

Привіт!

```
gem "rutils"
gem "gilenson"
gem "russian"
gem "ru_propisju"
gem "timecode"
gem "edl"
```

"и это тоже".mb\_chars.upcase

```
gem "prorate"
gem "activerecord_autoreplica"
gem "format_parser"
gem "zip_tricks"
gem "apiculture"
gem "tracksperanto"
```

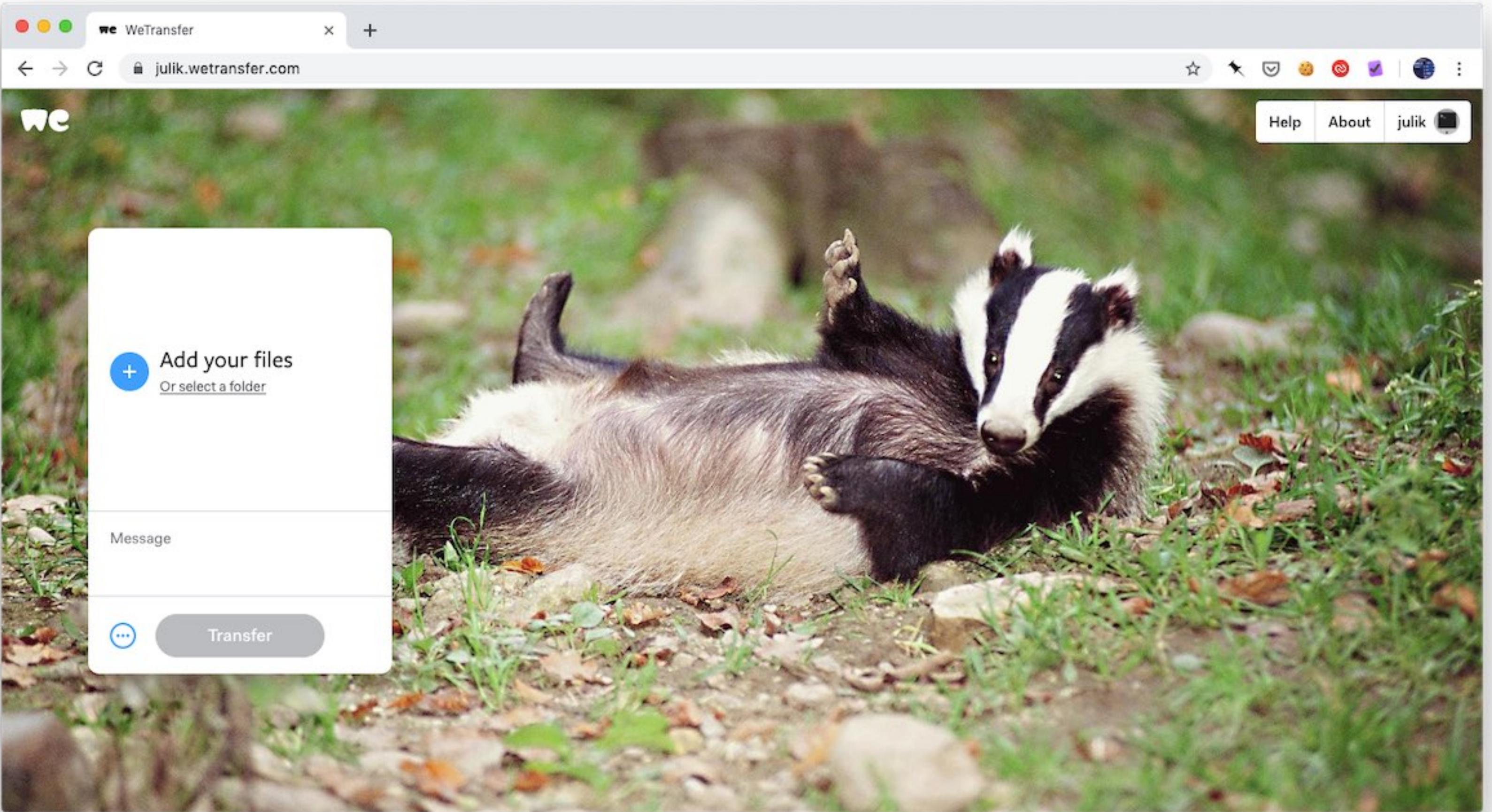
# WeTransfer

## Tools to move ideas



# WeTransfer

Много файлов, много клиентов, много Ruby и JS



Амстердам

Х

Х

Х

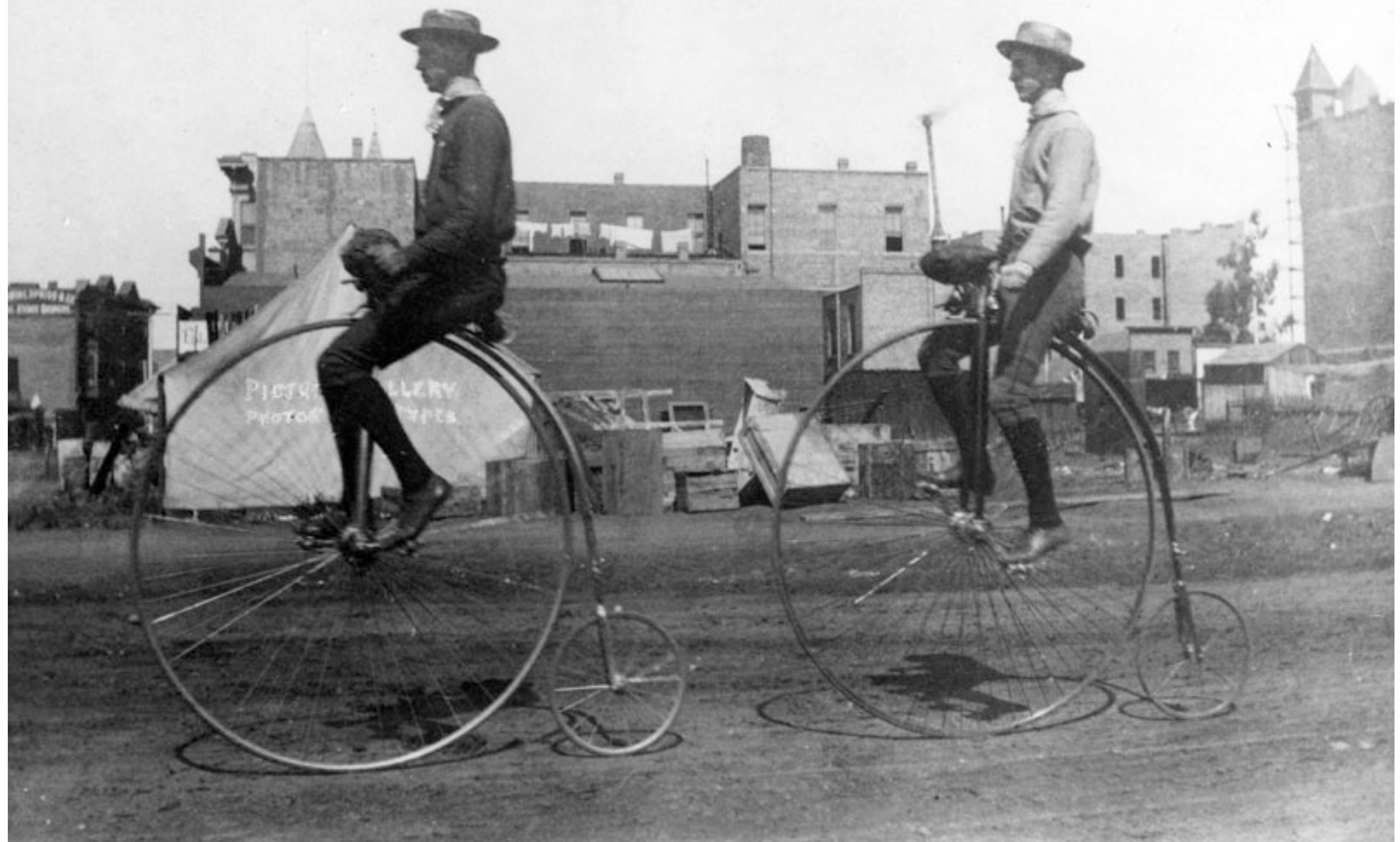


- ≈ 881000 велосипедов - **правда-правда!**
- на ≈ 800000 жителей



Draisienne,

Cette Voiture, par Brevet d'invention, pour faire 14 Lieues en 15 jours.









# Основная модель для местности

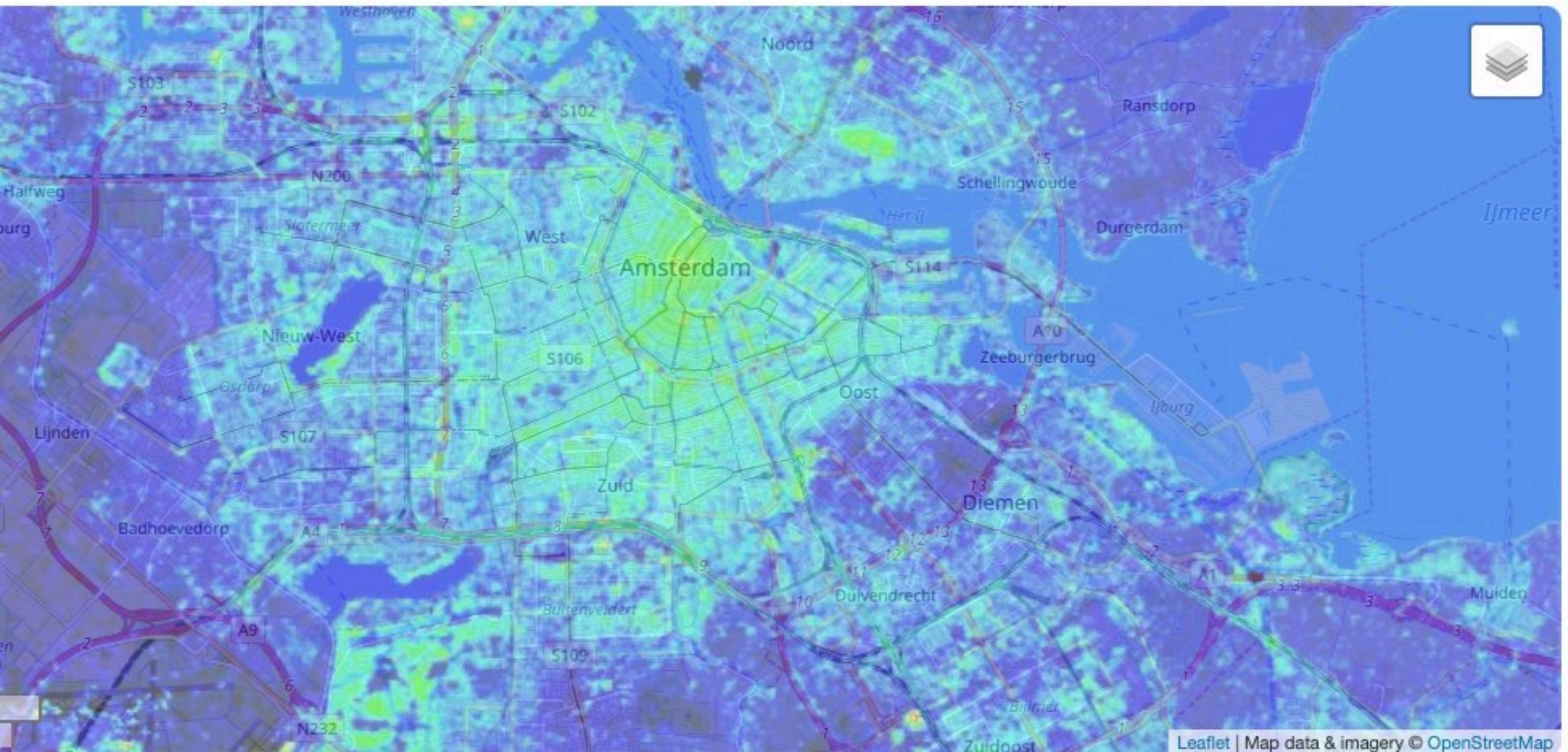
- Один тормоз - педальный задний. Передний ручной - скорее баловство
- Навесное оборудование - звонок (иногда, пугать туристов) и свет (без него оштрафуют)
- Иногда багажники

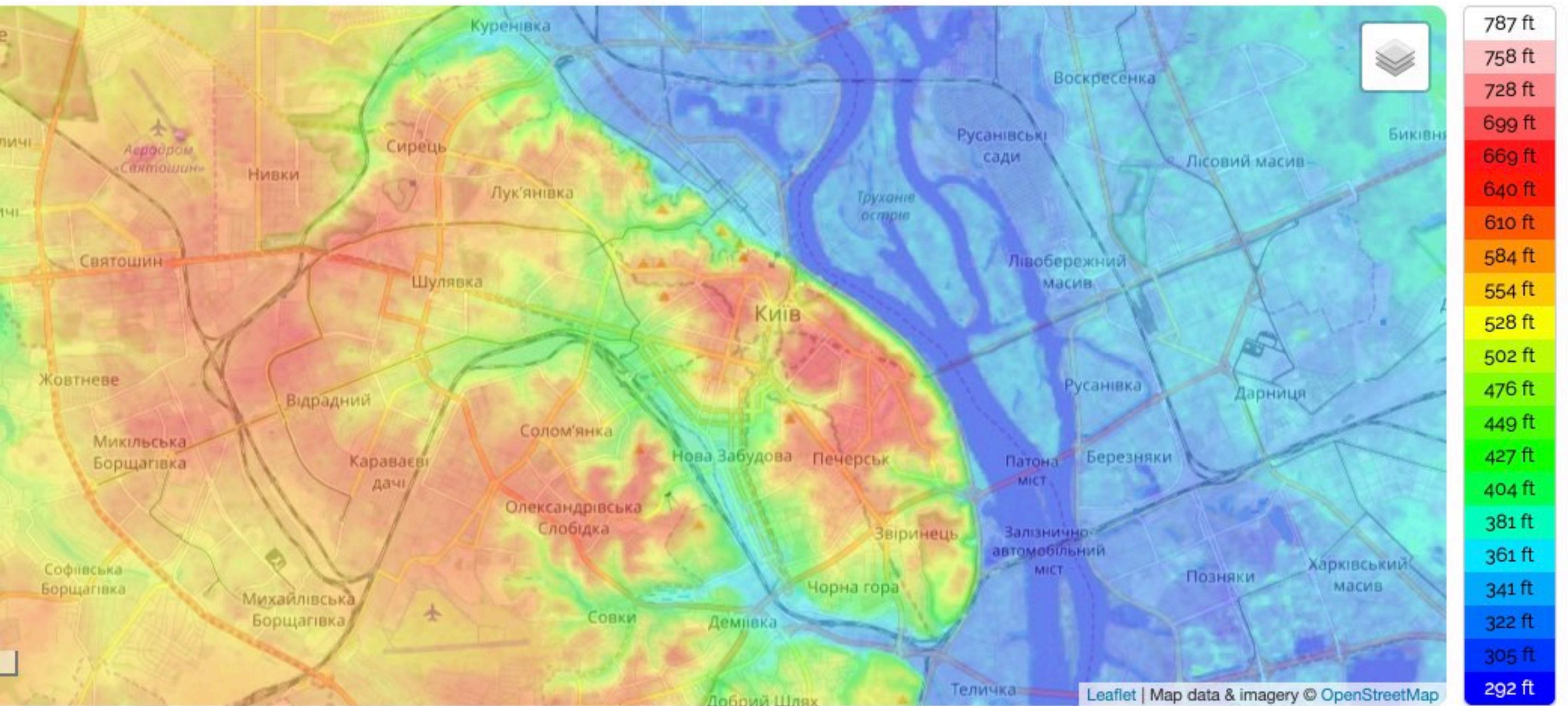
# И нету

- Ручных тормозов
- Сликов
- Клипс
- Амортизаторов
- Переключения скоростей
- Сумок и бардачков





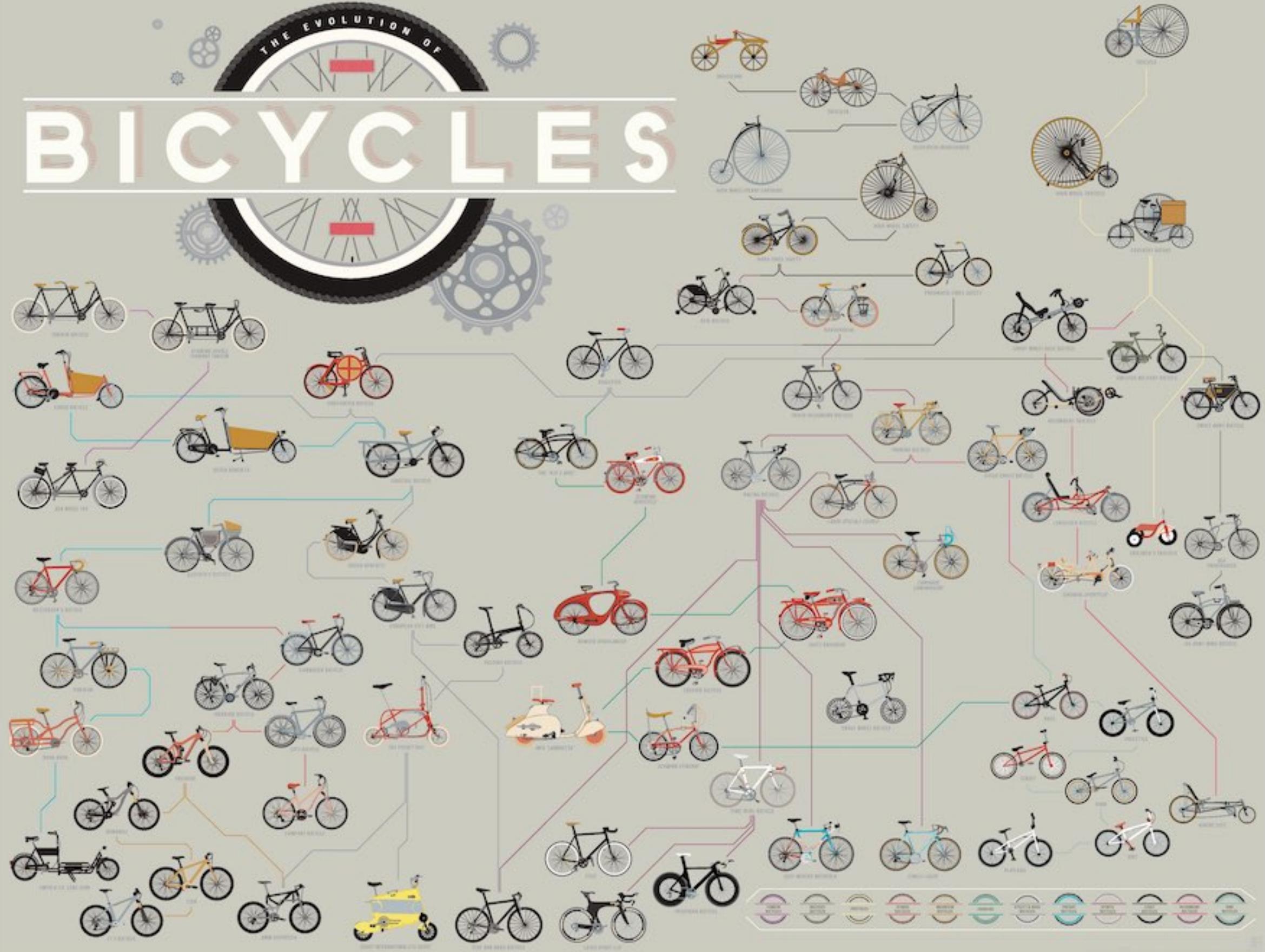








# THE EVOLUTION OF BICYCLES



# Это не изобретение

- Это решение *разных* задач
- Это исследование
- Это поиск и творчество
- Это доводка того что вам не подходит до того состояния, в котором оно вам подойдет

# Кейс первый. zip\_tricks

Как-бы "еще одна" ZIP-библиотека для Ruby.

 Used by ▾

492k

 Watch ▾

44

 Star

1.1k

 Fork

258

 Used by ▾

48

 Unwatch ▾

24

 Star

91

 Fork

17

# Зачем это было нужно

У нас есть сервер для скачивания файлов. Он тащит данные с S3 и подшивает их на ходу в ZIP. Ну примерно как mod\_zip для nginx. Но нам еще надо отправлять оповещение о том что скачивание закончено. А еще...

- Какого размера будет ZIP после скачки?
- А рестартовать скачку с определенного места?
- А Zip64?
- А юникодные имена файлов?

Например, хочется вот так:

```
exact_archive_size = ArchiveLibrary.estimate_size do |arch|
  arch.add_file(name: "File.doc", size_in_archive: 1586215)
  arch.add_file(name: "TOC.xml", size_in_archive: 1452)
end
```

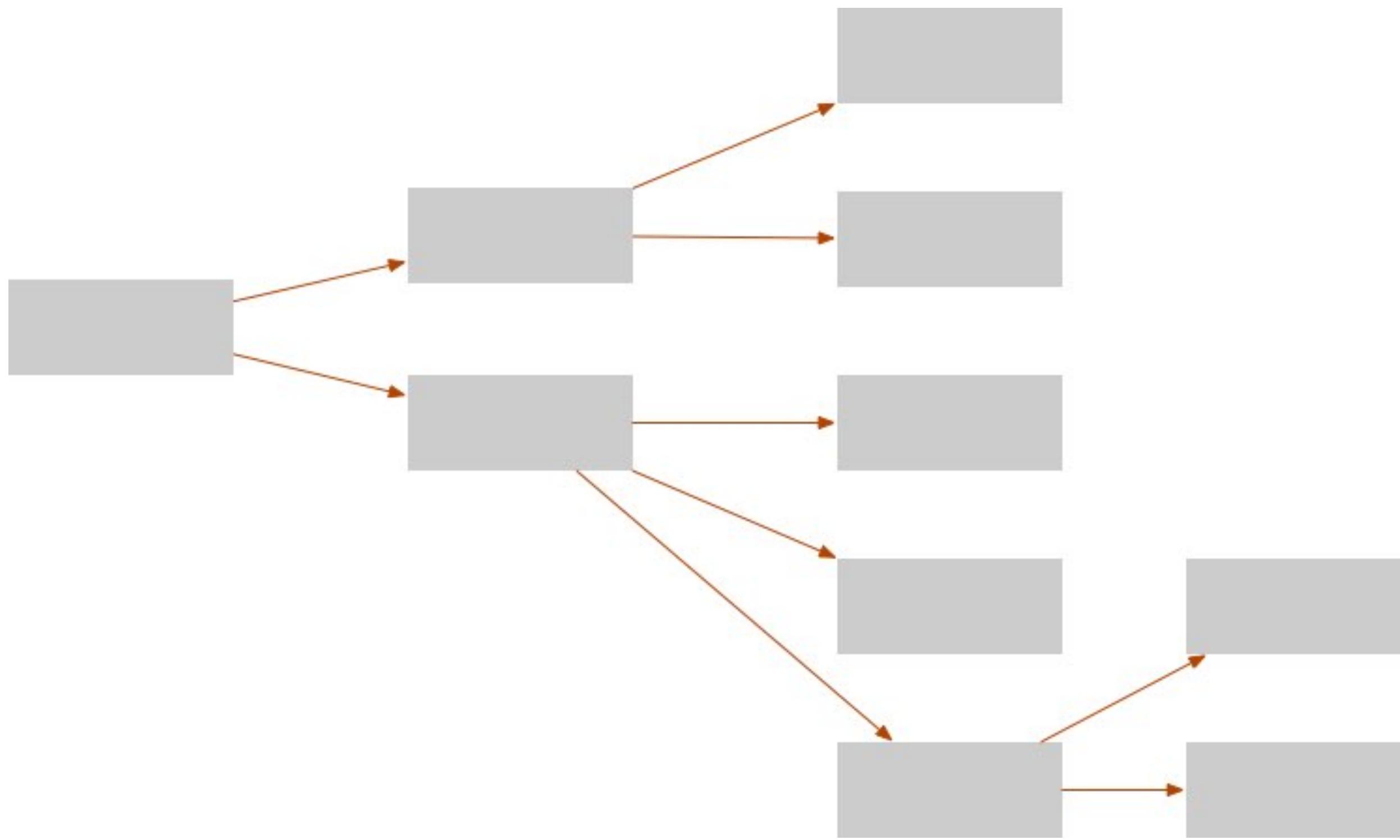
# А самое главное - хочется вот так:

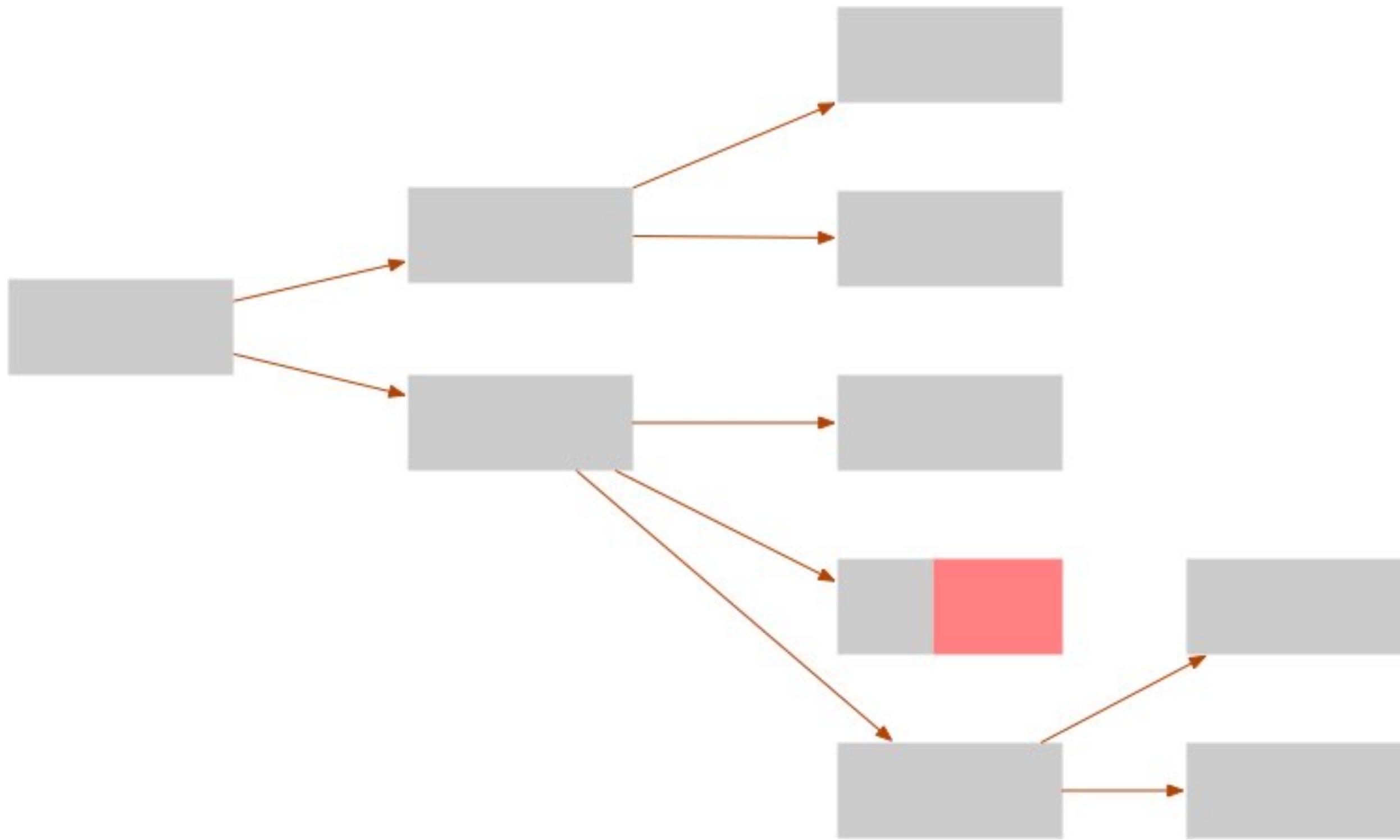
```
ArchiveLibrary.write(socket) do |arch|
  arch.add_local_file_header(name: "File.doc", size_in_archive: 1586215)
  IO.copy_stream(remote_file_contents, arch)
end
```

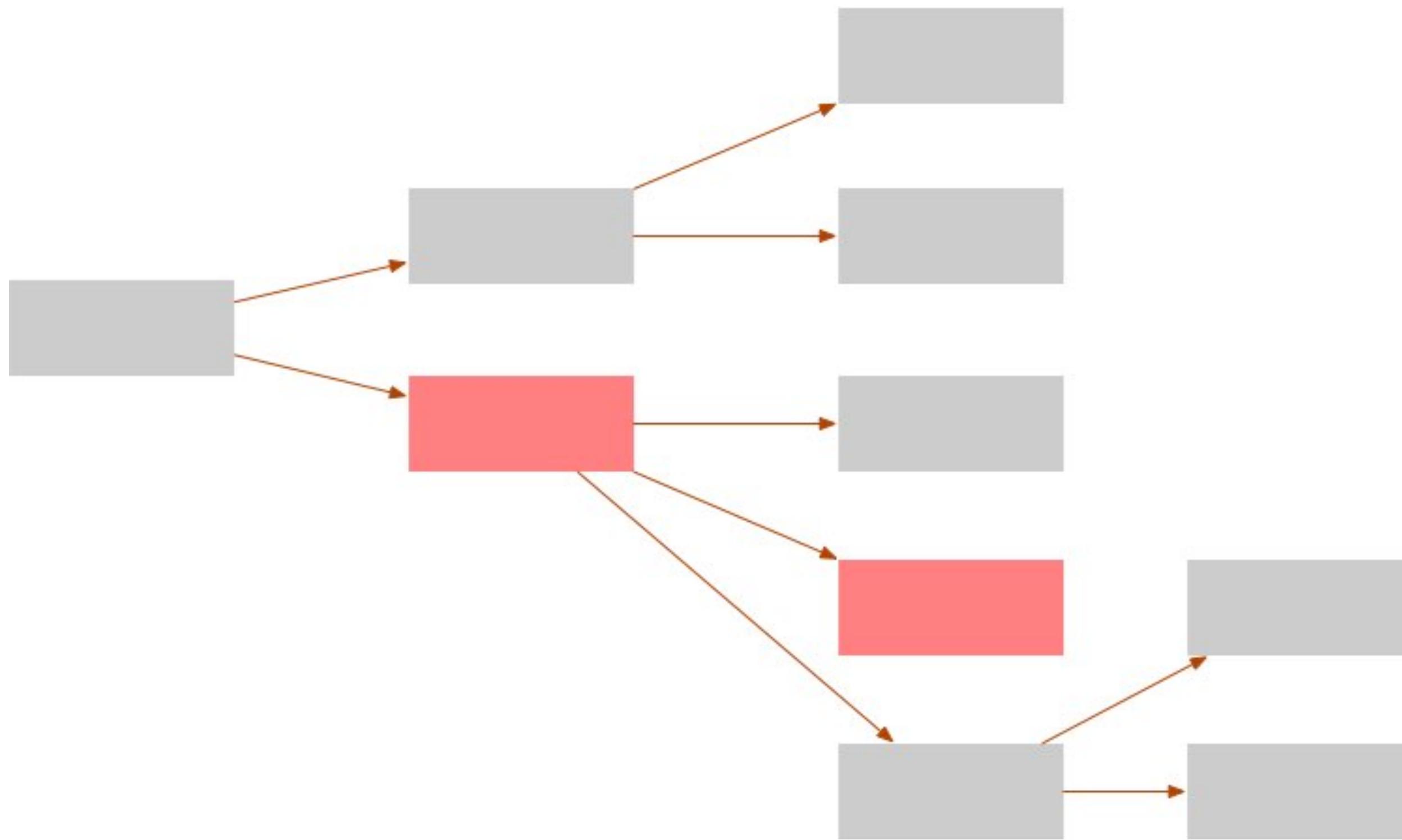
socket не может seek и rewind

То есть - интеграция чужой библиотеки  
нам видится так:

(потому что ООП и модули и кумбайя)







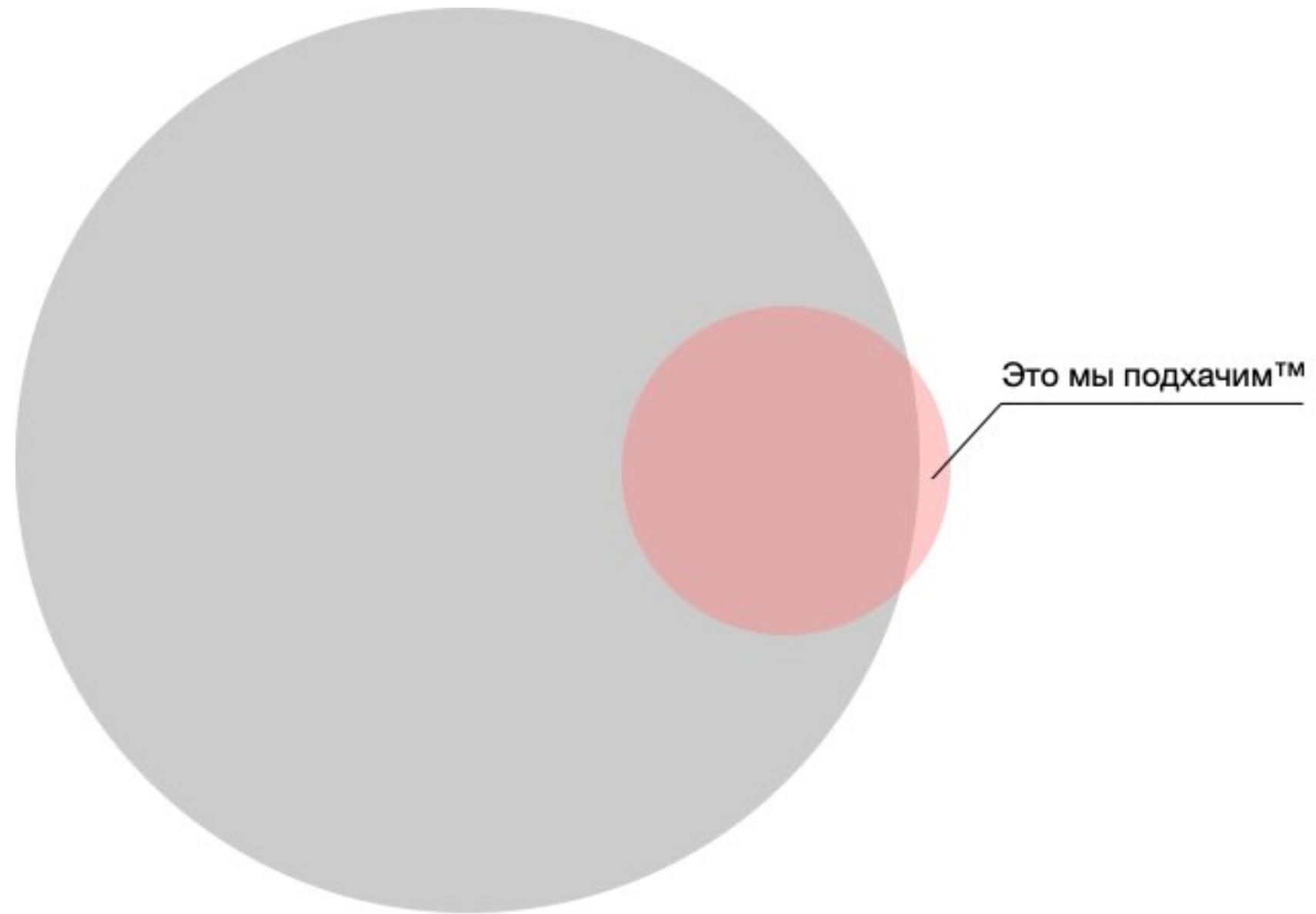


Рис. 1 - Интеграция “по книжке”

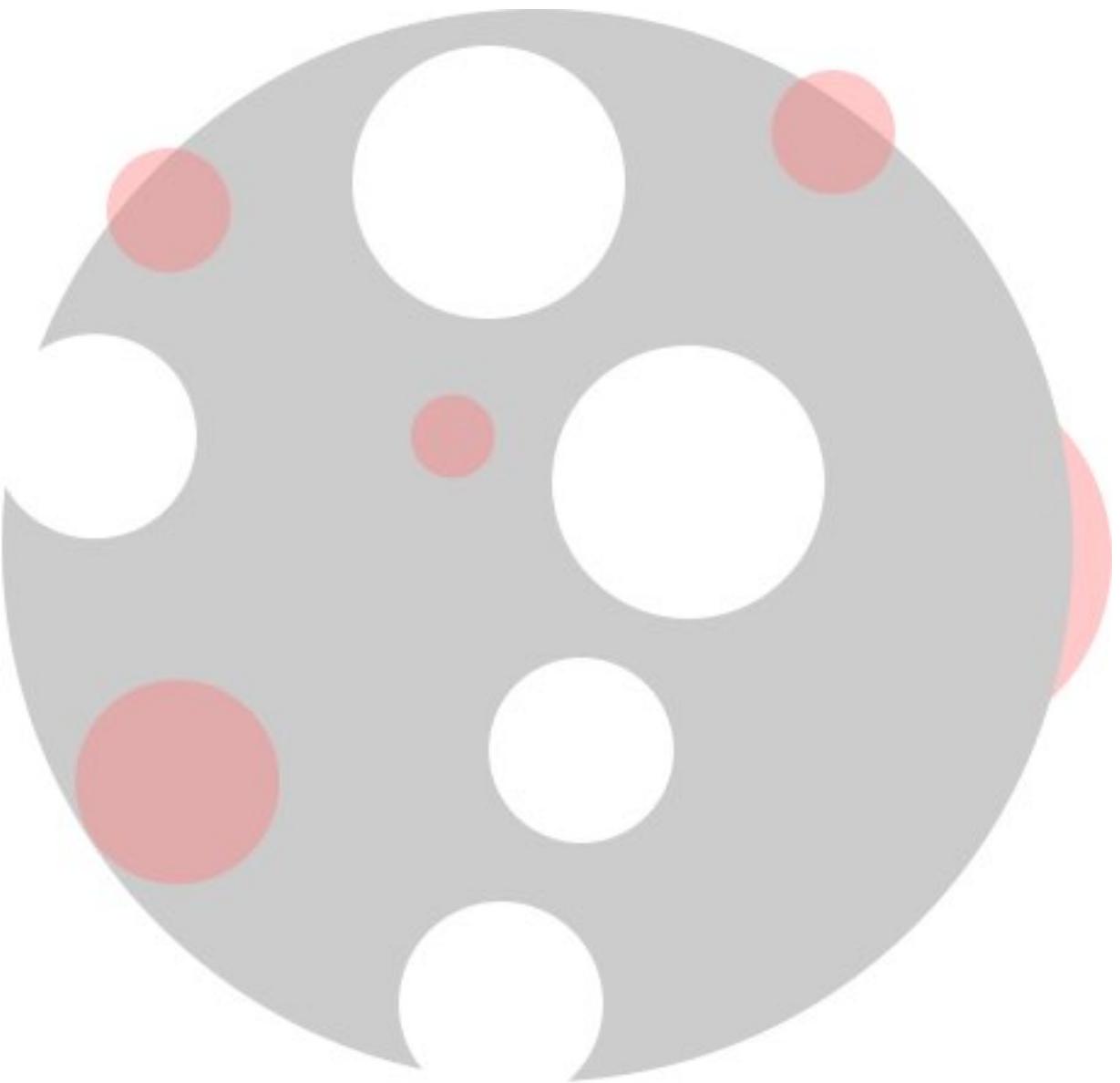
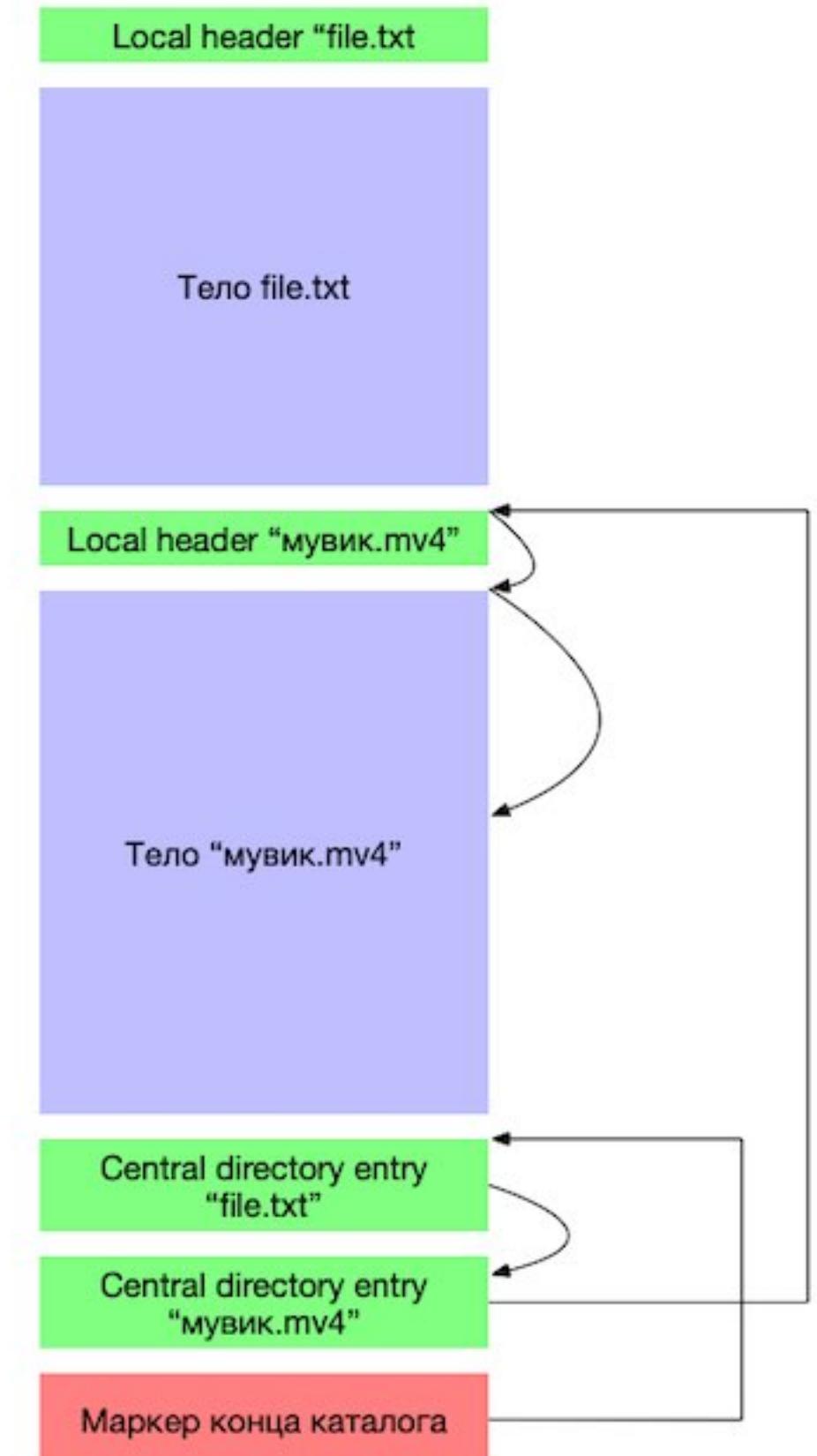
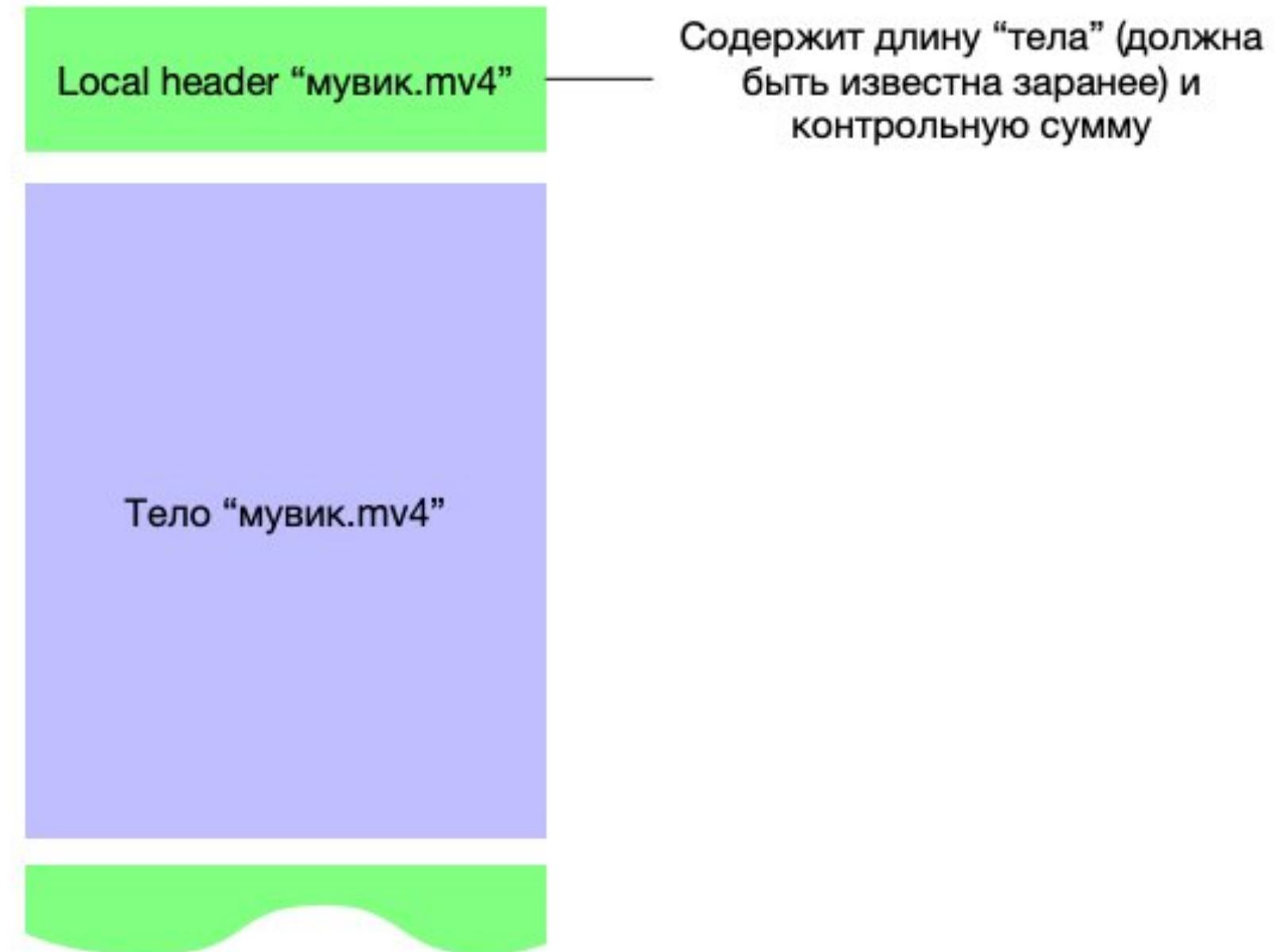


Рис. 2 - Как оно на самом деле

Вспоминаем ZIP формат







Range: bytes=4589-9862

На самом деле zip\_tricks начался как патч для rubyzip. И ох - если бы можно было его таким оставить.

```
class ZipTricks::OutputStreamPrefab < ::Zip::OutputStream
  def initialize(io, stream_flag=false, encrypter=nil)
    super StringIO.new, stream_flag=true, encrypter=nil
    @output_stream = io
  end
  def put_next_entry(entry_name, size, crc)
    new_entry = ::Zip::Entry.new(@file_name, entry_name)
    new_entry.compression_method = Zip::Entry::STORED
    new_entry.crc, new_entry.size = crc, size
    new_entry.compressed_size = size + new_entry.calculate_local_header_size
    super(new_entry, nil, nil, new_entry.compression_method, level=Zlib::NO_COMPRESSION)
  end

  # We never need to update local headers, because we set all the data in the header ahead of time.
  # And this is the method that tries to rewind the IO, so fuse it out and turn it into a no-op.
  def update_local_headers
    nil
  end
end
```

Есть вариации похожего (библиотека zipline):

```
def finalize_current_entry
  if current_entry
    entry = current_entry
    super
    write_local_footer(entry)
  end
end

def write_local_footer(entry)
  @output_stream << [ 0x08074b50, entry.crc, entry.compressed_size, entry.size].pack('VVV')
end
```





# ТУПИК.

```
module Zip
  # placeholder to reserve space for a Zip64 extra information record, for the
  # local file header only, that we won't know if we'll need until after
  # we write the file data
  class ExtraField::Zip64Placeholder < ExtraField::Generic
    HEADER_ID = ['9999'].pack('H*') # this ID is used by other libraries such as .NET's Ionic.zip
    register_map

    def initialize(_binstr = nil); end

    def pack_for_local
      "\x00" * 16
    end
  end
end
```

# Ну и соответственно

- ZIP - хитрозаковыристый формат
- В rubyzip реализация этого формата размазана тонким слоем по десятку-двум отдельных модулей



Как это можно сделать по-  
другому?

- Находим все мелкие детальки спецификации формата ZIP
- Собираем их в один объект с visitor pattern
- Люто тестируем во всех направлениях, и только его, побайтово

```
writer.write_local_file_header(io:, filename:, compressed_size:,
    uncompressed_size:, crc32:, gp_flags:, mtime:, storage_mode:)
```

```
writer.write_central_directory_file_header(io:, local_file_header_location:,
    gp_flags:, storage_mode:, compressed_size:, uncompressed_size:, mtime:, crc32:, filename:)
```

```
writer.write_data_descriptor(io:, compressed_size:, uncompressed_size:, crc32:)
```

```
writer.write_end_of_central_directory(io:, start_of_central_directory_location:,
    central_directory_size:, num_files_in_archive:, comment: ZIP_TRICKS_COMMENT)
```

- 430 строк с обильными комментариями
- EFS (UTF-8 имена файлов) автоматом и только если надо
- Zip64 автоматом и только если надо
- Data descriptor (футеры для записи CRC32 и размеров "после" файла)
- Keyword arguments для всего
- Внятная "простыня" вместо равиoli

# И еще всякие плюшки

- Чтение ZIP-файлов по HTTP с доступом к отдельным файлам в архиве
- Size estimator (потому что можно создать "виртуальный" зип без файлов)
- Параллельная компрессия на нескольких серверах сразу
- YARD для всего

# Наблюдения

- Начинать всегда с доработки напильником и изучения предмета
- Отслеживаем, когда клеммы отваливаются и шунты горят.
- "Еще одна" библиотека это нормально. Даже в cargo библиотек для zip две.

# Нету социального протокола

Для ситуации "Все это хорошо бы переделать, давайте  
придумаем как."

Или?

# Кейс 2 : LaunchDarkly

LaunchDarkly держит на своих серверах таблицу features и таблицу АВ-экспериментов. Библиотека внутри вашего приложения скачивает их периодически, и позволяет распределять посетителей в определенные "корзины" (функциональность включена/выключена и так далее). LaunchDarkly предоставляет `get` которым можно пользоваться.

Ничто не предвещало беды.

При этом API который "свисает наружу" предельно маленький:

```
# в конфиге приложения
ld_client = LaunchDarkly::LDClient.new("YOUR_SDK_KEY")

# и далее где надо
show_feature = ld_client.variation("your.flag.key", {key: "user@test.com"}, false)
if show_feature
    #
end
```

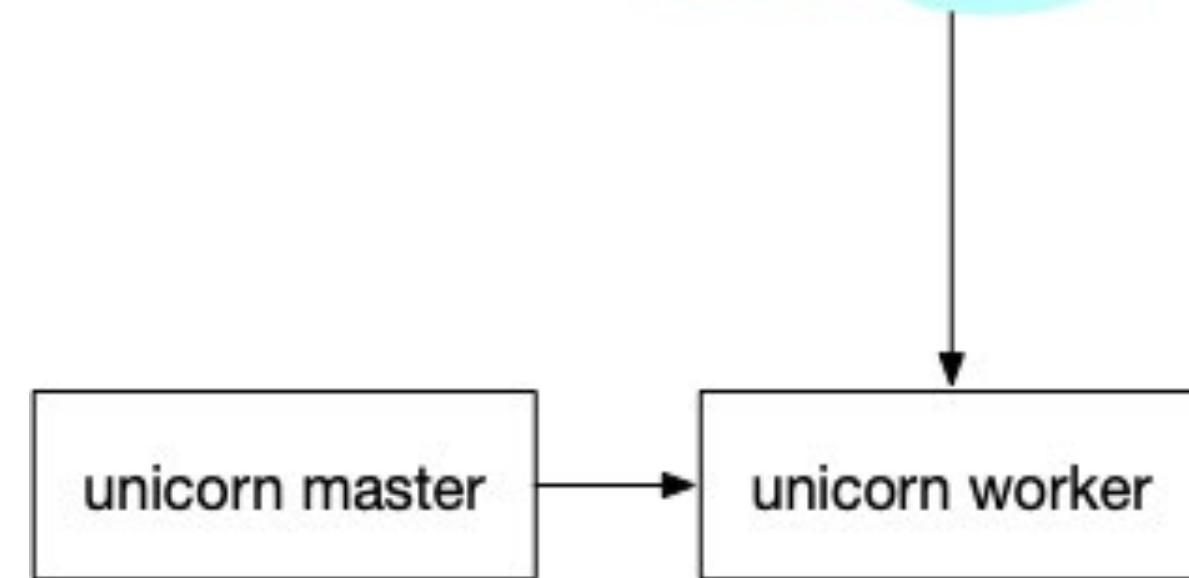
Стандартный способ получения таблицы - простой HTTP-запрос. Все ок.

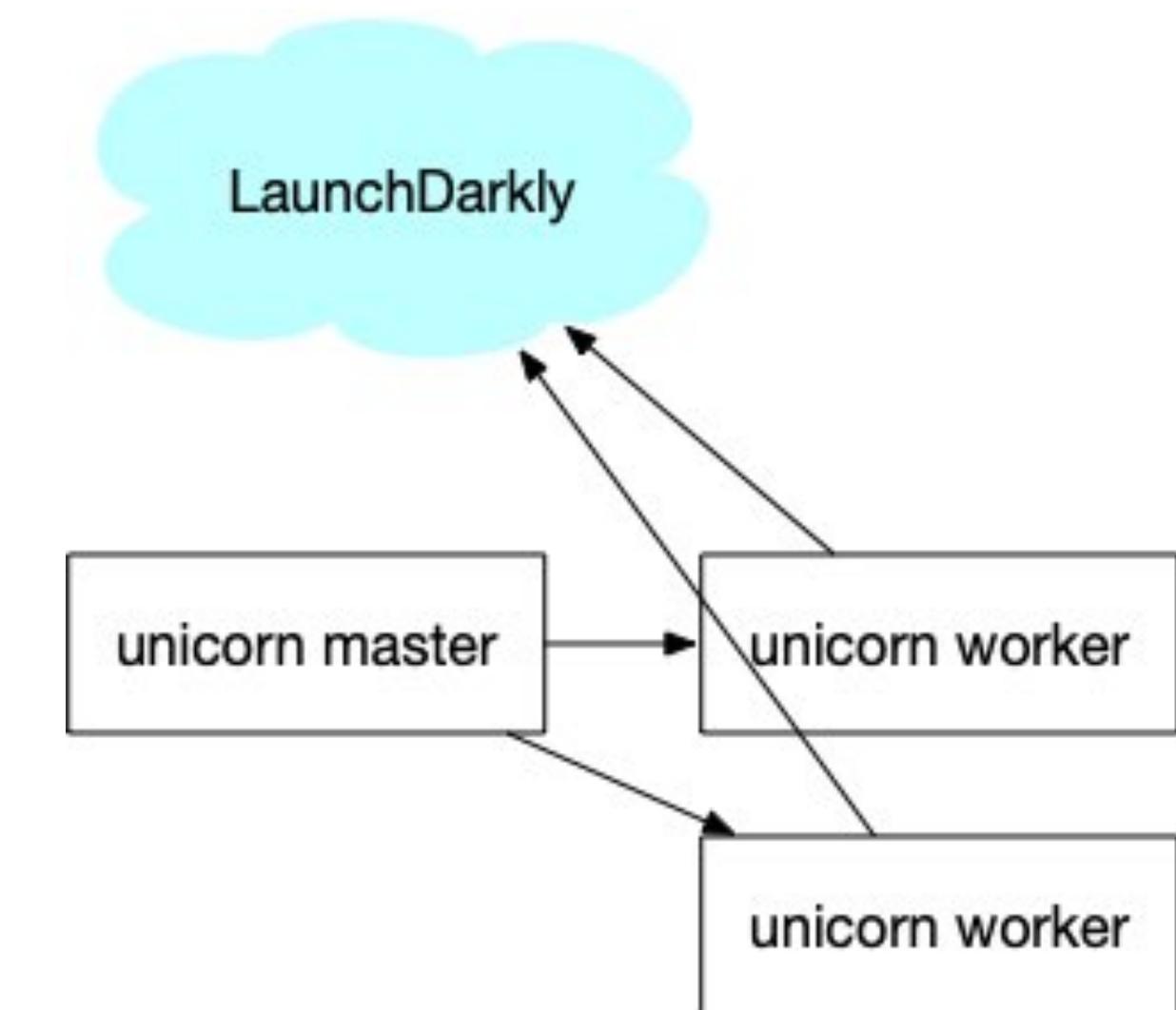
- А еще они открывают SSE-соединение (EventSource) со своим сервером. Из каждого руби-процесса. Чтобы получать апдейты этой таблицы с задержкой меньше секунды.
- А еще они генерят события при просчете флага и отправляют эти события на свой сервер.

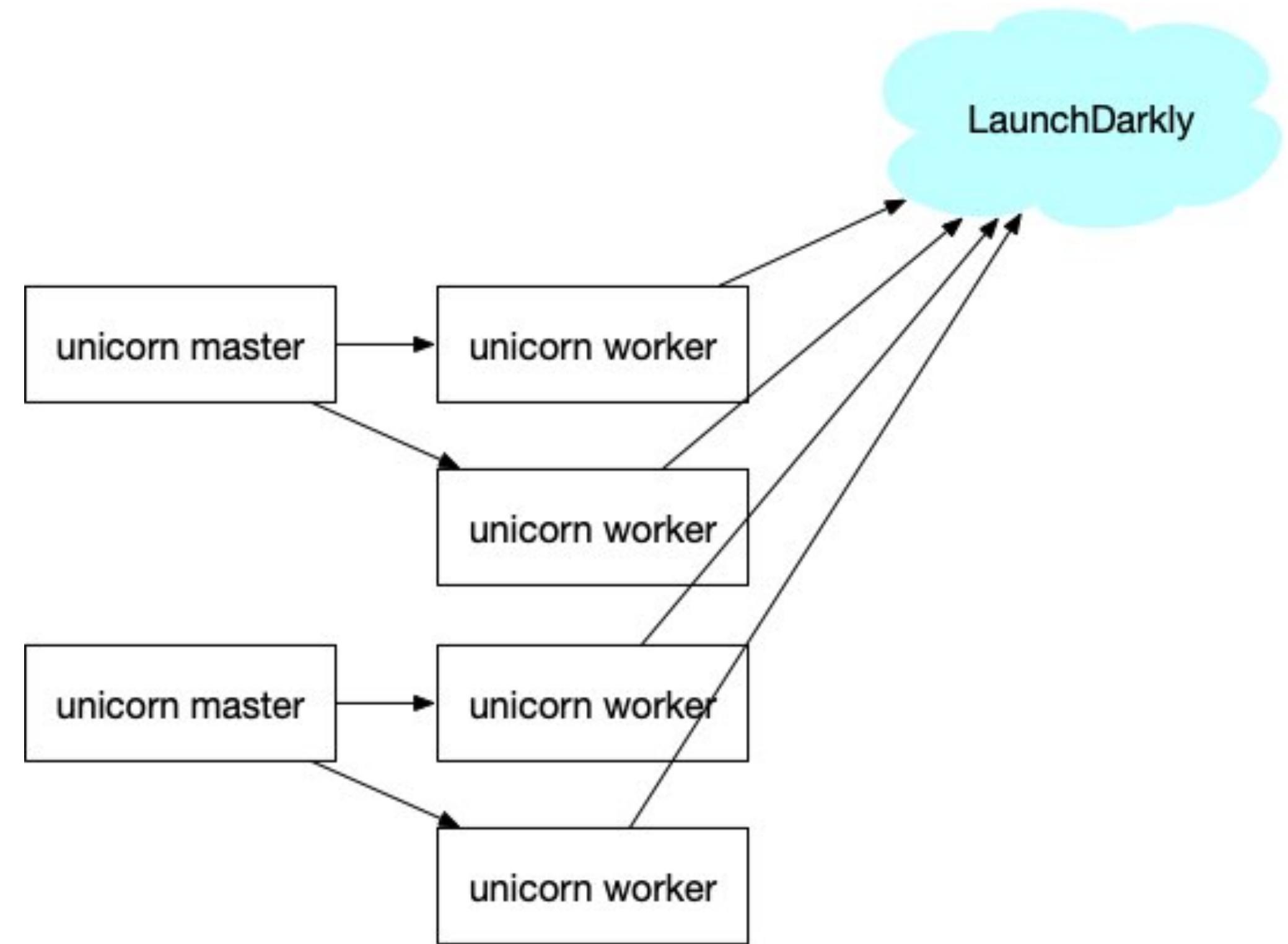
Каждый процесс периодически выполняет два HTTP запроса, и держит постоянно открытым HTTP соединение.

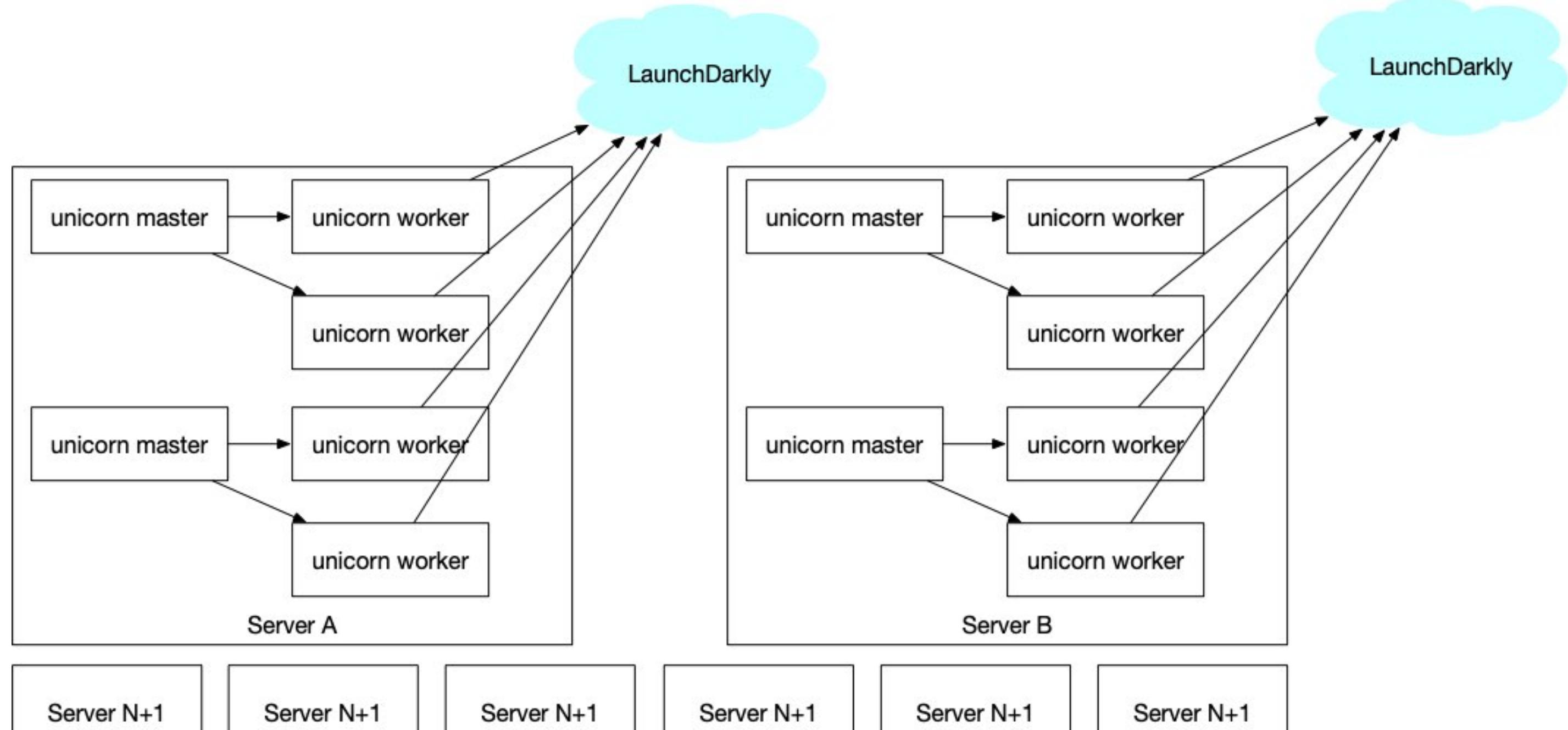


LaunchDarkly









Ничего что у нас теперь будет 200  
постоянно открытых сокетов в  
интернет чтобы у нас были  
feature flags? С каждого сервера?

Но дело даже не в этом. А в том что для того чтобы это обеспечить, в самой библиотеке должен быть либо thread, либо event-реактор.

Повторяем. Чтобы сделать вот это:

```
rng = Random.new(derive_numeric_seed(user_id))
rng.rand < rollout_ratio
```

```
spec.add_runtime_dependency "json", "~> 1.8"
spec.add_runtime_dependency "faraday", "~> 0.9"
spec.add_runtime_dependency "faraday-http-cache", "~> 1.3.0"
spec.add_runtime_dependency "thread_safe", "~> 0.3"
spec.add_runtime_dependency "net-http-persistent", "~> 2.9"
spec.add_runtime_dependency "concurrent-ruby", "~> 1.0.4"
spec.add_runtime_dependency "hashdiff", "~> 0.2"
spec.add_runtime_dependency "ld-celluloid-eventsource", "~> 0.10.0"
spec.add_runtime_dependency "celluloid", "~> 0.18.0.pre" # transitive dep; specified here for more control

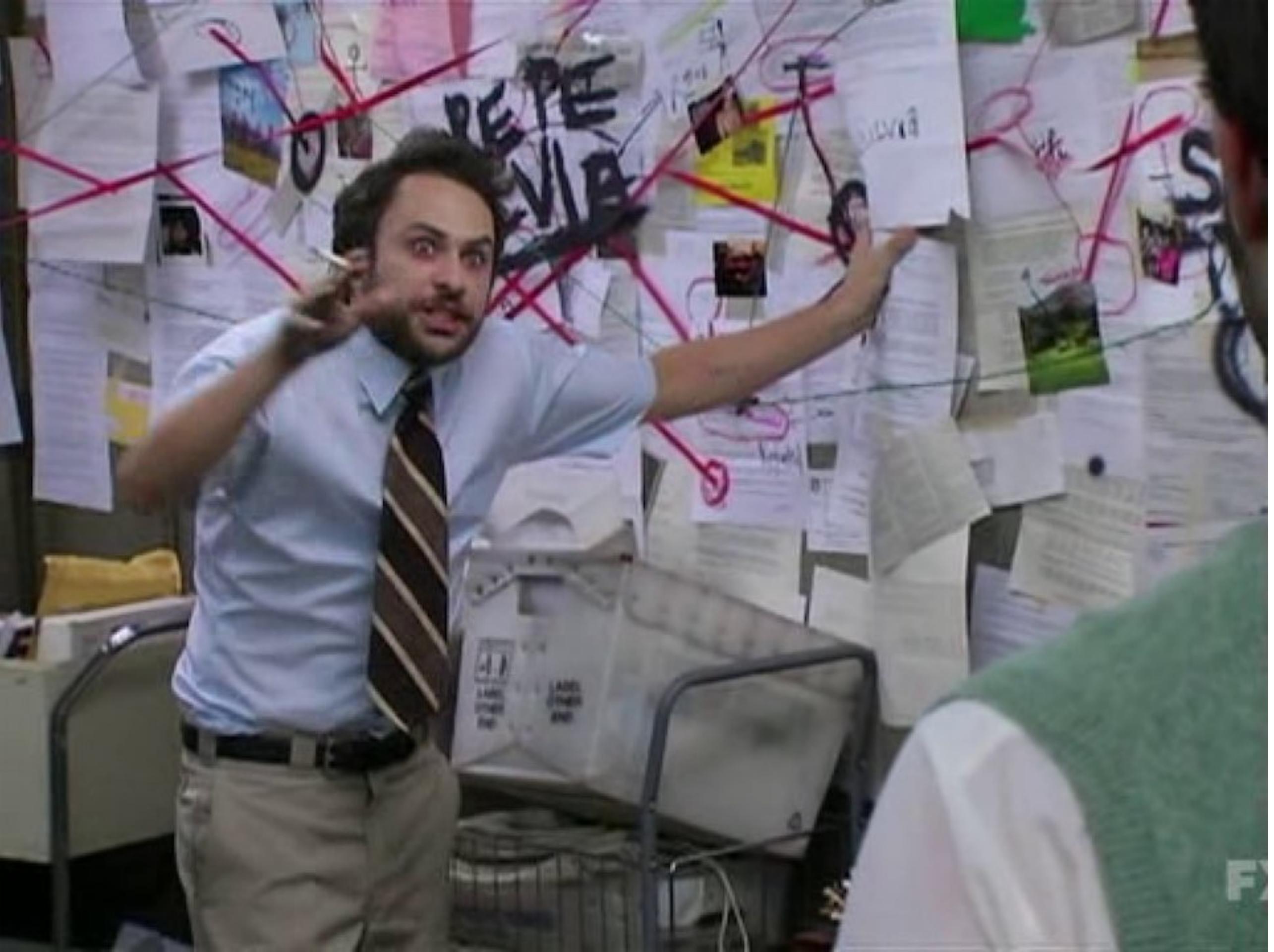
if RUBY_VERSION >= "2.2.2"
  spec.add_runtime_dependency "nio4r", "< 3" # for maximum ruby version compatibility.
else
  spec.add_runtime_dependency "nio4r", "~> 1.1" # for maximum ruby version compatibility.
end

spec.add_runtime_dependency "waitutil", "0.2"
```

То есть у нас будет и concurrent-ruby (потоки) и celluloid (event loop с актерами), который занимается этим самым EventSource

Желание иметь и Celluloid и потоки в какой-то момент привело к тому что LD просто был несовместим с нашей версией Ruby.

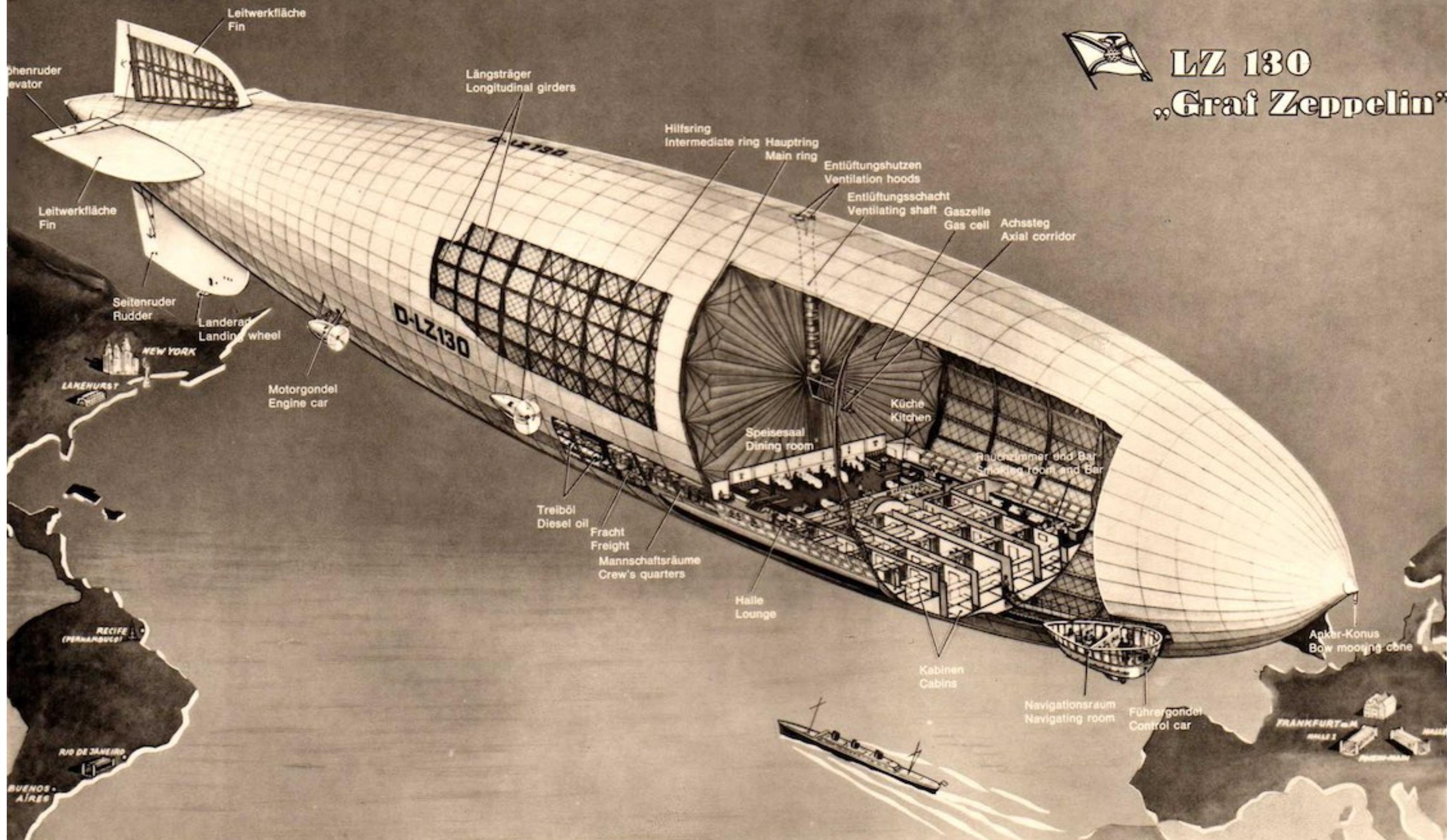
Эвенты которые создаются при просчете флагов запускаются в Queue, и таск работающий в фоне (для этого им нужен concurrent-ruby) их оттуда забирает, трансформирует, агрегирует и отправляет...





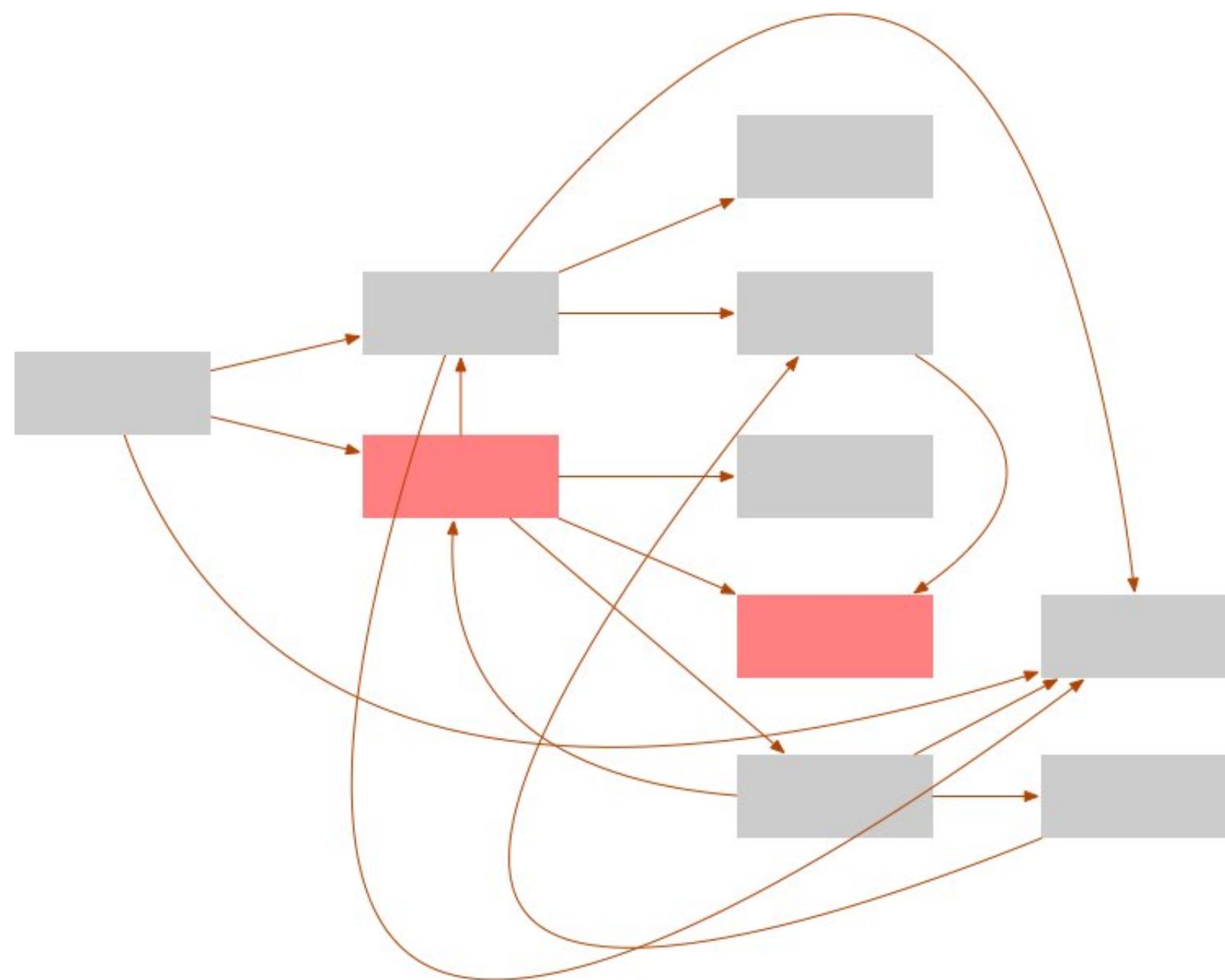


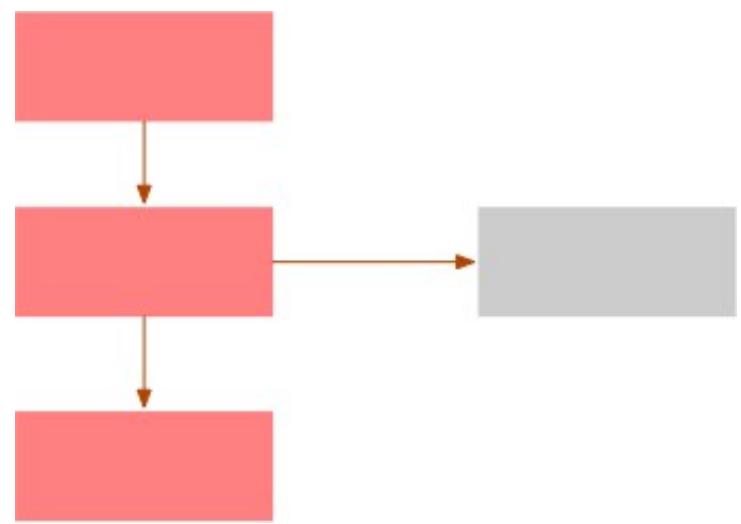
# LZ 130 „Graf Zeppelin“



Предлагается правда прокси-сервер на Go чтобы ваше  
приложение подсоединялось к нему, а не к серверу LD в  
Америке.

- Puma / unicorn
- Sidekiq worker
- Faye / ActionCable





- Никаких синглтонов
- Никаких потоков
- Очередь с событиями для отправки - в Redis
- Флаги тоже в Redis

```
# Ставим его в приложение
config.middleware.use "Lad::RackMiddleware", redis: config.redis_connection_pool

# и используем из контроллера
def ld_client
  request.env['lad.client']
end
```

# И в фоновом режиме

На одном сервере два cron job'a

- "Забрать эвенты из Redis и отправить их в LaunchDarkly"
- "Загрузить флаги из LaunchDarkly в Redis"

Крутятся раз в пять секунд.

# А что дальше?

У нас теперь есть свой велосипед, который нужно патчить, поддерживать и всячески смазывать и перебирать. А upstream на месте не стоит.

# Все интересно

- Celluloid они оттуда выпилили и заменили на socketry/async
- Concurrent-ruby еще не выпилили

# Socketry/async тоже непрост

Мы с ним наелись в другом приложении.

- Иногда подтекает память
- Текут сокеты
- SSL-соединения виснут
- Библиотеки обновляются раз в неделю

# Но есть подстраховка

```
show_feature = ld_client.variation("your.flag.key", {key: "user@test.com"}, false)
if show_feature
    #
end
```

Социальный протокол пришлось  
придумать на ходу

- LD - коммерческая организация, и у их клиентов другие заботы
- Они предоставляют Ruby-библиотеку и уже это подвиг
- У нас не цель показать что писать Java на Ruby не надо. У нас цель - иметь работающие feature flags

Мы решили наши трейдоффы - а также мои личные взгляды на то как надо (и не надо) писать библиотеки - из избы не выносить. Потому что отношения с поставщиками / партнерами важнее.

# И это нормально!

Не все надо опенсорсить, особенно если это может вызвать скандалы. Но можно рассказать об этом историю. Например - всем вам 😎

# Наблюдения

- Начинать всегда с доработки напильником и изучения предмета
- Отслеживаем, когда клеммы отваливаются и шунты горят (у нас перегорело все до старта), и только потом...
- "Еще одна" библиотека это нормально, но открыто поливать оригинал не надо.

← Tweet



Андрей Ситник  
@andrey\_sitnik

Зачем крайности? Почему не бороться с критиканством в обществе (когда комментарий не помогает автору) оставляя критику?

Увы, от критиканства страдают все. Авторы просто не выдерживают поток неконструктивного негатива и перестают выкладывать работы.

[Translate Tweet](#)



Ivan @zenkov · Sep 24

Replying to @andrey\_sitnik

По мне так лучше самая всратая критика до соплей хейта, чем сидеть в плену единомышленников и обмазываться своим творчеством допуская лишь конструктивную критику и по делу (то есть цензурировать её), но это конечно каждый сам для себя решит

1:06 PM · Sep 24, 2019 · [Twitter for Android](#)

2 Retweets 25 Likes





Roman Mints

4 uur ·

...

Сначала роди. Вот второго сначала роди, потом поговорим. Ты сначала двух одновременно роди. Сначала воспитай, сначала вырасти. Сначала поучись. Поживи с моё. Поработай от звонка до звонка. Сначала стань кем-то. Сначала сними кино, сначала сам книгу напиши, сначала сайт построй, сначала высшую школу экономики окончи, сначала школу дизайна пройди, сначала отслужи, сначала стань мужиком, побудь в моей шкуре, сначала похорони кого-нибудь, сначала песню напиши, как Лоза, сначала разговаривать научись, сначала хамить разучись, сначала на Луну слетай, сначала напиши три оперы, сначала заработай много денег, сначала качели зимой языком лизни, сначала упражнения Коргуева поиграй, сначала этюды Годовского выучи, сначала выиграй конкурс большого шлема, сначала страной поуправляй, сначала денег в хоспис перечисли, сначала отсиди, сначала сам гулаг построй, потом посмотрим, лучше ли у тебя получится. А ты кто такой вообще, чых будешь, под кем ходишь, ты кому это сказал, а ну иди сюда, а ну иди отсюда, нет, сюда иди, я сказал, чтобы я тебя больше здесь не гитлер.



Anton Karetnikov en 267 anderen 100 opmerkingen 15 keer gedeeld

# А у нас не все гладко

Очень много чего на Ruby в приличном варианте просто нет.

- Machine learning
- GPU
- Байндинги для Qt
- async/await
- UI
- Транспиляции в JS

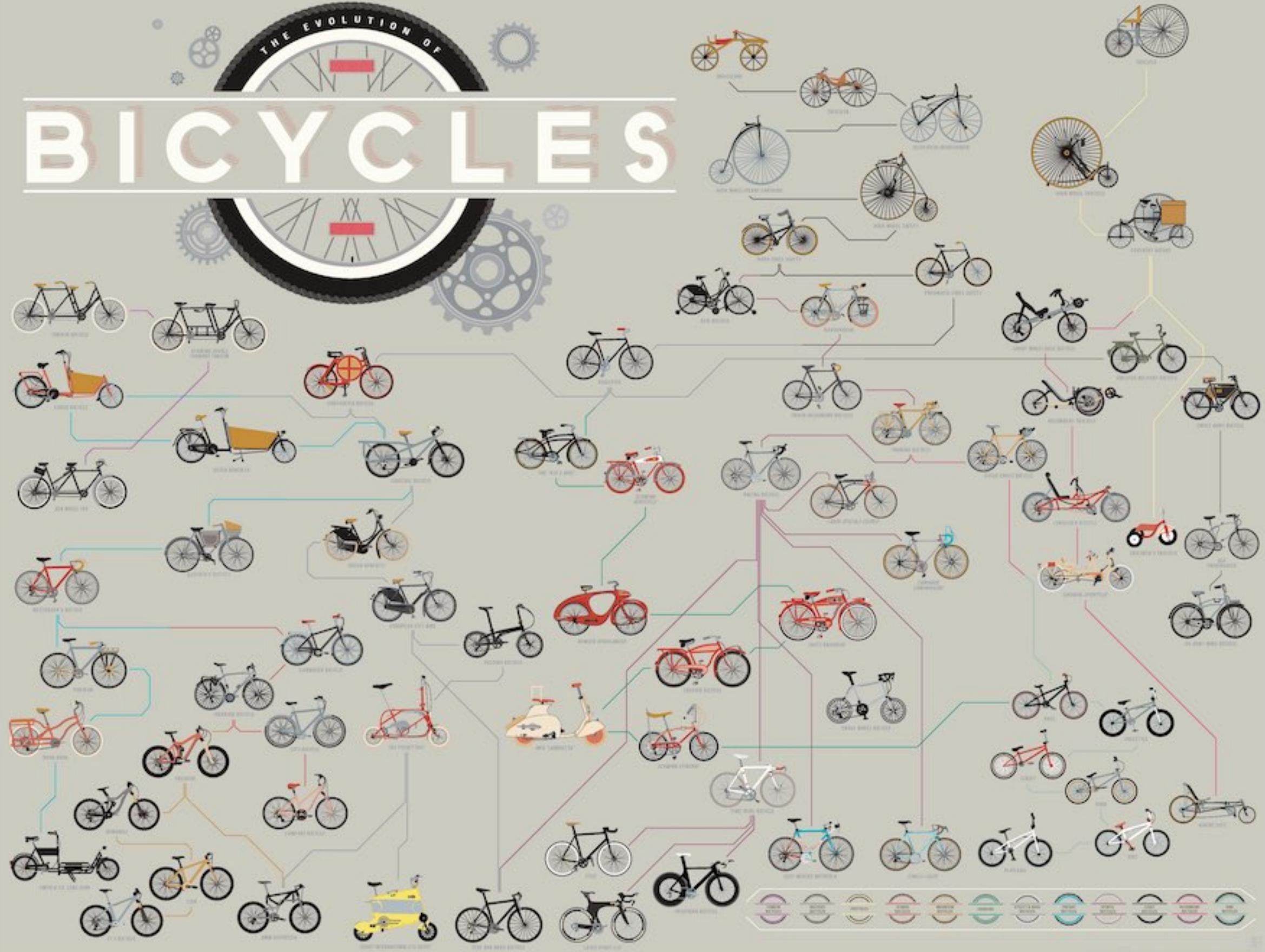
Каждый раз когда мы, улюлюкая, даем кому-то понять что он занят глупостями - мы теряем контрибьюторов. А язык и так усыхает и завядаeт.

Вы хотите продолжать писать на  
Ruby? Я - да.

Вместе с вами.



# THE EVOLUTION OF BICYCLES



У нас есть работа

<https://wetransfer.homerun.co>

# Изобретайте велосипеды и не стесняйтесь.

- <https://github.com/WeTransfer>
- <https://github.com/julik>
- [twitter / @julikt](#)