Alibaba Cloud ApsaraDB for MongoDB

Best Practices

Issue: 20190704

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. **You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.**
2. **No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.**
3. **The content of this document may be changed due to product version upgrades , adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.**
4. **This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults " and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity , applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use**

or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified , reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates . The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).
6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 Danger: Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 Warning: Restarting will cause business interruption. About 10 minutes are required to restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus , page names, and other UI elements.	Click OK.
`Courier font`	It is used for commands.	Run the `cd /d C:/windows` command to enter the Windows system folder.
Italics	It is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *[-all|-t]*

Style	Description	Example
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich {stand | slave}`

Contents

1 Set common alert rules for monitoring ApsaraDB for MongoDB

ApsaraDB for MongoDB provides an instance status monitoring and alerting feature. This topic describes how to configure common metrics such as the usage of the disk space, input/output operations per second (IOPS), connections, and CPU.

Background

- With the growth of data and development of business, more and more performanc e resources of ApsaraDB for MongoDB instances are consumed and even used up.
- In some scenarios, lots of performance resources of ApsaraDB for MongoDB instances are abnormally consumed. For example, a great number of slow queries increase the CPU usage and a large amount of data is written to fully occupy the disk space.

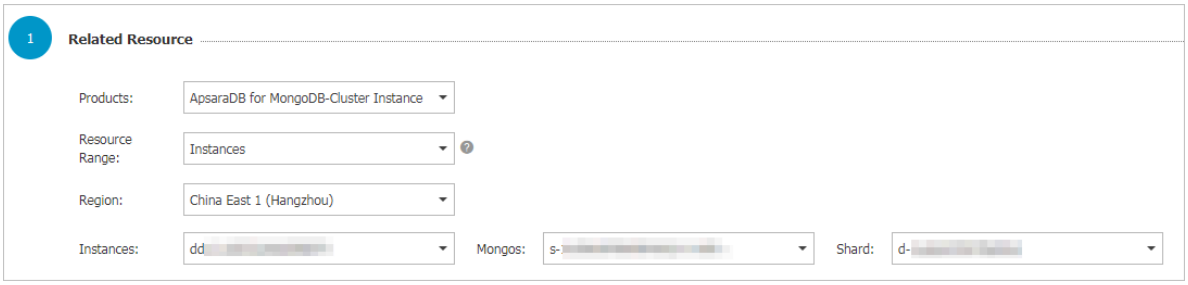
> Note:
> Instances with insufficient disk space may be locked. If an instance is locked, you can submit a ticket. After unlocking the instance, you can change the configuration to increase its disk space.

You can set alert rules for monitoring the key performance metrics of instances to help you detect abnormal data in a timely manner and quickly locate and handle faults.

Procedure

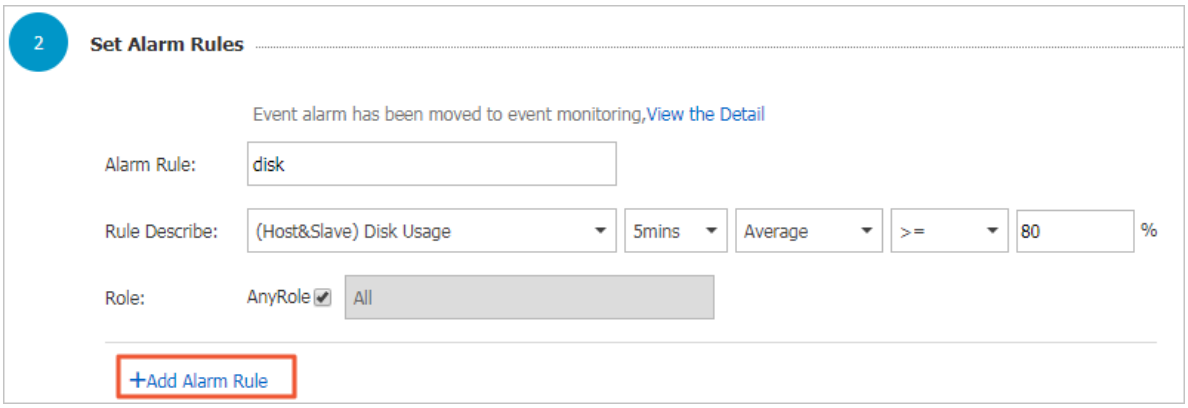
1. Log on to the ApsaraDB for MongoDB console.
2. In the upper-left corner of the home page, select the region where the target instance is located.
3. Locate the target instance and click its instance ID.
4. In the left-side navigation pane, click Alarm Rules.
5. Click Set Alarm Rule to jump to the CloudMonitor console.
6. In the upper-right corner of the CloudMonitor console, click Create Alarm Rule.

7. On the Create Alarm Rule page that appears, specify related resources.



Parameter	Description
Products	The architecture of the instance. · ApsaraDB for MongoDB-Instance Copy · ApsaraDB for MongoDB-Cluster Instance · ApsaraDB for MongoDB-Single node instance Note: If you select ApsaraDB for MongoDB-Cluster Instance, you need to select mongos nodes and shards to be monitored for Mongos and Shard, respectively.
Resource Range	· If you select All Resources, the alerting service sends an alert notification when any ApsaraDB for MongoDB instances match alert rules. · If you select Instances, the alerting service sends an alert notification when any selected ApsaraDB for MongoDB instances match alert rules.
Region	The region where the instance is located.
Instances	The ID of the instance to be monitored. You can select multiple instance IDs.

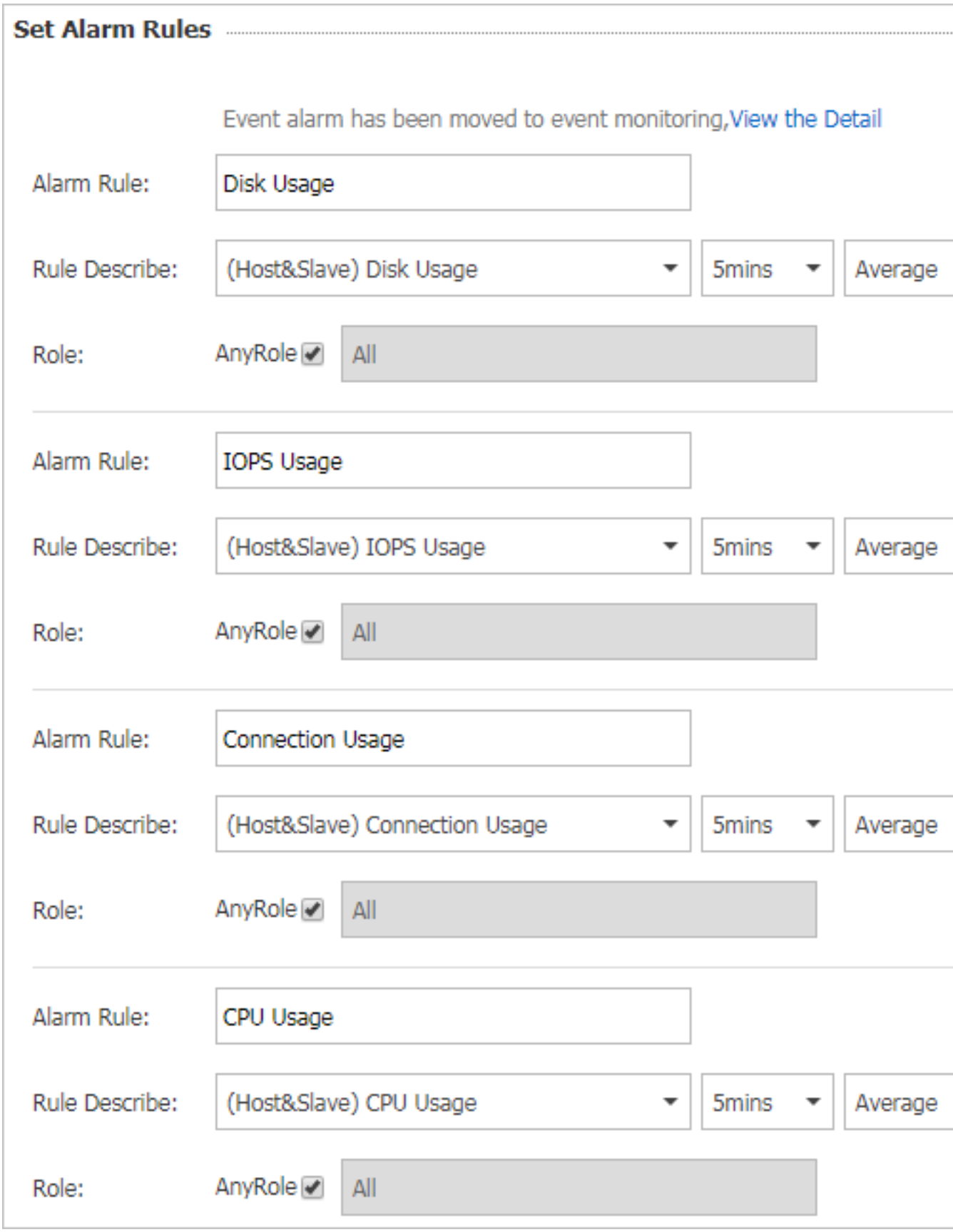
8. Set alert rules. You need to set the disk space usage first, and then click Add Alarm Rule.



Note:

- For example, if you set Rule Describe to Disk Usage 5mins Average >= 80%, the alerting service checks the disk space usage every 5 minutes to detect whether the average disk space usage in the last 5 minutes is greater than or equal to 80 %. You can adjust each alert threshold based on your business scenarios.
- If you select AnyRole for Role, the alerting service monitors all primary and secondary nodes for each instance.

9. Repeat the last step to set alert rules for the usage of the IOPS, connections, and CPU.



10. Set other parameters for alert rules.

Parameter	Description
Mute for	The interval at which the alerting service sends an alert notification repeatedly if an alert is not cleared.
Triggered when threshold is exceeded for	The number of consecutive times that the alerting service detects data that exceeds an alert threshold so as to generate an alert. We recommend that you set the value to 3. For example, if you set Rule Describe to CPU Usage 5mins Average > 80%, the alerting service generates an alert after it detects that the average CPU usage within 5 minutes exceeds 80% three consecutive times.
Effective Period	The period during which alert rules take effect.

11. Configure notification methods.

Parameter	Description
Notification Contact	The contacts or contact group to which the alerting service sends an alert notification. For more information, see Manage alert contacts and alert contact groups.
Notification Methods	The notification methods corresponding to the alert severity, which can be Critical, Warning, or Info. · Critical: Email + DingTalk · Warning: Email + DingTalk · Info: Email + DingTalk
Email Subject	The subject of the alert notification email. You can customize an email subject. The default email subject is as follows: Product + Metric + Instance ID.
Email Remark	The custom additional information of the alert notificati on email. After you enter email remarks, the alerting service sends an alert notification email that contains your custom remarks.
HTTP CallBack	For more information, see Use alert callback.

12. Click Confirm. Alert rules automatically take effect.

2 Migrate Azure Cosmos DB API for MongoDB to Alibaba Cloud

MongoDB provides native backup utilities that you can use to migrate Azure Cosmos DB API for MongoDB to Alibaba Cloud.

Notes

- This data migration is a full migration. To avoid data inconsistencies, we recommend that you stop all write operations to the database before migration.
- If you have used mongodump commands to back up the database, move the files in the dump folder to other directories. Make sure that the default dump folder is empty before data migration. Otherwise, existing backup files in this folder will be overwritten.
- Run mongodump and mongorestore commands on servers on which MongoDB is installed. Do not run these commands in the mongo shell.

Required database permissions

Migration type	Full migration
Azure Cosmos DB	Read
Target MongoDB instance	Read and write

Environment configuration

1. Create an ApsaraDB for MongoDB instance. For more information, see Create an instance.

   Note:

   - The instance storage capacity must be larger than Azure Cosmos DB.
   - Select MongoDB version 3.4.

2. Set a password for the ApsaraDB for MongoDB instance. For more information, see Set a password.
3. Install MongoDB on a server. For more information, see Install MongoDB.

   Note:

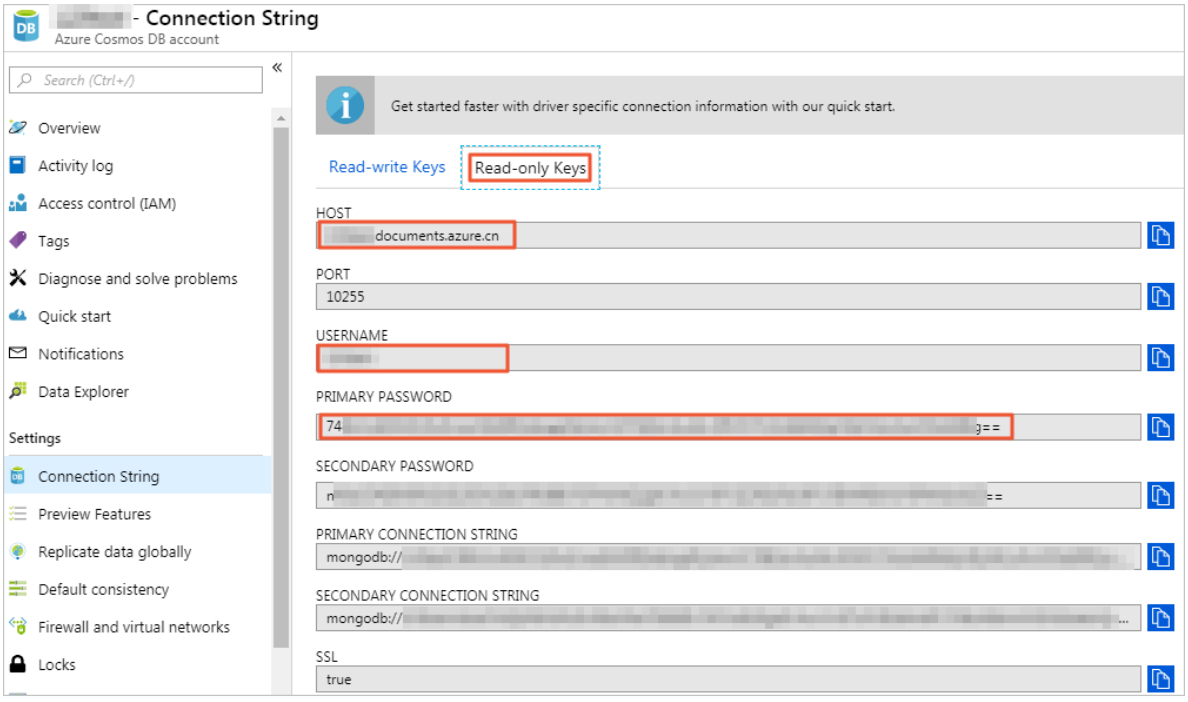
- Install MongoDB 3.0 or a later version.
- This server is used to temporarily store data during backup and recovery, and is not needed after the migration is complete.
- The capacity of the disk where the backup is stored must be larger than Azure Cosmos DB.

This example installs MongoDB on a Linux server. You can also use other operating systems, such as Windows.

Procedure

1. Log on to the Azure portal.
2. Click Azure Cosmos DB from the left-side navigation pane.
3. On the Azure Cosmos DB page, click the account name of the Azure Cosmos DB that you want to migrate.
4. On the account details page, click Connection String.
5. Click the Read-only Keys tab to view the database connection information.

Figure 2-1: Azure connection information



Note:

To migrate data, you only need a database account that has read-only permissions.

6. Run the following command on the MongoDB server to back up the Azure Cosmos DB to this server.

```
mongodump  -- host  < HOST >: 10255  -- authentica  tionDataba  se
  admin  - u  < USERNAME > - p  < PRIMARY   PASSWORD > -- ssl  --
sslAllowIn  validCerti  ficates
```

   Note: Replace <HOST>, <USERNAME>, and <PRIMARY PASSWORD> with the corresponding values shown in the Azure connection information figure.

   After the backup is complete, backups of the Azure Cosmos DB are stored in the dump folder.

7. Obtain the endpoint of the primary node of the ApsaraDB for MongoDB instance. For more information, see Retrieve the seven connection elements.
8. Run the following command on the MongoDB server to export the backups to the ApsaraDB for MongoDB instance.

```
 mongoresto  re  -- host  < mongodb_ho  st >: 3717  -- authentica
tionDataba  se  admin  - u  < username > - p  < password >  dump
```

   Description:

   - <mongodb_host>: The endpoint of the primary node of the MongoDB instance.
   - <username>: The username for the MongoDB instance.
   - <password>: The password for the MongoDB instance.

After the recovery is complete, backups of the Azure Cosmos DB are migrated to the ApsaraDB for MongoDB instance.

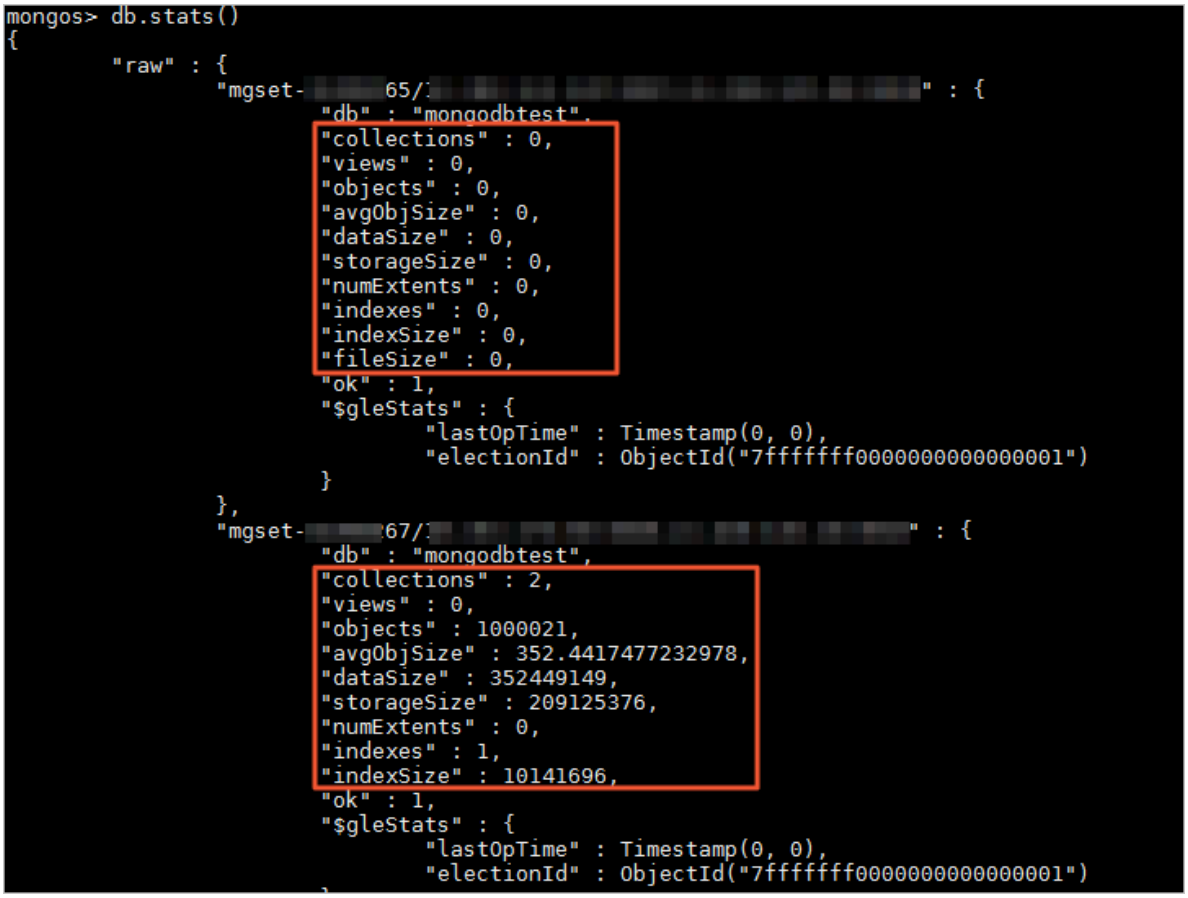
3 Configure sharding to maximize the performance of shards

You can configure sharding at the collection level for databases in a sharded cluster instance to make full use of the storage space and maximize the computing performance of shards in the sharded cluster.

Precautions

- This operation applies only to sharded cluster instances.
- After you configure sharding, the balancer shards existing data that meets the specified criteria. Because sharding affects the performance of an instance, we recommend that you perform this operation during off-peak hours.
- You cannot change the configured shard key after sharding.
- The choice of a shard key affects the performance of a sharded cluster instance. For more information about how to choose a shard key, see Shard Keys.

- If you do not configure sharding, data is written to the primary shard. In this case, you cannot make full use of the storage space and maximize the computing performance of other shards in the same sharded cluster.



Procedure

The following procedure uses the database named mongodbtest and the collection named customer as an example.

1. Connect to a sharded cluster instance through the mongo shell.
2. Enable sharding for the database where the collection to be sharded is located.

```
sh . enableShar  ding ("< database >")
```

In the preceding command, <database> indicates the database name.

Example:

```
sh . enableShar  ding (" mongodbtes  t ")
```

> **Note:**
> You can run the `sh . status ()` command to check the sharding status.

3. Create an index on a field in the collection.

```
db .< collection >. createInde  x (< keys >,< options >)
```

Notes:

- <collection>: The collection name.
- <keys>: The field and value pair where the field is the index key and the value describes the sort order for this field.

  A value of `1` indicates an ascending index on the field. A value of `-1` indicates a descending index on the field.
- <options>: The additional options. For more information, see db.collection.createIndex(). This parameter is not used in this example.

Example:

```
db . customer . createInde  x ({" name ": 1 })
```

4. Configure sharding for the collection.

```
sh . shardColle  ction ("< database >.< collection >",{ "< key >":<
value > } )
```

Notes:

- <database>: The database name.
- <collection>: The collection name.
- <key>: The shard key, based on which ApsaraDB for MongoDB shards data.
- <value>:
  - 1: indicates an ascending index on the shard key, which can properly support range queries based on the shard key.
  - -1: indicates a descending index on the shard key, which can properly support range queries based on the shard key.
  - hashed: indicates a hashed shard key, which can be used to write data evenly to various shards.

[DO NOT TRANSLATE]

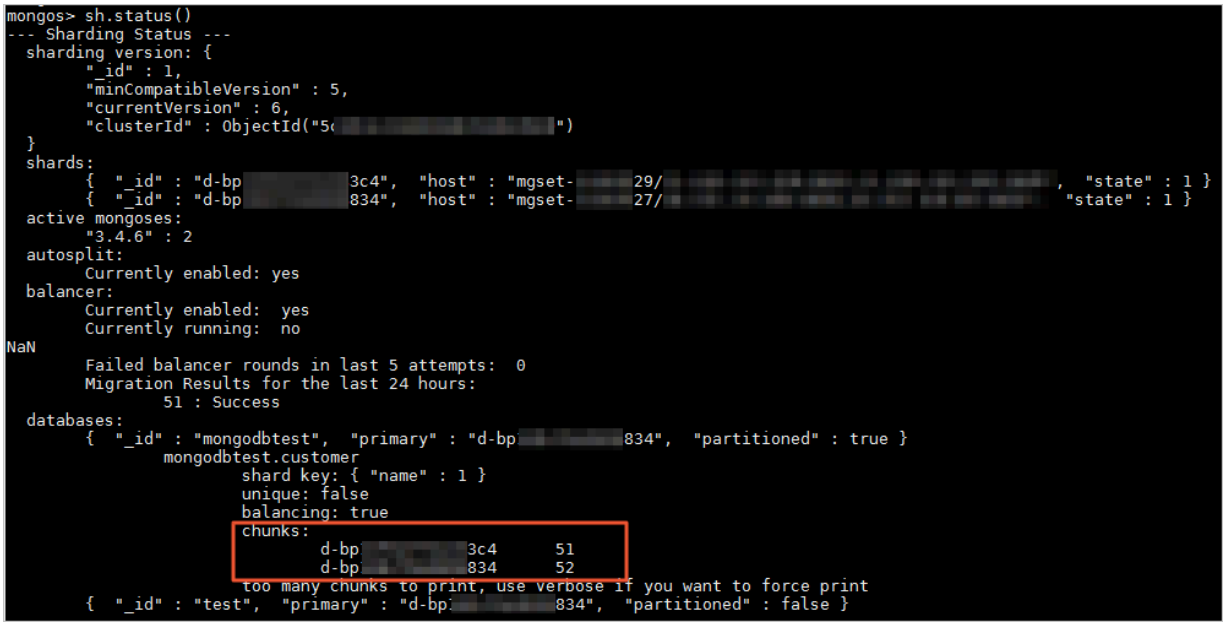
Example:

```
sh . shardColle  ction (" mongodbtes  t . customer ",{" name ": 1
})
```

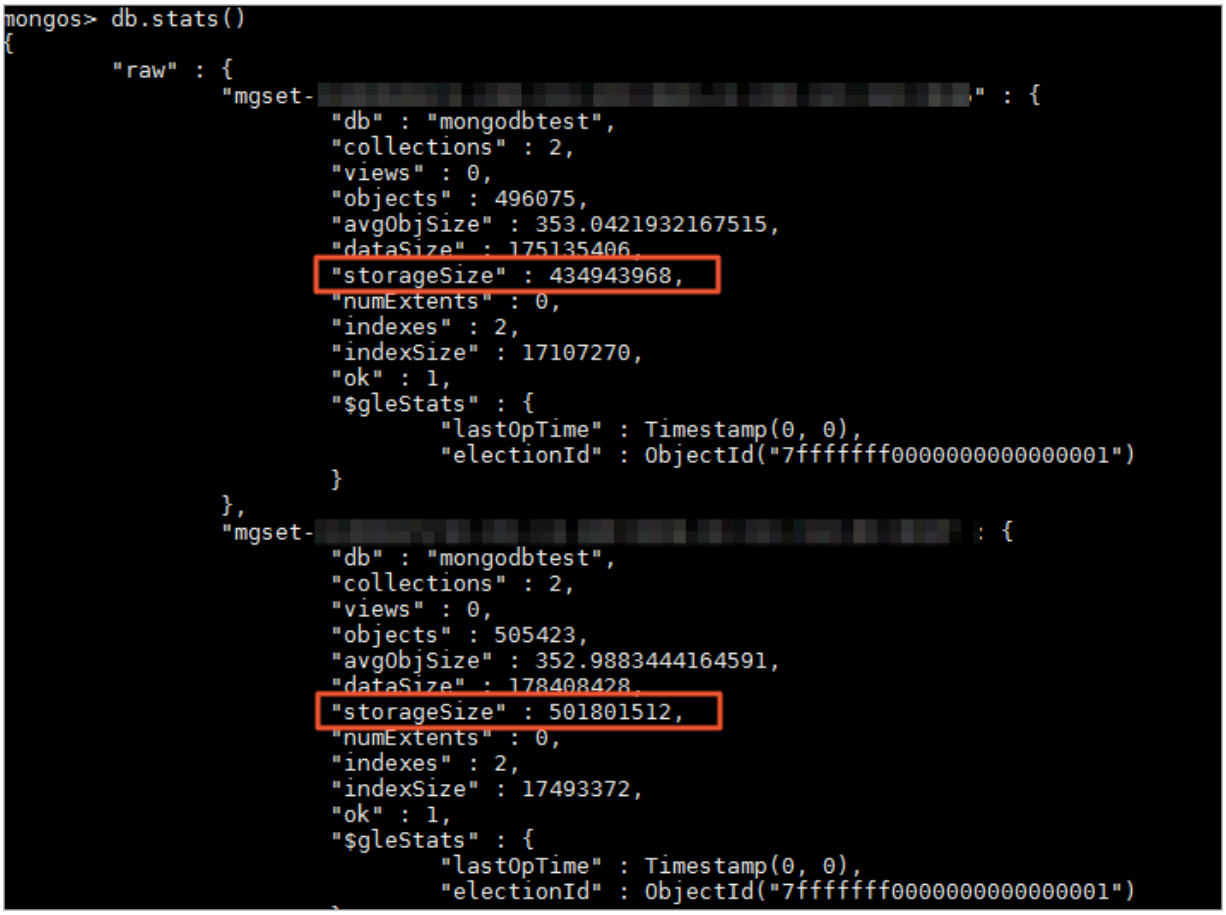
If this collection contains data in the database, the backend balancer can automatica lly shard data after your configuration. The sharding process is transparent to you.

Subsequent operations

After the database has been running and data has been written for a while, you can run the `sh . status ()` command in the mongo shell to check the sharding configuration and the chunk information on shards.



You can also run the `db . stats ()` command to check the data storage of this database on various shards.


mongos> db.stats()
{
"raw" : {
"mgset-
" : {
"db" : "mongodbtest",
"collections" : 2,
"views" : 0,
"objects" : 496075,
"avgObjSize" : 353.0421932167515,
"dataSize" : 175135406,
"storageSize" : 434943968,
"numExtents" : 0,
"indexes" : 2,
"indexSize" : 17107270,
"ok" : 1,
"$gleStats" : {
"lastOpTime" : Timestamp(0, 0),
"electionId" : ObjectId("7fffffff0000000000000001")
}
},
"mgset-
: {
"db" : "mongodbtest",
"collections" : 2,
"views" : 0,
"objects" : 505423,
"avgObjSize" : 352.9883444164591,
"dataSize" : 178408428,
"storageSize" : 501801512,
"numExtents" : 0,
"indexes" : 2,
"indexSize" : 17493372,
"ok" : 1,
"$gleStats" : {
"lastOpTime" : Timestamp(0, 0),
"electionId" : ObjectId("7fffffff0000000000000001")

4 Sort database fragments to improve disk usage

Fragmentation occurs when you frequently write and delete data on MongDB. These fragments occupy disk space and reduce disk usage. You can rewrite and defragment all the data and indexes in a collection to release unused space, improving disk usage and query performance.

Precautions

- We recommend that you back up the database before performing this operation.
- The database is locked during defragmentation and read/write operations are blocked. We recommend that you perform this operation during off-peak hours.
- We do not recommend that you frequently perform this operation.

Operation example of standalone instances or replica set instances

1. Connect to the primary node of ApsaraDB for MongoDB through the mongo shell. For more information, see Use the mongo shell to connect to an instance.
2. Run the following command to switch to the database where the collection is stored.

   ```
   use  < database_n  ame >
   ```

   Command description:

   <database_name>: the name of the database.
3. Run the following command to defragment a collection:

   ```
   db . runCommand ({ compact :"< collection  _name >", force : true
   })
   ```

   Command description:

   <collection_name>: the name of the collection.

   Note:
   The force parameter is optional.

   - If the value is true, the compact command can be executed on the primary node of a replica set.
   - If the value is false, the compact command executed on the primary node returns an error.

4. Wait for the command execution to complete. If `{" OK ": 1 }` is returned, the command is executed.

> Note:
> The compact operation is not passed to the secondary node. When the instance is a replica set instance, repeat the preceding steps to connect to the secondary node through the mongo shell and run the compact command.

After defragmentation is complete, you can run the `db . stats ()` command to view the disk space occupied by the database.

Operation example of sharded cluster instances

1. Connect to any mongos in the sharded cluster instance through the mongo shell. For more information, see Connect to an ApsaraDB for MongoDB sharded cluster instance through the mongo shell.
2. Run the following command to defragment a collection in the primary node of the shard:

```
db . runCommand ({ runCommand  OnShard :"< Shard   ID >"," command
":{ compact :"< collection  _name >", force : true }})
```

Command description:

<Shard ID>: the ID of the shard.

<collection_name>: the name of the collection.

3. Run the following command to defragment a collection in the secondary node of the shard:

```
db . runCommand ({ runCommand  OnShard :"< Shard   ID >","
command ":{ compact :"< collection  _name >"}, queryOptio  ns : {$
readPrefer  ence : { mode : ' secondary '}}})
```

Command instructions:

<Shard ID>: the ID of the shard.

<collection_name>: the name of the collection.

After the defragmentation is complete, you can run the `db . runCommand ({ dbstats : 1 })` command to view the disk space occupied by the database after defragmentation.

5 Manage the ApsaraDB for MongoDB balancer

ApsaraDB for MongoDB allows you to manage the balancer. In some special business scenarios, you can enable or disable the balancer, set an active window, and perform other operations related to the balancer.

Precautions

- The balancer is a feature of the sharded cluster architecture and applies only to sharded cluster instances.
- Because operations related to the balancer may occupy the resources of an instance, we recommend that you manage the balancer during off-peak hours.

Disable the balancer

ApsaraDB for MongoDB enables the balancer by default. To disable the balancer in special business scenarios, follow these steps:

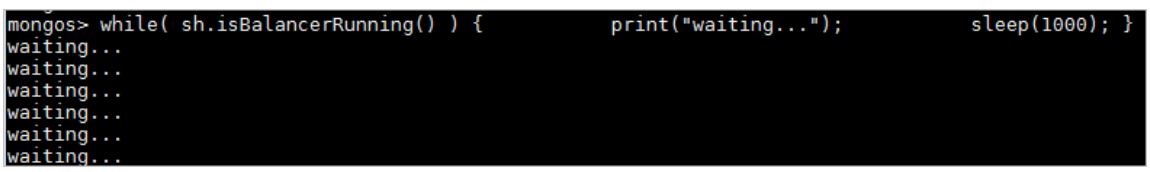
1. Connect to ApsaraDB for MongoDB through the mongo shell.
2. After connecting to a mongos node, run the following command in the mongo shell to switch to the config database:

```
use   config
```

3. Run the following command to check the running status of the balancer:

```
while (  sh . isBalancer  Running () ) {
          print (" waiting ...");
          sleep ( 1000 );
}
```

- If the command does not return any values, the balancer is not running any tasks. You can proceed to disable the balancer.
- If the command returns waiting, the balancer is migrating chunks. In this case, you cannot disable the balancer, otherwise data may become inconsistent.



4. After confirming that the balancer is not running any tasks, run the following command to disable the balancer:

```
sh . stopBalanc  er ()
```

Enable the balancer

If you have configured sharding, the balancer can immediately start a balancing procedure among shards after being enabled. Because balancing occupies the resources of an instance, we recommend that you perform this operation during off-peak hours.

1. Connect to ApsaraDB for MongoDB through the mongo shell.
2. After connecting to a mongos node, run the following command in the mongo shell to switch to the config database:

```
use   config
```

3. Run the following command to enable the balancer:

```
sh . setBalance  rState ( true )
```

Set an active time window for the balancer

To avoid negative impact on your business while the balancer is migrating chunks, you can set an active time window to allow the balancer to migrate chunks only within the specified period of time.

> Note:
> Before performing this operation, you must ensure that the balancer is enabled. For more information about how to enable the balancer, see Enable the balancer.

1. Connect to ApsaraDB for MongoDB through the mongo shell.
2. After connecting to a mongos node, run the following command in the mongo shell to switch to the config database:

```
use   config
```

3. Run the following command to set an active time window for the balancer:

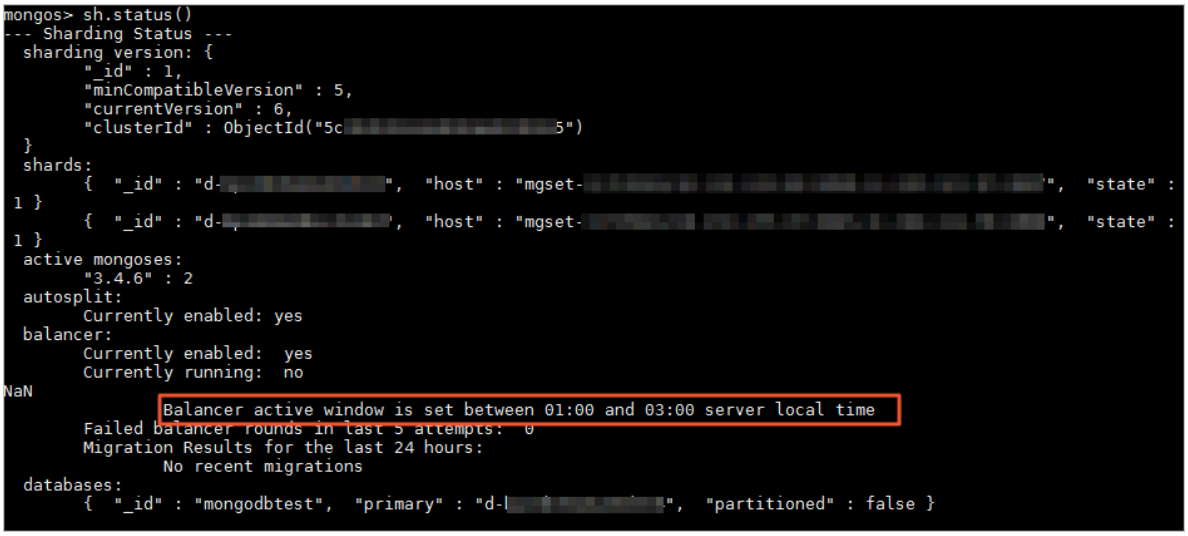
```
db . settings . update (
  {  _id : " balancer " },
  { $ set : {  activeWind  ow  : {  start  : "< start - time >",
stop  : "< stop - time >" } } },
  {  upsert :  true  }
```

```
)
```

Note:

- <start-time>: The start time in HH:MM format, where the value range of HH is 00–23 and that of MM is 00–59.
- <stop-time>: The end time in HH:MM format, where the value range of HH is 00–23 and that of MM is 00–59.

You can run the `sh . status ()` command to check the active time window of the balancer. For example, the following command output shows that the active time window of the balancer is 01:00–03:00.



You can also perform other related operations. For example, to ensure that the balancer is always running, you can run the following command to clear the active time window settings:

```
db . settings . update ({ _id : " balancer " }, { $ unset : {
activeWind ow : true } })
```

6 Connect to a replica set instance to achieve read/write split and high availability

An ApsaraDB for MongoDB replica set instance provides multiple copies of data to ensure the high reliability of data. It also provides an automatic failover mechanism to guarantee the high availability of services. You need to use a correct method to connect to a replica set instance to achieve high availability. You can also configure the connection to split read and write.

Before you start

- The primary node of a replica set instance is not permanent. A failover between the primary and secondary nodes may be triggered when nodes of the replica set instance are upgraded in turn, the primary node is faulty, or the network is partitioned. In these scenarios, the replica set can elect a new primary node and downgrade the original primary node to a secondary node.
- If you have used the address of the primary node to directly connect to the primary node of a replica set instance, the primary node needs to bear heavy load to process all read and write operations. If a failover is triggered in the replica set instance and the connected primary node is downgraded to a secondary node, you can no longer perform write operations and your business is seriously affected.

Connection string URI

To correctly connect to a replica set instance, you need to understand the connection string URI format of MongoDB. All official MongoDB drivers allow you to use a connection string URI to connect to MongoDB.

```
mongodb ://[ username : password @] host1 [: port1 ][, host2 [: port2
],...[, hostN [: portN ]]][/[ database ][?  options ]]
```

Notes:

- mongodb://: The prefix, which indicates that this address is a connection string URI.
- `username : password @`: The username and password used to log on to a database. If authentication is enabled, a password is required.

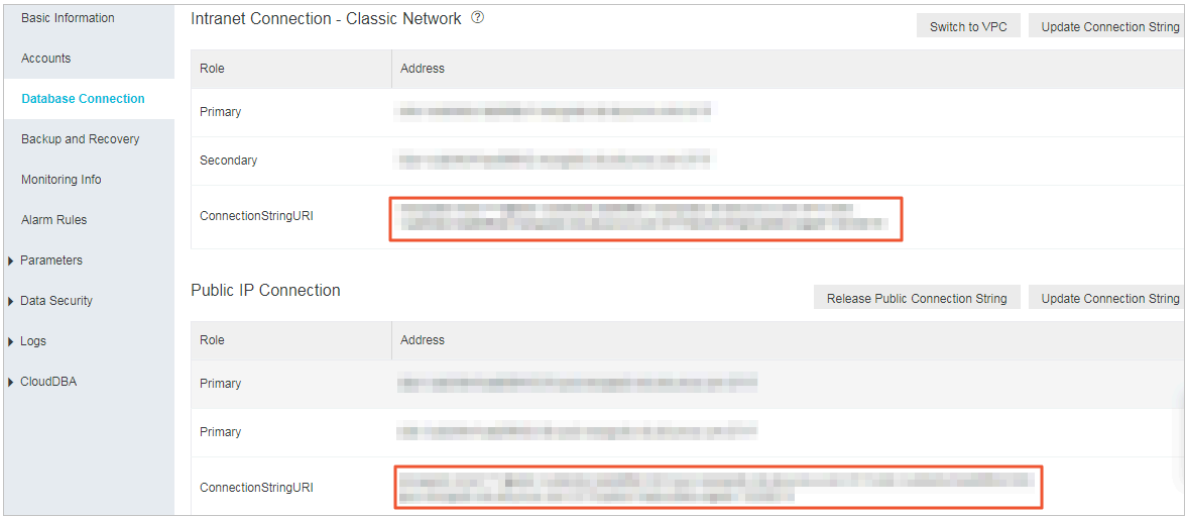
- `hostX : portX` : The list of addresses used to connect to nodes in the replica set instance. Each address consists of an IP address and a port number. Multiple addresses are separated with commas (,).
- `/ database` : The database corresponding to the username and password if authentication is enabled.
- `? options` : The additional connection options.

> **Note:**
> For more information about the connection string URI, see Connection String URI Format.

Use a connection string URI to connect to a replica set instance

ApsaraDB for MongoDB allows you to use a connection string URI to connect to a replica set instance.

1. Obtain the connection string URI of a replica set instance. For more information, see Obtain the replica set instance connection information.



2. Use the obtained connection string URI to connect your applications to the instance. For more information, see Connection sample code for MongoDB drivers.

> **Note:**
> To split read and write, you need to add `readPrefer ence = secondaryP referred` to options in a connection string URI to set read preference to secondary nodes.

For more information about read preference options, see Read Preference.

The following is an example of a connection string URI with the read preference specified:

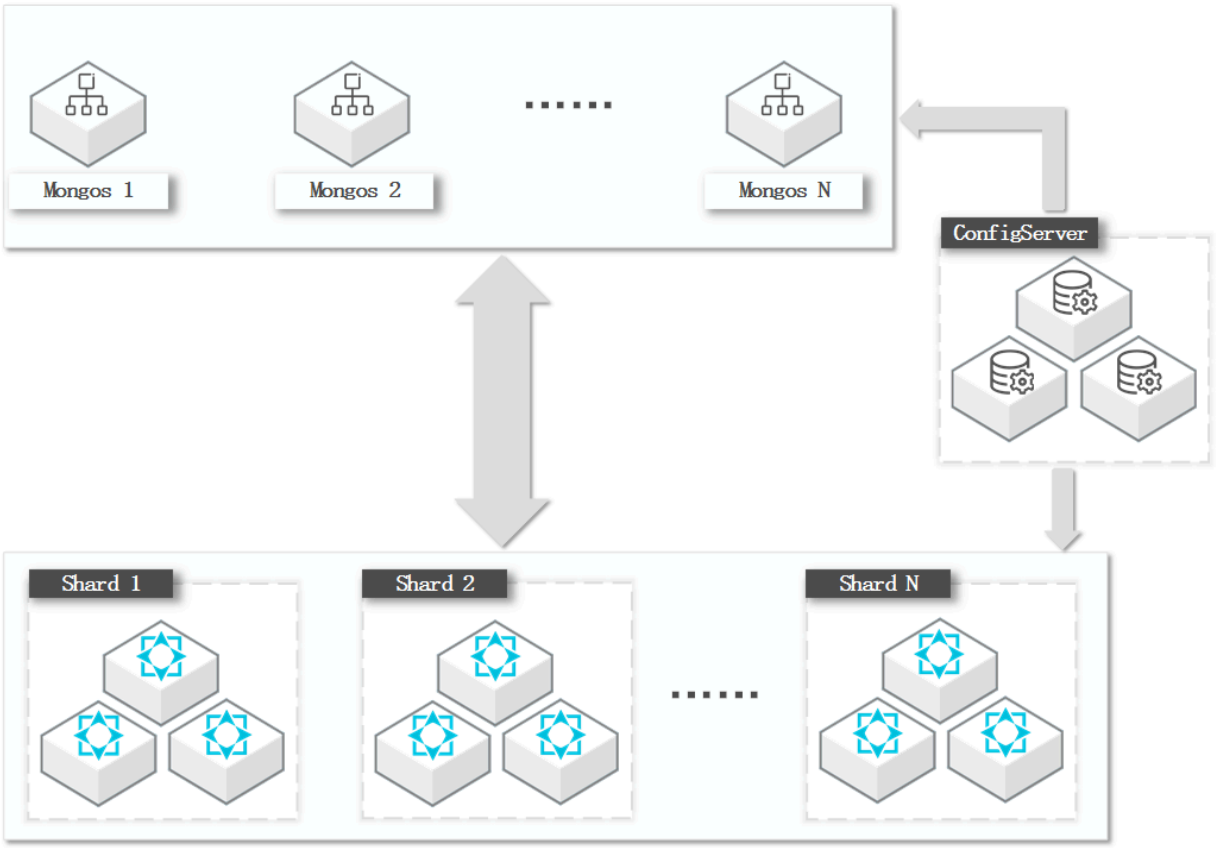
```
mongodb :// root : xxxxxxxx @ dds - xxxxxxxxxx  xx : 3717 ,
xxxxxxxxxx  xx : 3717 / admin ?  replicaSet = mgset - xxxxxx &
readPrefer  ence = secondaryP  referred
```

After you use the preceding method to connect to a replica set instance, a client can preferentially send read requests to secondary nodes to achieve read/write split. In the meantime, the client automatically detects the relationship between the primary and secondary nodes. If the primary node is changed, the client automatically switches over write operations to the new primary node to guarantee the high availability of services.

7 Use a connection string URI to connect to a sharded cluster instance

An ApsaraDB for MongoDB sharded cluster instance provides connection information for each mongos node. You can connect to a mongos node to access ApsaraDB for MongoDB. However, you need to use a correct method to connect to a sharded cluster instance to achieve load balancing and high availability.

Background



A MongoDB sharded cluster distributes and stores data on multiple shards to facilitate high scalability. When creating a sharded cluster, MongoDB introduces a config server to store the metadata of the cluster and introduces one or more mongos nodes to provide the entry to the cluster for applications. The mongos nodes read routing information from the config server to route requests to corresponding shards at the backend.

- When you connect to a mongos node, it can function as a mongod process.
- All mongos nodes are equal. You can connect to one or more mongos nodes to access a sharded cluster.

- The mongos nodes are stateless and can be scaled out as required. The service capability of a sharded cluster is subject to the smaller one between the total service capability of shards and that of mongos nodes.
- When you access a sharded cluster, we recommend that you share the load of applications evenly among multiple mongos nodes.

Connection string URI

To correctly connect to a sharded cluster instance, you need to understand the connection string URI format of MongoDB. All official MongoDB drivers allow you to use a connection string URI to connect to MongoDB.

The following example shows a connection string URI.

```
mongodb ://[ username : password @] host1 [: port1 ][, host2 [: port2
],...[, hostN [: portN ]]][/[ database ][?  options ]]
```

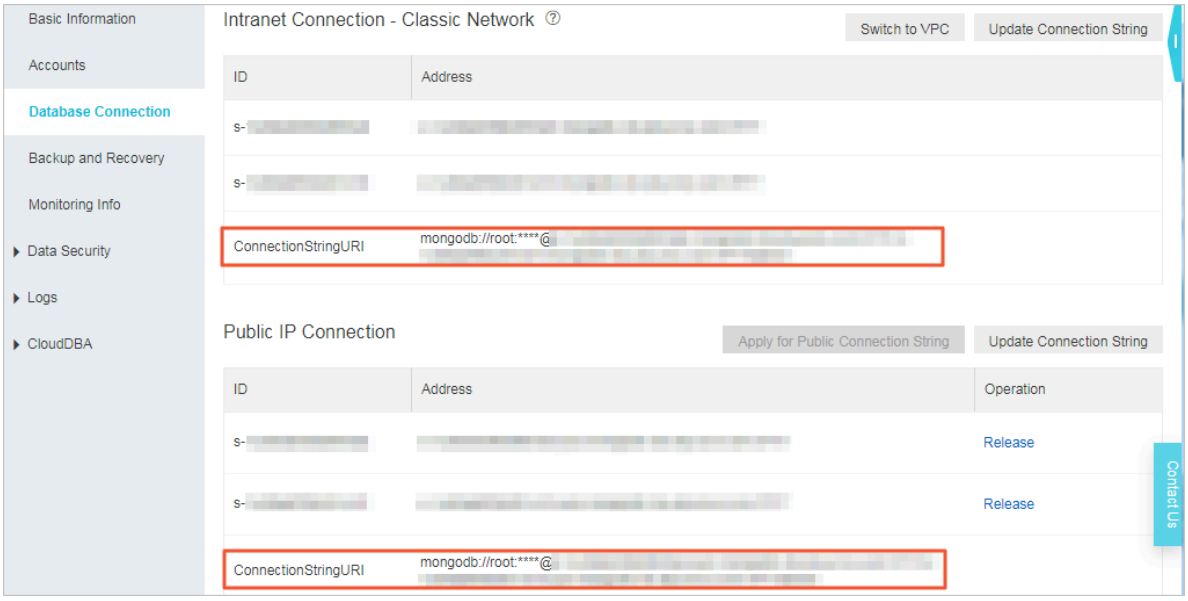
Note:

- `mongodb ://`: The prefix, which indicates that this address is a connection string URI.
- `username : password @`: The username and password used to log on to a database if authentication is enabled.
- `hostX : portX` : The list of addresses used to connect to mongos nodes.
- `/ database` : The database corresponding to the username and password if authentication is enabled.
- `? options` : The additional connection options.

Use a connection string URI to connect to a sharded cluster instance

ApsaraDB for MongoDB allows you to use a connection string URI to connect to a sharded cluster instance to achieve load balancing and high availability.

1. Obtain the connection string URI of a sharded cluster instance. For more information, see Retrieve the seven connection elements.



2. Use the obtained connection string URI to connect your applications to the instance. For more information, see MongoDB driver connection.

   The following is an example of the Java sample code:

```
MongoClien tURI connection String = new MongoClien tURI ("
mongodb ://:****@ s - xxxxxxxx . mongodb . rds . aliyuncs . com :
3717 , s - xxxxxxxx . mongodb . rds . aliyuncs . com : 3717 / admin
"); // Replace **** with the password of the root
user .
MongoClien t client = new MongoClien t ( connection String
);
MongoDatab ase database = client . getDatabas e (" mydb ");
MongoColle ction < Document > collection = database .
getCollect ion (" mycoll ");
```

> **Note:**
>
> After you use the preceding method to connect to a sharded cluster instance, a client can automatically distribute requests to multiple mongos nodes to balance their load. In the meantime, if you have used a connection string URI to connect to two or more mongos nodes and a mongos node is faulty, the client can automatically skip this faulty node and distribute requests to the other functional mongos nodes.

If there are many mongos nodes, you can group them by application. For example, you have application A, application B, and four mongos nodes. You can specify only the connection addresses of mongos 1 and mongos 2 in the URI for application A,

and specify only the connection addresses of mongos 3 and mongos 4 in the URI for application B. In this way, you can isolate mongos nodes to provide separate entry for different applications.

 **Note:**
Although applications are connected to mutually isolated mongos nodes, they share shards at the backend.

Common connection options

- Split read and write

  Add `readPrefer ence = secondaryP referred` to options in a connection string URI to set read preference to secondary nodes of shards.

  The following is an example of a connection string URI with the read preference specified:

  ```
  mongodb :// root : xxxxxxxx @ dds - xxxxxxxxxx xx : 3717 ,
  xxxxxxxxxx xx : 3717 / admin ? replicaSet = mgset - xxxxxx &
  readPrefer ence = secondaryP referred
  ```

- Limit connections

  Add `maxPoolSiz e = xx` to options in a connection string URI to limit the maximum number of connections in the connection pool of a client to xx.

- Guarantee acknowledgement after data has been written to the majority of nodes

  Add `w = majority` to options in a connection string URI to guarantee that ApsaraDB for MongoDB sends acknowledgement to a client after writing data to the majority of nodes for a write request.

8 Troubleshoot the high CPU usage of ApsaraDB for MongoDB

When you use ApsaraDB for MongoDB, its CPU usage may become excessively high or even close to 100%. The high CPU usage not only slows down data read and write operations, but also affects normal business operations. This topic describes how to troubleshoot the high CPU usage of ApsaraDB for MongoDB for your applications.

Analyze in-progress requests in ApsaraDB for MongoDB

1. Connect to an ApsaraDB for MongoDB instance through the mongo shell.
   - Connect to a standalone instance through the mongo shell
   - Connect to a replica set instance through the mongo shell
   - Connect to a sharded cluster instance through the mongo shell
2. Run the `db . currentOp ()` command to check in-progress operations in ApsaraDB for MongoDB.

   The following is an example of the command output:

```
{
        " desc " : " conn632530 ",
        " threadId " : " 1402981969  24160 ",
        " connection  Id " :  632530 ,
        " client " : " 11 . 192 . 159 . 236 : 57052 ",
        " active " :  true ,
        " opid " :  1008837885 ,
        " secs_runni  ng " :  0 ,
        " microsecs_  running " :  NumberLong ( 70 ),
        " op " : " update ",
        " ns " : " mygame . players ",
        " query " : {
            " uid " :  NumberLong ( 31577677 )
        },
        " numYields " :  0 ,
        " locks " : {
            " Global " : " w ",
            " Database " : " w ",
            " Collection " : " w "
        },
        ....
    },
```

The following table describes the fields that you need to focus on.

Field	Description
client	The client that sent the request.

Field	Description
opid	The unique ID of the requested operation.  Note: If necessary, you can run the `db . killOp ( opid )` command to terminate an operation.
secs_running	The duration that the operation has been running, in units of seconds. If this field returns a value that exceeds the defined threshold, check whether the request is appropriate.
microsecs_running	The duration that the operation has been running, in units of microseconds. If this field returns a value that exceeds the defined threshold, check whether the request is appropriate.
ns	The target collection of the operation.
op	The operation type, which is usually query, insert, update, or delete.
locks	The lock-related fields. For more information, see the official MongoDB manual.  Note: For more information about the db.currentOp() command, see db.currentOp().

You can run the `db . currentOp ()` command to check in-progress operations and analyze whether ApsaraDB for MongoDB is processing any time-consuming requests. For example, the CPU usage is normal for your routine business. When O&M management engineers log on to ApsaraDB for MongoDB to perform some operations that require a collection scan, the CPU usage significantly increases and ApsaraDB for MongoDB responds slowly. In this case, you need to check for long-running operations.

 Note:
If you find any abnormal requests, you can obtain the operation ID (specified by the opid field) of such a request and run the `db . killOp ( opid )` command to terminate this request.

Assume that the CPU usage of the relevant ApsaraDB for MongoDB instance immediately increases and remains high after your application starts running. If

you cannot find any abnormal requests in the output of the `db . currentOp ()` command, you can analyze slow requests in ApsaraDB for MongoDB.

Analyze slow requests in ApsaraDB for MongoDB

ApsaraDB for MongoDB enables slow request profiling by default. It automatically records requested operations that have been running for longer than 100 ms in the system.profile collection of the relevant database.

1. Connect to an ApsaraDB for MongoDB instance through the mongo shell.

   For more information, see Connect to a standalone instance through the mongo shell, Connect to a replica set instance through the mongo shell, or Connect to a sharded cluster instance through the mongo shell.

2. Run the `use < database >` command to access a database.

```
use mongodbtes t
```

3. Run the following command to check the slow request logs of this database:

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

4. Analyze the slow request logs to find the cause of high CPU usage in ApsaraDB for MongoDB.

   The following is an example of a slow request log. For this request, ApsaraDB for MongoDB did not query data based on an index, but has run a collection scan and scanned 11,000,000 documents.

```
{
        " op " : " query ",
        " ns " : " 123 . testCollec tion ",
        " command " : {
                " find " : " testCollec tion ",
                " filter " : {
                        " name " : " zhangsan "
                },
                "$ db " : " 123 "
        },
        " keysExamin ed " : 0 ,
        " docsExamin ed " : 11000000 ,
        " cursorExha usted " : true ,
        " numYield " : 85977 ,
        " nreturned " : 0 ,
        " locks " : {
                " Global " : {
                        " acquireCou nt " : {
                                " r " : NumberLong ( 85978 )
                        }
                },
                " Database " : {
                        " acquireCou nt " : {
```

```
                                    " r " :  NumberLong ( 85978 )
                            }
                    },
                    " Collection " : {
                            " acquireCou  nt " : {
                                    " r " :  NumberLong ( 85978 )
                            }
                    }
            },
            " responseLe  ngth " :  232 ,
            " protocol " : " op_command ",
            " millis " :  19428 ,
            " planSummar  y " : " COLLSCAN ",
            " execStats " : {
                    " stage " : " COLLSCAN ",
                    " filter " : {
                            " name " : {
                                    "$ eq " : " zhangsan "
                            }
                    },
                    " nReturned " :  0 ,
                    " executionT  imeMillisE  stimate " :  18233 ,
                    " works " :  11000002 ,
                    " advanced " :  0 ,
                    " needTime " :  11000001 ,
                    " needYield " :  0 ,
                    " saveState " :  85977 ,
                    " restoreSta  te " :  85977 ,
                    " isEOF " :  1 ,
                    " invalidate  s " :  0 ,
                    " direction " : " forward ",
.... in "
                    }
            ],
            " user " : " root @ admin "
}
```

In slow request logs, you need to focus on the following aspects:

- Collection scan (keywords: COLLSCAN and docsExamined)
  - COLLSCAN indicates a collection scan.

    A collection scan for a request (such as a query, update, or delete operation) can occupy lots of CPU resources. If you find a COLLSCAN keyword in slow request logs, your CPU resources may have been occupied by these slow requests.

    Note:

    If such slow requests are frequently submitted, we recommend that you create an index on queried fields to optimize query performance.

  - The docsExamined field indicates the number of documents that ApsaraDB for MongoDB has scanned for a request. A larger value of this field indicates greater CPU overhead occupied by this request.

- Inappropriate index (keywords: IXSCAN and keysExamined)

  The keysExamined field indicates the number of index keys that ApsaraDB for MongoDB has scanned for a request that uses an index. A larger value of this field indicates greater CPU overhead occupied by this request.

  If you create an index that is inappropriate or matches a large amount of data, it cannot reduce CPU overhead or shorten the running duration for a request.

  For example, for the data in a collection, the x field can be set only to 1 or 2, whereas the y field has a wider value range.

```
{ x : 1 , y : 1 }
{ x : 1 , y : 2 }
{ x : 1 , y : 3 }
......
{ x : 1 , y : 100000 }
{ x : 2 , y : 1 }
{ x : 2 , y : 2 }
{ x : 2 , y : 3 }
......
{ x : 1 , y : 100000 }
```

  To query data { x: 1, y: 2 }, you can create an index. The following example shows four indexes.

```
db . createInde x ( { x : 1 } ) // This index is
inappropri ate because a large amount of data has
the same value of the x field .
db . createInde x ( { x : 1 , y : 1 } ) // This index is
  inappropri ate because a large amount of data has
  the same value of the x field .
db . createInde x ( { y : 1 } ) // This index is
appropriat e because a small amount of data has
the same value of the y field .
db . createInde x ( { y : 1 , x : 1 } ) // This index is
  appropriat e because a small amount of data has
the same value of the y field .
```

  For the difference between indexes { y: 1 } and { y: 1, x: 1 }, see Compound Indexes.

- Sorting of a large amount of data (keywords: SORT and hasSortStage)

  The value of the hasSortStage field is true in the system.profile collection when a query cannot use the sort order in the index to return the requested sorted results. In this case, ApsaraDB for MongoDB must sort the query results. Considering that a sort operation consumes plenty of CPU resources, you can create an index on frequently sorted fields to optimize sorting performance.

  Note:

If you find a SORT keyword in the system.profile collection, you can consider using an index to optimize sorting performance.

Other operations such as index creation and aggregation (a combination of traverse, query, update, sort, and other operations) may also consume significant CPU resources. You can also use the preceding troubleshooting methods. For more information about profiling, see Database Profiler.

Assess the service capability

After you analyze and optimize in-progress requests and slow requests in ApsaraDB for MongoDB, all requests efficiently use indexes and the query performance of ApsaraDB for MongoDB is optimized.

If CPU resources are still fully occupied during business operations, the service capability of your instances may have reached the upper limit. In this case, you need to view the monitoring information to analyze the resource usage of instances. You can also test your ApsaraDB for MongoDB to check whether current instances satisfy the device performance and service capability requirements in your business scenarios.

If you need to upgrade instances, you can follow the instructions in Change the configuration.