**Running VDJbase and OGRDB under Docker**

**William Lees (william@lees.org.uk)**

**November 2021**

## 1. Introduction

**This document describes a configuration of VDJbase and OGRDB using shared infrastructure, running on a single Vserver as a set of containers under Docker Compose. It covers both installation and maintenance. All software, and the data required for most installations, is available from GitHub under open-source license.**

**The installation is defined in three github repos, which are cloned to the server as part of the installation process:**

|  |  |  |
| --- | --- | --- |
| **repo** | **Purpose** | **Github location** |
| **digby\_infra** | **Shared infrastructure** | [**williamdlees/digby\_infra**](https://github.com/williamdlees/digby_infra) |
| **digby\_docler** | **VDJbase nack-end server, and static code for VDJbase browser client** | [**williamdlees/digby\_docker**](https://github.com/williamdlees/digby_docker) |
| **ogrdb\_docker** | **OGRDB server** | [**airr-community/ogrdb\_docker**](https://github.com/airr-community/ogrdb_docker) |

**The repos follow a common file structure:**

**./containers includes the docker-compose file, and subdirectories defining custom containers.  
./config includes configuration files, log files, backups, and other persistent data  
./backup contains backups (but note exceptions in the Maintenance section)**

**Between them they define the following containers:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Container** | **Description** | **Provides** | **Consumes** |
| **swag** | **nginx and Let’s Encrypt (**[**ghcr.io/linuxserver/swag**](ghcr.io/linuxserver/swag)**)** | **digby\_infra/config/nginx:/config** | **digby\_infra/config/wordpress/html digby\_docker/config/nginx/www** |
| **mariadb** | **mariadb ([linuxserver/mariadb](https://docs.linuxserver.io/images/docker-mariadb))** | **digby\_infra/config/mysql** |  |
| **wordpress** | **Wordpress (**[**wordpress:5-fpm-alpine**](https://hub.docker.com/_/wordpress)**)** | **digby\_infra/config/wordpress/html digby\_infra/config/wordpress/backup** |  |
| **vdjbase\_flask** | **VDJbase server and client** | **digby\_docker/config/flask digby\_docker/backup digby\_docker/study\_data** |  |
| **ogrdb\_flask** | **OGRDB server** | **ogrdb\_docker/config/flask ogrdb\_docker/backup ogrdb\_docker/ogre** | **digby\_infra/config** |

## 2. Installation

## 2. 1 Requirements

* Root access to a Unix host with at least 4GB of RAM and 80GB of disk
* Some means of securing backups away from the server. Historical backups are maintained by the containers, so you don’t need to maintain deep history. On the production server, we keep a minimum of four weekly snapshots off site.
* Up-to-date versions of Docker, Docker Compose and git.
* OGRDB requires an [NCBI API key](https://ncbiinsights.ncbi.nlm.nih.gov/2017/11/02/new-api-keys-for-the-e-utilities/) .
* Access to SMTP to send messages (see 2.3 below)
* DNS names for the various services, with and without www prefixes. As an example, the production server has DNS names for:
  + vdjbase.org, [www.vdjbase.org](http://www.vdjbase.org)
  + ogrdb.airr-community.org, [www.airr-community.org](http://www.airr-community.org)
  + wordpress.vdjbase.org, [www.wordpress.vdjbase.org](http://www.wordpress.vdjbase.org)

You will therefore need 6 DNS names for the full three services. All should point to the Unix host.

A suitable virtual server can be obtained from Digital Ocean for $24/month as follows:

* Select ‘Create Droplets’
* Click on ‘Marketplace’ under ‘Choose an image’ and select the Docker image
* Select ‘Basic’ under ‘Choose a plan’ and select $20/mo 4GB/80GB/2TB
* Enable backups for a further $4
* Press Create Droplet

2.2 The ‘home’ directory

Decide where you wish to locate the installation: this will be referred to as the ‘home’ directory. By default this is /root. We recommend using the default if at all possible, as this reduces the number of configuration changes needed, and makes it easier to update the installation using ‘git pull’. Likewise we recommend using a dedicated virtual server if at all possible. This will simplify configuration and minimise the possibility of clashes with other applications.

2.3 Setting up a mail account

Both OGRDB and VDJbase use SMTP to send mail. OGRDB uses mail as part of user account setup and to notify committee members of incoming work. VDJbase provides notification of errors, for example in the running of reports.

We strongly recommend against the use of a mail account that is used for other purposes, because the credentials for the account will be stored on the server you are setting up. Unless your institution can provide details of an SMTP server that you can use, a good option is to create a dedicated gmail account. See the section on Gmail on [this page](https://mailtrap.io/blog/flask-email-sending/) for details of configuration. You should enable two-factor authentication and use an application password: the ‘non-secure app’ option discussed in the article needs manual resetting every 2-3 months.

When configuring mail in secret.cfg, note that setting TESTING to True at the top of the file will stop mail being sent. This is useful in certain circumstances, for example during initial configuration. The setting has no other effect.

2.4 Installation of the shared infrastructure

Open tcp ports 80 and 443, and ensure unnecessary ports are closed. All you need are 80 and 443 for http/https, and port 22 for SSH so that you can log in to the server. The applications use https exclusively. Let’s Encrypt uses http for ownership validation. All other ports should be firewalled against incoming connections, unless there’s a specific reason for enabling them.

For the Digital Ocean example above:

ufw allow 80/tcp  
ufw allow 443/tcp  
remove rules for other ports with ufw status numbered, ufw remove

Install an ‘entropy’ package. If this is not done, Docker commands can be very slow, and get slower over time:

apt-get install haveged  
update-rc.d haveged defaults

**A general note on Docker messages**

Because the container definitions are split into multiple .yml files – one for the infra, and one each for VDJbase and OGRDB, you can expect to see messages warning you about orphan containers and active endpoints on the network when you take containers down with docker-compose. These can be ignored.

Clone the repo:

git clone https://github.com/williamdlees/digby\_infra

Modify configuration files and bring up the infra:

* + Modify the urls in digby\_infra/config/nginx/ngnix/site-confs to match the DNS names you have established. If there is a service (ogrdb, vdjbase, wordpress) that you do not intend to provide, remove the corresponding file in this directory.
  + Likewise modify the urls in digby\_infra/containers/docker-compose.yml to match. Refer to [**ghcr.io/linuxserver/swag**](file:///D:\Research\digby_infra\ghcr.io\linuxserver\swag)for more information if necessary. Note that the configuration is set up to use DNS validation for Let’s Encrypt.
  + Fix links in digby\_infra/containers/docker-compose.yml if you need to change the home directory from /root.
  + For the time being, set STAGING to true in docker-compose.yml so that the Let’s Encrypt configuration can be tested without the risk of time-outs if it needs to be corrected.
  + Bring the infra up interactively:  
    cd digby\_infra/containers  
    docker-compose up
  + Review the output from this command and in particular check that the certificates were properly issued. Fix problems if necessary
  + Once you are happy that certificates are properly issued, change STAGING to false in docker-compose.yml and bring the infra up as a service:  
    cd digby\_infra/containers  
    docker-compose up -d
  + If you wish you can check the logs during startup with docker-compose logs -f.

To log in to a container:

* + Use docker ps to find its id
  + docker exec -it <id> bash

Set up databases and passwords:

* + For this section, log in to the mariadb container, as above.
  + Log in to mysql using the password predefined in docker-compose.yml (you are about to change this password: this is the last time it will be used):   
    mysql -u root -pgsdfgtwevdfg
  + Create 4 ‘hard’ passwords and make a note of them somewhere safe. These will be used for mysql root, and for the three databases.
  + Create databases and users as follows, substituting your passwords:

ALTER USER 'root' IDENTIFIED BY '<password>';

CREATE USER 'ogrdb' IDENTIFIED BY '<password>';

CREATE USER 'digby' IDENTIFIED BY '<password>';

CREATE USER 'wordpress' IDENTIFIED BY '<password>';

CREATE DATABASE ogrdb;

GRANT ALL PRIVILEGES ON ogrdb.\* TO 'ogrdb';

CREATE DATABASE digby;

GRANT ALL PRIVILEGES ON digby.\* TO 'digby';

CREATE DATABASE wordpress;

GRANT ALL PRIVILEGES ON wordpress.\* TO 'wordpress';

FLUSH PRIVILEGES;

* + Note that the response to each line is **always** ‘Query OK’. Check carefully to see if any errors are reported and fix if necessary.

2.5 Configuration of Wordpress

* + Log out of the container
  + Take the infra down:  
    cd digby\_infra/containers  
    docker-compose down
  + In docker-compose.yml, set MYSQL\_ROOT\_PASSWORD and WORDPRESS\_DB\_PASSWORD to the ones you chose above.
  + Restart the infra: docker-compose up -d
  + Browse to the Wordpress URL. You will be prompted to select the language and to configure a management account.

At this point, Wordpress can be configured with any plugins and styles that you wish to use. Here we’ll describe the minimum configuration necessary for operation with OGRDB and VDJbase:

* + Install a backup service. We recommend the free version of [UpdraftPlus](https://updraftplus.com/). Backups should be directed to the local folder /backup, from where they will be picked up by the overall infra backup system (described later in this document): you therefore do not need to save many backups in /backup, perhaps just 2-3 days.
  + From the Posts/Categories menu, create three categories: VDJbase News, VDJbase Help, OGRDB News. The slug names must be set to vdjbase\_news, vdjbase\_help, ogrdb\_news.
  + Write a brief post in each category for test purposes.

2.6 Installation of VDJbase

Although it’s envisaged that any installation will run an instance of VDJbase, you can in theory skip this section if you don’t intend to provide an instance of VDJbase, but please make sure you removed the nginx vdjbase site configuration file in section 2.5. This is important because it references the vdjbase\_flask host name. If this has not been registered on the internal Docker network, nginx will not service requests to any host. Also, please note that ALL DATABASE BACKUPS ARE RUN BY THE VDJBASE CONTAINER. It may, therefore, be a good idea to configure VDJbase even if you don’t wish to expose it: removing the nginx vdjbase site configuration file will ensure it is not accessible externally.

Temporarily remove the site definition file for ogrdb:

* + cd digby\_infra/config/nginx/nginx/site\_info  
    rm ogrdb

Clone the repo:

git clone <https://github.com/williamdlees/digby_docker>

Build the container:

* + In digby\_docker/containers/my\_flask/app, copy sample\_secret.cfg to secret.cfg.
  + Edit secret.cfg:
    1. Include the password for the digby user in the database URI. The line should read 'mysql+mysqldb://digby:password@mariadb:3306/digby'
    2. Enter details of the SMTP account that OGRDB should use to send mail. The settings are used by [flask\_mail](https://pythonhosted.org/Flask-Mail/): consult its documentation for further guidance.
    3. Enter your own random text into SECRET\_KEY and SECURITY\_PASSWORD\_SALT
    4. Modify BACKEND\_LINK, STATIC\_LINK and OUTPUT\_REPORT\_LINK so that they include the DNS name for your instance of VDJbase (without the www.). As an example, the value for STATIC\_LINK in the public server is ‘https://vdjbase.org/static/’.
    5. The default WORDPRESS settings point to the public site. If you are using the local Wordpress, you can change WORDPRESS\_NEWS\_URL to the URL for your Wordpress service so that you can publish your own news items. Usually you would keep WORDPRESS\_HELP\_URL pointing to the production server, but this can also be changed if you wish.
    6. If you wish to set up secured access, set JWT\_USER and JWT\_PASSWORD to the username and password you wish to use. Otherwise they should both be left as empty strings.
  + Edit the backup script digby\_docker/containers/my\_flask/app/do\_backup.sh:  
    in the line starting mysqldump replace the string after -p with the database root password.
  + Build the dependency container my\_R:  
    cd digby\_docker/containers/my\_flask/my\_R  
    docker build --no-cache -t my\_r . | tee docker.log | grep "DONE\|ERROR"  
    This will take up to an hour to build as many R packages are built from scratch. The log is extremely verbose. You will see a single line on the standard output for each package built. If there are failures:
    1. Check the tail of docker.log for the problem
    2. Usually the docker file is set to use the latest version of R and the latest version of packages. Problems may occur if you build shortly after a new version of R or a new version of your Unix is issued: these can generally be addressed by pinning the version of R or the OS to the previous version. Please ask for help if you need it. Package dependency in R is extremely fragile and it may be necessary to debug and fix dependency issues. If you decide to work through the issue yourself, alter the docker file so that the failing package is installed first, which will reduce the debug cycle time.
  + Build the parent container my\_flask:  
    cd digby\_docker/containers/my\_flask  
    docker build --no-cache -t my\_flask .  
    This build is much faster and can be repeated if the server code changes, without rebuilding my\_R unless updates to R are explicitly needed.
  + Fix paths in digby\_docker/containers/docker-compose.yml if you needed to change the home directory from /root.
  + Check with docker ps that the infra is running, start it if necessary. Then bring up the VDjbase container:  
    cd digby\_docker/containers  
    docker-compose up -d
  + Check the logs for errors:  
    docker-compose logs -f  
     Initialisation is complete when Celery is running, at which point you can Ctrl-C out of the logs.
  + You should now be able to reach your vdjbase site from the browser.
  + Browse to https://vdjbase.org/admin . You will be prompted to enter details for the administrator account.
  + Once the account is created, use it to log on, and rebuild the genomic data:  
    Browse to https://vdjbase.org/admin and log in  
    Browse to https://vdjbase.org/admin/update\_genomic. Wait a little while for it to complete and provide output.  
    Browse to <https://vdjbase.org/admin/build_gff>  
    You can ignore any note to run make\_bam as this is done automatically on Unix. The genomic data is now built.

The installation of VDJbase is now complete, and you should be able to explore both AIRR-Seq and genomic data.

Operational Checks - User Interface

- Check that the site is accessible through both http and https links, with http diverting to https.

- Check that you can navigate the site successfully

- Check that reports are working and that static files (e.g. ogrdbstats reports) are downloadable

- Check that the IGV genome browser works

2.7 Installation of OGRDB

You can skip this section if you don’t intend to provide an instance of OGRDB, but please make sure you removed the nginx site configuration file in section 2.5. This is important because it references the ogrdb\_flask host name. If this has not been registered on the internal Docker network, nginx will not service any requests.

Assuming that you previously installed VDJbase, you now need to reinstate the site definition file for ogrdb:

* + cd digby\_infra/config/nginx/nginx/site\_info  
    git checkout -– ogrdb

Clone the repo:

git clone <https://github.com/airr-community/ogrdb_docker>

Build the container:

* + In ogrdb\_docker/containers/my\_flask/app, copy sample\_secret.cfg to secret.cfg.
  + Edit secret.cfg:
    1. Include the password for the ogrdb user in the database URI. The line should read 'mysql+mysqldb://ogrdb:password@mariadb:3306/ogrdb'
    2. Enter details of the SMTP account that OGRDB should use to send mail. The settings are used by [flask\_mail](https://pythonhosted.org/Flask-Mail/): consult its documentation for further guidance.
    3. Enter your own random text into SECRET\_KEY and SECURITY\_PASSWORD\_SALT
    4. Enter the NCBI API key that ogrdb should use.
    5. The default WORDPRESS settings point to the public site. If you are using the local Wordpress, you can change WORDPRESS\_NEWS\_URL to the URL for your Wordpress service so that you can publish your own news items.
  + Build the container:   
    cd ogrdb\_docker/containers/my\_flask  
    docker build --no-cache -t my\_flask .

Prepare the data directories:

* + Create directories ogrdb\_docker/ogre/attachments and ogrdb\_docker/ogre/imgtfiles.
  + Fix paths in ogrdb\_docker/containers/docker-compose.yml if you need to change the home directory from /root.
  + Check using docker ps that the infra is running. Start it if necessary.
  + Start the container:   
    cd ogrdb\_docker/containers  
    docker-compose up -d
  + Check the logs for errors:   
    docker-compose logs -f  
    Initialisation is complete when gunicorn is running, at which point you can Ctrl-C out of the logs.

Create the Administrator account:

* + At this point you should be able to open the site in a browser.
  + You will be prompted to enter details for the administrator account.
  + you should then be able to log on using these details, and the Admin item should be in the top-level menu

The installation of OGRDB is now complete.

3 System Management

3.1 Operational checks

3.1.1 Infrastructure

- Check with docker ps that the swag, wordpress and mariadb containers are running

- Check that nginx logs at digby\_infra/config/nginx/log/nginx are present and updating

- Check that Wordpress backups are being created at digby\_infra/config/wordpress/backup according to the schedule that you defined

- Check that the database dump at digby\_docker/config/flask/log/sqldump is being created once every 24 hours, it is non-empty, and contains valid SQL to recreate the three databases. (this file is accumulated into the VDJbase backup cycle).

3.1.2 VDJbase checks

- Check with docker ps that the my\_flask container is running

- Check that the gunicorn access and error logs at digby\_docker/config/flask/log are present and are updating.

- Check that backups are being created at digby\_docker/backup. Unpack or list one of the daily backup archives and check that it contains sqldump, the study data and other app-related data

3.1.3 OGRDB Checks

- Check with docker ps that the ogrdb\_flask container is running

- Check that the gunicorn access and error logs at ogrdb\_docker/config/flask/log are present and are updating.

- Check that backups are being created at ogrdb\_docker/backup. Unpack or list one of the daily backup archives and check that it contains sqldump, the study data and other app-related data

3.2 Management of VDJbase

3.2.1 Data Management

AIRR-Seq and genomic data is located at digby\_docker/study\_data. When VDJbase is first built, it will be populated with the data currently held on the public server. Should you wish to remove this data, follow the instructions for data removal in the sections below.

3.2.1.1 Managing AIRR-seq data

AIRR-Seq data is located at digby\_docker/study\_data/VDJbase. There are two branches: db and samples. Each branch has folders for species, and sub-folders for that species’ datasets. The organisation of files within each dataset is described [here](https://github.com/williamdlees/digby_data/raw/master/airr-seq%20data%20format.docx).

The VDJbase processing pipeline provides three files:  
  
db.sqlite3 – a SQLite database  
db\_description.txt – a one-line description of the dataset  
samples.zip – the samples subtree

To add a dataset to VDJbase, create suitable directories under digby\_docker/study\_data/VDJbase/db and digby\_docker/study\_data/VDJbase/samples. Copy the first two files into the database folder and unzip the third into the samples folder. Stop and restart the VDJbase container with docker-compose. To remove a dataset, remove the associated folders, stop and restart.

3.2.1.2 Managing Genomic data

Genomic data is organised into folders per species and dataset, as for AIRR-seq data. The head is at digby\_docker/study\_data/Genomic. By default, some data for Rhesus macaque IGH is set up during installation. You can add additional datasets here, and/or remove the Rhesus macaque data.

Within each dataset folder, the contents are defined in a file called <dataset>.yml, for example macaque\_igh.yml. This contains metadata for each assembly or contig contained in the dataset. The associated data consists of a FASTA file containing the sequence of the assembly or contig, and a csv file containing annotations. The actions performed when genomic data is initialised or updated (detailed in the installation section and upgrade section below) rebuild the database with the genomic information, and prepare files for the IGV browser.

3.2.3 Reports

Reports are run by separate processes on the backend, under control of [Celery](http://celeryproject.org/). Reports are written in Python or R and it should not be necessary to change the front-end code to accommodate new reports. Please get in touch if you would like more details on how to create them.

3.2.4 Logs

The VDJbase backend is managed by gunicorn. Access and error logs will be found at digby\_docker/config/flask/log. Any issues with reports will be recorded in celery.log in the same directory. Assuming that you configured mail logging, you will also be sent details of any issues, and these should be reflected in the logs. Report intermediate files are retained as long as the container is live: the command used to run the report is logged in celery.log and you can log in to the container to retrieve the intermediate files or reproduce an error,

3.2.5 The Admin menu

There should be no need to log in to the backend ‘admin’ interface, except to rebuild genomic data. Likewise there should be little need for backend account maintenance. However, having logged on, the home page at /admin has an ‘Admin’ tab at the top: clicking this will allow editing of the user and roles tables. Other functions, such as managing AIRR-seq or genomic data, are best performed from the command line as outlined elsewhere in this document.

3.2.6 Upgrading the software

To upgrade the software, first update digby\_docker:

cd digby\_docker  
git pull

(if you have changed any files included in the repo, for example the Docker files, you may have to stash or commit your changes and re-merge after the pull. Be careful to reflect any updates in the files if you need to do this).

build the my\_flask container: this will automatically pull in the current code. The data provided on the public server is included in the build. To refresh your copy of this data (both AIRR-Seq and genomic), create a file called .gitkeep in the directory digby\_docker/study\_data. Then stop and start the containers:

touch study\_data/.gitkeep  
cd containers  
docker compose down  
docker-compose up -d

With .gitkeep in place, the current data displayed on the public server will be copied to study\_data. Without it, study\_data will be left alone. Neither option will interfere with other datasets you have added to study\_data (unless there are name conflicts).

If the genomic data has been changed, it should be rebuilt once the service is started, following the steps outlined in section 2.6 (log in to the backend as admin, browse to upgrade\_genomic, browse to build\_gff).

3.3 Management of OGRDB

3.3.1 The Admin menu

3.3.2 Configuring Users and Roles

3.3.3 The super-admin role

3.3.4 Logs

3.3.4 Starting the container without starting flask

3.3.5 Upgrading the software

To upgrade the software, build the my\_R and my\_flask containers, as detailed in the installation notes, and then stop and restart them with docker-compose. In most cases, this is all that’s required, but occasionally some data transformation is required. Notes will be provided when this is the case.

3.4 Management of the infrastructure

3.4.1 Logs

3.4.2 Managing space

3.4.2 Upgrading the software

1. Using the system

The system should now be available as follows:

http://<host>/ - front end  
http://<host>/admin - back-end maintenance login  
http://<host>/admin/api - api

The exports directory in the backend server is mapped to your host. Assuming the default configuration in docker-compose.yml, you will find it at /root/digby\_docker/config/flask/exports. You can take in new RepSeq pipeline data by putting it there and following the process outlined in *Running the VDJbase backend under PyCharm*, The study\_data directories are exposed at /root/digby\_docker/study\_data.

1. Updating the system

To update, stop the containers:

cd digby\_docker/containers

docker-compose down

Pull digby\_docker to refresh the Angular client image (this is stored in digby\_docker/config/nginx/www):

git pull

rebuild my\_flask as in step 4. This will pull the contents of the digby\_backend repo. My\_r only needs to be rebuilt first if the required R libraries have changed:

cd my\_flask

docker build --no-cache -t my\_flask .

restart the containers:

docker-compose up -d

1. Troubleshooting

The flask logs are at digby\_docker/config/flask/log  
The nginx logs are at digby\_docker/config/nginx/log/nginx  
The container logs can be viewed by executing docker-compose logs

You must be in the directory digby\_docker/config to issue docker-compose commands.

1. Managing datasets

Please refer to the document ‘Running the VDJbase backend under Pycharm’ in the digby\_backend repo.

Docker conservatively stores many built images and containers. If you build repeatedly you will eventually run out of space. To reclaim space, use the commands

docker image prune

and

docker container prune

Steps:

1. Set the swap:

dd if=/dev/zero of=/var/swap.1 bs=1M count=4024

mkswap /var/swap.1

chmod 0600 /var/swap.1

swapon /var/swap.1

add the following line to /etc/fstab:

/var/swap.1 swap swap noauto 0 0

Once the swap is configured, you can turn it on with the line

swapon /var/swap.1

And off with the line

swapoff -a

You should only turn it on when building containers, and make sure that it is off otherwise. If it is on during normal operation, the website will become unresponsive.

Maintenance tasks:

* Restore OGRDB/VDJbase/Wordpress database
* Res