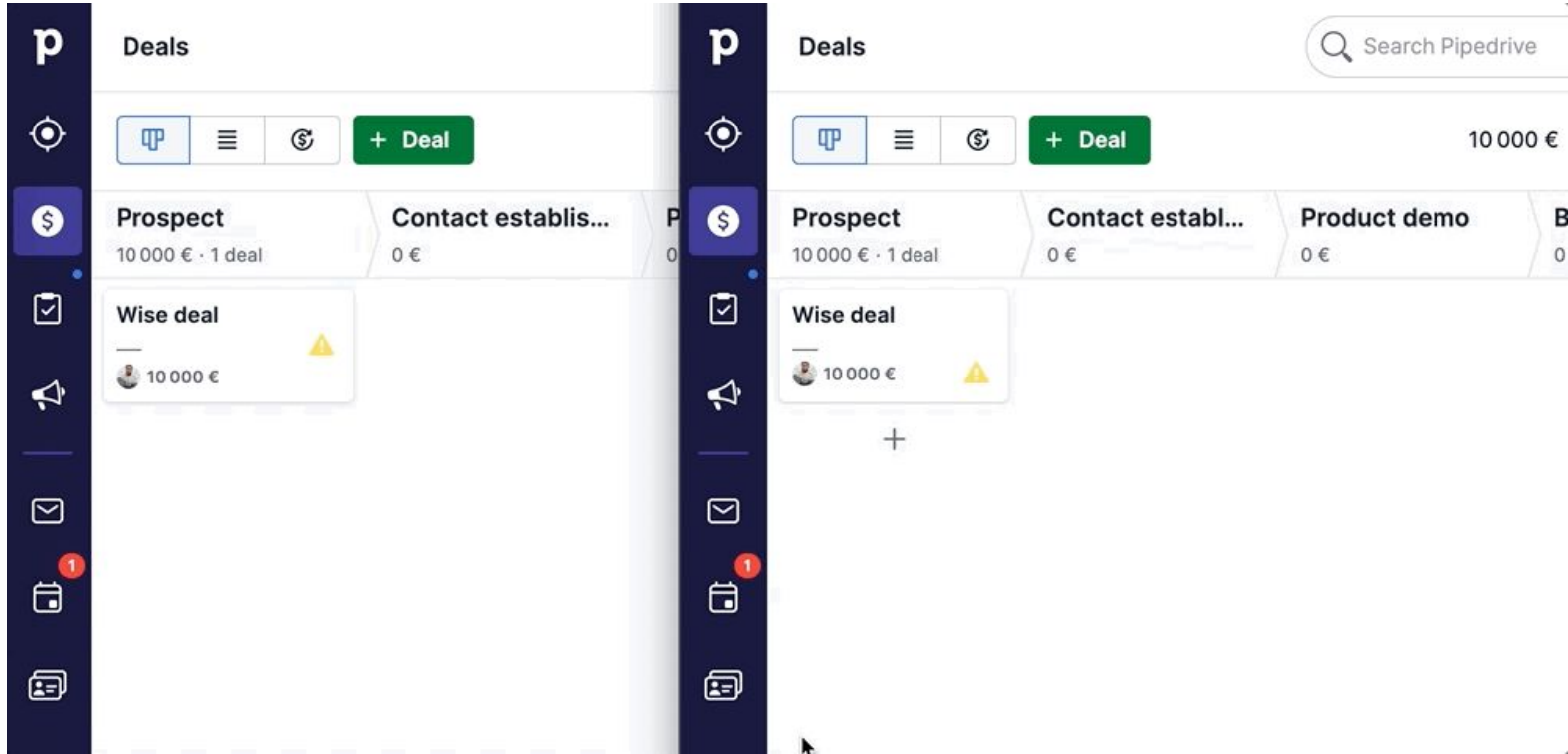
200+

libraries





Data synchronization / async event delivery



Why your business needs this?

Intuitive UX → **Simplicity** → Growth 🌳

Interactivity → **Efficiency** → Retention 🙌

Data consistency → **Trust** → Retention 🙌

How? Pusher

PUSHER
A MessageBird company

Products ▾ Developers ▾ User stories Blog Pricing ▾ Sign in [Sign up](#)

Powering **realtime** experiences for mobile and web

Bi-directional hosted APIs that are flexible, scalable and easy to use. We create and maintain complex messaging infrastructure so you can build the realtime features your users need, fast.

Get started today and find out what you can build with Pusher

[Use Cases](#) [Get your free account](#)

Pubsub **Notifications**

Publish

PHP **Node** Ruby ASP Java Python Go

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

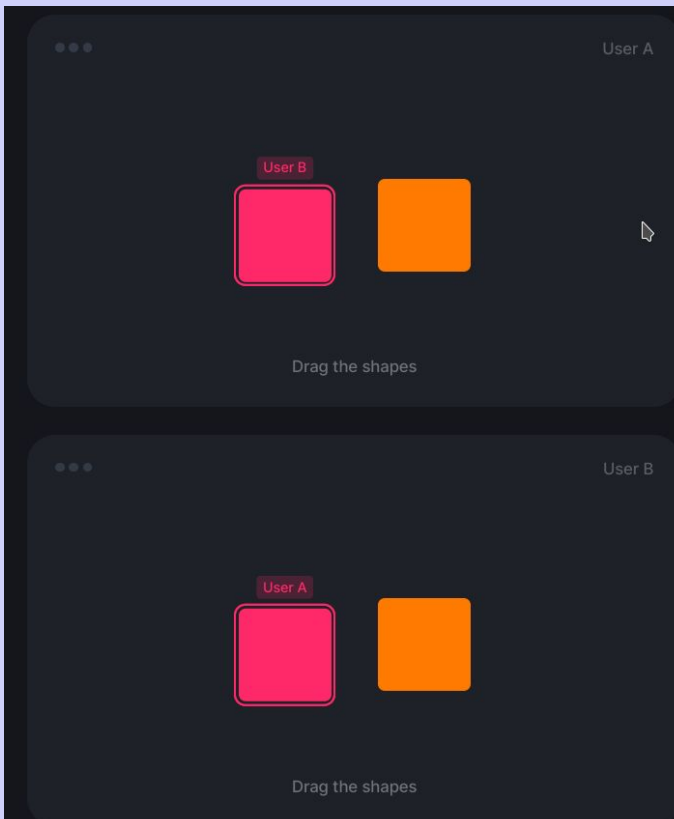
Subscribe

JS Android iOS (Swift) iOS (Obj-C)

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

[Learn more](#)

How? Liveblocks



The screenshot displays two browser windows side-by-side, representing different users in a live collaboration environment. The top window is labeled 'User A' and the bottom window is labeled 'User B'. Both windows show a dark-themed interface with a header containing three dots and the user's name. The main content area contains two colored squares: a pink square on the left and an orange square on the right. In the 'User A' window, the pink square is labeled 'User B' and the orange square is labeled 'User A', indicating that each user's presence is shown as a shape in the other's view. Below the squares, the text 'Drag the shapes' is visible. The bottom window shows the same interface but with the labels swapped: the pink square is labeled 'User A' and the orange square is labeled 'User B'. This visualizes how Liveblocks synchronizes the state of the application across multiple clients in real-time.

```
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 initialState = { 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: 9, 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 the Firebase logo, 'Products', 'Solutions', 'Pricing', 'Docs', 'Community', and 'Support'. A search bar is on the right. The left sidebar has a 'Filter' button and a list of product categories: Authentication, Realtime Database, and Cloud Firestore. Under 'Cloud Firestore', the 'Get real-time updates' link is highlighted. The main content area is titled 'Web version 9 (modular)' and includes a link to learn more about the SDK and upgrade from version 8. Below this is a code block for listening to document changes in Firestore. The code uses the modular SDK to query for cities in California and logs when they are added, modified, or removed. A file icon 'listen_diffs.js' is shown at the bottom right of the code block.

Web version 9 (modular) Web version 8 (namespaced) Swift Objective-C Java Android Kotlin+KTX Android Dart Flutter Go More ▾

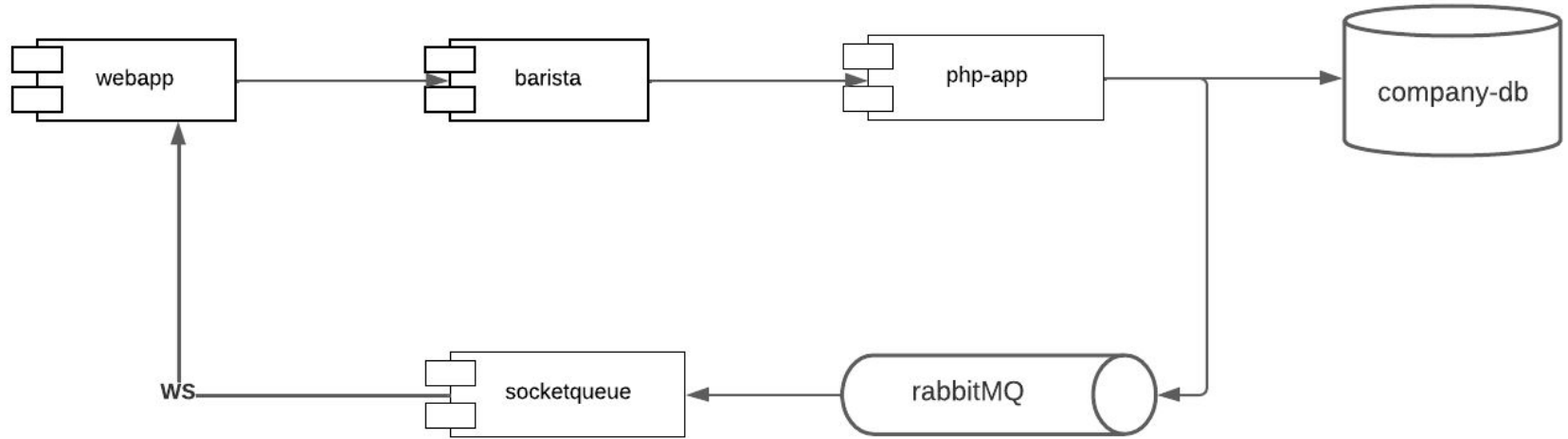
★ [Learn more](#) about the tree-shakeable Web v9 modular SDK and [upgrade](#) from version 8.

```
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());
    }
  });
});
```

[listen_diffs.js](#)

Pipedrive (oversimplified) architecture



pipedrive





Why GraphQL?



REST problems / schema documentation

DEVELOPERS go to pipedrive.com Log in Sign up

Docsguides Explore Tools Marketplace

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 /v3/filters/{id}

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

Path parameters

id	The ID of the filter
----	----------------------

INTEGER REQUIRED

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

LegacyTeams Mailbox Notes NoteFields

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

Add a new filter

POST /v3/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	The name of the filter
------	------------------------

STRING REQUIRED

conditions	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}]}]}] or [{"glue":"or","conditions":[{"CONDITION_OBJECTS}]}]. Replace CONDITION_OBJECTS with JSON objects of the following structure: {"object":"","field_id":"","operator":"","value":"","extra_value":""} or 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 NOT NULL", "IS NULL", "<<", ">>", "<", ">", "=>", "!=", "LIKE %%%", "LIKE %%%%", "NOT LIKE %%%". To get a better understanding of how filters work try creating them directly from the Pipedrive application.
------------	---

type	The type of filter to create
------	------------------------------

STRING REQUIRED

VALUES deals leads org people products activity

Response


200 OK

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

Expand all Copy code



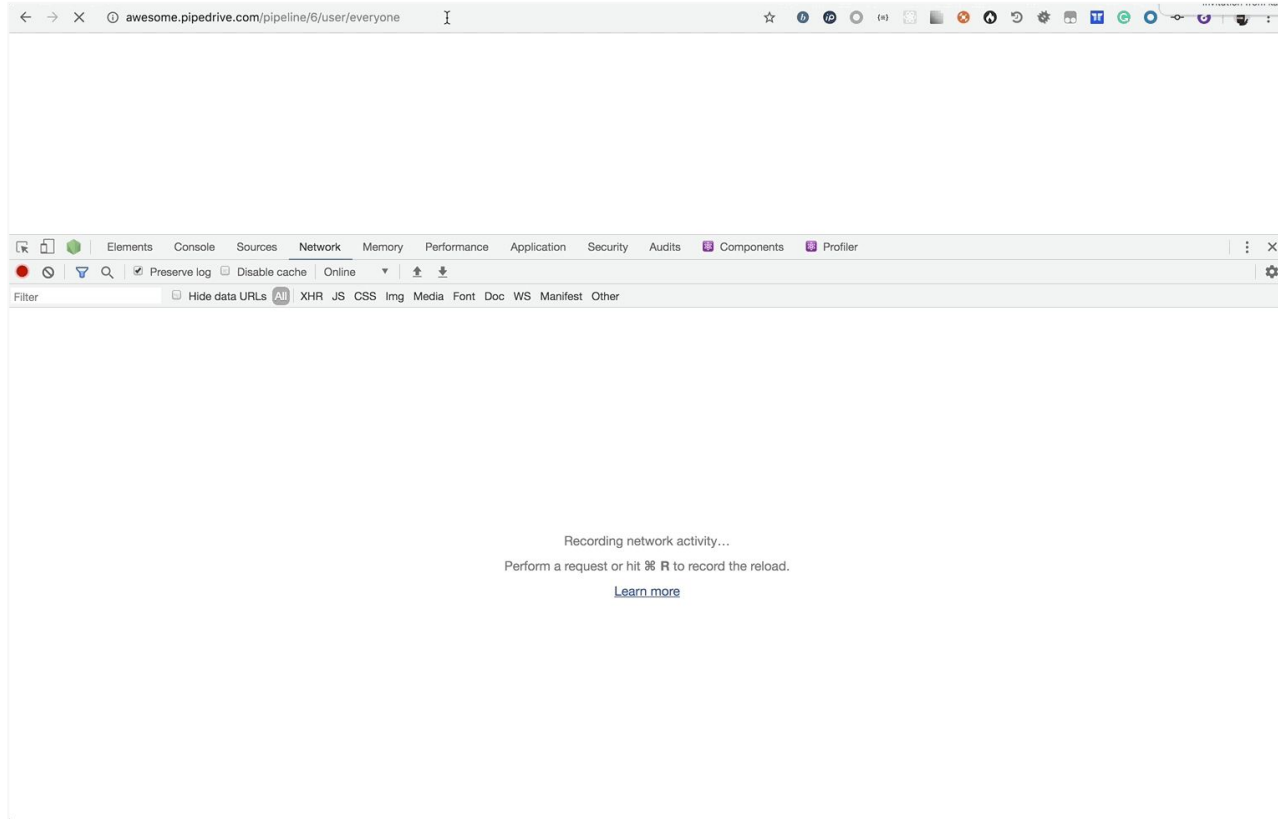
Overfetching

→   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  {
2  "success": true,
3  "data": {
4    "id": 2113502,
5    "name": "Артём Курянов",
6    "default_currency": "EUR",
7    "locale": "en_US",
8    "lang": "i",
9    "email": "artjom.kurapov@pipedrive.com",
10   "phone": null,
11   "activated": true,
12   "2fa_enabled": false,
13   "created": "2017-01-24 08:52:00",
14   "modified": "2019-10-09 08:18:01",
15   "signup_flow_variation": "short_form",
16   "has_created_company": true,
17   "is_admin": 0,
18   "company_name": "Awesome Web Design Agency",
19   "company_status": "freeloader",
20   "company_add_time": "2012-05-23 10:28:12",
21   "company_id": 48069,
22   "original_company": 0,
23   "active_flag": true,
24   "timezone_name": "Europe/Helsinki",
25   "role_id": 1,
26   "icon_url": "https://d3myhnglqW2314.cloudfront.net/profile_120x120_2113502_e3be9b279e1e0e235f305567210a153d",
27   "is_you": true,
28   "data_extra_field_limit": 50,
29   "counts": {},
30   "current_company_features": {},
31   "company_size": null,
32   "current_user_settings": {},
33   "companies": {},
34   "current_company_plan": {},
35   "current_company_mrr": 27423,
36   "connections": {},
37   "fields": {},
38   "stages": [],
39   "currencies": [],
40   "custom_views": [],
41   "activity_types": [],
42   "dialog_additional_fields": {},
43   "language": {},
44   "global_messages": [],
45   "cc_email": "",
46   "alternative_emails": [],
47   "pipelines": [],
48   "started_paying_time": "2013-01-14 23:13:40",
49   "seconds_to_trial_ends": 256745107,
50   "add_user_trial_extended_flag": false,
51   "intercom_user_hash": "",
52   "seats_quota": 277,
53   "user_count": 426,
54   "company_country": "EE",
55   "recurring_coupons": "gold_signup_monthly",
56   "current_company_test_cases": [],
57   "timezone_offset": 10800,
58   "3rd_party_auth_links": {},
59   "wgm_active": true
60  },
61  "additional_data": {}
62 }
```

Multiple API composition endpoints

https://app.pipedrive.com/menu-waitress/v2

```
{
  - menus: {
    + primary: [ ... ],
    + detached: [ ... ],
    + secondary: [ ... ],
    + more: [ ... ]
  },
  rootUrl: "/",
  + hiddenPaths: [ ... ],
  - user: {
    pic_url: "https://d3myhnlqkw2314.cloudfront.net/profile_1",
    name: "Artjom Kurapov",
    language: "en-US",
    email: "artjom.kurapov@pipedrive.com",
    id: "354645",
    + frootMenuState: [ ... ]
  },
  + company: { ... },
  + services: [ ... ],
  cacheVersion: "de7f35eb2b_2741",
  currentVersion: "de7f35eb2b_2741",
  + redirects: [ ... ],
  + blacklist: { ... },
  + viewSelects: [ ... ]
}
```

pipedrive

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: "egregreg",
        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: [
```


API composition mission

Make API efficient



Artjom Kurapov



Erik Schults

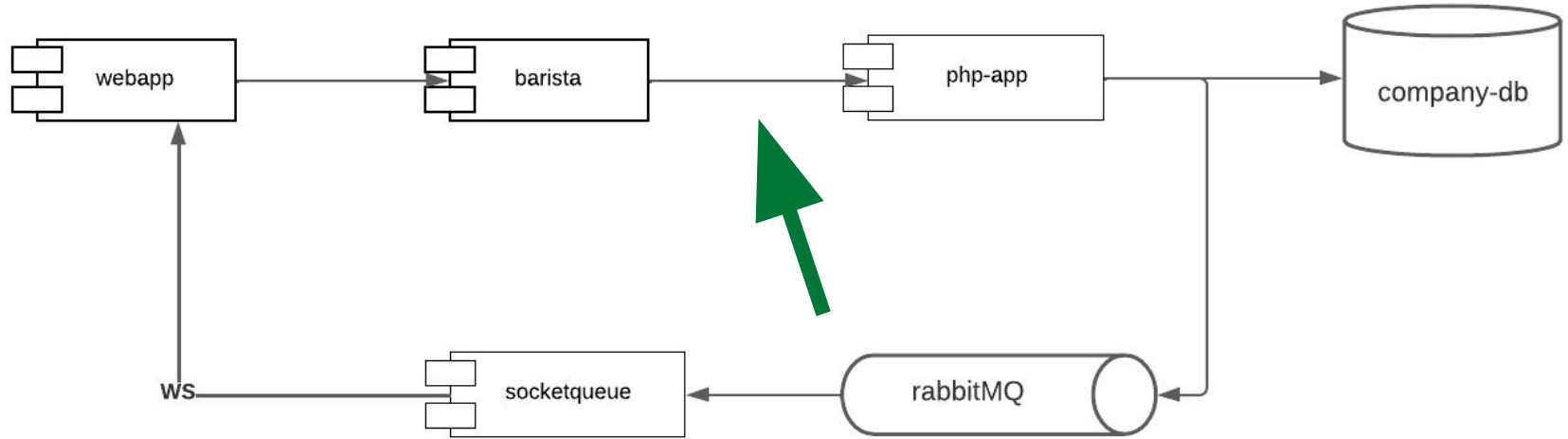


Oleksandr Shvechykov



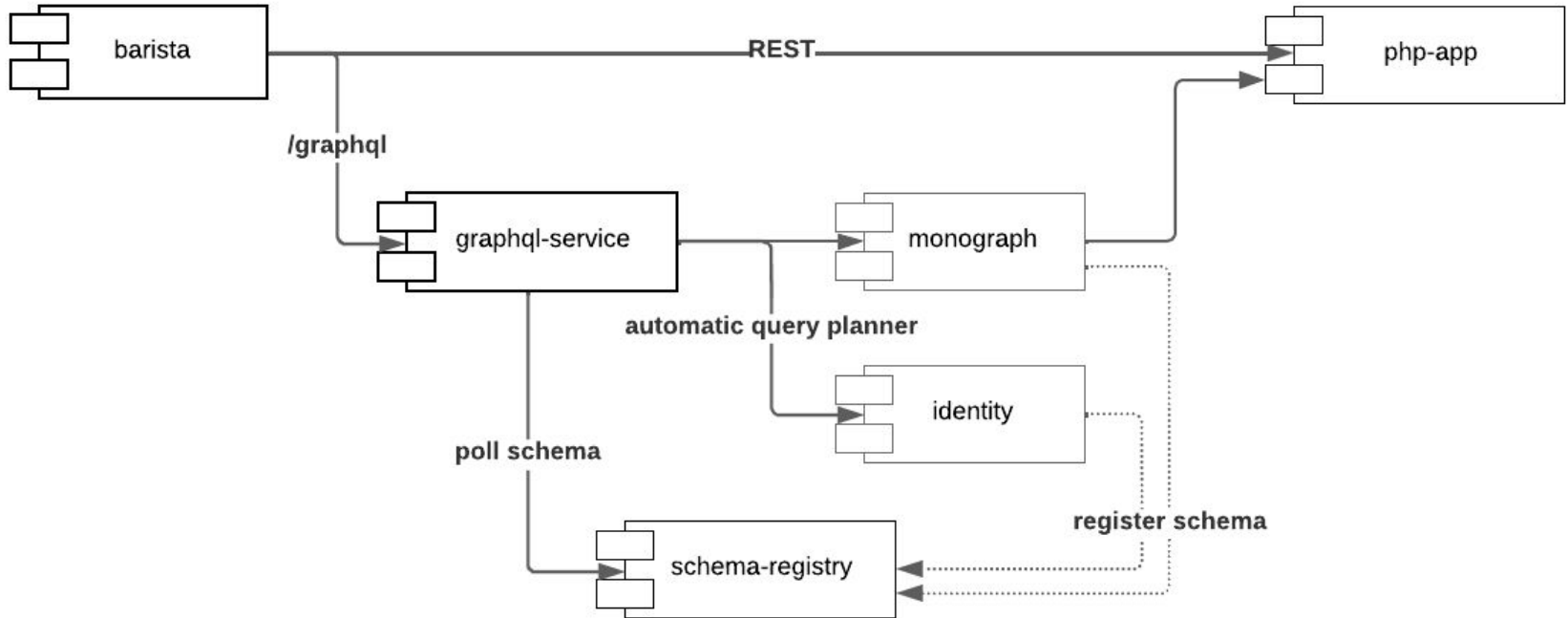
Aleksander Gasna

Add GraphQL here





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

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

SDL DIFF USAGE CONTAINERS (8)

Entity	Total hits
Query.hive	3
Query.apiary	
Query.apiaries	2
Query.inspection	
Mutation.addApiary	
Mutation.updateApiary	
Mutation.addHive	
Mutation.updateHive	
Mutation.deactivateHive	
Mutation.addInspection	
Mutation.uploadFrameSide	
Apiary.id	2
Apiary.name	2
Apiary.hives	77
Apiary.location	
Hive.id	5
Hive.name	5

Client Hits Day

Client Hits	Day
25	2022-07-14
50	2022-07-15
2	2022-07-16

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)

The image displays a GraphQL IDE interface with three main components:

- Queries Panel:** Shows a query for a user and their company, with a dropdown menu for the `id` field.
- Schema Panel:** Shows the schema for the `activities` field, including a description and a list of custom field implementations.
- Diagram:** A central diagram showing the relationships between various custom field implementations and their parent types.

```
query {
  user {
    name
  }
  company {
    country
  }
}
```

Schema:

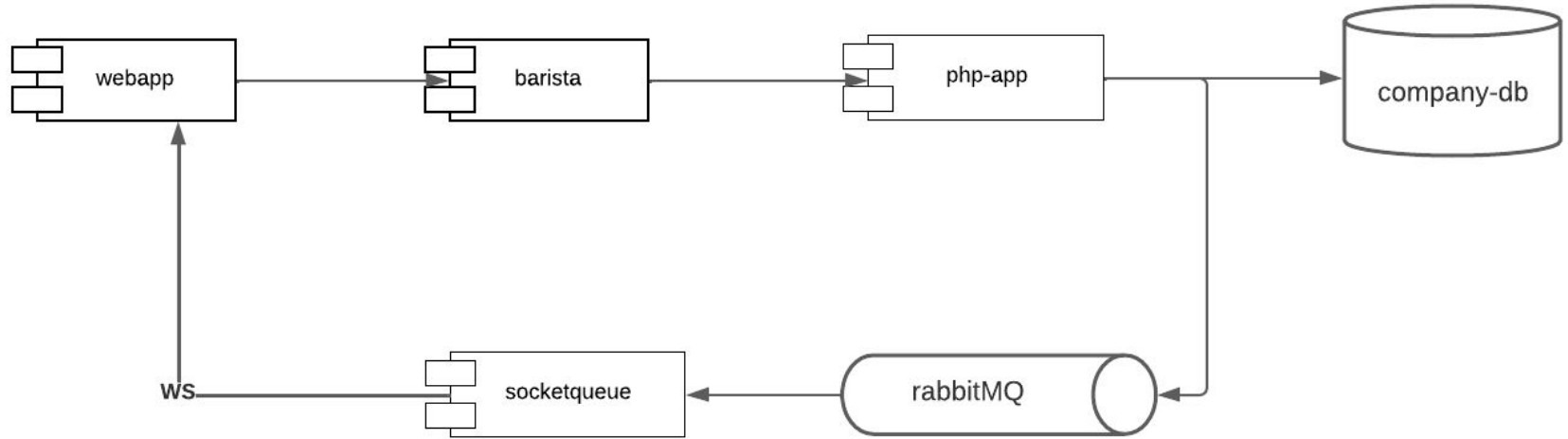
```
activities(
  userid: ID
  leadId: ID
  first: Int
  after: Cursor = ""
  done: ActivitiesDoneArg
): ActivitiesConnection

Returns activities either by user id (opaque string) or lead id (opaque string)
```

Custom Field Implementations:

- `CustomFieldText`
- `CustomFieldLongText`
- `CustomFieldNumeric`
- `CustomFieldPhone`
- `CustomFieldTime`
- `CustomFieldTimeRange`
- `CustomFieldDate`
- `CustomFieldDateRange`
- `CustomFieldSingleOption`
- `CustomFieldMultiOption`
- `CustomFieldPerson`
- `CustomFieldNote`

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 a Pipedrive 'Deals' page with a network inspector open. The network inspector displays a list of 'websocket' messages. The messages are highly repetitive, showing a denormalized and undocumented schema. The schema is a JSON object with a 'meta' field containing an 'action' of 'updated' and an 'object' field containing 'person' and 'company' information. The 'person' field includes 'id', 'company_id', 'user_id', and 'host'. The 'company' field includes 'id', 'company_id', 'user_id', and 'host'. The 'meta' field also includes 'action' and 'object'.

Name	Headers	Messages	Initiator	Timing
5-80DghzU4QYQS9L2-pKWJwf-O8oe4ZvyxFkX...	All	Enter regex, for example: (web)?socket		
websocket				

Data	Length	Time
<code>{\"meta\":{\"action\":\"updated\",\"object\":{\"person\":{\"id\":\"32975\",\"company_id\":\"48069\",\"user_id\":\"1551669\",\"host\":\"app.pipedrive.com\"},\"time...</code>	16931	14:00:37.780
<code>{\"meta\":{\"action\":\"updated\",\"object\":{\"person\":{\"id\":\"32862\",\"company_id\":\"48069\",\"user_id\":\"1551669\",\"host\":\"app.pipedrive.com\"},\"time...</code>	17343	14:00:37.782
<code>{\"meta\":{\"action\":\"updated\",\"object\":{\"person\":{\"id\":\"33829\",\"company_id\":\"48069\",\"user_id\":\"1551669\",\"host\":\"app.pipedrive.com\"},\"time...</code>	17238	14:00:37.798
<code>{\"meta\":{\"action\":\"updated\",\"object\":{\"person\":{\"id\":\"36493\",\"company_id\":\"48069\",\"user_id\":\"1551669\",\"host\":\"app.pipedrive.com\"},\"time...</code>	16768	14:00:37.861
<code>{\"meta\":{\"action\":\"updated\",\"object\":{\"person\":{\"id\":\"35558\",\"company_id\":\"48069\",\"user_id\":\"1551669\",\"host\":\"app.pipedrive.com\"},\"time...</code>	17349	14:00:37.921
<code>{\"meta\":{\"action\":\"updated\",\"object\":{\"person\":{\"id\":\"32862\",\"company_id\":\"48069\",\"user_id\":\"5866726\",\"host\":\"app.pipedrive.com\"},\"time...</code>	17362	14:00:37.990
<code>{\"meta\":{\"action\":\"updated\",\"object\":{\"person\":{\"id\":\"22049\",\"company_id\":\"48069\",\"user_id\":\"1551669\",\"host\":\"app.pipedrive.com\"},\"time...</code>	17721	14:00:38.006
<code>{\"meta\":{\"action\":\"updated\",\"object\":{\"person\":{\"id\":\"33829\",\"company_id\":\"48069\",\"user_id\":\"1551669\",\"host\":\"app.pipedrive.com\"},\"time...</code>	17239	14:00:38.245
<code>{\"meta\":{\"action\":\"updated\",\"object\":{\"person\":{\"id\":\"32975\",\"company_id\":\"48069\",\"user_id\":\"1551669\",\"host\":\"app.pipedrive.com\"},\"time...</code>	16913	14:00:38.264
<code>{\"meta\":{\"action\":\"updated\",\"object\":{\"person\":{\"id\":\"36493\",\"company_id\":\"48069\",\"user_id\":\"1551669\",\"host\":\"app.pipedrive.com\"},\"time...</code>	16769	14:00:38.303
<code>{\"meta\":{\"action\":\"updated\",\"object\":{\"person\":{\"id\":\"32862\",\"company_id\":\"48069\",\"user_id\":\"1551669\",\"host\":\"app.pipedrive.com\"},\"time...</code>	17363	14:00:38.325
<code>{\"meta\":{\"action\":\"updated\",\"object\":{\"person\":{\"id\":\"35558\",\"company_id\":\"48069\",\"user_id\":\"1551669\",\"host\":\"app.pipedrive.com\"},\"time...</code>	17350	14:00:38.334
<code>{\"meta\":{\"action\":\"updated\",\"object\":{\"person\":{\"id\":\"22049\",\"company_id\":\"48069\",\"user_id\":\"1551669\",\"host\":\"app.pipedrive.com\"},\"time...</code>	17741	14:00:38.344

timestamp\":\"1621249238\",\"timestamp_micro\":\"1621249238216019\",\"request_timestamp\":\"1621249238147\",\"permitted_user_ids\":[10257755,9825558,10105330,149778,2743084,1277750,100677

Problems - scalability

Name

- websocket
- info?t=1601493940673

× Headers Messages Initiator Timing

▼ 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

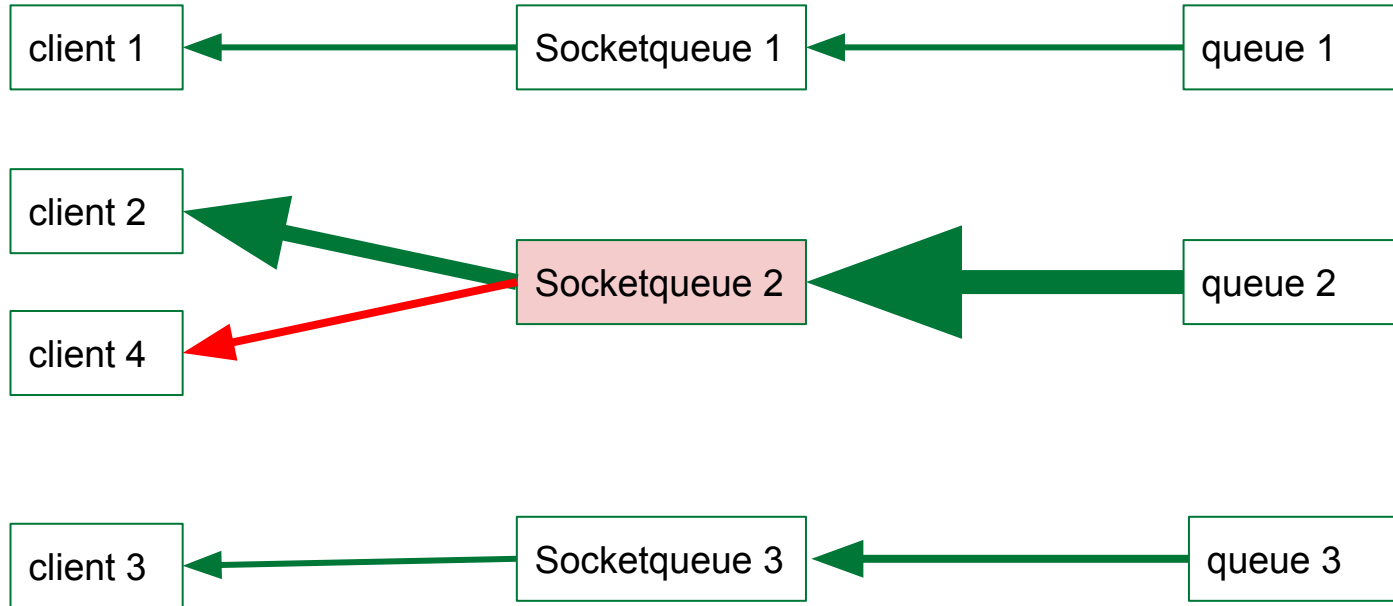
Cache-Control: no-cache

Connection: Upgrade

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

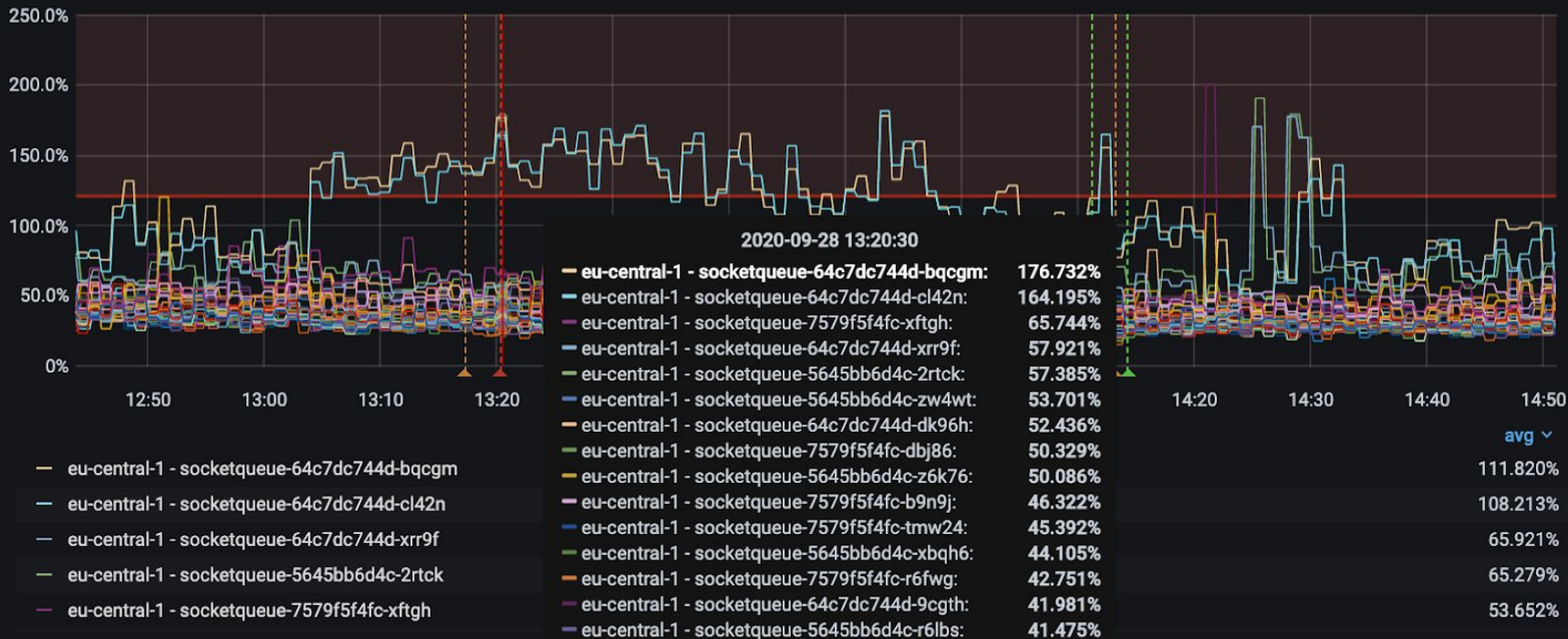
Origin: https://awesome.pipedrive.com

Problems - scalability with fixed queues



Problems - noisy neighbour → CPU → infra waste

CPU usage - Top 5 in eu-central-1 by container



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



The screenshot shows a web browser window with a GraphQL IDE. The address bar shows the URL `https://graphql-subscriptions.pipedrive.dev/graphql`. The query editor contains the following query:

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

The response is displayed in a tree view, with a red circle highlighting the root object. The JSON response is as follows:

```
{
  "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,
    }
  }
}
```

At the bottom of the IDE, there are tabs for "QUERY VARIABLES", "HTTP HEADERS", "TRACING", and "QUERY PLAN". A "Listening" indicator is visible near the response.

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**

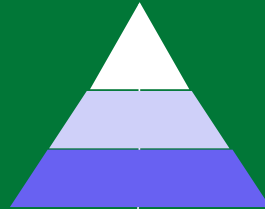


- Ended up still with nodejs:
apollosolutions/federation-subscription-tools
enisdenjo/graphql-ws

Dynamic federated schema merging



Schema



Mini demo

The image shows a split-screen view. On the left is a code editor window titled "Subscriptions" with a play button and buttons for "Prettify", "Merge", "Copy", and "History". It contains a JSON snippet:

```
1 subscription{
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 }
```

On the right is a CRM interface titled "Deals" with a search bar "Search Pipedrive". It features a "+ Deal" button and a navigation bar with "Prospect" (0 €), "Contact establis..." (10 000 € · 1 deal), and "Product demo" (0 €). A card titled "Wise deal" shows a user profile and "10 000 €" with a yellow warning triangle.

Subscriptions

Deals

Search Pipedrive

+ Deal

Prospect 0 €

Contact establis... 10 000 € · 1 deal

Product demo 0 €

Wise deal

10 000 €

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  
}
```

GraphiQL



Prettify

Merge

Copy

History

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



```
{  
  "data": {  
    "dealChanged": {  
      "delta": {  
        "updateTime":  
        "wonTime":  
        "closeTime":  
        "status":  
      }  
    }  
  }  
}
```

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**)

pipedrive



```
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
}
```

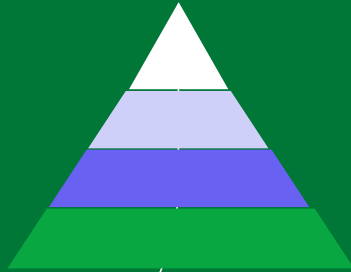
Live queries

- Query + Subscribe
- Async iterator magic

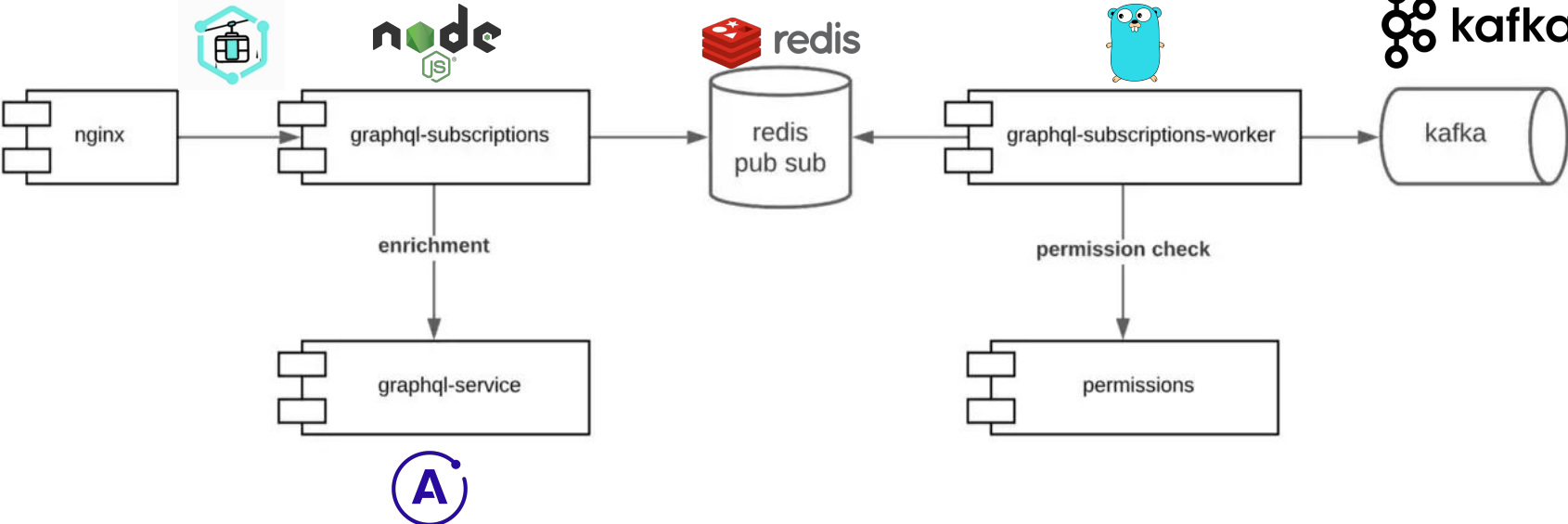
```
1 ▾ subscription{  
2   dealChanged(ids: ["436"], liveQuery: true){  
3     enriched{  
4       title  
5       creator{  
6         name  
7       }  
8     }  
9   }  
10 }
```



Architecture


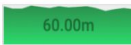






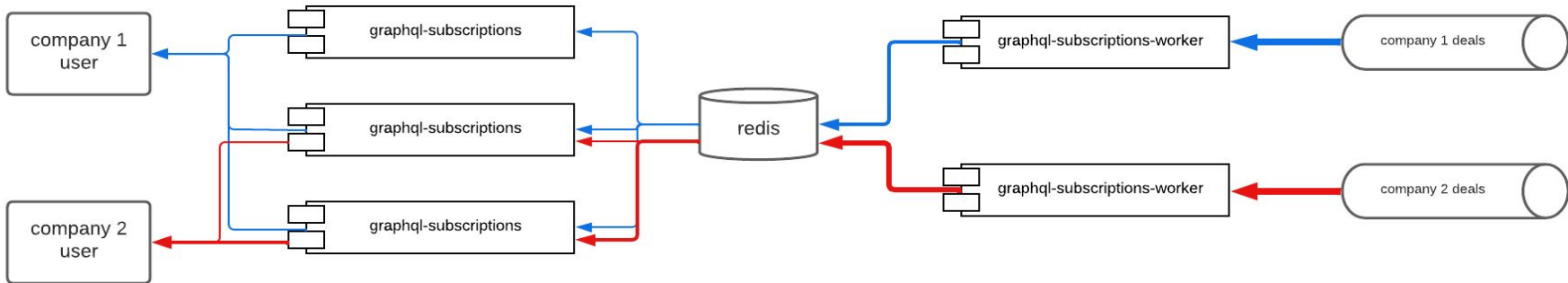
Services



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**

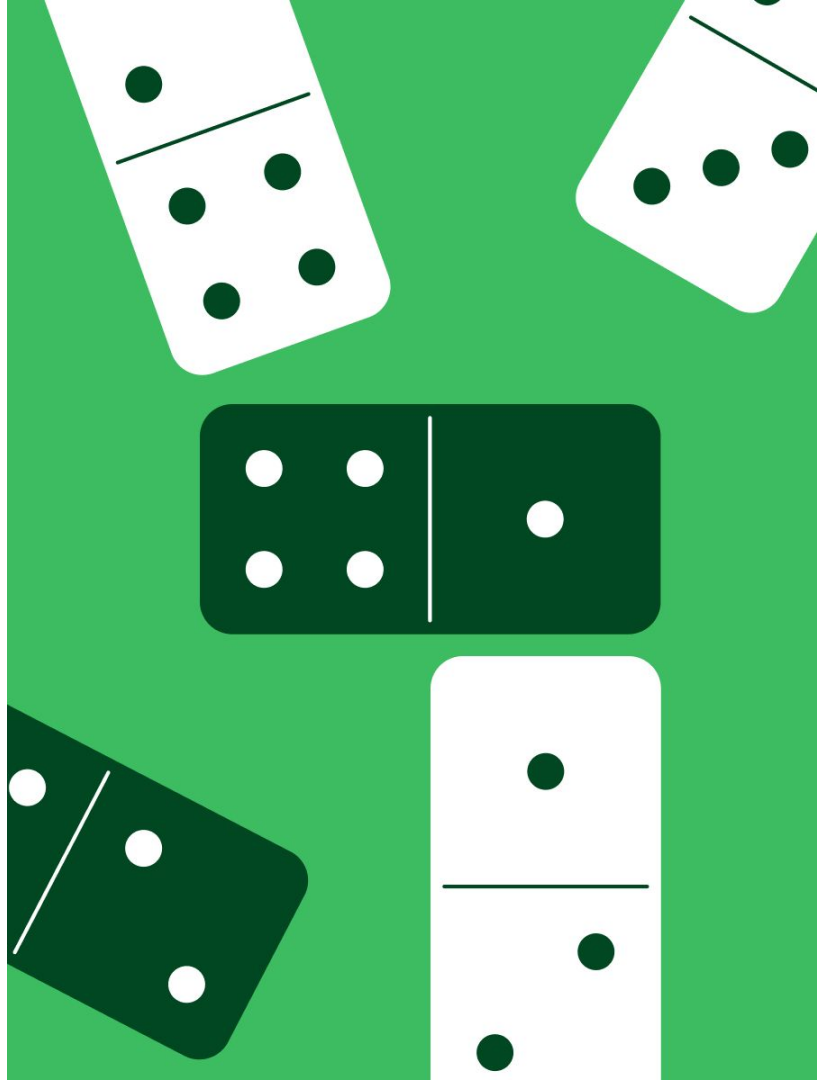

Pods							
Name	Namespace	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created ↑	
 graphql-subscriptions-7b85f4d65f-mxsdx	graphql	Running	0	 60.00m	 481.01Mi	3 days ago	
 graphql-subscriptions-worker-d4bfc7b6d-tb2ps	graphql	Running	0	 41.00m	 59.03Mi	3 days ago	



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`

Company id
Entity type
Entity id



Performance testing

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

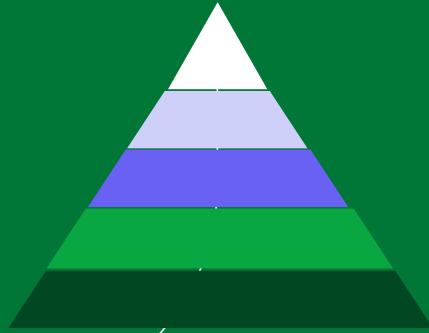


Limitations

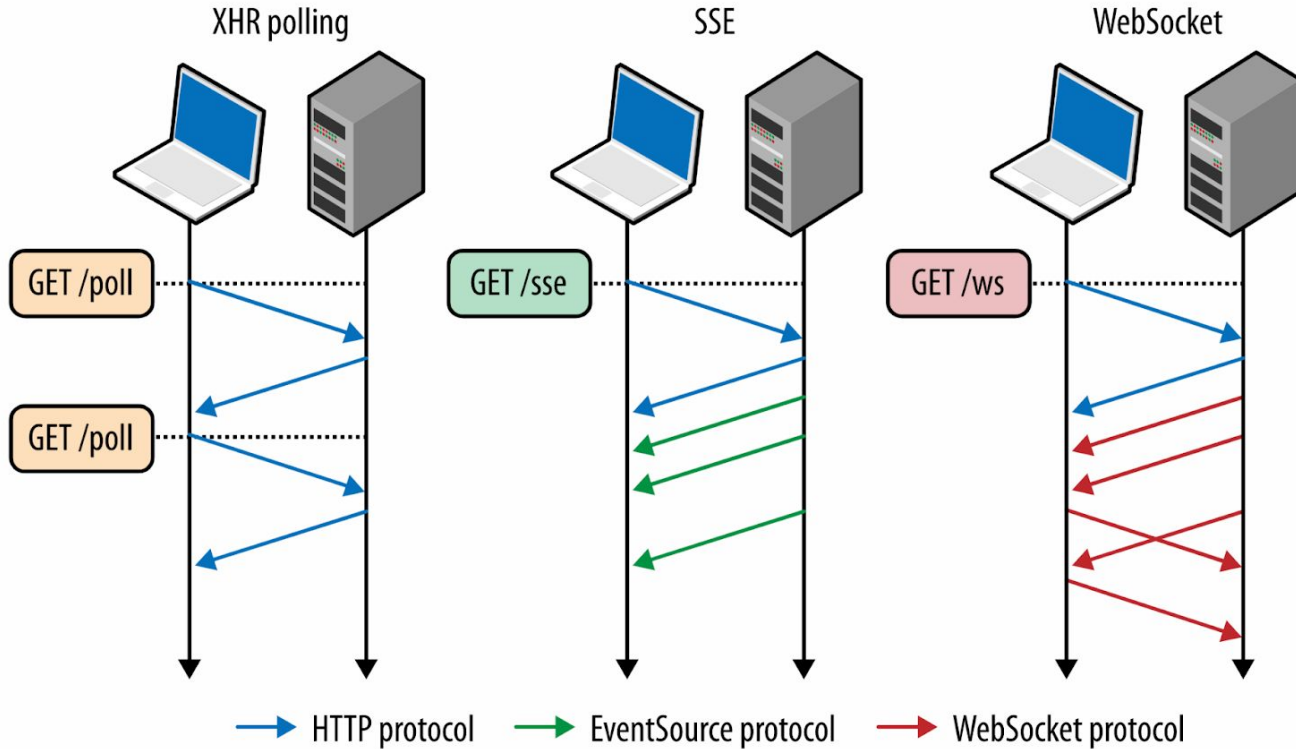
- Max amount of IDs per subscription
 - Redis channels per subscription
- Connection per user per pod
 - Subscriptions per connection
- Subscriptions per pod
- Enrichment request timeout
- Connection time to live
- Disconnect users
 - On logout
 - On permission change
- Automatic reconnect (by graphql-ws)



Code



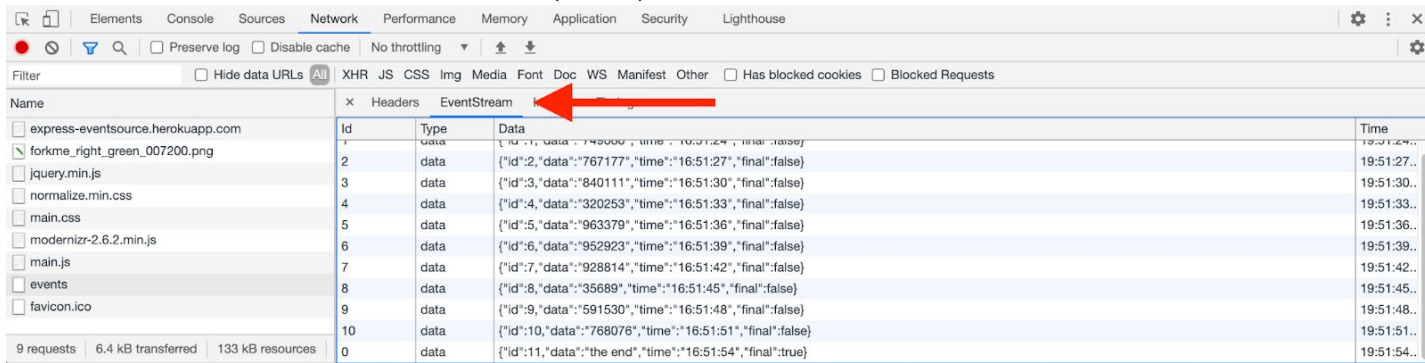
Transport



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);
```



Name	Id	Type	Data	Time
express-eventsource.herokuapp.com	1	data	{\"id\":\"1\",\"data\":\"745000\",\"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..
	0	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=${event.reason}`);
16   } else {
17     // e.g. server process killed or network down
18     // event.code is usually 1006 in this case
19     alert(`[close] Connection died!`);
20   }
21 };
```

Bit	+0..7	+8..15	+16..23	+24..31
0	FIN	Opcode	Mask	Length
32	Extended length (0–8 bytes) ...			
64	...		Masking key (0–4 bytes) ...	
96	...		Payload ...	
...	...			

The screenshot shows a browser's developer console with the 'Messages' tab selected. The messages are as follows:

- o
- a["[messageText]: Connection opened. socketqueue-d854cff97-9fsqd; 24c924fa6d26bfc27cc5e27bed40461c"]
- a["[messageText]: subscribed"]
- ↑ [{"meta":{"company_id":48069,"user_id":2113502,"user_name":"Артём Купанов","host":"awesome.pipedrive.com","timestamp":1620322592...}}
- a["[rabbitStateChange]: open", "messageText": "Connection to RabbitMQ opened"]
- a["[messageText]: Started listening for company 48069; my user 2113502", "routingKey": "auth", "auth": true]
- a["[meta":{"v":1,"action":"updated", "object":{"userCounts":{"id":2113502,"company_id":48069,"user_id":2113502,"host":"awesome.pipedri...}}
- a["[meta":{"v":1,"action":"updated", "object":{"userSetting":{"id":2113502,"company_id":48069,"user_id":2113502,"host":"awe...}}
- a["[meta":{"v":1,"action":"added", "object":{"proactive_card":{"id":28496532,"company_id":48069,"user_id":2113502,"host":"app.pipedrive...}}

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

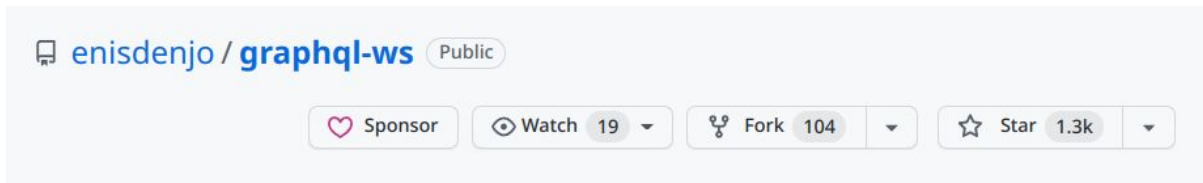
👁 Watch 72 ▾

🍴 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");
});
```

```
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
```

```
);
```

```
});
```

```
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";
```

Subscription (with generator function)




```
1 ▾ subscription{  
2   hi  
3 }
```



+ GraphQL

```
// 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)



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



+ GraphQL

```
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"  
  }  
}
```

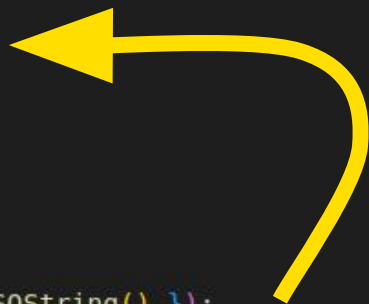
```
import { PubSub } from "graphql-subscriptions";

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';
```

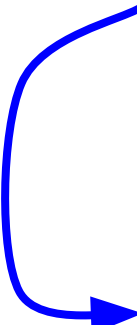
```
useServer({  
  execute, // from 'graphql'  
  subscribe, // from 'graphql'  
  context: (ctx) => contextHandler(ctx),  
  onConnect: (ctx) => connectHandler(ctx; fastify, redisSub),  
  onDisconnect: (ctx: any) => {},  
  onClose: (ctx: any) => {},  
  onSubscribe: (ctx, msg) => subscribeHandler(ctx,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

A blue arrow originates from the 'subscribe' property of the 'dealAdded' object and points to the 'resolve' property of the same object, indicating a relationship or flow between these two methods.

Pipedrive engineering blog



Artjom Kurapov

Sep 9, 2020 · 13 min read · Member-only · Listen



Journey to a Federated GraphQL

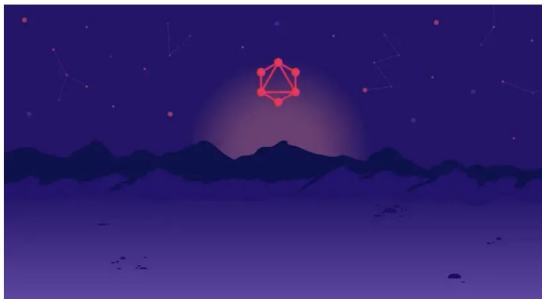
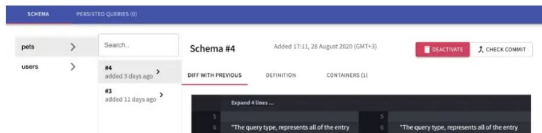


Image taken from dgraph.io

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](#).]

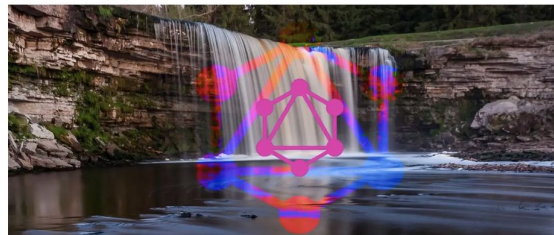


Artjom Kurapov

Nov 23 · 11 min read · Listen

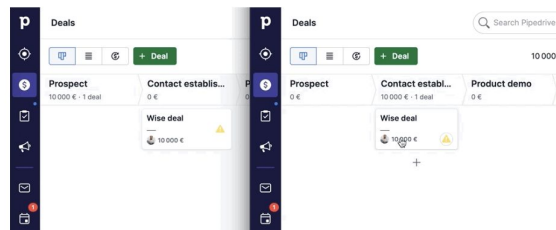


A dream of scalable and enriched GraphQL subscriptions



A stylised photo of Jagala juga (Estonia), original photo by Aleksandr Abrosimov, Wikimedia Commons

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.



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