**Engineer**

# Redis

Almacén en memoria para datos de corta duración

## Qué es

Redis es un servidor de base de datos que mantiene todo su conjunto de datos en memoria como estructuras con tipos definidos y permite que muchos procesos de aplicación las lean y modifiquen por la red, mediante comandos breves de uno en uno. El proyecto se describe como un servidor de estructuras de datos. La edición disponible gratuitamente se llama Redis Open Source, y Redis Ltd también vende Redis Cloud y Redis Software con licencias comerciales.

Cada clave contiene un valor de un tipo. Las cadenas son bytes sin más y también sirven como contadores; los hashes son registros de campos y valores; las listas conservan el orden de llegada de los elementos; los conjuntos contienen miembros únicos; y los conjuntos ordenados clasifican a sus miembros por una puntuación. La lista de tipos continúa con flujos, documentos JSON, índices geoespaciales, series temporales, conjuntos de vectores y estructuras probabilísticas como los filtros de Bloom y HyperLogLog.

Los comandos actúan directamente sobre esos tipos. `INCR` suma uno a un contador y devuelve el nuevo valor, `SADD` introduce miembros en un conjunto y `SPOP` extrae uno, y cualquier clave puede tener un plazo de caducidad tras el cual Redis la elimina por sí solo. Redis llama volátil a una clave así. Cuando varios pasos deben suceder como uno solo, un script Lua se ejecuta dentro del servidor, y Redis garantiza que no se ejecuta nada más mientras lo hace.

La memoria es donde viven los datos, y el disco es opcional. Redis puede escribir instantáneas de un momento dado, conocidas como archivos RDB, registrar cada escritura en un archivo de solo adición (AOF) que reproduce al arrancar, hacer ambas cosas o ninguna. Las réplicas copian un primario y pueden atender lecturas. Un techo de memoria, `maxmemory`, y una política de expulsión deciden qué ocurre cuando los datos superan el espacio que se les asignó.

La licencia cambió con Redis 8. Desde esa versión, Redis Open Source se ofrece con tres licencias a elegir: RSALv2, SSPLv1 y AGPLv3. Las dos primeras permiten acceder al código fuente, y AGPLv3 es la opción de código abierto.

Utilice el concepto en una decisión

Aplique «¿Qué es Redis?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Detectar las consultas y los índices que faltan en un sistema cada año más pesado y corregir lo que realmente importa.

**Explore nuestros servicios** Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. 30 días de datos eléctricos en 180 ms — 180 ms para agregar un mes de lecturas de contadores

**30 días de datos eléctricos en 180 ms** Póngase en contacto →

## Por qué importa

Una aplicación web rara vez se ejecuta como un solo proceso. Detrás de un equilibrador de carga hay varias instancias, cada una con su propia memoria, y cualquier cosa conservada en ella —un recuento de intentos de acceso, una sesión o una página en caché— solo existe en el proceso que atendió esa petición. Un límite contado así puede eludirse llegando a otra instancia. Redis ofrece a todas las instancias un lugar compartido para ese tipo de estado, lo bastante rápido para consultarlo en cada petición.

Los comandos atómicos importan tanto como la velocidad. Dos instancias que leen cada una un contador, suman uno y vuelven a escribirlo pueden escribir ambas el mismo número, y un incremento desaparece. `INCR` hace la lectura y la escritura dentro del servidor en un solo paso, de modo que cien peticiones concurrentes dejan el contador exactamente cien unidades más alto. Un script Lua amplía la misma garantía a una secuencia breve: una comprobación y un cambio que no deben separarse, como confirmar que aún queda sitio y ocuparlo, se ejecutan como uno solo.

La caducidad es la tercera pieza. Una ventana de limitación de solicitudes, una sesión o un código de un solo uso deberían desaparecer al agotarse su tiempo, y con un plazo de caducidad en la clave la limpieza corresponde al almacén en vez de a una tarea programada.

El precio es la durabilidad, y conviene conocerlo en cifras antes de elegir. Con el archivo de solo adición y su política predeterminada de un fsync por segundo, un fallo puede costar aproximadamente un segundo de escrituras. Solo con

instantáneas, que suelen ejecutarse cada pocos minutos, la documentación indica que «debería estar preparado para perder los últimos minutos de datos». La replicación es asíncrona de forma predeterminada, de modo que un primario puede confirmar una escritura y detenerse antes de que la haya recibido ninguna réplica. Por eso Redis resulta adecuado para un estado que puede reconstruirse o perderse sin consecuencias y poco adecuado para la única copia de un pedido, un pago o un saldo. La pregunta de diseño más importante es qué hace la aplicación cuando Redis es inaccesible o ha olvidado algo que se le comunicó.

## Cómo se aplica en la práctica

22 limitadores de solicitudes mantienen su estado en Redis en un directorio público de empresas y profesionales, para que un límite se mantenga en todas las instancias en vez de vivir en un proceso, donde sería trivial eludirlo. Cada limitador se construye para la ruta que protege, porque un intento de acceso, una búsqueda y una exportación masiva se convierten en abuso a tasas muy distintas. La elección decisiva es qué ocurre cuando falta Redis: los limitadores cierran el paso ante fallos y se rechaza el tráfico en vez de dejarlo pasar. Esa elección tiene un coste visible. Un problema de Redis se convierte en un problema que ven los usuarios, aceptado a propósito porque la alternativa es que un problema de Redis se convierta en uno de facturación.

La venta de existencias limitadas sitúa Redis delante de una base de datos como un control de admisión rápido: un contador Redis decrementado por un script Lua deja pasar aproximadamente a tantos compradores como unidades hay y rechaza a todos los demás en microsegundos. El script convierte la comprobación y el decremento en un solo paso. Ese control promete menos de lo que puede parecer. Redis replica de forma asíncrona, por lo que, tras una conmutación por error, el nuevo primario puede haber olvidado algunas admisiones y dejar pasar a unos pocos de más. Un `UPDATE` condicional en la base de datos sigue decidiendo cada venta, y un proceso de conciliación restablece el control a partir del número de la base de datos. Cuando las unidades son pocas y distintas, como asientos numerados, sus identificadores pueden estar en un conjunto Redis, y `SPOP` los entrega de uno en uno sin un contador por el que competir.

El papel más habitual es una simple caché. Una infraestructura Azure definida en Bicep construye cada entorno a partir de las mismas plantillas, tanto de pruebas como de preproducción y producción, y cada uno tiene Redis para caché junto a un PostgreSQL Flexible Server, las Container Apps que ejecutan la API y el sitio

web, y Front Door con una política WAF. Como la caché se declara en las plantillas junto con todo lo demás, cambiarla es una solicitud de cambios como cualquier otra modificación de un entorno.

## Dónde surgen los problemas

La mayoría de los problemas de Redis proceden de tratar un almacén rápido y olvidadizo como uno durable, o de suponer que recuerda lo que se le dijo sobre el tiempo.

**Una clave que debería caducar permanece para siempre.** `SET` sustituye el valor y descarta cualquier plazo de caducidad que tuviera la clave, de modo que el código que actualiza un valor en caché con un `SET` sin más convierte una entrada de diez minutos en una permanente:

```
> SET session:42 "v1" EX 600
OK
> SET session:42 "v2"
OK
> TTL session:42
(integer) -1
```

Indique el plazo de caducidad en cada escritura que lo necesite, o añada `KEEPTTL`, disponible desde 6.0.0, para conservar el anterior. La misma fuga se esconde en un contador construido con `INCR` seguido de un `EXPIRE` separado: cuando el segundo comando no llega, la clave nunca caduca. Envíe ambos dentro de un bloque `MULTI` y `EXEC`, o en un script Lua.

**Las escrituras empiezan a devolver errores mientras las lecturas siguen funcionando.** Los datos han alcanzado `maxmemory`, y la política de expulsión es `noeviction` o una política `volatile-` sobre un conjunto de datos donde ninguna clave tiene caducidad, que se comporta igual. Elija la política según el trabajo: una política `allkeys-` para una caché pura, donde puede desaparecer cualquier clave, y una `volatile-` solo cuando las claves en caché comparten instancia con otras que deben permanecer. Para esa mezcla, la documentación aconseja plantearse dos instancias separadas.

**Tras un reinicio o una conmutación por error faltan escrituras que Redis había confirmado.** Un fallo se produjo dentro de la ventana de fsync, o se promovió una réplica antes de que le llegara la escritura. Así está diseñado Redis, por lo que la solución depende de qué se guarda en él. Almacene cualquier dato que deba sobrevivir en una base de datos con commits durables, sitúe Redis delante como control de admisión o caché y concilie ambos desde el lado de la base de datos, como hace la nota sobre existencias.

**Todos los clientes se detienen en el mismo momento.** Se está ejecutando un script Lua largo, y Redis bloquea toda la demás actividad del servidor durante toda su ejecución. Mantenga los scripts breves y acotados y deje fuera de ellos los bucles sobre colecciones grandes.

**Un limitador deja pasar el tráfico durante una interrupción.** El código cliente interpreta un error de Redis como «límite no alcanzado», de modo que el limitador se abre precisamente cuando su almacén es inaccesible. Decida el modo de fallo deliberadamente y pruébelo deteniendo Redis: un limitador que proteja una ruta costosa debería rechazar, como hacen los 22 anteriores, con el coste de que un problema de Redis se haga visible para los usuarios.

## Lecturas adicionales

- Desarrollar con Redis — El punto de partida del proyecto: comandos para probar, bibliotecas cliente por lenguaje y tipos de datos.
- Persistencia de Redis — Instantáneas, archivo de solo adición, políticas de fsync y qué puede perder cada una.
- Replicación de Redis — Cómo siguen las réplicas a un primario y por qué sigue existiendo una ventana de pérdida de datos.
- Expulsión de claves — `maxmemory`, todas las políticas de expulsión y consejos para elegir una.
- EXPIRE — Plazos de caducidad de las claves y qué comandos los borran o los conservan.
- Scripts con Lua — Ejecución atómica de un script en el servidor y qué bloquea mientras tanto.

Glosario

Elija según sus restricciones reales

Traiga un conjunto de datos representativo, las preguntas que debe responder y el cuello de botella actual. Podemos acordar cómo medir la calidad, la capacidad de procesamiento y la recuperación.

**Vea cuánto cuesta** Póngase en contacto →

# Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es Redis?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

**Explore nuestros servicios**

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

**Creado con esmero y cariño por D.**

**CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N**

**Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.**

**Impreso desde https://engineer.company/es/glossary/redis/**