NoSQL – Key Value

REDIS - INTRODUCCIÓN

Redis es una base de datos en memoria primaria que ofrece alto rendimiento.

Basado en el almacenamiento NoSQL clave/valor.

Persistencia opcional.

REmote **DI**ctionary **S**erver (REDIS)

REDIS - INTRODUCCIÓN

Posee licencia de software libre BSD.

Escrito en C.

Base de datos clave/valor más popular.

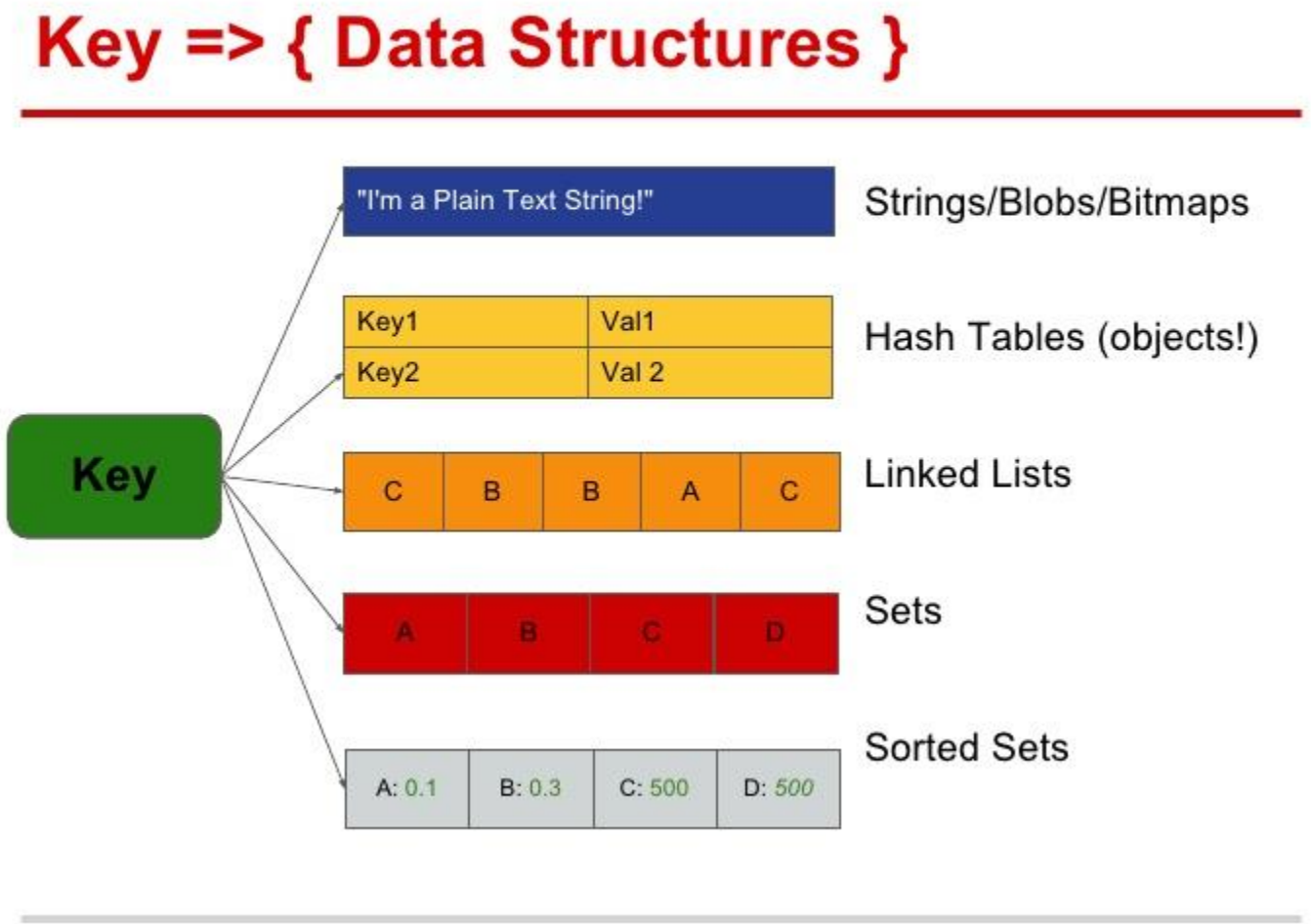
Admite distintos tipos de estructuras de datos.

Adecuada para una amplia variedad de problemas que pueden mapearse naturalmente.

REDIS – ESTRUCTURAS DE DATOS

Estructura	Que contiene
STRING	Cadenas, enteros y valores de coma flotante
LISTS	Colección de cadenas ordenadas según el orden de inserción.
SET	Colección desordenada de cadenas únicas.
HASH	Tabla hash desordenada de key/value.
ZSET	Similar a los SET pero cada elemento está asociado a un valor de número flotante, llamado puntuación.

REDIS – ESTRUCTURAS DE DATOS



REDIS – ESTRUCTURAS DE DATOS

Estructura	¿Que son?
STRINGs en Redis	Los **STRING** son similares a los que vemos en otros lenguajes. Las operaciones posibles son: **GET**, **SET**, **DEL, etc.**

REDIS – ESTRUCTURAS DE DATOS

Estructura	¿Que son?
LISTs en Redis	Colección de cadenas. Las operaciones que se pueden realizar son las típicas que existen en casi cualquier lenguaje de programación. **LPUSH / RPUSH. LPOP / RPOP, etc.**

REDIS – ESTRUCTURAS DE DATOS

Estructura	¿Que son?
SETs en Redis	Se trata de una colección de *Strings* sin un orden determinado de almacenamiento. No permite valores duplicados para una misma Clave. Útiles para efectuar operaciones entre *sets,* tales como uniones, intersecciones y diferencias.

REDIS – ESTRUCTURAS DE DATOS

Estructura	¿Que son?
Hashes en Redis	Usado para representar objetos. Almacena conjuntos de pares Clave-Valor de cadena, asociados a una misma clave. No existe un límite de campos dentro de una *Hash.* Esta estructura de hash se puede pensar como versiones en miniatura del propio redis. Algunas de las operaciones son: **HSET, HGET, HGETALL, HDEL.**

REDIS – ESTRUCTURAS DE DATOS

Estructura	¿Que son?
SORTED SETs en Redis (ZSETs)	Similar a los *Sets*, con la variante que cada elemento de un grupo de datos cuenta con una calificación *(Score)* asociado. Este *Score* determinará el orden.

REDIS – ALGUNAS VENTAJAS

DESEMPEÑO RÁPIDO

Los datos se encuentran en la memoria principal del servidor. Al eliminar la necesidad de acceder a discos, las bases de datos en memoria como Redis evitan los retrasos y pueden acceder a los datos con algoritmos más sencillos que utilizan menos instrucciones de la CPU. Las operaciones típicas tardan menos de un milisegundo en ejecutarse.

ESTRUCTURAS EN MEMORIA

Redis admite listas de cadenas en el orden en el que se van agregando, conjuntos de cadenas sin ordenar, conjuntos clasificados ordenados por puntuación, hashes que almacenan una lista de campos y valores, etc. Con Redis, se puede almacenar en la memoria prácticamente cualquier tipo de dato.

REDIS – ALGUNAS VENTAJAS

REPLICACIÓN Y PERSISTENCIA

Redis utiliza una arquitectura maestro-esclavo. Los datos se replican en numerosos servidores esclavos. De este modo, se logra una mejora en el desempeño de lectura y de recuperación cuando el servidor principal sufre un fallo.

COMPATIBILIDAD

Los desarrolladores que utilicen Redis tienen a su disposición un gran conjunto de lenguajes con soporte Redis. Entre los lenguajes admitidos se encuentran Java, Python, PHP, C, C++, C#, JavaScript, Node.js, Ruby y muchos otros.

REDIS – ALGUNAS DESVENTAJAS

PERSISTENCIA

El método de persistencia RDB consume mucho I/O (escritura en disco).

ESPACIO EN MEMORIA

Todos los datos trabajados deben encajar en la memoria (en caso de no usar persistencia física).

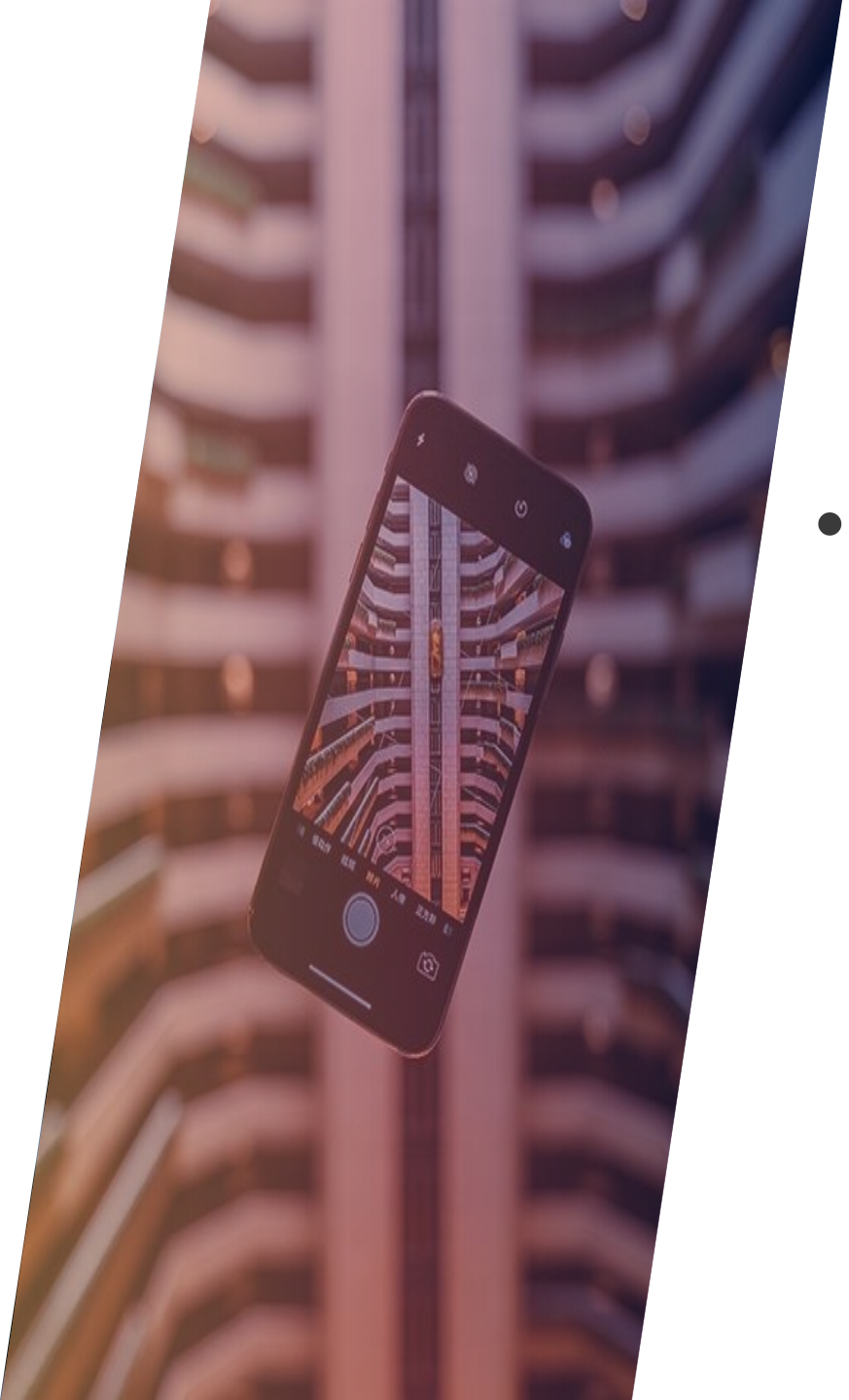
REDIS – PROBLEMAS CLÁSICOS

- Sistemas de chat y mensajería.
- Listado de elementos más recientes.
- Contadores y uso de estadísticas en tiempo real.

- Manejo y administración de carros de compra en línea.
- Almacenamiento de sesiones de usuario dentro de una aplicación.
- Soporte de caché de páginas web.

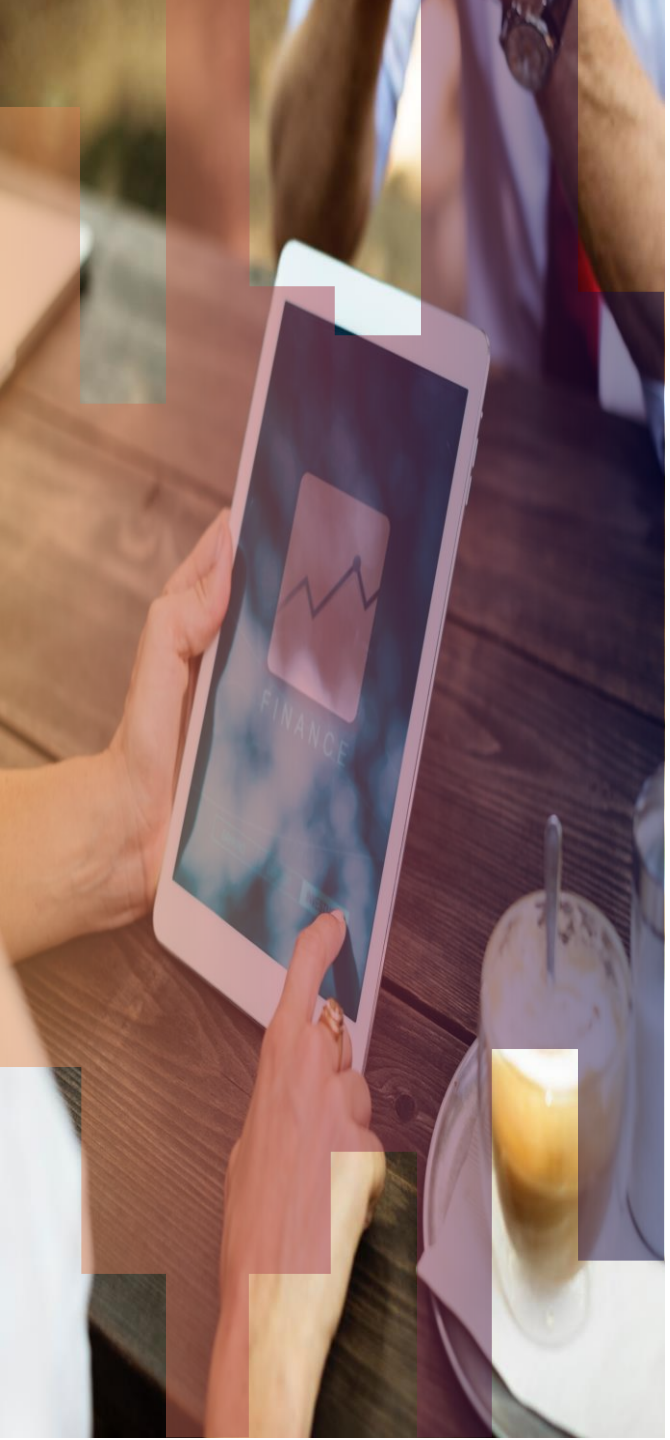
REDIS – INSTALACIÓN EN LINUX

- **sudo apt-get update**
- **sudo apt-get install redis-server php-redis**
- **Ejecutar el ciente: redis-cli**
- **Debería aparecer algo así:**
  - **127.0.0.1:6379>**
  - **127.0.0.1:6379>ping**
  - **127.0.0.1:6379> PONG**



REDIS – INSTALACIÓN EN LINUX

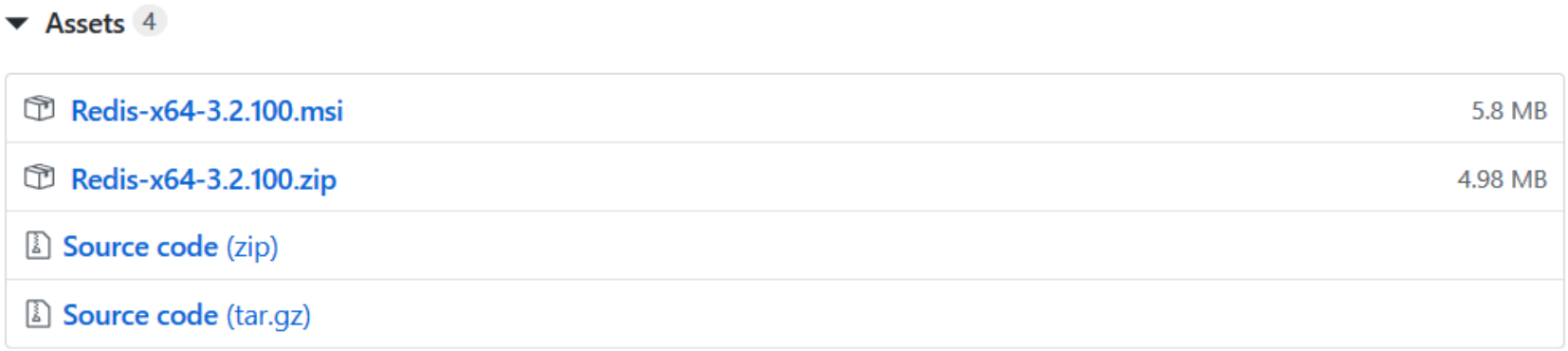
- Servidor:
  - /etc/init.d/redis-server **start**
  - /etc/init.d/redis-server **stop**
  - /etc/init.d/redis-server **restart**



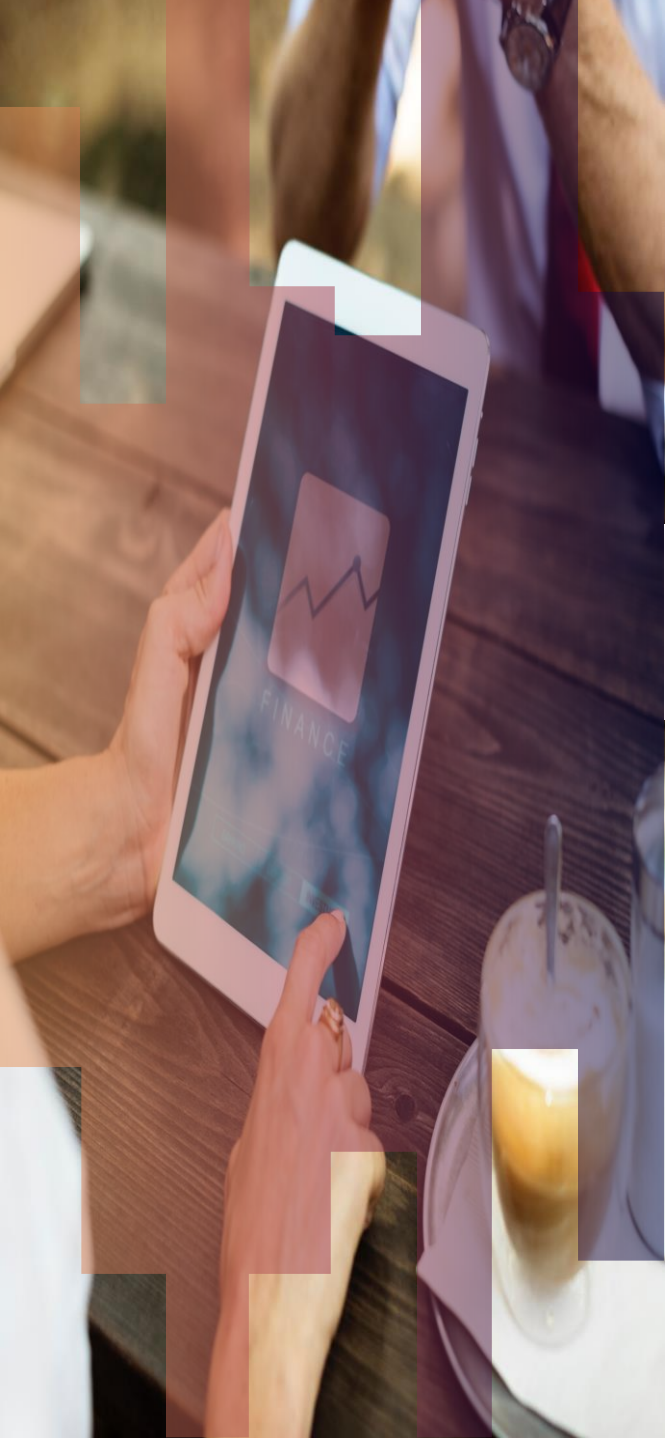
REDIS – INSTALACIÓN EN WINDOWS

github.com/MSOpenTech/redis/releases

- Descargue el archivo .msi o .zip, este tutorial le permitirá descargar el último archivo zip.



- Extraiga el archivo zip en un directorio.
- Ejecutar **redis-server.exe**.
- Ejecutar **redis-cli.exe**, después de ejecutar con éxito el servidor de redis.
- El comando **PING** se usa para probar si una conexión está aún activa (la respuesta espera es **PONG**).

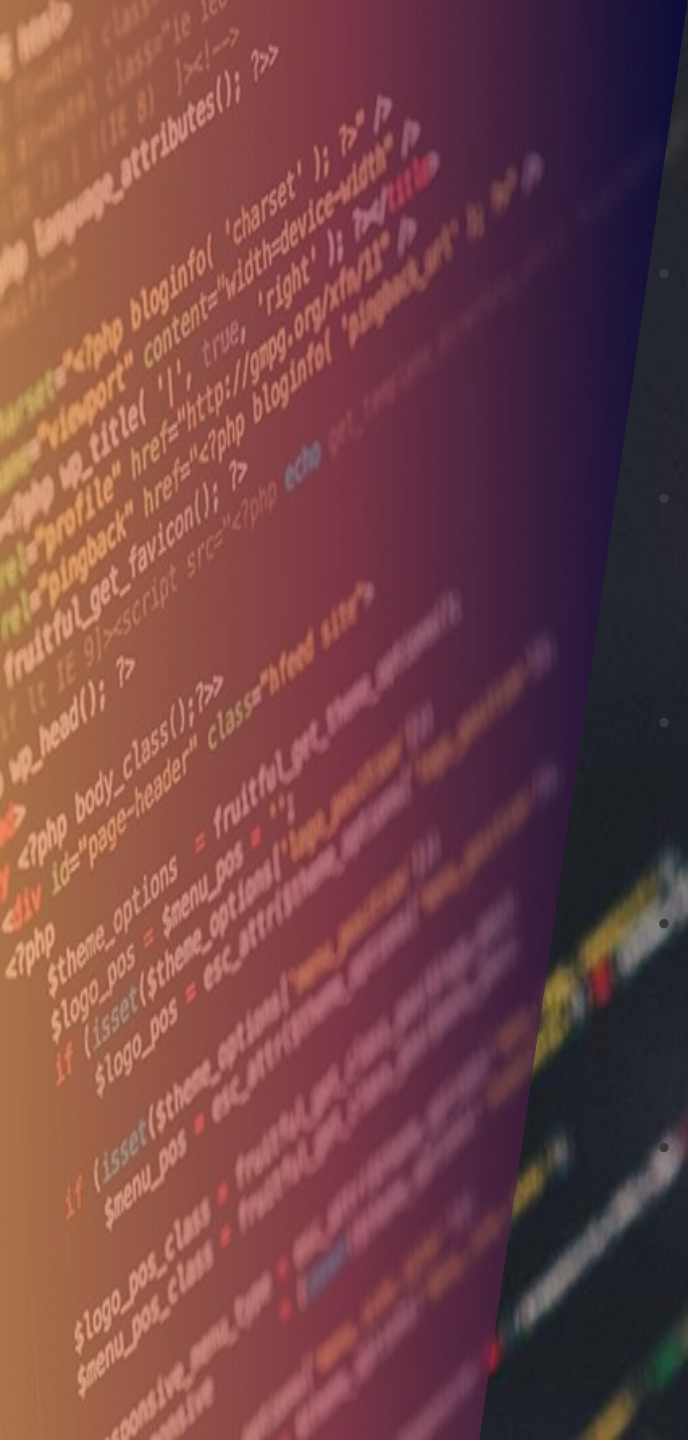


REDIS – EN DOCKER

https://redis.io/docs/latest/operate/oss_and_stack/install/install-stack/docker/

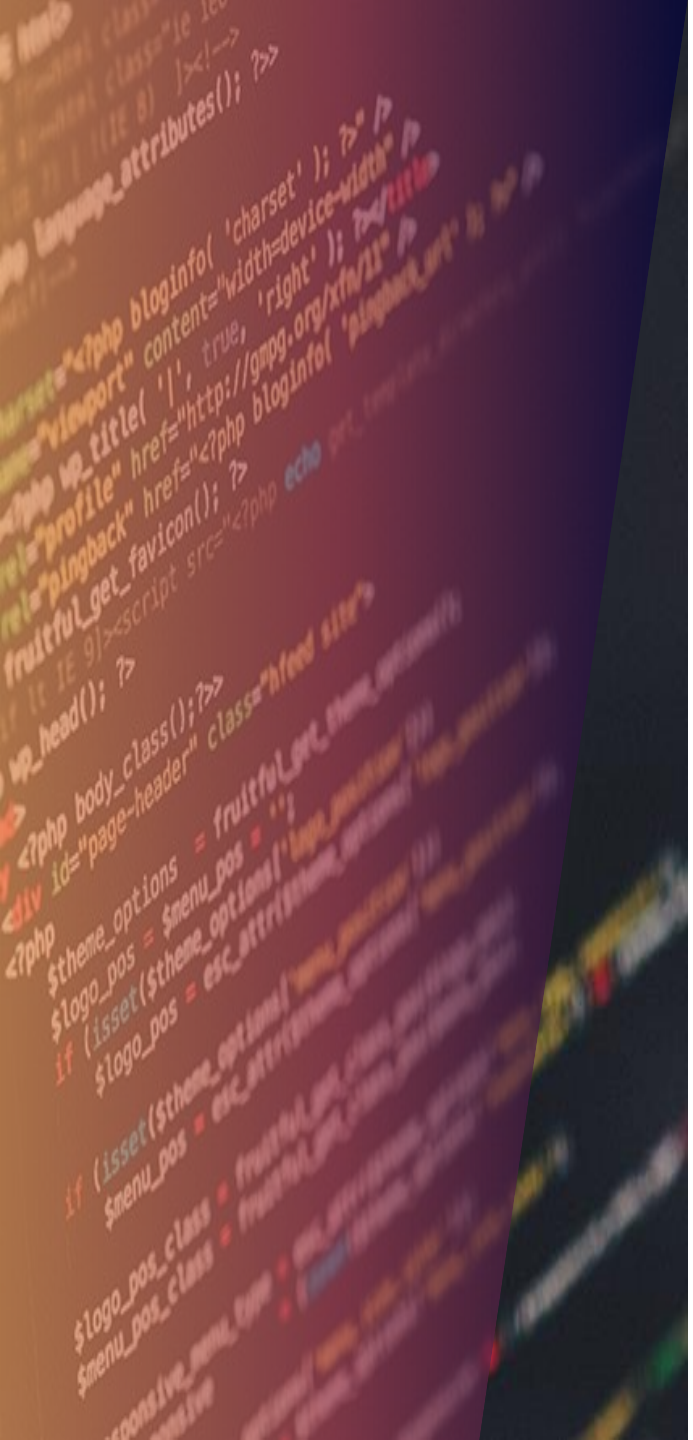
```
docker run -d --name redis -p 6379:6379 redis:<version>
```

```
# redis-cli
127.0.0.1:6379> ping
PONG
127.0.0.1:6379>
```



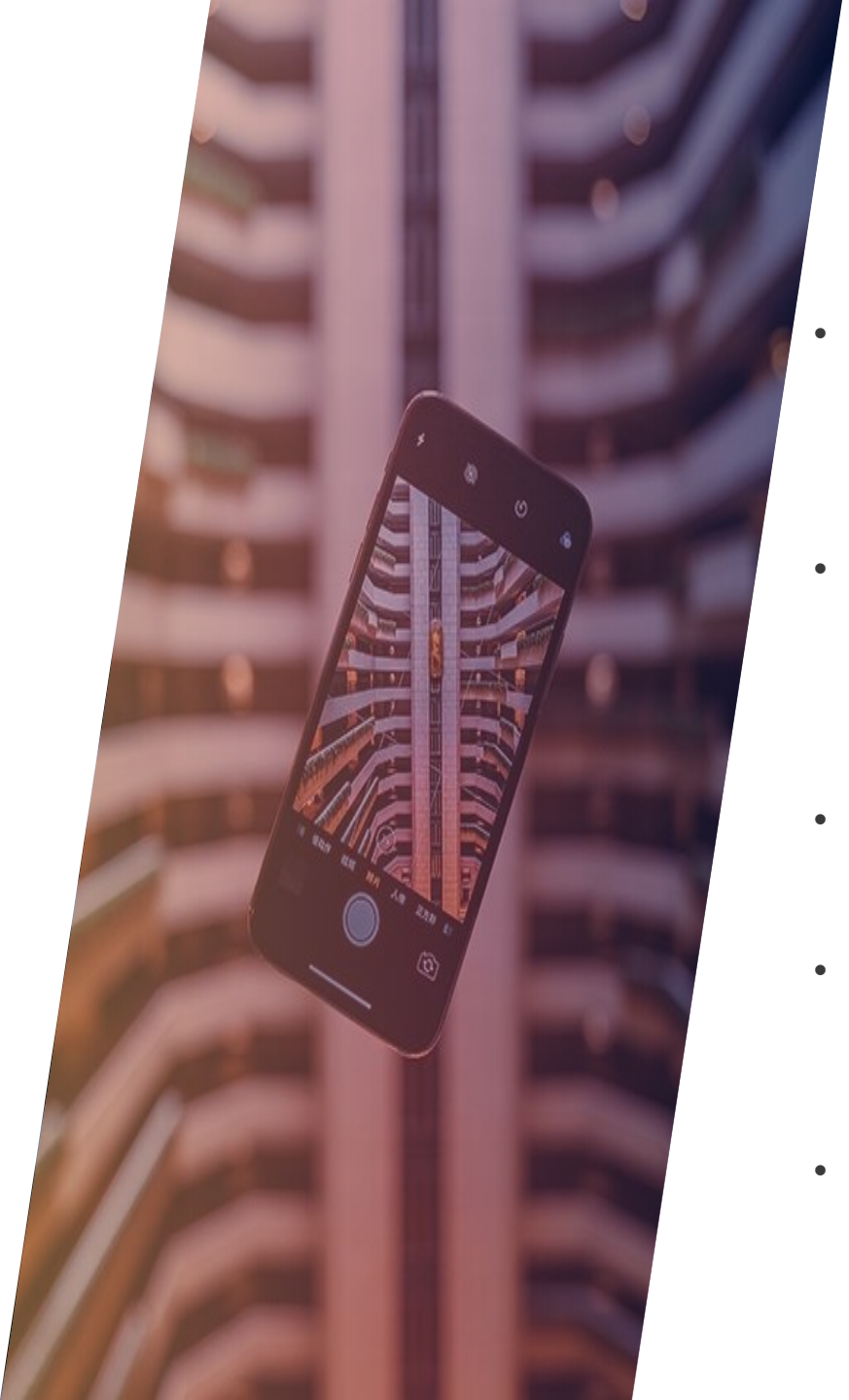
REDIS – ALGUNOS COMANDOS

- **SET: establece el valor para una clave.**
  - **SET una_clave "un valor".**
- **GET: recupera el valor para una clave.**
  - **GET una_clave.**
- **DEL: elimina una clave.**
  - **DEL una_clave.**
- **KEYS pattern: retorna todas las claves que coinciden con el patrón dado.**
  - **Por ejemplo, KEYS ***
- **INCR / DECR: incrementa / decrementar el valor almacenado en la clave.**
  - **INCR una_clave.**
  - **DECR una_clave.**



REDIS – ALGUNOS COMANDOS

- **MSET: asigna valor para múltiples claves.**
  - `MSET una_clave "un valor" [otra_clave "otro valor" ...]`
- **MGET: recupera el valor para todas las claves.**
  - `MGET una_clave [otra_clave ...]`
- **EXISTS: retorna si existe la clave.**
  - `EXISTS una_clave [otra_clave ...]`
- **EXPIRE: establece un tiempo en segundos para la clave.**
  - `EXPIRE una_clave un_tiempo`
  - Por ejemplo, `SET una_clave EX un_valor`
- **TTL: tiempo de vida que le queda a la clave antes de expirar.**
  - `TTL una_clave.`
  - `-2` indica que la clave ha expirado.
  - `-1` indica que la clave existe sin tiempo de expiración.



REDIS – LISTAS

- **RPUSH: agrega valores a la lista.**
  - `RPUSH mi_lista un_valor [otro_valor ...]`
  - Si la clave no existe, la crea.
  - Si la clave contiene un valor que no es una lista entonces retorna un error.

- **LRANGE: retorna los elementos de la lista según el rango especificado.**
  - `LRANGE mi_lista valor_inicial valor_final`
  - `LRANGE mi_lita 0 -1 (retorna toda la lista)`

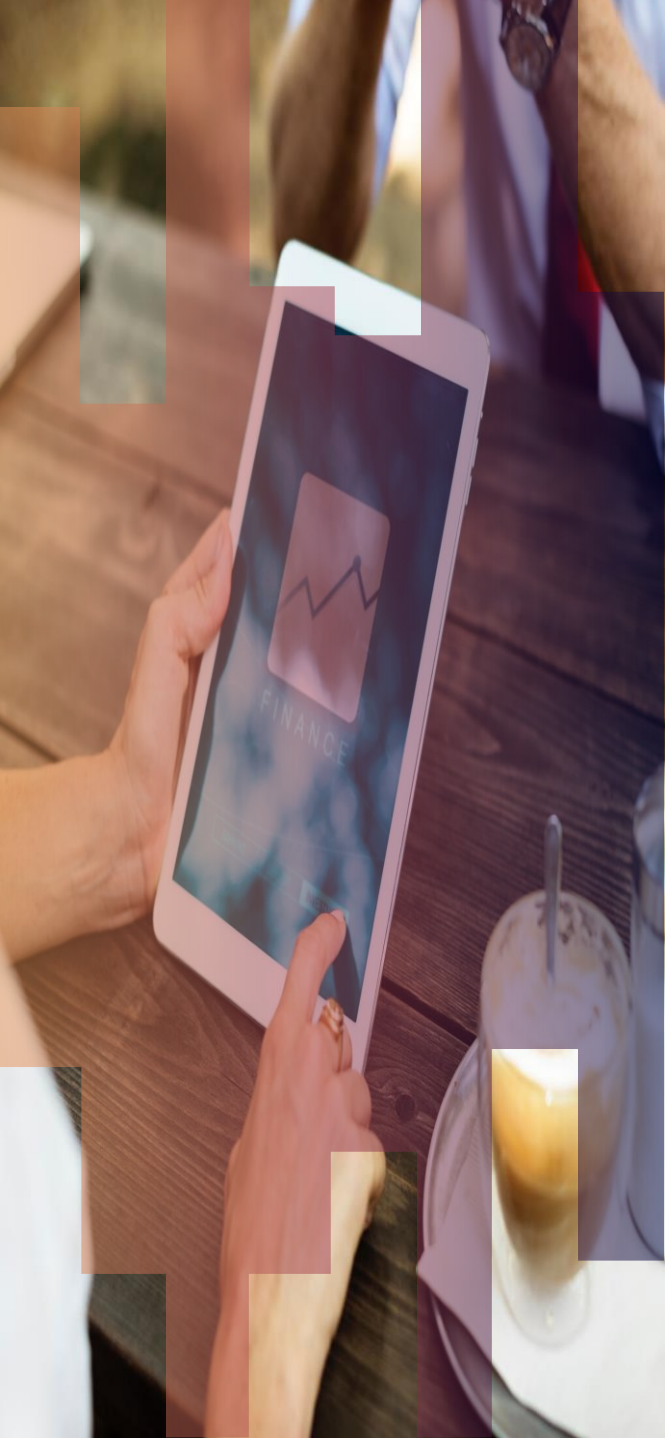
- **LPUSH: agrega al inicio valores a la lista.**
  - `LPUSH mi_lista un_valor [otro_valor ...]`

- **RPOP: elimina y retorna el último elemento de la lista.**
  - `RPOP mi_lista`

- **LPOP: elimina y retorna el primer elemento de la lista.**
  - `LPOP mi_lista`

REDIS – LISTAS

- **LTRIM: recorta la lista en un rango dado.**
  - `LTRIM mi_lista un_valor_inicial un_valor_final`
- **LLEN: retorna la longitud de la lista.**
  - `LLEN mi_lista`
- **BRPOP: en un RPOP bloqueante. Queda bloqueado cuando intenta sacar un elemento y la lista está vacía.**
  - `BRPOP mi_lista un_tiempo`
- **BLPOP: ídem a BRPOP.**
- **EXISTS.**



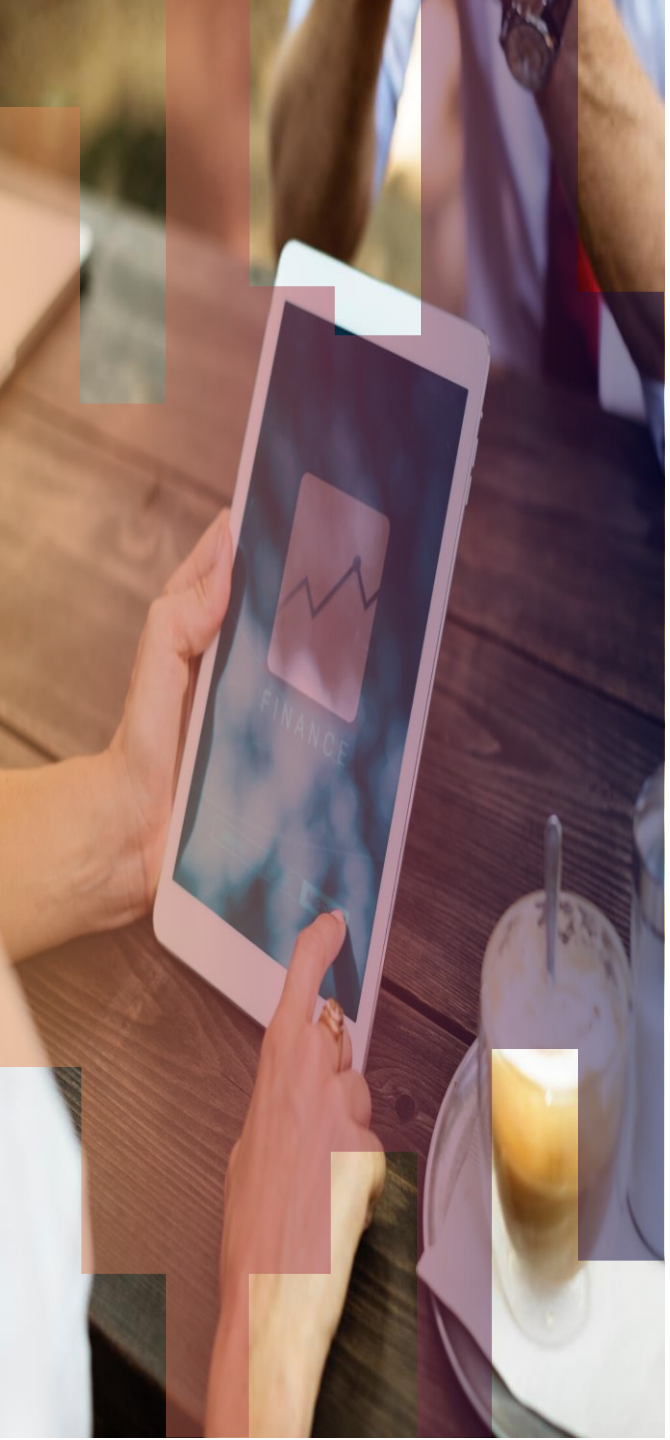
REDIS – HASHES

- Los hashes son la evolución de las listas. Permiten almacenar objetos con estructuras más complejas que simples cadenas.

- HMSET: establece el valor para los campos especificados en el hash almacenado en una clave.
  - HMSET una_clave un_campo un_valor [otro_campo otro_valor …]

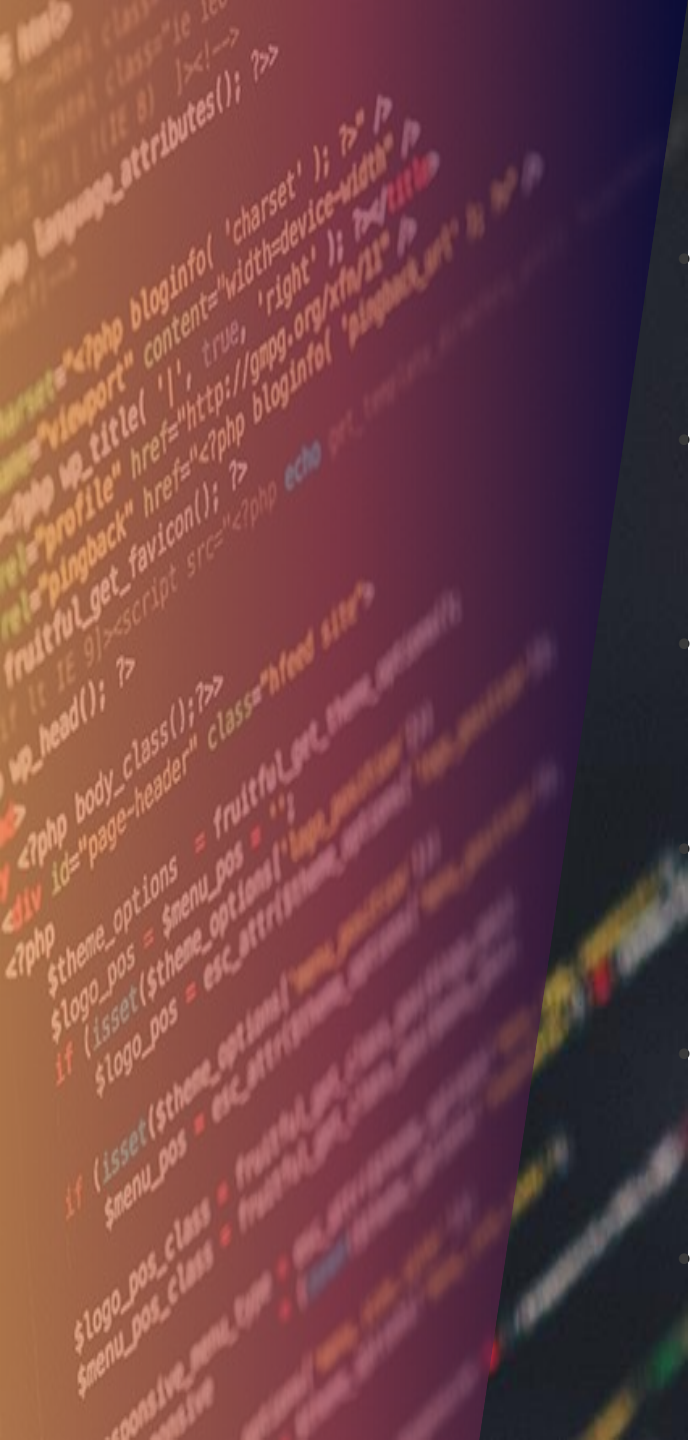
- HMGET: retorna el valor de los campos especificados.
  - HMGET una_clave un_campo [otro_campo …]

- HGETALL: retorna el valor de todos los campos.
  - HGETALL una_clave



REDIS – HASHES

- **HINCRBY: incrementa el valor almacenado en el campo especificado.**
  - **HINCRBY una_clave uncampo un_incremento.**
- **HKEYS: retorna los campos del hash.**
  - **HKEYS una_clave.**
- **HLEN: retorna la cantidad de campos que contiene el hash.**
  - **HLEN una_clave.**
- **HDEL: elimina del hash los campos especificados.**
  - **HDEL una_clave un_campo [otro_campo …]**



REDIS – SETS

- Redis es autónomos en el orden que almacena los elementos en un conjunto.

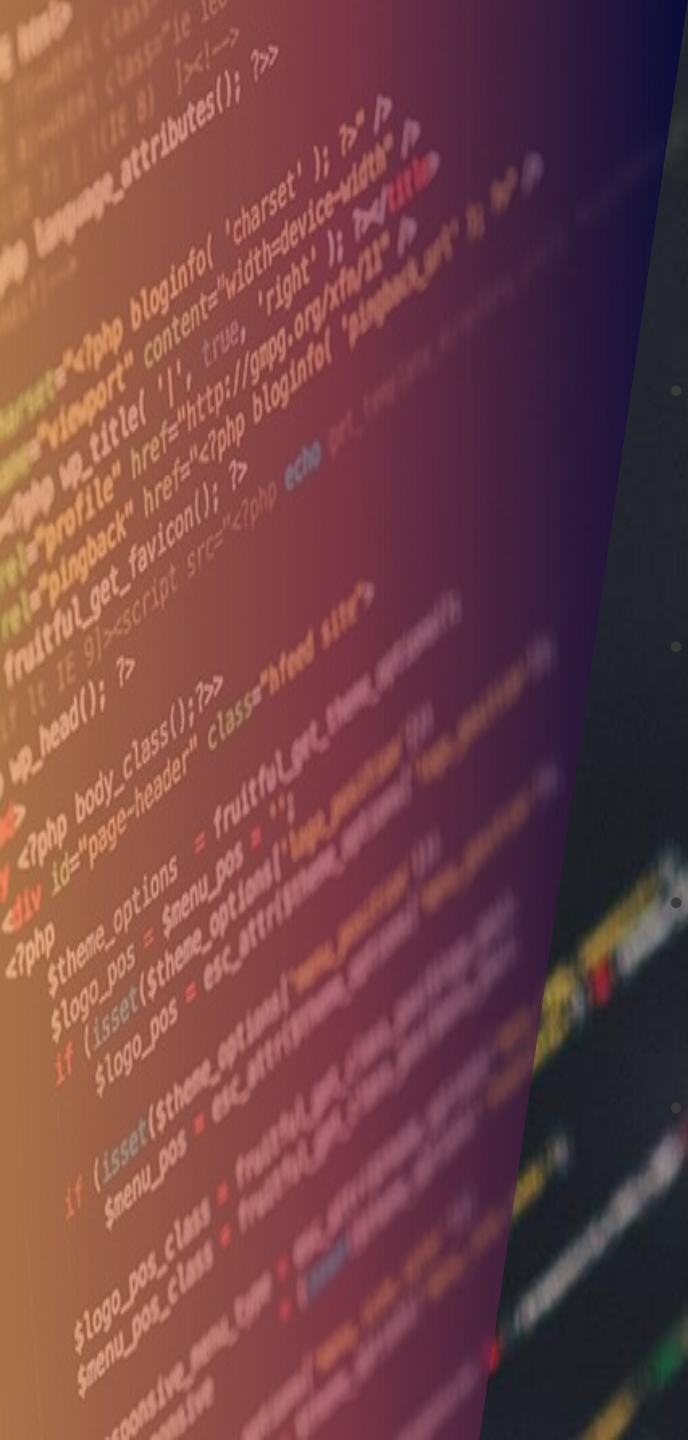
- SADD: agrega elementos al conjunto.
  - SADD una_clave un_valor [otro_valor …]

- SMEMBERS: retorna los elementos del conjunto.
  - SMEMBERS una_clave

- SREM: elimina un elemento del conjunto.
  - SREM una_clave un_elemento

- SINTER: intersección de conjuntos.
  - SINTER una_clave [otra_clave …]

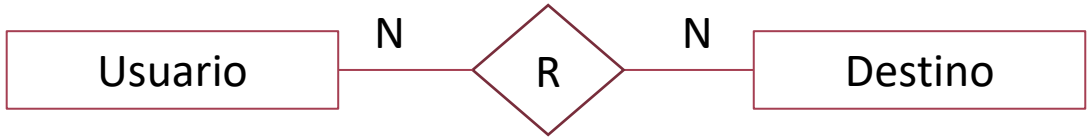
- SUNION: unión de conjuntos.
  - SUNION una_clave [otra_clave …]



REDIS – SETS

- **SINTERSTORE: similar a SINTER pero almacena el resultado en el conjunto indicado.**
  - **SINTERSTORE una_clave_set otra_clave_set [otra_clave_set …]**
- **SUNIONSTORE: similar a SUNION pero almacena el resultado en el conjunto indicado.**
  - **SUNION una_clave_set otra_clave_set [otra_clave_set …]**
- **SPOP: elimina y devuelve elementos aleatorios del conjunto.**
  - **SPOP una_clave_set [una_cantidad]**
- **SDIFF: retorna la diferencia de conjuntos.**
  - **SDIFF una_clave_set [otra_clave_set]**
  - **SDIFFSTORE una_clave_set otra_clave_set [otra_clave_set …]**

REDIS – SETS - EJEMPLO

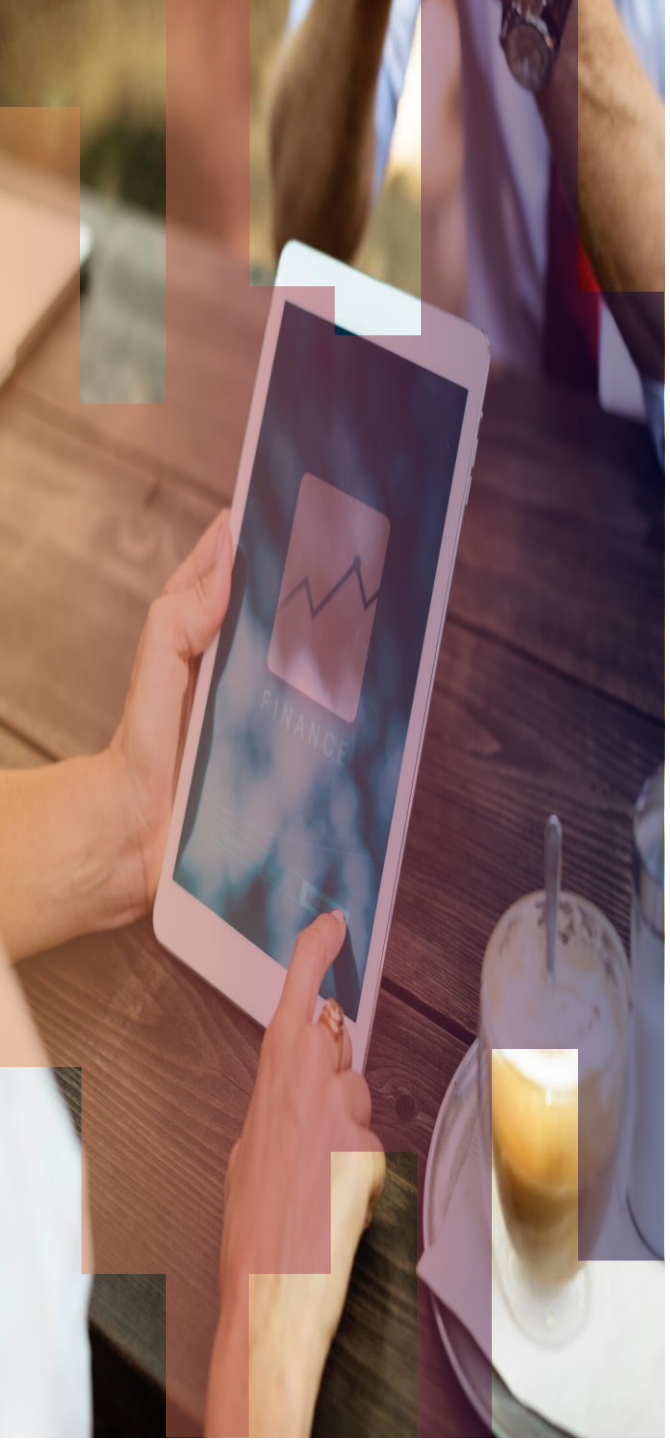


- **Un subsistema de usuarios relacionado con destinos turísticos.**

  - **sadd** usr:1:dest uruguay brasil colombia chile
  - **sadd** usr:2:dest costa_rica brasil peru
  - **sadd** usr:3:dest italia portugal francia holanda
  - **sadd** usr:4:dest mexico brasil argentina chile

  - **sadd** dest:ury:usr 1
  - **sadd** dest:br:usr 1 2 4
  - **sadd** dest:col:usr 1
  - **sadd** dest:chi:usr 1 4

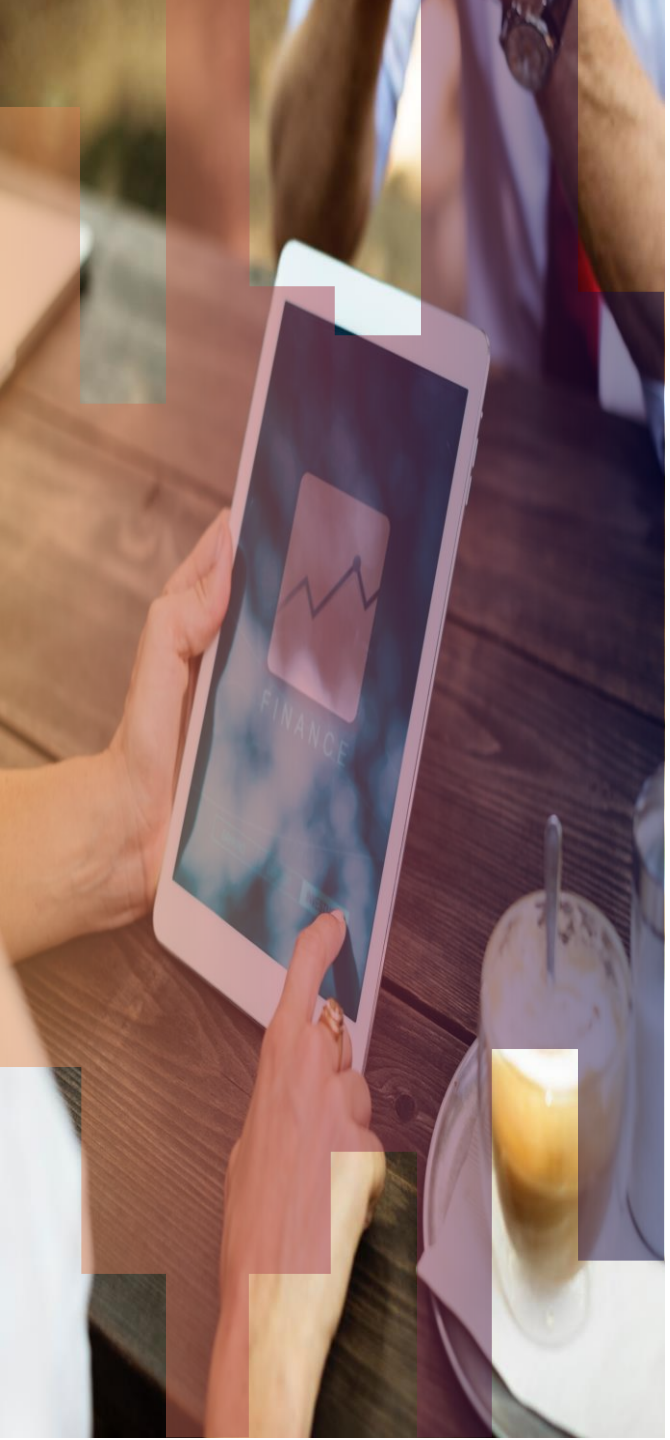
  - **sinter** dest:br:usr dest:chi:usr



REDIS – ZSETS

- **ZADD: agrega los elementos con sus puntajes al conjunto ordenado.**
  - **zadd una_clave [opciones] un_puntaje un_elemento [otro_puntaje otro_elemento …]**

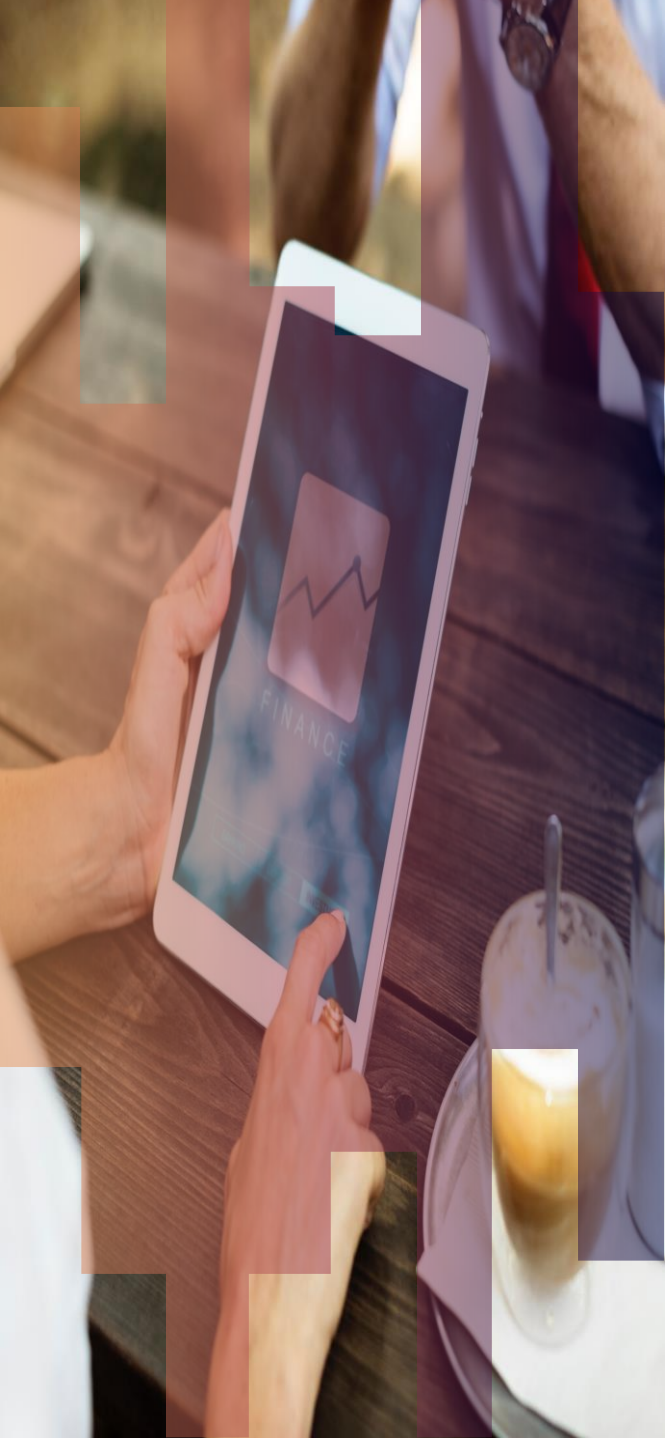
- **ZADD posee una lista de opciones que se pueden especificar después de la clave y antes de los valores de puntuación.**
  - **XX: solo actualizar elementos existentes. No agrega elementos.**
  - **NX: no se actualizan elementos existentes. Solo agrega elementos.**
  - **CH: retorna solamente el número de elementos cambiados y no agregados.**



REDIS – ZSETS

- **ZRANGE**: retorna los elementos según el rango especificado.
  - `zrange una_clave un_rango [WITHSCORES]`

- **ZRANGEBYSCORE**: retorna los elementos según el puntaje especificado.
  - `zrangebyscore una_clave un_rango [WITHSCORES]`
  - *Ejemplos*:
    - `zrangebyscore una_clave -inf +inf`
    - `zrangebyscore una_clave un_valor otro_valor`
    - `zrangebyscore una_clave (un_valor otro_valor`
    - `zrangebyscore una_clave (un_valor (otro_valor`

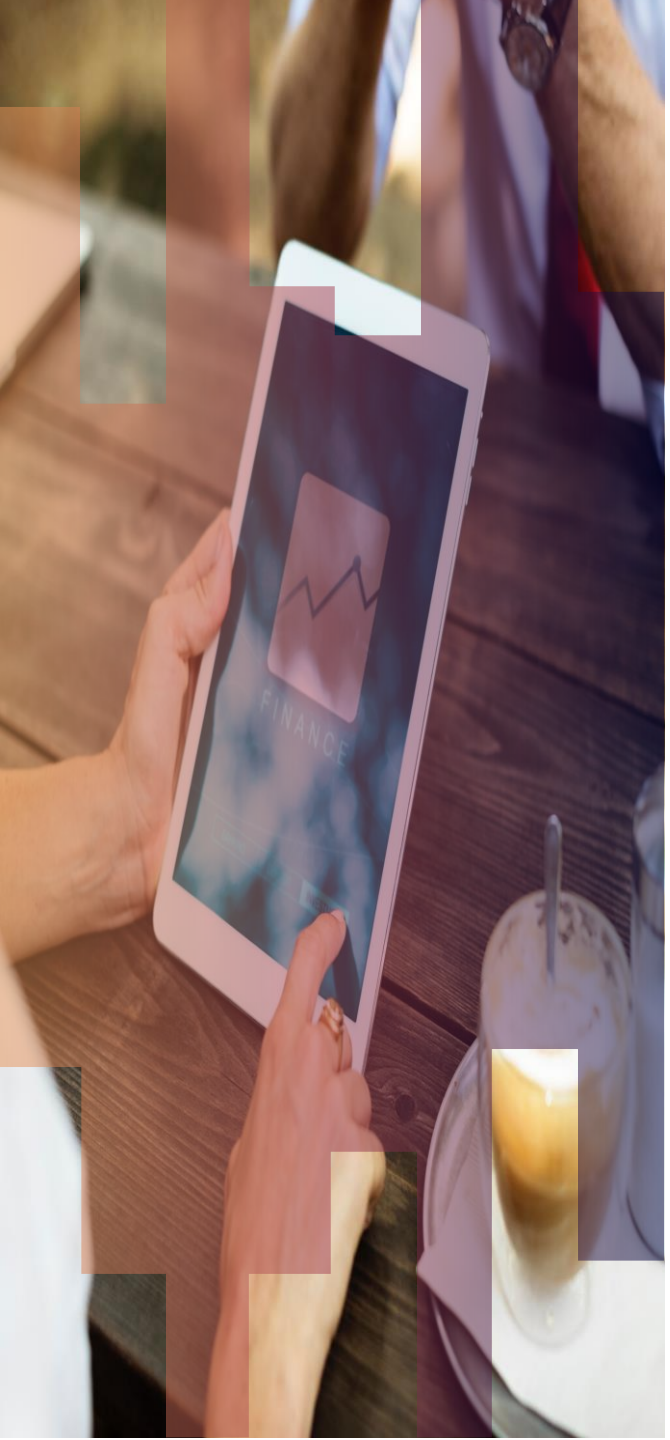


REDIS – ZSETS

- **ZRANK**: retorna la posición en la que se encuentra un elemento según su puntaje de menor a mayor.
  - `zrank una_clave un_elemento`

- **ZREVRANK**: retorna la posición en la que se encuentra un elemento según su puntaje de mayor a menor.
  - `zrevrank una_clave un_elemento`

- **ZCOUNT**: retorna la cantidad de elementos con el puntaje en el rango especificado.
  - `zcount una_clave un_puntaje otro_puntaje`



REDIS – ZSETS

- **ZINCRBY:** incrementa el puntaje de un elemento. Si el elemento no existe lo agrega con el incremento indicado.
  - **zincrby** una_clave un_incremento un_elemento
- **ZREM:** elimina un elemento del conjunto.
  - **zrem** una_clave un_elemento [otro_elemento]
- **ZREMRANGEBYSCOR:** elimina los elementos que posean un puntaje en el rango especificado.
  - **zremragnebyscor** una_clave un_elemento [otro_elemento]

REDIS – PERSISTENCIA

Redis permite la persistencia de su información en disco.

Mecanismo RDB (Redis Database Backup) (snapshooting): hace un snapshots del conjunto de datos en intervalos de tiempo.

Mecanismo AOF(Append Only File): escribe cada operación de escritura recibida por el servidor. En cada inicio del servidor se puede reconstruir el conjunto de datos original.

REDIS – PERSISTENCIA

RDB – Ventajas:

Estrategia adecuada para realizar *backups* de datos.

Adecuado para recuperar datos antes un desastre.

Maximiza el rendimiento de Redis.

Permite una recuperación de los datos más rápida que AOF.

RDB – Inconvenientes:

No es buena para minimizar la pérdida de datos en caso de desastre.

Si el conjunto de datos es grande hay demasiadas operaciones de E/S.

REDIS – PERSISTENCIA

AOF – Ventajas:

Se abre en modo *append*. No hay búsquedas y muy bajo riesgo de corrupción.

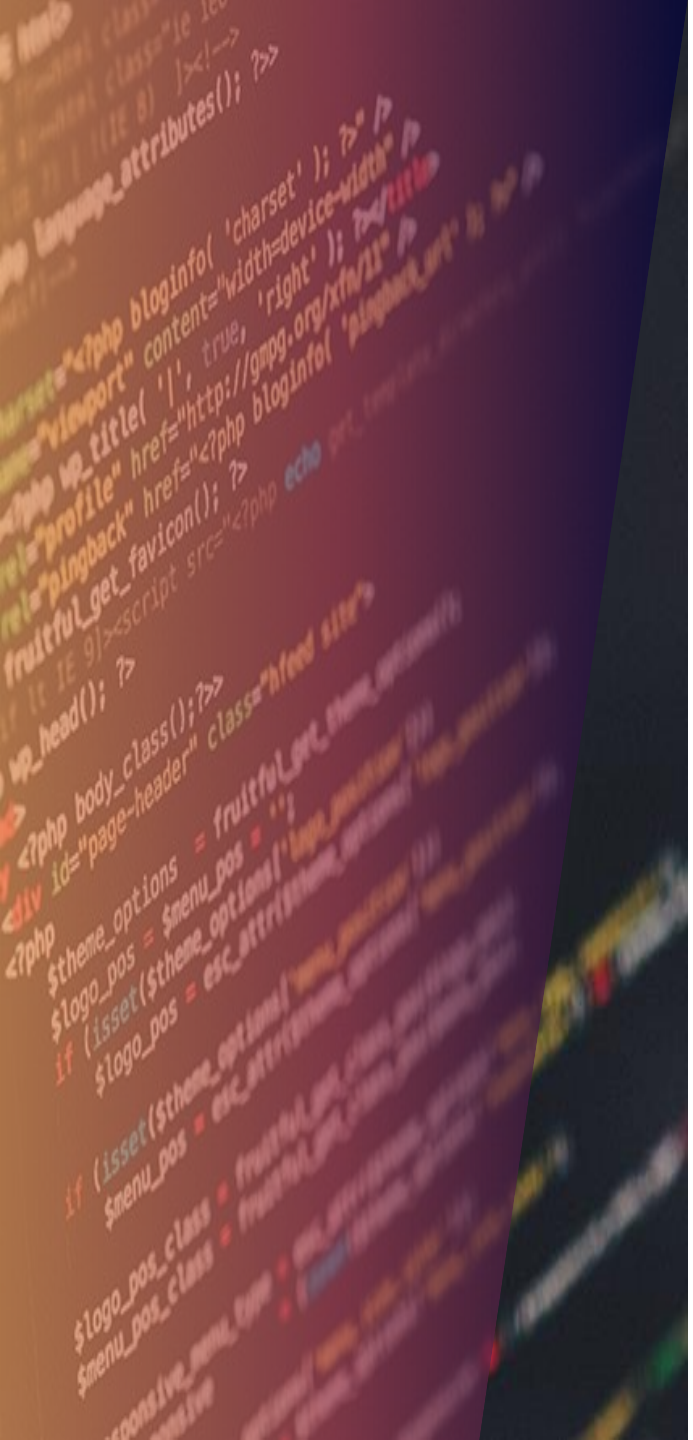
Cuando el archivo se hace demasiado grande se crean nuevos archivos.

Guarda todas las operaciones realizadas en el servidor.

AOF – Inconvenientes:

Los archivos pueden ser grandes.

Puede ser más lento que RDB.

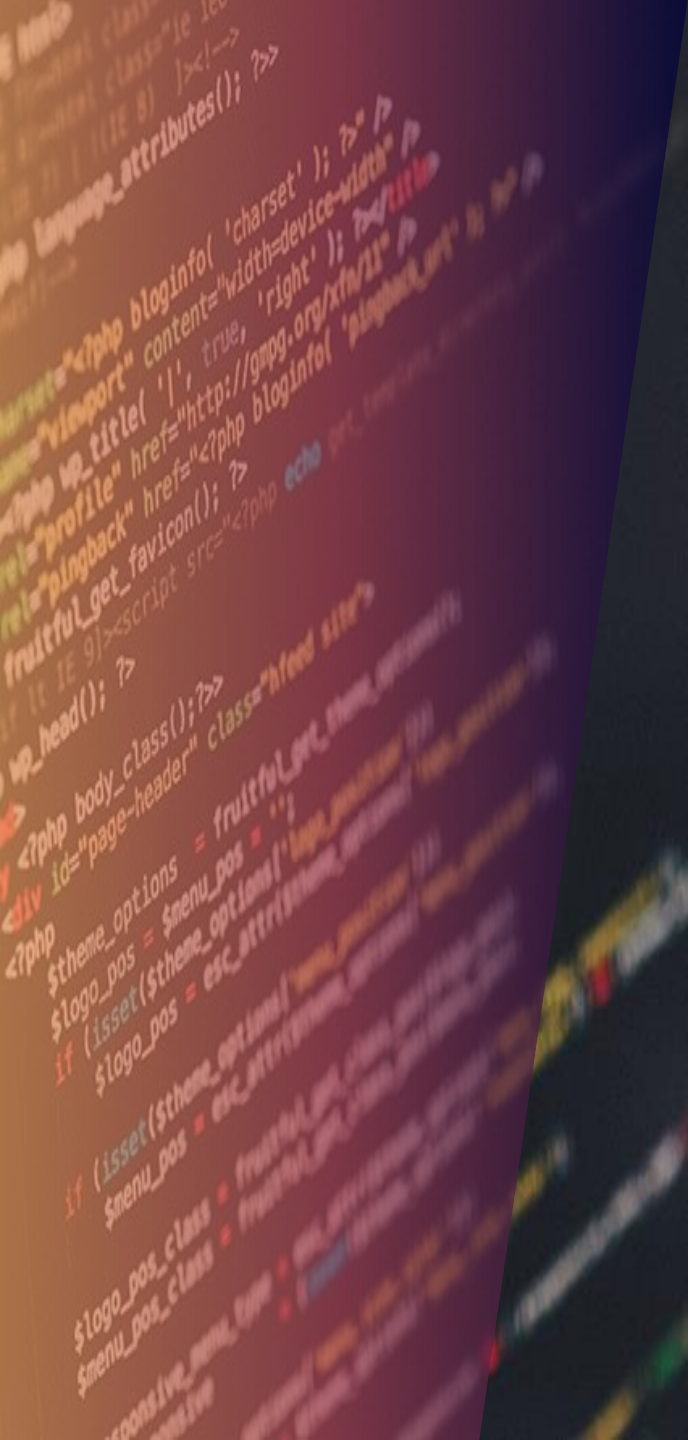


REDIS – PERSISTENCIA

De forma predeterminada Redis guarda instantáneas del conjunto de datos en un archivo binario llamado “**dump.rdb**”.

En Ubuntu: `/var/lib/redis`

- **SAVE:** el comando SAVE realiza un guardado síncrono del conjunto de datos. Produce un archivo RDB.
  - Se puede configurar Redis para que guarde el conjunto de datos cada N segundos si hay al menos M cambios.
    - ***SAVE [segundos] [cambios]***
    - ***SAVE** N M*



REDIS – PERSISTENCIA

- Se puede activar el modo **AOF** desde el archivo de configuración “**redis.conf**”.
  - `/etc/redis/`
  - **`appendonly yes`**
  - `El nombre por defecto del archivo es “appendonly.aof”`
- Una vez activado, cada vez que Redis reciba un comando que cambie el conjunto de datos (por ejemplo, un *SET*), lo agregará al **AOF**.

GRACIAS

Marrero Luciano