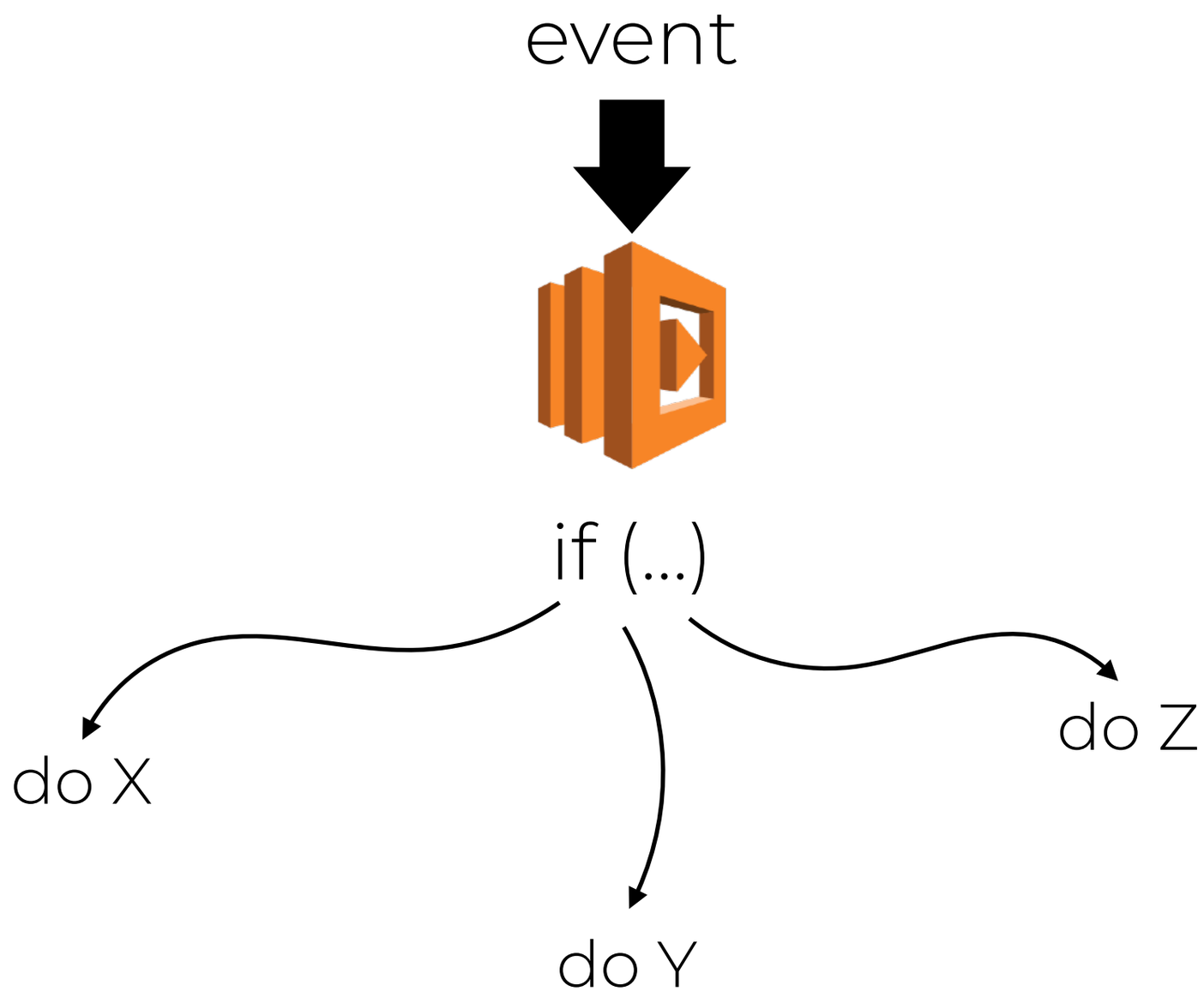
monolithic function

event

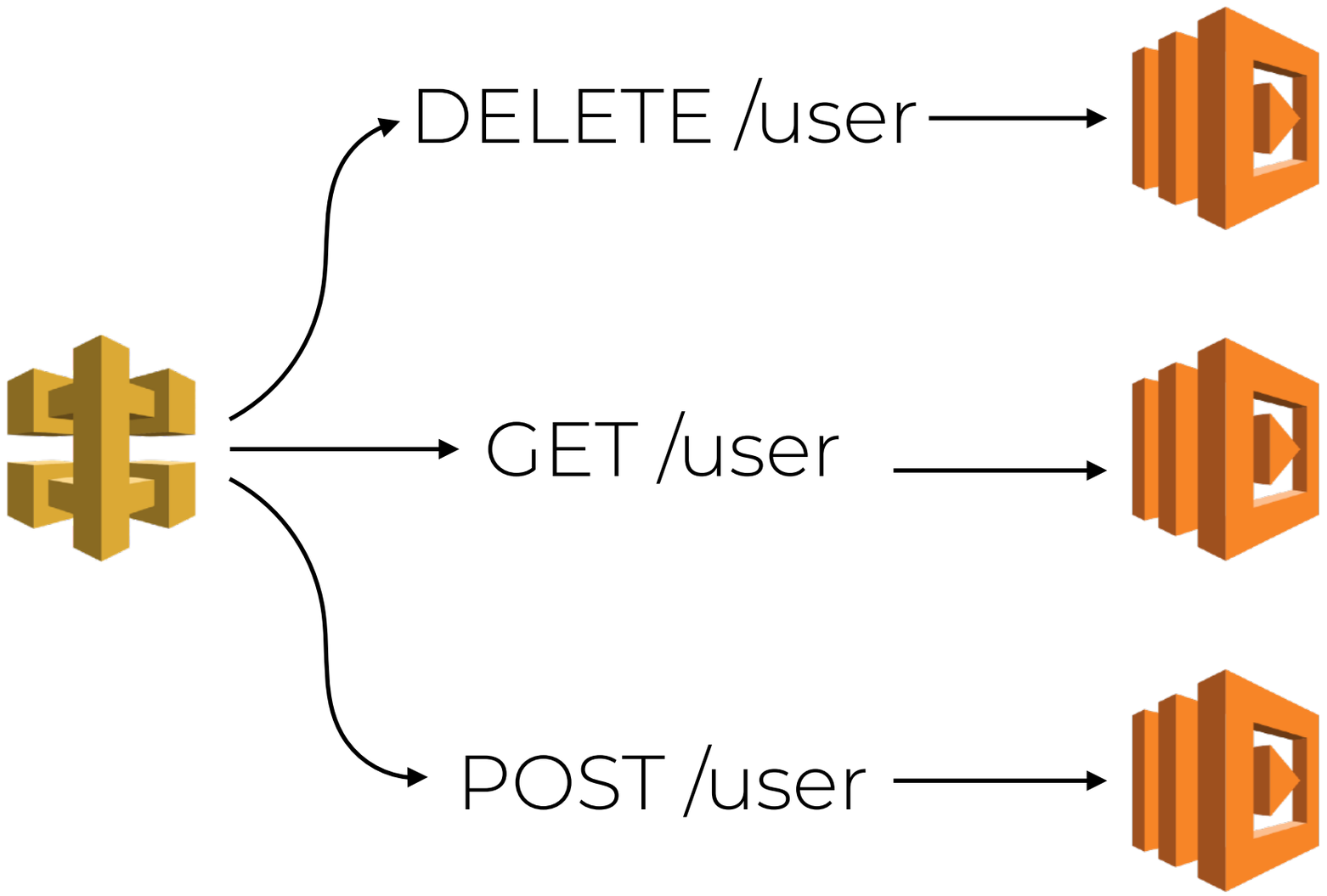
monolithic function



monolithic function


```
if (path == “/user” && method == “GET”) {
  return getUser(...);
} else if (path == “/user” && method == “DELETE”) {
  return deleteUser(...);
} else if (path == “/user” && method == “POST”) {
  return createUser(...);
} else if ....
```

single-purposed function



defining the problem

"should I have few monolithic functions, or many single-purposed functions?"

defining the problem

"should I have few monolithic functions, or many single-purposed functions?"

defining the problem

"given that we have a complex system that consists of many features and capabilities, that is maintained by many teams of developers..."

defining the problem

“given that we have a complex system that consists of many features and capabilities, that is maintained by many teams of developers…”

“how do we organise these features and capabilities into Lambda functions so that it’s optimised towards…”

defining the problem

discoverability

"how do I find out what features and capabilities exist in our system already?"

defining the problem

discoverability

"how do I find out what features and capabilities exist in our system already?"

"e.g. I'm working on a new social feature, do we have something that can tell me a user's associated Facebook ID already?"

discoverability

swardley
@swardley

Following

The worst example of duplication I've found in a Gov system is 118 workflow systems doing the same thing. In the private sector, I have a finance company with over 1,000 risk mgmt systems doing the same thing. Not convinced that State is the epitome of clueless.

> **ben goldacre, MBE LOL** @bengoldacre
> Quite. In digital health tech I often see money squandered, on digital snake oil that doesn't work, by clueless state customers. That's not just wasteful. It's a spiral of economic decline. twitter.com/mikeTakesKnigh…

2:08 PM - 24 Nov 2017

1 Retweet **7** Likes

2 1 7

discoverability

IT Weapons of Mass Duplication - Leader Board

Number of duplicate efforts in a single Organisation	Technology space	Industry
2,000+ (unconfirmed)	*Accounting systems*	*Global Defence*
1,000+ (estimated)	**Risk management**	**Global Finance**
380	ERP system	Global Energy
300+	ECM systems	Global Pharma
170	Cloud projects	Global Technology
118	Workflow systems	Government
22	Rules Engines	European Corporate
14	CRM system	National Bank
6	General Rule of Thumb	Everywhere

http://bit.ly/2m9S3cS

defining the problem

discoverability debugging

"how do I quickly identify & locate the code I need to look at to debug a problem?"

defining the problem

discoverability debugging

"how do I quickly identify & locate the code I need to look at to debug a problem?"

"e.g. there are lots of errors in the logs for system X, where is the code?"

defining the problem

discoverability debugging scaling the team

"how do I minimise friction and allow me to grow the engineering team?"

scaling

scaling

Supercell boasts 100m daily active users

"It's incredible that we've reached this milestone with just 180 people," says Supercell CEO Ilkka Paananen

Clash of Clans and Boom Beach developer Supercell proudly announced today that its daily active users count has hit 100 million. CEO Ilkka Paananen shared the news on Twitter and thanked players around the world with a celebratory video (see below).

James Brightman
Editor, North America
Monday 7th March 2016

COMPANIES IN THIS ARTICLE

Supercell

"I also want to give my thanks to all the Supercellians. It's incredible that we've reached this milestone with just 180 people. Every single person at Supercell has contributed towards reaching this milestone and I feel proud and lucky to be a part of the team," he said.

"What's next? Well, we founded Supercell with this idea of making it the best place for the best people to make the best games. A zero bureaucracy environment where people can just focus on creating great games. I feel that if we stay focused on this very idea, with some luck, even better games will follow over the years to come."

scaling

twitter

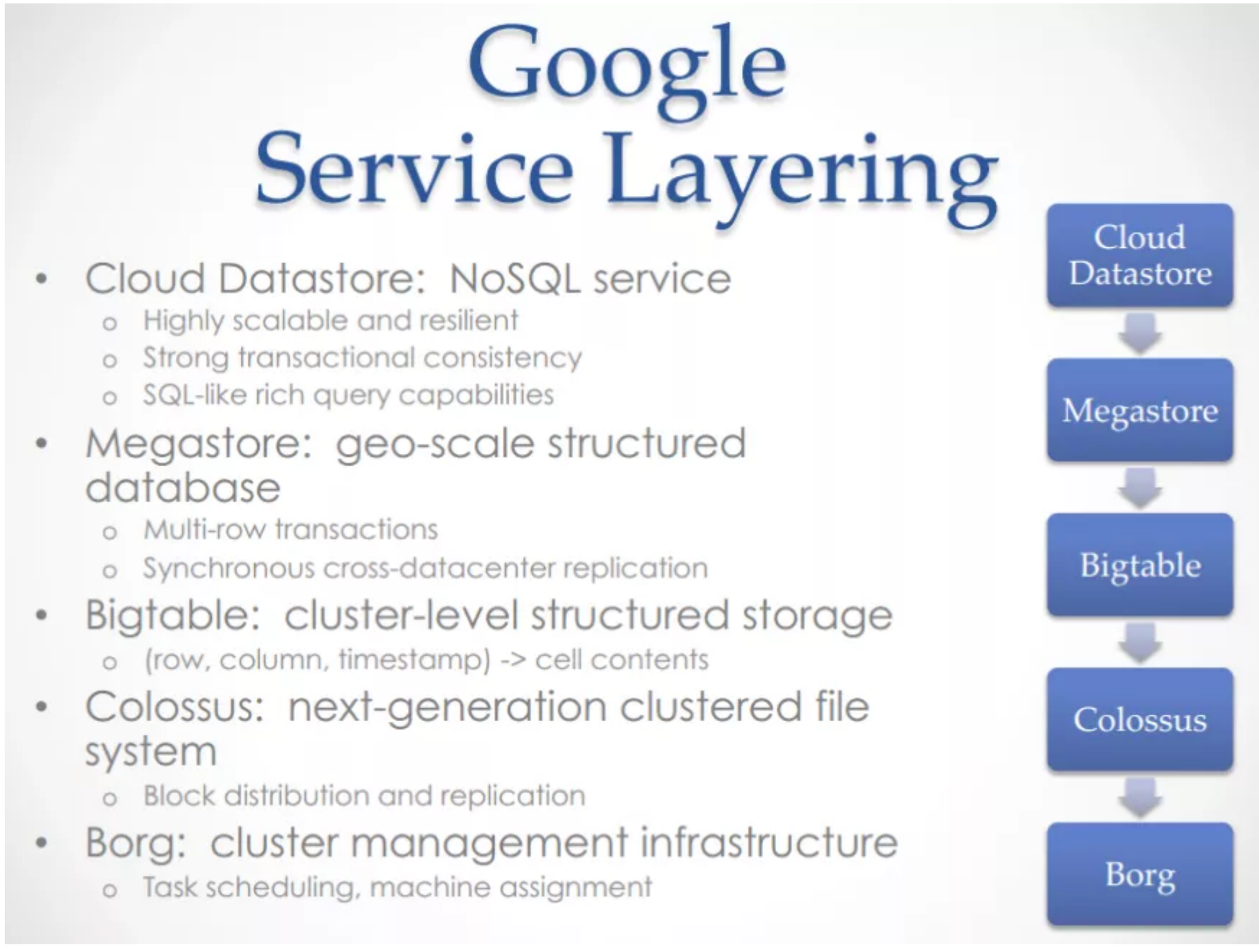
amazon

NETFLIX

UBER

Google

scaling

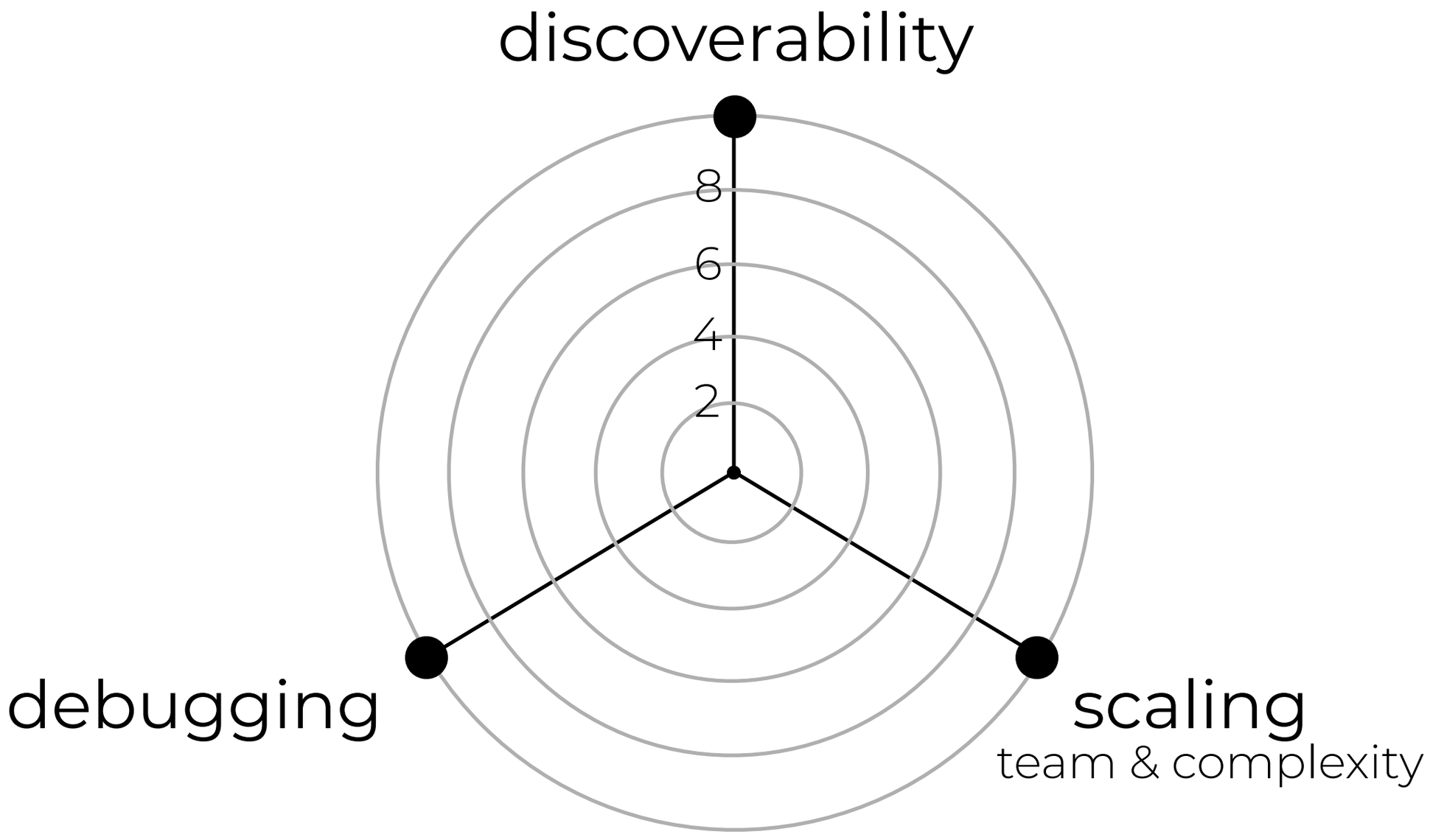


http://bit.ly/2CQx3C4

defining the problem

"given that we have a complex system that consists of many features and capabilities, that is maintained by many teams of developers, how do we organise these features and capabilities into Lambda functions so that it's optimised towards **discoverability**, ease of **debugging**, and the ability to **scale** the engineering team and the complexity of the system"

defining the problem



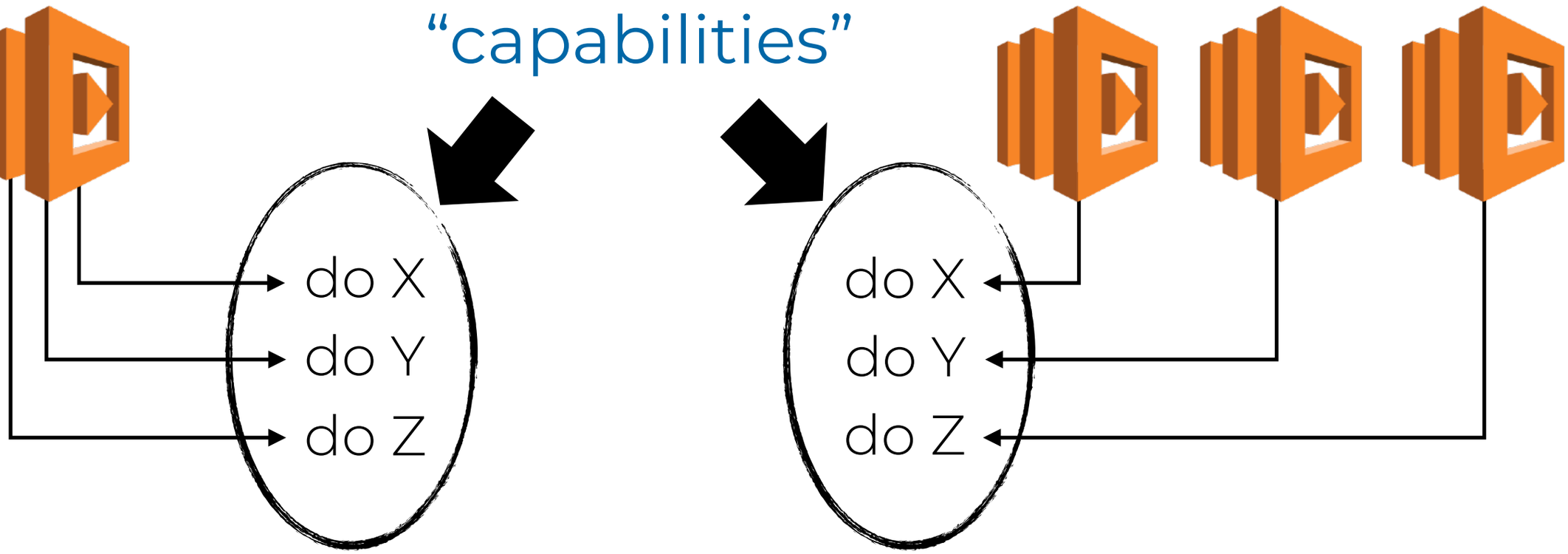
discoverability

“capabilities”

discoverability

monolithic

single-purposed



discoverability

"we chose monolithic functions because it reduces the **no. of functions** we have to manage in our environment, making them easier to find."

discoverability

discoverability

monolithic

user-api-dev

single-purposed

user-api-dev-get-user

user-api-dev-create-user

user-api-dev-delete-user

discoverability

monolithic

user-api-dev

single-purposed

user-api-dev-get-user

user-api-dev-create-user

user-api-dev-delete-user

discoverability

monolithic

author: yan.cui

feature: user-api

user-api-dev

single-purposed

author: yan.cui

feature: user-api

user-api-dev-get-user

author: yan.cui

feature: user-api

user-api-dev-create-user

author: yan.cui

feature: user-api

user-api-dev-delete-user

discoverability

monolithic

user-api-dev

?

single-purposed

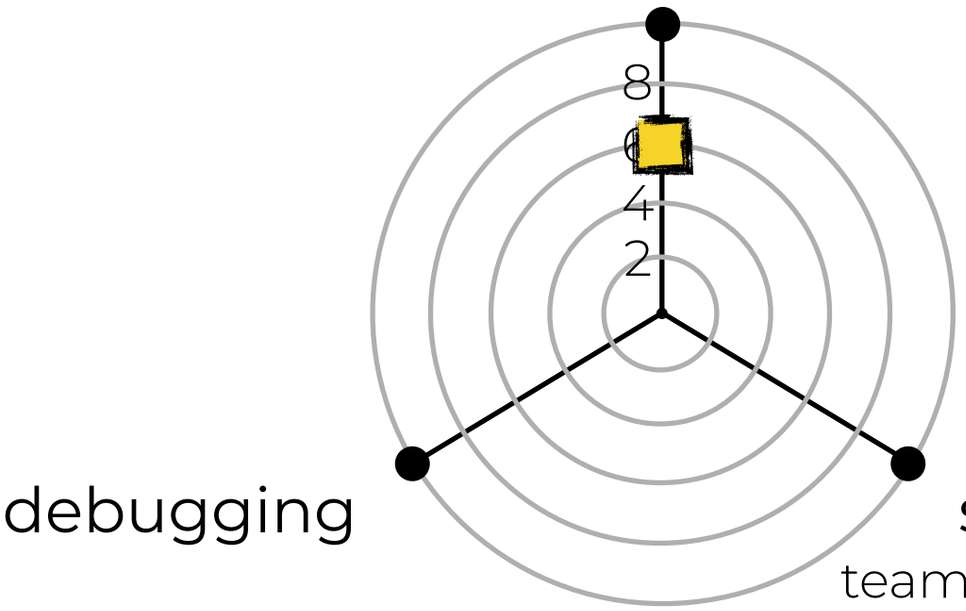
user-api-dev-get-user

user-api-dev-create-user

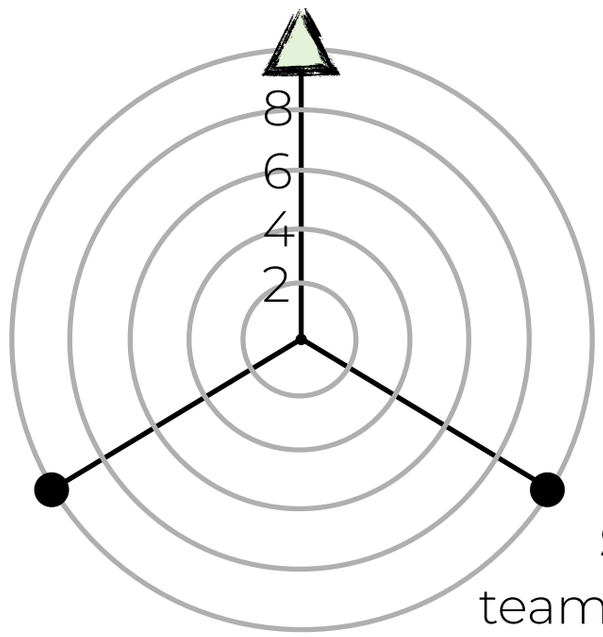
user-api-dev-delete-user

monolith vs single-purposed

monolithic



single-purposed



discoverability

Functions (47)

Actions ▼ | Create function

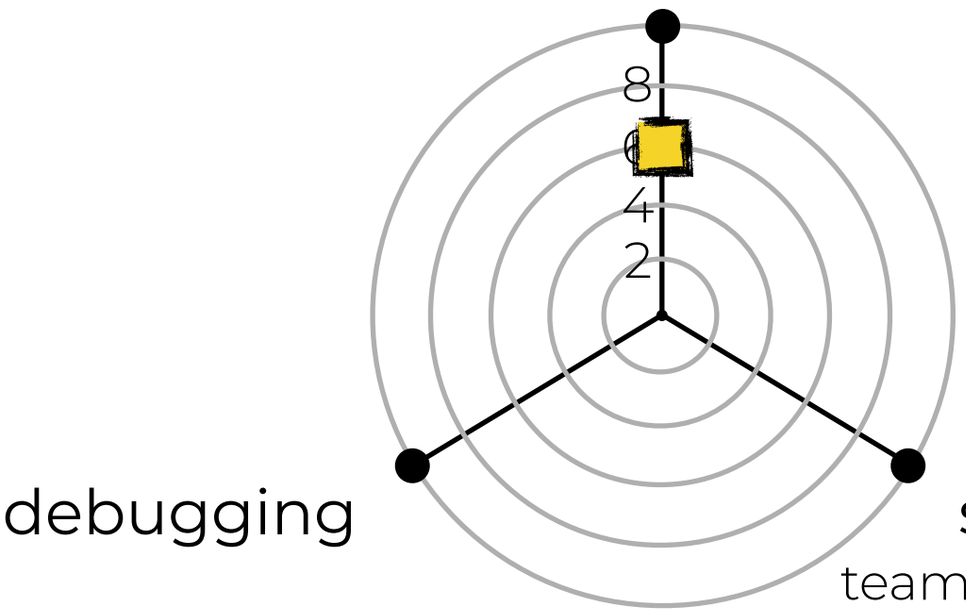
Filter by tags and attributes or search by keyword

1 2 3 4 5

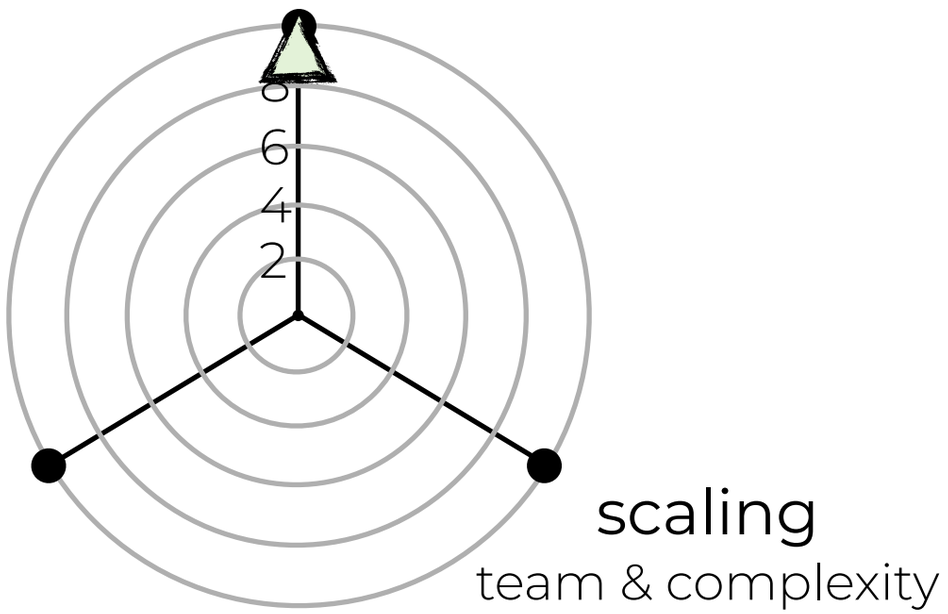
Function name	Description	Runtime	Code size	Last Modified
big-mouth-dev-get-restaurants		Node.js 6.10	699.9 kB	2 days ago
big-mouth-dev-get-index		Node.js 6.10	699.9 kB	2 days ago
big-mouth-dev-search-restaurants		Node.js 6.10	699.9 kB	2 days ago
manning-lambda-course-dev-get-restaurants		Node.js 6.10	712.1 kB	2 months ago
manning-lambda-course-dev-get-index		Node.js 6.10	712.1 kB	2 months ago
latency-injection-demo-dev-public-api-a		Node.js 6.10	2.6 MB	2 months ago
latency-injection-demo-dev-internal-api		Node.js 6.10	2.6 MB	2 months ago
latency-injection-demo-dev-public-api-b		Node.js 6.10	2.6 MB	2 months ago
canary-demo-prod-get		Node.js 6.10	1.2 kB	2 months ago
canary-demo-canary-get		Node.js 6.10	1.2 kB	2 months ago

monolith vs single-purposed

monolithic



single-purposed

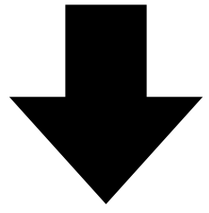


debugging

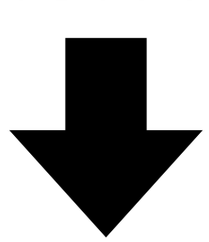
“how do I quickly identify & locate the code I need to look at to debug a problem?”

debugging

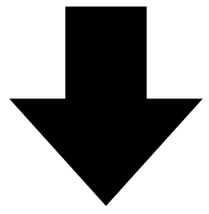
error message, stack trace in logs, alarms, etc.



function name



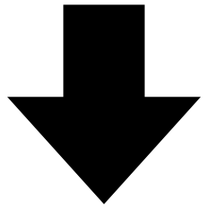
code repo



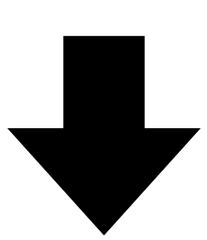
find relevant code that needs fixing

debugging

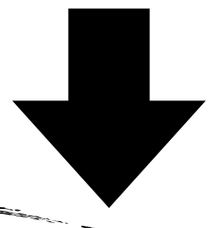
error message, stack trace in logs, alarms, etc.



function name



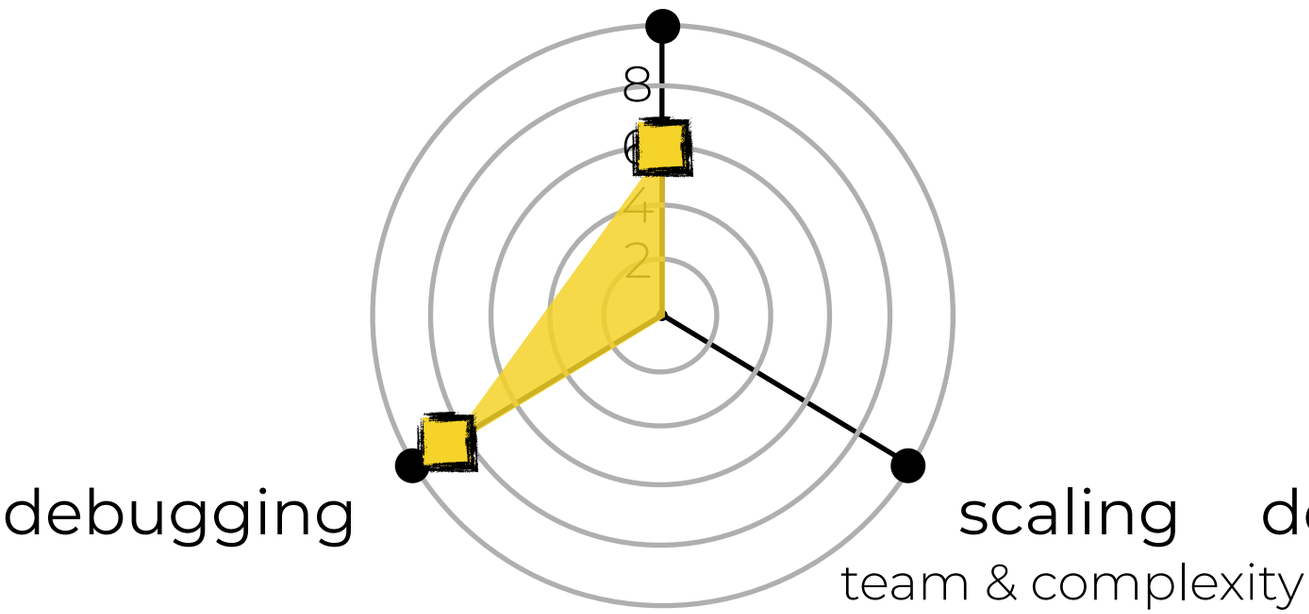
code repo



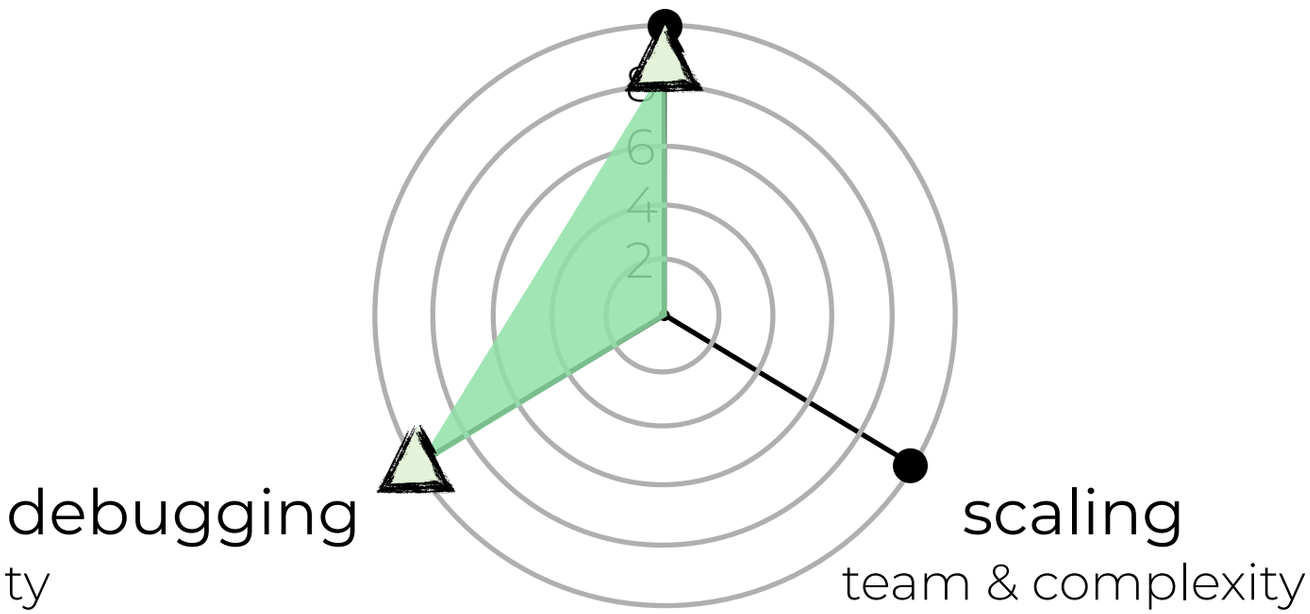
find relevant code that needs fixing

monolith vs single-purposed

monolithic



single-purposed



scaling team & complexity

service boundaries (microservices) enable greater division of labour, so you can grow the team without everyone constantly tripping over each other with merge conflicts and integration problems

scaling team & complexity

"if you have a high coherence penalty and too many people, then the team as a whole moves slower... It's about **reducing the overhead of sharing mental models**."

https://bit.ly/coherence-penalty

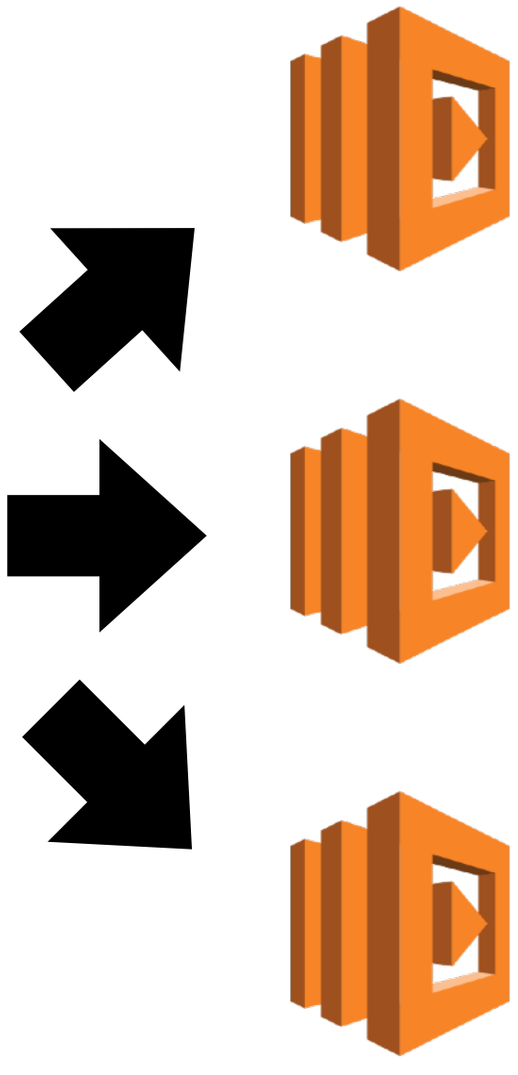
Michael Nygard

scaling team & complexity

monolithic

division of task

single-purposed

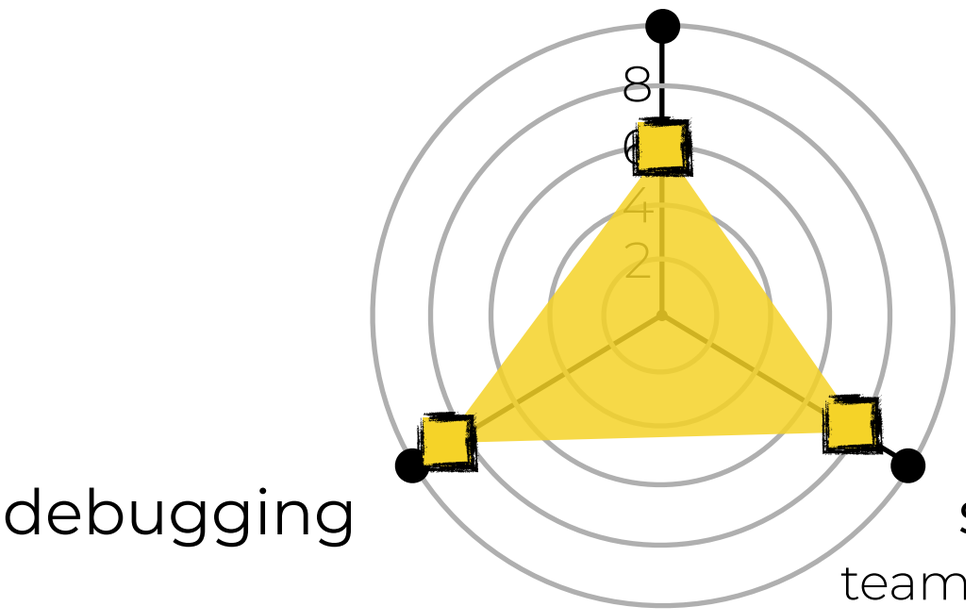


scaling team & complexity

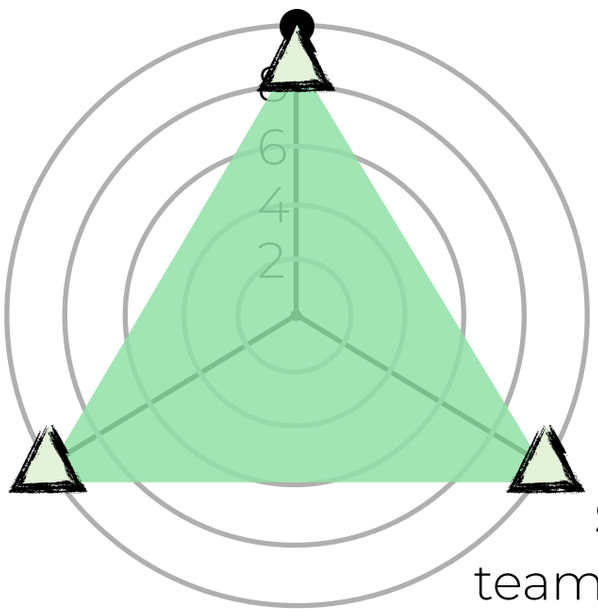
having a single purpose also caps the complexity ceiling of a function, and encourages use of composition to scale complexity

monolith vs single-purposed

monolithic



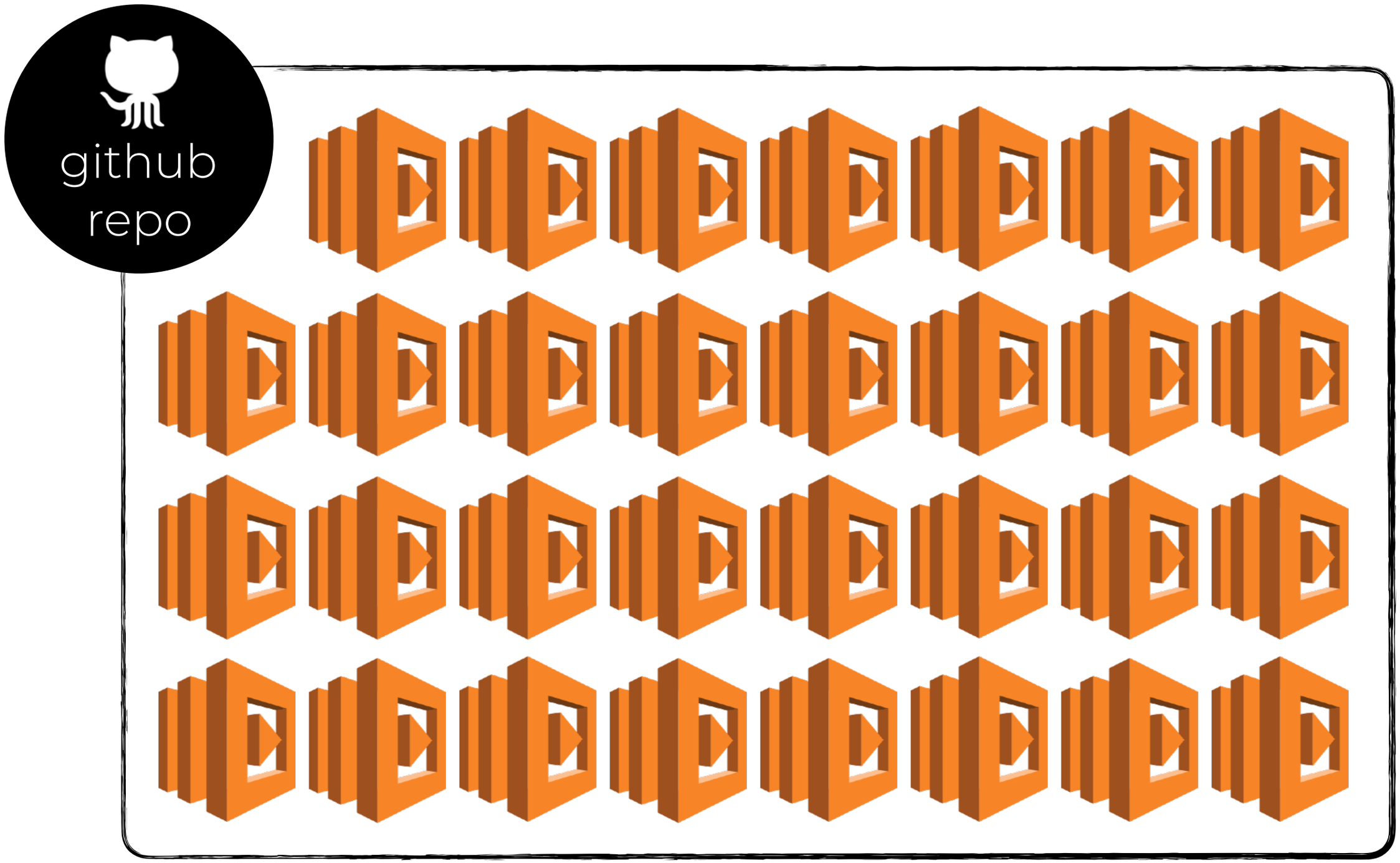
single-purposed



organizing functions

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

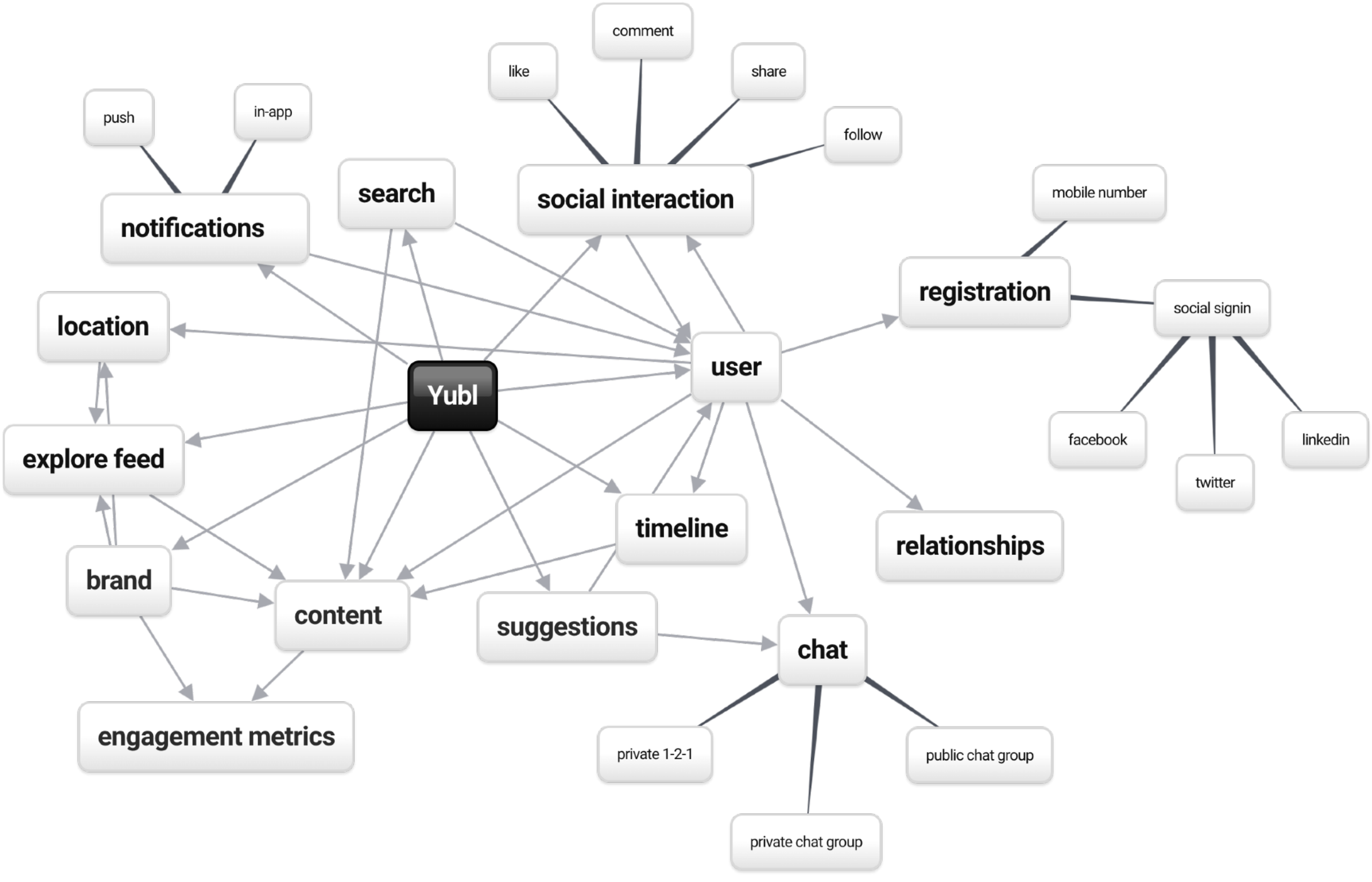
monolithic



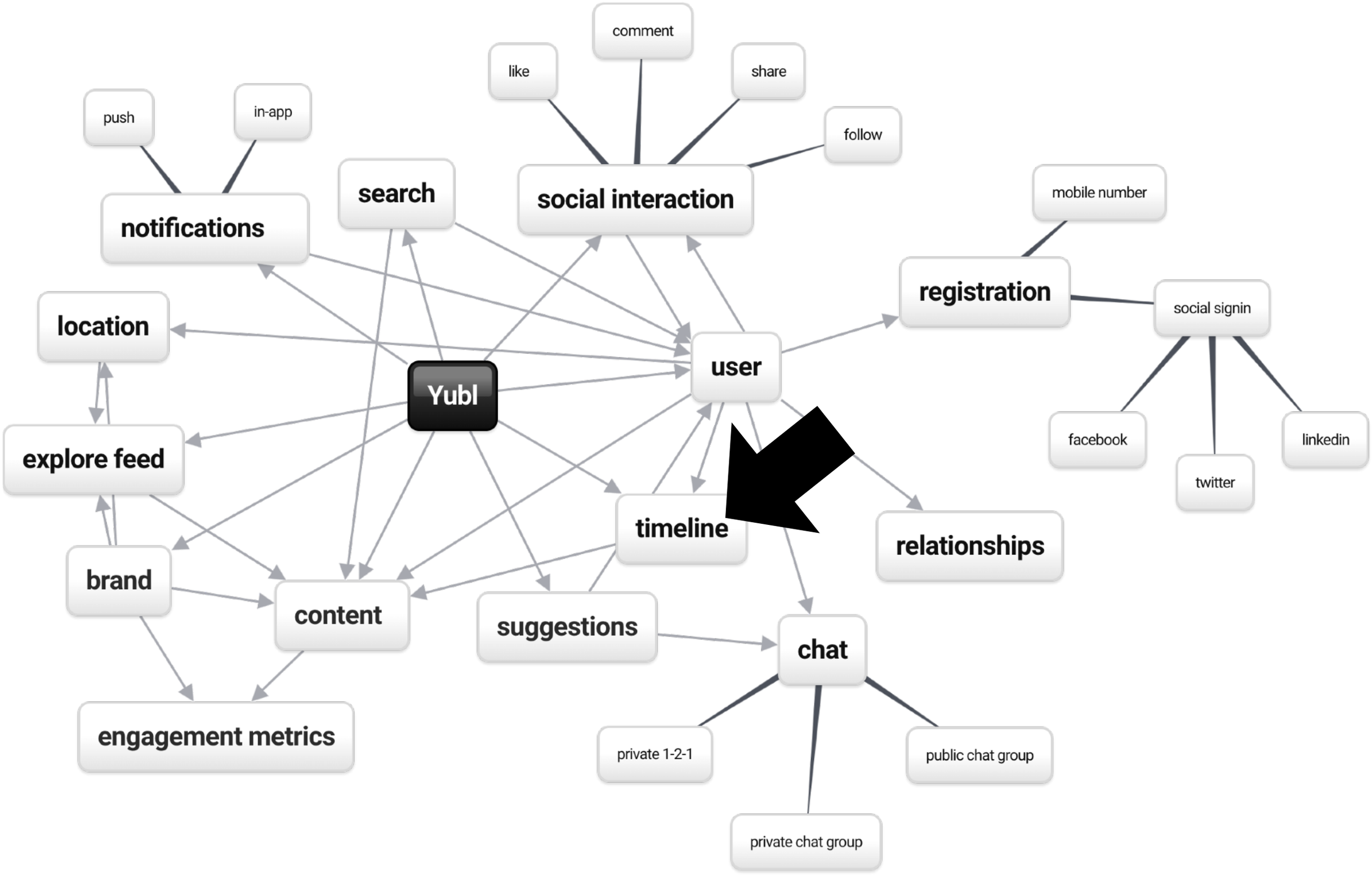
monolithic

don’t do it…

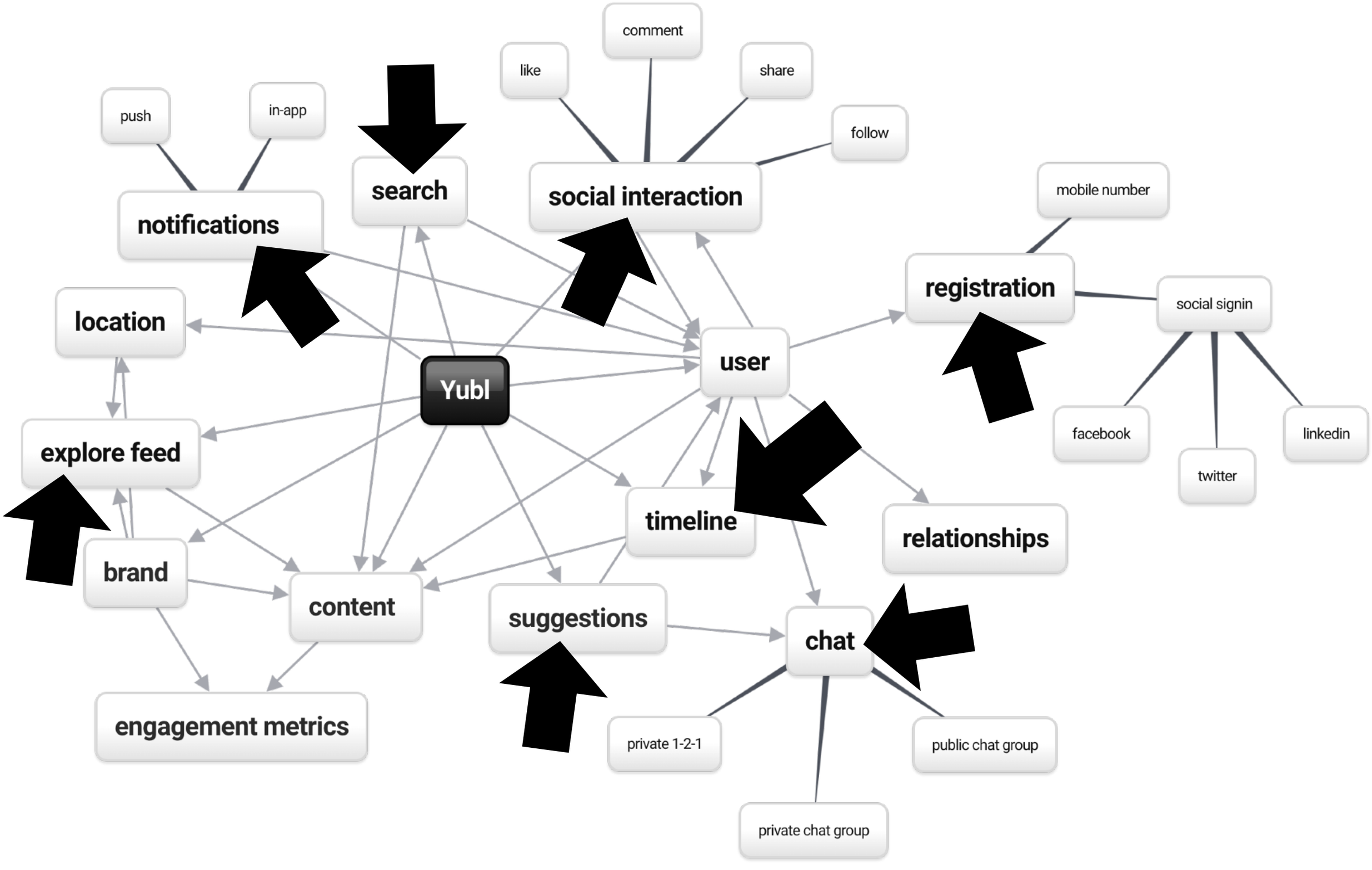
microservices



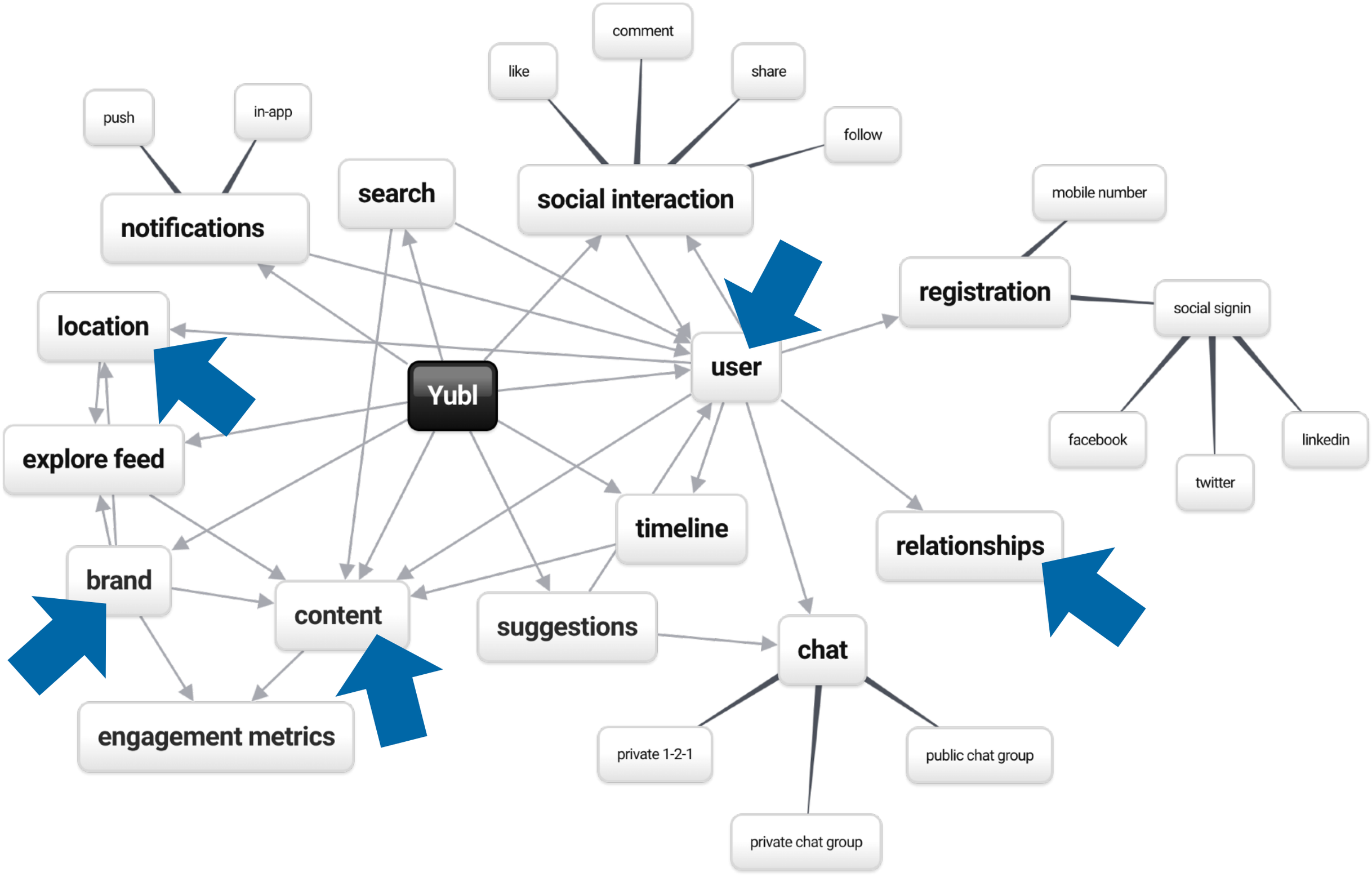
microservices



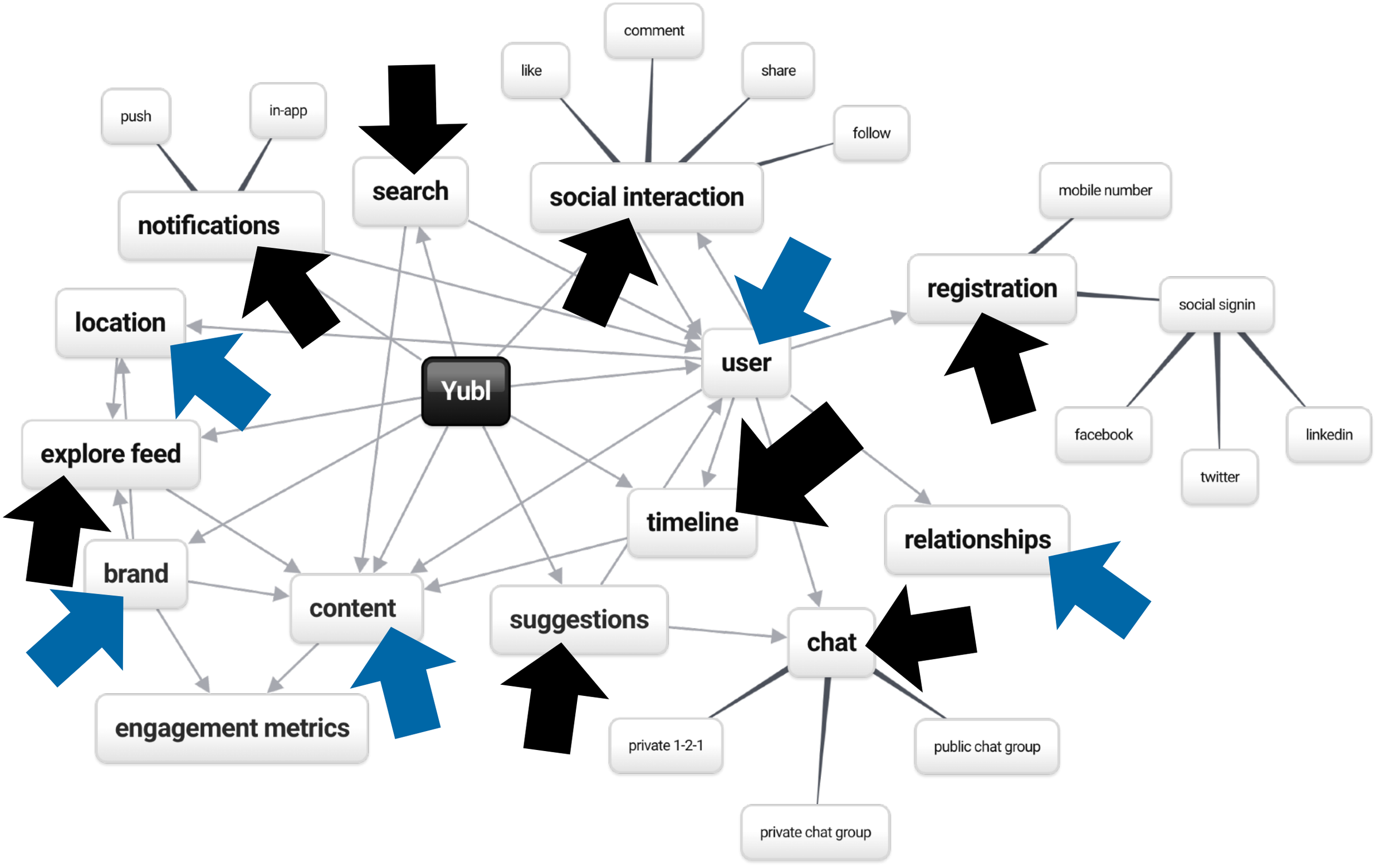
microservices



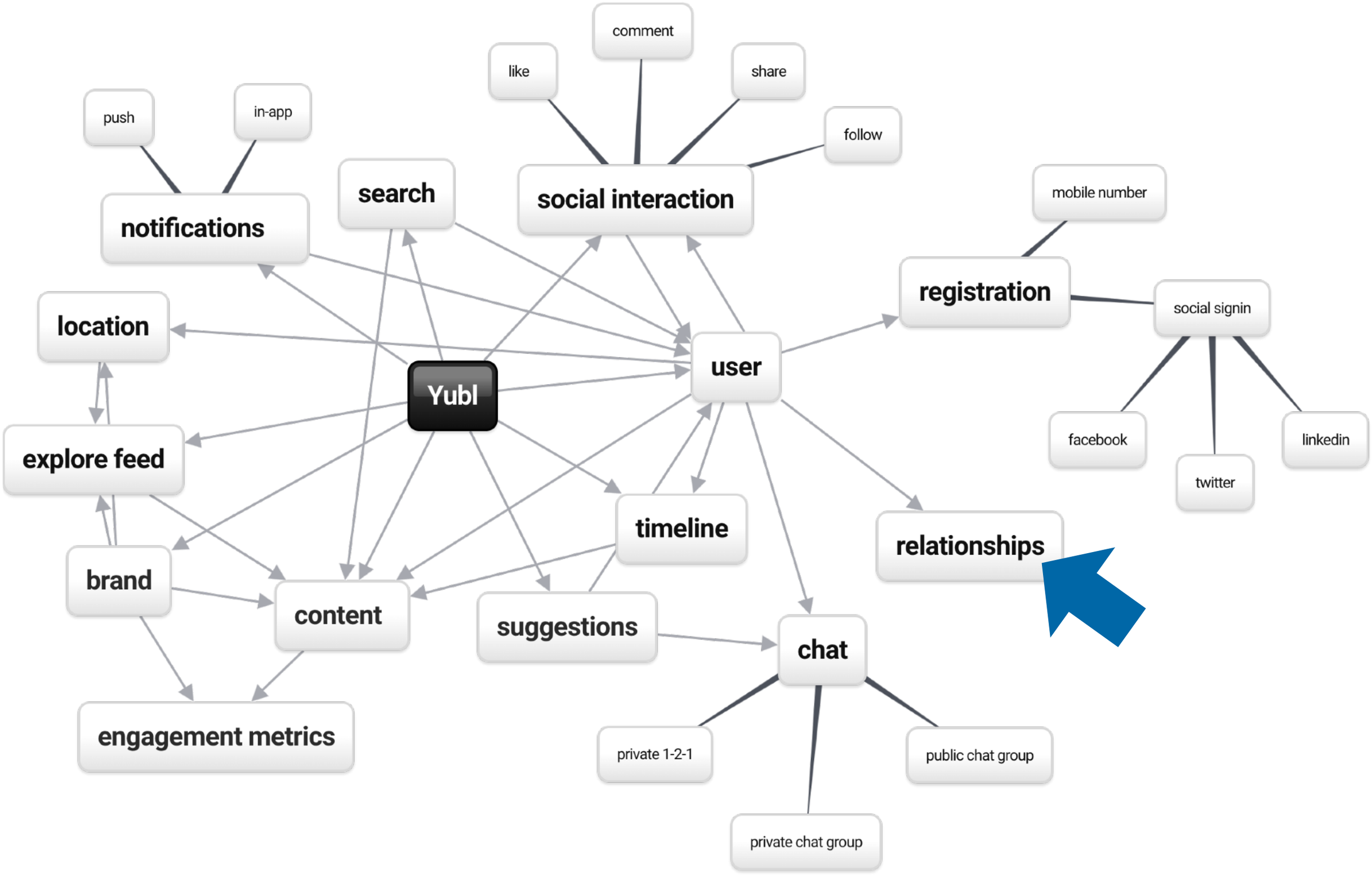
microservices



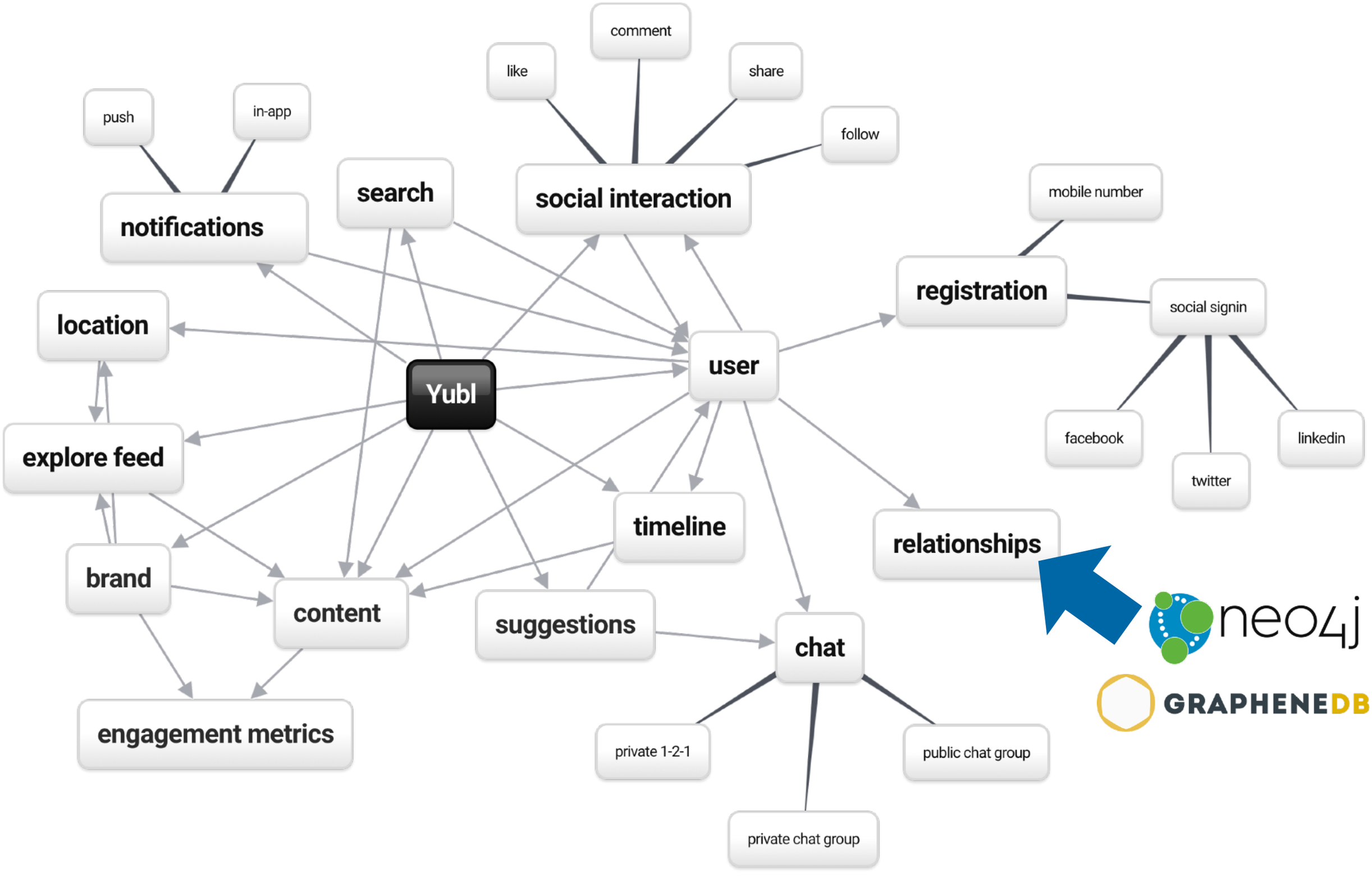
microservices



microservices



microservices



microservices

comment

like

share

loc

explor

linkedin

private chat group

Amazon Neptune

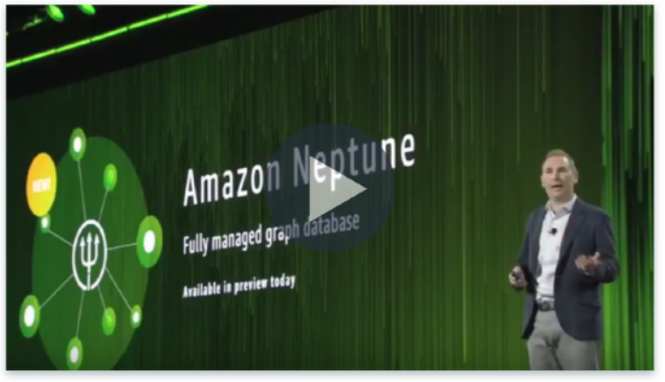
Fast, reliable graph database built for the cloud

Sign up for Preview

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.

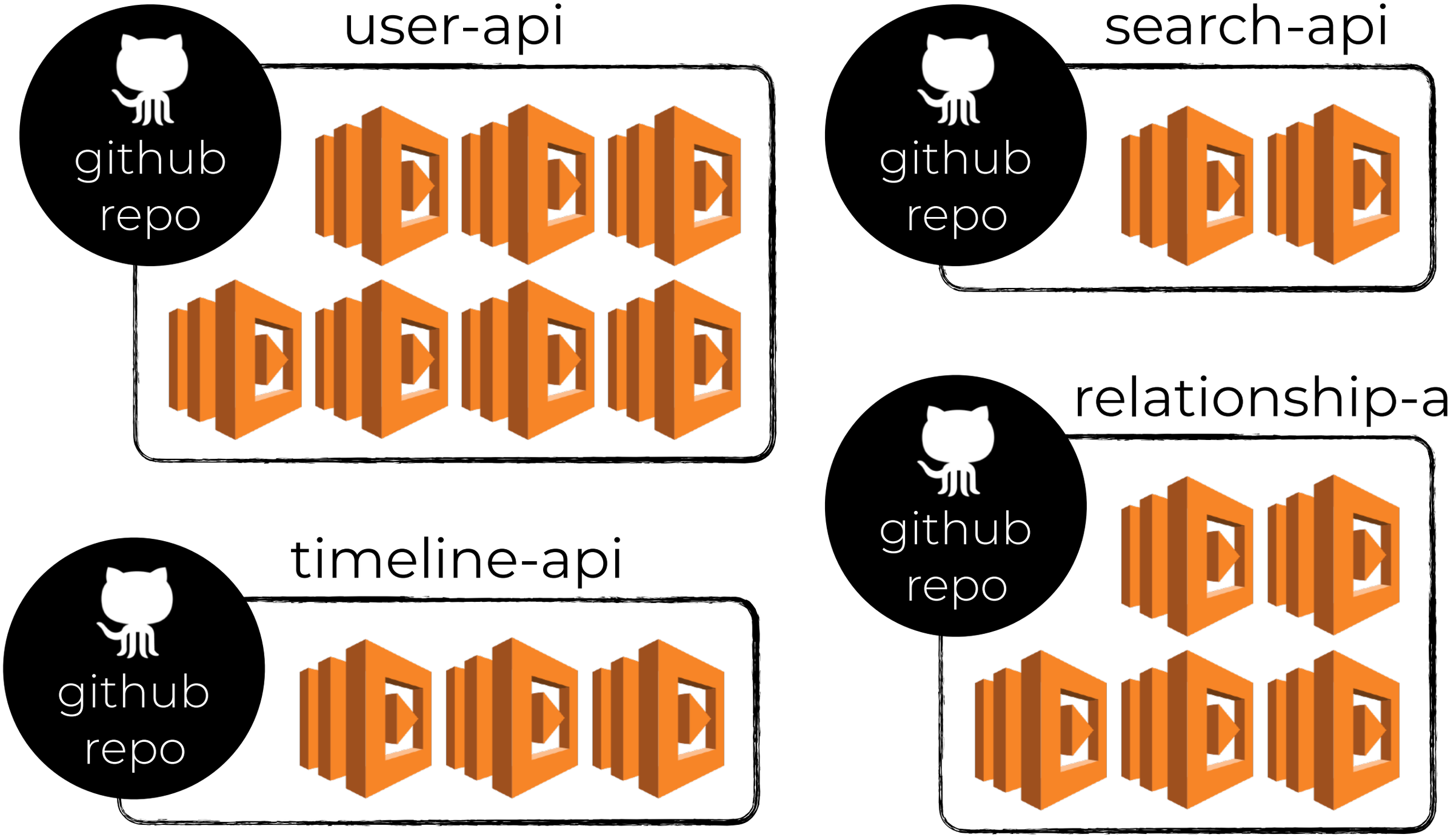
Amazon Neptune is highly available, with read replicas, point-in-time recovery, continuous backup to Amazon S3, and replication across Availability Zones. Neptune is secure, with support for encryption at rest and in transit. Neptune is fully-managed, so you no longer need to worry about database management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

Sign up for the Amazon Neptune preview here.

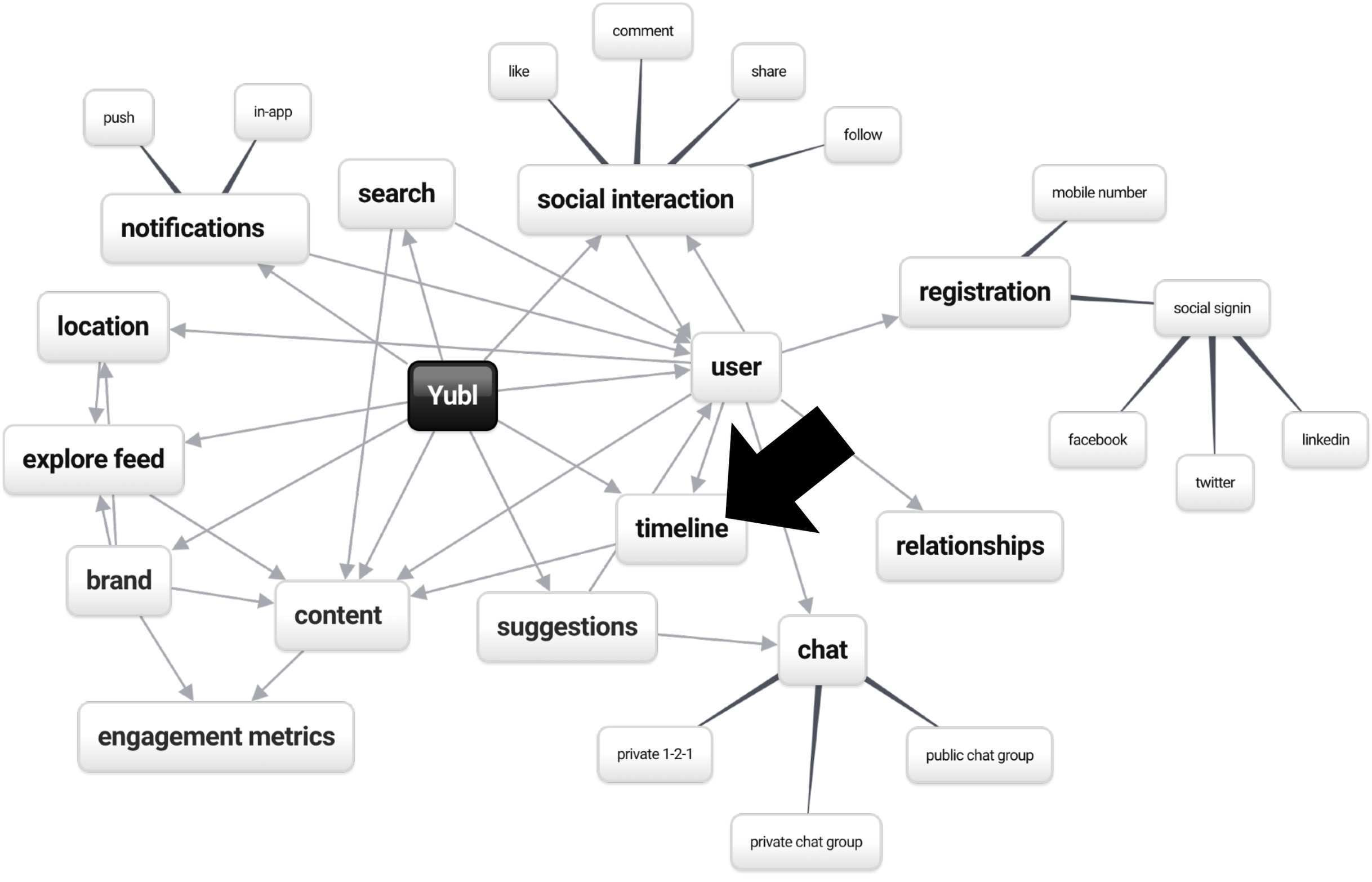


Amazon Neptune announcement at AWS re:Invent 2017

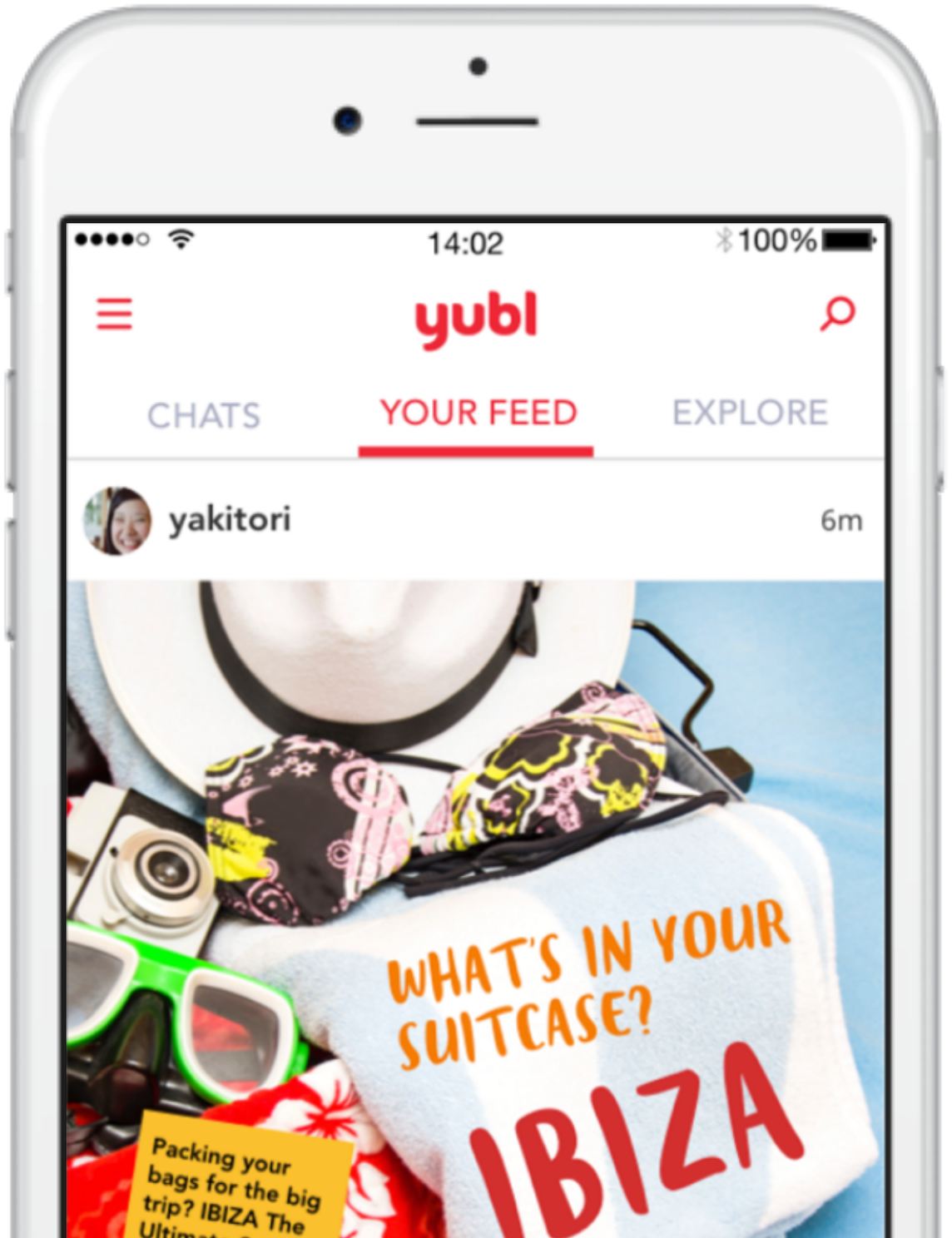
microservices



timeline feature

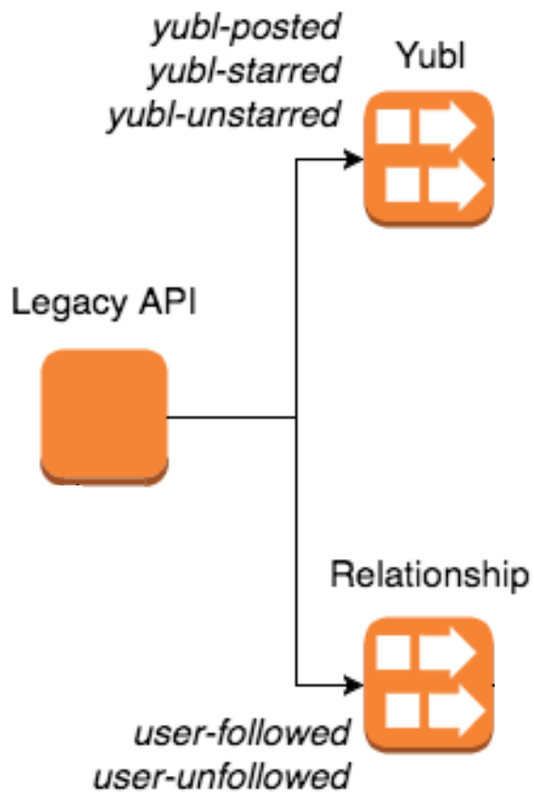


timeline feature

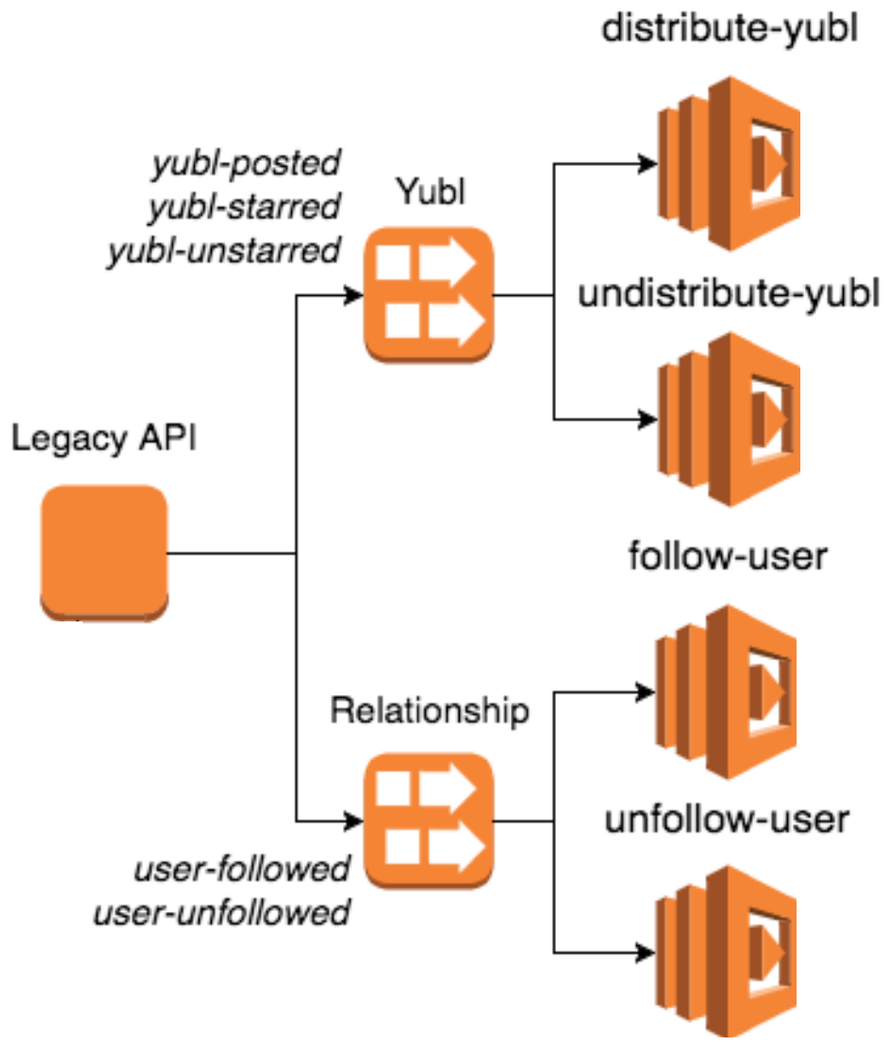


timeline feature

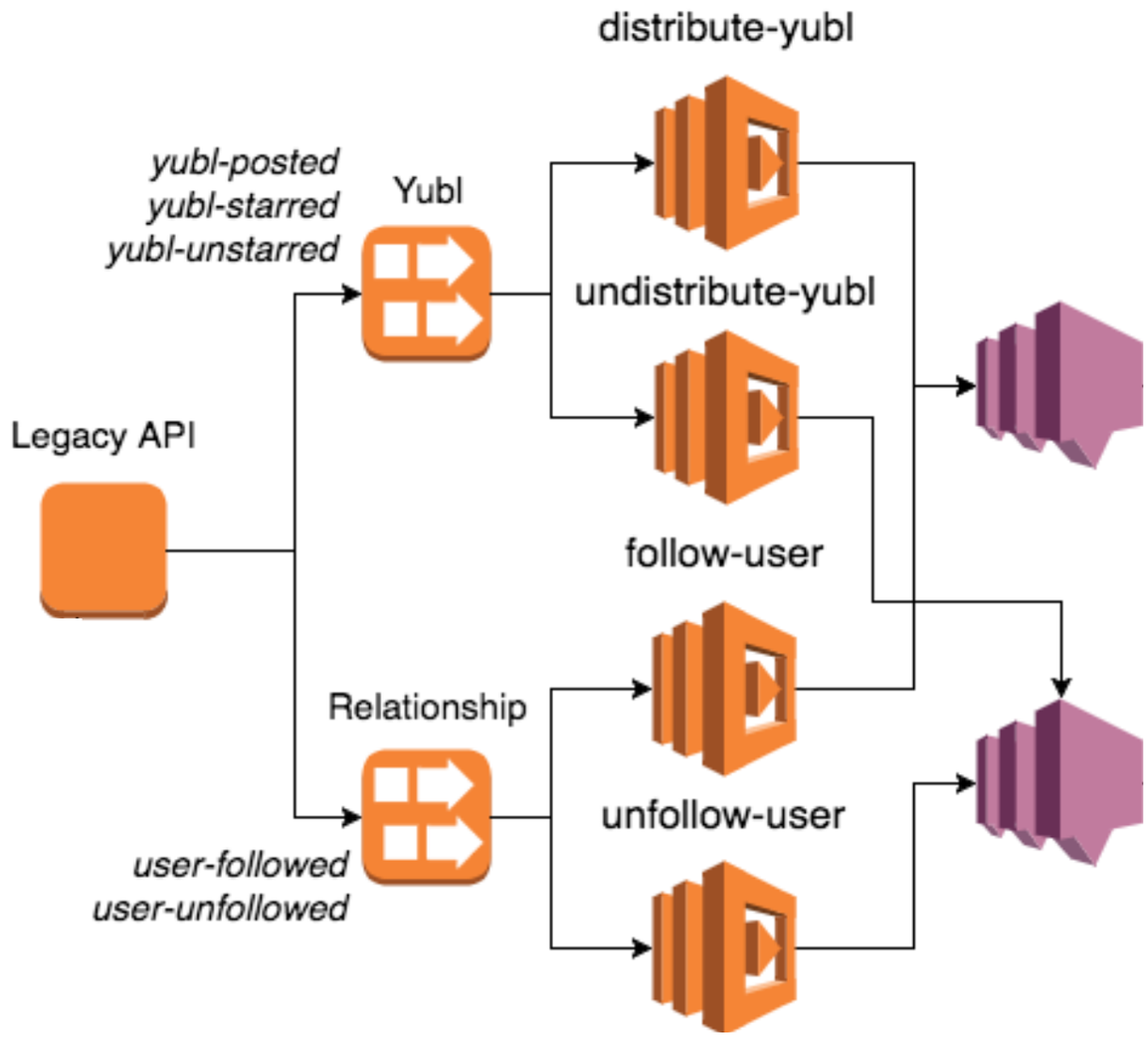
timeline feature



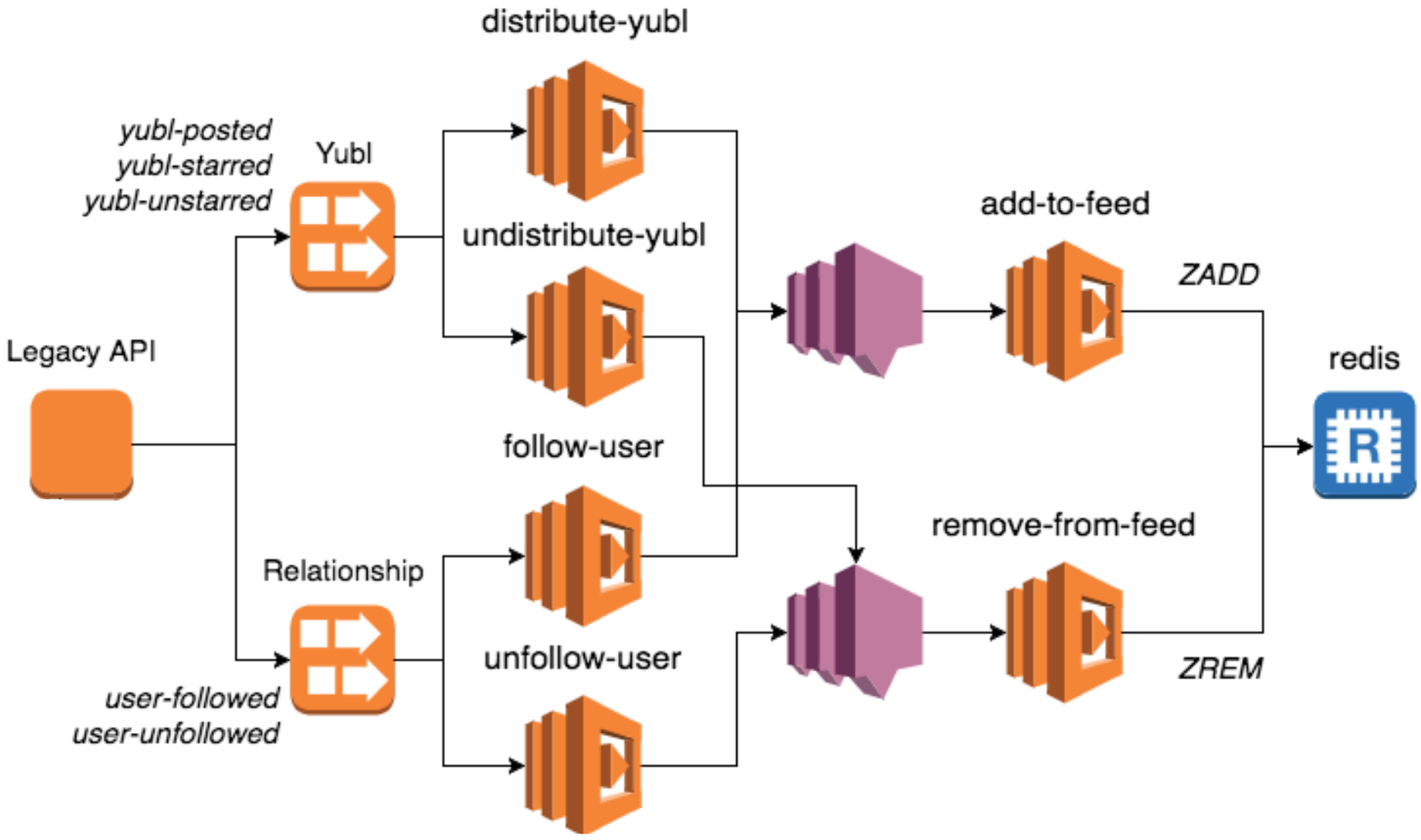
timeline feature



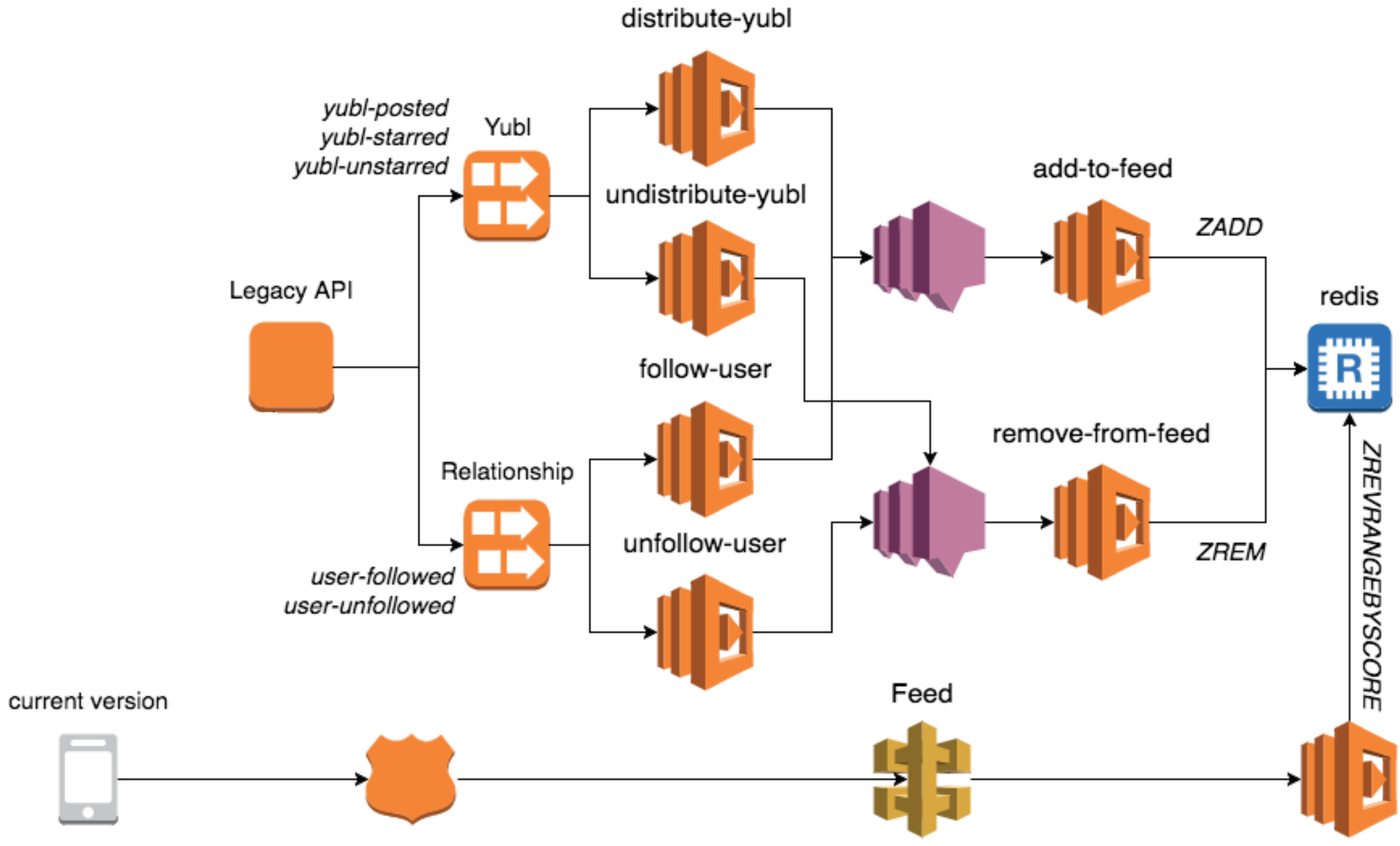
timeline feature



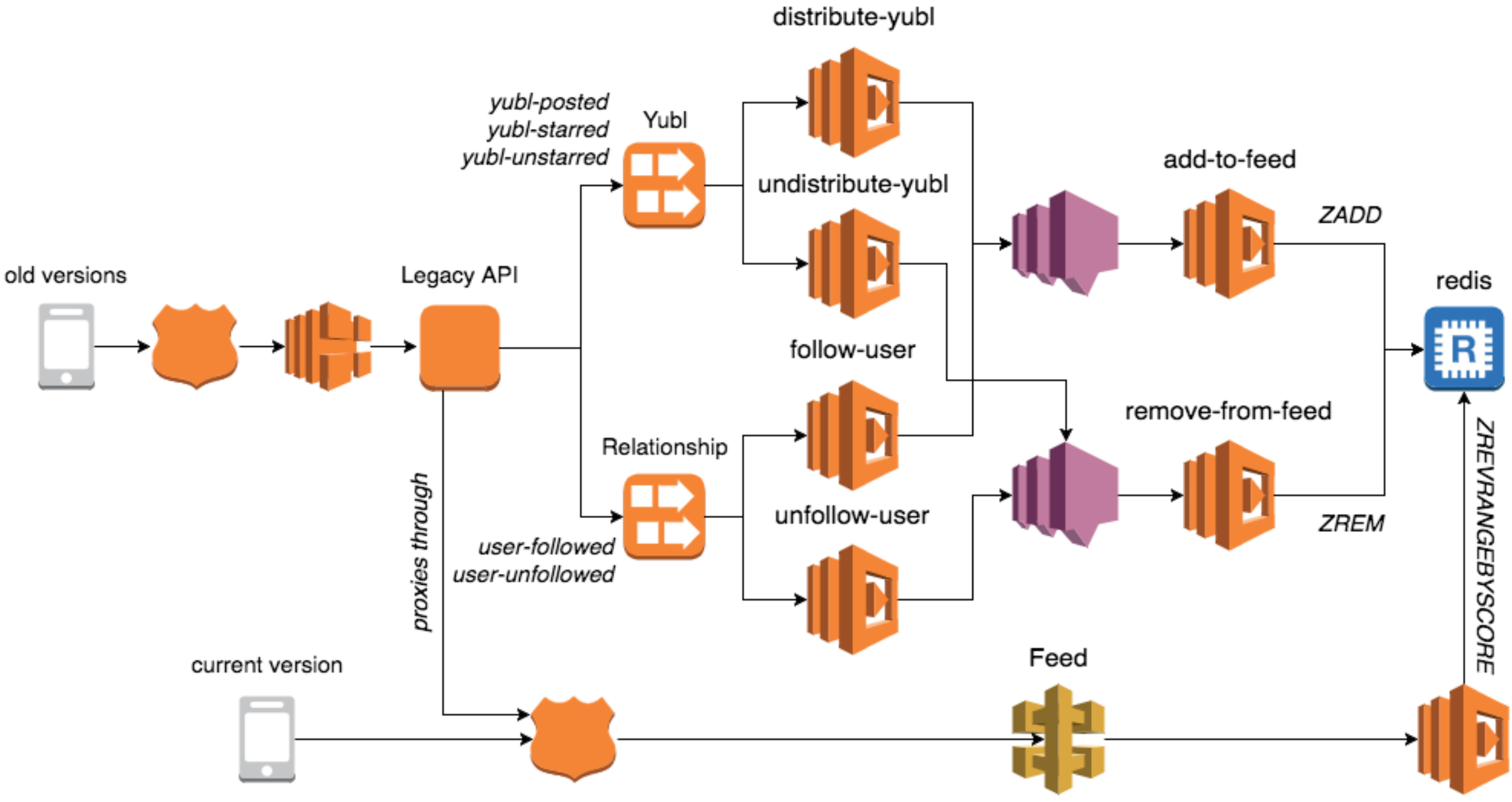
timeline feature



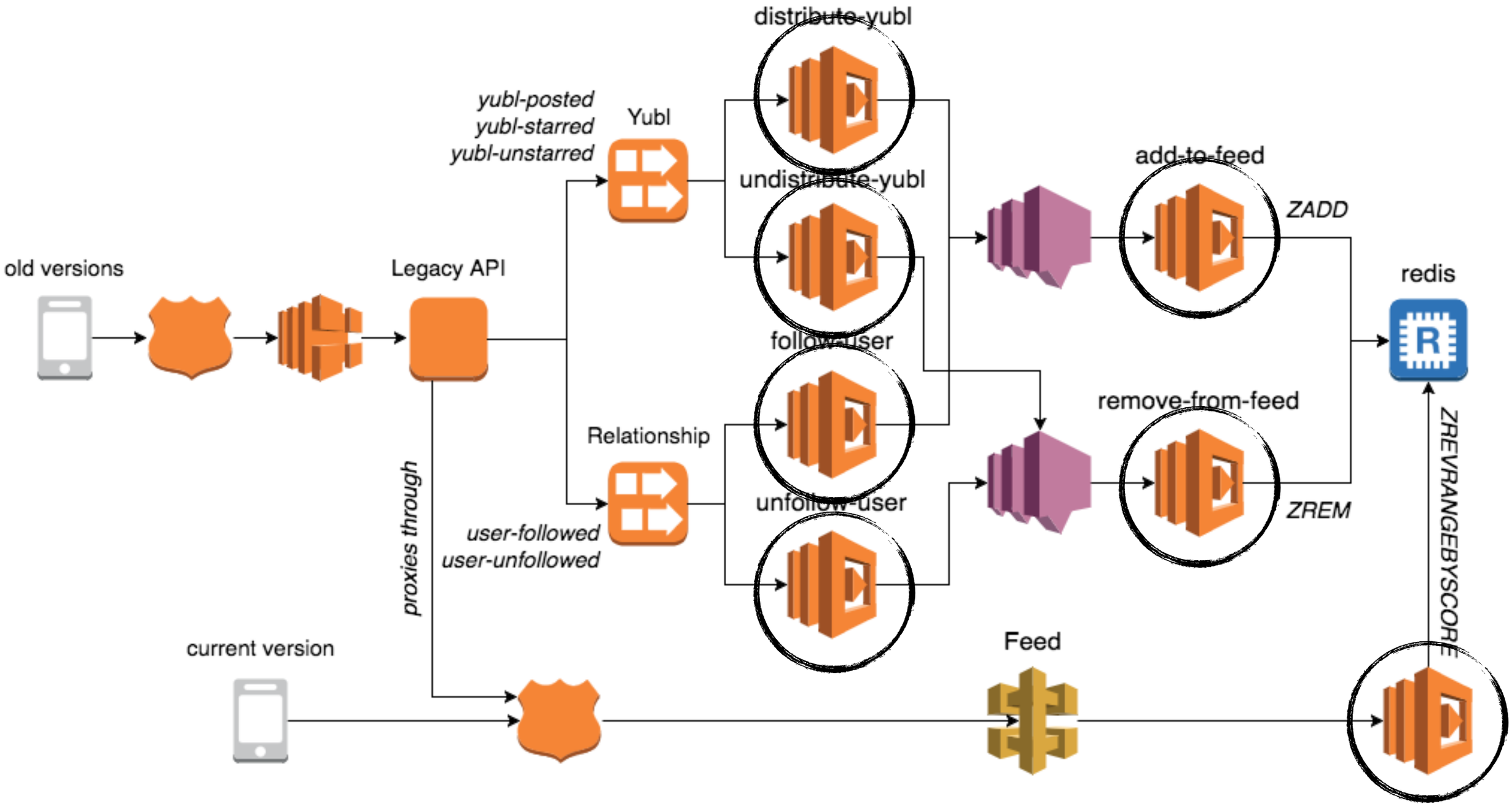
timeline feature



timeline feature



timeline feature



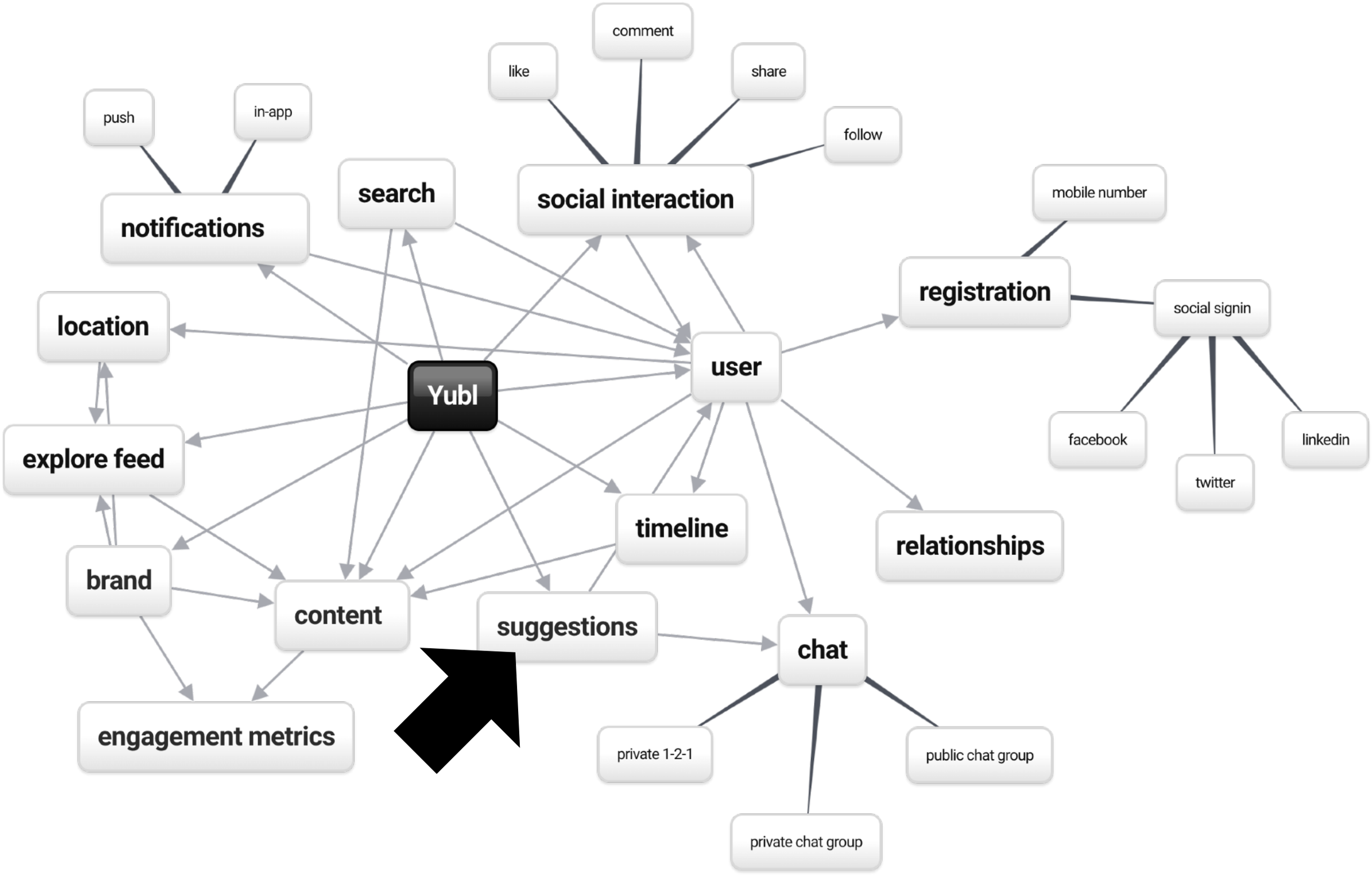
timeline feature

service: timeline-api

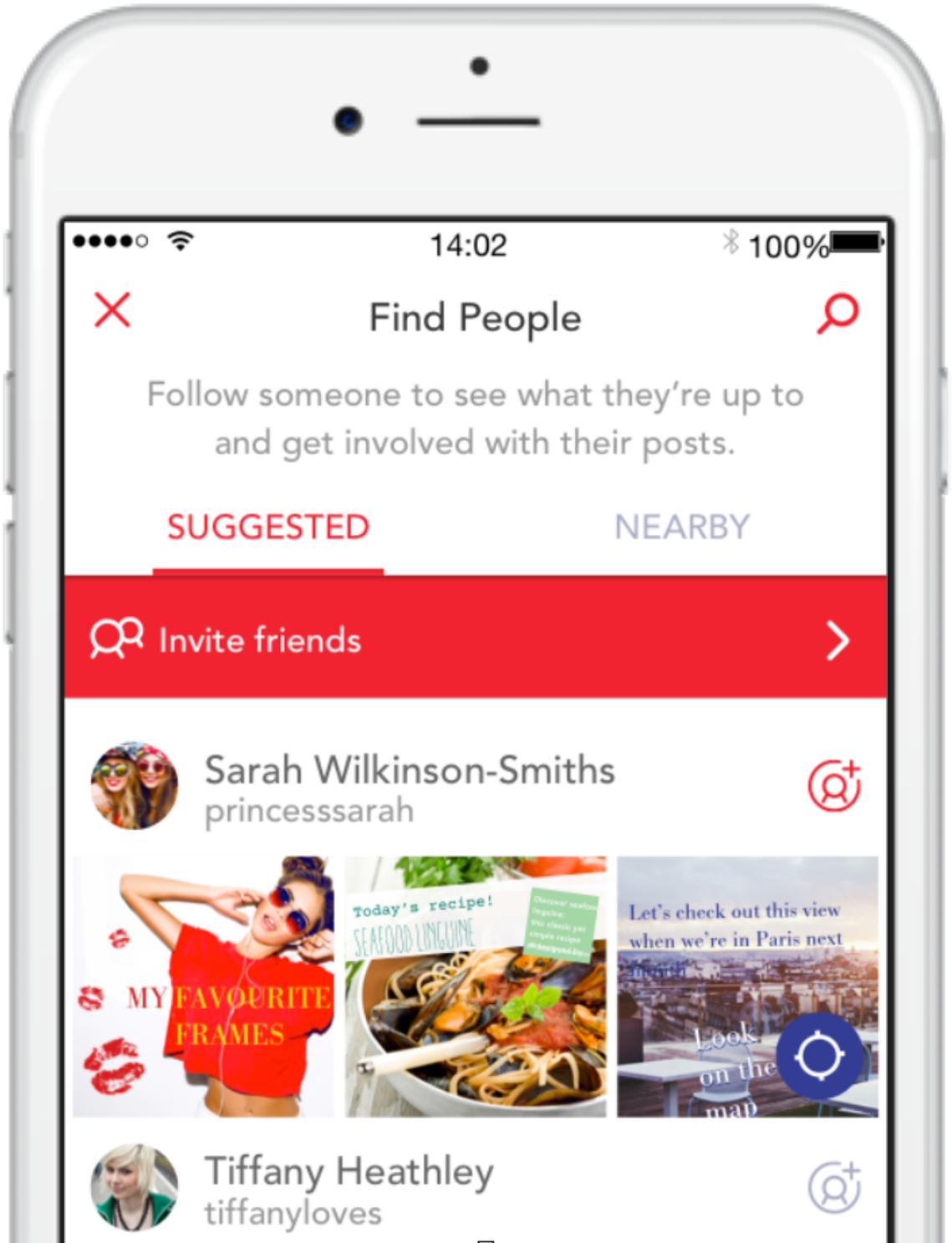
```
provider:
  name: aws
  runtime: nodejs6.10
  stage: dev
  region: us-east-1

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

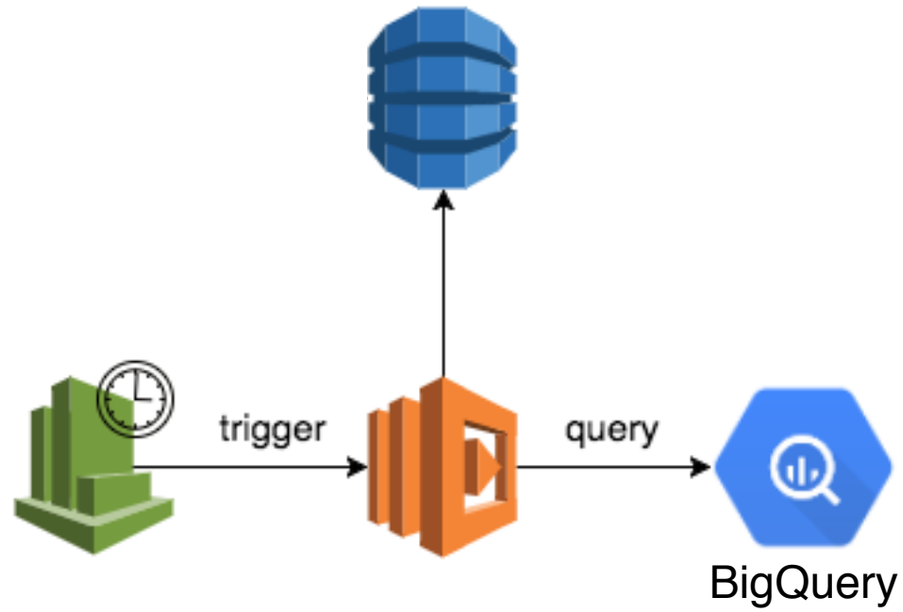
suggestions feature



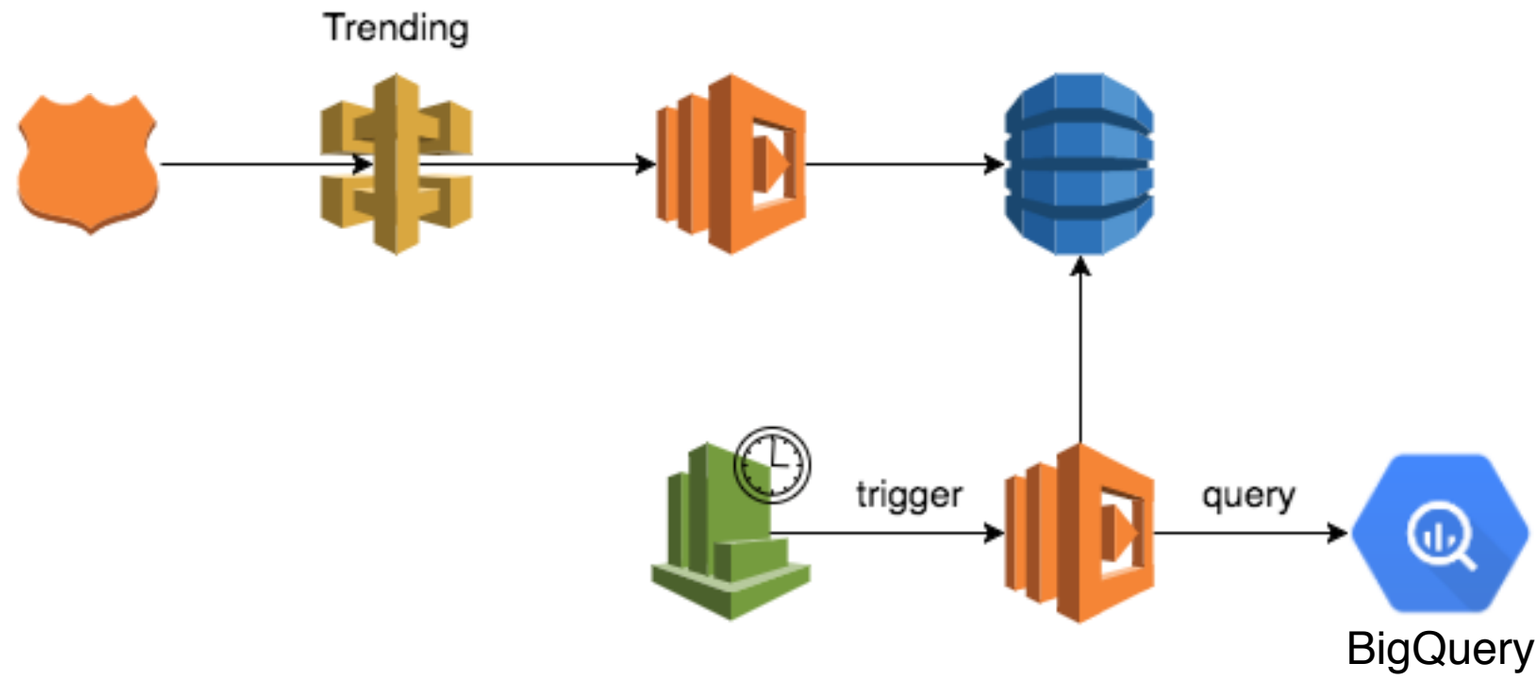
suggestions feature



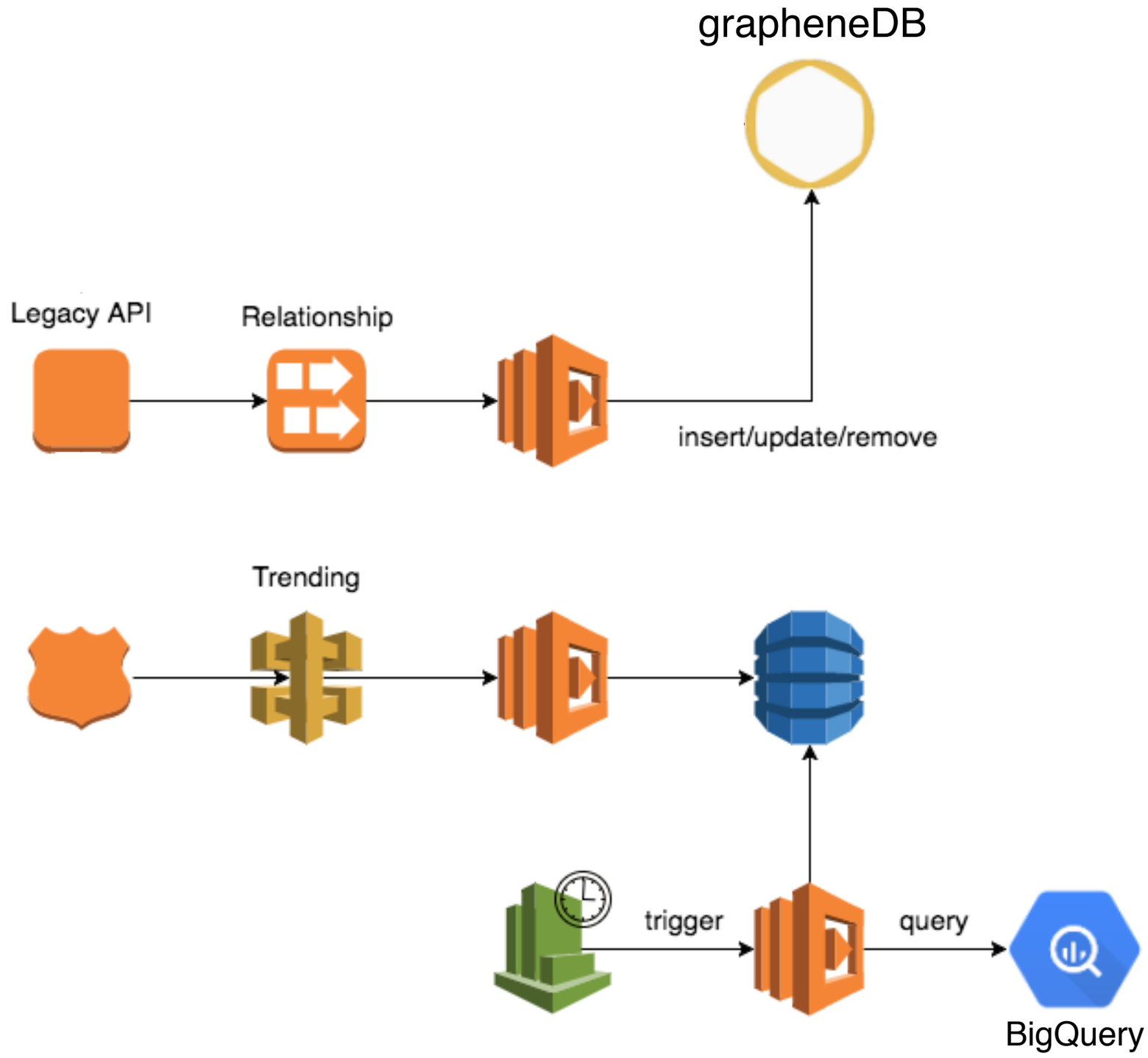
suggestions feature



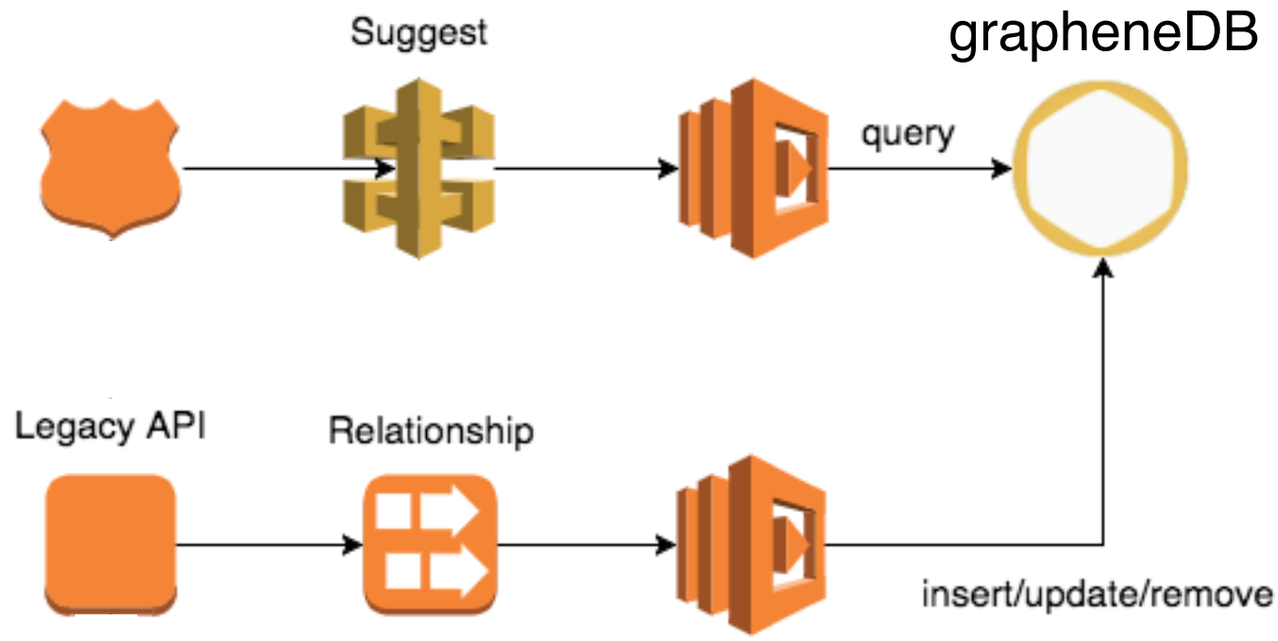
suggestions feature

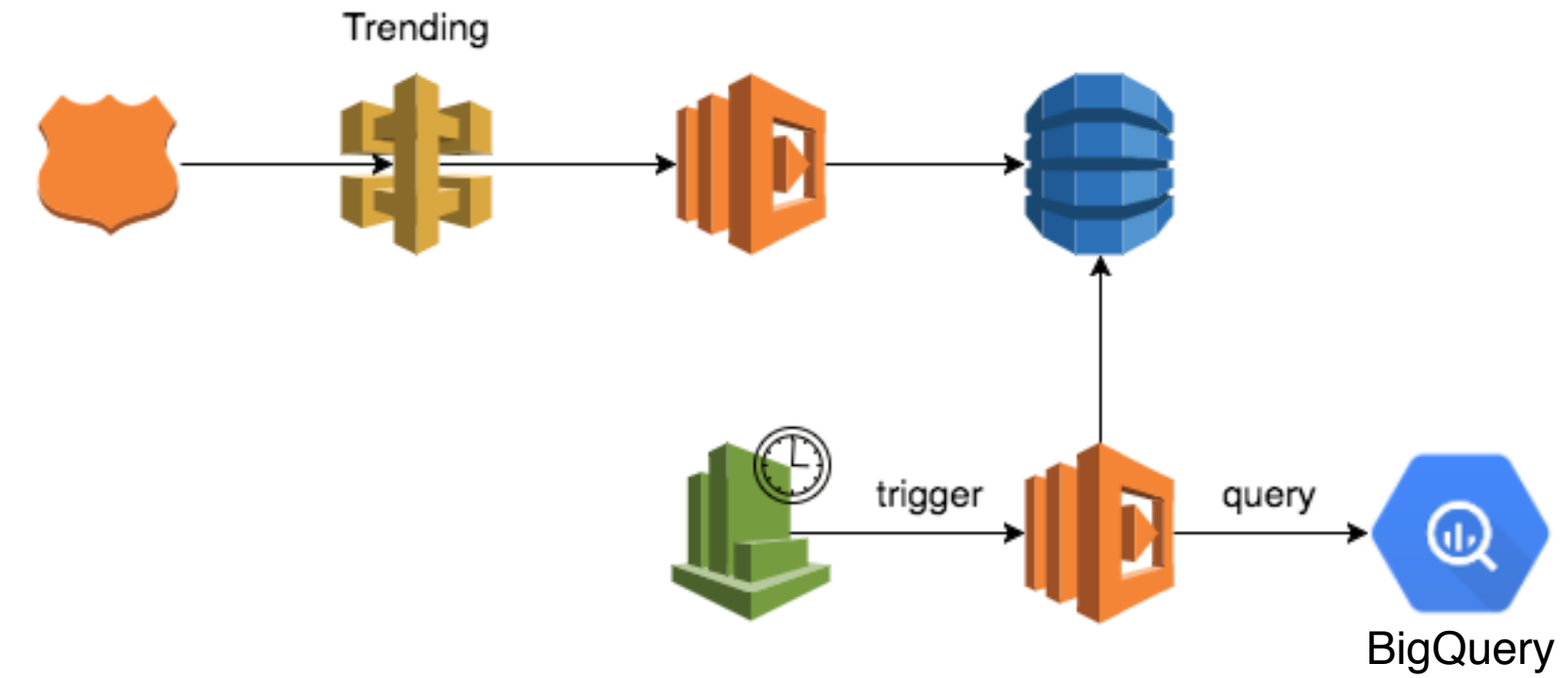


suggestions feature

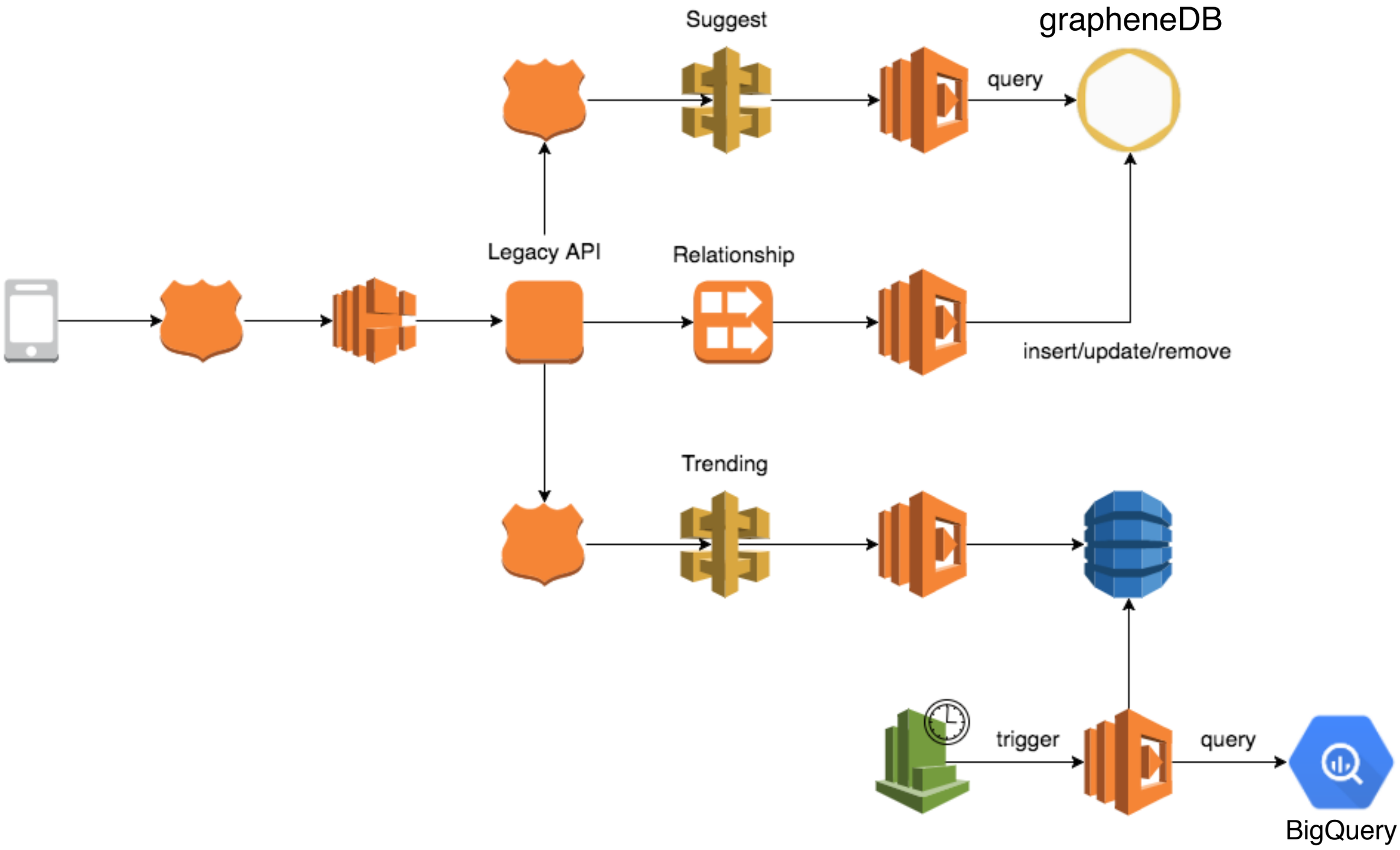


suggestions feature

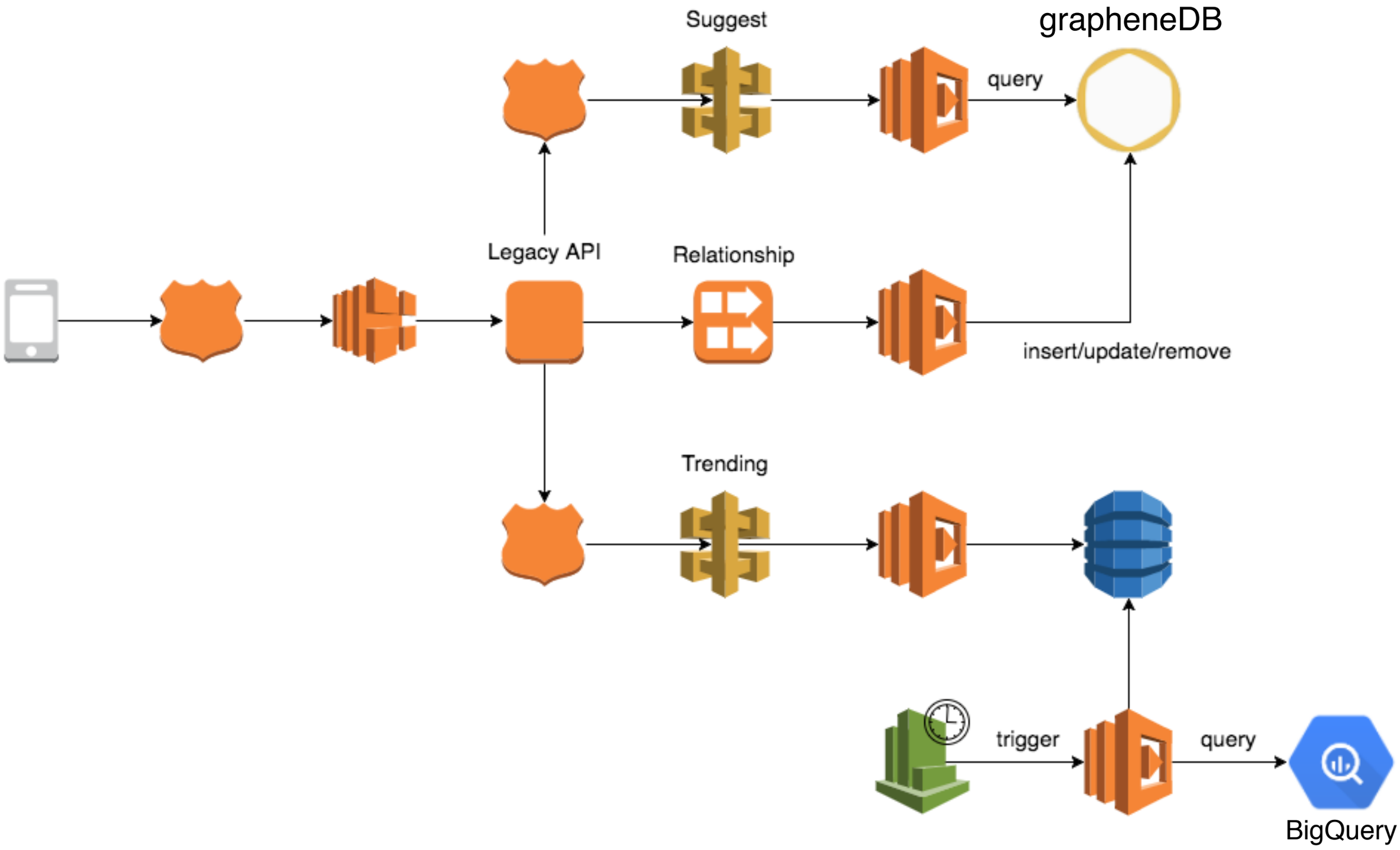




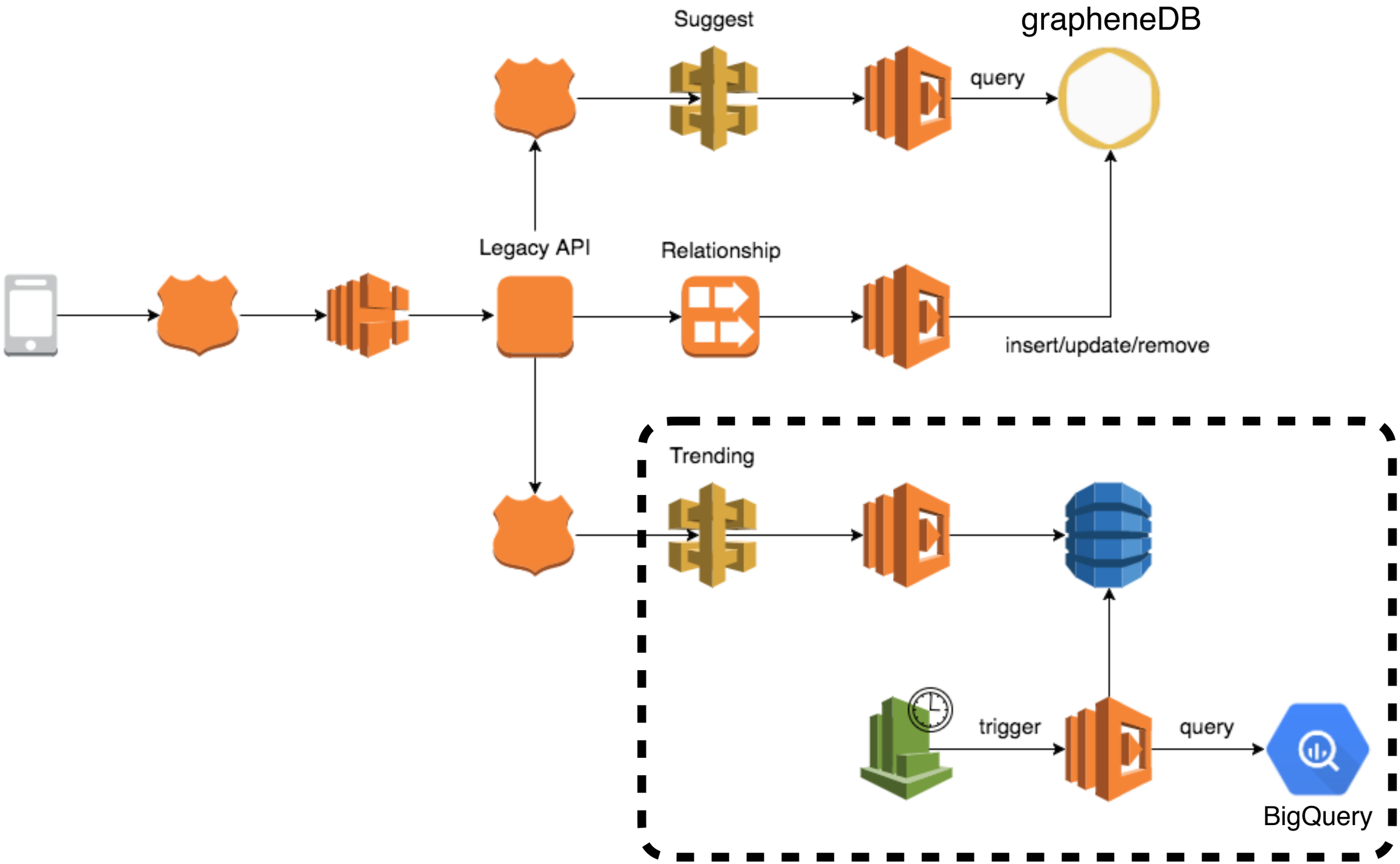
suggestions feature



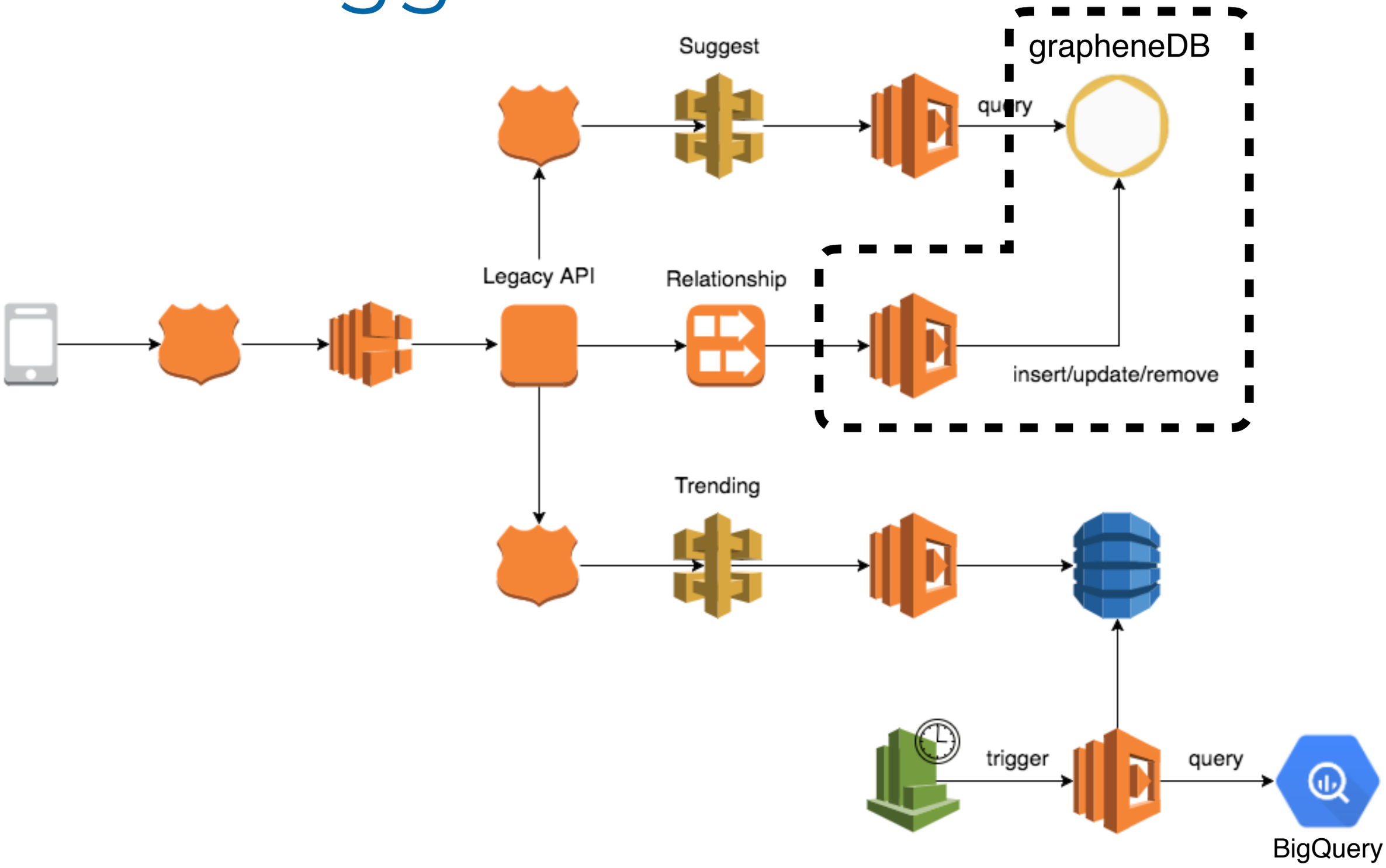
suggestions feature



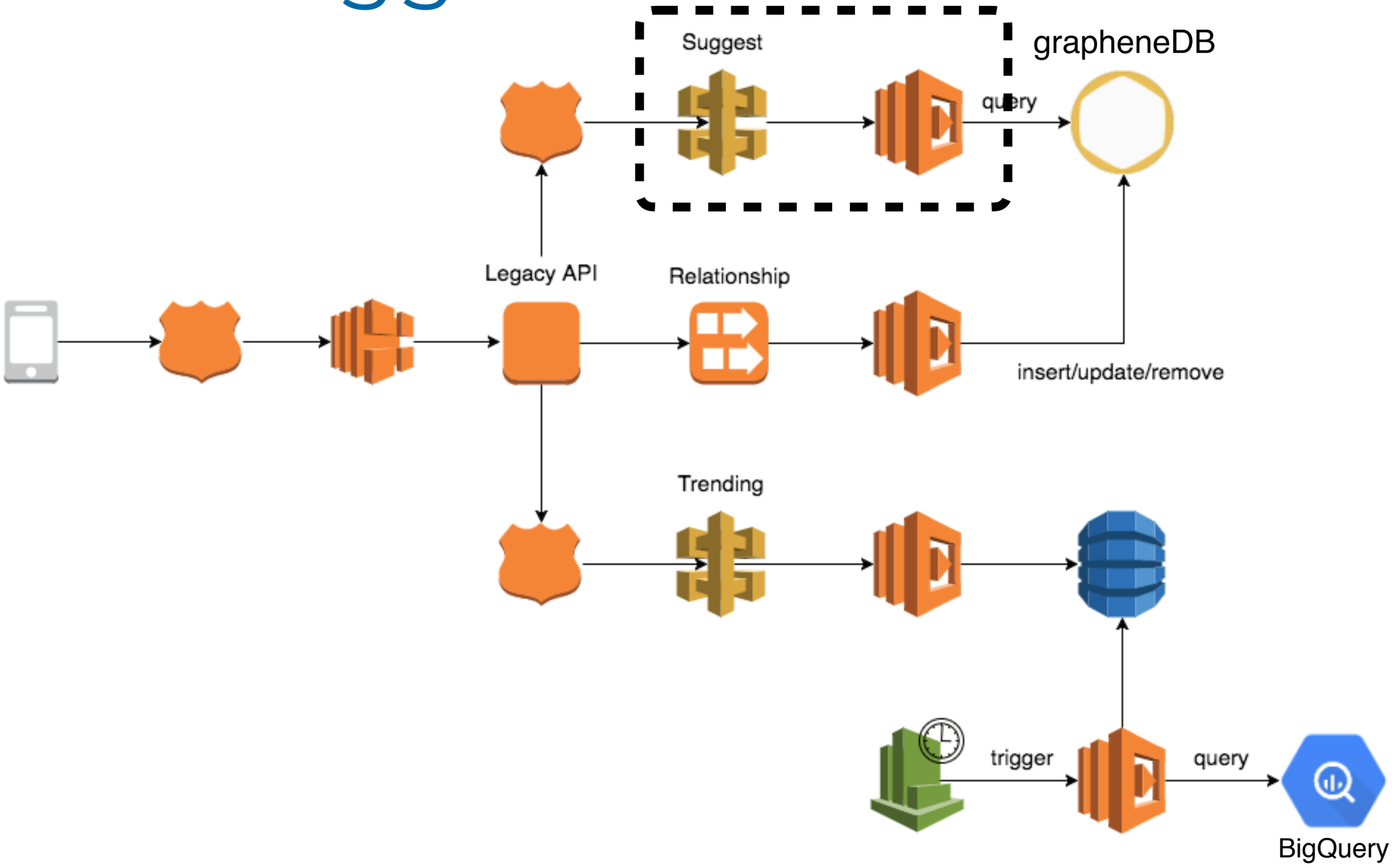
suggestions feature



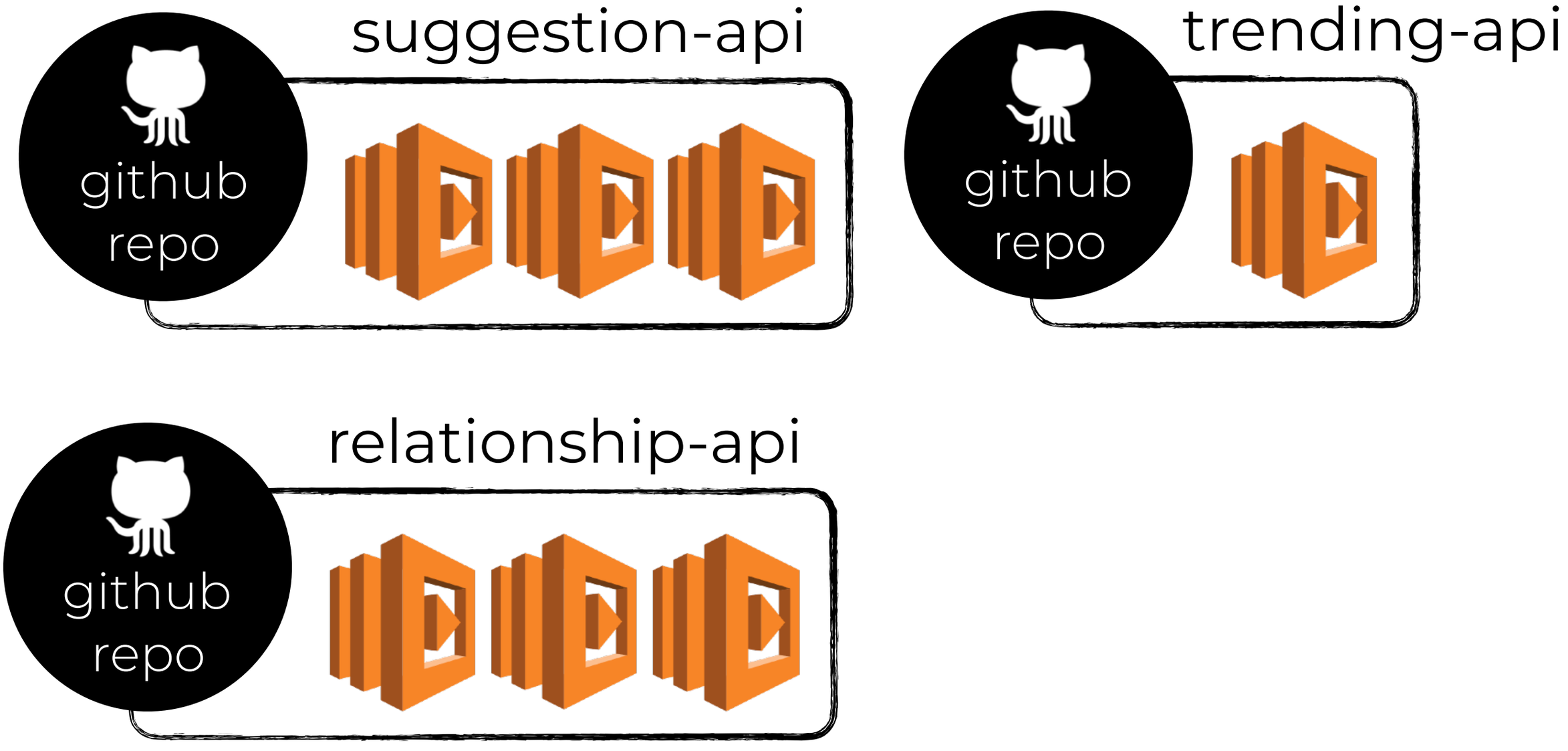
suggestions feature



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

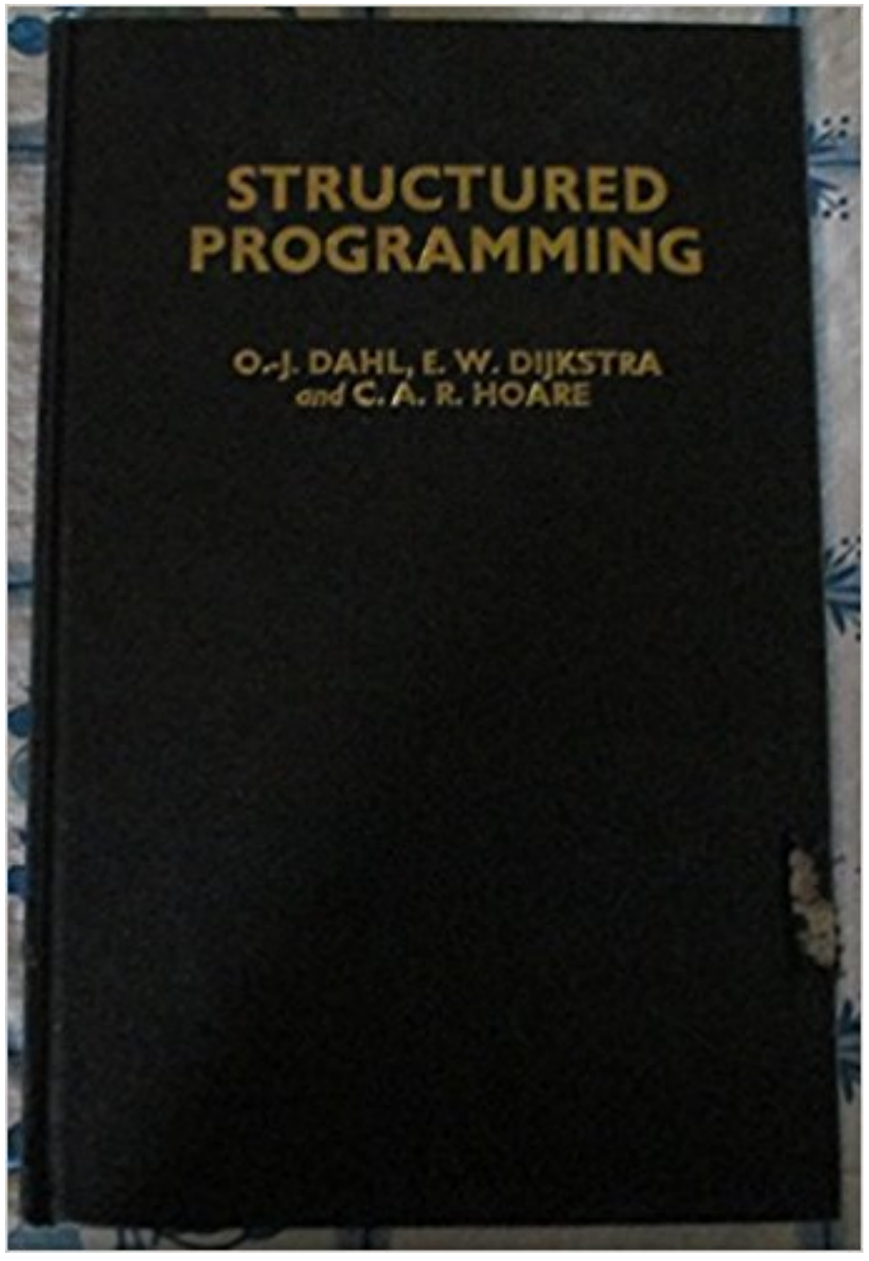
coupling

"the degree of **interdependencies** between software modules"

cohesion

"how **related** the functions within a single module are"

high cohesion, low coupling



Structured Programming
Academic Press
ISBN 0-12-200550-3
1972

high cohesion, low coupling

Edsger K. Dijkstra

Dijkstra's algorithm
ALGOL 60
semaphore
concurrent programming
call stack
shunting-yard algorithm
separation of concerns
principles of distributed computing

Ole-Johan Dahl

Simula
OOP

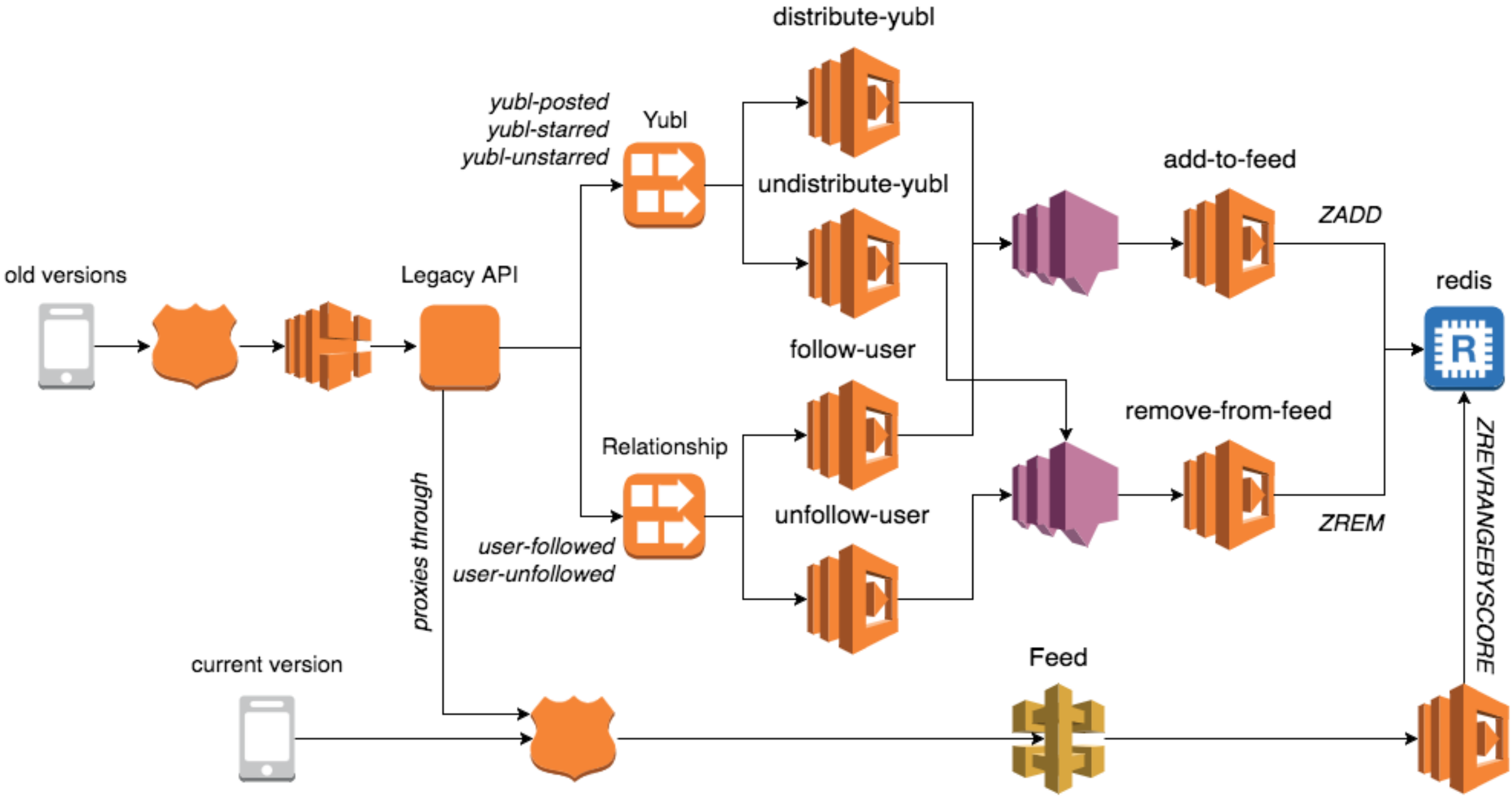
Tony Hoare

Quicksort
Quickselect
null reference
communicating sequential processes (CSP)

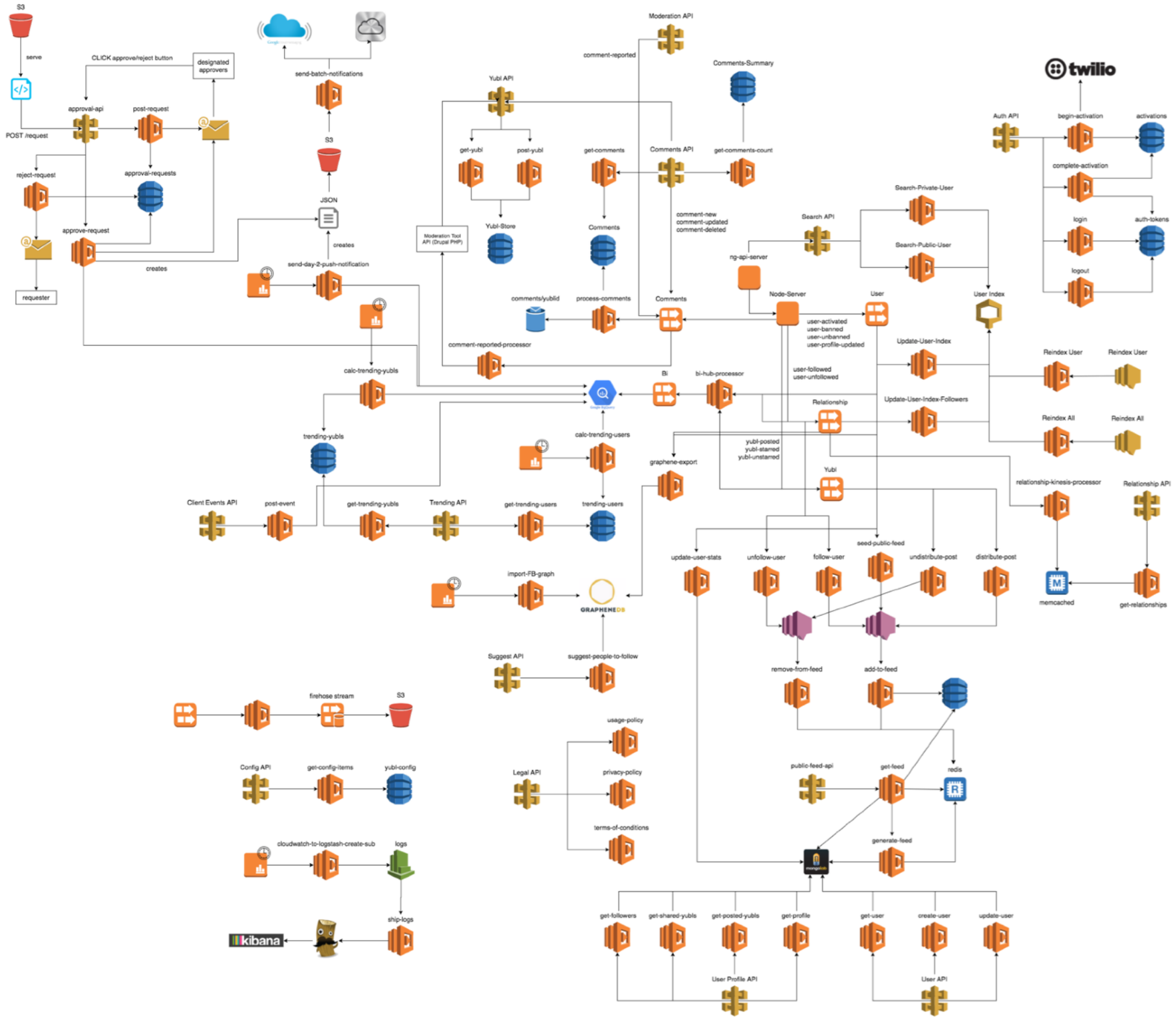
sharing code

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

sharing code



sharing code



sharing code

shared library

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

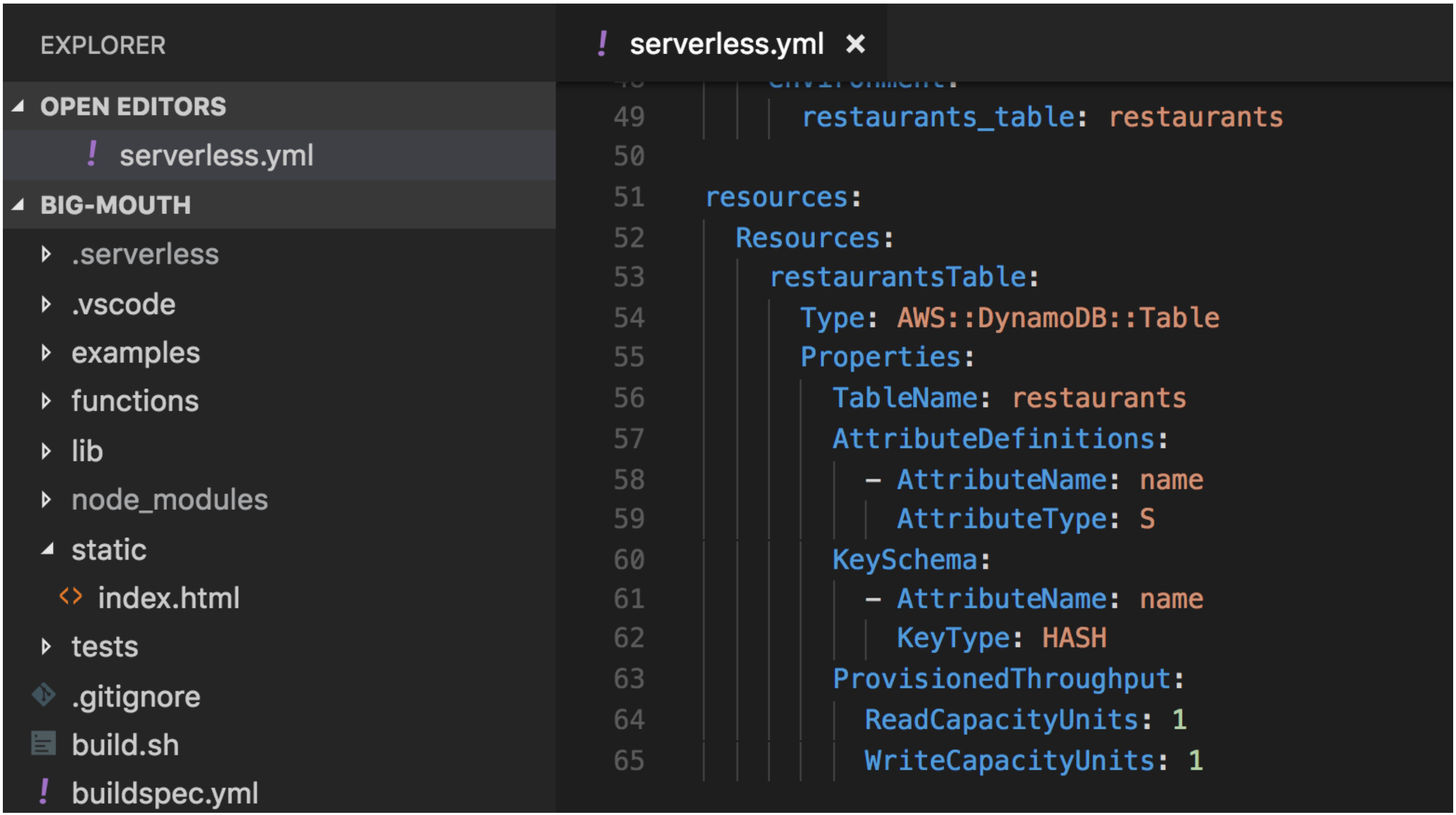
shared library vs service

	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



serverless.yml

sls remove can delete user data

serverless.yml

lock down IAM permissions

serverless.yml

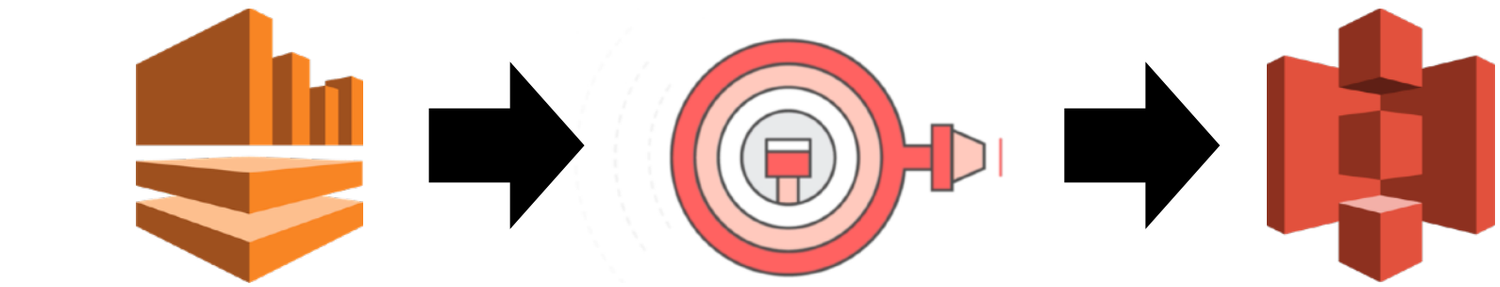
enable DynamoDB backup

serverless.yml

use CloudFormation DeletionPolicy

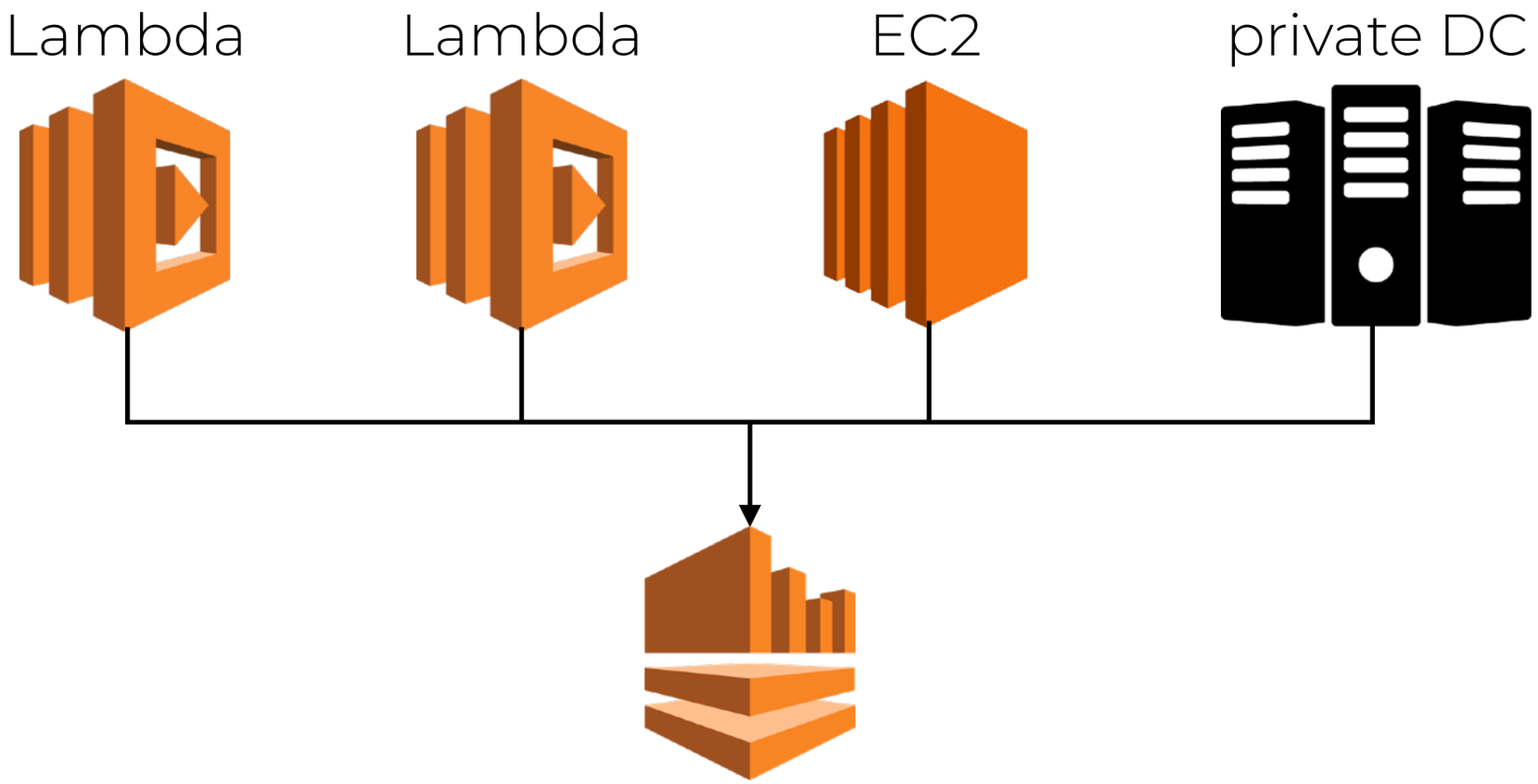
serverless.yml

back up Kinesis events with Kinesis Firehose and S3

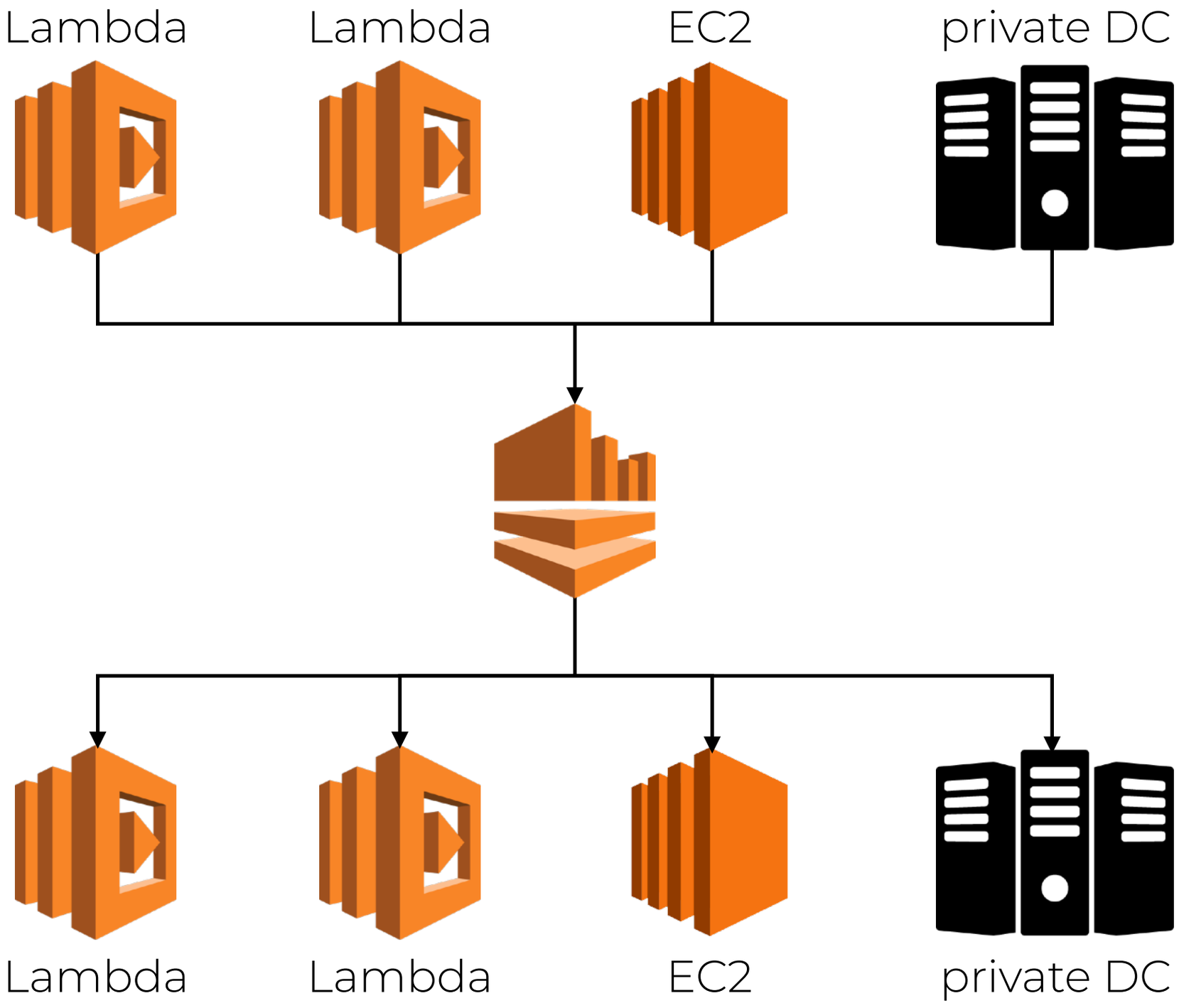


Kinesis Streams Kinesis Firehose S3

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