

Cloud Computing: Servicios y aplicaciones

Máster Profesional en Ingeniería Informática

Universidad de Granada

Curso académico 2018-2019

Diseño y despliegue de aplicaciones sobre plataformas cloud

Práctica entregable 1

Morales Garzón, Andrea (andreamgmg@correo.ugr.es)

5 de mayo de 2019

Índice

1. Descripción del problema	1
2. Diseño	1
2.1. Estructura en módulos de la aplicación	1
2.2. Interacción entre los servicios	1
2.2.1. Uso de contenedores	2
2.2.2. Proveedor de Cloud	3
2.3. Selección de las aplicaciones	4
2.3.1. Autenticación basada en LDAP	4
2.3.2. Alojamiento, sincronización y compartición de archivos	4
2.3.3. Base de datos	4
2.4. Escalabilidad de la aplicación	5
3. Despliegue	6
3.1. Servicio de Base de datos	6
3.1.1. Creación de la imagen de Docker	6
3.1.2. Despliegue en Azure	9
3.2. Servicio de Autenticación basada en LDAP	9
3.2.1. Despliegue en Azure	9
3.2.2. Gestión de los usuarios	10
3.3. Servicio de Alojamiento, sincronización y compartición de archivos	11
3.3.1. Despliegue en Azure	11
3.3.2. Configuración con servicio MySQL	12
3.3.3. Configuración con servicio OpenLDAP	15
4. Breve manual de usuario	18
4.1. Forma no automatizada	18
4.2. Forma automatizada	19
5. Mejoras opcionales incluidas por el estudiante	19

1. Descripción del problema

Se debe realizar el diseño y posterior despliegue en la nube de una aplicación destinada a gestión de archivos. Como requisitos, la aplicación precisa de los siguientes servicios:

- Autenticación basada en LDAP
- Alojamiento, sincronización y compartición de archivos
- Base de datos

Para ello, se tomarán diversas decisiones, como establecer qué paquetes de software se van a utilizar para implementar los distintos bloques funcionales, la interacción entre los mismos, los servicios de virtualización, o qué proveedor de Cloud utilizar para llevar a cabo el despliegue.

2. Diseño

2.1. Estructura en módulos de la aplicación

La aplicación que se va a desarrollar y posteriormente desplegar, va a estar formada, tal y como aparece en la descripción del problema, por tres servicios distinguibles:

- Servicio de alojamiento, sincronización y compartición de archivos, donde un usuario de la plataforma, podrá acceder y consultar, añadir, eliminar, modificar sus archivos. Para este servicio, utilizaremos *Owncloud*.
- Servicio de base de datos, donde se alojarán los archivos que se introduzcan en el servicio de alojamiento, sincronización y compartición de archivos, así como cualquier información relevante al mismo. Para este servicio, utilizaremos como base de datos *MySQL*.
- Servicio de autenticación, a través del cuál el usuario podrá autenticarse al servicio de alojamiento de archivos. Para este servicio, se hará uso de *OpenLDAP*.

En secciones posteriores se justificará el uso de cada una de las tecnologías utilizadas.

2.2. Interacción entre los servicios

A partir de la anterior división en servicios, obtenemos una interacción clara entre los mismos:

- El servicio de *Owncloud* depende del servicio de autenticación, en el sentido de que habrá que conectar el servicio de *OpenLDAP* con el servicio para *Owncloud* para que los usuarios incluidos en las listas de usuarios del servicio de OpenLdap puedan autenticarse en *Owncloud* y acceder a sus archivos.
- A su vez, el servicio de *Owncloud* también irá conectado con *MySQL*, el cuál se encargará de almacenar aquello que tiene cada uno de los usuarios en su cuenta.

En la imagen 1 se puede ver, de una forma gráfica, los servicios que se utilizarán, así como la interacción que existe entre los mismos.

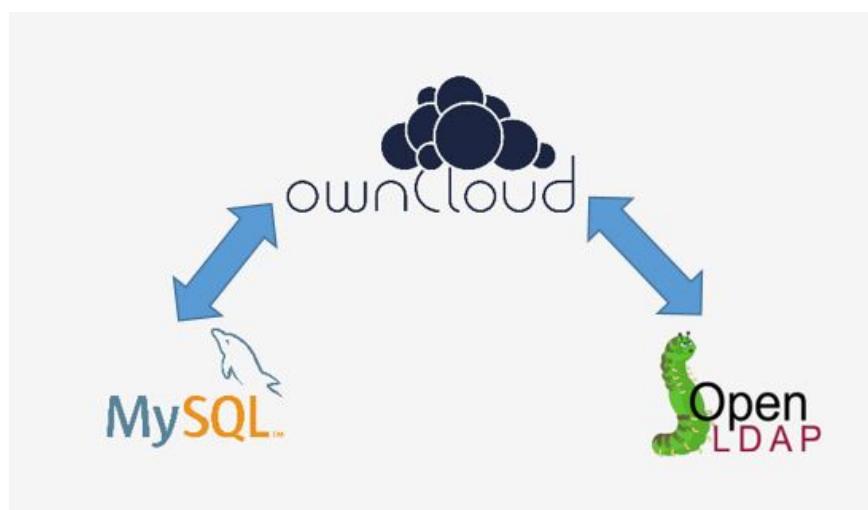


Figura 1: Interacción entre los servicios

2.2.1. Uso de contenedores

Por otra parte, como se explicará en secciones posteriores, para la implementación de cada uno de dichos servicios, (así como para poder llevar a cabo la interacción entre ellos), se hará uso de contenedores, de manera que, cada contenedor alojará uno de los servicios mencionados anteriormente, y tendremos que realizar una configuración que permita, a los distintos contenedores poder comunicarse mediante los servicios anteriormente mencionados.

Las razones por las que me he decantado por el uso de contenedores han sido las siguientes.

- Con contenedores, **podemos definir absolutamente la capa de software** (bibliotecas, sistema operativo, etc). Respecto a este aspecto, nos proporciona absoluta autonomía y flexibilidad, además de la autonomía de tener una máquina independiente. Dicho de otra manera, *incluimos toda la capa de software necesaria para resolver mi problema*.

Por ello, podemos coger una imagen ya existente de alguna plataforma o comunidad de usuarios (véase por ejemplo *Docker Hub*), o crear una nueva personalizada y ajustada justo a aquello que nos hace falta. De ambas formas, podemos coger dicha imagen en cualquier momento, realizar modificaciones, y generar nuevas imágenes con dichos cambios incluidos, que podamos desplegar de manera automática en diversas plataformas. Esta característica genera muchas ventajas, como puede ser la facilidad de uso para el desarrollador, así como la rapidez que le proporciona el uso de las mismas.

- Como hemos visto en clase, **su despliegue requiere menos recursos**, además de que por sí requiere menos espacio físico. Donde antes desplegábamos 10 máquinas virtuales, ahora podemos desplegar 100 contenedores en su lugar. Además, nos proporciona un despliegue rápido con poca carga de usuario.
- Por otra parte, **permite su despliegue en múltiples plataformas**. Si funciona bien en local, va a funcionar bien en cualquier entorno operativo. De esta manera, podremos desplegar nuestra aplicación de la manera más adecuada posible, ya sea, directamente sobre contenedores, o sobre máquinas virtuales, y también independientemente del proveedor de servicios cloud que utilicemos. Esto es una ventaja puesto que, podremos tomar la decisión que, desde el punto de vista económico, sea la más ventajosa para la aplicación.
- Son muy **útiles cuando se elige como paradigma uno orientado a arquitectura SOA** (*Service Oriented Architecture*), donde en lugar de hacer una aplicación muy grande, la dividimos en unidades lógicas a las que llamamos microservicios. Los contenedores son muy útiles en este caso, ya que se pueden ocupar de un único servicio sin saber lo que ocurre en los demás, proporcionando, entre otras cosas, velocidad a la hora de desarrollo.

2.2.2. Proveedor de Cloud

El despliegue se llevará a cabo en *Azure*. Se ha utilizado por dos razones principalmente:

- Es un proveedor de Cloud del que tengo crédito actualmente.
- Lo había utilizado en otras asignaturas, y además sabía la facilidad de uso que proporciona a la hora de desplegar un contenedor, ya sea desde la interfaz o desde línea de comandos.
- Es ampliamente reconocido, utilizado a nivel internacional por una gran cantidad de empresas, con una documentación muy clara y detallada. Además, también ha sido recomendado en una de las charlas de los seminarios de la asignatura.

2.3. Selección de las aplicaciones

2.3.1. Autenticación basada en LDAP

Para el servicio de autenticación basado en **LDAP**, se va a hacer uso de **OpenLDAP**. Entre las razones principales para escogerlo, tenemos, por una parte, que ha sido el recomendado en clase. Además, encontramos que es un servicio que tiene un gran protagonismo a la hora de abordar la gestión de identidades, ya que es ampliamente utilizado hoy en día. Como ejemplo de ello, tenemos que se utiliza en *Kubernetes* o en *Docker*. Además, proporciona mucha flexibilidad a la hora de trabajar con dicha herramienta, y funciona bien con entornos Linux, que son los que estamos utilizando para la aplicación [1].

2.3.2. Alojamiento, sincronización y compartición de archivos

Para este servicio, se ha elegido utilizar **Owncloud**. Por una parte, ha sido el recomendado en clase, y por tanto en el que se han centrado las clases de prácticas. Además, es de código abierto, y ofrece facilidades a la hora de la configuración, sobretodo con LDAP y con bases de datos (la configuración de ambos servicios se puede llevar a cabo mediante la interfaz de una manera relativamente sencilla).

2.3.3. Base de datos

En cuanto a la base de datos a elegir, la decisión se restringe a tres posibilidades, debido a las limitaciones que impone la configuración de *Owncloud*. Las tres opciones proporcionadas, son *MySQL*, *MariaDB* y *PostgreSQL*.

En un principio, se descartó la opción de utilizar PostgreSQL, debido a que la propia configuración de Owncloud la desaconseja, tal y como puede consultarse en [2], recomendando en su lugar una de las otras dos opciones. Además, como se puede consultar por ejemplo en [3], tiene desventajas como puede ser la dificultad para adaptarse a nuevas versiones, los problemas existentes relacionados con la corrupción de las tablas, y problemas de ineficiencia relacionados con la escritura y réplica de datos.

En cuanto a la elección entre *MySQL* y *MariaDB*, hay que tener en cuenta que son, a nivel de funcionalidad, muy similares, ya que *MariaDB* se formó a partir de *MySQL 5.5*, por lo que dicho código es su código de partida en cuanto a implementación. Sin embargo, a pesar del gran parecido, cada elección tiene sus ventajas y desventajas. **En mi caso, se ha escogido MySQL**, por las siguientes razones (teniendo en mi caso más peso la última de ellas).

- Por una parte, *MariaDB*, al tener como punto de partida una versión concreta de *MySQL*, esto tiene como consecuencia que tanto las funcionalidades añadidas como las correcciones de errores desarrolladas para *MySQL* después de dicha versión no formarán parte en *MariaDB*. Es por ello que se queda atrás en cuanto a los nuevos parches que va introduciendo *MySQL* (a pesar de que se llevan a cabo comprobaciones y fusiones del código de manera periódica, intentando esquivar dichos problemas, siempre hay un retraso).
- *MySQL* tiene una versión empresarial (Enterprise Edition), de la que, al ser código de propietario, no dispone *MariaDB*, y por tanto puede tener funcionalidad extra que beneficie a distintos usuarios.
- Además, tenemos que tener en cuenta que, *MySQL*, tiene una gran consolidación en el mercado, que la ha hecho una elección preferente durante muchos años. *MySQL* está disponible prácticamente en todas las plataformas Cloud, característica que *MariaDB* no comparte (no está disponible en *Google Cloud Platflorm*, por ejemplo) tal y como se puede consultar en [4]. De querer usar *MariaDB* en Google Cloud, el desarrollador tendría que instalar y gestionar el servicio de *MariaDB* por su propia iniciativa, lo cuál, puede ser una razón de valor para utilizar *MySQL* en su lugar.

Independientemente de ello, como ya se ha mencionado previamente, las dos son opciones muy similares, y la mejor opción, en muchos casos, dependerá de estudiar el caso de uso a fondo, con el fin de llegar a la opción que mejor resultados pueda proporcionar a la larga.

2.4. Escalabilidad de la aplicación

Respecto a lo necesario para garantizar la escalabilidad de la aplicación, hay que tener en cuenta dos aspectos principalmente.

- En primer lugar, que la estructura de la aplicación está dividida en micro-servicios, y cada uno de esos servicios se encuentra en un contenedor. Con este tipo de arquitectura, podemos escalar de manera sencilla solo aquellos servicios que necesitemos, ya que podemos replicar instancias haciendo uso de la misma imagen predefinida.
- Por otra parte, el otro elemento necesario sería utilizar algún tipo de balanceador de carga, como puede ser **nginx**, el cuál hemos visto en la Sesión 2 del guión de prácticas. Con esta herramienta, podemos escalar horizontalmente servidores de aplicaciones, reduciendo latencia y aumentando así el rendimiento ??.

3. Despliegue

En esta sección, se explicará cómo se ha llevado a cabo el despliegue y desarrollo de cada uno de los servicios por separado, así como la configuración necesaria para poder llevar a cabo la interacción entre ellos. Como ya se ha comentado anteriormente, el despliegue se hará mediante contenedores, para los cuales se ha hecho uso de *Docker*. Todas las pruebas realizadas hasta conseguir un funcionamiento correcto, se han llevado a cabo primero en local (haciendo uso de una red interna de *Docker*). Una vez que funcionaba de manera correcta, se realizó el propio despliegue en el cloud, que es lo que se explicará en esta sección.

3.1. Servicio de Base de datos

3.1.1. Creación de la imagen de Docker

En primer lugar, vamos a empezar con el desarrollo y despliegue del servicio de base de datos, el cuál, como ya se ha dicho anteriormente, utilizará *MySQL*. Para ello, en primer lugar se intentó utilizar la imagen de *Docker* oficial de *MySQL*, pero hubo problemas a la hora de realizar su conexión con otro servicio (problemas que ya se comentaron en clases de prácticas). Por ello, decidí utilizar otras imágenes no oficiales, aunque tampoco conseguí que me funcionaran de forma correcta (en algunos casos por falta de documentación por parte del desarrollador, y en otros casos, porque la configuración que traía me ponía más dificultades que ayuda para lo que yo quería hacer).

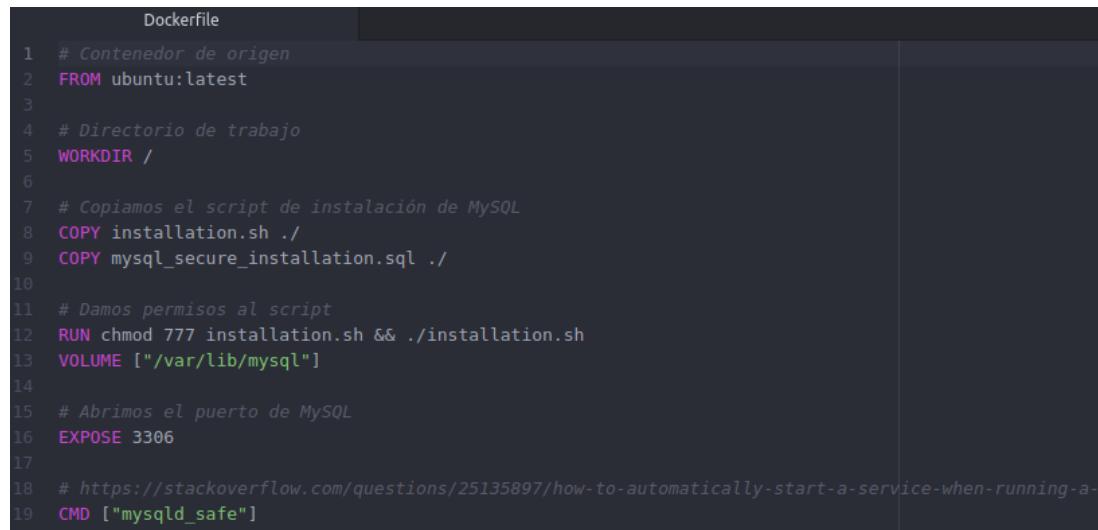
Finalmente me decanté por crear mi propia imagen con la configuración de *MySQL* que necesitaba exactamente, lo cuál, es una de las ventajas de utilizar *Docker*, ya que puedo utilizar imágenes con exactamente aquello que necesito y no más.

Para ello, seguí los siguientes pasos, tomando siempre como referencia, el *Dockerfile* oficial de *MySQL* [5]

1. Primero, partimos de una imagen de Linux, en mi caso, la última versión LTS de Ubuntu.
2. Instalar y configurar *MySQL* de la forma que necesitaba para mi aplicación. Estos cambios se han realizado mediante la ejecución de un script, llamado **installation.sh**.
3. Crear un directorio para guardar los datos de *MySQL*, totalmente necesario para su correcto funcionamiento.
4. Abrir el puerto de *MySQL*, que en mi caso dejaré el puerto por defecto, que es el 3306.
5. Especificar qué queremos que haga el contenedor una vez que se cree. En mi caso, he indicado que quiero que cuando el contenedor se cree, arranque

el servicio de *MySQL* y lo deje funcionando. De esta manera, tan sólo hay que lanzar y crear el contenedor para tener el servicio correctamente configurado y funcionando.

El contenido del fichero Dockerfile ya comentado se puede ver en la imagen 2



```

Dockerfile
1 # Contenedor de origen
2 FROM ubuntu:latest
3
4 # Directorio de trabajo
5 WORKDIR /
6
7 # Copiamos el script de instalación de MySQL
8 COPY installation.sh .
9 COPY mysql_secure_installation.sql .
10
11 # Damos permisos al script
12 RUN chmod 777 installation.sh && ./installation.sh
13 VOLUME ["/var/lib/mysql"]
14
15 # Abrimos el puerto de MySQL
16 EXPOSE 3306
17
18 # https://stackoverflow.com/questions/25135897/how-to-automatically-start-a-service-when-running-a-
19 CMD ["mysqld_safe"]

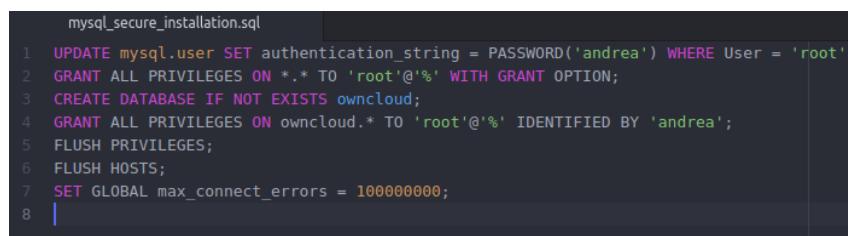
```

Figura 2: Dockerfile para MySQL

Instalación y configuración de MySQL (fichero installation.sh):

Para poder instalar y configurar de manera correcta MySQL, se llevaron a cabo los siguientes pasos.

1. Actualicé la lista de paquetes, e instalé `mysql-server`.
2. Una vez que estaba instalado, inicié el servicio de *MySQL*, y configuré *MySQL* con los requisitos que hubiera especificado de utilizar `my_secure_installation` (que no utilicé al ser interactivo) Estos cambios se ha realizado mediante la ejecución de un archivo con extensión .sql, llamado `my_sql_secure_installation.sql`, que se puede ver en la imagen ??.



```

mysql_secure_installation.sql
1 UPDATE mysql.user SET authentication_string = PASSWORD('andrea') WHERE User = 'root';
2 GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION;
3 CREATE DATABASE IF NOT EXISTS owncloud;
4 GRANT ALL PRIVILEGES ON owncloud.* TO 'root'@'%' IDENTIFIED BY 'andrea';
5 FLUSH PRIVILEGES;
6 FLUSH HOSTS;
7 SET GLOBAL max_connect_errors = 100000000;
8 |

```

Figura 3: Configuración de MySQL

Entre estos requisitos, se encuentran el permitir configuración remota, darle permisos a root o el establecerle una contraseña para el usuario **root**. Además, como necesitamos una base de datos con la que conectar desde *Owncloud*, he añadido a dicho fichero la creación de la misma.

Por último, en dicha imagen se puede ver cómo se establece un máximo de conexiones fallidas bastante alto (última línea). Se ha añadido esa característica dentro de la configuración ya que, algunas veces, Owncloud manda muchas solicitudes (fallidas) para conectarse con MySQL antes de conseguir conectarse, y a veces supera el máximo por defecto, provocando un fallo que se soluciona ejecutando en MySQL **FLUSH HOSTS**; para reiniciar dicho número. Para evitar este error en la medida de lo posible, se ha establecido un umbral alto.

3. Modifiqué manualmente el valor de **bind-address**, de manera que permitiera la conexión para IPs fuera de la máquina y/o contenedor que alojara el servicio.
4. Por último para que se tengan en cuenta estos cambios, tenemos que reiniciar el servicio de *MySQL*.

El contenido del script **installation.sh** se puede ver en la imagen 4.

```
installation.sh
#!/bin/bash

echo "Actualizamos ubuntu para poder encontrar los paquetes"
apt-get update
echo -e "\n"

echo "Instalamos MySQL"
apt-get install -y mysql-server
echo -e "\n\n"

-- echo "Lanzamos MySQL service"
12 service mysql start
13
14 echo "Configuramos MySQL con SECURE INSTALATION"
15 mysql -sfu root < "mysql_secure_installation.sql"
16 # https://stackoverflow.com/questions/30692812/mysql-user-db-does-not-have-password-columns-installing
17
18
19 # Permitimos conexión remota
20 sed -i "s/.*bind-address.*bind-address = 0.0.0.0/" /etc/mysql/mysql.conf.d/mysqld.cnf
21
22 echo "REINICIAR SERVICIO Y GUARDAR CAMBIOS"
23 service mysql restart
```

Figura 4: Instalación de MySQL

Por último, una vez que tenemos el fichero Dockerfile, nos situamos en una terminal en el mismo directorio y ejecutamos la siguiente orden, de manera que podamos crear una imagen a partir del Dockerfile que se ha explicado en el apartado anterior.

```
$ sudo docker build -t andreamorgar/mysql-owncloud .
```

3.1.2. Despliegue en Azure

Para poder desplegar el contenedor en Azure, debemos subir la imagen a *DockerHub*, por lo que, nos logeamos desde la terminal y ejecutamos la siguiente orden, que permite subir la imagen que acabamos de generar a *Dockerhub*.

```
$ sudo docker push andreamorgar/mysql-owncloud
```

En la imagen 5 se puede ver cómo dicha imagen se ha subido a la plataforma de manera correcta.

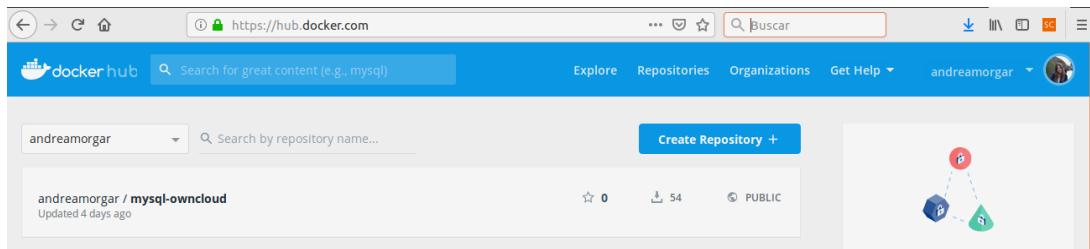


Figura 5: Imagen subida a DockerHub

Por último, faltaría desplegar el contenedor en Azure con la imagen recién proporcionada. Para ello, ejecutamos desde la terminal la siguiente orden:

```
$ az container create --resource-group myResourceGroup --name cmysql --image andreamorgar/mysql-owncloud --ip-address public --ports 3306
```

Con ella, le estamos indicando a Azure que queremos crear un contenedor, con la imagen que acabamos de subir a *DockerHub*, llamado **cmysql**, y con el puerto de *MySQL* abierto. Además, hay que establecer que queremos que cree una IP pública (puede que si no se lo indicamos no cree ninguna), y que forme parte de un grupo de recursos concreto (del que formarán parte todos los contenedores).

3.2. Servicio de Autenticación basada en LDAP

3.2.1. Despliegue en Azure

En este caso, utilizaremos directamente una imagen de *Docker* con *OpenLDAP* ya instalado y configurado. En este caso, he utilizado la que se ha recomendado y utilizado en clase de prácticas, llamada **larrycai/openldap** [6].

En primer lugar, desplegamos en Azure un contenedor, de manera idéntica a como hemos realizado en el apartado anterior, pero en este caso con la imagen de **OpenLDAP** en cuestión. Abrimos en este caso, el puerto que necesitamos, que es el 389:

```
$ az container create --resource-group myResourceGroup --name cldap --image larrycai/openldap --ip-address public --ports 389
```

3.2.2. Gestión de los usuarios

A continuación, insertamos usuarios en dicho contenedor, de manera que podemos probar que realmente nos podemos autenticar posteriormente a través de *Owncloud*. A continuación se muestra el contenido del fichero de prueba, con los datos el usuario de prueba que se va a almacenar en *OpenLDAP*.

```
andreamorgar@andreamorgar:~/MII_CC2_1819$ cat userAndrea2.ldif
dn: cn=andreamorgar,ou=Users,dc=openstack,dc=org
objectClass: inetOrgPerson
cn: andreamorgar
sn: andreamorgar
uid: andreamorgar
mail: andreamorgar@example.com
userPassword: andreamorgar
andreamorgar@andreamorgar:~/MII_CC2_1819$
```

Figura 6: Fichero.ldif del usuario

Para ello, una vez que tenemos el contenedor funcionando, podemos añadir usuarios y realizar consultas a través de su IP. Podemos conocer la IP del contenedor realizando una consulta con el CLI de Azure, y posteriormente filtrándola con `jq`, como se muestra a continuación.

```
$ az container show --name cldap --resource-group myResourceGroup |
    jq -r '.ipAddress' | jq -r '.ip'
```

Una vez conocemos la IP del contenedor, podemos insertar dicho usuario con la siguiente orden, donde `userAndrea2.ldif` es el nombre del fichero que contiene los datos del usuario.

```
$ ldapadd -H ldap://IP -x -D cn=admin,dc=openstack,dc=orgw
password -c -f userAndrea2.ldif
```

También podemos ver los usuarios que existen bajo un dominio concreto con la siguiente orden:

```
$ ldapsearch -H ldap://IP -LL -b ou=Users,dc=openstack,dc=org -x
```

En el ejemplo que estamos viendo, podemos usar dicha consulta para comprobar que realmente se ha añadido, tal y como se puede apreciar en la imagen 7. En ella podemos ver, en resaltado, el usuario que se ha añadido mediante la orden anterior (cuyos datos coinciden con los del fichero mostrado en la imagen 6).

```
andreamorgar@andreamorgar:~/MII_CC2_1819$ ldapsearch -H ldap://40.74.5.98 -LL -b ou=Users,dc=openstack,dc=org -x
version: 1
dn: ou=Users,dc=openstack,dc=org
objectClass: organizationalUnit
ou: Users

dn: cn=Robert Smith,ou=Users,dc=openstack,dc=org
objectClass: inetOrgPerson
cn: Robert Smith
cn: Robert J Smith
cn: bob_smith
sn: smith
uid: rsmith
carLicense: HISCAR 123
homePhone: 555-111-2222
mail: r.smith@example.com
mail: rsmith@example.com
mail: bob.smith@example.com
description: swell guy
ou: Human Resources

dn: cn=Larry Cai,ou=Users,dc=openstack,dc=org
objectClass: inetOrgPerson
cn: Larry Cai
cn: larrycai
sn: Caiyu
uid: larrycai
carLicense: HISCAR 123
homePhone: 555-111-2222
mail: larry.caiyu@gmail.com
description: hacker guy
ou: Development Department

dn: cn=andreamorgar,ou=Users,dc=openstack,dc=org
objectClass: inetOrgPerson
cn: andreamorgar
sn: andreamorgar
uid: andreamorgar
mail: andreamorgar@example.com

andreamorgar@andreamorgar:~/MII_CC2_1819$
```

Figura 7: Consulta y comprobación de usuario añadido en OpenLDAP

Nota: en este caso, se está utilizando el dominio que viene por defecto en el contenedor, tal y como vimos en clase, y no afecta en nada en la sincronización posterior con Owncloud, siempre que se mantenga la coherencia de que el dominio especificado en ambas partes coincida.

3.3. Servicio de Alojamiento, sincronización y compartición de archivos

3.3.1. Despliegue en Azure

En este caso seguiremos un procedimiento muy similar al llevado a cabo en el paso anterior. Para el servicio con *Owncloud*, utilizaremos la imagen de *Docker* oficial de *Owncloud* [7], con la etiqueta `latest`, para así coger la última versión disponible.

Para poder obtener en Azure un contenedor con dicha imagen, ejecutamos la siguiente orden. En ella se puede ver cómo se ha abierto el puerto 80, que será por el cuál se ejecutarán la aplicación.

```
$ az container create --resource-group myResourceGroup --name
owncloud --image owncloud:latest --ip-address public --ports 80
```

Si escribimos la IP correspondiente a este contenedor en el navegador, podemos ver que se accede a la página de inicio de *Owncloud*, donde se creará el administrador y podremos configurar otros aspectos, como la base de datos a utilizar.

3.3.2. Configuración con servicio MySQL

Como hemos comentado en el apartado anterior, nada más iniciar **Owncloud**, podemos configurar la base de datos con la que vamos a trabajar (suponemos que en este momento el contenedor no está apagado y tenemos el servicio funcionando) **aut1aut2**

En dicha página indicaremos el usuario de la base de datos con su contraseña y la base de datos a utilizar (todo lo que ya configuramos en su momento a la hora de crear la imagen para el contenedor que alojaría el servicio de base de datos). En la imagen 8 podemos apreciar cómo quedaría la configuración.

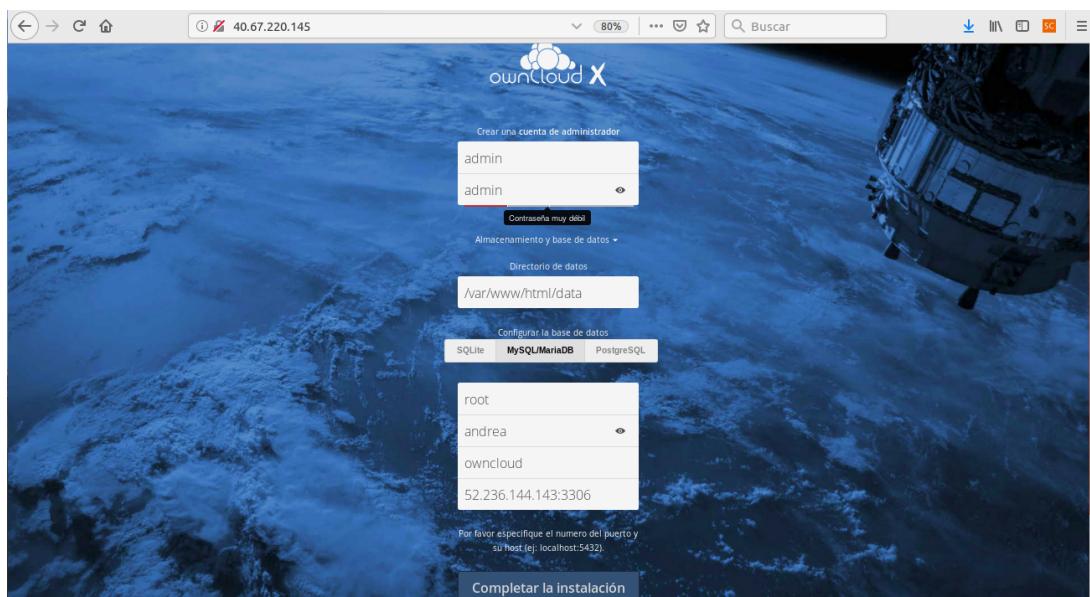


Figura 8: Configuración de la base de datos de Owncloud

Una vez le damos a continuar, nos vuelve a aparecer la página inicial de *Owncloud*, pero ahora ya no salen opciones de configuración, sino que es la página a partir de la cuál un usuario se identificaría en nuestra aplicación (imagen 9). Para entrar, introducimos el usuario y contraseña del administrador recién configurados.

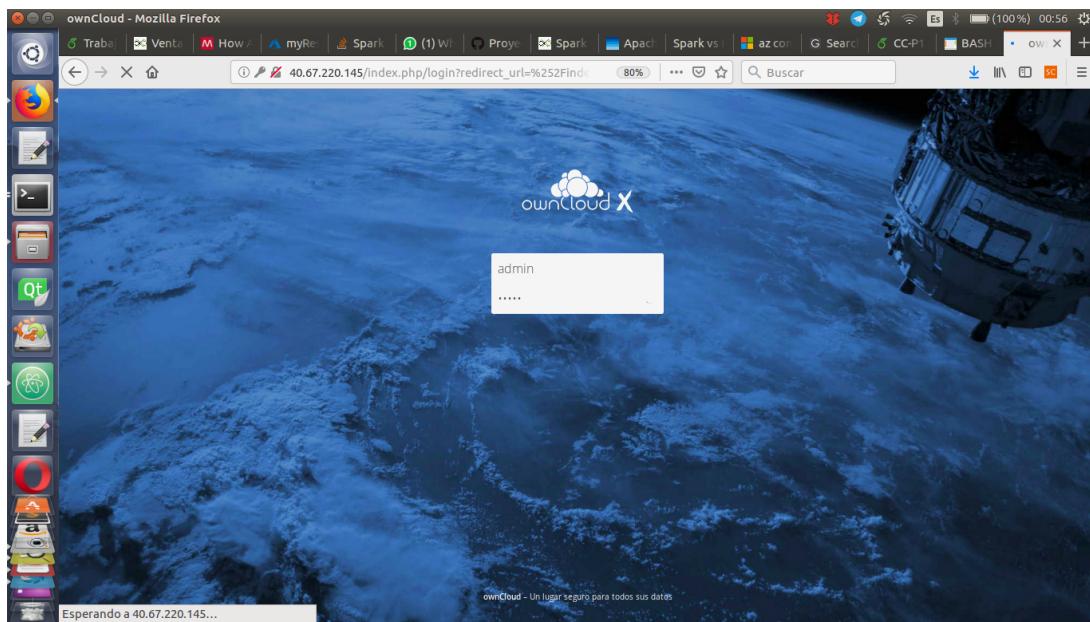


Figura 9: Identificación del usuario

En la imagen 10 podemos la página inicial del administrador en *Owncloud* tras identificarnos con éxito.

Nombre	Tamaño	Modificado
Documents	35 KB	hace segundos
Photos	663 KB	hace segundos
ownCloud Manual.pdf	4.7 MB	hace segundos

Figura 10: Página inicial del Administrador

Si ahora entramos en el servicio de MySQL como root, y usamos la tabla **owncloud** (la definida para este servicio), podemos ver que aparecen ya las tablas resultantes de esta conexión, tal y como se ve en la imagen 11

```

mysql> use owncloud;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_owncloud |
+-----+
| oc_account_terms   |
| oc_accounts        |
| oc_addressbookchanges |
| oc_addressbooks    |
| oc_appconfig        |
| oc_authtokens      |
| oc_calendarchanges |
| oc_calendarobjects |
| oc_calendars        |
| oc_calendarsubscriptions |
| oc_cards            |
| oc_cards_properties |
| oc_comments         |
| oc_comments_read_markers |
| oc_credentials      |
| oc_dav_status       |
| oc_dav_properties   |
| oc_dav_shares       |
| oc_external_applicable |
| oc_external_config  |
| oc_external_mounts  |
| oc_external_options |
| oc_federated_reshares |
| oc_file_locks       |
| oc_groupadmin       |
| oc_groupadmin       |
| oc_group_admin      |
| oc_group_user       |
| oc_groups           |
| oc_jobs             |
| oc_migrations      |
| oc_mimetypes        |
| oc_mounts           |
| oc_notifications    |
| oc_preferences      |
| oc_private_data     |
| oc_properties        |
| oc_schedulingobjects |
| oc_share            |
+-----+

```

Figura 11: Tablas de OwnCloud en MySQL

Por último, vamos a añadir un fichero a *OwnCloud*, con el objetivo de comprobar si se actualiza la base de datos o no, y así ver si está funcionando correctamente. En este caso, como ejemplo, se ha añadido un fichero llamado **prueba.txt**, tal y como se puede ver en la imagen 12.

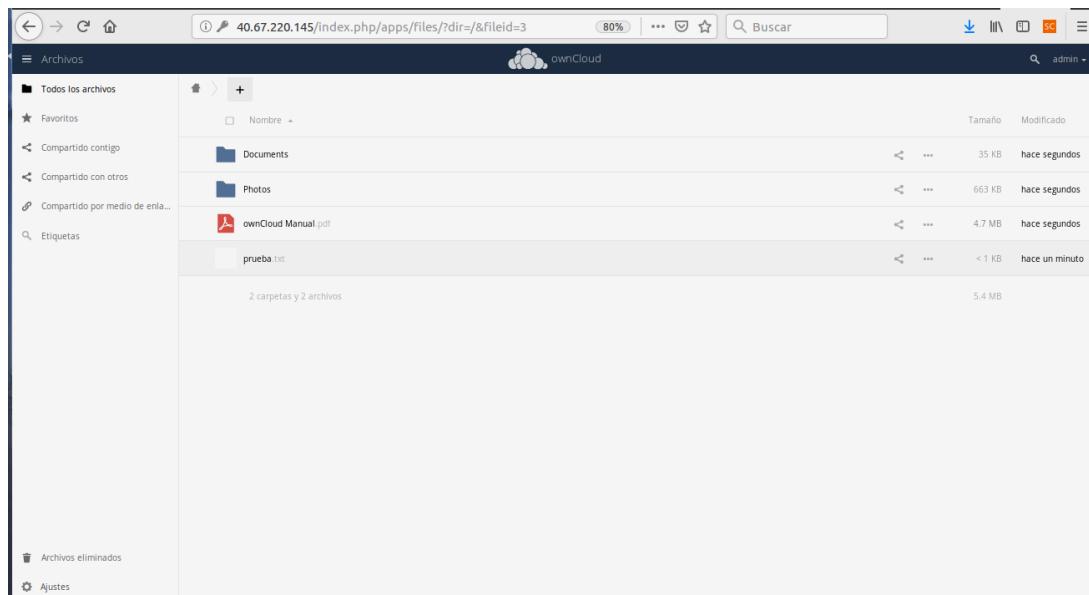


Figura 12: Añadimos un fichero a OwnCloud

En las imágenes 13a y 13b podemos ver el listado de archivos antes y después de añadir dicho fichero. Como podemos ver, funciona correctamente.

```

mysql> select name from oc_filecache;
+-----+
| name |
+-----+
| cache |
| files |
| Documents |
| Photos |
| ownCloud Manual.pdf |
| Example.odt |
| Paris.jpg |
| San Francisco.jpg |
| Squirrel.jpg |
| avatars |
| 21 |
| 23 |
| 2f297a57a5a743894a0e4a801fc3 |
+-----+
15 rows in set (0,06 sec)

mysql> 

```



```

mysql> select name from oc_filecache;
+-----+
| name |
+-----+
| cache |
| files |
| thumbnails |
| Documents |
| Photos |
| ownCloud Manual.pdf |
| drueba.txt |
| Example.odt |
| Paris.jpg |
| San Francisco.jpg |
| Squirrel.jpg |
| avatars |
| 21 |
| 23 |
| 2f297a57a5a743894a0e4a801fc3 |
| 16 |
| 2048-2048-max.png |
| 26-26.png |
| 60-60.png |
+-----+
21 rows in set (0,05 sec)

mysql> 

```

(a) Archivos antes

(b) Archivos después

Figura 13: Resultado de añadir un archivo a OwnCloud

3.3.3. Configuración con servicio OpenLDAP

Para poder configurar OpenLDAP desde la interfaz de Owncloud, lo primero es añadir la extensión. Para ello, nos dirigimos a *Market*, seleccionamos la extensión *LDAP Integration* y la instalamos (imagen 14).

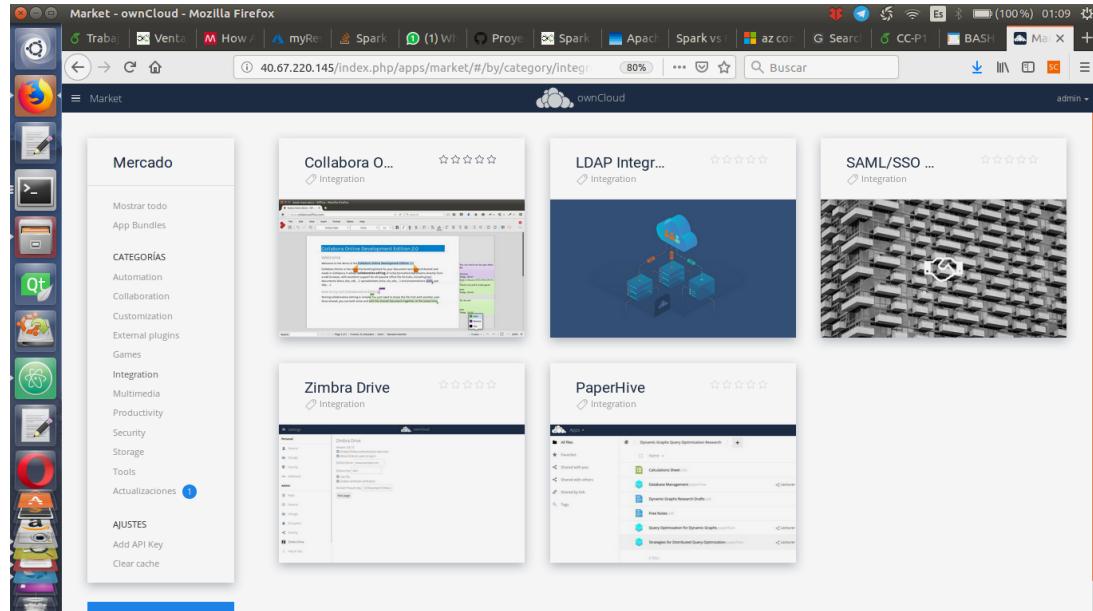


Figura 14: Extensión de LDAP para OwnCloud

De esta manera, cuando accedemos a la ventana de configuración del administrador

dor, ya podemos seleccionar la opción de autenticación de usuario. Lo primero para llevar a cabo la autenticación es la configuración del servidor. Para ello, indicamos la IP, y solicitamos que detecte el puerto. Además, tendremos que indicar el nombre de dominio con el que tenemos identificados a los usuarios dentro de LDAP (en este caso, se ha usado el que viene por defecto, como ya se comentó anteriormente). Podemos verlo en la imagen 15.

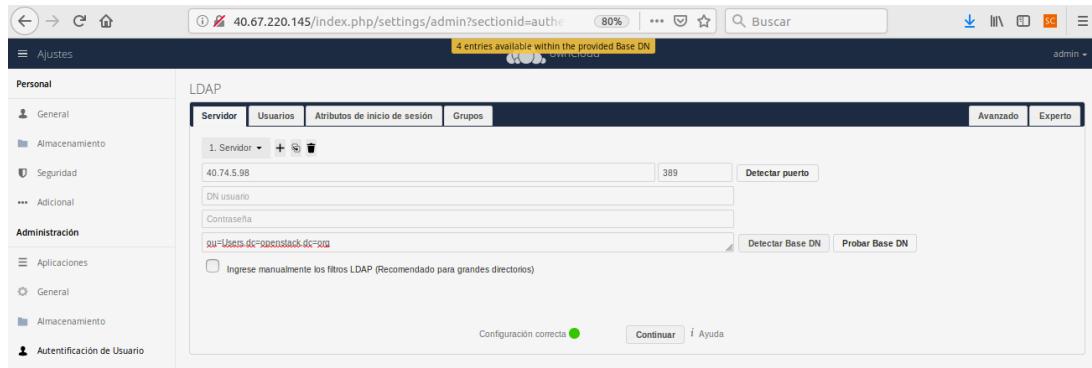


Figura 15: Configuración de autenticación en OwnCloud I

El siguiente paso es la configuración de los usuarios. En este paso, no se ha seleccionado ningún filtro, porque queremos detectar todos los grupos de usuarios almacenados. Se puede apreciar esta configuración en la imagen 16.

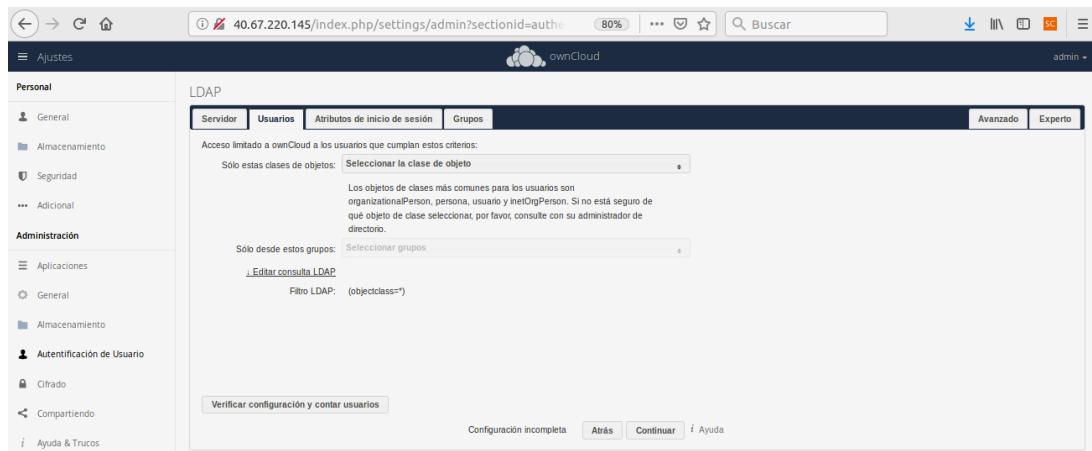


Figura 16: Configuración de autenticación en OwnCloud II

En la pestaña de *Atributos de sesión*, indicamos cómo queremos que los usuarios se identifiquen en el sistema. En este caso, hemos especificado que se utilice el correo, pero se podría haber utilizado el valor de **cn**, o el que se considere mejor. Este paso, es más bien decisión de diseño. Se puede apreciar esta parte de la configuración en la imagen 17. En dicha imagen se puede ver, además, el filtro en cuestión que se va aplicar para poder detectar los usuarios, que es precisamente el comentado.

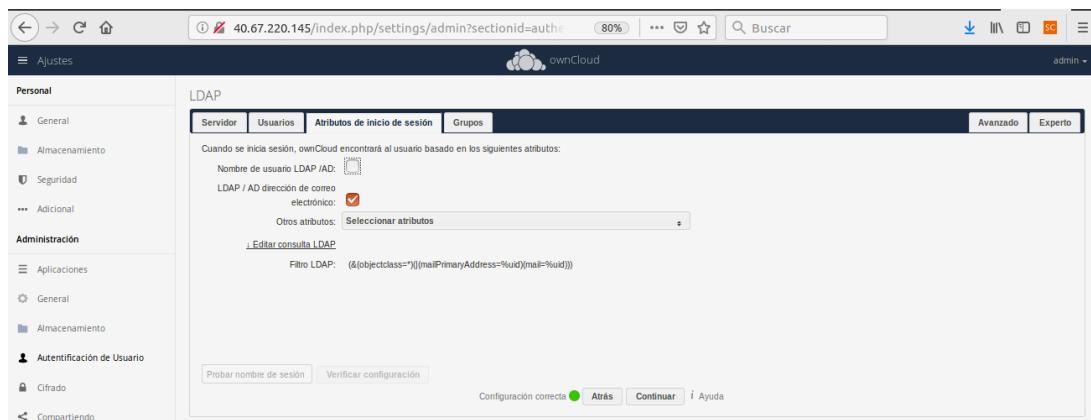


Figura 17: Configuración de autenticación en OwnCloud III

Como último paso de la configuración, quedaría la parte referente a los grupos, la cuál no se ha modificado, y se puede ver en la imagen 18.

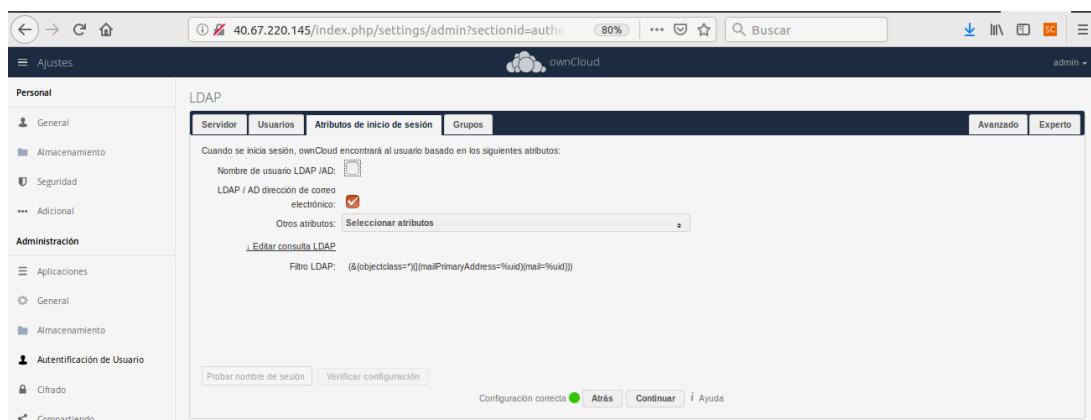


Figura 18: Configuración de autenticación en OwnCloud IV

Prueba de funcionamiento

Para comprobar que funciona, vamos a entrar con el usuario ya comentado anteriormente a Owncloud. Tras una identificación exitosa, aparece su página inicial, que es la mostrada en la imagen 19. En ella podemos ver, la identificación utilizada para el usuario (arriba a la derecha), y que no tiene el fichero **prueba.txt**, ya que fue añadido para el usuario administrador.

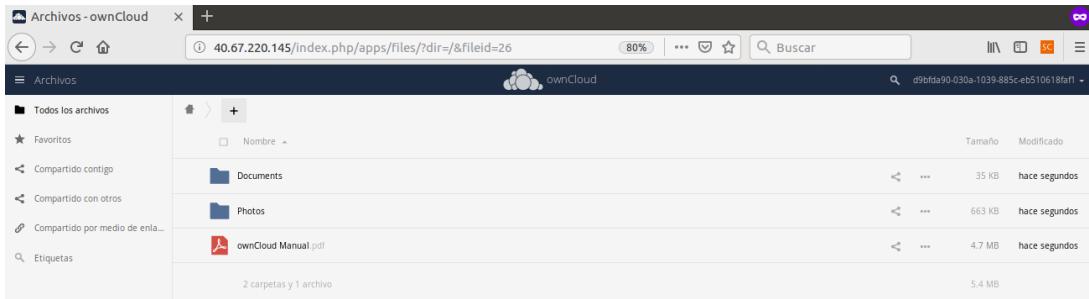


Figura 19: Usuario identificado en OwnCloud I

Si además, consultamos la lista de usuarios que se mantiene en el perfil del administrador, podemos ver que dicho usuario forma parte de esa lista, tal y como aparece en la imagen 20



Figura 20: Usuario identificado en OwnCloud II

4. Breve manual de usuario

Los pasos a seguir, son básicamente los indicados en la sección de despliegue. Sin embargo, para mayor facilidad, vamos a enumarlos de manera que quede todo más claro. Se puede realizar el despliegue de dos formas:

4.1. Forma no automatizada

Replicar los pasos de creación y configuración de contenedores explicados anteriormente. De manera concisa, serían los siguientes:

1. Crear grupo de recursos (si no existiese):

```
$ az group create --location westeurope --name
myResourceGroup
```

2. Crear contenedor para el servicio de la base de datos:

```
$ az container create --resource-group myResourceGroup --name
owncloud --image owncloud:latest --ip-address public --ports
80
```

3. Crear contenedor para el servicio de autenticación:

```
$ az container create --resource-group myResourceGroup --name cmysql --image andreamorgar/mysql-owncloud --ip-address public --ports 3306
```

4. Añadir usuarios al servicio de LDAP:

```
$ az container create --resource-group myResourceGroup --name cldap --image larrycai/openldap --ip-address public --ports 389
```

5. Crear contenedor para el servicio de alojamiento y compartición de ficheros:

```
$ ldapadd -H ldap://IP -x -D cn=admin,dc=openstack,dc=orgw password -c -f userAndrea2.ldif
```

6. Configurar la base de datos con los pasos en la sección 3.3.2.

7. Configurar *OpenLDAP* con los pasos en la sección 3.3.3.

4.2. Forma automatizada

La mejora del proyecto, consiste en un script que automatiza la parte de creación de contenedores. Por tanto, en este caso, los pasos serían los siguientes:

1. En el directorio de trabajo, ejecutar: `$ sh create-containers.sh`
2. Configurar la base de datos con los pasos en la sección 3.3.2.
3. Configurar *OpenLDAP* con los pasos en la sección 3.3.3.

5. Mejoras opcionales incluidas por el estudiante

Como ya se ha mencionado anteriormente, la mejora realizada ha consistido en automatizar la creación de los contenedores en Azure dentro de un mismo grupo de recursos, así como añadir usuarios al servicio de autenticación, con el fin de agilizar el proceso de configuración.

El contenido de este script (llamado `create-containers.sh`) se puede ver en la imagen 21. Nótese, como detalle, que se aprovecha la misma salida del contenedor generado para *OpenLDAP* para obtener el valor de la IP, y así poder añadir usuarios de manera más automatizada.

```
create-containers.sh
#!/bin/bash
# Creando grupo de recursos...
az group create --location westeurope --name myResourceGroup
echo "\n\n Creando contenedor para el servicio de Owncloud..."
az container create --resource-group myResourceGroup --name owncloud --image owncloud:latest --ip-address public --ports 80
echo "\n\n Creando contenedor para el servicio de MySQL..."
az container create --resource-group myResourceGroup --name cmysql --image andreamorgar/mysql-owncloud --ip-address public --ports 3306
echo "\n\n Creando contenedor para el servicio de LDAP..."
mv_ip=$(az container create --resource-group myResourceGroup --name cldap --image larrycai/openldap --ip-address public --ports 389 | jq -r '.ipAddress' | jq -r '.ip')
echo "\n\n Añadiendo usuarios a LDAP..."
echo $mv_ip
ldapadd -H ldap://$mv_ip -x -D "cn=admin,dc=openstack,dc=org" -w password -c -f userAndrea2.ldif
echo "\n\nFinalizado"
```

Figura 21: Script con parte de automatización del proceso

Referencias

- [1] *Why use OpenLDAP?, JumpCloud, Directory-As-Service*, dirección: <https://jumpcloud.com/blog/why-openldap/> (visitado 4 de mayo de 2019).
- [2] *Database configuration on Linux, Owncloud Documentation*, dirección: https://doc.owncloud.com/server/admin_manual/configuration/database/linux_database_configuration.html (visitado 4 de mayo de 2019).
- [3] *Why Uber Engineering Switched from Postgres to MySQL, Uber Engineering*, dirección: <https://eng.uber.com/mysql-migration/> (visitado 4 de mayo de 2019).
- [4] *MariaDB vs MySQL – Comparing MySQL 8.0 with MariaDB 10.3, Tomer Shay*, dirección: <https://www.eversql.com/mariadb-vs-mysql/> (visitado 4 de mayo de 2019).
- [5] *Dockerfile, MySql*, dirección: <https://github.com/mysql/mysql-docker/blob/mysql-server/5.7/Dockerfile> (visitado 4 de mayo de 2019).
- [6] *Imagen de Docker para OpenLDAP, Larry Cai*, dirección: <https://hub.docker.com/r/larrycai/openldap> (visitado 4 de mayo de 2019).
- [7] *Imagen de Docker para Owncloud, Owncloud Official*, dirección: https://hub.docker.com/_/owncloud (visitado 4 de mayo de 2019).