Configure the Comparative Pathology Workbench (CPW) in a simulated Production environment with Docker

This document describes how to install and setup the CPW software system and run it within Docker, in a simulated PRODUCTION environment.

See the Document "CPW_Configure-Development.docx" for instructions on how to setup the CPW in a DEVELOPMENT environment.

Mike Wicks
19th May 2023

Prerequisites

However, there are a number of issues with running the software in a development mode.

1. The CPW won’t work straight "out of the box”.
   The installer is required to run a series of commands to configure the system first (eg. populating the database with “base" data).

2. The Installer is required to download, run and configure a WordPress “stack”, also within Docker too.

3. The Installer is required to turn off any other web servers that might be running on your system (eg. Mac’s have Apache running by default).

4. The resulting system will not display images from external EGCA OMERO servers (security issues, however, it will display images from the IDR).

Install Required Software

Install docker, from https://www.docker.com/

Part A - CPW Software Setup

Step 1 - Pull the Software from Github

Download the code from the following repository:
https://github.com/Comparative-Pathology/comparativepathologyworkbench

This repository provides the required Docker Configuration Files to run the CPW within a PYTHON Application Container, hosted by an Nginx webserver. The resulting configuration emulates a PRODUCTION environment for the system.

There are specific Environment files for use with the CPW running in a Docker container, however, these do not need to be touched!

- **app/config/env_prod.txt**
- **app/config/env_prod_db.txt**

There are 3 files that are provided in the CPW repository that configure the system to run within Docker:

- **docker-compose.yml**
  - specifies the required Docker Containers
- **app/Dockerfile**
  - specifies the Docker environment required by the CPW software.
- **nginx/Dockerfile**
  - specifies the Docker environment required by the Nginx Webserver.

There are 2 files that are provided in the CPW repository that configure the system for Python and Nginx:

- **app/requirements.txt**
  - this specifies all the required Python libraries for the system.
- **nginx/nginx.conf**
  - specifies the Nginx Webserver configuration file.

Step 2 - Build the Docker Containers

In the cloned repository folder, run the following to build the software environment:

```
docker-compose up -d --build
```

Step 3 - Configure the Database via the Django ORM

The following commands create the database tables, and creates a superuser account

docker-compose exec web python manage.py migrate
docker-compose exec web python manage.py createsuperuser

(Use the account name of "**admin**" here)

You can list the running containers using this command and it variations:
docker container ls --filter name=web

Step 4 - Populate the Database via the Django ORM:

The CPW System requires certain tables to be pre-populated with data – this is achieved with the following 8 commands:

docker-compose exec web python manage.py loaddata fixtures/protocol_prod.json --app matrices.protocol
docker-compose exec web python manage.py loaddata fixtures/type_prod.json --app matrices.type
docker-compose exec web python manage.py loaddata fixtures/location_prod.json --app matrices.location
docker-compose exec web python manage.py loaddata fixtures/command_prod.json --app matrices.command
docker-compose exec web python manage.py loaddata fixtures/blog_prod.json --app matrices.blog
docker-compose exec web python manage.py loaddata fixtures/environment_prod.json --app matrices.environment
docker-compose exec web python manage.py loaddata fixtures/authority_prod.json --app matrices.authority
docker-compose exec web python manage.py loaddata fixtures/collectionauthority_prod.json --app matrices.collectionauthority
docker-compose exec web python manage.py loaddata fixtures/server_prod.json --app matrices.server

Step 5 - Set up the Views in Postgres:

The CPW system requires 4 Views to be created within the Postgres Database.

Find the Postrges Container, thus:

docker container ls --filter name=db

(Usually named similar to the folder you're working in)

Copy across the SQL scripts that set up the Postgres Views into the Postgres container

docker container cp app/fixtures/docker_authorise_views.sql comparativepathologyworkbench-db-1:/
docker container cp app/fixtures/docker_create_views.sql comparativepathologyworkbench-db-1:/
docker container cp app/fixtures/docker_drop_views.sql comparativepathologyworkbench-db-1:/

Execute the commands that setup the Views and Authorise them:

docker container exec -it comparativepathologyworkbench-db-1 psql --dbname=workbench_czi_prod --username workbench_czi -f /docker_create_views.sql
docker container exec -it comparativepathologyworkbench-db-1 psql --dbname=workbench_czi_prod --username workbench_czi-f /docker_authorise_views.sql

If you need to Drop the views and start again, execute this command:

docker container exec -it comparativepathologyworkbench-db-1 psql --dbname=workbench_czi_prod --username workbench_czi -f /docker_drop_views.sql

Step 6 - Populate the Static Files on NGINX

Run the following commands, to ensure the Nginx webserver servers up the required static files to the CPW application

docker-compose exec web python manage.py collectstatic --no-input --clear

Step 7 - Test the CPW

The system is now ready to be accessed by the previously created superuser account.

Go to a Browser, and point it at http://localhost:1337/home/

You should see the CPW Home Page - Hoorah!

Step 8 – Stopping the CPW

To bring the system down, use the following command:

docker-compose down

Step 9 - Prepare for WordPress

The resulting CPW system cannot do anything at the moment, as it needs linking to a WordPress server.

- If you have full admin rights to a pre-existing WordPress Server, go to Part B, Step 6.

- If not, you will need to setup a WordPress instance running locally, as described in the next section Part B.

Part B - Setup WordPress

Step 1 - Download the WordPress Software Stack

In a different folder, download the software from Github:
https://github.com/Comparative-Pathology/docker-compose-wordpress

This repository provides a full WordPress "Stack" (ie. MySQL, PHP, WordPress and Nginx), managed by Docker.

Step 2 - Download the WordPress Software

Download WordPress from here:
https://en-gb.wordpress.org/download/

Copy the downloaded and expanded WordPress source code into the "docker-compose-wordpress" folder, "wordpress"

Delete the file "wordpress/wp-config.php"
Rename the file "wordpress/wp-config_NEW.php" to "wordpress/wp-config.php"

Step 3 - Build the WordPress Software Stack

Run the command:

```
docker-compose up -d –build
```

When this completes, point a Browser at http://localhost/my-wordpress/

You should see the WordPress Install screen.

Step 4 – Install WordPress

Follow the instructions, making a note of account names and passwords.
I recommend setting up a user of "admin" to manage the system.

Step 5 - Configure WordPress

Login into the WordPress Instance, here: http://localhost/my-wordpress/wp-login.php

Configure "permalinks" within the WordPress "settings", set to "Post name"

Create a new Application Password for the "admin" account.

Write down this password (eg. “NRae TvA6 eq2Z 3Jpd B61K Ehm3”) and the “user_id” number (eg. “1”)

Step 6 - Link the WordPress and the CPW

The WordPress "admin" Account must be linked to the CPW "admin" Account.

To do this, a new Credential row linking the CPW "admin" Account to WordPress "admin" Account must be created, using the previously generated WordPress Application Password.

Login into the CPW with "admin" account, and go to “Authorisation”, “Blog Credentials”.

The new Credential row specifies the CPW User Name, the WordPress “user_id” number and the Application Password.

The CPW User Name and the WordPress User Name MUST match!

Save the new Credential Row.

Step 7 – Stopping WordPress

To bring the system down, use the following command:

```
docker-compose down
```

Part C – Testing

I recommend testing the resulting system by attempting the following tasks:

- Can you send an email?
    - (“Execute Command”, “Send Test Email”; the resulting email will appear in the folder “/home/app/web/emailfiles”, within the CPW Web Container)

- Can you create an Active Collection?

- Can you browse the IDR Image Source?

- Can you add an image from the IDR Image Source to your Active Collection?

- Can you browse the “Your WordPress” Image Source?
    - (You will need to add such an image directly to the WordPress system setup here)

- Can you add an image from the “Your WordPress” Image Source to your Active Collection?

- Can you upload an Image to the “THIS Comparative Pathology Workbench” Image Source?

- Can you add an image from the “THIS Comparative Pathology Workbench” Image Source to your Active Collection?

- Can you upload a Chart to the “EBI Single Cell Atlas” Image Source?

- Can you add an image from the “EBI Single Cell Atlas” Image Source to your Active Collection?

- Can you create a Bench?
    - (This will create a matching Blog Post on the WordPress system – browse that)

- Can you add an Image to a Bench Cell?
    - (This will create a matching Blog Post on the WordPress system – browse that)