

You'd Better Have Tested Backups...

PostgreSQL Conference Europe, 2014

Dimitri Fontaine `dimitri@2ndQuadrant.fr`
`@tapoueh`

Oct 22, 2014



2ndQuadrant France PostgreSQL Major Contributor

- pgloader
- prefix, skytools
- apt.postgresql.org
- CREATE EXTENSION
- CREATE EVENT TRIGGER
- *Bi-Directional Réplication*
- pginstall



You'd Better Have Tested Backups...



In fact, backups are not interesting



Actually, automated recovery testing



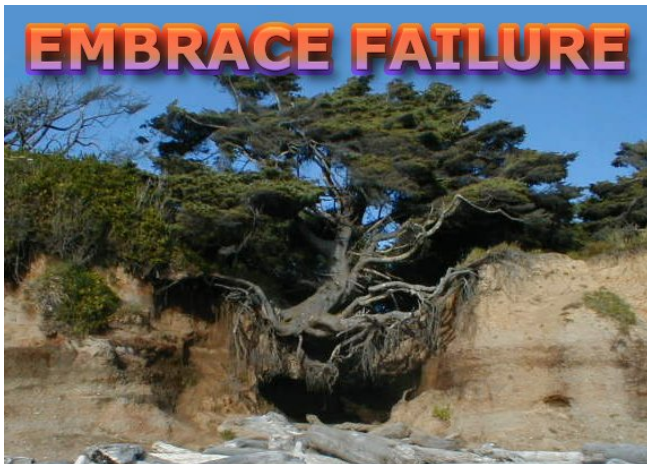
Use a battle tested tool

WHY USE BARMAN? Barman: disaster recovery for business critical PostgreSQL databases



Barman
Backup and recovery
manager for PostgreSQL

Today we're talking about what happens when you don't

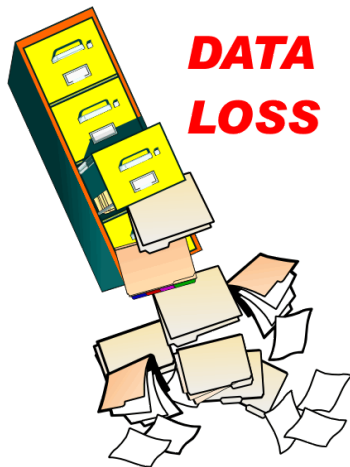


Backups? we have a shell script!



Shell Script, ENV, missing setup

```
find $PGDATA -mtime +5 | xargs rm -f
```



And now what?



Data recovery

Try to recover from what backups we do have:
only WAL files, no basebackup

- `pg_controldata` and `xlogdump`
- `initdb` then
- `hexedit pg_control`
- 5923145491842547187 to 52 33 3D 71 52 3B 3D F3
- then actually F3 3D 3B 52 71 3D 33 52
- play with the WAL we have and `pg_resetxlog`
- no luck this time, no working around missing WAL files



Back to having that POSTGRESQL running

First, **backup** the physical files
left available



Back to having that POSTGRESQL running

logs complain about pg_filenode.map

```
od -j 8 -N $((512-8-8)) -td4  
  < $PGDATA/global/pg_filenode.map
```



Back to having that POSTGRESQL running

logs complain about pg_clog

```
> (code-char #b01010101)
```

```
#\U
```

```
for c in 0000 0001 0002 0003 0004 0005 \  
        0006 0007 0008 0009 000A 000B 000C  
do  
    dd if=/dev/zero bs=256k count=1 | tr '\0' 'U' > $c  
done
```



Now POSTGRESQL starts.

But is complaining about missing pg_database mappings



Now POSTGRESQL starts.

But is complaining about missing pg_database mappings



How to provide for your own mapping?

```
select oid, relname, pg_relation_filenode(oid)
  from pg_class
 where relname = 'pg_database';
```

oid	relname	pg_relation_filenode
1262	pg_database	12319

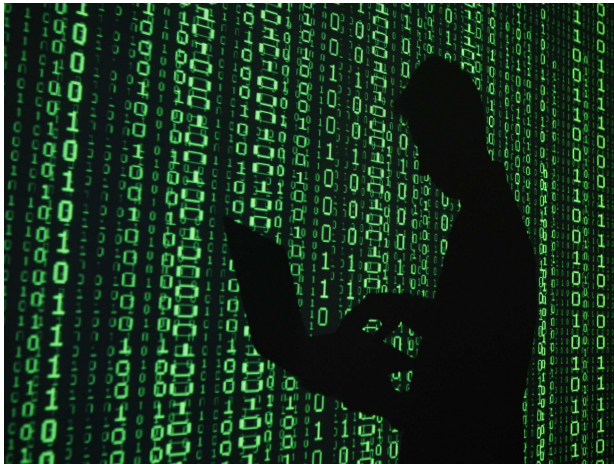
(1 row)

How to provide for your own mapping?

```
$ strings $PGDATA/global/12319  
postgres  
template0  
template1
```

How to provide for your own mapping?

CREATE DATABASE ... WITH OIDS ...;



Now we can connect to a database!

Only to hit a never seen before error message:

```
FATAL:  database "dbname" does not exists
DETAIL:  Database OID 17838 now seems to belong to "otherdb"
```



Start without the system indexes

As one does.

```
$ pg_ctl start -o "-P"  
$ cat > $PGDATA/postgresql.conf <<EOF  
enable_indexscan = off  
enable_bitmapscan = off  
enable_indexonlyscan = off  
EOF  
$ pg_ctl reload
```



Now we can query the catalogs!

psql and list tables, \dt
but
base/16384/12062 is missing

```
select oid, relname, pg_relation_filenode(oid)
from pg_class
where pg_relation_filenode(oid) = 12062;
oid | relname | pg_relation_filenode
```

```
-----+-----+-----
1255 | pg_proc | 12062
(1 row)
```



We lost the system catalogs...

Copy them over from a fresh `initdb` system.

Unless you did use some extensions...



Missing pg_namespace

But the application is using *custom* schemas.



How is pg_namespace stored?

```
select oid, * from pg_namespace;
```

oid	nspname	nspowner	nspacl
99	pg_toast	10	
11222	pg_temp_1	10	
11223	pg_toast_temp_1	10	
11	pg_catalog	10	{dim=UC/dim,=U/dim}
2200	public	10	{dim=UC/dim,=UC/dim}
11755	information_schema	10	{dim=UC/dim,=U/dim}

(6 rows)



Figuring out the file content

```
# copy pg_namespace to stdout with oids;
99      ↗pg_toast          ↗10      ↗\N
11222   ↗pg_temp_1        ↗10      ↗\N
11223   ↗pg_toast_temp_1 ↗10      ↗\N
11      ↗pg_catalog        ↗10      ↗{dim=UC/dim,=U/dim}
2200    ↗public ↗10        ↗{dim=UC/dim,=UC/dim}
11755   ↗information_schema ↗10      ↗{dim=UC/dim,=U/dim}
```



Add our namespaces live, with the right OIDs

```
# copy pg_namespace from stdin with oids;
```

Enter **data to** be copied followed **by** a newline.

End with a backslash **and** a period **on a line by** itself.

```
>> 16443          my_namespace      10      \N
```

```
>> \.
```



Wait, where the OID is coming from?

```
# select c.oid, relname, relnamespace, nspname
   from      pg_class c
      left join pg_namespace n
            on n.oid = c.relnamespace
 where relname = 'bar';
```

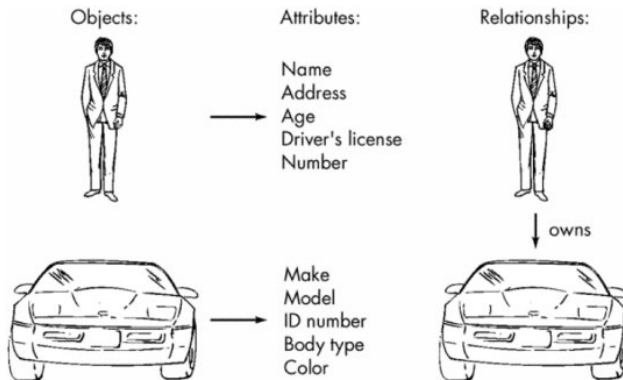
oid	relname	relnamespace	nspname
16446	bar	16443	

(1 row)



Now we can query the catalogs

But what we want is the data, not the metadata.



We are lucky here!

We didn't lose pg_attribute, only pg_attrdef

```
# \d a
```

Table "public.a"

Column	Type	Modifiers
id	integer	not null default nextval('a_id_seq'::regcl
f1	text	

Indexes:

"a_pkey" PRIMARY KEY, btree (id)



What's pg_attrdef like already?

```
# select adrelid, adnum, adsrc
   from pg_attrdef
   where adrelid = 'public.a'::regclass;
 adrelid | adnum |          adsrc
```

```
-----+-----+-----
 16411 |      1 | nextval('a_id_seq'::regclass)
(1 row)
```



What's pg_attrdef like already?

```
# select attnum, atthasdef
   from pg_attribute
  where attrelid = 'public.a'::regclass
        and atthasdef;
attnum | atthasdef
-----+-----
      1 | t
(1 row)
```

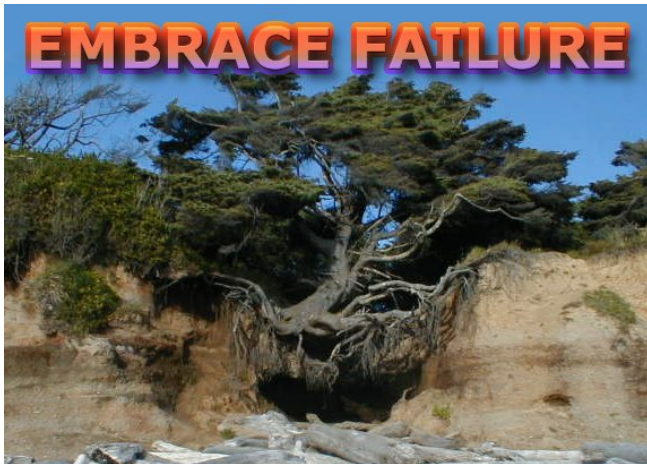


We are not creating new data in that instance, right?

```
# update pg_attribute
  set atthasdef = false
where attrelid = 'my_namespace.bar';
```



POSTGRESQL is amazingly resilient



You should have a proper *recovery plan*.

WHY USE BARMAN? Barman: disaster recovery for business critical PostgreSQL databases



Barman
Backup and recovery
manager for PostgreSQL

Questions?

Now is the time to ask!

