PostgreSQL Automation Made Easy

pglift

Table des matières

About me

- Damien Clochard
- Co-founder of Dalibo
- Active member of PostgreSQLFr Association
- Main developer of PostgreSQL Anonymizer

Dalibo

- Open Source company
- Worker cooperative (SCOP)
- 20 years of PostgreSQL expertise
- We're hiring !

Why is PostgreSQL Automation so hard ?

- A new Postgres major version every year
- The full Postgres stack is complex
- SELinux is hard to implement
- Idempotency is a nightmare

---

pglift

- Python library, independent from automation tools
- Works with all major version of Postgres
- Works on RHEL / Debian / Ubuntu

An interface between two worlds

- A modern CLI
- A collection of Ansible modules
- Possibly more interfaces in the future (k8s, Terraform/OpenTofu)

The project

- First commit in 2021 / In v1.x since 2023
- Key component of the « Dalibo Postgres Platform » product
- Fully open-source, No Enterprise Edition
- **Transparent** development (issues, MRs and exchanges on GitLab)

https://gitlab.com/dalibo/pglift/

Contributors

- Visible : (commits): **Dalibo members**
- Indirect : **Dalibo members (“integrators”)**
- Invisible : our **clients** / **users**

Components and scopes

> An “instance”, in pglift terminology, is composed of a **PostgreSQL cluster complemented by a number of satellite components** providing additional services such as backup or monitoring.

The Stack

- Adminstration : temBoard
- Backups : pgbackrest
- Monitoring : prometheus postgres exporter
- HA : Patroni
- Perf : PoWA
- …

Security

- Services are launched with systemd user mode
- No need to be root

- No need to use sudo
- SELinux works out of the box

Configuration

- **Highly Configurable**
- behavior guided by **Dalibo's expertise** or **upstream**.
- A global YAML configuration file (`settings.yaml`)
- Environment variables
- Config file templates (`postgresql.conf`, `pgbackrest.conf`, ...)

Create an instance

```
$ pglift instance create foo --version 14 --port 5224
```

Create a standby node

```
pglift instance create foo_standby \
        --standby-for "host=foo.prod port=5224 user=replication" \
        --standby-password

Password for the replication user:
Repeat for confirmation:
```

List the instances

```
$ pglift instance list
```

Instance info

```
$ pglift instance get foo --output-format json
{
  "name": "foo",
  "version": "15",
  "standby": null,
  "port": 5224,
  "settings": {
    "shared_buffers": "8 GB",
...
}
```

Exec and Shell env

Connect with exec

```
$ pglift instance exec foo -- psql
...
[14/foo] postgres@~=#
```

or connect with a shell env

```
$ pglift instance shell foo
$ psql
...
[14/foo] postgres@~=#
```

Modify the config

```
$ pglift pgconf --instance foo set 'shared_buffers=3 GB'
```

or

```
$ pglift pgconf --instance foo edit
```

Clone a database

```
$ pglift database create foo_db \
        --clone-from "postgresql://postgres@prod_srv:5224/foo_db"
```

Major Upgrade

```
$ pglift instance upgrade 14/foo --version 16
```

Setup a HA cluster

```
$ pglift instance create foo \
        --patroni-cluster=foocluster \
        --patroni-node=foo1
```

Ansible - pglift modules

pglift offers an interface (collection of modules) for Ansible, it allows management of these objects:

- database
- dsn_info
- instance (Postgres + components)
- postgres_exporter
- role (in PostgreSQL)

Available on Galaxy

```
ansible-galaxy collection install dalibo.pglift
```

Describe an instance

```yaml
- name: my foo instances
  hosts: all
  become: true
  become_user: postgres
  tasks:
    - name: production instance
      dalibo.pglift.instance:
        name: foo
        state: started
        port: 5224
        # [...]
```

Run !

```
ansible-playbook -i inventory foo.yaml
```

Alter the shared buffers

```
- name: production instance
  dalibo.pglift.instance:
    name: foo
    # [...]
    restart_on_changes: true
    settings:
      max_connections: 150
      shared_buffers: 2GB
      unix_socket_directories: /tmp
```

Run Again !

```
ansible-playbook -i inventory foo.yaml
```

Declare a Database

```
- name: A database named pagila, owned by bob
  dalibo.pglift.database:
    instance: foo
    name: pagila
    owner: bob
    settings:
      work_mem: 3MB
```

Build a HA cluster

```yaml
- name: A foo instance on each host, managed by Patroni
  dalibo.pglift.instance:
  name: foo
  version: 14
  [...]
  patroni:
    cluster: foo-cluster
    node: ""
    postgresql:
      connect_host: ""
    restapi:
      connect_address: ":8008"
      listen: ":8008"
```

In a nutshell

- Operations are consistent wether you use the CLI or Ansible
- Idempotency !!!!!
- Rootless by default
- Continuous Developments
- Battle tested in large scale production environments
- Easy to integrate with AAP / Ansible Tower

Don’t Trust Me !

Happy Customer Feedback, by MAIF

https://www.youtube.com/watch?v=hmpytry9vqo

Feedback

Try it out !

Let us know how we can improve it

https://pglift.readthedocs.io/

Credits

Special thanks to Julian Vanden Broeck !

https://www.youtube.com/watch?v=Qd1qIJe9tS8

Questions ?

Let’s meet at the DALIBO booth !