

Setting up your own Server

You do not need the server to do anything with hydrus! It is only for advanced users to do very specific jobs! The server is also hacked-together and quite technical. It requires a fair amount of experience with the client and its concepts, and it does not operate on a timescale that works well on a LAN. Only try running your own server once you have a bit of experience synchronising with something like the PTR and you think, 'Hey, I know exactly what that does, and I would like one!'

Here is a document put together by a user describing whether you want the server.

setting up a server

I will use two terms, *server* and *service*, to mean two distinct things:

- A **server** is an instantiation of the hydrus server executable (e.g. server.exe in Windows). It has a complicated and flexible database that can run many different services in parallel.
- A **service** sits on a port (e.g. 45871) and responds to certain http requests (e.g. /file or /update) that the hydrus client can plug into. A service might be a repository for a certain kind of data, the administration interface to manage what services run on a server, or anything else.

Setting up a hydrus server is easy compared to, say, Apache. There are no .conf files to mess about with, and everything is controlled through the client. When started, the server will place an icon in your system tray in Windows or open a small frame in Linux or macOS. To close the server, either right-click the system tray icon and select exit, or just close the frame.

The basic process for setting up a server is:

- Start the server.
- Set up your client with its address and initialise the admin account
- Set the server's options and services.
- Make some accounts for your users.
- ???
- Profit

Let's look at these steps in more detail:

start the server

Since the server and client have so much common code, I package them together. If you have the client, you have the server. If you installed in Windows, you can hit the shortcut in your start menu. Otherwise, go straight to 'server' or 'server.exe' or 'server.pyw' in your installation directory. The program will first try to take port 45870 for its administration interface, so make sure that is free. Open your firewall as appropriate.

_client

set up the client

In the *services->manage services* dialog, add a new 'hydrus server administration service' and set up the basic options as appropriate. If you are running the server on the same computer as the client, its hostname is 'localhost'.

In order to set up the first admin account and an access key, use 'init' as a registration key. This special registration key will only work to initialise this first super-account.

YOU'LL WANT TO SAVE YOUR ACCESS KEY IN A SAFE PLACE

If you lose your admin access key, there is no way to get it back, and if you are not sqlite-proficient, you'll have to restart from the beginning by deleting your server's database files.

If the client can't connect to the server, it is either not running or you have a firewall/port-mapping problem. If you want a quick way to test the server's visibility, just put `https://host:port` into your browser (make sure it is https! http will not work)--if it is working, your browser will probably complain about its self-signed https certificate. Once you add a certificate exception, the server should return some simple html identifying itself.

set up the server

You should have a new submenu, 'administrate services', under 'services', in the client gui. This is where you control most server and service-wide stuff.

admin->your server->manage services lets you add, edit, and delete the services your server runs. Every time you add one, you will also be added as that service's first administrator, and the admin menu will gain a new entry for it.

making accounts

Go *admin->your service->create new accounts* to create new registration keys. Send the registration keys to the users you want to give these new accounts. A registration key will only work once, so if you want to give several people the same account, they will have to share the access key amongst themselves once one of them has registered the account. (Or you can register the account yourself and send them all the same access key. Do what you like!)

Go *admin->manage account types* to add, remove, or edit account types. Make sure everyone has at least downloader (get_data) permissions so they can stay synchronised.

You can create as many accounts of whatever kind you like. Depending on your usage scenario, you may want to have all uploaders, one uploader and many downloaders, or just a single administrator. There are many combinations.

???

The most important part is to have fun! There are no losers on the INFORMATION SUPERHIGHWAY.

profit

I honestly hope you can get some benefit out of my code, whether just as a backup or as part of a far more complex system. Please mail me your comments as I am always keen to make improvements.

btw, how to backup a repo's db

All of a server's files and options are stored in its accompanying .db file and respective subdirectories, which are created on first startup (just like with the client). To backup or restore, you have two options:

- Shut down the server, copy the database files and directories, then restart it. This is the only way, currently, to restore a db.
- In the client, hit *admin->your server->make a backup*. This will lock the db server-side while it makes a copy of everything server-related to `server_install_dir/db/server_backup`. When the operation is complete, you can ftp/batch-copy/whatever the `server_backup` folder wherever you like.

OMG EVERYTHING WENT WRONG

If you get to a point where you can no longer boot the repository, try running SQLite Studio and opening server.db. If the issue is simple--like manually changing the port number--you may be in luck. Send me an email if it is tricky.

Remember that everything is breaking all the time. Make regular backups, and you'll minimise your problems.

Revision #1

Created 12 March 2021 16:49:28 by CuddleBear

Updated 12 March 2021 18:24:07 by CuddleBear