GAVO DaCHS installation and configuration

Author: Markus Demleitner, Chuanming Mao
Email: gavo@ari.uni-heidelberg.de
Date: 2024-02-14

Contents

These installation instructions cover the installation of the complete data center suite. Installing libraries or, say, the tapsh, is much less involved. See the respective pages at the GAVO DC's software distribution pages for details on those.

Debian Systems (and Derivatives)

The preferred way to run DaCHS is on Debian stable or compatible systems. Starting with Debian bullseye (11.0), DaCHS is part of Debian. On a suitably recent system, just execute:

```
sudo apt install gavodachs2-server
```

With that, you are ready to proceed to the tutorial.
On Debian bookworm, the Javascript part of the browser interface is broken. See 500 FileNotFoundError in our collection of common problems.
This is the recommended way to get up to speed with DaCHS regardless of whether you want to run the bleeding-edge version. You can switch later.
On older systems, you need to add our APT repository to your `/etc/apt/sources.list` before running the `apt install`.
You are probably saving yourself quite a bit of grief if you just choose Debian stable as your platform when you are going for Debian derivatives. *If* you insist on running, say, Ubuntu, please note that they do not package DaCHS (so you have to add our APT repo no matter what) and that they do not package the q3c and pgsphere packages that DaCHS cannot live without.
A workable fix is to install the packages from postgres' APT repository (these are basically equivalent to what Debian has). However, these have multiple postgres versions at the same time, and hence you will need to manually make sure that the postgres server and the extensions match.
Hence, on dpkg-based distributions more remote from Debian, try a sequence more or less like:

(1) Purge all postgres packages that may already be on your system

(2) Add the postgres APT repository

(3) Add our APT repository

(4) `sudo apt update`

(5) `export PGVER=13; sudo apt install postgresql-$PGVER postgresql-$PGVER-q3c postgresql-$PGVER-pgsphere`

(6) `apt install gavodachs2-server`

If this fails, make sure the postgres instance listening on your port 5432 really *is* the one you got from the postgres repo, *not* the one from Ubuntu.

RPM-based Distributions

We used to have a worked-out recipe for RPM boxes here, but that fell into disrepair. If you are running DaCHS on RPM-based boxes, please contribute a howto here. Perhaps some the old material (git commit dd73cfdd0e98879e91704d1806ebc15ed1ac118c still has it) will come in handy.

MacOS

You can install DaCHS natively on MacOS. We don't regularly try, so the following may need some creativity. If you find some of these instructions outdated or know better ways to do this, please let us know.

1. Perhaps install Python 3. DaCHS should work with the built-in python, but you may want to keep all the extra packages out of the system python, in which case you could install, for instance, anaconda.

2. install Homebrew. The upstream-recommended way to do that actually is:

   ```
   $ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh
   ```

   (whether or not I like it)

3. Install postgresql with homebrew:

   ```
   $ brew install postgresql
   $ brew services start postgresql
   ```

   (this may need additional dependencies).

4. Get the DaCHS source code:

   ```
    $ git clone https://gitlab-p4n.aip.de/gavo/dachs/
    $ cd dachs
    $ env DACHS_PIP_MESS=1 pip install -e .

   We have reports that an extra ``pip install psycopg2`` has
   occasionally been necessary.  If this applies to you, we would be
   interested in the details.
   ```

5. Create groups and users:

   ```
   $ sudo dscl . -create /Groups/gavo
   $ sudo dscl . -create /Groups/gavo PrimaryGroupID 1000
   $ sudo dscl . -create /Users/gavo
   $ sudo dscl . -create /Users/gavo UniqueID 1000
   $ sudo dscl . -create /Users/gavo PrimaryGroupID 1000
   $ sudo dscl . -create /Users/gavo UserShell /bin/zsh
   $ sudo dscl . -create /Users/gavo RealName "GAVO System User"
   $ sudo dscl . -create /Groups/gavo Password "*"
   $ sudo dseditgroup -o edit -a `id -nu` -t user gavo
   ```

   Here, you need to restart the terminal to update the group information. Then, run `newgrp gavo`.

6. initialize new Postgresql cluster in the specified directory:

   ```
   $ initdb /usr/local/var/postgres
   ```

7. Prepare a target directory for DaCHS:

   ```
   $ sudo chown `id -nu`:gavo ~/dachsroot
   ```

Let DaCHS know you intend to use it by editing `/etc/gavo.rc` and adding:

```
[general]
rootDir: PATH/TO/YOUR/HOMEDIR/dachsroot
```

in there.

8. Create database gavo for DaCHS' use:

```
createdb -T template0 --encoding=UTF-8 --locale=C gavo
```

9. Tell DaCHS to make itself comfortable:

```
$ dachs init
```

Installation without Package Management

Dependencies

You will need a running postgres with your platform's equivalents of postgresql-q3c and postgresql-pgsphere.
If you want to use boosters, you will additionally need:

```
build-essential libcfitsio3-dev
```

To install from our version control system (see below), you will also need git
To pull in DaCHS' python dependencies, you probably want to try pip; note, however, that we will only declare them if you define the `DACHS_PIP_MESS` environment variable.

PgSphere

PgSphere is a postgres extension for spherical geometry. It is needed for support of the geometric types in DaCHS' ADQL implementation and in the preferred SIAP backend, so you should definitely install it. So build it, install the server development packages for postgres (such as postgresql-server-dev-9.x or postgresql-devel), check out https://github.com/pgsphere/pgsphere, and in the source directory run:

```
USE_PGXS=1 make
sudo USE_PGXS=1 make install
```

Q3C

DaCHS uses the Q3C library by Sergey Koposov and Oleg Bartunov, http://www.sai.msu.su/~megera/oddmuse/index.cgi/SkyPixelization for positional indexes. DaCHS uses it for positional indexes (the scs#q3cindex mixin) and in the interpretation of ADQL. It is therefore highly recommended to install it.
To do that, get the source directly from https://github.com/segasai/q3c/releases/, install the server development packages for postgres (such as postgresql-server-dev-9.x or postgresql-devel), and in the source directory run:

```
make
sudo make install
```

Installing DaCHS

Getting the source

If you cannot use the Debian package (or do not want to), you can grab a gavodachs package from our distribution page. Choose whatever *gavodachs-latest.tar.gz* points to. If you want to follow the bleeding edge closely – DaCHS is being actively developed – check out whatever is in the git repository right now. For a read-only copy, say:

```
git clone https://gitlab-p4n.aip.de/gavo/dachs.git dachs
```

After that, the current source code is in the `dachs` subdirectory. This is development code, so *please* do not hesitate to contact us if something weird is going on with it. We mean it; even trivial reports help us to gauge where our software behaves contrary to expectations. Plus, we don't have oodles of users, so chances are you won't get on our nerves. For contact options see http://docs.g-vo.org/DaCHS/#support.

Installing from source

First, if you already have all the dependencies (e.g., from the Debian Package) and just want to run a bleeding-edge version of DaCHS, the recommended way to install it is.

```
sudo -H pip install --no-deps --break-system-packages --no-build-isolation -e
.
```

(in DaCHS < 2.8.1, you should instead simply say `sudo setup.py develop`). This looks ghastly and looks dangerous but is actually what you need to do if you want to install a single package system-wide without stepping on the distribution's packages and downloading code from pypi.

Otherwise, create a python virtual environment in whatever way you do that kind of thing (`virtualenv --system-site-packages -p python3 ~/.vpython` being what I consider the industry standard), activate it and then say:

```
env DACHS_PIP_MESS=1 pip install -e .
```

(no sudo in this case). `DACHS_PIP_MESS` makes us be honest about our dependencies and ought make pip pull them all in. This is not regularly tested; if anything fails for you, please report the failure immediately.

Both techniques install in "editable mode", that is, if you edit anything in your checkout, it will be reflected in what the system executes. There is nothing wrong with leaving out the `-e` flag, except that probably defeats the purpose of installing from git in the first place.

Note: If running from git, do not forget to run `dachs upgrade` *after* a `git pull`. The on-disk structures of DaCHS sometimes change, and `dachs upgrade` makes sure they are properly updated if necessary. Technically, you would only need to run gavo upgrade if, in gavo --version, the two numbers behind "Schema" are different, but since `dachs upgrade` is smart enough to figure out when there's no need to do anything, just make it a routine to run it.

Note: `python setup.py install` and friends do not install DaCHS' man page. Either do that manually (it's in docs/dachs.1) or use the online version at dachs.1.html

Setup

All this is taken care of by the Debian package, so don't do any of this if you installed from .deb.

Introduction

GAVO DaCHS is quite sensitive to a correct setup as regards permissions. Experience has shown that user setup is the number one reason for installation problems. So, up front, here's what the steps given below should create:

- A group that will own certain directories that must be writable by the server (by default gavo).
- A user that the server will run as (by default gavo).
- A unix account for you that should not be root (in particular not if you're using setup.py develop on an git checkout). This should be in the gavo group (for when you're running `dachs serve debug`) and will usually own resource directories and the like.

On the database side, the following must be ascertained:

- There's a postgres database cluster in the C locale, with a database already created (named, by default gavo).
- "you" (i.e., your unix id) have admin privileges on this (at least for installation) using ident authentication
- for connections from the local host, the three roles the server use can access the database using md5 authentication.

Account Management

You should first create a user that the DaCHS server runs as later, and a group for running DC-related processes in:

```
sudo adduser --system gavo
sudo addgroup --system gavo
sudo adduser gavo gavo
```

(or similar, depending on your environment). This user should not be able to log in, but it should have a home directory. Everyone that may issue a `dachs serve debug` must be in the group created (this is because the log directory will be writable by this group); in particular, you should add yourself:

```
sudo adduser ‘id -nu‘ gavo
```

You may want to create another account for "maintenance", or just use your normal account; if more than one person will feed the data center, you'll need more elaborate schemes.
To update the system's idea of your group membership, say `newgrp gavo` or log in and out now.
All users that are to ingest data into the database using DaCHS must be part of this gavo group.

Database setup

The most complicated step in setting up DaCHS is actually setting up the database. We currently only support postgres.
While it is conceivable to use DaCHS together with an existing postgres database, we do not recommend trying this the first time. Experiment with a database dedicated to DaCHS first, then consider whether it's worth interfacing to your existing database or whether a copy of that data is more convenient.

Cluster Creation

You first need a database to play with, preferably in a suitable cluster (you could skip this, but the all bets are off as to whether you'll be able to store non-ASCII characters in strings).
It is recommended to create a dedicated cluster first even if you want to connect DaCHS to a pre-existing database later to get a feeling for how it works. See Connecting to a remote database for information on what setup is necessary in this case.
Database cluster generator is very system-dependent, and ideally a database admin would assist you.
On Debian systems dedicated for GAVO DaCHS, you can try the following (*Warning*: This will destroy any previous content anyone put in postgres databases on that particular system):

(1) Find out the version of the server you will be running (e.g., using `dpkg -l`; in Debian, more than one version may be installed in parallel. It's probably a good idea to use the most recent one. Set your desired version for subsequent use:

```
export PGVERSION=11
```

(2) Drop the Debian default cluster (this will delete everything in there -- for a fresh install, that doesn't matter, but don't do this if other people use the database). If you don't do this, your database will listen do a different port, and you will have to adapt the default profiles:

```
sudo pg_dropcluster --stop $PGVERSION main
```

(3) Create the new cluster used by DaCHS:

```
sudo pg_createcluster -d /<path-to-where-your-db-should-reside> \
  --locale=C -e UNICODE\
  --lc-collate=C --lc-ctype=C $PGVERSION main
```

The locale should currently be C, because only the C locale will allow you databases with all kinds of encodings. The database stores descriptions and similar entities, and you may encounter funny characters in there. It would be a shame if you couldn't store them (plus, you would get odd error messages for those).

If unsure where to put the cluster: Debian's default is `/var/lib/postgresql/<postgres-version>/main`.

(4) Start the server:

```
sudo /etc/init.d/postgresql start
```

(5) Create the database itself:

```
sudo -u postgres createdb -Ttemplate0 --encoding=UTF-8 --locale=C gavo
```

On Debian, the configuration files for this cluster are at `/etc/postgresql/$PGVERSION/pgdata/`.

Initial Account Setup

At least during setup, you also need superuser privileges on the database. For `dachs init` below to work, your normal account must have such privileges. On Debian systems, you can simply say:

```
sudo -u postgres createuser -s `id -nu`
```

You can drop those privileges later if they make you nervous, but for gavo init you need to be DB superuser. Also note that DaCHS assumes your server is trusted, and if people have managed to take over an account in the gavo group, they can do with your database whatever they please anyway. In particular (don't complain we didn't tell you), DaCHS currently encrypts *no* passwords; for the DB passwords, sensible encryption would mean the software requires some passphrase during startup, which we don't want. For user passwords (for protecting web resources), it would make no sense since with HTTP basic authentication as employed by DaCHS, they travel through the net unencrypted anyway (which is sometimes called "mild security").

Connecting to a Remote Database

See opguide.html#two-server-operation.

Configuration File

Next, you need to decide on a "root" directory for DaCHS. Below it, there are data descriptions, cache files, logs, etc. (these locations can be changed later, but for a simple setup we recommend keeping everything together). By default, this is `/var/gavo`. DaCHS is configured in an INI-style configuration file in `/etc/gavo.rc` (overridable using the envirvonment variable `GAVOSETTINGS`). In addition, users, in particular the gavo user, can have ~/.gavorc files, the contents of which override settings in /etc/gavo.rc.
Configuration Settings gives a walkthrough through the most important settings; for now, you must set the DaCHS root dir if you are not happy with `/var/gavo`:

```
[general]
rootDir: /data/gavo
```

as /etc/gavo.rc.
Whatever `rootDir` is, it must exist and be writable by you, or you must have sufficient privileges to create it. Do *not* run `dachs init` as root, since the files and directories it creates will be owned by whoever ran the program. In the typical situation in which you may not write to `rootDir`'s parent, do something like:

```
sudo mkdir -p /data/gavo
sudo chown `id -nu`:gavo /data/gavo
```

You can now let DaCHS create its file system hierarchy:

```
dachs init
```

`dachs init` will spit out a warning about a missing file `defaultmeta.txt` on the first run. On that first run, you can ignore the warning; the missing file will be created by DaCHS.
If your database server is not on the same machine as your web server (which is not recommended for a test setup), you have to pass a complete DSN that lets DaCHS connect as a superuser to `dachs init`. A DSN ("Data Source Name") is a sequence of key-values pairs as used by ODBC or Postgres itself (with keys discussed in Database Connection Control Functions in the postgres documentation). You would say something like:

```
dachs init --dsn "host=myhost.xy port=5546 user=super password=secret dbname=wisdom"
```

– make sure you give at least dbname and whatever role DaCHS ends up using has superuser privileges during setup (that role is not used during normal DaCHS operation any more).
You can later run gavo init again. It will not clobber anything you did in the meantime (well, if it does, it's a bug and you should fiercely complain). In particular, this is the most convenient way to create directories if you changed locations in `gavo.rc`.

The init script

Though you can operate the server manually through `dachs serve (start|stop|reload)`, you will probably want to have your init system automatically start it. See https://docs.g-vo.org/DaCHS/tutorial.html#starting-and-stopping-the-server for details on integrating DaCHS with either sysvinit or systemd.

Docker

We do not really recommend using Docker containers on production machines.
We do, however, use Docker in our internal QA to check installability and upgradeability; in case you are curious what we do, see http://svn.ari.uni-heidelberg.de/svn/integration/dockerbased .
Still, if you want to build a Docker-able DaCHS take a look at the corresponding Git or, analogously, Docker repositories.