pg_repack

PostgreSQL users often face the challenge of database bloat, which can significantly impact performance. Traditional methods like `VACUUM FULL` and `CLUSTER` can be disruptive, but `pg_repack` offers a solution that works online, ensuring minimal downtime.

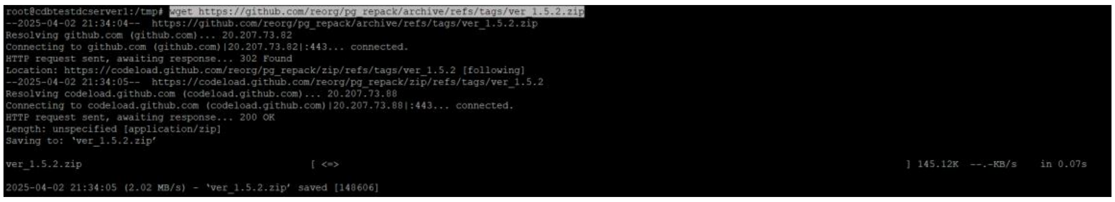
What is pg_repack?

`pg_repack` is a PostgreSQL extension designed to remove bloat from tables and indexes. Unlike `CLUSTER` and `VACUUM FULL`, `pg_repack` operates online without holding an exclusive lock on the processed tables, making it an efficient and less intrusive option.

Installation

Step 1: Download

wget https://github.com/reorg/pg_repack/archive/refs/tags/ver_1.5.2.zip

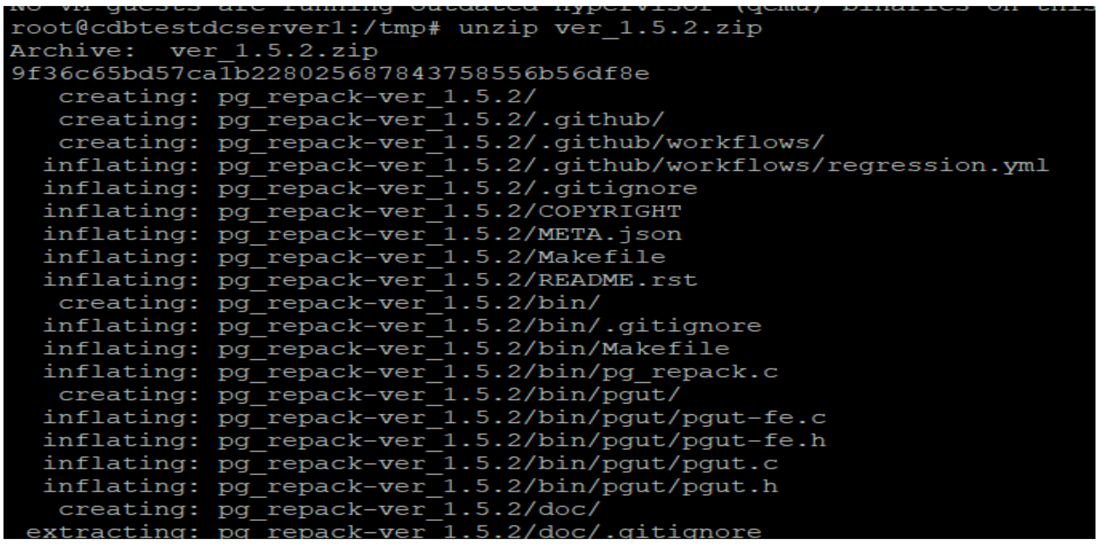


```
root@cdbtestdcserver1:/tmp# wget https://github.com/reorg/pg_repack/archive/refs/tags/ver_1.5.2.zip
--2025-04-02 21:34:04--  https://github.com/reorg/pg_repack/archive/refs/tags/ver_1.5.2.zip
Resolving github.com (github.com)... 20.207.73.82
Connecting to github.com (github.com)|20.207.73.82|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/reorg/pg_repack/zip/refs/tags/ver_1.5.2 [following]
--2025-04-02 21:34:05--  https://codeload.github.com/reorg/pg_repack/zip/refs/tags/ver_1.5.2
Resolving codeload.github.com (codeload.github.com)... 20.207.73.88
Connecting to codeload.github.com (codeload.github.com)|20.207.73.88|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/zip]
Saving to: 'ver_1.5.2.zip'

ver_1.5.2.zip                                          [ <=>                                                    ] 145.12K  --.-KB/s    in 0.07s

2025-04-02 21:34:05 (2.02 MB/s) - 'ver_1.5.2.zip' saved [148606]
```

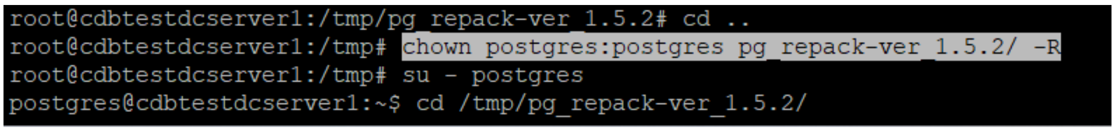
Step 2: unzip



```
No VM guests are running outdated hypervisor (qemu) binaries on this
root@cdbtestdcserver1:/tmp# unzip ver_1.5.2.zip
Archive:  ver_1.5.2.zip
9f36c65bd57ca1b228025687843758556b56df8e
   creating: pg_repack-ver_1.5.2/
   creating: pg_repack-ver_1.5.2/.github/
   creating: pg_repack-ver_1.5.2/.github/workflows/
  inflating: pg_repack-ver_1.5.2/.github/workflows/regression.yml
  inflating: pg_repack-ver_1.5.2/.gitignore
  inflating: pg_repack-ver_1.5.2/COPYRIGHT
  inflating: pg_repack-ver_1.5.2/META.json
  inflating: pg_repack-ver_1.5.2/Makefile
  inflating: pg_repack-ver_1.5.2/README.rst
   creating: pg_repack-ver_1.5.2/bin/
  inflating: pg_repack-ver_1.5.2/bin/.gitignore
  inflating: pg_repack-ver_1.5.2/bin/Makefile
  inflating: pg_repack-ver_1.5.2/bin/pg_repack.c
   creating: pg_repack-ver_1.5.2/bin/pgut/
  inflating: pg_repack-ver_1.5.2/bin/pgut/pgut-fe.c
  inflating: pg_repack-ver_1.5.2/bin/pgut/pgut-fe.h
  inflating: pg_repack-ver_1.5.2/bin/pgut/pgut.c
  inflating: pg_repack-ver_1.5.2/bin/pgut/pgut.h
   creating: pg_repack-ver_1.5.2/doc/
 extracting: pg_repack-ver_1.5.2/doc/.gitignore
```

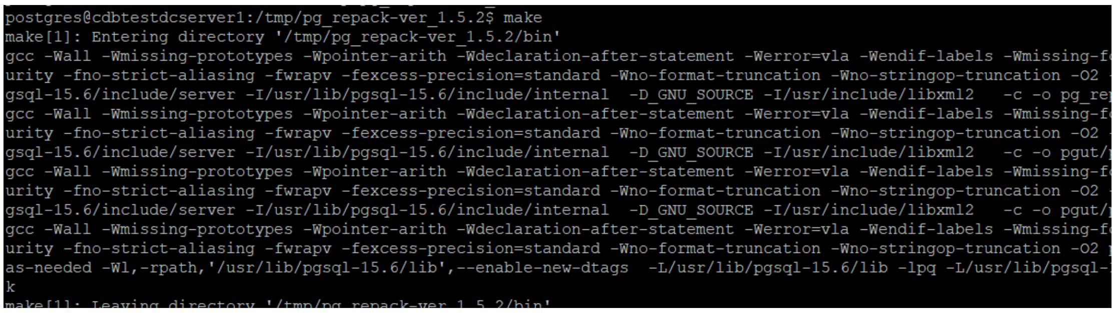
Step 3: change owner of directory

chown postgres:postgres pg_repack-ver_1.5.2/ -R



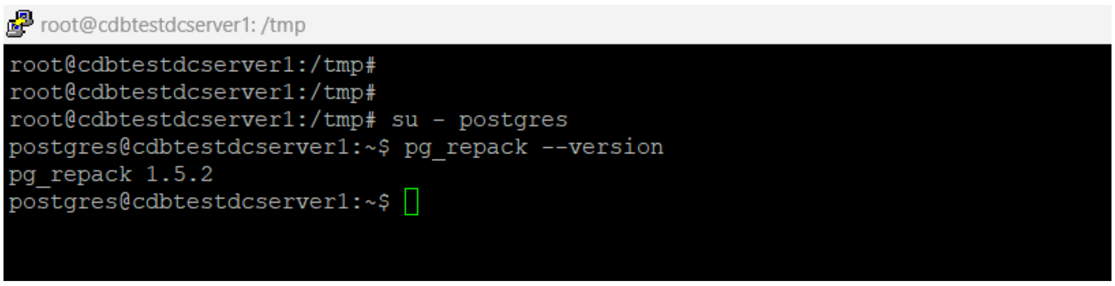
```
root@cdbtestdcserver1:/tmp/pg_repack-ver_1.5.2# cd ..
root@cdbtestdcserver1:/tmp# chown postgres:postgres pg_repack-ver_1.5.2/ -R
root@cdbtestdcserver1:/tmp# su - postgres
postgres@cdbtestdcserver1:~$ cd /tmp/pg_repack-ver_1.5.2/
```

Step 4: make && make install



```
postgres@cdbtestdcserver1:/tmp/pg_repack-ver_1.5.2$ make
make[1]: Entering directory '/tmp/pg_repack-ver_1.5.2/bin'
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-labels -Wmissing-f
urity -fno-strict-aliasing -fwrapv -fexcess-precision=standard -Wno-format-truncation -Wno-stringop-truncation -O2
gsql-15.6/include/server -I/usr/lib/pgsql-15.6/include/internal  -D_GNU_SOURCE -I/usr/include/libxml2   -c -o pg_re
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-labels -Wmissing-f
urity -fno-strict-aliasing -fwrapv -fexcess-precision=standard -Wno-format-truncation -Wno-stringop-truncation -O2
gsql-15.6/include/server -I/usr/lib/pgsql-15.6/include/internal  -D_GNU_SOURCE -I/usr/include/libxml2   -c -o pgut/
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-labels -Wmissing-f
urity -fno-strict-aliasing -fwrapv -fexcess-precision=standard -Wno-format-truncation -Wno-stringop-truncation -O2
gsql-15.6/include/server -I/usr/lib/pgsql-15.6/include/internal  -D_GNU_SOURCE -I/usr/include/libxml2   -c -o pgut/
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -Wendif-labels -Wmissing-f
urity -fno-strict-aliasing -fwrapv -fexcess-precision=standard -Wno-format-truncation -Wno-stringop-truncation -O2
as-needed -Wl,-rpath,'/usr/lib/pgsql-15.6/lib',--enable-new-dtags  -L/usr/lib/pgsql-15.6/lib -lpq -L/usr/lib/pgsql-
k
make[1]: Leaving directory '/tmp/pg_repack-ver_1.5.2/bin'
```

Step 5: Verify version



```
root@cdbtestdcserver1:/tmp#
root@cdbtestdcserver1:/tmp#
root@cdbtestdcserver1:/tmp# su - postgres
postgres@cdbtestdcserver1:~$ pg_repack --version
pg_repack 1.5.2
postgres@cdbtestdcserver1:~$
```

Configuration

Step 1: Create the Extension

Connect to your PostgreSQL database and create the `pg_repack` extension:

```
psql -c "CREATE EXTENSION pg_repack" -d employee
```

Step 2: Verify pg_repack Availability

To ensure `pg_repack` is available, you can list the PostgreSQL binaries:

```
pg_re
```

You should see `pg_repack` listed among the other PostgreSQL binaries:

```
pg_receivewal   pg_recvlogical  pg_repack       pg_resetwal     pg_restore      pg_rewind
# if you dont see, you may forget to source your bashrc (source ~/.bashrc)
```

Usage

Basic Usage

To repack a specific database (e.g., `employee`), use the following command:

```
pg_repack -d employee
```

You will see output similar to this:

```
INFO: repacking table "public.address"
INFO: repacking table "public.city"
INFO: repacking table "public.company"
...
```

Advanced Options

`pg_repack` offers several options to customize its behavior. Here are some useful ones:

- Repack all databases:

```
pg_repack -a
```

- Repack a specific table:

```
pg_repack -t tablename -d employee
```

- Repack tables in a specific schema:

```
pg_repack -c schemaname -d employee
```

- Move repacked tables to a new tablespace:

```
pg_repack -s newtablespace -d employee
```

- Order by specific columns:

```
pg_repack -o "column1, column2" -d employee
```

For a complete list of options, refer to the `pg_repack` help:

```
pg_repack --help
```

Conclusion

`pg_repack` is a powerful tool for managing bloat in PostgreSQL databases with minimal downtime. By following the steps outlined in this article, you can easily install and configure `pg_repack` on your PostgreSQL 14 setup running on RHEL 9. With `pg_repack`, you can maintain optimal database performance without the need for disruptive maintenance operations. For more detailed and technical articles like this, keep following our blog on Medium. If you have any questions or need further assistance, feel free to reach out in the comments below and directly.