

---

## pgsync

Sync data from one Postgres database to another (like [pg\\_dump/pg\\_restore](#)). Designed for:

- **speed** - tables are transferred in parallel
- **security** - built-in methods to prevent sensitive data from ever leaving the server
- **flexibility** - gracefully handles schema differences, like missing columns and extra columns
- **convenience** - sync partial tables, groups of tables, and related records

:tangerine: Battle-tested at Instacart



## Installation

pgsync is a command line tool. To install, run:

```
1 gem install pgsync
```

This will give you the [pgsync](#) command. If installation fails, you may need to install dependencies.

You can also install it with Homebrew:

```
1 brew install pgsync
```

## Setup

In your project directory, run:

```
1 pgsync --init
```

This creates `.pgsync.yml` for you to customize. We recommend checking this into your version control (assuming it doesn't contain sensitive information). [pgsync](#) commands can be run from this directory or any subdirectory.

## How to Use

First, make sure your schema is set up in both databases. We recommend using a schema migration tool for this, but pgsync also provides a few convenience methods. Once that's done, you're ready to sync data.

Sync tables

---

```
1 pgsync
```

Sync specific tables

```
1 pgsync table1,table2
```

Works with wildcards as well

```
1 pgsync "table*"
```

Sync specific rows (existing rows are overwritten)

```
1 pgsync products "where store_id = 1"
```

You can also preserve existing rows

```
1 pgsync products "where store_id = 1" --preserve
```

Or truncate them

```
1 pgsync products "where store_id = 1" --truncate
```

## Tables

Exclude specific tables

```
1 pgsync --exclude table1,table2
```

Add to `.pgsync.yml` to exclude by default

```
1 exclude:
2   - table1
3   - table2
```

Sync tables from all schemas or specific schemas (by default, only the search path is synced)

```
1 pgsync --all-schemas
2 # or
3 pgsync --schemas public,other
4 # or
5 pgsync public.table1,other.table2
```

## Groups

Define groups in `.pgsync.yml`:

---

```
1 groups:
2   group1:
3     - table1
4     - table2
```

And run:

```
1 pgsync group1
```

## Variables

You can also use groups to sync a specific record and associated records in other tables.

To get product 123 with its reviews, last 10 coupons, and store, use:

```
1 groups:
2   product:
3     products: "where id = {1}"
4     reviews: "where product_id = {1}"
5     coupons: "where product_id = {1} order by created_at desc limit 10"
6     stores: "where id in (select store_id from products where id = {1})"
      "
```

And run:

```
1 pgsync product:123
```

## Schema

Sync the schema before the data (this wipes out existing data)

```
1 pgsync --schema-first
```

Specify tables

```
1 pgsync table1,table2 --schema-first
```

Sync the schema without data (this wipes out existing data)

```
1 pgsync --schema-only
```

pgsync does not try to sync Postgres extensions.

---

## Sensitive Data

Prevent sensitive data like email addresses from leaving the remote server.

Define rules in `.pgsync.yml`:

```
1 data_rules:
2   email: unique_email
3   last_name: random_letter
4   birthday: random_date
5   users.auth_token:
6     value: secret
7   visits_count:
8     statement: "(RANDOM() * 10)::int"
9   encrypted_*: null
```

`last_name` matches all columns named `last_name` and `users.last_name` matches only the `users` table. Wildcards are supported, and the first matching rule is applied.

Options for replacement are:

- `unique_email`
- `unique_phone`
- `unique_secret`
- `random_letter`
- `random_int`
- `random_date`
- `random_time`
- `random_ip`
- `value`
- `statement`
- **`null`**
- `untouched`

Rules starting with `unique_` require the table to have a single column primary key. `unique_phone` requires a numeric primary key.

## Foreign Keys

Foreign keys can make it difficult to sync data. Three options are:

1. Defer constraints (recommended)
2. Manually specify the order of tables
3. Disable foreign key triggers, which can silently break referential integrity (not recommended)

---

To defer constraints, use:

```
1 pgsync --defer-constraints
```

To manually specify the order of tables, use `--jobs 1` so tables are synced one-at-a-time.

```
1 pgsync table1,table2,table3 --jobs 1
```

To disable foreign key triggers and potentially break referential integrity, use:

```
1 pgsync --disable-integrity
```

This requires superuser privileges on the `to` database. If syncing to (not from) Amazon RDS, use the `rds_superuser` role. If syncing to (not from) Heroku, there doesn't appear to be a way to disable integrity.

## Triggers

Disable user triggers with:

```
1 pgsync --disable-user-triggers
```

## Sequences

Skip syncing sequences with:

```
1 pgsync --no-sequences
```

## Append-Only Tables

For extremely large, append-only tables, sync in batches.

```
1 pgsync large_table --in-batches
```

The script will resume where it left off when run again, making it great for backfills.

## Connection Security

Always make sure your connection is secure when connecting to a database over a network you don't fully trust. Your best option is to connect over SSH or a VPN. Another option is to use `sslmode=verify-full`. If you don't do this, your database credentials can be compromised.

---

## Safety

To keep you from accidentally overwriting production, the destination is limited to `localhost` or `127.0.0.1` by default.

To use another host, add `to_safe: true` to your `.pgsync.yml`.

## Multiple Databases

To use with multiple databases, run:

```
1 pgsync --init db2
```

This creates `.pgsync-db2.yml` for you to edit. Specify a database in commands with:

```
1 pgsync --db db2
```

## Integrations

- Django
- Heroku
- Laravel
- Rails

### Django

If you run `pgsync --init` in a Django project, migrations will be excluded in `.pgsync.yml`.

```
1 exclude:  
2   - django_migrations
```

### Heroku

If you run `pgsync --init` in a Heroku project, the `from` database will be set in `.pgsync.yml`.

```
1 from: $(heroku config:get DATABASE_URL)?sslmode=require
```

---

## Laravel

If you run `pgsync --init` in a Laravel project, migrations will be excluded in `.pgsync.yml`.

```
1 exclude:
2   - migrations
```

## Rails

If you run `pgsync --init` in a Rails project, Active Record metadata and schema migrations will be excluded in `.pgsync.yml`.

```
1 exclude:
2   - ar_internal_metadata
3   - schema_migrations
```

## Debugging

To view the SQL that's run, use:

```
1 pgsync --debug
```

## Other Commands

Help

```
1 pgsync --help
```

Version

```
1 pgsync --version
```

List tables

```
1 pgsync --list
```

## Scripts

Use groups when possible to take advantage of parallelism.

For Ruby scripts, you may need to do:

---

```
1 Bundler.with_unbundled_env do
2   system "pgsync ..."
3 end
```

## Docker

Get the Docker image with:

```
1 docker pull ankane/pgsync
2 alias pgsync="docker run -ti ankane/pgsync"
```

This will give you the `pgsync` command.

## Dependencies

If installation fails, your system may be missing Ruby or libpq.

On Mac, run:

```
1 brew install libpq
```

On Ubuntu, run:

```
1 sudo apt-get install ruby-dev libpq-dev build-essential
```

## Upgrading

Run:

```
1 gem install pgsync
```

To use master, run:

```
1 gem install specific_install
2 gem specific_install https://github.com/ankane/pgsync.git
```

With Homebrew, run:

```
1 brew upgrade pgsync
```

With Docker, run:

```
1 docker pull ankane/pgsync
```



---

## Related Projects

Also check out:

- Dexter - The automatic indexer for Postgres
- PgHero - A performance dashboard for Postgres
- pgslice - Postgres partitioning as easy as pie

## Thanks

Inspired by [heroku-pg-transfer](#).

## History

[View the changelog](#)

## Contributing

Everyone is encouraged to help improve this project. Here are a few ways you can help:

- Report bugs
- Fix bugs and submit pull requests
- Write, clarify, or fix documentation
- Suggest or add new features

To get started with development:

```
1 git clone https://github.com/ankane/pgsync.git
2 cd pgsync
3 bundle install
4
5 createdb pgsync_test1
6 createdb pgsync_test2
7 createdb pgsync_test3
8
9 bundle exec rake test
```