

What is pg_cron?

pg_cron is a simple cron-based job scheduler for PostgreSQL (9.5 or higher) that runs inside the database as an extension. It uses the same syntax as regular cron, but it allows you to schedule PostgreSQL commands directly from the database:

```
-- Delete old data on Saturday at 3:30am (GMT)
SELECT cron.schedule('30 3 * * 6', $$DELETE FROM events WHERE event_time < now() - interval
schedule
-----
42

-- Vacuum every day at 10:00am (GMT)
SELECT cron.schedule('nightly-vacuum', '0 10 * * *', 'VACUUM');
schedule
-----
43

-- Change to vacuum at 3:00am (GMT)
SELECT cron.schedule('nightly-vacuum', '0 3 * * *', 'VACUUM');
schedule
-----
43

-- Stop scheduling jobs
SELECT cron.unschedule('nightly-vacuum' );
unschedule
-----
t
(1 row)

SELECT cron.unschedule(42);
unschedule
-----
t
```

pg_cron can run multiple jobs in parallel, but it runs at most one instance of a job at a time. If a second run is supposed to start before the first one finishes, then the second run is queued and started as soon as the first run completes.

The schedule uses the standard cron syntax, in which * means “run every time period”, and a specific number means “but only at this time”:

```
+----- 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)
| | | | |
| | | | |
| | | | |
* * * * *

```

An easy way to create a cron schedule is: crontab.guru.

The code in `pg_cron` that handles parsing and scheduling comes directly from the cron source code by Paul Vixie, hence the same options are supported. Be aware that `pg_cron` always uses GMT!

Installing `pg_cron`

Install on Red Hat, CentOS, Fedora, Amazon Linux with PostgreSQL 12 using [PGDG](https://pgdg.org/):

```

# Install the pg_cron extension
sudo yum install -y pg_cron_12

```

Install on Debian, Ubuntu with PostgreSQL 12 using apt.postgresql.org:

```

# Install the pg_cron extension
sudo apt-get -y install postgresql-12-cron

```

You can also install `pg_cron` by building it from source:

```

git clone https://github.com/citusdata/pg_cron.git
cd pg_cron
# Ensure pg_config is in your path, e.g.
export PATH=/usr/pgsql-12/bin:$PATH
make && sudo PATH=$PATH make install

```

Setting up `pg_cron`

To start the `pg_cron` background worker when PostgreSQL starts, you need to add `pg_cron` to `shared_preload_libraries` in `postgresql.conf`. Note that `pg_cron` does not run any jobs as long as a server is in `hot standby` mode, but it automatically starts when the server is promoted.

By default, the `pg_cron` background worker expects its metadata tables to be created in the “postgres” database. However, you can configure this by setting the `cron.database_name` configuration parameter in `postgresql.conf`.

```
# add to postgresql.conf:
shared_preload_libraries = 'pg_cron'
cron.database_name = 'postgres'
```

After restarting PostgreSQL, you can create the `pg_cron` functions and metadata tables using `CREATE EXTENSION pg_cron`.

```
-- run as superuser:
CREATE EXTENSION pg_cron;

-- optionally, grant usage to regular users:
GRANT USAGE ON SCHEMA cron TO marco;
```

Important: Internally, `pg_cron` uses `libpq` to open a new connection to the local database. It may be necessary to enable `trust` authentication for connections coming from localhost in `pg_hba.conf` for the user running the cron job. Alternatively, you can add the password to a `.pgpass` file, which `libpq` will use when opening a connection.

For security, jobs are executed in the database in which the `cron.schedule` function is called with the same permissions as the current user. In addition, users are only able to see their own jobs in the `cron.job` table.

Example use cases

Articles showing possible ways of using `pg_cron`:

- [Auto-partitioning using `pg_partman`](#)
- [Computing rollups in an analytical dashboard](#)
- [Deleting old data, vacuum](#)
- [Feeding cats](#)
- [Routinely invoking a function](#)

Managed services

The following table keeps track of which of the major managed Postgres services support `pg_cron`.

Service	Supported
Aiven	:heavy_check_mark:
Alibaba Cloud	:heavy_check_mark:

Service	Supported
Amazon RDS	:heavy_check_mark:
Azure	:heavy_check_mark:
Citus Cloud	:heavy_check_mark:
Crunchy Bridge	:heavy_check_mark:
DigitalOcean	:heavy_check_mark:
Google Cloud	:x:
Heroku	:x:
ScaleGrid	:heavy_check_mark:
Scaleway	:heavy_check_mark:
Supabase	:heavy_check_mark:
