CodeBeam SF - 2018

BUILDING AND INTEGRATING A DATA PLATFORM

barrel

- benoît chesneau
- craftsman working on P2P and custom data endpoints solution
- **enki multimedia**: the corporate interface
- member of the **Erlang Industrial User Group**

about me

barrel

micromobile

services

barrel

1. Does my service do only one thing?
2. Is my service autonomous?
3. Does this service own its own data?

a good micro-service?

barrel

- isolated
- own its own data
- resilient
- communicate with other by asynchronous messages

micro-servcice

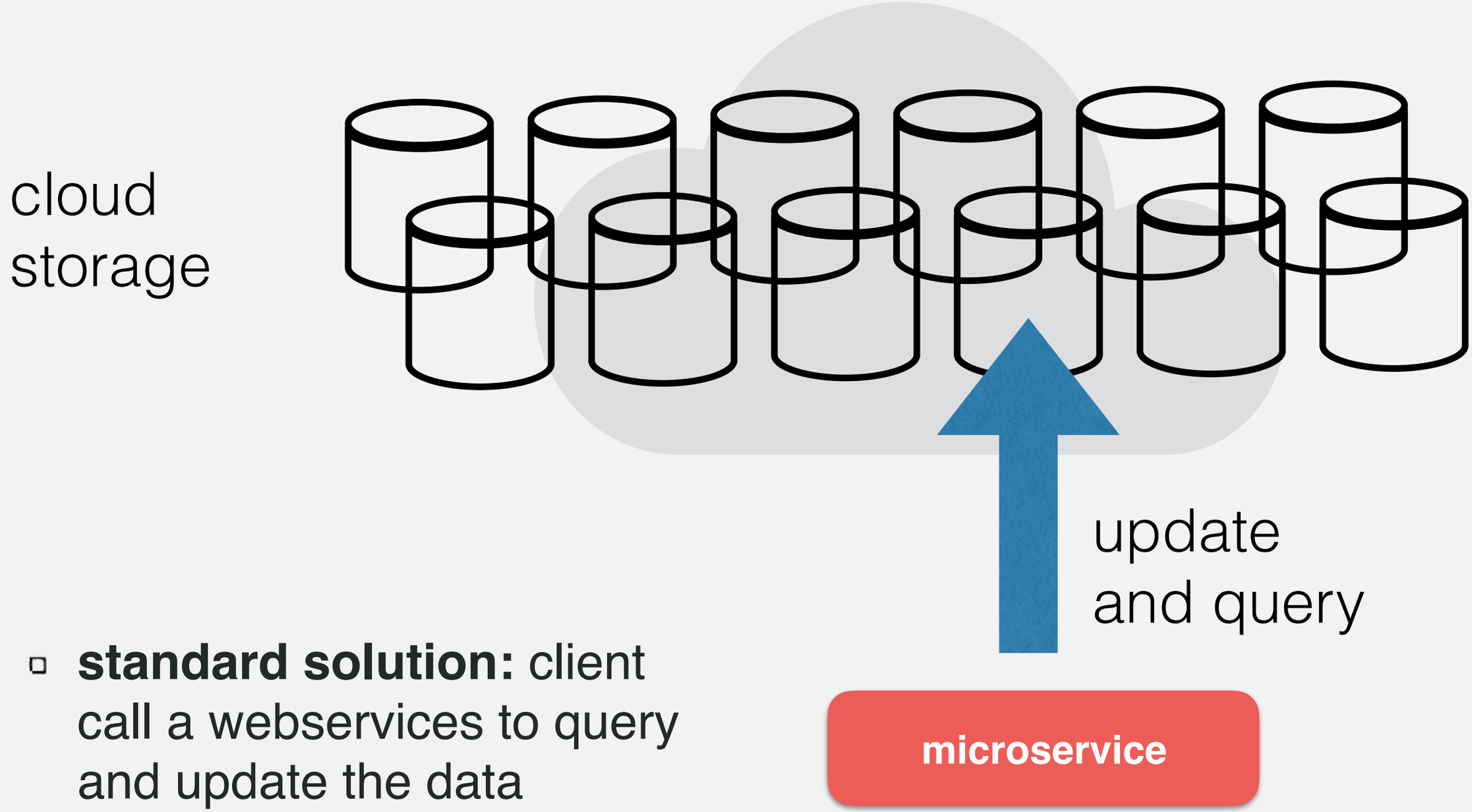
barrel

- sharing data in the mobile age between and across micro-services make applications more scalable and resilient

- Ex: messaging systems,

sharing data

barrel

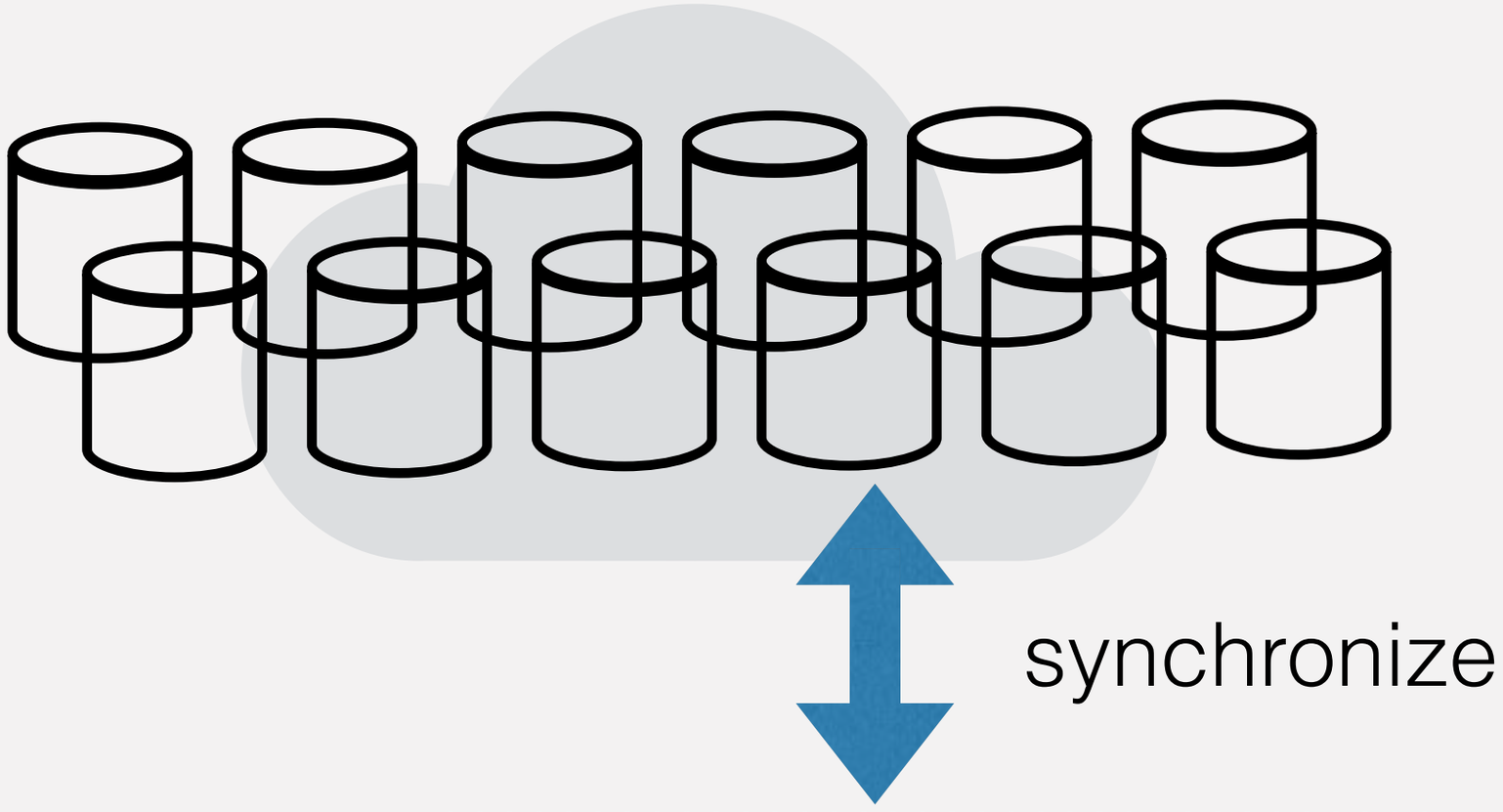


- **standard solution:** client call a webservices to query and update the data

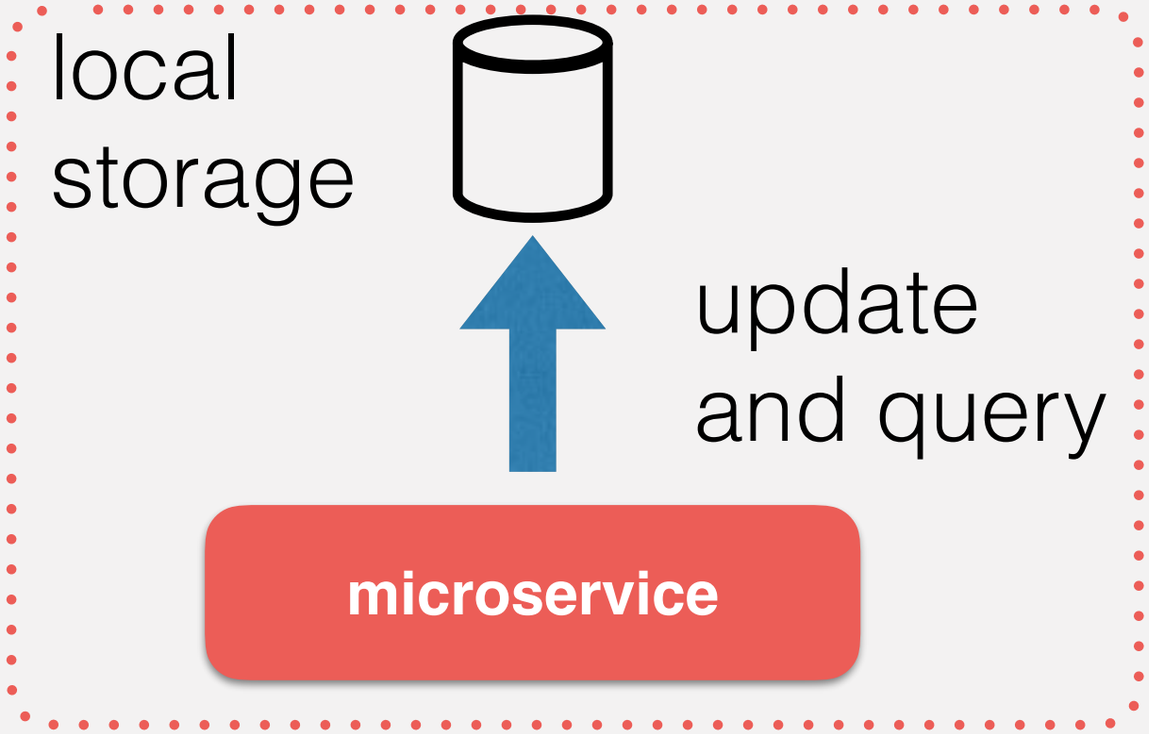
- **problem:** if connection is slow or absent the micro-service stops

sharing data

barrel



- **local storage** replicated always available
- eventually consistent



sharing data

barrel

barrel

Bring and keep a view of your data near your application

barrel

- a database focusing on simplicity
- document oriented
- Automatic indexing

Focusing on simplicity

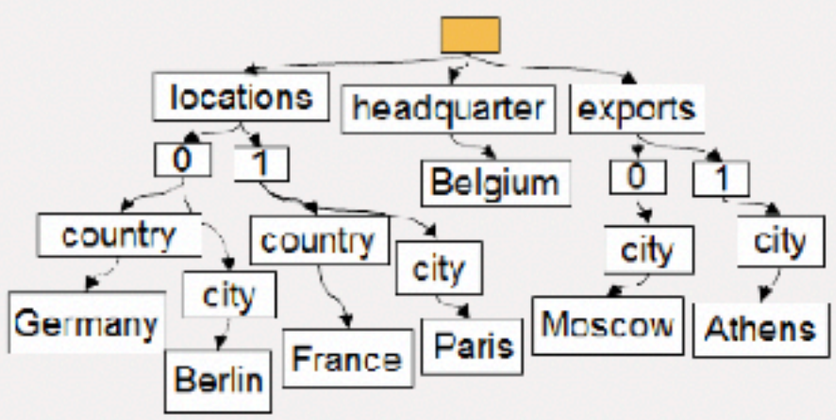
barrel

```
{ “id” :  “someid”,
   “Key” : “value” }
```

Docs are maps

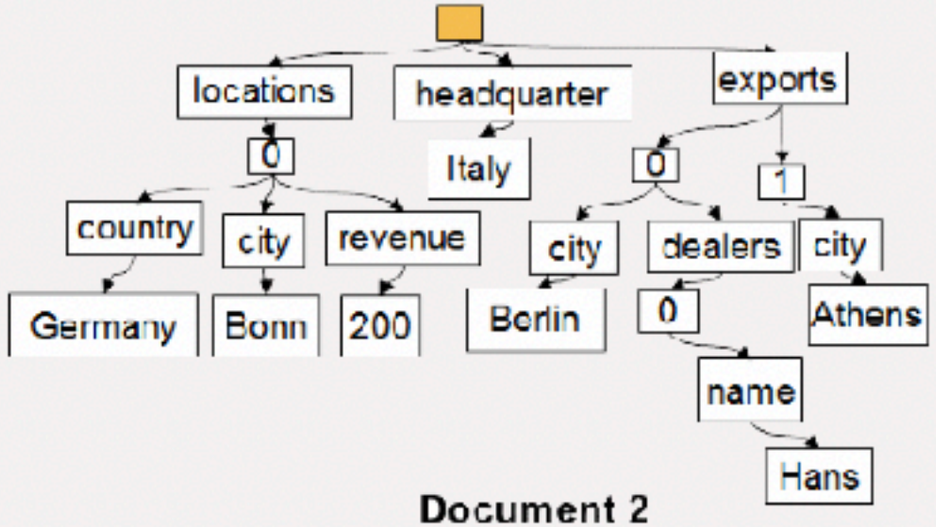
barrel

```
{ "locations": [
    { "country": "Germany", "city": "Berlin" },
    { "country": "France", "city": "Paris" }
  ],
  "headquarter": "Belgium",
  "exports":[{ "city": "Moscow" },
             { "city": "Athens"}]
};
```



Document 1

```
{ "locations": [
    { "country": "Germany",
      "city": "Bonn","revenue": 200
    }],
  "headquarter": "Italy",
  "exports": [
    [ "city": "Berlin","dealers":[["name": "Hans"]]},
    [ "city": "Athens" }]
};
```



Document 2

Access by path: /locations/country/Germany

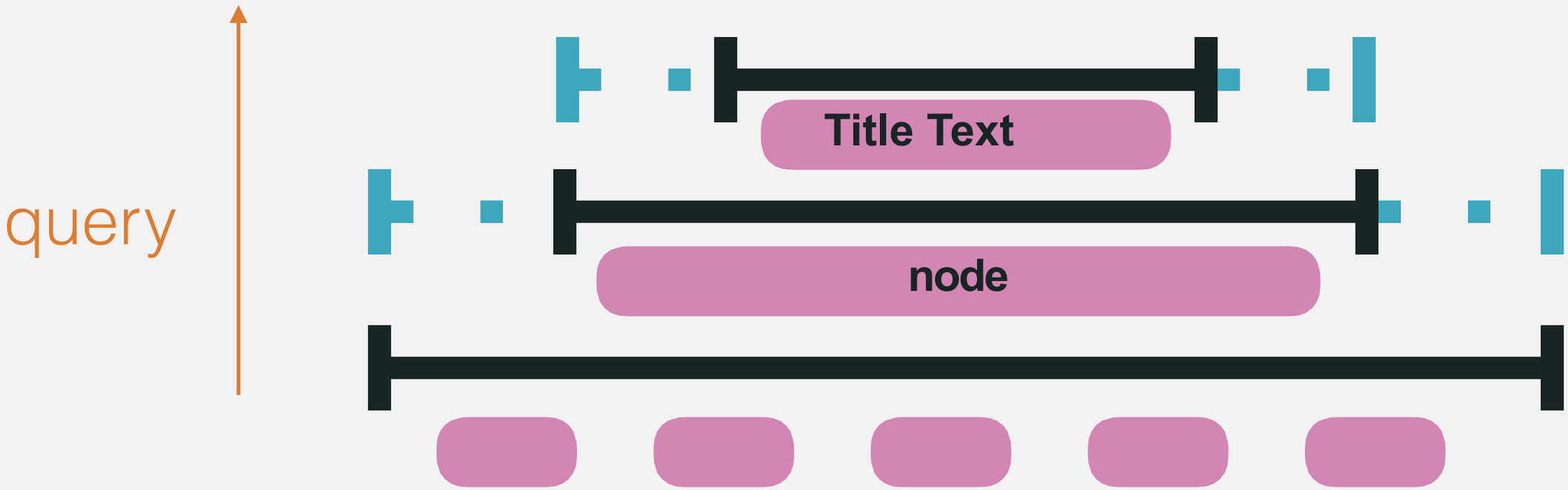
automatic indexing

barrel

- local first: bring and keep a view of your data near your application
- data is synchronised with other storages
- Replication to and from any sources

Local first

barrel



partial view

barrel

- library embedded in your Erlang application(*)
- available as a micro-service via HTTP(1,2) or via the Erlang distribution
- Peer to peer: a barrel is the unit
- Semantic to allow distributed transactions

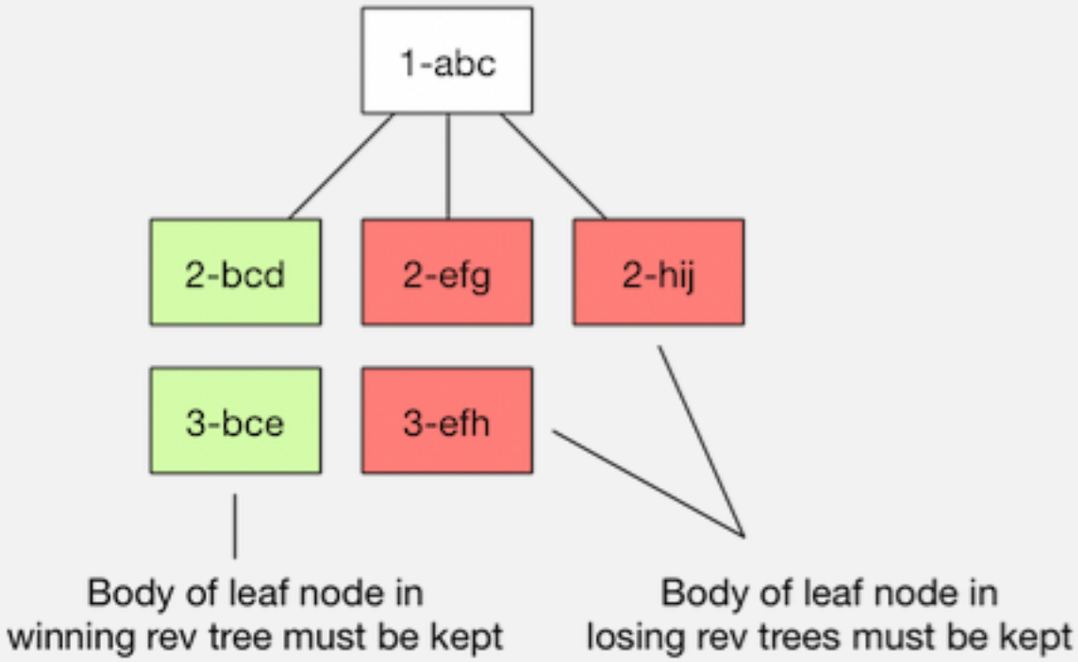
(*) including elixir or lfe, or ….

P2p

barrel

- every peers fork the master, updates are offline
- peers pull and merge from the main server
- works well for back pressure (writes can be delayed)
- CRDT semantic for conflict-free data structures

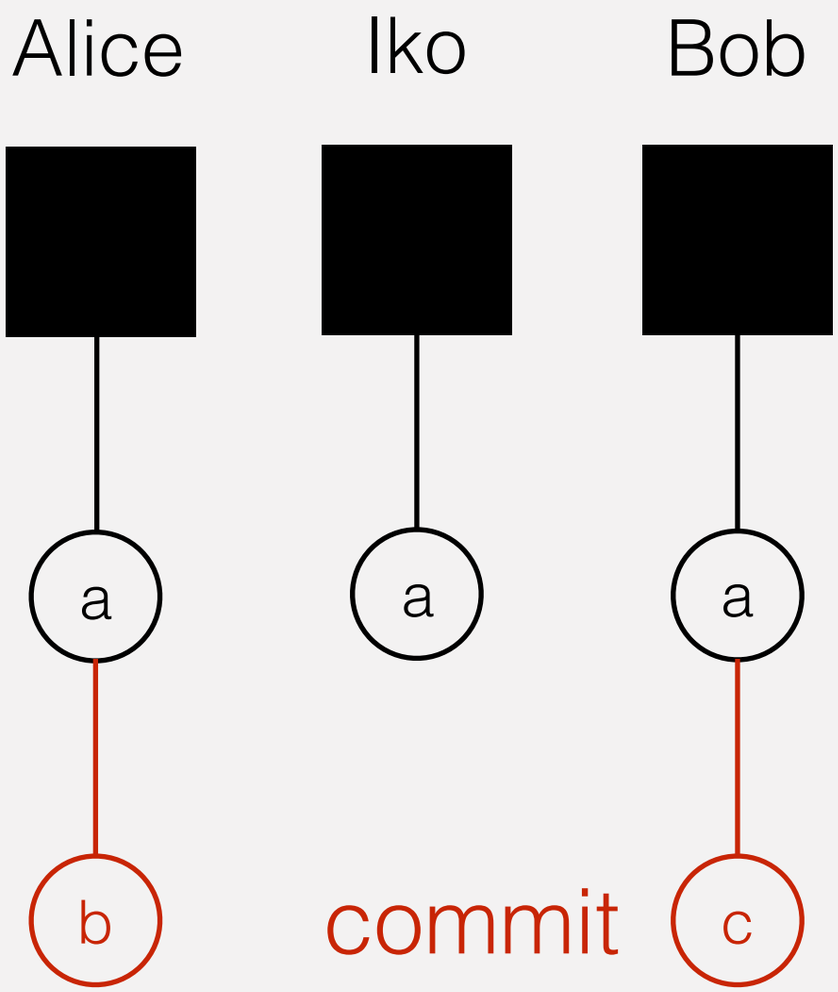
- no vector clock
- revision tree



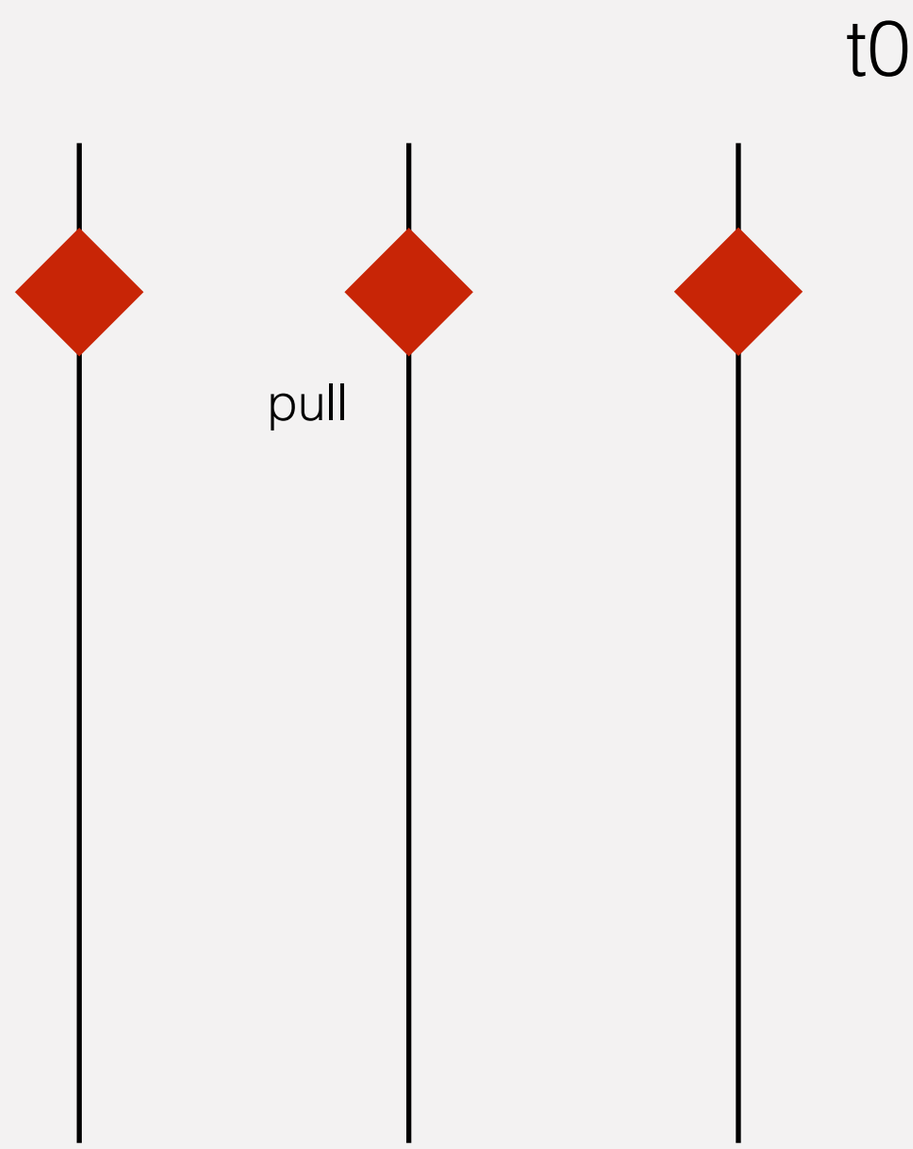
causality

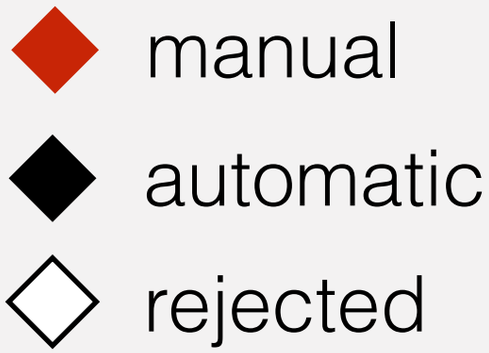
barrel

state



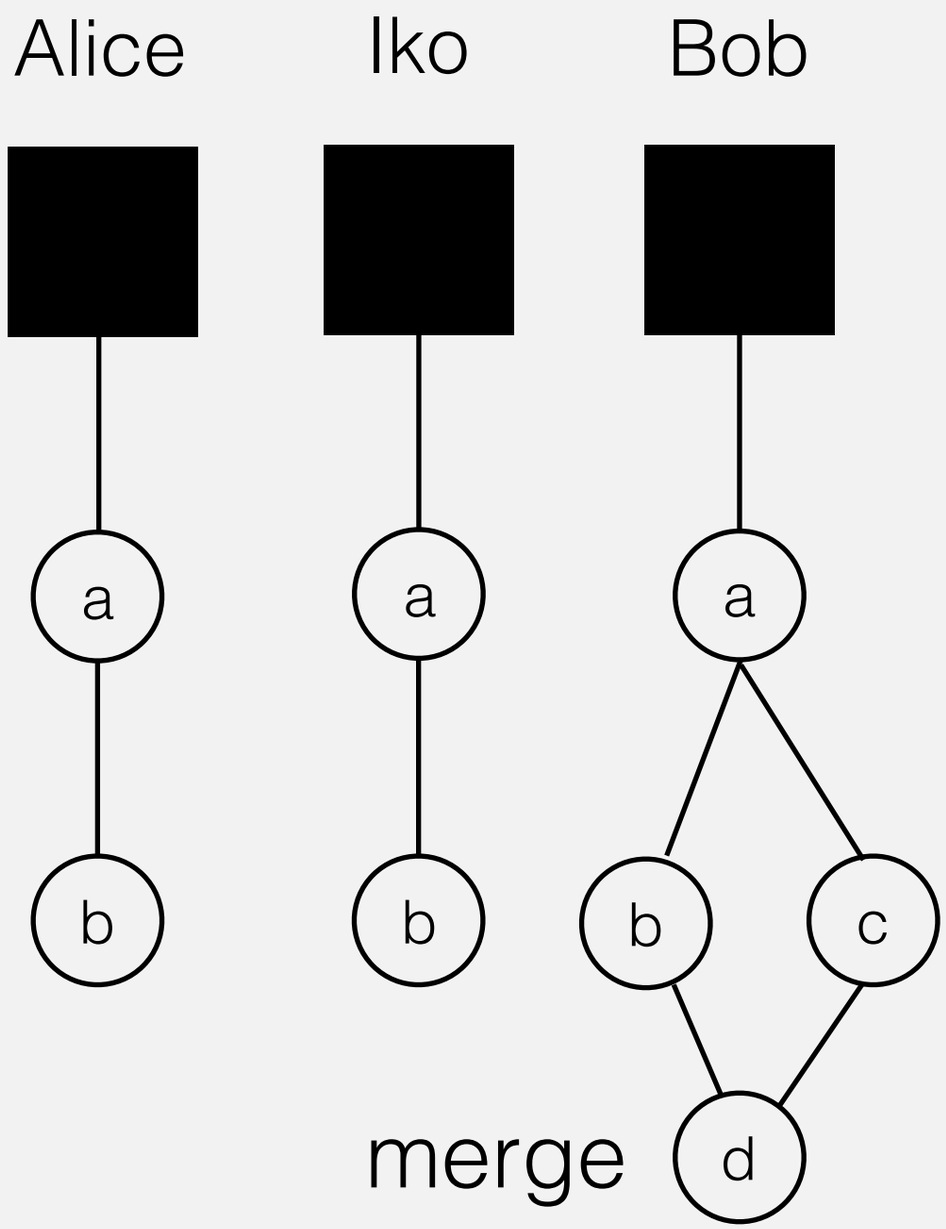
operations



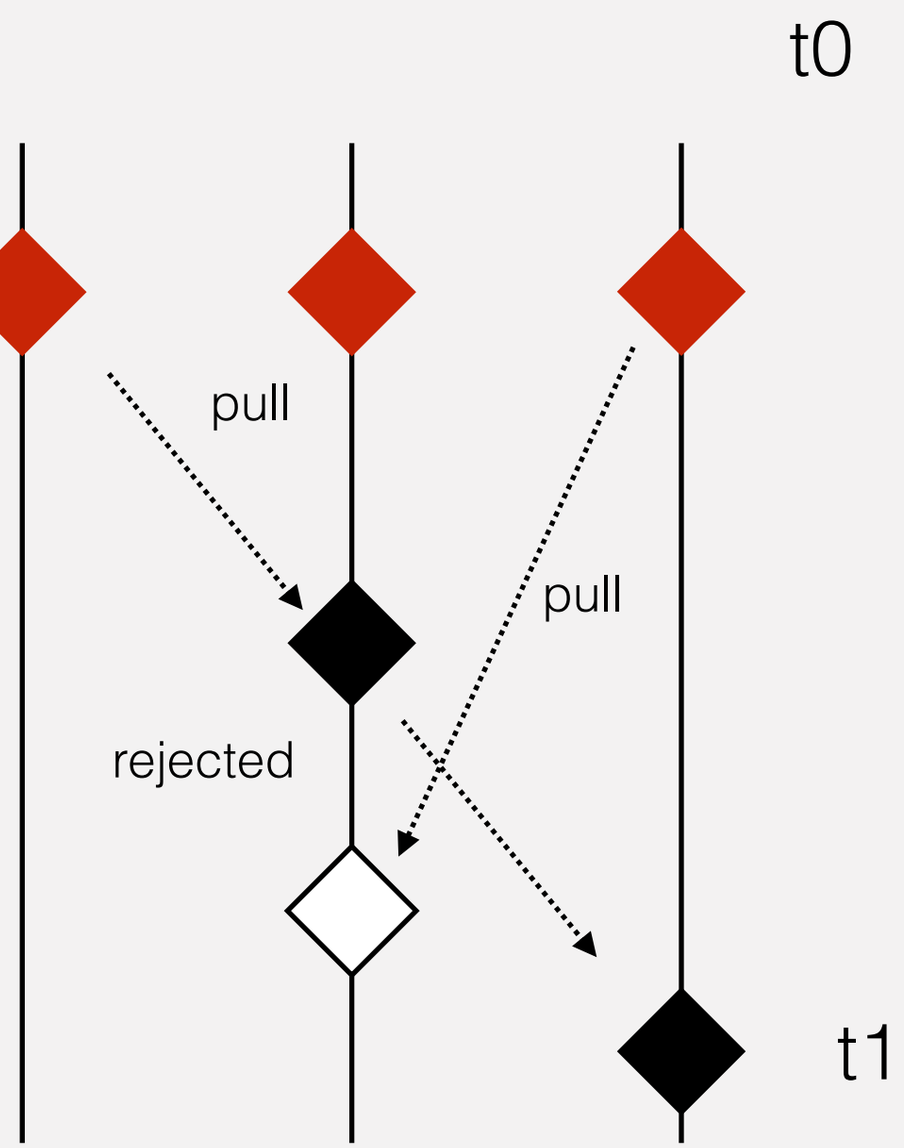


barrel

state

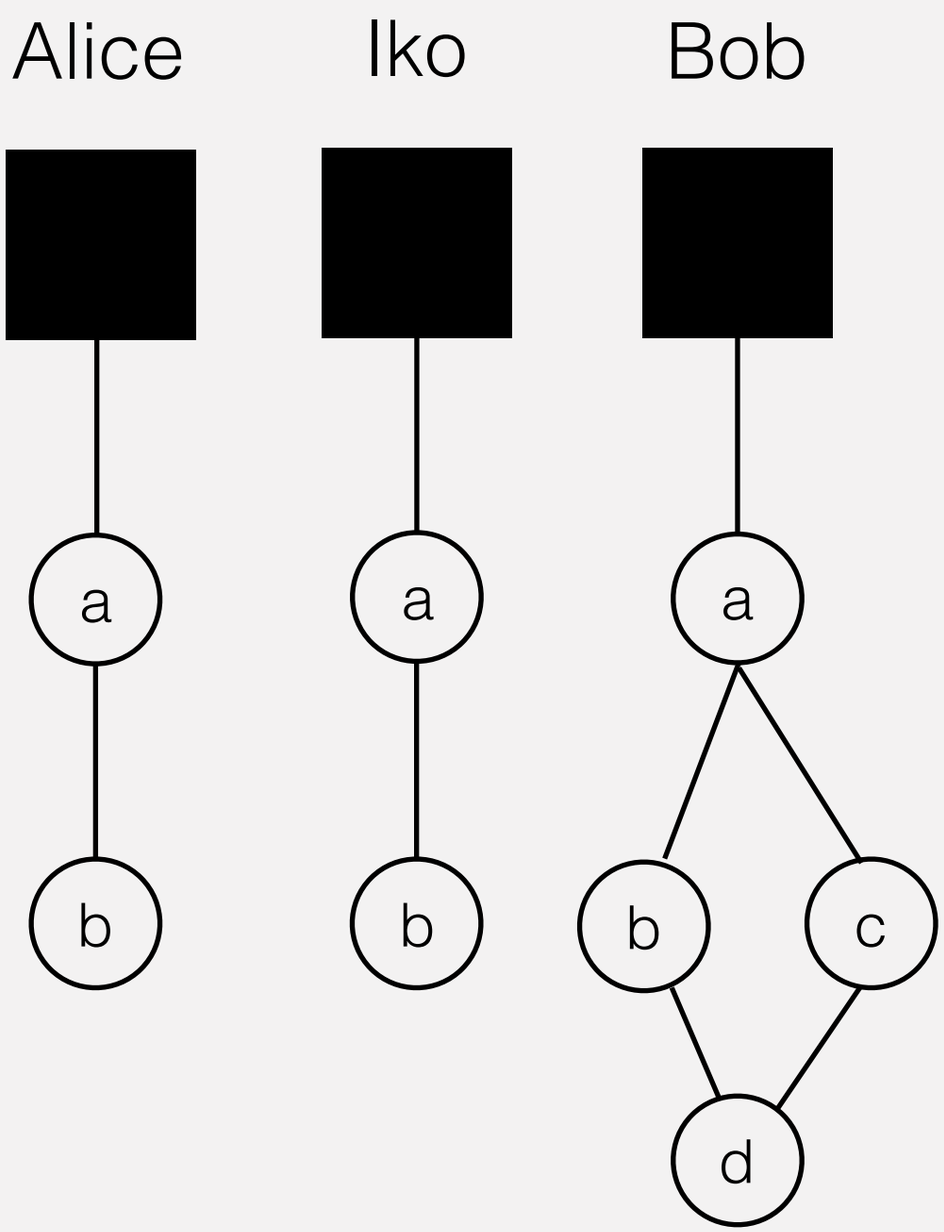


operations

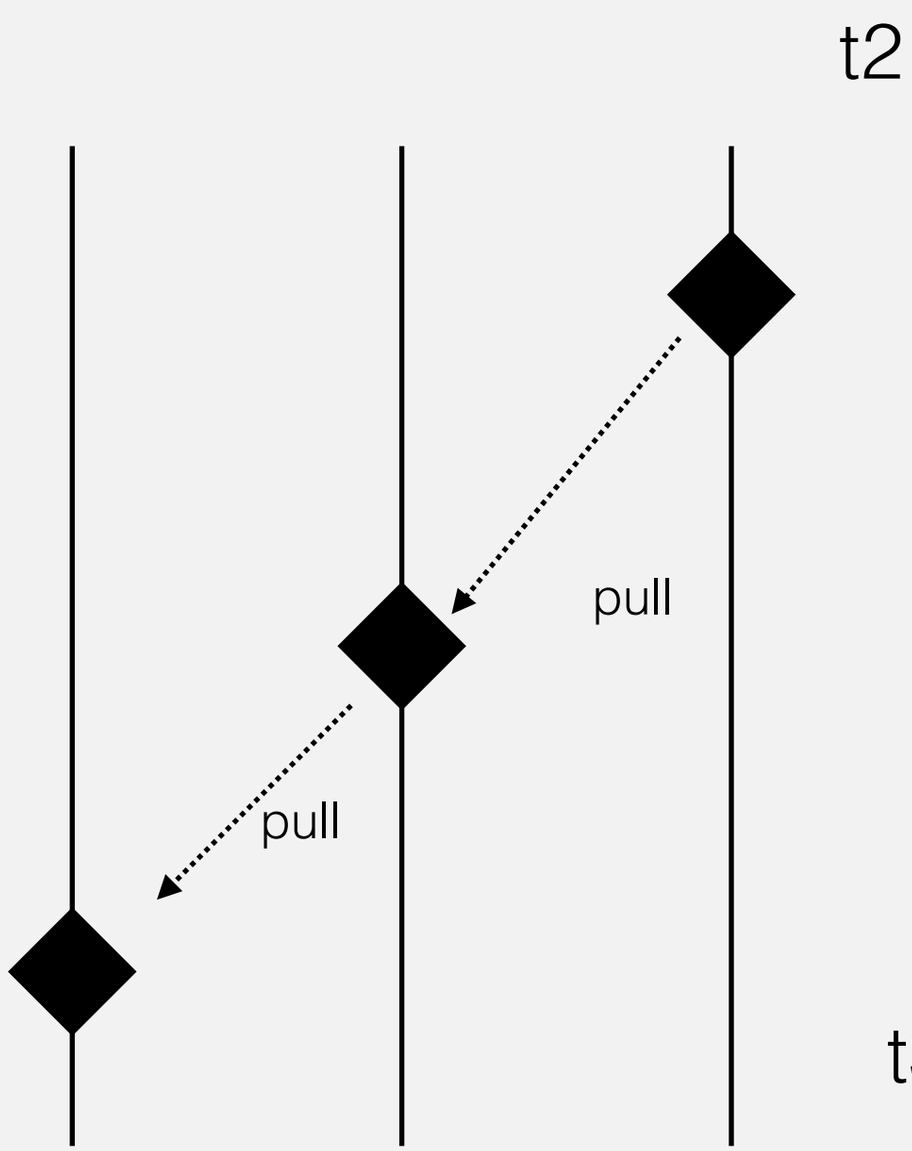


barrel

state



operations



barrel

Erlang ??

barrel

- Erlang is slow
- Erlang is only for communications protocols
- I should do it rust…
- No access to low level memory and file systems APIS

Why not Erlang

barrel

- Barrel is more a data orchestration service than a database
  - Basic indexing
  - Focus on replicating the data
- Nifs to help

Why Erlang

barrel

- Doc: Revision + Metadata data:
- Read-Modify-Write: concurrency issue
- Incremental changes log: append only
- Indexes: when a new winning version is found the doc is indexed.
- Blobs (attachments)

What we write

barrel

- Provides connectors for other storages
- RocksdDB for local persistent storage
  https://gitlab.com/barrel-db/erlang-rocksdb.git
- Dirty-nifs
- ETS?

Use the right tool for …

barrel

- Goal: anticipate the resource usage at the node level
- Return early to the client
- Control applied to all resources in the nodes
- Back-pressure

l*et it bend: be resilient*

barrel

- worker_pool
  https://github.com/inaka/worker_pool

- Hard to debug your program
- Little control on the pending requests
- Ecpocxy but handle back-pressure the reverse way

Simple pooling

barrel

- Clients and Jobs should be handled independently
- Active and passive regularion
- Request unit: to set the number of requests we want to serve / seconds
- Flow-Based programming?
- sbroker, partially fit the bill: https://github.com/fishcakez/sbroker

Dynamic regulation

barrel

- Started with a simple “Single Writer Multiple Readers” pattern
- bottleneck: A process to handle the final write to the database
- We do and // most of the work out of the write process
- Indexes are processed asynchronously (but a session can read its own writes if needed)

Concurrency challenge

barrel

- Read access is shared via ETS
- On request a monitor to the db is created

ets: to share the state between readers

barrel

- When using the erlang distribution, events are dispatched by nodes, processes always subscribe locally

Events

barrel

- Erlang distribution is not used to share the data
- Erlang distribution can be switched
- HTTP transports

Transport the data

barrel

Roadmap

barrel

- 1.0: 24 march 2018
- 1.1: 24 april 2018
- ...

Milestones

barrel

- 1.0: Websockets support (with new hackney)
- 1.1: Experimental: GRPC

Coming features

barrel

?

- barrel is released in **march 2018**
- https://barrel-db.org
- contact me @benoitc

barrel