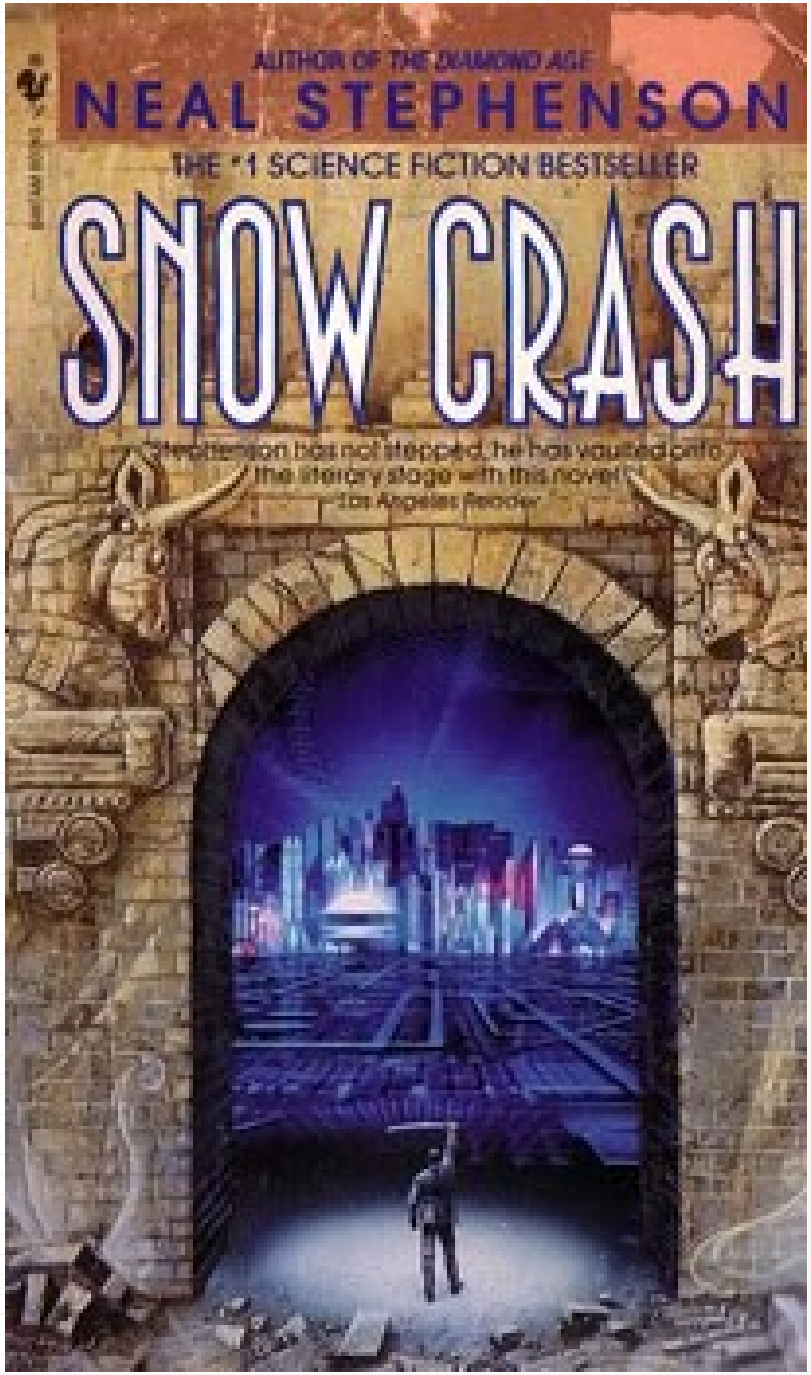
Refuge

Build a data platform over and on the web

Erlang User Conference 2013
Benoît Chesneau

About me

- Apache CouchDB committer and PMC member
- PSF Member
- Web craftsman
- Do opensource for a living



The big picture

What is Refuge?

- A way to **store**, **sync,** and **share** your data
- Decentralized
- Over and on the web
- Opensource
- Built in Erlang

Part of

Why?

Played too much with Apache CouchDB

- Document oriented
- Blobs can be attached to documents
- Master-master replication (P2P)
- Couchapps

But

What we really need at the end is...

- A simple way to store any blobs
- Index or render them
- And share them among people and machines
- Can work with off-line devices
- Or near you

Coffer

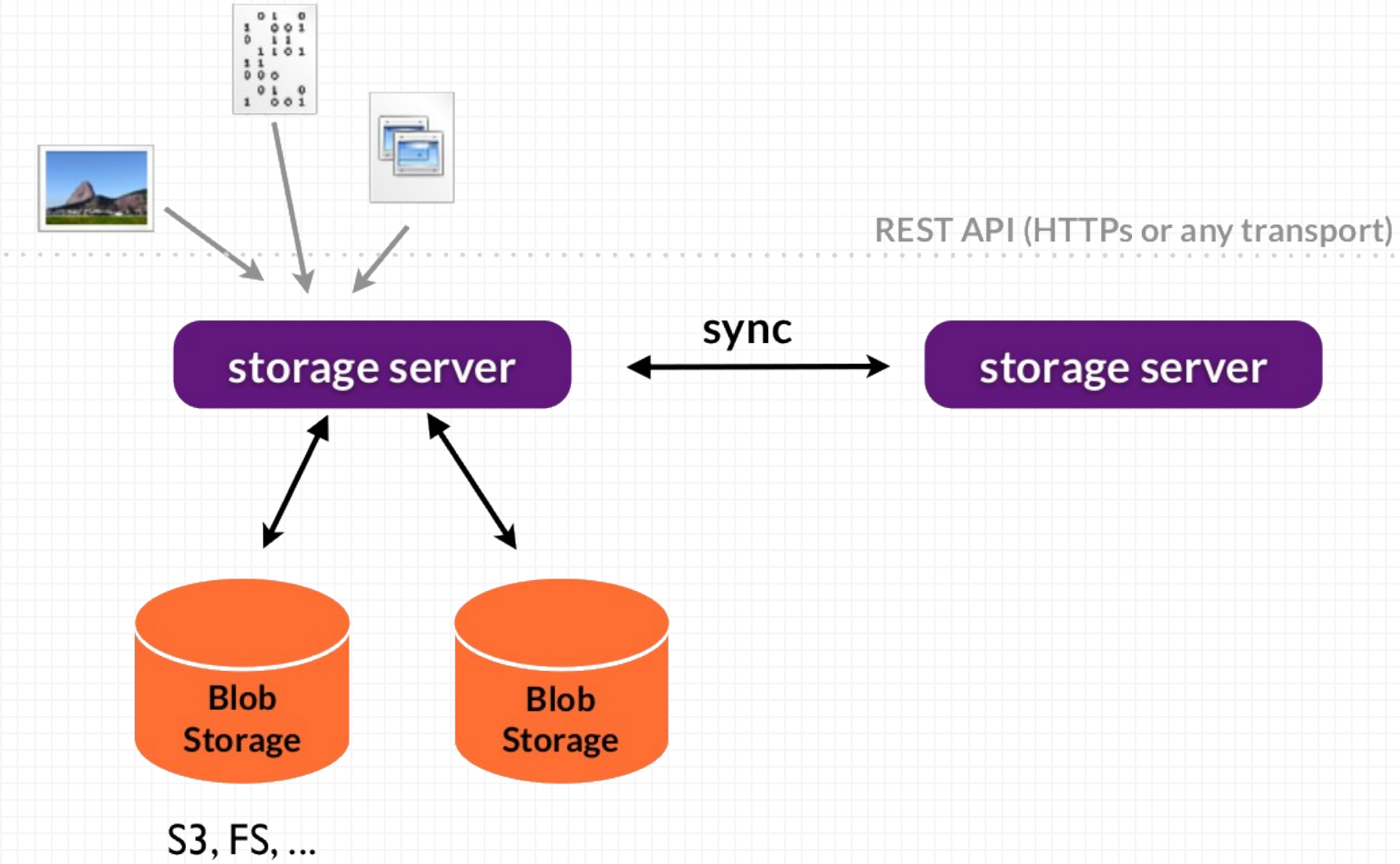
Versatile storage service

- Multi-backend (FS, Distributed FS, S3...)
- Simple REST API: GET,PUT, POST, DELETE
- Synchronization
- All blobs are uniquely identified (content-hash)

Coffer

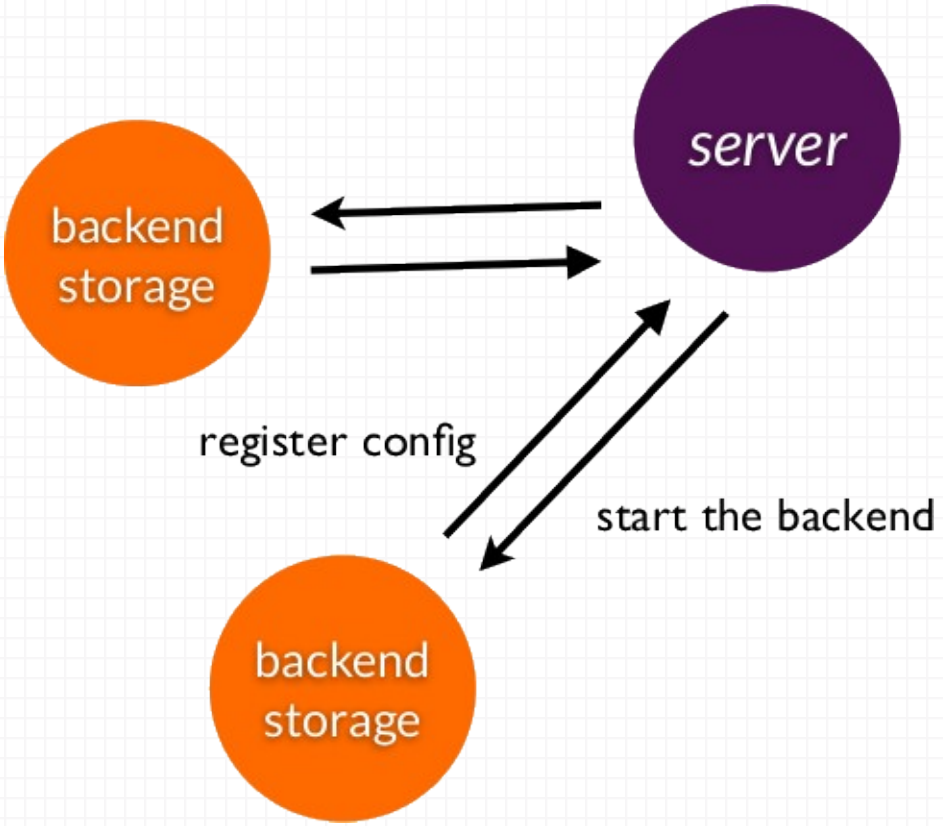
Versatile storage service

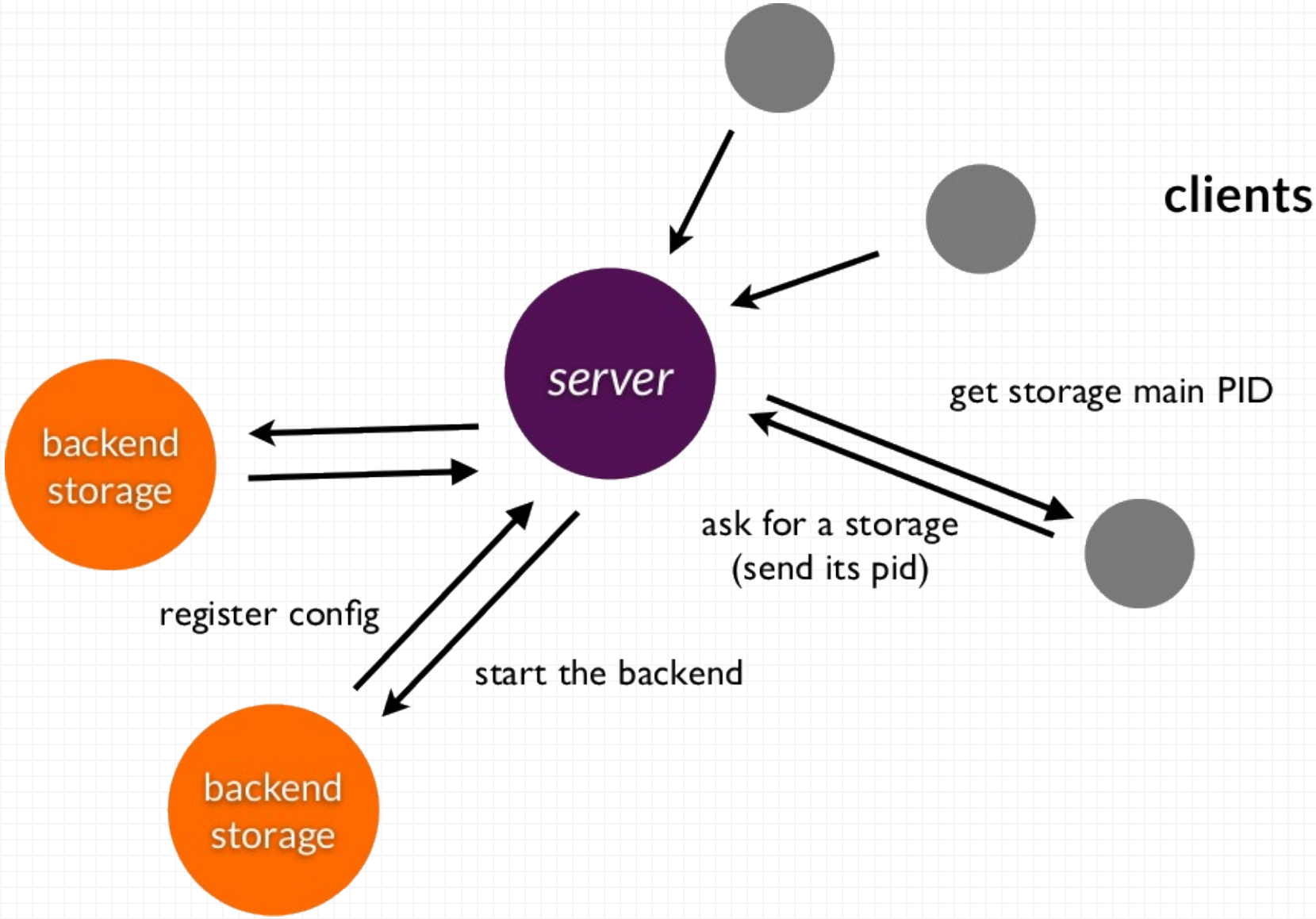
- Works with mobile & embedded devices
- Partials upload & downloads supported
- RESTFUL, resources visible on HTTP

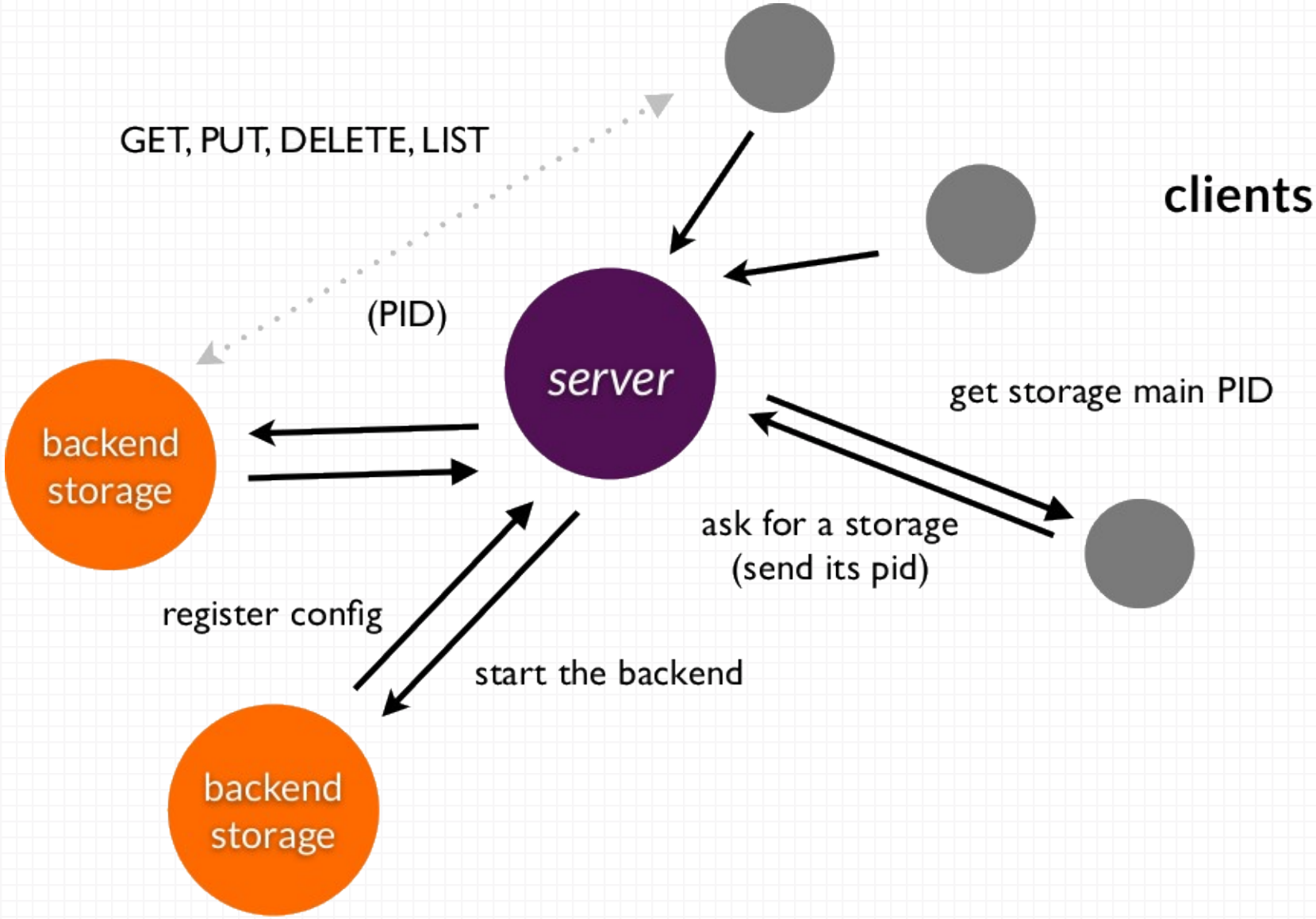

REST API (HTTPs or any transport)
storage server
sync
storage server
Blob Storage
Blob Storage
S3, FS, ...

Erlang Side

- A gen_server to keep all the storage backend configuration
- gen_storage , a behaviour to keep a storage states
- Handle conflicts in the backend. Same file can't be uploaded by 2 clients at the same time


server
backend storage
register config
start the backend
backend storage


clients
server
backend storage
get storage main PID
ask for a storage (send its pid)
register config
start the backend
backend storage


GET, PUT, DELETE, LIST
clients
(PID)
server
backend storage
get storage main PID
ask for a storage (send its pid)
register config
start the backend
backend storage

Synchronization algorithm

- Create a change queue on the source
- Enumerate (list) all blobs on the source and copy the blobs not on the target
- Enumerate is cheap (we only compare blobs ids)
- Blobs already on the target aren't sent
- Start to watch on the change queue
- Blobs delivery guarantee

How changes queues work in Erlang

- Queues are kept in memory and persited to the disk from time to time
- A process / queue
- A process / watcher is maintained
- Pub/sub over websockets, event source or longpolling

Today support

- Memory storage (ETS)
- FS Storage
- Distributed using riak-coree
- More to come (S3, Redis, Memcache)

Some tools we use

- **Modified cowboy** version – http server
- **Hackney** – http client
- **Gproc** for a cheap pub/sub internal system (changing)

Hackney in action

```
ping(#coffer_conn{url=URL, options=Opts}) ->
    case hackney:head(URL, [], <<>>, Opts) of
        {ok, 200, _, Client} ->
            hackney:close(Client),
            pong;
        _ ->
            pang
    end.
```


How to use the blobs?

- No metadata on the disk
- No history
- Just blobs

Create a backup of a folder

- 3 kinds of blobs
- 1 blob for the file
- 1 blob keeping file metadata (name, type...)
- 1 blob keeping the directory structure

file

```
{
  "blobid": "blobobid",
  "prev": "prevref or null"
}
```

tree

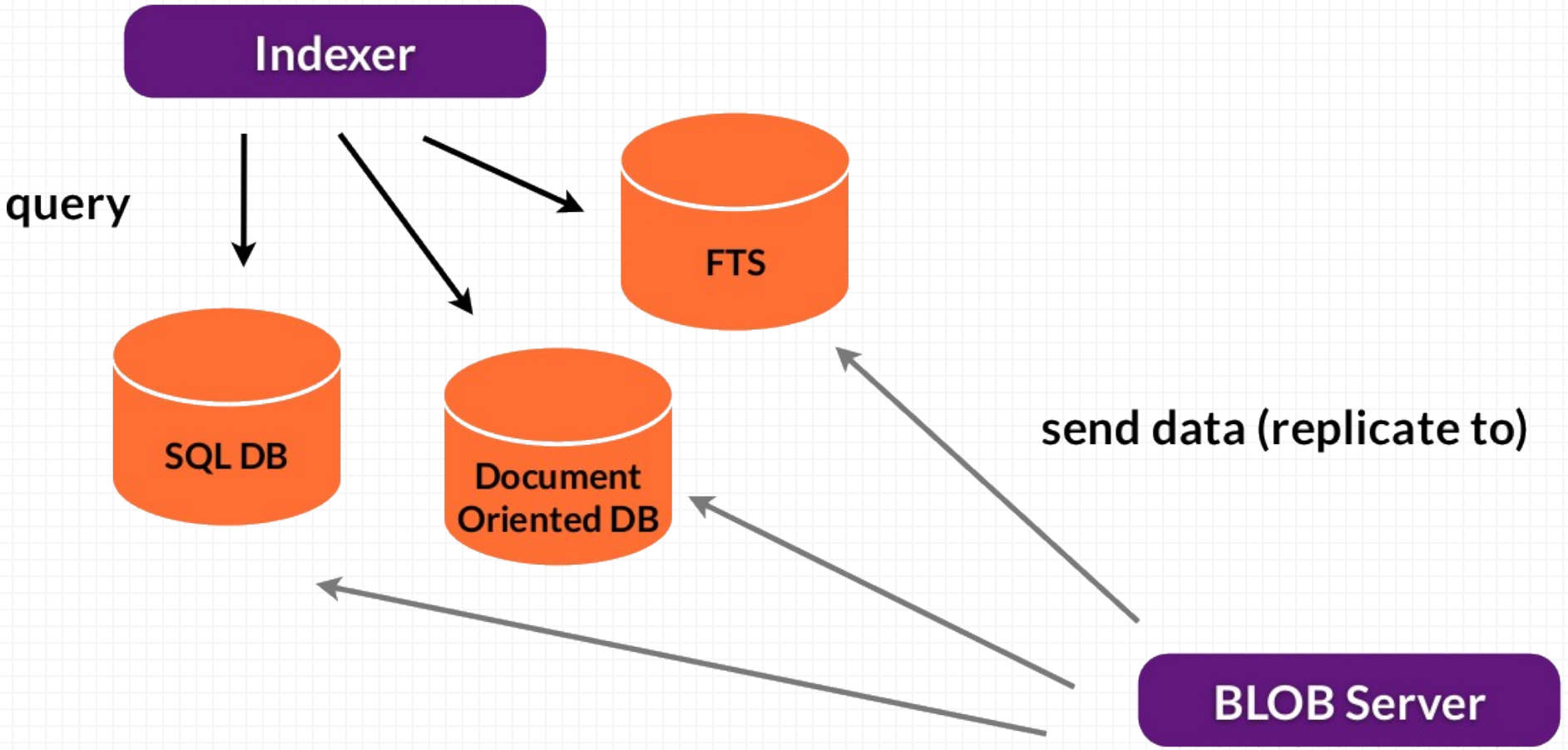
```
{
  "filename": {
    “blobid: "blobbid",
    “type”: “blob”
  },
  "foldername": {
    “blobid: "blobbid",
    “type”: “tree”
  }
}
```


Create a backup of a folder

- Create a reference (or link) to the last version of the tree
- Another blob with a level of indirection
- Can be signed
- Just another blob

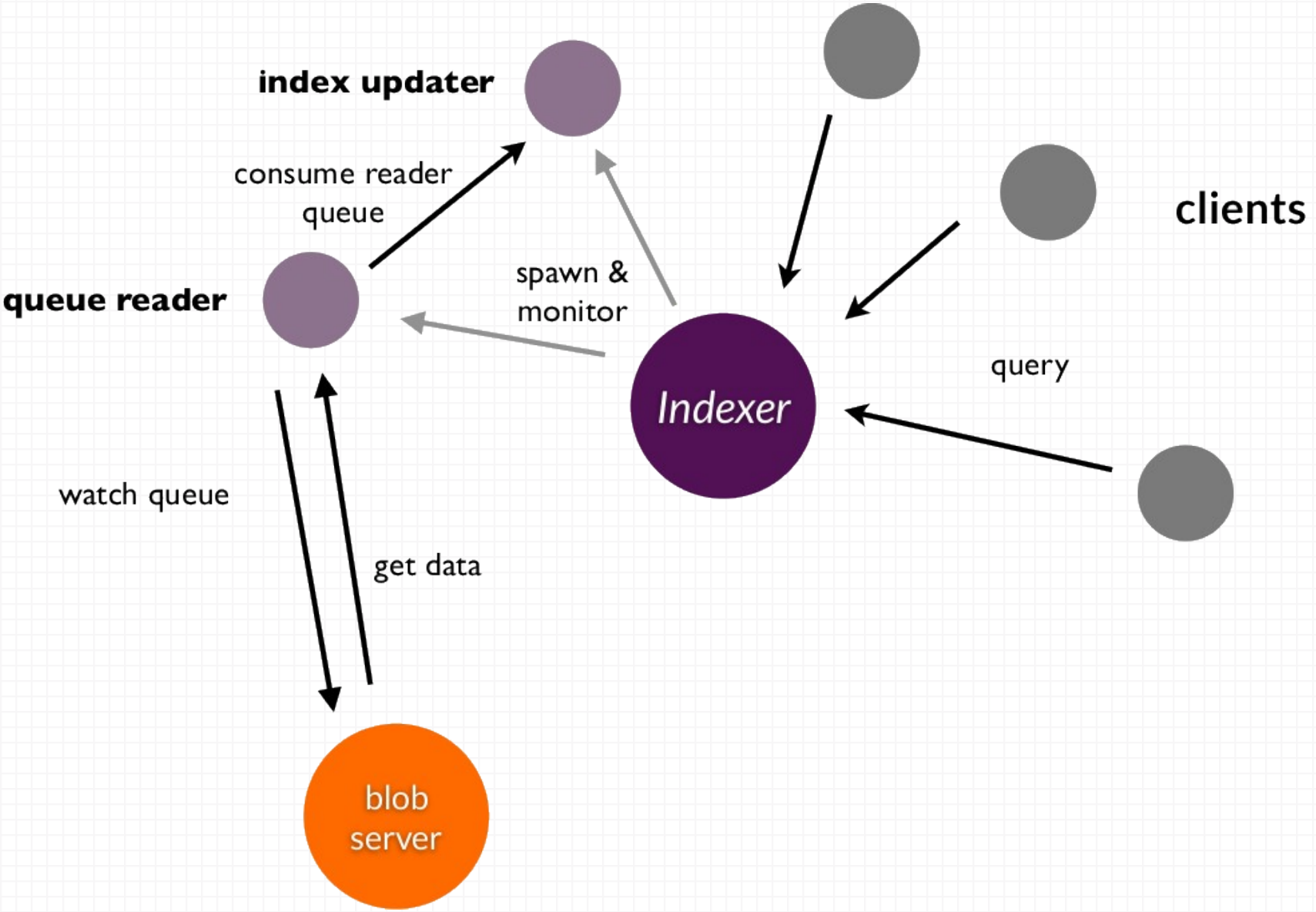
Index your content

- “just” replicate your index
- An indexer receive {blobref, Blob, timestamp} from the replication queue
- Can be any kind of indexer: sql, apache couchdb, an FTS (like elasticsearch)


Indexer
query
FTS
SQL DB
Document
Oriented DB
send data (replicate to)
BLOB Server

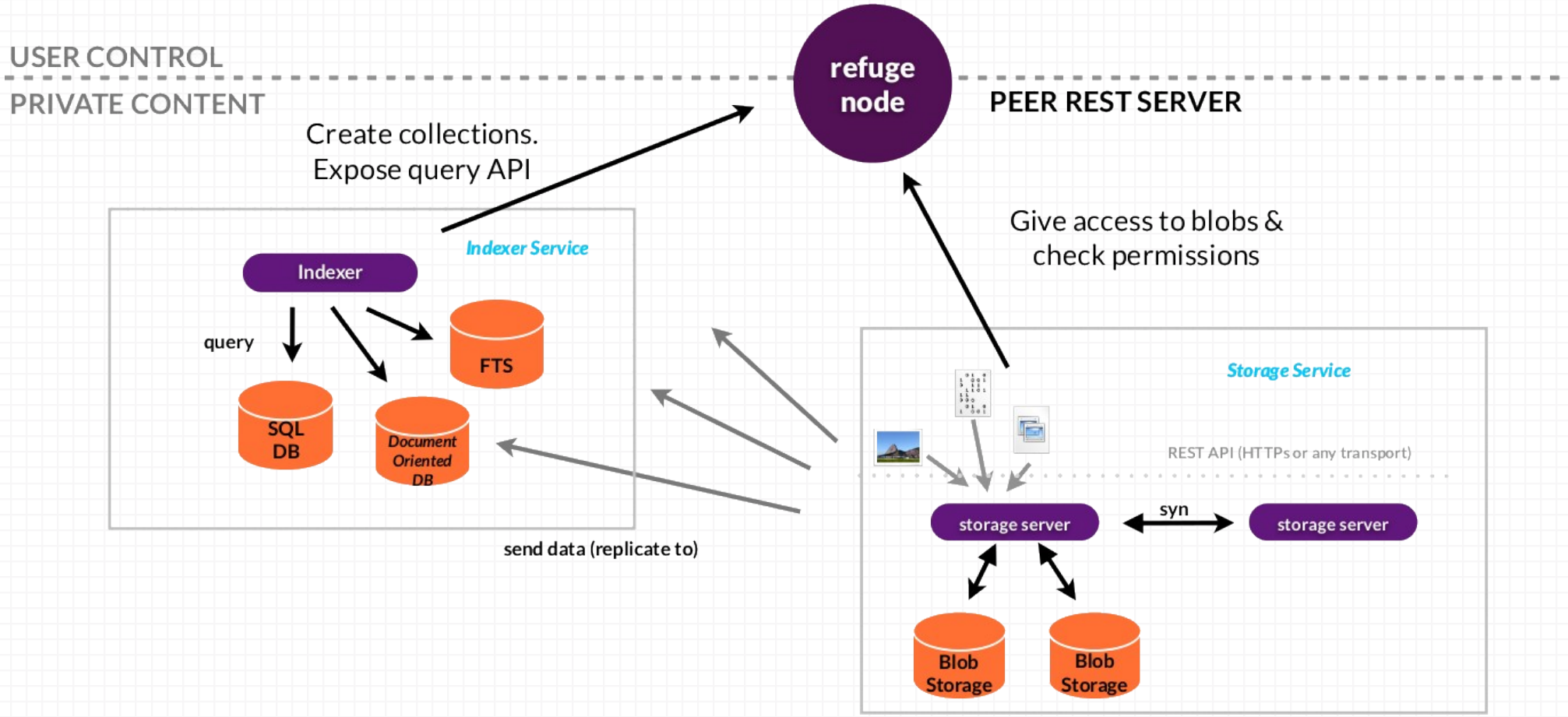
Behind the scene

- Mostly works like a blob server
- Except it only pass mapped data to the index
- Possibility to transform the data before indexing (mapping)
- Multi-language


index updater
consume reader queue
queue reader
spawn & monitor
clients
Indexer
query
watch queue
get data
blob server

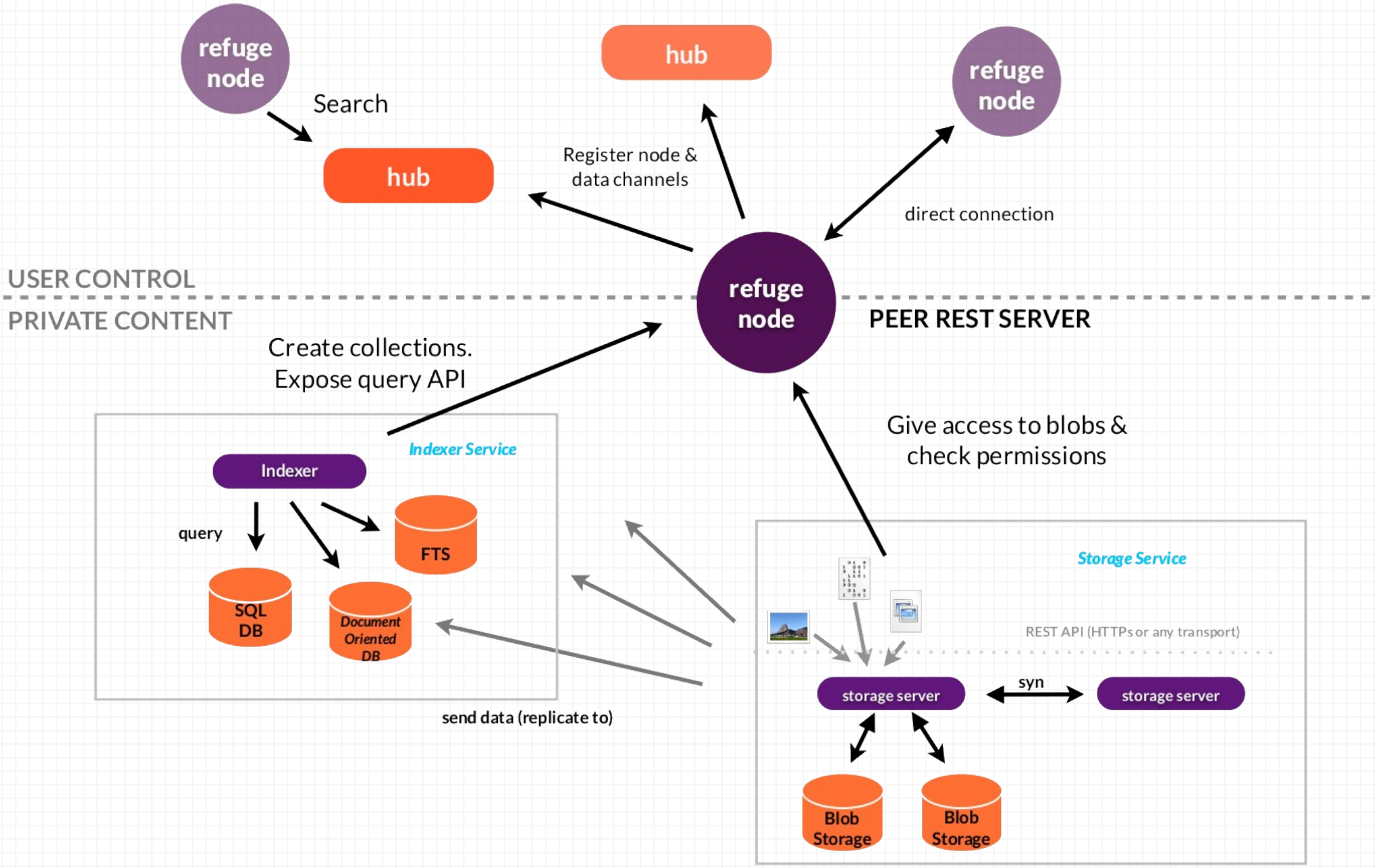
The refuge node

- Frontend to blobs servers and indexers
- Manage blobs claims and access
- Gateway to others nodes
- Forward requests
- WebRTC signaling


USER CONTROL
PRIVATE CONTENT
refuge node
PEER REST SERVER
Create collections.
Expose query API
Give access to blobs &
check permissions
Indexer Service
Indexer
query
FTS
SQL DB
Document Oriented DB
send data (replicate to)
Storage Service
REST API (HTTPs or any transport)
storage server
syn
storage server
Blob Storage
Blob Storage

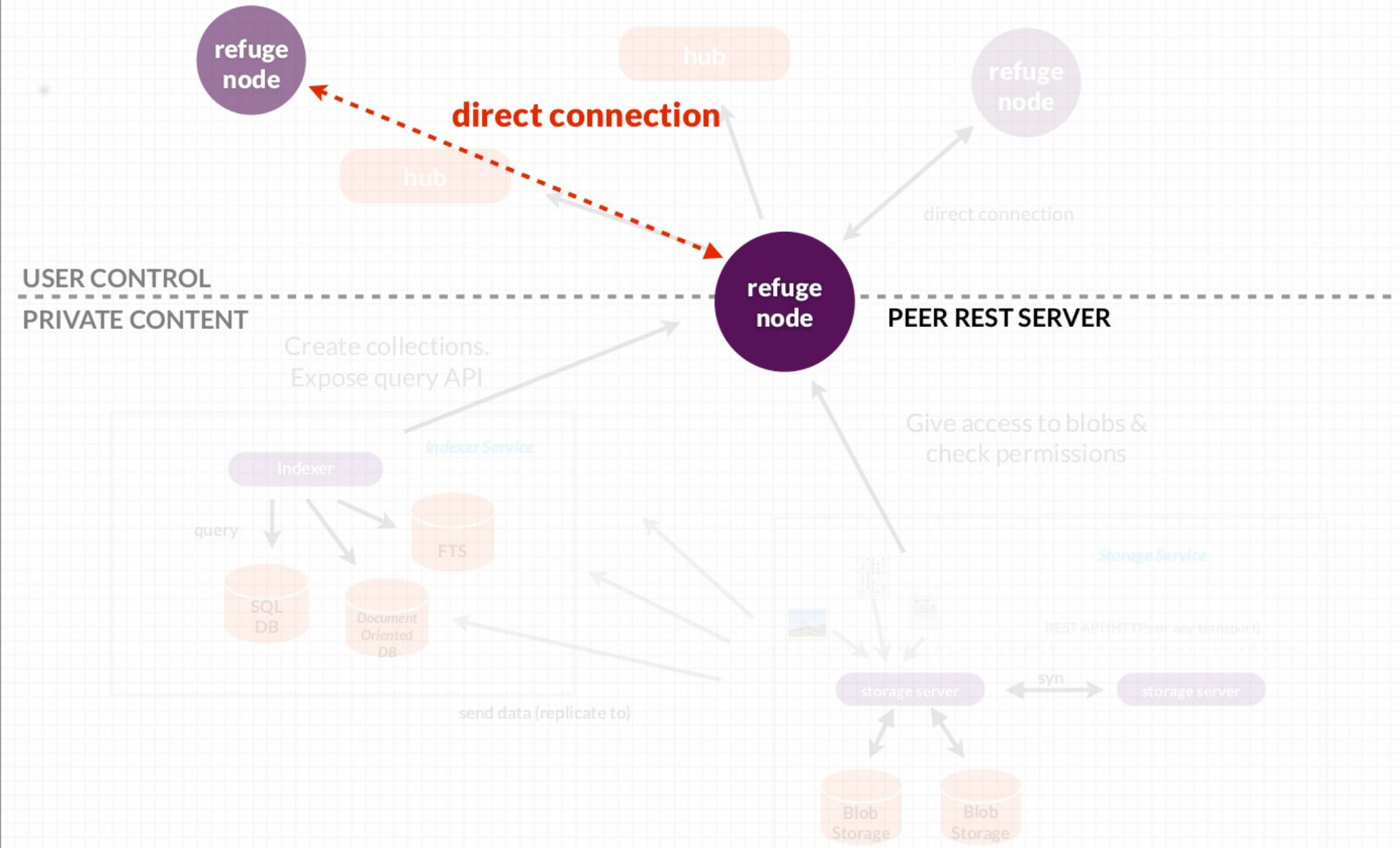
The hub

- True decentralized web system
- Once found nodes are directly connected
- A node can be authenticated using a signature
- Webfinger & host-meta


refuge node
Search
hub
hub
Register node & data channels
refuge node
direct connection
USER CONTROL
PRIVATE CONTENT
refuge node
PEER REST SERVER
Create collections.
Expose query API
Give access to blobs & check permissions
Indexer Service
Indexer
query
FTS
SQL DB
Document Oriented DB
send data (replicate to)
Storage Service
REST API (HTTPs or any transport)
storage server
syn
storage server
Blob Storage
Blob Storage

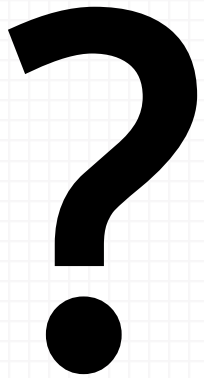
The hub

- Each node can open a connection to an hub using a websocket
- A node can connect to multiple hub
- Once connected a node authenticated itself with a signature


refuge node
direct connection
hub
refuge node
hub
direct connection
USER CONTROL
PRIVATE CONTENT
refuge node
PEER REST SERVER
Create collections.
Expose query API
Give access to blobs &
check permissions
Indexer Service
Indexer
query
FTS
SQL DB
Document Oriented DB
Storage Service
REST API (HTTPS or any transport)
storage server
syn
storage server
send data (replicate to)
Blob Storage
Blob Storage

The hub

- Each refuge node open a connection to an hub (websocket)
- A node can connect to multiple hub
- Once connected a node authentified itself with a signature



@benoitc
http://refuge.io

Thanks to
Laurent (@lolograph) for the website and logo design
Nicolas (@nrdufour) for the code and the ideas
Others for their feedback