Patroni: PostgreSQL HA nel cloud

Lucio Grenzi
l.grenzi@gmail.com

Who I am

- Delphi developer since 1999
- IT Consultant
- Front end web developer
- Postgresql addicted


Nonantolando.blogspot.com


lucio.grenzi


lucio grenzi

Agenda

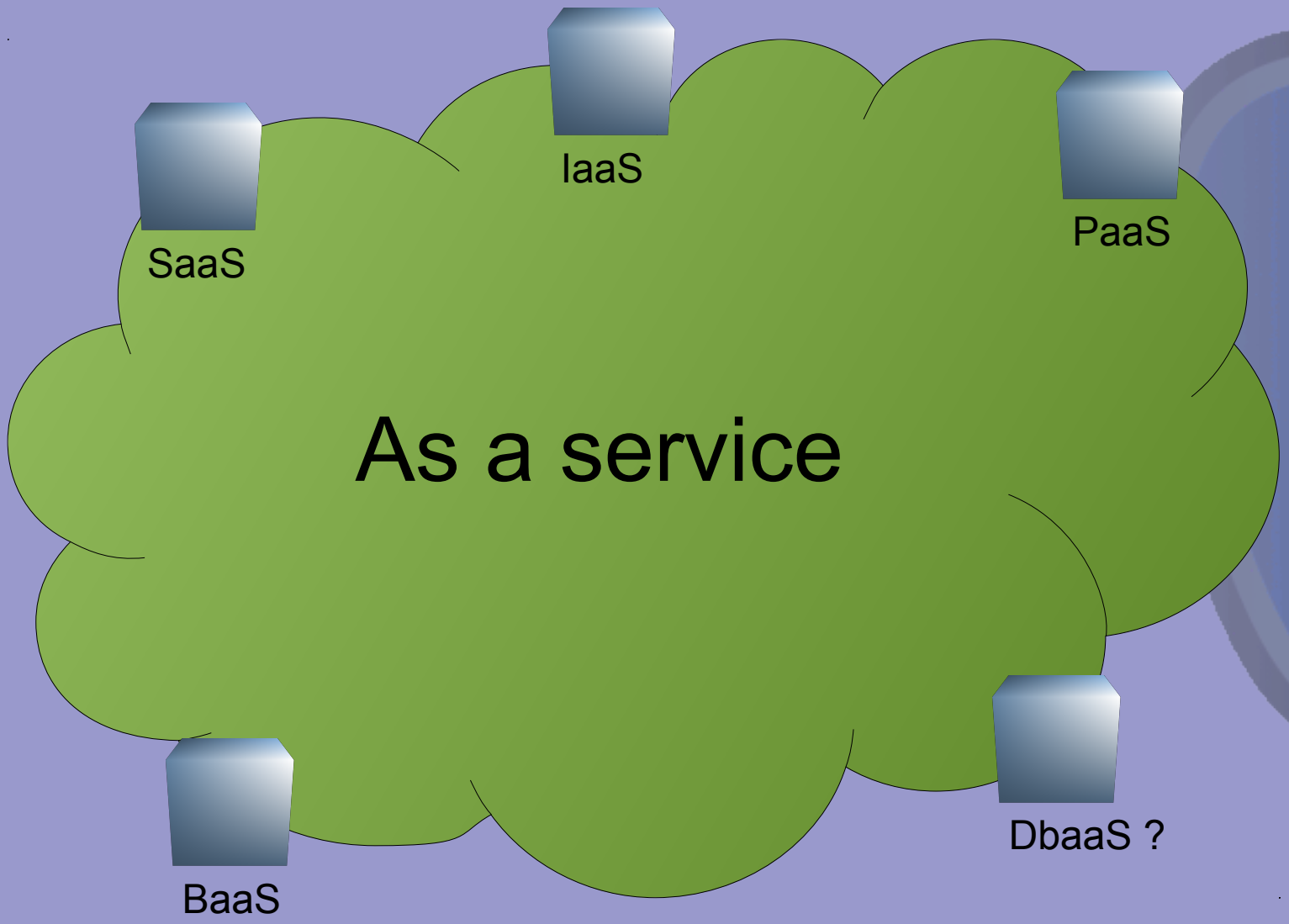
- Enterprise needs: high availability
- Cloud Databases?
- Patroni
- Data Synchonization: scalability vs performances
- Conclusion and final thoughts

Mission critical database

- High available
- Replication
- No human interaction for failover
- Rapid deployment

https://www.flickr.com/photos/my_public_domain_photos/15400918696/

As a service



Cloud databases

- Rapid deployment
- Scalability
- Provider's infrastructure optimization

- Failover?
- Flexibility?
- Security?

Master is down

- Manual failover
  - Pg_rewind

```
$ pg_rewind --target-pgdata=/var/lib/pgsql/9.6/master \
   --source-server="port=5432 user=postgres dbname=postgres"
```

    - Require superuser access privileges

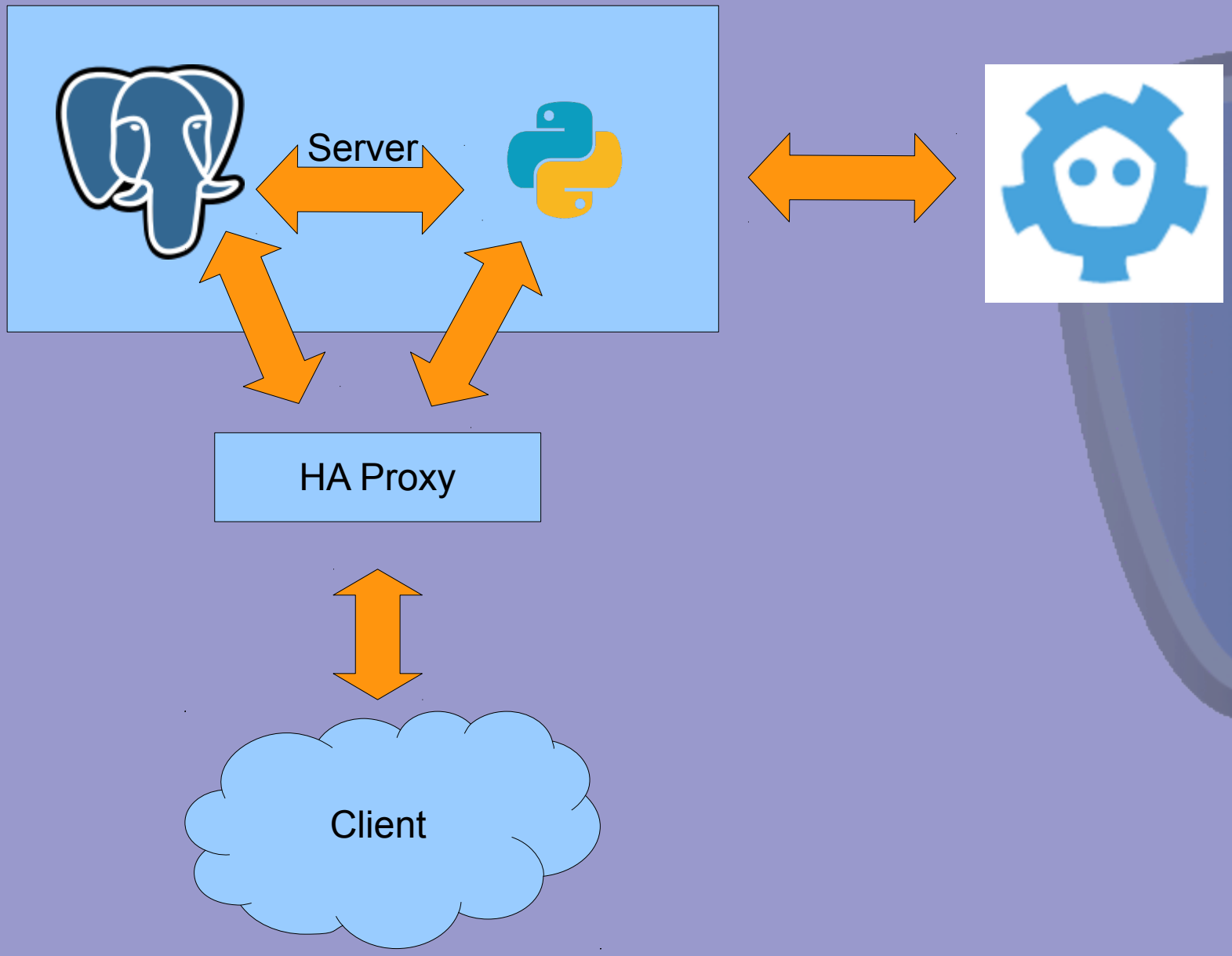
- Automatic failover
  - One supervisor node?
  - Distributed supervisor nodes?

Patroni: an introduction

Patoroni is a template for you to create your own customized, high-availability solution using Python and - for maximum accessibility - a distributed configuration store like ZooKeeper, etcd or Consul.

-https://github.com/zalando/patroni-

How Patroni works



Distributed Configuration System

- Distributed key/value store
- Like a directory tree
- The state machine is kept in sync using Raft
- Uses a discovery url
- json/rest api

Directory tree

```
$ etcdctl ls / --recursive

/service
/service/pgdayitcluster
/service/pgdayitcluster/initialize
/service/pgdayitcluster/config
/service/pgdayitcluster/leader
/service/pgdayitcluster/optime
/service/pgdayitcluster/optime/leader
/service/pgdayitcluster/members
/service/pgdayitcluster/members/dbnode2
/service/pgdayitcluster/members/dbnode3
/service/pgdayitcluster/members/dbnode1
```

Etcd members detail

```
 $ etcdctl get /service/testcluster/leader
dbnode2

$ etcdctl get /service/testcluster/members/dbnode2 | jq
{
  "role": "master",
  "state": "running",
  "conn_url": "postgres://172.17.0.3:5432/postgres",
  "api_url": "http://172.17.0.3:8008/patroni",
  "xlog_location": 67110528
}

$ etcdctl get /service/testcluster/members/dbnode1 | jq
{
  "role": "replica",
  "state": "running",
  "conn_url": "postgres://172.17.0.5:5432/postgres",
  "api_url": "http://172.17.0.5:8008/patroni",
  "xlog_location": 67110528
}
```

Replica

https://www.flickr.com/photos/roycin/4423082408/

Replication modes

- Patroni uses PostgreSQL streaming replication
- By default Patroni configures PostgreSQL for asynchronous replication.

Asynchronous mode

- Cluster is allowed to lose some committed transactions to ensure availability.
- When master server fails or becomes unavailable Patroni will automatically promote a sufficiently healthy standby to master.
- **maximum_lag_on_failover**

Synchronous replication

- It ensures consistency across a cluster by confirming that writes are written to a secondary before returning to the connecting client with a success.
- It is not guaranteed zero lost transactions under all circumstances.
- Add to the parameters section of your YAML configuration files:

```
synchronous_commit: "on"
synchronous_standby_names: "*"
```

Synchronous mode

- For use cases where losing committed transactions is not permissible
- Patroni will not promote a standby unless it is certain that the standby contains all transactions that may have returned a successful commit status to client

Synchronous mode implementation

- A node must be marked as the latest leader whenever it can accept write transactions.
- A node must be set as the synchronous standby in PostgreSQL as long as it is published as the synchronous standby.
- A node that is not the leader or current synchronous standby is not allowed to promote itself automatically.
- **synchronous_standby_names**

HAProxy

- Patroni provides an HAProxy configuration

  Used by your application in order to have a single endpoint for connecting to the cluster's leader

```
$ haproxy -f haproxy.cfg
```

```
$ psql --host 127.0.0.1 --port 5000 postgres
```

HAProxy.conf

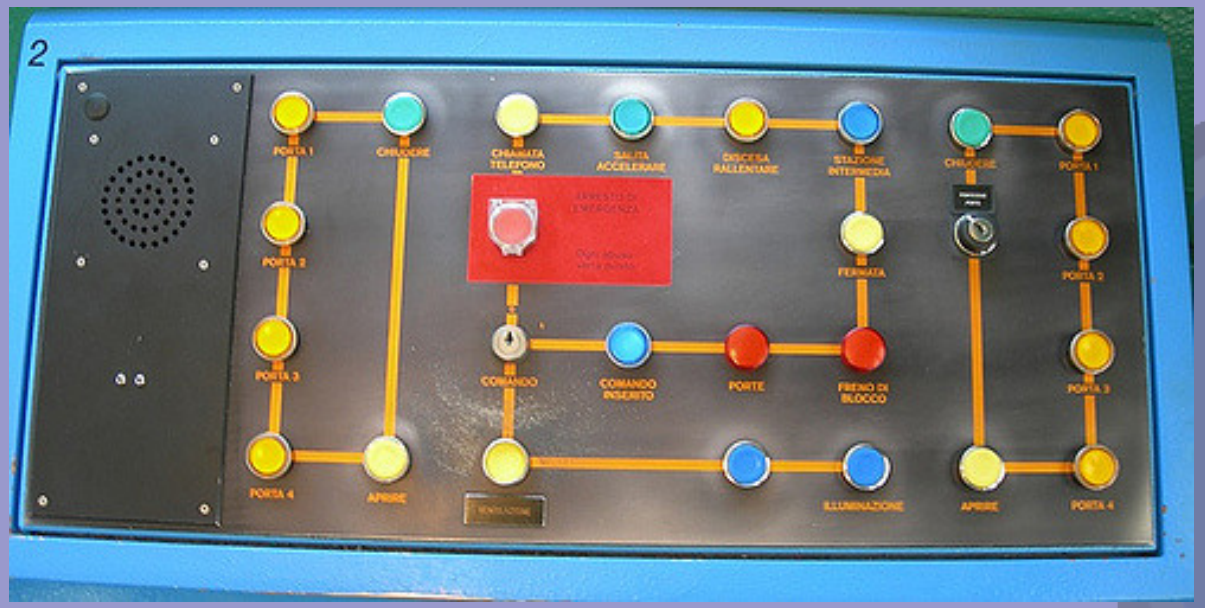
```
global
    maxconn 100

defaults
    log global
    mode tcp
    retries 2
    timeout client 30m
    timeout connect 4s
    timeout server 30m
    timeout check 5s

listen stats
    mode http
    bind *:7000
    stats enable
stats uri /

listen pgdayit
    bind *:5000
    option httpchk
    http-check expect status 200
    default-server inter 3s fall 3 rise 2 on-marked-down shutdown-sessions
    server postgresql_127.0.0.1_5432 127.0.0.1:5432 maxconn 100 check port 8008
    server postgresql_127.0.0.1_5433 127.0.0.1:5433 maxconn 100 check port 8009
```

Interfaces



https://www.flickr.com/photos/oldstretch/1051796285/

- API rest
- patronictl

API rest

- GET /config : Get current version of dynamic configuration

```
$ curl -s localhost:8008/config
```

- PATCH /config : Change parameters of an existing configuration

```
curl -s -XPATCH -d \
      '{"retry_timeout":5,"postgresql":{"parameters":{""max_wal_senders": “5”}}}'
```

API rest

- PUT /config : full rewrite of an existing dynamic

```
$ curl -s -XPUT -d \
      '{"maximum_lag_on_failover":1048576,
        "retry_timeout":10,
        "postgresql":
         {
         "use_slots":true,
         "use_pg_rewind":true,
         "parameters":
          {
          "hot_standby":"on",
          "wal_log_hints":"on",
          "wal_keep_segments":8,
          "wal_level":"hot_standby",
          "unix_socket_directories":".",
          "max_wal_senders":5
          }
        },
         "loop_wait":3,"ttl":20
      }' http://localhost:8008/config | jq
```

API rest

- POST /reload: change patroni.yml and reload at runtime wtihout stop any service

```
$ curl -s localhost:8008/reload
```

Patronictl

```
Usage: patronictl.py [OPTIONS] COMMAND [ARGS]...

Options:
  -c, --config-file TEXT  Configuration file
  -d, --dcs TEXT          Use this DCS
  --help                  Show this message and exit.

Commands:
  configure    Create configuration file
  dsn          Generate a dsn for the provided member,...
  edit-config  Edit cluster configuration
  failover     Failover to a replica
  flush        Flush scheduled events
  list         List the Patroni members for a given Patroni
  pause        Disable auto failover
  query        Query a Patroni PostgreSQL member
  reinit       Reinitialize cluster member
  remove       Remove cluster from DCS
  restart      Restart cluster member
  resume       Resume auto failover
  scaffold     Create a structure for the cluster in DCS
  show-config  Show cluster configuration
```

Patronictl failover

- Manual failover
- Promote a new master

```
$ ./patronictl.py failover <clustername>
 Master [dbnode2]:
 Candidate ['dbnode1', 'dbnode3'] []:
When should the failover take place (e.g. 2015-10-01T14:30)  [now]:
Are you sure you want to failover cluster testcluster, demoting current master dbnode2? [y/N]:
```

Spilo

- a Docker image that provides PostgreSQL and Patroni bundled together
- Multiple Spilos can create a resilient High Available PostgreSQL cluster
- You'll need to setup Spilo to create a database and roles for your application

```
$ psql -h mypgdaydb -p 5432 -U admin -d postgres
```

Risorse

- Patroni: https://github.com/zalando/patroni
- Spilo: https://github.com/zalando/spilo
- Etcd: https://github.com/coreos/etcd
- Raft: https://raft.github.io/raft.pdf

Questions?


Thanks...