h3-pg: Uber's H3 Hexagonal Hierarchical Geospatial Indexing System in PostgreSQL

This library provides PostgreSQL bindings for the H3 Core Library. For API reference, please see the H3 Documentation.

Developed in collaboration with Scandinavian Highlands.

Binary distributions

These don't require you have the development headers or cmake installed.

Debian/Ubuntu (Ubuntu 22.04 LTS (Jammy Jellyfish) +)

Replace 16 with your postgresql version. Refer to PGDG Ubuntu and PGDG Debian for installing PostgreSQL. More details about the various packages at https://apt.postgresql.org

```
sudo apt install postgresql-16-h3
```

Redhat Derivatives (Rocky / EL 8+ / Fedora 37+)

Replace 16 with your postgresql version Refer to PGDG Redhat Derivatives for installing PostgreSQL. More details about the various packages at https://yum.postgresql.org

```
sudo yum install h3-pg_16
```

Windows

Included as part of PostGIS Bundle 3.3+ for PostgreSQL 11-16 Details: postgis.net windows
Works with PostgreSQL Windows and PostGIS bundle is accessible via the included Application Stackbuilder.

Compiling Prerequisites

- PostgreSQL 11+ (*including server headers*, e.g. `postgresql-server-dev-14`)
- C compiler (e.g., `gcc`)
- CMake 3.20+
- GNU Make

Quick Overview

If the prerequisites are met you can use the PGXN Client to download, build, and install, e.g.:

```
$ pgxn install h3
$ pgxn load h3
$ psql
=# SELECT h3_lat_lng_to_cell(POINT('37.3615593,-122.0553238'), 5);
  h3_lat_lng_to_cell
-----------------
 85e35e73fffffff
(1 row)
```

(You can install a specific version using `pgxn install 'h3=3.7.2'` and `pgxn load 'h3=3.7.2'` for example)

See Building for other installation methods.

Usage

> :tada: **Note:** The following usage docs apply to **H3 v4**, which was released on August 23, 2022.
>
> - For v3 docs, see the latest v3.x.x release.
> - For breaking changes in v4, see the CHANGELOG. In particular, most function names have changed.

Generally, all functions have been renamed from camelCase in H3 to snake_case in SQL.

See API reference for all provided functions.

Building

```
# Generate native build system
cmake -B build -DCMAKE_BUILD_TYPE=Release

# Build extension(s)
cmake --build build

# Install extensions (might require sudo)
cmake --install build --component h3-pg
```

Contributing

Pull requests and GitHub issues are welcome. Please include tests for new work. Please note that the purpose of this extension is to expose the API of the H3 Core library, so we will rarely accept new features that are not part of that API. New proposed feature work is more appropriate in the core C library or in a new extension that depends on h3-pg.

See Development.

License

This project is released under the Apache 2.0 License. # API Reference The `h3` extension wraps the H3 Core Library. The detailed API reference is in the core H3 Documentation under the API Reference section. The `h3` core functions have been renamed from camelCase in H3 core to snake_case in SQL. The SQL function name is prefixed with `h3_`.

Base type

An unsigned 64-bit integer representing any H3 object (hexagon, pentagon, directed edge …) represented as a (or 16-character) hexadecimal string, like '8928308280fffff'.

Indexing functions

These function are used for finding the H3 index containing coordinates, and for finding the center and boundary of H3 indexes.

h3_lat_lng_to_cell(latlng `point`, resolution `integer`) → `h3index`

Since v4.0.0

See also: h3_lat_lng_to_cell(`geometry`, `integer`), h3_lat_lng_to_cell(`geography`, `integer`)

Indexes the location at the specified resolution.

h3_cell_to_lat_lng(cell `h3index`) → `point`

Since v4.0.0

See also: h3_cell_to_geometry(`h3index`), h3_cell_to_geography(`h3index`)

Finds the centroid of the index.

h3_cell_to_boundary(cell `h3index`) → `polygon`

Since v4.0.0

See also: h3_cell_to_boundary_geometry(`h3index`), h3_cell_to_boundary_geography(`h3index`)

Finds the boundary of the index.

Use `SET h3.extend_antimeridian TO true` to extend coordinates when crossing 180th meridian.

Index inspection functions

These functions provide metadata about an H3 index, such as its resolution or base cell, and provide utilities for converting into and out of the 64-bit representation of an H3 index.

h3_get_resolution(h3index) `integer`

Since v1.0.0

Returns the resolution of the index.

h3_get_base_cell_number(h3index) `integer`

Since v4.0.0

Returns the base cell number of the index.

h3_is_valid_cell(h3index) `boolean`

Since v1.0.0

Returns true if the given H3Index is valid.

h3_is_res_class_iii(h3index) `boolean`

Since v1.0.0

Returns true if this index has a resolution with Class III orientation.

h3_is_pentagon(h3index) `boolean`

Since v1.0.0

Returns true if this index represents a pentagonal cell.

h3_get_icosahedron_faces(h3index) `integer[]`

Since v4.0.0

Find all icosahedron faces intersected by a given H3 index.

Grid traversal functions

Grid traversal allows finding cells in the vicinity of an origin cell, and determining how to traverse the grid from one cell to another.

h3_grid_disk(origin `h3index`, [k `integer` = 1]) SETOF `h3index`

Since v4.0.0

Produces indices within "k" distance of the origin index.

h3_grid_disk_distances(origin `h3index`, [k `integer` = 1], OUT index `h3index`, OUT distance `int`) SETOF `record`

Since v4.0.0

Produces indices within "k" distance of the origin index paired with their distance to the origin.

h3_grid_ring_unsafe(origin `h3index`, [k `integer` = 1]) SETOF `h3index`

Since v4.0.0

Returns the hollow hexagonal ring centered at origin with distance "k".

h3_grid_path_cells(origin `h3index`, destination `h3index`) SETOF `h3index`

Since v4.0.0

See also: h3_grid_path_cells_recursive(`h3index`, `h3index`)

Given two H3 indexes, return the line of indexes between them (inclusive).

This function may fail to find the line between two indexes, for example if they are very far apart. It may also fail when finding distances for indexes on opposite sides of a pentagon.

h3_grid_distance(origin `h3index`, destination `h3index`) `bigint`

Since v4.0.0

Returns the distance in grid cells between the two indices.

h3_cell_to_local_ij(origin `h3index`, index `h3index`) `point`

Since v0.2.0

Produces local IJ coordinates for an H3 index anchored by an origin.

h3_local_ij_to_cell(origin `h3index`, coord `point`) `h3index`

Since v0.2.0

Produces an H3 index from local IJ coordinates anchored by an origin.

Hierarchical grid functions

These functions permit moving between resolutions in the H3 grid system. The functions produce parent (coarser) or children (finer) cells.

h3_cell_to_parent(cell `h3index`, resolution `integer`) `h3index`

Since v4.0.0

Returns the parent of the given index.

h3_cell_to_children(cell `h3index`, resolution `integer`) SETOF `h3index`

Since v4.0.0

Returns the set of children of the given index.

h3_cell_to_center_child(cell `h3index`, resolution `integer`) `h3index`

Since v4.0.0

Returns the center child (finer) index contained by input index at given resolution.

h3_compact_cells(cells `h3index[]`) SETOF `h3index`

Since v4.0.0

Compacts the given array as best as possible.

h3_cell_to_child_pos(child `h3index`, parentRes `integer`) `int8`

Since v4.1.0

Returns the position of the child cell within an ordered list of all children of the cells parent at the specified resolution parentRes. The order of the ordered list is the same as that returned by cellToChildren. This is the complement of childPosToCell.

h3_child_pos_to_cell(childPos `int8`, parent `h3index`, childRes `int`) `h3index`

Since v4.1.0

Returns the child cell at a given position within an ordered list of all children of parent at the specified resolution childRes. The order of the ordered list is the same as that returned by cellToChildren. This is the complement of cellToChildPos.

h3_uncompact_cells(cells `h3index[]`, resolution `integer`) SETOF `h3index`

Since v4.0.0

Uncompacts the given array at the given resolution.

h3_cell_to_parent(cell `h3index`) `h3index`

Since v4.0.0

Returns the parent of the given index.

h3_cell_to_children(cell `h3index`) SETOF `h3index`

Since v4.0.0

Returns the set of children of the given index.

h3_cell_to_center_child(cell `h3index`) `h3index`

Since v4.0.0

Returns the center child (finer) index contained by input index at next resolution.

h3_uncompact_cells(cells `h3index[]`) SETOF `h3index`

Since v4.0.0

Uncompacts the given array at the resolution one higher than the highest resolution in the set.

h3_cell_to_children_slow(index `h3index`, resolution `integer`) SETOF `h3index`

Since v4.0.0

Slower version of H3ToChildren but allocates less memory.

h3_cell_to_children_slow(index `h3index`) SETOF `h3index`

Slower version of H3ToChildren but allocates less memory.

Region functions

These functions convert H3 indexes to and from polygonal areas.

h3_polygon_to_cells(exterior `polygon`, holes `polygon[]`, [resolution `integer` = 1]) SETOF `h3index`

Since v4.0.0

See also: h3_polygon_to_cells(`geometry`, `integer`), h3_polygon_to_cells(`geography`, `integer`)

Takes an exterior polygon [and a set of hole polygon] and returns the set of hexagons that best fit the structure.

h3_cells_to_multi_polygon(`h3index[]`, OUT exterior `polygon`, OUT holes `polygon[]`) SETOF `record`

Since v4.0.0

See also: h3_cells_to_multi_polygon_geometry(`h3index[]`), h3_cells_to_multi_polygon_geography(`h3ind`
h3_cells_to_multi_polygon_geometry(setof `h3index`), h3_cells_to_multi_polygon_geography(setof `h3index`)

Create a LinkedGeoPolygon describing the outline(s) of a set of hexagons. Polygon outlines will follow GeoJSON MultiPolygon order: Each polygon will have one outer loop, which is first in the list, followed by any holes.

Unidirectional edge functions

Unidirectional edges allow encoding the directed edge from one cell to a neighboring cell.

h3_are_neighbor_cells(origin `h3index`, destination `h3index`) `boolean`

Since v4.0.0

Returns true if the given indices are neighbors.

h3_cells_to_directed_edge(origin `h3index`, destination `h3index`) `h3index`

Since v4.0.0

Returns a unidirectional edge H3 index based on the provided origin and destination.

h3_is_valid_directed_edge(edge `h3index`) `boolean`

Since v4.0.0

Returns true if the given edge is valid.

h3_get_directed_edge_origin(edge `h3index`) `h3index`

Since v4.0.0

Returns the origin index from the given edge.

h3_get_directed_edge_destination(edge `h3index`) `h3index`

Since v4.0.0

Returns the destination index from the given edge.

h3_directed_edge_to_cells(edge `h3index`, OUT origin `h3index`, OUT destination `h3index`) `record`

Since v4.0.0

Returns the pair of indices from the given edge.

h3_origin_to_directed_edges(`h3index`) SETOF `h3index`

Since v4.0.0

Returns all unidirectional edges with the given index as origin.

h3_directed_edge_to_boundary(edge `h3index`) `polygon`

Since v4.0.0

Provides the coordinates defining the unidirectional edge.

H3 Vertex functions

Functions for working with cell vertexes.

h3_cell_to_vertex(cell `h3index`, vertexNum `integer`) `h3index`

Since v4.0.0

Returns a single vertex for a given cell, as an H3 index.

h3_cell_to_vertexes(cell `h3index`) SETOF `h3index`

Since v4.0.0

Returns all vertexes for a given cell, as H3 indexes.

h3_vertex_to_lat_lng(vertex `h3index`) `point`

Since v4.0.0

Get the geocoordinates of an H3 vertex.

h3_is_valid_vertex(vertex `h3index`) `boolean`

Since v4.0.0

Whether the input is a valid H3 vertex.

Miscellaneous H3 functions

These functions include descriptions of the H3 grid system.

h3_great_circle_distance(a `point`, b `point`, [unit `text` = km]) `double precision`

Since v4.0.0

The great circle distance in radians between two spherical coordinates.

h3_get_hexagon_area_avg(resolution `integer`, [unit `text` = km]) `double precision`

Since v4.0.0

Average hexagon area in square (kilo)meters at the given resolution.

h3_cell_area(cell `h3index`, [unit `text` = km^2]) `double precision`

Since v4.0.0

Exact area for a specific cell (hexagon or pentagon).

h3_get_hexagon_edge_length_avg(resolution `integer`, [unit `text` = km]) `double precision`

Since v4.0.0

Average hexagon edge length in (kilo)meters at the given resolution.

h3_edge_length(edge `h3index`, [unit `text` = km]) `double precision`

Since v4.0.0

Exact length for a specific unidirectional edge.

h3_get_num_cells(resolution `integer`) `bigint`

Since v4.0.0

Number of unique H3 indexes at the given resolution.

h3_get_res_0_cells() SETOF `h3index`

Since v4.0.0

Returns all 122 resolution 0 indexes.

h3_get_pentagons(resolution `integer`) SETOF `h3index`

Since v4.0.0

All the pentagon H3 indexes at the specified resolution.

Operators

Operator: `h3index` <-> `h3index`

Since v3.7.0

Returns the distance in grid cells between the two indices (at the lowest resolution of the two).

B-tree operators

Operator: `h3index` = `h3index`

Since v0.1.0

Returns true if two indexes are the same.

Operator: `h3index` <> `h3index`

Since v0.1.0

R-tree Operators

Operator: `h3index` && `h3index`

Since v3.6.1

Returns true if the two H3 indexes intersect.

Operator: `h3index` @> `h3index`

Since v3.6.1

Returns true if A contains B.

Operator: `h3index` <@ `h3index`

Since v3.6.1

Returns true if A is contained by B.

Type casts

`h3index :: bigint`

Convert H3 index to bigint, which is useful when you need a decimal representation.

`bigint :: h3index`

Convert bigint to H3 index.

`h3index :: point`

Convert H3 index to point.

Extension specific functions

h3_get_extension_version() `text`

Since v1.0.0

Get the currently installed version of the extension.

h3_pg_migrate_pass_by_reference(`h3index`) `h3index`

Since v4.1.0

Migrate h3index from pass-by-reference to pass-by-value.

Deprecated functions

h3_cell_to_boundary(cell `h3index`, extend_antimeridian `boolean`) `polygon`

DEPRECATED: Use `SET h3.extend_antimeridian TO true` instead.

PostGIS Integration

The `GEOMETRY` data passed to `h3-pg` PostGIS functions should be in SRID 4326. This is an expectation of the core H3 library. Using other SRIDs, such as 3857, can result in either errors or invalid data depending on the function. For example, the `h3_polygon_to_cells()` function will fail with an error in this scenario while the `h3_lat_lng_to_cell()` function will return an invalid geometry.

PostGIS Indexing Functions

h3_lat_lng_to_cell(geometry, resolution integer) h3index

Since v4.0.0

Indexes the location at the specified resolution.

h3_lat_lng_to_cell(geography, resolution integer) h3index

Since v4.0.0

Indexes the location at the specified resolution.

h3_cell_to_geometry(h3index) geometry

Since v4.0.0

Finds the centroid of the index.

h3_cell_to_geography(h3index) geography

Since v4.0.0

Finds the centroid of the index.

h3_cell_to_boundary_geometry(h3index) geometry

Since v4.0.0

Finds the boundary of the index.

Splits polygons when crossing 180th meridian.

h3_cell_to_boundary_geography(h3index) geography

Since v4.0.0

Finds the boundary of the index.

Splits polygons when crossing 180th meridian.

PostGIS Grid Traversal Functions

**h3_grid_path_cells_recursive(origin h3index, destination h3index)
SETOF h3index**

Since v4.1.0

PostGIS Region Functions

h3_polygon_to_cells(multi `geometry`, resolution `integer`) SETOF `h3index`

Since v4.0.0

h3_polygon_to_cells(multi `geography`, resolution `integer`) SETOF `h3index`

Since v4.0.0

h3_cells_to_multi_polygon_geometry(`h3index[]`) `geometry`

Since v4.1.0

h3_cells_to_multi_polygon_geography(`h3index[]`) `geography`

Since v4.1.0

h3_cells_to_multi_polygon_geometry(setof `h3index`)

Since v4.1.0

h3_cells_to_multi_polygon_geography(setof `h3index`)

Since v4.1.0

PostGIS Operators

Operator: `geometry` @ `integer`

Since v4.1.3

Index geometry at specified resolution.

Operator: `geography` @ `integer`

Since v4.1.3

Index geography at specified resolution.

PostGIS casts

`h3index` :: `geometry`

Since v0.3.0

`h3index :: geography`

Since v0.3.0

WKB indexing functions

h3_cell_to_boundary_wkb(cell `h3index`) `bytea`

Since v4.1.0

Finds the boundary of the index, converts to EWKB.

Splits polygons when crossing 180th meridian.

This function has to return WKB since Postgres does not provide multipolygon type.

WKB regions functions

h3_cells_to_multi_polygon_wkb(`h3index[]`) `bytea`

Since v4.1.0

Create a LinkedGeoPolygon describing the outline(s) of a set of hexagons, converts to EWKB.

Splits polygons when crossing 180th meridian.

Raster processing functions

Continuous raster data

For rasters with pixel values representing continuous data (temperature, humidity, elevation), the data inside H3 cells can be summarized by calculating number of pixels, sum, mean, standard deviation, min and max for each cell inside a raster and grouping these stats across multiple rasters by H3 index.

```
SELECT
    (summary).h3 AS h3,
    (h3_raster_summary_stats_agg((summary).stats)).*
FROM (
    SELECT h3_raster_summary(rast, 8) AS summary
    FROM rasters
) t
GROUP BY 1;
       h3        | count |        sum         |        mean         |       stddev       |
-----------------+-------+--------------------+---------------------+--------------------+--
 882d638189fffff |    10 |  4.607657432556152 | 0.46076574325561526 | 1.3822972297668457 |
 882d64c4d1fffff |    10 | 3.6940908953547478 |  0.3694090895354748 |  1.099336879464068 |
```

```
 882d607431fffff |    11 |  6.219290263950825 |  0.5653900239955295 | 1.7624673707119065 |
<...>
```

Since v4.1.1

h3_raster_summary_stats_agg(setof `h3_raster_summary_stats`)

Since v4.1.1

h3_raster_summary_clip(rast `raster`, resolution `integer`, [nband `integer` = 1]) TABLE (h3 `h3index`, stats `h3_raster_summary_stats`)

Since v4.1.1

Returns `h3_raster_summary_stats` for each H3 cell in raster for a given band. Clips the raster by H3 cell geometries and processes each part separately.

h3_raster_summary_centroids(rast `raster`, resolution `integer`, [nband `integer` = 1]) TABLE (h3 `h3index`, stats `h3_raster_summary_stats`)

Since v4.1.1

Returns `h3_raster_summary_stats` for each H3 cell in raster for a given band. Finds corresponding H3 cell for each pixel, then groups values by H3 index.

h3_raster_summary_subpixel(rast `raster`, resolution `integer`, [nband `integer` = 1]) TABLE (h3 `h3index`, stats `h3_raster_summary_stats`)

Since v4.1.1

Returns `h3_raster_summary_stats` for each H3 cell in raster for a given band. Assumes H3 cell is smaller than a pixel. Finds corresponding pixel for each H3 cell in raster.

h3_raster_summary(rast `raster`, resolution `integer`, [nband `integer` = 1]) TABLE (h3 `h3index`, stats `h3_raster_summary_stats`)

Since v4.1.1

Returns `h3_raster_summary_stats` for each H3 cell in raster for a given band. Attempts to select an appropriate method based on number of pixels per H3 cell.

Discrete raster data

For rasters where pixels have discrete values corresponding to different classes of land cover or land use, H3 cell data summary can be represented by a JSON object with separate fields for each class. First, value, number of pixels and approximate area are calculated for each H3 cell and value in a raster, then the stats are grouped across multiple rasters by H3 index and value, and after that

stats for different values in a cell are combined into a single JSON object. The following example query additionally calculates a fraction of H3 cell pixels for each value, using a window function to get a total number of pixels:

```
WITH
    summary AS (
        -- get aggregated summary for each H3 index/value pair
        SELECT h3, val, h3_raster_class_summary_item_agg(summary) AS item
        FROM
            rasters,
            h3_raster_class_summary(rast, 8)
        GROUP BY 1, 2),
    summary_total AS (
        -- add total number of pixels per H3 cell
        SELECT h3, val, item, sum((item).count) OVER (PARTITION BY h3) AS total
        FROM summary)
SELECT
    h3,
    jsonb_object_agg(
        concat('class_', val::text),
        h3_raster_class_summary_item_to_jsonb(item)                 -- val, count, area
            || jsonb_build_object('fraction', (item).count / total) -- add fraction value
        ORDER BY val
    ) AS summary
FROM summary_total
GROUP BY 1;
      h3        |
----------------+-------------------------------------------------------------------------------
88194e6f3bfffff | {"class_1": {"area": 75855.5748, "count": 46, "value": 1, "fraction": 0.45
88194e6f37fffff | {"class_1": {"area": 255600.3064, "count": 155, "value": 1, "fraction": 0.
88194e6f33fffff | {"class_1": {"area": 336402.9840, "count": 204, "value": 1, "fraction": 0.
<...>
```

Area covered by pixels with the most frequent value in each cell:

```
SELECT DISTINCT ON (h3)
    h3, val, (item).area
FROM (
    SELECT
        h3, val, h3_raster_class_summary_item_agg(summary) AS item
    FROM
        rasters,
        h3_raster_class_summary(rast, 8)
    GROUP BY 1, 2
) t
ORDER BY h3, (item).count DESC;
       h3        | val |        area
```

```
----------------+-----+-------------------
 88194e6f3bfffff |   5 | 23238.699360251427
 88194e6f37fffff |   9 |  60863.26022922993
 88194e6f33fffff |   8 |  76355.72646939754
<...>
```

Since v4.1.1

h3_raster_class_summary_item_to_jsonb(item h3_raster_class_summary_item) jsonb

Since v4.1.1

Convert raster summary to JSONB, example: `{"count": 10, "value": 2, "area": 16490.3423}`

h3_raster_class_summary_item_agg(setof h3_raster_class_summary_item)

Since v4.1.1

h3_raster_class_summary_clip(rast raster, resolution integer, [nband integer = 1]) TABLE (h3 h3index, val integer, summary h3_raster_class_summary_item)

Since v4.1.1

Returns `h3_raster_class_summary_item` for each H3 cell and value for a given band. Clips the raster by H3 cell geometries and processes each part separately.

h3_raster_class_summary_centroids(rast raster, resolution integer, [nband integer = 1]) TABLE (h3 h3index, val integer, summary h3_raster_class_summary_item)

Since v4.1.1

Returns `h3_raster_class_summary_item` for each H3 cell and value for a given band. Finds corresponding H3 cell for each pixel, then groups by H3 and value.

h3_raster_class_summary_subpixel(rast raster, resolution integer, [nband integer = 1]) TABLE (h3 h3index, val integer, summary h3_raster_class_summary_item)

Since v4.1.1

Returns `h3_raster_class_summary_item` for each H3 cell and value for a given band. Assumes H3 cell is smaller than a pixel. Finds corresponding pixel for each H3 cell in raster.

h3_raster_class_summary(rast raster, resolution integer, [nband integer = 1]) TABLE (h3 h3index, val integer, summary h3_raster_class_summary_item)

Since v4.1.1

Returns `h3_raster_class_summary_item` for each H3 cell and value for a given band. Attempts to select an appropriate method based on number of pixels per H3 cell.

Development

In order to build and test your changes, simply run `./scripts/develop`.

Documentation is generated from the sql files, using the script `scripts/documentaion` (requires poetry).

Release Process

1. Update version number
   - Don't follow semver, simply use major and minor from H3 core and increment patch.
   - Version number should be changed in root `CMakeLists.txt`.
   - Set `INSTALL_VERSION` to "${PROJECT_VERSION}".
   - Update files (and cmake references) suffixed `--unreleased` should be renamed.
   - Installer `.sql` files should have `@ availability` comments updated.
   - Update changelog by moving from `Unreleased` to a new section
   - Push and merge changes in `release-x.y.z` branch.
2. Create a release on GitHub
   - Draft new release "vX.Y.Z"
   - Copy CHANGELOG.md entry into release description
3. Distribute the extension on PGXN
   - Run `scripts/bundle` to package the release
   - Upload the distribution on PGXN Manager (username: `bytesandbrains`)
4. Prepare for development
   - Set `INSTALL_VERSION` to `unreleased` in root `CMakeLists.txt`.
   - Create new update files with `--unreleased` suffix.
   - Add them to relevant `CMakeLists.txt` files. # PGXN Client

   `pgxnclient` does not normally ship with PostgreSQL, so you will probably have to install it manually.

The PGXN Client is a command line tool designed to interact with the PostgreSQL Extension Network allowing searching, compiling, installing, and removing extensions in PostgreSQL databases.

Ubuntu

On Ubuntu you can install using `apt`

```
apt-get install pgxnclient
```

MacOS

On MacOs you can install using `brew`

```
brew install pgxnclient
```