WHY POSTGRESQL DESERVES NOSQL FANS' RESPECT

CLEMENT DELAFARGUE

@CLEMENTD

WHO AM I ?

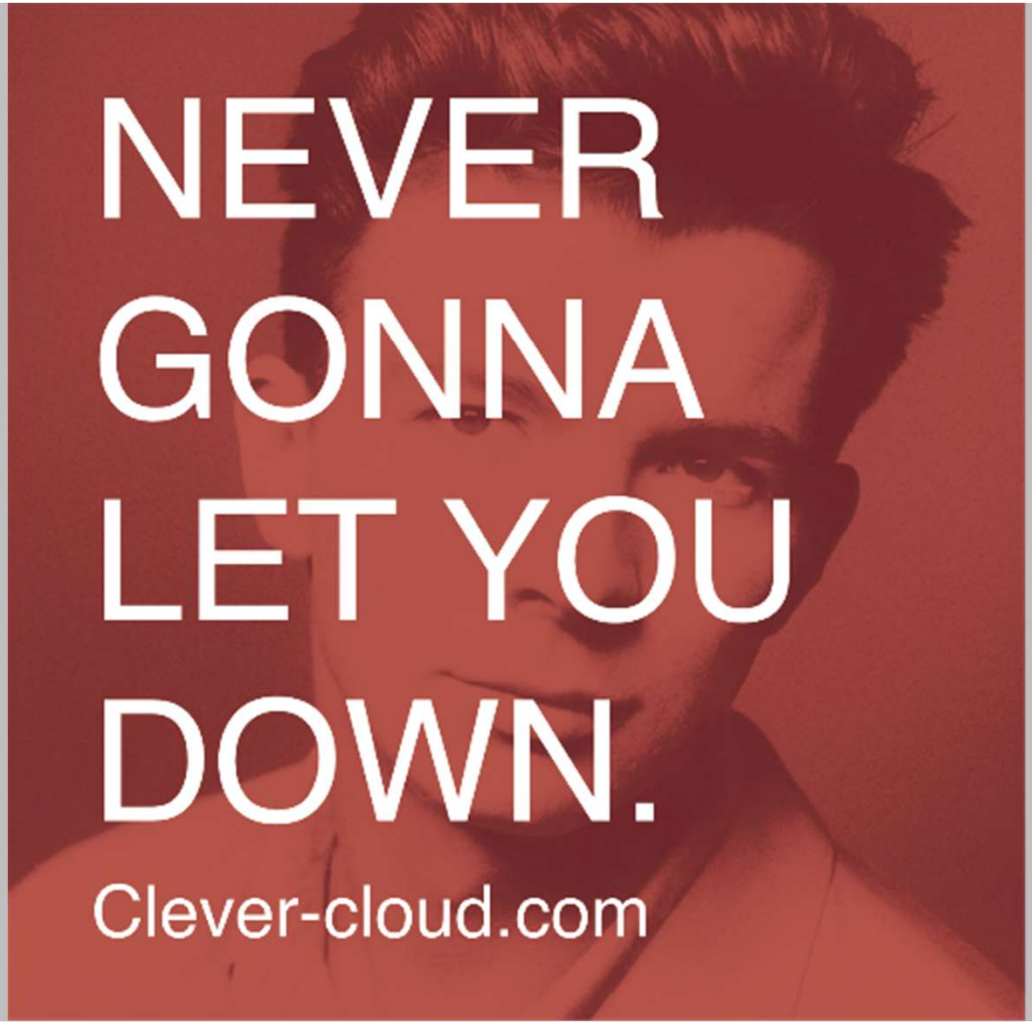
Clement DELAFARGUE from Clever Cloud

@clementd on twitter – @divarvel on github

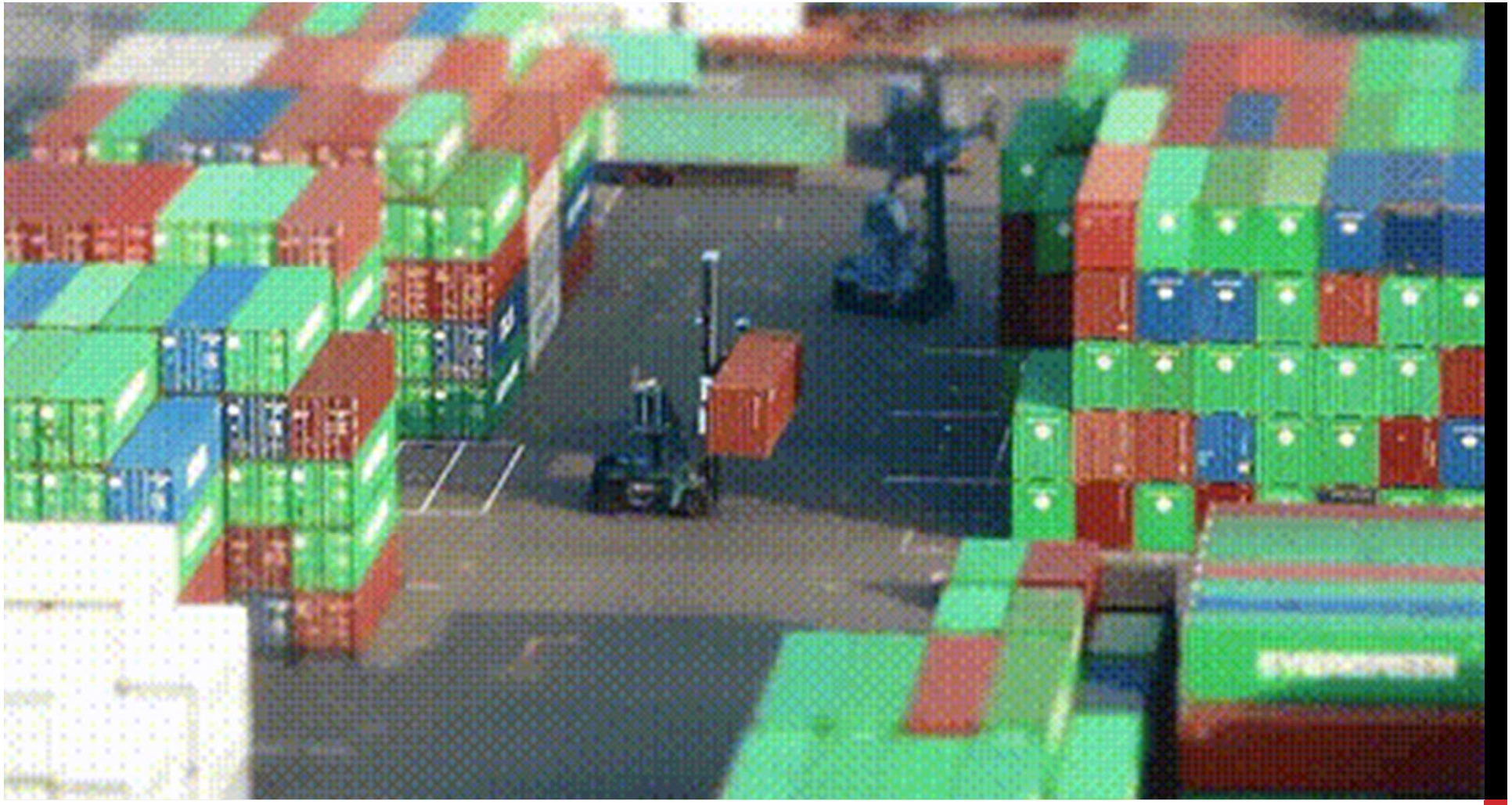
MY DAY TO DAY WORK : CLEVER CLOUD, MAKE YOUR APP RUN ALL THE TIME

clever cloud

KEEP YOUR APPS ONLINE. MADE WITH NODE.JS, SCALA, JAVA, RUBY, PHP, PYTHON, GO...



AND LEARN A LOT OF THINGS ABOUT YOUR CODE, APPS, AND GOOD/BAD DESIGN…



NOSQL WORLD

Mongo
Riak
Riak2
Redis
Couchbase
CouchDB
ElasticSearch
Titan
Cassandra
Kafka
Hbase
Neo4J
Datomic

SOME OF THE COOL THINGS

- ACID or not
- Schemaless or not
- Concurrent writing or not
- Consistent or eventually consistent
- HA or not
- …

THINK ABOUT DATA STORAGE

EXAMPLE TIME

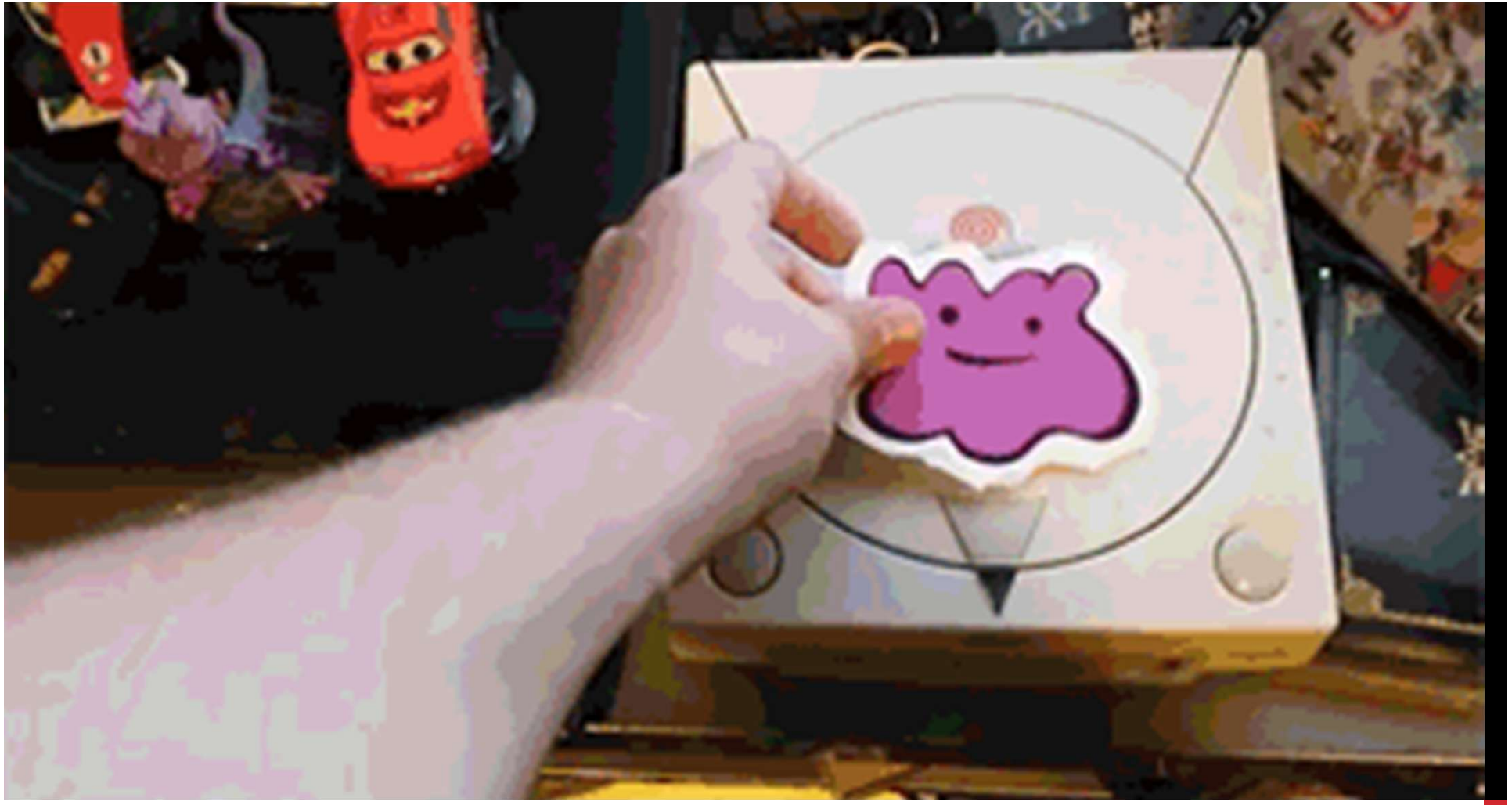
TO DO

PLAYLIST MANAGEMENT APP, MULTIPLE BACKENDS, PLAYLIST HISTORY

ENTITIES

- user management
- Playlists
- Tracks history (deezer, spotify, soundcloud)

TRACK DATA IS HETEROGENEOUS



WE NEED SCHEMALESS STORAGE

ARBITRARY DATA DEPTH

EXTERNAL DATA IS ALREADY JSON

“I NEED USER EMAILS TO BE UNIQUE”

CONSTRAINTS

“WHICH TRACKS ARE PLAYED BETWEEN 8PM AND 9PM ON WEEK DAYS?”

COMPLEX DATA EXPLORATION IS HARD

“… IN PARIS AREA”

COMPLEX DATA EXPLORATION IS HARD

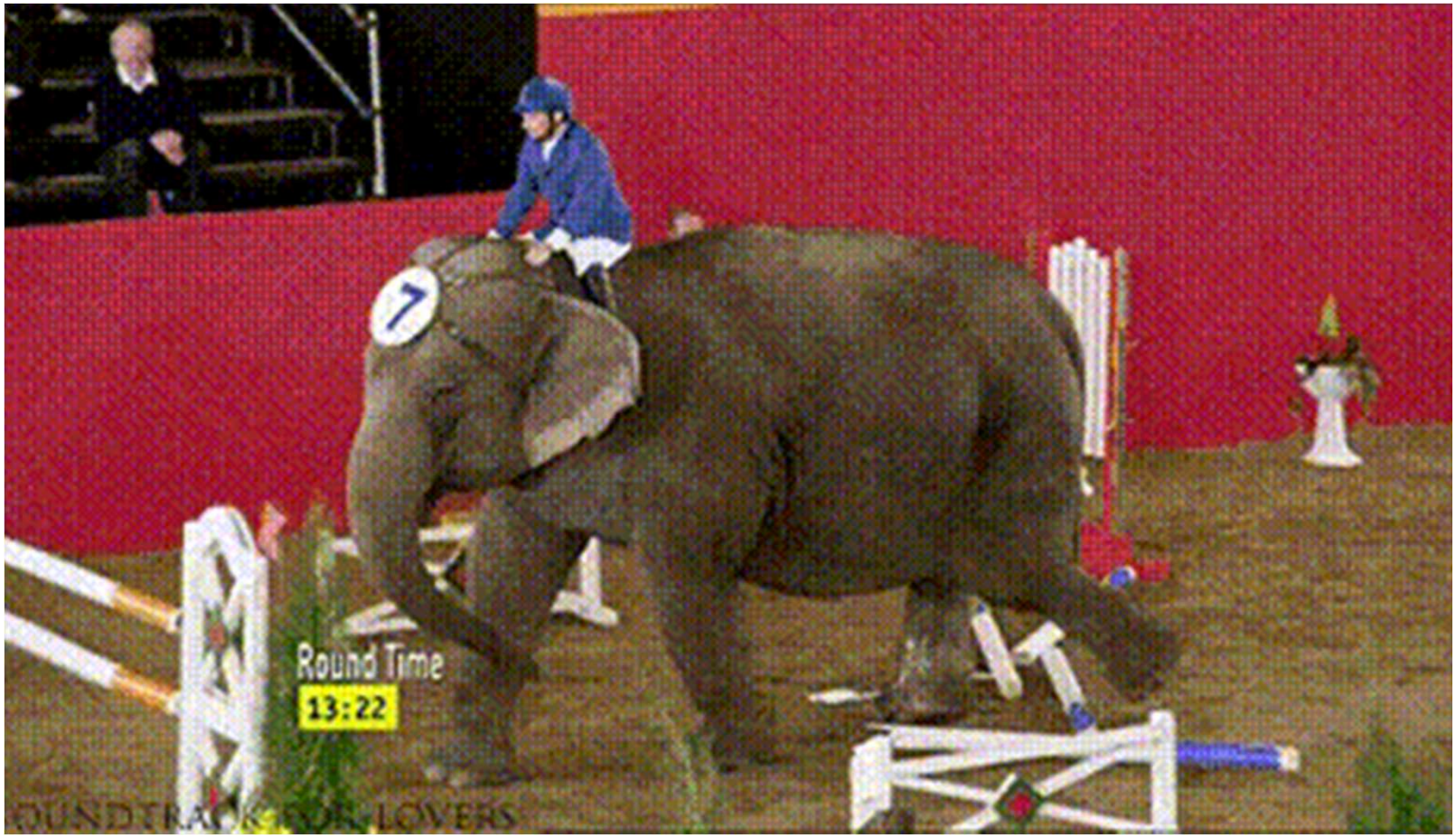
“… BY NEW USERS”

NEED SOME RELATIONS IN YOUR DATA STORE?

“…ORDER BY..”
“…PAGINATED UI…”

WE ALREADY KNOW ALL THIS

POSTGRESQL



LET'S CREATE OUR TRACKS TABLE

```
create table track(
    track_id uuid primary key,

    playlist_id uuid not null
      references playlist(playlist_id),

    user_id uuid not null
      references "user"(user_id),
```

UUID AS PRIMARY KEY & NO SERIAL

“WE HAVE 2000000 REGISTERED USERS”

http://www.my-platform.com/users-creation-confirm/42

?valid_key=jgqsdkfhqksdjfhkjsdhf2342345

SAME ID FOR VARIOUS ENTITIES IN YOUR DATABASE

LET'S CREATE OUR TRACKS TABLE

```
    status track_status not null,

    name text not null,

    length interval not null,

    played_at timestamptz,

    created_at timestamptz not null
      default now()
```

USE ENUMS

USE TIMESTAMPTZ

LET'S CREATE OUR TRACKS TABLE

```
track_data jsonb not null
```

NATIVE JSON & JSONB TYPES

LET’S CREATE OUR TRACKS TABLE

```
    user_id uuid not null references "user"(user_id),
```

THEN EMBRACE SQL POWER

*WITH A REAL SQL DATABASE LIKE POSTGRES

REQUEST JSON

```
select
    payload->>'date'
from a
```

USE SQL FUNCTIONS ON IT

```
select
  date_trunc(
    'month',
    payload->>'date'::timestamptz),
  count(*)
from a
group by
  date_trunc(
    'month',
    payload->>'date'::timestamptz);
```

TIME FUNCTIONS IN PGSQL

Function	Return Type	Description	Example	Result
age(timestamp, timestamp)	interval	Subtract arguments, producing a "symbolic" result that uses years and months, rather than just days	age(timestamp '2001-04-10', timestamp '1957-06-13')	43 years 9 mons 27 days
age(timestamp)	interval	Subtract from current_date (at midnight)	age(timestamp '1957-06-13')	43 years 8 mons 3 days
clock_timestamp()	timestamp with time zone	Current date and time (changes during statement execution); see Section 9.9.4		
current_date	date	Current date; see Section 9.9.4		
current_time	time with time zone	Current time of day; see Section 9.9.4		
current_timestamp	timestamp with time zone	Current date and time (start of current transaction); see Section 9.9.4		
date_part(text, timestamp)	double precision	Get subfield (equivalent to extract); see Section 9.9.1	date_part('hour', timestamp '2001-02-16 20:38:40')	20
date_part(text, interval)	double precision	Get subfield (equivalent to extract); see Section 9.9.1	date_part('month', interval '2 years 3 months')	3
date_trunc(text, timestamp)	timestamp	Truncate to specified precision; see also Section 9.9.2	date_trunc('hour', timestamp '2001-02-16 20:38:40')	2001-02-16 20:00:00
date_trunc(text, interval)	interval	Truncate to specified precision; see also Section 9.9.2	date_trunc('hour', interval '2 days 3 hours 40 minutes')	2 days 03:00:00
extract(field from timestamp)	double precision	Get subfield; see Section 9.9.1	extract(hour from timestamp '2001-02-16 20:38:40')	20
extract(field from interval)	double precision	Get subfield; see Section 9.9.1	extract(month from interval '2 years 3 months')	3
isfinite(date)	boolean	Test for finite date (not +/-infinity)	isfinite(date '2001-02-16')	true
isfinite(timestamp)	boolean	Test for finite time stamp (not +/-infinity)	isfinite(timestamp '2001-02-16 21:28:30')	true
isfinite(interval)	boolean	Test for finite interval	isfinite(interval '4 hours')	true
justify_days(interval)	interval	Adjust interval so 30-day time periods are represented as months	justify_days(interval '35 days')	1 mon 5 days
justify_hours(interval)	interval	Adjust interval so 24-hour time periods are represented as days	justify_hours(interval '27 hours')	1 day 03:00:00
justify_interval(interval)	interval	Adjust interval using justify_days and justify_hours, with additional sign adjustments	justify_interval(interval '1 mon -1 hour')	29 days 23:00:00
localtime	time	Current time of day; see Section 9.9.4		
localtimestamp	timestamp	Current date and time (start of current transaction); see Section 9.9.4		
make_date(year int, month int, day int)	date	Create date from year, month and day fields	make_date(2013, 7, 15)	2013-07-15
make_interval(years int DEFAULT 0, months int DEFAULT 0, weeks int DEFAULT 0, days int DEFAULT 0, hours int DEFAULT 0, mins int DEFAULT 0, secs double precision DEFAULT 0.0)	interval	Create interval from years, months, weeks, days, hours, minutes and seconds fields	make_interval(days := 10)	10 days
make_time(hour int, min int, sec double precision)	time	Create time from hour, minute and seconds fields	make_time(8, 15, 23.5)	08:15:23.5
make_timestamp(year int, month int, day int, hour int, min int, sec double precision)	timestamp	Create timestamp from year, month, day, hour, minute and seconds fields	make_timestamp(2013, 7, 15, 8, 15, 23.5)	2013-07-15 08:15:23.5
make_timestamptz(year int, month int, day int, hour int, min int, sec double precision, [timezone text])	timestamp with time zone	Create timestamp with time zone from year, month, day, hour, minute and seconds fields. When timezone is not specified, then current time zone is used.	make_timestamptz(2013, 7, 15, 8, 15, 23.5)	2013-07-15 08:15:23.5+01
now()	timestamp with time zone	Current date and time (start of current transaction); see Section 9.9.4		
statement_timestamp()	timestamp with time zone	Current date and time (start of current statement); see Section 9.9.4		
timeofday()	text	Current date and time (like clock_timestamp, but as a text string); see Section 9.9.4		
transaction_timestamp()	timestamp with time zone	Current date and time (start of current transaction); see Section 9.9.4		

COUNT(*) ON NOSQL?

GEO BASED REQUEST?

INDEX

```
create index myindex on a
  using GIN ((payload->'date'));
```

TRANSACTIONS

ACID

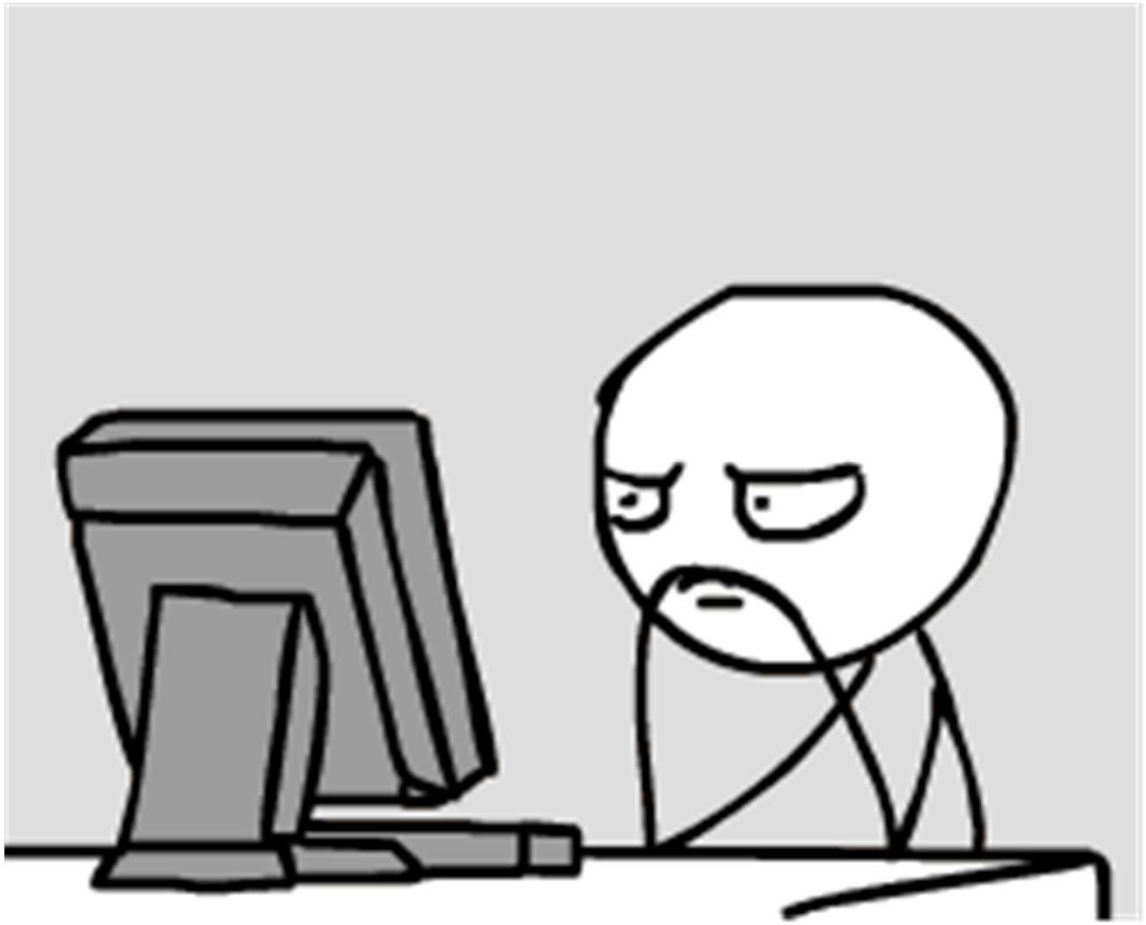
BOOTSTRAPPING?

“IS MY PLAIN OLD SQL THIS POWERFUL?”

CHOOSE THE RIGHT DB (AND THE RIGHT VERSION)

KILL THE ORM

JDBC DRIVERS



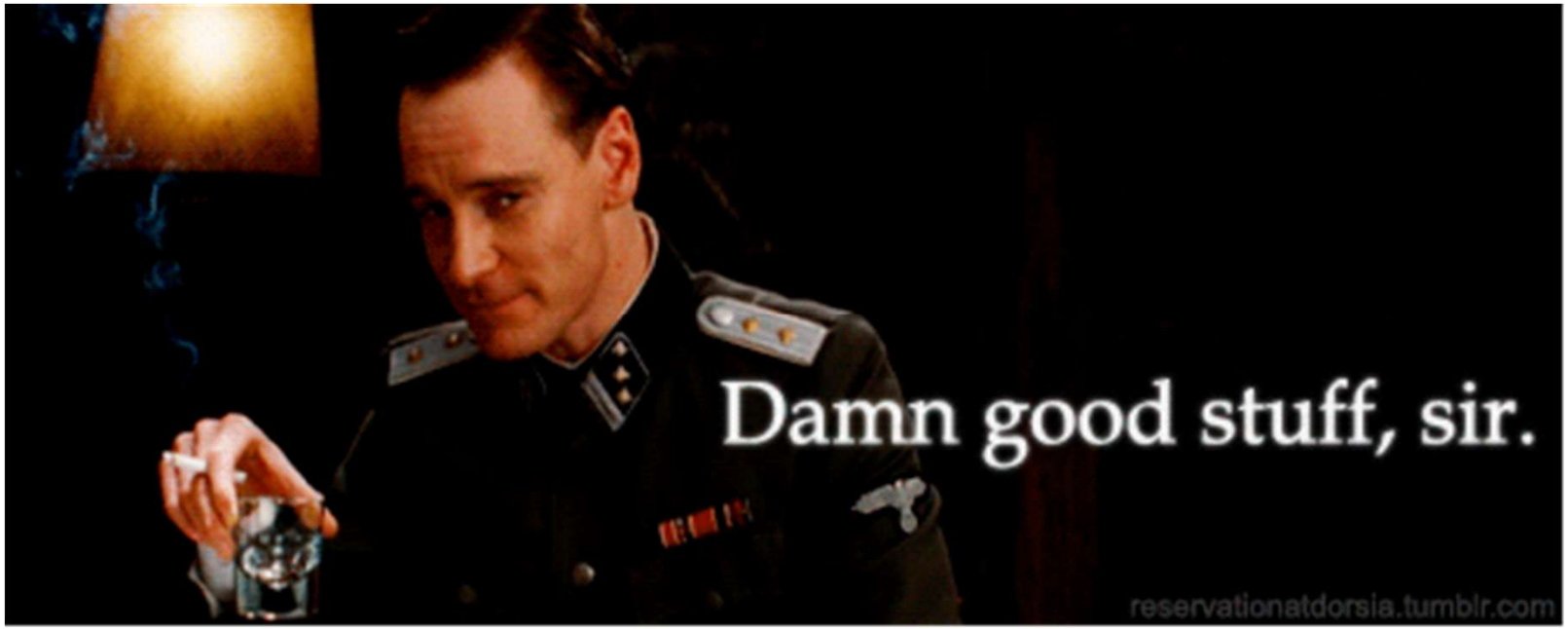
WORKAROUND

```
select
  array_to_json(
    array_agg(
      row_to_json(a.*)))
from a
group by
  date_trunc(
    'month',
    payload->>'date'::timestamptz);
```

WORKAROUND

```
[
    {
        "a_id": "f9675d56-7ce7-47c7-9f2d-5713f0cab364",
        "payload": {
            "date": "2015-03-12T15:10:05+0100",
            "value": "bar"
        }
    },
    …
]
```

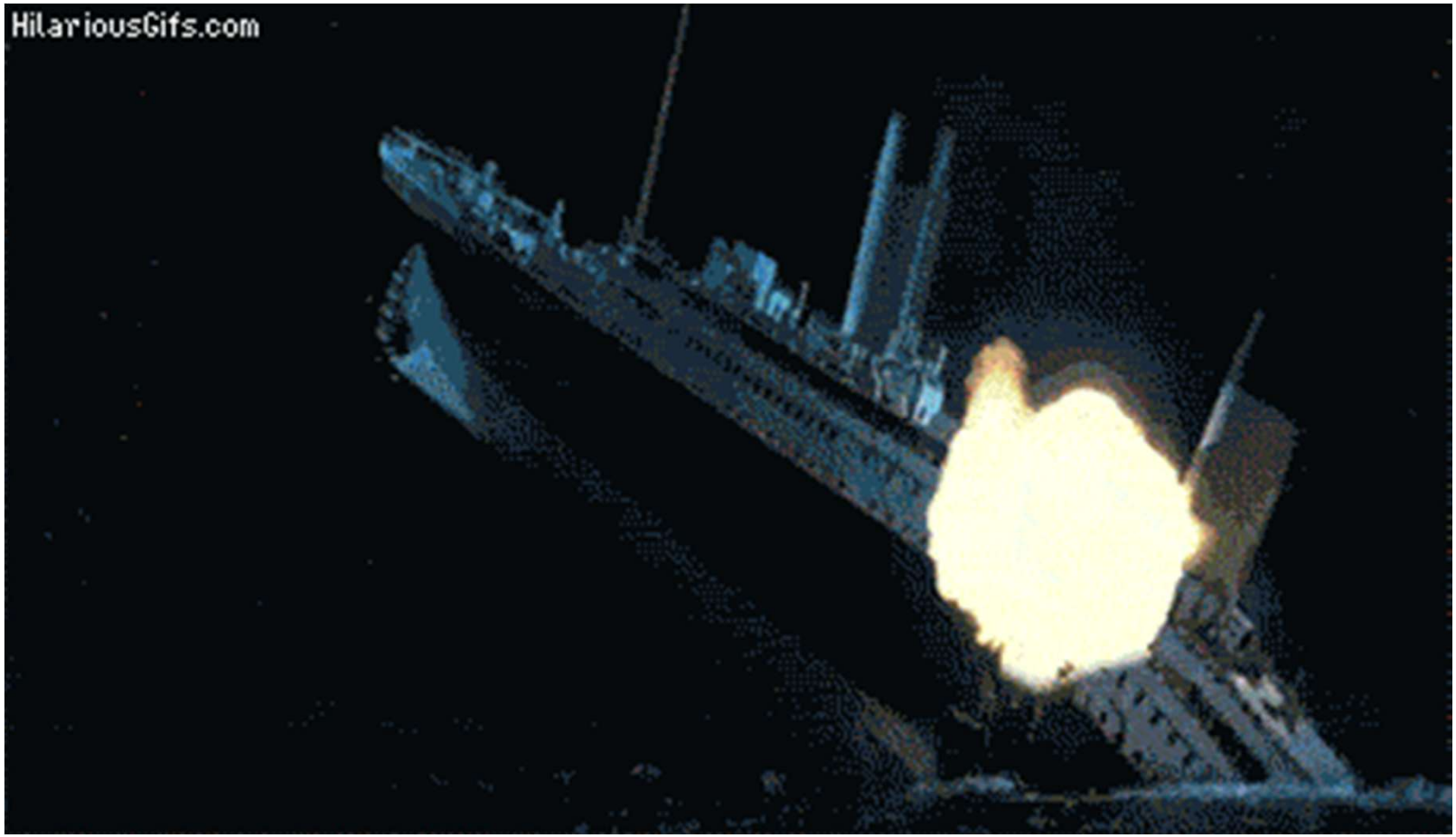
MANAGE IT LIKE REGULAR JSON IN YOUR CODE



FREE YOUR MIND ABOUT DATASTORES + DRIVER ACCESS

“SO IS PGSQL BETTER THAN DOCUMENT STORES?”

ATOMIC CHANGES INSIDE A JSON VALUE



CONCURRENT INSERTS

SHARDING & REPLICA

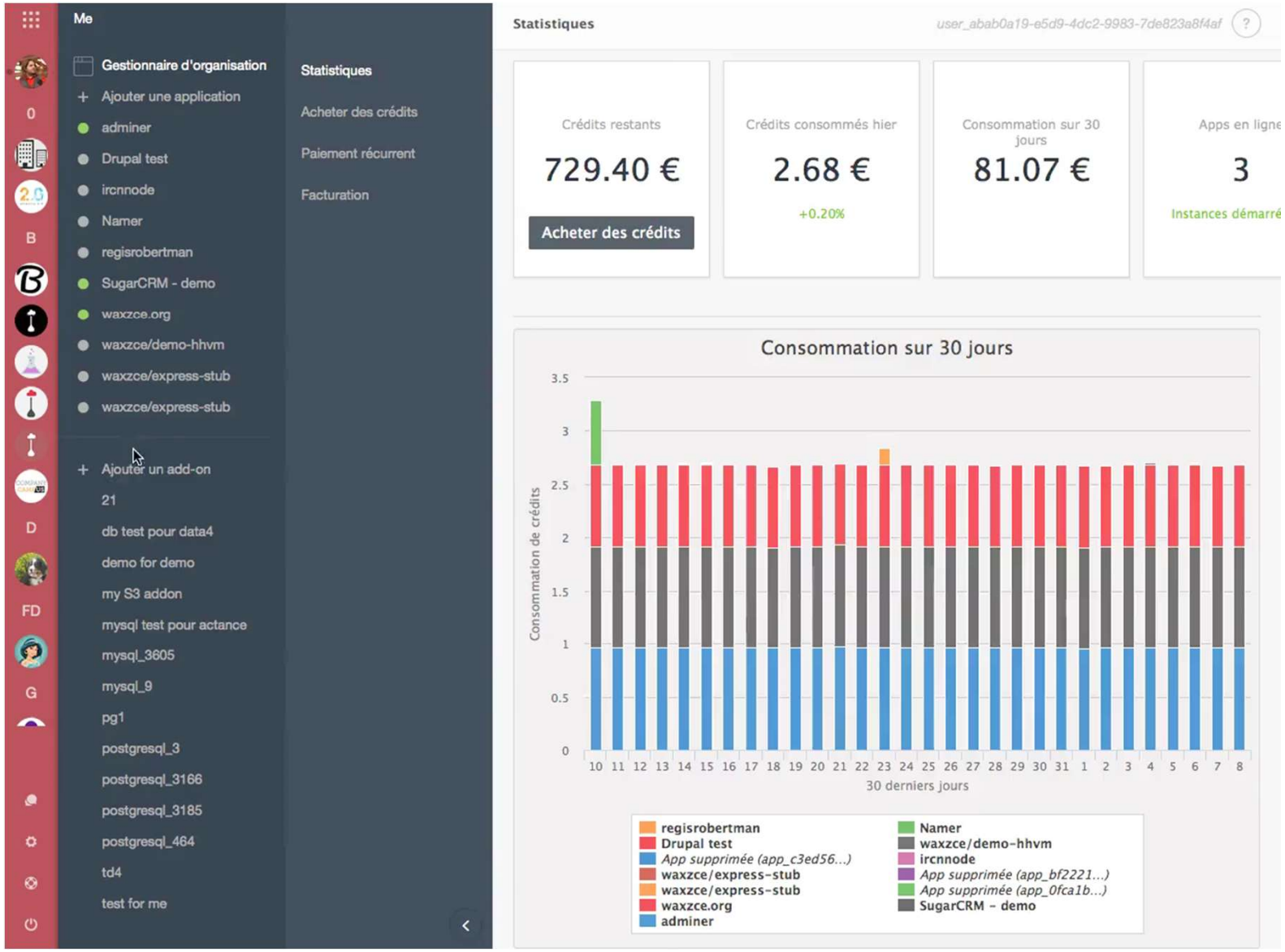
HIGH AVAILABILITY

CHOOSING A DATASTORE IS A REAL ARCHITECT JOB

TRY BEFORE YOU BUY :-)

FREE DEV PLAN ON CLEVER CLOUD

ONE CLICK LAUNCH



THX FOR LISTENING
& QUESTIONS TIME

I'm @clementd on twitter

I'm the CTO of

clever cloud

A PaaS provider, give it a try ;-)

Clever Cloud coupon : geecon15roxx