ALIBABA CLOUD

Alibaba Cloud

ApsaraDB for MongoDB

Quick Start

Document Version: 20220301

Alibaba Cloud

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 because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should 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 this document based on the "status quo", "being defective", and "existing functions" of its products and services. 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 take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, 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 contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, 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 this document 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 directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice 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.
Warning	A warning notice 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 restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	**Notice:** If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	**Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.Quick start

This topic describes how to create an ApsaraDB for MongoDB instance, configure the instance information, and connect to the instance.

Deployment architectures

ApsaraDB for MongoDB supports multiple flexible deployment architectures to meet the requirements of different business scenarios. The following section describes the architectures:

- Architecture of standalone instances

  Standalone instances apply to development, testing, education, and scenarios where non-core enterprise data is stored. You can select the instance specifications that are most suitable for your business scenarios to minimize costs. For more information, see Standalone instances.

- Architecture of replica set instances

  Replica set instances are suitable for burst traffic scenarios that require significantly more reads than writes or temporary activities. A replica set instance consists of a primary node that supports read and write operations, one, three, or five high-availability secondary nodes, a hidden node, and up to five optional read-only nodes. You can add or remove secondary nodes and read-only nodes based on your business needs. For more information, see Architecture of replica set instances.

- Architecture of sharded cluster instances

  Sharded cluster instances are suitable for scenarios that require highly concurrent read and write operations. A sharded cluster instance is based on multiple three-node replica set instances. Each replica set instance contains three nodes in primary/secondary mode and up to five optional read-only nodes. A sharded cluster instance consists of three components: mongos, shard, and Configserver nodes. You can specify the number and specifications of mongos and shard nodes to create sharded cluster instances that have different service capabilities. For more information, see Architecture of sharded cluster instances.

Procedure

The following flowchart shows all the operations that you need to perform, from purchasing an instance to using it.

1. Create an instance. The methods that you can use to create instances vary based on instance categories. For more information, see the following topics:
   - Create a standalone instance
   - Create a replica set instance
   - Create a sharded cluster instance
2. (Optional)If you have not set a password or have forgotten your password, set or reset a password. For more information, see (Optional) Reset a password.
3. Configure a whitelist. To allow specific external devices to access your instance, you must add the IP addresses of these devices to the whitelist of your instance. For more information, see Configure a whitelist for an ApsaraDB for MongoDB instance.

4. (Optional)If you want to connect to your instance by using a public endpoint, first apply for a public endpoint for the instance. For more information, see (Optional) Apply for a public endpoint for an ApsaraDB for MongoDB instance.
5. Connect to the instance. The methods that you can use to connect to instances vary based on instance categories. For more information, see the topics listed in the following table.

Instance category	References
Standalone instance	◦ Connect to a standalone ApsaraDB for MongoDB instance by using DMS (recommended) ◦ Connect to a standalone ApsaraDB for MongoDB instance by using the mongo shell ◦ Connect to an ApsaraDB for MongoDB instance by using program code
Replica set instance	◦ Connect to an ApsaraDB for MongoDB replica set instance by using DMS (recommended) ◦ Connect to a replica set instance by using the mongo shell ◦ Connect to an ApsaraDB for MongoDB instance by using the program code
Sharded cluster instance	◦ Connect to an ApsaraDB for MongoDB sharded cluster instance by using DMS (recommended) ◦ Connect to a sharded cluster ApsaraDB for MongoDB instance by using the mongo shell ◦ Connect to an ApsaraDB for MongoDB instance by using the program code

2.Limits

Before you use ApsaraDB for MongoDB, you must understand the limits of the instance architectures of Apsara for MongoDB, which helps you select the instance architectures that best suit your business scenarios.

Standalone instances

Feature	Limit
Instance deployment	Standalone instances can be created only in the following regions for single-zone deployment. You can also change the configurations of standalone instances. For more information, see Change the configurations of a standalone instance. • China (Hangzhou): Hangzhou Zone G, H, I, or J • China (Shanghai): Shanghai Zone B, D, or G • China (Qingdao): Qingdao Zone B or C • China (Beijing): Beijing Zone E, F, or H • China (Shenzhen): Shenzhen Zone E • Singapore (Singapore): Singapore Zone A, B, or C
Database version	Standalone instances are supported only for MongoDB 4.0 and MongoDB 3.4.
Database connection	Encryption features cannot be enabled or disabled for internal network access.
Data backup	Only **Snapshot Backup** is supported. **Note** A snapshot is a point-in-time backup of disk data.
Data restoration	• Only data restoration from backup sets is supported. For more information, see Restore backup data to a new ApsaraDB for MongoDB instance by backup point. • Point-in-time restoration is not supported.
Service availability	High availability features including primary/secondary switchover, role switching, or instance migration across zones are not supported.
Data security	• Only the **VPC** network type is supported. For more information about how to create a virtual private cloud (VPC), see Create and manage a VPC. • Whitelists are supported. For more information about how to configure a whitelist, see Configure a whitelist for an ApsaraDB for MongoDB standalone instance. • **Audit Logs**, **SSL**, or **TDE** is not supported.
Log management	Log management features including **Slow Query Logs**, **Error Logs**, or **Running Logs** are not supported.
CloudDBA	**Performance**, **Real-time Performance**, and **Sessions** are supported. **Storage Analysis** or **Slow Query Logs** is not supported.

Feature	Limit

Data migration and synchronization	• The following methods can be used to migrate standalone instances:
	  ◦ Migrate data from a self-managed database to an ApsaraDB for MongoDB instance
	  ▪ Migrate an ECS-hosted self-managed MongoDB database that uses the standalone or replica set architecture to ApsaraDB for MongoDB
	  ▪ Migrate a self-managed MongoDB database that uses the standalone architecture to ApsaraDB for MongoDB by using DTS
	  ▪ Migrate self-managed MongoDB databases to standalone instances by using tools provided by MongoDB
	  ◦ Migrate data between ApsaraDB for MongoDB instances
	  ▪ Migrate data from a standalone instance to a replica set or sharded cluster instance
	  ▪ Migrate data between ApsaraDB for MongoDB instances of different Alibaba Cloud accounts
	  ▪ Migrate data between ApsaraDB for MongoDB instances across regions
	• Incremental data migration or synchronization is not supported.

Replica set instances

Feature	Limit
Instance deployment	Replica set instances that run MongoDB 4.4 or 5.0 can be created only in the following regions for single-zone deployment: • China (Hangzhou): Hangzhou Zone G, H, or I • China (Shanghai): Shanghai Zone B or G • China (Qingdao): Qingdao Zone C • China (Beijing): Beijing Zone F or H • China (Zhangjiakou): Zhangjiakou Zone A or C • China (Hohhot): Hohhot Zone B • China (Shenzhen): Shenzhen Zone E • China (Heyuan): Heyuan Zone A or B • China (Chengdu): Chengdu Zone A or B • Singapore (Singapore): Singapore Zone A, B, or C • US (Silicon Valley): Silicon Valley Zone B • US (Virginia): Virginia Zone A or B
Database version	MongoDB versions and storage engines of replica set instances impose constraints on each other. For more information, see MongoDB versions and storage engines.

Feature	Limit
Replica set instance creation	• A replica set instance automatically created by ApsaraDB for MongoDB consists of a primary node, a hidden node that is invisible to users, and one or more secondary nodes. • You can change the configurations of a replica set instance when the instance is running. For more information, see Change the configurations of a replica set instance. **Note** ◦ You cannot connect to a replica set instance from a secondary node of a self-managed database. If you want to synchronize data from a replica set instance to a self-managed MongoDB database for data testing or analysis, you can use MongoShake. For more information, see Use MongoShake to implement one-way synchronization between ApsaraDB for MongoDB replica set instances. ◦ Replica set instances that run MongoDB 4.4 support only the three-node architecture and cannot contain read-only nodes.
Minor database version	ApsaraDB for MongoDB automatically updates the minor version of the instance to the latest version if the minor version of an ApsaraDB for MongoDB instance expires or is not included in the maintenance list and when you perform some operations. This ensures that the ApsaraDB for MongoDB instance can provide higher performance and stability. For more information, see Database version upgrade, Data migration, Instance configuraiton chagnes, Restore backup data to a new ApsaraDB for MongoDB instance by backup point, Restore backup data to a new ApsaraDB for MongoDB instance by point in time, and Restore one or more databases of an ApsaraDB for MongoDB instance.
Database connection	You cannot migrate replica set instances that run MongoDB 4.4 from the classic network to a VPC.
Data backup	• **Physical Backup** and **Logical Backup** are supported for replica set instances that run MongoDB 4.2 or earlier. **Note** If the database version of a replica set instance is MongoDB 3.2 or 3.4, the total number of collections and indexes in the instance cannot exceed 10,000. Otherwise, physical backup may fail. If your business involves more than 10,000 collections and indexes, you can create a replica set instance that runs MongoDB 4.0 or 4.2 or upgrade the database version of the instance to MongoDB 4.0 or 4.2. For more information about how to upgrade the database version, see Upgrade MongoDB versions. • **Snapshot Backup** is supported for replica set instances that run MongoDB 4.4. **Note** A snapshot is a point-in-time backup of disk data.

Feature	Limit
Data restoration	Data restoration is supported only for three-node replica set instances. For more information, see Restore backup data to the current instance.
Data security	• Whitelists are supported. For more information about how to configure a whitelist, see Configure a whitelist for an ApsaraDB for MongoDB standalone instance. • SSL encryption is supported. For more information, see Configure SSL encryption for an ApsaraDB for MongoDB instance. • Transparent Data Encryption (TDE) is supported only for replica set instances that run MongoDB 4.2 or earlier. For more information, see Configure TDE for an ApsaraDB for MongoDB instance. • The Audit Logs feature is supported only for replica set instances that run MongoDB 4.2 or earlier. **Note** **Audit Logs** or **TDE** is not supported for replica set instances that run MongoDB 4.4.
Log management	Log management features including **Slow Query Logs**, **Error Logs**, or **Running Logs** are not supported for replica set instances that run MongoDB 4.4.
CloudDBA	The **Slow Query Logs** feature is not supported for replica set instances that run MongoDB 4.4.

Feature	Limit
Data migration and synchronization	• The following methods can be used to migrate data of replica set instances: ◦ Migrate data from a self-managed database to an ApsaraDB for MongoDB instance ■ Migrate an ECS-hosted self-managed MongoDB database that uses the standalone or replica set architecture to ApsaraDB for MongoDB ■ Migrate a self-managed MongoDB database that uses the replica set architecture to ApsaraDB for MongoDB by using DTS ■ Migrate self-managed databases to Alibaba Cloud by using tools provided by MongoDB ◦ Migrate data from a third-party cloud database to an ApsaraDB for MongoDB instance ■ Migrate an Amazon DynamoDB database to ApsaraDB for MongoDB by using NimoShake ■ Migrate data from MongoDB Atlas to ApsaraDB for MongoDB by using mongodump and mongorestore ■ Migrate a MongoDB Atlas database to ApsaraDB for MongoDB ◦ Migrate data between ApsaraDB for MongoDB instances ■ Migrate data from a replica set instance to a sharded cluster instance ■ Migrate data between ApsaraDB for MongoDB instances of different Alibaba Cloud accounts ■ Migrate data between ApsaraDB for MongoDB instances across regions ◦ Migrate data from an ApsaraDB for MongoDB instance to a self-managed MongoDB database ■ Restore data of an ApsaraDB for MongoDB instance to self-managed MongoDB databases by using logical backup ■ Restore data of an ApsaraDB for MongoDB instance to a self-managed MongoDB database by using physical backup • The following methods can be used to synchronize data of replica set instances: ◦ Synchronize data from an ApsaraDB for MongoDB instance (replica set architecture) to another ApsaraDB for MongoDB instance (replica set architecture or sharded cluster architecture) ◦ Use MongoShake to implement one-way synchronization between ApsaraDB for MongoDB replica set instances ◦ Use MongoShake to implement delayed synchronization among ApsaraDB for MongoDB instances

Sharded cluster instances

Feature	Limit
Database version	MongoDB versions and storage engines of sharded cluster instances impose constraints on each other. For more information, see MongoDB versions and storage engines.

Feature	Limit
Sharded cluster instance creation	• When you create a sharded cluster instance, you can specify the specifications and numbers of mongos and shard nodes. • You can also add or remove mongos or shard nodes to or from a sharded cluster instance when the instance is running. For more information, see Overview. **Note** ◦ A mongos node can be released for a sharded cluster instance that has at least three mongos nodes. However, at least two mongos nodes must be retained for a sharded cluster instance. ◦ A shard node can be released for a sharded cluster instance that has at least three shard nodes. However, at least two shard nodes must be retained for a sharded cluster instance. ◦ Before you release a shard node from a sharded cluster instance, make sure that the data of the shard node can be offloaded onto the remaining shard nodes in the sharded cluster instance. Otherwise, the sharded cluster instance stays in the **Deleting Node** state. In this case, you cannot perform operations such as resetting the password, switching roles, modifying the node connection string, and modifying parameters.
Minor database version	ApsaraDB for MongoDB automatically updates the minor version of the instance to the latest version if the minor version of an ApsaraDB for MongoDB instance expires or is not included in the maintenance list and when you perform some operations. This ensures that the ApsaraDB for MongoDB instance can provide higher performance and stability. For more information, see Database version upgrade, Data migration, Instance configuraiton chagnes, Restore backup data to a new ApsaraDB for MongoDB instance by backup point, Restore backup data to a new ApsaraDB for MongoDB instance by point in time, and Restore one or more databases of an ApsaraDB for MongoDB instance.
Data read and write	You can only read data from the admin database of a sharded cluster instance, and you cannot write data to the admin database.
Data backup	**Physical Backup** and **Logical Backup** are supported for sharded cluster instances.
Data restoration	Only point-in-time data restoration is supported. For more information, see Restore backup data to a new ApsaraDB for MongoDB instance by point in time.

Feature	Limit
Data security	• Whitelists are supported. For more information about how to configure a whitelist, see Configure a whitelist for an ApsaraDB for MongoDB standalone instance. • TDE is supported. For more information, see Configure TDE for an ApsaraDB for MongoDB instance. • The Audit Logs feature is supported. • SSL encryption is not supported. **Note** Only the **VPC** network type is supported for DynamoDB-compatible sharded cluster instances. For more information about how to create a VPC, see Create and manage a VPC.
CloudDBA	CloudDBA is not supported for DynamoDB-compatible sharded cluster instances. CloudDBA features include **Performance**, **Real-time Performance**, **Sessions**, **Storage Analysis**, and **Slow Query Logs**.
Data migration and synchronization	• The following methods can be used to migrate data of sharded cluster instances: ◦ Migrate data from a self-managed database to an ApsaraDB for MongoDB instance ■ Migrate an ECS-hosted self-managed MongoDB database that uses the sharded cluster architecture to ApsaraDB for MongoDB ■ Migrate a self-managed MongoDB database that uses the sharded cluster architecture to ApsaraDB for MongoDB by using DTS ■ Migrate a self-managed MongoDB database to ApsaraDB for MongoDB by using tools provided by MongoDB ◦ Migrate data from a third-party cloud database to an ApsaraDB for MongoDB instance ■ Migrate an Amazon DynamoDB database to ApsaraDB for MongoDB by using NimoShake ■ Migrate data from MongoDB Atlas to ApsaraDB for MongoDB by using mongodump and mongorestore ■ Migrate a MongoDB Atlas database to ApsaraDB for MongoDB ◦ Migrate data between ApsaraDB for MongoDB instances Migrate data between ApsaraDB for MongoDB instances across regions • The following methods can be used to synchronize data of sharded cluster instances: Configure two-way data synchronization between ApsaraDB for MongoDB instances (sharded cluster architecture)
Recycle	The Recycle feature is not supported for sharded cluster instances.

3.Single-node quick start

3.1. Create a standalone instance

This topic describes how to create an ApsaraDB for MongoDB standalone instance. Standalone instances provide a high level of data fault tolerance and are suitable for database systems that do not store crucial data. For example, you can use standalone instances in scenarios such as development, testing, and training. This topic describes how to create a standalone instance in the ApsaraDB for MongoDB console.

Prerequisites

An Alibaba Cloud account is created. For more information, see Sign up with Alibaba Cloud.

Precautions

If your application is deployed on an Elastic Compute Service (ECS) instance, make sure that your standalone instance and the ECS instance meet the following requirements to ensure network connectivity:

- Your standalone instance and the ECS instance reside in the same region, and preferably belong to the same zone. This reduces network latency. You can view the region and zone of a created ECS instance. For more information, see View instance information.
- Your standalone instance and the ECS instance reside in **virtual private clouds (VPCs)**. You can view the network type of a created ECS instance. For more information, see View instance information. If the ECS instance resides in the **classic network**, you can migrate the ECS instance from the **classic network** to a **VPC**. For more information, see Migrate ECS instances from the classic network to a VPC.

Limits

- Standalone instances are supported only for MongoDB 4.0 and MongoDB 3.4.
- Standalone instances do not support incremental data migration and synchronization or point-in-time data restoration.
- Standalone instances support only the single-zone deployment method and can be created only in the following regions.

> **Note**
>
> - For example, **Hangzhou Zone F** supports the single-zone deployment method, and **Hangzhou Zones (B + E + F)** support the multi-zone deployment method.
> - For more information about regions and zones, see Regions and zones.

Region	Single-zone deployment
China (Hangzhou)	Hangzhou Zone G, H, I, or J
China (Shanghai)	Shanghai Zone B, D, or G
China (Qingdao)	Qingdao Zone B or C

Region	Single-zone deployment
China (Beijing)	Beijing Zone E, F, or H
China (Shenzhen)	Beijing Zone A, C, or E
Singapore (Singapore)	Singapore Zone A, B, or C

Billing

ApsaraDB for MongoDB supports the subscription and pay-as-you-go billing methods. You can select a billing method based on your business requirements.

- Subscription: If you purchase a subscription instance, you must pay an upfront fee for the instance.
- Pay-as-you-go: A pay-as-you-go instance is charged per hour based on the configurations of the instance. Fees are automatically deducted from your Alibaba Cloud account.

For more information, see Billable items and pricing.

Procedure

After you perform the following steps, ApsaraDB for MongoDB automatically creates one or more standalone instances. No manual interventions are required.

1. Log on to the ApsaraDB for MongoDB console.
2. In the upper-left corner of the page, select the resource group and the region of the target instance.
3. On the **Replica Set Instances** page, click **Create Instance**.
4. Click the **Replica Set (Subscription)** tab or the **Replica Set (Pay-as-you-go)** tab.
5. Configure the following parameters.

Parameter	Description

Parameter	Description
Region	The region where the standalone instance is deployed. For more information about the regions that are supported by standalone instances, see the Limits section of this topic. **Note** ○ After the standalone instance is created, you cannot change the region of the instance. Proceed with caution when you select a region. ○ If your application is deployed on an ECS instance, the standalone instance and the ECS instance must belong to the same region. If the standalone instance and the ECS instance belong to different regions, your application cannot communicate with the standalone instance over an internal network.
Zone	The zone where the standalone instance is deployed. For more information about the zones that are supported by standalone instances, see the Limits section of this topic. **Note** If your application is deployed on an ECS instance, we recommend that you make sure that the standalone instance and the ECS instance belong to the same zone. This reduces network latency.
Database Version	The MongoDB version of the standalone instance. Valid values: ○ **MongoDB 4.0** ○ **MongoDB 3.4**
Storage Engine	The storage engine of the standalone instance. Set the value to **WiredTiger**.
Nodes	The number of nodes in the standalone instance. Set the value to **Single Node**.
Read-only Nodes	The number of read-only nodes in the standalone instance. Set the value to **No Read-only Node**.

Parameter	Description
Network Type	The network type of the standalone instance. Set the value to **VPC**. **Note** ○ A VPC is an isolated network that provides higher security and higher performance than the classic network. ○ If your application is deployed on an ECS instance, you must make sure that the ECS instance resides within a **VPC**. This ensures network connectivity between your application and the standalone instance.
VPC	The ID of the VPC to which the standalone instance belongs. If no VPCs are available, you can create a VPC in the VPC console. For more information, see Create and manage a VPC.
vSwitch	The ID of the vSwitch to which the standalone instance belongs. If no vSwitches are available, you can create a vSwitch in the VPC console. For more information, see Work with vSwitches.
Specification	The number of cores and memory space of the standalone instance. For more information about the specifications that are supported by standalone instances, see Instance types.
Storage Space	The storage space of the standalone instance. **Note** The storage space stores the data, system, and log files of the standalone instance.
Set Password	The time at which you want to set the password of the root user. Valid values: ○ **Set Now**: You want to immediately set the password of the root user. ○ **Set Later**: You want to set the password of the root user after the standalone instance is created. For more information, see (Optional) Reset a password.
Password	The password of the root user. If you set the Set Password parameter to Set Now, you must set the password of the root user in compliance with the following rules: ○ The password must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. Special characters include !@#$%^&*()_+-= ○ The password must be 8 to 32 characters in length.

Parameter	Description
Duration	The subscription period of the standalone instance. This parameter appears only when you specify to create a subscription instance. Valid monthly subscription periods range from one month to nine months. Valid yearly subscription periods range from one year to five years. **Note** If you want to enable auto-renewal for the standalone instance, select **Auto-renewal** and make sure that you have sufficient balance within your account before the system automatically renews the instance.
Quantity	The number of standalone instances that you want to create. All the created standalone instances have the same specifications. Valid values: 1 to 10.

6. Purchase the standalone instance based on the billing method of the instance.

Billing method	Procedure
Subscription	i. Confirm the parameter settings and click **Buy Now**. ii. On the **Confirm Order** page, check the parameter settings. Then, read MongoDB Replica Set (Subscription) Agreement of Service and select I have read and agree to MongoDB Replica Set (Subscription) Agreement of Service. iii. Click **Pay**. On the **Purchase** page, complete the payment. **Note** You can also click **Add to Cart** to pay for orders in the cart in a lump sum. For more information, see Pay for orders in the cart.
Pay-as-you-go	i. Confirm the parameter settings and click **Buy Now**. ii. On the **Confirm Order** page, check the parameter settings. Then, read MongoDB Replica Set (Pay-as-you-go) Agreement of Service and select I have read and agree to MongoDB Replica Set (Pay-as-you-go) Agreement of Service. iii. Click **Activate Now**. The system collects the amount due within the next hour.

7. Check whether the standalone instance has been created.
   i. After you complete the payment, click **Console** to go to the ApsaraDB for MongoDB console.
   ii. In the top navigation bar, select the resource group and region to which the standalone instance belongs.

iii. In the instance list that appears, check whether the standalone instance that you created is displayed.

- If the standalone instance is displayed, the instance is created. If the standalone instance is in the **Running** state, the instance is running.
- If the standalone instance is not displayed, wait for 10 minutes to 15 minutes. Then, refresh the page. If the standalone instance is still not displayed, check whether the issues that are described in the following table occur. If none of the issues occur, contact Alibaba Cloud technical support.

  The following table describes the possible causes of and solutions to the issues due to which the created standalone instance is not displayed.

Possible cause	Solution
The standalone instance does not belong to the region that you select.	In the top navigation bar of the ApsaraDB for MongoDB console, select the resource group and region to which the standalone instance belongs.
The standalone instance does not belong to the architecture of instance that you select.	In the left-side navigation pane of the ApsaraDB for MongoDB console, click **Replica Set Instances**.
The standalone instance is not created due to insufficient resources.	The system may fail to create the instance due to insufficient resources. In this case, your payment is refunded. Go to the Billing Management console and open the Orders page to check whether you receive a refund. After you confirm the refund, perform one of the following operations: ■ Select a different zone and try again. ■ Submit a ticket. To submit a ticket, go to the New Ticket page.

Pay for orders in the cart

If you want to purchase a subscription standalone instance, you can also click **Add to Cart** to pay for orders in the cart in a lump sum after you complete the parameter settings. You can perform the following steps to pay for orders in the cart:

1. In the lower-right corner of the page, click the **Cart** icon.
2. In the **Cart** panel, select the products that you want to buy and click **Proceed to Checkout**.
3. On the **Cart** page, check whether the products that you want to buy are selected.

   > **Note** You can also re-specify the **Subscription Cycle** and **Quantity** parameters.

   - If the products that you want to buy are selected, click **Buy Now**.
   - If the products that you want to buy are not selected, select them and click **Buy Now**.

4. On the **Confirm Order** page, read MongoDB Replica Set (Subscription) Service Agreement, select I have read and agreed to the terms, and then click **Confirm Purchase**.
5. On the **Purchase** page, complete the payment as instructed.

Related API operations

Operation	Description
CreateDBInstance	• Creates an ApsaraDB for MongoDB standalone instance or replica set instance. • Clones an ApsaraDB for MongoDB standalone instance or replica set instance.
DescribeInstanceAutoRenewalAttribute	Queries whether auto-renewal is enabled for an ApsaraDB for MongoDB instance.
DescribeDBInstanceAttribute	Queries the detailed information of an ApsaraDB for MongoDB instance.

What's next

After the standalone instance is created, perform the following operations:

- Configure IP address whitelists for the standalone instance. The IP address whitelists of a standalone instance contain the IP addresses or CIDR blocks that are granted access to the standalone instance. For more information, see Configure a whitelist for an ApsaraDB for MongoDB standalone instance.
- (Optional)If you have not set the password of the root user for the standalone instance when you created the instance, set the password of the root user. For more information, see (Optional) Reset a password.
- (Optional)If you want to connect to the standalone instance over the Internet, apply for a public endpoint for the standalone instance. For more information, see Apply for a public endpoint for a standalone ApsaraDB for MongoDB instance.

3.2. (Optional) Reset a password

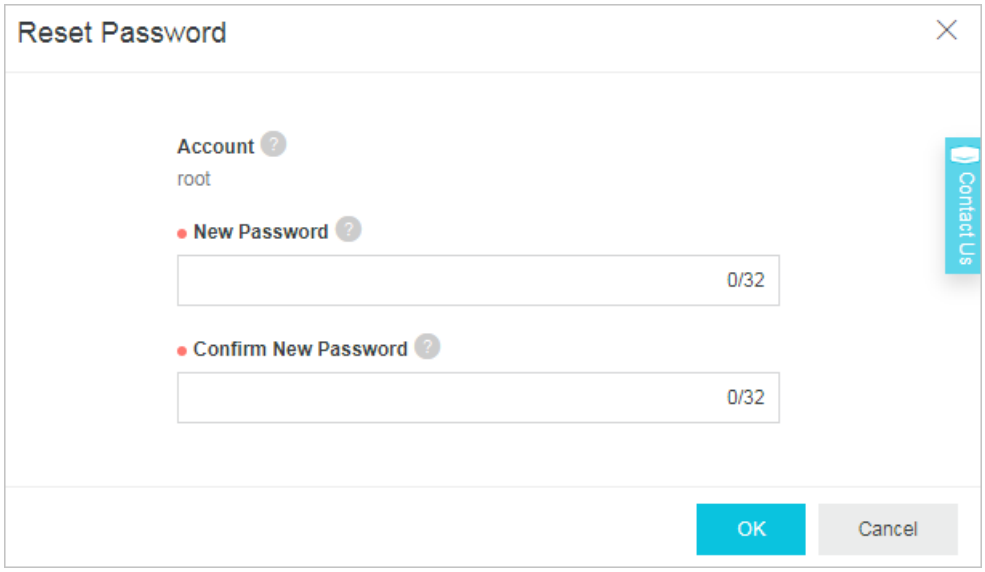
This topic describes how to set or reset a password for a standalone ApsaraDB for MongoDB instance.

Procedure

1. Log on to the ApsaraDB for MongoDB console.
2. In the upper-left corner of the page, select the resource group and region to which the instance belongs.
3. In the left-side navigation pane, click .
4. On the page that appears, find the instance that you want to manage and click its ID.
5. In the left-side navigation pane, click **Accounts**.
6. Click **Reset Password**.

7. In the **Reset Password** panel, enter a new password and confirm it. Click **OK**.

> **Note** A password must meet the following rules:
>
> - The password contains at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. Special characters include
>
> ! @ # $ % ^ & * ( ) _ + =
>
> - The password is 8 to 32 characters in length.



3.3. Configure a whitelist for an ApsaraDB for MongoDB standalone instance

This topic describes how to configure a whitelist for an ApsaraDB for MongoDB standalone instance. Only the devices whose IP addresses are added to the whitelists of the instance are allowed to access the instance. The default whitelist only contains the IP address 127.0.0.1, which indicates that no devices can connect to the instance. Proper configuration of whitelists can enhance access security of ApsaraDB for MongoDB. We recommend that you maintain your whitelists on a regular basis.

Procedure

1. Log on to the ApsaraDB for MongoDB console.
2. In the upper-left corner of the page, select the resource group and region to which the instance belongs.
3. In the left-side navigation pane, click .
4. On the page that appears, find the instance that you want to manage and click its ID.
5. In the left-side navigation pane, choose **Data Security > Whitelist Settings**.
6. In the **Add Whitelist Group** section, select one of the following methods to configure a whitelist for the instance.
   - Manually add a whitelist

> **Note**
>
> - The IP addresses can be in the following formats:
>   - A single IP address. Example: 10.23.12.24.
>   - 0.0.0.0/0
>   - One or more CIDR blocks. Example: 10.23.12.24/24. 24 indicates that the prefix of the CIDR block is 24 bits in length. You can replace 24 with a value within the range of 1 to 32.
> - Multiple IP addresses. Separate multiple IP addresses with commas (,).
> - If the value is 0.0.0.0/0 or empty, the ApsaraDB for MongoDB instance can be accessed by all IP addresses. In this situation, the database is at high security risk.

a. Click ⋮ in the **Actions** column and select **Manually Modify**.

b. In the **Manually Modify** panel, enter an IP address or CIDR block in the **IP Whitelist** text box.

c. Click **OK**.

- Load IP addresses of ECS instances

a. Click ⋮ in the **Actions** column and select **Import ECS Intranet IP**.

b. In the **IP Whitelist** of the **Import ECS Intranet IP** panel, select the ECS internal IP address to be added.

c. Click >.

d. Click **OK**.

Result

After you configure the whitelist, the endpoints of the instance appear on the **Basic Information** and **Database Connections** pages.

3.4. Apply for a public endpoint for a standalone ApsaraDB for MongoDB instance

This topic describes how to apply for a public endpoint for a standalone ApsaraDB for MongoDB instance when you want to connect to this instance over the Internet.

Context

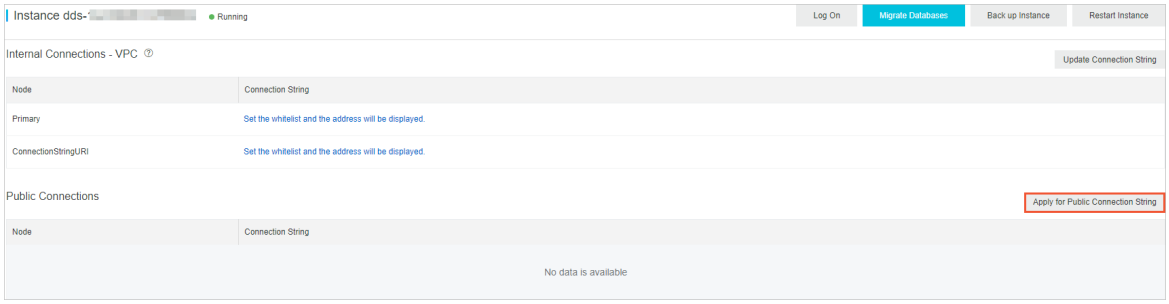
The following table describes the connections supported by a standalone ApsaraDB for MongoDB instance.

Address type	Description
VPC connection address	• A VPC is an isolated virtual network with better security and performance than a classic network. • By default, an ApsaraDB for MongoDB instance provides VPC connection addresses.
Public connection address	• By default, ApsaraDB for MongoDB instances do not provide public connection addresses because connecting to instances over the Internet poses security risks. • If you want to connect to an ApsaraDB for MongoDB instance from a device outside of Alibaba Cloud (such as a local device), you must apply for a public endpoint.

Procedure

Note To ensure data security, promptly release public connection addresses you no longer need. For more information, see Release a public connection address.

1. Log on to the ApsaraDB for MongoDB console.
2. In the upper-left corner of the page, select the resource group and region to which the instance belongs.
3. In the left-side navigation pane, click .
4. On the page that appears, find the instance that you want to manage and click its ID.
5. In the left-side navigation pane, click **Database Connections**.
6. Click **Apply for Public Connection String** on the right of **Public Connections**.



7. In the dialog box that appears, click **OK**.

Note If you want to connect to an instance by using a requested public endpoint, you must add the public IP address of the device that connects to the instance to a whitelist of the instance. For more information, see Configure a whitelist.

References

Connect to an ApsaraDB for MongoDB instance over the Internet

3.5. Connect to an instance

3.5.1. Connect to an ApsaraDB for MongoDB standalone instance by using DMS

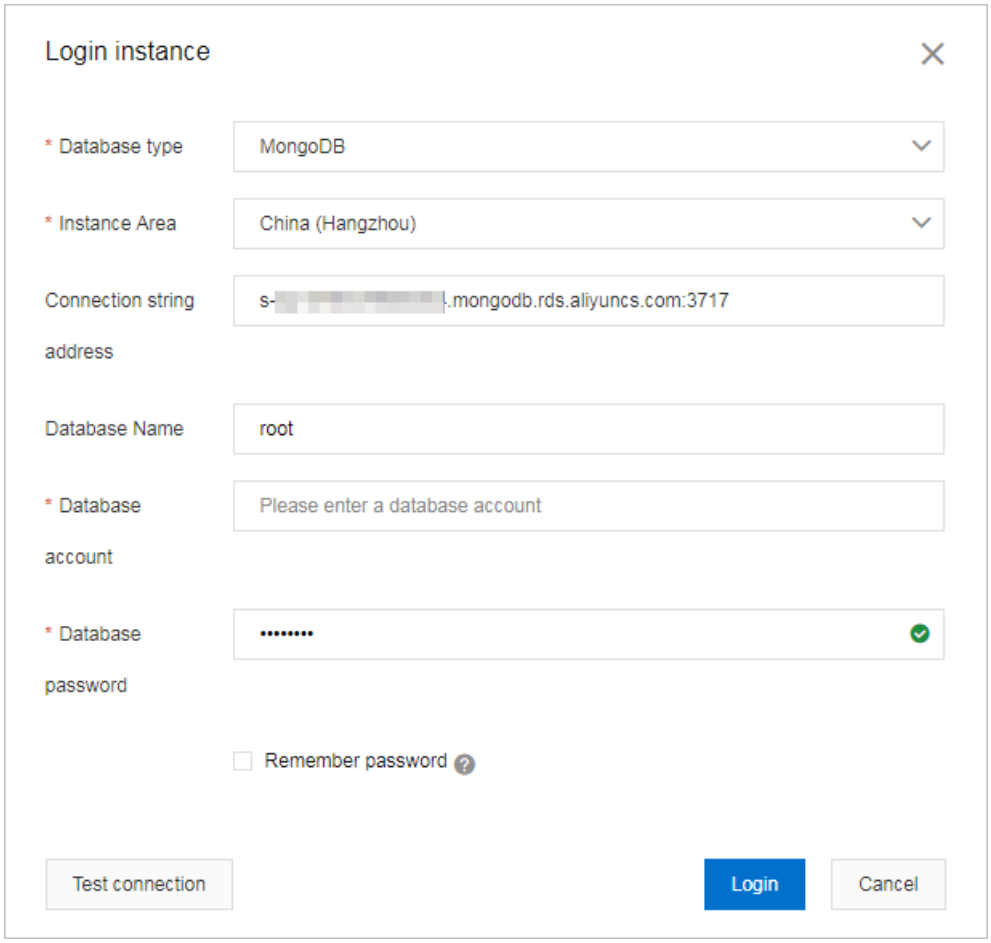
Data Management (DMS) is an integrated and visualized database solution that offers data management, structure management, user authorization, security auditing, data trend analysis, data tracking, BI charts, performance optimization, and server management. You can use DMS to connect to ApsaraDB for MongoDB instances for remote access and online management.

Preparations

Add the IP address of the DMS server (100.104.0.0/16) to a whitelist of an ApsaraDB for MongoDB instance. For more information, see Configure a whitelist for an ApsaraDB for MongoDB standalone instance.

Procedure

1. Log on to the ApsaraDB for MongoDB console.
2. In the upper-left corner of the page, select the resource group and region to which the instance belongs.
3. In the left-side navigation pane, click .
4. On the page that appears, find the instance that you want to manage and click its ID.
5. In the upper-right corner of the page, click **Log On**. You are redirected to the **Data Management** console.
6. In the **Login instance** dialog box, specify the parameters.



Parameter	Description
Database Type	The database engine of the instance. By default, this parameter is set to the database engine of the instance that you want to access.
Instance Region	The region where the instance is deployed. By default, this parameter is set to the region where the current instance is deployed.
Connection string address	The endpoint of the instance. By default, this parameter is set to the endpoint of the current instance.
Database Name	The name of the database corresponding to the account if authentication is enabled. **Note** ◦ If **Database Account** is set to root, the database name is admin. ◦ We recommend that you do not log on to a database as the root user in the production environment. You can create users and grant permissions to them based on your requirements. For more information, see Manage user permissions on MongoDB databases.

Parameter	Description
Database Account	The account that is used to access the database. The initial account is `root`.
Database password	The password of the account that is used to access the database. **Note** If you forget the password of the root account, you can reset the password by using the method described in Set a password for a standalone ApsaraDB for MongoDB instance.

7. (Optional)Click **Test Connection** to check whether the connection is successful.
   - If the **Success** message appears, the connection is successful. Click **OK**.
   - In the dialog box that appears, read the message carefully, click **OK**, and modify the configuration based on the message until the connection is successful.
8. Click **Login**.

Note If you want the browser to remember the password, select **Remember password** and then click **Login**.

Common connection scenarios

- Connect to an ApsaraDB for MongoDB instance over the Internet
- Connect an ECS instance to an ApsaraDB for MongoDB instance when their network types are different
- How to connect an ECS instance to an ApsaraDB for MongoDB instance when they are in different regions
- Connect an ECS instance with an ApsaraDB for MongoDB instance in another Alibaba Cloud account

FAQ

- How do I troubleshoot logon issues for the mongo shell?
- How to troubleshoot database connection failures after the number of connections reaches the upper limit
- How do I troubleshoot the high CPU utilization issues of ApsaraDB for MongoDB?
- How do I query and limit the number of connections?

3.5.2. Connect to a standalone ApsaraDB for MongoDB instance by using the mongo shell

This topic describes how to connect to a standalone ApsaraDB for MongoDB instance by using the mongo shell. The mongo shell is a database management tool that comes with MongoDB. You can install the mongo shell on your client or in an ECS instance.

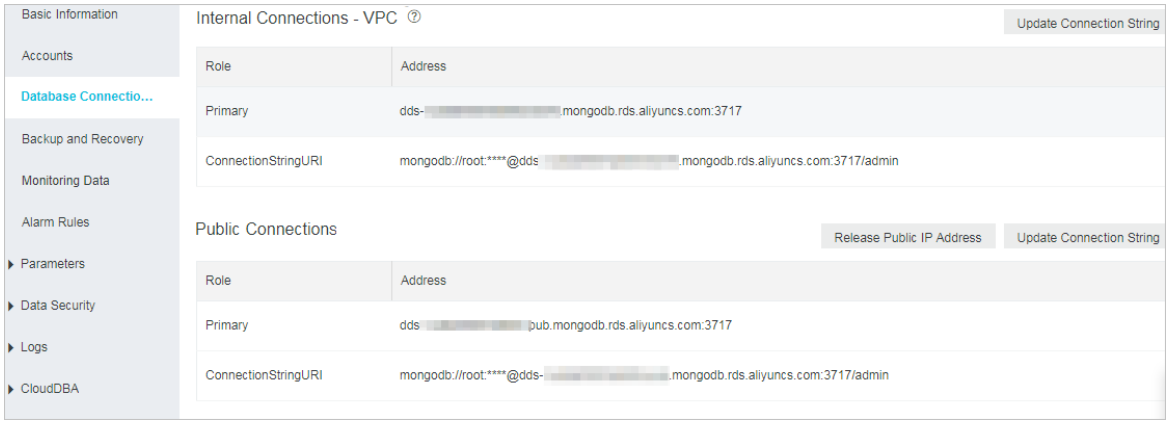
Prerequisites

- The version of the mongo shell is the same as your instance. This ensures successful authentication. For more information about the installation procedure, visit Install MongoDB. Select the correct version based on your client.
- The IP address of your client is added to a whitelist of the ApsaraDB for MongoDB instance. For more information, see Configure a whitelist for a standalone ApsaraDB for MongoDB instance.

> **Note** If you want to connect to the instance over the Internet, you must apply for a public endpoint. For more information, see Apply for a public endpoint for a standalone ApsaraDB for MongoDB instance.

Procedure

1. Log on to the ApsaraDB for MongoDB console.
2. In the upper-left corner of the page, select the resource group and region to which the instance belongs.
3. In the left-side navigation pane, click .
4. On the page that appears, find the instance that you want to manage and click its ID.
5. In the left-side navigation pane, click **Database Connections** to obtain the endpoint of the primary node.



Database connection information

Item	Description
Connection type	○ **Internal Connections**: A virtual private cloud (VPC) is an isolated virtual network that provides higher security and higher performance than the classic network. By default, ApsaraDB for MongoDB provides endpoints on a VPC. ○ **Public Connections**: By default, ApsaraDB for MongoDB instances do not provide public endpoints because connecting to instances over the Internet poses security risks. If you want to connect to a sharded cluster instance from a device outside Alibaba Cloud (for example, a local client), you must apply for a public endpoint. For more information, see Apply for a public endpoint for a standalone ApsaraDB for MongoDB instance.
Node	Primary: indicates the primary node of the ApsaraDB for MongoDB instance. You can connect to this node to perform read/write operations on the database.

Item	Description
Address	The connection string of the primary node is in the format of `<host>:<port>`. **Note** - <host>: the endpoint of the primary node. - <port>: the service port number of the primary node.
	The connection string URI is in the following format: `mongodb://[username:password@]host1[:port1][,host2[:port2],...[,hostN[:portN]]][/[database][? options]]` - `mongodb://`: the prefix, which indicates a connection string URI. - `username:password@`: the account and password used to log on to a database of the ApsaraDB for MongoDB instance. Separate the username and password with a colon (:). - `hostX:portX`: the endpoint and port number of the instance. - `/database`: the name of the authentication database to which the database account belongs. - `? options`: additional connection options.

6. Run the following command on the local server or ECS instance where the mongo shell is installed to connect to the database:

```
mongo --host <host:port> -u <username> -p --authenticationDatabase <database>
```

Note

- `<host:port>`: the endpoint of the primary node, which includes the domain name and port number.
- `<username>`: the database account of the ApsaraDB for MongoDB instance. The initial account is root. We recommend that you do not log on to a database as the root user in a production environment. You can create accounts and grant permissions to the accounts. For more information, see Manage user permissions on MongoDB databases.
- `<database>`: the name of the authentication database to which the database account belongs. If the database account is root, enter admin. If you want to specify a database other than the authentication database, run the db.createUser() command to create an account and then use the account to connect to the database.

Example:

```
mongo --host dds-bpxxxxxxxxxx.mongodb.rds.aliyuncs.com:3717 -u root -p --authentication
Database admin
```

7. When `Enter password:` is displayed, enter the password for the database account and press Enter. If you forget the password of the root user, you can reset the password. For more information, see Set a password for a standalone ApsaraDB for MongoDB instance.

   > **Note** The password you enter is not displayed.

Common connection scenarios

- Connect to an ApsaraDB for MongoDB instance over the Internet
- Connect an ECS instance to an ApsaraDB for MongoDB instance when their network types are different
- How to connect an ECS instance to an ApsaraDB for MongoDB instance when they are in different regions
- Connect an ECS instance with an ApsaraDB for MongoDB instance in another Alibaba Cloud account

Related FAQ

- How do I troubleshoot logon issues for the mongo shell?
- How to troubleshoot database connection failures after the number of connections reaches the upper limit
- How do I troubleshoot the high CPU utilization of ApsaraDB for MongoDB?
- How do I query and limit the number of connections?

3.5.3. Connect to an ApsaraDB for MongoDB instance by using program code

ApsaraDB for MongoDB is compatible with the MongoDB protocol. This topic describes the sample code used to connect to an ApsaraDB for MongoDB instance in different languages.

Preparations

- Obtain the connection strings of an ApsaraDB for MongoDB instance based on the instance type. For more information, see the following topics:
  - Connect to a replica set instance
  - Connect to a sharded cluster instance
- Download and install the official driver package of your language. For more information, visit Start Developing with MongoDB.

> **Note**
>
> - The sample code in this topic applies only to internal endpoints of ApsaraDB for MongoDB replica set instances. For other types of instances, replace the relevant content based on actual conditions.
> - To connect to sharded cluster instances, you do not need to specify the replicaSet-related parameters.

Node.js

Related link: MongoDB Node.js Driver.

1. Initialize a project.

```
 mkdir node-mongodb-demo
cd node-mongodb-demo
npm init
```

2. Install the driver package and toolkit.

```
npm install mongodb node-uuid sprintf-js -save
```

3. Obtain the information required to connect to an ApsaraDB for MongoDB instance based on the instance type.
4. Use the following Node.js sample code:

```
 'use strict';
var uuid = require('node-uuid');
var sprintf = require("sprintf-js").sprintf;
var mongoClient = require('mongodb').MongoClient;
var host1 = "**********.mongodb.tbc3.newtest.rdstest.aliyun-inc.com";
var port1 = 27017;
var host2 = "**********.mongodb.tbc3.newtest.rdstest.aliyun-inc.com";
var port2 = 27017;
var username = "demouser";
var password = "**********";
var replSetName = "mgset-**********";
var demoDb = "test";
var demoColl = "testColl";
// The officially recommended solution.
var url = sprintf("mongodb://%s:%d,%s:%d/%s?replicaSet=%s", host1, port1, host2, port2,
demoDb, replSetName);
console.info("url:", url);
// Obtain the MongoClient.
mongoClient.connect(url, function (err, db) {
    if (err) {
        console.error("connect err:", err);
        return 1;
    }
    // Authenticate the username and password used to log on to ApsaraDB for MongoDB. T
he username in this sample code is used to log on to the admin database.
    var adminDb = db.admin();
    adminDb.authenticate(username, password, function (err, result) {
        if (err) {
            console.error("authenticate err:", err);
            return 1;
        }
        // Obtain the collection handle.
        var collection = db.collection(demoColl);
        var demoName = "NODE:" + uuid.v1();
        var doc = { "DEMO": demoName, "MESG": "Hello AliCoudDB For MongoDB" };
        console.info("ready insert document: ", doc);
        // Insert data.
        collection.insertOne(doc, function (err, data) {
            if (err) {
```

```
                console.error("insert err:", err);
                return 1;
            }
            console.info("insert result:", data["result"]);
            // Read data.
            var filter = { "DEMO": demoName };
            collection.find(filter).toArray(function (err, items) {
                if (err) {
                    console.error("find err:", err);
                    return 1;
                }
                console.info("find document: ", items);
                // Close the client and release resources.
                db.close();
            });
        });
    });
});
```

PHP

Related link: MongoDB PHP Driver.

1. Install the driver package and toolkit.

```
$ pecl install mongodb
$ echo "extension=mongodb.so" >> `php --ini | grep "Loaded Configuration" | sed -e "s|.
*:\s*||"`
$ composer require "mongodb/mongodb=^1.0.0"
```

2. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
3. Use the following PHP sample code:

```
<?php
require 'vendor/autoload.php'; // include Composer goodies
# Specify instance information.
$demo_seed1 = '**********.mongodb.test.aliyun-inc.com:3717';
$demo_seed2 = '**********.mongodb.test.aliyun-inc.com:3717';
$demo_replname = "mgset-**********";
$demo_user = 'root';
$demo_password = '**********';
$demo_db = 'admin';
# Construct a MongoDB connection string URI based on the instance information.
# mongodb://[username:password@]host1[:port1][,host2[:port2],...[,hostN[:portN]]][/[dat
abase][?options]]
$demo_uri = 'mongodb://' . $demo_user . ':' . $demo_password . '@' .
    $demo_seed1 . ',' . $demo_seed2 . '/' . $demo_db . '?replicaSet=' . $demo_replname;
$client = new MongoDB\Client($demo_uri);
$collection = $client->testDb->testColl;
$result = $collection->insertOne(['name' => 'ApsaraDB for Mongodb', 'desc' => 'Hello, M
ongodb']);
echo "Inserted with Object ID '{$result->getInsertedId()}'", "\n";
$result = $collection->find(['name' => 'ApsaraDB for Mongodb']);
foreach ($result as $entry) {
    echo $entry->_id, ': ', $entry->name, "\n";
}
?>
```

Java

Related links:

- Official Getting Started.
- JAR package download.

1. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
2. Use the following Java sample code:
   - Maven configurations

```
<dependencies>
    <dependency>
        <groupId>org.mongodb</groupId>
        <artifactId>mongo-java-driver</artifactId>
        <version>4.1.0</version>
    </dependency>
</dependencies>
```

   - Java sample code

```
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import org.bson.BsonDocument;
import org.bson.BsonString;
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoClientOptions;
```

```
import com.mongodb.MongoClientURI;
import com.mongodb.MongoCredential;
import com.mongodb.ServerAddress;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
 public class Main {
        public static ServerAddress seed1 = new ServerAddress("**********.mongodb.tbc
3.newtest.rdstest.aliyun-inc.com",
                        27017);
        public static ServerAddress seed2 = new ServerAddress("**********.mongodb.tbc
3.newtest.rdstest.aliyun-inc.com",
                        27017);
        public static String username = "demouser";
        public static String password = "**********";
        public static String ReplSetName = "mgset-**********";
        public static String DEFAULT_DB = "admin";
        public static String DEMO_DB = "test";
        public static String DEMO_COLL = "testColl";
         public static MongoClient createMongoDBClient() {
                // Construct a seed list.
                List<ServerAddress> seedList = new ArrayList<ServerAddress>();
                seedList.add(seed1);
                seedList.add(seed2);
                // Construct authentication information.
                List<MongoCredential> credentials = new ArrayList<MongoCredential>();
                credentials.add(MongoCredential.createScramSha1Credential(username, D
EFAULT_DB,
                                password.toCharArray()));
                // Construct operation options. Configure options other than required
ReplicaSetName based on your actual requirements. The default parameter settings are
sufficient in most scenarios.
                MongoClientOptions options = MongoClientOptions.builder().requiredRep
licaSetName(ReplSetName)
                                .socketTimeout(2000).connectionsPerHost(1).build();
                return new MongoClient(seedList, credentials, options);
        }
         public static MongoClient createMongoDBClientWithURI() {
                // Use a URI to initialize the MongoClient.
                // mongodb://[username:password@]host1[:port1][,host2[:port2],...[,ho
stN[:portN]]][/[database][?options]]
                MongoClientURI connectionString = new MongoClientURI("mongodb://" + u
sername + ":" + password + "@"
                                + seed1 + "," + seed2 + "/" + DEFAULT_DB + "?replicaS
et=" + ReplSetName);
                return new MongoClient(connectionString);
        }
         public static void main(String args[]) {
                MongoClient client = createMongoDBClient();
                // or
                // MongoClient client = createMongoDBClientWithURI();
                try {
                        // Obtain the collection handle.
                        MongoDatabase database = client.getDatabase(DEMO_DB);
                        MongoCollection<Document> collection = database.getCollection
```

```
                        MongoCollection<Document> collection = database.getCollection
(DEMO_COLL);
                        // Insert data.
                        Document doc = new Document();
                        String demoname = "JAVA:" + UUID.randomUUID();
                        doc.append("DEMO", demoname);
                        doc.append("MESG", "Hello AliCoudDB For MongoDB");
                        collection.insertOne(doc);
                        System.out.println("insert document: " + doc);
                        // Read data.
                        BsonDocument filter = new BsonDocument();
                        filter.append("DEMO", new BsonString(demoname));
                        MongoCursor<Document> cursor = collection.find(filter).iterat
or();
                        while (cursor.hasNext()) {
                                System.out.println("find document: " + cursor.next())
;
                        }
                } finally {
                        // Close the client and release resources.
                        client.close();
                }
                return;
        }
}
```

Python

Related links:

- PyMongo download.
- Official documentation.

1. Install PyMongo.

   ```
   pip install pymongo
   ```

2. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
3. Use the following Python sample code:

```
import uuid
from pymongo import MongoClient
CONN_ADDR1 = '**********.mongodb.****.******.rdstest.aliyun-inc.com:27017'
CONN_ADDR2 = '**********.mongodb.****.******.rdstest.aliyun-inc.com:27017'
REPLICAT_SET = 'mgset-**********'
 username = 'demouser'
password = '********'
# Obtain the MongoClient.
client = MongoClient([CONN_ADDR1, CONN_ADDR2], replicaSet=REPLICAT_SET)
# Authenticate the username and password used to log on to ApsaraDB for MongoDB. The us
ername in this sample code is used to log on to the admin database.
client.admin.authenticate(username, password)
# Insert doc and search for documents based on the demo name. The collection:testColl o
f the test database is used in the example.
demo_name = 'python-' + str(uuid.uuid1())
print 'demo_name:', demo_name
 doc = dict(DEMO=demo_name, MESG="Hello ApsaraDB For MongoDB")
doc_id = client.test.testColl.insert(doc)
print 'doc_id:', doc_id
 for d in client.test.testColl.find(dict(DEMO=demo_name)):
    print 'find documents:', d
```

C#

Related link: MongoDB C# Driver.

1. Install the following driver package:

```
mongocsharpdriver.dll
```

2. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
3. Use the following C# sample code:

```
using MongoDB.Driver;
using System;
using System.Collections.Generic;
namespace Aliyun
{
    class Program
    {
        static void Main(string[] args)
        {
            // Specify instance information.
            const string host1 = "dds-t4n**************.mongodb.singapore.rds.aliyuncs.
com";
            const int port1 = 3717;
            const string host2 = "dds-t4n**************.mongodb.singapore.rds.aliyuncs.
com";
            const int port2 = 3717;
            const string replicaSetName = "mgset-300******";
            const string admin = "admin";
            const string userName = "root";
            const string passwd = "********";
            try
            {
```

```
            {
                Console.WriteLine("connecting...");
                MongoClientSettings settings = new MongoClientSettings();
                List<MongoServerAddress> servers = new List<MongoServerAddress>();
                servers.Add(new MongoServerAddress(host1, port1));
                servers.Add(new MongoServerAddress(host2, port2));
                settings.Servers = servers;
               // Set ReplicaSetName.
                settings.ReplicaSetName = replicaSetName;
                // Set ConnectTimeout to 3.
                settings.ConnectTimeout = new TimeSpan(0, 0, 0, 3, 0);
                MongoCredential credentials = MongoCredential.CreateCredential(admin, u
serName, passwd);
                settings.Credential = credentials;
                MongoClient client = new MongoClient(settings);
                var server = client.GetServer();
                MongoDatabase database = server.GetDatabase("test");
                var collection = database.GetCollection<User>("test_collection");
                User user = new User();
                user.id = "1";
                user.name = "mongo_test";
                user.sex = "female";
                // Insert data user.
                collection.Insert(user);
                // Obtain a data entry.
                User result = collection.FindOne();
                Console.WriteLine("id:" + result.id + " name:" + result.name + " sex:"+
result.sex);
                Console.WriteLine("connection successful...");
            }
            catch (Exception e)
            {
                Console.WriteLine("connection failed:"+e.Message);
            }
        }
    }
    class User
    {
        public string id { set; get; }
        public string name { set; get; }
        public string sex { set; get; }
    }
}
```

Go

Related link: MongoDB Go Driver.

1. Install the following driver package:

```
go get gopkg.in/mgo.v2
```

2. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
3. Use the following Go sample code:

```
package main
import (
"fmt"
"log"
"time""gopkg.in/mgo.v2"
"gopkg.in/mgo.v2/bson"
)
type Person struct {
Name  string
Phone string
}
func main() {
fmt.Println("hello world")
dialInfo := &mgo.DialInfo{
Addrs:     []string{"dds-bp1***********-pub.mongodb.rds.aliyuncs.com:3717", "dds-bp1***
********-pub.mongodb.rds.aliyuncs.com:3717"},
Direct:    false,
Timeout:   time.Second * 1,
Database:  "admin",
Source:    "admin",
Username:  "root",
Password:  "********",
}
session, err := mgo.DialWithInfo(dialInfo)
if err != nil {
fmt.Printf("dial failed: %v",err)
return
}
defer session.Close()
session.SetMode(mgo.Monotonic, true)
c := session.DB("test").C("test_collection")
err = c.Insert(&Person{"Ale", "+55 53 8116 9639"},
    &Person{"Cla", "+55 53 8402 8510"})
if err != nil {
    log.Fatal(err)
}
result := Person{}
err = c.Find(bson.M{"name": "Ale"}).One(&result)
if err != nil {
    log.Fatal(err)
}
fmt.Println("Phone:", result.Phone)
}
```

4.Quick Start of Replica Set

4.1. Create a replica set instance

ApsaraDB for MongoDB provides replica set instances. These instances are suitable for scenarios in which the number of read operations is larger than the number of write operations or the number of operations surges due to impromptu events. These scenarios are common in websites that provide online reading services and in systems that provide order queries. This topic describes how to create a replica set instance in the ApsaraDB for MongoDB console.

Prerequisites

An Alibaba Cloud account is created. For more information, see Sign up with Alibaba Cloud.

Precautions

If your application is deployed on an Elastic Compute Service (ECS) instance, make sure that your replica set instance and the ECS instance meet the following requirements to ensure network connectivity:

- Your replica set instance and the ECS instance reside in the same region, and preferably belong to the same zone. This reduces network latency. You can view the region of a created ECS instance. For more information, see View instance information.
- Your replica set instance and the ECS instance reside in the same type of network. You can view the network type of a created ECS instance. For more information, see View instance information. If an ECS instance resides in the **classic network**, you can migrate the ECS instance from the **classic network** to a **virtual private cloud (VPC)**. For more information, see Migrate ECS instances from the classic network to a VPC.

Limits

Replica set instances that run MongoDB 5.0 or 4.4 can be created only in the regions and zones listed in the following table.

Region	Zone
China (Hangzhou)	Hangzhou Zone G, H, or I
China (Shanghai)	Shanghai Zone B or G
China (Qingdao)	Qingdao Zone C
China (Beijing)	Beijing Zone F or H
China (Zhangjiakou)	Zhangjiakou Zone A or C
China (Hohhot)	Hohhot Zone B
China (Shenzhen)	Shenzhen Zone E
China (Heyuan)	Heyuan Zone A or B
China (Chengdu)	Chengdu Zone A or B

Region	Zone
Singapore (Singapore)	Singapore Zone A, B, or C
US (Silicon Valley)	Silicon Valley Zone B
US (Virginia)	Virginia Zone A or B

Billing

ApsaraDB for MongoDB supports the subscription and pay-as-you-go billing methods. You can select a billing method based on your business requirements.

- Subscription: If you purchase a subscription instance, you must pay an upfront fee for the instance.
- Pay-as-you-go: A pay-as-you-go instance is charged per hour based on the configurations of the instance. Fees are automatically deducted from your Alibaba Cloud account.

For more information, see Billable items and pricing.

Procedure

If your application is deployed on an Elastic Compute Service (ECS) instance, make sure that your replica set instance and the ECS instance meet the following requirements to ensure network connectivity.

1. Log on to the ApsaraDB for MongoDB console.
2. In the upper-left corner of the page, select the resource group and the region of the target instance.
3. On the **Replica Set Instances** page, click **Create Instance**.
4. Click the **Replica Set (Subscription)** tab or the **Replica Set (Pay-as-you-go)** tab.
5. Configure the following parameters.

Parameter	Description

Parameter	Description
Region	The region where the replica set instance is deployed. **Note** ◦ After the replica set instance is created, you cannot change the region of the instance. Proceed with caution. ◦ If your application is deployed on an ECS instance, the replica set instance and the ECS instance must belong to the same region. If the replica set instance and the ECS instance belong to different regions, your application cannot communicate with the replica set instance over an internal network. ◦ Replica set instances that run MongoDB 5.0 or 4.4 can be created only in some regions. For more information about these regions, see Limits.
Zone	The zone where the replica set instance is deployed. **Note** ◦ If your application is deployed on an ECS instance, we recommend that you make sure that the replica set instance and the ECS instance belong to the same zone. This reduces network latency. If the replica set instance runs MongoDB 4.2 or earlier, you can migrate the instance to a different zone. For more information, see Migrate an ApsaraDB for MongoDB instance to different zones in the same region. ◦ You can select multiple zones for the replica set instance to implement zone-disaster recovery. For more information, see Create a multi-zone replica set instance. ◦ Replica set instances that run MongoDB 5.0 or 4.4 can be created only in some zones. For more information about these zones, see Limits.

Parameter	Description
Database Version	The MongoDB version of the replica set instance. Valid values: ○ **MongoDB 5.0** ○ **MongoDB 4.4** ○ **MongoDB 4.2** ○ **MongoDB 4.0** ○ **MongoDB 3.4** **Note** ○ Replica set instances that run MongoDB 5.0 or 4.4 can be created only in some regions and zones. For more information about these regions and zones, see Limits. ○ When a replica set instance is in the Running state, you can manually upgrade the MongoDB version of the instance. For more information, see Upgrade MongoDB versions.
Storage Engine	The storage engine of the replica set instance. Set the value to **WiredTiger**.
Nodes	The number of nodes in the replica set instance. Set this parameter based on your business requirements. **Note** Set the value to **Three Nodes Replicaset** for replica set instances that run MongoDB 5.0 or 4.4.
Read-only Nodes	The number of read-only nodes in the replica set instance. Set this parameter based on your business requirements. For more information about read-only nodes, see ApsaraDB for MongoDB read-only nodes. **Note** Read-only nodes are not supported for replica set instances that run MongoDB 5.0 or 4.4.
Network Type	The network type of the standalone instance. Set the value to**VPC**. **Note** If your application is deployed on an ECS instance, you must make sure that the ECS instance resides in a **VPC**. This ensures the network connectivity between your application and the replica set instance.
VPC	The ID of the VPC to which the replica set instance belongs. If no VPCs are available, you can create a VPC in the VPC console. For more information, see Create and manage a VPC.

Parameter	Description
vSwitch	The ID of the vSwitch to which the replica set instance is connected. If no vSwitches are available, you can create a vSwitch in the VPC console. For more information, see Work with vSwitches.
Specification	The number of cores and memory space of the replica set instance. For more information about the specifications that are supported by replica set instances, see Instance types.
Storage Space	The storage space of the replica set instance. **Note** The storage space stores the data, system, and log files of the replica set instance.
Set Password	The time at which you want to set the password of the root user. Valid values: ○ **Set Now**: You want to immediately set the password of the root user. ○ **Set Later**: You want to set the password of the root user after the replica set instance is created. For more information, see Set a password for a standalone ApsaraDB for MongoDB instance.
Password	The password of the root user. If you set the Set Password parameter to Set Now, you must set the password of the root user in compliance with the following rules: ○ The password must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. Special characters include !@#$%^&*()_+-= ○ The password must be 8 to 32 characters in length.
Duration	The subscription period of the replica set instance. This parameter appears only when you specify to create a subscription replica set instance. Valid monthly subscription periods range from one month to nine months. Valid yearly subscription periods range from one year to five years. **Note** If you want to enable auto-renewal for the replica set instance, select **Auto-renewal** and make sure that you have sufficient balance within your account before the system automatically renews the instance.
Quantity	The number of replica set instances that you want to create. All the created replica set instances have the same specifications. Valid values: 1 to 10.

6. Purchase the replica set instance based on the billing method of the instance.

Billing method	Procedure

Billing method	Procedure
Subscription	i. Confirm the parameter settings and click **Buy Now**. ii. On the **Confirm Order** page, check the parameter settings. Then, read MongoDB Replica Set (Subscription) Agreement of Service and select I have read and agree to MongoDB Replica Set (Subscription) Agreement of Service. iii. Click **Pay**. On the **Purchase** page, complete the payment. **Note** You can also click **Add to Cart** to pay for orders in the cart in a lump sum. For more information, see Pay for orders in the cart.
Pay-as-you-go	i. Confirm the parameter settings and click **Buy Now**. ii. On the **Confirm Order** page, check the parameter settings. Then, read MongoDB Replica Set (Pay-as-you-go) Agreement of Service and select I have read and agree to MongoDB Replica Set (Pay-as-you-go) Agreement of Service. iii. Click **Activate Now**. The system collects the amount due within the next hour.

7. Check whether the replica set instance has been created.
   i. After you complete the payment, click **Console** to go to the ApsaraDB for MongoDB console.
   ii. In the top navigation bar, select the resource group and region to which the replica set instance belongs.
   iii. In the left-side navigation pane, click **Replica Set Instances**.
   iv. In the instance list that appears, check whether the standalone instance that you created is displayed.
      - If the standalone instance is displayed, the instance is created. If the standalone instance is in the **Running** state, the instance is running.

- If the standalone instance is not displayed, wait for 10 minutes to 15 minutes. Then, refresh the page. If the standalone instance is still not displayed, check whether the issues that are described in the following table occur. If none of the issues occur, contact Alibaba Cloud technical support.

  The following table describes the possible causes of and solutions to the issues due to which the created standalone instance is not displayed.

Possible cause	Solution
The standalone instance does not belong to the region that you select.	In the top navigation bar of the ApsaraDB for MongoDB console, select the resource group and region to which the standalone instance belongs.
The standalone instance does not belong to the architecture of instance that you select.	In the left-side navigation pane of the ApsaraDB for MongoDB console, click **Replica Set Instances**.
The standalone instance is not created due to insufficient resources.	The system may fail to create the instance due to insufficient resources. In this case, your payment is refunded. Go to the Billing Management console and open the Orders page to check whether you receive a refund. After you confirm the refund, perform one of the following operations: ■ Select a different zone and try again. ■ Submit a ticket. To submit a ticket, go to the New Ticket page.

Pay for orders in the cart

If you want to purchase a subscription replica set instance, you can also click **Add to Cart** to pay for orders in the cart in a lump sum after you complete the parameter settings. You can perform the following steps to pay for orders in the cart:

1. In the lower-right corner of the page, click the **Cart** icon.
2. In the **Cart** panel, select the products that you want to buy and click **Proceed to Checkout**.
3. On the **Cart** page, check whether the products that you want to buy are selected.

   > **Note** You can also re-specify the **Subscription Cycle** and **Quantity** parameters.

   - If the products that you want to buy are selected, click **Buy Now**.
   - If the products that you want to buy are not selected, select them and click **Buy Now**.
4. On the **Confirm Order** page, read MongoDB Replica Set (Subscription) Service Agreement, select I have read and agreed to the terms, and then click **Confirm Purchase**.
5. On the **Purchase** page, complete the payment as instructed.

Related API operations

Operation	Description
CreateDBInstance	• Creates an ApsaraDB for MongoDB standalone instance or replica set instance. • Clones an ApsaraDB for MongoDB standalone instance or replica set instance.
DescribeInstanceAutoRenewalAttribute	Queries whether auto-renewal is enabled for an ApsaraDB for MongoDB instance.
DescribeDBInstanceAttribute	Queries the detailed information of an ApsaraDB for MongoDB instance.

What's next

After the replica set instance is created, perform the following operations:

- (Optional)If you have not set the password of the root user for the replica set instance when you created the instance, set the password of the root user. For more information, see Set a password for a standalone ApsaraDB for MongoDB instance.
- Configure IP address whitelists for the replica set instance. The IP address whitelists of a replica set instance contain the IP addresses or CIDR blocks that are granted access to the replica set instance. For more information, see Configure a whitelist for an ApsaraDB for MongoDB instance.
- (Optional)If you want to connect to the replica set instance over the Internet, apply for a public endpoint for the replica set instance. For more information, see (Optional) Apply for a public endpoint for an ApsaraDB for MongoDB instance.

4.2. Set a password for a standalone ApsaraDB for MongoDB instance

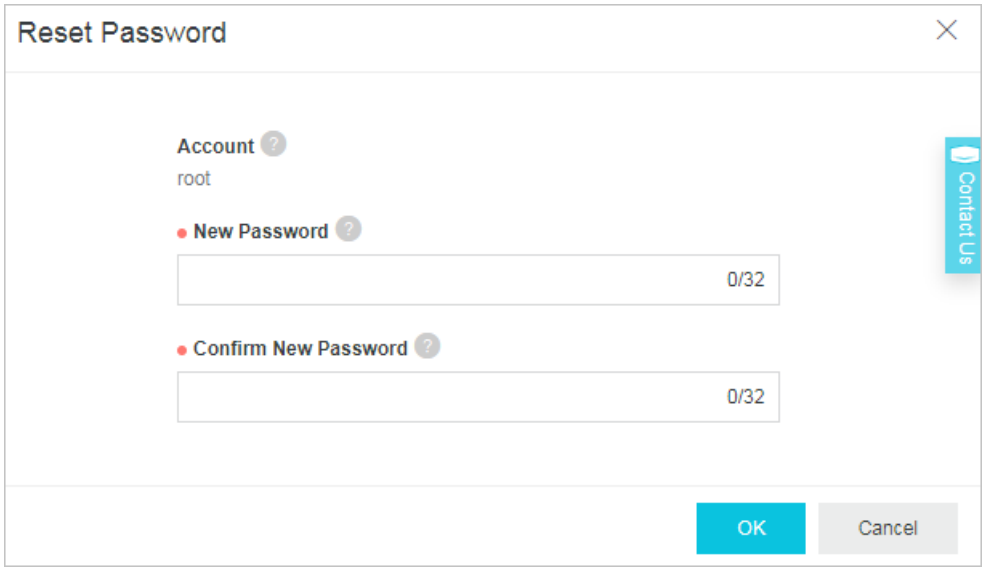
This topic describes how to set or reset a password for a standalone ApsaraDB for MongoDB instance.

Procedure

1. Log on to the ApsaraDB for MongoDB console.
2. In the upper-left corner of the page, select the resource group and region to which the instance belongs.
3. In the left-side navigation pane, click .
4. On the page that appears, find the instance that you want to manage and click its ID.
5. In the left-side navigation pane, click **Accounts**.
6. Click **Reset Password**.

7. In the **Reset Password** panel, enter a new password and confirm it. Click **OK**.

> **Note** A password must meet the following rules:
>
> - The password contains at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. Special characters include
>
> ! @ # $ % ^ & * ( ) _ + =
>
> - The password is 8 to 32 characters in length.



4.3. Configure a whitelist for an ApsaraDB for MongoDB instance

This topic describes how to configure a whitelist for an ApsaraDB for MongoDB standalone instance. Only the devices whose IP addresses are added to the whitelists of the instance are allowed to access the instance. The default whitelist only contains the IP address 127.0.0.1, which indicates that no devices can connect to the instance. Proper configuration of whitelists can enhance access security of ApsaraDB for MongoDB. We recommend that you maintain your whitelists on a regular basis.

Procedure

1. Log on to the ApsaraDB for MongoDB console.
2. In the upper-left corner of the page, select the resource group and region to which the instance belongs.
3. In the left-side navigation pane, click .
4. On the page that appears, find the instance that you want to manage and click its ID.
5. In the left-side navigation pane, choose **Data Security > Whitelist Settings**.
6. In the **Add Whitelist Group** section, select one of the following methods to configure a whitelist for the instance.
   - Manually add a whitelist

> **Note**
> - The IP addresses can be in the following formats:
>   - A single IP address. Example: 10.23.12.24.
>   - 0.0.0.0/0
>   - One or more CIDR blocks. Example: 10.23.12.24/24. 24 indicates that the prefix of the CIDR block is 24 bits in length. You can replace 24 with a value within the range of 1 to 32.
> - Multiple IP addresses. Separate multiple IP addresses with commas (,).
> - If the value is 0.0.0.0/0 or empty, the ApsaraDB for MongoDB instance can be accessed by all IP addresses. In this situation, the database is at high security risk.

a. Click ⋮ in the **Actions** column and select **Manually Modify**.

b. In the **Manually Modify** panel, enter an IP address or CIDR block in the **IP Whitelist** text box.

c. Click **OK**.

- Load IP addresses of ECS instances

a. Click ⋮ in the **Actions** column and select **Import ECS Intranet IP**.

b. In the **IP Whitelist** of the **Import ECS Intranet IP** panel, select the ECS internal IP address to be added.

c. Click >.

d. Click **OK**.

4.4. (Optional) Apply for a public endpoint for an ApsaraDB for MongoDB instance

ApsaraDB for MongoDB supports public endpoints. You can apply for a public endpoint for an ApsaraDB for MongoDB instance and use the public endpoint to connect to databases of the instance over the Internet. This topic describes how to apply for a public endpoint for an ApsaraDB for MongoDB replica set instance.

Context

The following table describes the endpoint types supported by ApsaraDB for MongoDB instances.

Endpoint type	Description
VPC endpoint	• A virtual private cloud (VPC) is an isolated network that provides higher security and performance than the classic network. • By default, ApsaraDB for MongoDB provides VPC endpoints for instances to ensure high security and high performance.

Endpoint type	Description
Classic network endpoint	Cloud services on the classic network are not isolated. Unauthorized access can be blocked only by using security groups or whitelists. You can switch the network type to VPC. For more information, see Switch the network type of an ApsaraDB for MongoDB instance from classic network to VPC. **Note** The classic network is not supported for DynamoDB-compatible sharded cluster instances.
Public endpoint	• Your ApsaraDB for MongoDB instance is at risk when you connect to it over the Internet. For this reason, ApsaraDB for MongoDB does not provide public endpoints by default. • If you want to connect to an ApsaraDB for MongoDB instance from a device outside Alibaba Cloud (such as an on-premise device), you must apply for a public endpoint.

Precautions

- When you apply for a public endpoint for an instance, the instance may need to restart. We recommend that you perform this operation during off-peak hours.
- If you want to connect to an ApsaraDB for MongoDB instance by using a public endpoint, you must add the public IP address of your client to a whitelist of this instance. For more information, see Configure a whitelist for an ApsaraDB for MongoDB instance.

Procedure

1. Log on to the ApsaraDB for MongoDB console.
2. In the upper-left corner of the page, select the resource group and region to which the instance belongs.
3. In the left-side navigation pane, click .
4. On the page that appears, find the instance that you want to manage and click its ID.
5. In the left-side navigation pane, click **Database Connections**.
6. In the **Public Connections** section, click **Apply for Public Connection String**.
7. Click **OK**.

Result

After you have applied for public endpoints, you can view the public endpoints in the **Public Connections** section. For more information about endpoints, see Connect to a replica set instance.

References

- For more information about how to connect to an instance by using a public endpoint, see Connect to an ApsaraDB for MongoDB instance over the Internet.
- To ensure data security, you can release a public endpoint that you no longer need. For more information about how to release a public endpoint, see Release a public endpoint.
- Before you connect to a replica set instance over the Internet, we recommend that you enable SSL

encryption. For more information, see Use the mongo shell to connect to an ApsaraDB for MongoDB database in SSL encryption mode.

4.5. Connect to an instance

4.5.1. Connect to an ApsaraDB for MongoDB replica set instance by using DMS

Data Management (DMS) is an integrated and visualized database solution that offers data management, structure management, user authorization, security auditing, data trend analysis, data tracking, business intelligence (BI) charts, performance optimization, and server management. You can use DMS to connect to ApsaraDB for MongoDB instances for remote access and online management.

Preparations

Add the IP addresses of the DMS server to a whitelist of an ApsaraDB for MongoDB instance based on the network type. For more information, see Configure a whitelist for an ApsaraDB for MongoDB instance.

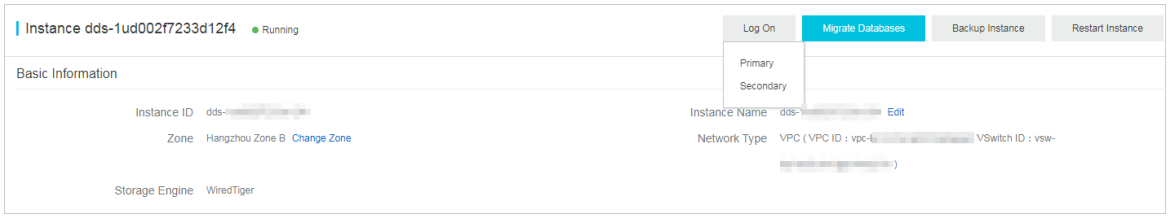
> **Note** Skip this step if you have added the IP addresses of the DMS server to a whitelist of an ApsaraDB for MongoDB instance.

IP addresses available for the DMS server

Network type of the ApsaraDB for MongoDB instance	IP address of the DMS server
Virtual Private Cloud (VPC)	100.104.0.0/16
Classic network	120.55.177.0/24 121.43.18.0/24 101.37.74.0/24 10.153.176.0/24 10.137.42.0/24 11.193.54.0/24 10.152.163.0/24

Procedure

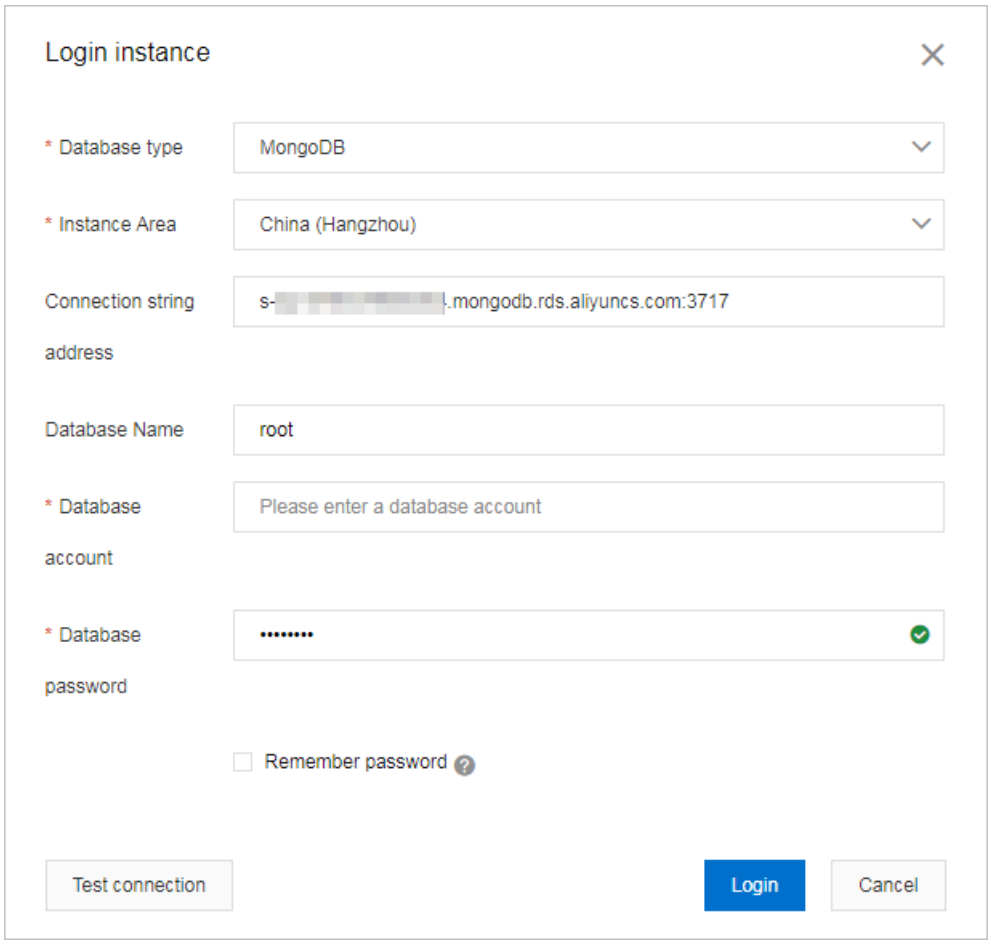
1. Log on to the ApsaraDB for MongoDB console.
2. In the upper-left corner of the page, select the resource group and region to which the instance belongs.
3. In the left-side navigation pane, click .
4. On the page that appears, find the instance that you want to manage and click its ID.
5. In the upper-right corner of the page, click **Log On** and select **Primary** or **Secondary**. Then, you are redirected to the **DMS** console.



Note

- Primary: the primary node in the replica set instance. If you connect to this node, you can perform both read and write operations on the databases of the replica set instance.
- Secondary: the secondary node in the replica set instance. If you connect to this node, you can perform only read operations on the databases of the replica set instance.

6. In the **Login instance** dialog box, specify the parameters.



Parameter	Description
Database Type	The database engine of the instance. By default, this parameter is set to the database engine of the instance that you want to access.
Instance Region	The region where the instance is deployed. By default, this parameter is set to the region where the current instance is deployed.

Parameter	Description
Connection string address	The endpoint of the instance. By default, this parameter is set to the endpoint of the current instance.
Database Name	The name of the database corresponding to the account if authentication is enabled. **Note** ◦ If **Database Account** is set to root, the database name is admin. ◦ We recommend that you do not log on to a database as the root user in the production environment. You can create users and grant permissions to them based on your requirements. For more information, see Manage user permissions on MongoDB databases.
Database Account	The account that is used to access the database. The initial account is `root`.
Database password	The password of the account that is used to access the database. **Note** If you forget the password of the root account, you can reset the password by using the method described in Set a password for a standalone ApsaraDB for MongoDB instance.

7. Click **Login**.

Note If you want your web browser to remember the password, select **Remember password** before you click **Login**.

Common connection scenarios

- Connect to an ApsaraDB for MongoDB instance over the Internet
- Connect an ECS instance to an ApsaraDB for MongoDB instance when their network types are different
- How to connect an ECS instance to an ApsaraDB for MongoDB instance when they are in different regions
- Connect an ECS instance with an ApsaraDB for MongoDB instance in another Alibaba Cloud account

Related FAQ

- How do I troubleshoot logon issues for the mongo shell?
- How to troubleshoot database connection failures after the number of connections reaches the upper limit
- How do I troubleshoot the high CPU utilization issues of ApsaraDB for MongoDB instances?
- How do I query and limit the number of connections?

4.5.2. Connect to a replica set instance by using the mongo shell

This topic describes how to use the mongo shell to connect to a replica set instance.The mongo shell is a database management tool provided by ApsaraDB for MongoDB. You can install it on your client or in an ECS instance. This topic describes how to log on to an ApsaraDB for MongoDB instance.

Prerequisites

- The version of the mongo shell is the same as that of your instance. This ensures successful authentication. For information about the installation procedure, visit MongoDB official documentation. Choose the version in the upper-left corner of the page based on your client version.
- The IP address of your client is added to a whitelist of the sharded cluster instance. For more information, see Configure a whitelist for a standalone ApsaraDB for MongoDB instance.

> **Note** If you want to connect to the instance over the Internet, you must apply for a public endpoint. For more information, see Apply for a public endpoint for a standalone ApsaraDB for MongoDB instance.

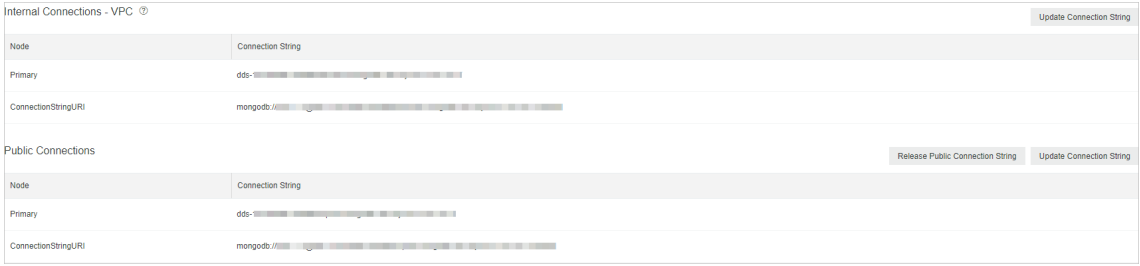
Procedure

1. Log on to the ApsaraDB for MongoDB console.
2. In the upper-left corner of the page, select the resource group and region to which the instance belongs.
3. In the left-side navigation pane, click .
4. On the page that appears, find the instance that you want to manage and click its ID.
5. In the left-side navigation pane, click **Database Connections** to obtain the connection string of a node and the connection string URI.

   > **Note** For more information about how to query the connection addresses, see Introduction to connection strings and URIs.

6. Connect to the sharded cluster instance from your client or ECS instance that has the mongo shell installed.
   - Single-node connection

     During routine tests, you can directly connect to a primary or secondarynode. A failover changes the roles of connected nodes, which affects read/write operations.

     Command syntax:

```
mongo --host <host> -u <username> -p --authenticationDatabase <database>
```

Note

- <host>: the connection string of the primary or secondary node.
  - Primary node: If you connect to this node, you can perform read/write operations on the databases of the replica set instance.
  - Secondary node: If you connect to this node, you can perform only read operations on the databases of the replica set instance.
- <username>: the username you use to log on to a database of the ApsaraDB for MongoDB instance. The default username is root. We recommend that you do not log on to a database as the root user in a production environment. You can create users and grant permissions to the users. For more information, see Manage user permissions on MongoDB databases.
- <database>: the name of the authentication database. It is the database where the database user is created. If the database username is root, enter admin. If you want to specify another database, run the db.createUser() command to create an account, and then use the account to connect to this database.

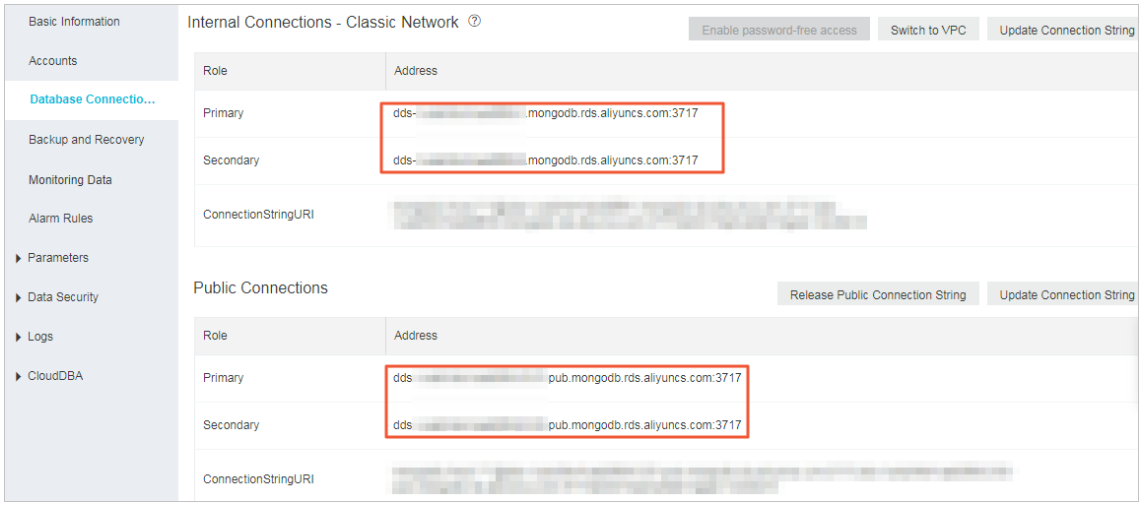
Example:

```
mongo --host dds-bp**********.mongodb.rds.aliyuncs.com:3717 -u root -p --authenticati
onDatabase admin
```

When `Enter password:` is displayed, enter the password of the database user and press the Enter key. If you forget the password of the root user, you can reset the password. For more information, see Reset the password.

Note The password characters are not displayed when you enter the password.

- HA connection (recommended): You can use a connection string URI to connect to both the primary and secondary nodes of a replica set instance. This ensures that your application is always connected to the primary node and the read/write operations of your application are not affected even if the roles of the primary and secondary nodes are switched.



Command syntax:

```
mongo "<ConnectionStringURI>"
```

> **Note**
>
> - The connection string URI must be enclosed in a pair of double quotation marks ("").
> - <ConnectionStringURI>: the connection string URI of the replica set instance.
>
> You must replace `****` in the connection string URI with the database password. For more information about how to set a database password, see Reset the password.

Common connection scenarios

- Connect to an ApsaraDB for MongoDB instance over the Internet

  > **Note** Before you connect to a replica set instance over the Internet, we recommend that you enable SSL encryption. For more information, see Use the mongo shell to connect to an ApsaraDB for MongoDB database in SSL encryption mode.

- Connect an ECS instance to an ApsaraDB for MongoDB instance when their network types are different
- How to connect an ECS instance to an ApsaraDB for MongoDB instance when they are in different regions
- Connect an ECS instance with an ApsaraDB for MongoDB instance in another Alibaba Cloud account

FAQ

- How to troubleshoot logon issues for the mongo shell
- How to troubleshoot database connection failures after the number of connections reaches the upper limit
- Troubleshoot the high CPU usage of ApsaraDB for MongoDB
- How to query and limit the number of connections

4.5.3. Connect to an ApsaraDB for MongoDB instance by using the program code

ApsaraDB for MongoDB is compatible with the MongoDB protocol. This topic describes the sample code used to connect to an ApsaraDB for MongoDB instance in different languages.

Preparations

- Obtain the connection strings of an ApsaraDB for MongoDB instance based on the instance type. For more information, see the following topics:
  - Connect to a replica set instance
  - Connect to a sharded cluster instance
- Download and install the official driver package of your language. For more information, visit Start Developing with MongoDB.

> **Note**
>
> - The sample code in this topic applies only to internal endpoints of ApsaraDB for MongoDB replica set instances. For other types of instances, replace the relevant content based on actual conditions.
> - To connect to sharded cluster instances, you do not need to specify the replicaSet-related parameters.

Node.js

Related link: MongoDB Node.js Driver.

1. Initialize a project.

```
 mkdir node-mongodb-demo
cd node-mongodb-demo
npm init
```

2. Install the driver package and toolkit.

```
npm install mongodb node-uuid sprintf-js -save
```

3. Obtain the information required to connect to an ApsaraDB for MongoDB instance based on the instance type.
4. Use the following Node.js sample code:

```
 'use strict';
var uuid = require('node-uuid');
var sprintf = require("sprintf-js").sprintf;
var mongoClient = require('mongodb').MongoClient;
var host1 = "**********.mongodb.tbc3.newtest.rdstest.aliyun-inc.com";
var port1 = 27017;
var host2 = "**********.mongodb.tbc3.newtest.rdstest.aliyun-inc.com";
var port2 = 27017;
var username = "demouser";
var password = "**********";
var replSetName = "mgset-**********";
var demoDb = "test";
var demoColl = "testColl";
// The officially recommended solution.
var url = sprintf("mongodb://%s:%d,%s:%d/%s?replicaSet=%s", host1, port1, host2, port2,
demoDb, replSetName);
console.info("url:", url);
// Obtain the MongoClient.
mongoClient.connect(url, function (err, db) {
    if (err) {
        console.error("connect err:", err);
        return 1;
    }
    // Authenticate the username and password used to log on to ApsaraDB for MongoDB. T
he username in this sample code is used to log on to the admin database.
    var adminDb = db.admin();
    adminDb.authenticate(username, password, function (err, result) {
        if (err) {
```

```
            console.error("authenticate err:", err);
            return 1;
        }
        // Obtain the collection handle.
        var collection = db.collection(demoColl);
        var demoName = "NODE:" + uuid.v1();
        var doc = { "DEMO": demoName, "MESG": "Hello AliCoudDB For MongoDB" };
        console.info("ready insert document: ", doc);
        // Insert data.
        collection.insertOne(doc, function (err, data) {
            if (err) {
                console.error("insert err:", err);
                return 1;
            }
            console.info("insert result:", data["result"]);
            // Read data.
            var filter = { "DEMO": demoName };
            collection.find(filter).toArray(function (err, items) {
                if (err) {
                    console.error("find err:", err);
                    return 1;
                }
                console.info("find document: ", items);
                // Close the client and release resources.
                db.close();
            });
        });
    });
});
```

PHP

Related link: MongoDB PHP Driver.

1. Install the driver package and toolkit.

```
$ pecl install mongodb
$ echo "extension=mongodb.so" >> `php --ini | grep "Loaded Configuration" | sed -e "s|.
*:\s*||"`
$ composer require "mongodb/mongodb=^1.0.0"
```

2. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
3. Use the following PHP sample code:

```
<?php
require 'vendor/autoload.php'; // include Composer goodies
# Specify instance information.
$demo_seed1 = '**********.mongodb.test.aliyun-inc.com:3717';
$demo_seed2 = '**********.mongodb.test.aliyun-inc.com:3717';
$demo_replname = "mgset-**********";
$demo_user = 'root';
$demo_password = '**********';
$demo_db = 'admin';
# Construct a MongoDB connection string URI based on the instance information.
# mongodb://[username:password@]host1[:port1][,host2[:port2],...[,hostN[:portN]]][/[dat
abase][?options]]
$demo_uri = 'mongodb://' . $demo_user . ':' . $demo_password . '@' .
    $demo_seed1 . ',' . $demo_seed2 . '/' . $demo_db . '?replicaSet=' . $demo_replname;
$client = new MongoDB\Client($demo_uri);
$collection = $client->testDb->testColl;
$result = $collection->insertOne(['name' => 'ApsaraDB for Mongodb', 'desc' => 'Hello, M
ongodb']);
echo "Inserted with Object ID '{$result->getInsertedId()}'", "\n";
$result = $collection->find(['name' => 'ApsaraDB for Mongodb']);
foreach ($result as $entry) {
    echo $entry->_id, ': ', $entry->name, "\n";
}
?>
```

Java

Related links:

- Official Getting Started.
- JAR package download.

1. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
2. Use the following Java sample code:
   - Maven configurations

```
<dependencies>
    <dependency>
        <groupId>org.mongodb</groupId>
        <artifactId>mongo-java-driver</artifactId>
        <version>4.1.0</version>
    </dependency>
</dependencies>
```

   - Java sample code

```
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import org.bson.BsonDocument;
import org.bson.BsonString;
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoClientOptions;
```

```
import com.mongodb.MongoClientURI;
import com.mongodb.MongoCredential;
import com.mongodb.ServerAddress;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
 public class Main {
        public static ServerAddress seed1 = new ServerAddress("**********.mongodb.tbc
3.newtest.rdstest.aliyun-inc.com",
                        27017);
        public static ServerAddress seed2 = new ServerAddress("**********.mongodb.tbc
3.newtest.rdstest.aliyun-inc.com",
                        27017);
        public static String username = "demouser";
        public static String password = "**********";
        public static String ReplSetName = "mgset-**********";
        public static String DEFAULT_DB = "admin";
        public static String DEMO_DB = "test";
        public static String DEMO_COLL = "testColl";
         public static MongoClient createMongoDBClient() {
                // Construct a seed list.
                List<ServerAddress> seedList = new ArrayList<ServerAddress>();
                seedList.add(seed1);
                seedList.add(seed2);
                // Construct authentication information.
                List<MongoCredential> credentials = new ArrayList<MongoCredential>();
                credentials.add(MongoCredential.createScramSha1Credential(username, D
EFAULT_DB,
                                password.toCharArray()));
                // Construct operation options. Configure options other than required
ReplicaSetName based on your actual requirements. The default parameter settings are
sufficient in most scenarios.
                MongoClientOptions options = MongoClientOptions.builder().requiredRep
licaSetName(ReplSetName)
                                .socketTimeout(2000).connectionsPerHost(1).build();
                return new MongoClient(seedList, credentials, options);
        }
         public static MongoClient createMongoDBClientWithURI() {
                // Use a URI to initialize the MongoClient.
                // mongodb://[username:password@]host1[:port1][,host2[:port2],...[,ho
stN[:portN]]][/[database][?options]]
                MongoClientURI connectionString = new MongoClientURI("mongodb://" + u
sername + ":" + password + "@"
                                + seed1 + "," + seed2 + "/" + DEFAULT_DB + "?replicaS
et=" + ReplSetName);
                return new MongoClient(connectionString);
        }
         public static void main(String args[]) {
                MongoClient client = createMongoDBClient();
                // or
                // MongoClient client = createMongoDBClientWithURI();
                try {
                        // Obtain the collection handle.
                        MongoDatabase database = client.getDatabase(DEMO_DB);
                        MongoCollection<Document> collection = database.getCollection
```

```
                        MongoCollection<Document> collection = database.getCollection
(DEMO_COLL);
                        // Insert data.
                        Document doc = new Document();
                        String demoname = "JAVA:" + UUID.randomUUID();
                        doc.append("DEMO", demoname);
                        doc.append("MESG", "Hello AliCoudDB For MongoDB");
                        collection.insertOne(doc);
                        System.out.println("insert document: " + doc);
                        // Read data.
                        BsonDocument filter = new BsonDocument();
                        filter.append("DEMO", new BsonString(demoname));
                        MongoCursor<Document> cursor = collection.find(filter).iterat
or();
                        while (cursor.hasNext()) {
                                System.out.println("find document: " + cursor.next())
;
                        }
                } finally {
                        // Close the client and release resources.
                        client.close();
                }
                return;
        }
}
```

Python

Related links:

- PyMongo download.
- Official documentation.

1. Install PyMongo.

   ```
   pip install pymongo
   ```

2. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
3. Use the following Python sample code:

```
import uuid
from pymongo import MongoClient
CONN_ADDR1 = '**********.mongodb.****.******.rdstest.aliyun-inc.com:27017'
CONN_ADDR2 = '**********.mongodb.****.******.rdstest.aliyun-inc.com:27017'
REPLICAT_SET = 'mgset-**********'
 username = 'demouser'
password = '********'
# Obtain the MongoClient.
client = MongoClient([CONN_ADDR1, CONN_ADDR2], replicaSet=REPLICAT_SET)
# Authenticate the username and password used to log on to ApsaraDB for MongoDB. The us
ername in this sample code is used to log on to the admin database.
client.admin.authenticate(username, password)
# Insert doc and search for documents based on the demo name. The collection:testColl o
f the test database is used in the example.
demo_name = 'python-' + str(uuid.uuid1())
print 'demo_name:', demo_name
 doc = dict(DEMO=demo_name, MESG="Hello ApsaraDB For MongoDB")
doc_id = client.test.testColl.insert(doc)
print 'doc_id:', doc_id
 for d in client.test.testColl.find(dict(DEMO=demo_name)):
    print 'find documents:', d
```

C#

Related link: MongoDB C# Driver.

1. Install the following driver package:

```
mongocsharpdriver.dll
```

2. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
3. Use the following C# sample code:

```
using MongoDB.Driver;
using System;
using System.Collections.Generic;
namespace Aliyun
{
    class Program
    {
        static void Main(string[] args)
        {
            // Specify instance information.
            const string host1 = "dds-t4n**************.mongodb.singapore.rds.aliyuncs.
com";
            const int port1 = 3717;
            const string host2 = "dds-t4n**************.mongodb.singapore.rds.aliyuncs.
com";
            const int port2 = 3717;
            const string replicaSetName = "mgset-300******";
            const string admin = "admin";
            const string userName = "root";
            const string passwd = "********";
            try
            {
```

```
            {
                Console.WriteLine("connecting...");
                MongoClientSettings settings = new MongoClientSettings();
                List<MongoServerAddress> servers = new List<MongoServerAddress>();
                servers.Add(new MongoServerAddress(host1, port1));
                servers.Add(new MongoServerAddress(host2, port2));
                settings.Servers = servers;
               // Set ReplicaSetName.
                settings.ReplicaSetName = replicaSetName;
                // Set ConnectTimeout to 3.
                settings.ConnectTimeout = new TimeSpan(0, 0, 0, 3, 0);
                MongoCredential credentials = MongoCredential.CreateCredential(admin, u
serName, passwd);
                settings.Credential = credentials;
                MongoClient client = new MongoClient(settings);
                var server = client.GetServer();
                MongoDatabase database = server.GetDatabase("test");
                var collection = database.GetCollection<User>("test_collection");
                User user = new User();
                user.id = "1";
                user.name = "mongo_test";
                user.sex = "female";
                // Insert data user.
                collection.Insert(user);
                // Obtain a data entry.
                User result = collection.FindOne();
                Console.WriteLine("id:" + result.id + " name:" + result.name + " sex:"+
result.sex);
                Console.WriteLine("connection successful...");
            }
            catch (Exception e)
            {
                Console.WriteLine("connection failed:"+e.Message);
            }
        }
    }
    class User
    {
        public string id { set; get; }
        public string name { set; get; }
        public string sex { set; get; }
    }
}
```

Go

Related link: MongoDB Go Driver.

1. Install the following driver package:

```
go get gopkg.in/mgo.v2
```

2. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
3. Use the following Go sample code:

```
package main
import (
"fmt"
"log"
"time""gopkg.in/mgo.v2"
"gopkg.in/mgo.v2/bson"
)
type Person struct {
Name  string
Phone string
}
func main() {
fmt.Println("hello world")
dialInfo := &mgo.DialInfo{
Addrs:     []string{"dds-bp1***********-pub.mongodb.rds.aliyuncs.com:3717", "dds-bp1***
********-pub.mongodb.rds.aliyuncs.com:3717"},
Direct:    false,
Timeout:   time.Second * 1,
Database:  "admin",
Source:    "admin",
Username:  "root",
Password:  "********",
}
session, err := mgo.DialWithInfo(dialInfo)
if err != nil {
fmt.Printf("dial failed: %v",err)
return
}
defer session.Close()
session.SetMode(mgo.Monotonic, true)
c := session.DB("test").C("test_collection")
err = c.Insert(&Person{"Ale", "+55 53 8116 9639"},
    &Person{"Cla", "+55 53 8402 8510"})
if err != nil {
    log.Fatal(err)
}
result := Person{}
err = c.Find(bson.M{"name": "Ale"}).One(&result)
if err != nil {
    log.Fatal(err)
}
fmt.Println("Phone:", result.Phone)
}
```

5.Quick start for sharded clusters

5.1. Create a sharded cluster instance

ApsaraDB for MongoDB sharded cluster instances are suitable for scenarios in which a large number of high concurrency read and write operations must be processed. This topic describes how to create a sharded cluster instance in the ApsaraDB for MongoDB console.

Prerequisites

An Alibaba Cloud account is created. For more information, see Sign up with Alibaba Cloud.

Precautions

If your application is deployed on an Elastic Compute Service (ECS) instance, make sure that your sharded cluster instance and the ECS instance meet the following requirements to ensure network connectivity:

- Your sharded cluster instance and the ECS instance belong to the same region. You can view the region of a created ECS instance. For more information, see View instance information.
- (Optional)Your sharded cluster instance and the ECS instance belong to the same zone. This reduces network latency. You can view the zone of a created ECS instance. For more information, see View instance information.
- Your sharded cluster instance and the ECS instance reside in the same type of network. You can view the network type of a created ECS instance. For more information, see View instance information. If an ECS instance resides in the **classic network**, you can migrate the ECS instance from the **classic network** to a **virtual private cloud (VPC)**. For more information, see Migrate ECS instances from the classic network to a VPC.

Billing

ApsaraDB for MongoDB supports the subscription and pay-as-you-go billing methods. You can select a billing method based on your business requirements.

- Subscription: If you purchase a subscription instance, you must pay an upfront fee for the instance.
- Pay-as-you-go: A pay-as-you-go instance is charged per hour based on the configurations of the instance. Fees are automatically deducted from your Alibaba Cloud account.

For more information, see Billable items and pricing.

Procedure

After you perform the following steps, ApsaraDB for MongoDB automatically creates one or more standalone instances. No manual interventions are required.

1. Log on to the ApsaraDB for MongoDB console.
2. In the upper-left corner of the page, select the resource group and the region of the target instance.
3. In the left-side navigation pane, click **Sharded Cluster Instances**.
4. On the **Sharded Cluster Instances** page, click **Create Instance**.
5. Click the **Sharded Cluster (Subscription)** or **Sharded Cluster (Pay-as-you-go)** tab based on your business requirements.

6. Configure the following parameters.

Parameter	Description
Region	The region where the sharded cluster instance is deployed. **Note** • After the sharded cluster instance is created, you cannot change the region of the instance. Proceed with caution. • If your application is deployed on an ECS instance, the sharded cluster instance and the ECS instance must belong to the same region. If the sharded cluster instance and the ECS instance belong to different regions, your application cannot communicate with the sharded cluster instance over an internal network.
Zone	The zone where the sharded cluster instance is deployed. **Note** • If your application is deployed on an ECS instance, we recommend that you make sure that the sharded cluster instance and the ECS instance belong to the same zone. This reduces network latency. If the zone that you select for the sharded cluster instance is different from the zone of the ECS instance, you can migrate the sharded cluster instance to the same zone as the ECS instance after the sharded cluster instance is created. For more information, see Migrate an ApsaraDB for MongoDB instance to different zones in the same region. • You can select multiple zones for the sharded cluster instance to implement zone-disaster recovery. For more information, see Create a multi-zone sharded cluster instance.
Protocol Type	The protocol type of the sharded cluster instance. • **MongoDB** • **DynamoDB** **Note** For more information about DynamoDB, see Compatibility details of DynamoDB-compatible ApsaraDB for MongoDB instances.

Parameter	Description
Database Version	The MongoDB version of the sharded cluster instance. Valid values: ○ **MongoDB 4.2** ○ **MongoDB 4.0** ○ **MongoDB 3.4** **Note** ○ You can select only **MongoDB 4.0** for sharded cluster instances that are compatible with the DynamoDB protocol. ○ When a sharded cluster instance is in the Running state, you can manually upgrade the MongoDB version of the instance. For more information, see Upgrade MongoDB versions.
Storage Engine	The storage engine of the sharded cluster instance. This value can be set only to **WiredTiger**.
Network Type	The type of network in which the sharded cluster instance resides. We recommend that you select **VPC**. ○ **Classic Network**: In the classic network, you can block unauthorized traffic only by configuring security groups or IP address whitelists. ○ **VPC**: A VPC is an isolated virtual network that provides higher security and higher performance than the classic network. **Note** ■ You can set the value only to **VPC** for pay-as-you-go sharded cluster instances that are compatible with the DynamoDB protocol. ■ If your application is deployed on an ECS instance, you must make sure that the ECS instance resides in a **VPC**. This ensures network connectivity between your application and the sharded cluster instance.
VPC	The ID of the VPC to which the sharded cluster instance belongs. If no VPCs are available, you can create a VPC in the VPC console. For more information, see Create and manage a VPC. **Note** ○ You can switch a sharded cluster instance to a different type of network. For more information, see Switch the network type of an ApsaraDB for MongoDB instance. ○ If you want to migrate your application to the cloud, you can connect your data center to the resources in a VPC over a leased line or a VPN to build a virtual data center. For more information, see Configure a hybrid access solution to switch the network type of an ApsaraDB for MongoDB instance from classic network to VPC.

Parameter	Description
vSwitch	The ID of the vSwitch to which the sharded cluster instance is connected. If no vSwitches are available, you can create a vSwitch in the VPC console. For more information, see Work with vSwitches.
Mongos Specification	The specifications of each mongos node in the sharded cluster instance. For more information about the specifications that are supported by mongos nodes, see Instance types. **Note** When the sharded cluster instance is in the Running state, you can change the configurations and quantity of mongos nodes.
Quantity	The number of mongos nodes in the sharded cluster instance. Valid values: 2 to 32. **Note** By default, mongos nodes use the standalone architecture. We recommend that you select two or more mongos nodes to ensure high availability.
Shard Type	The specifications of each shard node in the sharded cluster instance. For more information about the specifications that are supported by shard nodes, see Instance types. **Note** When the sharded cluster instance is in the Running state, you can change the configurations and quantity of shard nodes.
Shard Storage Capacity	The storage space of each shard node in the sharded cluster instance. **Note** The storage space stores the data, system, and log files.
readonly_replicas	The number of read-only nodes. This parameter appears only when the sharded cluster instance is charged on a subscription basis. For more information about read-only nodes, see ApsaraDB for MongoDB read-only nodes.
Quantity	The number of shard nodes in the sharded cluster instance. Valid values: 2 to 32. **Note** By default, shard nodes use the three-node replica set architecture. We recommend that you select two or more shard nodes to ensure a proper configuration of data shards. This way, the storage space and computing performance of shard nodes can be fully utilized. For more information, see Configure sharding to maximize the performance of shards.
Config Server Type	The specifications of the Configserver node in the sharded cluster instance. This value can be set only to 1 core and 2 GB memory.

Parameter	Description
Config Server	The storage space of the Configserver node in the sharded cluster instance. This value can be set only to 20 GB.
Password	The time at which you want to set the password of the root user. Valid values: ○ **Set Password**: You want to immediately set the password of the root user. ○ **Set Later**: You want to set the password of the root user after the sharded cluster instance is created. For more information, see (Optional) Reset a password.
Password	The password of the root user. If you set the Password parameter to Set Password, you must set the password of the root user in compliance with the following rules: ○ The password must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. Special characters include !@#$%^&*()_+-= ○ The password must be 8 to 32 characters in length.
Validity	The subscription period of the sharded cluster instance. This parameter appears only when the sharded cluster instance is charged on a subscription basis. Valid monthly subscription periods range from one month to nine months. Valid yearly subscription periods range from one year to five years. **Note** If you want to enable auto-renewal for the sharded cluster instance, select **Auto-renewal** and make sure that you have sufficient balance within your account.

7. Purchase the sharded cluster instance based on the billing method of the instance.

Billing method	Procedure
Subscription	i. Complete the parameter settings and click **Buy Now**. ii. On the **Confirm Order** page, check the parameter settings. Then, read MongoDB Sharded Cluster (Subscription) Agreement of Service and select I have read and agree to MongoDB Sharded Cluster (Subscription) Agreement of Service. iii. Click **Pay**. On the **Purchase** page, complete the payment. **Note** You can also click **Add to Cart** to pay for orders in the cart in a lump sum. For more information, see Pay for orders in the cart.

Billing method	Procedure
Pay-as-you-go	i. Complete the parameter settings and click **Buy Now**. ii. On the **Confirm Order** page, check the parameter settings. Then, read MongoDB Sharded Cluster (Pay-as-you-go) Agreement of Service and select I have read and agree to MongoDB Sharded Cluster (Pay-as-you-go) Agreement of Service. iii. Click **Activate Now**.The system collects the amount due within the next hour.

8. Check whether the sharded cluster instance has been created.
   i. After you complete the payment, click **Console** to go to the ApsaraDB for MongoDB console.
   ii. In the upper-left corner of the page, select the resource group and region to which the sharded cluster instance belongs.
   iii. In the left-side navigation pane, click **Sharded Cluster Instances**.
   iv. Check whether the created sharded cluster instance is in the sharded cluster instance list.
      - If the sharded cluster instance is displayed, the instance is created. If the sharded cluster instance is in the **Running** state, the instance is running as expected.

- If the sharded cluster instance is not displayed, wait for 10 to 15 minutes. Then, refresh the page. If the sharded cluster instance is still not displayed, check whether the issues that are described in the following table occur. If none of the issues occur, contact Alibaba Cloud technical support.

  The following table describes the possible causes of and solutions to the issues due to which the created sharded cluster instance is not displayed.

Possible cause	Solution
The sharded cluster instance does not belong to the region that you select.	In the upper-left corner of the ApsaraDB for MongoDB console, select the resource group and region to which the sharded cluster instance belongs.
The sharded cluster instance does not belong to the instance architecture that you select.	In the left-side navigation pane of the ApsaraDB for MongoDB console, click **Sharded Cluster Instances**.
The sharded cluster instance is not created due to insufficient resources.	The system may fail to create the instance due to insufficient resources. In this case, your payment is refunded. Go to the Billing Management console and open the Orders page to check whether you receive a refund. After you confirm the refund, perform one of the following operations: ■ Select a different zone and try again. ■ Submit a ticket. To submit a ticket, go to the New Ticket page.

Pay for orders in the cart

If you want to purchase a subscription sharded cluster instance, you can also click **Add to Cart** to pay for orders in the cart in a lump sum after you complete the parameter settings. You can perform the following steps to pay for orders in the cart:

1. In the lower-right corner of the page, click the **Cart** icon.
2. In the **Cart** panel, select the products that you want to buy and click **Proceed to Checkout**.
3. On the **Cart** page, check whether the products that you want to buy are selected.

   > **Note** You can also re-specify the **Subscription Cycle** and **Quantity** values.

   - Yes: Click **Buy Now**.
   - No: Select the products that you want to buy and click **Buy Now**.
4. On the **Confirm Order** page, read MongoDB Sharded Cluster (Subscription) Service Agreement, select I have read and agreed to the terms, and then click **Confirm Purchase**.
5. On the **Purchase** page, complete the payment as instructed.

Related operations

Operation	Description
CreateShardingDBInstance	• Creates an ApsaraDB for MongoDB sharded cluster instance. • Clones an ApsaraDB for MongoDB sharded cluster instance.
DescribeInstanceAutoRenewalAttribute	Queries whether auto-renewal is enabled for an ApsaraDB for MongoDB instance.
DescribeDBInstanceAttribute	Queries the detailed information of an ApsaraDB for MongoDB instance.

What's next

After the sharded cluster instance is created, perform the following operations:

- (Optional)If you did not set the password of the root user for the sharded cluster instance when you created the instance, set the password of the root user. For more information, see (Optional) Reset a password.
- Configure IP address whitelists for the sharded cluster instance. The IP address whitelists of a sharded cluster instance contain the IP addresses or CIDR blocks that are granted access to the sharded cluster instance. For more information, see Configure a whitelist or an ECS security group for an ApsaraDB for MongoDB instance.
- (Optional)If you want to connect to the sharded cluster instance over the Internet, apply for a public endpoint for the sharded cluster instance. For more information, see Apply for a public endpoint.

5.2. (Optional) Reset a password

If you forget your account password, want to change your password, or have not set a password when you created an instance, you can use the password reset feature provided by ApsaraDB for MongoDB.

Procedure

1. Log on to the ApsaraDB for MongoDB console.
2. In the upper-left corner of the page, select the resource group and the region of the target instance.
3. In the left-side navigation pane, click **Replica Set Instances**.
4. Find the target instance and click its ID.
5. In the left-side navigation pane, click **Accounts**.
6. Click **Reset Password** in the **Actions** column.
7. In the **Reset Password** panel, configure the following parameters.

Parameter	Description
Account	Set the value to root.

Parameter	Description
New Password	Specify the new password of the account based on the following rules: ○ The password must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. Special characters include !#$%^&*()_+= ○ The password must be 8 to 32 characters in length.
Confirm New Password	Enter the new password again.

8. Click **OK**.

5.3. Configure a whitelist for an ApsaraDB for MongoDB instance

This topic describes how to configure a whitelist for an ApsaraDB for MongoDB standalone instance. Only the devices whose IP addresses are added to the whitelists of the instance are allowed to access the instance. The default whitelist only contains the IP address 127.0.0.1, which indicates that no devices can connect to the instance. Proper configuration of whitelists can enhance access security of ApsaraDB for MongoDB. We recommend that you maintain your whitelists on a regular basis.

Procedure

1. Log on to the ApsaraDB for MongoDB console.
2. In the upper-left corner of the page, select the resource group and region to which the instance belongs.
3. In the left-side navigation pane, click .
4. On the page that appears, find the instance that you want to manage and click its ID.
5. In the left-side navigation pane, choose **Data Security > Whitelist Settings**.
6. In the **Add Whitelist Group** section, select one of the following methods to configure a whitelist for the instance.
   - Manually add a whitelist

> **Note**
>
> - The IP addresses can be in the following formats:
>   - A single IP address. Example: 10.23.12.24.
>   - 0.0.0.0/0
>   - One or more CIDR blocks. Example: 10.23.12.24/24. 24 indicates that the prefix of the CIDR block is 24 bits in length. You can replace 24 with a value within the range of 1 to 32.
> - Multiple IP addresses. Separate multiple IP addresses with commas (,).
> - If the value is 0.0.0.0/0 or empty, the ApsaraDB for MongoDB instance can be accessed by all IP addresses. In this situation, the database is at high security risk.

a. Click ⋮ in the **Actions** column and select **Manually Modify**.

b. In the **Manually Modify** panel, enter an IP address or CIDR block in the **IP Whitelist** text box.

c. Click **OK**.

- Load IP addresses of ECS instances

a. Click ⋮ in the **Actions** column and select **Import ECS Intranet IP**.

b. In the **IP Whitelist** of the **Import ECS Intranet IP** panel, select the ECS internal IP address to be added.

c. Click >.

d. Click **OK**.

5.4. (Optional) Apply for a public endpoint for an ApsaraDB for MongoDB instance

ApsaraDB for MongoDB supports public endpoints. You can apply for a public endpoint for an ApsaraDB for MongoDB instance and use the public endpoint to connect to databases of the instance over the Internet. This topic describes how to apply for a public endpoint for an ApsaraDB for MongoDB sharded cluster instance.

Context

The following table describes the endpoint types supported by ApsaraDB for MongoDB instances.

Endpoint type	Description
VPC endpoint	• A virtual private cloud (VPC) is an isolated network that provides higher security and performance than the classic network. • By default, ApsaraDB for MongoDB provides VPC endpoints for instances to ensure high security and high performance.

Endpoint type	Description
Classic network endpoint	Cloud services on the classic network are not isolated. Unauthorized access can be blocked only by using security groups or whitelists. For information about how to switch the network type to VPC, see Switch the network type of an ApsaraDB for MongoDB instance from classic network to VPC. **Note** The classic network is not supported for DynamoDB-compatible sharded cluster instances.
Public endpoint	• Security risks may arise if you connect to an ApsaraDB for MongoDB instance over the Internet. For this reason, ApsaraDB for MongoDB provides VPC endpoints by default. • If you want to connect to an ApsaraDB for MongoDB instance from a device outside of Alibaba Cloud (such as an on-premise device), you must apply for a public endpoint.

Precautions

- When you apply for a public endpoint for an instance, the instance may need to be restarted. We recommend that you perform this operation during off-peak hours.
- If you want to connect to an ApsaraDB for MongoDB instance by using a public endpoint, you must add the public IP address of your client to a whitelist of this instance. For more information, see Configure a whitelist for an ApsaraDB for MongoDB instance.

Procedure

1. Log on to the ApsaraDB for MongoDB console.
2. In the upper-left corner of the page, select the resource group and region to which the instance belongs.
3. In the left-side navigation pane, click .
4. On the page that appears, find the instance that you want to manage and click its ID.
5. In the left-side navigation pane, click **Database Connections**.
6. In the **Public Connections** section, click **Apply for Public Connection String**.
7. In the **Apply for Public Connection String** panel, set **Node Type** and **Node ID**.

Parameter	Value	Description

Parameter	Value	Description
Node Type	Shard	The shard node. To apply for a public endpoint for a shard node, you must first apply for an endpoint for the shard node. For more information, see Apply for an endpoint for a shard or Configserver node. **Note** If you want to read the oplog data of a shard node over the Internet when you perform specific operations such as data synchronization between instances, you must apply for a public endpoint for the shard node.
	CS	The Configserver node. To apply for a public endpoint for a Configserver node, you must first apply for an endpoint for the Configserver node. For more information, see Apply for an endpoint for a shard or Configserver node. **Note** If you want to read the configuration information of a Configserver node over the Internet when you perform specific operations such as data synchronization between instances, you must apply for a public endpoint for the Configserver node.
	Mongos	The mongos node. **Note** In most cases, mongos nodes are sufficient to meet your read and write needs.
Node ID	Node ID of the current instance	The ID of the node for which you want to apply for a public endpoint.

8. Click **OK**.
9. (Optional)To apply for public endpoints for multiple nodes in the sharded cluster instance, repeat the preceding steps.

Note To apply for a public endpoint for another node in the instance, you must wait until the state of the instance becomes **Running**.

Results

After you apply for public endpoints, you can view the created endpoints in the **Public Connections** section. For more information about endpoints, see Introduction to connection strings and URIs.

References

- For more information about how to connect to an instance by using a public endpoint, see Connect

to an ApsaraDB for MongoDB instance over the Internet.

- To ensure data security, you can release public endpoints that are no longer needed. For more information about how to release a public endpoint, see Release a public endpoint.
- Before you connect to an ApsaraDB for MongoDB instance over the Internet, we recommend that you enable SSL encryption. For more information about how to enable SSL encryption, see Use the mongo shell to connect to an ApsaraDB for MongoDB database in SSL encryption mode.

5.5. Connect to an instance

5.5.1. Connect to an ApsaraDB for MongoDB sharded cluster instance by using DMS

Data Management (DMS) is an integrated and visualized database solution that offers data management, structure management, user authorization, security auditing, data trend analysis, data tracking, business intelligence (BI) charts, performance optimization, and server management. You can use DMS to connect to ApsaraDB for MongoDB instances for remote access and online management.

Preparations

Add the IP addresses of the DMS server to a whitelist of an ApsaraDB for MongoDB instance based on the network type. For more information, see Configure a whitelist for a sharded cluster instance.

> **Note** Skip this step if you have added the IP addresses of the DMS server to a whitelist of an ApsaraDB for MongoDB instance.

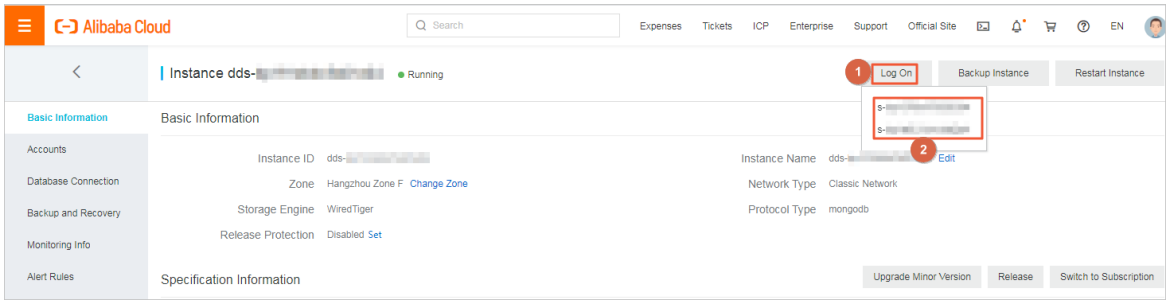
IP addresses available for the DMS server

Network type of the ApsaraDB for MongoDB instance	IP address of the DMS server
Virtual Private Cloud (VPC)	100.104.0.0/16
Classic network	120.55.177.0/24 121.43.18.0/24 101.37.74.0/24 10.153.176.0/24 10.137.42.0/24 11.193.54.0/24

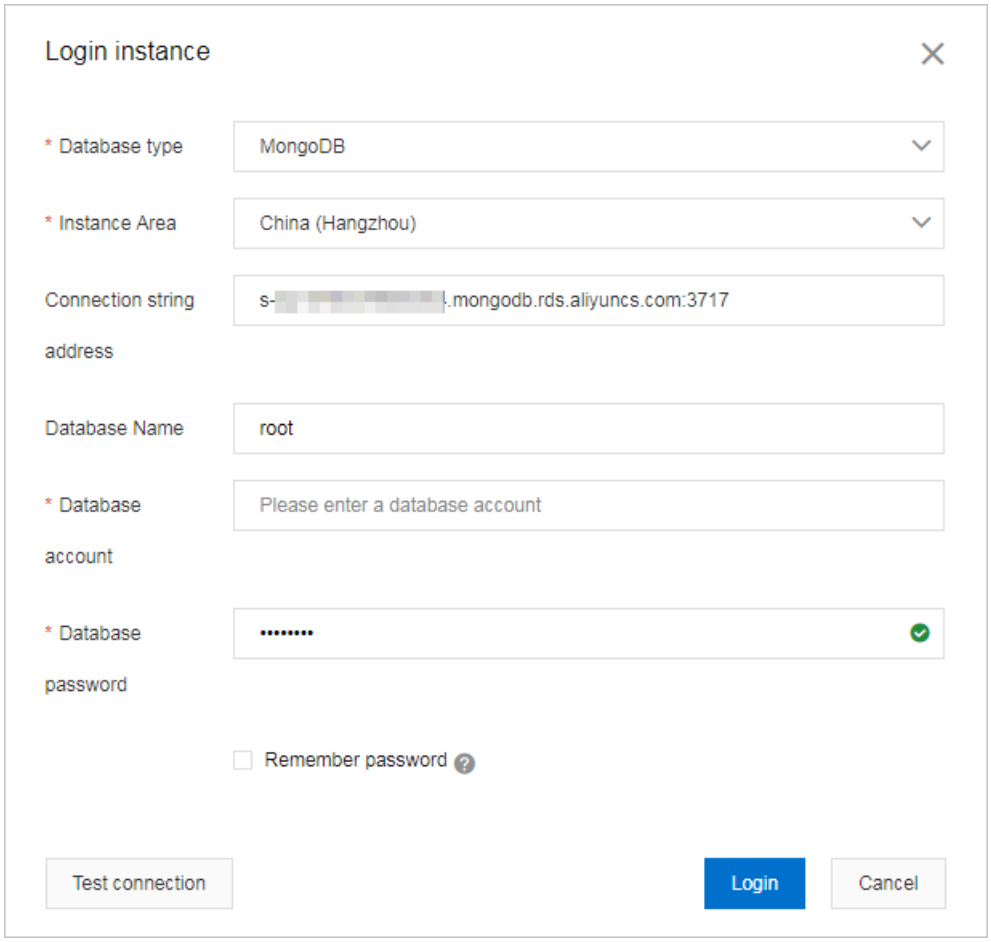
Procedure

1. Log on to the ApsaraDB for MongoDB console.
2. In the upper-left corner of the page, select the resource group and region to which the instance belongs.
3. In the left-side navigation pane, click .
4. On the page that appears, find the instance that you want to manage and click its ID.

5. Click **Log On** and select a mongos node ID. You are redirected to the **DMS** console.



6. In the **Login instance** dialog box, specify the parameters.



Parameter	Description
Database Type	The database engine of the instance. By default, this parameter is set to the database engine of the instance that you want to access.
Instance Region	The region where the instance is deployed. By default, this parameter is set to the region where the current instance is deployed.
Connection string address	The endpoint of the instance. By default, this parameter is set to the endpoint of the current instance.

Parameter	Description
Database Name	The name of the database corresponding to the account if authentication is enabled. **Note** ◦ If **Database Account** is set to root, the database name is admin. ◦ We recommend that you do not log on to a database as the root user in the production environment. You can create users and grant permissions to them based on your requirements. For more information, see Manage user permissions on MongoDB databases.
Database Account	The account that is used to access the database. The initial account is `root`.
Database password	The password of the account that is used to access the database. **Note** If you forget the password of the root account, you can reset the password by using the method described in Set a password for a standalone ApsaraDB for MongoDB instance.

7. Click **Login**.

Note If you want your web browser to remember the password, select **Remember password** before you click **Login**.

Common connection scenarios

- Connect to an ApsaraDB for MongoDB instance over the Internet
- Connect an ECS instance to an ApsaraDB for MongoDB instance when their network types are different
- How to connect an ECS instance to an ApsaraDB for MongoDB instance when they are in different regions
- Connect an ECS instance with an ApsaraDB for MongoDB instance in another Alibaba Cloud account

Related FAQ

- How do I troubleshoot logon issues for the mongo shell?
- How to troubleshoot database connection failures after the number of connections reaches the upper limit
- How do I troubleshoot the high CPU utilization issues of ApsaraDB for MongoDB?
- How do I query and limit the number of connections?

5.5.2. Connect to a sharded cluster ApsaraDB for MongoDB instance by using the mongo shell

This topic describes how to connect to a sharded cluster instance by using the mongo shell. The mongo shell is a database management tool that comes with MongoDB. You can install the mongo shell on your client or in an ECS instance.

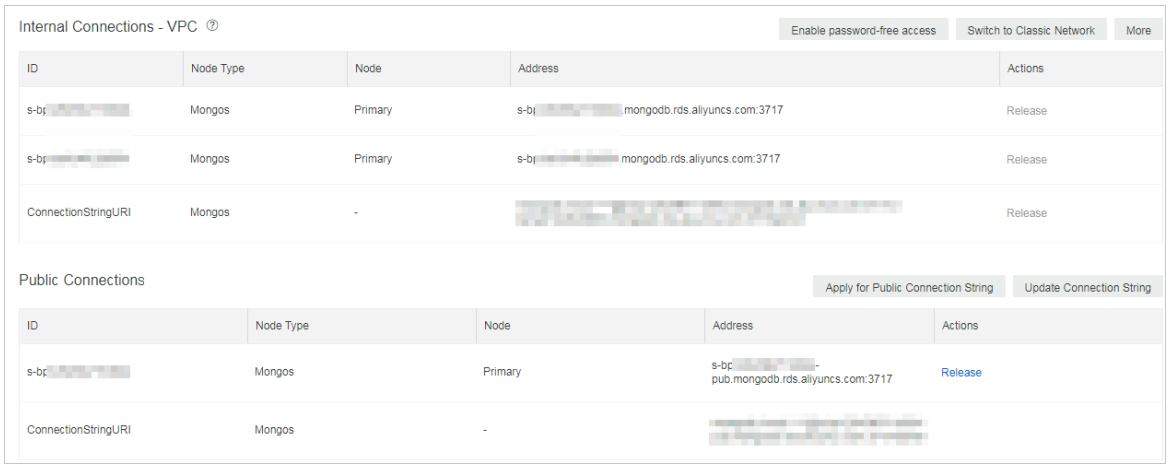
Prerequisites

- The version of the mongo shell is the same as your instance. This ensures successful authentication. For more information about the installation procedure, visit Install MongoDB. Select the correct version based on your client.
- The IP address of your client is added to a whitelist of the ApsaraDB for MongoDB instance. For more information, see Configure a whitelist for a standalone ApsaraDB for MongoDB instance.

> **Note** If you want to connect to the instance over the Internet, you must apply for a public endpoint. For more information, see Apply for a public endpoint for a sharded cluster instance.

Procedure

1. Log on to the ApsaraDB for MongoDB console.
2. In the upper-left corner of the page, select the resource group and region to which the instance belongs.
3. In the left-side navigation pane, click .
4. On the page that appears, find the instance that you want to manage and click its ID.
5. Click **Database Connections** to obtain the endpoint of a mongos.



6. Connect to the sharded cluster instance from your client or ECS instance that has the mongo shell installed.

```
mongo --host <mongos_host> -u <username> -p --authenticationDatabase <database>
```

> **Note**
>
> - <mongos_host>: the endpoint of a mongos in the sharded cluster instance.
> - <username>: the database account of the ApsaraDB for MongoDB instance. The initial account is root. We recommend that you do not log on to a database as the root account in a production environment. You can create accounts and grant permissions to the accounts. For more information, see Manage user permissions on MongoDB databases.
> - <database>: the name of the authentication database to which the database account belongs. If the database account is root, enter admin. If you want to specify a database other than the authentication database, run the db.createUser() command to create an account and then use the account to connect to the database.

Example:

```
mongo --host s-bp**********.mongodb.rds.aliyuncs.com:3717 -u root -p --authenticationDa
tabase admin
```

7. When `Enter password:` is displayed, enter the password of the database account and press Enter. If you forget the password of the root account, you can reset the password. For more information, see Set a password for a standalone ApsaraDB for MongoDB instance.

> **Note** The password you enter is not displayed.

Common connection scenarios

- Connect to an ApsaraDB for MongoDB instance over the Internet
- Connect an ECS instance to an ApsaraDB for MongoDB instance when their network types are different
- How to connect an ECS instance to an ApsaraDB for MongoDB instance when they are in different regions
- Connect an ECS instance with an ApsaraDB for MongoDB instance in another Alibaba Cloud account

Related FAQ

- How do I troubleshoot logon issues for the mongo shell?
- How to troubleshoot database connection failures after the number of connections reaches the upper limit
- How do I troubleshoot the high CPU utilization of ApsaraDB for MongoDB?
- How do I query and limit the number of connections?

5.5.3. Connect to an ApsaraDB for MongoDB instance by using the program code

ApsaraDB for MongoDB is compatible with the MongoDB protocol. This topic describes the sample code used to connect to an ApsaraDB for MongoDB instance in different languages.

Preparations

- Obtain the connection strings of an ApsaraDB for MongoDB instance based on the instance type. For more information, see the following topics:
  - Connect to a replica set instance
  - Connect to a sharded cluster instance
- Download and install the official driver package of your language. For more information, visit Start Developing with MongoDB.

> **Note**
> - The sample code in this topic applies only to internal endpoints of ApsaraDB for MongoDB replica set instances. For other types of instances, replace the relevant content based on actual conditions.
> - To connect to sharded cluster instances, you do not need to specify the replicaSet-related parameters.

Node.js

Related link: MongoDB Node.js Driver.

1. Initialize a project.

```
 mkdir node-mongodb-demo
cd node-mongodb-demo
npm init
```

2. Install the driver package and toolkit.

```
npm install mongodb node-uuid sprintf-js -save
```

3. Obtain the information required to connect to an ApsaraDB for MongoDB instance based on the instance type.
4. Use the following Node.js sample code:

```
 'use strict';
var uuid = require('node-uuid');
var sprintf = require("sprintf-js").sprintf;
var mongoClient = require('mongodb').MongoClient;
var host1 = "**********.mongodb.tbc3.newtest.rdstest.aliyun-inc.com";
var port1 = 27017;
var host2 = "**********.mongodb.tbc3.newtest.rdstest.aliyun-inc.com";
var port2 = 27017;
var username = "demouser";
var password = "**********";
var replSetName = "mgset-**********";
var demoDb = "test";
var demoColl = "testColl";
// The officially recommended solution.
var url = sprintf("mongodb://%s:%d,%s:%d/%s?replicaSet=%s", host1, port1, host2, port2,
demoDb, replSetName);
console.info("url:", url);
// Obtain the MongoClient.
mongoClient.connect(url, function (err, db) {
    if (err) {
```

```
        console.error("connect err:", err);
        return 1;
    }
    // Authenticate the username and password used to log on to ApsaraDB for MongoDB. T
he username in this sample code is used to log on to the admin database.
    var adminDb = db.admin();
    adminDb.authenticate(username, password, function (err, result) {
        if (err) {
            console.error("authenticate err:", err);
            return 1;
        }
        // Obtain the collection handle.
        var collection = db.collection(demoColl);
        var demoName = "NODE:" + uuid.v1();
        var doc = { "DEMO": demoName, "MESG": "Hello AliCoudDB For MongoDB" };
        console.info("ready insert document: ", doc);
        // Insert data.
        collection.insertOne(doc, function (err, data) {
            if (err) {
                console.error("insert err:", err);
                return 1;
            }
            console.info("insert result:", data["result"]);
            // Read data.
            var filter = { "DEMO": demoName };
            collection.find(filter).toArray(function (err, items) {
                if (err) {
                    console.error("find err:", err);
                    return 1;
                }
                console.info("find document: ", items);
                // Close the client and release resources.
                db.close();
            });
        });
    });
});
```

PHP

Related link: MongoDB PHP Driver.

1. Install the driver package and toolkit.

```
$ pecl install mongodb
$ echo "extension=mongodb.so" >> `php --ini | grep "Loaded Configuration" | sed -e "s|.
*:\s*||"`
$ composer require "mongodb/mongodb=^1.0.0"
```

2. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
3. Use the following PHP sample code:

```
<?php
require 'vendor/autoload.php'; // include Composer goodies
# Specify instance information.
$demo_seed1 = '**********.mongodb.test.aliyun-inc.com:3717';
$demo_seed2 = '**********.mongodb.test.aliyun-inc.com:3717';
$demo_replname = "mgset-**********";
$demo_user = 'root';
$demo_password = '**********';
$demo_db = 'admin';
# Construct a MongoDB connection string URI based on the instance information.
# mongodb://[username:password@]host1[:port1][,host2[:port2],...[,hostN[:portN]]][/[dat
abase][?options]]
$demo_uri = 'mongodb://' . $demo_user . ':' . $demo_password . '@' .
    $demo_seed1 . ',' . $demo_seed2 . '/' . $demo_db . '?replicaSet=' . $demo_replname;
$client = new MongoDB\Client($demo_uri);
$collection = $client->testDb->testColl;
$result = $collection->insertOne(['name' => 'ApsaraDB for Mongodb', 'desc' => 'Hello, M
ongodb']);
echo "Inserted with Object ID '{$result->getInsertedId()}'", "\n";
$result = $collection->find(['name' => 'ApsaraDB for Mongodb']);
foreach ($result as $entry) {
    echo $entry->_id, ': ', $entry->name, "\n";
}
?>
```

Java

Related links:

- Official Getting Started.
- JAR package download.

1. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
2. Use the following Java sample code:
   - Maven configurations

```
<dependencies>
    <dependency>
        <groupId>org.mongodb</groupId>
        <artifactId>mongo-java-driver</artifactId>
        <version>4.1.0</version>
    </dependency>
</dependencies>
```

   - Java sample code

```
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import org.bson.BsonDocument;
import org.bson.BsonString;
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.MongoClientOptions;
```

```
import com.mongodb.MongoClientURI;
import com.mongodb.MongoCredential;
import com.mongodb.ServerAddress;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
 public class Main {
        public static ServerAddress seed1 = new ServerAddress("**********.mongodb.tbc
3.newtest.rdstest.aliyun-inc.com",
                        27017);
        public static ServerAddress seed2 = new ServerAddress("**********.mongodb.tbc
3.newtest.rdstest.aliyun-inc.com",
                        27017);
        public static String username = "demouser";
        public static String password = "**********";
        public static String ReplSetName = "mgset-**********";
        public static String DEFAULT_DB = "admin";
        public static String DEMO_DB = "test";
        public static String DEMO_COLL = "testColl";
         public static MongoClient createMongoDBClient() {
                // Construct a seed list.
                List<ServerAddress> seedList = new ArrayList<ServerAddress>();
                seedList.add(seed1);
                seedList.add(seed2);
                // Construct authentication information.
                List<MongoCredential> credentials = new ArrayList<MongoCredential>();
                credentials.add(MongoCredential.createScramSha1Credential(username, D
EFAULT_DB,
                                password.toCharArray()));
                // Construct operation options. Configure options other than required
ReplicaSetName based on your actual requirements. The default parameter settings are
sufficient in most scenarios.
                MongoClientOptions options = MongoClientOptions.builder().requiredRep
licaSetName(ReplSetName)
                                .socketTimeout(2000).connectionsPerHost(1).build();
                return new MongoClient(seedList, credentials, options);
        }
         public static MongoClient createMongoDBClientWithURI() {
                // Use a URI to initialize the MongoClient.
                // mongodb://[username:password@]host1[:port1][,host2[:port2],...[,ho
stN[:portN]]][/[database][?options]]
                MongoClientURI connectionString = new MongoClientURI("mongodb://" + u
sername + ":" + password + "@"
                                + seed1 + "," + seed2 + "/" + DEFAULT_DB + "?replicaS
et=" + ReplSetName);
                return new MongoClient(connectionString);
        }
         public static void main(String args[]) {
                MongoClient client = createMongoDBClient();
                // or
                // MongoClient client = createMongoDBClientWithURI();
                try {
                        // Obtain the collection handle.
                        MongoDatabase database = client.getDatabase(DEMO_DB);
                        MongoCollection<Document> collection = database.getCollection
```

```
                        MongoCollection<Document> collection = database.getCollection
(DEMO_COLL);
                        // Insert data.
                        Document doc = new Document();
                        String demoname = "JAVA:" + UUID.randomUUID();
                        doc.append("DEMO", demoname);
                        doc.append("MESG", "Hello AliCoudDB For MongoDB");
                        collection.insertOne(doc);
                        System.out.println("insert document: " + doc);
                        // Read data.
                        BsonDocument filter = new BsonDocument();
                        filter.append("DEMO", new BsonString(demoname));
                        MongoCursor<Document> cursor = collection.find(filter).iterat
or();
                        while (cursor.hasNext()) {
                                System.out.println("find document: " + cursor.next())
;
                        }
                } finally {
                        // Close the client and release resources.
                        client.close();
                }
                return;
        }
}
```

Python

Related links:

- PyMongo download.
- Official documentation.

1. Install PyMongo.

   ```
   pip install pymongo
   ```

2. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
3. Use the following Python sample code:

```
import uuid
from pymongo import MongoClient
CONN_ADDR1 = '**********.mongodb.****.******.rdstest.aliyun-inc.com:27017'
CONN_ADDR2 = '**********.mongodb.****.******.rdstest.aliyun-inc.com:27017'
REPLICAT_SET = 'mgset-**********'
 username = 'demouser'
password = '********'
# Obtain the MongoClient.
client = MongoClient([CONN_ADDR1, CONN_ADDR2], replicaSet=REPLICAT_SET)
# Authenticate the username and password used to log on to ApsaraDB for MongoDB. The us
ername in this sample code is used to log on to the admin database.
client.admin.authenticate(username, password)
# Insert doc and search for documents based on the demo name. The collection:testColl o
f the test database is used in the example.
demo_name = 'python-' + str(uuid.uuid1())
print 'demo_name:', demo_name
 doc = dict(DEMO=demo_name, MESG="Hello ApsaraDB For MongoDB")
doc_id = client.test.testColl.insert(doc)
print 'doc_id:', doc_id
 for d in client.test.testColl.find(dict(DEMO=demo_name)):
    print 'find documents:', d
```

C#

Related link: MongoDB C# Driver.

1. Install the following driver package:

```
mongocsharpdriver.dll
```

2. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
3. Use the following C# sample code:

```
using MongoDB.Driver;
using System;
using System.Collections.Generic;
namespace Aliyun
{
    class Program
    {
        static void Main(string[] args)
        {
            // Specify instance information.
            const string host1 = "dds-t4n**************.mongodb.singapore.rds.aliyuncs.
com";
            const int port1 = 3717;
            const string host2 = "dds-t4n**************.mongodb.singapore.rds.aliyuncs.
com";
            const int port2 = 3717;
            const string replicaSetName = "mgset-300******";
            const string admin = "admin";
            const string userName = "root";
            const string passwd = "********";
            try
            {
```

```
            {
                Console.WriteLine("connecting...");
                MongoClientSettings settings = new MongoClientSettings();
                List<MongoServerAddress> servers = new List<MongoServerAddress>();
                servers.Add(new MongoServerAddress(host1, port1));
                servers.Add(new MongoServerAddress(host2, port2));
                settings.Servers = servers;
               // Set ReplicaSetName.
                settings.ReplicaSetName = replicaSetName;
                // Set ConnectTimeout to 3.
                settings.ConnectTimeout = new TimeSpan(0, 0, 0, 3, 0);
                MongoCredential credentials = MongoCredential.CreateCredential(admin, u
serName, passwd);
                settings.Credential = credentials;
                MongoClient client = new MongoClient(settings);
                var server = client.GetServer();
                MongoDatabase database = server.GetDatabase("test");
                var collection = database.GetCollection<User>("test_collection");
                User user = new User();
                user.id = "1";
                user.name = "mongo_test";
                user.sex = "female";
                // Insert data user.
                collection.Insert(user);
                // Obtain a data entry.
                User result = collection.FindOne();
                Console.WriteLine("id:" + result.id + " name:" + result.name + " sex:"+
result.sex);
                Console.WriteLine("connection successful...");
            }
            catch (Exception e)
            {
                Console.WriteLine("connection failed:"+e.Message);
            }
        }
    }
    class User
    {
        public string id { set; get; }
        public string name { set; get; }
        public string sex { set; get; }
    }
}
```

Go

Related link: MongoDB Go Driver.

1. Install the following driver package:

```
go get gopkg.in/mgo.v2
```

2. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
3. Use the following Go sample code:

```
package main
import (
"fmt"
"log"
"time""gopkg.in/mgo.v2"
"gopkg.in/mgo.v2/bson"
)
type Person struct {
Name  string
Phone string
}
func main() {
fmt.Println("hello world")
dialInfo := &mgo.DialInfo{
Addrs:     []string{"dds-bp1***********-pub.mongodb.rds.aliyuncs.com:3717", "dds-bp1***
********-pub.mongodb.rds.aliyuncs.com:3717"},
Direct:    false,
Timeout:   time.Second * 1,
Database:  "admin",
Source:    "admin",
Username:  "root",
Password:  "********",
}
session, err := mgo.DialWithInfo(dialInfo)
if err != nil {
fmt.Printf("dial failed: %v",err)
return
}
defer session.Close()
session.SetMode(mgo.Monotonic, true)
c := session.DB("test").C("test_collection")
err = c.Insert(&Person{"Ale", "+55 53 8116 9639"},
    &Person{"Cla", "+55 53 8402 8510"})
if err != nil {
    log.Fatal(err)
}
result := Person{}
err = c.Find(bson.M{"name": "Ale"}).One(&result)
if err != nil {
    log.Fatal(err)
}
fmt.Println("Phone:", result.Phone)
}
```

5.5.4. Connect to an ApsaraDB for MongoDB instance by using the program code

ApsaraDB for MongoDB is compatible with the MongoDB protocol. This topic describes the sample code used to connect to an ApsaraDB for MongoDB instance in different languages.

Preparations

- Obtain the connection strings of an ApsaraDB for MongoDB instance based on the instance type. For more information, see the following topics:
  - Connect to a replica set instance
  - Connect to a sharded cluster instance
- Download and install the official driver package of your language. For more information, visit Start Developing with MongoDB.

> **Note**
> - The sample code in this topic applies only to internal endpoints of ApsaraDB for MongoDB replica set instances. For other types of instances, replace the relevant content based on actual conditions.
> - To connect to sharded cluster instances, you do not need to specify the replicaSet-related parameters.

Node.js

Related link: MongoDB Node.js Driver.

1. Initialize a project.

```
 mkdir node-mongodb-demo
cd node-mongodb-demo
npm init
```

2. Install the driver package and toolkit.

```
npm install mongodb node-uuid sprintf-js -save
```

3. Obtain the information required to connect to an ApsaraDB for MongoDB instance based on the instance type.
4. Use the following Node.js sample code:

```
 'use strict';
var uuid = require('node-uuid');
var sprintf = require("sprintf-js").sprintf;
var mongoClient = require('mongodb').MongoClient;
var host1 = "**********.mongodb.tbc3.newtest.rdstest.aliyun-inc.com";
var port1 = 27017;
var host2 = "**********.mongodb.tbc3.newtest.rdstest.aliyun-inc.com";
var port2 = 27017;
var username = "demouser";
var password = "**********";
var replSetName = "mgset-**********";
var demoDb = "test";
var demoColl = "testColl";
// The officially recommended solution.
var url = sprintf("mongodb://%s:%d,%s:%d/%s?replicaSet=%s", host1, port1, host2, port2,
demoDb, replSetName);
console.info("url:", url);
// Obtain the MongoClient.
mongoClient.connect(url, function (err, db) {
    if (err) {
```

```
        console.error("connect err:", err);
        return 1;
    }
    // Authenticate the username and password used to log on to ApsaraDB for MongoDB. T
he username in this sample code is used to log on to the admin database.
    var adminDb = db.admin();
    adminDb.authenticate(username, password, function (err, result) {
        if (err) {
            console.error("authenticate err:", err);
            return 1;
        }
        // Obtain the collection handle.
        var collection = db.collection(demoColl);
        var demoName = "NODE:" + uuid.v1();
        var doc = { "DEMO": demoName, "MESG": "Hello AliCoudDB For MongoDB" };
        console.info("ready insert document: ", doc);
        // Insert data.
        collection.insertOne(doc, function (err, data) {
            if (err) {
                console.error("insert err:", err);
                return 1;
            }
            console.info("insert result:", data["result"]);
            // Read data.
            var filter = { "DEMO": demoName };
            collection.find(filter).toArray(function (err, items) {
                if (err) {
                    console.error("find err:", err);
                    return 1;
                }
                console.info("find document: ", items);
                // Close the client and release resources.
                db.close();
            });
        });
    });
});
```

PHP

Related link: MongoDB PHP Driver.

1. Install the driver package and toolkit.

```
$ pecl install mongodb
$ echo "extension=mongodb.so" >> `php --ini | grep "Loaded Configuration" | sed -e "s|.
*:\s*||"`
$ composer require "mongodb/mongodb=^1.0.0"
```

2. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
3. Use the following PHP sample code:

```
<?php
require 'vendor/autoload.php'; // include Composer goodies
# Specify instance information.
$demo_seed1 = '**********.mongodb.test.aliyun-inc.com:3717';
$demo_seed2 = '**********.mongodb.test.aliyun-inc.com:3717';
$demo_replname = "mgset-**********";
$demo_user = 'root';
$demo_password = '**********';
$demo_db = 'admin';
# Construct a MongoDB connection string URI based on the instance information.
# mongodb://[username:password@]host1[:port1][,host2[:port2],...[,hostN[:portN]]][/[dat
abase][?options]]
$demo_uri = 'mongodb://' . $demo_user . ':' . $demo_password . '@' .
    $demo_seed1 . ',' . $demo_seed2 . '/' . $demo_db . '?replicaSet=' . $demo_replname;
$client = new MongoDB\Client($demo_uri);
$collection = $client->testDb->testColl;
$result = $collection->insertOne(['name' => 'ApsaraDB for Mongodb', 'desc' => 'Hello, M
ongodb']);
echo "Inserted with Object ID '{$result->getInsertedId()}'", "\n";
$result = $collection->find(['name' => 'ApsaraDB for Mongodb']);
foreach ($result as $entry) {
    echo $entry->_id, ': ', $entry->name, "\n";
}
?>
```

Java

Related links:

- Official Getting Started.
- JAR package download.

1. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
2. Use the following Java sample code:
   - Maven configurations

```
<dependencies>
    <dependency>
        <groupId>org.mongodb</groupId>
        <artifactId>mongo-java-driver</artifactId>
        <version>4.1.0</version>
    </dependency>
</dependencies>
```

   - Java sample code

```
package mongodb;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import org.bson.BsonDocument;
import org.bson.BsonString;
import org.bson.Document;
import com.mongodb.MongoClient;
```

```
import com.mongodb.MongoClientOptions;
import com.mongodb.MongoClientURI;
import com.mongodb.MongoCredential;
import com.mongodb.ServerAddress;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
public class shardingConnectTest {
    public static ServerAddress seed1 = new ServerAddress("11.238.102.108",
            3717);
    public static ServerAddress seed2 = new ServerAddress("11.238.102.61",
            3717);
    public static String username = "autotest";
    public static String password = "autotest";
//    public static String ReplSetName = "mgset-**********";
    public static String DEFAULT_DB = "admin";
    public static String DEMO_DB = "test";
    public static String DEMO_COLL = "testColl";
    public static MongoClient createMongoDBClient() {
        // 构建Seed列表
        List<ServerAddress> seedList = new ArrayList<ServerAddress>();
        seedList.add(seed1);
        seedList.add(seed2);
        // 构建鉴权信息
        List<MongoCredential> credentials = new ArrayList<MongoCredential>();
        credentials.add(MongoCredential.createScramSha1Credential(username, DEFAULT_D
B,
                password.toCharArray()));
        // 构建操作选项，requiredReplicaSetName属性外的选项根据自己的实际需求配置，默认参数满
足大多数场景
        MongoClientOptions options = MongoClientOptions.builder().socketTimeout(2000)
.connectionsPerHost(1).build();
        return new MongoClient(seedList, credentials, options);
    }
    public static MongoClient createMongoDBClientWithURI() {
        // 另一种通过URI初始化
        // mongodb://[username:password@]host1[:port1][,host2[:port2],...[,hostN[:por
tN]]][/[database][?options]]
        MongoClientURI connectionString = new MongoClientURI("mongodb://" + username
+ ":" + password + "@"
                + seed1 + "," + seed2 + "/" + DEFAULT_DB);
        return new MongoClient(connectionString);
    }
    public static void main(String args[]) {
        MongoClient client = createMongoDBClientWithURI();
        // or
        // MongoClient client = createMongoDBClientWithURI();
        try {
            // 取得Collection句柄
            MongoDatabase database = client.getDatabase(DEMO_DB);
            MongoCollection<Document> collection = database.getCollection(DEMO_COLL);
            // 插入数据
            Document doc = new Document();
            String demoname = "JAVA:" + UUID.randomUUID();
            doc.append("DEMO", demoname);
```

```
            doc.append("DEMO", demoname);
            doc.append("MESG", "Hello AliCoudDB For MongoDB");
            collection.insertOne(doc);
            System.out.println("insert document: " + doc);
            // 读取数据
            BsonDocument filter = new BsonDocument();
            filter.append("DEMO", new BsonString(demoname));
            MongoCursor<Document> cursor = collection.find(filter).iterator();
            while (cursor.hasNext()) {
                System.out.println("find document: " + cursor.next());
            }
        } finally {
            // 关闭Client，释放资源
            client.close();
        }
        return;
    }
}
```

Python

Related links:

- PyMongo download.
- Official documentation.

1. Install PyMongo.

```
pip install pymongo
```

2. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
3. Use the following Python sample code:

```
import uuid
from pymongo import MongoClient
CONN_ADDR1 = '**********.mongodb.****.******.rdstest.aliyun-inc.com:27017'
CONN_ADDR2 = '**********.mongodb.****.******.rdstest.aliyun-inc.com:27017'
REPLICAT_SET = 'mgset-**********'
 username = 'demouser'
password = '********'
# Obtain the MongoClient.
client = MongoClient([CONN_ADDR1, CONN_ADDR2], replicaSet=REPLICAT_SET)
# Authenticate the username and password used to log on to ApsaraDB for MongoDB. The us
ername in this sample code is used to log on to the admin database.
client.admin.authenticate(username, password)
# Insert doc and search for documents based on the demo name. The collection:testColl o
f the test database is used in the example.
demo_name = 'python-' + str(uuid.uuid1())
print 'demo_name:', demo_name
 doc = dict(DEMO=demo_name, MESG="Hello ApsaraDB For MongoDB")
doc_id = client.test.testColl.insert(doc)
print 'doc_id:', doc_id
 for d in client.test.testColl.find(dict(DEMO=demo_name)):
    print 'find documents:', d
```

C#

Related link: MongoDB C# Driver.

1. Install the following driver package:

```
mongocsharpdriver.dll
```

2. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
3. Use the following C# sample code:

```
using MongoDB.Driver;
using System;
using System.Collections.Generic;
namespace Aliyun
{
    class Program
    {
        static void Main(string[] args)
        {
            // Specify instance information.
            const string host1 = "dds-t4n**************.mongodb.singapore.rds.aliyuncs.
com";
            const int port1 = 3717;
            const string host2 = "dds-t4n**************.mongodb.singapore.rds.aliyuncs.
com";
            const int port2 = 3717;
            const string replicaSetName = "mgset-300******";
            const string admin = "admin";
            const string userName = "root";
            const string passwd = "********";
            try
            {
                Console.WriteLine("connecting...");
                MongoClientSettings settings = new MongoClientSettings();
                List<MongoServerAddress> servers = new List<MongoServerAddress>();
                servers.Add(new MongoServerAddress(host1, port1));
                servers.Add(new MongoServerAddress(host2, port2));
                settings.Servers = servers;
               // Set ReplicaSetName.
                settings.ReplicaSetName = replicaSetName;
                // Set ConnectTimeout to 3.
                settings.ConnectTimeout = new TimeSpan(0, 0, 0, 3, 0);
                MongoCredential credentials = MongoCredential.CreateCredential(admin, u
serName, passwd);
                settings.Credential = credentials;
                MongoClient client = new MongoClient(settings);
                var server = client.GetServer();
                MongoDatabase database = server.GetDatabase("test");
                var collection = database.GetCollection<User>("test_collection");
                User user = new User();
                user.id = "1";
                user.name = "mongo_test";
                user.sex = "female";
                // Insert data user.
```

```
                collection.Insert(user);
                // Obtain a data entry.
                User result = collection.FindOne();
                Console.WriteLine("id:" + result.id + " name:" + result.name + " sex:"+
result.sex);
                Console.WriteLine("connection successful...");
            }
            catch (Exception e)
            {
                Console.WriteLine("connection failed:"+e.Message);
            }
        }
    }
    class User
    {
        public string id { set; get; }
        public string name { set; get; }
        public string sex { set; get; }
    }
}
```

Go

Related link: MongoDB Go Driver.

1. Install the following driver package:

```
go get gopkg.in/mgo.v2
```

2. Obtain the information required to connect to an ApsaraDB for MongoDB instance.
3. Use the following Go sample code:

```
package main
import (
"fmt"
"log"
"time""gopkg.in/mgo.v2"
"gopkg.in/mgo.v2/bson"
)
type Person struct {
Name  string
Phone string
}
func main() {
fmt.Println("hello world")
dialInfo := &mgo.DialInfo{
Addrs:     []string{"dds-bp1***********-pub.mongodb.rds.aliyuncs.com:3717", "dds-bp1***
********-pub.mongodb.rds.aliyuncs.com:3717"},
Direct:    false,
Timeout:   time.Second * 1,
Database:  "admin",
Source:    "admin",
Username:  "root",
Password:  "********",
}
session, err := mgo.DialWithInfo(dialInfo)
if err != nil {
fmt.Printf("dial failed: %v",err)
return
}
defer session.Close()
session.SetMode(mgo.Monotonic, true)
c := session.DB("test").C("test_collection")
err = c.Insert(&Person{"Ale", "+55 53 8116 9639"},
    &Person{"Cla", "+55 53 8402 8510"})
if err != nil {
    log.Fatal(err)
}
result := Person{}
err = c.Find(bson.M{"name": "Ale"}).One(&result)
if err != nil {
    log.Fatal(err)
}
fmt.Println("Phone:", result.Phone)
}
```

5.5.5. Connect to a DynamoDB-compatible ApsaraDB for MongoDB instance by using application code

This topic describes how to connect to a DynamoDB-compatible ApsaraDB for MongoDB instance by using the Amazon Web Services (AWS) CLI and SDKs for programing languages such as Python and Java.

Prerequisites

A DynamoDB-compatible sharded cluster instance is created. To create a DynamoDB-compatible instance, you must select the **DynamoDB** protocol. For more information about how to create a DynamoDB-compatible instance, see Create a sharded cluster instance.

Preparations

- Obtain the connection string of a DynamoDB-compatible ApsaraDB for MongoDB instance.
  i. Log on to the ApsaraDB for MongoDB console.
  ii. In the top navigation bar, select the resource group and region to which the sharded cluster instance belongs.
  iii. In the left-side navigation pane, click **Sharded Cluster Instances**.
  iv. On the page that appears, find the instance and click its ID.
  v. In the left-side navigation pane, click **Database Connection**.
  vi. In the **Internal Connections - VPC** section, view the connection sting of the instance.

- (Optional)If your application is deployed on an Elastic Compute Service (ECS) instance, make sure that your sharded cluster instance and the ECS instance meet the following requirements to ensure network connectivity:
  - Your sharded cluster instance and the ECS instance belong to the same region. You can view the region of a created ECS instance. For more information, see View instance information.
  - (Optional)Your sharded cluster instance and the ECS instance belong to the same zone. This reduces network latency. You can view the zone of a created ECS instance. For more information, see View instance information.
  - Your sharded cluster instance and the ECS instance reside in the same type of network. You can view the network type of a created ECS instance. For more information, see View instance information. If an ECS instance resides in the **classic network**, you can migrate the ECS instance from the **classic network** to a **virtual private cloud (VPC)**. For more information, see Migrate ECS instances from the classic network to a VPC.

Use the AWS CLI to connect to the instance

This section demonstrates how to use the AWS CLI to connect to the DynamoDB-compatible ApsaraDB for MongoDB instance. Ubuntu 16.04.6 LTS is used in the example. For more information about the AWS CLI, see What is the AWS Command Line Interface?

1. Install the AWS CLI client.
   i. Run the following command to obtain the installation package of the latest AWS CLI version and rename it as `awscliv2.zip`:

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

ii. Run the following command to decompress the `awscliv2.zip` package:

```
unzip awscliv2.zip
```

> **Note** If you have not installed the `unzip` utility, run the `apt install unzip` command to install the unzip utility. Then, run the preceding command to decompress the awscliv2.zip package.

iii. Run the following command to install the AWS CLI client:

```
sudo ./aws/install
```

If the `You can now run: /usr/local/bin/aws --version` prompt appears in the CLI window, the AWS CLI client is installed. You can run the `/usr/local/bin/aws --version` command to query the version number of the AWS CLI.

2. Run the `/usr/local/bin/aws configure` command to configure the AWS CLI by specifying the parameters listed in the following table. Each time you specify a parameter, you must press the Enter key.

Parameter	Description	Example
`AWS Access Key ID`	The access key ID of your AWS account. If you do not have an access key ID, enter random characters.	`XXXXXXXXXX`
`AWS Secret Access Key`	The secret access key of your AWS account. If you do not have a secret access key, enter random characters.	`XXXXXXXXXX`
`Default region name`	The region where your AWS DynamoDB database resides. If you do not have a region, enter the one in the example.	`us-west-2`
`Default output format`	The default output format. This parameter can be left empty.	`json`

> **Note** For more information about AWS CLI configurations, see Configuration basics.

3. Run the following command to connect to the DynamoDB-compatible ApsaraDB for MongoDB instance and create a table:

```
aws dynamodb --endpoint-url <Connection string of the DynamoDB-compatible ApsaraDB for MongoDB instance> \
    create-table --table-name <Name of the table that you want to create> \
    --attribute-definitions AttributeName=<Name of the attribute>,AttributeType=<Data type of the attribute> \
    --key-schema AttributeName=<Attribute name of the primary key>,KeyType=<Role of the primary key> \
    --provisioned-throughput ReadCapacityUnits=<Provisioned read throughput>,WriteCapacityUnits=<Provisioned write throughput>
```

Parameter	Description
`--endpoint-url`	The connection string of the DynamoDB-compatible ApsaraDB for MongoDB instance. It must start with `HTTP://`.
`create-table`	The command used to create the table. **Note** For more information, see create-table.
`--table-name`	The name of the table that you want to create.
`--attribute-definitions`	An array of attributes that describe the key schema for the table and indexes. This parameter consists of the following attributes: ○ `AttributeName`: the name of the attribute. ○ `AttributeType`: the data type of the attribute. **Note** For more information, see create-table.
`--key-schema`	The attributes that make up the primary key for a table or an index. This parameter consists of the following attributes: ○ `AttributeName`: the name of the key attribute that is created by using the `--attribute-definitions` parameter. ○ `KeyType`: the role of the key attribute. **Note** For more information, see create-table.
`--provisioned-throughput`	The provisioned throughput settings for a table or an index. This parameter consists of the following attributes: ○ `ReadCapacityUnits`: the provisioned read throughput. ○ `WriteCapacityUnits`: the provisioned write throughput. **Note** For more information, see create-table.

Example:

```
/usr/local/bin/aws dynamodb --endpoint-url http://dds-xxxx.mongodb.rds.aliyuncs.com:3717 #Specify the connection string used to connect to the DynamoDB-compatible ApsaraDB for MongoDB instance.  \
    create-table --table-name student #Create a table named student.  \
    --attribute-definitions AttributeName=name,AttributeType=S AttributeName=age,AttributeType=N #Specify a STRING-typed name attribute and a NUMBER-typed age attribute for the table.  \
    --key-schema AttributeName=name,KeyType=HASH AttributeName=age,KeyType=RANGE #Specify the name attribute as a partition key and the age attribute as a sort key.  \
    --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5 #Set the provisioned read throughput and write throughput both to 5.
```

When the following content is displayed in the command output, you are connected to the DynamoDB-compatible ApsaraDB for MongoDB instance and a table is created.

```
{
    "TableDescription": {
        "TableName": "student",
        "KeySchema": [
            {
                "AttributeName": "name",
                "KeyType": "HASH"
            },
            {
                "AttributeName": "age",
                "KeyType": "RANGE"
            }
        ],
        "TableStatus": "CREATING",
        "TableSizeBytes": 0,
        "ItemCount": 0
    }
}
```

Use the SDK for Python to connect to the instance

- Python 2.6 or later is installed. For more information, visit the official website of Python.
- Boto3 is installed. For more information, see Boto3 documentation.

The following sample code shows how to use the SDK for Python to connect to the DynamoDB-compatible ApsaraDB for MongoDB instance and create a table named `Book`:

```
import boto3
def create_book_table(dynamodb=None):
    if not dynamodb:
        dynamodb = boto3.resource('dynamodb', endpoint_url="http://dds-xxxx.mongodb.rds.ali
yuncs.com:3717")
    table = dynamodb.create_table(
        TableName='Book',
        KeySchema=[
            {
                'AttributeName':'title',
                'KeyType':'HASH'
            },
            {
                'AttributeName':'year',
                'KeyType':'RANGE'
            }
        ],
        AttributeDefinitions=[
            {
                'AttributeName':'title',
                'AttributeType':'S'
            },
            {
                'AttributeName':'year',
                'AttributeType':'N'
            },
        ],
        ProvisionedThroughput={
            'ReadCapacityUnits':5,
            'WriteCapacityUnits':5
        }
    )
    return table

if __name__ == '__main__':
    book_table =create_book_table()
    print("Tablestatus:", book_table.table_status)
```

Use the SDK for Java to connect to the instance

Install AWS SDK for Java. For more information, see Setting up the AWS SDK for Java 2.x.

The following sample code shows how to use the SDK for Java to connect to the DynamoDB-compatible ApsaraDB for MongoDB instance and create a table named `Book`:

```
package com.amazonaws.codesamples.gsg;
import java.util.Arrays;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;
import com.amazonaws.services.dynamodbv2.document.Table;
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;
import com.amazonaws.services.dynamodbv2.model.KeySchemaElement;
import com.amazonaws.services.dynamodbv2.model.KeyType;
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;
import com.amazonaws.services.dynamodbv2.model.ScalarAttributeType;
public class MoviesCreateTable {
    public static void main(String[] args) throws Exception {

        AmazonDynamoDB client =AmazonDynamoDBClientBuilder.standard()
            .withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("http://d
ds-xxxx.mongodb.rds.aliyuncs.com:3717",
"us-east-1"))
            .build();
        DynamoDB dynamoDB = new DynamoDB(client);
        String tableName ="Book";
        try {
            System.out.println("Creating table...");
            Table table =dynamoDB.createTable(tableName,
                Arrays.asList(new
KeySchemaElement("title", KeyType.HASH), // Partition key
                    new KeySchemaElement("year", KeyType.RANGE)), // Sort key
                Arrays.asList(new AttributeDefinition("title", ScalarAttributeType.S),
                    new AttributeDefinition("year", ScalarAttributeType.N)),
                new ProvisionedThroughput(5L, 5L));
            table.waitForActive();
           System.out.println("OK. Table status: " + table.getDescription().getTableStatus(
));
        }
        catch (Exception e) {
           System.err.println("Unable to create table: ");
           System.err.println(e.getMessage());
        }
    }
}
```

Use SDKs for other programing languages to connect to the instance

For more information, see Getting Started with DynamoDB and AWS SDKs.