extractorAPI: deployment to production

July 2023

Set Up

Development server

During development, the API can be run on the development server provided by Flask through the `run` command. By default, the app is run on `port 5000`.
The development server is intended to be used during local development and is not secure or efficient enough to be used in production.

Production server

In production, we use **Gunicorn** to serve the application.

Install Gunicorn:

- In the application folder, activate the environment
- `pip install gunicorn`
- `gunicorn==20.1.0` added in `requirements.txt`

Run with command `gunicorn -w 4 -b 0.0.0.0:5000 'app:app'`. The API runs on port 5000 but we can choose a different port by modifying the command.
The command was added to the service script for it to be automatically launched with Celery when starting the service.

GPU

exapi on dishas-ia

For the API to be deployed on the GPU DISHAS-IA, a user `exapi` was created. This user on the remote server can be accessed through SSH connection from the local machine. The app was installed on the user account on the GPU following the instructions available on the GitHub repository of extractorAPI.
Redis was already installed on the remote server.
This user was granted `sudo` rights to launch the service `celery_flask.service`.

Service

A script was created to launch both Celery and Flask in the Python environment in which the requirements were installed:

```
/home/exapi/extractorAPI/start_celery_flask.sh
```

A service was created to call the script:

```
/etc/systemd/system/celery_flask.service
```

The sudo rights allow user exapi to launch or interrupt the API through the command `sudo systemctl start celery_flask` or `sudo systemctl stop celery_flask`. To check if the API is running, `systemctl status celery_flask`.
Redis is already running on the GPU and is managed by the Observatory's IT team.

Updating the API after deployment

To update the application after its deployment, modifications to the code can be made on the local computer and pushed on the GitHub repository.
Modifications must be made and pushed on the branches `detect`, `cron` or `iiif` and a pull request must be opened before merging with the `main` branch.
From the extractorAPI repository on the exapi account on dishas-ia, the commits can be pulled from the main branch to update the deployed application.

Sending requests to the deployed API

URL for diagram detection:

```
http://dishas-ia.obspm.fr:5000/run_detect
```

To send a list of manifest URL:

```
curl -X POST -F url_file=@[filename] http://dishas-ia.obspm.fr:5000/detect_all
```

Ressources

How To Set Up Django with Postgres, Nginx, and Gunicorn on Ubuntu 22.04 | DigitalOcean
Usage — waitress 2.1.2 documentation
Deploying Python Applications with Gunicorn | Heroku Dev Center

Flask documentation

Development Server — Flask Documentation (2.3.x)
Deploying to Production — Flask Documentation (2.3.x)
Gunicorn — Flask Documentation (2.3.x)