

Asignatura	Datos del alumno	Fecha
Bases de Datos para el Big Data	Apellidos:	
	Nombre:	

## Seminario Redis - Actividad

### Objetivos

Esta actividad te permitirá profundizar en el uso de Redis como motor de base de datos clave-valor. Realizarás una serie de ejercicios guiados que te ayudarán a conocer las diferentes funcionalidades que brindan las claves-valor simples sobre el motor de base de datos Redis.

### Descripción

Para realizar esta actividad, se asume que tienes instalado Docker Desktop en tu ordenador. Si la instalación no está disponible, además de instalar Docker Desktop, debes instalar VirtualBox en su última versión.



**Importante:** mientras realices cada taller en la consola cliente de Redis, procura no limpiar pantalla para que veas los cambios que se van produciendo.

### Instalación de Redis con Docker

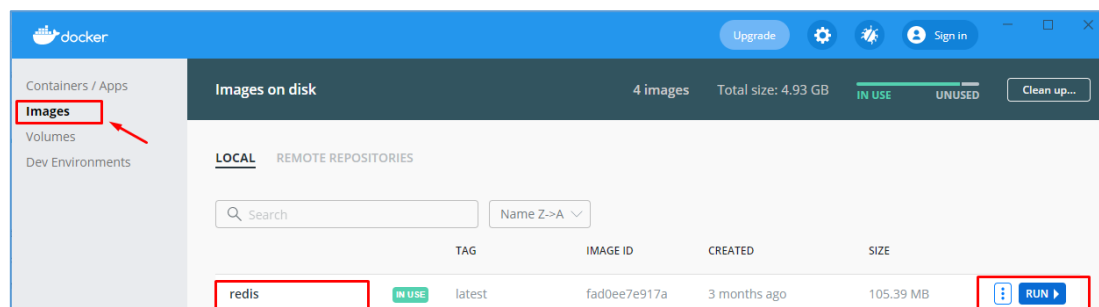
1. Abre una consola de Windows CMD
2. Instala la imagen de Redis sobre Docker:

a) `docker pull redis`

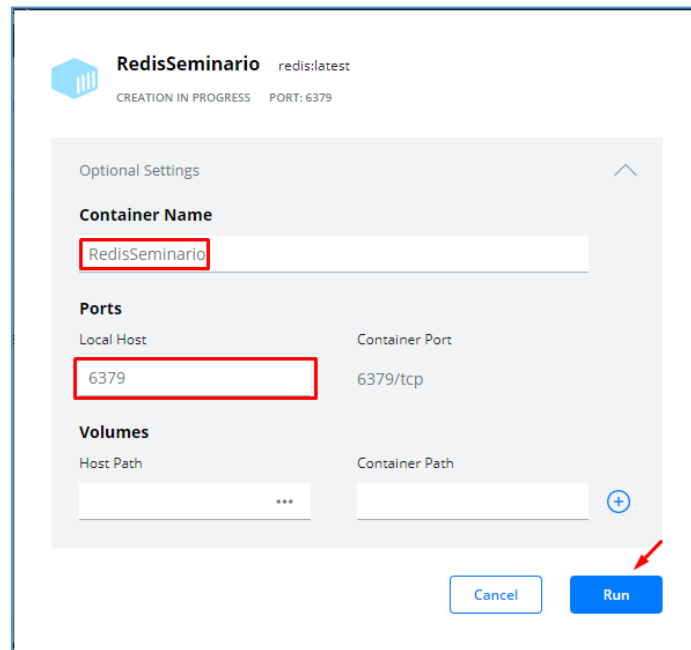
3. Ejecuta un Contenedor de Redis con Docker Desktop:



(tened en cuenta solo las secciones marcadas con recuadro en la imagen, configura el puerto según como se indica en la imagen siguiente).



Asignatura	Datos del alumno	Fecha
Bases de Datos para el Big Data	Apellidos:	
	Nombre:	



**RedisSeminario** redis:latest  
CREATION IN PROGRESS PORT: 6379

Optional Settings

**Container Name**  
RedisSeminario

**Ports**

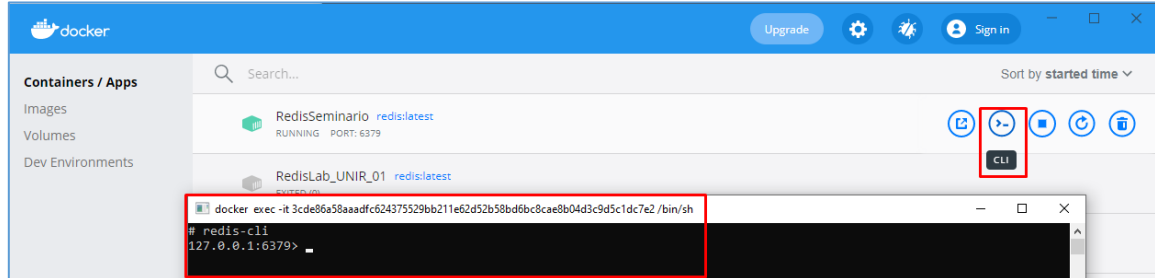
Local Host	Container Port
6379	6379/tcp

**Volumes**

Host Path	Container Path
...	

Cancel Run

4. Accede al cliente de Redis (**# redis-cli**) para continuar con los siguientes talleres:



Llegados a este punto, debes tener un contenedor de Redis en funcionamiento, es decir, un servidor de Redis operativo y aceptando peticiones entrantes:

Asignatura	Datos del alumno	Fecha
Bases de Datos para el Big Data	Apellidos:	
	Nombre:	

The screenshot shows the Docker Desktop application. On the left, there's a sidebar with 'Containers / Apps', 'Images', 'Volumes', and 'Dev Environments'. The main area displays the logs for a container named 'RedisSeminario' (redis:latest) which is in a 'RUNNING' state. The logs show the Redis 6.2.4 startup sequence, including warnings about the config file and the server initialization process. The logs end with 'Ready to accept connections'.

## Taller 1: opera con String

### 1. Ejecuta los siguientes grupos de instrucciones:

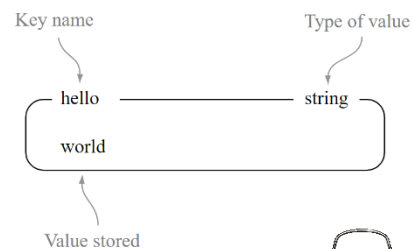
```
SET usuario:1 "Juanito"
SET usuario:2 "Pepe Díaz"
SET web01 "<html><body><h1>404 Error en Web 01</h1></body></html>"
```

```
GET usuario:1
GET usuario:2
GET web01
¿fácil no?
```

```
SET usuario:root "Root"
APPEND usuario:root " Linux"
GET usuario:root
```

```
GETSET usuario:1 "Camilito"
¿Qué ocurre con este otro comando?
```

```
MSET usuario:3 "Javier" usuario:4 "Pablito" usuario:5 "Carlos"
MGET usuario:1 usuario:2 usuario:3 usuario:4
¿Y esta otra forma de crear variables?
```



Asignatura	Datos del alumno	Fecha
<b>Bases de Datos para el Big Data</b>	Apellidos:	
	Nombre:	

SETEX usuario:admin 20 "Administrador"  
*setex usa segundos*  
TTL usuario:admin  
*Repite esta instrucción hasta que se muestre (Integer) 0.*  
GET usuario:admin

PSETEX usuario:root 15 "Root"  
*psetex usa milisegundos*

SETNX usuario:2 "Pepe Daza"  
*¿Se crea la clave?*  
SETNX usuario:8 "Pepe Daza"  
*¿y ahora?*

GETRANGE usuario:2 5 8  
GETRANGE usuario:2 5 -1  
*¿Qué significan Los números 5, 8 y -1?*

SETRANGE web01 14 "2"  
SETRANGE web01 38 "2"  
GET web01  
*¿Qué ha ocurrido con el String?*

keys \*  
keys web\*  
keys usuario\*

STRLEN web01  
STRLEN usuario:4  
*¿Qué valor retorna el comando en ambos casos?*

MSETNX version1 v01 version2 v02 version3 v03  
keys vers\*  
MSETNX version1 v01 version4 v04 version5 v05  
*¿Qué ocurre ahora?*

SET counter\_web01 0  
INCR counter\_web01  
INCR counter\_web01  
INCR counter\_web01  
GET counter\_web01  
*¿Qué ocurre con La variable?*  
INCRBY counter\_web01 5  
*¿Qué ocurre con La variable?*  
INCRBYFLOAT counter\_web01 1.5  
*¿Qué ocurre con La variable?*

INCR counter\_web01  
*¿Algún problema 😊? ¿Cómo Lo resolverías?*

flushdb  
keys \*  
*¿Qué ha pasado? 😊*

```
127.0.0.1:6379> SETEX usuario:admin 15 "Administrador"
OK
127.0.0.1:6379> TTL usuario:admin
(integer) 9
127.0.0.1:6379> TTL usuario:admin
(integer) 8
127.0.0.1:6379> TTL usuario:admin
(integer) 7
127.0.0.1:6379> TTL usuario:admin
(integer) 6
127.0.0.1:6379> TTL usuario:admin
(integer) 5
127.0.0.1:6379> TTL usuario:admin
(integer) 4
127.0.0.1:6379> TTL usuario:admin
(integer) 3
127.0.0.1:6379> TTL usuario:admin
(integer) 2
127.0.0.1:6379> TTL usuario:admin
(integer) 1
127.0.0.1:6379> TTL usuario:admin
(integer) 0
127.0.0.1:6379> GET usuario:admin
(nil)
127.0.0.1:6379>
```

Asignatura	Datos del alumno	Fecha
Bases de Datos para el Big Data	Apellidos:	
	Nombre:	

## Taller 2: manipula List

```
RPUSH myList "Pablito" "Andrea" "Juan" "Gina"
keys m*
```

¿Qué ha pasado? 😊

```
LPUSE myList "Anita"
```

```
LRANGE mylist 0 -1
```

¿Qué ha pasado? 😊

```
LRANGE myList 0 -1
```

¿Y ahora? 😊

```
RPUSHX mylist "Sandra"
```

```
LPUSE mylist "Sandra"
```

¿Qué retorna la instrucción? ¿Por qué? 😊

```
RPUSHX myList "Sandra Derecha"
```

```
LPUSE myList "Sandra Izquierda"
```

```
LRANGE mylist 0 -1
```

```
LLEN myList
```

```
LINDEX myList 1
```

```
LINDEX myList 4
```

```
LINDEX myList 20
```

¿Qué es (nil) en Python?

```
LINSERT myList BEFORE "Gina" "Manuel"
```

```
LINSERT myList AFTER "Gina" "Miguel"
```

```
LRANGE myList 0 -1
```

¿Qué ha ocurrido? 😊

Sandra Izq.	Anita	Pablito	Andrea	Juan	Manuel	Gina	Miguel	Sandra Der
-------------	-------	---------	--------	------	--------	------	--------	------------

```
RPOP myList
```

```
LPOP myList
```

```
LRANGE myList 0 -1
```

¿Qué ha pasado? 😊

LPOP								RPOP	
Sandra Izq.	Anita	Pablito	Andrea	Juan	Manuel	Gina	Miguel	Sandra Der	

```
RPOPLPUSH myList userLogin
```

```
LRANGE myList 0 -1
```

```
LRANGE userLogin 0 -1
```

¿Qué ha pasado? 😊

```
LINSERT userLogin BEFORE "Miguel" "Admin"
```

```
LINSERT userLogin AFTER "Admin" "Gina"
```

```
LINSERT userLogin BEFORE "Gina" "Admin"
```

```
LINSERT userLogin AFTER "Gina" "Admin"
```

```
LINSERT userLogin AFTER "Gina" "dbadmin"
```

```
LINSERT userLogin AFTER "Miguel" "Admin"
```

```
LINSERT userLogin AFTER "dbadmin" "cliente"
```

```
LINSERT userLogin BEFORE "dbadmin" "cliente"
```

```
LRANGE userLogin 0 -1
```

¿Qué ha ocurrido?

```
LREM userLogin 2 "Admin"
```

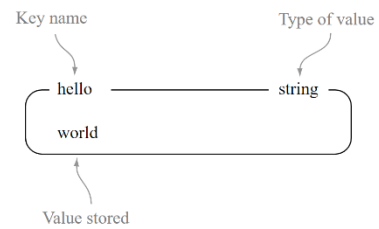
```
LRANGE userLogin 0 -1
```

¿Qué ha ocurrido?

```
LREM userLogin -2 "Admin"
```

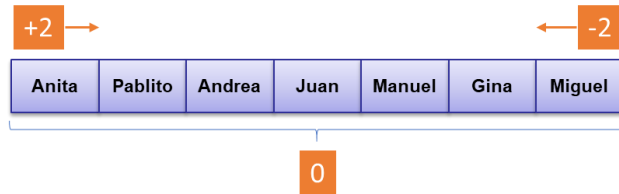
```
LRANGE userLogin 0 -1
```

¿Qué ha ocurrido?



Asignatura	Datos del alumno	Fecha
Bases de Datos para el Big Data	Apellidos:	
	Nombre:	

LREM userLogin 0 "cliente"  
*¿Y ahora? ¿Qué significa el 2, el -2 y el 0?*



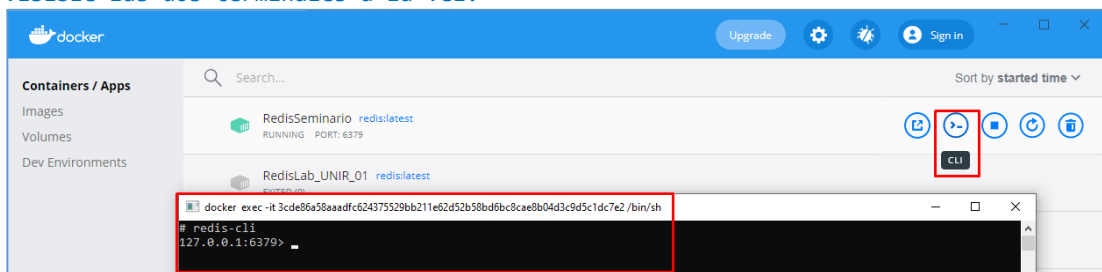
LSET userLogin 2 "dbmongo"  
LSET userLogin 4 "Miguelito"  
LRANGE userLogin 0 -1  
*¿Qué ha ocurrido?*

LPUSH vocales A 0  
LPUSH consonantes B C  
Con BLPOP vas a eliminar elementos de una o varias listas por su izquierda (LPOP) pero, además, podrás añadir un temporizador (timeout en segundos) para que el comando espere (cuando la lista esté vacía) a que otro proceso añada más elementos.  
BLPOP vocales consonantes 1  
BLPOP vocales consonantes 1  
BLPOP vocales consonantes 1  
BLPOP vocales consonantes 1  
BLPOP vocales consonantes 1  
(nil) ya no habría elementos

BLPOP vocales consonantes 5  
El comando esperaría 5 segundos para sacar de las listas un elemento.

**Realiza la siguiente prueba:**

1 - Abre otra terminal de Redis y conéctate en ella al cliente redis-cli. Ten visible las dos terminales a la vez.



2 - En la terminal original ejecuta la instrucción:  
BLPOP vocales consonantes 20

3 - En la nueva terminal ejecuta la instrucción:  
LPUSH vocales E I

4 - Observa qué ocurre en la terminal original.  
5 - Realiza la misma prueba, pero con el comando **BRPOP**.  
6 - Realiza la misma prueba, pero con el comando **BRPOPLPUSH**.  
El comando **BRPOPLPUSH** pasará los elementos de una lista *origen* a otra lista *destino*, y también podrá utilizar el tiempo de espera para que a la lista *origen* le lleguen valores en caso de que esté vacía.

Asignatura	Datos del alumno	Fecha
Bases de Datos para el Big Data	Apellidos:	
	Nombre:	

## Taller 3: juega con Sets

SADD usuariosweb cliente15 admin01 userweb01  
 userweb02  
 SMEMBERS usuariosweb  
*¿Qué ha ocurrido con el orden de los elementos?*

SCARD usuariosweb  
*¿Qué retorna?*

SADD usuariosdb admin01 userweb01 userweb03  
 SMEMBERS usuariosweb  
 SMEMBERS usuariosdb  
*¿Hay usuarios comunes en el Set?*

SDIFF usuariosdb usuariosweb  
*¿Qué retorna?*

SDIFF usuariosweb usuariosdb  
*¿Qué retorna?*

SDIFFSTORE usuariosdiff usuariosweb usuariosdb  
 SMEMBERS usuariosdiff  
*¿Qué ha ocurrido?*

SINTER usuariosdb usuariosweb  
*¿Qué retorna?*  
 SINTERSTORE usuarioscomunes usuariosdb usuariosweb  
 SMEMBERS usuarioscomunes  
*¿Qué ha ocurrido?*

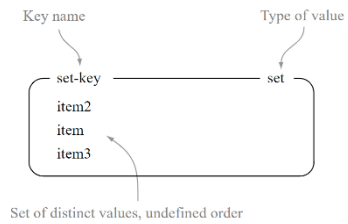
SISMEMBER usuariosdb userweb01  
*¿Qué retorna?*  
 SISMEMBER usuariosdb userweb15  
*¿Qué retorna?*  
*¿Cómo interpretas el resultado en cada caso?*

SMOVE usuariosweb usuariosdb cliente15  
 SMEMBERS usuariosdb  
*¿Qué ha ocurrido?*

SMEMBERS usuariosweb  
 SPOP usuariosweb 1  
 SMEMBERS usuariosweb  
*¿Qué ha ocurrido?*  
 SPOP usuariosweb 2  
 SMEMBERS usuariosweb  
*¿y ahora?*

SADD usuariosweb cliente15 admin01 userweb01 userweb02  
 SMEMBERS usuariosweb  
 SRANDMEMBER usuariosweb 1  
 SMEMBERS usuariosweb  
*¿Qué ha ocurrido en este caso?*  
 SRANDMEMBER usuariosweb 2  
 SMEMBERS usuariosweb  
*¿Qué diferencia hay entre SPOP y SRANDMEMBER?*

SREM usuariosweb userweb01 userweb02  
 SMEMBERS usuariosweb  
*¿Qué ha ocurrido en este caso?*



Asignatura	Datos del alumno	Fecha
<b>Bases de Datos para el Big Data</b>	Apellidos:	
	Nombre:	

SADD usuariosweb userweb01 userweb02

SMEMBERS usuariosweb

SMEMBERS usuariosdb

SUNION usuariosweb usuariosdb

*¿Qué ha ocurrido en este caso? ¿Por qué no se repiten los elementos?*

SUNIONSTORE usuariosunionset usuariosweb usuariosdb

SMEMBERS usuariosunionset

*No merece explicación, ¿a qué no?*



Asignatura	Datos del alumno	Fecha
Bases de Datos para el Big Data	Apellidos:	
	Nombre:	

## Taller 4: crea con Hash

```
HMSET usuario1 id "1" nombre "Juan" email
"juan@email.com" perfil "cliente"
HGETALL usuario1
Muy fácil, ¿no?
```

```
HSET usuario1 ip "1.1.2.3"
HGETALL usuario1
```

```
HKEYS usuario1
HVALS usuario1
HGET usuario1 id
HMGET usuario1 id nombre ip
¿Qué retorna cada instrucción?
```

```
HSETNX usuario1 ip "2.5.8.0"
¿Qué retorna?
HGETALL usuario1
Recuerda, NX equivale a "si no existe"
¿Qué cambio ocurre en usuario1?
HSETNX usuario1 host "2.5.8.0"
HGETALL usuario1
¿y ahora?
```

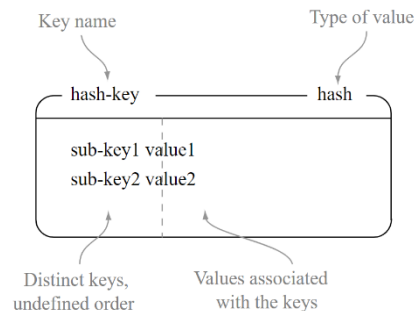
```
HEXISTS usuario1 id
HEXISTS usuario1 url
¿Qué retorna?
¿Cómo interpretas el resultado en cada caso?
```

```
HGETALL usuario1
HDEL usuario1 host
HGETALL usuario1
¿Qué ha ocurrido?
```

```
HLEN usuario1
HKEYS usuario1
```

```
HSETNX usuario1 visitas 0
HGETALL usuario1
HINCRBY usuario1 visitas 1
HGETALL usuario1
HINCRBY usuario1 visitas 3
HGETALL usuario1
Creo que no merece explicación.
```

```
HGETALL usuario1
HINCRBYFLOAT usuario1 visitas 1.3
HGETALL usuario1
HINCRBY usuario1 visitas 1
¿Funciona?
HINCRBYFLOAT usuario1 visitas 0.7
HINCRBY usuario1 visitas 1
¿Y ahora?
```

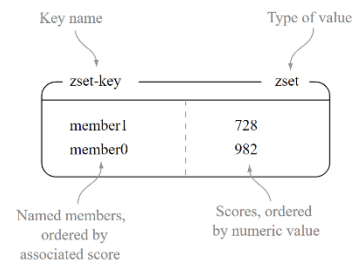


Asignatura	Datos del alumno	Fecha
Bases de Datos para el Big Data	Apellidos:	
	Nombre:	

## Taller 4: avanza con Sorted Sets

Cuatro cosas que debes recordar:

1. Sorted Sets no permite colecciones repetidas
2. Cada miembro del Sorted Set está asociado a un score, el cual es usado para ordenar dicho elemento.
3. Los miembros son únicos pero los scores pueden ser valores repetidos.
4. Usted puede agregar, remover o actualizar elementos muy rápidamente en un Sorted Set (el tiempo proporcional es  $\log(n)$  donde  $n$  es el número de elementos).



**¡ Empecemos !**

```
ZADD idusers 0 "abc000" 1 "abc101" 2 "abc123" 3 "abc155" 3 "abc157" 4 "abc159"
ZRANGE idusers 0 -1
ZRANGE idusers 0 -1 WITHSCORES
¿Todo en orden?
ZRANGEBYSCORE idusers 1 3 WITHSCORES
¿Qué se visualiza ahora?
ZRANGEBYSCORE idusers 1 5 WITHSCORES LIMIT 0 2
```

posición mín. máx.

```
ZRANGE idusers 1 5 WITHSCORES
key
```

score mín. máx.

posición desde - núm. elem.

```
ZRANGEBYSCORE idusers 1 5 WITHSCORES LIMIT 0 2
key
```

```
ZSCORE idusers abc159
ZCOUNT idusers 0 10
0: mín. score y 10: máx. score
ZCARD idusers
¿Qué retorna?
ZRANGE idusers 0 -1 WITHSCORES
¿Concuerda?
```

```
ZPOPMIN idusers 1
1: el número de miembros que eliminaremos
ZRANGE idusers 0 -1 WITHSCORES
Observa la Lista, ¿qué hizo ZPOPMIN?
ZPOPMAX idusers 1
1: el número de miembros que eliminaremos
ZRANGE idusers 0 -1 WITHSCORES
Observa la Lista, ¿qué hizo ZPOPMAX?
```

```
ZRANGE idusers 0 -1 WITHSCORES
ZINCRBY idusers 2 "abc157"
ZRANGE idusers 0 -1 WITHSCORES
Observa la Lista, ¿qué hizo ZINCRBY?
ZINCRBY idusers 3 "abc155"
ZRANGE idusers 0 -1 WITHSCORES
Observa la Lista, ¿qué hizo ZINCRBY? ¿el orden de los miembros es el mismo?
```

Asignatura	Datos del alumno	Fecha
Bases de Datos para el Big Data	Apellidos:	
	Nombre:	

```
ZADD mongouser 0 "hugo" 1 "paco" 2 "luis" 3 "donald" 3 "micky"
ZADD redisuser 0 "miguelangel" 1 "donatelo" 2 "leonardo" 3 "hugo" 3 "luis"
ZRANGE mongouser 0 -1 withscores
ZRANGE redisuser 0 -1 withscores
¿Hasta aquí todo claro no?
```

```
ZINTERSTORE users 2 mongouser redisuser
ZRANGE users 0 -1 withscores
Observa la nueva lista, ¿Qué contiene?
¿A qué equivalen los nuevos scores?
```

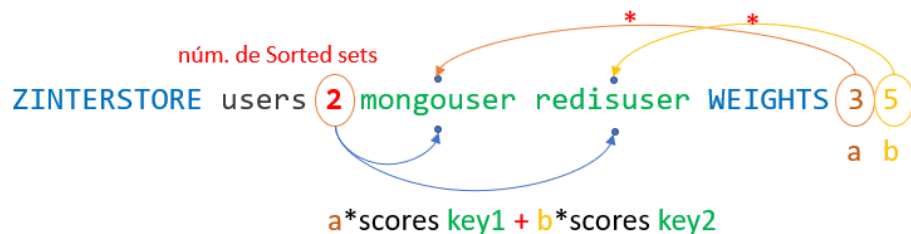


**Recomendación:**  
Visualizar la lista previamente para ver los cambios aplicados.

```
ZRANGE mongouser 0 -1 withscores
ZRANGE redisuser 0 -1 withscores
ZINTERSTORE users 2 mongouser redisuser AGGREGATE MIN
ZRANGE users 0 -1 withscores
Observa la nueva lista, ¿a qué equivalen los nuevos scores?
```

```
ZRANGE mongouser 0 -1 withscores
ZRANGE redisuser 0 -1 withscores
ZINTERSTORE users 2 mongouser redisuser AGGREGATE MAX
ZRANGE users 0 -1 withscores
Observa la nueva lista, ¿a qué equivalen los nuevos scores?
```

```
ZRANGE mongouser 0 -1 withscores
ZRANGE redisuser 0 -1 withscores
ZINTERSTORE users 2 mongouser redisuser WEIGHTS 3 5
ZRANGE users 0 -1 withscores
Observa la nueva lista, ¿a qué equivalen los nuevos scores?
```



```
ZRANGE mongouser 0 -1 withscores
ZRANGE redisuser 0 -1 withscores
ZINTERSTORE users 2 mongouser redisuser WEIGHTS 3 5 AGGREGATE MIN
ZRANGE users 0 -1 withscores
Observa la nueva lista, ¿a qué equivalen los nuevos scores?
```

```
ZRANGE mongouser 0 -1 withscores
ZRANGE redisuser 0 -1 withscores
ZINTERSTORE users 2 mongouser redisuser WEIGHTS 3 5 AGGREGATE MAX
ZRANGE users 0 -1 withscores
Observa la nueva lista, ¿a qué equivalen los nuevos scores?
```

Asignatura	Datos del alumno	Fecha
<b>Bases de Datos para el Big Data</b>	Apellidos:	
	Nombre:	

ZRANGE mongouser 0 -1 withscores  
 ZRANGE redisuser 0 -1 withscores  
 ZUNIONSTORE usersunion 2 mongouser  
 redisuser  
 ZRANGE usersunion 0 -1 withscores

*Observa la nueva lista, ¿a qué equivalen los nuevos scores?*

**Recomendación:**  
 Visualizar la lista previamente para ver los cambios aplicados.

ZRANGE mongouser 0 -1 withscores  
 ZRANGE redisuser 0 -1 withscores  
 ZUNIONSTORE usersunion 2 mongouser redisuser AGGREGATE MIN  
 ZRANGE usersunion 0 -1 withscores

*Observa la nueva lista, ¿a qué equivalen los nuevos scores?*

ZRANGE mongouser 0 -1 withscores  
 ZRANGE redisuser 0 -1 withscores  
 ZUNIONSTORE usersunion 2 mongouser redisuser AGGREGATE MAX  
 ZRANGE usersunion 0 -1 withscores

*Observa la nueva lista, ¿a qué equivalen los nuevos scores?*

ZRANGE mongouser 0 -1 withscores  
 ZRANGE redisuser 0 -1 withscores  
 ZUNIONSTORE usersunion 2 mongouser redisuser WEIGHTS 3 5  
 ZRANGE usersunion 0 -1 withscores

*Observa la nueva lista, ¿a qué equivalen los nuevos scores?*

ZRANGE mongouser 0 -1 withscores  
 ZRANGE redisuser 0 -1 withscores  
 ZUNIONSTORE usersunion 2 mongouser redisuser WEIGHTS 3 5 AGGREGATE MIN  
 ZRANGE usersunion 0 -1 withscores

*Observa la nueva lista, ¿a qué equivalen los nuevos scores?*

ZRANGE mongouser 0 -1 withscores  
 ZRANGE redisuser 0 -1 withscores  
 ZUNIONSTORE usersunion 2 mongouser redisuser WEIGHTS 3 5 AGGREGATE MAX  
 ZRANGE usersunion 0 -1 withscores

*Observa la nueva lista, ¿a qué equivalen los nuevos scores?*

ZRANK mongouser "hugo"  
 ZRANK mongouser "donald"  
 ZRANK mongouser "micky"

*¿Qué retorna, la posición (índice) o el score?*

ZREM mongouser "donald"  
 ZRANGE mongouser 0 -1 withscores

*¿Qué ha pasado con "donald"?*

ZREMRANGEBYRANK mongouser 0 1  
 ZRANGE mongouser 0 -1 withscores

*¿Qué ha pasado ahora?*

ZREMRANGEBYSCORE mongouser 0 2  
 ZRANGE mongouser 0 -1 withscores

*¿Qué ha pasado ahora?*

ZREVRANGE redisuser 0 -1 withscores  
 ¿Qué retorna? ¿Cómo ordena el resultado?  
 ZREVRANGEBYSCORE redisuser 10 0 withscores  
 10: máx. score, 1: mín. score

ZREVRANK redisuser "hugo"  
 ¿Qué retorna, la posición (índice) o el score?



Asignatura	Datos del alumno	Fecha
Bases de Datos para el Big Data	Apellidos:	
	Nombre:	

## Taller 6: Redis y Python



*Si has ejecutado el paso 3 del apartado [Instalación de Redis con Docker Desktop](#), los pasos 1 y 2 a continuación no son necesarios (estos pasos son equivalentes al paso comentado).*

1. Detén la ejecución del contenedor anterior con Ctrl + C (la que creaste con Docker Desktop).
2. Abre una consola CMD de Windows y ejecuta un contenedor de Redis con puerto disponible en el Host (usa el puerto por defecto 6379):

a) `docker run -it -p 6379:6379 redis`

```

C:\Users\mfcardenas>docker run -it -p 6379:6379 redis
1:C 20 Aug 2021 23:49:30.159 # oOoOoOoOoOoOo Redis is starting oOoOoOoOoOoOo
1:C 20 Aug 2021 23:49:30.159 # Redis version=6.2.4, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 20 Aug 2021 23:49:30.159 # Warning: no config file specified, using the default config. In order to specify a config
file use redis-server /path/to/redis.conf
1:M 20 Aug 2021 23:49:30.160 * monotonic clock: POSIX clock_gettime

Redis 6.2.4 (00000000/0) 64 bit

Running in standalone mode
Port: 6379
PID: 1

https://redis.io

1:M 20 Aug 2021 23:49:30.161 # Server initialized
1:M 20 Aug 2021 23:49:30.161 # WARNING overcommit_memory is set to 0! Background save may fail under low memory conditio
n. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.ov
ercommit_memory=1' for this to take effect.
1:M 20 Aug 2021 23:49:30.161 * Ready to accept connections

```

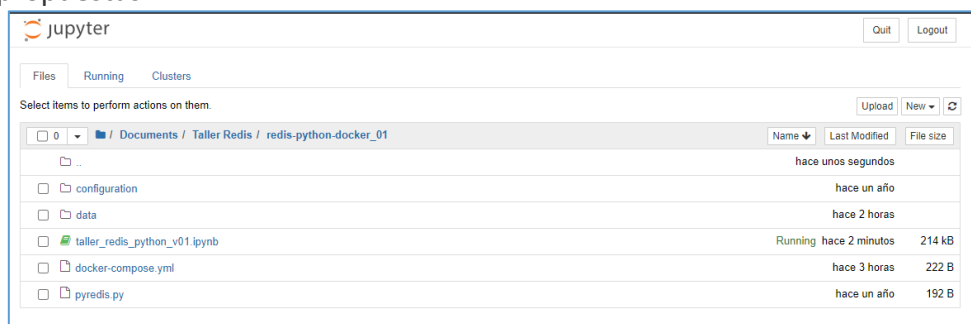
3. Copia el Notebook *"taller\_redis\_python\_v01.ipynb"* a un directorio conocido y accesible por Jupyter, por ejemplo:

`C:/Users/<<Paquito>>/Documents/seminario_redis`



*Este directorio es de ejemplo, es una buena opción si pretendes que Jupyter ejecute el notebook con el que vas a trabajar.*

4. Abre el Notebook con tu instancia de Jupyter y ejecuta todas las celdas propuestas.



Asignatura	Datos del alumno	Fecha
Bases de Datos para el Big Data	Apellidos:	
	Nombre:	

jupyter taller\_redis\_python\_v01 Last Checkpoint: hace 12 minutos (autosaved)

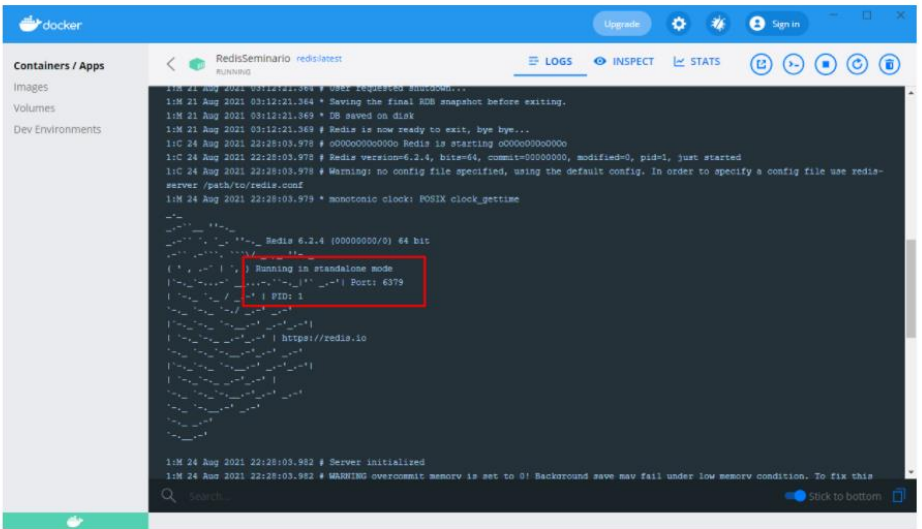
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

## Redis y Python

unir LA UNIVERSIDAD EN INTERNET

### Usa Redis desde Python

Recuerda que al desplegar Redis con Docker, en la terminal puedes observar el puerto de conexión del contenedor.



Instala la api de Redis para Python si aún no cuentas con ella.

## Material de consulta

- Lista de comandos: <https://redis.io/commands/mget>
- Documentación general: <https://redis.com/ebook/redis-in-action/>
- Instalación y configuración de un servidor: <https://docs.redis.com/latest/rs/getting-started/getting-started-active-active/>
- Lista de módulos: <https://redis.com/modules/get-started/>

## Persistencia en Redis

- SAVE: guardar de forma **síncrona** los datos, ejemplo:
  - **SAVE 120 1500** (Guarda cada 120 seg. si se modifican 1500 claves)
  - BGSAVE: guarda de forma asíncrona los datos, ejemplo: **BGSAVE**
- BGREWRITEAOF: inicia el mecanismo de Append Only File. Use INFO para saber cómo va este proceso.
- Configuración de la persistencia: <https://redis.io/topics/persistence>
- Fichero de configuración: <https://gist.github.com/mathieue/5253651>
  - Conoce su ruta con el comando **CONFIG GET DIR**