Rygen OCR Docker containerization documentation

As in this project we have a total of 3 Repositories.

1. OCR API
2. OCR UI
3. OCR Engine

So for each repository we have created Dockerfile and docker-compose.yml file. Apart from it we have installed Docker version 20.10.21, build baeda1f and docker-compose version 1.29.2, build 5becea4c.
To connect all containers we have also created a docker network named as “external-ocrnet” (which you will find in each docker-compose.yml file).

Following is the command to create docker network.

```
$docker network create external-ocrnet
```

We are also using RabbitMQ and following is the command to create it.

```
$docker run -it --rm --detach--name rabbitmqdev --network external-ocrnet -p 5672:5672 -p 15672:15672 rabbitmq:3.11.4-management
```

1. OCR API
   Following is Dockerfile

```
FROM node:16.15.0-alpine

WORKDIR /app

COPY package.json .

RUN npm install
COPY . .
EXPOSE 2000

CMD ["npm", "start"]
```

Following is docker-compose.yml

```
version: '3.8'
services:

  nginx:
    image: nginx:1.18.0
    container_name: nginx-nodejs
    restart: unless-stopped
    ports:
      - 8444:80
    volumes:
      - ./nginx.conf:/etc/nginx/conf.d/default.conf
    networks:
      - nginxf
    depends_on:
      - api

  api:
    build: .
    container_name: api
    restart: always
    networks:
       - nginxf
    volumes:
      - ./:/app
      - node_modules:/app/node_modules
    depends_on:
      - mongodb

  mongodb:
    image: mongo:4.2.3-bionic
    container_name: mongodb
    volumes:
      - /home/cc/sites/ocr-api-docker-be/shared/mongodb:/data/db
      - /home/cc/sites/ocr-api-docker-
be/shared/mongoconfig:/data/configdb
    networks:
      - nginxf
    ports:
      - 27010:27017


networks:
  nginxf:
    external:
      name: external-ocrnet
volumes:
  mongodb:
  mongoconfig:
```

```
node_modules:
```

To build/create these containers (mentioned above in docker-compose.yml). You have to run

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

[Please note: you must have to be in the same directory where the docker-compose.yml file is.]
This will bring up the following containers, plus we have 1 container of RabbitMQ.

$docker ps

CONTAINER ID	IMAGE	COMMAND NAMES	CREATED	STATUS	PORTS	NAMES
2061d2ac5c38	202301243-32798-11977_ocr-nodejs_api	"docker-entrypoint.s…"	25 seconds ago	Up 25 seconds	2000/tcp	api
e6e9d98c7c51	nginx:1.18.0	"/docker-entrypoint.…."	27 seconds ago	Up 27 seconds	0.0.0.0:8444->80/tcp, :::8444->80/tcp	nginx-nodejs
0c1f040cac40	mongo:4.2.3-bionic	“docker-entrypoint.s…"	27 seconds ago	Up 27 seconds	0.0.0.0:27010->27017/tcp, :::27010->27017/tcp	mongodb
b12fd1d7a0e0	rabbitmq:3.11.4-management	"docker-entrypoint.s…"	27 seconds ago	Up 27 seconds	4369/tcp, 5671/tcp, 0.0.0.0:5672->5672/tcp, :::5672->5672/tcp, 15671/tcp, 15691-15692/tcp, 25672/tcp, 0.0.0.0:15672->15672/tcp, :::15672->15672/tcp	rabbitmqdev

2. OCR UI

Following is Dockerfile

```
FROM node:14.16.1-alpine
WORKDIR /app

COPY package.json .
COPY . .
```

In this we started from Node:14.16.1-alpine, setting up the working directory, and copied all the files.
We are configuring it with the rest of all commands, once this container is created.

Following is docker-compose.yml

```
version: '3.8'
services:

  nginx:
    image: nginx:1.18.0
    container_name: nginx-vuejs
    restart: unless-stopped
    ports:
      - 8445:80
    volumes:
      - ./nginx.conf:/etc/nginx/conf.d/default.conf
      - ./dist:/usr/share/nginx/html
    networks:
      - nginxf
    depends_on:
      - client

  client:
    build: .
    container_name: client
    networks:
       - nginxf
    volumes:
      - ./:/app
      - node_modules:/app/node_modules
    stdin_open: true
    tty: true
networks:
  nginxf:
volumes:
  node_modules:
```

To build/create these containers (mentioned above in docker-compose.yml). You have to run

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

[Please note: you must have to be in the same directory where the docker-compose.yml file is.]
This will bring up the following containers

$docker ps

CONTAINER ID	IMAGE	COMMAND NAMES	CREATED	STATUS	PORTS	NAMES
c7ab15d870cd	nginx:1.18.0	"/docker-entrypoint.…."	1 min ago	Up 1 min	0.0.0.0:8445->80/tcp, :::8445->80/tcp	nginx-vuejs
1a8ad626d195	202301112-32196-11715_ocr-vuejs_client	"docker-entrypoint.s…"	1 min ago	Up 1 min		client

Now, at this stage it's time to update the “client” container that we have built using Dockerfile.
Use following commands from the same directory.

```
$docker-compose exec client npm install
$docker-compose exec client npm rebuild node-sass
$docker-compose exec client npm run build
$docker-compose exec client chown -R node:node dist/
$docker-compose exec client chown -R node:node node_modules/
$docker-compose down
$docker-compose up -d
```

3. OCR Engine
   Following is Dockerfile

```
FROM ubuntu:20.04
RUN apt-get update
RUN apt-get install python3.8 -y
```

```
RUN apt-get install python3-pip -y
WORKDIR /app
ARG DEBIAN_FRONTEND=noninteractive
COPY requirement.txt requirement.txt
#RUN python -m pip install --upgrade pip
RUN pip3 install cython
RUN apt-get update
RUN apt-get install poppler-utils -y
RUN apt install build-essential -y
#RUN apt install python3-dev -y
RUN apt install tesseract-ocr -y
RUN apt install libtesseract-dev -y
RUN pip3 install -r requirement.txt

COPY . .
RUN apt-get install ffmpeg libsm6 libxext6  -y

CMD [ "python3", "main.py"]
```

Following is docker-compose.yml

```
version: '3.8'
services:
  pythondev:
    build: .
    container_name: pythondev
    networks:
       - rabbitmqdev
    volumes:
      - ./:/app
    stdin_open: true
    tty: true
```

```
networks:
  rabbitmqdev:
    external:
      name: external-ocrnet
```

To build/create these containers (mentioned above in docker-compose.yml). You have to run

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

[Please note: you must have to be in the same directory where the docker-compose.yml file is.]
This will bring up the following container

$docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
e51455539afd	2023012412-32834-12004_ocr-python_pythondev	python3 main.py	24 hours ago	Up 24 hours		pythondev

At this point all 7 containers are up and running and their images have also been created. To see all the images use the following command.

$docker images

REPOSITORY	TAG	IMAGE ID	CREATED
202301243-32798-11977_ocr-nodejs_api	latest	1a509a115080	30 hours ago
202301112-32196-11715_ocr-vuejs_client	latest	48ebe531b46e	30 hours ago

2023012412-32834-12004_ocr-python_pythondev	latest	d67e4a619de4	30 hours ago
mongo	4.2.3-bionic	97a9a3e85158	30 hours ago
nginx	1.18.0	c2c45d506085	30 hours ago
rabbitmq	3.11.4-management	f42f6d9c3578	30 hours ago

Finally use docker commit command to save the state of running containers to an image.

```
docker commit [CONTAINER_ID] [new_image_name]
```

These all images could be push either to docker hub, AWS or GCP