Distributed Cache Service

Descripción general del servicio

Edición	01
Fecha	2022-11-07

HUAWEI TECHNOLOGIES CO., LTD.

Marcas y permisos


y otras marcas registradas de Huawei pertenecen a Huawei Technologies Co., Ltd.
Todas las demás marcas registradas y los otros nombres comerciales mencionados en este documento son propiedad de sus respectivos titulares.

Aviso

Las funciones, los productos y los servicios adquiridos están estipulados en el contrato celebrado entre Huawei y el cliente. Es posible que la totalidad o parte de los productos, las funciones y los servicios descritos en el presente documento no se encuentren dentro del alcance de compra o de uso. A menos que el contrato especifique lo contrario, ninguna de las afirmaciones, informaciones ni recomendaciones contenidas en este documento constituye garantía alguna, ni expresa ni implícita.

La información contenida en este documento se encuentra sujeta a cambios sin previo aviso. En la preparación de este documento se realizaron todos los esfuerzos para garantizar la precisión de sus contenidos. Sin embargo, ninguna declaración, información ni recomendación contenida en el presente constituye garantía alguna, ni expresa ni implícita.

Índice

1 ¿Qué es DCS?

Distributed Cache Service (DCS) de Huawei Cloud es un servicio de caché en memoria en línea, distribuido y compatible con Redis y Memcached. Es fiable, escalable, fácil de usar y fácil de gestionar. Cumple con sus requisitos de alto rendimiento de lectura/escritura y acceso rápido a los datos.

- Preparado y fácil de usar

  DCS proporciona las instancias del nodo único, de los nodos principal/en standby, del clúster de Proxy, del clúster de Redis y de la separación de lectura/escritura con las especificaciones que varían de 128 MB a 1024 GB. Las instancias de DCS se pueden crear con pocos clics en la consola, sin la necesidad de preparar los servidores.

  Las instancias de DCS Redis 4.0, 6.0, y 5.0 están en contenedores y se pueden crear en cuestión de segundos.

- Seguridad y confiabilidad

  El almacenamiento y el acceso a los datos de instancias están protegidos de forma segura a través de los servicios de gestión de seguridad de Huawei Cloud, incluidos Identity and Access Management (IAM), Virtual Private Cloud (VPC), Cloud Eye, and Cloud Trace Service (CTS).

  Las instancias principal/en standby y de clúster se pueden implementar dentro de una zona de disponibilidad (AZ) o a través de AZ.

- Escalamiento automático

  Las instancias DCS se pueden escalar hacia arriba o hacia abajo en línea, lo que le ayuda a controlar los costos en función de los requisitos de servicio.

- Fácil gestión

  Se proporciona una consola basada en web para que realice varias operaciones, como reiniciar instancias, modificar parámetros de configuración y realizar copias de seguridad y restaurar datos. También se proporcionan interfaces de programación de aplicaciones (API) RESTful para la gestión automática de instancias.

- Migración en línea

  Puede crear una tarea de migración de datos en la consola para importar los archivos de copia de seguridad o migrar los datos en línea.

Para obtener más información sobre cómo seleccionar un motor de caché, consulte **Comparación entre Redis y Memcached**.

DCS for Redis

DCS for Redis soporta Redis 3.0, 4.0, 5.0 y 6.0.

- Huawei Cloud DCS for Redis 3.0, 4.0 y 5.0

  NOTA

  DCS for Redis 3.0 ya no se proporciona. Puede utilizar DCS for Redis 4.0 o 5.0 en su lugar.

  Redis es un sistema de almacenamiento que admite varios tipos de estructuras de datos, incluidos los pares clave-valor. Se puede utilizar en los escenarios tales como el almacenamiento en caché de datos, la publicación/suscripción de eventos y la cola de alta velocidad, como se describe en **Escenarios de la aplicación**. Redis se escribe en ANSI C, soporta lectura/escritura directa de **strings**, **hashes**, **lists**, **sets**, **sorted sets**, y **streams**. Redis trabaja con un conjunto de datos en memoria que puede persistir en el disco.

  Las instancias de DCS Redis se pueden personalizar en función de sus requisitos.

Tabla 1-1 Configuraciones de instancia de DCS Redis

Tipo de instancia	DCS for Redis proporciona los siguientes tipos de instancias para adaptarse a diferentes escenarios de servicio: Nodo único: Adecuado para almacenar en caché datos temporales en escenarios de baja confiabilidad. Las instancias de nodo único admiten operaciones de lectura/escritura altamente simultáneas, pero no admiten persistencia de datos. Los datos se eliminarán después de reiniciar las instancias. Principal/En standby: Cada instancia principal/en standby se ejecuta en dos nodos (un principal y un en standby). El nodo en standby replica datos de forma síncrona desde el nodo principal. Si el nodo principal falla, el nodo en standby se convierte automáticamente en el nodo principal. Puede dividir las operaciones de lectura y escritura escribiendo en el nodo principal y leyendo desde el nodo en standby. Esto mejora el rendimiento general de lectura/escritura de caché. Clúster de proxy: Además del clúster de Redis nativo, una instancia de clúster de proxy tiene los proxy y balanceadores de carga. Los balanceadores de carga implementan el equilibrio de carga. Se distribuyen diferentes solicitudes a diferentes proxy para lograr una alta simultaneidad. Cada partición del clúster tiene un nodo principal y un nodo en standby. Si el nodo principal es defectuoso, el nodo en standby en la misma partición es promovida a la función del principal para asumir los servicios. Clúster de Redis: Cada instancia de clúster de Redis consta de varias **particiones** y cada partición incluye un nodo principal y varias réplicas (o ninguna réplica). Las partición son invisibles a usted. Si el nodo principal falla, una réplica en la misma partición se hace cargo de los servicios. Puede dividir las operaciones de lectura y escritura escribiendo en el nodo principal y leyendo desde las réplicas. Esto mejora el rendimiento general de lectura/escritura de caché. Separación de lectura/escritura: Una instancia de separación de lectura/escritura tiene proxy y balanceadores de carga además de la arquitectura principal/en standby. Los balanceadores de carga implementan el equilibrio de carga, y diferentes solicitudes se distribuyen a diferentes proxy. Los proxy distinguen entre las solicitudes de lectura y escritura, y las envía a nodos principales o nodos en standby, respectivamente.
Especificaciones de las instancias	DCS for Redis proporciona las instancias de diferentes especificaciones, que van desde 128 MB a 1024 GB.
Compatibilidad con software de código abierto:	Las instancias de DCS son compatibles con Redis 3.0, 4.0, 5.0 y 6.0 de código abierto.

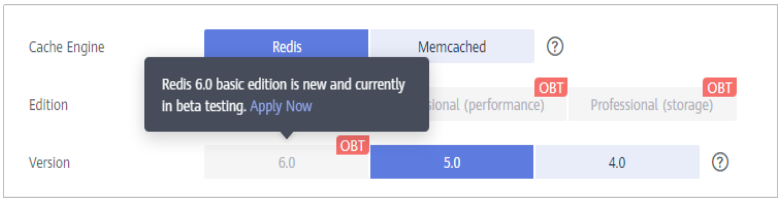
Arquitectura subyacente	Redis **estándar** basado en las máquinas virtuales: admite hasta 100,000 consultas por segundo (QPS) en un nodo único.
Alta disponibilidad (HA) y recuperación ante desastres (DR)	Todas las instancias, excepto las de nodo único, se pueden implementar en AZ dentro de una región con fuentes de energía y redes físicamente aisladas.

Para obtener más información acerca de Redis de código abierto, visite **https://redis.io/**.

- Huawei Cloud DCS for Redis 6.0 (OBT)

  NOTA

  Redis 6.0 está en OBT. Puede ir a la página para crear una instancia en la consola DCS y hacer clic en **Apply Now** para solicitar el permiso OBT, como se muestra en la siguiente figura.

  Huawei Cloud DCS for Redis ahora cuenta con alto rendimiento basado en subprocesos múltiples.

  El alto rendimiento basado en subprocesos múltiples se logra basándose en KeyDB de código abierto, que es una bifurcación Redis que ofrece un alto rendimiento. KeyDB se centra en el multiproceso, la eficiencia de la memoria y el alto rendimiento, y proporciona características que solo se encuentran en Redis Enterprise. KeyDB es totalmente compatible con el protocolo, módulos y scripts de Redis, y garantiza la atómica de scripts y transacciones. El desarrollo de KeyDB sigue el ritmo de Redis, por lo que KeyDB proporciona un superconjunto de funcionalidades de Redis y puede reemplazar las implementaciones de Redis existentes. Dado el mismo hardware, KeyDB maneja el doble de consultas por segundo que Redis con una latencia un 60% menor. Active-replication simplifica la migración por falla en standby inmediata, lo que le permite asignar fácilmente operaciones de escritura en réplicas y utilizar un simple equilibrio de carga basado en TCP o migración por falla. El alto rendimiento de KeyDB le permite lograr más con menos hardware, reduciendo costos de operación y complejidad.

  En KeyDB, los bucles de E/S y event loops se ejecutan en varios subprocesos. KeyDB soporta características tales como expiraciones de subkey, múltiples maestros, hashes anidados y CRON for Lua scripts, que no están disponibles en Redis.

  En versiones anteriores a Redis 6.0, una consulta lenta a menudo causa que otras consultas se retrasen debido al modelo de subproceso único. Para abordar los problemas de rendimiento, la nueva edición ha realizado una serie de optimización basada en un modelo multiproceso. Se ha mejorado la simultaneidad de subprocesos múltiples para el

procesamiento de eventos de E/S y backend; el acceso a los datos almacenados en caché se acelera aún más a través del bloqueo de giro justo; las claves caducadas se pueden eliminar dos veces más rápido gracias a algoritmos optimizados; La compatibilidad con las subclaves caduca también mejora el rendimiento de lectura/escritura de las claves grandes. Como resultado, la nueva edición es adecuada para escenarios que requieren un alto rendimiento de un solo nodo, como temas de tendencias y eventos de transmisión en vivo en Internet.

Las instancias de DCS Redis se pueden personalizar en función de sus requisitos.

Tabla 1-2 Configuraciones de instancia de DCS Redis

Tipo de instancia	DCS for Redis proporciona los siguientes tipos de instancias para adaptarse a diferentes escenarios de servicio: Nodo único: Adecuado para almacenar en caché datos temporales en escenarios de baja confiabilidad. Las instancias de nodo único admiten operaciones de lectura/escritura altamente simultáneas, pero no admiten persistencia de datos. Los datos se eliminarán después de reiniciar las instancias. Principal/En standby: Cada instancia principal/en standby se ejecuta en dos nodos (un principal y un en standby). El nodo en standby replica datos de forma síncrona desde el nodo principal. Si el nodo principal falla, el nodo en standby se convierte automáticamente en el nodo principal. Puede dividir las operaciones de lectura y escritura escribiendo en el nodo principal y leyendo desde el nodo en standby. Esto mejora el rendimiento general de lectura/escritura de caché. Clúster de proxy: Además del clúster de Redis nativo, una instancia de clúster de proxy tiene los proxy y balanceadores de carga. Los balanceadores de carga implementan el equilibrio de carga. Se distribuyen diferentes solicitudes a diferentes proxy para lograr una alta simultaneidad. Cada partición del clúster tiene un nodo principal y un nodo en standby. Si el nodo principal es defectuoso, el nodo en standby en la misma partición es promovida a la función del principal para asumir los servicios. Clúster de Redis: Cada instancia de clúster de Redis consta de varias **particiones** y cada partición incluye un nodo principal y varias réplicas (o ninguna réplica). Las partición son invisibles a usted. Si el nodo principal falla, una réplica en la misma partición se hace cargo de los servicios. Puede dividir las operaciones de lectura y escritura escribiendo en el nodo principal y leyendo desde las réplicas. Esto mejora el rendimiento general de lectura/escritura de caché. Separación de lectura/escritura: Una instancia de separación de lectura/escritura tiene proxy y balanceadores de carga además de la arquitectura principal/en standby. Los balanceadores de carga implementan el equilibrio de carga, y diferentes solicitudes se distribuyen a diferentes proxy. Los proxy distinguen entre las solicitudes de lectura y escritura, y las envía a nodos principales o nodos en standby, respectivamente.
Especificaciones de las instancias	Rango de 4 GB a 8 GB, 16 GB, 32 GB y 64 GB.

Compatibilidad con software de código abierto:	KeyDB 6.0.16
Arquitectura subyacente	Redis **estándar** basado en las máquinas virtuales: admite hasta 300,000 consultas por segundo (QPS) en un nodo único.
HA y DR	Todas las instancias, excepto las de nodo único, se pueden implementar en AZ dentro de una región con fuentes de energía y redes físicamente aisladas.

Para obtener más información acerca de KeyDB, visite **https://keydb.dev/**.

NOTA

La edición básica de DCS for Redis 6.0 proporciona las instancias principales/en espera y de un nodo único. Las ediciones Enterprise (rendimiento) y Enterprise (almacenamiento de información) solo proporcionan instancias principales/en espera.

DCS para Memcached (Próximamente no disponible)

NOTA

DCS for Memcached está a punto de no estar disponible y ya no se vende en algunas regiones. Puede utilizar DCS for Redis 4.0 o 5.0 en su lugar.

Memcached es un sistema de almacenamiento en caché de clave-valor en memoria que admite la lectura/escritura de cadenas simples. A menudo se utiliza para almacenar en caché los datos de la base de datos back-end para aliviar la carga en estas bases de datos y acelerar las aplicaciones web. Para obtener más información sobre los escenarios de su aplicación, consulte **Escenarios de aplicaciones de Memcached**.

Además de la compatibilidad total con Memcached, DCS for Memcached proporciona el modo de standby inmediata y la persistencia de datos.

Tabla 1-3 Configuración de instancia de DCS Memcached

Tipo de instancia	DCS for Memcached proporciona los dos tipos de instancias siguientes para adaptarse a diferentes escenarios de servicio: Nodo único: Adecuado para almacenar en caché datos temporales en escenarios de baja confiabilidad. Las instancias de nodo único admiten operaciones de lectura/escritura altamente simultáneas, pero no admiten persistencia de datos. Los datos se eliminarán después de reiniciar las instancias. Principal/En standby: Cada instancia principal/en standby se ejecuta en dos nodos (un principal y un en standby). El nodo en standby replica datos de forma sincrónica desde el nodo principal, pero no soporta las operaciones de lectura/escritura. Si el nodo principal falla, el nodo en standby se convierte automáticamente en el nodo principal.
Memoria	Especificación de instancias DCS Memcached de nodo único o principal/en standby: 2 GB, 4 GB, 8 GB, 16 GB, 32 GB y 64 GB.
HA y DR	Las instancias principal/en standby de DCS Memcached se pueden implementar en AZs en la misma región con fuentes de energía y redes físicamente aisladas.

Para obtener más información acerca de Memcached de código abierto, visite **https://memcached.org/**.

DCS Video de introducción

Vea el siguiente video para obtener más información sobre DCS.

Distributed Cache Service

2 Escenarios de la aplicación

Escenarios de la aplicación Redis

Muchos sitios web de comercio electrónico a gran escala y aplicaciones de transmisión de video y juegos requieren un acceso rápido a grandes cantidades de datos que tienen estructuras de datos simples y no necesitan consultas frecuentes de unión. En tales escenarios, puede usar Redis para lograr un acceso rápido pero económico a los datos. Redis le permite recuperar datos de almacenes de datos en memoria en lugar de depender completamente de bases de datos basadas en disco más lentas. Además, ya no es necesario realizar las tareas de gestión adicionales. Estas características hacen de Redis un complemento importante de las bases de datos tradicionales basadas en disco y un servicio básico esencial para las aplicaciones de Internet que reciben acceso de alta simultaneidad.

Los escenarios de aplicación típicos de DCS for Redis son los siguientes:

1. **Ventas flash de comercio electrónico**

   El catálogo de productos de comercio electrónico, las ofertas y los datos de ventas flash se pueden almacenar en caché en Redis.

   Por ejemplo, el acceso a datos de alta concurrencia en las ventas flash difícilmente puede ser manejado por las bases de datos relacionales tradicionales. Requiere que el hardware tenga una configuración más alta, como E/S de disco. Por el contrario, Redis soporta 100,000 QPS por nodo y le permite implementar el bloqueo usando comandos simples como SET, GET, DEL y RPUSH para manejar ventas flash.

   Para obtener más información sobre el bloqueo, consulte **Implementación de bloqueos distribuidos con Redis** en *Prácticas recomendadas*.

2. **Comentarios de vídeo en directo**

   En la transmisión en vivo, los datos de usuario en línea, clasificación de regalos y comentarios de viñetas se pueden almacenar como conjuntos ordenados en Redis.

   Por ejemplo, los comentarios de viñetas se pueden devolver mediante el comando **ZREVRANGEBYSCORE**. Los commandos ZPOPMAX y ZPOPMIN en Redis 5.0 pueden facilitar adicionalmente el procesamiento de mensajes.

3. **Tabla de clasificación del juego**

   En los juegos en línea, los jugadores de mayor clasificación se muestran y se actualizan en tiempo real. La clasificación de la tabla se puede almacenar como conjuntos ordenados, que son fáciles de usar con hasta 20 comandos.

   Para obtener más información, consulte **Clasificación con Redis** en *Prácticas recomendadas*.

4. **Comentarios en redes sociales**

   En las aplicaciones web, las consultas de comentarios de publicaciones a menudo implican ordenar por tiempo en el orden descendente. A medida que los comentarios se acumulan, la clasificación se vuelve menos eficiente.

   Mediante el uso de listas en Redis, se puede devolver un número preestablecido de comentarios desde la caché, en lugar de desde el disco, facilitando la carga de la base de datos y acelerando las respuestas de la aplicación.

Escenarios de aplicaciones de Memcached

Memcached es adecuado para almacenar datos simples clave-valor.

1. Páginas web

   El almacenamiento en caché de datos estáticos como páginas HTML, Cascading Style Sheets (CSS) e imágenes en instancias de DCS Memcached mejora el rendimiento de acceso de las páginas web.

2. Base de datos de frontend

   En sistemas dinámicos como redes sociales y sitios de blogs, las operaciones de escritura son mucho menos que las operaciones de lectura, como consultar a usuarios, amigos y artículos. Para facilitar la carga de la base de datos y mejorar el rendimiento, los siguientes datos se pueden almacenar en caché en Memcached:

   - Datos de acceso frecuente que no requieren actualizaciones en tiempo real y pueden caducar automáticamente

     Ejemplo: últimas listas de artículos y clasificaciones. Aunque los datos se generan constantemente, su impacto en la experiencia del usuario es limitado. Tales datos pueden almacenarse en caché durante un periodo de tiempo preestablecido y accederse desde la base de datos después de este periodo. Si los editores de páginas web desean ver la clasificación más reciente, se puede configurar una política de borrado o actualización de caché.

   - Datos de acceso frecuente que requieren actualizaciones en tiempo real

     Ejemplo: listas de amigos, listas de artículos y registros de lectura. Tales datos pueden almacenarse en caché en Memcached primero y, a continuación, actualizarse siempre que se produzcan cambios (agregar, modificar y eliminar datos).

3. Ventas flash

   Es difícil para las bases de datos tradicionales escribir una operación de colocación de pedidos durante las ventas flash en la base de datos, modificar los datos de inventario y garantizar la coherencia de las transacciones mientras se garantiza la experiencia del usuario ininterrumpida.

   Los comandos incr y decr de Memcached se pueden usar para almacenar información de inventario y completar la colocación de pedidos en memoria. Una vez que se envía un pedido, se genera un número de pedido. Entonces, el pedido puede ser pagado.

NOTA

Escenarios en los que Memcached no es adecuado:

- El tamaño de un único objeto de caché es superior a 1 MB.

  Memcached no puede almacenar en caché un objeto de más de 1 MB. En tales casos, utilice Redis.

- La clave contiene más de 250 caracteres.

  Para usar Memcached en tal escenario, puede generar un hash MD5 para la clave y almacenar en caché el hash en su lugar.

- Se requiere una alta fiabilidad de los datos.

  Memcached de código abierto no proporciona replicación, backup y migración de datos, por lo que la persistencia de datos no es compatible.

  Las instancias Principal/En standby DCS Memcached soportan persistencia de datos. Para obtener más información, póngase en contacto con el soporte técnico.

- Se requieren estructuras y procesamiento de datos complejos.

  Memcached solo admite los pares simples de clave-valor, y no admite estructuras de datos complejas como listas y conjuntos, ni operaciones complejas como la ordenación.

3 Tipos de instancia de DCS

3.1 Redis de nodo único

Hay disponibles tres versiones de Redis para las instancias de DCS Redis de nodo único: Redis 3.0, Redis 4.0 y Redis 5.0.

NOTA

- DCS for Redis 3.0 ya no se proporciona. Puede utilizar DCS for Redis 4.0 o 5.0 en su lugar.
- No puede actualizar la versión de Redis para una instancia. Por ejemplo, una instancia de DCS Redis 3.0 de un nodo único no se puede actualizar a una instancia de DCS Redis 4.0 o 5.0 de un nodo único. Si el servicio requiere las características de versiones de Redis superiores, cree una instancia de DCS Redis de una versión superior y, a continuación, migre los datos de la instancia antigua a la nueva.

Características

1. Baja sobrecarga del sistema y alto QPS

   Las instancias de nodo único no admiten sincronización de datos ni persistencia de datos, lo que reduce la sobrecarga del sistema y admite una mayor simultaneidad. El QPS de las instancias de DCS Redis de un solo nodo alcanza hasta 100,000.

2. Monitoreo de procesos y recuperación automática de fallos

   Con un mecanismo de monitoreo de HA, si una instancia de DCS de un nodo único resulta defectuosa, se inicia un nuevo proceso en 30 segundos para reanudar el aprovisionamiento del servicio.

3. Usabilidad lista para usar y sin persistencia de datos

   Las instancias de un nodo único de DCS se pueden usar de inmediato porque no implican la carga de datos. Si su servicio requiere un alto QPS, puede calentar los datos de antemano para evitar un fuerte impacto de simultaneidad en la base de datos backend.

4. Bajo costo y adecuado para desarrollo y pruebas

   Las instancias de un nodo único son un 40% más baratas que las instancias DCS principal/en standby, y son adecuadas para configurar entornos de desarrollo o pruebas.

En resumen, las instancias DCS de nodo único admiten las operaciones de lectura/escritura altamente simultáneas, pero no admiten persistencia de datos. Los datos se eliminarán después de reiniciar las instancias. Son adecuados para escenarios que no requieren persistencia de

datos, como el almacenamiento en caché front-end de la base de datos, para acelerar el acceso y facilitar la carga de concurrencia fuera del back-end. Si los datos deseados no existen en la caché, las solicitudes irán a la base de datos. Al reiniciar el servicio o la instancia de DCS, puede pregenerar los datos de caché de la base de datos de disco para aliviar la presión sobre el backend durante el inicio.

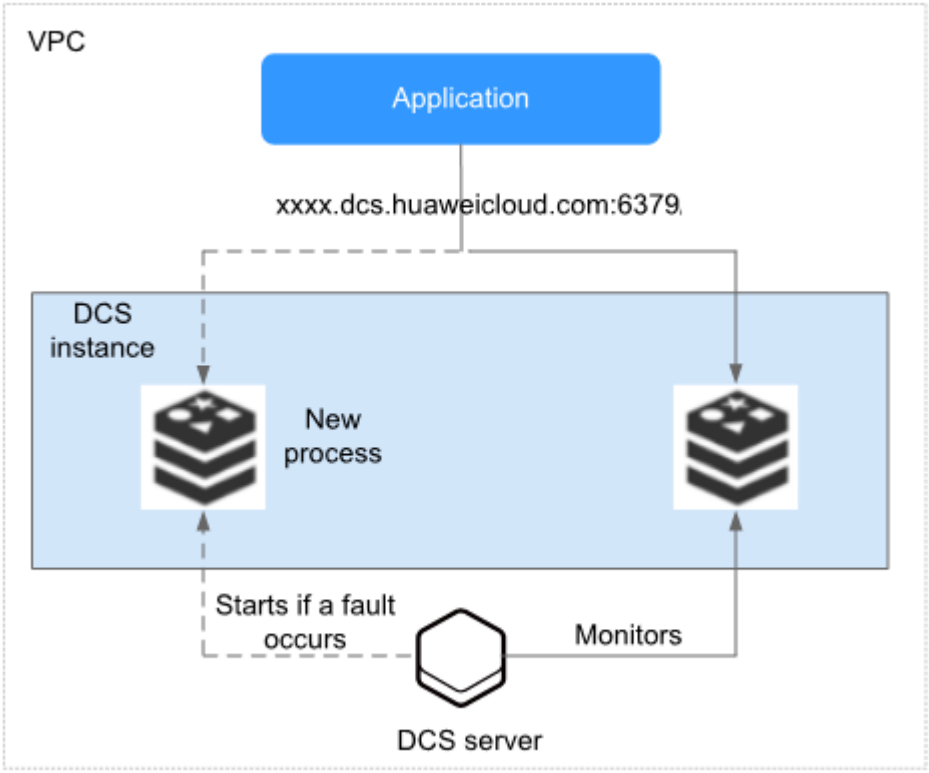
Arquitectura

Figura 3-1 muestra la arquitectura de instancias de DCS Redis de nodo único.

NOTA

Redis 3.0 no admite la personalización de puertos y solo permite el puerto 6379. Para Redis 4.0 y 5.0, puede especificar un puerto o utilizar el puerto predeterminado 6379. En la siguiente arquitectura, se utiliza el puerto 6379. Si ha personalizado un puerto, reemplace 6379 por el puerto real.

Figura 3-1 Arquitectura de instancia de DCS Redis de nodo único



Descripción de la arquitectura:

- **VPC**
  La VPC donde se ejecutan todos los nodos de la instancia.

NOTA

Para el acceso intra-VPC, el cliente y la instancia deben estar en la misma VPC con configuraciones de reglas de grupo de seguridad especificadas.

Se puede acceder a una instancia de DCS para Redis 3.0 desde una VPC o a través de redes públicas. El cliente se puede implementar fuera de la VPC y acceder a la instancia a través de elastic IP address (EIP) enlazada a la instancia. Las instancias de DCS Redis 4.0 y 5.0 no admiten el acceso público.

Para obtener más información, vea **Acceso público a una instancia de DCS para Redis** y **¿Cómo configuro un grupo de seguridad?**

- **Aplicación**

  El cliente de la instancia, que es la aplicación que se ejecuta en un Elastic Cloud Server (ECS).

  Las instancias de DCS para Redis y Memcached son respectivamente compatibles con los protocolos Redis y Memcached, y se puede acceder a través de clientes de código abierto. Para obtener ejemplos de acceso a instancias de DCS con diferentes lenguajes de programación, consulte las **instrucciones de acceso a instancias**.

- **Instancia de DCS**

  Una instancia de DCS de un nodo único, que tiene solo un nodo y un proceso de Redis.

  DCS monitorea la disponibilidad de la instancia en tiempo real. Si el proceso de Redis se vuelve defectuoso, DCS inicia un nuevo proceso en cuestión de segundos para reanudar el aprovisionamiento del servicio.

3.2 Redis principal/en standby

Tanto DCS for Redis como DCS for Memcached admiten el tipo de instancia principal/en standby. En esta sección se describen las instancias principal/en standby de DCS Redis. Las versiones de Redis disponibles para instancias principal/en standby de DCS compatibles con Redis incluyen Redis 6.0, Redis 5.0, Redis 4.0 y Redis 3.0.

La división de lectura/escritura es compatible con las instancias principal/en standby de DCS para Redis 4.0 o 5.0 de forma predeterminada, y no con las instancias principal/en standby de DCS Redis 3.0 y Redis 6.0. Para obtener más información, consulte **¿DCS for Redis admite la separación de lectura/escritura?**

NOTA

- DCS for Redis 3.0 ya no se proporciona. Puede utilizar DCS for Redis 4.0 o 5.0 en su lugar.
- No puede actualizar la versión de Redis para una instancia. Por ejemplo, una instancia principal/en standby de DCS para Redis 3.0 no se puede actualizar a una instancia principal/en standby de DCS para Redis 4.0 o 5.0. Si el servicio requiere las características de versiones de Redis superiores, cree una instancia de DCS Redis de una versión superior y, a continuación, migre los datos de la instancia antigua a la nueva.

Características

Las instancias principal/en standby de DCS tienen mayor disponibilidad y confiabilidad que las instancias de un nodo único de DCS.

Las instancias principal/en standby de DCS tienen las siguientes características:

1. **Persistencia de los datos y alta confiabilidad**

De forma predeterminada, la persistencia de datos está habilitada tanto por el nodo principal como por el en standby de una instancia principal/en standby de DCS para Redis.

El nodo en standby de una instancia de Redis 3.0 es invisible para usted. Solo el nodo principal proporciona las operaciones de lectura/escritura de datos.

El nodo en standby de una instancia de Redis 4.0 o 5.0 es visible para usted. Puede leer datos del nodo en standby conectándose con la dirección de sólo lectura de la instancia.

El nodo en standby de una instancia de Redis 6.0 es invisible para usted. Puede leer datos del nodo en standby conectándose con la dirección de sólo lectura de la instancia.

2. **Sincronización de datos**

   Los datos en los nodos principal y en standby se mantienen consistentes a través de la sincronización incremental.

   NOTA

   Después de recuperarse de una excepción de red o un fallo de nodo, las instancias principal/en standby realizan una sincronización completa para garantizar la coherencia de los datos.

3. **Conmutación automática principal/en standby**

   Si el nodo principal se vuelve defectuoso, la instancia se desconecta y no está disponible durante varios segundos. El nodo en standby se hace cargo en 30 segundos sin operaciones manuales para reanudar los servicios estables.

4. **Múltiples políticas de recuperación ante desastres**

   Cada instancia principal/en standby de DCS se puede implementar en AZs con fuentes de alimentación y redes físicamente aisladas. Las aplicaciones también se pueden implementar en AZ para lograr HA tanto para datos como para aplicaciones.

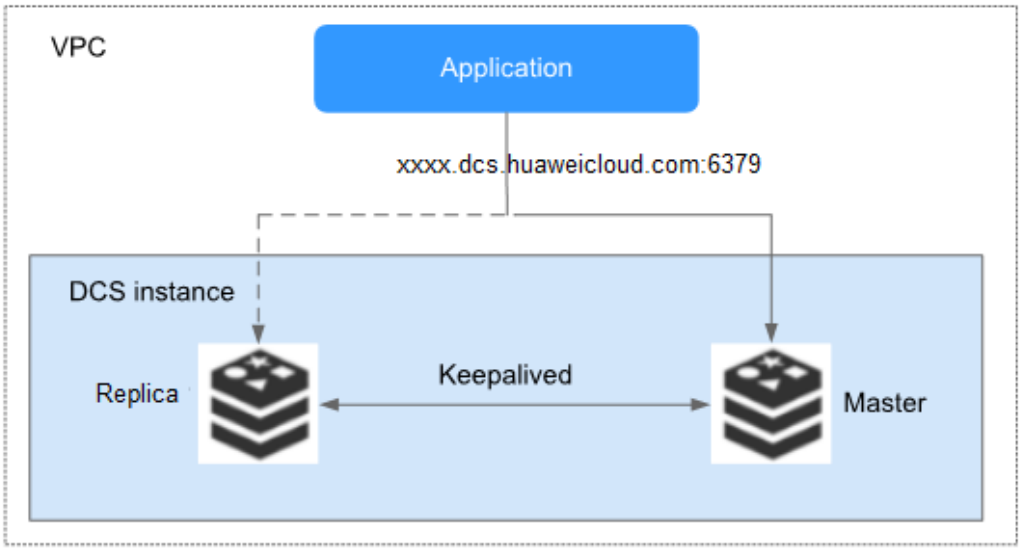
5. **Separación de lecturas/escrituras**

   Las instancias principal/en standby de DCS compatibles con Redis 4.0 y 5.0 admiten la separación de lectura/escritura del cliente. Al conectarse a una instancia de este tipo, puede utilizar la dirección de lectura/escritura para conectarse al nodo principal o utilizar la dirección de sólo lectura para conectarse al nodo en standby.

Arquitectura de instancias principal/en standby compatibles con Redis 3.0

Figura 3-2 muestra la arquitectura de una instancia principal/en standby de DCS compatible con Redis 3.0.

Figura 3-2 Arquitectura de instancia principal/en standby de DCS compatible con Redis 3.0.



Descripción de la arquitectura:

- **VPC**

  La VPC donde se ejecutan todos los nodos de la instancia.

  NOTA

  Para el acceso intra-VPC, el cliente y la instancia deben estar en la misma VPC con configuraciones de reglas de grupo de seguridad especificadas.

  Se puede acceder a una instancia de DCS para Redis 3.0 desde una VPC o a través de redes públicas. El cliente que accede a la instancia se puede desplegar fuera de la VPC y acceder a la instancia a través del EIP enlazado a la instancia. Las instancias de DCS compatibles con Redis 6.0, 5.0 y 4.0 no admiten el acceso público.

  Para obtener más información, vea **Acceso público a una instancia de DCS para Redis** y **¿Cómo configuro un grupo de seguridad?**

- **Application**

  El cliente Redis de la instancia, que es la aplicación que se ejecuta en el ECS.

  Las instancias de DCS para Redis y Memcached son respectivamente compatibles con los protocolos Redis y Memcached, y se puede acceder a través de clientes de código abierto. Para obtener ejemplos de acceso a instancias de DCS con diferentes lenguajes de programación, consulte las **instrucciones de acceso a instancias**.

- **DCS instance**

  Una instancia principal/en standby de DCS que tiene un nodo principal y un nodo de réplica. De forma predeterminada, la persistencia de datos está habilitada y los datos se sincronizan entre los dos nodos.

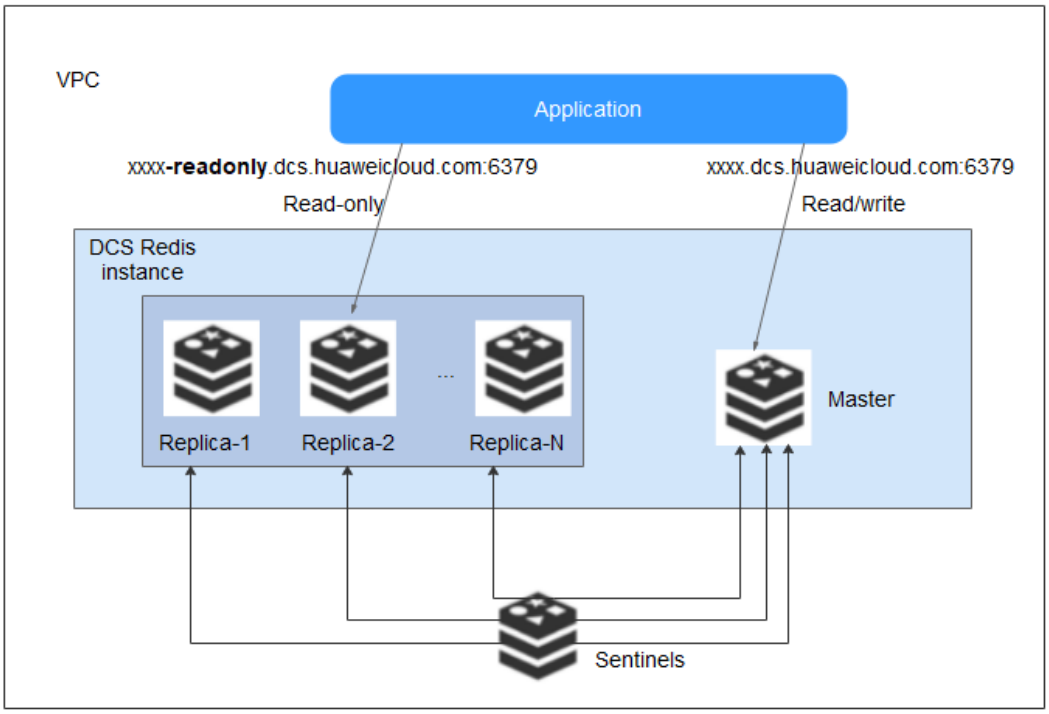
  DCS monitorea la disponibilidad de la instancia en tiempo real. Si el nodo principal se vuelve defectuoso, el nodo en espera se convierte en el nodo principal y reanuda el aprovisionamiento de servicio.

  El puerto Redis predeterminado es 6379.

Arquitectura de instancias principal/en standby compatibles con Redis 4.0 y 5.0

Figura 3-3 muestra la arquitectura de las instancias principal/en standby de DCS compatibles con Redis 4.0 y 5.0.

Figura 3-3 Arquitectura de una instancia principal/en standby de DCS compatible con Redis 4.0 o 5.0



Descripción de la arquitectura:

1. Cada instancia principal/en standby compatible con Redis 4.0 o 5.0 tiene dos direcciones de conexión. Al conectarse a una instancia de este tipo, puede utilizar la dirección de nombre de dominio de lectura/escritura para conectarse al nodo principal o utilizar la dirección de nombre de dominio de sólo lectura para conectarse al nodo en espera.

   Las direcciones de conexión se pueden obtener en la página de detalles de la instancia en la consola DCS.

2. Puede configurar Sentinel para una instancia principal/en standby para Redis 4.0 o 5.0. Los Sentinel supervisan el estado de ejecución de los nodos principal y en standby. Cuando el nodo principal se vuelve defectuoso, se realizará una migración por falla.

   Los Sentinel son invisibles para usted y solo se utilizan en el servicio. Para obtener más información sobre Sentinel, consulte **¿Qué es Sentinel?**

3. Un nodo de sólo lectura tiene las mismas especificaciones que un nodo de lectura/escritura. Cuando se crea una instancia principal/en standby, se incluyen un par de nodos principal y en espera en la instancia de forma predeterminada.

NOTA

- Para las instancias de DCS compatibles con Redis 4.0 o 5.0, puede personalizar el puerto. Si no se especifica ningún puerto, se utilizará el puerto predeterminado 6379. En el diagrama de arquitectura, se utiliza el puerto 6379. Si ha personalizado un puerto, reemplace **6379** por el puerto real.
- Los nombres de dominio de solo lectura de las instancias principal/en standby de DCS compatibles con Redis 4.0 y 5.0 no admiten el equilibrio de carga. Para una alta confiabilidad y baja latencia, utilice instancias de separación de clúster o lectura/escritura.

3.3 Clúster Proxy para Redis

DCS for Redis proporciona dos tipos de instancias de clúster: Clúster Proxy y Clúster Redis. Clúster Proxy es compatible con Redis 3.0, 4.0 y 5.0, y utiliza Linux Virtual Server (LVS) y los proxy para lograr alta disponibilidad. Clúster Redis es la implementación distribuida nativa de Redis y es compatible con Redis 4.0 y 5.0.

La separación de lectura y escritura se admite mediante la configuración del cliente para las instancias de Clúster Redis (Redis 4.0 y 5.0), pero no se admite para las instancias de Clúster Proxy (Redis 3.0, 4.0 y 5.0). **Leer más** sobre el soporte de DCS para la separación de lectura/escritura.

En esta sección se describen las instancias de Clúster Proxy de DCS compatibles con Redis 3.0, 4.0 y 5.0.

NOTA

- DCS for Redis 3.0 ya no se proporciona. Puede utilizar DCS for Redis 4.0 o 5.0 en su lugar.
- No puede actualizar la versión de Redis para una instancia. Por ejemplo, una instancia de Clúster Proxy de DCS compatible con Redis 3.0 no se puede actualizar a una instancia de Clúster Proxy de DCS compatible con Redis 4.0 o 5.0. Si el servicio requiere las características de versiones de Redis superiores, cree una instancia de DCS Redis de una versión superior y, a continuación, migre los datos de la instancia antigua a la nueva.
- Una instancia de Clúster Proxy puede conectarse de la misma manera que una instancia de nodo único o principal/en standby está conectada, sin ninguna configuración especial en el cliente. Puede utilizar la dirección IP o el nombre de dominio de la instancia, y no es necesario conocer o utilizar las direcciones de proxy o de partición.

Instancia de Clúster Proxy de DCS compatible con Redis 3.0

Las instancias de Clúster Proxy de DCS compatibles con Redis 3.0 se basan en x86, son compatibles con **codis de código abierto** y vienen con especificaciones que van desde 64 GB a 1024 GB, que cumplen con los requisitos para **millones de conexiones simultáneas** y **caché de datos masivos**. El almacenamiento y acceso de datos distribuidos es implementado por DCS, sin necesidad de desarrollo o mantenimiento.

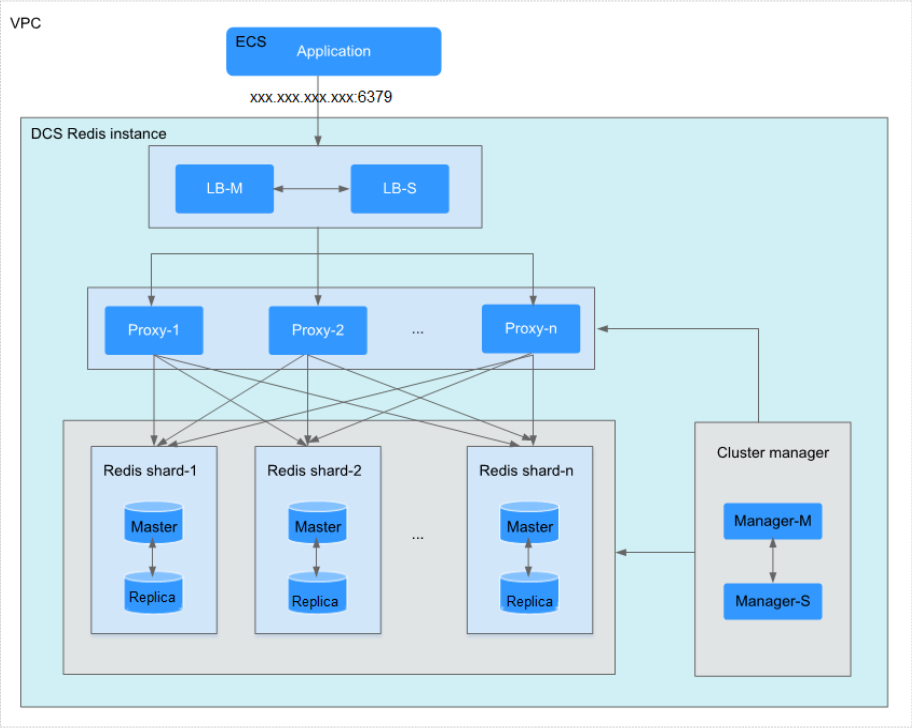
Cada instancia de Clúster Proxy consta de balanceadores de carga, proxy, administradores de clústeres y **particiones**.

Tabla 3-1 Especificaciones de instancias de Clúster Proxy de DCS compatibles con Redis 3.0

Memoria total	Proxy	Particiones (Shards)
64 GB	3	8

Memoria total	Proxy	Particiones (Shards)
128 GB	6	16
256 GB	8	32
512 GB	16	64
1024 GB	32	128

Figura 3-4 Arquitectura de una instancia de Clúster Proxy de DCS compatible con Redis 3.0



Descripción de la arquitectura:

- **VPC**

  La VPC donde se ejecutan todos los nodos de la instancia.

  NOTA

  Si el acceso público no está habilitado para la instancia, asegúrese de que el cliente y la instancia están en la misma VPC y configure las reglas de grupo de seguridad para la VPC.

  Si el acceso público está habilitado para la instancia, el cliente puede desplegarse fuera de la VPC para acceder a la instancia a través del EIP enlazado a la instancia.

  Para obtener más información, vea **Acceso público a una instancia de DCS compatible con Redis 3.0** y **¿Cómo configuro un grupo de seguridad?**

- **Application**

  El cliente utilizado para acceder a la instancia.

Se puede acceder a las instancias de DCS para Redis mediante clientes de código abierto. Para obtener ejemplos de acceso a instancias de DCS con diferentes lenguajes de programación, consulte **Acceso a una instancia de DCS compatible con Redis**.

- **LB-M/LB-S**

  Los balanceadores de carga, que se despliegan en modo HA principal/en standby. Las direcciones de conexión (**dirección IP:Puerto** y **nombre de dominio:Puerto**) de la instancia del clúster de DCS compatible con Redis son las direcciones de los balanceadores de carga.

- **Proxy**

  El servidor proxy utilizado para lograr una alta disponibilidad y procesar solicitudes de clientes de alta simultaneidad.

  Puede conectarse a una instancia Clúster Proxy en las direcciones IP de sus proxy.

- **Redis shard**

  Una partición del clúster.

  Cada partición consta de un par de nodos principal/de réplica. Si el nodo principal se vuelve defectuoso, el nodo de réplica se hace cargo automáticamente de los servicios del clúster.

  Si tanto el nodo principal como el de réplica de una partición son defectuosos, el clúster aún puede proporcionar servicios, pero los datos de la partición defectuosa son inaccesibles.

- **Cluster manager**

  Los administradores de configuración del clúster, que almacenan las configuraciones y las políticas de particionamiento del clúster. No puede modificar la información acerca de los administradores de configuración.

Instancias de Clúster Proxy de DCS compatibles con Redis 4.0 y 5.0

NOTA

Las instancias de Clúster Proxy de DCS compatibles con Redis 4.0 y 5.0 solo se proporcionan en algunas regiones.

Las instancias de Clúster Proxy de DCS compatibles con Redis 4.0 y 5.0 se construyen basadas en Redis 4.0 y 5.0 de código abierto y son compatibles con **codis de código abierto**. Proporcionan varias especificaciones de gran capacidad que van desde 4 GB a 1024 GB y son compatibles con las arquitecturas de CPU x86 y Arm.

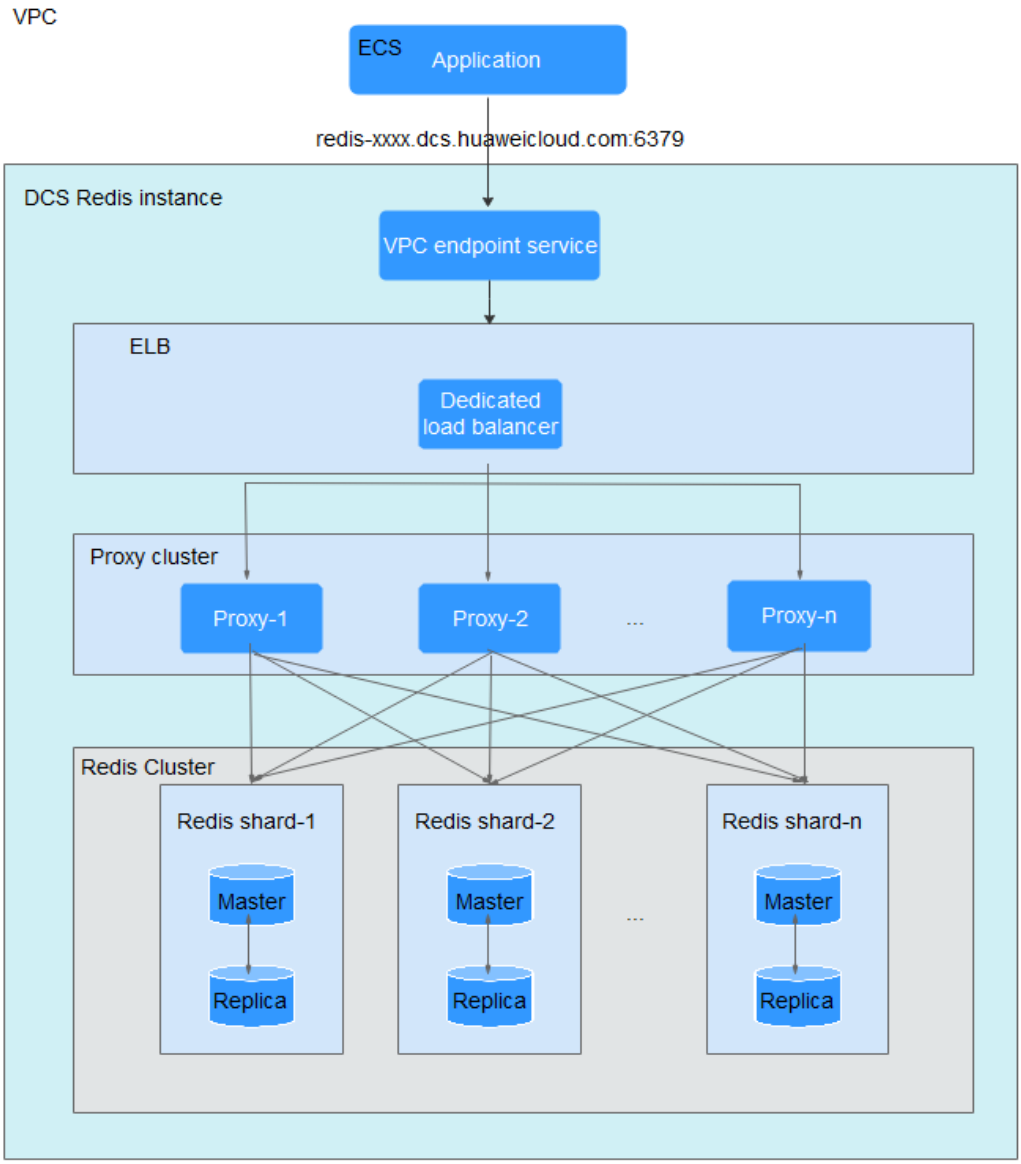
Tabla 3-2 enumera el número de particiones correspondientes a diferentes especificaciones. Puede personalizar el tamaño del partición al crear una instancia. Actualmente, el número de fragmentos y réplicas no se puede personalizar. Por defecto, cada partición tiene dos réplicas.

Memoria por partición=Especificación de instancia/Número de particiones. Por ejemplo, si una instancia de 48 GB tiene 6 particiones, el tamaño de cada partición es de 48 GB/6 = 8 GB.

Tabla 3-2 Especificaciones de instancias de Clúster Proxy de DCS compatibles con Redis 4.0 y 5.0

Memoria total	Proxy	Particiones (Shards)	Memoria por partición (GB)
4 GB	3	3	1.33
8 GB	3	3	2.67
16 GB	3	3	5.33
24 GB	3	3	8
32 GB	3	3	10.67
48 GB	6	6	8
64 GB	8	8	8
96 GB	12	12	8
128 GB	16	16	8
192 GB	24	24	8
256 GB	32	32	8
384 GB	48	48	8
512 GB	64	64	8
768 GB	96	96	8
1024 GB	128	128	8

Figura 3-5 Arquitectura de una instancia de Clúster Proxy de DCS compatible con Redis 4.0 y 5.0



Descripción de la arquitectura:

- **VPC**

  La VPC donde se ejecutan todos los nodos de la instancia.

  NOTA

  El cliente y la instancia del clúster deben estar en la misma VPC, y la lista blanca de la instancia debe permitir el acceso desde la dirección IP del cliente.

- **Application**

  El cliente utilizado para acceder a la instancia.

  Se puede acceder a las instancias de DCS para Redis mediante clientes de código abierto. Para obtener ejemplos de acceso a instancias de DCS con diferentes lenguajes de programación, consulte **Acceso a una instancia de DCS compatible con Redis**.

- **VPC endpoint service**

  Puede configurar su instancia de DCS compatible con Redis como un servicio de endpoint de VPC y acceder a la instancia en la dirección de servicio de endpoint de VPC.

  La dirección IP o la dirección de nombre de dominio de la instancia Clúster Proxy de DCS compatible con Redis es la dirección del servicio de endpoint de VPC.

- **ELB**

  Los balanceadores de carga se implementan en modo HA de clúster y admiten la implementación multi-AZ.

- **Proxy**

  El servidor proxy utilizado para lograr una alta disponibilidad y procesar solicitudes de clientes de alta simultaneidad.

  No se puede conectar a una instancia de Clúster Proxy en las direcciones IP de sus proxies.

- **Clúster Redis**

  Una partición del clúster.

  Cada partición consta de un par de nodos principal/de réplica. Si el nodo principal se vuelve defectuoso, el nodo de réplica se hace cargo automáticamente de los servicios del clúster.

  Si tanto el nodo principal como el de réplica de una partición son defectuosos, el clúster aún puede proporcionar servicios, pero los datos de la partición defectuosa son inaccesibles.

3.4 Clúster Redis

DCS proporciona dos tipos de instancias de clúster para Redis: Clúster Proxy y Clúster Redis. Clúster Proxy utiliza Linux Virtual Server (LVS) y los proxy. Clúster Redis es la implementación distribuida nativa de Redis. Las instancias de Clúster Proxy son compatibles con Redis 3.0, 4.0 y 5.0, mientras que las instancias de Clúster Redis son compatibles con Redis 4.0 y 5.0.

La separación de lectura y escritura se admite mediante la configuración del cliente para las instancias de Clúster Redis (Redis 4.0 y 5.0), pero no se admite para las instancias de Clúster Proxy (Redis 3.0, 4.0 y 5.0). **Leer más** sobre el soporte de DCS para la separación de lectura/escritura.

En esta sección se describen las instancias de Clúster Redis de DCS compatibles con Redis 4.0 y 5.0.

Instancias de Clúster Redis de DCS compatibles con Redis 4.0 y 5.0

El tipo de instancia de Clúster Redis proporcionado por DCS es compatible con el **Clúster Redis nativo**, que utiliza clientes inteligentes y una arquitectura distribuida para realizar la partición.

Tabla 3-3 enumera las especificaciones de la partición para diferentes especificaciones de instancia.

Puede personalizar el tamaño de la partición al crear una instancia de Clúster Redis. Si el tamaño de la partición no está personalizado, se utiliza el tamaño predeterminado. **Tamaño de un partición = Especificación de instancia/Número de particiones**. Por ejemplo, si una instancia de 48 GB tiene 6 particiones, el tamaño de cada partición es de 48 GB/6 = 8 GB.

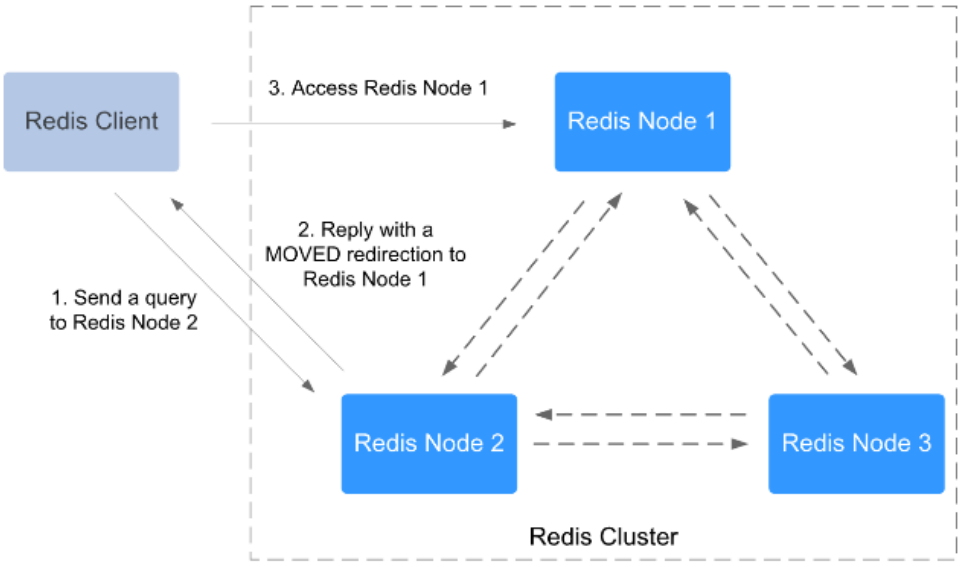
Tabla 3-3 Especificaciones de las instancias de Clúster Redis de DCS

Memoria total	Particiones (Shards)
4 GB/8 GB/16 GB/24 GB/32 GB	3
48 GB	6
64 GB	8
96 GB	12
128 GB	16
192 GB	24
256 GB	32
384 GB	48
512 GB	64
768 GB	96
1024 GB	128

- Arquitectura distribuida

  Cualquier nodo de un Clúster Redis puede recibir solicitudes. Las solicitudes recibidas se redirigen luego al nodo correcto para su procesamiento. Cada nodo consta de un subconjunto de un principal y una (de forma predeterminada) o múltiples réplicas. Los roles principal o réplica se determinan a través de un algoritmo de elección.

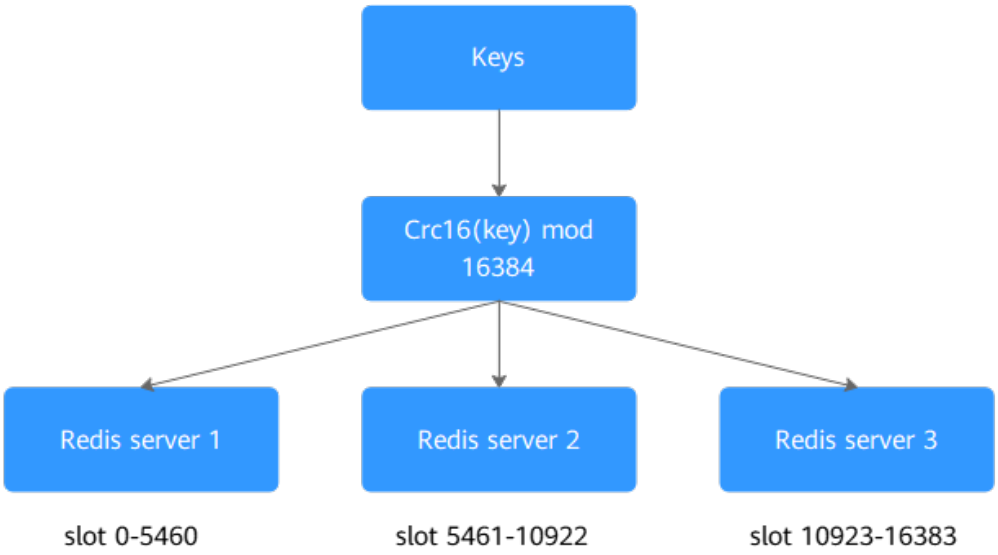
  Figura 3-6 Arquitectura distribuida del Clúster Redis

- Partición previa

En cada Clúster Redis hay 16,384 ranuras hash. La asignación entre las ranuras hash y los nodos de Redis se almacena en los servidores de Redis. Para calcular cuál es la ranura hash de la clave dada, simplemente tome el CRC16 del módulo de clave 16384.

Figura 3-7 Partición previa de Clúster Redis



3.5 Separación de lecturas/escrituras

Esta sección describe las instancias DCS principal/en standby en las que se implementa la separación de lectura/escritura en el lado del servidor o en el lado del cliente. La separación de lectura/escritura es adecuada para escenarios con alta simultaneidad de lectura y pocas solicitudes de escritura, con el objetivo de mejorar el rendimiento de la alta simultaneidad y reducir los costos de operación.

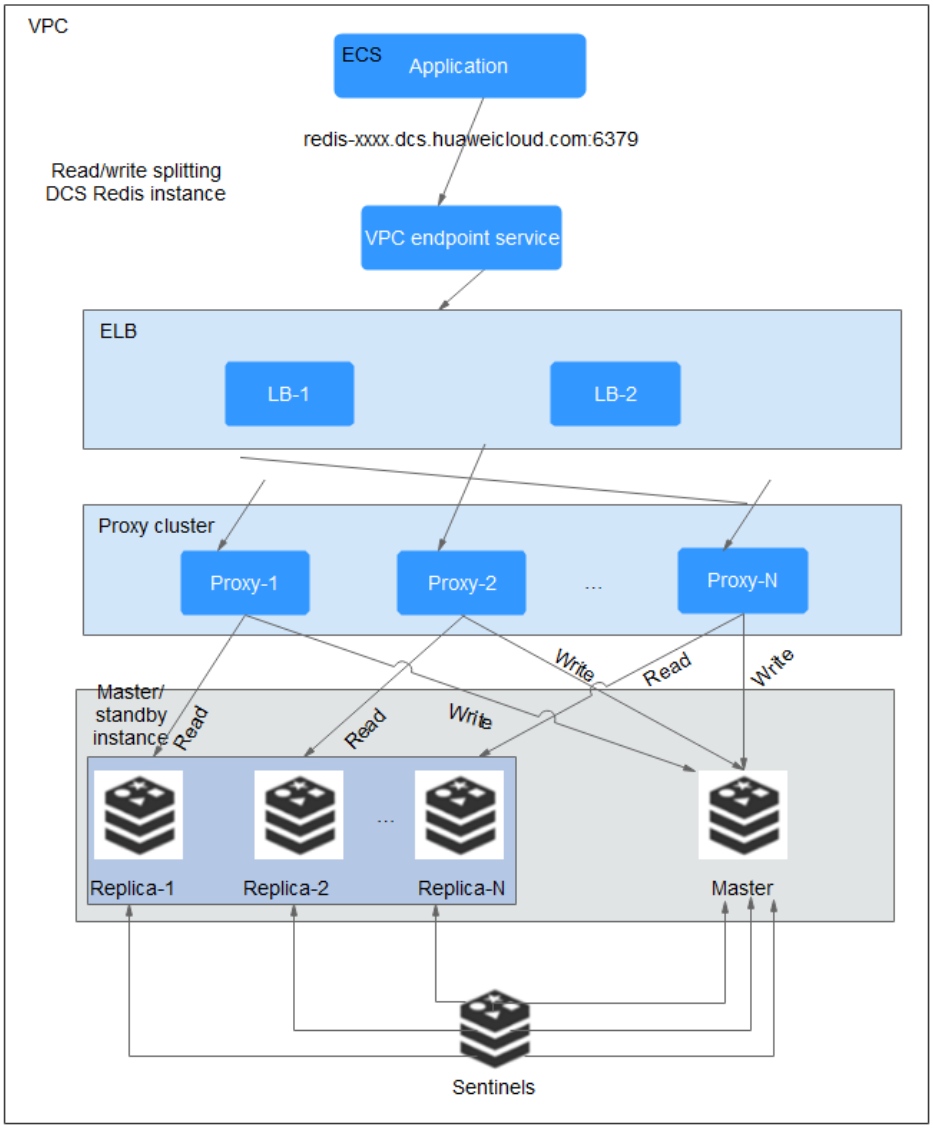
NOTA

La separación de lectura/escritura del servidor y la separación de lectura/escritura del cliente solo se admiten en algunas regiones.

- Para una instancia de separación de lectura/escritura de DCS para Redis 4.0 o 5.0, la separación de lectura/escritura se implementa en el lado del servidor de forma predeterminada. Los proxy distinguen entre las peticiones de lectura y las de escritura, y reenvían las de escritura al nodo principal y las de lectura al nodo en standby. No es necesario configurar el cliente.
- Para una instancia de DCS para Redis 4.0 o 5.0 principal/en standby, la separación de lectura/escritura se implementa por defecto en el lado del cliente. La instancia proporciona un nombre de dominio de lectura/escritura y un nombre de dominio de solo lectura para recibir respectivamente solicitudes de lectura y escritura del cliente. De esta manera, se consigue la separación de lectura/escritura.

Separación de lectura/escritura del servidor

Figura 3-8 Arquitectura de una instancia de separación de lectura/escritura



Descripción de la arquitectura:

- **Servicio de endpoint de VPC**

  Puede configurar su instancia de DCS compatible con Redis como un servicio de endpoint de VPC y acceder a la instancia mediante la dirección de servicio de endpoint de VPC.

  La dirección IP o el nombre de dominio de la instancia de separación de lectura/escritura de DCS compatible con Redis es la dirección del servicio de endpoint de VPC.

- **ELB**

  Los balanceadores de carga se implementan en modo HA de clúster y admiten la implementación multi-AZ.

- **Proxy**

Se usa un clúster proxy para distinguir entre las solicitudes de lectura y las de escritura, y reenviar las solicitudes de escritura al nodo principal y las de lectura al nodo de reserva. No es necesario configurar el cliente.

- **Sentinel cluster**

  Los Sentinel monitorean el estado del principal y de las réplicas. Si el nodo principal es defectuoso o anormal, se realiza una conmutación por error para garantizar que los servicios no se interrumpan.

- **Master/standby instance**

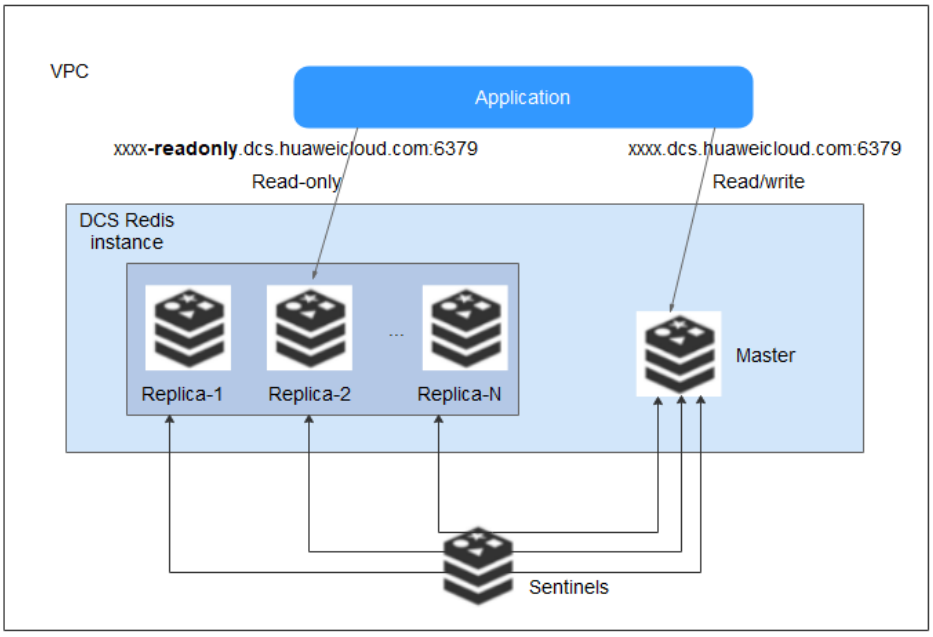
  Una instancia de separación de lectura/escritura es esencialmente una instancia principal/en standby que consiste en un nodo principal y un nodo en standby. De forma predeterminada, la persistencia de datos está habilitada y los datos se sincronizan entre los dos nodos.

  Los nodos principal y en standby pueden desplegarse en diferentes AZ.

Separación de lectura/escritura del cliente

Figura 3-9 muestra la arquitectura de una instancia principal/en standby de DCS compatible con Redis 4.0 y 5.0.

Figura 3-9 Arquitectura de una instancia principal/en standby de DCS compatible con Redis 4.0 o 5.0



Descripción de la arquitectura:

1. Una instancia de separación de lectura/escritura de DCS para Redis 4.0 o 5.0 tiene un nombre de dominio de lectura/escritura y un nombre de dominio de sólo lectura. El cliente distingue entre las peticiones de lectura y las de escritura, y envía las de escritura al nombre de dominio de lectura/escritura y las de lectura al nombre de dominio de sólo lectura para conectarse al nodo principal o nodo en standby.

   Los dos nombres de dominio se pueden obtener en la página de detalles de instancia de la consola.

2. Las instancias principal/en standby de DCS compatibles con Redis 4.0 y 5.0 soportan los Sentinel. Los Sentinel supervisan el estado de ejecución de los nodos principal y en standby. Si el nodo principal se vuelve defectuoso, se realizará una migración por falla.

   Los Sentinel son invisibles para usted y solo se utilizan en el servicio.

3. Un nodo de sólo lectura tiene las mismas especificaciones que un nodo de lectura/escritura. Una instancia principal/en standby consta de un par de nodos principal y en standby de forma predeterminada.

Cuando utilice instancias de separación de lectura/escritura, tenga en cuenta lo siguiente:

1. Las solicitudes de lectura se envían a réplicas. Hay un retraso cuando los datos se sincronizan desde el principal a las réplicas.

   Si sus servicios son sensibles al retraso, no utilice instancias de separación de lectura/escritura. En su lugar, utilice instancias de principal/en standby o de clúster.

2. La separación de lectura/escritura es adecuada cuando hay más solicitudes de lectura que solicitudes de escritura.If there are a lot of write requests, the master and replicas may be disconnected, or the data synchronization between them may fail after the disconnection. As a result, the read performance deteriorates.

   If your services are write-heavy, use master/standby or cluster instances.

3. If a replica is faulty, it takes some time to synchronize all data from the master. During the synchronization, the replica does not provide services, and the read performance of the instance deteriorates.

   To reduce the impact of the fault, use an instance with less than 32 GB memory. The smaller the memory, the shorter the time for full data synchronization between the master and replicas, and the smaller the impact of the interruption.

3.6 Comparación de tipos de instancias de DCS para Redis

Tabla 3-4 describe las diferencias entre los diferentes tipos de instancia de Redis en términos de características y comandos.

NOTA

DCS for Redis 3.0 ya no se proporciona. Puede utilizar DCS for Redis 4.0 o 5.0 en su lugar.

Tabla 3-4 Diferencias entre los tipos de instancia de DCS

Artículo	Nodo único o Principal/En standby	Clúster Proxy	Clúster Redis
Compatibilidad con versiones de Redis	Redis 3.0, 4.0 y 5.0 Redis 6.0 es compatible con KeyDB Open Source (disponible solo para instancias principal/en standby). Puede seleccionar una versión al crear una instancia.	Redis 3.0, 4.0 y 5.0	Redis 4.0 y 5.0 Puede seleccionar una versión al crear una instancia.
Soporte	● Notificaciones de espacio de claves ● Pipelining	● Pipelining, comando **MSET**, y comando **MGET** ● Comando **SCAN**, comando **KEYS** y Registro lento de Redis ● Pub/Sub	● Notificaciones de espacio de claves ● Comandos BRPOP, BLPOP y BRPOPLPUSH ● Pub/Sub

Artículo	Nodo único o Principal/En standby	Clúster Proxy	Clúster Redis
Restricciones	La persistencia de datos no es compatible con las instancias de nodo único.	● El script LUA está restringido: Todas las claves deben estar en la misma ranura hash para evitar errores. Se recomiendan Hash tags. ● Si un comando contiene varias claves, las claves deben estar en la misma ranura hash para evitar errores. Se recomiendan Hash tags. ● No se admiten las notificaciones de espacio de claves.	● El script LUA está restringido: todas las claves deben estar en la misma ranura hash. Se recomiendan Hash tags. ● El SDK del cliente debe ser compatible con Clúster Redis y ser capaz de procesar errores MOVED. ● Cuando se utiliza pipelining, el comando MSET o el comando MGET, todas las claves deben estar en el mismo intervalo hash para evitar errores. Se recomiendan Hash tags. ● Cuando utilice las notificaciones de espacio de claves, establezca conexiones con cada servidor Redis y procese eventos en cada conexión. ● Cuando utilice un comando transversal o global como SCAN y KEYS, ejecute el comando en cada servidor Redis.
Cliente	Cualquier cliente de Redis	Cualquier cliente Redis (sin necesidad de soportar el protocolo Clúster Redis)	Cualquier cliente que soporte el protocolo de Clúster Redis
Comandos deshabilitados	Algunos comandos de Redis no son compatibles. Para obtener más información, consulte **Tabla 5-3**, **Tabla 5-11** y **Tabla 5-21**.	Algunos comandos de Redis no son compatibles. Para más detalles, consulte **Tabla 5-4**.	Algunos comandos de Redis no son compatibles. Para obtener más información, consulte **Tabla 5-13** y **Tabla 5-23**.

Artículo	Nodo único o Principal/En standby	Clúster Proxy	Clúster Redis
Réplicas	Una instancia de nodo único solo tiene una réplica. Una instancia principal/en standby tiene dos réplicas. Actualmente, el número de réplicas no se puede personalizar para instancias principal/en standby de DCS para Redis 3.0 y DCS Redis 6.0. De forma predeterminada, una instancia principal/en standby tiene un nodo principal y un nodo en standby. Al crear una instancia principal/en standby de DCS para Redis 4.0 o 5.0, puede personalizar el número de réplicas, siendo una de ellas el principal.	Cada partición en el clúster tiene y solo puede tener dos réplicas, siendo una de ellas el principal.	De forma predeterminada, cada partición del clúster tiene dos réplicas. El número de réplicas se puede personalizar, siendo una de ellas la maestra. Al crear una instancia, puede establecer la cantidad de réplica en uno, lo que indica que la instancia solo tiene el nodo principal. En este caso, no se puede garantizar una alta fiabilidad de los datos.

3.7 Memcached de nodo único (no disponible pronto)

NOTA

DCS for Memcached está a punto de no estar disponible y ya no se vende en algunas regiones. Puede utilizar DCS for Redis 4.0 o 5.0 en su lugar.

En esta sección se describen las características y la arquitectura de las instancias de nodo único de DCS compatibles con Memcached.

Características

1. Baja sobrecarga del sistema y alto QPS

   Las instancias de nodo único no admiten sincronización de datos ni persistencia de datos, lo que reduce la sobrecarga del sistema y admite una mayor simultaneidad. El QPS de las instancias de nodo único de DCS para Memcached alcanza hasta 100,000.

2. Monitoreo de procesos y recuperación automática de fallos

   Con un mecanismo de monitoreo de HA, si una instancia de DCS de un nodo único resulta defectuosa, se inicia un nuevo proceso en 30 segundos para reanudar el aprovisionamiento del servicio.

3. Usabilidad lista para usar y sin persistencia de datos

   Las instancias de un nodo único de DCS se pueden usar de inmediato porque no implican la carga de datos. Si su servicio requiere un alto QPS, puede calentar los datos de antemano para evitar un fuerte impacto de simultaneidad en la base de datos backend.

4. Bajo costo y adecuado para desarrollo y pruebas

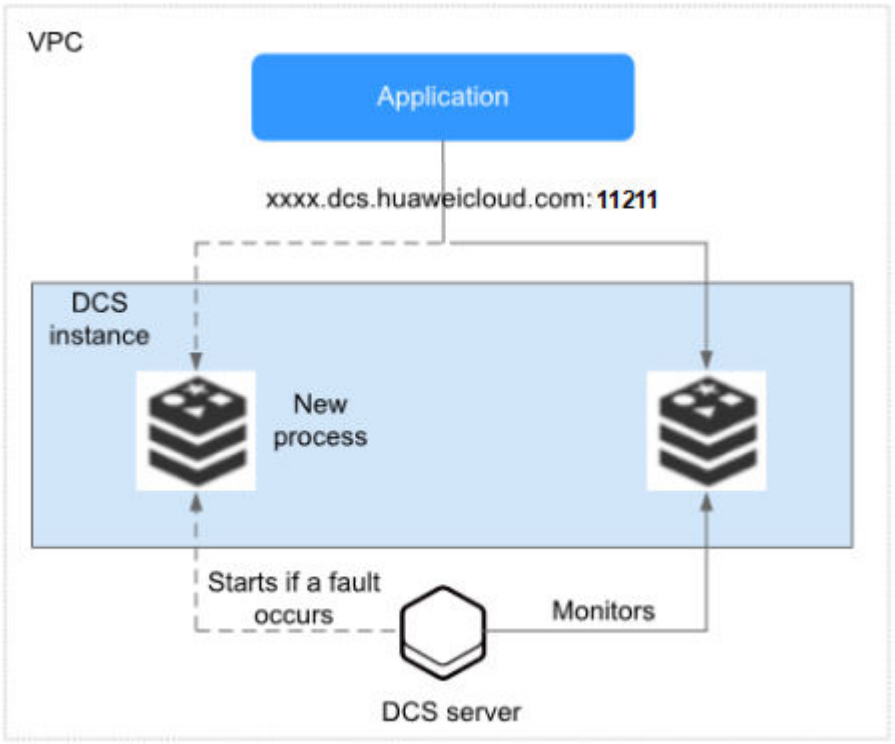
   Las instancias de un nodo único son un 40% más baratas que las instancias DCS principal/en standby, y son adecuadas para configurar entornos de desarrollo o pruebas.

En resumen, las instancias de nodo único de DCS admiten las operaciones de lectura/escritura altamente simultáneas, pero no admiten persistencia de datos. Los datos se eliminarán después de reiniciar las instancias. Son adecuados para escenarios que no requieren persistencia de datos, como el almacenamiento en caché front-end de la base de datos, para acelerar el acceso y facilitar la carga de concurrencia fuera del back-end. Si los datos deseados no existen en la caché, las solicitudes irán a la base de datos. Al reiniciar el servicio o la instancia de DCS, puede pregenerar los datos de caché de la base de datos de disco para aliviar la presión sobre el backend durante el inicio.

Arquitectura

Figura 3-10 muestra la arquitectura de instancias de un solo nodo de DCS para Memcached.

Figura 3-10 Arquitectura de instancia de nodo único de DCS



Descripción de la arquitectura:

- **VPC**

  La VPC donde se ejecutan todos los nodos de la instancia.

  NOTA

  Las instancias de nodo único de DCS para Memcached no admiten el acceso público. El cliente y la instancia deben estar en la misma VPC con las configuraciones de reglas de grupo de seguridad.

  Para obtener más información, vea **¿Cómo configuro un grupo de seguridad?**

- **Application**

  El cliente de la instancia, que es la aplicación que se ejecuta en un Elastic Cloud Server (ECS).

  Las instancias de DCS para Memcached son compatibles con el protocolo de Memcached y se puede acceder a ellas a través de clientes de código abierto. Para obtener ejemplos de acceso a instancias de DCS con diferentes lenguajes de programación, consulte **Acceso a una instancia de DCS para Memcached**.

- **DCS instance**

  Una instancia de nodo único de DCS, que tiene un solo nodo y un proceso de Memcached.

  DCS monitorea la disponibilidad de la instancia en tiempo real. Si el proceso de Memcached se vuelve defectuoso, DCS inicia un nuevo proceso para reanudar el aprovisionamiento del servicio.

  Utilice el puerto 11211 para acceder a una instancia de DCS compatible con Memcached.

3.8 Memcached principal/en standby (no disponible pronto)

NOTA

DCS for Memcached está a punto de no estar disponible y ya no se vende en algunas regiones. Puede utilizar DCS for Redis 4.0 o 5.0 en su lugar.

En esta sección se describen las instancias principal/en standby de DCS compatibles con Memcached.

Características

Las instancias principal/en standby tienen mayor disponibilidad y confiabilidad que las instancias de un nodo único.

Las instancias principal/en standby de DCS compatibles con Memcached tienen las siguientes características:

1. **Persistencia de los datos y alta confiabilidad**

   De forma predeterminada, la persistencia de datos está habilitada tanto por el nodo principal como por el en standby de una instancia principal/en standby de DCS para Memcached. Además, se admite la persistencia de datos para garantizar una alta fiabilidad de los datos.

   El nodo en standby de una instancia de DCS para Memcached es invisible para usted. Solo el nodo principal proporciona las operaciones de lectura/escritura de datos.

2. **Sincronización de datos**

   Los datos en los nodos principal y en standby se mantienen consistentes a través de la sincronización incremental.

   NOTA

   Después de recuperarse de una excepción de red o un fallo de nodo, las instancias de principal/en standby realizan una sincronización completa para garantizar la coherencia de los datos.

3. **Conmutación automática principal/en standby**

   Si el nodo principal se vuelve defectuoso, el nodo de reserva se hace cargo en 30 segundos, sin requerir ninguna interrupción de servicio o operaciones manuales.

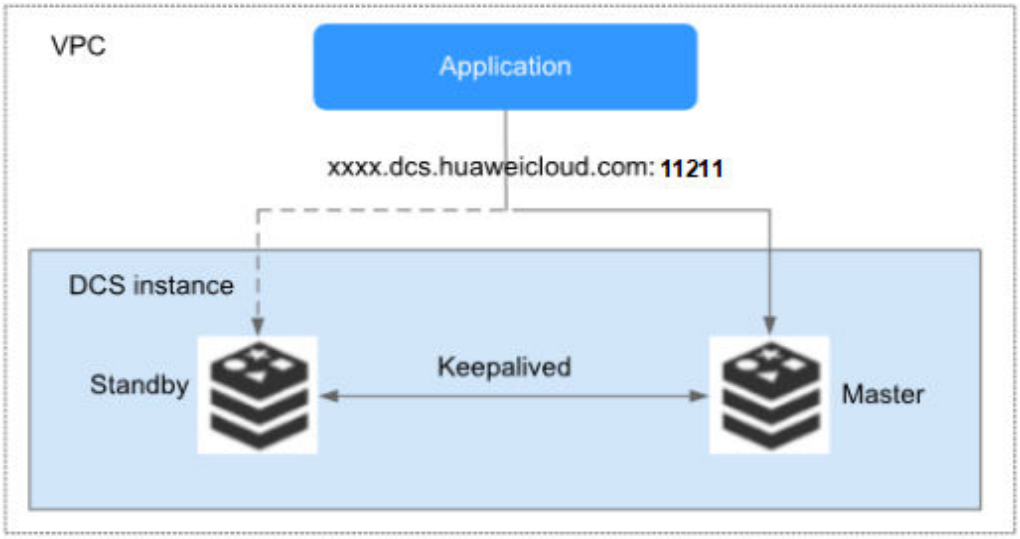
4. **Múltiples políticas de recuperación ante desastres**

   Cada instancia principal/en standby se puede implementar en AZs con fuentes de alimentación y redes físicamente aisladas. Las aplicaciones también se pueden implementar en AZ para lograr HA tanto para datos como para aplicaciones.

Arquitectura de instancias principal/en standby de DCS compatibles con Memcached

Figura 3-11 muestra la arquitectura de las instancias principal/en standby de DCS compatibles con Memcached.

Figura 3-11 Arquitectura de instancia principal/en standby de DCS compatible con Memcached



Descripción de la arquitectura:

- **VPC**

  La VPC donde se ejecutan todos los nodos de la instancia.

  NOTA

  Las instancias principal/en standby de DCS compatibles con Memcached no admiten el acceso público. El cliente y la instancia deben estar en la misma VPC con las configuraciones de reglas de grupo de seguridad.

  Para obtener más información, vea **¿Cómo configuro un grupo de seguridad?**

- **Application**

  El cliente Memcached de la instancia, que es la aplicación que se ejecuta en el ECS.

  Las instancias de DCS para Memcached son compatibles con el protocolo de Memcached y se puede acceder a ellas a través de clientes de código abierto. Para obtener ejemplos de acceso a instancias de DCS con diferentes lenguajes de programación, consulte **Acceso a una instancia de DCS para Memcached**.

- **DCS instance**

  Indica una instancia principal/en standby de DCS que tiene un nodo principal y un nodo en standby. De forma predeterminada, la persistencia de datos está habilitada y los datos se sincronizan entre los dos nodos.

  DCS monitorea la disponibilidad de la instancia en tiempo real. Si el nodo principal se vuelve defectuoso, el nodo en espera se convierte en el nodo principal y reanuda el aprovisionamiento de servicio.

  Utilice el puerto 11211 para acceder a una instancia de DCS compatible con Memcached.

4 Especificaciones de instancias de DCS

4.1 Especificaciones de la instancia de Redis 3.0 (descontinuado)

Esta sección describe las especificaciones de instancia de DCS para Redis 3.0, incluida la memoria total, la memoria disponible, el número máximo de conexiones permitidas, el ancho de banda máximo/asegurado y el rendimiento de referencia.

Las siguientes métricas están relacionadas con las especificaciones de instancia:

- Memoria usada: puede comprobar el uso de memoria de una instancia mediante la consulta de las métricas **Memory Usage** (Uso de memoria) y **Used Memory** (Memoria usada).
- Conexiones máximas: el número máximo de conexiones permitidas es el número máximo de clientes que se pueden conectar a una instancia. Para comprobar el número de conexiones a una instancia, consulte la métrica **Connected Clients** (Clientes conectados).
- QPS representa consultas por segundo, que es el número de comandos procesados por segundo.

NOTA

- Las instancias de DCS para Redis 3.0 están disponibles en los tipos de nodo único, principal/en standby y Clúster Proxy.
- Se ha descontinuado la venta de DCS for Redis 3.0. Puede utilizar DCS for Redis 4.0 o 5.0 en su lugar.

Instancias de nodo único

Para cada instancia de nodo único de DCS para Redis, la memoria disponible es menor que la memoria total porque alguna memoria está reservada para sobrecargas del sistema, como se muestra en **Tabla 4-1**.

Tabla 4-1 Especificaciones de instancias de nodo único de DCS compatibles con Redis 3.0

Memoria total (GB)	Memoria disponible (GB)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
2	1.5	5000/50,000	42/512	50,000	dcs.single_node
4	3.2	5000/50,000	64/1536	100,000	dcs.single_node
8	6.8	5000/50,000	64/1536	100,000	dcs.single_node
16	13.6	5000/50,000	85/3072	100,000	dcs.single_node
32	27.2	5000/50,000	85/3072	100,000	dcs.single_node
64	58.2	5000/60,000	128/5120	100,000	dcs.single_node

Instancia principal/en standby

Para cada instancia principal/en standby de DCS compatible con Redis, la memoria disponible es menor que la de una instancia de DCS compatible con Redis de nodo único porque alguna memoria está reservada para la persistencia de datos, como se muestra en **Tabla 4-2**. La memoria disponible de una instancia principal/en standby se puede ajustar para soportar tareas en segundo plano tales como persistencia de datos y sincronización principal/en standby.

Tabla 4-2 Especificaciones de instancias principal/en standby de DCS compatibles con Redis 3.0

Memoria total (GB)	Memoria disponible (GB)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
2	1.5	5000/50,000	42/512	50,000	dcs.master_standby
4	3.2	5000/50,000	64/1536	100,000	dcs.master_standby
8	6.4	5000/50,000	64/1536	100,000	dcs.master_standby
16	12.8	5000/50,000	85/3072	100,000	dcs.master_standby
32	25.6	5000/50,000	85/3072	100,000	dcs.master_standby
64	51.2	5000/60,000	128/5120	100,000	dcs.master_standby

Instancias de Clúster Proxy

Además de una mayor memoria, las instancias de clúster cuentan con más conexiones permitidas, mayor ancho de banda permitido y más QPS que las instancias de nodo único y principal/en standby.

Tabla 4-3 Especificaciones de instancias de Clúster Proxy de DCS compatibles con Redis 3.0

Memoria total (GB)	Memoria disponible (GB)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
64	64	90,000/90,000	600/5120	500,000	dcs.cluster
128	128	180,000/180,000	600/5120	500,000	dcs.cluster
256	256	240,000/240,000	600/5120	500,000	dcs.cluster
512	512	480,000/480,000	600/5120	500,000	dcs.cluster
1024	1024	960,000/960,000	600/5120	500,000	dcs.cluster

NOTA

- Las instancias de Clúster Proxy de DCS compatibles con Redis de pago por uso están disponibles en 64 GB, 128 GB y 256 GB.
- Las instancias de Clúster Proxy de DCS compatibles con Redis anuales/mensuales están disponibles en 64 GB, 128 GB, 256 GB, 512 GB y 1024 GB.

  Actualmente, solo la región CN-Hong Kong admite la facturación anual/mensual. Si necesita utilizar este modo de facturación en otras regiones, envíe un ticket de servicio en la consola para solicitar al personal técnico que active la función en el fondo.

4.2 Especificaciones de las instancias de Redis 4.0 y 5.0

Esta sección describe las especificaciones de instancia de DCS para Redis 4.0 y 5.0, incluida la memoria total, la memoria disponible, el número máximo de conexiones permitidas, el ancho de banda máximo/asegurado y el rendimiento de referencia.

Las siguientes métricas están relacionadas con las especificaciones de instancia:

- Memoria usada: puede comprobar el uso de memoria de una instancia mediante la consulta de las métricas **Memory Usage** (Uso de memoria) y **Used Memory** (Memoria usada).

- Conexiones máximas: el número máximo de conexiones permitidas es el número máximo de clientes que se pueden conectar a una instancia. Para comprobar el número de conexiones a una instancia, consulte la métrica **Connected Clients** (Clientes conectados).
- QPS representa consultas por segundo, que es el número de comandos procesados por segundo.
- Ancho de banda: Puede ver la métrica **Flow Control Times** (Tiempos de control de flujo) para comprobar si el ancho de banda ha excedido el límite. También puede comprobar la métrica **Bandwidth Usage** (uso de ancho de banda). Esta métrica es solo para referencia, ya que puede ser superior al 100%. Para obtener más información, consulte **¿Por qué el uso del ancho de banda supera el 100%?**

NOTA

- Las instancias de DCS compatibles con Redis 4.0 y 5.0 están disponibles en tipos de nodo único, principal/standby, Clúster Proxy, Clúster Redis y separación de lectura/escritura.
- Tanto DCS Redis 4.0 como 5.0 soportan las arquitecturas de CPU x86 y Arm. Para obtener más información sobre las diferencias, consulte la siguiente descripción.

Instancias de nodo único

Las instancias de nodo único de DCS compatibles con Redis 4.0 o 5.0 admiten las arquitecturas de CPU x86 y Arm. La siguiente tabla muestra las especificaciones.

Tabla 4-4 Especificaciones de instancias de nodo único de DCS compatibles con Redis 4.0 o 5.0

Memoria total (GB)	Memoria disponible (GB)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
0.125	0.125	10,000/10,000	40/40	80,000	x86: redis.single.xu1.tiny.128 Arm: redis.single.au1.tiny.128
0.25	0.25	10,000/10,000	80/80	80,000	x86: redis.single.xu1.tiny.256 Arm: redis.single.au1.tiny.256

Memoria total (GB)	Memoria disponible (GB)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
0.5	0.5	10,000/10,000	80/80	80,000	x86: redis.single.xu1.tiny.512 Arm: redis.single.au1.tiny.512
1	1	10,000/50,000	80/80	80,000	x86: redis.single.xu1.large.1 Arm: redis.single.au1.large.1
2	2	10,000/50,000	128/128	80,000	x86: redis.single.xu1.large.2 Arm: redis.single.au1.large.2
4	4	10,000/50,000	192/192	80,000	x86: redis.single.xu1.large.4 Arm: redis.single.au1.large.4
8	8	10,000/50,000	192/192	100,000	x86: redis.single.xu1.large.8 Arm: redis.single.au1.large.8
16	16	10,000/50,000	256/256	100,000	x86: redis.single.xu1.large.16 Arm: redis.single.au1.large.16

Memoria total (GB)	Memoria disponible (GB)	Máx. Conexiones (predeterminadas/ límite) (Recuento)	Ancho de banda máximo/ asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
24	24	10,000/50,000	256/256	100,000	x86: redis.single.xu1.large.24 Arm: redis.single.au1.large.24
32	32	10,000/50,000	256/256	100,000	x86: redis.single.xu1.large.32 Arm: redis.single.au1.large.32
48	48	10,000/50,000	256/256	100,000	x86: redis.single.xu1.large.48 Arm: redis.single.au1.large.48
64	64	10,000/50,000	384/384	100,000	x86: redis.single.xu1.large.64 Arm: redis.single.au1.large.64

Instancia principal/en standby

Las instancias principal/en standby admiten las arquitecturas de CPU x86 y Arm. Una instancia puede tener de 2 a 5 réplicas. Por ejemplo, las especificaciones de una instancia basada en Arm pueden ser **Arm | principal/en standby | 2 réplicas** o **Arm | principal/en standby | 5 réplicas**. De forma predeterminada, una instancia principal/en standby tiene un nodo principal y dos réplicas (incluida la réplica principal).

Dado el mismo tamaño de memoria, las diferencias entre las instancias principal/en standby basadas en x86, las instancias principal/en standby basadas en Arm y las instancias principal/en standby con múltiples réplicas son las siguientes:

- La memoria disponible, el número máximo de conexiones, el ancho de banda asegurado/ máximo y el QPS son los mismos.
- Nombre de especificación: **Tabla 4-5** solo muestra los nombres de especificación de instancias basadas en x86 y Arm. Los nombres de las especificaciones reflejan el número

de réplicas, por ejemplo, redis.ha.au1.large.**r2**.8 (principal/en standby | Arm | 2 réplicas | 8 GB) y redis.ha.au1.large.**r3**.8 (principal/en standby | Arm | 3 réplicas | 8 GB).

- Direcciones IP: Número de direcciones IP ocupadas = Número de nodos principales x Número de réplicas. Por ejemplo:

  2 réplicas: Número de direcciones IP ocupadas = 1 x 2 = 2

  3 réplicas: Número de direcciones IP ocupadas = 1 x 3 = 3

Tabla 4-5 Especificaciones de instancias principales/en standby de DCS para Redis 4.0 o 5.0

Memoria total (GB)	Memoria disponible (GB)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
0.125	0.125	10,000/10,000	40/40	80,000	x86: redis.ha.xu1.tiny.r2.128 Arm: redis.ha.au1.tiny.128
0.25	0.25	10,000/10,000	80/80	80,000	x86: redis.ha.xu1.tiny.r2.256 Arm: redis.ha.au1.tiny.256
0.5	0.5	10,000/10,000	80/80	80,000	x86: redis.ha.xu1.tiny.r2.512 Arm: redis.ha.au1.tiny.512
1	1	10,000/50,000	80/80	80,000	x86: redis.ha.xu1.large.r2.1 Arm: redis.ha.au1.large.1
2	2	10,000/50,000	128/128	80,000	x86: redis.ha.xu1.large.r2.2 Arm: redis.ha.au1.large.2

Memoria total (GB)	Memoria disponible (GB)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
4	4	10,000/50,000	192/192	80,000	x86: redis.ha.xu1.large.r2.4 Arm: redis.ha.au1.large.4
8	8	10,000/50,000	192/192	100,000	x86: redis.ha.xu1.large.r2.8 Arm: redis.ha.au1.large.8
16	16	10,000/50,000	256/256	100,000	x86: redis.ha.xu1.large.r2.16 Arm: redis.ha.au1.large.16
24	24	10,000/50,000	256/256	100,000	x86: redis.ha.xu1.large.r2.24 Arm: redis.ha.au1.large.24
32	32	10,000/50,000	256/256	100,000	x86: redis.ha.xu1.large.r2.32 Arm: redis.ha.au1.large.32
48	48	10,000/50,000	256/256	100,000	x86: redis.ha.xu1.large.r2.48 Arm: redis.ha.au1.large.48

Memoria total (GB)	Memoria disponible (GB)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
64	64	10,000/50,000	384/384	100,000	x86: redis.ha.xu1.large.r2.64 Arm: redis.ha.au1.large.64

Instancias de Clúster Proxy

Las instancias de Clúster Proxy soportan las arquitecturas de CPU x86 y Arm. **Tabla 4-6** enumera las especificaciones.

Las instancias de Clúster Proxy no admiten la personalización de particiones y réplicas. Para obtener más información sobre el número predeterminado de particiones, consulte **Tabla 3-2**. Por defecto, cada partición tiene dos réplicas.

Tabla 4-6 Especificaciones de instancias de Clúster Proxy de DCS compatibles con Redis 4.0 y 5.0

Especificaciones (GB)	Memoria disponible (GB)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
4	4	20,000/20,000	1,000/1,000	240,000	x86: redis.proxy.xu1.large.4 Arm: redis.proxy.au1.large.4
8	8	30,000/30,000	2,000/2,000	240,000	x86: redis.proxy.xu1.large.8 Arm: redis.proxy.au1.large.8

Especificaciones (GB)	Memoria disponible (GB)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
16	16	30,000/30,000	3,072/3,072	240,000	x86: redis.proxy.xu1.large.16 Arm: redis.proxy.au1.large.16
24	24	30,000/30,000	3,072/3,072	240,000	x86: redis.proxy.xu1.large.24 Arm: redis.proxy.au1.large.24
32	32	30,000/30,000	3,072/3,072	240,000	x86: redis.proxy.xu1.large.32 Arm: redis.proxy.au1.large.32
48	48	60,000/60,000	4608/4608	480,000	x86: redis.proxy.xu1.large.48 Arm: redis.proxy.au1.large.48
64	64	80,000/80,000	6144/6144	640,000	x86: redis.proxy.xu1.large.64 Arm: redis.proxy.au1.large.64
96	96	120,000/120,000	9216/9216	960,000	x86: redis.proxy.xu1.large.96 Arm: redis.proxy.au1.large.96

Especificaciones (GB)	Memoria disponible (GB)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
128	128	160,000/160,000	10,000/10,000	1,280,000	x86: redis.proxy.xu1.large.128 Arm: redis.proxy.au1.large.128
192	192	240,000/240,000	10,000/10,000	1,920,000	x86: redis.proxy.xu1.large.192 Arm: redis.proxy.au1.large.192
256	256	320,000/320,000	10,000/10,000	> 2,000,000	x86: redis.proxy.xu1.large.256 Arm: redis.proxy.au1.large.256
384	384	480,000/480,000	10,000/10,000	> 2,000,000	x86: redis.proxy.xu1.large.384 Arm: redis.proxy.au1.large.384
512	512	640,000/640,000	10,000/10,000	> 2,000,000	x86: redis.proxy.xu1.large.512 Arm: redis.proxy.au1.large.512
768	768	640,000/640,000	10,000/10,000	> 2,000,000	x86: redis.proxy.xu1.large.768 Arm: redis.proxy.au1.large.768

Especificaciones (GB)	Memoria disponible (GB)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
1024	1024	640,000/640,000	10,000/10,000	> 2,000,000	x86: redis.proxy.xu1.large.1024 Arm: redis.proxy.au1.large.1024

Instancias de Clúster Redis

Las instancias de Clúster Redis soportan las arquitecturas de CPU x86 y Arm. Cada instancia puede tener de 1 a 5 réplicas. Por ejemplo, las especificaciones de instancia pueden ser **Clúster Redis | 1 réplica** o **Clúster Redis | 5 réplicas**. De forma predeterminada, una instancia de Clúster Redis tiene dos réplicas. Una instancia de Clúster Redis con solo 1 réplica indica que se ha reducido la cantidad de réplicas.

Dado el mismo tamaño de memoria, las diferencias entre las instancias Clúster Redis basadas en x86, las instancias Clúster Redis basadas en Arm y las instancias Clúster Redis con múltiples réplicas son las siguientes:

- La memoria disponible, la cantidad de partición (cantidad de nodo principal), el número máximo de conexiones, el ancho de banda asegurado/máximo y el QPS son los mismos.

  NOTA

  El ancho de banda máximo y el ancho de banda asegurado de un Clúster Redis es para toda la instancia, y no para una sola partición.

- Nombre de especificación: **Tabla 4-7** solo muestra los nombres de especificación de instancias basadas en x86 y Arm con 2 réplicas. Los nombres de las especificaciones reflejan el número de réplicas, por ejemplo, redis.cluster.au1.large.**r2**.24 (Clúster Redis | Arm | 2 réplicas | 24 GB) y redis.cluster.au1.large.**r3**.24 (Clúster Redis | Arm | 3 réplicas | 24 GB).
- Direcciones IP: Número de direcciones IP ocupadas = Número de particiones x Número de réplicas. Por ejemplo:

  24 GB | Clúster Redis | 3 réplicas: Número de direcciones IP ocupadas = 3 x 3 = 9
- Memoria disponible por nodo = Memoria disponible de instancia/cantidad de nodo principal

  Por ejemplo, una instancia basada en x86 de 24 GB tiene 24 GB de memoria disponible y 3 nodos principales. La memoria disponible por nodo es de 24/3 = 8 GB.
- Límite máximo de conexiones por nodo = Límite máximo de conexiones/cantidad de nodo principal Por ejemplo:

Por ejemplo, una instancia basada en x86 de 24 GB tiene 3 nodos principals y el límite máximo de conexiones es de 150,000. El límite máximo de conexiones por nodo = 150,000/3 = 50,000.

Tabla 4-7 Especificaciones de instancias de Clúster Redis de DCS compatible con Redis 4.0 y 5.0

Memoria total (GB)	Memoria disponible (GB)	Particiones (nodos principales)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
4	4	3	30,000 / 150,000	2304/2304	240,000	x86: redis.cluster.xu1.large.r2.4 Arm: redis.cluster.au1.large.r2.4
8	8	3	30,000 / 150,000	2304/2304	240,000	x86: redis.cluster.xu1.large.r2.8 Arm: redis.cluster.au1.large.r2.8
16	16	3	30,000 / 150,000	2304/2304	240,000	x86: redis.cluster.xu1.large.r2.16 Arm: redis.cluster.au1.large.r2.16
24	24	3	30,000 / 150,000	2304/2304	300,000	x86: redis.cluster.xu1.large.r2.24 Arm: redis.cluster.au1.large.r2.24
32	32	3	30,000 / 150,000	2304/2304	300,000	x86: redis.cluster.xu1.large.r2.32 Arm: redis.cluster.au1.large.r2.32

Memoria total (GB)	Memoria disponible (GB)	Particiones (nodos principales)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
48	48	6	60,000 / 300,000	4608/4608	> 300,000	x86: redis.cluster.xu1.large.r2.48 Arm: redis.cluster.au1.large.r2.48
64	64	8	80,000 / 400,000	6144/6144	500,000	x86: redis.cluster.xu1.large.r2.64 Arm: redis.cluster.au1.large.r2.64
96	96	12	120,000 / 600,000	9216/9216	> 500,000	x86: redis.cluster.xu1.large.r2.96 Arm: redis.cluster.au1.large.r2.96
128	128	16	160,000 / 800,000	12,288/12,288	1,000,000	x86: redis.cluster.xu1.large.r2.128 Arm: redis.cluster.au1.large.r2.128
192	192	24	240,000 / 1,200,000	18,432/18,432	> 1,000,000	x86: redis.cluster.xu1.large.r2.192 Arm: redis.cluster.au1.large.r2.192
256	256	32	320,000 / 1,600,000	24,576/24,576	> 2,000,000	x86: redis.cluster.xu1.large.r2.256 Arm: redis.cluster.au1.large.r2.256

Memoria total (GB)	Memoria disponible (GB)	Particiones (nodos principales)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
384	384	48	480,000 / 2,400,000	36,864/36,864	> 2,000,000	x86: redis.cluster.xu1.large.r2.384 Arm: redis.cluster.au1.large.r2.384
512	512	64	640,000 / 3,200,000	49,152/49,152	> 2,000,000	x86: redis.cluster.xu1.large.r2.512 Arm: redis.cluster.au1.large.r2.512
768	768	96	960,000 / 4,800,000	73,728/73,728	> 2,000,000	x86: redis.cluster.xu1.large.r2.768 Arm: redis.cluster.au1.large.r2.768
1024	1024	128	1,280,000 / 6,400,000	98,304/98,304	> 2,000,000	x86: redis.cluster.xu1.large.r2.1024 Arm: redis.cluster.au1.large.r2.1024

Instancias de separación de lectura/escritura

- Actualmente, las instancias de separación de lectura/escritura solo admiten la arquitectura de CPU x86. **Tabla 4-8** enumera las especificaciones.
- No se puede modificar el número máximo de conexiones de una instancia de separación de lectura/escritura de DCS compatible con Redis 4.0 o 5.0.
- Límite de ancho de banda por servidor Redis (MB/s) = Límite de ancho de banda total (MB/s)/Número de réplicas (incluidos los principales)
- Performance de referencia por nodo (QPS) = Performance de referencia (QPS)/Número de réplicas (incluidos los principales)
- Cuando utilice instancias de separación de lectura/escritura, tenga en cuenta lo siguiente:

a. Las solicitudes de lectura se envían a réplicas. Hay un retraso cuando los datos se sincronizan desde el principal a las réplicas.

   Si sus servicios son sensibles al retraso, no utilice instancias de separación de lectura/escritura. En su lugar, puede usar instancias de principal/en standby o de clúster.

b. La separación de lectura/escritura es adecuada cuando hay más solicitudes de lectura que solicitudes de escritura. Si hay muchas solicitudes de escritura, el principal y las réplicas pueden desconectarse, o la sincronización de datos entre ellas puede fallar después de la desconexión. Como resultado, el rendimiento de lectura se deteriora.

   Si sus servicios son pesados en escritura, use instancias principal/en standby o de clúster.

c. Si una réplica es defectuosa, toma algún tiempo sincronizar todos los datos del principal. Durante la sincronización, la réplica no proporciona servicios y el rendimiento de lectura de la instancia se deteriora.

   Para reducir el impacto de la interrupción, utilice una instancia con menos de 32 GB de memoria. Cuanto menor sea la memoria, menor será el tiempo para la sincronización completa de datos entre el principal y las réplicas, y menor será el impacto de la interrupción.

Tabla 4-8 Especificaciones de las instancias de separación de lectura/escritura de DCS compatibles con Redis 4.0 o 5.0

Especificaciones	Memoria disponible (GB)	Réplicas (incluidos los principales)	Máx.	Límite de ancho de banda (MB/s)	Límite de ancho de banda por servidor Redis (MB/s)	Rendimiento de referencia (QPS)	Rendimiento de referencia por nodo (QPS)	Código de especificación (spec_code en la API)
8	8	2	20,000	192	96	160,000	80,000	redis.ha.xu1.large.p2.8
8	8	3	30,000	288	96	240,000	80,000	redis.ha.xu1.large.p3.8
8	8	4	40,000	384	96	320,000	80,000	redis.ha.xu1.large.p4.8
8	8	5	50,000	480	96	400,000	80,000	redis.ha.xu1.large.p5.8

Especificaciones	Memoria disponible (GB)	Réplicas (incluidos los principales)	Máx.	Límite de ancho de banda (MB/s)	Límite de ancho de banda por servidor Redis (MB/s)	Rendimiento de referencia (QPS)	Rendimiento de referencia por nodo (QPS)	Código de especificación (spec_code en la API)
8	8	6	60,000	576	96	480,000	80,000	redis.ha.xu1.large.p6.8
16	16	2	20,000	192	96	160,000	80,000	redis.ha.xu1.large.p2.16
16	16	3	30,000	288	96	240,000	80,000	redis.ha.xu1.large.p3.16
16	16	4	40,000	384	96	320,000	80,000	redis.ha.xu1.large.p4.16
16	16	5	50,000	480	96	400,000	80,000	redis.ha.xu1.large.p5.16
16	16	6	60,000	576	96	480,000	80,000	redis.ha.xu1.large.p6.16
32	32	2	20,000	192	96	160,000	80,000	redis.ha.xu1.large.p2.32
32	32	3	30,000	288	96	240,000	80,000	redis.ha.xu1.large.p3.32
32	32	4	40,000	384	96	320,000	80,000	redis.ha.xu1.large.p4.32

Especificaciones	Memoria disponible (GB)	Réplicas (incluidos los principales)	Máx.	Límite de ancho de banda (MB/s)	Límite de ancho de banda por servidor Redis (MB/s)	Rendimiento de referencia (QPS)	Rendimiento de referencia por nodo (QPS)	Código de especificación (spec_code en la API)
32	32	5	50,000	480	96	400,000	80,000	redis.ha.xu1.large.p5.32
32	32	6	60,000	576	96	480,000	80,000	redis.ha.xu1.large.p6.32

4.3 Especificaciones de la instancia de Redis 6.0 (Prueba beta abierta)

Esta sección describe las especificaciones de instancia de DCS para Redis 6.0, incluida la memoria total, la memoria disponible, el número máximo de conexiones permitidas, el ancho de banda máximo/asegurado y el rendimiento de referencia.

Las siguientes métricas están relacionadas con las especificaciones de instancia:

- Memoria usada: puede comprobar el uso de memoria de una instancia mediante la consulta de las métricas **Memory Usage** (Uso de memoria) y **Used Memory** (Memoria usada).
- Conexiones máximas: el número máximo de conexiones permitidas es el número máximo de clientes que se pueden conectar a una instancia. Para comprobar el número de conexiones a una instancia, consulte la métrica **Connected Clients** (Clientes conectados).
- QPS representa consultas por segundo, que es el número de comandos procesados por segundo. Para obtener más información sobre las pruebas de QPS, consulte el **Libro Blanco sobre el rendimiento**.

DCS for Redis 6.0 viene en ediciones básicas, empresariales (rendimiento) y empresariales (almacenamiento de información). Están disponibles para la prueba beta abierta en la región CN North-Beijing4.

Edición básica

Actualmente, la edición básica de DCS for Redis 6.0 admite las instancias principales/en standby y de nodo único basadas en CPU x86.

Tabla 4-9 Especificaciones de instancias principales/en standby de la edición básica de DCS compatible con Redis 6.0

Memoria total (GB)	Memoria disponible (GB)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
4	4	10,000/50,000	192/192	280,000	redis.ha.xu1.large.r2.v6.4
8	8	10,000/50,000	192/192	300,000	redis.ha.xu1.large.r2.v6.8
16	16	10,000/50,000	256/256	300,000	redis.ha.xu1.large.r2.v6.16

Tabla 4-10 Especificaciones de instancias de nodo único de la edición básica de DCS compatible con Redis 6.0

Memoria total (GB)	Memoria disponible (GB)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
4	4	10,000/50,000	192/192	280,000	redis.single.xu1.large.v6.4
8	8	10,000/50,000	192/192	300,000	redis.single.xu1.large.v6.8
16	16	10,000/50,000	256/256	300,000	redis.single.xu1.large.v6.16

Edición profesional (Rendimiento)

Actualmente, la edición profesional (rendimiento) de DCS for Redis 6.0 admite instancias principales/en standby basadas en CPU x86.

Tabla 4-11 Especificaciones de instancias de edición profesional (rendimiento) de DCS compatible con Redis 6.0

Memoria total (GB)	Memoria disponible (GB)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
4	4	10,000/50,000	768/768	200,000	redis.ha.xu1.large.enthp.4
8	8	10,000/50,000	1536/1536	400,000	redis.ha.xu1.large.enthp.8
16	16	10,000/50,000	1536/1536	400,000	redis.ha.xu1.large.enthp.16
32	32	10,000/50,000	1536/1536	400,000	redis.ha.xu1.large.enthp.32
64	64	10,000/50,000	1536/1536	400,000	redis.ha.xu1.large.enthp.64

Edición profesional (almacenamiento)

Actualmente, la edición profesional (almacenamiento) de DCS for Redis 6.0 admite instancias principales/en standby basadas en CPU x86.

Tabla 4-12 Especificaciones de instancias de edición profesional (almacenamiento) de DCS compatible con Redis 6.0

Memoria total (GB)	Almacenamiento máximo (GB)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
8	64	10,000/50,000	768/768	70,000	redis.ha.xu1.large.entst.8
16	128	10,000/50,000	768/768	70,000	redis.ha.xu1.large.entst.16
32	256	10,000/50,000	768/768	70,000	redis.ha.xu1.large.entst.32

4.4 Especificaciones de instancias de Memcached (no disponibles pronto)

NOTA

DCS for Memcached está a punto de no estar disponible y ya no se vende en algunas regiones. Puede utilizar DCS for Redis 4.0 o 5.0 en su lugar.

Esta sección describe las especificaciones de instancia de DCS para Memcached, incluida la memoria total, la memoria disponible, el número máximo de conexiones permitidas, el ancho de banda máximo/asegurado y el rendimiento de referencia.

El número máximo de conexiones permitidas es el número máximo de clientes conectados a una instancia. Para comprobar el número de conexiones a una instancia, consulte la métrica **Connected Clients** (Clientes conectados).

QPS representa consultas por segundo, que es el número de comandos procesados por segundo.

NOTA

Las instancias de DCS Memcached están disponibles en tipos de nodo único y principal/en standby.

Instancias de nodo único

Para cada instancia de nodo único de DCS para Memcached, la memoria disponible es menor que la memoria total porque alguna memoria está reservada para sobrecargas del sistema, como se muestra en **Tabla 4-13**.

Tabla 4-13 Especificaciones de instancias de nodo único de DCS compatible con Memcached

Memoria total (GB)	Memoria disponible (GB)	Máx. Conexiones (predeterminadas/límite) (Recuento)	Ancho de banda máximo/asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
2	1.5	5,000/50,000	42/128	50,000	dcs.memcached.single_node
4	3.2	5,000/50,000	64/192	100,000	dcs.memcached.single_node
8	6.8	5,000/50,000	64/192	100,000	dcs.memcached.single_node
16	13.6	5,000/50,000	85/256	100,000	dcs.memcached.single_node
32	27.2	5,000/50,000	85/256	100,000	dcs.memcached.single_node

Memoria total (GB)	Memoria disponible (GB)	Máx. Conexiones (predeterminadas/ límite) (Recuento)	Ancho de banda máximo/ asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
64	58.2	5,000/50,000	128/384	100,000	dcs.memcached.single_node

Instancia principal/en standby

Para cada instancia principal/standby de DCS compatible con Memcached, la memoria disponible es menor que la memoria total porque cierta memoria está reservada para la persistencia de datos, como se muestra en **Tabla 4-14**. La memoria disponible de una instancia principal/en standby se puede ajustar para soportar tareas en segundo plano tales como persistencia de datos y sincronización principal/en standby.

Tabla 4-14 Especificaciones de instancias principal/en standby de DCS compatible con Memcached

Memoria total (GB)	Memoria disponible (GB)	Máx. Conexiones (predeterminadas/ límite) (Recuento)	Ancho de banda máximo/ asegurado (Mbit/s)	Rendimiento de referencia (QPS)	Código de especificación (spec_code en la API)
2	1.5	5,000/50,000	42/128	50,000	dcs.memcached.master_standby
4	3.2	5,000/50,000	64/192	100,000	dcs.memcached.master_standby
8	6.8	5,000/50,000	64/192	100,000	dcs.memcached.master_standby
16	13.6	5,000/50,000	85/256	100,000	dcs.memcached.master_standby
32	27.2	5,000/50,000	85/256	100,000	dcs.memcached.master_standby
64	58.2	5,000/50,000	128/384	100,000	dcs.memcached.master_standby

5 Compatibilidad de los comandos

5.1 Comandos de Redis 3.0

DCS for Redis 3.0 está desarrollado basado en Redis 3.0.7 y es compatible con los protocolos y comandos de código abierto. Esta sección describe la compatibilidad de DCS for Redis 3.0 con los comandos de Redis, incluidos los comandos compatibles, los comandos deshabilitados, las secuencias de comandos y los comandos no compatibles de versiones posteriores de Redis y las restricciones en el uso de los comandos.

NOTA

DCS for Redis 3.0 ya no se proporciona. Puede utilizar DCS for Redis 4.0 o 5.0 en su lugar.

Las instancias de DCS para Redis admiten la mayoría de los comandos de Redis. Cualquier cliente compatible con el protocolo de Redis puede acceder a DCS.

- Por motivos de seguridad, algunos comandos de Redis están deshabilitados en DCS, como se indica en **Comandos deshabilitados por DCS for Redis 3.0**.
- Algunos comandos de Redis son compatibles con instancias DCS de clúster para operaciones de varias claves en la misma ranura. Para más detalles, consulte **Restricciones de comandos**.
- Algunos comandos de Redis tienen restricciones de uso, que se describen en **Otras restricciones del uso de comandos**.

Comandos compatibles con DCS for Redis 3.0

A continuación se enumeran los comandos compatibles con DCS for Redis 3.0. Para obtener más información sobre la sintaxis del comando, visite el **sitio web oficial de Redis**. Por ejemplo, para ver detalles sobre el comando SCAN, escriba SCAN en el cuadro de búsqueda de **esta página**.

NOTA

- Los comandos disponibles desde las versiones posteriores de Redis no son compatibles con las instancias de versiones anteriores. Ejecute un comando en redis-cli para comprobar si es compatible con DCS for Redis. Si se devuelve el mensaje "(error) ERR unknown command", el comando no es compatible.
- Las instancias de Clúster Proxy no admiten los siguientes comandos enumerados en las tablas:
  - Grupo de **List**: **BLPOP**, **BRPOP**, y **BRPOPLRUSH**
  - Los comandos de **CLIENT** en el grupo de **Server**: **CLIENT KILL**, **CLIENT GETNAME**, **CLIENT LIST**, **CLIENT SETNAME**, **CLIENT PAUSE**, y **CLIENT REPLY**.
  - Grupo de **Server**: **MONITOR**
  - Grupo de **Transactions**: **UNWATCH** y **WATCH**
  - Grupo de **Key**: **RANDOMKEY** (para las instancias antiguas)

Tabla 5-1 Comandos soportados por instancias de DCS para Redis 3.0 (1)

Keys	String	Hash	List	Set	Sorted Set	Server
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOUNT	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	HGET	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE	KEYS
PTTL	GET	HKEYS	LPOP	SISMEMBER	ZRANK	CLIENT KILL
RANDOMKEY	GETRANGE	HMGET	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	CLIENT LIST
RENAME	GETSET	HMSET	LRANGE	SMOVE	ZREMRANGEBYCORE	CLIENT GETNAME
RENAMENX	INCR	HSET	LREM	SPOP	ZREVRANGE	CLIENT SETNAME
RESTORE	INCRBY	HSETNX	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	CONFIG GET
SORT	INCRBYFLOAT	HVALS	LTRIM	SREM	ZREVRANK	MONITOR

Keys	String	Hash	List	Set	Sorted Set	Server
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	SLOWLOG
TYPE	MSET	-	RPOPLPU	SUNIONSTORE	ZUNIONSTORE	ROLE
SCAN	MSETNX	-	RPOPLPUSH	SSCAN	ZINTERSTORE	-
OBJECT	PSETEX	-	RPUSH	-	ZSCAN	-
-	SET	-	RPUSHX	-	ZRANGEBYLEX	-
-	SETBIT	-	-	-	-	-
-	SETEX	-	-	-	-	-
-	SETNX	-	-	-	-	-
-	SETRANGE	-	-	-	-	-
-	STRLEN	-	-	-	-	-

Tabla 5-2 Comandos soportados por instancias de DCS para Redis 3.0 (2)

HyperLogLog	Pub/Sub	Transactions	Connection	Scripting	Geo
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS
-	PUNSUBSCRIBE	UNWATCH	QUIT	SCRIPT FLUSH	GEODIST
-	SUBSCRIBE	WATCH	SELECT	SCRIPT KILL	GEORADIUS
-	UNSUBSCRIBE	-	-	SCRIPT LOAD	GEORADIUSBYMEMBER

Comandos deshabilitados por DCS for Redis 3.0

A continuación se enumeran los comandos deshabilitados por DCS for Redis 3.0.

Tabla 5-3 Comandos de Redis desactivados en las instancias de DCS compatibles con Redis 3.0 de nodo único y principal/en standby

Keys	Server
MIGRATE	SLAVEOF
-	SHUTDOWN
-	LASTSAVE
-	DEBUG commands
-	COMMAND
-	SAVE
-	BGSAVE
-	BGREWRITEAOF

Tabla 5-4 Comandos de Redis deshabilitados en instancias de Clúster Proxy de DCS compatibles con Redis 3.0

Keys	Server	List	Transactions	Connection	Cluster	codis
MIGRATE	SLAVEOF	BLPOP	DISCARD	SELECT	CLUSTER	TIME
MOVE	SHUTDOWN	BRPOP	EXEC	-	-	SLOTSINFO
-	LASTSAVE	BRPOPLPUSH	MULTI	-	-	SLOTSDEL
-	DEBUG commands	-	UNWATCH	-	-	SLOTSMGRTSLOT
-	COMMAND	-	WATCH	-	-	SLOTSMGRTONE
-	SAVE	-	-	-	-	SLOTSCHECK
-	BGSAVE	-	-	-	-	SLOTSMGRTTAGSLOT
-	BGREWRITEAOF	-	-	-	-	SLOTSMGRTTAGONE
-	SYNC	-	-	-	-	-
-	PSYNC	-	-	-	-	-

Keys	Server	List	Transactions	Connection	Cluster	codis
-	MONITOR	-	-	-	-	-
-	CLIENT commands	-	-	-	-	-
-	OBJECT	-	-	-	-	-
-	ROLE	-	-	-	-	-

5.2 Comandos de Redis 4.0

DCS for Redis 4.0 está desarrollado basado en Redis 4.0.14 y es compatible con los protocolos y comandos de código abierto. Esta sección describe la compatibilidad de DCS for Redis 4.0 con los comandos de Redis, incluidos los comandos compatibles y deshabilitados.

Las instancias de DCS para Redis admiten la mayoría de los comandos de Redis. Cualquier cliente compatible con el protocolo de Redis puede acceder a DCS.

- Por motivos de seguridad, algunos comandos de Redis están deshabilitados en DCS, como se indica en **Comandos deshabilitados por DCS for Redis 4.0**.
- Algunos comandos de Redis son compatibles con instancias DCS de clúster para operaciones de varias claves en la misma ranura. Para más detalles, consulte **Restricciones de comandos**.
- Algunos comandos de Redis tienen restricciones de uso, que se describen en **Otras restricciones del uso de comandos**.

Comandos compatibles con DCS for Redis 4.0

- **Tabla 5-5** y **Tabla 5-6** enumeran los comandos de Redis admitidos por las instancias de DCS para Redis 4.0.
- **Tabla 5-7** y **Tabla 5-8** enumeran los comandos de Redis soportados por las instancias de Clúster Proxy de DCS para Redis 4.0.
- **Tabla 5-9** y **Tabla 5-10** enumeran los comandos de Redis admitidos por la separación de lectura/escritura de instancias de DCS para Redis 4.0.

Para obtener más información sobre la sintaxis del comando, visite el **sitio web oficial de Redis**. Por ejemplo, para ver detalles sobre el comando SCAN, escriba SCAN en el cuadro de búsqueda de **esta página**.

NOTA

- Los comandos disponibles desde las versiones posteriores de Redis no son compatibles con las instancias de versiones anteriores. Ejecute un comando en redis-cli para comprobar si es compatible con DCS for Redis. Si se devuelve el mensaje "(error) ERR unknown command", el comando no es compatible.
- Para las instancias de DCS para Redis 4.0 en el modo Clúster Redis, asegúrese de que todos los comandos de una canalización se ejecuten en la misma partición.

Tabla 5-5 Comandos soportados por instancias de DCS para Redis 4.0 (1)

Keys	String	Hash	List	Set	Sorted Set	Server
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOUNT	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	HGET	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE	KEYS
PTTL	GET	HKEYS	LPOP	SISMEMBER	ZRANK	CLIENTKILL
RANDOMKEY	GETRANGE	HMGET	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	CLIENTLIST
RENAME	GETSET	HMSET	LRANGE	SMOVE	ZREMRANGEBYCORE	CLIENTGETNAME
RENAMENX	INCR	HSET	LREM	SPOP	ZREVRANGE	CLIENTSETNAME
RESTORE	INCRBY	HSETNX	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	CONFIGGET
SORT	INCRBYFLOAT	HVALS	LTRIM	SREM	ZREVRANK	MONITOR
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	SLOWLOG
TYPE	MSET	HSTRLEN	RPOPLPU	SUNIONSTORE	ZUNIONSTORE	ROLE
SCAN	MSETNX	HLEN	RPOPLPUSH	SSCAN	ZINTERSTORE	SWAPDB
OBJECT	PSETEX	-	RPUSH	SPOP	ZSCAN	MEMORY
PEXPIRE	SET	-	RPUSHX	-	ZRANGEBYLEX	CONFIG
PEXPIREAT	SETBIT	-	LPUSH	-	ZLEXCOUNT	-

Keys	String	Hash	List	Set	Sorted Set	Server
-	SETEX	-	-	-	ZREMRANGEBYSCORE	-
-	SETNX	-	-	-	ZREM	-
-	SETRANGE	-	-	-	-	-
-	STRLEN	-	-	-	-	-
-	BITFIELD	-	-	-	-	-

Tabla 5-6 Comandos soportados por instancias de DCS para Redis 4.0 (2)

HyperLogLog	Pub/Sub	Transactions	Connection	Scripting	Geo
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS
-	PUNSUBSCRIBE	UNWATCH	QUIT	SCRIPT FLUSH	GEODIST
-	SUBSCRIBE	WATCH	SELECT	SCRIPT KILL	GEORADIUS
-	UNSUBSCRIBE	-	-	SCRIPT LOAD	GEORADIUSBYMEMBER

Tabla 5-7 Comandos soportados por instancias de Clúster Proxy de DCS para Redis 4.0 (1)

Keys	String	Hash	List	Set	Sorted Set	Server
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOUNT	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	HGET	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY	TIME

Keys	String	Hash	List	Set	Sorted Set	Server
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE	ROLE
PTTL	GET	HKEYS	LPOP	SISMEMBER	ZRANK	MEMORY
RENAME	GETRANGE	HMGET	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	COMMAND
RENAMENX	GETSET	HMSET	LRANGE	SMOVE	ZREMRANGEBYCORE	COMMAND COUNT
RESTORE	INCR	HSET	LREM	SPOP	ZREVRANGE	COMMAND GETKEYS
SORT	INCRBY	HSETNX	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	COMMAND INFO
TTL	INCRBYFLOAT	HVALS	LTRIM	SREM	ZREVRANK	CONFIG GET
TYPE	MGET	HSCAN	RPOP	SUNION	ZSCORE	CONFIG RESETSTAT
SCAN	MSET	HSTRLEN	RPOPLPUSH	SUNIONSTORE	ZUNIONSTORE	CONFIG REWRITE
OBJECT	MSETNX	HLEN	RPUSH	SSCAN	ZINTERSTORE	CONFIG SET
PEXPIRE	PSETEX	HKEYS	RPUSHX	SPOP	ZSCAN	-
PEXPIREAT	SET	-	LPUSH	-	ZRANGEBYLEX	-
EXPIREAT	SETBIT	-	-	-	ZLEXCOUNT	-
KEYS	SETEX	-	-	-	ZREMRANGEBYSCORE	-
TOUCH	SETNX	-	-	-	ZREM	-

Keys	String	Hash	List	Set	Sorted Set	Server
UNLINK	SETRANGE	-	-	-	ZREMRANGEBYLEX	-
-	STRLEN	-	-	-	ZREVRANGEBYLEX	-
-	BITFIELD	-	-	-	-	-
-	GETBIT	-	-	-	-	-

Tabla 5-8 Comandos soportados por instancias de Clúster Proxy de DCS para Redis 4.0 (2)

HyperLogLog	Pub/Sub	Transactions	Connection	Scripting	Geo
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS
-	PUNSUBSCRIBE	UNWATCH	QUIT	SCRIPT FLUSH	GEODIST
-	SUBSCRIBE	WATCH	SELECT	SCRIPT KILL	GEORADIUS
-	UNSUBSCRIBE	-	CLIENT KILL	SCRIPT LOAD	GEORADIUSBYMEMBER
-	-	-	CLIENT LIST	SCRIPT DEBUG YES|SYNC|NO	GEOSEARCH
-	-	-	CLIENT GETNAME	-	GEOSEARCHSTORE
-	-	-	CLIENT SETNAME	-	-

Tabla 5-9 Comandos soportados por la separación de lectura/escritura de instancias de DCS para Redis 4.0 (1)

Keys	String	Hash	List	Set	Sorted Set	Server
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOUNT	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	HGET	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE	MONITOR
PTTL	GET	HKEYS	LPOP	SISMEMBER	ZRANK	SLOWLOG
RANDOMKEY	GETRANGE	HMGET	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	ROLE
RENAME	GETSET	HMSET	LRANGE	SMOVE	ZREMRANGEBYCORE	SWAPDB
RENAMENX	INCR	HSET	LREM	SPOP	ZREVRANGE	MEMORY
RESTORE	INCRBY	HSETNX	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	COMMAND
SORT	INCRBYFLOAT	HVALS	LTRIM	SREM	ZREVRANK	COMMAND COUNT
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	COMMAND GETKEYS
TYPE	MSET	HSTRLEN	RPOPLPUSH	SUNIONSTORE	ZUNIONSTORE	COMMAND INFO
SCAN	MSETNX	HLEN	RPUSH	SSCAN	ZINTERSTORE	CONFIG GET

Keys	String	Hash	List	Set	Sorted Set	Server
OBJECT	PSETEX	-	RPUSHX	SPOP	ZSCAN	CONFIG RESETSTAT
PEXPIRE	SET	-	LPUSH	-	ZRANGEBYLEX	CONFIG REWRITE
PEXPIREAT	SETBIT	-	-	-	ZLEXCOUNT	CONFIG SET
EXPIREAT	SETEX	-	-	-	ZREMRANGEBYSCORE	-
KEYS	SETNX	-	-	-	ZREM	-
TOUCH	SETRANGE	-	-	-	ZREMRANGEBYLEX	-
UNLINK	STRLEN	-	-	-	ZREVRANGEBYLEX	-
-	BITFIELD	-	-	-	-	-
-	GETBIT	-	-	-	-	-

Tabla 5-10 Comandos soportados por la separación de lectura/escritura de instancias de DCS para Redis 4.0 (2)

HyperLogLog	Pub/Sub	Transactions	Connection	Scripting	Geo
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS
-	PUNSUBSCRIBE	UNWATCH	QUIT	SCRIPT FLUSH	GEODIST
-	SUBSCRIBE	WATCH	SELECT	SCRIPT KILL	GEORADIUS
-	UNSUBSCRIBE	-	CLIENT KILL	SCRIPT LOAD	GEORADIUSBYMEMBER

HyperLoglog	Pub/Sub	Transactions	Connection	Scripting	Geo
-	-	-	CLIENT LIST	SCRIPT DEBUG YES|SYNC|NO	GEOSEARCH
-	-	-	CLIENT GETNAME	-	GEOSEARCHSTORE
-	-	-	CLIENT SETNAME	-	-

Comandos deshabilitados por DCS for Redis 4.0

A continuación se enumeran los comandos deshabilitados por DCS for Redis 4.0.

Tabla 5-11 Comandos de Redis desactivados en las instancias compatibles con Redis 4.0 de nodo único y principal/en standby

Keys	Server
MIGRATE	SLAVEOF
-	SHUTDOWN
-	LASTSAVE
-	DEBUG commands
-	COMMAND
-	SAVE
-	BGSAVE
-	BGREWRITEAOF
-	SYNC
-	PSYNC

Tabla 5-12 Comandos de Redis deshabilitados en instancias de Clúster Proxy de DCS compatibles con Redis 4.0

Keys	Server	Sorted Set	Cluster
MIGRATE	BGREWRITEAOF	BZPOPMAX	READONLY

Keys	Server	Sorted Set	Cluster
MOVE	BGSAVE	BZPOPMIN	READWRITE
RANDOMKEY	CLIENT commands	ZPOPMAX	-
WAIT	DEBUG OBJECT	ZPOPMIN	-
-	DEBUG SEGFAULT	-	-
-	LASTSAVE	-	-
-	PSYNC	-	-
-	SAVE	-	-
-	SHUTDOWN	-	-
-	SLAVEOF	-	-
-	LATENCY commands	-	-
-	MODULE commands	-	-
-	LOLWUT	-	-
-	SWAPDB	-	-
-	REPLICAOF	-	-
-	SYNC	-	-

Tabla 5-13 Comandos de Redis deshabilitados en instancias de Clúster Redis de DCS compatibles con Redis 4.0

Keys	Server	Cluster
MIGRATE	SLAVEOF	CLUSTER MEET
-	SHUTDOWN	CLUSTER FLUSHSLOTS
-	LASTSAVE	CLUSTER ADDSLOTS
-	DEBUG commands	CLUSTER DELSLOTS
-	COMMAND	CLUSTER SETSLOT
-	SAVE	CLUSTER BUMPEPOCH
-	BGSAVE	CLUSTER SAVECONFIG
-	BGREWRITEAOF	CLUSTER FORGET
-	SYNC	CLUSTER REPLICATE
-	PSYNC	CLUSTER COUNT-FAILURE-REPORTS

Keys	Server	Cluster
-	-	CLUSTER FAILOVER
-	-	CLUSTER SET-CONFIG-EPOCH
-	-	CLUSTER RESET

Tabla 5-14 Comandos de Redis deshabilitados en la separación de lectura/escritura de instancias de DCS para Redis 4.0

Cluster	Keys	Server	Sorted Set
READONLY	MIGRATE	BGREWRITEAOF	BZPOPMAX
READWRITE	WAIT	BGSAVE	BZPOPMIN
-	-	DEBUG OBJECT	ZPOPMAX
-	-	DEBUG SEGFAULT	ZPOPMIN
-	-	LASTSAVE	-
-	-	LOLWUT	-
-	-	MODULE LIST/ LOAD/UNLOAD	-
-	-	PSYNC	-
-	-	REPLICAOF	-
-	-	SAVE	-
-	-	SHUTDOWN [NOSAVE|SAVE]	-
-	-	SLAVEOF	-
-	-	SWAPDB	-
-	-	SYNC	-

5.3 Comandos de Redis 5.0

DCS for Redis 5.0 está desarrollado basado en Redis 5.0.9 y es compatible con los protocolos y comandos de código abierto. Esta sección describe la compatibilidad de DCS for Redis 5.0 con los comandos de Redis, incluidos los comandos compatibles y deshabilitados.

Las instancias de DCS para Redis admiten la mayoría de los comandos de Redis. Cualquier cliente compatible con el protocolo de Redis puede acceder a DCS.

- Por motivos de seguridad, algunos comandos de Redis están deshabilitados en DCS, como se indica en **Comandos deshabilitados por DCS for Redis 5.0**.

- Algunos comandos de Redis son compatibles con instancias DCS de clúster para operaciones de varias claves en la misma ranura. Para más detalles, consulte **Restricciones de comandos**.
- Algunos comandos de Redis tienen restricciones de uso, que se describen en **Otras restricciones del uso de comandos**.

Comandos compatibles con DCS for Redis 5.0

- **Tabla 5-15** y **Tabla 5-16** los comandos list soportados por DCS for Redis 5.0.
- **Tabla 5-17** y **Tabla 5-18** lista de comandos soportados por las instancias de Clúster Proxy de DCS para Redis 5.0.
- **Tabla 5-19** y **Tabla 5-20** enumeran los comandos de Redis admitidos por la separación de lectura/escritura de instancias de DCS para Redis 5.0.

Para obtener más información sobre la sintaxis del comando, visite el **sitio web oficial de Redis**. Por ejemplo, para ver detalles sobre el comando SCAN, escriba SCAN en el cuadro de búsqueda de **esta página**.

NOTA

- Los comandos disponibles desde las versiones posteriores de Redis no son compatibles con las instancias de versiones anteriores. Ejecute un comando en redis-cli para comprobar si es compatible con DCS for Redis. Si se devuelve el mensaje "(error) ERR unknown command", el comando no es compatible.
- Para las instancias de DCS para Redis 5.0 en el modo Clúster Redis, asegúrese de que todos los comandos de una canalización se ejecuten en la misma partición.

Tabla 5-15 Comandos soportados por instancias de DCS para Redis 5.0 (1)

Keys	String	Hash	List	Set	Sorted Set	Server
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOUNT	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	HGET	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE	KEYS
PTTL	GET	HKEYS	LPOP	SISMEMBER	ZRANK	CLIENTKILL
RANDOMKEY	GETRANGE	HMGET	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	CLIENTLIST

Keys	String	Hash	List	Set	Sorted Set	Server
RENAME	GETSET	HMSET	LRANGE	SMOVE	ZREMRANGEBYCORE	CLIENT GETNAME
RENAMENX	INCR	HSET	LREM	SPOP	ZREVRANGE	CLIENT SETNAME
RESTORE	INCRBY	HSETNX	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	CONFIG GET
SORT	INCRBYFLOAT	HVALS	LTRIM	SREM	ZREVRANK	MONITOR
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	SLOWLOG
TYPE	MSET	HSTRLEN	RPOPLPU	SUNIONSTORE	ZUNIONSTORE	ROLE
SCAN	MSETNX	HLEN	RPOPLPUSH	SSCAN	ZINTERSTORE	SWAPDB
OBJECT	PSETEX	-	RPUSH	SPOP	ZSCAN	MEMORY
PEXPIREAT	SET	-	RPUSHX	-	ZRANGEBYLEX	CONFIG
PEXPIRE	SETBIT	-	LPUSH	-	ZLEXCOUNT	-
-	SETEX	-	-	-	ZPOPMIN	-
-	SETNX	-	-	-	ZPOPMAX	-
-	SETRANGE	-	-	-	ZREMRANGEBYSCORE	-
-	STRLEN	-	-	-	ZREM	-
-	BITFIELD	-	-	-	-	-

Tabla 5-16 Comandos soportados por instancias de DCS para Redis 5.0 (2)

HyperLoglog	Pub/Sub	Transactions	Connection	Scripting	Geo	Stream
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD	XACK
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH	XADD
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS	XCLAIM

HyperLoglog	Pub/Sub	Transactions	Connection	Scripting	Geo	Stream
-	PUNSUBSCRIBE	UNWATCH	QUIT	SCRIPT FLUSH	GEODIST	XDEL
-	SUBSCRIBE	WATCH	SELECT	SCRIPT KILL	GEORADIUS	XGROUP
-	UNSUBSCRIBE	-	-	SCRIPT LOAD	GEORADIUSBYMEMBER	XINFO
-	-	-	-	-	-	XLEN
-	-	-	-	-	-	XPENDING
-	-	-	-	-	-	XRANGE
-	-	-	-	-	-	XREAD
-	-	-	-	-	-	XREADGROUP
-	-	-	-	-	-	XREVRANGE
-	-	-	-	-	-	XTRIM

Tabla 5-17 Comandos soportados por instancias de Clúster Proxy de DCS para Redis 5.0 (1)

Keys	String	Hash	List	Set	Sorted Set	Server
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOUNT	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	HGET	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE	ROLE
PTTL	GET	HKEYS	LPOP	SISMEMBER	ZRANK	MEMORY

Keys	String	Hash	List	Set	Sorted Set	Server
RENAME	GETRANGE	HMGET	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	COMMAND
RENAMENX	GETSET	HMSET	LRANGE	SMOVE	ZREMRANGEBYCORE	COMMAND COUNT
RESTORE	INCR	HSET	LREM	SPOP	ZREVRANGE	COMMAND GETKEYS
SORT	INCRBY	HSETNX	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	COMMAND INFO
TTL	INCRBYFLOAT	HVALS	LTRIM	SREM	ZREVRANK	CONFIG GET
TYPE	MGET	HSCAN	RPOP	SUNION	ZSCORE	CONFIG RESETSTAT
SCAN	MSET	HSTRLEN	RPOPLPUSH	SUNIONSTORE	ZUNIONSTORE	CONFIG REWRITE
OBJECT	MSETNX	HLEN	RPUSH	SSCAN	ZINTERSTORE	CONFIG SET
PEXPIRE	PSETEX	HKEYS	RPUSHX	SPOP	ZSCAN	-
PEXPIREAT	SET	-	LPUSH	-	ZRANGEBYLEX	-
EXPIREAT	SETBIT	-	-	-	ZLEXCOUNT	-
KEYS	SETEX	-	-	-	ZREMRANGEBYSCORE	-
MIGRATE	SETNX	-	-	-	ZREM	-
UNLINK	SETRANGE	-	-	-	ZREMRANGEBYLEX	-
TOUCH	STRLEN	-	-	-	ZPOPMAX	-

Keys	String	Hash	List	Set	Sorted Set	Server
-	BITFIELD	-	-	-	ZPOPMIN	-
-	GETBIT	-	-	-	BZPOPMAX	-
-	-	-	-	-	BZPOPMIN	-
-	-	-	-	-	ZREVRANGEBYLEX	-

Tabla 5-18 Comandos soportados por instancias de Clúster Proxy de DCS para Redis 5.0 (2)

HyperLogLog	Pub/Sub	Transactions	Connection	Scripting	Geo
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS
-	PUNSUBSCRIBE	UNWATCH	QUIT	SCRIPT FLUSH	GEODIST
-	SUBSCRIBE	WATCH	SELECT	SCRIPT KILL	GEORADIUS
-	UNSUBSCRIBE	-	CLIENT KILL	SCRIPT LOAD	GEORADIUSBYMEMBER
-	-	-	CLIENT LIST	SCRIPT DEBUG YES|SYNC|NO	GEOSEARCH
-	-	-	CLIENT GETNAME	-	GEOSEARCHSTORE
-	-	-	CLIENT SETNAME	-	-

Tabla 5-19 Comandos soportados por la separación de lectura/escritura de instancias de DCS para Redis 5.0 (1)

Keys	String	Hash	List	Set	Sorted Set	Server
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOUNT	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	HGET	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE	MONITOR
PTTL	GET	HKEYS	LPOP	SISMEMBER	ZRANK	SLOWLOG
RANDOMKEY	GETRANGE	HMGET	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	ROLE
RENAME	GETSET	HMSET	LRANGE	SMOVE	ZREMRANGEBYCORE	SWAPDB
RENAMENX	INCR	HSET	LREM	SPOP	ZREVRANGE	MEMORY
RESTORE	INCRBY	HSETNX	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	COMMAND
SORT	INCRBYFLOAT	HVALS	LTRIM	SREM	ZREVRANK	COMMAND COUNT
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	COMMAND GETKEYS
TYPE	MSET	HSTRLEN	RPOPLPUSH	SUNIONSTORE	ZUNIONSTORE	COMMAND INFO
SCAN	MSETNX	HLEN	RPUSH	SSCAN	ZINTERSTORE	CONFIG GET

Keys	String	Hash	List	Set	Sorted Set	Server
OBJECT	PSETEX	-	RPUSHX	SPOP	ZSCAN	CONFIG RESETSTAT
PEXPIRE	SET	-	LPUSH	-	ZRANGEBYLEX	CONFIG REWRITE
PEXPIREAT	SETBIT	-	-	-	ZLEXCOUNT	CONFIG SET
EXPIREAT	SETEX	-	-	-	ZREMRANGEBYSCORE	-
KEYS	SETNX	-	-	-	ZREM	-
MIGRATE	SETRANGE	-	-	-	ZREMRANGEBYLEX	-
UNLINK	STRLEN	-	-	-	BZPOPMAX	-
TOUCH	BITFIELD	-	-	-	BZPOPMIN	-
-	GETBIT	-	-	-	ZPOPMAX	-
-	-	-	-	-	ZPOPMIN	-
-	-	-	-	-	ZREVRANGEBYLEX	-

Tabla 5-20 Comandos soportados por la separación de lectura/escritura de instancias de DCS para Redis 5.0 (2)

HyperLogLog	Pub/Sub	Transactions	Connection	Scripting	Geo
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS

HyperLoglog	Pub/Sub	Transactions	Connection	Scripting	Geo
-	PUNSUBSCRIBE	UNWATCH	QUIT	SCRIPT FLUSH	GEODIST
-	SUBSCRIBE	WATCH	SELECT	SCRIPT KILL	GEORADIUS
-	UNSUBSCRIBE	-	CLIENT KILL	SCRIPT LOAD	GEORADIUSBYMEMBER
-	-	-	CLIENT LIST	SCRIPT DEBUG YES|SYNC|NO	GEOSEARCH
-	-	-	CLIENT GETNAME	-	GEOSEARCHSTORE
-	-	-	CLIENT SETNAME	-	-

Comandos deshabilitados por DCS for Redis 5.0

A continuación se enumeran los comandos deshabilitados por DCS for Redis 5.0.

Tabla 5-21 Comandos de Redis desactivados en las instancias compatibles con Redis 5.0 de nodo único y principal/en standby

Keys	Server
MIGRATE	SLAVEOF
-	SHUTDOWN
-	LASTSAVE
-	DEBUG commands
-	COMMAND
-	SAVE
-	BGSAVE
-	BGREWRITEAOF
-	SYNC
-	PSYNC

Tabla 5-22 Comandos de Redis deshabilitados en instancias de Clúster Proxy de DCS compatibles con Redis 5.0

Keys	Server	Sorted Set	Cluster
MIGRATE	BGREWRITEAOF	-	READONLY
MOVE	BGSAVE	-	READWRITE
RANDOMKEY	CLIENT commands	-	-
WAIT	DEBUG OBJECT	-	-
-	DEBUG SEGFAULT	-	-
-	LASTSAVE	-	-
-	PSYNC	-	-
-	SAVE	-	-
-	SHUTDOWN	-	-
-	SLAVEOF	-	-
-	LATENCY commands	-	-
-	MODULE commands	-	-
-	LOLWUT	-	-
-	SWAPDB	-	-
-	REPLICAOF	-	-
-	SYNC	-	-

Tabla 5-23 Comandos de Redis deshabilitados en instancias de Clúster Redis de DCS compatibles con Redis 5.0

Keys	Server	Cluster
MIGRATE	SLAVEOF	CLUSTER MEET
-	SHUTDOWN	CLUSTER FLUSHSLOTS
-	LASTSAVE	CLUSTER ADDSLOTS
-	DEBUG commands	CLUSTER DELSLOTS
-	COMMAND	CLUSTER SETSLOT
-	SAVE	CLUSTER BUMPEPOCH
-	BGSAVE	CLUSTER SAVECONFIG

Keys	Server	Cluster
-	BGREWRITEAOF	CLUSTER FORGET
-	SYNC	CLUSTER REPLICATE
-	PSYNC	CLUSTER COUNT-FAILURE-REPORTS
-	-	CLUSTER FAILOVER
-	-	CLUSTER SET-CONFIG-EPOCH
-	-	CLUSTER RESET

Tabla 5-24 Comandos de Redis deshabilitados en la separación de lectura/escritura de instancias de DCS para Redis 5.0

Cluster	Keys	Server
READONLY	MIGRATE	BGREWRITEAOF
READWRITE	WAIT	BGSAVE
-	-	DEBUG OBJECT
-	-	DEBUG SEGFAULT
-	-	LASTSAVE
-	-	LOLWUT
-	-	MODULE LIST/LOAD/UNLOAD
-	-	PSYNC
-	-	REPLICAOF
-	-	SAVE
-	-	SHUTDOWN [NOSAVE|SAVE]
-	-	SLAVEOF
-	-	SWAPDB
-	-	SYNC

5.4 Comandos de Redis 6.0 (Prueba beta abierta)

DCS for Redis 6.0 es compatible con los protocolos y comandos de código abierto. La edición básica se basa en Redis 6.2.7 y la edición profesional se basa en KeyDB 6.0.16.

Esta sección describe la compatibilidad de DCS for Redis 6.0 con los comandos de KeyDB, incluidos los comandos compatibles y deshabilitados.

Para obtener más información acerca de la sintaxis del comando, visite el **sitio web oficial de KeyDB**.

Las instancias de DCS para Redis admiten la mayoría de los comandos de Redis. Cualquier cliente compatible con el protocolo de Redis puede acceder a DCS.

- Por motivos de seguridad, algunos comandos de Redis están deshabilitados en DCS, como se indica en **Comandos deshabilitados por DCS for Redis 6.0**.
- Algunos comandos de Redis son compatibles con instancias DCS de clúster para operaciones de varias claves en la misma ranura. Para más detalles, consulte **Restricciones de comandos**.
- Algunos comandos de Redis tienen restricciones de uso, que se describen en **Otras restricciones del uso de comandos**.

Comandos compatibles con DCS for Redis 6.0

A continuación se enumeran los comandos compatibles con DCS for Redis 6.0.

Tabla 5-25 Comandos compatibles con DCS for Redis 6.0 (1)

Keys	String	Hash	List	Set	Sorted Set	Server
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOUNT	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	HGET	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE	KEYS
PTTL	GET	HKEYS	LPOP	SISMEMBER	ZRANK	CLIENTKILL
RANDOMKEY	GETRANGE	HMGET	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	CLIENTLIST
RENAME	GETSET	HMSET	LRANGE	SMOVE	ZREMRANGEBYCORE	CLIENTGETNAME
RENAMENX	INCR	HSET	LREM	SPOP	ZREVRANGE	CLIENTSETNAME

Keys	String	Hash	List	Set	Sorted Set	Server
RESTORE	INCRBY	HSETNX	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	CONFIG GET
SORT	INCRBYFLOAT	HVALS	LTRIM	SREM	ZREVRANK	MONITOR
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	SLOWLOG
TYPE	MSET	HSTRLEN	RPOPLPU	SUNIONSTORE	ZUNIONSTORE	ROLE
SCAN	MSETNX	HLEN	RPOPLPUSH	SSCAN	ZINTERSTORE	SWAPDB
OBJECT	PSETEX	-	RPUSH	SPOP	ZSCAN	MEMORY
PEXPIREAT	SET	-	RPUSHX	-	ZRANGEBYLEX	CONFIG
PEXPIRE	SETBIT	-	LPUSH	-	ZLEXCOUNT	-
-	SETEX	-	-	-	ZPOPMIN	-
-	SETNX	-	-	-	ZPOPMAX	-
-	SETRANGE	-	-	-	ZREMRANGEBYSCORE	-
-	STRLEN	-	-	-	ZREM	-
-	BITFIELD	-	-	-	-	-

Tabla 5-26 Comandos compatibles con DCS for Redis 6.0 (2)

HyperLoglog	Pub/Sub	Transactions	Connection	Scripting	Geo	Stream
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD	XACK
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH	XADD
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS	XCLAIM
-	PUNSUBSCRIBE	UNWATCH	QUIT	SCRIPT FLUSH	GEODIST	XDEL

HyperLoglog	Pub/Sub	Transactions	Connection	Scripting	Geo	Stream
-	SUBSCRIBE	WATCH	SELECT	SCRIPT KILL	GEORADIUS	XGROUP
-	UNSUBSCRIBE	-	-	SCRIPT LOAD	GEORADIUSBYMEMBER	XINFO
-	-	-	-	-	-	XLEN
-	-	-	-	-	-	XPENDING
-	-	-	-	-	-	XRANGE
-	-	-	-	-	-	XREAD
-	-	-	-	-	-	XREADGROUP
-	-	-	-	-	-	XREVRANGE
-	-	-	-	-	-	XTRIM

Comandos deshabilitados por DCS for Redis 6.0

A continuación se enumeran los comandos deshabilitados por DCS for Redis 6.0.

Tabla 5-27 Comandos de Redis deshabilitados por instancias principal/en standby de DCS compatible con Redis 6.0

Keys	Server
MIGRATE	SLAVEOF
-	SHUTDOWN
-	LASTSAVE
-	DEBUG commands
-	COMMAND
-	SAVE
-	BGSAVE
-	BGREWRITEAOF
-	SYNC
-	PSYNC

5.5 Comandos de la CLI web

Web CLI es una herramienta de línea de comandos proporcionada en la consola de DCS. Esta sección describe la compatibilidad de la CLI web con los comandos de Redis, incluidos los comandos compatibles y deshabilitados. Actualmente, solo DCS for Redis 4.0, 6.0 y 5.0 soportan la CLI web.

NOTA

- Currently, no command parameters on the Web CLI support Chinese. Spaces cannot be included in keys and values.
- Si el valor está vacío, se devuelve **nil** después de ejecutar el comando **GET**.

Comandos admitidos en la CLI web

A continuación se enumeran los comandos admitidos cuando se usa la CLI web. Para obtener más información sobre la sintaxis del comando, visite el **sitio web oficial de Redis**. Por ejemplo, para ver detalles sobre el comando SCAN, escriba SCAN en el cuadro de búsqueda de **esta página**.

Tabla 5-28 Comandos soportados por CLI Web (1)

Keys	String	List	Set	Sorted Set	Server
DEL	APPEND	RPUSH	SADD	ZADD	FLUSHALL
OBJECT	BITCOUNT	RPUSHX	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	LLEN	SINTERSTORE	ZRANGEBYSCORE	CLIENT KILL
PTTL	GET	LPOP	SISMEMBER	ZRANK	CLIENT LIST
RANDOMKEY	GETRANGE	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	CLIENT GETNAME
RENAME	GETSET	LRANGE	SMOVE	ZREMRANGEBYCORE	CLIENT SETNAME
RENAMENX	INCR	LREM	SPOP	ZREVRANGE	CONFIG GET
SCAN	INCRBY	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	MONITOR

Keys	String	List	Set	Sorted Set	Server
SORT	INCRBYFLOAT	LTRIM	SREM	ZREVRANK	SLOWLOG
TTL	MGET	RPOP	SUNION	ZSCORE	ROLE
TYPE	MSET	RPOPLPU	SUNIONSTORE	ZUNIONSTORE	SWAPDB
-	MSETNX	RPOPLPUSH	SSCAN	ZINTERSTORE	MEMORY
-	PSETEX	-	SPOP	ZSCAN	-
-	SET	-	-	ZRANGEBYLEX	-
-	SETBIT	-	-	ZLEXCOUNT	-
-	SETEX	-	-	-	-
-	SETNX	-	-	-	-
-	SETRANGE	-	-	-	-
-	STRLEN	-	-	-	-
-	BITFIELD	-	-	-	-

Tabla 5-29 Comandos soportados por CLI Web (2)

Hash	HyperLoglog	Connection	Scripting	Geo
HDEL	PFADD	AUTH	EVAL	GEOADD
HEXISTS	PFCOUNT	ECHO	EVALSHA	GEOHASH
HGET	PFMERGE	PING	SCRIPT EXISTS	GEOPOS
HGETALL	-	QUIT	SCRIPT FLUSH	GEODIST
HINCRBY	-	-	SCRIPT KILL	GEORADIUS
HINCRBYFLOAT	-	-	SCRIPT LOAD	GEORADIUSBYMEMBER
HKEYS	-	-	-	-
HMGET	-	-	-	-
HMSET	-	-	-	-
HSET	-	-	-	-

Hash	HyperLoglog	Connection	Scripting	Geo
HSETNX	-	-	-	-
HVALS	-	-	-	-
HSCAN	-	-	-	-
HSTRLEN	-	-	-	-

Comandos deshabilitados en la CLI web

A continuación se enumeran los comandos deshabilitados cuando se usa la CLI web.

Tabla 5-30 Comandos de Redis deshabilitados en la CLI web para instancias de nodo único y principal/en standby (1)

Keys	Server	Transactions	Pub/Sub
MIGRATE	SLAVEOF	UNWATCH	PSUBSCRIBE
WAIT	SHUTDOWN	REPLICAOF	PUBLISH
DUMP	DEBUG commands	DISCARD	PUBSUB
RESTORE	CONFIG SET	EXEC	PUNSUBSCRIBE
-	CONFIG REWRITE	MULTI	SUBSCRIBE
-	CONFIG RESETSTAT	WATCH	UNSUBSCRIBE
-	SAVE	-	-
-	BGSAVE	-	-
-	BGREWRITEAOF	-	-
-	COMMAND	-	-
-	KEYS	-	-
-	MONITOR	-	-
-	SYNC	-	-
-	PSYNC	-	-
-	ACL	-	-

Tabla 5-31 Comandos de Redis deshabilitados en la CLI web para instancias de nodo único y principal/en standby (2)

List	Connection	Sorted Set
BLPOP	SELECT	BZPOPMAX
BRPOP	-	BZPOPMIN
BLMOVE	-	BZMPOP
BRPOPLPUSH	-	-
BLMPOP	-	-

Tabla 5-32 Comandos de Redis desactivados en Web CLI para instancias de Clúster Redis (1)

Keys	Server	Transactions	Cluster
MIGRATE	SLAVEOF	UNWATCH	CLUSTER MEET
WAIT	SHUTDOWN	REPLICAOF	CLUSTER FLUSHSLOTS
DUMP	DEBUG commands	DISCARD	CLUSTER ADDSLOTS
RESTORE	CONFIG SET	EXEC	CLUSTER DELSLOTS
-	CONFIG REWRITE	MULTI	CLUSTER SETSLOT
-	CONFIG RESETSTAT	WATCH	CLUSTER BUMPEPOCH
-	SAVE	-	CLUSTER SAVECONFIG
-	BGSAVE	-	CLUSTER FORGET
-	BGREWRITEAOF	-	CLUSTER REPLICATE
-	COMMAND	-	CLUSTER COUNT-FAILURE-REPORTS
-	KEYS	-	CLUSTER FAILOVER
-	MONITOR	-	CLUSTER SET-CONFIG-EPOCH
-	SYNC	-	CLUSTER RESET
-	PSYNC	-	-
-	ACL	-	-

Tabla 5-33 Comandos de Redis desactivados en Web CLI para instancias de Clúster Redis (2)

Pub/Sub	List	Connection	Sorted Set
PSUBSCRIBE	BLPOP	SELECT	BZPOPMAX
PUBLISH	BRPOP	-	BZPOPMIN
PUBSUB	BLMOVE	-	BZMPOP
PUNSUBSCRIBE	BRPOPLPUSH	-	-
SUBSCRIBE	BLMPOP	-	-
UNSUBSCRIBE	-	-	-

5.6 Comandos de Memcached (No disponibles pronto)

NOTA

DCS for Memcached está a punto de no estar disponible y ya no se vende en algunas regiones. Puede utilizar DCS for Redis 4.0 o 5.0 en su lugar.

Memcached soporta el protocolo de texto basado en TCP y el protocolo binario. Cualquier cliente compatible con un protocolo de Memcached puede acceder a las instancias de DCS.

Protocolo de texto de Memcached

El protocolo de texto Memcached utiliza texto ASCII para transferir comandos, lo que le ayuda a compilar clientes y problemas de depuración. Las instancias de DCS para Memcached pueden incluso conectarse directamente mediante Telnet.

En comparación con el protocolo binario de Memcached, el protocolo de texto de Memcached es compatible con más clientes de código abierto, pero el protocolo de texto no admite autenticación.

NOTA

Los clientes pueden usar el protocolo de texto Memcached para acceder a las instancias de Memcached de DCS solo si está habilitado el acceso sin contraseña. El acceso sin contraseña significa que el acceso a las instancias de Memcached de DCS no estará protegido por nombre de usuario y contraseña, y cualquier cliente de Memcached que satisfaga las reglas del grupo de seguridad en la misma VPC puede acceder a las instancias. Habilitar el acceso sin contraseña plantea los riesgos de seguridad. Tenga cuidado al habilitar el acceso sin contraseña.

Tabla 5-34 enumera los comandos admitidos por el protocolo de texto Memcached y describe si estos comandos son compatibles con las instancias de DCS Memcached.

Tabla 5-34 Comandos compatibles con el protocolo de texto de Memcached

Comando	Descripción	Soportado por DCS
add	Agrega datos.	Sí

Comando	Descripción	Soportado por DCS
set	Establece datos, incluida la adición o modificación de datos.	Sí
replace	Reemplaza los datos.	Sí
append	Agrega datos después del valor de la clave especificada.	Sí
prepend	Agrega datos antes del valor de una clave especificada.	Sí
cas	Comprueba y establece los datos.	Sí
get	Consulta datos.	Sí
gets	Consulta detalles de datos.	Sí
delete	Elimina los datos.	Sí
incr	Agrega la cantidad especificada al contador solicitado.	Sí
decr	Elimina la cantidad especificada en el contador solicitado.	Sí
touch	Actualiza el tiempo de caducidad de los datos existentes.	Sí
quit	Cierra la conexión.	Sí
flush_all	Invalida todos los datos existentes. **NOTA** El valor de la opción delay (si existe) debe ser **0**.	Sí
version	Consulta información de la versión de Memcached.	Sí
stats	Gestiona las estadísticas de operación. **NOTA** Actualmente, solo se pueden consultar estadísticas básicas. No se pueden consultar los comandos de los parámetros opcionales.	Sí
cache_memlimit	Ajusta el límite de memoria caché.	No
slabs	Consulta el uso de estructuras de almacenamiento internas.	No
lru	Gestiona las políticas de eliminación de datos caducados.	No
lru_crawler	Gestiona los hilos de eliminación de datos caducados.	No

Comando	Descripción	Soportado por DCS
verbosity	Establece el nivel de detalle de la salida del registro.	No
watch	Inspecciona lo que está pasando internamente.	No

Protocolo binario de Memcached

El protocolo binario Memcached codifica comandos y operaciones en estructuras específicas antes de enviarlos. Los comandos se representan mediante cadenas de caracteres predefinidas.

El protocolo binario de Memcached proporciona más características pero menos clientes que el protocolo de texto de Memcached. El protocolo binario de Memcached es más seguro que el protocolo de texto de Memcached, ya que además admite la autenticación simple y autenticación de capa de seguridad (SASL).

Tabla 5-35 enumera los comandos admitidos por el protocolo de binario de Memcached y describe si estos comandos son compatibles con las instancias de DCS Memcached.

Tabla 5-35 Comandos soportados por el protocolo binario de Memcached

Código de Comando	Comando	Descripción	Soportado por DCS
0x00	GET	Consulta datos.	Sí
0x01	SET	Establece datos, incluida la adición o modificación de datos.	Sí
0x02	ADD	Agrega datos.	Sí
0x03	REPLACE	Reemplaza los datos.	Sí
0x04	DELETE	Elimina los datos.	Sí
0x05	INCREMENT	Agrega la cantidad especificada al contador solicitado.	Sí
0x06	DECREMENT	Elimina la cantidad especificada en el contador solicitado.	Sí
0x07	QUIT	Cierra la conexión.	Sí
0x08	FLUSH	Invalida todos los datos existentes. **NOTA** El valor de la opción delay (si existe) debe ser **0**.	Sí
0x09	GETQ	Consulta datos. El cliente no recibirá ninguna respuesta en caso de fallo.	Sí

Código de Comando	Comando	Descripción	Soportado por DCS
0x0a	NOOP	Instrucción de no operación, equivalente a ping.	Sí
0x0b	VERSION	Consulta información de la versión de Memcached.	Sí
0x0c	GETK	Consulta datos y agrega una clave al paquete de respuesta.	Sí
0x0d	GETKQ	Consulta datos y devuelve una clave. El cliente no recibirá ninguna respuesta en caso de fallo.	Sí
0x0e	APPEND	Agrega datos después del valor de la clave especificada.	Sí
0x0f	PREPEND	Agrega datos antes del valor de una clave especificada.	Sí
0x10	STAT	Consulta estadísticas de instancias de DCS Memcached. **NOTA** Actualmente, solo se pueden consultar estadísticas básicas. No se pueden consultar los comandos de los parámetros opcionales.	Sí
0x11	SETQ	Establece datos, incluida la adición o modificación de datos. El comando **SETQ** solo devuelve una respuesta en caso de errores. El cliente no recibirá ninguna respuesta en caso de éxito.	Sí
0x12	ADDQ	Agrega datos. A diferencia del comando **ADD**, el comando **ADDQ** solo devuelve una respuesta en caso de errores. El cliente no recibirá ninguna respuesta en caso de éxito.	Sí
0x13	REPLACEQ	Reemplaza los datos. A diferencia del comando **REPLACE**, el comando **REPLACEQ** solo devuelve una respuesta en caso de errores. El cliente no recibirá ninguna respuesta en caso de éxito.	Sí

Código de Comando	Comando	Descripción	Soportado por DCS
0x14	DELETEQ	Elimina los datos. A diferencia del comando **DELETE**, el comando **DELETEQ** solo devuelve una respuesta en caso de errores. El cliente no recibirá ninguna respuesta en caso de éxito.	Sí
0x15	INCREMENTQ	Agrega la cantidad especificada al contador solicitado. A diferencia del comando **INCREMENT** , el comando **INCREMENTQ** solo devuelve una respuesta en caso de errores. El cliente no recibirá ninguna respuesta en caso de éxito.	Sí
0x16	DECREMENTQ	Elimina la cantidad especificada en el contador solicitado. A diferencia del comando **DECREMENT**, el comando **DECREMENTQ** solo devuelve una respuesta en caso de errores. El cliente no recibirá ninguna respuesta en caso de éxito.	Sí
0x17	QUITQ	Cierra la conexión.	Sí
0x18	FLUSHQ	Borra datos y no devuelve información. **NOTA** El valor de la opción delay (si existe) debe ser **0**.	Sí
0x19	APPENDQ	Agrega datos después del valor de la clave especificada. A diferencia del comando **APPEND** , el comando **APPENDQ** solo devuelve una respuesta en caso de errores. El cliente no recibirá ninguna respuesta en caso de éxito.	Sí

Código de Comando	Comando	Descripción	Soportado por DCS
0x1a	PREPENDQ	Agrega datos antes del valor de una clave especificada. A diferencia del comando **PREPEND**, el comando **PREPENDQ** solo devuelve una respuesta en caso de errores. El cliente no recibirá ninguna respuesta en caso de éxito.	Sí
0x1c	TOUCH	Actualiza el tiempo de caducidad de los datos existentes.	Sí
0x1d	GAT	Consulta los datos y actualiza el tiempo de caducidad de los datos existentes.	Sí
0x1e	GATQ	Consulta datos y devuelve una clave. El cliente no recibirá ninguna respuesta en caso de fallo.	Sí
0x23	GATK	Consulta datos, agrega una clave al paquete de respuesta y actualiza el tiempo de caducidad de los datos existentes.	Sí
0x24	GATKQ	Consulta datos, devuelve una clave y actualiza el tiempo de caducidad de los datos existentes. El cliente no recibirá ninguna respuesta en caso de fallo.	Sí
0x20	SASL_LIST_MECHS	Pregunta al servidor qué mecanismos de autenticación SASL admite.	Sí
0x21	SASL_AUTH	Inicia la autenticación SASL.	Sí
0x22	SASL_STEP	Se requieren pasos de autenticación adicionales.	Sí

5.7 Restricciones de comandos

Algunos comandos de Redis son compatibles con instancias DCS de clúster para operaciones de varias claves en la misma ranura. Para más detalles, consulte **Tabla 5-36**.

Tabla 5-37 y **Tabla 5-38** enumeran los comandos restringidos para las instancias de Clúster Proxy y de separación de lectura/escritura de DCS para Redis 4.0.

Tabla 5-39 y **Tabla 5-40** enumeran los comandos restringidos para las instancias de Clúster Proxy y de separación de lectura/escritura de DCS para Redis 5.0.

Tabla 5-36 Comandos restringidos de Redis en las instancias de Clúster Redis de DCS

Categoría	Descripción
Set (Conjunto)	
SINTER	Devuelve los miembros del conjunto resultante de la intersección de todos los conjuntos dados.
SINTERSTORE	Igual a **SINTER**, pero en lugar de devolver el conjunto de resultados, se almacena en *destination*.
SUNION	Devuelve los miembros del conjunto resultante de la unión de todos los conjuntos dados.
SUNIONSTORE	Igual a **SUNION**, pero en lugar de devolver el conjunto de resultados, se almacena en *destination*.
SDIFF	Devuelve los miembros del conjunto resultante de la diferencia entre el primer conjunto y todos los conjuntos sucesivos.
SDIFFSTORE	Igual a **SDIFF**, pero en lugar de devolver el conjunto de resultados, se almacena en *destination*.
SMOVE	Mueve **member** del conjunto en **source** al conjunto en *destination*.
Sorted Set (Conjunto ordenado)	
ZUNIONSTORE	Calcula la unión de conjuntos ordenados de *numkeys* dadas por las claves especificadas.
ZINTERSTORE	Calcula la intersección de conjuntos ordenados por *numkeys* dadas por las claves especificadas.
HyperLogLog	
PFCOUNT	Devuelve la cardinalidad aproximada calculada por la estructura de datos HyperLogLog almacenada en la variable especificada.
PFMERGE	Combina varios valores HyperLogLog en un valor único.
Keys	
RENAME	Cambia el nombre de *key* a *newkey*.
RENAMENX	Cambia el nombre de *key* a *newkey* si *newkey* aún no existe.
BITOP	Realiza una operación a nivel de bits entre varias claves (que contienen valores de cadena) y almacena el resultado en la clave de destino.
RPOPLPUSH	Devuelve y elimina el último elemento (tail) de la lista almacenada en el origen y lo empuja al primer elemento (head) de la lista almacenada en *destination*.

Categoría	Descripción
String	
MSETNX	Establece las claves dadas en sus respectivos valores.

NOTA

Mientras se ejecutan comandos que tardan mucho en ejecutarse, como **FLUSHALL**, las instancias DCS pueden no responder a otros comandos y pueden cambiar al estado defectuoso. Después de que el comando termine de ejecutarse, la instancia volverá a la normalidad.

Tabla 5-37 Comandos de Redis restringidos para instancias de Clúster Proxy de DCS para Redis 4.0

Categoría	Comando	Restricción
Sets	SMOVE	Para una instancia de Clúster Proxy, las claves de origen y destino deben estar en la misma ranura.
Geo	GEORADIUS	● Para una instancia de Clúster Proxy, todas las claves transferidas deben estar en la misma ranura. ● Para una instancia de Clúster Proxy con múltiples bases de datos, la opción **STORE** no es compatible.
	GEORADIUSBYMEMBER	
	GEOSEARCHSTORE	
Connection	CLIENT KILL	● Solo se admiten los dos formatos siguientes: – CLIENT KILL ip:port – CLIENT KILL ADDR ip:port ● El campo **id** tiene un valor aleatorio y no cumple con el requisito idc1<idc2→Tc1<Tc2.
	CLIENT LIST	● Solo se admiten los dos formatos siguientes: – CLIENT LIST – CLIENT LIST [TYPE normal|master|replica|pubsub] ● El campo **id** tiene un valor aleatorio y no cumple con el requisito idc1<idc2→Tc1<Tc2.

Categoría	Comando	Restricción
	SELECT index	Multi-DB de instancias de Clúster Proxy se puede implementar cambiando las claves. No se recomienda esta solución. Restricciones sobre el soporte multi-DB para una instancia de Clúster Proxy: 1. El almacenamiento backend reescribe las claves basadas en ciertas reglas. Las claves del archivo RDB exportado no son las claves originales, pero se puede acceder a ellas a través del protocolo de Redis. 2. El comando **FLUSHDB** elimina las claves una por una, lo que lleva mucho tiempo. 3. No se admite **SWAPDB**. 4. El comando **INFO KEYSPACE** no devuelve datos de varias bases de datos. 5. El comando **DBSIZE** consume mucho tiempo. No lo utilice en el código. 6. Si se utilizan varias bases de datos, el rendimiento de los comandos **KEYS** y **SCAN** se deteriora hasta en un 50%. 7. Los scripts LUA no admiten el uso de varias bases de datos. 8. El comando **RANDOMKEY** no admite el uso de varias bases de datos. 9. De forma predeterminada, multi-DB está deshabilitado. Antes de habilitar o deshabilitar esta opción para una instancia, borre los datos de la instancia.
HyperLogLog	PFCOUNT	Para una instancia de Clúster Proxy, todas las claves transferidas deben estar en la misma ranura.
	PFMERGE	
Keys	RENAME	Para una instancia de Clúster Proxy, todas las claves transferidas deben estar en la misma ranura.
	RENAMENX	
	SCAN	Las instancias de Clúster Proxy no soportan el comando **SCAN** en canalizaciones.

Categoría	Comando	Restricción
Lists	BLPOP	Para una instancia de Clúster Proxy, todas las claves transferidas deben estar en la misma ranura.
	BRPOP	
	BRPOPLPUSH	
Pub/Sub	PSUBSCRIBE	Las instancias de Clúster Proxy no admiten la suscripción a eventos de espacio de claves, por lo que no habría un fallo en la suscripción a eventos de espacio de claves.
Scripting	EVAL	● Para una instancia de Clúster Proxy, todas las claves transferidas deben estar en la misma ranura. ● Cuando la función multi-DB está habilitada para una instancia Clúster Proxy, se modifica el parámetro **KEYS**. Preste atención al parámetro **KEYS** usado en el script Lua.
	EVALSHA	
Server	MEMORY DOCTOR	Para una instancia de Clúster Proxy, agregue el *ip:port* del nodo al final del comando.
	MEMORY HELP	
	MEMORY MALLOC-STATS	
	MEMORY PURGE	
	MEMORY STATS	
	MEMORY USAGE	
	MONITOR	
Strings	BITOP	Para una instancia de Clúster Proxy, todas las claves transferidas deben estar en la misma ranura.
	MSETNX	
Transactions	WATCH	Para una instancia de Clúster Proxy, todas las claves transferidas deben estar en la misma ranura.
Streams	XACK	Actualmente, las instancias Clúster Proxy no soportan Streams.
	XADD	
	XCLAIM	
	XDEL	
	XGROUP	
	XINFO	
	XLEN	

Categoría	Comando	Restricción
	XPENDING	
	XRANGE	
	XTRIM	
	XREVRANGE	
	XREAD	
	XREADGROUP GROUP	

Tabla 5-38 Comandos de Redis restringidos para las instancias de la separación de lectura/escritura de DCS para Redis 4.0

Categoría	Comando	Restricción
Conexión	CLIENT KILL	● Solo se admiten los dos formatos siguientes: – CLIENT KILL ip:port – CLIENT KILL ADDR ip:port ● El campo **id** tiene un valor aleatorio y no cumple con el requisito idc1<idc2→Tc1<Tc2.
	CLIENT LIST	● Solo se admiten los dos formatos siguientes: – CLIENT LIST – CLIENT LIST [TYPE normal|master|replica|pubsub] ● El campo **id** tiene un valor aleatorio y no cumple con el requisito idc1<idc2→Tc1<Tc2.

Tabla 5-39 Comandos de Redis restringidos para instancias de Clúster Proxy de DCS para Redis 5.0

Categoría	Comando	Restricción
Sets	SMOVE	Para una instancia de Clúster Proxy, las claves de origen y destino deben estar en la misma ranura.
Sorted sets	BZPOPMAX	Para una instancia de Clúster Proxy, todas las claves transferidas deben estar en la misma ranura.
	BZPOPMIN	

Categoría	Comando	Restricción
Geo	GEORADIUS	● Para una instancia de Clúster Proxy, todas las claves transferidas deben estar en la misma ranura. ● Para una instancia de Clúster Proxy en modo multi-DB, la opción **STORE** no es compatible.
	GEORADIUSBYMEMBER	
	GEOSEARCHSTORE	
Connection	CLIENT KILL	● Solo se admiten los dos formatos siguientes: – CLIENT KILL ip:port – CLIENT KILL ADDR ip:port ● El campo **id** tiene un valor aleatorio y no cumple con el requisito idc1<idc2→Tc1<Tc2.
	CLIENT LIST	● Solo se admiten los dos formatos siguientes: – CLIENT LIST – CLIENT LIST [TYPE normal|master|replica|pubsub] ● El campo **id** tiene un valor aleatorio y no cumple con el requisito idc1<idc2→Tc1<Tc2.

Categoría	Comando	Restricción
	SELECT index	Multi-DB de instancias de Clúster Proxy se puede implementar cambiando las claves. No se recomienda esta solución. Restricciones sobre el soporte multi-DB para una instancia de Clúster Proxy: 1. El almacenamiento backend reescribe las claves basadas en ciertas reglas. Las claves del archivo RDB exportado no son las claves originales, pero se puede acceder a ellas a través del protocolo de Redis. 2. El comando **FLUSHDB** elimina las claves una por una, lo que lleva mucho tiempo. 3. No se admite **SWAPDB**. 4. El comando **INFO KEYSPACE** no devuelve datos de varias bases de datos. 5. El comando **DBSIZE** consume mucho tiempo. No lo utilice en el código. 6. Si se utilizan varias bases de datos, el rendimiento de los comandos **KEYS** y **SCAN** se deteriora hasta en un 50%. 7. Los scripts LUA no admiten el uso de varias bases de datos. 8. El comando **RANDOMKEY** no admite el uso de varias bases de datos. 9. De forma predeterminada, multi-DB está deshabilitado. Antes de habilitar o deshabilitar esta opción para una instancia, borre los datos de la instancia.
HyperLogLog	PFCOUNT	Para una instancia de Clúster Proxy, todas las claves transferidas deben estar en la misma ranura.
	PFMERGE	
Keys	RENAME	Para una instancia de Clúster Proxy, todas las claves transferidas deben estar en la misma ranura.
	RENAMENX	
	SCAN	Las instancias de Clúster Proxy no soportan el comando **SCAN** en canalizaciones.

Categoría	Comando	Restricción
Lists	BLPOP	Para una instancia de Clúster Proxy, todas las claves transferidas deben estar en la misma ranura.
	BRPOP	
	BRPOPLPUSH	
Pub/Sub	PSUBSCRIBE	Las instancias de Clúster Proxy no admiten la suscripción a eventos de espacio de claves, por lo que no habría un fallo en la suscripción a eventos de espacio de claves.
Scripting	EVAL	● Para una instancia de Clúster Proxy, todas las claves transferidas deben estar en la misma ranura. ● Cuando la función multi-DB está habilitada para una instancia Clúster Proxy, se modificará el parámetro **KEYS**. Preste atención al parámetro **KEYS** usado en el script Lua.
	EVALSHA	
Server	MEMORY DOCTOR	Para una instancia de Clúster Proxy, agregue el *ip:port* del nodo al final del comando.
	MEMORY HELP	
	MEMORY MALLOC-STATS	
	MEMORY PURGE	
	MEMORY STATS	
	MEMORY USAGE	
	MONITOR	
Strings	BITOP	Para una instancia de Clúster Proxy, todas las claves transferidas deben estar en la misma ranura.
	MSETNX	
Transactions	WATCH	Para una instancia de Clúster Proxy, todas las claves transferidas deben estar en la misma ranura.
Streams	XACK	Actualmente, las instancias Clúster Proxy no soportan Streams.
	XADD	
	XCLAIM	
	XDEL	
	XGROUP	
	XINFO	
	XLEN	

Categoría	Comando	Restricción
	XPENDING	
	XRANGE	
	XTRIM	
	XREVRANGE	
	XREAD	
	XREADGROUP GROUP	

Tabla 5-40 Comandos de Redis restringidos para las instancias de la separación de lectura/escritura de DCS para Redis 5.0

Categoría	Comando	Restricción
Connection	CLIENT KILL	● Solo se admiten los dos formatos siguientes: – CLIENT KILL ip:port – CLIENT KILL ADDR ip:port ● El campo **id** tiene un valor aleatorio y no cumple con el requisito idc1<idc2→Tc1<Tc2.
	CLIENT LIST	● Solo se admiten los dos formatos siguientes: – CLIENT LIST – CLIENT LIST [TYPE normal|master|replica|pubsub] ● El campo **id** tiene un valor aleatorio y no cumple con el requisito idc1<idc2→Tc1<Tc2.
Streams	XREAD	No se admite la opción **BLOCK**.
	XREADGROUP GROUP	

5.8 Otras restricciones del uso de comandos

En esta sección se describen las restricciones de algunos comandos de Redis.

Comando de KEYS

En el caso de una gran cantidad de datos almacenados en caché, la ejecución del comando KEYS puede bloquear la ejecución de otros comandos durante mucho tiempo u ocupar una memoria excepcionalmente grande. Por lo tanto, cuando ejecute el comando KEYS, describa

el patrón exacto y no utilice **keys *** difusas. No utilice el comando KEYS en el entorno de producción. De lo contrario, el servicio que se ejecuta se verá afectado.

Comandos en el grupo de Server

- Mientras se ejecutan comandos que tardan mucho en ejecutarse, como **FLUSHALL**, las instancias DCS pueden no responder a otros comandos y pueden cambiar al estado defectuoso. Después de que el comando termine de ejecutarse, la instancia volverá a la normalidad.
- Cuando se ejecuta el comando **FLUSHDB** o **FLUSHALL**, la ejecución de otros comandos de servicio puede bloquearse durante mucho tiempo en el caso de una gran cantidad de datos almacenados en caché.

Comandos EVAL y EVALSHA

- Cuando se ejecuta el comando EVAL o EVALSHA, se debe contener al menos una key en el parámetro de comando. De lo contrario, el mensaje de error, Se muestra "ERR eval/evalsha numkeys must be bigger than zero in redis cluster mode" ("ERR eval/evalsha numkeys debe ser mayor que cero en el modo redis cluster").
- Cuando se ejecuta el comando EVAL o EVALSHA, una instancia de DCS Redis de clúster utiliza la primera key para calcular ranuras. Asegúrate de que las llaves que se van a utilizar en tu código estén en la misma ranura. Para obtener más información, visite el **sitio web oficial de Redis**.
- Para el comando EVAL:
  - Aprenda las características del script Lua de Redis antes de ejecutar el comando EVAL. Para obtener más información, visite el **sitio web oficial de Redis**.
  - El tiempo de ejecución de un script de Lua es de 5 segundos. Deben evitarse las declaraciones que consumen mucho tiempo, como las declaraciones de sueño durante mucho tiempo y las declaraciones de bucle grande.
  - Al llamar a un script Lua, no utilice funciones aleatorias para especificar claves. De lo contrario, los resultados de la ejecución son inconsistentes en los nodos principal y en standby.

Otras restricciones

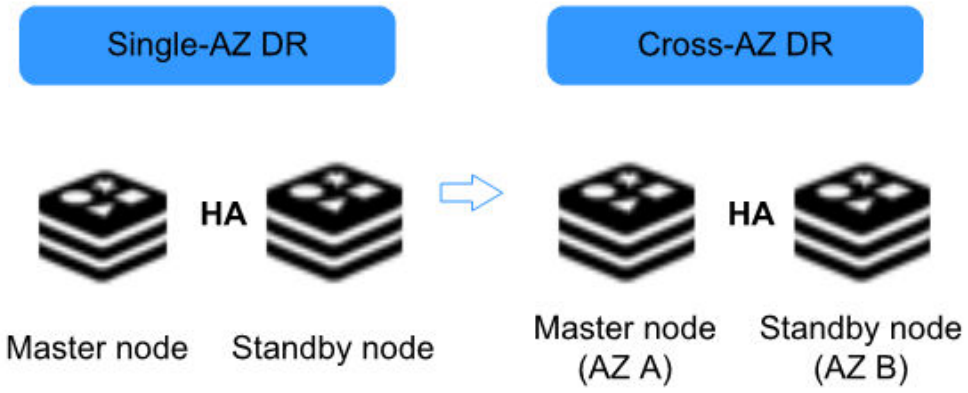
- El límite de tiempo para ejecutar un comando Redis es de 15 segundos. Para evitar que otros servicios fallen, se activará un switchover principal/replicado una vez que se agote el tiempo de ejecución del comando.
- Las instancias de clúster de DCS para Redis creadas antes del 10 de julio de 2018 deben actualizarse para que admitan los siguientes comandos:

  SINTER, SDIFF, SUNION, PFCOUNT, PFMERGE, SINTERSTORE, SUNIONSTORE, SDIFFSTORE, SMOVE, ZUNIONSTORE, ZINTERSTORE, EVAL, EVALSHA, BITOP, RENAME, RENAMENX, RPOPLPUSH, MSETNX, SCRIPT LOAD, SCRIPT KILL, SCRIPT EXISTS, and SCRIPT FLUSH

6 Recuperación ante desastres y solución multiactiva

Ya sea que utilice DCS como caché frontend o almacenamiento de datos backend, DCS siempre está listo para garantizar la confiabilidad de los datos y la disponibilidad del servicio. La siguiente figura muestra la evolución de las arquitecturas DR de DCS.

Figura 6-1 Evolución de la arquitectura DR de DCS



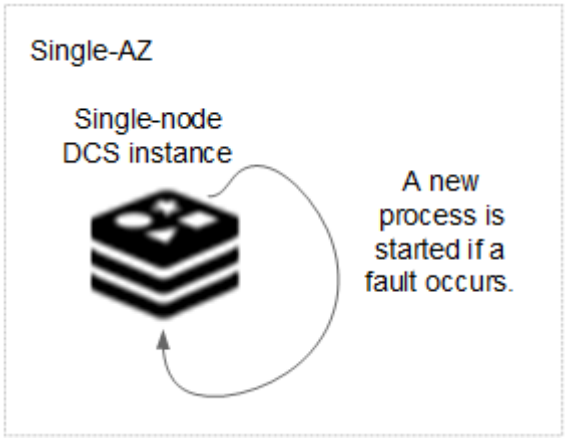
Para cumplir con los requisitos de confiabilidad de sus datos y servicios, puede optar por implementar su instancia de DCS dentro de una única AZ o entre AZ.

HA de la AZ única dentro de una región

Despliegue de la AZ única significa desplegar una instancia dentro de una sala de equipos físicos. DCS proporciona HA de proceso/servicio, persistencia de datos y políticas de DR en standby activa para diferentes tipos de instancias de DCS.

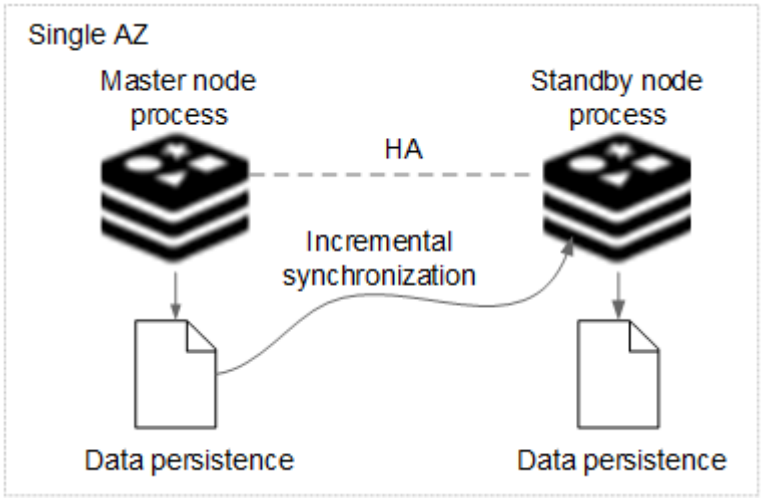
Instancia de nodo único de DCS: cuando DCS detecta un fallo de proceso, se inicia un nuevo proceso para garantizar el HA del servicio.

Figura 6-2 HA para una instancia de DCS de nodo único implementada dentro de una AZ



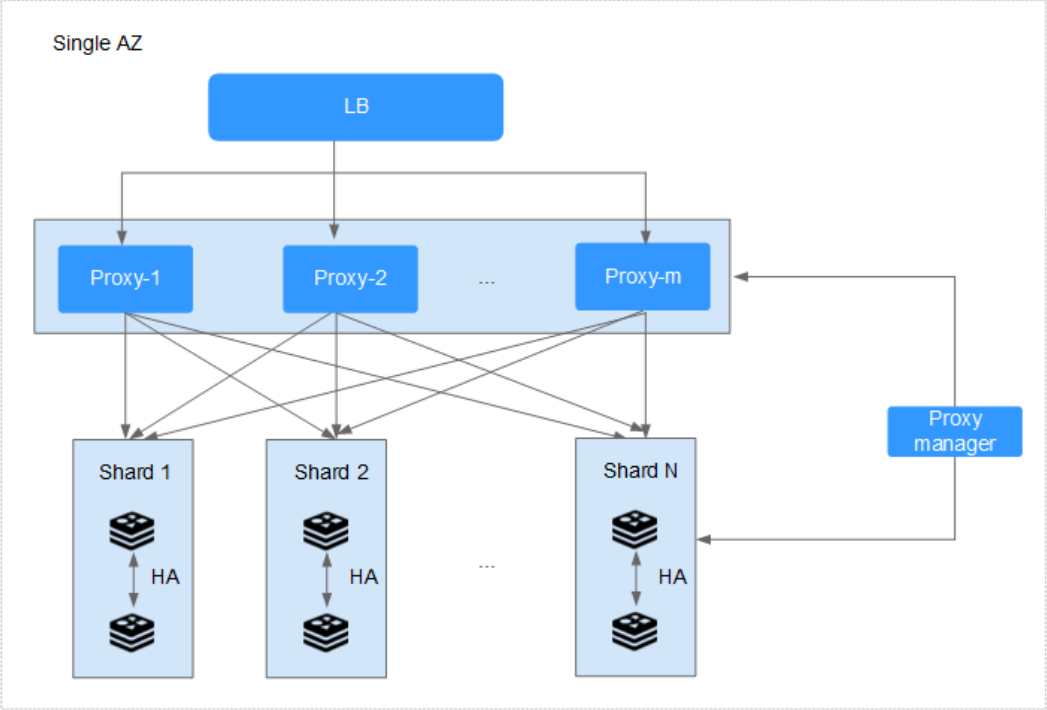
Instancia principal/en standby de DCS: Los datos se mantienen en el disco en el nodo principal y se sincronizan y persisten incrementalmente en el nodo en standby, logrando la persistencia de los datos y en standby caliente.

Figura 6-3 HA para una instancia principal/en standby de DCS implementada dentro de una AZ



Instancia de clúster de DCS: Similar a una instancia principal/en standby, los datos de cada partición (proceso de instancia) de una instancia de clúster se sincronizan entre los nodos principal y en standby y se mantienen en ambos nodos.

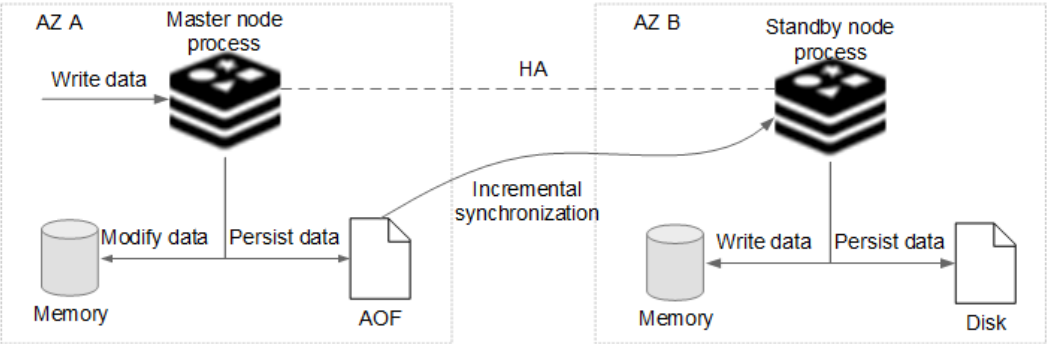
Figura 6-4 HA para una instancia DCS de clúster implementada dentro de una AZ



DR entre las AZ dentro de una región

Los nodos principal y en standby de una instancia de DCS principal/en standby o de clúster se pueden implementar a través de AZ (en diferentes salas de equipos). Las fuentes de alimentación y las redes de diferentes AZ están aisladas físicamente. Cuando se produce un fallo en el AZ donde se despliega el nodo principal, el nodo en standby se conecta al cliente y se hace cargo de las operaciones de lectura y escritura de datos.

Figura 6-5 Implementación entre AZ de una instancia principal/en standby de DCS



NOTA

Este mecanismo se aplica de manera similar a una instancia de clúster de DCS, en la que cada partición (proceso) se implementa a través de AZ.

Al crear una instancia principal/en standby de DCS, seleccione una AZ en standby que sea diferente de la AZ principal como se muestra a continuación.

Figura 6-6 Selección de diferentes AZ

NOTA

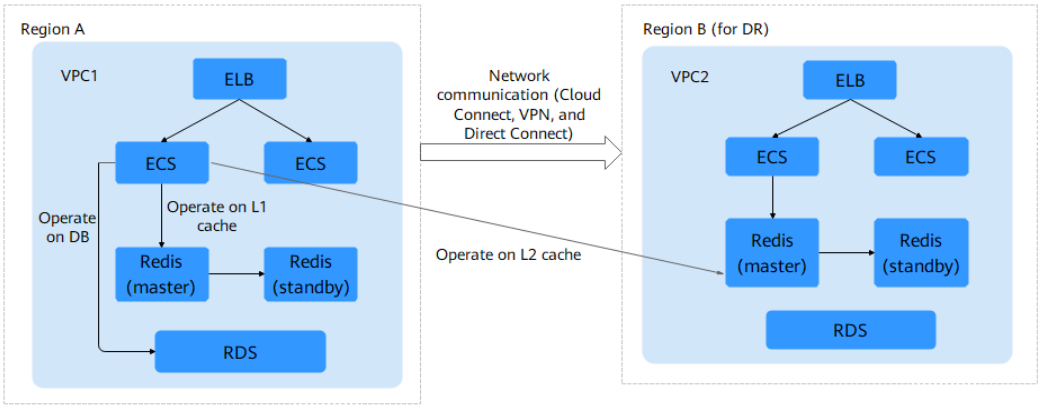
También puede implementar su aplicación en todas las AZ para garantizar la fiabilidad de los datos y la disponibilidad del servicio en caso de interrupciones en el suministro eléctrico o en la red.

Multiactividad entre las regiones

Actualmente, Huawei Cloud DCS no admite multiactividad entre regiones porque Redis no tiene una solución madura de actividad-actividad. **Actividad-actividad es diferente de recuperación ante desastres o HA principal/en standby.**

No se puede lograr Redis activo-activo entre las nubes o las regiones porque los protocolos de serialización de REdis (RESP) personalizados no están unificados. Si se requiere activo-activo, se puede implementar mediante **escritura dual en el extremo de la aplicación**.

Figura 6-7 Escritura dual en el lado de la aplicación para lograr multi-actividad



Nota:

1. La solución de escritura dual **no puede garantizar la coherencia de la caché** (debido a problemas de red). Las **aplicaciones** deben **tolerar la incoherencia de la caché** (estableciendo el tiempo de vida para lograr una eventual consistencia). Si las aplicaciones requieren una fuerte coherencia de la caché, esta solución no es adecuada. Actualmente, no existe una solución en la industria para garantizar una sólida consistencia de la caché entre regiones.

2. Se recomienda realizar operaciones en la caché L2 entre regiones en modo asincrónico.

7 Diferencias del motor de caché

7.1 Comparación entre las versiones de Redis

Al crear una instancia de DCS compatible con Redis, puede seleccionar la versión del motor de caché y el tipo de instancia.

NOTA

DCS for Redis 3.0 ya no se proporciona. Puede utilizar DCS for Redis 4.0 o 5.0 en su lugar.

- **Versión**

  DCS soporta Redis 5.0, 4.0 y 3.0. **Tabla 7-1** describe las diferencias entre estas versiones. Para obtener más información sobre las nuevas características de Redis 4.0 y 5.0, consulte **Nuevas características de DCS for Redis 4.0** y **Nuevas características de DCS for Redis 5.0**.

Tabla 7-1 Diferencias entre versiones de Redis

Característica	Redis 3.0	Redis 4.0 & Redis 5.0	Redis 6.0
Compatibilidad con software de código abierto:	Redis 3.0.7	Redis 4.0.14 y 5.0.9, respectivamente	Edición básica: Redis 6.2.7 Edición profesional: KeyDB 6.0.16
Modo de implementación de instancia	Basado en las máquinas virtuales	Containerizado basado en los servidores físicos	Containerizado basado en los servidores físicos
Arquitectura de CPU	x86	x86 y Arm	x86

Característica	Redis 3.0	Redis 4.0 & Redis 5.0	Redis 6.0
Tiempo necesario para crear una instancia	3 ‑ 15 minutos Instancia de clúster: 10 ‑ 30 minutos	8 segundos	8 segundos
QPS	100,000 QPS por nodo	100,000 QPS por nodo	300,000 QPS por nodo
Acceso a la red pública	Soportado	No soportado	No soportado
Conexión de nombre de dominio	Compatible con VPC	Compatible con VPC	Compatible con VPC
Gestión de datos visualizados	No soportado	Proporciona CLI web para el acceso a Redis y la gestión de datos.	Proporciona CLI web para el acceso a Redis y la gestión de datos.
Tipo de instancia	Nodo único, principal/en standby y Clúster Proxy	Nodo único, principal/en standby, Clúster Proxy y Clúster Redis	Nodo único y principal/en standby
Memoria total de instancia	Rango de 2 GB a 1024 GB.	Las especificaciones regulares van desde 2 GB hasta 1024 GB. Las pequeñas especificaciones de 128 MB, 256 MB, 512 MB y 1 GB también están disponibles para las instancias de nodo único y principal/en standby.	4 GB, 8 GB, 16 GB, 32 GB y 64 GB
Ampliación/ reducción de la capacidad	Ampliación y reducción de la capacidad en línea	Ampliación y reducción de la capacidad en línea	Ampliación y reducción de la capacidad en línea
Respaldo y restauración	Compatible con las instancias principales/en standby y de Clúster Proxy	Compatible con las instancias de separación de lectura/ escritura, principal/standby, de Clúster Proxy y de Clúster Redis	Compatible con las instancias principal/en standby

NOTA

Las arquitecturas subyacentes varían según la versión de Redis. Una vez que se elige una versión de Redis, no se puede cambiar. Por ejemplo, no puede actualizar una instancia de DCS Redis 3.0 a Redis 4.0 o 5.0. Si necesita una versión de Redis superior, cree una nueva instancia que cumpla con sus requisitos y, a continuación, migre los datos de la instancia antigua a la nueva.

- **Tipo de instancia**

  DCS proporciona los tipos de instancia de nodo único, principal/en standby, de Clúster Proxy, de Clúster Redis y de separación de lectura/escritura. Para obtener más información sobre sus arquitecturas y escenarios de aplicación, consulte **Tipos de instancia de DCS**.

7.2 Comparación entre Redis y Memcached

Redis y Memcached son bases de datos en memoria de código abierto populares que son fáciles de usar y proporcionan un mayor rendimiento que las bases de datos relacionales.

¿Cómo puedo seleccionar entre las dos bases de datos clave-valor?

Memcached es adecuado para almacenar estructuras de datos simples, mientras que Redis es adecuado para almacenar datos más complejos y más grandes que requieren persistencia.

Para más detalles, mira la siguiente tabla.

Tabla 7-2 Diferencias entre Redis y Memcached

Artículo	Redis	Memcached
Latencia	Base de datos en memoria con la latencia de submilisegundos	Base de datos en memoria con la latencia de submilisegundos
Facilidad de uso	Sintaxis sencilla y fácil de usar	Sintaxis sencilla y fácil de usar
Almacenamiento distribuido	Expansión horizontal en modo clúster	Soportado
Cliente multilingüe	Admite conexiones de clientes en más de 30 idiomas, incluidos Java, C y Python.	Admite conexiones de clientes en más de 10 idiomas, incluidos Java, C y Python.
Hilo/Proceso	Un solo núcleo y un solo hilo Comunicación de subproceso único, evitando la conmutación y la contención innecesarias del contexto La E/S sin bloqueo (multiplexación de E/S) se utiliza para reducir el consumo de recursos cuando se conectan varios clientes.	Multi-hilo y escalable El rendimiento de Memcached se puede mejorar aumentando el número de CPUs. Hay una ventaja de rendimiento obvia en el escenario donde el valor de la clave es grande.

Artículo	Redis	Memcached
Almacenamiento persistente	Soportado Cada operación de escritura (agregar, eliminar o modificar datos) se puede grabar en disco (archivo AOF).	Soportado **NOTA** La persistencia no es compatible con Memcached de código abierto, pero es compatible con Huawei Cloud DCS for Memcached.
Estructura de datos	Soporta estructuras de datos complejas como hash, lista, conjunto y conjunto ordenado, atendiendo a varios escenarios.	Soporta strings simples.
Soporte de scripts Lua	Soportado	No soportado
Copia de respaldo de la instantánea	Soportado Las instantáneas se generan periódicamente. Por lo tanto, no hay garantía de que los datos no se pierdan. Redis bifurca un subproceso para generar instantáneas. Cuando hay una gran cantidad de datos, el servicio Redis puede interrumpirse durante un corto tiempo.	No soportado
Migración de datos	Soportado Los datos se pueden realizar copias de seguridad y migrar a una nueva instancia de Redis mediante la restauración de instantáneas RDB o la reproducción de archivos AOF.	No soportado
Restricción del valor clave	El valor de una clave puede ser de hasta 1 GB.	1 MB
Bases de datos múltiples	Una instancia de DCS Redis de un solo nodo o principal/en standby admite hasta 256 bases de datos de Redis. Una instancia de Clúster Proxy o de Clúster Redis soporta solo una base de datos, es decir, DB0.	No soportado

En base a la comparación anterior, tanto Redis como Memcached son fáciles de usar y tienen un alto rendimiento. Sin embargo, Redis y Memcached son diferentes en cuanto al almacenamiento de estructura de datos, persistencia, respaldo, migración y compatibilidad con scripts. Se recomienda seleccionar el motor de caché más adecuado en función de los escenarios reales de la aplicación.

NOTA

Memcached es adecuado para escenarios de almacenamiento en caché de pequeña cantidad de datos estáticos, donde los datos solo se leen sin más cómputo y procesamiento, por ejemplo, fragmentos de código HTML.

Redis tiene las estructuras de datos más ricas y los escenarios de aplicación más amplios.

8 Infografía para comparar DCS for Redis con Redis de código abierto

9 Comparación entre servicios DCS y servicios de caché de código abierto

DCS admite las instancias de nodo único, principal/en standby y de clúster, lo que garantiza un alto rendimiento de lectura/escritura y un acceso rápido a los datos. También soporta varias operaciones de gestión de instancias para facilitar su O&M. Con DCS, solo necesita centrarse en la lógica del servicio, sin preocuparse por los problemas de implementación, monitoreo, escalado, seguridad y recuperación de fallas.

DCS es compatible con Redis y Memcached de código abierto, y se puede personalizar según sus requisitos. Esto hace que DCS características únicas además de las ventajas de las bases de datos de caché de código abierto.

DCS for Redis vs. Redis de código abierto

Tabla 9-1 Diferencias entre DCS for Redis y Redis de código abierto

Característica	Redis de código abierto	DCS for Redis
Despliegue del servicio	Requiere de 0.5 a 2 días para preparar servidores.	● Se puede crear una instancia de DCS Redis 3.0 en 5 a 15 minutos. ● Las instancias de DCS Redis 4.0, 6.0, y 5.0 están en contenedores y se pueden crear en 8 segundos.
Versión	-	Sigue de cerca las tendencias de código abierto y soporta la última versión de Redis. Actualmente, Redis 6.0, 5.0, 4.0 y 3.0 son compatibles.
Seguridad	La seguridad de la red y del servidor es responsabilidad del usuario.	● La seguridad de la red se garantiza mediante las VPC y los grupos de seguridad de Huawei Cloud. ● La confiabilidad de los datos está garantizada por la replicación de datos y el backup programado.
Rendimiento	-	100,000 QPS por nodo. DCS for Redis 6.0 puede alcanzar 300,000 QPS por nodo.

Característica	Redis de código abierto	DCS for Redis
Monitoreo	Proporciona solo estadísticas básicas.	Proporciona más de 30 métricas de monitoreo y umbral de alarma personalizable y políticas. ● Diversas métricas: – Las métricas externas incluyen el número de comandos, operaciones simultáneas, conexiones, clientes y conexiones denegadas. – Las métricas de uso de recursos incluyen uso de CPU, uso de memoria física, rendimiento de entrada de red y rendimiento de salida de red. – Las métricas internas incluyen el uso de la capacidad de instancia, así como el número de claves, claves caducadas, canales de PubSub y patrones de PubSub y aciertos de espacio de claves. ● Umbrales de alarma personalizados y políticas para diferentes métricas para ayudar a identificar fallas de servicio.
Respaldo y restauración	Soportado	● Soporta copias de seguridad programadas y manuales. Los archivos de copia de seguridad se pueden descargar. ● Los datos de copia de seguridad se pueden restaurar en la consola.
Gestión de parámetros	Sin gestión de parámetros visualizados	● La gestión de parámetros visualizados es compatible con la consola. ● Los parámetros de configuración se pueden modificar en línea. ● Se puede acceder a los datos y modificarlos en la consola.
Escalamiento vertical	Interrumpe los servicios e implica un procedimiento complejo, desde modificar la memoria RAM del servidor hasta modificar la memoria Redis y reiniciar el SO y los servicios.	● Soporta escalamiento en línea y escalamiento en línea sin interrumpir los servicios. ● Las especificaciones se pueden escalar hacia arriba o hacia abajo dentro de la gama disponible en función de los requisitos de servicio.
O&M	Manual O&M	Servicios O&M de extremo a extremo las 24 horas de los 7 días

DCS for Memcached vs. Memcached de código abierto

Tabla 9-2 Diferencias entre DCS for Memcached y Memcached de código abierto

Característica	Memcached de código abierto	DCS for Memcached
Despliegue del servicio	Requiere de 0.5 a 2 días para preparar servidores.	Crea una instancia en 5 a 15 minutos.
Seguridad	La seguridad de la red y del servidor es responsabilidad del usuario.	● La seguridad de la red se garantiza mediante las VPC y los grupos de seguridad de Huawei Cloud. ● La confiabilidad de los datos está garantizada por la replicación de datos y el backup programado.
Rendimiento	-	100,000 QPS por nodo
Monitoreo	Proporciona solo estadísticas básicas.	Proporciona más de 30 métricas de monitoreo y umbral de alarma personalizable y políticas. ● Diversas métricas: – Las métricas externas incluyen el número de comandos, operaciones simultáneas, conexiones, clientes y conexiones denegadas. – Las métricas de uso de recursos incluyen uso de CPU, uso de memoria física, rendimiento de entrada de red y rendimiento de salida de red. – Las métricas internas incluyen el uso de la capacidad de instancia, así como el número de claves, claves caducadas, canales de PubSub y patrones de PubSub y aciertos de espacio de claves. ● Umbrales de alarma personalizados y políticas para diferentes métricas para ayudar a identificar fallas de servicio.
Respaldo y restauración	No soportado	● Soporta copias de seguridad programadas y manuales. ● Los datos de copia de seguridad se pueden restaurar en la consola.
Mantenimiento visualizado	Sin gestión de parámetros visualizados	● La gestión de parámetros visualizados es compatible con la consola. ● Los parámetros de configuración se pueden modificar en línea.

Característica	Memcached de código abierto	DCS for Memcached
Escalamiento vertical	Interrumpe los servicios e implica un procedimiento complejo, desde modificar la memoria RAM del servidor hasta modificar la memoria Redis y reiniciar el SO y los servicios.	● Soporta escalamiento en línea sin interrumpir los servicios. ● Las especificaciones se pueden escalar hacia arriba o hacia abajo dentro de la gama disponible en función de los requisitos de servicio.
O&M	Manual O&M	Servicios O&M de extremo a extremo las 24 horas de los 7 días
Persistencia de datos	No soportado	Compatible con las instancias principal/en standby

10 Notas y restricciones

Cuenta y Cuota

- Una cuenta puede crear una instancia DCS solo después de pasar la autenticación de nombre real.
- Solo puede crear un cierto número de instancias con cierta memoria total de forma predeterminada. Las cuotas son diferentes por usuario y por región. Consulte la cuota que se muestra en la consola. Para obtener más información acerca de las cuotas, vea **Cuotas**. Para aumentar la cuota, envíe un **ticket de servicio**.

Red

Una nube pública utiliza VPC para gestionar la seguridad de red de los servicios en la nube.

- El cliente debe implementarse en un ECS que pertenezca a la misma VPC que la instancia de DCS para Redis o para Memcached.
- Para una instancia de DCS para Redis 3.0 o para Memcached, seleccione el mismo grupo de seguridad para la instancia de DCS y el ECS donde se implementa el cliente. Si la instancia de DCS y de ECS pertenecen a diferentes grupos de seguridad, agregue las reglas entrantes y salientes para los grupos de seguridad. Para obtener más información sobre cómo agregar las reglas, vea **¿Cómo configuro un grupo de seguridad?**
- Para una instancia de DCS compatible con Redis 4.0 o 5.0, agregue la dirección IP del ECS donde se implementa el cliente a la lista blanca de la instancia. Para obtener más información, consulte **Gestión de la lista blanca de direcciones IP**.
- Para una instancia de DCS compatible con Redis 6.0, agregue la dirección IP del ECS donde se implementa el cliente a la lista blanca de la instancia. Para obtener más información, consulte **Gestión de la lista blanca de direcciones IP**.

Ajuste de escala

Puede escalar una instancia solo dentro del ámbito de cuota restante. Si su cuota es insuficiente, envíe un **ticket de servicio** para aumentar la cuota.

11 Facturación

DCS admite los modos de pago por uso y de facturación anual/mensual. Para obtener más información, consulte **Detalles de precios del producto**.

NOTA

Actualmente, solo la región CN-Hong Kong admite la facturación anual/mensual. Si necesita utilizar este modo de facturación en otras regiones, envíe un ticket de servicio en la consola para solicitar al personal técnico que active la función en el fondo.

Conceptos de facturación:

El uso de DCS se factura según la especificación de instancia de DCS.

Concepto de facturación	Descripción
Instancia de DCS	Facturación basada en las especificaciones de instancia de DCS.

Nota: Los cargos de DCS de Huawei Cloud se basan en las especificaciones de instancia de DCS seleccionadas en lugar de la capacidad de caché real.

Modos de facturación

DCS ofrece dos modos de facturación: pago por uso y anual/mensual. Se recomienda pagar por uso si no está seguro de sus necesidades futuras de servicio y desea evitar pagar por los recursos no utilizados. Sin embargo, si está seguro de sus necesidades, anual/mensual será menos costoso.

- Anual/Mensual: Ofrece un descuento mayor que el modo de pago por uso y se recomienda para usuarios a largo plazo.
- Pago por uso (por hora): puede iniciar y detener las instancias de DCS según sea necesario y se le facturará en función de la duración de su uso de las instancias DCS. La facturación se inicia cuando se crea una instancia de DCS y finaliza cuando se elimina la instancia de DCS. La unidad de tiempo mínima es de un segundo.
- Puede cambiar entre los modos anual/mensual y de pago por uso.

Cambios de configuración

Puede cambiar las especificaciones de una instancia de DCS Redis o Memcached, es decir, escalar o reducir una instancia y cambiar el tipo de instancia de nodo único a principal/en standby. Después de cambiar correctamente las especificaciones, la instancia se factura en función de las nuevas especificaciones. Para obtener más información, consulte **Modificación de la especificaciones de la instancia de DCS**.

Renovación

Cuando se caduca un paquete de recursos, puede renovarlo o establecer reglas de renovación automática para ello. Para obtener más información acerca de cómo renovar paquetes de recursos, consulte **Gestión de renovación**.

FAQ

Para obtener más información sobre la facturación de DCS, consulte **FAQ sobre compras y facturación**.

12 Gestión de permisos

Si necesita asignar diferentes permisos a los empleados de su empresa para acceder a sus recursos de DCS, IAM es una buena opción para la gestión de permisos detallada. IAM proporciona autenticación de identidad, gestión de permisos y control de acceso, lo que le ayuda a proteger el acceso a sus recursos de Huawei Cloud.

Con IAM, puede usar su cuenta de Huawei Cloud para crear usuarios de IAM para sus empleados y asignar permisos a los usuarios para controlar su acceso a tipos de recursos específicos. Por ejemplo, algunos desarrolladores de software de su empresa necesitan usar recursos de DCS, pero no deben poder eliminar instancias de DCS ni realizar otras operaciones de alto riesgo. En este escenario, puede crear usuarios de IAM para los desarrolladores de software y concederles solo los permisos necesarios para usar los recursos de DCS.

Si su cuenta de Huawei Cloud no requiere usuarios individuales de IAM para la administración de permisos, omita esta sección.

IAM se puede utilizar de forma gratuita. Solo paga por los recursos de su cuenta. Para obtener más información acerca de IAM, consulte **Descripción general del servicio IAM**.

Permisos de DCS

De forma predeterminada, los nuevos usuarios de IAM no tienen permisos asignados. Debe agregar un usuario a uno o más grupos y adjuntar políticas o roles de permisos a estos grupos. Los usuarios heredan permisos de los grupos a los que se agregan y pueden realizar operaciones específicas a servicios en la nube según los permisos.

DCS es un servicio a nivel de proyecto implementado y accedido en regiones físicas específicas. Para asignar permisos de DCS a un grupo de usuarios, especifique el ámbito como proyectos específicos de la región y seleccione proyectos para que los permisos surtan efecto. Si se selecciona **All projects**, los permisos surtirán efecto para el grupo de usuarios en todos los proyectos específicos de la región. Al acceder a DCS, los usuarios deben cambiar a una región en la que se les haya autorizado a usar este servicio.

Puede conceder permisos a los usuarios mediante roles y políticas.

- Roles: Tipo de mecanismo de autorización de grano grueso que define permisos relacionados con las responsabilidades del usuario. Este mecanismo proporciona solo un número limitado de roles de nivel de servicio para la autorización. Al usar roles para conceder permisos, también debe asignar otros roles de los que dependen los permisos para que surtan efecto. Sin embargo, los roles no son una opción ideal para la autorización detallada y el control de acceso seguro.

- Políticas: Un tipo de mecanismo de autorización detallado que define los permisos necesarios para realizar operaciones en recursos de nube específicos bajo ciertas condiciones. Este mecanismo permite una autorización más flexible basada en políticas, cumpliendo los requisitos para un control de acceso seguro. Por ejemplo, puede conceder a los usuarios de DCS solo los permisos para las instancias de DCS operativas. La mayoría de las políticas definen permisos basados en API. Para ver las acciones de API admitidas por DCS, consulte **Políticas de permisos y acciones admitidas**.

En la **tabla 1** se enumeran todas las funciones y políticas definidas por el sistema compatibles con DCS.

Tabla 12-1 Roles definidos por el sistema y políticas admitidas por DCS

Nombre de rol/ política	Descripción	Tipo	Dependencia
DCS FullAccess	Todos los permisos para DCS. Los usuarios con estos permisos pueden operar y usar todas las instancias de DCS.	Política definida por el sistema	Ninguna
DCS UserAccess	Permisos de usuario comunes para DCS, excluyendo los permisos para crear, modificar, eliminar instancias DCS y modificar especificaciones de instancia.	Política definida por el sistema	Ninguna
DCS ReadOnlyAccess	Permisos de sólo lectura para DCS. Los usuarios a los que se han concedido estos permisos sólo pueden ver los datos de instancia de DCS.	Política definida por el sistema	Ninguna
DCS Administrator	Permisos de administrador para DCS. Los usuarios con estos permisos pueden operar y usar todas las instancias de DCS.	Rol definido por el sistema	Los roles **Server Administrator** y **Tenant Guest** deben asignarse en el mismo proyecto.

NOTA

La política de **DCS UserAccess** es diferente de la política de **DCS FullAccess**. Si configura ambas instancias, no podrá crear, modificar, eliminar ni escalar instancias de DCS porque las instrucciones de denegación tendrán prioridad sobre las instrucciones permitidas.

El **tabla 2** enumera las operaciones comunes soportadas por cada política de sistema de DCS. Elija las políticas de sistema adecuadas de acuerdo con esta tabla.

Tabla 12-2 Operaciones comunes apoyadas por cada política de sistema

Operación	DCS FullAccess	DCS UserAccess	DCS ReadOnlyAccess	DCS Administrator
Modificación de parámetros de configuración de instancia	√	√	×	√
Eliminación de tareas en segundo plano	√	√	×	√
Acceso a instancias mediante Web CLI	√	√	×	√
Modificación del estado de ejecución de instancia	√	√	×	√
Ampliación de la capacidad de instancia	√	×	×	√
Cambio de contraseñas de instancia	√	√	×	√
Modificación de instancias DCS	√	×	×	√
Realización de una conmutación principal/en standby	√	√	×	√
Copia de seguridad de datos de instancia	√	√	×	√

Operación	DCS FullAccess	DCS UserAccess	DCS ReadOnlyAccess	DCS Administrator
Análisis de claves grandes o claves de acceso rápido	√	√	×	√
Creación de instancias DCS	√	×	×	√
Eliminación de archivos de copia de seguridad de instancia	√	√	×	√
Actualización de la versión de instancia	√	√	×	√
Restauración de datos de instancia	√	√	×	√
Reajuste de la contraseñas de instancia	√	√	×	√
Migración de datos de instancia	√	√	×	√
Descarga de datos de copia de seguridad de instancia	√	√	×	√
Eliminación de instancias de DCS	√	×	×	√

Operación	DCS FullAccess	DCS UserAccess	DCS ReadOnlyAccess	DCS Administrator
Consulta de los parámetros de configuración de instancia	√	√	√	√
Consulta de registros de restauración de instancia	√	√	√	√
Consulta de registros de copia de seguridad de instancia	√	√	√	√
Consulta de instancias de DCS	√	√	√	√
Consultar tareas en el fondo de instancia	√	√	√	√
Consultar información de actualización de instancia	√	√	√	√
Consultar todas las instancias	√	√	√	√
Funcionamiento de consultas lentas	√	√	√	√

Enlaces útiles

- **Descripción general del servicio IAM**
- **Creación de un usuario y concesión de permisos DCS**

- **Políticas de permisos y acciones admitidas**

13 Conceptos básicos

Instancia de DCS

Una instancia es la unidad de recurso mínima proporcionada por DCS.

Puede seleccionar el motor de caché de Redis o de Memcached. Los tipos de instancia pueden ser de nodo único, principal/en standby o de clúster. Para cada tipo de instancia, hay varias especificaciones disponibles.

Para obtener más información, consulte **Especificaciones de instancias de DCS** y **Tipos de instancia de DCS**.

Proyecto

Los proyectos se utilizan para agrupar y aislar recursos de OpenStack (recursos de cómputo, recursos de almacenamiento y recursos de red). Un proyecto puede ser un departamento o un equipo de proyecto. Se pueden crear varios proyectos para una cuenta.

Réplica

Una réplica es un nodo de una instancia de DCS. Una instancia de réplica única no tiene nodo en standby. Una instancia de dos réplicas tiene un nodo principal y un nodo en standby. Por defecto, cada instancia principal/en standby y cada partición de una instancia de Clúster Redis tienen dos réplicas. Por ejemplo, si el número de réplicas se establece en tres para una instancia principal/en standby, la instancia tiene un nodo principal y dos nodos en standby.

Acceso a la red pública

Un EIP puede estar enlazado a una instancia de DCS compatible con Redis 3.0. Puede acceder a la instancia a través de clientes mediante el EIP. Las instancias de DCS compatible con Redis 4.0, 6.0, and 5.0 no admiten el acceso público.

Stunnel se utiliza para cifrar el contenido de comunicación en el acceso a la red pública. El retardo de la red es ligeramente mayor que el de la VPC, por lo que el acceso a la red pública es adecuado para la puesta en marcha local en la fase de desarrollo.

Para obtener más información, consulte las **Instrucciones de acceso público**.

Acceso sin contraseña

Se puede acceder a las instancias de DCS Redis y Memcached en la VPC sin contraseñas. La latencia es menor porque no hay autenticación de contraseña.

Puede habilitar el acceso sin contraseña para instancias que no tienen datos confidenciales. Para garantizar la seguridad de los datos, no se le permite habilitar el acceso sin contraseña para instancias habilitadas con acceso a red pública.

Para obtener más información, consulte **Habilitar el acceso sin contraseña a una instancia de DCS Redis**.

Ventana de tiempo de mantenimiento

La ventana de tiempo de mantenimiento es el período en el que el equipo de servicio de DCS actualiza y mantiene la instancia.

El mantenimiento de la instancia de DCS tiene lugar solo una vez cada trimestre y no interrumpe los servicios. Aun así, se recomienda seleccionar un período de tiempo cuando la demanda de servicio es baja.

Al crear una instancia, debe especificar una ventana de tiempo de mantenimiento, que se puede modificar después de crear la instancia.

Para obtener más información, consulte: **Modificación de la ventana de tiempo de mantenimiento de una instancia**

Despliegue entre AZ

Las instancias Principal/En standby se implementan en diferentes AZ con fuentes de alimentación y redes aisladas físicamente. Las aplicaciones también se pueden implementar en AZ para lograr HA tanto para datos como para aplicaciones.

Al crear una instancia principal/en standby de DCS Redis o Memcached, puede seleccionar **Cross-AZ Deployment** y seleccionar una AZ para el nodo en standby.

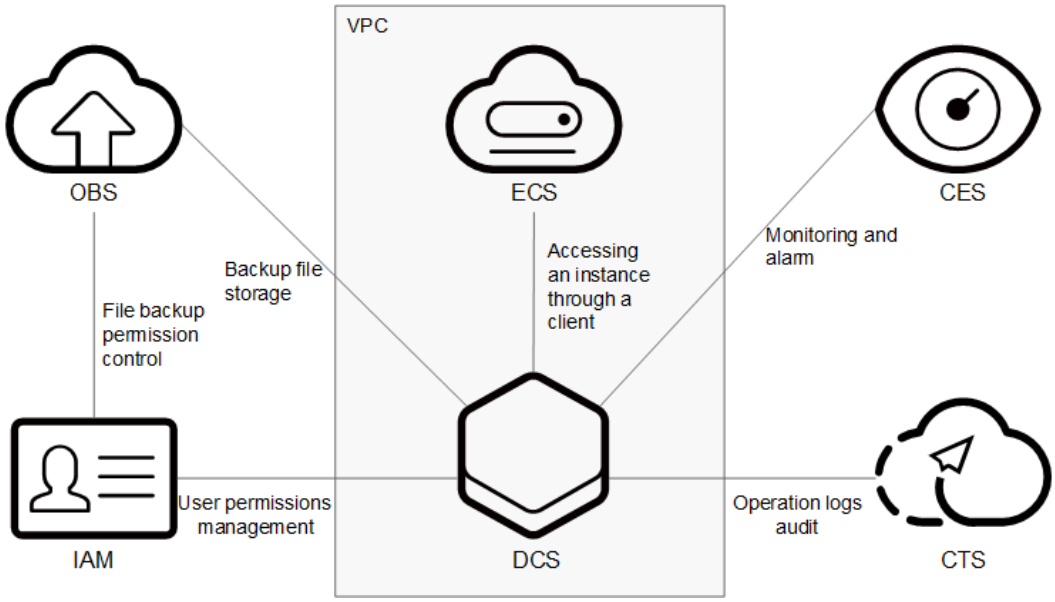
Partición

Una partición es una unidad de gestión de una instancia de clúster de DCS Redis. Cada partición corresponde a un proceso de redis-servidor. Un clúster consta de varias particiones. Y cada partición tiene varias ranuras. Los datos se almacenan de forma distribuida en las ranuras. El uso de particiones aumenta la capacidad de caché y las conexiones simultáneas.

14 Servicios relacionados

DCS se utiliza junto con otros servicios en Huawei Cloud, incluidos VPC, ECS, IAM, Cloud Eye, CTS y Object Storage Service (OBS).

Figura 14-1 Relaciones entre DCS y otros servicios



VPC

Una VPC es un entorno de red virtual aislado en Huawei Cloud. Puede configurar intervalos de direcciones IP, subredes y grupos de seguridad, asignar EIP y asignar ancho de banda en una VPC.

DCS se ejecuta en las VPC. El servicio de VPC gestiona los EIP y ancho de banda, y proporciona grupos de seguridad. Puede configurar reglas de acceso para grupos de seguridad para proteger el acceso a DCS.

ECS

Elastic Cloud Server (ECS) es un servidor en la nube que ofrece recursos de cómputo bajo demanda y escalables para aplicaciones seguras, flexibles y eficientes.

Puede acceder y gestionar sus instancias de DCS mediante un ECS.

IAM

IAM proporciona la autenticación de identidad, la gestión de permisos y el control de acceso.

Con IAM, puede controlar el acceso a DCS.

Cloud Eye

Cloud Eye es un servicio de monitoreo seguro, escalable e integrado. Con Cloud Eye, puede supervisar su servicio DCS y configurar reglas de alarma y notificaciones.

Cloud Trace Service (CTS)

CTS le proporciona un historial de operaciones realizadas en recursos de servicios en la nube. Con CTS, puede consultar, auditar y realizar operaciones de retroceso. Las trazas incluyen las solicitudes de operación enviadas mediante la consola de gestión o las API abiertas y los resultados de estas solicitudes.

OBS

OBS proporciona un servicio de almacenamiento seguro y rentable utilizando objetos como unidades de almacenamiento. Con OBS, puede almacenar y gestionar el ciclo de vida de grandes cantidades de datos.

Puede almacenar archivos de copia de seguridad de instancia de DCS en OBS.

