PostgreSQL

PostgreSQL Information

**Note: If you don't have the Utopia solution cloned to your local machine, follow these instructions

Installation and Setup

- In order to get a postgres db up and running on your local machine, you first need to install PostgreSQL.
- You can get the installer from the site or from this eFile quick link . Download v13.2 for Windows.
- During installation it will ask you for a password. Feel free to make this any password you like, but be sure to remember it, as you will need to use it to connect to PgAdmin.
- **DO NOT** just download PgAdmin 4. This will install the PgAdmin client, but you will not have a running PostgreSQL server on your local machine. If this applies to you, just follow the link above to actually install PostgreSQL and not just PgAdmin.
- After the installation is complete, the Pg Stack Builder will open asking if you want to install any other dev tools. You can just close it.

Connecting to your Server

- Once the installation has finished, open PgAdmin - you can find it by typing PgAdmin in your taskbar search.
- The installer should have automatically configured your local server and named it PostgreSQL 13. If you do see this server, right click on it and click properties.
- Once there, click connection and make sure the username is postgres.
- If it asks you for a password to view the server, type pass@word1
- If the password is **not** pass@word1 for your server, you will need to change it
- If you do not see a connected server, you can create your own connection to your local server by right clicking on Servers, then hovering over Create, and selecting server.
- In the name, you can call it PostgreSQL 13, though it doesn't really matter. On the connection tab, the host is localhost, the port is 5432, the username is postgres, and the password is pass@word1.
- Click save, and if you get any errors, it simply means that your postgres wasn't installed correctly and you can start over by following the link in the Installation and Setup section.

Creating your Database

- Open the Utopia solution. Make sure you've fetched the most recent changes for the branch you're on.
- Set the startup project as DBMigratorEF, make sure Azure Storage Emulator is running, and click run.
- If there are issues, it's probably because your server password is incorrect. If you can't change your password, you can actually just change your configuration data string using the utopia configuration tool to have the password of your PG server.
- Once that's done running, you can open PgAdmin and verify that inside the Schemas of your postgres database, you now have public and workflows. Yay, you have a configured database!
- **Important:** To make sure there are no conflicts with the old database, delete the UtopiaDB database. You should now only have the postgres database.
- Now open the root folder of the utopia source code and copy the contents of the DB Instance Settings Local.config.sql file. Paste the contents in a new query for PgAdmin in the postgres DB and click run.
- Once the query is done, your DB is configured and you can run the batch worker and reset your password.
- Celebrate! Your DB should be good to go!

Changing the postgres user's password

If you accidentally didn't set the postgres user's password to our standard *pass@word1*, then here are steps to help you fix it.

1. Open Command Prompt and navigate to the PostgreSQL bin folder on your machine. Assuming you didn't change the default installation directory, the path should be C:\Program Files\PostgreSQL\13\bin
2. Once there, you can use the psql command (another way to do this is to add that bin folder path to your environment PATH variable). Type the following command: `psql -U postgres`
3. You will be asked for the current password of the pstgress user. Enter it to continue.
4. Type `alter user postgres with password 'pass@word1'`, and you can substitute pass@word1 with any password you happen to need.
5. The command `\q` exits you back to the regular command line. Yay, your postgres user password will now be what you want!

Distribute Foreign Keys

When it's time to distribute the foreign keys in postgres:

```
select 'alter table '||quote_ident(ns.nspname)||'.'||quote_ident(tb.relname)||
' drop constraint '||quote_ident(conname)||';'||chr(10)||
'alter table '||quote_ident(ns.nspname)||'.'||quote_ident(tb.relname)||
' add constraint '||quote_ident(conname)||' '||
pg_get_constraintdef(c.oid, true)||';' as ddl
from pg_constraint c
join pg_class tb on tb.oid = c.conrelid
join pg_namespace ns on ns.oid = tb.relnamespace
where ns.nspname in ('public') --<<< adjust the schema name(s) here
and c.contype = 'f'; (edited)
```

Pulled from: https://dba.stackexchange.com/questions/125578/recreate-all-foreign-keys-in-all-tables-as-deferrable-batch

Backup and Restore

See pgAdmin Backup and restore.pdf that Ty Toon created for RPC backups. Most of the steps should work for a local backup. See the differences below. The backup will be saved to the file path and name saved in the backup step.

The Key Difference:

- You'll see different binary path options, and the path should be something like: C:\Program Files\PostgreSQL\14\bin image.png Image not found or type unknown
- In the backup steps you'll see Data/Objects instead of Dump options image.png Image not found or type unknown