

New Pricing and Packaging announced! [Learn more...](#)



Search for great c

Explore

Repositories

Organizations

Get
Help

Explore

bioconductor/experimenthub_docker

Using 0 of 1 private repositories. [Get more](#)



bioconductor/experiment...



Pulls 474

By [bioconductor](#) • Updated 5 months ago

Experiment Hub

Container

Overview

Tags

Dockerfile

Builds

Docker Container for ExperimentHub Server

Table of Contents

- Overview
- Setting up the docker container
 - Prerequisites
 - Run the container
 - Determine URL of server

- [Using the container](#)
- [Testing the changes](#)
- [Pushing changes to production](#)

Overview

This Docker Container provides an isolated environment for modifying the metadata records in the ExperimentHub database. It can be used for testing only or the modified database can be dumped and used to replace the production db.

The docker can be used by anyone who has the password for the 'read only' user on the production server. With these credentials, a copy of the mysql production database is downloaded locally to the docker container. From within an R session, global options are set to point code from ExperimentHubData and ExperimentHub to the local db. Changes to the metadata (inserts, modifications or extractions) are made to the docker db instead of production.

Setting up the docker container

Prerequisites

Set the environment variable `MYSQL_REMOTE_PASSWORD` to the password found in the "ExperimentHub production server" section of the Google Doc "Credentials For Bioconductor Cloud resources".

The container will not work properly unless this is set.

You can set it by doing:

```
export MYSQL_REMOTE_PASSWORD=XXX
```

where XXX is replaced with the correct password. This password should also be added to 'MYSQL_REMOTE_PASSWORD:' in the docker-compose.yml file.

You may also need to export the database type for mysql or sqlite:

```
export HUBSERVER_DATABASE_TYPE=mysql
```

If you are on Linux, install docker using [Docker Engine](#). Also install [Docker Compose](#). If you are on Mac or Windows, install Docker Desktop instead. You will not need to install Docker Compose because it is included in Docker Desktop. [Docker Desktop for Mac](#), [Docker Desktop for Windows](#).

Clone the HubServer code to the same directory as this README:

```
git clone https://github.com/Bioconductor/HubServer.git
```

Or if you already have this repository checked out elsewhere on your system, make a symbolic link to it in the current directory (the same directory as this README).

Create a directory called "data" in the same directory as this README.

Run The Container

Open a terminal window (a Docker Quickstart Terminal if you are on Mac or windows) and change to the same directory where this README is. Issue the command:

```
docker-compose up
```

This command will *pull down the database contents from the production ExperimentHub server* and then start a local server.

When you see a line like this:

```
experimenthub_1 | == Shotgun/WEBrick on http://0.0.0.0:3000/
```

...the ExperimentHub server is running in the container.

To verify that it is running, you can determine its URL (in the next section).

Determining URL of server

If you are on linux, the URL of the server is likely <http://localhost:4000/resource>.

If you are in the cloud you need to use the public DNS name of your instance as your IP address.

If you are using boot2docker (deprecated by Docker Toolbox), you can determine your Docker host's IP address with the command `boot2docker ip`. If this returns `1.2.3.4`, your URL would be `http://1.2.3.4:4000/resource`.

If you are using Docker Toolbox, the command to determine your Docker host's IP address is

```
docker-machine ip default
```

If this returns `1.2.3.4`, your URL would be `http://1.2.3.4:4000/resource`.

Using the container

Start a new R session in a new terminal window. Assuming your server URL is `http://1.2.3.4:4000/resource`, enter the following at the R prompt:

```
## used by ExperimentHubData to insert metadata
options(EXPERIMENT_HUB_SERVER_POST_URL="http://1.2.3.4:4000/resource")
## used by ExperimentHub to get metadata
options(EXPERIMENT_HUB_URL="http://1.2.3.4:4000/resource")
```

Replace the URL with your actual URL, of course. These options must be set before loading `ExperimentHub` and `ExperimentHubData`.

```
library(ExperimentHub)
library(ExperimentHubData)
```

Now you can run recipes, etc. and the insertions will happen inside the docker container, not in the production database.

Testing the changes

You can interact with the docker db with R or mysql.

R session

To view the docker db from R you must first convert the mysql db to sqlite. From another terminal window do

```
docker exec -ti experimenthubdocker_experimenthub_1 bash
```

That will log you in to the experimenthub server container. Then do the following:

```
cd /HubServer/  
ruby convert_db.rb
```

That will convert the mysql database to sqlite. You can then type

```
exit
```

to exit the container, start R and set the variables as described in [Using the container](#). ExperimentHub should recognize that there are changes and download the new sqlite database, but if not you can remove the old one from your cache to trigger a copy.

mysql session

mysql is not exposed to the host, so you need to do this from the mysql machine. Run

```
docker ps
```

to see a list of containers that are running. One will have the string 'db' in it. Let's say the full name is 'db1'. Connect to the container with:

```
docker exec -ti db_1 bash
```

From within the resulting prompt start mysql and query the db as usual:

```
mysql -p -u hubuser
```

When you are satisfied that the changes you have made are correct, you can update the production database (see next section). If you have messed up and you don't want to push your changes to production, you can just exit the container (press Control-C in the window where it is running) and start over again.

Pushing changes to production

You need to back up the database inside the docker container. You can do it like this:

```
docker exec experimenthubdocker_experimenthub_1 bash /bin/backup_db.sh
```

Note that `experimenthub_experimenthub_1` is the name of the docker container that has the experiment hub server on it; this name may vary, the `docker ps` command will give you the accurate container name.

Now in the `data` directory on your local machine, there is a file called `experimenthub.sql.gz`. Upload this to the production machine.

Log into the production machine and make a backup of production db:

```
mysqldump -p -u hubuser experimenthub | gzip > dbdump_YYYY_MM_DD_fromProd.sql.gz
```

Drop the old db and create an empty one:


```
mysql -p -u root
drop database experimenthub;
create database experimenthub;
quit;
```

Fill the empty with the modified db:


```
zcat dbdump_YYYY_MM_DD_fromDocker.sql.gz | mysql -p -u root experimenthub
```

`docker pull bioconductor/experimenthub_docker`

Owner

 bioconductor

Source Repository

 Github
[Bioconductor/ExperimentHub_docker](#)



Explore

- Docker Editions
- Containers
- Plugins
- Pricing

Account

- My Content
- Billing

Publish

- Publisher Center

Resources

- Docker Blog

 [Cookie Preferences](#)