PostgreSQL DBA's Troubleshooting Toolkit: Unraveling Complex Issues with Expertise.

Lalit Choudhary

Database Engineer @PERCONA

About Percona and Me

PERCC

open source database software, support, and services company

MySQL

mongoDB

PERCONA for MySQL

PERCONA for PostgreSQL

PERCONA for MongoDB

PERCONA Monitoring and Management

PERCONA Kubernetes Operators

PERCONA Toolkit

PERCONA

Supported Databases and Technologies

Databases

Platforms

Fully compatible with most common platforms

Hyperscalers

3

Cloud-native

Why DBA needs testing and monitoring tools ?

- Can't test on production.
- Experiments (new version/feature/extension test/POC).
- Analysing impact/change (before implementing to Prod).
- Historic data analysis and patterns to optimize postgresql configuration.
- Real time data analysis sessions , locks, load, resource utilization.

Obstacles for testing setup

- Time consuming process get new test servers.
- Approvals for certain software installation.
- Access issue and approvals for security.
- Cost of VM/licencing (cloud/on-premises) $$$
- Time consuming setups, like pg install, replication , patroni cluster setup etc

Available platforms

- Utilizing open source tools, Containerization/virtualization tools.
- Virtualbox : Time consuming (VMWare platform, Oracle virtual box).
- LXD, docker : Less time consuming but need extra installation steps.
- Docker could be also used with systemd (build own image).

Anydbver Tool [https://github.com/ihanick/anydbver]

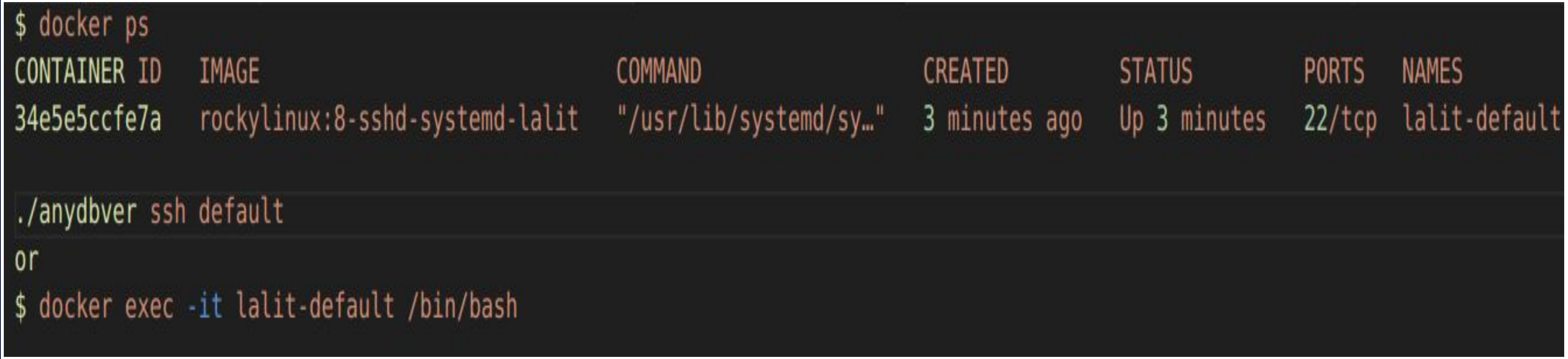
- Configuring MySQL, Percona MySQL/Postgresql/Mongo, MongoDB with ansible scripts. Running multi-node replication clusters in Docker, LXD and Kubernetes.
- (Developed and Maintained by Percona engineer)

PostgreSQL setup options:

- Standalone PostgreSQL server
- Postgresql Replication (Streaming Replication)
- Patroni Cluster
- K8 postgresql operator

Anydbver [Standalone PostgreSQL server setup]

```
./anydbver deploy os:el7 pg:15.5
./anydbver deploy os:el8 pg:14
./anydbver deploy node0 pg:latest
```

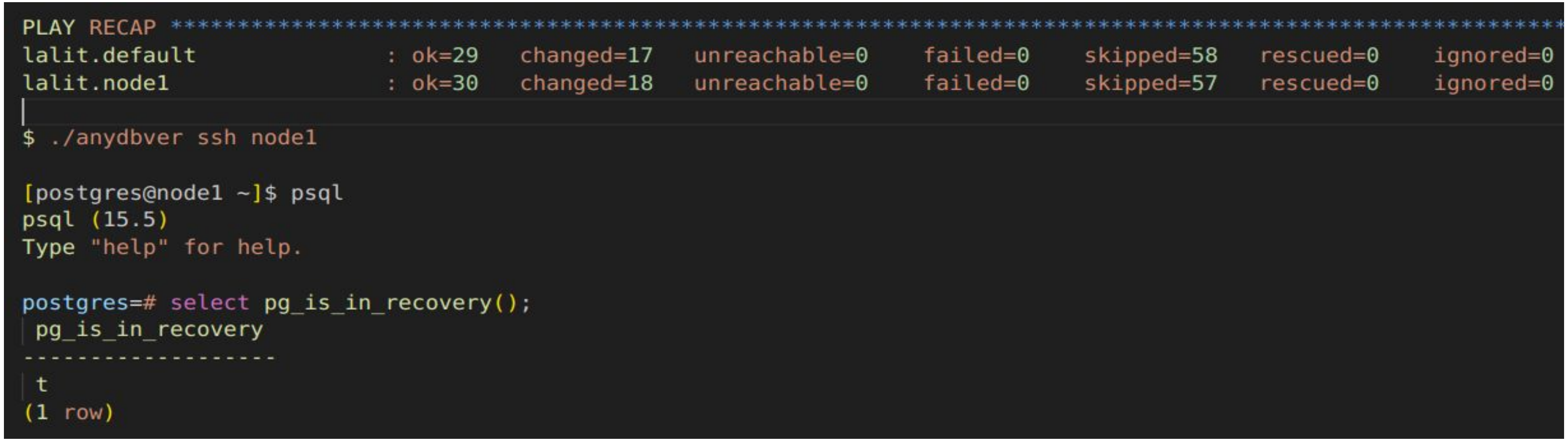


```
$ docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS    NAMES
34e5e5ccfe7a   rockylinux:8-sshd-systemd-lalit   "/usr/lib/systemd/sy…"   3 minutes ago   Up 3 minutes   22/tcp   lalit-default

./anydbver ssh default
or
$ docker exec -it lalit-default /bin/bash
```

Anydbver [Postgresql Replication (Streaming Replication)]

```
$ ./anydbver deploy node0 pg:latest,wal=logical node1 pg:latest,primary=node0,wal=logical
```

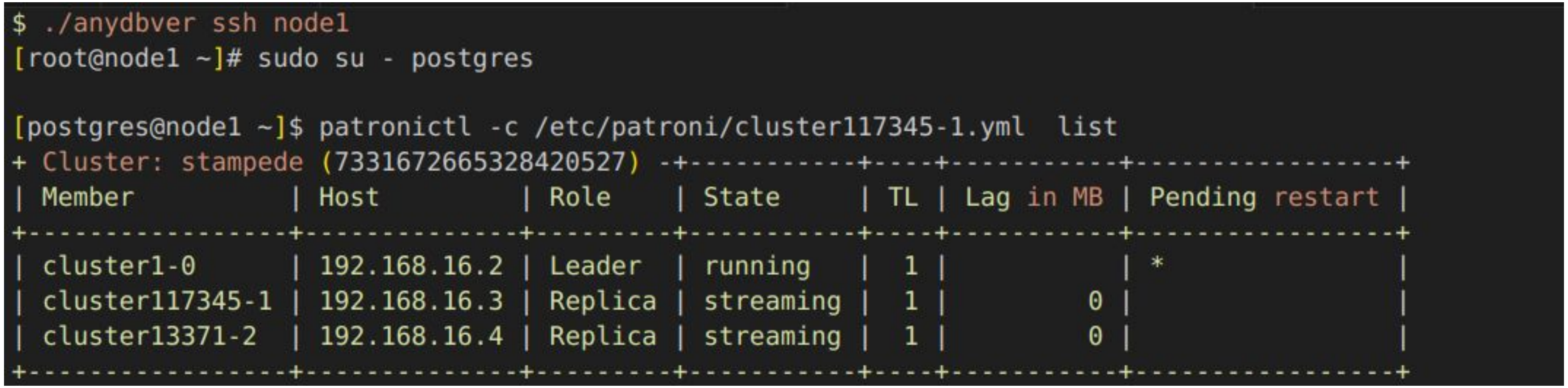
```
$ ./anydbver deploy node0 pg:15,wal=logical node1 pg:15,primary=node0,wal=logical
```



```
PLAY RECAP *******************************************************************************
lalit.default              : ok=29   changed=17   unreachable=0    failed=0    skipped=58   rescued=0    ignored=0
lalit.node1                : ok=30   changed=18   unreachable=0    failed=0    skipped=57   rescued=0    ignored=0

$ ./anydbver ssh node1

[postgres@node1 ~]$ psql
psql (15.5)
Type "help" for help.

postgres=# select pg_is_in_recovery();
 pg_is_in_recovery
-------------------
 t
(1 row)
```

Anydbver [Patroni Cluster]

```
$ ./anydbver deploy pg patroni node1 pg:master=node0 patroni:master=node0 node2 pg:master=node0
patroni:master=node0
PLAY RECAP
**********************************************************************************************************************
lalit.default            : ok=35   changed=22   unreachable=0    failed=0    skipped=58   rescued=0    ignored=0
lalit.node1              : ok=36   changed=23   unreachable=0    failed=0    skipped=57   rescued=0    ignored=0
lalit.node2              : ok=36   changed=23   unreachable=0    failed=0    skipped=57   rescued=0    ignored=0
```

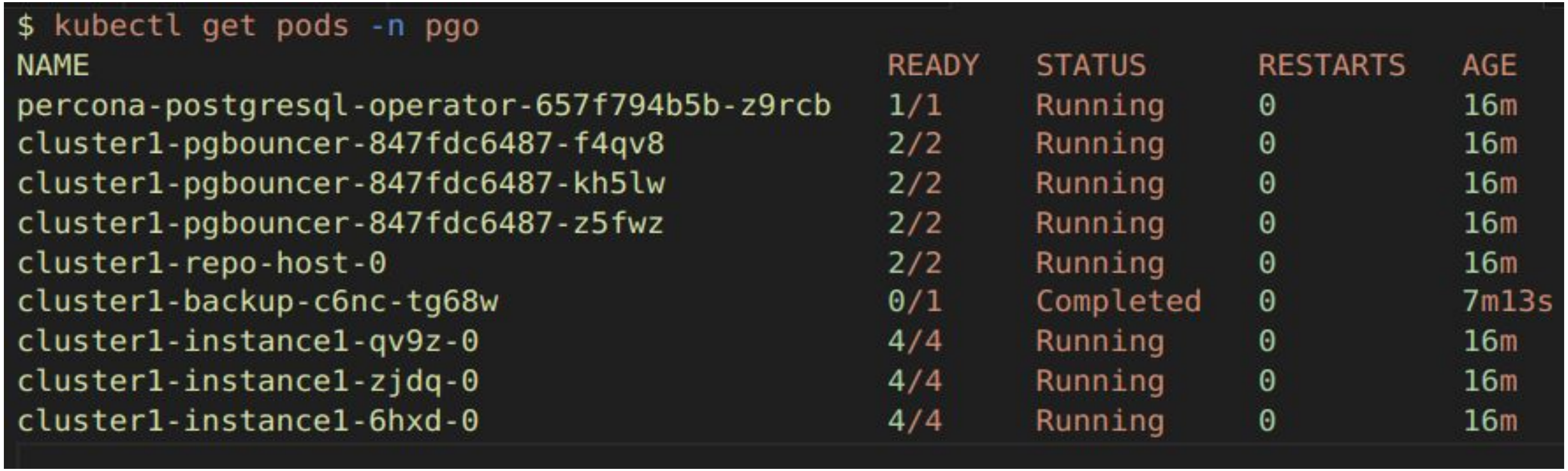


```
$ ./anydbver ssh node1
[root@node1 ~]# sudo su - postgres

[postgres@node1 ~]$ patronictl -c /etc/patroni/cluster117345-1.yml  list
+ Cluster: stampede (7331672665328420527) -+-----------+----+-----------+-----------------+
| Member           | Host         | Role    | State     | TL | Lag in MB | Pending restart |
+------------------+--------------+---------+-----------+----+-----------+-----------------+
| cluster1-0       | 192.168.16.2 | Leader  | running   |  1 |           | *               |
| cluster117345-1  | 192.168.16.3 | Replica | streaming |  1 |         0 |                 |
| cluster13371-2   | 192.168.16.4 | Replica | streaming |  1 |         0 |                 |
+------------------+--------------+---------+-----------+----+-----------+-----------------+
```

Anydbver [K8 operator]

https://operatorhub.io/?category=Database

```
$ ./anydbver deploy k3d:latest
$ ./anydbver deploy k3d:latest k8s-pg:2.2.0
```



```
$ kubectl get pods -n pgo
NAME                                          READY   STATUS      RESTARTS   AGE
percona-postgresql-operator-657f794b5b-z9rcb  1/1     Running     0          16m
cluster1-pgbouncer-847fdc6487-f4qv8           2/2     Running     0          16m
cluster1-pgbouncer-847fdc6487-kh5lw           2/2     Running     0          16m
cluster1-pgbouncer-847fdc6487-z5fwz           2/2     Running     0          16m
cluster1-repo-host-0                          2/2     Running     0          16m
cluster1-backup-c6nc-tg68w                    0/1     Completed   0          7m13s
cluster1-instance1-qv9z-0                     4/4     Running     0          16m
cluster1-instance1-zjdq-0                     4/4     Running     0          16m
cluster1-instance1-6hxd-0                     4/4     Running     0          16m
```

Anydbver [other options]

`namespace` usage for multiple setups:

```
$ ./anydbver --namespace pgrep_setup  deploy pg:15,docker-image

$ ./anydbver --namespace pgrep_setup destroy
```

Using existing k8 setup:

```
PROVIDER= kubectl
```

PMM Monitoring [Graphs]

Real time and Historic monitoring capabilities.

- PostgreSQL and OS monitoring
- Query Analytics: QAN (pg_stat_statements, pg_stat_monitor extension)
- Peak hours vs non-peak hours
- Resource usage over period of the time
- Other features like Alerting, custom graphs , etc

https://pmmdemo.percona.com/

Pg_gather [https://github.com/jobinau/pg_gather]

- Scan and collect the minimal amount of data needed to identify potential problems in your PostgreSQL database, and then generate an analysis report using that data.
- (Developed and Maintained by Percona engineer)
  - Everything is SQL-only, leveraging the built-in features of psql, the command-line utility of PostgreSQL.
  - Supported PostgreSQL Versions : 10, 11, 12, 13, 14, 15 & 16

Pg_gather

This project provides three SQL scripts for users:

- **gather.sql**: Gathers performance and configuration data from PostgreSQL databases.
- **gather_schema.sql:** Importing collected data
- **gather_report.sql**: Analyzes the collected data and generates detailed HTML reports.

An alternative is to use the `generate_report.sh` script, which can spin up a PostgreSQL Docker container and automate the entire process.

Pg_gather [Collect data , Generate report]

Collecting data:

```
 psql -d postgres -X -f gather.sql > out.tsv
```

Generating report:

```
psql -f gather_schema.sql -f out.tsv ## Creates unlogged tables pg_*
psql -X -f gather_report.sql > GatherReport.html ##Generates HTML report
```

Docker method:

```
./generate_report.sh out.tsv
```

PERCONA

Pg_gather [Examples and Use cases]

1. Table/index issues.
2. Vacuum related issues.
3. PostgreSQL Parameters review.
4. Sessions: Idle connections/transaction , Wait Events.
5. Sessions: Chain blockers.
6. Buffers/Flushing and summary.

Full Examples: https://github.com/lalitvc/pg_conf_2024_talk_ref/tree/main/pg_gather_use_cases_tests

PERCONA

Happy to chat more ..

Meet me at Percona Booth

PERCONA
Open source components and services to spin up a reliable production cluster for PostgreSQL
Certified PostgreSQL distribution
Kubernetes Operator for PostgreSQL
PostgreSQL monitoring and observability
24/7 support, managed services and consulting
Pre- and post-migration support and services
PERCONA
percona.com