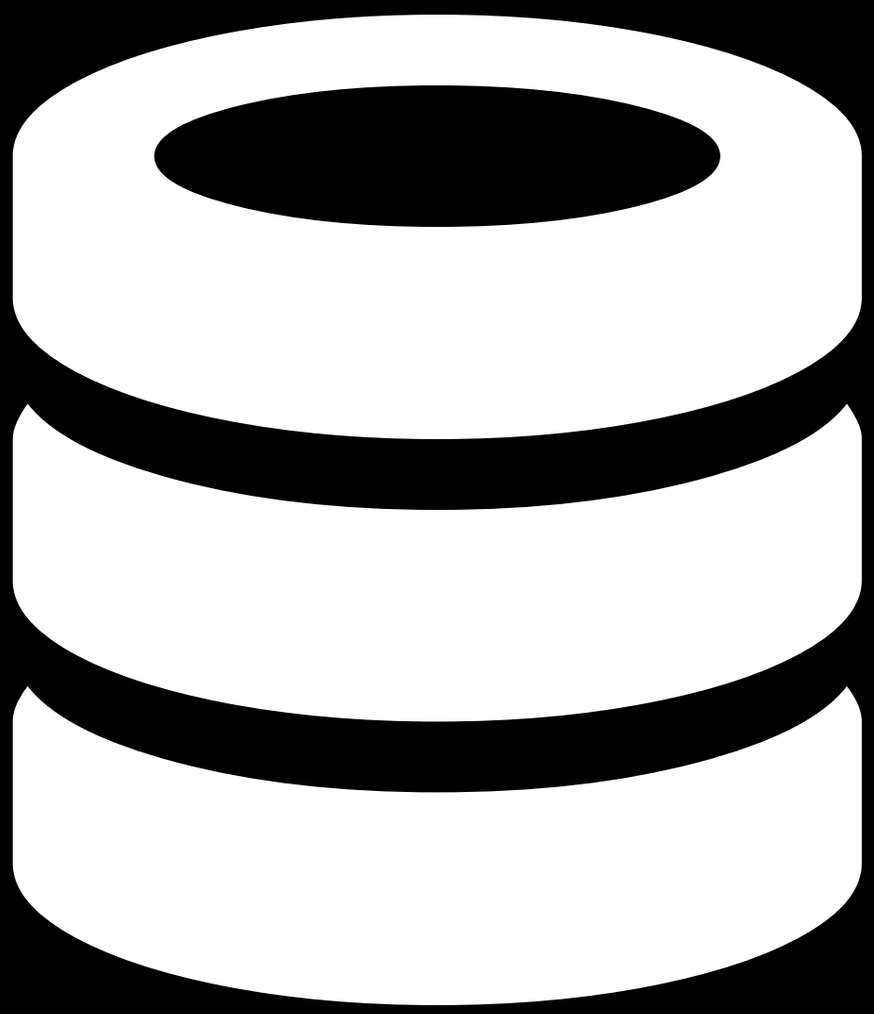
The GHTorrent Dataset and Tool Suite

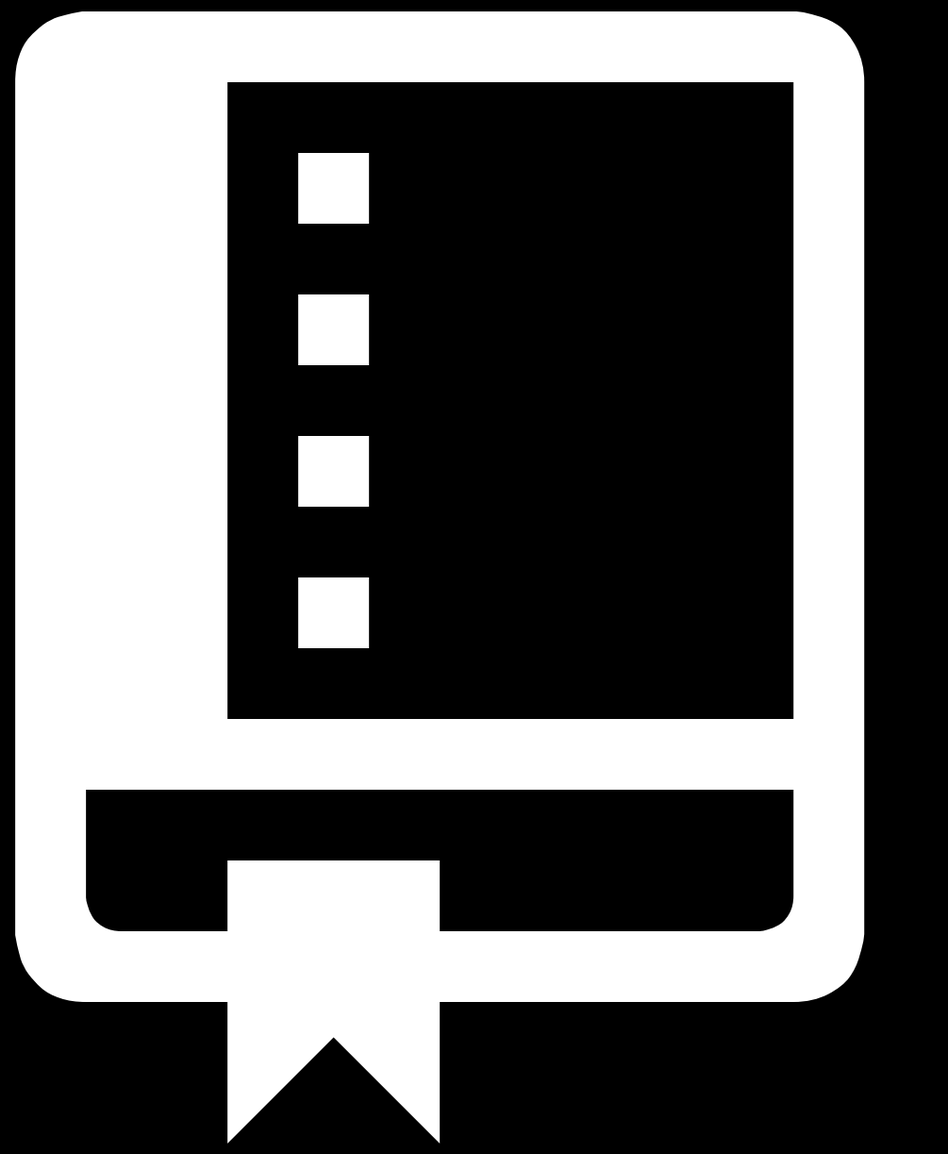
Georgios Gousios
Software Engineering Research Group
TU Delft

All data from Github

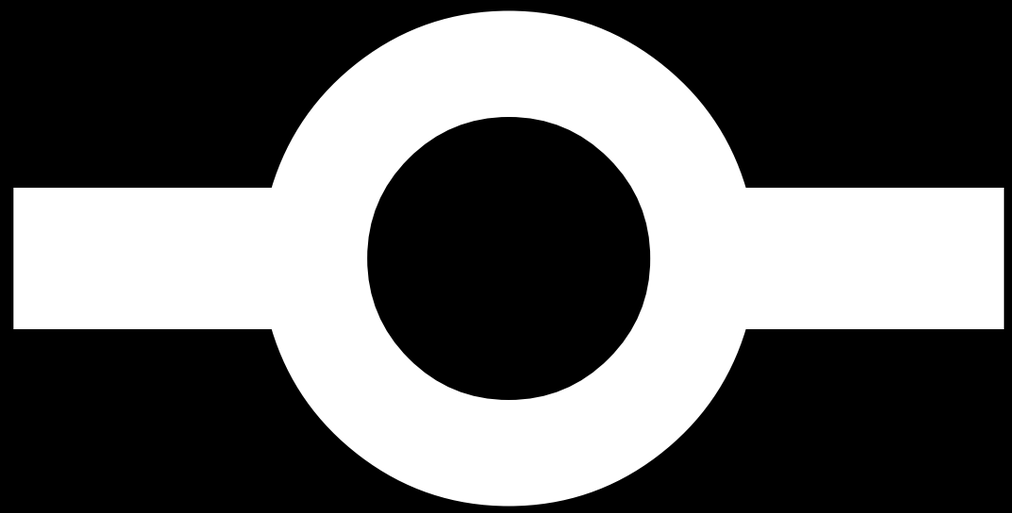


Ready to be queried

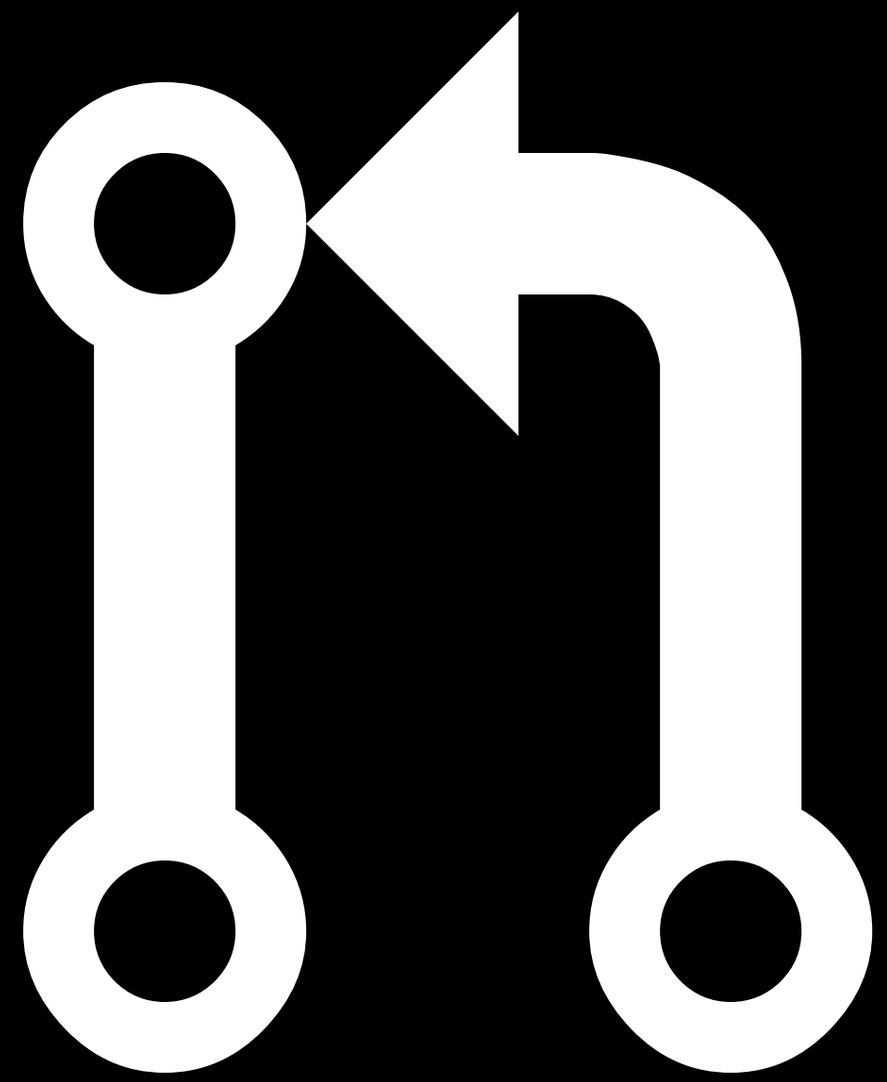
ghtorrent.org

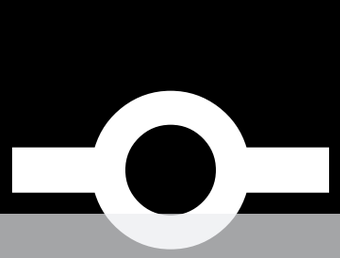


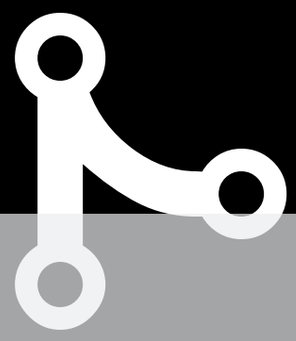
Repositories



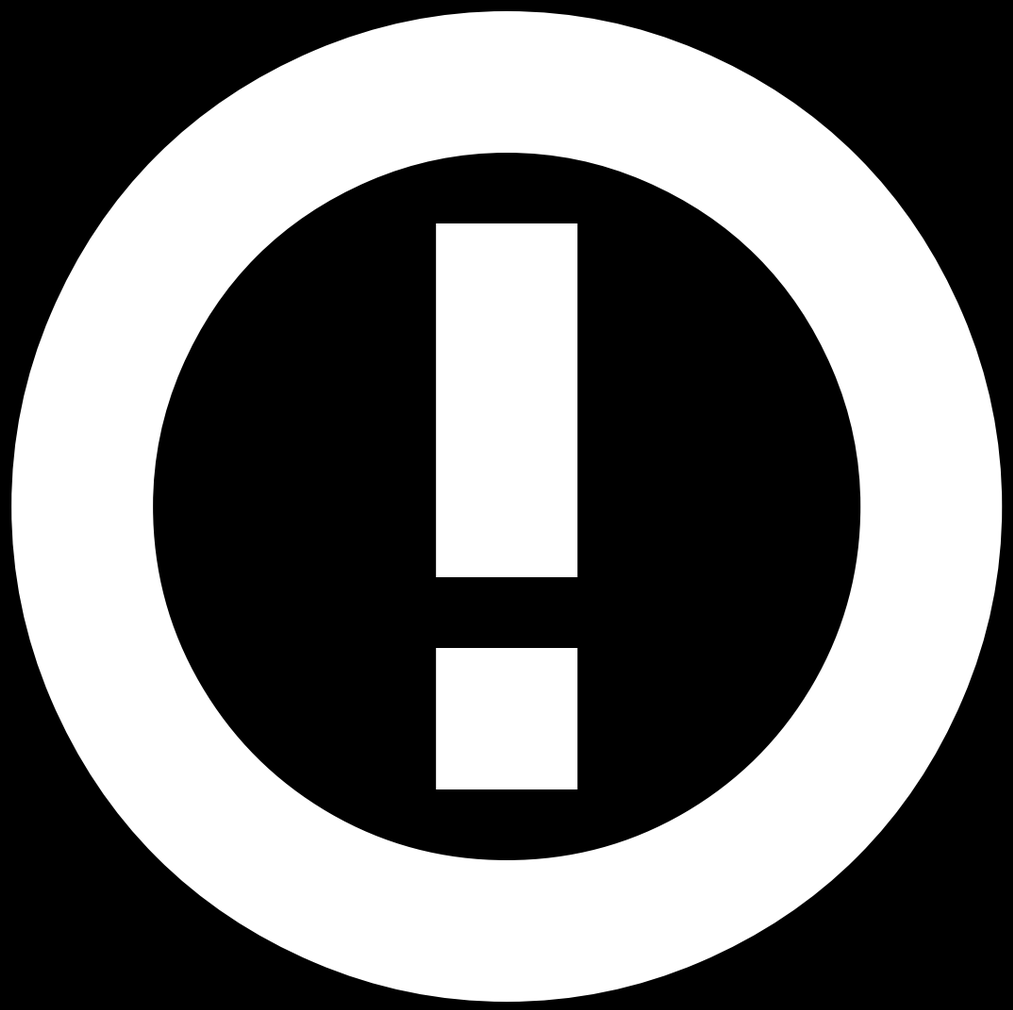
Commits

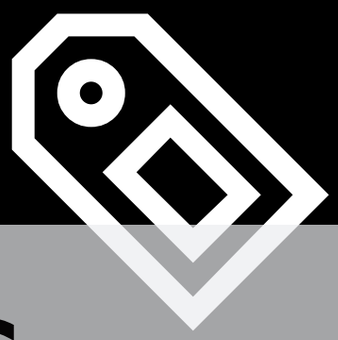


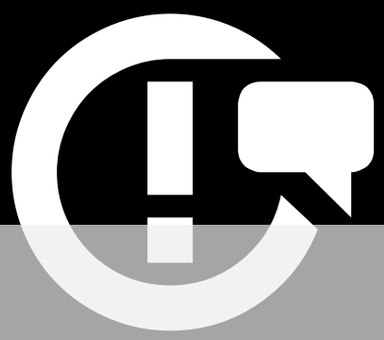


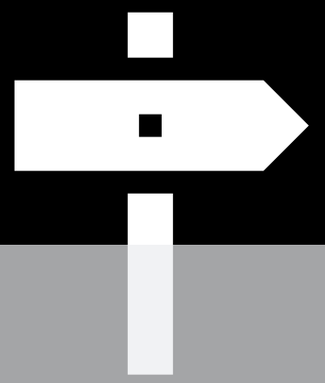


Pull requests

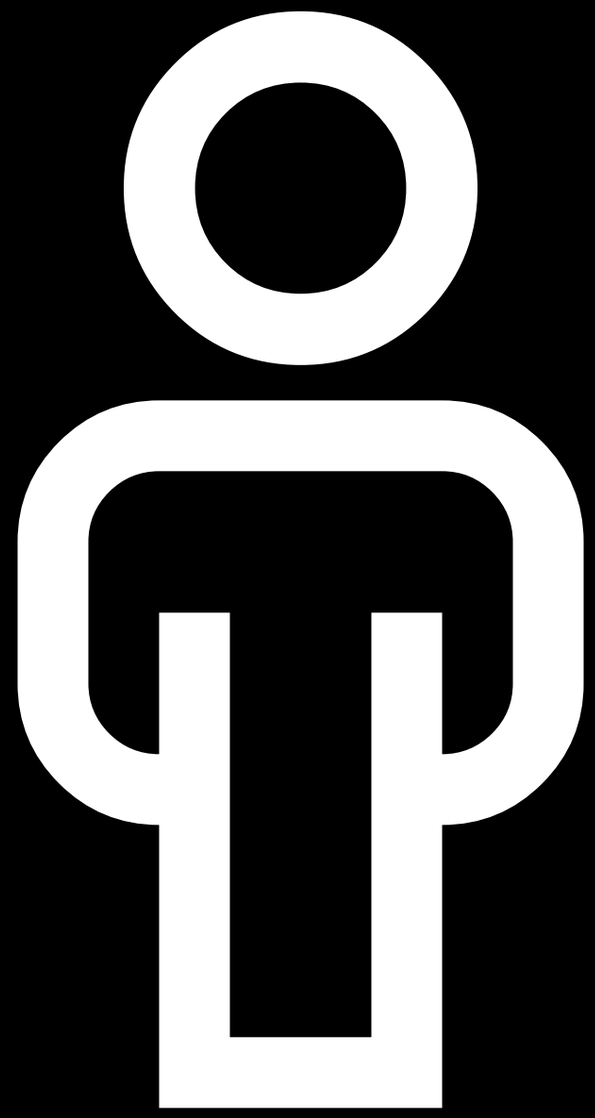


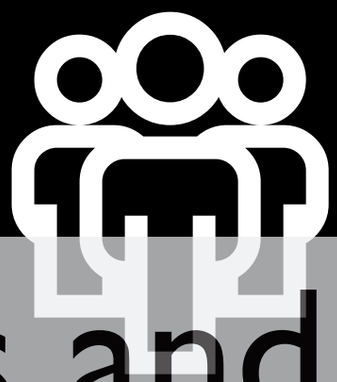


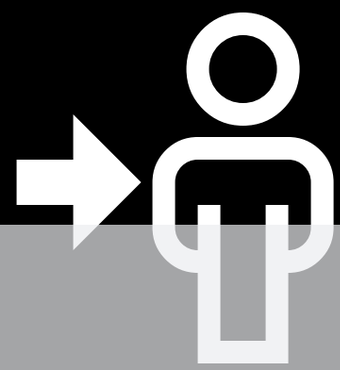




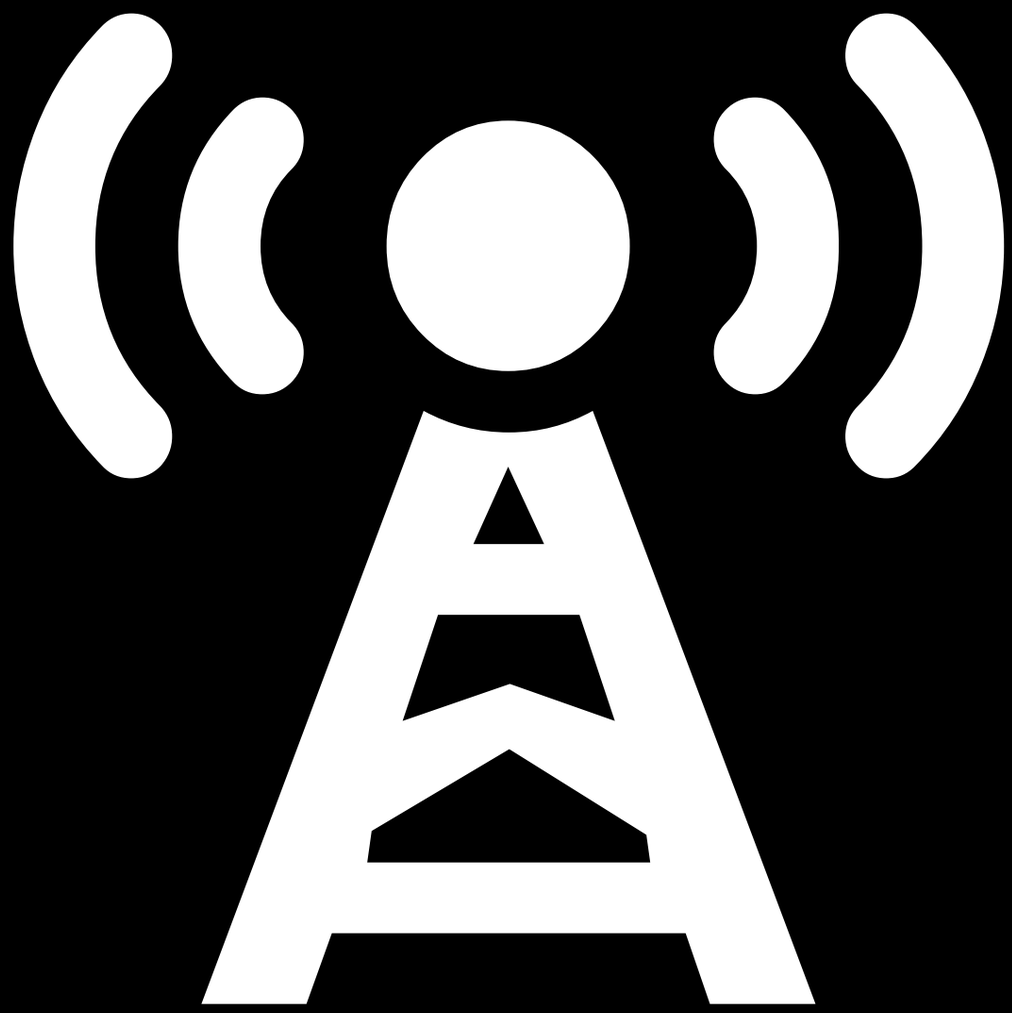
Issues



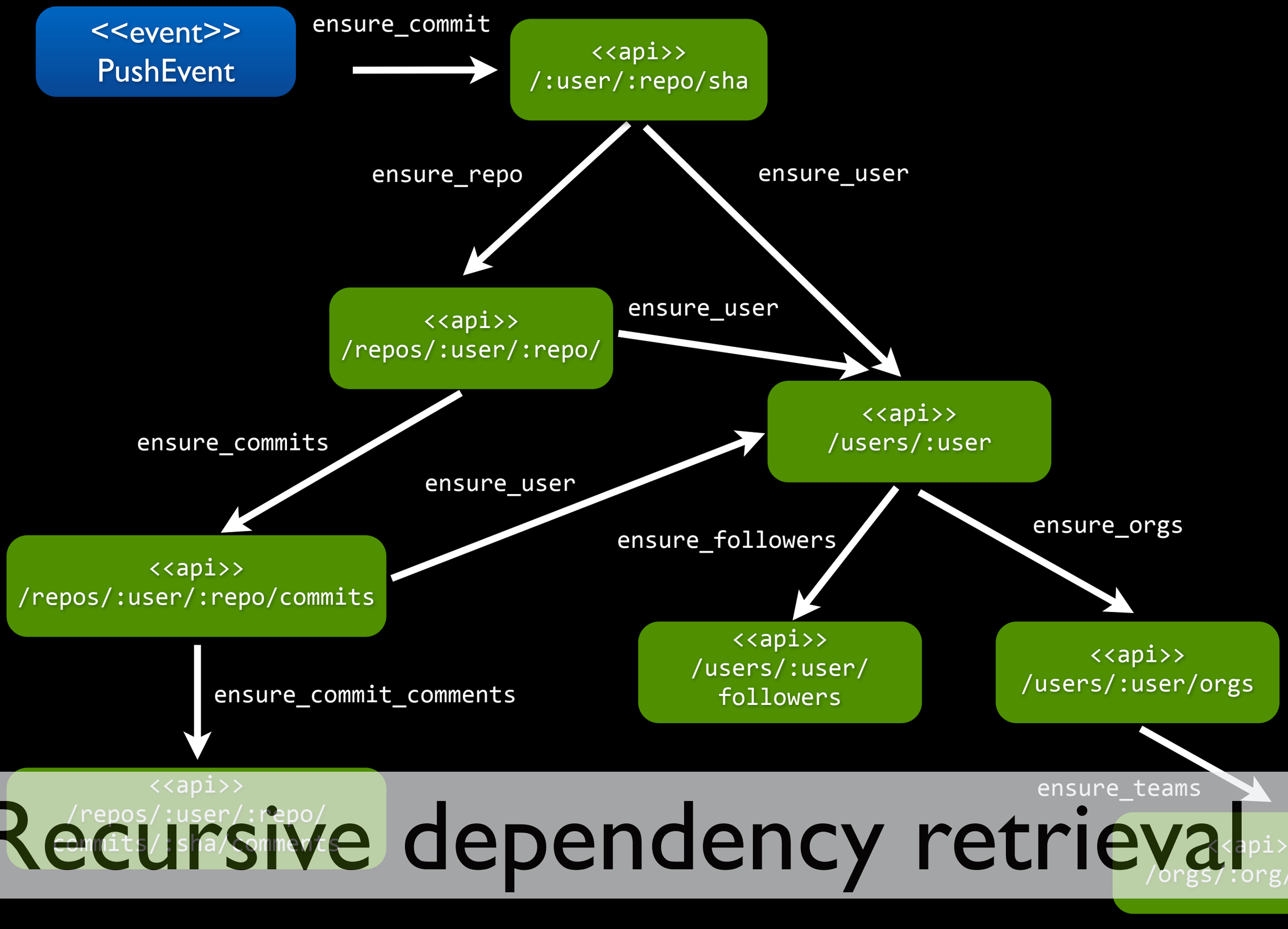




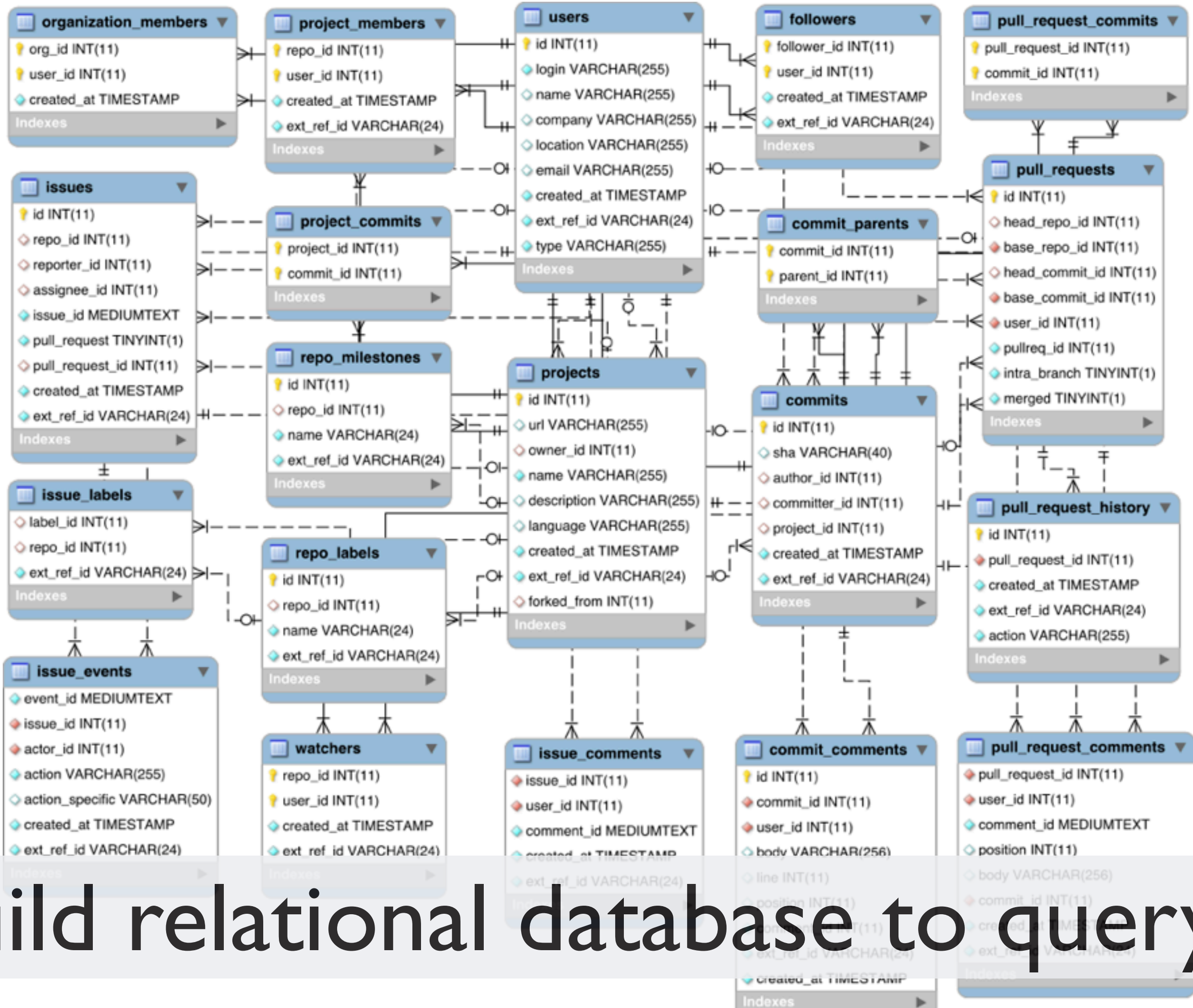
Users and Organizations



Mirror event stream



Recursive dependency retrieval



Build relational database to query

users

repositories

organizations

issues

/users/:user

/user/repos

/orgs/:org

/repos/:user/:repo/issues


```
{
  "type": "User",
  "public_gists": 0,
  "login": "gousiosg",
  "followers": 8,
  "name": "Georgios Gousios"
  "public_repos": 4,
  "created_at": ...,
  "id": 386172,
  "following": 4,
}
```


mongoDB

{name: “mongo”, type:“DB”}

CoSQL database as cache

Downloads

MySQL database dumps

You can also query MySQL. It is always loaded with the latest dump.

- 2013-02-23

Available Torrents

List of available torrents (Last dump date: 2013-03-29)

Dump date	commit comments	commits	events	followers	forks	issue comments	issue events	issues	org members	pull request comments	pull requests	repo collaborators	repos	users	watchers
2012-03-30		810 MB	1221 MB												
2012-05-30		17962 MB	1945 MB												
2012-07-30	1 MB	34533 MB	1951 MB	35 MB								18 MB	29 MB		
2012-09-29	2 MB	10351 MB	320 MB	15 MB	40 MB				1 MB		30 MB	11 MB	34 MB	16 MB	194 MB
2012-11-29	3 MB	11921 MB	3702 MB	13 MB	34 MB	155 MB	84 MB	360 MB		12 MB	35 MB	14 MB	60 MB	7 MB	132 MB
2013-01-29	8 MB	32483 MB	3484 MB	59 MB	211 MB	385 MB	172 MB	335 MB		170 MB	64 MB	70 MB	617 MB	40 MB	256 MB
2013-03-29	22 MB	172471 MB	17185 MB	148 MB	451 MB	797 MB	311 MB	926 MB	2 MB	536 MB	943 MB	108 MB	1985 MB	125 MB	657 MB

Periodic dumps of DBs online

DBLite

Database Explorer

localhost

- localhost
  - information_schema
  - ghtorrent
    - Tables
      - commit_comments
      - commit_parents
      - commits
      - followers
      - issue_comments
      - issue_events
      - issue_labels
      - issues
      - organization_members
      - project_commits
      - project_members
      - projects
      - pull_request_comments
      - pull_request_commits
      - pull_request_history
      - pull_requests
      - repo_labels
      - repo_milestones
      - schema_info
      - users
      - watchers
    - Views
    - Stored Procs
    - Functions
  - test

SQL*

```
1  select month(p.created_at) as month, year(p.created_at) as year, count(*) as contributing
2  from projects p
3  where exists (
4      select *
5      from pull_requests pr
6      where pr.head_repo_id = p.id
7  )
8  group by p.id, month, year
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
```

Conn: localhost | DB: ghtorrent | Table Structure | History | 1. Result

Export | Show SQL | Refresh | Auto Refresh

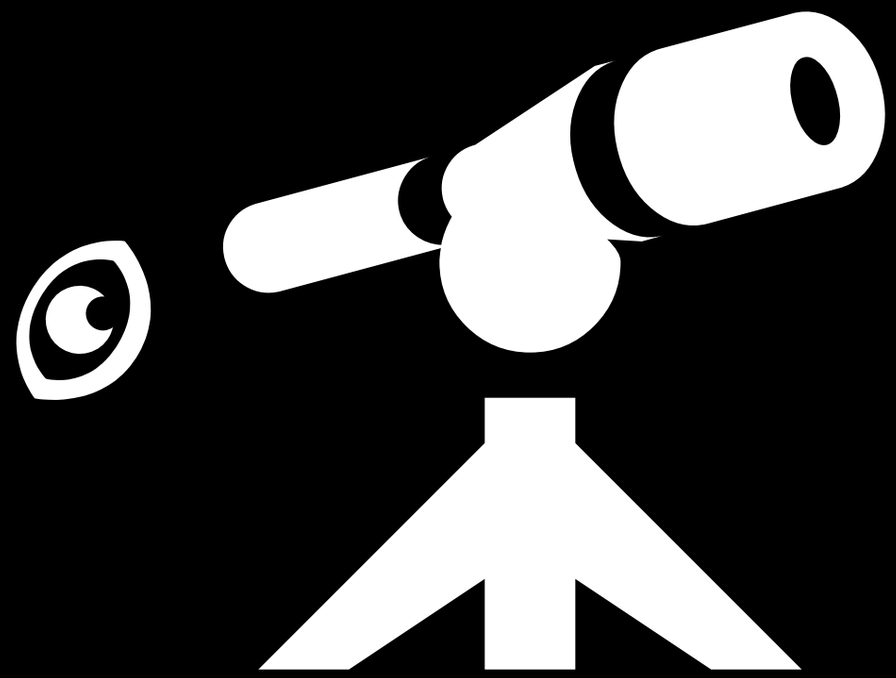
month	year	contributing
8	2011	1
6	2010	1
8	2011	1
8	2012	1
4	2011	1
7	2012	1
8	2012	1

Page 1 of 7487

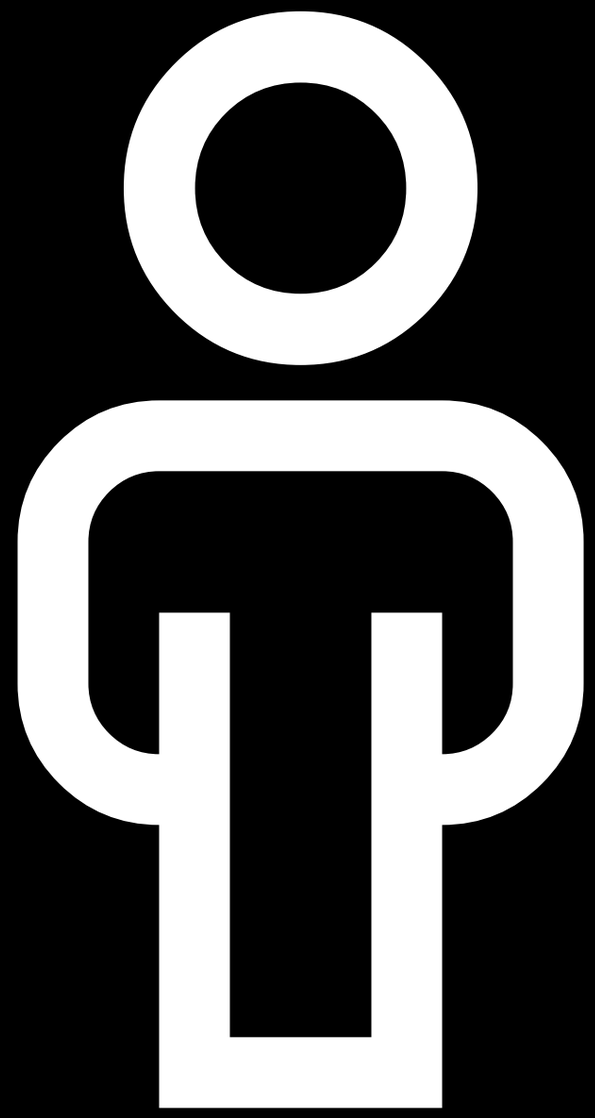
Query relational DB online

```
$ gem install sqlite3 ghtorrent
$ (edit config.yaml)
$ ght-retrieve-repo mojombo jekyll
```

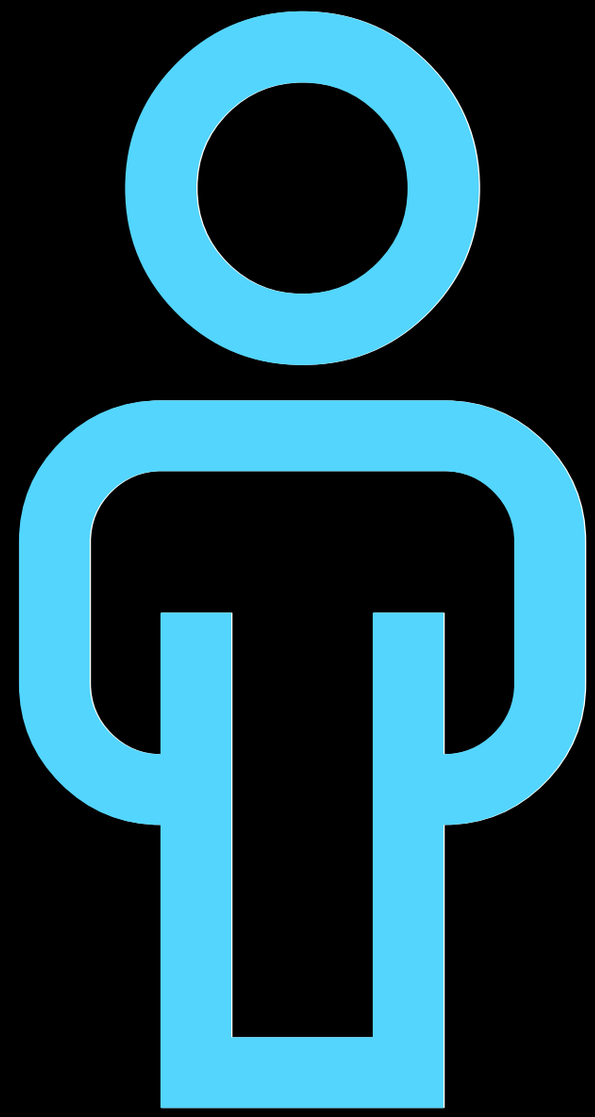
Roll your own tools



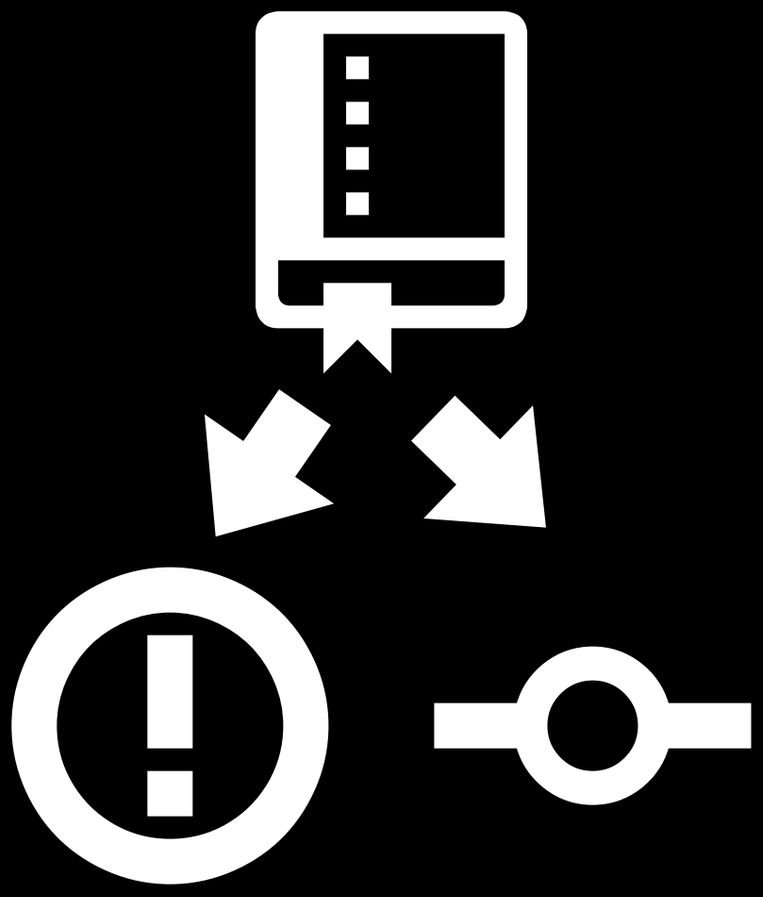
Research !

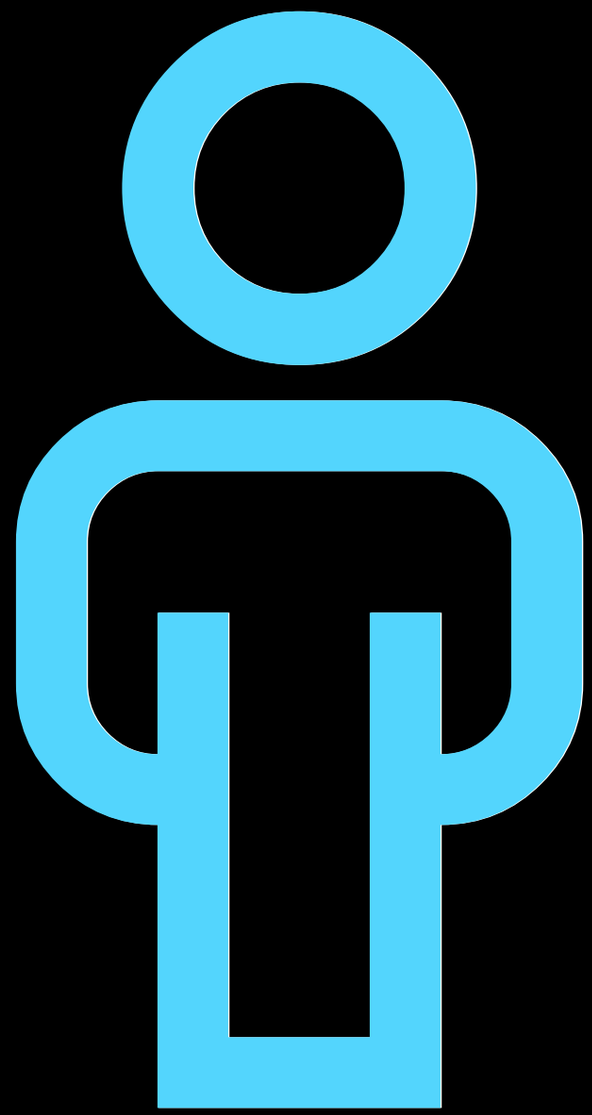


Single developer identities

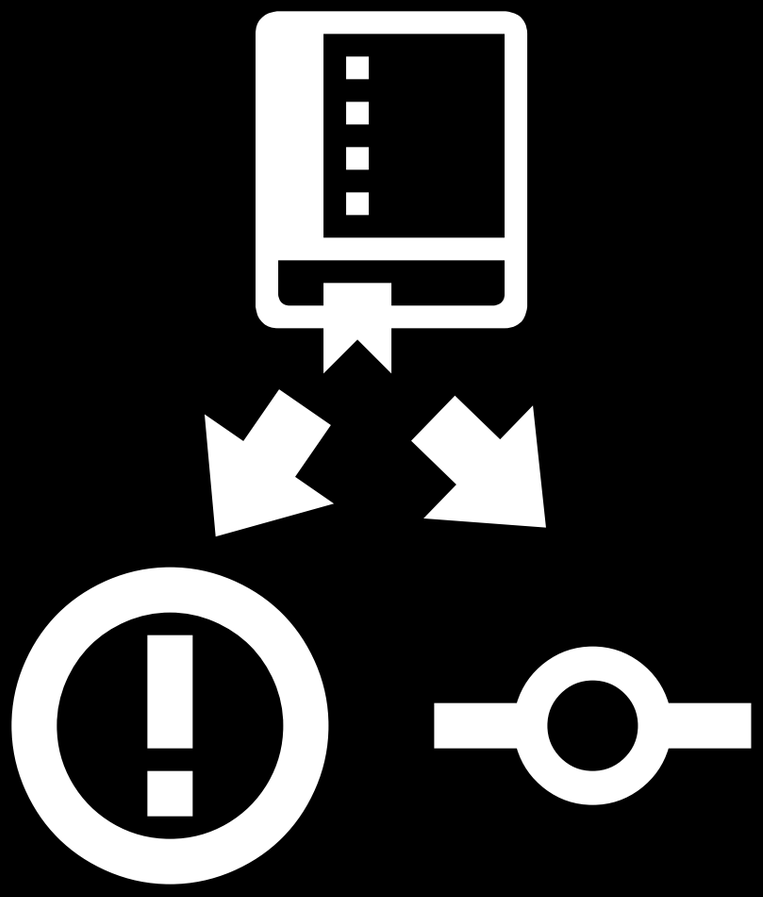


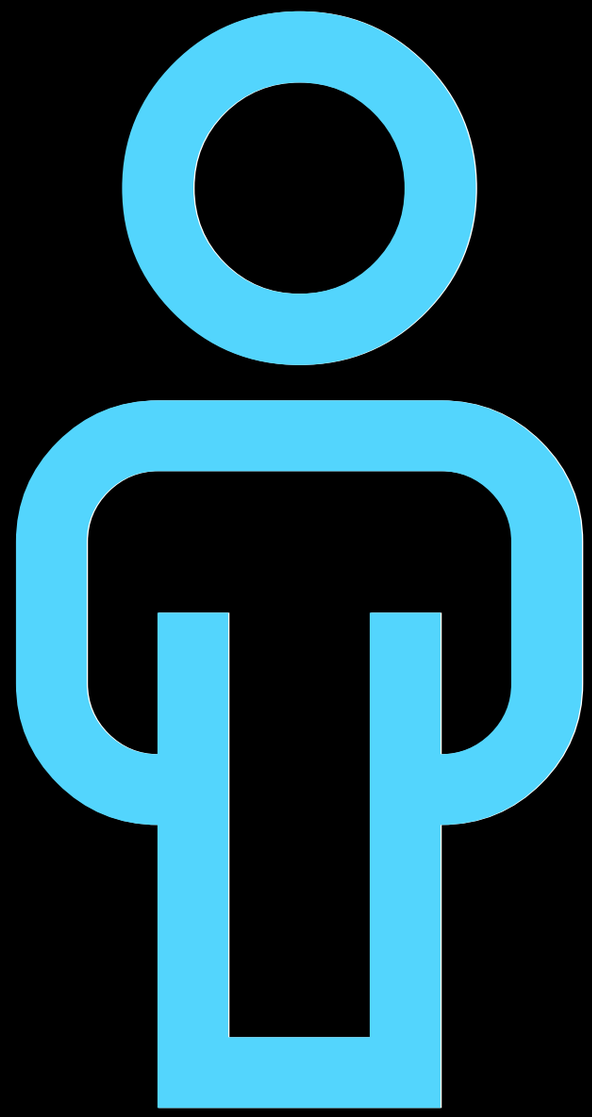
Single developer identities

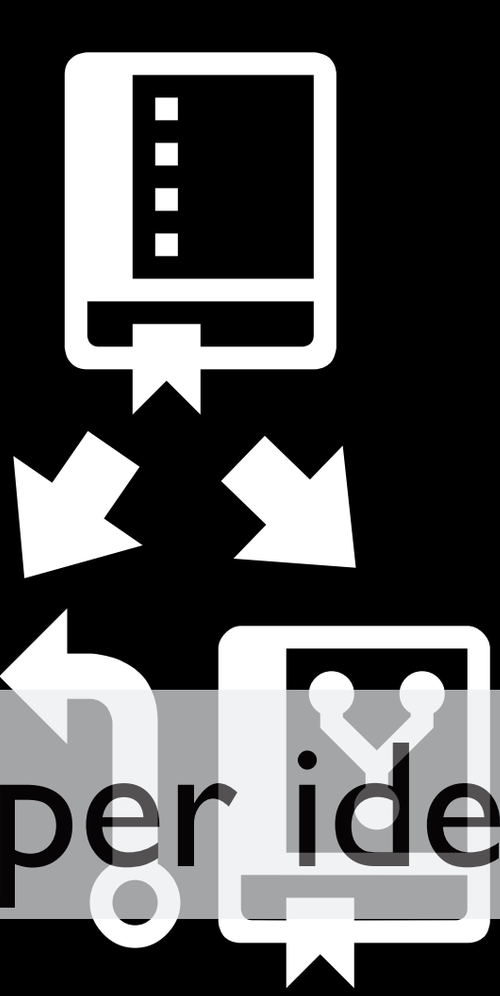




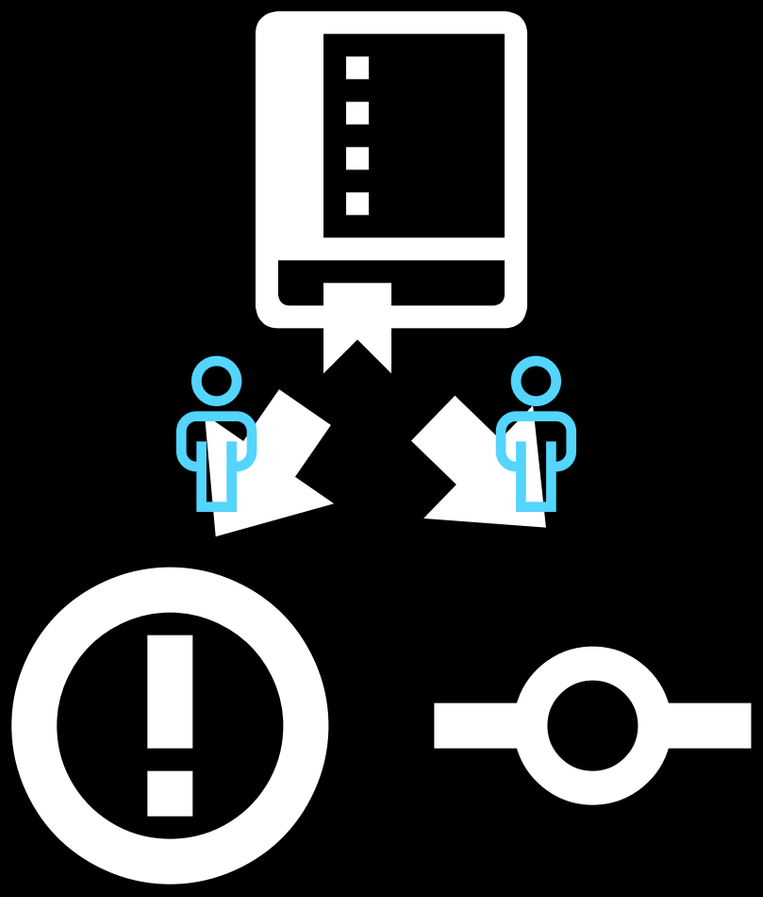
Single developer identities

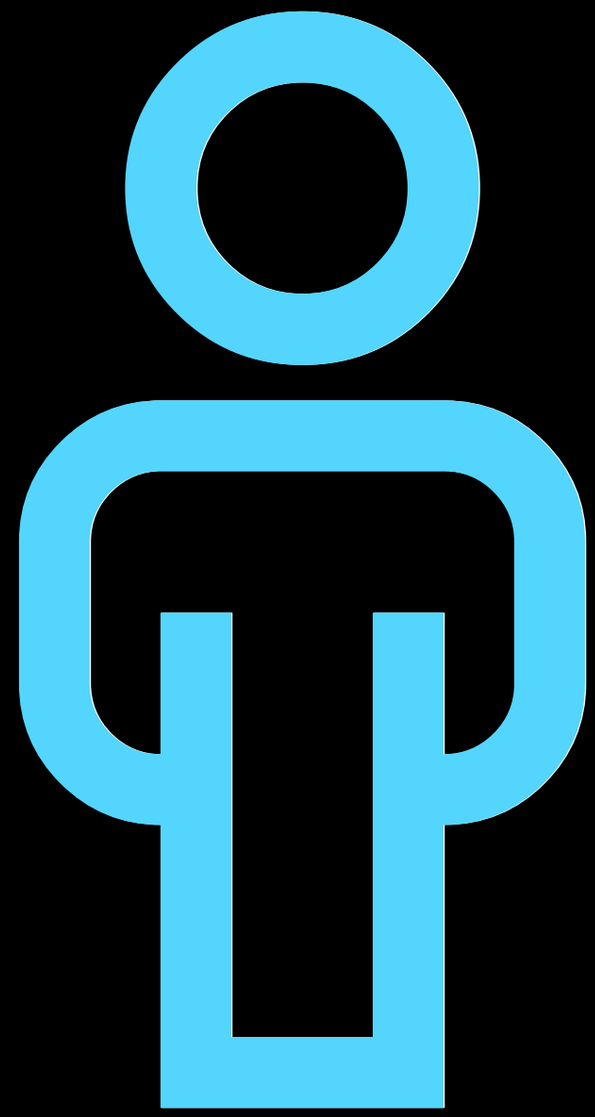




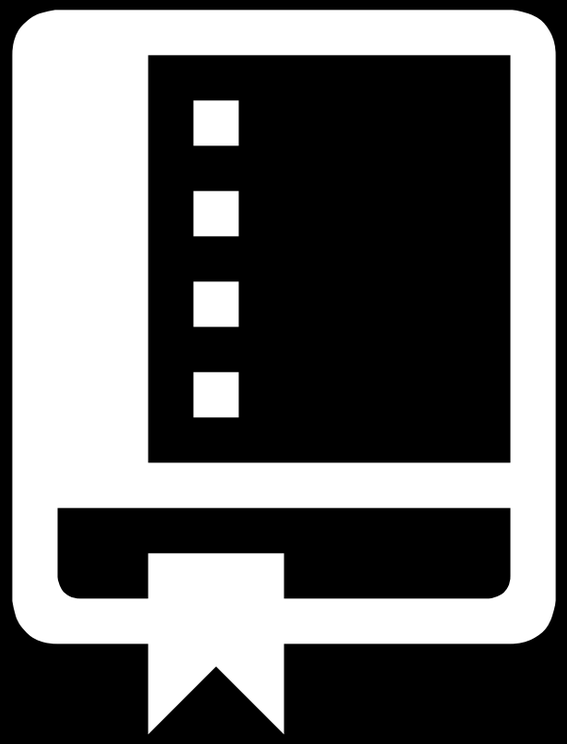


Single developer identities

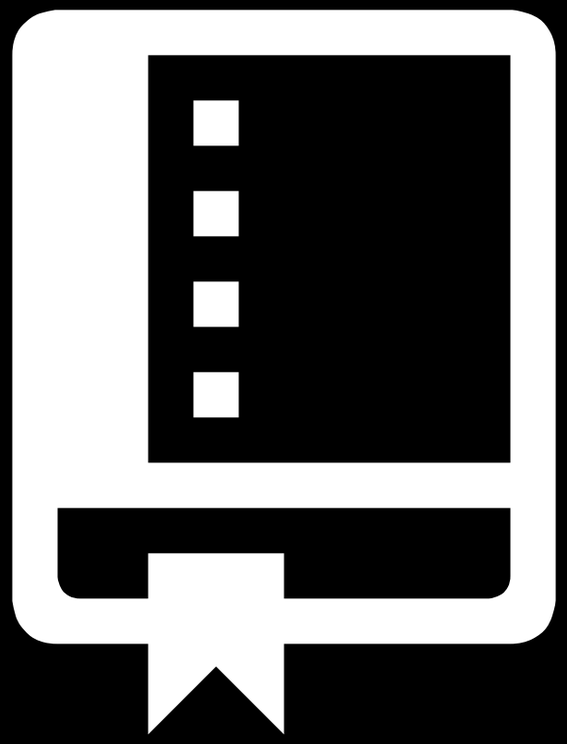




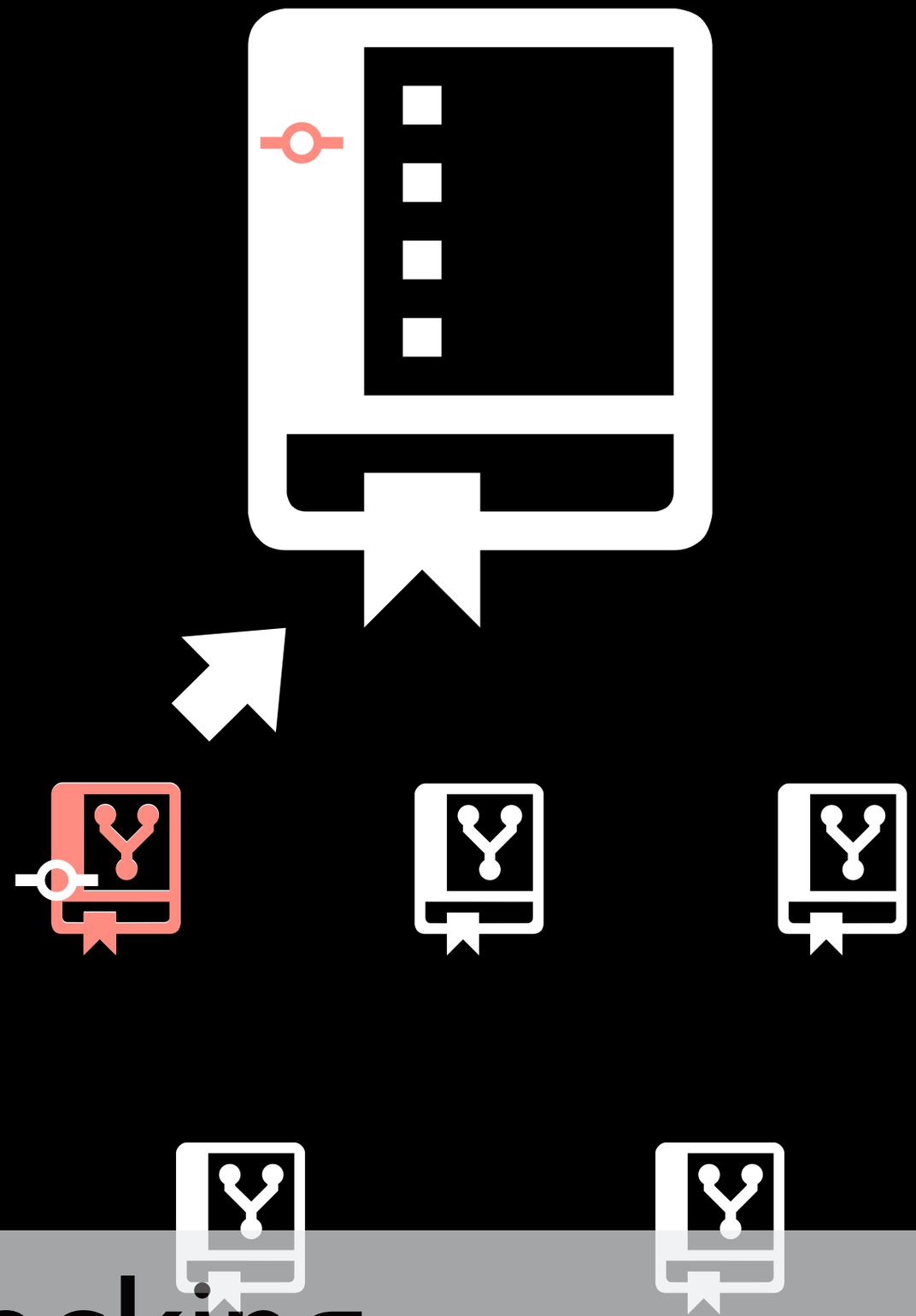
Single developer identities



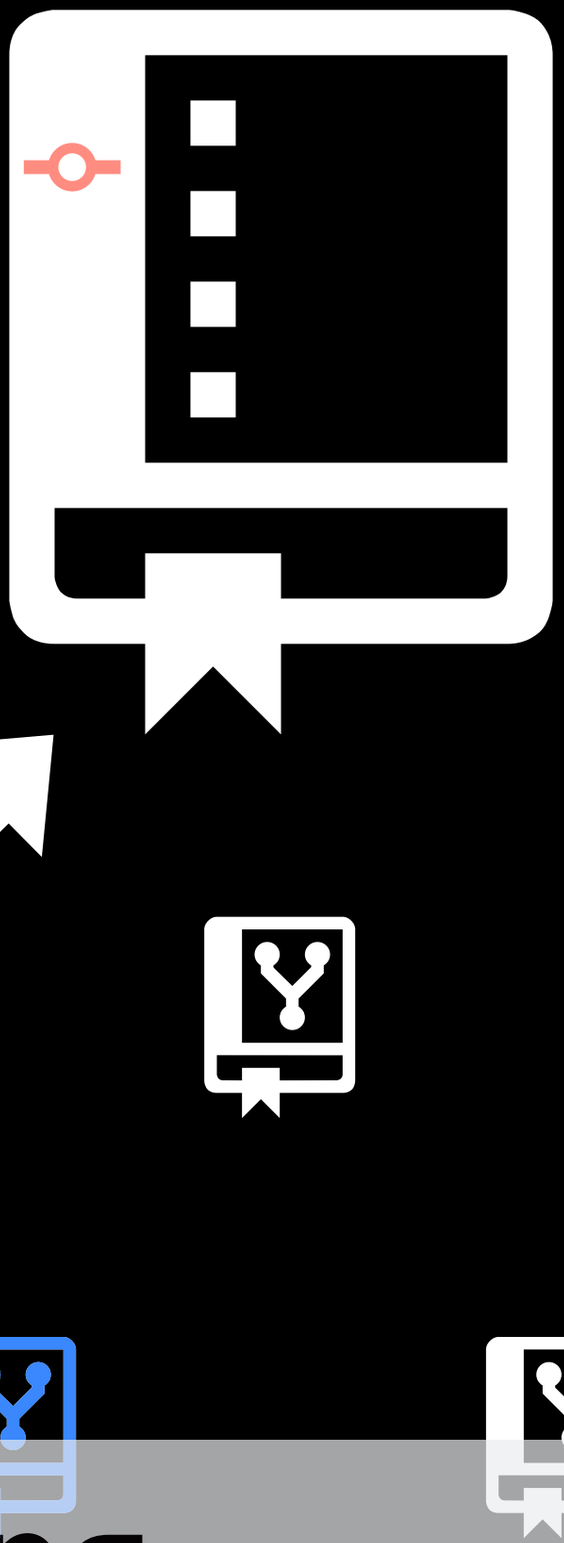
Source tracking



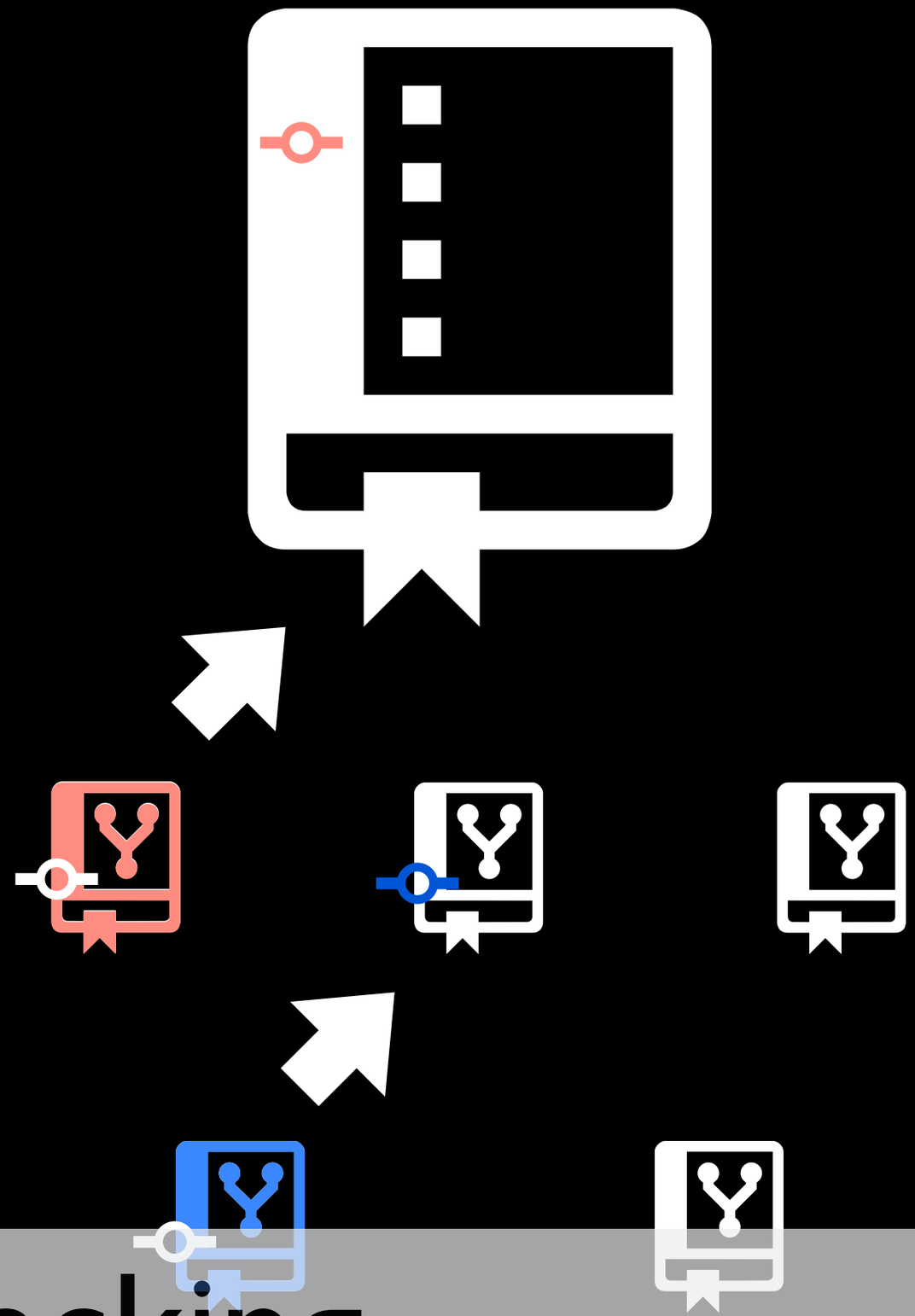
Source tracking



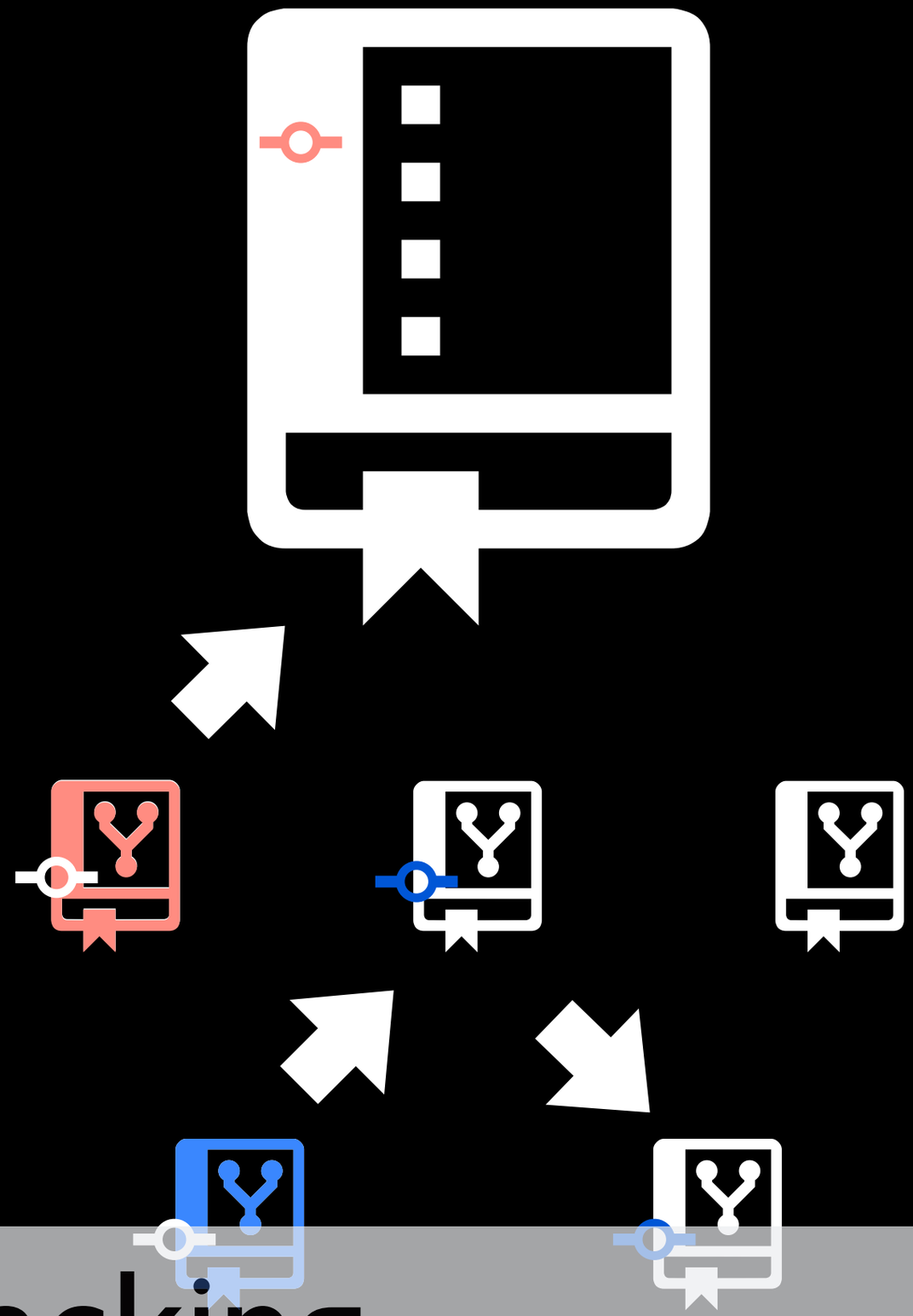
Source tracking



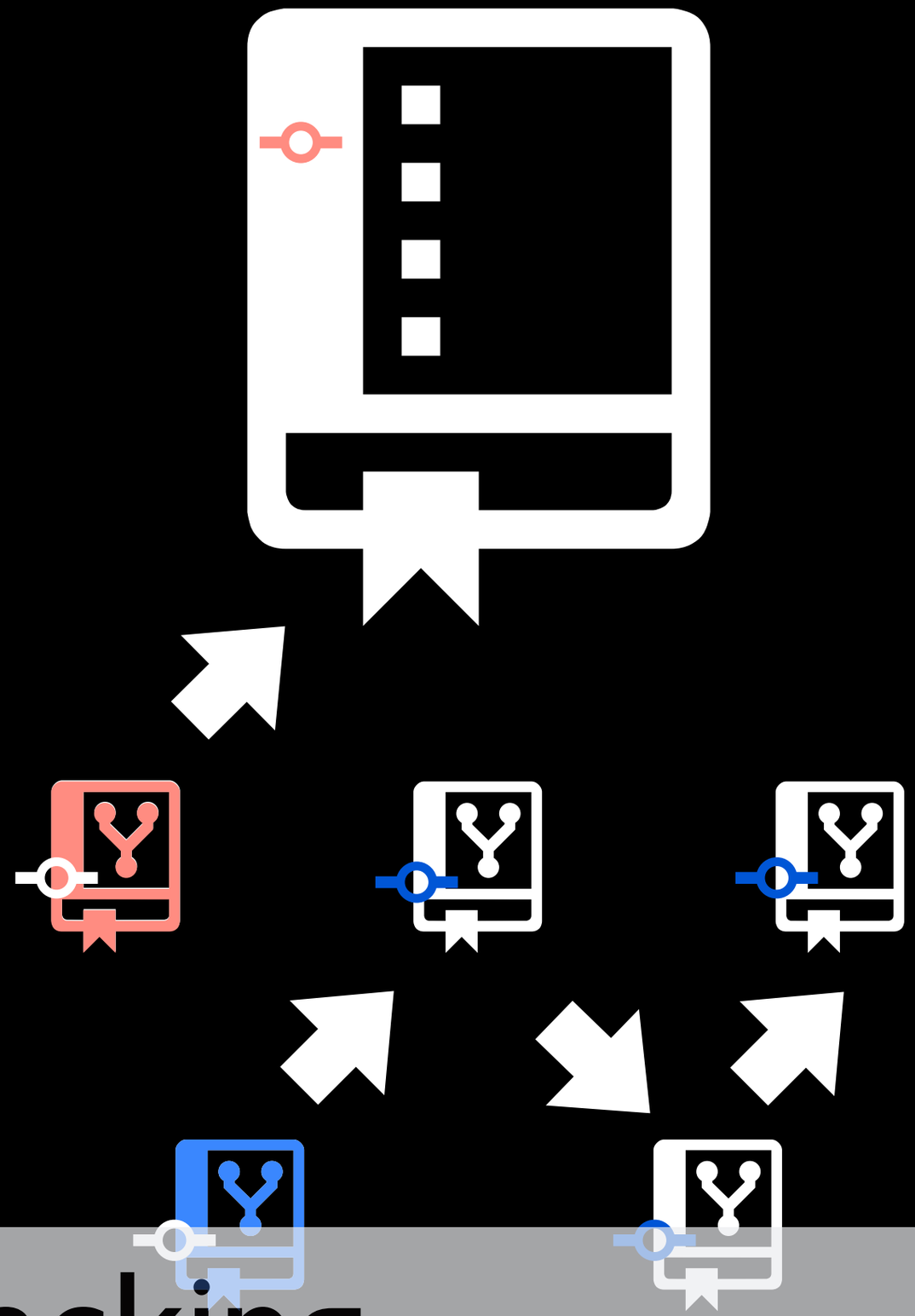
Source tracking



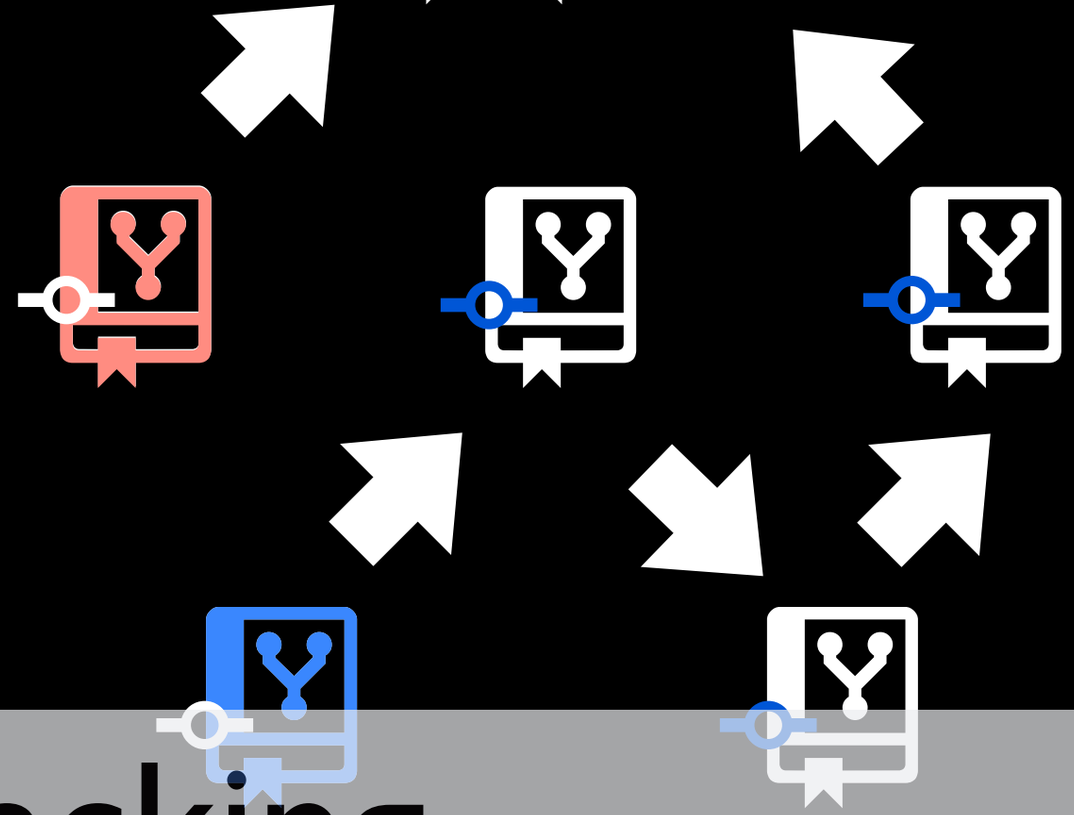
Source tracking



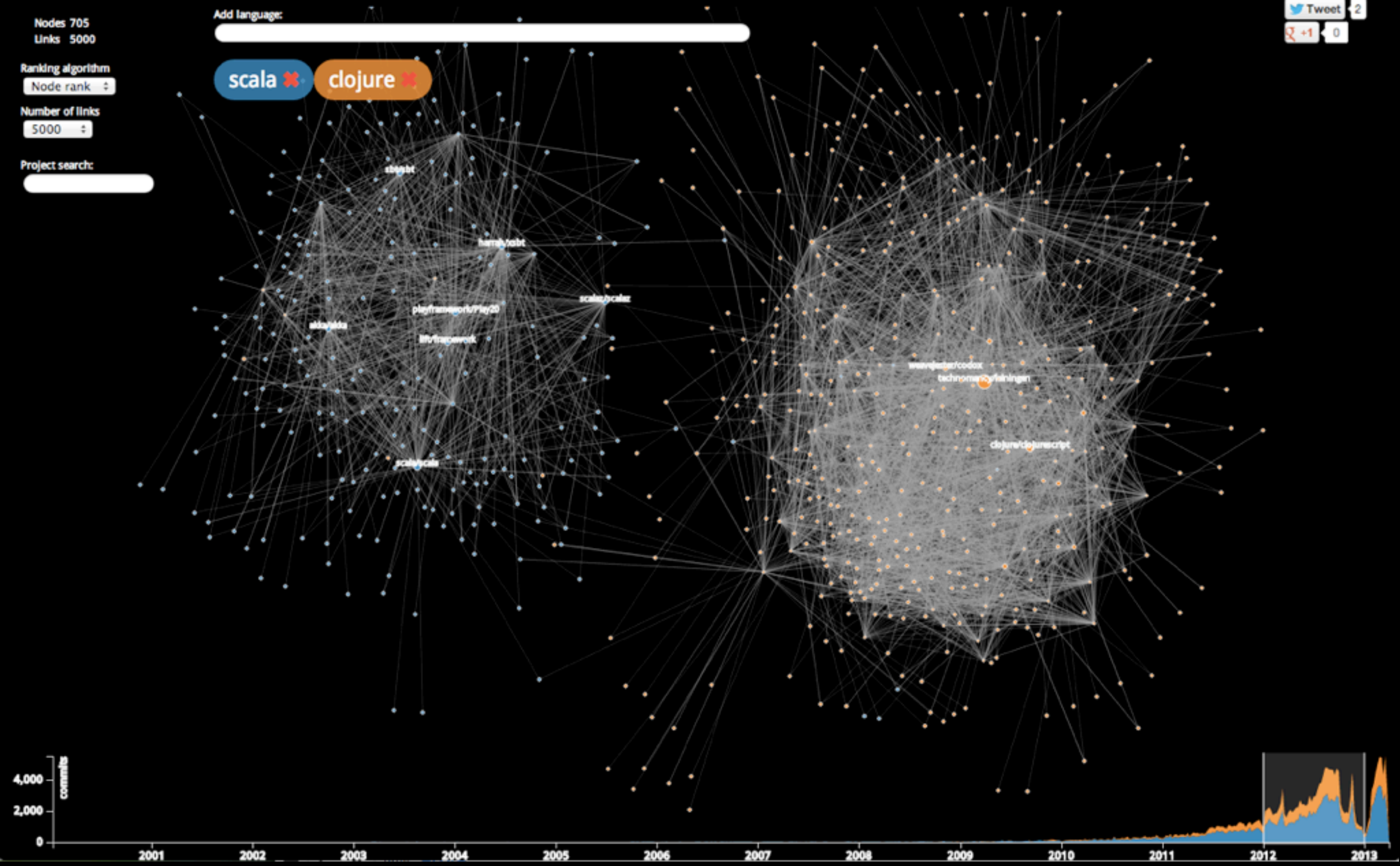
Source tracking



Source tracking



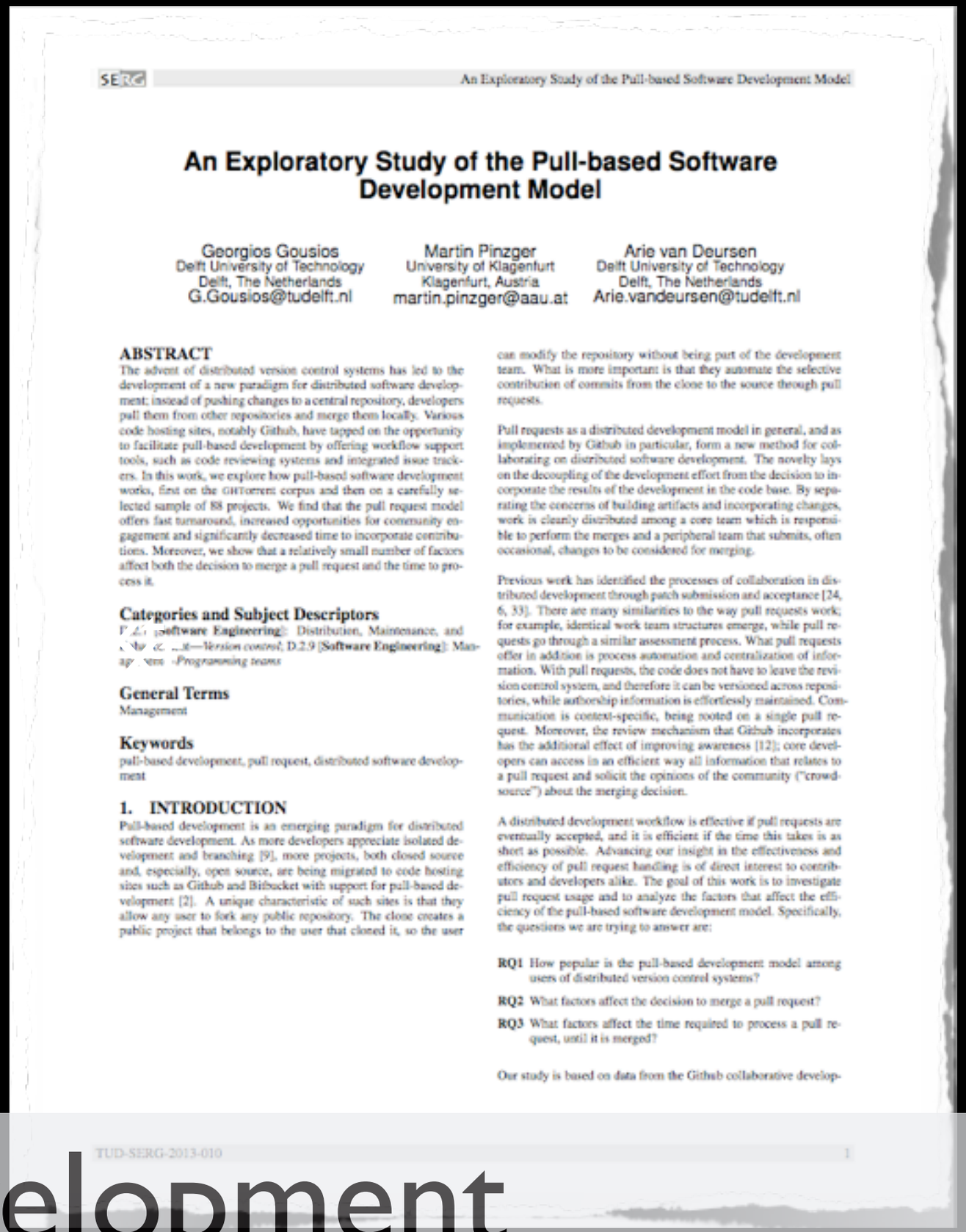
Source tracking



Network analysis

An Exploratory Study of the Pull-based Software Development model

TUD-SERG-2013-10



SERG An Exploratory Study of the Pull-based Software Development Model

An Exploratory Study of the Pull-based Software Development Model

Georgios Gousios
Delft University of Technology
Delft, The Netherlands
G.Gousios@tudelft.nl

Martin Pinzger
University of Klagenfurt
Klagenfurt, Austria
martin.pinzger@aau.at

Arie van Deursen
Delft University of Technology
Delft, The Netherlands
Arie.vandeursen@tudelft.nl

ABSTRACT

The advent of distributed version control systems has led to the development of a new paradigm for distributed software development: instead of pushing changes to a central repository, developers pull them from other repositories and merge them locally. Various code hosting sites, notably Github, have tapped on the opportunity to facilitate pull-based development by offering workflow support tools, such as code reviewing systems and integrated issue trackers. In this work, we explore how pull-based software development works, first on the GHTorrent corpus and then on a carefully selected sample of 88 projects. We find that the pull request model offers fast turnaround, increased opportunities for community engagement and significantly decreased time to incorporate contributions. Moreover, we show that a relatively small number of factors affect both the decision to merge a pull request and the time to process it.

Categories and Subject Descriptors

D.2.3 [**Software Engineering**]: Distribution, Maintenance, and [illegible]—*Version control*; D.2.9 [**Software Engineering**]: Management—*Programming teams*

General Terms

Management

Keywords

pull-based development, pull request, distributed software development

1. INTRODUCTION

Pull-based development is an emerging paradigm for distributed software development. As more developers appreciate isolated development and branching [9], more projects, both closed source and, especially, open source, are being migrated to code hosting sites such as Github and Bitbucket with support for pull-based development [2]. A unique characteristic of such sites is that they allow any user to fork any public repository. The clone creates a public project that belongs to the user that cloned it, so the user can modify the repository without being part of the development team. What is more important is that they automate the selective contribution of commits from the clone to the source through pull requests.

Pull requests as a distributed development model in general, and as implemented by Github in particular, form a new method for collaborating on distributed software development. The novelty lays on the decoupling of the development effort from the decision to incorporate the results of the development in the code base. By separating the concerns of building artifacts and incorporating changes, work is cleanly distributed among a core team which is responsible to perform the merges and a peripheral team that submits, often occasional, changes to be considered for merging.

Previous work has identified the processes of collaboration in distributed development through patch submission and acceptance [24, 6, 33]. There are many similarities to the way pull requests work; for example, identical work team structures emerge, while pull requests go through a similar assessment process. What pull requests offer in addition is process automation and centralization of information. With pull requests, the code does not have to leave the revision control system, and therefore it can be versioned across repositories, while authorship information is effortlessly maintained. Communication is context-specific, being rooted on a single pull request. Moreover, the review mechanism that Github incorporates has the additional effect of improving awareness [12]; core developers can access in an efficient way all information that relates to a pull request and solicit the opinions of the community ("crowdsource") about the merging decision.

A distributed development workflow is effective if pull requests are eventually accepted, and it is efficient if the time this takes is as short as possible. Advancing our insight in the effectiveness and efficiency of pull request handling is of direct interest to contributors and developers alike. The goal of this work is to investigate pull request usage and to analyze the factors that affect the efficiency of the pull-based software development model. Specifically, the questions we are trying to answer are:

RQ1 How popular is the pull-based development model among users of distributed version control systems?

RQ2 What factors affect the decision to merge a pull request?

RQ3 What factors affect the time required to process a pull request, until it is merged?

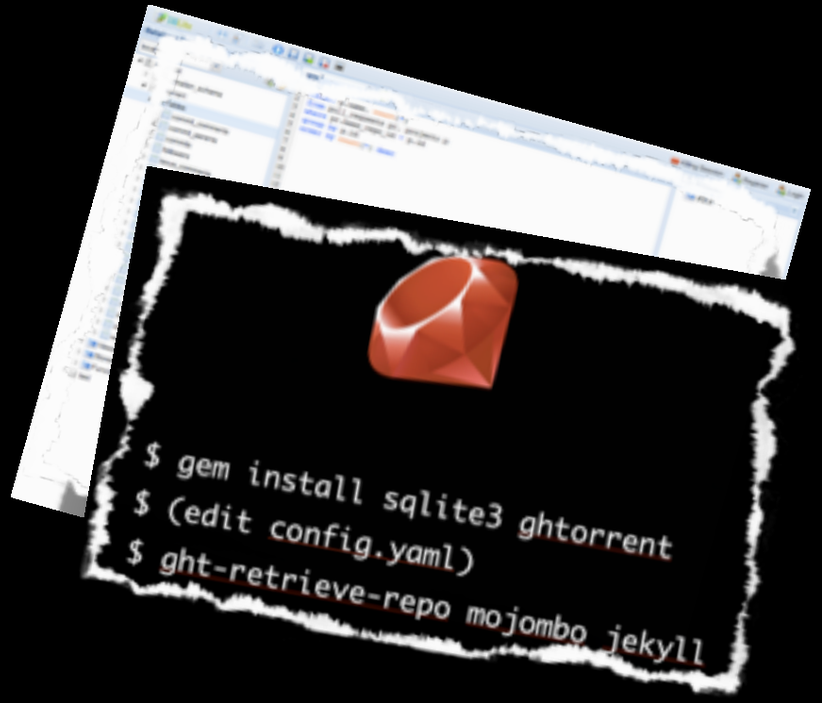
Our study is based on data from the Github collaborative develop-

Distributed development

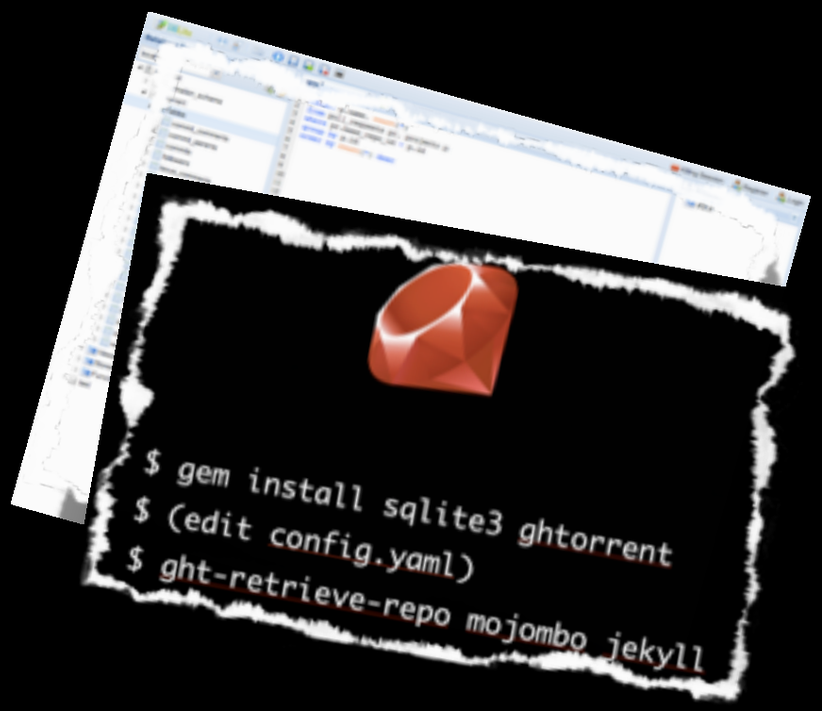

organization_members
project_members
users
followers
pull_request_commits
pull_requests
commit_parents
project_commits
repo_milestones
projects
commits
pull_request_history
repo_labels
pull_events
watchers
issue_comments
commit_comments
pull_request_comments


Downloads
MySQL database dumps
Available Torrents


Downloads
MySQL database dumps
Available Torrents


$ gem install sqlite3 ghtorrent
$ (edit config.yaml)
$ ght-retrieve-repo mojombo jekyll


Downloads
MySQL database dumps
Available Torrents


$ gem install sqlite3 ghtorrent
$ (edit config.yaml)
$ ght-retrieve-repo mojombo jekyll


An Exploratory Study of the Pull-based Software Development Model
scala
haskell
clojure

ghtorrent.org

Octicons font: courtesy Github