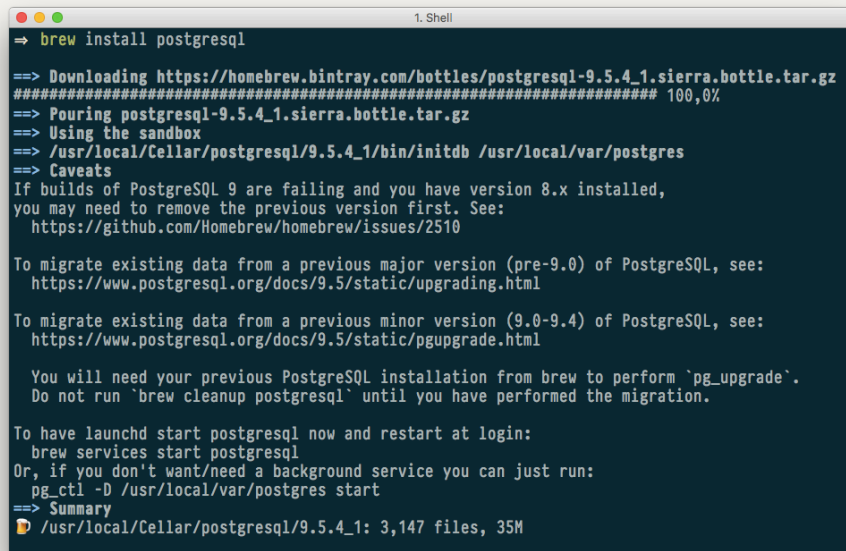


A first contact with PostgreSQL.

Install.



```
1. Shell
=> brew install postgresql
=> Downloading https://homebrew.bintray.com/bottles/postgresql-9.5.4_1.sierra.bottle.tar.gz
##### 100,0%
=> Pouring postgresql-9.5.4_1.sierra.bottle.tar.gz
=> Using the sandbox
=> /usr/local/Cellar/postgresql/9.5.4_1/bin/initdb /usr/local/var/postgres
=> Caveats
If builds of PostgreSQL 9 are failing and you have version 8.x installed,
you may need to remove the previous version first. See:
https://github.com/Homebrew/homebrew/issues/2510

To migrate existing data from a previous major version (pre-9.0) of PostgreSQL, see:
https://www.postgresql.org/docs/9.5/static/upgrading.html

To migrate existing data from a previous minor version (9.0-9.4) of PostgreSQL, see:
https://www.postgresql.org/docs/9.5/static/pgupgrade.html

You will need your previous PostgreSQL installation from brew to perform `pg_upgrade`.
Do not run `brew cleanup postgresql` until you have performed the migration.

To have launchd start postgresql now and restart at login:
brew services start postgresql
Or, if you don't want/need a background service you can just run:
pg_ctl -D /usr/local/var/postgres start
=> Summary
📦 /usr/local/Cellar/postgresql/9.5.4_1: 3,147 files, 35M
```

1.- Role, Database, Conection.

En *createuser*, **-s** nos sirve para darle los permisos de un superusuario al usuario que estamos creando.

En *createuser*, **-P** nos sirve para obligar al usuario a definir su contraseña.

En *psql*, **-d** nos sirve para indicar que el siguiente parámetro es el nombre de la base de datos.

En *psql*, **-W** nos sirve para obligar la solicitud del password.

En *psql*, **-U** nos sirve para indicar que el siguiente parámetro es el nombre del usuario.

```
1. Shell
~) • ~/Documents • % •
⇒ createdb BankDB

~) • ~/Documents • % •
⇒ createuser -s -P DanielSN

Enter password for new role:
Enter it again:

~) • ~/Documents • % •
⇒ psql -d BankDB -WU DanielSN

Password for user DanielSN:
psql (9.5.4)
Type "help" for help.

BankDB=# \du

          List of roles
Role name | Attributes | Member of
-----|-----|-----
DanielSN | Superuser, Create role, Create DB | {}
gophre   | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
prueba   | Create role, Create DB | {}

BankDB=#
```

2.- BankDB.

Write a SQL script.

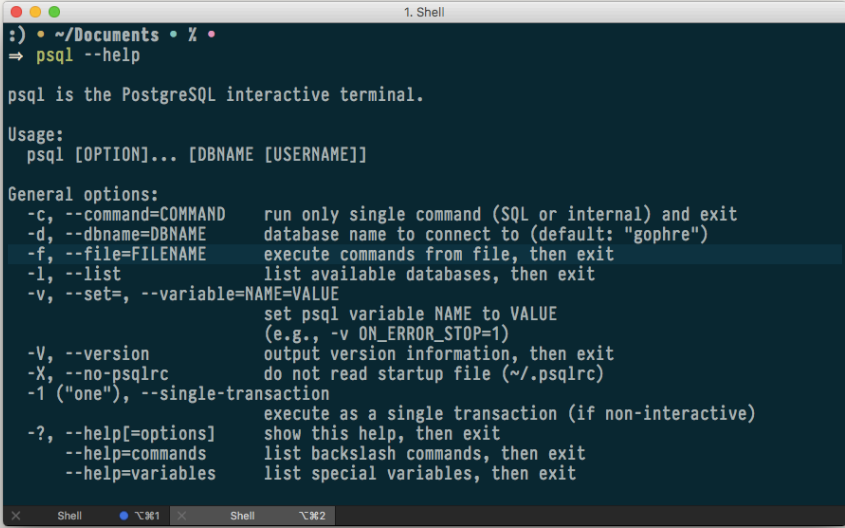
```
1 CREATE TABLE bank (
2     bank_id integer NOT NULL,
3     bank_name text,
4     bank_address text,
5     PRIMARY KEY(bank_id)
6 );
7 CREATE TABLE branch (
8     branch_id integer NOT NULL,
9     bank_fk integer REFERENCES bank(bank_id),
10    branch_address text,
11    branch_tlf numeric,
12    PRIMARY KEY(branch_id, bank_fk)
13 );
14 CREATE TABLE customer (
15     customer_id integer NOT NULL,
16     customer_ssn numeric UNIQUE NOT NULL,
17     customer_name text,
18     customer_address text,
19     PRIMARY KEY(customer_id)
20 );
21 CREATE TYPE account_currency_enum AS ENUM ( '€',
        '$', '£');
```

```

22 CREATE TYPE account_type_enum AS ENUM ( 'standard',
      'plus', 'premium' );
23 CREATE TABLE account (
24     customer_id integer REFERENCES
      customer(customer_id),
25     account_id integer NOT NULL,
26     account_type account_type_enum NOT NULL,
27     account_currency account_currency_enum NOT
      NULL,
28     balance numeric,
29     PRIMARY KEY(account_id),
30     UNIQUE(customer_id, account_id)
31 );
32 CREATE TYPE transaction_type_enum AS ENUM (
33     'purchase', 'loan', 'montage', 'bank',
      'credit', 'debit' );
34 CREATE TABLE transaction (
35     account_fk integer REFERENCES
      account(account_id),
36     transaction_date timestamp WITH time zone,
37     transaction_type transaction_type_enum,
38     amount numeric,
39     PRIMARY KEY(account_fk, transaction_date)
40 );

```

Load the script.



```

1. Shell
~/.Documents • %
⇒ psql --help

psql is the PostgreSQL interactive terminal.

Usage:
  psql [OPTION]... [DBNAME [USERNAME]]

General options:
-c, --command=COMMAND      run only single command (SQL or internal) and exit
-d, --dbname=DBNAME        database name to connect to (default: "gophre")
-f, --file=FILENAME        execute commands from file, then exit
-l, --list                  list available databases, then exit
-v, --set=, --variable=NAME=VALUE
                           set psql variable NAME to VALUE
                           (e.g., -v ON_ERROR_STOP=1)
-V, --version              output version information, then exit
-X, --no-psqlrc            do not read startup file (~/.psqlrc)
-1 ("one"), --single-transaction
                           execute as a single transaction (if non-interactive)
-?, --help[=options]      show this help, then exit
--help=commands            list backslash commands, then exit
--help=variables           list special variables, then exit

```

```
1. Shell

:) • SN/Postgres • master % U •
⇒ k

total 40
-rw-r--r-- 1 gophre staff 8,1K 5 nov 04:12 | .DS_Store
-rw-r--r-- 1 gophre staff 1,6K 5 nov 04:16 + Lab5.md
-rw-r--r-- 1 gophre staff 1,3K 5 nov 04:16 + bank.psql
drwxr-xr-x 5 gophre staff 170 5 nov 04:05 | screenshots

:) • SN/Postgres • master % U •
⇒ psql -d BankDB -f bank.psql

CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TYPE
CREATE TYPE
CREATE TABLE
CREATE TYPE
CREATE TABLE

:) • SN/Postgres • master % U •
⇒
```

3.- Using PSYCOPG2.

Install.

```
1. Shell

:( • SN/Postgres • master % U •
⇒ pip install psycpg2

Collecting psycpg2
  Downloading psycpg2-2.6.2.tar.gz (376kB)
    100% |#####| 378kB 1.6MB/s
Building wheels for collected packages: psycpg2
  Running setup.py bdist_wheel for psycpg2 ... done
  Stored in directory: /Users/dcl/Library/Caches/pip/wheels/49/47/2a/5c3f874990ce267228c2dfe7a0589f3b0651aa590e329ad382
Successfully built psycpg2
Installing collected packages: psycpg2
Successfully installed psycpg2-2.6.2
You are using pip version 8.1.2, however version 9.0.0 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.

:) • SN/Postgres • master % U •
⇒ pip install --upgrade pip

Collecting pip
  Downloading pip-9.0.0-py2.py3-none-any.whl (1.3MB)
    100% |#####| 1.3MB 713kB/s
Installing collected packages: pip
  Found existing installation: pip 8.1.2
  Uninstalling pip-8.1.2:
    Successfully uninstalled pip-8.1.2
Successfully installed pip-9.0.0

:) • SN/Postgres • master % U •
⇒
```

Write a script.

```
1  import psycopg2
2
3  # Abre la conexion con la db y un usuario
   concreto.
4  conn = psycopg2.connect("dbname=BankDB
   user=DanielSN")
5  # Concede un 'cursor' con el que trabajar contra
   la base de datos.
6  cur = conn.cursor()
7
8  # Crea una nueva tabla.
9  cur.execute("CREATE TABLE test (id serial PRIMARY
   KEY, num integer, data varchar);")
10 # A la tabla anterior, introduce datos.
11 cur.execute("INSERT INTO test (num, data) VALUES
   (%s, %s)", (100, "abc'def"))
12 # Obten una lista en base al 'Select'.
13 cur.execute("SELECT * FROM test;")
14 # Iterando sobre la lista previa, retorna el
   primer elemento.
15 cur.fetchone()
16
17 # Convierte en persistentes los cambios realizados
   en la base de datos.
18 conn.commit()
19
20 # Cerrar el cursor y la propia conexion.
21 cur.close()
22 conn.close()
```

Run script.

```
⇒ k

total 48
-rw-r--r-- 1 gophre staff 8,1K 5 nov 04:20 | .DS_Store
-rw-r--r-- 1 gophre staff 3,0K 5 nov 04:43 + Lab5.md
-rw-r--r-- 1 gophre staff 1,3K 5 nov 04:16 + bank.psql
-rw-r--r-- 1 gophre staff 749 5 nov 04:42 + psy.py
drwxr-xr-x 7 gophre staff 238 5 nov 04:21 | screenshots

:) • SN/Postgres • master % U •
⇒ py psy.py

:) • SN/Postgres • master % U •
⇒ psql -d BankDB -WU DanielSN

Password for user DanielSN:
psql (9.5.4)
Type "help" for help.

BankDB=# \dt
          List of relations
Schema | Name      | Type  | Owner
-----+-----+-----+-----
public | account   | table | gophre
public | bank      | table | gophre
public | branch    | table | gophre
public | customer  | table | gophre
public | test      | table | DanielSN
public | transaction | table | gophre
(6 rows)

BankDB=#
```

4.- SELECTs y TRIGGER.

Script PL/psql.

```
1  -- ...DEFINICION BASE DE DATOS...
2  -- Creacion de tablas.
3  create table coche (
4      id_coche integer not null,
5      marca varchar(30),
6      modelo Varchar(30),
7      color varchar(30),
8      primary key( id_coche)
9  );
10 create table venta (
11     id_venta integer not null,
12     id_coche integer,
13     precio integer,
14     fecha_venta varchar(30),
15     primary key (id_venta)
16 );
17 create table stock (
```

```

18     id_coche integer not null,
19     disponible integer,
20     primary key(id_coche)
21 );
22 -- Inserccion de datos de prueba.
23 insert into coche (id_coche, marca, modelo, color)
24     values (0001, 'Citroen', 'Picaso', 'Blanco');
25 insert into coche (id_coche, marca, modelo, color)
26     values (0002, 'Citroen', 'Picaso', 'Negro');
27 insert into coche (id_coche, marca, modelo, color)
28     values (0003, 'Citroen', 'Picaso', 'Rojo');
29 insert into coche (id_coche, marca, modelo, color)
30     values (0004, 'Citroen', 'Picaso', 'Azul');
31 insert into coche (id_coche, marca, modelo, color)
32     values (0005, 'Citroen', 'Picasa', 'Rojo');
33 insert into stock (id_coche, disponible)
34     values (0001, 8);
35 insert into venta (id_venta, id_coche, precio,
36     fecha_venta)
37     values (1,0001,8000,'03/10/2012');
38 insert into venta (id_venta, id_coche, precio,
39     fecha_venta)
40     values (2,0001,8000,'07/10/2012');
41 insert into venta (id_venta, id_coche, precio,
42     fecha_venta)
43     values (3,0002,8000,'03/10/2015');
44
45 -- ...SELECTS_BASICOS...
46 -- Seleccionar todos los coches rojos.
47 select *
48 from coche
49 where color = 'Rojo';
50
51 -- Seleccionar el id de todos los coches
52 vendidos el año pasado.
53 select coche.id_coche
54 from coche
55 left join venta on coche.id_coche = venta.id_coche
56 where fecha_venta like '%2015';
57
58 -- ...TRIGGER...
59 -- DEFINICION TRIGGER.
60 -- Funcion a lanzar por el trigger.
61 CREATE OR REPLACE FUNCTION venta_menos_stock()
62 RETURNS TRIGGER AS $$
63 BEGIN
64     UPDATE stock
65     SET disponible = disponible - 1

```



```

61     WHERE stock.id_coche = new.id_coche;
62     RETURN new;
63 END;
64 $$ language plpgsql;
65
66 -- Trigger a lanzar cuando se registra una nueva
    venta.
67 CREATE TRIGGER nueva_venta
68 AFTER INSERT ON venta
69 FOR EACH ROW
70 EXECUTE PROCEDURE venta_menos_stock();
71
72 -- COMPROBACION.
73 -- Comprobamos el estado actual de ambas tablas.
74 select * from stock;
75 select * from venta;
76
77 -- Hacemos una inserccion en venta para activar el
    trigger.
78 insert into venta (id_venta, id_coche, precio,
    fecha_venta) values (5,0001,8000,'07/10/2012');
79
80 -- Comprobamos ambas tablas tras la inserccion,
    comprando el trigger.
81 select * from stock;
82 select * from venta;

```

Verificación del funcionamiento del trigger.

```
-----
      2
(1 row)

CREATE FUNCTION
CREATE TRIGGER
  id_coche | disponible
-----+-----
      1 |          8
(1 row)

  id_venta | id_coche | precio | fecha_venta
-----+-----+-----+-----
      1 |          1 |    8000 | 03/10/2012
      2 |          1 |    8000 | 07/10/2012
      3 |          2 |    8000 | 03/10/2015
(3 rows)

INSERT 0 1
  id_coche | disponible
-----+-----
      1 |          7
(1 row)

  id_venta | id_coche | precio | fecha_venta
-----+-----+-----+-----
      1 |          1 |    8000 | 03/10/2012
      2 |          1 |    8000 | 07/10/2012
      3 |          2 |    8000 | 03/10/2015
      5 |          1 |    8000 | 07/10/2012
(4 rows)

:) • SN/Postgres • master % U •
=> |
```