腾讯云数据仓库 TCHouse-X

操作指南

腾讯云

文档目录

操作指南
访问授权

最近更新时间：2026-05-06 16:28:12

首次使用 TCHouse-X 前，请确保已通过主账号创建必要的关联角色，以获取云资源访问授权。

> **说明：**
> 腾讯云数据仓库 TCHouse-X 目前处于邀测中。您可 申请试用 ，我们将尽快为您发送测试邀请。

前置条件

- 已在腾讯云国内站 注册账号 并完成**实名认证**。
- 已收到腾讯云数据仓库 TCHouse-X（以下简称 TCHouse-X）的**测试邀请**。

Step 1：访问 TCHouse-X 产品控制台

通过 TCHouse-X 产品控制台 直接访问产品页面。**初始状态页面如下：**

创建角色前的TCHouse-X产品页面

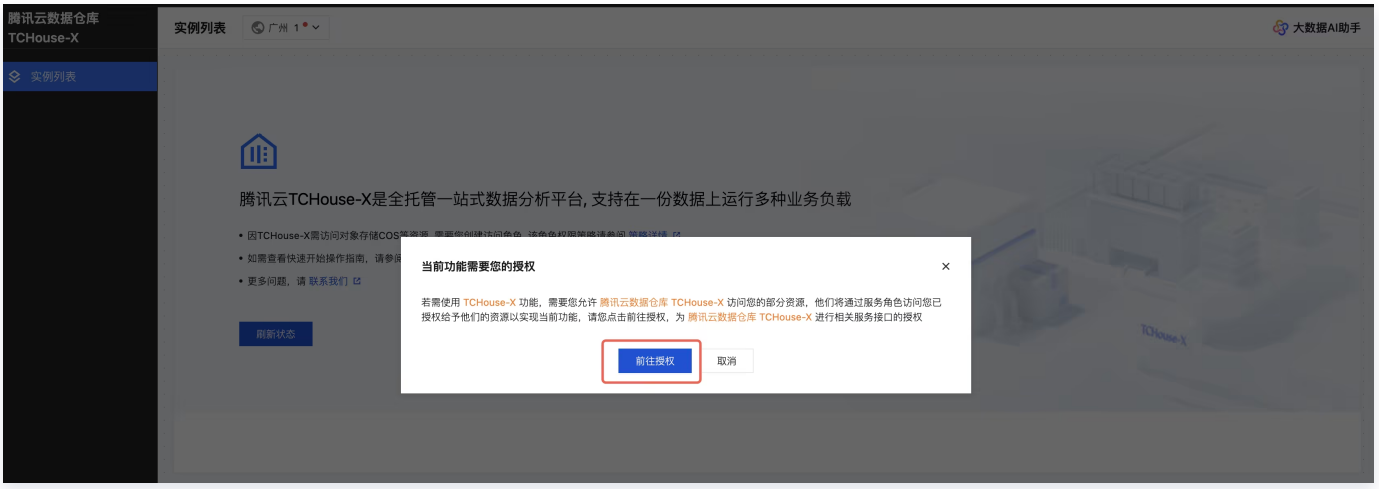
> **说明：**
> 若未收到 TCHouse-X 测试邀请，控制台页面将无法正常使用。

Step 2：创建 TCHouse-X 产品服务角色

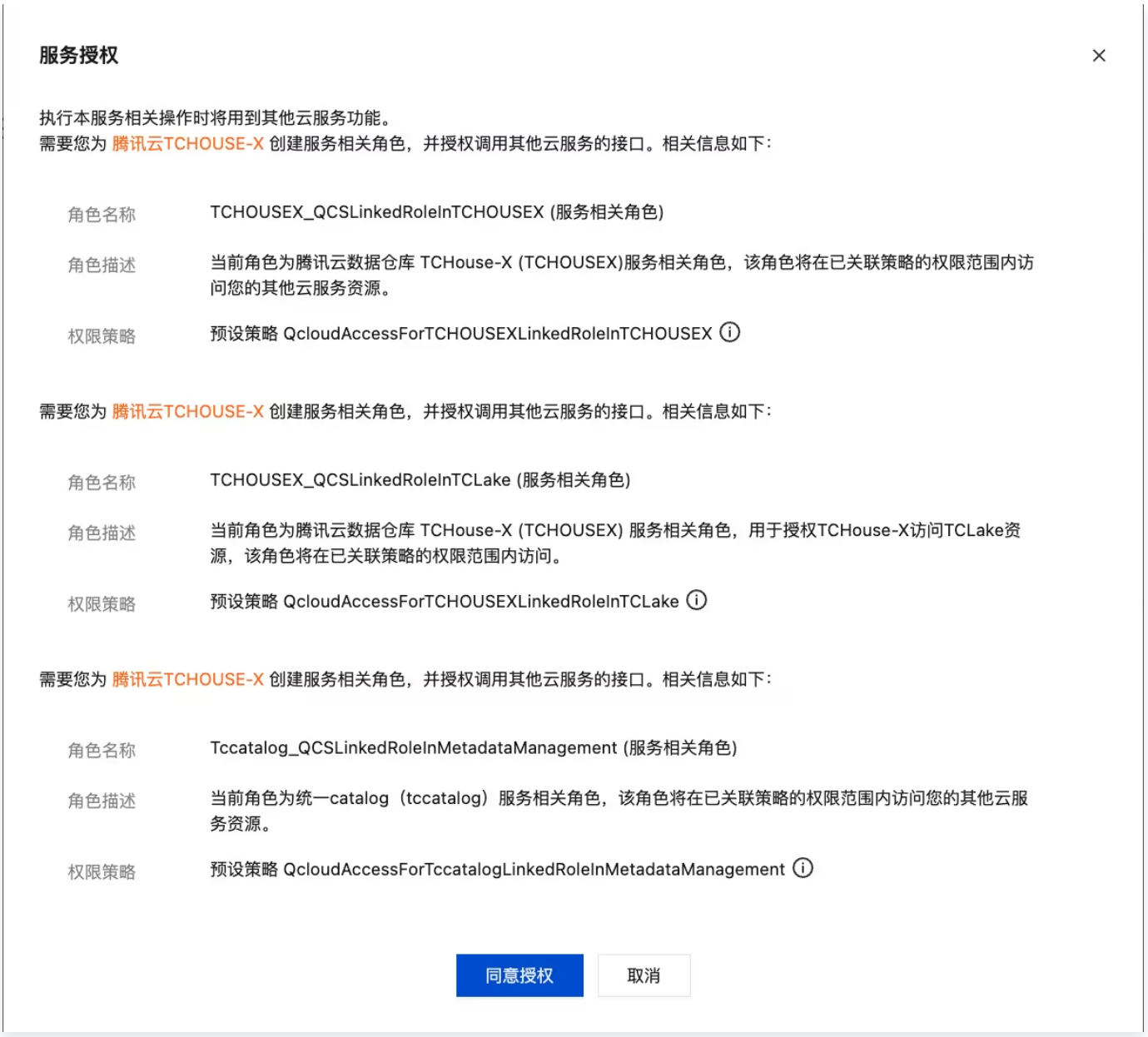
使用 TCHouse-X 过程中需要访问您的部分腾讯云资源，因此需通过创建产品服务角色的方式，授予 TCHouse-X 相关权限。

1. 单击**创建角色**。

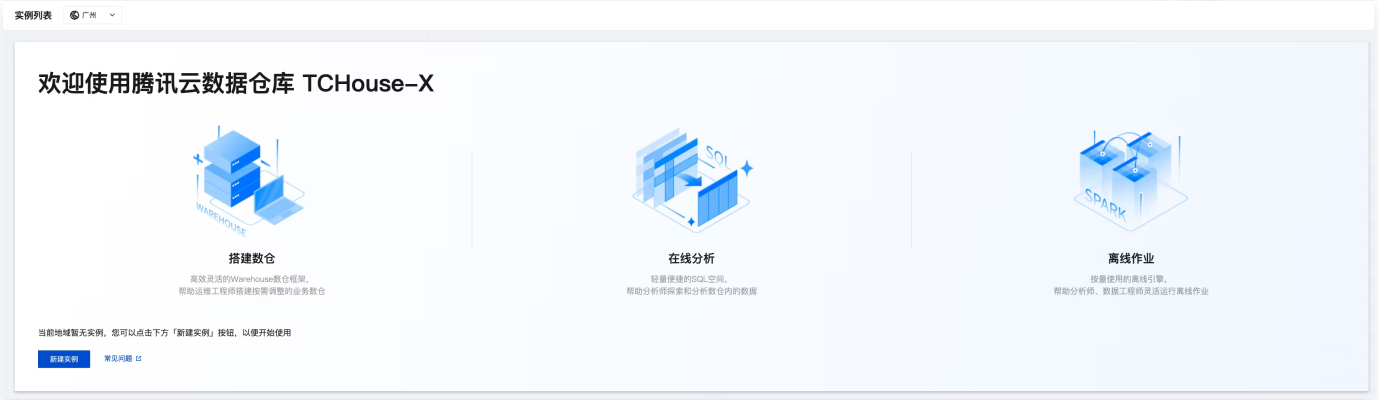
2. 在二次确认弹窗中，单击**前往授权**。



3. 在跳转的腾讯云访问管理（CAM）页面中，单击**同意授权**，此步骤将授予 TCHouse-X 访问您的部分腾讯云资源的权限。



4. 授权完成后，自动跳转回 TCHouse-X 产品页面，单击**已完成授权**。完成授权后，即可开始使用 TCHouse-X。



权限说明

TCHouse-X 依赖以下服务相关角色获取跨服务访问权限，每个角色默认关联一条预设策略：

服务角色	预设策略	策略简述
TCHOUSEX_QCSLinkedRoleInTCHOUSEX	QcloudAccessForTCHOUSEXLinkedRoleInTCHOUSEX	授予 TCHouse-X 访问对象存储（COS）和云 HDFS（CHDFS）资源的权限
TCHOUSEX_QCSLinkedRoleInTCLake	QcloudAccessForTCHOUSEXLinkedRoleInTCLake	授予 TCHouse-X 访问 TCCatalog 资源的权限
Tccatalog_QCSLinkedRoleInMetadataManagement	QcloudAccessForTccatalogLinkedRoleInMetadataManagement	授予 TCCatalog 访问其他云服务资源的权限

预设策略详情

QcloudAccessForTCHOUSEXLinkedRoleInTCHOUSEX 策略内容

```
{
    "version": "2.0",
    "statement": [
        {
            "effect": "allow",
            "action": [
                "cos:GetService",
                "cos:GetBucket",
                "cos:ListMultipartUploads",
                "cos:GetObject*",
                "cos:HeadObject",
                "cos:GetBucketObjectVersions",
                "cos:OptionsObject",
                "cos:ListParts",
                "cos:DeleteObject",
                "cos:PostObject",
                "cos:PostObjectRestore",
                "cos:PutObject*",
                "cos:InitiateMultipartUpload",
                "cos:UploadPart",
                "cos:UploadPartCopy",
                "cos:CompleteMultipartUpload",
                "cos:AbortMultipartUpload",
                "cos:DeleteMultipleObjects",
                "cos:AppendObject",
                "cos:HeadBucket",
                "cos:GetBucket*",
                "cos:PutBucket*",
                "cos:DeleteBucket*",
```

```
                "cos:RenameObject",
                "chdfs:DescribeMountPoint",
                "chdfs:DescribeFileSystem",
                "chdfs:DescribeAccessGroups",
                "chdfs:DescribeAccessRules",
                "chdfs:ModifyFileSystem",
                "chdfs:ModifyAccessRules",
                "chdfs:CreateAccessGroup",
                "chdfs:CreateAccessRules",
                "chdfs:AssociateAccessGroups",
                "chdfs:DisassociateAccessGroups",
                "chdfs:DeleteAccessGroup",
                "chdfs:DeleteAccessRules"
            ],
            "resource": "*"
        }
    ]
}
```

QcloudAccessForTCHOUSEXLinkedRoleInTCLake 策略内容

```
{
    "statement": [
        {
            "action": [
                "vpc:DescribeVpcEndPointService",
                "vpc:DescribeVpcEndPoint",
                "tccatalog:DropCatalog",
                "tccatalog:DescribeCatalog",
                "tccatalog:DescribeMetastoreInstances",
                "tccatalog:CreateCatalog",
                "tccatalog:CreateTCCatalogEndpoint",
                "tccatalog:DescribeCatalogs",
                "tccatalog:DescribeTccCatalog",
                "tccatalog:CreateUsers",
                "tccatalog:DescribeUsers",
                "tccatalog:DescribeRoles",
                "tccatalog:DescribeRolePermissionList",
                "tccatalog:DescribeCatalogNames",
```

```
                "tccatalog:DescribeTccCatalogs",
                "tccatalog:CreateRole",
                "tccatalog:DeleteRoles",
                "tccatalog:GrantRolesToUser",
                "tccatalog:GrantUsersToRole",
                "tccatalog:RevokeRolesFromUser",
                "tccatalog:RevokeUsersFromRole",
                "tccatalog:GrantPermissionToRole",
                "tccatalog:RevokePermissionToRole",
                "tccatalog:DeleteUsers",
                "tccatalog:ModifyUser",
                "tccatalog:ModifyRole",
                "tccatalog:CheckUserRoleGranted",
                "cam:ListMaskedSubAccounts",
                "tccatalog:DescribeStorageUsage",
                "tccatalog:SetMetadataObjectOwner",
                "tccatalog:DescribeMetastoreInstance",
                "tccatalog:GrantPermissionToUser",
                "tccatalog:RevokePermissionToUser",
                "tccatalog:DescribeRolesPrivilegeList",
                "tccatalog:CreateMetastoreInstance",
                "tccatalog:DescribeMetadataObjectsOwner",
                "tccatalog:DescribeMetadataObjectOwner",
                "tccatalog:UpdatePermissionToResource",
                "tccatalog:DescribePrivilegesPointList",
                "tccatalog:DescribeTccVipInternal",
                "tccatalog:CheckCatalogConnectivity",
                "tccatalog:CheckServiceRoleGranted",
                "tccatalog:CreateSchema",
                "tccatalog:CreateVolume",
                "tccatalog:DescribeRegionWhitelist",
                "tccatalog:DescribeSchema",
                "tccatalog:DescribeSchemaNames",
                "tccatalog:DescribeSupportCatalogType",
                "tccatalog:DescribeUsageStatistics",
                "tccatalog:ModifyCatalog",
                "tccatalog:DescribeFrontMenuWhitelist",
                "tccatalog:DescribeStorageUsageTrends",
                "tccatalog:AcceptTccVpcEndPointConnect",
                "tccatalog:BindTccVpcEndPointServiceWhiteList",
```

```
                "tccatalog:CheckUserExists",
                "tccatalog:DescribeCatalogNamesPage",
                "tccatalog:SyncAllCamUsers",
                "tccatalog:ModifyCatalogProperties",
                "tccatalog:AssociateTagsWithMetadataObject",
                "tccatalog:DescribeCatalogsByNames",
                "tccatalog:ModifyCatalogName",
                "tccatalog:DescribeSchemas",
                "tccatalog:DescribeTableNames",
                "tccatalog:DropTable",
                "tccatalog:DropSchema",
                "tccatalog:*"
            ],
            "effect": "allow",
            "resource": "*"
        }
    ],
    "version": "2.0"
}
```

QcloudAccessForTccatalogLinkedRoleInMetadataManagement 策略内容

```
{
    "statement": [
        {
            "action": [
                "vpc:DescribeRouteTable",
                "vpc:CreateRoute",
                "vpc:AcceptVpcPeeringConnection",
                "vpc:CreateVpcPeeringConnectionEx",
                "vpc:CreateVpcPeeringConnection",
                "vpc:DeleteVpcPeeringConnection",
                "vpc:DeleteVpcPeeringConnectionEx",
                "vpc:AcceptVpcPeeringConnectionEx",
                "vpc:DescribeVpcPeeringConnections",
                "vpc:DescribeAssistantCidr",
                "vpc:DescribeVpcEx",
                "vpc:DescribeVpcEndPoint",
                "vpc:CreateVpcEndPoint",
```

```
                "vpc:DeleteVpcEndPoint",
                "dlc:GrantDLCCatalogAccess",
                "cos:GetBucket",
                "cos:GetService",
                "cos:HeadBucket",
                "cos:HeadObject",
                "cos:PutObject",
                "privatedns:DescribePrivateZoneList",
                "privatedns:DescribePrivateZone",
                "privatedns:DescribePrivateZoneRecordList",
                "privatedns:CreatePrivateZone",
                "privatedns:CreatePrivateZoneRecord",
                "privatedns:DescribeRecord",
                "cam:ListMaskedSubAccounts",
                "cam:DescribeRoleList",
                "cam:DescribeSubAccounts",
                "chdfs:CreateAccessGroup",
                "chdfs:DeleteAccessGroup",
                "chdfs:DescribeAccessGroup",
                "chdfs:DescribeAccessGroups",
                "chdfs:ModifyAccessGroup",
                "chdfs:CreateAccessRules",
                "chdfs:DeleteAccessRules",
                "chdfs:DescribeAccessRules",
                "chdfs:ModifyAccessRules",
                "vpc:DescribeSubnets",
                "vpc:DescribeSubnetEx",
                "cloudaudit:DescribeEvents",
                "vpc:CreateVpcEndPointService",
                "vpc:DescribeVpcEndPointService",
                "vpc:DeleteVpcEndPointService",
                "vpc:CreateVpcEndPointServiceWhiteList",
                "vpc:DescribeVpcEndPointServiceWhiteList",
                "vpc:DeleteVpcEndPointServiceWhiteList",
                "cos:ListMultipartUploads",
                "cos:GetObject*",
                "cos:GetBucketObjectVersions",
                "cos:OptionsObject",
                "cos:ListParts",
                "cos:DeleteObject*",
```

```
                "cos:PostObject",
                "cos:PostObjectRestore",
                "cos:PutObject*",
                "cos:InitiateMultipartUpload",
                "cos:UploadPart",
                "cos:UploadPartCopy",
                "cos:CompleteMultipartUpload",
                "cos:AbortMultipartUpload",
                "cos:DeleteMultipleObjects",
                "cos:AppendObject"
            ],
            "effect": "allow",
            "resource": "*"
        }
    ],
    "version": "2.0"
}
```

结语

至此，您已成功通过创建 TCHouse-X 产品服务角色完成授权，可以正式开始使用 TCHouse-X。

实例管理
实例创建与销毁

最近更新时间：2026-05-06 16:28:12

前置条件

您已成功创建 TCHouse-X 产品服务角色，详细操作请参见 创建产品服务角色 。

创建 TCHouse-X 实例

进入 TCHouse-X 控制台，单击**新建实例**后进入新建实例页面，您可自定义基本信息、用户管理、存储资源、计算资源等配置信息。

Step 1：填写基本信息

在基本信息中，您可自定义实例名称，选择地域、内核版本、计费模式、可用区、TCLake 服务状态、网络，在网络中选择VPC、子网。
字段说明与名称规范性要求如下：

字段名称	字段释义	字段可选值说明及规范性要求
实例名称	实例的名称	• 长度：6~36个字符，1个中文作为1个字符 • 内容：支持中文、英文、数字及特殊字符"-"、"_" • 唯一性：实例名称无唯一性约束，自动生成的实例ID为全局唯一
内核版本	实例运行的数据仓库内核版本	-
地域	实例所属的地域	• 暂仅支持广州，如需使用其他地域，可联系 TCHouse-X 产研团队
可用区	实例所属的可用区	• 暂仅支持广州七区，如需使用其他可用区，可联系 TCHouse-X 产研团队
TCLake 服务	TCHouse-X 存储 TCLake 服务的状态	• 配额限制： 单个主账号在单个地域内，仅支持存在一个 TCLake 服务 • 前置依赖： 开通 TCLake 服务是创建 TCHouse-X 实例的前提条件 • 快捷开通： 用户可在创建 TCHouse-X 实例的过程中，选择授权并自动开通 TCLake 服务

网络	实例所属的 VPC 及子网	• 支持选择主账号下的 VPC 及子网 • 所选子网的剩余可用IP数需多于 6 个

Step 2：选择用户管理体系

TCHouse-X 当前支持 3 种用户体系：

用户体系	说明
腾讯云 CAM 用户	用户可使用腾讯云子账号 ID，以及其密钥凭证（SecretId、SecretKey 或临时 Token）连接 TCHouse-X 实例。用户身份和权限的校验由腾讯云 CAM 服务负责。
自定义用户	允许用户设置自定义的用户名和密码，并将其绑定至腾讯云子账号。用户使用该自定义凭证连接 TCHouse-X 实例时，由 TCHouse-X 负责完成用户名和密码的校验。
自定义 LDAP 用户	将自有 LDAP 服务中的用户身份与腾讯云子账号绑定。用户可使用自有 LDAP 凭证连接 TCHouse-X 实例，由自有 LDAP 服务进行用户名和密码校验。

说明：
自定义 LDAP 用户体系目前处于实验性阶段。如需申请开通和使用此功能，请联系我们。

自定义 LDAP 用户认证开启方法

若用户体系选择自定义 LDAP 用户，新建实例时需配置自定义 LDAP 服务信息， LDAP 配置经过校验后，新建实例流程才可继续。
需配置的LDAP 服务信息如下：

配置项	说明	样例
LdapUrl	LDAP URL 是一个字符串，用于封装目录服务器的地址和端口	`ldap://ds.example.com:389`
LdapBaseDn	LDAP Base DN（基础辨别名）用于指定应用程序在 LDAP 目录中搜索用户和信息的层级起始点和边界范围。	`dc=admin, dc=com`
LdapBindDn	TCHouse-X 用于连接和认证到 LDAP 服务器，以便获取执行用户搜索和信息查询权限的服务账号的完整辨别名（DN）	`cn=admin, dc=example, dc=com`
LdapBindPasswd	与 LdapBindDn（服务账号）配对使用的密码，用于 TCHouse-X 完成对 LDAP 服务器的身份	-

	认证（即绑定操作）。	
LdapUserFilter	LDAP 搜索语法的过滤器，用于精确筛选出在 Base DN 范围内被识别为有效用户账号的条目。	-
VerifyClientCa	用于配置 LDAP 服务器在建立 TLS/SSL 安全连接时，是否需要校验客户端证书的有效性和可信度，以实现双向身份验证。默认关闭。	-
VerifyServerCa	用于配置 LDAP 客户端在建立 TLS/SSL 安全连接时，是否需要验证 LDAP 服务器证书的有效性和可信度，以确保连接到正确的服务器。默认关闭	-
服务端证书	LDAP 服务端认证开启时需填充	-

Step 3：配置存储资源

TCHouse-X 以 TCLake 作为存储基座，因此需配置以下 TCLake 相关参数：

配置项	说明
TCLake 权限	TCHouse-X 需要在 TCLake 上执行数据读写、配置管理及获取权限等操作。因此，开通 TCHouse-X 服务时，您必须授予其操作 TCLake 的权限。此权限的授权粒度为腾讯云主账号级别。
Catalog 名称	TCHouse-X 实例创建时，系统将在 TCLake 中新建 Catalog，用于保存实例内的表数据。该 Catalog 名称支持用户自定义，默认由系统自动生成。
存储类型	目前仅支持选择 TCLake 全托管的批流一体存储方案。

Step 4：配置数据安全

TCHouse-X 支持 SSL/TLS 传输加密功能，用于保障传输中数据的机密性和完整性，以满足高安全性和合规性要求。此功能默认处于关闭状态。

Step 5：配置计算资源

TCHouse-X 实例的 OLAP 任务使用 Warehouse 计算资源，离线任务使用 Serverless 计算资源。创建实例时支持同步创建 Warehouse 资源。
TCHouse-X 目前支持以下两种 Warehouse 计算资源：

计算资源类型	特点
标准 Warehouse 计算资源	由一个或多个相同规格的计算节点组成。它支持随时进行配置变更、暂停、重启、销毁等操作，并具备分时弹性、定时启停和自动扩缩容等多种弹性能力。

自适应 Warehouse 计算资源	自适应 Warehouse 计算资源包含多个独立的虚拟集群。它具备无基础设施管理的优势（仅需配置最大/单 SQL CU 数），支持根据查询复杂度按需动态分配资源。为提高资源利用率，系统会在集群空闲后自动回收资源。此外，该资源支持虚拟集群异构：集群内部节点规格相同，但集群间规格可不同。

说明：
自适应 Warehouse 计算资源目前处于实验性功能阶段（Beta）。如需申请开通和使用此功能，请联系我们。

创建标准 Warehouse 计算资源

如果您选择创建标准 Warehouse 计算资源，您需要配置如下信息：

字段名称	字段释义
Warehouse 名称	用户可自定义 Warehouse 名称，默认为“我的Warehouse”
节点规格	Warehouse 中节点的规格，可选择的规格和说明如下： • 2X-Small：每节点包含 4 CU 计算资源 • X-Small：每节点包含 8 CU 计算资源 • Small：每节点包含 16 CU 计算资源 • Medium：每节点包含 32 CU 计算资源 • Large：每节点包含 64 CU 计算资源
节点个数	用户可结合数据规模、查询复杂度和查询并发，设定合适的 Warehouse 节点个数。

创建自适应 Warehouse 计算资源

如果您选择创建自适应 Warehouse 计算资源，您需要配置如下信息：

字段名称	字段释义
Warehouse 名称	同标准 Warehouse 计算资源
Warehouse 最大可用 CU 数	多虚拟集群架构中，资源池能够同时提供给整个 Warehouse 中所有正在运行的虚拟集群的计算资源（CU）总量的上限。
单条 SQL 最大可用 CU 数	单虚拟集群最大可用 CU 数量，其值不允许超过 Warehouse 最大可用 CU 数。

Step 6：创建实例

1. 确认无误后，阅读并确认同意《腾讯云服务协议》，单击**新建实例**，等待实例创建完成。等待约3~5分钟后，实例创建完成。单击**实例ID**，进入实例。

> **注意：**
> TCHouse-X 的运行依赖于对元数据、权限及存储的访问。为确保服务正常启动，系统需创建以下服务关联角色：
> - TCHOUSEX_QCSLinkedRoleInTCLake(关联策略：QcloudAccessForTCHOUSEXLinkedRoleInTCLake)
> - Tccatalog_QCSLinkedRoleInMetadataManagement(关联策略：QcloudAccessForTccatalogLinkedRoleInMetadataManagement)
>
> 若您的账号尚未创建上述角色，系统将在新建流程中通过弹窗引导授权。请务必确认，否则实例将无法完成创建。

2. 进入实例后，可查看实例的基本信息。

> **说明：**
> 首次创建 TCHouse-X 实例时同步开通 TCLake 服务，预计实例创建完成时间为 20 至 30 分钟。

销毁 TCHouse-X 实例

进入 TCHouse-X 控制台，进入实例列表，找到并单击相应 TCHouse-X 实例后操作列中的**销毁**按钮，即可进入实例销毁阶段。
对于 TCHouse-X 中的存量数据，用户目前可以从以下两种方式进行选择：

选项	说明
删除	保留 TCLake 服务中当前 TCHouse-X 默认数据目录（Catalog）及数据，但会产生存储费用。
销毁	删除 TCLake 服务中当前 TCHouse-X 默认数据目录（Catalog）及数据。

> **警告：**
> 在执行实例销毁操作前，请务必确认业务现状：
> 1. 数据安全性： 请确保已完成必要的数据备份或迁移。
> 2. 链路完整性： 请确认已下线所有指向该实例的 API 接口或报表连接。

计算资源管理
计算资源介绍

最近更新时间：2026-05-06 16:28:12

计算资源 是 TCHouse-X 执行查询、数据加载（DML）等计算任务的核心单元。其核心优势包括：

- **存算分离，极速弹性**：计算与存储完全解耦。您可以根据业务波动，随时启动、停止、销毁或调整配置，实现资源的按需调度。
- **精细计费，成本受控**：采用按秒计费模式（起步 60 秒），仅在运行时产生费用。任务结束即可关停，最大限度降低闲置成本。
- **物理隔离，互不干扰**：各计算集群间天然隔离。通过为 ETL 任务、BI 报表或临时查询分配专属资源，确保不同业务负载间性能零干扰。

TCHouse-X 支持三种计算资源，具体如下：

计算资源类型	说明	操作文档
标准 Warehouse 计算资源	• **定位**：稳定承载在线计算任务。 • **特性**：由同规格节点组成。支持手动平滑变更配置、定时启停及自动扩缩容。 • **价值**：提供高度可控的资源管理，适合负载相对规律的业务场景。	标准 Warehouse 计算资源管理

自适应 Warehouse 计算资源	• **定位**：全自动化的在线计算资源池。
	• **特性**：
	  ○ 免运维管理：仅需设定最大资源阈值，无需关注底层基础设施。
	  ○ 瞬时弹性：基于查询复杂度，动态、秒级调配虚拟集群执行任务。
	  ○ 资源高效利用：任务结束自动回收，实现资源利用率最大化。
		自适应 Warehouse 计算资源管理

Serverless 计算资源	• **定位**：专为离线批处理任务设计。 • **特性**：任务触发即启动，任务结束即销毁。 • **价值**：计算资源随用随走，确保离线任务与在线业务物理隔离，互不干扰。	-

标准 Warehouse 计算资源管理

最近更新时间：2026-05-06 16:28:12

TCHouse-X 是一款离在线一体化引擎。在 OLAP 在线场景中，用户默认通过标准 Warehouse 获取计算资源。一个 TCHouse-X 实例内可创建多个标准 Warehouse，它们之间计算隔离，但共享该实例的持久化存储。每个标准 Warehouse 是由多个相同规格的计算节点组成的分布式集群。用户可根据业务需求，灵活选择节点规格和节点数量。在总计算资源相同的情况下：高并发场景下，更高的节点规格通常带来更好的性能；其他场景下，更多的节点数量则更有利于性能提升。
本文档将详细介绍标准 Warehouse 的管理操作。

创建标准 Warehouse

标准 Warehouse 有两种创建方式：

1. **随实例创建**：新建实例时同步创建， 操作步骤详情请参见 实例创建与销毁 。
2. **单独创建**：在计算资源页面单独创建，具体步骤如下：
   2.1 **进入配置页面**： 进入**计算资源**页面，单击**新建Warehouse**，打开配置窗口。
   2.2 **配置参数并创建**：在弹窗中配置以下信息，确认无误后单击**确认**。

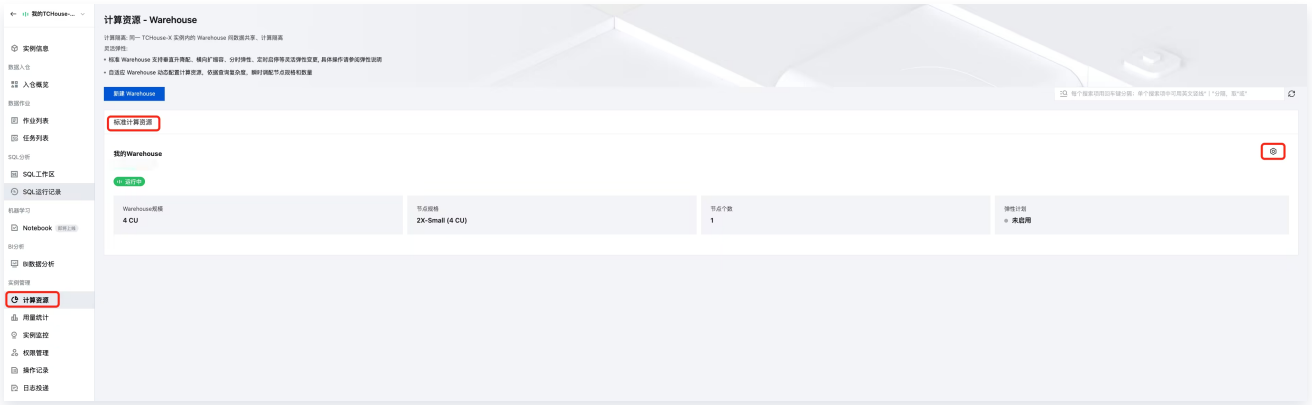
> **说明：**
> 创建过程约需 2 分钟，完成后即可在列表中查看新建的 Warehouse。

字段名称	说明
Warehouse 名称	自定义名称，默认为“我的 Warehouse”。
Warehouse 类型	选择“标准 Warehouse”。
子网	选择当前可用的子网。
节点规格	选择单节点的计算资源规格： • 2X-Small：4 CU / 节点 • X-Small：8 CU / 节点 • Small：16 CU / 节点 • Medium：32 CU / 节点 • Large：64 CU / 节点
节点个数	设置 Warehouse 包含的节点数量。
弹性计划	设置资源伸缩策略。可选：不启用、分时弹性、定时启停、自动弹性， 具体配置逻辑请参阅本文档对应章节。

手动弹性

垂直升降配(变更节点规格)

1. 进入计算资源页面，对于拟变更节点规格的标准 Warehouse，单击**设置**。



2. 在弹窗中可调整该 Warehouse 中各节点的规格，进行垂直升降配：
   - 如不开启优雅变配，变配期间该 Warehouse 正在运行的 SQL 及新提交的 SQL 均会执行失败。
   - 如需在变配期间保持 Warehouse 正常使用，可开启「优雅变配」开关，开启后将滚动变配每个节点。

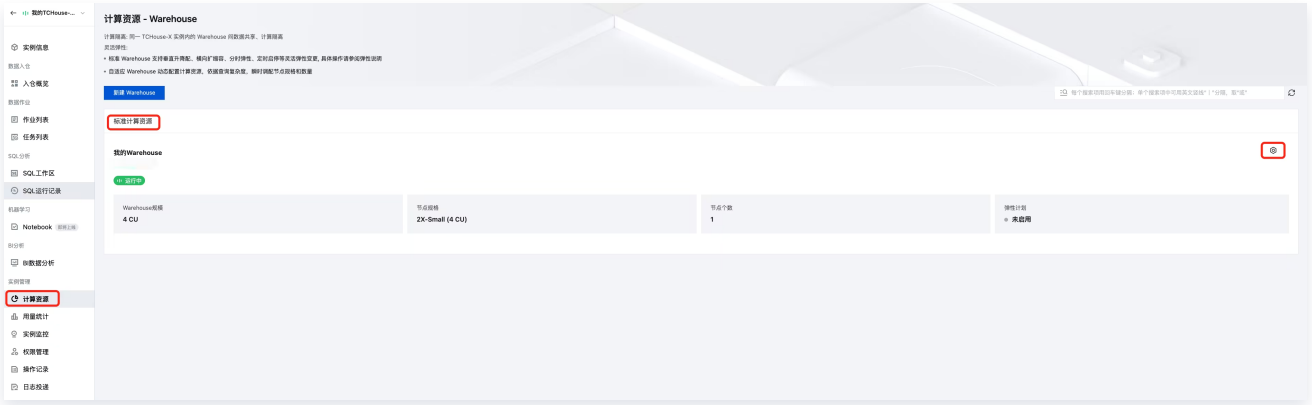
3. 单击**确定**后，系统将开始对该 Warehouse 执行垂直升降配（即变更节点规格）。操作完成后，等待一段时间（具体时长取决于变更内容），Warehouse 的状态将恢复为“运行中”，这表示本次弹性变更已成功。

> ⚠ **注意：**

> 已配置弹性计划的 Warehouse 不支持手动进行垂直或水平弹性操作。如需调整，请先取消现有的弹性计划，然后单击**设置**进行配置。

横向扩缩容(增减节点个数)

1. 进入计算资源页面，对于拟变更节点规格的标准 Warehouse，单击**设置**。



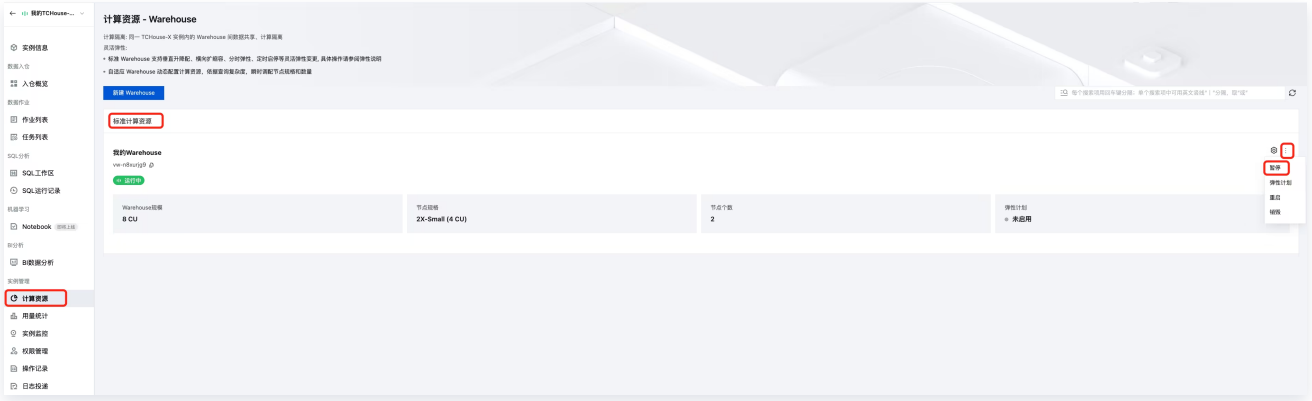
2. 在弹窗中，您可以调整该 Warehouse 的节点数量，实现横向扩缩容。
   - 横向扩容（增加节点数）：变配期间，正在运行的 SQL 与新提交的 SQL 均可正常执行，业务无感知。
   - 横向缩容（减少节点数）：默认情况下，变配期间该 Warehouse 正在运行及新提交的 SQL 均会执行失败。若需在缩容期间保持业务正常，可开启「优雅变配」开关。开启后，系统将采用滚动方式依次变更每个节点，最大程度减少对业务的影响。

3. 单击**确定**后，系统将开始对该 Warehouse 执行扩缩容操作。等待一段时间（具体时长取决于变更的规模）后，Warehouse 的状态将恢复为“运行中”，这表示弹性变更已成功完成。

> **注意：**
> 已配置弹性计划的 Warehouse 不支持手动进行垂直或水平弹性操作。如需调整，请先取消现有的弹性计划，然后单击**设置**进行配置。

手动暂停

1. 进入计算资源页面，对于拟变更节点规格的标准 Warehouse，单击**更多**，选择**暂停**按钮。



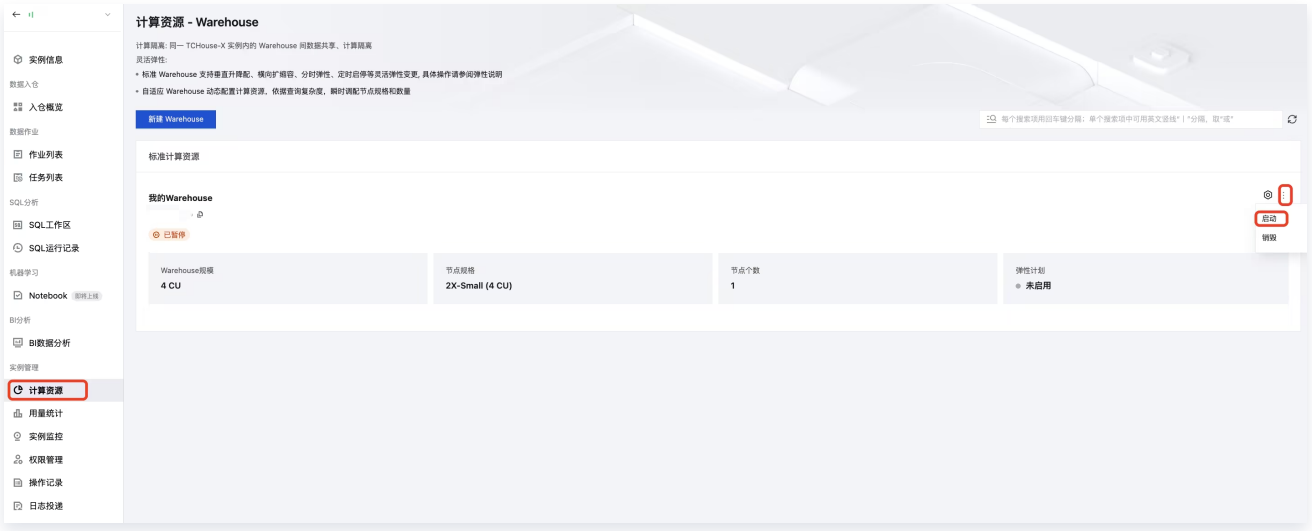
2. 单击**暂停**后，系统会弹出窗口进行二次确认。
   - 若不开启“优雅暂停”：正在运行的 SQL 和新提交的 SQL 均会执行失败。
   - 若开启“优雅暂停”：系统将拒绝新提交的 SQL，并为正在运行的 SQL 预留最长 5 分钟的宽限期。

3. 单击**确定**后，系统将开始暂停该 Warehouse。等待一段时间（具体时长取决于系统负载和数据处理状态），Warehouse 的状态将最终变为“已暂停”。

> **注意：**
> 手动暂停会自动删除弹性计划。

手动启动

1. 定位目标 Warehouse：导航至控制台的 “计算资源” 页面，找到您计划启动的 Warehouse 实例，在该实例对应的操作列中，单击**更多**，选择 “启动”



2. 二次确认：在弹出的启动确认窗口中，系统会要求您进行二次确认

3. 执行启动：单击**确定**后，系统将对该 Warehouse 执行启动操作。

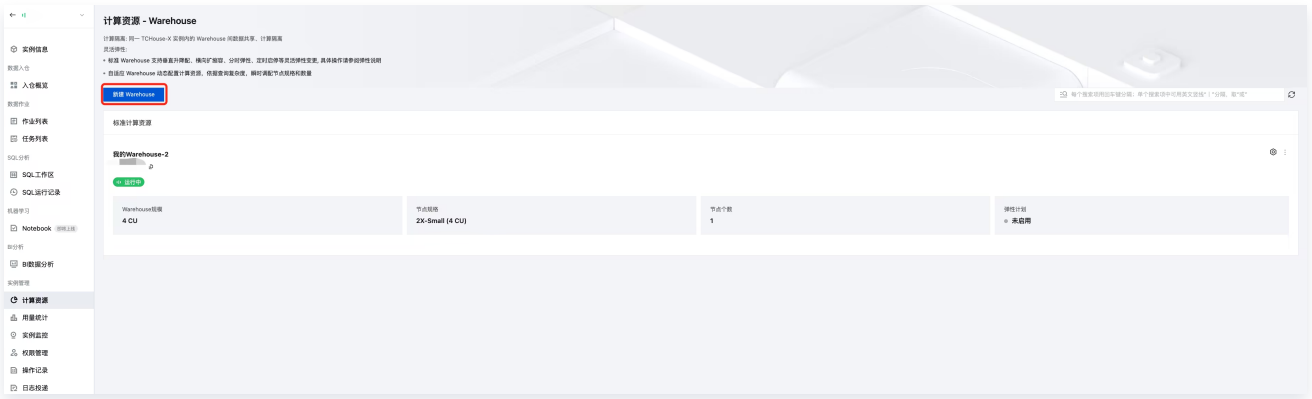
半自动弹性

分时弹性

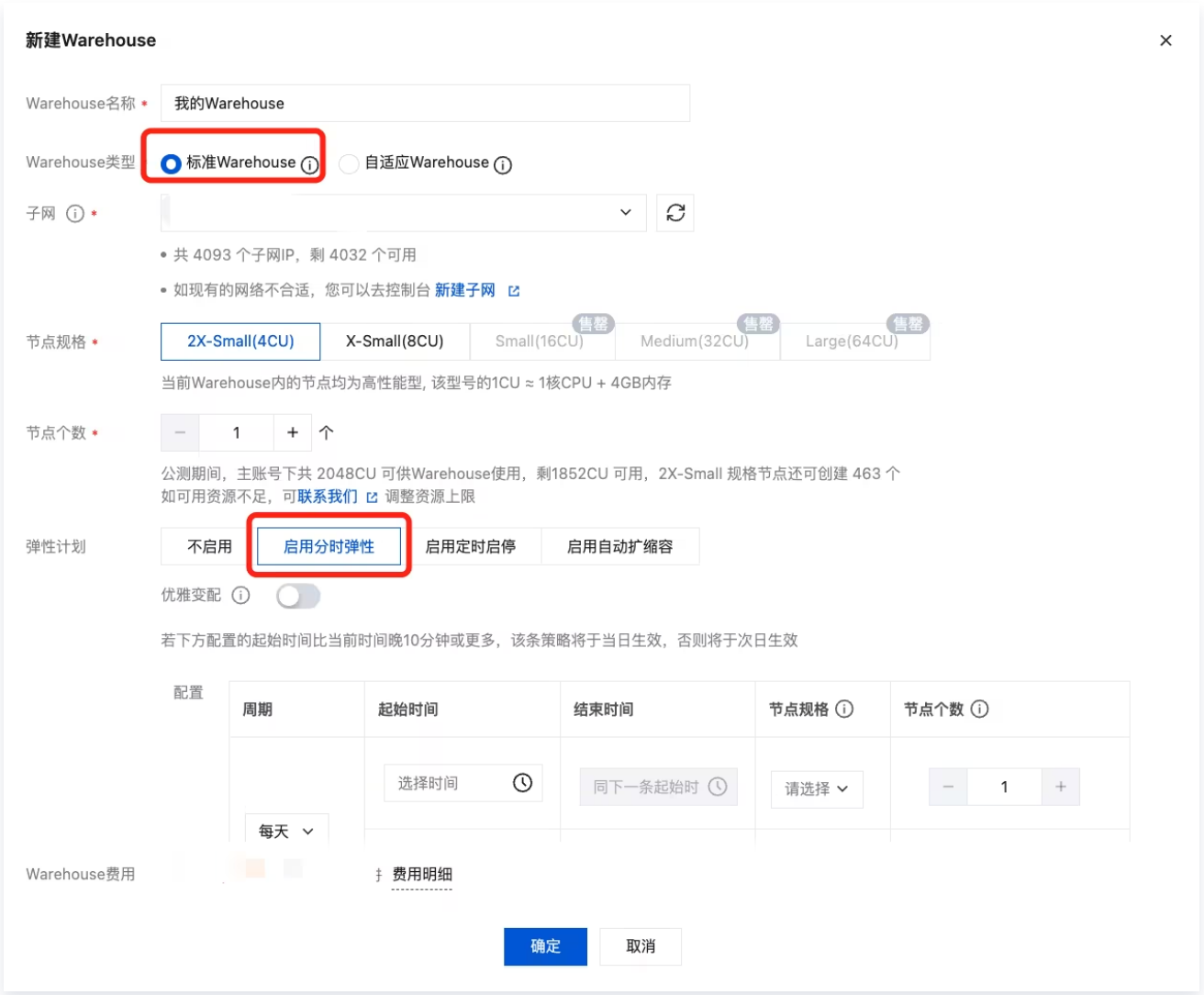
配置分时弹性规则

新建 Warehouse 配置

1. 进入配置： 在“计算资源”页面，单击**新建 Warehouse** 唤起配置弹窗。



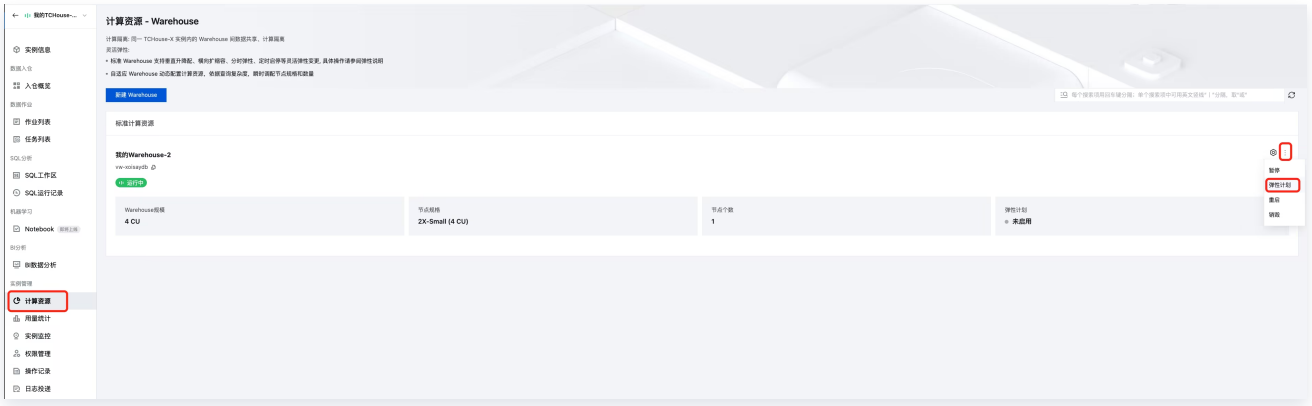
2. 启用弹性计划： 将“Warehouse 类型”设置为 标准 Warehouse，并选择弹性计划 启用分时弹性。



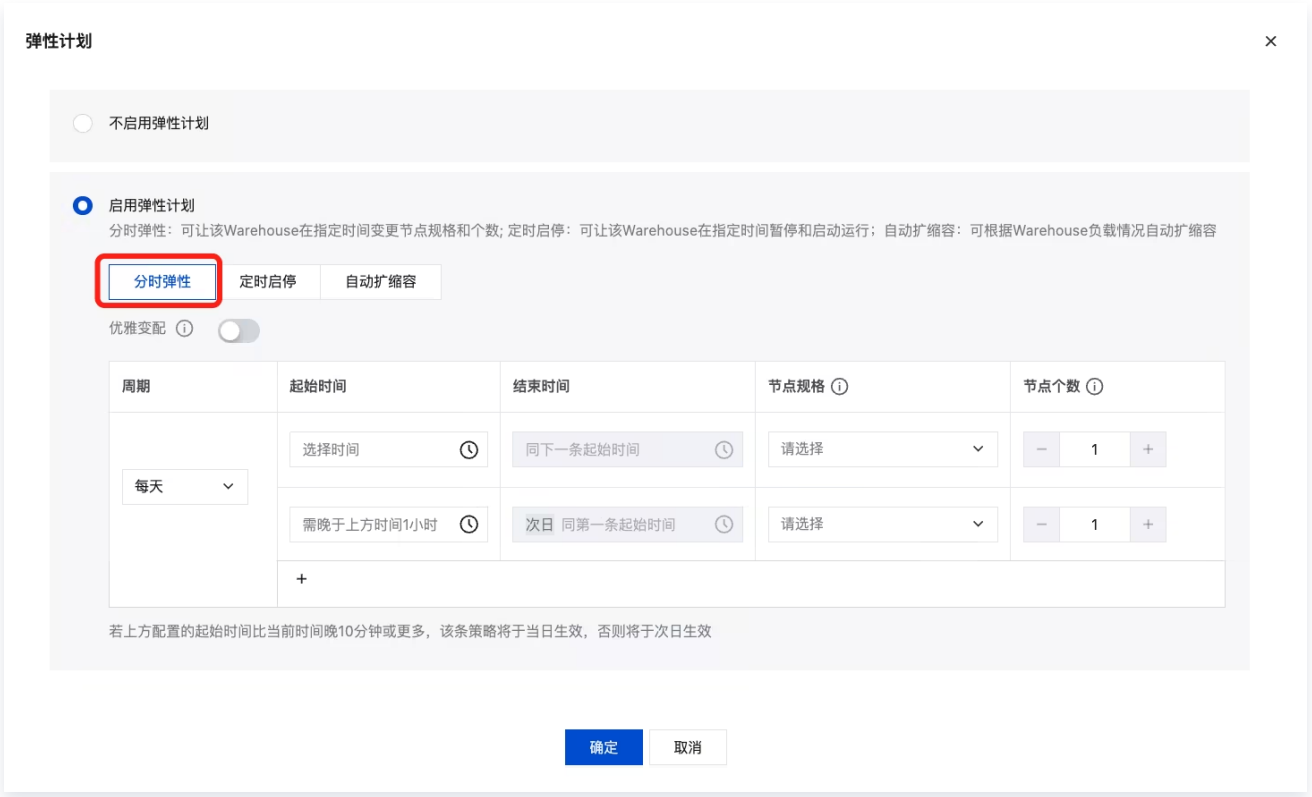
3. 生效： 填写定时参数后单击**确定**即可完成创建。

存量 Warehouse 配置

1. 定位弹性配置：在“计算资源”页面找到目标实例，单击**更多 > 弹性计划**。



2. 开启定时策略：在弹窗中勾选 启用弹性计划，配置分时弹性计划。

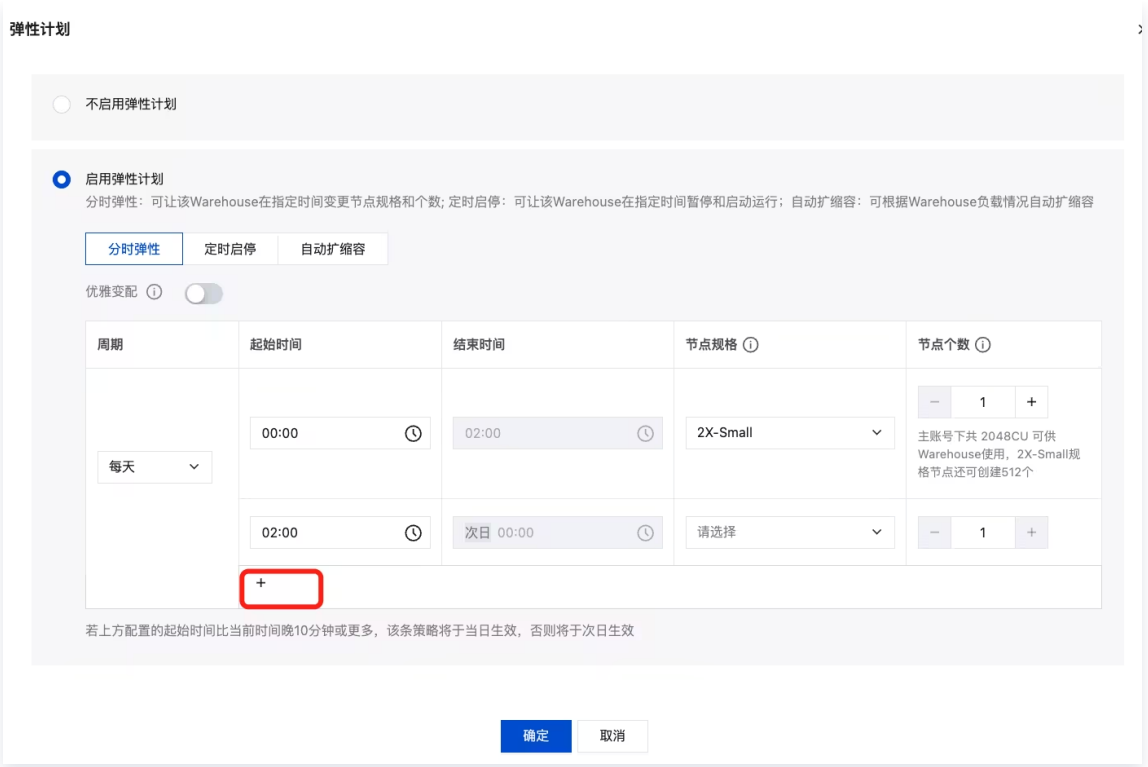


3. 保存并生效：完成时间参数配置后单击**确定**，该 Warehouse 即可按计划自动启停。

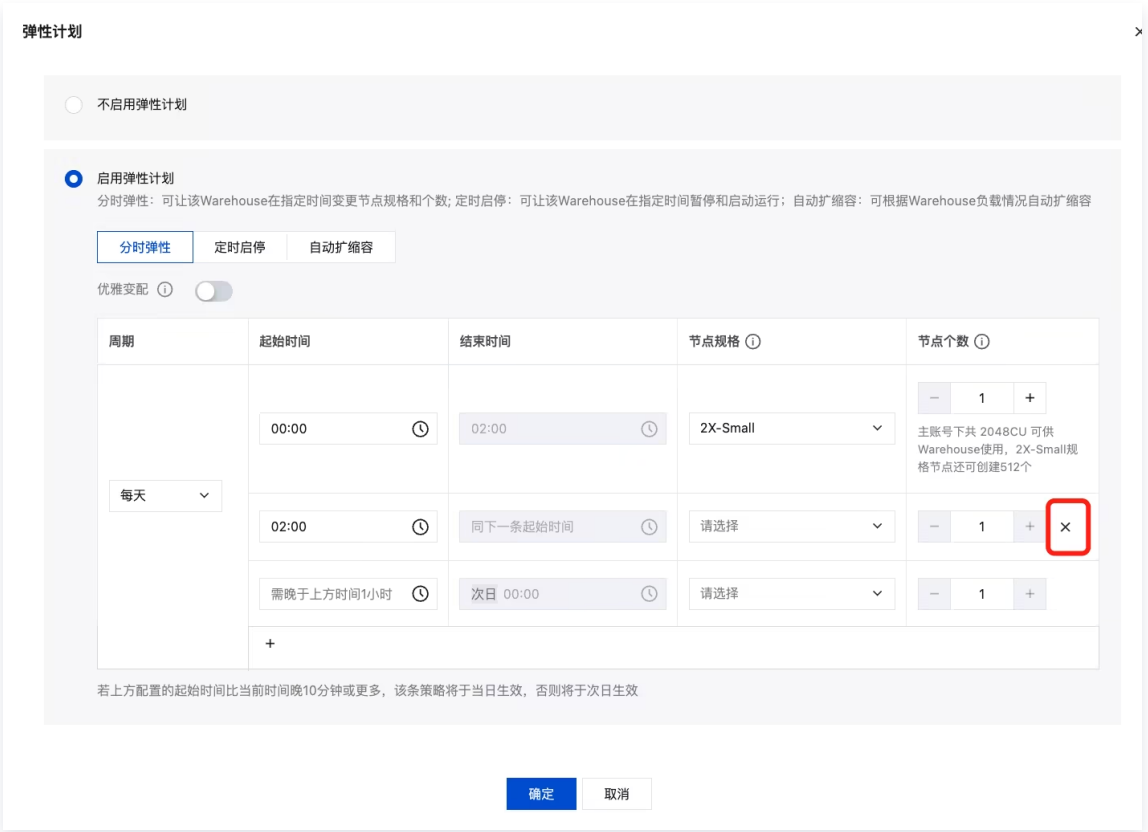
配置参数说明

参数	配置	约束与限制

维度	说明	
周期类型	支持每天、每周、每月	每周/每月支持多选；若月份无对应日期（如29-31号）则自动跳过。
起始时间	00:00 ~ 23:45	步长为 15 分钟；相邻策略起始时间须间隔 ⩾ 1 小时。
结束时间	系统自动计算	无需手动配置。采取首尾衔接机制，形成 24 小时闭环。
策略数量	最小2条，最多24条	单击 + 按钮增加策略。



单击 ✖ 按钮删除策略。



计算资源	设置计算	相邻两条（含首尾）策略的规格或节点数不得相同。

规格与节点个数	

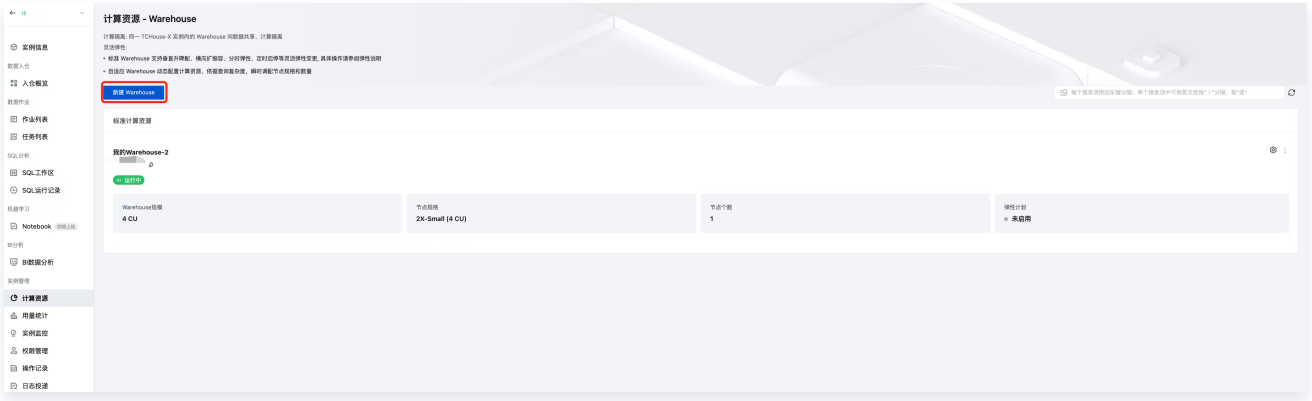
定时启停

定时启停计划允许您根据业务低谷期自动暂停（停止）Warehouse，并在高峰期自动恢复运行（启动）。

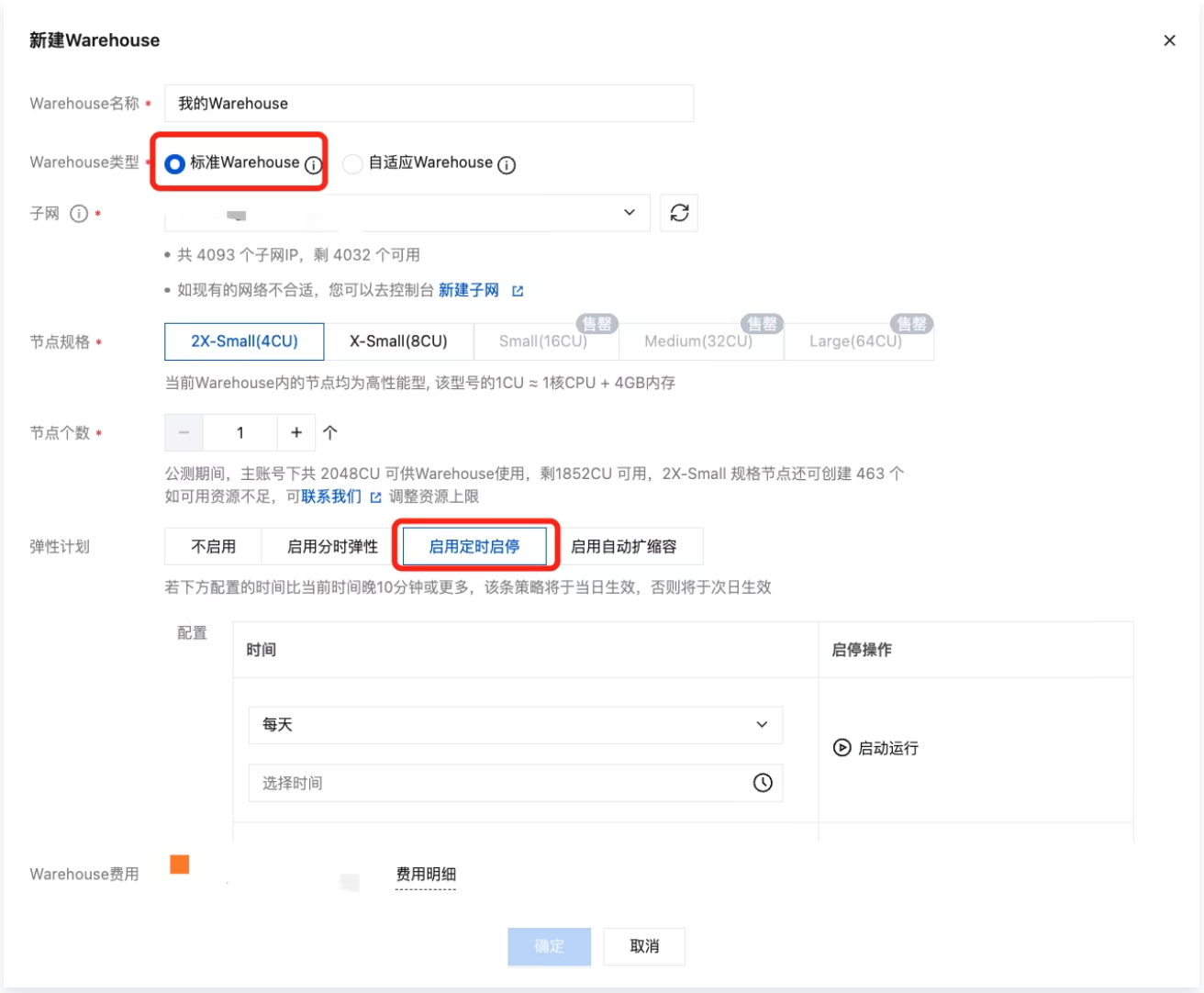
配置定时启停规则

新建 Warehouse 配置

1. 进入配置： 在“计算资源”页面，单击**新建 Warehouse** 唤起配置弹窗。



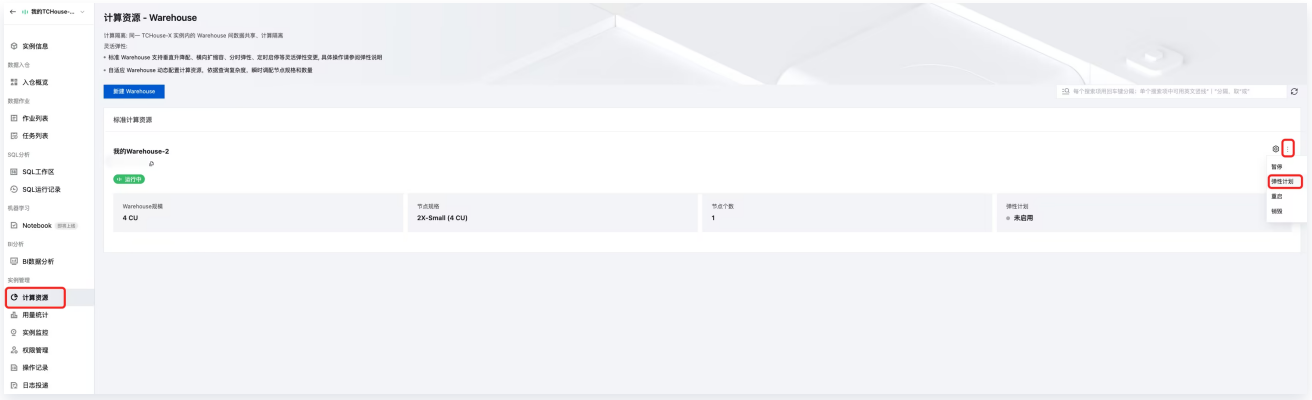
2. 启用弹性计划： 将“Warehouse 类型”设置为 标准 Warehouse，并选择弹性计划启用定时启停。



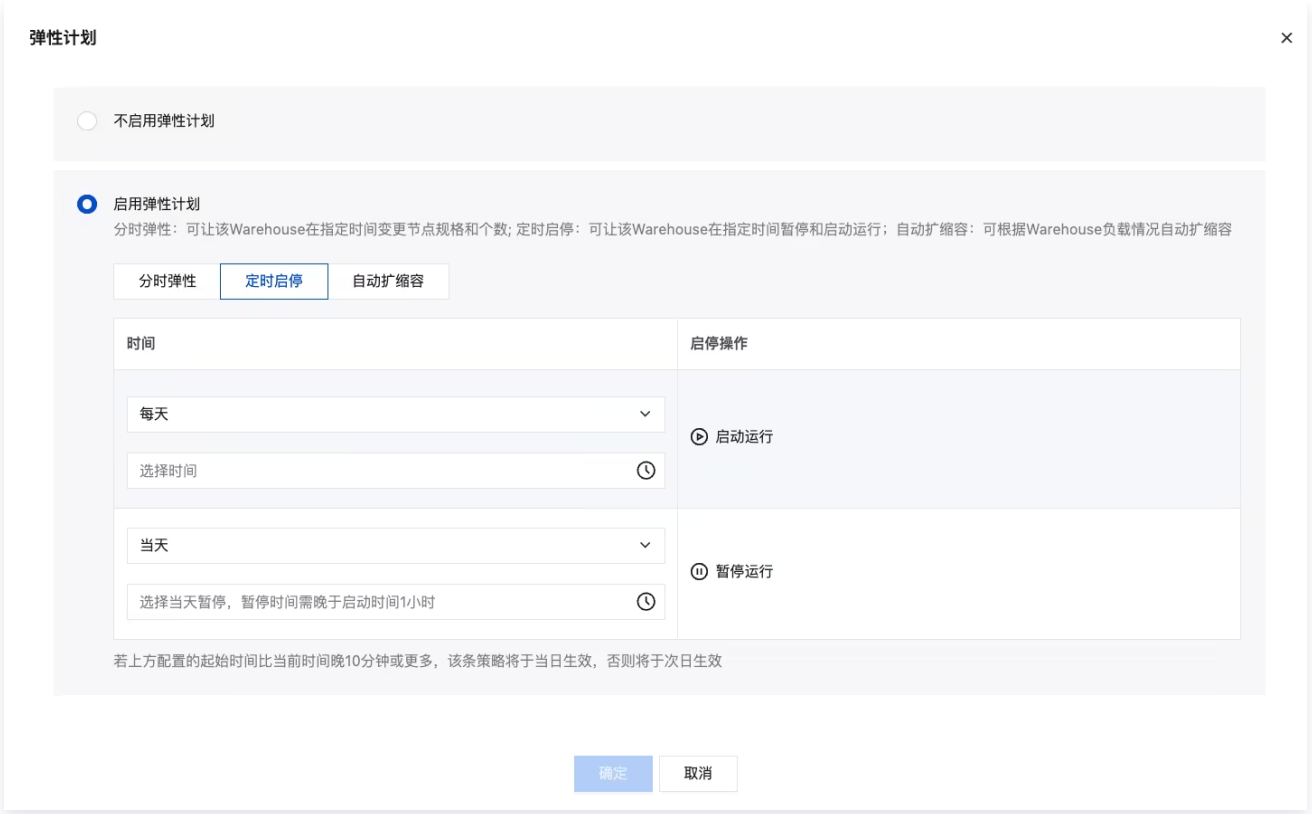
3. 生效： 填写分时参数后单击**确定**即可完成创建。

存量 Warehouse 配置

1. 定位弹性配置：在“计算资源”页面找到目标实例，单击**更多 > 弹性计划**。



2. 开启定时策略：在弹窗中勾选 启用弹性计划，配置定时启停。



3. 保存并生效：完成时间参数配置后单击 **确定**，该 Warehouse 即可按计划自动启停。

配置参数说明

配置项	描述	配置要求/范围
周期	定义计划的重复频率。	可选 每天、每周、每月。 每周： 支持多选周一至周日。 每月： 支持多选 1 至 31 号（若当月无对应日期，则自动跳过）。
时间	设定启动和暂停的具体时间点。	选择范围：00:00~23:45，步长为 15 分钟。

> **注意：**
> 如果您配置的起始时间晚于提交保存时的时间 10 分钟或更多，该计划将于当日生效；否则将于次日生效。

自动弹性

TCHouse-X 标准 Warehouse 支持基于资源利用率指标的自动横向扩容功能，旨在帮助用户应对业务高峰，并确保资源的最优使用。系统将结合用户配置的扩缩容条件，在满足触发条件时自动执行扩容操作。

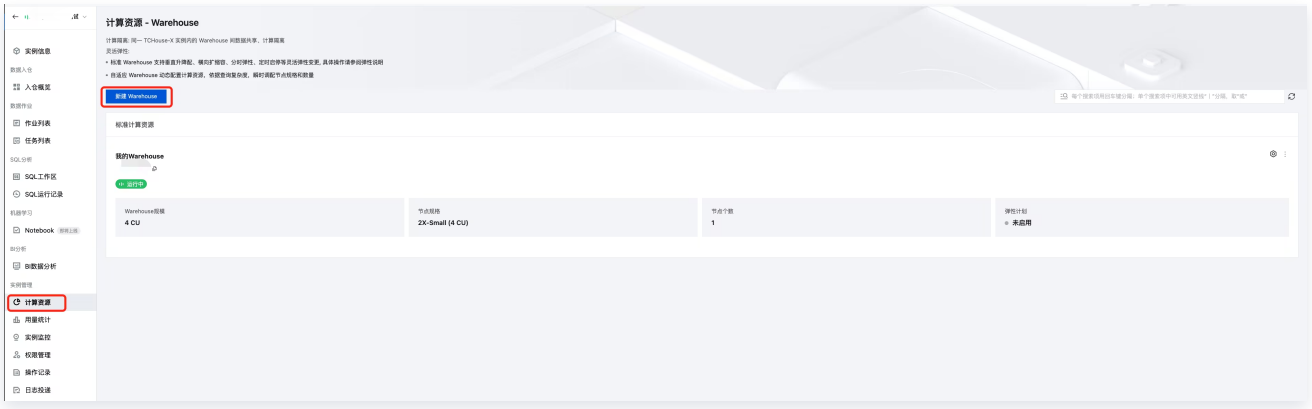
关键配置项说明

配置项	描述	配置范围/限制
观测时间	观测时间周期内Warehouse平均利用率大于等于设定值时，将自动执行扩容。	最小值 5 分钟；输入值范围 5~999
扩容检测指标	触发自动扩容所观测的标准 Warehouse 内节点平均 CPU 利用率。	40%~100%
最大扩容规格	自动扩容后，集群所能达到的最大节点规格。	下限： 当前标准 Warehouse 内节点数 * 单节点规格 上限： 当前标准 Warehouse 节点规格 * floor(主账号可用CU数/当前标准 Warehouse 节点规格)
静默时间	两次自动扩容操作之间的最小间隔时间。在此期间，系统不会触发新的扩容操作。	输入值范围：10~999

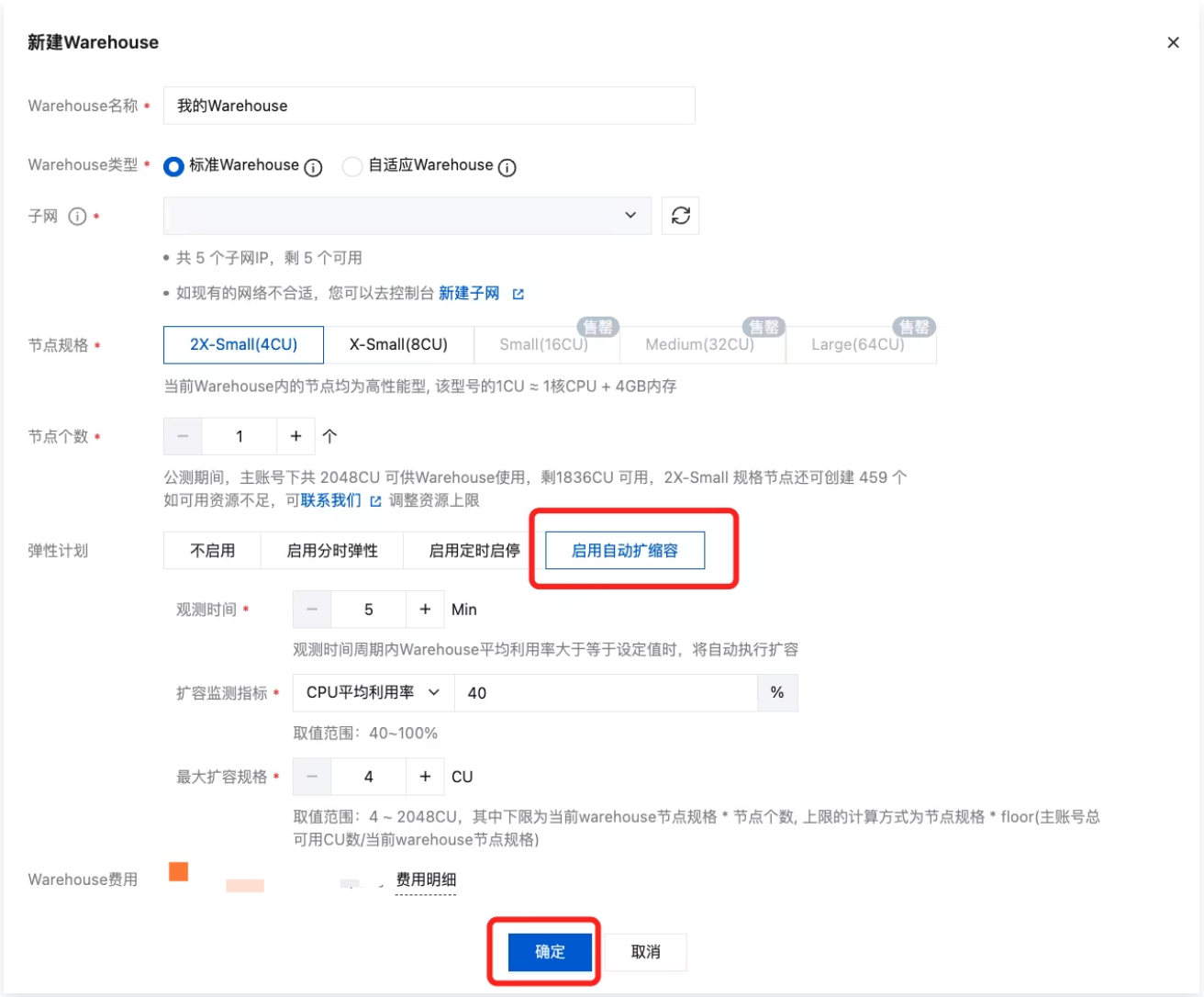
自动弹性开启步骤

新建标准 Warehouse 配置自动扩缩容

1. 进入计算资源页面并新建： 导航至“计算资源”页面，单击**新建 Warehouse**。

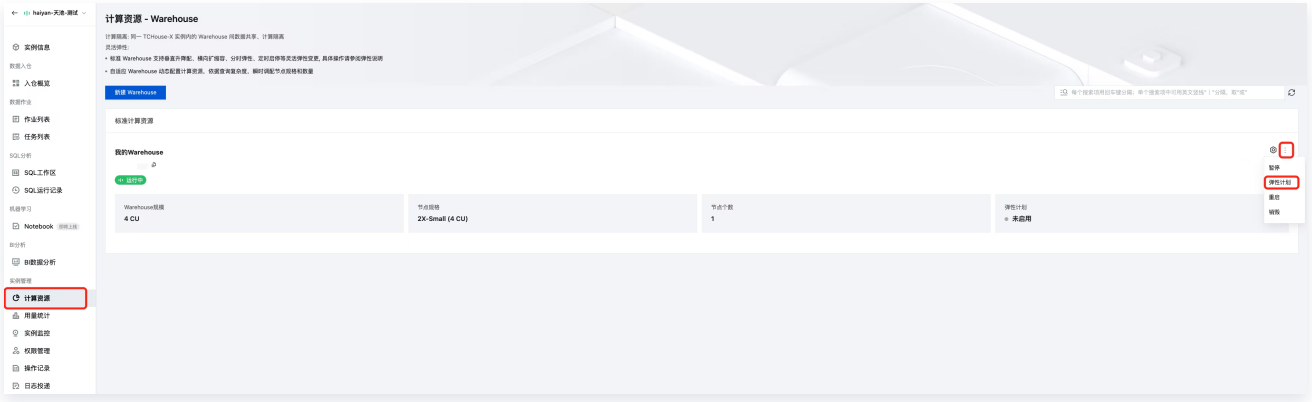


2. 配置参数并启用自动扩缩容： 在新建窗口中，勾选 “启用自动扩缩容”，然后根据业务需求配置 扩缩容策略 的相关参数（如 CPU 利用率阈值、观测时间、最大/最小规格等），最后单击**确认**完成创建。



单独开启弹性计划配置自动扩缩容

1. 进入计算资源页面并新建： 导航至“计算资源”页面，找到拟配置弹性计划的标准 Warehouse，单击更多，选择“弹性计划”

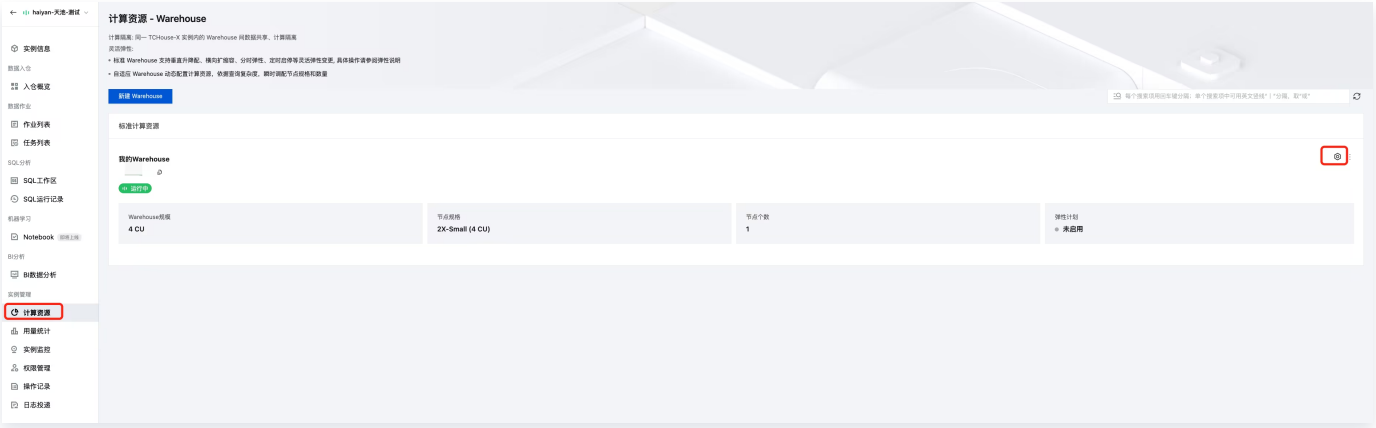


2. 启用弹性计划，选择自动扩容，然后根据业务需求配置扩缩容策略的相关参数（如 CPU 利用率阈值、观测时间、最大规格等），最后单击**确定**完成开启。

标准 Warehouse 其他变更

变更标准 Warehouse 名称

1. 进入 “计算资源” 页面，找到目标标准 Warehouse，单击其对应的**设置**按钮。

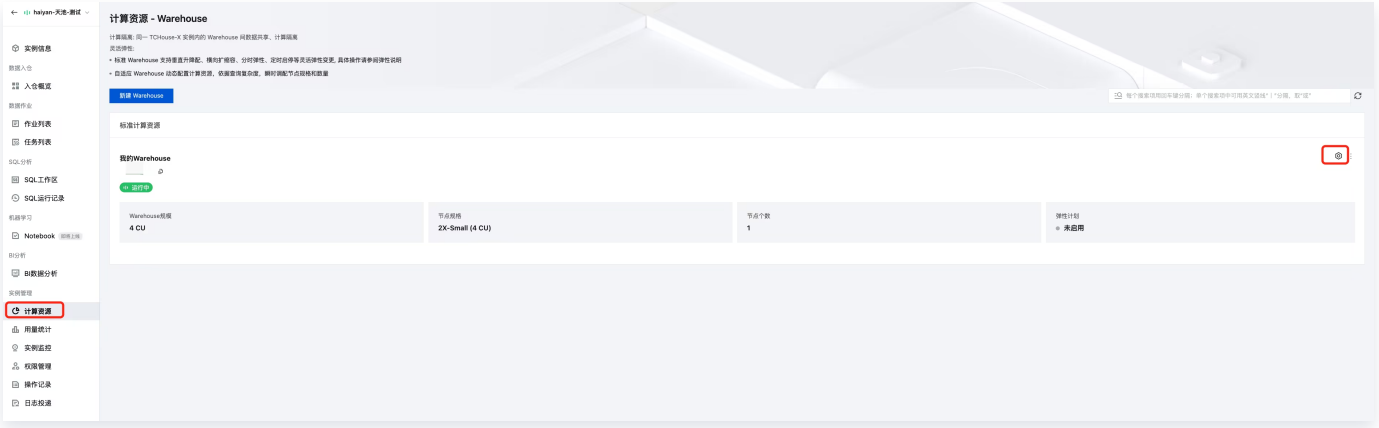


2. 单击后可在弹窗中修改 Warehouse 名称，单击**确定**即可。

> **注意：**
> Warehouse 的唯一标识 是系统自动生成的 Warehouse ID。虽然 Warehouse 名称不强制唯一，但为方便管理，推荐使用不同的名称来区分不同的实例。

变更标准 Warehouse 所属子网

1. 进入 “计算资源” 页面，找到目标标准 Warehouse，单击其对应的**设置**按钮。



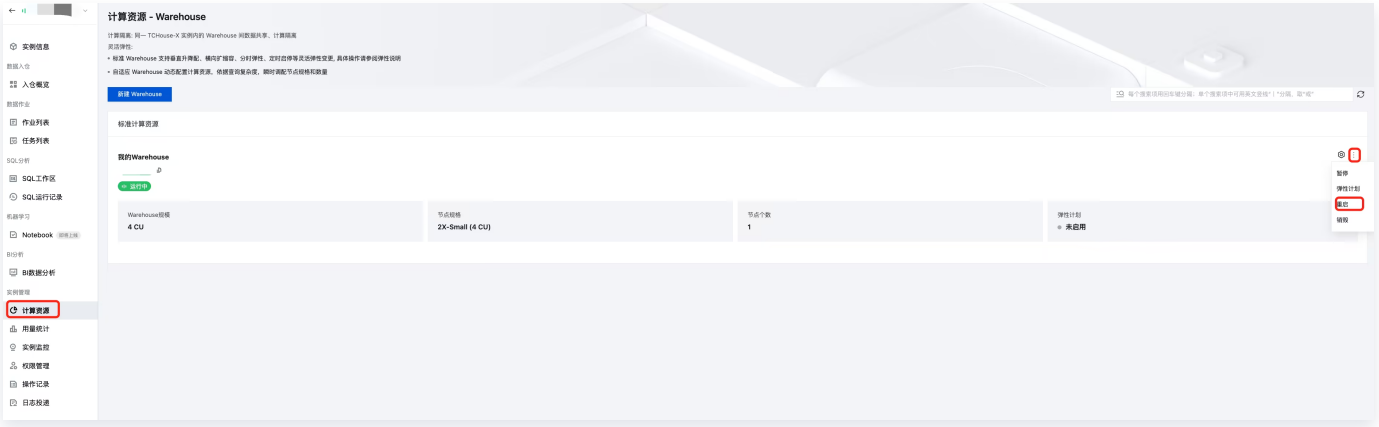
2. 可在弹窗中修改标准 Warehouse 所属子网，单击**确定**即可。

> **注意：**
> 切换子网将导致标准 Warehouse 重启。您只能选择实例所属 VPC 内的子网。

重启标准 Warehouse

运行中的标准 Warehouse 支持手动重启功能。具体操作步骤和重启机制如下：

1. 定位目标 Warehouse：导航至控制台的 “计算资源” 页面，找到您计划重启的 Warehouse 实例，在该实例对应的操作列中，单击**更多**，选择 “重启”。



2. 二次确认与选择重启模式：在弹出的重启确认窗口中，系统会要求您进行二次确认。

> **注意：**
> 重要： 在此步骤，请根据需求选择是否启用 “优雅重启” 选项。重启机制说明：

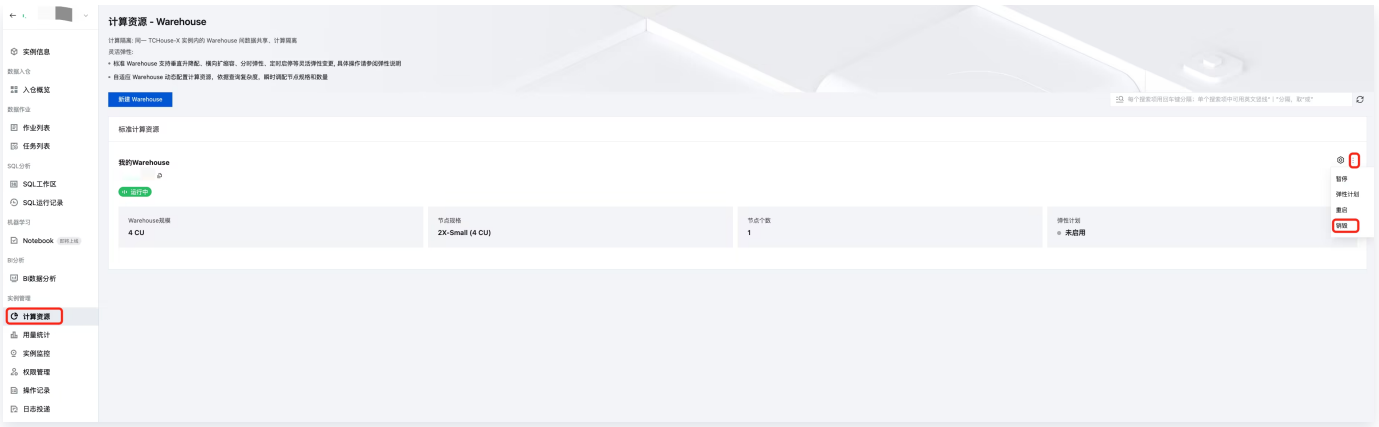
重启模式	对 SQL 任务的影响	适用场景
未启用优雅重启	正在运行的 SQL 和重启期间新提交的 SQL 都会失败。	追求快速重启，或无活跃业务时。
启用优雅重启	正在运行的 SQL 预留最长 5 分钟的宽限期；新提交的 SQL 也可以正常执行。	存在活跃业务和运行中查询，需确保业务连续性。

3. 执行重启：单击**确定**后，系统将对该 Warehouse 执行重启操作。等待一段时间，当 Warehouse 状态重新转为 “运行中” 时，表示重启成功。

> **注意：**
> 若 Warehouse 重启时间与定时启停或分时弹性计划的时间冲突，系统将自动延迟执行相应的启停或弹性任务。

销毁标准 Warehouse

1. 定位目标 Warehouse：导航至控制台的 “计算资源” 页面，找到您计划销毁的 Warehouse 实例，在该实例对应的操作列中，单击**更多**，选择 “销毁” 。



2. 二次确认：在弹出的销毁确认窗口中，输入“销毁”进行二次确认。

3. 执行销毁：单击**确定**后，系统将对该 Warehouse 执行销毁操作。

自适应 Warehouse 计算资源管理

最近更新时间：2026-05-06 16:28:12

简介

在标准 Warehouse 模式下，计算资源被限制在单一集群（Cluster）中。当高并发查询请求涌入时，有限的资源会导致查询进入等待队列，产生明显的延迟。
自适应 Warehouse（Adaptive Warehouse）是 TCHouse-X 提供的自动化资源管理方案。它能够根据实时负载，动态地在预设范围内调整计算资源，平衡“性能保障”与“成本控制”。

> **说明：**
> 自适应 Warehouse 计算资源目前处于实验性功能阶段（Beta）。如需申请开通和使用此功能，请 联系我们。

工作原理

自适应机制通过以下逻辑实现计算资源的自动化管理：

- **智能预估**：系统在 SQL 执行前对其资源消耗量进行预估。
- **动态调度**：
  - **复用**：若当前已运行的集群有足够余量，则直接复用。
  - **横向扩展（Scale-out）**：若现有集群负载过高，系统将自动启动新的集群（Cluster）来承载新任务。
- **自动回收**：当集群进入空闲状态并达到设定时间后，系统会自动触发回收机制，停止计费。

配置参数说明

为了精准控制资源使用，您需要配置以下两个核心上限：

配置项	说明	业务价值
Warehouse 最大可用 CU 数	Warehouse 资源上限，即该 Warehouse 允许使用的总 CU 数。	控制总预算，防止并发过高导致云账单超出预期。
单条 SQL 最大可用 CU 数	单 SQL 资源上限，即单条查询允许申请的最大 CU 数。	防止大查询拖垮系统，避免个别复杂 SQL 占用过多共享资源。

自适应 Warehouse 管理指引

新建自适应 Warehouse

自适应 Warehouse 有两种创建方式：

1. **随实例创建**：新建实例时同步创建， 操作步骤详见 实例创建与销毁。

2. **单独创建：**在计算资源页面单独创建，具体步骤如下。

2.1 **进入配置页面：** 进入**计算资源**页面，单击**新建 Warehouse**，打开配置窗口。

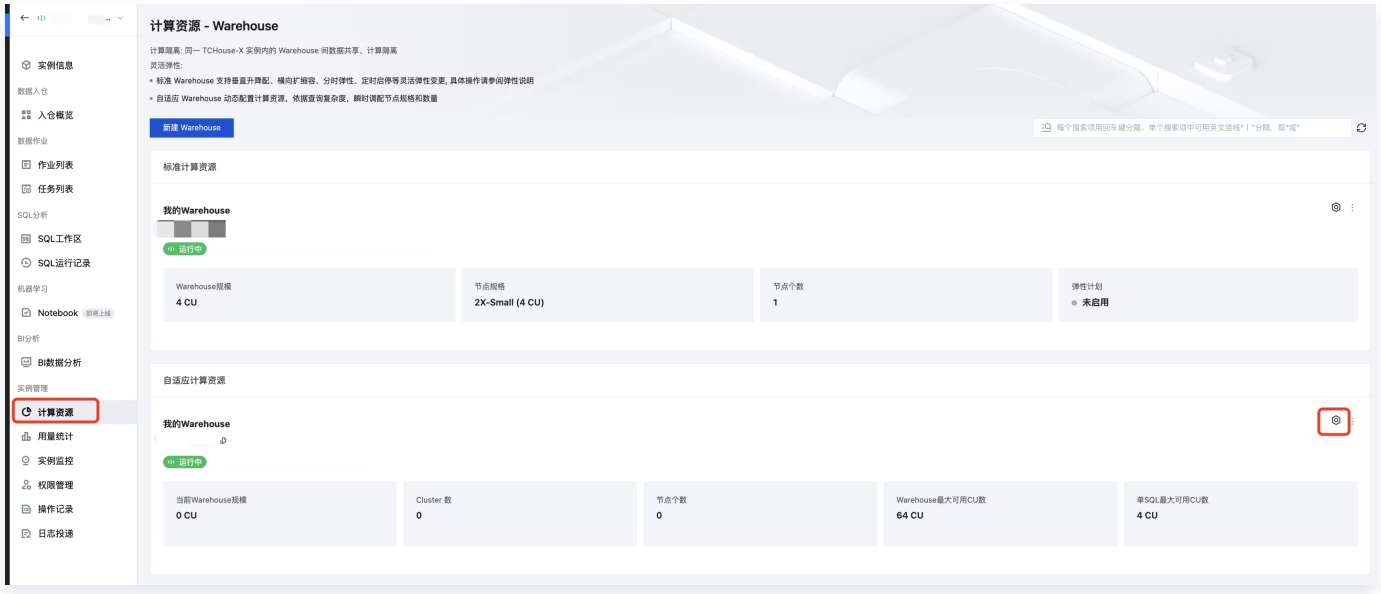
2.2 **配置参数并创建：**在弹窗中配置以下信息，确认无误后单击**确认**。

> **说明：**
> 创建过程约需 2 分钟，完成后即可在列表中查看新建的 Warehouse。

字段名称	说明
Warehouse 名称	自定义名称，默认为“我的 Warehouse”。
Warehouse 类型	选择“自适应 Warehouse”。
Warehouse 最大可用 CU 数	Warehouse 资源上限，即该 Warehouse 允许使用的总 CU 数。
单条 SQL 最大可用 CU 数	单 SQL 资源上限，即单条查询允许申请的最大 CU 数。

变更自适应 Warehouse

1. 进入 “计算资源” 页面，找到目标自适应 Warehouse，单击其对应的**设置**按钮， 打开配置窗口。



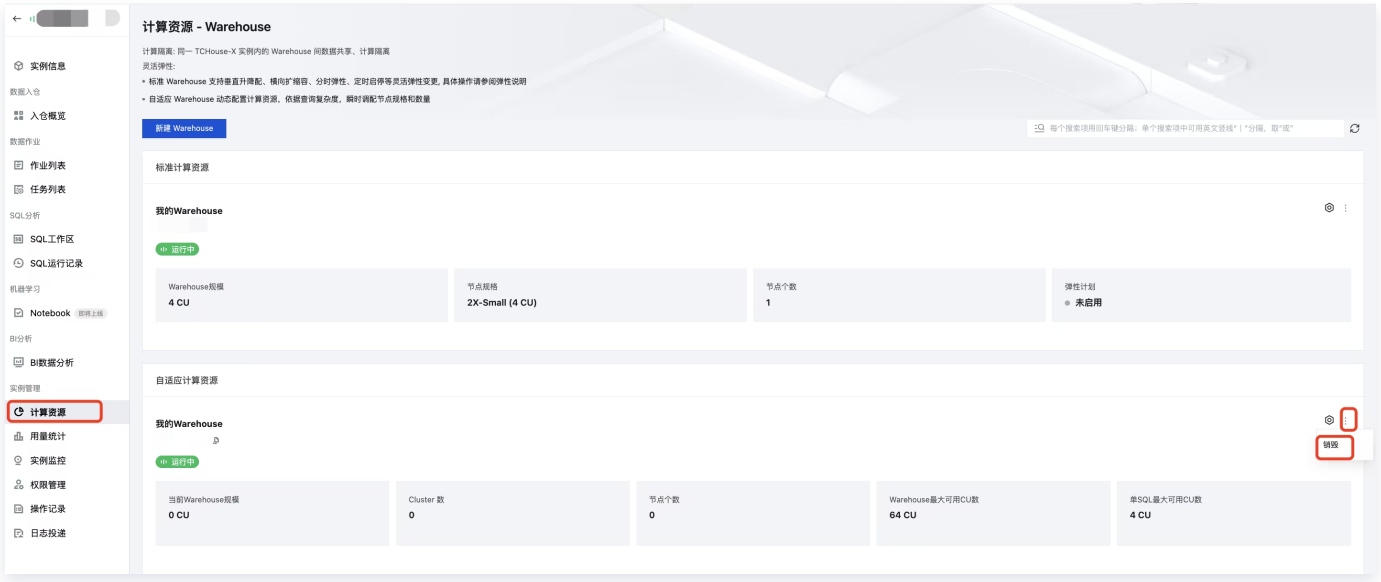
2. 按需配置如下参数，确认无误后单击**确认**。

字段名称	说明
Warehouse 名称	自定义名称，默认为“我的 Warehouse”。

Warehouse 最大可用 CU 数	Warehouse 资源上限，即该 Warehouse 允许使用的总 CU 数。
单条 SQL 最大可用 CU 数	单 SQL 资源上限，即单条查询允许申请的最大 CU 数。

销毁自适应 Warehouse

1. 定位目标 Warehouse：导航至控制台的 “计算资源” 页面，找到您计划销毁的自适应 Warehouse 实例，在该实例对应的操作列中，单击 **更多**，选择 “销毁” 。



2. 二次确认：在弹出的销毁确认窗口中，系统会要求您进行二次确认。
3. 执行销毁：单击**确定**后，系统将对该 Warehouse 执行销毁操作。

用户与权限管理
权限体系介绍

最近更新时间：2026-05-06 16:28:12

TCHouse-X 作为一款强大的数据引擎产品，内置库、表等多种数据库对象。它提供权限管理功能，允许用户对 TCHouse-X 内的用户及角色的库表访问权限进行精细化管理。目前，TCHouse-X 支持基于角色的权限管理（RBAC）。

基于角色的权限管理（RBAC）

在 RBAC 模型中，权限不是直接分配给用户，而是通过角色间接分配：

1. 定义角色： 根据工作职能（例如：管理员、编辑、普通用户）创建不同的角色。
2. 分配权限给角色： 为每个角色分配其完成工作所需的权限集（例如：“dba”角色拥有“创建、修改库表”的权限）。
3. 分配角色给用户： 将一个或多个角色分配给用户。用户继承角色的所有权限。
4. 当用户的工作职责发生变化时，管理员只需要更改用户的角色分配，访问权限就会自动调整，无需逐一修改用户的上百条权限。

用户

详情请参见 用户管理 。

角色

详情请参见 角色管理 。

Catalog-Schema-Table 三级权限管理机制

TCHouse-X 采用 Catalog-Schema-Table 三级访问控制模型来组织数据权限。这种结构不仅提供了清晰的数据资产视图，更实现了基于角色的分级权限管理。所有权限均自上而下继承：Catalog 级别的授权将直接作用于其包含的所有 Schema 和 Table。

> **说明：**
> TCHouse-X 当前版本仅支持默认 Catalog，因此在权限配置界面无需手动选择。

三级权限结构详解

Catalog（数据目录）

TCHouse-X 实例中数据的集合。其权限类型有：

权限	权限分类	权限解释
`ALL_PRIVILEGES`	数据目录权限	选中的 Catalog 及以下所有库、表的 ALL 权限
`CreateSchema`	数据目录权限	在选中 Catalog 下创建 Schema
`UseSchema`	数据目录下任意库权限	使用选中 Catalog 下任意 Schema
`AlterSchema`	数据目录下任意库权限	修改选中 Catalog 下任意 Schema
`DropSchema`	数据目录下任意库权限	移除选中 Catalog 下任意 Schema
`CreateTable`	数据目录下任意库权限	在选中 Catalog 下任意 Schema 下创建表
`SelectTable`	数据目录下任意表权限	读取选中 Catalog 下任意表的数据
`AlterTable`	数据目录下任意表权限	修改选中 Catalog 下任意表的元数据
`DropTable`	数据目录下任意表权限	移除选中 Catalog 下任意表
`InsertTable`	数据目录下任意表权限	向选中 Catalog 下任意表插入数据
`DeleteTable`	数据目录下任意表权限	从选中 Catalog 下任意表删除数据

Schema（数据库）

Schema 是 TCHouse-X 实例中的逻辑分组，相当于传统数据库中的“数据库（Database）”或“模式（Schema）”。它用于将相关的表、视图、函数等对象组织在一起。其权限类型有：

权限	权限分类	权限解释
`ALL_PRIVILEGES`	数据库权限	选中的 Schema 及以下所有表的 ALL 权限
`UseSchema`	数据库权限	使用选中 Schema
`AlterSchema`	数据库权限	修改选中 Schema 的元数据
`DropSchema`	数据库权限	移除选中 Schema
`CreateTable`	数据库权限	在选中 Schema 下创建表

SelectTable	数据库下任意表权限	读取选中 Schema 下任意表的数据
AlterTable	数据库下任意表权限	修改选中 Schema 下任意表的元数据
DropTable	数据库下任意表权限	移除选中 Schema 下任意表
InsertTable	数据库下任意表权限	向选中 Schema 下任意表插入数据
DeleteTable	数据库下任意表权限	从选中 Schema 下任意表删除数据

Table (数据表/视图)

实际存储数据的最小单位（表），或基于表定义的查询结果（视图），其作用有：

- 实际存储和呈现数据。
- 权限控制： 提供最细粒度的权限控制。可以对单个表或视图授予精确的权限，其权限类型有：

权限	权限分类	权限解释
ALL_PRIVILEGES	表/视图权限	选中 Table/View 的 ALL 权限
SelectTable	表/视图权限	读取选中 Table/View 的数据
AlterTable	表/视图权限	修改选中 Table/View 元数据
DropTable	表/视图权限	移除选中 Table/View
InsertTable	表权限	向选中 Table 插入数据
DeleteTable	表权限	从选中 Table 删除数据

用户管理

最近更新时间：2026-05-06 16:28:12

用户认证体系概述

腾讯云 TCHouse-X 支持如下三种用户体系：

用户认证体系	说明
腾讯云 CAM 用户	用户可使用腾讯云子账号 ID，以及其密钥凭证（SecretId、SecretKey 或临时 Token）连接 TCHouse-X 实例。用户身份和权限的校验由腾讯云 CAM 服务负责。
自定义用户	允许用户设置自定义的用户名和密码，并将其绑定至腾讯云子账号。用户使用该自定义凭证连接 TCHouse-X 实例时，由 TCHouse-X 负责完成用户名和密码的校验。
自定义 LDAP 用户	将自有 LDAP 服务中的用户身份与腾讯云子账号绑定。用户可使用自有 LDAP 凭证连接 TCHouse-X 实例，由自有 LDAP 服务进行用户名和密码校验。

注意：

1. TCHouse-X 现已开放自定义 LDAP 用户功能的预览体验，欢迎联系我们申请试用。
2. 实例支持的用户认证体系由实例创建时配置的“用户体系”决定。

用户身份概述

TCHouse-X 中有如下三类用户身份：

用户身份	说明
超级管理员	固定为腾讯云主账号、TCHouse-X 实例创建子账号。 支持管理内部子账号：包括创建角色与用户、配置角色权限、授予或撤销管理员身份，并支持通过客户端连接访问。
管理员	仅可创建、删除普通用户、通过客户端连接 TCHouse-X 实例。
普通用户	仅可通过客户端连接 TCHouse-X 实例。

用户体系选择

用户体系配置需在创建实例时指定，实例创建后不支持修改。

用户创建

进入 TCHouse-X 控制台，通过**实例列表 > 进入实例 > 权限管理 > 创建用户**路径进入用户创建页面。

说明：
仅用户身份为 “超级管理员”、 “管理员” 的用户才可以创建用户。

腾讯云 CAM 用户

1. 用户体系选择 “腾讯云 CAM 用户”，单击选择“腾讯云用户” 选择需要添加的腾讯云账号列表。
2. 选择好需要添加的腾讯云用户，单击**确认**，进入用户身份配置页面。
3. 按需调整所选用户的用户身份，单击**创建用户**即可。

自定义用户

说明：
- 只有在新建实例时在“用户体系”中勾选“自定义用户”选项，该实例才具备创建和使用自定义用户的功能。

1. 用户体系选择 “自定义用户” 进入自定义用户配置页。
2. 配置好如下字段后，单击**创建用户**即可。

字段名	说明
用户名	登录数据库的唯一身份标识，用于验证身份并绑定操作权限。用户名定义有如下限制： • 实例内必须保证唯一性，且创建后不可更改，请谨慎命名 • 长度2-30个字符，支持字母 A-Za-z(不区分大小写)、数字0-9、特殊字符 _-.@，开头需为字母数字或下划线
密码	配合用户名校验身份的机密凭证，防止未授权访问，保障数据安全。密码定义有如下限制： • 用户创建成功后无法再查看密码，请妥善保存 • 可以在「账户管理」页面重置密码 • 长度8~30位字符，必须包含大写字母 A~Z、小写字母 a~z、数字0～9，支持特殊字符，不能以/开头
确认密码	密码的二次输入验证栏，确保两次输入完全一致，避免因输入错误而导致后续登录失败
用户身份	新用户的用户身份，用户身份选择范围取决于当前登录控制台用户的身份： • 超级管理员：可创建 普通用户 或 管理员。

	• 管理员：仅可创建普通用户。
角色权限继承	基于角色的权限控制 (RBAC) TCHouse-X 采用 基于角色的权限控制（RBAC）机制。用户通过绑定一个或多个角色，即可继承相应的操作权限。 为实现对用户权限的更精细化管理，TCHouse-X 系统提供了以下两种角色权限继承模式： • 全部继承：用户绑定角色后，默认继承所有关联角色的聚合权限。 • 切换继承：用户绑定多角色后，支持在当前会话（Session）连接中动态切换活动角色，从而切换继承不同的权限集。
关联腾讯云账号	自定义用户必须与腾讯云主账号/子账号绑定，且一个腾讯云账号只能绑定一个自定义用户，不可重复绑定。

自定义 LDAP 用户

说明：

- 只有在新建实例时在“用户体系”中勾选“自定义 LDAP 用户”选项，该实例才具备创建和使用自定义 LDAP 用户的功能。

1. 用户体系选择 “自定义 LDAP 用户” 进入自定义用户配置页。
2. 配置好如下字段后，单击**创建用户**即可。

字段名	说明
LDAP 用户名	需从已配置的自定义 LDAP 服务的用户列表中选择。
用户身份	同自定义用户
角色权限继承	同自定义用户
关联腾讯云账号	同自定义用户

说明：

角色权限继承说明：

- CAM 用户： 固定使用“全部继承”模式。
- 自定义用户/ LDAP 用户： 支持在“全部继承”与“切换继承”模式间灵活选择。
- 若选择切换继承， 可通过如下方式使用 session role：

- MySQL Client：
  - 查看当前 session role：
    - `show variables like 'role';`
    - `show current roles;`
  - 切换 session role：
    - `set role = 'none'` ：不继承任何已绑定角色权限
    - `set role = 'all'` ：继承所有已绑定角色权限
    - `set role = 'xxxx` ： 继承指定已绑定角色权限
- JDBC Client：TCHouse-X 的 JDBC 连接兼容 MySQL，可通过在连接串中的 `sessionVariables` 参数指定 role：
  - 语法：`sessionVariables=name=value`
  - 示例：`String url = "jdbc:mysql://localhost:33060/mydb?user=myuser&password=mypassword&sessionVariables=role=xxx";`

用户身份编辑

说明：

- 超级管理员用户身份不可更改。
- 超级管理员可授予或撤销用户管理员身份。

进入 TCHouse-X 控制台，通过**实例列表 > 进入实例 > 权限管理 > 用户**路径进入用户列表页面。

腾讯云 CAM 用户

1. 单击**腾讯云 CAM 用户**，进入“腾讯云 CAM 用户”列表。
2. 编辑相应用户的用户身份即可。

自定义用户

1. 单击**自定义用户**，进入“自定义用户”列表。
2. 编辑相应用户的用户身份即可。

自定义 LDAP 用户

1. 单击**自定义 LDAP 用户**，进入“自定义 LDAP 用户”列表。
2. 编辑相应用户的用户身份即可。

用户绑定/解除绑定角色

> **说明：**
> 仅超级管理员可执行用户角色绑定或解除绑定操作。

进入 TCHouse-X 控制台，通过**实例列表 > 进入实例 > 权限管理 > 用户**路径进入用户列表页面。

腾讯云 CAM 用户

1. 单击**腾讯云 CAM 用户**，进入“腾讯云 CAM 用户”列表。
2. 单击需要操作用户的“腾讯云用户 ID”或者操作列表的“详情”，进入“腾讯云 CAM 用户”详情页。
3. 用户绑定角色：单击**绑定角色**，在弹窗中选取需要绑定的角色后，单击**确认**，即可在用户的角色列表中看到已绑定的角色。
4. 用户与角色解除绑定：
   - 解绑单个角色： 用户已经绑定的角色列表中，单击需要解除绑定角色后的“解绑”按钮， 在弹窗中单击“解绑”确认后，系统将解除用户与所选角色的绑定关系。
   - 解绑多个角色：用户已经绑定的角色列表中，勾选需要与用户解除绑定的角色，单击**批量解绑**，在弹窗中单击“解绑”确认后，系统将解除用户与所选角色的绑定关系。

自定义用户

1. 单击**自定义用户**，进入“自定义用户”列表。
2. 单击需要操作用户的“自定义用户名”或者操作列表的“详情”，进入“自定义用户”详情页。
3. 用户绑定角色：单击**绑定角色**，在弹窗中选取需要绑定的角色后，单击**确认**，即可在用户的角色列表中看到已绑定的角色。
4. 用户与角色解除绑定：
   - 解绑单个角色： 用户已经绑定的角色列表中，单击需要解除绑定角色后的“解绑”按钮， 在弹窗中单击“解绑”确认后，系统将解除用户与所选角色的绑定关系。
   - 解绑多个角色：用户已经绑定的角色列表中，勾选需要与用户解除绑定的角色，单击**批量解绑**，在弹窗中单击“解绑”确认后，系统将解除用户与所选角色的绑定关系。

自定义 LDAP 用户

1. 单击**自定义 LDAP 用户**，进入“自定义 LDAP 用户”列表。
2. 单击需要操作用户的“自定义 LDAP 用户名”或者操作列表的“详情”，进入“自定义 LDAP 用户”详情页。
3. 用户绑定角色：单击**绑定角色**，在弹窗中选取需要绑定的角色后，单击**确认**，即可在用户的角色列表中看到已绑定的角色。
4. 用户与角色解除绑定：
   - 解绑单个角色： 用户已经绑定的角色列表中，单击需要解除绑定角色后的**解绑**按钮， 在弹窗中单击“解绑”确认后，系统将解除用户与所选角色的绑定关系。
   - 解绑多个角色：用户已经绑定的角色列表中，勾选需要与用户解除绑定的角色，单击**批量解绑**，在弹窗中单击“解绑”确认后，系统将解除用户与所选角色的绑定关系。

说明：
TCHouse-X 用户批量解绑角色时，一次最多解绑 30 个角色。

用户删除

进入 TCHouse-X 控制台，通过**实例列表 > 进入实例 > 权限管理 > 用户**路径进入用户列表页面。

说明：
- 超级管理员不可被删除。
- 超级管理员可删除管理员、普通用户。
- 管理员仅可删除普通用户。
- 普通用户不可删除任何用户。

单用户删除

单击**腾讯云 CAM 用户**或**自定义用户**或**自定义 LDAP 用户**，进入指定体系用户列表，在相应用户后单击**删除**即可。

多用户删除

单击**腾讯云 CAM 用户**或**自定义用户**或**自定义 LDAP 用户**，进入指定体系用户列表，勾选用户，单击**批量删除用户**即可。

注意：

> TCHouse-X 批量删除用户时，一次最多删除 30 个用户。

角色管理

最近更新时间：2026-05-06 16:28:12

角色 (Role) 代表一组权限的集合，而不是一个具体的用户。TCHouse-X 简化授权过程，实现了基于角色的访问控制 (Role-Based Access Control, RBAC)。

> **注意：**
> 在 TCHouse-X 中，只有超级管理员（及腾讯云主账号和 TCHouse-X 实例创建子账号）可执行权限管理相关操作。

角色创建

进入 TCHouse-X 控制台，通过**实例列表 > 进入实例 > 权限管理 > 创建角色**路径进入角色创建页面，填写角色名称、角色描述信息，单击**创建角色**即可。

角色删除

1. 进入 TCHouse-X 控制台，通过**实例列表 > 进入实例 > 权限管理 > 角色**路径进入角色列表。
2. 根据业务需求，删除单个角色或者批量删除角色：
   - 删除单个角色：在相应角色后单击**删除**即可。
   - 批量删除角色：勾选需要删除的角色，单击**批量删除**即可。

> **注意：**
> 在 TCHouse-X 中进行批量删除角色操作时，系统限制用户一次最多只能删除 30 个 角色。

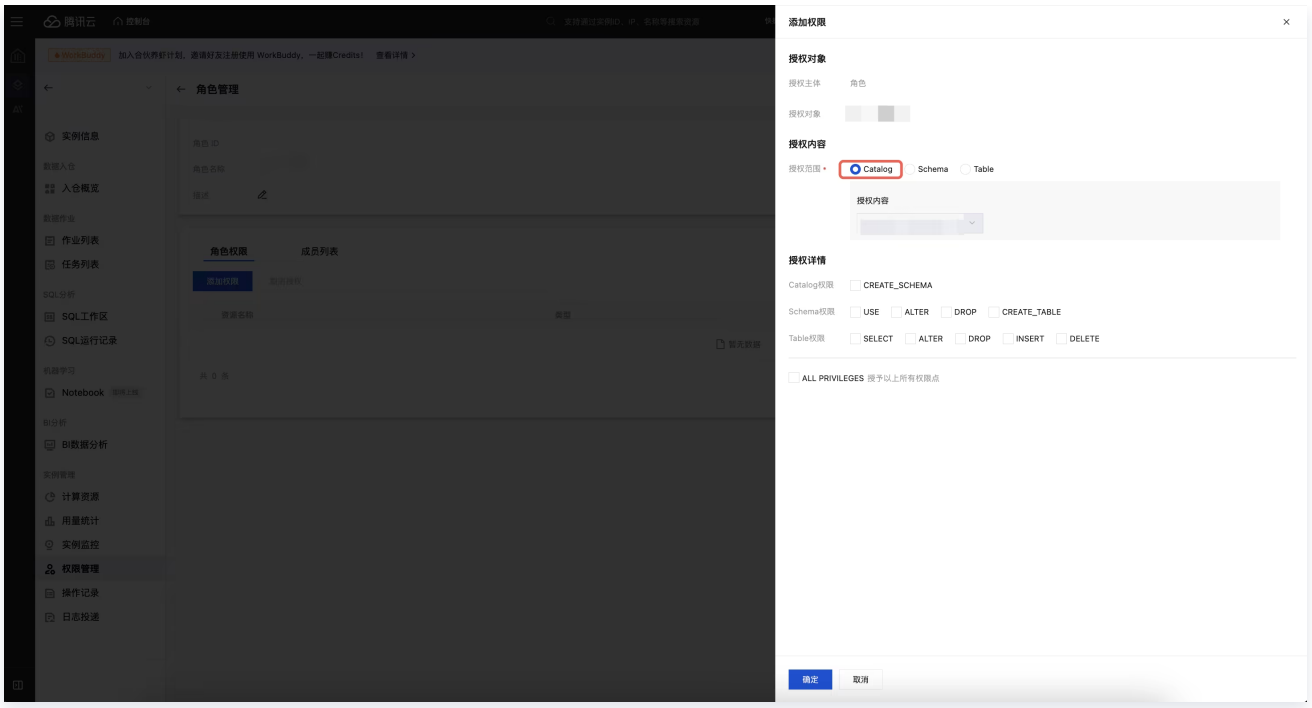
角色权限管理

进入 TCHouse-X 控制台，通过**实例列表 > 进入实例 > 权限管理 > 角色**路径进入角色列表，找到需要编辑权限的角色，单击**角色名称**或者角色操作列中的**管理权限**，进入角色详情页。

添加权限

添加 Catalog 权限

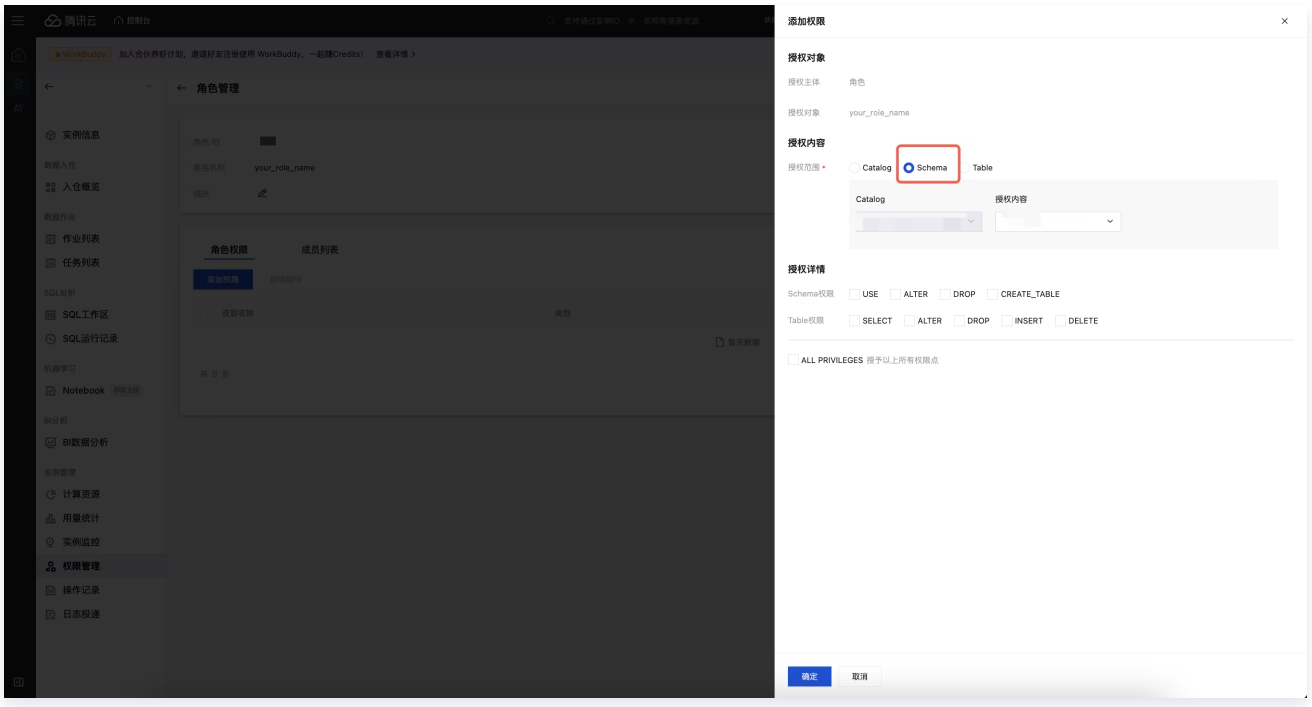
1. 单击**添加权限**，在右侧弹窗中将**授权内容**设为 Catalog。



2. 根据需求勾选或配置具体的权限详情。
3. 单击**确认**保存。页面将自动返回角色权限列表，您可在此查看新增的权限。

添加 Schema 权限

1. 单击**添加权限**，在右侧弹窗中将**授权内容**设为 Schema，并指定目标 Schema（Database）名称。

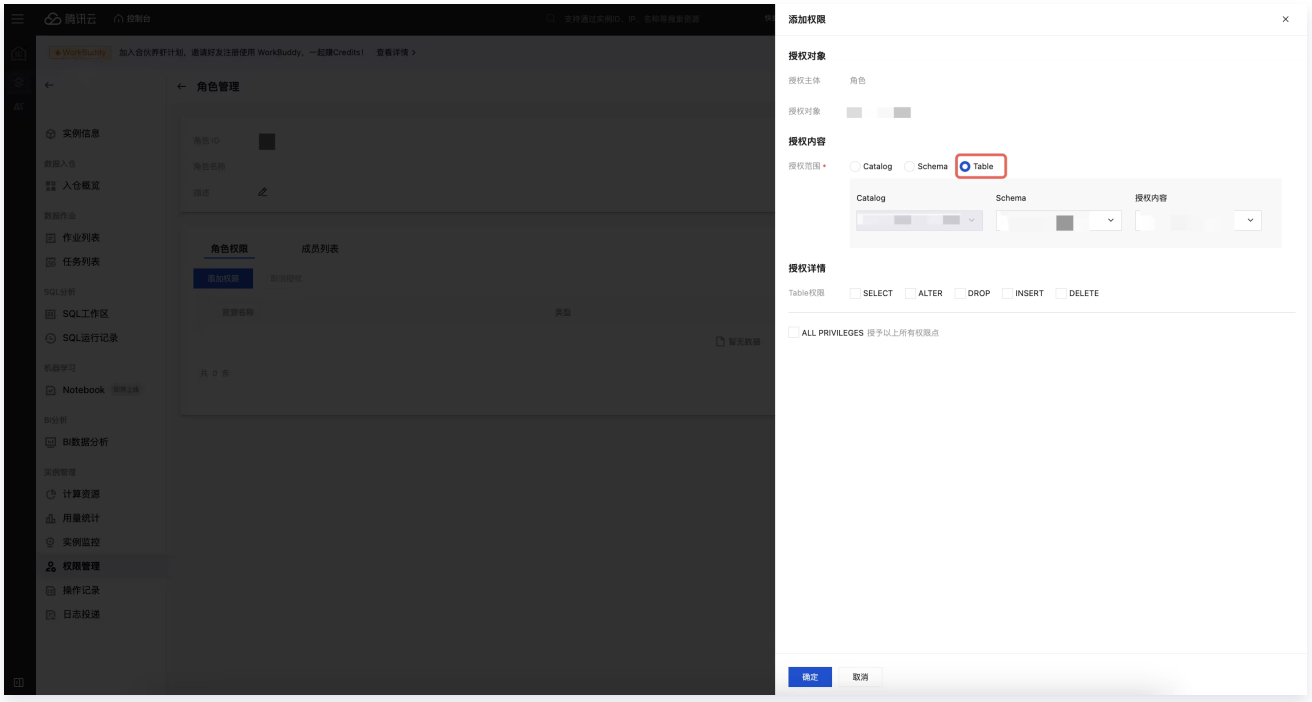


2. 在**授权详情**区域，勾选或配置您需要的具体权限。

3. 单击**确认**保存。页面将自动跳转回角色权限列表，您可在此查看新配置的权限。

添加 Table 权限

1. 单击 **添加权限**，在右侧弹窗中将 **授权内容** 设为 Table，并指定所属 Schema（Database）及目标 Table 名称。

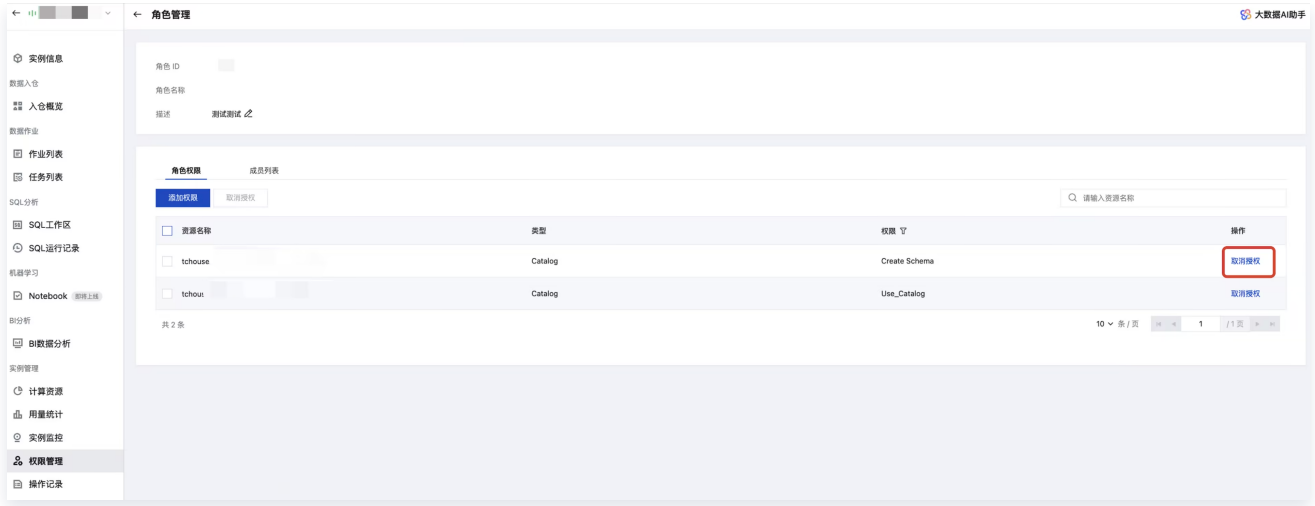


2. 在**授权详情**中，勾选或设置需要配置的具体权限。
3. 单击**确认**保存。页面将自动跳转回角色权限列表，您可在此查看新配置的权限。

取消授权

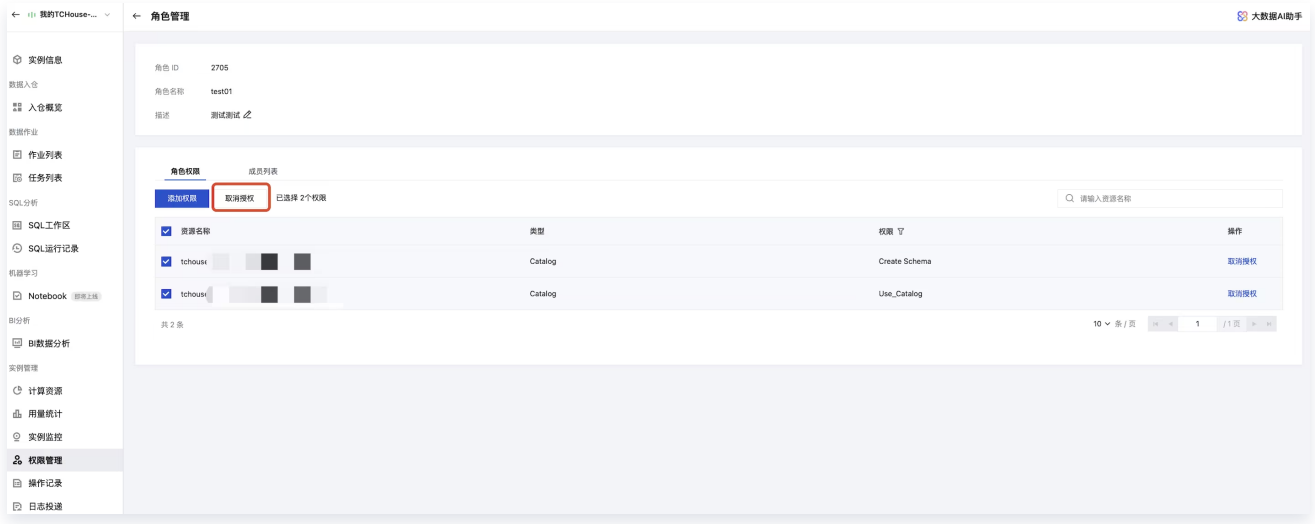
取消单个授权

1. 定位到目标权限。
2. 在其对应的操作列中，单击**取消授权** 。即可完成授权的取消。



取消多个授权

1. 定位到目标权限并勾选。
2. 单击**取消授权**，即可完成授权的取消。



角色成员管理

访问 TCHouse-X 控制台，通过"实例列表 > 进入实例 > 权限管理 > 角色"路径进入角色列表。找到需要管理成员的角色后，可通过以下两种方式进入成员管理界面：

1. 单击**角色名称 > 成员列表**进入角色成员列表。
2. 在“操作”列中单击**管理成员**进入角色成员列表。

添加成员

添加腾讯云 CAM 用户

1. 单击**添加成员**，系统将弹出成员配置页。
2. 在配置页中，选择用户体系为“腾讯云 CAM 用户”，勾选需要绑定的用户，然后单击**确认**完成添加。
3. 返回“成员列表”页，您即可在“腾讯云 CAM 用户”列表中查看到新添加的成员。

添加自定义用户

1. 单击**添加成员**，系统将弹出成员配置页。
2. 在配置页中，选择用户体系为“自定义用户”，勾选需要绑定的用户，然后单击**确认**完成添加。
3. 返回“成员列表”页，您即可在“自定义用户”列表中查看到新添加的成员。

添加自定义 LDAP 用户

1. 单击“添加成员”按钮，系统将弹出成员配置页。
2. 在配置页中，选择用户体系为“自定义 LDAP 用户”，勾选需要绑定的用户，然后单击“确认”完成添加。
3. 返回“成员列表”页，您即可在“自定义 LDAP 用户”列表中查看到新添加的成员。

删除成员

删除腾讯云 CAM 用户

1. 进入成员管理，在成员列表中，选择 “腾讯云 CAM 用户” 选项卡，进入对应的成员管理页面。
2. 移除成员您可以根据需求选择“单个删除”或“批量删除”操作：
   - 单个删除： 在列表中定位目标成员，单击其右侧操作列的**删除**。
   - 批量删除： 勾选列表中一个或多个成员，单击顶部的**批量删除**。

删除自定义用户

1. 进入成员管理，在成员列表中，选择 “自定义用户” 选项卡，进入对应的成员管理页面。
2. 移除成员您可以根据需求选择“单个删除”或“批量删除”操作：
   - 单个删除： 在列表中定位目标成员，单击其右侧操作列的 **删除**。

- 批量删除： 勾选列表中一个或多个成员，单击顶部的**批量删除**。

删除自定义 LDAP 用户

1. 进入成员管理，在成员列表中，选择 “自定义 LDAP 用户” 选项卡，进入对应的成员管理页面。
2. 移除成员您可以根据需求选择“单个删除”或“批量删除”操作：
   - 单个删除： 在列表中定位目标成员，单击其右侧操作列的**删除**。
   - 批量删除： 勾选列表中一个或多个成员，单击顶部的**批量删除**”。

连接实例

使用 MySQL Client 连接实例

最近更新时间：2026-05-06 16:28:12

原生 MySQL Client

原生 MySQL Client 不支持腾讯云 CAM 认证，仅适用于独立用户和第三方 LDAP 账号连接。具体用户可参考文档 The MySQL Command-Line Client 。

TCHouse-X MySQL Client

TCHouse-X Client 兼容原生 MySQL 客户端用法，并新增了对 CAM 认证的支持。

环境依赖

在运行客户端之前，请确保系统已安装 OpenSSL 1.0 库：

```
yum install -y compat-openssl10
```

工具下载

TCHouse-X 提供 Linux 和 Mac 版本的多种 MySQL 版本兼容客户端：

版本	Linux 版	Mac 版
5.7.44	5.7.44-linux-glibc2.17-x86_64-minimal.tar.gz	5.7.44-osx10.20-arm64-minimal.tar.gz
8.0.44	8.0.44-linux-glibc2.17-x86_64-minimal.tar.gz	8.0.44-macos15.6-arm64-minimal.tar.gz
9.5.0	9.5.0-linux-glibc2.17-x86_64-minimal.tar.gz	9.5.0-macos15.6-arm64-minimal.tar.gz

客户端配置

```
# 进入目标路径
cd <your path>

# 下载客户端，以 5.7.44 版本的 linux 客户端为例
wget <MySQL Client Download URL>
```

```
# 客户端解压并重命名文件夹，以 5.7.44 版本的 linux 客户端为例
tar -zxvf mysql-tchousex-5.7.44-linux-glibc2.17-x86_64-minimal.tar.gz &&
mv mysql-tchousex-5.7.44-linux-x86_64-minimal mysql-tchousex

# 配置环境变量
echo "export PATH=$PATH:/<your path>/mysql-tchousex/bin" >> ~/.bashrc
source ~/.bashrc
```

新增参数说明

参数	描述
`--cam-login`	用于指定是否为腾讯云 CAM 用户登录。1 – 是, 0 – 否。
`--token`	用于指定腾讯云临时密钥的 Token 值。

客户端登录样例

认证方式	说明	命令示例

独立用户和 LDAP 用户登录	与原生 MySQL Client 用法一致	`mysql-tchousex \`
		`-h**.**.**.** \`
		`-uroot \`
		`-P33060 \`
		`-p`

腾讯云持久密钥登录	• `-u` 填 secretId
	• `-p` 填 secretKey
	• `--cam-login` 填 1
		`mysql-tchousex \`
		` -h172.**.**.** \`
		` -u'********************' \`
		` -P33060 \`
		` -p \`
		` --cam-login=1`

腾讯云临时密钥登录	• `-u` 填临时密钥的 secretId	`mysql-tchousex \`

- `-p` 填临时密钥的 secretKey
- `--cam-login` 填 1
- `--token` 填临时密钥的 token

```
  -h**.**.**.** \
  -u'***************' \
  -P33060 \
  -p \
  --
token='****************************'
\
  --cam-login=1
```

使用 JDBC 连接实例

最近更新时间：2026-05-06 16:28:12

TCHouse-X 与 MySQL 完全兼容，您可以选择使用原生 MySQL JDBC 或 TCHouse-X 专属连接器。

原生 MySQL JDBC

此方法适用于非腾讯云 CAM 用户进行连接。

依赖配置 (Maven)

使用 Maven 项目时，请添加以下依赖：

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.33</version>
</dependency>
```

编写代码

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class MySQLJDBCExample {
    // Database connection parameters
    private static final String URL =
"jdbc:mysql://localhost:3306/your_database?
useSSL=false&serverTimezone=UTC";
    private static final String USERNAME = "your_username";
    private static final String PASSWORD = "your_password";


    public static void main(String[] args) {
        Connection connection = null;
```

```
        try {
            // Load the JDBC driver (This step can be omitted for MySQL
8.0+).
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Establish database connection
            connection = DriverManager.getConnection(URL, USERNAME,
PASSWORD);
            System.out.println("Successfully established database
connection!");

            // Performing various database operations
            createTable(connection);
            insertData(connection);
            queryData(connection);
            updateData(connection);
            deleteData(connection);

        } catch (ClassNotFoundException e) {
            // Handling JDBC driver not found issue
        } catch (SQLException e) {
            // Handling database operation errors
        } finally {
            // Closing database connection
        }
    }


    // Create table
    private static void createTable(Connection connection) throws
SQLException {
        String sql = "CREATE TABLE IF NOT EXISTS users (" +
                     "id INT, " +
                     "name STRING, " +
                     "age INT" +
                     ")";

        try (Statement statement = connection.createStatement()) {
            statement.executeUpdate(sql);
            System.out.println("Data Created successfully");
```

```
        }
    }

    // Insert data
    private static void insertData(Connection connection) throws
SQLException {
        String sql = "INSERT INTO users (id, name, age) VALUES (?, ?,
?)";

        try (PreparedStatement pstmt = connection.prepareStatement(sql))
{
            // Insert the first record
            pstmt.setInt(1, 1);
            pstmt.setString(2, "bob");
            pstmt.setInt(3, 25);
            pstmt.executeUpdate();

            // Insert the second record
            pstmt.setInt(1, 2);
            pstmt.setString(2, "jimmy");
            pstmt.setInt(3, 30);
            pstmt.executeUpdate();

            System.out.println("Data Inserted successfully");
        }
    }

    // Query data
    private static void queryData(Connection connection) throws
SQLException {
        String sql = "SELECT id, name, age FROM users";

        try (Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery(sql)) {

            System.out.println("ID\tname\tage");
            System.out.println("----------------------------------------
");

            while (resultSet.next()) {
```

```
                int id = resultSet.getInt("id");
                String name = resultSet.getString("name");
                int age = resultSet.getInt("age");
                System.out.printf("%d\t%s\t%d%n", id, name, age);
            }
        }
    }

    // update data
    private static void updateData(Connection connection) throws
SQLException {
        String sql = "UPDATE users SET age = ? WHERE name = ?";

        try (PreparedStatement pstmt = connection.prepareStatement(sql))
{
            pstmt.setInt(1, 26);
            pstmt.setString(2, "alex");

            int rowsAffected = pstmt.executeUpdate();
            System.out.println("Successfully updated " + rowsAffected +
" records");
        }
    }

    // delete data
    private static void deleteData(Connection connection) throws
SQLException {
        String sql = "DELETE FROM users WHERE age > ?";

        try (PreparedStatement pstmt = connection.prepareStatement(sql))
{
            pstmt.setInt(1, 28);

            int rowsAffected = pstmt.executeUpdate();
            System.out.println("Successfully deleted " + rowsAffected +
" records");
        }
    }
 }
```

> **注意：**
> 当您的登录用户属于第三方 LDAP 认证体系时，需要根据认证方式在 URL 中配置 defaultAuthenticationPlugin 属性。

认证方式	URL 参数示例
明文密码传输	...&defaultAuthenticationPlugin=mysql_clear_password
SHA256 加密传输	...&defaultAuthenticationPlugin=sha256_password&allowPublicKeyRetrieval=true

> **注意：**
> 要启用 SSL，请在 URL 中增加 `useSSL=true&allowPublicKeyRetrieval=true`。

TCHouse-X JDBC

TCHouse-X 专属连接器在原生功能基础上，增加了对腾讯云 CAM 密钥的认证支持。

驱动下载与依赖配置

项目	说明
下载地址	tchousex-connector-j-8.4.0.jar

Maven 依赖	`<dependency>`
	`  <groupId>com.tchousex</groupId>`
	`  <artifactId>tchousex-connector-j</artifactId>`
	`  <version>8.4.0</version>`
	`</dependency>`

连接方式 (URL 模板)

认证类型	URL 格式	必须参数
腾讯云用户登录 (推荐)	`jdbc:tchousex://{host}:{port}/{dbname}?camSecretId=%s&camSecretKey=%s`	• `camSecretId` • `camSecretKey`

	`&camToken=%s`	• `camToken` (临时密钥) 或留空 (持久密钥)
独立用户登录	`jdbc:tchousex://{host}:{port}/{dbname}?user=%s&password=%s`	• `user` • `password`

注意:
要启用 SSL，请在 URL 中增加 `useSSL=true&allowPublicKeyRetrieval=true` 。

使用 Spark-Submit 命令行工具连接实例

最近更新时间：2026-05-06 16:28:12

腾讯云数据仓库 TCHouse-X 提供了定制化的 Spark-Submit 命令行工具。该工具是向 TCHouse-X 离线计算引擎 提交 Spark 应用的官方接口。它在开源 Spark-Submit 的基础上进行了扩展和优化，旨在实现与 TCHouse-X 环境的深度集成和高效运行。

TCHouse-X Spark-Submit 用法指南

Step1：下载 CLI 工具

1. 登录您计划提交 Spark 作业的 Linux 客户端机器，使用 `curl` 或 `wget` 命令下载 TCHouse-X-Spark-Submit 命令行工具压缩包：

```
curl -O https://tchousex-doc-1375622477.cos.ap-guangzhou.myqcloud.com/spark-submit/tchouse-x-toolkit-spark-submit-1.0.0.zip
```

或者

```
wget https://tchousex-doc-1375622477.cos.ap-guangzhou.myqcloud.com/spark-submit/tchouse-x-toolkit-spark-submit-1.0.0.zip
```

> ⚠ **注意：**
> 若您的 Linux 环境中未安装 `curl` 或 `wget`，请根据系统类型使用相应的包管理器进行安装（例如：`sudo apt install wget` 或 `sudo yum install curl`）。

2. 下载完成后，执行以下命令解压工具压缩包：

```
unzip tchouse-x-toolkit-spark-submit-1.0.0.zip
```

3. 解压操作将创建一个名为 `tchouse-x-spark-submit` 的目录。该目录是工具的根目录，其结构如下：
   - `<bin>` 目录： 包含核心的可执行程序 `spark-submit`，用于在 Linux 环境中提交作业。
   - `<conf>` 目录： 包含配置文件 `spark-defaults.conf`，用于配置 TCHouse-X 集群连接信息和默认运行参数。

Step2：配置连接参数

进入 `<tchouse-x-spark-submit>` 根目录，编辑 `conf/spark-defaults.conf` 文件，配置连接 TCHouse-X 实例所需的参数。

默认配置文件内容预览

```
## 请配置腾讯云CAM用户的SecretID和SecretKey，获取方式请参见：
https://cloud.tencent.com/document/product/598/37140
## 必填；若不配置，执行命令会失败
secretId =
secretKey =

## 请配置TCHouse-X实例所在的地域，暂仅支持广州地域(ap-guangzhou)
## 必填；暂不支持修改；若修改，执行命令会失败
region = ap-guangzhou

## 可配置默认使用的TCHouse-X实例ID，实例ID可在控制台查看：
https://console.cloud.tencent.com/tchousex?region=ap-guangzhou
## 非必填；若不配置、或配置后需使用其他实例，可在执行命令时指定--instanceId
<instanceId>
#instanceId =


## 可配置默认使用的driver资源规格
## 非必填；默认值是2x-small，可选值{2x-small，x-small，small，medium}，分别对
应{4，8，16，32}CU，1CU=1核CPU4GB内存
spark.driver.resourceSpec = 2x-small

## 可配置默认使用的executor资源规格
## 非必填；默认值是2x-small，可选值{2x-small，x-small，small，medium}，分别对
应{4，8，16，32}CU，1CU=1核CPU4GB内存
spark.executor.resourceSpec = 2x-small

## 可配置默认使用的executor数量
## 非必填；默认值是1，可选值[1，999]
spark.executor.instances = 1

## 可配置默认使用的存储上传文件的COS桶名称
## 非必填；若不配置、或配置后需使用其他cos桶，可在执行--upload命令时指定--bucket
<bucketName>
#cosUploadBucket =

## 可配置其他以spark开头的参数的默认值
```

```
## 非必填；若不配置、或配置后需使用其他参数值，可在执行提交命令时通过--conf <key>=<value>指定
```

必填参数配置说明

参数名称	参数说明	参数值说明	示例
secretId	用户访问腾讯云 API 进行身份验证时需要用到的安全凭证，由 SecretId 和 SecretKey 一起组成，SecretId用于标识 API 调用者身份	获取方式请参见子账号访问密钥管理	secretId = q**********f
secretKey	用户访问腾讯云 API 进行身份验证时需要用到的安全凭证，由 SecretId 和 SecretKey 一起组成，SecretKey 用于验证 API 调用者身份	获取方式请参见子账号访问密钥管理	secretKey = b**********n
region	TCHouse-X 实例所在的地域		region = ap-guangzhou

选填参数配置说明

参数名称	描述	默认值及可选值	示例配置
instanceId	TCHouse-X 实例 ID。	非必填。配置后，提交命令将默认使用该实例。若要覆盖，可在提交时指定 --instanceId。	instanceId = warehouse-p******m
spark.driver.resourceSpec	Spark 作业 Driver 进程的资源规格。	默认值：2x-small。 可选值：{2x-small, x-small, small, medium} 规格对应：2x-small=4C16G；medium=32C128G。	spark.driver.resourceSpec = 2x-small
spark.executor.resourceSpec	Spark 作业 Executor 进程的资源规格。	默认值：2x-small。 可选值：{2x-small, x-small, small, medium} 规格对应：2x-small=4C16G；medium=32C128G。	spark.executor.resourceSpec = x-small
spark.executor.i	运行 Spark 作业所需的 Executor 数量。	默认值：1。 可选值：[1, 999]。	spark.executor.instances = 5

nstances			
cosUploadBucket	用于文件上传（如 JAR 包）的 COS 存储桶名称。	非必填。若不配置，执行 --upload 命令时必须指定 --bucket 参数。	cosUploadBucket = my-bucket-12345678

⚠ 注意：

1. **大小写敏感：** 所有参数名称及其对应值都是大小写敏感的。
2. **配置扩展：** 允许添加其他以 spark. 开头的配置参数作为默认值，但不支持添加其他非 spark 开头的 option 参数。
3. **优先级：** `spark-defaults.conf` 中的配置项会被命令行提交时通过 `--conf <key>=<value>` 指定的参数覆盖（仅单次提交生效）。

Step3：提交 Spark 作业

进入 `<tchouse-x-spark-submit>` 目录，执行 `bin/spark-submit` 命令提交您的 Spark 应用程序。

提交命令语法

以下是 Spark-Submit 命令的完整结构。请注意，应用程序文件和程序参数必须放在命令的末尾。

```
./bin/spark-submit \
    --name <job-name> \
    [--class <main-class>] \
    [--jars <dependency-jar-path>] \
    [--py-files <dependency-py-files-path>] \
    [--files <dependency-files-path>] \
    [--archives <dependency-archives-path>]
    [--driver-cores <value> | --conf spark.driver.resourceSpec=<value>]
\
    [--executor-cores <value> | --conf spark.executor.resourceSpec=
<value>] \
    [--num-executors <value> | --conf spark.executor.instances=<value>]
\
    [--config <key>=<value> | --conf <key>=<value>] \
    [--confPath <confPath>] \
    [--instanceId <instanceId>] \
    <application-file> [<application-arguments>]
```

请求参数说明

必填参数

参数名称	描述	约束与说明	示例
--name <job-name>	作业的显示名称，将显示在 TCHouse-X 控制台页面。	必填。支持中文、英文、数字、_ 和 -，最多 100 个字符。	--name Submit_ETL_Job
<application-file>	Spark 主程序包的存储路径。	必填。必须是 cosn:// 开头的绝对路径。支持 JAR 和 Py 文件。	cosn://bucket1/dir1/sparkjar.jar

依赖文件参数 (路径均需以 cosn:// 开头)

参数名称	描述	支持文件类型	格式与说明
--jars	依赖的 Java/Scala JAR 文件。	jar	多个文件间以英文逗号 , 分隔。
--py-files	PySpark 依赖文件。	py, zip, egg	多个文件间以英文逗号 , 分隔。
--files	分发到所有节点的附加文件（如配置文件）。	jar, zip	多个文件间以英文逗号 , 分隔。
--archives	分发的归档文件（如虚拟环境）。	gz, tgz, tar	必须在路径后配置打包方式（如 #venv）。

资源与配置参数 (可覆盖配置文件中的默认值)

参数名称	别名/配置键	描述	可选值	优先级说明	示例
--driver-cores	spark.driver.resourceSpec	Driver 进程的资源规格。	Cores: [4, 8, 16, 32] Spec: {2x-small, x-small, small, medium}	两者任选其一；--conf 优先级高于 --driver-cores。	--conf spark.driver.resourceSpec=x-small
--executor-cores	spark.executor.resourceSpec	Executor 进程的资源规格。	Cores: [4, 8, 16, 32] Spec: {2x-small, x-small, small, medium}	两者任选其一；--conf 优先级高于 --executor-cores。	--executor-cores 16
--num-executors	spark.executor.instances	Executor 数量。	[1, 999]	两者任选其一；--conf 优先级高于 --num-executors。	--num-executors 8
--conf 或 --config	N/A	配置其他以 spark. 开头的参数。	<key>=<value> 格式的 Spark 参数。	每次提交生效；命令行指定的值会覆盖 spark-defaults.conf。	--conf spark.sql.shuffle.partitions=20000
--confPath	N/A	指定自定义配置文件路径。	相对路径或绝对路径。	默认值：./conf/spark-defaults.conf。	--confPath /data/custom.conf

环境参数

参数名称	别名	描述	说明	示例
--instanceId	N/A	目标 TCHouse-X 实例 ID。	若未在配置文件中设置，则必填。	--instanceId instance-p******m
--class	N/A	Java/Scala 程序的主入口类名。	如果提交的是 JAR 文件，则必填。PySpark 不需此参数。	--class org.apache.spark.examples.SparkPi
<application-arguments>	N/A	主程序包的程序入口参数。	必须放在命令末尾。包含空格等特殊字符时，请使用英文双引号包裹。	arg1 "SQL statement"

注意：

1. **位置严格要求：** `<application-file>` **和** `<application-arguments>` **必须按顺序放在命令行的末尾。其他参数顺序可自由调整。**
2. **参数优先级：**
   - **命令行中多次指定同一参数，取最后一次指定的值。**
   - **同时使用 -cores/-executors 和 --conf spark.***.resourceSpec/instances 时，以 --conf 指定的值为准。**
3. **覆盖关系： 命令行参数会覆盖** `spark-defaults.conf` **中的默认配置。**

提交样例

```
./bin/spark-submit \
--name submit-Cli_job1 \
--class org.apache.spark.examples.SparkSql \
--jars cosn://bucket1/path1/jar1.jar,cosn://bucket1/path1/jar2.jar,
cosn://bucket1/path1/jar3.jar \
cosn://bucket1/dir1/sparkjar.jar "-sql create table db1.dt3 as select *
from db1.dt1;"
```

返回参数说明

成功提交 Spark 作业后，命令行工具将立即返回以下信息。TCHouse-X 会自动生成作业 ID 和运行任务 ID。

字段名称	描述	示例值

RequestId	腾讯云 API 请求的唯一标识。	1******************6
ErrorMsg	错误信息（提交成功时为空）。	(空)
SparkJobId	TCHouse-X 中生成的作业 ID（Job ID）。	batch-job-123456
SparkTaskId	当前提交的运行任务 ID（Task ID）。	batch-task-123456

返回示例

```
RequestId    1******************6
ErrorMsg
SparkJobId    batch-job-123456
SparkTaskId   batch-task-123456
```

其他操作说明

查看参数说明

进入 `<tchouse-x-spark-submit>` 目录, 执行以下命令, 查看操作说明：

```
./bin/spark-submit --help
```

查看任务运行状态

进入 `<tchouse-x-spark-submit>` 目录，执行以下命令，使用已获取的 SparkTaskId 查询 Spark 作业任务的实时状态和详情。

```
./bin/spark-submit --status \
--taskId <taskId> \
```

请求参数说明

参数名称	是否必填	描述	示例
`--status`	是	指定执行查看任务状态的命令。	`--status`
`--taskId <taskId>`	是	需要查询的 Spark 作业任务 ID。	`--taskId batch-task-123456`

返回参数说明

参数名称	描述	参数值说明	示例
`RequestId`	腾讯云 API 请求的唯一标识 ID。	-	`RequestId 6****** ****************** ******b`
`ErrorMsg`	错误信息。	若查询成功，则该字段为空。	ErrorMsg error: wrong taskId
`Status`	任务运行状态。	任务状态列表 • `initializing`：初始化中 • `running`：运行中 • `success`：运行成功 • `failed`：运行失败 • `canceled`：已取消运行	`Status running`
`BeginTime`	Spark 任务开始运行的时间。	格式：`YYYY-MM-DD HH:MM:SS`	`BeginTime 2024-10-29 10:02:45`
`EndTime`	Spark 任务结束运行的时间。	仅在任务状态为 `success`、`failed` 或 `canceled` 时存在。	`EndTime 2024-10-29 10:03:30`
`ExecuteTime`	Spark 任务运行总耗时。	带有单位（如 `s`、`min`）。	`ExecuteTime 45s`
`SparkErrLog`	Spark 任务运行中的错误日志摘要。	若任务无错误，则值为空。	`SparkErrLog wrong sparkContext`

查看任务运行日志详情

进入 `<tchouse-x-spark-submit>` 目录，执行 `--get-log` 命令获取指定 Spark 作业任务的详细运行日志，以便进行故障诊断和流程跟踪。

```
./bin/spark-submit --get-log \
    --taskId <taskId> \
    [--pod <podName>] \
    [--orderBy <desc | asc>] \
    [--limit <limit>] \
    [--confPath <confPath>] \
```

请求参数说明

参数名称	是否必填	描述	可选值/说明	默认值/示例
`--get-log`	是	指定执行查看任务日志的命令。	无参数值。	`--get-log`
`--taskId <taskId>`	是	需要查询日志的 Spark 任务 ID。	可通过 `--list` 命令查看已提交的任务列表。	`--taskId batch-task-123456`
`--pod <podName>`	否	期望查看的 Pod（Driver 或 Executor）日志。	可选值：`{driver, exec-0, exec-1, ...}` (Executor 编号从 0 开始)。	默认值：`driver` 示例：`--pod exec-1`
`--orderBy <desc | asc>`	否	日志的排序方式（时间戳）。	可选值：`{desc, asc}`。	默认值：`desc`
`--limit <limit>`	否	单次请求返回的日志行数限制。	范围：`(0, 1000]`。	默认值：`1000` 示例：`--limit 100`
`--confPath <confPath>`	否	指定自定义配置文件路径。	支持相对路径、绝对路径。	默认值：`./conf/spark-defaults.conf`

请求示例

```
# 获取 batch-task-123456 任务中第一个 Executor 的日志，并按时间升序排列：
./bin/spark-submit --get-log \
    --taskId batch-task-123456
    --pod exec-0
    --orderBy asc
```

返回参数说明

参数名称	描述	示例值

`RequestId`	腾讯云 API 请求的唯一标识 ID。	`6**********************************b`
`ErrorMsg`	错误信息。	若查询成功，则该字段为空。
`SparkLogList`	Spark 任务的运行日志内容。	(返回实际的日志文本内容)

查看作业列表

进入 `<tchouse-x-spark-submit>` 目录，执行 `--list` 命令，查询目标 TCHouse-X 实例下用户提交的 Spark 作业（Job）列表。

```
./bin/spark-submit --list \
    [--confPath <confPath>] \
    [--instanceId <instanceId>] \
    [--pageSize <pageSize>] \
    [--pageNumber <pageNumber>]
```

请求参数说明

参数名称	是否必填	描述	默认值/说明
`--list`	是	指定执行查看作业列表的命令。	无参数值。
`--instanceId <instanceId>`	条件必填	期望查看的 Spark 作业列表所属的 TCHouse-X 实例 ID。	若已在配置文件中设置，可省略。主账号内实例 ID 唯一。
`--pageSize <pageSize>`	否	返回的 Spark 作业列表中每页显示的记录数。	默认值：`10`。可选值：大于 0 的整数。
`--pageNumber <pageNumber>`	否	期望查看的页数。	默认值：`1`。
`--confPath <confPath>`	否	指定自定义配置文件路径。	默认值：`./conf/spark-defaults.conf`。

返回参数说明

执行 `--list` 命令后，系统返回结果将包含请求信息摘要、总数以及作业列表的详细字段。

参数名称	描述	参数值说明	示例
`RequestId`	腾讯云 API 请求的唯一标识 ID。	-	`RequestId 6*******b`
`ErrorMsg`	错误信息。	若请求成功，该字段为空。	`ErrorMsg error: wrong instanceId`
`TotalCount`	符合查询条件的总作业数量。	-	`TotalCount 70`
`InstanceId`	Spark 作业所属的 TCHouse-X 实例 ID。	-	`InstanceId warehouse-p******m`
`JobId`	Spark 作业 ID（Job ID）。	提交作业时自动生成的唯一标识。	`SparkJobId batch-job-*****`
`SparkName`	Spark 作业名称。	提交时 `--name` 参数的值。	`SparkName ETL_Processing_Job`
`File`	Spark 作业主程序包的 COS 存储路径。	提交时 `<application-file>` 参数的值。	`File cosn://bucket1/dir1/sparkjar.jar`
`ClassName`	Java 或 Scala 程序的主入口类名。	提交时 `--class` 参数的值。	`ClassName org.apache.spark.examples.SparkPi`
`Args`	Spark 主程序包的程序入口参数。	提交时 `<application-arguments>` 参数的值。	`Args arg1 arg2`
`Configs`	Spark 运行时配置参数。	提交时 `--conf` 参数的值。	`Configs spark.network.timeout=120s`
`Jars`	依赖的 Jar 文件路径列表。	提交时 `--jars` 参数的值。	`Jars cosn://bucket1/path1/jar1.jar`
`Files`	依赖的 Files 文件路径列表。	提交时 `--files` 参数的值。	`Files cosn://bucket1/path1/jar1.zip`

PyFiles	依赖的 PyFiles 文件路径列表。	提交时 --py-files 参数的值。	PyFiles cosn://bucket1/path1/pyfiles.py
Archives	依赖的 Archives 文件路径列表。	提交时 --archives 参数的值。	Archives cosn://bucket1/path1/archives.gz#venv
DriverCores	Spark Driver 资源的核数。	根据提交时配置的资源规格（spark.driver.resourceSpec 或 --driver-cores）转换。	DriverCores 4
ExecutorCores	Spark Executor 资源的核数。	根据提交时配置的资源规格（spark.executor.resourceSpec 或 --executor-cores）转换。	ExecutorCores 4
ExecutorNum	Spark Executor 的数量。	提交时配置的 spark.executor.instances 或 --num-executors 参数的值。	ExecutorNum 4
CreateTime	Spark 作业在服务端创建的时间。	-	CreateTime 2024-10-29 00:06:24
ModifyTime	Spark 作业的最后修改时间。	-	ModifyTime 2024-10-29 00:06:28
Creator	提交 Spark 作业的用户 UIN（主账号 ID）。	配置文件中 secretId 对应的用户 UIN。	Creator 747951370
UserName	提交 Spark 作业时使用的 TCHouse-X 用户名。	-	UserName user1
CurrentTaskNum	该 Spark 作业下当前正在运行的任务数量。	-	CurrentTaskNum 0

取消运行任务

进入 <tchouse-x-spark-submit> 目录，执行 --kill 命令可以中断并取消正在运行中的 Spark 作业任务。

```
./bin/spark-submit --kill \
    --taskId <taskId> \
    [--instanceId <instanceId>]
```

请求参数说明

参数名称	是否必填	描述	默认值/说明	示例
`--kill`	是	指定执行取消运行任务的命令。	无参数值。	`--kill`
`--taskId <taskId>`	是	期望取消运行的 Spark 任务 ID。	可通过 `--list` 命令或提交成功后的返回结果获取。	`--taskId batch-task-******g`
`--instanceId`	条件必填	TCHouse-X 实例 ID。	若已在配置文件中设置，可省略。	`--instanceId warehouse-p******m`

请求示例

```
./spark-submit --kill \
    --taskId batch-task-b***g
```

返回参数说明

参数名称	描述	示例
`RequestId`	取消运行命令的 API 请求 ID。	`RequestId 6**********b`
`ErrorMsg`	错误信息或操作结果。	• 取消成功： `ErrorMsg` 字段将为空，表示任务已成功取消（任务状态变为 `canceled` ）。 • 取消失败： 仅当 Spark 任务处于 `initializing` 或 `running` 状态时才能被取消。如果任务已结束（例如 `success` 或 `failed` ），操作将失败，并在 `ErrorMsg` 中提示失败原因。

上传文件

要将您的本地文件上传至 腾讯云对象存储（COS），请首先进入 `<tchouse-x-spark-submit>` 目录，然后使用以下格式运行 `spark-submit --upload` 命令。

```
./bin/spark-submit --upload \
    [--confPath <confPath>] \
    [--bucket-name  <bucket-name>] \
    --key <cosPath/fileName.fileFormat> \
    --localFilePath <localPath/fileName.fileFormat>
```

请求参数说明

参数名称	是否必填	描述	参数值说明	示例
`--upload`	是	指定运行上传操作。	无参数值。	`--upload`
`--confPath <path>`	否	指定配置文件的路径。 默认值：`./conf/spark-defaults.conf`	支持相对路径或绝对路径。	`--confPath ./conf/custom-config.conf`
`--bucket <name>`	条件必填	存储上传文件的 COS 存储桶名称。 说明：若配置文件中已配置 cosUploadBucket 参数且您希望上传到该桶，则可省略此参数。	COS 存储桶名称。无需指定地域。	`--bucket my-data-bucket-12345678`
`--key <cosPath>`	是	文件在 COS 存储桶中的目标路径。 注意：如果目标文件夹不存在，您的 secretId 必须具有创建 COS 文件夹的权限，否则上传将失败。	完整路径，包括文件夹、文件名和文件格式。	`--key project/jars/mySparkJob.jar`
`--localFilePath <path>`	是	本地待上传文件的完整路径。	本地文件路径，支持相对路径或绝对路径。	`--localFilePath /tmp/data/localFile.csv`

请求示例

```
./bin/spark-submit --upload \
    --bucket my-bucket-12345678 \
    --key folder1/subFolder1/myCosSparkJar.jar \
```

```
    --localFilePath localSubFoler1/myLocalSparkJar.jar
```

返回示例

```
Transfer Start    [ConsumedBytes/TotalBytes: 0/26722018]
Transfer Data     [ConsumedBytes/TotalBytes: 26722018/26722018, 100%]
Transfer Complete [ConsumedBytes/TotalBytes: 26722018/26722018]
upload file success, cos path: cosn://my-bucket-12345678/--key
folder1/subFolder1/myCosSparkJar.jar
```

TCHouse-X 与社区版 Spark-Submit CLI 工具对比

配置参数对比

参数名称	参数说明 (社区版定义)	社区版是否支持	TCHouse-X 是否支持	TCHouse-X 定制化说明
`spark.driver.cores`	Driver 进程使用的核心数（仅限 cluster 模式）。	是 (默认值 1)	否	参数名称已调整为 `spark.driver.resourceSpec`，值必须是规格名称（如 `x-small`）。资源配比固定为 CPU:内存 ≈ 1:4。
`spark.driver.memory`	Driver 进程使用的内存量（带单位，如 `512m`，`2g`）。	是 (默认值 1g)	否	
`spark.executor.cores`	每个 Executor 使用的核心数。	是 (默认值 1)	否	参数名称已调整为 `spark.executor.resourceSpec`，值必须是规格名称（如 `small`）。资源配比固定为 CPU:内存 ≈ 1:4。
`spark.executor.memory`	每个 Executor 进程使用的内存量（带单位，如 `512m`，`2g`）。	是 (默认值 1g)	否	
`spark.executo`	运行的 Executor 数量。	是	是	TCHouse-X 完全兼容此参数，用于配置 Executor 实例数。

r.instances				
其他 spark. 开头的配置参数	其他配置参数（如超时、分区等）。	是	是	TCHouse-X 在服务端配置了默认值。如需针对特定作业任务调整，可在配置文件或提交命令中通过 --conf 指定。

提交参数对比

参数名称	参数说明 (社区版定义)	社区版支持	TCHouse-X 支持	TCHouse-X 定制化说明
--deploy-mode	指定 Spark 应用的部署模式（client 或 cluster）。	是	否	TCHouse-X 采用统一的 cluster 模式运行，无需用户显式指定。
--master	指定 Spark 集群的 URL 或集群管理器类型（yarn，mesos，spark://，k8s，local）。	是	否	TCHouse-X 引擎环境固定，通过配置文件中的 instanceId 连接，无需此参数。
--driver-cores	Driver 进程使用的核心数。	是	是 (兼容)	TCHouse-X 兼容此参数，用于配置 Driver 核心数。但推荐使用 spark.driver.resourceSpec 统一配置。
--driver-memory	Driver 进程使用的内存量（带单位，如 2g）。	是	否	资源规格统一化。请使用 spark.driver.resourceSpec 间接控制内存，资源配比固定为 1:4。
--executor-cores	每个 Executor 使用的核心数。	是	是 (兼容)	TCHouse-X 兼容此参数，用于配置 Executor 核心数。但推荐使用 spark.executor.resourceSpec 统一配置。
--executor-memory	每个 Executor 进程使用的内存量。	是	否	资源规格统一化。请使用 spark.executor.resourceSpec 间接控制内存，资源配比固定为 1:4。
--num-ex	运行的 Executor 实例数量。	是	是	完全兼容，或使用别名 spark.execu

ecutors				tor.instances 。
--total-executor-cores	所有 Executor 总共使用的核心数限制（Standalone 模式常用）。	是	否	TCHouse-X 集群资源由 num-executors 和 executor-cores 确定，不支持此总数限制。
--verbose	显示详细信息，将配置写入日志文件。	是	否	TCHouse-X 的 CLI 暂不支持此参数。作业日志请通过 --get-log 命令查看。
--conf <key>=<value> / --config	指定任意 Spark 配置属性（key=value 格式）。	是	是	完全兼容，用于覆盖 spark-defaults.conf 中的默认值或设置其他 Spark 参数。
--files	指定供 Spark 应用使用的附加文件（逗号分隔）。	是	是	完全兼容。文件必须先上传到 COS 并使用 cosn:// 路径。
--packages	指定 Maven 坐标，用于自动处理依赖包及其传递性依赖。	是	否	TCHouse-X 提交时不支持 Maven 依赖解析。请将所有依赖 JAR 包手动上传至 COS，并使用 --jars 参数指定。

数据入仓
对象存储 COS

最近更新时间：2026-05-06 16:28:12

简介

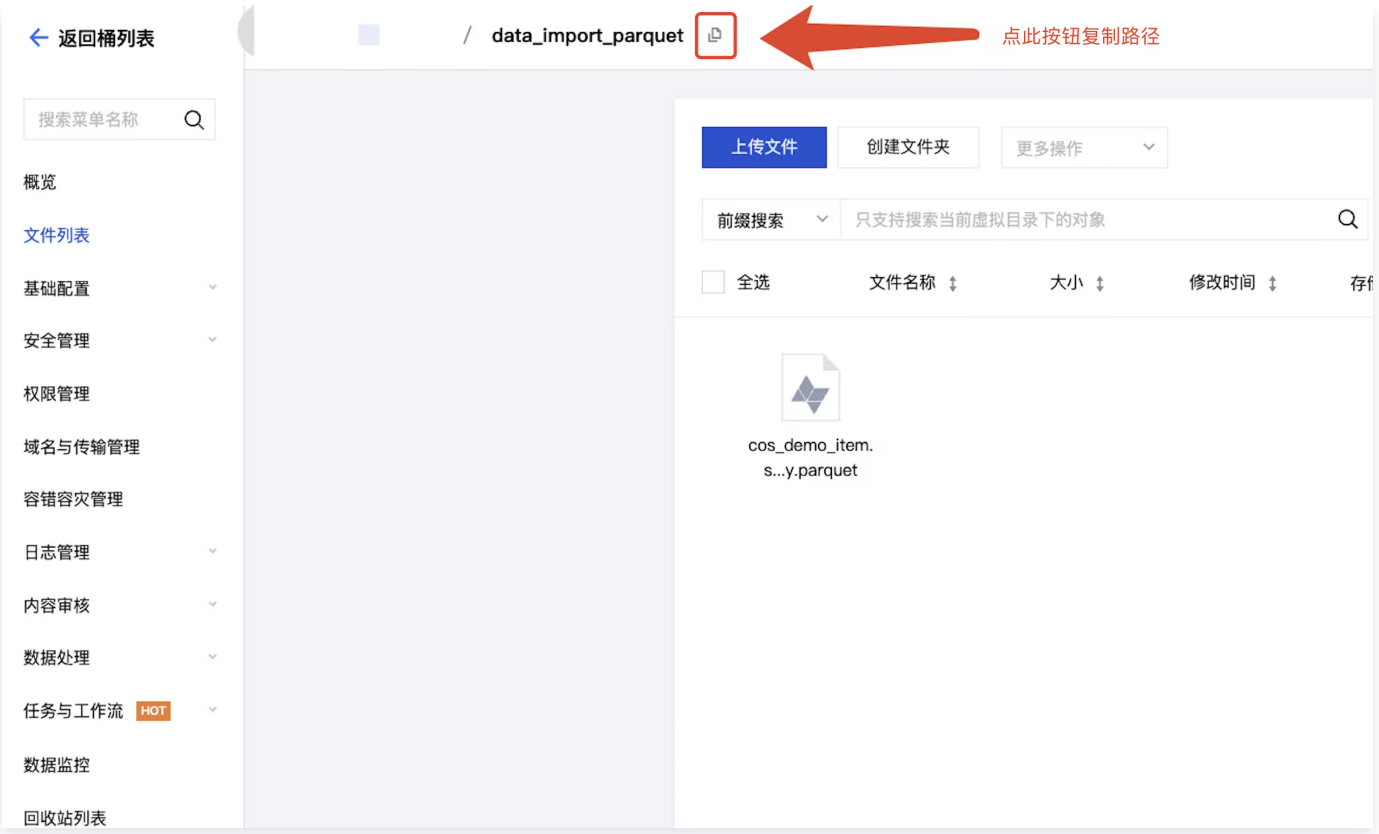
对于已存储于 COS 的 Parquet、CSV 等文件，用户可以在 TCHouse-X 中创建外表直接分析使用。如需提升查询性能，用户还可将外表导入 TCHouse-X 内表。

前置条件

1. 创建 TCHouse-X 实例，操作指南请参见 实例创建与销毁 。
2. 创建对象存储 COS 桶、上传文件、获取文件 cosn 路径，操作指南可参阅：

- 创建存储桶：
  - 控制台创建
  - API 创建
- 上传文件：
  - 控制台上传对象
  - SDK/API-PUT Objects
- 获取文件的 cosn 路径：

> **说明：**
> cosn 路径为 'cosn://' 前缀 + 复制的路径名。



- COS 工具概览：对象存储 > 工具概览

场景1：仅创建外表，不导入数据至 TCHouse-X

前置条件：已完成前置准备。

基于 Parquet、TextFile（CSV/TXT）文件创建外表

可参阅以下建表语法创建外表：

```
CREATE EXTERNAL TABLE [IF NOT EXISTS] [db_name.]table_name
  (col_name data_type
    [COMMENT 'col_comment']
    [, ...]
  )
  [PARTITIONED BY (col_name data_type [COMMENT 'col_comment'], ...)]
  [SORT BY ([column [, column ...]])]
  [COMMENT 'table_comment']
  [ROW FORMAT row_format]
  [STORED AS file_format]
  LOCATION 'cosn://bucket-name/path';

Support DataType:
```

```
    TINYINT
  | SMALLINT
  | INT
  | BIGINT
  | BOOLEAN
  | FLOAT
  | DOUBLE
  | DECIMAL
  | STRING
  | CHAR
  | VARCHAR
  | TIMESTAMP

Support DataType for HASH:
    SMALLINT
  | INT
  | BIGINT
  | DATE
  | CHAR
  | VARCHAR
  | STRING

Support DataType for VALUE:
    BOOLEAN
  | TINYINT
  | SMALLINT
  | INT
  | BIGINT
  | FLOAT
  | DOUBLE
  | DECIMAL
  | CHAR
  | VARCHAR
  | STRING
  | DATE
```

如操作时无可用数据，可下载 demo parquet 文件后，上传至 COS。该 demo parquet 文件的建表语句如下:

```
CREATE EXTERNAL TABLE default.external_item (
```

```
  i_item_sk INT,
  i_item_id STRING,
  i_rec_start_date DATE,
  i_rec_end_date DATE,
  i_item_desc STRING,
  i_current_price DECIMAL(7,2),
  i_wholesale_cost DECIMAL(7,2),
  i_brand_id INT,
  i_brand STRING,
  i_class_id INT,
  i_class STRING,
  i_category_id INT,
  i_category STRING,
  i_manufact_id INT,
  i_manufact STRING,
  i_size STRING,
  i_formulation STRING,
  i_color STRING,
  i_units STRING,
  i_container STRING,
  i_manager_id INT,
  i_product_name STRING
)
STORED AS PARQUET
LOCATION 'cosn://your-bucket-name/your-file-path';
```

场景2：创建外表并导入数据至 TCHouse-X 内表

前置条件：已完成前置准备。

Step 1：创建外表

操作步骤与场景1相同，详情请参见 基于 Parquet、TextFile（CSV/TXT）文件创建外表 。

Step 2：创建内表

如操作时无可用数据，可下载 demo parquet 文件、上传至 COS、创建外表。
若使用该 demo parquet 文件, 则创建内表的建表语句如下:

```
CREATE TABLE default.inner_item (
  i_item_sk INT,
```

```
  i_item_id STRING,
  i_rec_start_date DATE,
  i_rec_end_date DATE,
  i_item_desc STRING,
  i_current_price DECIMAL(7,2),
  i_wholesale_cost DECIMAL(7,2),
  i_brand_id INT,
  i_brand STRING,
  i_class_id INT,
  i_class STRING,
  i_category_id INT,
  i_category STRING,
  i_manufact_id INT,
  i_manufact STRING,
  i_size STRING,
  i_formulation STRING,
  i_color STRING,
  i_units STRING,
  i_container STRING,
  i_manager_id INT,
  i_product_name STRING
);
```

Step 3：写入数据

使用 `INSERT INTO ... SELECT ...` 方式将数据从外表-自管表查询出来写入内表，推荐在 SQL 前添加/*+engine=batch*/调用离线引擎执行数据写入。
如在 Step 1和 Step 2中使用 demo parquet 文件构建了外表和内表，则写入语句为：

```
/*+engine=batch*/
insert into inner_item select * from external_item;
```

说明：

- `/*+engine=batch*/` **是 TCHouse-X 的优化 Hint，用于调用离线计算引擎，以处理大规模数据的批量导入，推荐在数据迁移和初始化同步时使用。默认资源配置如下：**
  - **Executor 节点规格： 2X-Small (4CU)**
  - **Executor 节点数量： 1 个**
  - **Driver 节点规格： 2X-Small (4CU)**

- Driver 节点数量：1 个

- 如果您需要进一步配置离线计算引擎的资源，可使用 Hint 组合 `/*+engine=batch, executor_specs=2x-small, executor_count=1, driver_specs=2x-small*/` 进行配置，其中 `executor_specs` 、 `driver_specs` 可选规格有：
  - 2X-Small (4CU)
  - X-Small (8CU)
  - Small (16CU)
  - Medium (32CU)
  - Large (64CU)
- 您可以替换 `SELECT *` 为特定的列名，或添加 `WHERE` 子句以实现增量或条件导入。

Iceberg

最近更新时间：2026-05-06 16:28:12

简介

对于已存储于 对象存储（COS）上的 Iceberg 表，您可以使用 `CREATE EXTERNAL TABLE` 命令将其元数据信息添加到 TCHouse-X 服务。TCHouse-X 实例即可对该表进行读操作。

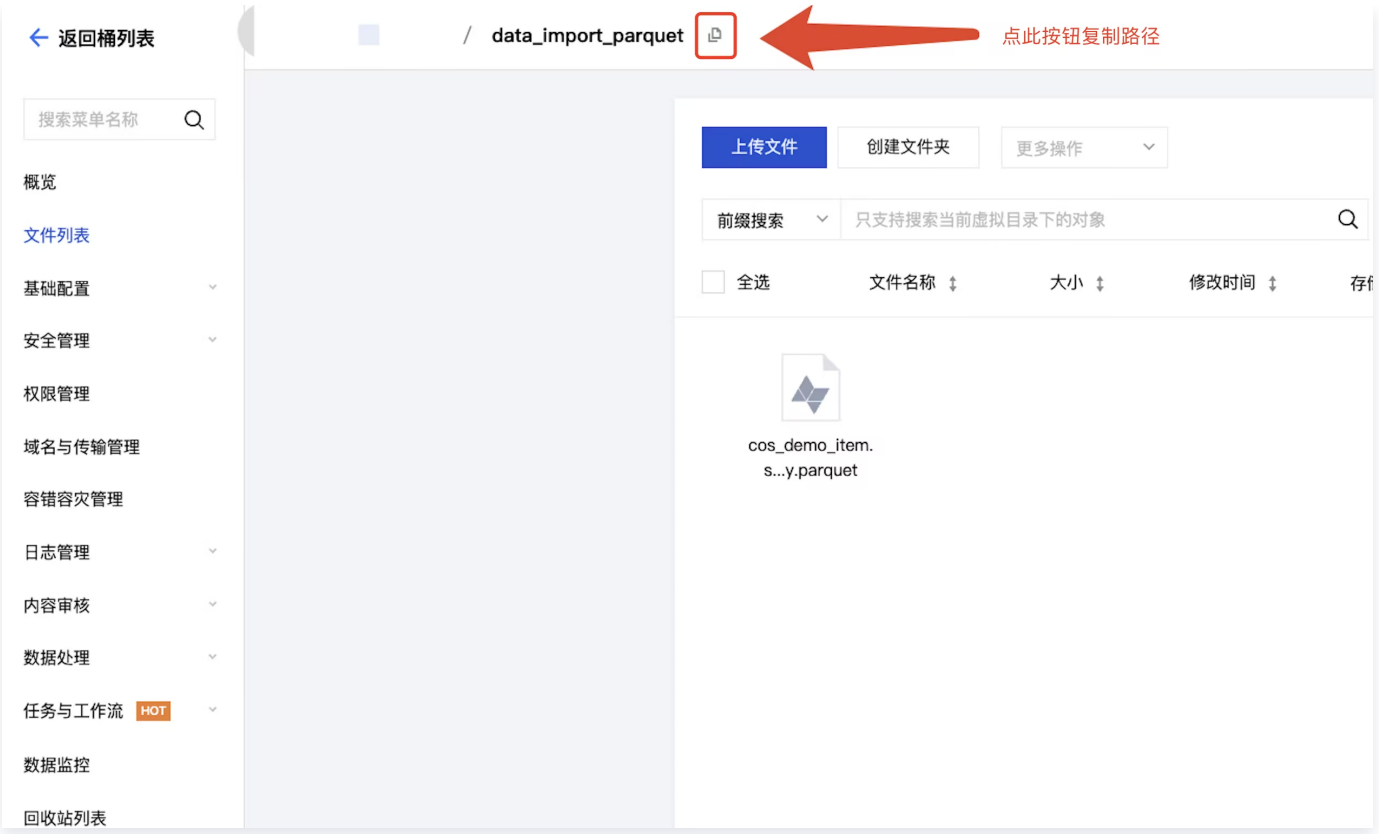
前置准备

根据您的数据存储位置，请完成以下相应准备工作：

存储于 COS 上的 Iceberg 表

1. 创建 TCHouse-X 实例，操作指南请参见 实例创建与销毁 。
2. 准备对象存储资源：

- 创建存储桶
  - 控制台创建
  - API 创建
- 上传文件
  - 控制台上传对象
  - SDK/API-PUT Objects
- 获取文件的 cosn 路径：cosn 路径为 'cosn://' 前缀 + 复制的路径名。



- 相关工具概览：对象存储 > 工具概览

场景 1：仅创建外表，不导入数据至 TCHouse-X

本场景适用于将已有的 Iceberg 表元数据映射到 TCHouse-X，实现对外表的查询分析。

创建 Iceberg 外表

参考语法如下：

```
CREATE EXTERNAL TABLE [IF NOT EXISTS] [db_name.]table_name
  [COMMENT 'table_comment']
  STORED AS ICEBERG
  [LOCATION 'table_path']
  [TBLPROPERTIES ('key1'='value1', 'key2'='value2', ...)]
```

场景 2：创建外表，并导入数据至 TCHouse-X 内表

本场景适用于将 Iceberg 外部存储系统中的数据迁移或同步到 TCHouse-X 的内部管理表，以实现数据的高效存储和查询优化。

Step 1：创建 Iceberg 外表

操作详情与“场景 1：仅创建外表，不导入数据至 TCHouse-X”完全相同。

Step 2：创建 TCHouse-X 内表

在执行数据导入之前，您需要创建用于存储数据的 TCHouse-X 内表。内表的 Schema 应与 Iceberg 外表的 Schema 保持一致。

Step 3：写入数据

使用 `INSERT INTO...SELECT` 语句，将数据从已创建的 Iceberg 外表查询出来，并写入 TCHouse-X 的内表。为了确保数据导入操作的性能和稳定性，强烈推荐使用离线（Batch）引擎执行数据写入任务。

```
/*+engine=batch*/
insert into inner_item select * from external_item;
```

说明：

- `/*+engine=batch*/` 是 TCHouse-X 的优化 Hint，用于调用离线计算引擎，以处理大规模数据的批量导入，推荐在数据迁移和初始化同步时使用。默认资源配置如下：
  - Executor 节点规格： 2X-Small (4CU)
  - Executor 节点数量： 1 个
  - Driver 节点规格： 2X-Small (4CU)
  - Driver 节点数量： 1 个
- 如果您需要进一步配置离线计算引擎的资源，可使用 Hint 组合 `/*+engine=batch, executor_specs=2x-small, executor_count=1, driver_specs=2x-small*/` 进行配置，其中 `executor_specs` 、`driver_specs` 可选规格有：
  - 2X-Small (4CU)
  - X-Small (8CU)
  - Small (16CU)
  - Medium (32CU)
  - Large (64CU)
- 您可以替换 `SELECT *` 为特定的列名，或添加 `WHERE` 子句以实现增量或条件导入。

数据类型映射

Iceberg 与 TCHouse-X 数据类型略有差异，其映射关系如下：

Iceberg DATA Type	DATA Type in TCHouse-X
`BOOLEAN`	`BOOLEAN`
`INT`	`INTEGER`

`LONG`	`BIGINT`
`FLOAT`	`FLOAT`
`DOUBLE`	`DOUBLE`
`DECIMAL(P, S)`	`DECIMAL(P, S)`
`DATE`	`DATE`
`TIME`	Not supported
`TIMESTAMP`	`TIMESTAMPNTZ`
`TIMESTAMPTZ`	`TIMESTAMPLTZ` / `TIMESTAMP`
`STRING`	`STRING`
`UUID`	Not supported
`FIXED(L)`	Not supported
`BINARY`	Not supported
`STRUCT`	`STRUCT` (read only)
`LIST`	`ARRAY` (read only)
`MAP`	`MAP` (read only)

数据作业
作业列表

最近更新时间：2026-05-06 16:28:12

简介

TCHouse-X 提供了基于 Spark 的离线数据处理能力，支持用户通过数据任务进行复杂的数据处理、ETL 等操作。TCHouse-X 离线引擎完全兼容 Spark 3.5.3 内核，Spark 作业的开发可参考 开发指南 。

使用指引

前置准备

在开始创建数据作业前，您需先创建一个 TCHouse-X 实例，并进入实例空间创建数据作业。实例创建请参见 实例创建与销毁 。

创建数据作业

1. 登录 TCHouse-X 控制台 ，进入实例，单击左侧菜单**数据作业 > 作业列表**进入数据作业管理页。
2. 单击**新建作业**按钮，进入创建页。
3. 配置作业内容, 配置参数如下：

配置参数	说明
作业名称	支持中文、英文、数字和“_”，最多100个字符
Spark 版本	当前仅支持 Spark 3.5.3 版本
程序包	支持 jar、py 格式文件，仅可配置一种类型 支持选择对象存储 cos 内文件或本地上传，本地上传先选择 cos 存储桶后，可进行文件上传操作
程序入口参数	非必填。程序的入口参数，支持填写多个。多个参数使用“空格”分割，多个 SQL 请使用 -sql 换行填写，不超过65535 个字符。
作业参数（--config）	非必填。作业--config 信息，spark.开头的参数，按照 k=v 格式填写，多个参数换行填写，不超过 65535 个字符。 示例：spark.network.timeout=120s
依赖 jar 资源（--jar）	非必填。仅支持 jar 格式，可选择多个 支持选择对象存储 cos 内文件或本地上传，本地上传先选择 cos 存储桶后，可进行文件上传操作。

依赖 py 资源（--py-files）	非必填。支持 py、zip、egg 格式，可选择多个 支持选择对象存储 cos 内文件或本地上传，本地上传先选择 cos 存储桶后，可进行文件上传操作。
依赖 files 资源（--files）	非必填。暂不支持 jar、zip 格式，可选择多个 支持选择对象存储 cos 内文件或本地上传，本地上传先选择 cos 存储桶后，可进行文件上传操作。
依赖 archives 资源（--archives）	非必填。支持 tar.gz、tgz、tar 格式，可选择多个 支持选择对象存储 cos 内文件或本地上传，本地上传先选择 cos 存储桶后，可进行文件上传操作。
资源配置	配置数据作业的引擎资源，离线数据作业使用 Serverless 弹性资源，作业启动时资源开始运行，作业停止后资源自动销毁，无需您手动管理资源。 资源说明：1CU≈1核4G 计费 CU 数 = Executor 资源 * Executor 数量 + Driver 资源 计算费用将按计算 CU 数的使用量*使用时间收取
Executor 规格	Executor 规格配置可选规格：2X-Small (4CU)、X-Small (8CU)、Small (16CU)、Medium (32CU)。注：1CU ≈ 1vCPU + 4GB RAM
Executor 个数	Executor 数量配置支持固定分配与动态分配两种模式。在动态分配模式下，仅需设置 Executor 的最小数量与最大数量，系统将根据作业负载自动调整资源。
Driver 规格	Driver 规格配置可选规格：2X-Small (4CU)、X-Small (8CU)、Small (16CU)、Medium (32CU)。注：1CU ≈ 1vCPU + 4GB RAM

4. 完成配置填写后，单击**创建并启动**可以直接运行任务。单击**仅创建**保存任务配置。

管理数据作业

作业列表展示了您已创建的所有数据作业，您可以通过操作栏对作业进行以下管理：

- **编辑：**修改作业的配置信息。
- **启动控制：**手动启动作业任务。
- **删除：**移除不再需要的作业实例。

编辑数据作业

已创建的数据作业支持编辑操作，您能够修改作业相关配置。

1. 进入管理页：登录 TCHouse-X 控制台 并进入实例，在左侧导航栏依次选择 **数据作业** > **作业列表**。
2. 在**作业列表**找到需要编辑的作业，点击**编辑**按钮进入作业编辑页面。
3. 编辑完成后支持**保存并启动**或者**仅保存**。

> ⚠ **注意：**

> 数据作业存在运行中的数据任务时，无法编辑。

启动数据作业

您可以通过启动或停止操作来控制作业的运行。每次启动作业都会触发生成一个独立的数据任务。
操作步骤：

1. 执行启动：在“作业列表”中定位目标作业，单击操作列的 **启动** 按钮。
2. 追踪任务：启动后，系统将生成对应的任务实例。您可以在作业名称左侧的展开列表或作业运行记录中查看任务进度。

查看作业详情

1. 进入管理页：登录 TCHouse-X 控制台 并进入实例，在左侧导航栏依次选择 **数据作业** > **作业列表**。
2. 查看详情：在作业列表中，点击目标 作业名称 即可进入作业详情页。

删除数据作业

1. 进入管理页：登录 TCHouse-X 控制台 并进入实例，在左侧导航栏依次选择 **数据作业** > **作业列表**。
2. 在作业列表找到需要删除的作业，点击**删除**按钮。
3. 二次确认后即可完成删除。

> ⚠ **注意：**
>
> - 数据作业存在运行中的数据任务时，将无法被删除。
> - 删除数据作业将同时删除对应的数据任务信息，请谨慎操作。

任务列表

最近更新时间：2026-05-06 16:28:12

简介

任务列表管理集中监控并管理由各类数据作业生成的任务实例，提升运维效率：

- 详情追踪：深度解析任务执行参数，实时监控运行状态与进度。
- 灵活调度：支持手动干预，一键终止异常执行或冗余排队的任务。

任务状态

数据任务状态如下：

状态	说明
初始化	任务已启动，正在进行环境准备与资源预分配。
执行中	任务正在运行。注意： 此阶段对应的数据作业将被锁定，不支持编辑或删除。
成功	任务已顺利完成所有计算逻辑并成功退出。
失败	任务运行中断。建议通过 运行日志 或 Spark UI 定位具体报错信息。
已取消	任务已被用户手动终止。
结果处理中	任务计算已完成，正在进行数据写回、元数据更新或资源回收。

查看详情

1. 登录 TCHouse-X 控制台，进入实例，单击左侧菜单**数据作业 > 任务列表** 进入任务管理页。
2. 单击**任务名称**或者**查看详情**即可进入数据任务详情页。任务详情页支持查看如下信息：

任务信息	说明
基本信息	展示任务 ID、当前状态及运行耗时等核心指标。
资源详情	实时监控任务运行状态，包括计费资源用量及关联 Pod 的详细参数。
运行日志	• 在线查看：支持按 Pod 维度检索日志；界面默认展示最新的 1,000 条数据。 • 日志下载：支持完整日志下载，系统将同步生成下载任务记录。
日志下载记录	记录日志提取历史。包含 Pod 信息、提交时间及下载状态；完成后可直接点击“保存到本地”。

Spark UI

系统支持全生命周期的 Spark UI 访问：无论任务处于执行中还是已完成，您均可点击进入 Web UI 页面，获取详细的执行计划与监控指标。

取消运行

系统支持对处于“初始化”或“执行中”状态的数据任务执行取消操作。任务终止后，系统将状态统一更新为“已取消”。

PySpark 依赖包管理

最近更新时间：2026-05-06 16:28:12

简介

运行环境 TCHouse-X PySpark 默认基础环境为 Python 3.9.2。
依赖提交方式您可以通过以下两种方式配置作业依赖，且两者可根据需求并行使用：

1. 使用 `--py-files` 指定依赖：若您的依赖项为纯 Python 实现的模块或脚本，建议优先使用此方式，直接上传相关文件即可。
2. 使用 `--archives` 挂载虚拟环境：若您的项目依赖较复杂，或包含需要编译安装的 C 相关依赖（如深度学习库、科学计算库），推荐将整个开发测试环境打包，并通过 `--archives` 方式分发，以保证运行环境的完整性。

使用 `--py-files` 依赖包

该方式适用于纯 Python 实现的模块或文件，其中未包含任何 C 依赖。

步骤一：打包模块/文件

PyPI 外部包，需要在本地环境中，使用 pip 指令安装并打包常用依赖，要求依赖包使用纯 Python 实现，不依赖 C 相关库。

```
pip install -i https://mirrors.tencent.com/pypi/simple/ <packages...>  -
t dep

cd dep

zip -r ../dep.zip .
```

单文件模块（functions.py）和自定义 Python 模块都可以通过上述方法打包，需要注意的是自定义 Python 模块需要按照 Python 官方要求标准化，详情可参考 Python 官方文档 Python Packaging User Guide 。

步骤二：引入打包好的模块

在 TCHouse-X 控制台 的数据作业模块中新建作业。在 `--py-files` 参数中引入打包好的 dep.zip 文件，该文件可以通过上传到 COS 或者本地上传的方式引入。

使用虚拟环境

使用虚拟环境可以有效解决 Python 依赖包（尤其是涉及 C 扩展的包，如 `numpy`，`pandas`，`jieba` 等）在分布式集群中的兼容性问题。通过将依赖预编译并打包上传，可以确保作业运行环境的一致性。

前置要求：打包环境建议

由于部分 Python 依赖涉及 C 语言编译，为保证与计算节点环境兼容，强烈建议在以下环境中进行打包：

- **架构：**x86_64
- **操作系统：**Debian 11 (bullseye)
- `Python` **版本**：3.9.2

步骤一：打包虚拟环境

我们提供了三种打包方式，您可以根据习惯选择其一。

方法 1：使用 Venv 打包

适用于依赖较少、环境简单的场景。

```
# 1. 创建虚拟环境
python3 -m venv pyspark_venv

# 2. 激活并安装依赖（建议使用腾讯云镜像源加速）
source pyspark_venv/bin/activate
pip3 install -i https://mirrors.tencent.com/pypi/simple/ <依赖包名>

# 3. 退出并压缩
# 注意：必须使用 tar 命令打包，确保链接文件不丢失
deactivate
tar -czvf pyspark_venv.tar.gz pyspark_venv/
```

方法 2：使用 Conda 打包

适用于复杂依赖管理，建议配合 `conda-pack` 使用以确保环境完整迁移。

```
# 1. 创建环境并安装 conda-pack
conda create -y -n pyspark_env python=3.9.2 conda-pack <其他包名>

# 2. 激活环境
conda activate pyspark_env

# 3. 使用 conda-pack 导出环境
conda pack -f -o pyspark_env.tar.gz
```

方法 3：使用 Docker 自动化脚本打包

如果您本地有 Docker 环境（支持 Linux/Mac），可以使用我们提供的自动化脚本。该脚本会自动在标准 Debian 镜像中完成编译。
脚本参数说明：

参数	描述	备注
-r	指定 requirements.txt 路径	必填
-n	指定虚拟环境及输出压缩包的名称	默认：py3env
-o	指定保存本地的目录	默认：当前目录
-h	打印帮助信息	-

步骤二：上传并配置虚拟环境

Step1: 上传资源

上传资源：将打包好的 .tar.gz 文件（如 `py3env.tar.gz` ）上传至您的 COS（对象存储） 中。

Step2：配置 `--archives` 参数（环境挂载）

在作业配置界面的 `--archives` 参数项中，填入虚拟环境包的完整资源路径。

- 格式： `[资源路径]#[解压文件夹名称]`
- 示例： `cosn://mybucket/envs/python_env.tar.gz#venv`

说明：
符号 `#` 用于指定压缩包在计算节点上的解压目录名称。该名称（如上例中的 `venv`）将直接影响后续运行环境参数的选择，请务必保持一致。

Step3：配置 `--config` 参数（指定解释器）

为了让 Spark 任务识别并调用虚拟环境，需通过 `--config` 明确 Python 解释器的相对路径。请根据您的打包方式选择对应的路径：

打包方式	推荐配置项与路径	说明
Venv 打包	`spark.pyspark.python=venv/pyspark_venv/bin/python3`	默认 venv 结构通常多一层目录
Conda 打包	`spark.pyspark.python=venv/bin/python3`	常用 Conda 打包后的标准路径
脚本打包	`spark.pyspark.python=venv/bin/python3`	自定义脚本封装的常用路径

venv 打包示例：

说明：

- **路径验证：** `venv` 和 `conda` 因打包工具及参数不同，内部目录层级可能存在差异。
- **手动核对：** 若任务运行报错（找不到文件），建议先在本地解压 `.tar.gz` 包，确认 python3 可执行文件相对于根目录的具体位置。
- **参数引用：** 配置项中的 `venv/` 前缀必须与第一步中 `#` 号后的名称严格对应。

SQL开发
SQL 工作区

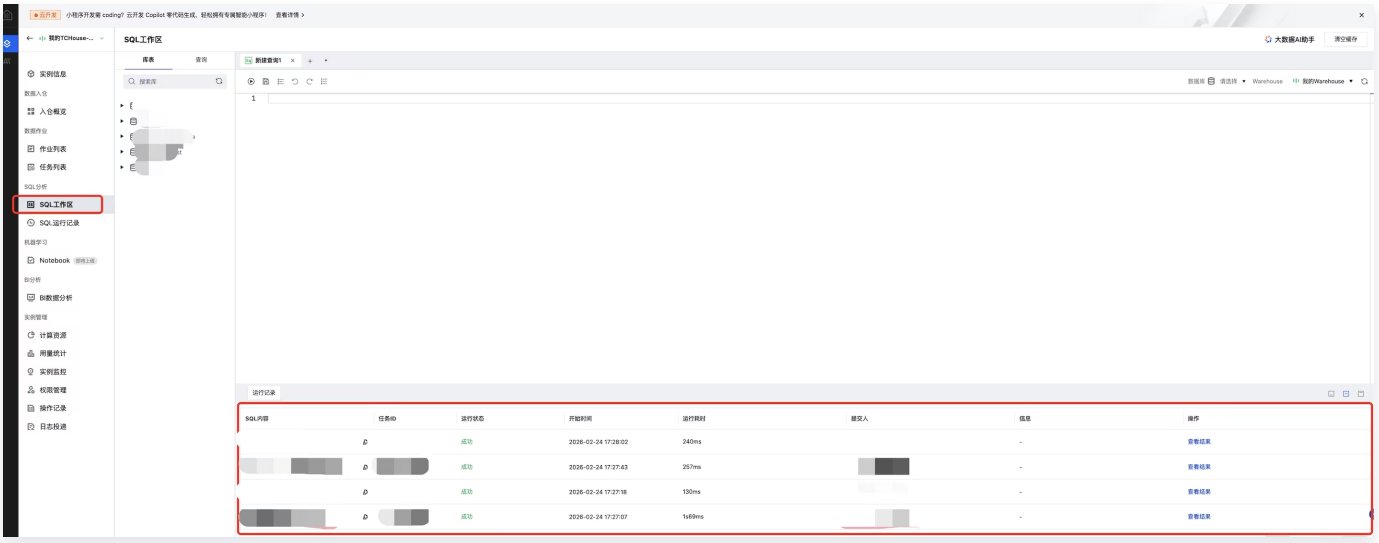
最近更新时间：2026-05-06 16:28:12

简介

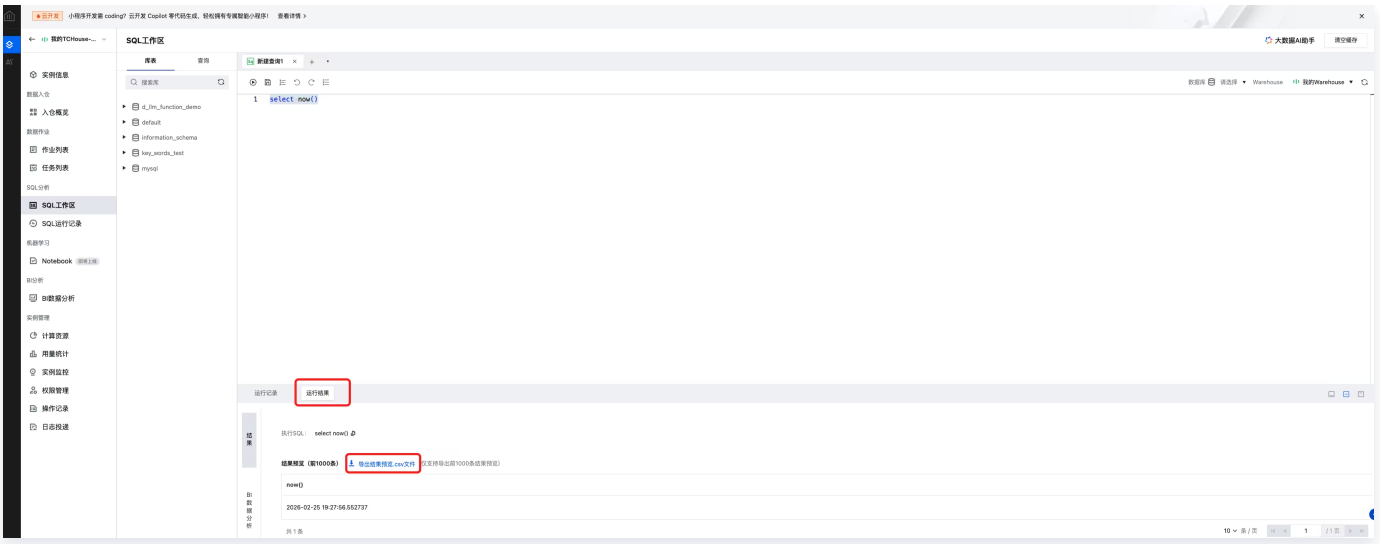
通过 SQL 工作区，您可以快捷地连接集群，使用 SQL 命令开展一系列操作。

操作指南

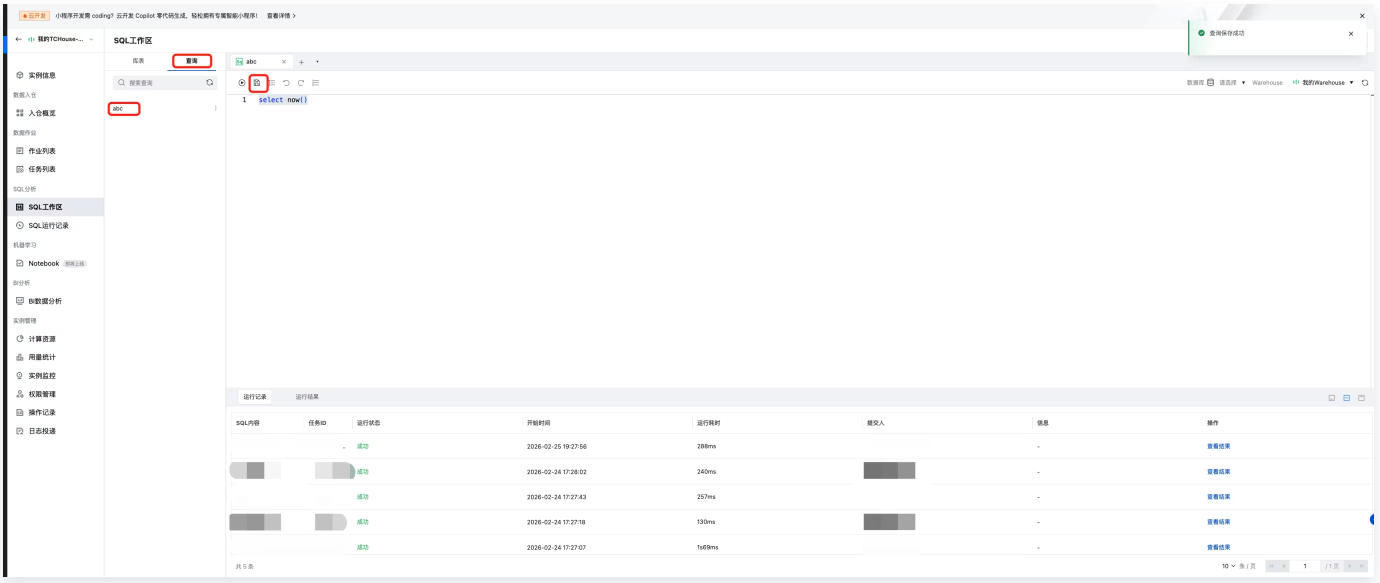
1. 登录 腾讯云数据仓库 TCHouse-X控制台 ，选择需要操作的实例，在左侧列表单击即可访问 SQL 工作区，SQL 工作区自动使用登录 TCHouse-X 的腾讯云账号。
2. 进入 SQL 工作区后，首先需要连接集群。数据操作依赖登入账号的数据权限，可在对应集群“账号管理”中进行权限授予，详情请参见 账户与权限管理 。
3. 在 SQL 工作区内，您可以查看当前登录用户在 SQL 工作区提交的 SQL 语句执行记录。



4. 在 SQL 工作区执行 SQL 后，查询结果将在新标签页中展示。若当前查询未完成，则执行状态为“运行中”。查询完成后，将自动刷新展示 SQL 查询结果。
5. SQL 工作区支持查询结果下载，点击导出结果预览即可下载包含前 1000 条结果的 CSV 格式文件。



6. 您可将在 SQL 工作区编写的 SQL 保存下来，在**查询窗口**可见，供后续运行。

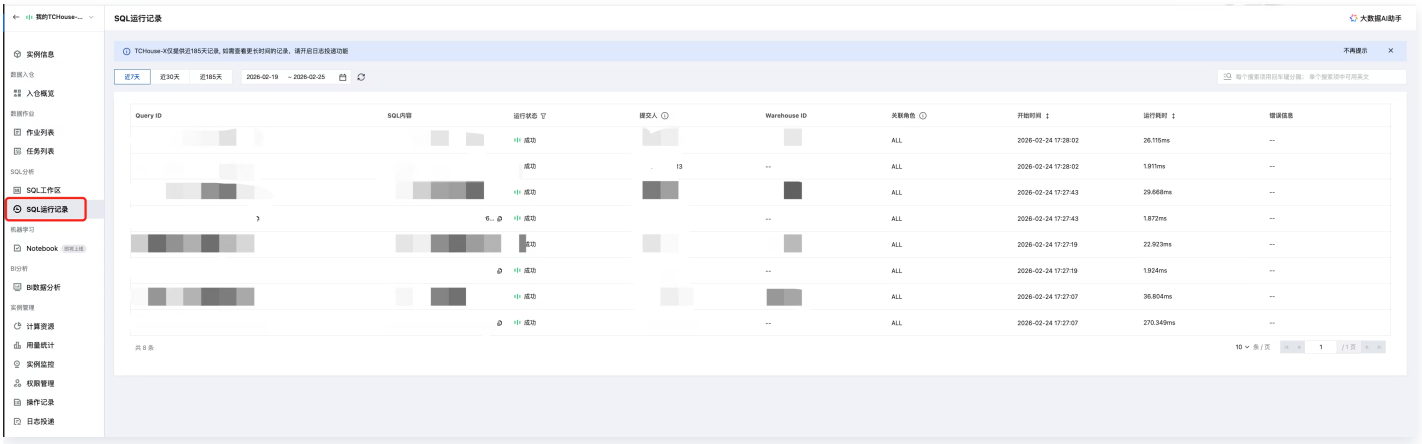


SQL 运行记录

最近更新时间：2026-05-06 16:28:12

功能介绍

帮助用户快速了解云数据仓库 TCHouse-X SQL 执行情况。通过交互式页面，简化用户操作查询原始 QueryLog 表的繁琐过程，节省查询成本，提高操作效率。



操作指南

1. 时间范围选择， 默认是近7天， 控制台提供近30天、近185天 SQL 记录的快捷查询方式。
2. 可根据运行状态做二次筛选。
3. 可根据 SQL 开始时间、运行耗时做排序。
4. 另外，TCHouse-X提供 Query ID，提交人、关联角色、任务ID 作为过滤条件查找 SQL 运行记录。

日志投递

最近更新时间：2026-05-06 16:28:12

功能简介

TCHouse-X 的日志投递功能支持将实例运行过程中的各类日志实时同步至 腾讯云日志服务 (CLS)。依托 CLS 强大的检索、分析与告警能力，助力用户构建自动化运维体系、实现故障深度排查及满足安全合规审计需求。

核心价值

- **高效运维：**通过实时监控与自动化告警，缩短故障响应时间。
- **性能优化：**深度分析作业轨迹，精准定位性能瓶颈。
- **安全合规：**完整记录操作行为，满足企业审计与追溯要求。

支持的日志类型

TCHouse-X 当前支持以下四类关键日志的自动化投递：

日志类型	应用场景	核心价值
Spark 任务记录	监控 Spark 作业的执行进度与生命周期。	用于分析作业性能瓶颈，优化资源分配。
SQL 运行日志	记录 SQL 语句的执行轨迹及错误详情。	快速定位慢查询原因，辅助 SQL 调优与排障。
Session 管理记录	追踪用户会话（Session）的建立、持续与释放。	监控集群连接状态，保障系统访问稳定性。
操作记录	记录用户对实例及资源的所有配置变更。	满足企业安全审计需求，实现行为可追溯。

操作指南

步骤 1：定位功能入口

登录 TCHouse-X 控制台，进入需要投递日志的实例，在左侧导航栏中依次选择**实例管理 > 日志投递**。

步骤 2：服务开通与授权

在正式开启投递前，请确保环境满足以下条件：

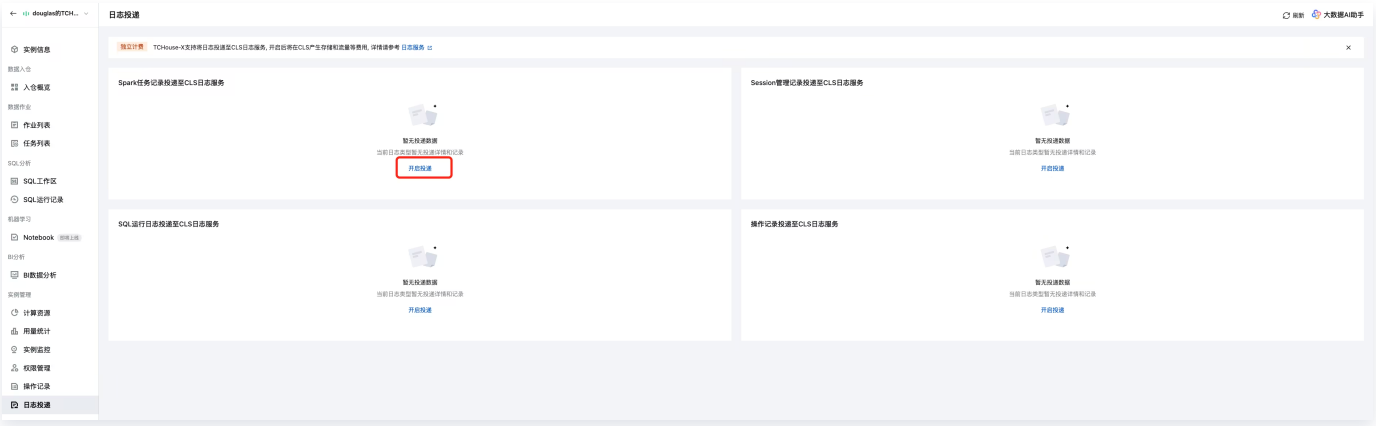
1. 开通服务：已激活腾讯云日志服务 (CLS)。
2. 角色授权：已授权 TCHouse-X 访问您的 CLS 资源。

⚠ 注意：

若尚未完成，请根据页面浮窗指引完成 CLS 服务开通与访问授权。

步骤 3：开启日志投递

1. 在日志列表页面，找到目标日志类型，单击**开启投递**，进入日志投递配置弹窗。



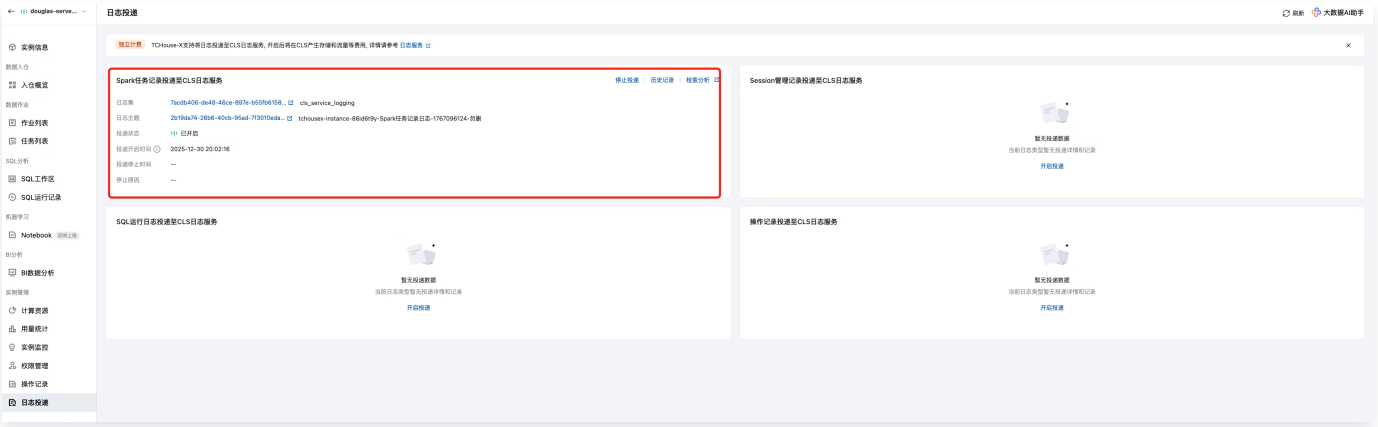
2. 在弹出的配置窗口中，选择存放日志的日志集。

⚠ 注意：

如无可用日志集，可单击**前往CLS创建**跳转至 CLS 控制台新建后再行选择。

3. 确认无误后，单击**开启投递**。

等待片刻后即可看到日志投递信息。



步骤 4：日志检索分析

参照 CLS 文档 检索分析 ，进行日志检索分析。

BI 数据分析
腾讯云 BI 连接 TCHouse-X 操作指南

最近更新时间：2026-05-06 16:28:12

本文档旨在指导您如何通过腾讯云 BI 产品对接 TCHouse-X 数据仓库，实现企业级数据的快速分析与可视化展示。

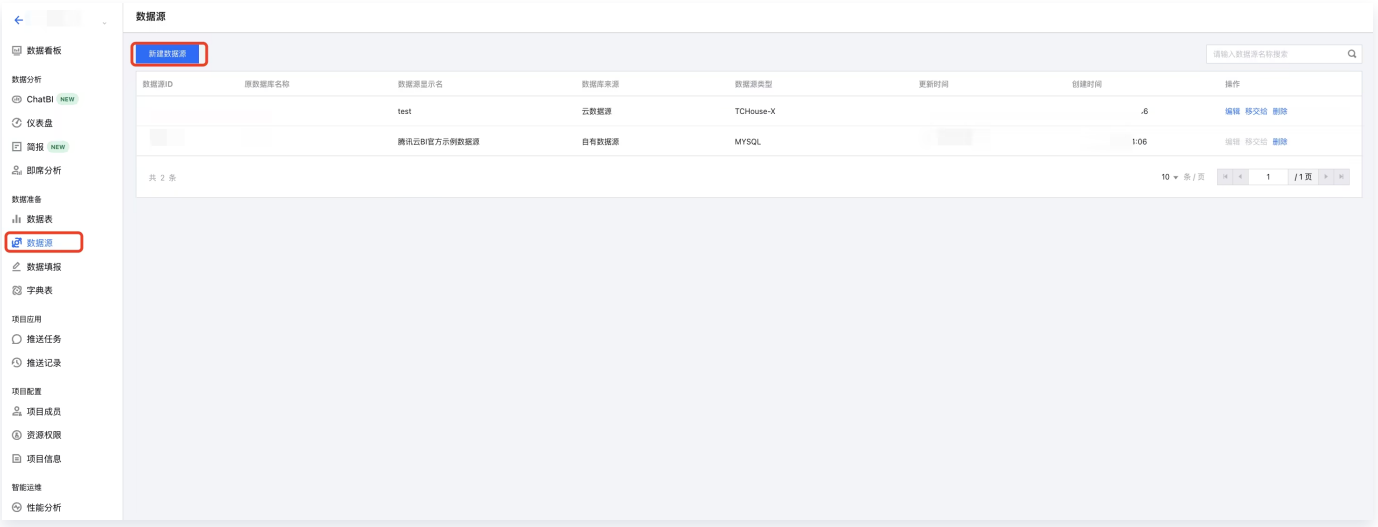
前置环境准备

- BI 产品：确保已试用开通或正式购买 腾讯云 BI 服务 。
- TCHouse-X 实例：确保 TCHouse-X 实例运行正常，并已获取具备查询权限的数据库账号。
- 网络连通性：确保 腾讯云 BI 环境与 TCHouse-X 实例所在 VPC 网络互通。

TCHouse-X 数据源配置步骤

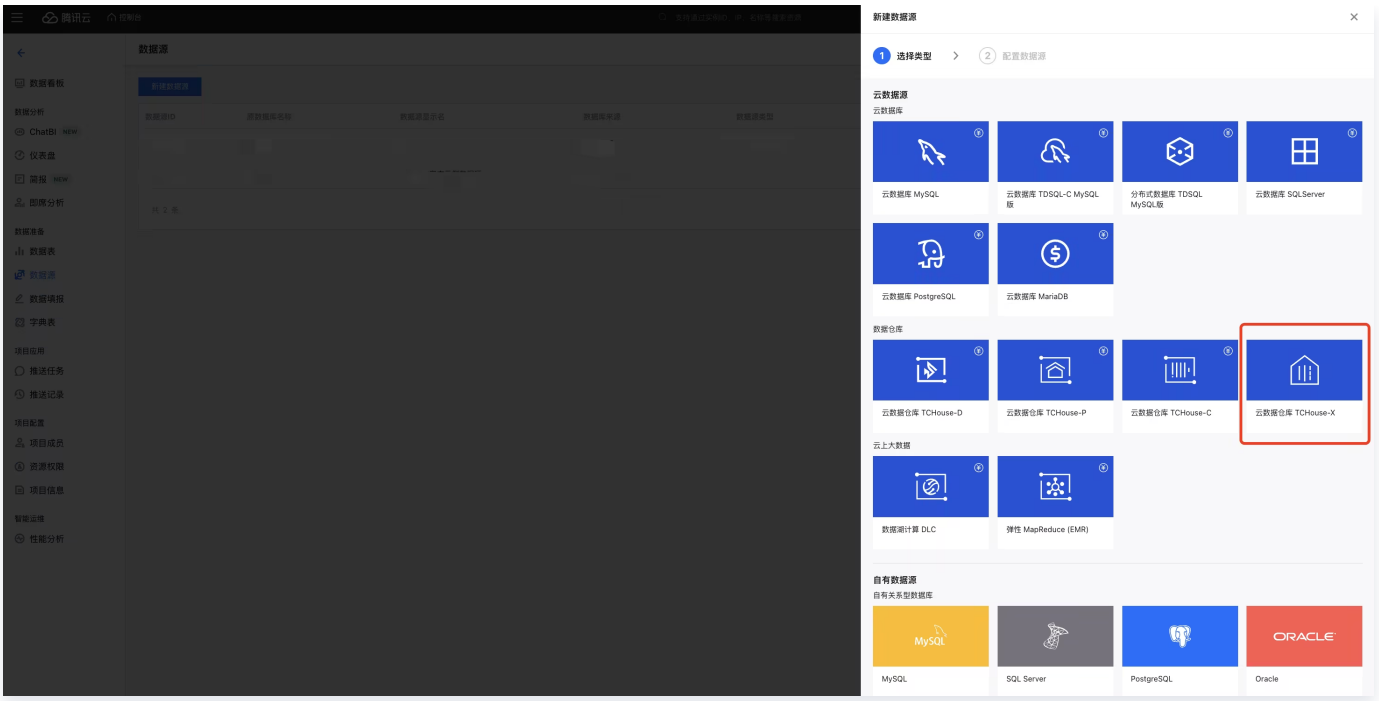
1. **进入数据源管理。**

   登录 腾讯云 BI 控制台，进入目标项目空间。在左侧导航栏点击 “**数据源**”，随后点击页面上方的 “**新建数据源**” 。



2. **选择数据源类型。**

   在弹出的“新建数据源”右侧抽屉面板中，定位至“云数据源”分类，选择 TCHouse-X。



3. 配置连接参数

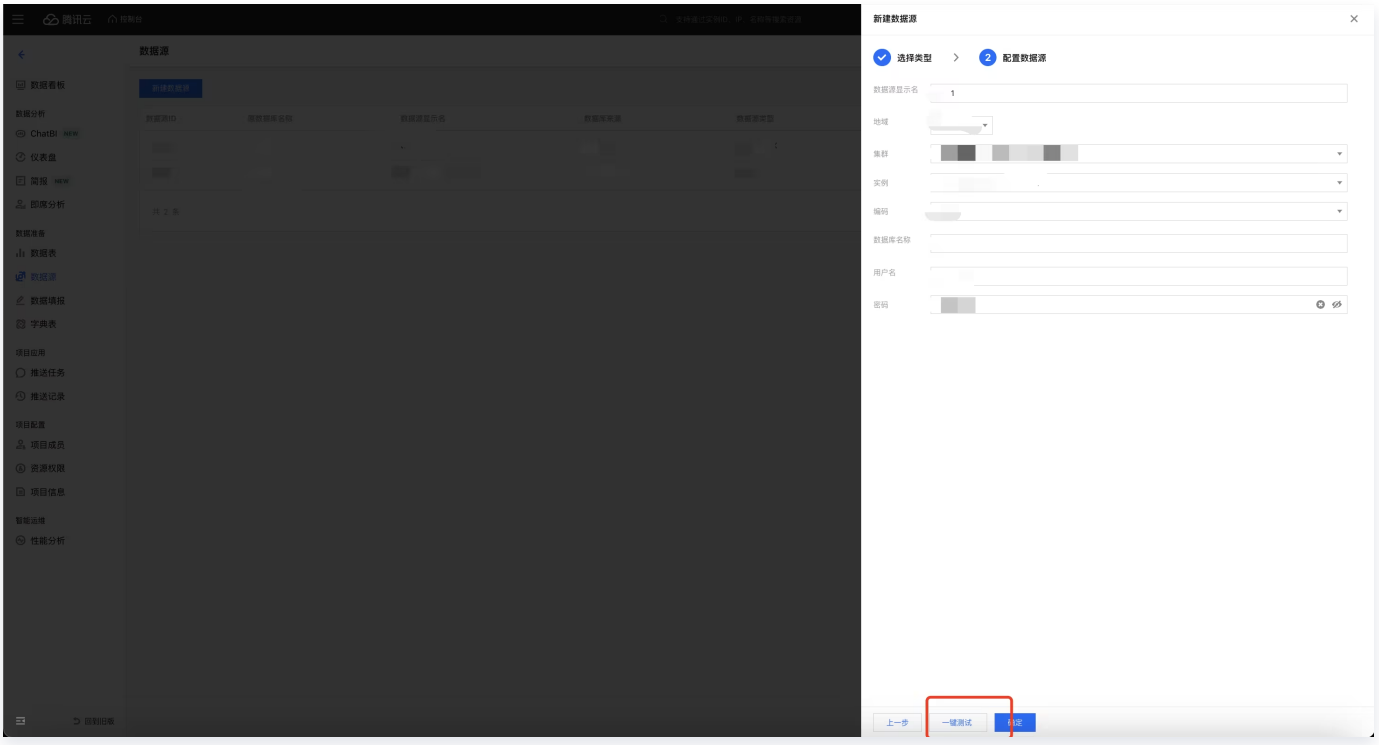
请根据您的 TCHouse-X 实例信息填写下表参数：

参数名称	说明	配置建议 / 示例
数据源显示名	用户自定义的数据源标识名称，用于在 BI 列表中区分不同连接。	建议按“环境-业务”命名，如 PROD_销售分析库
地域	TCHouse-X 实例所在的物理地理位置。	需与实例实际所在地域一致，如华南地区(广州)
集群	需要关联的 TCHouse-X 实例唯一标识符。	格式通常为 instance-xxxx
实例	指定腾讯云 BI 提交 SQL 任务时的目标 Virtual Warehouse（虚拟仓库）。	根据业务负载选择合适的 Virtual Warehouse
编码	腾讯云 BI 与数据库传输数据时使用的字符集。	建议使用 utf8，以确保多语言字符不乱码。
数据库名称	腾讯云 BI 建立连接后默认切换到的 TCHouse-X 数据库（Database）。	默认连接库，如 default 或自定义业务库
用户名	用于数据库访问身份验证的账号。	需具备对应库表的 SELECT 权限
密码	对应用户名的访问凭证。	-

说明：

> 腾讯云 BI 与 TCHouse-X 的连接目前支持以下两类用户身份：
> - 自定义用户
> - 自定义 LDAP 用户

4. **连通性测试与保存**
   - **测试连接**：参数填写完成后，务必单击底部的 **“一键测试”** 。
   - **确认结果**：若提示 “数据库连接成功”，说明网络及账号权限均通过校验。
   - **完成创建**：单击**确定**，该数据源将正式加入当前项目的数据列表。



后续操作

数据建模

成功连接后，您可以开始将 TCHouse-X 中的表添加至数据集，进行维度和指标的定义。

报表分析

利用腾讯云 BI 强大的可视化能力，通过拖拽式操作构建仪表盘、大屏或自助分析报表。

> **说明：**
> 详情请参见 腾讯云 BI 官方文档 。

Power BI 连接 TCHouse-X 操作指南

最近更新时间：2026-05-06 16:28:12

本指南介绍如何通过 TCHouse-X 专用连接器在 Power BI Desktop 中进行数据查询与可视化分析。

前置环境准备

在开始配置前，请确保您的 Windows 客户端实例满足以下要求，并确保其与 TCHouse-X 实例之间的网络连通性（如安全组规则已开放）。

硬件与系统要求

- 操作系统：Windows 10 或 Windows Server 2016 及更高版本。
- 处理器：x64 架构；最低 1.4 GHz，推荐 2.0 GHz 或更高。
- 内存：最低 1 GB，推荐 4 GB 或以上。
- 框架：安装 .NET Framework 4.8。

必要组件安装

请依次检查并安装以下组件，这是驱动正常运行的基础：

1. MDAC：确保 Microsoft 数据访问组件为最新版本。
2. Visual C++ Redistributable：必须安装 VC++ Runtime 2017 或更高版本 (x64)，例如 Microsoft Visual C++ Redistributable for Visual Studio 2022（x64）。
3. OpenSSL：系统中需预装 OpenSSL 环境。

驱动与连接器安装步骤

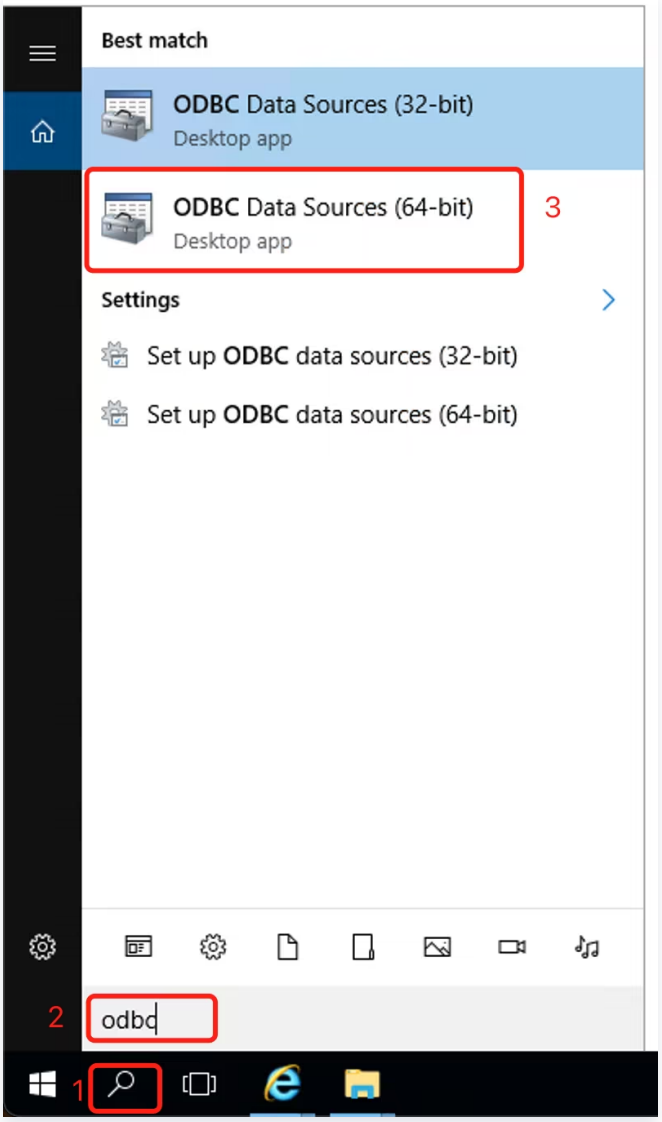
Step 1：安装并配置 TCHouse-X ODBC 驱动

解压安装

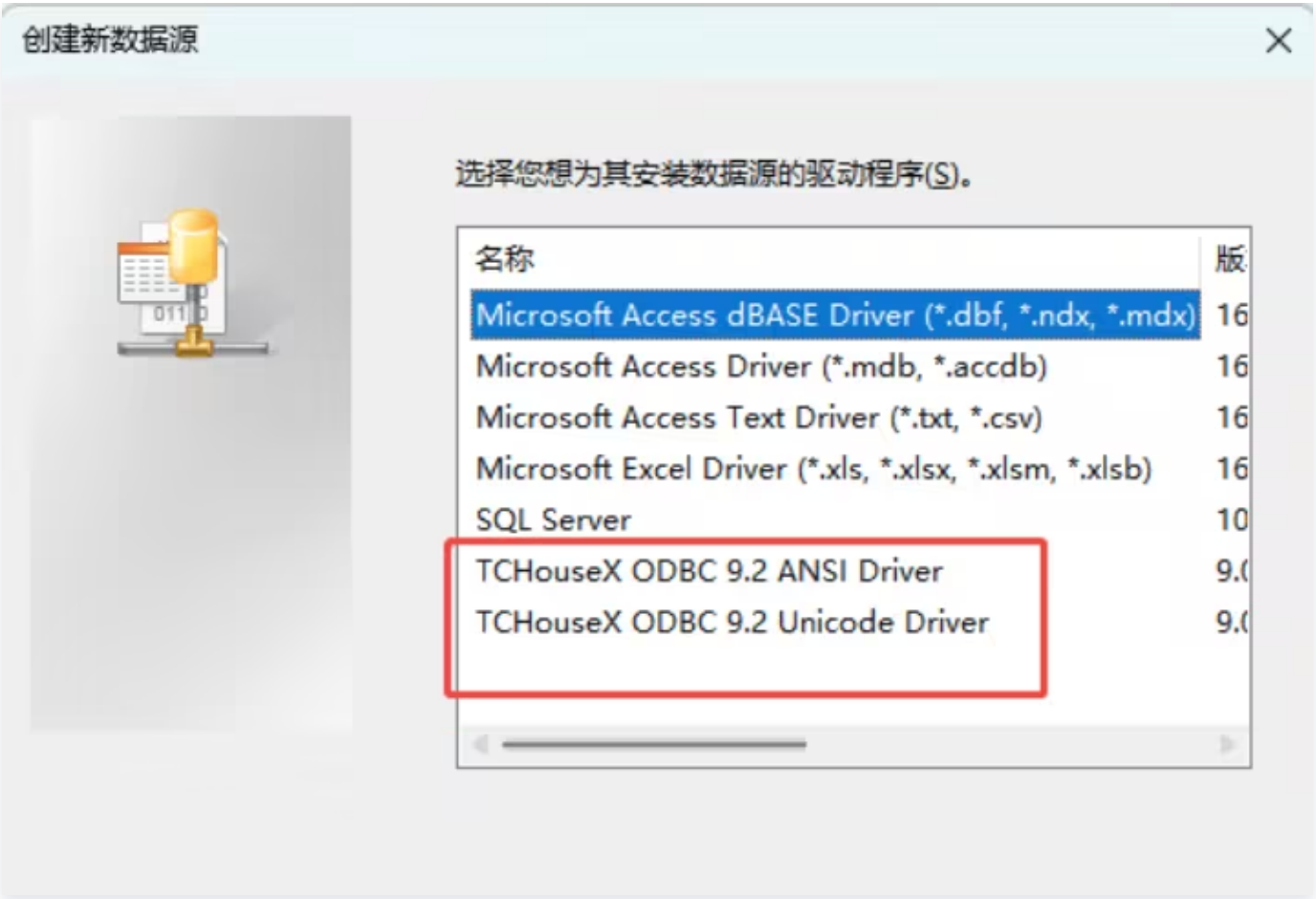
- 将 TCHouse-X Connector ODBC 9.2.3.zip 解压至自定义目录（如 `C:\Program Files\MySQL\TCHouseX Connector ODBC 9.2`）。
- 以管理员身份运行目录下的 `Install.bat` 完成注册。

配置 DSN (数据源名称)

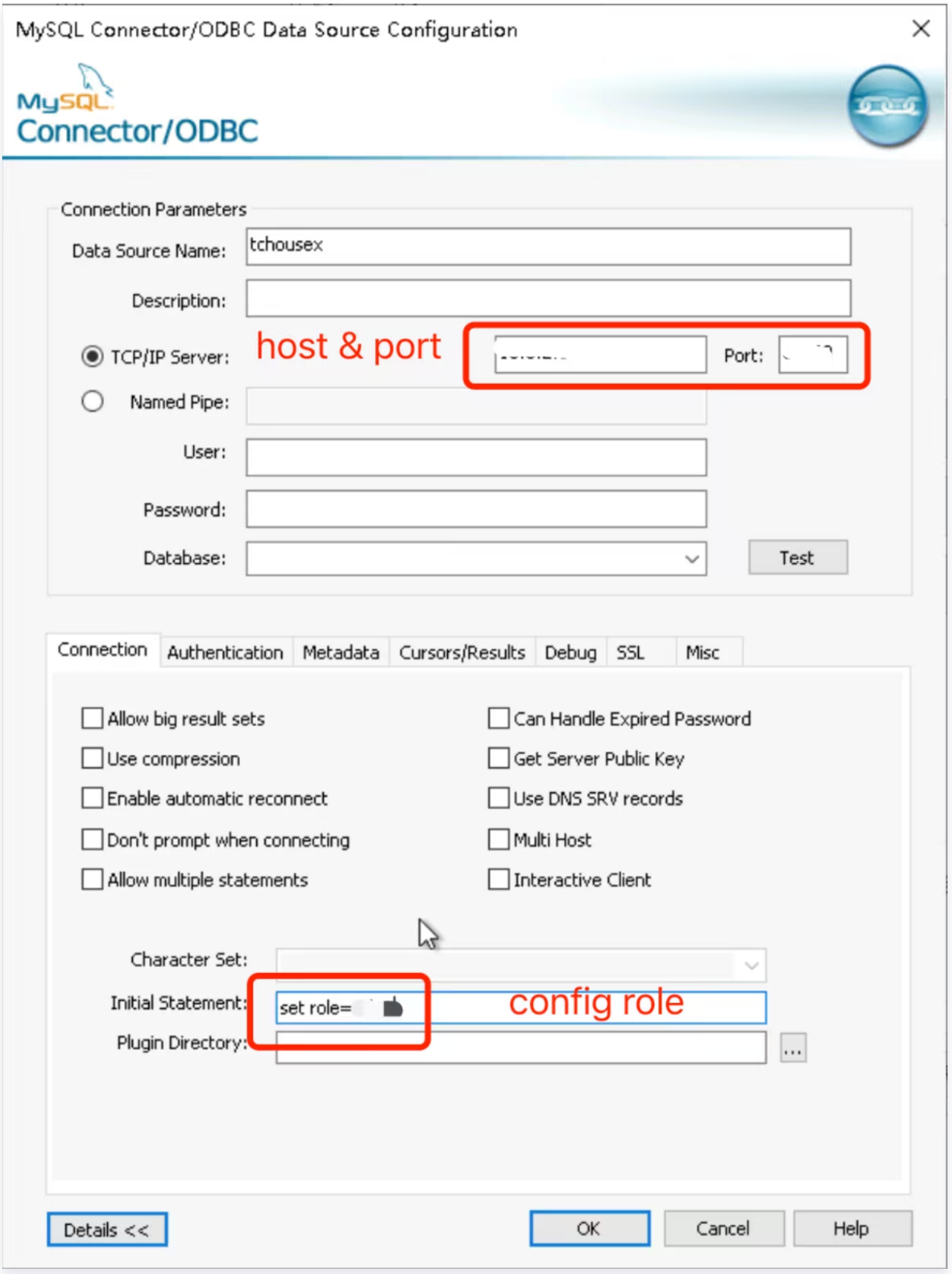
- 按照截图顺序，打开 Windows “ODBC 数据源管理器 (64位)”。



- 在“系统 DSN”页签点击“添加”，选择如下 TCHouse-X Driver 中的一个，优先选择 TCHouseX ODBC 9.2 Unicode Driver。

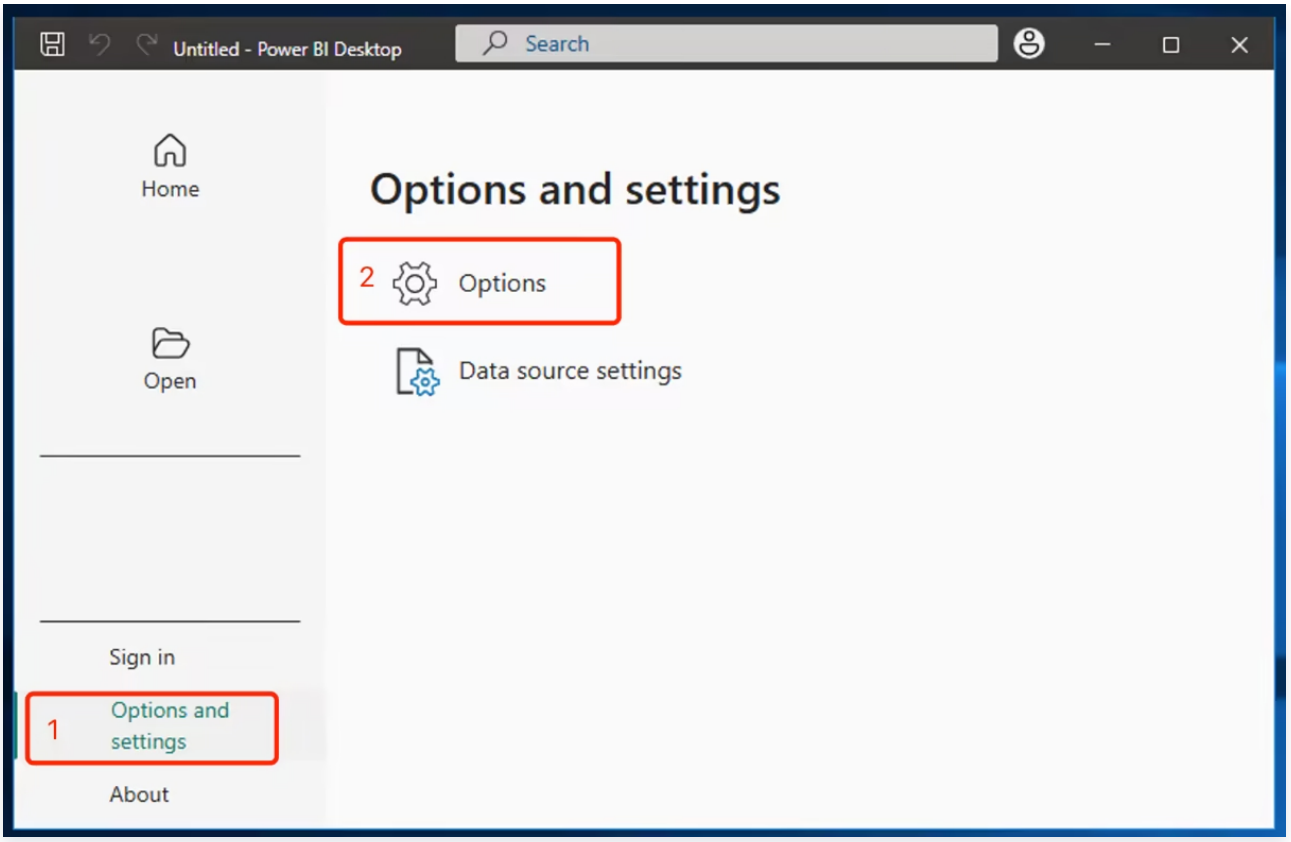


- 关键配置项：
  - Data Source Name：自定义 Data Source Name（必填）。
  - TCP/IP Server：填入 TCHouse-X 实例的连接地址和端口（必填）。
  - 用户验证：若需点击“Test Connection”验证连通性，请填写 User 和 Password。
  - 角色配置（可选）：若需指定 Role，在 Initial Statement 框中填写 `set role = <your_role>` 。

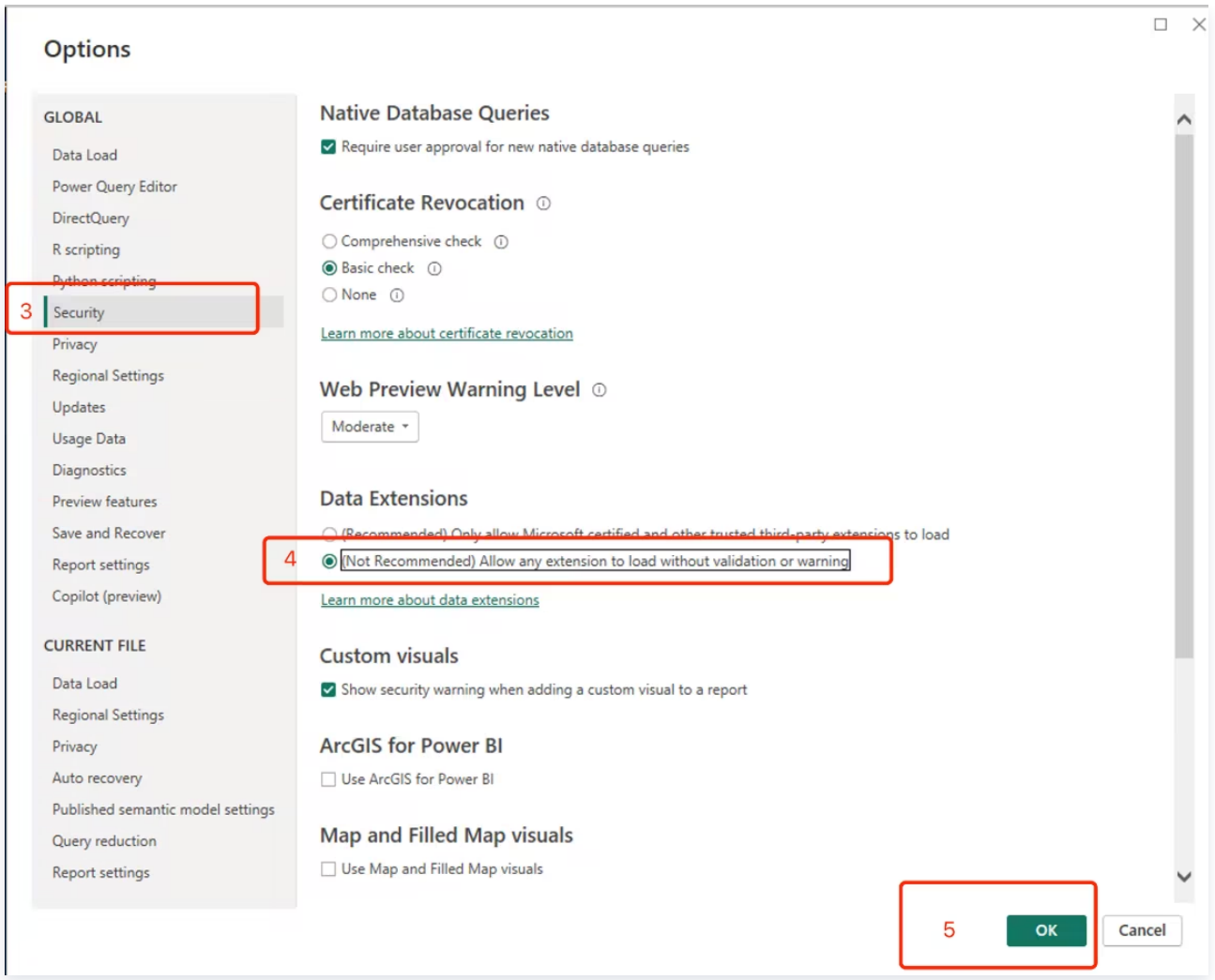


Step 2：安装 TCHouseX SDK 插件

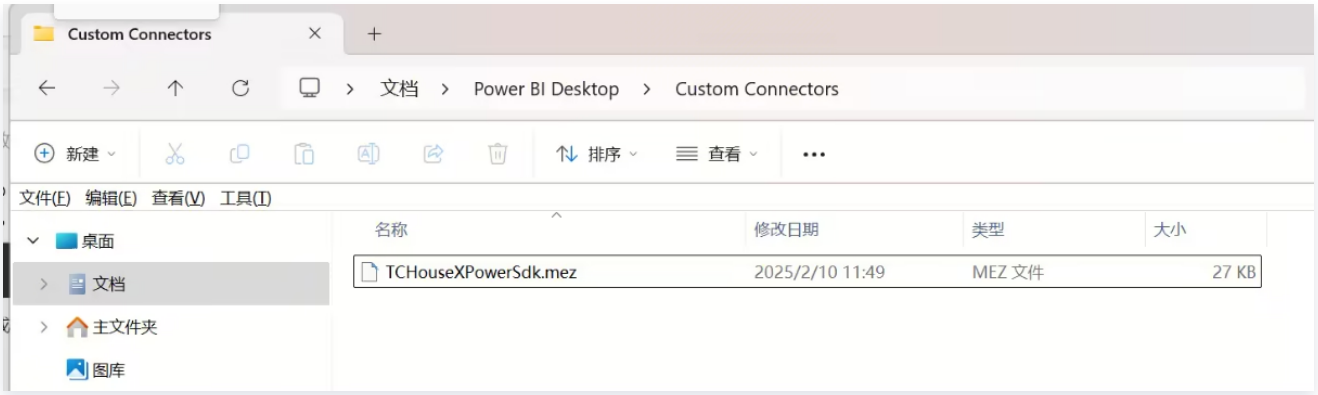
1. 启用自定义扩展：
   - 打开 Power BI Desktop，进入 Options and settings > Options。



- 在 Global > Security 中，勾选 “(Not recommended) Allow any extension to load without verification or warning”。

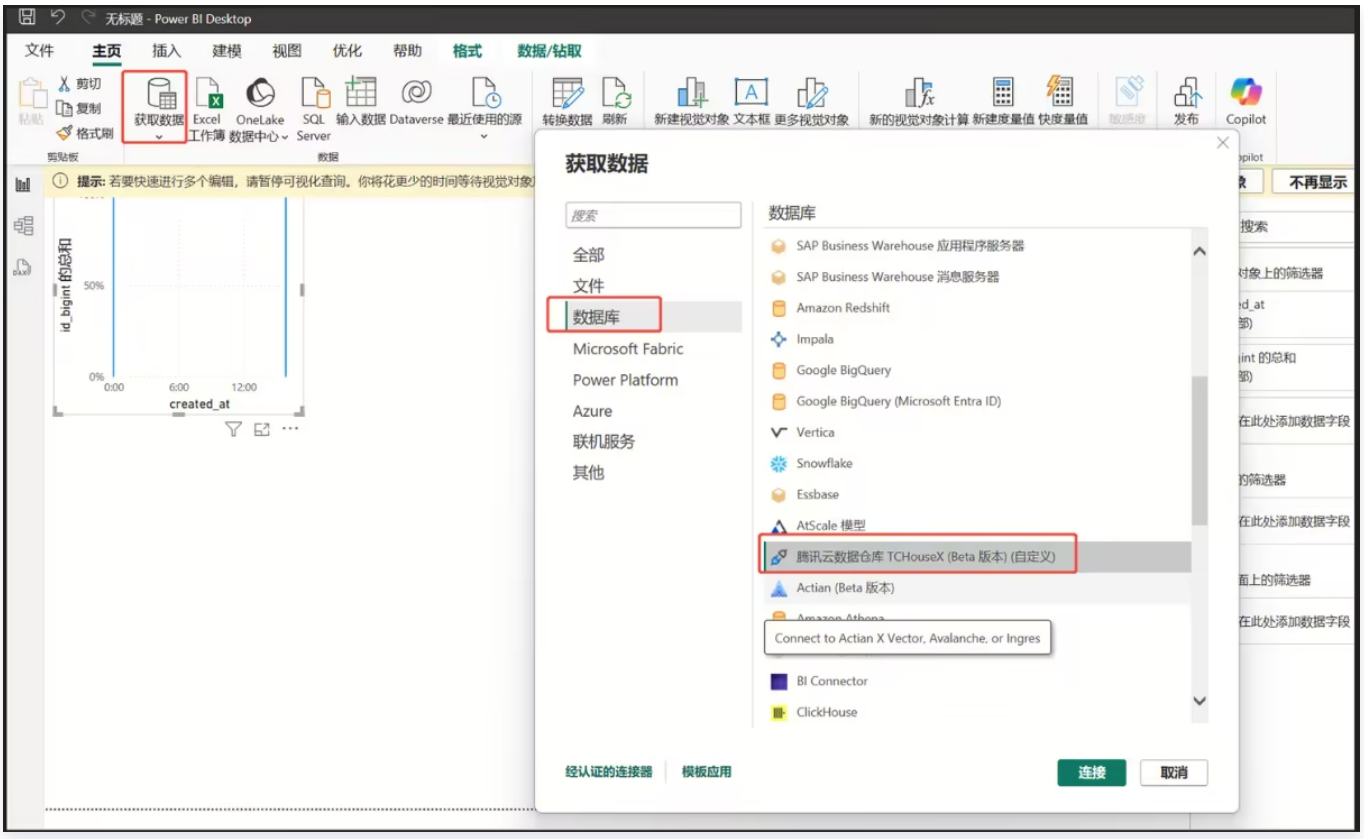


2. 启用自定义扩展：将 TCHouse-X-Power-SDK.mez 拷贝至 Power BI 插件路径： 文档\Power BI Desktop\Custom Connectors

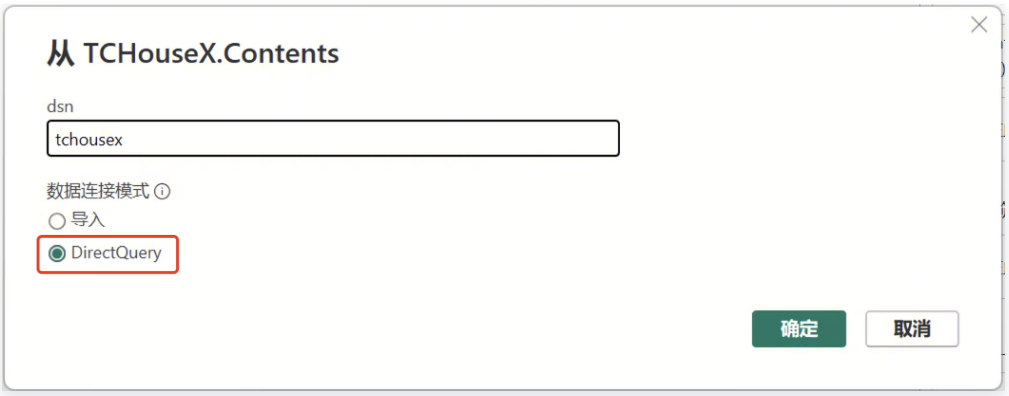


连接 TCHouse-X 进行分析

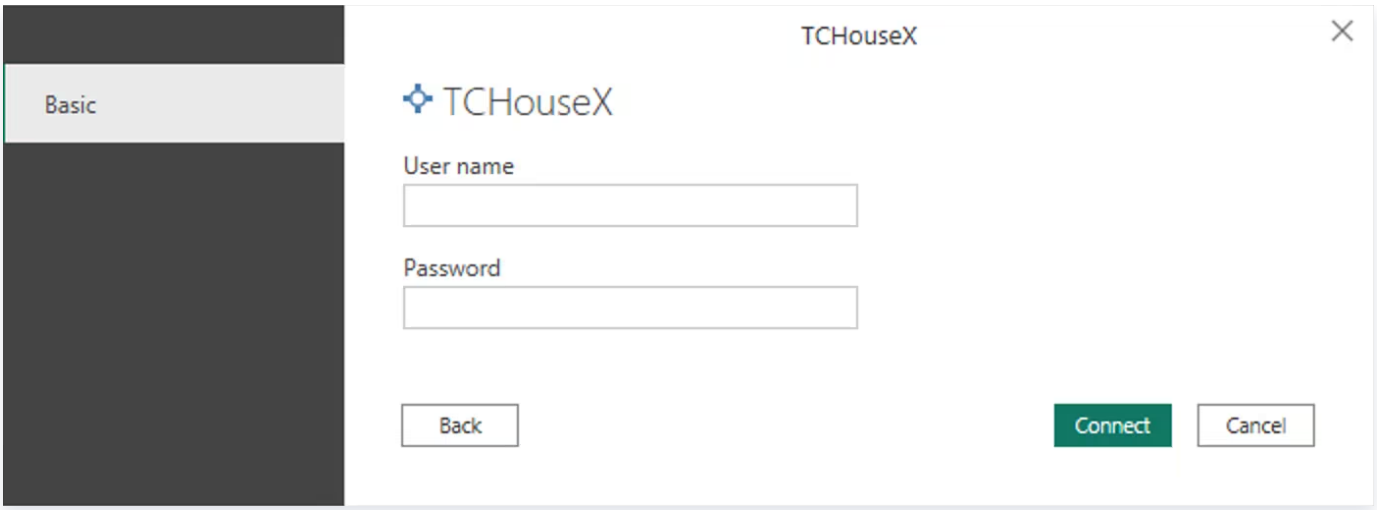
1. 获取数据：启动 Power BI Desktop，点击 获取数据 > 更多，搜索并选择 TCHouseX SDK。



2. 设置连接模式：
   - 在 DSN 栏输入 Step 1 中配置好的 Data Source Name（如 tchousex）。
   - 数据连接模式：选择 DirectQuery 以获得实时查询性能。

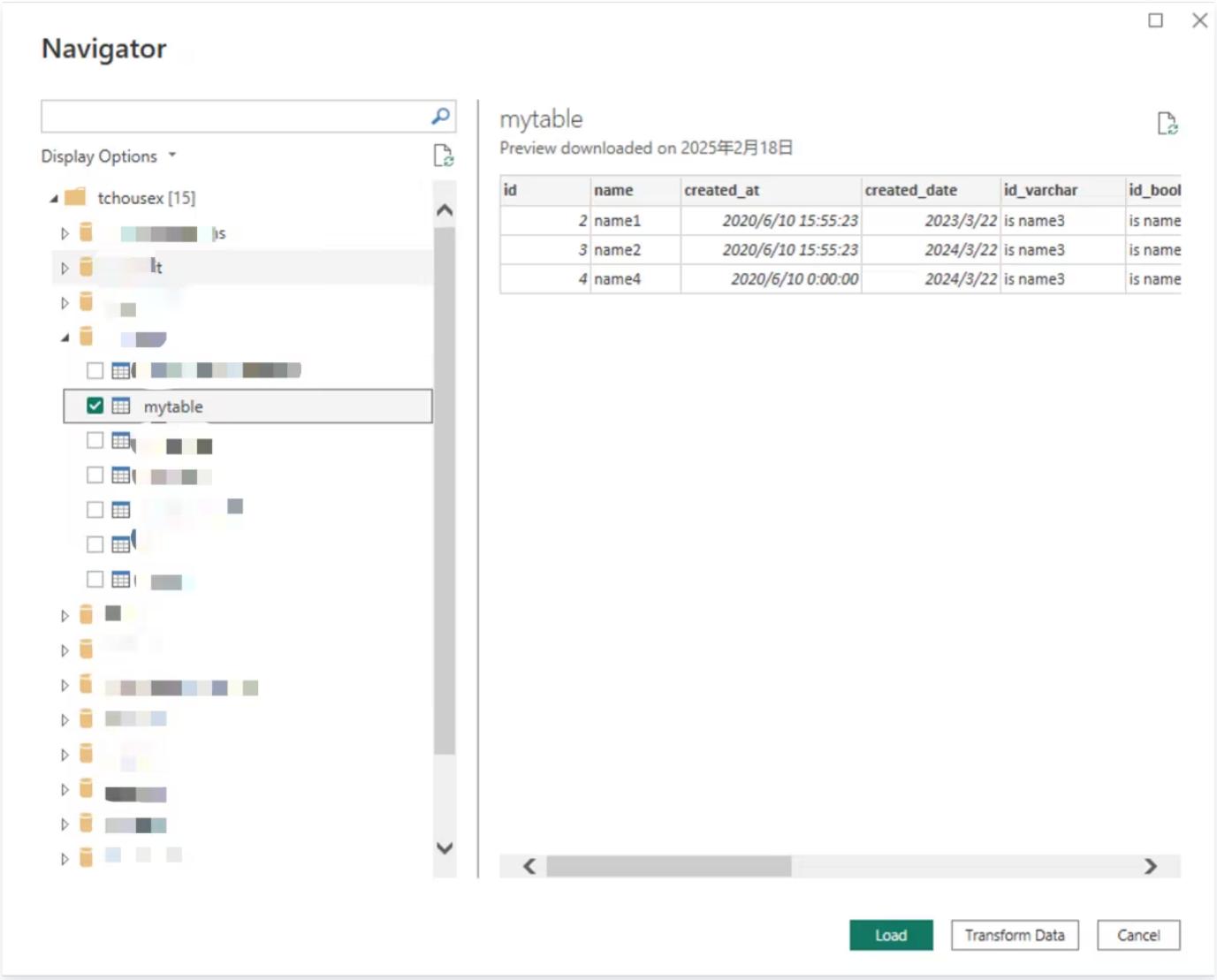


3. 身份验证：在弹出的窗口中输入 TCHouse-X 的账号和密码。

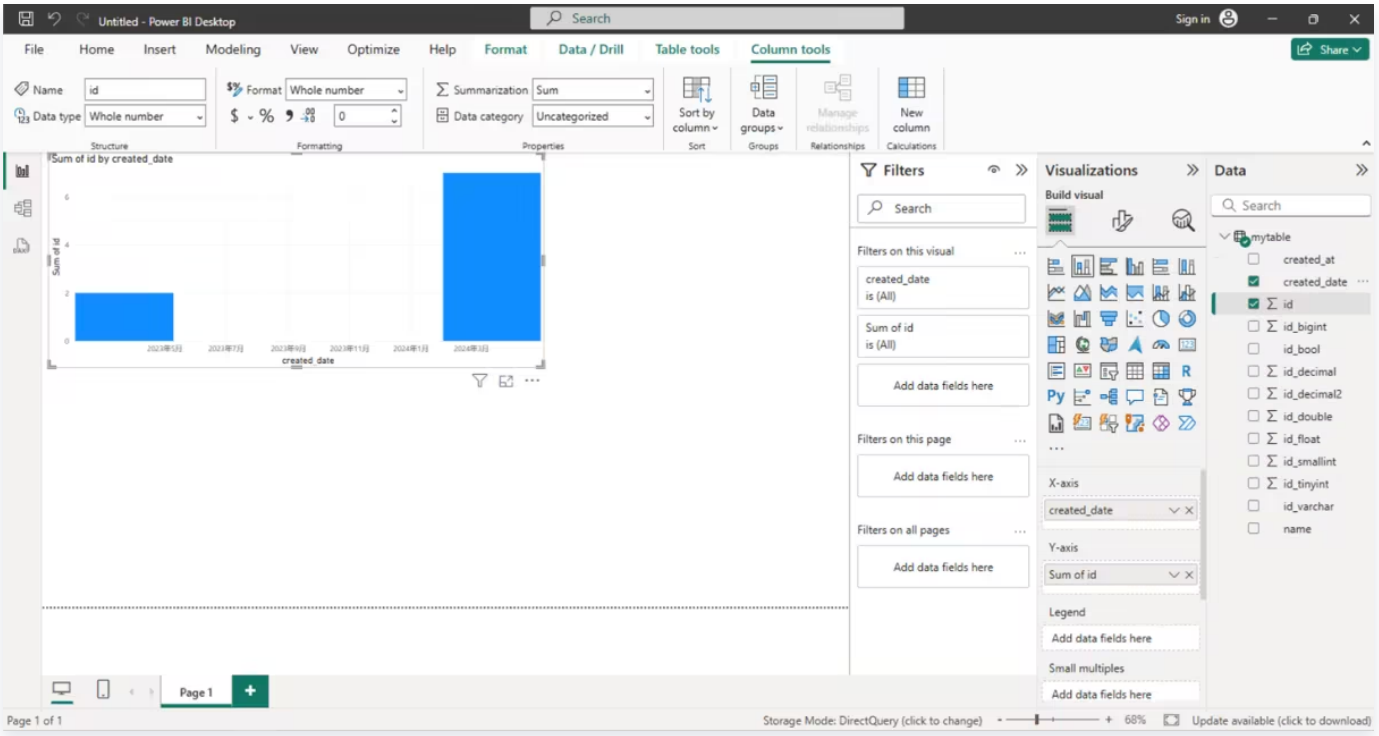


> **注意：**
> Power BI 仅支持 TCHouse-X 的**自定义用户与自定义 LDAP 用户**登录。

4. 加载数据：在导航器中勾选需要的表，点击 Load。



5. 此时，即可使用 Power BI 的拖拽功能，进行数据分析



使用限制与注意事项

为避免查询报错，请在设计报表时注意以下限制：

维度	说明
空值计数	执行计数计算（Count）时，`NULL` 值会被计入。如需排除，请在 Power BI 中手动配置过滤条件。
日期范围	Date 数据类型仅支持：`1400-01-01` ~ `9999-12-31`。
数据量限制	单次查询结果不能超过 100 万行，否则会触发“提取此视觉对象数据时出错”的报错。