Extending GrimoireLab capabilities

GrimoireCon, Brussels, 02-02-2018

Alberto Pérez, Valerio Cosentino
@alpgarcia, @_valcos_
[alpgarcia, valcos]@bitergia.com
https://speakerdeck.com/bitergia

Outline

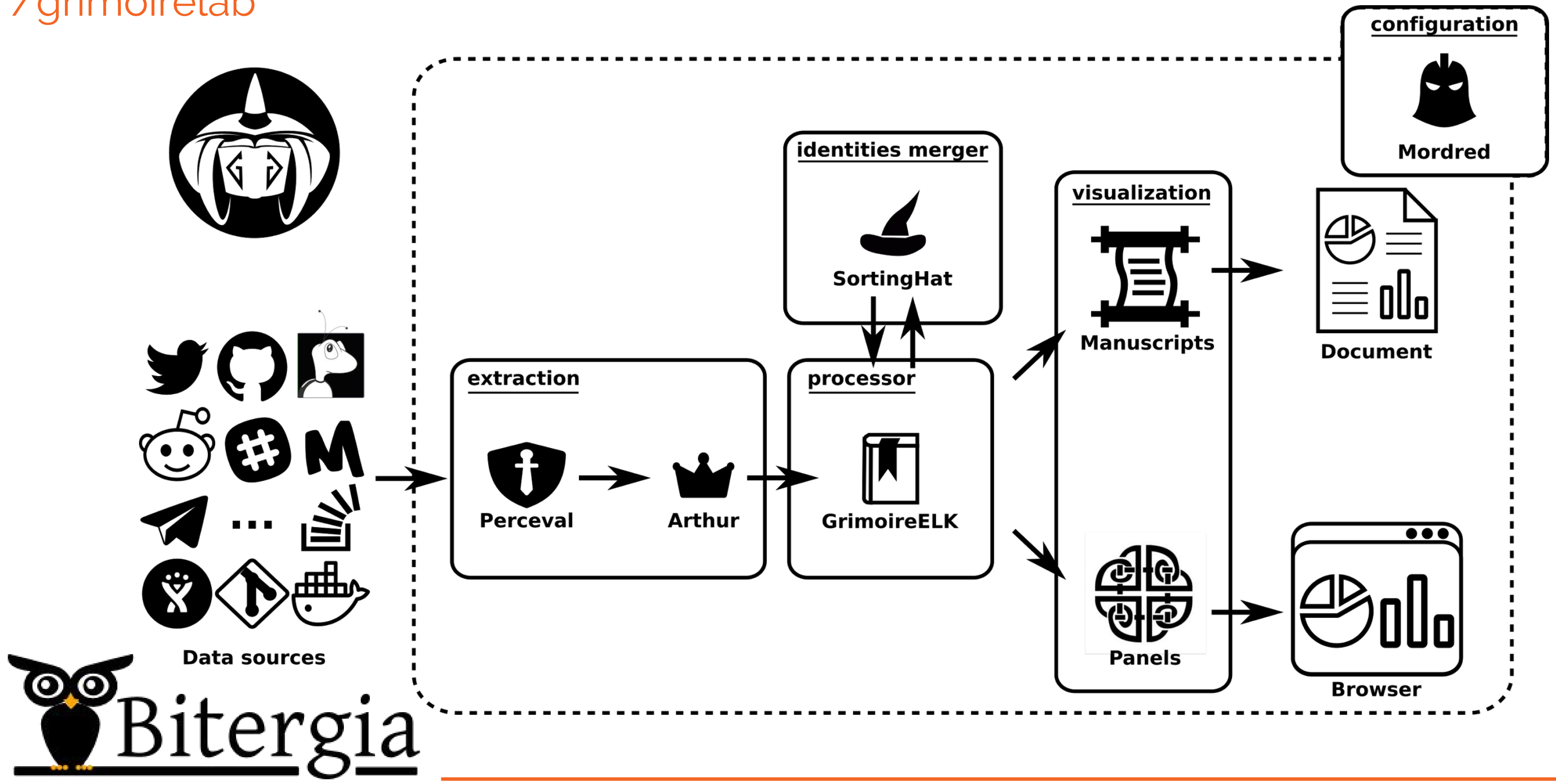
GrimoireLab overview

Use case

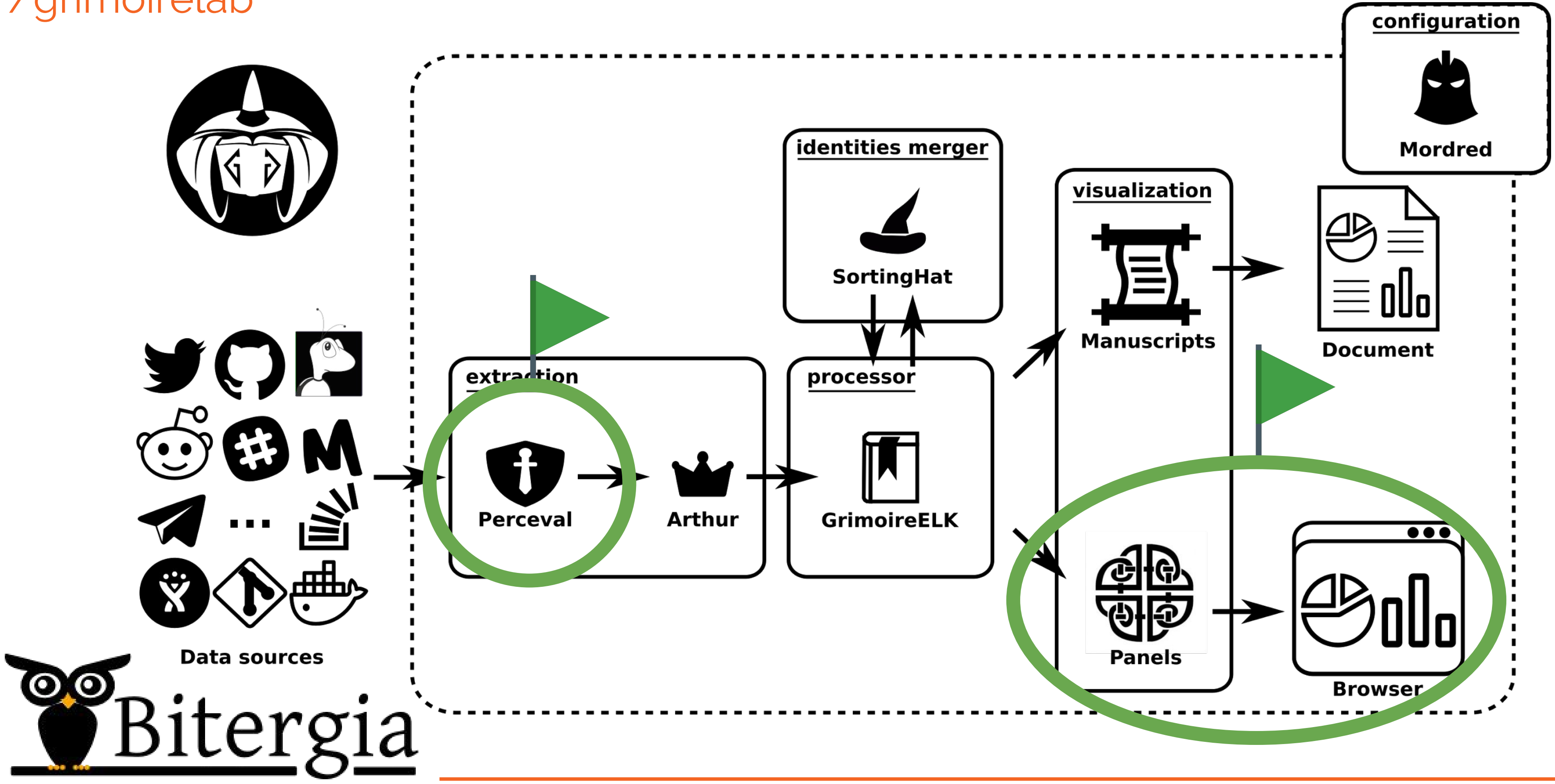
Data extraction

Data visualization

/grimoirelab



/grimoirelab

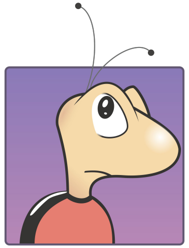


/use_case

Can you prepare a use case?

Sure, what do we show?

Commit's authors and **issues**, ok?



Bitergia

/use_case

Can you prepare a use case?

Sure, what do we show?

Commit's authors and **issues**, ok?

Ok, **comics' authors** and **issues**

Bitergia

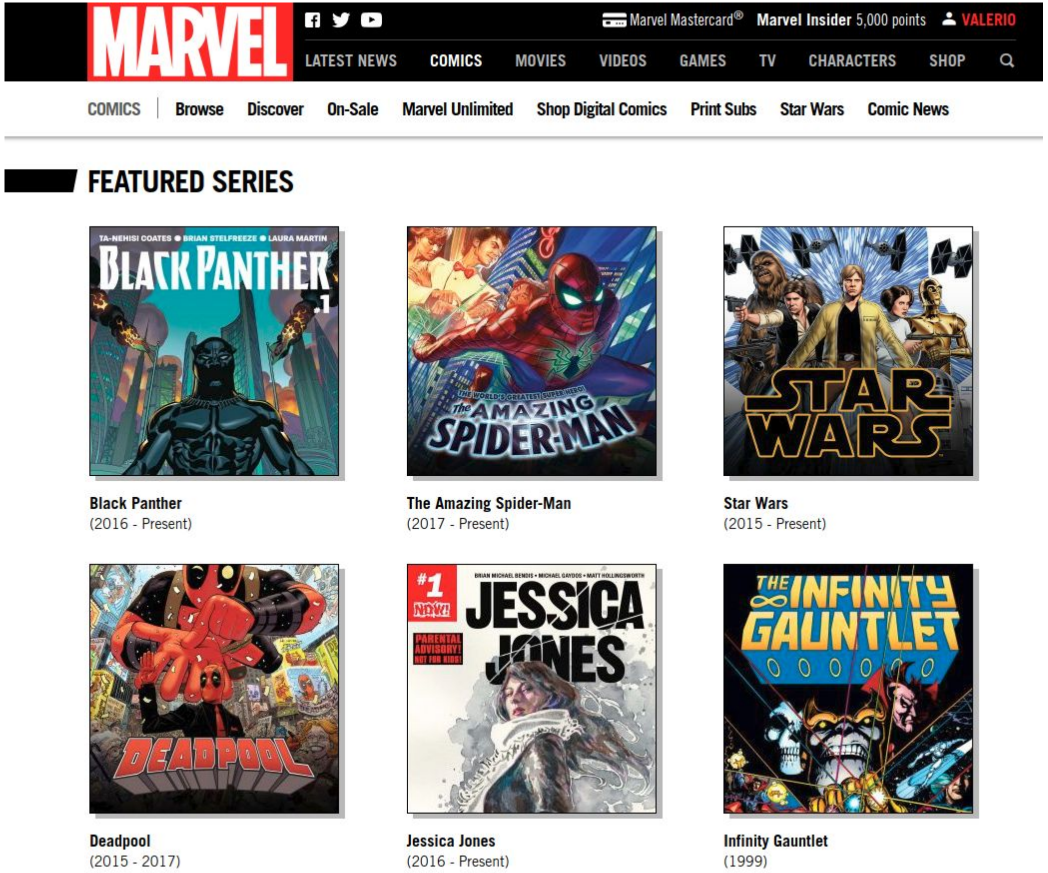
/use_case

Comics

Characters

Creators

Stories



Bitergia

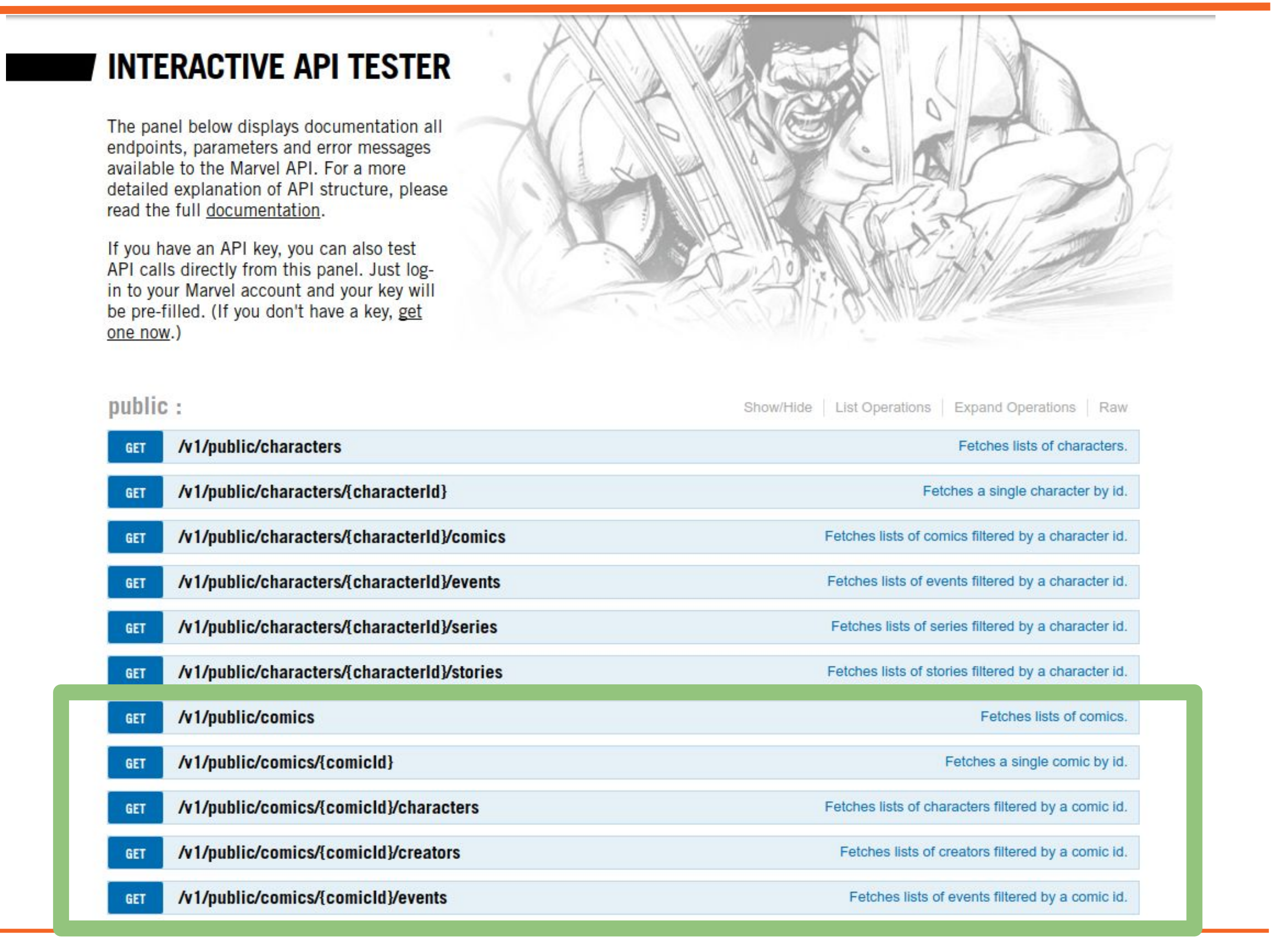
/use_case

Comics

Characters

Creators

Stories



INTERACTIVE API TESTER

The panel below displays documentation all endpoints, parameters and error messages available to the Marvel API. For a more detailed explanation of API structure, please read the full documentation.

If you have an API key, you can also test API calls directly from this panel. Just log-in to your Marvel account and your key will be pre-filled. (If you don't have a key, get one now.)

public : Show/Hide | List Operations | Expand Operations | Raw

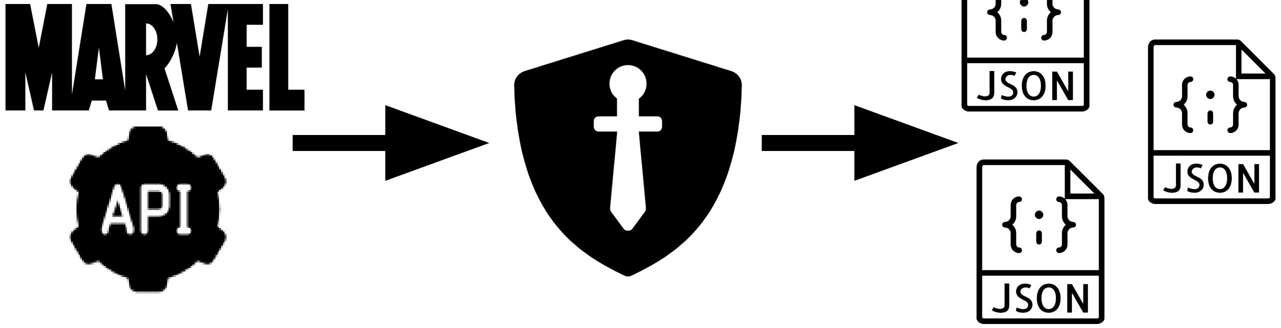
GET	/v1/public/characters	Fetches lists of characters.
GET	/v1/public/characters/{characterId}	Fetches a single character by id.
GET	/v1/public/characters/{characterId}/comics	Fetches lists of comics filtered by a character id.
GET	/v1/public/characters/{characterId}/events	Fetches lists of events filtered by a character id.
GET	/v1/public/characters/{characterId}/series	Fetches lists of series filtered by a character id.
GET	/v1/public/characters/{characterId}/stories	Fetches lists of stories filtered by a character id.
GET	/v1/public/comics	Fetches lists of comics.
GET	/v1/public/comics/{comicId}	Fetches a single comic by id.
GET	/v1/public/comics/{comicId}/characters	Fetches lists of characters filtered by a comic id.
GET	/v1/public/comics/{comicId}/creators	Fetches lists of creators filtered by a comic id.
GET	/v1/public/comics/{comicId}/events	Fetches lists of events filtered by a comic id.

Bitergia

/data_extraction

Goal -> retrieve information* from data sources

Perceval



Bitergia

* information: collection of items (issues, commits, **comics**)

/data_extraction

API (data source) and Perceval data

Perceval

```
{   "backend_name": "Marvel",
    "backend_version": "0.1.0",
    "category": "comic",
    "data": {
            "format": "Comic",
            "id": 37030,
            "issueNumber": 2,
            "modified": "2010-08-04T01:32:01-0400",
            "pageCount": 32,
            "prices": [...],
            "characters": {...},
            "characters_data": {...}
    },
    "origin": "https://developer.marvel.com/",
    "perceval_version": "0.9.10",
    "tag": "https://developer.marvel.com/",
    "timestamp": 1517421033.892423,
    "updated_on": 1280899921.0,
    "uuid": "cc6fc7e818e48a18e498b2e865e554a1aa27b317" }
```

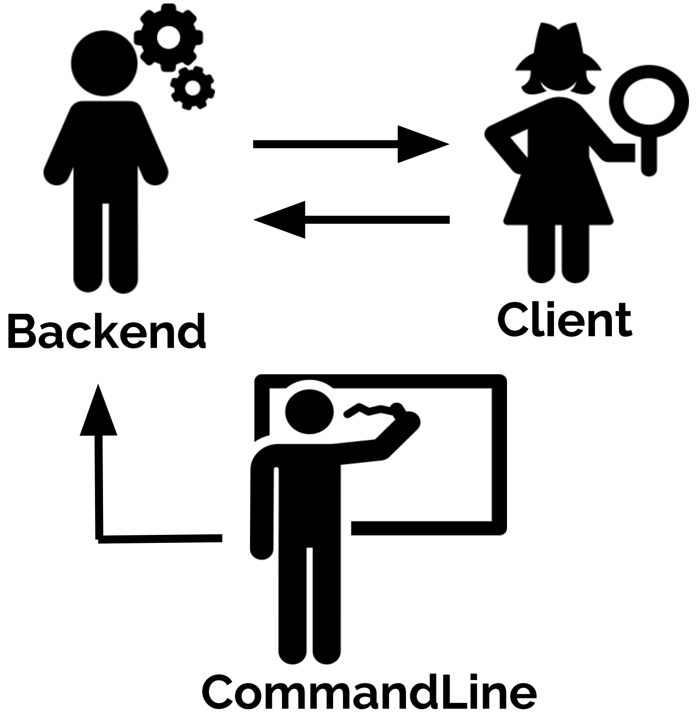
API data

Perceval data

Bitergia

Perceval

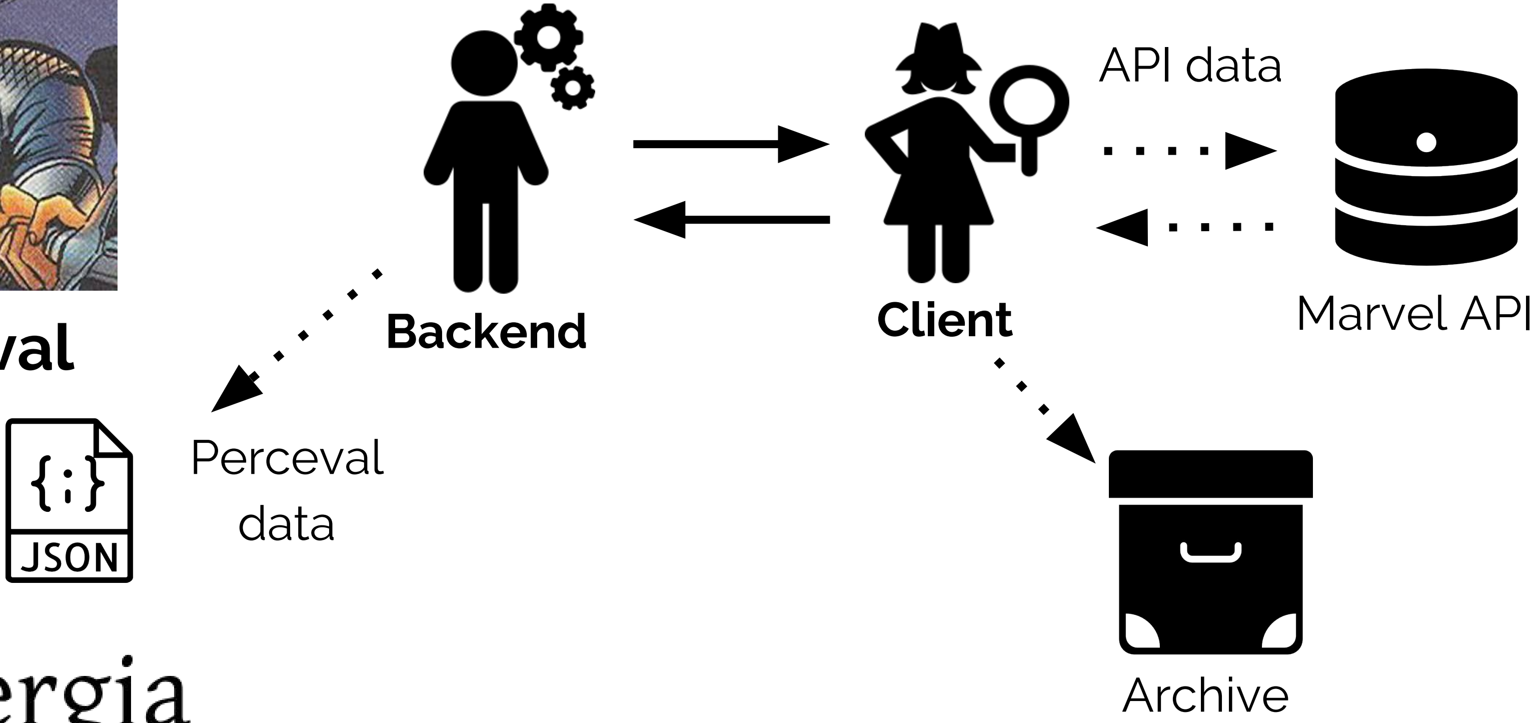
Organization -> 3 actors



Bitergia

Operations -> **fetch** & fetch-from-archive

Perceval



Bitergia

/fetch

Perceval

Backend

```
def fetch(self, from_date=DEFAULT_DATETIME):
  ...
  from_date = datetime_to_utc(from_date)

  kwargs = {"from_date": from_date}
  items = super().fetch("comic", **kwargs)

  return items
```

Bitergia

/fetch

Perceval

Backend

```
def fetch(self, from_date=DEFAULT_DATETIME):
  ...
  from_date = datetime_to_utc(from_date)

  kwargs = {"from_date": from_date}
  items = super().fetch("comic", **kwargs)

  return items
```

```
def fetch(self, category, **kwargs):
  if self.archive:
     self.archive.init_metadata(...)

self.client = self._init_client()
  for item in self.fetch_items(**kwargs):
     yield self.metadata(item)
```

```
def _init_client(self, from_archive=False):
   return MarvelClient(self.public_key,
                       self.private_key,
                       ...,
                      self.max_retries,
                      self.archive,
                      from_archive)
```

Bitergia

/fetch

Perceval

Backend

```
def fetch(self, from_date=DEFAULT_DATETIME):
  ...
  from_date = datetime_to_utc(from_date)

  kwargs = {"from_date": from_date}
  items = super().fetch("comic", **kwargs)

  return items
```

```
def fetch(self, category, **kwargs):
  if self.archive:
     self.archive.init_metadata(...)

self.client = self._init_client()
  for item in self.fetch_items(**kwargs):
     yield self.metadata(item)
```

```
def fetch_items(self, **kwargs):
  from_date = kwargs['from_date']
  comic_groups = self.client.comics(from_date)

  for comics in comic_groups:
     for comic in comics:
        ...
        comic['characters_data'] =
                                self.client.comic_data(...)
        ...
        yield comic
```

Bitergia

/fetch

Perceval

Backend

```
def fetch(self, from_date=DEFAULT_DATETIME):
  ...
  from_date = datetime_to_utc(from_date)

  kwargs = {"from_date": from_date}
  items = super().fetch("comic", **kwargs)

  return items
```

```
def fetch(self, category, **kwargs):
  if self.archive:
    self.archive.init_metadata(...)

self.client = self._init_client()
  for item in self.fetch_items(**kwargs):
    yield self.metadata(item)
```

```
def fetch_items(self, **kwargs):
  from_date = kwargs['from_date']
  comic_groups = self.client.comics(from_date)

  for comics in comic_groups:
    for comic in comics:
      ...
      comic['characters_data'] =
                        self.client.comic_data(...)
      ...
      yield comic
```

Client

Bitergia

/fetch

Client

Perceval

```
def comics(self, from_date=None):
    payload = {
        'orderBy': 'modified',
        'limit': self.items_per_page
    }
    if from_date:
        payload['modifiedSince'] = from_date.isoformat()
    ...
    path = urijoin(MARVEL_API_URL, "comics")
    return self.fetch_items(path, payload)
```

/fetch

Perceval

Client

```
def comics(self, from_date=None):
  payload = {
     'orderBy': 'modified',
     'limit': self.items_per_page
  }
  if from_date:
     payload['modifiedSince'] = from_date.isoformat()
  ...
  path = urijoin(MARVEL_API_URL, "comics")
  return self.fetch_items(path, payload)
```

```
def fetch_items(self, path, payload):
  response = self.fetch(path, payload=payload)
  items_info = response.json()['data']

  total = items_info['total']
  count = items_info['count']

  while True:
     yield items_info['results']
    ...code for pagination..
```

Bitergia

/fetch

Perceval

Bitergia

Client

```
def comics(self, from_date=None):
  payload = {
     'orderBy': 'modified',
     'limit': self.items_per_page
  }
  if from_date:
     payload['modifiedSince'] = from_date.isoformat()
  ...
  path = urijoin(MARVEL_API_URL, "comics")
  return self.fetch_items(path, payload)
```

```
def fetch_items(self, path, payload):
  response = self.fetch(path, payload=payload)
  items_info = response.json()['data']

  total = items_info['total']
  count = items_info['count']

  while True:
     yield items_info['results']
    ...code for pagination..
```

```
def fetch(self, url, payload=None, headers=None, ...):
 if self.from_archive:
     response = self._fetch_from_archive(url, payload,
                                                   headers)
  else:
     response = self._fetch_from_remote(url, payload,
                                                   headers, ...)

  return response
```

/fetch

Perceval

Client

```
def comics(self, from_date=None):
  payload = {
     'orderBy': 'modified',
     'limit': self.items_per_page
  }
  if from_date:
     payload['modifiedSince'] = from_date.isoformat()
  ...
  path = urijoin(MARVEL_API_URL, "comics")
  return self.fetch_items(path, payload)
```

```
def fetch_items(self, path, payload):
  response = self.fetch(path, payload=payload)
  items_info = response.json()['data']

  total = items_info['total']
  count = items_info['count']

  while True:
     yield items_info['results']
    ...code for pagination..
```

```
def fetch(self, url, payload=None, headers=None, ...):
 if self.from_archive:
     response = self._fetch_from_archive(url, payload,
                                                      headers)
 else:
     response = self._fetch_from_remote(url, payload,
                                                       headers, ...)

 return response
```

```
def _fetch_from_remote(self, ...):
  response = ...
  try:
      response.raise_for_status()
  except Exception as e:
      response = e
      raise e
  finally:
     if self.archive:
        self.archive.store(..., response)
  return response
```

Bitergia

Perceval

Recap

Backend

```
def fetch(self, from_date=DEFAULT_DATETIME):
↓
def fetch(self, category, **kwargs):
↓
def fetch_items(self, **kwargs):

def _init_client(...):
```

Client

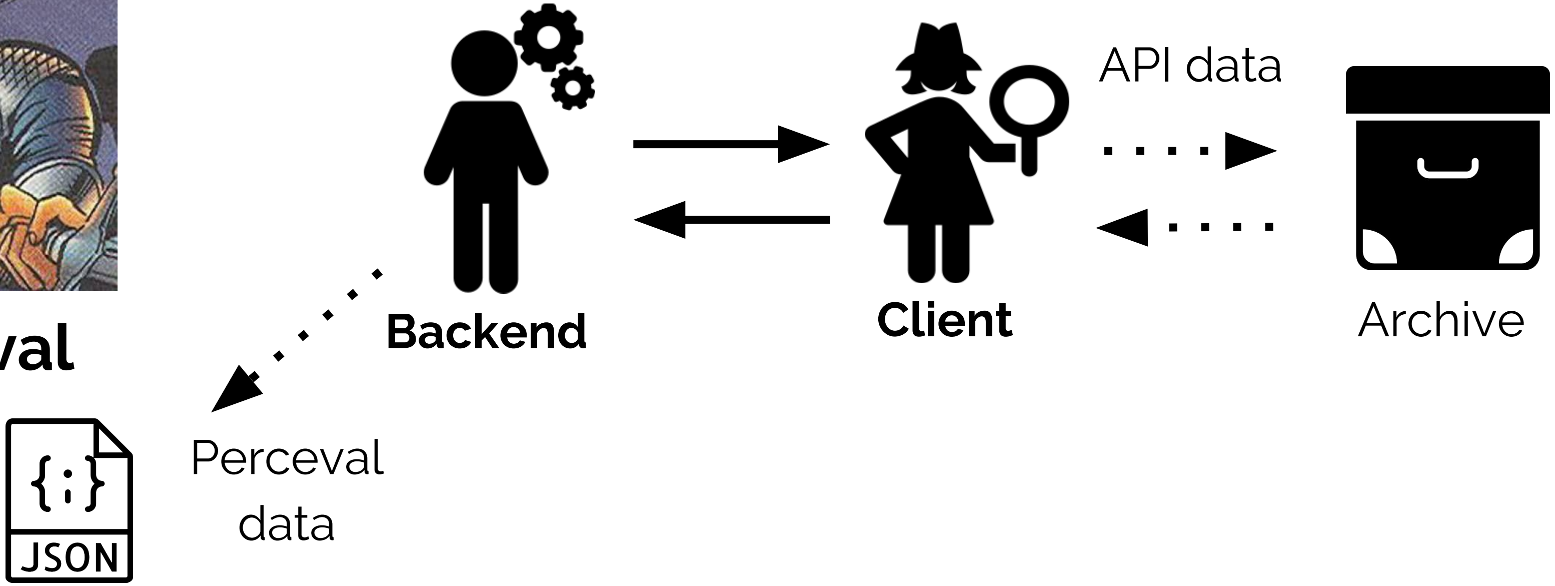
```
def comics(self, from_date=None):
↓
def fetch_items(self, path, payload):
↓
def fetch(self, url, payload=None, headers=None, ...):
↓
def _fetch_from_remote(self, ...):
```

Bitergia

/fetch-from-archive

Perceval

Operations -> fetch & **fetch-from-archive**



Bitergia

/fetch-from-archive

Perceval

Backend

```
def fetch_from_archive(self):
  if not self.archive:
     raise ArchiveError(cause="...")

  self.client = self._init_client(from_archive=True)
  self.archive._load_metadata()

  for item in

self.fetch_items(**self.archive.backend_params):
     yield self.metadata(item)
```

/fetch-from-archive

Perceval

Backend

```
def fetch_from_archive(self):
  if not self.archive:
     raise ArchiveError(cause="...")

  self.client = self._init_client(from_archive=True)
  self.archive._load_metadata()

  for item in

self.fetch_items(**self.archive.backend_params):
     yield self.metadata(item)
```

```
def fetch_items(self, path, payload):
```

Bitergia

/fetch-from-archive

Perceval

Backend

```
def fetch_from_archive(self):
  if not self.archive:
     raise ArchiveError(cause="...")

  self.client = self._init_client(from_archive=True)
  self.archive._load_metadata()

  for item in

self.fetch_items(**self.archive.backend_params):
     yield self.metadata(item)
```

```
def fetch_items(self, path, payload):
```

Client

/fetch-from-archive

Perceval

Client

```
def comics(self, from_date=None):
```

```
def fetch_items(self, path, payload):
```

```
def fetch(self, url, payload=None, headers=None, ...):
  if self.from_archive:
     response = self._fetch_from_archive(url, payload,
                                         headers)
  else:
     response = self._fetch_from_remote(url, payload,
                                        headers, ...)

  return response
```

/fetch-from-archive

Perceval

Bitergia

Client

```
def comics(self, from_date=None):
```

```
def fetch_items(self, path, payload):
```

```
def fetch(self, url, payload=None, headers=None, ...):
 if self.from_archive:
     response = self._fetch_from_archive(url, payload,
                                         headers)
 else:
     response = self._fetch_from_remote(url, payload,
                                        headers, ...)

 return response
```

```
def _fetch_from_archive(self, ...):

  response = self.archive.retrieve(url,
                                   payload,
                                   headers)

  if not isinstance(response, requests.Response):
     raise response

  return response
```

/fetch-from-archive

Perceval

Recap

Backend

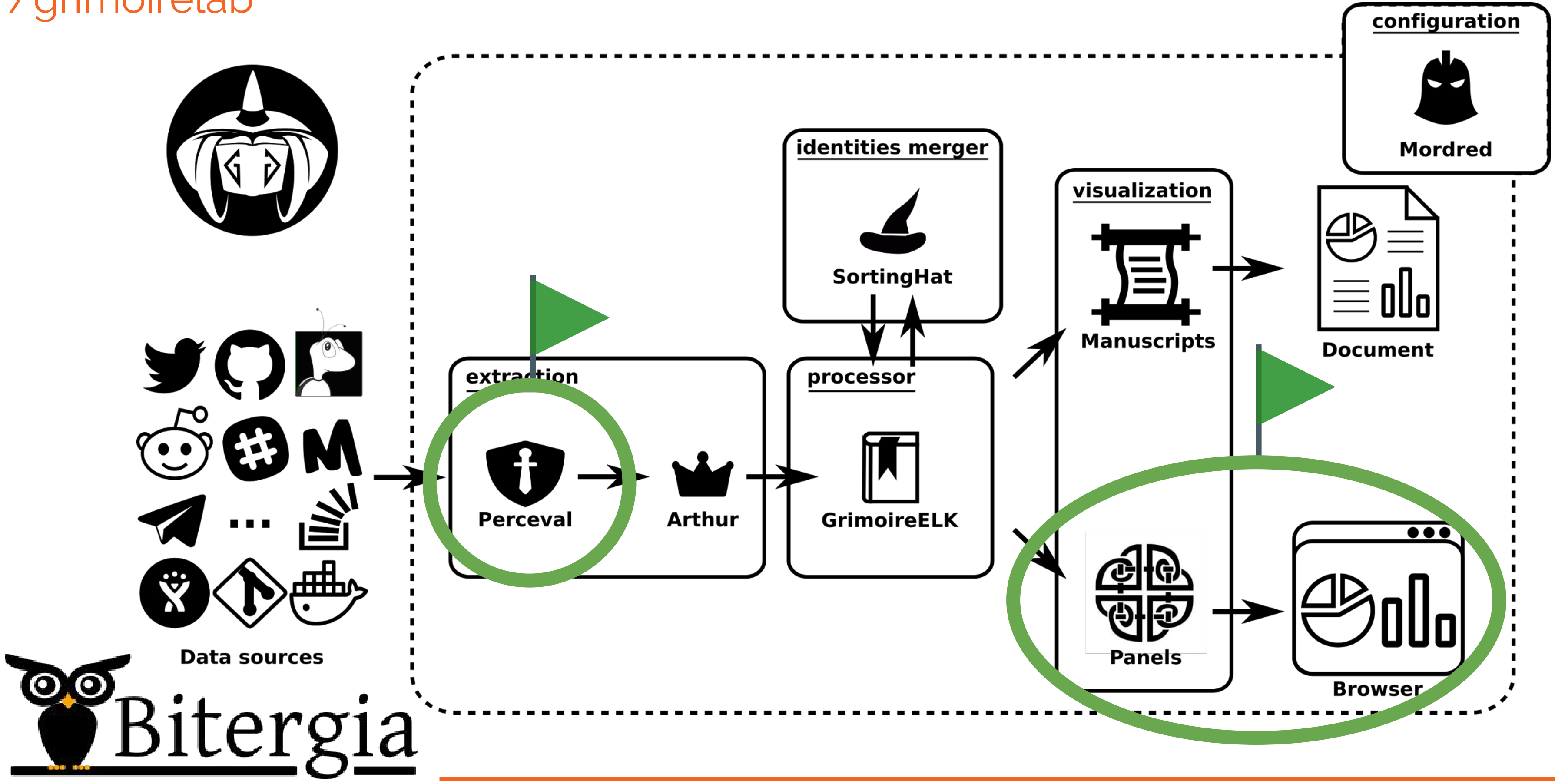
```
def fetch_from_archive(self):
def fetch_items(self, **kwargs):
def _init_client(...):
```

Client

```
def comics(self, from_date=None):
def fetch_items(self, path, payload):
def fetch(self, url, payload=None, headers=None, ...):
def _fetch_from_archive(self, ...):
```

Bitergia

/grimoirelab



Raw index:

```
comic: {
     comic_id: …,
     title: …,
     creators: [{
          name: …,
          role: …
     },{
     …
     }],
     a lot of additional info
}
```

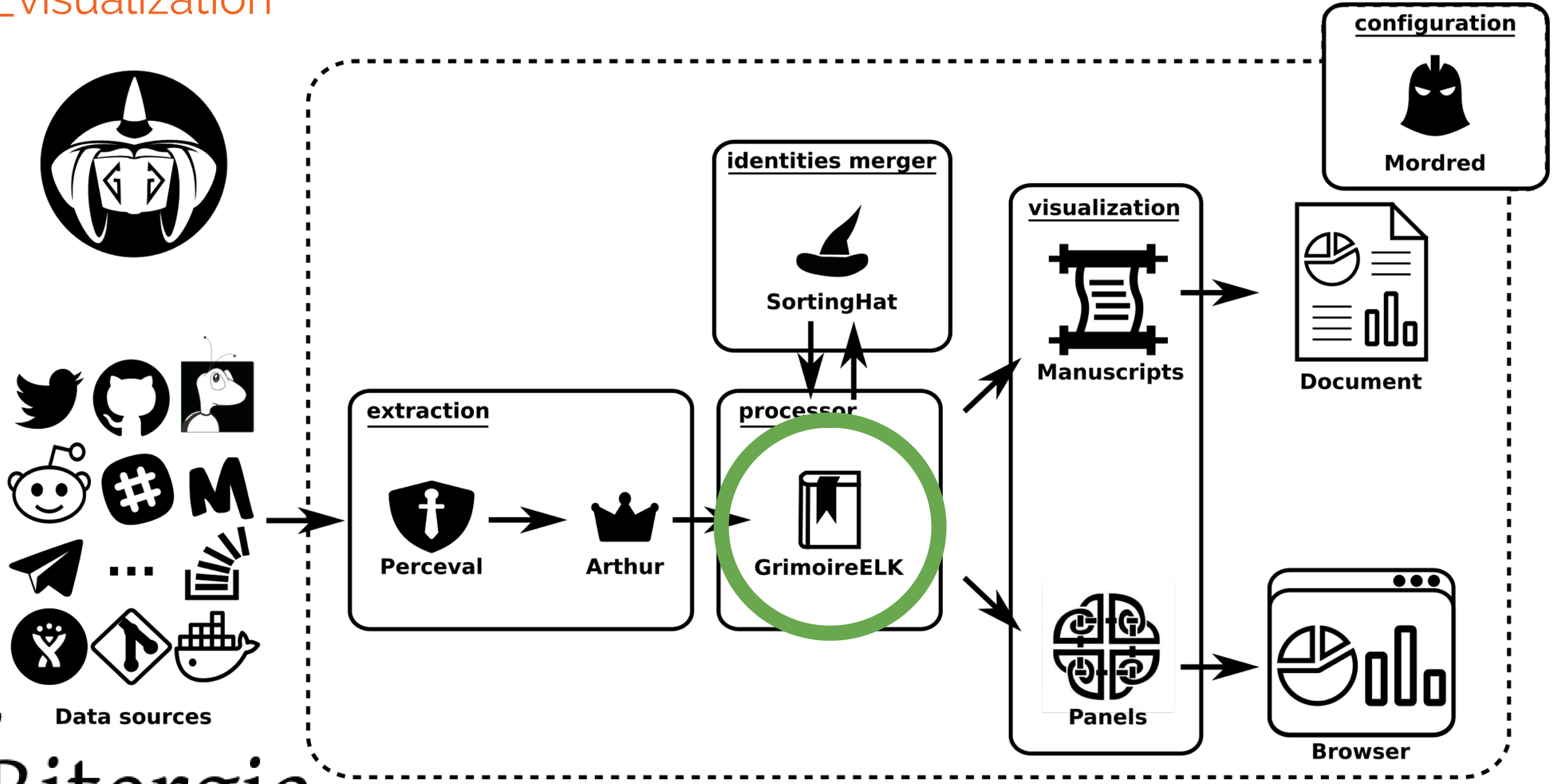
Problem:
there is no way to associate author and role in Kibana.

```
Enriched index:

    author: {
        comic_id: …,
        title: …,
        name: …,
        role: …,
        only some carefully selected info
    }
```

Solution:
Store data from author point of view.

/data_visualization

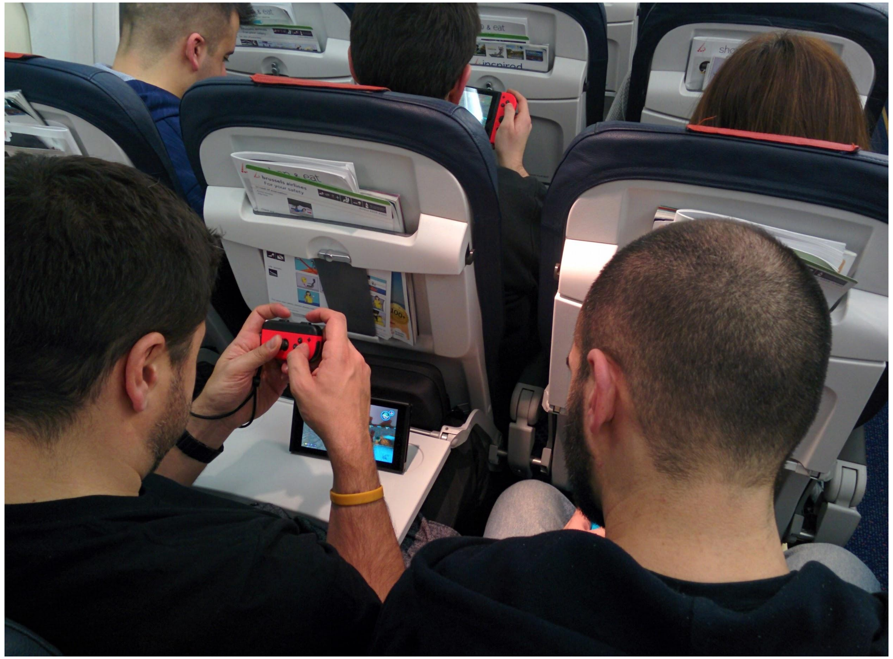


Bitergia

/data_visualization

We needed some help, but our colleagues were a bit busy...

We needed some help, but our colleagues were a bit busy...



Hey Álvaro, we need you as the one and only expert in Gelk!

Mmmm, what do you need guys?

We need to enrich some data related to….Marvel comics

Are you kidding me? Marvel comics???

/data_visualization

Hey Álvaro, we need you as the one and only expert in Gelk!

Mmmm, what do you need guys?

We need to enrich some data related to….Marvel comics

Are you kidding me? Marvel comics???

I'M IN!!!

Bitergia

Extend **Enrich** class:

```
class MarvelEnrich(Enrich):
```

From each **raw item** (comic) create N **enriched items** (creators):

```
def enrich_items(self, ocean_backend):
    ....
```

For each comic, extract creators

```
    for item in items:
        creators = self.get_rich_item_creators(item)
        rich_item_creators += creators
```

Upload new items

```
    if rich_item_creators:
        ncreators = self.elastic.bulk_upload(rich_item_creators, "id")
```

Bitergia

For each **creator** just copy things from here to there:

```
def get_rich_item_creators(self, item):
        ...
        for creator in item['data']['creators']['items']:
            ecreator = self.get_rich_comic_creator(item, creator)
            creators_enrich.append(ecreator)

        return (creators_enrich)
```

And add some **common fields**:

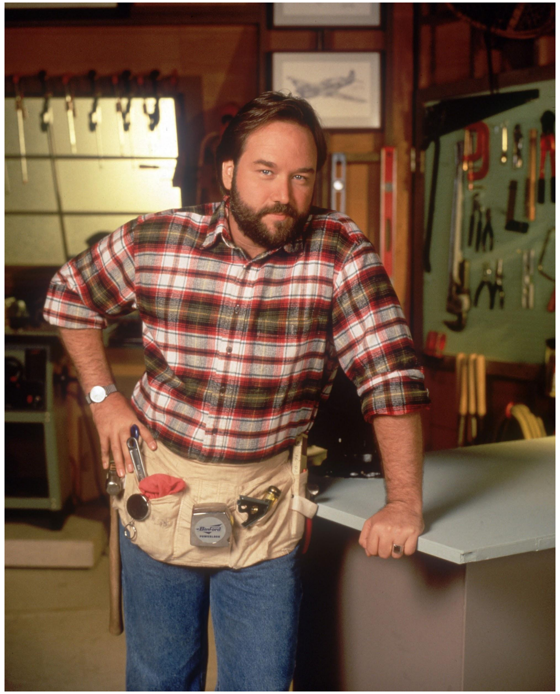
```
# Thumbnails
eitem['url_thumbnail'] = item['data']['thumbnail']['path']
```

/data_visualization

...some hours of Kibana hacking later...

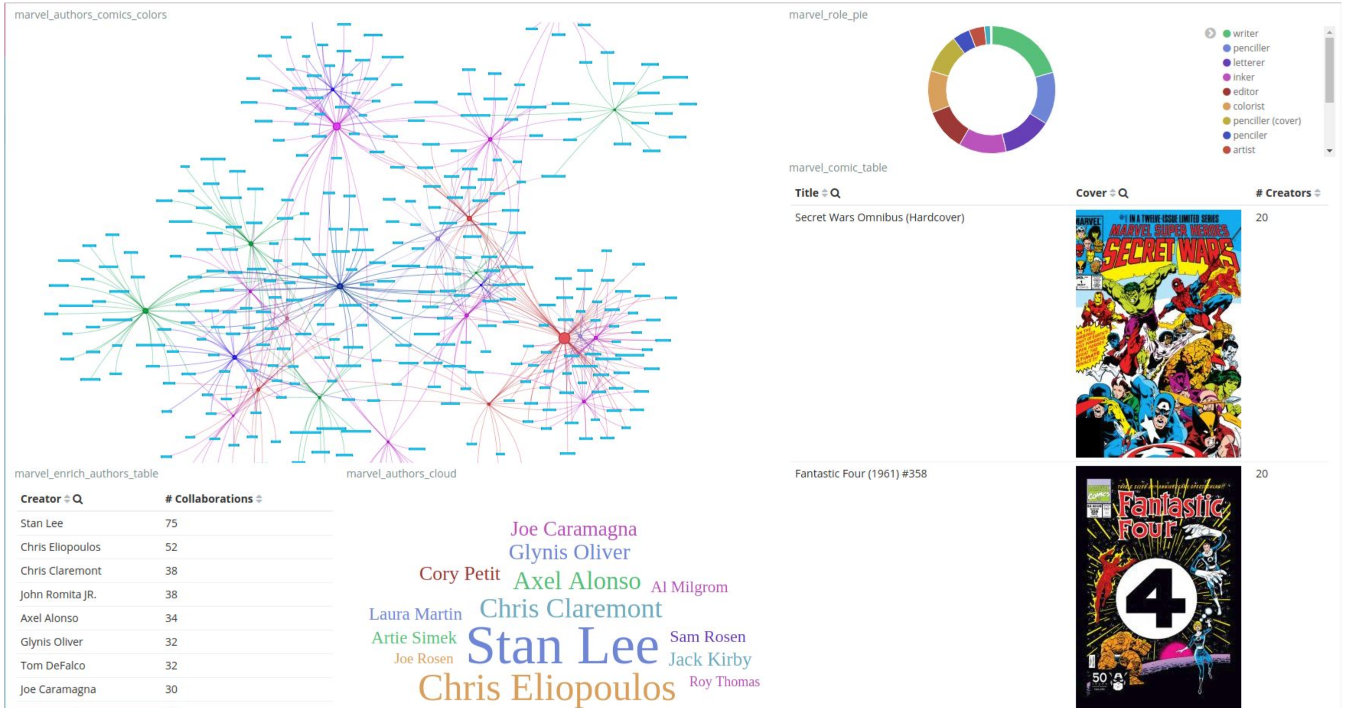
...happy hacking hours, let me say...

...and after some hours more with some help of @dmoreno



/data_visualization

...and after some hours more



marvel_authors_comics_colors

marvel_role_pie

marvel_comic_table

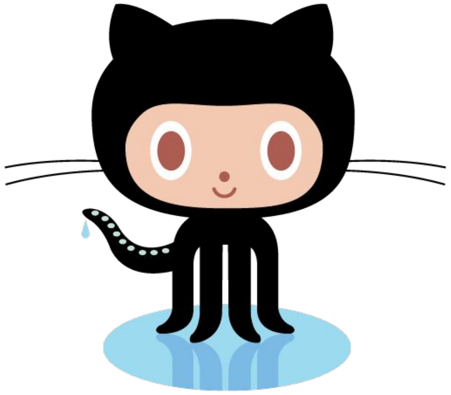
Title	Cover	# Creators
Secret Wars Omnibus (Hardcover)		20
Fantastic Four (1961) #358		20

marvel_enrich_authors_table

Creator	# Collaborations
Stan Lee	75
Chris Eliopoulos	52
Chris Claremont	38
John Romita JR.	38
Axel Alonso	34
Glynis Oliver	32
Tom DeFalco	32
Joe Caramagna	30

marvel_authors_cloud

/resources



grimoirelab/panels
grimoirelab/perceval

alpgarcia/grimoirecon18/marvel
alpgarcia/grimoireELK/tree/marvel-enrich
valeriocos/perceval/tree/marvel-backend

@grimoirelab
@alpgarcia
@_valcos_

Bitergia