

Backup diario con docker y dropbox - Part II

27 MAY 2016 on docker, backup, cron, dropbox, postgres

En la primera parte de este tutorial vimos como crear la aplicación en dropbox y el docker-compose de la base de datos.

Container

Vamos a crear el segundo contenedor que se va a encargar de hacer los respaldos y subirlos a dropbox.

Para eso editamos nuestro archivo ***docker-compose.yml***

```
1  version: "2"
2
3  services:
4    db:
5      image: postgres
6      environment:
7        - POSTGRES_USER=myuser
8        - POSTGRES_PASSWORD=secret
9        - POSTGRES_DB=mydb
10
11   backup:
12     build: .
13     environment:
14       - POSTGRES_USER=myuser
15       - POSTGRES_PASSWORD=secret
```

```
16 - POSTGRES_DB=mydb
17 - POSTGRES_HOST=db
18 - DROPBOX_ACCESS_TOKEN=4cc3sst0k3nd3dr0pb0x
```

Lo que estamos diciendo es que queremos construir nuestra propia imagen de docker. Nos basaremos en la de postgres pero agregaremos un par de cosas adicionales.

Las líneas de *environment* están repetidas deliberadamente para que sea fácil de leer, pero se puede usar un environment file para evitar duplicar las variables de entorno.

La variable **POSTGRES_HOST** es el nombre del contenedor de la base de datos, en este caso es **db**.

Para poder crear la imagen necesitamos un archivo llamado **Dockerfile**

```
1 # Basamos nuestra imagen en la postgres
2 # esto es importante porque necesitamos el
3 # comando de pg_dump que viene en dicha imagen
4 FROM postgres
5
6 # Actualizamos los repositorios e instalamos curl
7 RUN apt-get update && apt-get -y install curl
8
9 # Copiamos el script que hace los backups en el root
10 COPY backup.sh /
11
12 # Copiamos el archivo con el cronjob en /etc/cron.d/
13 COPY cron-backup /etc/cron.d/
14
15 # Creamos el archivo de log para el cron
16 RUN touch /var/log/cron.log
17
18 # Ejecutamos el comando cron para que corra el cron job
19 # dentro de cron-backup
20 # El comando tail nos ayuda a mantener vivo el contenedor
```

```
21 | # y que no salga despues de ejecutar "cron"  
22 | CMD cron && tail -f /var/log/cron.log
```

Como vemos, nos hacen falta dos archivos, el script y el cron. Sigamos con eso.

Script

Este script es el que se va a encargar de generar el **dump** y de subirlo a **dropbox**. Lo llamaremos **backup.sh**

```
1 | #!/bin/bash  
2 |  
3 | # Este comando hace que si hay un error el script  
4 | # llame un exit con el número del error y no siga  
5 | # ejecutando  
6 | set -e  
7 |  
8 | # Capturamos la fecha actual en formato yyyy-mm-dd  
9 | today=$(date +%Y-%m-%d)  
10 |  
11 | # Generamos el nombre que va a tener el archivo sql  
12 | # por ejemplo mydb-2016-09-29.sql  
13 | # la variable $POSTGRES_DB la toma de las variables de en  
14 | # que pasamos en el docker-compose.yml  
15 | filename="$POSTGRES_DB-$today.sql"  
16 |  
17 | # Generamos el dump de la base de datos y lo guardamos en  
18 | # con el nombre que definimos antes  
19 | pg_dump -h $POSTGRES_HOST -U $POSTGRES_USER $POSTGRES_DB  
20 |  
21 | # Para subir un archivo a dropbox basta con hacer un POST  
22 | # con un access token  
23 | curl -X POST https://content.dropboxapi.com/2/files/upload  
24 |     -H "Authorization: Bearer $DROPBOX_ACCESS_TOKEN" \  
25 |     -H "Dropbox-API-Arg: {\\"path\\": \"/$filename\\", \\"mode\\": "overwrite\\", \\"parent_id\\": null, \\"strict_conflict_resolution\\": true}" \  
26 |     -H "Content-Type: application/octet-stream" \  
27 |     --data-binary @/tmp/$filename
```

Nota: El comando de curl lo tomamos basado en la [documentación de dropbox](#).

Algo muy importante que no se nos puede olvidar a la hora de crear estos scripts (y créanme, con docker van a ser bastantes) es darle permiso de ejecución.

```
[user@localhost] $ | chmod +x backup.sh
[user@localhost] $ | ls -l backup.sh
-rwxr-xr-x  1 harveydf  staff   1020 May 26
```

Esas equis (x) quiere decir que ahora tiene permisos de ejecución.

Cron

```
1 | # La línea en blanco al final es requerido
2 | # para que sea un archivo cron válido
3 | 0 0 * * * root ./backup.sh >> /var/log/cron.log 2>&1
```

Con esto le decimos a cron que queremos correr el script *backup.sh* todos los días a media noche (00:00) y que el resultado se guarde en el archivo `/var/log/cron.log` y que ese mismo resultado lo envíe a la error estándar `/dev/stderr/` y salida estándar `/dev/stdout/` respectivamente. Para mayor información sobre los *cron jobs* pueden empezar con la [Wikipedia](#).

Password

Estamos prácticamente listos pero postgres, a diferencia de mysql, **no** permite pasar el password dentro del comando **pg_dump** sino que debemos crear un archivo oculto llamado

pgpass. Para mayor información en la [documentación de postgres](#).

Para generar este archivo nos vamos apoyar en otra utilidad de docker que es el [entrypoint](#). En resumidas cuentas es un script que se ejecuta cada vez que el container se levanta y antes que el proceso principal se ejecute. Es muy útil a la hora de configurar nuestro contenedor antes de lanzar gunicorn, node, gulp, etc.

Creemos un archivo llamado ***docker-entrypoint.sh***

```
1  #!/bin/bash
2
3  set -e
4
5  # la estructura del archivo es hostname:port:database:user
6  echo "$POSTGRES_HOST:5432:$POSTGRES_DB:$POSTGRES_USER:$PC
7
8  chmod 600 ~/.pgpass
9
10 exec "$@"
```

Nota: recordemos darle permisos de ejecución a este script

Y por último cambiamos el **Dockerfile** para incluir el entrypoint

```
1  # Basamos nuestra imagen en la postgres
2  # esto es importante porque necesitamos el
3  # comando de pg_dump que viene en dicha imagen
4  FROM postgres
5
6  # Actualizamos los repositorios e instalamos curl
7  RUN apt-get update && apt-get -y install curl
8
9  # Copiamos el script que hace los backups en el root
```

```
10 COPY backup.sh /
11
12 # Copiamos el archivo con el cronjob en /etc/cron.d/
13 COPY cron-backup /etc/cron.d/
14
15 # Creamos el archivo de log para el cron
16 RUN touch /var/log/cron.log
17
18 # Copiamos el script de configuracion en el root
19 COPY docker-entrypoint.sh /
20
21 # Usamos el entrypoint
22 ENTRYPOINT ["/docker-entrypoint.sh"]
23
24 # Ejecutamos el comando cron para que corra el cron job
25 # dentro de cron-backup
26 # El comando tail nos ayuda a mantener vivo el contenedor
27 # y que no salga despues de ejecutar "cron"
28 CMD cron && tail -f /var/log/cron.log
```

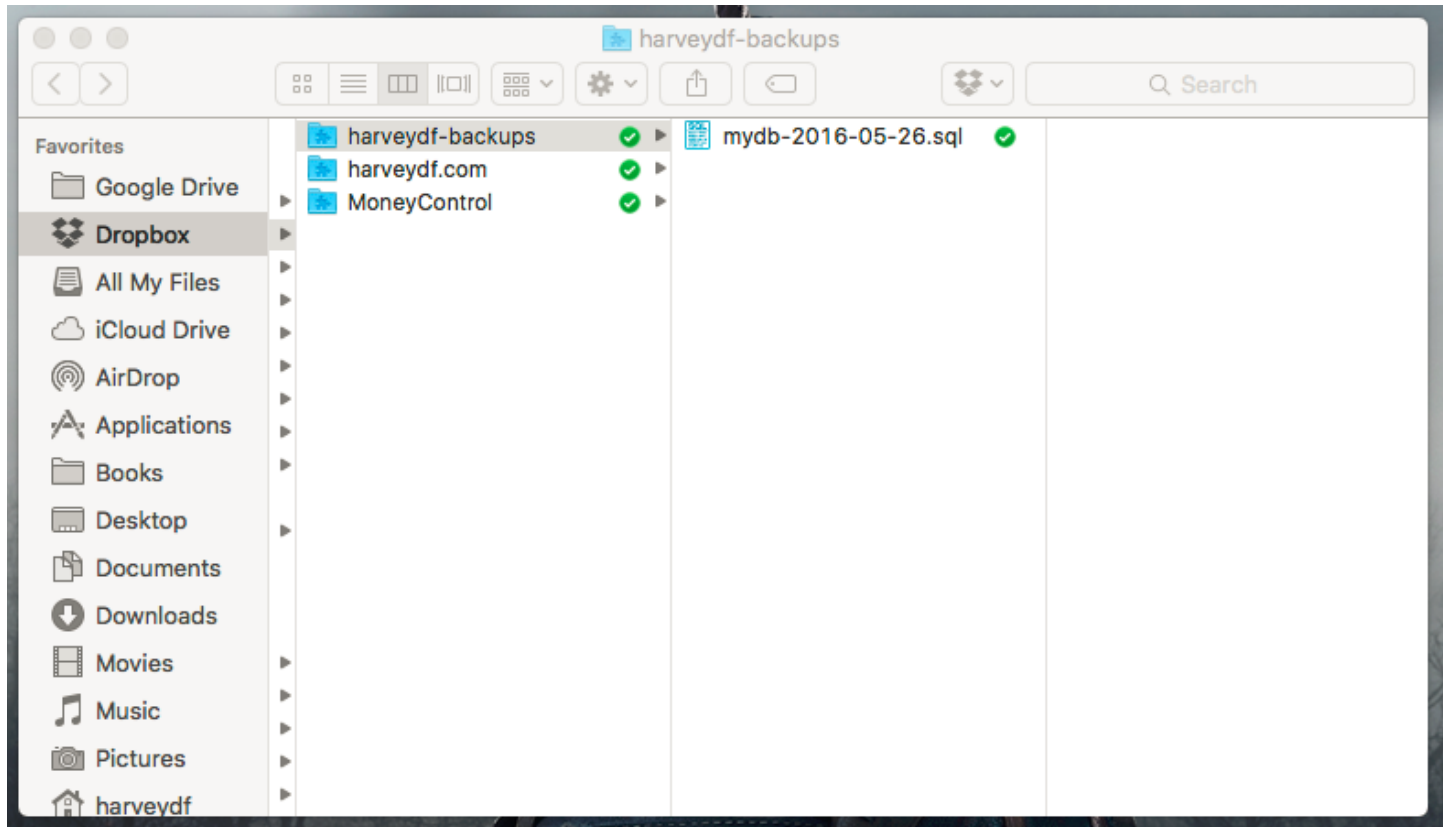
Conclusión

Antes de terminar no se nos olvide generar las imagen de backup `docker-compose build` de nuevo para que tome los cambios hechos en el **Dockerfile**.

Levantamos los contenedores con `docker-compose up -d` y listo, tenemos nuestra base de datos corriendo y un contenedor que se encarga de hacer backup de la base de datos todos los días a media noche y subir ese respaldo a dropbox.

Si no quieren esperar a media noche para ver si funciona, pueden probar el script con `run`

```
[user@localhost] $ docker-compose run backup ./backup.sh
{"name": "mydb-2016-05-26.sql", "path_lower
```



Ahora el reto que les queda es hacer un script que descargue un archivo de dropbox y haga el restore a la base de datos.

Todo el código del tutorial lo pueden encontrar en el [repositorio de github](#).

Si les gustaría algún tutorial sobre algo que vimos (por ejemplo volúmenes en docker) déjenmelo saber en los comentarios. Igualmente cualquier aporte, crítica, comentario y/o duda la pueden dejar en los comentarios.

Hasta una próxima.

Harvey David Forero

Read [more posts](#) by this author.

Share this post



READ THIS NEXT

Django en docker: buenas prácticas


En el post Backup diario con docker y dropbox hablamos de algunos de los beneficios de docker, como la...

YOU MIGHT ENJOY

Backup diario con docker y dropbox - Part I

Docker es una tecnología que está en auge, por su "facilidad de uso", por su portabilidad, por ser multi...

1 Comment harveydf.com

 Javier Martín Alon... ▾ Recommend Share

Sort by Best ▾



Join the discussion...

**Bienvenido Sáez Muelas** • a year ago

Muy bueno y práctico

1 ^ | v • Reply • Share ›




ALSO ON HARVEYDF.COM

Tareas de celery sobre django en tiempo real usando sockets

6 comments • a year ago •

**Bienvenido Sáez Muelas** — Genial, estaría estupendo hacerlo con los Channels de Django**Django en docker: buenas prácticas**

3 comments • a year ago •

**Uriel Reina Abadía** — Buenas Harvey. ¿Como debería poner una carpeta Media? Me imagino que con Subscribe  Add Disqus to your siteAdd DisqusAdd  Privacy