pyobs-archive

Tim-Oliver Husser

Feb 15, 2023

CONTENTS

pyobs-archive is a stand-alone archive for FITS images. It provides a general look&feel and an REST API similar to the one used by the LCO archive.

CHAPTER

ONE

INSTALLATION

While it is definitely possible to run *pyobs-weather* without Docker, we highly recommend it for its simplicity.

First, build the image:

```
cd https://github.com/pyobs/pyobs-archive.git
cd pyobs-archive
docker build . -t pyobs-archive
```

pyobs-archive requires a database for storing its data and nginx for serving static files. Easiest way to deploy everything is using docker-compose.

A typical docker-compose.yml looks like this:

```
version: '3'

services:
  db:
    image: postgres:11
    volumes:
      - pgdata:/var/lib/postgresql/data
    restart: always

  archive:
    image: pyobs-archive
    volumes:
      - /local_data/:/data/
      - ./local_settings.py:/archive/pyobs_archive/local_settings.py
      - static:/archive/static
    depends_on:
      - db
    restart: always
    command: bash -c "python manage.py collectstatic --no-input && python manage.py␣
↪makemigrations && python manage.py migrate && gunicorn --workers=3 pyobs_archive.wsgi -
↪b 0.0.0.0:8000"

  nginx:
    image: nginx
    volumes:
      - ./nginx.conf:/etc/nginx/conf.d/default.conf
      - static:/static/static
    ports:
      - 8001:80
```

(continues on next page)

(continued from previous page)

```
    restart: always

volumes:
  pgdata:
  static:
```

The configuration for the "archive" container contains a volume that points to the local directory /local_data/. This will be the directory where the archive stores its files. Please adjust as necessary.

In this example, nginx needs a configuration file nginx.conf in the same directory, which might look like this:

```
server {
    listen 80;
    server_name  127.0.0.1;
    client_max_body_size 50M;

    location / {
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $host;
        proxy_redirect off;
        if (!-f $request_filename) {
            proxy_pass http://archive:8000;
            break;
        }
    }

    location /static/ {
        root /static;
    }
}
```

And *pyobs-archive* itself needs a configuration file called local_settings.py. Here is the file for MONET/S as an example:

```python
# disable debug
DEBUG = False

# we're reverse proxying, so only localhost is allowed to access
ALLOWED_HOSTS = ['localhost']

# settings for archive
ARCHIV_SETTINGS = {
    'HTTP_ROOT': 'https://archive.example.com/',
    'ARCHIVE_ROOT': '/data/',
    'PATH_FORMATTER': '{SITEID}/{DAY-OBS}/',
    'FILENAME_FORMATTER': None,
}
```

You have to adjust the settings in ARCHIV_SETTINGS as necessary:

- HTTP_ROOT is the URL the archive will be accessible at.
- ARCHIVE_ROOT is the internal data directory and is mapped in the docker-compose file to a local directory. Should be left as it is.

- PATH_FORMATTER: A format for the pathes to store the image in. Uses FITS header keywords as placeholders.
- FILENAME_FORMATTER: Same as the PATH_FORMATTER, but for the filename. If None, filenames won't be changed.

With all three files in one directory, you can easily do:

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

and the whole system should be up and running within a minute.

Finally, you need to get into the container and create a superuser:

```
docker exec -it weather_weather_1 bash
./manage.py createsuperuser
```

The web frontend should now be accessible via web browser at http://localhost:8001/ and the admin panel at http://localhost:8001/admin.

CHAPTER

TWO

REST API REFERENCE

2.1 Authentication

All requests to the REST API must contain a HTTP header of the form:

```
Authentication: Token <token>
```

Where <token> is an auth token that can be obtained by calling the /api-token-auth/ endpoint with valid credentials.

As an example, you can get a token like this:

```
http https://archive.example.com/api-token-auth/ username=husser password=topsecret
```

Which might return something like this:

```
{"token":"3d46d6b98edef947402e032e73eca7b54661c968"}
```

The token can now be used in other requests:

```
http https://archive.example.com/frames/ "Authorization: Token␣
↪3d46d6b98edef947402e032e73eca7b54661c968"
```

2.2 List images

Images in the archive can easily be listed using the /frames/ endpoint. It accepts HTTP GET parameters for filtering. A typical example would be:

```
http https://archive.example.com/frames/?night=2020-02-01
```

for getting a list of all images taken in the night of 1 Feb, 2020.

Other possible filter parameters are:

- IMAGETYPE: Type of image (see *Filter options* for details).
- binning: Binning of image (see *Filter options* for details).
- SITE: Site the image was taken (see *Filter options* for details).
- TELESCOPE: Telescope the image was taken with (see *Filter options* for details).
- INSTRUMENT: Instrument the image was taken with (see *Filter options* for details).
- FILTER: Filter the image was taken with (see *Filter options* for details).

- RLEVEL: Reduction level (0=unreduced, 1=reduced).
- OBJECT: Name of observed object.
- EXPTIME: Exposure time in seconds.
- night: Night of observation in yyyy-mm-dd format.
- basename: Name of FITS file.
- REQNUM: Request number from robotic system.
- start: Limit to images taken after this, given in isot format.
- end: Limit to images taken before this, given in isot format.
- RA: If RA/DEC are given, limit search to 10’ around position
- DEC: See above.
- limit: Maximum number of images to return.
- offset: Offset for list of images to return, use for pagination together with limit above.
- order: Order results using this column.
- asc: If given, order ascending instead of descending.

2.3 Filter options

A call to the /frames/aggregate/ endpoint gives possible choices for some of the filter options:

```
http https://archive.example.com/frames/aggregate/
```

Might result in something like:

```
{
    "binnings": ["1x1", "3x3"],
    "filters": ["B", "V", "R"],
    "imagetypes": ["bias", "dark", "object", "skyflat"],
    "instruments": ["instr1", "instr2"],
    "sites": ["Paranal", "Mauna Kea"],
    "telescopes": ["39m0","30m0"]
}
```

2.4 Image information

Information about a single image can be retrieved using the /frames/image/ endpoint, e.g.:

```
http https://archive.example.com/frames/1000/
```

More specific information can be obtaines using:

```
# for a list of related images.
http https://archive.example.com/frames/1000/related/

# for the FITS headers in JSON format.
```

(continues on next page)

(continued from previous page)

```
http https://archive.example.com/frames/1000/headers/

# for a preview image.
http https://archive.example.com/frames/1000/preview/
```

A single image can be downloaded via:

```
wget https://archive.example.com/frames/1000/download/
```

2.5 Downloading images

A whole bunch of images can be downloaded via /zip/ and listing the frames in the body, e.g.:

```
wget https://archive.example.com/frames/zip/ --post-data="auth_token=<token>&frame_
↪ids[]=1000&frame_ids[]=1001" -O data.zip
```

The auth token needs to go into the POST body in this case, and a list of image IDs can be added using “frame_ids[]”.

2.6 Uploading images

A registered admin user may send new files to the /create/ endpoint for automatically inserting images into the archive.