Installing CVAT on MAC OS (development)

The project CVAT has 6 different components, which makes it complicated to configure for development purpose.

- PostgreSQL: Database (default port: 5432)
- Redis-Server: Cache (default port: 6379)
- RQ Worker: Redis Job Management
- Django: Server-side Applications (default port: 8000 -dev 8080 -prod)
- ReactJS: Client-side Applications
- Docker: Containerized Management

This document is dedicated to explain step-by-step how to configure a 'development' environment for CVAT at your local server.

Docker

First, we need to setup a PostgreSQL database server for our CVAT project. Since it is independent from main CVAT project, we will initialize it inside a docker container.

```
[docker-compose-development.yml]
version: "2.3"
services:
  cvat_db:
    container_name: cvat_db
    image: postgres:10.3-alpine
    networks:
      default:
        aliases:
          - db
    restart: always
    ports:
      - "5432:5432"
    environment:
      POSTGRES_USER: root
      POSTGRES_DB: cvat
    volumes:
      - cvat_db:/var/lib/postgresql/data
volumes:
  cvat_db:
```

In the configuration file, we exposes 5432 as the PostgreSQL database port, and we set up volumes that persist the information that is stored in **/var/lib/postgresql/data** of the database server.

Volume is a way to communicate between host and docker container. Once a container is stopped, the docker daemon will release all its data. By using volume, our localhost server is able to preserve data among different docker sessions.

```
# building docker images based on docker-compose-development specification
docker-compose build -f docker-compose-development.yml
# running docker containers
docker-compose -f docker-compose-development.yml up -d
```

To ensure the docker container is running, show container status with:

```
$ docker ps
---------------------------------------------------
CONTAINER ID        IMAGE                   COMMAND                  CREATED
STATUS              PORTS                     NAMES
712025d0ee7a        postgres:10.3-alpine    "docker-entrypoint.s…"   3 mins ago
Up 3 mins           0.0.0.0:5432->5432/tcp    cvat_db
```

We also need to make sure that the Database Port in the Django development setting is changed accordingly with our newly exposed PostgreSQL port. We did this by changing **cvat/settings/development.py**

```
DATABASES = {
  'default': {
      'ENGINE': 'django.db.backends.postgresql',
      'HOST': 'localhost',
      'PORT': 5432,
      'NAME': 'cvat',
      'USER': 'root',
      'PASSWORD': os.getenv('POSTGRES_PASSWORD', ''),
    }
}
```

Redis

Redis is a distributed memcache server that is used specifically for fast, regular data retrievals. To set it up, simply run (on MAC OS):

```
redis-server
```

This command is used for initiating a local redis server with an exposed port of 6379. More information in regards to the configuration of Redis can be found in **cvat/settings/base.py** with **CACHEOPS / CACHEOPS_REDIS** configuration parameters. If "redis-server" command isn't found, try to use **brew install** to get the redis server library downloaded.

To monitor the behavior of Redis Server, commonly used for debugging purposes, the "redis-cli monitor" command becomes handy.

```
redis-cli monitor
```

RQ Worker

CVAT uses *ffmpeg* system command to extract media files from uploaded client mp4 file. To avoid job conflicts, the server uses RQ processes to provide additional job management. Note that RQ processes uses Redis server as its memory. Hence, it is important to start RQ worker processes after running redis server.

Two processes need to be created for RQ worker processing two different channels: **low and default**. (Traditional command "rqworker -v … low" will not work since it is dependent to the configurations of CVAT python modules)

```
python3 manage.py rqworker -v 3 low
```

```
python3 manage.py rqworker -v 3 default
```

Update git states process is (optional) to run, and the run script is such:

```
python3 manage.py update_git_states
```

Start Django Server

Once all of the above processes are successfully running in the background, we can finally run our Django Server in development mode.

We first need to create several directories to store our log files and server files.

```
cd ${ROOT_CVAT_PROJECT}
mkdir logs keys
```

Now we can initiate our server in development mode

```
python3 manage.py runserver --mode=development
```

A couple of things to mention here (common errors):

- If the application crushed with "database errors", it is because the existing database structure is different from the definitions in CVAT platform. In this case, you need to run

```
python3 manage.py makemigrations
python3 manage.py migrate
```

  This will allow you to synchronize the PostgreSQL database structure with existing database definitions in Django applications.

- If the application crushed with "cvat.apps… modules not found", it is because the **settings/development.py** did not add the project root in the PYTHONPATH environment variable. A hacky way to solve this is to add this line in **~/.bash_profile** (no space around "=" sign)

```
CVAT_PROJECT_ROOT=~/.../.../cvat
PATH=${CVAT_PROJECT_ROOT}:${PATH}
export PATH
```

Apply changes in the system settings:

```
source ~/.bash_profile
```