14- stress testing database using pgbench

pgbench is tools that used to stress test database base on deferent scenarios you can define by simulating database traffic .

pgbench can help use also now our database setup limitations
and also get clear understand how our configuration will be able to tackle live use .

i personly use pgbench for two propose

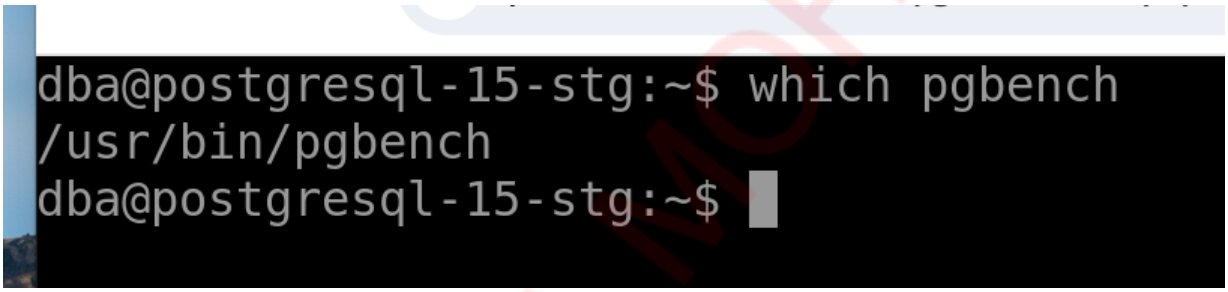
1- in my home lab to study performance tuning
2- for customer we will do stress test on this setup of HA or DR to determine its readiness before we handover the setup .

installing pgbench

pgbench come bundled with postgresql-contrib which one of core component of postgresql installation

if you have postgresql already installed , then you most properly have postgresql-contrib installed
to determine whether pgbench is installed use the `which` command

```
which pgbench
```



if pgbench was not installed then you can simply run below command to get it up and running

```
sudo apt install postgresql-contrib -y
```

using pgbench

for testing propose i like to create separate database for stress test ,
login to psql and create database

```
create database testdb'
```

```
postgres=# create database testdb;
CREATE DATABASE
postgres=# \l
                                                    List of databases
      Name      |  Owner   | Encoding |   Collate   |    Ctype    | ICU Locale | Locale Provider |   Access privileges
----------------+----------+----------+-------------+-------------+------------+-----------------+-----------------------
 Adventureworks | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            | =Tc/postgres         +
                |          |          |             |             |            |                 | postgres=CTc/postgres+
                |          |          |             |             |            |                 | docker=CTc/postgres
 postgres       | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            |
 prodcution     | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            |
 template0      | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            | =c/postgres          +
                |          |          |             |             |            |                 | postgres=CTc/postgres
 template1      | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            | =c/postgres          +
                |          |          |             |             |            |                 | postgres=CTc/postgres
 testdb         | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            |
(6 rows)

postgres=#
```

now we can start bench mark we will use the below command

```
pgbench -i -s 50 [database-name]
pgbench -i -s 50 testdb
```

```
dba@postgresql-15-stg:~$ sudo -u postgres pgbench -i -s 50 testdb
dropping old tables...
NOTICE:  table "pgbench_accounts" does not exist, skipping
NOTICE:  table "pgbench_branches" does not exist, skipping
NOTICE:  table "pgbench_history" does not exist, skipping
NOTICE:  table "pgbench_tellers" does not exist, skipping
creating tables...
generating data (client-side)...
5000000 of 5000000 tuples (100%) done (elapsed 11.30 s, remaining 0.00 s)
vacuuming...
creating primary keys...
done in 26.48 s (drop tables 0.00 s, create tables 0.01 s, client-side generate 11.43 s, vacuum 6.28 s, primary keys 8.77 s).
dba@postgresql-15-stg:~$
```

`-i` This option stands for "initialize." It tells `pgbench` to set up the benchmarking tables and data in the database.
`-s` his is the scale factor. The scale factor controls the size of the data set that `pgbench` creates. A scale factor of `50` means that `pgbench` will create tables and data that are 50 times larger than the default size

this means that database will be create 50 times size the default size which is 16MB

the number of record per table that will be genrated during bench will 5,000,000 records.

```
\list+
```

```
                                                                                                List of databases
      Name      |  Owner   | Encoding |   Collate   |    Ctype    | ICU Locale | Locale Provider |   Access privileges   |  Size   | Tablespace |
          Description
----------------+----------+----------+-------------+-------------+------------+-----------------+-----------------------+---------+------------+-----
----------------------------------------
 Adventureworks | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            | =Tc/postgres         +| 11 MB   | pg_default |
                |          |          |             |             |            |                 | postgres=CTc/postgres+|         |            |
                |          |          |             |             |            |                 | docker=CTc/postgres   |         |            |
 postgres       | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            |                       | 7453 kB | pg_default | defa
ult administrative connection database
 prodcution     | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            |                       | 2433 MB | pg_default |
 template0      | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            | =c/postgres          +| 7297 kB | pg_default | unmo
difiable empty database
                |          |          |             |             |            |                 | postgres=CTc/postgres |         |            |
 template1      | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            | =c/postgres          +| 7525 kB | pg_default | defa
ult template for new databases
                |          |          |             |             |            |                 | postgres=CTc/postgres |         |            |
 testdb         | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |            | libc            |                       | 755 MB  | pg_default |
(6 rows)
```

```
\dt+
```

```
~
postgres=# \c testdb
psql (15.7 (Ubuntu 15.7-1.pgdg22.04+1), server 15.8 (Ubuntu 15.8-1.pgdg22.04+1))
You are now connected to database "testdb" as user "postgres".
testdb=# \dt+
                                          List of relations
 Schema |       Name       | Type  |  Owner   | Persistence | Access method |  Size   | Description
--------+------------------+-------+----------+-------------+---------------+---------+-------------
 public | pgbench_accounts | table | postgres | permanent   | heap          | 641 MB  |
 public | pgbench_branches | table | postgres | permanent   | heap          | 40 kB   |
 public | pgbench_history  | table | postgres | permanent   | heap          | 0 bytes |
 public | pgbench_tellers  | table | postgres | permanent   | heap          | 56 kB   |
(4 rows)

testdb=#
```

threading and transaction stress test

now we will more real world test
with the `-j` option we can define number of thread that we want for our stress test
and with option `-t` we can specify number of transaction
`-c` we define number of client per transaction

this stress test is very useful to test setup such as pgpool
or patroni with load balance installed such haproxy
this command we will simulate a total of 100,000 transactions from 10 clients.

```
pgbench -c 10 -j 2 -t 10000 testdb
```

```
dba@postgresql-15-stg:~$ sudo -u postgres pgbench -c 10 -j 2 -t 10000 testdb
pgbench (15.8 (Ubuntu 15.8-1.pgdg22.04+1))
starting vacuum...end.
transaction type: <builtin: TPC-B (sort of)>
scaling factor: 50
query mode: simple
number of clients: 10
number of threads: 2
maximum number of tries: 1
number of transactions per client: 10000
number of transactions actually processed: 100000/100000
number of failed transactions: 0 (0.000%)
latency average = 12.065 ms
initial connection time = 26.564 ms
tps = 828.820866 (without initial connection time)
dba@postgresql-15-stg:~$
```

we can see from result that test seceded .

reference : https://medium.com/@c.ucanefe/pgbench-load-test-166bdfb5c75a