**PostgreSQL**

Установка posgresql:

pip install psycopg2-binary

brew install postgres

pg\_ctl -D /usr/local/var/postgres start | stop | restart | status

psql postgres

Список баз данных:

psql -l

Либо:  
select \* from pg\_database;

# \l

Список пользователей:

select \* from pg\_shadow;

# \du

Создание базы данных:

CREATE DATABASE sysadmin;

Далее вам необходимо создать нового пользователя базы данных (sysadminuser) для того, чтобы подключиться к базе данных и взаимодействовать с ней. Обязательно установите длинный и сложный пароль:

CREATE USER sysadminuser WITH PASSWORD '1234';

Установите UTF-8 в качестве кодировки по умолчанию (это необходимо для Django). Уровень изолированности транзакций нужно по умолчанию установить как “read committed” (чтение фиксированных данных) – этот уровень обеспечивает защиту от чернового чтения. Не забудьте про временную зону: изначально проекты на Django имеют установку UTC. Чтобы выполнить все изменения, нужно ввести следующие команды:

ALTER ROLE sysadminuser SET client\_encoding TO 'utf8';

ALTER ROLE sysadminuser SET default\_transaction\_isolation TO 'read committed';

ALTER ROLE sysadminuser SET timezone TO 'UTC';

Теперь последнее, что необходимо сделать, это дать только что созданному пользователю права доступа к созданной базе:

GRANT ALL PRIVILEGES ON DATABASE sysadmin TO sysadminuser;

Заходим в наш Django проект и в файле settings.py меняем sqlite3 на postgresql:

DATABASES = {  
 **'default'**: {  
 **'ENGINE'**: **'django.db.backends.postgresql'**,  
 **'NAME'**: **'sysadmin'**,  
 **'USER'**: **'sysadminuser'**,  
 **'PASSWORD'**: **'1234'**,  
 **'HOST'**: **'localhost'**,  
 **'PORT'**: **''**,  
 }  
}

Выполняем минрации:

python manage.py makemigrations

python manage.py migrate

Подключение к базе данных databasename

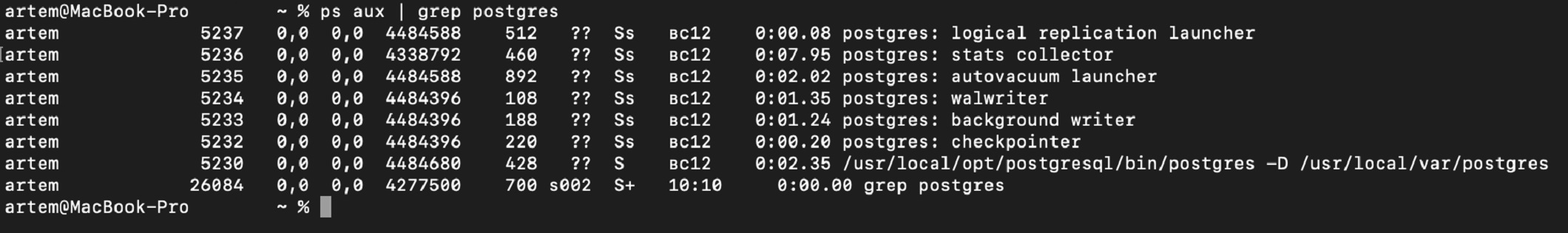
\c sysadmin;

Переименовать базуданных:

alter database insys rename to insys\_old;

**Конфиги postgresql**

ps aux | grep postgres



В каталоге /usr/local/var/postgres находим файл pg\_hba.conf

**Отключить подключение к базе данных без пароля при локальном подключении**

В файле pg\_hba.conf в строчке ‘IPV6 local connections:’ устанавливаем md5



Для подключения к postgres пишем в консоли:

psql postgres

Список баз данных:

\l

Подключение к базе данных:

\connect sysadmin

Получение списка имеющихся таблиц:

\dt

Просмотр содержимого таблицы:

SELECT \*

FROM account\_profile;

Удаление из таблицы:

DELETE FROM django\_migrations

WHERE app='account';

получить список имен столбцов и типов данных таблицы

\d+ blog\_post

1. sudo apt install postgresql

2. python -m pip install psycopg2-binary

3. sudo su - postgres

4. psql

5. create role djangouser with login password '1234';

6. create database namedb with owner djangouser;

7. \q

8. exit

9. psql namedb -h localhost -U djangouser

10. Изменить настройки в settings.py:

DATABASES = {  
 **'default'**: {  
 **'ENGINE'**: **'django.db.backends.postgresql\_psycopg2'**,  
 **'NAME'**: **'inportal'**,  
 **'USER'**: **'djangouser'**,  
 **'PASSWORD'**: os.getenv(**'SECRET\_KEY\_DB'**),  
 **'HOST'**: **'localhost'**,  
 **'PORT'**: **'5432'**,  
 }  
}

11. python manage.py migrate

**Установка пароля для postgresql**

userpc@userpc:~$ sudo su — postgres

postgres@userpc:~$ psql

postgres-# \password postgres

Enter new password:

Enter it again:

**Резервное копирование базы данных**

**pg\_dump** — подходит для случаев когда нужно сделать резервную копию таблицы, базы, схемы или данных.

**pg\_basebackup** — подходит для случаев когда нужно сделать резервную копию целиком всего кластера

pg\_basebackup -h localhost -U postgres -D ./BackUp/

-U, --username=NAME подключение в качестве указанного пользователя базы данных

-D, --pgdata=DIRECTORY получить базовую резервную копию в каталог

Создаем бэкап с помощью pg\_dump:

pg\_dump -U postgres -W -h localhost insys\_test > ./BackUp/outfile

(sudo -u postgres pg\_dump -Fc name\_db > name.db.dump)

Перед восстановлением создаем базу данных «insys\_test»:

postgres=# create database insys\_test with owner djangouser;

Для восстановления из бэкапа выполняем:

userpc@userpc:~$ psql -U postgres -W -h localhost insys\_test < ./BackUp/outfile

pg\_restore -U postgres -W -h localhost -d org ./backups/org.db.dump

Ссылки:

1. https://wiki.dieg.info/rezervnoe\_kopirovanie\_i\_vosstanovlenie\_v\_postgresql

**Настройка автоматизированного резервного копирования баз данных**

1. Создаем скрипт (к примеру в каталоге /usl/local/bin), в котором пишем:

#! /bin/bash

# backup-postgresql.sh

# by Craig Sanders &lt;cas@taz.net.au&gt;

# This script is public domain. feel free to use or modify

# as you like.

DUMPALL='/usr/bin/pg\_dumpall'

PGDUMP='/usr/bin/pg\_dump'

PSQL='/usr/bin/psql'

# directory to save backups in, must be rwx by postgres user

BASE\_DIR='/var/backups/postgres'

YMD=$(date "+%Y-%m-%d")

DIR="$BASE\_DIR/$YMD"

mkdir -p "$DIR"

cd "$DIR"

# get list of databases in system , exclude the tempate dbs

DBS=( $($PSQL --list --tuples-only | awk '!/template[01]/ && $1 != "|" {print $1}') )

# first dump entire postgres database, including pg\_shadow etc.

$DUMPALL --column-inserts | gzip -9 > "$DIR/db.out.gz"

# next dump globals (roles and tablespaces) only

$DUMPALL --globals-only | gzip -9 > "$DIR/globals.gz"

# now loop through each individual database and backup the

# schema and data separately

for database in "${DBS[@]}" ; do

SCHEMA="$DIR/$database.schema.gz"

DATA="$DIR/$database.data.gz"

INSERTS="$DIR/$database.inserts.gz"

# export data from postgres databases to plain text:

# dump schema

$PGDUMP --create --clean --schema-only "$database" |

gzip -9 > "$SCHEMA"

# dump data

$PGDUMP --disable-triggers --data-only "$database" |

gzip -9 > "$DATA"

# dump data as column inserts for a last resort backup

$PGDUMP --disable-triggers --data-only --column-inserts \

"$database" | gzip -9 > "$INSERTS"

done

# delete backup files older than 30 days

echo deleting old backup files:

find "$BASE\_DIR/" -mindepth 1 -type d -mtime +30 -print0 |

xargs -0r rm -rfv

3. Делаем владельцем пользователя postgres и даем права на исполнение.

sudo chown postgres /usl/local/bin/backup\_postgres.sh

sudo chmod 755 /usl/local/bin/backup\_postgres.sh

4. Создаем каталог для backup:

mkdir /var/backups/postgres

5. Делаем владельцем этого каталога пользователя postgres:

chown -R postgres /var/backups/postgres

6. Редактируем crontab для postgres

sudo crontab -u postgres -e

0 21 \* \* \* /usr/local/bin/backup\_postgresql.sh

| | | | |

| | | | ---- день недели (0 - 7) (воскресенье = 0 или 7)

| | | ------- месяц (1 - 12)

| | --------- день (1 - 31)

| ------------ час (0 - 23)

---------------- минута (0 - 59)

Чтобы файлы из backup хранились еще на удаленном сервере, делаем авторизацию по ключу ssh и дописываем в скрип команду rsync, либо на удаленном сервере создаем свой cron:

vi /ect/cron.d/backup\_postgresql\_cron

В нем пишем:

0 23 \* \* \* /usr/local/bin/backup\_postgresql.sh

И создаем сам скрипт:

vi /usr/local/bin/backup\_postgresql.sh

В нем пишем:

#! /bin/bash

# backup-postgresql.sh

BASE\_DIR='/var/backups/postgres'

rsync -avp [user@ip\_address](mailto:user@ip_address):/var/backups/postgres/ /var/backups/postgres

# delete backup files older than 30 days

echo deleting old backup files:

find "$BASE\_DIR/" -mindepth 1 -type d -mtime +30 -print0 |

xargs -0r rm -rfv