UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE INGENIERÍA

Año 2020 - 1er Cuatrimestre

Taller de Programación II (75.52)

ChoTuve

AppServer

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 (alejandrodaneri07@gmail.com)

Contenido

1. Objetivo

El objetivo de este documento es proveer una guía práctica para descargar, configurar y ejecutar una instancia del AppServer 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 AppServer, configurada a su gusto.

2. Instalación

En esta sección se describen los requisitos previos y las acciones necesarias para generar una copia local del AppServer. Adicionalmente, se explica como generar un *tarball* con los archivos fuentes, para su distribución.

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

https://github.com/romicasal/fiuba-taller-2-app-server

2.1 Requisitos previos

Los requisitos previos para el **deploy** son los siguientes:

1. 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.
2. Si se desean hacer pruebas, deberá contarse con un cliente REST como Advanced Rest Client (ARC), o alguna otra forma de realizar *requests* sobre el servidor.

Nota: no existe un requerimiento fuerte de hardware o de sistema operativo, estos se ven satisfechos junto a Docker.

Los requisitos para generar los *distributables* mediante **setuptools** son:

1. Python v3.7.7 o superior.
2. Setuptools v47.1.1 o superior.

Nota: no existe un requerimiento fuerte de hardware o de sistema operativo, alcanza con poder ejecutar scripts de Python en el equipo en el que se desee realizar la operación.

2.2 Deploy Local

Para realizar el deploy local del ambiente deberán seguirse estos pasos:

1. Clonar el repositorio de GitHub a un directorio, por ejemplo:

```
~/taller-2-fiuba-app-server
```

2. Verificar la existencia de los archivos, en el *root*:

```
Dockerfile
docker-compose.yml
Requirements.txt
```

2.3 Uso de Setuptools

En caso de querer contar con un archivo para distribución de los fuentes, se incluye un archivo *setup.py* que puede ser utilizado con tal propósito.

La lista de pasos a seguir es la siguiente:

1. Clonar el repositorio de GitHub a un directorio, por ejemplo:

   ```
   ~/taller-2-fiuba-app-server
   ```

2. Ejecutar el siguiente comando, en dicho directorio:

   ```
   python setup.py sdist
   ```

3. Como resultado, se habrá creado un sub-directorio *dist* donde se encontrará el siguiente archivo con los fuentes de la aplicación:

   ```
   fiuba-taller-2-app-server-1.0.tar.gz
   ```

3. Configuración

En esta sección se exponen los pasos necesarios para configurar el *deploy* realizado en la sección anterior.

Cabe destacar que existen configuraciones propias de Compose, y otras del AppServer: estas últimas se pueden controlar mediante el uso de variables de entorno. Adicionalmente, prácticamente todos estos settings traen valores propuestos pre-completados, a fin de facilitar el proceso.

Como última barrera, el código también provee valores default hardcodeados para los valores críticos, pueden consultarse en los archivos fuente.

3.1 Configuraciones propias de Compose

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

`docker-compose.yml`

Los valores editables son:

Clave	Valor default	Descripción
ports	80:8000	Vincula el valor del puerto del container con el frontend (8000) con el equipo local (80).
ports	27017:27017	Vincula el valor del puerto del container con la base de datos (27017) con el equipo local (27017). **Comentado por default, no se expone la DB al equipo local.**

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.

Clave	Valor default	Descripción
GUNICORN_LOG_LEVEL	debug	El nivel de log a usar por Gunicorn: "debug", "info", "warning", "error", "critical".
GUNICORN_WORKERS	4	Cantidad de workers a utilizar por Gunicorn para ejecutar las tareas.
APP_PORT	8000	El puerto que utilizará Gunicorn en el container donde ejecuta.

3.3 Configuración de Gunicorn

A continuación se detallan las configuraciones más importantes de Gunicorn que pueden variarse, sin alterar el funcionamiento del AppServer:

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

`gunicorn_config.py`

Los valores editables son:

Clave	Valor default	Descripción
proc_name	app_server	Nombre del proceso del AppServer.

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 80 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):

- Nombre de la base: “app-server-db”
- Usuario administrador y password: “sa” / “123456”
- Usuario de aplicación y password: “appserveruser” / ”123456”

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

Clave	Valor default	Descripción
MONGODB_DATABASE	app-server-db	El nombre de la base de datos a la que se conectará el frontend.
MONGODB_USERNAME	appserveruser	Usuario que utilizará el frontend para conectarse a la base de datos
MONGODB_PASSWORD	123456	La contraseña a utilizar por el frontend para conectarse a la base de datos.
MONGODB_HOSTNAME	app-server-mongodb	En nombre del container donde se encuentra el servidor de base de datos.
MONGODB_PORT	27017	El puerto a utilizar por el frontend para conectarse a la base de datos.
MONGO_INITDB_ROOT_USERNAME	sa	Nombre de usuario administrador para el servidor de base de datos.
MONGO_INITDB_ROOT_PASSWORD	123456	Contraseña para el usuario administrador para el servidor de base de datos.
MONGO_NON_ROOT_USERNAME	appserveruser	Nombre de usuario de aplicación para el servidor de base de datos.
MONGO_NON_ROOT_PASSWORD	123456	Contraseña de usuario de aplicación para el servidor de base de datos.
MONGODB_DATABASE	app-server-db	Nombre de base de datos inicial para el servidor de base de datos.

Los datos presentes en la base se guardarán en un volumen, el cual persiste entre ejecuciones del ambiente. Para resetear la base de datos, basta con eliminar de docker-compose el volumen del *container* de la base de datos (mongodbdata).

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 definido en el punto 2.2.1:

```
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.

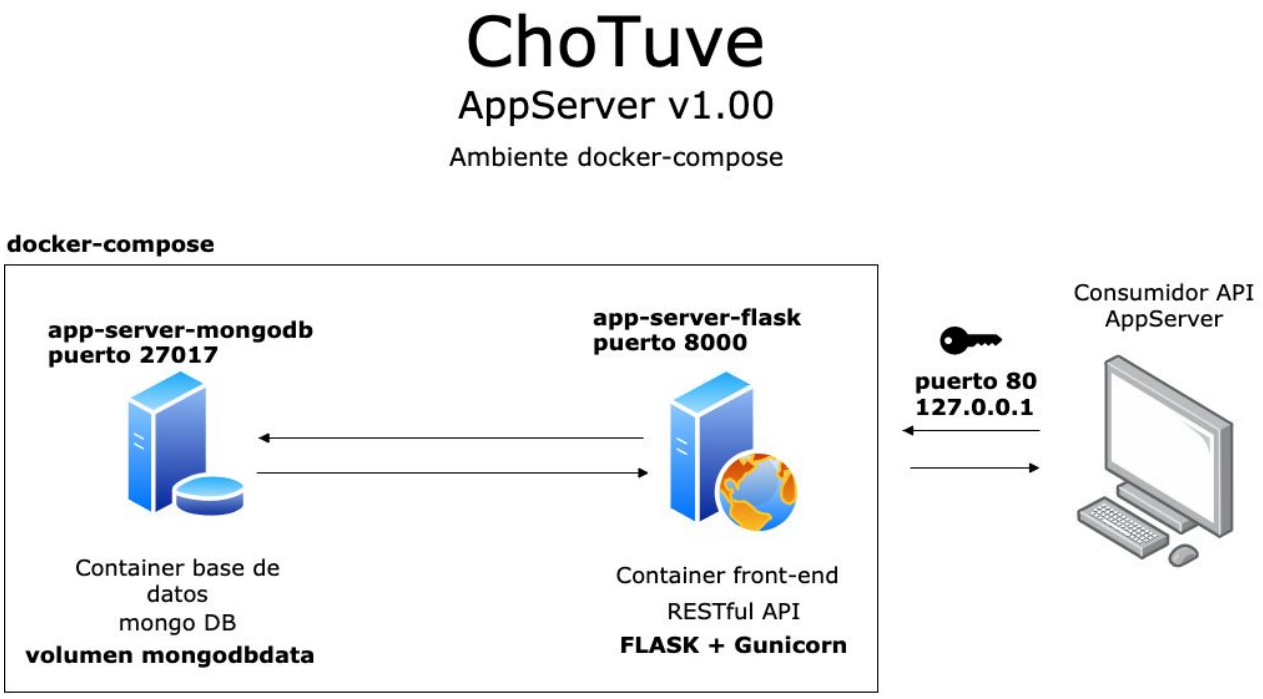
Si se deseara ejecutar el ambiente en forma de *daemon*, el segundo comando debe reemplazarse por el siguiente:

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

Para ejecutar las pruebas que vienen incluidas con la aplicación, pueden ejecutarse los siguientes comandos:

```
docker exec -u root -it app-server-flask bash -c "coverage run --omit
*/virtualenv/* -m unittest discover -v"
docker exec -u root -it app-server-flask bash -c "coverage report"
```

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.