CURART

User Manual

CURART: Stolen Image Detection

Thomas Doyle - 15350316 - 19-05-2019

User Manual

Contents

Dependencies

- This project depends on the following packages:
    - opencv
        - This is an image processing library, we are using the SIFT algorithm in the contrib_modules. This library is precompiled in the docker image
    - tlsh_hash
        - This is a locally sensitive hashing library. This allows us to hash keypoints and the resulting hashes will be similar if they are in fact similar key points
    - flask
        - This is the webframe work that is used to create the application
    - gunicorn
        - This is the production web server that is used to server the flask application
    - psycopg2-binary
        - A library to connect and query the postgres database

How To Deploy Locally

- This application comes as part of a docker container
- All you will need is `docker` and `docker-compose`
    - Install
        - Ubuntu18.0+

```
# Run with admin privileges i.e sudo
apt-get install docker.io
curl -L
"https://github.com/docker/compose/releases/download/1.24.0/docker-compose-
$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
```

- Other systems and problems please refer to https://docs.docker.com/compose/install/

To deploy create a docker compose with configuration similar to

```
version: '3'

networks:
  proxy:
    external: true
  internal:
    external: false

services:
    curart:
        restart: always
        build:
            context: .
        environment:
            # These should be set in the local environment
            DBUSER: ${DBUSER:-latest}
            DBPASSWORD: ${DBPASSWORD:-latest}
            DBHOST: ${DBHOST:-latest}
            DBNAME: ${DBNAME:-latest}
            DBPORT: 5432 # This is default postgresql port
        labels:
            - traefik.backend=upload
            # Change host to your domain name to work
            - traefik.frontend.rule=Host:upload.dtom.dev
            - traefik.docker.network=proxy
            - traefik.port=8080
        networks:
            - internal
            - proxy
    traefik:
        image: traefik:alpine
        container_name: traefik
        restart: always
        command: --docker
        ports:
            - "443:443"
            - "80:80"
        networks:
            - proxy
        volumes:
            - "/var/run/docker.sock:/var/run/docker.sock:ro"
            # Make sure these point to the config files that you have
            # Already created on the server
            - "/etc/containers/traefik/traefik.toml:/traefik.toml:ro"
            - "/etc/containers/traefik/acme.json:/acme.json:rw"
```

```
        labels:
            - "traefik.frontend.rule=Host:monitor.dtom.dev"
            - "traefik.port=8080"
```

-

Once you have this saved as a `docker-compose.yml` file then you can run the command :

```
docker-compose up -d --scale curart=$instances_required
```

-

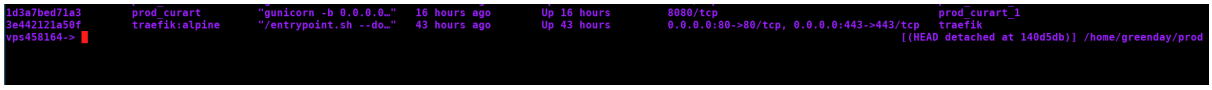
What does this do?

- This is the configuration file to create two docker containers.
    - The first is this application.
        - The `restart` section we tell the container to restart every time it is stopped unless stopped by the command `docker stop`
        - The `build` section we give the path where it will find the files to build into the container
        - The `environment` section we pass in environment variable that we have in the system into the docker containers environment. We are doing this to pass secrets.
        - The `labels` section is for working with traefik.
            - We give the backend a name, here we call it upload
            - We give it a domain name to route to the container
            - We tell it what docker network to use to connect the traefik container and the curart container
            - Then we give the traefik port to the container
        - In `networks` section we declare what docker networks this container should be running on
    - The other configuration is for traefik (A reverse proxy)
        - `image`: Because this is not our own image we will pull the image from dockerhub and we specify the image we want here
        - `container_name`: We name the container
        - `restart`: section we tell the container to restart every time it is stopped unless stopped by the command `docker stop`
        - `command`: Once the container starts running we will give it a subcommand to tell it what to run and in what context it is running
        - `ports`: We need to tell the container what to map internal ports to on external ports. Traefik is our only web proxy in this case so we are using 80 and 433

        - `networks`: We declare what docker networks this container should be running on
        - `volumes`: We map directories in the host system to inside the container.
            - Here we are mapping docker.sock so that traefik can see other containers that are stopping and starting and automatically route traffic to them in real time.
            - The next two volume mappings are config files that are independent of the container. So we have created them on the host system and passed them in by mounting the volumes. We don't put these in the repo because they contain secrets and keys for traefik.
        - `labels`: Passes in the traefik port and give a url to route the main monitoring page to.

Verify

- By Running `docker ps` on the host system you should see output similar to the following:



[If you cannot read this image click here]

Scaling

- If you need more application instances simply place that in the docker-compose command as so

```
docker-compose up -d --scale curart=2
```

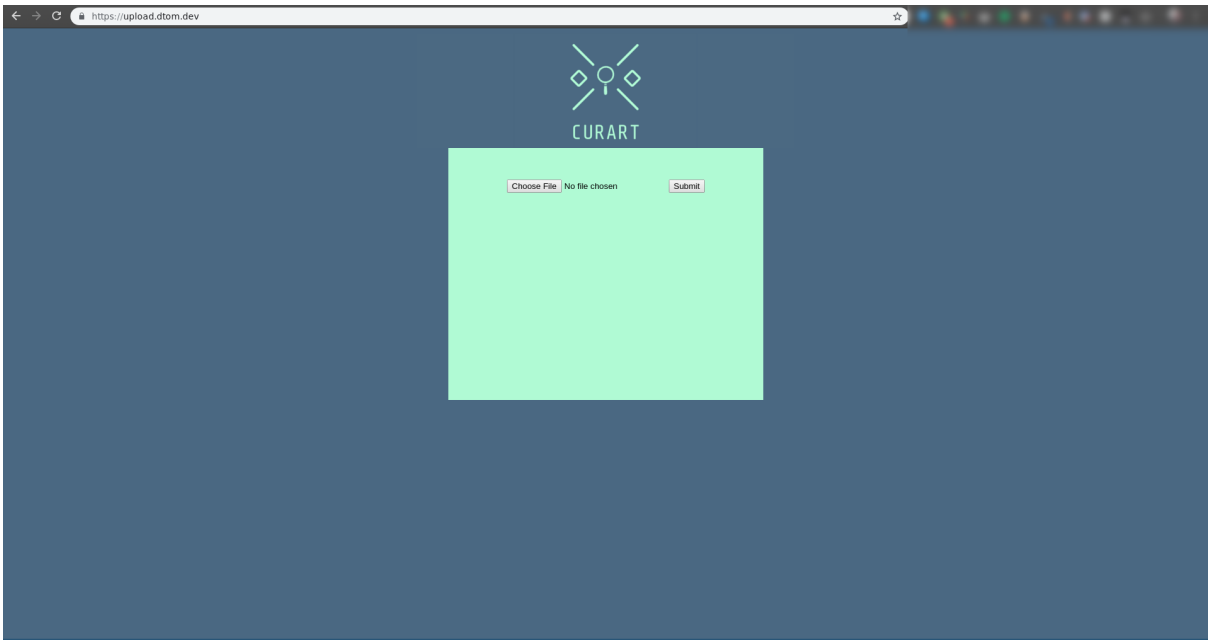
- This will create 2 versions of curart and traefik will automatically recognise there are two and load balance between them
- The load balance method is round robin
- There is no need to stop the containers to anything to scale this and there will be no downtime when scaling up

TODO How to configure and deploy database

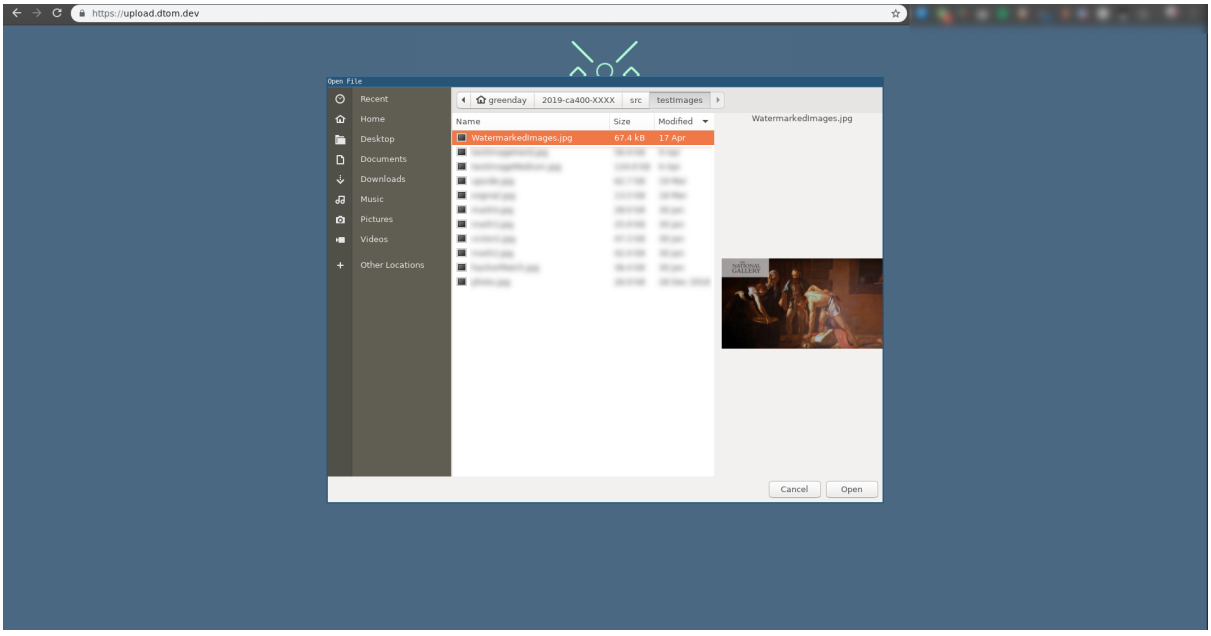
How To Use

Once you have the containers running you can navigate to your url, mine was upload.dtom.dev

You should see the upload screen as so:



Click the `Choose File` button and navigate your system for images:



Once the file is chosen click the `submit` button

An image will be returned that is the original found image

https://upload.dtom.dev/uploader

CURART

Back