UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE INGENIERÍA

Año 2020 - 1er Cuatrimestre

Taller de Programación II (75.52)

ChoTuve

Media Server

Manual del Administrador

Fecha de entrega: 30/07/2020

Grupo 8

79979 – Gonzalez, Juan Manuel (juanmg0511@gmail.com)
82449 – Laghi, Guido (guido321@gmail.com)
86429 – Casal, Romina (casal.romina@gmail.com)
96453 – Ripari, Sebastian (sebastiandripari@gmail.com)
97839 – Daneri, Alejandro (alejandrodaner07@gmail.com)

Contenido

1. Objetivo

El objetivo de este documento es proveer una guía práctica para descargar, configurar y ejecutar una instancia del Media Server utilizado en la aplicación ChoTuve.

El resultado de completar los pasos aquí detallados permitirá al administrador contar con una copia inicializada y totalmente funcional del Media Server, configurada a su gusto.

2. Instalación

En esta sección se describen los requisitos previos y las acciones necesarias para generar una copia local del Media Server.

Para poder comenzar, se necesitará navegar al repositorio de GitHub donde se encuentra disponible el código fuente:

https://github.com/AlejandroDaneri/fiuba-taller-2-media-server

2.1 Requisitos previos

Los requisitos para ejecutar el media server son:

1. NodeJS v12 o superior.
2. NPM v6.14.5 o superior.
3. Contar con Docker instalado en el equipo:
    a. Docker engine v19.03.08 o superior.
    b. Compose v1.25.5 o superior.
    c. No es necesario estar registrado en Docker.

Nota: no existe un requerimiento fuerte de hardware o de sistema operativo adicionales a los necesarios para la utilización de las tecnologías previamente nombradas.

4. Clonar el repositorio de GitHub a un directorio mencionado en la sección anterior

3. Configuración

3.1 Configuraciones en docker-compose

Estas configuraciones se realizarán sobre el siguiente archivo, localizado en el root del directorio seleccionado:

`docker-compose.yml`

Para el servicio *app,* el cual se corresponde a la aplicación propia sobre Node:

Clave	Valor	Descripción
container-name	node-app	Es el nombre que con el que se puede identificar que lo logs que aparecen sobre la ejecución de la imagen de docker se corresponden con la app de node
image	media_server:latest	Nombre de la imagen de docker generada seguida de su respectivo tag, en este caso *latest*
build / context	.	Especifica en qué directorio se encuentra toda la configuración
depends_on	postgres_db	Referencia hacia el servicio de la db para especificar que empiece su ejecución luego de que el container de postgres haya empezado su ejecución.
env_file	.env	Especifica el nombre del archivo que contiene las configuraciones de entorno.
command	npm run dev / npm run test	Se tienen estos dos posibles valores para alternar en el caso que se quiera correr la imagen en modo *development* o *testing* respectivamente
ports	3000:3000	Redirige el puerto de con el que se puede comunicar externamente hacia el container de docker, por simplicidad en este caso se ha optado por el mismo
volumes	.:/chotuve /chotuve/node_modules	Especifica los volúmenes asociados al container. El primero deja una sincronización entre los archivos externos al container y los internos El segundo es necesario para que se

		utilicen los módulos generados estrictamente dentro del docker, y así dejar sus dependencias completamente encapsuladas
links	postgres_db	Específica que genere un canal de comunicación interno con el servicio de la db

Para el servicio *postgres_db,* el cual se corresponde a la aplicación de postgreSQL sólo se nombraran las configuración que son de distinto tipo que las mencionadas para el servicio *app*:

Clave	Valor	Descripción
dockerfile	Dockerfile	Especifica el nombre del archivo dockerfile asociado a esta imagen docker
enviroment	POSTGRES_USER POSTGRES_PASSWORD POSTGRES_DB	Especifica las configuraciones de entorno relativas a la base de datos, en este caso al ser pocas, se optó por definirlas directamente sobre el compose

También se tiene un archivo docker-compose para el entorno de CI Travis, el cual es:

`docker-compose.travis.yml`

El mismo es casi idéntico al anterior, con la salvedad de que el comando a correr sobre el servicio *app* es el del script coverage, para después publicarse sobre coveralls

3.2 Variables de entorno

Estas configuraciones se realizarán sobre el siguiente archivo, localizado en el root del directorio seleccionado en el paso 2.2.1:

`Docker-compose.yml`

En la tabla expuesta a continuación pueden consultarse todos los valores configurables correspondientes al entorno de trabajo local. Las mismas también podrán ser configuradas via plataforma de CI/CD y sobreescribiran a las que están aquí mencionadas

Clave	Valor default	Descripción
LOG_LEVEL	debug	El nivel de log a usar por Gunicorn: "debug", "info", "warning", "error".
PORT	3000	El puerto que utilizará el server Node en el container donde ejecuta.
API_KEY	test_api_key	La clave que protege la API.
DATABASE_URL	postgres://user:pass@postgres_db:5432/test	URL con la cual el servidor se conectará hacia la base de datos.
GOOGLE_APPLICATION_CREDENTIAL	./chotuve-grupo8-afb7494394a6.json	Path relativo para acceder al archivo de configuración de la credenciales de Firebase.

3.4 Condiciones de inicialización

Completos los pasos de instalación y configuración, se obtendrá un un ambiente integrado de front end y base de datos, escuchando conexiones en el puerto 3000 del equipo sobre el que sea ejecutado docker-compose.

El mismo tendrá una base de datos vacía, inicializada con las estructuras correspondientes (estos valores son configurados por variables de entorno en `docker-compose.yml`, pero se recomienda no cambiarlos para tareas de desarrollo o pruebas):

Las variables en `docker-compose.yml` que controlan estos valores son:

Clave	Valor default	Descripción
POSTGRES_DB	db	El nombre de la base de datos a la que se conectará el frontend.
POSTGRES_USER	user	Usuario que utilizará el frontend para conectarse a la base de datos
POSTGRES_PASSWORD	pass	La contraseña a utilizar por el frontend para conectarse a la base de datos.

4. Ejecución

Luego de completados los pasos de instalación y configuración, podrá procederse a levantar el ambiente. Para ello, se deberán ejecutar los siguientes dos comandos, desde el directorio utilizado en la sección 2.1 para clonar el repositorio:

```
docker-compose build
docker-compose up
```

El primer comando generará el ambiente, por lo que la primera vez puede tardar aproximadamente unos 5 minutos mientras todas las imágenes y dependencias son bajadas de Internet. El segundo levantará el ambiente, dejándolo listo para recibir conexiones. El *shell* será ocupado por el output de ambos containers.

Si se deseara ejecutar el ambiente en forma de *daemon*, el segundo comando debe reemplazarse por el siguiente:

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

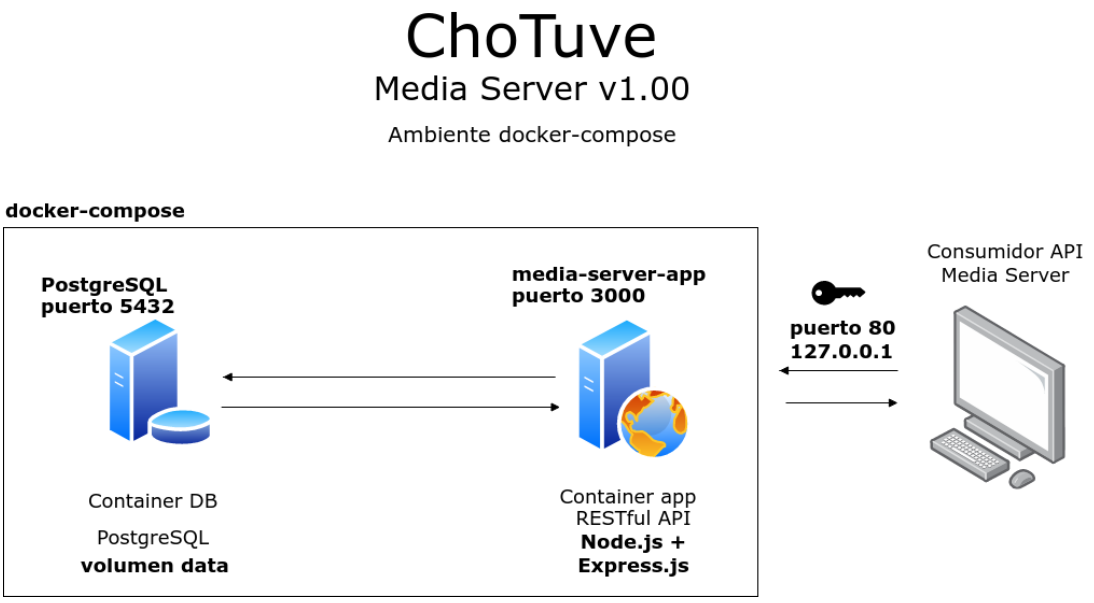
Para ejecutar las pruebas que vienen incluidas con la aplicación, debe cambiarse el valor del docker-compose:

```
command: npm run test
```

Para trabajar en un entorno de desarrollo entonces el comando a utilizar sera:

```
command: npm run dev
```

Esquemáticamente, a nivel ambiente, el resultado obtenido será:



De esta forma, y con una configuración adecuada de puertos, podrían levantarse los tres servers de la solución en el mismo equipo, para realizar por ejemplo tareas de desarrollo.