Document Database Service

Best Practices

Issue	01
Date	2024-10-26

HUAWEI TECHNOLOGIES CO., LTD.

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 Overview

This document provides best practices for Huawei Cloud Document Database Service (DDS) and guides you through using DDS to best suit your business needs.

Service	Reference	Overview
Document Database Service	**Security Best Practices**	This section provides actionable guidance for enhancing the overall security of using DDS.
	Common Methods for Connecting to a DDS Instance	This section describes common DDS connection methods.
	From Other Cloud MongoDB to DDS	These sections describe how to migrate DDS data.
	From On-Premises MongoDB to DDS	
	From ECS-hosted MongoDB to DDS	
	How Do Replica Sets Achieve High Availability and Read/Write Splitting?	This section describes how to connect to a replica set instance to achieve high availability.
	Sharding	This section describes how to set cluster shards to improve database performance.
	How Do I Improve DDS Performance by Optimizing SQL Statements?	This section describes DDS usage suggestions.
	How Do I Prevent the dds mongos Cache Problem?	This section describes how to avoid the mongos route cache defect of the cluster.

Servic e	Reference	Overview
	How Do I Solve the High CPU Usage Issue?	This section describes how to troubleshoot high CPU usage.
	How Do I Troubleshoot High Memory Usage of DDS DB Instances?	This section describes how to troubleshoot high memory usage.
	What Can I Do If the Number of Connections of an Instance Reaches Its Maximum?	This section describes how to troubleshoot the fault that there are too many connections to a DB instance.
	Creating a User and Granting the Read-Only Permission to the User	This section describes how to use IAM to grant read-only permissions to DDS.
	Proper Use of Data Definition Languages (DDL) Statements	This section describes DDL statements used to create, modify, and delete the structure of databases and collections.

2 Security Best Practices

Security is a shared responsibility between Huawei Cloud and you. Huawei Cloud is responsible for the security of cloud services to provide a secure cloud. As a tenant, you should properly use the security capabilities provided by cloud services to protect data, and securely use the cloud. For details, see **Shared Responsibilities**.

This section provides actionable guidance for enhancing the overall security of using DDS. You can continuously evaluate the security status of your DDS DB instances and enhance their overall security defense by combining different security capabilities provided by DDS. By doing this, data stored in DDS DB instances can be protected from leakage and tampering both at rest and in transit.

Make security configurations from the following dimensions to meet your business needs.

- **No Binding an EIP to Access DDS over the Internet**
- **No Using Weak Passwords**
- **No Using the Default Port**
- **Limiting the Maximum Number of DDS Connections**
- **Disabling IPv6**
- **Disabling Script Execution**
- **Configuring the Audit Policy**
- **Enabling SSL**
- **Enabling Disk Encryption**
- **Enabling Data Backups**
- **Configuring Monitoring by Seconds and Alarm Rules**
- **Upgrading the Version of a DB Instance**
- **Checking Roles**

No Binding an EIP to Access DDS over the Internet

Do not deploy DDS on the Internet or DMZ. Instead, deploy DDS on the internal network of your company and use routers or firewalls to protect DDS. Do not bind an EIP to access DDS from the Internet. By doing so, DDS can be protected from unauthorized access and DDoS attacks. You are not advised **binding an EIP** to access DDS from the Internet. If necessary, you must **set security group rules**.

No Using Weak Passwords

When creating or changing an account password, ensure that the password meets the password complexity requirements and do not use weak passwords. By doing so, passwords can be protected from hacker and rainbow table attacks. For details about how to check for weak passwords, see **Checking for Weak Passwords**.

No Using the Default Port

The default port for MongoDB is 27017. If the default port is used, it is easy to be listened on, which poses security risks. You are advised to use a non-default port. For details, see **Changing a Database Port**.

Limiting the Maximum Number of DDS Connections

Excessive DDS connections will consumer excessive server resources, leading to sluggish response of the OPS operations (such as **query**, **insert**, **update**, and **delete**). Also, you need to set **net.maxIncomingConnections** to an appropriate value based on the operating system environment. If the value is greater than the maximum number of threads received by the operating system, the setting is invalid. For details, see **Parameters**.

Disabling IPv6

IPv6 subnets are not supported. You are advised to select an IPv4 subnet when creating a DDS DB instance.

Disabling Script Execution

If the **security.javascriptEnabled** parameter is enabled, JavaScript scripts can be executed on mongod, which poses security risks. If the **javascriptEnabled** option is disabled, the **mapreduce** and **group** commands cannot be used. If your application does not require operations such as MapReduce, you are advised to disable **javascriptEnabled**. For details, see **Parameters**.

Configuring the Audit Policy

The audit function can be used to record all database operations performed by users. Auditing logs can enhance your database security and help you analyze the cause of failed operations to improve system O&M. For details, see **Audit Logs**.

Enabling SSL

If SSL is not configured, data transmitted between the MongoDB client and server is in plaintext, which is vulnerable to eavesdropping, tampering, and man-in-the-middle attacks. To improve the security of data transmission, you are advised to enable SSL. For details, see **Enabling or Disabling SSL**.

Enabling Disk Encryption

Enabling disk encryption improves data security. For details, see "Disk Encryption" in **Custom Config**.

Enabling Data Backups

DDS supports automated and manual backups. You can periodically back up databases. If a database is faulty or data is damaged, you can restore the database using backups to ensure data reliability. For details, see **Data Backups**.

Configuring Monitoring by Seconds and Alarm Rules

DDS DB instances can be monitored by default. If the value of a metric exceeds the threshold, an alarm is triggered. The system automatically sends an alarm notification to the cloud account contact through SMN, helping you learn about the status of your DDS instance in a timely manner. Configure proper monitoring and alarm rules based on service requirements. For details, see **Monitoring and Alarm Reporting**.

Upgrading the Version of a DB Instance

DDS supports **minor version upgrade** and **major version update**. You can upgrade your DB instance to the latest version to add new functions, fix problems, and improve security and performance. You are advised to upgrade the version of a DB instance in a timely manner.

Checking Roles

DDS allows you to grant role-based permissions to a database account for data and command access. You are advised to create **user-defined roles** based on service requirements and grant the minimum permission to a database account. You can also **update** or **delete** a database user as needed.

Table 2-1 DDS role-based permissions

No.	Check Item	Description
1	The user with the userAdmin role	After the user with the userAdmin role is defined in the admin database, the user is provided with all permissions of all users. That is, the user with this role can define their own permissions on any database.
2	The user with the userAdminAnyDatabase role	After the user with the userAdminAnyDatabase role is defined, the user is provided with all permissions of all users. That is, the user with this role can define their own permissions on any database.

No.	Check Item	Description
3	The role with the anyAction action	After the role with the anyAction action is defined, the role-based user is provided with all operation permissions on the corresponding database, which affects permission management.
4	The role with the anyResource action	After the role with the anyResource action is defined, the role-based user is provided with all resource permissions on the corresponding database, which affects permission management.
5	The role with the changeCustomData action	After the role with the changeCustomData action is defined, the role-based user is provided with the permission to modify all user-defined information in the corresponding database, which affects permission management.
6	The role with the changePassword action	After the role with the changePassword action is defined, the role-based user is provided with the permission to change the passwords of all users in the corresponding database, which affects permission management.
7	The role with the createRole action	After the role with the createRole action is defined, the role-based user is provided with the permission to create all roles in the corresponding database, which affects permission management.
8	The role with the createUser action	After the role with the createUser action is defined, the role-based user is provided with the permission to create all users in the corresponding database, which affects permission management.
9	The role with the dropRole action	After the role with the dropRole action is defined, the role-based user is provided with the permission to delete any role in the corresponding database, which affects permission management.
10	The role with the dropUser action	After the role with the dropUser action is defined, the role-based user is provided with the permission to delete any user in the corresponding database, which affects permission management.

No.	Check Item	Description
11	The role with the grantRole action	After the role with the grantRole action is defined, the role-based user is provided with the permission to grant all role permissions to all users in the corresponding database, which affects permission management.
12	The role with the revokeRole action	After the role with the revokeRole action is defined, the role-based user is provided with the permission to revoke any role permission from any user in the corresponding database, which affects permission management.
13	The role with the authSchemaUpgrade action	After the role with the authSchemaUpgrade action is defined, the role-based user is provided with the permission to execute **authschemaupgrade**, which affects permission management. The **authschemaupgrade** command is used to modify the user authentication conversion format.
14	The role with the closeAllDatabases action	After the role with the closeAllDatabases action is defined, the role-based user is provided with the permission to run the **closeAllDatabases** command, which affects permission management. The **closeAllDatabases** command is used to close all databases and release the memory occupied by MongoDB.
15	The role with the dropDatabase action	After the role with the dropDatabase action is defined, the role-based user is provided with the permission to run the **dropDatabase** command to delete any database, which affects permission management.
16	The role with the getParameter action	After the role with the getParameter action is defined, the role-based user is provided with the permission to run the **getParameter** command to view the values of all command line options, which affects permission management.
17	The role with the setParameter action	After the role with the setParameter action is defined, the role-based user is provided with the permission to run the **setParameter** command to change the value of any command line option, which affects permission management.

No.	Check Item	Description
18	The role with the shutdown action	After the role with the shutdown action is defined, the role-based user is provided with the permission to run the **shutdown** command to clear all database resources and stop processes, which affects permission management.
19	The role with the getCmdLineOpts action	After the role with the getCmdLineOpts action is defined, the role-based user is provided with the permission to run the **getCmdLineOpts** command to obtain the **argv** and **parsed** fields, which affects permission management. The **argv** field contains the mongod or mongos command string, and the **parsed** field contains all runtime options.
20	The role with the internal action	After the role with the internal action is defined, the role-based user is provided with the permission to perform all operations on the corresponding database, which affects permission management.
21	The user with the readWrite role	After the user with the readWrite role is defined, the role-based user is provided with the read permission and data modification permission on the corresponding database, which affects permission management.
22	The user with the backup role	After the user with the backup role is defined, the role-based user is provided with the insert and update permissions in the **mms.bak** file in the admin database, which affects permission management.
23	The user with the clusterAdmin role	After the user with the clusterAdmin role is defined, the role-based user is provided with the highest cluster management permission, which affects permission management. The role has the permissions of the clusterManager, clusterMonitor, and hostManager roles.
24	The user with the clusterManager role	After the user with the clusterManager role is defined to manage and monitor operations in a cluster, the role-based user is provided with the permission to manage local databases that are shared and replicated, which affects permission management.

No.	Check Item	Description
25	The user with the clusterMonitor role	After the user with clusterMonitor role is defined, the role-based user is provided with the read-only permission for the monitoring tool, which affects permission management.
26	The user with the dbAdmin role	After the user with the dbAdmin role is defined, the role-based user is provided with the administrator permissions on the corresponding database, which affects permission management.
27	The user with the dbAdminAnyDatabase role	After the user with the dbAdminAnyDatabase role is defined, the role-based user is provided with the same permissions as the dbAdmin role, which affects permission management. The role applies to all databases in a cluster, and also has the listDatabases operation permission on the cluster.
28	The user with the dbOwner role	After the user with the dbOwner role is defined, the role-based user is provided with the permission to perform all database management operations, which affects permission management. This role has the permissions of the readWrite, dbAdmin, and userAdmin roles.
29	The user with the hostManager role	After the user with hostManager role is defined, the role-based user is provided with the permission to monitor and manage servers, which affects permission management.
30	The user with the readAnyDatabase role	After the user with readAnyDatabase role is defined, the role-based user is provided with the permission to read data from all databases, which affects permission management. This role also has the listdatabases operation permission on the cluster.
31	The user with the readWriteAnyDatabase role	After the user with readWriteAnyDatabase role is defined, the role-based user is provided with the permission to read data from and write data to all databases, which affects permission management. This role also has the listdatabases operation permission on the cluster.

No.	Check Item	Description
32	The user with the restore role	After the user with restore role is defined, the role-based user is provided with the permission required for restoring backups, which affects permission management.
33	The user with the root role	After the user with root role is defined, the role-based user is provided with all operation permissions on all resources, which affects permission management. The role has the permissons of the readWriteAnyDatabase, dbAdminAnyDatabase, userAdminAnyDatabases, clusterAdmin, and restore roles.
34	The user with the userAdmin role	After the user with the userAdmin role is defined, the role-based user is provided with the permissions of all users, including their own permissions, which affects permission management.
35	The user with the userAdminAnyDatabase role	After the user with the userAdminAnyDatabase role is defined, the role-based user is provided with the permissions of all users on all databases, including their own permissions, which affects permission management.

3 Common Methods for Connecting to a DDS Instance

This section describes how to connect to a DDS instance using the following four methods:

- Mongo Shell
- Python Mongo
- Java Mongo
- Using Spring MongoTemplate to Perform MongoDB Operations

Mongo Shell

- Prerequisites
  a. To connect an ECS to a DDS instance, run the following command to connect to the IP address and port of the instance server to test the network connectivity.

     curl *ip:port*

     If the message **It looks like you are trying to access MongoDB over HTTP on the native driver port** is displayed, the ECS and DDS instance can communicate with each other.
  b. Download the client installation package whose version is the same as the instance version from the **MongoDB official website**. Decompress the package, obtain the **mongo** file, and upload it to the ECS.
  c. If SSL is enabled, download the root certificate and upload it to the ECS.
- Connection commands
  – SSL is enabled.

    ./mongo *ip:port* **--authenticationDatabase admin -u** *username* **-p** *password* **--ssl --sslCAFile $***path to certificate authority file* **--sslAllowInvalidHostnames**

  – SSL is disabled.

    ./mongo *ip:port* **--authenticationDatabase admin -u** *username* **-p** *password*

Table 3-1 Parameter description

Parameter	Description
ip	If you access an instance from an ECS, *ip* is the private IP address of the instance.
	If you access an instance from a device over a public network, *ip* is the EIP bound to the instance,
port	Database port displayed on the **Basic Information** page. Default value: **8635**
username	Current username
password	Password of the current username
path to certificate authority file	Path of the SSL certificate

- Precautions
  a. If SSL is enabled, the connection command must contain **--ssl** and **--sslCAFile**.
  b. **--authenticationDatabase** must be set to **admin**. If you log in to the database as user **rwuser**, switch to **admin** for authentication.

For details, see **Connecting to an Instance** in *Getting Started with Document Database Service*.

Python Mongo

- Prerequisites
  a. To connect an ECS to a DDS instance, run the following command to connect to the IP address and port of the instance server to test the network connectivity.

     curl *ip*:*port*

     If the message **It looks like you are trying to access MongoDB over HTTP on the native driver port** is displayed, the network connectivity is normal.
  b. Install Python and third-party installation package **pymongo** on the ECS. Pymongo 2.8 is recommended.
  c. If SSL is enabled, download the root certificate and upload it to the ECS.
- Input the connection code.
  – SSL is enabled.

```
import ssl
import os
from pymongo import MongoClient
# 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.
rwuser = os.getenv('EXAMPLE_USERNAME_ENV')
password = os.getenv('EXAMPLE_PASSWORD_ENV')
conn_urls="mongodb://%s:%s@ip:port/{mydb}?authSource=admin"
connection = MongoClient(conn_urls % (rwuser,
password),connectTimeoutMS=5000,ssl=True,
ssl_cert_reqs=ssl.CERT_REQUIRED,ssl_match_hostname=False,ssl_ca_certs=${path to
certificate authority file})
dbs = connection.database_names()
print "connect database success! database names is %s" % dbs
```

   - SSL is disabled.

```
import ssl
import os
from pymongo import MongoClient
# 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.
rwuser = os.getenv('EXAMPLE_USERNAME_ENV')
password = os.getenv('EXAMPLE_PASSWORD_ENV')
conn_urls="mongodb://%s:%s@ip:port/{mydb}?authSource=admin"
connection = MongoClient(conn_urls % (rwuser, password),connectTimeoutMS=5000)
dbs = connection.database_names()
print "connect database success! database names is %s" % dbs
```

- Precautions
  a. *{mydb}* is the name of the database to be connected.
  b. The authentication database in the URL must be **admin**. Set **authSource** to **admin**.

Java Mongo

- Prerequisites
  a. To connect an ECS to a DDS instance, run the following command to connect to the IP address and port of the instance server to test the network connectivity.

     curl *ip:port*

     If the message **It looks like you are trying to access MongoDB over HTTP on the native driver port** is displayed, the ECS and DDS instance can communicate with each other.
  b. Download the **MongoDB JAR** package compatible with the instance version by referring to the **MongoDB Compatibility** table.
  c. JDK is installed on the ECS.
  d. If SSL is enabled, download the root certificate and upload it to the ECS.
- Input the connection code.

  Use keytool to generate a trustStore.

  // 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");

keytool -import -file */var/chroot/mongodb/CA/ca.crt* **-keystore** */home/Mike/jdk1.8.0_112/jre/lib/security/mongostore* **-storetype pkcs12 -storepass ${password}**

NOTE

- **/var/chroot/mongodb/CA/ca.crt** is the root certificate path.
- **/home/Mike/jdk1.8.0_112/jre/lib/security/mongostore** indicates the path of the generated truststore.

– SSL is enabled.

```
import java.util.ArrayList;
import java.util.List;
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
import com.mongodb.ServerAddress;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.MongoCollection;
import com.mongodb.MongoClientURI;
import com.mongodb.MongoClientOptions;
public class MongoDBJDBC {
public static void main(String[] args){
    try {
            System.setProperty("javax.net.ssl.trustStore", "/home/Mike/
jdk1.8.0_112/jre/lib/security/mongostore");
            // 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");
            System.setProperty("javax.net.ssl.trustStorePassword", password);
            ServerAddress serverAddress = new ServerAddress("ip", port);
            List addrs = new ArrayList();
            addrs.add(serverAddress);
            // 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 userName = System.getenv("EXAMPLE_USERNAME_ENV");
            String rwuserPassword = System.getenv("EXAMPLE_PASSWORD_ENV");
            MongoCredential credential =
MongoCredential.createScramSha1Credential("rwuser", "admin",
rwuserPassword.toCharArray());
            List credentials = new ArrayList();
            credentials.add(credential);
            MongoClientOptions opts= MongoClientOptions.builder()
            .sslEnabled(true)
            .sslInvalidHostNameAllowed(true)
            .build();
            MongoClient mongoClient = new MongoClient(addrs,credentials,opts);
            MongoDatabase mongoDatabase = mongoClient.getDatabase("testdb");
            MongoCollection collection =
mongoDatabase.getCollection("testCollection");
            Document document = new Document("title", "MongoDB").
            append("description", "database").
```

```
              append("likes", 100).
              append("by", "Fly");
              List documents = new ArrayList();
              documents.add(document);
              collection.insertMany(documents);
              System.out.println("Connect to database successfully");
              } catch (Exception e) {
              System.err.println( e.getClass().getName() + ": " + e.getMessage() );
         }
      }
}
```

Sample codes:

```
javac -cp .:mongo-java-driver-3.2.0.jar MongoDBJDBC.java
java -cp .:mongo-java-driver-3.2.0.jar MongoDBJDBC
```

– SSL is disabled.

```
import java.util.ArrayList;
import java.util.List;
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
import com.mongodb.ServerAddress;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.MongoCollection;
import com.mongodb.MongoClientURI;
import com.mongodb.MongoClientOptions;
public class MongoDBJDBC {
public static void main(String[] args){
      try {
              ServerAddress serverAddress = new ServerAddress("ip", port);
              List addrs = new ArrayList();
              addrs.add(serverAddress);
              // 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 userName = System.getenv("EXAMPLE_USERNAME_ENV");
              String rwuserPassword = System.getenv("EXAMPLE_PASSWORD_ENV");
              MongoCredential credential =
MongoCredential.createScramSha1Credential("rwuser", "admin",
rwuserPassword.toCharArray());
              List credentials = new ArrayList();
              credentials.add(credential);
              MongoClient mongoClient = new MongoClient(addrs,credentials);
              MongoDatabase mongoDatabase = mongoClient.getDatabase("testdb");
              MongoCollection collection =
mongoDatabase.getCollection("testCollection");
              Document document = new Document("title", "MongoDB").
              append("description", "database").
              append("likes", 100).
              append("by", "Fly");
              List documents = new ArrayList();
              documents.add(document);
              collection.insertMany(documents);
              System.out.println("Connect to database successfully");
              } catch (Exception e) {
              System.err.println( e.getClass().getName() + ": " + e.getMessage() );
         }
       }
}
```

Using Spring MongoTemplate to Perform MongoDB Operations

- How to Use

  The following describes how to use Spring MongoTemplate to perform operations on MongoDB. For details, visit the **MongoDB official website**.

- Prerequisites

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-mongodb</artifactId>
    <exclusions>
        <exclusion>
            <artifactId>spring-boot-starter-logging</artifactId>
            <groupId>org.springframework.boot</groupId>
        </exclusion>
    </exclusions>
</dependency>
```

- Configuration Guide

```
spring:
  data:
    mongodb:            #MongoDB configuration, which is for reference only
      // 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 userName = System.getenv("EXAMPLE_USERNAME_ENV");
      String rwuserPassword = System.getenv("EXAMPLE_PASSWORD_ENV");
      uri: mongodb://" + userName + ":" + rwuserPassword +
"@192.***.***.***:8635,192.***.***.***:8635/${mongodb.database}
      database: ${mongodb.database}
```

- Development Guide

```
/**
 * MongoDB execution
 */
@Autowired
private MongoTemplate template;

/**
 * Log configuration
 */
@Autowired
private LoggingProperties properties;

@Override
public void write(BaseLog businessLog, LoggingOption option) {
    if (template != null) {
        LoggingConfig config = properties.getBusinessConfig(businessLog.getCategory());
        String collection = config.getMeta().get("collection");
        if (StringUtils.isNotEmpty(collection)) {
            Object data = mapping(businessLog, config);
            template.save(data, collection);
            if (log.isDebugEnabled()) {
                log.debug("save audit log to mongodb successfully!, message: {}",

StringEscapeUtils.escapeJava(TransformUtil.toJsonByJackson(businessLog)));
            }
        } else {
            log.warn("mongo log write log failed, mongoconfig is null");
        }
    } else {
        log.warn("mongo log write log failed, mongoTemplate is null");
    }
}
```

- Precautions
  a. In SSL mode, you need to manually generate the trustStore file.
  b. Change the authentication database to **admin**, and then switch to the service database after authentication.

4 From Other Cloud MongoDB to DDS

DRS helps you migrate MongoDB databases from other cloud platforms to DDS on the current cloud. With DRS, you can migrate databases online with zero downtime and your services and databases can remain operational during migration.

This section describes how to use DRS to migrate MongoDB databases from another cloud to DDS on the current cloud. Migration scenarios include:

- Migrating MongoDB databases from another cloud to DDS on the current cloud.
- Migrating self-built MongoDB databases from servers on another cloud to DDS on the current cloud.

Resource Planning

Table 4-1 Resource planning

Category	Subcategory	Planned Value	Description
VPC	VPC name	vpc-dds	Specify a name that is easy to identify.
	Region	AP-Singapore	To achieve lower network latency, select the region nearest to you.
	AZ	AZ1	-
	Subnet	10.0.0.0/24	Select a subnet with sufficient network resources.
	Subnet name	subnet-default	Specify a name that is easy to identify.

Category	Subcategory	Planned Value	Description
MongoDB database on other clouds	Database version	MongoDB 4.4	-
	IP address	192.168.0.1	This is only an example.
	Port	8635	This is only an example.
DDS instance	Instance name	dds-test	Specify a name that is easy to identify.
	Database version	DDS 4.4	-
	AZ type	Single-AZ	This is only an example.
	AZ	AZ1	AZ1 is selected in this example.
	Instance specifications	Enhanced II 4 vCPUs | 16 GB	The specifications in the left column are selected in this example. You need to select specifications meeting your workload requirements.
DRS migration task	Task name	DRS-test-migrate	Specify a name that is easy to identify.
	Source DB engine	MongoDB	-
	Destination DB engine	DDS	-
	Network type	Public network	Public network is used in this example.

Diagram

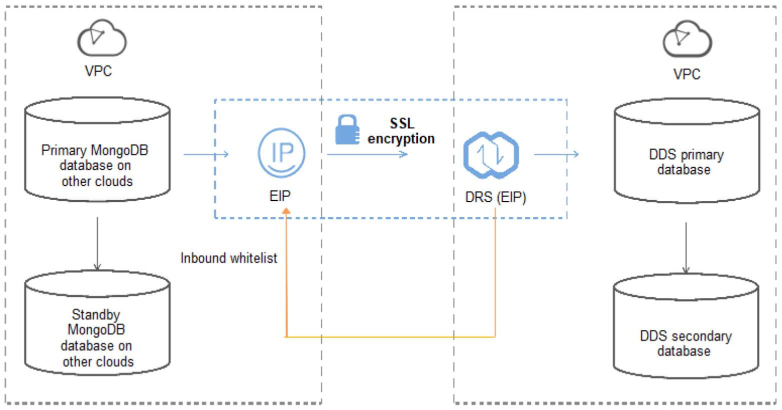
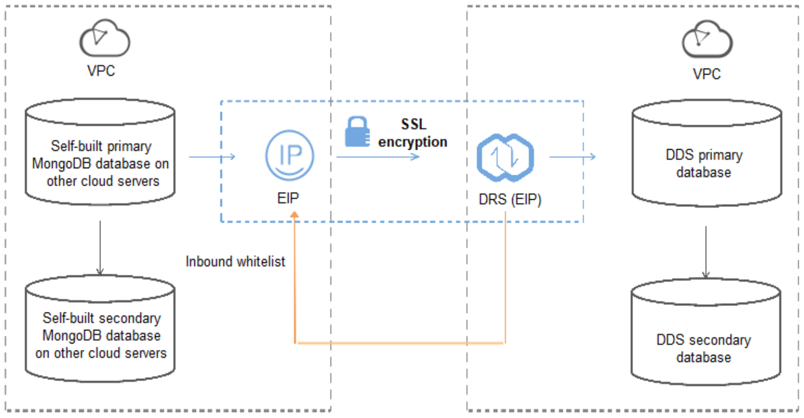


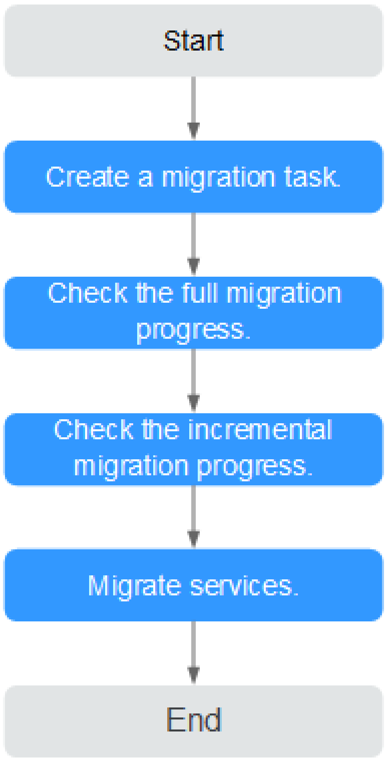
Figure 4-1 Migrating MongoDB databases from other clouds

Figure 4-2 Migrating MongoDB databases from other cloud servers


VPC
Self-built primary MongoDB database on other cloud servers
Self-built secondary MongoDB database on other cloud servers
EIP
SSL encryption
Inbound whitelist
DRS (EIP)
VPC
DDS primary database
DDS secondary database

Migration Process

Figure 4-3 Flowchart


Start
Create a migration task.
Check the full migration progress.
Check the incremental migration progress.
Migrate services.
End

Migration Suggestions (Important)

- Database migration is closely impacted by a wide range of environmental and operational factors. To ensure the migration goes smoothly, perform a test run before the actual migration to help you detect and resolve any potential issues in advance. Recommendations on how to minimize any potential impacts on your data base are provided in this section.
- It is strongly recommended that you start your migration task during off-peak hours. A less active database is easier to migrate successfully. If the data is fairly static, there is less likely to be any severe performance impacts during the migration.

Notes on Migration (Important)

NOTICE

Before creating a migration task, read the migration notes carefully.

For details, see **precautions** on using specific migration tasks in *Data Replication Service Real-Time Migration*.

Preparations

1. Permissions

   Table 4-2 lists the permissions required for the source and destination databases when migrating a MongoDB database from another cloud to DDS on the current cloud.

Table 4-2 Required permissions

Database	Full Migration Permission	Full+Incremental Migration Permission
Source	● Replica set: The source database user must have the read permission for the database to be migrated. ● Single node: The source database user must have the read permission for the database to be migrated. ● Cluster: The source database user must have the read permission for the databases to be migrated and the config database. ● To migrate accounts and roles of the source database, the source database user must have the read permission for the **system.users** and **system.roles** system tables of the admin database.	● Replica set: The source database user must have the read permission for the databases to be migrated and the local database. ● Single node: The source database user must have the read permission for the databases to be migrated and the local database. ● Cluster: The source mongos node user must have the readAnyDatabase permission for the databases to be migrated and the config database. The source shard node user must have the readAnyDatabase permission for the admin database and the read permission for the local database. ● To migrate accounts and roles of the source database, the source database user must have the read permission for the **system.users** and **system.roles** system tables of the admin database.
Destination	The destination database user must have the dbAdminAnyDatabase permission for the admin database and the readWrite permission for the destination database. If the destination database is a cluster instance, the migration account must have the read permission for the config database.	

- Source database permissions:

  The source MongoDB database user must have all the required permissions listed in **Table 4-2**. If the permissions are insufficient, create a user that has all of the permissions on the source database.
- Destination database permissions:

  If the destination database is a DDS database, the initial account can be used.

2. Network settings

   Enable public accessibility for the source database.

   - Source database network settings:

     Any source database MongoDB instances will need to be accessible from the Internet.
   - Destination database network settings: No settings are required.

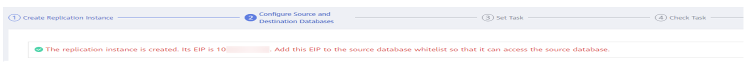
3. Security rules
   - Source database security group settings:

     The replication instance needs to be able to access the source MongoDB instance. That means that the EIP of the replication instance must be on the whitelist of the source MongoDB instance.

     Before configuring the network whitelist, you need to obtain the EIP of the replication instance.

     - After creating a replication instance on the DRS console, you can find the EIP on the **Configure Source and Destination Databases** page as shown in **Figure 4-4**.

       Figure 4-4 EIP of the replication instance


① Create Replication Instance
② Configure Source and Destination Databases
③ Set Task
④ Check Task

The replication instance is created. Its EIP is 10[redacted]. Add this EIP to the source database whitelist so that it can access the source database.

     You can also add 0.0.0.0/0 to the source database whitelist to allow any IP address to access the source database but this action may result in security risks.

     If you do take this step, then once the migration is complete, you should delete this item from the whitelist or your system will insecure.
   - Destination database security group settings:

     By default, the destination database and the DRS replication instance are in the same VPC and can communicate with each other. No further configuration is required.

4. Other

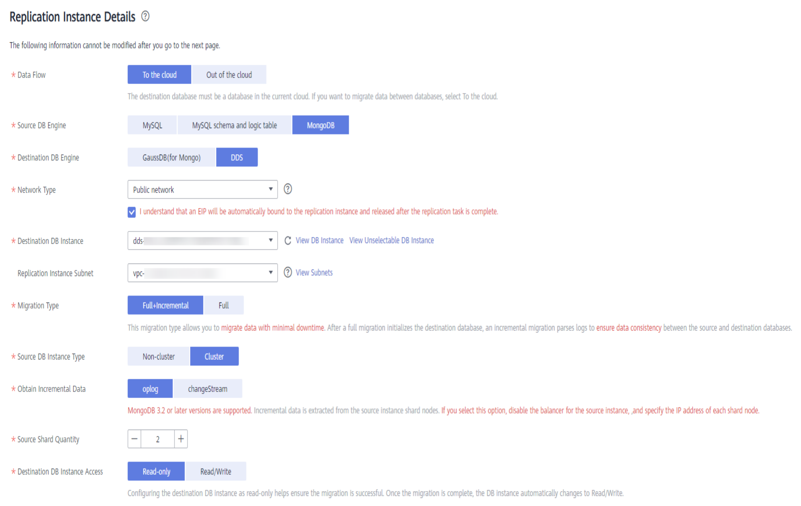
   You need to export the user information of the MongoDB database first and manually add it to the destination DDS DB instance because the user information will not be migrated.

Migration Procedure

Step 1 Create a migration task.

1. Log in to the management console and choose **Databases** > **Data Replication Service** to go to the DRS console.
2. On the **Online Migration Management** page, click **Create Migration Task**.
3. On the **Replication Instance Information** page, configure the task details, description, and replication instance details and click **Next**.

Figure 4-5 Replication instance information



Replication Instance Details

The following information cannot be modified after you go to the next page.

* Data Flow: [✓ To the cloud] Out of the cloud
The destination database must be a database in the current cloud. If you want to migrate data between databases, select To the cloud.

* Source DB Engine: MySQL MySQL schema and logic table [✓ MongoDB]

* Destination DB Engine: GaussDB(for Mongo) [✓ DDS]

* Network Type: Public network
[✓] I understand that an EIP will be automatically bound to the replication instance and released after the replication task is complete.

* Destination DB Instance: dds-[redacted] View DB Instance View Unselectable DB Instance

Replication Instance Subnet: vpc-[redacted] View Subnets

* Migration Type: [✓ Full+Incremental] Full
This migration type allows you to migrate data with minimal downtime. After a full migration initializes the destination database, an incremental migration parses logs to ensure data consistency between the source and destination databases.

* Source DB Instance Type: Non-cluster [✓ Cluster]

* Obtain Incremental Data: [✓ oplog] changeStream
MongoDB 3.2 or later versions are supported. Incremental data is extracted from the source instance shard nodes. If you select this option, disable the balancer for the source instance, ,and specify the IP address of each shard node.

* Source Shard Quantity: 2

* Destination DB Instance Access: [✓ Read-only] Read/Write
Configuring the destination DB instance as read-only helps ensure the migration is successful. Once the migration is complete, the DB instance automatically changes to Read/Write.

Table 4-3 Task settings

Parameter	Description
Region	The region where the replication instance is deployed. You can change the region. To reduce latency and improve access speed, select the region closest to your workloads.
Project	The project corresponds to the current region and can be changed.
Task Name	The task name consists of 4 to 50 characters, starts with a letter, and can contain only letters (case-insensitive), digits, hyphens (-), and underscores (_).
Description	The description consists of a maximum of 256 characters and cannot contain the following special characters: =<>&'\"

Table 4-4 Replication instance settings

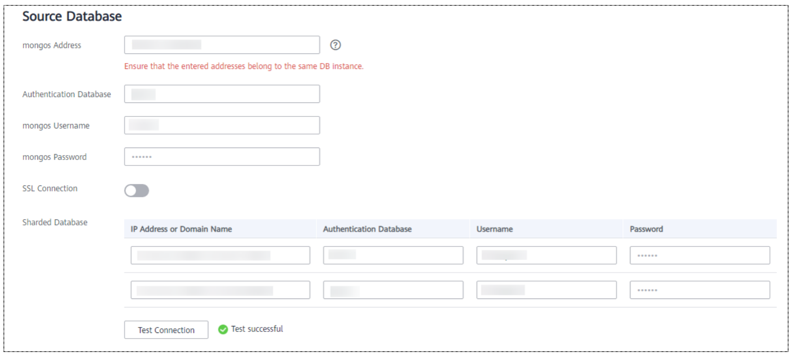
Parameter	Description
Data Flow	**To the cloud**

Parameter	Description
Source DB Engine	Select **MongoDB**.
Destination DB Engine	Select **DDS**.
Network Type	Select **Public network**.
Destination DB Instance	The DDS DB instance you purchased.
Replication Instance Subnet	The subnet where the replication instance resides. You can also click **View Subnet** to go to the network console to view the subnet where the instance resides. By default, the DRS instance and the destination DB instance are in the same subnet. You need to select the subnet where the DRS instance resides, and there are available IP addresses for the subnet. To ensure that the replication instance is successfully created, only subnets with DHCP enabled are displayed.
Migration Type	– **Full** This migration type is suitable for scenarios where service interruption is acceptable. All objects in non-system databases are migrated to the destination database at one time. The objects include collections and indexes. – **Full+Incremental** The full+incremental migration type allows you to migrate data without interrupting services. After a full migration initializes the destination database, an incremental migration parses logs to ensure data consistency between the source and destination databases.
Source DB Instance Type	If you select **Full+Incremental** for **Migration Type**, set this parameter based on the source database. – If the source database is a cluster instance, set this parameter to **Cluster**. – If the source database is a replica set or a single node instance, set this parameter to **Non-cluster**.

Parameter	Description
Obtain Incremental Data	This parameter is available for configuration if **Source DB Instance Type** is set to **Cluster**. You can determine how to capture data changes during the incremental synchronization. – oplog: For MongoDB 3.2 or later, DRS directly connects to each shard of the source DB instance to extract data. If you select this mode, you must disable the balancer of the source instance. When testing the connection, you need to enter the connection information of each shard node of the source instance. – changeStream: This method is recommended. For MongoDB 4.0 and later, DRS connects to mongos nodes of the source instance to extract data. If you select this method, you must enable the WiredTiger storage engine of the source instance. **NOTE** Only whitelisted users can use **changeStream**. To use this function, submit a service ticket. In the upper right corner of the management console, choose **Service Tickets** > **Create Service Ticket** to submit a service ticket.
Source Shard Quantity	If **Source DB Instance Type** is set to **Cluster** and **Obtain Incremental Data** is set to **oplog**, enter the number of source shard nodes. The default minimum number of source DB instances is 2 and the maximum number is 32. You can set this parameter based on the number of source database shards.

4. On the **Configure Source and Destination Databases** page, wait until the replication instance is created. Then, specify source and destination database information and click **Test Connection** for both the source and destination databases to check whether they have been connected to the replication instance. After the connection tests are successful, select the check box before the agreement and click **Next**.

Figure 4-6 Source database information



Ensure that the entered addresses belong to the same DB instance.

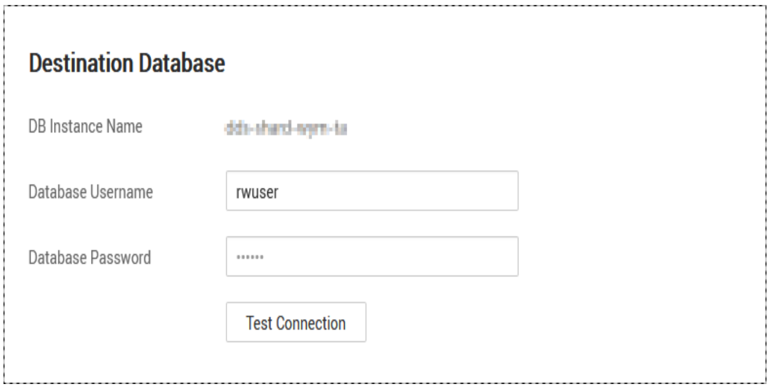
Test successful

Table 4-5 Source database settings

Parameter	Description
mongos Address	IP address or domain name of the source database in the **IP address/Domain name:Port** format. The port of the source database. Range: 1 - 65534 You can enter a maximum of three groups of IP addresses or domain names of the source database. Separate multiple values with commas (,). For example: 192.168.0.1:8080,192.168.0.2:8080. Ensure that the entered IP addresses or domain names belong to the same sharded cluster. **NOTE** If multiple IP addresses or domain names are entered, the test connection is successful as long as one IP address or domain name is accessible. Therefore, you must ensure that the IP address or domain name is correct.
Authentication Database	The name of the authentication database. For example: The default authentication database of Huawei Cloud DDS instance is **admin**.
mongos Username	A username for the source database.
mongos Password	The password for the source database username.
SSL Connection	SSL encrypts the connections between the source and destination databases. If SSL is enabled, upload the SSL CA root certificate.
Sharded Database	Enter the information about the sharded databases in the source database.

– Destination database configuration

Figure 4-7 Destination database information


Destination Database

DB Instance Name

Database Username rwuser

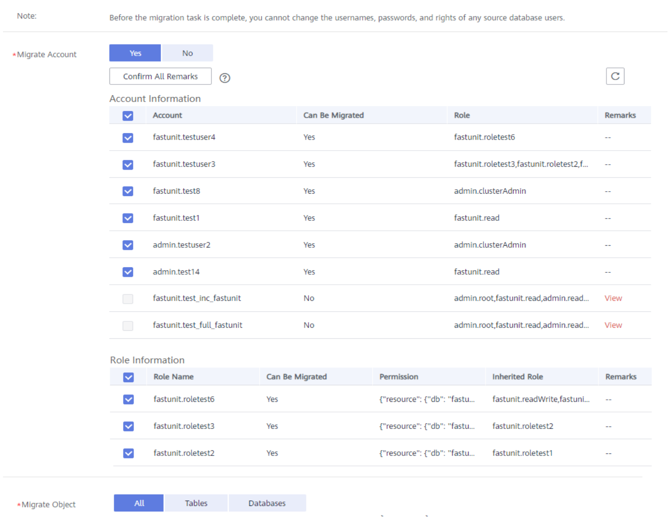
Database Password ••••••

Table 4-6 Destination database settings

Parameter	Description
DB Instance Name	The DB instance you selected when creating the migration task and cannot be changed.
Database Username	The username for accessing the destination database.
Database Password	The password for the database username.

5. On the **Set Task** page, select migration objects and click **Next**.

Figure 4-8 Migration object



Note: Before the migration task is complete, you cannot change the usernames, passwords, and rights of any source database users.

*Migrate Account [✓ Yes] No

Account Information

✓	Account	Can Be Migrated	Role	Remarks
✓	fastunit.testuser4	Yes	fastunit.roletest6	--
✓	fastunit.testuser3	Yes	fastunit.roletest3,fastunit.roletest2,f...	--
✓	fastunit.test8	Yes	admin.clusterAdmin	--
✓	fastunit.test1	Yes	fastunit.read	--
✓	admin.testuser2	Yes	admin.clusterAdmin	--
✓	admin.test14	Yes	fastunit.read	--
	fastunit.test_inc_fastunit	No	admin.root,fastunit.read,admin.read...	View
	fastunit.test_full_fastunit	No	admin.root,fastunit.read,admin.read...	View

Role Information

✓	Role Name	Can Be Migrated	Permission	Inherited Role	Remarks
✓	fastunit.roletest6	Yes	{"resource": {"db": "fastu...	fastunit.readWrite,fastuni...	--
✓	fastunit.roletest3	Yes	{"resource": {"db": "fastu...	fastunit.roletest2	--
✓	fastunit.roletest2	Yes	{"resource": {"db": "fastu...	fastunit.roletest1	--

*Migrate Object [✓ All] Tables Databases

Table 4-7 Migration object

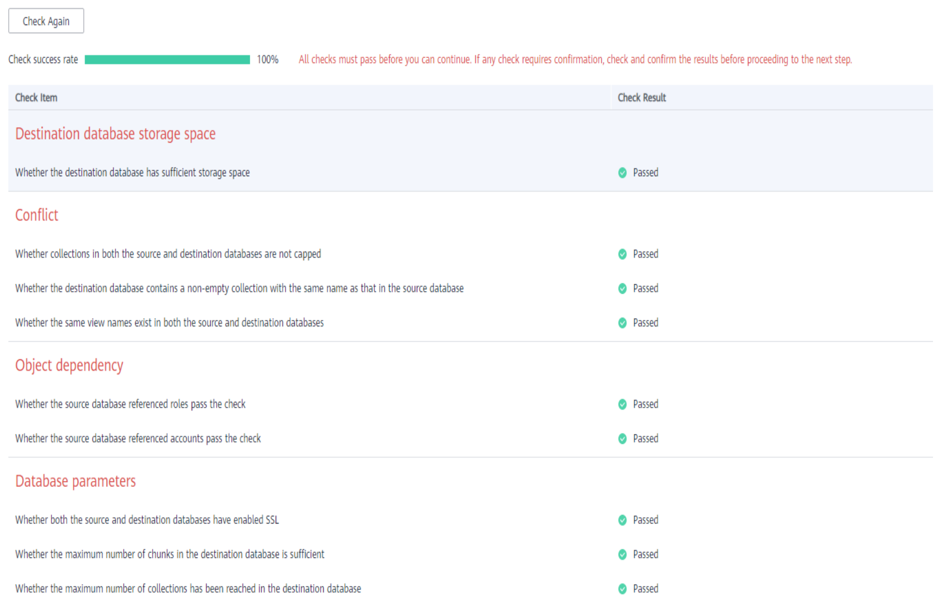
Parameter	Description
Migrate Account	There are accounts that can be migrated completely and accounts that cannot be migrated. You can choose whether to migrate the accounts. Accounts that cannot be migrated or accounts that are not selected will not exist in the destination database. Ensure that your services will not be affected by these accounts. – **Yes** If you choose to migrate accounts, see **Migrating Accounts** in *Data Replication Service User Guide* to migrate database users and roles. – **No** During the migration, accounts and roles are not migrated.

Parameter	Description
Migrate Object	You can choose to migrate all objects, tables, or databases based on your service requirements. – **All**: All objects in the source database are migrated to the destination database. After the migration, the object names will remain the same as those in the source database and cannot be modified. – **Tables**: The selected table-level objects will be migrated. – **Databases**: The selected database-level objects will be migrated. If the source database is changed, click ↻ in the upper right corner before selecting migration objects to ensure that the objects to be selected are from the changed source database. **NOTE** – If you choose not to migrate all of the databases, the migration may fail because the objects, such as stored procedures and views, in the database to be migrated may have dependencies on other objects that are not migrated. To ensure a successful migration, you are advised to migrate all of the databases. – When you select an object, the spaces before and after the object name are not displayed. If there are two or more consecutive spaces in the middle of the object name, only one space is displayed. – The search function can help you quickly select the required database objects.

6. On the **Check Task** page, check the migration task.
   - If any check fails, review the cause and rectify the fault. After the fault is rectified, click **Check Again**.

     For details about how to handle check failures, see **Checking Whether the Source Database Is Connected** in *Data Replication Service User Guide*.
   - If all check items are successful, click **Next**.

Figure 4-9 Task Check



Check Again

Check success rate 100% All checks must pass before you can continue. If any check requires confirmation, check and confirm the results before proceeding to the next step.

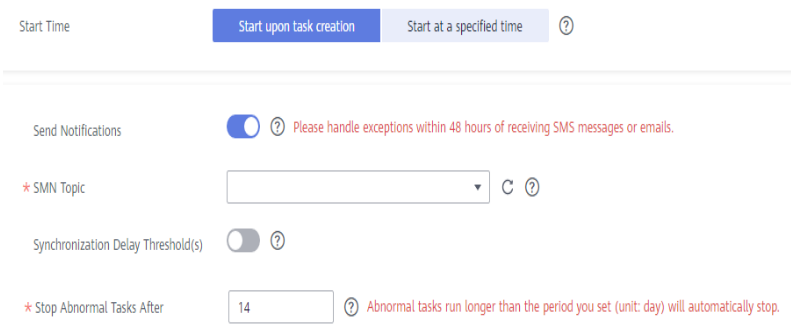
Check Item	Check Result
Destination database storage space	
Whether the destination database has sufficient storage space	✓ Passed
Conflict	
Whether collections in both the source and destination databases are not capped	✓ Passed
Whether the destination database contains a non-empty collection with the same name as that in the source database	✓ Passed
Whether the same view names exist in both the source and destination databases	✓ Passed
Object dependency	
Whether the source database referenced roles pass the check	✓ Passed
Whether the source database referenced accounts pass the check	✓ Passed
Database parameters	
Whether both the source and destination databases have enabled SSL	✓ Passed
Whether the maximum number of chunks in the destination database is sufficient	✓ Passed
Whether the maximum number of collections has been reached in the destination database	✓ Passed

NOTE

You can proceed to the next step only when all check items are successful. If any alarms are generated, view and confirm the alarm details first before proceeding to the next step.

7. On the displayed page, specify **Start Time**, **Send Notification**, **SMN Topic**, **Synchronization Delay Threshold**, and **Stop Abnormal Tasks After** and confirm that the configured information is correct and click **Submit** to submit the task.

Figure 4-10 Task startup settings



Start Time	[✓ Start upon task creation] Start at a specified time
Send Notifications	On. Please handle exceptions within 48 hours of receiving SMS messages or emails.
* SMN Topic	
Synchronization Delay Threshold(s)	Off
* Stop Abnormal Tasks After	14 — Abnormal tasks run longer than the period you set (unit: day) will automatically stop.

Table 4-8 Task startup settings

Parameter	Description
Start Time	Set **Start Time** to **Start upon task creation** or **Start at a specified time** based on site requirements. The **Start at a specified time** option is recommended. **NOTE** The migration task may affect the performance of the source and destination databases. You are advised to start the task in off-peak hours and reserve two to three days for data verification.
Send Notifications	SMN topic. This parameter is optional. If an exception occurs during migration, the system will send a notification to the specified recipients.
SMN Topic	This parameter is available only after you enable **Send Notification** and create a topic on the SMN console and add a subscriber. For details, see **Simple Message Notification User Guide**.
Synchronization Delay Threshold	During an incremental migration, a synchronization delay indicates a time difference (in seconds) of synchronization between the source and destination database. If the synchronization delay exceeds the threshold you specify, DRS will send alarms to the specified recipients. The value ranges from 0 to 3,600. To avoid repeated alarms caused by the fluctuation of delay, an alarm is sent only after the delay has exceeded the threshold for six minutes. **NOTE** – In the early stages of an incremental migration, there is more delay because more data is waiting to be synchronized. In this situation, no notifications will be sent. – Before setting the delay threshold, enable **Send Notification**. – If the delay threshold is set to 0, no notifications will be sent to the recipient.
Stop Abnormal Tasks After	Number of days after which an abnormal task is automatically stopped. The value must range from 14 to 100. The default value is **14**. **NOTE** Tasks in the abnormal state are still charged. If tasks remain in the abnormal state for a long time, they cannot be resumed. Abnormal tasks run longer than the period you set (unit: day) will automatically stop to avoid unnecessary fees.

8. After the task is submitted, go back to the **Online Migration Management** page to view the task status.

Step 2 Manage the migration task.

The migration task contains two phases: full migration and incremental migration. You can manage them in different phases.

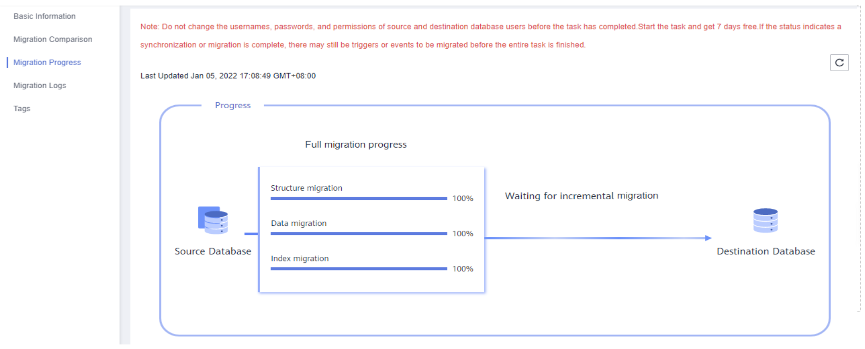
- Full migration

- Viewing the migration progress: Click the target full migration task, and on the **Migration Progress** tab, you can see the migration progress of the structure, data, indexes, and migration objects. When the progress reaches 100%, the migration is complete.
- Viewing migration details: In the migration details, you can view the migration progress of a specific object. If the number of objects is the same as that of migrated objects, the migration is complete. You can view the migration progress of each object in detail. Currently, this function is available only to whitelisted users. You can submit a service ticket to apply for this function.

● Incremental Migration Permission

- Viewing the synchronization delay: After the full migration is complete, an incremental migration starts. On the **Online Migration Management** page, click the target migration task. On the displayed page, click **Migration Progress** to view the synchronization delay of the incremental migration. If the synchronization delay is 0s, the destination database is being synchronized with the source database in real time. You can also view the data consistency on the **Migration Comparison** tab.

Figure 4-11 Viewing the synchronization delay

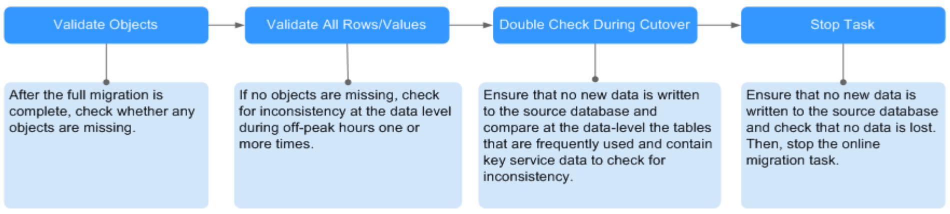


Note: Do not change the usernames, passwords, and permissions of source and destination database users before the task has completed.Start the task and get 7 days free.If the status indicates a synchronization or migration is complete, there may still be triggers or events to be migrated before the entire task is finished.

Last Updated Jan 05, 2022 17:08:49 GMT+08:00

- Viewing the migration results: On the **Online Migration Management** page, click the target migration task. On the displayed page, click **Migration Comparison** and perform a migration comparison in accordance with the comparison process, which should help you determine an appropriate time for migration to minimize service downtime.

Figure 4-12 Database comparison process



After the full migration is complete, check whether any objects are missing.

If no objects are missing, check for inconsistency at the data level during off-peak hours one or more times.

Ensure that no new data is written to the source database and compare at the data-level the tables that are frequently used and contain key service data to check for inconsistency.

Ensure that no new data is written to the source database and check that no data is lost. Then, stop the online migration task.

For details, see **Comparing Migration Items** in *Data Replication Service User Guide*.

Step 3 Cut over services.

You are advised to start the cutover process during off-peak hours. At least one complete data comparison is performed during off-peak hours. To obtain accurate

comparison results, start data comparison at a specified time point during off-peak hours. If it is needed, select **Start at a specified time** for **Comparison Time**. Due to slight time difference and continuous operations on data, inconsistent comparison results may be generated, reducing the reliability and validity of the results.

1. Interrupt services first. If the workload is not heavy, you may not need to interrupt the services.
2. Run the following statement on the source database and check whether any new sessions execute SQL statements within the next 1 to 5 minutes. If there are no new statements executed, the service has been stopped.
   ```
   db.currentOp()
   ```
   NOTE

   The process list queried by the preceding statement includes the connection of the DRS replication instance. If no additional session executes SQL statements, the service has been stopped.
3. On the **Migration Progress** page, view the synchronization delay. When the delay is displayed as 0s and remains stable for a period, then you can perform a data-level comparison between the source and destination databases. For details about the time required, refer to the results of the previous comparison.
   - If there is enough time, compare all objects.
   - If there is not enough time, use the data-level comparison to compare the tables that are frequently used and that contain key business data or inconsistent data.
4. Determine an appropriate time to cut the services over to the destination database. After services are restored and available, the migration is complete.

Step 4 Stop or delete the migration task.

1. Stopping the migration task. After databases and services are migrated to the destination database, to prevent operations on the source database from being synchronized to the destination database to overwrite data, you can stop the migration task. This operation only deletes the replication instance, and the migration task is still displayed in the task list. You can view or delete the task. DRS will not charge for this task after you stop it.
2. Delete the migration task. After the migration task is complete, you can delete it. After the migration task is deleted, it will no longer be displayed in the task list.

----End

5 From On-Premises MongoDB to DDS

DRS helps you migrate data from on-premises MongoDB databases to DDS on the current cloud. With DRS, you can migrate databases online with zero downtime and your services and databases can remain operational during migration.

This section describes how to use DRS to migrate an on-premises MongoDB database to DDS on the current cloud. The following network types are supported:

- VPN
- Public network

Resource Planning

Table 5-1 Resource planning

Category	Subcategory	Planned Value	Description
VPC	VPC name	vpc-dds	Specify a name that is easy to identify.
	Region	AP-Singapore	To achieve lower network latency, select the region nearest to you.
	AZ	AZ1	-
	Subnet	10.0.0.0/24	Select a subnet with sufficient network resources.
	Subnet name	subnet-default	Specify a name that is easy to identify.
On-premises MongoDB database	Database version	MongoDB 4.4	Specify a name that is easy to identify.
DDS	DDS instance name	dds-test	Specify a name that is easy to identify.
	DB engine	DDS	-
	DB engine version	4.4	-

Category	Subcategory	Planned Value	Description
	AZ type	Single-AZ	-
	AZ	AZ1	-
	Instance specifications	Enhanced II	-
	CPU architecture	x86 8 vCPUs | 32 GB	-
DRS migration task	Task name	DRS-dds	Specify a name that is easy to identify.
	Source DB engine	MongoDB	In this example, the source is an on-premises MongoDB database.
	Destination DB engine	DDS	In this example, a DDS instance serves as the destination database.
	Network type	Public network	Public network is used in this example.

Diagram

Figure 5-1 VPN

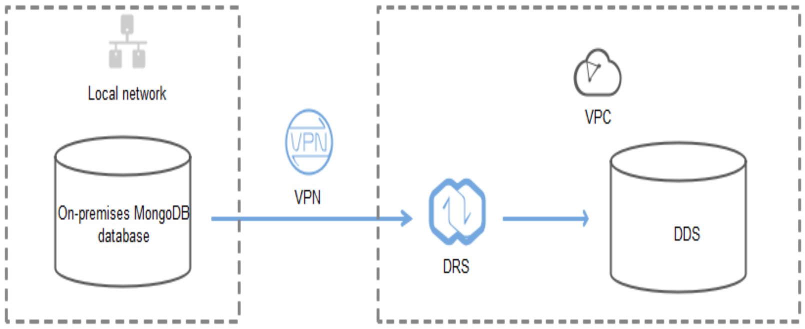
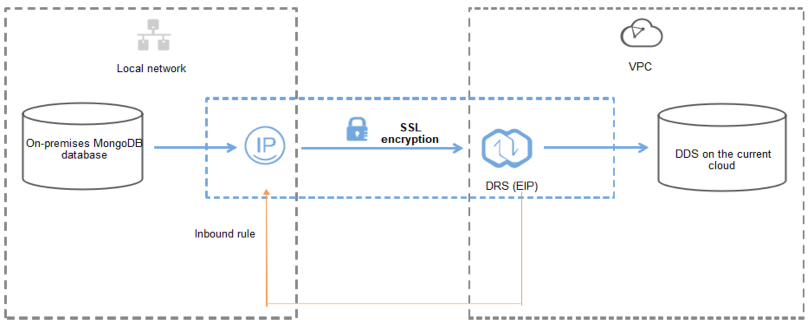
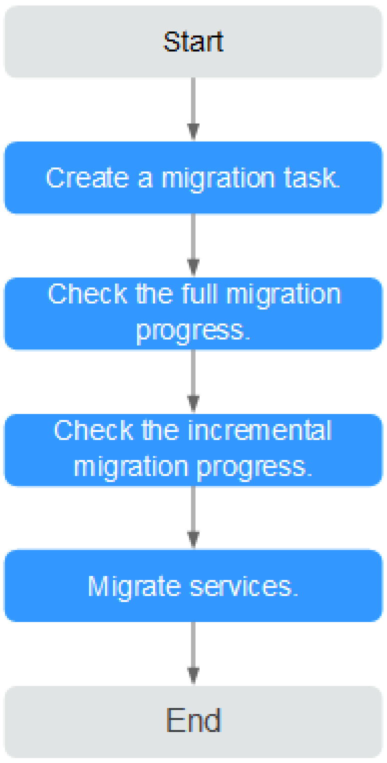


Figure 5-2 Public network+SSL connection



Migration Process

Figure 5-3 Flowchart


Start
Create a migration task.
Check the full migration progress.
Check the incremental migration progress.
Migrate services.
End

Migration Suggestions (Important)

- Database migration is closely impacted by a wide range of environmental and operational factors. To ensure the migration goes smoothly, perform a test run before the actual migration to help you detect and resolve any potential issues in advance. Recommendations on how to minimize any potential impacts on your data base are provided in this section.
- It is strongly recommended that you start your migration task during off-peak hours. A less active database is easier to migrate successfully. If the data is fairly static, there is less likely to be any severe performance impacts during the migration.

Notes on Migration (Important)

NOTICE

Before creating a migration task, read the migration notes carefully.

For details, see **precautions** on using specific migration tasks in *Data Replication Service Real-Time Migration*.

Preparations

1. Permissions

   Table 5-2 lists the permissions required for the source and destination databases when migrating data from on-premises MongoDB databases to DDS DB instances.

Table 5-2 Required permissions

Database	Full Migration Permission	Full+Incremental Migration Permission
Source	• Replica set: The source database user must have the read permission for the database to be migrated. • Single node: The source database user must have the read permission for the database to be migrated. • Cluster: The source database user must have the read permission for the databases to be migrated and the config database. • To migrate accounts and roles of the source database, the source database user must have the read permission for the **system.users** and **system.roles** system tables of the admin database.	• Replica set: The source database user must have the read permission for the databases to be migrated and the local database. • Single node: The source database user must have the read permission for the databases to be migrated and the local database. • Cluster: The source mongos node user must have the readAnyDatabase permission for the databases to be migrated and the config database. The source shard node user must have the readAnyDatabase permission for the admin database and the read permission for the local database. • To migrate accounts and roles of the source database, the source database user must have the read permission for the **system.users** and **system.roles** system tables of the admin database.

Database	Full Migration Permission	Full+Incremental Migration Permission
Destination	The destination database user must have the dbAdminAnyDatabase permission for the admin database and the readWrite permission for the destination database. If the destination database is a cluster instance, the migration account must have the read permission for the config database.	

   – Source database permissions:

     The source database user must have all the required permissions listed in **Table 5-2**. If the permissions are insufficient, create a user that has all of the permissions on the source database.

   – Destination database permissions:

     If the destination database is a DDS database, the initial account can be used.

2. Network settings
   – Source database network settings:

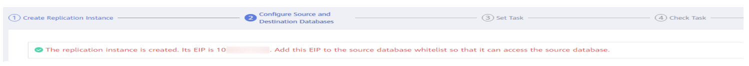
     You can migrate on-premises MongoDB databases to DDS through a VPN or public network. Enable public accessibility or establish a VPN for local MongoDB databases based on the site requirements. You are advised to migrate data through a public network, which is more convenient and cost-effective.

   – Destination database network settings:
     - If the source database accesses the destination database through a VPN, enable the VPN service first so that the source database can communicate with the destination DDS network.
     - If you access the DDS DB instance through a public network, no network settings are required.

3. Security rules
   a. Source database network settings:
     - The replication instance needs to be able to access the source DB. That means that the EIP of the replication instance must be on the whitelist of the source MongoDB instance. Before configuring the network whitelist for the source database, you need to obtain the EIP of the DRS replication instance.

       After creating a replication instance on the DRS console, you can find the EIP on the **Configure Source and Destination Databases** page as shown in **Figure 5-4**.

Figure 5-4 EIP of the replication instance


1 Create Replication Instance
2 Configure Source and Destination Databases
3 Set Task
4 Check Task

The replication instance is created. Its EIP is 10[redacted]. Add this EIP to the source database whitelist so that it can access the source database.

You can also add 0.0.0.0/0 to the source database whitelist to allow any IP address to access the source database but this action may result in security risks.

- If the migration is performed over a VPN network, add the private IP address of the DRS replication instance to the whitelist of the source database to enable the source database to communicate with the destination database.

If you do take this step, then once the migration is complete, you should delete this item from the whitelist or your system will insecure.

b. Destination database security group settings:

By default, the destination database and the DRS replication instance are in the same VPC and can communicate with each other. No further configuration is required.

4. Other

You need to export the user information of the MongoDB database first and manually add it to the destination DDS DB instance because the user information will not be migrated.

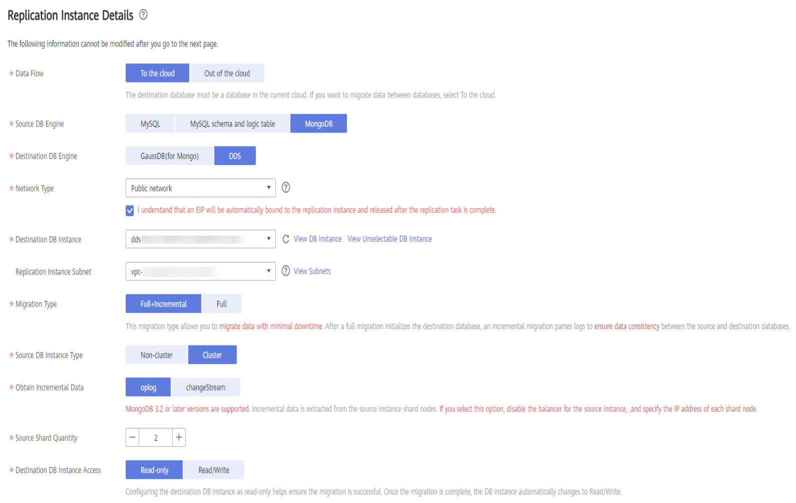
Migration Procedure

The following describes how to use DRS to migrate an on-premises MongoDB database to a DDS DB instance.

Step 1 Create a migration task.

1. Log in to the management console and choose **Databases** > **Data Replication Service** to go to the DRS console.
2. On the **Online Migration Management** page, click **Create Migration Task**.
3. On the **Create Replication Instance** page, configure the task details, recipient, and replication instance and click **Next**.

Figure 5-5 Replication instance information



Replication Instance Details

The following information cannot be modified after you go to the next page.

* Data Flow	[✓ To the cloud] Out of the cloud
	The destination database must be a database in the current cloud. If you want to migrate data between databases, select To the cloud.
* Source DB Engine	MySQL MySQL schema and logic table [✓ MongoDB]
* Destination DB Engine	GaussDB(for Mongo) [✓ DDS]
* Network Type	Public network
	✓ I understand that an EIP will be automatically bound to the replication instance and released after the replication task is complete.
* Destination DB Instance	dds-[redacted] View DB Instance View Unselectable DB Instance
Replication Instance Subnet	vpc-[redacted] View Subnets
* Migration Type	[✓ Full+Incremental] Full
	This migration type allows you to migrate data with minimal downtime. After a full migration initializes the destination database, an incremental migration parses logs to ensure data consistency between the source and destination databases.
* Source DB Instance Type	Non-cluster [✓ Cluster]
* Obtain Incremental Data	[✓ oplog] changeStream
	MongoDB 3.2 or later versions are supported. Incremental data is extracted from the source instance shard nodes. If you select this option, disable the balancer for the source instance, ,and specify the IP address of each shard node.
* Source Shard Quantity	2
* Destination DB Instance Access	[✓ Read-only] Read/Write
	Configuring the destination DB instance as read-only helps ensure the migration is successful. Once the migration is complete, the DB instance automatically changes to Read/Write.

Table 5-3 Task settings

Parameter	Description
Region	The region where the replication instance is deployed. You can change the region. To reduce latency and improve access speed, select the region closest to your workloads.
Project	The project corresponds to the current region and can be changed.
Task Name	The task name consists of 4 to 50 characters, starts with a letter, and can contain only letters (case-insensitive), digits, hyphens (-), and underscores (_).
Description	The description consists of a maximum of 256 characters and cannot contain the following special characters: =<>&'\"

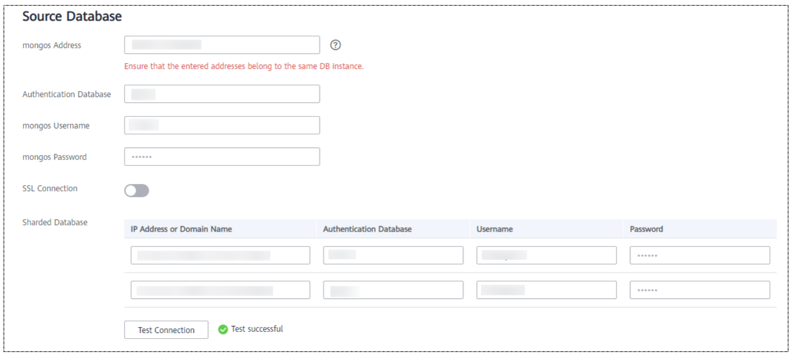
Table 5-4 Replication instance settings

Parameter	Description
Data Flow	Select **To the cloud**.
Source DB Engine	Select **MongoDB**.
Destination DB Engine	Select **DDS**.
Network Type	Select **Public network**. Enabling SSL is recommended. It may slow down the migration by 20% to 30% but it ensures data security.
Destination DB Instance	The DDS DB instance you purchased.
Migration Type	– **Full** It migrates all data at one time. If you perform a full migration, you are advised to stop operations on the source database. Otherwise, data generated in the source database during the migration will not be synchronized to the destination database. – **Full+Incremental** An incremental migration can keep data consistency after a full migration is complete.
Source DB Instance Type	If you select **Full+Incremental** for **Migration Type**, set this parameter based on the source database. – If the source database is a cluster instance, set this parameter to **Cluster**. – If the source database is a replica set or a single node instance, set this parameter to **Non-cluster**.

Parameter	Description
Obtain Incremental Data	This parameter is available for configuration if **Source DB Instance Type** is set to **Cluster**. You can determine how to capture data changes during the incremental synchronization. – oplog: For MongoDB 3.2 or later, DRS directly connects to each shard of the source DB instance to extract data. If you select this mode, you must disable the balancer of the source instance. When testing the connection, you need to enter the connection information of each shard node of the source instance. – changeStream: This method is recommended. For MongoDB 4.0 and later, DRS connects to mongos nodes of the source instance to extract data. If you select this method, you must enable the WiredTiger storage engine of the source instance. **NOTE** Only whitelisted users can use **changeStream**. To use this function, submit a service ticket. In the upper right corner of the management console, choose **Service Tickets** > **Create Service Ticket** to submit a service ticket.
Source Shard Quantity	If **Source DB Instance Type** is set to **Cluster** and **Obtain Incremental Data** is set to **oplog**, enter the number of source shard nodes. The default minimum number of source DB instances is 2 and the maximum number is 32. You can set this parameter based on the number of source database shards.

4. On the **Configure Source and Destination Databases** page, wait until the replication instance is created. Then, specify source and destination database information and click **Test Connection** for both the source and destination databases to check whether they have been connected to the replication instance. After the connection tests are successful, select the check box before the agreement and click **Next**.

Figure 5-6 Source database information



Source Database

mongos Address	[redacted]
	Ensure that the entered addresses belong to the same DB instance.
Authentication Database	[redacted]
mongos Username	[redacted]
mongos Password	••••••
SSL Connection	
Sharded Database	

IP Address or Domain Name	Authentication Database	Username	Password
[redacted]	[redacted]	[redacted]	••••••
[redacted]	[redacted]	[redacted]	••••••

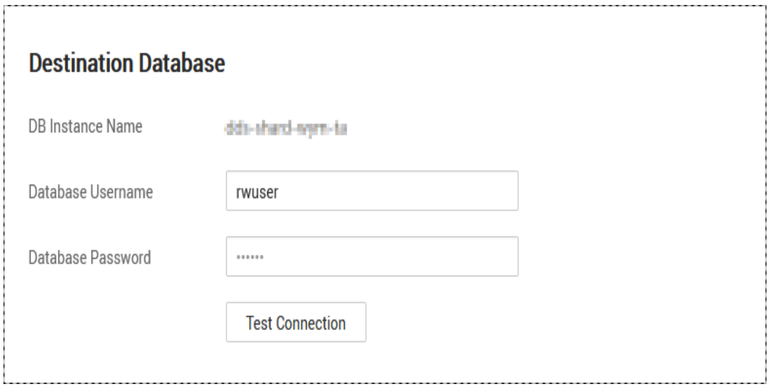
Test Connection ✓ Test successful

Table 5-5 Source database settings

Parameter	Description
mongos Address	IP address or domain name of the source database in the **IP address/Domain name:Port** format. The port of the source database. Range: 1 - 65534 You can enter a maximum of three groups of IP addresses or domain names of the source database. Separate multiple values with commas (,). For example: 192.168.0.1:8080,192.168.0.2:8080. Ensure that the entered IP addresses or domain names belong to the same sharded cluster. **NOTE** If multiple IP addresses or domain names are entered, the test connection is successful as long as one IP address or domain name is accessible. Therefore, you must ensure that the IP address or domain name is correct.
Authentication Database	The name of the authentication database. For example: The default authentication database of Huawei Cloud DDS instance is **admin**.
mongos Username	A username for the source database.
mongos Password	The password for the source database username.
SSL Connection	SSL encrypts the connections between the source and destination databases. If SSL is enabled, upload the SSL CA root certificate.
Sharded Database	Enter the information about the sharded databases in the source database.

– Destination database configuration

Figure 5-7 Destination database information


Destination Database

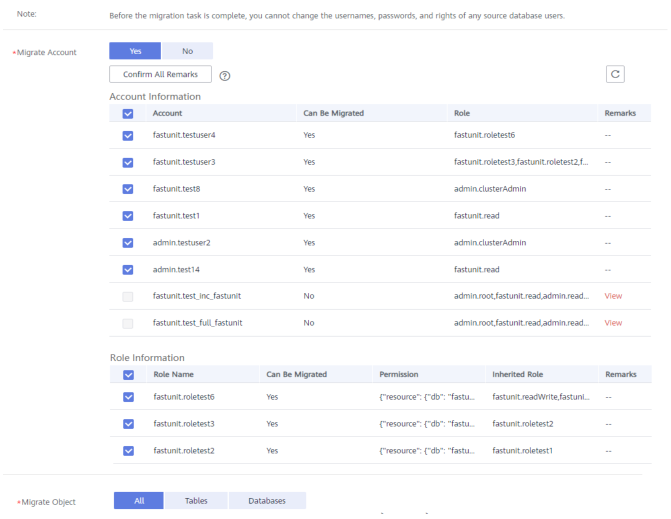
DB Instance Name	dds-shard-wym-ia
Database Username	rwuser
Database Password	******

Table 5-6 Destination database settings

Parameter	Description
DB Instance Name	The DB instance you selected when creating the migration task and cannot be changed.
Database Username	The username for accessing the destination database.
Database Password	The password for the database username.

5. On the **Set Task** page, select migration objects and click **Next**.

Figure 5-8 Migration object



Note: Before the migration task is complete, you cannot change the usernames, passwords, and rights of any source database users.

*Migrate Account [✓ Yes] No

Confirm All Remarks

Account Information

✓	Account	Can Be Migrated	Role	Remarks
✓	fastunit.testuser4	Yes	fastunit.roletest6	--
✓	fastunit.testuser3	Yes	fastunit.roletest3,fastunit.roletest2,f...	--
✓	fastunit.test8	Yes	admin.clusterAdmin	--
✓	fastunit.test1	Yes	fastunit.read	--
✓	admin.testuser2	Yes	admin.clusterAdmin	--
✓	admin.test14	Yes	fastunit.read	--
	fastunit.test_inc_fastunit	No	admin.root,fastunit.read,admin.read...	View
	fastunit.test_full_fastunit	No	admin.root,fastunit.read,admin.read...	View

Role Information

✓	Role Name	Can Be Migrated	Permission	Inherited Role	Remarks
✓	fastunit.roletest6	Yes	{"resource": {"db": "fastu...	fastunit.readWrite,fastuni...	--
✓	fastunit.roletest3	Yes	{"resource": {"db": "fastu...	fastunit.roletest2	--
✓	fastunit.roletest2	Yes	{"resource": {"db": "fastu...	fastunit.roletest1	--

*Migrate Object [✓ All] Tables Databases

Table 5-7 Migration object

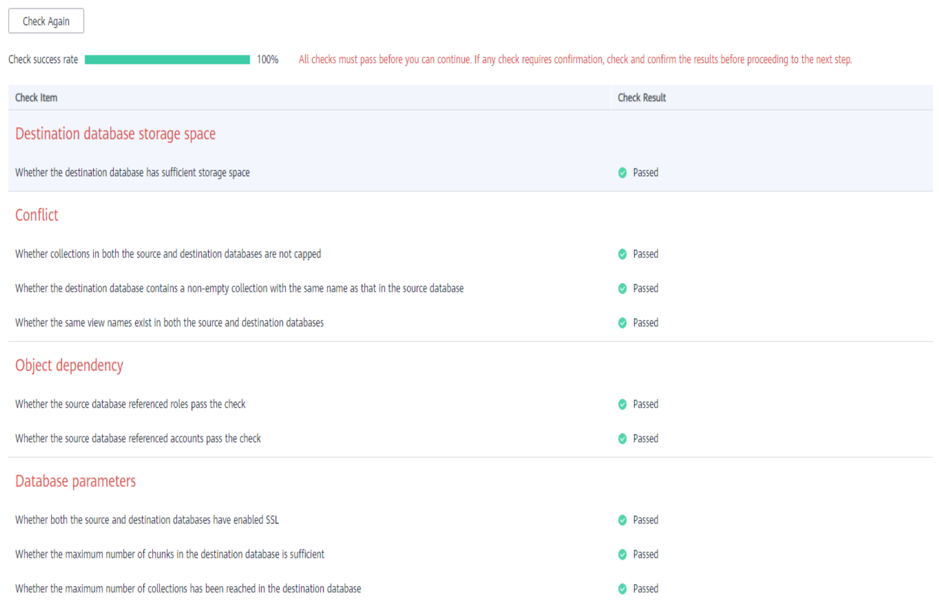
Parameter	Description
Migrate Account	There are accounts that can be migrated completely and accounts that cannot be migrated. You can choose whether to migrate the accounts. Accounts that cannot be migrated or accounts that are not selected will not exist in the destination database. Ensure that your services will not be affected by these accounts. – **Yes** If you choose to migrate accounts, see **Migrating Accounts** in *Data Replication Service User Guide* to migrate database users and roles. – **No** During the migration, accounts and roles are not migrated.

Parameter	Description
Migrate Object	You can choose to migrate all objects, tables, or databases based on your service requirements. – **All**: All objects in the source database are migrated to the destination database. After the migration, the object names will remain the same as those in the source database and cannot be modified. – **Tables**: The selected table-level objects will be migrated. – **Databases**: The selected database-level objects will be migrated. If the source database is changed, click ↻ in the upper right corner before selecting migration objects to ensure that the objects to be selected are from the changed source database. **NOTE** – If you choose not to migrate all of the databases, the migration may fail because the objects, such as stored procedures and views, in the database to be migrated may have dependencies on other objects that are not migrated. To ensure a successful migration, you are advised to migrate all of the databases. – When you select an object, the spaces before and after the object name are not displayed. If there are two or more consecutive spaces in the middle of the object name, only one space is displayed. – The search function can help you quickly select the required database objects.

6. On the **Check Task** page, check the migration task.
   - If any check fails, review the cause and rectify the fault. After the fault is rectified, click **Check Again**.

     For details about how to handle check failures, see **Checking Whether the Source Database Is Connected** in *Data Replication Service User Guide*.
   - If all check items are successful, click **Next**.

Figure 5-9 Task Check



Check Again

Check success rate 100% All checks must pass before you can continue. If any check requires confirmation, check and confirm the results before proceeding to the next step.

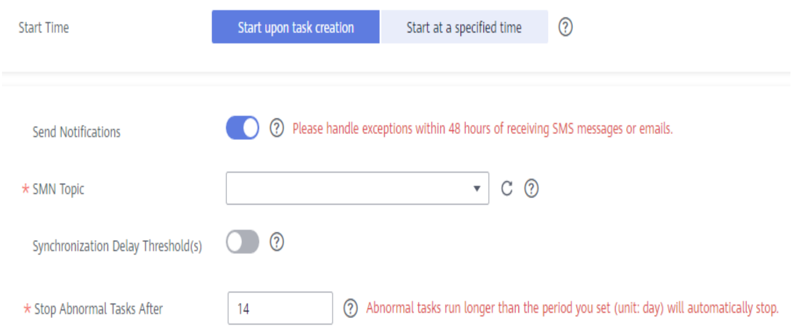
Check Item	Check Result
Destination database storage space	
Whether the destination database has sufficient storage space	Passed
Conflict	
Whether collections in both the source and destination databases are not capped	Passed
Whether the destination database contains a non-empty collection with the same name as that in the source database	Passed
Whether the same view names exist in both the source and destination databases	Passed
Object dependency	
Whether the source database referenced roles pass the check	Passed
Whether the source database referenced accounts pass the check	Passed
Database parameters	
Whether both the source and destination databases have enabled SSL	Passed
Whether the maximum number of chunks in the destination database is sufficient	Passed
Whether the maximum number of collections has been reached in the destination database	Passed

NOTE

You can proceed to the next step only when all check items are successful. If any alarms are generated, view and confirm the alarm details first before proceeding to the next step.

7. On the displayed page, specify **Start Time**, **Send Notification**, **SMN Topic**, **Synchronization Delay Threshold**, and **Stop Abnormal Tasks After** and confirm that the configured information is correct and click **Submit** to submit the task.

Figure 5-10 Task startup settings



Start Time	Start upon task creation / Start at a specified time
Send Notifications	Please handle exceptions within 48 hours of receiving SMS messages or emails.
* SMN Topic	
Synchronization Delay Threshold(s)	
* Stop Abnormal Tasks After	14 — Abnormal tasks run longer than the period you set (unit: day) will automatically stop.

Table 5-8 Task startup settings

Parameter	Description
Start Time	Set **Start Time** to **Start upon task creation** or **Start at a specified time** based on site requirements. The **Start at a specified time** option is recommended. **NOTE** The migration task may affect the performance of the source and destination databases. You are advised to start the task in off-peak hours and reserve two to three days for data verification.
Send Notifications	SMN topic. This parameter is optional. If an exception occurs during migration, the system will send a notification to the specified recipients.
SMN Topic	This parameter is available only after you enable **Send Notification** and create a topic on the SMN console and add a subscriber. For details, see **Simple Message Notification User Guide**.
Synchronization Delay Threshold	During an incremental migration, a synchronization delay indicates a time difference (in seconds) of synchronization between the source and destination database. If the synchronization delay exceeds the threshold you specify, DRS will send alarms to the specified recipients. The value ranges from 0 to 3,600. To avoid repeated alarms caused by the fluctuation of delay, an alarm is sent only after the delay has exceeded the threshold for six minutes. **NOTE** – In the early stages of an incremental migration, there is more delay because more data is waiting to be synchronized. In this situation, no notifications will be sent. – Before setting the delay threshold, enable **Send Notification**. – If the delay threshold is set to 0, no notifications will be sent to the recipient.
Stop Abnormal Tasks After	Number of days after which an abnormal task is automatically stopped. The value must range from 14 to 100. The default value is **14**. **NOTE** Tasks in the abnormal state are still charged. If tasks remain in the abnormal state for a long time, they cannot be resumed. Abnormal tasks run longer than the period you set (unit: day) will automatically stop to avoid unnecessary fees.

8. After the task is submitted, go back to the **Online Migration Management** page to view the task status.

Step 2 Manage the migration task.

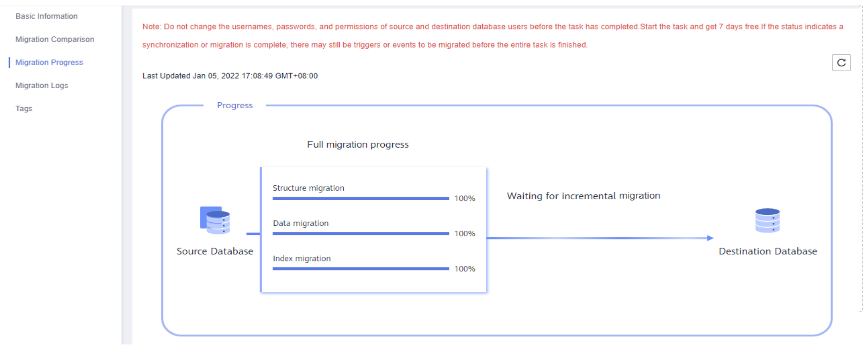
The migration task contains two phases: full migration and incremental migration. You can manage them in different phases.

- Full migration

- Viewing the migration progress: Click the target full migration task, and on the **Migration Progress** tab, you can see the migration progress of the structure, data, indexes, and migration objects. When the progress reaches 100%, the migration is complete.
- Viewing migration details: In the migration details, you can view the migration progress of a specific object. If the number of objects is the same as that of migrated objects, the migration is complete. You can view the migration progress of each object in detail. Currently, this function is available only to whitelisted users. You can submit a service ticket to apply for this function.

- Incremental Migration Permission
  - Viewing the synchronization delay: After the full migration is complete, an incremental migration starts. On the **Online Migration Management** page, click the target migration task. On the displayed page, click **Migration Progress** to view the synchronization delay of the incremental migration. If the synchronization delay is 0s, the destination database is being synchronized with the source database in real time. You can also view the data consistency on the **Migration Comparison** tab.

Figure 5-11 Viewing the synchronization delay

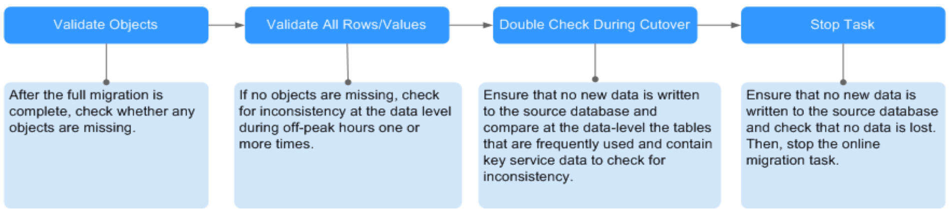


Note: Do not change the usernames, passwords, and permissions of source and destination database users before the task has completed.Start the task and get 7 days free.If the status indicates a synchronization or migration is complete, there may still be triggers or events to be migrated before the entire task is finished.

Last Updated Jan 05, 2022 17:08:49 GMT+08:00

  - Viewing the migration results: On the **Online Migration Management** page, click the target migration task. On the displayed page, click **Migration Comparison** and perform a migration comparison in accordance with the comparison process, which should help you determine an appropriate time for migration to minimize service downtime.

Figure 5-12 Database comparison process


Validate Objects
Validate All Rows/Values
Double Check During Cutover
Stop Task
After the full migration is complete, check whether any objects are missing.
If no objects are missing, check for inconsistency at the data level during off-peak hours one or more times.
Ensure that no new data is written to the source database and compare at the data-level the tables that are frequently used and contain key service data to check for inconsistency.
Ensure that no new data is written to the source database and check that no data is lost. Then, stop the online migration task.

For details, see **Comparing Migration Items** in *Data Replication Service User Guide*.

Step 3 Cut over services.

You are advised to start the cutover process during off-peak hours. At least one complete data comparison is performed during off-peak hours. To obtain accurate

comparison results, start data comparison at a specified time point during off-peak hours. If it is needed, select **Start at a specified time** for **Comparison Time**. Due to slight time difference and continuous operations on data, inconsistent comparison results may be generated, reducing the reliability and validity of the results.

1. Interrupt services first. If the workload is not heavy, you may not need to interrupt the services.
2. Run the following statement on the source database and check whether any new sessions execute SQL statements within the next 1 to 5 minutes. If there are no new statements executed, the service has been stopped.
   ```
   db.currentOp()
   ```
   NOTE

   The process list queried by the preceding statement includes the connection of the DRS replication instance. If no additional session executes SQL statements, the service has been stopped.
3. On the **Migration Progress** page, view the synchronization delay. When the delay is displayed as 0s and remains stable for a period, then you can perform a data-level comparison between the source and destination databases. For details about the time required, refer to the results of the previous comparison.
   - If there is enough time, compare all objects.
   - If there is not enough time, use the data-level comparison to compare the tables that are frequently used and that contain key business data or inconsistent data.
4. Determine an appropriate time to cut the services over to the destination database. After services are restored and available, the migration is complete.

Step 4 Stop or delete the migration task.

1. Stopping the migration task. After databases and services are migrated to the destination database, to prevent operations on the source database from being synchronized to the destination database to overwrite data, you can stop the migration task. This operation only deletes the replication instance, and the migration task is still displayed in the task list. You can view or delete the task. DRS will not charge for this task after you stop it.
2. Delete the migration task. After the migration task is complete, you can delete it. After the migration task is deleted, it will no longer be displayed in the task list.

----End

6 From ECS-hosted MongoDB to DDS

DRS helps you migrate data from MongoDB databases on ECSs to DDS instances on the current cloud. With DRS, you can migrate databases online with zero downtime and your services and databases can remain operational during migration.

This section describes how to use DRS to migrate data from an ECS database to a DDS instance on the current cloud. The following network scenarios are supported:

- Source and destination databases are in the same VPC.
- Source and destination databases are in different VPCs.

Resource Planning

Table 6-1 Resource planning

Category	Subcategory	Planned Value	Description
VPC	VPC name	vpc-dds	Specify a name that is easy to identify.
	Region	AP-Singapore	To achieve lower network latency, select the region nearest to you.
	AZ	AZ1	-
	Subnet	10.0.0.0/24	Select a subnet with sufficient network resources.
	Subnet name	subnet-default	Specify a name that is easy to identify.
ECS	ECS name	ecs-mongodb	Specify a name that is easy to identify.
	Specifications	s6.xlarge.2 4vCPUs|8GiB	The specifications in the left column are selected in this example. In your project, select specifications meeting your workload requirements. For details, see **ECS Specifications**.
	OS	CentOS 7.6 64	-

Category	Subcategory	Planned Value	Description
	System disk	General purpose SSD 40 GiB	-
	Data disk	Ultra-high I/O, 100 GiB	-
	EIP	Buy now	Buy an EIP because the public network is selected for the migration task.
DDS	DDS instance name	dds-test	Specify a name that is easy to identify.
	DB engine	DDS	-
	DB engine version	4.4	-
	AZ type	Single-AZ	-
	AZ	AZ1	-
	Instance specifications	Enhanced II	-
	CPU architecture	x86 8 vCPUs | 32 GB	-
DRS migration task	Task name	DRS-dds	Specify a name that is easy to identify.
	Source DB engine	MongoDB	In this example, the source is a MongoDB database built on an ECS.
	Destination DB engine	DDS	In this example, a DDS instance serves as the destination database.
	Network type	Public network	Public network is used in this example.

Diagram

Figure 6-1 Source and destination databases in the same VPC

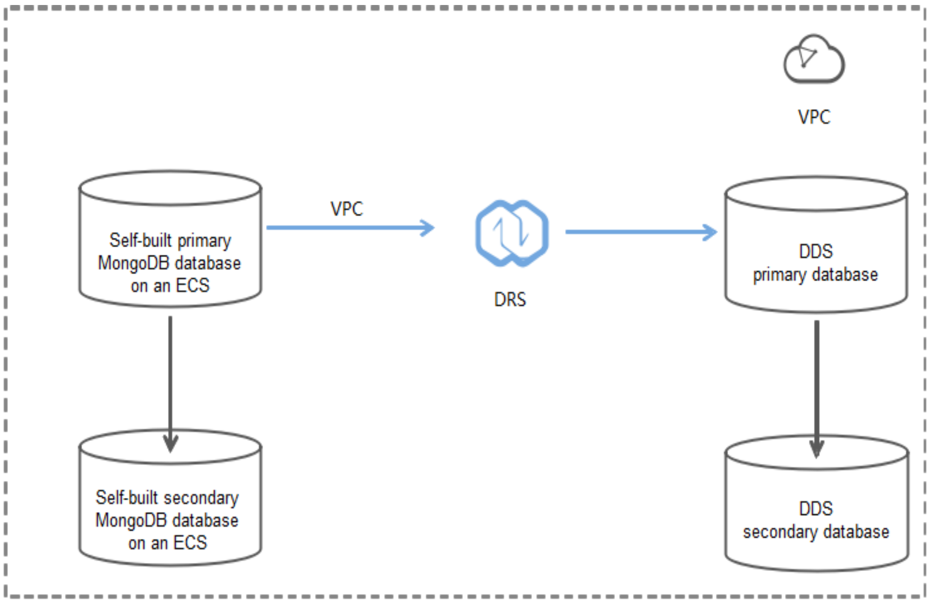
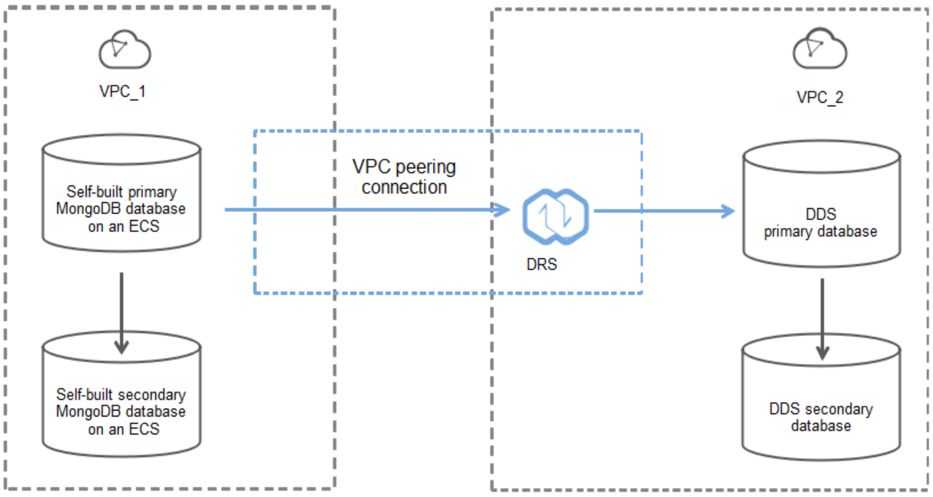
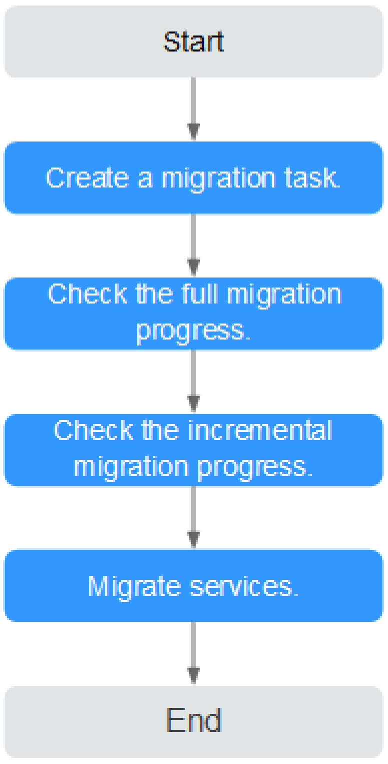

VPC
VPC
Self-built primary MongoDB database on an ECS
DRS
DDS primary database
Self-built secondary MongoDB database on an ECS
DDS secondary database

Figure 6-2 Source and destination databases in the same region and different VPCs


VPC_1
VPC_2
VPC peering connection
Self-built primary MongoDB database on an ECS
DRS
DDS primary database
Self-built secondary MongoDB database on an ECS
DDS secondary database

Migration Process

Figure 6-3 Flowchart


Start
Create a migration task.
Check the full migration progress.
Check the incremental migration progress.
Migrate services.
End

Migration Suggestions (Important)

- Database migration is closely impacted by a wide range of environmental and operational factors. To ensure the migration goes smoothly, perform a test run before the actual migration to help you detect and resolve any potential issues in advance. Recommendations on how to minimize any potential impacts on your data base are provided in this section.
- It is strongly recommended that you start your migration task during off-peak hours. A less active database is easier to migrate successfully. If the data is fairly static, there is less likely to be any severe performance impacts during the migration.

Notes on Migration (Important)

NOTICE

Before creating a migration task, read the migration notes carefully.

For details, see **precautions** on using specific migration tasks in *Data Replication Service Real-Time Migration*.

Preparations

1. Permissions:

   Table 6-2 lists the permissions required for the source and destination databases when migrating data from a MongoDB database on an ECS to DDS on the current cloud.

Table 6-2 Required permissions

Database	Full Migration Permission	Full+Incremental Migration Permission
Source	• Replica set: The source database user must have the read permission for the database to be migrated. • Single node: The source database user must have the read permission for the database to be migrated. • Cluster: The source database user must have the read permission for the databases to be migrated and the config database. • To migrate accounts and roles of the source database, the source database user must have the read permission for the **system.users** and **system.roles** system tables of the admin database.	• Replica set: The source database user must have the read permission for the databases to be migrated and the local database. • Single node: The source database user must have the read permission for the databases to be migrated and the local database. • Cluster: The source mongos node user must have the readAnyDatabase permission for the databases to be migrated and the config database. The source shard node user must have the readAnyDatabase permission for the admin database and the read permission for the local database. • To migrate accounts and roles of the source database, the source database user must have the read permission for the **system.users** and **system.roles** system tables of the admin database.

Database	Full Migration Permission	Full+Incremental Migration Permission
Destination	The destination database user must have the dbAdminAnyDatabase permission for the admin database and the readWrite permission for the destination database. If the destination database is a cluster instance, the migration account must have the read permission for the config database.	

- Source database permissions:

  The source MongoDB database user must have all the required permissions listed in **Table 6-2**. If the permissions are insufficient, create a user that has all of the permissions on the source database.
- Destination database permissions:

  The initial account of the DDS instance has the required permissions.

2. Network settings
   - The source database and destination DDS DB instance must be in the same region.
   - The source database and destination DDS DB instance can be either in the same VPC or different VPCs.
     - If the source and destination databases are in different VPCs, the subnets of the source and destination databases are required to be in different CIDR blocks. You need to create a VPC peering connection between the two VPCs.

       For details, see **VPC Peering Connection Overview** in the *Virtual Private Cloud User Guide*.
     - If the source and destination databases are in the same VPC, the networks are interconnected by default.
3. Security rules
   - In the same VPC, the network is connected by default. You do not need to set a security group.
   - In different VPCs, establish a VPC peering connection between the two VPCs. You do not need to set a security group.
4. Other

   You need to export the user information of the MongoDB database first and manually add it to the destination DDS DB instance because the user information will not be migrated.

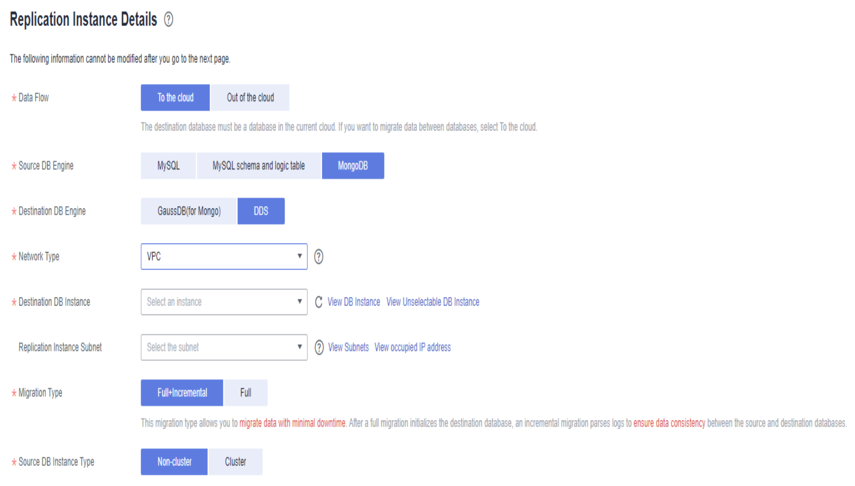
Migration Procedure

Step 1 Create a migration task.

1. Log in to the management console and choose **Databases** > **Data Replication Service** to go to the DRS console.

2. On the **Online Migration Management** page, click **Create Migration Task**.
3. On the **Create Replication Instance** page, configure the task details, recipient, and replication instance and click **Next**.

Figure 6-4 Replication instance information



Replication Instance Details

The following information cannot be modified after you go to the next page.

* Data Flow	[✓ To the cloud] Out of the cloud The destination database must be a database in the current cloud. If you want to migrate data between databases, select To the cloud.
* Source DB Engine	MySQL / MySQL schema and logic table / [✓ MongoDB]
* Destination DB Engine	GaussDB(for Mongo) / [✓ DDS]
* Network Type	VPC
* Destination DB Instance	Select an instance
Replication Instance Subnet	Select the subnet
* Migration Type	[✓ Full+Incremental] Full This migration type allows you to migrate data with minimal downtime. After a full migration initializes the destination database, an incremental migration parses logs to ensure data consistency between the source and destination databases.
* Source DB Instance Type	[✓ Non-cluster] Cluster

Table 6-3 Task settings

Parameter	Description
Region	The region where the replication instance is deployed. You can change the region. To reduce latency and improve access speed, select the region closest to your workloads.
Project	The project corresponds to the current region and can be changed.
Task Name	The task name consists of 4 to 50 characters, starts with a letter, and can contain only letters (case-insensitive), digits, hyphens (-), and underscores (_).
Description	The description consists of a maximum of 256 characters and cannot contain the following special characters: =<>&'\"

Table 6-4 Replication instance information

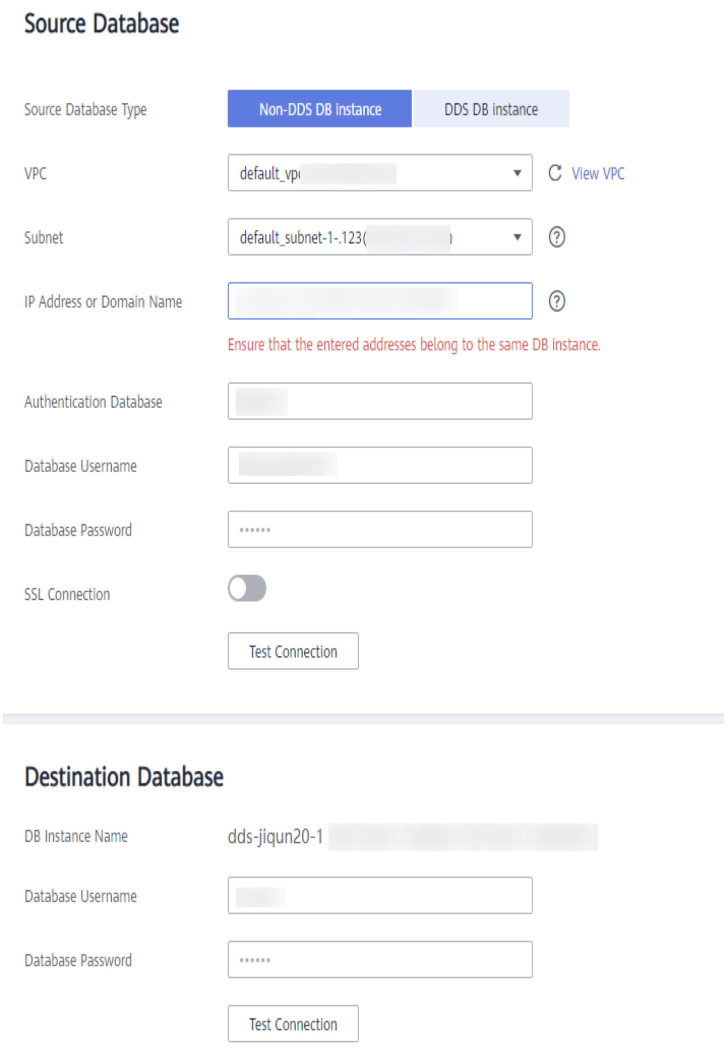
Parameter	Description
Data Flow	**To the cloud**
Source DB Engine	Select **MongoDB**.
Destination DB Engine	Select **DDS**.
Network Type	Select **VPC**.

Parameter	Description
Destination DB Instance	The DDS DB instance you purchased.
Migration Type	Select **Full+Incremental** as an example: – **Full**: This migration type is suitable for scenarios where a service interruption is acceptable. All objects and data in non-system databases are migrated to the destination database at one time. The objects include tables, views, and stored procedures. **NOTE** If you perform a full migration, you are advised to stop operations on the source database. Otherwise, data generated in the source database during the migration will not be synchronized to the destination database. – **Full+Incremental**: This migration type allows you to migrate data without interrupting services. After a full migration initializes the destination database, an incremental migration initiates and parses logs to ensure data consistency between the source and destination databases. **NOTE** If you select the **Full+Incremental** migration type, data generated during the full migration will be synchronized to the destination database with zero downtime, ensuring that both the source and destination databases remain accessible.
Source DB Instance Type	If you select **Full+Incremental** for **Migration Type**, set this parameter based on the source database. **Non-cluster** is selected as an example. – If the source database is a cluster instance, set this parameter to **Cluster**. – If the source database is a replica set or a single node instance, set this parameter to **Non-cluster**.

Parameter	Description
Obtain Incremental Data	This parameter is available for configuration if **Source DB Instance Type** is set to **Cluster**. You can determine how to capture data changes during the incremental synchronization. – oplog: For MongoDB 3.2 or later, DRS directly connects to each shard of the source DB instance to extract data. If you select this mode, you must disable the balancer of the source instance. When testing the connection, you need to enter the connection information of each shard node of the source instance. – changeStream: This method is recommended. For MongoDB 4.0 and later, DRS connects to mongos nodes of the source instance to extract data. If you select this method, you must enable the WiredTiger storage engine of the source instance. **NOTE** Only whitelisted users can use **changeStream**. To use this function, submit a service ticket. In the upper right corner of the management console, choose **Service Tickets** > **Create Service Ticket** to submit a service ticket.
Source Shard Quantity	If **Source DB Instance Type** is set to **Cluster** and **Obtain Incremental Data** is set to **oplog**, enter the number of source shard nodes. The default minimum number of source DB instances is 2 and the maximum number is 32. You can set this parameter based on the number of source database shards.

4. On the **Configure Source and Destination Databases** page, wait until the replication instance is created. Then, specify source and destination database information and click **Test Connection** for both the source and destination databases to check whether they have been connected to the replication instance. After the connection tests are successful, select the check box before the agreement and click **Next**.

Figure 6-5 Source and destination database details


Source Database

Source Database Type: Non-DDS DB Instance | DDS DB instance

VPC: default_vp

Subnet: default_subnet-1-.123(

IP Address or Domain Name:

Ensure that the entered addresses belong to the same DB instance.

Authentication Database:

Database Username:

Database Password: ••••••

SSL Connection:

Destination Database

DB Instance Name: dds-jiqun20-1

Database Username:

Database Password: ••••••

Table 6-5 Source database information

Parameter	Description
Source Database Type	Select **Self-built on ECS**.
VPC	A dedicated virtual network in which the source database is located. It isolates networks for different services. You can select an existing VPC or create a VPC. For details on how to create a VPC, see **Creating a VPC**.
Subnet	A subnet provides dedicated network resources that are logically isolated from other networks, improving network security. The subnet must be in the AZ where the source database resides. You need to enable DHCP for creating the source database subnet. For details on how to create a VPC, see the **Creating a VPC** section in the *Virtual Private Cloud User Guide*.

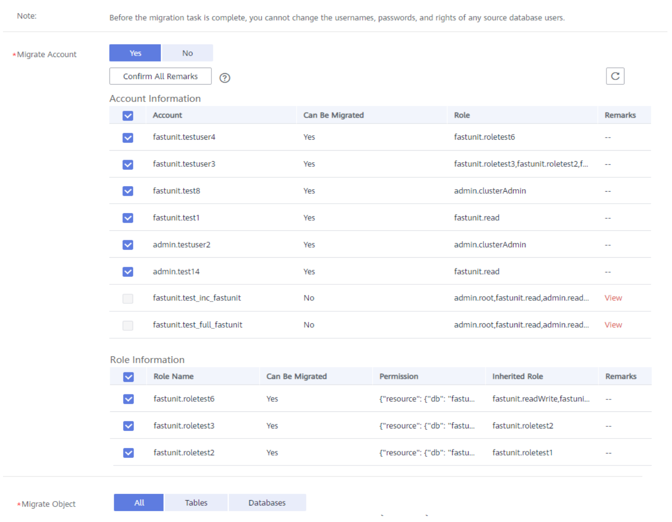
Parameter	Description
IP Address or Domain Name	The IP address or domain name of the source database.
Port	The port of the source database. Range: 1 - 65535
Database Username	A username for the source database.
Database Password	The password for the database username.
SSL Connection	To improve data security during the migration, you are advised to enable SSL to encrypt migration links and upload a CA certificate.

Table 6-6 Destination database information

Parameter	Description
DB Instance Name	The DDS DB instance you have selected during the migration task creation is displayed by default and cannot be changed.
Database Username	The username for accessing the destination DDS DB instance.
Database Password	The password for the database username.

5. On the **Set Task** page, select migration objects and click **Next**.

Figure 6-6 Migration object



Note: Before the migration task is complete, you cannot change the usernames, passwords, and rights of any source database users.

*Migrate Account [✓ Yes] No

Confirm All Remarks

Account Information

✓	Account	Can Be Migrated	Role	Remarks
✓	fastunit.testuser4	Yes	fastunit.roletest6	--
✓	fastunit.testuser3	Yes	fastunit.roletest3,fastunit.roletest2,f...	--
✓	fastunit.test8	Yes	admin.clusterAdmin	--
✓	fastunit.test1	Yes	fastunit.read	--
✓	admin.testuser2	Yes	admin.clusterAdmin	--
✓	admin.test14	Yes	fastunit.read	--
	fastunit.test_inc_fastunit	No	admin.root,fastunit.read,admin.read...	View
	fastunit.test_full_fastunit	No	admin.root,fastunit.read,admin.read...	View

Role Information

✓	Role Name	Can Be Migrated	Permission	Inherited Role	Remarks
✓	fastunit.roletest6	Yes	{"resource": {"db": "fastu...	fastunit.readWrite,fastuni...	--
✓	fastunit.roletest3	Yes	{"resource": {"db": "fastu...	fastunit.roletest2	--
✓	fastunit.roletest2	Yes	{"resource": {"db": "fastu...	fastunit.roletest1	--

*Migrate Object [✓ All] Tables Databases

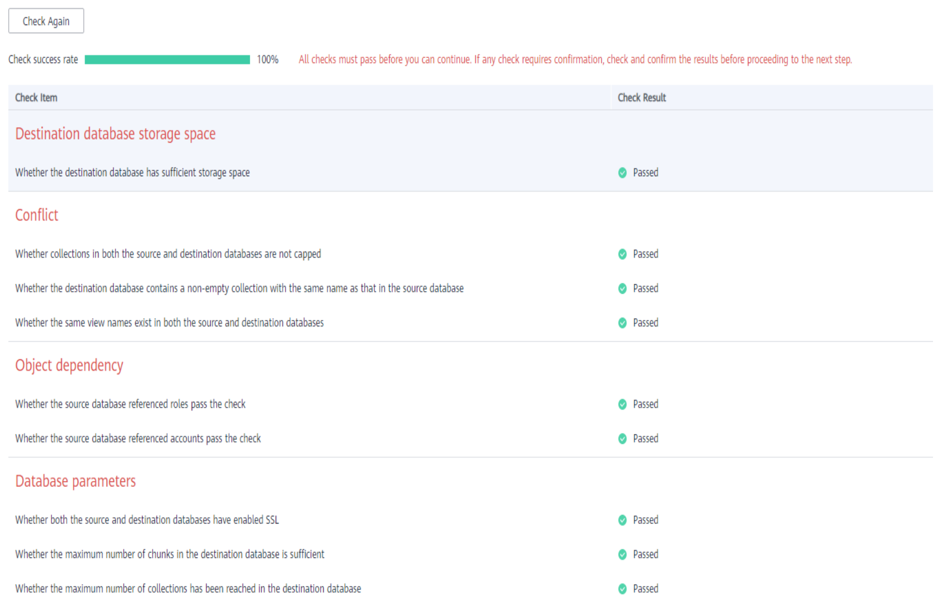
Table 6-7 Migration object

Paramete r	Description
Migrate Account	There are accounts that can be migrated completely and accounts that cannot be migrated. You can choose whether to migrate the accounts. Accounts that cannot be migrated or accounts that are not selected will not exist in the destination database. Ensure that your services will not be affected by these accounts. – **Yes** If you choose to migrate accounts, see **Migrating Accounts** in *Data Replication Service User Guide* to migrate database users and roles. – **No** During the migration, accounts and roles are not migrated.
Migrate Object	You can choose to migrate all objects, tables, or databases based on your service requirements. – **All**: All objects in the source database are migrated to the destination database. After the migration, the object names will remain the same as those in the source database and cannot be modified. – **Tables**: The selected table-level objects will be migrated. – **Databases**: The selected database-level objects will be migrated. If the source database is changed, click ↻ in the upper right corner before selecting migration objects to ensure that the objects to be selected are from the changed source database. **NOTE** – If you choose not to migrate all of the databases, the migration may fail because the objects, such as stored procedures and views, in the database to be migrated may have dependencies on other objects that are not migrated. To ensure a successful migration, you are advised to migrate all of the databases. – When you select an object, the spaces before and after the object name are not displayed. If there are two or more consecutive spaces in the middle of the object name, only one space is displayed. – The search function can help you quickly select the required database objects.

6. On the **Check Task** page, check the migration task.
   - If any check fails, review the cause and rectify the fault. After the fault is rectified, click **Check Again**.

     For details about how to handle check failures, see **Checking Whether the Source Database Is Connected** in *Data Replication Service User Guide*.
   - If all check items are successful, click **Next**.

Figure 6-7 Task Check



Check Again

Check success rate 100% All checks must pass before you can continue. If any check requires confirmation, check and confirm the results before proceeding to the next step.

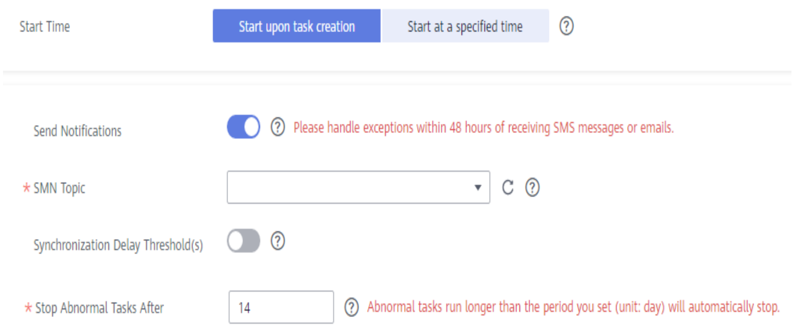
Check Item	Check Result
Destination database storage space	
Whether the destination database has sufficient storage space	✓ Passed
Conflict	
Whether collections in both the source and destination databases are not capped	✓ Passed
Whether the destination database contains a non-empty collection with the same name as that in the source database	✓ Passed
Whether the same view names exist in both the source and destination databases	✓ Passed
Object dependency	
Whether the source database referenced roles pass the check	✓ Passed
Whether the source database referenced accounts pass the check	✓ Passed
Database parameters	
Whether both the source and destination databases have enabled SSL	✓ Passed
Whether the maximum number of chunks in the destination database is sufficient	✓ Passed
Whether the maximum number of collections has been reached in the destination database	✓ Passed

NOTE

You can proceed to the next step only when all check items are successful. If any alarms are generated, view and confirm the alarm details first before proceeding to the next step.

7. On the displayed page, specify **Start Time**, **Send Notification**, **SMN Topic**, **Synchronization Delay Threshold**, and **Stop Abnormal Tasks After** and confirm that the configured information is correct and click **Submit** to submit the task.

Figure 6-8 Task startup settings



Start Time	[✓ Start upon task creation] Start at a specified time
Send Notifications	On — Please handle exceptions within 48 hours of receiving SMS messages or emails.
* SMN Topic	
Synchronization Delay Threshold(s)	Off
* Stop Abnormal Tasks After	14 — Abnormal tasks run longer than the period you set (unit: day) will automatically stop.

Table 6-8 Task startup settings

Parameter	Description
Start Time	Set **Start Time** to **Start upon task creation** or **Start at a specified time** based on site requirements. The **Start at a specified time** option is recommended. **NOTE** The migration task may affect the performance of the source and destination databases. You are advised to start the task in off-peak hours and reserve two to three days for data verification.
Send Notifications	SMN topic. This parameter is optional. If an exception occurs during migration, the system will send a notification to the specified recipients.
SMN Topic	This parameter is available only after you enable **Send Notification** and create a topic on the SMN console and add a subscriber. For details, see **Simple Message Notification User Guide**.
Synchronization Delay Threshold	During an incremental migration, a synchronization delay indicates a time difference (in seconds) of synchronization between the source and destination database. If the synchronization delay exceeds the threshold you specify, DRS will send alarms to the specified recipients. The value ranges from 0 to 3,600. To avoid repeated alarms caused by the fluctuation of delay, an alarm is sent only after the delay has exceeded the threshold for six minutes. **NOTE** – In the early stages of an incremental migration, there is more delay because more data is waiting to be synchronized. In this situation, no notifications will be sent. – Before setting the delay threshold, enable **Send Notification**. – If the delay threshold is set to 0, no notifications will be sent to the recipient.
Stop Abnormal Tasks After	Number of days after which an abnormal task is automatically stopped. The value must range from 14 to 100. The default value is **14**. **NOTE** Tasks in the abnormal state are still charged. If tasks remain in the abnormal state for a long time, they cannot be resumed. Abnormal tasks run longer than the period you set (unit: day) will automatically stop to avoid unnecessary fees.

8. After the task is submitted, go back to the **Online Migration Management** page to view the task status.

Step 2 Manage the migration task.

The migration task contains two phases: full migration and incremental migration. You can manage them in different phases.

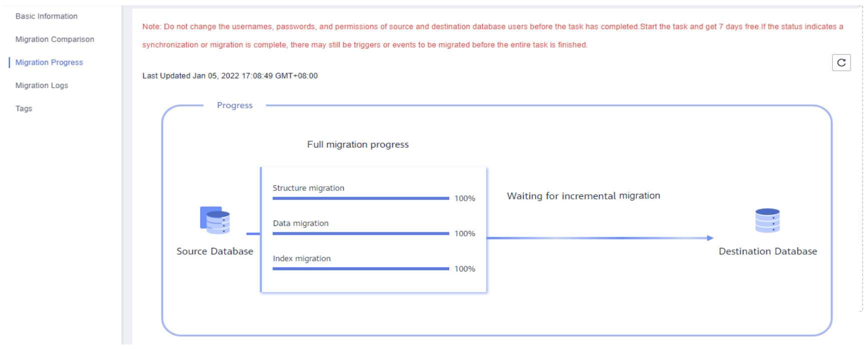
- Full migration

- Viewing the migration progress: Click the target full migration task, and on the **Migration Progress** tab, you can see the migration progress of the structure, data, indexes, and migration objects. When the progress reaches 100%, the migration is complete.
- Viewing migration details: In the migration details, you can view the migration progress of a specific object. If the number of objects is the same as that of migrated objects, the migration is complete. You can view the migration progress of each object in detail. Currently, this function is available only to whitelisted users. You can submit a service ticket to apply for this function.

● Incremental Migration Permission

- Viewing the synchronization delay: After the full migration is complete, an incremental migration starts. On the **Online Migration Management** page, click the target migration task. On the displayed page, click **Migration Progress** to view the synchronization delay of the incremental migration. If the synchronization delay is 0s, the destination database is being synchronized with the source database in real time. You can also view the data consistency on the **Migration Comparison** tab.

Figure 6-9 Viewing the synchronization delay

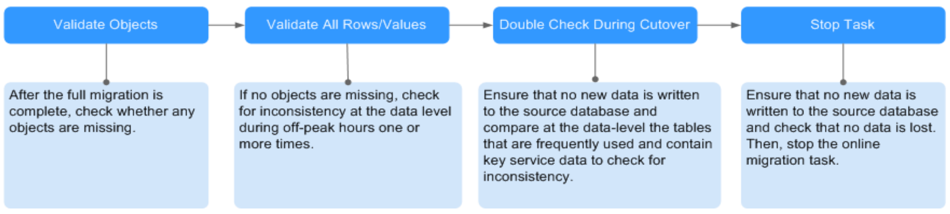


Note: Do not change the usernames, passwords, and permissions of source and destination database users before the task has completed.Start the task and get 7 days free.If the status indicates a synchronization or migration is complete, there may still be triggers or events to be migrated before the entire task is finished.

Last Updated Jan 05, 2022 17:08:49 GMT+08:00

- Viewing the migration results: On the **Online Migration Management** page, click the target migration task. On the displayed page, click **Migration Comparison** and perform a migration comparison in accordance with the comparison process, which should help you determine an appropriate time for migration to minimize service downtime.

Figure 6-10 Database comparison process


Validate Objects
After the full migration is complete, check whether any objects are missing.
Validate All Rows/Values
If no objects are missing, check for inconsistency at the data level during off-peak hours one or more times.
Double Check During Cutover
Ensure that no new data is written to the source database and compare at the data-level the tables that are frequently used and contain key service data to check for inconsistency.
Stop Task
Ensure that no new data is written to the source database and check that no data is lost. Then, stop the online migration task.

For details, see **Comparing Migration Items** in *Data Replication Service User Guide*.

Step 3 Cut over services.

You are advised to start the cutover process during off-peak hours. At least one complete data comparison is performed during off-peak hours. To obtain accurate

comparison results, start data comparison at a specified time point during off-peak hours. If it is needed, select **Start at a specified time** for **Comparison Time**. Due to slight time difference and continuous operations on data, inconsistent comparison results may be generated, reducing the reliability and validity of the results.

1. Interrupt services first. If the workload is not heavy, you may not need to interrupt the services.
2. Run the following statement on the source database and check whether any new sessions execute SQL statements within the next 1 to 5 minutes. If there are no new statements executed, the service has been stopped.
   ```
   db.currentOp()
   ```
   NOTE

   The process list queried by the preceding statement includes the connection of the DRS replication instance. If no additional session executes SQL statements, the service has been stopped.
3. On the **Migration Progress** page, view the synchronization delay. When the delay is displayed as 0s and remains stable for a period, then you can perform a data-level comparison between the source and destination databases. For details about the time required, refer to the results of the previous comparison.
   - If there is enough time, compare all objects.
   - If there is not enough time, use the data-level comparison to compare the tables that are frequently used and that contain key business data or inconsistent data.
4. Determine an appropriate time to cut the services over to the destination database. After services are restored and available, the migration is complete.

Step 4 Stop or delete the migration task.

1. Stopping the migration task. After databases and services are migrated to the destination database, to prevent operations on the source database from being synchronized to the destination database to overwrite data, you can stop the migration task. This operation only deletes the replication instance, and the migration task is still displayed in the task list. You can view or delete the task. DRS will not charge for this task after you stop it.
2. Delete the migration task. After the migration task is complete, you can delete it. After the migration task is deleted, it will no longer be displayed in the task list.

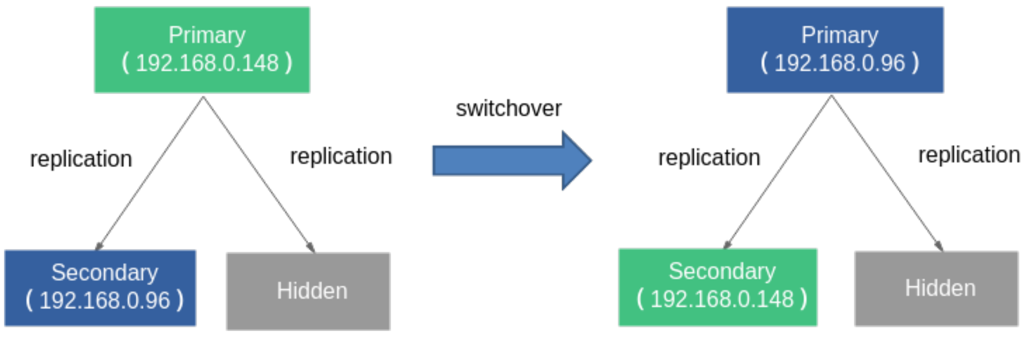
----End

7 How Do Replica Sets Achieve High Availability and Read/Write Splitting?

DDS replica set instances can store multiple duplicates to ensure data high availability and support the automatic switch of private IP addresses to ensure service high availability. To enhance the read and write performance of your client for connecting to the instance, you can use your client to read different data copies. You need to connect to replica set instances using HA connection addresses. You can also configure read/write splitting. Otherwise, the high availability and high read performance of replica set instances cannot be guaranteed.

The primary node of a replica set instance is not fixed. If the instance settings are changed, or the primary node fails, or primary and secondary nodes are switched, a new primary node will be elected and the previous one becomes a secondary node. The following figure shows the process of a switchover.

Figure 7-1 Primary/Secondary switchover



Connecting to a Replica Set Instance (HA)

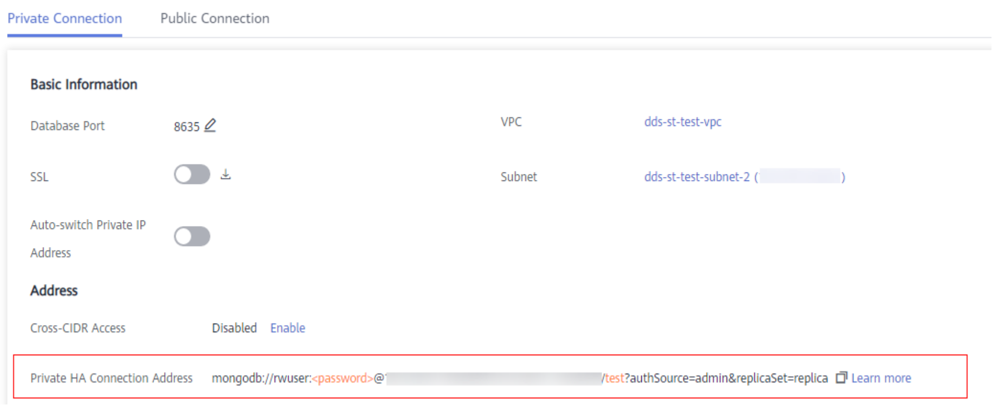
A DDS replica set consists of the primary, secondary, and hidden nodes. The hidden node is invisible to users. Read/Write splitting and HA can be realized only when you connect to the IP addresses and ports of the primary and secondary nodes of the replica set at the same time (in HA mode).

The following describes how to use URL and Java to connect to an instance in HA mode.

Method 1: Using a URL

On the **Instances** page, click the instance name. The **Basic Information** page is displayed. Choose **Connections**. Click the **Private Connection** tab and obtain the connection address of the current instance from the **Private HA Connection Address** field.

Figure 7-2 Obtaining the private HA connection address



Private Connection | Public Connection

Basic Information

Database Port 8635

VPC dds-st-test-vpc

SSL

Subnet dds-st-test-subnet-2 ([redacted])

Auto-switch Private IP Address

Address

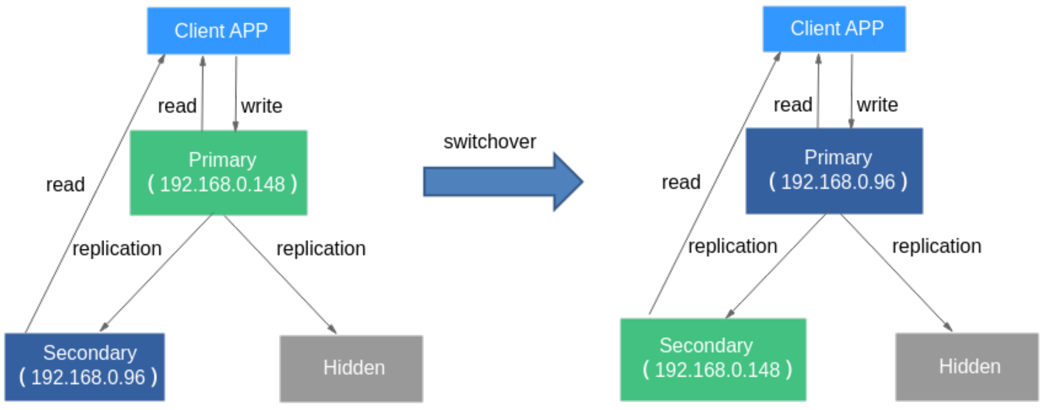
Cross-CIDR Access Disabled Enable

Private HA Connection Address mongodb://rwuser:<password>@[redacted]/test?authSource=admin&replicaSet=replica Learn more

Example: **mongodb://rwuser:****@***192.168.0.148:8635,192.168.0.96:8635***/test?authSource=admin&replicaSet=replica**

In the preceding URL, **192.168.0.148:8635** and **192.168.0.96:8635** are the IP addresses and ports of the primary and secondary nodes, respectively. If you use this address, the connection between your client and the instance can be ensured even when a primary/standby switchover occurs. In addition, using multiple IP addresses and port numbers can enhance the read and write performance of the entire database.

Figure 7-3 Data read and write process



Method 2: Using a Java Driver

Sample code:

```
MongoClientURI connectionString = new MongoClientURI("mongodb://
rwuser:****@192.168.0.148:8635,192.168.0.96:8635/test?authSource=admin&replicaSet=replica");
MongoClient client = new MongoClient(connectionString);
```

```
MongoDatabase database = client.getDatabase("test");
MongoCollection<Document> collection = database.getCollection("mycoll");
```

Table 7-1 Parameter description

Parameter	Description
rwuser:****	Username and password for starting authentication
192.168.0.148:8635, 192.168.0.96:8635	IP addresses and ports of the primary and secondary nodes in a replica set instance
test	Name of the database to be connected
authSource=admin	Database username for authentication
replicaSet=replica	Name of the replica set instance type

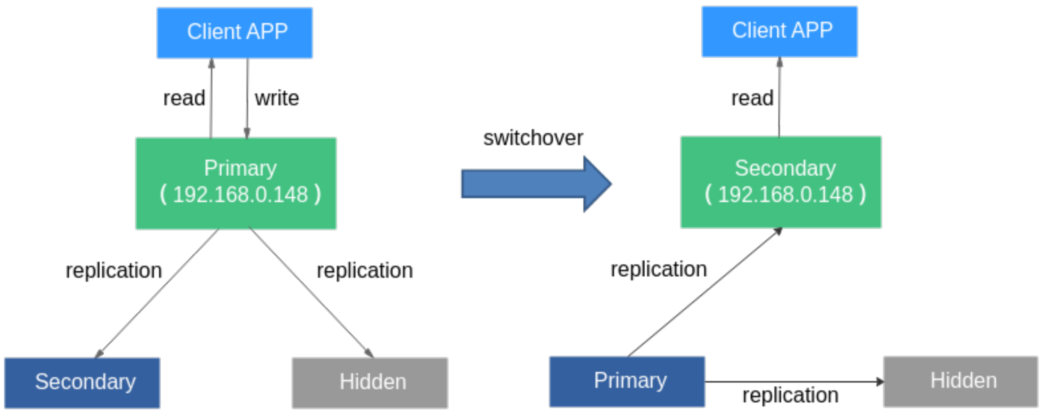
(Not Recommended) Connecting to a Replica Set Instance

Using the Connection Address

mongodb://rwuser:**@***192.168.0.148:8635***/test?
authSource=admin&replicaSet=replica**

In the preceding URL, **192.168.0.148:8635** is the IP address and port number of the current primary node. If a switchover occurs or the primary node is changed, the client fails to connect to the replica set instance because the IP address and port of the newly elected primary node is unknown. As a result, the database service becomes unavailable. In addition, read and write operations can only be performed on a fixed primary node, so the read and write performance cannot be improved by adding nodes.

Figure 7-4 Data read and write process



Read/Write Splitting

The following HA connection address is used as an example to describe how to connect to a DDS replica set instance:

```
mongodb://rwuser:<password>@192.168.xx.xx:8635,192.168.xx.xx:8635/test?
authSource=admin&replicaSet=replica&readPreference=secondaryPreferred
```

The database account is **rwuser**, and the database is **admin**.

NOTE

After the DB instance is connected, read requests are preferentially sent to the secondary node to implement read/write splitting. If the relationship between the primary and secondary nodes changes, write operations are automatically switched to the new primary node to ensure high availability of DDS.

8 Sharding

You can shard a large-size collection for a sharded cluster instance. Sharding distributes data across different machines to make full use of the storage space and compute capability of each shard.

Number of Shards

The following is an example using database **mytable**, collection **mycoll**, and the field **name** as the shard key.

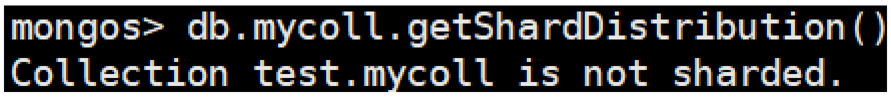
Step 1 Log in to a sharded cluster instance using Mongo Shell.

Step 2 Check whether a collection has been sharded.

```
use <database>
db.<collection>.getShardDistribution()
```

Example:

```
use mytable
db.mycoll.getShardDistribution()
```



```
mongos> db.mycoll.getShardDistribution()
Collection test.mycoll is not sharded.
```

Step 3 Enable sharding for the databases that belong to the cluster instance.

- Method 1

  ```
  sh.enableSharding("<database>")
  ```

  Example:

  ```
  sh.enableSharding("mytable")
  ```

- Method 2

  ```
  use admin
  db.runCommand({enablesharding:"<database>"})
  ```

Step 4 Shard a collection.

- Method 1

  ```
  sh.shardCollection("<database>.<collection>",{"<keyname>":<value> })
  ```

  Example:

  ```
  sh.shardCollection("mytable.mycoll",{"name":"hashed"},false,{numInitialChunks:5})
  ```

- Method 2

```
use admin
db.runCommand({shardcollection:"<database>.<collection>",key:{"keyname":<value> }})
```

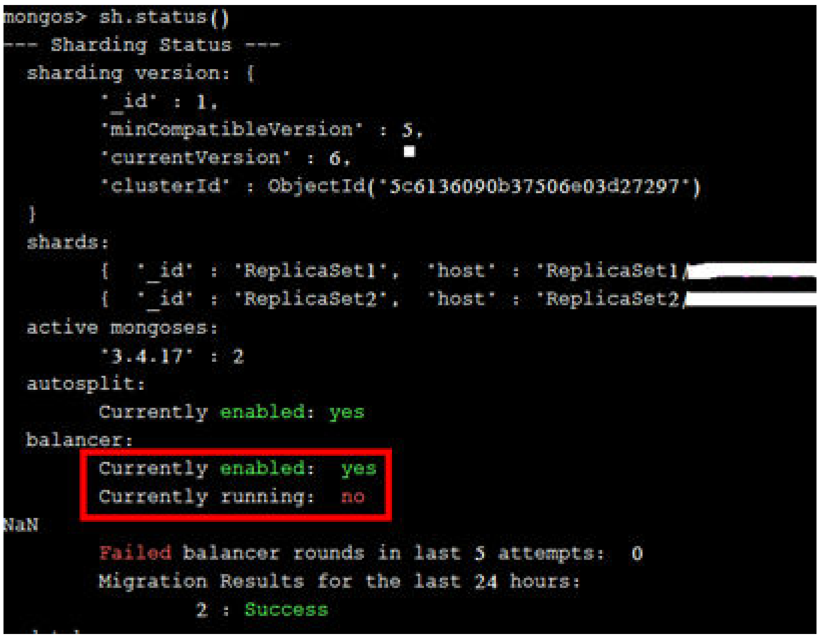
Table 8-1 Parameter description

Parameter	Description
<database>	Database name
<collection>	Collection name.
<keyname>	Shard key. Cluster instances are sharded based on the value of this parameter. Select a proper shard key for the collection based on your service requirements. For details, see **Selecting a Shard Key**.
<value>	The sort order based on the range of the shard key. ● 1: Ascending indexes ● -1: Descending indexes ● hashed: indicates that hash sharding is used. Hashed sharding provides more even data distribution across the sharded cluster. For details, see **sh.shardCollection()**.
numInitialChunks	Optional. The minimum number of shards initially created is specified when an empty collection is sharded using a hashed shard key.

Step 5 Check the data storage status of the database on each shard.

```
sh.status()
```

Example:



```
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("5c6136090b37506e03d27297")
  }
  shards:
        {  "_id" : "ReplicaSet1",  "host" : "ReplicaSet1/[redacted]
        {  "_id" : "ReplicaSet2",  "host" : "ReplicaSet2/[redacted]
  active mongoses:
        "3.4.17" : 2
  autosplit:
        Currently enabled: yes
  balancer:
        Currently enabled:  yes
        Currently running:  no
NaN
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                2 : Success
```

----End

Selecting a Shard Key

- **Background**

  Each sharded cluster contains collections as its basic unit. Data in the collection is partitioned by the shard key. Shard key is a field in the collection. It distributes data evenly across shards. If you do not select a proper shard key, the cluster performance may deteriorate, and the sharding statement execution process may be blocked.

  Once the shard key is determined it cannot be changed. If no shard key is suitable for sharding, you need to use a sharding policy and migrate data to a new collection for sharding.

- **Characteristics of proper shard keys**
  - All inserts, updates, and deletes are evenly distributed to all shards in a cluster.
  - The distribution of keys is sufficient.
  - Rare scatter-gather queries.

  If the selected shard key does not have all the preceding features, the read and write scalability of the cluster is affected. For example, If the workload of the find() operation is unevenly distributed in the shards, hot shards will be generated. Similarly, if your write load (inserts, updates, and deletes) is not uniformly distributed across your shards, then you could end up with a hot shard. Therefore, you need to adjust the shard keys based on service requirements, such as read/write status, frequently queried data, and written data.

  After existing data is sharded, if the **filter** filed of the update request does not contain shard keys and **upsert:true** or **multi:false**, the update request will report an error and return message "An upsert on a sharded collection must contain the shard key and have the simple collation.".

- **Judgment criteria**

You can use the dimensions provided in **Table 8-2** to determine whether the selected shard keys meet your service requirements:

Table 8-2 Reasonable shard keys

Identification Criteria	Description
Cardinality	Cardinality refers to the capability of dividing chunks. For example, if you need to record the student information of a school and use the age as a shard key, data of students of the same age will be stored in only one data segment, which may affect the performance and manageability of your clusters. A much better shard key would be the student number because it is unique. If the student number is used as a shard key, the relatively large cardinality can ensure the even distribution of data.
Write distribution	If a large number of write operations are performed in the same period of time, you want your write load to be evenly distributed over the shards in the cluster. If the data distribution policy is ranged sharding, a monotonically increasing shard key will guarantee that all inserts go into a single shard.
Read distribution	Similarly, if a large number of read operations are performed in the same period, you want your read load to be evenly distributed over the shards in a cluster to fully utilize the computing performance of each shard.
Targeted read	The dds mongos query router can perform either a targeted query (query only one shard) or a scatter/gather query (query all of the shards). The only way for the dds mongos to be able to target a single shard is to have the shard key present in the query. Therefore, you need to pick a shard key that will be available for use in the common queries while the application is running. If you pick a synthetic shard key, and your application cannot use it during typical queries, all of your queries will become scatter/gather, thus limiting your ability to scale read load.

Choosing a Distribution Policy

A sharded cluster can store a collection's data on multiple shards. You can distribute data based on the shard keys of documents in the collection.

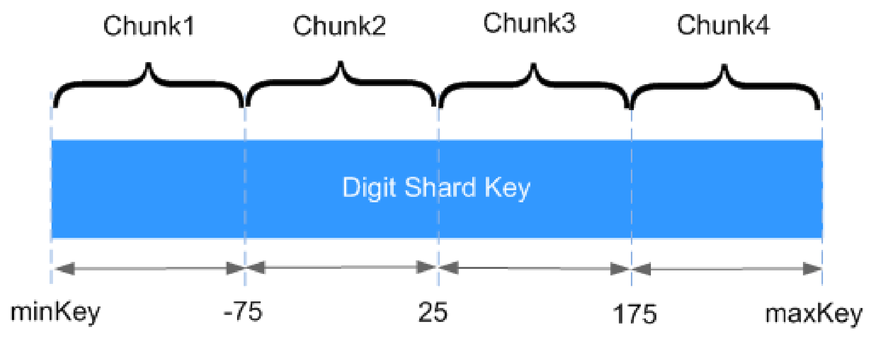
There are two data distribution policies: ranged sharding and hashed sharding. For details, see **Step 4**.

The following describes the advantages and disadvantages of the two methods.

- **Ranged sharding**

Ranged-based sharding involves dividing data into contiguous ranges determined by the shard key values. If you assume that a shard key is a line stretched out from positive infinity and negative infinity, each value of the shard key is the mark on the line. You can also assume small and separate segments of a line and that each chunk contains data of a shard key within a certain range.

Figure 8-1 Distribution of data



As shown in the preceding figure, field **x** indicates the shard key of ranged sharding. The value range is [*minKey*,*maxKey*] and the value is an integer. The value range can be divided into multiple chunks, and each chunk (usually 64 MB) contains a small segment of data. For example, chunk 1 contains all documents in range [minKey, -75] and all data of each chunk is stored on the same shard. That means each shard containing multiple chunks. In addition, the data of each shard is stored on the config server and is evenly distributed by dds mongos based on the workload of each shard.

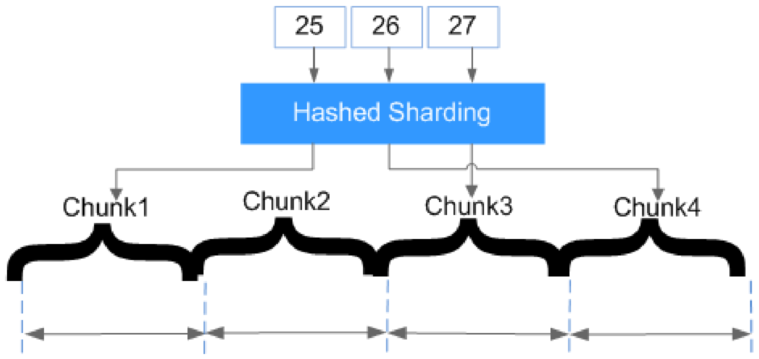
Ranged sharding can easily meet the requirements of query in a certain range. For example, if you need to query documents whose shard key is in range [-60,20], dds mongos only needs to forward the request to chunk 2.

However, if shard keys are in ascending or descending order, newly inserted documents are likely to be distributed to the same chunk, affecting the expansion of write capability. For example, if **_id** is used as a shard key, the high bits of **_id** automatically generated in the cluster are ascending.

- **Hashed sharding**

  Hashed sharding computes the hash value (64-bit integer) of a single field as the index value; this value is used as your shard key to partition data across your shared cluster. Hashed sharding provides more even data distribution across the sharded cluster because documents with similar shard keys may not be stored in the same chunk.

Figure 8-2 Distribution of data



Hashed sharding randomly distributes documents to each chunk, which fully expands the write capability and makes up for the deficiency of ranged sharding. However, queries in a certain range need to be distributed to all backend shards to obtain documents that meet conditions, resulting in low query efficiency.

9 How Do I Improve DDS Performance by Optimizing SQL Statements?

DDS is inherently a NoSQL database with high performance and strong extensibility. Similar to relational databases, such as RDS for MySQL, RDS for SQL Server, and Oracle, DDS instance performance may also be affected by database design, statement optimization, and index creation.

The following provides suggestions for improving DDS performance in different dimensions:

Creating Databases and Collections

1. Use short field names to save storage space. Different from an RDS database, each DDS document has its field names stored in the collection. Short name is recommended.
2. Limit the number of documents in a collection to avoid the impact on the query performance. Archive documents periodically if necessary.
3. Each document has a default **_id**. Do not change the value of this parameter.
4. Capped collections have a faster insertion speed than other collections and can automatically delete old data. You can create capped collections to improve performance based on your service requirements.

For details, see **Usage Suggestions** in the *Document Database Service Developer Guide*.

Query

Indexes

1. Create proper number of indexes for frequently queried fields based on service requirements. Indexes occupy some storage space, and the insert and indexing operations consume resources. It is recommended that the number of indexes in each collection should not exceed 5.

   If data query is slow due to lack of indexes, create proper indexes for frequently queried fields.
2. For a query that contains multiple shard keys, create a compound index that contains these keys. The order of shard keys in a compound index is

important. A compound index support queries that use the leftmost prefix of the index, and the query is only relevant to the creation sequence of indexes.

3. TTL indexes can be used to automatically filter out and delete expired documents. The index for creating TTL must be of type date. TTL indexes are single-field indexes.
4. You can create field indexes in a collection. However, if a large number of documents in the collection do not contain key values, you are advised to create sparse indexes.
5. When you create text indexes, the field is specified as **text** instead of **1** or **-1**. Each collection has only one text index, but it can index multiple fields.

Command usage

1. The findOne method returns the first document that satisfies the specified query criteria from the collection according to the natural order. To return multiple documents, use this method.
2. If the query does not require the return of the entire document or is only used to determine whether the key value exists, you can use **$project** to limit the returned field, reducing the network traffic and the memory usage of the client.
3. In addition to prefix queries, regular expression queries take longer to execute than using selectors, and indexes are not recommended.
4. Some operators that contain **$** in the query may deteriorate the system performance. The following types of operators are not recommended in services. $or, $nin, $not, $ne, and $exists.

NOTE

- $or: The times of queries depend on the number of conditions. It is used to query all the documents that meet the query conditions in the collection. You are advised to use $in instead.
- $nin: Matches most of indexes, and the full table scan is performed.
- $not: The query optimizer may fail to match a specific index, and the full table scan is performed.
- $ne: Selects the documents where the value of the field is not equal to the specified value. The entire document is scanned.
- $exists: matches each document that contains the field.

For more information, see **official MongoDB documents**.

Precautions

1. Indexes cannot be used in operators $where and $exists.
2. If the query results need to be sorted, control the number of result sets.
3. If multiple field indexes are involved, place the field used for exact match before the index.
4. If the key value sequence in the search criteria is different from that in the compound index, DDS automatically changes the query sequence to the same as index sequence.
   - Modification operation

     Modifying a document by using operators can improve performance. This method does not need to obtain and modify document data back and forth on the server, and takes less time to serialize and transfer data.

- Batch insert

  Batch insert can reduce the number of times data is submitted to the server and improve the performance. The BSON size of the data submitted in batches cannot exceed 48 MB.

- Aggregated operation

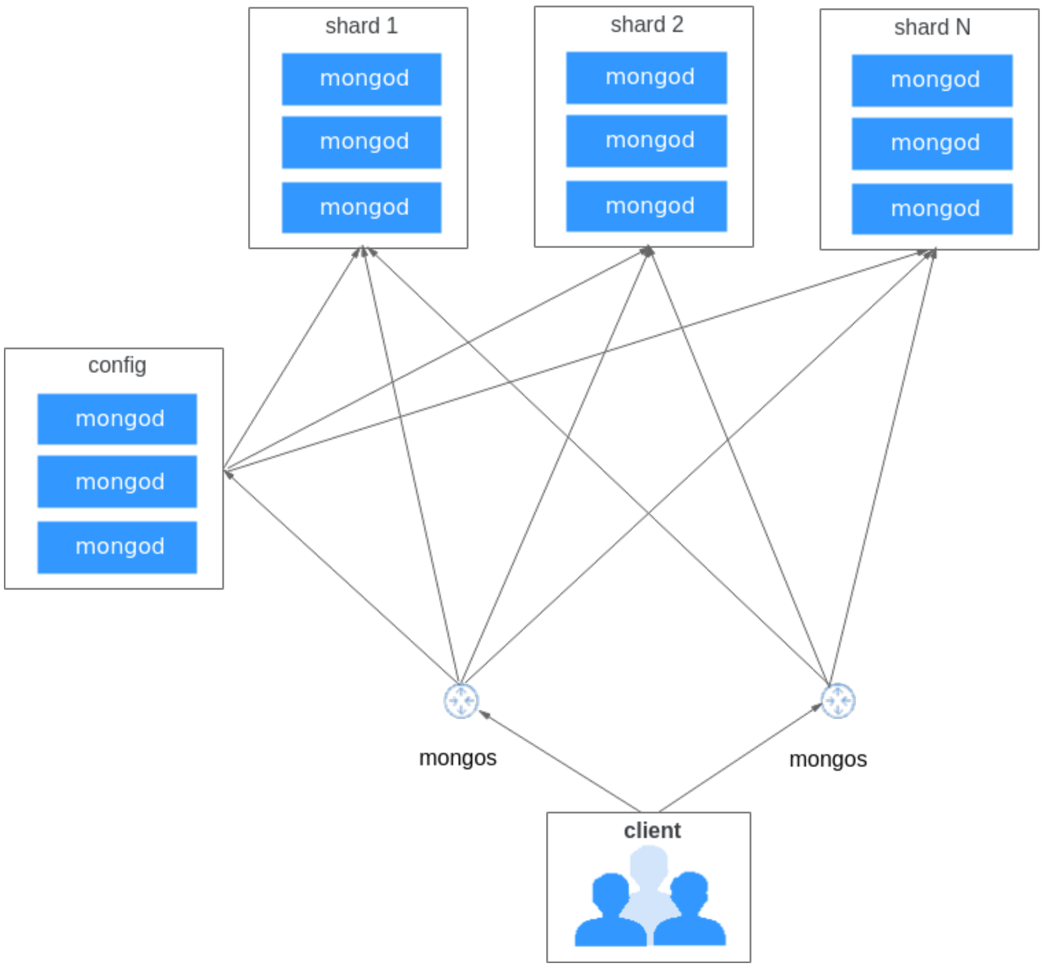
  During aggregation, $match must be placed before $group to reduce the number of documents to be processed by the $group operator.

10 How Do I Prevent the dds mongos Cache Problem?

Background

DDS is a document-oriented database service based on distributed file storage, famed for its scalability, high performance, open source, and free mode.

Figure 10-1 DDS cluster architecture



A cluster instance consists of the following three parts:

- dds mongos is deployed on a single node. It provides APIs to allow access from external users and shields the internal complexity of the distributed database. A DDS cluster can contain 2 to 12 dds mongos nodes. You can add them as required.
- Config server is deployed as a replica set. It stores metadata for a sharded cluster. The metadata include information about routes and shards. A cluster contains only one config server.
- Shard server is deployed as a replica set. It stores user data on shards. You can add shard servers in a cluster as required.

Sharding

Sharding is a method for distributing data evenly across multiple shard servers based on a specified shard key. The collection that has a shard key is called sharded collection. If the collection is not sharded, data is stored on only one shard server. DDS cluster mode allows the coexistence of sharded collection and non-sharded collection.

You can run the **sh.shardCollection** command to convert a non-sharded collection into a sharded collection. Before sharding, ensure that the sharding function is enabled on the database where the collections to be sharded are located. You can run the **sh.enableSharding** command to enable the sharding function.

Caching Metadata with dds mongos

User data is stored in the shard server and metadata is stored in the config server. The route information belongs to metadata and is also stored in the config server. When a user needs to access data through dds mongos, dds mongos sends the user's requests to the corresponding shard server according to the route information stored on the config server.

This means that every time the user accesses the data, dds mongos needs to connect to the config server for the route information, which may affect the system performance. Therefore, a cache mechanism is developed for the dds mongos to cache the route information of the config server. In this scenario, not only the config server stores the route information, but also the dds mongos caches the route information.

If no operation is performed on dds mongos, mongos does not cache any route information. In addition, the route information cached on dds mongos may not be the latest because the information is only updated in the following scenarios:

- If the dds mongos is started, it will obtain the latest route information from the config server and caches them locally.
- If the dds mongos processes the data request for the first time, it will obtain the route information from the config server. After that, the information is cached and can be used directly at the time when it is required.
- Updating route information by running commands on dds mongos.

NOTE

Only the metadata related to the requested data is updated.

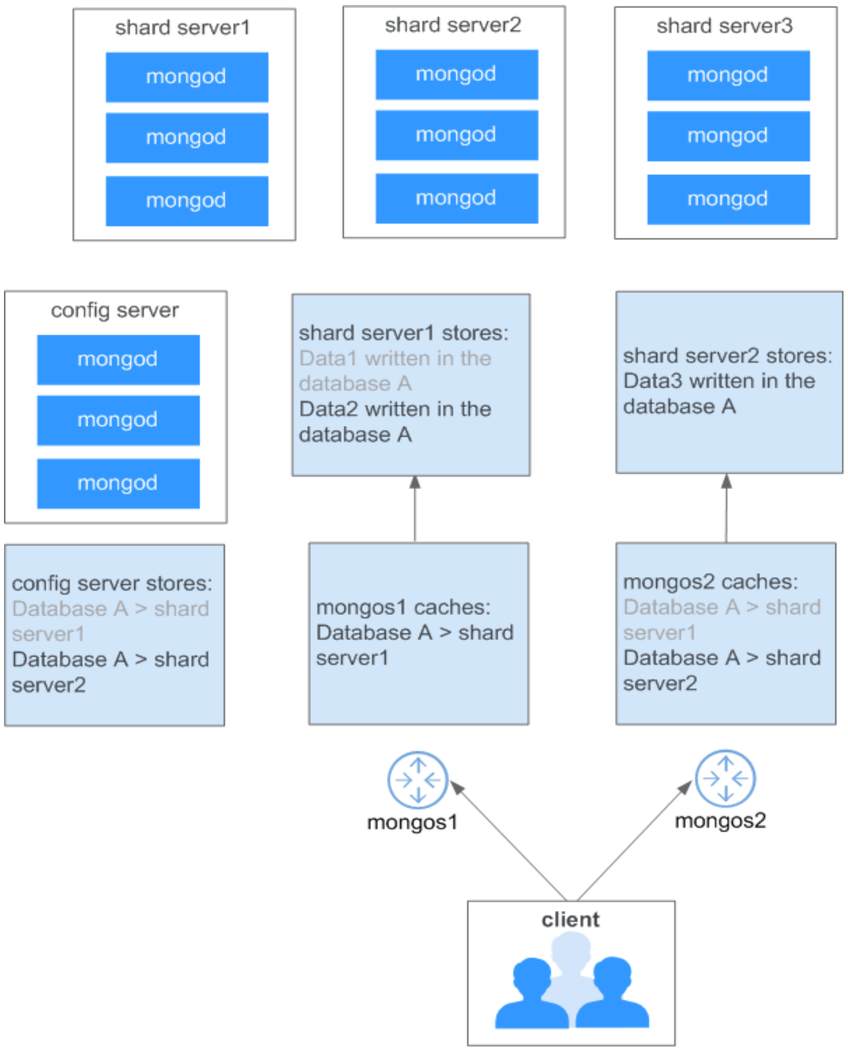
The data to be updated is in the unit of DB.

Scenarios

In the scenario where data is not sharded and multiple dds mongos nodes exist in a sharded cluster, if data is accessed through different dds mongos nodes, the cached route information on each dds mongos may become different. The following shows an example scenario:

1. Create database A with sharding disabled through mongos1. After data1 is written, data1 is allocated to shard server1 for storage. Then, mongos2 is used to query data. Both mongos1 and mongos2 have cached the route information of database A.
2. If database A is deleted through mongos2, the information about database A in the config server and shard server1 is deleted. As a result, mongos1 cannot identify data1 because database A has been deleted.
3. When data2 is written to database A through mongos1, data2 will be stored on shard server1 based on the cached route information but actually database A has been deleted. Then, when data3 is written into database A through mongos2, new information about database A will be generated again on the config server and shard server2 because mongos2 has identified that database A has been deleted.
4. In this case, the route information cached in the mongos1 and mongos2 is inconsistent. mongos1 and mongos2 are associated with different shard servers, and data is not shared between them. As a result, data inconsistency occurs.

Figure 10-2 mongos cache defect scenario



The client queries data through different mongos:

- mongos1: Data2 can be queried, but data3 cannot be queried.
- mongos2: Data3 can be queried, but data2 cannot be queried.

Workaround Suggestion

MongoDB official suggestions: After deleting databases or collections, run **db.adminCommand("flushRouterConfig")** on all mongos nodes to update the route information.

Reference link:

- **https://docs.mongodb.com/manual/reference/method/db.dropDatabase/index.html#replica-set-and-sharded-clusters**
- **https://jira.mongodb.org/browse/SERVER-17397**

Workaround Suggestion

- For the cluster mode, you are advised to enable the sharding function and then shard the collections in the cluster.

- If the database with sharding disabled is deleted, do not create a database or collection with the same name as the deleted database or collection.

  If you need to create a database or collection with the same name as the deleted database or collection, log in to all the mongos nodes to update the route information before creating the database and collection.

11 How Do I Solve the High CPU Usage Issue?

If the CPU usage is high or close to 100% when you use DDS, data read and write will slow down, affecting your services.

The following describes how to analyze current slow queries. After the analysis and optimization, queries will be processed better and indexes will be used more efficiently.

Analyzing Current Queries

1. Connect to an instance using Mongo Shell.

   To access an instance from the Internet

   For details, see

   – **Connecting to a Cluster Instance over a Public Network**
   – **Connecting to a Replica Set Instance over a Public Network**
   – **Connecting to a Single Node Instance over a Public Network**

   To access an instance that is not publicly accessible

   For details, see

   – **Connecting to a Cluster Instance over a Private Network**
   – **Connecting to a Replica Set Instance over a Private Network**
   – **Connecting to a Single Node Instance over a Private Network**

2. Run the following command to view the operations being performed on the database:

   db.currentOp()

   Command output:

```
{
        "raw" : {
                "shard0001" : {
                        "inprog" : [
                                {
                                        "desc" : "StatisticsCollector",
                                        "threadId" : "140323686905600",
                                        "active" : true,
                                        "opid" : 9037713,
                                        "op" : "none",
                                        "ns" : "",
                                        "query" : {

                                        },
                                        "numYields" : 0,
                                        "locks" : {

                                        },
                                        "waitingForLock" : false,
                                        "lockStats" : {

                                        }
                                },
                                {
                                        "desc" : "conn2607",
                                        "threadId" : "140323415066368",
                                        "connectionId" : 2607,
                                        "client" : "172.16.36.87:37804",
                                        "appName" : "MongoDB Shell",
                                        "active" : true,
                                        "opid" : 9039588,
                                        "secs_running" : 0,
                                        "microsecs_running" : NumberLong(63),
                                        "op" : "command",
                                        "ns" : "admin.",
                                        "query" : {
                                                "currentOp" : 1
                                  },
                                        "numYields" : 0,
                                        "locks" : {

                                        },
                                        "waitingForLock" : false,
                                        "lockStats" : {

                                        }
                                }
                        ],
                        "ok" : 1
                },
              ...
}
```

NOTE

- **client**: IP address of the client that sends the request
- **opid**: unique operation ID
- **secs_running**: elapsed time for execution, in seconds. If the returned value of this field is too large, check whether the request is reasonable.
- **microsecs_running**: elapsed time for execution, in microseconds. If the returned value of this field is too large, check whether there is something wrong with the request.
- **op**: operation type. The operations can be query, insert, update, delete, or command.
- **ns**: target collection
- For details, see the **db.currentOp()** command in **official document**.

3. Based on the command output, check whether there are requests that take a long time to process.

   If the CPU usage is low while services are being processed but then becomes high during just certain operations, analyze the requests that take a long time to execute.

   If an abnormal query is found, find the **opid** corresponding to the operation and run **db.killOp(***opid***)** to kill it.

Analyzing Slow Queries

Slow query profiling is enabled for DDS by default. The system automatically records any queries whose execution takes longer than 100 ms to the **system.profile** collection in the corresponding database. You can:

1. Connect to an instance using Mongo Shell.

   To access an instance from the Internet

   For details, see

   – **Connecting to a Cluster Instance over a Public Network**
   – **Connecting to a Replica Set Instance over a Public Network**
   – **Connecting to a Single Node Instance over a Public Network**

   To access an instance that is not publicly accessible

   For details, see

   – **Connecting to a Cluster Instance over a Private Network**
   – **Connecting to a Replica Set Instance over a Private Network**
   – **Connecting to a Single Node Instance over a Private Network**

2. Select a specific database (using the **test** database as an example):

   use test

3. Check whether slow SQL queries have been collected in **system.profile**.

   show collections;

   – If the command output includes **system.profile**, slow SQL queries have been generated. Go to the next step.
```
mongos> show collections
system.profile
test
```
   – If the command output does not contain **system.profile**, no slow SQL queries have been generated, and slow query analysis is not required.
```
mongos> show collections
test
```
4. Check the slow query logs in the database.

   db.system.profile.find().pretty()

5. Analyze slow query logs to find the cause of the high CPU usage.

   The following is an example of a slow query log. The log shows a request that scanned the entire table, including 1,561,632 documents and without using a search index.
```
{
        "op" : "query",
        "ns" : "taiyiDatabase.taiyiTables$10002e",
        "query" : {
                "find" : "taiyiTables",
                "filter" : {
                        "filed19" : NumberLong("852605039766")
                },
                "shardVersion" : [
                        Timestamp(1, 1048673),
                        ObjectId("5da43185267ad9c374a72fd5")
                ],
                "chunkId" : "10002e"
        },
        "keysExamined" : 0,
        "docsExamined" : 1561632,
        "cursorExhausted" : true,
        "numYield" : 12335,
        "locks" : {
                "Global" : {
                        "acquireCount" : {
                                "r" : NumberLong(24672)
                        }
                },
                "Database" : {
                        "acquireCount" : {
                                "r" : NumberLong(12336)
                        }
                },
                "Collection" : {
                        "acquireCount" : {
                                "r" : NumberLong(12336)
                        }
                }
        },
        "nreturned" : 0,
        "responseLength" : 157,
        "protocol" : "op_command",
        "millis" : 44480,
        "planSummary" : "COLLSCAN",
        "execStats" : {
              "stage" :
"SHARDING_FILTER",
                [3/1955]
                "nReturned" : 0,
                "executionTimeMillisEstimate" : 43701,
                "works" : 1561634,
                "advanced" : 0,
                "needTime" : 1561633,
                "needYield" : 0,
                "saveState" : 12335,
                "restoreState" : 12335,
                "isEOF" : 1,
                "invalidates" : 0,
                "chunkSkips" : 0,
                "inputStage" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "filed19" : {
                                        "$eq" : NumberLong("852605039766")
                                }
                        },
                        "nReturned" : 0,
                        "executionTimeMillisEstimate" : 43590,
                        "works" : 1561634,
                        "advanced" : 0,
                        "needTime" : 1561633,
                        "needYield" : 0,
                        "saveState" : 12335,
                        "restoreState" : 12335,
                        "isEOF" : 1,
                        "invalidates" : 0,
                        "direction" : "forward",
                        "docsExamined" : 1561632
                }
        },
        "ts" : ISODate("2019-10-14T10:49:52.780Z"),
        "client" : "172.16.36.87",
        "appName" : "MongoDB Shell",
        "allUsers" : [
                {
                        "user" : "__system",
                        "db" : "local"
                }
        ],
    "user" : "__system@local"
}
```

The following stages can be causes for a slow query:

– **COLLSCAN** involves a full collection (full table) scan.

  When a request (such as query, update, and delete) requires a full table scan, a large amount of CPU resources are occupied. If you find **COLLSCAN** in the slow query log, CPU resources may be occupied.

  If such requests are frequent, create indexes for the fields to be queried.

– **docsExamined** involves a full collection (full table) scan.

  You can view the value of **docsExamined** to check the number of documents scanned. A larger value indicates a higher CPU usage.

– **IXSCAN** and **keysExamined** scan indexes.

  NOTE

  An excessive number of indexes can affect the write and update performance.

  If your application has more write operations, creating indexes may increase write latency.

  You can view the value of **keyExamined** to see how many indexes are scanned in a query. A larger value indicates a higher CPU usage.

  If the index is not appropriate or there are many matching results, the CPU usage may spike and the execution can slow down.

  Example: For the data of a collection, the number of values of the **a** field is small (only **1** and **2**), but the **b** field has more values.

```
{ a: 1, b: 1 }
{ a: 1, b: 2 }
{ a: 1, b: 3 }
......
{ a: 1, b: 100000}
{ a: 2, b: 1 }
{ a: 2, b: 2 }
{ a: 2, b: 3 }
......
{ a: 1, y: 100000}
```

  The following shows how to implement the {a: 1, b: 2} query.

```
db.createIndex({a: 1}): The query is not effective because the a field has too many
same values.
db.createIndex({a: 1, b: 1}): The query is not effective because the a field has too
many same values.
db.createIndex({b: 1}): The query is effective because the b field has a few same
values.
db.createIndex({b: 1, a: 1}): The query is effective because the b field has a few same
values.
```

  For the differences between {a: 1} and {b: 1, a: 1}, see the **official documents**.

– **SORT** and **hasSortStage** may involve sorting a large amount of data.

  If the value of **hasSortStage** in the **system.profile** collection is **true**, the query request involves sorting. If the sorting cannot be implemented through indexes, the query results are sorted, and sorting is a CPU intensive operation. In this scenario, you need to create indexes for fields that are frequently sorted.

If the **system.profile** collection contains **SORT**, you can use indexing to improve sorting speed.

Other operations, such as index creation and aggregation (combinations of traversal, query, update, and sorting), also apply to the preceding scenarios because they are also CPU intensive operations. For more information about profiling, see **official documents**.

Analysis Capability

After the analysis and optimization of the requests that are being executed and slow requests, all requests use proper indexes, and the CPU usage becomes stable. If the CPU usage remains high after the analysis and troubleshooting, the current instance may have reached the performance bottleneck and cannot meet service requirements. In this case, you can perform the following operations to solve the problem:

1. View monitoring information to analyze instance resource usage. For details, see **Viewing Monitoring Metrics**.
2. Change the DDS instance class or add shard nodes.

12 How Do I Troubleshoot High Memory Usage of DDS DB Instances?

During DDS usage, if your memory usage is too high or close to 100%, service requests will be responded slowly and OOM risks will increase, affecting stable services.

This section describes how to troubleshoot the high memory usage of DDS DB instances by checking the number of database connections, analyzing slow requests, and checking cursors. After the analysis and optimization, query performance will be improved, indexes will be used more efficiently for all requests, and cursors will be used in a standard manner. If the memory usage increases due to business growth, you are advised to **upgrade the specifications** in a timely manner.

Checking the Number of Connections

1. Check the **percentage of connections**. The total number of connections cannot exceed 80% of the maximum number of connections supported by the current instance. If there are too many connections, the memory and multi-thread context overhead increases, affecting the delay in request processing.
2. Configure a connection pool. It is recommended that the maximum number of connections in a connection pool be 200. For details, see **How Do I Query and Limit the Number of Connections?**

Analyzing Slow Query Logs

In addition to reducing the number of connections, pay attention to the memory overhead of a single request to avoid full table scan and memory sorting in query statements.

1. Query **slow query logs** generated for the current DB instance.
2. Analyze slow query logs to locate the cause of memory usage increase. The following is an example of a slow request log. The log shows that a full table scan is performed for the request, 1,561,632 documents are scanned, and no index is used for query.
```
{
        "op" : "query",
        "ns" : "taiyiDatabase.taiyiTables$10002e",
        "query" : {
                "find" : "taiyiTables",
                "filter" : {
                        "filed19" : NumberLong("852605039766")
                },
                "shardVersion" : [
                        Timestamp(1, 1048673),
                        ObjectId("5da43185267ad9c374a72fd5")
                ],
                "chunkId" : "10002e"
        },
        "keysExamined" : 0,
        "docsExamined" : 1561632,
        "cursorExhausted" : true,
        "numYield" : 12335,
        "locks" : {
                "Global" : {
                        "acquireCount" : {
                                "r" : NumberLong(24672)
                        }
                },
                "Database" : {
                        "acquireCount" : {
                                "r" : NumberLong(12336)
                        }
                },
                "Collection" : {
                        "acquireCount" : {
                                "r" : NumberLong(12336)
                        }
                }
        },
        "nreturned" : 0,
        "responseLength" : 157,
        "protocol" : "op_command",
        "millis" : 44480,
        "planSummary" : "COLLSCAN",
        "execStats" : {
              "stage" :
"SHARDING_FILTER",
                                  [3/1955]
                "nReturned" : 0,
                "executionTimeMillisEstimate" : 43701,
                "works" : 1561634,
                "advanced" : 0,
                "needTime" : 1561633,
                "needYield" : 0,
                "saveState" : 12335,
                "restoreState" : 12335,
                "isEOF" : 1,
                "invalidates" : 0,
                "chunkSkips" : 0,
                "inputStage" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "filed19" : {
                                        "$eq" : NumberLong("852605039766")
                                }
                        },
                        "nReturned" : 0,
                        "executionTimeMillisEstimate" : 43590,
                        "works" : 1561634,
                        "advanced" : 0,
                        "needTime" : 1561633,
                        "needYield" : 0,
                        "saveState" : 12335,
                        "restoreState" : 12335,
                        "isEOF" : 1,
                        "invalidates" : 0,
                        "direction" : "forward",
                        "docsExamined" : 1561632
                }
        },
        "ts" : ISODate("2019-10-14T10:49:52.780Z"),
        "client" : "172.16.36.87",
        "appName" : "MongoDB Shell",
        "allUsers" : [
                {
                        "user" : "__system",
                        "db" : "local"
                }
        ],
       "user" : "__system@local"
}
```

The following stages can be causes for a slow query:

– **COLLSCAN** involves a full collection (full table) scan.
    - When a request (such as query, update, and delete) requires a full table scan, a large amount of memory resources are occupied. If you find **COLLSCAN** in the slow query log, memory resources may be occupied.
    - If such requests are frequent, create indexes for the fields to be queried.

– **docsExamined** involves a full collection (full table) scan.
    - You can view the value of **docsExamined** to check the number of documents scanned. A larger value indicates a higher memory usage.

– **IXSCAN** and **keysExamined** scan indexes.

– **SORT** and **hasSortStage** may involve sorting a large amount of data.

If the value of the **hasSortStage** parameter is **true**, the query request involves sorting. If the sorting cannot be implemented through indexes, the query results are sorted, and sorting is a memory intensive operation. In this scenario, you need to create indexes for fields that are frequently sorted.

If there is **SORT**, you can use indexing to improve sorting speed.

NOTE

An excessive number of indexes can affect the write and update performance. You are advised to create indexes based on the ESR principle to improve query efficiency.

- "Equality" refers to an exact match on a single value. Place fields that require exact matches first in an index.
- "Sort" determines the order for results. A sort condition follows equality matches.
- "Range" filters scan fields. Place a range filter after a sort condition.

Checking Cursors

If cursors are used improperly, the memory usage may increase and the cursors may not be released for a long time. When a cursor is used on the client, release it in a timely manner (For details, see **official description**).

1. Check whether a cursor is set to **noTimeout**. By default, the database automatically releases a cursor 10 minutes later. The following is an example of cursor timeout code provided by the Java driver:
```
MongoCursor<Document> cursor = collection.find(query)
                    .maxTime(10, TimeUnit.MINUTES)
                    .iterator();
```
2. Check whether the client actively releases a cursor after the cursor is used. The following is an example of releasing a cursor in the Java driver:
```
cursor.close()
```
3. If cursors are set to **noTimeout** and they cannot be released on the client, **restart the instance node** with high memory usage to release these cursors. You are advised to optimize the service code to avoid setting **noTimeout** cursors and release them after using them.

13 What Can I Do If the Number of Connections of an Instance Reaches Its Maximum?

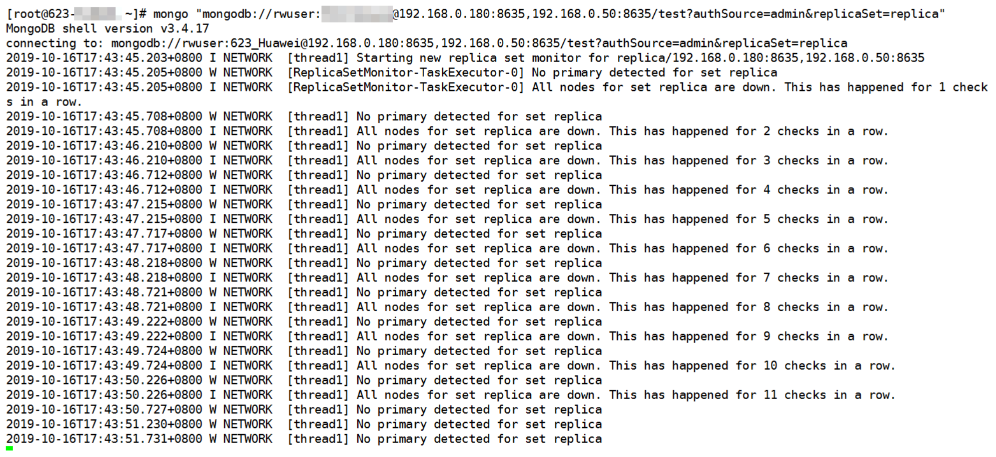
The number of connections indicates the number of applications that can be simultaneously connected to the database. The number of connections is irrelevant to the maximum number of users allowed by your applications or websites.

- For a cluster instance, the number of connections is the number of connections between the client and the mongos.
- For a replica set instance, the number of connections is the number of connections between the client and the primary and secondary nodes.

Symptom

When the number of connections to a DDS DB instance reaches the maximum, new connection requests cannot be responded. As a result, the connection to the DB instance fails.

- If the following information is displayed when you use Mongo Shell to connect to an instance, no more connections can be established.



```
[root@623-[redacted] ~]# mongo "mongodb://rwuser:[redacted]@192.168.0.180:8635,192.168.0.50:8635/test?authSource=admin&replicaSet=replica"
MongoDB shell version v3.4.17
connecting to: mongodb://rwuser:623_Huawei@192.168.0.180:8635,192.168.0.50:8635/test?authSource=admin&replicaSet=replica
2019-10-16T17:43:45.203+0800 I NETWORK  [thread1] Starting new replica set monitor for replica/192.168.0.180:8635,192.168.0.50:8635
2019-10-16T17:43:45.205+0800 W NETWORK  [ReplicaSetMonitor-TaskExecutor-0] No primary detected for set replica
2019-10-16T17:43:45.205+0800 I NETWORK  [ReplicaSetMonitor-TaskExecutor-0] All nodes for set replica are down. This has happened for 1 check
s in a row.
2019-10-16T17:43:45.708+0800 W NETWORK  [thread1] No primary detected for set replica
2019-10-16T17:43:45.708+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 2 checks in a row.
2019-10-16T17:43:46.210+0800 W NETWORK  [thread1] No primary detected for set replica
2019-10-16T17:43:46.210+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 3 checks in a row.
2019-10-16T17:43:46.712+0800 W NETWORK  [thread1] No primary detected for set replica
2019-10-16T17:43:46.712+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 4 checks in a row.
2019-10-16T17:43:47.215+0800 W NETWORK  [thread1] No primary detected for set replica
2019-10-16T17:43:47.215+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 5 checks in a row.
2019-10-16T17:43:47.717+0800 W NETWORK  [thread1] No primary detected for set replica
2019-10-16T17:43:47.717+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 6 checks in a row.
2019-10-16T17:43:48.218+0800 W NETWORK  [thread1] No primary detected for set replica
2019-10-16T17:43:48.218+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 7 checks in a row.
2019-10-16T17:43:48.721+0800 W NETWORK  [thread1] No primary detected for set replica
2019-10-16T17:43:48.721+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 8 checks in a row.
2019-10-16T17:43:49.222+0800 W NETWORK  [thread1] No primary detected for set replica
2019-10-16T17:43:49.222+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 9 checks in a row.
2019-10-16T17:43:49.724+0800 W NETWORK  [thread1] No primary detected for set replica
2019-10-16T17:43:49.724+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 10 checks in a row.
2019-10-16T17:43:50.226+0800 W NETWORK  [thread1] No primary detected for set replica
2019-10-16T17:43:50.226+0800 I NETWORK  [thread1] All nodes for set replica are down. This has happened for 11 checks in a row.
2019-10-16T17:43:50.727+0800 W NETWORK  [thread1] No primary detected for set replica
2019-10-16T17:43:51.230+0800 W NETWORK  [thread1] No primary detected for set replica
2019-10-16T17:43:51.731+0800 W NETWORK  [thread1] No primary detected for set replica
```

- If the following information is displayed when you use Python to connect to an instance, the number of connections reaches its maximum.

pymongo.errors.ServerSelectionTimeoutError: connection closed, connection closed

- You can also view the **instance monitoring metrics** to check whether the number of instance connections is used up.

Solution

Handle the problem based on its cause of burst traffic or long-term workloads.

- If the number of connections reaches the upper limit due to burst traffic, **restart the instance or node** to release the current connections. Also, check whether there is a large number of retry requests on the client. If the number of retry requests is used up, modify the client parameters to prevent the number of connections from being stacked. You need to modify the retry logic and increase the timeout retry duration.

  CAUTION

  Restarting an instance will restart the nodes of the instance in turn. Each node will be intermittently disconnected for about 30 seconds. If the number of collections exceeds 10,000, the intermittent disconnection duration will be prolonged. Before restarting an instance, arrange workloads and ensure that the application supports automatic reconnection.

- If the number of connections is used up due to long-term workloads, **increase the maximum number of connections** by changing the value of **net.maxIncomingConnections**. The change takes effect immediately. Ensure that a changed value is within 20% of the original value each time. After the value change, **observe the load changes**. If the load is too high after the number of connections is increased, the instance load has reached the bottleneck. In this case, **upgrade the instance specifications** in a timely manner.

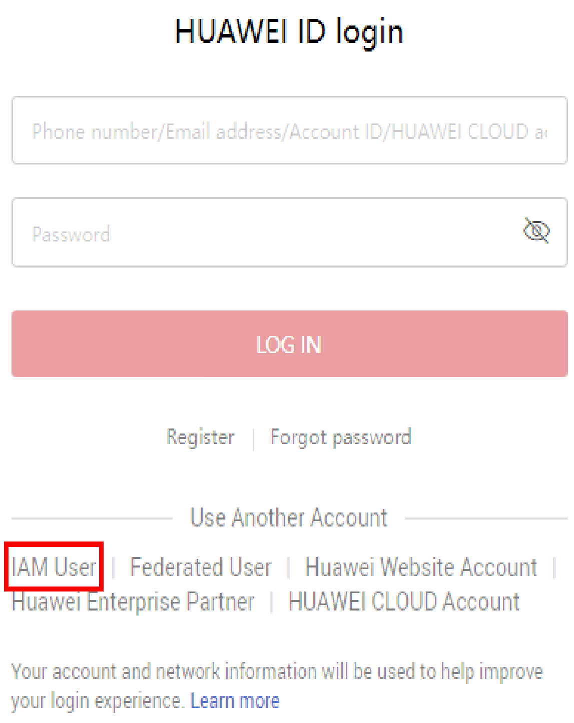
14 Creating a User and Granting the Read-Only Permission to the User

Step 1: Create a User Group and Grant Permissions

Users in the same user group have the same permissions. Users created in IAM inherit permissions from the groups to which they belong. Users created in IAM inherit permissions from the groups they belong to. To create a user group, perform the following steps:

Step 1 Log in to Huawei Cloud using your Huawei account. Select **HUAWEI ID login**.

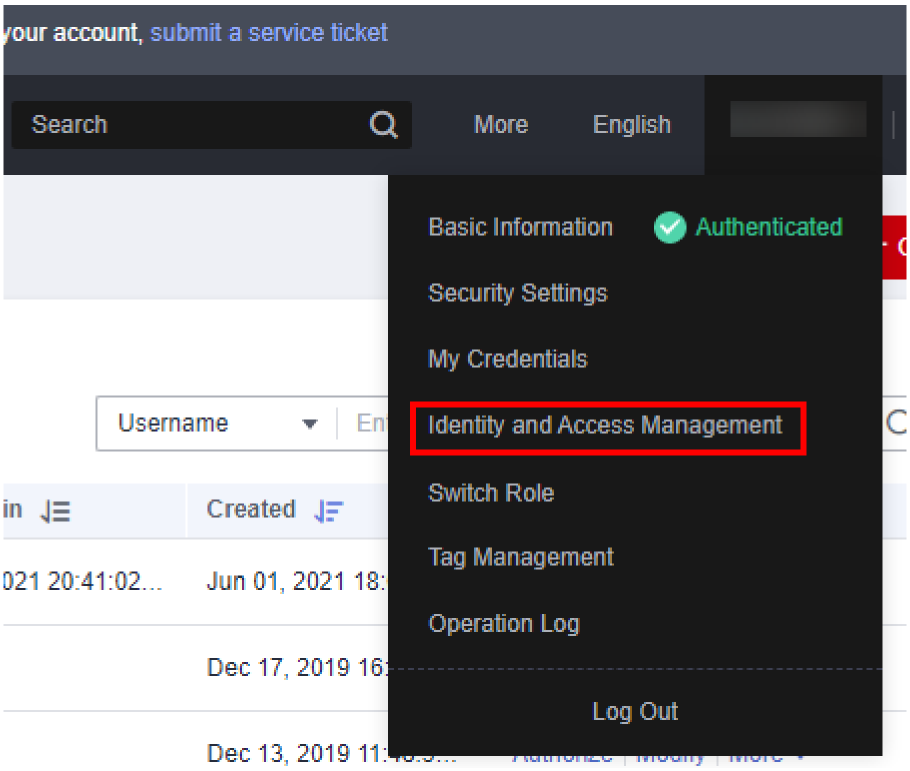
Figure 14-1 HUAWEI ID Login


HUAWEI ID login
Phone number/Email address/Account ID/HUAWEI CLOUD a
Password
LOG IN
Register | Forgot password
Use Another Account
IAM User | Federated User | Huawei Website Account |
Huawei Enterprise Partner | HUAWEI CLOUD Account

Your account and network information will be used to help improve your login experience. Learn more

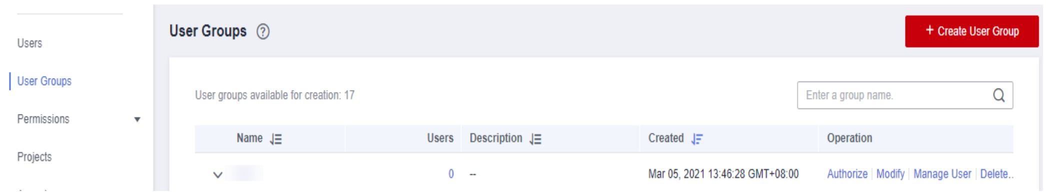
Step 2 On the management console, click the username in the upper right corner and then choose **Identity and Access Management**.

Figure 14-2 Choosing IAM



Step 3 On the IAM console, choose **User Groups** in the navigation pane. Then click **Create User Group**.

Figure 14-3 User group



User groups available for creation: 17

Name	Users	Description	Created	Operation
[redacted]	0	--	Mar 05, 2021 13:46:28 GMT+08:00	Authorize | Modify | Manage User | Delete..

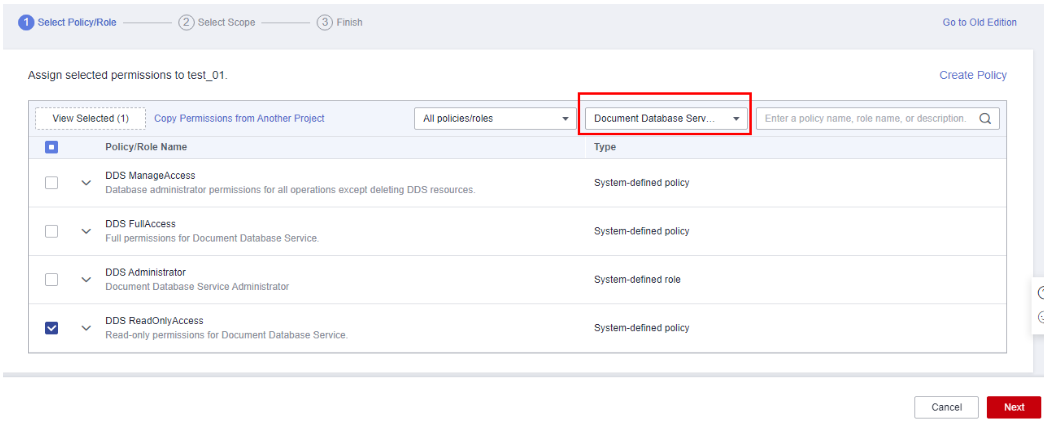
Step 4 Enter a user group name (for example, **test_01**), set the password, and click **OK**.

The user group is then displayed in the user group list.

Step 5 In the user group list, choose **Authorize** in the row that contains the **test_01** user group.

Step 6 Select **Document Database Service** from the drop-down list, select **DDS ReadOnlyAccess**, and click **Next**.

Figure 14-4 Authorization



1 Select Policy/Role — 2 Select Scope — 3 Finish   Go to Old Edition

Assign selected permissions to test_01.   Create Policy

View Selected (1)   Copy Permissions from Another Project   All policies/roles   Document Database Serv...   Enter a policy name, role name, or description.

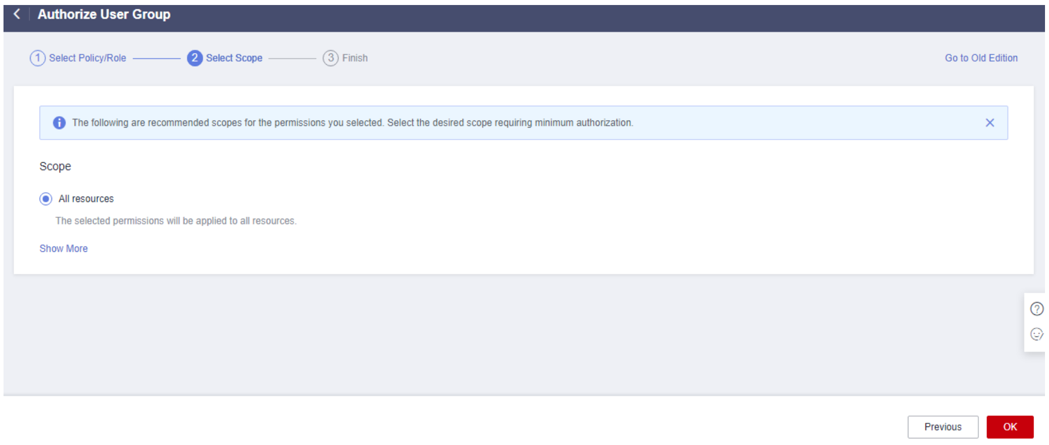
	Policy/Role Name	Type
☐	DDS ManageAccess Database administrator permissions for all operations except deleting DDS resources.	System-defined policy
☐	DDS FullAccess Full permissions for Document Database Service.	System-defined policy
☐	DDS Administrator Document Database Service Administrator	System-defined role
✓	DDS ReadOnlyAccess Read-only permissions for Document Database Service.	System-defined policy

Cancel   Next

Step 7 Specify the scope and click **OK**.

- All resources
- Region-specific projects: The selected permissions will be applied to resources in the region-specific projects you select.

Figure 14-5 Specifying the scope



Authorize User Group

1 Select Policy/Role — 2 Select Scope — 3 Finish   Go to Old Edition

The following are recommended scopes for the permissions you selected. Select the desired scope requiring minimum authorization.

Scope

● All resources

The selected permissions will be applied to all resources.

Show More

Previous   OK

----End

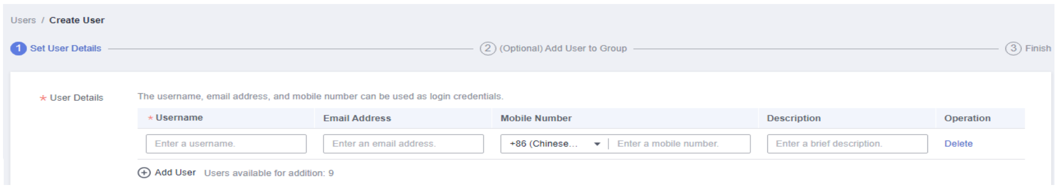
Step 2: Create an IAM User

IAM users can be created for employees or applications of an enterprise. Each IAM user has their own security credentials, and inherits permissions from the groups it is a member of. To create an IAM user, perform the following steps:

Step 1 On the IAM console, choose **Users** in the navigation pane. Then click **Create User**.

Step 2 Specify the user information on the **Create User** page. To create more users, click **Add User**. You can add a maximum of 10 users at a time.

Figure 14-6 Creating a user



Users / **Create User**

1 Set User Details —— 2 (Optional) Add User to Group —— 3 Finish

* User Details

The username, email address, and mobile number can be used as login credentials.

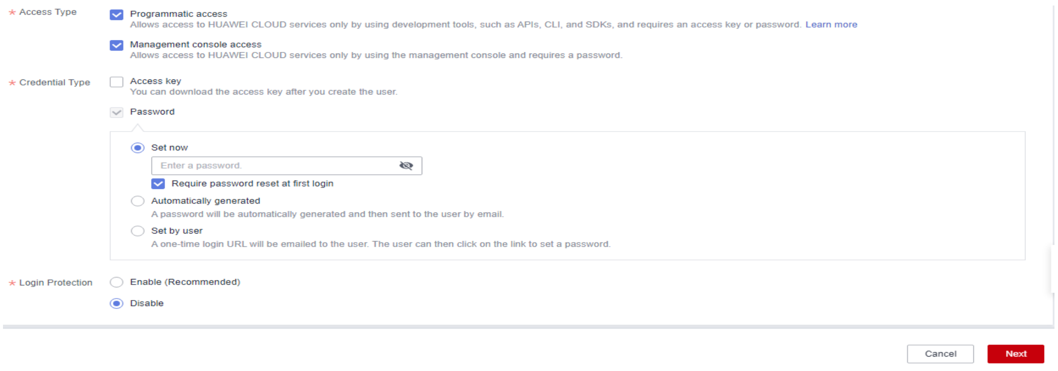
* Username	Email Address	Mobile Number	Description	Operation
Enter a username.	Enter an email address.	+86 (Chinese... ▾ Enter a mobile number.	Enter a brief description.	Delete

⊕ Add User Users available for addition: 9

- **Username**: Used for logging in to Huawei Cloud. For this example, enter **James**.
- **Email Address**: Email address bound to the IAM user. This parameter is mandatory if the access type is specified as **Set by user**.
- (Optional) **Mobile Number**: Mobile number bound to the IAM user.
- (Optional) **Description**: Description of the user.

Step 3 Configure required parameters and click **Next**.

Figure 14-7 Configuring user details



* Access Type
[✓] Programmatic access
Allows access to HUAWEI CLOUD services only by using development tools, such as APIs, CLI, and SDKs, and requires an access key or password. Learn more
[✓] Management console access
Allows access to HUAWEI CLOUD services only by using the management console and requires a password.

* Credential Type
[ ] Access key
You can download the access key after you create the user.
[✓] Password
(●) Set now
Enter a password.
[✓] Require password reset at first login
( ) Automatically generated
A password will be automatically generated and then sent to the user by email.
( ) Set by user
A one-time login URL will be emailed to the user. The user can then click on the link to set a password.

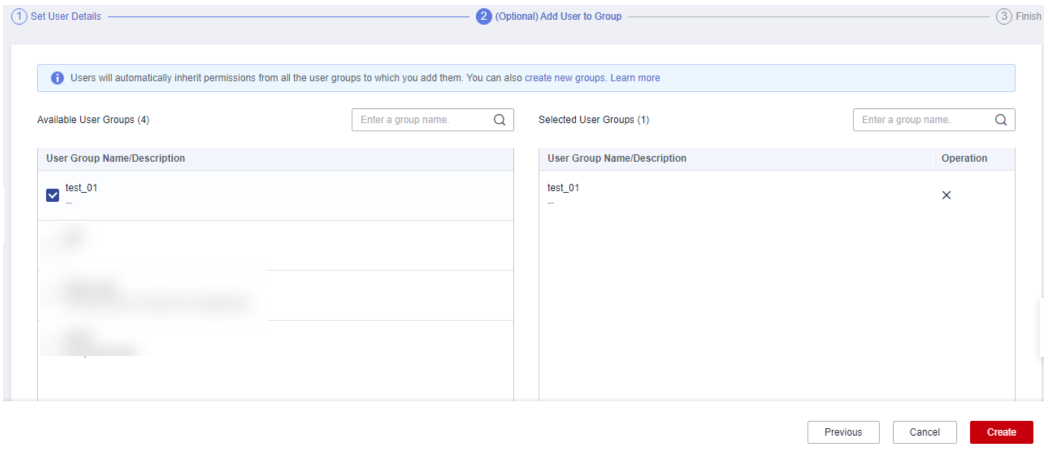
* Login Protection
( ) Enable (Recommended)
(●) Disable

Table 14-1 Configuration items

Parameter	Description
Access Type	• **Programmatic access**: Select this option to allow the user to access cloud services using development tools, such as APIs, CLI, and SDKs. You can generate an access key or set a password for the user. • **Management console access**: Select this option to allow the user to access cloud services using the management console. You can set or generate a password for the user or request the user to set a password at first login.
Credential Type	• Access key: Download the access key after the user is created. • Password: If you create multiple users, set a password for the users and determine whether to require the users to reset the password at first login. If you create one user, you can select **Automatically generated** and the system automatically generates a login password for the user.

Parameter	Description
Login Protection	To ensure account security, you are advised to select **Enable**. To enable or disable login protection for an IAM user after creation, see **Login Protection**.

Step 4 Add the users to user group created in **Step 4** and click **Create User**.



Users will automatically inherit permissions from all the user groups to which you add them. You can also create new groups. Learn more

Available User Groups (4)

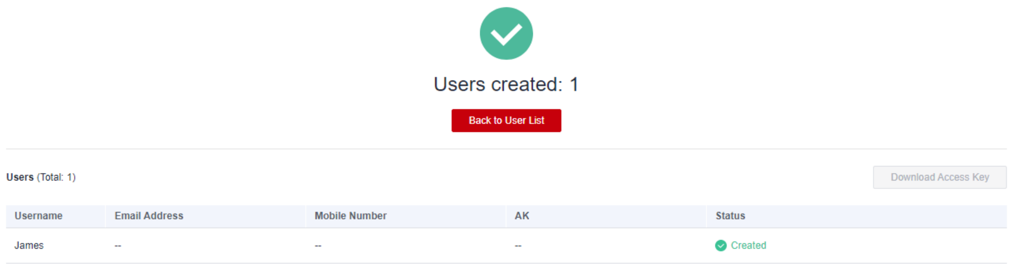
User Group Name/Description
☑ test_01 --

Selected User Groups (1)

User Group Name/Description	Operation
test_01 --	×

Figure 14-8 Creating a user

Step 5 Check the created users in the user list. If you select **Access key** for **Credential Type**, you can download the access key after you create the user. You can also manage the **Access Keys** on the **My Credentials** page.



Users created: 1

Users (Total: 1)

Username	Email Address	Mobile Number	AK	Status
James	--	--	--	Created

Figure 14-9 Viewing the results

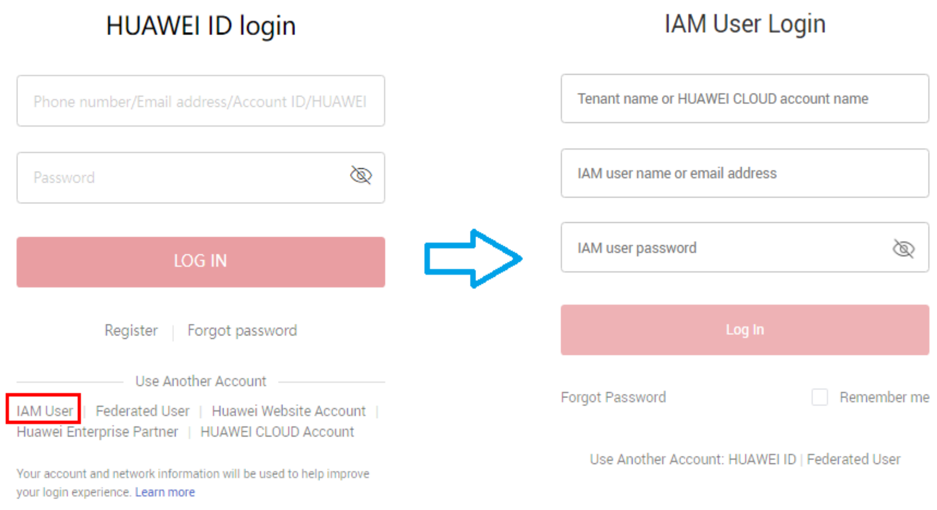
----End

Step 3: Log In and Verify Permissions

After the user is created, use the username and identity credential to log in to Huawei Cloud, and verify that the user has the permissions defined by the **DDS ReadOnlyAccess** policy.

Step 1 On the Huawei Cloud login page, click **IAM User** in the lower left corner.

Figure 14-10 IAM user login


HUAWEI ID login
Phone number/Email address/Account ID/HUAWEI
Password
LOG IN
Register | Forgot password
Use Another Account
IAM User | Federated User | Huawei Website Account | Huawei Enterprise Partner | HUAWEI CLOUD Account
Your account and network information will be used to help improve your login experience. Learn more
IAM User Login
Tenant name or HUAWEI CLOUD account name
IAM user name or email address
IAM user password
Log In
Forgot Password
Remember me
Use Another Account: HUAWEI ID | Federated User

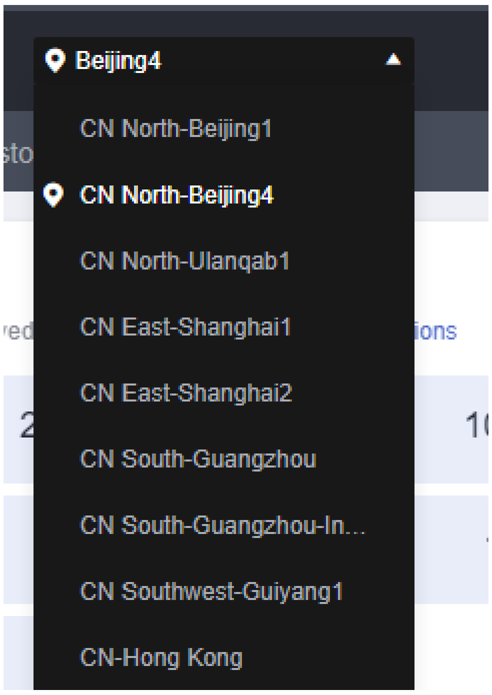
Step 2 Enter the account name, username, and password, and click **Log In**.

- The account name is the name of the Huawei account that created the IAM user.
- The username and password are those set by the account when creating the IAM user.

If the login fails, contact the entity owning the account to verify the username and password. Alternatively, you can reset the password by following the procedure in **Resetting Password for an IAM User**.

Step 3 After successful login, switch to a region where the user has been granted permissions on the management console.

Figure 14-11 Region



Step 4 Choose **Service List** > **Document Database Service**. Then click **Buy DB Instance** on the DDS console. If a message appears indicating insufficient permissions to perform the operation, the **DDS ReadOnlyAccess** policy has already taken effect.

Step 5 Choose any other service in the **Service List**. If a message appears indicating insufficient permissions to access the service, the **DDS ReadOnlyAccess** policy has already taken effect.

----End

15 Proper Use of Data Definition Languages (DDL) Statements

A data definition language (DDL) is a SQL used to create, modify, and delete the structure of databases and collections.

There are the following common DDLs in DDS:

- **createCollection**: used to create a collection in a specified database.
- **drop**: used to delete a specified collection.
- **createIndex**: used to create one or more indexes in a collection to improve query efficiency.
- **dropIndex**: used to delete a specified index from a collection.
- **shardCollection**: Run the **sh.shardCollection** command to modify the sharding rule of a collection so that the collection is distributed in different shards.
- **collection.stats**: used to view metadata of a specified collection, such as the number of documents and storage size.
- **db.stats**: used to view metadata of a specified database, such as the number of collections and storage.

Precautions for Using DDLs

- Creating indexes, deleting indexes, and deleting collections consume a large amount of I/O or compute resources. Perform these operations during off-peak hours to prevent affecting services.
- Do not execute multiple DDLs at the same time. Otherwise, the execution may fail due to blocking. For example, do not create an index and delete a collection at the same time.
- Create necessary indexes only to prevent storage waste caused by redundant indexes. For example, do not create an index based on the prefix field of a composite index.
- When creating an index, you need to create a background index using **db.<collection_name>.createIndex({ <field_name>: <index_type> }, { background: true })**. Note that creating a background index does not block other services, but still consumes a large amount of I/O resources.

- Before deleting an index, perform a performance test to ensure that the index to be deleted does not affect the query performance.
- Do not create too many collections. If there are too many collections, a large amount of metadata is generated, extra resources are consumed, and it is too difficult to perform maintenance. For example, do not name a collection by date or create a new collection every day. Instead, store the date as a field in a given collection.
- Before deleting a collection, confirm the collection name with caution because data cannot be directly restored after the collection is deleted. You are advised to back up important data first.
- If you want to delete a collection, do not use the **remove** or **delete** command without filtering conditions. Instead, use **db.<collection_name>.drop()** to delete a collection. If a query condition is specified for the **remove** or **delete** command, the corresponding index must be created.
- In a cluster instance, if a collection has a large storage space, you need to shard the collection. For details, see **Sharding**.
- You can use **db.stats()** and **db.<collection_name>.stats()** to view the metadata (such as the number of documents and storage size) of databases and collections. The information is important for performance optimization and capacity planning.

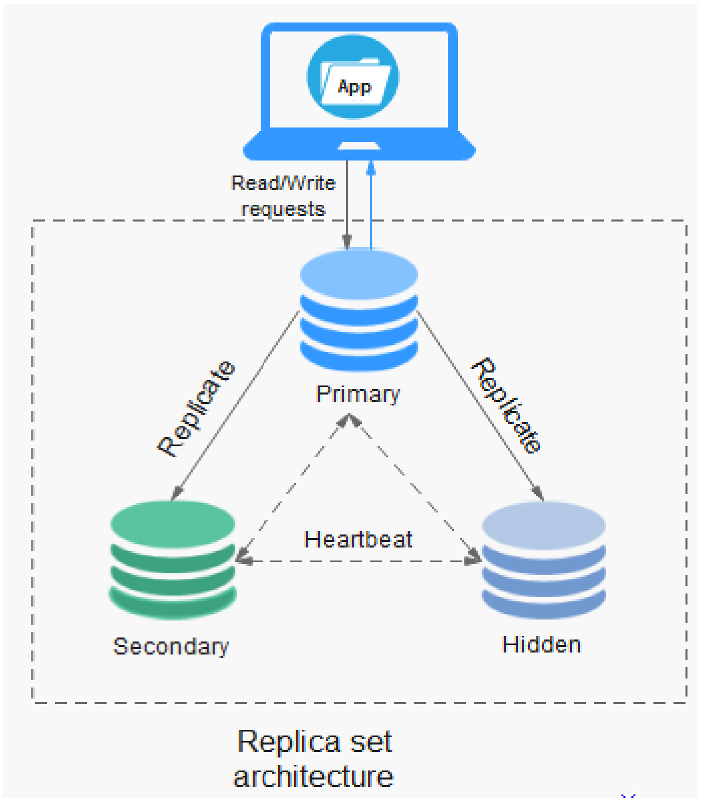
16 How Is a DDS Node Going to Be Disconnected and What Can I Do?

A replica set consists of three nodes: primary, secondary, and hidden. The three-node architecture is set up automatically, and the three nodes automatically synchronize data with each other to ensure data reliability. Replica sets are recommended for small- and medium-sized service systems that require high availability.

- Primary node: Primary nodes are used to process both read and write requests.
- Secondary node: Secondary nodes are used to process read requests only.
- Hidden node: Hidden nodes are used to back up service data.

You can perform operations on the primary and secondary nodes. If the primary node is faulty, the system automatically selects a new primary node. The following figure shows the replica set architecture.

Figure 16-1 Three-node replica set architecture



DDS can write data only on the primary node. When data is written to the primary node, oplogs are generated. The secondary and hidden nodes read oplogs from the primary node for replay to ensure data consistency.

The storage capacity of oplogs is determined by the value of **oplogSize** (10% of the default disk capacity).

- How is the primary/secondary latency generated?

  If the write speed of the primary node is too fast and exceeds the replay speed of the oplog read on the secondary node, the primary/secondary latency occurs.

- When is a node going to be disconnected?

  The storage capacity of oplogs is limited. If the capacity reaches the upper limit, the earliest oplog will be deleted. The secondary node reads oplogs and records the last oplog each time. If the primary/secondary latency reaches a certain value, the secondary node finds that the last oplog point has been deleted. In this case, the secondary node cannot continue to read oplogs, and the secondary node is disconnected.

- How to effectively prevent the secondary node from being disconnected?
  - Set **writeConcern** to **majority**. In this way, data is written to a majority of nodes, ensuring data consistency.
  - Increase the oplog storage capacity by changing the value of **oplogSizePercent** on the console. For details, see **Modifying DDS DB Instance Parameters**.

– Perform time-consuming DDL operations (such as index creation) and data backup operations during off-peak hours to avoid burst addition, deletion, and modification operations.

NOTE

If **writeConcern** is not set to **majority**, the data that is not synchronized to the secondary node may be lost when a primary/secondary switchover occurs.