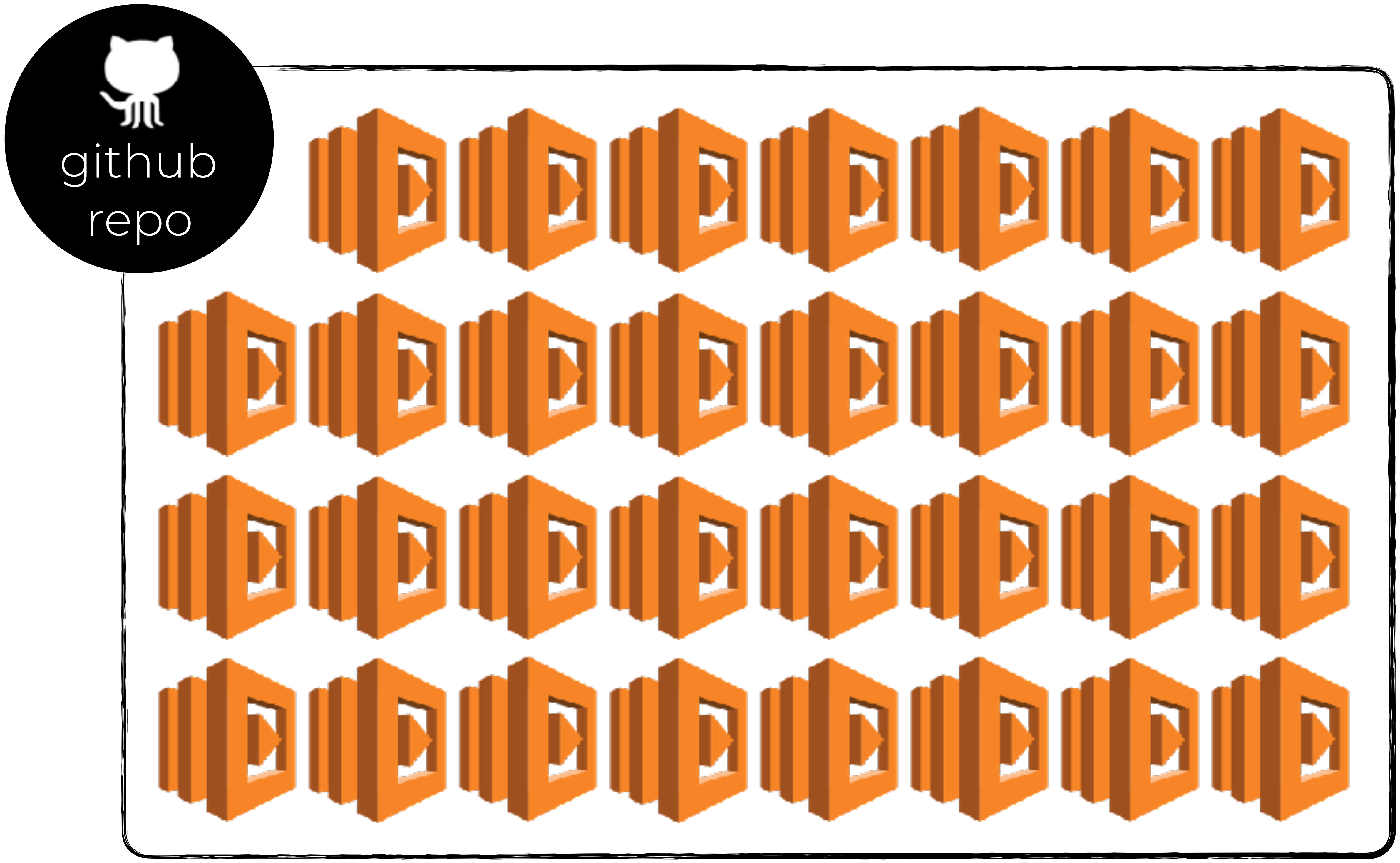
organizing functions

“how do I organize my functions into code repositories?”

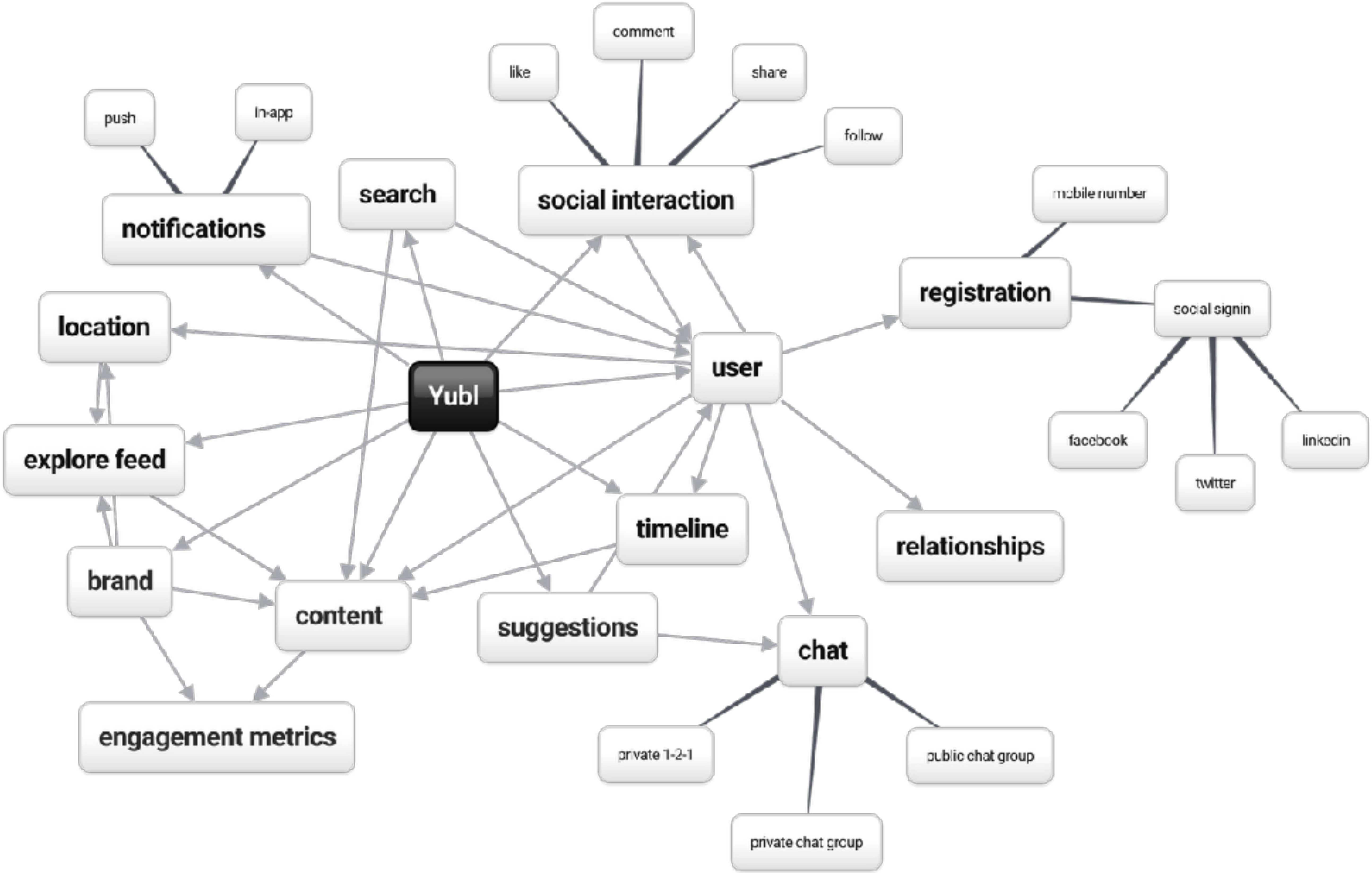
monolithic



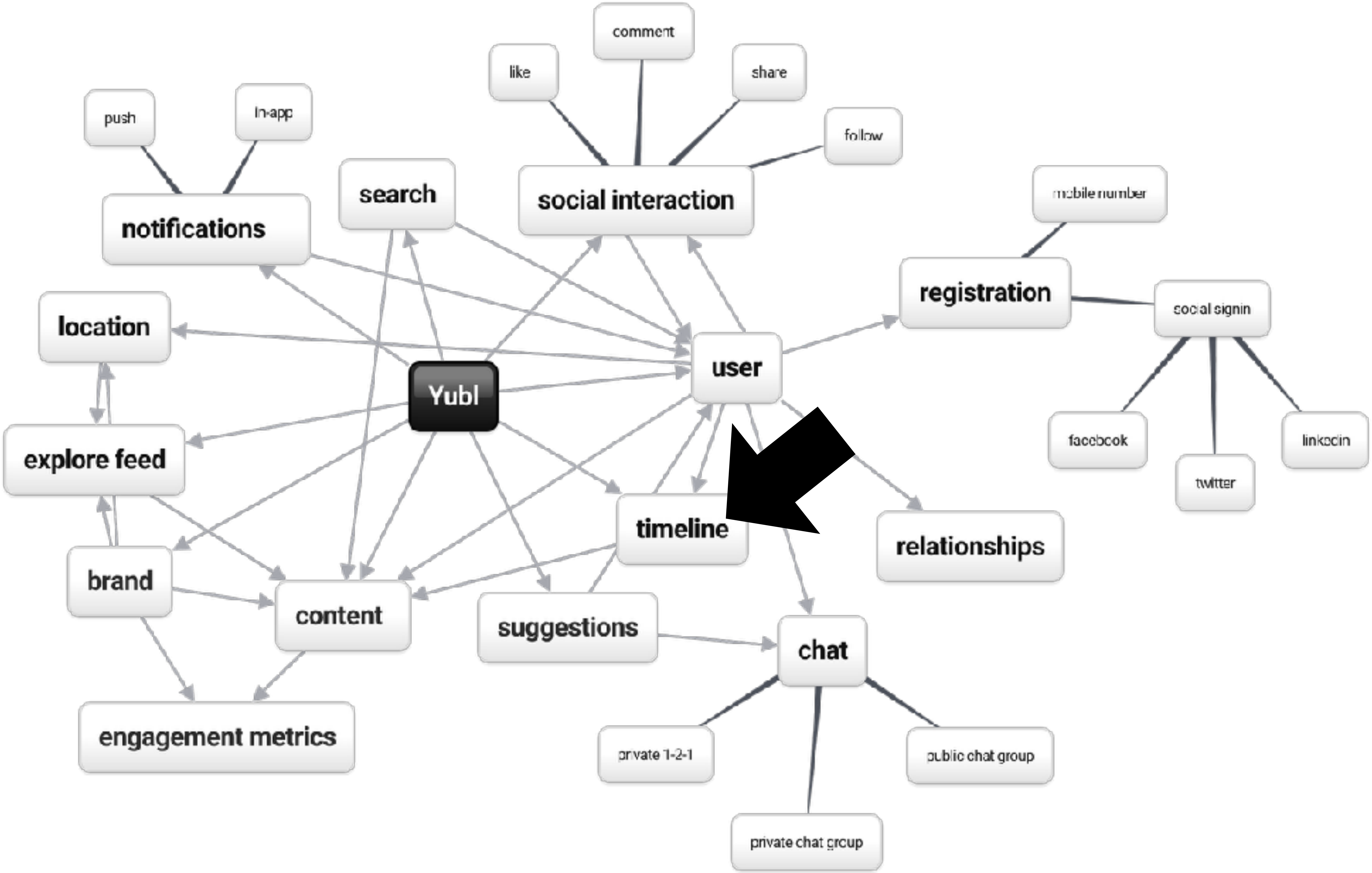
monolithic

don’t do it…

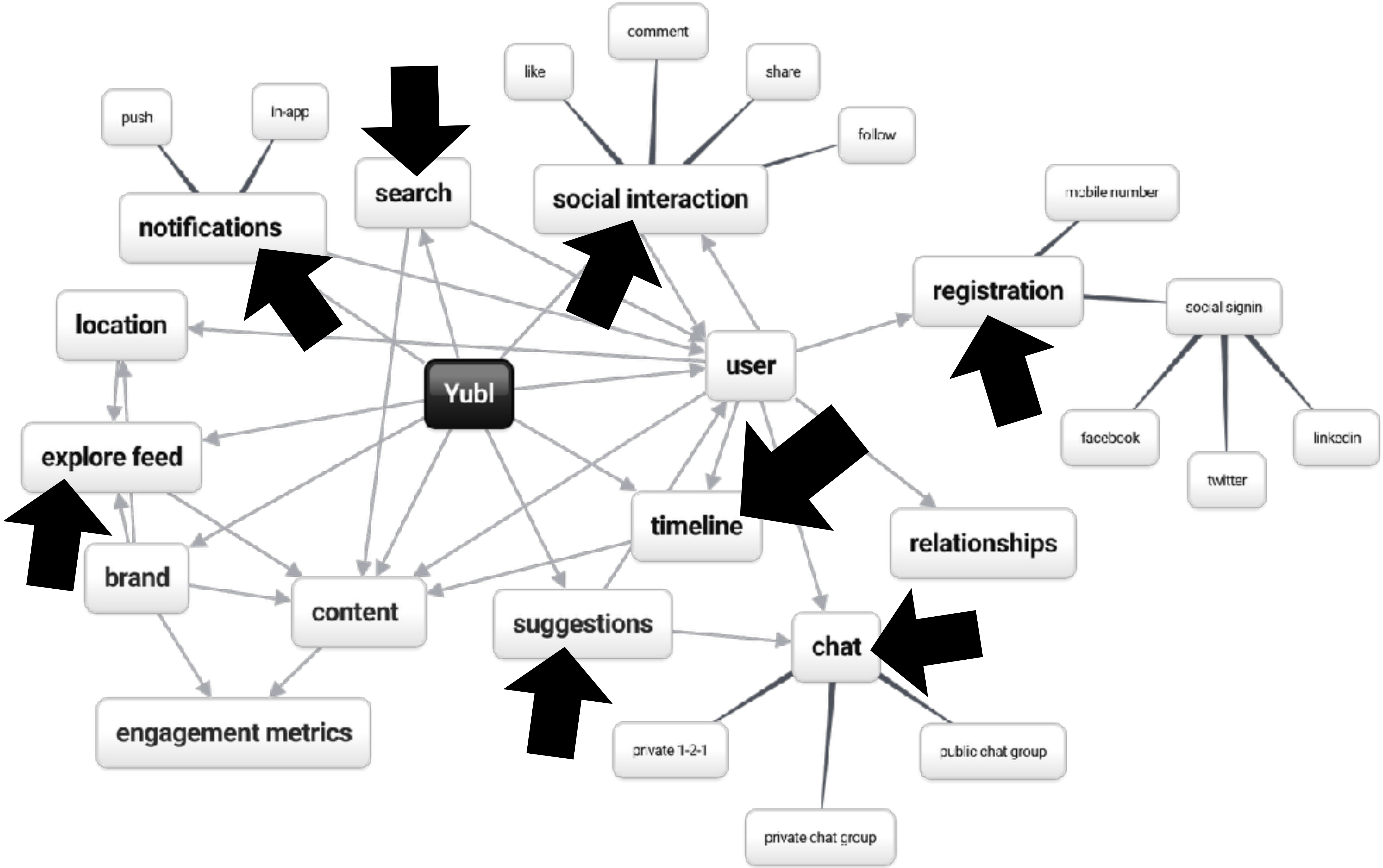
microservices



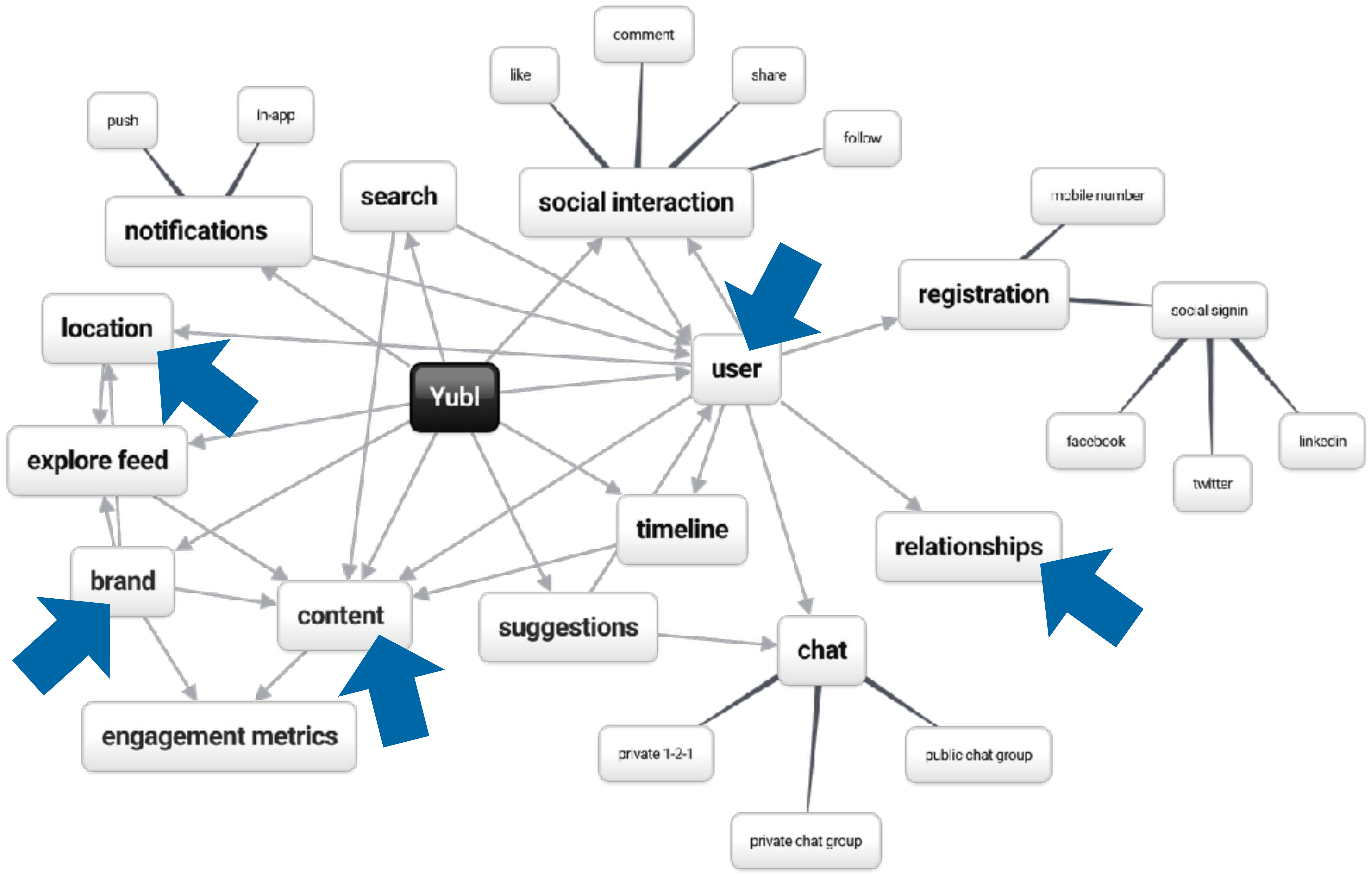
microservices



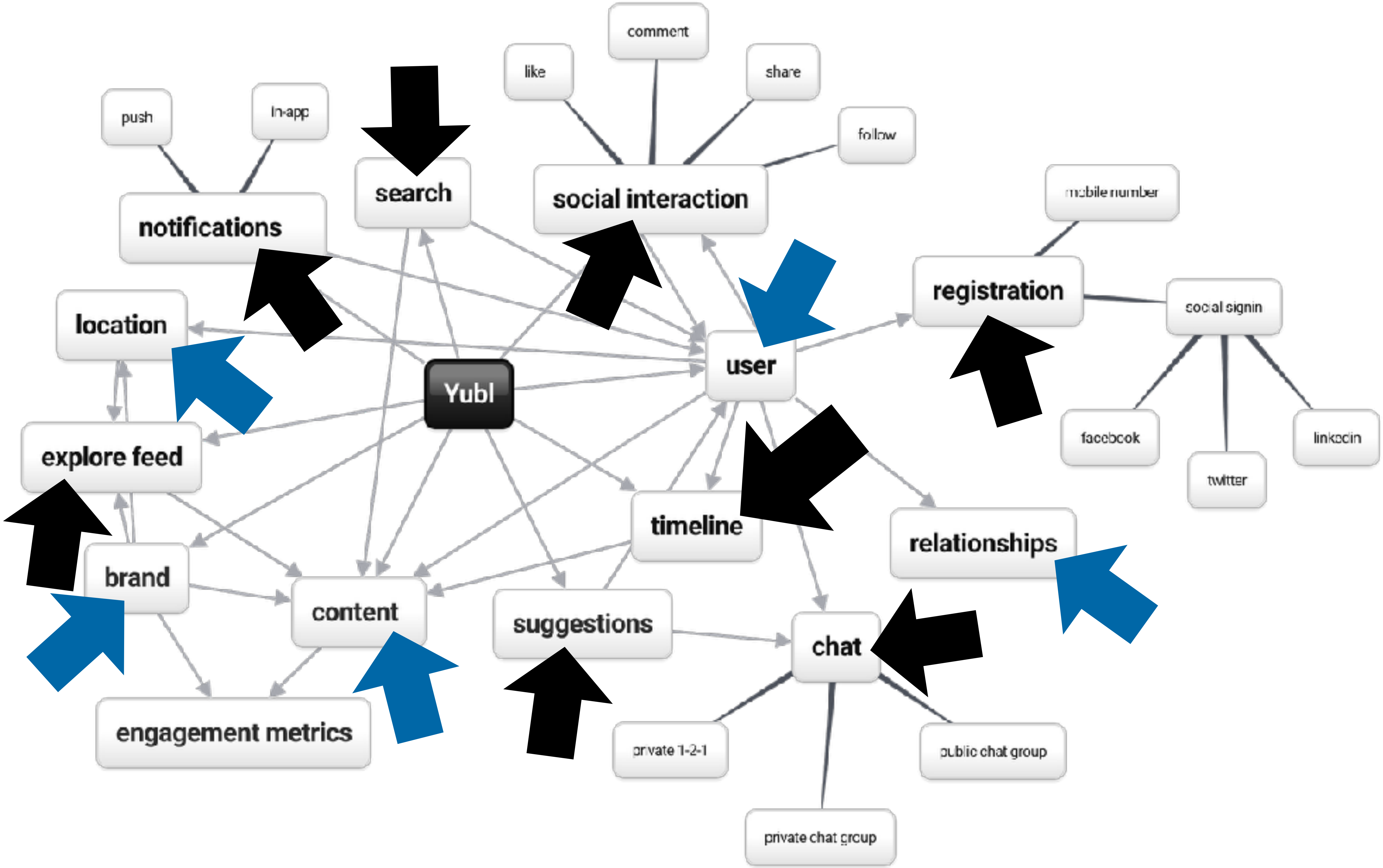
microservices



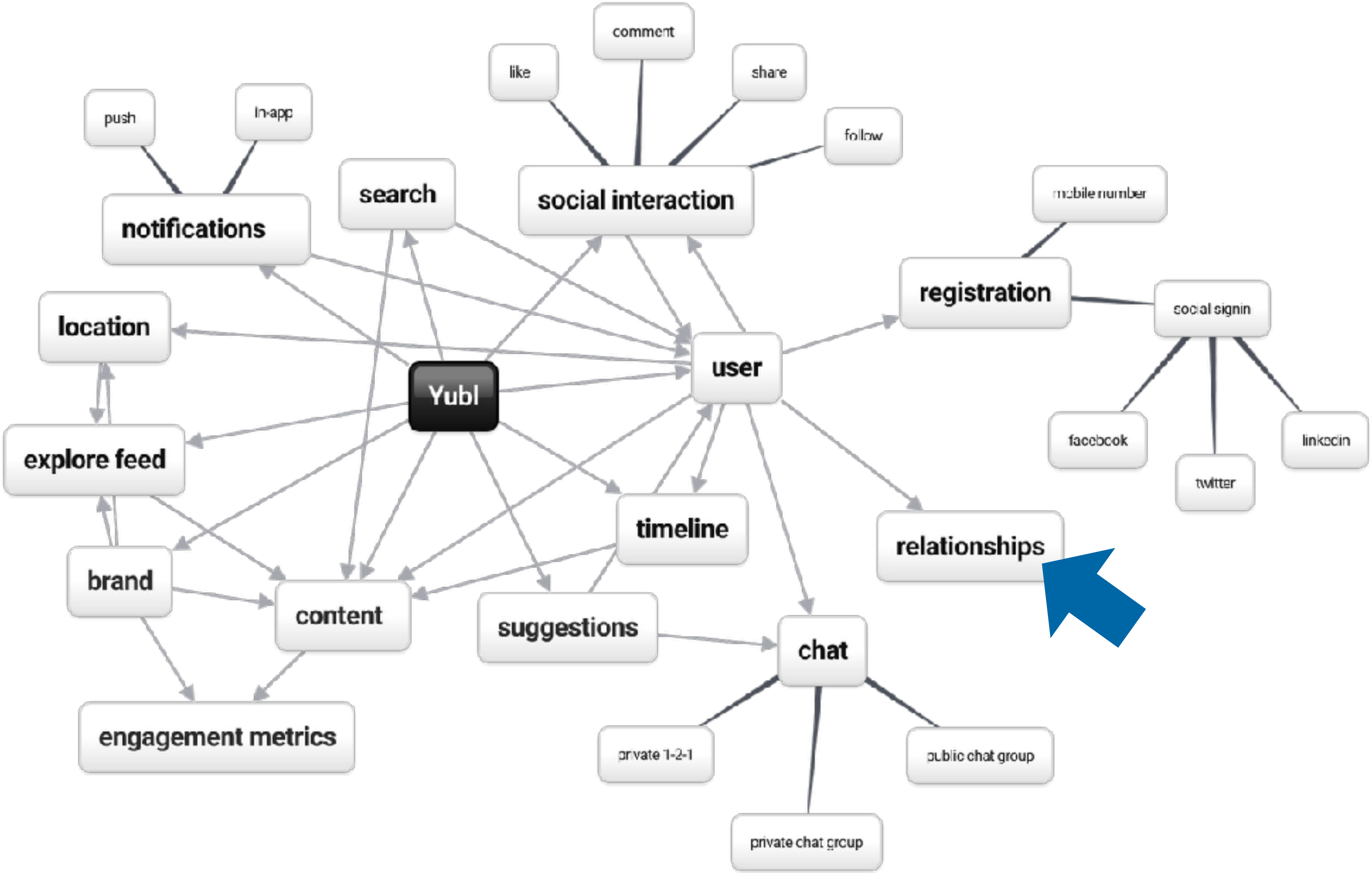
microservices



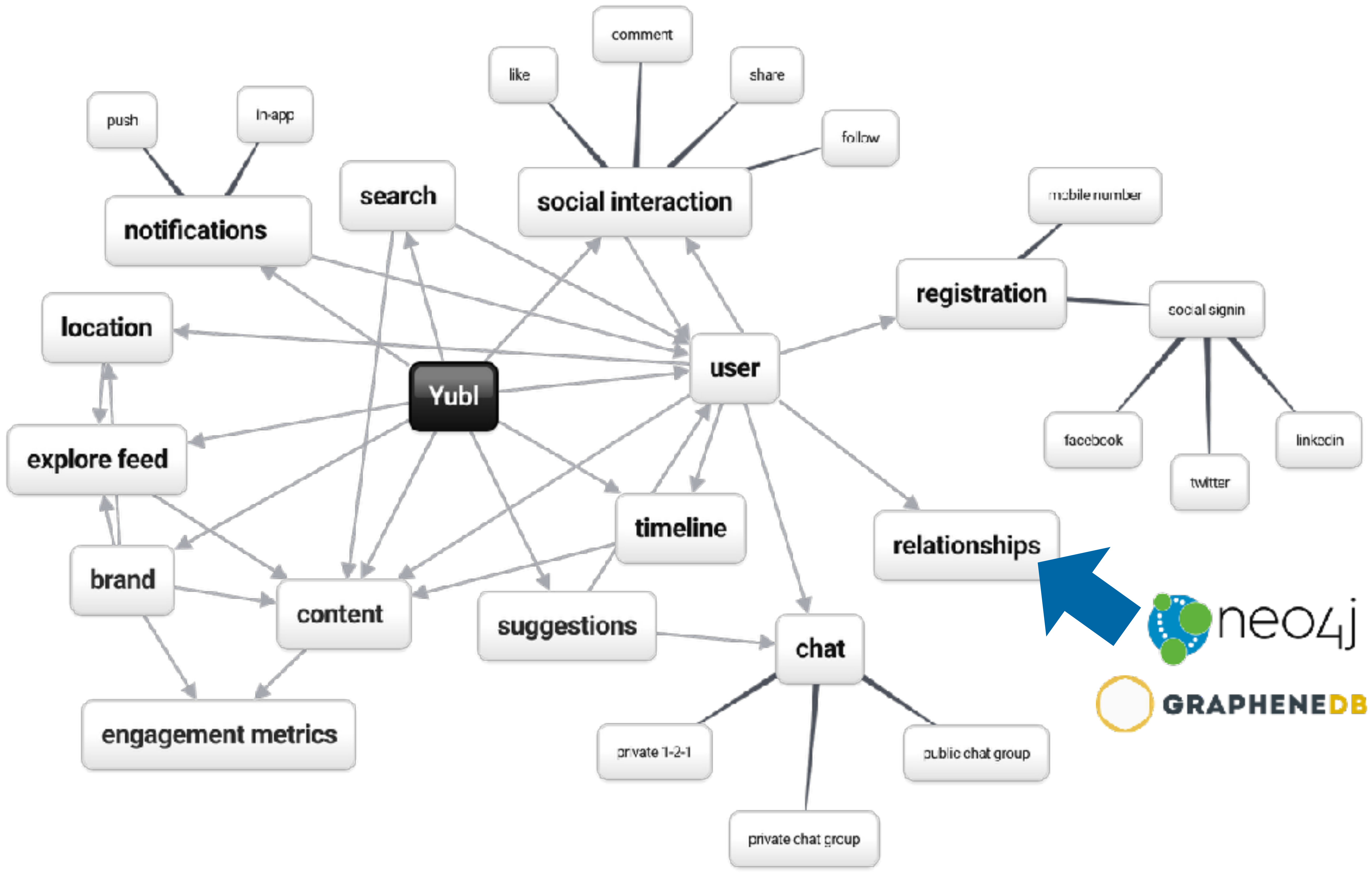
microservices



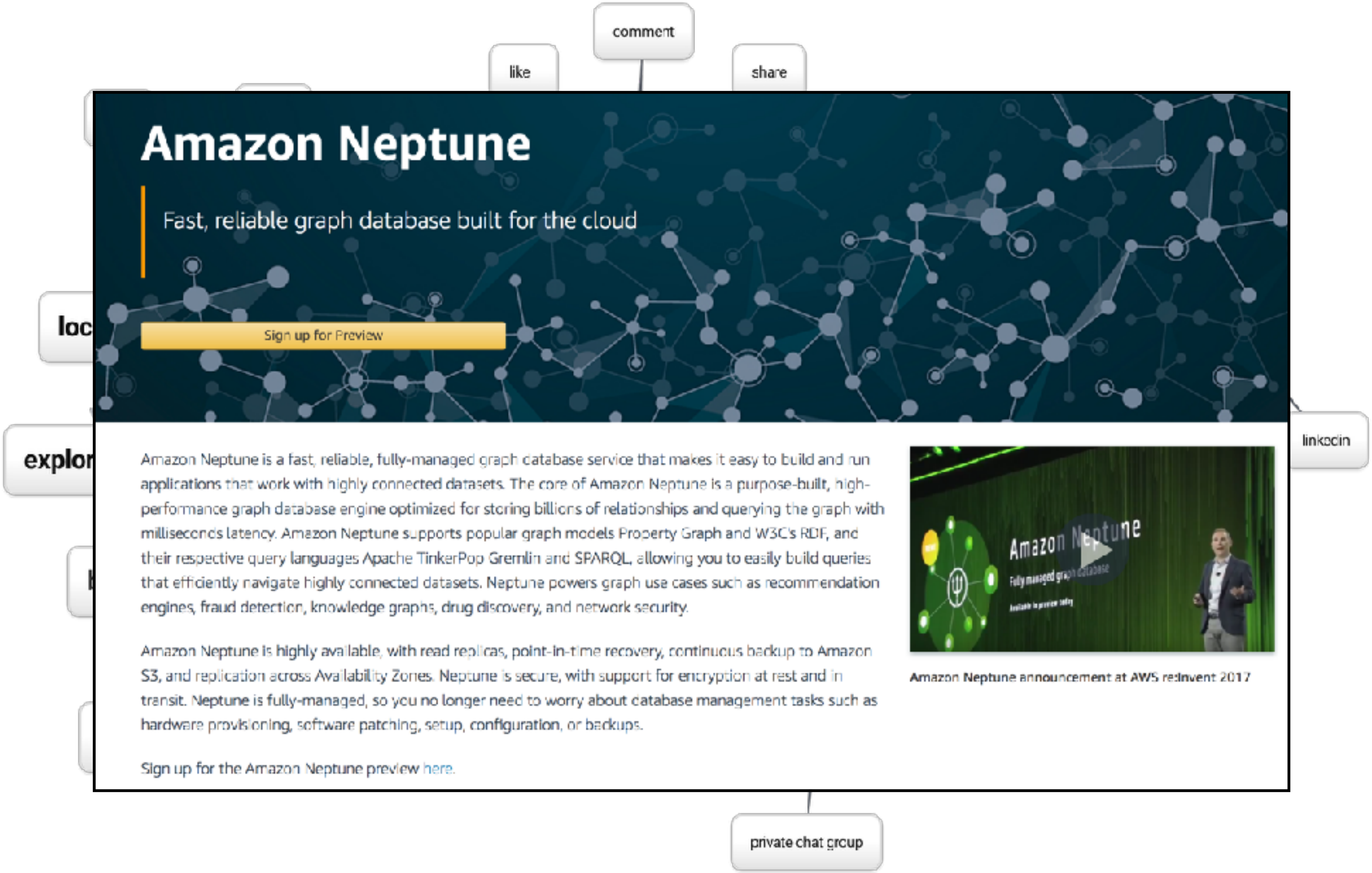
microservices



microservices



microservices



Amazon Neptune

Fast, reliable graph database built for the cloud

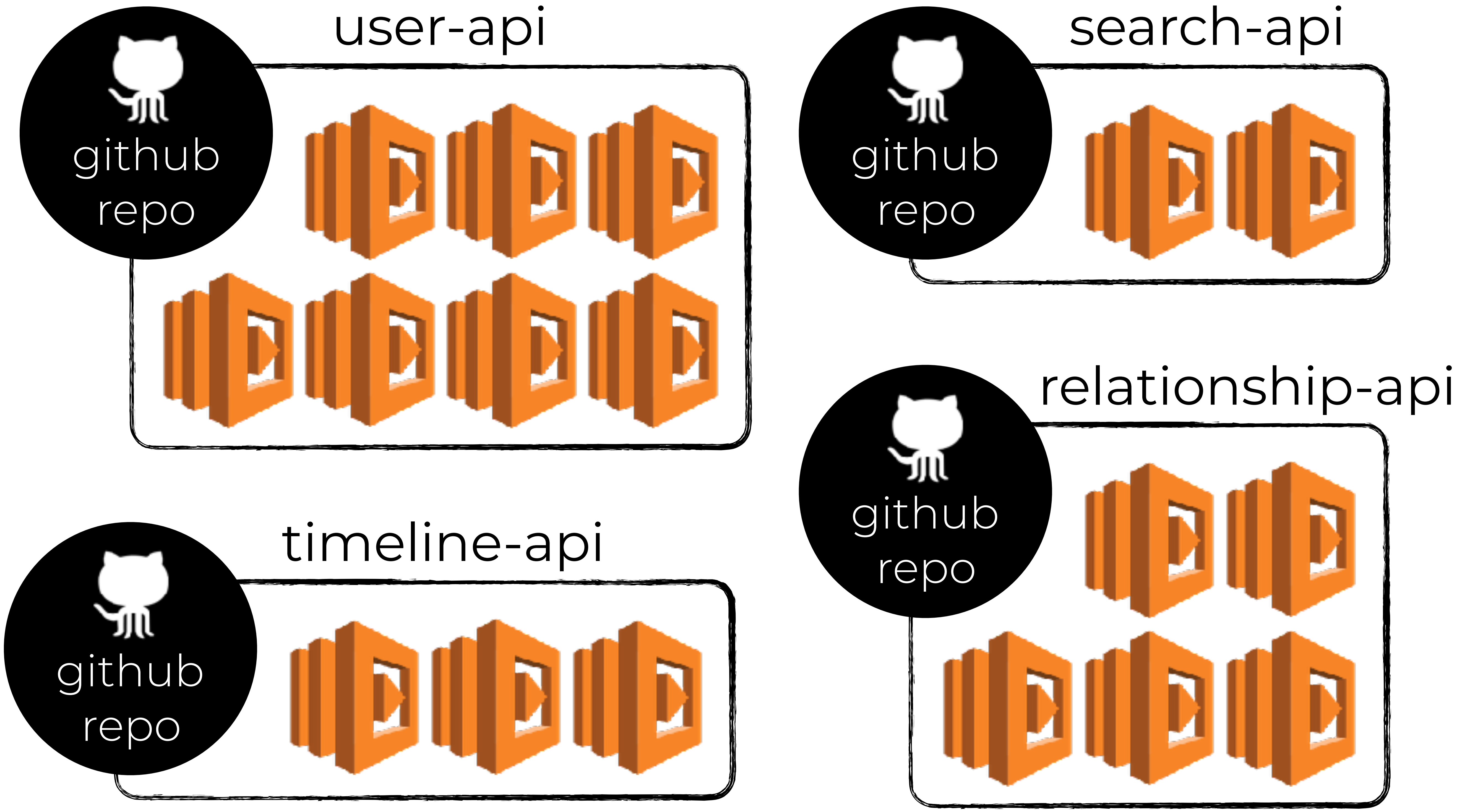
Amazon Neptune is a fast, reliable, fully-managed graph database service that makes it easy to build and run applications that work with highly connected datasets. The core of Amazon Neptune is a purpose-built, high-performance graph database engine optimized for storing billions of relationships and querying the graph with milliseconds latency. Amazon Neptune supports popular graph models Property Graph and W3C's RDF, and their respective query languages Apache TinkerPop Gremlin and SPARQL, allowing you to easily build queries that efficiently navigate highly connected datasets. Neptune powers graph use cases such as recommendation engines, fraud detection, knowledge graphs, drug discovery, and network security.

Amazon Neptune is highly available, with read replicas, point-in-time recovery, continuous backup to Amazon S3, and replication across Availability Zones. Neptune is secure, with support for encryption at rest and in transit. Neptune is fully-managed, so you no longer need to worry about database management tasks such as hardware provisioning, software patching, setup, configuration, or backups.

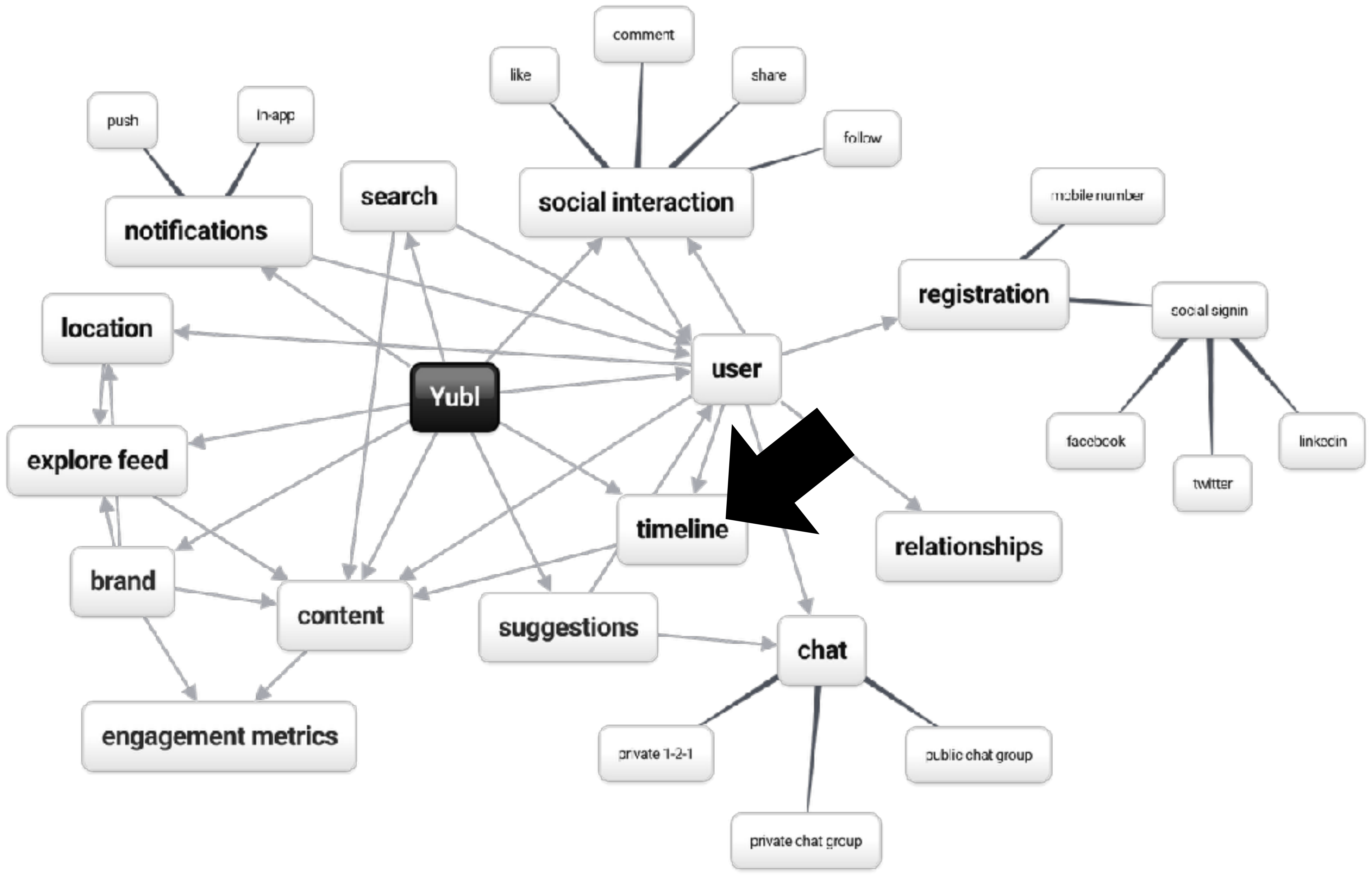
Sign up for the Amazon Neptune preview here.

Amazon Neptune announcement at AWS re:Invent 2017

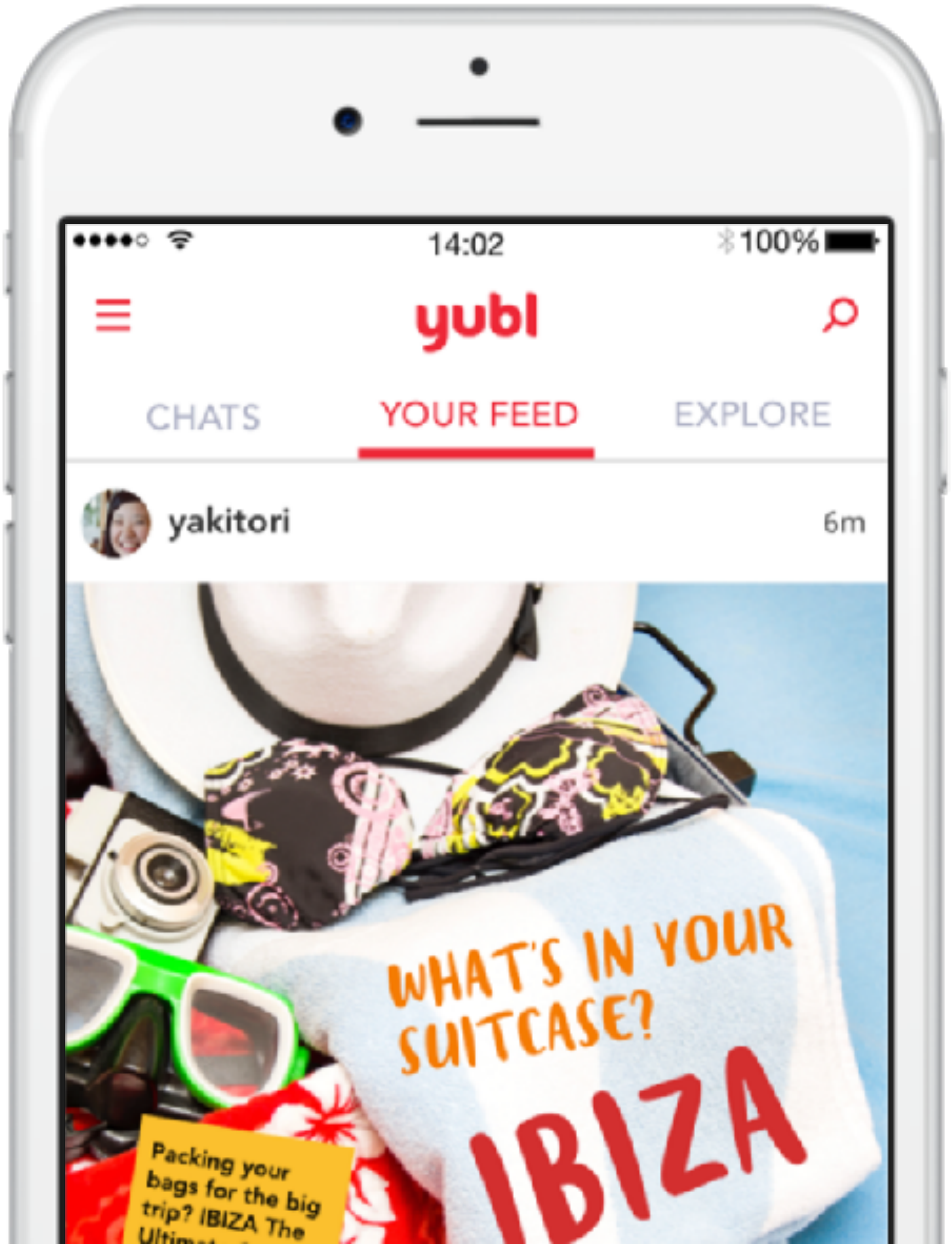
microservices



timeline feature

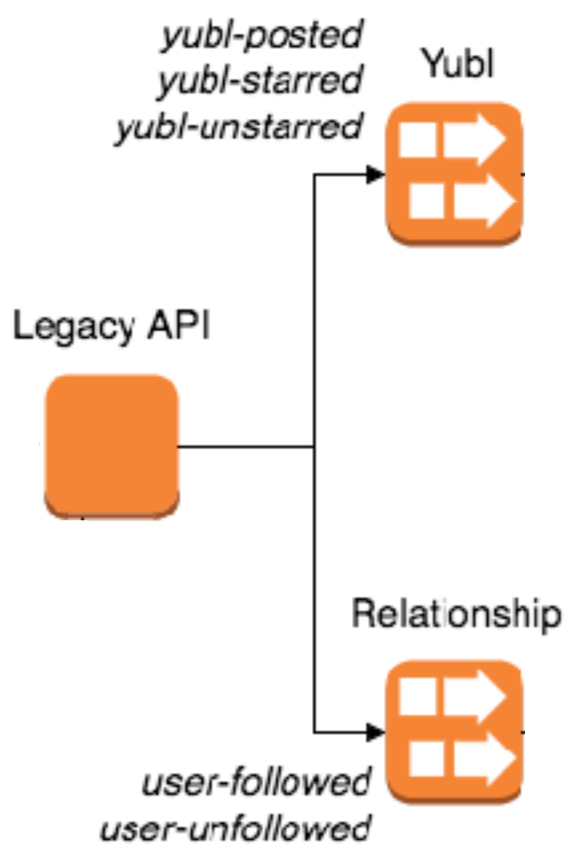


timeline feature

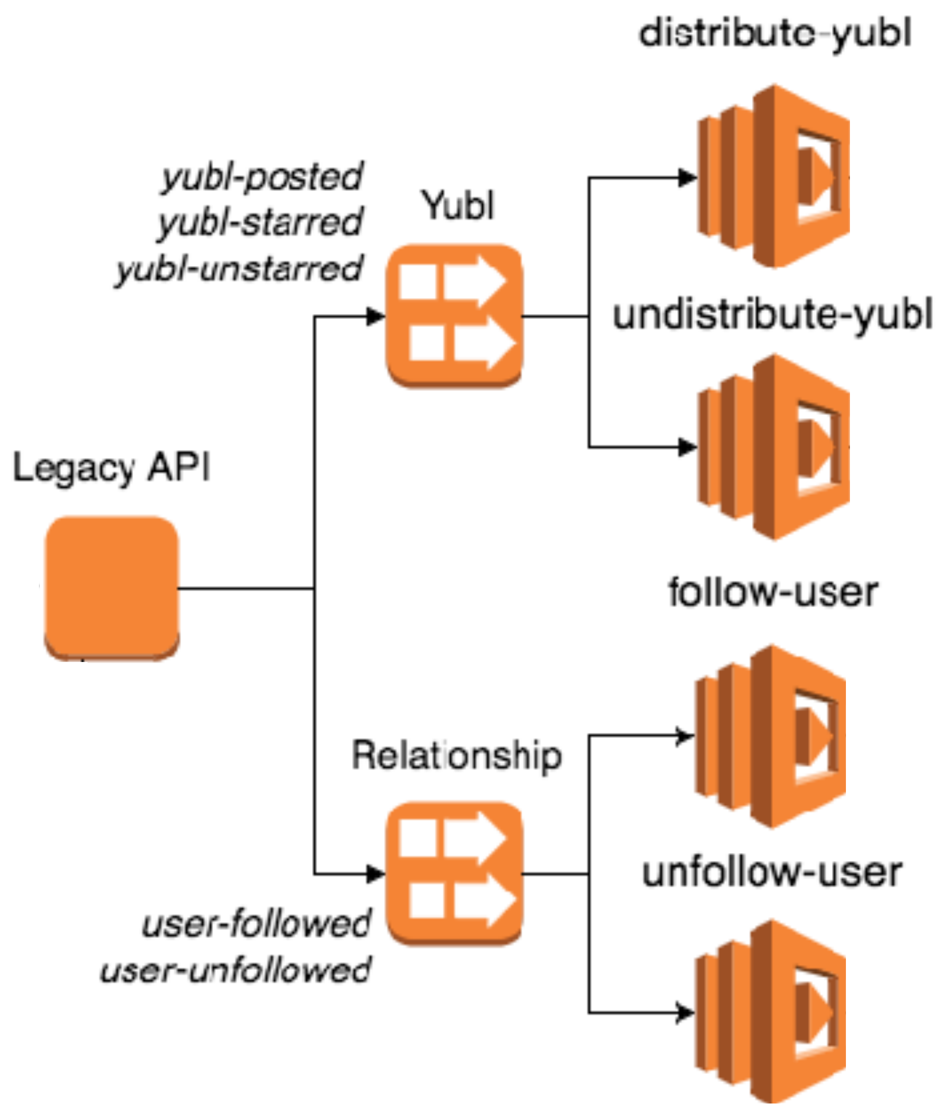


timeline feature

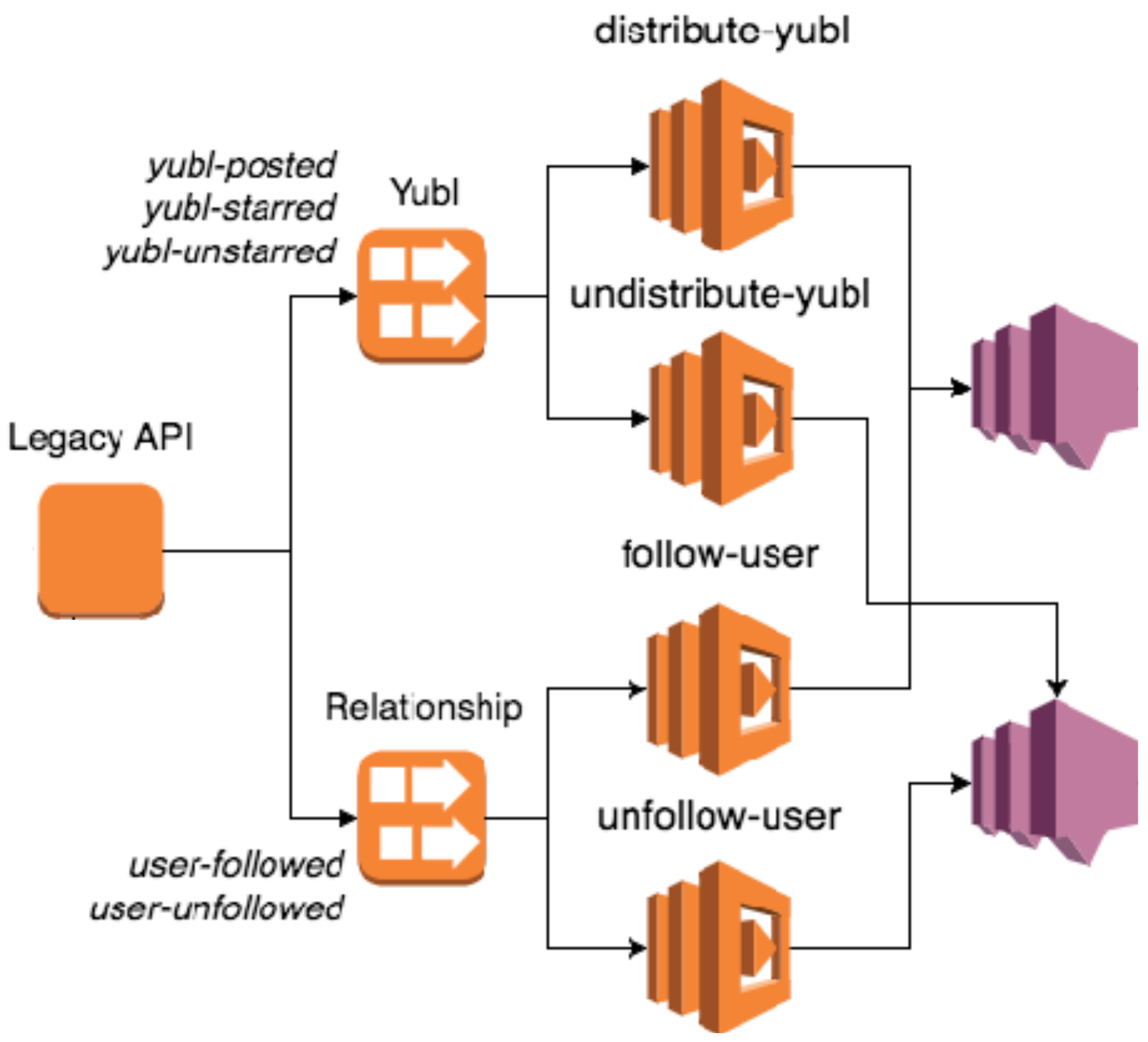
timeline feature



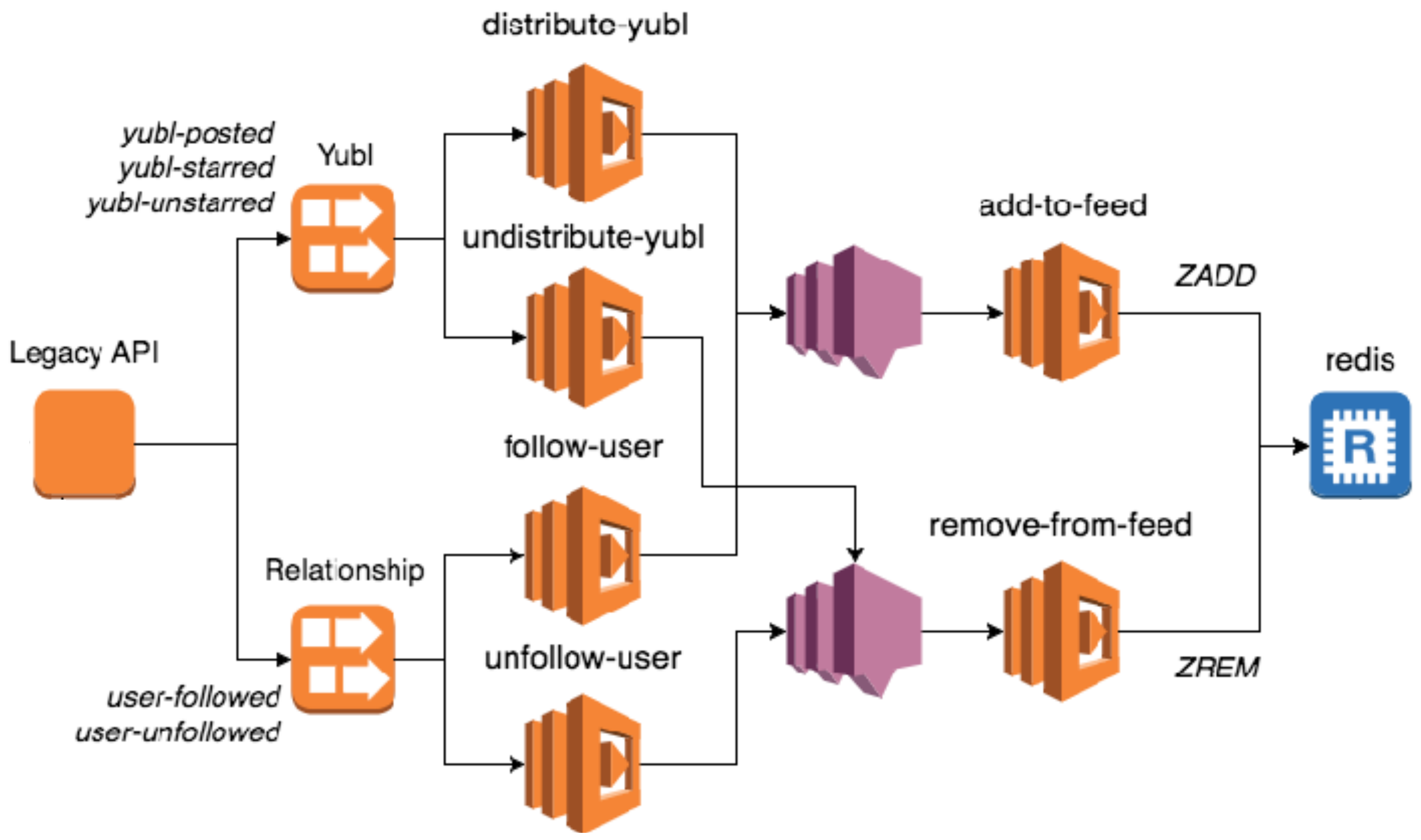
timeline feature



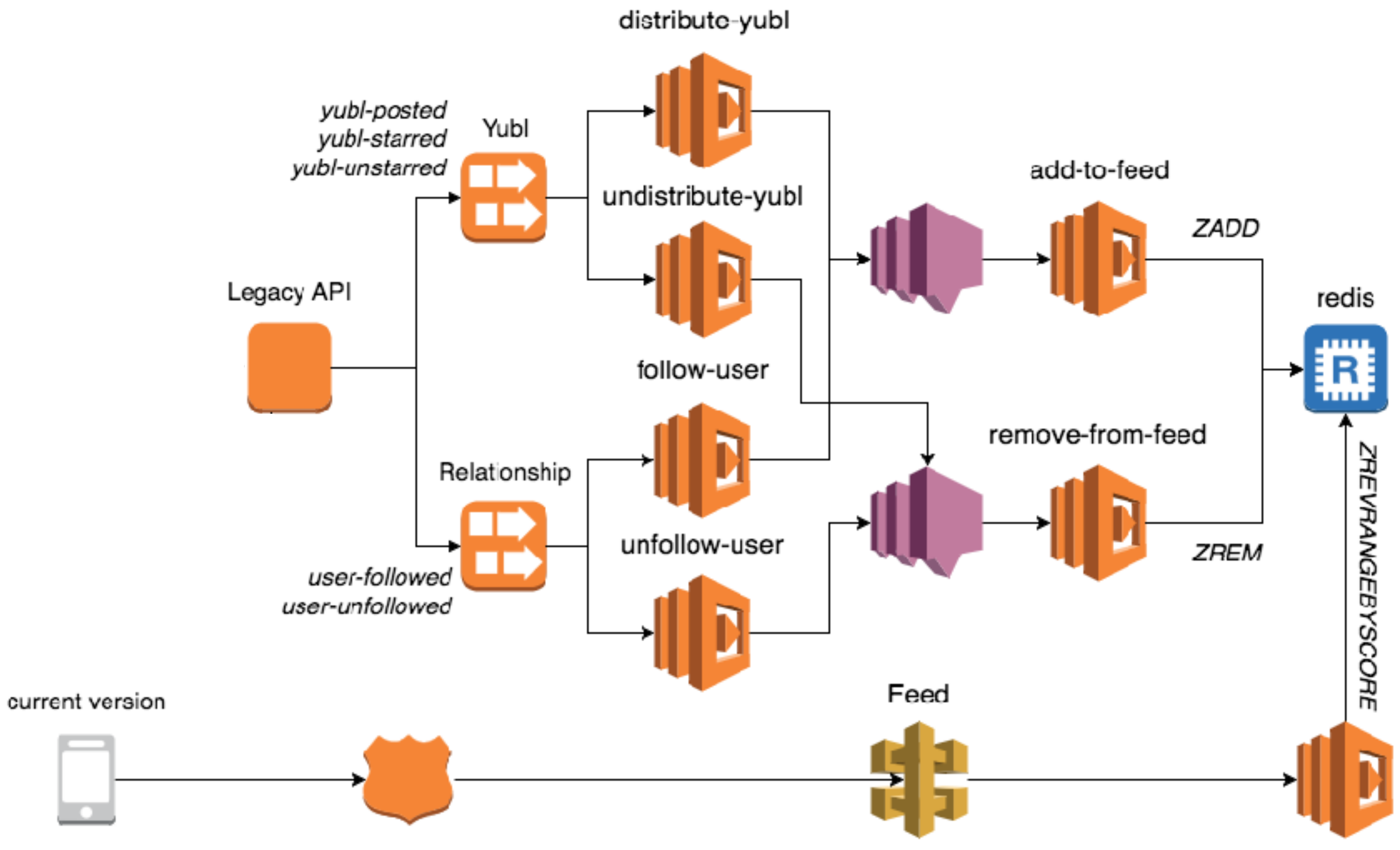
timeline feature



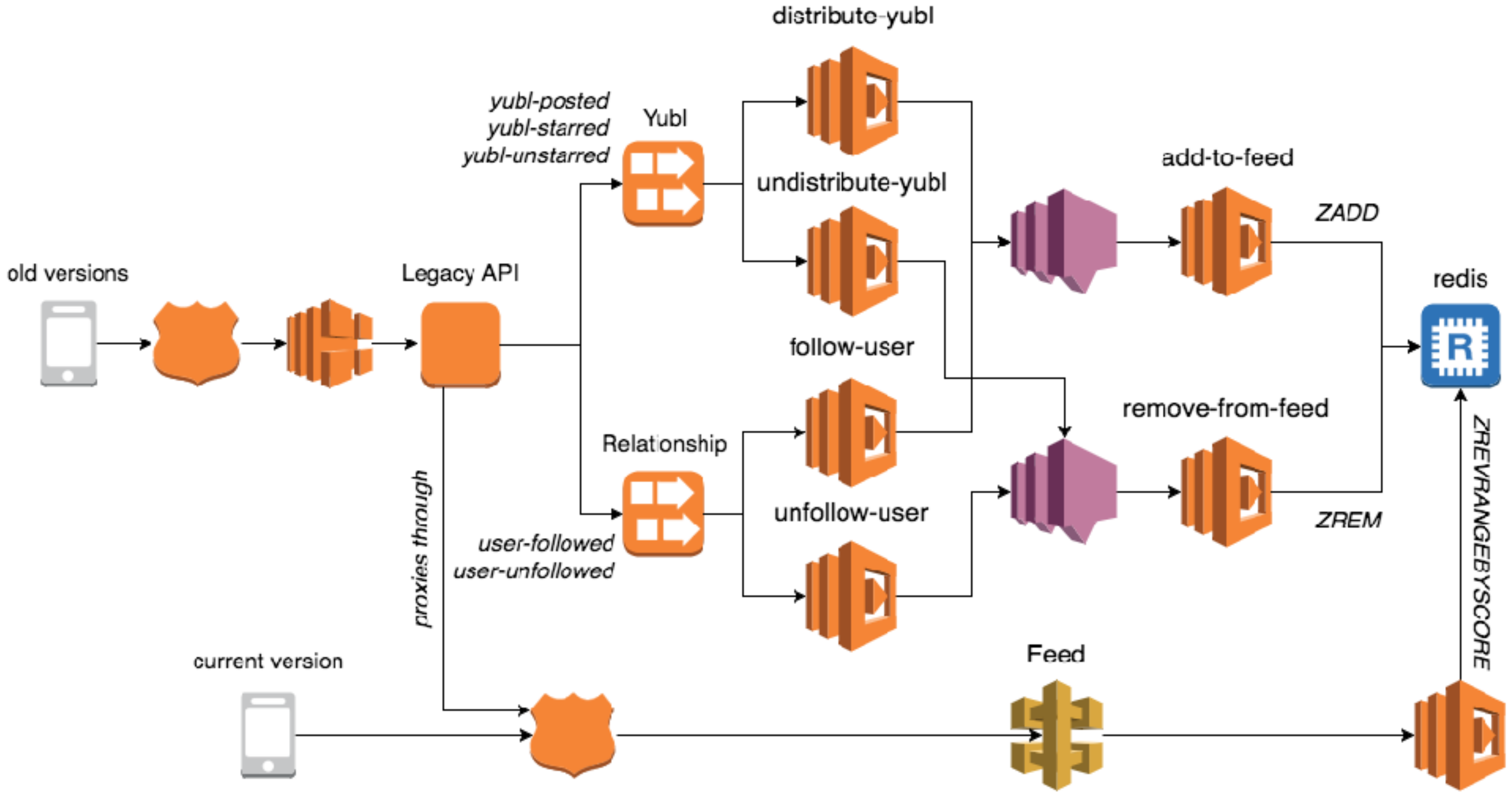
timeline feature



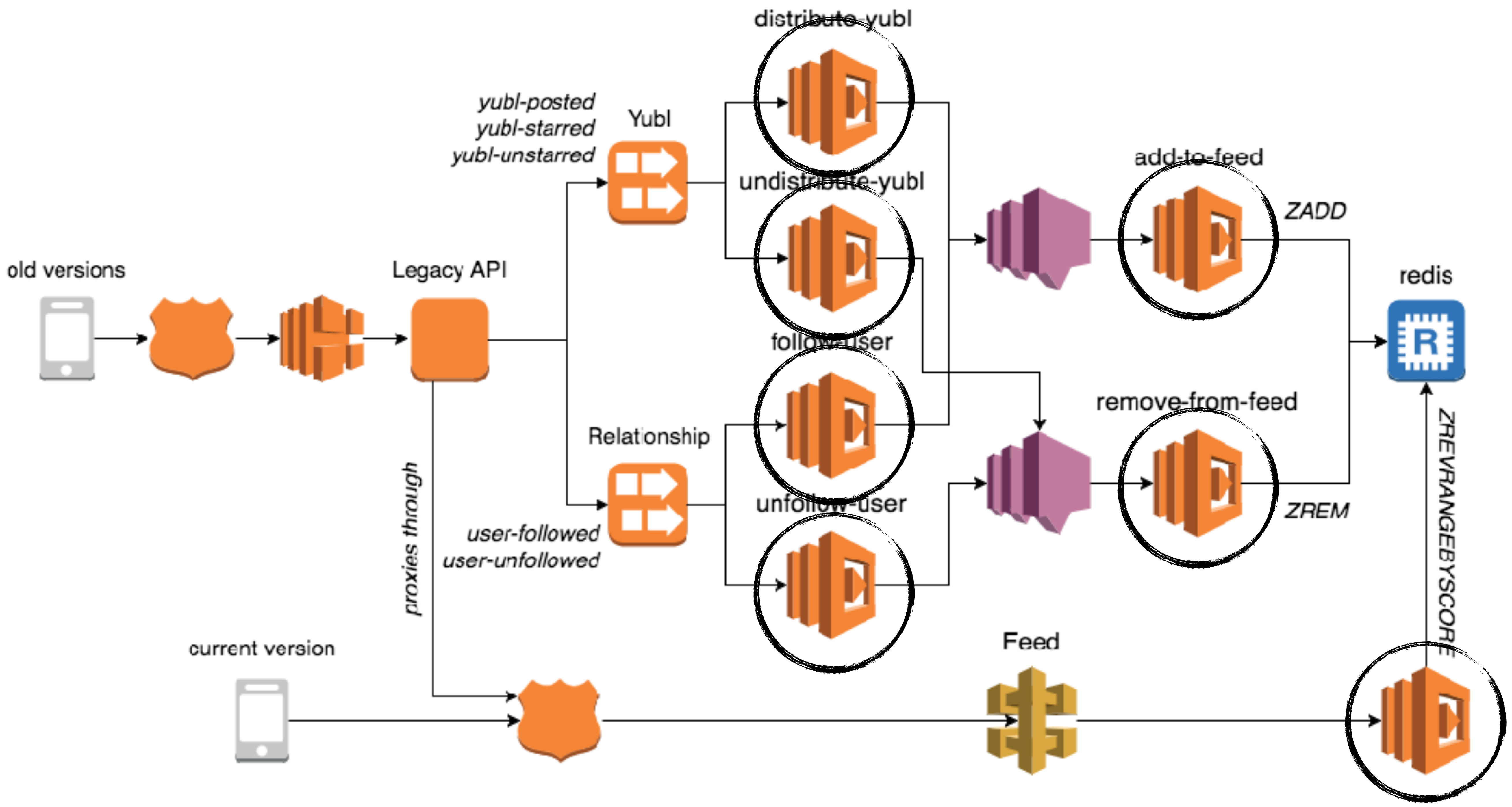
timeline feature



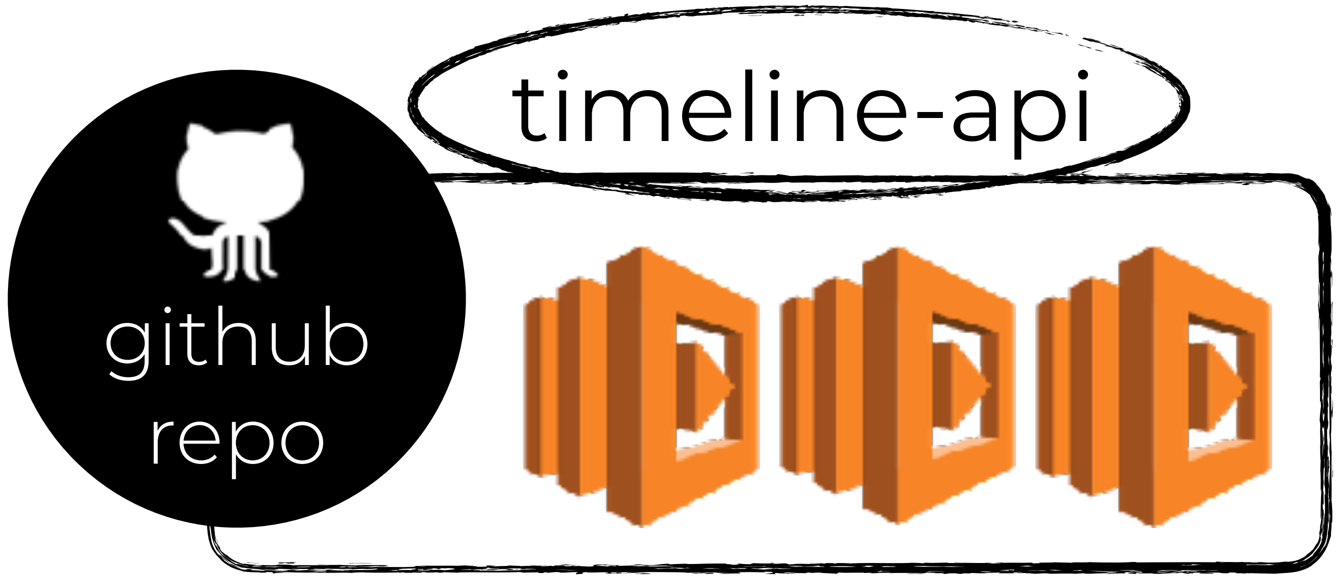
timeline feature



timeline feature

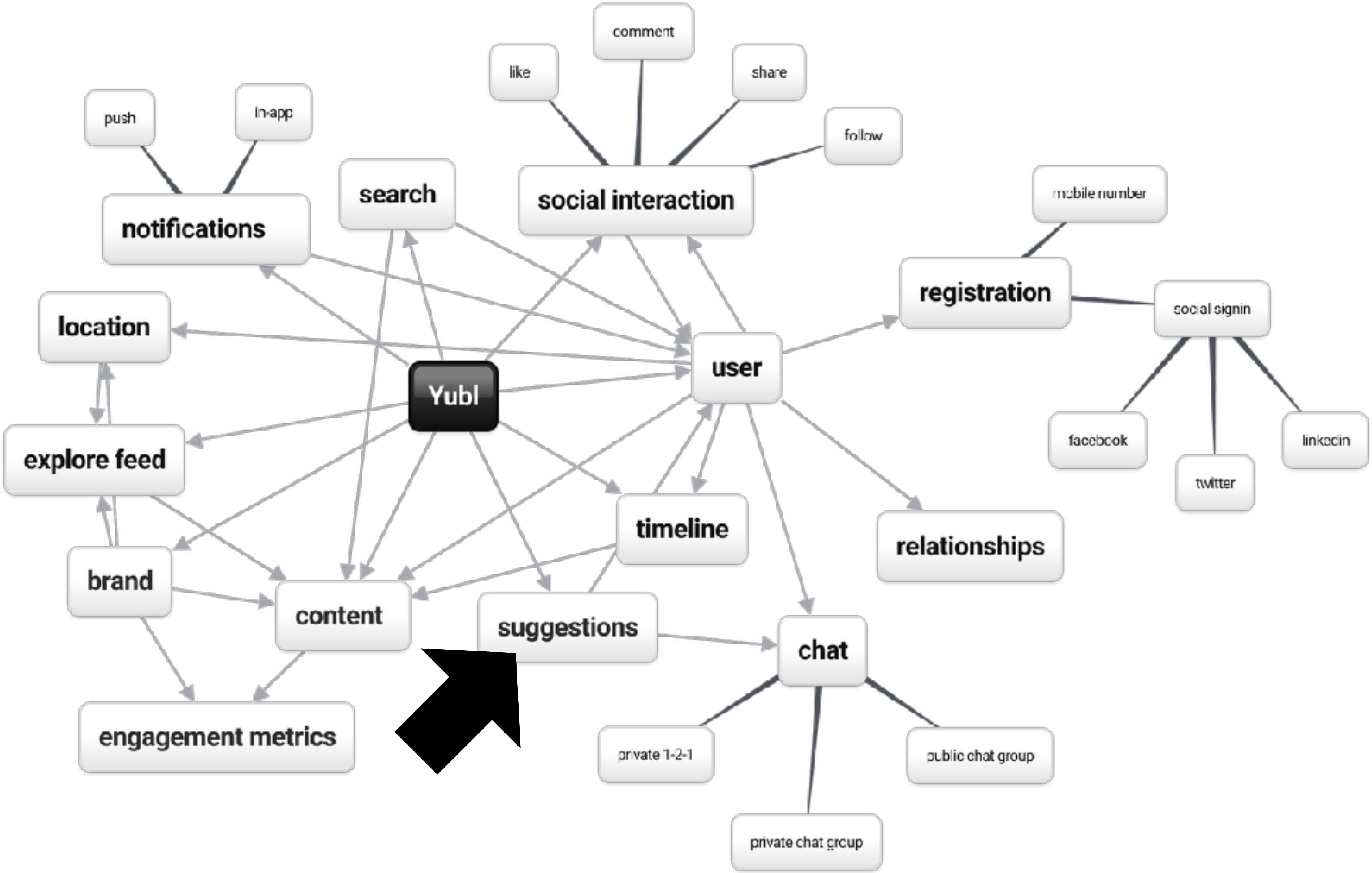


timeline feature

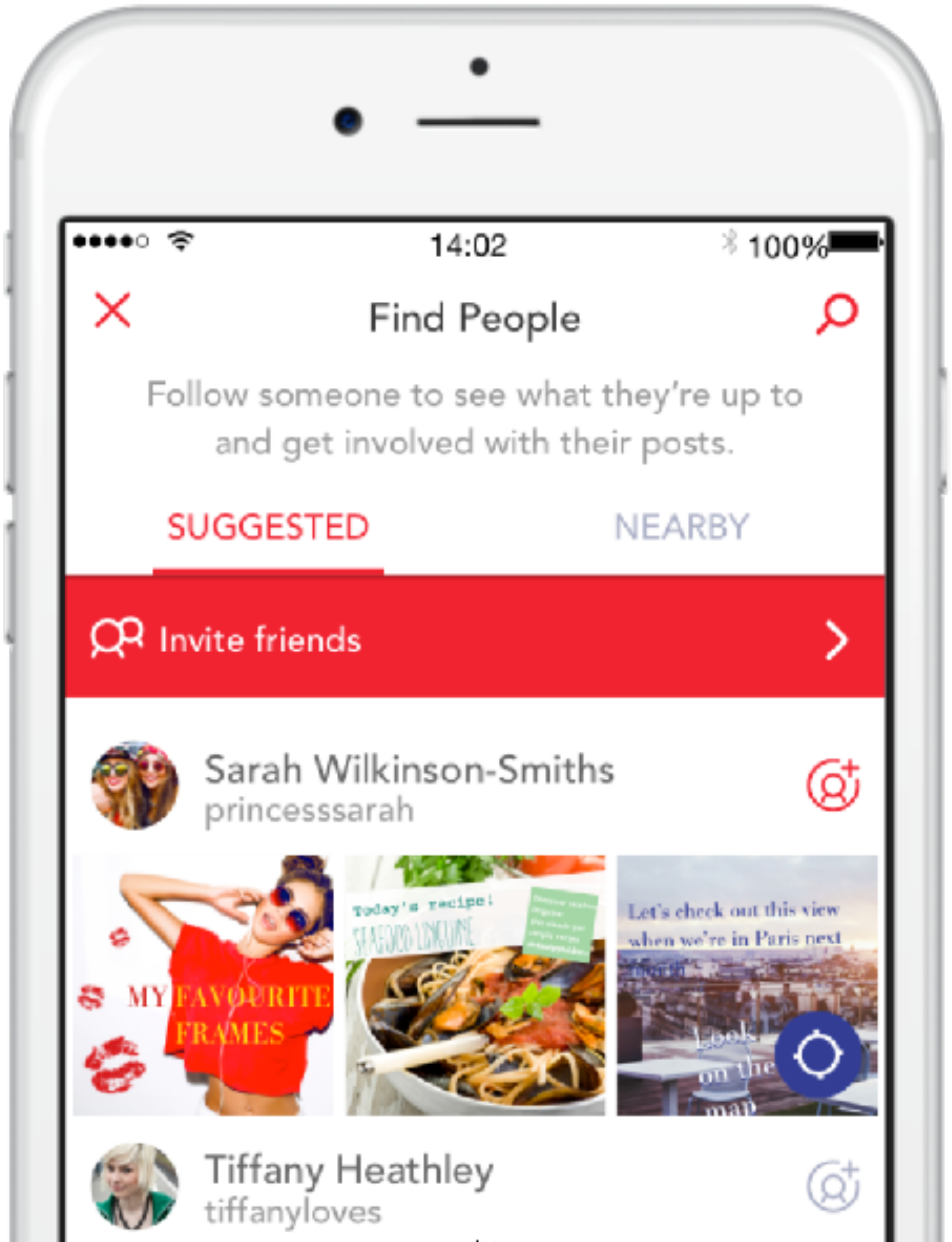


```
service: timeline-api

provider:
  name: aws
  runtime: nodejs6.10
  stage: dev
  region: us-east-1

functions:
  distribute-yubl:
    ...
  undistribute-yubl:
   ...
```

suggestions feature



suggestions feature



14:02

100%

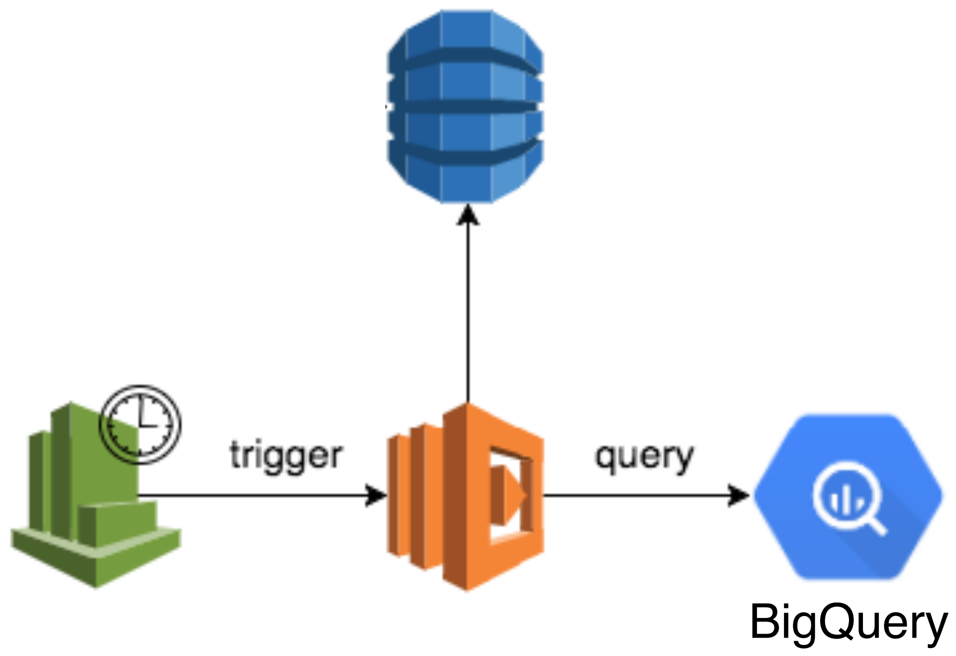
Find People

Follow someone to see what they're up to and get involved with their posts.

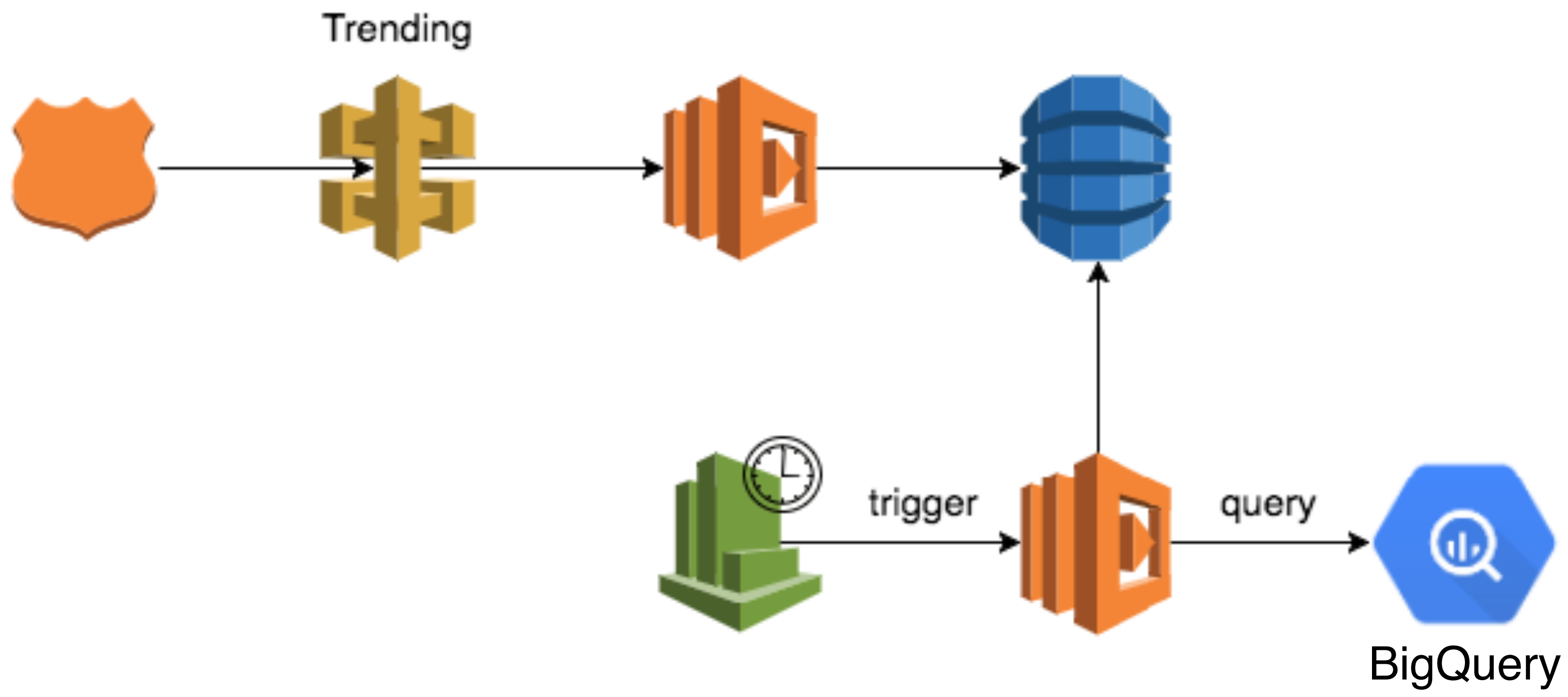
Sarah Wilkinson-Smiths
princesssarah

Tiffany Heathley
tiffanyloves

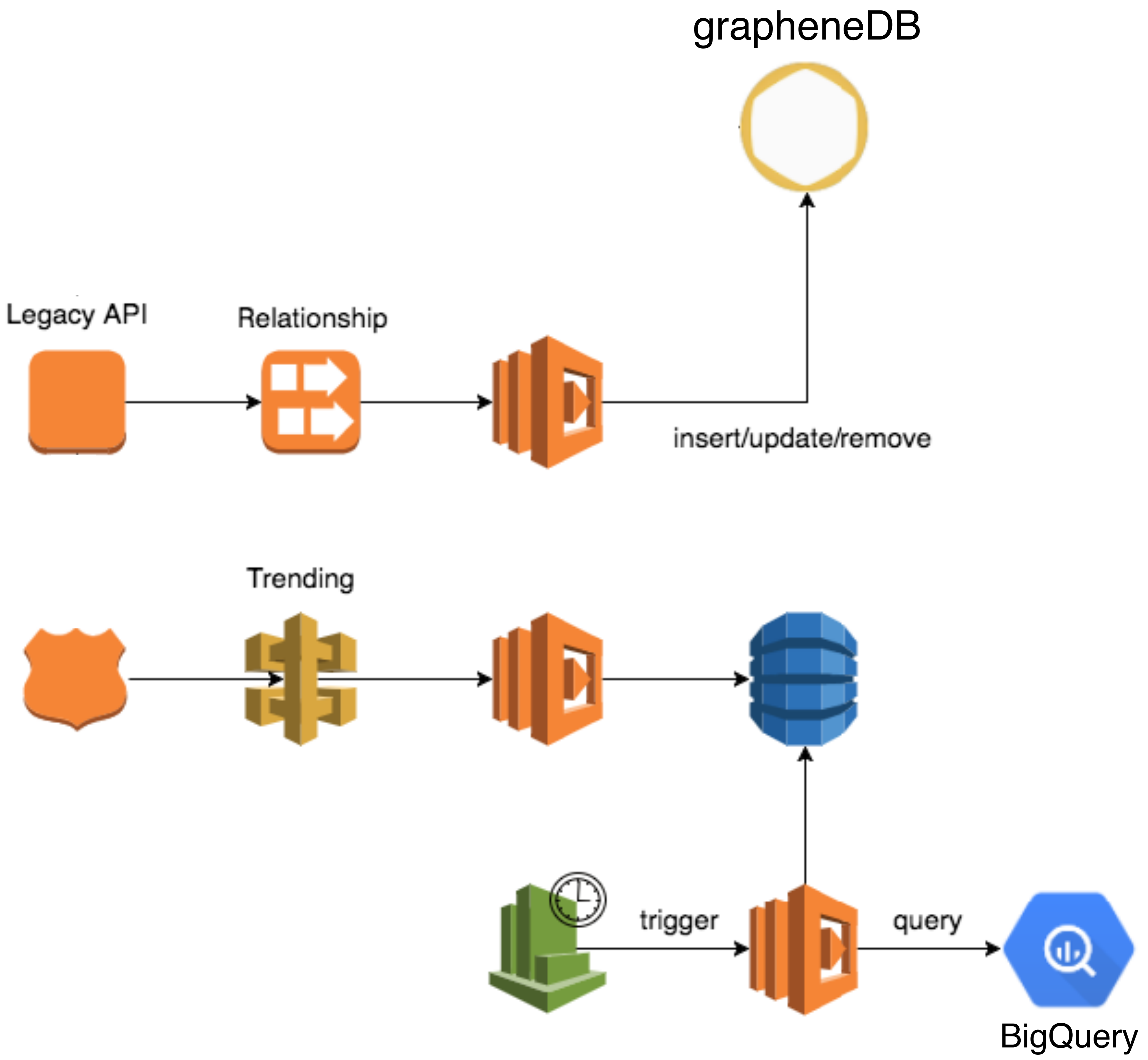
suggestions feature



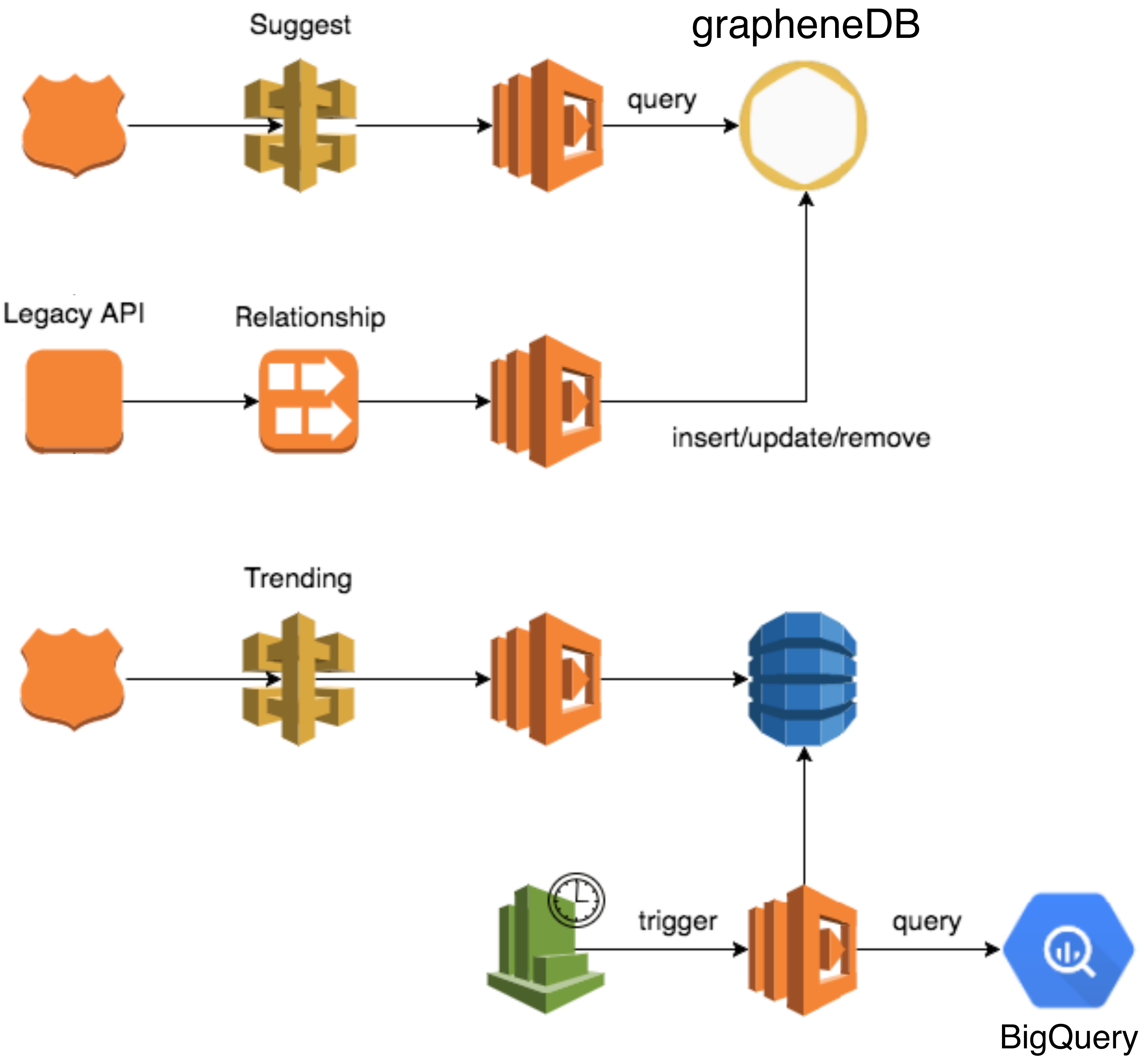
suggestions feature



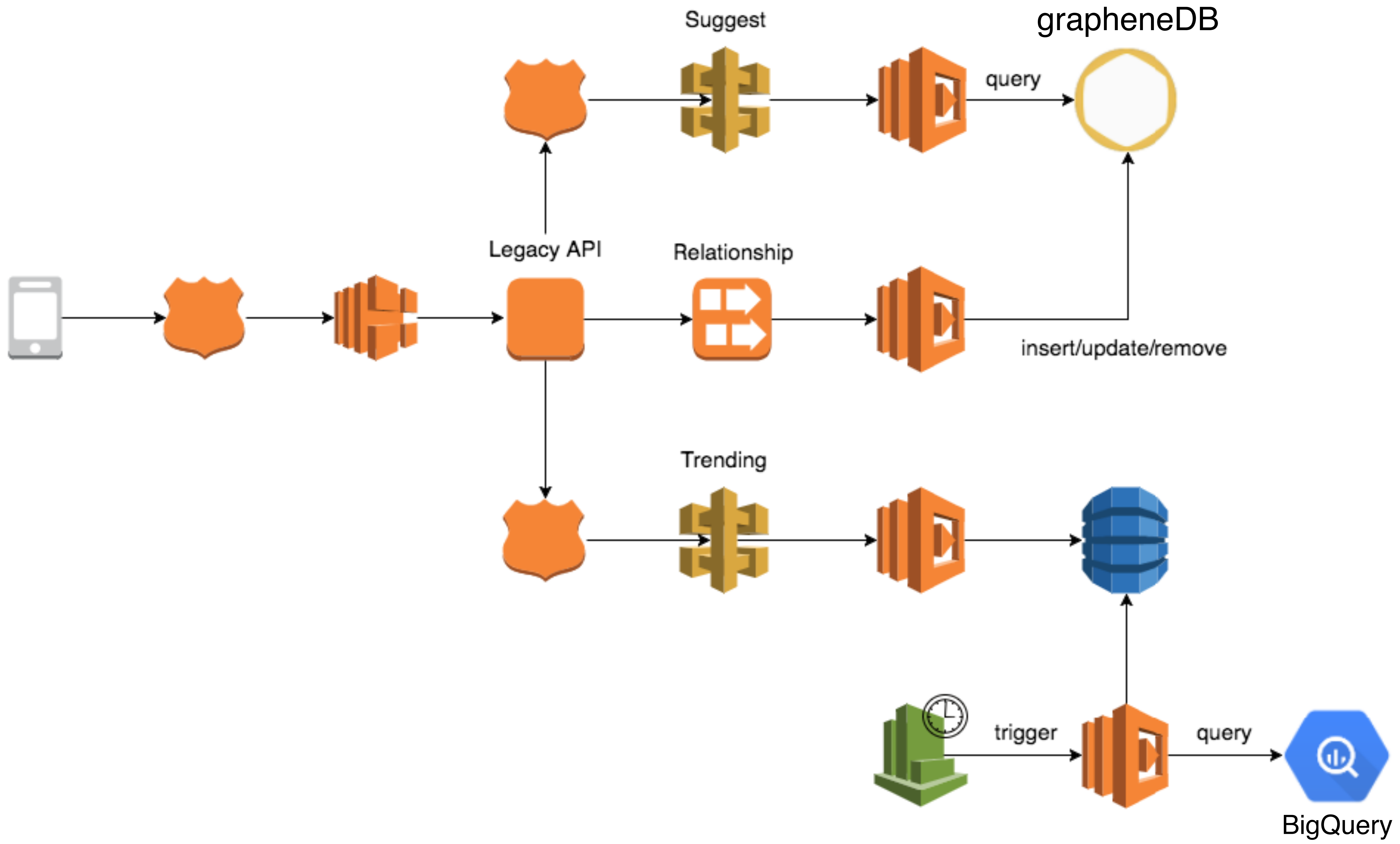
suggestions feature



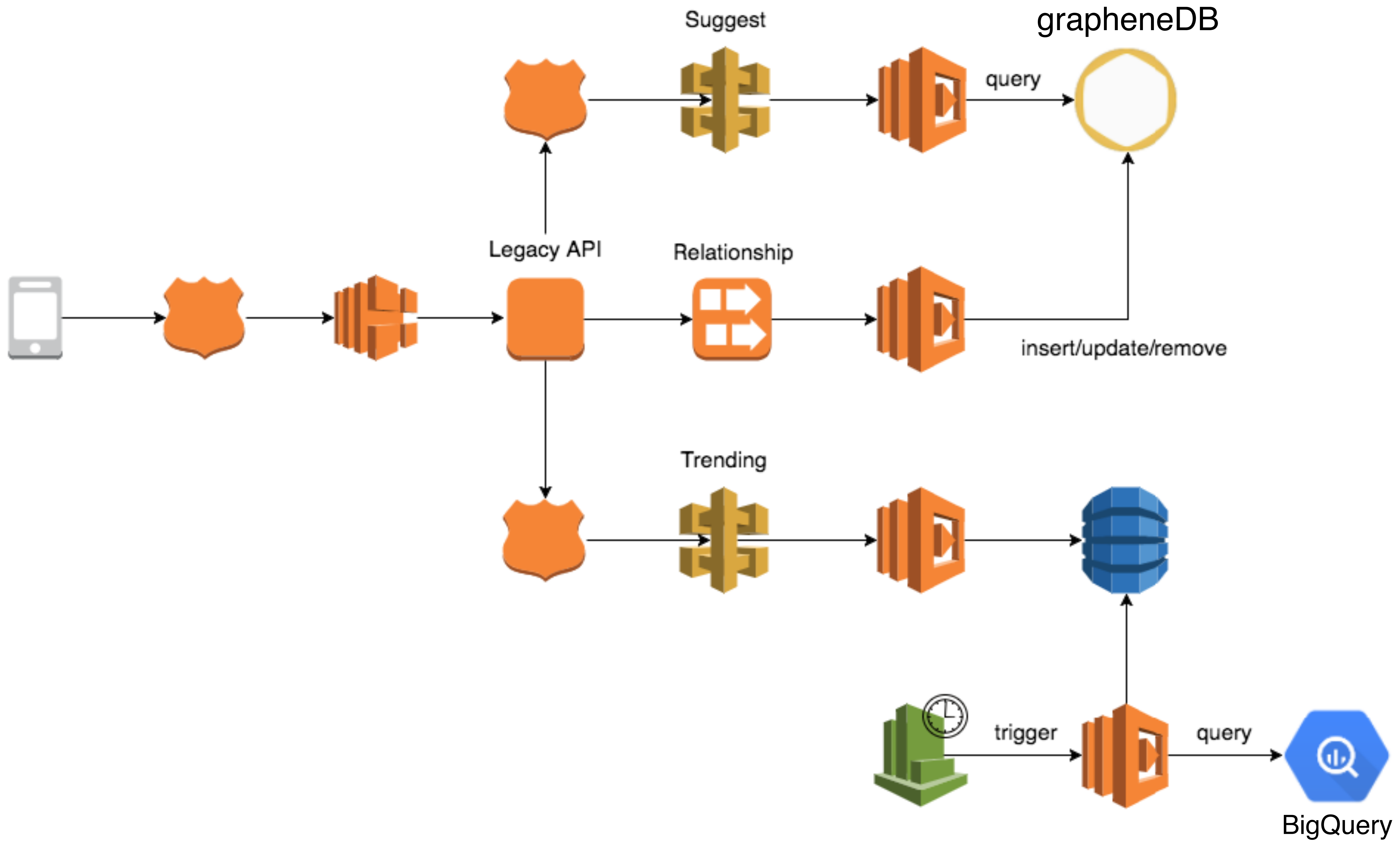
suggestions feature



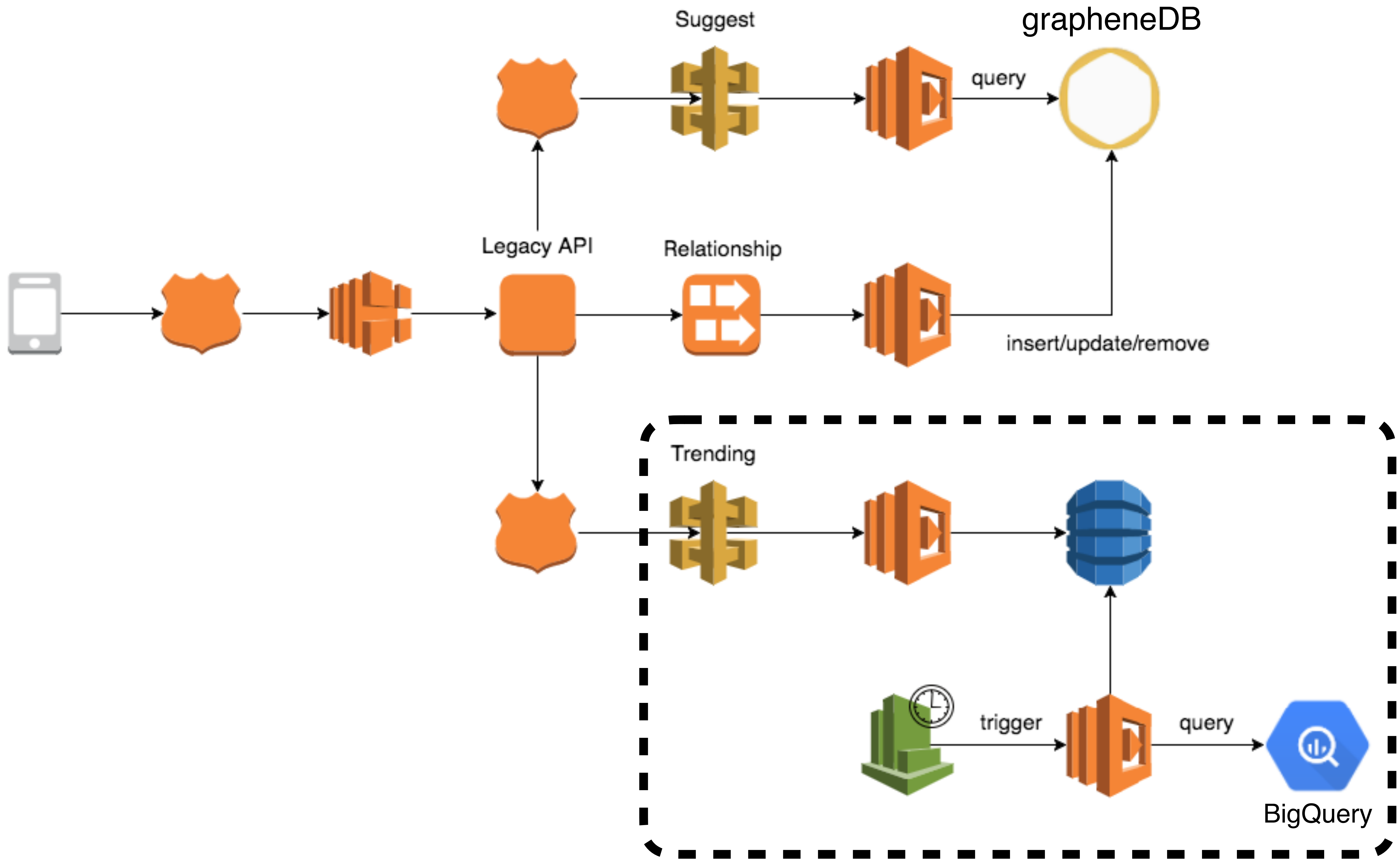
suggestions feature



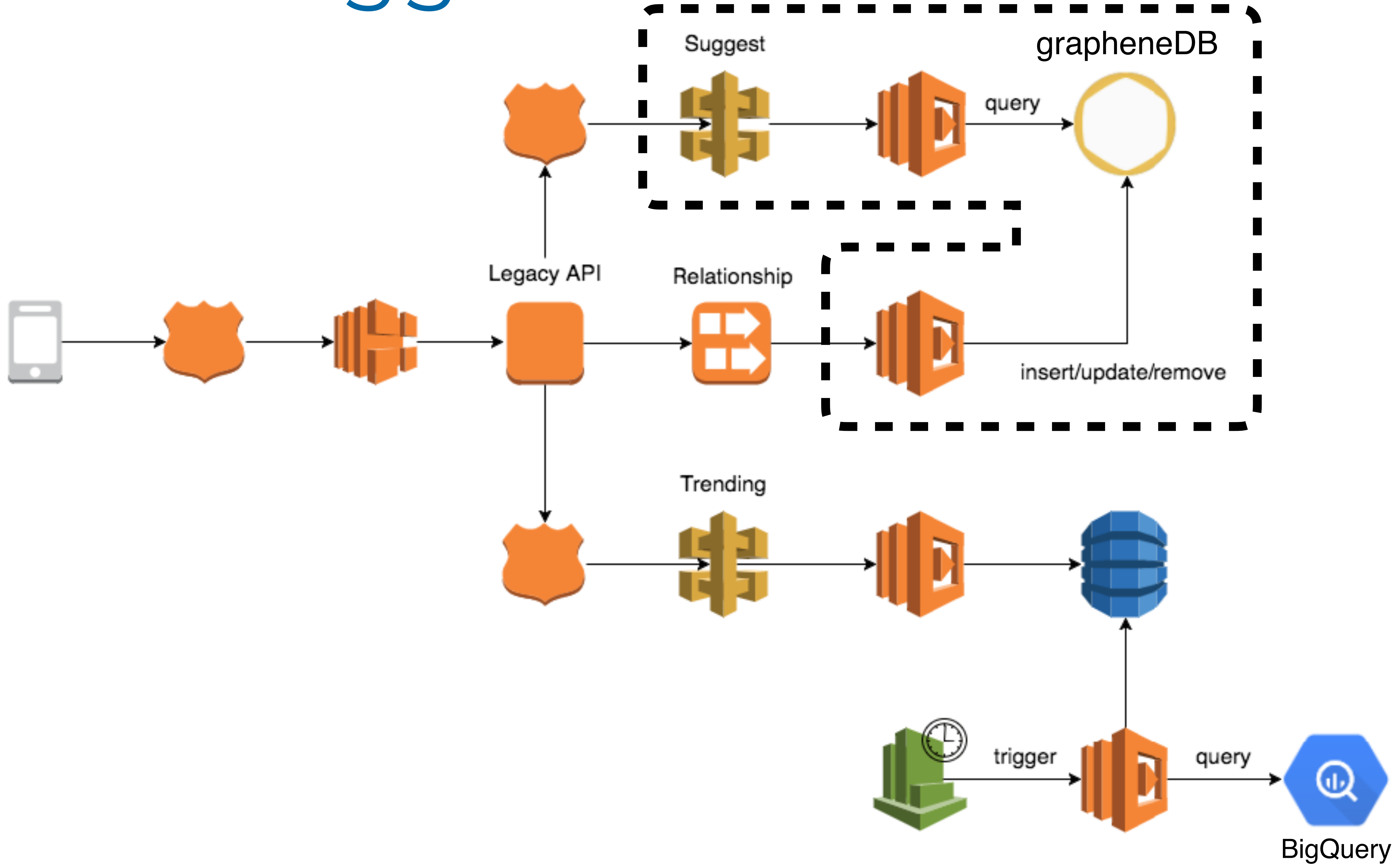
suggestions feature



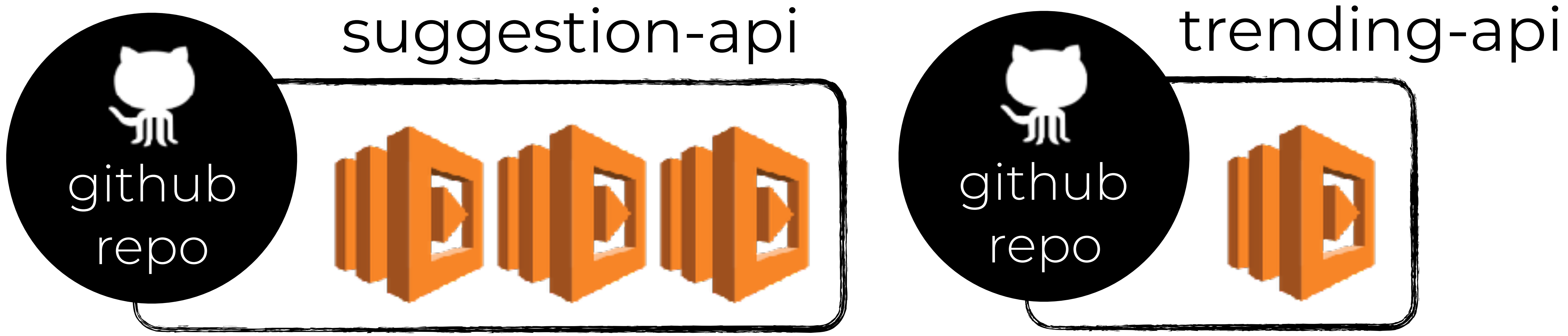
suggestions feature



suggestions feature



suggestions feature



organizing functions

organise functions into repositories according to boundaries you identify in your system

organizing functions

optimize for **high cohesion**

high cohesion, low coupling

coupling

“the degree of **interdependencies** between software modules”

high cohesion, low coupling

coupling

“the degree of **interdependencies** between software modules”

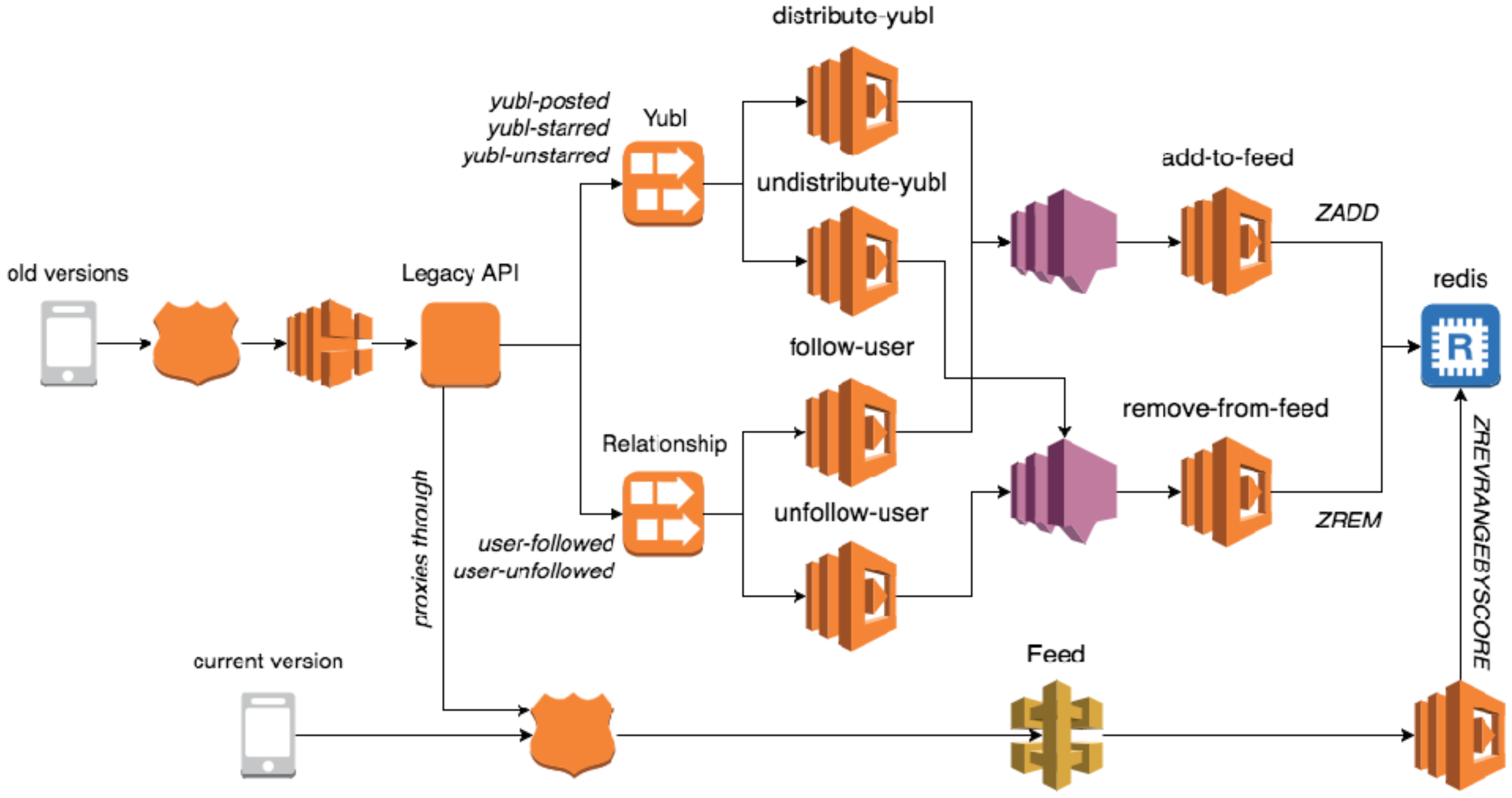
cohesion

“how **related** the functions within a single module are”

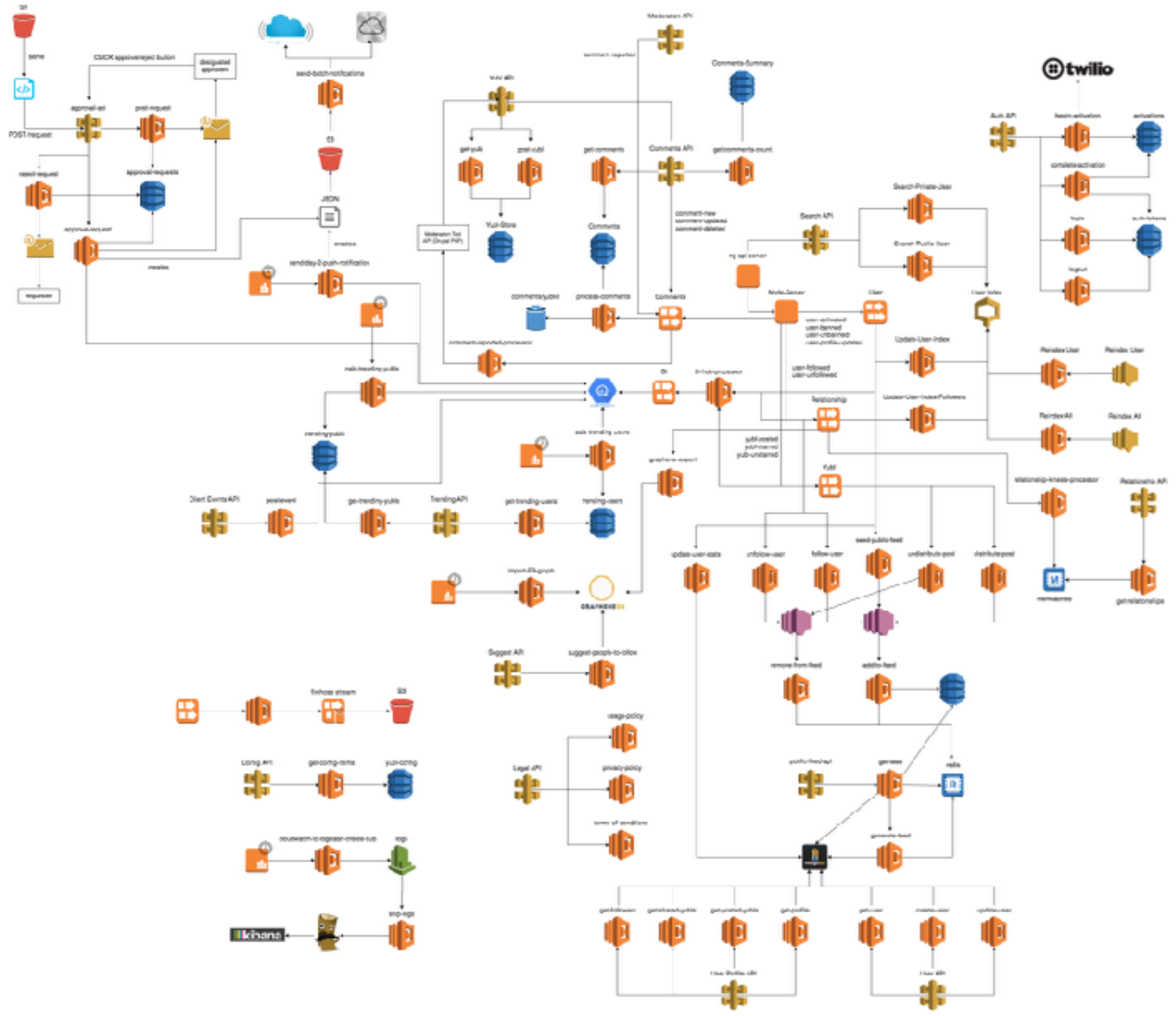
sharing code

“how do you share and reuse code between Lambda functions?”

sharing code



sharing code



sharing code

shared library VS service

shared library vs service

	shared library	service
visibility	explicit	often none

shared library vs service

	shared library	service
visibility	explicit	often none
deployment	consumer	service owner

shared library vs service

	shared library	service
visibility	explicit	often none
deployment	consumer	service owner
versioning	multiple active	singular/multiple active

shared library vs service

	shared library	service
visibility	explicit	often none
deployment	consumer	service owner
versioning	multiple active	singular/multiple active
backward compat	semantic versioning	don’t break it

shared library vs service

	shared library	service
visibility	explicit	often none
deployment	consumer	service owner
versioning	multiple active	singular/multiple active
backward compat	semantic versioning	don’t break it
isolation	internals accessible	no access to internals

shared library vs service

	shared library	service
visibility	explicit	often none
deployment	consumer	service owner
versioning	multiple active	singular/multiple active
backward compat	semantic versioning	don’t break it
isolation	internals accessible	no access to internals
failure	loud & clear	it’s complicated

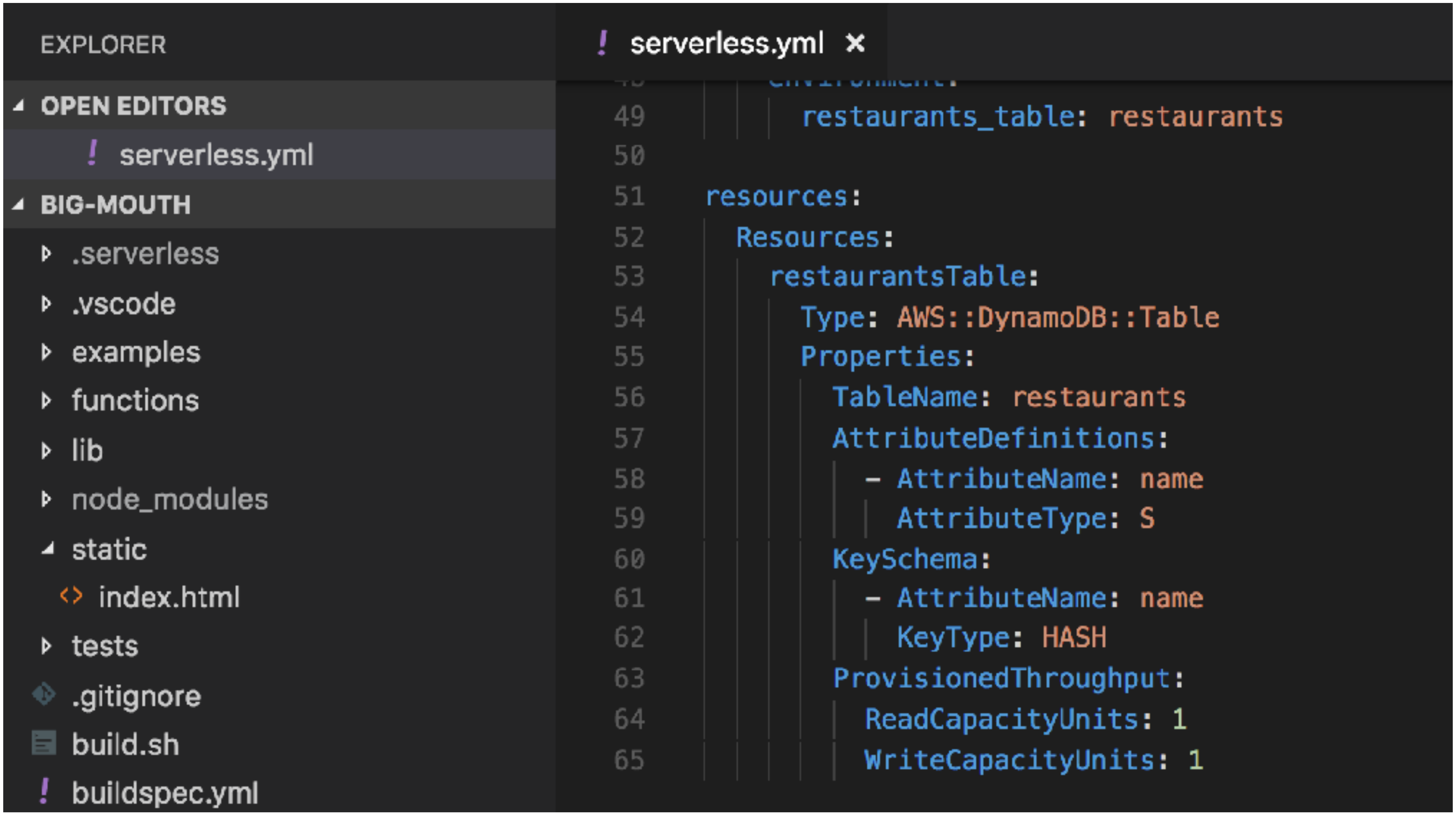
shared library vs service

	shared library	service
visibility	explicit	often none
deployment	consumer	service owner
versioning	multiple active	singular/multiple active
backward compat	semantic versioning	don’t break it
isolation	internals accessible	no access to internals
failure	loud & clear	it’s complicated

shared infrastructures

“how do you manage shared AWS resources like DynamoDB and Kinesis Streams?”

shared infrastructures



EXPLORER

OPEN EDITORS
- serverless.yml

BIG-MOUTH
- .serverless
- .vscode
- examples
- functions
- lib
- node_modules
- static
  - index.html
- tests
- .gitignore
- build.sh
- buildspec.yml

serverless.yml

```yaml
      restaurants_table: restaurants

resources:
  Resources:
    restaurantsTable:
      Type: AWS::DynamoDB::Table
      Properties:
        TableName: restaurants
        AttributeDefinitions:
          - AttributeName: name
            AttributeType: S
        KeySchema:
          - AttributeName: name
            KeyType: HASH
        ProvisionedThroughput:
          ReadCapacityUnits: 1
          WriteCapacityUnits: 1
```

serverless.yml

sls remove can delete user data

serverless.yml

lock down IAM permissions

serverless.yml

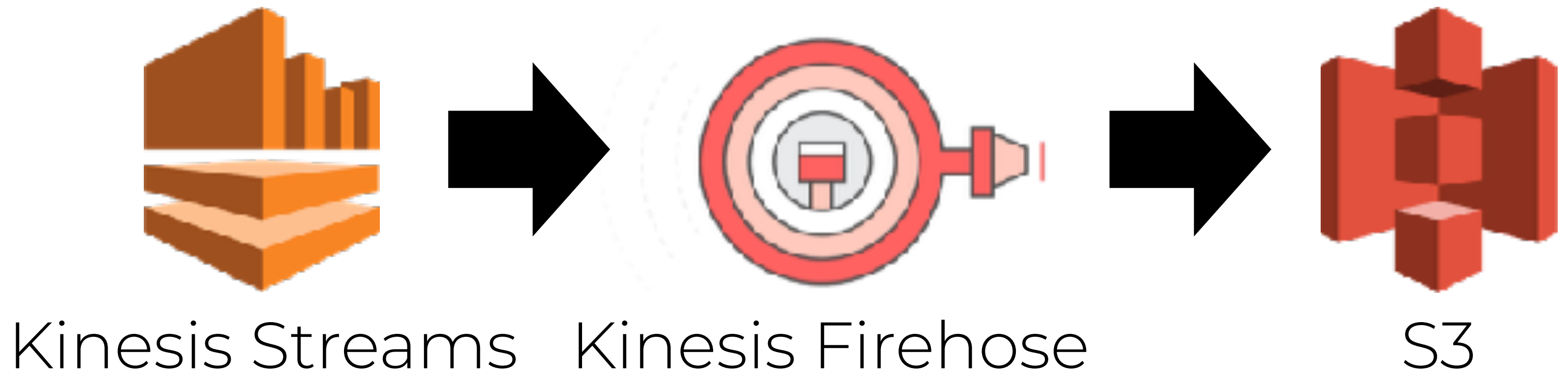
change CloudFormation's DeletionPolicy to Retain

serverless.yml

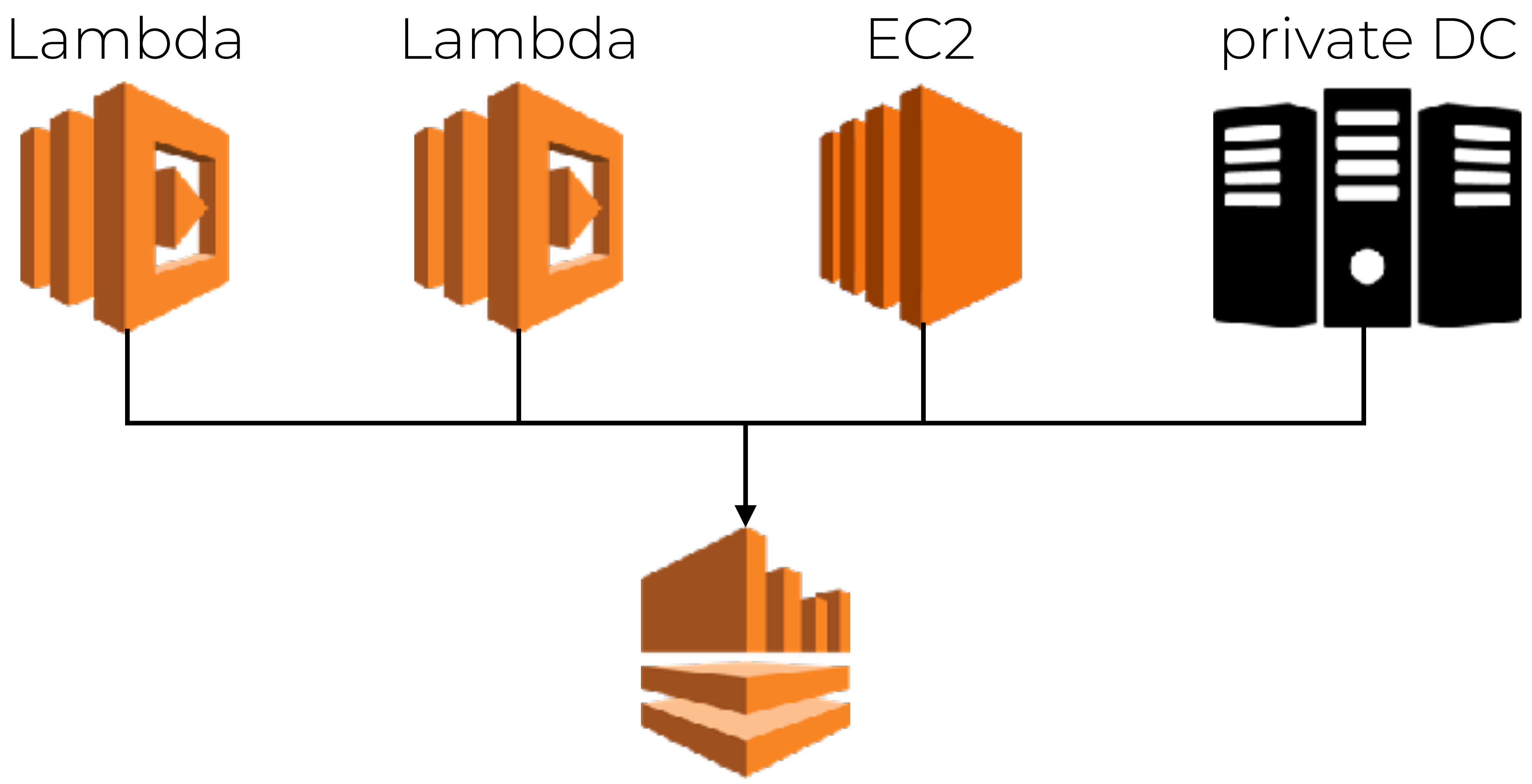
enable DynamoDB backup

serverless.yml

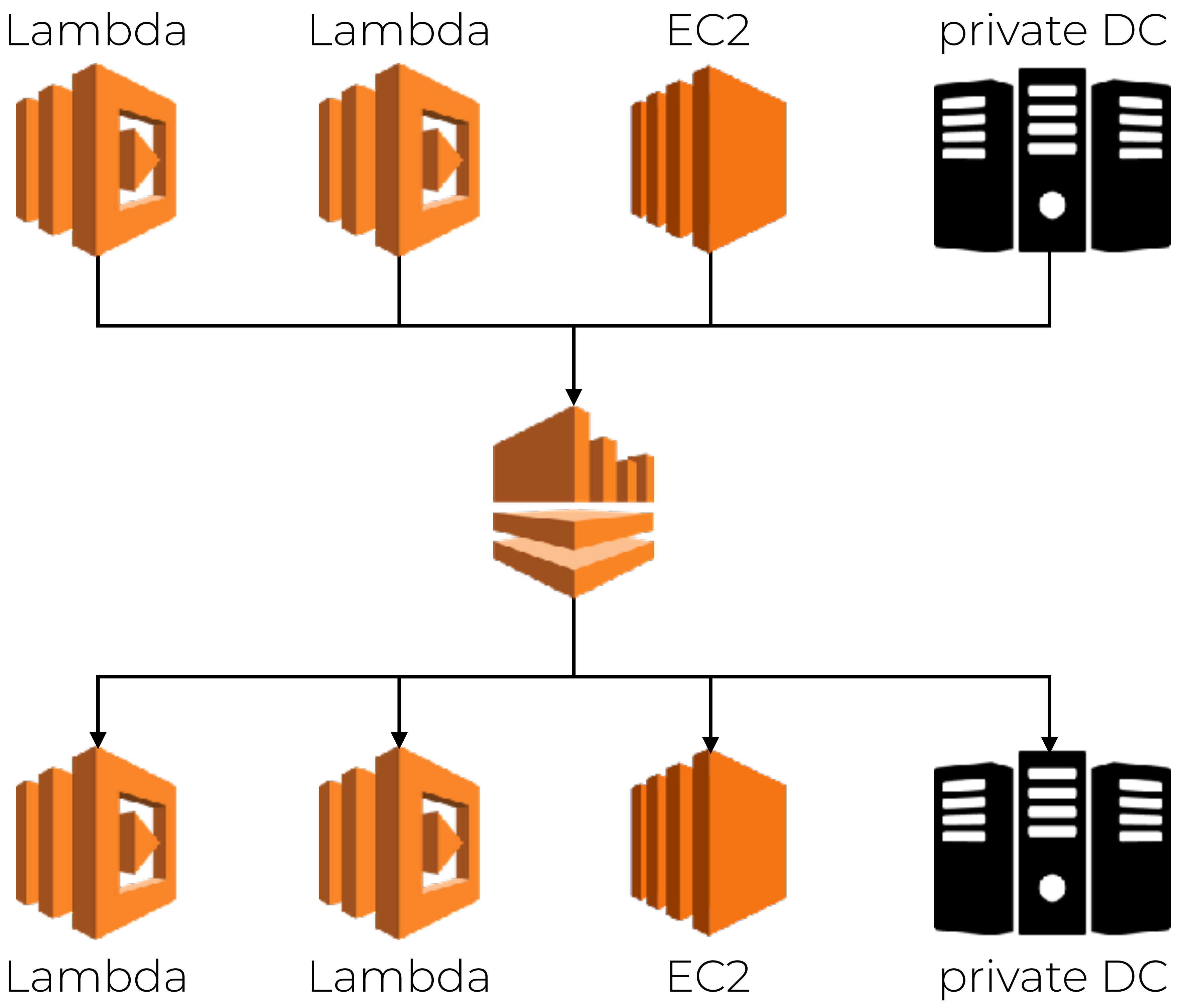
back up Kinesis events with Kinesis Firehose and S3



shared infrastructure



shared infrastructure



shared infrastructure

who owns the resource?
which project should be responsible for creating the resource?

shared infrastructure

manage shared AWS resources separately, using Terraform/CloudFormation

shared infrastructure

but it introduces other problems…

shared infrastructure

e.g. it can introduce inter-team dependencies