

VESPA Workshop 2021

"Dachs on Docker"

Carlos Brandt
Jacobs University Bremen

Roadmap

☐ Docker install check

- ☐ Docker-compose install check

● Simple run of dachs on docker

- Test empty run
- Test import/publish
- Test data access

● Share data with container

- Mount host' directory into container

❖ Simplify command-line with docker-compose

Check Docker

The simplest way to check if Docker is installed and running is by trying it:

- Simply running `docker`, for example, should output docker's help message.
- The same should happen with `docker-compose`.

If the system complains about

- "Unknown command": you don't have it installed
 - Install Docker
- "Daemon down": you have it installed, but Docker daemon is not running
 - Start docker using your system/applications interface;
If on Linux: use the command-line to "service start docker"

Simple run -- get image

First, let's guarantee we have the published `dachs:latest` (docker) image in our local machine:

```
'''
```

```
(host)$ docker pull gavodachs/dachs:latest
```

```
'''
```

- Use of `latest` in `dachs:latest` is optional, as `latest` is used if not specified.

Simple run -- start (1)

Run the container:

...

```
(host)$ docker run -it --name vespa21 -p 8080:8080 gavodachs/dachs
```

...

- `-it` will make it Interactively through a Terminal`
- `--name` give it a name for easier reference`
- `-p "host":"container"` bind hosts "host" port to container's "container"`

Simple run -- start (2)

Start services -- postgres and dachs -- in the container:

...

```
(dckr)$ service postgresql start
```

```
(dckr)$ dachs serve debug
```

...

- Go to <http://localhost:8080> (in your browser) to see Dachs frontpage

Simple run -- imp/pub (1)

Open another terminal to connect to the same container:

...

```
(host)$ docker exec -it vespa21 bash
```

...

This will land you on a Bash/terminal inside the container.

Simple run -- imp/pub (2)

Download DaCHS (*arihip*) example data/resource*:

...

```
(dckr)$ cd /var/gavo/inputs
```

```
(dckr)$ mkdir -p arihip/data
```

```
(dckr)$ cd arihip && curl -O http://svn.ari.uni-heidelberg.de/svn/gavo/hdinputs/arihip/q.rd
```

```
(dckr)$ cd data && curl -O http://dc.g-vo.org/arihip/q/cone/static/data.txt.gz
```

...

*: <https://dachs-doc.readthedocs.io/tutorial.html#quick-start-with-dachs>

Simple run -- imp/pub (3)

Import a sample (for quick test):

...

```
(dckr)$ cd /var/gavo/inputs
```

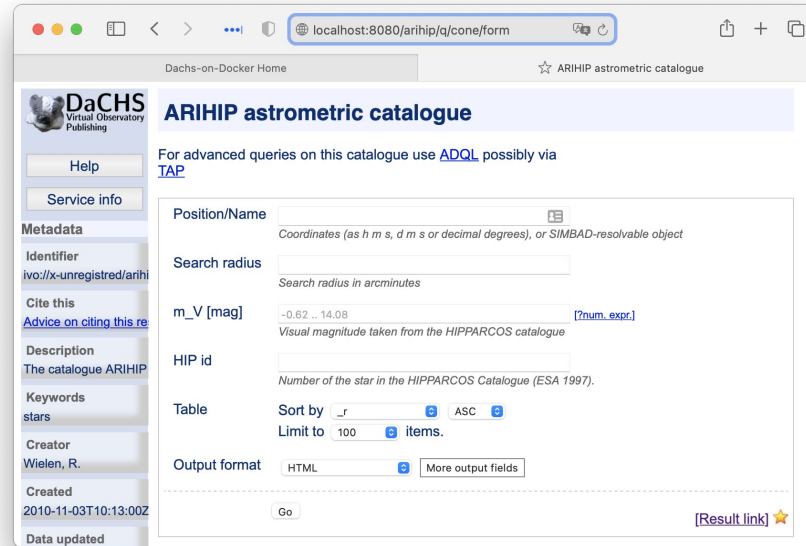
```
(dckr)$ dachs imp -M 1000 arihip/q.rd
```

...

➤ If you want to import all the data, remove "-M 1000" from your command.

Simple run -- data access (1)

Now we go to <http://localhost:8080/arihip/q/cone/form> to see the `arihip` service web interface:



The screenshot shows a web browser window with the URL `localhost:8080/arihip/q/cone/form`. The page title is "ARIHIP astrometric catalogue". On the left, there is a sidebar with the DaCHS logo and navigation links: "Help", "Service info", "Metadata", "Identifier", "Cite this", "Description", "Keywords", "Creator", "Created", and "Data updated". The main content area contains a search form with the following fields and options:

- Position/Name**: Input field with a help icon. Description: "Coordinates (as h m s, d m s or decimal degrees), or SIMBAD-resolvable object".
- Search radius**: Input field. Description: "Search radius in arcminutes".
- m_V [mag]**: Input field with a range of "-0.62 .. 14.08" and a "[?num_expr.]" link. Description: "Visual magnitude taken from the HIPPARCOS catalogue".
- HIP id**: Input field. Description: "Number of the star in the HIPPARCOS Catalogue (ESA 1997)".
- Table**: "Sort by" dropdown set to "_r" with "ASC" and "DESC" options. "Limit to" dropdown set to "100" with "items".
- Output format**: Dropdown set to "HTML" with a "More output fields" button.

At the bottom of the form is a "Go" button and a "[Result link] ★" link.

Simple run -- data access (2)

We can now go one step further and test the access through some VO-compliant tool; Topcat* for instance:

- Open Topcat > VO > TAP
- Set URL to [`http://localhost:8080/tap`](http://localhost:8080/tap)
- (SQL) select data from `arihip.main` table:
 - `SELECT * FROM arihip.main`
 - `Run Query` button
- You should now see "arihip.main" result in Topcat

*: <http://www.star.bris.ac.uk/~mbt/topcat/>

The screenshot displays the TOPCAT interface. At the top, the 'Table List' shows '1: TAP_2_arihip.main'. The 'Current Table Properties' panel on the right indicates the table's label, location, name, row count (1,000), and column count (63). Below this, the 'Table Access Protocol (TAP) Query' window shows the query: 'SELECT * from arihip.main'. The 'Table Browser' window displays the results of the query in a table format with columns: hipno, srcSel, raj2000, dec2000, and pmra. The results are sorted by hipno, showing 25 rows of data. The total row count is 1,000, and all 1,000 rows are visible.

hipno	srcSel	raj2000	dec2000	pmra
1	11 GCH	0.03734	46.94	3.11389E
2	15 T2H	0.05036	50.79119	3.84444E
3	30 T2H	0.09611	42.14147	-2.30833E
4	32 T2H	0.09858	51.93949	1.66667E
5	41 T2H	0.11657	54.30224	6.80556E
6	43 F63	0.1287	59.55968	-2.24556E
7	58 GCH	0.17352	62.1759	-1.30111E
8	59 T2H	0.17445	55.72246	-7.77778E
9	61 T2H	0.1763	53.92215	9.32778E
10	63 GCH	0.18181	45.25333	4.81667E
11	73 GCH	0.22033	66.84801	-5.74444E
12	83 T2H	0.24807	52.08056	-6.44444E
13	86 T2H	0.26046	69.6025	3.20556E
14	89 T2H	0.27715	53.16694	1.69444E
15	105 T2H	0.32781	70.92894	-1.35000E
16	106 F63	0.33021	49.98156	3.67222E
17	113 T2H	0.36562	49.01041	-2.47222E
18	119 GCH	0.39017	44.6754	-2.89722E
19	123 T2H	0.39989	72.23662	-1.24020E
20	124 GCH	0.40423	61.2228	-7.58333E
21	128 F63	0.41442	73.61185	2.12778E
22	131 T2H	0.41987	62.01809	-3.95833E
23	137 GCH	0.43271	42.36714	1.35278E
24	139 T2H	0.44527	67.50698	-8.94444E
25	142 GCH	0.45378	66.30603	1.39444E

Share (host/container) data

As it is usually the case, we have data/resources in our host system we want to import/publish on Dachs inside the container. We can share directories with containers by bind/mounting so-called *volumes when we run the container*:

...

```
(host)$ docker run -it --name vespa21 -p 8080:8080 \  
    -v /path/to/data/resourceX:/var/gavo/inputs/resourceX \  
    gavodachs/dachs
```

...

- The whole serve/import workflow stays the same, but you're using the data from "resourceX" you have in your host system. If you modify anything *inside* "resourceX" it will be seen on both sides (host/container)!

Simplify container start/stop (1)

Docker containers start/stop can be simplified through the use of docker-compose.

For that, we need a `docker-compose.yml` file. You can find an example at:

- github.com/gavodachs/dachs-docker/dockerfiles/docker-compose.yml

```
version: '3'
services:
  dachs:
    container_name: vespa2021
    image: gavodachs/dachs:latest
    tty: true
    network_mode: 'bridge'
    ports:
      - 8080:8080
    volumes:
      - "/path/to/my/dachs/data/resources:/var/gavo/inputs"
```

Simplify container start/stop (2)

We can then start up our container:

...

```
(host)$ docker-compose up
```

...

Connect to the container:

...

```
(host)$ docker exec -it vespa2021 bash
```

...

Do all the same.