Archiwizacja ciągła i przywracanie do puntu tuż przed awarią

W PostgreSQL działa mechanizm podobny do redo i archivelogów w Oracle. W PostgreSQL mowa o mechaniźmie WAL – Write Achead Log. Działa to tak :

1. Użytkownik zmienia jakieś dane
2. Informacja o zmianie trafia do plików WAL – w których przechowywane są informacje o zmianach na wypadek gdyby zmiany nie zostały utrwalone w plikach danych a nastąpiłoby np. odcięcie zasilania.
3. Zmiana danych następuje na poziomie bloków w buforze – od tej pory noszą miano brudnych bloków.
4. Kiedy następuje checkpoint brudne bloki są utrwalane na poziomie plików danych. Checkpoint następuje wtedy gdy upłynie czas określony w parametrze checkpoint_timeout, lub gdy skończy się miejsce w plikach WAL (domyślnie są to trzy pliki o rozmiarze 16MB każdy. Ich ilość określamy w parametrze checkpoint_segments).
5. Jeśli archiwizacja ciągła jest wyłączona (a jest wyłączona domyślnie) bieżące pliki WAL są kasowane z katalogu pg_xlog, a w ich miejsce powstają nowe o kolejnych numerach. Jeśli w wyniku np. długo wykonującego się backupu w katalogu pg_xlog powstanie więcej niż 3 pliki WAL taka ich ilość pozostanie i będzie rotowała, ale nadal w paczkach po tyle plików ile określimy w parametrze checkpoint_segments. Jeśli archiwizacja ciągła jest włączona to przed skasowaniem pliki WAL kopiowane są w miejsce archiwizacji które wskażesz w konfiguracji.

W punktach powyżej pojawia się pojęcie archiwizacji ciągłej. Co to takiego? Wyobraź sobie że zarządzasz dużym systemem którego baza danych oparta jest o PostgreSQL. Robisz backupy raz w tygodniu korzystając z pg_dump lub pg_dumpall. Czy to nam wystarczy? Nie zawsze. Jeśli zrobiłeś backup np. w niedzielę a w środę nastąpi awariaa to z takiego zrzutu odzyskasz stan bazy z niedzieli. Czy to się sprawdzi np. w banku? Dla klienta banku pewnie byłoby to korzystne, o ile backup był zrobiony tuż po wypłatach ;) Dla administratora i samego banku nieco mniej. W takiej sytuacji chcielibyśmy odzyskać stan bazy z momentu tuż przed awarią, a nie sprzed paru dni. Może też nastąpić taka sytuacja że w wyniku ludzkiego błędu będzie trzeba cofnąć stan bazy do wczoraj (a backup jest sprzed przykładowo 5 dni). W obu przypadkach mechanizm archiwizacji ciągłej będzie dla nas rozwiązaniem. Ok, ale jak to działa? Wyobraź sobie że masz kopię zapasową plików danych sprzed tygodnia, ale w ciągu tego tygodnia zbierałeś wszystkie pliki WAL jakie powstawały. Masz więc punkt wyjścia w postaci kopii plików danych i pliki WAL zawierające wszystkie instrukcje zmieniające cokolwiek w bazie od backupu do teraz. Mając taki zestaw możesz przywrócić pliki danych z kopii na pierwotne miejsce, a następnie powtórzyć wszystkie operacje z WAL lub ich część – do wskazanego punktu w czasie. Fajne? Pewnie, bo może nam uratować skórę. Jeśli nie włączysz archiwizacji ciągłej, przywracanie bazy w taki sposób nie będzie możliwe, ponieważ miałbyś backup sprzed jakiegoś czasu – np. kilku dni i pliki WAL zawierające informacje o zmianach z np. ostatniej godziny. Mielibyśmy kilka dni luki. Potrzebujemy całej historii operacji od backupu do teraz, by przywracać bazę do punktu w czasie. Aby włączyć archiwizację ciągłą i umożliwić sobie odtwarzanie baz do punktu w czasie musimy zmienić 3 parametry w pliku postgresql.conf

Edytujemy plik postgresql.conf:

```
bash-4.1$ whoami
postgres
bash-4.1$ nano /var/lib/pgsql/9.4/data/postgresql.conf
```

Odremowujemy parametr WAL_LEVEL i ustawiamy jego wartość na archive lub hot_standby. Jego domyślna wartość to minimal, która pozwala jedynie na odtworzenie bazy po awarii. Wartość archive powoduje dodawanie do WAL informacji które umożliwiają nam wykorzystanie archiwizacji ciągłej, a hot_standby pozwala na uruchamianie serwera w trybie hot_standby i puszczanie zapytań działających w trybie tylko do odczytu.

```
#------------------------------------------------------------------------------
# WRITE AHEAD LOG
#------------------------------------------------------------------------------

# - Settings -

wal_level = archive                     # minimal, archive, hot_standby, or logical
                                        # (change requires restart)
```

Odremowujemy teraz parametr ARCHIVE_MODE i ustawiamy jego wartość na ON. Umożliwia on uruchomienie archiwizacji.

```
# - Archiving -

archive_mode = on               # allows archiving to be done
                                # (change requires restart)
```

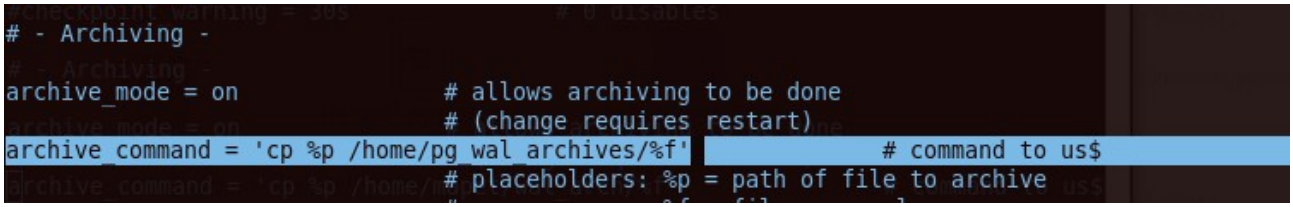
Zarchiwizowane pliki WAL muszą gdzieś lądować, więc przygotowujemy sobie odpowiedni katalog. Ważne jest by jego właścicielem był użytkownik systemowy postgres. Zadbaj o to by katalog leżał na jakimś dysku ze sporą ilością miejsca, ponieważ zarchiwizowane pliki WAL sporo go zajmą.

Przygotowanie katalogu do którego będą wrzucane zarchiwizowane pliki WAL:

```
[root@vps-1077604-8206 home]# mkdir /home/pg_wal_archives
[root@vps-1077604-8206 home]# chown postgres /home/pg_wal_archives/
[root@vps-1077604-8206 home]#
```

Została jeszcze jedna rzecz. Pliki WAL same się nie skopiują, więc musimy podać komendę która to zrobi. Można by sobie zadać pytanie – a dlaczego nie wystarczy wskazać ścieżki do katalogu? Komenda którą podajemy to nie musi być polecenie cp. Dzięki takiej konstrukcji możemy tu wykorzystać rsync, ftp, albo jakikolwiek inny mechanizm.

Odszukujemy parametr ARCHIVE_COMMAND i podajemy komendę kopiującą oraz odremowujemy go:



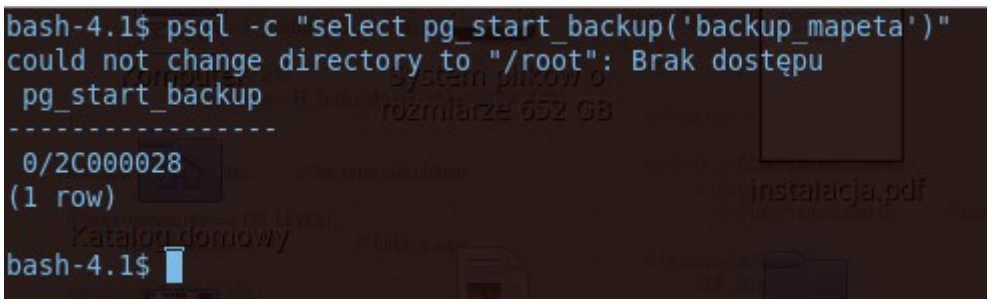
Ponieważ plik postgresql.conf nie jest czytany „na żywo”, musimy przeładować konfigurację. Przeładowanie konfiguracji:

service postgresql-9.4 restart

Zadbaliśmy już o kopiowanie plików WAL, teraz potrzebujemy naszego punktu wyjścia jakim będzie kopia zapasowa. Nie może to być kopia stworzona z użyciem pg_dump ani pg_dumpall ponieważ są to kopie logiczne a nie fizyczne. Aby wykonać kopię zapasową online, musmy przejść do trybu backupu. Sprawia on że zmiany w bazie nie będą utrwalane w plikach danych, a jedynie w plikach WAL. Zostaną one utrwalone dopiero po zakończeniu trybu backupu. Chodzi o spójność danych, nie możemy doprowadzić do sytuacji że coś się pozmienia w tych plikach w trakcie kopiowania.

Tryb backupu:

psql -c „select pg_start_backup('etykieta_backupu')”



Utworzymy sobie też katalog pod backup:

```
[root@vps-1077604-8206 backups]# pwd
/var/lib/pgsql/9.4/backups
[root@vps-1077604-8206 backups]# mkdir 20160404
[root@vps-1077604-8206 backups]# chown postgres 20160404
[root@vps-1077604-8206 backups]# ls -la
razem 12
drwx------ 3 postgres postgres 4096 04-04 17:25 .
drwx------ 4 postgres postgres 4096 03-22 11:41 ..
drwxr-xr-x 2 postgres dba      4096 04-04 17:25 20160404
[root@vps-1077604-8206 backups]#
```

W dalszej kolejności kopiujemy pliki. Możemy wykonać to dowolną komendą systemu operacyjnego.

```
bash-4.1$ tar cfP /var/lib/pgsql/9.4/backups/20160404/backup.tar /var/lib/pgsql/9.4/data
bash-4.1$ ls /var/lib/pgsql/9.4/backups/20160404/
backup.tar
bash-4.1$
```

Po zakończeniu kopiowania wychodzimy z trybu backupu:

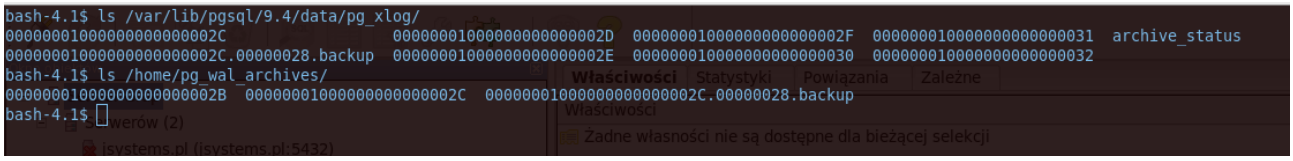
psql -c „select pg_stop_backup()”

```
bash-4.1$ psql -c "select pg_stop_backup()"
could not change directory to "/root": Brak dostępu
UWAGA:  pg_stop_backup kompletny, zarchiwizowano wszystkie wymagane segmenty WAL
 pg_stop_backup
----------------
 0/2C0000B8
(1 row)

bash-4.1$
```

Uruchomiłem tryb backupu i stworzyłem tabelkę do której wrzucam duży wolumen danych. Chodzi mi o to by troszkę plików WAL zostało dodanych do naszego nowego katalogu. Stworzyłem pustą tabelkę i uruchomiłem prosty skrypt w języku Plpg/SQL który do tej nowej tabelki dodaje 10 milionów wierszy.

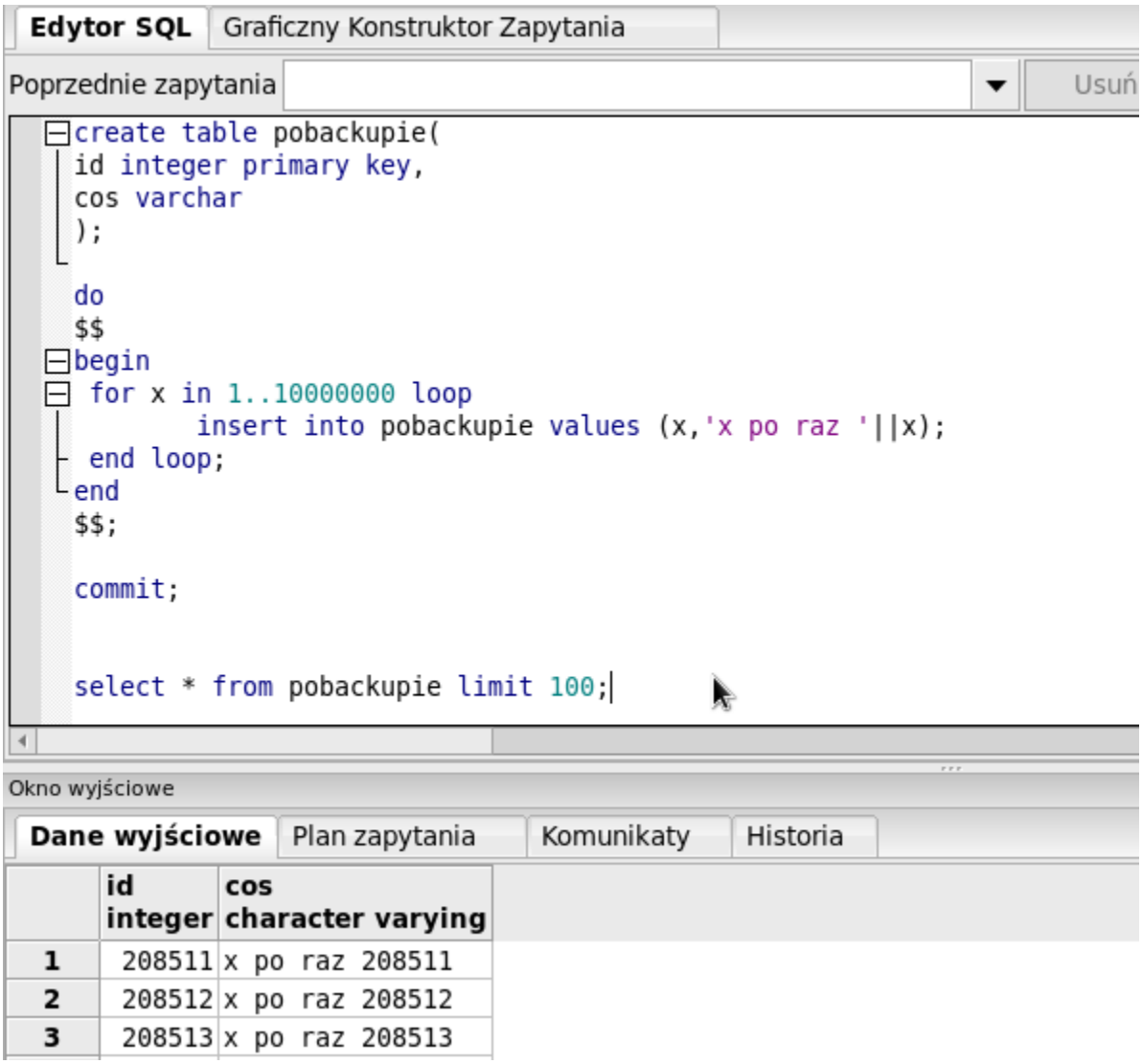
```
create table pobackupie(
id integer primary key,
cos varchar
);

do
$$
begin
 for x in 1..10000000 loop
        insert into pobackupie values (x,'x po raz '||x);
 end loop;
end
$$;

commit;
```

Zawartość katalogu pg_xlog i nowego katalogu na zarchwizowane pliki WAL w trakcie wykonywania powyższego skryptu:

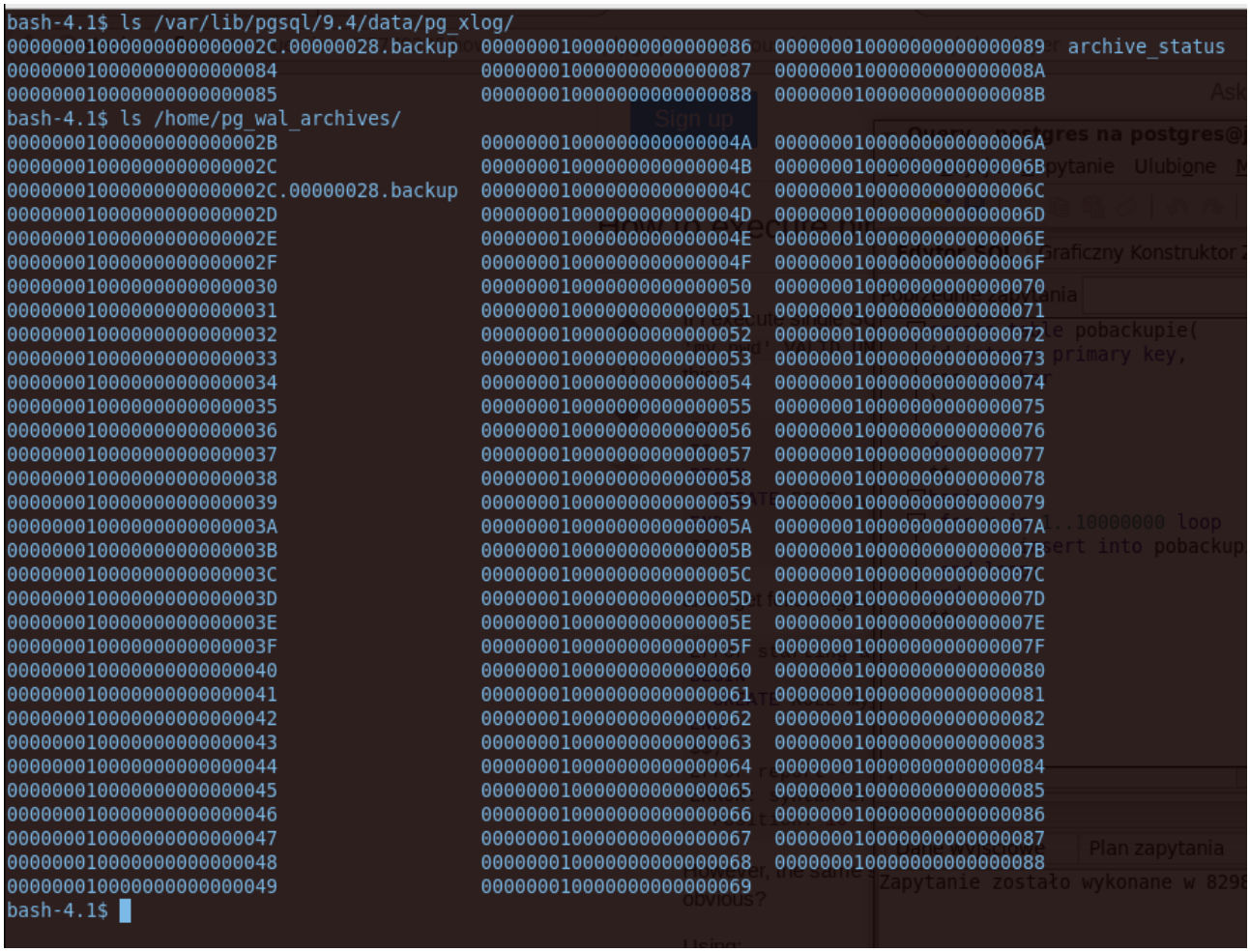


Zwróć uwagę że tabelkę stworzyłem po wykonaniu backupu, a więc w kopii zapasowej nie ma o niej informacji! To ważne bo za chwilę będziemy odtwarzać bazę z użyciem wszystkich zarchiwizowanych plików WAL.

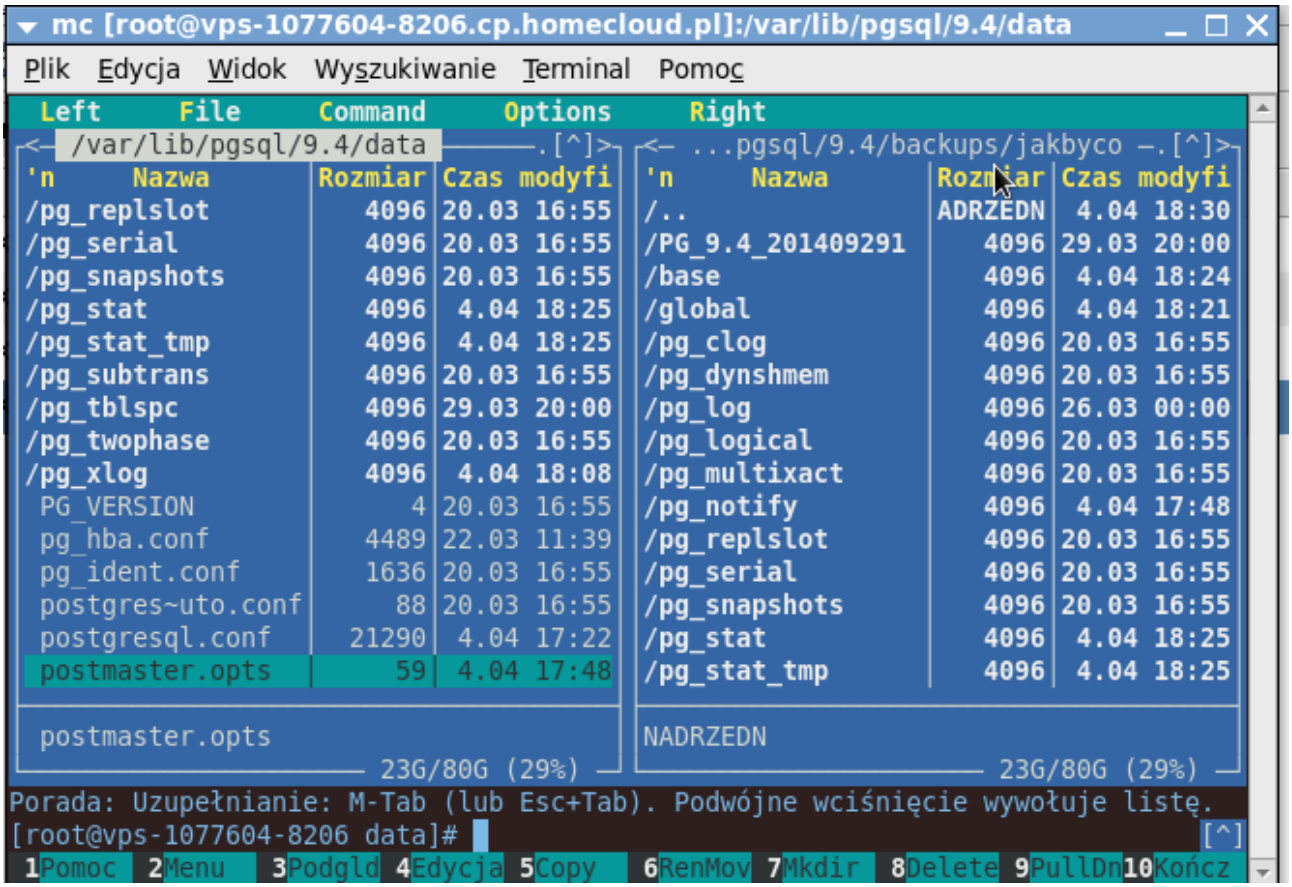
Po zakończeniu mielenia skryptu nasza tabelka zawiera sporo danych:



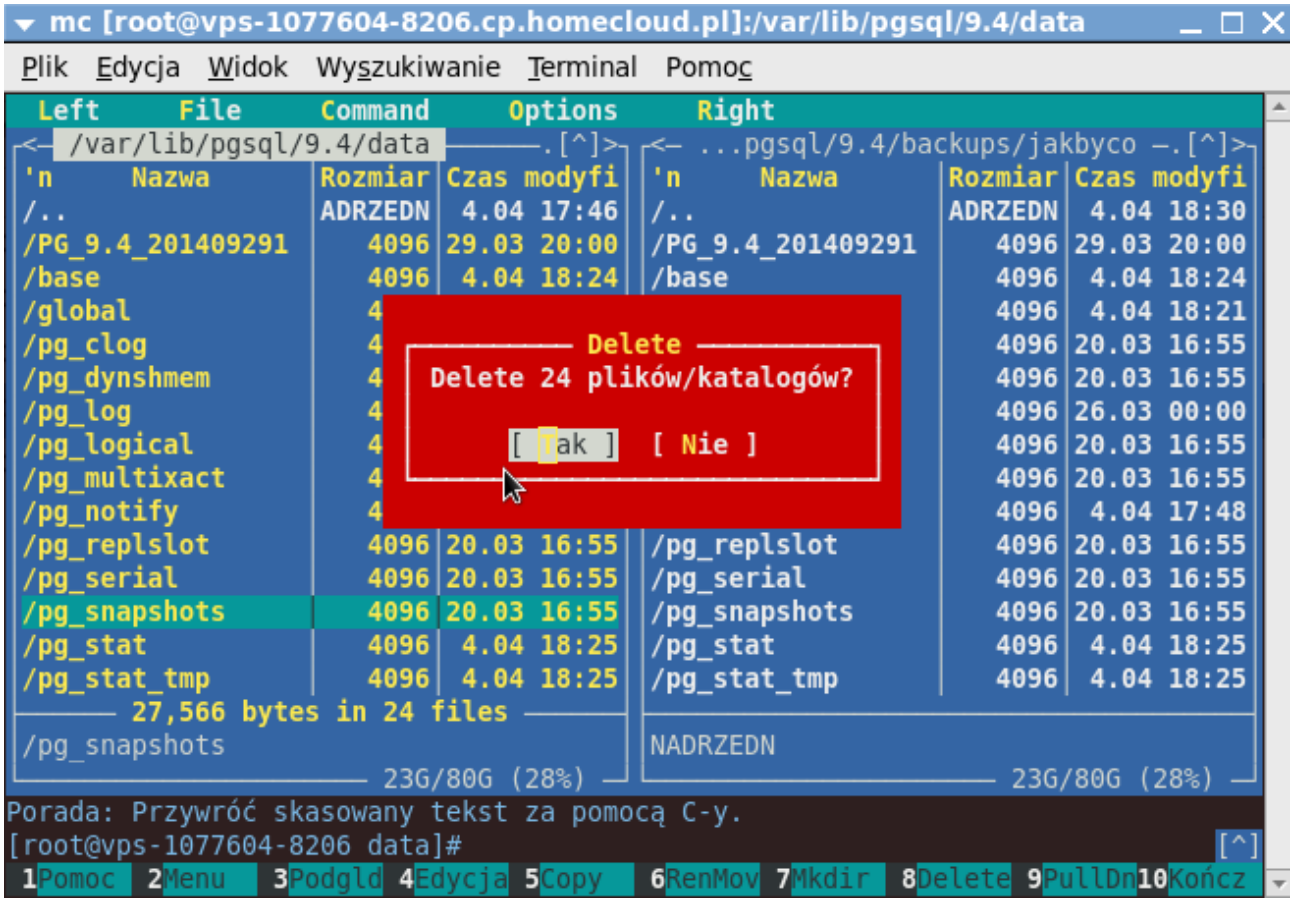
Mój katalog ze zarchiwizowanymi plikami WAL też trochę się wypełnił. Powstało w nim 89 plików, każdy po 16 MB - razem 1424 MB, a więc prawie 1,5 GB!

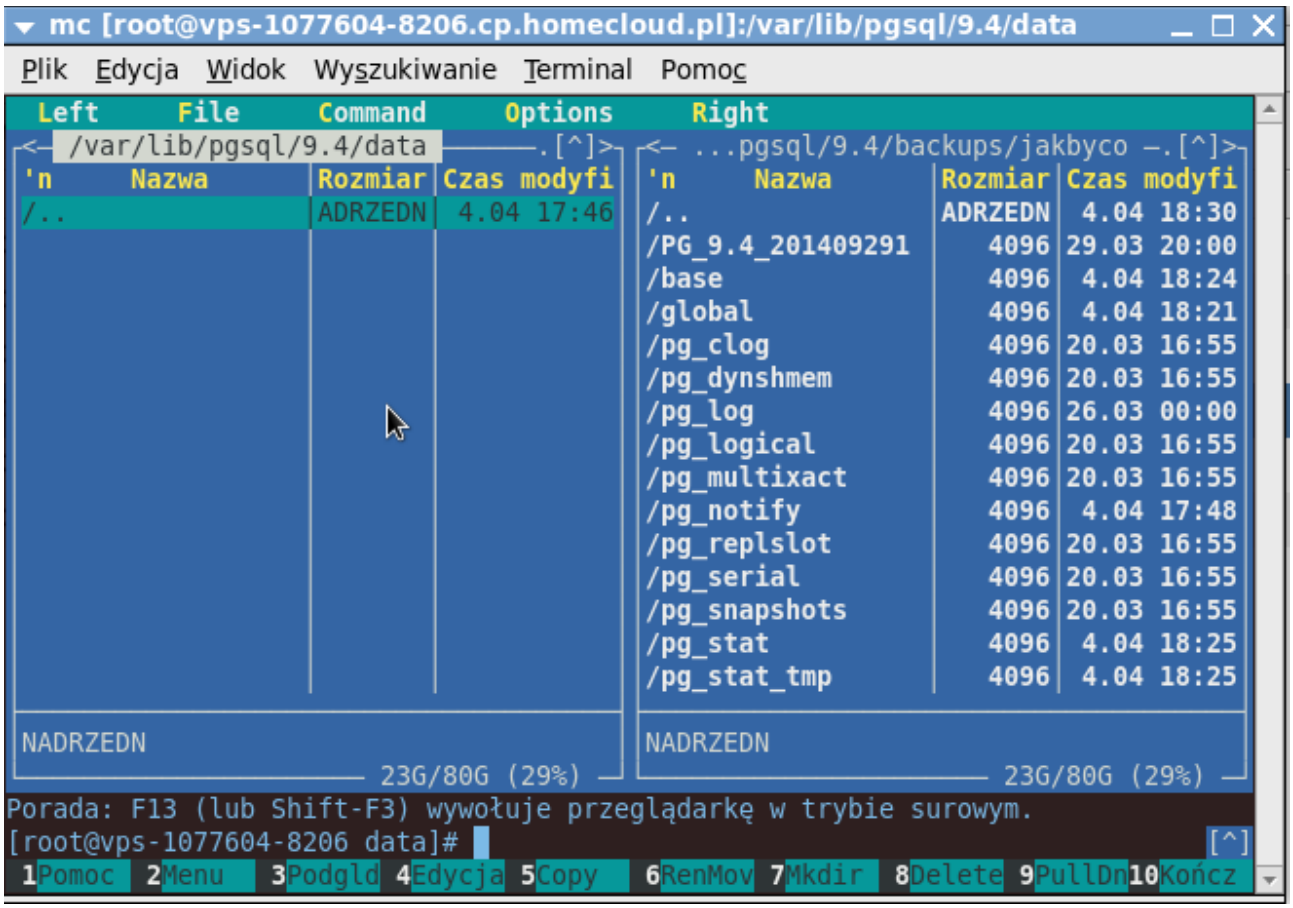


Zasymulujemy teraz awarię która będzie wymagała odtwarzania bazy do punktu tuż przed awarią. Zanim zaczniemy odtwarzać, warto zrobić kopię aktualnego stanu plików danych, nawet jeśli są one uszkodzone. Może się tak zdarzyć że w trakcie odtwarzania coś pójdzie nie tak i jeszcze bardziej napsujemy.

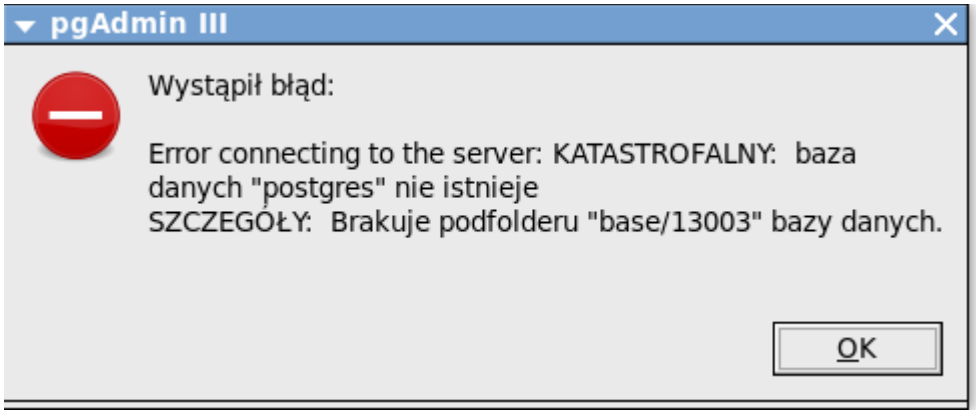


Przed odtwarzaniem usuwamy wszystkie pliki danych z katalogu $PGDATA, będą one zastępowane tymi z kopii zapasowej.





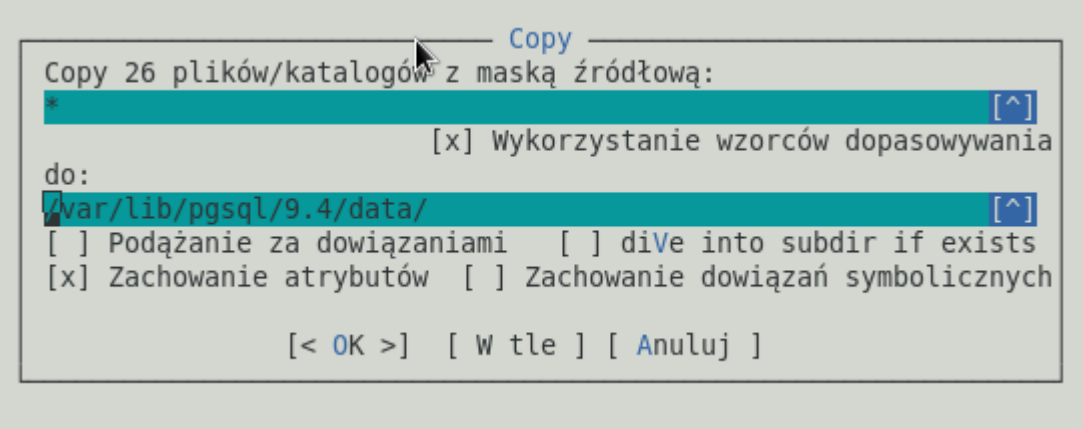
Przypominam że klaster jest cały czas uruchomiony. Próba dostępu do jakiejkolwiek bazy kończy się w tej chwili tak:



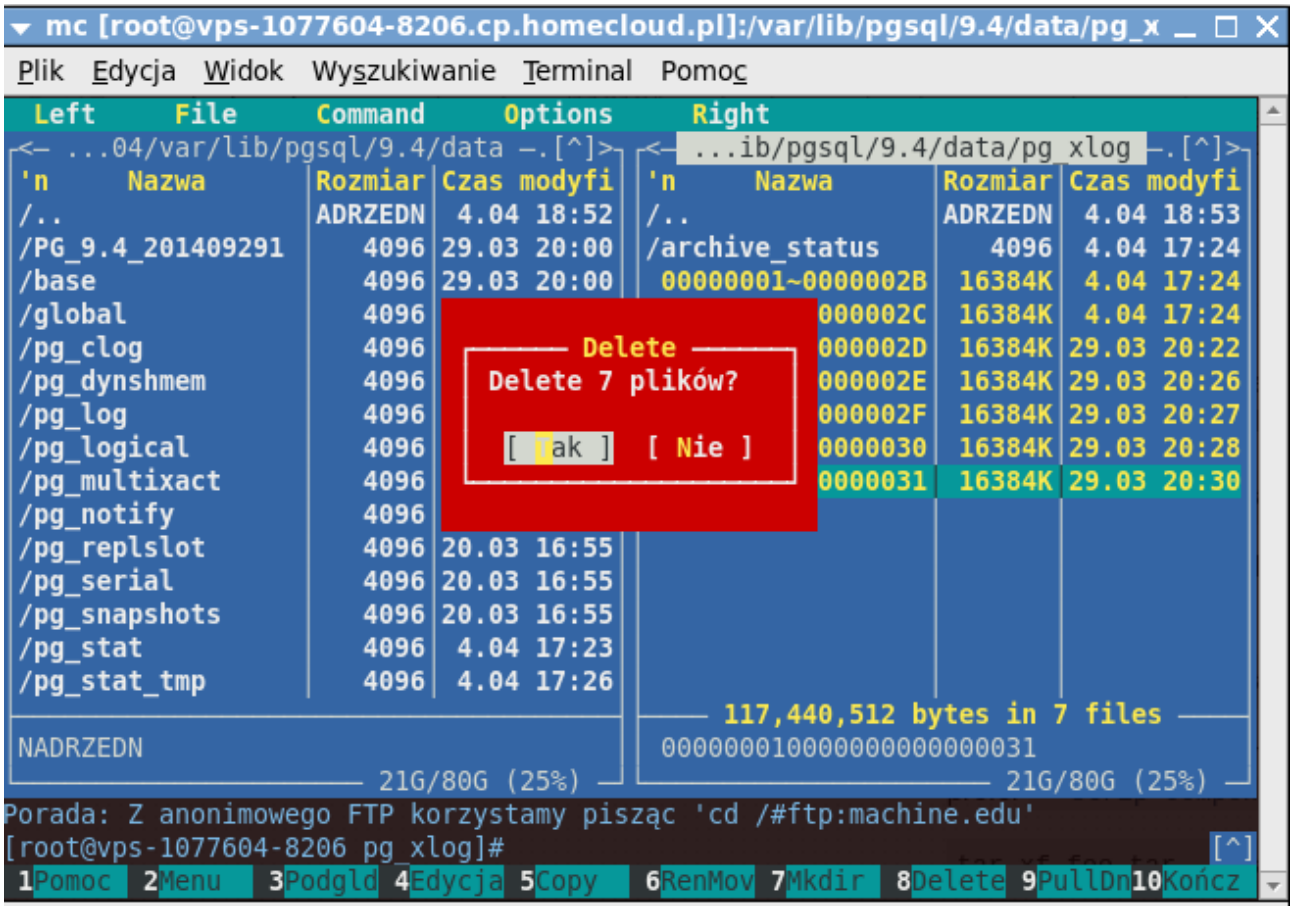
Zatrzymujemy serwer:

service postgresql-9.4 stop

Kopiujemy pliki z backupu :

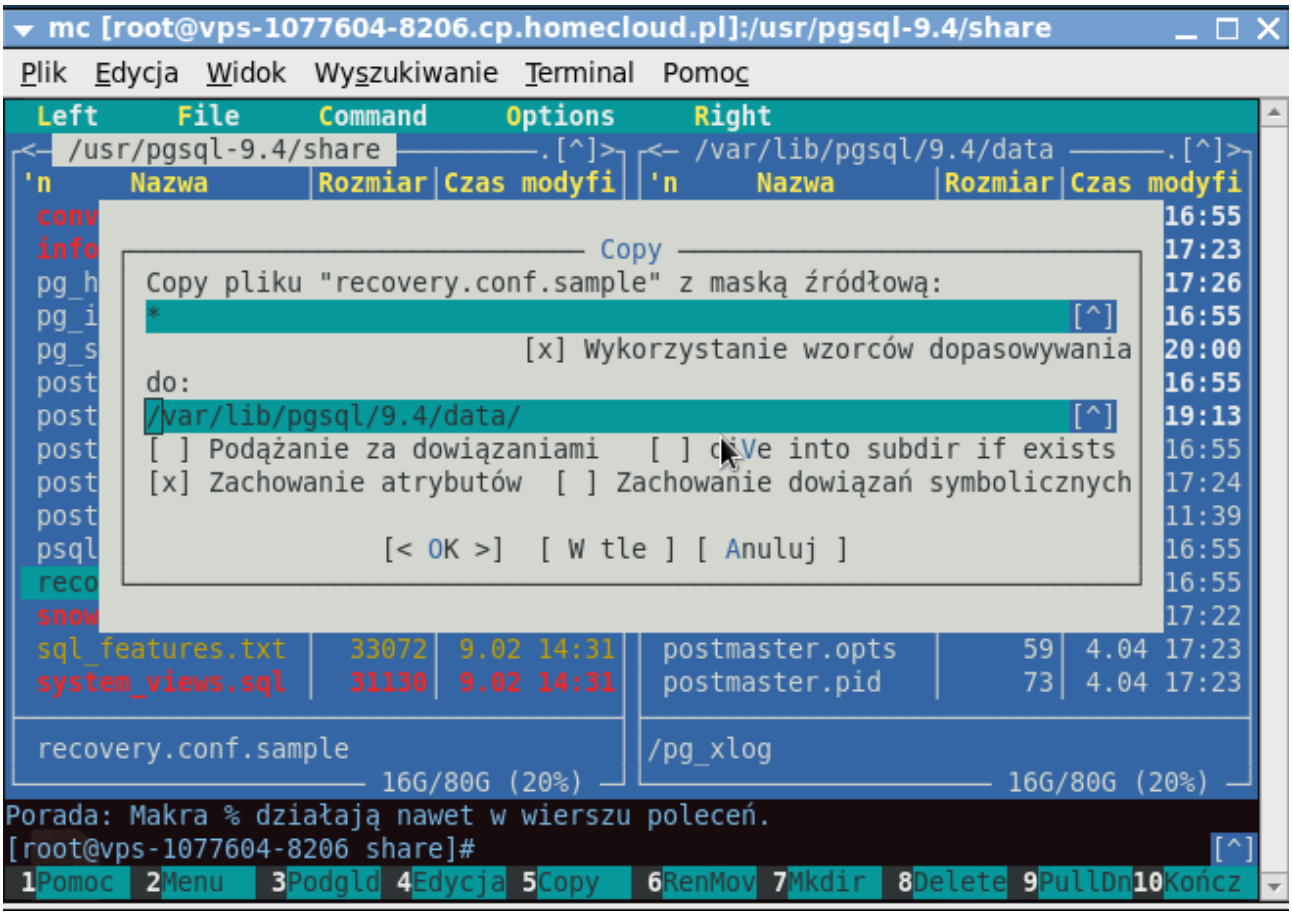


Wywalamy zawartość pg_xlog bo i tak nie koreluje z aktualnym stanem:



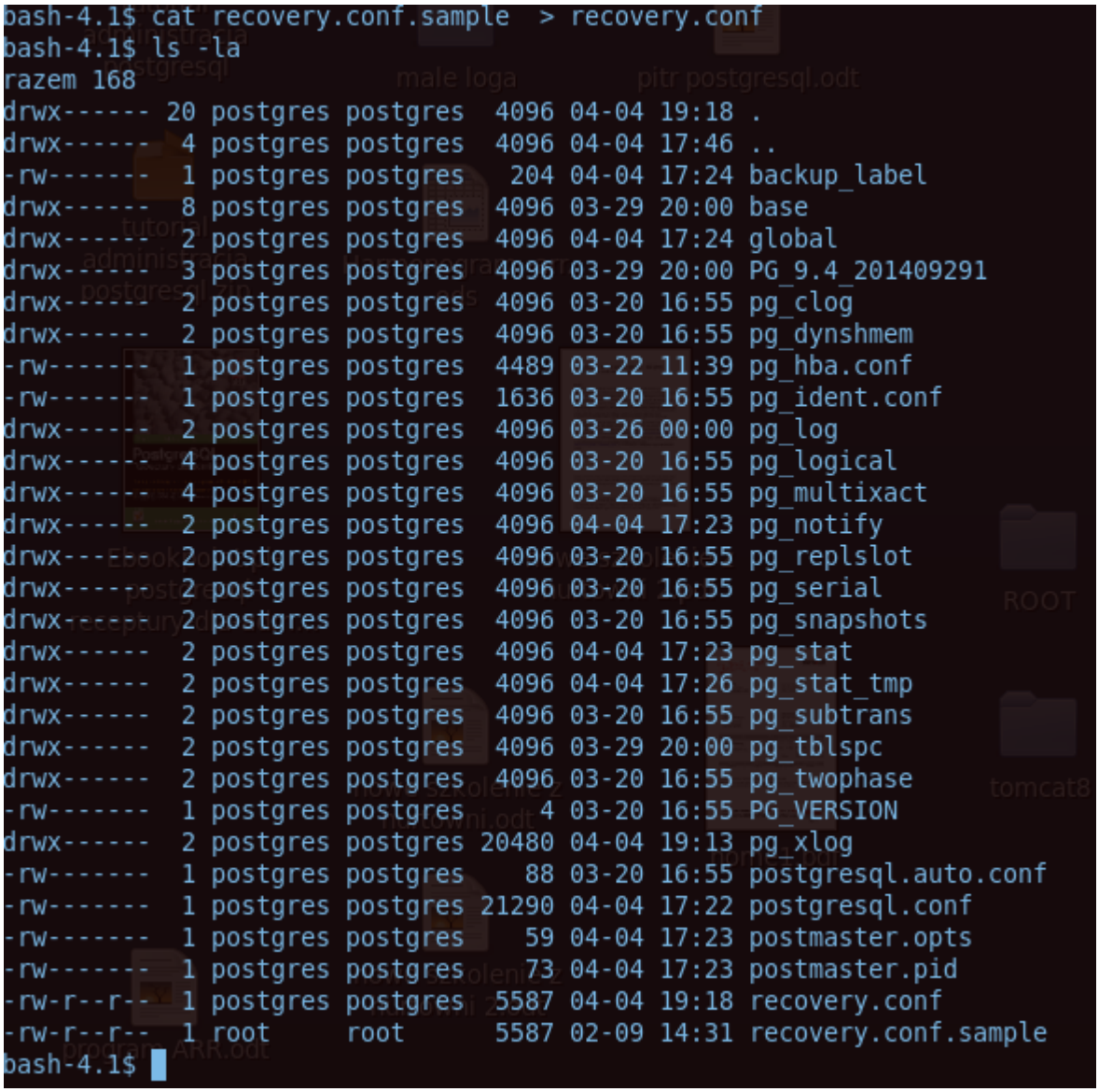
Do odzyskiwania będzie nam potrzebny plik recovery.conf. Jego obecność w katalogu $PGDATA spowoduje że klaster przy uruchamianiu przejdzie w tryb odtwarzania. Poza odtwarzaniem nie powinno go tam oczywiście być. W nim zapisane jest skąd ma zaczytać pliki WAL i jak ma je odtwarzać. Skąd go jednak wziąć? W katalogu /usr/pgsql-9.4/share mamy wiele przykładowych plików konfiguracyjnych, w tym właśnie recovery.conf. Jego nazwa będzie jednak zawierała końcówkę „.sample” którą będziemy musieli usunąć.

Kopiujemy przykładowy plik recovery.conf.sample do katalogu $PGDATA i zmieniamy jego nazwę na recovery.conf.

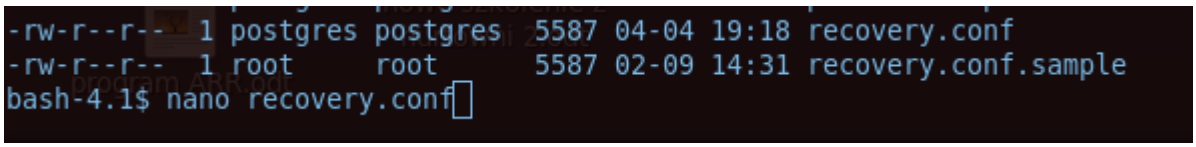


Kopiujemy zawartość pliku recovery.conf.sample do pliku recovery.conf:

cat recovery.conf.sample > recovery.conf



Edytujemy ten plik:



odnajdujemy w nim linijkę z restore_command, odkomentowujemy i piszemy tak:

'cp /home/pg_wal_archives/%f %p'

```
#
# NOTE that the basename of %p will be different from %f; do not
# expect them to be interchangeable.
#
restore_command = 'cp /home/pg_wal_archives/%f %p'              # e.g. 'c
#
```

Konstrukcja tej instrukcji jest łudząco podobna do instrukcji archive_command – owszem i działa w drugą stronę :)

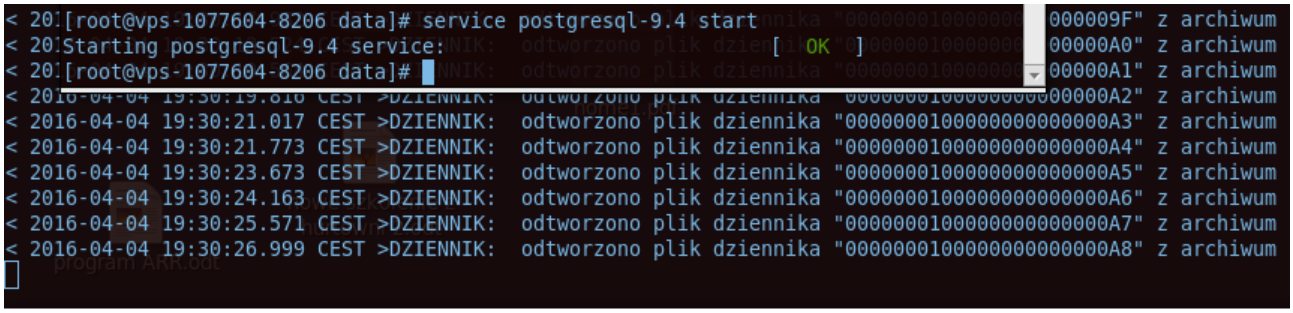
Z użyciem programu tail włączamy sobie podgląd logów :

```
[root@vps-1077604-8206 data]# tail -f /var/lib/pgsql/9.4/data/pg_log/postgresql-Mon.log
< 2016-04-04 02:33:31.780 CEST >DZIENNIK:  niekompletny pakiet uruchomieniowy
< 2016-04-04 17:23:00.106 CEST >DZIENNIK:  odebrano żądanie szybkiego zamknięcia
< 2016-04-04 17:23:00.136 CEST >DZIENNIK:  przerywanie wszelkich aktywnych transakcji
< 2016-04-04 17:23:00.142 CEST >DZIENNIK:  zamknięto program wywołujący autoodkurzanie
< 2016-04-04 17:23:00.226 CEST >DZIENNIK:  zamykanie
< 2016-04-04 17:23:00.538 CEST >DZIENNIK:  system bazy danych jest zamknięty
< 2016-04-04 17:23:01.412 CEST >DZIENNIK:  system bazy danych został zamknięty 2016-04-04 17:23:00 CEST
< 2016-04-04 17:23:01.510 CEST >DZIENNIK:  MultiXact member wraparound protections are now enabled
< 2016-04-04 17:23:01.513 CEST >DZIENNIK:  uruchomiono program wywołujący autoodkurzanie
< 2016-04-04 17:23:01.513 CEST >DZIENNIK:  system bazy danych jest gotowy do przyjmowania połączeń
```

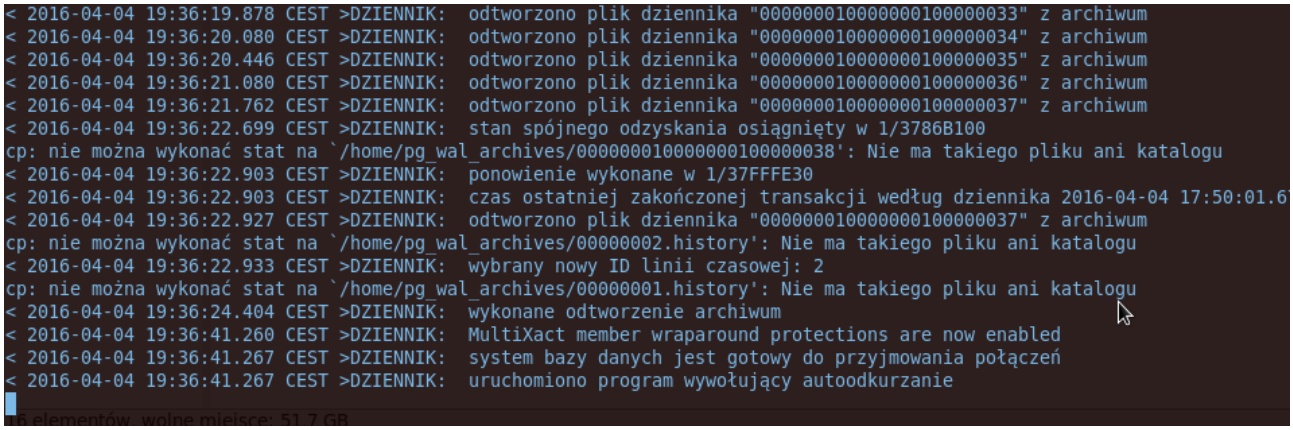
Uruchamiamy usługę, a PostgreSQL widząc istnienie pliku recovery.conf rozpoczyna odtwarzanie:

```
postgresql-Fri.log  postgresql-Mon.log  postgresql-Sat.log  postgresql-Sun.log  postgresql-Thu.log  postgresql-Tue.log
[root@vps-1077604-8206 data]# tail -f /var/lib/pgsql/9.4/data/pg_log/postgresql-Mon.log
< 2016-04-04 02:33:31.780 CEST >DZIENNIK:  niekompletny pakiet uruchomieniowy
< 2016-04-04 17:23:00.106 CEST >DZIENNIK:  odebrano żądanie szybkiego zamknięcia
< 2016-04-04 17:23:00.142 CEST >DZIENNIK:  zamknięto program wywołujący autoodkurzanie
< 2016-04-04 17:23:00.226 CEST >DZIENNIK:  zamykanie
< 2016-04-04 17:23:00.538 CEST >DZIENNIK:  system bazy danych jest zamknięty
< 2016-04-04 17:23:01.412 CEST >DZIENNIK:  system bazy danych został zamknięty 2016-04-04 17:23:00 CEST
< 2016-04-04 17:23:01.510 CEST >DZIENNIK:  MultiXact member wraparound protections are now enabled
< 2016-04-04 17:23:01.513 CEST >DZIENNIK:  uruchomiono program wywołujący autoodkurzanie
< 2016-04-04 17:23:01.513 CEST >DZIENNIK:  system bazy danych jest gotowy do przyjmowania połączeń
< 2016-04-04 19:29:43.784 CEST >DZIENNIK:  działanie systemu bazy danych zostało przerwane; ostatnie znane podniesienie
< 2016-04-04 19:29:43.784 CEST >DZIENNIK:  tworzenie brakującego folderu WAL "pg_xlog/archive_status"
< 2016-04-04 19:29:43.880 CEST >DZIENNIK:  rozpoczęto odzyskiwanie archiwum
< 2016-04-04 19:29:43.899 CEST >DZIENNIK:  odtworzono plik dziennika "00000001000000000000002C" z archiwum
< 2016-04-04 19:29:43.917 CEST >DZIENNIK:  ponowienie uruchamia się w 0/2C000090
< 2016-04-04 19:29:43.918 CEST >DZIENNIK:  stan spójnego odzyskania osiągnięty w 0/2C0000B8
< 2016-04-04 19:29:43.936 CEST >DZIENNIK:  odtworzono plik dziennika "00000001000000000000002D" z archiwum
< 2016-04-04 19:29:44.190 CEST >DZIENNIK:  odtworzono plik dziennika "00000001000000000000002E" z archiwum
< 2016-04-04 19:29:44.450 CEST >DZIENNIK:  odtworzono plik dziennika "00000001000000000000002F" z archiwum
< 2016-04-04 19:29:44.712 CEST >DZIENNIK:  odtworzono plik dziennika "000000010000000000000030" z archiwum
< 2016-04-04 19:29:44.996 CEST >DZIENNIK:  odtworzono plik dziennika "000000010000000000000031" z archiwum
< 2016-04-04 19:29:45.265 CEST >DZIENNIK:  odtworzono plik dziennika "000000010000000000000032" z archiwum
< 2016-04-04 19:29:45.523 CEST >DZIENNIK:  odtworzono plik dziennika "000000010000000000000033" z archiwum
< 2016-04-04 19:29:45.779 CEST >DZIENNIK:  odtworzono plik dziennika "000000010000000000000034" z archiwum
< 2016-04-04 19:29:46.050 CEST >DZIENNIK:  odtworzono plik dziennika "000000010000000000000035" z archiwum
< 2016-04-04 19:29:46.309 CEST >DZIENNIK:  odtworzono plik dziennika "000000010000000000000036" z archiwum
< 2016-04-04 19:29:46.583 CEST >DZIENNIK:  odtworzono plik dziennika "000000010000000000000037" z archiwum
```

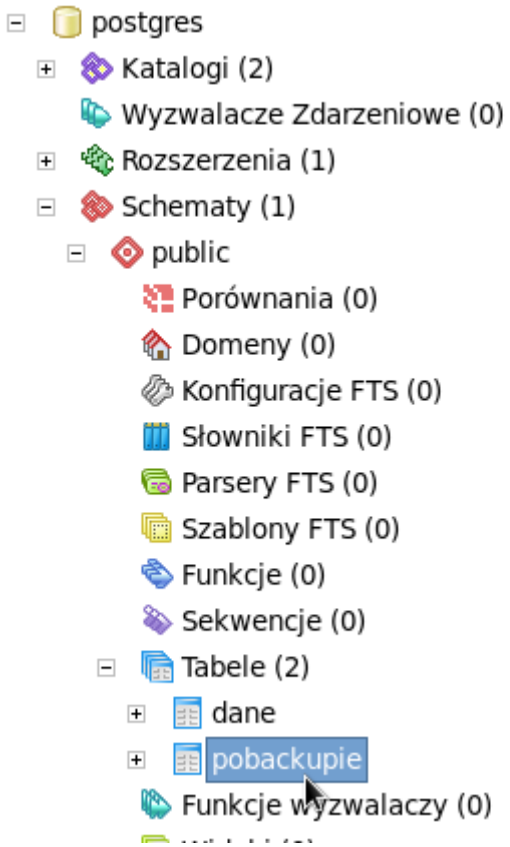
Zwróć szczególną uwagę na to, że mimo że teoretycznie usługa została uruchomiona jako taka, to odtwarzanie cały czas trwa i nadal nie będzie można się podłączyć do bazy.



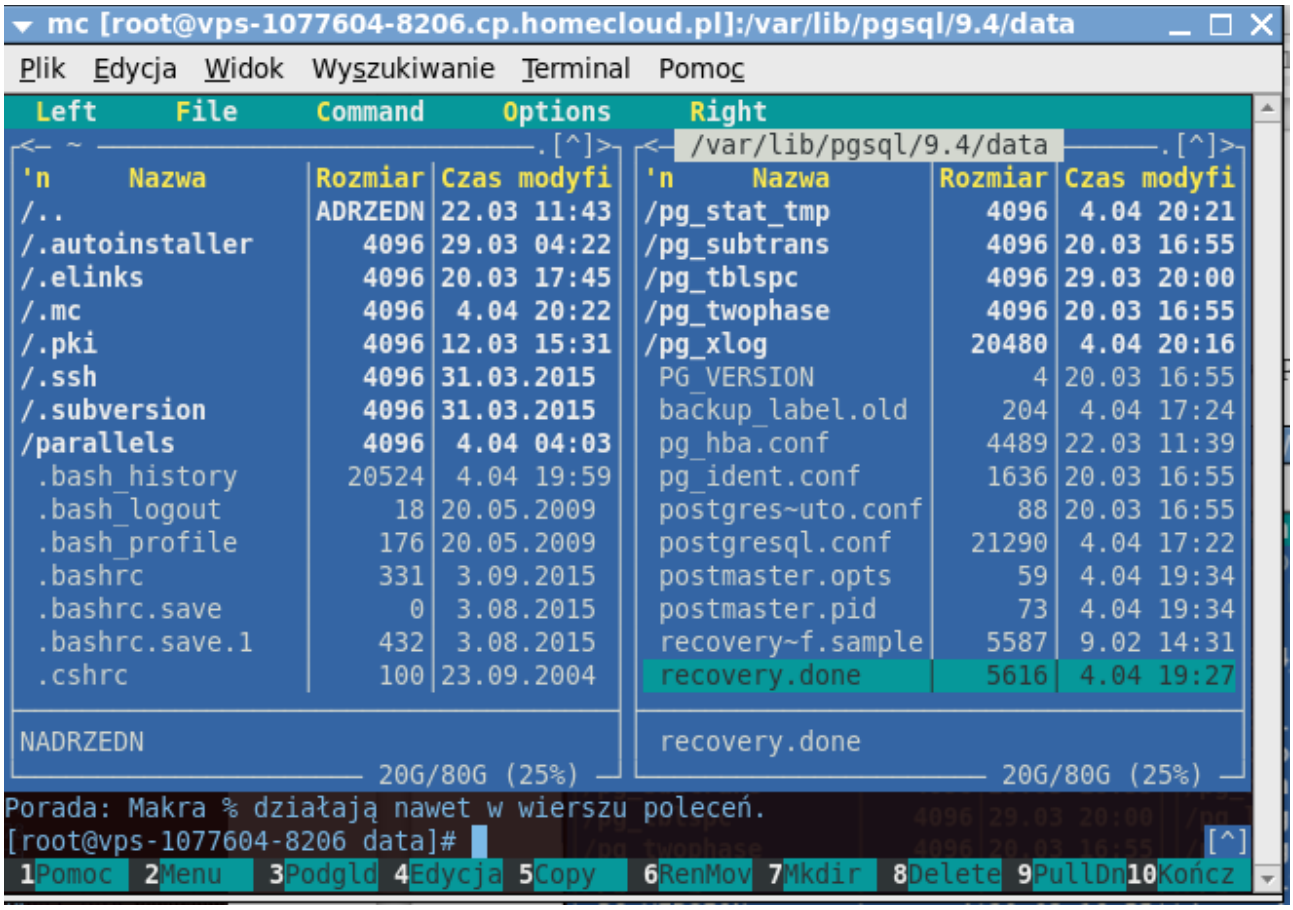
Gdy zakończy odtwarzanie:



Odtwarzanie zostało zakończone, podłączam się więc do bazy i sprawdzam czy istnieje moja tabelka:



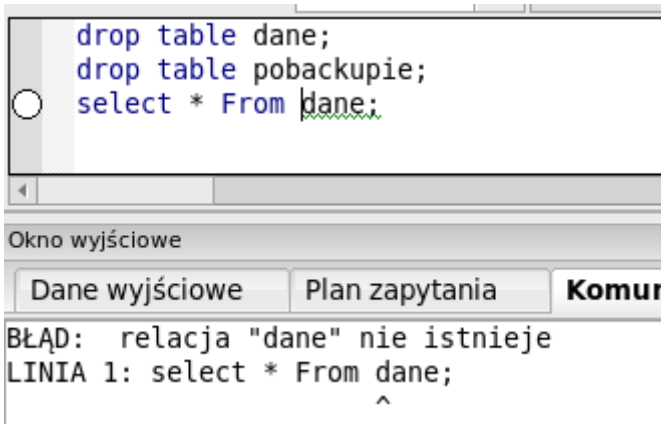
Przy okazji zauważ jedną bardzo ważną rzecz:



Po zakończeniu odtwarzania nazwa pliku recovery.conf została zmieniona na recovery.done
Dzieje się tak po to, by po restarcie serwer nowu nie wszedł w tryb odtwarzania.

Przywracanie do punktu w czasie:

Procedura wygląda niemal identycznie, z tą różnicą że do pliku recovery.conf dodajemy jeszcze jeden parametr określający punkt w czasie do którego przywracamy. Aby się upewnić odnośnie działania tej procedury skasuję dwie tabele, a następnie przywrócę bazę do stanu przed ich skasowaniem.



Zmiana i odkomentowanie parametru określającego punkt w czasie:

```
#
#recovery_target_name = ''      # e.g. 'daily backup 2011-01-26'
#
recovery_target_time = '2016-04-04 20:05:00 EST'       # e.g. '2004-07-14 22:39:00 EST'
#
#recovery_target_xid = ''
#
#recovery_target_inclusive = true
```

Wskazałem czas tuż przed skasowaniem tabel. Otwieram bazę jak zwykle, a po zakończonym procesie odtwarzania sprawdzam czy na pewno mam swoje skasowane tabele: