Roteiro de Instalação do PG_CRON no PostgreSQL 14 hospedado em Linux CentOS 7

Julho/2022

MODIFICAÇÕES NESTE DOCUMENTO
Item: Criação do Documento
Alteração/Inclusão: todo o documento
Responsável: Gilberto A. Oliveira
Data: 25/07/2022

Item: Instalação do PG_CRON
Alteração/Inclusão:
- Habilitação do loopback em IPV6
- Verificação da execução.
Responsável: Gilberto A. Oliveira
Data: 28/07/2022

Índice

Julho/2022

Objetivo

O objetivo desse roteiro é servir como guia (howto) para:

- Instalação da extensão de agendamento de tarefas no banco de dados PostgreSQL 14 em um sistema operacional Linux CentOS 7.

O leitor desse documento deve saber que, neste guia não temos a preocupação de explicar os fundamentos dos comandos apresentados e nem as razões que orientaram as escolhas aqui destacadas.

Alerta: Este documento não tem a pretensão de substituir as documentações oficiais dos sistemas abordados. Recomenda-se as documentações oficiais do PostgreSQL 14 (https://www.postgresql.org/docs/14/index.html) e Linux CentOS 7 (https://docs.centos.org/en-US/docs/) para exaurir dúvidas e/ou fundamentos que não estejam cobertos por esse roteiro.

Pré-requisitos

Este material considera que o leitor já tenha um ambiente montado e configurado com o banco de dados PostgreSQL 14 em uma distribuição Linux CentOS 7, com todos os pré-requisitos de softwares instalados e configurados.

- PGSQLDEV-14
- gcc
- make

O leitor pode recorrer o roteiro Postgresql com Login_hook no CentOS 7, presente neste repositório, para atender os pré-requistos desse documento.

Instalação do PG_CRON

A extensão PG_CRON permite ao PostgreSQL definir o agendamento de tarefas pelo próprio banco de dados. Para tal, será necessário a compilação e criação da extensão. A documentação e download dos arquivos necessários estão disponíveis em https://github.com/citusdata/pg_cron

Outras documentações de referência:
https://dev.to/renatoassis01/como-agendar-execucao-de-consultas-e-comandos-pelo-postgresql-no-rds-4b1c
https://access.crunchydata.com/documentation/pg_cron/1.3.1/
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/PostgreSQL_pg_cron.html#PostgreSQL_pg_cron.examples

ATENÇÃO: Antes de continuar se certifique que os cuidados de pré-instalação pgsqldev-14, make, gcc e path relatados anteriormente tenham sidos tomados.

1 – baixar e descompactar o pacote do pg_cron em um diretório temporário
2 – acessar o diretório temporário onde a extensão foi salva

```
# cd /tmp/pg_cron/
#
```

3 – Setar a variável de ambiente PATH para acessar o diretório do PostgreSQL:

```
export PATH=/usr/pgsql-14/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
```

```
# export PATH=/usr/pgsql-14/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin

#
```

4 - executar os comandos de compilação:

```
make && sudo PATH=$PATH make install
```

```
[root@centos73 pg_cron]# make && sudo PATH=$PATH make install
gcc -std=gnu99 -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Werror=vla -
Wendif-labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -
                            fexcess-precision=standard -O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -
                           pg_cron/src/task_states.bc pg_cron/src/entry.bc
...
...

 pg_cron/src/task_states.bc pg_cron/src/entry.bc

[root@centos73 pg_cron]#
```

4 – conectado no banco de dados criar a extensão pg_cron

ATENÇÃO: O comando abaixo deve ser executado para cada novo banco!!

```
CREATE EXTENSION pg_cron;
```

```
# su - postgres
Último login:Sex Abr 22 12:08:20 -03 2022em pts/0
-bash-4.2$ psql
WARNING:  Function login_hook.login() is not invoked because it does not exist in database postgres
psql (14.2)
Type "help" for help.

postgres=# CREATE EXTENSION pg_cron;
CREATE EXTENSION
postgres=#
```

5 – Concessão de privilégio de execução para public

```
GRANT USAGE ON SCHEMA cron TO public;
```

```
postgres=#  GRANT USAGE ON SCHEMA cron TO public;
GRANT
postgres=#
```

6 – Acrescentar a extensão no arquivo de configuração POSTGRESQL.CONF

6.1 - Para ver o local do arquivo você deve conectar no banco de dados e executar o comando

```
SHOW config_file;
```

```
# su - postgres
Último login:Sex Abr 22 10:56:21 -03 2022em pts/0
-bash-4.2$ psql
psql (14.2)
Type "help" for help.

postgres=# SHOW config_file;

              config_file
----------------------------------------
 /var/lib/pgsql/14/data/postgresql.conf
(1 row)

postgres=#
```

6.2 – Editar o arquivo postgresql.conf e acrescentar a linha

```
shared_preload_libraries = 'pg_cron'
```

```
#local_preload_libraries = ''
#session_preload_libraries = ''
session_preload_libraries = 'login_hook'
shared_preload_libraries = 'pg_cron'
#shared_preload_libraries = ''   # (change requires restart)
```

6.3 – Editar o arquivo pg_hba.conf conforme abaixo:

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     scram-sha-256

# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
host    all             all             0.0.0.0/0               trust
host    replication     all             127.0.0.1/0             trust
```

6.4 – restart o serviço do PostgreSQL

`systemctl restart postgresql-14`

```
# systemctl restart postgresql-14
#
```

Validação da Instalação

1 – Alterne para o usuário postgres

`su - postgres`

```
# su - postgres
Último login:Sex Abr 22 12:14:39 -03 2022em pts/0
-bash-4.2$
```

2 – Faça uma conexão com o banco de dados

`psql`

```
-bash-4.2$ psql
WARNING:  transaction left non-empty SPI stack
HINT:  Check for missing "SPI_finish" calls.
psql (14.2)
Type "help" for help.


postgres=#
```

3 – Verifique se a biblioteca foi carregada.

`SHOW shared_preload_libraries;`

```
postgres=# SHOW shared_preload_libraries;
 shared_preload_libraries
--------------------------
 pg_cron
(1 row)


postgres=#
```

O resultado esperado é o retorno do nome da biblioteca, caso não apareça será necessário revisar os passos anteriores.

Criação de um JOB

1 – Saiba que o PG_CRON mantém as mesmas características de agendamento do CRON do linux, sendo possível uma variedade de agendamentos.

ATENÇÃO: O agendamento obedece ao horário GMT, isso significa que você deve agendar sempre com 3 horas a mais.

1.1 – Criando um job

```
SELECT cron.schedule('30 9 * * 6', $$DELETE FROM events WHERE event_time < now()
- interval '1 week'$$);
```

```
postgres=# SELECT cron.schedule('30 9 * * 6', $$DELETE FROM events WHERE event_time < now() - interval '1 week'$$);
 schedule
----------
        2
(1 row)

postgres=#
```

A tarefa agendada acima (jobid = 2), será executada todo sábado as 06:30 h da manhã.

Outros exemplos podem ser obtidos nas documentações referenciadas no início desse roteiro.

O pg_cron pode executar vários trabalhos em paralelo, mas executa no máximo uma instância de um trabalho por vez. Se uma segunda execução deve começar antes que a primeira termine, a segunda execução é enfileirada e iniciada assim que a primeira execução for concluída.

A programação usa a sintaxe cron padrão, na qual * significa "executar a cada período de tempo" e um número específico significa "mas somente neste momento":

```
┌──────────── min (0 - 59)
│ ┌────────── hour (0 - 23)
│ │ ┌──────── day of month (1 - 31)
│ │ │ ┌────── month (1 - 12)
│ │ │ │ ┌──── day of week (0 - 6) (0 to 6 are Sunday to
│ │ │ │ │                Saturday, or use names; 7 is also Sunday)
│ │ │ │ │
│ │ │ │ │
* * * * *
```

1.3 – Verificar todas as tarefas agendadas

```
SELECT * FROM cron.job
```

```
postgres=# SELECT * FROM cron.job;
 jobid |  schedule  |                            command                            | nodename  | nodeport | database | username | active | jobname
-------+------------+---------------------------------------------------------------+-----------+----------+----------+----------+--------+---------
     2 | 30 9 * * 6 | DELETE FROM events WHERE event_time < now() - interval '1 week' | localhost |     5432 | postgres | postgres | t      |
(1 row)

postgres=#
```

1.4 – Excluir tarefas agendadas

```
SELECT cron.unschedule(2);
```

```
-bash-4.2$ psql
WARNING:  transaction left non-empty SPI stack
HINT:  Check for missing "SPI_finish" calls.
psql (14.4)
Type "help" for help.

postgres=# SELECT cron.unschedule(2);
 unschedule
------------
 t
(1 row)


postgres=#
```

1.5 – Verificar o resultado (status) do agendamento

```
SELECT * FROM cron.job_run_details;
```

```
postgres=# SELECT * FROM cron.job_run_details;
 jobid | runid | job_pid | database | username |                  command                  |  status   |                  return_message
       |              start_time               |           end_time
-------+-------+---------+----------+----------+-------------------------------------------+-----------+----------------------------------
    26 |    14 |    8360 | postgres | postgres | insert into minha_tabela values (1)       | succeeded | INSERT 0 1
       | 28/07/2022 11:38:00.02794 -03  | 28/07/2022 11:38:00.032154 -03
(14 rows)

postgres=#
```

A figura acima mostra que o agendamento 26 foi realizado com sucesso.