GeminiDB Redis

User Guide

Issue 01

Date 2025-01-30

HUAWEI

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: https://www.huawei.com

Email: support@huawei.com

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process.* For details about this process, visit the following web page:
https://www.huawei.com/en/psirt/vul-response-process
For vulnerability information, enterprise customers can visit the following web page:
https://securitybulletin.huawei.com/enterprise/en/security-advisory

Contents

1 Service Overview

1.1 Enterprise-Level Features

GeminiDB Redis API uses a cloud-native distributed architecture designed to decouple storage resources from compute resources and is fully compatible with Redis 7.0, 6.2 (including 6.2.*X*), 5.0 and earlier editions, so it provides more enterprise-grade features.

- **Exclusive resources, no traffic limiting for shards**
  - Compute nodes are deployed in exclusive containers. Tenants are isolated from each other, ensuring high stability. In the case of high concurrent traffic, traffic on nodes is unlimited.
  - Built-in exclusive load balancers provide higher forwarding performance and stability.
  - Public IP addresses can be bound to compute nodes, facilitating cloud migration and remote debugging.
- **Second-level auto scaling, easily coping with service peaks and valleys**
  - Independent scaling is supported for storage and compute resources. A single instance supports tens of millions of QPS and dozens of TB of storage space.

  - In scenarios where data volume increases, capacity expansions can be performed with a few clicks, without affecting service applications.
  - In scenarios where workloads increase suddenly (for example, gaming and e-commerce activities have higher QPS requirements), you can expand the capacity by adding nodes or increasing specifications. In the future, you can easily reduce the capacity. Only second-level reconnection may occur to services.
- **A unified database used to simplify the service architecture**
  - With high-performance storage pools, instances automatically load frequently accessed hot data to the memory of compute nodes and exchange cold and hot data internally. Services preferentially read hot data from the memory, ensuring high data reliability and low latency.
  - GeminiDB Redis API is suitable for storing ever-increasing important service data (such as game player data, user profiles, behavior logs, and article information). Compared with the Redis+MySQL architecture, it has a simpler architecture, more reliable data storage, and higher comprehensive performance and cost-effectiveness.
- **3-AZ deployment**
  - 3-AZ instances allow compute and storage resources to be evenly distributed across the AZs. The deployment rules strictly comply with anti-affinity groups, delivering ultra-high reliability.
  - If a node is faulty, services can be taken over in seconds. With the dedicated, decoupled storage and compute, GeminiDB Redis API provides fault tolerance (*N*-1 reliability) to allow you to restore service access in seconds, achieving ultra-high availability.
- **Account management for database-level permission control**
  - A maximum of 65,536 databases can be used, and up to 200 subaccounts can be created.
  - You cannot only set read-only or read/write permissions for sub-accounts, but also configure accessible databases for sub-accounts, preventing misoperations between tenants.
- **Setting an expiration time for each field of a hash key**
  - Open-source Redis supports only the setting of the expiration time for all hash keys. GeminiDB Redis API adds a group of hash commands. This allows you to set an expiration time for a specified field in a hash key and implement the elimination logic at the service layer in the database, simplifying the service architecture.
  - For details about the best practices of EXHASH, see **6.4 EXHASH for Ad Frequency Control**.
- **Strong data consistency, preventing dirty reads**
  - Open-source Redis adopts asynchronous replication, and data copies are weakly consistent. In common service scenarios where counters, rate limiters, and distributed locks are used, dirty reads may occur, which may cause service logic disorder.
  - GeminiDB Redis API connects data copies to a high-performance storage pool. Once data is successfully written into the storage pool, the three copies of data are stored consistently, preventing dirty reads from occurring in subsequent service access.

- **Enhanced transactions**
  - The transaction MULTI/EXEC is supported. Compared with the open-source Redis, GeminiDB Redis API has transactions complying with the ACID feature. It supports rollback from the bottom of its architecture, to meet transaction atomicity.
- **Better prefix scanning**
  - When you run a SCAN command, for example, **match prefix***, on an instance, the scanning performance is much higher than that of open-source Redis. GeminiDB Redis API optimizes the complexity of delivering commands to O(log$N$ + M), where N indicates the overall data volume and M indicates the matched data volume. The scanning complexity of open-source Redis is O(N), which is slower.
- **Real-time persistence**
  - GeminiDB Redis API uses the persistence mechanism of write-ahead logging (WAL) to ensure data atomicity and durability. To ensure write performance, a response is returned immediately after data is written to the OS buffer. Data is written to disks in real time asynchronously to ensure real-time persistence and high-speed low-latency writes.

1.2 Highlights

Cloud-native GeminiDB is a key-value (KV) database service featuring high stability, cost-effectiveness, elasticity, and easy O&M. It is fully compatible with the Redis protocol, supports advanced functions such as PITR recoveries for game rollback and FastLoad for feature data import, and it allows you to set the field expiration time for hash keys and blacklist for high-risk keys.

GeminiDB is widely used in scenarios such as game friends list and player rankings, ad placement, personalized recommendations, e-commerce inventory, IoV data storage, and ERP systems. For details, see **1.4 Application Scenarios**.

GeminiDB has the following advantages over open-source on-premises KV databases (such as Redis and Pika databases):

Table 1-1 Comparison between GeminiDB and open-source on-premises KV databases

Dimension	Item	Open-Source On-Premises KV Database	GeminiDB
Stability	Performance jitter caused by forks	**Service stability is severely affected by fork issues.** When RDB backups are generated, the Append Only File (AOF) is rewritten, or full data is synchronized, a fork is called. This increases latency and causes out of memory (OOM) issues.	**Service stability is improved as fork issues are addressed.** There is no performance jitter during backup and synchronization.
	Long latency in big key scenarios	**The single-thread architecture slows down subsequent requests.** In a single-thread architecture, big key requests slow down all subsequent requests and may trigger flow control or OOM issues on shards.	**The multi-thread architecture reduces the impact on subsequent keys.** GeminiDB uses a multi-thread architecture, which improves concurrency and reduces the impact of big keys on subsequent read and write operations of other keys.
	Bandwidth limiting during peak hours	**Flow control is easily triggered, affecting services.** Open-source on-premises databases typically use a hybrid deployment that strictly limits the bandwidth. Flow control is easily triggered for smaller instances.	**Up to 10 Gbit/s is supported, allowing GeminiDB to handle service surges.** By using an independent container deployment, GeminiDB can enable a load balancer to support a bandwidth of 10 Gbit/s.

Dimension	Item	Open-Source On-Premises KV Database	GeminiDB
	Impact of scale-out on services	**Scale-out can take several minutes or sometimes even hours, greatly affecting services.** Adding nodes involves data migration. Services may be affected for a few minutes or up to several hours.	**Smooth scale-out is supported and has minimal impact on services.** Scale-out can be completed in seconds and without interrupting services. Adding nodes does not requires any data migration. There is just a few seconds of jitter.
	HA scenarios such as node breakdowns and primary/secondary switchovers	**Long switchover time: RTO > 30s**	**Second-level jitters, RTO < 10s**
Performance	QPS	**QPS per shard: 80,000 to 100,000** In a single-thread architecture, the QPS of a single shard does not increase after CPUs are added.	**QPS per shard: 10,000 to 300,000** In a multi-thread architecture, the QPS can increase linearly as CPUs are added.
	Latency	**Low latency**	**Low latency** In most service scenarios, the average latency is 1 ms, and the p99 latency is about 2 ms.
O&M capabilities	Audit logs of risky operations	Not supported	**High-risk commands can be traced.**
	Circuit breakers triggered by abnormal requests to keys	Not supported	**Key blacklists and one-click circuit breakers for high-risk operations are supported, so the entire instance is not affected.**

Dimension	Item	Open-Source On-Premises KV Database	GeminiDB
	Slow query logs	Supported	**Supported. More details can be found in the logs.**
	Big key diagnosis	Not supported	**Online diagnosis of big keys by category is supported.**
	Hot key diagnosis	Supported	Online diagnosis of hot keys is supported.
Cost	Utilization cost	**The in-memory storage is expensive.**	**The cost of databases with the same specifications is 30% lower than open-source on-premises databases.** Users can purchase additional compute resources and storage resources independently to eliminate the resource waste associated with coupled storage and compute.
	Data compression	Not supported	**The compression ratio (4:1) enables databases with the same specifications to store more data.**
	Scale-out	**Coupled storage and compute increases costs exponentially.**	**Decoupled storage and compute supports independent scaling of compute and storage resources.**
Availability	/	**If any pair of primary and standby nodes is faulty, the entire cluster becomes unavailable.**	**GeminiDB provides superlative fault tolerance (N-1 reliability).**

Dimension	Item	Open-Source On-Premises KV Database	GeminiDB
Data reliability	/	**Weak** Thousands or tens of thousands of records will be lost if nodes are restarted and the network fluctuates. Weak data consistency may cause dirty reads.	**High reliability** GeminiDB provides three-copy storage, so it can serve as the primary database to replace the traditional DB+Cache solution, and it also ensures strong data consistency and avoids dirty reads.
Advanced features	Autoscaling	Not supported	Supported
	Setting the expiration time for fields in hashes	Not supported	**Supported. Service design is less complex and concurrency is increased.**
	Fast data loading	Not supported	**FastLoad allows feature data to be imported faster, reducing the impact on online services.**
	Point-In-Time Recovery (PITR)	Not supported	Supported PITR rollbacks and quick data restoration to the original instance are supported, making GeminiDB a great fit for gaming applications.
	DR instances	Not supported	**Intra-region and cross-region DR instances can be created.**

1.3 Product Architecture

GeminiDB Redis supports the following architectures: proxy cluster, Redis Cluster, and primary/standby.

- Both proxy cluster and Redis Cluster instances support horizontal and vertical scaling and can handle millions of QPS and tens of terabytes of data. Redis

Cluster is recommended because it features low latency, high concurrency, and high scalability.

- All instances in a proxy cluster or Redis Cluster can be read and written, improving resource utilization. Shared storage provides high availability. In the primary/standby architecture, only a primary instance can be read and written. Therefore, the cluster architecture is recommended.

The following table lists architecture types and application scenarios.

Type	Description	Architecture	Application Scenarios
Proxy cluster	With a sharded cluster architecture, a cluster instance can be accessed through proxies and is compatible with a single Redis node, Redis Sentinel, and Redis Cluster.	For details, see **Figure 1-1**.	• Advantages: This type is easy to use. Sharding is not a concern. You can use a cluster like a single node. The proxy can distribute your requests to corresponding GeminiDB Redis instances for processing. • Application scenario: The usage logic is simplified, and you do not need to pay much attention to shard management. For example, this type is recommended if you want to migrate data from a single node to a cluster and it is inconvenient to modify code on a client. Redis Cluster can be used to meet higher requirements on concurrency and latency.
Redis Cluster (recommended)	A sharded cluster is created without proxy components and is compatible with native Redis Cluster.	For details, see **Figure 1-2**.	• Advantages: This type of instance does not have a proxy, and delivers higher concurrency of a single shard than the proxy cluster instance. A client is directly connected to shards at lower latency. Up to 128 nodes are supported. • Application scenario: This type is applicable to services that are more sensitive to latency and have higher requirements on concurrency and scalability.

Type	Description	Architecture	Application Scenarios
Primary/ Standby	A primary/standby instance is compatible with a single Redis node and Redis Sentinel.	For details, see **Figure 1-3**.	This type is alternative to Redis master-replica architecture. No application code needs to be refatored. The cluster architecture is recommended for new applications because of its better performance and scalability.

Figure 1-1 Proxy cluster

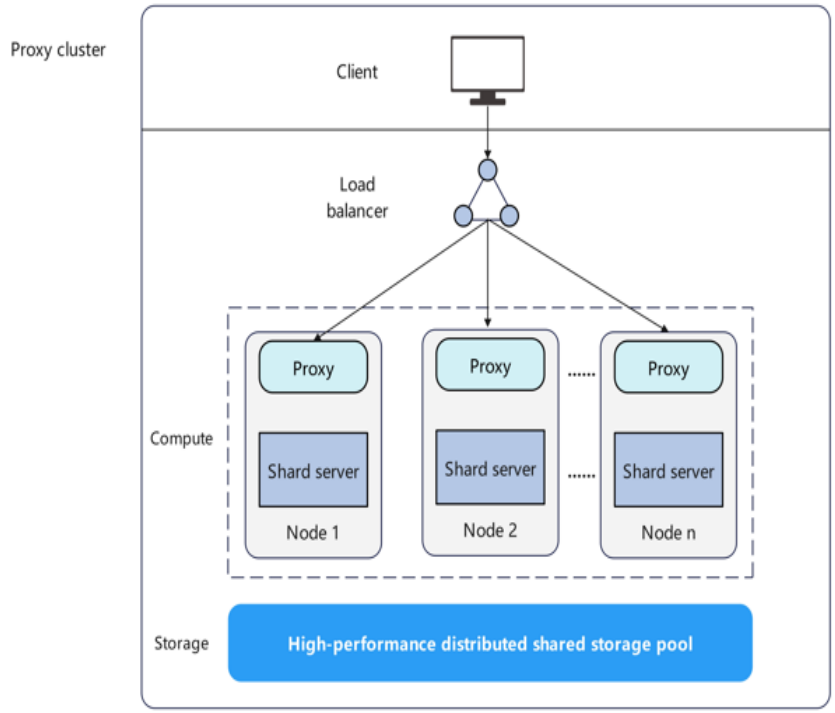


Figure 1-2 Redis Cluster

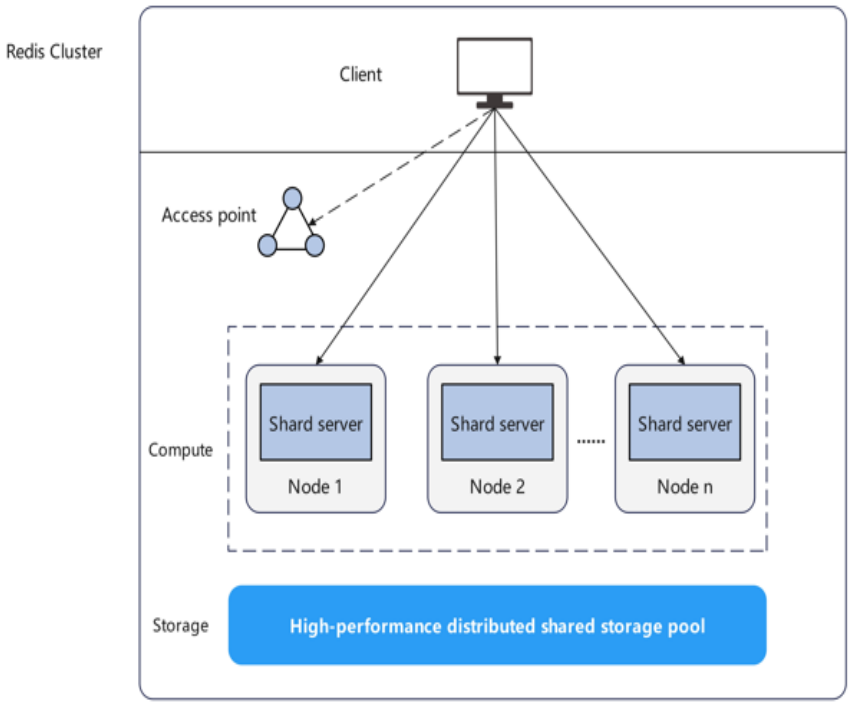
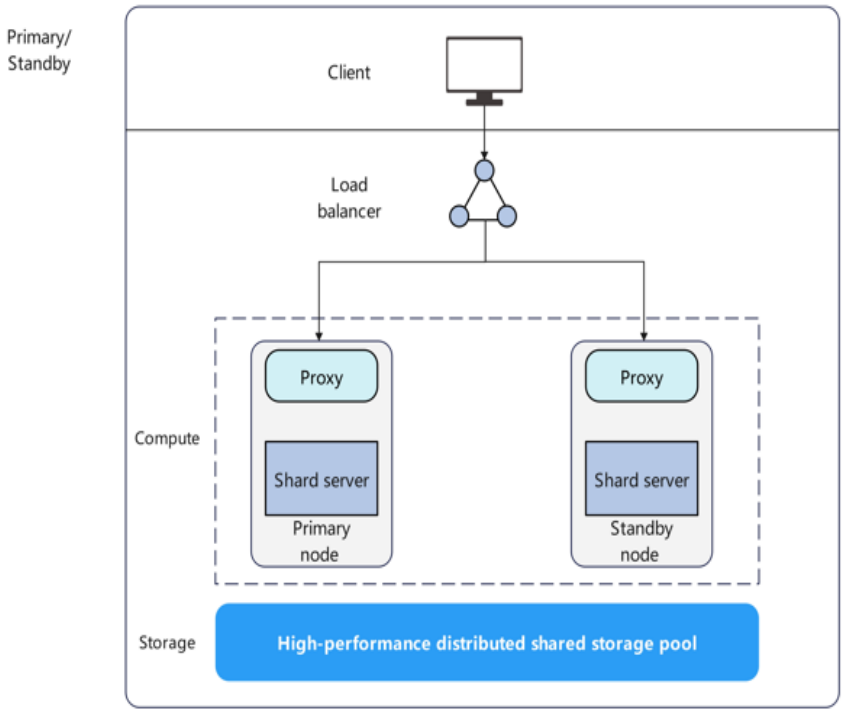


Figure 1-3 Primary/Standby



1.4 Application Scenarios

As a key-value database compatible with Redis APIs, GeminiDB Redis API extends application scenarios of Redis so that it can better meet diversified service requirements such as persistent and hybrid storage.

E-Commerce

- For e-commerce applications, some commodity data is more frequently queried than others. GeminiDB Redis API stores frequently queried commodity information in memory as hot data, and cold data in the shared storage pool. This not only meets the quick access requirements of popular commodities, but also avoid excessive in-memory storage costs
- GeminiDB Redis API can permanently store massive amounts of historical order data of e-commerce applications. It allows you to access data through the Redis API and provides TB-level storage.
- There may be a large number of concurrent access requests within a short period of time during an e-commerce promotion. GeminiDB Redis API works as a front-end cache (large memory required) to help back-end databases handle service peaks. You can easily add compute nodes in seconds to handle the expected peak traffic.

Gaming

- The schema of gaming services is simple. You can select GeminiDB Redis API as a persistent database and use simple Redis APIs to quickly develop and launch services. For example, the sorted set structure of Redis can be used to display game rankings in real time.
- In delay-sensitive gaming scenarios, GeminiDB Redis API can be used as the front-end cache (large memory required) to accelerate access to applications.

Live Streaming

The most-viewed live streaming content usually dominates most traffic of the live streaming applications. GeminiDB Redis API can efficiently use memory resources by retaining popular live streaming data in the memory and other data in the shared storage, reducing your business costs.

Online Education

Online education applications store a large amount of data such as courses and Qs&As. However, only hot data (including most-viewed courses, latest question libraries, and lectures by famous teachers) is frequently accessed. GeminiDB Redis API can save data in memory or shared storage based on data access frequency, achieving a balance between performance and costs.

Persistent Storage for Other Applications

With the rapid development of the Internet, various large-scale applications have increasing requirements for persistent storage. Specifically, a massive amount of

data needs to be stored, including historical orders, feature engineering, log records, location coordinates, machine learning, and user profiles. A common feature of these scenarios is large data volume and long validity period. Therefore, a large-capacity and low-cost key-value storage service is required to collect and transfer data. Redis is the most widely used key-value service. Its various data structures and operation APIs have innate advantages in storing such data. However, the native Redis can only be used as a cache and cannot guarantee persistence.

In addition to compatibility with Redis APIs, GeminiDB Redis API provides large-capacity, low-cost, and high-reliability data storage capabilities, making it well-suited to persistent storage scenarios.

1.5 Compatible APIs and Versions

This section describes the compatible APIs and versions supported by GeminiDB RedisAPI.

Table 1-2 Compatible APIs and versions

Compatible API	Version
Redis	7.0, 6.2 (including 6.2.*X*), 5.0, and earlier versions

1.6 Instance Specifications

This section describes available GeminiDB Redis instance specifications. The instance specifications depend on the selected CPU model.

GeminiDB Redis API allows for cold and hot data exchange and has the ability to handle a capacity that surpasses the limitations of memory. Hot data is stored in the memory, and full data is stored in the high-performance storage pool. The total instance space refers to the total storage capacity, which determines the upper limit of data storage. **Table 1-5** lists the node memory capacity.

Table 1-3 GeminiDB Redis cluster instance specifications (fast configuration)

Instance Type	Storage (GB)	Nodes	Node Flavor	vCPUs	QPS	Max. Connections	Databases	Accounts
Cluster	4	2	geminidb.redis.medium.2	1	20,000	20,000	256	200
	8	2	geminidb.redis.medium.4	1	20,000	20,000	256	200

Instance Type	Storage (GB)	Nodes	Node Flavor	vCPUs	QPS	Max. Connections	Databases	Accounts
	16	2	geminidb.redis.large.4	2	40,000	20,000	256	200
	24	3	geminidb.redis.large.4	2	60,000	30,000	256	200
	32	4	geminidb.redis.large.4	2	80,000	40,000	256	200
	48	3	geminidb.redis.xlarge.4	4	120,000	30,000	1,000	200
	64	4	geminidb.redis.xlarge.4	4	160,000	40,000	1,000	200
	96	3	geminidb.redis.2xlarge.4	8	240,000	30,000	1,000	200
	128	4	geminidb.redis.2xlarge.4	8	320,000	40,000	1,000	200
	192	6	geminidb.redis.2xlarge.4	8	480,000	60,000	1,000	200
	256	8	geminidb.redis.2xlarge.4	8	640,000	80,000	1,000	200
	384	10	geminidb.redis.2xlarge.4	8	800,000	100,000	1,000	200
	512	6	geminidb.redis.4xlarge.4	16	960,000	60,000	1,000	200
	768	9	geminidb.redis.4xlarge.4	16	1,440,000	90,000	1,000	200
	1024	12	geminidb.redis.4xlarge.4	16	1,920,000	120,000	1,000	200

Instance Type	Storage (GB)	Nodes	Node Flavor	vCPUs	QPS	Max. Connections	Databases	Accounts
	2048	22	geminidb.redis.4xlarge.4	16	3,520,000	220,000	1,000	200
	4096	24	geminidb.redis.8xlarge.4	32	7,680,000	240,000	1,000	200
	8192	36	geminidb.redis.8xlarge.4	32	11,520,000	360,000	1,000	200

Table 1-4 Primary/standby GeminiDB Redis instance specifications(standard configuration)

Instance Type	Storage (GB)	Shards	Node Flavor	QPS	Max. Connections	Databases	Accounts
Primary/standby	4	1	geminidb.redis.medium.2	8,000	10,000	1,000	200
	8	1	geminidb.redis.medium.2	8,000	10,000	1,000	200
	16	1	geminidb.redis.medium.4	10,000	10,000	1,000	200
	24	1	geminidb.redis.large.4	20,000	10,000	1,000	200
	32	1	geminidb.redis.large.4	20,000	1,0000	1,000	200
	48	1	geminidb.redis.xlarge.4	40,000	2,0000	1,000	200
	64	1	geminidb.redis.xlarge.4	40,000	2,0000	1,000	200

Instance Type	Storage (GB)	Shards	Node Flavor	QPS	Max. Connections	Databases	Accounts
	96	1	geminidb.redis.2xlarge.4	80,000	20,000	1,000	200
	128	1	geminidb.redis.4xlarge.4	160,000	20,000	1,000	200

Table 1-5 GeminiDB Redis node specifications

Node Flavor	vCPUs	Memory (GB)	Max. Persistent Storage per Node (GB)	Maximum Connections per Node	Assured Bandwidth (Mbit/s)	Databases
geminidb.redis.medium.2	1	2	4	10,000	800	256
geminidb.redis.large.2	2	4	8	10,000	1200	256
geminidb.redis.xlarge.2	4	8	16	10,000	2500	1,000
geminidb.redis.2xlarge.2	8	16	32	10,000	5000	1,000
geminidb.redis.4xlarge.2	16	32	64	10,000	9000	1,000
geminidb.redis.8xlarge.2	32	64	128	10,000	18000	1,000
geminidb.redis.medium.4	1	4	8	10,000	800	256
geminidb.redis.large.4	2	8	16	10,000	1200	256
geminidb.redis.xlarge.4	4	16	32	10,000	2500	1,000
geminidb.redis.2xlarge.4	8	32	64	10,000	5000	1,000

Node Flavor	vCPUs	Memory (GB)	Max. Persistent Storage per Node (GB)	Maximum Connections per Node	Assured Bandwidth (Mbit/s)	Databases
geminidb.redis.4xlarge.4	16	64	128	10,000	9000	1,000
geminidb.redis.8xlarge.4	32	128	256	10,000	18000	1,000
geminidb.redis.medium.8	1	8	16	10,000	800	256
geminidb.redis.large.8	2	16	32	10,000	1200	256
geminidb.redis.xlarge.8	4	32	64	10,000	2500	1,000
geminidb.redis.2xlarge.8	8	64	128	10,000	5000	1,000
geminidb.redis.4xlarge.8	16	128	256	10,000	9000	1,000
geminidb.redis.8xlarge.8	32	256	512	10,000	18000	1,000

The bandwidth of a single node of some existing instances is 768 Mbit/s, which is the same as that of other Redis Cloud services. If existing instances are subjected to heavy traffic, you can choose **Service Tickets > Create Service Ticket** in the upper right corner of the console to consult on the bandwidth usage.

1.7 Instance Statuses

The status of a DB instance indicates the health of the instance. You can view the DB instance statuses on the management console.

Table 1-6 DB instance statuses

Status	Description
Available	The instance is available.
Abnormal	The instance is abnormal.
Creating	The instance is being created.

Status	Description
Creation failed	The instance failed to be created.
Restarting	The instance is being restarted.
Resetting password	The administrator password is being reset.
Adding node	Nodes are being added to an instance.
Deleting node	Nodes are being deleted from an instance.
Scaling up	The storage space of an instance is being scaled up.
Changing instance class	The vCPUs and memory of an instance are being changed.
Changing to yearly/monthly	The billing mode is being changed from pay-per-use to yearly/monthly.
Changing to pay-per-use	The billing mode is being changed from yearly/monthly to pay-per-use.
Uploading backup	The backup file is being uploaded.
Backing up	A database backup is being created.
Checking restoration	The backup of the instance is being restored to a new instance.
Configuring SSL	SSL is being enabled or disabled.
Frozen	The instance is frozen because your balance drops to or below zero.
Unfreezing	The instance is being unfrozen after the overdue payments are cleared.
Checking changes	The yearly/monthly instance is pending check when its billing mode is changed.

1.8 Constraints

The following tables list the constraints designed to ensure stability and security of GeminiDB Redis instances.

Specifications

Table 1-7 Specifications

Resource Type	Specifications	Description
CPU and memory	GeminiDB Redis API supports proxy cluster, Redis Cluster, and primary/standby instances.	• For details about specifications of different instance types, see **1.6 Instance Specifications**. • You can change the specifications to meet your service requirements by following **4.6.4 Changing the CPU and Memory Specifications of an Instance**.
Storage space	The storage space depends on the selected **instance specifications**.	Storage capacity can be scaled up or down. For details, see **4.6.7.1 Scaling Disk Space**.
Connections	The maximum value is the number of instance nodes multiplied by 10,000.	The maximum number of connections varies depending on the memory. For details, see **1.6 Instance Specifications**.

Quotas

Table 1-8 Quotas

Resource Type	Constraint	Description
Tag	A maximum of 20 tags can be added for each instance.	For more information, see **4.14 Tag Management**.
Free backup space	GeminiDB Redis API provides free backup storage for backup data.	For more information, see **Backup Storage**.
Retention period	The default value is 7 days. The value ranges from 1 to 3660 days.	For more information, see **Configuring an Automated Backup Policy**.

Naming Rules

Table 1-9 Naming rules

Item	Description
Instance name	• Contains 4 to 64 characters. • Must start with a letter. Only letters (case-sensitive), digits, hyphens (-), and underscores (_) are allowed.
Backup name	• Contains 4 to 64 characters. • Must start with a letter. Only letters (case sensitive), digits, hyphens (-), and underscores (_) are allowed.
Parameter template name	• Contains 1 to 64 characters. • Only letters (case sensitive), digits, hyphens (-), underscores (_), and periods (.) are allowed.

Security

Table 1-10 Security

Item	Description
Password of database administrator **rwuser**	• Contains 8 to 32 characters. • Can contain at least two types of the following characters: uppercase letters, lowercase letters, digits, and special characters ~!@#%^*-_=+? For more information, see **4.6.3 Changing the Administrator Password of a GeminiDB Redis Database**. • Keep your password secure. The system cannot retrieve it if it is lost.
Database port	Database port number. You can specify a port number, which ranges from 1024 to 65535 except 2180, 2887, 3887, 6377, 6378, 6380, 8018, 8079, 8091, 8479, 8484, 8999, 12017, 12333, and 50069. If you do not specify a port number, port 6379 is used by default.
VPC	After a GeminiDB Redis instance is created, the VPC where the instance is deployed cannot be changed.

Item	Description
Security group	A security group controls access between GeminiDB Redis API and other services. Ensure that the security group you selected allows your client to access the instance. If no security group is available, the system creates one for you.
Access control	A load balancer address does not support security groups. After an instance is created, configure IP address access control. If no whitelist is configured, all IP addresses that can communicate with the VPC can access the instance.
ACL account	GeminiDB Redis API provides enterprise-grade multi-tenancy. You can add read-only or read/write accounts for your instance to control access to each database to avoid misoperations from other tenants. A maximum of 200 ACL accounts can be created for each instance. For more information, see **4.10.2 ACL Account Management**.

Instance Operations

Table 1-11 Instance operations

Function	Constraint
Database access	• If remote access is not enabled, GeminiDB Redis instances and their associated ECSs must be in the same VPC subnet. • The security group must allow access from the associated ECS. By default, a GeminiDB Redis instance cannot be accessed through an ECS in a different security group. You need to add an inbound rule to the security group. • The default port number of a GeminiDB Redis instance is 6379. • The database port can be set when an instance is created and can be changed after the instance is created.
Instance deployment	The servers where instances are deployed are not directly visible to you. You can only access the instances through IP addresses and database ports.

Function	Constraint
Restarting a GeminiDB Redis instance	• GeminiDB Redis instances cannot be rebooted through commands. They must be rebooted on the console. • Restarting an instance will interrupt services, so off-peak hours are the best time. Ensure that your application can be reconnected.
Viewing GeminiDB Redis instance backups	GeminiDB Redis instance backups are stored in OBS buckets and are invisible to you.
Changing the CPU or memory of a GeminiDB Redis instance	• Second-level intermittent disconnection occurs once when the specifications are changed on a single node. Therefore, the entire instance is intermittently disconnected several times. Ensure that the client can be reconnected. You are advised to change the specifications during off-peak hours. • For a node whose specifications are being changed, its computing tasks are handed over to other nodes. Change specifications of nodes during off-peak hours to prevent instance overload.
Primary/Standby switchover	Only primary/standby GeminiDB Redis instances are supported. During a primary/standby switchover, the instances are disconnected for less than 10 seconds, which can cause slow latency or command execution failures. Ensure commands can be retried or the client can be reconnected. You are advised to perform the switchover during off-peak hours.
Data restoration	To prevent data loss, you are advised to back up key data before data restoration.
Storage space	If the storage space of an instance is full, data cannot be written to databases. You are advised to periodically check the storage space. GeminiDB Redis instance storage can be automatically scaled up in case of a sudden surge in data volumes. Enable autoscaling by following **4.6.7.3 Automatically Scaling Up Disk Space**.

Function	Constraint
Recycle bin	• You can move unsubscribed yearly/monthly instances and deleted pay-per-use instances to the recycle bin. You can restore an instance that was deleted up to 7 days ago from the recycle bin. • The recycling policy is enabled by default and cannot be disabled. Instances in the recycle bin are retained for 7 days by default, and this will not incur any charges. • Currently, you can put a maximum of 100 instances into the recycle bin. • If you delete an instance in the storage-full status, it will not be moved to the recycle bin.

For details about other development and O&M specifications that can effectively evaluate and improve service system stability, see **5.1 Development and O&M Rules**.

2 Billing

2.1 Billing Overview

In this document, you will learn about how instances are billed, how you can renew subscriptions and manage costs, and what happens if your account goes into arrears.

- **Billing Modes**

  There are yearly/monthly and pay-per-use billing modes. Each one has different advantages and disadvantages.

  - Yearly/Monthly: You pay upfront for the amount of time you expect to use the service for. You will need to make sure you have a top-up account with a sufficient balance or have a valid payment method configured first.
  - Pay-per-use: You can start using the GeminiDB instance first and then pay as you go.

  For details about the two billing modes, see **2.2.1 Overview**.

You can also change the billing mode later if it no longer meets your needs. For details, see **2.5.1 Overview**.

- **Billing Items**

  You will be billed for instance specifications, storage space, backup space, and EIP bandwidths. For details about the billing factors and formulas for each billed item, see **2.3 Billing Items**.

  For more information about billing samples and the billing for each item, see **2.4 Billing Examples**.

- **Renewing Subscriptions**

  If you want to continue using an instance after it expires, you need to renew the instance subscription within the specified period. Otherwise, resources, such as compute and storage, will be automatically released, and data may be lost.

  You can renew your subscription manually or automatically. For details, see **2.6.1 Overview**.

- **Viewing Bills**

  You can choose **Billing & Costs** > **Bills** to check the instance transactions and bills. For details, see **2.7 Bills**.

- **Arrears**

  If there is not a sufficient account balance to pay for your bill and there is no other payment method configured, your account will go into arrears. If you want to continue using your cloud services, you will need to top up your account in a timely manner. For details, see **2.8 Arrears**.

- **Stopping Billing**

  If you no longer need to use your GeminiDB Redis instance, you can unsubscribe from or delete it to stop the billing. For details, see **2.9 Billing Termination**.

- **Managing Costs**

  GeminiDB Redis costs include resource costs and O&M costs. You can allocate, analyze, and optimize GeminiDB costs to save more money. For details, see **2.10 Cost Management**.

2.2 Billing Modes

2.2.1 Overview

There are yearly/monthly and pay-per-use billing modes. Each one has different advantages and disadvantages.

- Yearly/Monthly is a prepaid billing mode. You pay in advance for a subscription term, and in exchange, you get a discounted rate. The longer the subscription term, the bigger the discount. Yearly/Monthly billing is a good option for long-term, stable services.
- Pay-per-use is a postpaid billing mode. You pay as you go and just pay for what you use. The instance usage is calculated by the second but billed every hour. Pay-per-use billing is a good option for scenarios where there are sudden traffic bursts, such as e-commerce promotions.

Table 2-1 lists differences between the two billing modes.

Table 2-1 Differences between billing modes

Billing Mode	Yearly/Monthly	Pay-per-use
Payment	Prepaid Billed by the subscription term you purchase	Postpaid Billed for what you use
Billing Method	Billed by the subscription term you purchase	Calculated by the second but billed every hour
Billing Items	Instance specifications (vCPUs and memory), storage space, backup space, and EIPs	Instance specifications (vCPUs and memory), storage space, backup space, and EIPs
Changing the Billing Mode	Yearly/Monthly can be changed to pay-per-use. The change takes effect only after the yearly/monthly subscription expires. For details, see **2.5.3 Yearly/Monthly to Pay-per-Use**.	Pay-per-use can be changed to yearly/monthly. For details, see **2.5.2 Pay-per-Use to Yearly/Monthly**.
Changing the Specifications	Supported	Supported
Application Scenarios	Recommended for resources expected to be in use long term. A cost-effective option for scenarios where the resource usage duration is predictable.	Recommended when the resource demands are likely to fluctuate and you want more flexibility.

2.2.2 Yearly/Monthly Billing

If you expect to use resources for a longer period, you can save money by selecting yearly/monthly billing. This section describes billing rules for yearly/monthly GeminiDB Redis resources.

Application Scenarios

If you want to ensure resource stability over a certain period of time, yearly/monthly billing is a good choice for the following types of workloads:

- Long-term workloads with stable resource requirements, such as official websites, online malls, and blogs.

- Long-term projects, such as scientific research projects and large-scale events.
- Workloads with predictable traffic bursts, for example, e-commerce promotions or festivals.
- Workloads with high data security requirements.

Billed Items

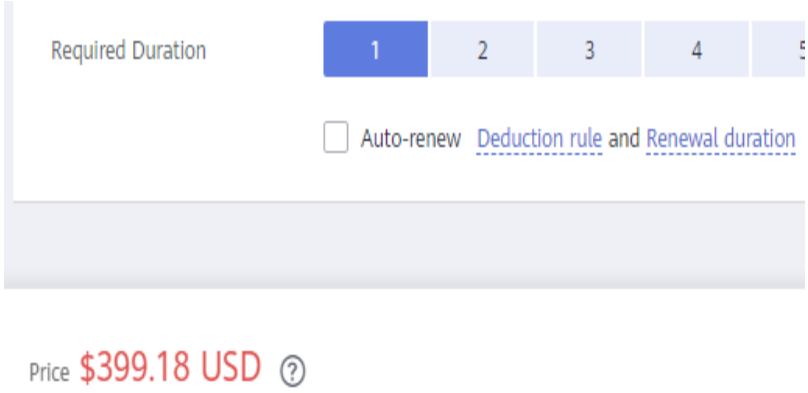
You are billed for the following items on a yearly/monthly basis.

Table 2-2 Items billed on a yearly/monthly basis

Billed Item	Description
Instance specifications	Instance specifications, including vCPUs and memory.
Storage space	If the actual storage usage exceeds your purchased storage, you will be billed for additional storage on a pay-per-use basis.
Backup space	GeminiDB Redis provides backup storage up to 100% of the database storage you purchase at no additional charge. After the free backup space is used up, charges are applied based on the backup space pricing details. Pricing is listed on a per-hour basis, but bills are calculated based on the actual usage duration.
(Optional) Public network bandwidth	GeminiDB Redis instances are accessible from public networks, and you are billed for the generated public network traffic, but not for private network traffic.

If you want to purchase a 3-node (specifications of each node: 2 vCPUs) GeminiDB Redis instance with 12 GB of storage space. At the bottom of the instance buying page, price details (excluding the backup space fee) will be displayed.

Figure 2-1 Example price



The price includes:

- Selected specifications for your instance
- Storage space

NOTE

The backup space fee is not included. For details about the backup price, see **Product Pricing Details**.

Backup Storage Space

DB Instance Type	Hourly	Currency
Cluster	0.00004	Price per GB

Billed Usage Period

A yearly/monthly GeminiDB Redis instance is billed for the purchased duration (UTC+8). The billing starts when you activated or renewed the subscription, and ends at 23:59:59 of the expiry date.

For example, if you purchased a one-month GeminiDB Redis instance on March 08, 2023, 15:50:04, the billed usage period is from March 08, 2023, 15:50:04 to April 08, 2023, 23:59:59.

Billing Examples

Suppose you purchased a one-month GeminiDB Redis instance (instance specifications: 2 vCPUs, Dedicated Edition; nodes: 3; storage: 40 GB; backup space: 50 GB (40 GB for free)) on March 08, 2023, 15:50:04, and renewed the subscription for one more month before the initial subscription expired. That would include two usage periods:

- March 08, 2023, 15:50:04 to April 08, 2023, 23:59:59
- April 08, 2023, 23:59:59 to May 08, 2023, 23:59:59
  - From April 08, 2023, 23:59:59 to May 01, 2023, 23:59:59, 20 GB of free backup space was used.
  - From May 01, 2023, 23:59:59 to May 08, 2023, 23:59:59, another 10 GB of backup space was used, which was billed for 168 hours.

You will be billed for both usage periods. GeminiDB Redis resources are billed individually as follows:

Table 2-3 Formulas for billing yearly/monthly resources

Resource	Formula	Unit Price
Instance specifications (including vCPUs and memory)	Unit price of the instance specifications x Required duration x Number of nodes	For details about the unit price, see **Cluster CPU/Memory** on **Product Pricing Details**.

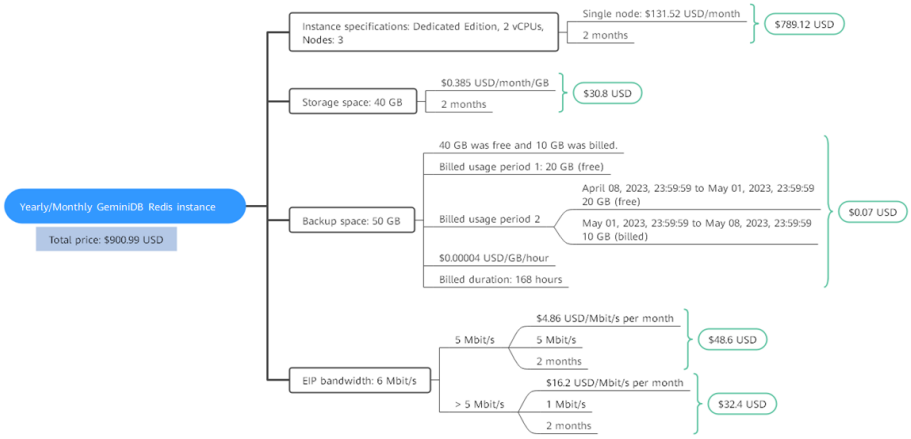
Resource	Formula	Unit Price
Storage space	Storage space unit price x Required duration x Storage space (GB)	For details about the unit price, see **Storage Space** on **Product Pricing Details**.
Backup space	Backup space unit price x Required duration x (Backup space – Storage space) (GB) **NOTE** The billed duration refers to the length of time the billed backup space was used for.	For details about the unit price, see **Backup Storage Space** on **Product Pricing Details**.
Public network bandwidth	Billed by fixed bandwidth	For details, see **Product Pricing Details**.

Figure 2-2 shows how the total price is calculated.

NOTICE

Prices in the figure are just examples. Actual prices are subject to **Product Pricing Details**.

Figure 2-2 Total price for a yearly/monthly GeminiDB Redis instance



Price Change After Specification Change

If the specifications of a yearly/monthly GeminiDB Redis instance no longer meet your needs, you can change the specifications on the console. The system will recalculate the price and either bill or refund you the difference.

- If you upgrade your GeminiDB Redis instance specifications, you need to pay the difference in price.

- If you downgrade your GeminiDB Redis instance specifications, Huawei Cloud will refund you the difference.

You are not advised to downgrade your GeminiDB Redis instance to a lower specification because the instance performance may be affected. Suppose you purchased a yearly/monthly instance (1 vCPU | 6 GB and 3 nodes) on April 08, 2023 and upgraded the instance specifications to 2 vCPUs | 12 GB and 3 nodes on April 18, 2023. The price for the original specifications was $199.59 USD/month, and that for the new specifications was $399.18 USD/month. The price difference will be calculated as follows:

Price difference for the specification upgrade = Price for the new specifications × Remaining period - Price for the original specifications × Remaining period

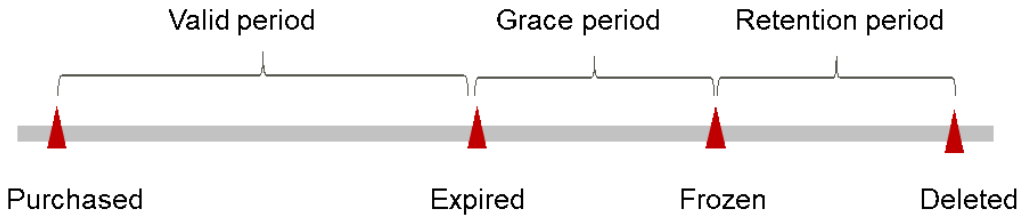
The remaining period in the formula is the remaining days of each calendar month divided by the maximum number of days in each calendar month. In this example, Remaining period = 12 (Remaining days in April)/30 (Maximum number of days in April) + 8 (Remaining days in May)/31 (Maximum number of days in May) = 0.6581. Cost of upgrade = $399.18 USD × 0.6581 – $199.59 USD × 0.6581 = $131.35 USD.

For more details, see **Pricing of a Changed Specification**.

Impact of Expiration

Figure 2-3 shows the statuses a yearly/monthly GeminiDB Redis instance can go through throughout its lifecycle. After a GeminiDB Redis instance is purchased, it enters the valid period and runs normally during this period. If the instance is not renewed after it expires, before being deleted, it first enters a grace period and then a retention period.

Figure 2-3 Lifecycle of a yearly/monthly GeminiDB Redis instance



Expiration Reminder

The system will send you a reminder (by email, SMS, or in-app message) 7 days before a yearly/monthly GeminiDB Redis instance expires to remind you to renew the subscription.

Impact of Expiration

If your yearly/monthly GeminiDB Redis instance is not renewed after it expires, it changes to the **Expired** state and enters a grace period. During the grace period, you can access the GeminiDB Redis instance but cannot:

- Change instance specifications.

- Change the billing mode from yearly/monthly to pay-per-use.
- Unsubscribe from it.

If the yearly/monthly GeminiDB Redis instance is not renewed after the grace period ends, its status turns to **Frozen** and it enters a retention period. You cannot perform any operations on the GeminiDB Redis instance while it is in the retention period.

If the yearly/monthly GeminiDB Redis instance is not renewed by the time the retention period ends, it will be released and data cannot be restored.

NOTE

- For details about renewals, see **2.6.1 Overview**.

2.2.3 Pay-per-Use Billing

Pay-per-use billing means you pay nothing up front and are not tied into any contract or commitment. This section describes billing rules for pay-per-use GeminiDB Redis instances.

Application Scenarios

Pay-per-use billing is good for short-term, bursty, or unpredictable workloads that cannot tolerate any interruptions, such as applications for e-commerce flash sales, temporary testing, and scientific computing.

Billing Items

You are billed for the following items on a pay-per-use basis.

Table 2-4 Items billed on a pay-per-use basis

Billing Item	Description
Instance specifications	Instance specifications, including vCPUs and memory.
Storage space	Instance storage space, which is billed hourly on a pay-per-use basis.
Backup space	GeminiDB Redis provides backup storage up to 100% of the database storage you purchase at no additional charge. After the free backup space is used up, charges are applied based on the backup space pricing details. Pricing is listed on a per-hour basis, but bills are calculated based on the actual usage duration.
(Optional) Public network bandwidth	GeminiDB Redis instances are accessible from public networks, and you are billed for the generated public network traffic, but not for private network traffic.

If you want to purchase a pay-per-use 3-node (specifications of each node: 2 vCPUs) GeminiDB Redis instance with 12 GB of storage space. At the bottom of the instance buying page, price details (excluding the backup space fee) will be displayed.

Figure 2-4 Example price

The price includes:

- Instance specifications (including vCPUs and memory)
- Selected storage space

NOTE

The backup space fee is not included. For details about the backup price, see **Product Pricing Details**.

Backup Storage Space

DB Instance Type	Hourly	Currency
Cluster	0.00004	Price per GB

Billed Usage Period

Pay-per-use GeminiDB Redis instance usage is calculated by the second and billed every hour. The billing starts when ECS instance is created and ends when the instance is deleted.

NOTE

It takes a certain time to create an instance. The billing starts from the time when the instance is successfully created. You can view the two time points on the **Basic Information** page. You can view the time when the instance is created beside the **Created** field.

For example, if you purchased a pay-per-use GeminiDB Redis instance at 8:45:30 and deleted it at 8:55:30, you are billed for the 600 seconds from 8:45:30 to 8:55:30.

Billing Examples

Suppose you purchased a pay-per-use instance on April 18, 2023, 9:59:30, and deleted it on April 18, 2023, 10:45:46. Two usage periods will be billed:

- Usage of 30 seconds from 9:59:30 to 10:00:00
- Usage of 2,746 seconds from 10:00:00 to 10:45:46
  - The free backup space is used from 10:00:00 to 10:45:00.
  - Ten GB of billing backup space is used from 10:45:00 to 10:45:46 and the billed duration is 46 seconds.

The price displayed in the pricing details is per hour, so you need to divide it by 3,600 to obtain the price for each second and then multiply the per-second price by the total number of seconds. GeminiDB Redis instances are billed individually as follows.

Table 2-5 Formulas for billing pay-per-use resources

Resource	Formula	Unit Price
Compute resources (including vCPUs and nodes)	Unit price of instance specifications x Required duration	For details about the unit price, see **Cluster CPU/Memory** on **Product Pricing Details**
Storage space	Storage space unit price x Purchase duration	For details about the unit price, see **Storage Space** on **Product Pricing Details**.
Backup space	Backup space unit price x Required duration x (Backup space – Storage space) (GB) **NOTE** The billed duration refers to the length of time the billed backup space was used for.	For details about the unit price, see **Backup Storage Space** on **Product Pricing Details**.
Public network traffic	Tiered billing by fixed bandwidth • 0 Mbit/s to 5 Mbit/s (included): billed at a fixed unit price per Mbit/s • Greater than 5 Mbit/s: billed at a different price per Mbit/s	For details, see **Bandwidth Price** on **Product Pricing Details** page or **Product Pricing Details**.

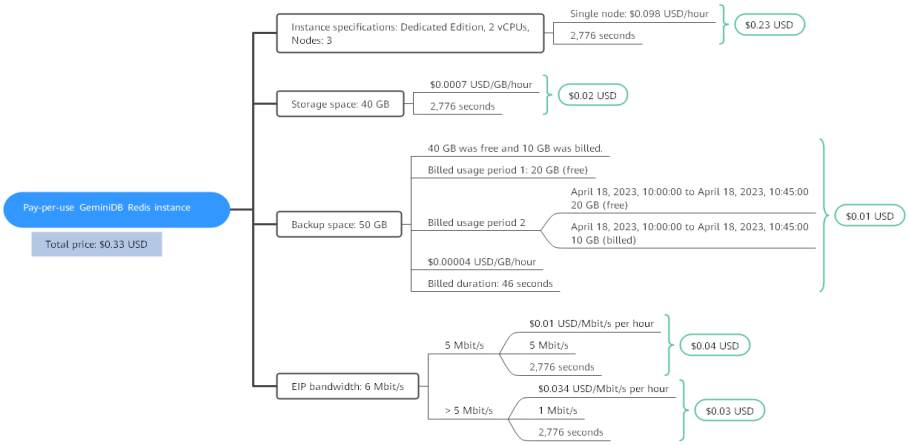
Figure 2-5 shows how the total price is calculated.

NOTICE

Prices in the figure are just examples. Actual prices are subject to **Product Pricing Details**.

For pay-per-use billing, decimal numerals on the price calculator are rounded off and are accurate to two decimal places. If the fee is less than $0.01 USD (after rounding off), $0.01 USD will be displayed.

Figure 2-5 Total price for a pay-per-use GeminiDB Redis instance



Price Change After Specification Change

If you change the specifications of a pay-per-use GeminiDB Redis instance, the original order will become invalid and a new order will be placed. You will be billed based on the new specifications.

If you change instance specifications within a given hour, multiple records will be generated. Different records record the billing for different specifications.

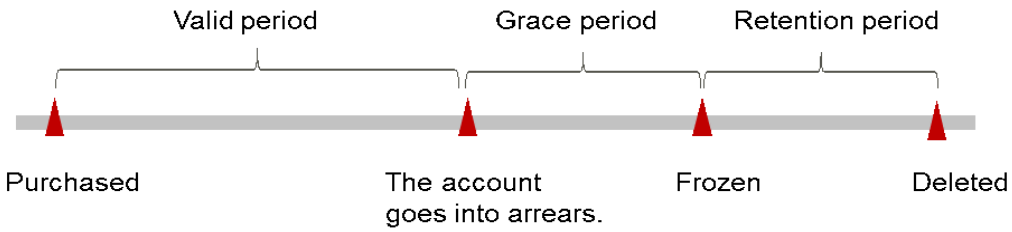
For example, if you purchased a pay-per-use instance (0.5 vCPUs | 3 GB) at 9:00:00 and changed the instance specifications to 1 vCPU | 6 GB at 9:30:00, the following items will be billed:

- Specifications 0.5 vCPUs | 3 GB usage from 9:00:00 to 9:30:00
- Specifications 1 vCPU | 6 GB usage from 9:30:00 to 10:00:00

Impacts of Arrears

Figure 2-6 shows the statuses a pay-per-use GeminiDB Redis instance can go through throughout its lifecycle. After a GeminiDB Redis instance is purchased, it enters the valid period and runs normally during this period. If your account goes into arrears, the instance enters a grace period and then a retention period.

Figure 2-6 Lifecycle of a pay-per-use GeminiDB Redis instance



Arrears Reminder

The system will bill you for pay-per-use resources after each billing cycle ends. If your account goes into arrears, we will notify you by email, SMS, or internal message.

Impacts of Arrears

If your configured payment method is unable to pay a bill for pay-per-use resources, the resources enter a grace period. You are still responsible for expenditures generated during the grace period. You can view the charges on the **Billing Center** > **Overview** page and pay any past due balance as needed.

If you do not bring your account balance current before the grace period expires, the GeminiDB Redis instance status turns to **Frozen** and it enters a retention period. You cannot perform any operations on a pay-per-use GeminiDB Redis instance in the **Frozen** status.

If you do not bring your account balance current before the retention period ends, your instance will be released, and data cannot be restored.

NOTE

- During the retention period, you cannot access or use your instance but the data stored in it can be retained. The retention period for Huawei Cloud International website is 15 days.
- During the grace period, you can access and use only some resources of your instance. The grace period for Huawei Cloud International website is 15 days.
- For details about top-up, see **Topping Up an Account**.

2.3 Billing Items

Billing

You will be billed for instance specifications, storage space, backup space, and public network traffic. For details, see **Table 2-6**.

NOTE

The billed items marked with asterisks (*) are mandatory.

Table 2-6 Billing items of a GeminiDB Redis instance

Billing Item	Description	Billing Mode	Formula
* Specifications	Billed by instance specifications, including vCPUs and memory. Computing and storage capabilities vary by the number of vCPUs and memory size.	Yearly/ Monthly and pay-per-use	**Unit price x Required duration** For details about the unit price, see **Cluster CPU/Memory** on **Product Pricing Details**.

Billing Item	Description	Billing Mode	Formula
* Storage space	Billed based on unified standards.	Yearly/ Monthly and pay-per-use	**Unit price x Storage space x Required duration** For details about the unit price, see **Storage Space** on **Product Pricing Details**.
Backup space	Billed based on unified standards.	Pay-per-use	**Unit price x Billed backup space x Required duration** For details about the unit price, see **Backup Storage Space** on **Product Pricing Details**. **NOTE** The billed duration refers to the length of time the billed backup space was used for.
Public network traffic	An EIP is required if a GeminiDB Redis instance needs to access the Internet. Billed by bandwidth, traffic, and the EIP reservation price. • EIP for a yearly/monthly GeminiDB Redis instance: billed by bandwidth. • EIP for a pay-per-use GeminiDB Redis instance: billed by bandwidth, traffic, or shared bandwidth. You are also charged for IP reservation if you do not bind the EIP to any instance.	Yearly/ Monthly and pay-per-use. You can purchase a bandwidth add-on package or a shared traffic package.	**Tiered pricing based on fixed bandwidth**. • 0 Mbit/s to 5 Mbit/s (included): billed at a fixed unit price per Mbit/s. • Greater than 5 Mbit/s: billed at a different price per Mbit/s. For details about the unit price, see **Bandwidth Price** on **Product Pricing Details** or **Product Pricing Details**.

Billing Examples

Suppose you purchased a one-month GeminiDB Redis instance (instance specifications: 2 vCPUs, Dedicated Edition; nodes: 3; storage: 40 GB; backup space: 50 GB (40 GB for free)) on March 08, 2023, 15:50:04, and renewed the subscription for one more month before the initial subscription expired. That would include two usage periods:

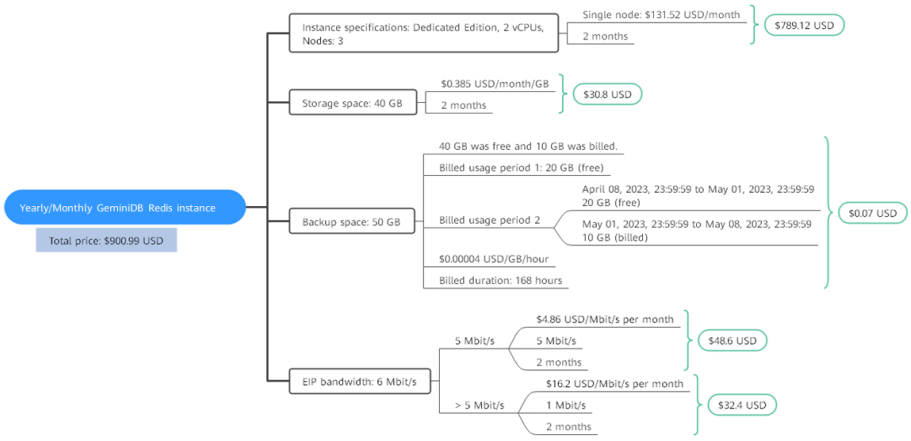
- March 08, 2023, 15:50:04 to April 08, 2023, 23:59:59
- April 08, 2023, 23:59:59 to May 08, 2023, 23:59:59
  - From April 08, 2023, 23:59:59 to May 01, 2023, 23:59:59, 20 GB of free backup space was used.
  - From May 01, 2023, 23:59:59 to May 08, 2023, 23:59:59, another 10 GB of backup space was used, which was billed for 168 hours.

Figure 2-7 shows how the total price is calculated.

NOTICE

Prices in the figure are only for reference. For details, see **Product Pricing Details**.

Figure 2-7 Total price for a yearly/monthly GeminiDB Redis instance



For more billing examples of a pay-per-use GeminiDB Redis instance, see **Billing Examples**.

2.4 Billing Examples

Billing Scenario

A user purchased a pay-per-use GeminiDB Redis instance at 15:30:00 on March 18, 2023. The instance configuration is as follows:

- Specifications: 2 vCPUs | 12 GB
- Nodes: 3
- Public network bandwidth: 6 Mbit/s

After a period of time, the user found that the current GeminiDB Redis instance specifications no longer met service requirements and updated the specifications to 4 vCPUs | 24 GB at 09:00:00 on March 20, 2023. Since the user wanted to use the instance long term, the user then changed the instance to yearly/monthly billing with a one-month duration at 10:30:00 on the same day. So how much will the user be billed for this GeminiDB Redis instance in March and April?

Billing Analysis

The total price of this GeminiDB Redis instance involves both pay-per-use and yearly/monthly usage:

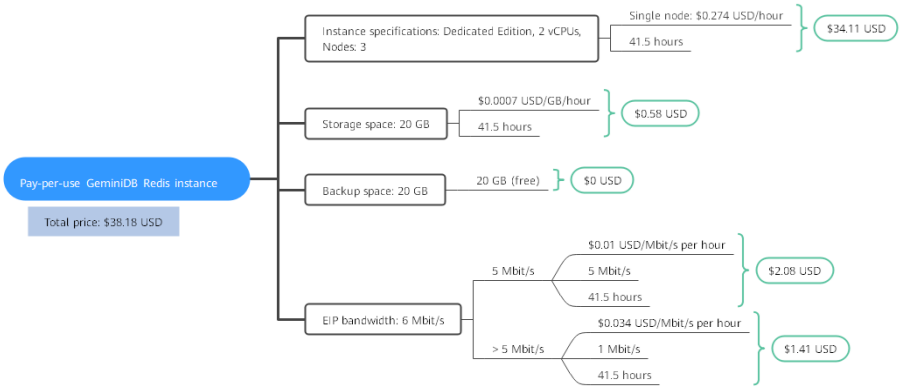
- Pay-per-use usage: March 18, 2023, 15:30:00 to March 20, 2023, 10:30:00
  - March 18, 2023, 15:30:00 to March 20, 2023, 9:00:00
    - Instance specifications: 2 vCPUs | 20 GB
    - Nodes: 3
    - Used storage space: 20 GB
    - Used backup space: 20 GB
    - Public network bandwidth: 6 Mbit/s
  - March 20, 2023, 9:00:00 to March 20, 2023, 10:30:00
    - Instance specifications: 4 vCPUs | 40 GB
    - Nodes: 3
    - Used storage space: 40 GB
    - Used backup space: 50 GB (billed on a pay-per-use basis from March 20, 2023, 10:00:00 to March 20, 2023, 10:30:00)
    - Public network bandwidth: 6 Mbit/s
- Yearly/Monthly: March 20, 2023, 10:30:00 to April 20, 2023, 23:59:59
  - Instance specifications: 4 vCPUs | 80 GB
  - Nodes: 3
  - Used storage space: 80 GB
  - Used backup space: 100 GB (billed on a pay-per-use basis from April 10, 2023, 23:59:59 to April 20, 2023, 23:59:59)
  - Public network bandwidth: 6 Mbit/s
  - Billed duration: one month

NOTICE

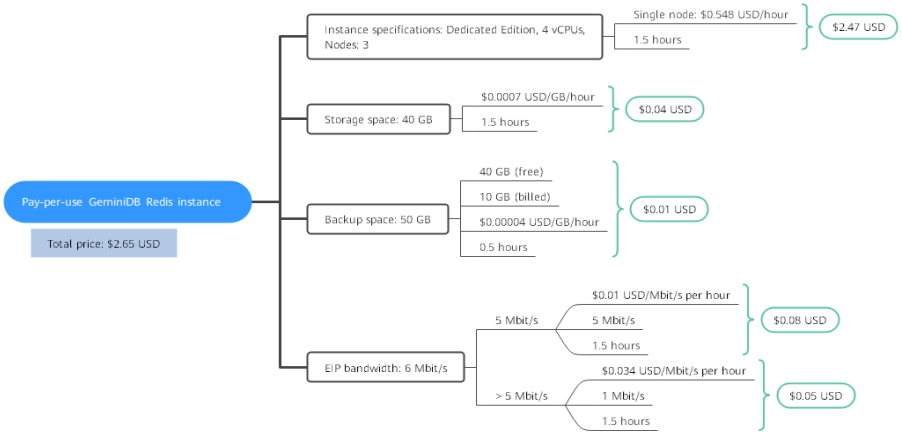
Unit prices in this example are used for reference only. The prices shown here are only estimates. As unit prices change from time to time, the prices shown here will differ from actual prices. For details, see the data released on the Huawei Cloud official website.

Pay-per-use

From March 18, 2023, 15:30:00 to March 20, 2023, 09:00:00, a GeminiDB Redis instance with specifications 2 vCPUs | 20 GB was used for 41.5 hours, so the price would be calculated as follows.

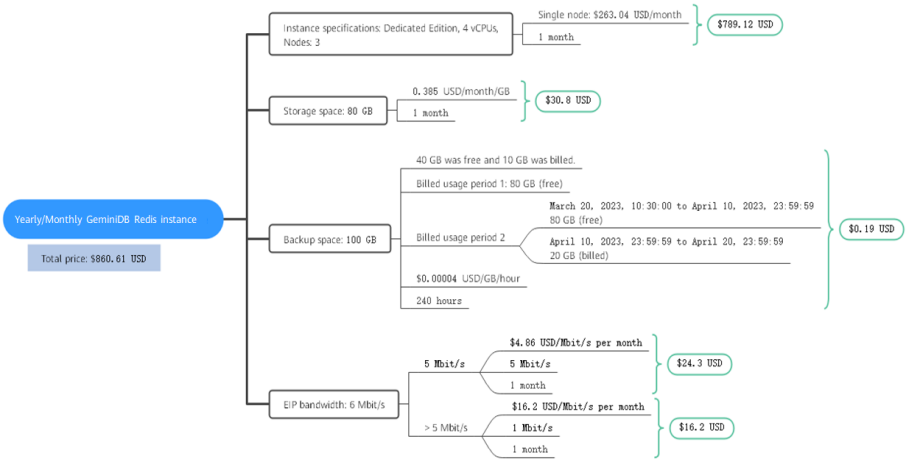


From March 20, 2023, 09:00:00 to March 20, 2023, 10:30:00, a GeminiDB Redis instance with specifications 4 vCPUs | 40 GB was used for 1.5 hours, so the price would be calculated as follows.



Yearly/Monthly

From March 20, 2023, 10:30:00 to April 20, 2023, 23:59:59, a GeminiDB Redis instance purchased using yearly/monthly billing was used for one month, so the price would be calculated as follows.



From March to April, the total price of this GeminiDB Redis instance is $901.44 USD (38.18 + 2.65 + 860.61).

2.5 Billing Mode Changes

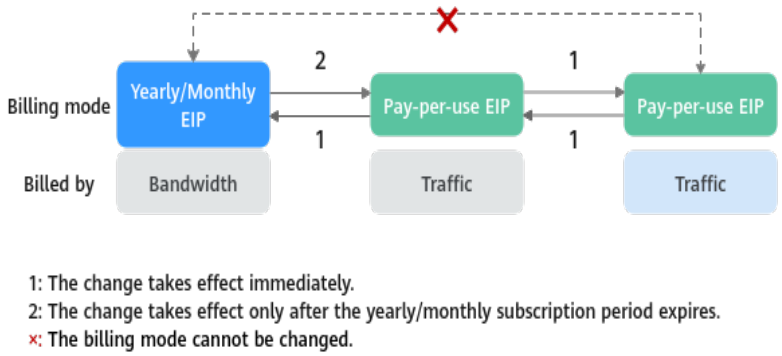
2.5.1 Overview

After purchasing a GeminiDB Redis instance, you can change the billing mode if it no longer meets your needs. **Table 2-7** lists changeable billing items of the GeminiDB Redis instance.

Table 2-7 Changeable billing items of GeminiDB Redis instances

Billing Item	Change Description	Reference
Instance specifications (vCPUs and nodes)	Changing the billing mode of a GeminiDB Redis instance includes the changes to compute resources (vCPUs and nodes). • Change from pay-per-use to yearly/monthly to enjoy lower prices. • Change from yearly/monthly to pay-per-use to use the GeminiDB Redis instance more flexibly. **NOTE** Such a change takes effect only after the yearly/monthly subscription ends.	• **2.5.2 Pay-per-Use to Yearly/Monthly** • **2.5.3 Yearly/Monthly to Pay-per-Use**
EIP	• A yearly/monthly EIP can be changed to a pay-per-use EIP billed by bandwidth after the yearly/monthly subscription ends. • A pay-per-use EIP billed by bandwidth can be changed to a yearly/monthly EIP. • Pay-per-use EIPs billed by bandwidth can be changed to pay-per-use EIPs billed by traffic, and pay-per-use EIPs billed by traffic can be changed to pay-per-use EIPs billed by bandwidth. For details, see **Figure 2-8**.	• **2.5.2 Pay-per-Use to Yearly/Monthly** • **2.5.3 Yearly/Monthly to Pay-per-Use**

Figure 2-8 EIP billing mode change



2.5.2 Pay-per-Use to Yearly/Monthly

If you have a pay-per-use GeminiDB Redis instance that you expect to use for a long time, you can change it to yearly/monthly billing to reduce costs. Doing so will create an order. After you pay for the order, yearly/monthly billing will be applied immediately.

Suppose you bought a pay-per-use GeminiDB Redis instance at 15:29:16 on April 18, 2023 and changed it to yearly/monthly billing at 16:30:30 on the same day. After you paid for the order, yearly/monthly billing was applied immediately. On the **Billing Center** > **Billing** page, three line items were generated.

- Pay-per-use expenditures for 15:29:16 to 16:00:00 on April 18, 2023
- Pay-per-use expenditures for 16:00:00 to 16:30:30 on April 18, 2023
- Yearly//monthly expenditure generated on April 18, 2023, 16:30:30

Constraints

Resources such as EIPs that are used by an instance may not support the change with this instance. For details about their billing mode change rules and handling methods, see **Table 2-8**.

Table 2-8 EIP billing mode change rules

Resource	Billing Mode	Billed By	Bandwidth Type	Change to Yearly/ Monthly Billing with GeminiDB Redis Instance	Handling Measure
EIP	Pay-per-use	Bandwidth	Dedicated	Supported	Change the EIP to yearly/ monthly billing on the EIP console. For details, see **Changing EIP Billing Mode**.
EIP	Pay-per-use	Traffic	Dedicated	Not supported	An EIP that is billed by traffic on a pay-per-use basis cannot be directly changed to be billed on a yearly/monthly basis. To change this: 1. Change the EIP to be billed by bandwidth on a pay-per-use basis. 2. Change the EIP to be billed on a yearly/ monthly basis. For details, see **Changing EIP Billing Mode**.

Prerequisites

- The billing mode of the instance is pay-per-use.
- The instance status is **Available**.

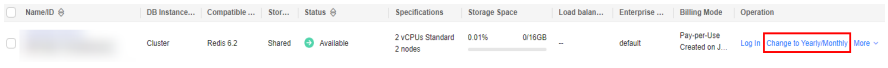
Procedure

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the target instance and click **Change to Yearly/ Monthly** in the **Operation** column.

Figure 2-9 Change to Yearly/Monthly



Alternatively, click the instance name to go to the **Basic Information** page. In the **Billing Information** area, click **Change to Yearly/Monthly** in the **Billing Mode** field.

Figure 2-10 Change to Yearly/Monthly

NOTE

The billing mode of multiple instances can be changed in batches. Perform the following steps:

1. Select the instances whose billing mode you want to change.
2. Click **Change to Yearly/Monthly** above the instance list.

Step 4 On the displayed page, specify a subscription duration in month. The minimum duration is one month.

If you do not need to modify your settings, click **Pay** to go to the payment page.

Step 5 Select a payment method and click **Confirm**.

Step 6 View the results on the **Instances** page.

In the upper right corner of the instance list, click to refresh the list. The instance status will become **Available** after the change is successful. The billing mode changes to **Yearly/Monthly**.

----End

2.5.3 Yearly/Monthly to Pay-per-Use

After creating a yearly/monthly GeminiDB Redis instance, you can change it to pay-per-use for more flexibility, and you can recoup part of what you paid for the subscription.

Suppose you bought a yearly/monthly GeminiDB Redis instance at 15:29:16 on April 18, 2023 and changed it to pay-per-use billing at 16:30:00 on the same day. On the **Billing Center** > **Billing** page, bills information is generated as follows:

- Yearly/Monthly expenditures for 15:29:16 on April 18 to 23:59:59 on May 18, 2023
- Pay-per-use expenditures for 23:59:59 on May 18, 2023 to the end time of pay-per-use billing. A bill was generated every hour.

NOTE

The pay-per-use billing mode will take effect only after the yearly/monthly subscription has expired. Auto-renewal will not be in effect.

Constraints

Resources such as EIPs that are used by an instance may not support the change with this instance. For details about their billing mode change rules and handling methods, see **Table 2-9**.

Table 2-9 EIP billing mode change rules

Resource	Billing Mode	Billed By	Bandwidth Type	Change to Pay-Per-Use Billing with GeminiDB Redis Instance	Handling Measure
EIP	Yearly/Monthly	Bandwidth	Dedicated	Not supported	Change the EIP to yearly/monthly billing on the EIP console. For details, see **Changing EIP Billing Mode**.
EIP	Yearly/Monthly	Traffic	Dedicated	Not supported	An EIP billed on a yearly/monthly basis cannot be directly changed to be billed by traffic on a pay-per-use basis. To change this: 1. Change the EIP to be billed by bandwidth on a pay-per-use basis. 2. Change the EIP to be billed by traffic on a pay-per-use basis. For details, see **Changing EIP Billing Mode**.

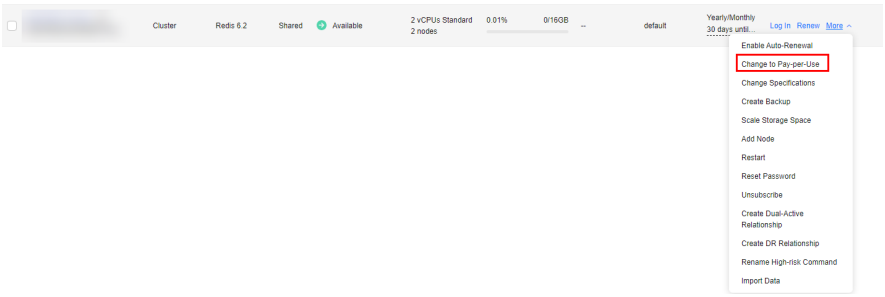
Procedure

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the instance whose billing mode you want to change and click **More** > **Change to Pay-per-Use** in the **Operation** column.

Figure 2-11 Change to Pay-per-Use



NOTE

The billing mode of multiple pay-per-use instances can be changed in batches. Perform the following steps:

1. Select the instances whose billing mode you want to change.
2. Click **More** > **Change to Pay-per-Use** in the **Operation** column

Step 4 On the displayed page, confirm the instance information and click **Change to Pay-per-Use**. The billing mode will change to pay-per-use after the instance expires. Auto renewal will be disabled after the billing mode of your instances change to pay-per-use. Exercise caution when performing this operation.

Step 5 After you submit the change, check whether a message is displayed in the **Billing Mode** column, indicating that the billing mode will be changed to pay-per-use after the subscription expires.

Step 6 To cancel the change, choose **Billing** > **Renewal** to enter the Billing Center. On the **Renewals** page, locate the instance and click **More** > **Cancel Change to Pay-per-Use**.

Step 7 In the displayed dialog box, click **Yes**.

----End

2.6 Renewing Subscriptions

2.6.1 Overview

When to Renew Subscriptions

If a yearly/monthly instance is about to expire but you want to continue using it, you need to renew the instance subscription within a specified period, or resources, such as vCPUs and memory, will be automatically released, and data will be lost and cannot be restored.

Only yearly/monthly instance subscriptions can be renewed. If you use pay-per-use instances, just ensure that your account has a valid payment method configured or a top-up account with a sufficient balance.

If you renew the instance before it expires, resources will be retained and you can continue using the instance. For details about statuses after instances have expired and the associated impacts, see **Impact of Expiration**.

How to Renew Subscriptions

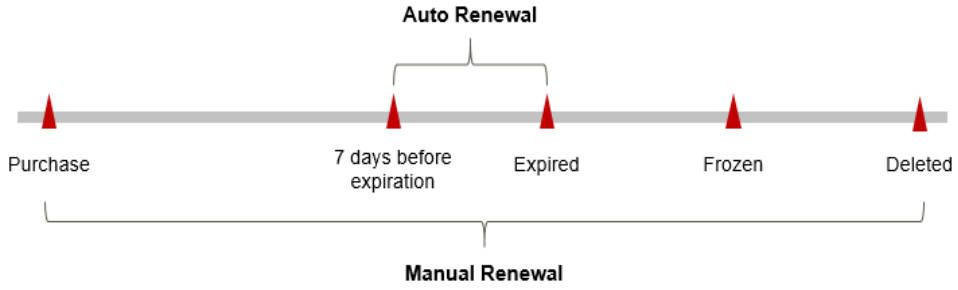
You can renew a yearly/monthly GeminiDB Redis instance manually or automatically.

Table 2-10 Renewing a yearly/monthly GeminiDB Redis instance

Method	Description
2.6.2 Manually Renewing an Instance	You can renew a yearly/monthly instance anytime on the console before it is automatically deleted.
2.6.3 Auto-renewing an Instance	You can enable auto-renewal to automatically renew the instance before it expires. This prevents resources from being deleted in case you forget to renew a subscription.

You can select a method to renew a yearly/monthly instance based on the phase the instance is currently in.

Figure 2-12 Selecting a renewal method based on the instance's current phase



- An instance is in the **Provisioned** state after it is provisioned.
- When an instance subscription expires, the status will change from **Provisioned** to **Expired**.
- If an expired instance is not renewed, it enters a grace period. If it is not renewed by the time the grace period expires, the instance will be frozen and enter a retention period.
- If you do not renew the subscription before the retention period expires, your resources will be automatically deleted.

NOTE

- During the retention period, you cannot access or use your instance but the data stored in it can be retained. The retention period for Huawei Cloud International website is 15 days.
- During the grace period, you can access and use only some resources of your instance. The grace period for Huawei Cloud International website is 15 days.

You can enable auto-renewal any time before an instance expires. By default, the system will make the first attempt to charge your account for the renewal at 03:00, seven days before the expiry date. If this attempt fails, it will make another

attempt at 03:00 every day until the subscription is renewed or expired. You can change the auto-payment date for renewal as required.

2.6.2 Manually Renewing an Instance

You can renew a yearly/monthly instance anytime on the console before it is automatically deleted.

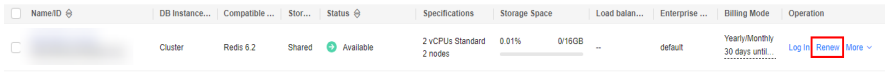
Renewing an Instance on the Console

Step 1 Log in to the management console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

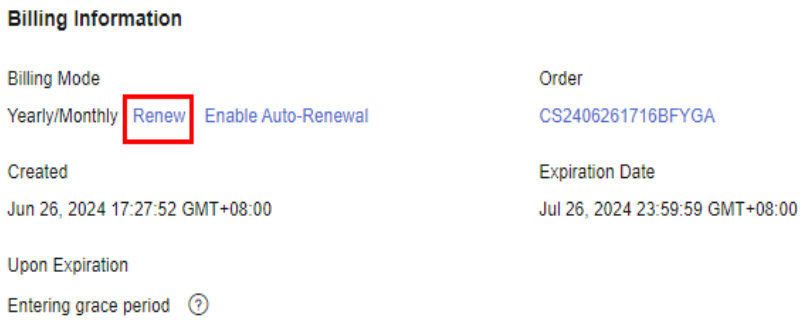
Step 3 On the **Instances** page, locate the instance that you want to renew and click **Renew** in the **Operation** column.

Figure 2-13 Renewing an instance



Alternatively, click the instance name to go to the **Basic Information** page. In the **Billing Information** area, click **Renew** next to the **Billing Mode** field.

Figure 2-14 Renewal button



NOTE

To renew multiple yearly/monthly instances at a time, perform the following steps:

1. Select the yearly/monthly instances to be renewed.
2. Click **Renew** above the instance list.

Step 4 On the displayed page, renew the instances.

----End

Renewing a Subscription in Billing Center

Step 1 Log in to the management console.

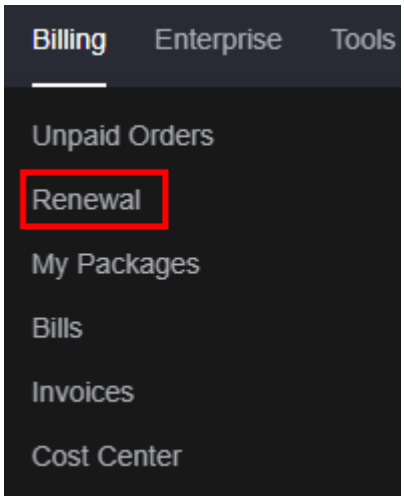
Step 2 Hover over **Billing & Costs** in the upper part of the console and choose **Renewal** from the drop-down list.

The **Renewals** page is displayed.

Step 3 Select the search criteria.

On the **Manual Renewals**, **Auto Renewals**, **Pay-per-Use After Expiration**, and **Renewals Canceled** pages, you can view the instances to be renewed.

Figure 2-15 Renewal management

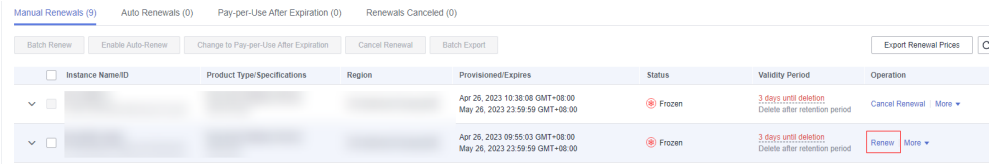


You can move all resources that need to be manually renewed to the **Manual Renewals** tab page. For details, see **Restoring to Manual Renewal**.

Step 4 Manually renew resources.

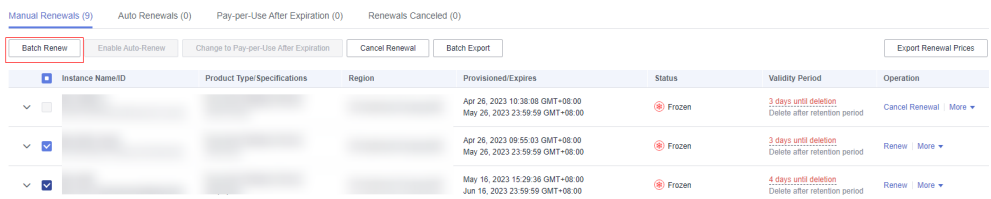
- Individual renewal: Locate an instance that you want to renew and click **Renew** in the **Operation** column.

  Figure 2-16 Individual renewal

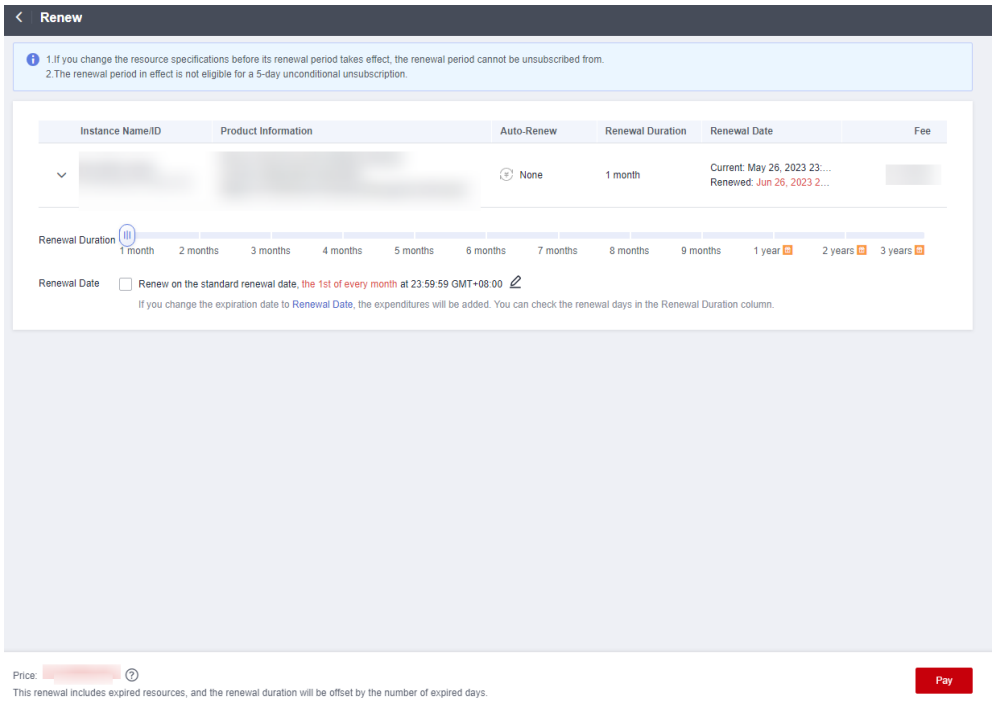
- Batch renewal: Select multiple instances that you want to renew and click **Batch Renew** in the upper left corner.

  Figure 2-17 Batch renewal

Step 5 Select a renewal duration and optionally select **Renew on the standard renewal date**. For details, see **Setting the Same Renewal Day for Yearly/Monthly Resources**. Confirm the price and click **Pay**.

Figure 2-18 Confirming renewal



Step 6 Select a payment method and make your payment. Once the order is paid for, the renewal is complete.

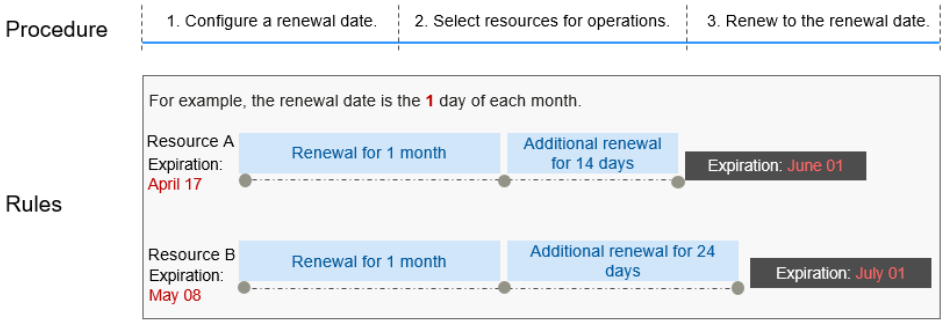
----End

Setting the Same Renewal Day for Yearly/Monthly Resources

If the instances have different expiry dates, you can set the same renewal day, for example, the first day of each month, to make it easier to manage renewals.

In **Figure 2-19**, a user sets the same renewal day for two resources that will expire at different dates.

Figure 2-19 Setting the same renewal day for resources with different expiry dates



For more details, see **Setting a Renewal Date**.

2.6.3 Auto-renewing an Instance

Auto-renewal can prevent instances from being automatically deleted if you forget to manually renew them. The auto-renewal rules are as follows:

- The first auto-renewal date is based on when an instance expires and the billing cycle.
- The auto-renewal period of an instance depends on the subscription term.
  - Monthly subscriptions renew each month.
  - Yearly subscriptions renew each year.
- You can enable auto-renewal any time before an instance expires. By default, the system will make the first attempt to charge your account for the renewal at 03:00 seven days before the expiry date. If this attempt fails, it will make another attempt at 03:00 every day until the subscription is renewed or expired.
- After auto-renewal is enabled, you can still renew the instance manually if you want to. After a manual renewal is complete, auto-renewal is still valid, and the renewal fee will be deducted from your account seven days before the new expiry date.
- By default, the renewal fee is deducted from your account seven days before the new expiry date. You can change this auto-renewal payment date as required.

For more information about auto-renewal rules, see **Auto-Renewal Rules**.

Prerequisites

Your yearly/monthly instance is not expired.

Enabling Auto-Renewal During Purchase

You can enable auto-renewal on the instance purchase page, as shown in **Figure 2-20**. For details, see **Buying an Instance**.

Figure 2-20 Enabling auto-renewal

Enabling Auto-Renewal on the Renewals Page

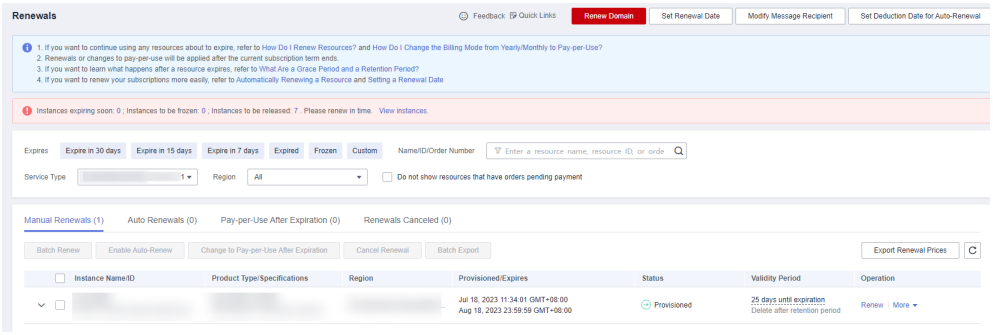
Step 1 Log in to the management console.

Step 2 Hover over **Billing & Costs** in the upper part of the console and choose **Renewal** from the drop-down list.

Step 3 Select the search criteria.

- On the **Auto Renewals** page, you can view the resources that auto-renewal has been enabled for.
- You can enable auto-renewal for resources on the **Manual Renewals**, **Pay-per-Use After Expiration**, and **Renewals Canceled** pages.

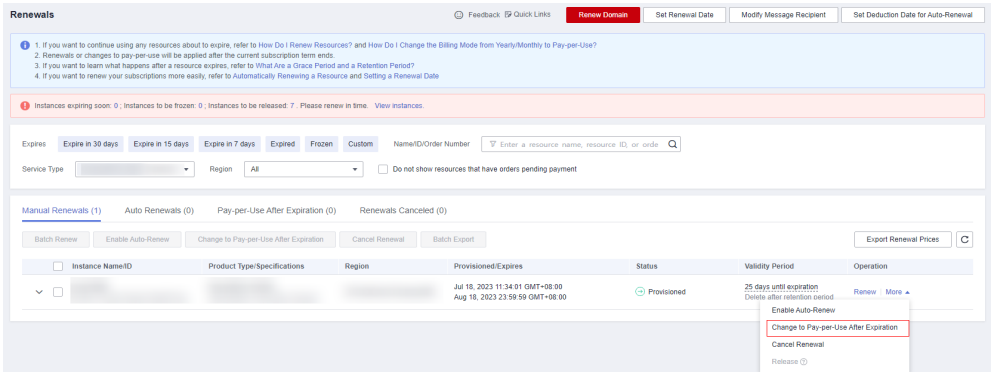
Figure 2-21 Renewal management



Step 4 Enable auto-renewal for yearly/monthly resources.

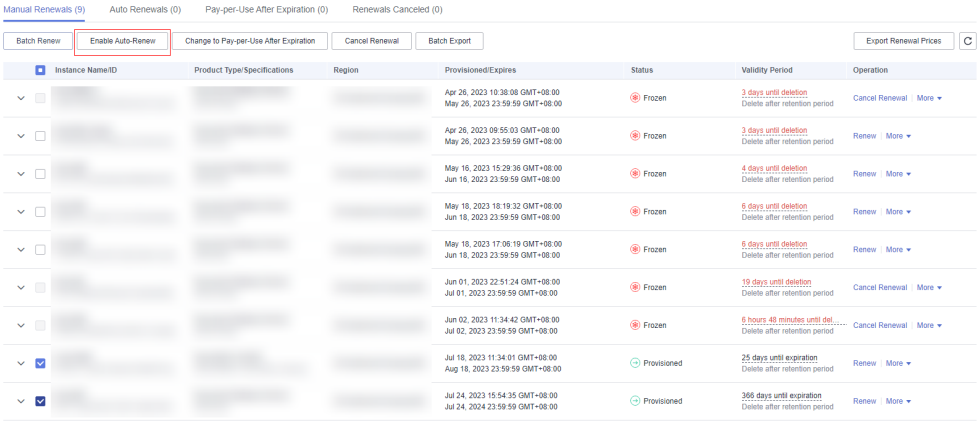
- Enabling auto-renewal for a single instance: Locate the instance that you want to enable auto-renewal for and choose **More** > **Enable Auto-Renew** in the **Operation** column.

  Figure 2-22 Enabling auto-renewal for an instance



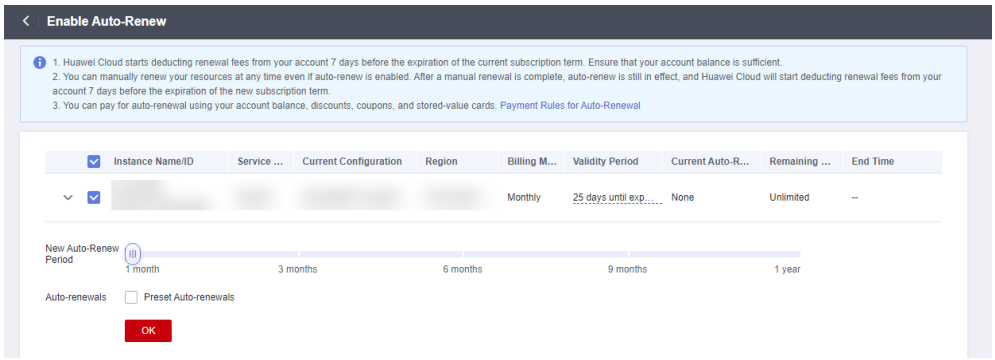
- Enabling auto-renewal for multiple instances at a time: Select the instances that you want to enable auto-renewal for and click **Enable Auto-Renew** above the list.

Figure 2-23 Enabling auto-renewal for multiple instances



Step 5 Select a renewal period, specify the auto-renewal times, and click **Pay**.

Figure 2-24 Enabling auto-renewal



----End

2.7 Bills

You can view the resource usage and bills for different billing cycles on the **Bills** page in the Billing Center.

Bill Generation

Transaction records for yearly/monthly subscriptions are generated immediately after being paid for.

A pay-per-use resource is billed by the hour, day, or month, depending on the resource's usage type. The GeminiDB Redis instance usage is billed by the hour. For details, see **Bill Run for Pay-per-Use Resources**

You are not charged immediately after a record is generated. For example, if a pay-per-use GeminiDB Redis instance (which is billed on an hourly basis) is deleted at 08:30, you will still have expenditures for the 08:00 to 09:00 hour. However, you will not likely be billed for the 08:00 to 09:00 hour until about 10:00.

On the **Bills** page of the Billing Center, select the **Bill Details** tab. **Expenditure Time** in the bill indicates the time when the pay-per-use resource is used.

Viewing Bills of a Specific Resource

[Method 1: Use the instance ID to search for a bill.]

Step 1 Log in to the management console and choose **Databases** > **GeminiDB Redis API**.

Step 2 On the **Instances** page, locate the instance whose bill you want to view and click its name.

Step 3 Click the icon shown in the figure below to copy the instance ID.

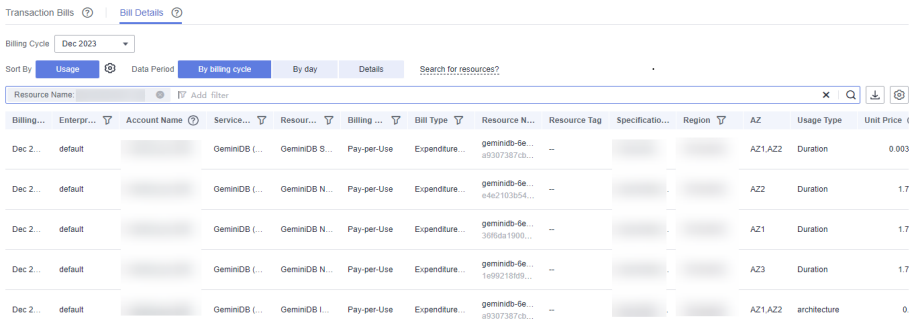
Figure 2-25 Copying the instance ID

Step 4 On the top menu bar, choose **Billing & Costs** > **Bills**.

The **Bills** page is displayed.

Step 5 Choose **Transactions and Detailed Bills** > **Bill Details**. On the displayed page, select **Resource ID** as the filter criteria, enter the obtained instance ID, and click the icon.

Figure 2-26 Searching for a bill



By default, the bill details are displayed by usage and billing cycle. You can choose other display options as required. For details, see **Bill Details**.

----End

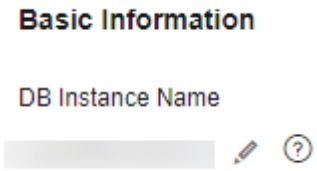
[Method 2: Use the resource name to search for a bill.]

Step 1 Log in to the management console and choose **Databases** > **GeminiDB Redis API**.

Step 2 On the **Instances** page, locate the instance whose bill you want to view and click its name.

Step 3 On the **Basic Information** page, obtain the instance name.

Figure 2-27 Obtaining an instance name

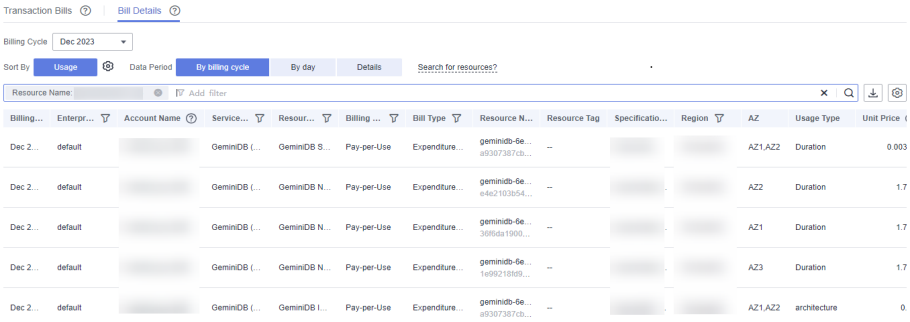


Step 4 On the top menu bar, choose **Billing & Costs** > **Bills**.

The **Bills** page is displayed.

Step 5 Choose **Transactions and Detailed Bills** > **Bill Details**. On the displayed page, select **Resource Name** as the filter criteria, enter the obtained instance ID, and click the icon.

Figure 2-28 Searching for a bill



By default, the bill details are displayed by usage and billing cycle. You can choose other display options as required. For details, see **Bill Details**.

----End

Scenario Example: Checking the Consistency of the Actual Usage and Billed Usage

Assume that you purchased a pay-per-use GeminiDB Redis instance at 10:09:06 on April 8, 2023 and deleted it later that day, at 12:09:06.

- Transaction Records

  Pay-per-use GeminiDB Redis instance usage is calculated by the second and but billed on an hourly basis. You can check the transaction records against the actual usage. The billed resources are billed separately. For details, see **Table 2-11**.

Table 2-11 GeminiDB Redis transaction records

Service Type	GeminiDB Redis
Resource Type	GeminiDB Redis storage
Billing Mode	Pay-per-use
Expenditure Time	For the period of time from 10:09:06 to 12:09:06 on April 08, 2023, 6 transaction records would be generated for the resource usage in the following periods: • 2023/04/08 10:09:06 - 2023/04/08 11:00:00 • 2023/04/08 11:00:00 - 2023/04/08 12:00:00 • 2023/04/08 12:00:00 - 2023/04/08 12:09:06
List Price	List price on the official website = Usage x Unit price x Capacity The GeminiDB Redis instance was used for 3,054 seconds in the first period, and the unit price can be obtained on the **Pricing Details** page. The list price for the first period = (3054 ÷ 3600) × 0.0007 × 40 = $0.02375333 USD. Similarly, you can calculate the GeminiDB Redis instance list price for the other periods.
Discounted Amount	Discounts offered for cloud services, for example, commercial discounts, partner authorized discounts, and promotional discounts. It is the discounted amount based on the list price.
Truncated Amount	Billing of Huawei Cloud is calculated to the 8th decimal place. However, the amount due is truncated to the 2nd decimal place. The third and later decimal places are referred to as the truncated amounts. Take the first period as an example. The truncated amount is $0.00375333 USD.
Amount Due	Amount due = List price – Discount amount – Truncated amount Take the first period as an example. If the discount amount is 0, the amount due is $0.02 USD (0.02375333 – 0 – 0.00375333).

- Bill details of the GeminiDB Redis instance

  Bill details can display in multiple ways. By default, the bill details of a resource are displayed by usage and by billing cycle. **Table 2-12** illustrates the GeminiDB Redis instance bill details, which can be used to check against the actual usage.

  Table 2-12 GeminiDB Redis bill details

Service Type	GeminiDB Redis

Resource Type	GeminiDB Redis storage
Billing Mode	Pay-per-use
Resource Name/ID	Name and ID of a specific GeminiDB Redis instance Example: nosql-b388 and 21e8811a64bf4de88bc2e2556da17983in12
Specifications	GeminiDB Redis storage
Usage Type	Duration for a pay-per-use GeminiDB Redis instance
Unit Price	When pay-per-use billing is used, the unit price is only provided if the amount is equal to the usage multiplied by the unit price. No unit price is provided in other pricing modes, for example, tiered pricing. You can search for the unit price for pay-per-use GeminiDB Redis instances on **Product Pricing Details**.
Unit	Displayed on the **Product Pricing Details** page. Example: USD/GB/hour.
Usage	Depends on the unit of the unit price, which is USD/GB/hour. Storage usage is billed by the hour. Example: 2 hours.
Usage Unit	Hour
List Price	List price on the official website = Usage x Unit price x Capacity GeminiDB Redis instance is used for 2 hours in total, and the unit price is obtained on the **Product Pricing Details** page. The list price = 2 * 0.0007 * 40 = $0.056 USD.
Discounted Amount	Discounts offered for cloud services, for example, commercial discounts, partner authorized discounts, and promotional discounts. It is the discounted amount based on the list price.
Amount Due	Amount that should be paid for used cloud services after discounts are applied.

2.8 Arrears

If your configured payment method is unable to pay for your bill, your account will be in arrears. You will need to update you payment method or to top up your account in a timely manner if you want to continue using your instance resources.

Arrears Reason

If you do not have yearly/monthly instances, your account falls into arrears any time your configured payment method is unable to pay for the used resources on the pay-per-use basis.

Arrears Impact

- Yearly/Monthly

  This is a pre-paid billing mode, so you can continue using yearly/monthly GeminiDB Redis resources even if your account is in arrears. However, you cannot perform operations such as purchasing GeminiDB Redis instances, upgrading instance specifications, and renewing subscriptions, because they will generate new expenditures.

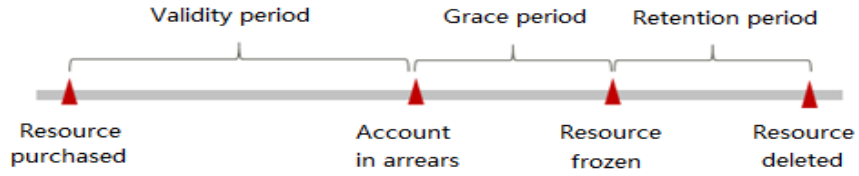
- Pay-per-Use

  If your configured payment method is unable to pay a bill for pay-per-use resources, the resources enter a grace period. After you top up your account, Huawei Cloud will bill you for expenditures generated by the resources during the grace period. You can view the expenditures on the **Overview** page of the Billing Center.

  If your account is still in arrears after the grace period ends, the resources enter the retention period and their status turns to **Frozen**. You cannot perform any operations on these resources.

  After the retention period ends, the compute resources (vCPUs and memory) and EIPs will be released and cannot be restored.

Figure 2-29 Lifecycle of a pay-per-use instance



NOTE

The grace period and retention period are both 15 days.

Avoiding and Handling Arrears

Make sure you have a valid payment method configured as soon as possible after your account is in arrears. For details, see **Topping Up an Account**.

If a GeminiDB Redis instance is no longer used, you can delete it to avoid generating further expenditures.

To help make sure your account never falls into arrears, you can configure the **Balance Alert** on the **Overview** page of the Billing Center. Then, any time an expenditure quota drops to below the threshold you specify, Huawei Cloud automatically notifies you by SMS or email.

2.9 Billing Termination

Yearly/Monthly Resources

When you purchase a yearly/monthly resource, such as a yearly/monthly GeminiDB Redis instance, you make a one-time up-front payment. By default, the billing automatically stops when the purchased subscription expires.

- If a yearly/monthly resource is no longer needed before the subscription expires, you can unsubscribe from the resource. The system will return a certain amount of money to your account based on whether the resource is subject to five-day unconditional unsubscription or whether cash coupons or discount coupons are used. For details about unsubscription rules, see **Unsubscriptions**.
- If you have enabled auto-renewal but no longer wish to automatically renew the subscription, disable it before the auto-renewal date (7 days before the expiration date by default) to avoid unexpected expenditures.

Pay-per-Use Resources

If pay-per-use resources, such as pay-per-use GeminiDB Redis instances, are no longer required, delete them in a timely manner.

Searching for Resources from Bills and Stopping Billing

To ensure that all related resources are deleted, you can search the billing records by resource ID, and then delete the resources you identify in this way.

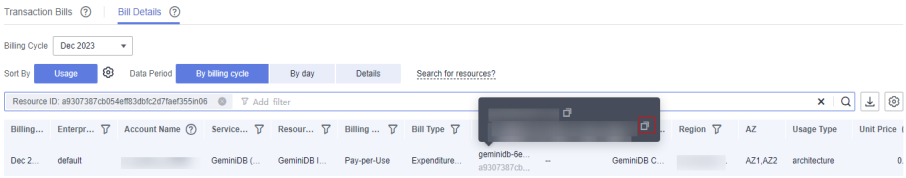
[Method 1: Use the resource ID in the bill to search for the resource.]

Step 1 Log in to the management console. On the top menu bar, choose **Billing & Costs** > **Bills**.

The **Bills** page is displayed.

Step 2 Choose **Transactions and Detailed Bills** > **Bill Details**, and click the icon shown in the following figure to copy the resource ID.

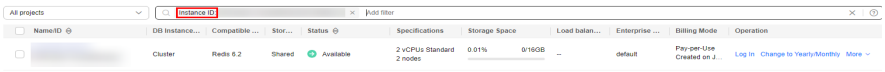
Figure 2-30 Copying the resource ID



Step 3 Log in to the management console and choose **Databases** > **GeminiDB Redis API**.

Step 4 Select the region where the resource is located, select **Instance ID** and enter the resource ID copied in **Step 2**, and click the icon to search for the resource.

Figure 2-31 Searching for resources



Step 5 Locate the instance you want to delete and click **More** > **Delete** in the **Operation** column. Ensure that the resource is not found in the list.

NOTE

You are billed one hour after the resource usage is calculated, so a bill may still be generated after the pay-per-use resource is deleted. For example, if you delete an instance (which is billed on an hourly basis) at 08:30, the expenditures for that hour from 08:00 to 09:00 are usually not billed until about 10:00.

----End

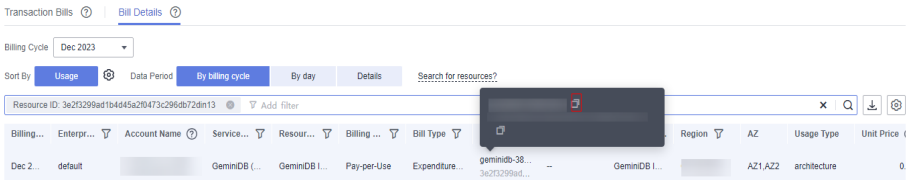
[Method 2: Use the resource name in the bill to search for the resource.]

Step 1 Log in to the management console. On the top menu bar, choose **Billing & Costs** > **Bills**.

The **Bills** page is displayed.

Step 2 Choose **Transactions and Detailed Bills** > **Bill Details**, and click the icon shown in the following figure to copy the resource name.

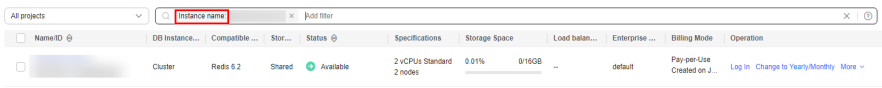
Figure 2-32 Copying the resource name



Step 3 Log in to the management console and choose **Databases** > **GeminiDB Redis API**.

Step 4 Enter the instance name copied in **Step 2** in the search box and click .

Figure 2-33 Searching for resources



Step 5 Locate the instance you want to delete and click **More** > **Delete** in the **Operation** column. Ensure that the resource is not found in the list.

NOTE

You are billed one hour after the resource usage is calculated, so a bill may still be generated after the pay-per-use resource is deleted. For example, if you delete an instance (which is billed on an hourly basis) at 08:30, the expenditures for that hour from 08:00 to 09:00 are usually not billed until about 10:00.

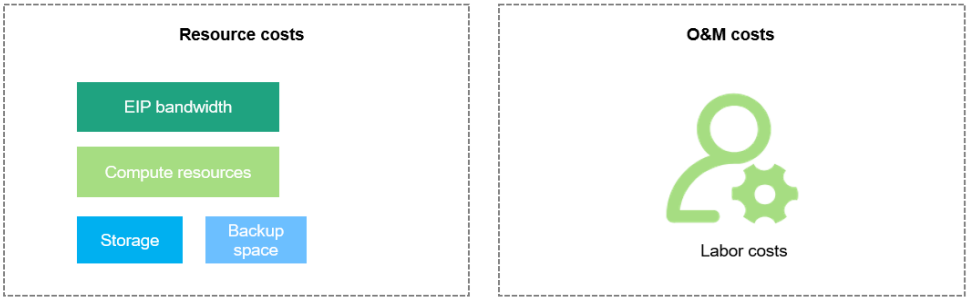
----End

2.10 Cost Management

2.10.1 Cost Composition

GeminiDB Redis costs consist of two parts:

- Resource costs: costs of compute and storage resources. For details, see **2.2 Billing Modes**.
- O&M costs: labor costs incurred during the use of GeminiDB Redis.



2.10.2 Cost Allocation

A good cost accountability system is a prerequisite for cost management. It ensures that departments, business teams, and owners are accountable for their respective cloud costs. An enterprise can allocate cloud costs to different teams or projects so as to have a clear picture of their respective costs.

Huawei Cloud **Cost Center** provides various tools for you to group costs in different ways. You can experiment with these tools and find a way that works best for you.

- **By linked account**

  The enterprise master account can manage costs by grouping the costs of its member accounts by linked account. For details, see **Viewing Costs by Linked Account**.

- **By enterprise project**

  Before allocating costs, enable Enterprise Project Management Service (EPS) and plan your enterprise projects based on your organizational structure or service needs. When purchasing cloud resources, select an enterprise project so that the costs of resources will be allocated to the selected enterprise project. For details, see **Viewing Costs by Enterprise Project**.

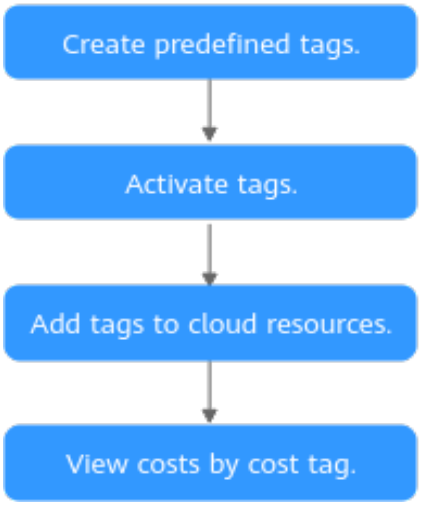
  Figure 2-34 Selecting an enterprise project

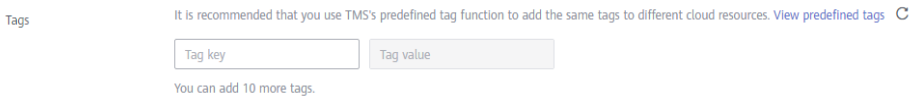
  

- **By cost tag**

You use tags to sort your Huawei Cloud resources in a variety of different ways, for example, by purpose, owner, or environment. The following is the process of managing costs by predefined tags (recommended).

Figure 2-35 Adding a tag





For details, see **Viewing Costs by Cost Tag**.

- **By cost category**

  You can use cost categories provided by **Cost Center** to split shared costs. Shared costs are the costs of resources (compute, network, storage, or resource packages) shared across multiple departments or the costs that cannot be directly split by cost tag or enterprise project. These costs are not directly attributable to a singular owner, and they cannot be categorized into a singular cost type. In this case, you can define cost splitting rules to fairly allocate these costs among teams or business units. For details, see **Viewing Cost By Cost Category**.

2.10.3 Cost Analysis

To precisely control and optimize your costs, you need a clear understanding of what parts of your enterprise incurred different costs. **Cost Center** visualizes your original costs and amortized costs using various dimensions and display filters for cost analysis so that you can analyze the trends and drivers of your service usage and costs from a variety of perspectives or within different defined scopes.

You can also use cost anomaly detection provided by **Cost Center** to detect unexpected expenses in a timely manner. In this way, costs can be monitored, analyzed, and traced.

For details, see **Performing Cost Analysis to Explore Costs and Usage** and **Enabling Cost Anomaly Detection to Identify Anomalies**.

2.10.4 Cost Optimization

You can identify resources with high costs based on the analysis results in the cost center, determine the causes of high costs, and take optimization measures accordingly.

Resource rightsizing

- View GeminiDB Redis monitoring metrics on Cloud Eye, such as the CPU, memory, and disk usage. If the current configuration is too high, you can reduce the configuration by changing specifications.
- Monitor idle GeminiDB Redis resources and delete idle instances in a timely manner.
- If your services require high performance stability, purchase a general-purpose instance to reduce your costs. For example, in the same specifications (4 vCPUs and 24 GB memory), the cost of a general-purpose instance is 30% lower than that of a dedicated instance.

Billing mode selection

Different types of services have different requirements on resource usage periods, so the most economical billing mode for one resource may not be the best option for another resource.

- For mature services that tend to be stable for the long term, select yearly/monthly billing.
- For short-term, unpredictable services that experience traffic bursts and cannot afford to be interrupted, select pay-per-use billing.
- Monitor the lifecycle of instances and renew yearly/monthly resources that are about to expire in a timely manner.

2.11 Billing FAQs

2.11.1 What Are the Differences Between Yearly/Monthly and Pay-per-Use Billing?

Yearly/Monthly is a prepaid billing mode in which resources are billed based on the service duration. This cost-effective mode is ideal when the duration of resource usage is predictable. It is recommended for long-term users.

Pay-per-use billing is a postpaid payment mode. This billing mode allows you to make or cancel subscriptions at any time. Pricing is listed on a per-hour basis, but bills are calculated based on the actual usage duration.

2.11.2 Can I Switch Between Yearly/Monthly and Pay-per-Use Billing?

You can change the billing mode of your instance from yearly/monthly to pay-per-use or vice versa.

- For details about how to change the billing mode from yearly/monthly to a pay-per-use, see **2.5.3 Yearly/Monthly to Pay-per-Use**.
- For details about how to change the billing mode from pay-per-use to yearly/monthly, see **2.5.2 Pay-per-Use to Yearly/Monthly**.

2.11.3 How Do I Renew a Single or Multiple Yearly/Monthly Instances?

This section describes how to renew your yearly/monthly GeminiDB Redis instances.

Precautions

- For billing information of a GeminiDB Redis instance, see **Billing Overview**.
- Pay-per-use instances cannot be renewed.

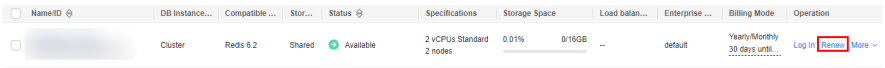
Renewing a Yearly/Monthly Instance

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

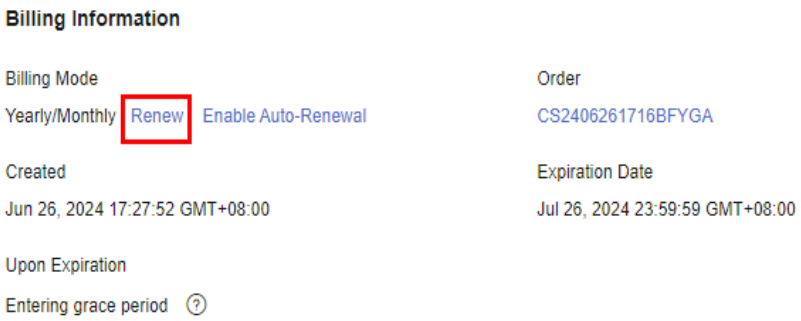
Step 3 On the **Instances** page, locate the instance that you want to renew and click **Renew** in the **Operation** column.

Figure 2-36 Renewal



Alternatively, click the instance name to go to the **Basic Information** page. In the **Billing Information** area, click **Renew** next to the **Billing Mode** field.

Figure 2-37 Renewal



Step 4 On the displayed page, renew the instance.

----End

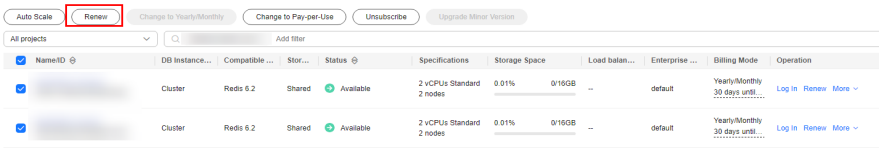
Renewing Instances in Batches

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, select the instances that you want to renew and click **Renew** above the instance list.

Figure 2-38 Renewing instances in batches



Step 4 In the displayed dialog box, click **Yes**.

----End

2.11.4 How Do I Unsubscribe from Yearly/Monthly Instances?

If you do not need a yearly/monthly instance any longer, unsubscribe from it.

Precautions

- For billing information of a GeminiDB Redis instance, see **Billing Overview**.
- The unsubscription action cannot be undone. To retain data, create a manual backup before unsubscription. For details, see **Creating a Manual Backup**.
- After an unsubscription request is submitted, resources and data will be deleted and cannot be retrieved. Ensure that the manual backup is complete before submitting the unsubscription request.

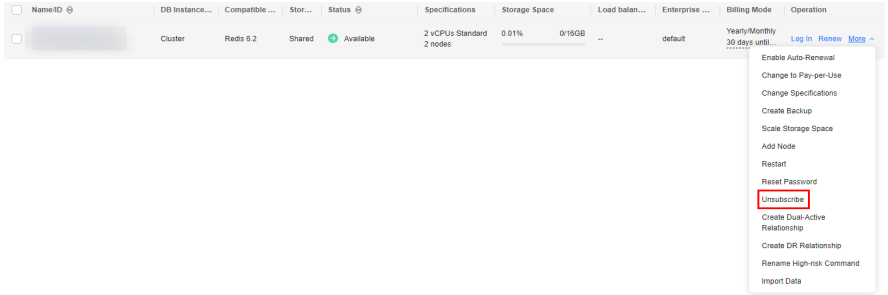
Unsubscribing from a Single Yearly/Monthly Instance

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the instance you want to unsubscribe and choose **More** > **Unsubscribe** in the **Operation** column.

Figure 2-39 Unsubscribing from a yearly/monthly instance



Step 4 In the displayed dialog box, click **Yes**.

Step 5 On the displayed page, confirm the order to be unsubscribed and select a reason. Then, click **Confirm**.

For details, see **Unsubscription Rules**.

Step 6 In the displayed dialog box, click **Yes**.

NOTICE

1. After an unsubscription request is submitted, resources and data will be deleted and cannot be retrieved.
2. Ensure that the manual backup is complete before submitting the unsubscription request.

Step 7 View the unsubscription result. After you unsubscribe from the instance order, the instance is no longer displayed in the instance list on the **Instances** page.

----End

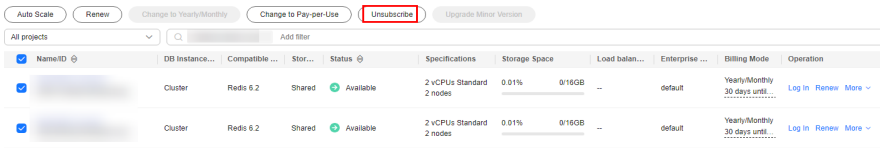
Unsubscribing from Multiple Yearly/Monthly Instances

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 Choose **Instances** in the navigation pane on the left, select the instances you want to unsubscribe from and click **Unsubscribe** above the instance list.

Figure 2-40 Unsubscribing from multiple yearly/monthly instances



Step 4 In the displayed dialog box, click **Yes**.

Step 5 On the displayed page, confirm the order to be unsubscribed and select a reason. Then, click **Confirm**.

For details, see **Unsubscription Rules**.

Step 6 In the displayed dialog box, click **Yes**.

NOTICE

1. After an unsubscription request is submitted, resources and data will be deleted and cannot be retrieved.
2. Ensure that the manual backup is complete before submitting the unsubscription request.

Step 7 View the unsubscription result. After you unsubscribe from the instance order, the instance is no longer displayed in the instance list on the **Instances** page.

----End

3 Getting Started with GeminiDB Redis API

3.1 Getting to Know GeminiDB Redis API

This section describes GeminiDB Redis product type and instance type, helping you quickly create and connect to a GeminiDB Redis instance.

Table 3-1 Product types

Product Type	Scenario	Supported Instance Type
Standard	Stable and low-latency performance is provided, suitable for common scenarios such as advertising and recommendation, gaming, e-commerce, and Internet of Vehicles (IoV).	• Proxy cluster • Redis Cluster • Primary/ Standby
Capacity-oriented	Large-capacity key-value storage is provided, suitable for average performance requirements and expectations of low costs.	Cluster

Table 3-2 Instance types

Instance Type	Scenario	Reference
Cluster	• With a sharded cluster architecture, a proxy cluster instance can be connected through proxies and is compatible with a single Redis node, Redis Sentinel, and Redis Cluster. The cluster instance has strong horizontal scaling capabilities and can handle millions of QPS and dozens of terabytes of data. • A Redis Cluster instance uses a sharded architecture and can be directly connected to Redis Cluster. This type of instance can greatly reduce latency while improving performance.	**3.2 Buying and Connecting to a Cluster Instance**
Primary/Standby	Primary/standby instances are compatible with single-node instances and can connect to a sentinel node. This instance type is used when hash tags are unavailable.	**3.3 Buying and Connecting to a Primary/Standby Instance**

Connection Methods

Data Admin Service (DAS) enables you to manage instances on a web-based console, simplifying database management and improving working efficiency. You can connect and manage instances through DAS. By default, you have the permission of remote login. DAS is secure and convenient for connecting to GeminiDB Redis instances.

Table 3-3 Connection on DAS

Method	Scenario	Remarks
DAS	You can log in to an instance on the console without using an IP address.	• Easy to use, secure, advanced, and intelligent • By default, you have the permission of remote login. DAS is secure and convenient for connecting to DB instances.

More Connection Operations

- See **4.3.1 Connection Methods**.

3.2 Buying and Connecting to a Cluster Instance

This section describes how to buy and connect to a GeminiDB Redis Cluster or proxy cluster instance on the GeminiDB console.

- With a sharded cluster architecture, a proxy cluster instance can be connected through proxies and is compatible with a single Redis node, Redis Sentinel, and Redis Cluster. The cluster instance has strong horizontal scaling capabilities and can handle millions of QPS and dozens of terabytes of data.
- A Redis Cluster instance uses a sharded architecture and can be directly connected to Redis Cluster. This type of instance can greatly reduce latency while improving performance.

Each tenant can create a maximum of 50 GeminiDB Redis instances by default. To request a higher quota, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service.

- **Step 1: Buy an instance**.
- **Step 2: Connect to the instance through DAS**.

  For details about other connection methods, see **4.3 Instance Connection and Management**.

Step 1: Buying an Instance

1. Log in to the Huawei Cloud console.
2. In the service list, choose **Databases** > **GeminiDB Redis API**.
3. On the **Instances** page, click **Buy DB Instance**.
4. On the displayed page, select a billing mode, configure instance specifications, and click **Next**.

The following parameters are for reference only. Select proper specifications as needed. **Table 4-1** lists details about the parameters.

Figure 3-1 Billing mode and basic information (proxy cluster)

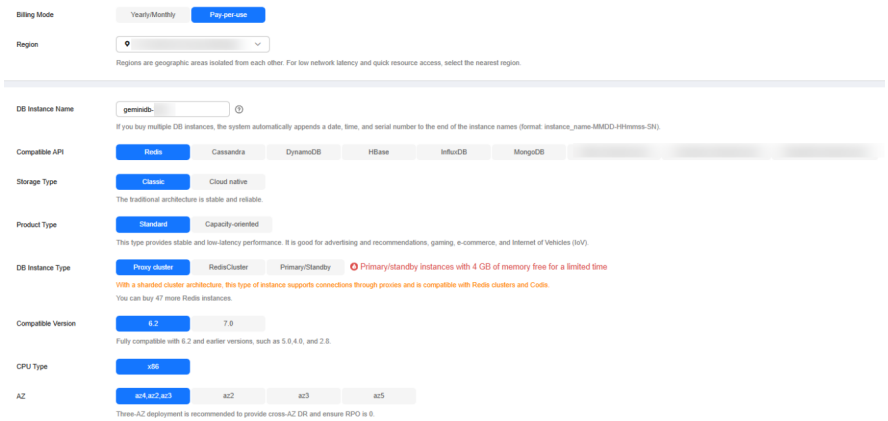
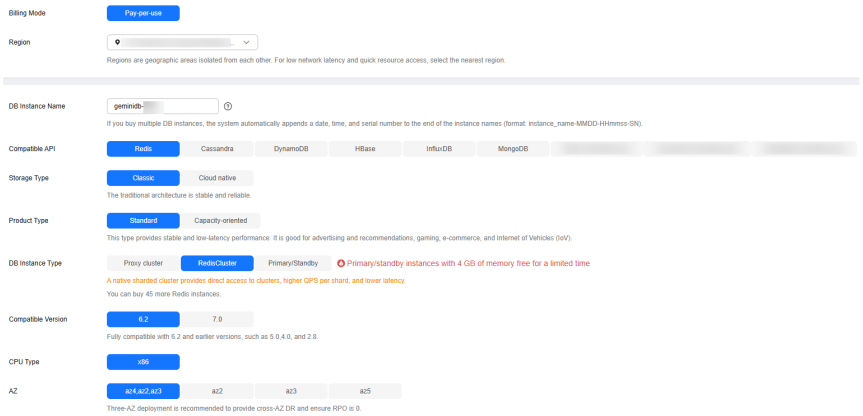


Figure 3-2 Billing mode and basic information (Redis Cluster)

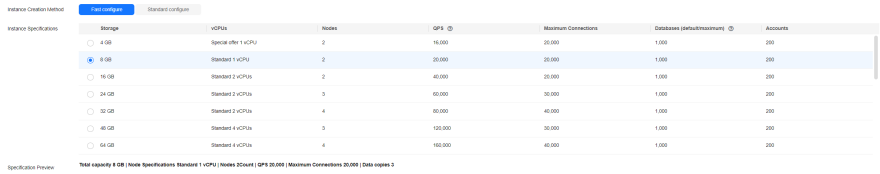


Parameter	Example Value	Description
Billing mode description	Pay-per-use	Billing mode of an instance • **Yearly/Monthly**: A prepaid billing mode in which you pay for resources before using it. Bills are settled based on the subscription period. The longer the subscription term, the bigger the discount. This mode is a good option for long-term stable services. • **Pay-per-use**: A postpaid billing mode. Pay as you go and just pay for what you use. The DB instance usage is calculated by the second but billed every hour. This mode allows you to adjust resource usage easily. You neither need to prepare for resources in advance, nor end up with excessive or insufficient preset resources.
Region	Select **CN-Hong Kong**.	Region where a tenant is located **NOTICE** To reduce network latency, select a region nearest from which you will access the instance. Instances deployed in different regions cannot communicate with each other over a private network. After you buy an instance, you cannot change its region.

Parameter	Example Value	Description
DB Instance Name	User-defined	The instance name: • Can be the same as an existing instance name. • Can contain 4 to 64 characters and must start with a letter. It is case-sensitive and allows only letters, digits, hyphens (-), and underscores (_). If the name contains Chinese characters, the length cannot exceed 64 bytes.
Compatible API	Redis.	-
Product Type	Standard	• **Standard**: Stable and low-latency performance is provided, suitable for common scenarios such as advertising and recommendation, gaming, e-commerce, and Internet of Vehicles (IoV). • **Capacity-oriented**: Large-capacity key-value storage is provided, suitable for average performance requirements and expectations of low costs.
DB Instance Type	Cluster	• **Proxy cluster**: With a sharded cluster architecture, a proxy instance can be connected through proxies and is compatible with a single Redis node, Redis Sentinel, and Redis Cluster. The cluster instance has strong horizontal scaling capabilities and can handle millions of QPS and dozens of terabytes of data. • **RedisCluster** A Redis Cluster instance uses a sharded architecture and can be directly connected to Redis Cluster. This type of instance can greatly reduce latency while improving performance. **NOTE** To create a Redis Cluster instance, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service to grant required permissions.
Compatible Version	6.2	7.0, 6.2 (including 6.2.*X*), 5.0, and earlier versions

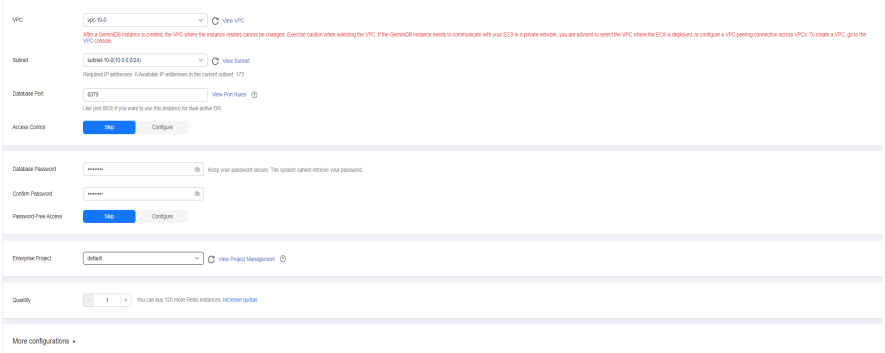
Parameter	Example Value	Description
CPU Type	x86	x86 CPUs use the Complex Instruction Set Computing (CISC) instruction set. Each instruction can be used to execute low-level hardware operations. Executing these instructions is complex and time-consuming.
AZ	AZ 1, AZ 2, and AZ 3	Availability zone where the instance is created. An AZ is a part of a region with its own independent power supplies and networks. AZs are physically isolated but can communicate with each other over a private network.

Figure 3-3 Specifications and storage



Parameter	Example Value	Description
Instance Creation Method	**Fast configure**	Two options are available: • **Fast configure** Provides you with recommended specifications. You can select one of them based on service requirements, without the need to specify the specifications, node quantity, and storage space. • **Standard configure** Provides a standard process to configure instance specifications, including specifying the specifications, node quantity, and storage space. Currently, a maximum of 36 nodes are supported. To create more nodes, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service.
Instance Specifications	2U8GB	Higher CPU specifications provide better performance. Select specifications based on your service requirements. For details, see **1.6 Instance Specifications**.

Figure 3-4 Network and database configurations



Parameter	Example Value	Description
VPC	default_vpc	Virtual private network where your instances are located. A VPC isolates networks for different services. You can select an existing VPC or create a VPC. **NOTE** • After a GeminiDB Redis instance is created, its VPC cannot be changed. • If you want to connect to a GeminiDB Redis instance through an ECS over an internal network, the GeminiDB Redis instance and the ECS must be in the same VPC. If they are not in the same VPC, you can create a **VPC peering connection** to enable access.
Subnet	default_subnet	A subnet provides dedicated network resources that are logically isolated from other networks for security purposes.
Database Port	6379	Port number for accessing a database. If you do not specify a port number, default port **6379** is used. You can specify a port number based on your requirements. The port number ranges from 1024 to 65535 except 2180, 2887, 3887, 6377, 6378, 6380, 8018, 8079, 8091, 8479, 8484, 8999, 12017, 12333, and 50069.
Access Control	**Skip**	• **Skip**: Access is restricted based on the VPC access policy by default. • **Configure**: Specify how access is controlled. Three options are available: **All IP addresses**: All IP addresses can access the instance. **Whitelist**: Only IP addresses in a group can access the instance. **Blacklist**: IP addresses in a group cannot access the instance.

Parameter	Example Value	Description
Database Password	Configured based on the password policy	Password of the administrator account. The password: • Must be 8 to 32 characters long. • Can contain at least two types of the following characters: uppercase letters, lowercase letters, digits, and special characters ~!@#%^*-_=+? • For security reasons, set a strong password. The system will verify the password strength. Keep your password secure. The system cannot retrieve it if it is lost.
Enterprise Project	default	This parameter is provided for enterprise users. An enterprise project groups cloud resources, so you can manage resources and members by project. The default project is **default**. Select an enterprise project from the drop-down list. For more information about enterprise projects, see ***Enterprise Management User Guide***.

Retain the default values for other parameters.

5. On the order confirmation page, check the instance information. If you need to modify the information, click **Previous**. If no modification is required, read and agree to the service agreement and click **Submit**.

6. Click **Back to Instance Management** to go to the instance list.

7. On the **Instances** page, view and manage the created instance.

- Creating an instance takes about 5 to 9 minutes. During the process, the instance status becomes **Creating**.
- After the creation is complete, the status changes to **Available**.

Figure 3-5 Available instance

Step 2: Connecting to an Instance Through DAS

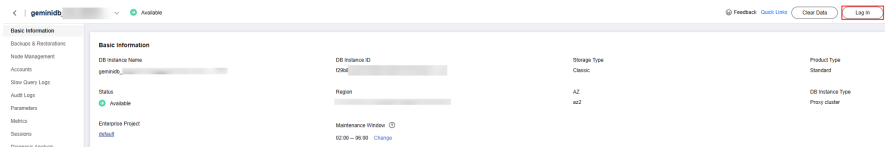
DAS enables you to manage DB instances from a web-based console, simplifying database management and improving efficiency. You can connect and manage instances through DAS. By default, you have the permission of remote login. DAS is secure and convenient for connecting to DB instances.

1. Log in to the Huawei Cloud console.
2. In the service list, choose **Databases** > **GeminiDB Redis API**.
3. In the instance list, locate the target instance and click **Log In** in the **Operation** column.

Figure 3-6 Connecting to a GeminiDB Redis instance

Alternatively, click the instance name to go to the **Basic Information** page. Click **Log In** in the upper right corner of the page.

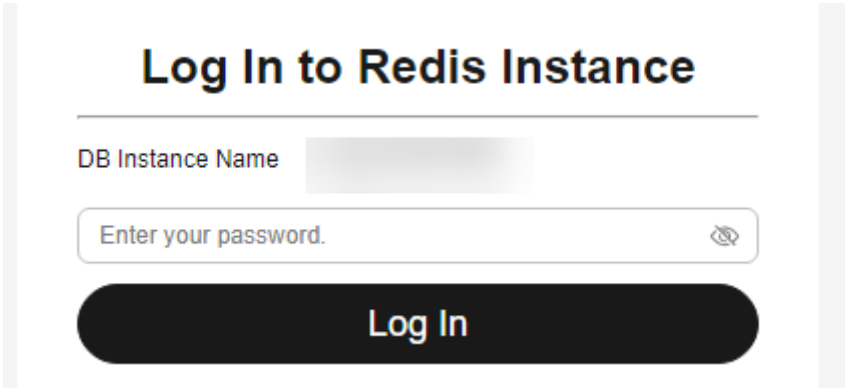
Figure 3-7 Connecting to a GeminiDB Redis instance



4. Enter a password for logging in to the instance.

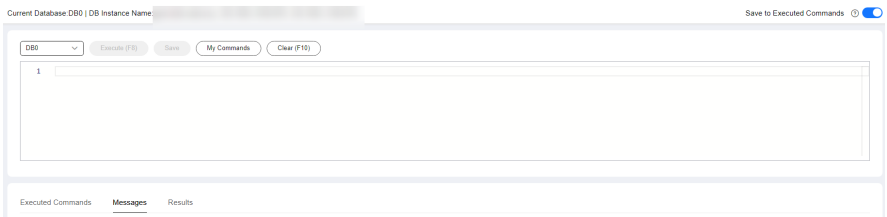
   You need to enter the password only when you log in to a GeminiDB Redis instance first time or after you reset the password.

Figure 3-8 Logging in to the GeminiDB Redis instance



5. Manage relevant databases.

Figure 3-9 Instance homepage



- Save commands to the execution record.

  This function is enabled by default to save the recently executed commands for your later query.

  Then you can click the **Executed Commands** tab on the lower page to view historical commands.

  Figure 3-10 Viewing executed commands

  If this function is disabled, the commands executed subsequently are not displayed. You can click next to **Save Executed SQL Statements** in the upper right corner to disable this function.

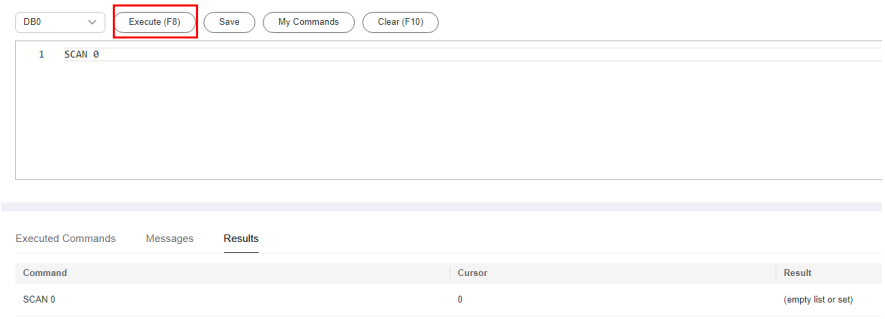
- Execute a command.

  Enter a command in the command window and click **Execute** or **F8**.

  NOTE

  - Do not use transactions, Lua scripts, Pub/Sub commands, or other commands that have blocking semantics.
  - For an instance that supports multiple databases, you can change the current database on the console but cannot change it using a SELECT statement.

  Figure 3-11 Executing a command

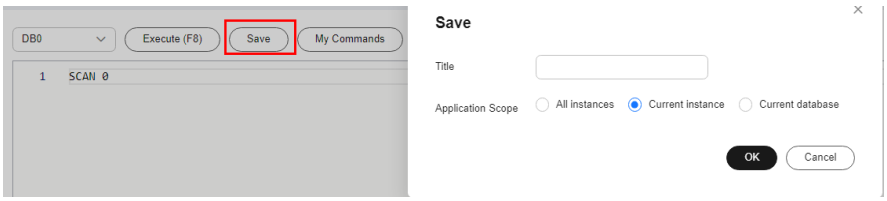
  

  After a command is executed, you can view the execution result on the **Results** page.

- Save a command.

  You can save a command to all instances, the current instance, or the current database. Then you can view details in **My Commands**.

  Figure 3-12 Saving a command

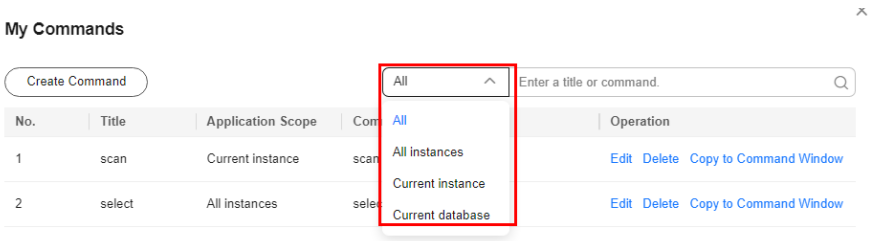
  

- View my commands.

  Common commands are displayed the **My Commands** page.

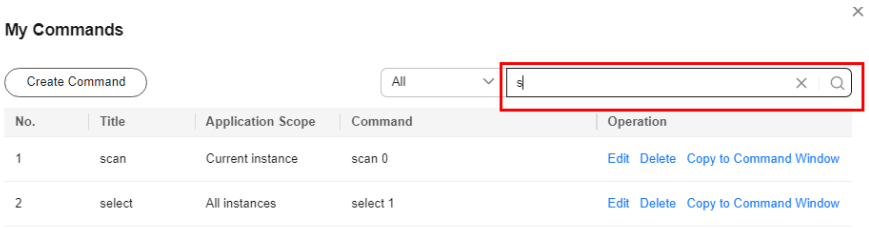
  You can set a filter to narrow the scope of commands. If you select **All**, all commands saved in the current account are displayed.

  Figure 3-13 Filtering commands

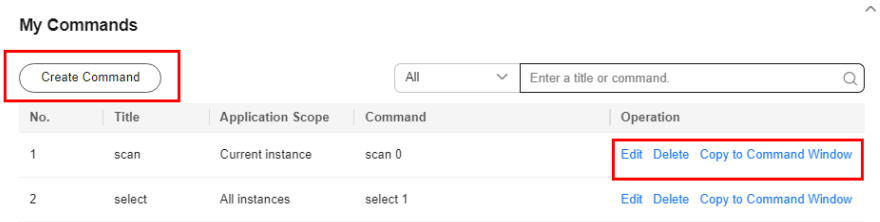
  Alternatively, you can enter a command title or statement in the search box to search for the corresponding command.

  Figure 3-14 Searching for a command

  On the **My Commands** page, you can also create, edit, and delete a command or copy it to the command window.

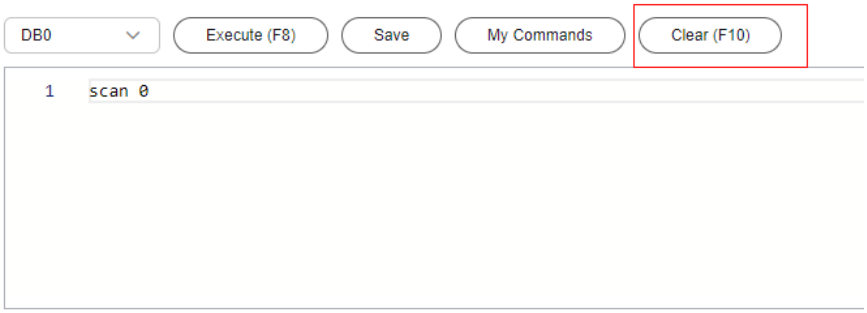
  Figure 3-15 Managing a command

- Clear a command.

  You can also press **F10** to clear the command in the command window.

Figure 3-16 Clearing a command



FAQs

Question: What should I do if the DAS console cannot be redirected after I click **Log In** in the **Operation** column in the instance list or click **Log In** on the **Basic Information** page?

Solution: Set your browser to allow pop-ups and try again.

3.3 Buying and Connecting to a Primary/Standby Instance

This section describes how to buy and connect to a primary/standby GeminiDB Redis instance on the GeminiDB console.

Primary/standby instances are compatible with single-node instances and can connect to a sentinel node. This instance type is used when hash tags are unavailable.

Each tenant can create a maximum of 50 GeminiDB Redis instances by default. To request a higher quota, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service.

- **Step 1: Buy an instance**.
- **Step 2: Connect to the instance through DAS**.

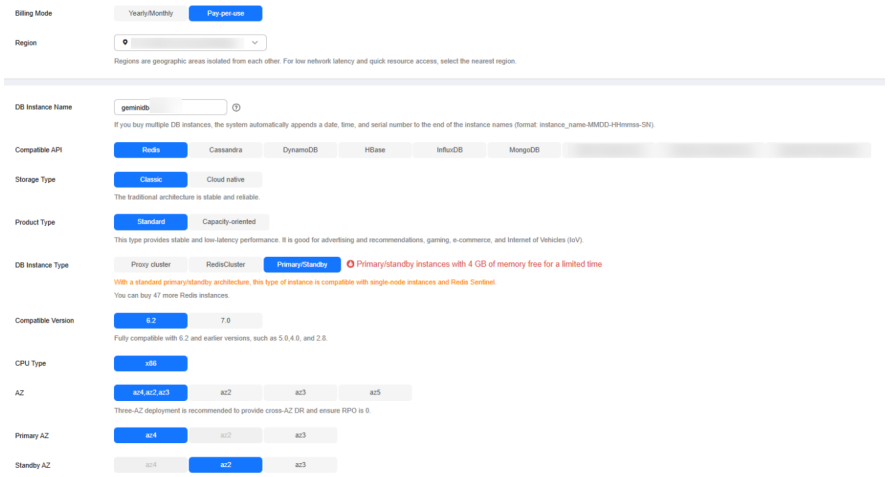
  For details about other connection methods, see **4.3 Instance Connection and Management**.

Step 1: Buying an Instance

1. Log in to the Huawei Cloud console.
2. In the service list, choose **Databases** > **GeminiDB Redis API**.
3. On the **Instances** page, click **Buy DB Instance**.
4. On the displayed page, select a billing mode, configure instance specifications, and click **Next**.

The following parameters are for reference only. Select proper specifications as needed. **Table 4-9** lists details about the parameters.

Figure 3-17 Billing mode and basic information

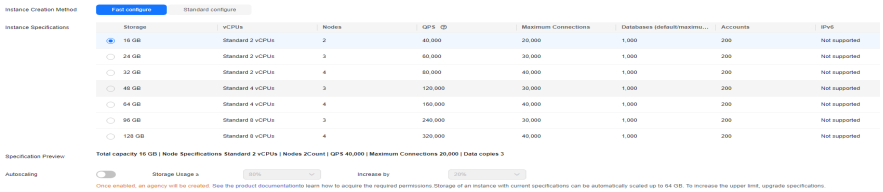


Parameter	Example Value	Description
Billing mode description	Pay-per-use	Billing mode of an instance • **Yearly/Monthly**: A prepaid billing mode in which you pay for resources before using it. Bills are settled based on the subscription period. The longer the subscription term, the bigger the discount. This mode is a good option for long-term stable services. • **Pay-per-use**: A postpaid billing mode. Pay as you go and just pay for what you use. The DB instance usage is calculated by the second but billed every hour. This mode allows you to adjust resource usage easily. You neither need to prepare for resources in advance, nor end up with excessive or insufficient preset resources.
Region	Select **CN-Hong Kong**.	Region where a tenant is located **NOTICE** To reduce network latency, select a region nearest from which you will access the instance. Instances deployed in different regions cannot communicate with each other over a private network. After you buy an instance, you cannot change its region.

Parameter	Example Value	Description
DB Instance Name	User-defined	The instance name: • Can be the same as an existing instance name. • Can contain 4 to 64 characters and must start with a letter. It is case-sensitive and allows only letters, digits, hyphens (-), and underscores (_). If the name contains Chinese characters, the length cannot exceed 64 bytes.
Compatible API	Redis.	-
Product Type	Standard	• **Standard**: Stable and low-latency performance is provided, suitable for common scenarios such as advertising and recommendation, gaming, e-commerce, and Internet of Vehicles (IoV). • **Capacity-oriented**: Large-capacity key-value storage is provided, suitable for average performance requirements and expectations of low costs.
DB Instance Type	Primary/ Standby.	Primary/Standby. Primary/standby instances are compatible with single-node instances and can connect to a sentinel node. This instance type is used when hash tags are unavailable.
Compatible Version	6.2	7.0, 6.2 (including 6.2.*X*), 5.0, and earlier versions
CPU Type	x86	x86 CPUs use the Complex Instruction Set Computing (CISC) instruction set. Each instruction can be used to execute low-level hardware operations. Executing these instructions is complex and time-consuming.

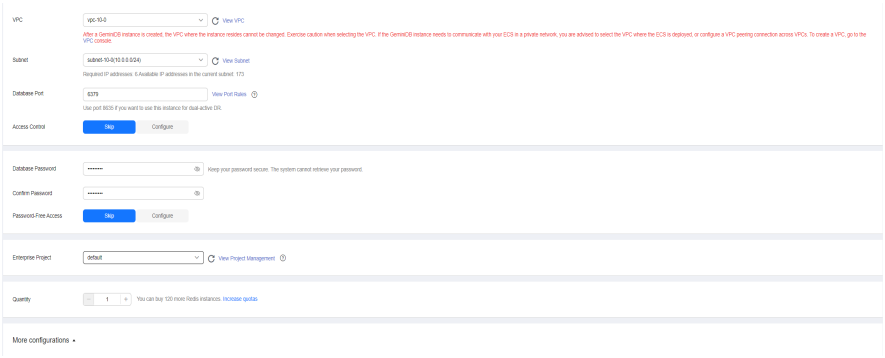
Parameter	Example Value	Description
AZ	AZ 1, AZ 2, and AZ 3	Availability zone where the instance is created. An AZ is a part of a region with its own independent power supplies and networks. AZs are physically isolated but can communicate with each other over a private network. AZs are classified into primary and standby AZs. Instances can be deployed in a single AZ or three AZs. ● If low network latency is required, deploy your instance in one AZ. ● To meet disaster recovery requirements, select three AZs and specify primary and standby AZs. – Primary AZ: AZ where a primary node is located – Standby AZ: AZ where a standby node is located

Figure 3-18 Specifications and storage



Parameter	Example Value	Description
Instance Creation Method	**Fast configure**	Two options are available: • **Fast configure** Provides you with recommended specifications. You can select one of them based on service requirements, without the need to specify the specifications, node quantity, and storage space. • **Standard configure** Provides a standard process to configure instance specifications, including specifying the specifications, node quantity, and storage space.
Instance Specifications	2U8GB	Higher CPU specifications provide better performance. Select specifications based on your service requirements. For details, see **1.6 Instance Specifications**.

Figure 3-19 Network and database configurations



Parameter	Example Value	Description
VPC	default_vpc	Virtual private network where your instances are located. A VPC isolates networks for different services. You can select an existing VPC or create a VPC. **NOTE** • After a GeminiDB Redis instance is created, its VPC cannot be changed. • If you want to connect to a GeminiDB Redis instance through an ECS over an internal network, the GeminiDB Redis instance and the ECS must be in the same VPC. If they are not in the same VPC, you can create a **VPC peering connection** to enable access.
Subnet	default_subnet	A subnet provides dedicated network resources that are logically isolated from other networks for security purposes.
Database Port	6379	Port number for accessing a database. If you do not specify a port number, default port **6379** is used. You can specify a port number based on your requirements. The port number ranges from 1024 to 65535 except 2180, 2887, 3887, 6377, 6378, 6380, 8018, 8079, 8091, 8479, 8484, 8999, 12017, 12333, and 50069.
Access Control	**Skip**	• **Skip**: Access is restricted based on the VPC access policy by default. • **Configure**: Specify how access is controlled. Three options are available: **All IP addresses**: All IP addresses can access the instance. **Whitelist**: Only IP addresses in a group can access the instance. **Blacklist**: IP addresses in a group cannot access the instance.

Parameter	Example Value	Description
Database Password	Configured based on the password policy	Password of the administrator account. The password: • Must be 8 to 32 characters long. • Can contain at least two types of the following characters: uppercase letters, lowercase letters, digits, and special characters ~!@#%^*-_=+? • For security reasons, set a strong password. The system will verify the password strength. Keep your password secure. The system cannot retrieve it if it is lost.
Enterprise project	default	This parameter is provided for enterprise users. An enterprise project groups cloud resources, so you can manage resources and members by project. The default project is **default**. Select an enterprise project from the drop-down list. For more information about enterprise projects, see ***Enterprise Management User Guide***.

Retain the default values for other parameters.

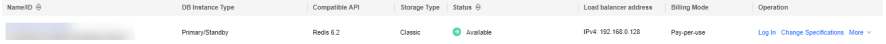
5. On the order confirmation page, check the instance information. If you need to modify the information, click **Previous**. If no modification is required, read and agree to the service agreement and click **Submit**.

6. Click **Back to Instance Management** to go to the instance list.

7. On the **Instances** page, view and manage the created instance.

- Creating an instance takes about 5 to 9 minutes. During the process, the instance status becomes **Creating**.
- After the creation is complete, the status changes to **Available**.

Figure 3-20 Available instance

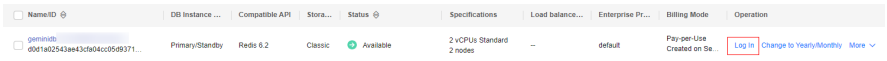


Step 2: Connecting to an Instance Through DAS

DAS enables you to manage DB instances from a web-based console, simplifying database management and improving efficiency. You can connect and manage instances through DAS. By default, you have the permission of remote login. DAS is secure and convenient for connecting to DB instances.

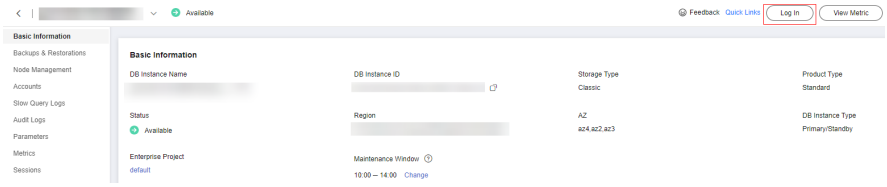
1. Log in to the Huawei Cloud console.
2. In the service list, choose **Databases** > **GeminiDB Redis API**.
3. In the instance list, locate the target instance and click **Log In** in the **Operation** column.

Figure 3-21 Connecting to a GeminiDB Redis instance



Alternatively, click the instance name to go to the **Basic Information** page. Click **Log In** in the upper right corner of the page.

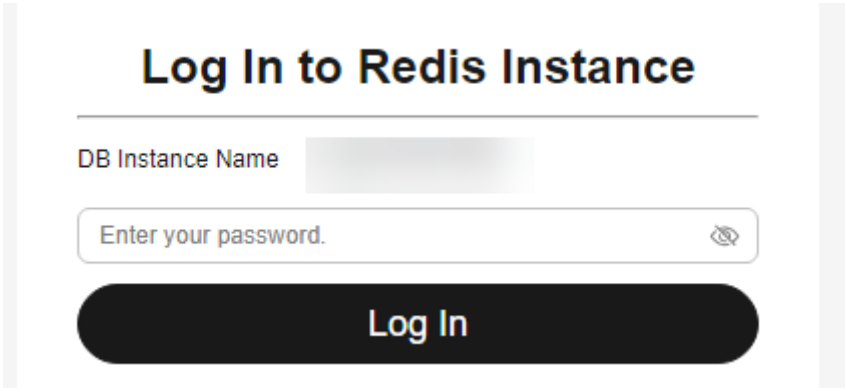
Figure 3-22 Connecting to a GeminiDB Redis instance



4. Enter a password for logging in to the instance.

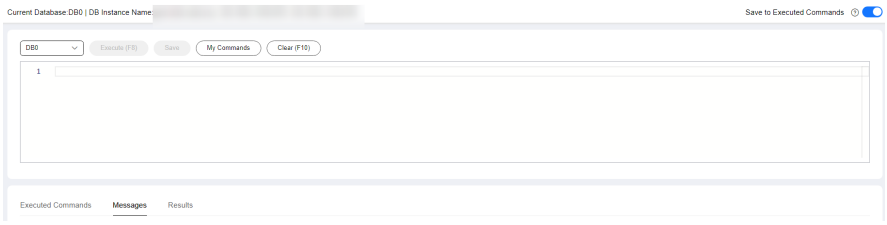
   You need to enter the password only when you log in to a GeminiDB Redis instance first time or after you reset the password.

Figure 3-23 Logging in to the GeminiDB Redis instance



5. Manage relevant databases.

Figure 3-24 Instance homepage

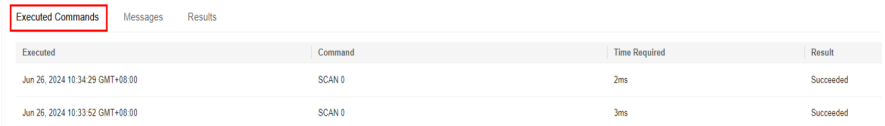


- Save commands to the execution record.

  This function is enabled by default to save the recently executed commands for your later query.

  Then you can click the **Executed Commands** tab on the lower page to view historical commands.

  Figure 3-25 Viewing executed commands

  If this function is disabled, the commands executed subsequently are not displayed. You can click next to **Save Executed SQL Statements** in the upper right corner to disable this function.

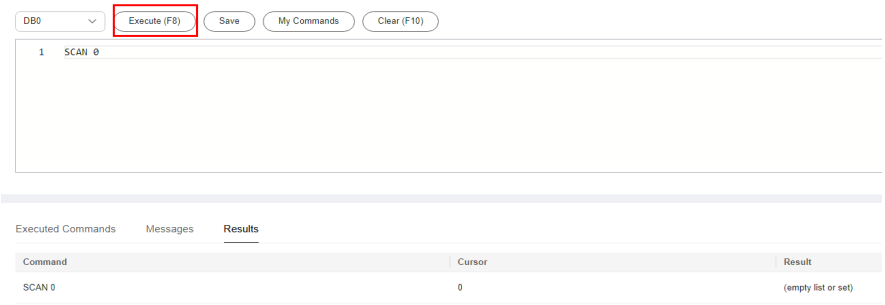
- Execute a command.

  Enter a command in the command window and click **Execute** or **F8**.

  NOTE

  - Do not use transactions, Lua scripts, Pub/Sub commands, or other commands that have blocking semantics.
  - For an instance that supports multiple databases, you can change the current database on the console but cannot change it using a SELECT statement.

  Figure 3-26 Executing a command

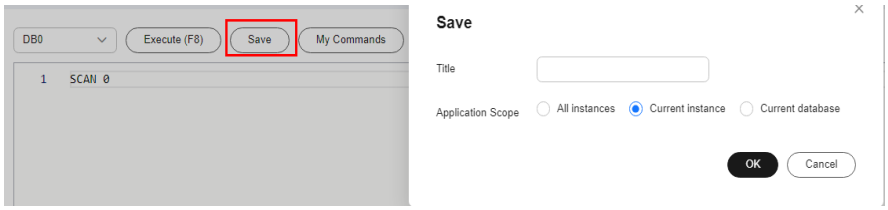
  

  After a command is executed, you can view the execution result on the **Results** page.

- Save a command.

  You can save a command to all instances, the current instance, or the current database. Then you can view details in **My Commands**.

Figure 3-27 Saving a command

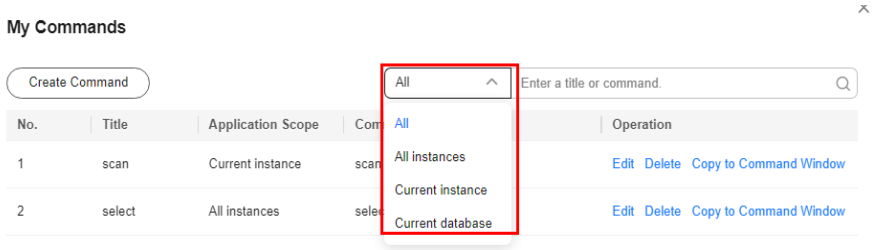


- View my commands.

  Common commands are displayed the **My Commands** page.

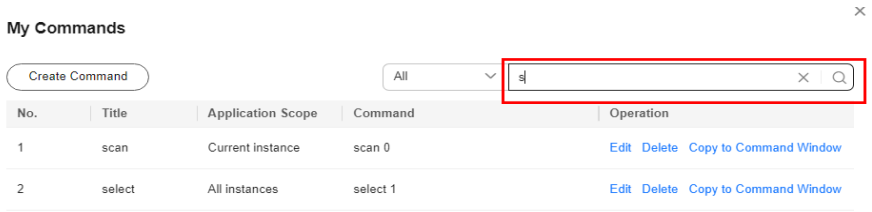
  You can set a filter to narrow the scope of commands. If you select **All**, all commands saved in the current account are displayed.

  Figure 3-28 Filtering commands

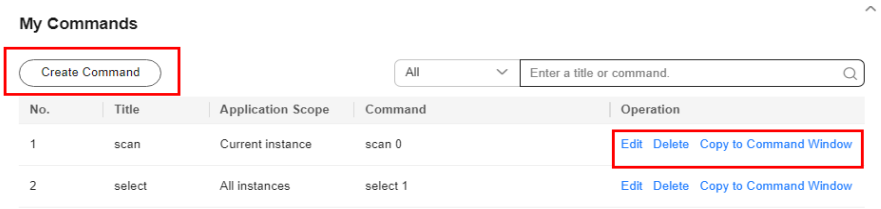
  Alternatively, you can enter a command title or statement in the search box to search for the corresponding command.

  Figure 3-29 Searching for a command

  On the **My Commands** page, you can also create, edit, and delete a command or copy it to the command window.

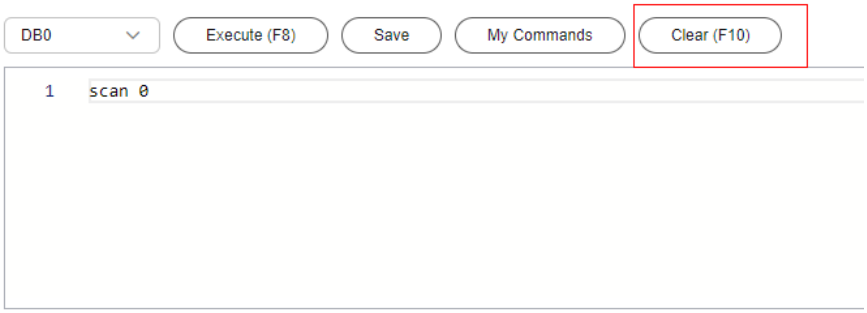
  Figure 3-30 Managing a command

- Clear a command.

  You can also press **F10** to clear the command in the command window.

Figure 3-31 Clearing a command



FAQs

Question: What should I do if the DAS console cannot be redirected after I click **Log In** in the **Operation** column in the instance list or click **Log In** on the **Basic Information** page?

Solution: Set your browser to allow pop-ups and try again.

3.4 Getting Started with Common Practices

After purchasing and connecting to a GeminiDB Redis DB instance, you can view common practices to better use it.

Table 3-4 Common practices

Reference		Description
Data migration	4.4.5 Migrating Data from Open-source Redis to GeminiDB Redis API Using Redis-Shake	Describes how to migrate data from an on-premises Redis instance to a GeminiDB Redis instance using Redis-Shake.
	4.4.8 From Kvrocks to GeminiDB Redis API	Describes how to migrate data from a Kvrocks instance to a GeminiDB Redis instance using kvrocks2redis.
	4.4.9 From Pika to GeminiDB Redis API	Describes how to migrate data from a Pika instance to a GeminiDB Redis instance using pika-port.
	4.4.10 From SSDB to GeminiDB Redis API	Describes how to migrate data from an SSDB instance to a GeminiDB Redis instance using ssdb-port.

Reference		Description
	4.4.11 From LevelDB to GeminiDB Redis API	Describes how to migrate data from a LevelDB instance to a GeminiDB Redis instance using leveldb-port.
	4.4.12 From Kvrocks to GeminiDB Redis API	Describes how to migrate data from a RocksDB instance to a GeminiDB Redis instance using rocksdb-port.
Data backup	4.7.2 Managing Automated Backups	Describes how to enable automated backup so that GeminiDB Redis API can automatically create backups for a DB instance during a backup window and saves the backups based on the configured retention period.
	4.7.3 Managing Manual Backups	Describes how to manually create backups for a DB instance. These backups can be used to restore data for improved reliability.
Data restoration	4.8.2 Restoring Data to a New Instance	Describes how to restore an existing automated or manual backup to a new instance. The restored data is the same as the backup data.
Log management	4.12.2 Viewing and Exporting Slow Query Logs	Describes how to view slow query logs of a GeminiDB Redis database. The unit of the execution time is ms. You can identify the SQL statements that take a long time to execute and tune them based on slow query logs.

4 Working with GeminiDB Redis API

4.1 Permission Management

4.1.1 Creating a User and Granting GeminiDB Redis API Permissions

This section describes how to use **IAM** to control fine-grained permissions for your GeminiDB resources. With IAM, you can:

- Create IAM users for employees based on your enterprise's organizational structure. Each IAM user will have their own security credentials for accessing GeminiDB resources.
- Grant only the permissions required for users to perform a specific task.
- Entrust a Huawei Cloud account or cloud service to perform efficient O&M on your GeminiDB resources.

If your Huawei Cloud account does not require individual IAM users, skip this section.

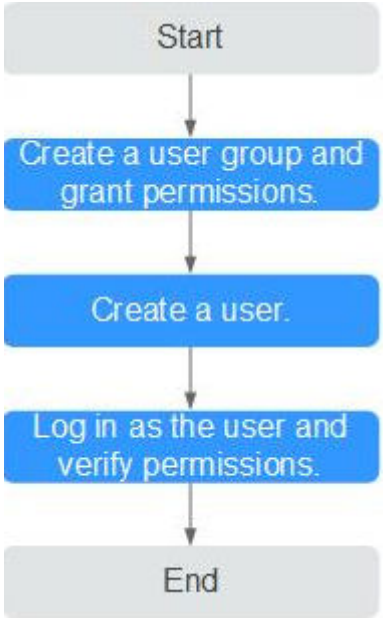
The following describes the procedure for granting permissions (see **Figure 4-1**).

Prerequisites

Learn about the permissions supported by GeminiDB and choose policies or roles based on your requirements. For details about the permissions, see **Permissions Management**. For system policies of other services, see **Permissions Policies**.

Process Flow

Figure 4-1 Process of granting GeminiDB permissions



1. **Create a user group and assign permissions** to it.

   Create a user group on the IAM console and attach the **GeminiDB FullAccess** policy to the group.

   NOTE

   To use some interconnected services, you also need to configure permissions of such services.

   For example, when using DAS to connect to a DB instance, you need to configure the **GaussDB FullAccess** and **DAS FullAccess** permissions.

2. **Create an IAM user** and add it to a user group.

   Create a user on the IAM console and add the user to the group created in **1**.

3. **Log in** and verify permissions.

   Log in to the management console using the created user, and verify the user's permissions:

   Choose **Service List** > **GeminiDB** and click **Buy DB Instance**. If you can buy an instance, the required permission policy has taken effect.

4.1.2 Custom Policies of GeminiDB Redis API

Custom policies can be created to supplement the system-defined policies of GeminiDB. For the actions supported for custom policies, see **GeminiDB Actions**.

You can create custom policies in either of the following ways:

- Visual editor: Select cloud services, actions, resources, and request conditions. This does not require knowledge of policy syntax.
- JSON: Edit JSON policies from scratch or based on an existing policy.

For details, see **Creating a Custom Policy**. The following describes examples of common GeminiDB custom policies.

Example Custom Policy

- Example 1: Allowing users to create GeminiDB instances

```
{
   "Version": "1.1",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "nosql:instance:create"
         ]
      }
   ]
}
```

- Example 2: Refusing users to delete GeminiDB instances

  A policy with only "Deny" permissions must be used in conjunction with other policies to take effect. If the policies assigned to a user contain both Allow and Deny actions, the Deny actions take precedence over the Allow actions.

  The following method can be used if you need to assign permissions of the **GeminiDB FullAccess** policy to a user but you want to prevent the user from deleting GeminiDB instances. Create a custom policy for denying instance deletion, and attach both policies to the group to which the user belongs. Then, the user can perform all operations on GeminiDB instances except deleting GeminiDB instances. The following is an example of the deny policy:

```
{
   "Version": "1.1",
   "Statement": [
      {
        "Effect": "Deny"
        "Action": [
            "nosql:instance:delete"
         ],
        }
   ]
}
```


- Example 3: Defining permissions for multiple services in a policy

  A custom policy can contain the actions of multiple services that are of the global or project-level type. The following is an example policy containing actions of multiple services:

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Action": [
                "nosql:instance:create",
                "nosql:instance:rename",
                "nosql:instance:delete",
                "vpc:publicIps:list",
                "vpc:publicIps:update"
            ],
            "Effect": "Allow"
        }
    ]
}
```

4.2 Buying a GeminiDB Redis Instance

4.2.1 Buying a GeminiDB Redis Cluster Instance

This section describes how to buy a cluster Redis instance on the GeminiDB console.

- With a sharded cluster architecture, a proxy cluster instance can be connected through proxies and is compatible with a single Redis node, Redis Sentinel, and Redis Cluster. The cluster instance has strong horizontal scaling capabilities and can handle millions of QPS and dozens of terabytes of data.
- A Redis Cluster instance uses a sharded architecture and can be directly connected to Redis Cluster. This type of instance can greatly reduce latency while improving performance.

Each tenant can create a maximum of 50 GeminiDB Redis instances by default. To request a higher quota, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service.

Prerequisites

- You have created a Huawei Cloud account.

Procedure

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click **Buy DB Instance**.

Step 4 On the displayed page, specify a billing mode and instance specifications and click **Next**.

Figure 4-2 Billing mode and basic information (proxy cluster)

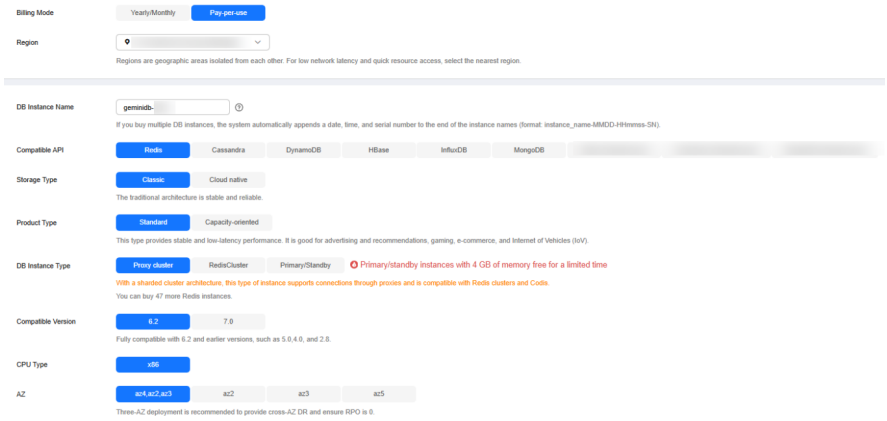


Figure 4-3 Billing mode and basic information (Redis Cluster)

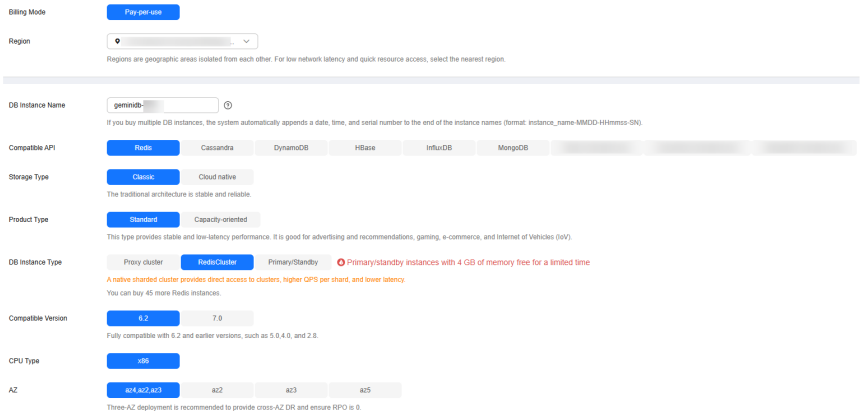


Table 4-1 Billing mode description

Parameter	Description
Billing Mode	Select **Yearly/Monthly** or **Pay-per-use**. • **Yearly/Monthly** – Specify **Required Duration**. The system deducts fees from your account based on the service price. – If you do not need such an instance any longer after it expires, change the billing mode to pay-per-use. For details, see **2.5.3 Yearly/Monthly to Pay-per-Use**. **NOTE** Yearly/Monthly instances cannot be deleted directly. If such an instance is no longer required, unsubscribe from it. For details, see **2.11.4 How Do I Unsubscribe from Yearly/Monthly Instances?**. • **Pay-per-use** – If you select this billing mode, you are billed based on how much time the instance is in use. – To use an instance for a long time, change its billing mode to yearly/monthly to reduce costs. For details, see **2.5.2 Pay-per-Use to Yearly/Monthly**.

Table 4-2 Basic information

Parameter	Description
Region	Region where a tenant is located **NOTICE** To reduce network latency, select a region nearest from which you will access the instance. Instances deployed in different regions cannot communicate with each other over a private network. After you buy an instance, you cannot change its region.
DB Instance Name	The instance name: • Can be the same as an existing instance name. • Can contain 4 to 64 characters and must start with a letter. It is case-sensitive and allows only letters, digits, hyphens (-), and underscores (_). If the name contains Chinese characters, the length cannot exceed 64 bytes. You can change the name of an instance after it is created. For details, see **4.6.2 Modifying a GeminiDB Redis Instance Name**.
Compatible API	Redis.

Parameter	Description
Product Type	• **Standard**: Stable and low-latency performance is provided, suitable for common scenarios such as advertising and recommendation, gaming, e-commerce, and Internet of Vehicles (IoV). • **Capacity-oriented**: Large-capacity key-value storage is provided, suitable for average performance requirements and expectations of low costs.
DB Instance Type	• **Proxy cluster**: With a sharded cluster architecture, a proxy cluster instance can be connected through proxies and is compatible with a single Redis node, Redis Sentinel, and Redis Cluster. The cluster instance has strong horizontal scaling capabilities and can handle millions of QPS and dozens of terabytes of data. • **RedisCluster** A Redis Cluster instance uses a sharded architecture and can be directly connected to Redis Cluster. This type of instance can greatly reduce latency while improving performance. **NOTE** To create a Redis Cluster instance, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service to grant required permissions.
Compatible Version	7.0, 6.2 (including 6.2.*X*), 5.0, and earlier versions
CPU Type	x86 x86 CPUs use the Complex Instruction Set Computing (CISC) instruction set. Each instruction can be used to execute low-level hardware operations. Executing these instructions is complex and time-consuming.
AZ	Availability zone where the instance is created. An AZ is a part of a region with its own independent power supplies and networks. AZs are physically isolated but can communicate with each other over a private network. Instances can be deployed in a single AZ or three AZs. • If low network latency is required, deploy your instance in one AZ. • If disaster recovery is required, select three AZs, and nodes of your instance will be evenly distributed across the three AZs.

Figure 4-4 Storage and specifications (standard)

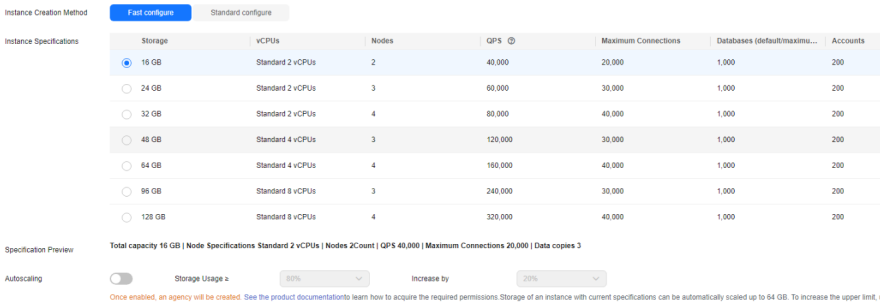


Table 4-3 Specifications and storage

Parameter	Description
Instance Creation Method	Two options are available: • **Fast configure** Provides you with recommended specifications. You can select one of them based on service requirements, without the need to specify the specifications, node quantity, and storage space. **NOTE** – Instance specifications with the memory of 8 GB and 16 GB are available only in single AZs. The console shows available specifications. – The QPS is only for reference. • **Standard configure** Provides a standard process to configure instance specifications, including specifying the specifications, node quantity, and storage space. Currently, a maximum of 36 nodes are supported. To create more nodes, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service.
Instance Specifications	Higher CPU specifications provide better performance. Select specifications based on your service requirements. For details, see **1.6 Instance Specifications**.
Specification Preview	After you select instance specifications, the system automatically shows details of the total capacity, node specifications, number of nodes, QPS benchmark, total number of connections, and number of data copies. This helps keep track of the selected instance specifications.

Parameter	Description
Autoscaling	You can determine whether to enable the function based on the site requirements. • **Storage Usage**: If the storage usage exceeds the value you set, the system automatically scales up your instance storage. The options are **60%**, **65%**, **70%**, **75%**, **80%**, **85%**, and **90%**. • **Increase by**: specifies the ratio of the increased storage space each time to the total storage space. The value can be **10%**, **15%**, or **20%**.
Static Data Encryption	You can determine whether to encrypt static data. • **Disable**: Data is not encrypted. • **Enable**: If you select this option, your data will be encrypted on disks and stored in ciphertext after you create an instance. When you download encrypted objects, the ciphertext will be decrypted into plaintext and then sent to you. Disk encryption can improve data security and may have slight impacts on database writes and reads. **Key Name**: Select an existing key or create one. **NOTE** – This function is in the OBT phase. To use it, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service. – An agency will be created after static data encryption is enabled. – After an instance is created, the static data encryption status and the key cannot be changed. – The key cannot be disabled, deleted, or frozen when it is in use. Otherwise, the database becomes unavailable. – For details about how to create a key, see "**Creating a Key**" in *Data Encryption Workshop User Guide*.

Table 4-4 Network

Parameter	Description
VPC	Virtual private network where your instances are located. A VPC isolates networks for different services. You can select an existing VPC or create a VPC. For details about how to create a VPC, see "Creating a VPC" in *Virtual Private Cloud User Guide*. With VPC sharing, you can also use a VPC and subnet shared by another account. VPC owners can share the subnets in a VPC with one or multiple accounts through Resource Access Manager (RAM), which ensures cost efficiency of network resources. For more information about VPC subnet sharing, see **VPC Sharing** in *Virtual Private Cloud User Guide*. If there are no VPCs available, the system allocates resources to you by default. **NOTE** • After a GeminiDB Redis instance is created, its VPC cannot be changed. • If you want to connect to a GeminiDB Redis instance through an ECS over an internal network, the GeminiDB Redis instance and the ECS must be in the same VPC. If they are not in the same VPC, you can create a **VPC peering connection** to enable access.
Subnet	A subnet where your instance is created. The subnet provides dedicated and isolated networks, improving network security. **NOTE** An IPv6 subnet cannot be associated with your instance. Select an IPv4 subnet.
Security Group	A security group controls access between GeminiDB Redis instances and other services. Ensure that the security group you selected allows your client to access the instance. If no security group is available, the system creates one for you.
SSL	A security protocol. Secure Sockets Layer (SSL) certificates set up encrypted connections between clients and servers, preventing data from being tampered with or stolen during transmission. You can enable SSL to improve data security. After an instance is created, you can connect to it using **SSL**. **NOTE** If SSL is not enabled when you create an instance, you can enable it after the instance is created. For details, see **4.3.5.6 Encrypting Data over SSL for a GeminiDB Redis Instance**.

Parameter	Description
Database Port	Database port number. You can specify a port number based on your requirements. The port number ranges from 1024 to 65535 except 2180, 2887, 3887, 6377, 6378, 6380, 8018, 8079, 8091, 8479, 8484, 8999, 12017, 12333, and 50069. If you do not specify a port number, port 6379 is used by default.

Table 4-5 Database configuration

Parameter	Description
Database Password	Password of database administrator **rwuser**: • Must be 8 to 32 characters long. • Can contain at least two types of the following characters: uppercase letters, lowercase letters, digits, and special characters ~!@#%^*-_=+? • For security reasons, set a strong password. The system will verify the password strength. Keep your password secure. The system cannot retrieve it if it is lost.
Confirm Password	Enter the administrator password again.
Enterprise Project	This parameter is provided for enterprise users. An enterprise project groups cloud resources, so you can manage resources and members by project. The default project is **default**. Select an enterprise project from the drop-down list. For more information about enterprise projects, see ***Enterprise Management User Guide***.

Table 4-6 Password-free access configuration

Parameter	Description
Password-Free Access	Configure password-free access for a CIDR Block of the instance you want to access. After that, the password is not required the instance access. • **Skip** If you select **Skip**, you can set password-free access after the GeminiDB Redis instance is created. For details, see **4.10.1 Enabling Password-Free Access**. • **Configure** Enter a CIDR block that you want to enable password-free access for. A maximum of 30 password-free CIDR blocks can be configured.

Table 4-7 Tags

Parameter	Description
Tags	This setting is optional. Adding tags helps you better identify and manage your instances. A maximum of 20 tags can be added for each instance. If your organization has configured a tag policy for your GeminiDB Redis instance, you need to add a tag to the instance based on the tag policy. If the tag does not comply with the tag policy, the instance may fail to be created. Contact the organization administrator to learn details about the tag policy. A tag consists of a tag key and a tag value. • Tag key: mandatory if the instance is going to be tagged. Each tag key is unique for each instance. It can include up to 36 characters, including digits, letters, underscores (_), and hyphens (-). • Tag value: optional if the instance is going to be tagged. The value can be empty. The value can contain up to 43 characters, including digits, letters, underscores (_), periods (.), and hyphens (-). After an instance is created, you can view its tag details on the **Tags** tab. In addition, you can add, modify, and delete tags of an existing instance. For details, see **4.14 Tag Management**.

Table 4-8 Required duration

Parameter	Description
Required Duration	The length of your subscription if you select **Yearly/Monthly** billing. Subscription lengths range from one month to three years.
Auto-renew	• By default, this option is not selected. • If you select this parameter, the auto-renew cycle is determined by the selected required duration.

Step 5 On the displayed page, confirm instance details.

- Yearly/Monthly
  - To modify the configurations, click **Previous**.
  - If no modification is required, read and agree to the service agreement, click **Pay Now**, and complete the payment.
- Pay-per-use
  - To modify the configurations, click **Previous**.
  - If no modification is required, read and agree to the service agreement and click **Submit**.

Step 6 On the **Instances** page, view and manage the created instance.

The instance creation process takes about 5 to 15 minutes. After the creation is complete, the status changes to **Available**.

You can click  in the upper right corner of the page to refresh the instance status.

----End

4.2.2 Buying a Primary/Standby GeminiDB Redis Instance

This section describes how to buy a primary/standby Redis instance on the GeminiDB console.

Each tenant can create a maximum of 50 GeminiDB Redis instances by default. To request a higher quota, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service.

Prerequisites

- You have created a Huawei Cloud account.

Procedure

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instance Management** page, click **Buy DB Instance**.

Step 4 On the displayed page, select a billing mode, select instance specifications and click **Next**.

Figure 4-5 Billing mode and basic information

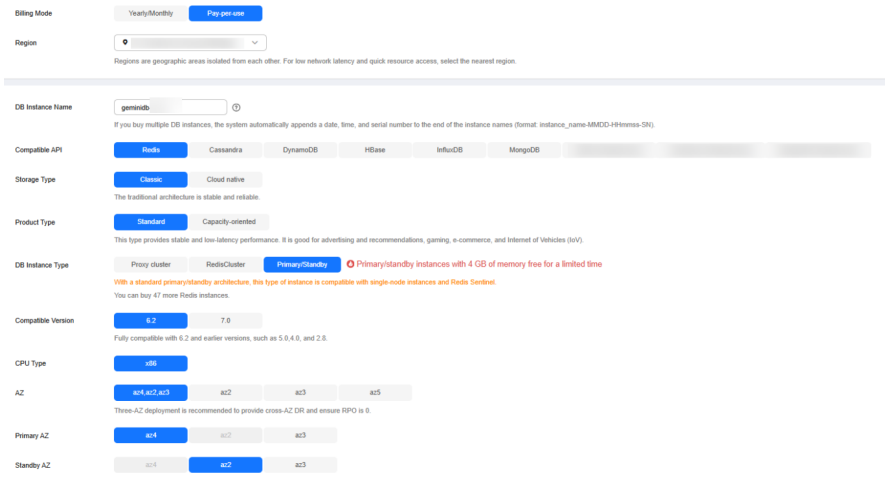


Table 4-9 Billing mode description

Parameter	Description
Billing Mode	Select **Yearly/Monthly** or **Pay-per-use**. • **Yearly/Monthly** – Specify **Required Duration**. The system deducts fees from your account based on the service price. – If you do not need such an instance any longer after it expires, change the billing mode to pay-per-use. For details, see **2.5.3 Yearly/Monthly to Pay-per-Use**. **NOTE** Yearly/Monthly instances cannot be deleted directly. If such an instance is no longer required, unsubscribe from it. For details, see **2.11.4 How Do I Unsubscribe from Yearly/Monthly Instances?**. • **Pay-per-use** – If you select this billing mode, you are billed based on how much time the instance is in use. – To use an instance for a long time, change its billing mode to yearly/monthly to reduce costs. For details, see **2.5.2 Pay-per-Use to Yearly/Monthly**.

Table 4-10 Basic information

Parameter	Description
Region	Region where a tenant is located **NOTICE** To reduce network latency, select a region nearest from which you will access the instance. Instances deployed in different regions cannot communicate with each other over a private network. After you buy an instance, you cannot change its region.
DB Instance Name	The instance name: • Can be the same as an existing instance name. • Can contain 4 to 64 characters and must start with a letter. It is case-sensitive and allows only letters, digits, hyphens (-), and underscores (_). If the name contains Chinese characters, the length cannot exceed 64 bytes. You can change the name of an instance after it is created. For details, see **4.6.2 Modifying a GeminiDB Redis Instance Name**.
Compatible API	Redis.
DB Instance Type	Primary/Standby. Primary/standby instances are compatible with single-node instances and can connect to a sentinel node. This instance type is used when hash tags are unavailable.
Compatible Version	7.0, 6.2 (including 6.2.*X*), 5.0, and earlier versions
CPU Type	x86. x86 CPUs use the Complex Instruction Set Computing (CISC) instruction set. Each instruction can be used to execute low-level hardware operations. Executing these instructions is complex and time-consuming.
AZ	Availability zone where the instance is created. An AZ is a part of a region with its own independent power supplies and networks. AZs are physically isolated but can communicate with each other over a private network. AZs are classified into primary and standby AZs. Instances can be deployed in a single AZ or three AZs. • If low network latency is required, deploy your instance in one AZ. • To meet disaster recovery requirements, select three AZs and specify primary and standby AZs. – Primary AZ: AZ where a primary node is located – Standby AZ: AZ where a standby node is located

Figure 4-6 Specifications and storage

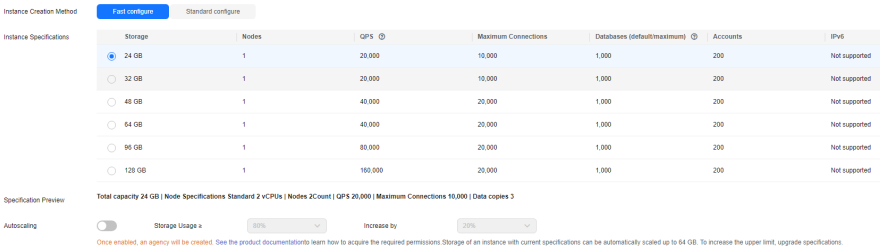


Table 4-11 Specifications and storage

Parameter	Description
Instance Creation Method	Two options are available: • **Fast configure** Provides you with recommended specifications. You can select one of them based on service requirements, without the need to specify the specifications, node quantity, and storage space. **NOTE** – The QPS is only for reference. • **Standard configure** Provides a standard process to configure instance specifications, including specifying the specifications, node quantity, and storage space.
Instance Specifications	Higher CPU specifications provide better performance. Select specifications based on your service requirements. For details, see **1.6 Instance Specifications**.
Specification Preview	After you select instance specifications, the system automatically shows details of the total capacity, node specifications, number of nodes, QPS benchmark, total number of connections, and number of data copies. This helps keep track of the selected instance specifications.
Autoscaling	You can determine whether to enable the function based on the site requirements. • **Storage Usage**: If the storage usage exceeds the value you set, the system automatically scales up your instance storage. The options are **60%**, **65%**, **70%**, **75%**, **80%**, **85%**, and **90%**. • **Increase by**: specifies the ratio of the increased storage space each time to the total storage space. The value can be **10%**, **15%**, or **20%**.

Parameter	Description
Static Data Encryption	You can determine whether to encrypt static data. • **Disable**: Data is not encrypted. • **Enable**: If you select this option, your data will be encrypted on disks and stored in ciphertext after you create an instance. When you download encrypted objects, the ciphertext will be decrypted into plain text and then sent to you. Disk encryption can improve data security and may have slight impacts on database writes and reads. **Key Name**: Select an existing key or create one. **NOTE** – This function is in the OBT phase. To use it, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service. – An agency will be created after static data encryption is enabled. – After an instance is created, the static data encryption status and the key cannot be changed. – The key cannot be disabled, deleted, or frozen when it is in use. Otherwise, the database becomes unavailable. – For details about how to create a key, see "**Creating a Key**" in *Data Encryption Workshop User Guide*.

Table 4-12 Network

Parameter	Description
VPC	Virtual private network where your instances are located. A VPC isolates networks for different services. You can select an existing VPC or create a VPC. For details about how to create a VPC, see "Creating a VPC" in *Virtual Private Cloud User Guide*. With VPC sharing, you can also use a VPC and subnet shared by another account. VPC owners can share the subnets in a VPC with one or multiple accounts through Resource Access Manager (RAM), which ensures cost efficiency of network resources. For more information about VPC subnet sharing, see **VPC Sharing** in *Virtual Private Cloud User Guide*. If there are no VPCs available, the system allocates resources to you by default. **NOTE** • After an instance is created, the VPC where the instance is deployed cannot be changed. • If you want to connect to an instance using an ECS over a private network, ensure that the instance and the ECS are in the same VPC. If they are not, create a **VPC peering connection** between them.

Parameter	Description
Subnet	A subnet where your instance is created. The subnet provides dedicated and isolated networks, improving network security. **NOTE** An IPv6 subnet cannot be associated with your instance. Select an IPv4 subnet.
Security Group	A security group controls access between instances and other services. Ensure that the security group you selected allows your client to access the instance. If no security group is available, the system creates one for you.
SSL	A security protocol. Secure Sockets Layer (SSL) certificates set up encrypted connections between clients and servers, preventing data from being tampered with or stolen during transmission. You can enable SSL to improve data security. After an instance is created, you can connect to it using **SSL**. **NOTE** If SSL is not enabled when you create an instance, you can enable it after the instance is created. For details, see **4.3.5.6 Encrypting Data over SSL for a GeminiDB Redis Instance**.
Database Port	Database port number. You can specify a port number based on your requirements. The port number ranges from 1024 to 65535 except 2180, 2887, 3887, 6377, 6378, 6380, 8018, 8079, 8091, 8479, 8484, 8999, 12017, 12333, and 50069. If you do not specify a port number, port 6379 is used by default.

Figure 4-7 Database configuration

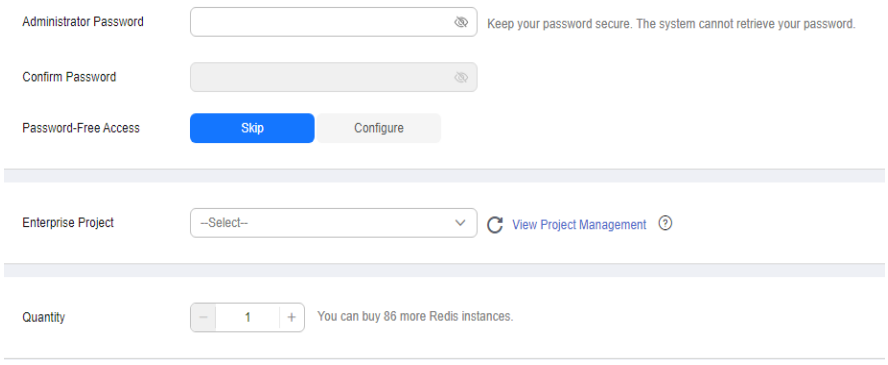


Table 4-13 Database configuration

Parameter	Description
Administrator Password	Password of database administrator **rwuser**: • Must be 8 to 32 characters long. • Can include two of the following: uppercase letters, lowercase letters, digits, and special characters: ~!@#$ %^&*()-_=+?$()& • For security reasons, set a strong password. The system will verify the password strength. Keep your password secure. The system cannot retrieve it if it is lost.
Confirm Password	Enter the administrator password again.
Enterprise project	This parameter is provided for enterprise users. An enterprise project groups cloud resources, so you can manage resources and members by project. The default project is **default**. Select an enterprise project from the drop-down list. For more information about enterprise projects, see ***Enterprise Management User Guide***.

Table 4-14 Password-free access configuration

Parameter	Description
Password-Free Access	Configure password-free access for a CIDR Block of the instance you want to access. After that, the password is not required the instance access. • **Skip** If you select **Skip**, you can set password-free access after the GeminiDB Redis instance is created. For details, see **4.10.1 Enabling Password-Free Access**. • **Configure** Enter a CIDR block that you want to enable password-free access for. A maximum of 30 password-free CIDR blocks can be configured.

Table 4-15 Tag

Parameter	Description
Tag	The setting is optional. Adding tags helps you better identify and manage your instances. A maximum of 20 tags can be added for each instance. If your organization has configured a tag policy for GeminiDB Redis, you need to add a tag to the instance based on the tag policy. If the tag does not comply with the tag policy, the instance may fail to be created. Contact the organization administrator to learn details about the tag policy. A tag consists of a tag key and a tag value. • Tag key: mandatory if the instance is going to be tagged. Each tag key is unique for each instance. The key can include up to 36 characters, including digits, letters, underscores (_), and hyphens (-). • Tag value: optional if the instance is going to be tagged. The value can be empty. The value can contain up to 43 characters, including digits, letters, underscores (_), periods (.), and hyphens (-). After an instance is created, you can view its tag details on the **Tags** tab. In addition, you can add, modify, and delete tags of an existing instance. For details, see **4.14 Tag Management**.

Table 4-16 Required duration

Parameter	Description
Required duration	The length of your subscription if you select **Yearly/Monthly** billing. Subscription lengths range from one month to three years.
Auto-renewing an Instance	• By default, this option is not selected. • If you select this option, the auto-renew cycle is determined by the selected required duration.

Step 5 On the displayed page, confirm instance details.

- Yearly/Monthly
  - To modify the configurations, click **Previous**.
  - If no modification is required, read and agree to the service agreement, click **Pay Now**, and complete the payment.
- Pay-per-use
  - To modify the configurations, click **Previous**.
  - If no modification is required, read and agree to the service agreement and click **Submit**.

Step 6 On the **Instances** page, view and manage the created instance.

The instance creation process takes about 5 to 15 minutes. After the creation is complete, the status changes to **Available**.

You can click  in the upper right corner of the page to refresh the instance status.

----End

4.3 Instance Connection and Management

4.3.1 Connection Methods

GeminiDB Redis API is compatible with open-source Redis and allows traffic from applications using different types of SDKs. It can also be accessed through Data Admin Service (DAS), private networks, and public networks.

Figure 4-8 shows the process of connecting to a GeminiDB Redis instance.

Figure 4-8 Connection Methods

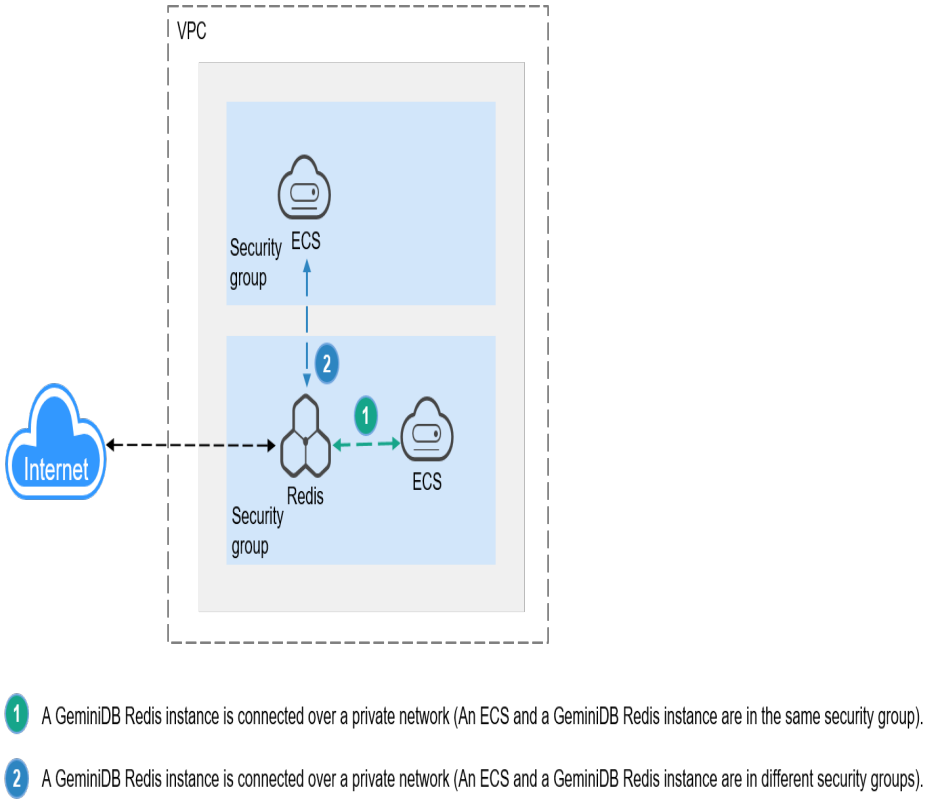


Table 4-17 Connection methods

Method	Scenario	Default Port	Description
DAS	You can connect to a GeminiDB Redis instance using a web-based console client.	-	-
Private network	You can connect to a GeminiDB Redis instance through a **private IP address**, **private domain name**, or **load balancer address**. This method is suitable when your application is deployed on an ECS that is in the same region and VPC as your instance.	6379	• You are advised to use the load balancer address to connect to the instance. This ensures high reliability and eliminates the impact of SPOFs. • High security and performance • If the ECS and GeminiDB Redis instance are in the same security group, they can communicate with each other by default. No security group rule needs to be configured. • If they are in different security groups, configure security group rules for them, separately. – Configure inbound rules of a security group for GeminiDB Redis instances by following **4.3.5.3 Configuring Security Group Rules for a GeminiDB Redis Instance**. – The default security group rule allows all outbound data packets, so you do not need to configure a security rule for the ECS. If not all access from the ECS is allowed, you need to configure an outbound rule for the ECS.

Method	Scenario	Default Port	Description
Public network	You can connect to a GeminiDB Redis instance through a **public domain name** or an **EIP**. This method is suitable when an instance cannot be accessed over a private network. You can connect to the instance from an ECS using a public domain name or an EIP.	6379	• For faster transmission and improved security, migrate your applications to an ECS that is in the same subnet as your instance and use a private IP address to access the instance. • Use a public domain name to ensure high reliability and eliminate SPOFs. • You need to purchase an EIP. For details, see **Billing Overview**.
Program code	You can connect to a GeminiDB Redis instance using different code. For details, see **5.3 Examples of Connecting to an Instance Using Programming Languages**.	6379	-

4.3.2 Connecting to a GeminiDB Redis Instance on the DAS Console

DAS enables you to manage DB instances from a web-based console, simplifying database management and improving efficiency. You can connect and manage instances through DAS. By default, you have permissions required for remote login. DAS is secure and convenient for connecting to DB instances.

Precautions

- If SSL is enabled, you cannot connect to a GeminiDB Redis instance through DAS.

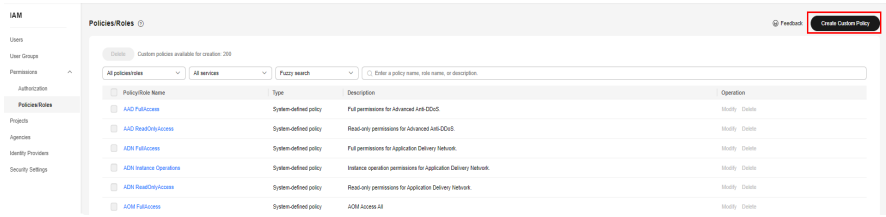
Configuring the Required Permissions

If you have an IAM account, assign DAS FullAccess permissions to all users of the account. For details, see **Create User Groups and Assign Permissions**.

You can create a custom policy to specify the type of databases that you have permissions for.

1. Log in to the IAM console and choose **Permissions** > **Policies/Roles**.

Figure 4-9 Creating a custom policy



2. Specify a policy name, policy view, and content.

Figure 4-10 Configuring a custom policy

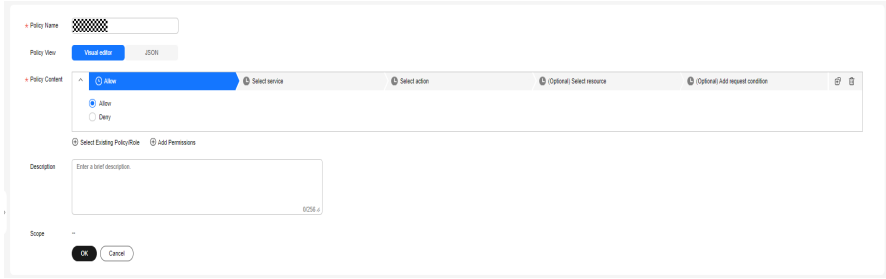


Table 4-18 Custom policy description

Parameter	Description
Policy Name	Enter a policy name.
Policy View	Select **JSON**.

Policy Content	Configure the following policy content:
	{
	"Version": "1.1",
	"Statement": [
	{
	"Action": [
	"das:*:*",
	"nosql:instance:list"
	],
	"Effect": "Allow"
	}
	]
	}
	Alternatively, click **Select Existing Policy/Role**, select **DAS FullAccess** as a template, and retain only the DB type information. In this example, retain only **nosql:instance:list**.

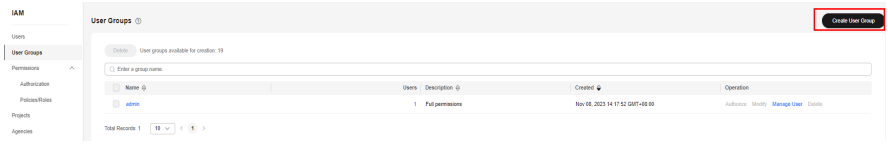
Description	Enter a policy description.
Scope	Retain the default settings (project-level service).

3. Click **OK**. You can then view the created custom policy on the **Permissions** page.

Figure 4-11 Viewing the created policy

4. Create a user group.

Figure 4-12 Creating a user group



5. Authorize the user group created in **4** using the created custom policy.

Figure 4-13 Authorizing the user group using the created custom policy

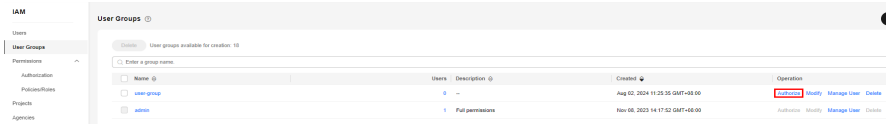
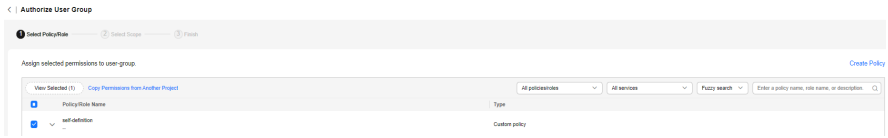
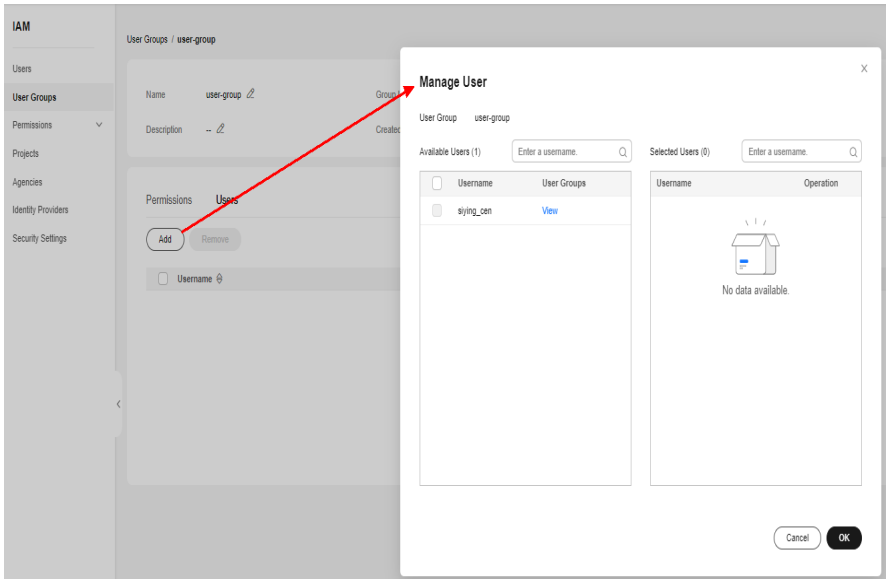


Figure 4-14 Selecting the created custom policy



6. Click the name of the user group and add the required users.

Figure 4-15 Adding users



Prerequisites

There is an available GeminiDB Redis instance.

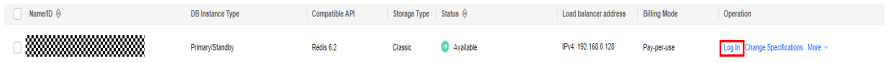
Procedure

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

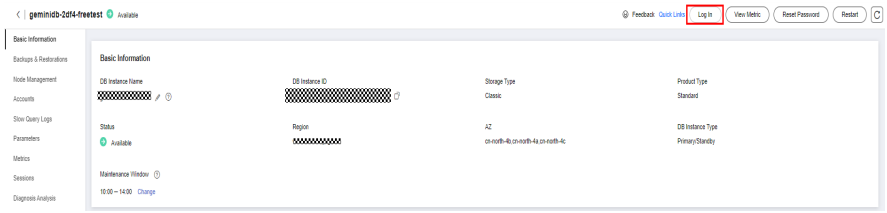
Step 3 In the instance list, locate the target instance and click **Log In** in the **Operation** column.

Figure 4-16 Logging in to a GeminiDB Redis instance



Alternatively, click the instance name to go to the **Basic Information** page. Click **Log In** in the upper right corner of the page.

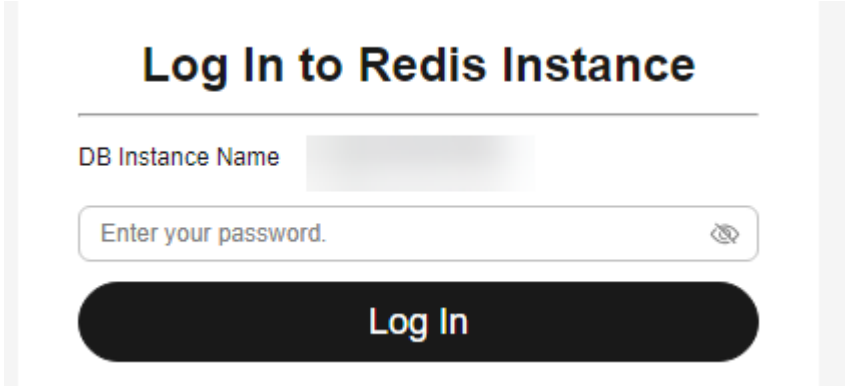
Figure 4-17 Logging in to a GeminiDB Redis instance



Step 4 Enter the password for logging in to the instance.

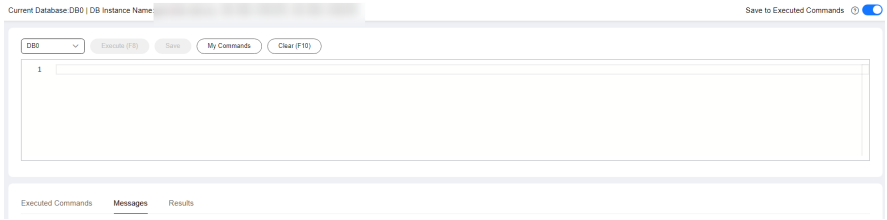
You need to enter the password only when you log in to a GeminiDB Redis instance for the first time or after you reset the password.

Figure 4-18 Login page



Step 5 Manage relevant databases.

Figure 4-19 Instance homepage



- Save to Executed Commands

  This function is enabled by default to save the recently executed commands for your later query.

  Then you can click the **Executed Commands** tab on the lower page to view historical commands.

  Figure 4-20 The Executed Commands tab

  If this function is disabled, the commands executed subsequently are not displayed any longer. You can click next to **Save Executed SQL Statements** in the upper right corner to disable this function.

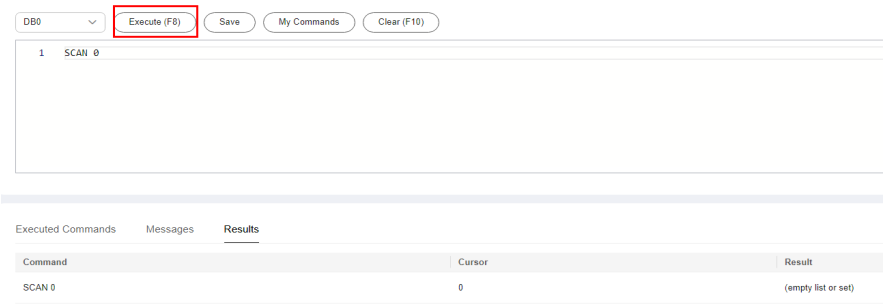
- Execute a command.

  You can enter a command in the command window and click **Execute** or **F8**.

  NOTE

  - Do not use transactions, Lua scripts, Pub/Sub commands, or other commands that have blocking semantics.
  - For an instance that supports multiple databases, you can change the current database on the console, but cannot change it using a SELECT statement.

  Figure 4-21 Executing a command

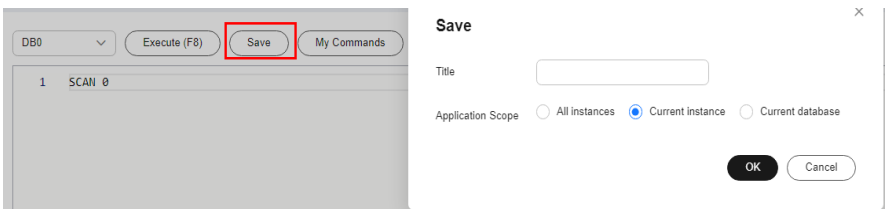
  

  After a command is executed, you can view the execution result on the **Results** page.

- Save

  You can save a command to all instances, the current instance, or the current database. Then you can view details in **My Commands**.

  Figure 4-22 Save

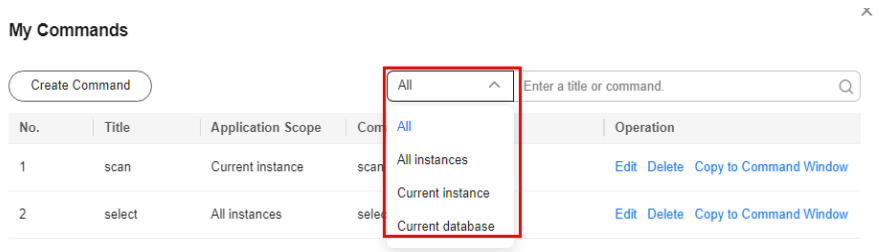
  

- My Commands

  Common commands are displayed the **My Commands** page.

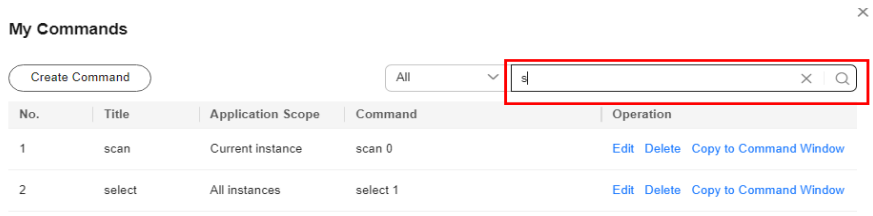
  Set a filter to narrow the scope of commands. If you select **All**, all commands saved in the current account are displayed.

  Figure 4-23 Filtering commands

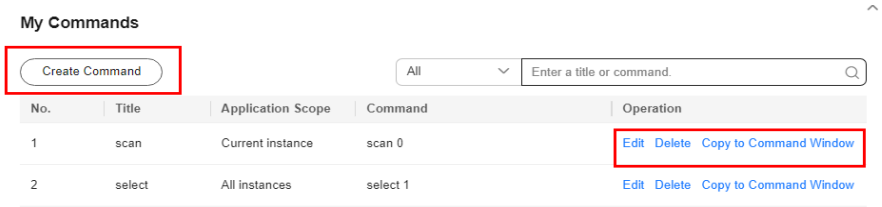
  Alternatively, enter a command title or statement in the search box to search for the corresponding command.

  Figure 4-24 Searching for a command

  On the **My Commands** page, you can also create, edit, and delete a command or copy it to the command window.

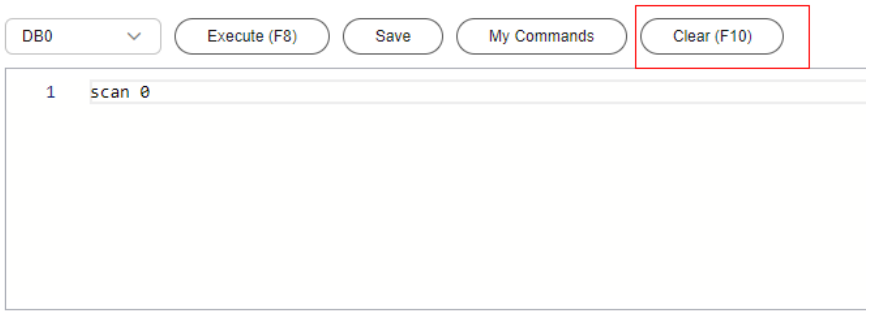
  Figure 4-25 Managing a command

- Clear

  You can also press **F10** to clear the command in the command window.

Figure 4-26 Clearing a command



----End

FAQs

Question: What should I do if the DAS console cannot be redirected after I click **Log In** in the **Operation** column in the instance list or click **Log In** on the **Basic Information** page?

Solution: Set your browser to allow pop-ups and try again.

4.3.3 Connecting to a GeminiDB Redis Instance Over a Private Network

4.3.3.1 Connecting to an Instance Using a Load Balancer Address (Recommended)

This section describes how to connect to a GeminiDB Redis instance using a load balancer address on a Linux ECS. Load balancing can improve data reliability and eliminate SPOFs.

Usage Notes

- The target instance must be in the same VPC and subnet as the ECS.
- The ECS must be in a security group that has access to the instances.

  Scenario 1: If the instance is associated with the default security group, you do not need to configure security group rules.

  Scenario 2: If the instance is not associated with the default security group, check whether the security group rules allow the ECS to connect to the instance.

  - If yes, the ECS can connect to the instance.
  - If no, add an inbound rule to the security group.

    For details about how to configure a security group, see **4.3.5.3 Configuring Security Group Rules for a GeminiDB Redis Instance**.

Prerequisites

- An ECS has been created. The following uses a Linux ECS as an example. For details, see **Purchasing an ECS** in *Getting Started with Elastic Cloud Server*.

- Download the **Redis client installation package**.

Procedure

Step 1 Log in to the ECS. For details, see **Logging In to an ECS** in *Getting Started with Elastic Cloud Server*.

Step 2 Obtain the Redis client.

Method 1

Run the following command to download the Redis client.

wget http://download.redis.io/releases/redis-6.2.0.tar.gz

Method 2

Download the Redis client from the address provided in **Prerequisites** and upload the Redis client installation package to the ECS.

Step 3 Decompress the client package.

tar -xzf redis-6.2.0.tar.gz

Step 4 Open the **src** directory and connect to the DB instance.

cd redis-6.2.0

make

cd src

./redis-cli -h <*DB_HOST*> **-p** <*DB_PORT*> **-a** <*DB_PWD*>

Example:

./redis-cli -h 192.xx.xx.xx -p 6379 -a <*DB_PWD*>

Table 4-19 Parameter description

Parameter	Description
<DB_HOST>	Load balancer IP address of the instance to be connected. After the load balancer IP address is created, click the instance name to go to the **Basic Information** page and obtain the load balancer IP address in the **Connection Information** area.
<DB_PORT>	Access port corresponding to the load balancer IP address of the instance. The procedure is as follows: Click the name of the instance to go to the **Basic Information** page. In the **Connection Information** area, you can find the access port in field **Database Port**.
<DB_PWD>	Administrator password set when you buy a GeminiDB Redis instance

Step 5 Check the results. If the following information is displayed, the connection is successful.

```
IP:port>
```

----End

4.3.3.2 Connecting to an Instance Using a Private Domain Name

This section describes how to connect to a ECS instance using a private domain name on a Linux ECS.

Usage Notes

- The target instance must be in the same VPC and subnet as the ECS.
- The ECS must be in a security group that has access to the instances.

  Scenario 1: If the instance is associated with the default security group, you do not need to configure security group rules.

  Scenario 2: If the instance is not associated with the default security group, check whether the security group rules allow the ECS to connect to the instance.

  – If yes, the ECS can connect to the instance.

  – If no, add an inbound rule to the security group.

    For details about how to configure a security group, see **4.3.5.3 Configuring Security Group Rules for a GeminiDB Redis Instance**.

Prerequisites

- An ECS has been created. The following uses a Linux ECS as an example. For details, see **Purchasing an ECS** in *Getting Started with Elastic Cloud Server*.
- Download the **Redis client installation package**.

Procedure

Configuring a Private Domain Name of the GeminiDB Redis Instance

Creating a Private Domain Name

Step 1 **Log in to the Huawei Cloud console.**

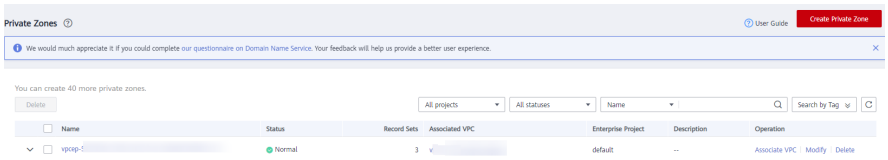
Step 2 Click **Service List**. Under **Network**, click **Domain Name Service**.

Step 3 On the displayed page, click **Private Zones**.

Figure 4-27 Private zones

Step 4 Click **Create Private Zone**.

Figure 4-28 Creating a private zone



Step 5 Set parameters as prompted.

Figure 4-29 Private zone parameters

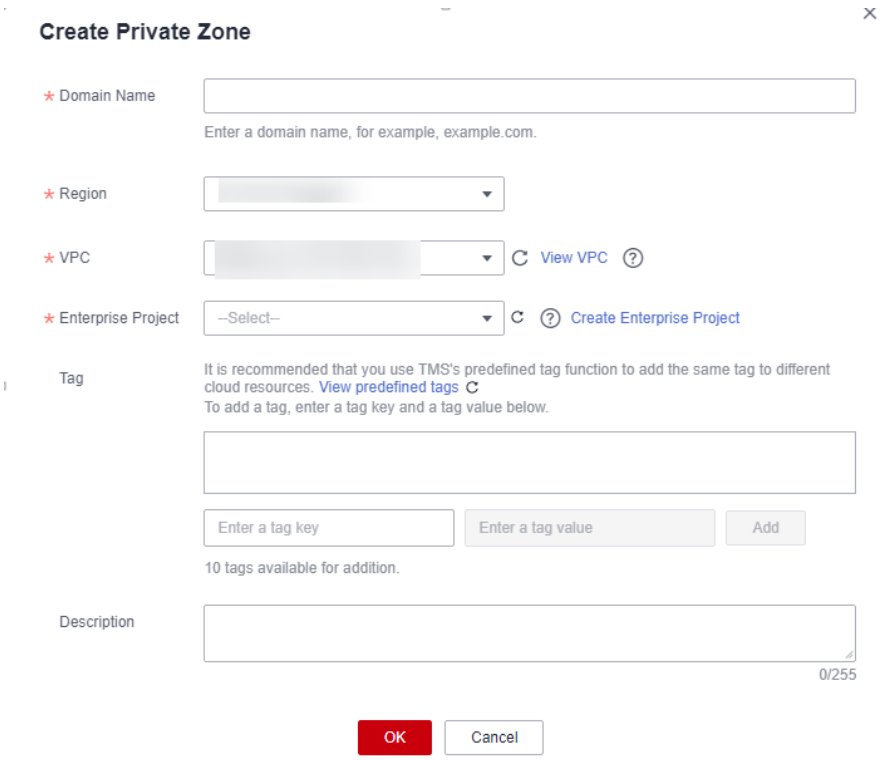


Table 4-20 Parameter description

Parameter	Description	Example Value
Domain Name	Domain name of a private zone You can enter a top-level domain that complies with the domain naming rules. For details about the domain name format, see **Domain Name Format and DNS Hierarchy**.	example.com
Region	Region where a tenant is located	CN East-Shanghai1

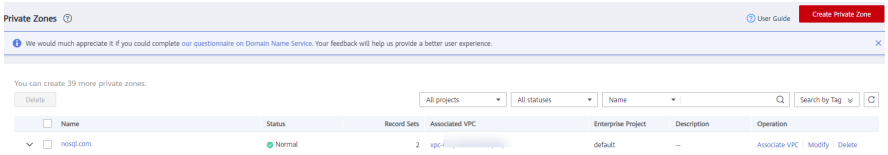
Parameter	Description	Example Value
VPC	The VPC associated with the private domain name must be the same as the VPC where the GeminiDB Redis instance is located. Otherwise, the private domain name cannot be resolved.	-
Enterprise Project	Enterprise project associated with the private domain name. You can manage private domain names by enterprise project. **NOTE** This parameter is available and mandatory only when **Account Type** is set to **Enterprise Account**. Configuration notes: ● If you do not manage domain names by enterprise project, select **default**. ● If you manage domain names by enterprise project, select an existing enterprise project.	default

Parameter	Description	Example Value
Tag	(Optional) Identifier of a resource. Each tag contains a key and a value. You can add a maximum of 20 tags to a domain name. Key and value naming rules: **Key**: • Cannot be left blank. • Must be unique for each resource. • Contains a maximum of 36 characters. • Cannot start or end with a space or contain special characters =*<> \,|/ **Value**: • Cannot be left blank. • Contains a maximum of 43 characters. • Cannot start or end with a space or contain special characters =*<> \,|/	example_key1 example_value1
Description	(Optional) Description of the zone, which cannot exceed 255 characters	This is a zone example.

Step 6 Click **OK**. On the **Private Zones** page, view the created private domain name in the zone list.

If the status of the private domain name is **Normal**, the domain name has been successfully created.

Figure 4-30 Private domain name status



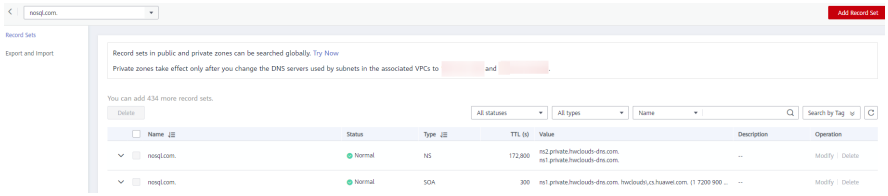
----End

Adding a Record Set for a Domain Name

After creating a private domain name, configure a record set for it so that you can access instances using the domain name.

Step 1 Click the private domain name you created. On the displayed page, click **Add Record Set** in the upper right corner.

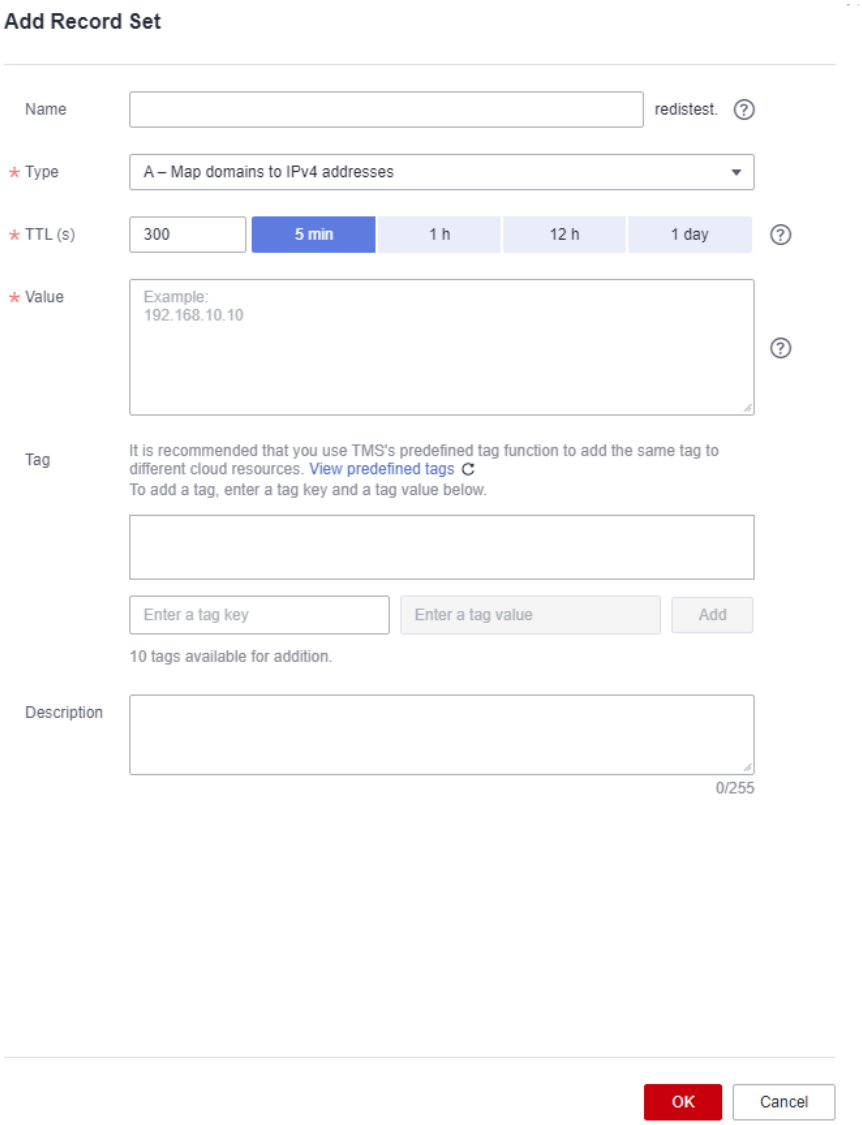
Figure 4-31 Adding a record set



Step 2 In the displayed **Add Record Set** dialog box, set parameters as prompted.

Value: Enter the load balancer IP address.

Figure 4-32 Adding a record set



For details about how to configure parameters, see **Adding an A Record Set**.

Step 3 Click **OK**.

Step 4 Switch back to the **Record Sets** page.

Step 5 View the created record set in the record set list. If the status of the record set is **Normal**, the record set is added successfully.

----End

Logging In to the ECS and Obtaining the Redis Client

Step 1 Log in to the ECS. For details, see **Logging In to an ECS** in *Getting Started with Elastic Cloud Server*.

Step 2 Obtain the Redis client.

Method 1

Run the following command to download the Redis client.

wget http://download.redis.io/releases/redis-6.2.0.tar.gz

Method 2

Download the Redis client from the address provided in **Prerequisites** and upload the Redis client installation package to the ECS.

Step 3 Decompress the client package.

tar -xzf redis-6.2.0.tar.gz

Step 4 Open the **src** directory and connect to the DB instance.

cd redis-6.2.0

make

cd src

./redis-cli -h <*DB_Domain_Name*> **-p** <*DB_PORT*> **-a** <*DB_PWD*>

Example:

./redis-cli -h redis.com -p 6379 -a *<DB_PWD>*

Table 4-21 Parameter description

Parameter	Description
<DB_Domain_Name>	Private domain name of the instance to be connected. The private domain name is the one created in **Configuring a Private Domain Name of the GeminiDB Redis Instance**.
<DB_PORT>	Port for accessing the target instance. Configure this parameter based on service requirements. To obtain the port number, perform the following steps: Click the target instance to go to the **Basic Information** page. In the **Network Information** area, you can find the database port.

Parameter	Description
<DB_PWD>	Administrator password set when you buy a GeminiDB Redis instance

Step 5 Check the results. If the following information is displayed, the connection is successful.

```
Domain_Name:port>
```

----End

4.3.3.3 Connecting to an Instance Using a Private IP Address

You can use the private IP address to connect to the GeminiDB Redis instance.

This section uses the Linux OS as an example to describe how to connect to a GeminiDB Redis instance using the Redis-cli client. You can connect to an instance through SSL to secure your data. For details, see **4.3.5.7 Connecting a GeminiDB Redis Instance over SSL** This section describes how to connect to a GeminiDB Redis instance in non-SSL mode.

To ensure data reliability, you are advised to use a **load balancer address** or **domain name** to access the instance.

Usage Notes

- The target instance must be in the same VPC and subnet as the ECS.
- The ECS must be in a security group that has access to the instances. For details, see **4.3.5.3 Configuring Security Group Rules for a GeminiDB Redis Instance**.
- To connect to a DB instance over a non-SSL connection, SSL must be disabled. For details about how to disable SSL, see **4.3.5.6 Encrypting Data over SSL for a GeminiDB Redis Instance**.

Prerequisites

An ECS has been created. The following uses a Linux ECS as an example. For details, see **Purchasing an ECS** in *Getting Started with Elastic Cloud Server*.

Procedure

Step 1 Log in to the ECS. For details, see **Logging In to an ECS** in *Getting Started with Elastic Cloud Server*.

Step 2 Obtain the Redis client.

Method 1

Run the following command to download the Redis client.

```
wget http://download.redis.io/releases/redis-6.2.0.tar.gz
```

Method 2

Download the **Redis client** and upload it to the ECS.

Step 3 Decompress the client package.

```
tar -xzf redis-6.2.0.tar.gz
```

Step 4 Open the **src** directory and connect to the DB instance.

```
cd redis-6.2.0
make
cd src
./redis-cli -h <DB_HOST> -p <DB_PORT> -a <DB_PWD>
```

Example:

```
./redis-cli -h 192.xx.xx.xx -p 6379 -a <DB_PWD>
```

Table 4-22 Parameter description

Parameter	Description
<DB_HOST>	Private IP address of an instance to be connected. To obtain this IP address, go to the **Instance Management** page and click the target DB instance name. The IP address can be found in the **Private IP Address** field under **Node Information** on the **Basic Information** page. If the instance you purchased has multiple nodes, select the private IP address of any node.
<DB_PORT>	Port for accessing the target instance. Configure this parameter based on service requirements. To obtain the port number, perform the following steps: Click the target instance to go to the **Basic Information** page. In the **Network Information** area, you can find the database port.
<DB_PWD>	Administrator password set when you buy a GeminiDB Redis instance

Step 5 Check the results. If the following information is displayed, the connection is successful.

```
IP:port>
```

----End

4.3.4 Connecting to a GeminiDB Redis Instance Over a Public Network

4.3.4.1 Connecting to an Instance Using an EIP Bound to a Load Balancer (Recommended)

This section describes how to access a GeminiDB Redis instance over a public network by creating a load balancer and binding it to an EIP. To prevent single points of failure (SPOFs) in the production environment and implement load

balancing, you are advised to connect to the GeminiDB Redis instance using an EIP bound to a load balancer.

To connect to a GeminiDB Redis instance over a public network, use a public domain name to ensure instance reliability. For details, see **4.3.4.3 Connecting to an Instance Using a Public Domain Name**.

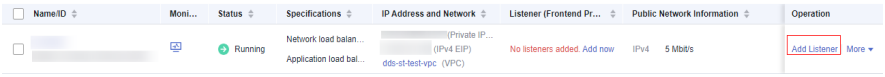
Creating and Configuring a Dedicated Load Balancer

Step 1 Purchase a **dedicated load balancer**. For details, see **Creating a Dedicated Load Balancer**. Pay attention to the following:

- When creating a flavor, you need to select the TCP/UDP network.
- In the network configuration, **Cross-VPC Backend** must be enabled so that backend IP addresses can be added to the load balancer.
- You need to use a new or existing EIP to support public network access.

Step 2 Add a listener. For details, see **Adding a TCP Listener**. Pay attention to the following:

Figure 4-33 Adding a listener



- When configuring a listener, select the TCP protocol to and the **6379** port, which is commonly used by Redis.
- When adding a backend server, click the **Cross-VPC Backend Servers** tab and then click **Add Cross-VPC Backend Server**. Configure the load balancer address and port number of the GeminiDB Redis instance in the cross-VPC backend IP address.
- Enable the health check.

Step 3 Create a VPC peering connection and select the local VPC and peer VPC.

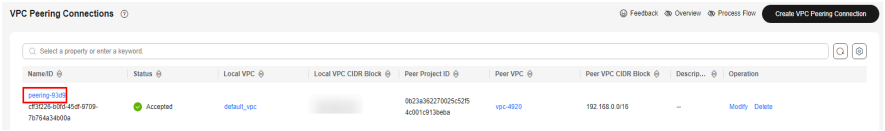
Local VPC: specifies the VPC to which the ELB belongs.

- If the selected VPC and GeminiDB Redis are in the same VPC, set the peer VPC to a VPC where no ELB is located.
- If the selected VPC and GeminiDB Redis are not in the same VPC, set the peer VPC to the VPC where the GeminiDB Redis instance resides.

For details, see **Creating a VPC Peering Connection to Connect Two VPCs in the Same Account**.

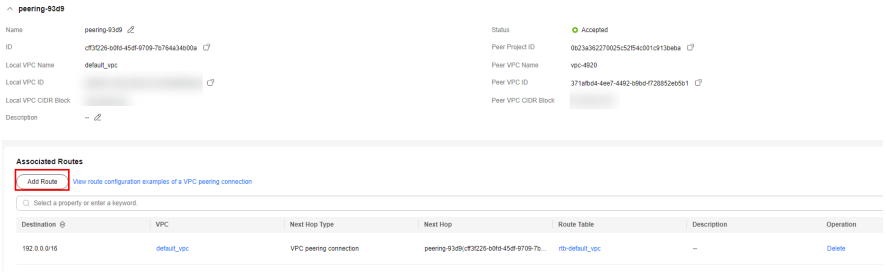
Step 4 Click the name of the VPC peering connection to go to its details page.

Figure 4-34 VPC peering connection



Step 5 Click **Route Tables**.

Figure 4-35 Route table



Step 6 Configure local and peer routes for the VPC peering connection.

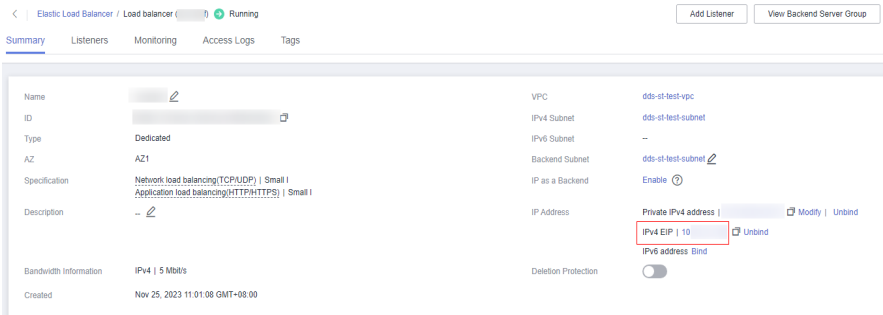
1. Local route: Click **Add Route**. In the displayed **Add Route** dialog box, set **Destination** to the value of **Peer VPC CIDR Block** of the VPC peering connection, set **Next Hop Type** to **VPC peering connection**, set **Next Hop** to the VPC peering connection created in **Step 3**, and click **OK**.
2. Peer route: Click **Add Route**. In the displayed **Add Route** dialog box, set **Destination** to the value of **Local VPC CIDR Block** of the VPC peering connection, set **Next Hop Type** to **VPC peering connection**, set **Next Hop** to the VPC peering connection created in **Step 3**, and click **OK**.

Step 7 Perform a health check on the load balancer address of GeminiDB Redis. Click the **Listeners** tab. If the health check result is **Healthy**, the address is available.

----End

After you create a dedicated load balancer, you can access the GeminiDB Redis instance using the service address displayed in the **Basic Information** area.

Figure 4-36 Service address



Procedure

Step 1 Log in to the ECS. For details, see **Logging In to an ECS** in *Getting Started with Elastic Cloud Server*.

Step 2 Obtain the Redis client.

Method 1

Run the following command to download the Redis client.


```
wget http://download.redis.io/releases/redis-6.2.0.tar.gz
```

Method 2

Download the **Redis client** and upload it to the ECS.

Step 3 Decompress the client package.

```
tar -xzf redis-6.2.0.tar.gz
```

Step 4 Open the **src** directory and connect to the DB instance.

```
cd redis-6.2.0
make
cd src
./redis-cli -h <DB_HOST> -p <DB_PORT> -a <DB_PWD>
```

Example:

```
./redis-cli -h 192.168.0.208 -p 6379 -a <DB_PWD>
```

Table 4-23 Parameter description

Parameter	Description
<DB_HOST>	EIP bound to the instance to be connected. To obtain the EIP, go to the **Instance Management** page and click the target instance name. The EIP can be found in the **EIP** column in the **Node Information** area on the **Basic Information** page. If the instance you bought has multiple nodes, you can bind the EIP to any node to connect to the instance. If a message is displayed indicating that no EIP has been bound to the instance, bind an EIP to the instance by following **4.3.5.5 Binding an EIP to a GeminiDB Redis Instance** .
<DB_PORT>	Port for accessing the target instance. Configure this parameter based on service requirements. To obtain the port number, perform the following steps: Click the target instance to go to the **Basic Information** page. In the **Network Information** area, you can find the database port.
<DB_PWD>	Administrator password set when you buy a GeminiDB Redis instance

Step 5 Check the results. If information similar to the following is displayed, the connection is successful.

```
IP:port>
```

----End

4.3.4.2 Connecting to an Instance Using an EIP

You can connect to a GeminiDB Redis instance from an ECS or a local device over a public network.

This section uses the Linux OS as an example to describe how to connect to a GeminiDB Redis instance using the Redis-cli client. You can connect to a GeminiDB Redis instance using an EIP bound to a load balancer to avoid SPOFs and achieve load balancing in the production environment.

You can connect to an instance over SSL or non-SSL connections. SSL encrypts data and is more secure. For details, see **4.3.5.7 Connecting a GeminiDB Redis Instance over SSL**. This section describes how to connect to a GeminiDB Redis instance over a non-SSL connection.

Usage Notes

- To connect to a DB instance over a non-SSL connection, SSL must be disabled. For details about how to disable SSL, see **4.3.5.6 Encrypting Data over SSL for a GeminiDB Redis Instance**.
- You need to estimate the bandwidth required by services and purchase an EIP with sufficient bandwidth resources. **Client access exceptions caused by poor public network performance will not be included in the SLA.**

Prerequisites

1. An ECS has been created. The following uses a Linux ECS as an example. For details, see **Purchasing an ECS** in *Getting Started with Elastic Cloud Server*.
2. You have bound an EIP to a node of the purchased instance and configure security group rules for the node. For details, see **4.3.5.5 Binding an EIP to a GeminiDB Redis Instance** and **4.3.5.3 Configuring Security Group Rules for a GeminiDB Redis Instance**.

   NOTE

   A GeminiDB Redis instance can have multiple nodes. Select any node and bind an EIP to it.

Procedure

Step 1 Log in to the ECS. For details, see **Logging In to an ECS** in *Getting Started with Elastic Cloud Server*.

Step 2 Obtain the Redis client.

Method 1

Run the following command to download the Redis client.

```
wget http://download.redis.io/releases/redis-6.2.0.tar.gz
```

Method 2

Download the **Redis client** and upload it to the ECS.

Step 3 Decompress the client package.

```
tar -xzf redis-6.2.0.tar.gz
```

Step 4 Open the **src** directory and connect to the DB instance.

```
cd redis-6.2.0
make
cd src
./redis-cli -h <DB_HOST> -p <DB_PORT> -a <DB_PWD>
```


Example:

```
./redis-cli -h 192.168.0.208 -p 6379 -a <DB_PWD>
```

Table 4-24 Parameter description

Parameter	Description
<DB_HOST>	EIP bound to the instance to be connected. To obtain the EIP, go to the **Instance Management** page and click the target instance name. The EIP can be found in the **EIP** column in the **Node Information** area on the **Basic Information** page. If the instance you bought has multiple nodes, you can bind the EIP to any node to connect to the instance. If a message is displayed indicating that no EIP has been bound to the instance, bind an EIP to the instance by following **4.3.5.5 Binding an EIP to a GeminiDB Redis Instance** .
<DB_PORT>	Port for accessing the target instance. Configure this parameter based on service requirements. To obtain the port number, perform the following steps: Click the target instance to go to the **Basic Information** page. In the **Network Information** area, you can find the database port.
<DB_PWD>	Administrator password set when you buy a GeminiDB Redis instance

Step 5 Check the results. If the following information is displayed, the connection is successful.

```
IP:port>
```

----End

4.3.4.3 Connecting to an Instance Using a Public Domain Name

A public domain name is a domain name used to access websites or web applications on the Internet.

You can use Domain Name Service (DNS) to translate common domain names (for example, www.example.com) into IP addresses (for example, 1.2.3.4) required for network connection. In this way, you can access GeminiDB Redis instances using the resolved IP addresses.

This section uses the Linux OS as an example to describe how to use the public network domain name configured by the DNS service to connect to a GeminiDB Redis instance.

Prerequisites

- An ECS has been created. The following uses a Linux ECS as an example. For details, see **Purchasing an ECS** in *Getting Started with Elastic Cloud Server*.

- You have registered a domain name and an EIP.
- You have bound an EIP to a node of the purchased instance and configure security group rules for the node. For details, see **4.3.5.5 Binding an EIP to a GeminiDB Redis Instance** and **4.3.5.3 Configuring Security Group Rules for a GeminiDB Redis Instance**.

  NOTE

  A GeminiDB Redis instance can have multiple nodes. Select any node and bind an EIP to it.
- Download the **Redis client installation package**.

Procedure

Configuring a Public Domain Name of the GeminiDB Redis Instance

Domain Name Not Created on Huawei Cloud

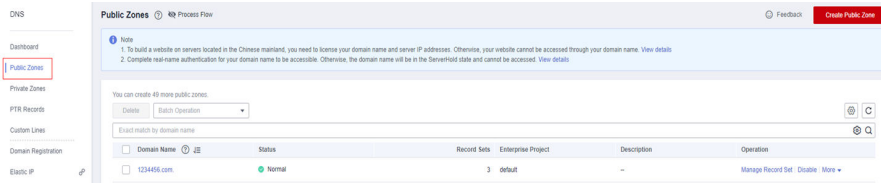
If a third-party domain name is used, create a public zone and add record sets to it on the DNS console.

Step 1 **Log in to the Huawei Cloud console.**

Step 2 Click **Service List** and choose **Network** > **Domain Name Service**.

Step 3 In the navigation pane, choose **Public Zones**.

Figure 4-37 Public zones



Step 4 In the upper right corner of the page, click **Create Public Zone**.

Step 5 Set the parameters as prompted.

Figure 4-38 Creating a public zone

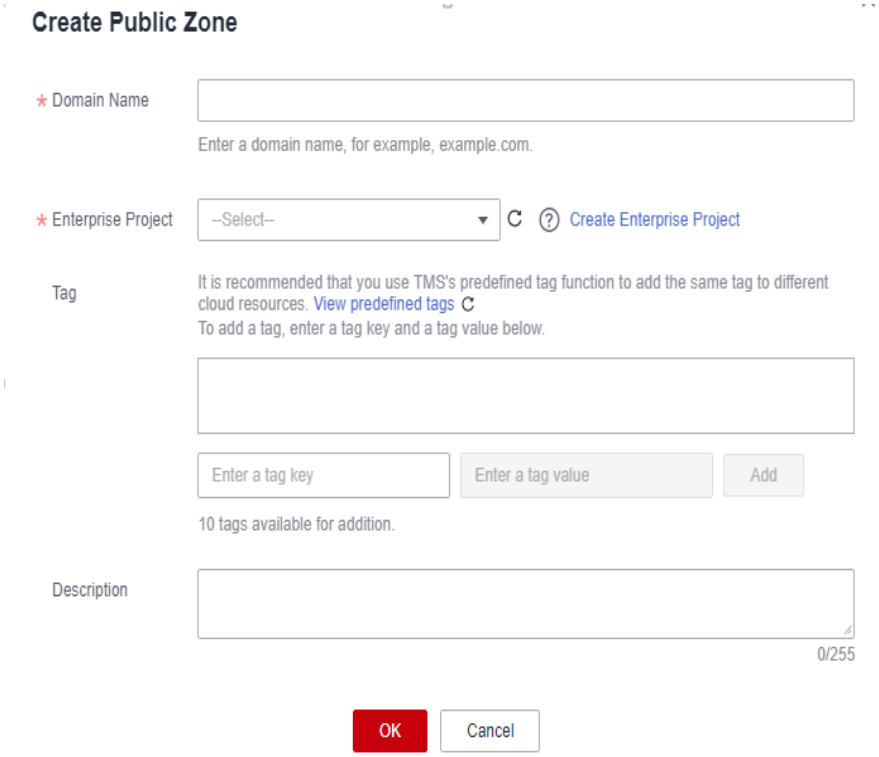


Table 4-25 Public zone parameters

Parameter	Description	Example Value
Domain Name	Domain name you have registered. It can include two levels in addition to the top-level domain, for example: • **abc.example.com**, the subdomain name of **example.com** • **abc.example.com.cn**, the subdomain name of **example.com.cn** For details about the domain name format, see **Domain Name Format and DNS Hierarchy**.	example.com

Parameter	Description	Example Value
Enterprise Project	Enterprise project associated with the public domain name. You can manage public domain names by enterprise project. **NOTE** This parameter is available and mandatory only when **Account Type** is set to **Enterprise Account**. Configuration notes: ● If you do not manage domain names by enterprise project, select **default**. ● If you manage domain names by enterprise project, select an existing enterprise project.	default
Tag	(Optional) Identifier of a resource. Each tag contains a key and a value. You can add a maximum of 20 tags to a domain name. Key and value naming rules: **Key**: ● Cannot be left blank. ● Must be unique for each resource. ● Contains a maximum of 36 characters. ● Cannot start or end with a space or contain special characters =*<>\,|/ **Value**: ● Cannot be left blank. ● Contains a maximum of 43 characters. ● Cannot start or end with a space or contain special characters =*<>\,|/	example_key1 example_value1
Description	(Optional) Description of the zone, which cannot exceed 255 characters	This is a zone example.

Step 6 Click **OK**.

After the domain name is created, you can view it in the domain name list on the **Public Zones** page.

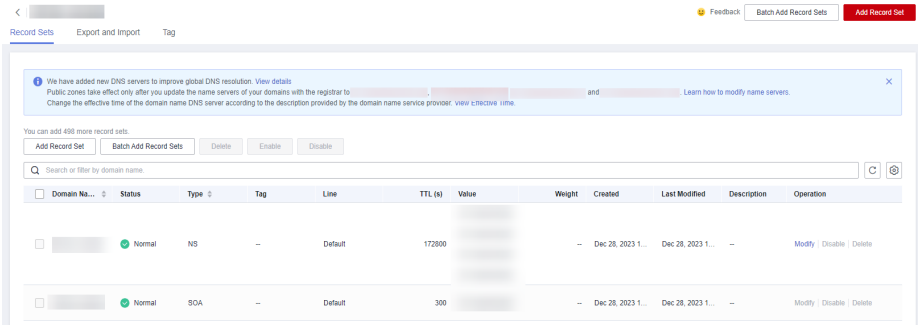
----End

Adding a Record Set for a Domain Name

After creating a public domain name, configure a record set for it so that you can access instances using the domain name.

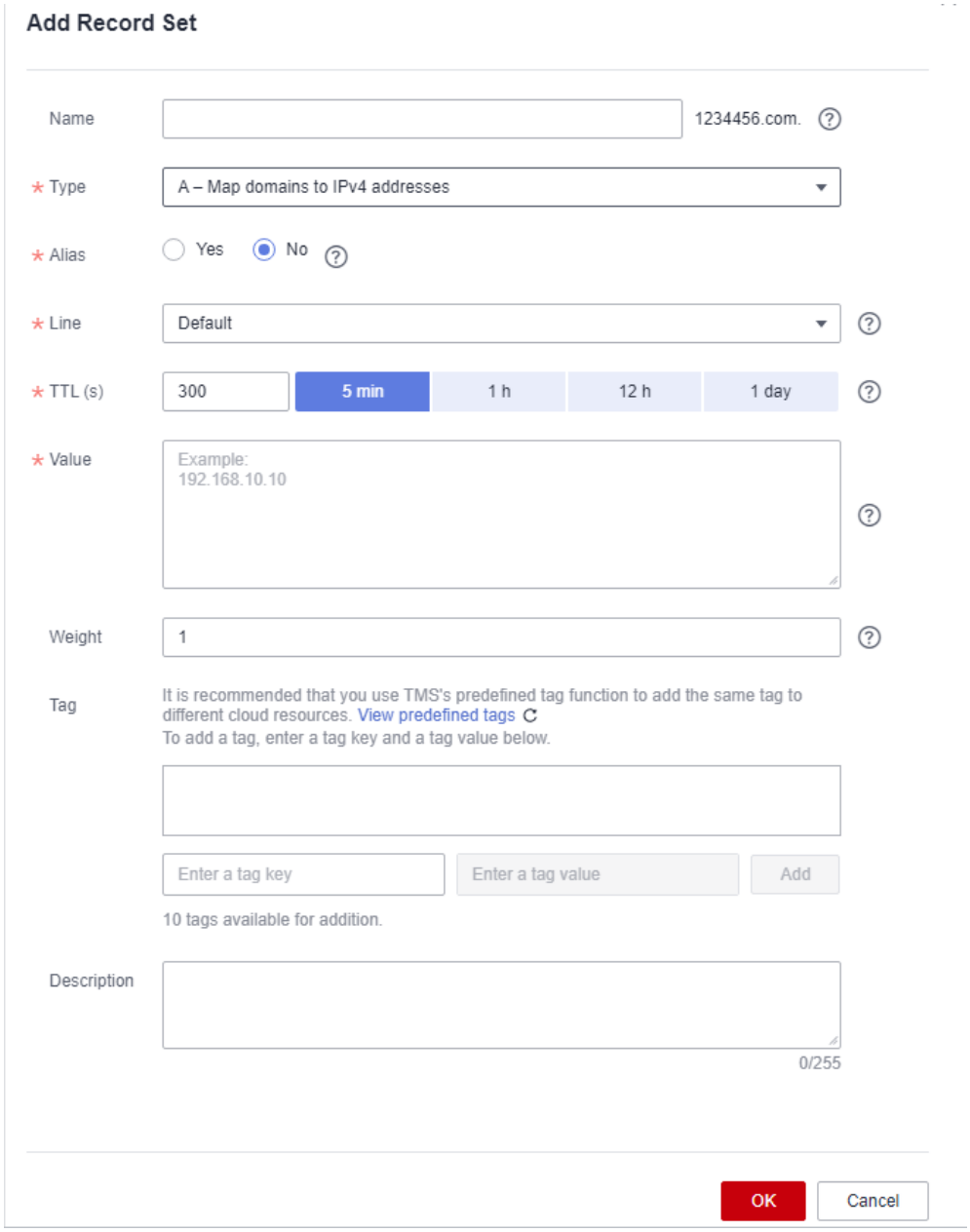
Step 1 Click the name of the public domain name you created. On the displayed page, click **Add Record Set** in the upper right corner.

Figure 4-39 Adding a record set



Step 2 In the displayed **Add Record Set** dialog box, set parameters as prompted.

Figure 4-40 Adding a record set



For details about how to configure parameters, see **Adding an A Record Set**.

Step 3 Click **OK**.

Step 4 Switch back to the **Record Sets** page.

Step 5 View the created record set in the record set list. If the status of the record set is **Normal**, the record set is added successfully.

----End

Logging In to the ECS and Obtaining the Redis Client

Step 1 Log in to the ECS. For details, see **Logging In to an ECS** in *Getting Started with Elastic Cloud Server*.

Step 2 Obtain the Redis client.

Method 1

Run the following command to download the Redis client.

wget http://download.redis.io/releases/redis-6.2.0.tar.gz

Method 2

Download the Redis client from the address provided in **Prerequisites** and upload the Redis client installation package to the ECS.

Step 3 Decompress the client package.

tar -xzf redis-6.2.0.tar.gz

Step 4 Connect to the instance in the **src** directory.

cd redis-6.2.0

make

cd src

./redis-cli -h <*DB_Domain_Name*> **-p** <*DB_PORT*> **-a** <*DB_PWD*>

Example:

./redis-cli -h redis.com -p 6379 -a *<DB_PWD>*

Table 4-26 Parameter description

Parameter	Description
<DB_Domain_Name>	Public domain name of the instance to be connected. The public domain name is the one created in **Configuring a Public Domain Name of the GeminiDB Redis Instance**.
<DB_PORT>	Port for accessing the target instance. Configure this parameter based on service requirements. To obtain the port number, perform the following steps: Click the target instance to go to the **Basic Information** page. In the **Network Information** area, you can find the database port.
<DB_PWD>	Administrator password set when you buy a GeminiDB Redis instance

Step 5 Check the results. If the following information is displayed, the connection is successful.

```
Domain_Name:port>
```

----End

4.3.5 Connection Information Management

4.3.5.1 Configuring a Private Domain Name for a GeminiDB Redis Instance

This section describes how to configure and resolve private domain names.

Creating a Private Domain Name

Step 1 **Log in to the GeminiDB console.**

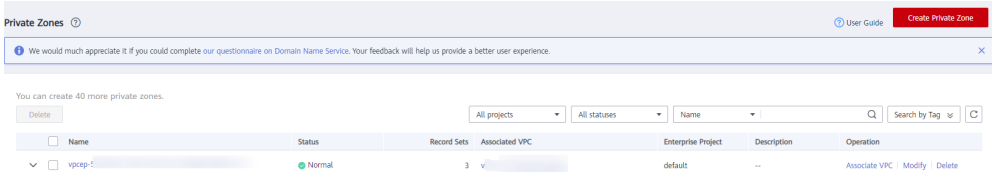
Step 2 Click **Service List**. Under **Network**, click **Domain Name Service**.

Step 3 On the DNS console, choose **Private Zones**.

Figure 4-41 Private zones

Step 4 Click **Create Private Zone**.

Figure 4-42 Creating a private zone



Step 5 Set parameters as prompted.

Figure 4-43 Creating a private zone

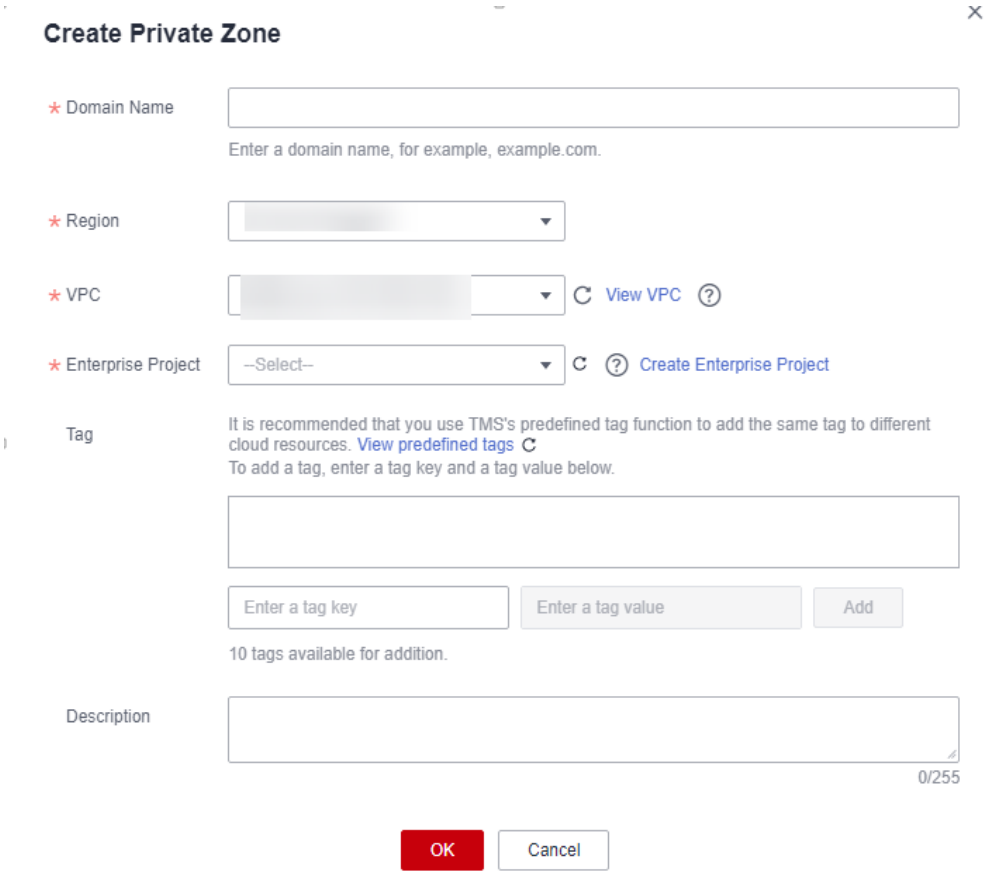


Table 4-27 Parameter description

Parameter	Description	Example
Domain Name	Domain name of the private zone. You can customize any correctly formatted domain names, even top-level ones. For details about the domain name format, see **Domain Name Format and DNS Hierarchy**.	example.com
Region	Region where a tenant is located	CN East-Shanghai1

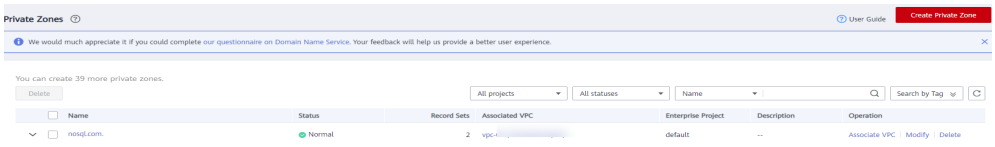
Parameter	Description	Example
VPC	The VPC associated with the private domain name must be the same as the VPC where the GeminiDB Redis instance is located. Otherwise, the private domain name cannot be resolved.	-
Enterprise Project	Enterprise project associated with the private domain name. You can manage private domain names by enterprise project. **NOTE** This parameter is available and mandatory only when **Account Type** is set to **Enterprise Account**. Configuration principles: ● If you do not manage domain names by enterprise project, select the **default** enterprise project. ● If you manage domain names by enterprise project, select an existing enterprise project.	default

Parameter	Description	Example
Tags	(Optional) Identifier of a resource. Each tag contains a key and a value. You can add a maximum of 20 tags to a domain name. The key and value naming rules are as follows: **Key**: • Cannot be left blank. • Must be unique for each resource. • Consists of a maximum of 36 characters. • Cannot start or end with a space or contain special characters =*<>\,|/ **Value**: • Cannot be left blank. • Consists of a maximum of 43 characters. • Cannot start or end with a space or contain special characters =*<>\,|/	example_key1 example_value1
Description	(Optional) Description of the zone, which cannot exceed 255 characters.	This is a zone example.

Step 6 Click **OK**. On the **Private Zones** page, view the created private domain name in the zone list.

If the status of the private domain name is **Normal**, the domain name has been successfully created.

Figure 4-44 Viewing the private domain name status



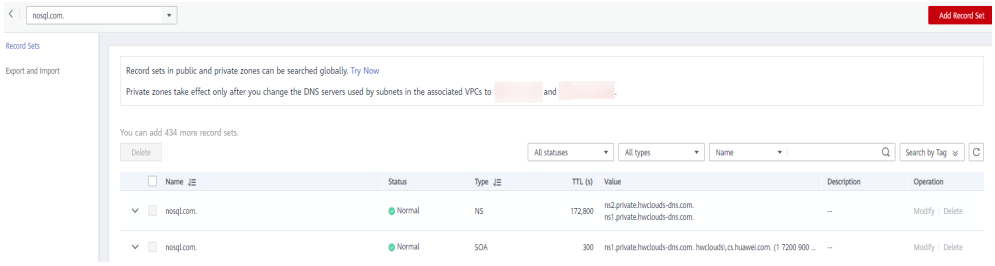
----End

Adding a Record Set for a Domain Name

After creating a private domain name, configure a record set for it so that you can access instances using the domain name.

Step 1 Click the private domain name you created. On the displayed page, click **Add Record Set** in the upper right corner.

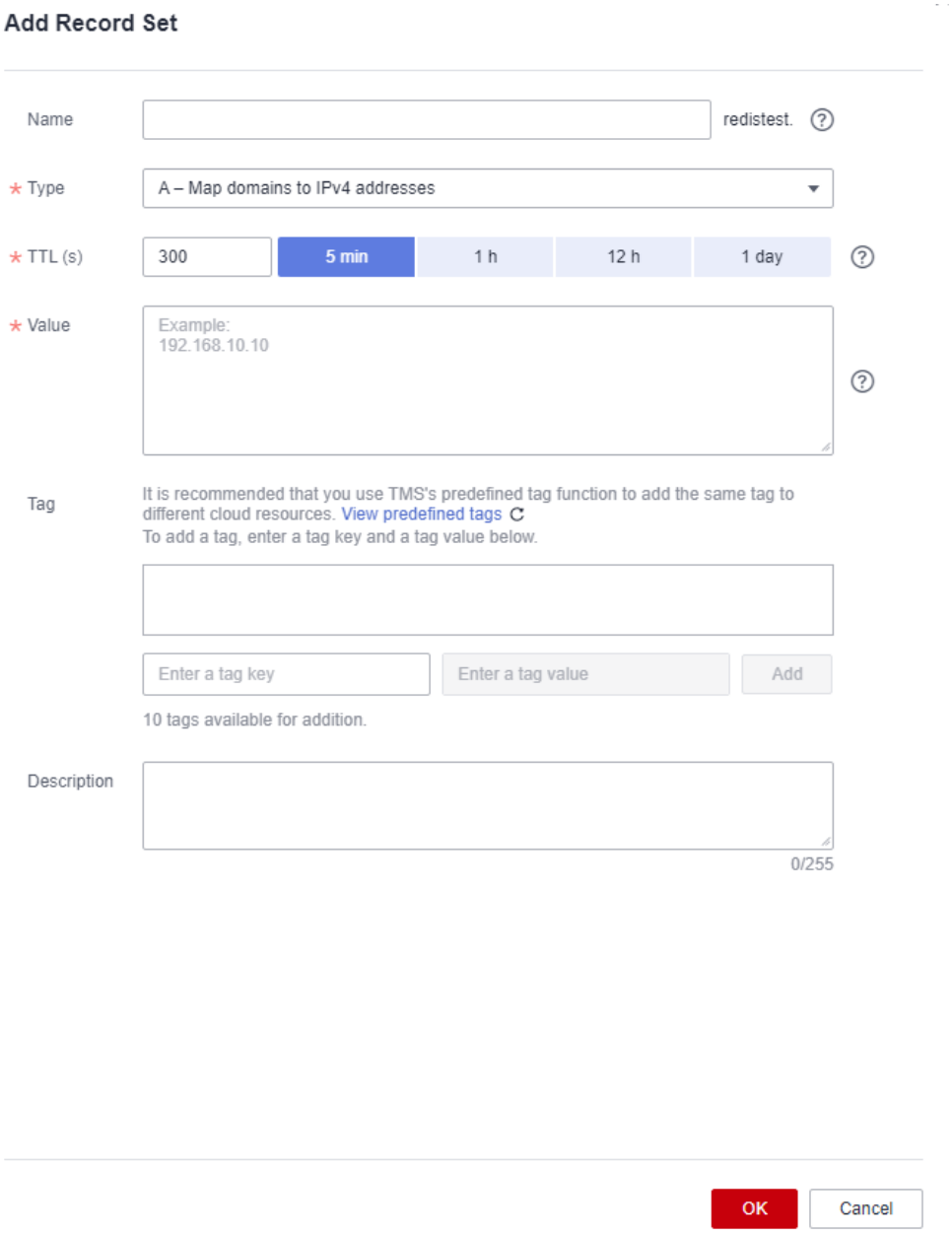
Figure 4-45 Adding a record set



Step 2 In the displayed **Add Record Set** dialog box, configure the required parameters.

Value: Enter the load balancer IP address of the instance.

Figure 4-46 Adding a record set



For details about how to configure parameters, see **Adding an A Record Set**.

Step 3 Click **OK**.

Step 4 Switch back to the **Record Sets** page.

Step 5 View the created record set in the record set list. If the status of the record set is **Normal**, the record set is added successfully.

----End

4.3.5.2 Configuring a Public Domain Name for a GeminiDB Redis Instance

This section describes how to configure and resolve a public domain name.

Procedure

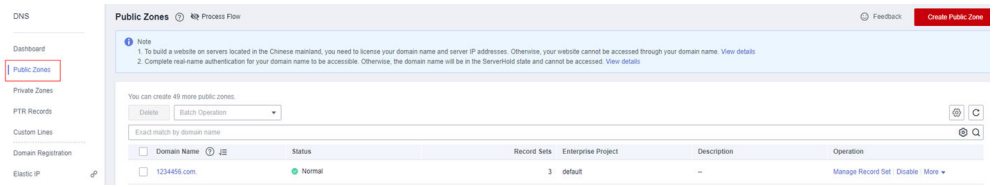
If your domain name is registered with a third-party registrar, create a public zone and add record sets to it on the DNS console.

Step 1 **Log in to the Huawei Cloud console.**

Step 2 Click **Service List** and choose **Network** > **Domain Name Service**.

Step 3 In the navigation pane on the left, choose **Public Zones**.

Figure 4-47 Public zones



Step 4 In the upper right corner of the page, click **Create Public Zone**.

Step 5 Set the required parameters.

Figure 4-48 Creating a public zone

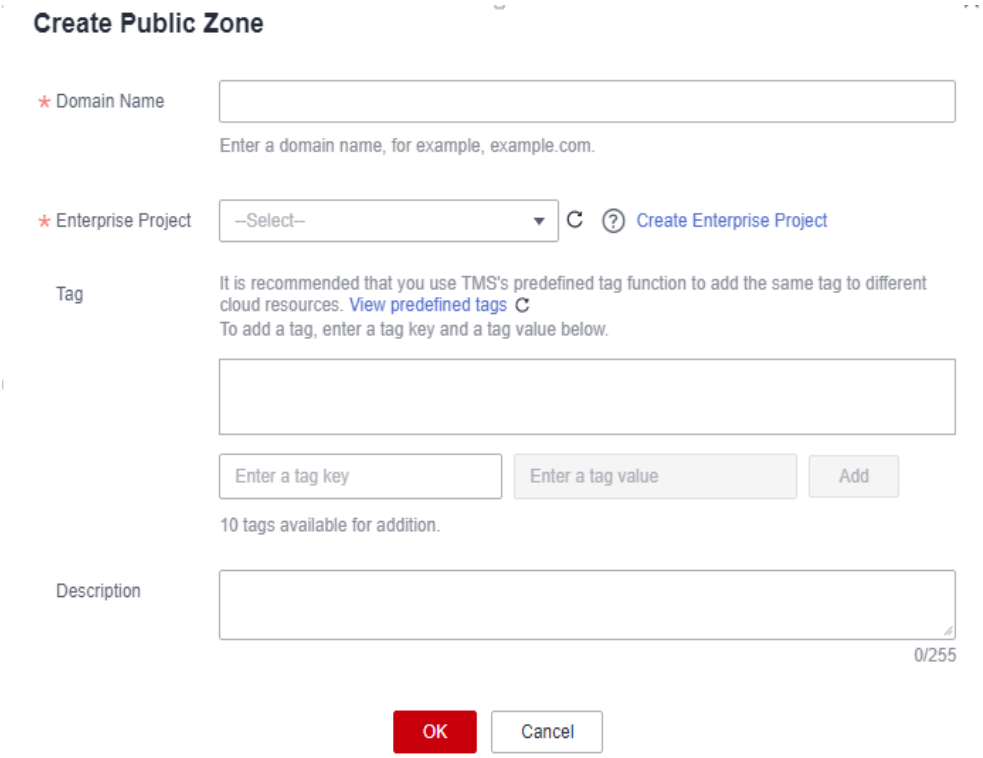


Table 4-28 Public zone parameters

Parameter	Description	Example
Domain Name	The domain name registered with the domain name registrar. The domain name can include two levels in addition to the top-level domain, for example: • abc.example.com, the subdomain name of example.com • abc.example.com.cn, the subdomain name of example.com.cn For details about the domain name format, see **Domain Name Format and DNS Hierarchy**.	example.com
Enterprise Project	Enterprise project associated with the public domain name. You can manage public domain names by enterprise project. **NOTE** This parameter is available and mandatory only when **Account Type** is set to **Enterprise Account**. Configuration principles: • If you do not manage domain names by enterprise project, select the **default** enterprise project. • If you manage domain names by enterprise project, select an existing enterprise project.	default

Parameter	Description	Example
Tag	(Optional) Identifier of a resource. Each tag contains a key and a value. You can add a maximum of 20 tags to a domain name. The key and value naming rules are as follows: **Key**: • Cannot be left blank. • Must be unique for each resource. • Consists of a maximum of 36 characters. • Cannot start or end with a space or contain special characters =*<>\,|/ **Value**: • Cannot be left blank. • Consists of a maximum of 43 characters. • Cannot start or end with a space or contain special characters =*<>\,|/	example_key1 example_value1
Description	(Optional) Description of the zone, which cannot exceed 255 characters.	This is a zone example.

Step 6 Click **OK**.

After the domain name is created, you can view it in the domain name list on the **Public Zones** page.

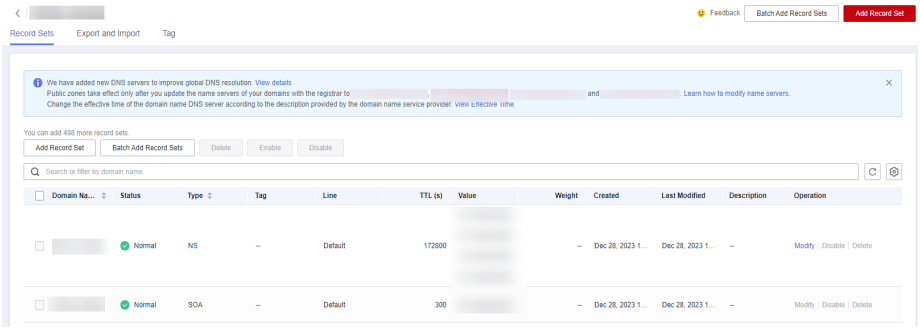
----End

Adding a Record Set for a Domain Name

After creating a public domain name, configure a record set for it so that you can access instances using the domain name.

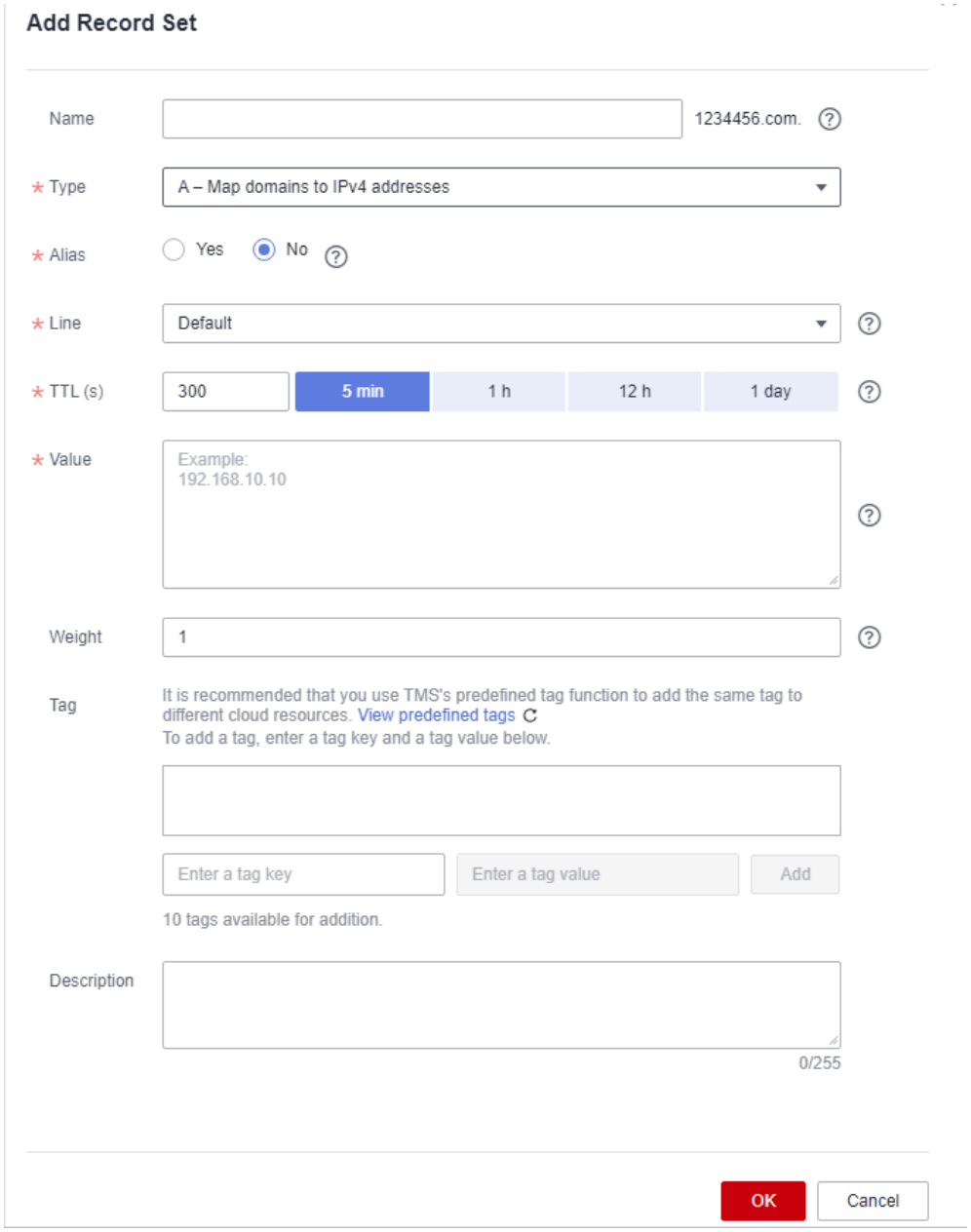
Step 1 Click the name of the public domain name you created. On the displayed page, click **Add Record Set** in the upper right corner.

Figure 4-49 Adding a record set



Step 2 In the displayed **Add Record Set** dialog box, configure the required parameters.

Figure 4-50 Adding a record set



For details about how to configure parameters, see **Adding an A Record Set**.

Step 3 Click **OK**.

Step 4 Switch back to the **Record Sets** page.

Step 5 View the created record set in the record set list. If the status of the record set is **Normal**, the record set is added successfully.

----End

4.3.5.3 Configuring Security Group Rules for a GeminiDB Redis Instance

A security group is a collection of access control rules for ECSs and GeminiDB Redis instances that have the same security protection requirements and are mutually trusted in a VPC.

To ensure database security and reliability, configure security group rules to allow specific IP addresses and ports to access the GeminiDB Redis instances.

This section describes how to configure security group rules for a GeminiDB Redis instance that is connected through a private or a public network.

Precautions

- Each account can create up to 500 security group rules by default.
- Too many security group rules will increase the first packet latency, so a maximum of 50 rules for each security group is recommended.
- One security group can be associated with only one GeminiDB Redis instance.
- For details about how to configure security group rules, see **Table 4-29**.

Table 4-29 Parameter description

Scenario	Description
Connecting to an instance over a private network	Configure security group rules as follows: • If a GeminiDB Redis instance and the ECS used for accessing the instance are in the same security group, they can communicate with each other by default. No security group rules need to be configured. • If the instance and the ECS are not in the same security group, configure security group rules, respectively. – Configure inbound rules for the security group associated with the GeminiDB Redis instance. For details, see **Procedure**. – There is no need to configure security rules for the ECS because the default security group rule of the ECS allows all outbound data packets. If not all outbound traffic is allowed in the security group, configure an outbound rule for the ECS. For details, see **Configuring a Security Group Rule**.
Connecting to an instance over a public network	If you connect to a GeminiDB Redis instance through a public network, configure inbound rules for the security group associated with the GeminiDB Redis instance. For details, see **Procedure**.

Procedure

Step 1 **Log in to the Huawei Cloud console.**

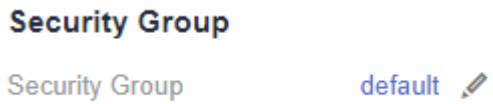
Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the instance that you want to configure security group rules for and click its name.

Step 4 Configure security group rules.

On the **Basic Information** page, choose **Node Management** in the navigation pane on the left. In the **Security Group** area on the right, click the name of the security group.

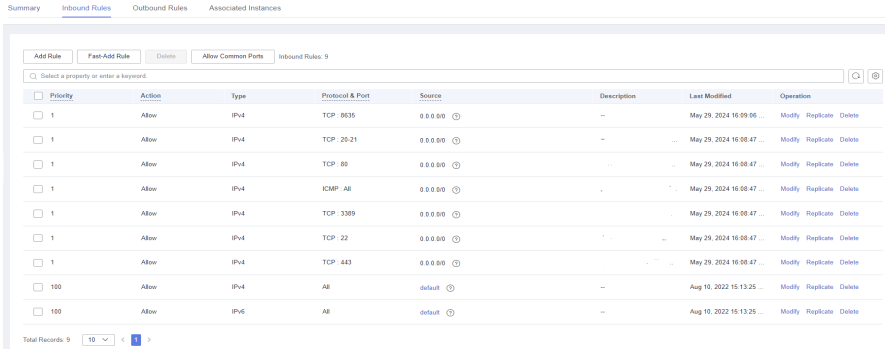
Figure 4-51 Security group



Step 5 Add Inbound Rule

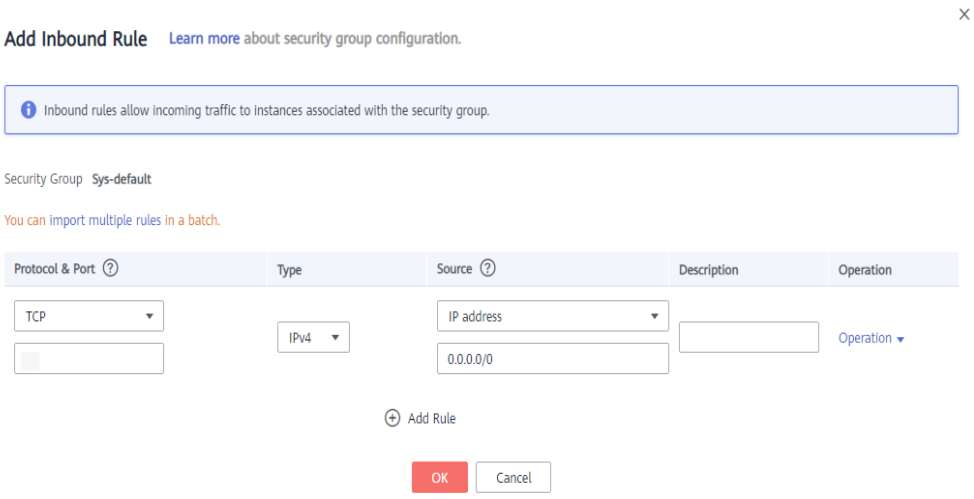
1. Click the **Inbound Rules** tab.

Figure 4-52 Inbound rules



2. Click **Add Rule**. The **Add Inbound Rule** dialog box is displayed.

Figure 4-53 Adding a rule



3. Add a security group rule as prompted.

Table 4-30 Inbound rule settings

Parameter	Description	Example Value
Protocol & Port	– The network protocol required for access. Available options: **All**, **TCP**, **UDP**, **ICMP**, or **GRE** – **Port**: The port or port range that allows the access to the ECS. Range: 1 to 65535 Common ports are listed in **Common Ports Used by ECS**.	TCP
Type	IP address type. This parameter is available after IPv6 is enabled. – IPv4 – IPv6	IPv4
Source	The IP address, IP address group, or security group that the rule applies to, which allows access from IP addresses or instances in another security group. Examples: – IPv4 single IP address: 192.168.10.10/32 – Subnet: 192.168.1.0/24 – All IP addresses: 0.0.0.0/0 – sg-abc (security group) For more information about IP address groups, see **IP Address Group**.	0.0.0.0/0
Description	(Optional) Provides supplementary information about the security group rule. The description can contain up to 255 characters and cannot contain angle brackets (<>).	-

Step 6 Click **OK**.

----End

4.3.5.4 Viewing the IP Address and Port Number of a GeminiDB Redis Instance

This section describes how to query the IP address and port number of an instance on the management console.

Viewing the Load Balancer IP Address and Port

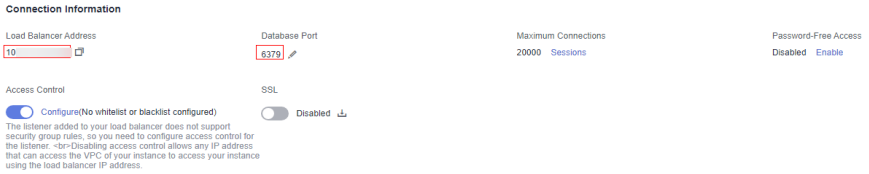
Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the instance whose IP address and port you want to view and click its name.

Step 4 In the **Connection Information** area, view the load balancer IP address and corresponding port.

Figure 4-54 Viewing the load balancer IP address and port



----End

Viewing the Private IP Address or EIP

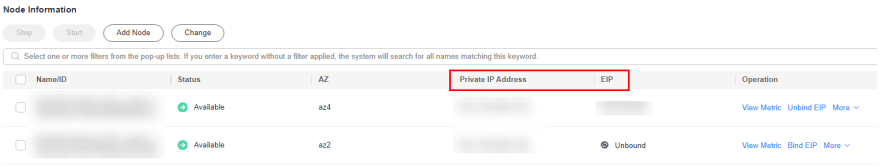
Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the instance whose node IP addresses you want to view and click its name.

In the navigation pane on the left, click **Node Management** to view the private IP addresses and EIPs of the instance.

Figure 4-55 Obtaining IP addresses



----End

Viewing the Port for Accessing Each Instance Node

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the instance whose node access ports you to want view and click its name.

In the **Connection Information** area on the **Basic Information** page, view the port of each instance node.

Figure 4-56 Obtaining the port number

----End

4.3.5.5 Binding an EIP to a GeminiDB Redis Instance

Scenarios

After you create a GeminiDB Redis instance, you can bind an EIP to it to allow external access. If later you want to prohibit external access, you can also unbind the EIP from the DB instance.

Precautions

- To change the EIP that has been bound to a node, unbind it from the node first.
- You need to estimate the bandwidth required by services and purchase an EIP with sufficient bandwidth resources. **Client access exceptions caused by poor public network performance will not be included in the SLA.**

Binding an EIP

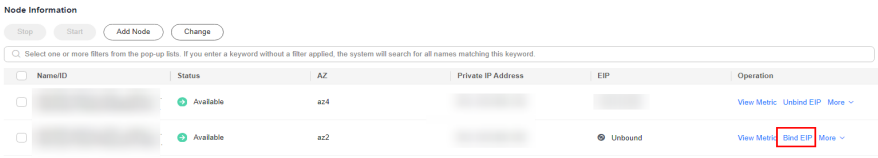
Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the instance that you want to bind an EIP to and click its name.

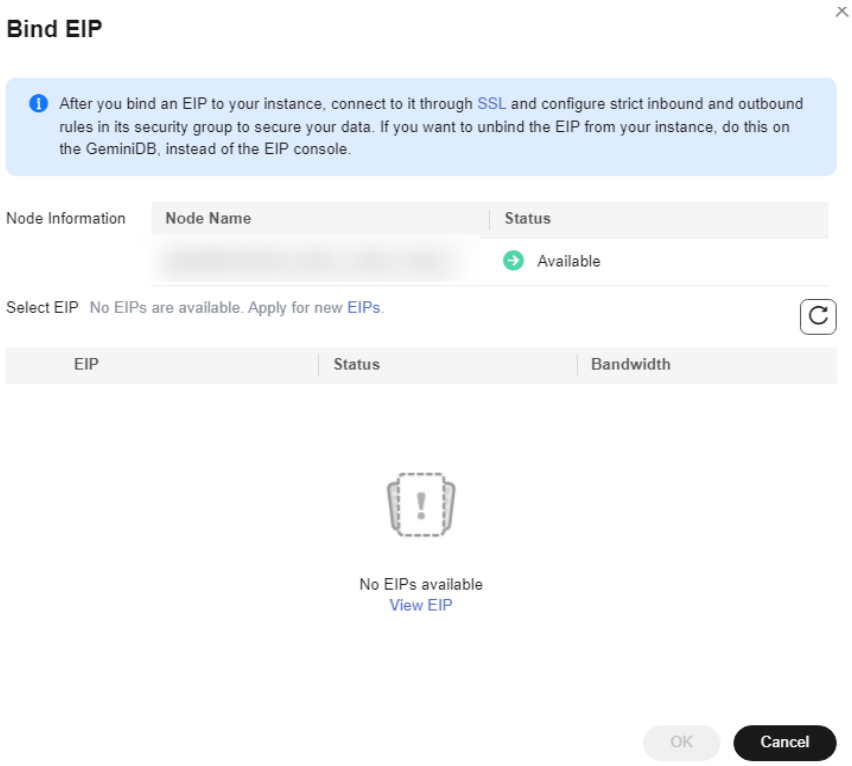
Step 4 In the **Node Information** area, locate the target node and click **Bind EIP** in the **Operation** column.

Figure 4-57 Binding an EIP



Step 5 In the displayed dialog box, view all available EIPs, select the required EIP, and click **OK**. If no available EIPs are displayed, click **View EIP** and create an EIP.

Figure 4-58 Selecting an EIP



Step 6 In the **EIP** column, view the EIP that is successfully bound.

To unbind the EIP from the DB instance, see **Unbinding an EIP**.

----End

Unbinding an EIP

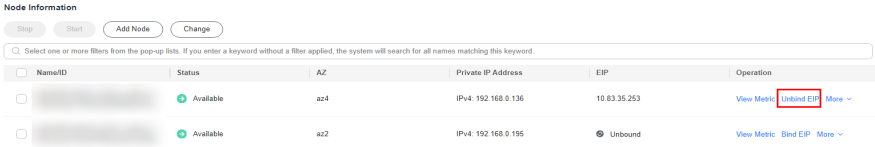
Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the instance that you want to unbind an EIP from.

Step 4 On the **Basic Information** page, in the **Node Information** area, locate the target node and click **Unbind EIP** in the **Operation** column.

Figure 4-59 Binding an EIP



Step 5 In the displayed dialog box, click **Yes**.

To bind an EIP to the DB instance again, see **Binding an EIP**.

----End

4.3.5.6 Encrypting Data over SSL for a GeminiDB Redis Instance

Secure Socket Layer (SSL) is an encryption-based Internet security protocol for establishing an encrypted link between a server and a client. It provides privacy, authentication, and integrity to Internet communications.

- Authenticates users and servers, ensuring that data is sent to the correct clients and servers.
- Encrypts data to prevent it from being intercepted during transfer.
- Ensures data integrity during transmission.

After SSL is enabled, you can establish an encrypted connection between your client and the instance you want to access to improve data security.

Precautions

- After you enable or disable SSL, the established connection is interrupted. Restart the instance to apply the change.
- Enabling SSL will prolong network connection response time and increase CPU usage. So, evaluate impacts on service performance before enabling SSL.
- The SSL function provided by GeminiDB Redis supports only TLS 1.3 or later.

Enabling SSL

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the target instance.

Step 4 In the **Connection Information** area, click  to enable SSL.

Figure 4-60 Enabling SSL

After SSL is enabled, you can connect to the instance through SSL connections. For details, see **4.3.5.7 Connecting a GeminiDB Redis Instance over SSL**.

----End

Disabling SSL

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the target instance.

Step 4 In the **Connection Information** area, click  to disable SSL.

Figure 4-61 Disabling SSL

After SSL is disabled, you can connect to the GeminiDB Redis instance over a non-SSL connection. For details, see **Procedure**.

----End

4.3.5.7 Connecting a GeminiDB Redis Instance over SSL

GeminiDB Redis allows you to connect to a GeminiDB Redis instance through Redis-cli in SSL mode for data encryption and higher security. This section describes how to connect to a GeminiDB Redis instance using SSL.

Usage Notes

- The target instance must be in the same VPC and subnet as the ECS.
- The ECS must be in a security group that has access to the instances. For details, see **4.3.5.3 Configuring Security Group Rules for a GeminiDB Redis Instance**.
- After the SSL connection is enabled, download the SSL certificate for your applications to access to the GeminiDB Redis instance.
- If the SSL connection is used, ensure that the Redis client, for example, Redis-cli 6.x, supports SSL.

Prerequisites

An ECS has been created. The following uses a Linux ECS as an example. For details, see **Purchasing an ECS** in *Getting Started with Elastic Cloud Server*.

Procedure

Step 1 Log in to the ECS. For details, see **Logging In to an ECS** in *Getting Started with Elastic Cloud Server*.

Step 2 Obtain the Redis client.

Method 1

Run the following command to download the Redis client.

```
wget https://download.redis.io/releases/redis-6.2.6.tar.gz
```

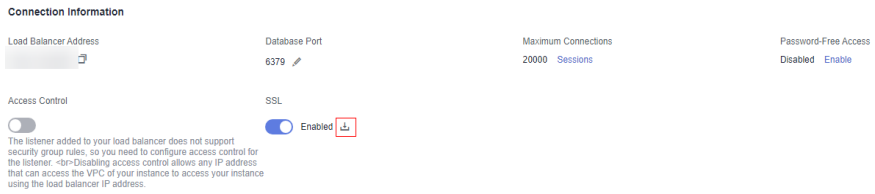

Method 2

Download the **Redis client** and upload the Redis client installation package to the ECS.

Step 3 Obtain the SSL certificate.

Click the target instance name. On the **Basic Information** page, in the **Connection Information** area, click the download button in the **SSL** field to obtain the SSL certificate.

Figure 4-62 Obtaining the SSL certificate



Step 4 Upload the SSL certificate to the ECS.

Step 5 Check the OpenSSL version supported by the ECS OS.

```
openssl version
```

NOTE

- The SSL function provided by GeminiDB Redis supports only TLS 1.3 or later.
- The OpenSSL version in the ECS OS must be 1.1.1 or later so that redis-cli can support TLS 1.3 or later.

- If the OS version is earlier than 1.1.1, perform the following steps to install OpenSSL:

```
wget https://www.openssl.org/source/openssl-1.1.1m.tar.gz
tar -zxvf openssl-1.1.1m.tar.gz
cd openssl-1.1.1m/
./config --prefix=/usr/local/openssl-1.1.1m_install_dir
make
make install
```

  After OpenSSL is installed, go to **Step 6**.

- If the OS is 1.1.1 or later, go to **Step 6**.

Step 6 Decompress the client package.

```
tar -xzf redis-6.2.6.tar.gz
```

Step 7 Open the **src** directory and connect to the DB instance.

- If the required OpenSSL version has been installed by performing **Step 5** and the version is earlier than 1.1.1, you can connect to the DB instance using the following method:

```
cd redis-6.2.6
make BUILD_TLS=yes OPENSSL_PREFIX=/usr/local/openssl-1.1.1m_install_dir
cd src
LD_PRELOAD=/usr/local/openssl-1.1.1m_install_dir/lib/libssl.so.1.1:/usr/local/
openssl-1.1.1m_install_dir/lib/libcrypto.so.1.1 ./redis-cli -h <DB_HOST> -p <DB_PORT> -a
<DB_PWD> --tls --cacert <CACERT_PATH>
```

  Example:

```
LD_PRELOAD=/usr/local/openssl-1.1.1m_install_dir/lib/libssl.so.1.1:/usr/local/
openssl-1.1.1m_install_dir/lib/libcrypto.so.1.1 ./redis-cli -h 192.168.0.208 -p 6379 -a <DB_PWD> --
tls --cacert ./cacert.crt
```


- If the OpenSSL version in the ECS OS is 1.1.1 or later, you can connect to the DB instance using the following method:

```
cd redis-6.2.6
make BUILD_TLS=yes
cd src
./redis-cli -h <DB_HOST> -p <DB_PORT> -a <DB_PWD> --tls --cacert <CACERT_PATH>
```

  Example:

```
./redis-cli -h 192.168.0.208 -p 6379 -a <DB_PWD> --tls --cacert ./cacert.crt
```

Table 4-31 Parameter Description

Parameter	Description
<DB_HOST>	Private IP address of an instance to be connected. To obtain this IP address, go to the **Instances** page, locate the instance, and click its name. The IP address can be found in the **Private IP Address** field under **Node Information** on the **Basic Information** page. If the instance you purchased has multiple nodes, select the private IP address of any node.
<DB_PORT>	Port for accessing the target instance. Configure this parameter based on service requirements. To obtain the port number, perform the following steps: On the **Instances** page, locate the instance and click its name. On the **Basic Information** page, in the **Network Information** area, view the database port.
<DB_PWD>	Administrator password set when you buy a GeminiDB Redis instance
<CACERT_PATH>	SSL certificate path

Step 8 Check the results. If information similar to the following is displayed, the connection is successful.

```
IP:port>
```

----End

4.3.5.8 Changing the Security Group of a GeminiDB Redis Instance

Scenarios

You can change the security group of a GeminiDB Redis instance.

Precautions

- If you are adding nodes to an instance, the security group cannot be changed.

Procedure

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the instance whose security group you want to change and click its name.

Step 4 In the navigation pane, choose **Node Management**.

Step 5 In the **Security Group** area, click to select a security group.

- To submit the change, click . This process takes about 1 to 3 minutes.
- To cancel the change, click .

Step 6 View the modification result.

----End

4.3.5.9 Configuring Private Network Access to a GeminiDB Redis Instance

Scenarios

GeminiDB Redis allows you to enable or disable private network access for a load balancer.

Precautions

- A load balancer address does not support security groups. After instance creation is complete, configure IP address access control. If no whitelist is configured, all IP addresses that can communicate with the VPC can access the instance.

Enabling a Blacklist/Whitelist for a Load Balancer IP Address

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

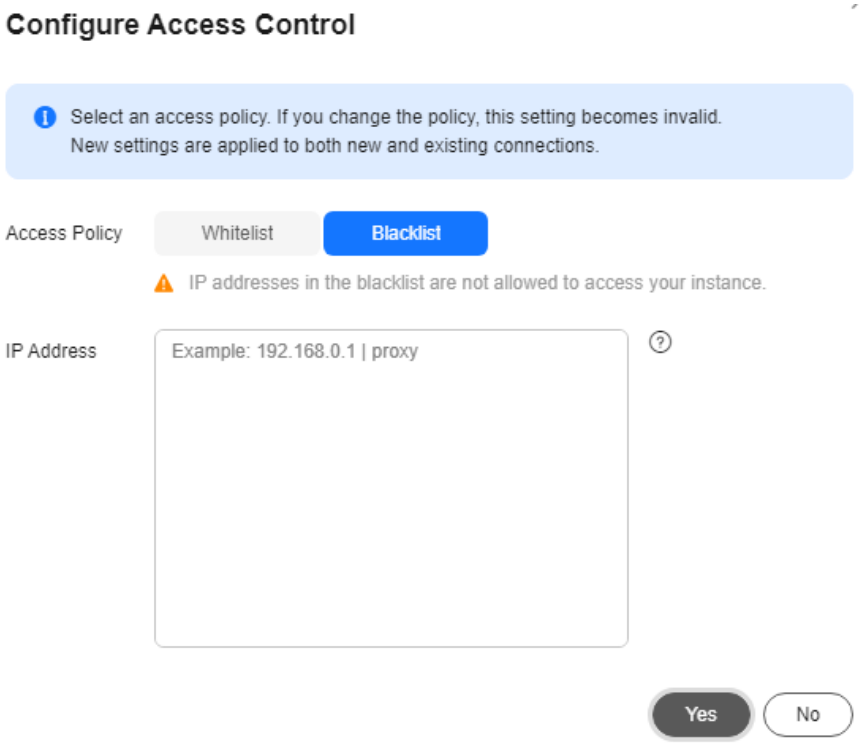
Step 3 On the **Instances** page, click the target instance.

Step 4 In the **Connection Information** area, click next to **Access Control**.

Figure 4-63 Enabling private network access for a load balancer

Step 5 Select **Blacklist** or **Whitelist** and specify IP addresses in that list.

Figure 4-64 Configuring access control



- Blacklist: The blocklist and allowlist cannot be configured at the same time. If you switch between lists, your previously entered settings will be lost. IP addresses in the blacklist cannot be accessed. Exercise caution when performing this operation.
- Whitelist: The blocklist and allowlist cannot be configured at the same time. If you switch between lists, your previously entered settings will be lost. Only IP addresses in the whitelist are allowed to access the system. Exercise caution when performing this operation.

----End

Disabling Private Network Access for a Load Balancer

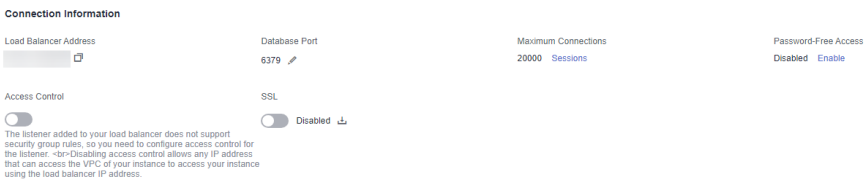
Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the target instance.

Step 4 In the **Connection Information** area, click  next to **Access Control**. In the displayed dialog box, click **Yes**.

Figure 4-65 Disabling private network access for a load balancer



Step 5 Check the load balancer address cannot take effect.

----End

4.4 Data Migration

4.4.1 Overview of the Redis Data Migration Solution

This section describes how to migrate services to a GeminiDB Redis instance. If you have any questions about the migration, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console to get technical support.

Usage Notes

Cloud services such as Alibaba Cloud Redis and TencentDB for Redis cannot use Huawei Cloud DRS for data migration due to the following factors:

- Some self-developed Redis-like databases are not compatible with the PSync protocol.
- Architecture restrictions: For many cloud vendors, the proxy component is added between users and Redis. PSync is not supported due to the proxy.
- Security restrictions: In native Redis, fork() is used over PSync, which causes memory expansion, user request delay increase, and even out of memory.
- Business strategy: A large number of users use RedisShake to migrate services from the cloud or change the cloud, so PSync is shielded.

Migration Tool

- Data Replication Service (DRS) is used for full and incremental data migration while ensuring data security. For details, see **Migration Overview**.
- **Redis-Shake** tool: is an open-source migration tool that supports migration modes such as full scanning (rump), data restoration (restore), and incremental synchronization (sync). You can download the tool to an ECS and use CLI to facilitate migration.

Required Permissions

- Ensure that the database port is enabled in the security group of the GeminiDB Redis instance.

Migration Scenarios

Table 4-32 Migration scenarios

No.	Source	Destination	Migration Solution
1	GeminiDB Redis API	Open-source Redis/ GeminiDB Redis API	**4.4.2 (Recommended) Using DRS to Migrate Data from a GeminiDB Redis Instance to an Open-Source Redis Instance**

No.	Source	Destination	Migration Solution
2	Alibaba Cloud Redis/Tair	GeminiDB Redis	4.4.3 Migrating the Alibaba Cloud Database Redis/Tair To GeminiDB Redis
3	Open-source Redis	GeminiDB Redis	4.4.4 (Recommended) Using DRS to Migrate Data from Open-source Redis or Redis Cluster to GeminiDB Redis API
4	Open-source Redis	GeminiDB Redis	4.4.5 Migrating Data from Open-source Redis to GeminiDB Redis API Using Redis-Shake
5	RDB file	GeminiDB Redis	4.4.6 Using Redis-Shake to Import an RDB or AOF File to a GeminiDB Redis Instance
6	RDB file	GeminiDB Redis	4.4.7 (Recommended) Importing Data to Restore RDB Files to a GeminiDB Redis Instance
7	Self-built Kvrocks	GeminiDB Redis	4.4.8 From Kvrocks to GeminiDB Redis API
8	Self-built Pika	GeminiDB Redis	4.4.9 From Pika to GeminiDB Redis API
9	Self-built SSDB	GeminiDB Redis	4.4.10 From SSDB to GeminiDB Redis API
10	Self-built LevelDB	GeminiDB Redis	4.4.11 From LevelDB to GeminiDB Redis API
11	Self-built RocksDB	GeminiDB Redis	4.4.12 From Kvrocks to GeminiDB Redis API
12	AWS ElastiCache for Redis	GeminiDB Redis	4.4.13 Migration from an AWS ElastiCache for Redis Database to a GeminiDB Redis Instance

4.4.2 (Recommended) Using DRS to Migrate Data from a GeminiDB Redis Instance to an Open-Source Redis Instance

Data Replication Service (DRS) is used for full and incremental data migration while ensuring data security. For details, see **Migration Overview**.

For details about how to use DRS to migrate data from a GeminiDB Redis instance to an open-source Redis instance, see **From GeminiDB Redis to Redis**.

4.4.3 Migrating the Alibaba Cloud Database Redis/Tair To GeminiDB Redis

This section describes how to migrate Alibaba Cloud databases Redis or Tair to GeminiDB Redis.

Migration Principles

- The data migration function of the Alibaba Cloud data migration tool DTS is used to migrate data from Alibaba Cloud Redis to other Redis services. This tool avoids the restrictions of shielding the sync and psync commands of Alibaba Cloud Redis and migrates data from Alibaba Cloud Redis to Huawei Cloud GeminiDB Redis.

Precautions

- The source end on Alibaba Cloud needs to communicate with the destination end on Huawei Cloud. Ensure that a private line is enabled or that binding a public IP address is performed.
- The Alibaba Cloud DTS data migration function is charged in real time. Before using this function, ensure that your Alibaba Cloud account balance is sufficient.
- The Huawei Cloud GeminiDB Redis capacity must be greater than or equal to the memory capacity of the Alibaba Cloud Redis database.
- Ensure that the security group configuration on the source and target ends is enabled.
- Some Redis databases on Alibaba Cloud are special. For example, Tair hybrid storage does not support online full and incremental migration. You can complete the migration by scanning all the data.

Preparations

- Migrating data using a public IP address
  - Purchase a Huawei Cloud EIP in advance. The bandwidth must be greater than the source database traffic.
  - Bind the EIP to a Huawei Cloud GeminiDB Redis node.
  - When configuring DTS, ensure that the destination database is connected through a public IP address.
- Migrating data using a private line
  - Purchase an Alibaba ECS in advance to ensure that it can connect to Huawei Cloud GeminiDB Redis.
  - Configure data forwarding to forward the traffic received by the local port to the destination end, implementing migration from Alibaba Cloud Redis to GeminiDB Redis.

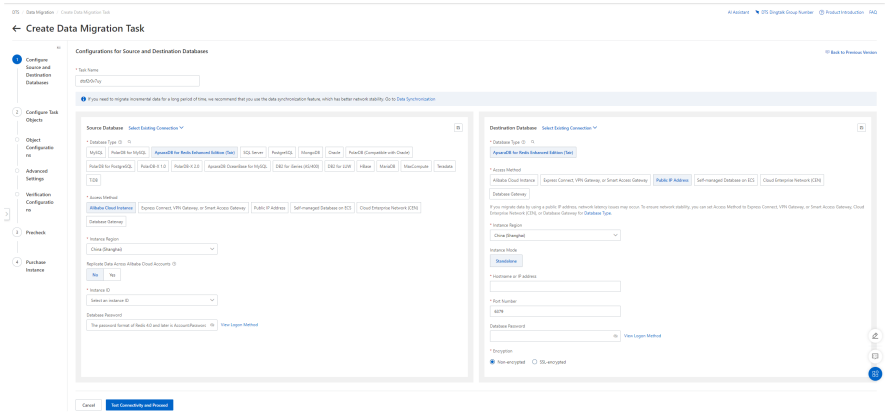
    ssh -g -L (*Forwarding port***): (***LB IP address of Huawei Redis***): (***Huawei Redis port***) -N -f root@ (***Local ECS IP Address***)**
  - When configuring DTS, ensure that the destination database is connected through a self-built ECS database.

Procedure

Purchasing the Data Synchronization Function of DTS

Step 1 Select the Redis service on Alibaba Cloud as the source end. If an EIP is used for migration, select a public IP address as the destination end and enter the EIP as the host name. If Direct Connect is used for migration, select the self-built Redis database on ECS as the destination end, set the host name to the IP address of the ECS, set the port number to the forwarding port number, enter the database password, and click the test link. If no exception occurs during the test, the next page is displayed. Otherwise, check whether the entire link is normal and whether the whitelist configuration is correct.

Figure 4-66 Source and destination configuration information



Step 2 Select **Full Data Migration** or **Full Data Migration + Incremental Data Migration**. Select **Pre-check and Report Errors** and select the database to be migrated.

CAUTION

If you use the multi-DB function, select all databases to be migrated. If the multi-DB function is not used, select only **DB0**.

Figure 4-67 Database to be migrated

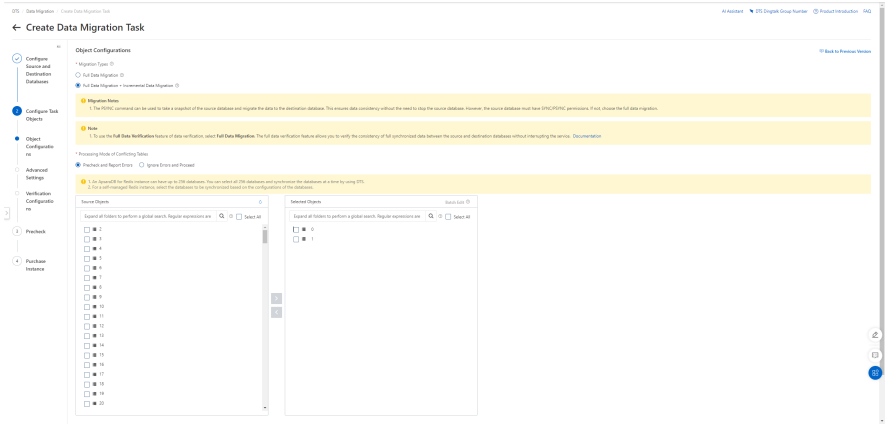
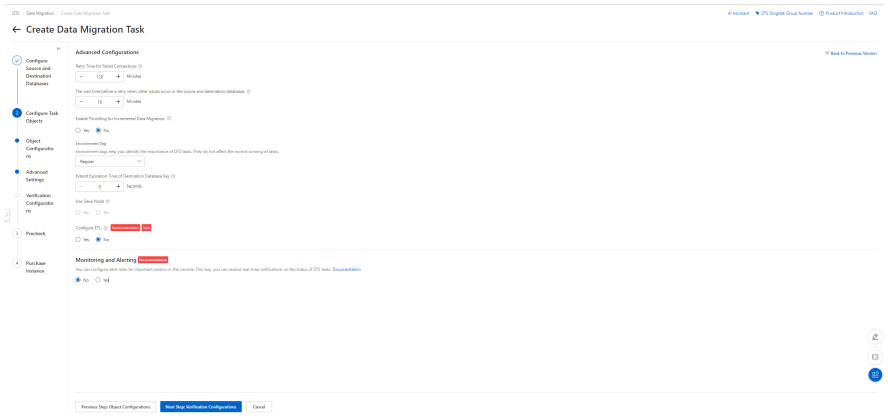
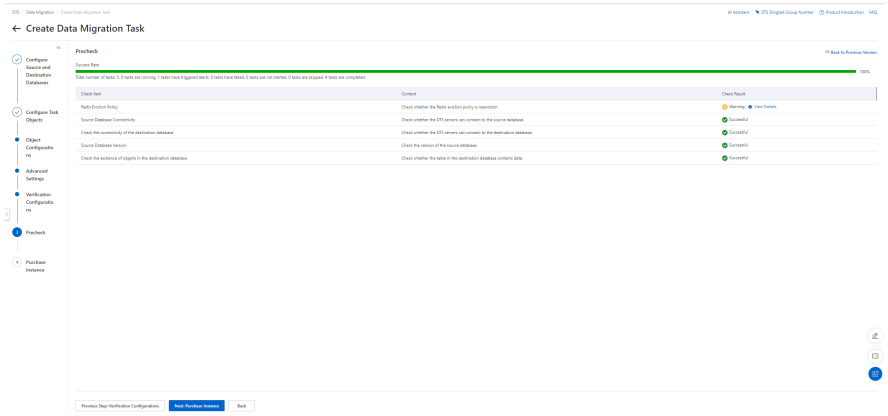


Figure 4-68 Configure advanced settings.



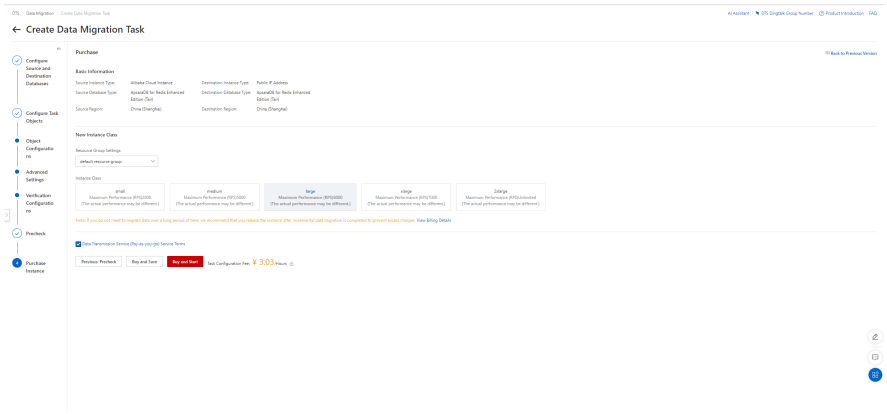
Step 3 After the pre-check is complete, click **Next: Purchase Instance**.

Figure 4-69 Pre-check



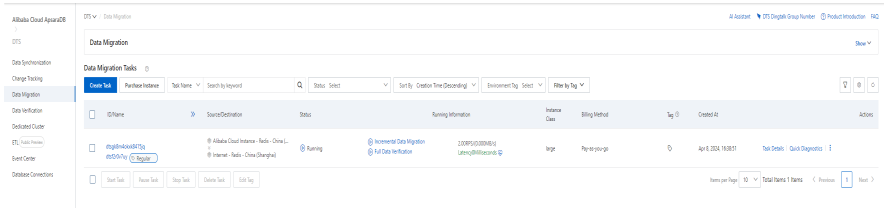
Step 4 Select the bandwidth for the migration and click **Buy and Start**.

Figure 4-70 Bandwidth configuration



Step 5 If **Full Data Migration + Incremental Data Migration** is selected, the migration task will not automatically end. If there is no delay (0 ms), the full synchronization is complete.

Figure 4-71 Task status

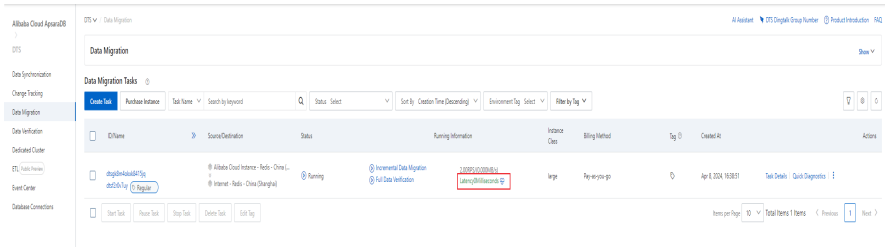


----End

Stopping the Data Synchronization Function of DTS

Step 1 After the Redis service migration, stop the data synchronization task.

Figure 4-72 Stopping the data migration task



----End

Verifying Redis Data Consistency After Migration

After the migration is complete, you can check the consistency of Redis data.

NOTICE

- Data has been migrated from Redis, or incremental migration has started.
- Redis-full-check must be deployed on the ECS, and the ECS is connected to the source and destination databases.
- During incremental migration, data may be inconsistent due to network latency between the source and destination databases. You are advised to stop writing data to the source and then verify data consistency.
- When Redis is used, an expiration time is usually set for keys. During migration, setting a key expiration time affects data consistency. Data may be inconsistent due to inconsistent expiration time.
- During migration, DTS writes temporary probing keys to Redis on the destination database. Non-service data may be detected during data verification, which is normal.

Procedure

Step 1 Log in to the ECS and ensure it is connected to the source and destination Redis databases.

Step 2 Deploy **redis-full-check**.

Step 3 Verify data.

/redis-full-check -s {*Source IP address***}:{***Source port***} -p {***Source password***} -t {***Destination IP address***}:{***Destination port***} -a {***Destination password***} -m 1**

Table 4-33 Parameter description

Parameter	Description	Example Value
-s	Source Redis database address and port number	-s 10.0.0.1:6379
-p	Password of the source Redis database	-
-t	Destination GeminiDB Redis database address and port number	-t 10.0.0.2:6379
-a	Password of the destination GeminiDB Redis database	-

Parameter	Description	Example Value
-m	Verification mode: 1. All key-value pairs 2. Value length only 3. Key integrity only 4. All key values are verified, but only the length of big keys is verified. By default, the second verification mode is used.	-m 1
-q	Maximum QPS. The default value is **15000**.	-q 5000
-d	Name of the file for saving the verification result. The default value is **result.db**.	-d result.db

Step 4 View the verification result file.

By default, three rounds of verification are performed and three verification result files are generated. Generally, you only need to view the last verification result file.

- Run the **sqlite3 result.db.3** command.
- Run the **select * from key** command.
- Check whether there are abnormal keys.

```
Enter ".help" for usage hints.
sqlite> select * from key;
1|b|string|lack_target|0|1|0
2|c|string|lack_target|0|1|0
3|a|string|lack_target|0|1|0
sqlite>
```

----End

4.4.4 (Recommended) Using DRS to Migrate Data from Open-source Redis or Redis Cluster to GeminiDB Redis API

Data Replication Service (DRS) is used for full and incremental data migration while ensuring data security. For details, see **Migration Overview**.

For details about how to use DRS to migrate data from Redis to GeminiDB Redis API, see **From Redis to GeminiDB Redis API**.

For details about how to use DRS to migrate data from Redis Cluster to GeminiDB Redis API, see **From Redis Cluster to GeminiDB Redis API**.

4.4.5 Migrating Data from Open-source Redis to GeminiDB Redis API Using Redis-Shake

You can use DRS or Redis-Shake to migrate data from open-source Redis to GeminiDB Redis API. Redis-Shake is taken as an example in this section.

Migration Principles

Use Redis-Shake to migrate data from an on-premises Redis instance (source) to a GeminiDB Redis instance (destination). Full and incremental migrations are both supported. The source can be a single-node, primary/standby, or cluster instance, or an RDB file.

- Full migration: Redis-Shake works as a slave node for the source, obtains data of an RDB file generated by the source, and then parses the data and sends it to the destination by running commands. You can also use an RDB file as the source to import snapshot data generated at a specific time point.
- Incremental migration: After full migration is complete, Redis-Shake continues sending incremental data to the destination by running commands until you stop Redis-Shake.

Usage Notes

- If data synchronization between master and slave Redis nodes is disconnected, stop Redis-Shake, clear all data in the destination, and retry a migration. To ensure a smooth synchronization, migrate data during off-peak hours and set a large value for parameter **client-output-buffer-limit** to increase the ring buffer size for incremental synchronization.
- Redis-Shake does not write data into the source, but may have a temporary impact on the source performance.
- If the migration involves multiple databases, ensure that source databases are correctly mapped to destination databases to prevent unexpected data overwriting.
- Streaming data cannot be migrated.
- Ensure that network communication among Redis-Shake, the source instance, and the destination instance is normal.
- To migrate data from open-source Redis to GeminiDB Redis API, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service.

Migrating Data from an Open-Source Redis Single-Node or Primary/Standby Instance to a GeminiDB Redis Instance

You can import a file similar to the above or perform the following operations to migrate data from an open-source single-node or primary/standby Redis instance to a GeminiDB Redis instance.

Step 1 Deploy the required migration tool.

1. Obtain the **Redis-Shake package**.

NOTE

Download the Redis-Shake release package and decompress it.

2. Modify the **Redis-Shake.conf** configuration file and configuring the following items:

   log.level = info #Default log level. A printed INFO log contains migration progress information, based on which you can judge whether the migration is complete.

   source.address = <host>:<port> # IP address and port of a host where an open-source Redis instance is deployed

   source.password_raw = *** #** Password for logging in to a source instance

   source.type = standalone # Source instance type

   target.address = <host>:6379 # Destination instance IP address

   target.password_raw = *** #** Password for logging in to a destination instance

   target.version = 5.0 # Version of the destination Redis instance

   target.type = standalone # Destination instance type

   target.db = -1 # Specific database on the destination that all data will be migrated to. If this parameter is set to **-1**, a mapping relationship is established between migrated databases and databases in the source instance.

3. Specify whether data of the destination is overwritten.

   key_exists = none

   NOTE

   If there are duplicate keys on the source and destination, specify whether data of the destination is overwritten. The options are as follows:

   - **rewrite** indicates that the source overwrites the destination.
   - **none** indicates that the migration process exists once duplicate keys are detected.
   - **ignore** indicates that keys in the source are retained and keys in the destination are ignored. This value does not take effect in rump mode.

   none is recommended. There will be no duplicate data because the source is an RDB file. If the migration exits unexpectedly, you can choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service.

Step 2 Migrate data.

Migration starting command:

./redis-shake.linux -conf=redis-shake.conf -type=sync

- If the following information is displayed, the full synchronization is completed and incremental synchronization begins.
  ```
  sync rdb done
  ```
- If the following information is displayed, no new data is incremented. You can stop the migration process to disconnect incremental synchronization:
  ```
  sync:  +forwardCommands=0      +filterCommands=0      +writeBytes=0
  ```

Step 3 Verify data.

Download and decompress **RedisFullCheck** and use it to verify data by referring to **Migrating Data from an Open-Source Redis Single-Node or Primary/Standby Instance to a GeminiDB Redis Instance**.

./redis-full-check -s SOURCE_IP:SOURCE_PORT -p SOURCE_PWD -t TARGET_IP:6379 -a TARGET_PWD

If the following information is displayed, the migration is successful, and data is consistent between the source and destination:

```
all finish successfully, totally 0 key(s) and 0 field(s) conflict
```

----End

Migrating Data from Redis Cluster to GeminiDB Redis API

Configure the following items in the configuration file:

source.address = <host1>:<port1>,<host2>:<port2>,<host2>:<port2> # IP addresses and ports of hosts at the source.

source.type = cluster # Cluster type of the source.

For other steps, see **Migrating Data from an Open-Source Redis Single-Node or Primary/Standby Instance to a GeminiDB Redis Instance**.

Migrating Data from an Open-Source Codis Cluster Instance to a GeminiDB Redis Instance

Obtain host IP addresses and ports of all shards of the Codis cluster instance and configure the configuration file as follows:

source.address = <host1>:<port1>,<host2>:<port2>,<host2>:<port2> # IP addresses and ports of hosts at the source.

source.type = cluster # Cluster type of the source.

For other steps, see **Migrating Data from an Open-Source Redis Single-Node or Primary/Standby Instance to a GeminiDB Redis Instance**.

Fully Scanning Data on and Migrating It from an Open-Source Redis Instance to a GeminiDB Redis Instance

If data cannot be migrated with any of the above methods, try rump of Redis-Shake to scan databases one by one and migrate them.

Step 1 Deploy the required migration tool.

1. Obtain the **Redis-Shake package**.

   NOTE

   Download the Redis-Shake release package and decompress it.

2. Modify the **Redis-Shake.conf** configuration file and configuring the following items:

   log.level = info #Default log level. A printed INFO log contains migration progress information, based on which you can judge whether the migration is complete.

source.address = <host>:<port> # IP address and port of a host where an open-source Redis instance is deployed

source.password_raw = *** #** Password for logging in to a source instance

source.type = standalone # Source instance type

target.address = <host>:6379 # Destination instance IP address

target.password_raw = *** #** Password for logging in to a destination instance

target.version = 5.0 # Version of the destination Redis instance

target.type = standalone # Destination instance type

target.db = -1 # Specific database on the destination that all data will be migrated to. If this parameter is set to **-1**, a mapping relationship is established between migrated databases and databases in the source instance.

3. Specify whether data of the destination is overwritten.

   key_exists = none

   NOTE

   If there are duplicate keys on the source and destination, specify whether data of the destination is overwritten. The options are as follows:

   - **rewrite** indicates that the source overwrites the destination.
   - **none** indicates that the migration process exists once duplicate keys are detected.
   - **ignore** indicates that keys in the source are retained and keys in the destination are ignored. This value does not take effect in rump mode.

   none is recommended. There will be no duplicate data because the source is an RDB file. If the migration exits unexpectedly, you can choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service.

Step 2 Migrate data.

Migration starting command:

./redis-shake.linux -conf=redis-shake.conf -type=rump

- If information similar to the following is displayed, synchronizing full data is complete.

```
dbRumper[0] executor[0] finish
```

Step 3 Verify data.

Download and decompress **RedisFullCheck** and use it to verify data.

./redis-full-check -s SOURCE_IP:SOURCE_PORT -p SOURCE_PWD -t TARGET_IP:6379 -a TARGET_PWD

If the following information is displayed, the migration is successful, and data is consistent between the source and destination:

```
all finish successfully, totally 0 key(s) and 0 field(s) conflict
```

----End

Verifying Data Consistency After Migration

After the migration is complete, you can check data consistency.

NOTICE

- Data has been migrated from Redis, or incremental migration has started.
- Redis-full-check must be deployed on the ECS, and the ECS is connected to the source and destination databases.
- During incremental migration, data may be inconsistent due to network latency between the source and destination databases. You are advised to stop writing data to the source and then verify data consistency.
- When Redis is used, an expiration time is usually set for keys. During migration, setting a key expiration time affects data consistency. Data may be inconsistent due to inconsistent expiration time.
- During migration, DTS writes temporary probing keys to Redis on the destination database. Non-service data may be detected during data verification, which is normal.

Procedure

Step 1 Log in to the ECS and ensure it is connected to the source and destination Redis databases.

Step 2 Deploy **redis-full-check**.

Step 3 Verify data.

/redis-full-check -s {*Source IP address***}:{***Source port***} -p {***Source password***} -t {***Destination IP address***}:{***Destination port***} -a {***Destination password***} -m 1**

Table 4-34 Parameter description

Parameter	Description	Example Value
-s	Source Redis database address and port number	-s 10.0.0.1:6379
-p	Password of the source Redis database	-
-t	Destination GeminiDB Redis database address and port number	-t 10.0.0.2:6379
-a	Password of the destination GeminiDB Redis database	-

Parameter	Description	Example Value
-m	Verification mode: 1. All key-value pairs 2. Value length only 3. Key integrity only 4. All key values are verified, but only the length of big keys is verified. By default, the second verification mode is used.	-m 1
-q	Maximum QPS. The default value is **15000**.	-q 5000
-d	Name of the file for saving the verification result. The default value is **result.db**.	-d result.db

Step 4 View the verification result file.

By default, three rounds of verification are performed and three verification result files are generated. Generally, you only need to view the last verification result file.

- Run the **sqlite3 result.db.3** command.
- Run the **select * from key** command.
- Check whether there are abnormal keys.

```
Enter ".help" for usage hints.
sqlite> select * from key;
1|b|string|lack_target|0|1|0
2|c|string|lack_target|0|1|0
3|a|string|lack_target|0|1|0
sqlite>
```

----End

4.4.6 Using Redis-Shake to Import an RDB or AOF File to a GeminiDB Redis Instance

Importing an RDB File to a GeminiDB Redis Instance

Step 1 Deploy the required migration tool.

1. Obtain **Redis-Shake**.

   NOTE

   Download the Redis-Shake release package and decompress it.

2. Modify the **Redis-Shake.conf** configuration file and configuring the following items:

   log.level = info #Default log level. A printed INFO log contains migration progress information, based on which you can judge whether the migration is complete.

   source.rdb.input = /xx/xx.rdb # Absolute path of the source RDB file

   target.address = <host>:6379 # Destination instance IP address

   target.password_raw = ***** # Password for logging in to a destination instance

   target.version = 5.0 # Version of the destination Redis instance

   target.type = standalone # Destination instance type

   target.db = -1 # Specific database on the destination that all data will be migrated to. If this parameter is set to **-1**, a mapping relationship is established between migrated databases and databases in the source instance.

   target.dbmap = #Configure the database migration mapping. The value of **target.db** must be **-1**, for example, **0-5**. **1-3** indicates that data in source database **db0** will be written to destination database **db5** and data in source database **db1** will be written to destination database **db3**.

   big_key_threshold = 52428800 # Big key threshold. If the number of value bytes corresponding to a key exceeds the threshold, data is written in batches.

   resume_from_break_point = false #Disable resumable download. This function is unavailable.

3. Specify whether data of the destination is overwritten.

   key_exists = none

   NOTE

   If there are duplicate keys on the source and destination, specify whether data of the destination is overwritten. The options are as follows:

   - **rewrite** indicates that the source overwrites the destination.
   - **none** indicates that the migration process exists once duplicate keys are detected.
   - **ignore** indicates that keys in the source are retained and keys in the destination are ignored. This value does not take effect in rump mode.

   none is recommended. There will be no duplicate data because the source is an RDB file. If the migration exits unexpectedly, you can choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service.

Step 2 Migrate data.

Migration starting command:

./redis-shake.linux -conf=redis-shake.conf -type=restore

NOTE

Use the restore mode because the source is an RDB file.

Stop the migration process after the migration is complete.

Step 3 Verify data.

Data is obtained from the RDB file. Therefore, you need to check the GeminiDB Redis data at the destination end from the service perspective.

----End

Importing the AOF File To GeminiDB Redis

Step 1 Upload the generated AOF file to an ECS.

Step 2 Start the open-source Redis 5.0 single-node process on the ECS to load the AOF file and wait till the process is started. Ensure that the startup directory of the open-source Redis is the same as the directory containing the AOF file.

Step 3 Run the SAVE command to generate an RDB file. Place the RDB file in the startup directory of the open-source Redis.

Step 4 Stop the open-source Redis 5.0 process.

Step 5 Perform the migration by following **Importing an RDB File to a GeminiDB Redis Instance**.

----End

4.4.7 (Recommended) Importing Data to Restore RDB Files to a GeminiDB Redis Instance

Scenarios

Redis data of other vendors or self-hosted Redis can be migrated to GeminiDB Redis API.

You need to download the source Redis data, then upload the data to an OBS bucket in the same region as the GeminiDB Redis instance, and create a data import task on the GeminiDB console to import the data to the GeminiDB Redis instance.

Precautions

- Importing data will overwrite data of the current database.
- Importing backups generated by a later-version Redis instance to an earlier one may fail.
- Before importing backups, ensure that resource-intensive commands (such as FLUSHALL, KEYS, and HGETALL) have been disabled on the target Redis instance.
- If a backup contains multi-DB data, its database count cannot exceed what is supported by the target Redis instance.
- Only .rdb files can be imported.

Creating an OBS Bucket and Uploading Backups

If the backup to be uploaded is larger than 5 GB, follow the **instructions** provided by OBS.

Perform the following steps if the backup is smaller than 5 GB:

Step 1 Create an OBS bucket.

When creating an OBS bucket, configure the following parameters. For details, see **Creating a Bucket** in *Object Storage Service User Guide*.

1. **Region**:

   The OBS bucket must be in the same region as the destination Redis instance.
2. **Storage Class**: Available options are **Standard**, **Infrequent Access**, and **Archive**.

   Do not select **Archive**. Otherwise, the backup may fail to be imported.
3. Click **Create Now**.

Step 2 In the bucket list, click the bucket created in **Step 1**.

Step 3 In the navigation pane, choose **Objects**.

Step 4 On the **Objects** tab page, click **Upload Object**.

Step 5 Specify **Storage Class**.

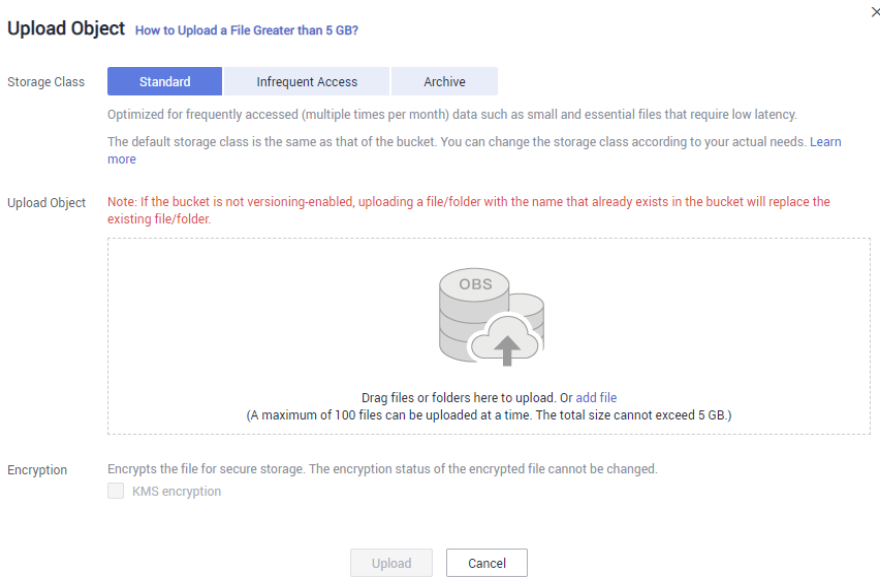
Do not select **Archive**. Otherwise, the backup may fail to be imported.

Step 6 Upload the objects.

Drag files or folders to the **Upload Object** area or click **add file**.

A maximum of 100 files can be uploaded at a time. The total size cannot exceed 5 GB.

Figure 4-73 Uploading objects in batches



Step 7 (Optional) Select **KMS encryption** to encrypt the uploaded files.

Step 8 Click **Upload**.

----End

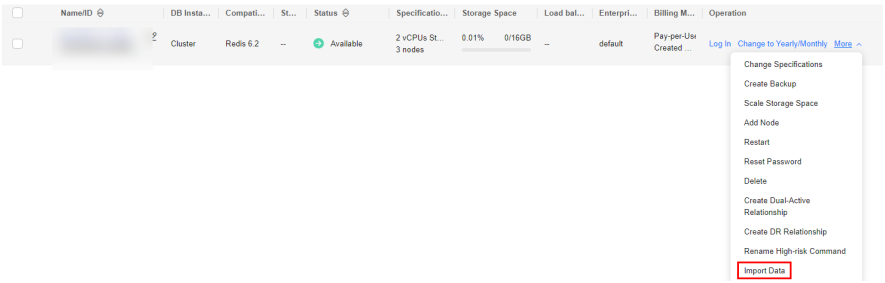
Importing Backups

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the target instance and choose **More** > **Import Data** in the **Operation** column.

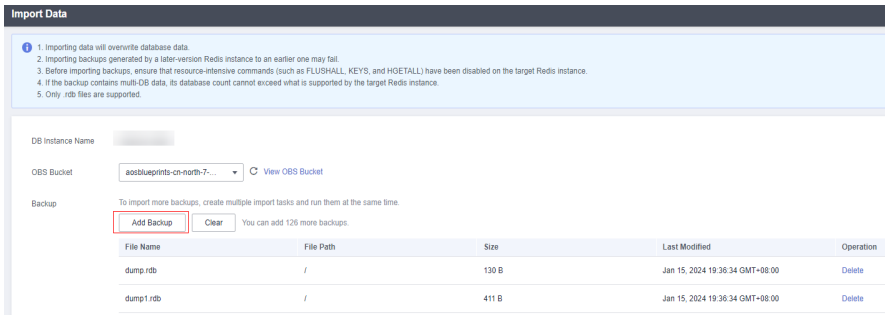
Figure 4-74 Importing data



Step 4 On the **Data Import** page, specify **OBS Bucket** to which a backup have been uploaded.

Step 5 Click **Add Backup** and select the backups to be imported.

Figure 4-75 Adding backups



- A maximum of 128 backups can be added at a time.
- To delete a backup, locate the target backup and click **Delete** in the **Operation** column.
- To delete all backups, select **Clear** for **Backup**.

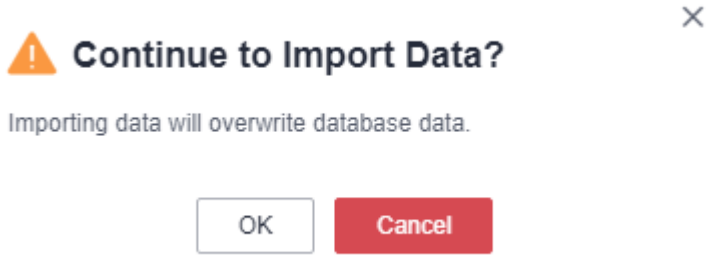
Step 6 Click **Create Now**.

Step 7 Confirm the data import and click **OK**.

CAUTION

Importing data will overwrite data of the current database.

Figure 4-76 Confirming to import data



----End

4.4.8 From Kvrocks to GeminiDB Redis API

Kvrocks is an open-source NoSQL key-value database that is compatible with the Redis ecosystem. It uses namespace to partition data based on the underlying RocksDB. However, it is relatively weak in cluster management. Kvrocks needs to cooperate with other components to create clusters and does not support some Redis commands, such as stream and hyperloglog that are frequently used in message flow and statistics scenarios.

GeminiDB Redis API is a cloud-native NoSQL database with decoupled compute and storage and full compatibility with Redis. To ensure data security and reliability, it provides multi-copy, strict consistency based on a shared storage pool. It provides high compatibility, cost-effectiveness, high reliability, elastic scalability, high availability, and hitless scale-out. GeminiDB Redis API functions as good as Redis Cluster does and is 100% compatible with native APIs. You can migrate your on-premises Redis databases to GeminiDB Redis ones without modifying any code. In addition to adapting to Kvrocks, GeminiDB Redis API also improves management capability and compatibility with Redis.

This section describes how to migrate data from Kvrocks to GeminiDB Redis API.

Migration Principles

The open-source tool kvrocks2redis is used to migrate data from Kvrocks to GeminiDB Redis API. At the code layer, Kvrocks namespace is adapted to the source GeminiDB Redis database.

The migration process consists of two phases: full migration and incremental migration. During full migration that is first performed, snapshots are created for Kvrocks and the corresponding data version (seq) is recorded. Then, the complete data files are parsed into Redis commands and written to GeminiDB Redis API. After the full migration is complete, the incremental migration starts. The migration tool cyclically sends PSYNC commands to Kvrocks and continuously forwards the obtained incremental data to GeminiDB Redis API.

Precautions

- Kvrocks2redis needs to extract data from Kvrocks to local files, parse commands from the files, and send the commands to the target GeminiDB

Redis instance. During this process, the performance of the source DB may be affected, but no data is compromised theoretically.

- If a fault occurs when the migration tool is running, the migration tool automatically stops to facilitate fault locating.
- For security purposes, GeminiDB Redis API does not provide database clearing commands. Ensure that no data exists in the database before the migration.

Prerequisites

- Deploy the kvrocks2redis on an independent host.
- Ensure that the source DB, target DB, and migration tool can communicate with each other.
- Back up data of the source Kvrocks instance in advance.
- Clear all data on the destination GeminiDB Redis instance.

Procedure

To migrate data from Kvrocks to GeminiDB Redis API, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service.

Verifying Data Consistency After Migration

After the migration is complete, you can check data consistency.

NOTICE

- Data has been migrated from Redis, or incremental migration has started.
- Redis-full-check must be deployed on the ECS, and the ECS is connected to the source and destination databases.
- During incremental migration, data may be inconsistent due to network latency between the source and destination databases. You are advised to stop writing data to the source and then verify data consistency.
- When Redis is used, an expiration time is usually set for keys. During migration, setting a key expiration time affects data consistency. Data may be inconsistent due to inconsistent expiration time.
- During migration, DTS writes temporary probing keys to Redis on the destination database. Non-service data may be detected during data verification, which is normal.

Procedure

Step 1 Log in to the ECS and ensure it is connected to the source and destination Redis databases.

Step 2 Deploy **redis-full-check**.

Step 3 Verify data.

/redis-full-check -s {*Source IP address***}:{***Source port***} -p {***Source password***} -t {***Destination IP address***}:{***Destination port***} -a {***Destination password***} -m 1**

Table 4-35 Parameter description

Parameter	Description	Example Value
-s	Source Redis database address and port number	-s 10.0.0.1:6379
-p	Password of the source Redis database	-
-t	Destination GeminiDB Redis database address and port number	-t 10.0.0.2:6379
-a	Password of the destination GeminiDB Redis database	-
-m	Verification mode: 1. All key-value pairs 2. Value length only 3. Key integrity only 4. All key values are verified, but only the length of big keys is verified. By default, the second verification mode is used.	-m 1
-q	Maximum QPS. The default value is **15000**.	-q 5000
-d	Name of the file for saving the verification result. The default value is **result.db**.	-d result.db

Step 4 View the verification result file.

By default, three rounds of verification are performed and three verification result files are generated. Generally, you only need to view the last verification result file.

- Run the **sqlite3 result.db.3** command.
- Run the **select * from key** command.
- Check whether there are abnormal keys.


```
Enter ".help" for usage hints.
sqlite> select * from key;
1|b|string|lack_target|0|1|0
2|c|string|lack_target|0|1|0
3|a|string|lack_target|0|1|0
sqlite>
```

----End

4.4.9 From Pika to GeminiDB Redis API

Pika is a persistent large-capacity Redis storage service. It breaks through the memory bottleneck of Redis due to the large amount of data. However, it is relatively weak in cluster management, and requires twemproxy or codis to shard static data. Compared with the Redis community edition, the database performance is significantly lowered because Pika stores all data in disks.

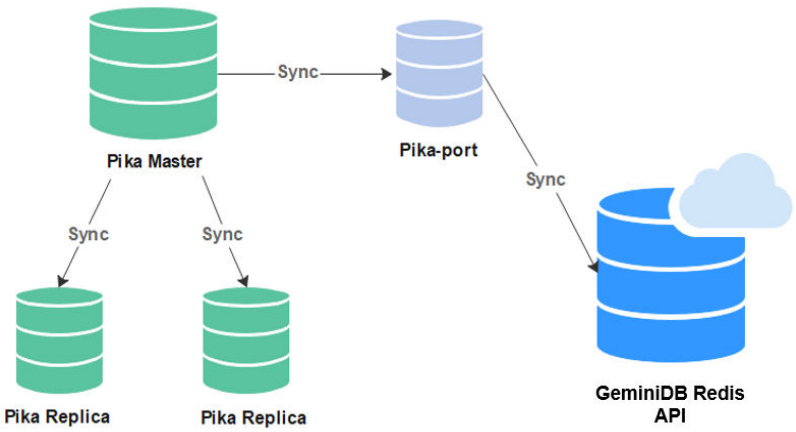
GeminiDB Redis API is a cloud-native NoSQL database with decoupled compute and storage and full compatibility with Redis. To ensure data security and reliability, it provides multi-copy, strict consistency based on a shared storage pool. It supports cold and hot data separation. Hot data can be read from the cache directly, improving read efficiency. RocksDB has been customized to allow the storage capacity to be scaled in seconds. A proxy is used to ensure that upper-layer applications are not affected by underlying sharding or scaling.

This section describes how to migrate data from Pika to GeminiDB Redis API.

Migration Principles

The pika-port tool is used and acts as a slave node of Pika and data is migrated in master/slave replication mode. The master Pika node compares pika-port with its own binlog offset to determine whether to perform full migration or incremental migration. If full migration is required, the master Pika node sends the full data snapshot to pika-port, and pika-port sends the parsed snapshot data to GeminiDB Redis API. After the full migration is complete, incremental migration starts. pika-port parses the incremental data and sends the data to GeminiDB Redis API in the form of Redis commands.

Figure 4-77 Migration Principles



Precautions

- pika-migrate and pika-port act as the slave node of the source Pika and reads only full and incremental data without damaging your data.
- The master/slave synchronization process between the source DB and pika-migrate and pika-port is added, which may affect the performance of the source DB.
- Full and incremental migration can be performed without service interruption. Services are interrupted for a short period of time when services are switched over to GeminiDB Redis API.

Prerequisites

Deploy the migration tool pika-port to ensure that the network connection between the source DB and target Pika instance is normal.

Procedure

To migrate data from Pika to GeminiDB Redis API, choose **Service Tickets > Create Service Ticket** in the upper right corner of the management console and contact customer service.

Migration Performance Reference

- Environment: Pika (single node) and pika-port are deployed on an ECS with 8 vCPUs and 32 GB memory on Huawei Cloud. The target DB is a three-node GeminiDB Redis instance with 8 vCPUs and 16 GB memory.
- Preset data: Use the memtier_benchmark tool to preset 200 GB of data.
- Migration performance: about 50,000 QPS.

4.4.10 From SSDB to GeminiDB Redis API

SSDB is a high-performance NoSQL database written in C/C++. It is compatible with Redis APIs and supports multiple data structures, including key-value pairs, hashmap, sorted set, and list. SSDB is a persistent KV storage system and uses leveldb as the underlying storage engine. Its services directly interact with LevelDB. Operations such as compaction have direct impact on service read and write. GeminiDB Redis API is a cloud-native NoSQL database with decoupled compute and storage and full compatibility with Redis. To ensure data security and reliability, it provides multi-copy, strict consistency based on a shared storage pool. RocksDB is used as the storage engine. Compared with leveldb, RocksDB greatly improves performance, solves the problem that leveldb proactively restricts write, and implements cold and hot separation, reducing the impact of operations at the storage layer on performance.

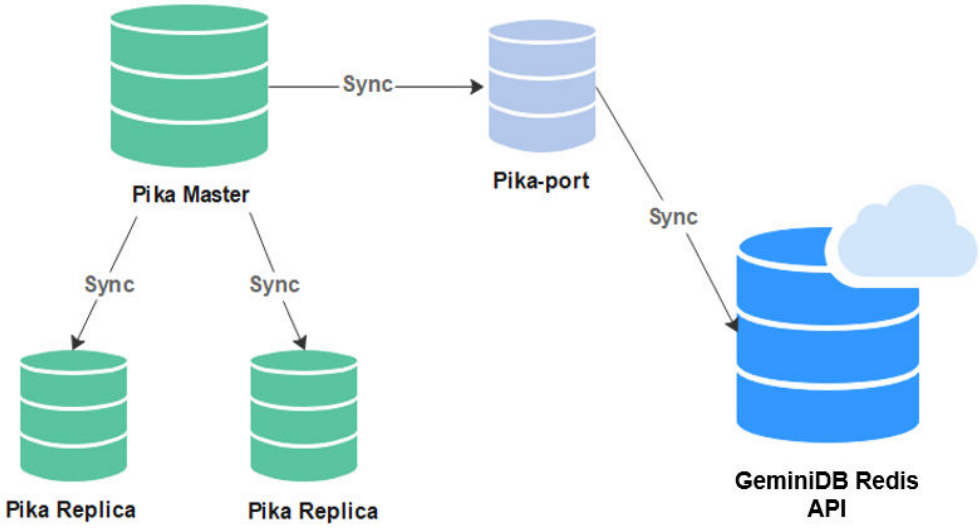
This section describes how to migrate data from SSDB to GeminiDB Redis API.

Migration Principles

ssdb-port acts as a slave node (replica) of the master node of the source SSDB database and migrates data through master/slave replication. Then, it parses and converts the obtained data into the format supported by Redis, and sends the data to the Redis instance specified in the configuration file. The following figure shows

the migration process. After the full synchronization is complete, the new data in SSDB is also synchronized to the Redis instance.

Figure 4-78 Migration diagram



Precautions

- As the slave node of the SSDB master node, ssdb-port reads only full and incremental data without damaging your data.
- The performance of the source SSDB is affected for running ssdb-port.
- Full migration and incremental migration can be performed without service interruption. After all data is migrated, services need to be stopped for a short period of time.

Prerequisites

Create an ECS in the VPC where the GeminiDB Redis instance is located and deploy the migration tool ssdb-port to ensure that the source SSDB instance can communicate with the target GeminiDB Redis instance.

Procedure

To migrate data from SSDB to GeminiDB Redis API, choose **Service Tickets > Create Service Ticket** in the upper right corner of the management console and contact customer service.

Verifying Data Consistency After Migration

After the migration is complete, you can check data consistency.

NOTICE

- Data has been migrated from Redis, or incremental migration has started.
- Redis-full-check must be deployed on the ECS, and the ECS is connected to the source and destination databases.
- During incremental migration, data may be inconsistent due to network latency between the source and destination databases. You are advised to stop writing data to the source and then verify data consistency.
- When Redis is used, an expiration time is usually set for keys. During migration, setting a key expiration time affects data consistency. Data may be inconsistent due to inconsistent expiration time.
- During migration, DTS writes temporary probing keys to Redis on the destination database. Non-service data may be detected during data verification, which is normal.

Procedure

Step 1 Log in to the ECS and ensure it is connected to the source and destination Redis databases.

Step 2 Deploy **redis-full-check**.

Step 3 Verify data.

/redis-full-check -s {*Source IP address***}:{***Source port***} -p {***Source password***} -t {***Destination IP address***}:{***Destination port***} -a {***Destination password***} -m 1**

Table 4-36 Parameter description

Parameter	Description	Example Value
-s	Source Redis database address and port number	-s 10.0.0.1:6379
-p	Password of the source Redis database	-
-t	Destination GeminiDB Redis database address and port number	-t 10.0.0.2:6379
-a	Password of the destination GeminiDB Redis database	-

Parameter	Description	Example Value
-m	Verification mode: 1. All key-value pairs 2. Value length only 3. Key integrity only 4. All key values are verified, but only the length of big keys is verified. By default, the second verification mode is used.	-m 1
-q	Maximum QPS. The default value is **15000**.	-q 5000
-d	Name of the file for saving the verification result. The default value is **result.db**.	-d result.db

Step 4 View the verification result file.

By default, three rounds of verification are performed and three verification result files are generated. Generally, you only need to view the last verification result file.

- Run the **sqlite3 result.db.3** command.
- Run the **select * from key** command.
- Check whether there are abnormal keys.

```
Enter ".help" for usage hints.
sqlite> select * from key;
1|b|string|lack_target|0|1|0
2|c|string|lack_target|0|1|0
3|a|string|lack_target|0|1|0
sqlite>
```

----End

Migration Performance Reference

- Environment: The source SSDB and ssdb-port are deployed on an ECS with 4 vCPUs and 16 GB memory. The destination is a three-node instance with 8 vCPUs and 16 GB memory.
- Preset data: Use the memtier_benchmark tool to preset 100 GB of data.
- Migration performance: about 3000 QPS.

4.4.11 From LevelDB to GeminiDB Redis API

LevelDB is an open-source, persistent, and single-node KV database engine. It provides high random write performance and sequential read/write performance,

and applies to write intensive scenarios. LevelDB does not provide the C/S network structure and must be deployed on the same server as your services. Compared with RocksDB developed based on LevelDB, LevelDB has many disadvantages. For example, it cannot make the most out of multi-core servers, and does not support TB-level data storage, and cannot read data from HDFS.

GeminiDB Redis API uses RocksDB as the storage engine. It is compatible with the Redis protocol and provides various data types to meet LevelDB requirements. In addition, RocksDB has been customized to allow storage to be scaled in seconds, making it easy to migrate LevelDB workloads to the Redis ecosystem. You do not need to migrate data during scaling.

This section describes how to migrate data from LevelDB to GeminiDB Redis API.

Migration Principles

- Use the self-developed migration tool leveldb-port to deploy LevelDB on the same server as your services, prepare the configuration file, and start the migration task to automatically complete full and incremental migration.
- The full migration process is efficient. It takes a snapshot of the LevelDB data, scans the entire database, packs the data into a format that can be identified by GeminiDB Redis API, and then sends the data to GeminiDB Redis API.
- During incremental migration, the WAL file of LevelDB and the LevelDB operations are parsed, and the keys in the WAL file are sharded and sent by multiple threads.

Precautions

- The migration tool needs to be deployed on the source DB, which consumes certain performance. You can modify the configuration file to control the performance.
- During the migration, the source data file of LevelDB is read-only. There is no risk of data damage.
- Services do not need to be stopped during the migration.
- If a fault occurs during the migration, clear the GeminiDB Redis instance and restart the migration.

Procedure

To migrate data from LevelDB to GeminiDB Redis API, choose **Service Tickets > Create Service Ticket** in the upper right corner of the management console and contact customer service.

Migration Performance Reference

- Environment: The source LevelDB and leveldb-port are deployed on a Huawei Cloud ECS with 4 vCPUs and 16 GB memory. The target DB is a three-node GeminiDB Redis instance with 2 vCPUs and 8 GB memory.
- Full migration: 10 GB data is preconfigured, and the migration speed is about 8 MB/s.
- Incremental migration: Set the value to 1 KB and the migration speed to 7,000 QPS.

4.4.12 From Kvrocks to GeminiDB Redis API

RocksDB is a persistent key-value store, single-node DB engine developed by Facebook based on LevelDB. It has powerful sequential read/write and random write performance. Compared with LevelDB, RocksDB has many optimizations. Its performance is greatly improved and the problem that LevelDB proactively restricts write operations is solved. As a DB engine, RocksDB does not provide the C/S network structure. It must be deployed on the same server as your services.

GeminiDB Redis API uses RocksDB as the storage engine and is compatible with the Redis protocol, meeting the usage requirements of RocksDB. In addition, RocksDB has been customized to allow storage to be scaled in seconds, making it easy to migrate RocksDB workloads to the Redis ecosystem. You do not need to migrate data during scaling.

This section describes how to migrate data from RocksDB to GeminiDB Redis API.

Migration Principles

- Use the self-developed migration tool rocksdb-port to deploy RocksDB on the same server as your services, prepare the configuration file, and start the migration task to automatically complete full and incremental migration.
- The full migration process is efficient. It takes a snapshot of the RocksDB data, scans the entire database, packs the data into a format that can be identified by GeminiDB Redis API, and then sends the data to GeminiDB Redis API.
- During incremental migration, the WAL file of RocksDB and the RocksDB operations are parsed, and the keys in the WAL file are sharded and sent by multiple threads.

Precautions

- The migration tool needs to be deployed on the source DB, which consumes certain performance. You can modify the configuration file to control the performance.
- During the migration, the source data file of RocksDB is read-only. There is no risk of data damage.
- Services do not need to be stopped during the migration.
- If a fault occurs during the migration, clear the GeminiDB Redis instance and restart the migration.

Procedure

To migrate data from RocksDB to GeminiDB Redis API, choose **Service Tickets > Create Service Ticket** in the upper right corner of the management console and contact customer service.

4.4.13 Migration from an AWS ElastiCache for Redis Database to a GeminiDB Redis Instance

Migration Principles

After backing up and exporting the RDB file in an AWS ElastiCache for Redis database, you can use Redis-Shake to restore data to a GeminiDB Redis instance.

Usage Notes

- AWS does not support the **psync/sync** command and data cannot be incrementally migrated.
- Before the migration, ensure that the network between the ECS where Redis-shake is deployed and the destination GeminiDB Redis is normal.
- Ensure that the security group configuration on the source and target ends is enabled.

Procedure

Step 1 Deploy the required migration tool.

1. Obtain **Redis-Shake**.

   NOTE

   Download the Redis-Shake release package and decompress it.

2. Modify the **Redis-Shake.conf** configuration file and configuring the following items:

   log.level = info #Default log level. A printed INFO log contains migration progress information, based on which you can judge whether the migration is complete.

   source.rdb.input = /xx/xx.rdb # Absolute path of the source RDB file

   target.address = <host>:6379 # Destination instance IP address

   target.password_raw = *** # Password for logging in to a destination instance

   target.version = 5.0 # Version of the destination Redis instance

   target.type = standalone # Destination instance type

   target.db = 0 #Data is migrated to the specified database of the destination GeminiDB Redis. The default value is **db0**.

   big_key_threshold = 1 #Setting the big key threshold

3. Specify whether data of the destination is overwritten.

   key_exists = none

> **NOTE**
>
> If there are duplicate keys on the source and destination, specify whether data of the destination is overwritten. The options are as follows:
>
> - **rewrite** indicates that the source overwrites the destination.
> - **none** indicates that the migration process exists once duplicate keys are detected.
> - **ignore** indicates that keys in the source are retained and keys in the destination are ignored. This value does not take effect in rump mode.
>
> **none** is recommended. There will be no duplicate data because the source is an RDB file. If the migration exits unexpectedly, you can choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service.

Step 2 Migrate data.

Migration starting command:

./redis-shake.linux -conf=redis-shake.conf -type=restore

> **NOTE**
>
> Use the restore mode because the source is an RDB file.

Stop the migration process after the migration is complete.

Step 3 Verify data.

Data is obtained from the RDB file. Therefore, you need to check the GeminiDB Redis data at the destination end from the service perspective.

----End

Verifying Data Consistency After Migration

After the migration is complete, you can check data consistency.

NOTICE

- Data has been migrated from Redis, or incremental migration has started.
- Redis-full-check must be deployed on the ECS, and the ECS is connected to the source and destination databases.
- During incremental migration, data may be inconsistent due to network latency between the source and destination databases. You are advised to stop writing data to the source and then verify data consistency.
- When Redis is used, an expiration time is usually set for keys. During migration, setting a key expiration time affects data consistency. Data may be inconsistent due to inconsistent expiration time.
- During migration, DTS writes temporary probing keys to Redis on the destination database. Non-service data may be detected during data verification, which is normal.

Procedure

Step 1 Log in to the ECS and ensure it is connected to the source and destination Redis databases.

Step 2 Deploy **redis-full-check**.

Step 3 Verify data.

/redis-full-check -s {*Source IP address***}:{***Source port***} -p {***Source password***} -t {***Destination IP address***}:{***Destination port***} -a {***Destination password***} -m 1**

Table 4-37 Parameter description

Parameter	Description	Example Value
-s	Source Redis database address and port number	-s 10.0.0.1:6379
-p	Password of the source Redis database	-
-t	Destination GeminiDB Redis database address and port number	-t 10.0.0.2:6379
-a	Password of the destination GeminiDB Redis database	-
-m	Verification mode: 1. All key-value pairs 2. Value length only 3. Key integrity only 4. All key values are verified, but only the length of big keys is verified. By default, the second verification mode is used.	-m 1
-q	Maximum QPS. The default value is **15000**.	-q 5000
-d	Name of the file for saving the verification result. The default value is **result.db**.	-d result.db

Step 4 View the verification result file.

By default, three rounds of verification are performed and three verification result files are generated. Generally, you only need to view the last verification result file.

- Run the **sqlite3 result.db.3** command.
- Run the **select * from key** command.
- Check whether there are abnormal keys.


```
Enter ".help" for usage hints.
sqlite> select * from key;
1|b|string|lack_target|0|1|0
2|c|string|lack_target|0|1|0
3|a|string|lack_target|0|1|0
sqlite>
```

----End

4.4.14 Verifying Redis Data Consistency After Migration

After the migration is complete, you can check the consistency of Redis data.

Constraints

- The Redis migration has been completed, or the incremental migration has started.
- The Redis-Full-Check open-source tool must be deployed on the ECS instance, and the ECS instance can communicate with the source and target networks.
- If the migration task is in the incremental state, data consistency cannot be ensured due to network latency between the source and target ends. If conditions permit, you are advised to stop writing data to the source end and then perform the verification.
- When Redis is used, an expiration time is usually set for keys. During migration, setting a key expiration time affects data consistency. If verification results show that data is inconsistent, the possible cause is that the key expiration time is inconsistent.
- During the migration, DTS writes temporary probing keys to Redis on the destination end. Non-service data may be detected during data verification, which is normal.

Procedure

Step 1 Log in to the ECS and ensure that the ECS can connect to the source and destination Redis databases.

Step 2 Deploy **Redis-Full-Check**.

Step 3 Verify data.

/redis-full-check -s {*Source IP address***}:{***Source port***} -p {***Source password***} -t {***Destination IP address***}:{***Destination port***} -a {***Destination password***} -m 1**

Table 4-38 Parameter description

Parameter	Description	Example Value
-s	Source Redis connection address and port number	-s 10.0.0.1:6379
-p	Password of the source Redis database.	-

Parameter	Description	Example Value
-t	Destination Redis connection address and port number	-t 10.0.0.2:6379
-a	Password of the destination Redis database.	-
-m	Verification mode: 1. Verify all key-value pairs. 2. Only value length is verified. 3. Only key integrity is verified. 4. All key values are verified, but only the length of big keys is verified. By default, the second verification mode is used.	-m 1
-q	Maximum QPS. The default value is **15000**.	-q 5000
-d	Name of the file for saving the verification result. The default value **is result.db**.	-d result.db

Step 4 View the verification result file.

By default, three rounds of verification are performed and three verification result files are generated. Generally, you only need to view the last verification result file.

- Run the **sqlite3 result.db.3** command.
- Run the **select * from key** command.
- Check whether abnormal keys exist.

```
Enter ".help" for usage hints.
sqlite> select * from key;
1|b|string|lack_target|0|1|0
2|c|string|lack_target|0|1|0
3|a|string|lack_target|0|1|0
sqlite>
```

----End

4.5 Instance Management

4.5.1 Managing Sessions of a GeminiDB Redis Instance

Scenarios

You can manage sessions of a GeminiDB Redis instance on the console.

Usage Notes

- To use this function, you need to update instances with earlier kernel minor versions to the most recent version.
- Redis Cluster instances do not support sessions.
- This function is available only when the target instance node is normal.
- Session details are displayed after a client connects to the instance.

CAUTION

Killing a session will disconnect applications.

Viewing Instance Sessions

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the target instance.

Step 4 In the navigation pane on the left, choose **Sessions**.

Step 5 On the **Sessions** page, select a node to view its session information. **Table 4-39** lists the session parameters.

Figure 4-79 Sessions

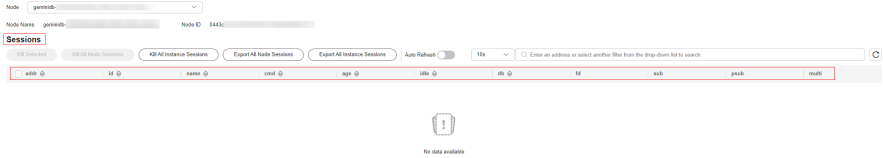


Table 4-39 Instance session parameters

Parameter	Description
addr	IP address and port number of a client
id	Session ID

Parameter	Description
name	Connection name
cmd	Last executed command
age	Connection duration, in seconds
idle	Idle duration, in seconds
db	ID of a database that is being used by the client, for example, DB0, DB1, and DB2
fd	File descriptor for sockets
sub	Number of subscribed channels
psub	Number of subscribed modes
multi	Number of commands executed in a transaction

Step 6 You can select **By source** or **By database**. Table 4-40 lists the parameters.

- **By source**: Sessions on clients connected to a node. IP addresses of top ten clients are displayed and ordered based on sessions on each client. There may be clients with the same quality of sessions.
- **By database**: IP addresses of clients connected to each database and sessions of each client. Top ten clients are displayed and ordered based on their sessions. There may be clients with the same quality of sessions.

Figure 4-80 Session statistics

Table 4-40 Session statistics

Parameter	Description
Item	Total and active clients • Total clients: Total client connections • Active clients: Active client connections
Result	Statistical results
Source	Client IP address
DB	ID of a GeminiDB Redis database, for example, DB0, DB1, and DB2
Total	Total client connections

----End

Auto Refresh

Auto Refresh is disabled by default. You can toggle it on. After **Auto Refresh** is enabled, the page is refreshed every 10s by default. You can set the interval to 10s, 30s, or 60s.

Figure 4-81 Auto Refresh

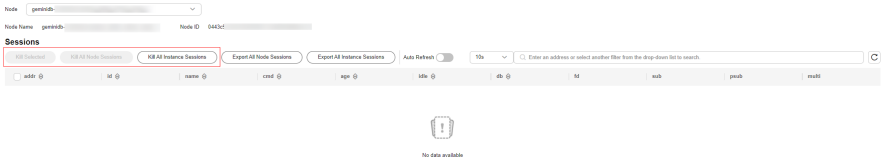
Killing Sessions

You can click **Kill Selected**, **Kill All Node Sessions**, or **Kill All Instance Sessions**.

CAUTION

Killing a session will disconnect applications.

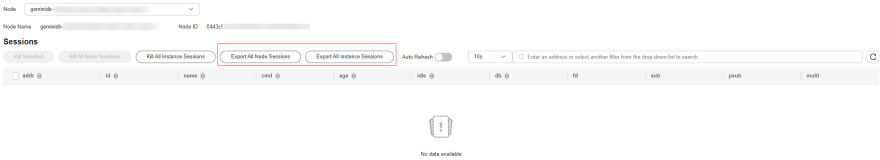
Figure 4-82 Killing sessions



Exporting Sessions

You can click **Export All Node Sessions** or **Export All Instance Sessions**.

Figure 4-83 Exporting sessions



4.5.2 Renaming Commands of a GeminiDB Redis Instance

Scenarios

Commands of a general-purpose GeminiDB Redis instance can be renamed. To prevent data loss, instance restart, and performance jitter caused by misoperations, you can rename high-risk commands.

Precautions

- To use the command renaming function, you need to update instances with earlier kernel minor versions to the most recent version.
- High-risk commands can be renamed only when your instance is in the **Available** state.
- Commands of a Redis Cluster instance cannot be renamed.

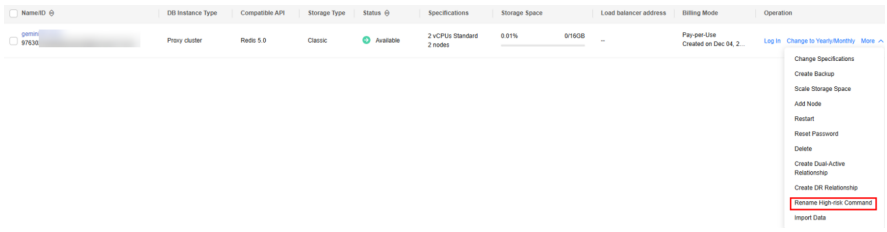
Procedure

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the target instance and choose **More** > **Rename High-risk Command** in the **Operation** column.

Figure 4-84 Command renaming



Step 4 Select the required commands and disable them. For involved parameters, see Table 4-41.

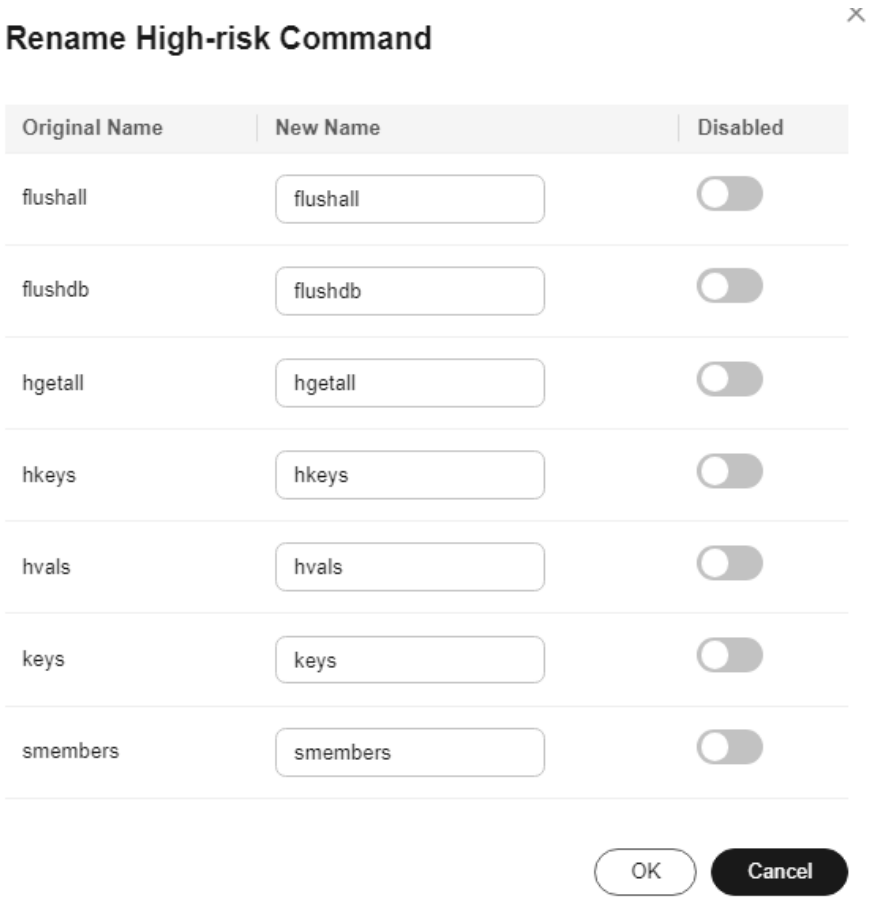
Table 4-41 Parameter description

Parameter	Description
flushall	Clears all buffers.
flushdb	Deletes all keys in the current database.
hgetall	Returns all fields and values in the hash table.
hkeys	Returns all keys in the hash table.
hvals	Returns all values in the hash table.
keys	Finds all keys that match a given pattern.
smembers	Returns all members in a set. Any set without keys is considered empty.
New Name	Name of the command that takes effect currently. The name can include 0 to 30 characters. Command names are case-insensitive and can contain only letters, digits, hyphens (-), and underscores (_). If the name is left blank, the command is disabled. The new name must be unique.

Parameter	Description
Disabled	The **Disabled** option is toggled off by default. You can click  to toggle on it. If a command is disabled, its new name is empty.

Step 5 Modify parameter information and click **OK**.

Figure 4-85 Renaming commands



Step 6 Check the renaming result.

- You can view new command names on the **Rename High-risk Command** page.
- After the renaming is complete, original commands become invalid and you need to use new commands to perform operations.

----End

4.5.3 Clearing GeminiDB Redis Instance Data

Scenarios

GeminiDB Redis API allows you to clear all data in an instance or data in a specified database to release instance space.

Precautions

- Cleared data cannot be restored. Back up the instance before you clear it. For details, see **Creating a Manual Backup**.
- After you select **Specified DB**, only the data in the selected database is cleared.

Procedure

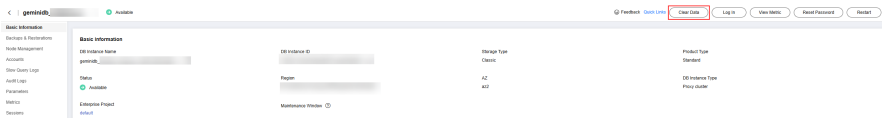
Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate a target instance and click its name.

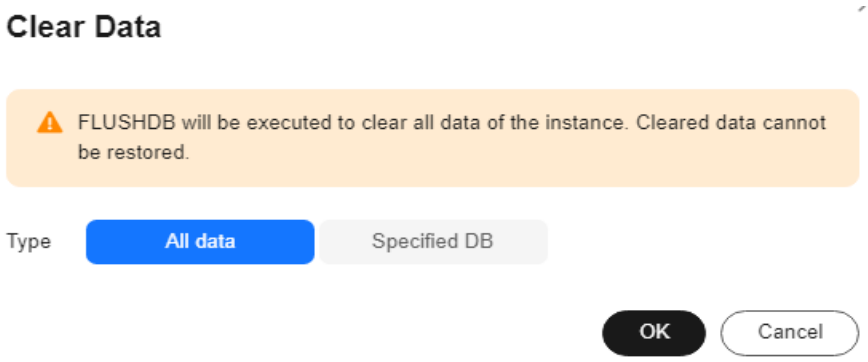
Step 4 In the upper right corner of the **Basic Information** area, click **Clear Data**.

Figure 4-86 Clearing data



- If you want to clear all data in the instance, select **All** and click **OK**.

Figure 4-87 Clearing all data

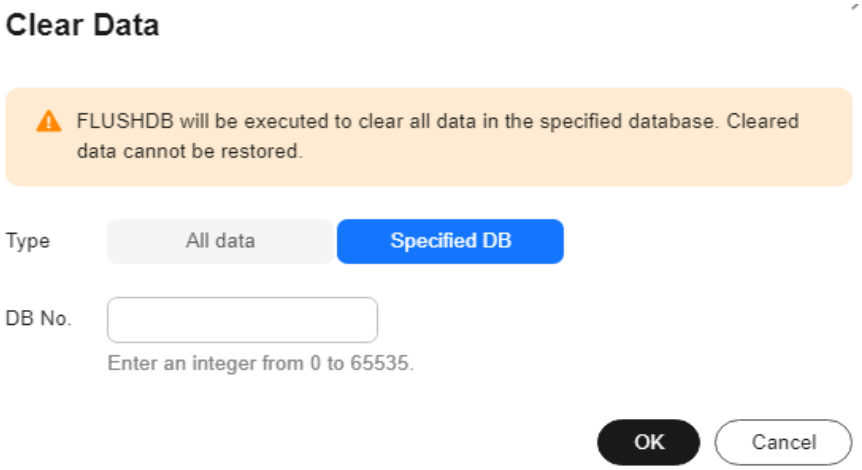


- If you need to clear data of a specified database, select **Specified DB**, enter the number of the target database, and click **OK**.

  NOTE

  The number of the database must be an integer ranging from 0 to 65535.

Figure 4-88 Clearing data of a specified database



----End

4.5.4 Instance Lifecycle Management

4.5.4.1 Restarting an Instance

Scenarios

You may need to restart an instance for routine maintenance.

Precautions

- Only instances in states **Available**, **Abnormal**, or **Checking restoration** can be restarted.
- After you restart an instance, all nodes in the instance are also restarted.
- Restarting an instance will interrupt services. Wait until off-peak hours and ensure that your application can re-connect.
- If you enable operation protection, two-factor authentication is required for sensitive operations to secure your account and cloud products. For details about how to enable operation protection, see ***Identity and Access Management User Guide***.

Procedure

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the instance you want to restart and in the **Operation** column choose **Restart** or **More** > **Restart**.

Alternatively, locate the instance you want to restart and click its name. On the displayed **Basic Information** page, click **Restart** in the upper right corner of the page.

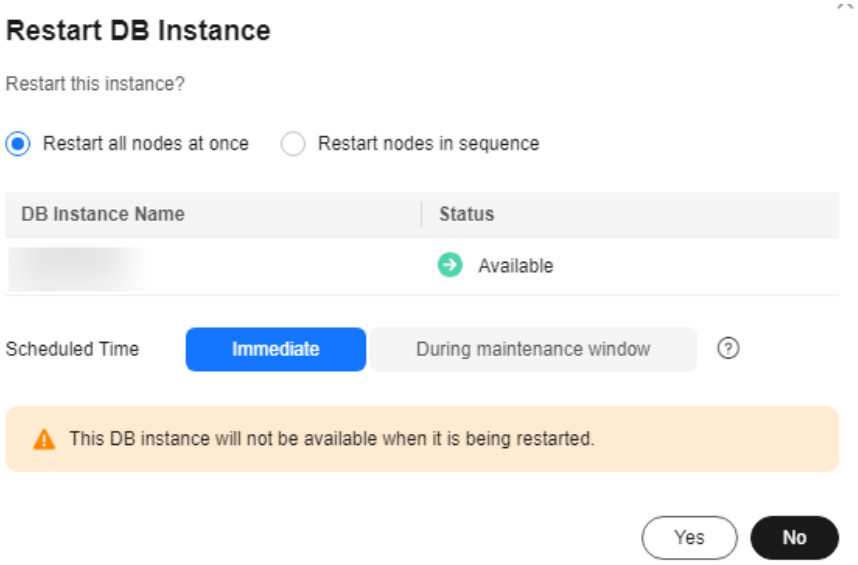
Step 4 If you have enabled operation protection, click **Start Verification** in the **Restart DB Instance** dialog box. On the displayed page, click **Send Code**, enter the verification code, and click **Verify**. The page is closed automatically.

Step 5 In the displayed dialog box, click **Yes** or **Immediate**.

- Instance in classic deployment mode

  For GeminiDB Redis instances in classic deployment mode, you can restart several nodes at the same time or in sequence based on service requirements.

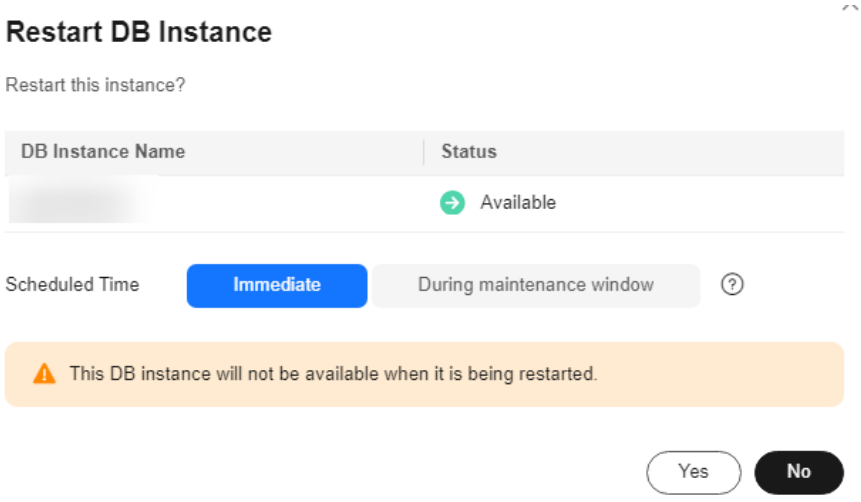
  Figure 4-89 Restarting an instance



- Instance in cloud native deployment mode

  For GeminiDB Redis instances deployed in cloud native mode, click **Yes** or **Immediate**.

  Figure 4-90 Restarting an instance



----End

4.5.4.2 Exporting Instance Information

Scenarios

You can export information about all or selected instances to view and analyze instance information.

Exporting All Instance Information

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click  in the upper right corner of the page. By default, information about all DB instances are exported. In the displayed dialog box, you can select the items to be exported and click **Export**.

Step 4 After the export task is complete, check an XLS file is generated locally.

----End

Exporting Information About Selected Instances

Step 1 On the **Instances** page, select the instances that you want to export or search for required instances by project, compatible API, name, ID, or tag and click  in the upper right corner of the page. In the displayed dialog box, select the items to be exported and click **Export**.

Step 2 After the export task is complete, check an XLS file is generated locally.

----End

4.5.4.3 Deleting a Pay-per-Use Instance

Scenarios

You can choose to delete a pay-per-use instance on the **Instances** page based on service requirements. To delete a yearly/monthly DB instance, you need to unsubscribe from the order. For details, see **2.11.4 How Do I Unsubscribe from Yearly/Monthly Instances?**.

Precautions

- Instances that an operation is being performed on cannot be deleted. They can be deleted only after the operations are complete.
- If a pay-per-use instance is deleted, its automated backups will also be deleted and you will no longer be billed for them. Manual backups, however, will be retained and generate additional costs.
- After an instance is deleted, all its data and automated backups are automatically deleted as well and cannot be recovered. You are advised to

create a backup before deleting an instance. For details, see **Creating a Manual Backup**.

- After you delete an instance, all of its nodes are deleted.
- A deleted instance will be retained in the recycle bin for a period of time after being released, so you can rebuild the instance and restore data from it.

Procedure

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the instance that you want to delete and in the **Operation** column choose **Delete** or **More** > **Delete**.

Step 4 If you have enabled operation protection, click **Start Verification** in the **Delete DB Instance** dialog box. On the displayed page, click **Send Code**, enter the verification code, and click **Verify**. The page is closed automatically.

NOTE

If you enable operation protection, two-factor authentication is required for sensitive operations to secure your account and cloud products. For details about how to enable operation protection, see ***Identity and Access Management User Guide***.

Step 5 In the displayed dialog box, click **Yes**.

Deleted instances are not displayed in the instance list any longer.

----End

4.5.4.4 Recycling an Instance

Unsubscribed yearly/monthly instances and deleted pay-per-use instances are moved to the recycle bin and can be restored.

Precautions

- The recycling bin is enabled by default and cannot be disabled. Instances in the recycle bin are retained for 7 days by default, and this will not incur any charges.
- You can put up to 100 instances into the recycle bin. If the maximum number of instances is reached, you cannot put instances into the recycle bin any more.
- If you delete an instance of full storage, the deleted instance will not be moved to the recycle bin.

Modifying the Recycling Policy

NOTICE

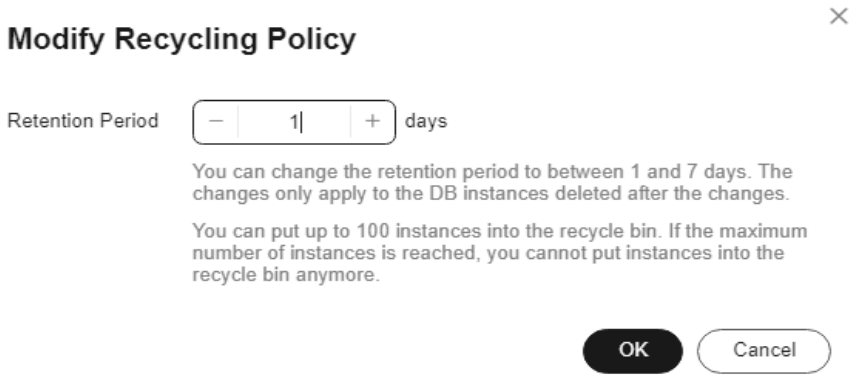
You can modify the retention period, and the new retention period only takes effect for the instances that are deleted after the modification.

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Recycling Bin** page, click **Modify Recycling Policy**. In the displayed dialog box, set the retention period from 1 day to 7 days. Then, click **OK**.

Figure 4-91 Modifying the recycling policy



----End

Rebuilding an Instance

You can rebuild instances from the recycle bin within the retention period to restore data.

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Recycling Bin** page, locate the instance that you want to rebuild and click **Rebuild** in the **Operation** column.

Figure 4-92 Rebuilding an instance

Step 4 On the displayed page, set required parameters (you are advised to set the specifications to be the same as those of the original instance) and submit the rebuilding task.

----End

4.6 Modifying Instance Settings

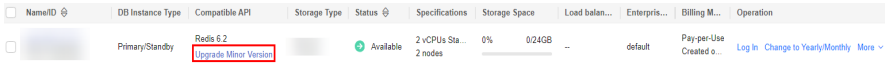
4.6.1 Upgrading a Minor Version

GeminiDB Redis API can be upgraded by installing patches to improve performance, release new features, or fix bugs.

After a new patch version involving performance improvement, new functions, or problem rectification is released, you can upgrade your instance to the latest version at a proper time based on service requirements.

If a new patch is released, you can upgrade your instance by clicking the upgrade button in the **Compatible API** column on the **Instances** page.

Figure 4-93 Patch installation



If the kernel version of your instance has potential risks or major defects, has expired, or has been brought offline, the system will notify you by SMS message or email and deliver an upgrade task during maintenance.

Precautions

- Upgrade your instance once there is a new patch released.
- If the database version is a risky version, the system prompts you to upgrade the database patch.
- Upgrading the minor version of an instance will restart each node of the instance in sequence. When a node is being restarted, its services will be taken over by another node. Each takeover will interrupt services for 3 to 5 seconds. So, perform an upgrade during off-peak hours and enable automatic reconnection so that each node can be reconnected immediately after being restarted.
- Upgrading basic components takes about 15 minutes. Upgrading data components takes about 1 to 2 minutes, which depends on how many nodes there are.
- The system automatically checks the minor version of the instance. If the **Upgrade Minor Version** button does not exist on the console, the current minor version is the latest version.

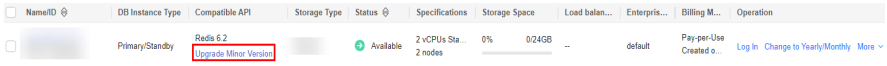
Procedure

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

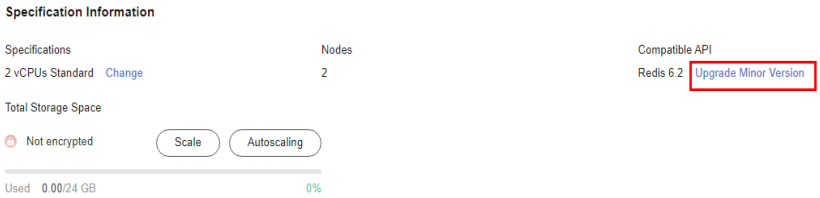
Step 3 On the **Instances** page, locate the instance you want to upgrade and click **Upgrade Minor Version** in the **Compatible API** column.

Figure 4-94 Patch installation



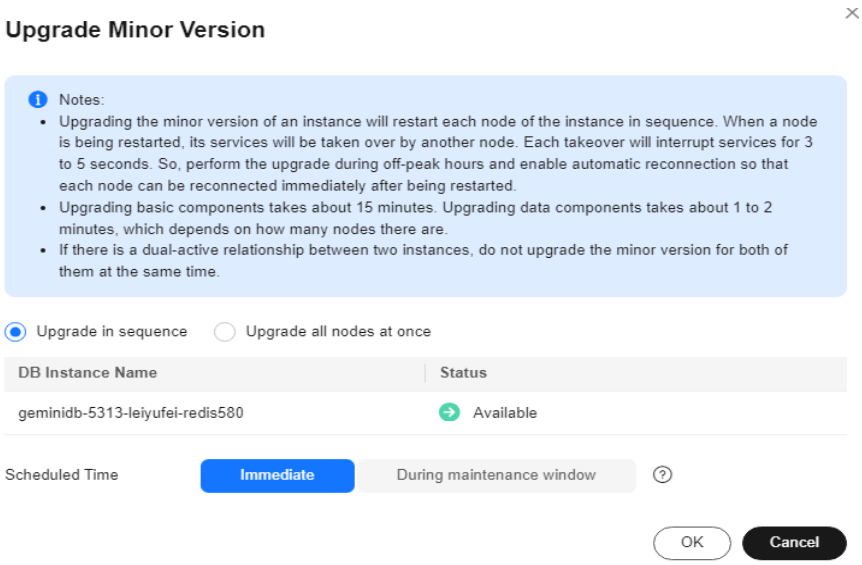
Alternatively, click the instance name to go to the **Basic Information** page. In the **Specification Information** area, click **Upgrade Minor Version** in the **Compatible API** field.

Figure 4-95 Patch installation



Step 4 In the displayed dialog box, click **OK**.

Figure 4-96 Confirming dialog box



Step 5 View the upgrade result on the **Instances** page.

- When the upgrade is ongoing, the instance status is **Upgrading minor version**.
- After the upgrade is complete, the instance status changes to **Available**.

----End

4.6.2 Modifying a GeminiDB Redis Instance Name

Scenarios

This section describes how to modify the name of a GeminiDB Redis instance.

Method 1

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the instance whose name you want to modify and click to the right of the instance.

- To submit the change, click **OK**.

- To cancel the change, click **Cancel**.

NOTE

The instance name:

- Can be the same as an existing instance name.
- Can include 4 to 64 bytes and must start with a letter. It is case-sensitive and allows only letters, digits, hyphens (-), and underscores (_).

Step 4 View the results on the **Instances** page.

----End

Method 2

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the instance whose name you want to modify and click its name.

Step 4 In the **Instance Information** area on the **Basic Information** page, click  in the **DB Instance Name** field.

- To submit the change, click  .
- To cancel the change, click  .

NOTE

The instance name:

- Can be the same as an existing instance name.
- Can include 4 to 64 bytes and must start with a letter. It is case-sensitive and allows only letters, digits, hyphens (-), and underscores (_).

Step 5 Check the results on the **Instances** page.

----End

4.6.3 Changing the Administrator Password of a GeminiDB Redis Database

Scenarios

For security reasons, regularly change your administrator password.

Precautions

- You can reset the administrator password only when the **instance status** is **Available**, **Backing up**, or **Scaling up**.
- If you enable operation protection, two-factor authentication is required for sensitive operations to secure your account and cloud products. For details about how to enable operation protection, see ***Identity and Access Management User Guide***.

Method 1

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the instance whose administrator password you want to reset and choose **More** > **Reset Password** in the **Operation** column.

Step 4 Enter and confirm the new administrator password and click **OK**.

The password must be 8 to 32 characters in length and contain any two of uppercase letters, lowercase letters, digits, and the following special characters: ~!@#%^*-_=+?$()&

Step 5 If you have enabled operation protection, click **Start Verification** in the displayed dialog box. On the displayed page, click **Send Code**, enter the verification code, and click **Verify**. The page is closed automatically.

----End

Method 2

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the instance whose administrator password you want to reset and click its name. The **Basic Information** page is displayed.

Step 4 In the **DB Information** area, click **Reset Password** in the **Administrator** field.

Step 5 Enter and confirm the new administrator password and click **OK**.

The password must be 8 to 32 characters in length and contain any two of uppercase letters, lowercase letters, digits, and the following special characters: ~!@#%^*-_=+?$()&

Step 6 If you have enabled operation protection, click **Start Verification** in the displayed dialog box. On the displayed page, click **Send Code**, enter the verification code, and click **Verify**. The page is closed automatically.

----End

4.6.4 Changing the CPU and Memory Specifications of an Instance

Scenarios

You can increase or decrease the CPU or memory of all nodes in an instance. You can change the vCPU and memory specifications of your instance to meet your service requirements. If an instance is overloaded and compute resources need to be added urgently, you are advised to add compute nodes first.

Precautions

- During online specification change, second-level intermittent disconnection occurs once when the change is performed on a single node. Therefore, the

entire instance is intermittently disconnected for several times. The client must have an automatic reconnection mechanism. You are advised to perform the specification change during off-peak hours.

- For a node whose specifications are being changed, its computing tasks are handed over to other nodes. Change specifications of nodes during off-peak hours to prevent the instance from overload.

Procedure

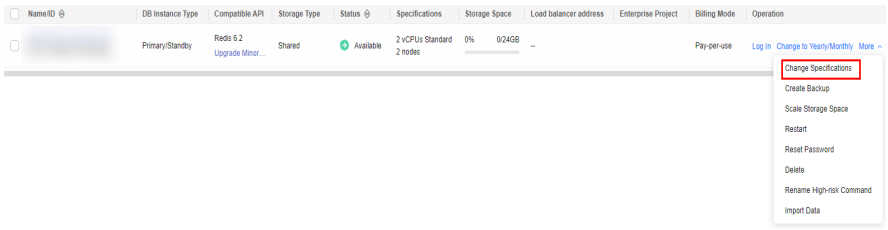
Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the instance whose specifications you want to change and click **More > Change Specifications** in the **Operation** column.

- Instance in classic deployment mode

Figure 4-97 Changing specifications



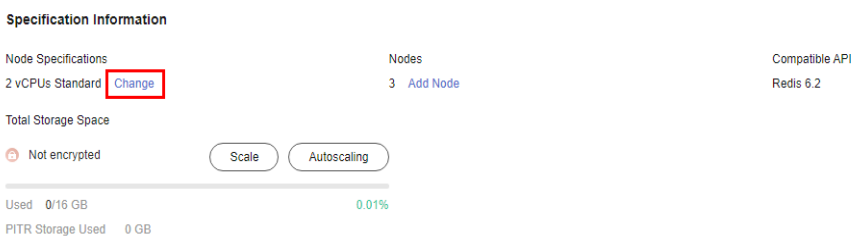
- Instance in cloud native deployment mode

Figure 4-98 Changing specifications

In the **DB Information** area on the **Basic Information** page, click **Change** next to the **Specifications** field.

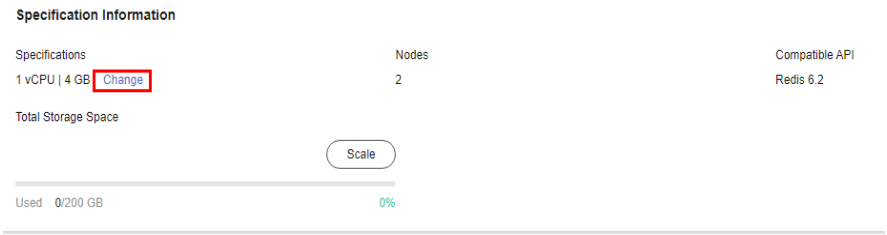
- Instance in classic deployment mode

Figure 4-99 Changing specifications



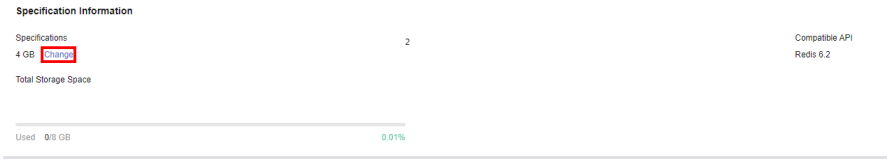
- Capacity-oriented instance in cloud native deployment mode

Figure 4-100 Changing specifications



- Standard instance in cloud native deployment mode

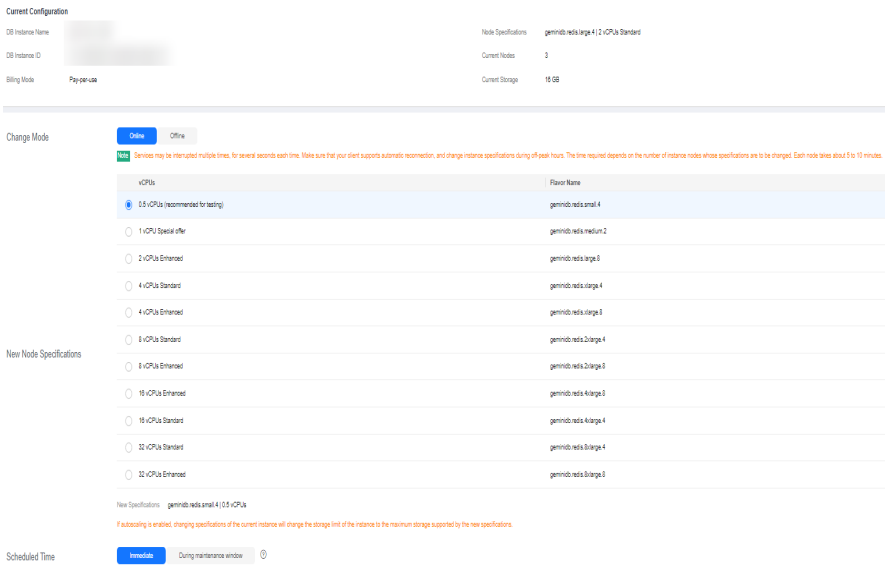
Figure 4-101 Changing specifications



Step 4 On the displayed page, select a specification change mode and required specifications, and click **Next**.

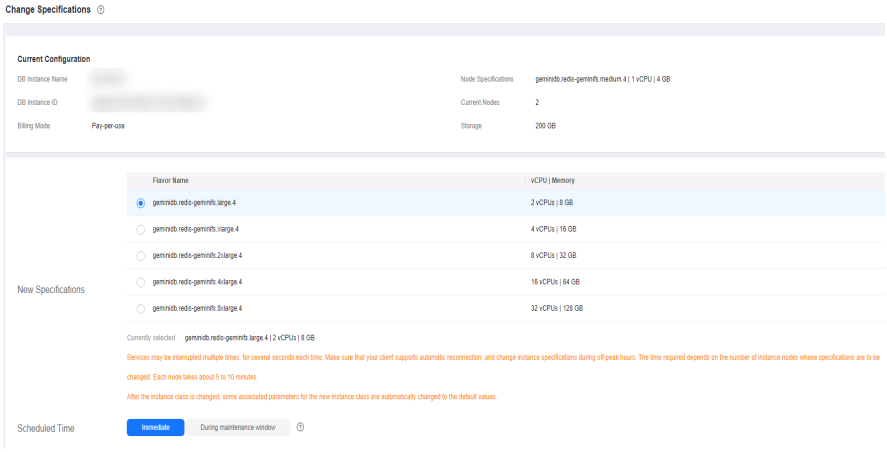
- Online change: During the change, instance nodes are upgraded in rolling mode, which has the minimum impact on services. The change duration is positively related to the number of nodes. Each node takes about 5 to 10 minutes. If there are a large number of nodes, wait patiently.
- Offline change: During offline change, all nodes are changed concurrently, which interrupts services for about 10 to 20 minutes. Exercise caution when performing this operation. For your online production services, you are advised to perform the change online.
- Instance in classic deployment mode

Figure 4-102 Changing specifications



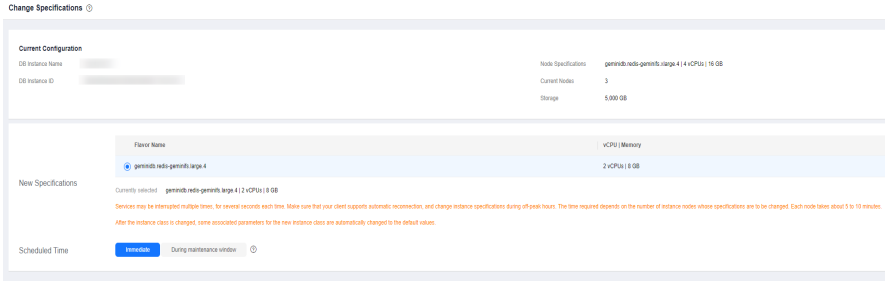
- Capacity-oriented instance in cloud native deployment mode

Figure 4-103 Changing specifications



- Standard instance in cloud native deployment mode

Figure 4-104 Changing specifications



Step 5 On the displayed page, confirm the specifications.

- If you need to modify your settings, click **Previous**.
- If you do not need to modify your settings, click **Submit** .

Step 6 View the results.

Go to the **Basic Information** page and in the **DB Information** area you can see the new instance specifications.

----End

4.6.5 Setting a Maintenance Window

The default maintenance window is 10:00–14:00 (GMT+08:00) but you can change it if needed. To prevent service interruption, set the maintenance window to off-peak hours. Before calling this API:

Usage Notes

- You can configure a maintenance window only for restarting a DB instance, changing an instance class, or upgrading the minor version of a DB instance.
- The specification change and patch upgrade that have been performed during the maintenance period cannot be performed immediately. The instance can be restarted immediately.
- You can cancel a task to be executed.

- Changing the maintenance window will not affect the timing that has already been scheduled.
- The maintenance window cannot overlap the time window configured for backups. Otherwise, scheduled tasks may fail.
- During the maintenance window, the scheduled task is scanned and executed every 10 minutes. If the task is delivered near the end of the maintenance period, the task may fail to be scanned and the execution is canceled.

Setting the Maintenance Period

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the instance whose specifications you want to change. The **Basic Information** page is displayed.

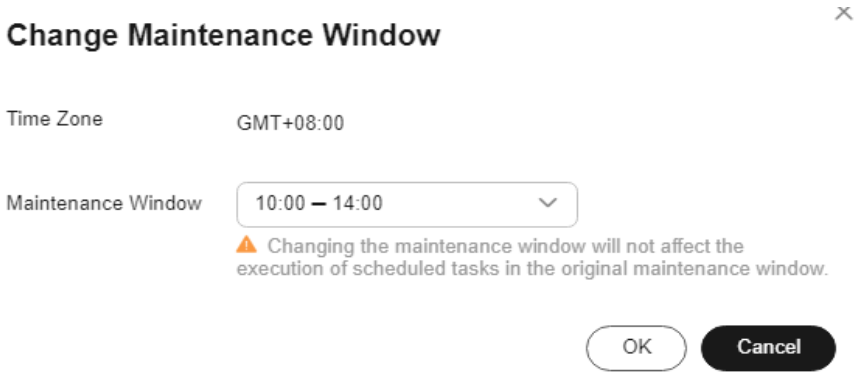
Step 4 On the **Basic Information** page, locate **Maintenance Window** and click **Change**.

Figure 4-105 The change button

Step 5 On the **Change Maintainable Window** page, select the maintenance time period as needed, and then click **OK**.

Supported time periods: 02:00-06:00, 06:00-10:00, 10:00-14:00, 14:00-18:00, 18:00-22:00, and 22:00-02:00

Figure 4-106 Changing a maintenance window



Step 6 Check the result.

On the **Basic Information** page, you can view the changed maintenance window.

----End

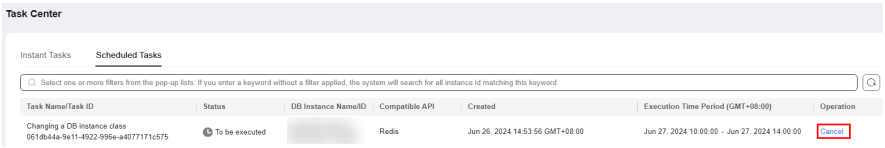
Canceling a Scheduled Task

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Task Center** page, locate a scheduled task, and click **Cancel** in the **Operation** column.

Figure 4-107 Canceling a task



Step 4 View the result.

On the **Task Center** page, you can view the result. After the task is cancelled, its status changes to **Cancelled**.

Figure 4-108 Viewing cancelled tasks

----End

4.6.6 Scaling Instances

4.6.6.1 Overview

After you purchase a GeminiDB Redis instance, resource requirements may change along with workload volumes. You can scale your instance in the following ways.

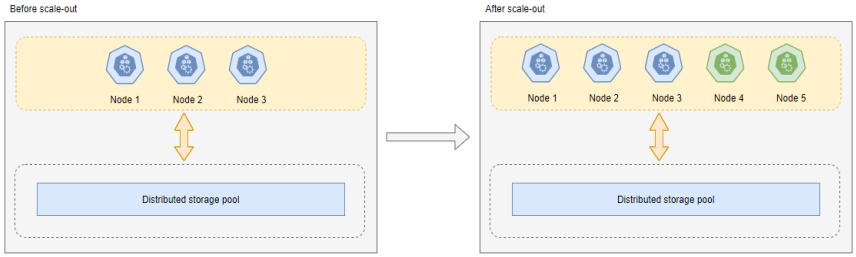
Table 4-42 Scaling methods

Method	Supported Instance Type
4.6.6.2 Adding Instance Nodes	• Proxy cluster • Redis Cluster
4.6.6.3 Adding Instance Shards	Standard cluster
4.6.6.4 Deleting Instance Nodes	• Proxy cluster • Redis Cluster

Adding Instance Nodes

For example, if three nodes have been deployed and two more nodes need to be added, there will be five nodes in total. For details, see **4.6.6.2 Adding Instance Nodes**.

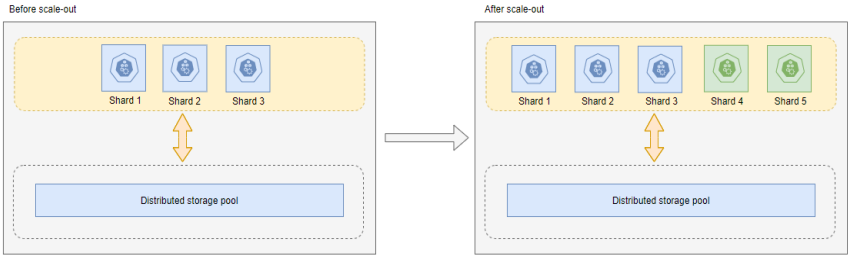
Figure 4-109 Adding instance nodes



Adding Instance Shards

For example, if three shards have been deployed and two more shards need to be added, there will be five shards in total. For details, see **4.6.6.3 Adding Instance Shards**.

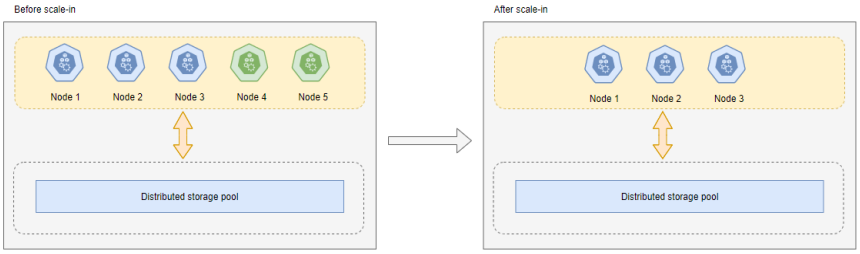
Figure 4-110 Adding instance shards



Deleting Instance Nodes

For example, if five nodes have been deployed and two of them need to be deleted, three nodes will be left. For details, see **4.6.6.4 Deleting Instance Nodes**.

Figure 4-111 Deleting instance shards



4.6.6.2 Adding Instance Nodes

Scenarios

This section describes how to add nodes to an instance to suit your service requirements. You can also delete a node as required. For details, see **4.6.6.4 Deleting Instance Nodes**.

Precautions

- Adding nodes will trigger fast load balancing, which may cause a request timeout for a few seconds. Enable automatic retry for services.
- You can add nodes only when the instance status is **Available** or **Checking restoration**.
- An instance cannot be deleted when one or more nodes are being added.
- If the storage is insufficient, adding nodes is not supported. Expand the storage first. For details about the storage supported by instances of different specifications, see **1.6 Instance Specifications**.
- Nodes cannot be added if any node is stopped.
- Currently, nodes can be added only to proxy cluster and Redis Cluster instances.
- Currently, a maximum of 36 nodes are supported. To obtain more nodes, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service.

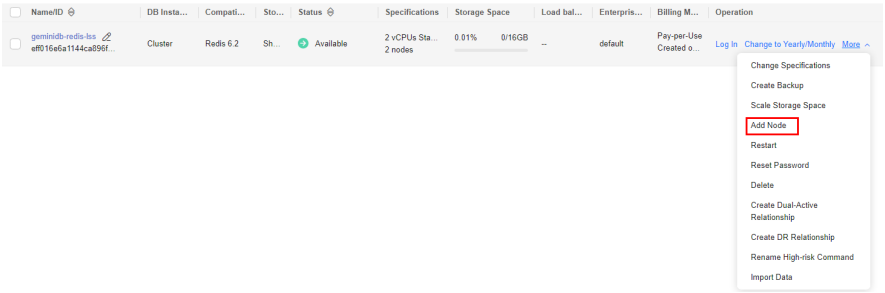
Method 1

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

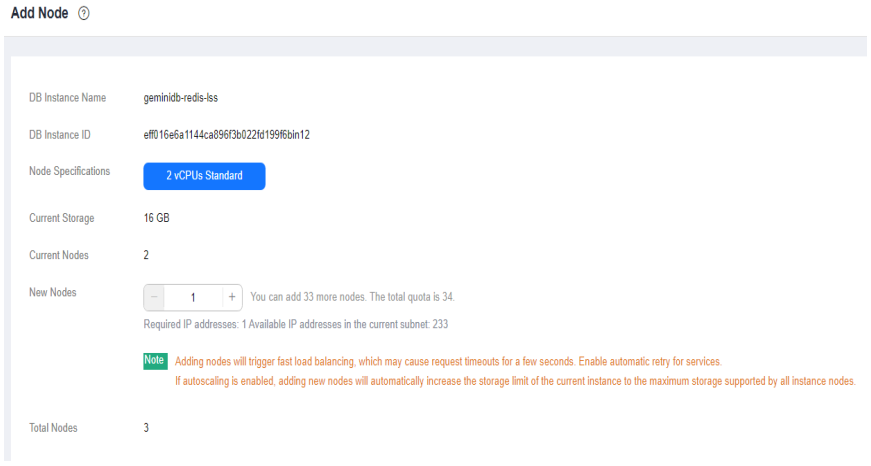
Step 3 On the **Instances** page, locate the instance which you want to add nodes for, click its name, and choose **More** > **Add Node** in the **Operation** column.

Figure 4-112 Adding nodes



Step 4 On the **Add Node** page, specify the number of nodes to be added and view the storage of the instance.

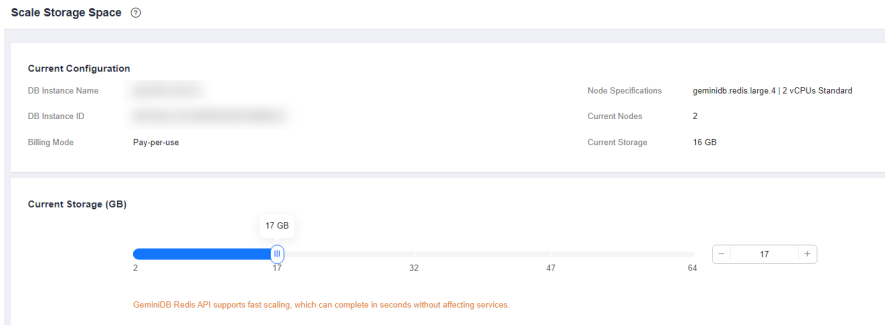
- If the storage capacity is sufficient, click **Next** and go to **Step 8**.
- If the storage capacity is insufficient, click **Next** and go to **Step 5**.



New nodes are of the same specifications as existing nodes. Once a new node is added, its specifications cannot be changed.

Step 5 On the **Scale Storage Space** page, select your target storage capacity and click **Next**.

Figure 4-113 Storage change



Step 6 On the displayed page, confirm the storage space.

- Yearly/Monthly
  - If you need to modify your settings, click **Previous**.
  - If you do not need to modify your settings, click **Next** and complete the payment.
- Pay-per-use
  - If you need to modify your settings, click **Previous**.
  - If you do not need to modify your settings, click **Submit** to scale up the storage space.

Step 7 After the storage is scaled up, go to **5** to add nodes again.

Step 8 On the displayed page, confirm the node configuration details.

- Yearly/Monthly
  - If you need to modify your settings, click **Previous**.
  - If you do not need to modify your settings, click **Submit** and complete the payment.
- Pay-per-use

- If you need to modify your settings, click **Previous**.
- If you do not need to modify your settings, click **Submit**.

----End

Method 2

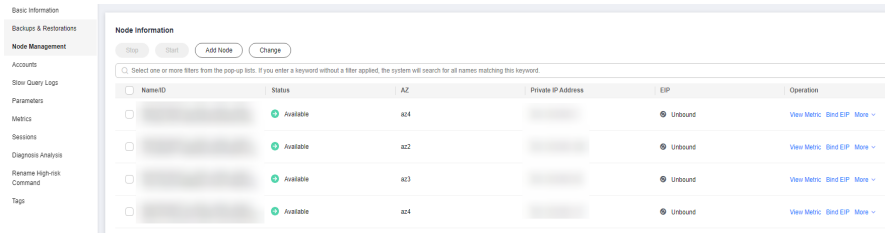
Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the instance you want to add nodes for and click its name.

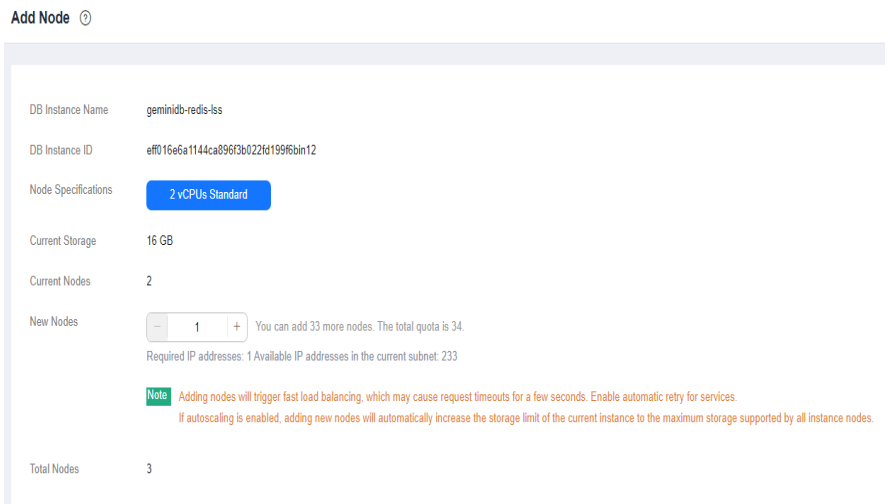
Step 4 In the navigation pane, choose **Node Management**.

Figure 4-114 Node management



Step 5 Click **Add Node**, on the displayed page, specify the number of nodes to be added and view the storage of the instance.

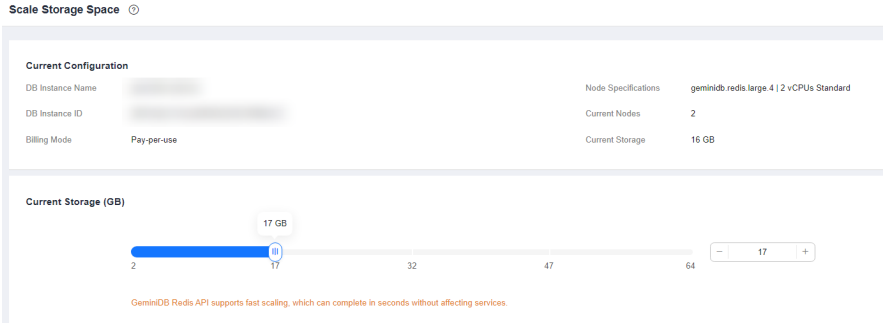
- If the storage is sufficient, click **Next** and go to **12**.
- If the storage is insufficient, click **Next** and go to **8**.



New nodes are of the same specifications as existing nodes. Once a new node is added, its specifications cannot be changed.

Step 6 On the **Scale Storage Space** page, select your target storage capacity and click **Next**.

Figure 4-115 Storage change



Step 7 On the displayed page, confirm the storage space.

- Yearly/Monthly
  - If you need to modify your settings, click **Previous**.
  - If you do not need to modify your settings, click **Next** and complete the payment.
- Pay-per-use
  - If you need to modify your settings, click **Previous**.
  - If you do not need to modify your settings, click **Submit** to scale up the storage space.

Step 8 After the storage capacity is expanded, go to **Step 3** to add nodes again.

Step 9 On the displayed page, confirm the node configuration details.

- Yearly/Monthly
  - If you need to modify your settings, click **Previous**.
  - If you do not need to modify your settings, click **Submit** and complete the payment.
- Pay-per-use
  - If you need to modify your settings, click **Previous**.
  - If you do not need to modify your settings, click **Submit**.

----End

4.6.6.3 Adding Instance Shards

Scenarios

This section describes how to add shards to a DB instance to suit your service requirements.

Precautions

- Currently, only standard instances in cloud native deployment mode can be added.
- Adding shards will trigger fast load balancing, which may cause a request timeout for a few seconds. Enable automatic retry for services.
- You can only add shards when the instance status is **Available** or **Checking restoration**.

- A DB instance cannot be deleted when one or more shards are being added.
- After shards are successfully added, the system automatically expands the storage capacity (*Number of new shards* x *Shard specification (GB)*).
- Currently, shards can be added only to proxy cluster instances.

Method 1

Step 1 Log in to the Huawei Cloud console.

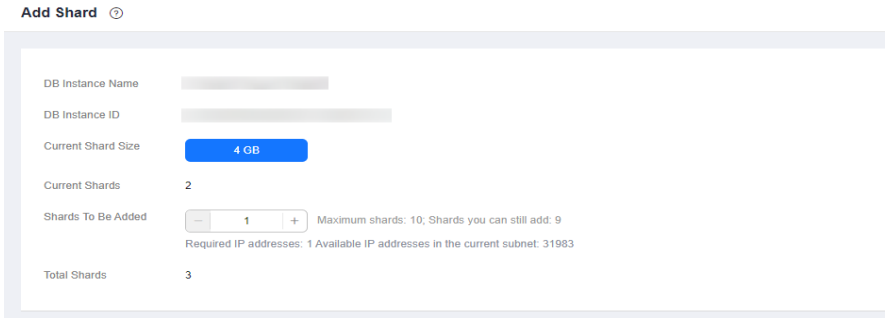
Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the target instance.

Step 4 In the navigation pane, choose **Shard Management**.

Figure 4-116 Shard management

Step 5 Click **Add Shard**. On the displayed page, select the number of shards to be added.



New shards are of the same specifications as existing shards. Once a shard is added, its specification cannot be changed.

Step 6 On the displayed page, confirm the shard configuration.

- Pay-per-use
  - To modify the configuration, click **Previous** to go back to the page where you specify details.
  - If you do not need to modify the configuration, click **Submit**.

----End

4.6.6.4 Deleting Instance Nodes

Scenarios

You can delete nodes from a pay-per-use or yearly/monthly instance to release resources.

Usage Notes

- Deleted nodes cannot be recovered. Exercise caution when performing this operation.
- If you enable operation protection, two-factor authentication is required for sensitive operations to secure your account and cloud products. For details about how to enable operation protection, see ***Identity and Access Management User Guide***.

Procedure

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the target instance.

Step 4 On the **Node Management** page, locate the node to be deleted.

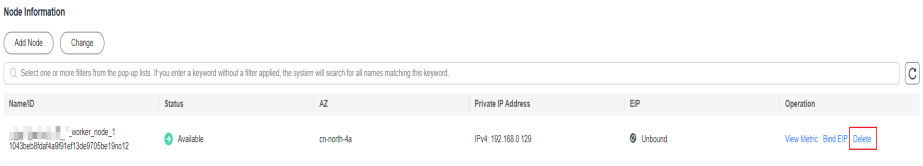
- Yearly/Monthly
  - In the **Node Information** area, locate the target node and choose **More** > **Delete** in the **Operation** column.

    Figure 4-117 Node information

- Pay-per-use
  - In the **Node Information** area, locate the target node and choose **More** > **Delete** in the **Operation** column.

    Figure 4-118 Node information

Step 5 If you have enabled operation protection, click **Start Verification** in the **Delete Node** dialog box. On the displayed page, click **Send Code**, enter the verification code, and click **Verify**. The page is closed automatically.

Step 6 In the displayed dialog box, click **Yes**.

- When the node is being deleted, the instance status is **Deleting node**.
- After the node is deleted, the instance status becomes **Available**.

----End

4.6.7 Scaling Disk Space

4.6.7.1 Scaling Disk Space

As more data is added, you may start to run out of space. This section describes how to scale up disk space of your instance. As data volumes decrease, you can scale down disk space to avoid low database node utilization and resource waste.**Table 4-43** lists the scaling methods supported by GeminiDB Redis instances.

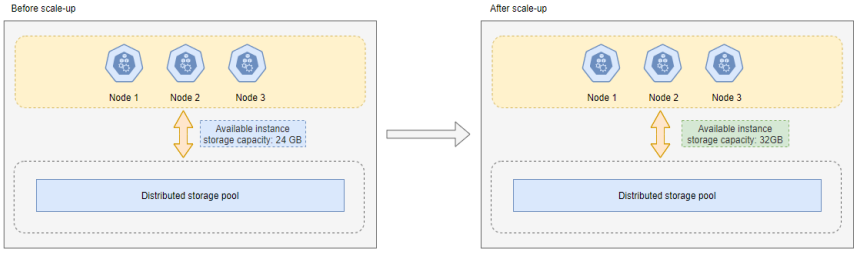
Table 4-43 Scaling methods

Method	Supported Instance Type	Description
4.6.7.2 Manually Scaling Up Disk Space	• Proxy cluster • Redis Cluster • Primary/ Standby	You can specify how much disk space needs to be added. The added value must be a multiple of 1 (GB). The total storage space cannot exceed the upper limit defined by your instance specifications.
4.6.7.3 Automatically Scaling Up Disk Space	Cluster	If storage usage exceeds the configured threshold, autoscaling will be triggered. The storage is scaled up by a percentage you specify. The added storage space is the current storage space multiplied by the scaling increment.
4.6.7.4 Manually Scaling Down Disk Space	• Proxy cluster • Redis Cluster • Primary/ Standby	You can specify how much disk space needs to be reduced. The storage space to be reduced must be a multiple of 1 GB and greater than or equal to 125% of the used storage space. The value is rounded up.

Manually Scaling Up Disk Space

For example, if the storage space of a GeminiDB Redis cluster instance is 24 GB and is increased by 8 GB, the storage space will become 32 GB.

Figure 4-119 Manually scaling up disk space



Automatically Scaling Up Disk Space

For example, the storage space of a GeminiDB Redis cluster instance is 24 GB before scale-up, the storage usage threshold for triggering autoscaling is set to 80%, and the total storage needs to be automatically scaled up by 10%. If the storage usage is greater than or equal to 80%, the storage space is automatically scaled up by 2.4 GB (24 x 10%), which is rounded up to 3 GB. In this case, the total storage space becomes 27 GB (24 + 3).

Figure 4-120 Automatically scaling up instance storage

Manually Scaling Down Instance Disk Space

For example, if the storage space of a GeminiDB Redis cluster instance is 32 GB and is decreased by 8 GB, the storage space will become 24 GB.

Figure 4-121 Manually scaling down disk space

4.6.7.2 Manually Scaling Up Disk Space

Scenarios

This section describes how to scale up storage of an instance to suit your service requirements.

Usage Notes

- To keep services accessible, scale up storage space when the storage usage exceeds 80%.
- Storage scaling does not interrupt your services. After storage scaling is complete, you do not need to restart your instance.

Setting an Instance Status to Read-only

To ensure that the instance can still be used if the storage space is about to be used up, the database is set to read-only, and data cannot be modified. If this happens, you can add more storage to restore the database to read/write status.

Table 4-44 Setting an instance status to read-only

Storage Space	Description
Less than 600 GB	• When the storage usage reaches 97%, the instance is set to read-only. • When the storage usage decreases to 85%, the read-only status is automatically disabled for the instance.
Greater than or equal to 600 GB	• The remaining storage space is less than 18 GB, and the instance is set to read-only. • If the remaining storage space is greater than or equal to 90 GB, the read-only status is automatically disabled for the instance.

Method 1

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the target instance.

Step 4 In the **Specification Information** area on the **Basic Information** page, click **Scale** for an instance.

Figure 4-122 Scaling classic storage

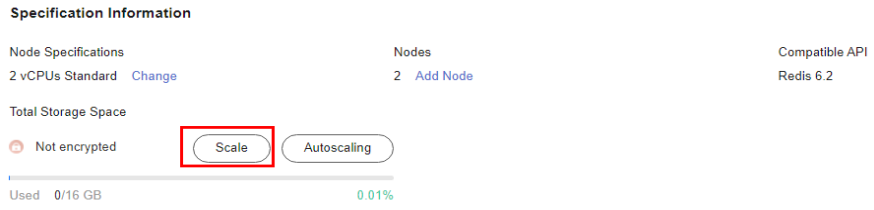


Figure 4-123 Scaling up cloud native storage

Step 5 On the displayed page, specify the new storage space and click **Next**.

Figure 4-124 Scaling up classic storage

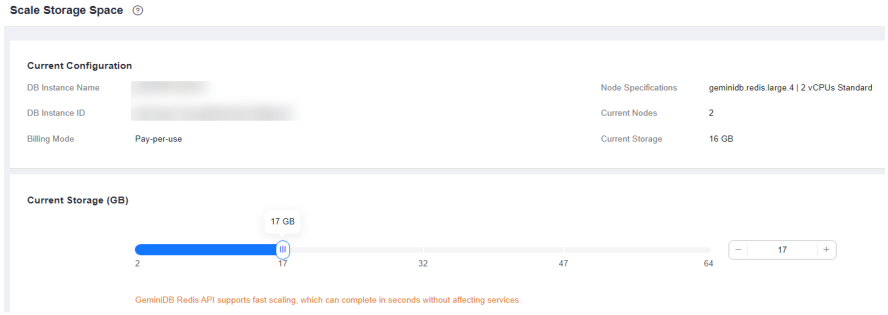


Figure 4-125 Scaling up cloud native storage

- To scale up classic storage, you need to add at least 1 GB each time. The value must be an integer.
- To scale up cloud native storage, you need to add at least 10 GB each time. The value must be an integer multiple of 10.

Step 6 On the displayed page, confirm the storage space.

- Yearly/Monthly
  - If you need to modify your settings, click **Previous**.
  - If you do not need to modify your settings, click **Submit** and complete the payment.
- Pay-per-use
  - If you need to modify your settings, click **Previous**.
  - If you do not need to modify your settings, click **Submit**.

Step 7 Check the results.

- When the scaling task is ongoing, the instance status is **Scaling storage space**.
- After the scaling task is complete, the instance status becomes **Available**.
- Click the instance name. In the **Specification Information** area on the **Basic Information** page, you can view the new storage space.

----End

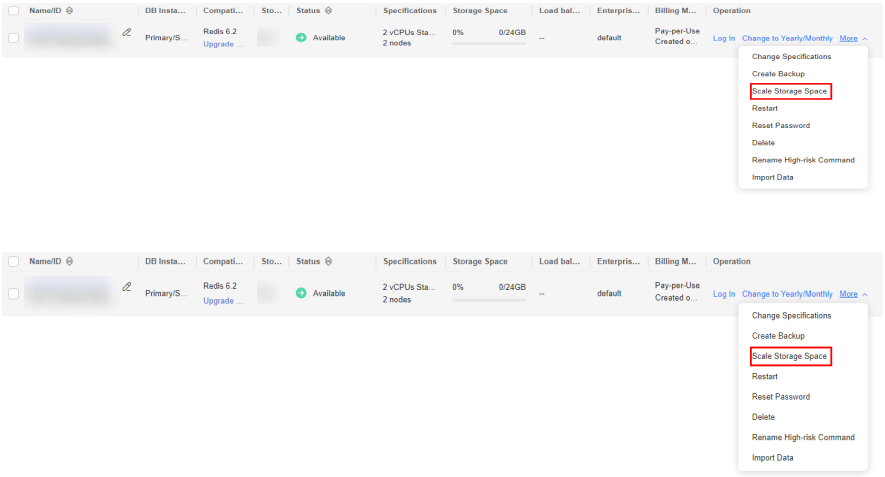
Method 2

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the instance whose storage space you want to scale and choose **More** > **Scale Storage Space** in the **Operation** column.

Figure 4-126 Scale Storage Space



Step 4 On the displayed page, specify the new storage space and click **Next**.

Figure 4-127 Scaling up classic storage

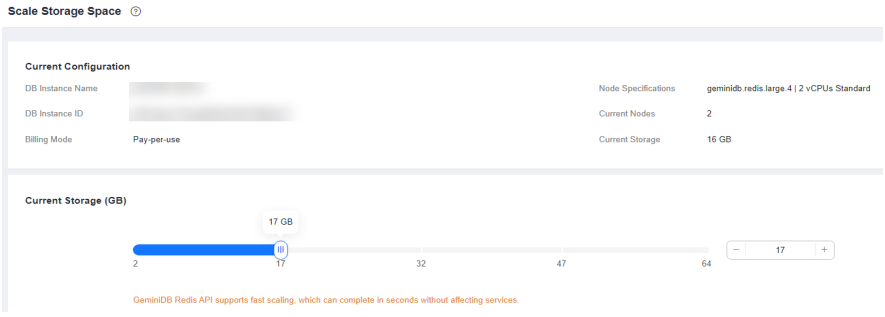
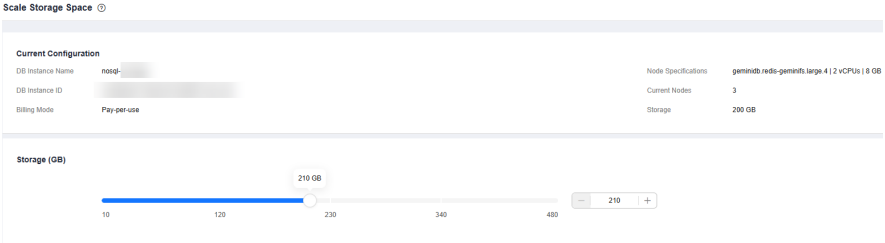


Figure 4-128 Scaling up cloud native storage



- To scale up classic storage, you need to add at least 1 GB each time. The value must be an integer.

- To scale up cloud native storage, you need to add at least 10 GB each time. The value must be an integer multiple of 10.

Step 5 On the displayed page, confirm the storage space.

- Yearly/Monthly
  - If you need to modify your settings, click **Previous**.
  - If you do not need to modify your settings, click **Submit** and complete the payment.
- Pay-per-use
  - If you need to modify your settings, click **Previous**.
  - If you do not need to modify your settings, click **Submit**.

Step 6 Check the results.

- When the scaling task is ongoing, the instance status is **Scaling storage space**.
- After the scaling task is complete, the instance status becomes **Available**.
- Click the instance name. In the **Specification Information** area on the **Basic Information** page, you can view the new storage space.

----End

4.6.7.3 Automatically Scaling Up Disk Space

You can enable storage autoscaling for GeminiDB Redis instances. When the storage space usage reaches the upper limit, autoscaling is triggered.

You can enable storage autoscaling when or after creating an instance. For details, see **4.2 Buying a GeminiDB Redis Instance**.

This section describes how to configure **Auto Scale** after an instance is created.

Configuring Permissions

If you are an IAM user, perform the following operations to configure GeminiDB permissions and IAM permissions before you enable storage autoscaling:

1. Configure fine-grained permissions for IAM and minimum permissions for GeminiDB.

   For details about how to configure IAM permissions, see **Creating a Custom Policy**.

```
{
    "Version":"1.1",
    "Statement":[
        {
            "Effect":"Allow",
            "Action":[
                "iam:permissions:listRolesForAgencyOnProject",
                "iam:permissions:grantRoleToGroupOnProject",
                "iam:agencies:createAgency",
                "iam:agencies:listAgencies",
                "iam:roles:listRoles",
                "iam:roles:createRole"
            ]
        }
    ]
}
```


2. **Create a user group and assign permissions to it.**

   You can create a user group on the IAM console and grant it custom permissions created in **1** and the security administrator role.

3. **Add a user to a user group.**

   Log in to the IAM console using a Huawei Cloud account or an IAM account, locate the IAM user that the target instance belongs to, and add it to the user group created in **2**. The IAM user will inherit permissions of the user group.

Usage Notes

- Autoscaling is available only when your account balance is sufficient.
- Autoscaling is in the OBT phase. To use it, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service.
- The instance is in the **Available** status.
- Once autoscaling is enabled, an agency will be created and fees will be automatically deducted.
- Only general-purpose GeminiDB Redis instances are supported.
- When the storage usage is greater than 98%:
  - If the total storage is less than 600 GB, the storage usage after autoscaling (used storage space/total storage space) will be less than 85%. For example, if the total storage is 500 GB and the used storage space is 495 GB, the storage usage (495/total storage space) after autoscaling will be less than 85%.
  - If the total storage is greater than or equal to 600 GB, the system automatically scales up the storage by over 90 GB. For example, if the total storage is 700 GB, the storage after autoscaling will be greater than 790 GB (700 + 90).
- Changing the instance class or the number of nodes will affect the upper limit of the storage space.

Autoscaling of a Single Instance

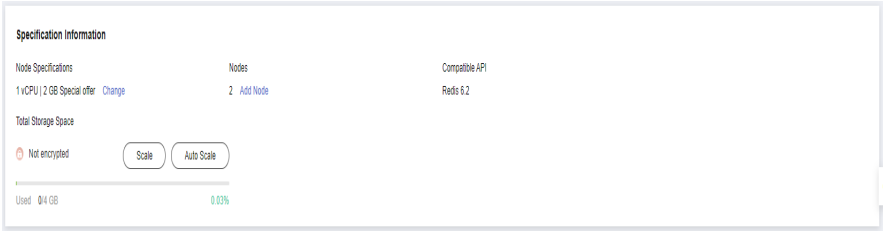
Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the target instance. The **Basic Information** page is displayed.

Step 4 In the **Total Storage Space** area, click **Auto Scale**.

Figure 4-129 Auto Scale



Step 5 Toggle on **Auto Scale** and specify the trigger condition and increment.

Figure 4-130 Configuring autoscaling parameters

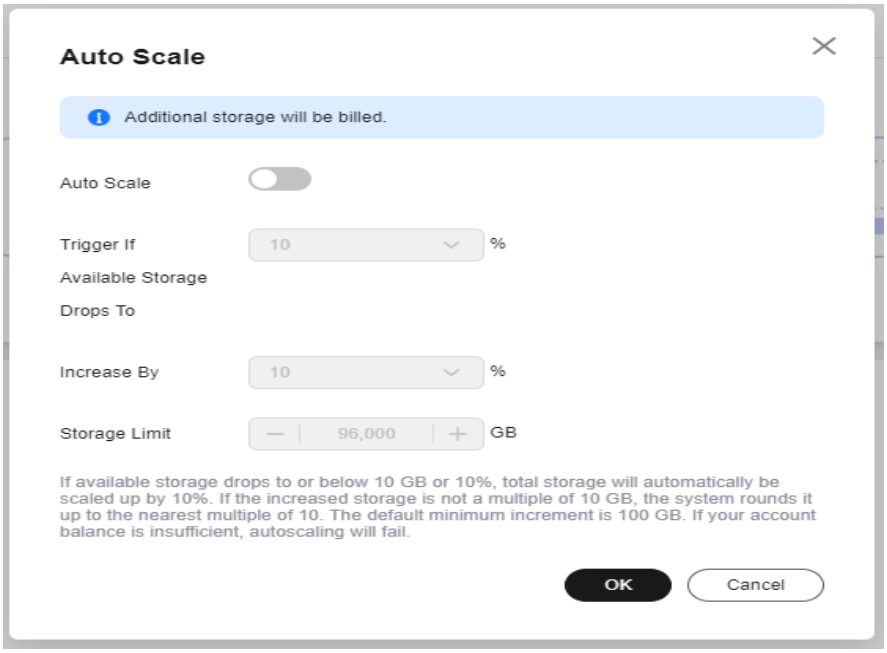


Table 4-45 Parameter description

Parameter	Description
Auto Scale	If you toggle on this switch, autoscaling is enabled.
Trigger If Available Storage Drops To	When the available storage usage drops to a specified threshold or the available storage drops to 10 GB, autoscaling is triggered.
Increase By	Percentage of the current storage to be automatically scaled. The value can be **10**, **15**, or **20**. If the value is not a multiple of 10, it is rounded up. At least 1 GB is added each time.

Step 6 Click **OK**.

----End

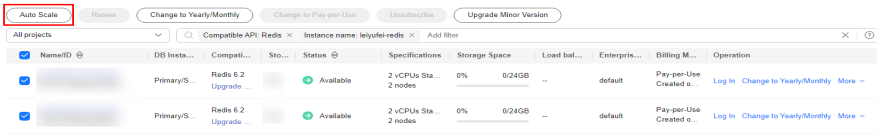
Autoscaling Storage Space of Instances in Batches

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the DB instance list page, select the instances you want to enable autoscaling for and click **Auto Scale**.

Figure 4-131 Auto Scale



Step 4 Toggle on **Autoscaling** and specify the trigger condition and increment.

Figure 4-132 Configuring autoscaling parameters

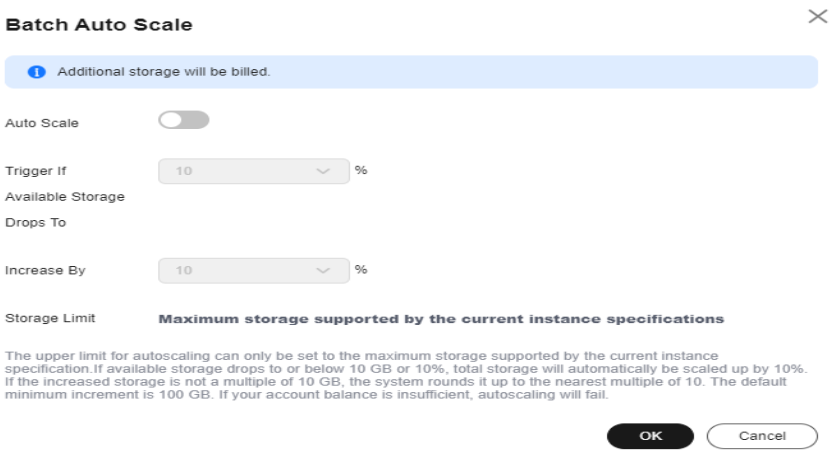


Table 4-46 Parameter description

Parameter	Description
Auto Scale	If you toggle on this switch, autoscaling is enabled.
Storage Usage	When the available storage usage drops to a specified threshold or the available storage drops to 10 GB, autoscaling is triggered.
Increase By	Percentage of the current storage to be automatically scaled. The value can be **10**, **15**, or **20**. If the value is not a multiple of 10, it is rounded up. At least 1 GB is added each time.

Step 5 Click **OK**.

----End

4.6.7.4 Manually Scaling Down Disk Space

Scenarios

As data volumes decrease, you can scale down disk space to avoid low database node utilization and resource waste.

Usage Notes

- When you scale down storage space of your instance, make sure that new storage space is 25% more than the used space and rounded up.
- Storage scaling does not interrupt your services. After storage space is scaled, you do not need to restart your instance.

Setting an Instance Status to Read-only

To ensure that the GeminiDB Redis instance can still run properly when the storage space is about to be used up, the database is set to read-only, and data cannot be modified. If this happens, you can scale up the storage to restore the database status to read/write.

Table 4-47 Setting an instance status to read-only

Storage Space	Description
< 600 GB	• When the storage usage reaches 97%, the instance is read-only. • When the storage usage decreases to 85%, the read-only status is automatically disabled for the instance.
≥ 600 GB	• When the remaining storage space is less than 18 GB, the instance is read-only. • When the remaining storage space is greater than or equal to 90 GB, the read-only status is automatically disabled for the instance.

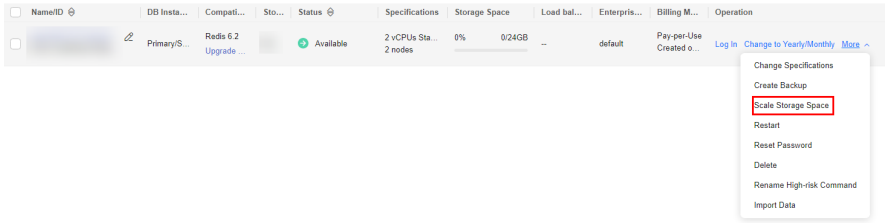
Method 1

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

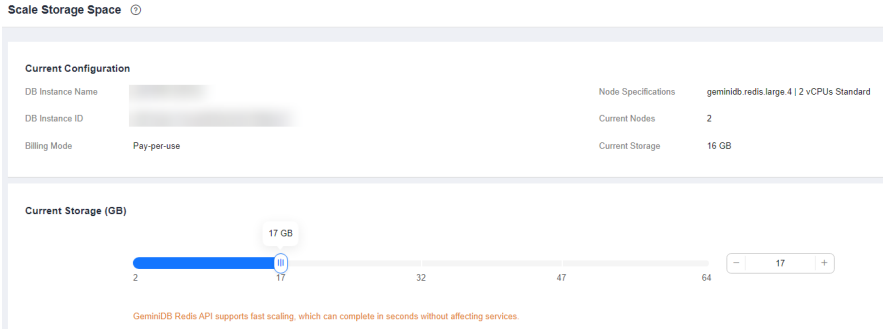
Step 3 On the **Instances** page, locate the target instance and choose **More** > **Scale Storage Space** in the **Operation** column.

Figure 4-133 Scale Storage Space



Step 4 On the displayed page, specify the new storage space and click **Next**.

Figure 4-134 Scale Storage Space



Select at least 1 GB each time you scale up the storage, and the storage size must be an integer.

Step 5 On the displayed page, confirm the storage space.

- Yearly/Monthly
  - If you need to modify your settings, click **Previous**.
  - If you do not need to modify your settings, click **Submit** and complete the payment.
- Pay-per-use
  - If you need to modify your settings, click **Previous**.
  - If you do not need to modify your settings, click **Submit**.

Step 6 Check the results.

- When the scaling task is ongoing, the instance status is **Scaling storage space**.
- After the scaling task is complete, the instance status becomes **Available**.
- Click the instance name. In the **Specification Information** area on the **Basic Information** page, you can view the new storage space.

----End

Method 2

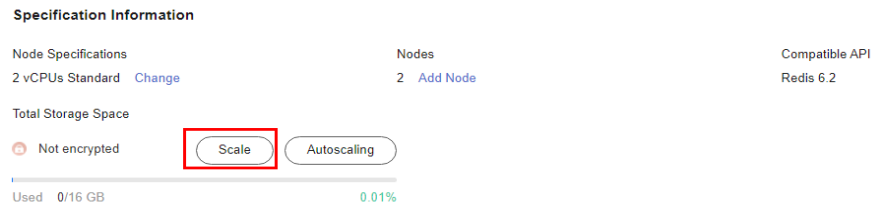
Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the target instance.

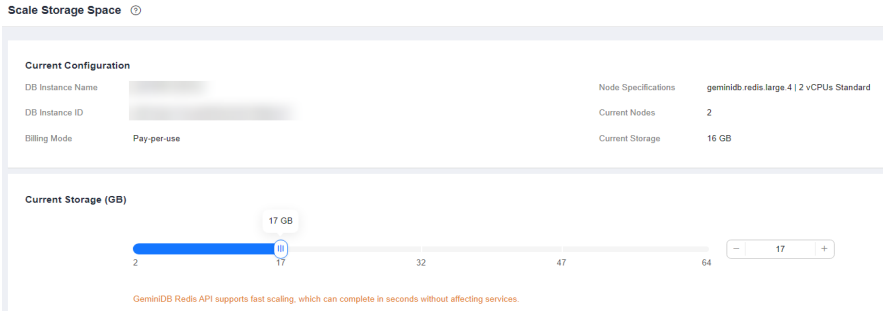
Step 4 In the **Specification Information** area on the **Basic Information** page, click **Scale**.

Figure 4-135 Scaling storage



Step 5 On the displayed page, specify the new storage space and click **Next**.

Figure 4-136 Scaling storage



Select at least 1 GB each time you scale up the storage, and the storage size must be an integer.

Step 6 On the displayed page, confirm the storage space.

- Yearly/Monthly
  - To modify your settings, click **Previous**.
  - If you do not need to modify your settings, click **Next** and complete the payment.
- Pay-per-Use
  - To modify your settings, click **Previous**.
  - If you do not need to modify your settings, click **Submit**.

Step 7 Check the results.

- When the scaling task is ongoing, the instance status is **Scaling storage space**.
- After the scaling task is complete, the instance status becomes **Available**.
- Click the instance name. In the **Specification Information** area on the **Basic Information** page, you can view the new storage space.

----End

4.6.8 Performing a Primary/Standby Switchover for GeminiDB Redis Instances

Scenarios

GeminiDB Redis instances provide an automatic HA mechanism. Generally, you do not need to manually perform a primary/standby switchover. You can use this feature to perform DR drills and test client processing capabilities in HA scenarios. You can also perform a primary/standby switchover as needed to meet service requirements.

Prerequisites

Currently, this operation can be performed only on primary/standby GeminiDB Redis instances that are in the **Available** state.

Precautions

- When you perform a primary/standby switchover, the instance IP addresses remain unchanged, so there is no need to update the service connection address.
- During a primary/standby switchover, the connection is disconnected for less than 10 seconds, which can cause slow latency or command execution failures. To address this issue, you need to have a command retry or connection retry mechanism on the client. Perform primary/standby switchovers during off-peak hours.

Procedure

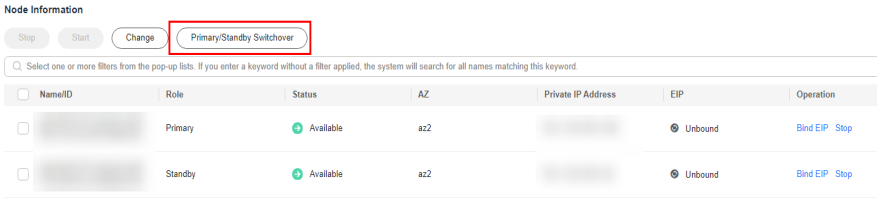
Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the target instance.

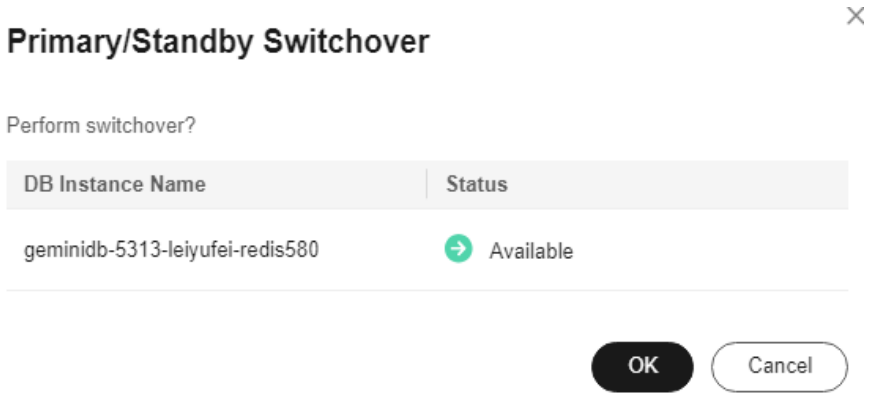
Step 4 In the **Node Information** area on the **Basic Information** page, click **Primary/Standby Switchover**.

Figure 4-137 Performing a primary/standby switchover.



Step 5 In the displayed dialog box, click **OK**.

Figure 4-138 Performing a primary/standby switchover.



----End

4.7 Data Backup

4.7.1 Overview

You can create backups for GeminiDB Redis instances to ensure data reliability. After an instance is deleted, its automated backups are also deleted while manual backups are retained. The back ups cannot be exported. GeminiDB Redis instances support only full backup.

Usage Notes

Backing up data consumes a few CPUs. Uploading backup files to OBS occupies bandwidth of compute nodes, causing slight latency and jitter.

Backup Methods

Both automatic backup and manual backup are supported.

- Automated backup

  You can **modify backup policy** on the GeminiDB console, and the system will automatically back up your instance data based on the time window and backup cycle you configure in the backup policy and will store the data for a length of time you specify.

  Automated backups cannot be manually deleted. You can adjust their retention period by referring to **Modifying an Automated Backup Policy**, and backups that expire will be automatically deleted.

- Manual backup

  A manual backup is a full backup of a DB instance and can be retained until you manually delete it. Manual backup can be triggered at any time to meet your service requirements.

  Regularly backing up your database is recommended. If your database becomes faulty or data is corrupted, you can restore it from backup.

Table 4-48 Backup methods

Method	Scenario
Automated backup	After you set a backup policy, the system automatically backs up your database based on the policy. You can also modify the policy based on service requirements.
Manual backup	You can manually create full backups for your instance based on service requirements.

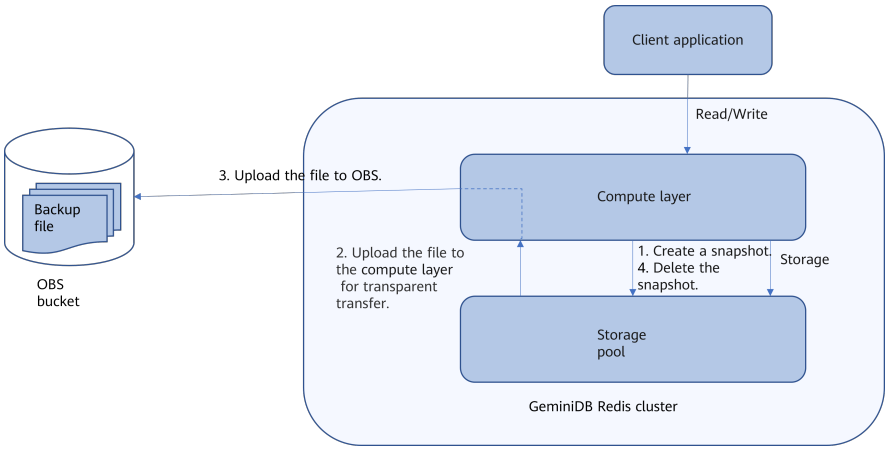
How Backup Works

GeminiDB Redis API uses an architecture with decoupled storage and compute. **Figure 4-139** shows how GeminiDB Redis API backs up data. GeminiDB Redis API takes snapshots of data in the DFV storage pool in seconds and transmits them to the compute layer, which will then pass them to OBS. Snapshots are stored as compressed files in OBS buckets and do not occupy any storage space of your instance. Snapshot creation and deletion have no impacts on the computing layer.

When snapshots are uploaded, some compute resources are consumed, so the instance CPU and memory usage may go high.

GeminiDB Redis backs up data faster than Redis and does not produce jitter.

Figure 4-139 Backup process



Backup Storage

Backups are stored in OBS buckets to provide disaster recovery and save storage space.

After you purchase an instance, GeminiDB Redis will provide additional backup storage of the same size as what you purchased. For example, if you purchase an instance with 100 GB of storage, you will obtain additional 100 GB of storage free of charge. If the backup data does not exceed 100 GB, it is stored on OBS free of charge. If there is more than 100 GB of data, you will be billed at standard OBS rates.

4.7.2 Managing Automated Backups

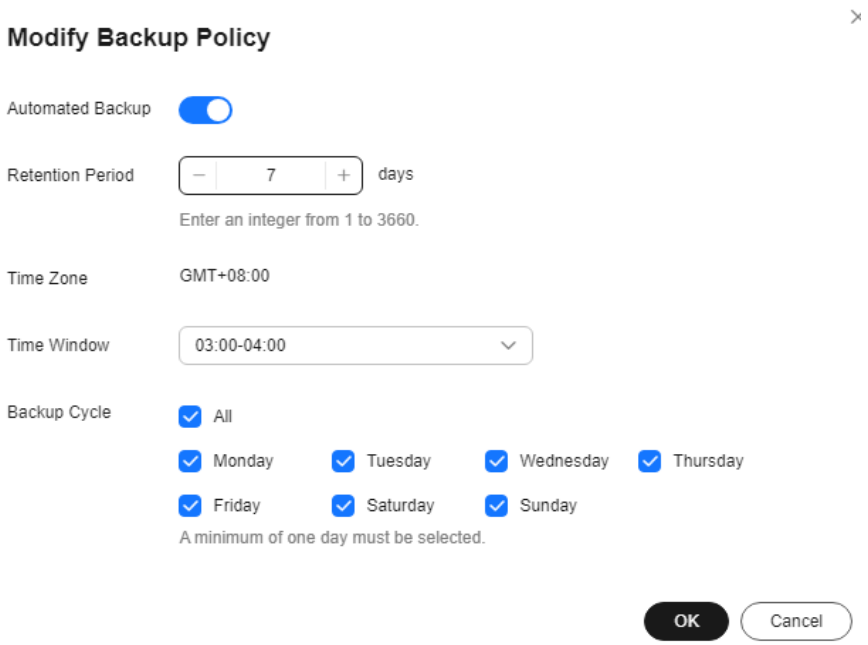
GeminiDB Redis allows you to create automated backups to protect your data. If a database or table is deleted, maliciously or accidentally, backups can help recover your data.

Configuring an Automated Backup Policy

Automated backups are generated based on a backup policy and saved as packages in OBS buckets to secure and protect your data. Regularly backing up your database is recommended. If your database becomes faulty or data is corrupted, you can restore it from backup. Backing up data affects the database read and write performance, so you are advised to set the automated backup time window to off-peak hours.

When you create an instance, automated backup is enabled by default.

Figure 4-140 Modifying a backup policy



- **Retention Period**: Automated backup files are saved for seven days by default. The retention period ranges from 1 to 3660 days. Full backups are retained till the retention period expires.
  - Extending the retention period improves data reliability. You can extend the retention period as needed.
  - If you shorten the retention period, the new backup policy takes effect for existing backups. Any automated backups (including full and incremental backups) that have expired will be automatically deleted. Manual backups will not be automatically deleted but you can delete them manually.

  NOTE

  - If the retention period is less than seven days, the system automatically backs up data daily.
  - The system checks existing automated backups and deletes any backups that exceed the backup retention period you configure.
  - **Time Window**: A one-hour period the backup will be scheduled for, such as 04:00–05:00. The backup time is in GMT format. If the DST or standard time is switched, the backup time segment changes with the time zone.

  If **Retention Period** is set to **2**, full and incremental backups that have been stored for more than two days will be automatically deleted. For instance, a backup generated on Monday will be deleted on Wednesday; or a backup generated on Tuesday will be deleted on Thursday.

  Policy for automatically deleting full backups:

  To ensure data integrity, even after the retention period expires, the most recent backup will be retained, for example,

  If **Backup Cycle** was set to **Monday** and **Tuesday** and the **Retention Period** was set to **2**:

  - A full backup generated on Monday will be automatically deleted on Thursday. The reasons are as follows:

    The full backup generated on Monday expires on Wednesday, but it is the last backup, so it will be retained until a new backup expires. The next backup will be generated on Tuesday and will expire on Thursday. So the full backup generated on Monday will not be automatically deleted until Thursday.
  - The full backup generated on Tuesday will be automatically deleted on the following Wednesday. The reasons are as follows:

    The backup generated on Tuesday will expire on Thursday, but as it is the last backup, so it will be retained until a new backup expires. The next backup will be generated on the following Monday and will expire on the following Wednesday. So the full backup generated on Tuesday will not be automatically deleted until the following Wednesday.
- **Backup Cycle**: All options are selected by default.
  - **All**: Each day of the week is selected. The system automatically backs up data every day.
  - You can select one or more days in a week. The system automatically backs up data at the specified time.

  NOTE

  A full backup starts within one hour of the time you specify. The amount of time required for the backup depends on the amount of data to be backed up. The more data has to be backed up, the longer it will take.
- After the DB instance is created, you can modify the automated backup policy as needed. You can change the time window after the DB instance is created. The system backs up data based on the automated backup policy you have set.
- If **Automated Backup** is disabled, any automated backups in progress stop immediately.

Modifying an Automated Backup Policy

Step 1 **Log in to the Huawei Cloud console.**

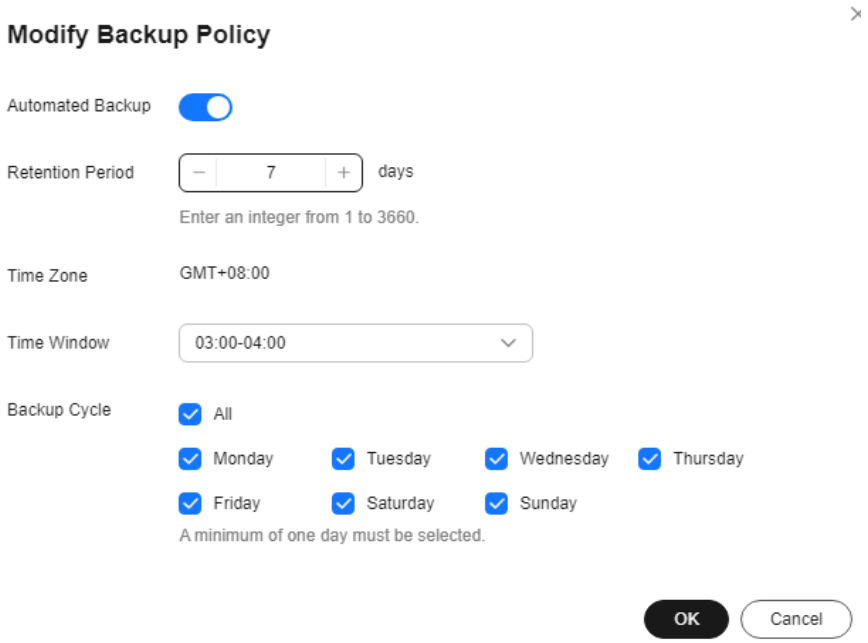
Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the instance whose backup policy you want to modify.

Step 4 Choose **Backups & Restorations** in the navigation pane one the left, and click **Modify Backup Policy**. In the displayed dialog box, set the backup policy. Click **OK**.

For details about how to set a backup policy, see **Configuring an Automated Backup Policy**.

Figure 4-141 Modifying a backup policy



Step 5 Check or manage the generated backups on the **Backups** or **Backups & Restorations** page.

----End

Disabling Automated Backup

Step 1 Log in to the Huawei Cloud console.

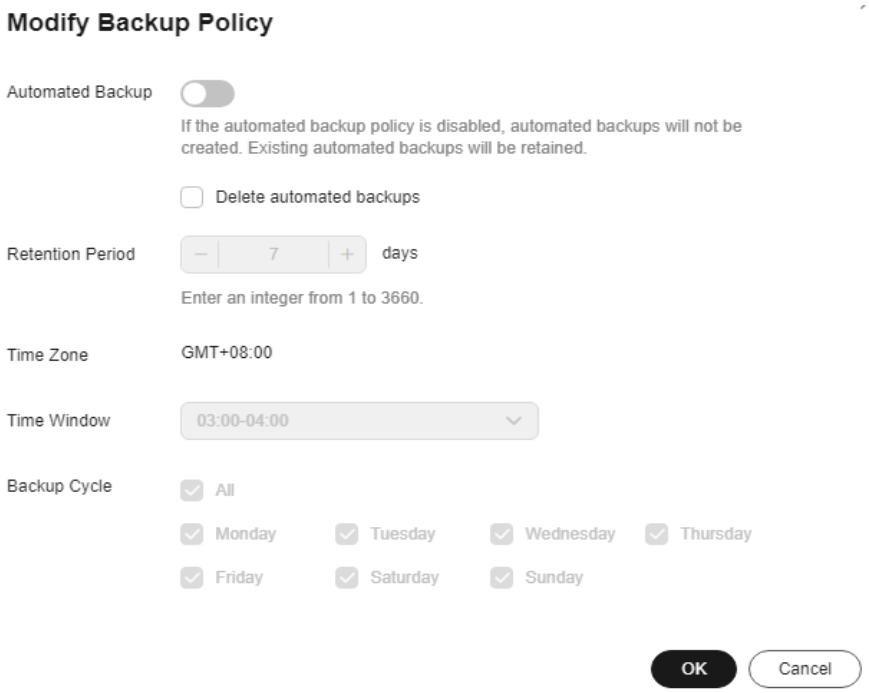
Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the instance whose backup policy you want to modify.

Step 4 Choose **Backups & Restorations** in the navigation pane one the left, and click **Modify Backup Policy**.

Step 5 In the displayed dialog box, click  to disable the backup policy and click **OK**.

Figure 4-142 Disabling backup policies



When disabling the automated backup policy, you can decide whether to delete the automated backups by selecting **Delete automated backups**.

- If you select it, all backup files within the retention period will be deleted. No automated backups are displayed in the backup list until you enable the automated backup policy again.
- If you do not select it, all backup files within the retention period will be retained, but you can still manually delete them later if needed. For details, see **Deleting an Automated Backup**.

If **Automated Backup** is disabled, any automated backups in progress stop immediately.

----End

Deleting an Automated Backup

If automated backup is disabled, you can delete stored automated backups to free up storage space.

If automated backup is enabled, the system will delete automated backups as they expire. You cannot delete them manually.

NOTICE

To delete an automated backup, disable the automated backup policy first. For details, see **Disabling Automated Backup**.

Deleted backups cannot be recovered. Exercise caution when performing this operation.

- **Method 1**
  a. **Log in to the Huawei Cloud console.**
  b. In the service list, choose **Databases** > **GeminiDB Redis API**.
  c. On the **Instances** page, click the instance whose backup you want to delete.
  d. Choose **Backups & Restorations** in the navigation pane, locate the target backup and click **Delete** in the **Operation** column.
  e. In the **Delete Backup** dialog box, confirm the backup details and click **Yes**.
- **Method 2**
  a. **Log in to the Huawei Cloud console.**
  b. In the service list, choose **Databases** > **GeminiDB Redis API**.
  c. On the **Backups** page, locate the backup that you want to delete and click **Delete**.
  d. In the **Delete Backup** dialog box, confirm the backup details and click **Yes**.

Setting the Policy for Restoring Data to a Specified Time Point

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the target instance.

Step 4 Choose **Backups & Restorations** in the navigation pane one the left, and click **Point in Time Restoration**. After the setting is complete, click **OK**.

Figure 4-143 Setting the policy for restoring data to a specified time point

- You can toggle on or off **Enable** to enable or disable the backup function.
- **Backup Interval** refers to the time interval, in minutes, for automated backups. The value ranges from 5 to 120. For example, if the initial backup is scheduled for 04:00, the subsequent backup will take place at 04:05.

- **Retention Period** determines how long automated backups are kept in days. The value ranges from 1 to 7. Full backups are retained till the retention period expires.

----End

4.7.3 Managing Manual Backups

GeminiDB Redis API allows you to manually back up instances whose status is **Available** to protect your data. If a database or table is deleted, maliciously or accidentally, backups can help recover your data. Manual backups are full backups.

Creating a Manual Backup

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 Create a manual backup.

Method 1

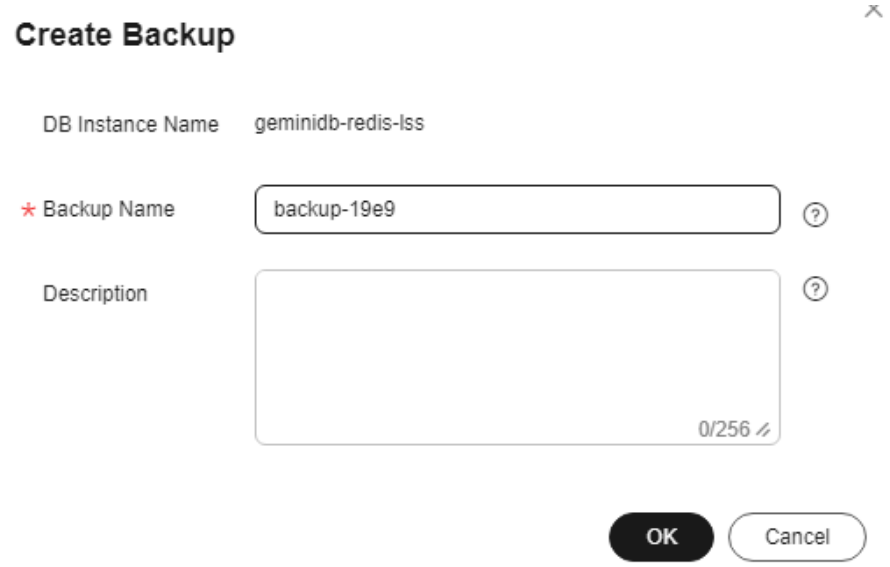
On the **Instances** page, locate the instance you want to back up and choose **More** > **Create Backup** in the **Operation** column.

Figure 4-144 Creating a manual backup

Method 2

1. On the **Instances** page, click the instance you want to back up.
2. Choose **Backups & Restorations** in the navigation pane on the left, and click **Create Backup**.

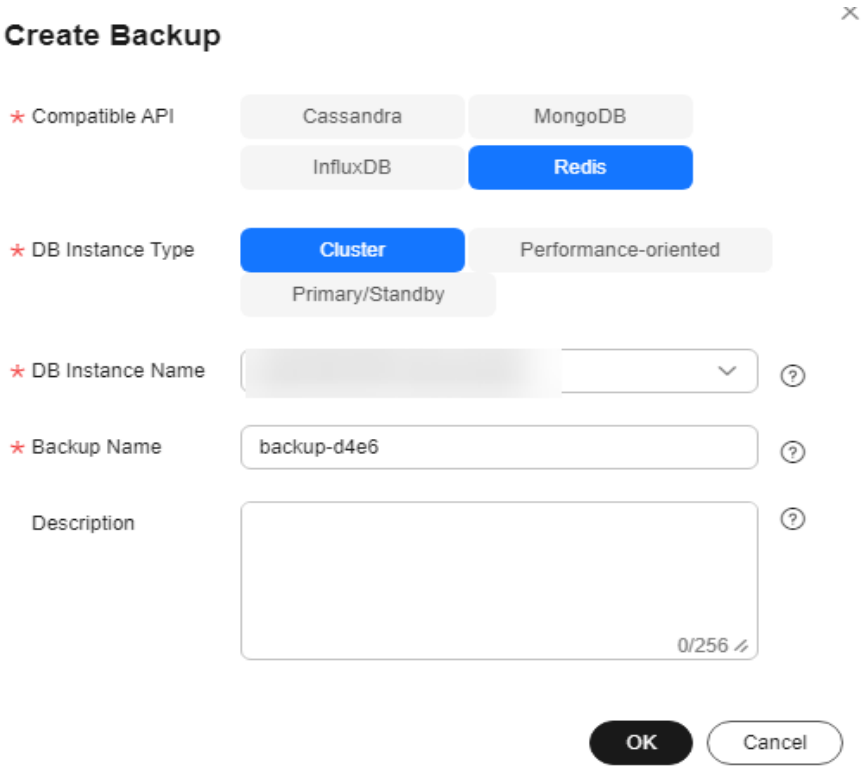
Figure 4-145 Creating a manual backup



Method 3

In the navigation pane on the left, choose **Backups**. On the displayed page, click **Create Backup**.

Figure 4-146 Creating a manual backup



Step 4 In the displayed dialog box, specify a backup name and description and click **OK**.

Table 4-49 Parameter description

Parameter	Description
DB Instance Name	Must be the name of the DB instance to be backed up and cannot be modified.
Backup Name	Must be 4 to 64 characters long and start with a letter. It is case-insensitive and contains only letters, digits, hyphens (-), and underscores (_).
Description	Can include a maximum of 256 characters and cannot contain line breaks and the following special characters: >!<"&'=

Step 5 View the backup status.

- When the backup is being created, query the backup status on the **Backups** or **Backups & Restorations** page. The backup status is **Backing up**.
- After the backup is created, the backup status changes to **Completed**.

----End

Deleting a Manual Backup

If you do not need a manual backup any longer, delete it on the **Backups** or **Backups & Restorations** page.

Deleted backups are not displayed in the backup list.

NOTICE

Deleted backups cannot be recovered. Exercise caution when performing this operation.

Method 1

1. **Log in to the Huawei Cloud console.**
2. In the service list, choose **Databases** > **GeminiDB Redis API**.
3. On the **Instances** page, locate the instance whose backup you want to delete and click its name.
4. Choose **Backups & Restorations** in the navigation pane, locate the target backup and click **Delete** in the **Operation** column.
5. In the displayed dialog box, confirm the backup details and click **Yes**.

Method 2

1. **Log in to the Huawei Cloud console.**
2. In the service list, choose **Databases** > **GeminiDB Redis API**.
3. On the **Backups** page, locate the backup that you want to delete and click **Delete**.

4. In the displayed dialog box, confirm the backup details and click **Yes**.

4.8 Data Restoration

4.8.1 Restoration Methods

GeminiDB Redis supports multiple forms of data restoration. You can select one based on service requirements.

Table 4-50 Restoration methods

Reference	Scenario
Rebuilding an Instance	If an instance is deleted by mistake, you can rebuild it within a retention period in the recycle bin.
4.8.2 Restoring Data to a New Instance	You can restore an existing backup file to a new instance.
4.8.3 Restoring to the Original Instance Using PITR	GeminiDB Redis API offers Point-In-Time Recovery (PITR), which allows for quick data recovery by restoring the database to its previous state before any errors occurred.

4.8.2 Restoring Data to a New Instance

Scenarios

GeminiDB Redis allows you to use an existing backup to restore data to a new instance.

Procedure

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 Restore a DB instance from the backup.

Method 1

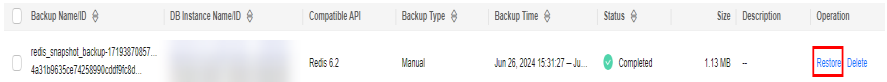
1. On the **Instances** page, locate the instance whose backup you want to restore and click its name.
2. Choose **Backups & Restorations** in the navigation pane on the left, locate the backup that you want to restore and click **Restore** in the **Operation** column.

Figure 4-147 Restoration

Method 2

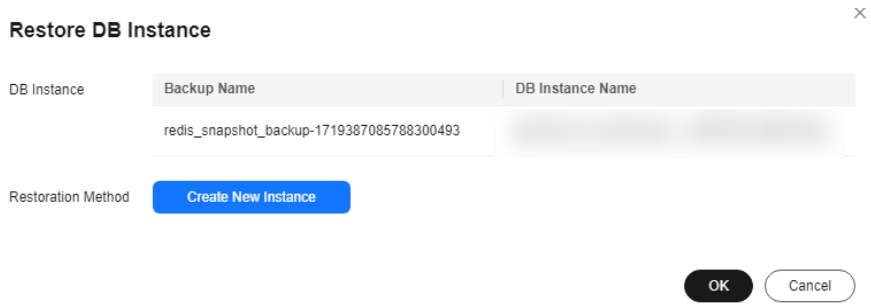
On the **Backups** page, locate the backup that you want to restore and click **Restore** in the **Operation** column.

Figure 4-148 Restoration



Step 4 In the displayed dialog box, confirm the current instance details and restoration method and click **OK**.

Figure 4-149 Restoring data to a new DB instance



- The default API type and DB engine version are the same as those of the original instance and cannot be changed.
- The new instance must have no less than nodes than the original instance.
- GeminiDB automatically calculates the minimum storage space required for restoration based on the size of the selected backup file. The storage capacity depends on the instance specifications, and must be an integer.
- You need to set a new administrator password.
- To modify other parameters, see the description of buying instances of other DB APIs in *Getting Started*.

Step 5 View the restoration results.

A new instance is created using the backup data. The status of the new instance changes from **Creating** to **Available**.

After the restoration, the system will perform a full backup.

The new DB instance is independent from the original one.

----End

4.8.3 Restoring to the Original Instance Using PITR

In real-world service scenarios, databases may experience faults such as data damage, loss, or accidental deletion. GeminiDB Redis API offers Point-In-Time Recovery (PITR), which allows for quick data recovery by restoring the database to its previous state before any errors occurred.

Functions

Point-in-Time Recovery (PITR) is a database feature that enables restoration to a specific time, useful for recovering data lost or damaged due to misoperations or accidental deletion.

In gaming scenarios, some players may exploit vulnerabilities to duplicate equipment and currency, leading to unfairness. Traditional databases are backed up once a day, with a long restoration time, making it difficult to restore data to a specific point in time. With PITR of GeminiDB Redis API, you can choose a specific time point for data recovery, with a minimum granularity of 5 minutes, ensuring speedy data restoration.

Constraints

- Only GeminiDB Redis cluster instances are supported. DR instances are not supported.
- Data can only be restored to the original instance, and the database is unavailable during the restoration.
- This function is in the open beta test (OBT). To use it, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact technical support.

Setting Point-in-Time Restoration

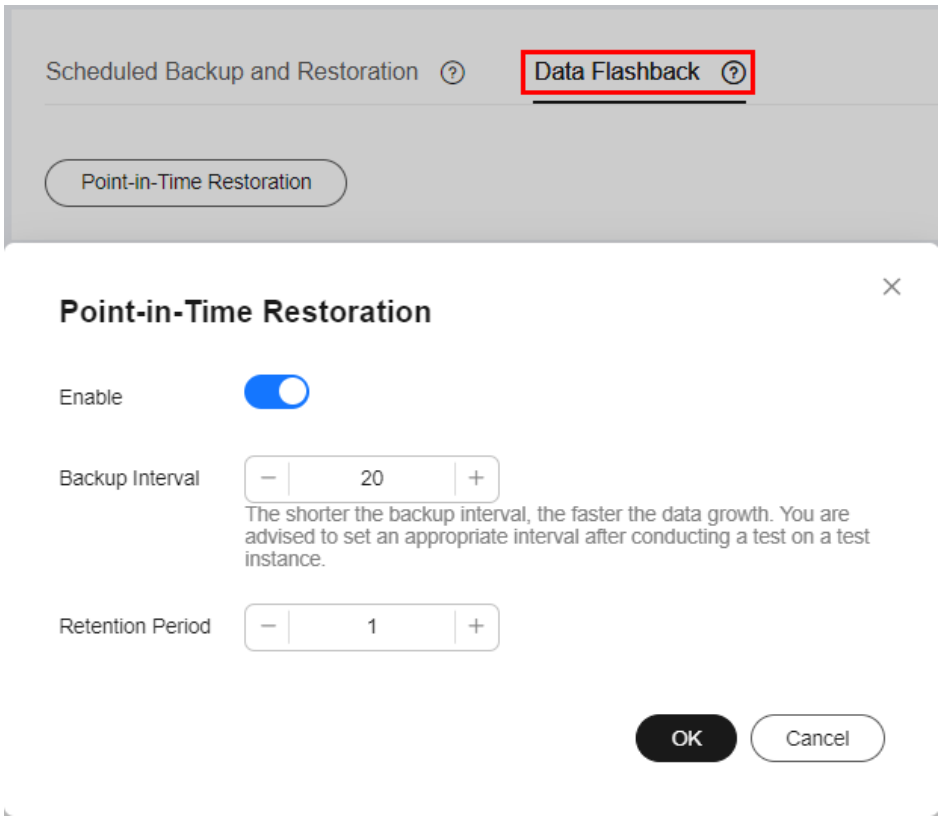
Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the target instance.

Step 4 Choose **Backups & Restorations** in the navigation pane. On the **Data Flashback** tab page, click **Point-in-Time Restoration**. Configure parameters in the displayed dialog box. After the setting is complete, click **OK**.

Figure 4-150 Point-in-Time Restoration



- You can toggle on or off **Enable** to enable or disable the backup function.
- **Backup Interval** refers to the time interval, in minutes, for automated backups. The range of values is from 5 to 120 minutes. For example, if the initial backup is scheduled for 04:00, the subsequent backup will take place at 04:05.
- **Retention Period** determines how long automated backups are kept in days. The range of values is from 1 to 7 days. Backups are retained till the retention period expires.

CAUTION

To avoid rapid data bloat, it is important to set an appropriate backup interval in the test environment before your actual use.

----End

Restoring Data to the Original Instance

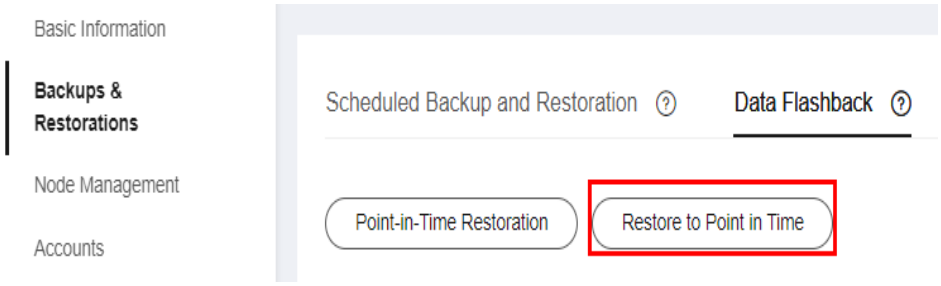
Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the target instance.

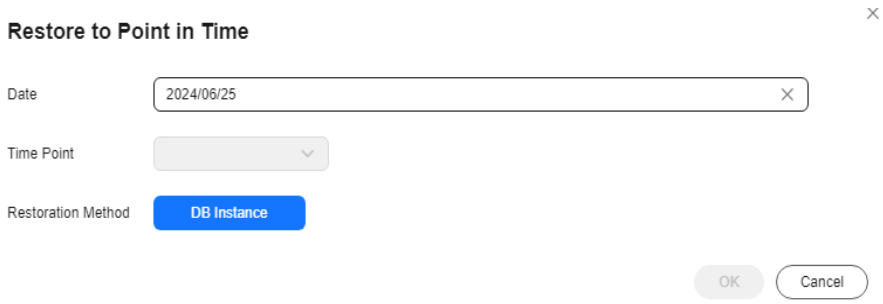
Step 4 Choose **Backups & Restorations** in the navigation pane. On the **Data Flashback** tab page, click **Restore to Point in Time**.

Figure 4-151 Backup and restoration



Step 5 Select the date and time point to which the data is restored.

Figure 4-152 Restoring data to a point in time



Step 6 Click **OK**.

----End

4.9 Diagnosis Analysis

4.9.1 Big Key Diagnosis

Functions

A key that contains a large volume of data is considered a big key. GeminiDB Redis API can diagnose big keys, allowing you to collect statistics on big keys in the current instance.

GeminiDB Redis API uses shared storage, so big keys do not cause data skew or out of memory (OOM) issues on shards. Access to big keys is common when you are using Redis databases. Big key diagnosis obtains analysis results from the background, minimizing the impact on services.

Viewing the Diagnostic Information of Big Keys

Step 1 Log in to the Huawei Cloud console.

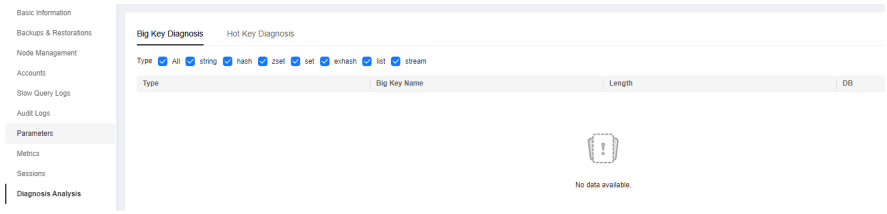
Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the name of the target instance to go to the **Basic Information** page.

Step 4 In the navigation pane on the left, choose **Diagnosis Analysis**.

Step 5 Choose **Big Key Diagnosis** and select the target types.

Figure 4-153 Big key diagnosis



Step 6 View big key parameters by referring to **Table 4-51**.

Table 4-51 Big key parameters

Parameter	Description
Type	Type of a big key. • **string** • **hash** • **zset** • **set** • **exhash** • **list** • **stream**
Big Key Name	Name of a big key.
Length	Length of the value.
DB	Database where a big key is located.

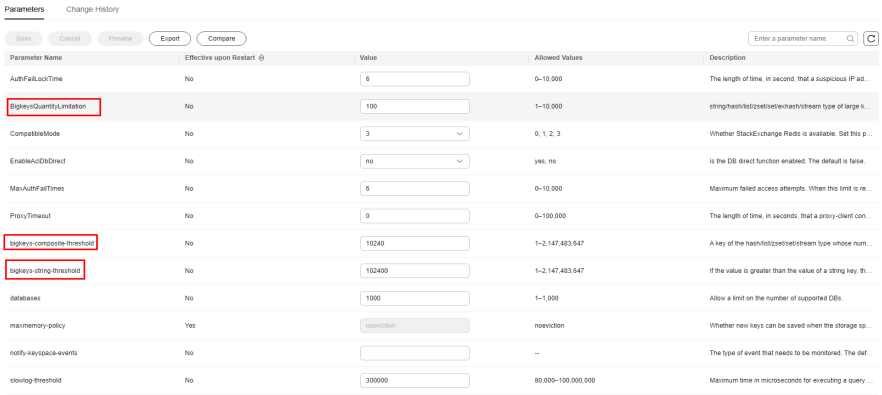
----End

Setting Parameters for Big Key Diagnosis

The value size of a key determines whether the key is a string key. The number of members in the key determines whether the key is a hash, list, zset, set, or stream key.

There are two parameters involved:

- **bigkeys-string-threshold**: If the value size of a string key is greater than the value of this parameter, the key is determined as a big key. The unit is byte. The default value is **102400** (1 MB).
- **bigkeys-composite-threshold**: If the number of elements in a hash, list, zset, set, or stream key is greater than the value of this parameter, the key is identified as a big key. The default value is **10240**.
- **Figure 4-154** Parameters for big key diagnosis



Do not set these two parameters to small values. Otherwise, too many invalid results are generated and occupy the network bandwidth, slowing down data access.

For details about how to modify parameters for big key diagnosis, see **Modifying Parameters of an Instance**.

4.9.2 Hot Key Diagnosis

A key that is frequently accessed is considered a hot key. This section describes how to use hot key diagnosis for GeminiDB Redis instances.

Constraints

- For GeminiDB Redis instances in a proxy cluster or Redis Cluster, the top 30 hot keys that are accessed most frequently can be diagnosed. For primary/standby instances, the top 20 hot keys that are accessed most frequently can be diagnosed.
- After audit log is enabled, hot key diagnosis history will be audited.
- A key that receives 1,000 or more queries per second (QPS) is considered a hot key. When the QPS value is greater than 6,000, the accurate value is not collected.

Procedure

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the target instance and click its name.

Step 4 In the navigation pane on the left, choose **Diagnosis Analysis**.

Step 5 Choose **Hot Key Diagnosis**.

Figure 4-155 Hot key diagnosis

Table 4-52 Parameters for hot key diagnosis

Parameter	Description
Name	Name of a hot key.
Type	Type of a hot key, which can be **string**, **hash**, **list**, **set**, or **sorted set**.
Latest Command	Latest command executed for the hot key.
QPS	Number of accesses to a hot key per second. **NOTE** The maximum QPS that can be displayed is 6,000. If a QPS exceeds 6,000, the accurate value will not be collected.
DB	Database where a hot key is located.

----End

4.10 Account and security

4.10.1 Enabling Password-Free Access

Constraints

Password-free access can be enabled for a maximum of 30 CIDR blocks.

Procedure

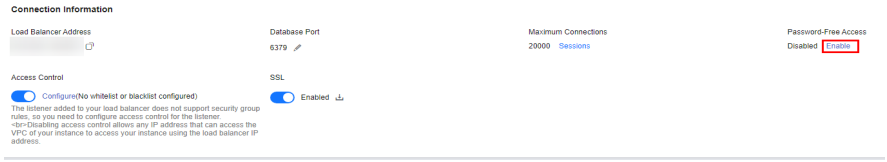
Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 Click the instance name to go to the **Basic Information** page.

Step 4 In the **Connection Information** area, click **Enable**.

Figure 4-156 Connection information



Step 5 In the displayed dialog box, enter the CIDR block that you want to enable password-free access for.

Figure 4-157 Configuring password-free access

- To add a CIDR block, click .
- To delete a CIDR block, click .

Step 6 Click **OK**.

----End

FAQs

Can I access an instance using a password if the instance supports password-free IP address?

Yes. A GeminiDB Redis instance can be accessed no matter you use a correct password or not.

4.10.2 ACL Account Management

Scenarios

GeminiDB Redis API provides the enterprise-grade multi-tenant capability. You can add read-only or read/write accounts in your instance to control access to each database to avoid misoperations on data of other tenants. This section describes how to manage accounts.

Precautions

- A maximum of 200 ACL accounts can be created for each GeminiDB Redis instance.

- Account change takes effect 10 seconds later after it is performed.
- If you use a backup to restore data to a new instance, the account information of the original instance will not be inherited.
- The account to be created must meet the requirements in **Table 4-53**.

Table 4-53 Parameter requirements

Parameter	Requirement	Example Value
Account Name	• Cannot be left blank. • Can contain a maximum of 36 characters. • Must start with a letter and can contain only digits, letters, underscores (_), and hyphens (-).	Organization
Permission	• **Read/Write** • **Read-only**	**Read/Write**
Database	• **Authorize all databases** • **Unauthorized** • **Authorized** **NOTE** • You can add a database on the right of **Database** as required. • You can select the databases to be authorized as required. • **Database** refers to the DB of the open-source Redis.	**Authorize all databases**
Password	• Cannot be left blank. • Can include 8 to 32 characters. • Must contain at least two of the following types: uppercase letters, lowercase letters, digits, and special characters. The following special characters are allowed: ~!@#%^*-_=+?$()&	test123456

Creating an ACL Account

Step 1 Log in to the Huawei Cloud console.

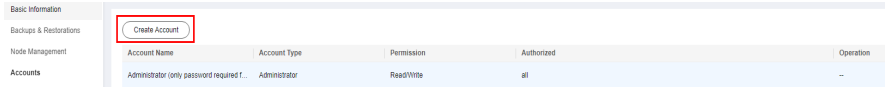
Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the instance list, select an instance and click its name to go to the **Basic Information** page.

Step 4 In the navigation pane on the left, choose **Accounts**.

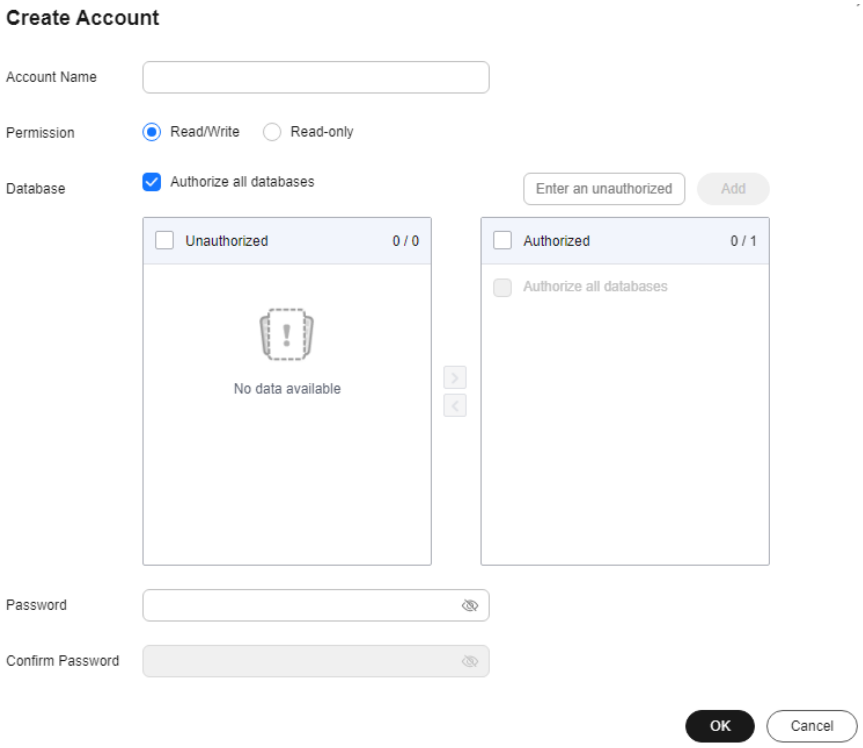
Step 5 On the displayed page, click **Create Account**.

Figure 4-158 Creating an account



Step 6 In the displayed dialog box, enter a username, select permissions for databases, authorize required databases (DBs), enter a password, confirm the password, and click **OK**.

Figure 4-159 Configuring required parameters



Step 7 View and manage the created account in the account list.

----End

Resetting a Password

Step 1 **Log in to the Huawei Cloud console.**

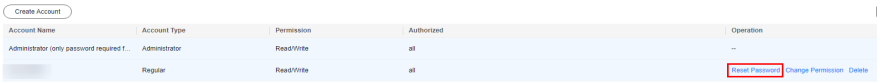
Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the instance list, locate the target instance and click its name to go to the **Basic Information** page.

Step 4 In the navigation pane on the left, choose **Accounts**.

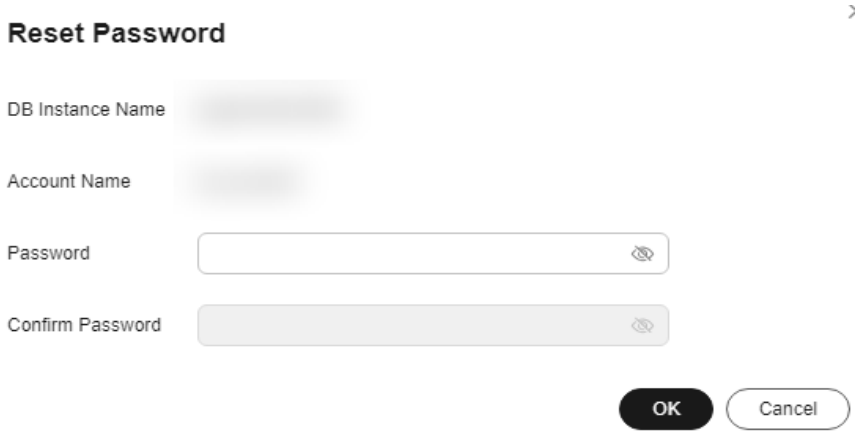
Step 5 On the displayed page, select the account whose password needs to be reset and click **Reset Password**.

Figure 4-160 Resetting a password



Step 6 In the displayed dialog box, enter a new password, confirm the password, and click **OK**.

Figure 4-161 Resetting a password



Step 7 Run **auth** *USER PWD* or **auth** *USER:PWD* to log in to the instance again.

----End

Changing Permissions of an Account

Step 1 **Log in to the Huawei Cloud console.**

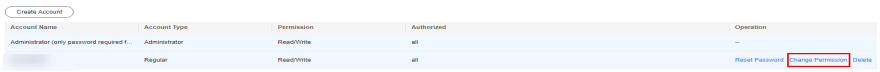
Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the instance list, locate the target instance and click its name to go to the **Basic Information** page.

Step 4 In the navigation pane on the left, choose **Accounts**.

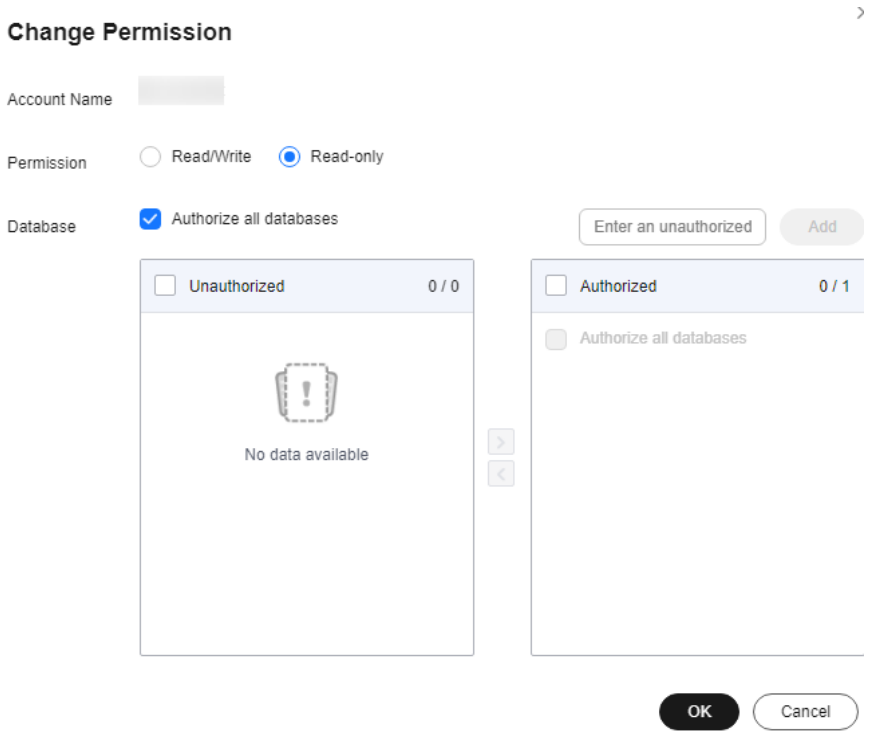
Step 5 On the displayed page, select the account whose permissions need to be modified and click **Change Permission**.

Figure 4-162 Changing permissions



Step 6 In the displayed dialog box, select the required permissions and database, and click **OK**.

Figure 4-163 Changing permissions



Step 7 Run **auth** *USER PWD* or **auth** *USER:PWD* to log in to the instance again.

----End

Deleting an ACL Account

Step 1 **Log in to the Huawei Cloud console.**

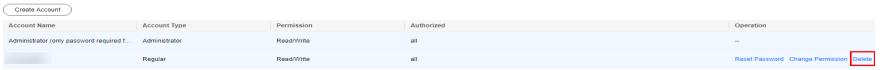
Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the instance list, locate the target instance and click its name to go to the **Basic Information** page.

Step 4 In the navigation pane on the left, choose **Accounts**.

Step 5 On the displayed page, locate the account that you want to delete and click **Delete**. In the displayed dialog box, click **OK**.

Figure 4-164 Deleting an account



Step 6 Verify that the deleted account is no longer displayed.

----End

How to Use a New Account to Access a Database

1. Access the database by running **auth** *USER PWD* or **auth** *USER:PWD*.
2. When you access the database using an SDK on your application, use the value of *USER* and *PWD* as the username and password, or use *USER:PWD* as the password parameter.

   When you access the database by running **auth** *argc* , ensure that *argc* does not contain colons. If an incorrect password contains colons, the returned value is the same as that of **auth** *argc1 argc2*.

4.10.3 Enabling Automated Database Redirection for ACL Accounts

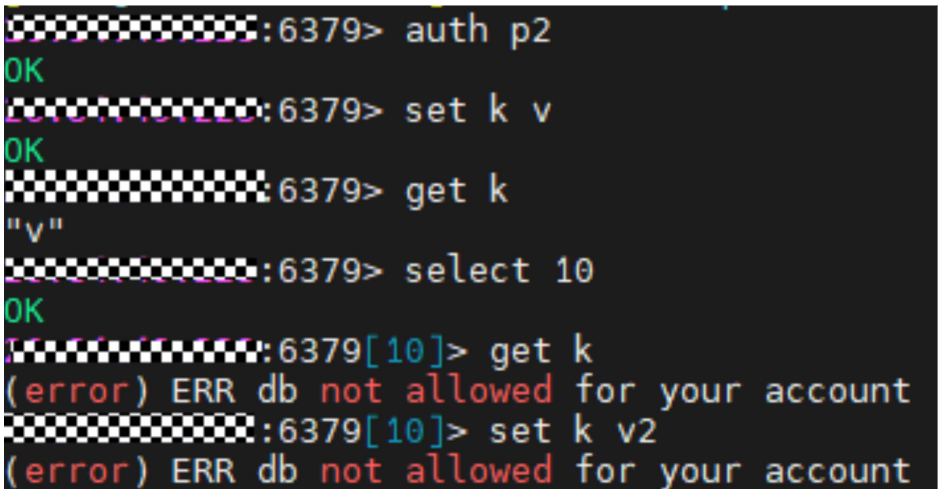
Scenarios

Multiple ACL accounts can be created for a GeminiDB Redis instance so that multiple applications can share one instance, helping DBA effectively reduce costs. Different ACL accounts can isolate databases (for example, DB0, DB1, and DB2) to prevent misoperations. Generally, account management supports only **auth** *<user> <pwd>* or **auth** *<user:pwd>*. If you do not enter **<user>** or it is inconvenient to change *<pwd>* to *<user:pwd>* in the code, you can use automated database redirection by executing **auth** *<pwd>* to isolate databases among multiple accounts.

For example:

1. An account (username **user1** and password **p1**) has been set to access only DB 10 in service A.
2. An account (username **user2** and password **p2**) has been set to access only DB 18 in service B.

Applications usually want fewer program changes. For example, to retain the default value of the database parameter of a client instead of setting it to **18** for application B, you can enable automated database redirection so that the account can be automatically redirected to database 18 using only a password. Even if SELECT is mistakenly executed on DB 10 for application B, application A is not affected.



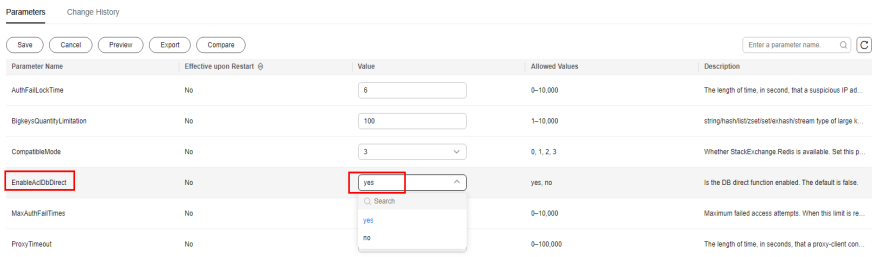
Precautions

- Only one database can be specified for each ACL account for automated redirection. Otherwise, the authentication fails.
- The password of a new account cannot be the same as an existing password. Otherwise, the authentication fails.

Enabling Automated Database Redirection for ACL Accounts

Set **EnableAclDbDirect** to **yes** to enable this feature. For details, see **Modifying Parameters of an Instance**.

Figure 4-165 Parameters



How to Use a New Account to Access a Database

1. Run **auth** *PWD*.
2. When you access a database using an SDK, use *PWD* as the password parameter.

   When you access a database by running **auth** *argc*, ensure that *argc* does not contain colons. If an incorrect password contains colons, the returned value is the same as that of **auth** *argc1 argc2*.

4.10.4 Brute Force Attack Defense

- Brute-force attack defense mechanism

  GeminiDB Redis enables brute force attack defense by default, to automatically lock out an IP address after 5 failed authentication attempts.
- Automatic unlocking

  After an IP address is locked for 5s, the IP address is automatically unlocked and can be authenticated again.
- Manual unlocking

  To manually unlock the IP address or disable anti-brute-force attack, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service for authorization.

> **CAUTION**
>
> To improve security, you can submit a service ticket and ask technical engineers to help adjust authentication times and locking duration.
>
> Make sure to fully evaluate risks and exercise caution when you disable or modify the security policy. After you adjust the security policy, risks and accidents incurred will not be accounted in the SLA and shall be borne by yourself.

4.11 Parameter Management

4.11.1 Modifying Parameters of GeminiDB Redis Instances

To ensure optimal GeminiDB Redis performance, you can modify instance parameters based on service requirements.

Precautions

- You can directly modify parameters on the parameter modification page of an instance.
- Note that parameter values in default parameter templates cannot be changed.
- Though parameter values in a default template cannot be changed, you can view details about a default parameter template.
- If a custom parameter template is set incorrectly, the database startup may fail. You can re-configure the custom parameter template according to the configurations of the default parameter template.

> **CAUTION**
>
> Exercise caution when modifying parameter values to prevent exceptions.

Modifying Parameters of an Instance

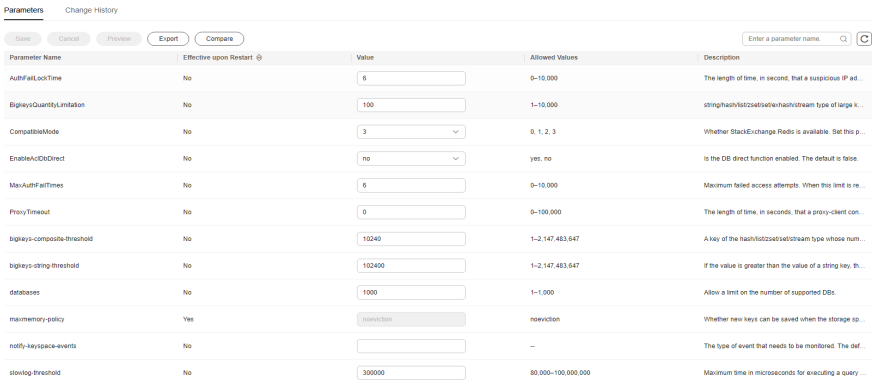
Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the navigation pane on the left, choose **Instances**. On the displayed page, locate the instance whose parameters you want to modify and click its name.

Step 4 In the navigation pane, choose **Parameters**. On the displayed page, modify parameters listed in **Table 4-54**.

Figure 4-166 Parameters



- To save the modifications, click **Save**.
- To cancel the modifications, click **Cancel**.
- To preview the modifications, click **Preview**.

Table 4-54 GeminiDB Redis instance parameters

Parameter	Effective upon Restart	Default Value	Value Range	Description	Exception
AuthFailLockTime	No	5	0–10,000	Authentication failure lock time, in seconds. This parameter specifies the duration during which a suspicious IP address is locked. After the duration expires, the IP address is automatically unlocked.	GeminiDB Redis instances in Redis Cluster do not support this parameter.

Parameter	Effective upon Restart	Default Value	Value Range	Description	Exception
BigkeysQuantityLimitation	No	100	1–10,000	Maximum number of big keys of the STRING, HASH, LIST, ZSET, SET, EXHASH, and STREAM data types that can be queried.	GeminiDB Redis instances in Redis Cluster do not support this parameter.

Parameter	Effective upon Restart	Default Value	Value Range	Description	Exception
CompatibleMode	No	3	0, 1, 2, 3	Adaptation switch for the StackExchange.Redis client. If StackExchange.Redis reports error "Multiple databases are not supported on this server", change the parameter value to **0**.	GeminiDB Redis instances in Redis Cluster do not support this parameter.

Parameter	Effective upon Restart	Default Value	Value Range	Description	Exception
EnableAclDbDirect	No	no	yes, no	Whether to enable direct database access The default value is **no**.	GeminiDB Redis instances in Redis Cluster do not support this parameter.

Parameter	Effective upon Restart	Default Value	Value Range	Description	Exception
MaxAuthFailTimes	No	5	0–10,000	Maximum number of authentication attempts permitted per connection. When the number of incorrect password attempts reaches the threshold, the instance will forbid access from a suspicious IP address for a short period of time. **0** indicates this function is disabled.	GeminiDB Redis instances in Redis Cluster do not support this parameter.

Parameter	Effective upon Restart	Default Value	Value Range	Description	Exception
ProxyTimeout	No	0	0–100,000	Timeout (in seconds) when a proxy receives no response from a client. When the timeout reaches the threshold, the proxy proactively closes the connection. If the value is **0**, the proxy will not proactively disconnect the connection.	GeminiDB Redis instances in Redis Cluster do not support this parameter.

Parameter	Effective upon Restart	Default Value	Value Range	Description	Exception
enable-acl-direct	No	no	yes, no	Whether to enable direct database access The default value is **no**.	Primary/Standby GeminiDB Redis instances and GeminiDB Redis instances in Redis Cluster do not support this parameter..

Parameter	Effective upon Restart	Default Value	Value Range	Description	Exception
bigkeys-composite-threshold	No	1024	1–2,147,483,647	If the number of elements in a HASH, LIST, ZSET, SET, or STREAM key is greater than the value of this parameter, the key is identified as a big key. The default value is **1024**.	-
bigkeys-string-threshold	No	102400	1–2,147,483,647	If the size of a STRING key is greater than the value of this parameter, the key is determined as a big key. The unit is byte. The default value is **102400**.	-
databases	No	1000	1–1,000	Maximum number of supported databases	-
key-scan-batch	No	5000	1–2,147,483,647	Number of keys scanned each time	-
maxmemory-policy	Yes	noeviction	noeviction	Key discarding policy after storage space is used up. GeminiDB Redis instance storage can be scaled up in seconds. After storage space is used up, the instance becomes read-only and its service data is retained. Autoscaling will be available later.	-
notify-keyspace-events	No	-	Combination of A, K, E, g, $, l, s, h, z, x, e, and t	Type of an event to be listened on. The default value is empty, indicating that the parameter does not take effect. Combination of A, K, E, g, $, l, s, h, z, x, e, and t	-

Parameter	Effective upon Restart	Default Value	Value Range	Description	Exception
slowlog-threshold	No	300000	80,000 – 100,000,000	Time threshold (in us) used to define when slow queries are logged on the console. A small value may affect instance performance. You are advised to retain the default value.	-

Step 5 After parameters are modified, click **Change History** to view parameter modification details.

For details about how to view parameter modification details, see **4.11.3 Viewing Parameter Change History**.

NOTICE

After you modify instance parameters, the modifications immediately take effect for the instance.

Check the value in the **Effective upon Restart** column.

- If the value is **Yes** and the instance status on the **Instances** page is **Parameter change. Pending reboot**, you must reboot the instance for the modifications to take effect.
- If the value is **No**, the modifications take effect immediately.

----End

Modifying a Custom Parameter Template and Applying It to an Instance

Step 1 **Log in to the Huawei Cloud console.**

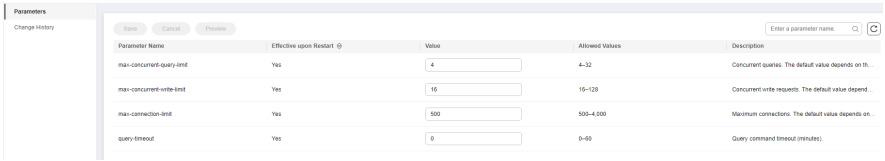
Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the navigation pane on the left, choose **Parameter Templates**.

Step 4 Click the **Custom Templates** tab, locate the parameter template you want to modify, and click its name.

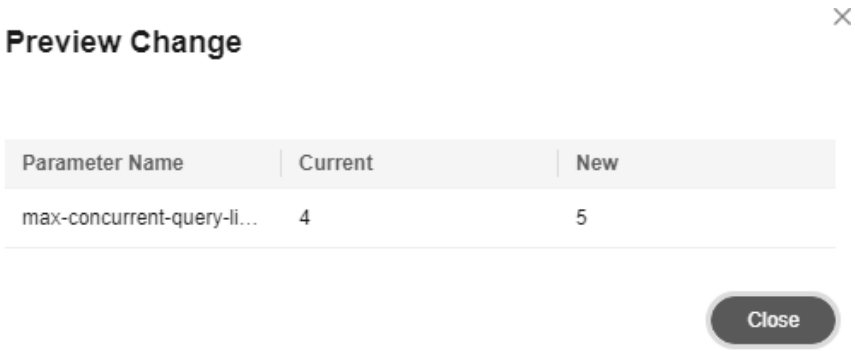
Step 5 Change parameter values as required.

Figure 4-167 Editing a parameter template



- To save the modifications, click **Save**.
- To cancel the modifications, click **Cancel**.
- To preview the modifications, click **Preview**.

Figure 4-168 Preview Change



Step 6 After parameters are modified, click **Change History** to view parameter modification details.

For details about how to view parameter modification details, see **4.11.3 Viewing Parameter Change History**.

NOTICE

- The modifications take effect only after you apply the parameter template to your instance. For details, see **4.11.8 Applying a Parameter Template**.
- The change history page displays only the modifications of the last seven days.

----End

4.11.2 Creating a Parameter Template

You can use database parameter templates to manage DB API configurations. A database parameter template acts as a container for API configuration values that can be applied to one or more DB instances.

Each user can create up to 100 parameter templates. All types of instances in the same project can share the quota.

Procedure

Step 1 **Log in to the Huawei Cloud console.**

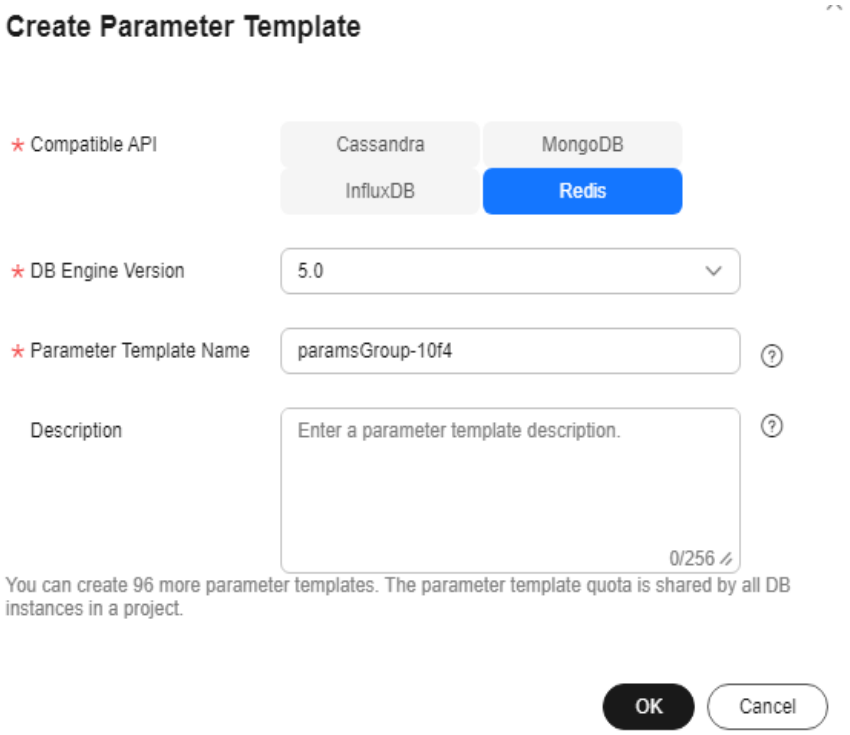
Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the navigation pane on the left, choose **Parameter Templates**.

Step 4 On the **Parameter Templates** page, click **Create Parameter Template**.

Step 5 Select a compatible API, specify a DB engine version and a parameter group description, and click **OK**.

Figure 4-169 Creating a parameter template



- **Compatible API**: Select the API type that is compatible with your DB engine parameter template.
- **DB Engine Version**: Select a DB engine version, for example, 5.0.
- **Parameter Template Name**: The template name is 1 to 64 characters long. It contains only uppercase letters, lowercase letters, digits, hyphens (-), underscores (_), and periods (.).
- **Description**: The description contains a maximum of 256 characters and cannot include line breaks or the following special characters >!<"&'=

Step 6 On the **Parameter Templates** page, view the created parameter template.

----End

4.11.3 Viewing Parameter Change History

Scenarios

You can view parameter change history of an instance or one of its custom parameter templates based on service requirements.

Precautions

In a newly exported or created parameter template, change history is left blank.

Viewing Change History of a Custom Parameter Template

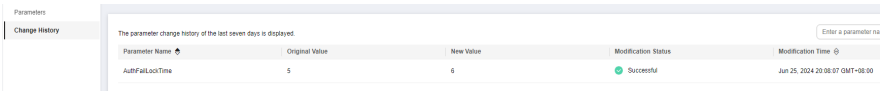
Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the navigation pane on the left, choose **Parameter Templates**. On the **Custom Templates** page, click the parameter template whose change history you want to view.

Step 4 In the navigation pane on the left, choose **Change History**. Then, view the name, original value, new value, modification status, and modification time of the target parameter.

Figure 4-170 Viewing change history of a customer parameter template



After you change a parameter template, you can apply it to an instance based on service requirements by referring to **4.11.8 Applying a Parameter Template**.

----End

Viewing Parameter Change History of an Instance

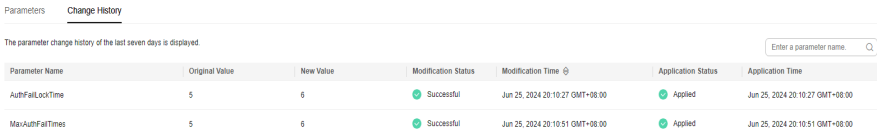
Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the instance whose parameter change history you want to view and click its name.

Step 4 In the navigation pane on the left, choose **Parameters**. On the **Change History** page, view the name, original value, new value, modification status, and modification time of the target parameter.

Figure 4-171 Viewing parameter change history of an instance



----End

4.11.4 Exporting a Parameter Template

- You can export parameters of your instance to a new parameter template for future use. To learn how to apply the parameter template to another instance, refer to **4.11.8 Applying a Parameter Template**.
- You can also export parameter template information (including parameter names, values, and descriptions) of your instance to a CSV file for review and analysis.

Procedure

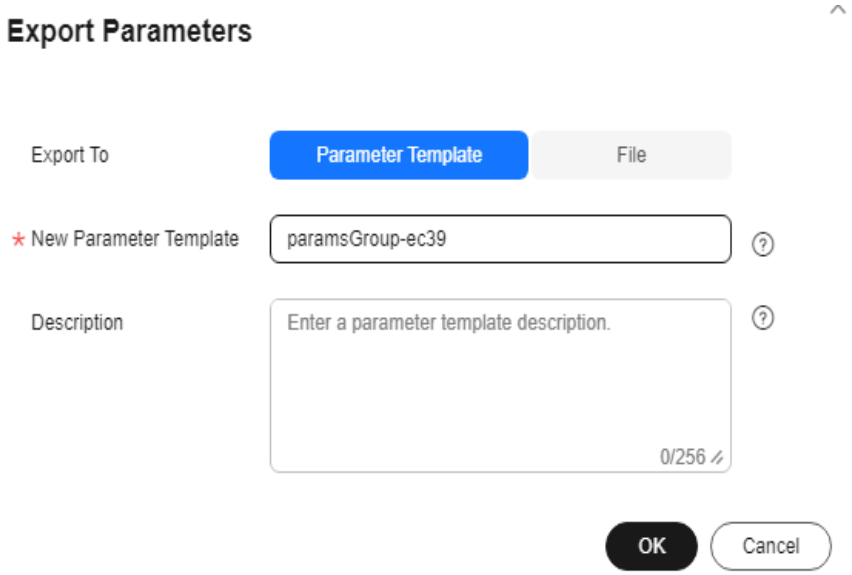
Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the navigation pane on the left, choose **Instances**, locate the instance whose parameters you want to export, and click its name.

Step 4 In the navigation pane on the left, choose **Parameters**. On the **Parameters** page, click **Export**.

Figure 4-172 Exporting a parameter template



- **Parameter Template**: You can export parameters of your instance to a template for future use.

  In the displayed dialog box, configure required parameters and click **OK**.

  NOTE

  - **Parameter Template Name**: The template name can include 1 to 64 characters. It can contain only uppercase letters, lowercase letters, digits, hyphens (-), underscores (_), and periods (.).
  - **Description**: The description can include a maximum of 256 characters and cannot include line breaks or the following special characters: >!<"&'=

  After the export is complete, a new parameter template is generated and displayed on the **Parameter Templates** page.

- **File**: You can export the parameter template information (including parameter names, values, and descriptions) of your instance to a CSV file for review and analysis.

  In the displayed dialog box, enter a file name and click **OK**.

NOTE

The file name must start with a letter and can include 4 to 81 characters. It can contain only letters, digits, hyphens (-), and underscores (_).

----End

4.11.5 Comparing Parameter Templates

This section describes how to compare two parameter templates of the same instance type and compatible API to learn about their configurations.

Comparing Parameter Templates

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the navigation pane on the left, choose **Parameter Templates**.

Step 4 In the parameter template list, locate the parameter template that you created and click **Compare** in the **Operation** column.

Step 5 In the displayed dialog box, select a parameter template that is of the same instance type and compatible API as the selected template and click **OK**.

Figure 4-173 Comparing two parameter templates

- If their parameters are different, the different parameter names and values are displayed.
- If their parameters are the same, no data is displayed.

----End

Comparing Parameter Templates of a Specific Instance

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the navigation pane on the left, choose **Instances**.

Step 4 On the **Instances** page, locate the instance whose parameter templates you want to compare and click its name.

Step 5 In the navigation pane on the left, choose **Parameters** and then click **Compare** above the parameter list.

Step 6 In the displayed dialog box, select a parameter template that is of the same instance type as the template of current instance and click **OK**.

Figure 4-174 Comparing the parameter template of the current instance with another parameter template

- If their parameters are different, the different parameter names and values are displayed.
- If their parameters are the same, no data is displayed.

----End

4.11.6 Replicating a Parameter Template

You can replicate a parameter template that you have created. When you have already created a parameter template and want to use most of the custom parameters and values from that template to a new parameter template, you can replicate that parameter template. You can also export a parameter template of a DB instance for future use.

Default parameter templates cannot be replicated, but you can create custom parameter templates based on the default templates provided.

Procedure

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

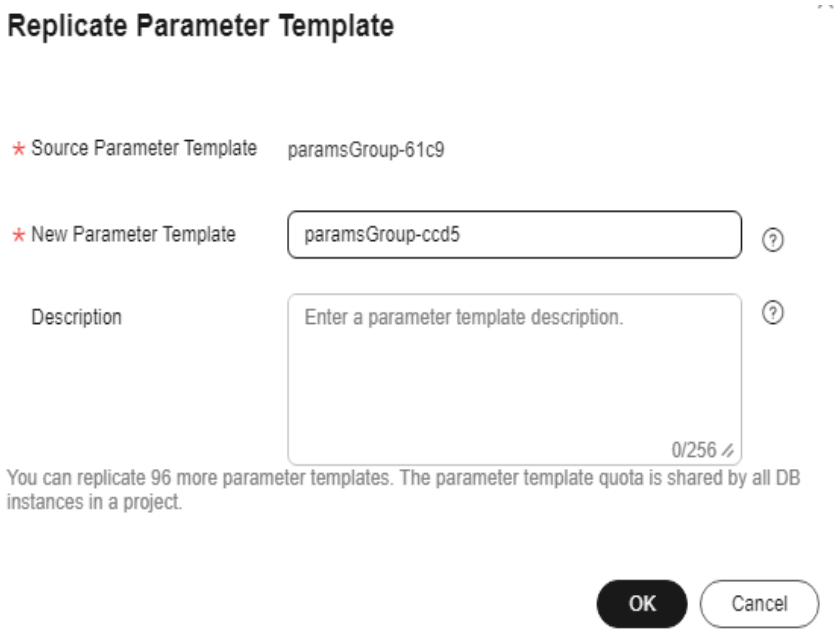
Step 3 In the navigation pane on the left, choose **Parameter Templates**.

Step 4 On the **Parameter Templates** page, click the **Custom Templates** tab. Locate the parameter template that you want to replicate and click **Replicate** in the **Operation** column.

Alternatively, click the instance name on the **Instances** page. On the **Parameters** page, click **Export** to generate a new parameter template for future use.

Step 5 In the displayed dialog box, enter a parameter template name and description and click **OK**.

Figure 4-175 Replicating a parameter template



- **New Parameter Template**: The template name can include 1 to 64 characters. It can contain only uppercase letters, lowercase letters, digits, hyphens (-), underscores (_), and periods (.).
- **Description**: The description can contain a maximum of 256 characters and cannot include line breaks or the following special characters: >!<"&'=

After replication is complete, a new template is generated and displayed on the **Parameter Templates** page.

----End

4.11.7 Resetting a Parameter Template

You can reset all parameters in a custom parameter template to their default settings.

Procedure

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the navigation pane on the left, choose **Parameter Templates**.

Step 4 On the **Parameter Templates** page, click the **Custom Templates** tab. Locate the parameter template whose parameters you want to reset and choose **More** > **Reset** in the **Operation** column.

Step 5 Click **Yes**.

----End

4.11.8 Applying a Parameter Template

GeminiDB Redis allows you to apply a parameter template. After you modify a parameter template, modifications you make do not take effect until you apply the template to an instance.

Procedure

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the navigation pane on the left, choose **Parameter Templates**.

Step 4 On the **Parameter Templates** page, perform the following operations as follows:

- To apply a default template, click **Default Templates**, locate the target parameter template, and click **Apply** in the **Operation** column.
- To apply a custom template, click **Custom Templates**, locate the target parameter template, and choose **More** > **Apply** in the **Operation** column.

A parameter template can be applied to one or more instances.

Step 5 In the displayed dialog box, select one or more instances to which the parameter template will be applied and click **OK**.

After a parameter template is applied, you can **view its application records**.

----End

4.11.9 Viewing Application Records of a Parameter Template

GeminiDB Redis allows you to view application records of a parameter template.

Procedure

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the navigation pane on the left, choose **Parameter Templates**.

Step 4 On the displayed page, locate the parameter template whose application records you want to view and choose **More** > **View Application Records** in the **Operation** column.

You can view the name or ID of the instance that the parameter template is applied to, as well as the application status, application time, and failure cause.

----End

4.11.10 Modifying the Description of a Parameter Template

You can modify the description of a custom parameter template if needed.

Procedure

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the navigation pane on the left, choose **Parameter Templates**.

Step 4 Click the **Custom Templates** tab. Locate the parameter template whose description you want to modify and click  in the **Description** column.

Step 5 Enter a new description and click  to submit or  to cancel the modification.

- After you submit the modification, you can view the new description in the **Description** column in the parameter template list.
- The description can include a maximum of 256 characters but cannot contain the following special characters: >!<"&'=

----End

4.11.11 Deleting a Parameter Template

You can delete a custom parameter template that is no longer in use.

Precautions

- Deleted templates cannot be recovered, so exercise caution when performing this operation.
- Default parameter templates cannot be deleted.

Procedure

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the navigation pane on the left, choose **Parameter Templates**.

Step 4 On the **Parameter Templates** page, click the **Custom Templates** tab. Locate the parameter template you wish to delete and choose **More** > **Delete** in the **Operation** column.

Step 5 Click **Yes** to delete the parameter template.

----End

4.12 Logs and Audit

4.12.1 Enabling or Disabling Log Reporting

Scenarios

If you enable log reporting for your GeminiDB Redis instance, new logs generated for the instance will be uploaded to Log Tank Service (LTS).

Precautions

- To use this function, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service.
- You will be billed for enabling this function.
- Ensure that there are available LTS log groups and log streams in the same region as your instance.

  For more information about log groups and log streams, see **Log Management**.

Enabling Log Reporting to LTS in Batches

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the navigation pane, choose **Log Reporting**.

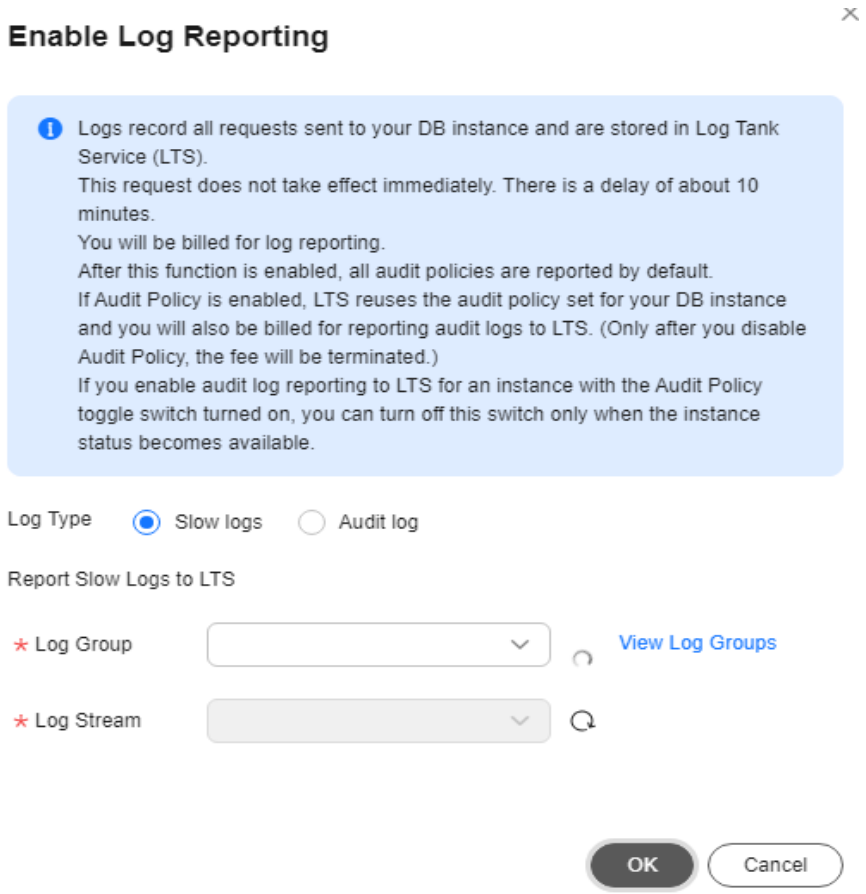
Step 4 Select one or more instances and click **Enable Log Reporting**.

Step 5 Select an LTS log group and log stream and click **OK**.

NOTE

- This function does not take effect immediately. There is a delay of about 10 minutes.
- You will be billed for enabling this function. For details, see LTS **pricing details**.

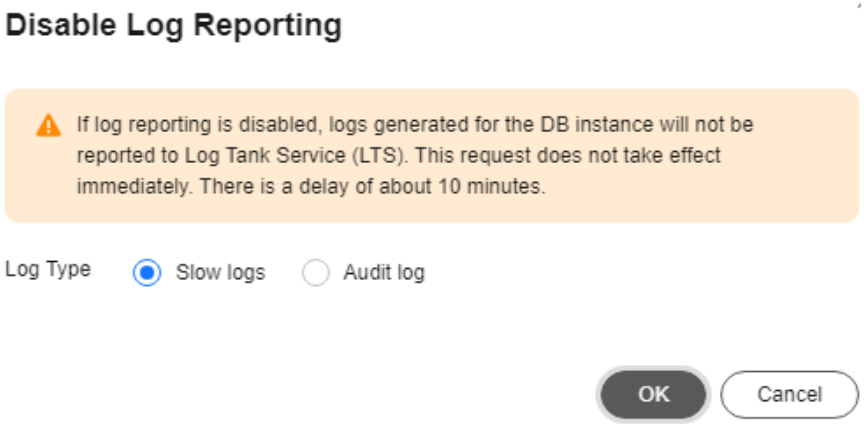
Figure 4-176 Enabling log reporting



Step 6 To disable log reporting, select one or more instances and click **Disable Log Reporting**.

Step 7 In the displayed dialog box, click **OK**.

Figure 4-177 Disabling log reporting



----End

4.12.2 Viewing and Exporting Slow Query Logs

GeminiDB Redis allows you to view slow query logs of databases. The unit of the execution time is ms. You can identify the SQL statements that take a long time to execute and tune them based on slow query logs.

Reporting Slow Query Logs to LTS

To use this function, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service.

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the navigation pane, choose **Log Reporting**.

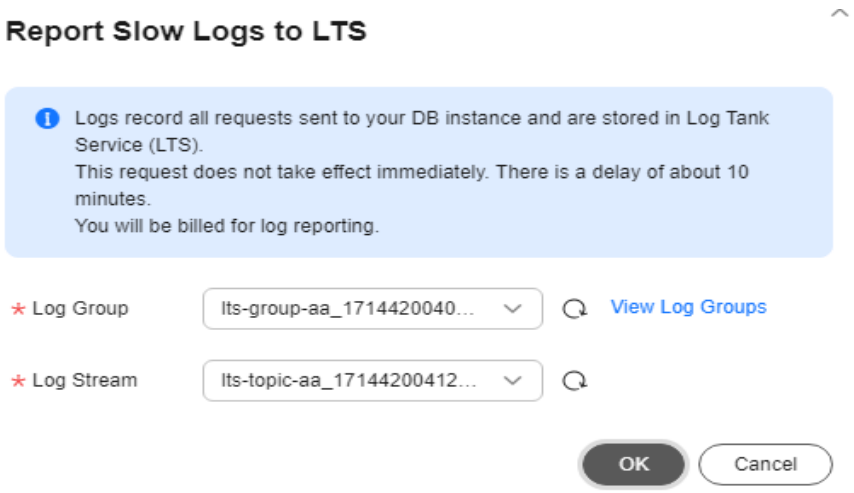
Step 4 Select an instance and click  next to **Report Slow Log to LTS**.

Step 5 Select an LTS log group and log stream and click **OK**.

NOTE

You will be billed for enabling this function. For details, see LTS **pricing details**.

Figure 4-178 Reporting slow query logs to LTS



----End

Viewing and Exporting Log Details

Step 1 **Log in to the Huawei Cloud console.**

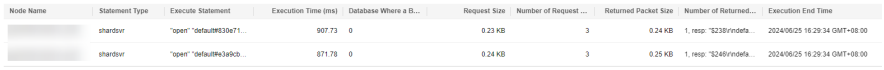
Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the target instance.

Step 4 In the navigation pane on the left, choose **Slow Query Logs**.

Step 5 On the **Slow Query Logs** page, set search criteria and click **Search** to view log information.

Figure 4-179 Viewing slow query logs

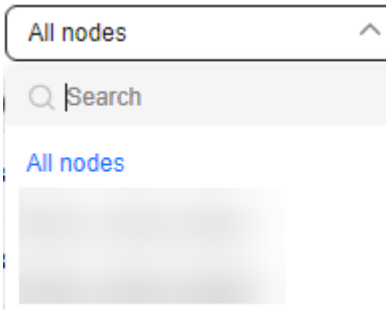


NOTE

For some instances of earlier versions, the kernel minor version needs to be upgraded to support metrics Database Where a Big Key Is Located, Request Size, Number of Request Parameters, Returned PACKET Size, and Number of Returned Values.

- Select **All nodes** and view slow query logs of all nodes. Alternatively, select a specific node to view its slow query logs.

Figure 4-180 Querying nodes



- Choose to view slow query logs of all types of SQL statements or a specific SQL statement.
  - SET
  - GET
  - DEL
  - INCR
  - INCRBY
  - INCRBYFLOAT
  - DECR
  - DECRBY
  - GETSET
  - APPEND
  - MGET

  ……

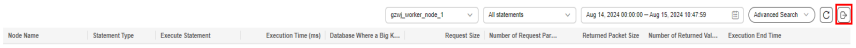
  NOTE

  You can view slow query logs of all types of Redis SQL statements.
- View slow query logs of a specific node in different time ranges.

Step 6 On the **Log Details** page, click in the upper right corner of the log list to export log details.

- You can view the CSV file exported to your local PC.
- Up to 2,000 logs can be exported at a time.

Figure 4-181 Exporting slow query logs



----End

4.12.3 Viewing Audit Logs

GeminiDB Redis allows you to view audit logs of databases. You can analyze, search for, monitor, download, and view real-time logs on the LTS console.

Usage Notes

- No audit logs are generated for operations on internal connections.
- Commands that are always audited include BIGKEYS, KEYS, FLUSHALL, FLUSHDB, SCRIPT, CLIENT, and CONFIG.
- Audit logs are only generated for the following commands that you need to configure many parameters for:

  BITOP, MSETNX, PFCOUNT, PFMERGE, HDEL, HMGET, HMSET, HSET, LPUSH, LPUSHX, SADD, SREM, ZADD, GEOADD, GEOHASH, BFINSERT, BFMADD, and BFMEXISTS.
- An audit log is generated only when EXEC executes more than 100 commands in a transaction.

- Logs of GeminiDB Redis instances in Redis Cluster cannot be audited.

Enabling Audit Log Reporting to LTS

To use this function, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact customer service.

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the navigation pane, choose **Log Reporting**.

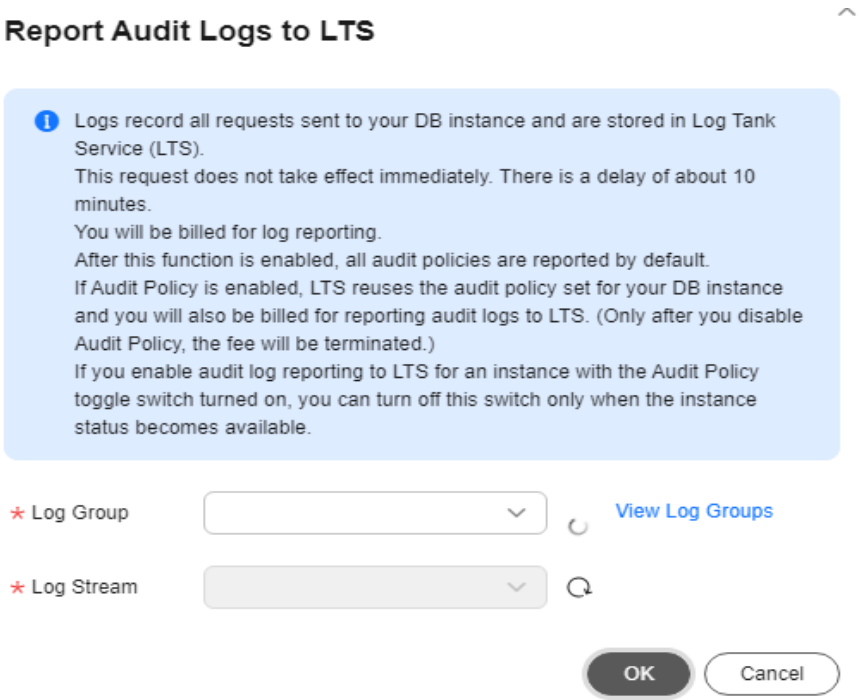
Step 4 Locate the row containing your target instance and click  in the **Report Audit Log to LTS** column.

Step 5 Select an LTS log group and log stream and click **OK**.

NOTE

You will be billed for enabling this function. For details, see LTS **pricing details**.

Figure 4-182 Enabling audit log reporting to LTS



----End

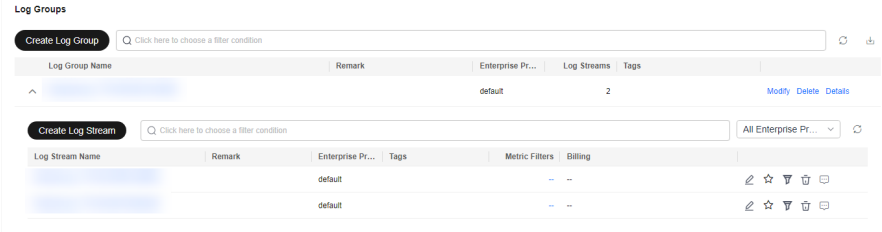
Viewing Log Details

Step 1 Log in to the Huawei Cloud console.

Step 2 Click  in the upper left corner of the page. Under **Management & Governance**, click **Log Tank Service**.

Step 3 On the **Log Management** page, click **Log Groups**, select a log group and log stream, and click the name of the selected log stream to go to the details page.

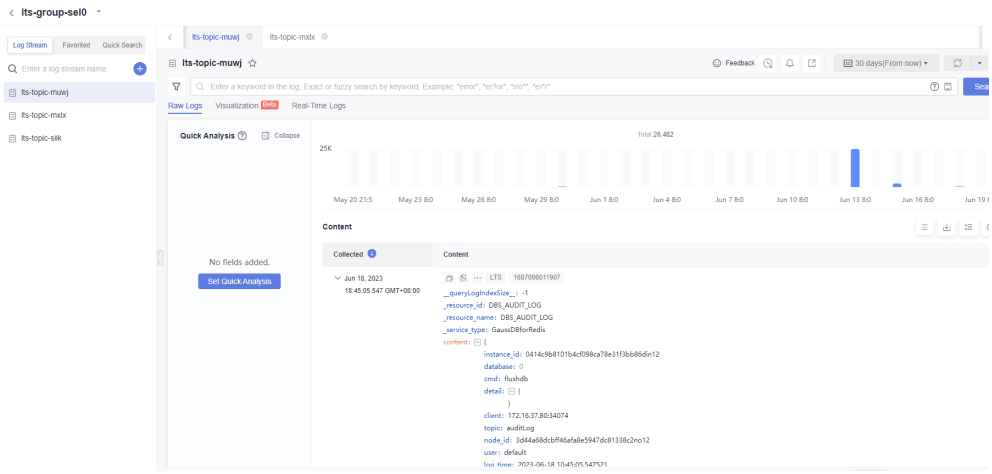
Figure 4-183 Selecting a log stream



Step 4 Set **From now** to select a relative time (15 minutes by default) in the upper right corner as needed.

Step 5 On the **Raw Logs** tab, view the audit logs generated in the relative time period.

Figure 4-184 Audit logs



----End

4.12.4 CTS Audit

4.12.4.1 Key Operations Supported by CTS

With CTS, you can record GeminiDB Redis key operations for later query, audit, and backtracking.

Table 4-55 GeminiDB Redis key operations

Operation	Resource Type	Trace Name
Creating an instance	instance	NoSQLCreateInstance
Deleting an instance	instance	NoSQLDeleteInstance

Operation	Resource Type	Trace Name
Adding nodes	instance	NoSQLEnlargeInstance
Deleting nodes	instance	NoSQLReduceInstance
Restarting an instance	instance	NoSQLRestartInstance
Restoring data to a new instance	instance	NoSQLRestoreNewInstance
Scaling up storage space of an instance	instance	NoSQLExtendInstanceVolume
Resetting the password of an instance	instance	NoSQLResetPassword
Modifying the name of an instance	instance	NoSQLRenameInstance
Binding an EIP	instance	NoSQLResizeInstance
Unbinding an EIP	instance	NoSQLBindEIP
Changing specifications	instance	NoSQLUnBindEIP
Freezing an instance	instance	NoSQLFreezeInstance
Unfreezing an instance	instance	NoSQLUnfreezeInstance
Creating a backup	backup	NoSQLCreateBackup
Deleting a backup	backup	NoSQLDeleteBackup
Setting a backup policy	backup	NoSQLSetBackupPolicy
Adding an instance tag	tag	NoSQLAddTags
Modifying an instance tag	tag	NoSQLModifyInstanceTag
Deleting an instance tag	tag	NoSQLDeleteInstanceTag
Creating a parameter template	parameterGroup	NoSQLCreateConfigurations
Modifying a parameter template	parameterGroup	NoSQLUpdateConfigurations
Modifying instance parameters	parameterGroup	NoSQLUpdateInstanceConfigurations
Replicating a parameter template	parameterGroup	NoSQLCopyConfigurations
Resetting a parameter template	parameterGroup	NoSQLResetConfigurations
Applying a parameter template	parameterGroup	NoSQLApplyConfigurations

Operation	Resource Type	Trace Name
Deleting a parameter template	parameterGroup	NoSQLDeleteConfigurations
Deleting the node that fails to be added	instance	NoSQLDeleteEnlargeFail-Node
Enabling SSL	instance	NoSQLSwitchSSL
Changing the security group of an instance	instance	NoSQLModifySecurityGroup
Modifying the recycling policy	instance	NoSQLModifyRecyclePolicy

4.12.4.2 Querying Traces

Scenarios

After CTS is enabled, CTS starts recording operations on cloud resources. The CTS console stores the last 7 days of operation records for later query, audit, and backtracking.

This section describes how to query the last 7 days of operation records on the CTS console.

Procedure

Step 1 **Log in to the Huawei Cloud console.**

Step 2 Click  in the upper left corner and select a region and a project.

Step 3 Click **Service List**. Under **Management & Governance**, click **Cloud Trace Service**.

Step 4 Choose **Trace List** in the navigation pane on the left.

Step 5 Click **Filter** and specify filter criteria as needed. The following filters are available:

- **Trace Type**: Select **Management** or **Data**.
- **Trace Source**, **Resource Type**, and **Search By**

  Select a filter from the drop-down list.

  When you select **Trace name** for **Search By**, you also need to select a specific trace name.

  When you select **Resource ID** for **Search By**, you also need to select or enter a specific resource ID.

  When you select **Resource name** for **Search By**, you also need to select or enter a specific resource name.
- **Operator**: Select a specific operator (a user rather than tenant).
- **Trace Status**: Available options include **All trace statuses**, **normal**, **warning**, and **incident**. You can only select one of them.

- Start time and end time: You can specify a time range for querying traces.

Step 6 Click ⌄ on the left of the record to be queried to extend its details.

Step 7 Locate a trace and click **View Trace** in the **Operation** column.

----End

4.13 Viewing Metrics and Configuring Alarms

4.13.1 GeminiDB Redis Instance Metrics

Description

This section describes GeminiDB Redis instance metrics reported to Cloud Eye as well as their namespaces and dimensions. You can use APIs provided by Cloud Eye to query the metrics and alarms.

You can view the instance-level and node-level GeminiDB Redis metrics described in GeminiDB Redis on each instance node by referring to **4.13.4 Viewing GeminiDB Redis Instance Metrics**. The instance-level metrics displayed on each instance node are the same.

Table 4-56 Metric classification

Metric Level	Metric Type
Instance	Instance metrics
Node	Basic metrics **String command metrics** **Hashes command metrics** **Lists command metrics** **Set command metrics.** **Zset command metrics** **Bitmap command metrics** **Stream command metrics** **Geo command metrics** **Hyperloglog command metrics** **Pub/Sub command metrics** **Scripting command metrics** **Transactions command metrics** **Common command metrics**

Namespace

SYS.NoSQL

Instance Metrics

Table 4-57 Instance metrics

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis816_cluster_disk_usage	Storage Usage	Storage usage of an instance Unit: %	0–100	GeminiDB Redis instances	1 minute
redis813_cluster_slow_query_count	Slow Queries per Instance	Number of slow queries on an instance Unit: count	≥ 0	GeminiDB Redis instances	1 minute
redis812_cluster_processed_command_count	Total Commands per Instance Total commands processed by an instance	Total commands processed by a node Unit: count	≥ 0	GeminiDB Redis instances	1 minute
redis811_cluster_max_connect_count	Max. Connections per Instance	Maximum connections to an instance Unit: count	≥ 0	GeminiDB Redis instances	1 minute
redis808_cluster_new_client_connection	Connections Created Per Second	Instance connections created per second Unit: count	≥ 0	GeminiDB Redis instances	1 minute
redis807_cluster_all_connection_count	Total Connections	Total connections to the instance Unit: count	≥ 0	GeminiDB Redis instances	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis806_cluster_max_connection_usage	Connection Usage	Max. node connection usage of the instance Unit: %	0–100	GeminiDB Redis instances	1 minute
redis805_cluster_avg_hit_rate	Average Hit Rate of the Instance	Average hit rate of multiple nodes in the instance Unit: %	0–100	GeminiDB Redis instances	1 minute
redis804_cluster_all_p99_usec	p99 Latency	p99 latency of the instance Unit: µs	≥ 0	GeminiDB Redis instances	1 minute
redis803_cluster_all_avg_usec	Average Latency	Average latency of the instance Unit: µs	≥ 0	GeminiDB Redis instances	1 minute
redis802_cluster_max_response_argc	Max. Elements Obtained in a Request	Max. elements obtained by the client in a request Unit: count	≥ 0	GeminiDB Redis instances	1 minute
redis801_cluster_max_response_bytes	Max. Bytes Obtained in a Request	Max. bytes obtained by the client in a request Unit: byte	≥ 0	GeminiDB Redis instances	1 minute
redis800_cluster_max_request_argc	Max. Parameters Sent in a Request	Max. parameters sent in a request Unit: count	≥ 0	GeminiDB Redis instances	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis799_cluster_max_request_bytes	Max. Bytes Sent in a Request	Max. bytes sent in a request Unit: byte	≥ 0	GeminiDB Redis instances	1 minute
redis798_cluster_expire_key_counts	Keys with an Expiration Time Configured	Instance keys with an expiration time configured Unit: count	≥ 0	GeminiDB Redis instances	1 minute
redis689_qps_receive_total	Total Traffic Received by the Instance	Total traffic received by the instance, a reflection of the traffic volume on the application side. Unit: byte/s	≥ 0	GeminiDB Redis instances	1 minute
redis688_qps_send_total	Total Traffic Sent by the Instance	Total traffic sent by the instance, a reflection of the traffic volume on the application side. Unit: byte/s	≥ 0	GeminiDB Redis instances	1 minute
redis668_cluster_key_counts	Instance Keys	Total keys of a cluster Unit: count	≥ 0	GeminiDB Redis instances	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis667_cluster_qps	QPS of the Instance	The value of this metric is the QPS of the instance. Unit: count/s	≥ 0	GeminiDB Redis instances	1 minute

Node Metrics

Table 4-58 Basic metrics

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis817_slow_query_count	Slow Queries per Node	Slow queries on a node Unit: count	≥ 0	GeminiDB Redis instance nodes	1 minute
redis810_max_connect_count	Max. Connections per Node	Maximum connections to a node Unit: count	≥ 0	GeminiDB Redis instance nodes	1 minute
redis809_processed_command_count	Total Commands per Node	Total commands processed by a node Unit: count	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis687_client_network_packet_return_p99	p99 Latency for Packets Returning to the Client (Send)	p99 latency for packets sent from the proxy to the client, a reflection of the network quality on the application side. Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis686_client_network_packet_return_max	Maximum Speed for Packets Returning to the Client (Send)	Maximum speed for packets sent by the proxy to the client, a reflection of the network quality on the application side. Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis685_client_network_packet_return_avg	Average Speed for Packets Returning to the Client (Send)	Average speed for packets sent by the proxy to the client, a reflection of the network quality on the application side. Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis684_client_packet_return_queue_p99	p99 Latency for Packets Returning to the Client (Queuing)	Queuing p99 latency for packets returned from the proxy to the client, a reflection of the network quality on the application side. Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis683_client_packet_return_queue_max	Maximum Speed for Packets Returning to the Client (Queuing)	Maximum queuing speed for packets returned from the proxy to the client, a reflection of the network quality on the application side. Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis682_client_packet_return_queue_avg	Average Speed for Packets Returning to the Client (Queuing)	Average queuing speed for packets returned from the proxy to the client, a reflection of the network quality on the application side. Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis678_max_response_argc	Maximum Elements Obtained in a Request	Maximum elements obtained by the client in a request Unit: count	≥ 0	GeminiDB Redis instance nodes	1 minute
redis677_max_response_bytes	Maximum Bytes Obtained in a Request	Maximum bytes obtained by the client in a request Unit: byte	≥ 0	GeminiDB Redis instance nodes	1 minute
redis676_max_request_argc	Maximum Parameters Sent in a Request	Maximum parameters sent by the client in a request Unit: count	≥ 0	GeminiDB Redis instance nodes	1 minute
redis675_max_request_bytes	Maximum Bytes Sent in a Request	Maximum bytes sent by the client in a request Unit: byte	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis674_max_pipelined	Maximum Commands Sent in a Pipeline	Maximum commands batch sent by the client in a pipeline Unit: count	≥ 0	GeminiDB Redis instance nodes	1 minute
redis673_wrong_auth	Failed Authentication Attempts	Failed authentication attempts per second on a node Unit: count	≥ 0	GeminiDB Redis instance nodes	1 minute
redis672_no_auth	Request Attempts Due to Authentication Failure	Failed request attempts due to authentication failure on a node per second Unit: count	≥ 0	GeminiDB Redis instance nodes	1 minute
redis671_new_client_connection	New Connections	Connections created on a node per second Unit: count	≥ 0	GeminiDB Redis instance nodes	1 minute
redis670_hit_rate	Hit Rate of a Key in Underlying Storage	Hit percentage of a key in underlying storage in a collection period. Formula: Hit keys/(Hit keys + Missed keys). Unit: %	0–100	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis669_connection_usage	Connection Usage	Connection usage in a collection period. Formula: Used connections/Total connections. Unit: %	0–100	GeminiDB Redis instance nodes	1 minute
redis319_all_qps	Proxy QPS	Proxy QPS on a node Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis318_all_p99	p99 Access Latency	p99 latency when a node executes all commands Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis317_all_max_usec	Maximum Access Latency	Maximum latency when a node executes all commands Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis316_all_avg_usec	Average Access Latency	Average latency when a node executes all commands Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis032_shard_qps	Shard QPS	Shard QPS on a node Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis021_proxy_send_client_bps	Traffic Send Speed	Outgoing traffic speed of a node Unit: byte/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis020_proxy_recv_client_bps	Traffic Receive Speed	Incoming traffic speed of a node Unit: byte/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis019_proxy_response_ps	Proxy Response Rate	Speed at which proxy responds to clients Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis018_proxy_request_ps	Request Receive Speed	Speed at which proxy receives requests from clients Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis017_proxy_accept	Client Connections	Client connections to a node Unit: count	≥ 0	GeminiDB Redis instance nodes	1 minute
nosql007_disk_used_size	Storage Space Usage	Used storage space of an instance Unit: GB	≥ 0	GeminiDB Redis instance nodes	1 minute
nosql006_disk_total_size	Total Storage Space	Total storage space of an instance Unit: GB	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
nosql005_disk_usage	Storage Usage	Storage usage of the current instance. Unit: %	0–100	GeminiDB Redis instance nodes	1 minute
nosql002_mem_usage	Memory Usage	Memory usage of the monitored system Unit: %	0–100	GeminiDB Redis instance nodes	1 minute
nosql001_cpu_usage	CPU Usage	CPU usage of the monitored system Unit: %	0–100	GeminiDB Redis instance nodes	1 minute

Table 4-59 String command metrics

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis107_decr_qps	DECR QPS	QPS when a node executes the DECR command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis106_decr_p99	DECR p99 Latency	p99 latency when a node executes the DECR command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis104_decr_avg_usec	DECR Average Latency	Average latency when a node executes the DECR command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis103_incr_qps	INCR QPS	QPS when a node executes the INCR command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis102_incr_p99	INCR p99 Latency	p99 latency when a node executes the INCR command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis100_incr_avg_usec	INCR Average Latency	Average latency when a node executes the INCR command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis099_strlen_qps	STRLEN QPS	QPS when a node executes the STRLEN command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis098_strlen_p99	STRLEN p99 Latency	p99 latency when a node executes the STRLEN command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis096_strlen_avg_usec	STRLEN Average Latency	Average latency when a node executes the STRLEN command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis095_substr_qps	SUBSTR QPS	QPS when a node executes the SUBSTR command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis094_substr_p99	SUBSTR p99 Latency	p99 latency when a node executes the SUBSTR command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis092_substr_avg_usec	SUBSTR Average Latency	Average latency when a node executes the SUBSTR command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis091_setrange_qps	SETRANGE QPS	QPS when a node executes the SETRANGE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis090_setrange_p99	SETRANGE p99 Latency	p99 latency when a node executes the SETRANGE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis088_setrange_avg_usec	SETRANGE Average Latency	Average latency when a node executes the SETRANGE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis087_getrange_qps	GETRANGE QPS	QPS when a node executes the GETRANGE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis086_getrange_p99	GETRANGE p99 Latency	p99 latency when a node executes the GETRANGE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis084_getrange_avg_usec	GETRANGE Average Latency	Average latency when a node executes the GETRANGE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis083_mset_qps	MSET QPS	QPS when a node executes the MSET command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis082_mset_p99	MSET p99 Latency	p99 latency when a node executes the MSET command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis080_mset_avg_usec	MSET Average Latency	Average latency when a node executes the MSET command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis079_mget_qps	MGET QPS	QPS when a node executes the MGET command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis078_mget_p99	MGET p99 Latency	p99 latency when a node executes the MGET command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis076_mget_avg_usec	MGET Average Latency	Average latency when a node executes the MGET command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis075_append_qps	APPEND QPS	QPS when a node executes the APPEND command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis074_append_p99	APPEND p99 Latency	p99 latency when a node executes the APPEND command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis072_append_avg_usec	APPEND Average Latency	Average latency when a node executes the APPEND command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis071_getset_qps	GETSET QPS	QPS when a node executes the GETSET command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis070_getset_p99	GETSET p99 Latency	p99 latency when a node executes the GETSET command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis068_getset_avg_usec	GETSET Average Latency	Average latency when a node executes the GETSET command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis067_get_qps	GET QPS	QPS when a node executes the GET command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis066_get_p99	GET p99 Latency	p99 latency when a node executes the GET command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis064_get_avg_usec	GET Average Latency	Average latency when a node executes the GET command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis063_set_qps	SET QPS	QPS when a node executes the SET command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis062_set_p99	SET p99 Latency	p99 latency when a node executes the SET command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis060_set_avg_usec	SET Average Latency	Average latency when a node executes the SET command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Table 4-60 Hashes command metrics

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis159_hscan_qps	HSCAN QPS	QPS when a node executes the HSCAN command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis158_hscan_p99	HSCAN p99 Latency	p99 latency when a node executes the HSCAN command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis156_hscan_avg_usec	HSCAN Average Latency	Average latency when a node executes the HSCAN command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis155_hvals_qps	HVALS QPS	QPS when a node executes the HVALS command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis154_hvals_p99	HVALS p99 Latency	p99 latency when a node executes the HVALS command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis152_hvals_avg_usec	HVALS Average Latency	Average latency when a node executes the HVALS command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis151_hstrlen_qps	HSTRLEN QPS	QPS when a node executes the HSTRLEN command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis150_hstrlen_p99	HSTRLEN p99 Latency	p99 latency when a node executes the HSTRLEN command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis148_hstrlen_avg_usec	HSTRLEN Average Latency	Average latency when a node executes the HSTRLEN command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis147_hlen_qps	HLEN QPS	QPS when a node executes the HLEN command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis146_hlen_p99	HLEN p99 Latency	p99 latency when a node executes the HLEN command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis144_hlen_avg_usec	HLEN Average Latency	Average latency when a node executes the HLEN command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis143_hkeys_qps	HKEYS QPS	QPS when a node executes the HKEYS command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis142_hkeys_p99	HKEYS p99 Latency	p99 latency when a node executes the HKEYS command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis140_hkeys_avg_usec	HKEYS Average Latency	Average latency when a node executes the HKEYS command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis139_hincrby_qps	HINCRBY QPS	QPS when a node executes the HINCRBY command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis138_hincrby_p99	HINCRBY p99 Latency	p99 latency when a node executes the HINCRBY command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis136_hincrby_avg_usec	HINCRBY Average Latency	Average latency when a node executes the HINCRBY command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis135_hexists_qps	HEXISTS QPS	QPS when a node executes the HEXISTS command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis134_hexists_p99	HEXISTS p99 Latency	p99 latency when a node executes the HEXISTS command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis132_hexists_avg_usec	HEXISTS Average Latency	Average latency when a node executes the HEXISTS command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis131_hgetall_qps	HGETALL QPS	QPS when a node executes the HGETALL command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis130_hgetall_p99	HGETALL p99 Latency	p99 latency when a node executes the HGETALL command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis128_hgetall_avg_usec	HGETALL Average Latency	Average latency when a node executes the HGETALL command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis127_hdel_qps	HDEL QPS	QPS when a node executes the HDEL command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis126_hdel_p99	HDEL p99 Latency	p99 latency when a node executes the HDEL command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis124_hdel_avg_usec	HDEL Average Latency	Average latency when a node executes the HDEL command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis123_hmget_qps	HMGET QPS	QPS when a node executes the HMGET command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis122_hmget_p99	HMGET p99 Latency	p99 latency when a node executes the HMGET command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis120_hmget_avg_usec	HMGET Average Latency	Average latency when a node executes the HMSET command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis119_hmset_qps	HMSET QPS	QPS when a node executes the HMSET command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis118_hmset_p99	HMSET p99 Latency	p99 latency when a node executes the HMSET command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis116_hmset_avg_usec	HMSET Average Latency	Average latency when a node executes the HMSET command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis115_hget_qps	HGET QPS	QPS when a node executes the HGET command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis114_hget_p99	HGET p99 Latency	p99 latency when a node executes the HGET command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis112_hget_avg_usec	HGET Average Latency	Average latency when a node executes the HGET command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis111_hset_qps	HSET QPS	QPS when a node executes the HSET command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis110_hset_p99	HSET p99 Latency	p99 latency when a node executes the HSET command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis108_hset_avg_usec	HSET Average Latency	Average latency when a node executes the HSET command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Table 4-61 List command metrics

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis207_ltrim_qps	LTRIM QPS	QPS when a node executes the LTRIM command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis206_ltrim_p99	LTRIM p99 Latency	p99 latency when a node executes the LTRIM command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis204_ltrim_avg_usec	LTRIM Average Latency	Average latency when a node executes the LTRIM command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis203_lset_qps	LSET QPS	QPS when a node executes the LSET command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis202_lset_p99	LSET p99 Latency	p99 latency when a node executes the LSET command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis200_lset_avg_usec	LSET Average Latency	Average latency when a node executes the LSET command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis199_lrem_qps	LREM QPS	QPS when a node executes the LREM command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis198_lrem_p99	LREM p99 Latency	p99 latency when a node executes the LREM command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis196_lrem_avg_usec	LREM Average Latency	Average latency when a node executes the LREM command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis195_lrange_qps	LRANGE QPS	QPS when a node executes the LRANGE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis194_lrange_p99	LRANGE p99 Latency	p99 latency when a node executes the LRANGE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis192_lrange_avg_usec	LRANGE Average Latency	Average latency when a node executes the LRANGE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis191_linsert_qps	LINSERT QPS	QPS when a node executes the LINSERT command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis190_linsert_p99	LINSERT p99 Latency	p99 latency when a node executes the LINSERT command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis188_linsert_avg_usec	LINSERT Average Latency	Average latency when a node executes the LINSERT command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis187_lindex_qps	LINDEX QPS	QPS when a node executes the LINDEX command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis186_lindex_p99	LINDEX p99 Latency	p99 latency when a node executes the LINDEX command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis184_lindex_avg_usec	LINDEX Average Latency	Average latency when a node executes the LINDEX command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis183_llen_qps	LLEN QPS	QPS when a node executes the LLEN command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis182_llen_p99	LLEN p99 Latency	p99 latency when a node executes the LLEN command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis180_llen_avg_usec	LLEN Average Latency	Average latency when a node executes the LLEN command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis179_rpoplpush_qps	RPOPLPUSH QPS	QPS when a node executes the RPOPLPUSH command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis178_rpoplpush_p99	RPOPLPUSH p99 Latency	p99 latency when a node executes the RPOPLPUSH command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis176_rpoplpush_avg_usec	RPOPLPUSH Average Latency	Average latency when a node executes the RPOPLPUSH command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis175_rpop_qps	RPOP QPS	QPS when a node executes the RPOP command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis174_rpop_p99	RPOP p99 Latency	p99 latency when a node executes the RPOP command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis172_rpop_avg_usec	RPOP Average Latency	Average latency when a node executes the RPOP command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis171_rpush_qps	RPUSH QPS	QPS when a node executes the RPUSH command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis170_rpush_p99	RPUSH p99 Latency	p99 latency when a node executes the RPUSH command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis168_rpush_avg_usec	RPUSH Average Latency	Average latency when a node executes the RPUSH command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis167_lpop_qps	LPOP QPS	QPS when a node executes the LPOP command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis166_lpop_p99	LPOP p99 Latency	p99 latency when a node executes the LPOP command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis164_lpop_avg_usec	LPOP Average Latency	Average latency when a node executes the LPOP command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis163_lpush_qps	LPUSH QPS	QPS when a node executes the LPUSH command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis162_lpush_p99	LPUSH p99 Latency	p99 latency when a node executes the LPUSH command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis160_lpush_avg_usec	LPUSH Average Latency	Average latency when a node executes the LPUSH command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis470_brpoplpush_qps	BRPOPLPUSH QPS	QPS when a node executes the BRPOPLPUSH command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis468_brpoplpush_p99	BRPOPLPUSH p99 Latency	p99 latency when a node executes the BRPOPLPUSH command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis466_brpoplpush_avg_usec	BRPOPLPUSH Average Latency	Average latency when a node executes the BRPOPLPUSH command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis465_brpop_qps	BRPOP QPS	QPS when a node executes the BRPOP command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis463_brpop_p99	BRPOP p99 Latency	p99 latency when a node executes the BRPOP command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis461_brpop_avg_usec	BRPOP Average Latency	Average latency when a node executes the BRPOP command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis460_blpop_qps	BLPOP QPS	QPS when a node executes the BLPOP command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis458_blpop_p99	BLPOP p99 Latency	p99 latency when a node executes the BLPOP command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis456_blpop_avg_usec	BLPOP Average Latency	Average latency when a node executes the BLPOP command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Table 4-62 Set command metrics

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis247_srandmember_qps	SRANDMEMBER QPS	QPS when a node executes the SRANDMEMBER command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis246_srandmember_p99	SRANDMEMBER p99 Latency	p99 latency when a node executes the SRANDMEMBER command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis244_srandmember_avg_usec	SRANDMEMBER Average Latency	Average latency when a node executes the SRANDMEMBER command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis243_sdiff_qps	SDIFF QPS	QPS when a node executes the SDIFF command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis242_sdiff_p99	SDIFF p99 Latency	p99 latency when a node executes the SDIFF command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis240_sdiff_avg_usec	SDIFF Average Latency	Average latency when a node executes the SDIFF command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis239_sismember_qps	SISMEMBER QPS	QPS when a node executes the SISMEMBER command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis238_sismember_p99	SISMEMBER p99 Latency	p99 latency when a node executes the SISMEMBER command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis236_sismember_avg_usec	SISMEMBER Average Latency	Average latency when a node executes the SISMEMBER command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis235_sinter_qps	SINTER QPS	QPS when a node executes the SINTER command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis234_sinter_p99	SINTER p99 Latency	p99 latency when a node executes the SINTER command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis232_sinter_avg_usec	SINTER Average Latency	Average latency when a node executes the SINTER command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis231_sunion_qps	SUNION QPS	QPS when a node executes the SUNION command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis230_sunion_p99	SUNION p99 Latency	p99 latency when a node executes the SUNION command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis228_sunion_avg_usec	SUNION Average Latency	Average latency when a node executes the SUNION command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis227_srem_qps	SREM QPS	QPS when a node executes the SREM command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis226_srem_p99	SREM p99 Latency	p99 latency when a node executes the SREM command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis224_srem_avg_usec	SREM Average Latency	Average latency when a node executes the SREM command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis223_smembers_qps	SMEMBERS QPS	QPS when a node executes the SMEMBERS command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis222_smembers_p99	SMEMBERS p99 Latency	p99 latency when a node executes the SMEMBERS command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis220_smembers_avg_usec	SMEMBERS Average Latency	Average latency when a node executes the SMEMBERS command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis219_scard_qps	SCARD QPS	QPS when a node executes the SCARD command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis218_scard_p99	SCARD p99 Latency	p99 latency when a node executes the SCARD command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis216_scard_avg_usec	SCARD Average Latency	Average latency when a node executes the SCARD command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis215_spop_qps	SPOP QPS	QPS when a node executes the SPOP command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis214_spop_p99	SPOP p99 Latency	p99 latency when a node executes the SPOP command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis212_spop_avg_usec	SPOP Average Latency	Average latency when a node executes the SPOP command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis211_sadd_qps	SADD QPS	QPS when a node executes the SADD command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis210_sadd_p99	SADD p99 Latency	p99 latency when a node executes the SADD command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis208_sadd_avg_usec	SADD Average Latency	Average latency when a node executes the SADD command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis485_sdiffstore_qps	SDIFFSTORE QPS	QPS when a node executes the SDIFFSTORE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis483_sdiffstore_p99	SDIFFSTORE p99 Latency	p99 latency when a node executes the SDIFFSTORE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis481_sdiffstore_avg_usec	SDIFFSTORE Average Latency	Average latency when a node executes the SDIFFSTORE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis480_sinterstore_qps	SINTERSTORE QPS	QPS when a node executes the SINTERSTORE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis478_sinterstore_p99	SINTERSTORE p99 Latency	p99 latency when a node executes the SINTERSTORE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis476_sinterstore_avg_usec	SINTERSTORE Average Latency	Average latency when a node executes the SINTERSTORE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis475_sunionstore_qps	SUNIONSTORE QPS	QPS when a node executes the SUNIONSTORE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis473_sunionstore_p99	SUNIONSTORE p99 Latency	p99 latency when a node executes the SUNIONSTORE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis471_sunionstore_avg_usec	SUNIONSTORE Average Latency	Average latency when a node executes the SUNIONSTORE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Table 4-63 Zset command metrics

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis315_zremrangebylex_qps	ZREMRANGEBYLEX QPS	QPS when a node executes the ZREMRANGEBYLEX command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis314_zremrangebylex_p99	ZREMRANGEBYLEX p99 Latency	p99 latency when a node executes the ZREMRANGEBYLEX command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis312_zremrangebylex_avg_usec	ZREMRANGEBYLEX Average Latency	Average latency when a node executes the ZREMRANGEBYLEX command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis311_zremrangebyscore_qps	ZREMRANGEBYSCORE QPS	QPS when a node executes the ZREMRANGEBYSCORE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis310_zremrangebyscore_p99	ZREMRANGEBYSCORE p99 Latency	p99 latency when a node executes the ZREMRANGEBYSCORE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis308_zremrangebyscore_avg_usec	ZREMRANGEBYSCORE Average Latency	Average latency when a node executes the ZREMRANGEBYSCORE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis307_zremrangebyrank_qps	ZREMRANGEBYRANK QPS	QPS when a node executes the ZREMRANGEBYRANK command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis306_zremrangebyrank_p99	ZREMRANGEBYRANK p99 Latency	p99 latency when a node executes the ZREMRANGEBYRANK command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis304_zremrangebyrank_avg_usec	ZREMRANGEBYRANK Average Latency	Average latency when a node executes the ZREMRANGEBYRANK command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis303_zpopmin_qps	ZPOPMIN QPS	QPS when a node executes the ZPOPMIN command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis302_zpopmin_p99	ZPOPMIN p99 Latency	p99 latency when a node executes the ZPOPMIN command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis300_zpopmin_avg_usec	ZPOPMIN Average Latency	Average latency when a node executes the ZPOPMIN command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis299_zpopmax_qps	ZPOPMAX QPS	QPS when a node executes the ZPOPMAX command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis298_zpopmax_p99	ZPOPMAX p99 Latency	p99 latency when a node executes the ZPOPMAX command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis296_zpopmax_avg_usec	ZPOPMAX Average Latency	Average latency when a node executes the ZPOPMAX command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis295_zlexcount_qps	ZLEXCOUNT QPS	QPS when a node executes the ZLEXCOUNT command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis294_zlexcount_p99	ZLEXCOUNT p99 Latency	p99 latency when a node executes the ZLEXCOUNT command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis292_zlexcount_avg_usec	ZLEXCOUNT Average Latency	Average latency when a node executes the ZLEXCOUNT command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis291_zrevrank_qps	ZREVRANK QPS	QPS when a node executes the ZREVRANK command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis290_zrevrank_p99	ZREVRANK p99 Latency	p99 latency when a node executes the ZREVRANK command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis288_zrevrank_avg_usec	ZREVRANK Average Latency	Average latency when a node executes the ZREVRANK command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis287_zrank_qps	ZRANK QPS	QPS when a node executes the ZRANK command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis286_zrank_p99	ZRANK p99 Latency	p99 latency when a node executes the ZRANK command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis284_zrank_avg_usec	ZRANK Average Latency	Average latency when a node executes the ZRANK command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis283_zscore_qps	ZSCORE QPS	QPS when a node executes the ZSCORE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis282_zscore_p99	ZSCORE p99 Latency	p99 latency when a node executes the ZSCORE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis280_zscore_avg_usec	ZSCORE Average Latency	Average latency when a node executes the ZSCORE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis279_zrem_qps	ZREM QPS	QPS when a node executes the ZREM command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis278_zrem_p99	ZREM p99 Latency	p99 latency when a node executes the ZREM command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis276_zrem_avg_usec	ZREM Average Latency	Average latency when a node executes the ZREM command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis275_zcount_qps	ZCOUNT QPS	QPS when a node executes the ZCOUNT command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis274_zcount_p99	ZCOUNT p99 Latency	p99 latency when a node executes the ZCOUNT command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis272_zcount_avg_usec	ZCOUNT Average Latency	Average latency when a node executes the ZCOUNT command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis271_zrange_qps	ZRANGE QPS	QPS when a node executes the ZRANGE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis270_zrange_p99	ZRANGE p99 Latency	p99 latency when a node executes the ZRANGE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis268_zrange_avg_usec	ZRANGE Average Latency	Average latency when a node executes the ZRANGE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis267_zrevrange_qps	ZREVRANGE QPS	QPS when a node executes the ZREVRANGE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis266_zrevrange_p99	ZREVRANGE p99 Latency	p99 latency when a node executes the ZREVRANGE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis264_zrevrange_avg_usec	ZREVRANGE Average Latency	Average latency when a node executes the ZREVRANGE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis263_zincrby_qps	ZINCRBY QPS	QPS when a node executes the ZINCRBY command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis262_zincrby_p99	ZINCRBY p99 Latency	p99 latency when a node executes the ZINCRBY command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis260_zincrby_avg_usec	ZINCRBY Average Latency	Average latency when a node executes the ZINCRBY command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis259_zscan_qps	ZSCAN QPS	QPS when a node executes the ZSCAN command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis258_zscan_p99	ZSCAN p99 Latency	p99 latency when a node executes the ZSCAN command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis256_zscan_avg_usec	ZSCAN Average Latency	Average latency when a node executes the ZSCAN command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis255_zcard_qps	ZCARD QPS	QPS when a node executes the ZCARD command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis254_zcard_p99	ZCARD p99 Latency	p99 latency when a node executes the ZCARD command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis252_zcard_avg_usec	ZCARD Average Latency	Average latency when a node executes the ZCARD command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis251_zadd_qps	ZADD QPS	QPS when a node executes the ZADD command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis250_zadd_p99	ZADD p99 Latency	p99 latency when a node executes the ZADD command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis248_zadd_avg_usec	ZADD Average Latency	Average latency when a node executes the ZADD command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis495_zinterstore_qps	ZINTERSTORE QPS	QPS when a node executes the ZINTERSTORE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis493_zinterstore_p99	ZINTERSTORE p99 Latency	p99 latency when a node executes the ZINTERSTORE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis491_zinterstore_avg_usec	ZINTERSTORE Average Latency	Average latency when a node executes the ZINTERSTORE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis490_zunionstore_qps	ZUNIONSTORE QPS	QPS when a node executes the ZUNIONSTORE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis488_zunionstore_p99	ZUNIONSTORE p99 Latency	p99 latency when a node executes the ZUNIONSTORE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis486_zunionstore_avg_usec	ZUNIONSTORE Average Latency	Average latency when a node executes the ZUNIONSTORE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Table 4-64 Bitmap command metrics

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis440_bitfield_qps	BITFIELD QPS	QPS when a node executes the BITFIELD command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis438_bitfield_p99	BITFIELD p99 Latency	p99 latency when a node executes the BITFIELD command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis436_bitfield_avg_usec	BITFIELD Average Latency	Average latency when a node executes the BITFIELD command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis435_bitop_qps	BITOP QPS	QPS when a node executes the BITOP command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis433_bitop_p99	BITOP p99 Latency	p99 latency when a node executes the BITOP command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis431_bitop_avg_usec	BITOP Average Latency	Average latency when a node executes the BITOP command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis430_bitpos_qps	BITPOS QPS	QPS when a node executes the BITPOS command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis428_bitpos_p99	BITPOS p99 Latency	p99 latency when a node executes the BITPOS command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis426_bitpos_avg_usec	BITPOS Average Latency	Average latency when a node executes the BITPOS command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis425_bitcount_qps	BITCOUNT QPS	QPS when a node executes the BITCOUNT command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis423_bitcount_p99	BITCOUNT p99 Latency	p99 latency when a node executes the BITCOUNT command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis421_bitcount_avg_usec	BITCOUNT Average Latency	Average latency when a node executes the BITCOUNT command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis420_getbit_qps	GETBIT QPS	QPS when a node executes the GETBIT command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis418_getbit_p99	GETBIT p99 Latency	p99 latency when a node executes the GETBIT command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis416_getbit_avg_usec	GETBIT Average Latency	Average latency when a node executes the GETBIT command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis415_setbit_qps	SETBIT QPS	QPS when a node executes the SETBIT command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis413_setbit_p99	SETBIT p99 Latency	p99 latency when a node executes the SETBIT command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis411_setbit_avg_usec	SETBIT Average Latency	Average latency when a node executes the SETBIT command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Table 4-65 Stream command metrics

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis590_xreadgroup_qps	XREADGROUP QPS	QPS when a node executes the XREADGROUP command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis588_xreadgroup_p99	XREADGROUP p99 Latency	p99 latency when a node executes the XREADGROUP command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis586_xreadgroup_avg_usec	XREADGROUP Average Latency	Average latency when a node executes the XREADGROUP command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis585_xread_qps	XREAD QPS	QPS when a node executes the XREAD command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis583_xread_p99	XREAD p99 Latency	p99 latency when a node executes the XREAD command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis581_xread_avg_usec	XREAD Average Latency	Average latency when a node executes the XREAD command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis580_xinfo_qps	XINFO QPS	QPS when a node executes the XINFO command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis578_xinfo_p99	XINFO p99 Latency	p99 latency when a node executes the XINFO command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis576_xinfo_avg_usec	XINFO Average Latency	Average latency when a node executes the XINFO command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis575_xpending_qps	XPENDING QPS	QPS when a node executes the XPENDING command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis573_xpending_p99	XPENDING p99 Latency	p99 latency when a node executes the XPENDING command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis571_xpending_avg_usec	XPENDING Average Latency	Average latency when a node executes the XPENDING command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis570_xclaim_qps	XCLAIM QPS	QPS when a node executes the XCLAIM command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis568_xclaim_p99	XCLAIM p99 Latency	p99 latency when a node executes the XCLAIM command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis566_xclaim_avg_usec	XCLAIM Average Latency	Average latency when a node executes the XCLAIM command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis565_xrevrange_qps	XREVRANGE QPS	QPS when a node executes the XREVRANGE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis563_xrevrange_p99	XREVRANGE p99 Latency	p99 latency when a node executes the XREVRANGE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis561_xrevrange_avg_usec	XREVRANGE Average Latency	Average latency when a node executes the XREVRANGE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis560_xrange_qps	XRANGE QPS	QPS when a node executes the XRANGE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis558_xrange_p99	XRANGE p99 Latency	p99 latency when a node executes the XRANGE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis556_xrange_avg_usec	XRANGE Average Latency	Average latency when a node executes the XRANGE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis555_xlen_qps	XLEN QPS	QPS when a node executes the XLEN command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis553_xlen_p99	XLEN p99 Latency	p99 latency when a node executes the XLEN command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis551_xlen_avg_usec	XLEN Average Latency	Average latency when a node executes the XLEN command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis550_xtrim_qps	XTRIM QPS	QPS when a node executes the XTRIM command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis548_xtrim_p99	XTRIM p99 Latency	p99 latency when a node executes the XTRIM command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis546_xtrim_avg_usec	XTRIM Average Latency	Average latency when a node executes the XTRIM command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis545_xdel_qps	XDEL QPS	QPS when a node executes the XDEL command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis543_xdel_p99	XDEL p99 Latency	p99 latency when a node executes the XDEL command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis541_xdel_avg_usec	XDEL Average Latency	Average latency when a node executes the XDEL command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis540_xgroup_qps	XGROUP QPS	QPS when a node executes the XGROUP command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis538_xgroup_p99	XGROUP p99 Latency	p99 latency when a node executes the XGROUP command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis536_xgroup_avg_usec	XGROUP Average Latency	Average latency when a node executes the XGROUP command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis535_xack_qps	XACK QPS	QPS when a node executes the XACK command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis533_xack_p99	XACK p99 Latency	p99 latency when a node executes the XACK command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis531_xack_avg_usec	XACK Average Latency	Average latency when a node executes the XACK command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis530_xadd_qps	XADD QPS	QPS when a node executes the XADD command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis528_xadd_p99	XADD p99 Latency	p99 latency when a node executes the XADD command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis526_xadd_avg_usec	XADD Average Latency	Average latency when a node executes the XADD command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Table 4-66 Geo command metrics

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis525_geopos_qps	GEOPOS QPS	QPS when a node executes the GEOPOS command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis523_geopos_p99	GEOPOS p99 Latency	p99 latency when a node executes the GEOPOS command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis521_geopos_avg_usec	GEOPOS Average Latency	Average latency when a node executes the GEOPOS command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis520_geodist_qps	GEODIST QPS	QPS when a node executes the GEODIST command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis518_geodist_p99	GEODIST p99 Latency	p99 latency when a node executes the GEODIST command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis516_geodist_avg_usec	GEODIST Average Latency	Average latency when a node executes the GEODIST command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis515_geohash_qps	GEOHASH QPS	QPS when a node executes the GEOHASH command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis513_geohash_p99	GEOHASH p99 Latency	p99 latency when a node executes the GEOHASH command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis511_geohash_avg_usec	GEOHASH Average Latency	Average latency when a node executes the GEOHASH command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis510_georadius_qps	GEORADIUS QPS	QPS when a node executes the GEORADIUS command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis508_georadius_p99	GEORADIUS p99 Latency	p99 latency when a node executes the GEORADIUS command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis506_georadius_avg_usec	GEORADIUS Average Latency	Average latency when a node executes the GEORADIUS command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis505_georadiusbymember_qps	GEORADIUSBYMEMBER QPS	QPS when a node executes the GEORADIUSBYMEMBER command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis503_georadiusbymember_p99	GEORADIUSBYMEMBER p99 Latency	p99 latency when a node executes the GEORADIUSBYMEMBER command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis501_georadiusbymember_avg_usec	GEORADIUSBYMEMBER Average Latency	Average latency when a node executes the GEORADIUSBYMEMBER command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis500_geoadd_qps	GEOADD QPS	QPS when a node executes the GEOADD command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis498_geoadd_p99	GEOADD p99 Latency	p99 latency when a node executes the GEOADD command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis496_geoadd_avg_usec	GEOADD Average Latency	Average latency when a node executes the GEOADD command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Table 4-67 Hyperloglog command metrics

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis455_pfmerge_qps	PFMERGE QPS	QPS when a node executes the PFMERGE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis453_pfmerge_p99	PFMERGE p99 Latency	p99 latency when a node executes the PFMERGE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis451_pfmerge_avg_usec	PFMERGE Average Latency	Average latency when a node executes the PFMERGE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis450_pfcount_qps	PFCOUNT QPS	QPS when a node executes the PFCOUNT command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis448_pfcount_p99	PFCOUNT p99 Latency	p99 latency when a node executes the PFCOUNT command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis446_pfcount_avg_usec	PFCOUNT Average Latency	Average latency when a node executes the PFCOUNT command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis445_pfadd_qps	PFADD QPS	QPS when a node executes the PFADD command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis443_pfadd_p99	PFADD p99 Latency	p99 latency when a node executes the PFADD command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis441_pfadd_avg_usec	PFADD Average Latency	Average latency when a node executes the PFADD command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Table 4-68 Pub/Sub command metrics

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis620_pubsub_qps	PUBSUB QPS	QPS when a node executes the PUBSUB command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis618_pubsub_p99	PUBSUB p99 Latency	p99 latency when a node executes the PUBSUB command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis616_pubsub_avg_usec	PUBSUB Average Latency	Average latency when a node executes the PUBSUB command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis615_punsubscribe_qps	PUNSUBSCRIBE QPS	QPS when a node executes the PUNSUBSCRIBE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis613_punsubscribe_p99	PUNSUBSCRIBE p99 Latency	p99 latency when a node executes the PUNSUBSCRIBE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis611_punsubscribe_avg_usec	PUNSUBSCRIBE Average Latency	Average latency when a node executes the PUNSUBSCRIBE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis610_psubscribe_qps	PSUBSCRIBE QPS	QPS when a node executes the PSUBSCRIBE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis608_psubscribe_p99	PSUBSCRIBE p99 Latency	p99 latency when a node executes the PSUBSCRIBE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis606_psubscribe_avg_usec	PSUBSCRIBE Average Latency	Average latency when a node executes the PSUBSCRIBE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis605_unsubscribe_qps	UNSUBSCRIBE QPS	QPS when a node executes the UNSUBSCRIBE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis603_unsubscribe_p99	UNSUBSCRIBE p99 Latency	p99 latency when a node executes the UNSUBSCRIBE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis601_unsubscribe_avg_usec	UNSUBSCRIBE Average Latency	Average latency when a node executes the UNSUBSCRIBE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis600_subscribe_qps	SUBSCRIBE QPS	QPS when a node executes the SUBSCRIBE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis598_subscribe_p99	SUBSCRIBE p99 Latency	p99 latency when a node executes the SUBSCRIBE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis596_subscribe_avg_usec	SUBSCRIBE Average Latency	Average latency when a node executes the SUBSCRIBE command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis595_publish_qps	PUBLISH QPS	QPS when a node executes the PUBLISH command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis593_publish_p99	PUBLISH p99 Latency	p99 latency when a node executes the PUBLISH command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis591_publish_avg_usec	PUBLISH Average Latency	Average latency when a node executes the PUBLISH command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Table 4-69 Scripting command metrics

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis635_script_qps	SCRIPT QPS	QPS when a node executes the SCRIPT command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis633_script_p99	SCRIPT p99 Latency	p99 latency when a node executes the SCRIPT command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis631_script_avg_usec	SCRIPT Average Latency	Average latency when a node executes the SCRIPT command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis630_evalsha_qps	EVALSHA QPS	QPS when a node executes the EVALSHA command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis628_evalsha_p99	EVALSHA p99 Latency	p99 latency when a node executes the EVALSHA command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis626_evalsha_avg_usec	EVALSHA Average Latency	Average latency when a node executes the EVALSHA command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis625_eval_qps	EVAL QPS	QPS when a node executes the EVAL command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis623_eval_p99	EVAL p99 Latency	p99 latency when a node executes the EVAL command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis621_eval_avg_usec	EVAL Average Latency	Average latency when a node executes the EVAL command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Table 4-70 Transactions command metrics

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis660_discard_qps	DISCARD QPS	QPS when a node executes the DISCARD command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis658_discard_p99	DISCARD p99 Latency	p99 latency when a node executes the DISCARD command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis656_discard_avg_usec	DISCARD Average Latency	Average latency when a node executes the DISCARD command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis655_exec_qps	EXEC QPS	QPS when a node executes the EXEC command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis653_exec_p99	EXEC p99 Latency	p99 latency when a node executes the EXEC command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis651_exec_avg_usec	EXEC Average Latency	Average latency when a node executes the EXEC command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis650_multi_qps	MULTI QPS	QPS when a node executes the MULTI command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis648_multi_p99	MULTI p99 Latency	p99 latency when a node executes the MULTI command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis646_multi_avg_usec	MULTI Average Latency	Average latency when a node executes the MULTI command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis645_unwatch_qps	UNWATCH QPS	QPS when a node executes the UNWATCH command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis643_unwatch_p99	UNWATCH p99 Latency	p99 latency when a node executes the UNWATCH command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis641_unwatch_avg_usec	UNWATCH Average Latency	Average latency when a node executes the UNWATCH command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis640_watch_qps	WATCH QPS	QPS when a node executes the WATCH command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis638_watch_p99	WATCH p99 Latency	p99 latency when a node executes the WATCH command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis636_watch_avg_usec	WATCH Average Latency	Average latency when a node executes the WATCH command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Table 4-71 Common command metrics

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis059_scan_qps	SCAN QPS	QPS when a node executes the SCAN command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis058_scan_p99	SCAN p99 Latency	p99 latency when a node executes the SCAN command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis056_scan_avg_usec	SCAN Average Latency	Average latency when a node executes the SCAN command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis055_persist_qps	PERSIST QPS	QPS when a node executes the PERSIST command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis054_persist_p99	PERSIST p99 Latency	p99 latency when a node executes the PERSIST command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis052_persist_avg_usec	PERSIST Average Latency	Average latency when a node executes the PERSIST command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis051_ttl_qps	TTL QPS	QPS when a node executes the TTL command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis050_ttl_p99	TTL p99 Latency	p99 latency when a node executes the TTL command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis048_ttl_avg_usec	TTL Average Latency	Average latency when a node executes the TTL command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis047_del_qps	DEL QPS	QPS when a node executes the DEL command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis046_del_p99	DEL p99 Latency	p99 latency when a node executes the DEL command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis044_del_avg_usec	DEL Average Latency	Average latency when a node executes the DEL command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis043_expire_qps	EXPIRE QPS	QPS when a node executes the EXPIRE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis042_expire_p99	EXPIRE p99 Latency	p99 latency when a node executes the EXPIRE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis040_expire_avg_usec	EXPIRE Average Latency	Average latency when a node executes the EXPIRE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis039_exists_qps	EXISTS QPS	QPS when a node executes the EXISTS command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis038_exists_p99	EXISTS p99 Latency	p99 latency when a node executes the EXISTS command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis036_exists_avg_usec	EXISTS Average Latency	Average latency when a node executes the EXISTS command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis410_randomkey_qps	RANDOMKEY QPS	QPS when a node executes the RANDOMKEY command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis408_randomkey_p99	RANDOMKEY p99 Latency	p99 latency when a node executes the RANDOMKEY command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis406_randomkey_avg_usec	RANDOMKEY Average Latency	Average latency when a node executes the RANDOMKEY command Unit: μs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis405_type_qps	TYPE QPS	QPS when a node executes the TYPE command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis403_type_p99	TYPE p99 Latency	p99 latency when a node executes the TYPE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis401_type_avg_usec	TYPE Average Latency	Average latency when a node executes the TYPE command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis400_info_qps	INFO QPS	QPS when a node executes the INFO command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis398_info_p99	INFO p99 Latency	p99 latency when a node executes the INFO command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis396_info_avg_usec	INFO Average Latency	Average latency when a node executes the INFO command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
redis395_ping_qps	PING QPS	QPS when a node executes the PING command Unit: count/s	≥ 0	GeminiDB Redis instance nodes	1 minute
redis393_ping_p99	PING p99 Latency	p99 latency when a node executes the PING command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute
redis391_ping_avg_usec	PING Average Latency	Average latency when a node executes the PING command Unit: µs	≥ 0	GeminiDB Redis instance nodes	1 minute

Dimensions

Key	Value
redis_cluster_id	Cluster ID of the GeminiDB Redis instance
redis_node_id	Node ID of the GeminiDB Redis instance

4.13.2 Configuring Alarm Rules

Scenarios

Setting alarm rules allows you to customize objects to be monitored and notification policies so that you can closely monitor your instances.

Alarm rules include the alarm rule name, instance, metric, threshold, monitoring interval, and whether to send notifications. This section describes how to set alarm rules.

Procedure

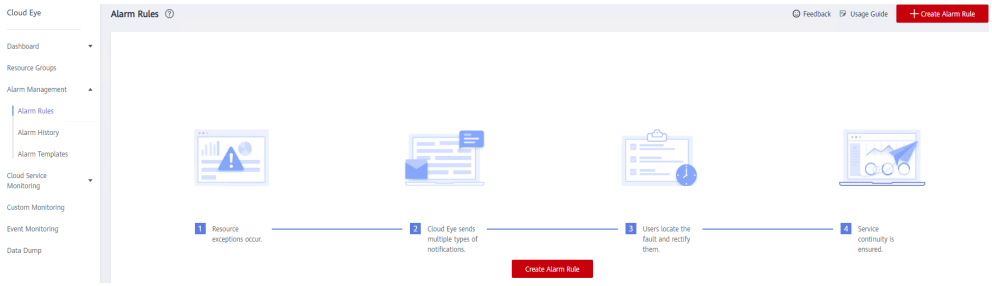
Step 1 Log in to the management console.

Step 2 Click **Service List**. Under **Management & Governance**, click **Cloud Eye**.

Step 3 In the navigation pane on the left, choose **Alarm Management** > **Alarm Rules**.

Step 4 On the **Alarm Rules** page, click **Create Alarm Rule**.

Figure 4-185 Creating an alarm rule



Step 5 Set alarm parameters.

1. Configure basic alarm information.

Figure 4-186 Configuring basic information for an alarm rule

Table 4-72 Basic alarm rule information

Parameter	Description	Example Value
Name	Name of the rule. The system generates a random name and you can modify it.	alarm-cag2
Description	(Optional) Alarm rule description.	-

2. Select objects to be monitored and specify the monitoring scope.

Table 4-73 Parameter description

Parameter	Description	Example Value
Alarm Type	Alarm type that the alarm rule is created for. The value can be **Metric** or **Event**.	Metric
Resource Type	Type of the resource the alarm rule is created for. Select **GeminiDB**.	-
Dimension	Metric dimension of the alarm rule. Select **Redis-Redis Nodes**.	-
Monitoring Scope	Monitoring scope the alarm rule applies to. **NOTE** – If you select **All resources**, an alarm notification will be sent when any instance meets an alarm policy, and existing alarm rules will be automatically applied for newly purchased resources. – If you select **Resource groups** and any resource in the group meets the alarm policy, an alarm notification will be sent. – To specify **Specific resources**, click **Select Specified Resources**, select one or more resources, and click **OK**.	Specified resources
Group	This parameter is mandatory when **Monitoring Scope** is set to **Resource groups**.	-

3. Configure an alarm policy.

Figure 4-187 Configuring the alarm policy

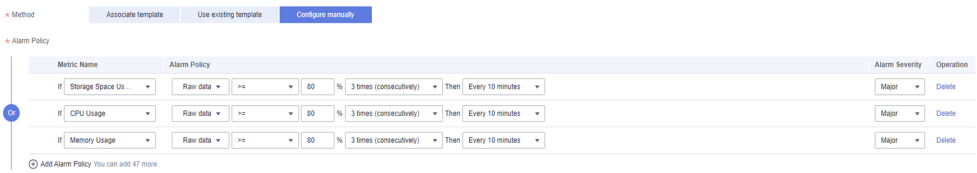


Table 4-74 Parameter description

Parameter	Description	Example Value
Method	Select **Associate template**, **Use existing template**, or **Configure manually**. **NOTE** If you set **Monitoring Scope** to **Specific resources**, you can set **Method** to **Use existing template**.	Configure manually

Parameter	Description	Example Value
Template	Select the template to be used. This parameter is available only when you select **Use existing template** for **Method**.	-
Alarm Policy	Policy for triggering an alarm. You can configure the threshold, consecutive periods, alarm interval, and alarm severity based on service requirements. – **Metric Name**: specifies the name of the metric configured in the alarm rule. The following metrics are recommended: **Storage Space Usage**, which is used to monitor the storage usage of GeminiDB Redis instances. If the storage usage is greater than 80%, scale up the storage in a timely manner by referring to **4.6.7.2 Manually Scaling Up Disk Space**. **CPU Usage** and **Memory Usage**, which are used to monitor the compute resource usage of each GeminiDB Redis instance node. If the CPU usage or memory usage is greater than 80%, you can **add nodes** or **upgrade node specifications** in a timely manner. For more metrics, see **4.13.1 GeminiDB Redis Instance Metrics**. – **Alarm Severity**: specifies the severity of the alarm. Valid values are **Critical**, **Major**, **Minor**, and **Informational**. **NOTE** A maximum of 50 alarm policies can be added to an alarm rule. If any one of these alarm policies is met, an alarm is triggered.	Take the CPU usage as an example. The alarm policy configured in **Figure 4-187** indicates that a major alarm notification will be sent to users every 10 minutes if the original CPU usage reaches 80% or above for three consecutive periods.

4. Configure alarm notification information.

Figure 4-188 Configuring alarm notification information

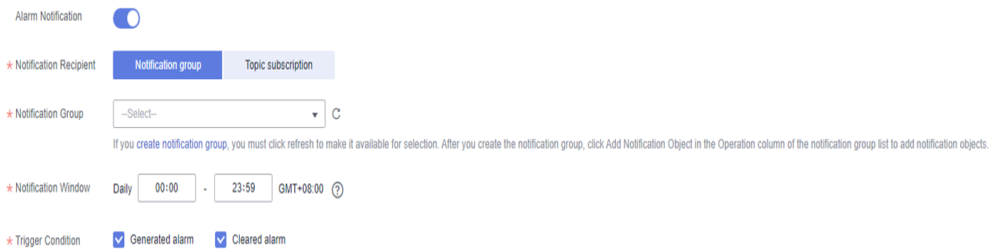


Table 4-75 Parameter description

Parameter	Description	Example Value
Alarm Notification	Whether to notify users when alarms are triggered. Notifications can be sent by email, text message, or HTTP/HTTPS message. Enabling alarm notification is recommended. When the metric data reaches the threshold set in the alarm rule, Cloud Eye immediately notifies you through SMN that an exception has occurred.	Enabled **Alarm Notification**.
Notification Recipient	Select **Notification group** or **Topic subscription**.	-
Notification Group	Notification group the alarm notification is to be sent to.	-
Notification Object	Specifies the object that receives alarm notifications. You can select the account contact or a topic. – Account contact is the mobile phone number and email address provided for registration. – **Topic** is used to publish messages and subscribe to notifications. If the required topic is unavailable, create one first and add subscriptions to it. For details, see **Creating a Topic** and **Adding Subscriptions**.	-
Notification Window	Cloud Eye sends notifications only within the notification window specified in the alarm rule. For example, if **Notification Window** is set to **00:00-8:00**, Cloud Eye sends notifications only within 00:00-08:00.	-

Parameter	Description	Example Value
Trigger Condition	Condition for triggering an alarm notification. You can select **Generated alarm** (when an alarm is generated), **Cleared alarm** (when an alarm is cleared), or both.	-

5. Configure advanced settings.

Figure 4-189 Advanced settings

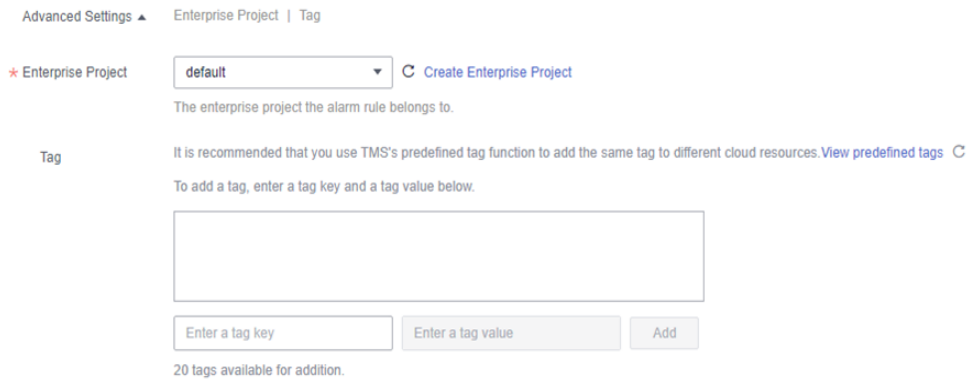


Table 4-76 Parameter description

Parameter	Description	Example Value
Enterprise Project	Enterprise project that the alarm rule belongs to. Only users with the enterprise project permissions can view and manage the alarm rule. For details about how to create an enterprise project, see **Creating an Enterprise Project**.	default
Tag	A tag is a key-value pair. Tags identify cloud resources so that you can easily categorize and search for your resources. You are advised to create predefined tags on TMS. For details about how to create predefined tags, see **Creating Predefined Tags**. – A key can contain a maximum of 128 characters, and a value can contain a maximum of 225 characters. – A maximum of 20 tags can be added.	-

Step 6 After the configuration is complete, click **Create**.

When the metric data reaches the threshold set in the alarm rule, Cloud Eye immediately notifies you through SMN that an exception has occurred.

----End

4.13.3 Recommended Alarm Policies

This section describes recommended alarm policies of GeminiDB Redis instances and nodes.

Table 4-77 Alarm policies for GeminiDB Redis instances

Metric	Dimension	Value Range	Alarm Policy
Total Traffic Sent by the Instance	Instance	≥ 0	Alarm severity: major Number of consecutive periods: 3 Alarm threshold: ≥ 875,000,000 bytes/s Alarm period: every 5 minutes
Total Traffic Received by the Instance	Instance	≥ 0	Alarm severity: major Number of consecutive periods: 3 Alarm threshold: ≥ 875,000,000 bytes/s Alarm period: every 5 minutes
Average Latency	Instance	≥ 0	Alarm severity: major Number of consecutive periods: 5 Alarm threshold: ≥ 15,000 us Alarm period: every 15 minutes

Metric	Dimension	Value Range	Alarm Policy
P99 Latency	Instance	≥ 0	Alarm severity: major Number of consecutive periods: 5 Alarm threshold: ≥ 30,000 us Alarm period: every 15 minutes
Storage Usage	Instance	0-100%	Alarm severity: major Number of consecutive periods: 3 Alarm threshold: ≥ 70% Alarm period: once a day
Max. Parameters Sent in a Request	Instance	≥ 0	Alarm severity: major Number of consecutive periods: 1 Alarm threshold: ≥ 10,000 Alarm period: every 15 minutes
Max. Elements Obtained in a Request	Instance	≥ 0	Alarm severity: major Number of consecutive periods: 1 Alarm threshold: ≥ 10,000 Alarm period: every 15 minutes

Metric	Dimension	Value Range	Alarm Policy
Max. Bytes Sent in a Request	Instance	≥ 0	Alarm severity: major Number of consecutive periods: 1 Alarm threshold: ≥ 1MiB Alarm period: every 15 minutes
Max. Bytes Obtained in a Request	Instance	≥ 0	Alarm severity: major Number of consecutive periods: 1 Alarm threshold: ≥ 1MiB Alarm period: every 15 minutes

Table 4-78 Alarm policies for GeminiDB Redis nodes

Metric	Dimension	Value Range	Alarm Policy
CPU Usage	Node	0-100%	Alarm severity: major Number of consecutive periods: 3 Alarm threshold: ≥ 70% Alarm period: every 15 minutes
Memory Usage	Node	0-100%	Alarm severity: warning Number of consecutive periods: 3 Alarm threshold: ≥ 70% Alarm period: every 15 minutes

Metric	Dimension	Value Range	Alarm Policy
Connection Usage	Node	0-100%	Alarm severity: major Number of consecutive periods: 3 Alarm threshold: ≥ 50% Alarm period: every 5 minutes
Traffic Receive Speed	Node	≥ 0	Alarm severity: major Number of consecutive periods: 3 Alarm threshold: ≥ 87,500,000 bytes/s Alarm period: every 5 minutes
Traffic Send Speed	Node	≥ 0	Alarm severity: major Number of consecutive periods: 3 Alarm threshold: ≥ 87,500,000 bytes/s Alarm period: every 5 minutes

4.13.4 Viewing GeminiDB Redis Instance Metrics

Scenarios

Cloud Eye monitors GeminiDB Redis instance running statuses. You can view the GeminiDB Redis monitoring metrics on the management console.

Monitored data requires a period of time for transmission and display. The status of the monitored object displayed on the Cloud Eye page is the status obtained 5 to 10 minutes before. You can view the monitored data of a newly created DB instance 5 to 10 minutes later.

Prerequisites

- The DB instance is running properly.

  Cloud Eye does not display the metrics of a faulty or deleted DB instance. You can view the monitoring information only after the instance is restarted or recovered.
- The DB instance has been properly running for at least 10 minutes.

  The monitoring data and graphics are available for a new DB instance after the instance runs for at least 10 minutes.

Method 1

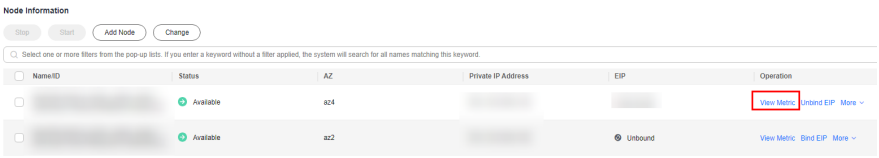
Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the instance whose metrics you want to view and click its name.

Step 4 In the navigation pane, choose **Node Management**. In the **Node Information** area, click **View Metric** in the **Operation** column.

Figure 4-190 Viewing metrics



Step 5 In the monitoring area, you can select a duration to view the monitoring data.

You can view the monitoring data of the service in the last 1, 3, or 12 hours.

To view the monitoring curve in a longer time range, click to enlarge the graph.

----End

4.13.5 Configuring a Dashboard for a GeminiDB Redis Instance

Dashboards, serving as custom monitoring platforms, allow you to view metrics.

This section describes how to configure a dashboard for a GeminiDB Redis instance.

Procedure

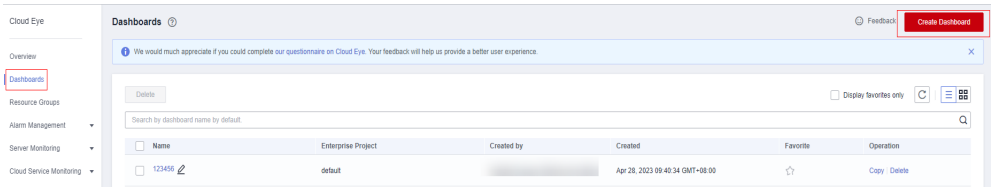
Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, click **Cloud Eye** to go to the Cloud Eye console.

Step 3 Create a dashboard.

1. In the navigation pane on the left, choose **Dashboards**. On the displayed page, click **Create Dashboard**.

Figure 4-191 Creating a dashboard



2. In the displayed **Create Dashboard** dialog box, set required parameters.

Figure 4-192 Configuring parameters

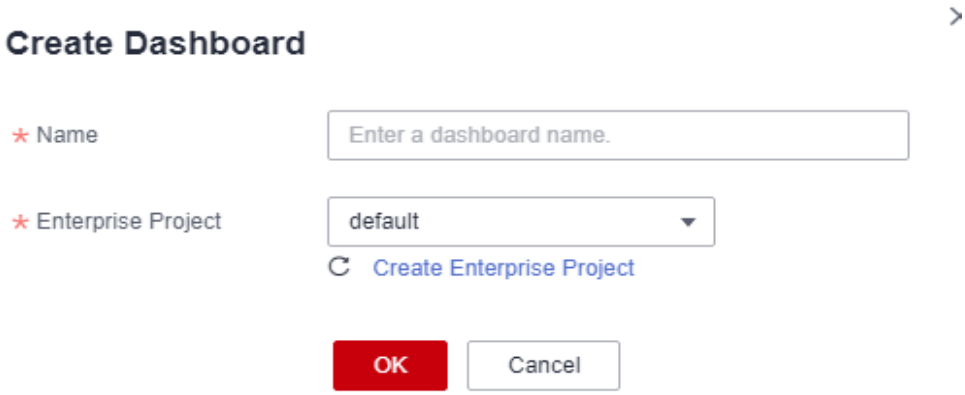


Table 4-79 Parameter description

Parameter	Description
Name	Dashboard name. The name can include a maximum of 128 characters. Only letters, digits, hyphens (-), and underscores (_) are allowed.
Enterprise Project	If you associate a monitoring dashboard with an enterprise project, only users who have the permissions of the enterprise project can manage the monitoring dashboard. **NOTE** The enterprise project feature is available only in some regions.

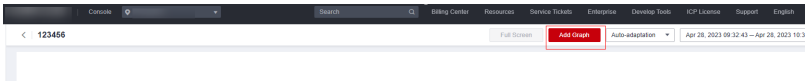
3. Click **OK**.

Step 4 Add a graph to the monitoring dashboard.

After a dashboard is created, you can add graphs to monitor your GeminiDB Redis instances.

1. In the navigation pane on the left, choose **Dashboard**. Locate the dashboard that you want to add a graph to, and click **Add Graph**.

Figure 4-193 Adding a graph



2. On the **Add Graph** page, select a line chart or bar chart.
   - A curve chart reflects changes and peak values of a metric over time.
   - A bar chart reflects metric data of top-ranked resources of the same type, helping you to understand upper and lower limits of a metric.
3. At the **Monitoring Item Configuration** area, configure required parameters by referring to **Table 4-80**.

Figure 4-194 Monitored item configuration

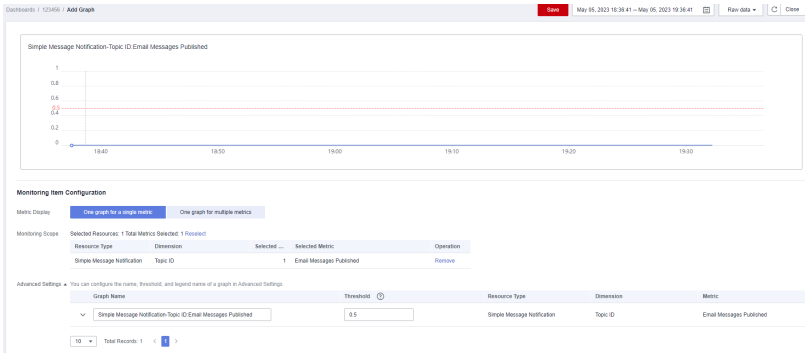


Table 4-80 Parameter description

Parameter	Description
Metric Display	– **One graph for a single metric**: One or more graphs can be generated, and all monitoring items in each graph represent the same metric. – **One graph for multiple metrics**: One graph is generated for multiple metrics, and monitoring items can represent different metrics.
Monitoring Scope	Specify resources and metrics.
Graph Name	Specifies the title of the graph to be added. The name can contain a maximum of 128 characters. Only letters, digits, underscores (_), and hyphens (-) are allowed. Example: CPU usage
Threshold	Configure a threshold to generate an auxiliary line. Data points higher than the line are highlighted in red.
Legend Name	The legend name is displayed on the line in a monitoring graph and can be changed. If you do not configure the legend name, it is displayed in the following format by default: *monitored object* (*resource type*) - *metric*. *monitored data*.

NOTE

When you add a graph, select **One graph for a single metric**. Then a graph is generated for each metric, making it easy for you to view and analyze monitored data. If you need multiple metrics, add monitoring graphs.

4. On the selected dashboard, view metric trends on the added graph.

----End

4.13.6 Event Monitoring

4.13.6.1 Introduction to Event Monitoring

Event monitoring provides event data reporting, query, and alarm reporting. You can create alarm rules for both system and custom events. When a specific event occurs, Cloud Eye generates and sends an alarm for you.

Key operations on GeminiDB Redis resources are monitored and recorded by Cloud Eye as events. Events include operations performed by specific users on specific resources, such as changing instance names and specifications.

Event monitoring provides an API for reporting custom events, which helps you collect and report abnormal events or important change events generated by services to Cloud Eye.

Event monitoring is enabled by default and allows you to view monitoring details of system events and custom events. For details about system events, see **Events Supported by Event Monitoring**.

4.13.6.2 Viewing Event Monitoring Data

Scenarios

Event monitoring provides event data reporting, query, and alarm reporting. You can create alarm rules for both system and custom events. When a specific event occurs, Cloud Eye generates and sends an alarm for you.

Event monitoring is enabled by default. You can view monitoring details about system events and custom events.

This topic describes how to view the event monitoring data.

Procedure

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 You can view event monitoring data in either of the following ways:

- On the **Instances** page, click the instance that you want to view its event monitoring data. In the navigation pane on the left, choose **Metrics**. You can select **DB Instance** or **Node-Level Metrics**.
- In the **Node List** area of the **Basic Information** page, locate a node and click **View Metric** in the **Operation** column.

Step 4 Click < to return to the Cloud Eye console.

Step 5 In the navigation pane on the left, choose **Event Monitoring**.

On the displayed **Event Monitoring** page, all system events generated in the last 24 hours are displayed by default.

You can also click **1h**, **3h**, **12h**, **1d**, **7d**, or **30d** to view events generated in different time periods.

Step 6 Locate an event and click **View Event** in the **Operation** column to view its details.

----End

4.13.6.3 Creating an Alarm Rule for Event Monitoring

Scenarios

This topic describes how to create an alarm rule for event monitoring.

Procedure

Step 1 **Log in to the Huawei Cloud console.**

Step 2 Click ☰ in the upper left corner of the page. Under **Management & Governance**, click **Cloud Eye**.

Step 3 In the navigation pane on the left, choose **Event Monitoring**.

Step 4 On the event list page, click **Create Alarm Rule** in the upper right corner.

Step 5 On the **Create Alarm Rule** page, configure the parameters.

Table 4-81 Parameter description

Parameter	Description
Name	Specifies the name of the alarm rule. The system generates a random name, but you can change it if needed.
Description	(Optional) Provides supplementary information about the alarm rule.
Enterprise Project	You can select an existing enterprise project or click **Create Enterprise Project** to create one.
Alarm Type	Specifies the alarm type corresponding to the alarm rule.
Event Type	Specifies the event type of the metric corresponding to the alarm rule.
Event Source	Specifies the service the event is generated for. Selecting GeminiDB.
Monitoring Scope	Specifies the monitoring scope for event monitoring.

Parameter	Description
Method	Specifies the event creation method.
Alarm Policy	**Event Name** indicates the instantaneous operations users performed on system resources, such as login and logout. For details about events supported by Event Monitoring, see **4.13.6.4 Events Supported by Event Monitoring**. You can select a trigger mode and alarm severity as needed.

Click  to enable alarm notification. The validity period is 24 hours by default. If the topics you require are not displayed in the drop-down list, click **Create an SMN topic**.

Table 4-82 Alarm notification parameters

Parameter	Description
Alarm Notification	Specifies whether to notify users when alarms are triggered. Notifications can be sent by email, text message, or HTTP/HTTPS message.
Notification Object	Specifies the object an alarm notification is to be sent to. You can select the account contact or a topic. • Account contact is the mobile phone number and email address provided for registration. • Topic is used to publish messages and subscribe to notifications. If the required topic is unavailable, create one first and add subscriptions to it. For details, see **Creating a Topic** and **Adding Subscriptions**.
Validity Period	Cloud Eye sends notifications only within the validity period specified in the alarm rule. If you set **Validity Period** to **08:00-20:00**, Cloud Eye sends notifications only within 08:00-20:00.
Trigger Condition	Specifies the condition for triggering the alarm notification.

Step 6 After the configuration is complete, click **Create**.

----End

4.13.6.4 Events Supported by Event Monitoring

Table 4-83 Events supported by Event Monitoring

Event Source	Event Name	Event ID	Event Severity	Description	Solution	Impact
NoSQL	Instance creation failure	NoSQLCreateInstanceFailed	Major	The instance quota or underlying resources are insufficient.	Release unnecessary instances and try again. You can also choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and submit a service ticket to adjust the quota.	Instances fail to be created.
	Specifications change failure	NoSQLResizeInstanceFailed	Major	The underlying resources are insufficient.	Choose **Service Tickets > Create Service Ticket** in the upper right corner of the console. Submit a service ticket to the O&M personnel to coordinate resources in the background and change the specifications again.	Services are interrupted.

Event Source	Event Name	Event ID	Event Severity	Description	Solution	Impact
	Node adding failure	NoSQLAddNodesFailed	Major	The underlying resources are insufficient.	Choose **Service Tickets > Create Service Ticket** in the upper right corner of the console. Submit a service ticket to O&M personnel to coordinate resources in the background, delete nodes that failed to be added, and add the nodes again.	None
	Node deletion failure	NoSQLDeleteNodesFailed	Major	Releasing underlying resources failed.	Delete the node again.	None
	Storage space scale-up failure	NoSQLScaleUpStorageFailed	Major	The underlying resources are insufficient.	Choose **Service Tickets > Create Service Ticket** in the upper right corner of the console. Submit a service ticket to O&M personnel to coordinate resources in the background and scale up storage again.	Services may be interrupted.

Event Source	Event Name	Event ID	Event Severity	Description	Solution	Impact
	Password resetting failure	NoSQLResetPasswordFailed	Major	Resetting the password times out.	Reset the password again.	None
	Parameter template change failure	NoSQLUpdateInstanceParamGroupFailed	Major	Changing a parameter template times out.	Change the parameter template again.	None
	Backup policy configuration failure	NoSQLSetBackupPolicyFailed	Major	The database connection is abnormal.	Configure the backup policy again.	None
	Manual backup creation failure	NoSQLCreateManualBackupFailed	Major	The backup files fail to be exported or uploaded.	Choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and submit a service ticket to O&M personnel.	Data cannot be backed up.
	Automated backup creation failure	NoSQLCreateAutomatedBackupFailed	Major	The backup files fail to be exported or uploaded.	Choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and submit a service ticket.	Data cannot be backed up.

Event Source	Event Name	Event ID	Event Severity	Description	Solution	Impact
	Instance status abnormal	NoSQLFaultyDBInstance	Major	This event is a key alarm event and is reported when an instance is faulty due to a disaster or a server failure.	Choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and submit a service ticket to O&M personnel.	The database service may be unavailable.
	Instance status recovery	NoSQLDBInstanceRecovered	Major	If a disaster occurs, NoSQL provides an HA tool to automatically or manually rectify the fault. After the fault is rectified, this event is reported.	No further action is required.	None
	Node status abnormal	NoSQLFaultyDBNode	Major	This event is a key alarm event and is reported when a database node is faulty due to a disaster or a server failure.	Check whether the database service is functional. Choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and submit a service ticket to O&M personnel.	The database service may be unavailable.

Event Source	Event Name	Event ID	Event Severity	Description	Solution	Impact
	Node status recovery	NoSQLDBNodeRecovered	Major	If a disaster occurs, NoSQL provides an HA tool to automatically or manually rectify the fault. After the fault is rectified, this event is reported.	No further action is required.	None
	Primary/standby switchover or failover	NoSQLPrimaryStandbySwitched	Major	This event is reported when a primary/secondary switchover or a failover is triggered.	No further action is required.	None
	Occurrence of hotspot partitioning keys	HotKeyOccurs	Major	Hotspot data is stored in one partition because the primary key is improper. Improper application design causes frequent read and write operations on a key.	1. Choose a proper partition key. 2. Add service cache so that service applications read hotspot data from the cache first.	The service request success rate is affected, and the cluster performance and stability deteriorates.

Event Source	Event Name	Event ID	Event Severity	Description	Solution	Impact
	BigKey occurrence	BigKeyOccurs	Major	The primary key design is improper. There are too many records or too much data in a single partition, causing load imbalance on nodes.	1. Choose a proper partition key. 2. Add a new partition key for hashing data.	As more and more data is stored in the partition, cluster stability deteriorates.
	Insufficient storage space	NoSQLRiskyDataDiskUsage	Major	The storage space is insufficient.	Scale up storage space. For details, see section "Scaling Up Storage Space" in the user guide of GeminiDB.	The instance is set to read-only and data cannot be written to the instance.
	Data disk expanded and being writable	NoSQLDataDiskUsageRecovered	Major	The data disk has been expanded and becomes writable.	No further action is required.	None

Event Source	Event Name	Event ID	Event Severity	Description	Solution	Impact
	Index creation failure	NoSQLCreateIndexFailed	Major	The service load exceeds what the instance specifications can take. In this case, creating indexes consumes more instance resources. As a result, the response is slow or even frame freezing occurs, and the creation times out.	Select matched instance specifications based on service load. Create indexes during off-peak hours. Create indexes in the background. Select indexes as required.	The index fails to be created or is incomplete. Delete the index and create a new one.
	Write speed decrease	NoSQLStallingOccurs	Major	The write speed is close to the maximum write speed allowed by the cluster scale and instance specifications. As a result, the database flow control mechanism is triggered, and requests may fail.	1. Adjust the cluster scale or node specifications based on the maximum write rate of services. 2. Measure the maximum write rate of services.	The success rate of service requests is affected.

Event Source	Event Name	Event ID	Event Severity	Description	Solution	Impact
	Data write stopped	NoSQLStoppingOccurs	Major	The data write is too fast, reaching the maximum write capability allowed by the cluster scale and instance specifications. As a result, the database flow control mechanism is triggered, and requests may fail.	1. Change the cluster scale or node specifications based on the maximum write rate of services. 2. Measure the maximum write rate of services.	The success rate of service requests is affected.
	Database restart failure	NoSQLRestartDBFailed	Major	The instance status is abnormal.	Choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and submit a service ticket to O&M personnel.	The instance status may be abnormal.

Event Source	Event Name	Event ID	Event Severity	Description	Solution	Impact
	Restoration to new instance failure	NoSQLRestoreToNewInstanceFailed	Major	The underlying resources are insufficient.	Choose **Service Tickets > Create Service Ticket** in the upper right corner of the console. Submit a service ticket to O&M personnel to coordinate resources in the background and add nodes again.	Data cannot be restored to a new instance.
	Restoration to existing instance failure	NoSQLRestoreToExistInstanceFailed	Major	The backup file fails to be downloaded or restored.	Choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and submit a service ticket to O&M personnel.	The current instance may be unavailable.
	Backup file deletion failure	NoSQLDeleteBackupFailed	Major	The backup files fail to be deleted from OBS.	Delete the backup files again.	None

Event Source	Event Name	Event ID	Event Severity	Description	Solution	Impact
	Failure to display slow query logs in plaintext	NoSQLSwitchSlowlogPlainTextFailed	Major	The API does not support this function.	Check whether that the API supports slow query logs in plaintext by following *GeminiDB User Guide*. Choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and submit a service ticket to O&M personnel.	None
	EIP binding failure	NoSQLBindEipFailed	Major	The node status is abnormal, an EIP has been bound to the node, or the EIP to be bound is invalid.	Check whether the node is normal and whether the EIP is valid.	The instance cannot be accessed from a public network.
	EIP unbinding failure	NoSQLUnbindEipFailed	Major	The node status is abnormal or the EIP has been unbound from the node.	Check whether the node and EIP status are normal.	None

Event Source	Event Name	Event ID	Event Severity	Description	Solution	Impact
	Parameter modification failure	NoSQLModifyParameterFailed	Major	The parameter value is invalid.	Check whether the parameter value is valid. Choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and submit a service ticket to O&M personnel.	None
	Parameter template application failure	NoSQLApplyParameterGroupFailed	Major	The instance status is abnormal. So, the parameter template cannot be applied.	Choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and submit a service ticket to O&M personnel.	None
	Enabling or disabling SSL failure	NoSQLSwitchSSLFailed	Major	Enabling or disabling SSL times out.	Try again or choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and submit a service ticket. Retain the SSL connection mode configured before the event.	The SSL connection mode cannot be changed.

Event Source	Event Name	Event ID	Event Severity	Description	Solution	Impact
	Too much data in a single row	LargeRowOccurs	Major	If there is too much data in a single row, queries may time out, causing faults like OOM error.	1. Limit the write length of each column and row so that the key and value length of each row does not exceed the preset threshold. 2. Check whether there are abnormal writes or coding, causing large rows.	If there are too many records in a single row, cluster stability will deteriorate as the data volume increases.

4.14 Tag Management

Scenarios

Tag Management Service (TMS) enables you to use tags on the management console to manage resources. TMS works with other cloud services to manage global tags, and other cloud services manage their own tags.

Adding tags to GeminiDB Redis instances helps you better identify and manage them. A DB instance can be tagged during or after it is created.

After a DB instance is tagged, you can search for the tag key or value to quickly query the instance details.

Usage Notes

- You are advised to set predefined tags on the TMS console.
- A tag consists of a key and value. You can add only one value for each key. For details about the naming rules of tag keys and tag values, see **Table 4-84**.
- A maximum of 20 tags can be added for each instance.
- The tag name must comply with the naming rules described in **Table 4-84**.

Table 4-84 Naming rules

Parameter	Requirement	Example Value
Tag key	• Cannot be left blank. • Must be unique for each instance. • Contains a maximum of 36 characters. • Can only consist of digits, letters, underscores (_), and hyphens (-).	Organization
Tag value	• Can be left blank. • Contains a maximum of 43 characters. • Can only consist of digits, letters, underscores (_), periods (.), and hyphens (-).	nosql_01

Adding a Tag

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, click the instance that you want to add tags to and click its name.

Step 4 In the navigation pane on the left, choose **Tags**.

Step 5 On the **Tags** page, click **Add Tag**. In the displayed dialog box, enter a tag key and value, and click **OK**.

Step 6 View and manage the tag on the **Tags** page.

----End

Editing a Tag

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the instance whose tags you want to edit and click its name.

Step 4 In the navigation pane on the left, choose **Tags**.

Step 5 On the **Tags** page, locate the tag to be edited and click **Edit** in the **Operation** column. In the displayed dialog box, change the tag value and click **OK**.

Only the tag value can be edited.

Step 6 View and manage the tag on the **Tags** page.

----End

Deleting a Tag

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 On the **Instances** page, locate the instance whose tags you want to delete and click its name.

Step 4 In the navigation pane on the left, choose **Tags**.

Step 5 On the **Tags** page, locate the tag to be deleted and click **Delete** in the **Operation** column. In the displayed dialog box, click **Yes**.

Step 6 View that the tag is no longer displayed on the **Tags** page.

----End

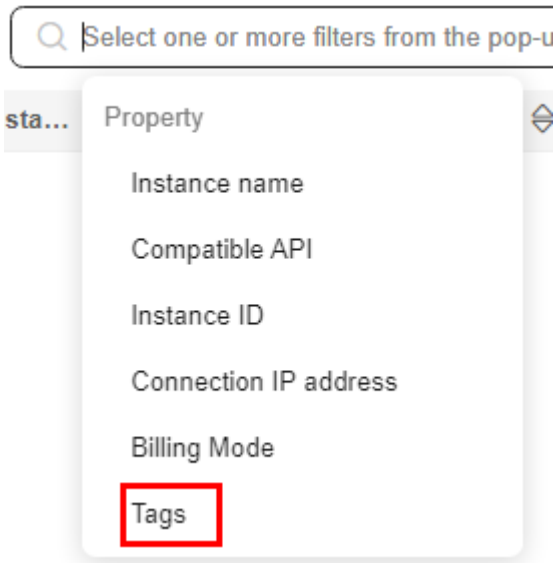
Searching an Instance by Tag

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

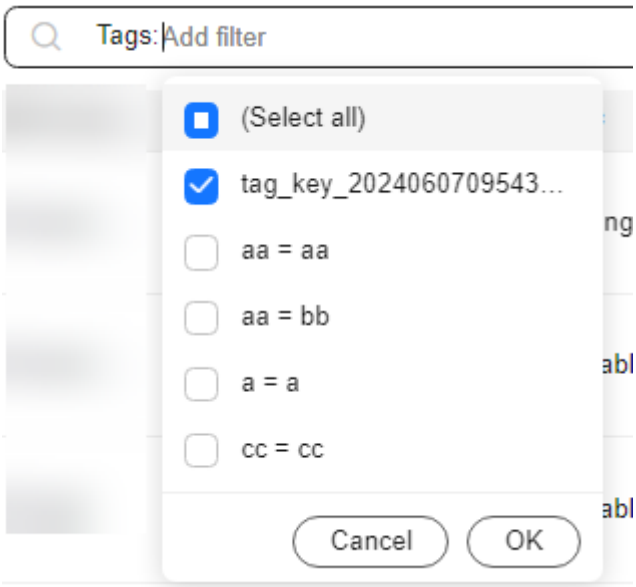
Step 3 On the **Instances** page, select **Tags** in the search box.

Figure 4-195 Selecting tags



Step 4 Select the tag to be queried and click **OK** to query information about instances associated with the tag.

Figure 4-196 Searching by tag



----End

4.15 Quota

Scenarios

Quotas are enforced for service resources on the platform to prevent unforeseen spikes in resource usage. Quotas limit the number or amount of resources available to users, for example, the maximum number of GeminiDB instances that you can create.

If a quota cannot meet your needs, apply for a higher quota.

Viewing Quotas

Step 1 **Log in to the Huawei Cloud console.**

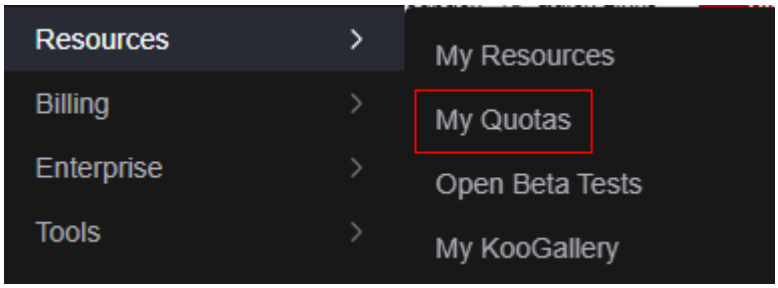
Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 Click in the upper left corner and select a region and project.

Step 4 In the upper right corner, choose **Resources** > **My Quotas**.

The **Quota** page is displayed.

Figure 4-197 My quotas



Step 5 On the **Quotas** page, view the used and total quotas of each type of GeminiDB resources.

----End

Increasing Quotas

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 Click in the upper left corner and select a region and project.

Step 4 In the upper right corner, choose **Resources** > **My Quotas**.

Step 5 In the upper right corner of the page, click **Increase Quota**.

Figure 4-198 Increasing quotas

Step 6 On the **Create Service Ticket** page, configure parameters as required.

In the **Problem Description** area, enter the required quota and reason for the quota adjustment.

Step 7 After all necessary parameters are configured, select the agreement and click **Submit**.

----End

4.16 MySQL Memory Acceleration

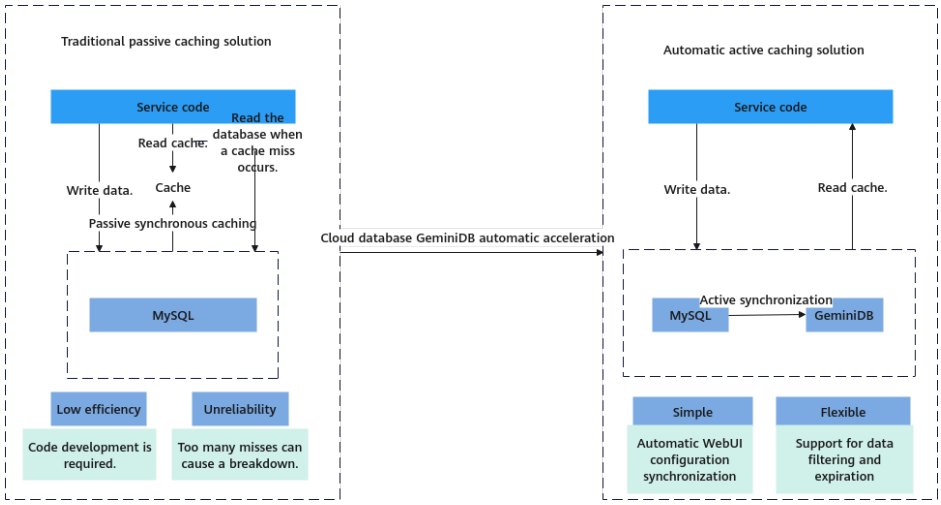
4.16.1 Memory Acceleration Overview

GeminiDB Redis API offers memory acceleration to enhance the conventional cache solution. With this feature, users can set up rules on the GUI to cache MySQL data automatically, thereby speeding up MySQL access.

The conventional cache solution is inefficient and unreliable as it necessitates users to create code for writing MySQL data to the cache. The active cache

solution with cloud data memory acceleration (DB Cache) supports visualized configuration on the GUI, making it easier to set up. Once the configuration is done, data can be synchronized automatically. DB Cache also supports data filtering and expiration time setting, which enhances development efficiency and data reliability.

Figure 4-199 Diagram



4.16.2 Enabling and Using Memory Acceleration

Enable memory acceleration.

Step 1: Create a GeminiDB instance.

Step 2: Create a mapping rule.

Step 3: Use the memory acceleration module.

Precautions

- After memory acceleration is enabled, commands such as RESET MASTER and FLUSH LOGS used to delete binlogs on MySQL instances are not allowed.
- Currently, only hash data from MySQL can be converted to GeminiDB Redis API.
- A Redis key prefix and a delimiter in a new rule can neither include those nor be included in those specified for an existing rule. For example, if the key prefix in a new rule is **pre1:** and is separated by a comma (,) and the key prefix in an existing rule is **pre1** and is separated by a colon (:), the new rule cannot be created.
- Currently, the ENUM, SET, and JSON data cannot be synchronized.

Procedure

Creating a GeminiDB Instance

Step 1 **Log in to the management console**.

Step 2 Click  in the upper left corner and select a region and project.

Step 3 Click  in the upper left corner of the page and choose **Databases** > **Relational Database Service**.

Step 4 On the **Instances** page, click the target instance.

Step 5 In the navigation pane on the left, choose **Memory Acceleration**. On the displayed page, click **Create GeminiDB Instance**.

Step 6 On the displayed page, set required parameters and click **Submit**.

Table 4-85 Basic information

Parameter	Description
Instance Class	CPU and memory of the instance. For details, see **Table 4-86**.
Database Port	Port number for accessing the instance. You can specify a port number based on your requirements. The port number ranges from 1024 to 65535 except 2180, 2887, 3887, 6377, 6378, 6380, 8018, 8079, 8091, 8479, 8484, 8999, 12017, 12333, and 50069. If you do not specify a port number, port 6379 is used by default. **NOTE** You cannot change the database port after an instance is created.
DB Instance Name	The instance name: • Can be the same as an existing instance name. • Can include 4 to 64 bytes and must start with a letter. It is case-sensitive and allows only letters, digits, hyphens (-), and underscores (_).
Database Password	Database password set by the user. • Must be 8 to 32 characters long. • Can include two of the following: uppercase letters, lowercase letters, digits, and special characters: ~!@#%^*-_=+? • For security reasons, set a strong password. The system will verify the password strength. Keep your password secure. The system cannot retrieve it if it is lost.
Confirm Password	Enter the administrator password again.

Table 4-86 GeminiDB Redis instance specifications

Storage (GB)	Nodes	vCPUs	QPS	Maximum Connections per Single-node Instance	Databases
24	2	2	40,000	10,000	1,000
32	2	2	40,000	10,000	1,000
48	2	4	80,000	20,000	1,000
64	2	4	80,000	20,000	1,000
96	2	8	160,000	20,000	1,000
128	2	16	320,000	20,000	1,000

----End

Creating a Mapping Rule

Step 1 Log in to the Huawei Cloud console.

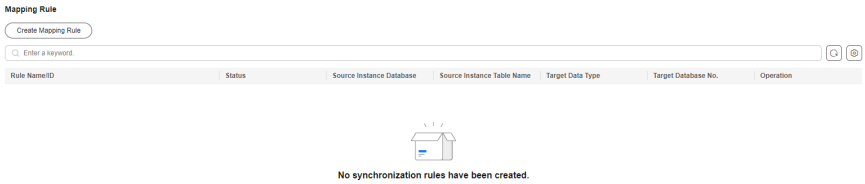
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **Relational Database Service**.

Step 4 On the **Instances** page, click the target instance.

Step 5 In the navigation pane, choose **Memory Acceleration**. In the **Mapping Rule** area, click **Create Mapping Rule**.

Figure 4-200 Mapping rule

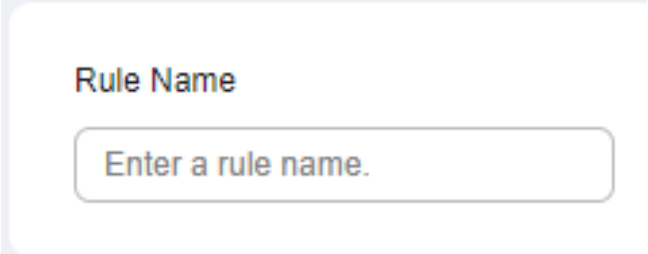


Step 6 On the displayed page, configure parameters.

1. Enter a rule name.

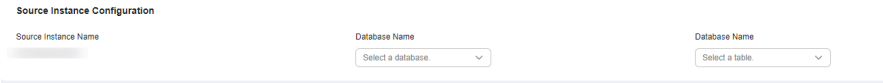
   Rule Name: Enter a mapping rule name. The rule name must be unique within a GeminiDB instance and cannot exceed 256 characters or include number signs (#).

Figure 4-201 Rule name



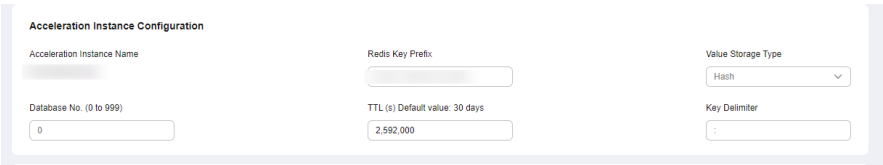
2. Configure source instance information.
   - **Database Name**: Select a database of the acceleration instance.
   - **Table Name**: Select a table of the acceleration instance.

Figure 4-202 Configuring source instance information



3. Configure acceleration instance information.
   - **Redis Key Prefix**: This parameter is optional. The default value is in the format of *Database name*:*Table name*:*Field name 1*:*Field name 2...* and can contain a maximum of 1,024 characters. If you have created a custom prefix, it will be used instead of the default one.
   - **Value Storage Type**: Data type of the cache. Currently, only hash data is supported.
   - **Database No. (0-999)**: ID of a database that stores cached data in the acceleration instance. The default value is **0**.
   - **TTL (s) Default value: 30 days**: Validity period of cached data in the acceleration instance. The default value is 30 days (2,592,000 seconds). If you enter **-1**, the cached data will never expire.
   - **Key Delimiter**: Separator among the Redis key prefix, key, and key fields. It is a single character in length.

Figure 4-203 Configuring acceleration instance information

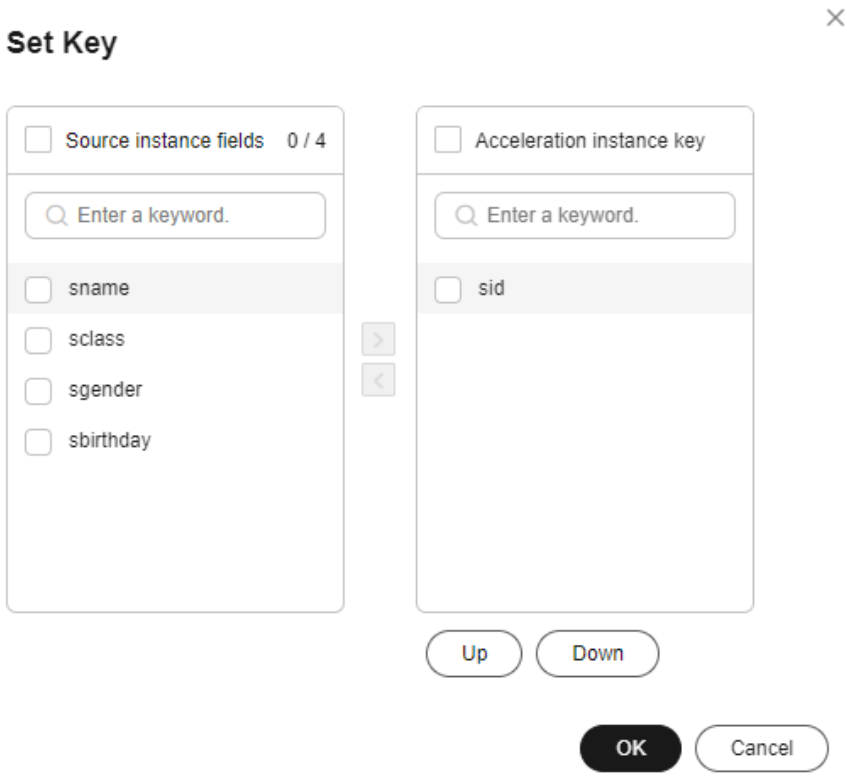


4. Click **Set Key**, select a key field of the acceleration instance, and click **OK**.

   NOTE

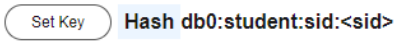
   If an acceleration instance key consists of multiple source instance fields, the key must be unique in a MySQL instance. You can click **Up** or **Down** to adjust the sequence of each field.

Figure 4-204 Key settings



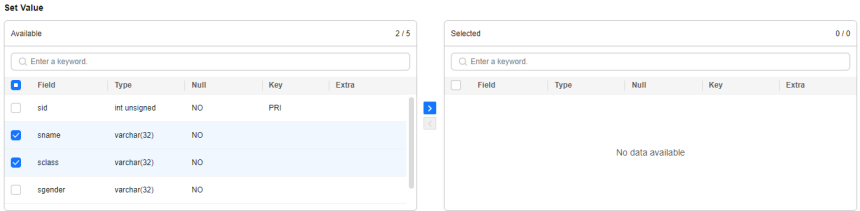
After the parameters are set, the key is displayed.

Figure 4-205 Key



5. Configure the acceleration instance fields.
   Move the required fields in the source instance to the acceleration instance.

   Figure 4-206 Configuring acceleration instance fields



6. After setting the parameters, click **Submit**.

----End

Using the Memory Acceleration Module

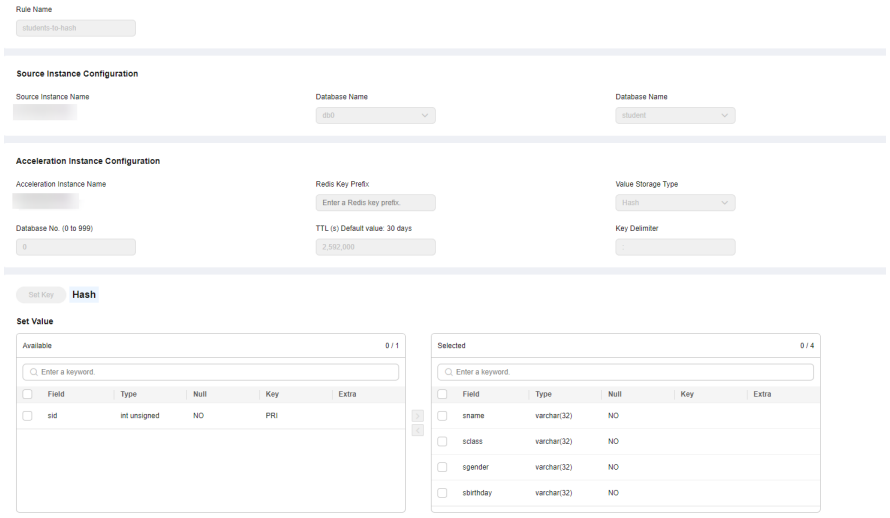
1. Create database **db1** in the source MySQL instance and create table **students** in **db1**.

```
mysql> CREATE DATABASE db1;
Query OK, 1 row affected (0.00 sec)
```

```
mysql> CREATE TABLE db1.students(
      sid INT UNSIGNED PRIMARY KEY AUTO_INCREMENT NOT NULL,
      sname VARCHAR(20),
      sclass INT,
      sgender VARCHAR(10),
      sbirthday DATE
      );
Query OK, 0 rows affected (0.00 sec)

mysql> DESC db1.students;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| sid       | int unsigned | NO   | PRI | NULL    | auto_increment |
| sname     | varchar(20)  | YES  |     | NULL    |                |
| sclass    | int          | YES  |     | NULL    |                |
| sgender   | varchar(10)  | YES  |     | NULL    |                |
| sbirthday | date         | YES  |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

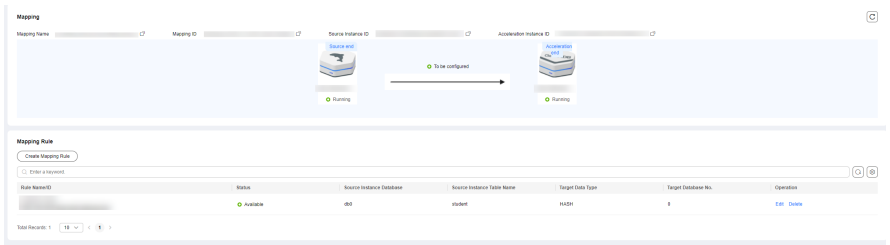
2. After the table is created, on the memory acceleration page, create a mapping rule to convert each row in the **students** table into a Redis hash. The key of a hash is in the format of *Database name*:*Data table name*:**sid:**<*sid value*>. The selected fields are **sname**, **sclass**, **sgender**, and **sbirthday**.

Figure 4-207 Configuring a mapping rule



3. After a mapping rule is created, check the mapping rule and information.

Figure 4-208 Mapping information



4. Insert a new data record to the **students** table in the MySQL instance.

```
mysql> INSERT INTO db1.students (sname, sclass, sgender, sbirthday) VALUES ('zhangsan', 1, 'male', '2015-05-20');
```



```
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM db1.students;
+-----+----------+--------+---------+------------+
| sid | sname    | sclass | sgender | sbirthday  |
+-----+----------+--------+---------+------------+
|   1 | zhangsan |      1 | male    | 2015-05-20 |
+-----+----------+--------+---------+------------+
1 row in set (0.00 sec)
```

5. After the mapping rule is created, the data is automatically synchronized to the GeminiDB instance. Run commands in the GeminiDB instance to query the data.

```
127.0.0.1:6379> KEYS *
1) "db1:students:sid:1"

127.0.0.1:6379> HGETALL db1:students:sid:1
1) "sbirthday"
2) "2015-05-20"
3) "sclass"
4) "1"
5) "sgender"
6) "male"
7) "sname"
8) "zhangsan"
```

6. Insert a new data record to the **students** table in the MySQL instance.

```
mysql> INSERT INTO db1.students (sname, sclass, sgender, sbirthday) VALUES ('lisi', 10, 'male',
'2015-05-22');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM db1.students;
+-----+----------+--------+---------+------------+
| sid | sname    | sclass | sgender | sbirthday  |
+-----+----------+--------+---------+------------+
|   1 | zhangsan |      1 | male    | 2015-05-20 |
|   2 | lisi     |     10 | male    | 2015-05-22 |
+-----+----------+--------+---------+------------+
2 rows in set (0.00 sec)
```

7. Check whether the new data is synchronized to the GeminiDB instance.

```
127.0.0.1:6379> KEYS *
1) "db1:students:sid:1"
2) "db1:students:sid:2"

127.0.0.1:6379> HGETALL db1:students:sid:2
1) "sbirthday"
2) "2015-05-22"
3) "sclass"
4) "10"
5) "sgender"
6) "male"
7) "sname"
```

8. Update data in the **students** table in the MySQL instance.

```
mysql> UPDATE db1.students SET sclass=12, sname='wangwu' WHERE sid = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM db1.students;
+-----+--------+--------+---------+------------+
| sid | sname  | sclass | sgender | sbirthday  |
+-----+--------+--------+---------+------------+
|   1 | wangwu |     12 | male    | 2015-05-20 |
|   2 | lisi   |     10 | male    | 2015-05-22 |
+-----+--------+--------+---------+------------+
2 rows in set (0.00 sec)
```

9. Check whether the data is updated in the GeminiDB instance.

```
127.0.0.1:6379> KEYS *
1) "db1:students:sid:1"
```



```
2) "db1:students:sid:2"

127.0.0.1:6379> HGETALL db1:students:sid:1
1) "sbirthday"
2) "2015-05-20"
3) "sclass"
4) "12"
5) "sgender"
6) "male"
7) "sname"
8) "wangwu"
```

10. Delete data from the **students** table in the MySQL instance.

```
mysql> DELETE FROM db1.students WHERE sid = 1;
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM db1.students;
+-----+-------+--------+---------+------------+
| sid | sname | sclass | sgender | sbirthday  |
+-----+-------+--------+---------+------------+
|   2 | lisi  |     10 | male    | 2015-05-22 |
+-----+-------+--------+---------+------------+
1 row in set (0.00 sec)
```

11. Check whether the data is deleted from the GeminiDB instance.

```
127.0.0.1:6379> KEYS *
1) "db1:students:sid:2"
```

4.16.3 Modifying and Deleting a Memory Acceleration Rule

A memory acceleration rule can enable automated data synchronization from MySQL to GeminiDB. You can also modify and delete this rule.

Precautions

- Currently, only hashes from MySQL can be converted to GeminiDB Redis API.
- If a table name of the MySQL instance in the memory acceleration rule is changed, you need to reconfigure the rule.
- Currently, the ENUM, SET, and JSON data cannot be synchronized.
- If you rename or delete one or more key fields of a memory acceleration rule, the rule becomes invalid.

Modifying a Mapping Rule

Step 1 **Log in to the management console**.

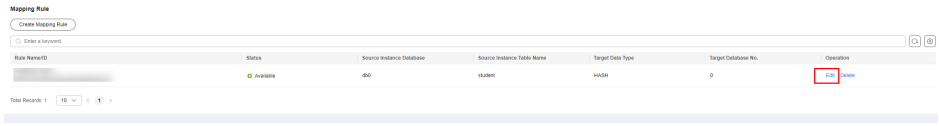
Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **Relational Database Service**.

Step 4 On the **Instances** page, click the target instance.

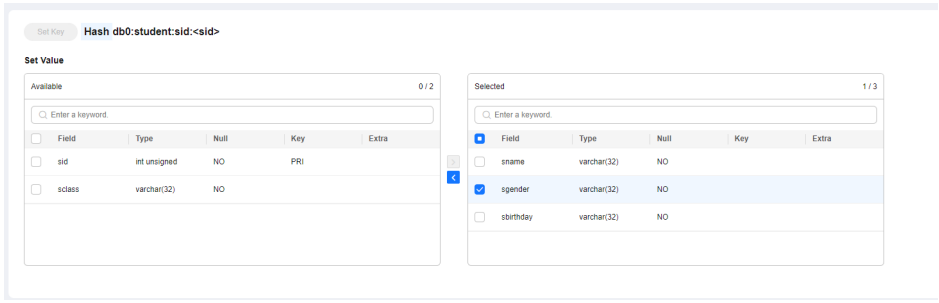
Step 5 In the navigation pane on the left, choose **Memory Acceleration**. In the **Mapping Rule** area, locate the target rule and click **Edit** in the **Operation** column.

Figure 4-209 The **Edit** button



Step 6 After editing the fields, click **Submit**.

Figure 4-210 Editing a mapping rule



----End

Deleting a Mapping Rule

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region and project.

Step 3 Click in the upper left corner of the page and choose **Databases** > **Relational Database Service**.

Step 4 On the **Instances** page, click the target instance.

Step 5 In the navigation pane on the left, choose **Memory Acceleration**. In the **Mapping Rule** area, locate the target rule and click **Delete** in the **Operation** column.

----End

4.16.4 Viewing and Removing Mappings

You can view the mapping list on the **Memory Acceleration Management** page and remove mappings.

Precautions

- After a mapping is removed, service applications cannot obtain the latest data in the source database through the acceleration instance. A free GeminiDB instance will be re-billed once the mapping is removed.
- The corresponding mapping rule will be cleared after a mapping is removed.
- If the source instance or acceleration instance is not normal, the mapping cannot be removed.

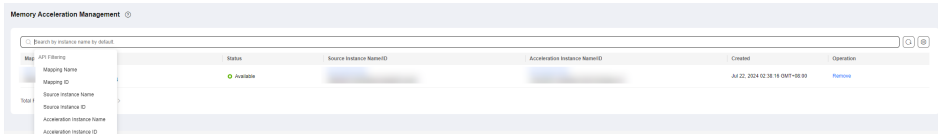
Querying the Mapping List

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the navigation pane on the left, choose **Memory Acceleration Management**. On the displayed page, search for your target mapping by keyword (such as the mapping name or mapping ID).

Figure 4-211 Mapping list



----End

Unmapping

Step 1 Log in to the Huawei Cloud console.

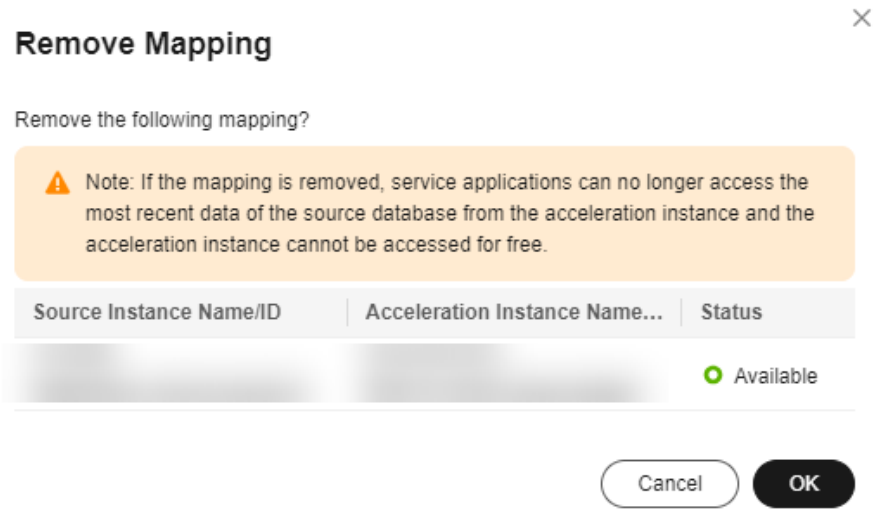
Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

Step 3 In the navigation pane on the left, choose **Memory Acceleration Management**. On the displayed page, locate the target mapping and click **Remove** in the **Operation**.

Figure 4-212 Memory acceleration management

Step 4 In the displayed dialog box, click **OK**.

Figure 4-213 Unmapping



----End

5 Development Reference

5.1 Development and O&M Rules

This section describes usage rules of the GeminiDB Redis database based on Huawei's years of experience in cloud database development and O&M, so as to help you effectively evaluate and improve service system stability.

Development Rules

When developing a service program, you need to pay attention to the following development rules to prevent service instability caused by improper usage.

Table 5-1 Development rules of GeminiDB Redis

No.	Development Rules	Description
1	The service program must have a proper automatic reconnection mechanism.	In scenarios for the specification change, patch upgrade, HA switchover, network link jitter, or packet loss, the connection between the service program and the DB instance may be interrupted for a short period of time. The service program must support automatic reconnection. **NOTE** Use Jedis instead of Lettuce because Lettuce is not automatically reconnected after multiple requests time out.
2	Provide a connection pool and configure sufficient connections for a service program.	To prevent the program from failing to obtain connections when the number of concurrent requests increases sharply, you are advised to use a connection pool for the service program and configure proper connection pool parameters. For details about the recommended configuration of the connection pool of the client, see **5.8 Configuring Parameters for a Client Connection Pool**.
3	The service program must have a proper command retry mechanism for important operations.	When the connection is interrupted or the request times out, requests from the service program may fail to be executed within a short period of time. Therefore, a service fault tolerance mechanism is needed. Proper command retry intervals and times can ensure important data is successfully written or modified.
4	A correct HA connection address needs to be used to prevent services from being affected by a single point of failure.	When connecting a service program to a database through a private network, use a load balancer IP address to achieve HA. Do not directly connect the service program to an independent compute node. **NOTE** If a database is accessed from a public network, do not directly connect the service program to an independent compute node.

No.	Development Rules	Description
5	A connection pool needs to be used to avoid using a single connection or a large number of short connections.	A single connection has HA risks. The performance of short connections is poor, and short connections consume a large number of CPUs and network resources, which may cause bottlenecks. Therefore, you are advised to use mainstream SDK connection pools to connect a service program to GeminiDB Redis instances.
6	You are not allowed to run the **KEYS** command when there are more than 1,000 keys in an instance.	The **KEYS** command is a typical high-risk command. It obtains all data in the entire instance at a time and returns the data to the client. If there are a large number of keys in an instance, running the **KEYS** command will block requests or make the instance unavailable.
7	Do not set element values to excessively large or small values and ensure the number of elements contained in a single key is not too large.	According to best practices, the size upper limit of a string key is 10 KB, the number of elements in a Hash, List, Zset, or Set key should be less than 5,000, and the size of a single element should be 1 KB or smaller. **NOTE** Similar to the community Redis, GeminiDB Redis does not strictly restrict the storage of big keys. Therefore, you need to develop service programs within the rule scope.

No.	Development Rules	Description
8	A single command does not contain many elements at a time. Too large network packets cannot be generated by a single command.	• It is recommended that no more than 1000 keys be concurrently operated using MSET or MGET. • Make sure that no more than 1000 key elements to be concurrently operated using HMSET, HGETALL, LRANGE, ZADD, or ZRANGE. • The **limit** parameter is not included in the Redis **ZREMRANGEBYSCORE** command, so the number of elements to be deleted at a time cannot be limited. You are advised to use **ZRANGEBYSCORE** (with limit) + **ZREM** instead. **NOTE** Similar to the community Redis, GeminiDB Redis does not strictly restrict the access to big keys. Therefore, you need to develop a service program within the rule scope.
9	Properly distribute keys to avoid performance bottlenecks in hotspot keys or hash tags.	Frequent access to a single key or a group of keys with the same hash tags may cause hot key problems, which may cause bottlenecks such as compute resource skew, request queuing, and slow response. Hot keys are usually created due to insufficient key splitting in service design. Therefore, service splitting needs to be optimized. **NOTE** Negative examples: • Ultra-large global rankings are accessed frequently and intensively. • Several hash keys are stored in an instance. Each key is used to store information of the entire table. • For product inventory, there are only a small number of bucket hash tags for all keys. As a result, requests for querying hot hash tags may queue.

No.	Development Rules	Description
10	Run less than 100 packaging commands at a time using the pipeline.	Do not include a large number of commands in a pipeline. Using pipelines improperly may cause bottlenecks in CPU and bandwidth resources and block requests. **NOTE** Compared with the community Redis, GeminiDB Redis does not strictly restrict the use of pipelines. Therefore, you need to develop a service program within the rule scope.
11	Do not use time-consuming code in Lua scripts.	When using LUA scripts, excessive command execution times and time-consuming statements such as long-time sleep and large loop statements should be avoided.
12	Do not pack too many commands in a transaction.	When using a transaction, do not pack too many commands or complex commands in a single transaction. If a transaction contains too many commands, requests are blocked, or the instance may become abnormal.
13	Different data types cannot use the same key.	The community Redis allows no use of the same key in different data types. Although GeminiDB Redis has no requirements for this, avoid using the same key in different data types during program development to ensure the program is clear and easy to maintain.
14	Exercise caution when running batch deletion commands.	Do not delete too much data (tens or hundreds of thousands of elements) of the LIST and ZSET types via a single batch deletion command (for example, **LREM**, **LTRIM**, and **ZREMBYSCORE**). Deleting a large amount of data may take a long time, affecting other commands. Processes causing OOM may restart and the instance may be abnormal. In extreme scenarios, service processes may fail to be started repeatedly.

O&M Rules

During routine O&M, you need to pay attention to the following O&M rules to prevent potential risks and master key emergency solutions.

Table 5-2 O&M rules of GeminiDB Redis

No.	Rule	Description
1	Ensure that the phone number and email address bound to your Huawei Cloud account are valid so that you can receive notifications in a timely manner.	Huawei Cloud will send notifications to you through websites, emails, SMS messages, or internal messages in scenarios for changes, upgrades, and fault notifications. Ensure that the contact information bound to your account is valid.
2	Subscribe to major alarms.	• You can subscribe to alarms such as big key access, high storage utilization, high connection usage, and high CPU usage to detect and handle instance risks in a timely manner. • For details, see **4.13.2 Configuring Alarm Rules**.
3	When the LB address is used for access, you need to configure access control instead of security groups.	A load balancer address does not support security groups. After instance creation is complete, configure IP address access control. If no whitelist is configured, all IP addresses that can communicate with the VPC can access the instance.
4	Configure autoscaling for instances.	GeminiDB Redis instance storage can be automatically scaled up in case of a sudden surge in data volumes. Enable autoscaling by following **4.6.7.3 Automatically Scaling Up Disk Space**.

No.	Rule	Description
5	Keep the load at the healthy level.	• If the service data volume is greater than 80% for a long time, you are advised to expand the capacity in a timely manner. • If the service traffic exceeds the QPS limit of an instance or the CPU usage maintains 80% or more for a long time, upgrade the specifications or add nodes to prevent overloading and affecting service access. • When the instance computing power is overloaded due to the sharp increase of service traffic and the number of connections, you can add nodes to quickly improve the cluster computing power. Scaling up CPU cores is performed in rolling mode and takes a long time, which is not recommended in emergency scenarios.
6	Rename high-risk commands.	Disable or rename high-risk commands such as FLUSHALL, or KEYS to enhance instance security. For details, see **Renaming Commands**.
7	Exercise caution when performing the FLUSHALL operation during account management.	• When an IAM user with the read and write permissions executes FLUSHALL, all data on the instance is cleared. • Do not perform FLUSHALL. You can perform FLUSHDB after confirming the database. Exercise caution when performing this operation.
8	Perform periodical online diagnosis of big keys.	You can periodically view the big key diagnosis report of the instance on the console to view keys that are the most frequently accessed in the Redis database. For details, see **4.9.1 Big Key Diagnosis**.

No.	Rule	Description
9	Run the **DBSIZE** command after data migration.	Running **DBSIZE** can ensure data consistency. For example, if there are no expired keys, you can run **DBSIZE** several minutes after data is imported to obtain the accurate number of keys and ensure data consistency.

5.2 Compatible Commands

GeminiDB Redis API provides cluster and primary/standby instances. It is compatible with Redis Community Edition 5.0 and earlier versions, Redis 6.2 (including 6.2.*X*), and Redis 7.0 commands. You do not need to modify code when migrating applications to the cloud.

This section describes the compatibility of commands supported by GeminiDB Redis API 5.0, 6.2 (including 6.2.*X*), and 7.0, providing references to DBAs and developers.

Basic GeminiDB Commands

The following table lists commands provided by community Redis and compatibility of GeminiDB Redis API 5.0, 6.2 (including 6.2.*X*), and 7.0.

Table 5-3 Command compatibility of GeminiDB Redis API

Redis Command Classification	Description	Compatibility with GeminiDB Redis 5.0	Compatibility with GeminiDB Redis API 6.2 (Including 6.2.*x*)	Compatibility with GeminiDB Redis API 7.0
String	String commands	100%	100%	100%
Hash	Hash commands	100%	100%	100%
List	List commands	100%	100%	100%

Redis Command Classification	Description	Compatibility with GeminiDB Redis 5.0	Compatibility with GeminiDB Redis API 6.2 (Including 6.2.*x*)	Compatibility with GeminiDB Redis API 7.0
Sorted set	**Sorted set commands**	100%	100%	100%
Set	**Set commands**	100%	100%	100%
Bitmap	**Bitmap commands**	100%	100%	100%
Stream	**Stream commands**	100%	The XGROUP subcommand createconsumer is not supported.	Features of version 7.0 are not supported.
GEO	**GEO commands**	100%	100%	100%
HyperLogLog	**HyperLogLog commands**	100%	100%	100%
Pub/Sub	**Pub/Sub commands**	100%	100%	100%

Redis Command Classification	Description	Compatibility with GeminiDB Redis 5.0	Compatibility with GeminiDB Redis API 6.2 (Including 6.2.*x*)	Compatibility with GeminiDB Redis API 7.0
Scripting and function	**Scripting and function commands**	100%	100%	Features of version 7.0 are not supported.
Transactions	**Transaction commands**	100%	100%	100%
Generic	**Generic commands**	SWAPDB and MOVE commands are not supported.	The following commands are not supported: SWAPDB and MOVE	100%

GeminiDB Redis API is compatible with some CLUSTER commands, including CLUSTER INFO, CLUSTER KEYSLOT, CLUSTER MYID, CLUSTER NODES, CLUSTER SLOTS, CLUSTER SHARDS and CLUSTER HELP.

Advanced GeminiDB Commands

- EXHASH
  - Application scenarios: GeminiDB allows you to set an expiration time for each field of a hash key and is suitable for services such as frequency control and shopping cart.
  - Command list: For details, see **5.6 EXHASH Commands**.
  - Usage: For details, see **6.4 EXHASH for Ad Frequency Control**.
- Bloom filter

- Functions: A Bloom filter enables you to check if an element is present in a large-size collection. It is applicable to scenarios such as web interceptors and anti-cache penetration.
- Command list: See **Bloom filter**.
- Usage: See **Bloom filter description**.

GeminiDB O&M Commands

GeminiDB provides comprehensive and easy-to-use O&M functions based on the community Redis commands such as INFO, CLIENT, SLOWLOG, MONITOR, and CONFIG.

Table 5-4 O&M functions of GeminiDB Redis API

Function	GeminiDB O&M Functions	Community Redis Commands/Capabilities
Metrics	QPS, average latency, and p99 latency of commands, various metrics of nodes, and aggregation metrics of instances are provided.	INFO
Instance sessions	Client IP addresses and the top sessions of an instance are displayed. Users can kill sessions in batches with a few clicks.	CLIENT
Parameter configuration	Kernel parameters can be queried and modified with only a few clicks. Enhancement of CONFIG command provided by Redis Community Edition.	CONFIG
Viewing slow query logs	Slow query logs and diagnosis information can be displayed.	SLOWLOG
User permission management	The account management function is supported.	ACL (Access Control List)
Viewing audit logs	High-risk commands and operations can be audited.	MONITOR
Real-time big key diagnosis	Big keys are monitored in real time and without affecting services.	Not supported
Real-time hot key diagnosis	Hot keys are monitored in real time and without affecting services.	Not supported

Function	GeminiDB O&M Functions	Community Redis Commands/Capabilities
Real-time key prefix analysis	Distribution of key prefixes is monitored in real time, which does not affect services.	Not supported
Critical command renaming	Users can modify command aliases and view renaming records.	You can rename commands by modifying configuration files.
Abnormal key circuit break	Specified keys can be shielded in one click to avoid access to services, which can be used for rapid data recovery.	Not supported

Precautions

- GeminiDB Redis API does not support RESP3. Redis Serialization Protocol (RESP) is used for communication between a Redis server and a client. Mainstream clients such as Jedis use RESP2 by default, which can be used to access a GeminiDB Redis instance.
- GeminiDB Redis API 7.0 does not support functions.

5.3 Examples of Connecting to an Instance Using Programming Languages

5.3.1 Connecting to an Instance Using Jedis

This section describes how to access a GeminiDB Redis instance using the Java client, Jedis.

The proxy cluster architecture of GeminiDB Redis API provides a unified load balancing address and high availability. So, JedisPool is recommended for easy access.

JedisSentinelPool and JedisCluster can also be used to connect to GeminiDB Redis instances.

Prerequisites

- A GeminiDB Redis instance has been created and is running properly. For details about how to create a GeminiDB Redis instance, see **4.2.1 Buying a GeminiDB Redis Cluster Instance**.
- An ECS is available. For details, see **Purchasing an ECS**.
- GNU Compiler Collection (GCC) has been installed on the ECS.
- The created ECS is in the same region, AZ, VPC, and security group as the GeminiDB Redis instance.

Dependencies on the POM File

```
<dependency>
    <groupId>redis.clients</groupId>
    <artifactId>jedis</artifactId>
    <version>4.3.2</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
    <version>2.3.6.RELEASE</version>
</dependency>
```

Using JedisPool for Access (Recommended)

Example code:

```
import org.apache.commons.pool2.impl.GenericObjectPoolConfig;
import redis.clients.jedis.Jedis;
import redis.clients.jedis.JedisPool;

public class JedisPoolTests {

  private static void testPool() {
    // There will be security risks if the username and password used for authentication are directly written into code. Store the username and password in ciphertext in the configuration file or environment variables.
    // In this example, the username and password are stored in the environment variables. Before running this example, set environment variables EXAMPLE_USERNAME_ENV and EXAMPLE_PASSWORD_ENV as needed.
    String pwd = System.getenv("EXAMPLE_PASSWORD_ENV");
    JedisPool pool = new JedisPool(new GenericObjectPoolConfig(), "172.xx.xx.xx", 6379,
        2000, pwd);
    Jedis jedis = pool.getResource();
    try {
      System.out.println(jedis.hgetAll("676296"));
      System.out.println(jedis.set("key1", "value1"));
    } finally {
      jedis.close();
    }
    pool.destroy();
  }

  public static void main(String[] args) {
    testPool();
  }
}
```

- In the preceding code, 172.*xx.xx.xx* indicates the load balancer IP address of the GeminiDB Redis instance that you want to connect to.

  You can click the instance name to go to the **Basic Information** page and obtain the load balancer IP address in the **Network Information** area.

  Figure 5-1 Viewing the load balancer IP address

- **6379** in the preceding code is the port of the instance to be connected. Replace it with the actual port number. For details about how to obtain the port number, see **4.3.5.4 Viewing the IP Address and Port Number of a GeminiDB Redis Instance**.
- For details about the supported and restricted commands, see **5.1 Development and O&M Rules**.
- Redis Cluster and GeminiDB Redis API use different hash algorithms. Adding hashtags to keys in some commands of GeminiDB Redis API can avoid unexpected exceptions. For details about how to use hashtags, see **5.1 Development and O&M Rules**.

Using JedisCluster for Access

Example code:

```
import org.apache.commons.pool2.impl.GenericObjectPoolConfig;
import redis.clients.jedis.HostAndPort;
import redis.clients.jedis.JedisCluster;

public class ClusterTests {

  private static void testCluster() {
    String pwd = "a";
        JedisCluster cluster = new JedisCluster(new HostAndPort("172.xx.xx.xx", 6379),
        200, 2000, 5, pwd, new GenericObjectPoolConfig());
    System.out.println(cluster.hgetAll("676296"));
    System.out.println(cluster.set("key1", "value1"));
  }

  public static void main(String[] args) {
    testCluster();
  }
}
```

- In the preceding code, 172.*xx.xx.xx* indicates the load balancer IP address of the GeminiDB Redis instance that you want to connect to.

  You can click the instance name to go to the **Basic Information** page and obtain the load balancer IP address in the **Network Information** area.

  Figure 5-2 Viewing the load balancer IP address

- **6379** in the preceding code is the port of the instance to be connected. Replace it with the actual port number. For details about how to obtain the port number, see **4.3.5.4 Viewing the IP Address and Port Number of a GeminiDB Redis Instance**.
- For details about the supported and restricted commands, see **5.1 Development and O&M Rules**.
- Redis Cluster and GeminiDB Redis API use different hash algorithms. Adding hashtags to keys in some commands of GeminiDB Redis API can avoid

unexpected exceptions. For details about how to use hashtags, see **5.1 Development and O&M Rules**.

Using JedisSentinelPool for Access

Example code:

```
import org.apache.commons.pool2.impl.GenericObjectPoolConfig;
import redis.clients.jedis.Jedis;
import redis.clients.jedis.JedisSentinelPool;
import java.util.HashSet;
import java.util.Set;

public void SentinelTest {
    public static void main(String[] args) {
        GenericObjectPoolConfig<Jedis> config = new GenericObjectPoolConfig<Jedis>();
        Set<String> mySentinels = new HashSet<String>();
        mySentinels.add("172.xx.xx.xx:6379");
        JedisSentinelPool pool = new JedisSentinelPool(master-name, mySentinels, config, 1000,
password, 0);
        Jedis jedis = pool.getResource();
        jedis.auth(password);
        jedis.set("foo", "bar");
        String s = jedis.get("foo");
        System.out.println(s);
        jedis.close();
        pool.close();
    }
}
```

- In the preceding code, 172.*xx.xx.xx* indicates the load balancer IP address of the GeminiDB Redis instance that you want to connect to.

  You can click the instance name to go to the **Basic Information** page and obtain the load balancer IP address in the **Network Information** area.

  Figure 5-3 Viewing the load balancer IP address

- **6379** in the preceding code is the port of the instance to be connected. Replace it with the actual port number. For details about how to obtain the port number, see **4.3.5.4 Viewing the IP Address and Port Number of a GeminiDB Redis Instance**.
- In the preceding code, **master-name** can only be set to **mymaster**.
- For details about the supported and restricted commands, see **5.1 Development and O&M Rules**.
- Redis Cluster and GeminiDB Redis API use different hash algorithms. Adding hashtags to keys in some commands of GeminiDB Redis API can avoid unexpected exceptions. For details about how to use hashtags, see **5.1 Development and O&M Rules**.

5.3.2 Connecting to an Instance Using Redisson

This section describes how to connect to a GeminiDB Redis instance using Redisson in single-node or sentinel mode.

Prerequisites

- A GeminiDB Redis instance has been created and is running properly. For details about how to create a GeminiDB Redis instance, see **4.2.1 Buying a GeminiDB Redis Cluster Instance**.
- An ECS is available. For details, see **Purchasing an ECS**.
- GNU Compiler Collection (GCC) has been installed on the ECS.
- The created ECS is in the same region, AZ, VPC, and security group as the GeminiDB Redis instance.

SingleServer Mode

Example code:

```
import org.redisson.Redisson;
import org.redisson.api.RedissonClient;
import org.redisson.config.Config;

public class SingleServerTests {

  private static void testSingleServer() {
    Config config = new Config();
    // There will be security risks if the username and password used for authentication are directly written into code. Store the username and password in ciphertext in the configuration file or environment variables.
    // In this example, the username and password are stored in the environment variables. Before running this example, set environment variables EXAMPLE_USERNAME_ENV and EXAMPLE_PASSWORD_ENV as needed.
    String password = System.getenv("EXAMPLE_PASSWORD_ENV");
    config.useSingleServer().setAddress("redis://172.xx.xx.xx:6379")
                            .setPassword(password);
    RedissonClient redisson = Redisson.create(config);
    execute(redisson);   // send requests to database
    redisson.shutdown();
  }

  public static void main(String[] args) {
    testSingleServer();
  }
}
```

- In the preceding code, 172.*xx.xx.xx* indicates the load balancer IP address of the GeminiDB Redis instance that you want to connect to.

  You can click the instance name to go to the **Basic Information** page and obtain the load balancer IP address in the **Network Information** area.

Figure 5-4 Viewing the load balancer IP address

- **6379** in the preceding code is the port of the instance to be connected. Replace it with the actual port number. For details about how to obtain the port number, see **4.3.5.4 Viewing the IP Address and Port Number of a GeminiDB Redis Instance**.
- For details about the supported and restricted commands, see **5.1 Development and O&M Rules**.
- Redis Cluster and GeminiDB Redis API use different hash algorithms. Adding hashtags to keys in some commands of GeminiDB Redis API can avoid unexpected exceptions. For details about how to use hashtags, see **5.1 Development and O&M Rules**.

Sentinel Mode

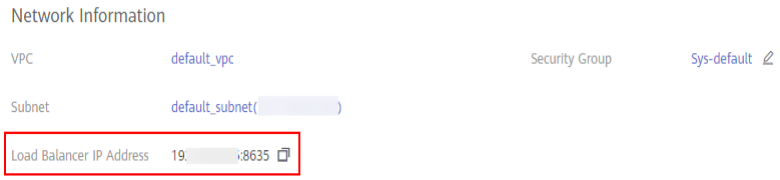
Example code:

```
import org.redisson.Redisson;
import org.redisson.api.RedissonClient;
import org.redisson.config.Config;
import static org.redisson.config.ReadMode.MASTER;

public class SingleServerTests {

  public static void testSentinel() {
    Config config = new Config();
    // There will be security risks if the username and password used for authentication are directly written into code. Store the username and password in ciphertext in the configuration file or environment variables.
    // In this example, the username and password are stored in the environment variables. Before running this example, set environment variables EXAMPLE_USERNAME_ENV and EXAMPLE_PASSWORD_ENV as needed.
    String password = System.getenv("EXAMPLE_PASSWORD_ENV");
    config.useSentinelServers()
            .setMasterName(master_name)
            .setCheckSentinelsList(false)
            .setReadMode(MASTER)
            .setPassword(password)
            .addSentinelAddress("redis://172.xx.xx.xx:6379");
    RedissonClient redisson = Redisson.create(config);
    execute(redisson);      // send requests to database
    redisson.shutdown();
}

  public static void main(String[] args) {
    testSentinel();
  }
}
```

- In the preceding code, 172.*xx.xx.xx* indicates the load balancer IP address of the GeminiDB Redis instance that you want to connect to.

  You can click the instance name to go to the **Basic Information** page and obtain the load balancer IP address in the **Network Information** area.

Figure 5-5 Viewing the load balancer IP address



- **6379** in the preceding code is the port of the instance to be connected. Replace it with the actual port number. For details about how to obtain the port number, see **4.3.5.4 Viewing the IP Address and Port Number of a GeminiDB Redis Instance**.
- The Sentinel mode is only used for connection and its native availability is not used. Thus, in the sample code, **master_name** is fixed to **mymaster**. **CheckSentinelsList** must be set to **false** and **ReadMode** must be set to **MASTER**.
- For details about the supported and restricted commands, see **5.1 Development and O&M Rules**.
- Redis Cluster and GeminiDB Redis API use different hash algorithms. Adding hashtags to keys in some commands of GeminiDB Redis API can avoid unexpected exceptions. For details about how to use hashtags, see **5.1 Development and O&M Rules**.

ClusterServer Mode

Example code:

```
import org.redisson.Redisson;
import org.redisson.api.RedissonClient;
import org.redisson.config.Config;

public class ClusterServerTests {

  private static void testClusterServer() {
    Config config = new Config();
    config.useClusterServers()
              .addNodeAddress("172.xx.xx.xx:6379").setPassword(password);

    RedissonClient redisson = Redisson.create(config);
    execute(redisson);   // send requests to database
    redisson.shutdown();
  }

  public static void main(String[] args) {
    testClusterServer();
  }
}
```

- In the preceding code, 172.*xx.xx.xx* indicates the load balancer IP address of the GeminiDB Redis instance that you want to connect to.

  You can click the instance name to go to the **Basic Information** page and obtain the load balancer IP address in the **Network Information** area.

Figure 5-6 Viewing the load balancer IP address

- **6379** in the preceding code is the port of the instance to be connected. Replace it with the actual port number. For details about how to obtain the port number, see **4.3.5.4 Viewing the IP Address and Port Number of a GeminiDB Redis Instance**.
- For details about the supported and restricted commands, see **5.1 Development and O&M Rules**.
- Redis Cluster and GeminiDB Redis API use different hash algorithms. Adding hashtags to keys in some commands of GeminiDB Redis API can avoid unexpected exceptions. For details about how to use hashtags, see **5.1 Development and O&M Rules**.

5.3.3 Connecting to an Instance Using Hiredis

This section describes how to use hiredis to access a GeminiDB Redis instance.

Prerequisites

- A GeminiDB Redis instance has been created and is running properly. For details about how to create a GeminiDB Redis instance, see **4.2.1 Buying a GeminiDB Redis Cluster Instance**.
- An ECS is available. For details, see **Purchasing an ECS**.
- GNU Compiler Collection (GCC) has been installed on the ECS.
- The created ECS is in the same region, AZ, VPC, and security group as the GeminiDB Redis instance.

Procedure

Step 1 Obtain the load balancer IP address and port of the GeminiDB Redis instance that you want to access.

- For how to obtain the load balancer IP address, see **Viewing the Load Balancer IP Address and Port**.
- For how to obtain the port, see **Viewing the Port for Accessing Each Instance Node**.

Step 2 Log in to the ECS. For details, see **Logging In to an ECS** in *Getting Started with Elastic Cloud Server*.

Step 3 Run the following command to download and decompress the hiredis package.

wget https://github.com/redis/hiredis/archive/master.zip

Step 4 Go to the directory where the decompressed hiredis package is saved, and compile and install hiredis.

make

make install

Step 5 Write the test code connRedisTst.cc.

NOTE

For details about how to use hiredis, see the usage description on the **Redis official website**.

The code is as follows:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <hiredis.h>
int main(int argc, char **argv) {
    unsigned int j;
    redisContext *conn;
    redisReply *reply;
    if (argc < 3) {
            printf("Usage: example {instance_ip_address} 6379 {password}\n");
            exit(0);
    }
    const char *hostname = argv[1];
    const int port = atoi(argv[2]);
    const char *password = argv[3];
    struct timeval timeout = { 1, 500000 }; // 1.5 seconds
    conn = redisConnectWithTimeout(hostname, port, timeout);
    if (conn == NULL || conn->err) {
            if (conn) {
            printf("Connection error: %s\n", conn->errstr);
            redisFree(conn);
            } else {
            printf("Connection error: can't allocate redis context\n");
            }
    exit(1);
    }
    /* AUTH */
    reply = redisCommand(conn, "AUTH %s", password);
    printf("AUTH: %s\n", reply->str);
    freeReplyObject(reply);

    /* Set */
    reply = redisCommand(conn,"SET %s %s", "key", "hiredis test ok!");
    printf("SET: %s\n", reply->str);
    freeReplyObject(reply);

    /* Get */
    reply = redisCommand(conn,"GET key");
    printf("GET key: %s\n", reply->str);
    freeReplyObject(reply);

    /* Disconnects and frees the context */
    redisFree(conn);
    return 0;
}
```

Step 6 Run the following command to perform compilation:

gcc connRedis.c -o connRedis -I /usr/local/include/hiredis -lhiredis

If an error is reported, locate the directory where the **hiredis.h** file is stored and modify the compile command.

After the compilation, an executable **connRedis** file is obtained.

Step 7 Run the following command to connect to the instance.

export LD_LIBRARY_PATH=/usr/local/lib/:$LD_LIBRARY_PATH

./connRedis <redis_ip_address> 6379 <password>

Replace the following information based on the site requirements:

- *<redis_ip_address>* indicates the load balancer IP address obtained in **Step 1**.
- **6379** is the port number of the GeminiDB Redis instance.
- *<password>* indicates the password set when the instance is created.

Step 8 If the following information is displayed, the instance is successfully connected.

```
AUTH: OK
SET: OK
GET key: Hello, hiredis test ok!
```

----End

5.3.4 Connecting to an Instance Using Node.js

This section describes how to use Node.js to access a GeminiDB Redis instance.

Prerequisites

- A GeminiDB Redis instance has been created and is in the **Available** status.
- An ECS is available. For details, see **Purchasing an ECS**.
- If the Linux operating system is used, ensure that compilation tools such as GCC have been installed on the ECS.
- The created ECS is in the same region, AZ, VPC, and security group as the GeminiDB Redis instance.

Procedure

Step 1 Obtain the load balancer IP address and port of the GeminiDB Redis instance that you want to access.

- For how to obtain the load balancer IP address, see **Viewing the Load Balancer IP Address and Port**.
- For how to obtain the port, see **Viewing the Port for Accessing Each Instance Node**.

Step 2 Log in to the ECS. For details, see **Logging In to an ECS** in *Getting Started with Elastic Cloud Server*.

Step 3 Run the following command to install Node.js:

- Method 1: Run the following command to install Node.js:

  yum install nodejs

  NOTE

  CentOS (Red Hat series) is used as an example. If Ubuntu (Debian series) is used, run the corresponding installation command.

- Method 2: If the method 1 fails, use the following method to install it.

  wget https://nodejs.org/dist/v0.12.4/node-v0.12.4.tar.gz --no-check-certificate ;

  tar -xvf node-v0.12.4.tar.gz;

  cd node-v0.12.4;

  ./configure;

  make;

  make install;

  NOTE

  CentOS (Red Hat series) is used as an example. If Ubuntu (Debian series) is used, run the corresponding installation command.

Step 4 After the Node.js is installed, run the following command to check the version number and ensure that the Node.js is successfully installed.

node -v

Step 5 Install the JS package management tool npm.

yum install npm

Step 6 Install the Node.js Redis client ioredis.

npm install ioredis

Step 7 Edit the sample script for connecting to the instance.

- Connect to a GeminiDB Redis cluster using the SDK for the single-node API on Node.js.

```
var Redis = require('ioredis');
// There will be security risks if the username and password used for authentication are directly written into code. Store the username and password in ciphertext in the configuration file or environment variables.
// In this example, the username and password are stored in the environment variables. Before running this example, set environment variables EXAMPLE_USERNAME_ENV and EXAMPLE_PASSWORD_ENV as needed.
var pwd = process.env.EXAMPLE_PASSWORD_ENV;
var redis = new Redis({
  port: 6379,          //Port number of the GeminiDB Redis instance obtained in step 1
  host: '192.xx.xx.xx',   //Enter the load balancer IP address obtained in step 1.
  family: 4,           //4 indicates IPv4, and the 6 indicates IPv6.
  password: pwd,
  db: 0
});
redis.set('key', 'Nodejs tst ok!');
redis.get('key', function (err, result) {
  console.log(result);
});
```

- Connect to a GeminiDB Redis cluster using the SDK of the cluster API on Node.js.

```
const Redis = require("ioredis");
// There will be security risks if the username and password used for authentication are directly written into code. Store the username and password in ciphertext in the configuration file or environment variables.
// In this example, the username and password are stored in the environment variables. Before running this example, set environment variables EXAMPLE_USERNAME_ENV and EXAMPLE_PASSWORD_ENV as needed.
```



```
var pwd = process.env.EXAMPLE_PASSWORD_ENV;
const cluster = new Redis.Cluster([
  {
    port: 6379,          //Port number of the GeminiDB Redis instance obtained in step 1
 host: '192.xx.xx.xx',   //Enter the load balancer IP address obtained in step 1.
    family: 4,           // 4 indicates IPv4, and the 6 indicates IPv6.
    password: pwd,
    db: 0
  },
]);

cluster.set("foo", "nodejs is ok!");
cluster.get("foo", (err, res) => {
  console.log(res);
});
```

Step 8 Run the sample script and verify that the result is normal.

node ioredisdemo.js

----End

5.3.5 Connecting to an Instance Using PHP

This section describes how to use PHP to access a GeminiDB Redis instance.

Prerequisites

- A GeminiDB Redis instance has been created and is in the **Available** status.
- An ECS is available. For details, see **Purchasing an ECS**.
- GNU Compiler Collection (GCC) has been installed on the ECS.
- The created ECS is in the same region, AZ, VPC, and security group as the GeminiDB Redis instance.

Procedure

Step 1 Obtain the load balancer IP address and port of the GeminiDB Redis instance that you want to access.

- For how to obtain the load balancer IP address, see **Viewing the Load Balancer IP Address and Port**.
- For how to obtain the port, see **Viewing the Port for Accessing Each Instance Node**.

Step 2 Log in to the ECS. For details, see **Logging In to an ECS** in *Getting Started with Elastic Cloud Server*.

Step 3 Install the PHP development kit and command line tool.

Run the following yum command to install the PHP development package:

yum install php-devel php-common php-cli

NOTE

CentOS (Red Hat series) is used as an example. If Ubuntu (Debian series) is used, run the corresponding installation command.

Step 4 After the installation is complete, check the version number to ensure that the installation is successful.

php --version

Step 5 Install the PHP client of Redis.

1. Run the following command to download the source phpredis package:

   wget http://pecl.php.net/get/redis-4.1.0RC3.tgz

   NOTE

   The preceding clients are of the latest version. You can download the phpredis client of other versions from the **PHP official website**.

2. Run the following commands to decompress the source phpredis package:

   tar -zxvf redis-4.1.0RC3.tgz

   cd redis-4.1.0RC3

3. Run the following extension command before compilation:

   phpize

4. Run the following command to configure the php-config file:

   ./configure --with-php-config=/usr/bin/php-config

   NOTE

   The PHP installation method and location depend on the operating system. Before the configuration, run the **find / -name php.ini** command to check the directory of the file.

5. Run the following command to compile and install the phpredis client:

   make && make install

6. After the installation, add the extension configuration in the **php.ini** file to reference the Redis module.

   Run the following command to find the **php.ini** file:

   vim /usr/local/php/etc/php.ini

   Add the following configuration item to the **php.ini** file:

   ```
   extension = "/usr/lib64/php/modules/redis.so"
   ```

   NOTE

   The directories of the **php.ini** and **redis.so** files may be different. You can run the following command to query the directories.

   find / -name php.ini

   find / -name redis.so

7. Save the configuration and exit. Then, run the following command to check whether the extension takes effect:

   php -m |grep redis

   If redis is returned, the PHP Redis client environment has been set up.

Step 6 Use the phpredis client to connect to the instance.

1. Compile the test code redis.php.
   - Connect to a GeminiDB Redis cluster using the SDK of the single-node API on PHP.


```
<?php
   // There will be security risks if the username and password used for
authentication are directly written into code. Store the username and password in
ciphertext in the configuration file or environment variables.
   // In this example, the username and password are stored in the environment
variables. Before running this example, set environment variables
EXAMPLE_USERNAME_ENV and EXAMPLE_PASSWORD_ENV as needed.
   $pwd =getenv('EXAMPLE_PASSWORD_ENV');
   $redis_host = "192.xx.xx.xx";  //Enter the load balancer IP address obtained in step
1.
   $redis_port = 6379;
   $user_pwd = pwd;
   $redis = new Redis();
   if ($redis->connect($redis_host, $redis_port) == false) {
      die($redis->getLastError());
   }
   if ($redis->auth($user_pwd) == false) {
       die($redis->getLastError());
   }
   if ($redis->set("key", "php test ok!") == false) {
       die($redis->getLastError());
   }
   $value = $redis->get("key");
   echo $value;
   $redis->close();
?>
```

- Connect to the GeminiDB Redis cluster using the SDK of the cluster API on PHP.

```
<?php
       $redis_host = "192.xx.xx.xx";  //Enter the load balancer IP address obtained in
step 1.
       $redis_port = 6379;
       $user_pwd = "pwd";
       // Connect with read/write timeout as well as specify that phpredis should use
       // persistent connections to each node.
       $redis = new RedisCluster(NULL, Array("$redis_host:$redis_port"), 1.5, 1.5, true,
$user_pwd);
       if ($redis->set("key", "php test ok!") == false) {
           die($redis->getLastError());
       }
       $value = $redis->get("key");
       echo $value;
       $redis->close();
   ?>
```

2. Run the redis.php command to check whether the result is normal.

----End

5.3.6 Connecting to an Instance Using Python

This section describes how to use Python to access a GeminiDB Redis instance.

Prerequisites

- A GeminiDB Redis instance has been created and is in the **Available** status.
- An ECS is available. For details, see **Purchasing an ECS**.
- GNU Compiler Collection (GCC) has been installed on the ECS.
- The created ECS is in the same region, AZ, VPC, and security group as the GeminiDB Redis instance.

Procedure

Step 1 Obtain the load balancer IP address and port of the GeminiDB Redis instance that you want to access.

- For how to obtain the load balancer IP address, see **Viewing the Load Balancer IP Address and Port**.
- For how to obtain the port, see **Viewing the Port for Accessing Each Instance Node**.

Step 2 Log in to the ECS. For details, see **Logging In to an ECS** in *Getting Started with Elastic Cloud Server*.

Step 3 Install the Python client Redis-py of Python and Redis.

1. If the system does not provide Python, you can use yum to install it.

   yum install python

2. Run the following command to download and decompress the redis-py package:

   wget https://github.com/andymccurdy/redis-py/archive/master.zip

3. Go to the decompression directory and install the Python client Redis-py of Redis.

   python setup.py install

4. After the installation, run the **python** command. If the following information is displayed, Redis-py is successfully installed:

```
Python 2.6.6 (r266:84292, Aug 18 2016, 15:13:37)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-17)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import redis
>>>
```

Step 4 Use the Redis-py client to connect to the instance.

In the following steps, commands are executed in CLI mode. (Alternatively, write the commands into a Python script and then execute the script.)

- **Connect to the GeminiDB Redis cluster using the single-node API.**

  a. Run the **python** command to enter the CLI mode.

  You have entered CLI mode if the following command output is displayed:

```
Python 2.6.6 (r266:84292, Aug 18 2016, 15:13:37)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-17)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import redis
>>>
```

  b. Run the following command in the CLI to check whether the result is normal.

```
>>> r = redis.StrictRedis(host='192.xx.xx.xx', port=6379, password='pwd');
>>> r.set('key', 'Python tst ok!')
True
>>> r.get('key')
'Python tst ok!'
```


NOTE

Modify the following information based on service requirements before running the preceding command.

- In the preceding command, **host** and **port** indicate the load balancer IP address and corresponding port of the GeminiDB Redis instance obtained in **Step 1**.
- **password** indicates the password of the instance.

- **Connect to the GeminiDB Redis cluster using the cluster API.**

  Configure **config set CompatibleMode ClusterClient** first.

```
Python 3.7.4 (default, Jan 30 2021, 09:00:44)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from redis.cluster import RedisCluster as Redis
>>> rc = Redis(host='127.0.0.1', port=6379, password='a')
>>> rc.set('key', 'Python test ok!')
True
>>> rc.get('key')
b'Python test ok!'
```

----End

5.3.7 Connecting to an Instance Using Go

This section describes how to use Go to access a GeminiDB Redis instance.

Prerequisites

- A GeminiDB Redis instance has been created and is in the **Available** status.
- An ECS is available. For details, see **Purchasing an ECS**.
- GNU Compiler Collection (GCC) has been installed on the ECS.
- The created ECS is in the same region, AZ, VPC, and security group as the GeminiDB Redis instance.

Procedure

Step 1 Obtain the load balancer IP address and port of the GeminiDB Redis instance that you want to access.

- For how to obtain the load balancer IP address, see **Viewing the Load Balancer IP Address and Port**.
- For how to obtain the port, see **Viewing the Port for Accessing Each Instance Node**.

Step 2 Log in to the ECS. For details, see **Logging In to an ECS** in *Getting Started with Elastic Cloud Server*.

Step 3 Use the Golang client to connect to the instance. The following uses the go-redis SDK as an example.

- Go-redis download address: **https://github.com/go-redis/redis**
- **Connect to the GeminiDB Redis cluster using the single-node API.**

```
package main
import (
    "fmt"
```



```
    "github.com/go-redis/redis"
    "os"
)
func main() {
    // There will be security risks if the username and password used for authentication are directly written into code. Store the username and password in ciphertext in the configuration file or environment variables.
    // In this example, the username and password are stored in the environment variables. Before running this example, set environment variables EXAMPLE_USERNAME_ENV and EXAMPLE_PASSWORD_ENV as needed.
    password = os.Getenv("EXAMPLE_PASSWORD_ENV")
    client := redis.NewClient(&redis.Options{
        Addr:     "xx.xx.xx.xx:6379",  //Enter the load balancer IP address obtained in step 1.
        Password: password,
        DB:       0,  // Use the default database 0.
    })
    pong, err := client.Ping().Result()
    fmt.Println(pong, err)
    err = client.Set("key1", "value1", 0).Err()
    if err != nil {
        panic(err)
    }
    val, err := client.Get("key1").Result()
    if err != nil {
        panic(err)
    }
    fmt.Println("key1", val)
}
```

The expected output is as follows:

```
PONG
key1 value1
```

NOTE

- If you use go-redis to connect a GeminiDB Redis instance, use the common mode instead of the cluster mode, as shown in the preceding sample code.
- In the preceding example, set the GeminiDB Redis address and password based on the site requirements.

- **Connect to the GeminiDB Redis cluster using the cluster API.**

```
package main
import (
    "fmt"
    "github.com/go-redis/redis"
)
func main() {
client := redis.NewClusterClient(&redis.ClusterOptions{
        Addrs: []string{ // Enter the load balancer IP address obtained in step 1.
            " xx.xx.xx.xx:6379",
        },
        Password:     "xx", // Password of the cluster.
    })
    pong, err := client.Ping().Result()
    fmt.Println(pong, err)

    err = client.Set("key1", "value1", 0).Err()
    if err != nil {
        panic(err)
    }
    val, err := client.Get("key1").Result()
    if err != nil {
        panic(err)
```



```
    }
    fmt.Println("key1", val)
}
```

----End

5.3.8 Connecting to an Instance Using C#

This section describes how to use C# to access a GeminiDB Redis instance.

Prerequisites

- A GeminiDB Redis instance has been created and is in the **Available** status.
- An ECS is available. For details, see **Purchasing an ECS**.
- GNU Compiler Collection (GCC) has been installed on the ECS.
- The created ECS is in the same region, AZ, VPC, and security group as the GeminiDB Redis instance.

Procedure

Step 1 Obtain the load balancer IP address and port of the GeminiDB Redis instance that you want to access.

- For how to obtain the load balancer IP address, see **Viewing the Load Balancer IP Address and Port**.
- For how to obtain the port, see **Viewing the Port for Accessing Each Instance Node**.
- For details about how to view the IP address of each instance, see **Viewing the Private IP Address or EIP**.

Step 2 Log in to the ECS. For details, see **Logging In to an ECS** in *Getting Started with Elastic Cloud Server*.

Step 3 Install .Net. For a Windows host, click **here** to download .NET. For a Linux host, you need to install .NET Core key and repository, and then install the .NET runtime and SDK.

```
sudo rpm -Uvh https://packages.microsoft.com/config/centos/8/packages-microsoft-prod.rpm
sudo yum install dotnet-sdk-7.0
sudo yum install dotnet-runtime-7.0
```

Run the following code.

```
dotnet --version
```

You'll see your .Net version information.

Step 4 Use the StackExchange.Redis client to connect to the GeminiDB Redis instance.

- Creating a project

  Run the following command to create a C# console application or create a new C# console application in Visual Studio.

  ```
  dotnet new console -o redisdemo
  ```

- Installing the **StackExchange.Redis** package of the C# client of the Redis. In Visual Studio, you can install StackExchange.Redis from the NuGet package manager. Run the following command in the command line window where the dotnet project is located:


```
dotnet add package StackExchange.Redis
```

- Connecting to GeminiDB Redis in single-node mode

```
using System;
using StackExchange.Redis;
namespace redisdemo
{
    class Program
    {
        static void  Main(string[] args)
        {
            //Create a ConnectionMultiplexer object connected to the Redis server.
            string redisConnectionString = "192.xx.xx.xx:6379"; // Load balancer address obtained in step 1
            ConfigurationOptions options = ConfigurationOptions.Parse(redisConnectionString);
             // There will be security risks if the username and password used for authentication are directly written into code. Store the username and password in ciphertext in the configuration file or environment variables.
              // In this example, the username and password are stored in the environment variables. Before running this example, set environment variables EXAMPLE_USERNAME_ENV and EXAMPLE_PASSWORD_ENV as needed.
            string password = Environment.GetEnvironmentVariable("EXAMPLE_PASSWORD_ENV");
            options.Password = password;
            ConnectionMultiplexer redis = ConnectionMultiplexer.Connect(options);

            //Obtain the Redis database object.
            IDatabase redisDb = redis.GetDatabase();

            //Set a key-value pair.
            string key = "mykey";
            string value = "myvalue";
            redisDb.StringSet(key, value);
            string valueGet = redisDb.StringGet(key);
            Console.WriteLine ($"The value of {key}: {valueGet}");
        }
    }
}
```

Expected output:

```
The value of mykey is myvalue.
```

- Connecting to the GeminiDB Redis cluster in cluster mode

```
using System;
using StackExchange.Redis;

namespace redisdemo
{
    class Program
    {
        static void  Main(string[] args)
        {
              ConfigurationOptions options = new ConfigurationOptions();
              options.EndPoints.Add("192.xx.xx.xx:6379"); // IP address and port number of node 1 in the instance cluster obtained in Step 1
              options.EndPoints.Add("192.xx.xx.xx:6379"); // IP address and port number of node 2 in the instance cluster obtained in Step 1
              options.Password = "your_password"; // Set the password.
              ConnectionMultiplexer redis = ConnectionMultiplexer.Connect(options);
            //Obtain the Redis database object.
              IDatabase redisDb = redis.GetDatabase();
            //Set a key-value pair.
```



```
            string key = "mykey";
            string value = "myvalue";
            redisDb.StringSet(key, value);
            string valueGet = redisDb.StringGet(key);
        Console.WriteLine ($"The value of {key}: {valueGet}");
      }
    }
}
```

Expected output:

```
The value of mykey is myvalue.
```

----End

5.3.9 Connecting to an Instance Using Sentinel

GeminiDB Redis API uses an in-house HA component and does not depend on Sentinel. To reduce code modifications and improve instance compatibility, GeminiDB Redis API is compatible with Redis Sentinel. After the Sentinel mode is enabled, you can connect to a GeminiDB Redis instance to Redis Sentinel.

Prerequisites

- A cluster or primary/standby GeminiDB Redis instance has been created and is in the **Available** state.
- An ECS is available. For details, see **Purchasing an ECS**.
- GNU Compiler Collection (GCC) has been installed on the ECS.
- The created ECS is in the same region, AZ, VPC, and security group as the GeminiDB Redis instance.
- To connect to a DB instance in Sentinel mode, you must enable the Sentinel compatibility mode first.

Enabling the Sentinel Compatibility Mode

Step 1 **Log in to the Huawei Cloud console.**

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

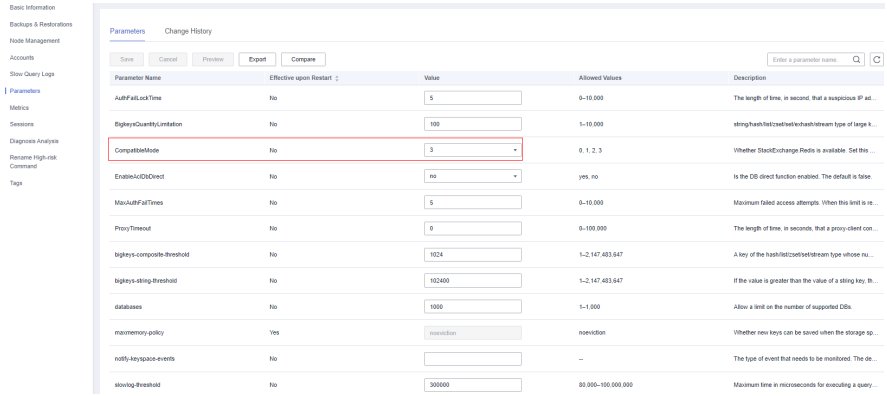
Step 3 On the **Instances** page, click the instance whose specifications you want to change. The **Basic Information** page is displayed.

Step 4 In the navigation pane on the left, choose **Parameters**.

Step 5 Change the value of **CompatibleMode** and click **Save**.

- For a cluster DB instance, change the value of **CompatibleMode** to **3**.
- For a primary/standby DB instance, change the value of **CompatibleMode** to **2**.

Figure 5-7 Changing parameters



----End

Connecting to a DB Instance in Sentinel Mode

This section uses Java as an example to describe how to access a GeminiDB Redis instance through the open-source libraries Redisson and Jedis.

Redisson code example:

```
import org.redisson.Redisson;
import org.redisson.api.RedissonClient;
import org.redisson.config.Config;
import static org.redisson.config.ReadMode.MASTER;

public class SingleServerTests {

  public static void testSentinel() {
    Config config = new Config();
    // There will be security risks if the username and password used for authentication are directly written into code. Store the username and password in ciphertext in the configuration file or environment variables.
    // In this example, the username and password are stored in the environment variables. Before running this example, set environment variables EXAMPLE_USERNAME_ENV and EXAMPLE_PASSWORD_ENV as needed.
    String password = System.getenv("EXAMPLE_PASSWORD_ENV");
    config.useSentinelServers()
            .setMasterName("mymaster")
            .setCheckSentinelsList(false)
            .setReadMode(ReadMode.MASTER)
            .setPassword(password)
            .addSentinelAddress("redis://172.xx.xx.xx:6379");
    RedissonClient redisson = Redisson.create(config);
    execute(redisson);      // send requests to database
    redisson.shutdown();
}

  public static void main(String[] args) {
    testSentinel();
  }
}
```

Jedis code example:

```
import java.util.HashSet;
import java.util.Set;
import org.apache.commons.pool2.impl.GenericObjectPoolConfig;
import redis.clients.jedis.Jedis;
import redis.clients.jedis.JedisSentinelPool;
public class TestJedisSentinel {
```

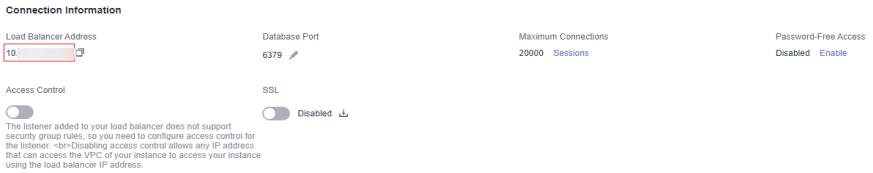


```
    public static void main(String[] args) {
        Set<String> sentinels = new HashSet<>();
        sentinels.add("192.xx.xx.xx:6379");
        GenericObjectPoolConfig<Jedis> poolConfig = new GenericObjectPoolConfig<>();
        poolConfig.setMaxIdle(100);
        poolConfig.setMaxWaitMillis(10000);
        poolConfig.setTestOnBorrow(true);
        int connectionTimeout = 5000;
        int soTimeout = 5000;
        int database = 0;
            // There will be security risks if the username and password used for authentication are directly written into code. Store the username and password in ciphertext in the configuration file or environment variables.
    // In this example, the username and password are stored in the environment variables. Before running this example, set environment variables EXAMPLE_USERNAME_ENV and EXAMPLE_PASSWORD_ENV as needed.
            String password = System.getenv("EXAMPLE_PASSWORD_ENV");
        try (JedisSentinelPool jspool = new JedisSentinelPool("mymaster", sentinels, poolConfig,
                connectionTimeout, soTimeout, password, database)) {
            Jedis jedis = jspool.getResource();
            jedis.mset("testkey", "AAA", "b", "BBB");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

- MasterName: fixed character string **mymaster**.
- CheckSentinelsList: The value must be **false**.
- ReadMode: **ReadMode.MASTER** is used.
- Password: password of the instance
- SentinelAddress: load balancer address of the GeminiDB Redis instance. Replace it with the actual IP address and port number.

  You can click the instance name to go to the **Basic Information** page and obtain the load balancer IP address in the **Connection Information** area.

  Figure 5-8 Viewing the load balancer IP address

The Sentinel mode is only used for connection and its native availability is not used. Thus, in the sample code, **master_name** is fixed to **mymaster**. **CheckSentinelsList** must be set to **false** and **ReadMode** must be set to **MASTER**.

5.4 Lua Script Compilation Specifications

Lua is a scripting language designed to be embedded in applications to provide flexible extension and customization functions for applications. GeminiDB Redis API uses Lua 5.1.5, which is the same as the Lua version used by the open-source Redis 5.0.

NOTE

When you use Lua scripts, make sure to perform a careful verification. Otherwise, infinite loops, request timeouts, or other exceptions may occur, or even services may become unavailable.

Differences from Open-Source Redis Lua

1. **EVAL/EVALSHA**

   Example command:

   EVAL script numkeys key [key ...] arg [arg ...]

   EVALSHA sha1 numkeys key [key ...] arg [arg ...]

   You can use the preceding commands the same as you do in open-source Redis. Ensure that the Redis key used in the script is explicitly transferred through the key instead of being directly encoded in the script.

   If a cluster instance is used and multiple key parameters are specified, all key parameters must have the same hash tag.

   If the preceding constraints are not complied with, error messages may be returned and data consistency may be damaged when Redis operations involving these keys are performed in Lua.

2. **SCRIPT**

   SCRIPT contains a group of subcommands for managing Lua scripts. You can run **SCRIPT HELP** to query specific operations.

   Most SCRIPT commands are compatible with open-source Redis. The following commands that need to be noted:

   - **SCRIPT KILL**

     GeminiDB Redis API is a multi-thread execution environment, so it allows multiple Lua scripts to be executed at the same time. If **SCRIPT KILL** is executed, all running Lua scripts will be terminated.

     For ease of use, it extends the **SCRIPT KILL** command. You can use **SCRIPT KILL SHA1** to terminate the script of a specified hash value. If multiple nodes are executing scripts with the same hash value at the same time, these scripts will be terminated.

     In addition, the Lua timeout period (config set lua-time-limit) cannot be configured. You can run **SCRIPT KILL** at any time to terminate the script, instead of waiting for the script to time out.

   - **SCRIPT DEBUG**

     Currently, the DEBUG command is not supported.

   - **SCRIPT GET**

     This command is added for querying scripts saved to a database with the SCRIPT LOAD command.

     The syntax is SCRIPT GET SHA1.

3. **Run Redis commands in Lua scripts.**

   Similar to the open-source Redis, the Lua environment of GeminiDB Redis also provides a global Redis table to provide various functions for interacting with Redis Server.

   Table 5-5 shows the operations supported and not supported by GeminiDB Redis.

Table 5-5 Function list

Supported Operation	Unsupported Operation
• redis.call() • redis.pcall() • redis.sha1hex() • redis.error_reply() • redis.status_reply()	• redis.log() • redis.LOG_DEBUG • redis.LOG_VERBOSE • redis.LOG_NOTICE • redis.LOG_WARNING • redis.replicate_commands() • redis.set_repl() • redis.REPL_NONE • redis.REPL_AOF • redis.REPL_SLAVE • redis.REPL_REPLICA • redis.REPL_ALL • redis.breakpoint() • redis.debug()

4. **Lua execution environment restrictions**

   The open-source Redis has restrictions on the execution of Lua scripts, for example, restrictions on global variables, random function results, and system libraries and third-party libraries that can be used.

   GeminiDB Redis inherits most restrictions of the open-source Redis, but there are some differences in the following scenarios.

   – **Write Dirty**

     According to the open-source Redis specifications, if a write operation has been executed by a script, the script cannot be terminated by SCRIPT KILL. You must run SHUTDOWN NOSAVE to directly stop Redis Server.

     GeminiDB Redis does not support the SHUTDOWN command, so you can still run **SCRIPT KILL** to stop the script execution.

   – **Random Dirty**

     Due to the master/slave replication, the open-source Redis stipulates that if a script executes a command (Time or randomkey) to get a random key, the script cannot execute the command for writing semantics.

     The following Lua script is used as an example.

     ```
     local t = redis.call("time")
     return redis.call("set", "time", t[1]);
     ```

     When the execution of the script is transferred to the slave node, the time obtained by the **Time** command must be later than that obtained by the master node. Therefore, the value of the **Set** command executed on the slave node conflicts with that on the master node. The open-source Redis introduces replicate_commands to allow users to determine the behavior mode in this scenario.

For GeminiDB Redis instances, there is no primary/standby relationship, and there is only one copy of data logically, so it is not limited by this restriction.

Forbidden Commands in the Lua Script

Hash commands: HSCAN.

List commands: BLPOP, BRPOP, and BRPOPLPUSH.

Set commands: SSCAN.

Sorted set commands: BZPOPMAX, BZPOPMIN, and ZSCAN.

Stream commands: XREAD and XREADGROUP.

Generic commands: RENAME, RENAMENX, RESTORE, SCAN, CLIENT, COMMAND, CONFIG, DBSIZE, FLUSHALL, FLUSHDB, INFO, and KEYS.

Lua commands: EVAL, EVALSHA, and SCRIPT.

Pub/sub commands: PSUBSCRIBE, PUBLISH, PUNSUBSCRIBE, SUBSCRIBE, and UNSUBSCRIBE.

Transactions commands: DISCARD, EXEC, MULTI, UNWATCH, and WATCH.

5.5 Keyspace Notification

The keyspace notification function is available on all clients that support subscription and release, without any modification.

Precautions

- The keyspace notification function is disabled by default because it consumes CPU resources if enabled.
- Do not enable the keyspace notification in high-pressure scenarios. Enabling this function will affect instance performance, and some event notifications may be missed.

Differences from Open-source Redis

1. Configuration methods

   Run CONFIG SET to enable or disable the keyspace notification.

   Config set notify-keyspace-events Ex

   - The keyspace notification is disabled if **notify-keyspace-events** is empty or does not contain **K** and **E**.

     NOTE

     - Double quotation marks indicate that the parameter is an empty string.
     - The GeminiDB Redis console client does not allow you to disable the keyspace notification by setting **notify-keyspace-events** to an empty string.

   - The keyspace notification is enabled if **notify-keyspace-events** is not empty and is correctly configured. For details, see **Table 5-6**.

2. Notification Types

Table 5-6 Supported notification types

Character	Notification	Supported by GeminiDB Redis
K	Keyspace notification. All notifications are prefixed by __**keyspace@**__.	Yes
E	Key event notification. All notifications are prefixed by __**keyevent@<db>**__ .	Yes
g	Notifications for generic commands such as DEL, EXPIRE, and RENAME	Yes
$	Notifications for string commands	Yes
l	Notifications for list commands	Yes
s	Notifications for set commands	Yes
h	Notifications for hash commands	Yes
z	Notifications for sorted set commands	Yes
x	EXPIRED event notifications	Yes
e	EVICT event notifications	N/A
A	Alias for parameter **g$lshzxe**	Yes

As shown in **Table 5-6**, EVICT event notifications are not applicable for the current version of GeminiDB Redis.

The parameter value must contain either **K** or **E**. Otherwise, no notifications are issued.

For example, if you want to subscribe only list-related keyspace notifications, set **notify-keyspace-events** to **Kl**.

The value **AKE** means all types of notifications are issued.

5.6 EXHASH Commands

EXHASH, a hash data type, allows you to specify expiration times and version numbers for fields. EXHASH is more flexible than HASH and can help you simplify service development in various scenarios.

Highlights

- The expiration time and version number can be specified for each field.
- Efficient, flexible active and passive expiration strategies are supported for fields.
- The syntax is similar to that native Redis hashes use.

Commands

Table 5-7 EXHASH commands

Command	Syntax	Description
EXHSET	EXHSET key field value [EX time] [EXAT time] [PX time] [PXAT time] [NX | XX] [VER | ABS | GT version] [KEEPTTL]	Adds a field to an EXHASH key. If there are no keys available, one will be created automatically. If the field has a value, this command overwrites the value.
EXHGET	EXHGET key field	Obtains the value of a field from an EXHASH key. If the key or field does not exist, **nil** is returned.
EXHPTTL	EXHPTTL key field	Queries the remaining expiration time of a field in an EXHASH key. Unit: milliseconds.
EXHTTL	EXHTTL key field	Queries the expiration time of a field in an EXHASH key. Unit: seconds.
EXHVER	EXHVER key field	Queries the current version of a field in an EXHASH key.
EXHINCRBY	EXHINCRBY key field num [EX time] [EXAT time] [PX time] [PXAT time] [VER | ABS | GT version] [MIN minval] [MAX maxval] [KEEPTTL]	Adds **num** to the value of a field in an EXHASH key. **num** is an integer. If no keys are found, a key will be created. If the field does not exist, this command adds the field and sets the value of the field to **0** before increasing the value of the field. **NOTE** If you do not specify an expiration time for a field when running this command, the field will never expire.

Command	Syntax	Description
EXHINCRBYFLOAT	EXHINCRBYFLOAT key field num [EX time] [EXAT time] [PX time] [PXAT time] [VER | ABS | GT version] [MIN minval] [MAX maxval] [KEEPTTL]	Adds **num** to the value of a field in an EXHASH key. **num** is a floating point number. If no keys are found, a key will be created. If the field does not exist, this command adds the field and sets the value of the field to **0** before increasing the value of the field. **NOTE** If you do not specify an expiration time for a field when running this command, the field will never expire.
EXHMGET	EXHMGET key field [field ...]	Obtains multiple field values from an EXHASH key. If the key or field does not exist, **nil** is returned.
EXHLEN	EXHLEN key [NOEXP]	Obtains the number of fields in an EXHASH key. The returned results may include the number of expired fields that are not deleted, because this command does not delete or filter out expired fields. To obtain only the number of fields that have not expired, you can set parameter **NOEXP** in the command.
EXHGETALL	EXHGETALL key	Obtains all fields from an EXHASH key and their values.
EXHDEL	EXHDEL key field [field ...]	Deletes a field from an EXHASH key. If the key or field does not exist, **0** is returned. If the field is deleted, **1** is returned.
DEL	DEL <key> [key ...]	Deletes one or more EXHASH keys.
EXISTS	EXISTS <key> [key ...]	Checks whether there is one or more EXHASH data records.

Complex Commands and Options

- EXHSET

Table 5-8 EXHSET commands

Item	Description
Syntax	EXHSET key field value [EX time] [EXAT time] [PX time] [PXAT time] [NX | XX] [VER | GT | ABS version] [KEEPTTL]
Description	Adds a field to an EXHASH key. If no keys exist, a key will be created. If the field has a value, this command overwrites the value. To add a field that does not expire, you can run this command to add the field without specifying an expiration time.
Parameters	**Key**: Object that you want to manage by running this command.
	field: An element in the EXHASH command. An EXHASH key can have multiple fields.
	value: Value of the field. A field can have only one value.
	EX: Relative expiration time of the field, in seconds. **0** indicates that the field will expire immediately. If this parameter is not specified, the field does not expire.
	EXAT: Absolute expiration time of the field, in seconds. **0** indicates that the field will expire immediately. If this parameter is not specified, the field does not expire.
	PX: Relative expiration time of the field, in milliseconds. **0** indicates that the field will expire immediately. If this parameter is not specified, the field does not expire.
	PXAT: Absolute expiration time of the field, in milliseconds. **0** indicates that the field will expire immediately. If this parameter is not specified, the field does not expire.
	NX: This parameter is added only when the field does not exist.
	XX: This parameter is inserted only if the field exists.
	VER: Version number of the field. If the field exists, the version number specified by this parameter is compared with the current version number. If they are the same, the system continues to run this command and increases the version number by 1. If they are different, an error message will be returned. If the field does not exist or the current version of the field is 0, the system ignores this parameter and runs the command. After the operation completes, the version number changes to 1.
	GT: Version later than the current one. If the set version number is earlier than the current one, a failure message is returned.

Item	Description
	ABS: Absolute version number of the field. When a field is inserted, the version number is the version number specified by this parameter.
	KEEPTTL: The current expiration setting of the field is retained if none of the EX, EXAT, PX, and PXAT parameters are specified.
Returned values	If a field is created and a value is set for it, **1** is returned.
	If a field already exists and the specified value overwrites the current value, **0** is returned.
	If **XX** is specified and the field does not exist, **-1** is returned.
	If **NX** is specified and the field exists, **-1** is returned.
	If **VER** is specified and the value does not match the current version, the error message "ERR update version is stale" is returned.
	Error messages are returned in other cases.

- Example

  Setting the expiration time for a field

```
127.0.0.1:6579> EXHSET k1 f1 v1 ex 10
(integer) 1
127.0.0.1:6579> EXHGET k1 f1
"v1"
127.0.0.1:6579> EXHSET k1 f2 v2 ex 10
(integer) 1
127.0.0.1:6579> EXHGET k1 f1
(nil)
127.0.0.1:6579> EXHGETALL k1
127.0.0.1:6579> EXHGETALL k1
(empty array)
```

  Setting a version number for a field

```
127.0.0.1:6579> EXHSET k1 f1 v1
(integer) 1
127.0.0.1:6579> EXHVER k1 f1
(integer) 1
127.0.0.1:6579> EXHSET k1 f1 v1 ver 2
(error) ERR update version is stale
127.0.0.1:6579> EXHSET k1 f1 v1 ver 1
(integer) 0
127.0.0.1:6579> EXHVER k1 f1
(integer) 2
127.0.0.1:6579> EXHSET k1 f1 v1
(integer) 0
127.0.0.1:6579> EXHVER k1 f1
(integer) 3
127.0.0.1:6579> EXHSET k1 f1 v1 GT 3
(error) ERR update version is stale
127.0.0.1:6579> EXHSET k1 f1 v1 GT 2
(error) ERR update version is stale
127.0.0.1:6579> EXHSET k1 f1 v1 GT 4
(integer) 0
127.0.0.1:6579> EXHVER k1 f1
```



```
(integer) 4
127.0.0.1:6579> EXHSET k1 f1 v1 abs 2
(integer) 0
127.0.0.1:6579> EXHVER k1 f1
(integer) 2
```

- EXHINCRBY

Table 5-9 EXHINCRBY commands

Item	Description
Syntax	EXHINCRBY key field num [EX time] [EXAT time] [PX time] [PXAT time] [VER | GT | ABS version] [MIN minval] [MAX maxval] [KEEPTTL]
Description	Adds **num** to the value of a field in an EXHASH key. **num** is an integer. If no keys are found, a key will be created. If the field does not exist, this command adds the field and sets the value of the field to **0** before increasing the value of the field. To add a field that does not expire, you can run this command to add the field without specifying an expiration time.
Parameters	**Key**: Object that you want to manage by running this command.
	field: An element in the EXHASH command. An EXHASH key can have multiple fields.
	num: Integer to be added to the value of the field.
	EX: Relative expiration time of the field, in seconds. **0** indicates that the field will expire immediately. If this parameter is not specified, the field does not expire.
	EXAT: Absolute expiration time of the field, in seconds. **0** indicates that the field will expire immediately. If this parameter is not specified, the field does not expire.
	PX: Relative expiration time of the field, in milliseconds. **0** indicates that the field will expire immediately. If this parameter is not specified, the field does not expire.
	PXAT: Absolute expiration time of the field, in milliseconds. **0** indicates that the field will expire immediately. If this parameter is not specified, the field does not expire.

Item	Description
	VER: Version number of the field. If the field exists, the version number specified by this parameter is compared with the current version number. If they are the same, the system continues to run this command and increases the version number by 1. If they are different, an error message will be displayed. If the field does not exist or the current version of the field is 0, ignore this parameter and run the command. After the operation completes, the version number changes to 1.
	GT: Version later than the current one. If the set version number is earlier than the current one, a failure message will be returned.
	ABS: Absolute version number of the field. When a field is inserted, the version number is the version number specified by this parameter.
	KEEPTTL: The current expiration setting of the field is retained if none of the EX, EXAT, PX, and PXAT parameters are specified.
	MIN: Minimum value of the field. If the field value is less than the minimum value, an error message is returned.
	MAX: Maximum value of the field. If the field value is greater than the maximum value, an error message is returned.
Returned values	If the operation is successful, the value after **num** is added to is returned.
	Otherwise, an error message is returned.

- Example

  An example of using the MIN and MAX parameters

```
127.0.0.1:6579> EXHINCRBY k1 f1 5 min 6
(error) ERR increment or decrement would overflow
127.0.0.1:6579> EXHINCRBY k1 f1 5 min 4
(integer) 5
127.0.0.1:6579> EXHINCRBY k1 f1 5 max 9
(error) ERR increment or decrement would overflow
127.0.0.1:6579> EXHINCRBY k1 f1 3 max 9
(integer) 8
```

Example of Using ExHash Commands

JAVA(Jedis)

```
package nosql.cloud.huawei.jedis;

import redis.clients.jedis.*;
import redis.clients.jedis.util.SafeEncoder;
```



```
import java.util.ArrayList;

public class Main{
    public static void main(String[] args) throws InterruptedException {
        // Initialize the Jedis resource pool configuration.
        JedisPoolConfig jedisPoolConfig = new JedisPoolConfig();
        // Set the maximum number of connections in the resource pool.
        jedisPoolConfig.setMaxTotal(10);
        // Set the maximum number of idle connections allowed by the pool.
        jedisPoolConfig.setMaxIdle(10);
        // Set the minimum number of idle connections retained in the pool.
        jedisPoolConfig.setMinIdle(2);

       // Initialize the Jedis resource pool based on the configuration.
        // Note: If the version does not support Access Control List (ACL), the value of user must be null.
        JedisPool jedisPool = new JedisPool(jedisPoolConfig, "127.0.0.1", 6379, null, "******");

        // Obtain connections from the pool.
        try (Jedis jedis = jedisPool.getResource()) {
            // example for: EXHSET key field value [EX time] [EXAT time] [PX time] [PXAT time] [NX | XX] [VER
| ABS | GT version] [KEEPTTL]
            jedis.sendCommand(() -> SafeEncoder.encode("exhset"), "key", "field1", "value1");
            jedis.sendCommand(() -> SafeEncoder.encode("exhset"), "key", "field2", "value2", "EX", "5");

            // example for: EXHGET key field
            byte[] byteArray = (byte[]) jedis.sendCommand(() -> SafeEncoder.encode("exhget"), "key", "field1");
            System.out.println(new String(byteArray));
            byteArray = (byte[]) jedis.sendCommand(() -> SafeEncoder.encode("exhget"), "key", "field2");
            System.out.println(new String(byteArray));

            // example for: EXHGETALL key
            ArrayList<byte[]> byteArrayList = (ArrayList<byte[]>) jedis.sendCommand(() ->
SafeEncoder.encode("exhgetall"), "key");
            for (byte[] ba : byteArrayList) {
                System.out.print(new String(ba));
                System.out.print(" ");
            }
            System.out.println();

            // sleep for 5 seconds
            Thread.sleep(5000);

            // exhgetall after sleeping
            byteArrayList = (ArrayList<byte[]>) jedis.sendCommand(() -> SafeEncoder.encode("exhgetall"),
"key");
            for (byte[] ba : byteArrayList) {
                System.out.print(new String(ba));
                System.out.print(" ");
            }
        }

        // Disable the pool.
        jedisPool.close();
    }
}
```

Click **6.4 EXHASH for Ad Frequency Control** to view the best practices of the EXHASH command.

5.7 Large Bitmap Initialization

The open-source Redis uses string bitmaps, which may create super large strings and affect the performance of big keys in some scenarios. GeminiDB Redis API uses bitmaps in a special encoding format. The internal sharding algorithm prevents super large strings from being created and allows you to insert and delete a random number of bits efficiently.

However, in practice, a super large bitmap of the string type may be obtained from other sources. For example, it takes a long time to use the **SET** command to insert a super large bitmap (64 MB) into GeminiDB Redis API. Other normal accesses may be interfered with and a jitter may be created because of latency. To address these issues, we provide a smooth insertion solution. A super large bitmap is split into smaller strings (for example, 1 MB). The **SET** command is used for the first insertion, and then a GETBIT read-only command is used to convert the strings to bitmaps. The subsequent character strings are inserted by running the **APPEND** command.

Precautions

- To use this function, you need to upgrade your instance to a specific version. Choose **Service Tickets > Create Service Ticket** in the upper right corner of the console. Contact customer service to check whether your instance version supports this function.
- The **APPEND** command has requirements on the sequence. Therefore, **APPEND** disorder must be avoided in the entire process (in concurrent APPEND scenarios).
- PIPELINE acceleration and PIPELINE can ensure the execution sequence.
- The smaller (recommended: 256 KB to 1 MB) the substrings are, the less the latency will vary, but initialization will take longer.

Code Reference

- C++

```
#include <string>
#include <vector>

#include "hiredis/hiredis.h"
constexpr std::size_t kBitmapSubSize = 1024 * 1024;  // 1 MB
void SmoothInitBitmap(std::string bitmap) {
 // Split bitmap
 std::vector<std::string> sub_bitmaps;
 std::size_t pos = 0;
 while (pos < bitmap.size()) {
   sub_bitmaps.emplace_back(bitmap.substr(pos, kBitmapSubSize));
   pos += kBitmapSubSize;
}
 std::string key = "BITMAP_KEY";
 // Connect to redis
 redisContext* redis = redisConnect("127.0.0.1", 6666);
 redisReply* reply = nullptr;
 // First part use 'SET' command
 reply = (redisReply*)redisCommand(redis, "SET %b %b", key.data(), key.size(), sub_bitmaps[0].data(),
sub_bitmaps[0].size());
 freeReplyObject(reply);
 // Use 'GETBIT' to transform to bitmap format
 reply = (redisReply*)redisCommand(redis, "GETBIT %b 0", key.data(), key.size());
 freeReplyObject(reply);
 // Use 'APPEND' for remaining bitmap data
 for (auto i = 1u; i < sub_bitmaps.size(); ++i) {
   reply = (redisReply*)redisCommand(redis, "APPEND %b %b", key.data(), key.size(),
sub_bitmaps[i].data(), sub_bitmaps[i].size());
   freeReplyObject(reply);
}
}
int main() {
 std::string bitmap
="123457890abcdef123457890abcdef123457890abcdef123457890abcdef123457890abcdef123456";
 SmoothInitBitmap(bitmap);
}
```


- JAVA(Jedis)

```
package nosql.cloud.huawei.jedis;

import redis.clients.jedis.Jedis;

import java.nio.ByteBuffer;
import java.util.BitSet;

public class BitMapOperation {
  private Jedis jedis;

  public BitMapOperation(Jedis jedis) {
      this.jedis = jedis;
  }

   /**
    * SetBit operation especially for big bitmap
    *
    * @param key         key
    * @param value       value
    * @param groupLength groupLength (Unit: byte)
    */
   public void setBitGrouped(byte[] key, BitSet value, int groupLength) {
      if (value.isEmpty()) {
          jedis.set(key, new byte[0]);
          return;
      }

      byte[] byteArray = disposeBitMap(value);

      // round count
      int round = byteArray.length % groupLength == 0 ? byteArray.length / groupLength :
byteArray.length / groupLength + 1;
      // last round length
      int lastPacketLength = byteArray.length % groupLength == 0 ? groupLength : byteArray.length %
groupLength;

      if (round == 1) {
          // if only one round
          byte[] lastPacketByte = new byte[lastPacketLength];
          System.arraycopy(byteArray, 0, lastPacketByte, 0, lastPacketLength);
          // set and getBit
          setAndGetBit(key, lastPacketByte);
          return;
      }

      byte[] packetByte = new byte[groupLength];
      byte[] lastPacketByte = new byte[lastPacketLength];
      for (int i = 0; i < round; i++) {
          if (i == 0) {
              // first set
              System.arraycopy(byteArray, i * groupLength, packetByte, 0, groupLength);
              // set and getBit
              setAndGetBit(key, packetByte);
          } else if (i != round - 1) {
              // regular append
              System.arraycopy(byteArray, i * groupLength, packetByte, 0, groupLength);
              jedis.append(key, packetByte);
          } else {
              // last append
              System.arraycopy(byteArray, i * groupLength, lastPacketByte, 0, lastPacketLength);
              jedis.append(key, lastPacketByte);
          }
      }
  }

  private byte[] disposeBitMap(BitSet bitSet) {
      // get words and count the number of word(Long)
      long[] words = bitSet.toLongArray();
```



```
        int n = words.length;
        if (n == 0)
            return new byte[0];
        for (int i = 0; i < n; i++) {
            // reverse
            words[i] = reverseLong(words[i]);
        }
        return longToBytes(words);
    }

    public static byte[] longToBytes(long[] longArray) {
        ByteBuffer buffer = ByteBuffer.allocate(longArray.length * 8);
        for (long value : longArray) {
            buffer.putLong(value);
        }
        return buffer.array();
    }

    public void setAndGetBit(byte[] key, byte[] value) {
        jedis.set(key, value);
        jedis.getbit(key, 0);
    }

    public static long reverseLong(long n) {
        n = (n >>> 32) | (n << 32);
        n = ((n & 0xFFFF0000FFFF0000L) >>> 16) | ((n & 0x0000FFFF0000FFFFL) << 16);
        n = ((n & 0xFF00FF00FF00FF00L) >>> 8) | ((n & 0x00FF00FF00FF00FFL) << 8);
        n = ((n & 0xF0F0F0F0F0F0F0F0L) >>> 4) | ((n & 0x0F0F0F0F0F0F0F0FL) << 4);
        n = ((n & 0xCCCCCCCCCCCCCCCCL) >>> 2) | ((n & 0x3333333333333333L) << 2);
        n = ((n & 0xAAAAAAAAAAAAAAAAL) >>> 1) | ((n & 0x5555555555555555L) << 1);
        return n;
    }
}
```

- Python

```
import redis
import random
import string
from bitmap import BitMap  # pip install bitmap
# Parameters
max_bytes = 1024 * 1024 * 64 # Construct a 64 MB bitmap.
max_bits = max_bytes * 8 # A byte consists of eight bits (over 500 million characters).
# Python built-in bitmaps are not required.
# index_list All subscripts that are to be set to 1 are stored.
index_list = []
for i in range(1000000):
    index_list.append(random.randint(0, max_bits - 1))
# Create a bitmap in a byte array.
byte_array = bytearray(max_bytes)
for i in index_list:
    index = i // 8
    offset = i % 8
    byte_array[index] |= (1 << (7 - offset))
# Convert the bitmap to bytes for subsequent operations.
bitmap_str = bytes(byte_array)
# Connect to Redis.
r = redis.Redis(host='127.0.0.1', port=6379)
r.execute_command("auth a")
key = "BITMAP_KEY"
#Separate parameters.
bitmap_pos = 0
bitmap_sub_size = 256 * 1024 # Adjust the splitting granularity.
step = bitmap_sub_size - 1
# Process the first part.
first_part = bitmap_str[bitmap_pos : bitmap_pos + step]
r.execute_command("SET", key, first_part)
r.execute_command("GETBIT", key, 0) # Run GETBIT to optimize bitmap code.
# Process the remaining part.
bitmap_pos += step
while bitmap_pos < len(bitmap_str) :
```



```
    rest_part = bitmap_str[bitmap_pos : bitmap_pos + step]
    r.execute_command("APPEND", key, rest_part)
    bitmap_pos += step
# The following is the test and verification code. Executing the code takes a long time as the GETBIT command will be executed for 1 million times.
# The BITCOUNT command with time complexity of O(N) generates 100-millisecond glitches. Do not use this command in the production environment.
# (Optional) Construct a Python built-in bitmap data verification.
bm = BitMap(max_bits)
for i in index_list:
    bm.set(i)
print('BitMap.count(): ' + str(bm.count()))
# Call the Redis command to check whether the settings are correct.
success = True
for i in index_list:
    if r.execute_command("GETBIT", key, i) != 1:
        print('GETBIT check error, pos is' + str(i))
        success = False

if success:
    print('GETBIT check success')

print("Bitcount: " + str(r.execute_command("BITCOUNT", key)))
```

5.8 Configuring Parameters for a Client Connection Pool

Setting proper parameters for a connection pool can effectively improve Redis performance of the client. Improper configuration (for example, the number of maximum connections is set to a small value) may cause applications not to be connected, affecting production services. This section uses JedisPool of the Redis client Jedis as an example to describe how to use JedisPool and its parameters, providing optimal configuration reference for service developers.

Usage Instructions

Take Jedis 4.3.1 as an example. The Maven dependency configuration is as follows:

```
<dependency>
    <groupId>redis.clients</groupId>
    <artifactId>jedis</artifactId>
    <version>4.3.1</version>
    <scope>compile</scope>
</dependency>
```

Jedis uses Apache Commons-pool2 to manage its connection pool. When defining JedisPool, pay attention to the key parameter **GenericObjectPoolConfig (Jedis)**. The following is an example for using this parameter:

```
GenericObjectPoolConfig<Jedis> config =  new GenericObjectPoolConfig<>();
config.setMaxTotal(100);
config.setMaxIdle(50);
config.setMinIdle(5);
config.setTestWhileIdle(true);
....
```

The initialization method of Jedis is as follows:

```
JedisPool pool = new JedisPool(config, host, port, timeout, password);// Create a connection pool.
try (Jedis jedis = pool.getResource()) {//Obtain a connection and automatically release the connection after the execution.
    //Run the following command on the Jedis client:
} catch (Exception e) {
```



```
    e.printStackTrace();
}
pool.close();//Close the connection pool.
```

Parameter Description

Jedis connections are resources managed by JedisPoo. JedisPool controls resources and protects threads. Proper **GenericObjectPoolConfig** configuration can improve Redis service performance and reduce resource overhead. The following table lists some important parameters and describes how to set the parameters.

Table 5-10 Common Jedis parameters

Jedis Parameter	Description	Default Value	Recommended Setting
maxTotal	Maximum number of concurrent connections in the current resource pool. The number of Redis connections must be set based on the service volume. If the value is set to too large, resources are wasted. If the value is set to too small, connections cannot be obtained, affecting services.	8	The number of client nodes multiplied by the value of **maxTotal** cannot exceed the maximum number of Redis connections. Assume that the QPS of a connection is about 1000 per second or millisecond and the expected QPS of a single Redis instance is 50,000. Theoretically, the required resource pool size (MaxTotal) is 50 (50000/1000).
maxIdle	Maximum number of idle connections in a resource pool. After the number of idle connections reaches the value of **maxIdle**, the resource pool starts to revoke idle connections until the number of idle connections reaches the value of **minIdle**. This prevents empty connections from being occupied and resources from being wasted.	8	**maxIdle** indicates the maximum number of connections required by the service, and **maxTotal** indicates the margin. You are not advised to set **maxIdle** to a small value. Otherwise, new Jedis (new connection) overhead occurs.

Jedis Parameter	Description	Default Value	Recommended Setting
minIdle	Minimum number of idle connections in a resource pool. These connections are not revoked to prevent delayed connection creation when traffic increases.	0	10 to 20
maxWaitMillis	Maximum wait time (in milliseconds) of the invoker after the resource pool connections are used up	-1	You are advised to set a proper timeout interval to prevent application blocking after connection pools are used up.
testWhileIdle	Indicates whether to use the **ping** command to monitor the connection validity during idle resource monitoring. Invalid connections will be destroyed.	false	true
testOnBorrow	Indicates whether to check the connection validity (by sending a ping request) each time a connection is obtained from a resource pool. Invalid connections will be released.	false	The preset value is recommended. If this parameter is set to **true**, a **ping** command is sent before each command is executed, which affects the performance of applications with a large number of concurrent requests. If high service availability is required, set this parameter to **true** to ensure that the connection is valid.
testOnReturn	Indicates whether to check the connection validity (by sending a ping request) each time a connection is returned to a connection pool. Invalid connections will be released.	false	The preset value is recommended. If this parameter is set to **true**, a **ping** command is sent after each command is executed, which affects the performance of applications with a large number of concurrent requests.
timeout	**Socket timeout** value of Jedis, in milliseconds	2000	200 to 1000

5.9 Using Parallel SCAN to Accelerate Full Database Scanning

When there are a large number of keys in an instance, Redis SCAN commands takes a long time to work. GeminiDB Redis API utilizes a distributed architecture that enables concurrent scanning of multiple data partitions, resulting in parallel acceleration.

Precautions

- This solution applies only to GeminiDB Redis cluster instances.
- When using the SCAN command with the **PARTITION** parameter, the returned cursor must match the same partition when continuing the scanning process. Value of the **PARTITION** should not be changed temporarily; otherwise, the scanned data may not meet the expected results, or an error may occur.

Procedure

Step 1 Obtain information about all partitions of a GeminiDB Redis instance for subsequent parallel scanning.

Data partitioning: There are many data partitions at the bottom layer of a GeminiDB Redis cluster instance, which are distributed across nodes. Each partition name is a 16-character ID. The name and total number of data partitions at the bottom layer of an instance remain fixed and do not change with any modifications made to the instance.

Obtain the data partition list: Run the **INFO ROUTE** command to obtain all data partitions of the GeminiDB Redis instance. In the following example, the instance has four data partitions: efb06d5c7a4ecb31, c7a36e9eee0103c1, 6fd3dfdbcca37686, 7f7666870a88501b.

```
127.0.0.1:6379>info route
 # Route
server: 127.0.0.1:16379 // Display the data partition on the first node.
    efb06d5c7a4ecb31 // Data partition.
    c7a36e9eee0103c1 // Data partition.
 server: 127.0.0.1:26379 // Display the data partition on the second node.
    6fd3dfdbcca37686 //Data partition.
    7f7666870a88501b // Data partition.
```

Step 2 Start multiple SCAN tasks to scan different data partitions.

GeminiDB Redis SCAN commands include a new parameter **PARTITION**, which allows users to scan specific data partitions using the open-source syntax. This feature allows for the creation of parallel scanning scripts, enabling SCAN operations on multiple data partitions simultaneously. As a result, scanning performance is greatly improved.

- For details about the standard SCAN command syntax, see **SCAN**.
- Syntax reference for the optional **PARTITION** parameter added to GeminiDB Redis API.


```
SCAN cursor [MATCH pattern] [COUNT count] [TYPE type] [PARTITION partition_index]
```

The syntax of the **MATCH**, **COUNT**, and **TYPE** parameters is the same as that of the open-source Redis.

- **PARTITION**: specifies a data partition to be scanned. If the cursor returned by the SCAN command is **0**, the data partition has been scanned.
- **partition_index**: indicates the dictionary sequence number of all data shard IDs, starting from 0. For example, if there are four data partitions in an instance, **partition_index** of the partitions is **[0,3]**. If there are 240 data partitions in an instance, **partition_index** of the partitions is **[0,239]**. For example:

```
127.0.0.1:6379> scan 0 count 2 partition 1
1) "1125900712148994"
2) 1) "memtier-1"
   2) "memtier-12
```

----End

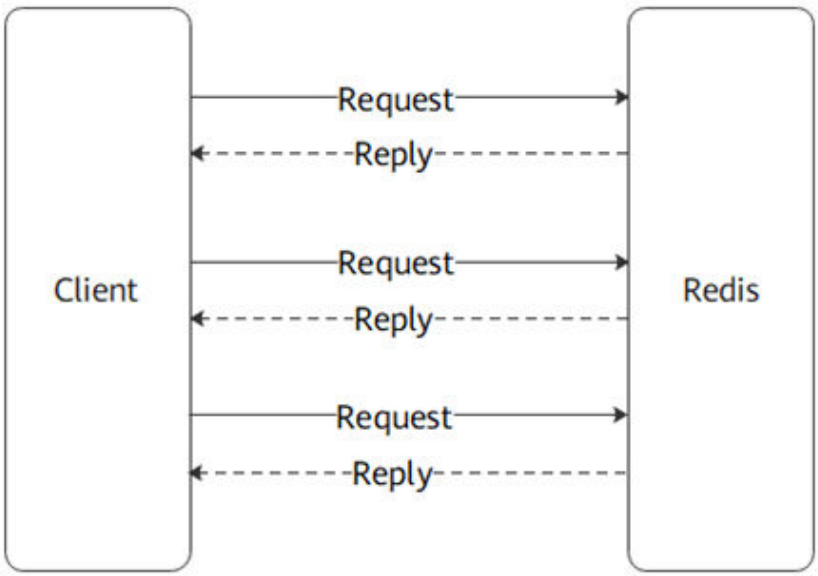
5.10 Accessing a GeminiDB Redis Instance Using a Pipeline

This section describes principles and precautions for using a pipeline to access a GeminiDB Redis instance.

Pipeline

GeminiDB Redis API is a request–reply model service.

Figure 5-9 Command execution process of GeminiDB Redis API



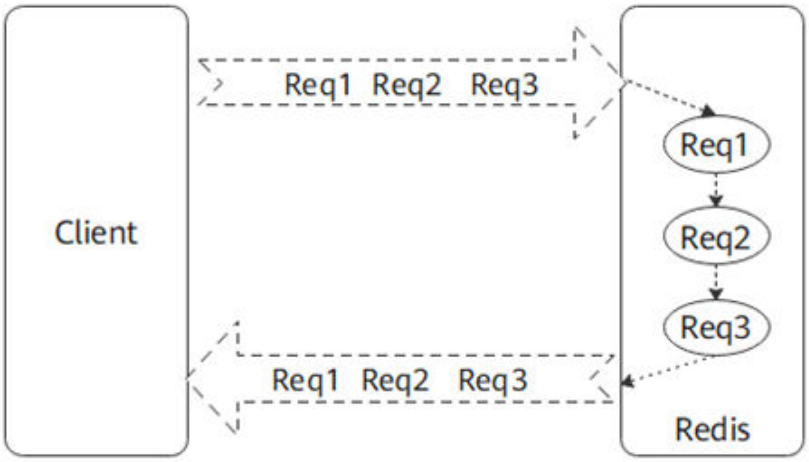
1. A client sends a command to a GeminiDB Redis server.

2. The GeminiDB Redis server receives the command and waits in a queue for processing.
3. The GeminiDB Redis server processes the command.
4. The GeminiDB Redis server sends the result to the client.

Steps 1 and 4 are I/O operations, which are slow and affected by network conditions. Therefore, bottlenecks may occur.

To reduce network costs and fully utilize performance of GeminiDB Redis API, you can execute multiple commands in a pipeline at once without waiting for individual command reply.

Figure 5-10 Accessing a GeminiDB Redis instance using a pipeline



The I/O operation is performed once to issue three commands.

Using a pipeline can reduce read() and write() system calls of a client and server and improve program execution efficiency.

Pipeline Size Selection and Precautions

Although pipelines can reduce I/O costs, a bigger pipeline size is not always better. Pipelines have a limit on improving program performance. If there are large amounts of data in a pipeline, the client has to wait longer time. If the socket buffer is full due to a large pipeline, network congestion may occur, causing performance deterioration.

Based on experience, 30 to 100 commands in a pipeline can fully utilize database performance. An optimal size is subject to the actual test result.

Other precautions:

- Pipelines do not guarantee atomicity. When processing batch commands, a server parses and executes commands in sequence. These commands are independent of each other. The server may also execute commands of other clients during this process. If a command fails, other commands are not affected. To achieve atomicity, you can execute transactions or Lua scripts.

- A GeminiDB Redis instance must cache the results before finishing processing all commands, so a large pipeline may cause out of memory (OOM) or network congestion. If a single command is too long, the pipeline size needs to be reduced.
- To fulfill stringent requirements on low latency, large pipelines are not recommended.

5.11 Processing Transactions on a GeminiDB Redis Instance

About Transactions

A transaction is a logical unit which groups a set of read or write operations, so that they either succeed or fail collectively. In a connection, after a client executes the **multi** command, a GeminiDB Redis instance starts to cache subsequent commands in a queue. When the client sends the **exec** command, the GeminiDB Redis instance executes all commands in the queue in sequence. If a command fails, transactions will be rolled back. All commands are either successful or failed.

Table 5-11 Related commands

Command	Description
WATCH	Monitors one or more keys. If a key is modified before a transaction is executed, the whole transaction aborts.
UNWATCH	Flushes all watched keys.
MULTI	Identifies start of a transaction block.
EXEC	Executes all commands in a transaction block.
DISCARD	Flushes queued commands in a transaction bock and exits the transaction block.

CAUTION

- When a proxy cluster is used, all keys in a transaction must have the same hashtag to ensure transaction atomicity. If hashtag is not used, a transaction will be split into common commands. In this case, the atomicity cannot be ensured.
- Atomic transactions mean that all of them will either succeed or fail, so you need to take care with command validity when compiling a transaction.
- The commands in a transaction are executed in sequence, so you need to take care with the command sequence when compiling a transaction.
- Do not pack too many or complex commands in a single transaction, or requests may be blocked or the instance status may be abnormal.

Example Code

When the client modifies **key1** and **key2** at the same time in a transaction, they are either successfully modified or fail to be modified at the same time.

```
package nosql.cloud.huawei.jedis;
import java.util.List;
import redis.clients.jedis.Jedis;
import redis.clients.jedis.Transaction;

public class TranscationTest {
    private static final String host = "127.0.0.1";
    private static final int port = 6379;
    private static final String pwd = "password";

    private static Jedis jedis;
    static {
        jedis = new Jedis(host, port);
        String authString = jedis.auth(password);
        if (!authString.equals("OK")) {
            jedis.close();
            jedis = null;
        }
    }
    public static void main(String[] args) {
        if (jedis == null) {
            return;
        }

        String str_key1 = "{str}key1";
        String str_key2 = "{str}key2";
        jedis.set(str_key1, "0");
        jedis.set(str_key2, "0");
        jedis.watch(str_key1);
        // Starts processing transactions.
        Transaction tx = jedis.multi();
        tx.set(str_key1, "500");
        tx.get(str_key1);
    tx.set(str_key2, "1000");
        tx.get(str_key2);
        List<Object> result = tx.exec();
        if (result.isEmpty()) {
           System.out.println ("Error: The transaction is interrupted.");
        } else {
          System.out.println ("Succ: The transaction is executed successfully.");
        }
        System.out.println("str_key1: {}, str_key2: {}", jedis.get(str_key1), jedis.get(str_key2));
        jedis.close();
    }
}
```

5.12 Retry Mechanism for GeminiDB Redis Clients

The retry mechanism for GeminiDB Redis clients can ensure high availability and stability of applications if the network is unstable or a server is temporarily faulty.

There may be the following temporary faults.

Cause	Description
HA is triggered.	GeminiDB Redis API automatically monitors node health. If a node breaks down, a primary/standby switchover or shard takeover is automatically triggered. Generally, HA may be triggered when: • A GeminiDB process on a node restarts due to OOM or hardware faults. • Nodes are automatically removed or added when they are scaled or specifications are changed. In these scenarios, clients may be intermittently disconnected in seconds or commands time out.
The network fluctuates.	Complex network environments between clients and GeminiDB Redis servers may cause problems such as occasional network jitter and data retransmission. In this case, requests initiated by the clients may temporarily fail.
Servers are overloaded.	Requests initiated by clients may not be responded immediately due to heavy loads and slow queries on GeminiDB Redis servers. As a result, the requests time out.

When setting retry rules on clients, follow the best practices below.

Best Practice	Description
Configure a proper interval and retry times.	Configure a proper interval and retry times based on business requirements. If an excessive number of retries are attempted, it takes a longer time to recover from a fault. If the interval between retries is shorter than expected, servers may become overwhelmed. In heavy-load scenarios, you are advised to increase the retry interval exponentially to prevent server breakdown due to a large number of concurrent retries.
Retry only idempotent operations.	Commands have been executed on a server, but a timeout occurs when the result is returned to a client. In this case, the commands may be executed repeatedly. Therefore, you are advised to retry only idempotent operations (for example, the **SET** command), and the result remains unchanged after multiple operations. For non-idempotent operations (for example, the **INCR** command), you need to confirm whether duplicate data can be tolerated, and multiple operations may increase a counter value.
Generate client logs.	You are advised to configure the system to generate client logs during the retry process, such as the connected IP address and port number, error commands, and keys, to facilitate troubleshooting.

The following SDK code examples are used for reference only.

Jedis

In JedisPool mode, Jedis 4.0.0 or later supports retries. The following uses Jedis 4.0.0 as an example:

```
package nosql.cloud.huawei.jedis;

import redis.clients.jedis.DefaultJedisClientConfig;
import redis.clients.jedis.HostAndPort;
import redis.clients.jedis.JedisClientConfig;
import redis.clients.jedis.UnifiedJedis;
import redis.clients.jedis.providers.PooledConnectionProvider;
import java.time.Duration;

// UnifiedJedis API supported in Jedis >= 4.0.0
public class UnifiedJedisDemo {
    private static final int MAX_ATTEMPTS = 5;
    private static final Duration MAX_TOTAL_RETRIES_DURATION = Duration.ofSeconds(15);
    public static void main(String[] args) {
        // Basic connection config
        JedisClientConfig jedisClientConfig = DefaultJedisClientConfig.builder().password("xxx").build();
        // Implement retry
        PooledConnectionProvider provider = new
            PooledConnectionProvider(HostAndPort.from("{ip}:{port}"), jedisClientConfig);
        UnifiedJedis jedis = new UnifiedJedis(provider, MAX_ATTEMPTS, MAX_TOTAL_RETRIES_DURATION);
        try {
            System.out.println("set key: " + jedis.set("key", "value"));
        } catch (Exception e) {
            // Signifies reaching either the maximum number of failures,
            MAX_ATTEMPTS, or the maximum query time, MAX_TOTAL_RETRIES_DURATION
            e.printStackTrace();
        }
    }
}
```

Redisson

```
package nosql.cloud.huawei.jedis;
import org.redisson.Redisson;
import org.redisson.api.RBucket;
import org.redisson.api.RedissonClient;
import org.redisson.config.Config;

public class RedissonDemo {
    private static final int TIME_OUT = 3000;
    private static final int RETRY_ATTEMPTS = 5;
    private static final int RETRY_INTERVAL = 1500;

    public static void main(String[] args) {
        Config config = new Config();
        config.useSingleServer()
              .setPassword("xxx")
              .setTimeout(TIME_OUT)
              .setRetryAttempts(RETRY_ATTEMPTS)
              .setRetryInterval(RETRY_INTERVAL)
              .setAddress("redis://{ip}:{port}");
        RedissonClient redissonClient = Redisson.create(config);
        RBucket<String> bucket = redissonClient.getBucket("key");
        bucket.set("value");
    }
}
```

Go-redis

```
package main
```



```
import (
	"context"
	"fmt"
	"time"

	"github.com/redis/go-redis/v9"
)

var ctx = context.Background()

func main() {

    client := redis.NewClient(&redis.Options{
        Addr:     "localhost:6379",
        Password: "", // no password set
        DB:       0,  // use default DB
        MaxRetries: 3, // set max retry times
        MinRetryBackoff: time.Duration(1) * time.Second, // set retry interval
        MaxRetryBackoff: time.Duration(2) * time.Second, // set retry interval
    })

    // Execute command
    err := client.Set(ctx, "key", "value", 0).Err()
    if err != nil {
        panic(err)
    }

    // Test
    pong, err := client.Ping(ctx).Result()
    if err != nil {
        fmt.Println("Failed:", err)
        return
    }
    fmt.Println("Success:", pong)
}
```

Redis-py

```
import redis
from redis.retry import Retry
from redis.exceptions import ConnectionError
from redis.backoff import ExponentialBackoff
from redis.client import Redis
from redis.exceptions import (
   BusyLoadingError,
   ConnectionError,
   TimeoutError
)

# Run 3 retries with exponential backoff strategy
retry_strategy = Retry(ExponentialBackoff(), 3)

# Redis client with retries
client = redis.Redis(
    host = 'localhost',
    port = 6379,
    retry = retry_strategy,
    # Retry on custom errors
    retry_on_error = [BusyLoadingError, ConnectionError, TimeoutError],
    # Retry on timeout
    retry_on_timeout = True
)

try:
    client.ping()
    print("Connected to Redis!")
except ConnectionError:
    print("Failed to connect to Redis after retries.")
```



```
try:
    client.set('key', 'value')
    print("Set key and value success!")
except ConnectionError:
    print("Failed to set key after retries.")
```

Hiredis

Hiredis is a minimalistic C client library and does not provide a preset automated retry mechanism. You need to manually compile the logic.

The following is a simple example of how to implement an automated connection retry in a loop and with a delay, similar to command retry settings.

```
#include <hiredis/hiredis.h>
#include <stdio.h>
#include <unistd.h>

redisContext* connect_with_retry(const char *hostname, int port, int max_retries, int retry_interval) {
    redisContext *c = NULL;
    int attempt = 0;

    while (attempt < max_retries) {
        c = redisConnect(hostname, port);
        if (c != NULL && c->err == 0) {
            printf("Connection success!\n");
            return c;
        }

        if (c != NULL) {
            printf("Connection error: %s\n", c->errstr);
            redisFree(c);
        } else {
            printf("Connection failed\n");
        }

        printf("Retrying in %d seconds...\n", retry_interval);
        sleep(retry_interval);
        attempt++;
    }

    return NULL;
}

int main() {
    const char* hostname = "127.0.0.1";
    int port = 6379;
    int max_retries = 5;
    int retry_interval = 2;

    redisContext *c = connect_with_retry(hostname, port, max_retries, retry_interval);
    if (c == NULL) {
        printf("Failed to connect to Redis after %d attempts\n", max_retries);
        return 1;
    }

    redisFree(c);
    return 0;
}
```

5.13 GeminiDB Redis API Pub/Sub

Huawei Cloud GeminiDB Redis is fully compatible with Pub/Sub of open-source Redis. This section describes how to configure this model.

Pub/Sub

SUBSCRIBE, **UNSUBSCRIBE**, and **PUBLISH** implement the **publish–subscribe pattern**. In this pattern, publishers do not directly send messages to a specific subscriber but publish them to a channel. All subscribers who are interested in this channel can receive the messages. Pub/Sub enables the decoupling of the publisher and subscribers, eliminating the need for publishers to know their subscribers.

Application Scenarios

Pub/Sub plays an important role in many scenarios, for example:

- Real-time chat

In IM applications, messages need to be quickly transferred. With Pub/Sub, users can subscribe to their own chat channels. After message are published, the subscribes on this channel receive the messages immediately. In this manner, real-time performance and high efficiency can be achieved.

- Real-time notification system

On e-commerce websites or social media platforms, users need to receive notifications such as order status updates, comments, and likes in real time. With Pub/Sub of GeminiDB Redis API, the system can immediately publish a notification when the status changes, and all related users will receive the notification in a timely manner.

- Monitoring and log system

In the microservice architecture, the Pub/Sub model can be used for status monitoring and log collection between services. Services can publish status information or log messages to specific channels. The monitoring service or log collection service can subscribe to these channels to implement real-time monitoring and data collection.

- Real-time gaming messages

In an online game, data between players each time an action occurs needs to be synchronized in time. Pub/Sub can be used for message transfer and game event notification to ensure that all players receive status updates at the same time.

- Data stream processing

Real-time processing and analysis are key to data stream applications. With Pub/Sub, data producers can publish data streams, and consumers can subscribe to these streams for real-time processing and analysis.

Basic operations

For example, to subscribe to "channel11" and "ch:00," clients can run the following command:

```
SUBSCRIBE channel11 ch:00
```

These clients will receive messages on these channels from other clients in the sequence in which the messages were sent.

Advanced function:

Pub/Sub supports pattern matching. Clients may subscribe to glob-style patterns to receive all the messages sent to channel names matching a given pattern. For example:

```
PSUBSCRIBE news.*
```

Subscribers will receive all messages sent to channels such as news.art.figurative and news.music.jazz.

CAUTION

- **Message loss**: Pub/Sub does not ensure message durability. Therefore, messages may be lost when the network is faulty or a subscriber is not connected.
- **Performance**: In a high-concurrency environment, the Pub/Sub performance may be limited. Performance need to be tested and improved based on specific scenarios.
- If both **SUBSCRIBE** and **PSUBSCRIBE** are executed, duplicate messages may be received. Check whether the business logic is correct.

Java Sample Code (Jedis)

Message publisher

```
import redis.clients.jedis.Jedis;
public class GeminiDBPubClient {
    private Jedis jedis;

    public GeminiDBPubClient(String ip, int port, String password){
        jedis = new Jedis(ip, port);
        // The instance password for GeminiDB.
        String authString = jedis.auth(password);
        if (!authString.equals("OK"))
        {
            System.err.println("AUTH Failed: " + authString);
            return;
        }
    }

    public void pub(String channel, String message){
        System.out.println("  >>> Publish > Channel: " + channel + " > Sent Message: " + message);
        jedis.publish(channel, message);
    }

    public void close(String channel){
        System.out.println("  >>> Publish End > Channel:" + channel + " > Message:quit");
        // The message publisher has finished sending, sending a "quit" message.
        jedis.publish(channel, "quit");
    }
}
```

Message subscriber

```
import redis.clients.jedis.Jedis;
import redis.clients.jedis.JedisPubSub;
public class GeminiDBSubClient extends Thread {
    private Jedis jedis;

    private String channel;
```



```
    private JedisPubSub listener;

    public GeminiDBSubClient(String ip, int port, String password){
        jedis = new Jedis(host,port);
        // The instance password for GeminiDB.
        String authString = jedis.auth(password); //password
        if (!authString.equals("OK"))
        {
            System.err.println("AUTH Failed: " + authString);
            return;
        }
    }

    public void setChannelAndListener(JedisPubSub listener, String channel){
        this.listener=listener;
        this.channel=channel;
    }

    private void subscribe(){
        if(listener==null || channel==null){
            System.err.println("Error:SubClient> listener or channel is null");
        }
        System.out.println("  >>> Subscribe > Channel:" + channel);
        // The receiver will block the process while listening for subscribed messages until it receives a "quit"
message (passive mode) or actively cancels the subscription.
        jedis.subscribe(listener, channel);
    }

    public void unsubscribe(String channel){
        System.out.println("  >>> Unsubscribe > Channel:" + channel);
        listener.unsubscribe(channel);
    }

    @Override
    public void run(){
        try {
            System.out.println("----------Subscribe Start-------");
            subscribe();
            System.out.println("----------Subscribe End-------");
        } catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

Message listener

```
import redis.clients.jedis.JedisPubSub;
public class GeminiDBListener extends JedisPubSub {
    @Override
    public void onMessage(String channel, String message) {
        System.out.println("  <<< Subscribe < Channel:" + channel + " > Receive Message:" + message );
        // When the received message is "quit," unsubscribe (passive mode).
        if(message.equalsIgnoreCase("quit")){
            this.unsubscribe(channel);
        }
    }
    @Override
    public void onPMessage(String pattern, String channel, String message) {
        // TODO Auto-generated method stub
    }
    @Override
    public void onSubscribe(String channel, int subscribedChannels) {
        // TODO Auto-generated method stub
    }
    @Override
    public void onUnsubscribe(String channel, int subscribedChannels) {
```



```
        // TODO Auto-generated method stub
    }
    @Override
    public void onPUnsubscribe(String pattern, int subscribedChannels) {
        // TODO Auto-generated method stub
    }
    @Override
    public void onPSubscribe(String pattern, int subscribedChannels) {
        // TODO Auto-generated method stub
    }
}
```

5.14 Implementing Distributed Locks Using Lua Scripts for GeminiDB Redis API

In a distributed system, distributed locks are used to ensure that only one process or thread can execute a specific code snippet at a time.

This section describes how to use Lua scripts to implement distributed locks.

Redis Distributed Locks

Redis offers a basic locking mechanism that relying on atomic commands to implement distributed locks. One of the simplest ways to implement distributed locks is using the SETNX command. SETNX sets the value of a key only if the key does not exist. In this way, the key value is successfully set for the first process that obtains a lock, and subsequent processes that attempt to obtain the lock fail until the lock is released.

To prevent a lock from being released forever (for example, the process that holds the lock crashes), an expiration time is usually set for the lock using the EXPIRE command. In versions later than Redis 2.6.12, the EX and NX options are added to the SET command. You can set the expiration time when setting the key. This operation is atomic.

- Acquiring a lock

You can run the following command to acquire a lock:

```
SET resource_name my_random_value NX PX 30000
```

The NX parameter is used to check whether the key exists. If it does not, that is, no one holds the lock, the lock is successfully acquired.

The PX parameter is mandatory and is used to set a lock validity period accurate to milliseconds. If a lock holder exits abnormally or a lock expires, the lock is automatically released to prevent deadlocks.

- Releasing a lock

Releasing a lock is more complex. A lock can be released only by its holder. To release multiple locks in sequence, you need to execute a Lua script.

Lua script:

```
if redis.call("get",KEYS[1]) == ARGV[1] then
    return redis.call("del",KEYS[1])
```



```
else
    return 0
end
```

The Lua script must be executed together with the EVAL command, for example:

```
EVAL 'if redis.call("get",KEYS[1]) == ARGV[1] then return redis.call("del",KEYS[1]) else return 0 end' 1
resource_name my_random_value
```

This script ensures that only the holder can release the lock.

- Analysis

  The preceding solution is easy to use but has the following disadvantages:

  – The expired lock is released, but transactions are incomplete.

  – Reentrant locks are not supported.

  – No notification mechanism is available. Locks need to be preempted in polling mode, consuming a lot of CPU resources.

For production applications, the Redis distributed lock library is recommended to balance functions and performance.

The following uses Redisson as an example to describe how to use the Redis distributed lock library.

Implementing Distributed Locks Using Redisson

Redisson is a Redis Java client and provides the distributed locking feature. A distributed lock is a mechanism used to synchronously access shared resources in a distributed system. Redisson implements distributed locks through atomic operations of Redis to ensure that only one client can access a resource at a time.

Distributed locks based on Redisson have the following features:

- High efficiency: Redis features high performance and memory storage, making distributed lock operations very fast.
- Easy to use: Various APIs allow developers to easily use distributed locks in Java applications.
- Reliability: Distributed locks based on Redisson are highly reliable. Even if a partition or node breaks down, the locks are not affected.

Example

```
import org.redisson.Redisson;
import org.redisson.api.RLock;
import org.redisson.api.RedissonClient;
import org.redisson.config.Config;

public class LockExamples {
    public static void main(String[] args) {
        // Creates a Redisson client.
        Config config = new Config();
        config.useSingleServer().setAddress("redis://127.0.0.1:7200");
        RedissonClient redisson = Redisson.create(config);

        // Obtains a distributed lock.
        RLock lock = redisson.getLock("myLock");

        try {
        // Acquires a lock.
            lock.lock();
            System.out.println("Lock acquired, executing critical section...");
```



```
            // Executes the code to acquire a lock.
            // ...
            System.out.println("Critical section executed, releasing lock...");
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
          // Release a lock.
            lock.unlock();
        }

      // Closes the Redisson client.
        redisson.shutdown();
    }
}
```

Other Recommended Implementations of Distributed Locks

Redisson is a Java client which has implemented distributed locks for various programming languages. Here are a few links to available implementations from the **Redis official website**:

- **Redlock-rb** (Ruby implementation). There is also a **fork of Redlock-rb** that adds a gem for easy distribution.
- **RedisQueuedLocks** (Ruby implementation).
- **Redlock-py** (Python implementation).
- **Pottery** (Python implementation).
- **Aioredlock** (Asyncio Python implementation).
- **RedisMutex** (PHP implementation with both **Redis extension** and **Predis library** clients support).
- **Redlock-php** (PHP implementation).
- **cheprasov/php-redis-lock** (PHP library for locks).
- **rtckit/react-redlock** (Async PHP implementation).
- **Redsync** (Go implementation).
- **Redisson** (Java implementation).
- **Redis::DistLock** (Perl implementation).
- **Redlock-cpp** (C++ implementation).
- **Redis-plus-plus** (C++ implementation).
- **Redlock-cs** (C#/.NET implementation).
- **RedLock.net** (C#/.NET implementation). Includes async and lock extension support.
- **ScarletLock** (C# .NET implementation with configurable datastore).
- **Redlock4Net** (C# .NET implementation).
- **node-redlock** (NodeJS implementation). Includes support for lock extension.
- **simple-redis-mutex** (Node.js implementation) Available as an **NPM package**.
- **Deno DLM** (Deno implementation)
- **Rslock** (Rust implementation). Includes async and lock extension support.

6 Best Practices

6.1 Automated Database Access Using an Account for Multitenancy Management of GeminiDB Redis Instances

GeminiDB Redis API continuously provides enhanced features for enterprises, one of which is multitenancy. With multi-tenancy, read-only accounts and read/write accounts can be added, and databases accessible to each account can be specified. This prevents misoperations on data of other tenants. This feature allows multiple tenants to use the same Redis instances while keeping their data isolated, facilitating enterprise development and management.

Scenarios

Multi-tenancy is a common function of database users. For example, an enterprise has service departments A and B, both of which need to use Redis to store their own data. If multi-tenancy is not used, data of departments A and B will be mixed. As a result, data breaches and misoperations may occur. After multitenancy is enabled, data of departments A and B can be stored in different Redis instances or databases, and permissions on these instances or databases can be controlled to ensure data security and reliability.

Multi-tenant databases usually have some standard attributes, such as read/write permission control as well as cross-database authentication and isolation. GeminiDB Redis instances use such comprehensive multitenancy technologies, allowing for read/write permission control and database isolation.

Advantages

In contrast to multi-tenant databases, open-source Redis supporting an access control list (ACL) in its new version only grants accounts read-only and read/write permissions. Each account can still be used to view all databases. For example, a development engineer wants to use database 1 but accidentally clears another engineer's database 0, causing a production accident. Permission isolation of GeminiDB Redis API can avoid this problem. For example, if engineer A has only the permission of database 1, database 0 is not affected even if misoperations are performed.

In addition, multitenancy of open-source Redis can be used only on a single node. Once the service volume increases and a cluster is required, multiple databases are unavailable. As a result, only database 0 is left. More than 1,000 GeminiDB Redis databases can be deployed in a cluster, and more than 200 ACL sub-accounts can be created.

Table 6-1 Comparison of permission management capabilities between open-source Redis and GeminiDB Redis API

Product	Account Read and Write Permission Control	Account Permission Isolation	Multi-DB Cluster	Default Quantity of Supported Databases
Open-source Redis	Supported	Supported	Not supported	16
GeminiDB Redis API	Supported	Supported	Supported	1,000

Solution

To use the tenant management function of GeminiDB Redis instances, you need to create accounts on the account management page and set read-only and read/write permissions for each account. For details, see **Managing Accounts**.

After an account is created, you can run **auth USER PWD** or **auth USER:PWD** for authentication and execute the SELECT DB statement to access the database on which the account has permissions. For details, see **Enabling Database Access With a Password**.

6.2 FastLoad for RTA-based Ad Placement

Scenarios

Advertisement (ad) placement is indispensable for enterprise promotion and marketing, especially for new media facing fierce competition. Now there are diversified advertising channels and detailed tailored ad placement.

Customer's increased awareness of ad ROI demands precise audience selection. For example, on a short video platform, advertisers have to configure rules on

placing ads such as age, gender, and education background of audiences. The inflexibility hinders advertisers from making decisions on ad placement, and hundreds of millions or even billions of advertising fees need to be paid every year. However, it is still difficult to accurately reach target audiences. To let advertisers have autonomy to deliver or reject each ad request, Real-Time API (RTA) rose to the challenge.

RTA is used to meet the real-time personalized delivery needs of advertisers.

Challenges

Advertisers' RTA system read data from core profile databases, helping advertisers make placement decisions. The newer the data, the better the placement effect. Therefore, the latest data generated by a big data platform needs to be written into profile databases in a timely manner. To address high concurrency, ultra-low latency, and ultra-large volumes of RTA requests, core profile databases must have the following features:

- **Quick import of vast amount of data; accurate decision-making**

  Hundreds of GB or even several TB of all profile data needs to be periodically imported to profile databases. The faster the data is imported, the more accurate the model is, and the better ad placement effect is.
- **High-concurrency access**

  The RTA system needs to handle a large number of real-time bidding requests. For example, hundreds of thousands to millions of QPS are sent by RTA systems of e-commerce and financial customers.
- **Stable low latency**

  The media asks advertisers to provide decisions within 40 ms to 100 ms. Databases need to execute requests within single-digit milliseconds.
- **Low cost**

  To achieve ultimate performance, open-source self-hosted Redis needs to be installed for handling RTA requests. However, expensive TB-level data storage devices give advertisers a dilemma in selecting suitable models.

In an RTA system, common profile databases have the following problems:

- MySQL: It is difficult to handle hundreds of thousands to millions of concurrent QPS at low latency.
- MongoDB/HBase: It is inexpensive to store TB-level data, but MongoDB and HBase instances cannot maintain a stable low latency. High timeout rate may cause project suspension, injurious to commercial interests.
- In-memory database: For example, open-source self-hosted Redis is widely used in the industry. In-memory databases provide ultimate performance with high concurrency and low latency. However, there are risks such as poor stability and data loss. It costs too much and takes long to import TB-level user profile data.

GeminiDB Redis API features stability, low latency, and cost-effectiveness, and data can be imported extremely fast using FastLoad.

Solution Overview

In an RTA system, storage of GeminiDB Redis instances costs less than that of open-source self-hosted Redis instances. FastLoad enables quick offline data

import. GeminiDB Redis API ensures stability and low latency and has rich practice cases of online advertising and recommendation services.

Advantages

- **5 to 10 times faster data import by FastLoad**

  Traditional databases can only write data one by one using standard protocols. Data is crunched by the compute layer through a complex process and then written to the storage layer. Therefore, it usually takes hours or days for a big data platform to periodically import hundreds of GB or even several TB of profile data, which has a great impact on online services.

  FastLoad, an enterprise-level feature of GeminiDB Redis API, improves the data import speed by 5 to 10 times and reduces the impact on online services. To address RTA requests, the big data platform handles highly-concurrent workloads, FastLoad directly transfers massive volumes of data into a storage engine of the GeminiDB Redis instance via a dedicated high-speed persistent channel, and the storage engine on the GeminiDB Redis database orchestrates the data.

- **Millions of concurrent requests and latency in sub-milliseconds**

  GeminiDB Redis API uses a separated storage and compute architecture. Three copies of data are stored in a distributed shared storage pool. All nodes support efficient reads and writes, and compute power can be scaled up and out, making it easy to cope with workload spikes.

  With a multi-thread architecture, high-performance storage pool, and in-depth optimization of the memory data structure and access algorithm, GeminiDB Redis API can respond to requests in sub-milliseconds. Such an ultra-low latency is critical to real-time data processing and analysis, especially in scenarios such as online games, financial technologies, advertising systems, and real-time recommendation systems. Therefore, GeminiDB Redis API is ideal for large-scale real-time interaction and high-frequency transactions.

  With vast experience on the live network, GeminiDB Redis API can handle more than one million QPS while ensuring average latency of 1 ms and p99 latency of 2 ms.

- **Efficient data compression and storage at low costs**

  By compressing both logical and block data, GeminiDB Redis API greatly reduces storage resource consumption while maintaining ultimate performance. Compute and storage resources are decoupled from each other, so they can be flexibly scaled out. Achieving optimized resource utilization means a range of benefits for enterprises, most notably saving costs in storing data.

  With vast experience on the live network, GeminiDB Redis API has a compression ratio of 4:1. That is, only about 3 TB out of 12 TB of data is occupied on a GeminiDB Redis instance.

6.3 PITR for Archiving Gaming Data

Scenarios

Databases may experience faults such as data corruption, loss, or accidental deletion. To ensure services are running properly, you need to restore a database to a normal state before the faults occurred. Traditional databases adopt a periodic backup policy. That is, data is restored when the system is faulty. Data restoration takes a long time. As a result, customer services are severely affected.

Solution Overview

Point-in-Time Recovery (PITR) allows you to restore the database to a particular point in time if data is lost or corrupted due to misoperations or accidental deletion.

Some game players may exploit vulnerabilities to duplicate equipment and currency, leading to unfairness. Traditional databases are backed up once a day, making it difficult to restore data to a specific point in time. With PITR of GeminiDB Redis API, you can specify a specific time point for data restoration. Data can be restored within 5 minutes at least.

Advantages

PITR of GeminiDB Redis API maintains your data from past timestamps and does not affect data snapshots. If there is a fault, data can be restored to a specified point within 5 minutes. Therefore, GeminiDB Redis AP is widely used in industries such as gaming and finance.

- **Backup tasks are not affected, and services are running stably.**

  PITR does not affect data backup and access.

  GeminiDB Redis API enables you to create snapshots by recording the file system status instead of by copying files. File metadata (such as data block information and addressing information) at the current moment is stored to generate snapshots. Therefore, services are not affected during snapshot creation.

- **Data is restored in minutes regardless of its volume.**

  PITR snapshots can be stored on your local PC and do not need to be uploaded to cold storage media. Therefore, data replication and migration are not involved, and data can be restored anytime.

  Even hundreds of GB of data can be restored within 5 minutes. After being restored to a specified point in time, data can also be restored multiple times to a state before or after that point in time.

- **GeminiDB Redis API has better backup performance than open-source Redis**

  Open-source Redis uses copy-on-write (CoW) to effectively enable snapshot persistence in its multi-thread architecture. When Redis calls fork() to create a child process, its parent process will be blocked for hundreds of milliseconds.

As a result, jitter will occur. CoW may cause memory overuse. If a large number of writes are performed while the parent process is forked, memory is severely wasted or even OOM occurs (memory usage < 50%). PITR frees you from copying or migrating data, so services are not affected. Snapshots can be quickly created, and data can be restored stably and securely.

Solution

For details about how to enable PITR of the GeminiDB Redis API and restore data to a specified time point, see **4.8.3 Restoring to the Original Instance Using PITR**.

6.4 EXHASH for Ad Frequency Control

EXHASH, a hash data type, provides general HASH functions and allows you to specify expiration times and version numbers for fields. It supports flexible data structures and can help you simplify service development in various scenarios.

This section describes how to use EXHASH commands of GeminiDB Redis API to simplify service development of frequency control and shopping cart.

EXHASH Commands

For details, see **EXHASH commands**.

Scenarios

- Frequency control

  Frequency control allows users to restrict the number of operations performed within a certain period (for example, one day, one week, or one month), and limit the number of times an ad or information displayed on a platform within a specified period. This helps prevent overexposure and ad fatigue, optimizes ad performance, improves the conversion rate, and avoid malicious activities, such as manipulating online traffic, comments and likes.

  Frequency control involves three elements: user ID (key), ad ID (field), and the number of times an activity is triggered (value). You can configure frequency control policies for an ad by referring the figure below.

  Figure 6-1 HASH solution

Key	key TTL	Field	Value
User_1	one day	AD_1	1
		AD_2	2
		AD_3	1
User_2	...	...	...
...	...	...	...

  Hash Solution 1

Key	key TTL	Field	Value
User_1	one week	AD_1	1#one day
		AD_2	2#8 hours
		AD_3	1#one week
User_2	...	...	...
...	...	...	...

  Hash Solution 2

Key	key TTL	Field	Value	field T
User_1	one week	AD_1	1	one day
		AD_2	2	8 hours
		AD_3	1	one week
User_2	...	...	...	...
...	...	...	...	...

  EXHASH Solution

  - HASH solution 1 makes it easy to implement frequency control. You can use the EXPIRE command to set the expiration time of **User_1** to one day and the HINCRBY command to increase how many times an ad is pushed. Before each ad push, you can use HGET to obtain this number. Data is

retained for one day and then is automatically deleted. However, you can set only one expiration time for each user (each key). Flexible frequency control policies cannot be set for a specified period, for example, three pushes within eight hours.

- HASH solution 2 allows you to splice timestamps into the value and use the HASH command to implement flexible frequency control. However, it increases the workload of service development.
- The EXHASH solution allows you to set the expiration time for fields. In the frequency control scenario, GeminiDB Redis API allows you to configure a different push frequency for each ad and in different time segments. Assume that the frequency control policy configured for **AD_2** is twice within 8 hours. Before pushing **AD_2** to **User_1**, you can obtain the value **2** by running the EXHGET command, **AD_2** will not be pushed to **User_1** any more. After eight hours, the field for **User_1** expires and the field information cannot be obtained by EXHGET. In this case, **AD_2** will be pushed to **User_1** again.

- Shopping cart

  The following describes and compares several Redis solutions in the shopping cart scenario.

  a. STRING solution

     The shopping cart works easily with STRING commands. The platform combines the user ID and item ID as a key, for example, **User_1#Earphones_1**. The key value is the number of items to be purchased. There is an expiration time for items on flash sales.

     Figure 6-2 STRING solution

- Related commands

```
incrby User_N#Product_N [Number] #Increases the product quantity.
set User_N#Product_N [Number] #Sets the product quantity.
expire User_N#Product_N Time_N # Sets the expiration time of a specified item in the shopping cart of a specified user.
get User_N#Product_N #Obtains the product quantity.
scan 0 match User_N* # Queries all items of User_N.
del User_N#Product_N #Deletes a specified item from the shopping cart of a specified user.
```

- Possible issues are as follows:
  - Extra splicing increases the encoding and decoding development workload.
  - When a user requests to obtain the shopping item list, the SCAN command prefix is used to scan all keys and the GET command is used to obtain the corresponding value.
  - To obtain the list length, the number of prefix keys should be scanned.
  - There are a large number of duplicate username prefixes occupying the storage space.

b. HASH solution

This solution uses the user ID as the key and the item ID as the field. The value indicates the number of the item in the shopping cart. For items in flash sales, the expiration time is combined to the value of the field.

Figure 6-3 HASH solution

Key	Field	Value
User_1	Earphones_1	1
	Keyboard_1	2
	Charger_2	1#time_1
User_2	...	...
...	...	...

HASH

- Related commands

```
hset User_N Product_N [Number#Time_N] # Sets the quantity and expiration time of a specified item in the shopping cart of a specified user.
hincrby User_N Product_N [Number] # Adds the number of a specified item to the shopping cart of a specified user.
hgetUser_N Product_N # Obtains information about a specified item in the shopping cart of a specified user.
hgetall User_N #Obtains all item information of a specified user.
hlen User_N # Obtains the number of items in the shopping cart of a specified user.
hdel User_N Product_N #Deletes a specified item from the shopping cart of a specified user.
```

- This solution is better than the STRING solution in the following ways:
  - Only one HGETALL command is required to obtain the shopping cart list of a user.
  - The HLEN command can be used to obtain the item list length of a user.
  - There are few duplicate username prefixes.

  However, the solution is complex for processing items in flash sales. For example, If the quantity for **Keyboard_1** of **User_1** needs to be added instead of using the HINCRBY command directly, you should obtain the value of **Keyboard_1** by executing the HGET command first and decode the value. Then, specify the quantity to be added and encode the value using HSET.

c. EXHASH solution

   Similar to the HASH solution, the user ID is used as the key, the item ID is used as the field, and the value is the number of the item in the shopping cart. This solution allows you to set an expiration time for items in flash sales by running the HSET command.

Figure 6-4 EXHASH solution

- Related commands

  ```
  exhset User_N Product_N ex Time_N # Sets the quantity and expiration time of a specified item in the shopping cart of a specified user.
  exhincrby User_N Product_N [Number] keepttl # Increases the number of specified items in the shopping cart of a specified user and retain the original expiration time. exhget User_N Product_N #Obtains information about specified items in the shopping cart of a specified user.
  exhgetall User_N #Obtains all item information of a specified user.
  exhlen User_N # Obtains the number of items in the shopping cart of a specified user.
  exhdel User_N Product_N #Deletes a specified item from the shopping cart of a specified user.
  del User_N #Empties the shopping cart of a specified user.
  ```

- Compared with the HASH solution, this solution allows you to set the expiration time for each field. The EXHASH command makes it easy to use and reconstruct on the service side.

Code Example for Ad Frequency Control

```
import redis
import datetime
import os
def get_cur_time():
    return "[" + datetime.datetime.utcnow().strftime('%Y-%m-%d %H:%M:%S.%f')[:-3] + "]"
def get_redis():
    """
This method is used to connect to a Redis instance.
    * host: Instance connection address.
    * port: Port of the instance. The default value is 6379.
    * password: Password for connecting to the instance.
    """
    # There will be security risks if the username and password used for authentication are directly written into code. Store the username and password in ciphertext in the configuration file or environment variables.
    # In this example, the username and password are stored in the environment variables. Before running this example, set environment variables EXAMPLE_USERNAME_ENV and EXAMPLE_PASSWORD_ENV as
```



```
needed.
    password = os.getenv('EXAMPLE_PASSWORD_ENV')
    return redis.Redis(host='***', port=6379, password=password)
''Global frequency control policy. Display ad 1 for up to 2 times within three seconds and ad 2 for five times
within five seconds.
frequency_strategy = {"ad_1" : [2, 3], "ad_2" : [5, 5]}
def push_ad_to_user(userId: str, adId: str):
    '''     This method is used to push a specified ad to a specified user.
    * userId: User ID.
    * adId: Ad ID.
    '''
    # If no frequency control policies are set for an ad, directly push the ad to the user.
    if adId not in frequency_strategy:
        print("no need control frequency, push ", adId, "to", userId)
        return True
    # Obtain how many times an ad is pushed for a user by user ID and ad ID.
    # Command usage: EXHGET key field.
    cnt = get_redis().execute_command("EXHGET " + userId + " " + adId)
    # If an ad has not been pushed to a user, directly push the ad to the user.
    if cnt == None:
        # Command usage: EXHINCRBY key field num [EX time].
        # Usage description: EXHINCRBY User ID Ad ID Push times (1) Expiration time of the ad
        cmd = "EXHINCRBY " + userId + " " + adId + " 1 EX " + str(frequency_strategy[adId][1])
        cur_cnt = get_redis().execute_command(cmd)
        print(get_cur_time(),"push", adId, "to", userId, "first time during", str(frequency_strategy[adId][1]),
"seconds")
        return True
    # The result returned from Redis Python client is in bytes. Convert the result to a string and then to an
integer.
    cnt = int(cnt.decode("utf-8"))
    if cnt < frequency_strategy[adId][0]:
        # Command usage: EXHINCRBY key field num KEEPTTL Retain the original expiration time of the field.
        cmd = "EXHINCRBY " + userId + " " + adId + " 1 KEEPTTL"
        cur_cnt = get_redis().execute_command(cmd)
        print(get_cur_time(), "push", adId, "to", userId, "current cnt:", cur_cnt)
        return True
    print(get_cur_time(), "Control frequency, can't push", adId, "to", userId, ", max cnt:",
frequency_strategy[adId][0])
    return False
if __name__ == "__main__":
    for i in range(3):
        push_ad_to_user("usr_1", "ad_1")
    for i in range(6):
        push_ad_to_user("usr_1", "ad_2")
    for i in range(3):
        push_ad_to_user("usr_1", "ad_1")
    for i in range(12):
        push_ad_to_user("usr_1", "ad_2")
```

The script output is as follows:

The Python script executes slowly, and the expiration time of ad 2 is set to 5 seconds. Ad 2 can thus be successfully pushed to the user at December 15, 2023 07:09:56.530, 5 seconds after the first push time of December 15, 2023 07:09:51.349.

```
[2023-12-15 07:09:50.086] push ad_1 to usr_1 first time during 3 seconds
[2023-12-15 07:09:50.503] push ad_1 to usr_1 current cnt: 2
[2023-12-15 07:09:50.794] Control frequency, can't push ad_1 to usr_1 , max cnt: 2
[2023-12-15 07:09:51.349] push ad_2 to usr_1 first time during 5 seconds
[2023-12-15 07:09:51.745] push ad_2 to usr_1 current cnt: 2
[2023-12-15 07:09:52.128] push ad_2 to usr_1 current cnt: 3
[2023-12-15 07:09:52.889] push ad_2 to usr_1 current cnt: 4
[2023-12-15 07:09:53.417] push ad_2 to usr_1 current cnt: 5
[2023-12-15 07:09:53.632] Control frequency, can't push ad_2 to usr_1 , max cnt: 5
[2023-12-15 07:09:54.120] push ad_1 to usr_1 first time during 3 seconds
[2023-12-15 07:09:54.769] push ad_1 to usr_1 current cnt: 2
[2023-12-15 07:09:54.915] Control frequency, can't push ad_1 to usr_1 , max cnt: 2
[2023-12-15 07:09:55.211] Control frequency, can't push ad_2 to usr_1 , max cnt: 5
```



```
[2023-12-15 07:09:55.402] Control frequency, can't push ad_2 to usr_1 , max cnt: 5
[2023-12-15 07:09:55.601] Control frequency, can't push ad_2 to usr_1 , max cnt: 5
[2023-12-15 07:09:55.888] Control frequency, can't push ad_2 to usr_1 , max cnt: 5
[2023-12-15 07:09:56.087] Control frequency, can't push ad_2 to usr_1 , max cnt: 5
[2023-12-15 07:09:56.530] push ad_2 to usr_1 first time during 5 seconds
[2023-12-15 07:09:57.133] push ad_2 to usr_1 current cnt: 2
[2023-12-15 07:09:57.648] push ad_2 to usr_1 current cnt: 3
[2023-12-15 07:09:58.107] push ad_2 to usr_1 current cnt: 4
[2023-12-15 07:09:58.623] push ad_2 to usr_1 current cnt: 5
[2023-12-15 07:09:58.865] Control frequency, can't push ad_2 to usr_1 , max cnt: 5
[2023-12-15 07:09:59.096] Control frequency, can't push ad_2 to usr_1 , max cnt: 5
```

This section describes the features, usage, and application scenarios of the EXHASH command provided by GeminiDB Redis API. EXHASH provides a similar syntax to the native Redis HASH and the use is isolated from that of HASH. It allows you to specify expiration times and version numbers for fields. GeminiDB Redis API is dedicated to developing more easy-to-use enterprise-class features, helping customers implement easy O&M and efficiently develop services.

6.5 GeminiDB Redis API for Instant Messaging

Context

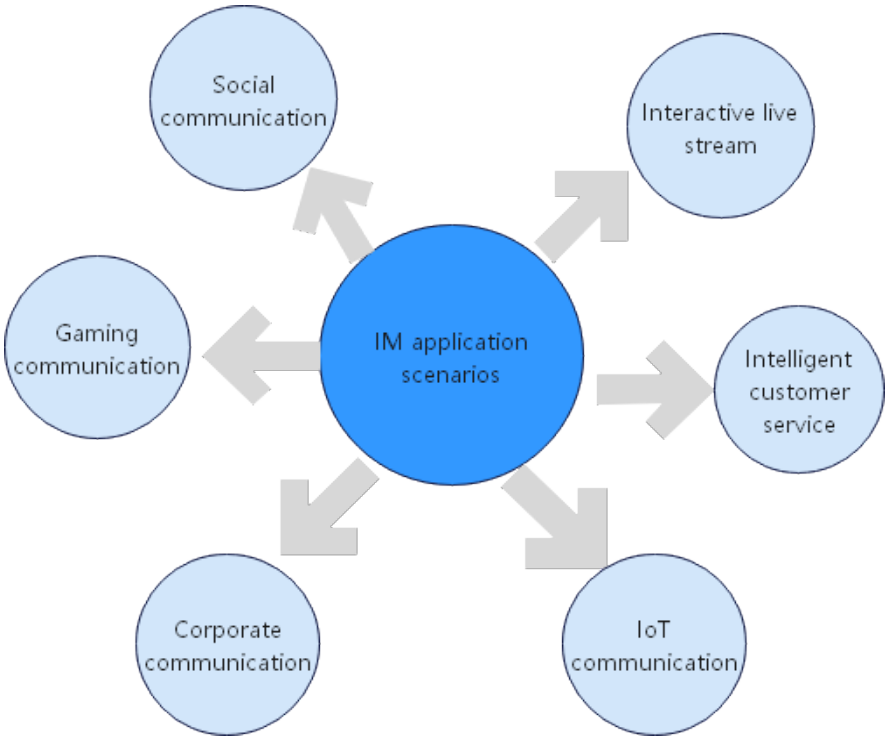
Instant messaging (IM) works by connecting two or more people through a messaging platform over a network. Once connected, users can send text messages, files, even make voice and video calls. In the highly information-based mobile Internet era, IM products (such as WeChat and QQ) have become a must-have item in our life. The core of an IM system is a messaging system, which is used for synchronization, retrieval, and storage of messages.

- Message synchronization: Transmitting integrate messages from the sender to the recipient quickly. The most important metrics of a message synchronization system are instantaneity, sequentiality, and integrity of transmitted messages, and the size of messages that can be supported.
- Message storage: The persistent storage of messages. Conventional message systems store messages on premises on a client, and data is not reliable. Modern message systems store messages on the cloud. This is the so-called "message roaming". You can log in to your account at any terminals to view all historical messages.
- Message retrieval: Messages are generally text. Therefore, full-text retrieval is also a mandatory capability. Conventional message systems usually create indexes based on local messages and support local retrieval. Modern message systems support online message storage and index creation while data is stored, providing comprehensive retrieval functions.

Application Scenarios

IM systems can be used in many industries, such as chatting, gaming, and intelligent customer service. Different industries have different requirements on the cost, performance, reliability, and latency of IM systems. These requirements need to be considered to achieve balance in architecture design.

Figure 6-5 IM application scenarios

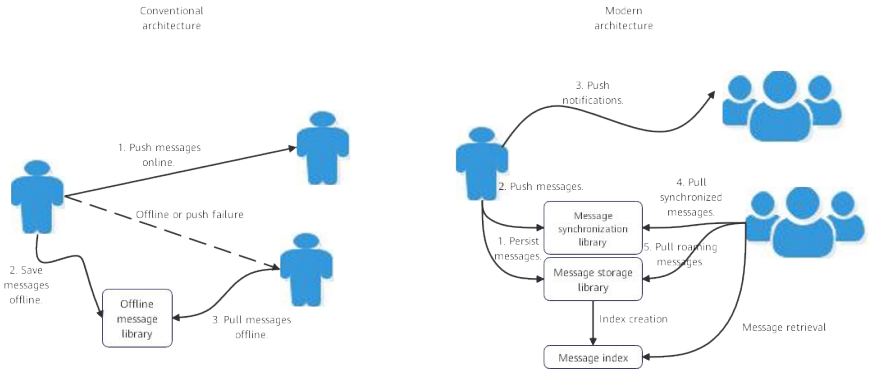


IM System Architecture

The basic concepts involved in IM system architecture design are as follows .

- **Comparison between conventional and modern architectures**

  Figure 6-6 Comparison between conventional and modern architectures



  Conventional architecture:

  – Messages are synchronized before being stored.
  – Messages are synchronized online and cached offline.
  – Servers do not persist messages or support message roaming.

  Modern architecture:

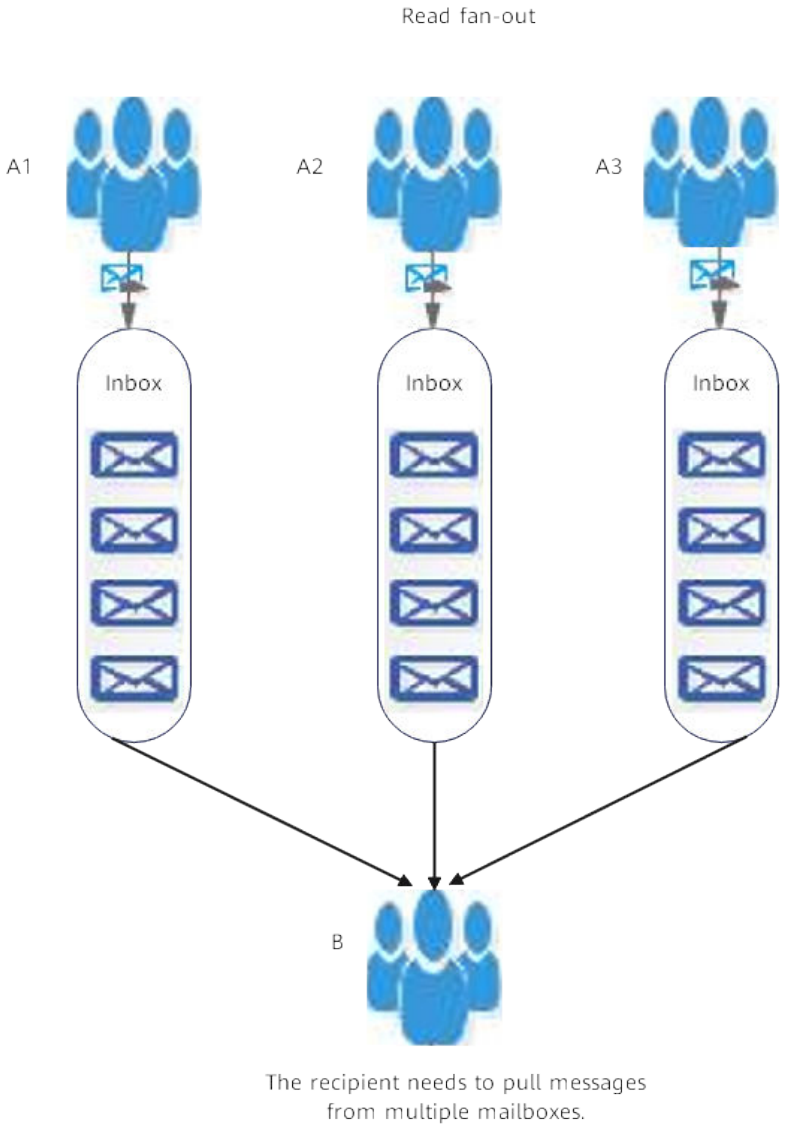
  – Messages are stored before being synchronized.

  - Messages are stored and synchronized in different libraries. The storage library stores all conversations and supports message roaming. The synchronization library stores synchronized messages by receiver.
  - Full-text retrieval is supported.

- **Comparison between read fan-out and write fan-out**

  A suitable read/write model ensures message reliability and consistency and effectively reduces workloads of servers or clients, which is critical to an IM system. This section describes two models: read fan-out and write fan-out.

  Figure 6-7 Read fan-out



  Messages from users A1, A2, and A3 are stored in three different mailboxes (an abstract data structure used to store messages) of user B. User B has to read new messages from all the mailboxes every time. In read fan-out mode, every two associated users have a mailbox.

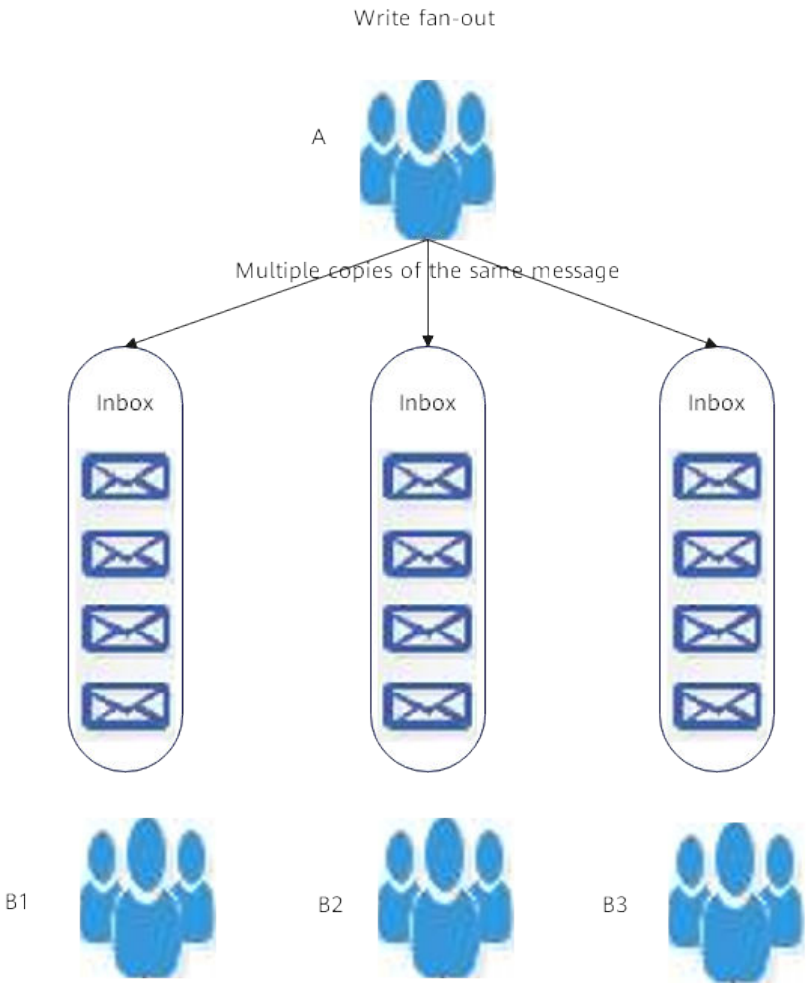
  Advantages of read fan-out:

- No matter a one-on-one chat or a group chat is initiated, messages need to be written into recipient's mailbox once.
- Each mailbox stores two users' chat records, which can be easily viewed and searched for.

Disadvantages of read fan-out:

- As the volume of read operations increases, the system may face challenges in scaling to handle the load efficiently.

Figure 6-8 Write fan-out



Users B1, B2, and B3 read messages only from their own mailboxes. They write or send messages in different ways for one-on-one chat and group chats.

- One-to-one chat: A message is written into both a sender's and a recipient's mailboxes. To view the chat history, another message needs to be written.
- Group chat: A sender needs to write a message to mailboxes of all group members. The group chat works in write fan-out mode, which consumes enormous resources. Therefore, a WeChat group can hold a maximum of 500 members.

Advantages of write fan-out:

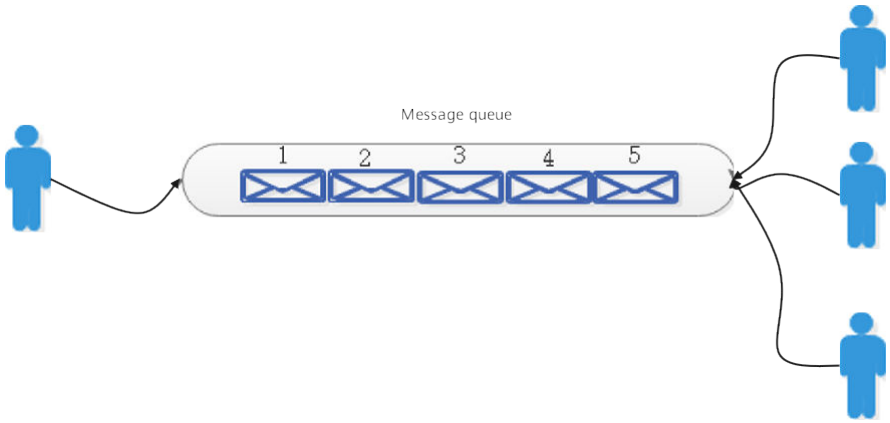
  - Users only need to read their own mailboxes.
  - It is convenient to synchronizing messages between multiple terminals.

  Disadvantages of write fan-out:

  - The system is subjected to heavy write loads, especially for group chats.

- **Comparison among the push, pull, and push-pull modes**

  Figure 6-9 Push, pull, and push-pull modes

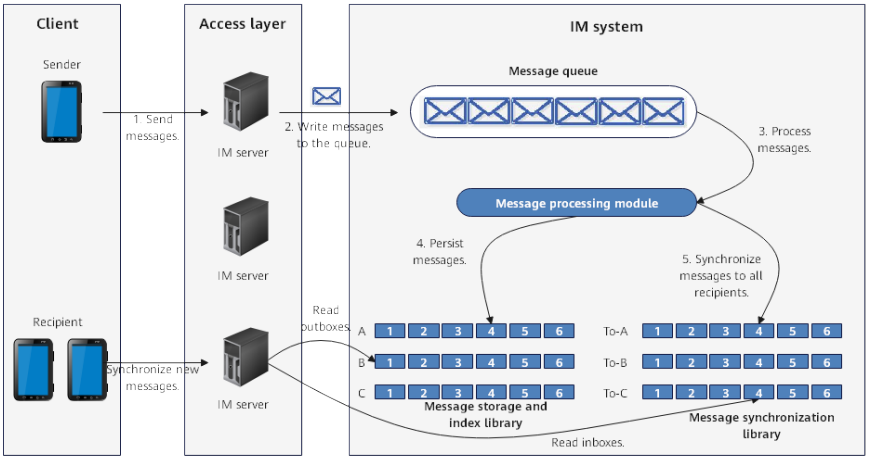
  

  In the IM system, messages can be obtained in the following modes:

  - Push: The server instantly pushes a new message to all clients. A persistent connection needs to be established between the client and the server to ensure real-time performance. The client only needs to receive and process the message. However the server does not know the message processing capability of the client, which may cause a data backlog.
  - Pull: The client requests messages from the frontend. This mode is used to obtain historical messages. The interval for the client to obtain new messages is not preset. If the interval is too short, a large number of connections may fail to obtain data. If the interval is too long, data cannot be received in time.
  - Push-pull: This hybrid mode integrates advantages of push and pull systems. The server pushes a new message notification to the frontend. After receiving the notification, the frontend pulls the message from the server.

IM Technology Challenges

Figure 6-10 IM system architecture



Messages between the clients are forwarded by servers. Core functions of IM are implemented by the message storage and synchronization libraries, which have high requirements on storage layer performance.

- Massive data storage: If messages need to be stored permanently, the data volume will grow gradually. The message storage library must support unlimited capacity expansion to cope with the increasing data volume.
- Low storage cost: Messages contain both hot and cold data. Hot data is generated in most queries. The cold tier has lower storage costs against increasing data volumes.
- Data life cycle management: The life cycle must be defined for message data storage and synchronization. The storage library stores data online. Generally, a long retention duration needs to be specified. The synchronization library is used for online or offline push in the write fan-out mode, and data is stored for a short period.
- High write throughput: The write fan-out mode is used in most IM systems, so storage hardware must offer enhanced write throughput to cope with message floods.
- Low-latency read: The messaging system is usually used online with high real-time performance. The read latency must be as low as possible.

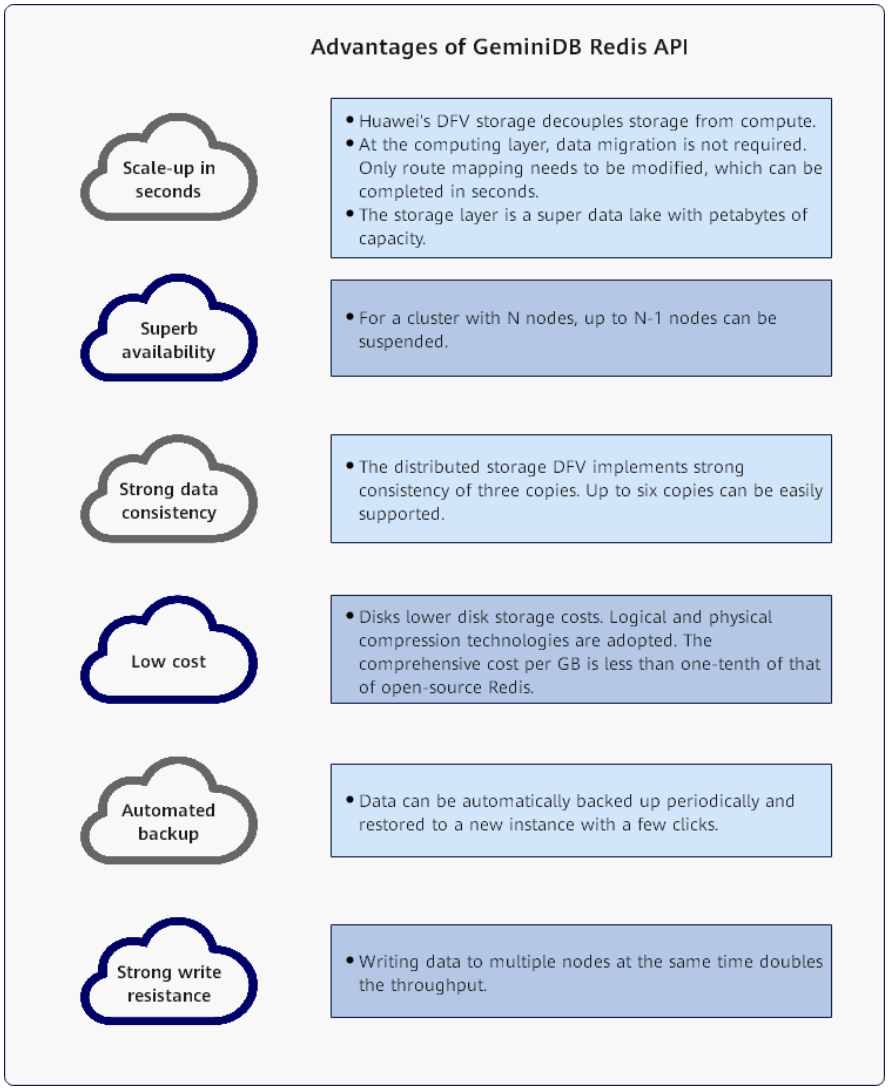
Advantages of GeminiDB Redis in IM Scenarios

At the heart of the IM system lies the storage layer, whose performance directly affects user experience. Currently, there are many database products at the storage layer, such as HBase and open-source Redis, which can be selected based on the business scale, cost, and performance. GeminiDB Redis API is an in-house NoSQL database service. It can meet strict requirements of IM systems on the storage layer in terms of performance and scale, including massive data storage, low storage cost, lifecycle management, high write throughput, and low read latency.

With a cloud native distributed architecture, GeminiDB Redis API is compatible with Redis 5.0 and adopts decoupled storage and compute. In-house storage

systems ensure unlimited capacity expansion, strong consistency, and high reliability. The compute layer leverages LSM-based storage engines. A large number of random writes are converted into sequential writes, which greatly enhances write performance. In addition, read performance is greatly improved by read caches and Bloom filters.

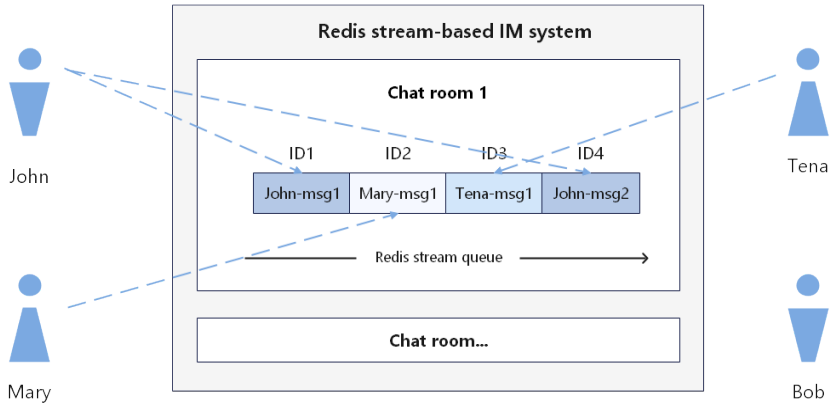
Figure 6-11 Advantages of GeminiDB Redis API



Application Cases of GeminiDB Redis API in IM Scenarios

The following figure shows an IM system based on GeminiDB Redis API. A stream is used as a basic data structure. A Redis stream acts as a message container and allows data exchange between producers and consumers. A Redis stream provides basic functions of IM systems, such as message subscription, distribution, and adding consumers. Users can quickly build an IM system using GeminiDB Redis API. When a group chat is created, a stream queue is also created for the group chat on a GeminiDB Redis instance. Each sender adds messages to the stream queue in time sequence. A stream is a persistent queue that ensures no information loss.

Figure 6-12 Redis stream-based IM system



GeminiDB Redis API uses a series of innovative technologies to improve read and write performance, scale up storage in seconds, and automatically back up data. A GeminiDB Redis API offers a storage layer of the IM system. Its excellent read and write performance and advanced features will greatly facilitate IM applications. In addition, GeminiDB Redis API balances performance and costs based on open-source Redis and can be widely used in fields such as smart healthcare, traffic control, and counter.

7 Performance White Paper

7.1 General Performance Data

7.1.1 Performance Test Methods

This section describes performance testing of GeminiDB Redis instances, including the test environment, tools, metrics, models, and procedure.

Test Environment

- Region: CN-Hong Kong
- AZ: AZ1
- Elastic Cloud Server (ECS): c6.4xlarge.2 with 16 vCPUs, 32 GB of memory, and CentOS 7.5 64-bit image
- Nodes per instance: 3
- Instance specifications: Specifications described in **Table 7-1**

Table 7-1 Instance specifications

No.	Specifications
Cluster 1	4 vCPUs x 3 nodes
Cluster 2	8 vCPUs x 3 nodes

Test Tool

This test used a multi-thread load test tool, memtier_benchmark, developed by Redis Labs. For details about how to use this tool, see **memtier_benchmark**. The following describes some functions of memtier_benchmark.


```
Usage: memtier_benchmark [options]

A memcache/redis NoSQL traffic generator and performance benchmarking tool.

Connection and General Options:
  -s, --server=ADDR                  Server address (default: localhost)
  -p, --port=PORT                    Server port (default: 6379)
  -a, --authenticate=PASSWORD        Authenticate to redis using PASSWORD
  -o, --out-file=FILE                Name of output file (default: stdout)

Test Options:
  -n, --requests=NUMBER              Number of total requests per client (default: 10000)
  -c, --clients=NUMBER               Number of clients per thread (default: 50)
  -t, --threads=NUMBER               Number of threads (default: 4)
      --ratio=RATIO                  Set:Get ratio (default: 1:10)
      --pipeline=NUMBER              Number of concurrent pipelined requests (default: 1)
      --distinct-client-seed         Use a different random seed for each client
      --randomize                    Random seed based on timestamp (default is constant value)

Object Options:
  -d --data-size=SIZE                Object data size (default: 32)
  -R --random-data                   Indicate that data should be randomized

Key Options:
  --key-prefix=PREFIX                Prefix for keys (default: memtier-)
  --key-minimum=NUMBER               Key ID minimum value (default: 0)
  --key-maximum=NUMBER               Key ID maximum value (default: 10000000)
```

Test Metrics

Table 7-2 Test metrics

Metric Abbreviation	Description
QPS	Number of read and write operations executed per second.
Avg Latency	Average latency of read and write operations, in milliseconds.
p99 Latency	• p99 latency of read and write operations. • 99% of operations can be completed within this latency. Only 1% of operations have a latency longer. • Unit: ms.

Test Models

- Workload model

Table 7-3 Workload models

Workload Model	Description
100% Write	100% write operations (string set)

Workload Model	Description
100% Read	100% read operations (string get). The even random access model is used in strict performance tests.
50% Read+50% Write	50% read operations (string get) plus 50% write operations (string set)

- Data model

  Table 7-4 Data model description

Data Model	Description
value length	A value in 100 bytes is generated randomly.

Test scenarios

Table 7-5 Test scenario description

Test Scenario	Description
The data volume is less than memory.	All data can be cached in memory.
The data volume is larger than memory.	Some data can be cached in memory, and some data can be accessed from the DFV storage pool.

Test Procedure

Use a DB instance with 3 nodes and 4 vCPUs for each node as an example:

Scenario 1: When the data volume is less than memory, data is written to and read data from the instance respectively and then concurrently, and record OPS, average latency, and P99 latency of each operation. Workload models and methods of testing performance metrics are as follows:

- Workload model: 100% write

  Use 30 threads and 3 client connections for each thread. That is, 100-byte data is written 60,000,000 times on total 90 connections. The data is generated randomly by all the clients using different seeds within the range of [1, 60,000,000]. Based on the specified range of keys, the total size of data written this time is less than the memory of the database cluster.

  ```
  ./memtier_benchmark -s ${ip} -a ${passwd} -p ${port} -c 3 -t 30 -n 1000000 --random-data --randomize --distinct-client-seed -d 100 --key-maximum=60000000 --key-minimum=1 --key-prefix= --ratio=1:0 --out-file=./output_filename
  ```


- Workload model: 100% read

  Use 30 threads and 3 client connections for each thread. That is, data is randomly and concurrently read 60,000,000 times over 90 connections. The key is within the range of [1, 60,000,000].

  ```
  ./memtier_benchmark -s ${ip} -a ${passwd} -p ${port} -c 3 -t 30 -n 1000000 --random-data --randomize --distinct-client-seed --key-maximum=60000000 --key-minimum=1 --key-prefix= --ratio=0:1 --out-file=./output_filename
  ```

- Workload model: 50% read and 50% write

  Use 30 threads and 3 client connections for each thread. That is, data is randomly and concurrently written and read 60,000,000 times over 90 connections. The key is within the range of [1, 60,000,000]. The write-read ratio is 1:1. Based on the specified range of keys, the total size of data written and read this time was less than the memory of the database cluster.

  ```
  ./memtier_benchmark -s ${ip} -a ${passwd} -p ${port} -c 3 -t 30 -n 1000000 --random-data --randomize --distinct-client-seed -d 100 --key-maximum=60000000 --key-minimum=1 --key-prefix= --ratio=1:1 --out-file=./output_filename
  ```

2. Scenario 2: When the data volume is larger than memory of the database cluster, use 30 threads and create 3 client connections for each thread. That is, 100-byte data is concurrently written 20,000,000 times over total 90 connections. The data is generated randomly by all the clients using different seeds within the range of [60,000,001, 780,000,000]. In addition, pipeline parameters were set to speed up data writes. Based on the specified range of keys and total writes, the total size of data written this time was larger than the memory of the database cluster.

```
./memtier_benchmark -s ${ip} -a ${passwd} -p ${port} -c 3 -t 30 -n 20000000 --random-data --randomize --distinct-client-seed -d 100 --key-maximum=780000000 --key-minimum=60000001 --pipeline=100 --key-prefix= --ratio=1:0 --out-file=./output_filename
```

3. When the data volume is larger than memory, data is written to and read from the database cluster respectively and then concurrently, and metrics OPS, average latency, and p99 latency of each operation were recorded. Workload models and methods of testing performance metrics are as follows:

- Test model: 100% write

  Use 30 threads and 3 clients for each thread. That is, 100-byte data is written 500,000 times over total 90 connections. The data is generated randomly by all the clients within the range of [1, 780,000,000].

  ```
  ./memtier_benchmark -s ${ip} -a ${passwd} -p ${port} -c 3 -t 30 -n 500000 --random-data --randomize --distinct-client-seed -d 100 --key-maximum=780000000 --key-minimum=1 --key-prefix= --ratio=1:0 --out-file=./output_filename
  ```

- Test model: 100% read

  Use 30 threads and 3 clients for each thread. That is, data is randomly and concurrently read 500,000 times over 90 connections. The key is within the range of [1, 780,000,000].

  ```
  ./memtier_benchmark -s ${ip} -a ${passwd} -p ${port} -c 3 -t 30 -n 500000 --random-data --randomize --distinct-client-seed --key-maximum=780000000 --key-minimum=1 --key-prefix= --ratio=0:1 --out-file=./output_filename
  ```

- Test model: 50% read and 50% write

  Use 30 threads and 3 clients for each thread. That is, data is written and read 500,000 times over total 90 connections. The data is generated randomly by all the clients within the range of [1, 780,000,000].


```
./memtier_benchmark -s ${ip} -a ${passwd} -p ${port} -c 3 -t 30 -n 500000 --random-data --randomize --distinct-client-seed -d 100 --key-maximum=780000000 --key-minimum=1 --key-prefix= --ratio=1:1 --out-file=./output_filename
```

7.1.2 Performance Test Results

This section describes performance metrics that are tested using different data and workload models in different scenarios. Only performance data of instances with small- and medium specifications is presented. To use higher specifications, you can scale out or up the instances.

- Table 7-6 describes the test data used when the data volume is less than memory.
- Table 7-7 describes the test data used when the data volume is greater than memory.

Table 7-6 Test data

Instance Specifications	Test Model	Workload Model	QPS	Average Latency (ms)	p99 Latency (ms)
4 vCPUs x 3 nodes	value_length = 100 bytes clients = 90	100% write	125590	0.66	1.85
	value_length = 100 bytes clients = 105	100% read	139741	0.62	1.51
	value_length = 100 bytes clients = 90	50% read and 50% write	125620	Read: 0.56 Write: 0.55	Read: 1.32 Write: 1.30
8 vCPUs x 3 nodes	value_length = 100 bytes clients = 128	100% write	216392	0.62	1.92
	value_length = 100 bytes clients = 128	100% read	202970	0.62	1.89
	value_length = 100 bytes clients = 128	50% read and 50% write	212052	Read: 0.63 Write: 0.63	Read: 1.94 Write: 1.92

Table 7-7 Test data

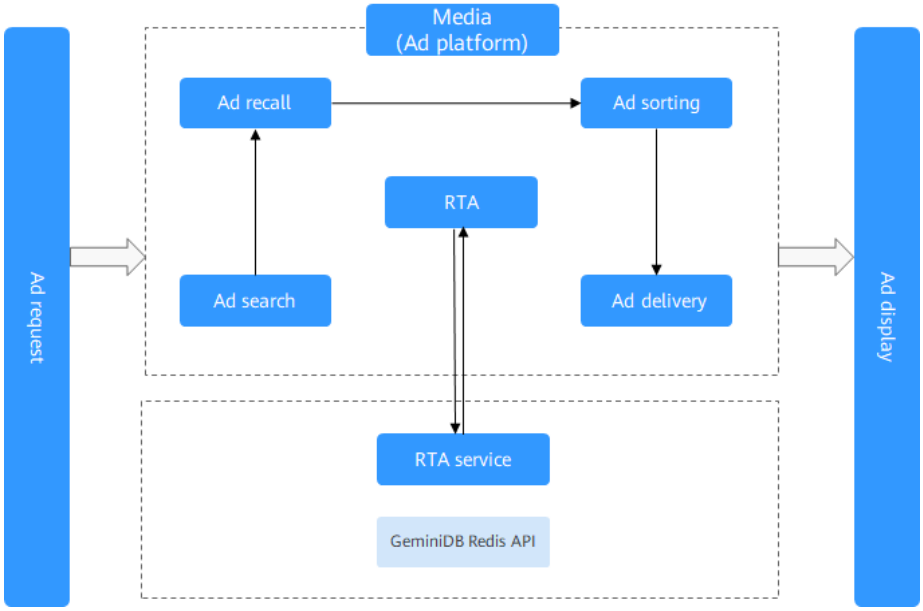
Instance Specifications	Test Model	Workload Model	QPS	Average Latency (ms)	p99 Latency (ms)
4 vCPUs x 3 nodes	value_length = 100 bytes clients = 75	100% write	123942	0.62	1.30
	value_length = 100 bytes clients = 96	100% read	125351	0.63	1.54
	value_length = 100 bytes clients = 96	50% read and 50% write	122485	Read: 0.64	Read: 1.65
				Write: 0.64	Write: 1.61
8 vCPUs x 3 nodes	value_length = 100 bytes clients = 120	100% write	196596	0.62	2.02
	value_length = 100 bytes clients = 120	100% read	187716	0.62	1.90
	value_length = 100 bytes clients = 120	50% read and 50% write	197097	Read: 0.62	Read: 1.94
				Write: 0.62	Write: 1.94

clients indicates the number of connections, which is the product of fields **t** and **c** in the **memtier** command.

7.2 Performance Data in RTA Scenarios

Real-time API (RTA)is a core technology for placing refined ads in real-time. As shown in the following figure, the ad platform asks advertisers first whether they want to participate in the bidding, then searches and recalls ads after receiving the bid responses.

Figure 7-1 RTA process



7.2.1 Performance Test Methods

Objectives

RTA-based advertising poses higher technical requirements for advertisers, including quick response from the media and lower costs in data storage. In recent years, GeminiDB Redis API is widely used as a key-value (KV) signature database in RTA scenarios and delivers good performance at low costs.

This section describes a pressure test of GeminiDB Redis instances in RTA scenarios, including the performance on data compression, QPS, bandwidth, and latency.

Test Environment

This test used GeminiDB Redis clusters and Elastic Cloud Servers (ECSs). The following table lists the specifications.

- GeminiDB Redis cluster specifications

Region	CN East-Shanghai1
AZ type	Deployment across AZ 1, AZ 2, and AZ 3
vCPUs of nodes	16
Nodes	20
Total storage space	2 TB

- ECS specifications

AZ type	AZ 1

Specifications	c7.4xlarge.2, 3 PCS
vCPUs	16
Memory	32 GiB
Operating System (OS)	CentOS 8.2 64-bit

Test Tool

This test used memtier_benchmark, which is a multi-thread load test tool developed by Redis Labs. For details, see **memtier_benchmark**.

Test Metrics

Service scale of the simulated RTA scenario: 1 TB of data, 1.6 million QPS, and 1.5 Gbit/s of bandwidth.

1. Data samples

   Categories of data samples are as follows.

Category	Key	Value
Hash	34 characters	10 field-value pairs. A field contains 10 characters and a value contains 20 to 80 characters.
String	68 characters	32 random characters
String	19 characters	500 to 2,000 random characters

   Four billion keys need to be stored in the Redis clusters. The proportion of each data category is about 2:7:1, and the frequently accessed data accounts for 50% of the total.

2. Metrics

   Test metrics of database operations are as follows.

Metric Abbreviation	Description
QPS	Number of requests executed per second.
Avg Latency (ms)	Average request latency, indicating the overall performance of a GeminiDB Redis cluster.
p99 Latency (ms)	p99 latency of a request, indicating that 99% of the request execution time is shorter than the value of this parameter.

p9999 Latency (ms)	p9999 latency of a request, indicating that 99.99% of the request execution time is shorter than the value of this parameter.

Test Procedure

1. Inject test data.

   Before the test, generate and inject test data. Configure the three categories of data as follows:

   a. Hash

      - A key consists of 34 characters in the format of string prefix + nine digits. The digits are consecutive from 100 million to 900 million. The key is used to control the total data volume and hot data distribution.
      - Inject 10 field-value pairs. A field contains 10 characters and a value contains 20 to 80 random characters. The average value of a field-value is 50 characters.
      - Construct and inject 800 million keys.

        ```
        memtier_benchmark -s ${ip} -a $(passwd} -p ${port} -c 20-t20 -n7500000 -d 32 -key-maximum=3
        800000000 -key-minimum =1000000000 --key-pr efix ='cefkljrithuir123894873h4523blj4b2jkjh2iw13b
        nfdhsbnkfhsdjkh' --key-pattern=P:P--ratio=1:0 -pipelire=100
        ```

   b. String

      - A key consists of 68 characters in the format of string prefix + 10 digits. The digits are consecutive from 1 billion to 3.8 billion. The key is used to control the total data volume and hot data distribution.
      - Inject 32 random characters for a value.
      - Construct and inject 2.8 billion keys.

        ```
        memtier_benchmark -s ${ip} -a ${passwd} -p ${port} -c 20 -t 20 -n 2500000 --command='hset __key__ mendke398d __data__ mebnejkehe __data__ fmebejdbnf __data__ j3i45u8923 __data__ j43245i908 __data__ jhiriu2349 __data__ 21021034ji __data__ jh23ui45j2 __data__ jiu5rj9234 __data__ j23io45u29 __data__' -d 50 --key-maximum=900000000 --key-minimum=100000000 --key-prefix='ewfdjkff43ksdh41fuihikucl' --command-key-pattern=P --pipeline=100
        ```

   c. String

      - A key consists of 19 characters in the format of string prefix + 9 digits. The digits are consecutive from 100 million to 300 million. The key is used to control the total data volume and hot data distribution.
      - Inject 500 to 2,000 random characters for a value. The average value is 1,250 bits.
      - Construct and inject 400 million keys.

        ```
        memtier_benchmark -s ${ip} -a ${passwd} -p ${port} -c 20 -t 20 -n 520000 -d 1250 --key-maximum=300000000 --key-minimum=100000000 --key-prefix='miqjkfdjiu' --key-pattern=P:P --ratio=1:0 --pipeline=100
        ```


After data is injected, there were 3,809,940,889 (about 3.8 billion) keys. Obtain the total data volume on the GeminiDB Redis API console and calculate the data compression ratio. The compressed storage space was 155 GB, and the compression ratio was 13.8%.

CAUTION

- About 4 billion data records were generated by memtier_benchmark of the current version. Data distribution among different categories is not affected.
- A random character string constructed by memtier_benchmark contains many consecutive characters, so the compression ratio was low. The data compression ratio is about 30% to 50% in actual production.

2. Pressure test commands

   Perform pressure tests on GeminiDB Redis clusters deployed on three ECSs, separately. The pressure test tasks are as follows:

   a. On ECS 1, run the HGETALL command for hashes and set a range for keys to allow access to hot data only.

   ```
   memtier_benchmark -s ${ip} -a ${passwd} -p ${port} -c 20 -t 30 --test-time 1200 --random-data --randomize --distinct-client-seed --command='hgetall __key__' --key-maximum=600000000 --key-minimum=200000000 --key-prefix='ewfdjkff43ksdh41fuihikucl' --out-file=./output_filename
   ```

   b. Run the GET command for type data 2 and set a range for keys to allow access to hot data only.

   ```
   memtier_benchmark -s ${ip} -a ${passwd} -p ${port} -c 70 -t 30 --test-time 1200 --random-data --randomize --distinct-client-seed --key-maximum=2400000000 --key-minimum=1000000000 --key-prefix='cefkljrithuin123894873h4523bhj4b2jkjh2iu13bnfdhsbnkfhsdjkh' --ratio=0:1 --out-file=./output_filename
   ```

   c. Run the GET command for type data 3 and set a range for keys to allow access to hot data only.

   ```
   memtier_benchmark -s ${ip} -a ${passwd} -p ${port} -c 10 -t 30 --test-time 1200 --random-data --randomize --distinct-client-seed --key-maximum=300000000 --key-minimum=100000000 --key-prefix='miqjkfdjiu' --ratio=0:1 --out-file=./output_filename
   ```

   The number of connections (the product of **c** and **t**) was adjusted to modify the number of clients and configuration of each instance, so as to achieve a QPS of 1,600,000 and a read request traffic of 1.5 Gbit/s. Remain the service volume unchanged and evaluate the performance of GeminiDB Redis API.

7.2.2 Performance Test Results

Over 1 TB of data was injected for pressure testing. The test results are as follows:

- Data compression ratio

  1.1 TB of data (about 3.8 billion data records) is written and the occupied space is 155 GB after compression. The data compression ratio is about 13.8%.

- Performance

  The total QPS reaches 1.6 million, the total read request traffic is 1.5 Gbit/s, and the CPU usage ranges from 60% to 70%.

The average latency is about 0.7 ms, and the p99 long-tail latency is about 1.77 ms.

The results show that GeminiDB Redis API has stable latency in large-scale RTA scenarios. With data compression and decoupled compute and storage architecture, GeminiDB Redis API is an ideal KV database for advertising services.

8 FAQs

8.1 About GeminiDB Redis API

8.1.1 What Are the Differences Between GeminiDB Redis API, Open-Source Redis, and Other Open-Source Redis Cloud Services?

Redis, an open-source in-memory data structure store, is used as a cache broker. GeminiDB Redis API, an enhanced version of open-source Redis, is an elastic KV database compatible with the Redis protocol, supports much larger capacity than memory, and delivers ultimate performance. Hot data is stored in memory, and full data is stored in a high-performance storage pool. GeminiDB Redis API features:

- Low stable latency

  The average single-point read/write latency is shorter than 1 ms, and the P99 latency is shorter than 2 ms. By adopting a multi-thread architecture, GeminiDB Redis API allows for flexible QPS adjustment ranging from 10,000 to 10,000,000.

- High cost-effectiveness

  30% lower comprehensive costs: Because no standby node is required and GeminiDB Redis API offers an ultra-high data compression ratio of 4:1, it is cheaper to scale out storage capacity.

- Higher O&M efficiency

  2 GB to 100 TB more capacity can be added to storage devices without any impact on services. Point-in-Time Recovery (PITR) restores databases up to a specific moment in time.

- More enhanced features for enterprises

  An expiration time can be specified for individual fields in a hash. A Bloom filter can be used. Data can be imported extremely fast. Memory acceleration can be enabled.

For details about the comparison between GeminiDB Redis and open-source self-built KV databases, see **Highlights**.

8.1.2 How Is the Performance of GeminiDB Redis API Compared with Open-Source Redis?

GeminiDB Redis API uses the multi-thread architecture. More CPUs can improve QPS (10,000–10,000,000).

Generally, the average latency of single-point access is less than 1 ms, and the p99 latency is less than 2 ms, similar to that of open-source Redis.

For details about performance data, see **Performance Test Results**.

8.1.3 What Redis Versions and Commands Are Compatible with GeminiDB Redis API? Whether Application Code Needs to Be Refactored for Connecting to a Redis Client?

GeminiDB Redis API is fully compatible with Redis 6.2 (including 6.2.*x*) and earlier versions, such as 5.0, 4.0, and 2.8. It is partially compatible with Redis 7.0.

You can migrate data of Redis 6.2 and earlier instances (such as 5.0, 4.0, and 2.8) to GeminiDB Redis instances, without the need of code modifications. Any Redis client can be connected to GeminiDB Redis instances.

8.1.4 Can Data Be Migrated from a Self-Built Redis Instance to a GeminiDB Redis Instance? What Are the Precautions?

Yes. Before migration, ensure that:

- Version: If the version of the source database is 6.2 or earlier (including 6.2.*x*), data can be directly migrated. If the version of the source database is later than 6.2, you need to evaluate the migration project and then migrate data to GeminiDB Redis 6.2. You can submit a service ticket for consultation.
- Specifications: Configure proper specifications based on QPS and data volumes of the source instance.

8.1.5 What Is the Availability of a GeminiDB Redis Instance?

The formula for calculating the instance availability is as follows:

DB instance availability = (1 – Failure duration/Total service duration) × 100%

The failure duration refers to the total duration of faults that occur during the running of a DB instance after you buy the instance. The total service duration refers to the total running time of the DB instance.

8.1.6 Are Total Memory and Total Capacity of a GeminiDB Redis Instance the Same? What Is the Relationship Between Memory and Capacity?

No.

In an open-source Redis instance, all data is stored in memory, and the total capacity is the amount of memory that can be utilized.

In a GeminiDB Redis instance, all data is stored in a high-performance shared storage pool, and hot data is stored in memory. Generally, you only need to pay attention to the total capacity and usage of the instance. The CPU usage increases as QPS increases. In this case, you need to scale up the specifications.

8.1.7 How Do I Select Proper Node Specifications and Node Quantity When Purchasing a GeminiDB Redis Instance?

When purchasing a GeminiDB Redis instance, pay attention to QPS and data volume. You can select **Fast configure** or **Standard configure** for **Instance Creation Method**.

- **Fast configure**: If 16 GB of storage space is used for a cluster, you can select **16 GB** in the **Instance Specifications** area. If QPS does not meet requirements, select higher specifications.
- **Standard configure**: Select specifications of compute and storage resources separately. The node specifications and number of nodes determine instance QPS, and the total instance capacity determines the maximum storage space. After selecting the node specifications, number of nodes, and total instance capacity, you can view the QPS and number of connections of the selected instance next to **Specification Preview**.

8.1.8 Is a Primary/Standby or Cluster Deployment Mode Preferred for GeminiDB Redis Instances with Several GB of Storage Space?

A cluster is preferred. At least 4 GB of storage space is recommended. Compared with the primary/standby architecture, the cluster has better scalability and higher QPS. The GeminiDB Redis cluster has the following advantages:

- All compute nodes in the GeminiDB Redis cluster support writes and reads, and the node resource utilization is 100%. In the primary/standby mode,

shards on the standby nodes do not support writes, resulting in low resource utilization.

- Both a single node and a cluster can access the GeminiDB Redis cluster, which is a proxy cluster.

8.1.9 How Does GeminiDB Redis API Persist Data? Will Data Be Lost?

Open-Source Redis persists data periodically, so there is a high probability that data loss occurs in abnormal scenarios. GeminiDB Redis API data is updated to a storage pool in real time, improving data security.

Similar to other NoSQL databases, backend processes on the GeminiDB Redis API instance write write-ahead logs (WALs) into OS buffers. The buffers immediately return and then are updated to the storage pool. Therefore, a small amount of data may be lost in the case of an unexpected power failure.

GeminiDB Redis API ensures data is not lost during routine O&M, such as changing specifications, upgrading versions, and adding nodes. Synchronous writes greatly reduce write performance. To achieve higher data reliability, you need to enable synchronous writes. You can choose **Service Tickets > Create Service Ticket** in the upper right corner of the console.

8.1.10 What Is the Memory Eviction Policy of GeminiDB Redis API?

If keys of an open-source Redis instance are evicted from the memory, the key values cannot be read later.

By default, GeminiDB Redis API supports a noeviction policy, that is, user keys are not evicted. All data is stored in a storage pool. Hot data evicted from the memory can be read from the storage pool. The data is reloaded to the memory after being accessed, and user keys are not deleted.

Therefore, GeminiDB Redis API users do not need to set or modify the **maxmemory-policy** parameter. If unnecessary data is stored, users need to add an expiration time to avoid dramatical increase in data volumes.

8.1.11 Does GeminiDB Redis API Support Modules Such as a Bloom Filter?

A Bloom filter can be used to check whether an element is in a large-size data set. It is suitable for scenarios such as web interceptors and anti-cache penetration.

GeminiDB Redis API supports Bloom filters.

In addition, you can set an expiration time for individual fields in a hash shard. Shards can be scanned in parallel. Data can be imported extremely fast using FastLoad.

8.2 Billing

8.2.1 What Are the Differences Between Yearly/Monthly and Pay-per-use Billing Mode?

Yearly/Monthly is a prepaid billing mode in which resources are billed based on the service duration. This cost-effective mode is ideal when the duration of resource usage is predictable. It is recommended for long-term users.

Pay-per-use is a post payment mode, so you can start or stop an instance at any time. Pricing is listed on a per-hour basis, but bills are calculated based on the actual usage duration.

For details, see **2.2.1 Overview**.

8.2.2 Can I Switch Between Yearly/Monthly and Pay-per-Use Payments?

You can change the billing mode from yearly/monthly to pay-per-use or vice versa.

- For details about how to change the billing mode from yearly/monthly to a pay-per-use, see **2.5.3 Yearly/Monthly to Pay-per-Use**.
- For details about how to change the billing mode from pay-per-use to yearly/monthly, see **2.5.2 Pay-per-Use to Yearly/Monthly**.

8.3 Database Usage

8.3.1 Why Is the Key Not Returned Using Scan Match?

Symptom

As shown in the following figure, the value of key is **test** and exists in the database. However, no data is returned using this scan match command.

```
139.9.177.148:6379> scan 1 match tes*
1) "21"
2) (empty list or set)
139.9.177.148:6379> get test
"abc"
139.9.177.148:6379>scan 0 match tes*
1) "21"
2) (empty list or set)
139.9.177.148:6379>
```

Possible Causes

The MATCH command is used to iterate elements that only match a specified pattern. Pattern matching is performed after the command obtain elements from the data set and before the elements are returned to the client. If all the extracted elements do not match the pattern, no element is returned.

Solution

If multiple scans are performed, the iteration is complete when the returned cursor is 0. The cursor returned from the last scan is used for the next scan.

8.3.2 How Do I Process Existing Data Shards After Migrating Workloads to GeminiDB Redis API?

GeminiDB Redis API uses decoupled compute and storage and allows adding data shards dynamically, making scaling smooth.

After an GeminiDB Redis instance is connected, data sharding is not required on the service side.

8.3.3 Does GeminiDB Redis API Support Fuzzy Queries Using KEYS?

Yes.

Fuzzy queries using KEYS may cause OOM and longer latency.

KEYS can be used only in a test environment. In a production environment, use SCAN and MATCH instead.

8.3.4 Does the GeminiDB Redis API Support Multiple Databases?

GeminiDB Redis API allows you to create multiple databases in an instance since March 2022. Instances created before March 2022 do not support this function and cannot be upgraded to support it.

This feature has the following constraints:

- The number of databases ranges from 0 to 999.
- The SWAPDB command is not supported.
- The result of the **dbsize** command is not updated in real time. The result does not decrease to 0 immediately after **flushdb** is executed, and will change to 0 after a while.
- Executing SELECT and FLUSHDB commands in LUA scripts is not supported.
- Executing SELECT and FLUSHDB commands in transactions is not supported.
- The MOVE command is not supported.

8.3.5 Why the Values Returned by Scan Operations Are Different Between GeminiDB Redis API and Open-Source Redis 5.0?

GeminiDB Redis API may return values in a different sequence from open-source Redis, but they both comply with open-source document description requirements. This is because open-source Redis does not specify the sorting rules for:

- Returned values of SCAN/HSCAN/SSCAN operations
- Returned values of ZSCAN operations ZSET when its elements have the same score

8.3.6 Why Are Error Messages Returned by Some Invalid Commands Different Between GeminiDB Redis API and Open-Source Redis 5.0?

GeminiDB Redis API checks command syntax and checks for keys each time it executes a command. However, open-source Redis has no specific rules and returns the results for invalid commands in random.

Therefore, error messages returned by some invalid commands may be different.

8.3.7 How Do I Resolve the Error "CROSSSLOT Keys in request don't hash to the same slot"?

Scenarios

When multi-key commands are executed in a GeminiDB Redis instance, the error "CROSSSLOT Keys in request don't hash to the same slot" may be reported.

Error Cause

Commands involving multiple keys were executed across slots in a GeminiDB Redis cluster instance. For example, EVAL and BRPOPLPUSH were executed across slots.

Solution

- Change key names and use hash tags to ensure that the involved keys are in the same slot. Avoid data skew when you use hash tags. For more information, see **8.3.9 Which Commands Require Hash Tags in GeminiDB Redis Cluster Instances?**.
- When hash tags cannot be used, change the instance type to primary/standby. For details, see **1.5 Compatible APIs and Versions**.

8.3.8 How Many Commands Can Be Contained in a GeminiDB Redis Transaction?

It is recommended that a transaction contain a maximum of 100 commands.

Exercise caution when using commands with time complexity of O(N).

8.3.9 Which Commands Require Hash Tags in GeminiDB Redis Cluster Instances?

In a GeminiDB Redis cluster instance, if you need to run the following commands that manage multiple keys, use hash tags to when designing key names:

MSETNX, BLPOP, BRPOP, BRPOPLPUSH, RPOPLPUSH, SDIFF, SDIFFSDIFFSTORE, SINTER, SINTERSTORE, SMOVE, SUNION, SUNIONSUNIONSTORE, ZINTERSTORE, ZUNIONSTORE, XREAD, XREADGROUP, PFCOUNT, PFMERGE, GEORADIUS, GEORADIUS_RO, GEORADIUSBYMEMBER, GEORADIUSBYMEMBER_RO, GEOSEARCHSTORE, BITOP, RENAME, RENAMENX, and SORT.

8.3.10 What Do I Do If the Error "ERR Unknown Command Sentinel" Is Displayed?

Scenarios

When **SENTINEL** commands are executed on a GeminiDB Redis instance, the error message "ERR unknown command sentinel" may be displayed.

Error Cause

If the value of **CompatibleMode** of a GeminiDB Redis cluster instance is not **3** or the value of **CompatibleMode** of a primary/standby GeminiDB Redis instance is not **2**, executing **SENTINEL** commands is not allowed.

Solution

Step 1 Log in to the Huawei Cloud console.

Step 2 In the service list, choose **Databases** > **GeminiDB Redis API**.

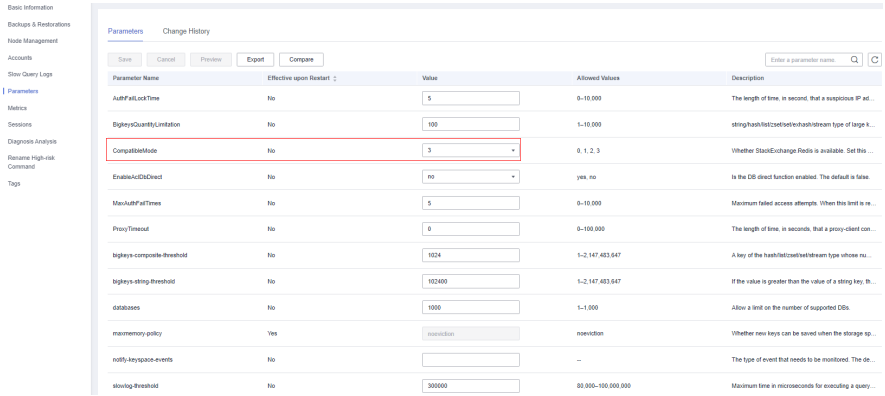
Step 3 On the **Instances** page, click the instance whose specifications you want to change. The **Basic Information** page is displayed.

Step 4 In the navigation pane on the left, choose **Parameters**.

Step 5 Change the value of **CompatibleMode** and click **Save**.

- For a cluster DB instance, change the value of **CompatibleMode** to **3**.
- For a primary/standby DB instance, change the value of **CompatibleMode** to **2**.

Figure 8-1 Changing parameters



----End

8.3.11 Why Return Values of Blocking Commands Differ Between Primary/Standby GeminiDB Redis Instances and Open-Source Redis Instances?

A return value is not specified when a blocking command is executed to block an open-source Redis client until keys are written concurrently.

Although their return values are different, both of them meet requirements described in open-source Redis documentation.

8.3.12 How Long Does It Take to Scale Up GeminiDB Redis Instance Storage? Will Services Be Affected?

GeminiDB Redis instance storage can be scaled up in seconds, and services are not affected.

For details about how to scale storage, see **4.6.7.2 Manually Scaling Up Disk Space**.

For details about automated scale-up, see **4.6.7.3 Automatically Scaling Up Disk Space**.

8.3.13 How Long Does It Take to Add GeminiDB Redis Nodes at the Same Time? What Are the Impacts on Services?

GeminiDB Redis nodes can be added at the same time, which can be completed within 5 minutes.

NOTICE

Shared storage is used. After nodes are added, data does not need to be migrated, but slots are rebalanced. A retry mechanism is needed to avoid service interruptions due to a few seconds of jitter or latency.

8.3.14 What Are the Differences Between Online and Offline Specification Changes of GeminiDB Redis Nodes? How Long Will the Changes Take? What Are the Impacts on Services?

- Online change: Nodes are changed in rolling mode. The change duration is positively related to the number of nodes. Each node takes about 5 to 10 minutes. In addition, both primary/standby and cluster instances contain three internal management nodes, which are changed at the same time. For example, a GeminiDB Redis instance consists of six nodes, including three worker nodes and three internal management nodes. The online change takes about 30 to 60 minutes. While specifications are changed, the node is disconnected, and its slots become disabled and are taken over by a functional node. In addition to node disconnection , there are also other interruptions in several seconds, for example, access timeout and invisible data partitions, so a reconnection mechanism must be established. You are

advised to change the node specifications during off-peak hours and keep the CPU and memory usage at a low level. This prevents exceptions such as heavy load on other nodes and process startup failures.

- Offline change: Specifications of all nodes are changed concurrently. During the change, services are interrupted for about 10 to 20 minutes. Offline change is applicable when services are stopped or no service is accessed. Exercise caution when performing this operation.

For your online production services, you are advised to perform the change online. For details, see **4.6.4 Changing the CPU and Memory Specifications of an Instance**.

8.3.15 What Are the Differences Between Online and Offline Patch Installation of GeminiDB Redis Nodes? How Long Will the Upgrades Take? What Are the Impacts on Services?

- Online patch installation: Nodes are upgraded in rolling mode. The upgrade duration is positively related to the number of nodes. Each node takes about 2 to 5 minutes. Both primary/standby and cluster instances contain three internal management nodes, which are upgraded at the same time. For example, a GeminiDB Redis instance consists of six nodes, including three worker nodes and three internal management nodes. The online upgrade takes about 12 to 30 minutes. During the upgrade of a single node, services are affected due to a few seconds of jitter. Therefore, a reconnection mechanism is required. You are advised to upgrade the node during off-peak hours and keep the CPU and memory usage at a low level. This prevents exceptions such as heavy load on other nodes and process startup failures.
- Offline patch installation: All nodes are upgraded concurrently. During the upgrade, services are interrupted for about 10 to 20 minutes. Offline upgrade is applicable when services are stopped or no service is accessed. Exercise caution when performing this operation.

For details about how to install the patch of GeminiDB Redis API, see **4.6.1 Upgrading a Minor Version**.

8.3.16 Can I Download Backups of a GeminiDB Redis Instance to a Local PC and Restore Data Offline?

Backups of a GeminiDB Redis instance differ from RDB files of an open-source Redis instance and cannot be used by users. Therefore, the backups cannot be downloaded to a local PC.

If instance data is corrupted, you can restore backup data to a new instance.

For details, see **4.7.1 Overview** and **4.8.2 Restoring Data to a New Instance**.

8.3.17 What Is the Data Backup Mechanism of GeminiDB Redis API? What Are the Impacts on Services?

To back up data of a GeminiDB Redis instance, snapshots need be created in seconds only for the storage layer, which does not affect compute nodes. Therefore, services are not affected as well.

When backup data is uploaded, a small amount of CPU and bandwidth resources are consumed, which may cause slight jitter.

GeminiDB Redis instances support automated and manual backup. For details, see **4.7.1 Overview**.

8.3.18 Why Does the CPU Usage Remain High Despite Low Service Access Volume on a GeminiDB Redis Preferential Instance with 1 CPU and 2 Nodes?

GeminiDB Redis API collects metrics and reports monitoring data. The CPU usage of your nodes is high because of its small specifications.

A GeminiDB Redis Preferential instance with one CPU and two nodes is recommended in the test environment. A GeminiDB Redis instance with one CPU (standard) or two or more CPUs is recommended in the production environment.

For details about instance specifications, see **1.6 Instance Specifications**.

8.3.19 Why Does the Number of Keys Decrease and Then Become Normal on the Monitoring Panel on the GUI of GeminiDB Redis API?

The number of keys is scanned and counted asynchronously by a GeminiDB Redis server to ensure final consistency.

When an instance process is restarted (due to node restart, instance fault, specification change, or version upgrade), the keys are counted again. In this case, the number of keys displayed decreases temporarily and becomes accurate gradually.

8.3.20 Why Is CPU Usage of GeminiDB Redis Nodes Occasionally High?

There are many possible reasons, such as sudden spike in service traffic, large key operations, network jitter, data backup and garbage recycle tasks on a server.

If the CPU usage is occasionally high, just ignore it.

If there are other service reasons (excluding high QPS), you can submit a service ticket.

8.3.21 How Do I Upgrade GeminiDB Redis API from 5.0 to 6.2?

GeminiDB Redis API is compatible with Redis 7.0, 6.2 (including 6.2.*x*), 5.0, and earlier versions.

To upgrade an existing instance, submit a service ticket to enable the upgrade trustlist.

For details about the upgrade operations, see **4.6.1 Upgrading a Minor Version**.

8.3.22 When Does a GeminiDB Redis Instance Become Read-Only?

To ensure that the GeminiDB Redis instance can still run properly when the storage space is about to be used up, the database is set to read-only, and data cannot be modified. If this happens, you can scale up the storage to restore the database status to read/write.

Table 8-1 Setting an instance status to read-only

Storage Capacity	Description
< 600 GB	• When the storage usage reaches 97%, the instance is read-only. • When the storage usage decreases to 85%, the read-only status is automatically disabled for the instance.
≥ 600 GB	• When the remaining storage space is less than 18 GB, the instance is read-only. • When the remaining storage space is greater than or equal to 90 GB, the read-only status is automatically disabled for the instance.

8.4 Database Connection

8.4.1 How Do I Connect to a GeminiDB Redis Instance?

You can connect to a GeminiDB Influx instance using a private network, public network, load balancer IP address, DAS, or program code. For details, see **4.3.1 Connection Methods**.

- You can connect to a GeminiDB Redis instance using a **web-based console client**.
- You can connect to a GeminiDB Redis instance through a **private IP address**, **private domain name**, or **load balancer address**.
- You can connect to a GeminiDB Redis instance through a **public domain name** or an **EIP**.
- You can connect to a GeminiDB Redis instance using different code. For details, see **5.3 Examples of Connecting to an Instance Using Programming Languages**.

8.4.2 How Do I Use Multiple Node IP Addresses Provided by GeminiDB Redis API?

GeminiDB Redis API provides multiple IP addresses for you to access a cluster and achieve load balancing and disaster recovery.

You can use multiple IP addresses in any of the following ways.

1. Use the connection pool on the service side implement load balancing and fault detection.
2. Choose **Service Tickets > Create Service Ticket** in the upper right corner of the console. Contact customer service to configure Elastic Load Balance (ELB) and provide a unique IP address.
3. Configure domain names for multiple proxy IP addresses. For details about how to connect to an instance through a private domain name, see **4.3.3.1 Connecting to an Instance Using a Load Balancer Address (Recommended)**.

8.4.3 How Does Load Balancing Work in GeminiDB Redis API?

GeminiDB Redis API uses dedicated load balancers with scalable specifications and supports a maximum bandwidth of 10 Gbit/s. For details, see **Dedicated Load Balancer Overview**.

8.4.4 How Can I Create and Connect to an ECS?

1. To create an ECS, see *Elastic Cloud Server User Guide*.
   - The ECS to be created must be in the same VPC and security group with the GeminiDB Redis instance to which it connects.
   - Configure the security group rules to allow the ECS to access to the instance.
2. To connect to an ECS, see "Logging in to an ECS" *Getting Started with Elastic Cloud Server User Guide*.

8.4.5 Can I Change the VPC of a GeminiDB Redis Instance?

After a GeminiDB Redis instance is created, the VPC where the instance resides cannot be changed.

However, you can change a VPC by restoring the full backup of your instance to the VPC you want to use.

For details, see **4.8.2 Restoring Data to a New Instance**.

8.4.6 Why Can't I Connect to the Instance After an EIP Is Bound to It?

Possible Cause

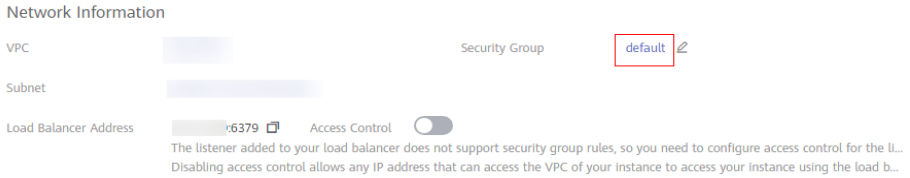
The current port has not been enabled in the inbound rules of the security group.

Handling Procedure

Step 1 Click the instance name to go to the **Basic Information** page.

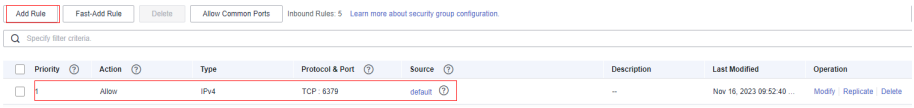
Step 2 In the **Network Information** area, click the value of the **Security Group** parameter.

Figure 8-2 Network information



Step 3 Click the **Inbound Rules** tab and click **Add Rule**. Then, specify parameters as shown in the following figure.

Figure 8-3 Inbound rules



You can also specify parameters by referring to **Configuring Security Group Rules**.

----End

8.4.7 How Do I Access a GeminiDB Redis Instance from a Private Network?

You can access a GeminiDB Redis instance through a load balancer or a directly-connected node.

- Access through a load balancer (recommended): The load balancer is associated with a high-availability backend cluster, using an internal IP address that is accessible only to clients. Periodical health checks are performed on backend nodes to prevent single points of failure (SPOFs).
- Access through a directly-connected node: An agent installed on a GeminiDB Redis node enables you to connect to any node. Then you can access the entire cluster. To prevent SPOFs, this access mode is only recommended in test scenarios.

For details about how to connect to a GeminiDB Redis instance over a private network, see **4.3.3 Connecting to a GeminiDB Redis Instance Over a Private Network**.

8.4.8 Do I Need to Enable Private Network Access Control for a Load Balancer After Setting a Security Group?

You can access a GeminiDB Redis instance through a node or load balancer. Therefore, you need to configure both a security group and private network access control for a load balancer to ensure instance security.

- Security groups take effect only for nodes. It is a collection of access control rules for ECSs and GeminiDB Redis instances that have the same security requirements and are mutually trusted in a VPC. For details, see **4.3.5.3 Configuring Security Group Rules for a GeminiDB Redis Instance**.

- Security groups cannot take effect for load balancers. If access control is disabled, all IP addresses that can access the VPC of the GeminiDB Redis instance also can access the instance using a load balancer IP address. Therefore, you need to configure access control properly. For details, see **4.3.5.9 Configuring Private Network Access to a GeminiDB Redis Instance**.

8.4.9 What Should I Do If the Client Connection Pool Reports Error " Could not get a resource from the pool"?

Scenarios

A large number of GeminiDB Redis instance requests are suspended on the clients, and the following error is displayed in the abnormal stack:

- Jedis client:
  ```
  redis.clients.jedis.exceptions.JedisConnectionException: Could not get a resource from the pool
  ```
- Lettuce client:
  ```
  redis.connection.lettuce.LettucePoolingConnectionProvider.getConnection
  ```
- Go-redis client:
  ```
  redis: connection pool timeout
  You can get that error when there are no free connections in the pool for Options.PoolTimeout duration. If you are using redis.PubSub or redis.Conn, make sure to properly release PuSub/Conn resources by calling Close method when they are not needed any more.
  You can also get that error when Redis processes commands too slowly and all connections in the pool are blocked for more than PoolTimeout duration.
  ```

However, the instance information queried by following **4.13.4 Viewing GeminiDB Redis Instance Metrics** shows that the database QPS, latency, and number of connections are normal and no slow request is displayed.

Possible Causes

Generally, the preceding issue is caused by incorrect configurations of a client connection pool. The maximum number of the connection pools is limited for applications. If the QPS of an application exceeds the limit of the connection pool or connections are not released in time, the thread cannot obtain new connections and services are affected.

Solution

Check whether the QPS and traffic metrics increase sharply last two hours and whether connection pool parameters configured on the Redis clients (such as Jedis and Lettuce) meet service requirements.

NOTE

For details about how to configure the Redis clients, see **5.1 Development and O&M Rules** and **5.8 Configuring Parameters for a Client Connection Pool**.

8.4.10 Common Client Errors and Troubleshooting Methods

Symptom 1

- The client displayed a message indicating that the network timed out for 10 seconds and the connection failed.


```
CommonResponseAspect exception!Redis command timed out; nested exception is
io.lettuce.core.RedisCommandTimeoutException: Command timed out after 10 second(s)
```

- Client: Lettuce
- Possible cause: The bandwidth of the client is used up.
- Solution: Check service resources and solve the resource bottleneck.

Symptom 2

- The client occasionally displayed a message indicating that the connection was unavailable.

```
[redisClient=[addr=XXXX], channel=[id: 0x0a0d20bc, L:0.0.0.0/0.0.0.0:53192]] is not active!
```

- Client: Redisson
- Possible cause: The client reconnection mechanism is incomplete. This issue may occur after an HA switchover is triggered on servers.
- Solution: Restart the client.

Symptom 3

- Error "Could not get a resource from the pool" was displayed, and there were a large number of TCP connections in the **CLOSE_WAIT** state on ECSs running the service program.
- Clients: Jedis and Lettuce
- Possible cause: The connection pool size configured for the client program is too small. As a result, connections to the Redis instance are insufficient when concurrent services increase dramatically.
- Solution: Check service code and configure sufficient size for the connection pool.

Symptom 4

- The connection pool of the client program timed out. The error information is as follows:

```
"redis: connection pool timeout
You can get that error when there are no free connections in the pool for Options.PoolTimeout
duration. If you are using redis.PubSub or redis.Conn, make sure to properly release PuSub/Conn
resources by calling Close method when they are not needed any more.
You can also get that error when Redis processes commands too slowly and all connections in the
pool are blocked for more than PoolTimeout duration."
```

- Client: Go-redis
- Possible cause: The connection pool size configured for the client program is too small. As a result, connections to the Redis instance are insufficient when concurrent services increase dramatically.
- Solution: Check service code and configure sufficient size for the connection pool.

8.5 Backup and Restoration

8.5.1 How Long Can a GeminiDB Redis Instance Backup Be Saved?

Automated backup data is kept based on the backup retention period you specified. There is no limit for the manual backup retention period. You can delete manual backups as needed.

For more backup information, see **4.7.2 Managing Automated Backups** and **4.7.3 Managing Manual Backups**.

8.6 Regions and AZs

8.6.1 Can Different AZs Communicate with Each Other?

An AZ is a part of a physical region with its own independent power supply and network. An AZ is generally an independent physical equipment room, ensuring independence of the AZ.

Each region contains multiple AZs. If one AZ becomes faulty, the other AZs in the same region can continue to provide services normally.

By default, different AZs in the same VPC can communicate with each other through an internal network.

For more information, see **Regions and AZs**.

8.6.2 Can I Change the Region of a GeminiDB Redis Instance?

Not supported. After an instance is created, its region cannot be changed.

NOTICE

To reduce network latency, select a region nearest from which you will access the instance. Instances deployed in different regions cannot communicate with each other over a private network. After you buy an instance, you cannot change its region.

8.7 Data Migration

8.7.1 What Do I Do if the GeminiDB Redis Link Cannot Be Found on DRS?

Currently, the whitelist mechanism is used for the GeminiDB Redis link to the cloud. You need to submit a service ticket to contact DRS engineers to enable the whitelist mechanism.

In the upper right corner of the console, choose **Service Tickets > Create Service Ticket** and contact customer service.

8.7.2 What Do I Do if the Error "ERR the worker queue is full, and the request cannot be executed" Is Displayed?

If the internal queue is full due to heavy migration traffic, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact technical support.

8.7.3 What Do I Do If the Error "ERR the request queue of io thread is full, and the request cannot be executed" Is Displayed?

If the internal queue is full due to heavy migration traffic, choose **Service Tickets > Create Service Ticket** in the upper right corner of the console and contact technical support.

8.7.4 What Do I DO If the Error "read error, please check source redis log or network" Is Displayed?

The sending buffer of the source end is too small. You need to modify the Redis parameter configuration of the source end. This parameter takes effect immediately after the value of **client-output-buffer-limit** is changed.

8.7.5 What Do I Do If the Error "slaveping_thread.cc-ThreadMain-90: error: Ping master error" Is Displayed?

When the **pika-port** command is executed, the specified IP address is **127.0.0.1**. You need to set it to another IP address, for example, the IP address of eth0.

8.7.6 What Do I Do If the Forward Migration Speed of the Synchronization Status Is Too Slow?

The value of **source.rdb.parallel** of redis-shake to adjust the migration concurrency. The default value is **0**, which is determined by the number of databases and shards.

8.7.7 What Do i Do When the Forward Migration Speed of the Synchronization Status Is Too Fast, and the Error Message "ERR Server Reply Timeout, Some Responses May Lose, but Requests Have Been Executed" Is Displayed?

The value of **parallel** is changed to adjust the concurrency of RDB transmission in the full process. The default value is **32**.

8.7.8 Can Data Be Migrated from Self-Built Redis 4.0, 5.0, and 6.2 to GeminiDB Redis API?

Yes. GeminiDB Redis API is compatible with Redis 6.2 (including 6.2.*x*) and earlier versions, such as 5.0, 4.0, and 2.8.

8.7.9 How Do I Migrate Data from Self-Built Primary/Standby and Cluster Redis Instances to GeminiDB Redis Instances?

DRS can be used for online migration.

- For details about how to migrate data from a single Redis instance to primary/standby GeminiDB Redis instances, see **From Redis to GeminiDB Redis**.
- For details about how to migrate data from a Redis cluster to a GeminiDB Redis cluster, see **From Cluster Redis to GeminiDB Redis**.

For details about how to migrate RDB files to GeminiDB Redis instances, see **4.4.7 (Recommended) Importing Data to Restore RDB Files to a GeminiDB Redis Instance**.

8.7.10 Why Cannot DRS Migrate Data from Third-Party Redis Such as ApsaraDB for Redis and TencentDB for Redis?

Generally, there are the following possible causes:

- Some self-developed Redis-like databases are not compatible with the PSync protocol.
- Architecture restrictions: For many cloud vendors, the proxy component is added between users and Redis. PSync is not supported due to the proxy.
- Security restrictions: In native Redis, fork() is used over PSync, which causes memory expansion, user request delay increase, and even out of memory.
- Business strategy: A large number of users use RedisShake to migrate services from the cloud or change the cloud, so PSync is shielded.

Generally, you can use a data migration service suitable for the corresponding cloud service. For details, see **4.4.1 Overview of the Redis Data Migration Solution**.

8.7.11 Which of the Following Factors Need to Be Considered When Data Is Migrated from Self-Built Primary/Standby Redis Instances to a GeminiDB Redis cluster?

GeminiDB Redis API is deployed in a proxy cluster and can be directly accessed by a single node or primary/standby nodes. No modification is required. Multi-key operations of primary/standby Redis instances are different from those of a Redis cluster, so services need to be modified.

All data of self-built single and primary/standby Redis instances is stored on the same node. Therefore, atomicity of multi-key operations, such as Lua, RPOPLPUSH, SDIFF, and SUNION, can be ensured. In a self-built Redis cluster, the modular hash function is executed for keys to determine which shard (node) will process the keys. Therefore, it is difficult to ensure atomicity when operations are performed on multiple keys across shards. To ensure atomicity of multi-key operations in a cluster, the Redis cluster uses a hashtag to ensure that multi-key operations are performed on the same node.

To use the hashtag, add the same strings to those multiple keys, for example, *{aaa}list1* and *{aaa}list2*. When processing the preceding keys, Redis identifies *{}* and calculates hash values only based on *aaa*. Therefore, multi-key operations are performed on the same node.

For details about commands for adding hashtags in a GeminiDB Redis cluster, see **8.3.9 Which Commands Require Hash Tags in GeminiDB Redis Cluster Instances?**.

8.7.12 Only 20% to 30% of 100 GB of Data Was Migrated to GeminiDB Redis. Is the Migration Incomplete?

GeminiDB Redis API offers an ultra-high data compression ratio of 4:1. Migrated data of multiple users complies with this rule. You can verify data consistency to determine whether the migration is complete. For example, verify key samples and the number of keys.

8.8 Memory Acceleration

8.8.1 Will All Data Be Cached to GeminiDB Redis Instances After Memory Acceleration Is Enabled and MySQL Database Data Is Updated?

No. You need to specify conversion rules of the MySQL database tablespaces, table names, and fields of GeminiDB Redis instances on the GUI. After the configuration is complete, data that meets the rules is automatically synchronized to a GeminiDB Redis instance.

8.8.2 If Memory Acceleration Is Enabled, GeminiDB Redis Instance Data Increases Continuously. Do I Need to Scale Out the Capacity? How Do I Manage Cached Data?

By default, each piece of data of GeminiDB Redis instances will expire in 30 days. You can adjust the expiration time. If the data volume keeps increasing, you need to scale out storage capacity of GeminiDB Redis instances in a timely manner.

8.8.3 Is Memory Acceleration Recommended When Customers' Service Data Can Be Synchronized Between MySQL and Redis? In Which Scenarios Can Memory Acceleration Be enabled?

If customers' service data can be synchronized between MySQL and Redis, you are advised to migrate cache data to GeminiDB Redis instances. Memory acceleration is recommended for new services to simplify development.

8.8.4 How Long Is the Latency of Synchronization from RDS for MySQL to GeminiDB Redis API? What Factors Affect the Latency?

Data can be synchronized in real time. The latency may be affected by the following factors and needs to be measured:

- Physical distance between RDS for MySQL and GeminiDB Redis instances. It is recommended that the instances be in the same region.
- You are advised to set the CPU specifications of RDS for MySQL to GeminiDB Redis instances to the same value.

8.8.5 Will the Source MySQL Database Be Affected After Memory Acceleration Is Enabled?

Memory acceleration works based on MySQL binlogs, which has little impact on the source MySQL database.

8.8.6 GeminiDB Redis Instances with Memory Acceleration Enabled Needs to Process a Large Number of Binlogs in a Short Period of Time. Will a Large Number of Resources Be Occupied and Online Services Be Affected?

If a large number of DDL operations are performed on the source MySQL database, a large number of GeminiDB Redis resources are consumed. You can query OPS (**dbcache_ops_per_sec**) after memory acceleration is enabled. You are advised to configure basic resource alarms. For details, see **4.13.2 Configuring Alarm Rules**.

8.9 Instance Freezing, Release, Deletion, and Unsubscription

Why Are My GeminiDB Redis Instances Released?

If your subscriptions have expired but not been renewed, or you are in arrears due to insufficient balance, your instances enter a grace period. If you do not renew the subscriptions or top up your account after the grace period expires, your instances will enter a retention period and become unavailable. If you still do not renew them or top up your account after the retention period ends, your instances will be released and your data stored will be deleted. For details, see **Service Suspension and Resource Release**.

Why Are My GeminiDB Redis Instances Frozen?

Your instances may be frozen for a variety of reasons. The most common reason is that you are in arrears.

Can I Still Back Up Data If My Instances Are Frozen?

No. If your GeminiDB Redis instances are frozen because your account is in arrears, go to top up your account to unfreeze your instances and then back up instance data.

How Do I Unfreeze My Instances?

If your GeminiDB Redis instances are frozen because your account is in arrears, you can unfreeze them by renewing them or topping up your account. The frozen GeminiDB Redis instances can be renewed, released, or deleted. Yearly/Monthly instances that have expired cannot be unsubscribed from, while those that have not expired can be unsubscribed from.

What Impacts Does Instance Freezing, Unfreezing or Release Have on My Services?

- After an instance is frozen:
  - It cannot be accessed, and your services will be interrupted. For example, if a GeminiDB Redis instance is frozen, it cannot be connected.
  - No changes can be performed on it if it is a yearly/monthly instance.
  - It can be unsubscribed from or deleted manually.
- After it is unfrozen, you can connect to it again.
- Releasing an instance means deleting it. Before the deletion, GeminiDB Redis API determines whether to **move the instance to the recycle bin** based on the recycling policy you specified.

How Do I Renew My Instances?

After a yearly/monthly GeminiDB Redis instance expires, you can renew it on the **Renewals** page. For details, see **Renewal Management**.

Can My Instances Be Recovered After They Are Released or Unsubscribed From?

If your instance is moved to the recycle bin after being deleted, you can recover it from the recycle bin by referring to **4.5.4.4 Recycling an Instance**. If the recycling policy is not enabled, you cannot recover it.

When you unsubscribe from an instance, confirm the instance information carefully. If you have unsubscribed from an instance by mistake, purchase a new one.

How Do I Delete a GeminiDB Redis Instance?

- To delete a pay-per-use instance, see **4.5.4.3 Deleting a Pay-per-Use Instance**.
- To delete a yearly/monthly instance, see **2.11.4 How Do I Unsubscribe from Yearly/Monthly Instances?**.

