

Alibaba Cloud ApsaraDB for PolarDB

PolarDB PostgreSQL Database

Issue: 20200522









Legal disclaimer

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

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

5. By law, all the 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 contact Alibaba Cloud directly if you discover any errors in this document.

Document conventions

Style	Description	Example
	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.
	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.
	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.
	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 <code>cd /d C:/window</code> command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	<code>bae log list --instanceid Instance_ID</code>
[] or [a b]	This format is used for an optional value, where only one item can be selected.	<code>ipconfig [-all -t]</code>

Style	Description	Example
{ } or {a b}	This format is used for a required value, where only one item can be selected.	switch {active stand}

Contents

Legal disclaimer.....	1
Document conventions.....	1
1 Overview.....	1
2 PolarDB PostgreSQL Quick Start.....	2
3 Overview of data migration plans.....	3
4 Data Migration.....	4
4.1 Migrate data from a user-created PostgreSQL database to POLARDB for PostgreSQL.....	4
4.2 Migrate data from ApsaraDB RDS for PostgreSQL to POLARDB for PostgreSQL.....	8
5 Read/write splitting.....	12
6 Pending events.....	16
7 Configure a whitelist for a POLARDB for PostgreSQL cluster....	18
8 Billing management.....	19
8.1 Change the billing method from pay-as-you-go to subscription.....	19
8.2 Manually renew the subscription to a cluster.....	20
8.3 Automatically renew the subscription to a cluster.....	21
9 Connect to a database cluster.....	26
9.1 View connection endpoints.....	26
9.2 Connect to a POLARDB for PostgreSQL cluster.....	27
10 Cluster management.....	30
10.1 Create a POLARDB for PostgreSQL cluster.....	30
10.2 Configure cluster parameters.....	34
10.3 Change the cluster specifications.....	36
10.4 Add or remove a read-only node.....	39
10.5 Set the maintenance window.....	43
10.6 Restart a node.....	44
10.7 Release a cluster.....	45
10.8 Clone a cluster.....	46
10.9 Upgrade the minor version.....	47
10.10 Switch workloads from writer nodes to reader nodes.....	49
11 Account management.....	52
11.1 Overview.....	52
11.2 Register and log on to an Alibaba Cloud account.....	53
11.3 Create and authorize a RAM user.....	54
11.4 Create a database account.....	58
11.5 Manage a database account.....	60
12 Database management.....	62

13 Backup and restoration.....	65
13.1 Back up data.....	65
13.2 Restore data.....	67
14 Diagnostics and optimization.....	70
14.1 Performance monitoring and alert configuration.....	70
14.2 Performance insight.....	71
15 SQL Explorer.....	75
16 Plug-ins.....	80
16.1 Read and write external data files by using oss_fdw.....	80
16.2 Use the pg_pathman plug-in.....	85
16.3 Enable the zhparser plug-in.....	112

1 Overview

PolarDB is a next-generation cloud-based service developed by Alibaba Cloud for relational databases, which is compatible with MySQL, PostgreSQL, and Oracle. Based on a distributed storage architecture, PolarDB provides high-capacity, low-latency online transaction processing (OLTP) services, and cost-effective scalable services.

Basic concepts

- Cluster

A PolarDB cluster contains one primary instance and up to 15 read-only instances (at least one read-only instance must be provided to guarantee active-active high availability support). A PolarDB cluster ID starts with pc, which stands for PolarDB cluster.

- Instance

An instance is an independent database server in which you can create and manage multiple databases. An instance ID starts with pi, which stands for PolarDB instance.

- Database

A database is a logical unit created in an instance. The name of each PolarDB database under the same instance must be unique.

- Region and zone

Each region is a separate geographic area. Zones are distinct locations within a region that operate on independent power grids and networks. For more information, see [Alibaba Cloud's Global Infrastructure](#).

Console

Alibaba Cloud offers a web-based and easy-to-use console where you can manage various products and services including PolarDB. In the console, you can create, access, and configure your PolarDB database.

For more information about the console layout, see [Alibaba Cloud console](#).

[PolarRDB console](#).

2 PolarDB PostgreSQL Quick Start

This topic provides a quick start guide about how to manage PolarDB PostgreSQL clusters, such as creating a cluster, specifying basic configurations, and connecting to a cluster. It allows you to familiarize yourself with the entire process of purchasing and using a PolarDB PostgreSQL cluster.

Procedure

To purchase and use a PolarDB PostgreSQL cluster, follow these steps:

1. [Create a PolarDB PostgreSQL cluster](#)
2. [Configure whitelists.](#)
3. [Create accounts.](#)
4. [View connection endpoints.](#)
5. [Connect to the cluster.](#)

3 Overview of data migration plans

ApsaraDB for POLARDB provides various data migration solutions to meet different business needs such as migrating data to the cloud and migrating data between different cloud service providers. This allows you to smoothly migrate your database to Alibaba Cloud ApsaraDB for POLARDB without affecting your business. By using Alibaba Cloud [Data Transmission Service](#) (DTS), you can implement the schema migration and full migration of POLARDB databases.

Data migration

Scenario	Reference
Migrate data from ApsaraDB for RDS to ApsaraDB for POLARDB	Migrate data from ApsaraDB RDS for PostgreSQL to POLARDB for PostgreSQL
Migrate data from a user-created database to ApsaraDB for POLARDB	Migrate data from a user-created PostgreSQL database to POLARDB for PostgreSQL

4 Data Migration

4.1 Migrate data from a user-created PostgreSQL database to POLARDB for PostgreSQL

This topic describes how to migrate data from a user-created PostgreSQL database to POLARDB for PostgreSQL by running the `pg_dumpall`, `pg_dump`, and `pg_restore` commands.

For details about how to migrate data from an ApsaraDB RDS for PostgreSQL database, see [Migrate data from ApsaraDB RDS for PostgreSQL to POLARDB for PostgreSQL](#).

Prerequisites

The storage capacity of the POLARDB for PostgreSQL instance must be greater than that of the user-created PostgreSQL database.

Precautions

This is a full migration. To avoid inconsistencies in data, stop the services related to the user-created database and stop data writing before migration.

Preparations

1. Create a Linux ECS instance. This example uses an ECS instance running 64-bit Ubuntu 16.04. For more information, see [Create an ECS instance](#).

**Note:**

- The ECS instance and the destination POLARDB for PostgreSQL instance must be in the same VPC.
- You can create a pay-as-you-go ECS instance and release it after the migration.

2. Install PostgreSQL on the ECS instance to run the data restoration commands. For more information, see [PostgreSQL official documentation](#).

**Note:**

Ensure that the version of the installed PostgreSQL database is the same as that of the user-created PostgreSQL database.

Step 1: Back up the user-created PostgreSQL database

This is a full migration. To avoid inconsistencies in data, stop the services related to the user-created database and stop data writing before migration.

1. Run the following command on the user-created PostgreSQL database server to back up all the role information in the database.

```
pg_dumpall -U <username> -h <hostname> -p <port> -r -f <filename>
```

Parameter description:

- <username>: the account used to log on to the user-created PostgreSQL database.
- <hostname>: the endpoint of the user-created PostgreSQL database. localhost can be used for a local host.
- <port>: the port number of the database service.
- <filename>: the name of the generated backup file.

Example:

```
pg_dumpall -U postgres -h localhost -p 5432 -r -f roleinfo.sql
```

2. Enter the password in the Password: prompt to start role information backup.
3. Run the vim command to replace SUPERUSER in the role information backup file with polar_superuser.



Note:

If the role information backup file does not contain SUPERUSER information, you can skip this step.

```
-- PostgreSQL database cluster dump
--
SET default_transaction_read_only = off;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
--
-- Roles
--
CREATE ROLE data1;
ALTER ROLE data1 WITH NOSUPERUSER INHERIT CREATEROLE CREATEDB LOGIN NOREPLICATION NOBYPASSRLS PASSWORD 'md5';
CREATE ROLE manisha;
ALTER ROLE manisha WITH NOSUPERUSER INHERIT NOCREATEROLE NOCREATEDB LOGIN NOREPLICATION NOBYPASSRLS PASSWORD 'md5';
CREATE ROLE postgres;
ALTER ROLE postgres WITH SUPERUSER INHERIT CREATEROLE CREATEDB LOGIN REPLICATION BYPASSRLS PASSWORD 'md5';
CREATE ROLE testuser;
ALTER ROLE testuser WITH NOSUPERUSER INHERIT NOCREATEROLE CREATEDB LOGIN NOREPLICATION NOBYPASSRLS;
--
-- PostgreSQL database cluster dump complete
```

4. Run the following command to back up data of the user-created PostgreSQL database.

```
pg_dump -U <username> -h <hostname> -p <port> <dbname> -Fd -j <njobs> -f <dumpdir>
```

Parameter description:

- <username>: the account used to log on to the user-created PostgreSQL database.
- <hostname>: the endpoint of the user-created PostgreSQL database. localhost can be used for a local host.
- <port>: the port number of the database service.
- <dbname>: the name of the database to be backed up.
- <njobs>: the number of concurrent backup jobs.



Note:

- Specifying the <njobs> parameter can shorten the dump time, but it also increases the load on the database server.
 - If the version of the user-created PostgreSQL database is earlier than 9.2, you must specify the `--no-synchronized-snapshots` parameter.
- <dumpdir>: the directory of the generated backup file.

Example:

```
pg_dump -U postgres -h localhost -p 5432 mytestdata -Fd -j 5 -f postgresdump
```

5. Enter the password in the Password: prompt to start data backup.

6. Wait until the backup is completed. The data in the PostgreSQL database is backed up to the specified directory. In this example, the data is stored in the postgresdump directory.

Step 2: Migrate data to POLARDB for PostgreSQL

1. Upload the directory of backup files to the ECS instance.



Note:

Backup files include role information backup files and database backup files.

2. Run the following command on the ECS instance to migrate role information in backup files to the POLARDB for PostgreSQL instance.

```
psql -U <username> -h <hostname> -p <port> -d <dbname> -f <filename>
```

Parameter description:

- <username>: the account used to log on to the POLARDB for PostgreSQL database.
- <hostname>: the primary endpoint (private network) of the POLARDB for PostgreSQL instance.
- <port>: the port number of the database service. The default value is **1921**.
- <dbname>: the name of the database to connect to. The default value is postgres.
- <filename>: the name of the role information backup file.

```
psql -U gctest -h pc-xxxxxxx.pg.polardb.cn-qd-pldb1.rds.aliyuncs.com -d postgres -p 1921 -f roleinfo.sql
```

3. Enter the password in the Password: prompt to start role information import.
4. Run the following command on the ECS instance to restore data to the POLARDB for PostgreSQL instance.

```
pg_restore -U <username> -h <hostname> -p <port> -d <dbname> -j <njobs> <dumpdir>
```

Parameter description:

- <username>: the account used to log on to the POLARDB for PostgreSQL database.
- <hostname>: the primary endpoint (private network) of the POLARDB for PostgreSQL instance. For more information, see [View connection endpoints](#).
- <port>: the port number of the database service. The default value is **1921**.
- <dbname>: the name of the destination database to connect to and restore data.



Note:

A destination database must be available. If not, create a database in the destination instance.

- <njobs>: the number of concurrent data restoration jobs.



Note:

Specifying this parameter can shorten data restoration time, but it also increases the load on the database server.

- <dumpdir>: the directory where the backup file is located.

Example:

```
pg_restore -U gctest -h pc-mxxxxxxx.pg.polardb.cn-qd-pldb1.rds.aliyuncs.com -p 1921 -d mytestdata -j 6 postgresdump
```

5. Enter the password in the Password: prompt to start data migration.

**Note:**

For details about how to change the password if you forget your password, see [Reset the password of a database account](#).

Wait until the data migration is complete.

4.2 Migrate data from ApsaraDB RDS for PostgreSQL to POLARDB for PostgreSQL

This topic describes how to migrate data from a user-created PostgreSQL database to POLARDB for PostgreSQL by running the `pg_dump` and `pg_restore` commands.

For details about how to migrate data from an ApsaraDB RDS for PostgreSQL database, see [Migrate data from a user-created PostgreSQL database to POLARDB for PostgreSQL](#).

Prerequisites

The storage capacity of the POLARDB for PostgreSQL instance must be greater than that of the ApsaraDB RDS for PostgreSQL instance.

Precautions

This is a full migration. To avoid inconsistencies in data, stop the services related to the ApsaraDB RDS for PostgreSQL database and stop data writing before migration.

Preparations

1. Create a Linux ECS instance. This example uses an ECS instance running 64-bit Ubuntu 16.04. For more information, see [Create an ECS instance](#).

**Note:**

- The ECS instance and the destination POLARDB for PostgreSQL instance must be in the same VPC.

- You can create a pay-as-you-go ECS instance and release it after the migration.

2. Install PostgreSQL on the ECS instance to run the data restoration commands. For more information, see [PostgreSQL official documentation](#).

**Note:**

Ensure that the version of the installed PostgreSQL database is the same as that of the ApsaraDB RDS for PostgreSQL database.

Step 1: Back up the ApsaraDB RDS for PostgreSQL database

This is a full migration. To avoid inconsistencies in data, stop the services related to the ApsaraDB RDS for PostgreSQL database and stop data writing before migration.

1. Run the following command on the ECS instance to back up data in the database.

```
pg_dump -U <username> -h <hostname> -p <port> <dbname> -Fd -j <njobs> -f <dumpdir>
```

Parameter description:

- <username>: the account used to log on to the ApsaraDB RDS for PostgreSQL database.
- <hostname>: the endpoint of the ApsaraDB RDS for PostgreSQL database. localhost can be used for a local host.
- <port>: the port number of the database service.
- <dbname>: the name of the database to connect to. The default value is postgres.
- <njobs>: the number of concurrent backup jobs.

**Note:**

- Specifying the <njobs> parameter can shorten the dump time, but it also increases the load on the database server.
 - If your ApsaraDB RDS for PostgreSQL database is earlier than 9.2, you must specify the --no-synchronized-snapshots parameter.
- <dumpdir>: the directory of the generated backup file.

Example:

```
pg_dump -U postgres -h localhost -p 5432 postgres -Fd -j 5 -f postgresdump
```

2. Enter the password in the Password: prompt to start data backup.

3. Wait until the backup is completed. The data in the PostgreSQL database is backed up to the specified directory. In this example, the data is stored in the postgresdump directory.

Step 2: Migrate data to POLARDB for PostgreSQL

1. Connect to the POLARDB for PostgreSQL database from the ECS instance.

```
psql -U <username> -h <hostname> -p <port> -d <dbname>
```

Parameter description:

- <username>: the account used to log on to the POLARDB for PostgreSQL database.
- <hostname>: the primary endpoint (private network) of the POLARDB for PostgreSQL instance. For more information, see [View connection endpoints](#).
- <port>: the port number of the database service. The default value is **1921**.
- <dbname>: the name of the database to connect to.

Example:

```
psql -h pc-mxxxxxxx.pg.polardb.cn-qd-pldb1.rds.aliyuncs.com -p 3433 -d postgres -U gctest
```

2. Create a role in the destination POLARDB for PostgreSQL instance based on the role information in the source ApsaraDB RDS for PostgreSQL database and grant permissions to the destination database for data restoration. For more information, see [CREATE ROLE](#) and [GRANT](#) in official documentation.
3. Run the following command on the ECS instance to migrate data of the source database to the POLARDB for PostgreSQL instance.

```
pg_restore -U <username> -h <hostname> -p <port> -d <dbname> -j <njobs> <dumpdir>
```

Parameter description:

- <username>: the account used to log on to the POLARDB for PostgreSQL database.
- <hostname>: the primary endpoint (private network) of the POLARDB for PostgreSQL instance.
- <port>: the port number of the database service. The default value is **1921**.
- <dbname>: the name of the destination database to connect to and restore data.



Note:

A destination database must be available. If not, create a database in the destination instance.

- <njobs>: the number of concurrent data restoration jobs.

**Note:**

Specifying this parameter can shorten data restoration time, but it also increases the load on the database server.

- <dumpdir>: the directory where the backup file is located.

Example:

```
pg_restore -U gctest -h pc-mxxxxxxx.pg.polardb.cn-qd-pldb1.rds.aliyuncs.com -p 1921 -d postgres -j 6 postgresdump
```

4. Enter the password in the Password: prompt to start data migration.

**Note:**

For details about how to change the password if you forget your password, see [Reset the password of a database account](#).

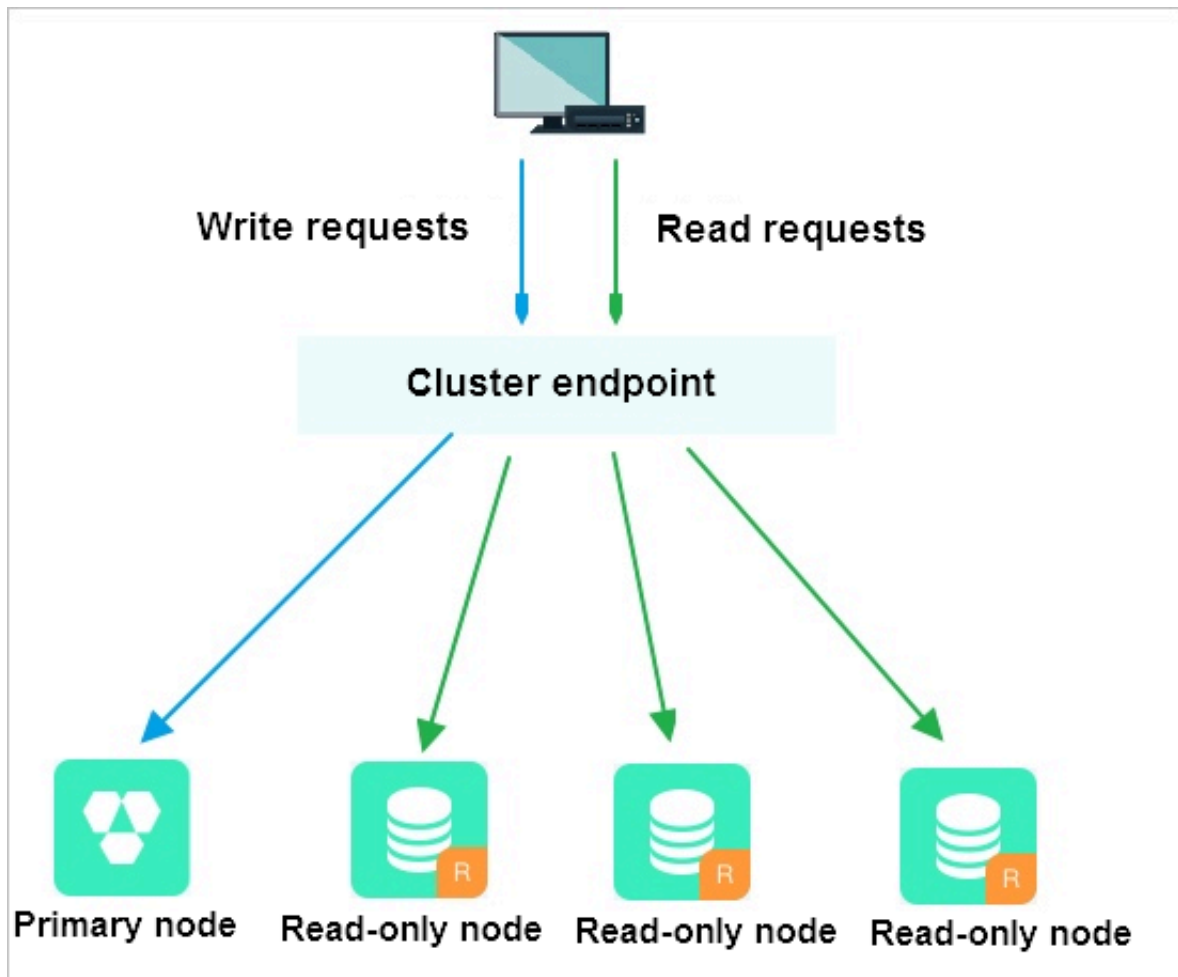
Wait until the data migration is complete.

5 Read/write splitting

POLARDB for PostgreSQL clusters support read/write splitting. Read and write requests sent to a cluster endpoint are automatically forwarded to the relevant nodes.

Context

When there is a large number of read requests but few write requests to a database, a single node may not be able to handle the workload. This may cause core services to be affected. Cluster endpoints automatically forward write requests to the primary node, while read requests are automatically forwarded to read-only nodes. This way, the read capability can be elastically scaled to handle a large number of read requests that are sent to databases.



Benefits

- One endpoint, simplified maintenance

If you do not send requests to the cluster endpoint, you must configure the endpoints of the primary node and each read-only node in the application to send write requests to the primary node and read requests to the read-only nodes. ApsaraDB for POLARDB provides a cluster endpoint. After you connect to this endpoint, read and write requests are automatically forwarded to the primary node and read-only nodes. This reduces maintenance costs. You can expand the capacity of an ApsaraDB for POLARDB cluster by adding read-only nodes, which saves you from making any modifications to applications.

- Session-level read consistency

When a client connects to the backend through the cluster endpoint, the built-in proxy for read/write splitting automatically establishes a connection with the primary node and each read-only node. In the same session, the built-in proxy first selects an appropriate node based on the data synchronization progress of each database node. Then, the proxy forwards read and write requests to the nodes whose data is up-to-date and correct, balancing the load between read and write requests.

- Load balancing of PREPARE statements

The built-in proxy automatically finds the database nodes that have previously executed PREPARE statements based on the information in EXECUTE statements, balancing the load of extended queries.

- Support for native high security links, improving performance

You can use a user-created proxy on the cloud to achieve read/write splitting. However, excessive latency may occur because data is parsed and forwarded by multiple components before arriving at a database. ApsaraDB for POLARDB utilizes a built-in proxy for read/write splitting, which offers reduced latency and enhanced query performance when compared with external components.

- Node health checks to enhance database availability

The read/write splitting module of ApsaraDB for POLARDB performs health checks on the primary node and read-only nodes of a cluster. When a node fails or its latency exceeds a specified threshold, ApsaraDB for POLARDB stops distributing read requests to this node and redirects these requests to other healthy nodes. This ensures that applications can

access the ApsaraDB for POLARDB cluster even if a read-only node fails. When the node is repaired, the node is automatically added to the request distribution system.

Limits

- The following commands or functions are not supported:
 - Connecting to a cluster through the replication-mode method. If you need to set up dual-node clusters based on a primary/secondary replication architecture, use the endpoint of the primary node.
 - The name of the temporary table cannot be used to declare the %ROWTYPE attribute.

```
create temp table fullname (first text, last text);
select '(Joe,von Blow)'::fullname, '(Joe,d"Blow)'::fullname;
```
 - Creating temporary resources by using functions.
 - Executing an SQL statement to query a temporary table that is created by a function may receive an error message indicating that the table does not exist.
 - Executing a function that contains the PREPARE statement may return an error message indicating that the PREPARE statement name does not exist.
- Routing-related restrictions:
 - Multi-statements are routed to the primary node, and all subsequent requests within this session are routed to the primary node.
 - A request message that is greater than or equal to 16 MB is routed to the primary node, and all subsequent requests within this session are routed to the primary node.
 - Requests in the transaction are routed to the primary node, and load balancing is resumed after the transaction terminates.
 - All statements that use functions (except aggregate functions such as COUNT and SUM) are routed to the primary node.

Apply for or change a cluster endpoint

1. Log on to the [ApsaraDB for POLARDB console](#).
2. In the upper-left corner of the console, select a region.
3. Click the ID of the target cluster.
4. On the **Overview** page, find **Cluster Endpoints** in the **Connection Information** section.

5. Click **Apply**. In the dialog box that appears, click **Confirm**. Refresh the page to view the cluster endpoint.




Note:


If an existing cluster does not have a cluster endpoint, you must manually apply for a cluster endpoint. A cluster endpoint is automatically assigned to newly purchased clusters. If an ApsaraDB for POLARDB cluster has a cluster endpoint, you can skip to Step 6 to change the endpoint.

6. Click **Modify**. In the Modify Endpoint dialog box, enter a new cluster endpoint and click **Submit**.

Connection Information

Whitelists  [Create Whitelist](#)

> default [Configure](#) [Delete](#)

Primary Endpoints 

VPC-facing Endpoint [\[redacted\]](#) [Modify](#)


Public-facing Endpoint [Apply](#)

Primary Node ([\[redacted\]](#))

VPC-facing Endpoint [\[redacted\]](#)

Read-only Node ([\[redacted\]](#))

VPC-facing Endpoint [\[redacted\]](#)

Cluster Endpoints [【Public Preview】](#) 

• Default Cluster Endpoint ([\[redacted\]](#)) [Modify](#)

Read/write Mode [Read and Write \(Automatic Read-write Splitting\)](#)

VPC-facing Endpoint [\[redacted\]](#) [Modify](#)

Public-facing Endpoint [Apply](#)

> Node Settings

> Advanced Settings

6 Pending events

When an ApsaraDB for POLARDB event is pending for processing, you will be notified to handle the event in a timely manner in the console.

For ApsaraDB for POLARDB O&M events, including database software upgrade events and hardware maintenance and upgrade events, you are notified not only by SMS messages, phone calls, emails, or internal messages, but also in the console. You can view the details of each event, including the event type, task ID, cluster name, and switch time. You can also change the switch time.

Prerequisites

There are unprocessed O&M events.



Note:

If there are unprocessed O&M events, you can see notification badges on the **Pending Events** page.

Task ID	Cluster Name	Compatible Database Engine	Start Time
---------	--------------	----------------------------	------------

Change the switch time

1. Log on to the [ApsaraDB for POLARDB console](#).
2. In the left-side navigation pane, click **Pending Events**.



Note:

For an O&M event for which you must reserve the switch time, a dialog box appears, asking you to complete the reservation as soon as possible.

3. On the **Pending Events** page, select the type of event that you want to handle.



Note:

Different notices are displayed on the tabs for different types of events.

Database Software Upgrade
Hardware Maintenance and Upgrade

Dear User, to provide you with better stability and performance, we will perform hardware and network upgrades for some of your instances.

1. The upgrade may use the hot migration method to replace the underlying hardware (server) of your database, but will not change the connection address (including IP and port) of the database.
2. The upgrade process takes up to 1 hour to complete. An up to 30-second disconnection may occur during the process. You can specify the disconnection time. Make sure that the business has a reconnection mechanism.

Modify Switch Time

<input type="checkbox"/>	Task ID	Cluster Name	Compatible Database Engine	Start Time	Switch Time	Start Deadline
No Hardware Maintenance and Upgrade Events						

4. View event details in the event list. To change the switch time, select an event, and then click **Change Switch Time**. In the dialog box that appears, set the switch time, and then click **OK**.



Note:

The switch time cannot be later than the latest operation time allowed.

Historical events

You can view completed events on the **Event History** page.

Clusters
Pending Events
Event History

Database Software Upgrade
Hardware Maintenance and Upgrade

Task ID	Cluster Name	Compatible Database Engine	Start Time

7 Configure a whitelist for a POLARDB for PostgreSQL cluster

**Note:**

POLARDB for PostgreSQL does not support configuring a cluster whitelist. Only instances in the same VPC can access the cluster.

8 Billing management

8.1 Change the billing method from pay-as-you-go to subscription

You can change the billing method of a cluster from pay-as-you-go to subscription based on your needs. Changing billing methods will not impact the performance of your cluster.

**Note:**

If a cluster uses a specification that is no longer available, you cannot change the billing method of the cluster to subscription. In this case, you must [change the cluster specifications](#) before changing the billing method.

Precautions

You cannot change the billing method of a cluster from subscription to pay-as-you-go. Consider your resource requirements before switching the billing method to subscription to avoid resource wastage.

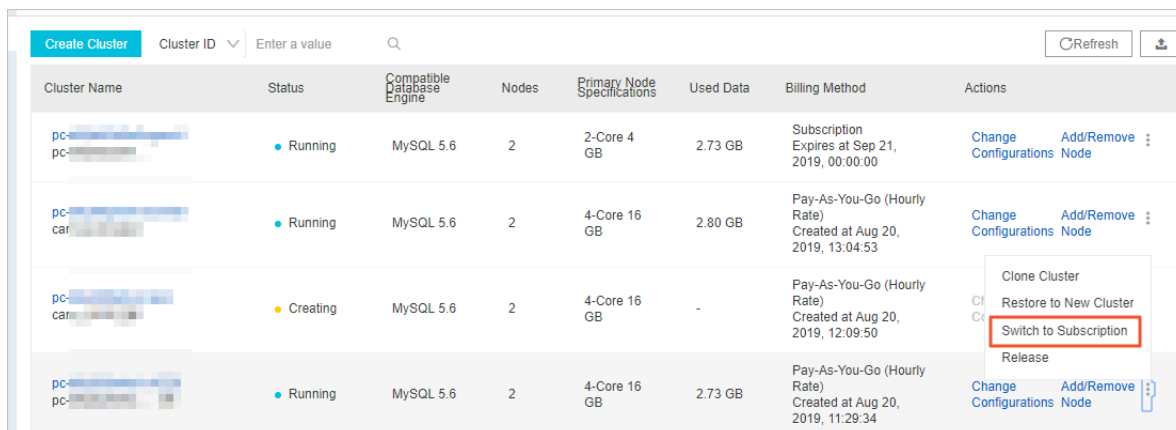
Prerequisites

- The cluster must be in the **Running** state.
- There are no pending orders for changing the billing method from pay-as-you-go to subscription. If there are any pending orders, you must complete payment for or discard them on the [Orders](#) page.

Procedure

1. Log on to the [ApsaraDB for POLARDB console](#).
2. Select the region where the cluster resides.

3. Find the target cluster. In the **Actions** column corresponding to the cluster, choose ... > **Switch to Subscription**.



Cluster Name	Status	Compatible Database Engine	Nodes	Primary Node Specifications	Used Data	Billing Method	Actions
pc-...	Running	MySQL 5.6	2	2-Core 4 GB	2.73 GB	Subscription Expires at Sep 21, 2019, 00:00:00	Change Configurations Add/Remove Node
pc-car	Running	MySQL 5.6	2	4-Core 16 GB	2.80 GB	Pay-As-You-Go (Hourly Rate) Created at Aug 20, 2019, 13:04:53	Change Configurations Add/Remove Node
pc-car	Creating	MySQL 5.6	2	4-Core 16 GB	-	Pay-As-You-Go (Hourly Rate) Created at Aug 20, 2019, 12:09:50	Clone Cluster Restore to New Cluster Switch to Subscription Release
pc-...	Running	MySQL 5.6	2	4-Core 16 GB	2.73 GB	Pay-As-You-Go (Hourly Rate) Created at Aug 20, 2019, 11:29:34	Change Configurations Add/Remove Node

4. Specify **Purchase Plan**, read the **ApsaraDB for POLARDB Subscription Agreement of Service**. Select the check box to indicate that you agree to it, and then click **Pay** to complete the payment.



Note:

- The new billing method takes effect after you complete the payment.
- If the order is unpaid or the payment fails, an unfinished order appears on the [Orders](#) page. You cannot buy a new cluster or switch the billing method of existing clusters before the unfinished order is completed. You must complete payment for or discard the order before placing a new one.

8.2 Manually renew the subscription to a cluster

You can renew your subscription to clusters in the ApsaraDB for PolarDB console or in the Renew console. In the Renew console, you can renew your subscription to multiple clusters at the same time.



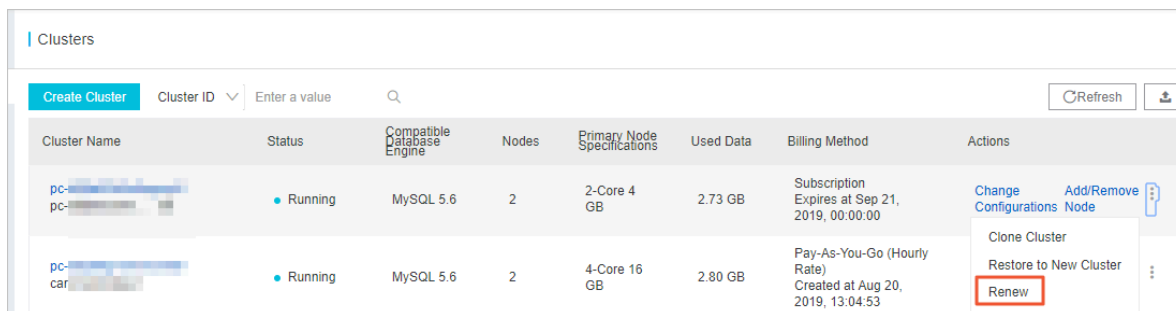
Note:

Clusters purchased through the pay-as-you-go (hourly rate) billing method do not involve expiration and renewal.

Method 1: Renew the subscription in the ApsaraDB for PolarDB console

- Log on to the [ApsaraDB for PolarDB console](#).
- Select a region in the upper-left corner to view all the clusters that you deploy in this region.

- Find the target cluster, click the **More** icon in the **Actions** column, and choose **Renew** from the shortcut menu.

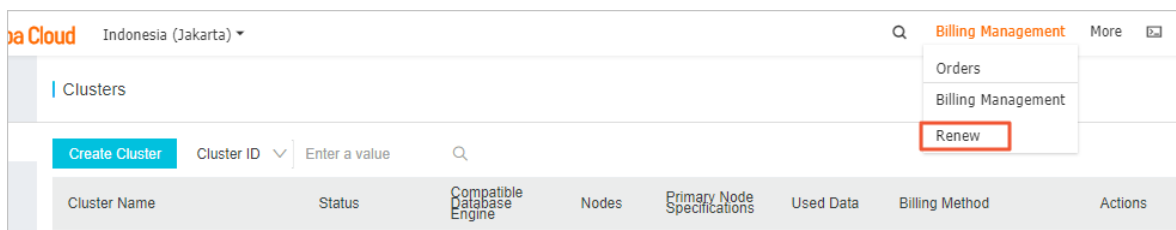


Cluster Name	Status	Compatible Database Engine	Nodes	Primary Node Specifications	Used Data	Billing Method	Actions
pc-xxxxxx	Running	MySQL 5.6	2	2-Core 4 GB	2.73 GB	Subscription Expires at Sep 21, 2019, 00:00:00	Change Configurations Add/Remove Node
pc-carxxxxx	Running	MySQL 5.6	2	4-Core 16 GB	2.80 GB	Pay-As-You-Go (Hourly Rate) Created at Aug 20, 2019, 13:04:53	Clone Cluster Restore to New Cluster Renew

- Specify the renewal duration, select the service agreement, and click **Pay**.

Method 2: Renew the subscription in the Renew console

- Log on to the [ApsaraDB for PolarDB console](#).
- In the upper-right corner of the console, choose **Billing Management** > **Renew**.



- In the left-side navigation pane, click **ApsaraDB for PolarDB**.
- Click the **Manually Renew** tab. Set the filtering conditions to find the target cluster. Click **Renew** in the **Actions** column corresponding to the cluster.



Note:

To enable manual renewal for a cluster on the **Auto-Renew** or **Don't Renew** tab, click **Enable Manual Renew**, and then click **OK** in the dialog box that appears.

- Specify the renewal duration, select the service agreement, and click **Pay**.

Enable automatic renewal

If you enable automatic renewal, you will be free from regular manual renewal operations and concerns of service interruptions. For more information, see [Automatically renew the subscription to a cluster](#).

8.3 Automatically renew the subscription to a cluster

A subscription-based cluster has a validity period. If the cluster is not renewed in a timely manner, service interruptions or even data loss will occur after it expires. If you enable

automatic renewal, you will be free from regular manual renewal operations and concerns of service interruptions.

**Note:**

Clusters purchased through the pay-as-you-go (hourly rate) billing method do not involve expiration and renewal.

Precautions

- Automatic fee deduction will begin nine days prior to the expiration of the cluster, supporting cash and coupons. Keep your account balance adequate.
- If you manually renew the cluster before the automatic deduction, the system will automatically renew the cluster nine days prior to the next expiration.
- The automatic renewal feature takes effect the next day after it is enabled. If your cluster expires the next day, renew it manually to prevent service interruptions. For more information, see [Manually renew the subscription to a cluster](#).

Enable automatic renewal when purchasing a cluster**Note:**

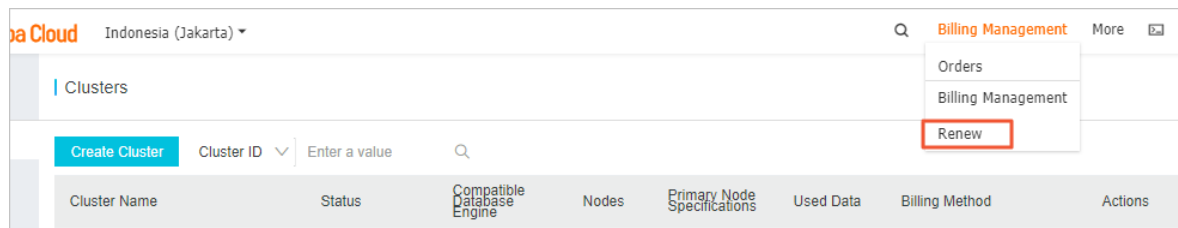
After you enable automatic renewal, the system will automatically renew the subscription based on the **subscription period**. For example, if you purchase a cluster for three months and select automatic renewal, you will be charged a fee of the three-month subscription upon each automatic renewal.

When creating a cluster, you can select **Auto Renew**.

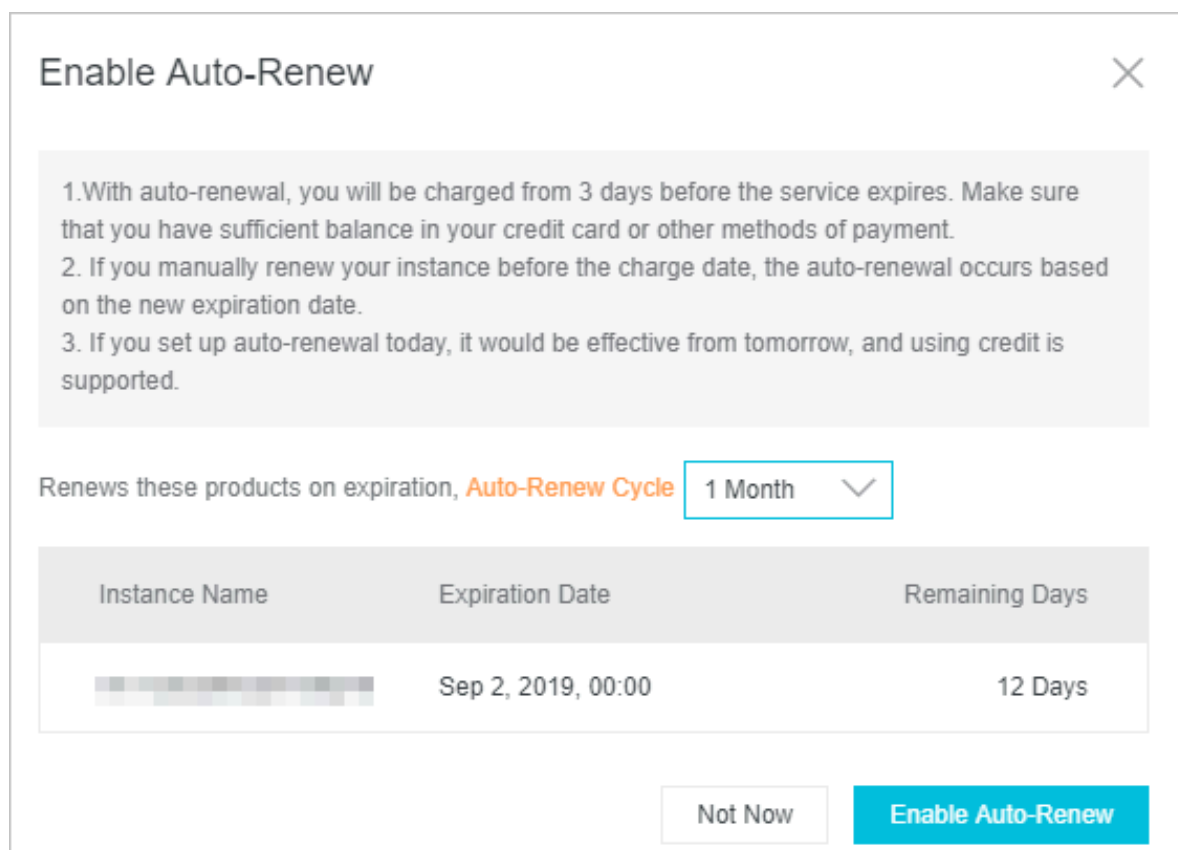
Enable automatic renewal after purchasing a cluster**Note:**

After you enable automatic renewal, the system will automatically renew the subscription based on the renewal cycle you select. For example, if you select a three-month renewal cycle, you will be charged a fee of the three-month subscription upon each automatic renewal.

1. Log on to the [ApsaraDB for POLARDB console](#).
2. In the upper-right corner of the console, choose **Billing Management > Renew**.



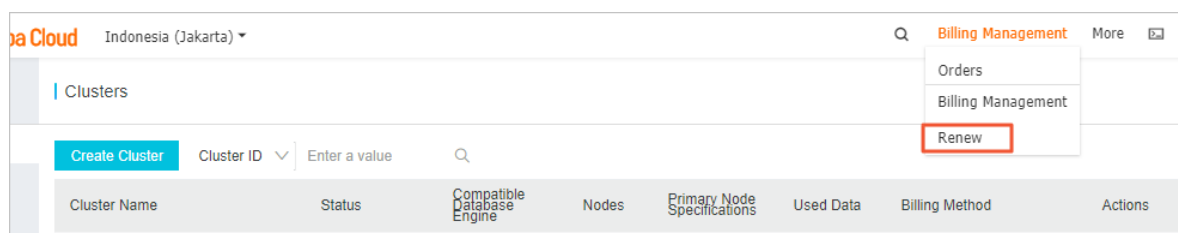
3. In the left-side navigation pane, click **ApsaraDB for POLARDB**.
4. Click the **Manually Renew** or **Don't Renew** tab in the Renew console. Set the filtering conditions to find the target cluster. Click **Enable Auto-Renew** in the **Actions** column corresponding to the cluster.
5. In the dialog box that appears, select the automatic renewal cycle, and click **Enable Auto-Renew**.



Edit the automatic renewal cycle

1. Log on to the [ApsaraDB for POLARDB console](#).

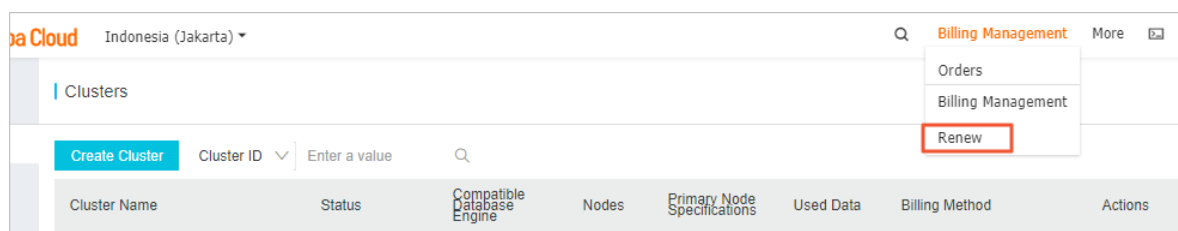
2. In the upper-right corner of the console, choose **Billing Management** > **Renew**.



3. In the left-side navigation pane, click **ApsaraDB for POLARDB**.
4. Click the **Auto-Renew** tab on the Renew console. Set the filtering conditions to find the target cluster. Click **Enable Auto-Renew** in the **Actions** column corresponding to the cluster.
5. Click the **Auto** tab. Set the filtering conditions to find the target cluster. Click **Modify Auto-Renew** in the **Actions** column corresponding to the cluster.
6. In the dialog box that appears, edit the automatic renewal cycle, and click **OK**.

Disable automatic renewal

1. Log on to the [ApsaraDB for POLARDB console](#).
2. In the upper-right corner of the console, choose **Billing Management** > **Renew**.



3. In the left-side navigation pane, click **ApsaraDB for POLARDB**.
4. Click the **Auto-Renew** tab in the Renew console. Set the filtering conditions to find the target cluster. Click **Modify Auto-Renew** in the **Actions** column corresponding to the cluster.

5. Select **Disable Auto-Renew** and click **OK**.


Modify Auto-Renew ✕

1. With auto-renewal, you will be charged from 3 days before the service expires. Make sure that you have sufficient balance in your credit card or other methods of payment.
2. If you manually renew your instance before the charge date, the auto-renewal occurs based on the new expiration date.
3. If you set up auto-renewal today, it would be effective from tomorrow, and using credit is supported.

The changes will be applied to the following products:



☐ Modify Auto-Renew Cycle
1 Month ▼

☒ Disable Auto-Renew

Instance Name	Expiration Date	Remaining Days
	Sep 2, 2019, 00:00	12 Days

Cancel OK

Related API operations

API operation	Description
#unique_21	<p>Creates a POLARDB cluster.</p> <div> Note: You can enable automatic renewal when you create a cluster.</div>
#unique_22	<p>Enables automatic renewal for a subscription-based cluster.</p> <div> Note: You can enable automatic renewal after you create a cluster.</div>
#unique_23	<p>Queries the automatic renewal status of a subscription-based cluster.</p>

9 Connect to a database cluster

9.1 View connection endpoints

This topic describes how to view connection endpoints of the POLARDB cluster and introduces primary endpoints and private endpoints.

Procedure

1. Log on to the [ApsaraDB for POLARDB console](#).
2. Find the target cluster and click the cluster ID.
3. In the **Connection Information** section, view the connection endpoints.

Primary endpoints

Type	Description	Supported network type
Primary endpoint	A primary endpoint always connects to the primary node and supports read and write operations. If the primary node becomes faulty, the primary endpoint is automatically switched to the read-only node that is promoted to the primary node.	VPC
Primary node endpoint (not recommended)	The endpoint of the primary node. It is not recommended to connect directly to the primary node because the node is unavailable when it fails.	VPC
Read-only node endpoint (not recommended)	The endpoint of the read-only node. It is not recommended to connect directly to the read-only node because the node is unavailable when it fails.	VPC

Private endpoints

Type	Description	Scenario
Private endpoint	<ul style="list-style-type: none">POLARDB can achieve optimal performance when accessed through the private endpoint.The private endpoint cannot be released.	<p>For example:</p> <ul style="list-style-type: none">If your ECS instance is located in the same VPC as the POLARDB cluster, then your ECS instance can access the POLARDB cluster through the private endpoint.You can access the POLARDB cluster through the private endpoint by using DMS.

Next step

Connect to the cluster. For more information, see [Connect to a POLARDB for PostgreSQL cluster](#).

Related API operations

9.2 Connect to a POLARDB for PostgreSQL cluster

This topic describes how to connect to a POLARDB for PostgreSQL cluster.

Use DMS to connect to a POLARDB for PostgreSQL cluster

[Data Management](#) (DMS) provides an integrated solution for data management. DMS supports data management, schema management, access control, BI charts, trend analysis, data tracing, performance optimization, and server management. DMS supports relational databases such as MySQL, SQL Server, and PostgreSQL, as well as NoSQL databases such as MongoDB and Redis. DMS also supports the management of Linux servers.

Prerequisites

You have created a privileged or standard account for an existing database cluster. For more information, see [Create a database account](#).

Procedure

1. Find the target cluster and click the cluster ID to go to the basic information page.

2. Click **Log On to Database** in the upper-right corner of the page.



3. On the database logon page, enter the endpoint and the port number, and separate them with a colon (:). Then enter the username and the password of the privileged or standard account, and click **Log On**.

**Note:**

DMS logon only supports the endpoint and does not support the cluster address.

For more information about how to view the endpoint, see [View connection endpoints](#).

A screenshot of the 'RDS Database Logon' page. It features a green database icon on the left. The title 'RDS Database Logon' is in large black font, with 'Independent Unit' and a dropdown arrow to its right. Below the title are four input fields: 1. A dropdown menu showing 'pc-bp...' and 'mysql.polaradb.rds.aliyuncs.com:3306' with a downward arrow. 2. A dropdown menu labeled 'Databases Username' with a downward arrow. 3. A text input field labeled 'Password'. 4. A checkbox labeled 'Remember Password'. At the bottom is a large blue button labeled 'Log On'.**Use a client to connect to a POLARDB for PostgreSQL cluster**

POLARDB for PostgreSQL does not support setting a whitelist for a cluster. Only instances that are in the same VPC can access the cluster. Therefore, the server where the client resides and the POLARDB compatible with Oracle cluster must be in the same VPC.

1. Start the pgAdmin 4 client.
2. Right-click **Servers** and choose **Create > Server** from the shortcut menu, as shown in the following figure.
3. On the **General** tab of the **Create - Server** dialog box, enter the name of the server, as shown in the following figure.

4. Click the **Connection** tab and enter the information of the instance to connect to, as shown in the following figure.

Parameter description:

- **Hostname or IP Address:** the internal IP address of the POLARDB for PostgreSQL cluster. To view the endpoint and port information of the POLARDB for PostgreSQL cluster, follow these steps:
 - a. Log on to the [ApsaraDB for POLARDB console](#).
 - b. Find the target cluster and click the cluster ID.
 - c. In the **Connection Information** section, view the endpoint and port information.
- **Port:** the internal port of the POLARDB for PostgreSQL cluster.
- **User Name:** the name of the privileged account of the POLARDB for PostgreSQL cluster.
- **Password:** the password of the privileged account of the POLARDB for PostgreSQL cluster.

5. Click **Save**.

6. If the connection information is correct, choose **Servers > Server Name > Databases > postgres**. The connection is successful if the following interface is displayed.



Note:

postgres is the default system database of the POLARDB for PostgreSQL cluster. Do not perform any operation on the database.

10 Cluster management

10.1 Create a POLARDB for PostgreSQL cluster

This topic describes how to create a POLARDB for PostgreSQL cluster in the console.

Prerequisites

You have created an Alibaba Cloud account or created a Resource Access Management (RAM) account.

- Click [here](#) to register an Alibaba Cloud account.
- For more information about how to create and grant permissions to a RAM user, see [Create and authorize a RAM user](#).

Context

A cluster contains one primary node and a maximum of 15 read-only nodes. At least one read-only node is required to implement active-active high availability architecture. A node is a virtual database server, where you can create and manage multiple databases.



Note:

- ApsaraDB for POLARDB supports Virtual Private Cloud (VPC) only. VPC is an isolated network in Alibaba Cloud that is more secure than a classic network.
- To achieve optimal performance, use ApsaraDB for POLARDB with Elastic Compute Service (ECS) and place them in the same VPC. If your ECS instance is created in a classic network, you must migrate it to a VPC.


Procedure

1. Log on to Alibaba Cloud.
 - Click [here](#) to log on with your Alibaba Cloud account.
 - Click [here](#) to log on with your RAM user account. For more information, see [Log on as a RAM user](#).
2. Click Create Cluster to go to the [ApsaraDB for POLARDB purchase page](#).


3. Select **Subscription** or **Pay-As-You-Go**.

- **Subscription:** You must pay for the compute nodes (a primary node and a read-only node) when you create the cluster. Storage consumed by your database is billed in GB/hour increments and the charges are deducted from your account on an hourly basis. The **Subscription** method is more cost-effective if you want to use the new cluster for a long period of time. The longer the subscription period, the greater the discount.
- **Pay-As-You-Go:** This method does not require any upfront payment. Compute nodes and storage consumed by your database are billed on an hourly basis and the charges are deducted from your account on an hourly basis. The **Pay-As-You-Go** method is suitable if you only want to use the new cluster for a short period of time. You can save costs by releasing clusters as needed.

4. Configure the following parameters.

Console section	Parameter	Description
Basic	Region	<p>The region where the cluster resides. You cannot change the region after you confirm your order.</p> <div>  Note: <p>Make sure that you place your cluster in the same VPC as the ECS instance you want to connect to. Otherwise, they cannot communicate through the internal network and achieve optimal performance.</p> </div>
	Primary Availability Zone	<ul style="list-style-type: none"> • The zone of the cluster. Zones are independent physical areas in one region. There are no differences between the zones. • Your cluster and the ECS instance to be connected can be located in the same zone or in different zones.
	Network Type	<ul style="list-style-type: none"> • You do not need to specify this parameter. • ApsaraDB for POLARDB supports VPC only. A VPC is an isolated virtual network with higher security and performance than a classic network.

Console section	Parameter	Description
	VPC VSwitch	<p>Make sure that you place your cluster in the same VPC as the ECS instance you want to connect to. Otherwise, they cannot communicate through the internal network and achieve optimal performance.</p> <ul style="list-style-type: none"> If you have created a VPC that meets your network plan, select the VPC. For example, if you have created an ECS instance and the VPC where it resides meets your network plan, select this VPC. Alternatively, use the default VPC and VSwitch. <ul style="list-style-type: none"> Default VPC: <ul style="list-style-type: none"> It is a unique VPC in your selected zone. The network mask for a default VPC has 16 bits, such as 172.31.0.0/16, providing up to 65,536 internal IP addresses. It is not included in the total number of VPCs that you can create. Default VSwitch: <ul style="list-style-type: none"> It is a unique VSwitch in your selected zone. The network mask for a default VSwitch has 20 bits, such as 172.16.0.0/20, providing up to 4,096 private IP addresses. The default VSwitch is not included in the total number of VSwitches that you can create in a VPC. If the default VPC and VSwitch cannot satisfy your requirements, you can create your own VPC and VSwitch.
Instance	Database Engine	<ul style="list-style-type: none"> Fully compatible with MySQL 8.0. Native concurrent queries are supported, performance in specific scenarios (measured by TPC-H test) increase tenfold. . Fully compatible with MySQL 5.6. Compatible with Oracle (Highly compatible).
	Node Specifications	Select the specifications as needed. All ApsaraDB for POLARDB nodes are dedicated, providing stable and reliable performance. For more information, see #unique_28 .

Console section	Parameter	Description
	Number of Nodes	<ul style="list-style-type: none"> You do not need to specify this parameter. By default, the system will create a read-only node that has the same specifications as the primary node. If the primary node fails, the system automatically promotes the read-only node as the primary node, and generate a new read-only node. For more information about read-only nodes, see #unique_29.
	Storage Cost	<p>You do not need to specify this parameter. The system will charge you on an hourly basis based on the actual data usage. For more information, see #unique_28.</p> <div>  Note: You do not need to select a storage capacity when you purchase a cluster. The storage capacity will automatically resize based on your data usage. </div>

5. Specify **Purchase Plan** (only applicable to subscription clusters) and **Number**, and click **Buy Now**.



Note:

A maximum of 50 clusters can be created at a time, which is suitable for business scenarios such as launching multiple gaming servers at a time.

6. On the **Confirm Order** page, confirm your order information, read and accept the **ApsaraDB for POLARDB Subscription Agreement of Service**, and then click **Pay**.

After the payment is completed, the cluster is created in about 10 minutes. The created cluster is displayed in the cluster list.



Note:

- The cluster is unavailable and is still being created if some of the nodes are in the **Running** state. The cluster is available only when the cluster is in the **Running** state.
- Make sure that you have selected the correct region. Otherwise, you cannot view your clusters.

Next step

Create database accounts. For more information, see [Create a database account](#).

Related API operations

API operation	Description
#unique_21	Creates a POLARDB cluster.
#unique_30	Lists POLARDB clusters.
#unique_31	Used to view the attributes of a POLARDB cluster.
#unique_23	Used to query the automatic renewal status of a POLARDB cluster that uses the Subscription billing method.
#unique_22	Used to set the automatic renewal status of a POLARDB cluster that uses the Subscription billing method.

10.2 Configure cluster parameters

This topic describes how to modify parameter values of a cluster in the ApsaraDB for POLARDB console.

Precautions

- You must modify parameter values according to the **Value Range** column on the Parameters page.

Backup and Restore	character_set_server ⓘ	utf8	Yes	utf8	[utf8 latin1 gbk utf8mb4]
Parameters	default_time_zone ⓘ	SYSTEM	Yes	SYSTEM	[SYSTEM -12:00 -11:00 -10:00 -9:00 -8:00 -7:00 -6:00 -5:00 -4:00 -3:00 -2:00 -1:00 +0:00 +1:00 +2:00 +3:00 +4:00 +5:00 +5:30 +6:00 +6:30 +7:00 +8:00 +9:00 +10:00 +11:00 +12:00 +13:00]
Diagnostics and Opti...	loose_polar_log_bin ⓘ	ON_WITH_GTID	Yes	OFF	[ON_WITH_GTID OFF]
Cluster Overview					
Monitoring					

- For some parameters, you must restart all nodes after the parameter values are modified. We recommend that you make appropriate service arrangements before you restart the nodes. Proceed with caution. You can determine whether the modification

of a parameter value requires a node restart according to the value in the **Force Restart** column on the **Parameters** page.

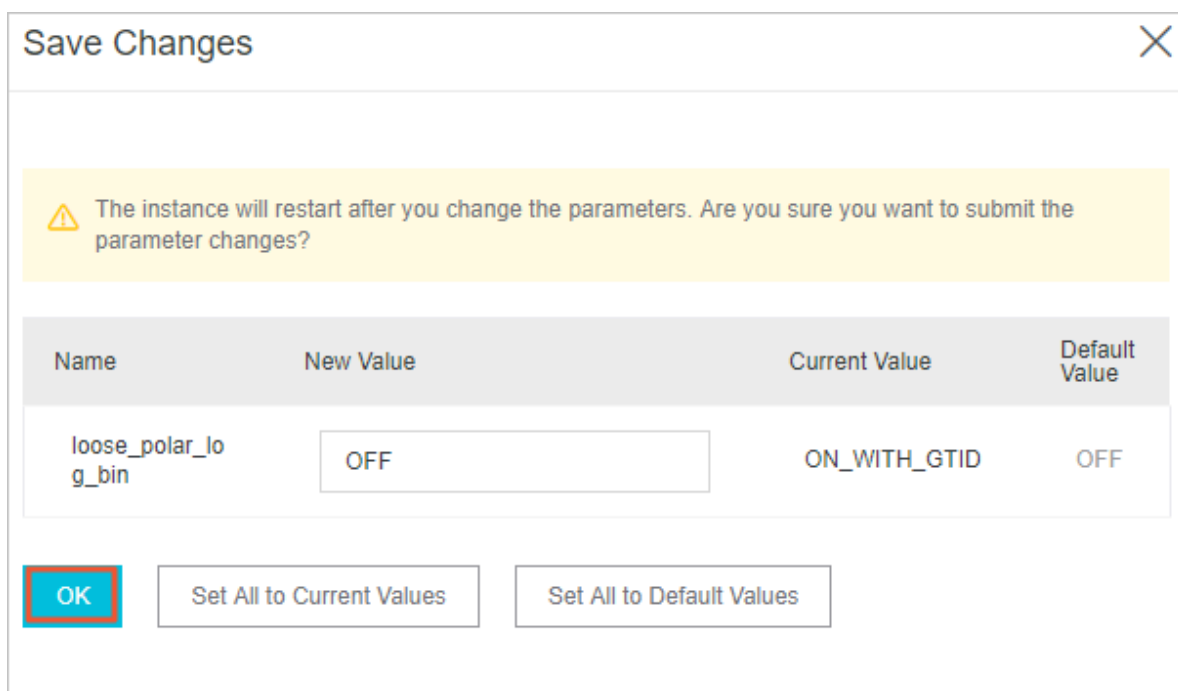
Name	Current Value	Force Restart	Default Value
character_set_filesystem ?	<input type="text" value="binary"/>	No	binary
character_set_server ?	<input type="text" value="utf8"/>	Yes	utf8
default_time_zone ?	<input type="text" value="SYSTEM"/>	Yes	SYSTEM
loose_polar_log_bin ?	<input type="text" value="ON_WITH_GTID"/>	Yes	OFF
autocommit ?	<input type="text" value="ON"/>	No	ON
automatic_sp_privileges ?	<input type="text" value="ON"/>	No	ON

Procedure

1. Log on to the [ApsaraDB for POLARDB console](#).
2. Select a region.
3. Find the target cluster and click the cluster ID in the **Cluster Name** column.
4. In the left-side navigation pane, choose **Settings and Management > Parameters**.
5. Modify the values of one or more parameters in the **Current Value** column, and click **Apply Changes**.

Apply Changes	Undo All	<input type="text" value="Enter a value"/>	<input type="button" value="Q"/>
Name	Current Value	Force Restart	Default Value
character_set_filesystem ?	<input type="text" value="binary"/>	No	binary
character_set_server ?	<input type="text" value="utf8"/>	Yes	utf8
default_time_zone ?	<input type="text" value="SYSTEM"/>	Yes	SYSTEM
loose_polar_log_bin ?	<input type="text" value="ON_WITH_GTID"/>	Yes	OFF

6. In the **Save Changes** dialog box that appears, click **OK**.



The image shows a 'Save Changes' dialog box with a yellow warning banner at the top stating: 'The instance will restart after you change the parameters. Are you sure you want to submit the parameter changes?'. Below the banner is a table with four columns: 'Name', 'New Value', 'Current Value', and 'Default Value'. The table contains one row for the parameter 'loose_polar_log_bin', with 'New Value' set to 'OFF', 'Current Value' set to 'ON_WITH_GTID', and 'Default Value' set to 'OFF'. At the bottom of the dialog, there are three buttons: 'OK' (highlighted with a red box), 'Set All to Current Values', and 'Set All to Default Values'.

Name	New Value	Current Value	Default Value
loose_polar_log_bin	OFF	ON_WITH_GTID	OFF

Related API operations

API operation	Description
#unique_33	Views cluster parameters.
#unique_34	Modifies the values of cluster parameters.

10.3 Change the cluster specifications

This topic describes how to change the specifications of your cluster to meet business requirements.

ApsaraDB for POLARDB supports capacity scaling in three dimensions:

- Vertical scaling of computing power

You can upgrade or downgrade the specifications of a cluster. This topic describes the details.

- Horizontal scaling of computing power

You can add or delete read-only nodes. For more information, see [Add or remove a read-only node](#).

- Horizontal scaling of storage capacity

The storage capacity is provisioned in a serverless model. As your data increases in size, the storage is automatically expanded.

You can upgrade or downgrade the specifications of an ApsaraDB for POLARDB cluster. It takes only 5 minutes to 10 minutes for the new specifications of each node to take effect.

Specification change fees

For more information, see [#unique_36](#).

Prerequisites

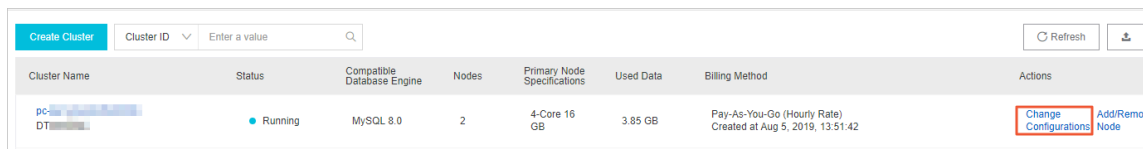
You can only change cluster specifications when the cluster does not have pending specification changes.

Precautions

- Specification upgrades or downgrades only apply to clusters. You cannot change the specifications of a node.
- Specification upgrades or downgrades do not affect the existing data in the cluster.
- We recommend that you modify cluster specifications during off-peak periods. During a specification upgrade or downgrade, the ApsaraDB for POLARDB service will be disconnected for a few seconds and some of the functions will be disabled. You will need to reconnect from your applications after the service is disconnected.

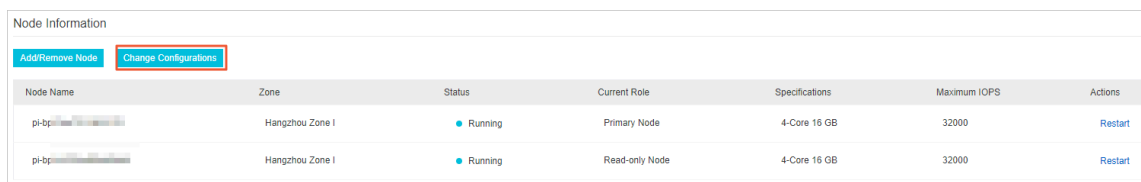
Procedure

1. Log on to the [ApsaraDB for POLARDB console](#).
2. In the upper-left corner of the page, select the region.
3. Go to the **Change Configurations** page. Perform the operation by using either of the following methods:
 - Find the target cluster and click **Change Configurations** in the **Actions** column corresponding to the target cluster.



Cluster Name	Status	Compatible Database Engine	Nodes	Primary Node Specifications	Used Data	Billing Method	Actions
pc- DT	Running	MySQL 8.0	2	4-Core 16 GB	3.85 GB	Pay-As-You-Go (Hourly Rate) Created at Aug 5, 2019, 13:51:42	Change Configurations Add/Remove Node

- Find the target cluster, click the cluster ID, and then click **Change Configurations** in the **Node Information** section.



Node Name	Zone	Status	Current Role	Specifications	Maximum IOPS	Actions
pi-bp	Hangzhou Zone I	Running	Primary Node	4-Core 16 GB	32000	Restart
pi-bp	Hangzhou Zone I	Running	Read-only Node	4-Core 16 GB	32000	Restart

4. Select **Upgrade** or **Downgrade** and click **OK**.

Change Configurations (Subscription) ✕

The current billing method is **Subscription**. The following configuration change plans are available.

☒ Upgrade

You can immediately upgrade the specifications of the POLARDB cluster within the current lifecycle. The configuration changes will take effect in 10 minutes. An up to 30-second disconnection from each endpoint will occur during the upgrade process. Make sure that the connected application has a reconnection mechanism. See: [Change configurations](#)

☐ Temporary Upgrade

You can temporarily upgrade the specifications of the POLARDB cluster to meet the business peak requirements in the specified validity period (generally less than 7 days). You need to pay upfront only for the validity period before you perform the temporary upgrade. **You cannot add nodes or use the regular method to change configurations within the validity period. If necessary, add nodes before you perform the temporary upgrade. We also recommend that you upgrade the cluster to the highest possible specifications to avoid repeated upgrades.** During the temporary upgrade process and the configuration restore process, disconnections will occur. Make sure that the connected application has a reconnection mechanism. See: [Temporary upgrades](#)

☐ Downgrade

You can immediately downgrade the specifications of the POLARDB cluster within the current lifecycle. The configuration changes will take effect in 10 minutes. An up to 30-second disconnection from each endpoint will occur during the upgrade process. Make sure that the connected application has a reconnection mechanism. See: [Change configurations](#) and [Refund rules for configuration downgrade](#)

OK

Cancel

5. Select a specification.



Note:

All nodes in a cluster have the same specifications.

6. Read and agree to the service agreement by selecting the check box, and click **Pay** to complete the payment.



Note:

It takes about ten minutes for the new specifications to take effect.

Related API operations

API	Description
#unique_37	Changes the specifications of a POLARDB cluster.

10.4 Add or remove a read-only node

You can manually add or remove read-only nodes after creating an ApsaraDB for POLARDB cluster. An ApsaraDB for POLARDB cluster can contain a maximum of 15 read-only nodes. The cluster must have at least one read-only node to guarantee high availability. All nodes in a cluster have the same specifications.

Billing

The billing methods for nodes added to an existing cluster are as follows:

- If nodes are added to a subscription cluster, the nodes are billed as subscription nodes.
- If nodes are added to a pay-as-you-go cluster, the nodes are billed as pay-as-you-go nodes.



Note:

- Read-only nodes that you purchase in either subscription or pay-as-you-go mode can be released at any time. After they are released, the system will [refund or stop billing](#).
- The added nodes are only charged based on the node specifications. For more information, see [#unique_28](#). The storage fee is charged based on the actual data volume, regardless of the number of nodes.

Precautions

- You can only add or remove read-only nodes when the cluster does not have pending specification changes.
- To avoid misoperations, only one read-only node can be added or removed at a time. You must perform the add or remove operation for each node.
- It takes about 5 minutes to add or remove a node.

Add a read-only node

1. Log on to the [ApsaraDB for POLARDB](#) console.
2. Select a region.

3. Go to the **Add/Remove Node** page. Perform the operation by using either of the following methods:

- Find the target cluster and click **Add/Remove Node** in the **Actions** column.

Clusters

Create Cluster Cluster ID Enter a value Refresh

Cluster Name	Status	Compatible Database Engine	Nodes	Primary Node Specifications	Used Data	Billing Method	Actions
...	Running	MySQL 5.6	2	16-Core 128 GB	2.88 GB	Subscription Expires at Aug 28, 2019, 00:00:00	Change Configurations Add/Remove Node

- Find the target cluster, click the cluster ID, and then click **Add/Remove Node** in the **Node Information** section.

Node Information

Add/Remove Node Change Configurations

Node Name	Zone	Status	Current Role	Specifications	Maximum IOPS	Actions
pi-bp-...	Hangzhou Zone I	Running	Primary Node	16-Core 128 GB	128000	Restart
pi-bp-...	Hangzhou Zone I	Running	Read-only Node	16-Core 128 GB	128000	Restart

4. Select **Add Node** and click **OK**.

Add/Remove Node

The current billing method is **Subscription**. The following configuration change plans are available.


☒ **Add Node**

You can immediately add a database compute node to a POLARDB cluster within the current lifecycle. It takes about 5 minutes to add a node. The entire process does not affect the databases. You can use the default cluster endpoint to automatically identify the new node and load balance requests to the new node to achieve load balancing without modifying the application configurations. See: [Add a node](#) and [Pricing for adding a node to a subscription cluster](#)

☐ **Remove Node**

You can immediately remove a database compute node from the POLARDB cluster within the current lifecycle. All connections on the removed node will be terminated, but other nodes will not be affected. You can use the cluster endpoint to automatically ignore the failed node without modifying the application configurations. See: [Remove a node](#) and [Refund rules for removing a node from a subscription cluster](#)

OK
Cancel

5. Click the  icon to add a read-only node. Read and agree to the service agreement by selecting the check box, and click **Pay** to complete the payment.

Remove a read-only node



1. Log on to the [ApsaraDB for POLARDB console](#).
2. Select a region.
3. Go to the **Add/Remove Node** page. Perform the operation by using either of the following methods:
 - Find the target cluster and click **Add/Remove Node** in the **Actions** column.

Clusters

Create Cluster Cluster ID Enter a value

Cluster Name	Status	Compatible Database Engine	Nodes	Primary Node Specifications	Used Data	Billing Method	Actions
mysql-cluster-galaxy-1	Running	MySQL 5.6	2	16-Core 128 GB	2.88 GB	Subscription Expires at Aug 28, 2019, 00 00 00	Change Configurations Add/Remove Node

- Find the target cluster, click the cluster ID, and then click **Add/Remove Node** in the Node Information section.

Node Information						
Add/Remove Node Change Configurations						
Node Name	Zone	Status	Current Role	Specifications	Maximum IOPS	Actions
pi-bp- 	Hangzhou Zone I	● Running	Primary Node	16-Core 128 GB	128000	Restart
pi-bp- 	Hangzhou Zone I	● Running	Read-only Node	16-Core 128 GB	128000	Restart

4. Select **Remove Node** and click **OK**.

Add/Remove Node

×

The current billing method is **Subscription**. The following configuration change plans are available.

☐ Add Node


You can immediately add a database compute node to a POLARDB cluster within the current lifecycle. It takes about 5 minutes to add a node. The entire process does not affect the databases. You can use the default cluster endpoint to automatically identify the new node and load balance requests to the new node to achieve load balancing without modifying the application configurations. See: [Add a node](#) and [Pricing for adding a node to a subscription cluster](#)

☒ Remove Node

You can immediately remove a database compute node from the POLARDB cluster within the current lifecycle. All connections on the removed node will be terminated, but other nodes will not be affected. You can use the cluster endpoint to automatically ignore the failed node without modifying the application configurations. See: [Remove a node](#) and [Refund rules for removing a node from a subscription cluster](#)

OK

Cancel

5. Click the  icon next to the node that you want to remove. In the dialog box that appears, click **OK**.



Note:

You must keep at least one read-only node in the cluster to guarantee high availability.

6. Read and agree to the service agreement by selecting the check box, and click **OK**.

Related API operations

API operation	Description
#unique_38	Adds a node to an ApsaraDB for POLARDB cluster.
#unique_37	Changes the specifications of a node in an ApsaraDB for POLARDB cluster.
#unique_39	Restarts a node in an ApsaraDB for POLARDB cluster.

API operation	Description
#unique_40	Removes a node from an ApsaraDB for POLARDB cluster.

10.5 Set the maintenance window

This topic describes how to set the maintenance window for an ApsaraDB for POLARDB cluster. To guarantee the stability of ApsaraDB for POLARDB, the backend system performs maintenance operations on the clusters from time to time. We recommend that you set the maintenance window within the off-peak hours of your business to minimize the impact on the business during the maintenance process.

Important notes

- Before the maintenance is performed, ApsaraDB for POLARDB sends SMS messages and emails to contacts listed in your Alibaba Cloud account.
- To guarantee stability during the maintenance process, clusters first enter the **Under Maintenance** state before the preset maintenance window arrives on the day of maintenance. When a cluster is in this state, normal data access to the database is not affected. However, except for the account management, database management, and IP address whitelisting functions, other services concerning changes (such as common operations like upgrade, degrade, and restart) are unavailable in the console of this cluster. Query services such as performance monitoring are still available.
- Within the maintenance window of a cluster, the cluster may experience one or two disconnections. Make sure that your application can automatically reconnect to the cluster. The cluster restores to normal immediately after the disconnection occurs.

Procedure

1. Log on to the [ApsaraDB for POLARDB console](#).
2. Select a region.
3. Find the target cluster and click the cluster ID in the **Cluster Name** column.

4. In the **Basic Information** section on the **Basics** page, click **Modify** next to **Maintenance Window**.

Basic Information			
Cluster ID	pc-	Cluster Name	pc- Edit
Region	China (Hangzhou)	Zones	Hangzhou Zone G (Primary), Hangzhou Zone I
Compatible Database Engine	MySQL 5.6	Status	● Running
VPC	vpc-	VSwitch	vsw-
Maintenance Window	02:00-03:00 Modify		

5. In the **Modify Maintenance Window** dialog box that appears, select a maintenance window for the cluster and click **Submit**.

APIs

API	Description
CreateDBCluster	Creates an ApsaraDB for POLARDB cluster.
ModifyDBClusterMaintainTime	Modifies the maintenance window for an ApsaraDB for POLARDB cluster.

10.6 Restart a node

This topic describes how to manually restart a node when the number of connections exceeds the threshold or any performance issue occurs on the node. Restarting a node causes service interruptions. We recommend that you make appropriate service arrangements before you restart the nodes. Proceed with caution

Procedure

1. Log on to the [ApsaraDB for PolarDB console](#).
2. Select a region.
3. Find the target cluster and click the cluster ID in the **Cluster Name** column.
4. In the **Node Information** section on the **Basics** page, find the node to be restarted.
5. Click **Restart** in the **Actions** column of the node.

Add/Remove Node		Change Configurations				
Node Name	Zone	Status	Current Role	Specifications	Maximum IOPS	Actions
pi-bp-	Hangzhou Zone G	● Running	Primary Node	2-Core 4 GB	8000	Restart
pi-bp-	Hangzhou Zone G	● Running	Read-only Node	2-Core 4 GB	8000	Restart
pi-bp-	Hangzhou Zone G	● Running	Read-only Node	2-Core 4 GB	8000	Restart

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

Related API operations

API operation	Description
#unique_39	Restarts a database node.

10.7 Release a cluster

You can manually release a pay-as-you-go cluster according to your business requirements

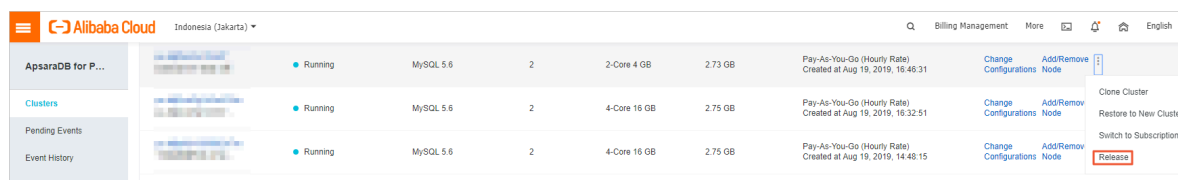
.

Precautions

- A subscription cluster cannot be manually released and will be automatically released after the subscription expires.
- A pay-as-you-go cluster can only be manually released when it is in the Running state.
- All the data in your cluster will be deleted when the cluster is released. Proceed with caution.
- This function is used to release a cluster, including all nodes in the specified cluster. To release one read-only node, see [Add or remove a read-only node](#).
- You can switch the billing method of a cluster from pay-as-you-go to subscription. For more information, see [Change the billing method from pay-as-you-go to subscription](#).

Procedure

1. Log on to the [ApsaraDB for POLARDB console](#).
2. Select a region.
3. Find the target cluster. In the **Actions** column corresponding to the cluster, click ... > **Release**.



4. In the message that appears, click **OK**.

Related API operations

API	Description
#unique_30	Views the list of POLARDB clusters.
#unique_45	Deletes a POLARDB cluster.

10.8 Clone a cluster


This topic describes how to clone an ApsaraDB for PolarDB cluster. You can create an ApsaraDB for PolarDB cluster that is the same as an existing ApsaraDB for PolarDB cluster by cloning the data of the existing one. The data includes the account information, but excludes parameter settings of the cluster.

The data generated before the execution of the clone action is cloned. When cloning starts, the newly written data will not be cloned.

Procedure

1. Log on to the [ApsaraDB for PolarDB console](#).
2. Select the region where the target cluster is located.
3. Find the cluster you want to clone. In the **Actions** column of the cluster, click the **More** icon, and then select **Restore to New Cluster**.
4. On the page that appears, set the parameters. The following table describes the parameters.

Parameter	Description
Clone Source Type	The type of the clone source. Select Current Cluster .
Region	The region where the cluster resides. The region of the new cluster is the same as that of the source cluster and cannot be modified.
Primary Availability Zone	<ul style="list-style-type: none">• The zone of the new cluster. A zone is an independent physical area located within a region. There are no substantive differences between the zones.• You can deploy the ApsaraDB for PolarDB cluster and ECS instance in the same zone or in different zones.
Network Type	<ul style="list-style-type: none">• The type of the network. Use the default setting.• ApsaraDB for PolarDB supports Virtual Private Cloud (VPC) networks only. A VPC is an isolated virtual network with higher security and performance than a classic network.

Parameter	Description
VPC Vswitch	<p>The VPC and VSwitch of the new cluster. Select a VPC and a VSwitch from the corresponding drop-down lists, or create a VPC and a VSwitch.</p> <div>  Note: Make sure that you place your ApsaraDB for PolarDB cluster and the ECS instance to be connected in the same VPC. Otherwise, they cannot intercommunicate through the internal network and achieve optimal performance. </div>
Database Engine	The database engine of the new cluster. Use the default setting.
Node Specification	The node specification of the new cluster. Select a specification according to your needs. Clusters with different specifications have different storage capacity and performance. For more information, see #unique_28 .
Number Nodes	The number of nodes in the new cluster. Use the default setting. By default, the system creates a read-only node with the same specification as the primary node.
Cluster Name	<ul style="list-style-type: none"> Optional. The name of the new cluster. The system will automatically create a name for your ApsaraDB for PolarDB cluster if you leave it blank. You can rename the cluster after it is created.
Purchase Plan	The subscription duration of the new cluster. This parameter is valid only for subscription clusters.
Number	The number of clusters. The default value 1 is used and cannot be modified.

5. Read the **ApsaraDB for PolarDB Agreement of Service**, select the check box to agree to it, and then complete the payment.

10.9 Upgrade the minor version

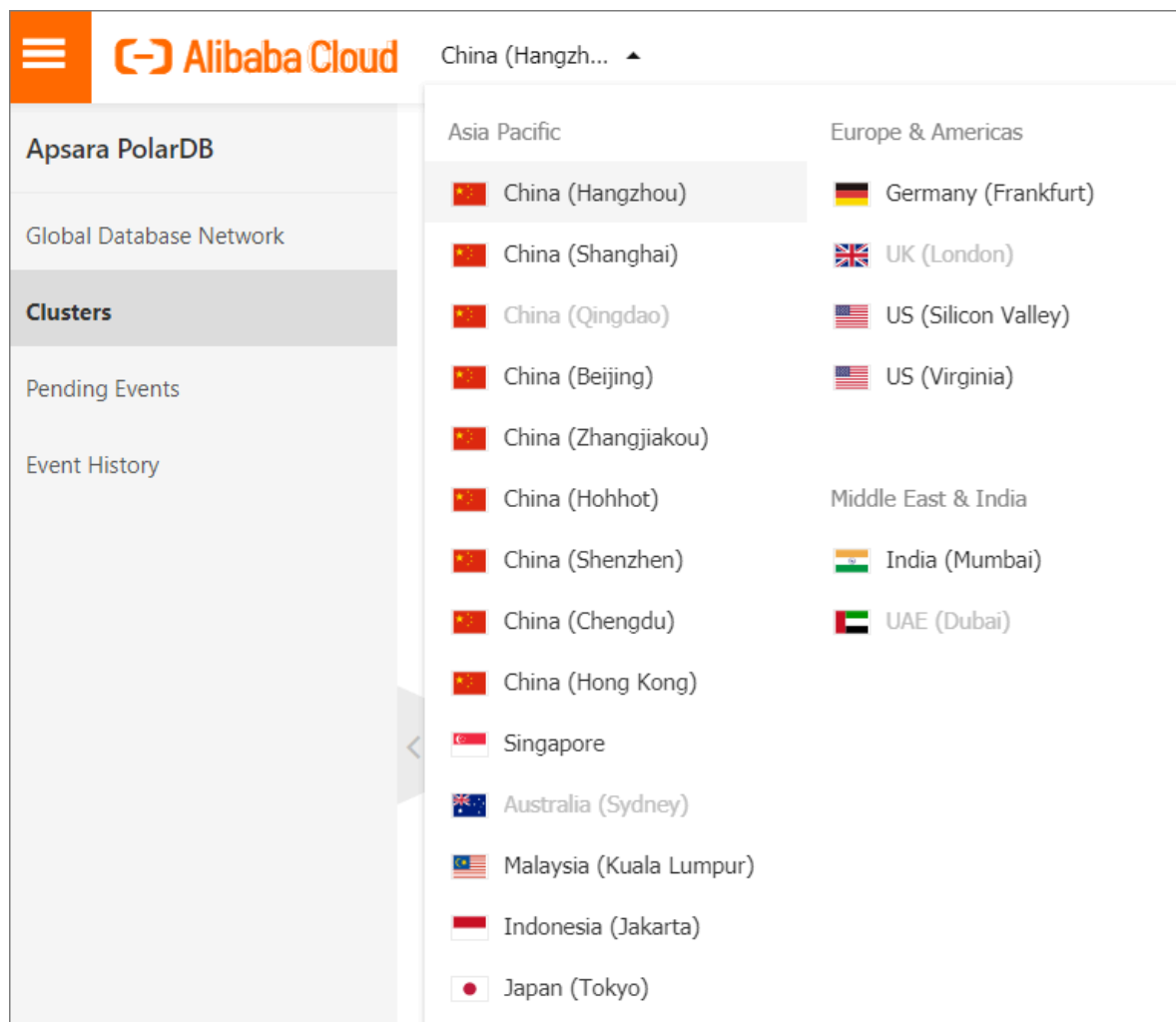
You can manually upgrade the minor kernel version of ApsaraDB PolarDB for PostgreSQL. The upgrades improve performance, provide new feature, or fix bugs.

Precautions

- Upgrading the kernel minor version will restart the instance. We recommend that you perform the upgrade during off-peak hours or make sure that your applications can automatically reconnect to the instance.
- You cannot downgrade the minor version after an upgrade.

Procedure

1. Log on to the [PolarDB console](#).
2. In the upper-left corner of the page, select the region where the PolarDB cluster is located.

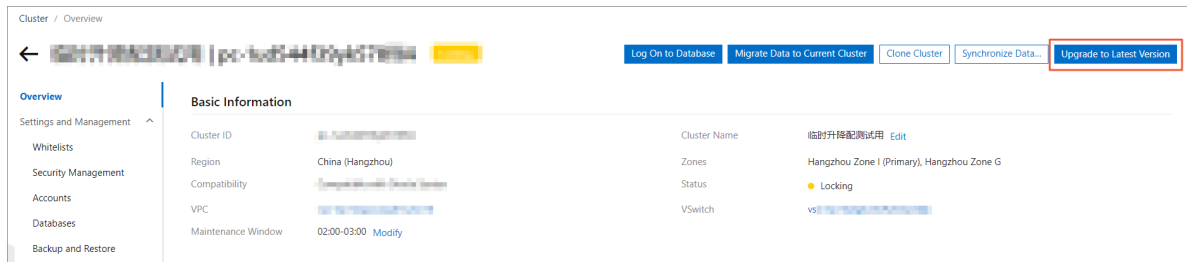


3. Find the target cluster and click the cluster ID.
4. In **Basic Information**, click **Upgrade to Latest Version**.



Note:

If your cluster kernel version is already the latest, the **Upgrade to Latest Version** button is not displayed.



5. In **Upgrade to Latest Version** dialog box, click **OK**.



Note:

During the upgrade, services may be interrupted for about 60 seconds. Make sure that your applications can automatically reconnect to the instance.

10.10 Switch workloads from writer nodes to reader nodes

An Apsara PolarDB cluster consists of one writer node and one or more reader nodes. This topic describes how to switch your workloads from a writer node to a reader node. If a failure occurs on a writer node, the system can automatically perform a failover. You may want to manually switch your workloads from the writer node to a reader node to run a disaster recovery drill and to specify a certain reader node as the writer node.

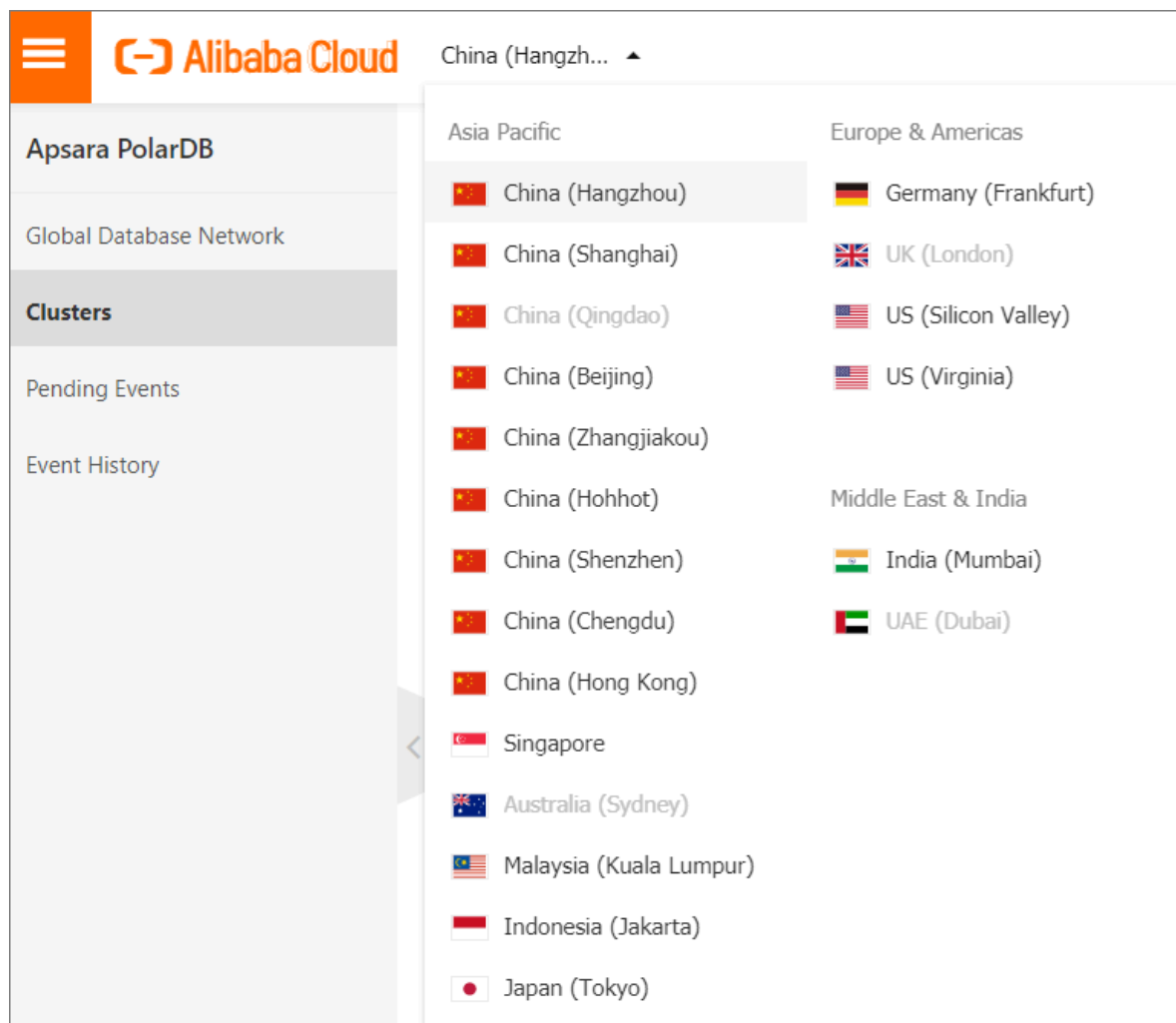
Considerations

An Apsara PolarDB cluster may be disconnected for approximately 30 seconds during switchover. We recommend that you perform the switchover during off-peak hours and make sure that your application can automatically reconnect to the Apsara PolarDB cluster.

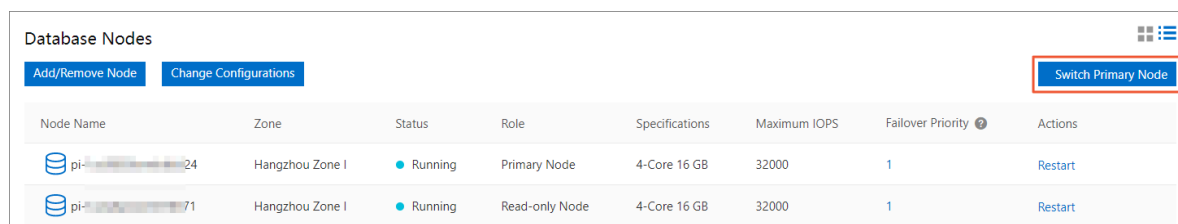
Manual switchover

1. Log on to the [PolarDB console](#).

2. In the upper-left corner of the page, select the region where the PolarDB cluster is located.



3. Find the target cluster and click the cluster ID.
4. In the upper-right corner of the **Node Information** section, click **Switch Primary Node**.



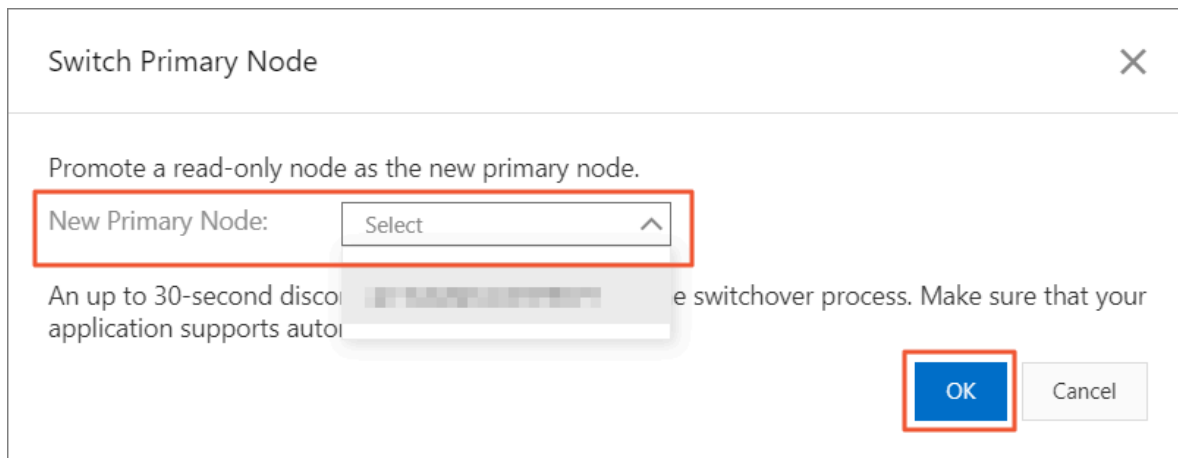
5. In the **Switch Primary Node** dialog box that appears, select a new writer node from the **New Primary Node** drop-down list, and click **OK**.



Note:

If you do not select a new writer node from the **New Primary Node** drop-down list, the system automatically promotes a reader node with the highest failover priority to the

new writer node. The cluster may be disconnected for approximately 30 seconds during switchover. Make sure that your application can automatically reconnect to the cluster.



The image shows a 'Switch Primary Node' dialog box. At the top, it says 'Promote a read-only node as the new primary node.' Below this is a label 'New Primary Node:' followed by a dropdown menu with 'Select' and an upward arrow. A red rectangle highlights this dropdown. Below the dropdown, there is a warning message: 'An up to 30-second disconnect will occur during the switchover process. Make sure that your application supports automatic reconnection.' At the bottom right, there are two buttons: 'OK' (highlighted with a red rectangle) and 'Cancel'.

Automatic failover between the writer node and reader nodes

An Apsara PolarDB cluster runs in an active-active high-availability architecture. This architecture allows for automatic failovers between the writer node and reader nodes.

Each node of the cluster has a failover priority. This priority determines the probability that the system promotes this node to the writer node if a failover occurs. If multiple nodes have the same failover priority, they all have the same probability of being promoted to the writer node.

The system follows these steps to promote a reader node to the writer node:

1. Find all reader nodes that can be promoted to the writer node.
2. Select one or more reader nodes that have the highest failover priority.
3. If the failover to the first node fails due to network or replication errors, the system tries to switch the workloads to the next available node. The system continues the failover until one of the available nodes is promoted to the writer node.

Related API operations

Operation	Description
#unique_49	Manually switches your workloads from the writer node to a specified reader node in an Apsara PolarDB cluster.

11 Account management

11.1 Overview

Console accounts

You can use the following accounts to log on to the console:

- Alibaba Cloud account

This account allows flexible control of all your Alibaba Cloud resources and is used for billing purposes. You must create an Alibaba Cloud account before you can purchase any Alibaba Cloud services.

- RAM user account

Optional. You can create and manage RAM user accounts in the Resource Access Management (RAM) console to share resources to multiple users. A RAM user account does not have ownership over any resources. Charges incurred are billed to the parent Alibaba Cloud account.

Database cluster accounts

You can use the following accounts to log on to your database cluster. For more information, see [Create a database account](#).

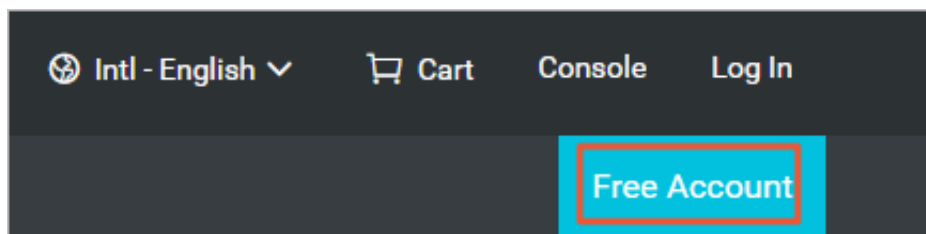
Account type	Description
Privileged account	<ul style="list-style-type: none">• You can only create and manage the account in the console.• A cluster can have only one privileged account. A privileged account can manage all standard accounts and databases.• A privileged account has more permissions, which allows you to perform more management operations. For example, you can grant permissions of querying different tables to different users.• The account has all permissions on all databases in the cluster.• The account can disconnect any account from the instance.

11.2 Register and log on to an Alibaba Cloud account

Register an Alibaba Cloud account

You can register an Alibaba Cloud account by using one of the following two methods:

- On the [Alibaba Cloud website](#), click **Free Account** in the upper-right corner.

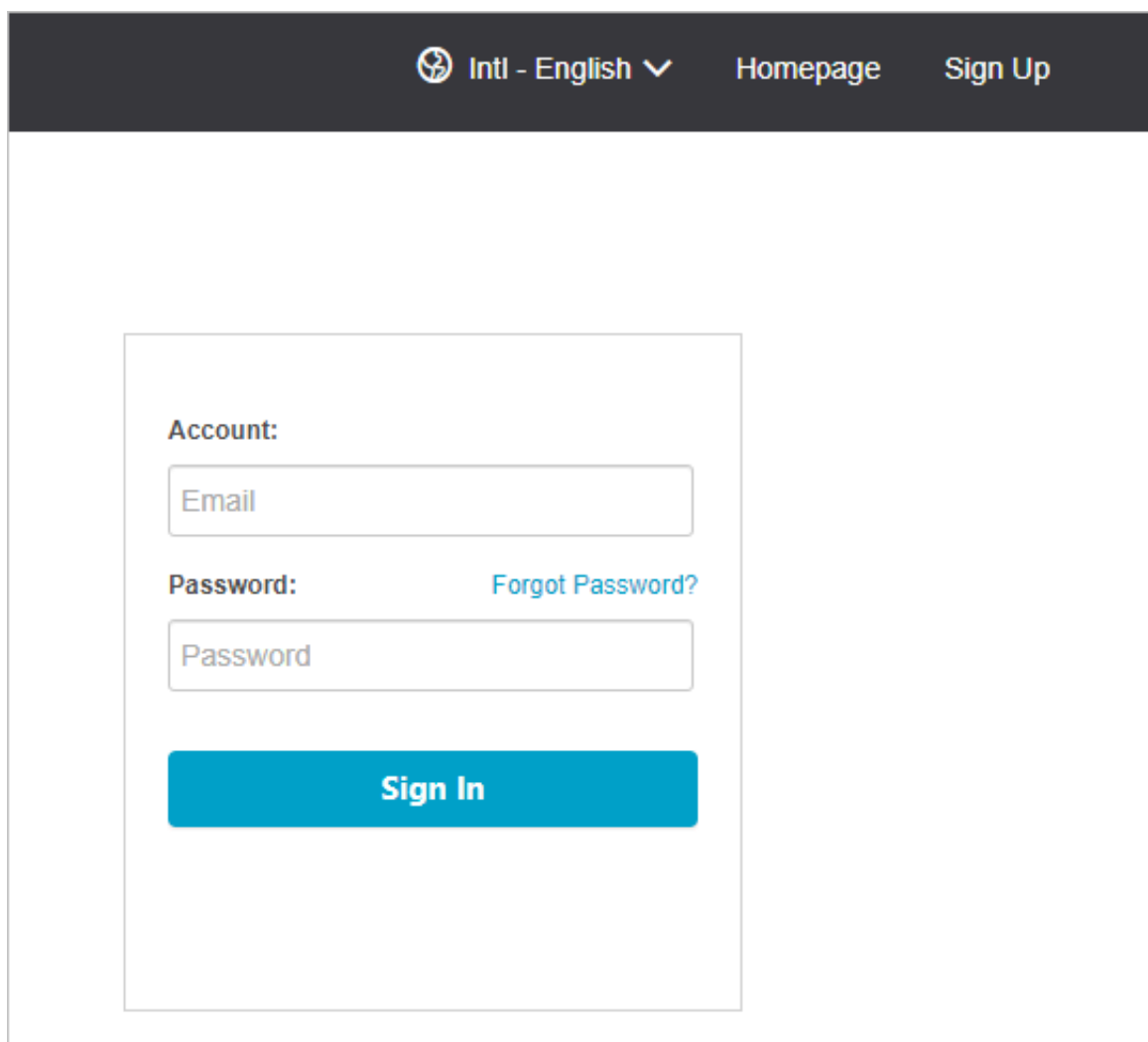


- Visit the [Alibaba Cloud account registration page](#).

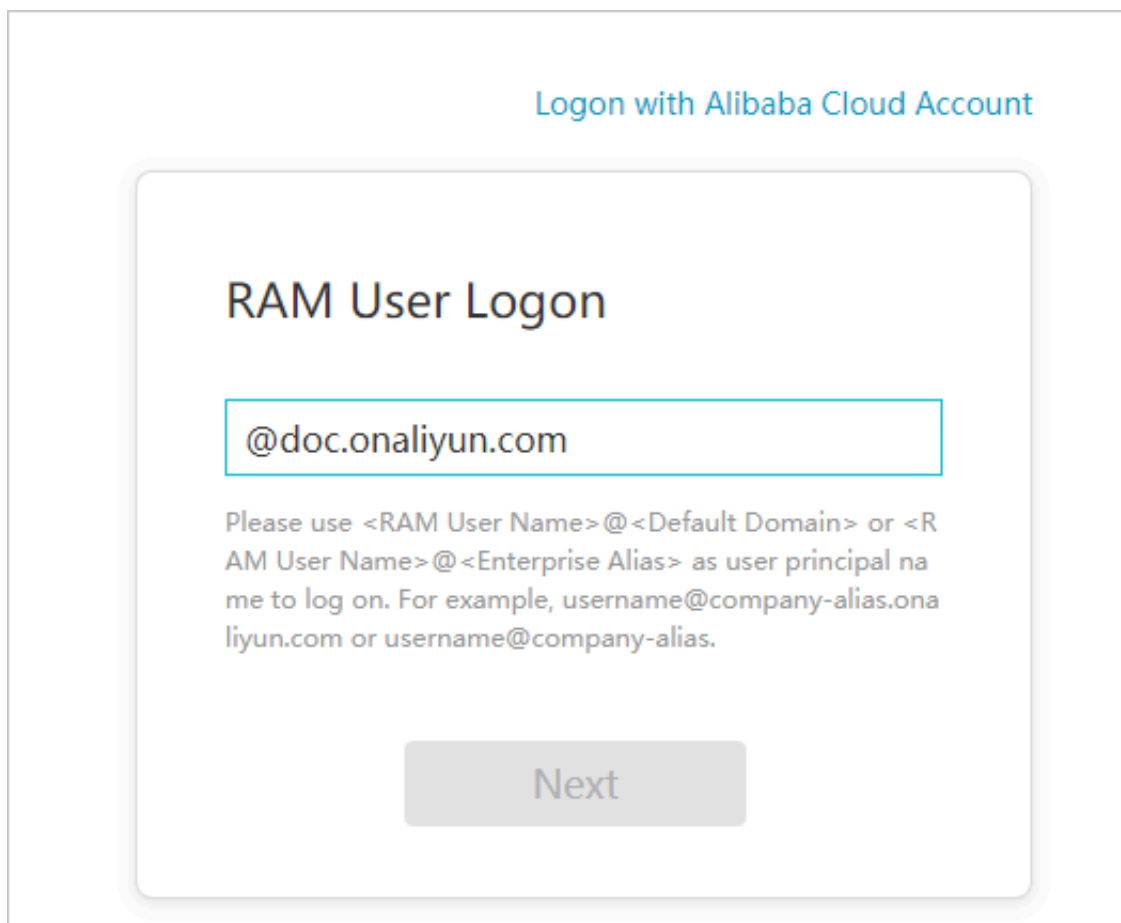
Log on to your Alibaba Cloud account.

Your Alibaba Cloud account and RAM user account have different login pages.

- The logon page for Alibaba Cloud accounts is [Alibaba Cloud accounts](#).

A screenshot of the Alibaba Cloud account login page. At the top is a dark grey header bar with a globe icon, 'Intl - English' with a dropdown arrow, 'Homepage', and 'Sign Up'. The main content area is white and contains a login form. The form has two input fields: 'Account:' with a placeholder 'Email' and 'Password:' with a placeholder 'Password'. To the right of the 'Password:' label is a blue link that says 'Forgot Password?'. Below the input fields is a large blue button with white text that says 'Sign In'.

- The logon page for RAM users is [RAM User Logon](#).



Logon with Alibaba Cloud Account

RAM User Logon

Please use <RAM User Name>@<Default Domain> or <RAM User Name>@<Enterprise Alias> as user principal name to log on. For example, username@company-alias.onaliyun.com or username@company-alias.

Next

11.3 Create and authorize a RAM user

This topic describes how to create and authorize a Resource Access Management (RAM) user. You can use your Alibaba Cloud account to access your ApsaraDB for PolarDB resources. If you want to share the resources under your Alibaba Cloud account with other users, create and authorize a RAM user. The RAM user can then be used to access specified resources.

Create a RAM user

1. You can use an Alibaba Cloud account or a RAM user to create one or more RAM users. First, log on to the RAM console.
 - Click [Alibaba Cloud account Logon](#) to log on with your Alibaba Cloud account.
 - Click [RAM User Logon](#) to log on with your RAM user.



Note:

Enter the RAM username in the format of RAM username@enterprise alias on the logon page.

2. In the left-side navigation pane, click **Users** under **Identities**.
3. Click **Create User**.

**Note:**

To create multiple RAM users at a time, click **Add User**.

4. Specify the **Logon Name** and **Display Name** parameters.
5. In the **Access Mode** section, select **Console Password Logon**.
6. Under **Console Password Logon**, select **Automatically Generate Default Password** or **Custom Logon Password**.
7. Under **Password Reset**, select **Required at Next Logon** or **Not Required**.
8. Under **Multi-factor Authentication**, select **Not Required**.
9. Click **OK**.

Grant permission to a RAM user on the Grants page

1. In the left-side navigation pane, click **Grants** under **Permissions**.
2. Click **Grant Permission**.
3. Under **Principal**, enter the username, and click the target RAM user.
4. In the **Policy Name** column, select the target policies by clicking the corresponding rows.

**Note:**

You can click **X** in the section on the right side of the page to delete the selected policy.

5. Click **OK**.
6. Click **Finished**.

Grant permission to a RAM user on the Users page

1. In the left-side navigation pane, click **Users** under **Identities**.
2. In the **User Logon Name/Display Name** column, find the target RAM user.
3. Click **Add Permissions**. On the page that appears, the principal is automatically filled in.
4. In the **Policy Name** column, select the target policies by clicking the corresponding rows.

**Note:**

You can click **X** in the section on the right side of the page to delete the selected policy.

5. Click **OK**.

6. Click **Finished.****Log on as a RAM user**

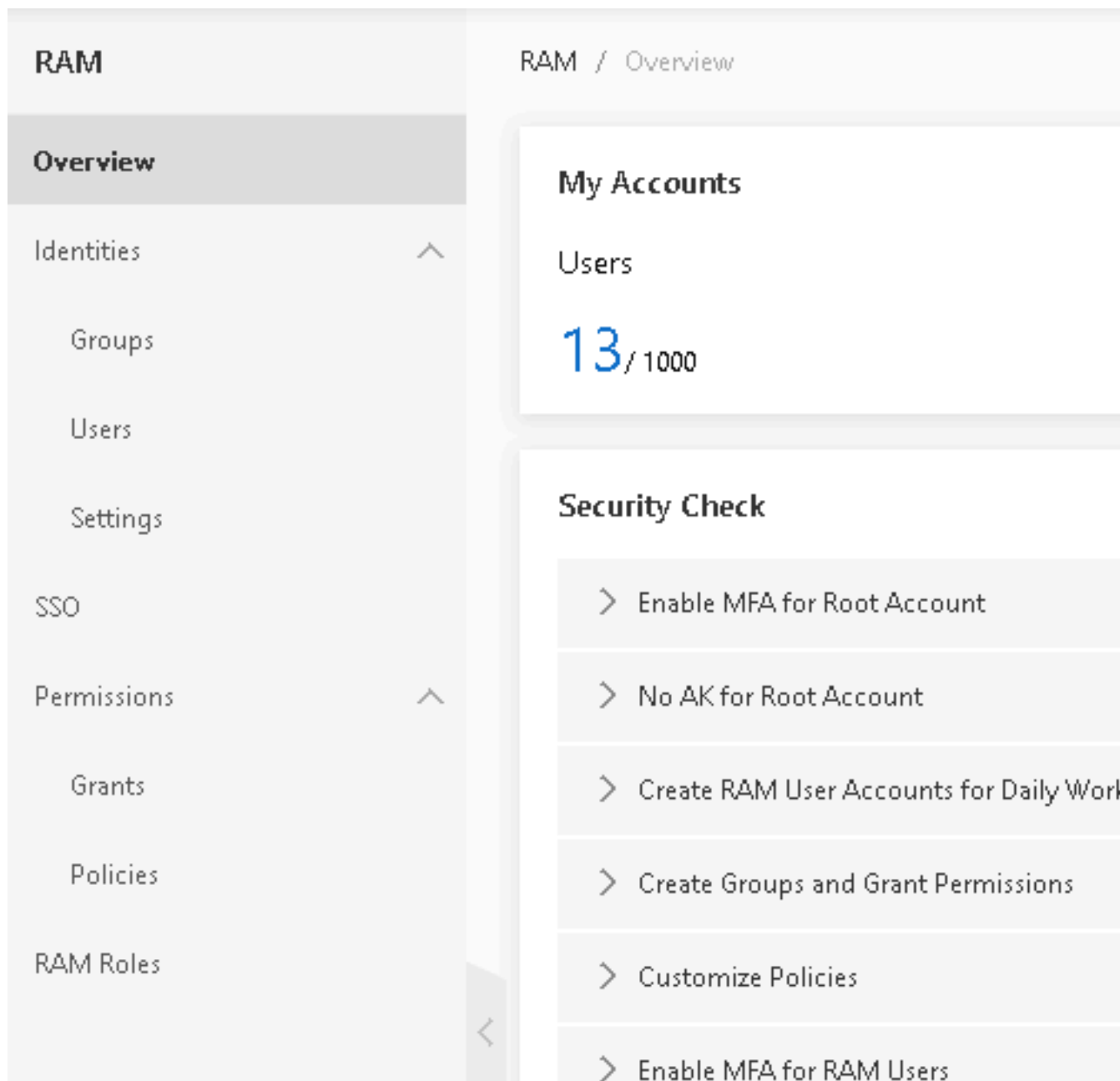
Prerequisites: You must complete the preceding authorization procedures.

You can log on as a RAM user at the following addresses:

- Universal logon address: [RAM User Logon](#).

If you log on at the universal logon address, you must enter the RAM username and company alias manually. The address format is `RAM username@company alias`.

- Dedicated logon address: You can view the logon address dedicated to your RAM users in the [RAM console](#).



The system will enter your company alias automatically if you log on using this dedicated address. You only need to enter the RAM username.

More actions

You can also add a RAM user to a group, assign roles to a RAM user, and authorize a user group or roles. For more information, see [RAM User Guide](#).

11.4 Create a database account

This topic describes how to create a database account and reset permissions of the account.

POLARDB for PostgreSQL supports only privileged accounts and allows you to manage the accounts in the console.

Account type	Description
Privileged account	<ul style="list-style-type: none">You can only create and manage the account in the console.A cluster can have only one privileged account. A privileged account can manage all standard accounts and databases.A privileged account has more permissions, which allows you to perform more management operations. For example, you can grant permissions of querying different tables to different users.The account has all permissions on all databases in the cluster.The account can disconnect any account from the instance.

Create a privileged account

1. Log on to the [ApsaraDB for POLARDB console](#).
2. Find the target cluster and click the cluster ID.
3. In the left-side navigation pane, click **Accounts**.
4. Click **Create Account**.
5. In the dialog box that appears, configure the following parameters.

Parameter	Description
Account Name	Enter the account name. The requirements are as follows: <ul style="list-style-type: none">It must start with a lowercase letter and end with a letter or digit.It can contain lowercase letters, digits, and underscores (_).It must be 2 to 16 characters in length.It cannot be system reserved usernames, such as root and admin.
Account Type	Select Privileged Account .

Parameter	Description
Password	Enter the password of the privileged account. The requirements are as follows: <ul style="list-style-type: none"> The password must contain at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters. The password must be 8 to 32 characters in length. Special characters include ! @ # \$ % ^ & * () _ + - =
Confirm Password	Enter the password again.
Description	Enter related information about the account for account management. The requirements are as follows: <ul style="list-style-type: none"> It cannot start with http:// or https://. The description must start with an uppercase or lowercase letter. The description can contain uppercase or lowercase letters, digits, underscores (_), and hyphens (-). The description must be 2 to 256 characters in length.

Reset permissions of a privileged account

If the privileged account of a POLARDB for PostgreSQL cluster encounters a problem, for example, permissions are unexpectedly revoked, you can recover the account by resetting the account permissions.

1. Log on to the [ApsaraDB for POLARDB console](#).
2. Find the target cluster and click the cluster ID.
3. In the left-side navigation pane, click **Accounts**.
4. Click **Reset Permissions** to the right of **Privileged Account**.
5. In the dialog box that appears, enter the password of the privileged account to reset permissions.

Next step

[View connection endpoints](#).

Related API operations

API	Description
#unique_53	Used to create an account.
#unique_54	Used to list accounts.

API	Description
#unique_55	Used to modify the description of an account.
#unique_56	Used to change the password of an account .
#unique_57	Used to grant permissions to an account.
#unique_58	Used to revoke the permissions of an account.
#unique_59	Used to reset the permissions of an account.

11.5 Manage a database account

This topic describes how to manage a database account, including changing the password, locking the account, cancel the account locking, and deleting the account.

Create a database account

For more information, see [Create a database account](#).

Reset the password of a database account

1. Log on to the [ApsaraDB for POLARDB console](#).
2. In the upper-left corner, select the region where the target cluster is located.
3. Find the target cluster and click the cluster ID.
4. In the left-side navigation pane, choose **Settings and Management > Accounts**.
5. Find the target account and click **Modify Password**.
6. In the dialog box that appears, enter a new password and click **Confirm**.

Related API operations

Operation	Description
#unique_53	Creates a database account.
#unique_54	Queries the list of database accounts.
#unique_55	Modifies the description of a database account.
#unique_56	Changes the password of a database account.
#unique_57	Grants permissions on one or more databases.

Operation	Description
#unique_58	Revokes permissions on one or more database from a database account.
#unique_59	Resets permissions of a database account.
#unique_60	Deletes a database account.


12 Database management

This topic provides an overview of database management, including how to create and delete databases.

You can create and manage all databases in the ApsaraDB for PolarDB console.

Create a database

1. Log on to the [ApsaraDB for PolarDB console](#).
2. Select a region.
3. Find the target cluster and click the cluster ID in the **Cluster Name** column.
4. In the left-side navigation pane, choose **Settings and Management > Databases**.
5. Click **Create Database**.
6. In the dialog box that appears, set parameters for creating a database. The following table describes the parameters.

Parameter	Description
Database Name	<ul style="list-style-type: none">• It must start with a letter and end with a letter or digit.• It can contain lowercase letters, digits, underscores (_), and hyphens (-).• It must be 2 to 64 characters in length.• Each database name in an instance must be unique.
Supported Character Set	Select utf8mb4 , utf8 , gbk , or latin1 . You can also select other required character sets from the drop-down list on the right.
Authorized Account	Select the account that you want to authorize for accessing this database. You can leave this parameter blank, and bind an account after the database is created. <div> Note: Only standard accounts are available in the drop-down list. The privileged account has all the permissions on all databases. You do not need to authorize the privileged account to access the database that you create.</div>
Account Permission	Select the permission that you want to grant to your account. Valid values: Read and Write Read Only DML Only .

Parameter	Description
Description	<p>Enter the remarks of the database to facilitate subsequent database management. The requirements are as follows:</p> <ul style="list-style-type: none"> The description cannot start with http:// or https://. The description must start with an uppercase or lowercase letter or a Chinese character. The description can contain uppercase or lowercase letters, Chinese characters, digits, underscores (_), and hyphens (-). The description must be 2 to 256 characters in length.

Create Database

* Database Name

0/64

The name must be up to 64 characters in length and can contain lowercase letters, digits, hyphens (-), and underscores (_). It must start with a letter and end with a letter or digit.

* Supported

☒ utf8mb4
☐ utf8
☐ gbk
☐ latin1

Select

▼

Character Set

Authorized Account

Select

▼

Create Account

Description

0/256

OK

7. Click **OK**.

Delete a database

- Log on to the [ApsaraDB for PolarDB console](#).
- Select a region.
- Find the target cluster and click the cluster ID in the **Cluster Name** column.
- In the left-side navigation pane, choose **Settings and Management** > **Databases**.
- Find the target database and click **Delete** in the **Actions** column.
- In the dialog box that appears, click **OK**.

Related API operations

API operation	Description
#unique_62	Creates a database.
#unique_63	Views the database list.
#unique_64	Modifies the description of a database.
#unique_65	Deletes a database.

13 Backup and restoration

13.1 Back up data

ApsaraDB for POLARDB uses a physical backup (snapshot backup), which is automatically performed once a day. You can also manually start a backup. Both the automatic backup and manual backup do not affect the normal running of the cluster. Backup files are retained for seven days.

Backup types

Backup type	Description
Automatic backup	<ul style="list-style-type: none">It is performed once a day by default. You can configure the time period and cycle for automatic backup. For more information, see Configure an automatic backup policy.Backup files cannot be deleted.
Manual backup	<ul style="list-style-type: none">It can be started at any time. You can create a maximum of three manual backups for a cluster. For more information, see Manually create a backup.Backup files can be deleted.

Pricing

The storage occupied by ApsaraDB for POLARDB backup files is free of charge.

Configure an automatic backup policy

1. Log on to the [ApsaraDB for POLARDB console](#).
2. Select a region.
3. Find the target cluster and click the cluster ID in the **Cluster Name** column.
4. In the left-side navigation pane, choose **Settings and Management** > **Backup and Restore**.

5. Click **Backup Settings**.

Settings and Manag...	Create Backup	Point-in-time Restore	Backup Settings	Jun 13, 2019	- Aug 13, 2019	
Accounts						
Databases						
Backup and Restore						
Parameters						
Start Time/End Time	Backup Method		Backup Type	Backup Policy		
2019-08-13 15:13:05 - 2019-08-13 15:13:15	Snapshot Backup		Full Backup	System Backup		
2019-08-12 15:13:01 - 2019-08-12 15:13:11	Snapshot Backup		Full Backup	System Backup		

6. In the dialog box that appears, configure the time period and cycle for automatic backup.



Note:

For security reasons, automatic backup must be performed at least twice a week.

Manually create a backup

1. Log on to the [ApsaraDB for POLARDB console](#).
2. Select a region.
3. Find the target cluster and click the cluster ID in the **Cluster Name** column.
4. In the left-side navigation pane, choose **Settings and Management > Backup and Restore**.
5. Click **Create Backup**.

Settings and Manag...	Create Backup	Point-in-time Restore	Backup Settings	Jun 13, 2019	- Aug 13, 2019	
Accounts						
Databases						
Backup and Restore						
Parameters						
Start Time/End Time	Backup Method		Backup Type	Backup Policy		
2019-08-13 15:13:05 - 2019-08-13 15:13:15	Snapshot Backup		Full Backup	System Backup		
2019-08-12 15:13:01 - 2019-08-12 15:13:11	Snapshot Backup		Full Backup	System Backup		

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



Note:

You can create a maximum of three manual backups for a cluster.

Restore data

For more information, see [Restore data](#).

Related API operations

Operation	Description
#unique_71	Creates a full snapshot backup for a specified ApsaraDB for POLARDB cluster.

Operation	Description
#unique_72	Queries the backup data of a specified ApsaraDB for POLARDB cluster.
#unique_73	Deletes the backup data of a specified ApsaraDB for POLARDB cluster.
#unique_74	Queries the automatic backup policy of a specified ApsaraDB for POLARDB cluster.
#unique_75	Modifies the automatic backup policy of a specified ApsaraDB for POLARDB cluster.

13.2 Restore data

POLARDB for PostgreSQL supports [Restore by backup set \(snapshot\)](#) to restore historical data to a new cluster.



Note:

The restored cluster contains the data and account information of the original cluster, but does not contain the parameter settings of the original cluster.

Restore data from a backup set (snapshot)

1. Log on to the [ApsaraDB for POLARDB console](#).
2. Select the region where the cluster resides.
3. Find the target cluster and click the cluster ID.
4. In the left-side navigation pane, choose **Settings and Management > Backup and Restore**.
5. Find the target backup set (snapshot) and click **Restore** in the Actions column. In the dialog box that appears, click **OK**.
6. On the page that appears, select a billing method for the new cluster:
 - **Subscription:** An upfront payment must be made for each new compute cluster (contains a primary node and a read-only node by default). The storage occupied by the new cluster is billed on an hourly basis based on the actual data volume. The payment will be deducted from your Alibaba Cloud account on an hourly basis.

Subscription is more cost-effective for long term use. You can save more with longer subscription periods.

- **Pay-As-You-Go:** For the new cluster created, you do not need to pay any subscription fee for a compute cluster in advance. The compute cluster is billed on an hourly basis. The storage occupied by the new cluster is billed on an hourly basis based on the actual data volume. The payment will be deducted from your Alibaba Cloud account on an hourly basis. The **Pay-As-You-Go** method is most suitable for temporary applications, as you can release the cluster as soon as you do not need it anymore, saving costs.

7. Configure the following parameters:


- **Clone Source Type:** Select **Backup Set**.
- **Clone Source Backup Set:** Confirm that the backup set is the one that you want to restore from.
- **Region:** Use the default setting. It is the same as the region of the original cluster.
- **Primary Availability Zone:** Use the default setting.
- **Network Type:** Use the default setting.
- **VPC and VSwitch:** The default settings are recommended, namely, the VPC and VSwitch of the original cluster.
- **Database Engine:** Use the default setting.
- **Node Specification:** Clusters with different specifications have different storage capacity and performance. For more information, see [Node specifications](#).
- **Number of Nodes:** Use the default setting. By default, the system will create a read-only node with the same specifications as the primary node.
- **Cluster Name:** The system will automatically create a name for your cluster if you leave it blank. You can rename the cluster after it is created.
- **Purchase Plan:** Set this parameter if you create a subscription cluster.
- **Number:** The default value is 1, which cannot be modified.

8. Read the **ApsaraDB for POLARDB Agreement of Service**, select the check box to agree to it, and then complete the payment.

References

[Back up data](#)

Related API operations

Operation	Description
#unique_21	<p>Creates an ApsaraDB for POLARDB cluster.</p> <div> Note: When you clone a cluster, set the value of CreationOption to CloneFromPolarDB.</div>

14 Diagnostics and optimization

14.1 Performance monitoring and alert configuration

The ApsaraDB for POLARDB console provides a variety of performance metrics for you to monitor the status of your instances.

Performance monitoring

1. Log on to the [ApsaraDB for POLARDB console](#).
 2. Select a region.
 3. Find the target cluster and click the **cluster ID** in the **Cluster Name** column.
 4. In the left-side navigation pane, choose **Diagnostics and Optimization > Monitoring**.
 5. You can view the performance information of a **Cluster** or **Node** as needed. For more information, see [Metric description](#).
- To monitor cluster performance, click the **Cluster** tab and set the monitoring time period. Click **OK**.
 - To monitor node performance, click the **Node** tab, select a node from the **Select Node** drop-down list, and set the monitoring time period. Click **OK**.

**Note:**

You can click **More** at the lower part of the **Node** tab to view more metrics.

Metric description

Category	Metric	Description
Cluster	Storage	Displays the usage of data space, log space, temporary space, and WAL log space.
	CPU	Displays the CPU usage of each node.
	Memory	Displays the memory usage of each node.
Node	TPS	Displays the number of transactions per second of the selected node, including the number of committed transactions per second, deadlocked transactions per second, and rollback transactions per second.
	CPU	Displays the CPU usage of the selected node.
	Memory	Displays the memory usage of the selected node.

Category	Metric	Description
	Connections	Displays the total number of current connections, active connections, and idle connections of the selected node.
	Scanned Rows	Displays the numbers of rows inserted, read, updated, deleted, and returned per second of the selected node.
	Maximum Database Age	Displays the difference between the transaction IDs of the oldest and newest transactions in the database.
	I/O Throughput	Displays the total I/O throughput, I/O read throughput, and I/O write throughput of the selected node.
	IOPS	Displays the input/output operations per second (IOPS) of the selected node, including the total IOPS, read IOPS, and write IOPS.
	Cache	Displays the cache reads per second and disk reads per second of the selected node.
	Cache Hit Ratio	Displays the cache hit ratio of the selected node.
	Temporary Files	Displays the number and total size of temporary files on the selected node.

Alert settings

1. Log on to the [CloudMonitor console](#).
2. In the left-side navigation pane, choose **Alerts** > **Alert Rules**.
3. On the **Alert Rules** page, click **Create Alert Rule** to go to the **Create Alert Rule** page.
4. Select **ApsaraDB for POLARDB-PostgreSQL/Oracle** from the **Product** drop-down list and select a resource range from the **Resource Range** drop-down list. Set the alert rule and notification method, and click **Confirm**.

**Note:**

For more information about alert rules, see [#unique_80](#).

14.2 Performance insight

ApsaraDB for POLARDB provides the performance insight feature, which focuses on monitoring the load, analyzing the load, and optimizing the performance of an ApsaraDB

for POLARDB cluster. The feature helps you easily evaluate database loads, find the causes for performance problems, and enhance database stability.

Scenarios

Performance insight can be applied in the following scenarios:

- Analyze the cluster metrics

Performance insight helps you monitor the key metrics of an ApsaraDB for POLARDB cluster. It also allows you to check the status and trend of the loads for the cluster. You can identify the sources that generate loads and the distribution of loads within a certain period from the trend charts of key metrics.

- Evaluate database loads

ApsaraDB for POLARDB provides the trend chart of average active sessions (AAS), which alleviates the need to analyze the complicated trend charts of various metrics. AAS trend chart shows the information of all key metrics to help you evaluate the sources that generate loads and cause performance bottlenecks. You can determine the causes for performance bottlenecks, such as high CPU usage, lock-waiting, and I/O latency, and find the corresponding SQL statement that incurs the problem.



Note:

AAS is the number of average active sessions of an ApsaraDB for POLARDB cluster within a certain period. The trends of AAS reflect the changes of the loads for the cluster. In the performance insight feature, AAS is a key metric used to measure the loads for an ApsaraDB for POLARDB cluster.

- Find the sources that cause performance problems

You can analyze the trend chart of AAS and load sources in multiple dimensions to determine whether a performance problem is caused by improper cluster configurations or the database architecture. You can also find the corresponding SQL statement that incurs the performance problem.

Procedure

1. Log on to the [ApsaraDB for POLARDB console](#).
2. Select a region.
3. Find the target cluster and click the cluster ID in the **Cluster Name** column.

4. In the left-side navigation pane, choose **Diagnostics and Optimization > Performance Insight**.

5. Select filtering conditions.

Description of the metrics page

- Trend charts of key metrics

You can use the trend charts of key metrics to check the load status and resource bottlenecks of an ApsaraDB for POLARDB cluster.

You can select a given time period or specify a custom time period to retrieve the trend charts of key metrics within the corresponding time period.

- Trend chart of AAS

After you use the trend charts of key metrics to check the load status, you can identify the load sources.



Note:

max Vcores indicates the maximum number of CPU cores that can be used by an ApsaraDB for POLARDB cluster. The value determines the processing capacity of CPUs in the cluster.

From the real-time trend chart of AAS, you can find the load sources, the time when loads occur, and the trend of loads over a period of time.

- Load sources from multiple dimensions

You can learn the trend of the loads for an ApsaraDB for POLARDB cluster by analyzing the trend chart of AAS. You can find the specific SQL statements that cause performance bottlenecks, and the related users, hosts, and databases.

As shown in the lower section of the preceding figure, you can find the SQL statements that affect the loads, and the usage ratio of each statement in a specified AAS.

Performance insight supports six dimensions of AAS. You can switch dimensions by using the drop-down list of **AAS Type** in the upper-right corner of the AAS page.

Type	Description
SQL	The trends of top 10 SQL statements in your business.
Waits	The trends of wait events within the specified time period.
Users	The trends of logon users.

Type	Description
Hosts	The trends of hostnames or IP addresses of clients.
Databases	The trends of the databases where your businesses are located.
Status	The trends of active sessions within the specified time period.

15 SQL Explorer

Apsara PolarDB provides the SQL Explorer feature. You can use SQL Explorer for database security auditing and performance diagnostics.

Pricing

- The trial edition of Apsara PolarDB is available for free. In the trial edition, audit logs are retained for only one day. You can query only data that is stored in the retained audit logs. The trial edition does not support advanced features. For example, data cannot be exported, and data integrity cannot be ensured.
- If you want to retain the audit logs for 30 days or longer, you can view the pricing details in [#unique_28](#).

Features

- SQL logging

SQL audit logs record all operations that are performed on databases. You can use audit logs to identify database failures, analyze behaviors, and perform security auditing.

- Advanced search

SQL Explorer allows you to search data by database, user, client IP, thread ID, execution duration, or execution status. You can also export and download search results.

The screenshot shows the SQL Explorer interface. At the top, there's a header with 'SQL Explorer' and a 'Service Settings' button. Below the header is a 'Search' tab. Under 'Set Filters', there are input fields for 'Time Range' (May 21, 2020 17:03:54 to May 21, 2020 17:18:54), 'Keywords', 'Users', and 'Databases'. There are also 'Enable Advanced Search' and 'Search' buttons. Below the filters is a 'Log Entries' section with a table. The table has columns: SQL Statement, Database, Thread ID, User, Client IP Address, Status, Time Consumption(ms), Executed At, Updated Rows, and Scanned Rows. There are also 'More Actions' buttons: 'Export' and 'View Exported List'.

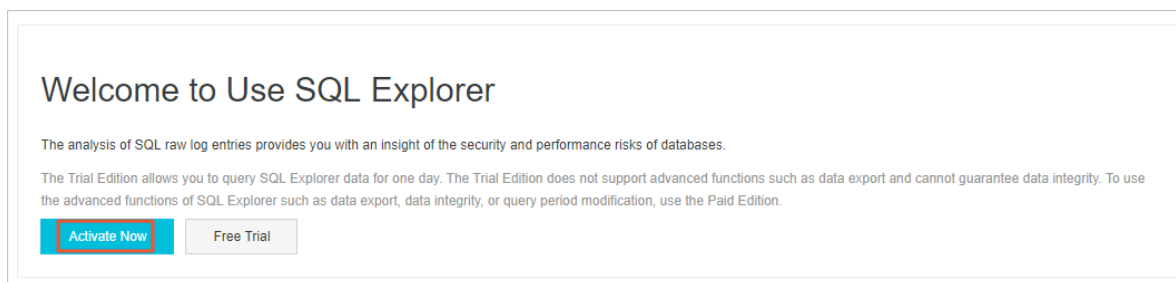
SQL Statement	Database	Thread ID	User	Client IP Address	Status	Time Consumption(ms)	Executed At	Updated Rows	Scanned Rows
---------------	----------	-----------	------	-------------------	--------	----------------------	-------------	--------------	--------------

Enable SQL Explorer

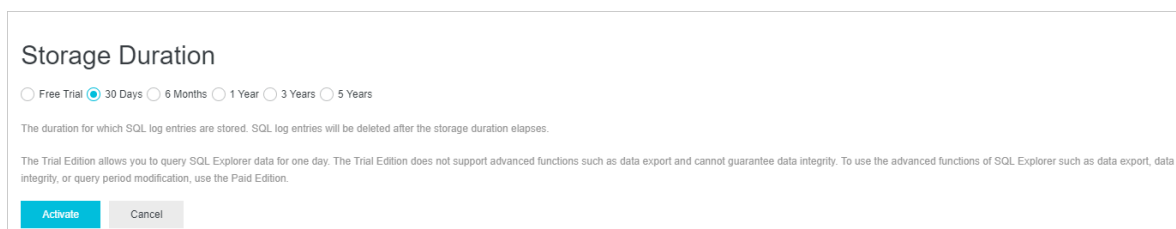
- Log on to the [Apsara PolarDB console](#).
- In the upper-left corner of the console, select the region where the target cluster resides.
- Find the target cluster and click its ID.

4. In the left-side navigation pane, choose **Log and Audit > SQL Explorer**.

5. Click **Activate Now**.



6. Specify the storage duration of SQL audit logs, and then click **Activate**.



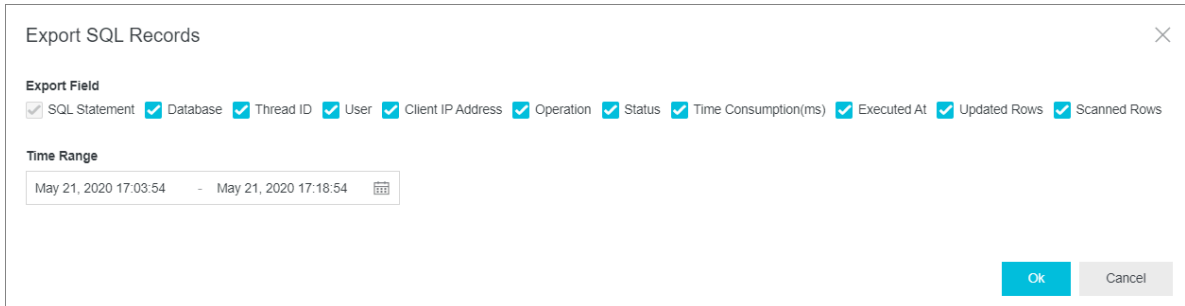
Change the storage duration of SQL audit logs

1. Log on to the [Apsara PolarDB console](#).
2. In the upper-left corner of the console, select the region where the target cluster resides.
3. Find the target cluster and click its ID.
4. In the left-side navigation pane, choose **Log and Audit > SQL Explorer**.
5. In the upper-right corner of the page, click **Service Settings**.
6. Change the storage duration and click **OK**.

Export SQL records

1. Log on to the [Apsara PolarDB console](#).
2. In the upper-left corner of the console, select the region where the target cluster resides.
3. Find the target cluster and click its ID.
4. In the left-side navigation pane, choose **Log and Audit > SQL Explorer**.
5. On the right side of the page, click **Export**.

6. In the dialog box that appears, specify the **Export Field** and **Time Range** parameters, and click **OK**.

A dialog box titled "Export SQL Records" with a close button (X) in the top right corner. It contains two sections: "Export Field" and "Time Range". The "Export Field" section has a list of checkboxes, all of which are checked: SQL Statement, Database, Thread ID, User, Client IP Address, Operation, Status, Time Consumption(ms), Executed At, Updated Rows, and Scanned Rows. The "Time Range" section has a date and time range selector showing "May 21, 2020 17:03:54" to "May 21, 2020 17:18:54" with a calendar icon. At the bottom right, there are "OK" and "Cancel" buttons.

Export SQL Records

Export Field

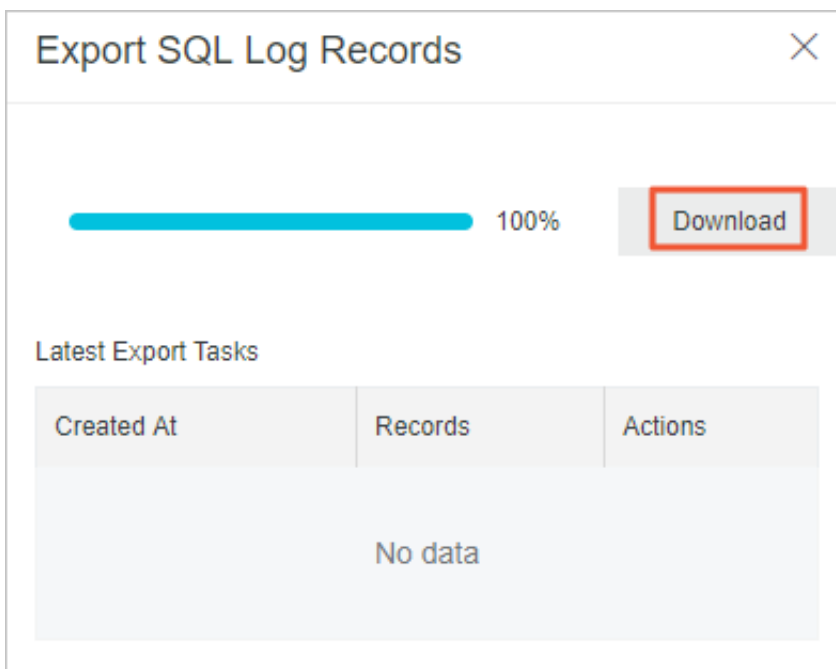
☒ SQL Statement ☒ Database ☒ Thread ID ☒ User ☒ Client IP Address ☒ Operation ☒ Status ☒ Time Consumption(ms) ☒ Executed At ☒ Updated Rows ☒ Scanned Rows

Time Range

May 21, 2020 17:03:54 - May 21, 2020 17:18:54

OK Cancel

7. After the export is complete, download the log files in the **Export SQL Log Records** dialog box.

A dialog box titled "Export SQL Log Records" with a close button (X) in the top right corner. It features a progress bar at 100% and a "Download" button highlighted with a red rectangle. Below this is a section titled "Latest Export Tasks" containing a table with columns "Created At", "Records", and "Actions". The table is currently empty, displaying "No data".

Export SQL Log Records

100% Download

Latest Export Tasks

Created At	Records	Actions
No data		

Disable SQL Explorer



Note:

After SQL Explorer is disabled, SQL audit logs are deleted. We recommend that you [export](#) and save SQL log files to your computer before you disable SQL Explorer.

1. Log on to the [Apsara PolarDB console](#).
2. In the upper-left corner of the console, select the region where the target cluster resides.
3. Find the target cluster and click its ID.
4. In the left-side navigation pane, choose **Log and Audit > SQL Explorer**.
5. In the upper-right corner of the page, click **Service Settings**.

6. Change the storage duration and click **OK**.
7. Turn off the Activate SQL Explorer switch.

Service Settings

Activate SQL Explorer ☒

Storage Duration ☒ 30 Days ☐ 6 Months ☐ 1 Year ☐ 3 Years ☐ 5 Years

The duration for which SQL log entries are stored. SQL log entries will be deleted after the storage duration elapses.

OK Cancel

View the size and consumption details of audit logs

1. Log on to the [Alibaba Cloud console](#).
2. In the upper-right corner of the page, choose **Billing > User Center**.
3. In the left-side navigation pane, choose **Spending Summary > Instance Spending Detail**.
4. Click **Click here to experience the new version**.

Billing Management
Account Overview
Spending Summary
Spending Summary
Instance Spending De...
Bills
Orders
Usage Records

Instance Spending Details

The new version of Spending Summary is online, you are welcome to experience! The current version will be offlined in December. [Click here to experience the new version >](#)

Month (Updated: Aug 19, 2019, 16:00:00)
Aug 2019

Search By
All

Product Family
All Product Family

Product Name
All Product Name

Billing Method
Pay-As-You-Go



Note:

Skip this step if you have switched to the new version.

5. Click the **Details** tab, select **Instance Name** from the drop-down list, enter the instance name in the search box, and click Search.

Bills

Overview Bills **Details**

2019-08 All Resource Groups Instance ID pc- Search

Statistic Item: ☒ Billing Item ☐ Instance ☐ Product ☐ Account ☐ Cost Center

Statistic Period: ☒ Billing Cycle ☐ By Day ☐ Billing Period

[Customize Column](#)

Billing Cycle	Cost Center	Account Name	Product Name	Product Detail	Subscription Type	Instance ID	Resource Group	Region	Billing Item	List Price
2019-08	Not Allocated		ApsaraDB for POLARDB	polardb	Subscription	pc-		Indonesia (Jakarta)	create_type	0.000000
2019-08	Not Allocated		ApsaraDB for POLARDB	polardb	Subscription	pc-		Indonesia (Jakarta)	master_ha	0.000000
2019-08	Not Allocated		ApsaraDB for POLARDB	polardb	Subscription	pc-		Indonesia (Jakarta)	master_spec	0.000000
2019-08	Not Allocated		ApsaraDB for POLARDB	polardb	Subscription	pc-		Indonesia (Jakarta)	net_type	0.000000
2019-08	Not Allocated		ApsaraDB for POLARDB	polardb	Subscription	pc-		Indonesia (Jakarta)	region	0.000000
2019-08	Not Allocated		ApsaraDB for POLARDB	polardb	Subscription	pc-		Indonesia (Jakarta)	iz	0.000000
2019-08	Not Allocated		ApsaraDB for POLARDB	polardb	Subscription	pc-		Indonesia (Jakarta)	vpclid	0.000000
2019-08	Not Allocated		ApsaraDB for POLARDB	polardb	Subscription	pc-		Indonesia (Jakarta)	engine_type	0.000000
2019-08	Not Allocated		ApsaraDB for POLARDB	polardb	Subscription	pc-		Indonesia (Jakarta)	vswitchID	0.000000

6. View the billing details whose **Billing Item** is **sql_explorer**.

16 Plug-ins

16.1 Read and write external data files by using oss_fdw

Alibaba Cloud allows you to use the `oss_fdw` plug-in to load data in OSS to POLARDB for PostgreSQL databases and write data in POLARDB for PostgreSQL databases to OSS.

`oss_fdw` parameters

The `oss_fdw` plug-in uses a method similar to other Foreign Data Wrapper (FDW) interfaces to encapsulate external data stored in OSS. You can use `oss_fdw` to read data stored in OSS. This process is similar to reading data tables. `oss_fdw` provides unique parameters to connect and parse file data in OSS.



Note:

- `oss_fdw` can read and write files of the following types in OSS: TEXT and CSV files as well as gzip-compressed TEXT and CSV files.
- The value of each parameter must be enclosed in double quotation marks (") and cannot contain any unnecessary spaces.

CREATE SERVER parameters

- `ossendpoint`: the endpoint used to access OSS through the internal network, also known as the host.
- `id oss`: the ID of your OSS account.
- `key oss`: the key of your OSS account.
- `bucket`: the OSS bucket. You must create an OSS account before specifying this parameter.

The following fault tolerance parameters can be used for data import and export. If network connectivity is poor, you can adjust these parameters to guarantee successful import and export.

- `oss_connect_timeout`: indicates the connection timeout period. Default value: 10. Unit: seconds.
- `oss_dns_cache_timeout`: indicates the DNS timeout period. Default value: 60. Unit: seconds.

- `oss_speed_limit`: indicates the minimum data transmission rate. Default value: 1. Unit: Kbit/s.
- `oss_speed_time`: indicates the maximum period when the data transmission rate is lower than the minimum value. Default value: 15. Unit: seconds.

If the default values of `oss_speed_limit` and `oss_speed_time` are used, a timeout error occurs when the transmission rate is smaller than 1 Kbit/s for 15 consecutive seconds.

CREATE FOREIGN TABLE parameters

- `filepath`: a file name that contains a path in OSS.
 - A file name contains a path but not a bucket name.
 - This parameter matches multiple files in the corresponding path in OSS. You can load multiple files to a database.
 - You can import files named in the format of `filepath` or `filepath.x` to a database. The values of `x` must be consecutive numbers starting from 1.

For example, among the files named `filepath`, `filepath.1`, `filepath.2`, `filepath.3`, and `filepath.5`, the first four files are matched and imported. The `filepath.5` file is not imported.

- `dir`: the virtual file directory in OSS.
 - `dir` must end with a forward slash (/).
 - All files (excluding subfolders and files in subfolders) in the virtual file directory specified by `dir` will be matched and imported to a database.
- `prefix`: the prefix of the path name corresponding to the data file. The prefix does not support regular expressions. Only one parameter among `prefix`, `filepath`, and `dir` can be specified at a time because they are mutually exclusive.
- `format`: the file format, which can only be `csv`.
- `encoding`: the file data encoding format. It supports common PostgreSQL encoding formats, such as UTF-8.
- `parse_errors`: the fault-tolerant parsing mode. If an error occurs during the parsing process, the entire row of data is ignored.
- `delimiter`: the column delimiter.
- `quote`: the quote character for files.
- `escape`: the escape character for files.

- `null`: sets the column matching a specified string to null. For example, `null 'test'` is used to set the value of the 'test' column to null.
- `force_not_null`: sets the value of a column to a non-null value. For example, `force_not_null 'id'` is used to set the value of the 'id' column to empty strings.
- `compressiontype`: specifies the format of the files to be read or written in OSS.
 - `none`: uncompressed text files. This is the default value.
 - `gzip`: The files to be read must be gzip compressed.
- `compressionlevel`: specifies the compression level of the compression format written to OSS. Valid values: 1 to 9. Default value: 6.

**Note:**

- You must specify filepath and dir in the `OPTIONS` parameter.
- You must specify either filepath or dir.
- The export mode only supports virtual folders, that is, only dir is supported.

Export mode parameters for CREATE FOREIGN TABLE

- `oss_flush_block_size`: the buffer size for the data written to OSS at a time. Default value: 32 MB. Valid values: 1 MB to 128 MB.
- `oss_file_max_size`: the maximum file size for the data written to OSS (subsequent data is written in another file when the maximum file size is exceeded). Default value: 1024 MB. Valid values: 8 MB to 4000 MB.
- `num_parallel_worker`: the number of parallel compression threads in which the OSS data is written. Valid values: 1 to 8. Default value: 3.

Auxiliary functions

FUNCTION `oss_fdw_list_file` (relnname text, schema text DEFAULT 'public')

- Obtains the name and size of the OSS file that an external table matches.
- The unit of file size is Byte.

```
select * from oss_fdw_list_file('t_oss');
      name      | size
-----+-----
oss_test/test.gz.1 | 739698350
oss_test/test.gz.2 | 739413041
oss_test/test.gz.3 | 739562048
```

(3 rows)

Auxiliary features

oss_fdw.rds_read_one_file: In read mode, it is used to specify a file to match the external table. If the file is specified, the external table only matches this file during data import.

Example: set oss_fdw.rds_read_one_file = 'oss_test/example16.csv.1';

```
set oss_fdw.rds_read_one_file = 'oss_test/test.gz.2';
select * from oss_fdw_list_file('t_oss');
      name      | size
-----+-----
 oss_test/test.gz.2 | 739413041
(1 rows)
```

oss_fdw example

```
# Create a plug-in
create extension oss_fdw;
# Create a server
CREATE SERVER osserver FOREIGN DATA WRAPPER oss_fdw OPTIONS
  (host 'oss-cn-hangzhou.aliyuncs.com', id 'xxx', key 'xxx', bucket 'mybucket');
# Create an OSS external table
CREATE FOREIGN TABLE ossexample
  (date text, time text, open float,
   high float, low float, volume int)
  SERVER osserver
  OPTIONS ( filepath 'osstest/example.csv', delimiter ',',
    format 'csv', encoding 'utf8', PARSE_ERRORS '100');
# Create a table to load data to
create table example
  (date text, time text, open float,
   high float, low float, volume int)
# Load data from ossexample to example.
insert into example select * from ossexample;
# Result
# oss_fdw estimates the file size in OSS and formulates a query plan correctly.
explain insert into example select * from ossexample;
      QUERY PLAN
-----
Insert on example (cost=0.00..1.60 rows=6 width=92)
-> Foreign Scan on ossexample (cost=0.00..1.60 rows=6 width=92)
    Foreign OssFile: osstest/example.csv.0
    Foreign OssFile Size: 728
(4 rows)
# Write the data in the example table to OSS.
insert into ossexample select * from example;
explain insert into ossexample select * from example;
      QUERY PLAN
-----
Insert on ossexample (cost=0.00..16.60 rows=660 width=92)
-> Seq Scan on example (cost=0.00..16.60 rows=660 width=92)
```

(2 rows)

oss_fdw usage considerations

- oss_fdw is an external table plug-in developed based on the PostgreSQL FOREIGN TABLE framework.
- The data import efficiency is subject to the POLARDB for PostgreSQL cluster resources (CPU, I/O, memory, and MET) and OSS.
- To guarantee data import performance, make sure that POLARDB for PostgreSQL is in the same region as OSS. For more information, see [Endpoints](#).
- If the error "oss endpoint userendpoint not in aliyun white list" is reported during reading of SQL statements for external tables, use the endpoints listed in [Regions and endpoints](#). If the problem persists, submit a ticket.

Error handling

When an import or export error occurs, the log displays the following error information:

- code: the HTTP status code of the request that has failed.
- error_code: the error code returned by OSS.
- error_msg: the error message returned by OSS.
- req_id: the UUID that identifies the request. If you cannot solve the problem, you can seek help from OSS development engineers by providing the req_id.

For more information about error types, see the following references. Timeout errors can be handled using oss_ext parameters.

- [OSS help](#)
- [OSS error handling](#)
- [OSS error response](#)

ID and key encryption

If id and key parameters for CREATE SERVER are not encrypted, executing the `select * from pg_foreign_server` statement will display the information in plaintext. Your ID and key will be exposed. You can use symmetric encryption to hide the ID and key. Use different keys for different instances to further protect your information. However, to avoid incompatibility with earlier versions, do not add data types as you do in Greenplum.

Encrypted information:

```
postgres=# select * from pg_foreign_server ;
  srvname | srvowner | srvfdw | srvtype | srvversion | srvacl |
-----+-----+-----+-----+-----+-----
  ossserver | 10 | 16390 |  |  |  | {host=oss-cn-hangzhou-zmf.aliyuncs.com,id=MD5xxxxxxxx,key=MD5xxxxxxxx,bucket=067862}
```

The encrypted information is preceded by the MD5 hash value. The remainder of the total length divided by 8 is 3. Therefore, encryption is not performed again when the exported data is imported. But you cannot create the key and ID preceded by an MD5 hash value.

16.2 Use the pg_pathman plug-in

This topic describes common usage scenarios of the pg_pathman plug-in.

Context

To improve the performance of partitioned tables, the pg_pathman plug-in is introduced to POLARDB for PostgreSQL. This plug-in enables you to manage partitions and optimize partitioning.

Create the pg_pathman extension

```
test=# create extension pg_pathman;
CREATE EXTENSION
```

View installed extensions

Run the following commands to view installed extensions and the version of the pg_pathman plug-in.

```
test=# \dx
          List of installed extensions
  Name | Version | Schema | Description
-----+-----+-----+-----
 pg_pathman | 1.5 | public | Partitioning tool for PostgreSQL
 plpgsql | 1.0 | pg_catalog | PL/pgSQL procedural language
(2 rows)
```

Upgrade the plug-in

POLARDB for PostgreSQL upgrades the plug-in on a regular basis to provide better database services. To upgrade the plug-in, perform the following steps:

- Upgrade the corresponding cluster to the latest version. You need to submit a ticket to perform the upgrade because this function is in the development phase.

- Execute the following statements to complete the update:

```
ALTER EXTENSION pg_pathman UPDATE;  
SET pg_pathman.enable = t;
```

Features

- Support for hash and range partitioning.
- Support for automatic and manual partition management. In automatic partition management, the system uses functions to create partitions and migrate data in primary tables to partitions. In manual partition management, you can use functions to attach existing tables to partitioned tables or detach tables from partitioned tables.
- Support for several partition fields including custom domains and common data types such as integer, floating point, and date.
- Effective query planning for partitioned tables by using joins and subselects.
- `RuntimeAppend` and `RuntimeMergeAppend` custom plan nodes to pick partitions at runtime.
- `PartitionFilter`: an efficient drop-in replacement for INSERT triggers.
- Automatic partition creation for newly inserted data (only for range partitioning).
- Support for the `COPY FROM/TO` statement, allowing direct read or write operations on partitions.
- Partition fields update. To update partition fields, you need to add a trigger. If you do not need to update partition fields, we recommend that you do not add the trigger to avoid negative impacts on performance.
- User-defined callback functions are automatically triggered when partitions are created.
- Non-blocking concurrent table partitioning and automatic data migration from primary tables to partitions in the background.
- Support for `postgres_fdw` or any Foreign Data Wrappers (FDW) by configuring the `pg_pathman.insert_into_fdw=(disabled | postgres | any_fdw)` parameter.

Usage

- [Views and tables](#)
- [Partition management](#)
- [Advanced partition management](#)

For more information, see https://github.com/postgrespro/pg_pathman.

Views and tables

The pg_pathman plug-in uses functions to maintain partitioned tables and creates views for viewing the status of partitioned tables, as described in the following examples:

1. pathman_config

```
CREATE TABLE IF NOT EXISTS pathman_config (
    partrel    REGCLASS NOT NULL PRIMARY KEY, -- The OID of the primary table.
    attname    TEXT NOT NULL, -- The column name of the partition.
    parttype   INTEGER NOT NULL, -- The type of the partition (hash or range).
    range_interval TEXT, -- The interval of range partitions.

    CHECK (parttype IN (1, 2)) /* check for allowed part types */ );
```

2. pathman_config_params

```
CREATE TABLE IF NOT EXISTS pathman_config_params (
    partrel    REGCLASS NOT NULL PRIMARY KEY, -- The OID of the primary table.
    enable_parent BOOLEAN NOT NULL DEFAULT TRUE, -- Specifies whether to filter the
    primary table in the optimizer.
    auto       BOOLEAN NOT NULL DEFAULT TRUE, -- Specifies whether to automatically
    expand partitions that do not exist during INSERT operations.
    init_callback REGPROCEDURE NOT NULL DEFAULT 0; -- The OID of the callback
    function when the partition is created.
```

3. pathman_concurrent_part_tasks

```
-- helper SRF function
CREATE OR REPLACE FUNCTION show_concurrent_part_tasks()
RETURNS TABLE (
    userid    REGROLE,
    pid       INT,
    dbid      OID,
    relid     REGCLASS,
    processed INT,
    status    TEXT)
AS 'pg_pathman', 'show_concurrent_part_tasks_internal'
LANGUAGE C STRICT;

CREATE OR REPLACE VIEW pathman_concurrent_part_tasks
AS SELECT * FROM show_concurrent_part_tasks();
```

4. pathman_partition_list

```
-- helper SRF function
CREATE OR REPLACE FUNCTION show_partition_list()
RETURNS TABLE (
    parent    REGCLASS,
    partition REGCLASS,
    parttype  INT4,
    partattr  TEXT,
    range_min TEXT,
    range_max TEXT)
AS 'pg_pathman', 'show_partition_list_internal'
LANGUAGE C STRICT;

CREATE OR REPLACE VIEW pathman_partition_list
```

```
AS SELECT * FROM show_partition_list();
```

Partition management

1. Range partitions

Four management functions are used to create range partitions. Two functions are used to specify the start value, interval, and number of partitions. The definition of these two functions are as follows:

```
create_range_partitions(relation REGCLASS, -- The OID of the primary table.
                        attribute TEXT, -- The column name of the partition.
                        start_value ANYELEMENT, -- The start value.
                        p_interval ANYELEMENT, -- The interval, applicable to any type of
partitioned tables.
                        p_count INTEGER DEFAULT NULL, -- The number of partitions.
                        partition_data BOOLEAN DEFAULT TRUE) -- Specifies whether to
immediately migrate data from the primary table to partitions. We recommend
that you call the partition_table_concurrently() function to run non-blocking data
migration.

create_range_partitions(relation REGCLASS, -- The OID of the primary table.
                        attribute TEXT, -- The column name of the partition.
                        start_value ANYELEMENT, -- The start value.
                        p_interval INTERVAL, -- The interval of the interval type, applicable to
ingestion-time partitioned tables.
                        p_count INTEGER DEFAULT NULL, -- The number of partitions.
                        partition_data BOOLEAN DEFAULT TRUE) -- Specifies whether to
immediately migrate data from the primary table to partitions. We recommend
that you call the partition_table_concurrently() function to run non-blocking data
migration.
```

The other two functions are used to specify the start value, end value, and interval. The definitions are as follows:

```
create_partitions_from_range(relation REGCLASS, -- The OID of the primary table.
                             attribute TEXT, -- The column name of the partition.
                             start_value ANYELEMENT, -- The start value.
                             end_value ANYELEMENT, -- The end value.
                             p_interval ANYELEMENT, -- The interval, applicable to any type of
partitioned tables.
                             partition_data BOOLEAN DEFAULT TRUE) -- Specifies whether to
immediately migrate data from the primary table to partitions. We recommend
that you call the partition_table_concurrently() function to run non-blocking data
migration.

create_partitions_from_range(relation REGCLASS, -- The OID of the primary table.
                             attribute TEXT, -- The column name of the partition.
                             start_value ANYELEMENT, -- The start value.
                             end_value ANYELEMENT, -- The end value.
                             p_interval INTERVAL, -- The interval of the interval type, applicable
to ingestion-time partitioned tables.
                             partition_data BOOLEAN DEFAULT TRUE) -- Specifies whether to
immediately migrate data from the primary table to partitions. We recommend
```

that you call the `partition_table_concurrently()` function to run non-blocking data migration.

Example:

Create a primary table that needs to be partitioned.
 postgres=# create table part_test(id int, info text, crt_time timestamp not null); -- All partition columns must contain the NOT NULL constraint.
 CREATE TABLE

Insert a large amount of test data to simulate a primary table that already contains data.

postgres=# insert into part_test select id,md5(random()::text),clock_timestamp() + (id || ' hour')::interval from generate_series(1,10000) t(id);

INSERT 0 10000

postgres=# select * from part_test limit 10;

id	info	crt_time
1	36fe1adedaa5b848caec4941f87d443a	2016-10-25 10:27:13.206713
2	c7d7358e196a9180efb4d0a10269c889	2016-10-25 11:27:13.206893
3	005bdb063550579333264b895df5b75e	2016-10-25 12:27:13.206904
4	6c900a0fc50c6e4da1ae95447c89dd55	2016-10-25 13:27:13.20691
5	857214d8999348ed3cb0469b520dc8e5	2016-10-25 14:27:13.206916
6	4495875013e96e625afbf2698124ef5b	2016-10-25 15:27:13.206921
7	82488cf7e44f87d9b879c70a9ed407d4	2016-10-25 16:27:13.20693
8	a0b92547c8f17f79814dfbb12b8694a0	2016-10-25 17:27:13.206936
9	2ca09e0b85042b476fc235e75326b41b	2016-10-25 18:27:13.206942
10	7eb762e1ef7dca65faf413f236dff93d	2016-10-25 19:27:13.206947

(10 rows)

Note:

1. All partition columns must contain the NOT NULL constraint.
2. The number of partitions must be sufficient to cover all existing records.

Create partitions and ensure that each partition contains one month of data

postgres=# select
 create_range_partitions('part_test'::regclass, -- The OID of the primary table.
 'crt_time', -- The column name of the partition.
 '2016-10-25 00:00:00'::timestamp, -- The start value.
 interval '1 month', -- The interval of the interval type, applicable
 to ingestion-time partitioned tables.
 24, -- The number of partitions.
 false); -- The data is not migrated.

NOTICE: sequence "part_test_seq" does not exist, skipping

create_range_partitions

 24

(1 row)

postgres=# \d+ part_test

Column	Type	Modifiers	Storage	Stats target	Description
id	integer		plain		
info	text		extended		
crt_time	timestamp without time zone	not null	plain		

Child tables: part_test_1,
 part_test_10,
 part_test_11,
 part_test_12,
 part_test_13,
 part_test_14,
 part_test_15,

```

part_test_16,
part_test_17,
part_test_18,
part_test_19,
part_test_2,
part_test_20,
part_test_21,
part_test_22,
part_test_23,
part_test_24,
part_test_3,
part_test_4,
part_test_5,
part_test_6,
part_test_7,
part_test_8,
part_test_9

```

The data is still in the primary table because it is not migrated.

```

postgres=# select count(*) from only part_test;
count
-----
10000
(1 row)

```

Run non-blocking data migration.

```

partition_table_concurrently(relation REGCLASS,          -- The OID of the primary
table.
                             batch_size INTEGER DEFAULT 1000, -- The number of records to copy
from the primary table at a time.
                             sleep_time FLOAT8 DEFAULT 1.0) -- The time interval between
migration attempts if one or more rows in the batch are locked by other queries.
pg_pathman waits for the specified time and tries again up to 60 times before quitting
.

```

```

postgres=# select partition_table_concurrently('part_test'::regclass,
10000,
1.0);
NOTICE: worker started, you can stop it with the following command: select
stop_concurrent_part_task('part_test');
partition_table_concurrently
-----

```

```

(1 row)

```

After the migration, all data is migrated to the partitions, and the primary table is empty.

```

postgres=# select count(*) from only part_test;
count
-----
0
(1 row)

```

After the data is migrated, we recommend that you disable the primary table so that the primary table will not be included in the execution plan.

```

postgres=# select set_enable_parent('part_test'::regclass, false);
set_enable_parent
-----

```

(1 row)

```
postgres=# explain select * from part_test where crt_time = '2016-10-25 00:00:00'::timestamp;
```

QUERY PLAN

```
-----
Append (cost=0.00..16.18 rows=1 width=45)
  -> Seq Scan on part_test_1 (cost=0.00..16.18 rows=1 width=45)
      Filter: (crt_time = '2016-10-25 00:00:00'::timestamp without time zone)
(3 rows)
```



Note:

When using range partitioning, note the following items:

- All partition columns must contain the NOT NULL constraint.
- The number of partitions must be sufficient to cover all existing records.
- Run non-blocking data migration.
- After data migration is completed, disable the primary table.

2. Hash partitioning

You can use a management function to create range partitions. You can specify the start value, interval, and number of partitions, as described in the following examples:

```
create_hash_partitions(relation REGCLASS, -- The OID of the primary table.
                      attribute TEXT, -- The column name of the partition.
                      partitions_count INTEGER, -- The number of partitions to be created.
                      partition_data BOOLEAN DEFAULT TRUE) -- Specifies whether to
immediately migrate data from the primary table to partitions. We recommend
that you call the partition_table_concurrently() function to run non-blocking data
migration.
```

Example:

```
Create a primary table that needs to be partitioned.
postgres=# create table part_test(id int, info text, crt_time timestamp not null); -- All
partition columns must contain the NOT NULL constraint.
CREATE TABLE
```

Insert a large amount of test data to simulate a primary table that already contains data.

```
postgres=# insert into part_test select id,md5(random()::text),clock_timestamp() + (id
|| ' hour')::interval from generate_series(1,10000) t(id);
INSERT 0 10000
```

```
postgres=# select * from part_test limit 10;
```

id	info	crt_time
1	29ce4edc70dbfbe78912beb7c4cc95c2	2016-10-25 10:47:32.873879
2	e0990a6fb5826409667c9eb150fef386	2016-10-25 11:47:32.874048
3	d25f577a01013925c203910e34470695	2016-10-25 12:47:32.874059
4	501419c3f7c218e562b324a1bebfe0ad	2016-10-25 13:47:32.874065
5	5e5e22bdf110d66a5224a657955ba158	2016-10-25 14:47:32.87407
6	55d2d4fd5229a6595e0dd56e13d32be4	2016-10-25 15:47:32.874076
7	1dfb9a783af55b123c7a888afe1eb950	2016-10-25 16:47:32.874081
8	41eeb0bf395a4ab1e08691125ae74bff	2016-10-25 17:47:32.874087
9	83783d69cc4f9bb41a3978fe9e13d7fa	2016-10-25 18:47:32.874092

```
10 | affc9406d5b3412ae31f7d7283cda0dd | 2016-10-25 19:47:32.874097
(10 rows)
```

Note:

1. All partition columns must contain the NOT NULL constraint.

Create 128 partitions

```
postgres=# select
create_hash_partitions('part_test'::regclass, -- The OID of the primary table.
                      'crt_time',           -- The column name of the partition.
                      128,                   -- The number of partitions to be created.
                      false);               -- The data is not migrated.
create_hash_partitions
-----
128
```

```
(1 row)
```

```
postgres=# \d+ part_test
```

```
Table "public.part_test"
Column |          Type          | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
id      | integer               |           |         |              |
info    | text                  |           |         |              |
crt_time | timestamp without time zone | not null | plain   |              |
```

Child tables: part_test_0,

```
part_test_1,
part_test_10,
part_test_100,
part_test_101,
part_test_102,
part_test_103,
part_test_104,
part_test_105,
part_test_106,
part_test_107,
part_test_108,
part_test_109,
part_test_11,
part_test_110,
part_test_111,
part_test_112,
part_test_113,
part_test_114,
part_test_115,
part_test_116,
part_test_117,
part_test_118,
part_test_119,
part_test_12,
part_test_120,
part_test_121,
part_test_122,
part_test_123,
part_test_124,
part_test_125,
part_test_126,
part_test_127,
part_test_13,
part_test_14,
part_test_15,
part_test_16,
part_test_17,
part_test_18,
```

```
part_test_19,  
part_test_2,  
part_test_20,  
part_test_21,  
part_test_22,  
part_test_23,  
part_test_24,  
part_test_25,  
part_test_26,  
part_test_27,  
part_test_28,  
part_test_29,  
part_test_3,  
part_test_30,  
part_test_31,  
part_test_32,  
part_test_33,  
part_test_34,  
part_test_35,  
part_test_36,  
part_test_37,  
part_test_38,  
part_test_39,  
part_test_4,  
part_test_40,  
part_test_41,  
part_test_42,  
part_test_43,  
part_test_44,  
part_test_45,  
part_test_46,  
part_test_47,  
part_test_48,  
part_test_49,  
part_test_5,  
part_test_50,  
part_test_51,  
part_test_52,  
part_test_53,  
part_test_54,  
part_test_55,  
part_test_56,  
part_test_57,  
part_test_58,  
part_test_59,  
part_test_6,  
part_test_60,  
part_test_61,  
part_test_62,  
part_test_63,  
part_test_64,  
part_test_65,  
part_test_66,  
part_test_67,  
part_test_68,  
part_test_69,  
part_test_7,  
part_test_70,  
part_test_71,  
part_test_72,  
part_test_73,  
part_test_74,  
part_test_75,  
part_test_76,
```

```

part_test_77,
part_test_78,
part_test_79,
part_test_8,
part_test_80,
part_test_81,
part_test_82,
part_test_83,
part_test_84,
part_test_85,
part_test_86,
part_test_87,
part_test_88,
part_test_89,
part_test_9,
part_test_90,
part_test_91,
part_test_92,
part_test_93,
part_test_94,
part_test_95,
part_test_96,
part_test_97,
part_test_98,
part_test_99

```

The data is still in the primary table because it is not migrated.

```

postgres=# select count(*) from only part_test;
count
-----
10000
(1 row)

```

Run non-blocking data migration

```

partition_table_concurrently(relation REGCLASS,          -- The OID of the primary
table.
                             batch_size INTEGER DEFAULT 1000, -- The number of records to copy
                             from the primary table at a time.
                             sleep_time FLOAT8 DEFAULT 1.0) -- The time interval between
migration attempts if one or more rows in the batch are locked by other queries.
pg_pathman waits for the specified time and tries again up to 60 times before quitting
.

```

```

postgres=# select partition_table_concurrently('part_test'::regclass,
10000,
1.0);

```

```

NOTICE: worker started, you can stop it with the following command: select
stop_concurrent_part_task('part_test');
partition_table_concurrently
-----

```

```

(1 row)

```

After the migration, all data is migrated to the partitions, and the primary table is empty.

```

postgres=# select count(*) from only part_test;
count
-----
0
(1 row)

```


After the data is migrated, we recommend that you disable the primary table so that the primary table will not be included in the execution plan.

```
postgres=# select set_enable_parent('part_test'::regclass, false);
set_enable_parent
```

(1 row)

Query only a single partition.

```
postgres=# explain select * from part_test where crt_time = '2016-10-25 00:00:00'::timestamp;
```

QUERY PLAN

```
-----
Append (cost=0.00..1.91 rows=1 width=45)
  -> Seq Scan on part_test_122 (cost=0.00..1.91 rows=1 width=45)
      Filter: (crt_time = '2016-10-25 00:00:00'::timestamp without time zone)
(3 rows)
```

The constraints on partitions are as follows:

pg_pathman automatically completes the conversion. For traditional inheritance, expressions similar to `select * from part_test where crt_time = '2016-10-25 00:00:00'::timestamp;` cannot filter partitions.

```
postgres=# \d+ part_test_122
```

```
Table "public.part_test_122"
Column |          Type          | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
id      | integer                |           |         |              |
info    | text                   |           |         |              |
crt_time | timestamp without time zone | not null | plain   |              |
```

Check constraints:

```
"pathman_part_test_122_3_check" CHECK (get_hash_part_idx(timestamp_hash(
crt_time), 128) = 122)
```

Inherits: part_test



Note:

When using hash partitioning, note the following items:

- All partition columns must contain the NOT NULL constraint.
- Run non-blocking data migration.
- After data migration is completed, disable the primary table.
- pg_pathman is not subject to expressions. So the command `select * from part_test where crt_time = '2016-10-25 00:00:00'::timestamp;` can also be used for hash partitioning.
- HASH partition columns are not limited to int type columns. The column types are automatically converted by a hash function.

3. Migrate data to a partition

If the data of the primary table is not migrated to partitions when the partitions are created, the data can be migrated to the partitions by calling a non-blocking migration function. Run the following commands:

```
with tmp as (delete from a primary table limit xx nowait returning *) insert into a
partition select * from tmp
```

You can also use `select array_agg(ctid) from a primary table limit xx for update nowait`. Then execute the `DELETE` and `INSERT` statements.

The function is as follows:

```
partition_table_concurrently(relation REGCLASS,          -- The OID of the primary
                             table.
                             batch_size INTEGER DEFAULT 1000, -- The number of records to copy
                             from the primary table at a time.
                             sleep_time FLOAT8 DEFAULT 1.0) -- The time interval between
                             migration attempts if one or more rows in the batch are locked by other queries.
                             pg_pathman waits for the specified time and tries again up to 60 times before quitting
                             .
```

Example:

```
postgres=# select partition_table_concurrently('part_test'::regclass,
        10000,
        1.0);
NOTICE: worker started, you can stop it with the following command: select
stop_concurrent_part_task('part_test');
partition_table_concurrently
-----
(1 row)
```

To stop the migration task, call the following function:

```
stop_concurrent_part_task(relation REGCLASS)
```

View the background data migration task.

```
postgres=# select * from pathman_concurrent_part_tasks;
userid | pid | dbid | relid | processed | status
-----+-----+-----+-----+-----+-----
(0 rows)
```

4. Split range partitions

If a partition is too large and you want to split the partition into two partitions, use the following method (only range partitions are supported):

```
split_range_partition(partition REGCLASS,          -- The OID of the partition.
                      split_value ANYELEMENT,      -- The split value.
```

```
partition_name TEXT DEFAULT NULL) -- The name of the new partition.
```

Example:

```
postgres=# \d+ part_test
               Table "public.part_test"
  Column |          Type          | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
id       | integer                |           |         |              |
info     | text                   |           |         |              |
crt_time | timestamp without time zone | not null | plain   |              |
Child tables: part_test_1,
               part_test_10,
               part_test_11,
               part_test_12,
               part_test_13,
               part_test_14,
               part_test_15,
               part_test_16,
               part_test_17,
               part_test_18,
               part_test_19,
               part_test_2,
               part_test_20,
               part_test_21,
               part_test_22,
               part_test_23,
               part_test_24,
               part_test_3,
               part_test_4,
               part_test_5,
               part_test_6,
               part_test_7,
               part_test_8,
               part_test_9

postgres=# \d+ part_test_1
               Table "public.part_test_1"
  Column |          Type          | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
id       | integer                |           |         |              |
info     | text                   |           |         |              |
crt_time | timestamp without time zone | not null | plain   |              |
Check constraints:
 "pathman_part_test_1_3_check" CHECK (crt_time >= '2016-10-25 00:00:00'::
timestamp without time zone AND crt_time < '2016-11-25 00:00:00'::timestamp
without time zone)
Inherits: part_test
```

Splitting

```
postgres=# select split_range_partition('part_test_1'::regclass,          -- The OID of the
partition.
               '2016-11-10 00:00:00'::timestamp, -- The split value.
               'part_test_1_2');                -- The name of the partitioned table.
split_range_partition
-----
{"2016-10-25 00:00:00","2016-11-25 00:00:00"}
```

(1 row)

Two tables that are created from splitting are as follows:

```
postgres=# \d+ part_test_1
Table "public.part_test_1"
Column |          Type          | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
id      | integer                |           |         |              |
info    | text                   |           |         |              |
crt_time | timestamp without time zone | not null | plain   |              |
Check constraints:
    "pathman_part_test_1_3_check" CHECK (crt_time >= '2016-10-25 00:00:00'::
timestamp without time zone AND crt_time < '2016-11-10 00:00:00'::timestamp
without time zone)
Inherits: part_test

postgres=# \d+ part_test_1_2
Table "public.part_test_1_2"
Column |          Type          | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
id      | integer                |           |         |              |
info    | text                   |           |         |              |
crt_time | timestamp without time zone | not null | plain   |              |
Check constraints:
    "pathman_part_test_1_2_3_check" CHECK (crt_time >= '2016-11-10 00:00:00'::
timestamp without time zone AND crt_time < '2016-11-25 00:00:00'::timestamp
without time zone)
Inherits: part_test
```

Data is automatically migrated to the other partition.

```
postgres=# select count(*) from part_test_1;
count
-----
   373
(1 row)

postgres=# select count(*) from part_test_1_2;
count
-----
   360
(1 row)
```

The inheritance relationship is as follows:

```
postgres=# \d+ part_test
Table "public.part_test"
Column |          Type          | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
id      | integer                |           |         |              |
info    | text                   |           |         |              |
crt_time | timestamp without time zone | not null | plain   |              |
Child tables: part_test_1,
               part_test_10,
               part_test_11,
               part_test_12,
               part_test_13,
```

```

part_test_14,
part_test_15,
part_test_16,
part_test_17,
part_test_18,
part_test_19,
part_test_1_2, -- The added table.
part_test_2,
part_test_20,
part_test_21,
part_test_22,
part_test_23,
part_test_24,
part_test_3,
part_test_4,
part_test_5,
part_test_6,
part_test_7,
part_test_8,
part_test_9

```

5. Merge range partitions

Only range partitions are supported. Call the following function:

Specify two partitions to be merged, which must be adjacent partitions.
`merge_range_partitions(partition1 REGCLASS, partition2 REGCLASS)`

Example:

```

postgres=# select merge_range_partitions('part_test_2'::regclass, 'part_test_12'::
regclass);
ERROR: merge failed, partitions must be adjacent
CONTEXT: PL/pgSQL function merge_range_partitions_internal(regclass,regclass,
regclass,anyelement) line 27 at RAISE
SQL statement "SELECT public.merge_range_partitions_internal($1, $2, $3, NULL::
timestamp without time zone)"
PL/pgSQL function merge_range_partitions(regclass,regclass) line 44 at EXECUTE
An error is returned because the partitions are not adjacent.

```

Adjacent partitions can be merged.

```

postgres=# select merge_range_partitions('part_test_1'::regclass, 'part_test_1_2'::
regclass);
merge_range_partitions
-----

```

(1 row)

After the merge is completed, one of the partitions are deleted.

```

postgres=# \d part_test_1_2
Did not find any relation named "part_test_1_2".

postgres=# \d part_test_1
      Table "public.part_test_1"
  Column |          Type          | Modifiers
-----+-----+-----
 id      | integer                |
 info    | text                   |
 crt_time | timestamp without time zone | not null
Check constraints:

```

```
"pathman_part_test_1_3_check" CHECK (crt_time >= '2016-10-25 00:00:00'::
timestamp without time zone AND crt_time < '2016-11-25 00:00:00'::timestamp
without time zone)
Inherits: part_test

postgres=# select count(*) from part_test_1;
count
-----
    733
(1 row)
```

6. Add a range partition following the last partition

There are several methods to add partitions for primary tables that have been previously partitioned. One method is to add partitions following the last partition.

When a new partition is added, the interval of the previously partitioned table will be used. You can query the interval of each partitioned table when it is created for the first time by running the `pathman_config` command:

```
postgres=# select * from pathman_config;
 partrel | attname | parttype | range_interval
-----+-----+-----+-----
 part_test | crt_time |      2 | 1 mon
(1 row)
```

Add a new range partition (the tablespace cannot be specified).

```
append_range_partition(parent REGCLASS,      -- The OID of the primary table.
                       partition_name TEXT DEFAULT NULL, -- (Optional) The name of the new
partition. Default value: null.
                       tablespace TEXT DEFAULT NULL) -- (Optional) The tablespace where
the new partition is stored. Default value: null.
```

Example:

```
postgres=# select append_range_partition('part_test'::regclass);
append_range_partition
-----
public.part_test_25
(1 row)

postgres=# \d+ part_test_25
Table "public.part_test_25"
Column |          Type          | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
id      | integer                |           |         |              |
info    | text                   |           |         |              |
crt_time | timestamp without time zone | not null | plain   |              |
Check constraints:
"pathman_part_test_25_3_check" CHECK (crt_time >= '2018-10-25 00:00:00'::
timestamp without time zone AND crt_time < '2018-11-25 00:00:00'::timestamp
without time zone)
Inherits: part_test

postgres=# \d+ part_test_24
Table "public.part_test_24"
```

Column	Type	Modifiers	Storage	Stats target	Description
id	integer	plain			
info	text	extended			
crt_time	timestamp without time zone	not null	plain		
Check constraints:					
"pathman_part_test_24_3_check" CHECK (crt_time >= '2018-09-25 00:00:00'::timestamp without time zone AND crt_time < '2018-10-25 00:00:00'::timestamp without time zone)					
Inherits: part_test					

7. Add a range partition at the beginning of partitions

Add a partition at the beginning of the partitions. The function is as follows:

```
prepend_range_partition(parent REGCLASS,
                        partition_name TEXT DEFAULT NULL,
                        tablespace TEXT DEFAULT NULL)
```

Example:

```
postgres=# select prepend_range_partition('part_test'::regclass);
prepend_range_partition
-----
public.part_test_26
(1 row)

postgres=# \d+ part_test_26
Table "public.part_test_26"
Column |      Type      | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
id      | integer        |          |         |              |
info    | text           |          |         |              |
crt_time | timestamp without time zone | not null | plain   |              |
Check constraints:
    "pathman_part_test_26_3_check" CHECK (crt_time >= '2016-09-25 00:00:00'::timestamp without time zone AND crt_time < '2016-10-25 00:00:00'::timestamp without time zone)
Inherits: part_test

postgres=# \d+ part_test_1
Table "public.part_test_1"
Column |      Type      | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
id      | integer        |          |         |              |
info    | text           |          |         |              |
crt_time | timestamp without time zone | not null | plain   |              |
Check constraints:
    "pathman_part_test_1_3_check" CHECK (crt_time >= '2016-10-25 00:00:00'::timestamp without time zone AND crt_time < '2016-11-25 00:00:00'::timestamp without time zone)
Inherits: part_test
```

8. Add a partition

You can create new partitions by specifying the start value of the partitions. New partitions can be created if the ranges do not overlap with existing partitions. This

method allows you to create non-continuous partitions. For example, if the range of existing partitions are from 2010 to 2015, you can create a new partition from 2020. You do not need to create a partition between 2015 and 2020. The function is as follows:

```
add_range_partition(relation REGCLASS, -- The OID of the primary table.
                    start_value ANYELEMENT, -- The start value.
                    end_value ANYELEMENT, -- The end value.
                    partition_name TEXT DEFAULT NULL, -- The name of the partition.
                    tablespace TEXT DEFAULT NULL) -- The name of the tablespace in which a
partition resides.
```

Example:

```
postgres=# select add_range_partition('part_test'::regclass, -- The OID of the
primary table.
'2020-01-01 00:00:00'::timestamp, -- The start value.
'2020-02-01 00:00:00'::timestamp); -- The end value.
add_range_partition
-----
public.part_test_27
(1 row)

postgres=# \d+ part_test_27
Table "public.part_test_27"
Column |          Type          | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
id      | integer                |           | plain   |               |
info    | text                   |           | extended|               |
crt_time | timestamp without time zone | not null | plain   |               |
Check constraints:
"pathman_part_test_27_3_check" CHECK (crt_time >= '2020-01-01 00:00:00'::
timestamp without time zone AND crt_time < '2020-02-01 00:00:00'::timestamp
without time zone)
Inherits: part_test
```

9. Delete a partition

To delete a single partition range, call the following function:

```
drop_range_partition(partition TEXT, -- The name of the partition to be deleted.
                     delete_data BOOLEAN DEFAULT TRUE) -- Specifies whether to delete the
data of the partition. If you set the value to FALSE, the data of the partition is migrated
to the primary table.
```

Drop RANGE partition and all of its data if delete_data is true.

Example:

```
Delete a partition and migrate data of the partition to the primary table.
postgres=# select drop_range_partition('part_test_1',false);
NOTICE: 733 rows copied from part_test_1
drop_range_partition
-----
part_test_1
(1 row)

postgres=# select drop_range_partition('part_test_2',false);
```



```

NOTICE: 720 rows copied from part_test_2
drop_range_partition
-----
part_test_2
(1 row)

postgres=# select count(*) from part_test;
count
-----
10000
(1 row)

Delete a partition and the data of the partition without migrating the data to the
primary table.
postgres=# select drop_range_partition('part_test_3',true);
drop_range_partition
-----
part_test_3
(1 row)

postgres=# select count(*) from part_test;
count
-----
9256
(1 row)

postgres=# select count(*) from only part_test;
count
-----
1453
(1 row)

```

Delete all partitions and specify whether to migrate data to the primary table. The function is as follows:

```

drop_partitions(parent REGCLASS,
                delete_data BOOLEAN DEFAULT FALSE)

```

Drop partitions of the parent table (both foreign and local relations).
If delete_data is false, the data is copied to the parent table first.
Default is false.

Example:

```

postgres=# select drop_partitions('part_test'::regclass, false); -- Delete all partitions
and migrate the data to the primary table.
NOTICE: function public.part_test_upd_trig_func() does not exist, skipping
NOTICE: 744 rows copied from part_test_4
NOTICE: 672 rows copied from part_test_5
NOTICE: 744 rows copied from part_test_6
NOTICE: 720 rows copied from part_test_7
NOTICE: 744 rows copied from part_test_8
NOTICE: 720 rows copied from part_test_9
NOTICE: 744 rows copied from part_test_10
NOTICE: 744 rows copied from part_test_11
NOTICE: 720 rows copied from part_test_12
NOTICE: 744 rows copied from part_test_13
NOTICE: 507 rows copied from part_test_14
NOTICE: 0 rows copied from part_test_15
NOTICE: 0 rows copied from part_test_16
NOTICE: 0 rows copied from part_test_17

```

```

NOTICE: 0 rows copied from part_test_18
NOTICE: 0 rows copied from part_test_19
NOTICE: 0 rows copied from part_test_20
NOTICE: 0 rows copied from part_test_21
NOTICE: 0 rows copied from part_test_22
NOTICE: 0 rows copied from part_test_23
NOTICE: 0 rows copied from part_test_24
NOTICE: 0 rows copied from part_test_25
NOTICE: 0 rows copied from part_test_26
NOTICE: 0 rows copied from part_test_27
drop_partitions
-----
          24
(1 row)

postgres=# select count(*) from part_test;
count
-----
  9256
(1 row)

postgres=# \dt part_test_4
No matching relations found.

```

10. Attach a table to a partitioned table

Attach a table to a partitioned primary table. The table must have the same schema as the primary table, including the dropped columns (check the consistency of pg_attribute). The function is as follows:

```

attach_range_partition(relation REGCLASS, -- The OID of the primary table.
                      partition REGCLASS, -- The OID of the partition table.
                      start_value ANYELEMENT, -- The start value.
                      end_value ANYELEMENT) -- The start value.

```

Example:

```

postgres=# create table part_test_1 (like part_test including all);
CREATE TABLE
postgres=# \d+ part_test
               Table "public.part_test"
  Column |          Type          | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
 id      | integer                |           | plain   |              |
 info    | text                   |           | extended|              |
 crt_time | timestamp without time zone | not null | plain   |              |
postgres=# \d+ part_test_1
               Table "public.part_test_1"
  Column |          Type          | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
 id      | integer                |           | plain   |              |
 info    | text                   |           | extended|              |
 crt_time | timestamp without time zone | not null | plain   |              |
postgres=# select attach_range_partition('part_test'::regclass, 'part_test_1'::regclass,
'2019-01-01 00:00:00'::timestamp, '2019-02-01 00:00:00'::timestamp);
attach_range_partition

```

```

-----
part_test_1
(1 row)

When the table is attached,
inheritance relationships and constraints are created automatically.
postgres=# \d+ part_test_1
          Table "public.part_test_1"
  Column |          Type          | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
id       | integer                |           |         |              |
info     | text                   |           |         |              |
crt_time | timestamp without time zone | not null | plain   |              |
Check constraints:
"pathman_part_test_1_3_check" CHECK (crt_time >= '2019-01-01 00:00:00'::
timestamp without time zone AND crt_time < '2019-02-01 00:00:00'::timestamp
without time zone)
Inherits: part_test

```

11. Detach a partition from the primary table (convert the partition into a normal table)

Delete a partition from the primary table inheritance. The data is not deleted. The inheritance and constraints are deleted. The function is as follows:

```
detach_range_partition(partition REGCLASS) -- Specify the name of the partition and
convert the partition to a normal table.
```

Example:

```

postgres=# select count(*) from part_test;
count
-----
  9256
(1 row)

postgres=# select count(*) from part_test_2;
count
-----
   733
(1 row)

postgres=# select detach_range_partition('part_test_2');
detach_range_partition
-----
part_test_2
(1 row)

postgres=# select count(*) from part_test_2;
count
-----
   733
(1 row)

postgres=# select count(*) from part_test;
count
-----
  8523

```

(1 row)

12. Permanently disable the pg_pathman plug-in for a partitioned table

You can disable the pg_pathman plug-in for a single partitioned primary table. The function is as follows:

```
disable_pathman_for(relation TEXT)
```

Permanently disable pg_pathman partitioning mechanism for the specified parent table and remove the insert trigger if it exists.
All partitions and data remain unchanged.

```
postgres=# \sf disable_pathman_for
CREATE OR REPLACE FUNCTION public.disable_pathman_for(parent_relid regclass)
RETURNS void
LANGUAGE plpgsql
STRICT
AS $function$
BEGIN
    PERFORM public.validate_relname(parent_relid);

    DELETE FROM public.pathman_config WHERE partrel = parent_relid;
    PERFORM public.drop_triggers(parent_relid);

    /* Notify backend about changes */
    PERFORM public.on_remove_partitions(parent_relid);
END
$function$
```

Example:

```
postgres=# select disable_pathman_for('part_test');
NOTICE: drop cascades to 23 other objects
DETAIL: drop cascades to trigger part_test_upd_trig on table part_test_3
drop cascades to trigger part_test_upd_trig on table part_test_4
drop cascades to trigger part_test_upd_trig on table part_test_5
drop cascades to trigger part_test_upd_trig on table part_test_6
drop cascades to trigger part_test_upd_trig on table part_test_7
drop cascades to trigger part_test_upd_trig on table part_test_8
drop cascades to trigger part_test_upd_trig on table part_test_9
drop cascades to trigger part_test_upd_trig on table part_test_10
drop cascades to trigger part_test_upd_trig on table part_test_11
drop cascades to trigger part_test_upd_trig on table part_test_12
drop cascades to trigger part_test_upd_trig on table part_test_13
drop cascades to trigger part_test_upd_trig on table part_test_14
drop cascades to trigger part_test_upd_trig on table part_test_15
drop cascades to trigger part_test_upd_trig on table part_test_16
drop cascades to trigger part_test_upd_trig on table part_test_17
drop cascades to trigger part_test_upd_trig on table part_test_18
drop cascades to trigger part_test_upd_trig on table part_test_19
drop cascades to trigger part_test_upd_trig on table part_test_20
drop cascades to trigger part_test_upd_trig on table part_test_21
drop cascades to trigger part_test_upd_trig on table part_test_22
drop cascades to trigger part_test_upd_trig on table part_test_23
drop cascades to trigger part_test_upd_trig on table part_test_24
drop cascades to trigger part_test_upd_trig on table part_test_25
disable_pathman_for
-----
(1 row)
```

```

postgres=# \d+ part_test
               Table "public.part_test"
  Column |          Type          | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
id       | integer                |           |         |              |
info     | text                   |           |         |              |
crt_time | timestamp without time zone | not null | plain   |              |
Child tables: part_test_10,
               part_test_11,
               part_test_12,
               part_test_13,
               part_test_14,
               part_test_15,
               part_test_16,
               part_test_17,
               part_test_18,
               part_test_19,
               part_test_20,
               part_test_21,
               part_test_22,
               part_test_23,
               part_test_24,
               part_test_25,
               part_test_26,
               part_test_27,
               part_test_28,
               part_test_29,
               part_test_3,
               part_test_30,
               part_test_31,
               part_test_32,
               part_test_33,
               part_test_34,
               part_test_35,
               part_test_4,
               part_test_5,
               part_test_6,
               part_test_7,
               part_test_8,
               part_test_9

postgres=# \d+ part_test_10
               Table "public.part_test_10"
  Column |          Type          | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
id       | integer                |           |         |              |
info     | text                   |           |         |              |
crt_time | timestamp without time zone | not null | plain   |              |
Check constraints:
  "pathman_part_test_10_3_check" CHECK (crt_time >= '2017-06-25 00:00:00'::
timestamp without time zone AND crt_time < '2017-07-25 00:00:00'::timestamp
without time zone)

```

Inherits: part_test

After the pg_pathman plug-in is disabled, the inheritance and constraints remain unchanged. The pg_pathman plug-in does not intervene in the custom scan execution plan. The execution plan after the pg_pathman plug-in is disabled:

```
postgres=# explain select * from part_test where crt_time='2017-06-25 00:00:00'::timestamp;
```

QUERY PLAN

```
-----
Append (cost=0.00..16.00 rows=2 width=45)
-> Seq Scan on part_test (cost=0.00..0.00 rows=1 width=45)
    Filter: (crt_time = '2017-06-25 00:00:00'::timestamp without time zone)
-> Seq Scan on part_test_10 (cost=0.00..16.00 rows=1 width=45)
    Filter: (crt_time = '2017-06-25 00:00:00'::timestamp without time zone)
(5 rows)
```



Notice:

The disable_pathman_for operation is irreversible. Proceed with caution.

Advanced partition management

1. Disable a primary table

After all data of a primary table is migrated to the partitions, you can disable the primary table. The function is as follows:

```
set_enable_parent(relation REGCLASS, value BOOLEAN)
```

Include/exclude parent table into/from query plan.

In original PostgreSQL planner parent table is always included into query plan even if it's empty which can lead to additional overhead.

You can use disable_parent() if you are never going to use parent table as a storage.

Default value depends on the partition_data parameter that was specified during initial partitioning in create_range_partitions() or create_partitions_from_range() functions.

If the partition_data parameter was true then all data have already been migrated to partitions and parent table disabled.

Otherwise it is enabled.

Example:

```
select set_enable_parent('part_test', false);
```

2. Auto partition propagation

Auto partition propagation is supported for range partitioned tables. If the inserted data is not within the range of the existing partitions, a partition is automatically created.

```
set_auto(relation REGCLASS, value BOOLEAN)
```

Enable/disable auto partition propagation (only for RANGE partitioning).

It is enabled by default.

Example:

```
postgres=# \d+ part_test
               Table "public.part_test"
  Column |          Type          | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
 id      | integer                |           |         |              |
 info    | text                   |           |         |              |
 crt_time | timestamp without time zone | not null | plain   |              |
Child tables: part_test_10,
               part_test_11,
               part_test_12,
               part_test_13,
               part_test_14,
               part_test_15,
               part_test_16,
               part_test_17,
               part_test_18,
               part_test_19,
               part_test_20,
               part_test_21,
               part_test_22,
               part_test_23,
               part_test_24,
               part_test_25,
               part_test_26,
               part_test_3,
               part_test_4,
               part_test_5,
               part_test_6,
               part_test_7,
               part_test_8,
               part_test_9

postgres=# \d+ part_test_26
               Table "public.part_test_26"
  Column |          Type          | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
 id      | integer                |           |         |              |
 info    | text                   |           |         |              |
 crt_time | timestamp without time zone | not null | plain   |              |
```

Check constraints:

```
"pathman_part_test_26_3_check" CHECK (crt_time >= '2018-09-25 00:00:00'::timestamp without time zone AND crt_time < '2018-10-25 00:00:00'::timestamp without time zone)
```

Inherits: part_test

```
postgres=# \d+ part_test_25
```

```
Table "public.part_test_25"
Column |          Type          | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
id      | integer                |           |         |              |
info    | text                   |           |         |              |
crt_time | timestamp without time zone | not null | plain   |              |
```

Check constraints:

```
"pathman_part_test_25_3_check" CHECK (crt_time >= '2018-08-25 00:00:00'::timestamp without time zone AND crt_time < '2018-09-25 00:00:00'::timestamp without time zone)
```

Inherits: part_test

When the inserted data is beyond the partitioning range, a large number of new partitions are created based on the interval when the primary table is partitioned.

```
postgres=# insert into part_test values (1,'test','2222-01-01'::timestamp);
```

After the data is inserted, a large number of partitions are created because the range of the inserted values is too large.

```
postgres=# \d+ part_test
```

```
Table "public.part_test"
Column |          Type          | Modifiers | Storage | Stats target | Description
-----+-----+-----+-----+-----+-----
id      | integer                |           |         |              |
info    | text                   |           |         |              |
crt_time | timestamp without time zone | not null | plain   |              |
```

Child tables: part_test_10,

part_test_100,

part_test_1000,

part_test_1001,

.....

A large number of partitions



Note:

We recommend that you disable auto partition propagation for range partitioning because inappropriate auto propagation may consume a lot of time.

3. Callback functions that are triggered for each partition creation

A callback function is a function that is automatically triggered for each partition creation. For example, a callback function can record the DDL statements that you use to run logical replication and store the statements in a table. The callback function is as follows:

```
set_init_callback(relation REGCLASS, callback REGPROC DEFAULT 0)
```

Set partition creation callback to be invoked for each attached or created partition (both HASH and RANGE).

The callback must have the following signature:

`part_init_callback(args JSONB) RETURNS VOID.`

Parameter `arg` consists of several fields whose presence depends on partitioning type:

```
/* RANGE-partitioned table abc (child abc_4) */
{
  "parent": "abc",
  "parttype": "2",
  "partition": "abc_4",
  "range_max": "401",
  "range_min": "301"
}

/* HASH-partitioned table abc (child abc_0) */
{
  "parent": "abc",
  "parttype": "1",
  "partition": "abc_0"
}
```

Example:

```
Callback function
postgres=# create or replace function f_callback_test(jsonb) returns void as
$$
declare
begin
  create table if not exists rec_part_ddl(id serial primary key, parent name, parttype int
, partition name, range_max text, range_min text);
  if ($1->>'parttype')::int = 1 then
    raise notice 'parent: %, parttype: %, partition: %', $1->>'parent', $1->>'parttype', $1-
->>'partition';
    insert into rec_part_ddl(parent, parttype, partition) values (($1->>'parent')::name,
($1->>'parttype')::int, ($1->>'partition')::name);
  elsif ($1->>'parttype')::int = 2 then
    raise notice 'parent: %, parttype: %, partition: %, range_max: %, range_min: %', $1-
->>'parent', $1->>'parttype', $1->>'partition', $1->>'range_max', $1->>'range_min';
    insert into rec_part_ddl(parent, parttype, partition, range_max, range_min) values
(($1->>'parent')::name, ($1->>'parttype')::int, ($1->>'partition')::name, $1->>'
range_max', $1->>'range_min');
  end if;
end;
$$ language plpgsql strict;

Test table
postgres=# create table tt(id int, info text, crt_time timestamp not null);
CREATE TABLE

Set the callback function for the test table.
select set_init_callback('tt'::regclass, 'f_callback_test'::regproc);

Create a partition
postgres=# select
create_range_partitions('tt'::regclass,          -- The OID of the primary table.
  'crt_time',          -- The column name of the partition.
  '2016-10-25 00:00:00'::timestamp, -- The start value.
  interval '1 month',  -- The interval of the interval type, applicable
to ingestion-time partitioned tables.
  24,                  -- The number of partitions.
  false);
```

create_range_partitions

24

(1 row)

Check whether the callback function is called.

postgres=# select * from rec_part_ddl;

id | parent | parttype | partition | range_max | range_min

-----+-----+-----+-----+-----+-----					
1	tt	2	tt_1	2016-11-25 00:00:00	2016-10-25 00:00:00
2	tt	2	tt_2	2016-12-25 00:00:00	2016-11-25 00:00:00
3	tt	2	tt_3	2017-01-25 00:00:00	2016-12-25 00:00:00
4	tt	2	tt_4	2017-02-25 00:00:00	2017-01-25 00:00:00
5	tt	2	tt_5	2017-03-25 00:00:00	2017-02-25 00:00:00
6	tt	2	tt_6	2017-04-25 00:00:00	2017-03-25 00:00:00
7	tt	2	tt_7	2017-05-25 00:00:00	2017-04-25 00:00:00
8	tt	2	tt_8	2017-06-25 00:00:00	2017-05-25 00:00:00
9	tt	2	tt_9	2017-07-25 00:00:00	2017-06-25 00:00:00
10	tt	2	tt_10	2017-08-25 00:00:00	2017-07-25 00:00:00
11	tt	2	tt_11	2017-09-25 00:00:00	2017-08-25 00:00:00
12	tt	2	tt_12	2017-10-25 00:00:00	2017-09-25 00:00:00
13	tt	2	tt_13	2017-11-25 00:00:00	2017-10-25 00:00:00
14	tt	2	tt_14	2017-12-25 00:00:00	2017-11-25 00:00:00
15	tt	2	tt_15	2018-01-25 00:00:00	2017-12-25 00:00:00
16	tt	2	tt_16	2018-02-25 00:00:00	2018-01-25 00:00:00
17	tt	2	tt_17	2018-03-25 00:00:00	2018-02-25 00:00:00
18	tt	2	tt_18	2018-04-25 00:00:00	2018-03-25 00:00:00
19	tt	2	tt_19	2018-05-25 00:00:00	2018-04-25 00:00:00
20	tt	2	tt_20	2018-06-25 00:00:00	2018-05-25 00:00:00
21	tt	2	tt_21	2018-07-25 00:00:00	2018-06-25 00:00:00
22	tt	2	tt_22	2018-08-25 00:00:00	2018-07-25 00:00:00
23	tt	2	tt_23	2018-09-25 00:00:00	2018-08-25 00:00:00
24	tt	2	tt_24	2018-10-25 00:00:00	2018-09-25 00:00:00

(24 rows)

16.3 Enable the zhparser plug-in

This topic describes how to enable the zhparser plug-in and customize a Chinese word segment dictionary in POLARDB for PostgreSQL.

Enable the zhparser plug-in

Execute the following statements to enable the zhparser plug-in:

```
CREATE EXTENSION zhparser;
CREATE TEXT SEARCH CONFIGURATION testzhcfg (PARSER = zhparser);
ALTER TEXT SEARCH CONFIGURATION testzhcfg ADD MAPPING FOR n,v,a,i,e,l WITH simple;
--Optional parameter configuration
alter role all set zhparser.multi_short=on;
--Perform a simple test
SELECT * FROM ts_parse('zhparser', 'hello world! 2010年保障房建设在全国范围内获全面启动，从中央到地方纷纷加大了保障房的建设和投入力度。2011年，保障房进入了更大规模的建设阶段。住房城乡建设部党组书记、部长姜伟新去年底在全国住房城乡建设工作会议上表示，要继续推进保障性安居工程建设。');
SELECT to_tsvector('testzhcfg', "今年保障房新开工数量虽然有所下调，但实际的年度在建规模以及竣工规模会超以往年份，相对应的对资金的需求也会创历史纪录。" 陈国强说。在他看来，与2011年相比，2012年的保障房建设在资金配套上的压力将更为严峻。');
```

```
SELECT to_tsquery('testzhcfg', '保障房资金压力');
```

Execute the following statements to use the zhparser plug-in to run a full-text index:

```
--Create a full-text index for the name field of table T1
create index idx_t1 on t1 using gin (to_tsvector('zhcfg',upper(name) ));
--Use the full-text index
select * from t1 where to_tsvector('zhcfg',upper(t1.name)) @@ to_tsquery('zhcfg','(防火)');
```

Customize a Chinese word segment dictionary

Execute the following statements to customize a Chinese word segment dictionary

```
-- The segmentation result
SELECT to_tsquery('testzhcfg', '保障房资金压力');
-- Insert a new word segment to the dictionary
insert into pg_ts_custom_word values ('保障房资');
-- Make the inserted word segment take effect
select zhprs_sync_dict_xdb();
-- End the connection
\c
-- Requery to obtain new segmentation results
SELECT to_tsquery('testzhcfg', '保障房资金压力');
```

Instructions to use custom word segments:

- A maximum of 1 million custom word segments can be added. If the number of word segments exceed the limit, the word segments outside the limit are not processed. Ensure that the number of word segments is within this range. The custom and default word segmentation dictionaries take effect at the same time.
- Each word segment can be a maximum of 128 bytes in length. The section after the 128th byte will be truncated.
- After adding, deleting, or changing word segments, execute the `select zhprs_sync_dict_xdb();` statement and re-establish a connection to make the operation take effect.