数据开发治理平台 WeData

快速入门

腾讯云

文档目录

快速入门
整体介绍

最近更新时间：2024-12-26 17:04:12

> **说明：**
> 本案例中所涉及的产品功能仅为模块内比较常用的功能，如需更细致的了解模块全部功能请查看 WeData 操作指南 。
> 除了文档之外，您也可以通过 腾讯云 WeData 大数据开发与治理训练营 学习“快速入门”的内容。

背景

本文档是腾讯云 WeData 的基础使用文档，目的是帮助您快速了解 WeData，并对业务数据加工全流程有一个基本的概念。
本文以订单数据同步分析场景为例，串联数据表结构设计、数据集成、数据开发、数据质量、数据服务等各模块功能，帮助您完成 WeData 全流程初体验。

通过本文档学习，您能够了解以下内容：

- 了解业务数据开发的全流程。
- 了解产品各模块的作用与上下游协同。
- 了解数据表结构设计基础概念。
- 掌握数据离线数据同步流程。
- 掌握数据离线数据开发流程。
- 掌握数据质量检测流程。
- 掌握数据服务开发流程。

本文档涉及以下角色与分工：

- **企业管理员：**
  - 负责注册并认证腾讯云账号。
  - 负责搭建网络环境。
  - 负责购买各种云资源，包括：EMR、WeData、MySQL、数据服务资源。
  - 负责创建子账号、创建项目、创建数据表。
  - 负责在 WeData 中添加子账号、绑定数据源。
- **数据开发人员**
  - 负责数据表结构设计。
  - 负责数据集成模块。

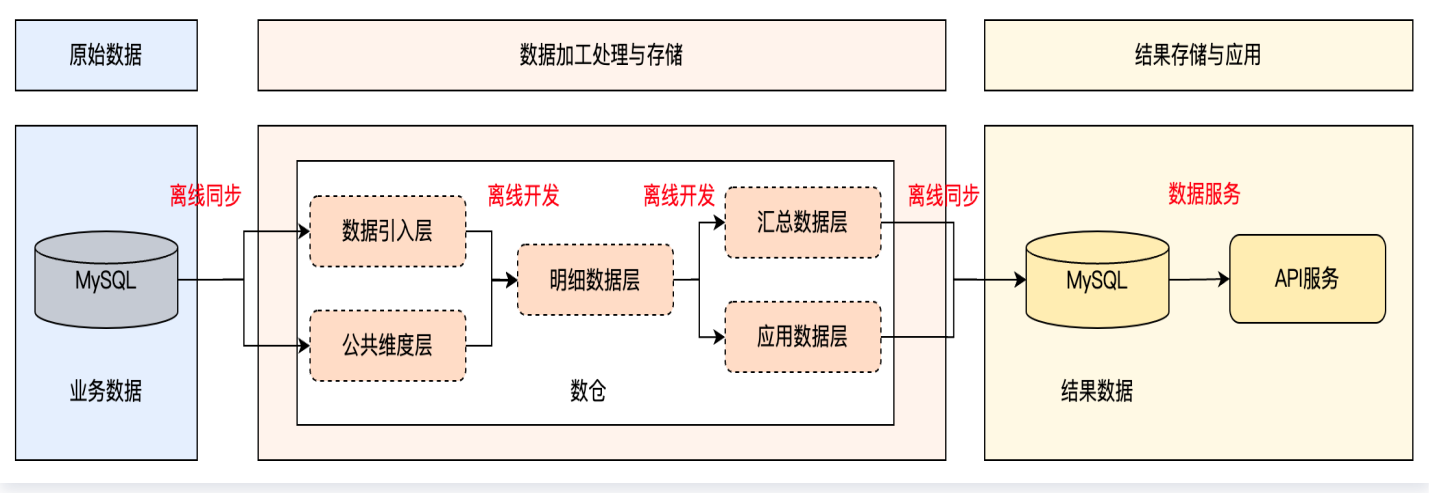
- 负责数据开发模块。
- 负责数据质量模块。
- 负责数据服务模块。

场景介绍

在某电商平台中，业务方**期望通过分析订单数据，了解不同城市各个品类的销售情况**，以便针对不同城市调整运营推广策略。

流程全貌

在 WeData 产品整个产品链路中，会涉及到原始数据调研、数据加工处理与存储，结果存储与应用等阶段。



准备工作

最近更新时间：2024-09-04 20:44:31

在开始 WeData 数据开发之前，我们首先需要进行以下准备：

> **说明：**
> 以下操作涉及资源购买和付费内容，需要由企业管理员进行操作。

操作流程

具体操作包括：

步骤	说明
注册腾讯云账号	• 注册腾讯云账户。 • 实名认证（建议企业认证）。 • 创建腾讯云子账户。
准备网络环境	• 新建私有网络。 • 绑定子网。 • 申请公网 IP。 • 购买公网 NAT 网关，并绑定私有网络、绑定公网 IP。
准备引擎资源环境（以腾讯云 EMR 为例）	• 购买 EMR 集群。 • 购买 WeData，包含集成资源、调度资源。 • 创建项目，并绑定 EMR，绑定集成资源、调度资源。 • 在 WeData 项目中，添加用户。
准备业务数据资源环境（以腾讯云 MySQL 为例）	• 购买 MySQL，并初始化业务数据。 • 在 WeData 项目中，添加数据源。

注册腾讯云账号

本次教程涉及的所有云资源，均通过腾讯云账号进行购买，请使用同一个腾讯云主账号。如果您已经有了腾讯云账号，并完成了企业认证，请跳过此步骤。
操作角色：企业管理员。

注册腾讯云账号

进入 腾讯云官注册页面 。您可以使用微信扫码或者使用邮箱注册，具体注册过程请参见 腾讯云注册指南 。

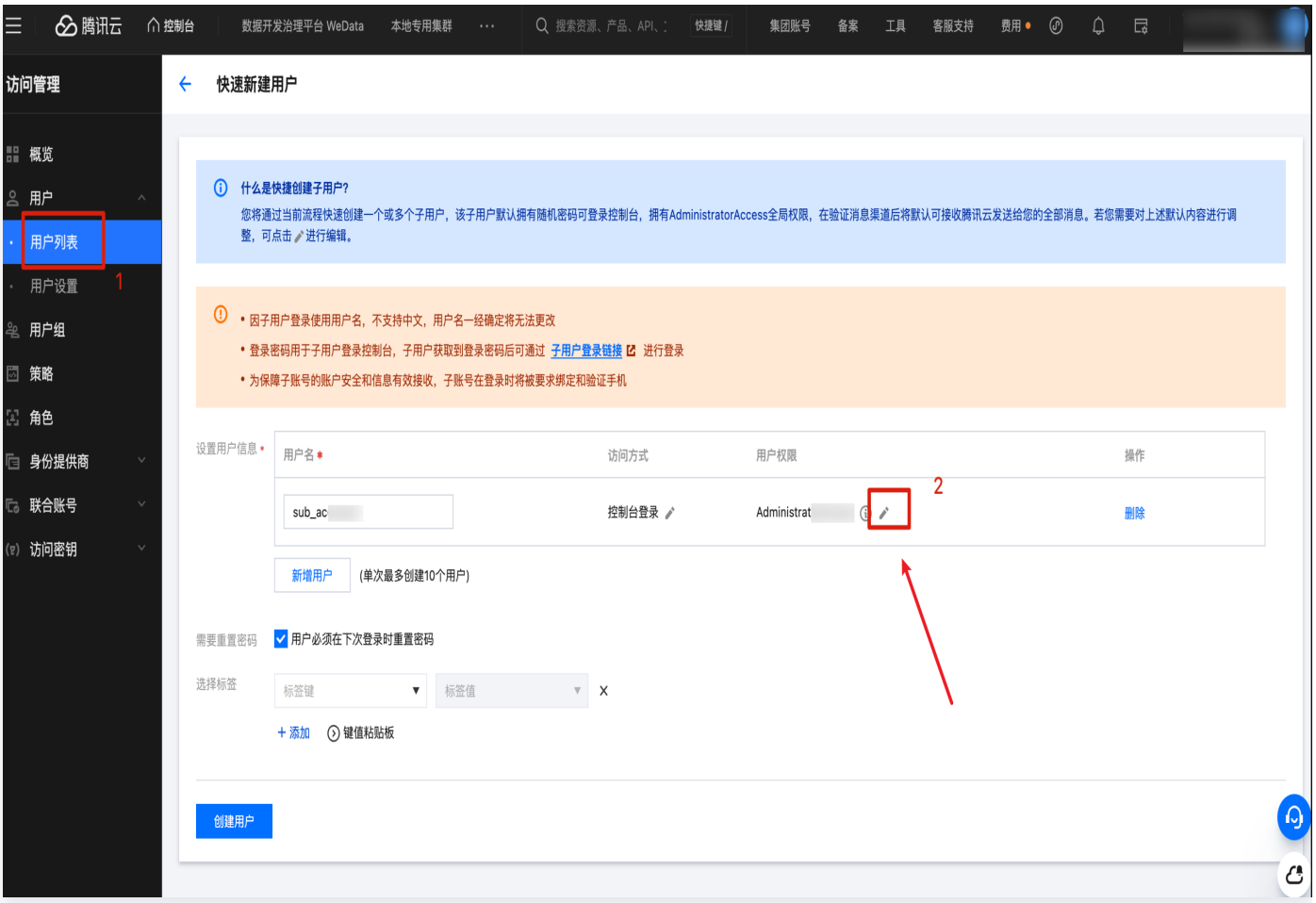
企业实名认证

在完成注册后，需要进行 企业认证 。可选择的认证方式如下，具体注册过程请参见 腾讯云企业实名认证 。

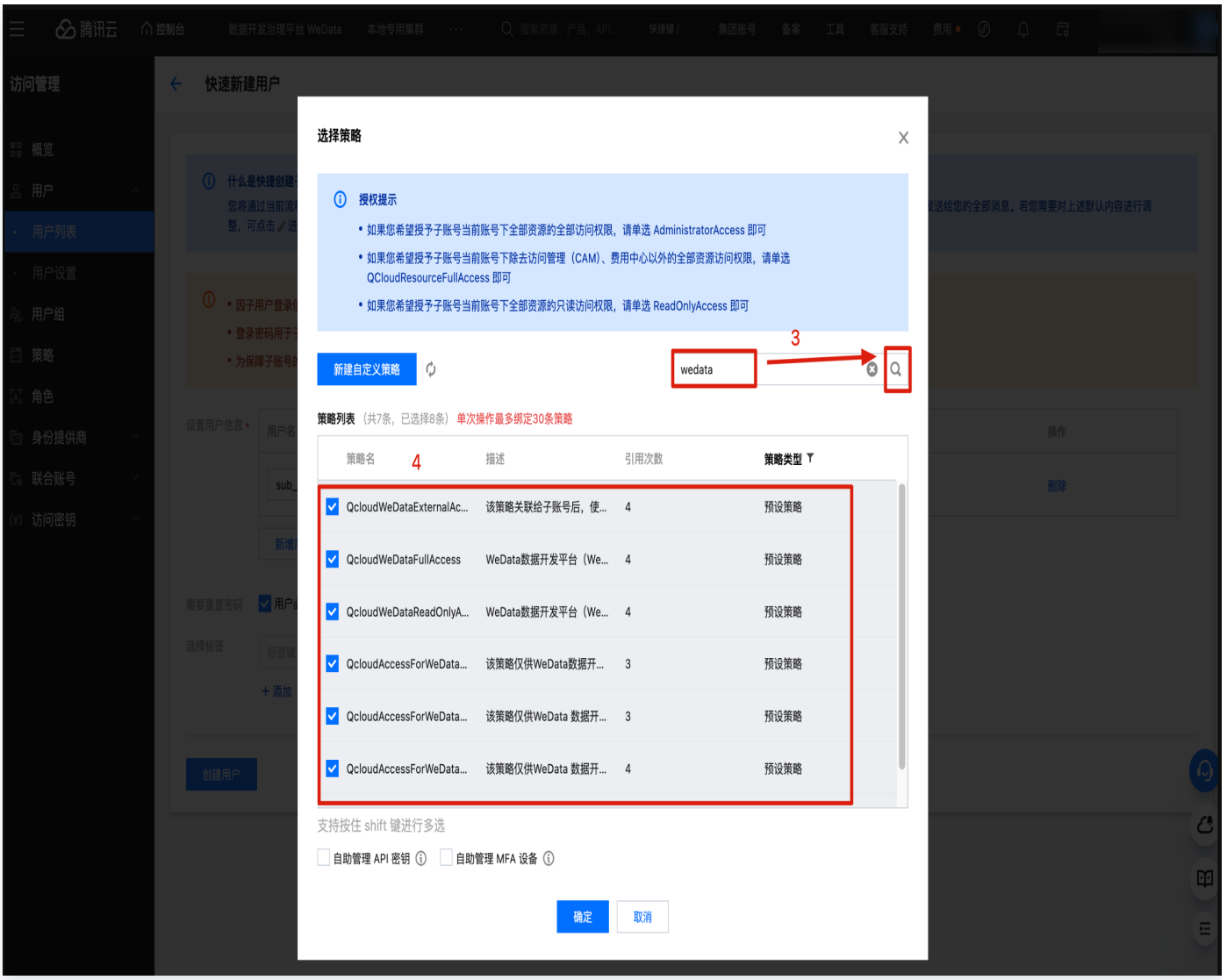
认证方式	认证时长	说明
微信公众平台认证	即时完成	已注册微信公众号且经过微信实名认证的企业，使用此方式，即可立即认证。
企业法人微信扫码认证	即时完成	使用企业法人的个人微信扫码认证，法人微信扫码授权后，即可完成认证。
企业法人人脸识别认证	即时完成	使用企业法人的个人微信扫码进行人脸识别，人脸识别通过后，即可完成认证。
腾讯云充值认证	1个工作日	通过企业银行账户充值一笔系统随机生成的指定金额且充值总金额少于1元的小额验证金（将充入余额），腾讯云收到充值后，即可完成认证。
企业对公打款认证	1 – 5个工作日	输入企业银行账号信息，待腾讯云打款成功后，回填打款金额完成认证。

创建腾讯云子账号

1. 进入**腾讯云控制台** > 用户列表 > **新建用户** > **快速创建**。修改用户权限，单击用户权限的**编辑图标**。



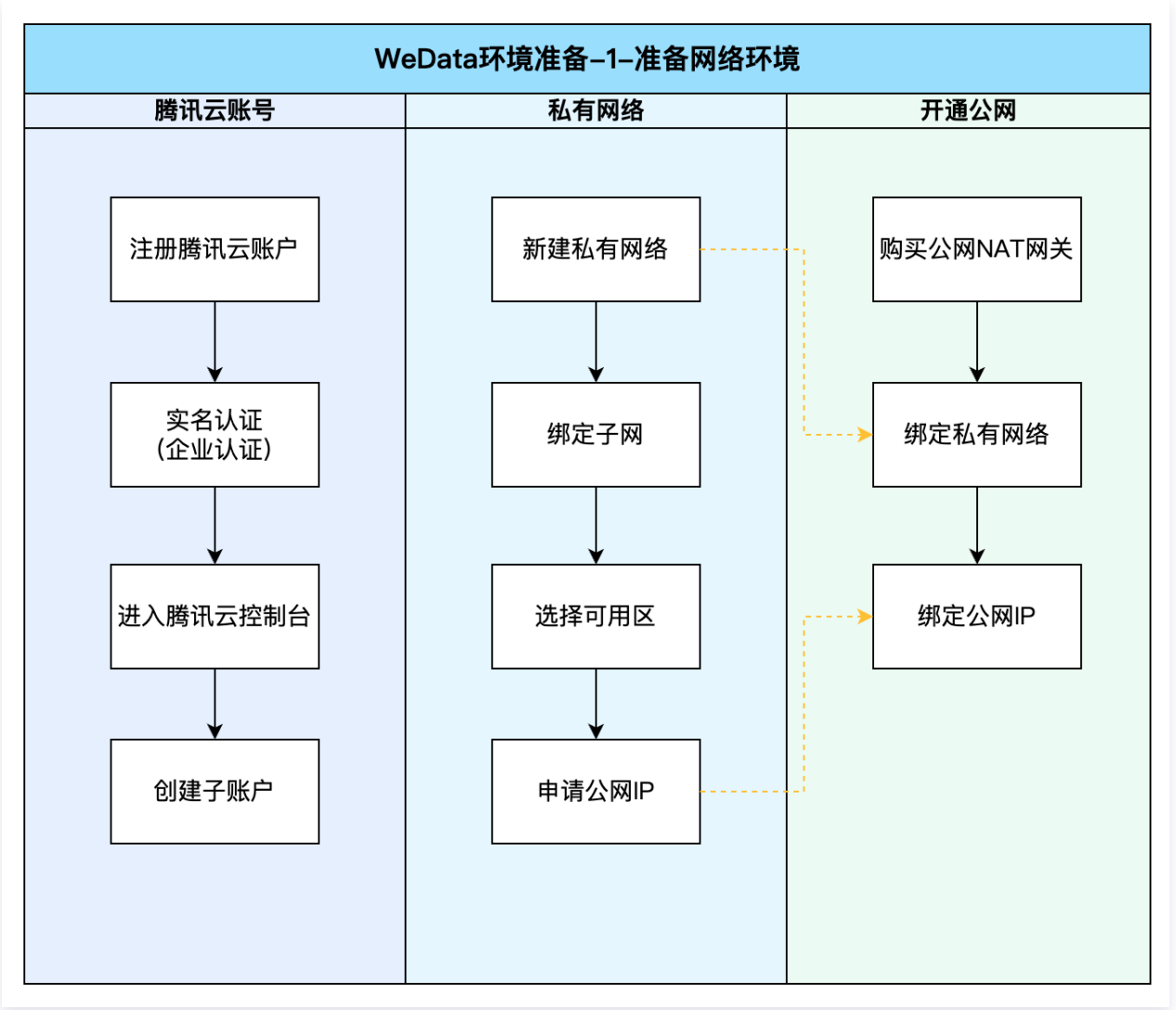
2. 输入 wedata，单击搜索图标。选择全部策略后，单击**确定**。



准备网络环境

在本次教程中涉及多种云资源，为了保证网络联通性，需要搭建私有网络环境。

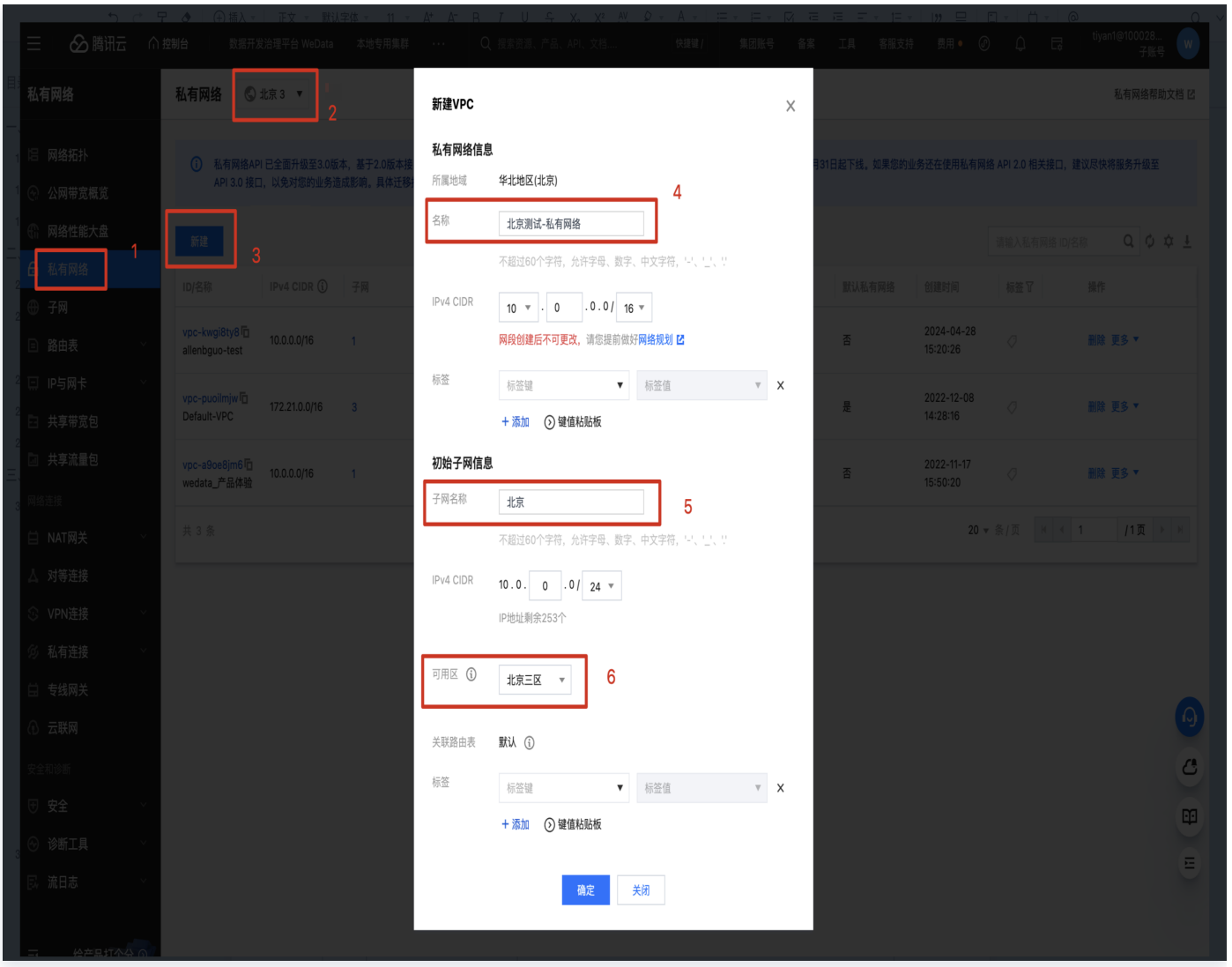
- **操作角色：**企业管理员。
- **操作账号：**腾讯云主账号。
- **操作流程：**



新建私有网络

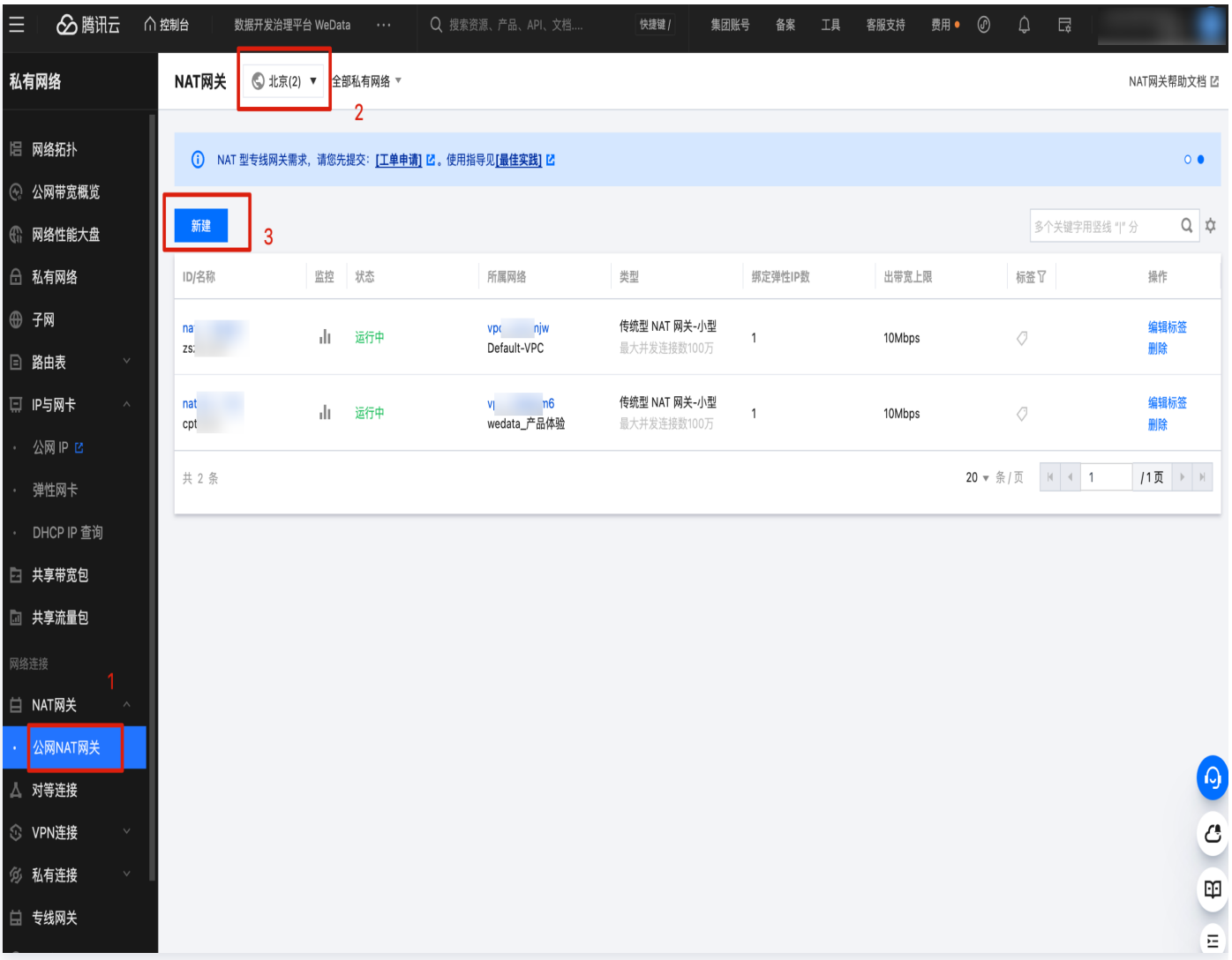
1. 登录腾讯云 私有网络控制台，在**私有网络**页面顶部，选择 VPC 所属地域，示例：选择北京，单击**新建**。
2. 进入新建 VPC 界面，填写私有网络信息和初始子网信息，填写完成后，单击**确定**。
   - 私有网络名称：可随意命名，便于区分即可，示例：北京-私有网络。
   - 子网名称：可随意命名，便于区分即可，建议与下方的可选区保持一致，示例：北京三区。
   - 可选区：此时可任意选择，示例：北京三区，后续购买其他资源时，如没有此可用区，可在私有网络中添加子网。

> **说明：**
> - 地域选择北京，仅为示例，建议您选择距离自己较近的地域。
> - 本次教程中后续购买的资源均会选择北京地域，请务必慎重选择。

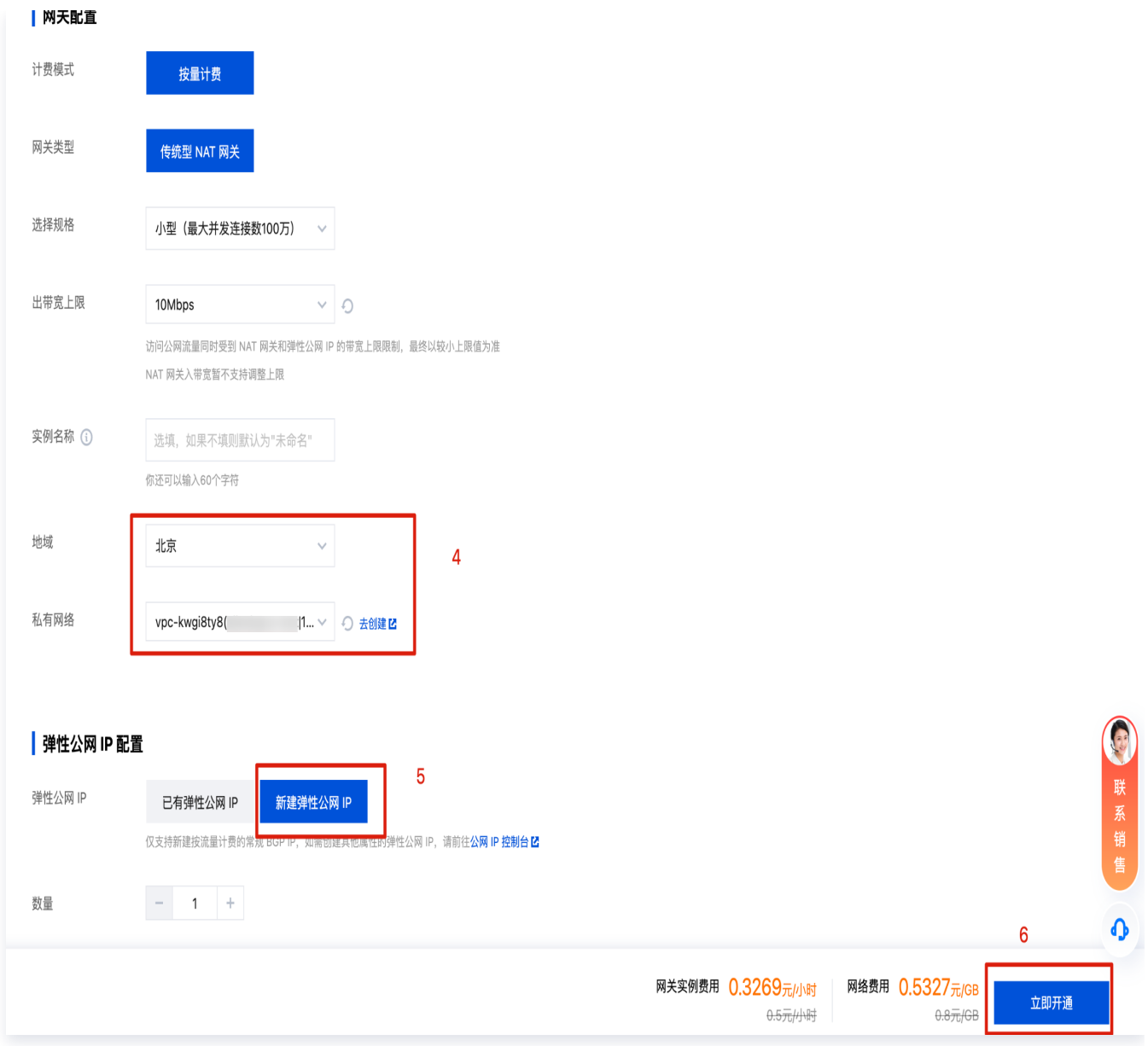


购买 NAT 网关

1. 进入腾讯云 公网 NAT 网关 页面，在 **NAT 网关**页面顶部，选择 VPC 所属地域，示例：选择北京，单击**新建**。



2. 如果您暂无 NAT 网关，请进入购买页进行购买，选择完配置后，单击**立即开通**，核实账单后，付费开通即可。
   - 地域：选择北京，
   - 私有网络：选择刚创建好的私有网络，
   - 弹性公网 IP：选择新建弹性公网 IP，如您已经申请了公网 IP，也可在此处直接进行绑定。

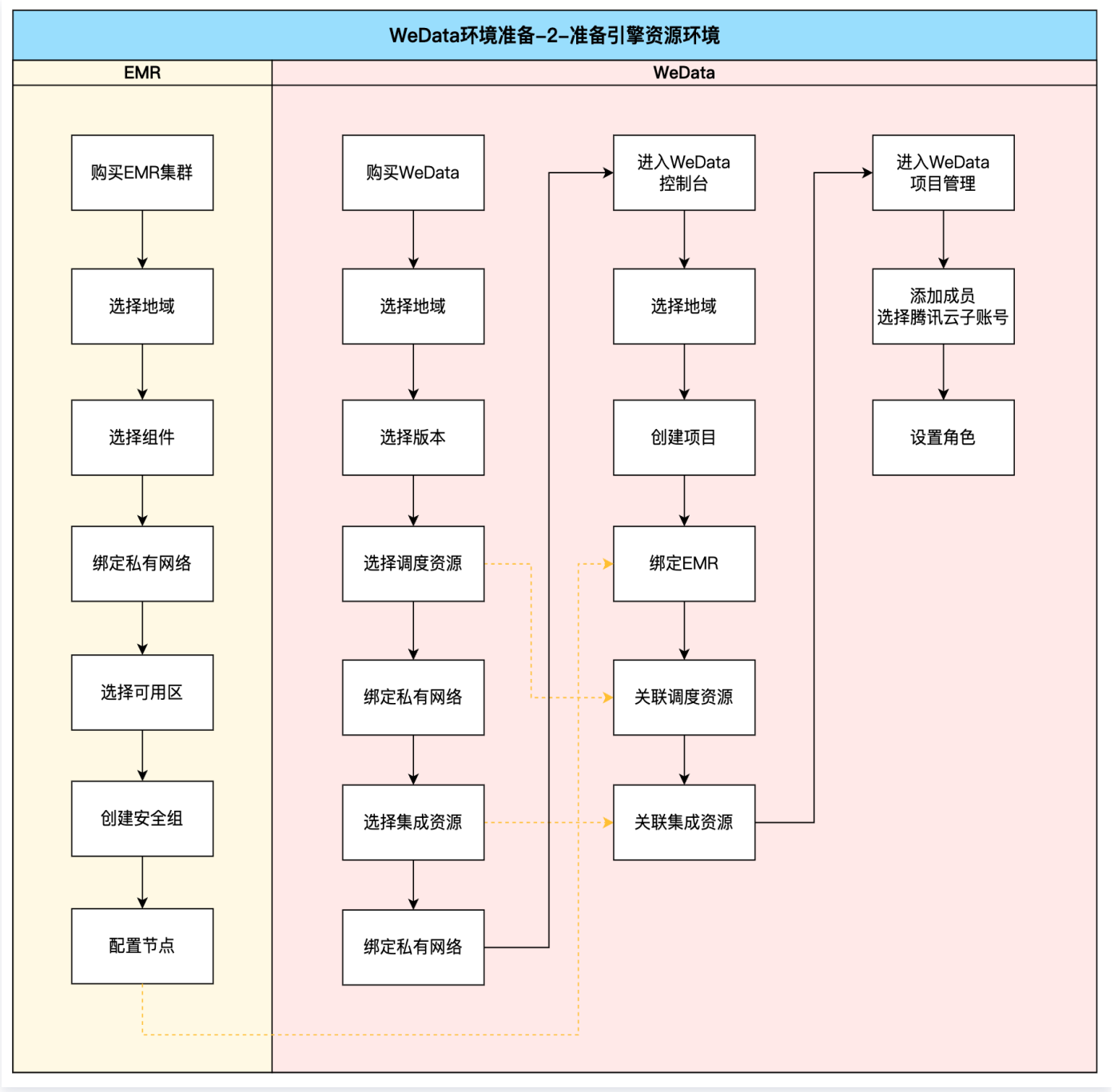


准备引擎资源环境

WeData 作为数据开发治理平台，需要绑定腾讯云大数据套件作为数据存储和数据计算引擎，例如，腾讯云 EMR、DLC、TCHouse 等产品。

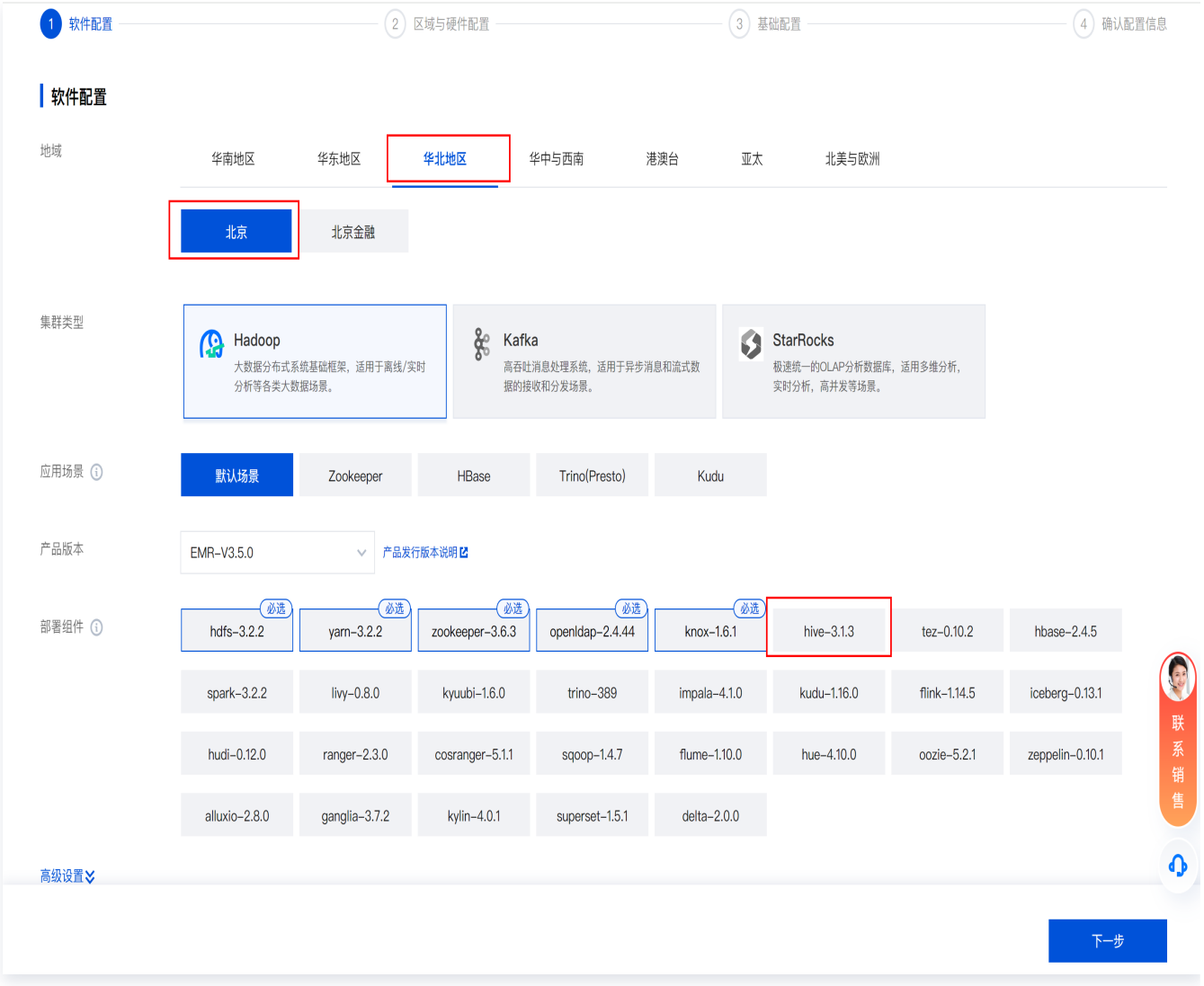
本教程中采用 EMR 作为示例，介绍 WeData 的数据同步、数据开发过程，因此我们需要先在腾讯云上购买一套 EMR 环境。

- **操作角色：**企业管理员。
- **操作账号：**腾讯云主账号。
- **操作流程：**

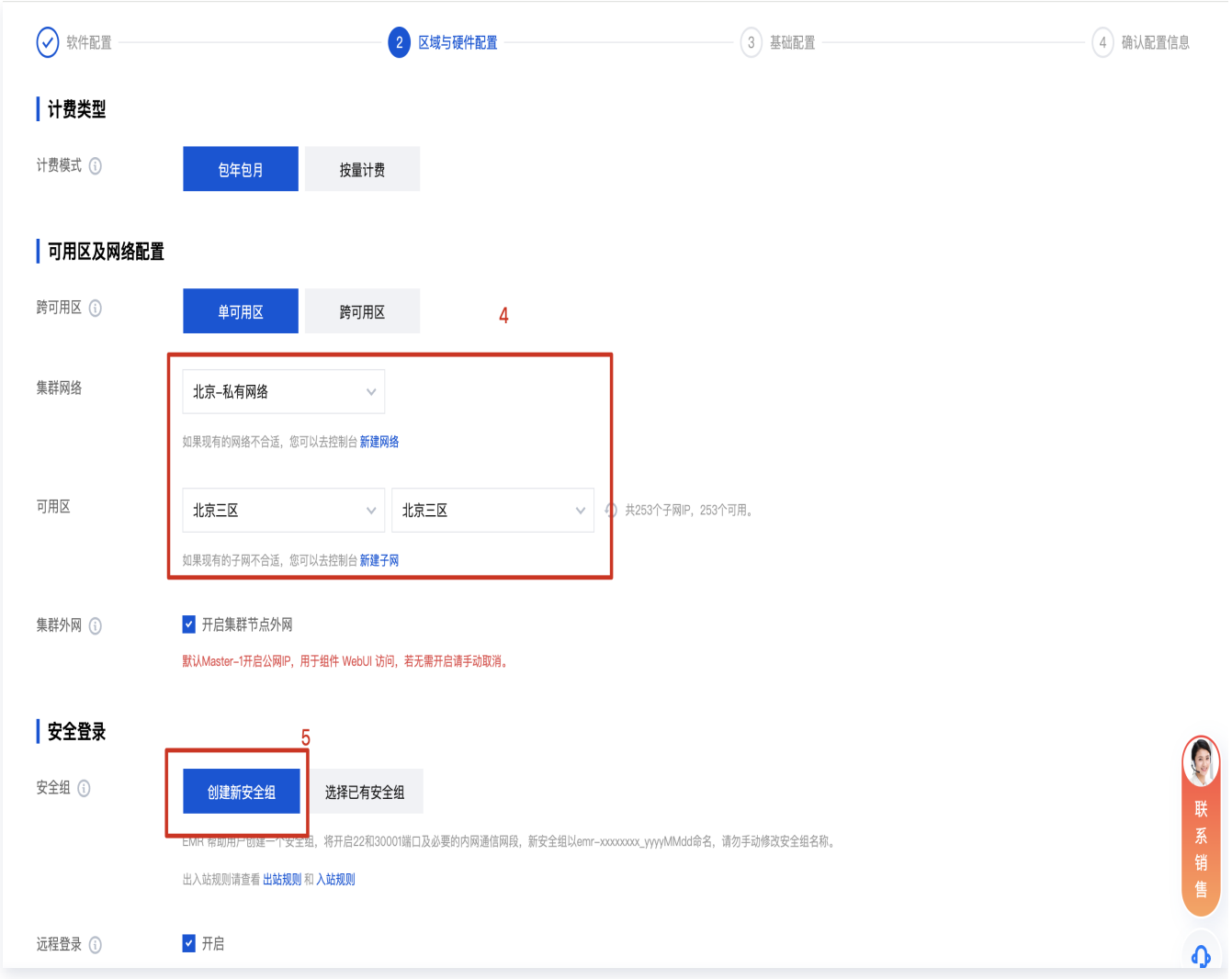


购买 EMR

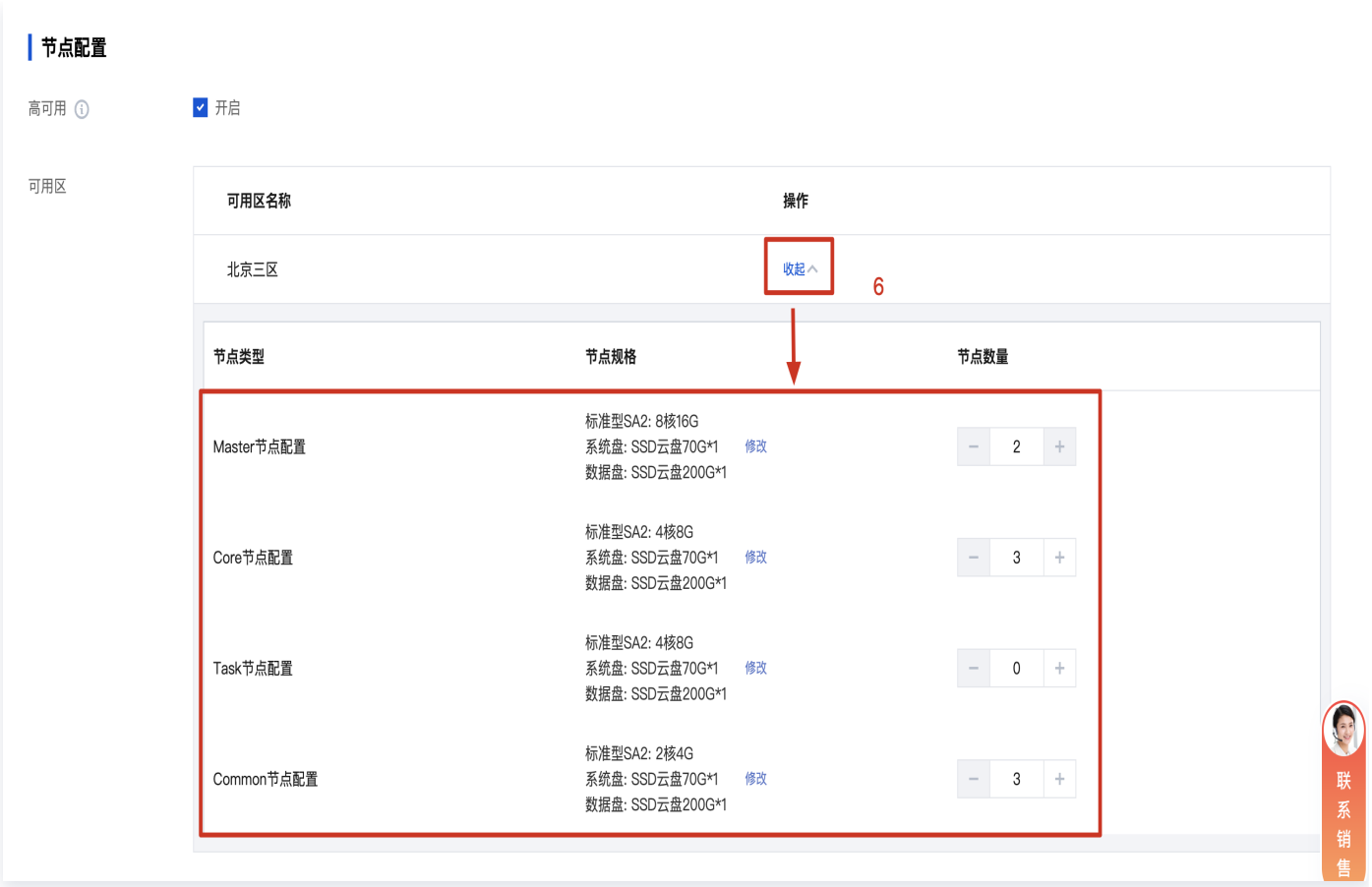
1. 进入腾讯云 弹性 MapReduce 购买页，第一步选择软件配置，选择完成后，单击**下一步**。
   - 地域：选择华北地区-北京
   - 应用场景：默认场景
   - 部署组件：选择 Hive-3.1.3，本次教程中采用 Hive 作为存储计算引擎。



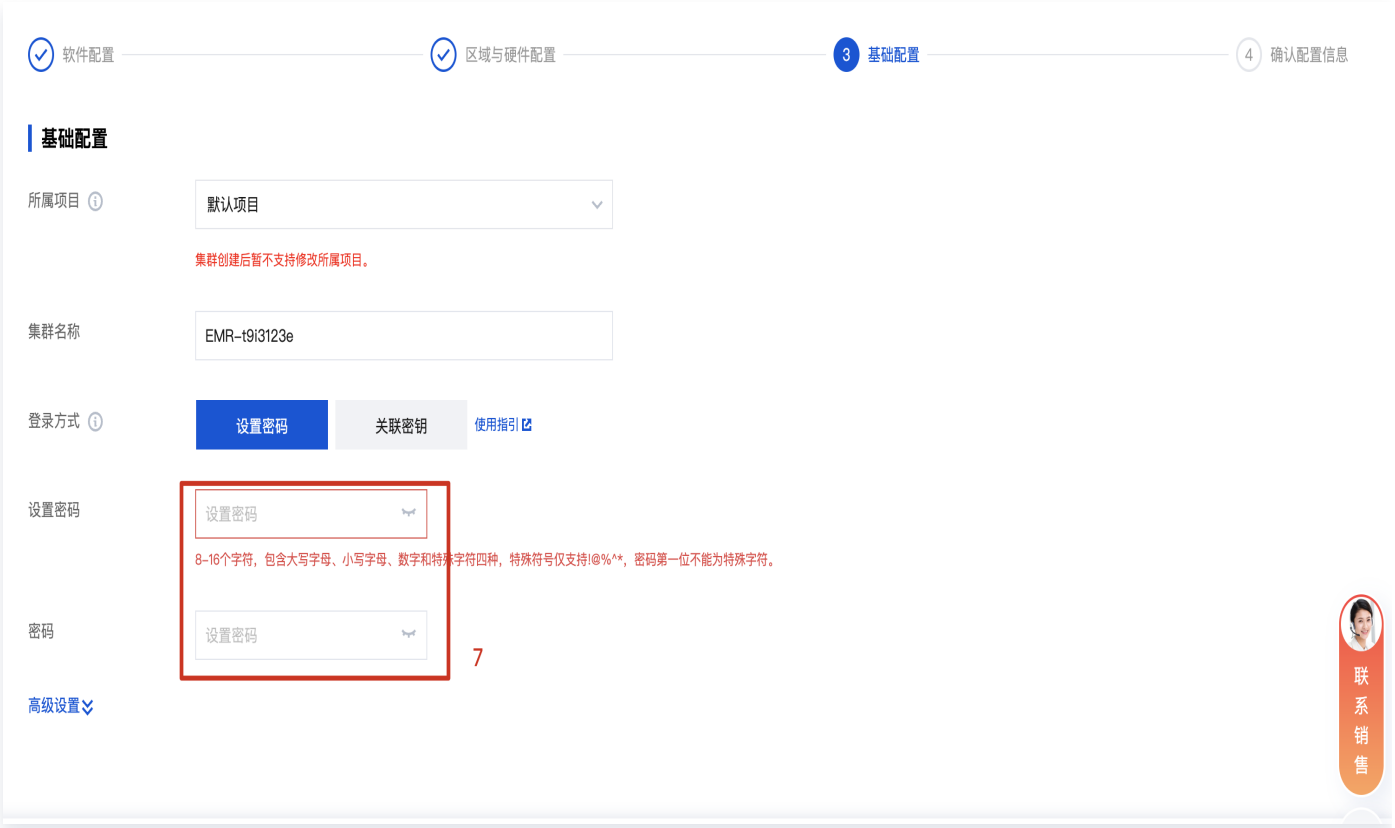
2. 第二步选择区域与硬件配置，选择完成后，单击**下一步**。
   - 集群网络：选择刚创建好的私有网络。
   - 可用区：选择私有网络中子网所在的可用区，如果此处没有，可返回 私有网络 页面，绑定子网。
   - 安全组：此处默认创建新的安全组即可。



3. 进入节点配置页面，您只需要展开明细，设置节点数量，按照默认的配置即可。

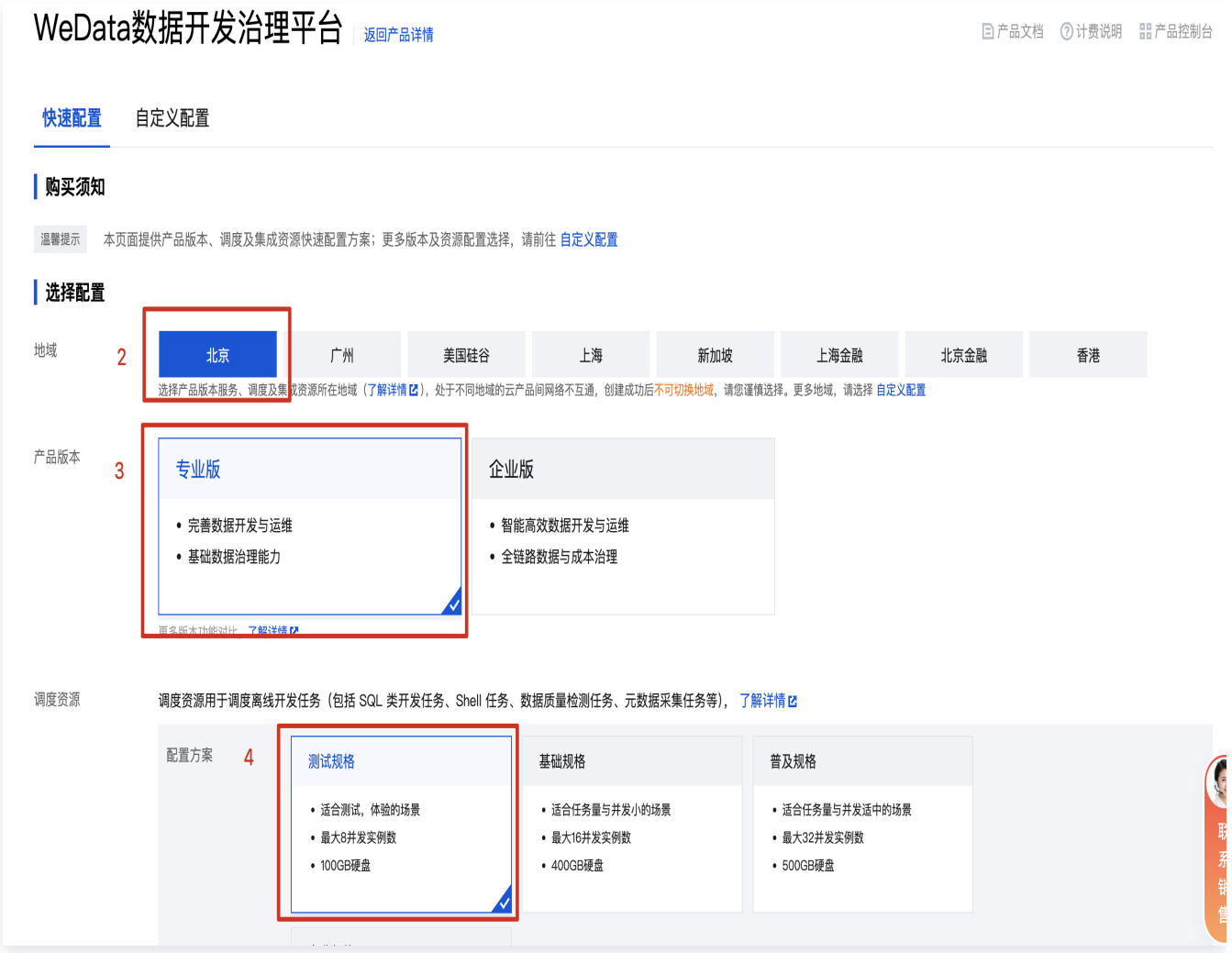


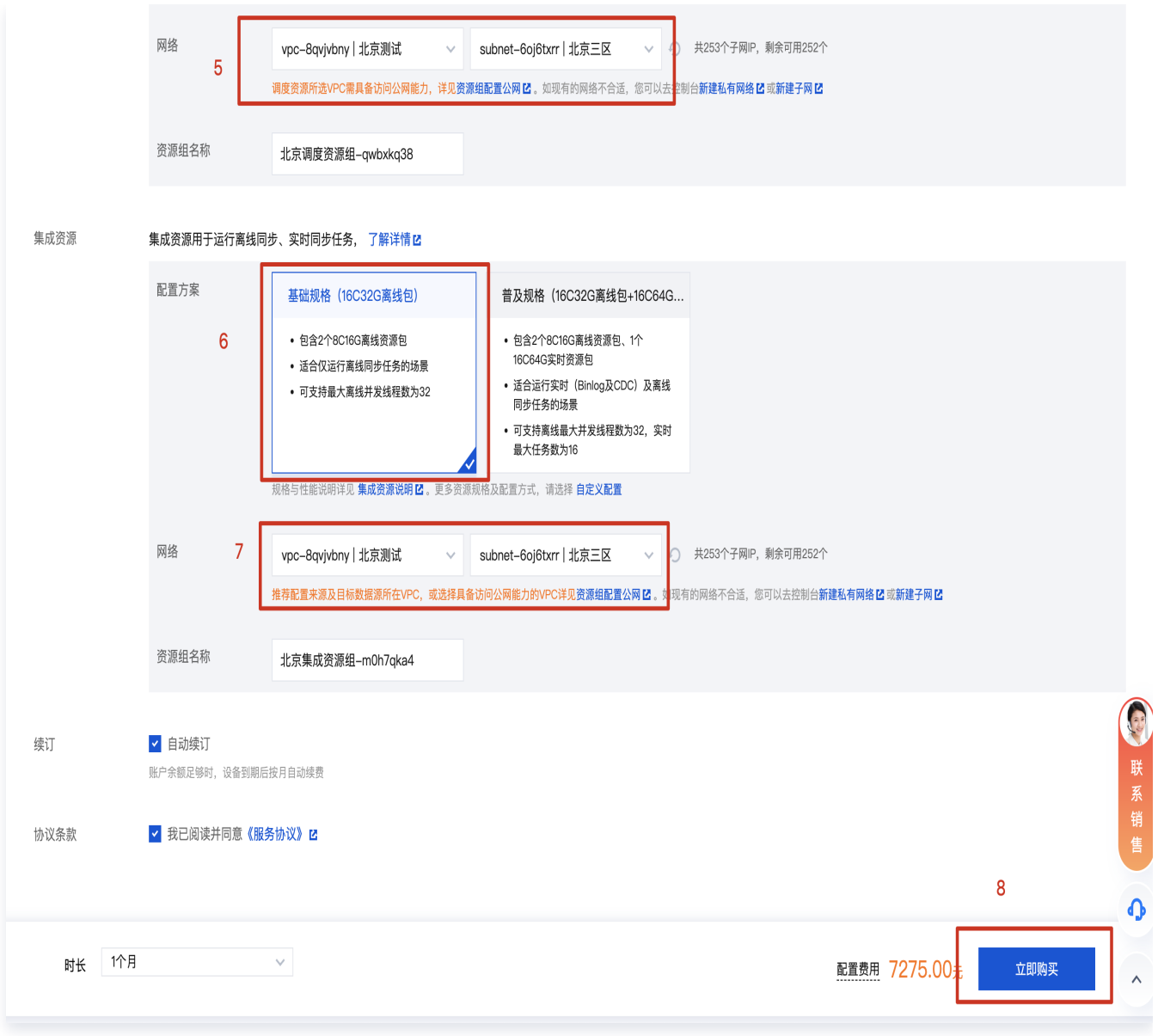
4. 进入基础配置界面，设置服务器密码，勾选**自动续费**和**协议条款**后，单击**立即购买**，核实账单后，付费开通即可。您可参考 密码设置格式 。



购买 WeData

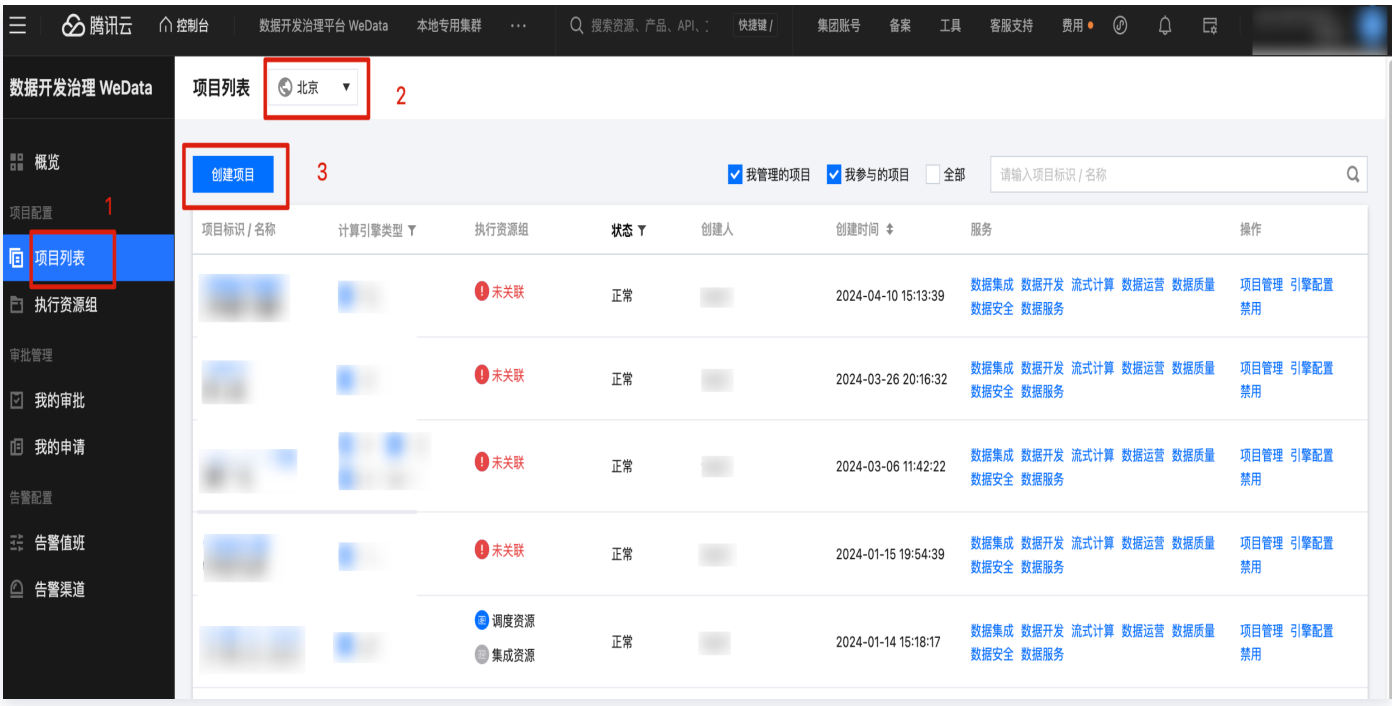
1. 进入腾讯云 数据开发治理平台 WeData 购买页，完成快速配置，单击**立即购买**，核实账单后，付费开通即可。
   - 地域：选择北京，建议可选择距离自己较近的地域。
   - 产品版本：选择专业版，了解版本详细内容，请参见 WeData 各版本区别。
   - 调度资源：选择测试规格，了解调度资源详细内容，请参见 调度资源计费说明。
   - 调度资源网络：选择刚创建好的私有网络。
   - 配置方案：选择基础规格，了解调度资源详细内容，请参见 集成资源计费说明。
   - 网络：选择刚创建好的私有网络。



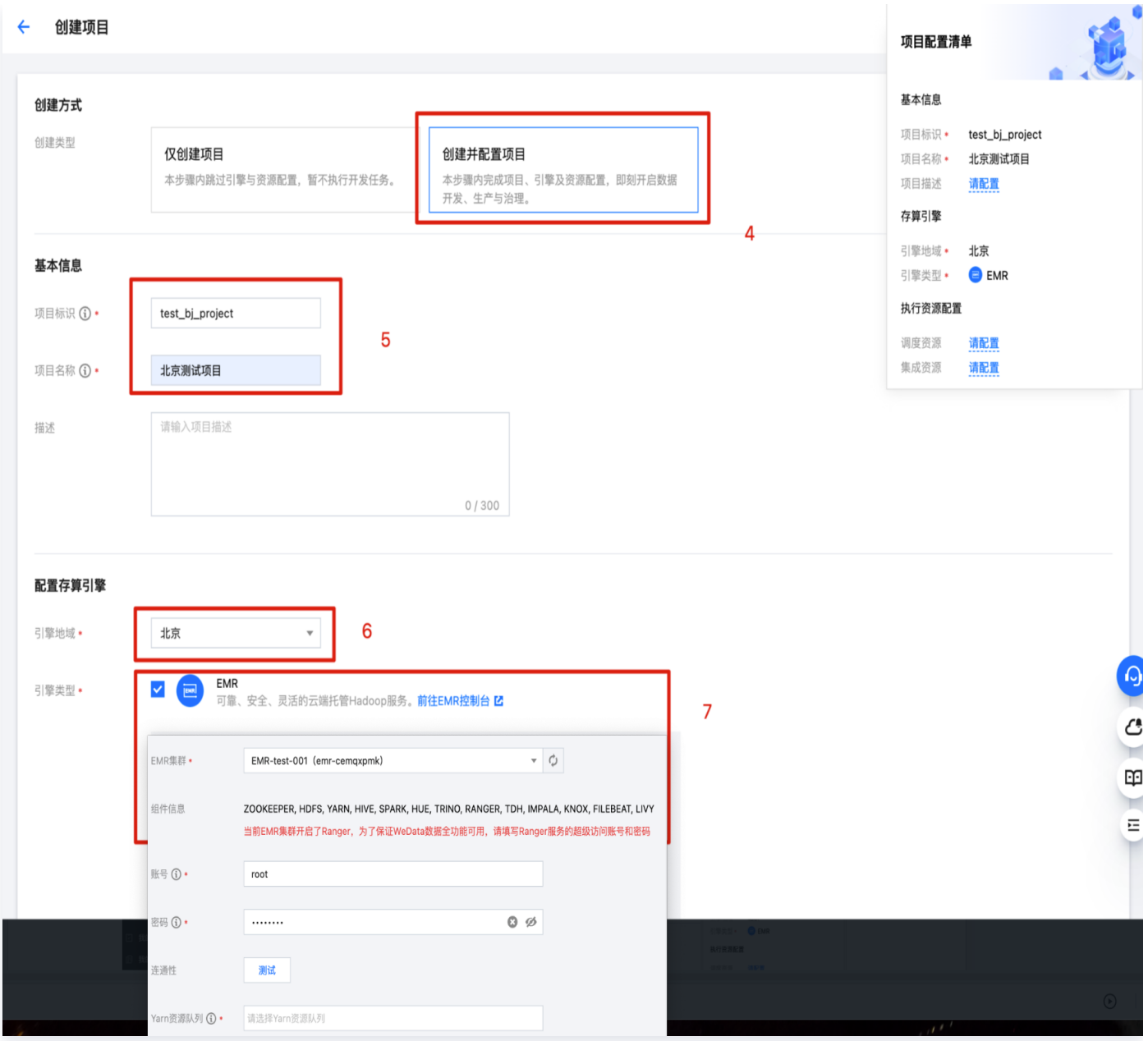


在 WeData 中创建项目

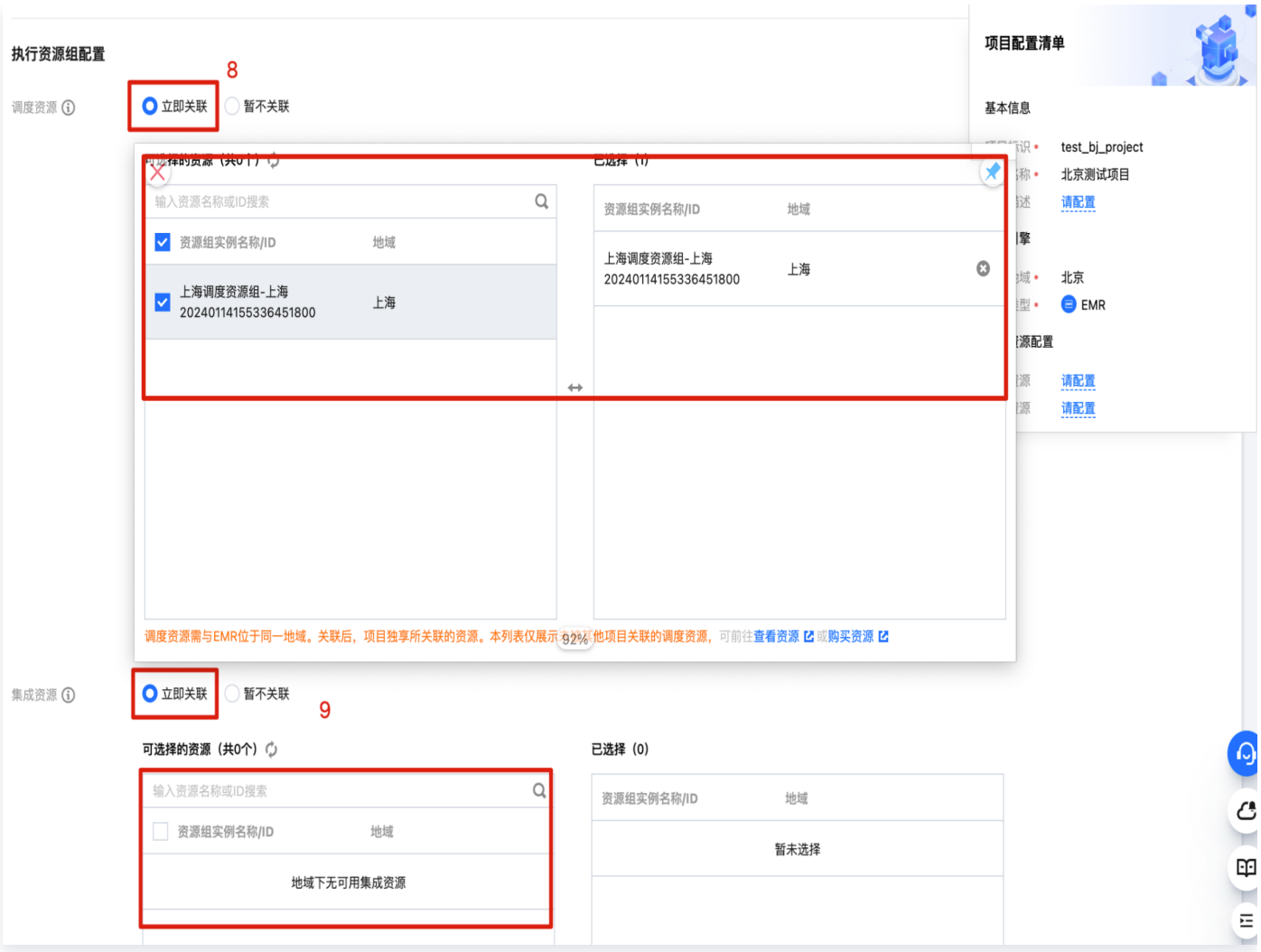
1. 登录腾讯云 数据开发治理平台 WeData 控制台，单击左侧左侧菜单**项目列表**，进入 项目列表 界面，选择顶部地域为**北京**，单击**创建项目**。



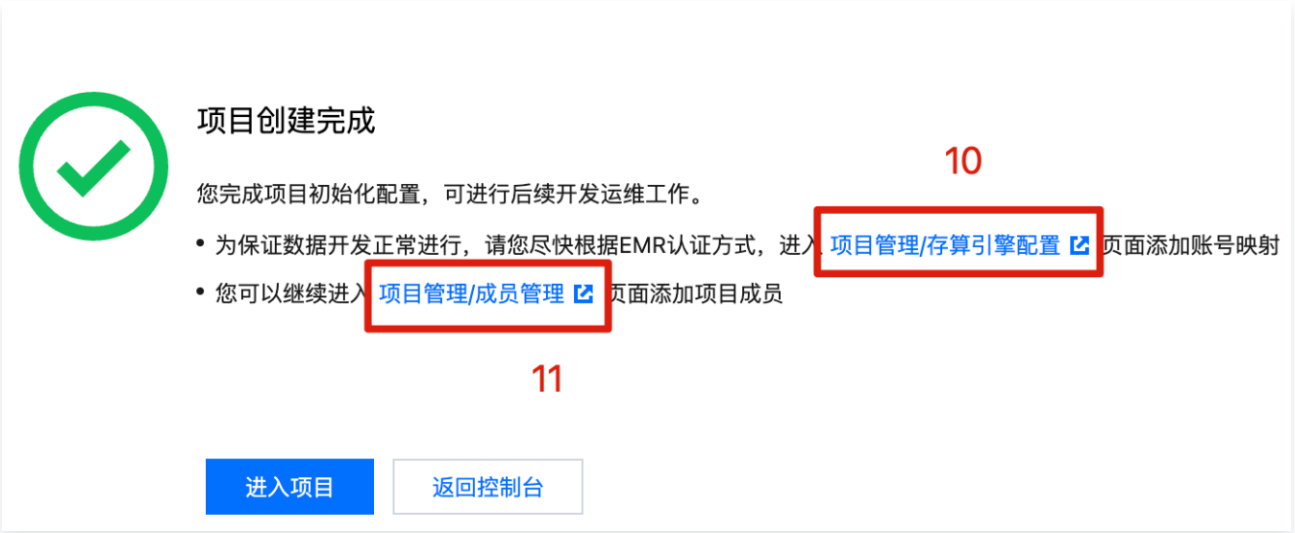
2. 进入创建项目界面，选择并填写相关信息后，单击**确认**，即完成项目创建。
   - **创建方式**：选择创建并配置项目。
   - **基本信息**：
     - 项目标识：任意填写，便于区分即可，示例：test_bj_project。
     - 项目名称：任意填写，便于区分即可，示例：北京测试项目。
   - **配置存算引擎**：引擎地域：选择北京。
   - **引擎类型**：选择 EMR。
     - EMR 集群：下拉绑定即可，如已经在2.3.1中完成购买，此处会展示 EMR 集群名称。
     - 账号：默认为 root。
     - 密码：在 购买 EMR 时设置的密码。
     - 连通性：请单击**测试**。
     - Yarn 资源队列：默认为 default。
     - 引擎元数据采集：是。



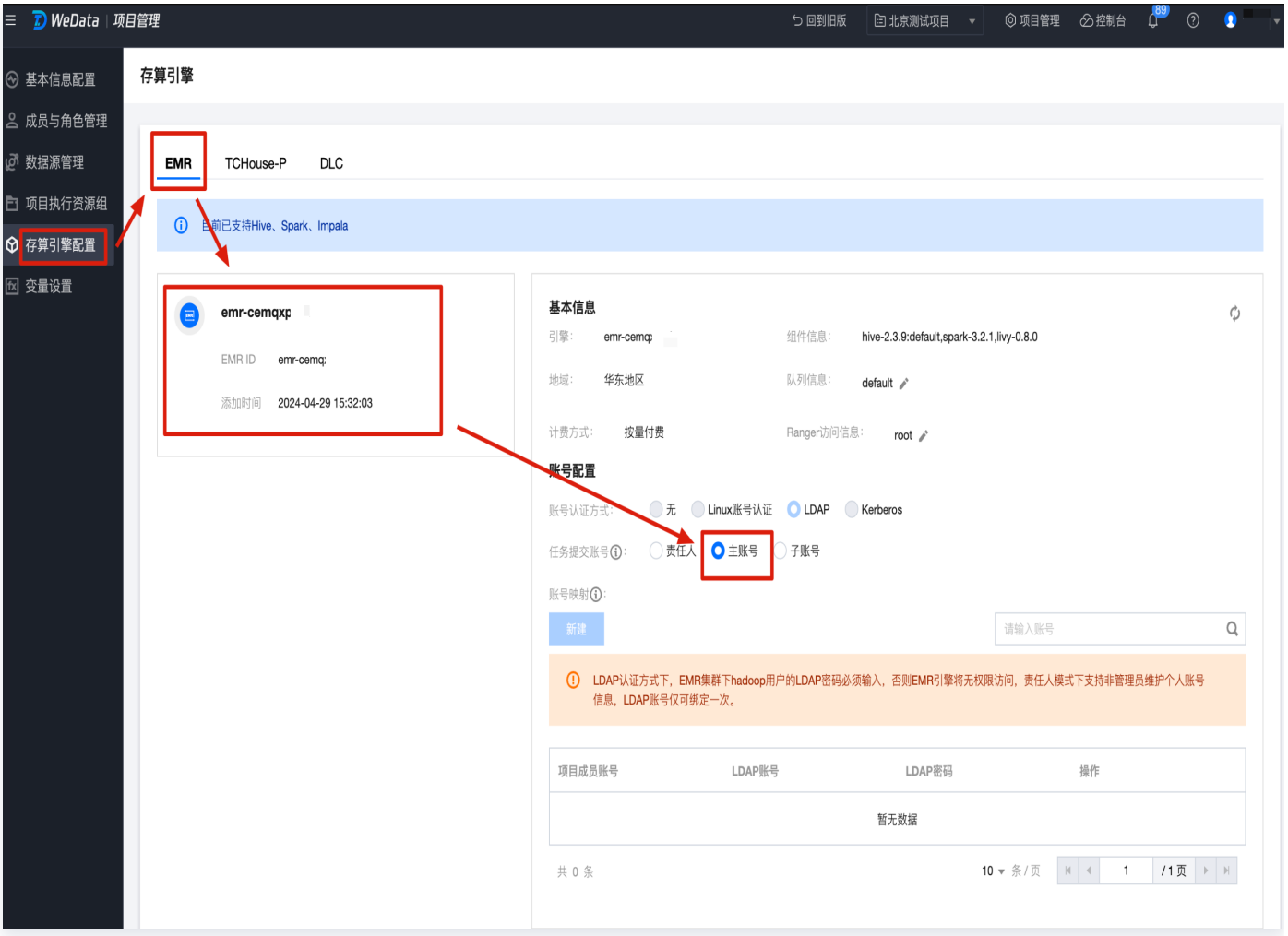
- **调度资源：选择立即关联，勾选之前创建的调度资源进行绑定，这里会展示可用资源组。**
- **集成资源：选择立即关联，勾选之前创建的集成资源进行绑定，这里会展示可用资源组。**



3. 项目创建完成后，您可以通过单击**项目管理/存算引擎配置**和**项目管理/成员管理**，进行账号配置和添加成员操作。

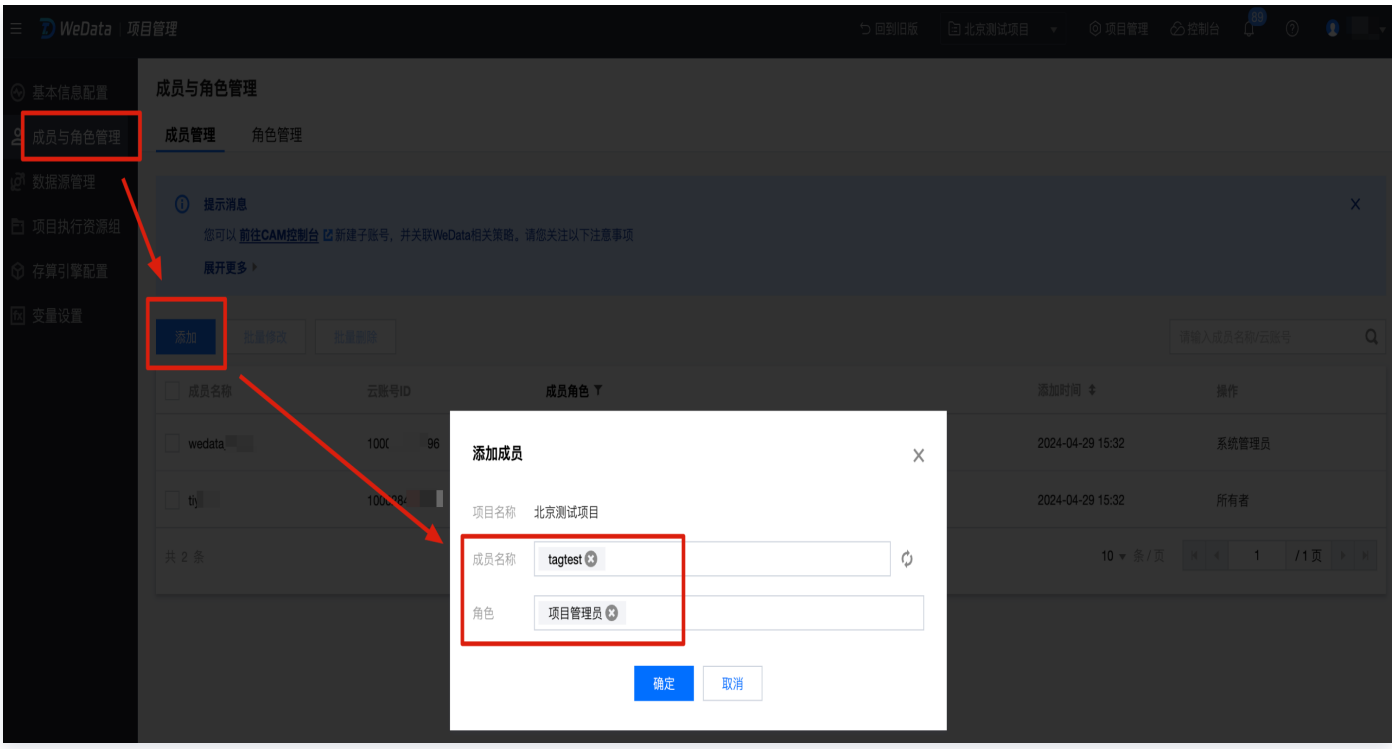


4. **存算引擎配置：**进入存算引擎配置界面，将 EMR 设置为**主账号**。



5. **成员配置：**在成员与角色管理界面，单击**添加**，进入添加成员页面，添加腾讯云子账号作为项目管理员。

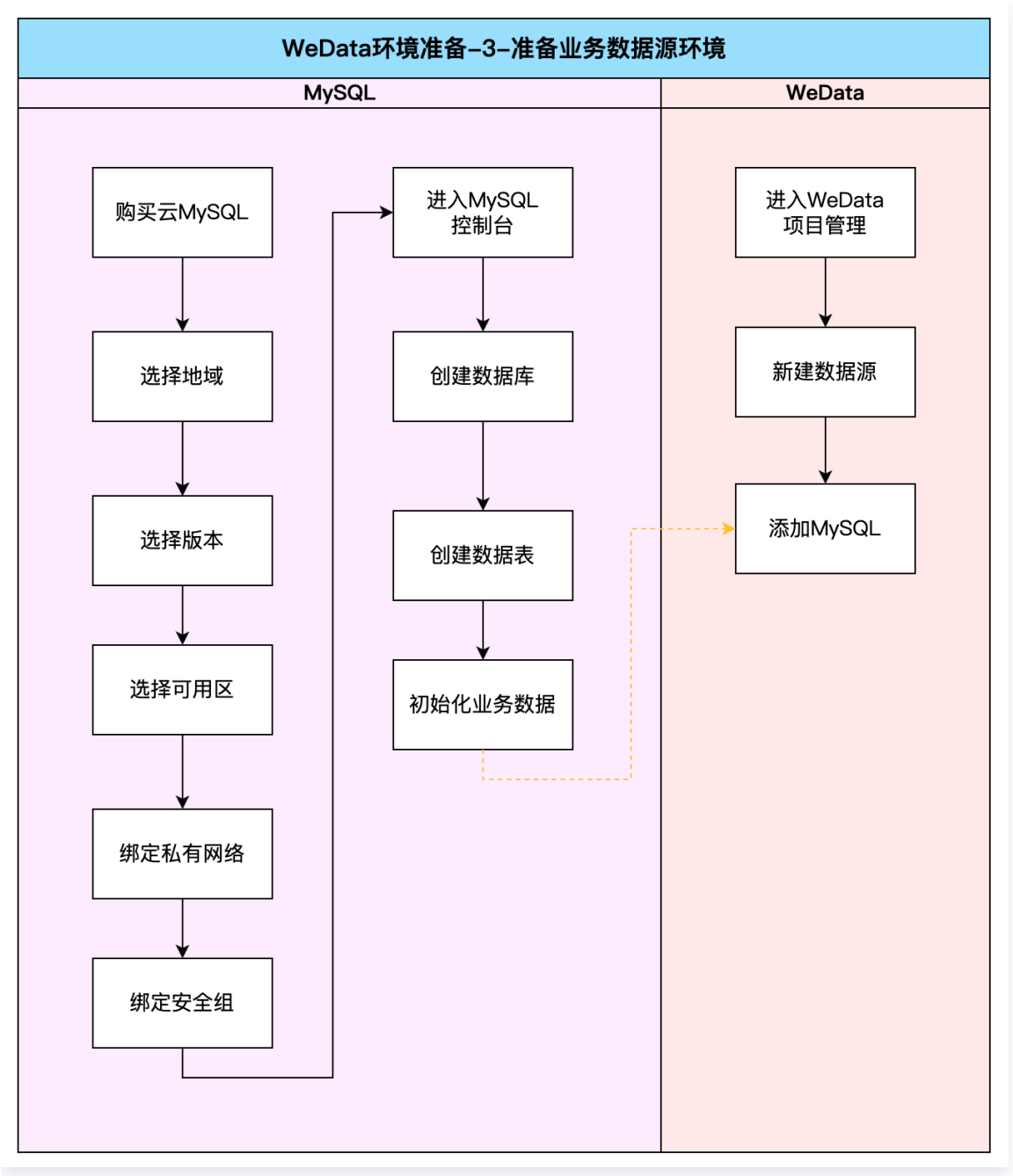
> **说明：**
> 此子账号可进行后续数据同步、数据开发操作。



准备业务数据资源环境

在本次教程中，我们模拟电子商城订单数据同步分析场景，因此需要准备电子商城原始数据。
本教程中采用腾讯云 MySQL 作为示例，介绍 WeData 的数据同步过程，因此，我们需要先在腾讯云上购买一套 MySQL 数据库。

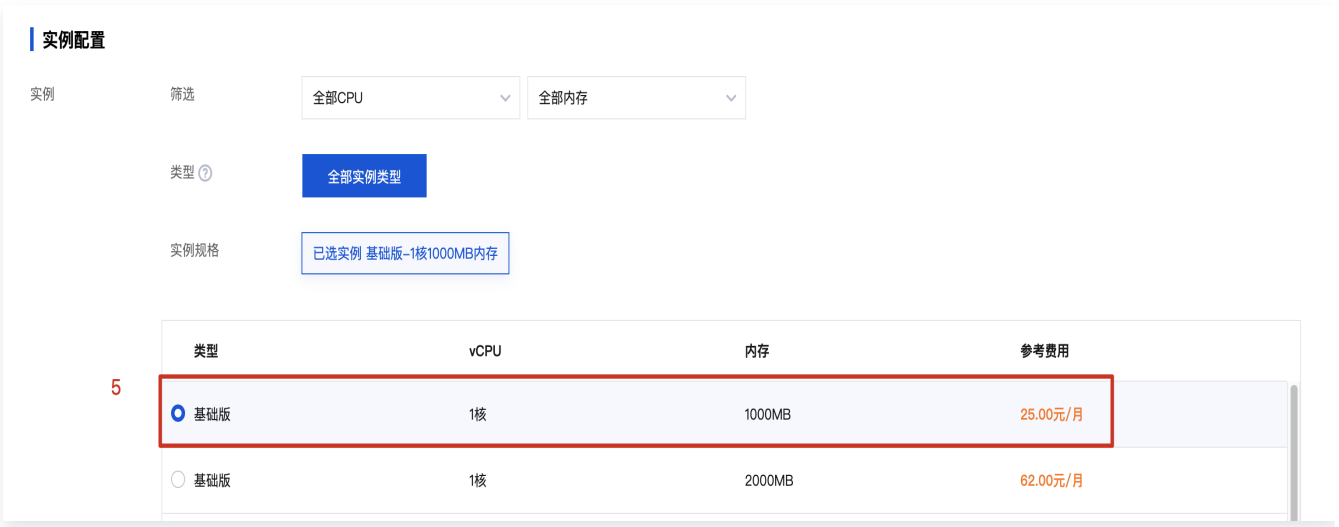
- **操作角色：**企业管理员。
- **操作账号：**腾讯云主账号。
- **操作流程：**



购买 MySQL

1. 进入云数据库 TencentDB for MySQL 购买页，完成快速配置，单击**立即购买**，核实账单后，付费开通即可。
   - 地域：选择北京（仅作为示例，可选择距离自己较近的地域）。
   - 架构：选择**单节点**。
   - 可用区：选择**北京三区**，选择私有网络中子网所在的可用区。

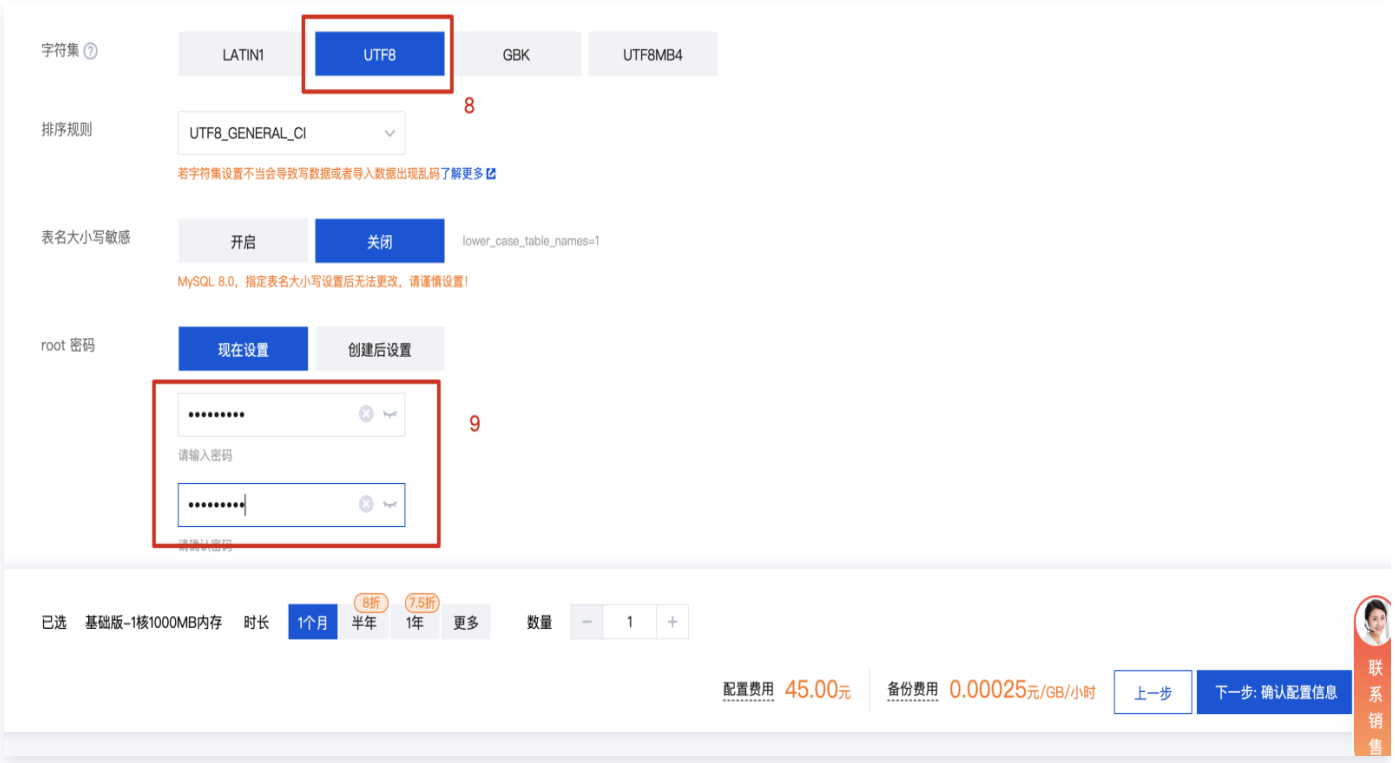
- 实例规格：选择**基础版**。



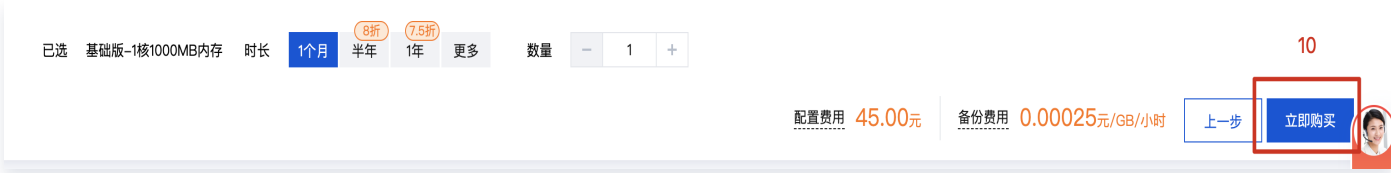
- 选择网络：选择在上文中创建好的私有网络。
- 选择安全组：选择在上文中默认创建的安全组。

- 字符集：选择 **UTF8**。
- root 密码：设置 root 用户密码。

2. 设置完成后，单击**下一步，确认配置信息**。



3. 单击**立即购买**，核实账单后，付费开通即可。



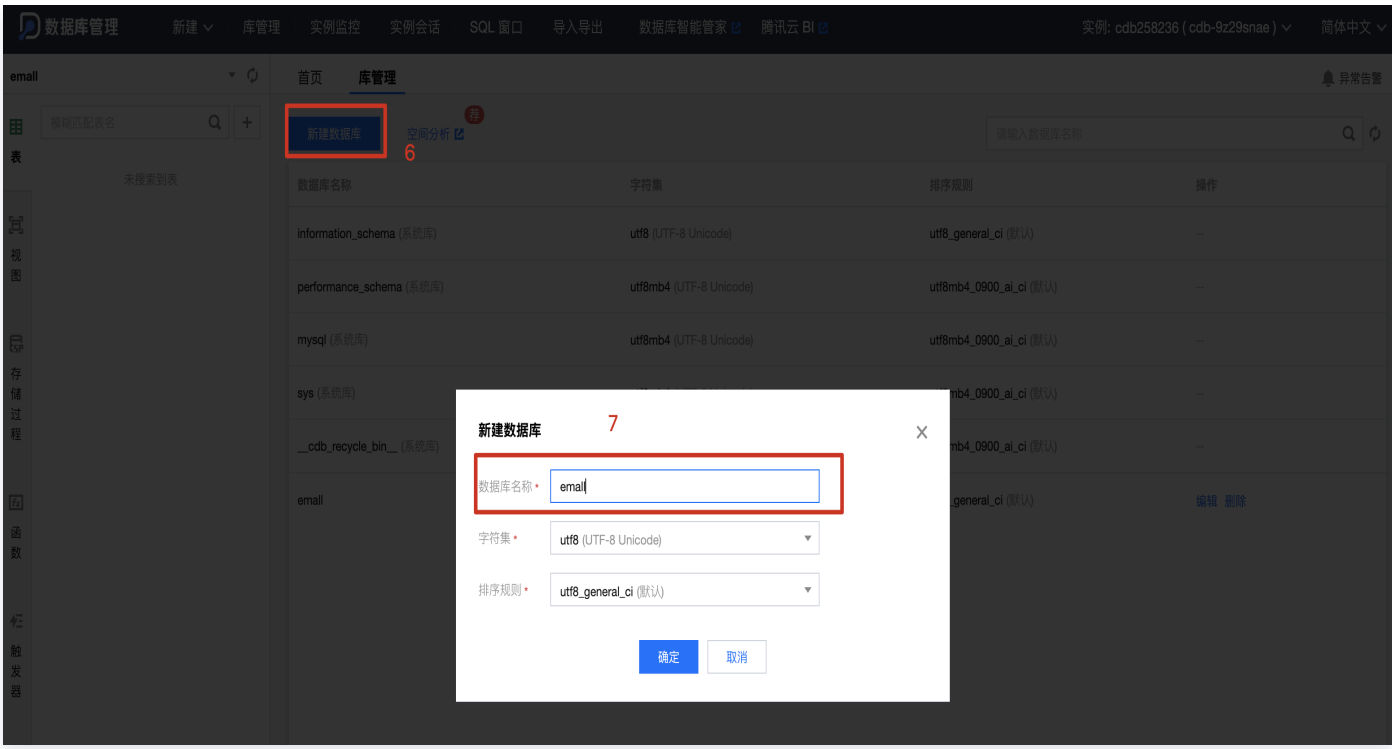
在 MySQL 中初始化业务数据

1. 进入腾讯云 数据库 MySQL 控制台，单击左侧菜单**实例列表**，进入 MySQL 实例列表界面。在顶部选择地域为**北京**，单击**登录**。

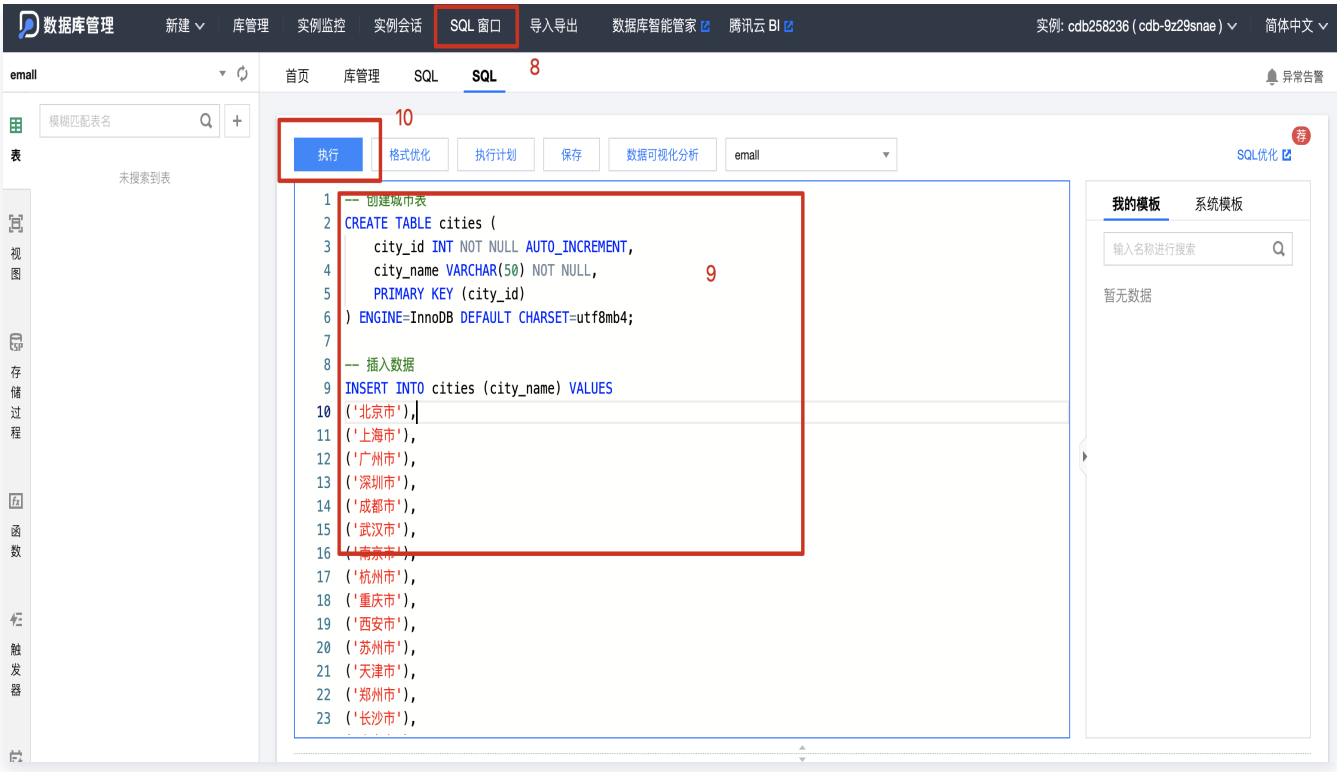
2. 进入数据库管理登录界面，输入账号和密码后，单击**登录**。

3. 在数据库管理界面，选择顶部菜单**新建 > 新建库**。

4. 进入新建库界面，单击**新建数据库**，进入新建数据库界面，填写数据库名称，建议填写 emall，填写完成后，单击**确定**。



5. 在数据库管理界面，单击顶部菜单 **SQL 窗口 > SQL**，进入SQL界面，通过执行 SQL 语句快速建表。
   - 依次复制下面的建表 SQL 语句，每复制一次 SQL 语句，单击一次**执行**。执行后，清空 SQL 内容再复制下一个建表语句。



   - 具体建表语句如下：
     - **创建城市表（cities）**

```
-- 在MySQL中创建城市表
CREATE TABLE cities (
    city_id INT NOT NULL AUTO_INCREMENT,
    city_name VARCHAR(50) NOT NULL,
    PRIMARY KEY (city_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- 插入数据
INSERT INTO cities (city_name) VALUES
('北京市'),
('上海市'),
('广州市'),
('深圳市'),
('成都市'),
('武汉市'),
('南京市'),
('杭州市'),
('重庆市'),
('西安市'),
('苏州市'),
('天津市'),
('郑州市'),
('长沙市'),
('青岛市'),
('沈阳市');
```

- **创建商品品类表（categories）**

```
-- 创建商品品类表
CREATE TABLE categories (
    category_id INT NOT NULL AUTO_INCREMENT,
    category_name VARCHAR(50) NOT NULL,
    PRIMARY KEY (category_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--插入数据
INSERT INTO categories (category_name) VALUES
('电子产品'),
('家用电器'),
('服装鞋帽'),
('食品饮料'),
('图书音像'),
```

```
('运动户外'),
('家居建材'),
('母婴用品'),
('汽车用品');
```

- **创建商品表（products）**

```
-- 创建商品表
CREATE TABLE products (
    product_id INT NOT NULL AUTO_INCREMENT,
    category_id INT NOT NULL,
    product_name VARCHAR(100) NOT NULL,
    PRIMARY KEY (product_id),
    FOREIGN KEY (category_id) REFERENCES
categories(category_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- 插入数据
INSERT INTO products (category_id, product_name) VALUES
(1, '智能手机'),
(1, '笔记本电脑'),
(1, '平板电脑'),
(2, '空调'),
(2, '洗衣机'),
(3, '男士外套'),
(3, '女士裙子'),
(4, '碳酸饮料'),
(4, '矿泉水'),
(5, '现代小说'),
(5, '历史书籍'),
(6, '跑步鞋'),
(6, '瑜伽垫'),
(7, '实木家具'),
(7, '床上用品'),
(8, '婴儿奶粉'),
(8, '儿童玩具');
```

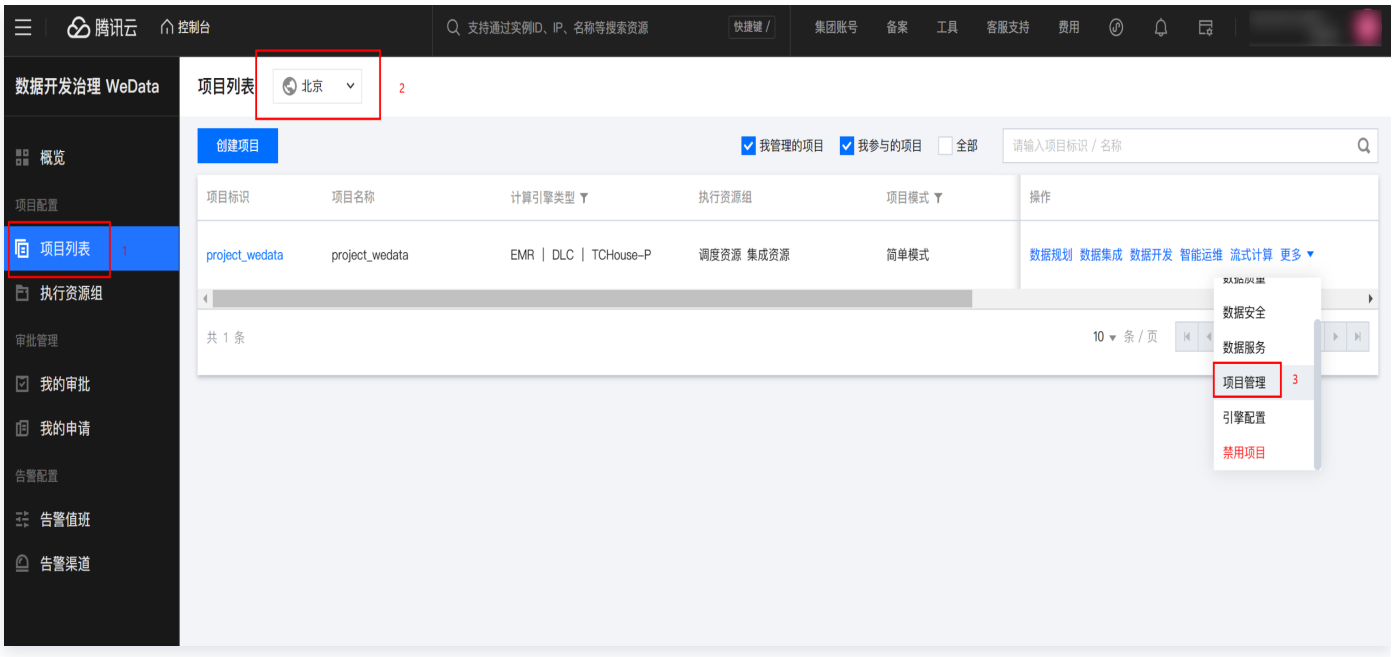
- **创建订单表（orders）**

```
-- 创建订单表
CREATE TABLE orders (
    order_id INT NOT NULL AUTO_INCREMENT,
```

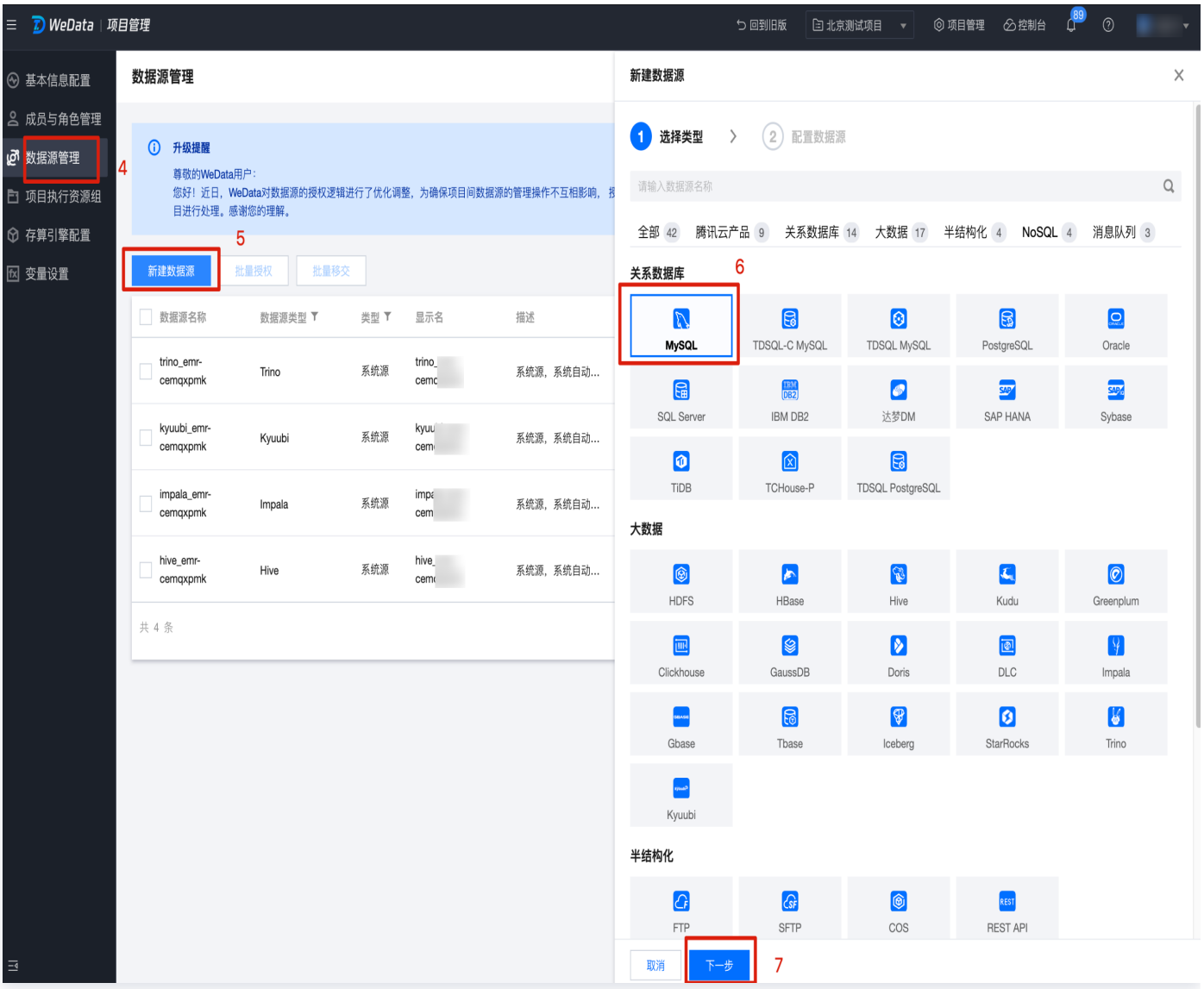
```
    product_id INT NOT NULL,
    quantity INT NOT NULL CHECK (quantity > 0),
    unit_price DECIMAL(10, 2) NOT NULL,
    amount DECIMAL(10, 2) NOT NULL,
    order_time DATETIME NOT NULL,
    shipping_city_id INT NOT NULL,
    shipping_address TEXT NOT NULL,
    PRIMARY KEY (order_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id),
    FOREIGN KEY (shipping_city_id) REFERENCES cities(city_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- 插入数据
INSERT INTO orders (product_id, quantity, unit_price, amount,
order_time, shipping_city_id, shipping_address) VALUES
(1, 1, 4999.00, 4999.00, '2024-04-01 10:00:00', 1, '北京市海淀区
某小区'),
(2, 1, 6999.00, 6999.00, '2024-04-02 11:00:00', 2, '上海市浦东新
区某小区'),
(3, 2, 3999.00, 7998.00, '2024-04-03 12:00:00', 3, '广州市天河区
某小区'),
(4, 1, 5999.00, 5999.00, '2024-04-04 13:00:00', 4, '深圳市南山区
某小区'),
(5, 1, 999.00, 999.00, '2024-04-05 14:00:00', 5, '成都市武侯区某
小区'),
(6, 1, 699.00, 699.00, '2024-04-06 15:00:00', 6, '武汉市江汉区某
小区'),
(7, 1, 2999.00, 2999.00, '2024-04-07 16:00:00', 7, '南京市鼓楼区
某小区'),
(8, 1, 3999.00, 3999.00, '2024-04-08 17:00:00', 8, '杭州市西湖区
某小区'),
(9, 1, 4999.00, 4999.00, '2024-04-09 18:00:00', 9, '重庆市渝中区
某小区'),
(10, 1, 1999.00, 1999.00, '2024-04-10 19:00:00', 10, '西安市碑林
区某小区');
```

在 WeData 中绑定 MySQL

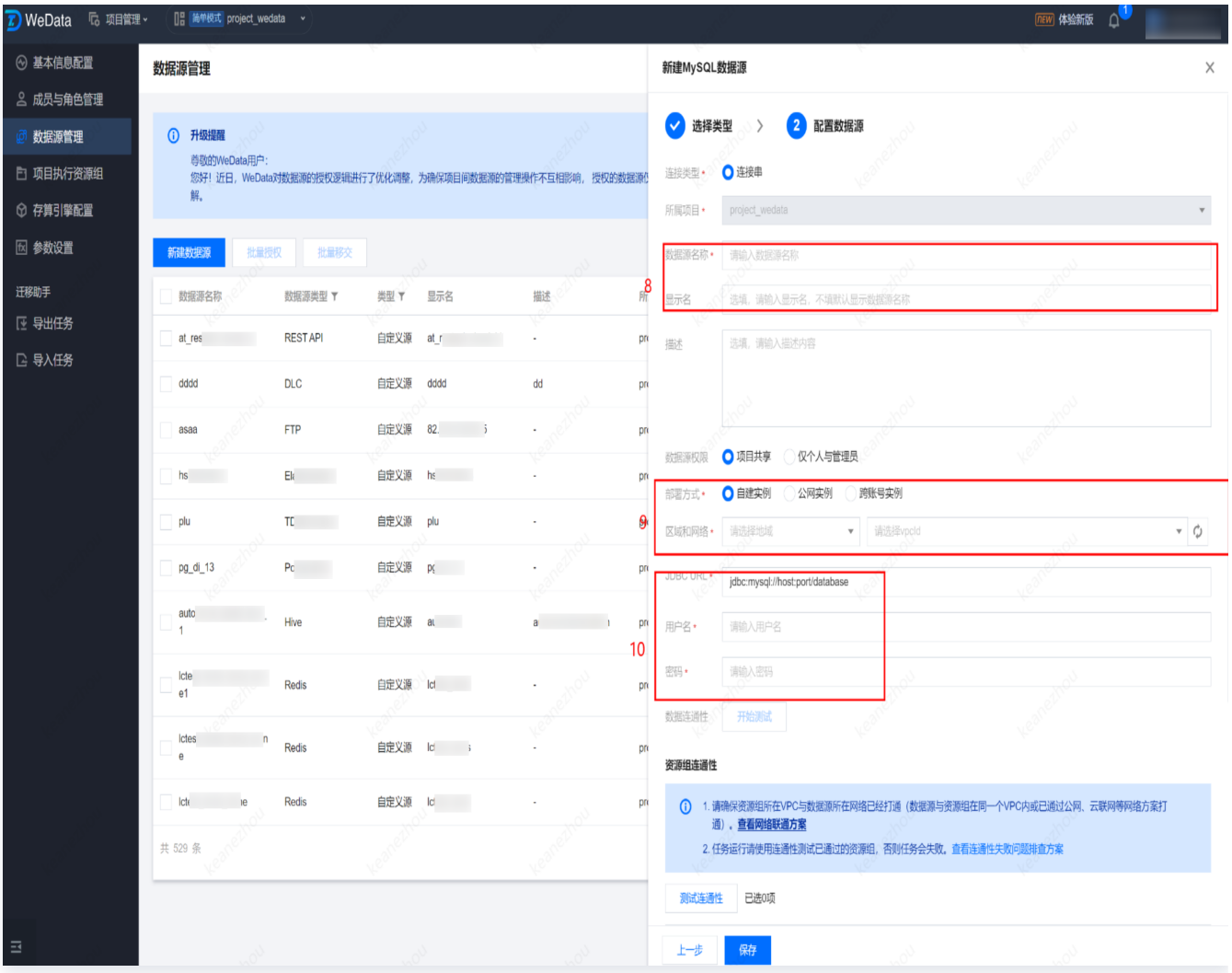
1. 登录腾讯云 数据开发治理平台 WeData 控制台，单击左侧菜单**项目列表**，选择顶部地域为北京，在对应的项目操作栏，单击**项目管理**。



2. 在数据源管理界面，单击**新增数据源**，选择数据源类型为 MySQL，单击**下一步**。



3. 进入新建 MySQL 数据源界面，填写相关信息后，单击**保存**。
   - 数据源名称：
     - 数据源名称：任意填写，便于区分即可，示例：bj_mall。
     - 显示名：任意填写，便于区分即可，示例：北京-测试-商城。
   - 实例信息：
     - 地域：选择**北京**。
     - 选择实例：下拉选择即可。
   - 数据表连接信息：
     - 数据库名称：emall
     - 用户名：root
     - 密码：填写上文中购买 MySQL 设置的密码。



至此所有准备工作均已完成，下面正式开始本教程的数据同步与数据开发部分。后续操作内容可使用腾讯云子账号进行操作。

数据表结构设计

最近更新时间：2024-07-19 11:17:21

业务调研

源数据存储位置

通过调研商城系统的原有的技术架构，了解到数据存储在 MySQL 数据库。
此处假设使用腾讯云 MySQL 数据库。

目标业务场景分析

通过分析目标业务场景：各城市、各品类的销售情况，因此我们需要获取以下几张表：

- 订单表：（此处先忽略订单明细等子表设计，假设订单表包含商品 ID、商品数量、商品价格、收货地址、下单时间等信息）。
- 商品表：（此处先忽略 SKU 等子表设计，假设商品表包含商品 ID、商品品类等信息）。
- 城市表：（假设地理位置编码表仅到城市级别，城市表包含：城市编码、城市名称）。
- 商品品类表：（假设品类仅存在一级类目，类目表包含：品类编码、品类名称）。

实际结构

以下为调研到的订单表与商品表的实际结构：

1. 订单表（orders）

字段名	字段类型	字段长度	字段说明	示例
order_id	INT	10	订单 ID，主键，自增	10001
product_id	INT	10	商品 ID，外键	1001
quantity	INT	5	商品数量，正整数	2
unit_price	DECIMAL(10,2)	-	商品单价，保留两位小数	99.99
amount	DECIMAL(10,2)	-	商品金额小计，即商品数量乘以单价	199.98
order_time	DATETIME	-	下单时间，记录精确到分钟	"2024-04-04 10:30:00"
shipping_city	INT	10	收货地址城市 ID，外键	1101

_id				
shipping_address	TEXT	–	收货地址，包含省、市、区、详细地址	"北京市朝阳区 XXX 小区"

2. 商品表（products）

字段编码	字段类型	字段长度	字段说明	示例
product_id	INT	10	商品 ID，主键，自增	1001
category_id	INT	10	商品品类 ID，外键	101
product_name	VARCHAR(100)	–	商品名称	"智能手机"

3. 城市表（cities）

字段编码	字段类型	字段长度	字段说明	示例
city_id	INT	10	城市编码，主键，自增	1101
city_name	VARCHAR(50)	–	城市名称	"北京市"

商品品类表（categories）

字段编码	字段类型	字段长度	字段说明	示例
category_id	INT	10	品类ID，主键，自增	1
category_name	VARCHAR(50)	–	品类名称	"电子产品"

结构设计

根据业务场景需要，下面按照最终业务输出，涉及数仓分层和数据表结构。

模型规范

模型规范可以帮助团队统一数据仓库设计规则，统一数据开发过程，更好地沉淀数据资产，为建设数据服务与数据集市打下基础。
在数仓模型规范设计过程中，会包含以下多个类目，如数据域、主体域等。
在此场景中，核心目的为数据集成与数据开发过程，因此在此教程中不做数据模型规范的详细教学。
以下为本场景的相关模型规范示例：

类目	中文描述	英文名称

业务分类	销售	trade
数据域	订单 商品	order product
业务过程	订单创建	ordercreate
主题域	商品	product
维度	日期 城市 品类	date city category
指标	销售额 销售数量	amount quantity

数仓分层

1. 数据引入层 ODS

将没有经过任何处理的原始数据导入到数据仓库。ODS 层的数据表结构与原始数据所在的数据系统中的表结构一致。
因此，我们需要根据原始数据创建4张hive表（此处不需要建表操作，后续教学中会有涉及），表结构与 MySQL 源数据表完全相同。
以下为四张表的命名：

- 订单表：ods_order_order
- 商品表：ods_product_product
- 类目表：ods_product_category
- 城市表：ods_order_city

> **说明：**
> 建议命名格式为：ods_{数据域}_{自定义内容}。

2. 公共维度层 DIM

此处重点介绍数据同步逻辑，暂时忽略维度层设计。

> **说明：**
> 建议将维度表中的字段属性将冗余到明细数据表中。

3. 明细数据层 DWD

构建最细颗粒度的明细数据表，在此表中可以适当冗余一些字段，减少明细数据表与维度表的关联。

- 构建明细表：此处不需要建表操作，后续教学中会有涉及。
- 商品销售情况明细表：dwd_trade_order_ordercreate_productsales。

> **说明：**
> 建议命名格式为：dwd_{业务分类}_{数据域}_{业务过程}_{自定义内容}。

字段编码	字段类型	字段长度	字段说明	示例
order_id	INT	10	订单 ID，主键	10001
product_id	INT	10	商品 ID	1001
category_id	INT	10	商品品类 ID	101
category_name	STRING	50	商品品类名称	"电子产品"
product_name	STRING	50	商品名称	"智能手机"
quantity	INT	5	商品数量，正整数	2
unit_price	DECIMAL	10,2	商品单价，保留两位小数	99.99
amount	DECIMAL	10,2	商品金额小计，即商品数量乘以单价	199.98
order_time	DATETIME	-	下单时间，精确到分钟	"2024-04-04 10:30:00"
shipping_city_id	INT	10	收货地址城市 ID，外键	1101
shipping_city_name	STRING	50	城市名称	"北京市"
shipping_address	TEXT	-	收货地址，包含省、市、区、详细地址	"北京市朝阳区 XXX 小区"
pt_date	STRING	50	分区字段	"2024-04-01"

补充说明：Hive 表分区

什么是分区

分区是一种重要的数据库优化技术，它通过将数据集划分为更小的、逻辑上独立的部分，来提高性能、简化管理、降低成本，并提高数据的可用性和安全性。

尤其在大数据场景下，对表设置分区尤其重要。

Hive 表分区存储的好处

对 Hive 进行分区存储的好处可体现以下多个方面：

优点	说明
提高查询性能	通过将数据分散存储在不同的分区中，查询时可以针对特定分区进行，避免了扫描整个表的数据，从而显著减少了查询时间。
优化数据管理	分区是一种逻辑上的数据组织方式，便于进行数据维护、清理和批量操作，如备份和恢复。
水平扩展	分区可以水平分散数据存储压力，将数据物理上分布到不同的存储单元，提高了系统的扩展性。
减少数据倾斜	在数据量不均匀的情况下，分区可以避免数据倾斜问题，即避免某些分区的数据量过大，而其他分区数据量过小。
数据隔离	分区可以用于数据隔离，例如，可以根据时间将数据划分为不同的分区，便于实现数据的版本控制和历史数据的管理。
减少数据加载时间	在数据加载或 ETL 过程中，可以更快地将数据加载到特定的分区，而不需要对整个表进行操作。
节省存储空间	通过分区可以删除或归档旧的分区数据，从而节省存储空间。
并行处理	分区表可以更好地利用 Hadoop 的 MapReduce 并行处理能力，因为查询可以并行地在不同的分区上执行。
数据安全和访问控制	分区可以用于实现更细粒度的数据访问控制，例如，可以对某些分区设置更严格的访问权限。
维护数据完整性	分区可以确保数据的完整性，因为每个分区可以有自己的数据完整性约束。
支持数据的冷热分层	通过分区，可以根据数据的使用频率将其分为“热数据”和“冷数据”，并采取不同的存储策略。
简化数据 ETL 过程	在数据抽取、转换和加载过程中，分区可以简化数据的组织和处理流程。
提高数据可用性	分区可以提高数据的可用性，因为即使某个分区不可用，也不影响对其他分区的访问。

因此，在创建 Hive 表时尽量在建立之初就规划好分区字段。

4. 汇总数据层 DWS

构建可供业务使用粒度的汇总指标数据表。

- 构建汇总表：此处不需要建表操作，后续教学中会有涉及。
- 商品销售情况每日汇总表：dws_trade_order_productsales_1d。
- 建议命名格式为：dws_{业务分类}_{数据域}_{自定义内容}_{时间周期}。

字段编码	字段类型	字段长度	字段说明	示例
order_date	DATE	-	日期	2021-04-01
city_id	INT	10	城市 ID	1
category_id	INT	10	商品品类 ID	1
city_name	STRING	50	城市名称	"北京"
category_name	STRING	50	商品品类名称	"电子产品"
quantity	INT	10	商品总销量	100
amount	DECIMAL	(10, 2)	商品总销售额	9999.99
pt_date	STRING	50	分区字段	"2021-04-01"

5. 应用数据层 ADS

构建面向最终业务分析需求的指标表，由于此场景比较简单，此处暂时忽略。

数据集成

最近更新时间：2024-12-31 14:43:42

我们将在此步骤中，将原始数据同步到数仓中。

新增数据源

原始数据源：MySQL

我们已将数据源绑定到了项目中，此处可忽略。

目标数据源：Hive

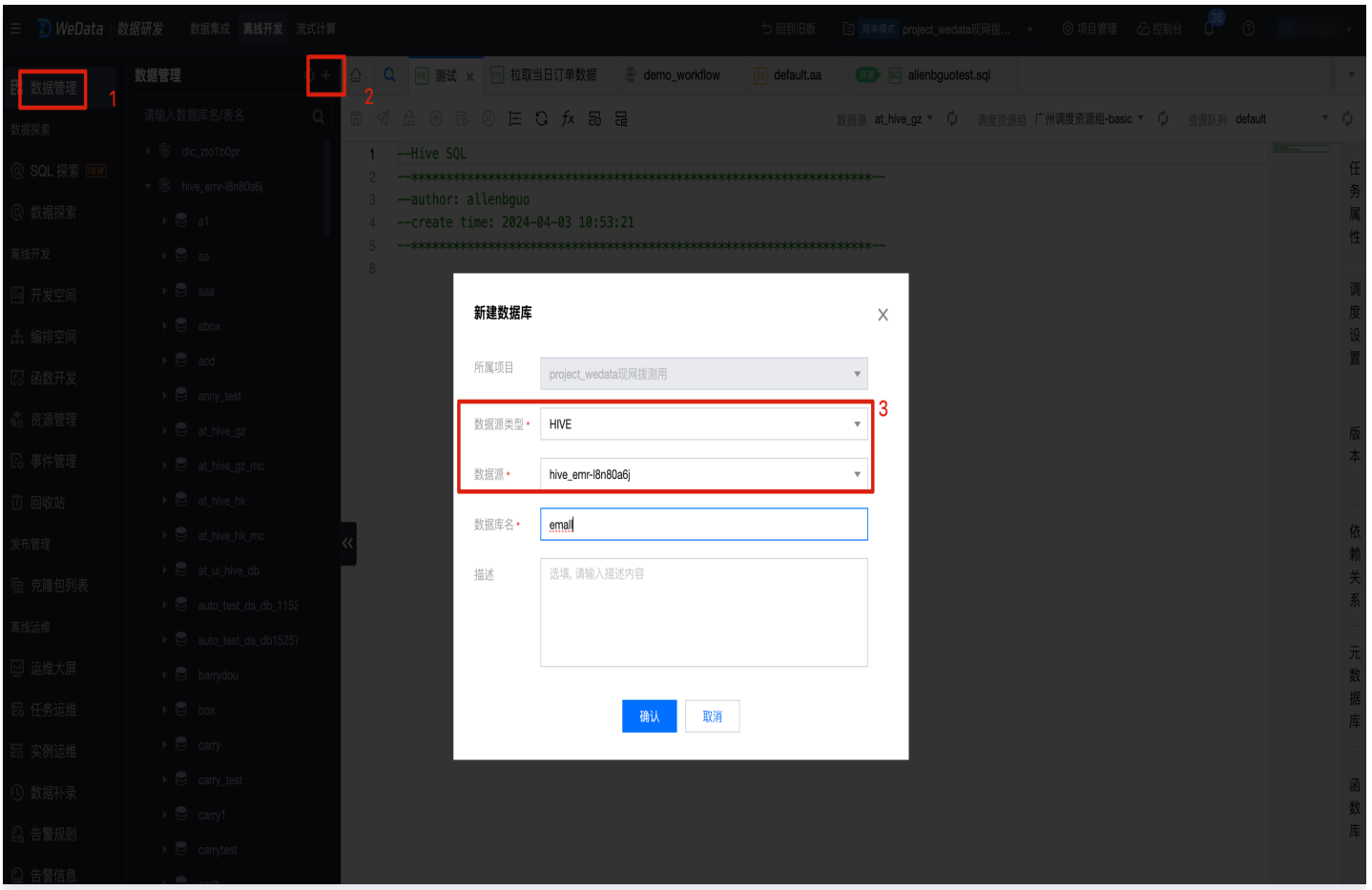
在您绑定存算引擎 EMR 后，系统将在10分钟内采集 EMR 集群中的 Hive 数据源。因此不需要您主动绑定 Hive 数据源。
但是我们需要在 Hive 数据源中**创建数据库**，存储采集后的原始数据。

新建数据库

进入**数据开发** > **数据管理**，单击 + 号新建数据库，选择并填写所需内容，填写完成后，单击**确认**。

- 数据源类型选择：Hive。
- 数据源选择：hive_emr-XXX。

> **说明：**
> 在您绑定存算引擎 EMR 后，系统将在10分钟内采集 EMR 集群中的 Hive 数据源。

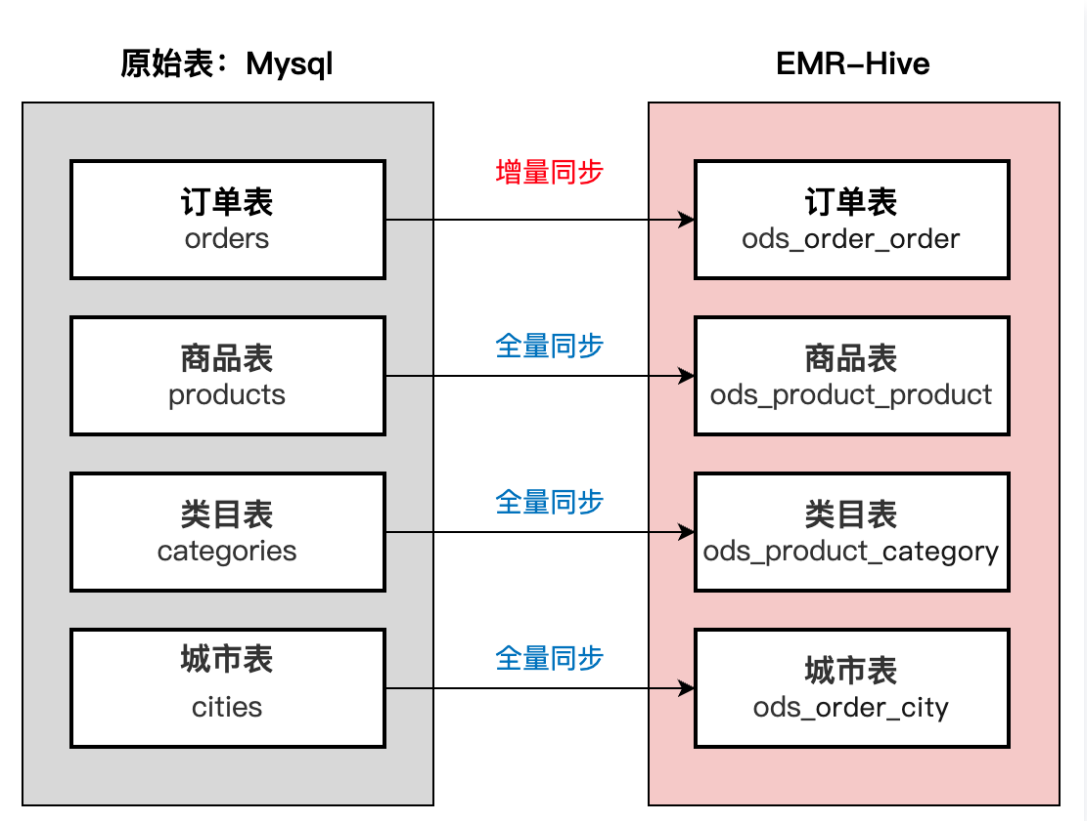


离线同步任务设计

现在我们将创建离线同步任务，将 MySQL 数据源中的原始数据，同步至 EMR 集群中的 Hive 表中。
从上面的操作已经知道，我们需要同步4张原始数据表，分别为：

序号	表名	原始数据源：MySQL	目标数据源：Hive 表名
1	订单表	orders	ods_order_order
2	商品表	products	ods_product_product
3	类目表	categories	ods_product_category
4	城市表	cities	ods_order_city

任务开发方案设计如下：



补充说明

全量同步与增量同步的区别：

名称	说明
全量同步	**定义：**全量同步是指在每次同步操作时，系统会传输两个数据库或数据仓库中的所有数据。
	适用场景：全量同步通常适用于数据量不大或者数据变化不频繁的情况，以及在初始同步或数据迁移时。
	优点： • 简单易实现：不需要跟踪数据变化，直接复制所有数据。 • 完整性：确保数据的完整性和一致性，因为所有数据都被重新同步。
	缺点： • 时间和资源消耗大：需要传输大量数据，耗时且占用带宽。 • 成本高：对于大数据量，可能需要更多的存储和计算资源。
增量同步	**定义：**增量同步只同步自上次同步以来发生变化的数据，而不是全部数据。
	适用场景：适用于数据量较大或数据频繁更新的环境。
	优点：

- 效率高：只同步变化的数据，节省时间和带宽。
- 成本低：减少了存储和计算资源的需求。
- 实时性：可以更快地反映数据的最新状态。

缺点：

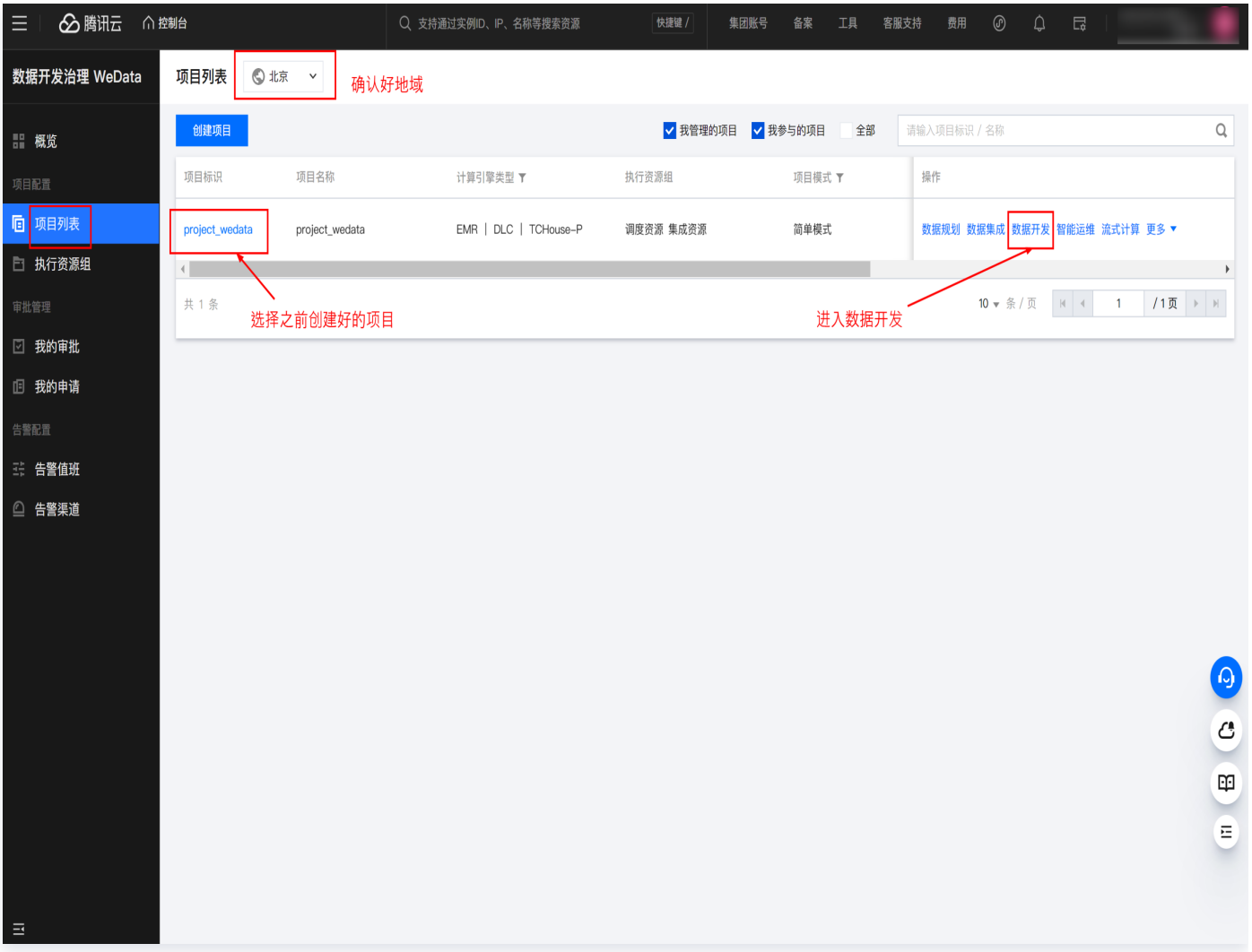
- 复杂性高：需要有机制来跟踪和记录数据的变化。
- 可能存在一致性问题：如果同步过程中出现问题，可能会导致数据不一致。

总结：
选择哪种同步方式取决于具体的应用场景和需求。如果数据量不大，变化不频繁，全量同步可能更简单高效。反之，如果数据量大且更新频繁，增量同步可以显著提高效率和降低成本。在实际应用中，有时也会结合使用这两种策略，例如，定期进行全量同步以确保数据的完整性，同时在日常操作中使用增量同步以提高效率。

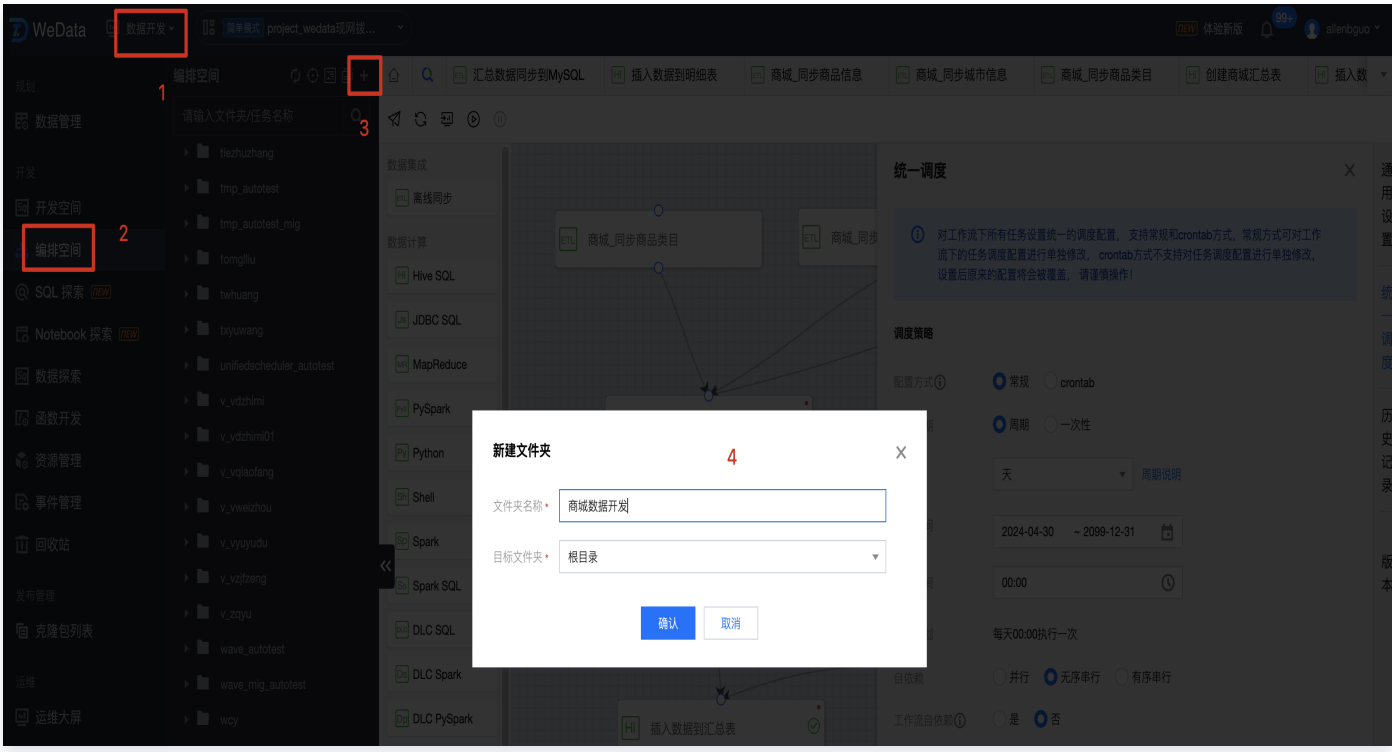
离线同步任务开发

创建工作流

1. 从 项目列表 进入离线开发页面。

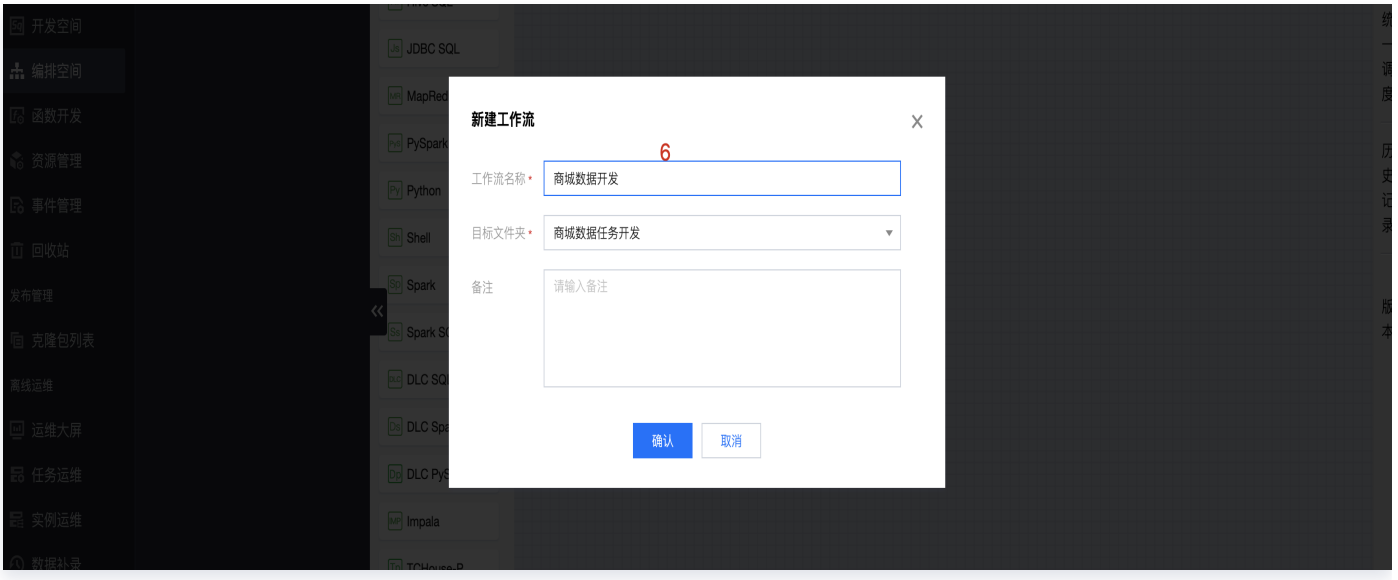


2. 选择离线的**数据开发** > **编排空间**，单击 + 号新建一个文件夹（命名为：商城数据任务开发），存放后续的开发任务。



3. 找到**新建的文件夹** > 鼠标右键**新建工作流**。

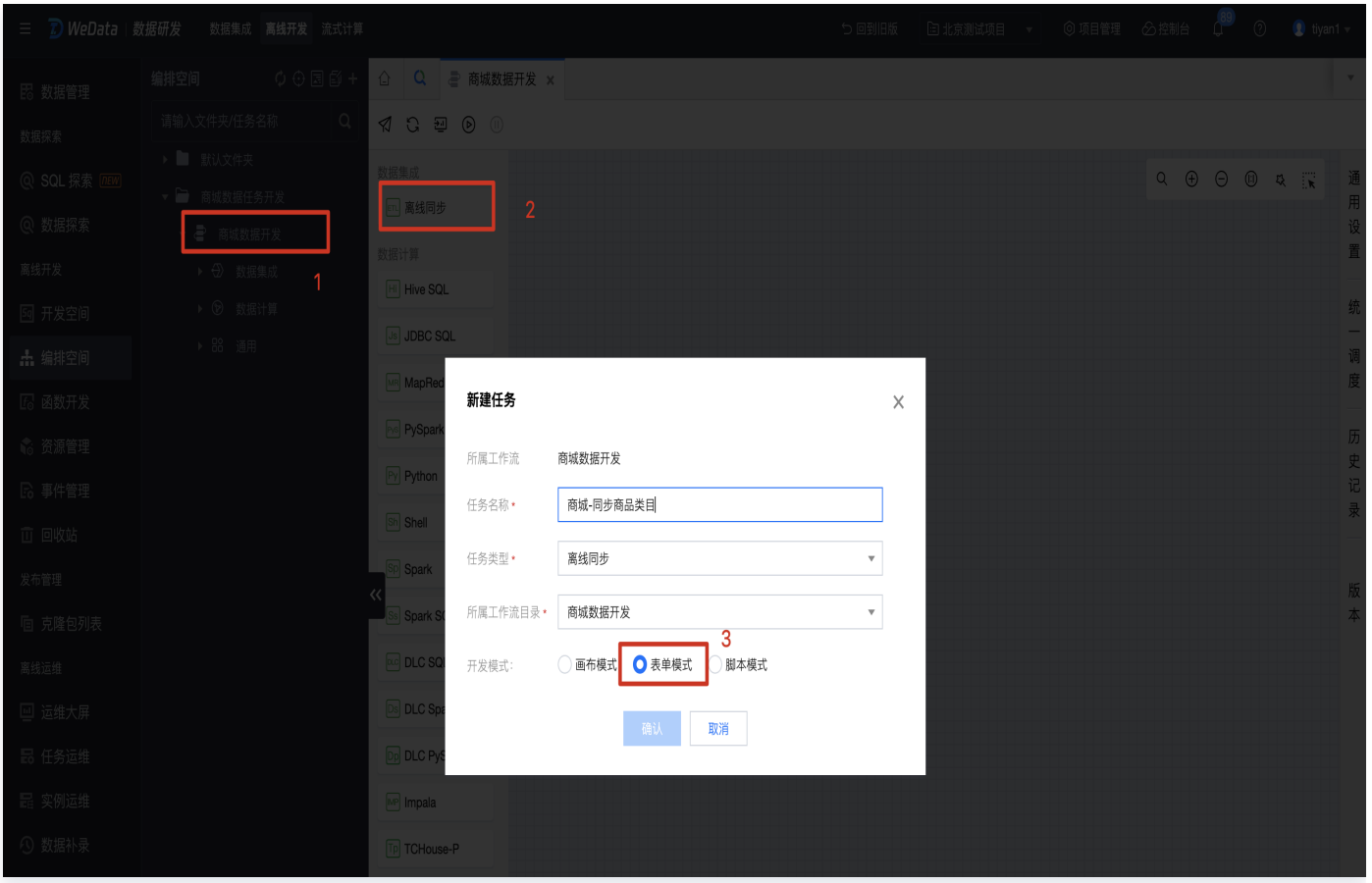
4. 新建工作流命名为：商城数据开发，选择对应的目标文件夹后单击**确认**。



同步类目表

下面我们先将类目表从 MySQL 同步到 Hive 表中。

1. 在编排空间找到刚创建的**商城数据任务开发** > **商城数据开发** > 单击**离线同步**。选择配置模式（任务名称：商城_同步商品类目，开发模式：此处选择**表单模式**），单击**确认**。



2. 配置原始数据源。

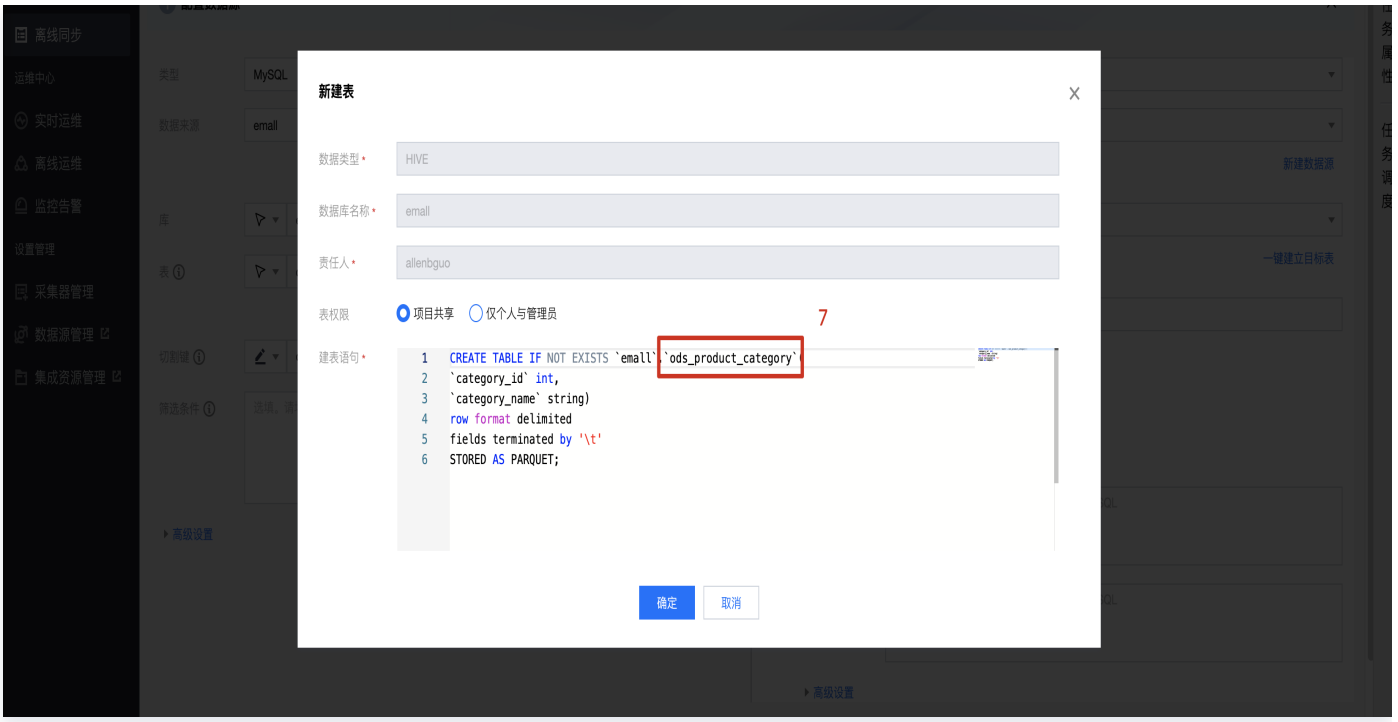
   2.1 此处选择原始数据表存放的库表，请选择上个步骤中添加的 MySQL 数据源。下文中，我们在介绍订单表同步时，将介绍通过设置筛选条件，实现增量数据同步。

   > **注意：**
   > 由于类目表为基础信息，且一般数据量不大，此处我们不需要设置筛选条件（即 Where 语句）。

   2.2 配置目标数据源，此处选择需要存放数据的 Hive 表，请选择上个步骤中添加的 Hive 数据源与数据库。

   - 类型：Hive
   - 数据去向：搜索 hive_emr
   - 库：emall（上个步骤中创建的库）

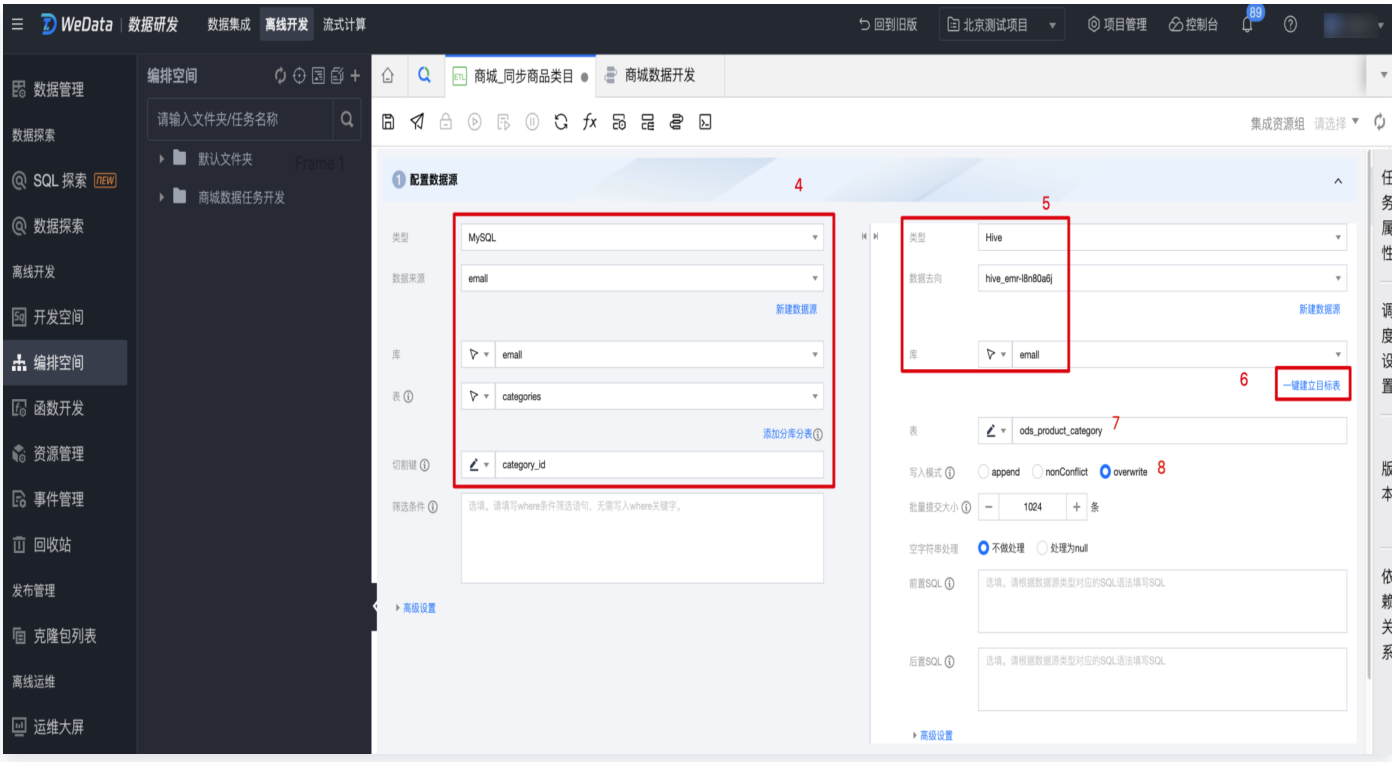
   2.3 建立目标表，此处使用**一键建立目标表**，复制 MySQL 表结构。



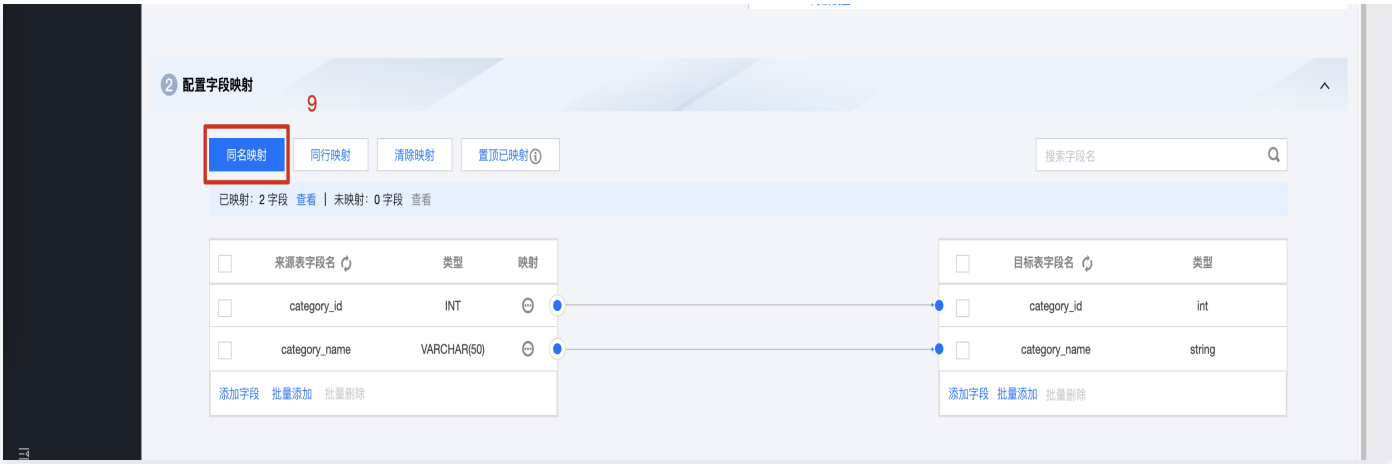
> **注意：**
> 请在弹框中修改建表语句，修改表名：ods_product_category。

2.4 选择表：ods_product_category

2.5 由于类目表为基础信息，此处我们选择 **overwrite**，即每次覆盖更新。

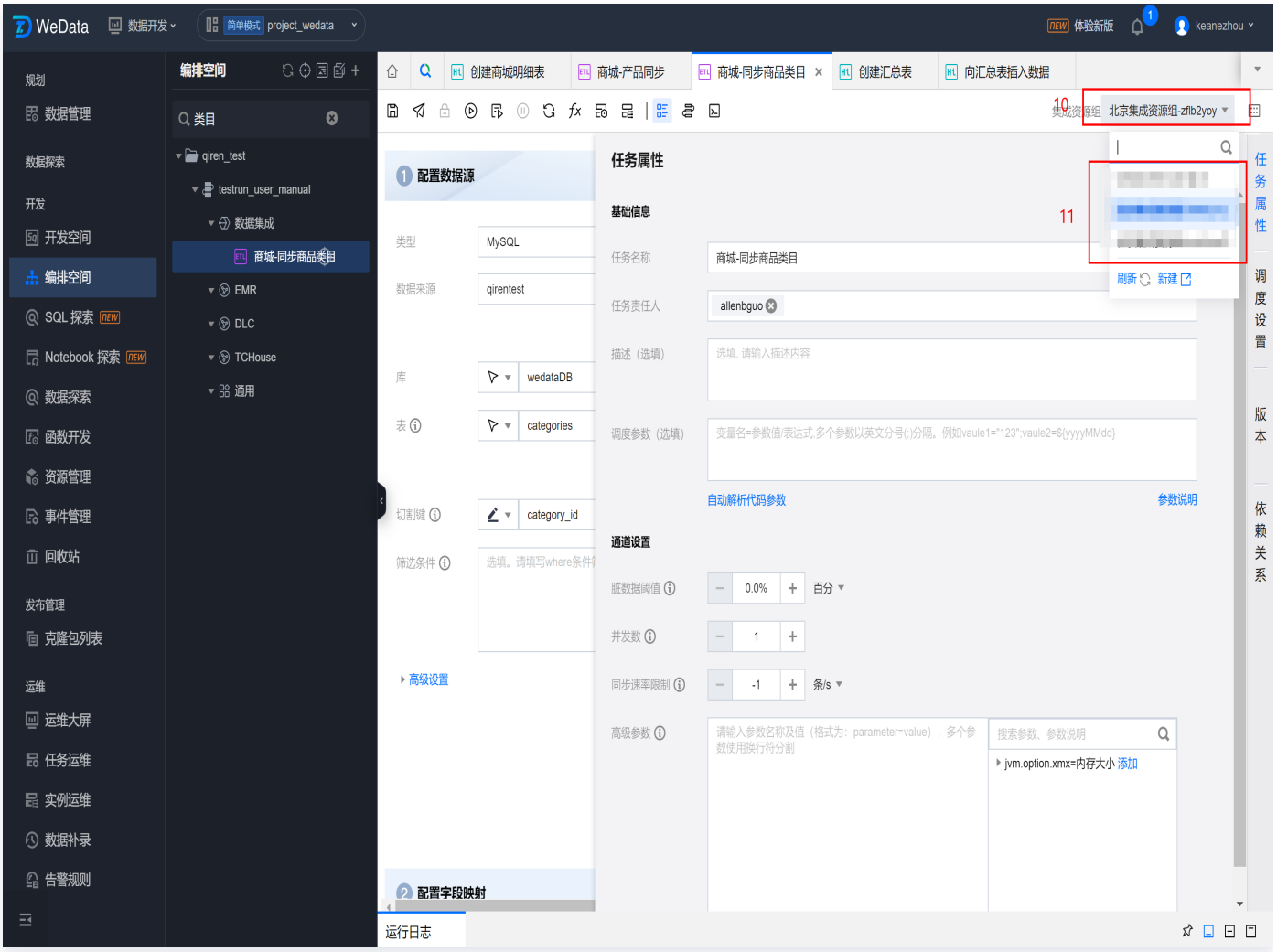


3. 配置字段映射，此处我们需要将源表与目标表的字段一一对应，由于我们的表结构是相同的，因此可以**同名映射**。



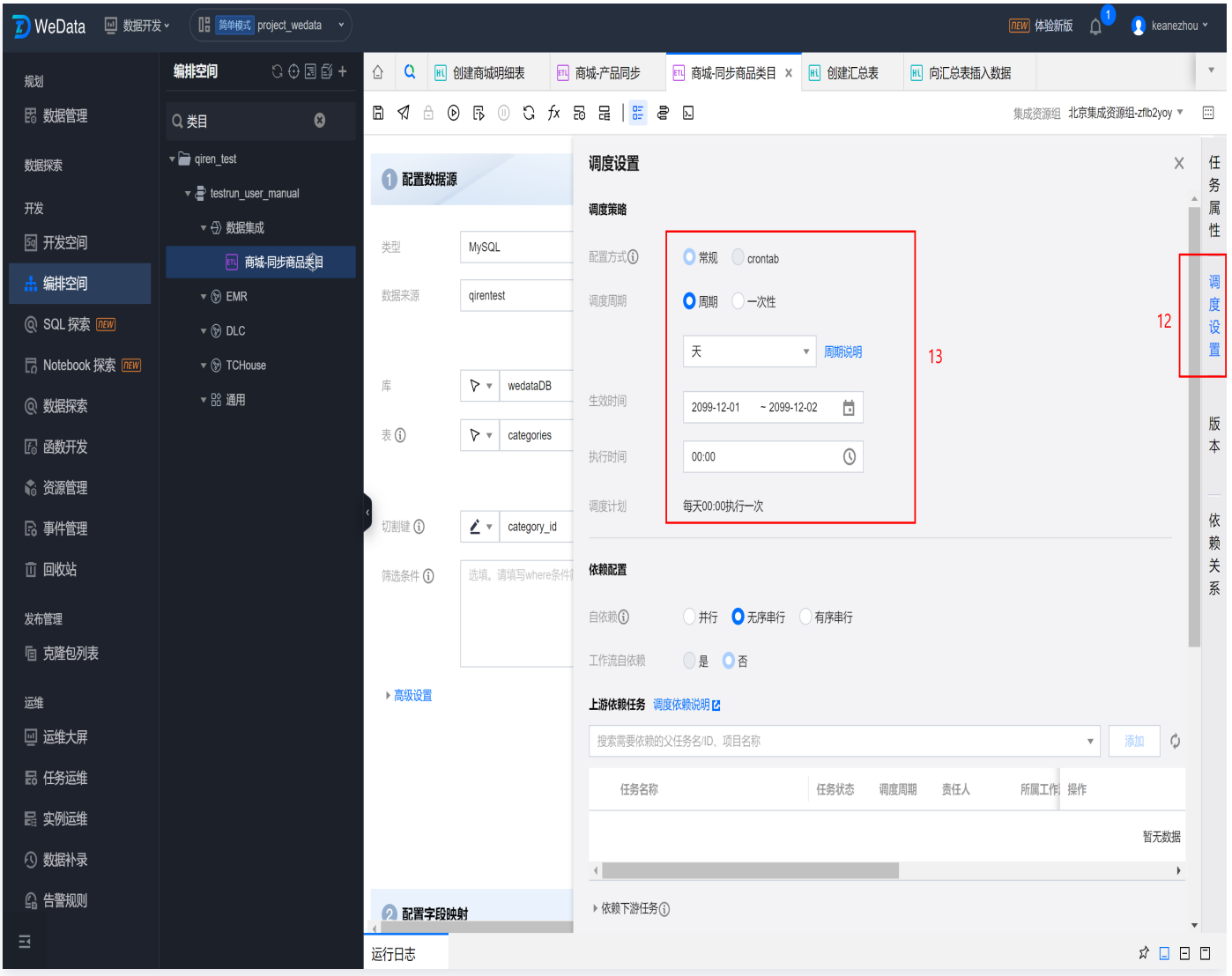
4. 设置任务属性。选择设置运行时需要使用的集成资源组。

> **注意：**
> 此处务必保证集成资源组、MySQL 实例、EMR 集群的网络联通性，必须购买同一地域的腾讯云资源。



5. 设置任务调度，此处我们需要设置任务的运行策略。由于类目表变化频率不大，我们设定每天凌晨同步一次。

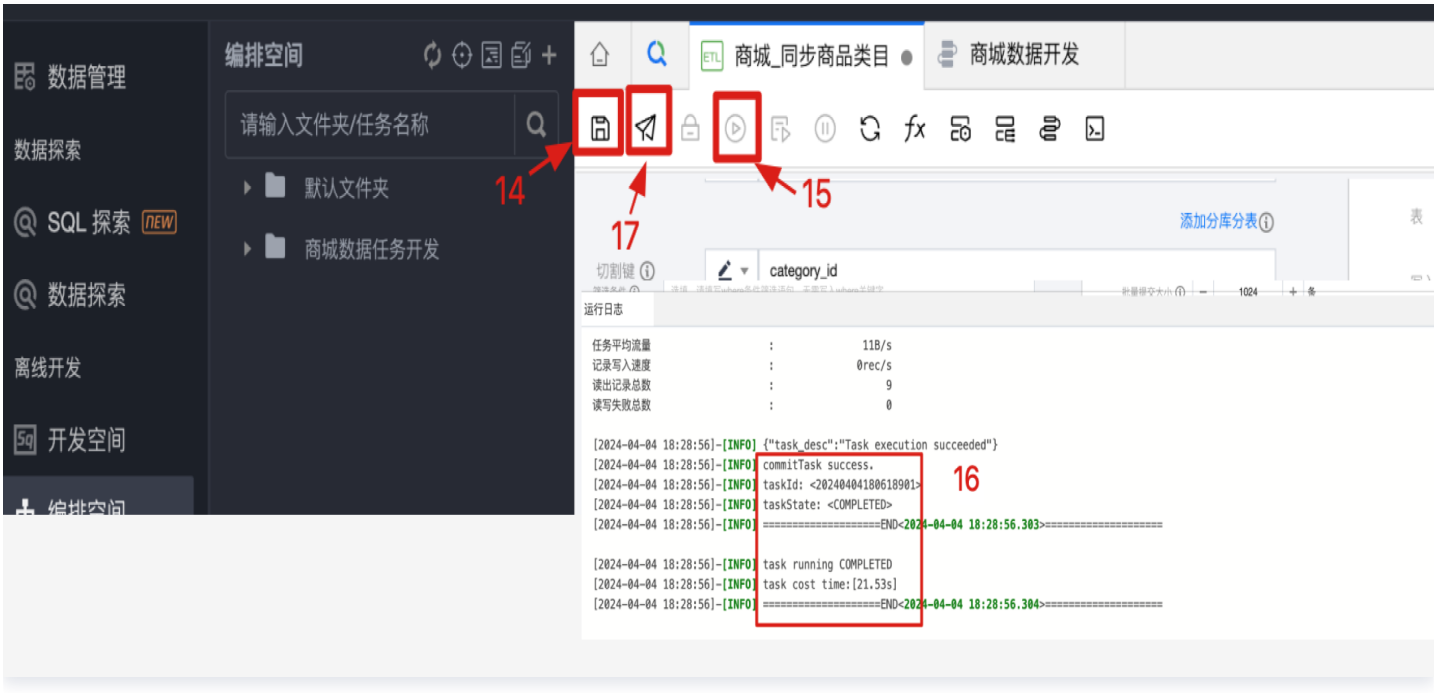
- 调度方式：周期调度
- 生效日期：默认
- 调度周期：天
- 执行时间：00:00



6. 设置任务属性：
7. 上述步骤完成后，请及时保存数据。
8. 在正式提交之前可以先模拟运行一次，此时也会先检测配置完整性和网络联通性，待检测通过，系统将立即开始运行。页面下方将出现运行日志。
9. 在页面下方可以看到试运行的日志和进度，当出现 Success、Completed 则表示试运行成功。
10. 指将任务提交至调度资源服务器上，待到达设置的运行时间，任务会自动开始运行。至此，数据集成任务配置就完成了。

> **注意：**
> 提交时，系统将自动检测配置完整性和网络联通性。

如果提示网络联通性问题，不要忽略，请立即检查：集成资源组、MySQL 实例、EMR 集群的网络是否互通。



通过完成1 – 17步骤，您已经完成了将类目表从 MySQL 数据库同步到 EMR 集群中，并且每日凌晨，WeData 将自动拉取全量数据覆盖更新。

同步城市表

下面我们创建第2个离线同步任务，将城市表从MySQL同步到Hive表中。
由于您已经完成了一张数据表的同步任务，应该对数据同步有了一定的认识，此处我们增加一个环节：制定同步策略。
同步策略一般包括：

- **离线同步还是实时同步?**

  当业务对实时性要求不大时，我们一般选择离线同步。

- **补充说明**

  实时同步与离线同步的区别:

方式	说明
实时同步	**定义：**实时同步指的是数据在源系统产生变化后，几乎立即被传输到目标系统。
	适用场景：需要数据高度实时性的应用，如金融交易、在线协作工具等。
	优点： • 实时性：数据变化可以立即反映到目标系统，减少数据不一致的时间窗口。

	• 数据一致性：由于同步速度快，可以减少因数据不一致导致的问题。
	缺点： • 系统资源消耗大：需要持续的网络连接和较高的系统资源以维持实时性。 • 成本高：实时同步可能需要更复杂的技术支持和更高的运维成本。 • 复杂性：实现实时同步需要更复杂的逻辑来处理数据冲突和同步状态。
离线同步	**定义：**离线同步指的是数据在源系统产生变化后，并不立即传输，而是在特定的时间点或条件下进行数据的批量传输。
	适用场景：数据实时性要求不高，或者网络条件不稳定的环境，如移动设备的数据备份、某些企业数据的定期同步等。
	优点： • 系统资源消耗低：可以根据网络和系统资源情况安排同步，减少对实时资源的需求。 • 成本低：相比实时同步，离线同步的运维成本和技术支持要求较低。 • 灵活性：可以根据实际需要安排同步的时间和频率。
	缺点： • 延迟：数据同步存在延迟，可能无法立即反映最新的数据变化。 • 数据一致性风险：如果同步间隔较长，可能会导致数据不一致的问题。
总结： 选择哪种同步方式取决于业务需求、数据的重要性、网络条件以及成本预算。实时同步适用于对数据实时性要求极高的场景，而离线同步则适用于可以容忍一定数据延迟的环境。在某些情况下，也可以结合使用这两种策略，例如，在网络条件不佳时使用离线同步，而在网络条件良好时使用实时同步，以此来平衡实时性和成本。	

- **同步策略选择增量还是全量?**

  在真实的业务场景中，一般都会选择增量同步。只有在数据表初始化时会选择全量同步。
  本次教程中，城市、类目、商品表中未设置日期等切片字段，我们依赖选择全量同步。
  下面步骤中，在同步订单表时，我们会介绍如何设置增量同步。

- **如果是离线同步，那么同步频次选择每天还是每小时?**

  同步频次需要根据业务需要进行确定，频次越小对资源的消耗越大。
  本次教程中，我们均选择每天凌晨同步一次。
  通过上面的思考，城市表与类目表的同步策略完全相同，请参照类目表中的步骤1-17，重复操作一遍，

> **注意：**
> - **以下步骤均为配图里标明的步骤序号**
> - 步骤3：任务名称：商城_同步城市信息
> - 步骤4：选择表：cities；
> - 步骤6、7：表名需要修改为：ods_order_city；
> - 步骤10 – 13：这四个步骤比较容易忽略；
> - 步骤15：无论有多熟悉操作，请记得在提交前试运行一次，保证任务运行准确。

同步商品表

下面我们创建第3个离线同步任务，将商品表从 MySQL 同步到 Hive 表中。
依然先思考再操作：

序号	问题	结论
1	离线同步还是实时同步?	离线
2	同步策略选择增量还是全量?	全量 **说明：** 其实应该增量，此处仅为了教学我们选择全量。
3	如果是离线同步，那么同步频次选择每天还是每小时?	每天凌晨同步

通过上面的思考，商品表与类目表、城市表的同步策略也完全相同，请继续参照类目表中的步骤1–17，重复再操作一遍。

> **注意：**
> - **以下步骤均为配图里标明的步骤序号**
> - 步骤3：任务名称：商城_同步商品信息。
> - 步骤4：选择表：products。
> - 步骤6、7：表名需要修改为：ods_product_product。
> - 步骤10 – 13：这四个步骤比较容易忽略。
> - 步骤15：无论有多熟悉操作，请记得在提交前试运行一次，保证任务运行准确。

同步订单表

下面我们创建第4个离线同步任务，将订单表从 MySQL 同步到 Hive 表中。
依然先思考再操作：

序号	问题	结论
1	离线同步还是实时同步?	离线
2	同步策略选择增量还是全量?	增量 **说明：** 由于订单表往往数据量较大，更适合增量同步，即每天同步前一天的订单数据。
3	如果是离线同步，那么同步频次选择每天还是每小时?	每天凌晨同步

通过上面的思考，订单表与上面三张表的同步策略是有差异的，此处我们选择了增量同步，在此步骤中我们将重点介绍。
增量同步逻辑：
在原始数据订单表中，有个特殊字段：order_time，它是会随着时间推移。
因此我们可以以订单创建时间为分区，根据 order_time 保证每日拉取增量数据。
为了能够在调度中动态比较**任务运行时间**（使用 ${yyyy-MM-dd} 表示）与 **order_time** 的大小关系
例如：当运行时间为2024 - 04 - 01 00:10，则 ${yyyy-MM-dd} = 2024-04-01，同时 ${yyyy-MM-dd-1d} = 2024-03-31
因此我们可以用 date(order_time) = '${yyyy-MM-dd-1d} ' 表示昨天的数据。
创建离线同步任务
首先请继续参照类目表中的步骤1-13，重复再操作一遍。

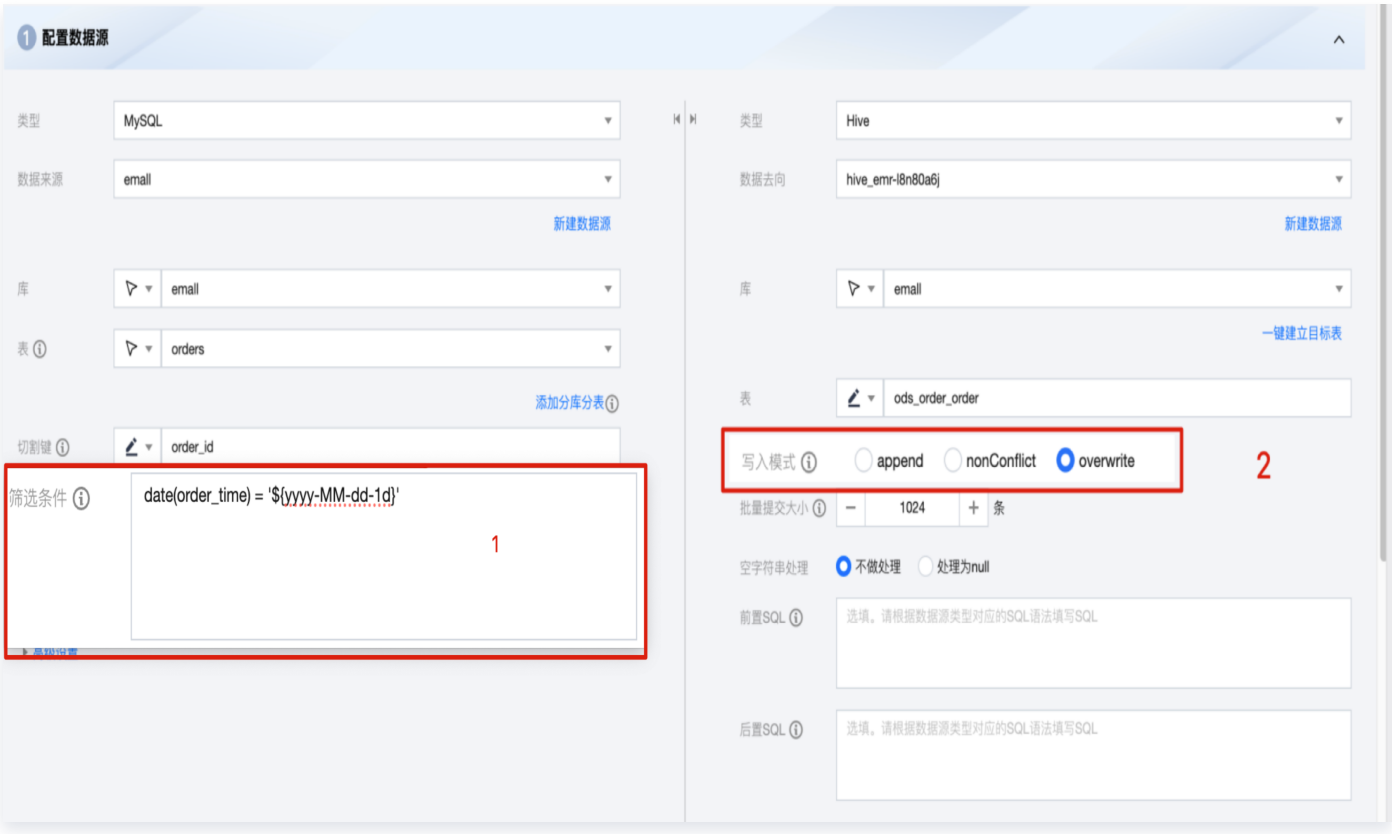
注意：
- **以下序号均为配图里标明的步骤序号**
- 步骤4：选择表：orders；
- 步骤6、7：表名需要修改为：ods_order_order；
- 先不要操作步骤14 - 17（不要提交），我们需要修改下面配置。

操作演示截图：
1. 打开配置数据源页面，左侧填写**筛选条件**，右侧选择**写入模式**为 **overwrite**。
- 筛选条件：根据数据类型填写对应筛选语句，该语句会作为将要同步数据的筛选条件。
- 写入模式：

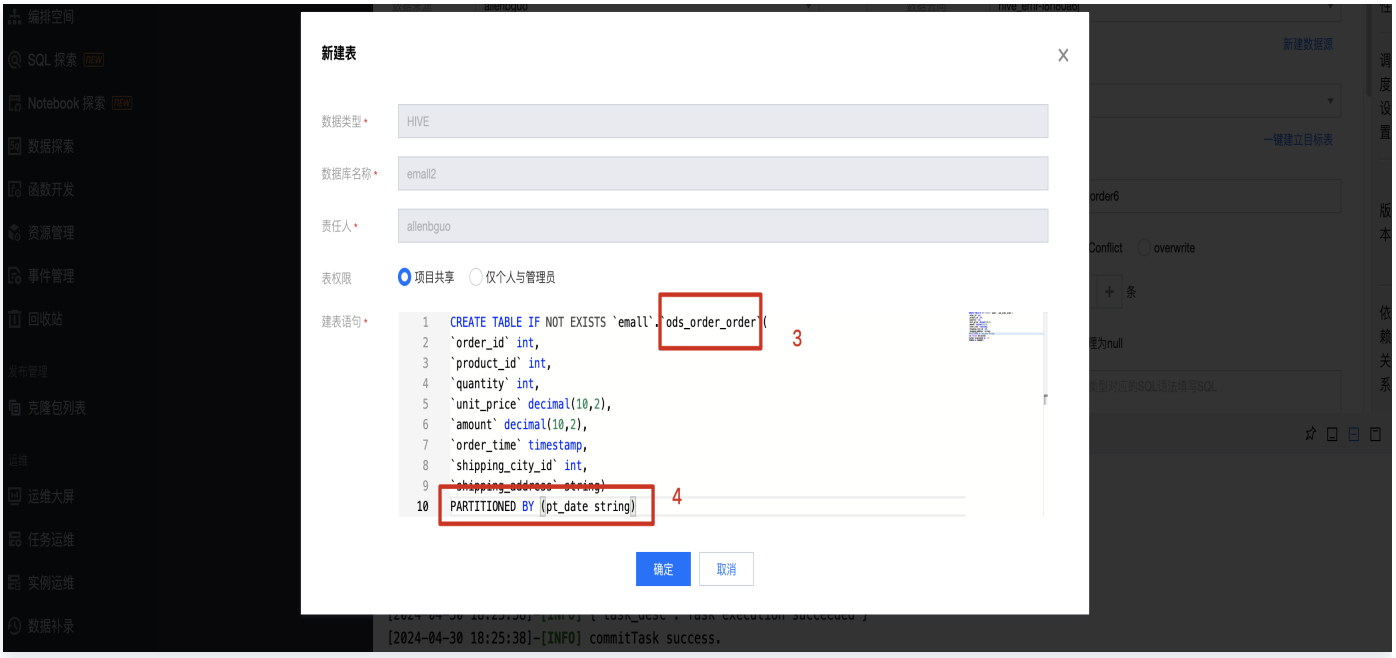
1.1 Append：保留原始数据, 新行追加写入。

1.2 nonConflict：数据冲突时报错。

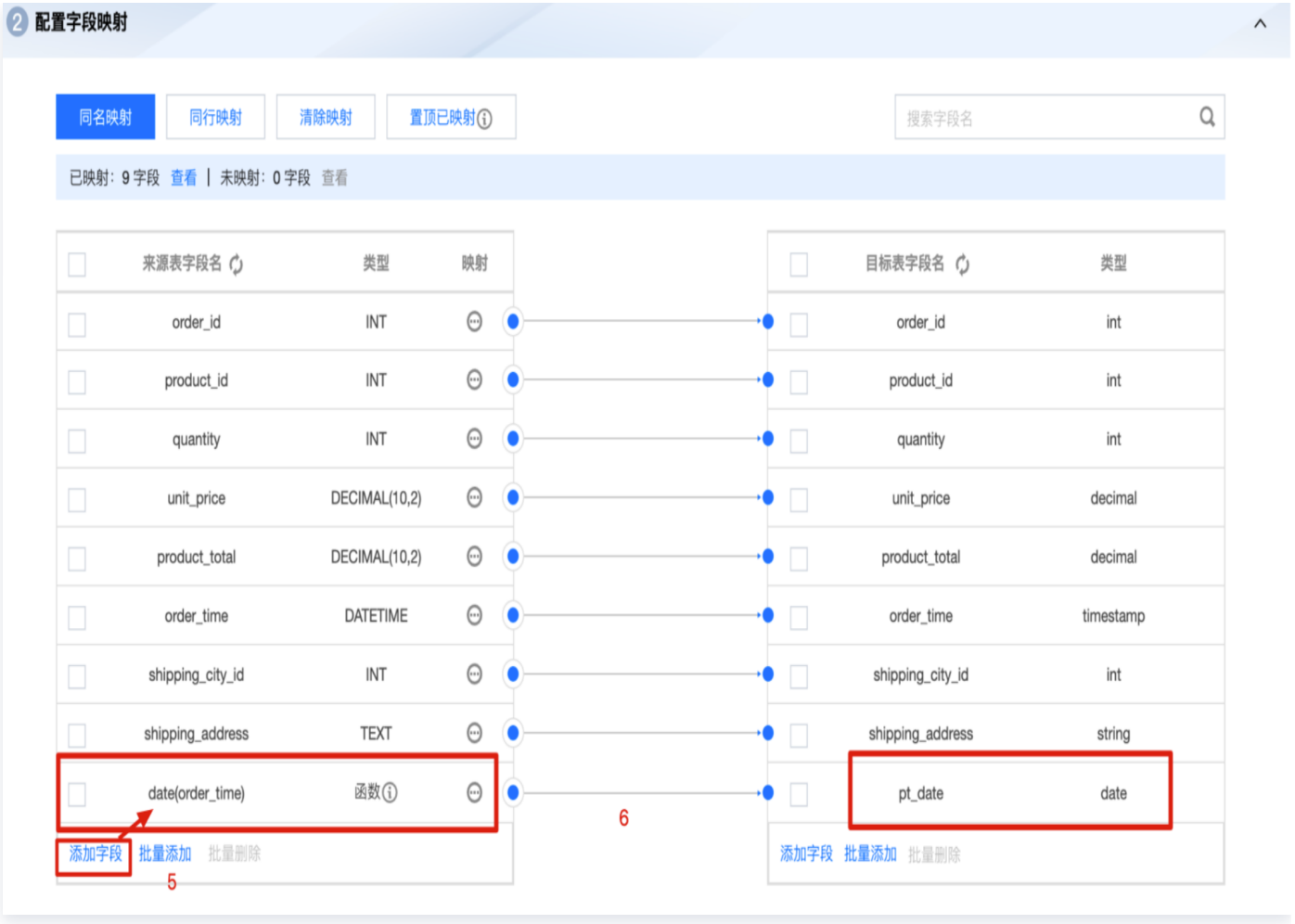
1.3 Overwrite：删除原有数据重新写入。



2. 打开**新建表**页面，修改表名为 ods_order_order，同时增加分区字段 PARTITIONED BY (pt_date date)。



3. 在**配置字段映射**页面中，点击**添加字段**，字段名为 date(order_time)，类型为函数，单击右侧小圆圈，以鼠标拖拽方式向右侧目标字段 pt_date **建立映射关系**。



4. 单击右侧任务栏中**任务调度**按钮，在任务调度页面中找到**执行时间**，修改执行时间为：00:10。

建立订单表 SQL：

```
--一键创建订单表
CREATE TABLE IF NOT EXISTS `emall`.`ods_order_order`(
`order_id` int,
`product_id` int,
`quantity` int,
```

```
`unit_price` decimal(10,2),
`amount` decimal(10,2),
`order_time` timestamp,
`shipping_city_id` int,
`shipping_address` string)
PARTITIONED BY (pt_date date)
row format delimited
fields terminated by '\t'
STORED AS PARQUET;
```

至此，我们已经完成了所有原始数据表到 Hive 表的离线同步任务。并且每日凌晨，WeData 将自动进行全量/增量数据同步。

总结

现在您已经完成了数据集成部分的学习，现在进行总结：

序号	步骤名称
1	**确定数据原始表与数据目标表** 原始表：读：数据来源 目标表：写：数据目的地
2	**确定离线同步还是实时同步** 根据业务需要，如果没有必要，可选择离线同步，减少资源消耗
3	**确定增量同步还是全量同步** 一般数据初始化时需要全量同步，周期同步时均为增量同步； 增量同步时，需要设置筛选条件，确保拉取数据不重叠
4	**确定网络环境互通性** 同步过程中涉及到三个环境： 1. 原始数据库实例 2. 集成资源组 3. EMR 集群 ⚠ **注意：** 必须保证集成资源组可以访问原始数据库实例和 EMR 集群。

下面我们将进行离线开发部分的学习，即在 EMR 集群的 Hive 表中进行数据加工。

离线开发

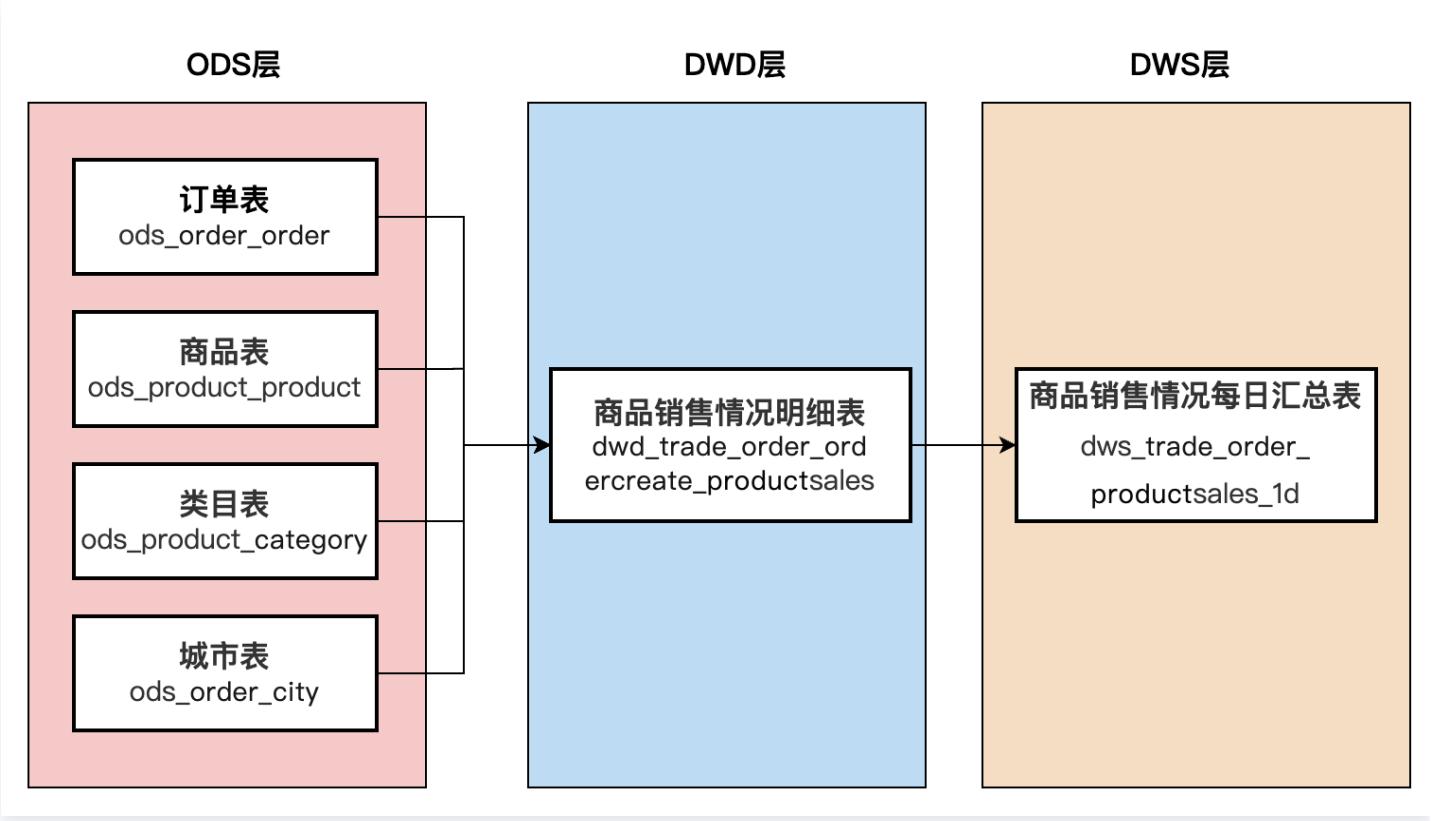
最近更新时间：2024-07-19 11:17:21

通过前面步骤的学习，我们已经将所有的原始数据同步到了 EMR 集群的 Hive 表中。

- 但是，这些数据均为原始数据结构，无法直接提供给业务使用。
- 结合**数据表结构设计**步骤中的内容，我们已经对业务需求进行了分析，并且对数仓层级进行了划分。

下面我们将通过数据开发，完成明细表与汇总表的生成与数据处理。

离线开发任务设计



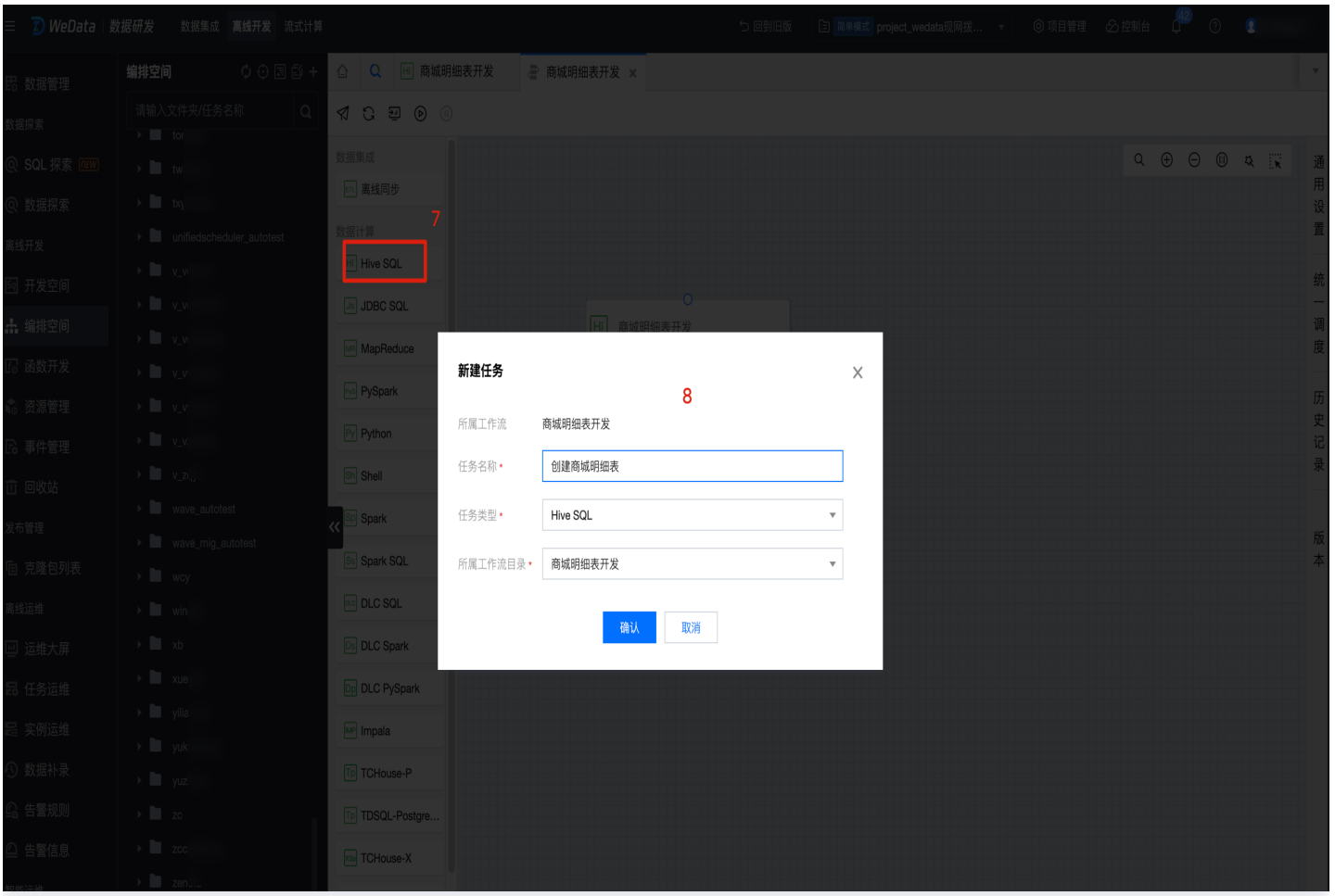
离线开发任务开发

明细表开发

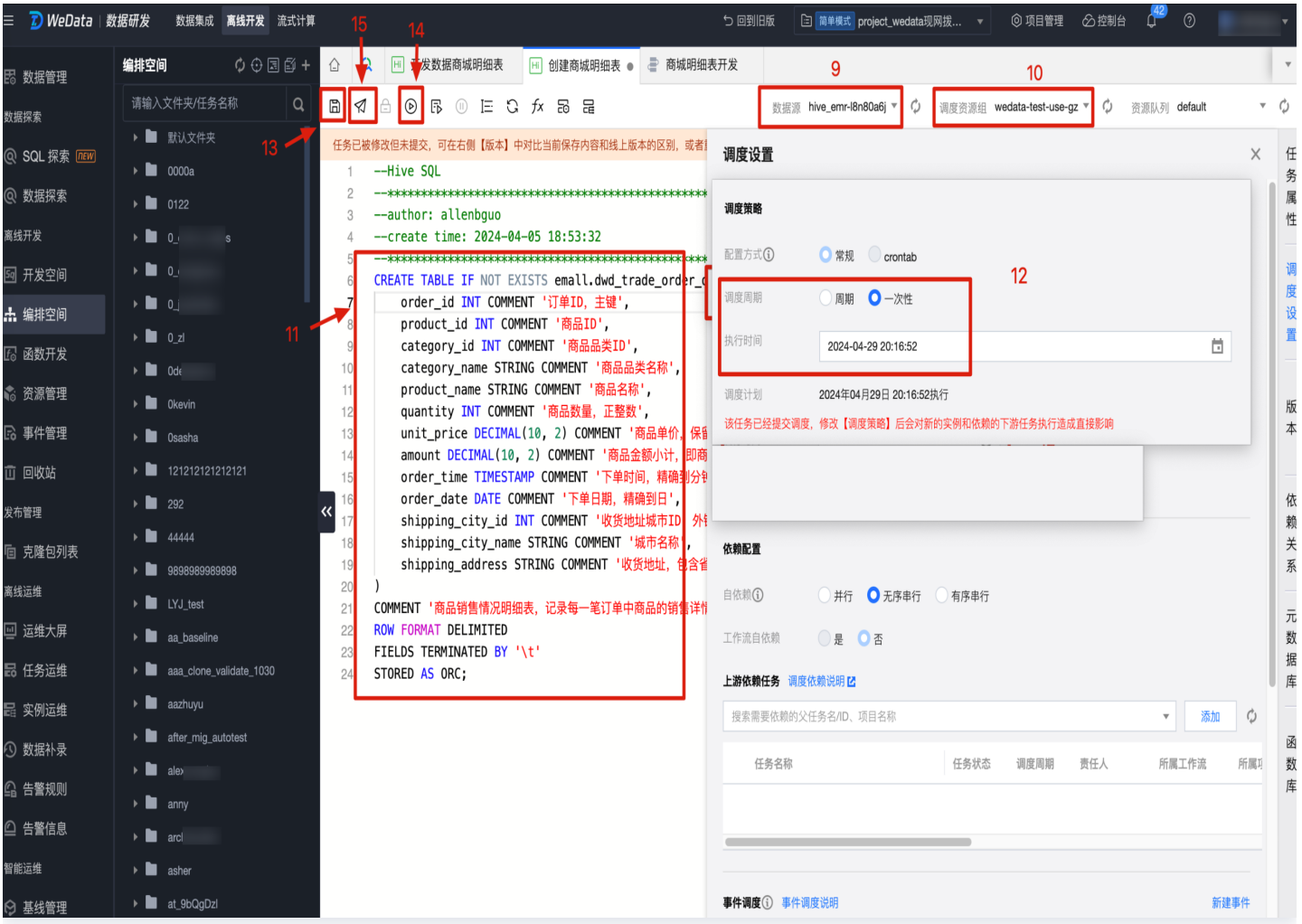
完成明细表开发主要包括以下4步：

创建商城明细表

1. 在**编排空间**模块，任意文件夹下的商城明细表开发工作流目录下，单击数据计算下的 **Hive SQL 图标**，新建一个 Hive SQL 任务，任务名称为创建商城明细表，任务类型是 Hive SQL，单击**确认**。



2. 确认完成后，会弹出 Hive SQL 开发脚本页面，按以下步骤顺序完成 Hive SQL 开发任务。
   - 数据源：选择 hive_emr-XXX。
   - 调度资源组：选择我们购买的资源组，建议同网域。
   - 开发脚本：在脚本页面中编写好商城明细表的建表 SQL 语句。
   - 调度设置：在右侧任务栏里，单机**调度设置**，选择调度周期为**一次性**，执行事件默认即可。
     - 一次性：该 Hive SQL 根据执行时间只执行一次。
     - 周期：该 Hive SQL 根据执行时间定时执行。
   - **单击保存**按钮：保存该 Hive SQL 任务。
   - **单击运行**按钮：执行一次该任务，可校验脚本任务正确性。
   - **单击提交**按钮：正式提交该任务到调度资源服务器，到达指定时间后即可根据调度周期执行任务。



3. 创建商城明细表 HiveQL 语句：

```
--创建明细表HiveQL语句
CREATE TABLE
  IF NOT EXISTS emall.dwd_trade_order_ordercreate_productsales (
    order_id INT COMMENT '订单ID，主键',
    product_id INT COMMENT '商品ID',
    product_name STRING COMMENT '商品名称',
    category_id INT COMMENT '商品品类ID',
    category_name STRING COMMENT '商品品类名称',
    quantity INT COMMENT '商品数量，正整数',
    unit_price DECIMAL(10, 2) COMMENT '商品单价，保留两位小数',
    amount DECIMAL(10, 2) COMMENT '商品金额小计，即商品数量乘以单价',
    order_time TIMESTAMP COMMENT '下单时间，精确到分钟',
    shipping_city_id INT COMMENT '收货地址城市ID，外键',
    shipping_city_name STRING COMMENT '城市名称',
    shipping_address STRING COMMENT '收货地址，包含省、市、区、详细地址'
  )
  COMMENT '商品销售情况明细表，记录每一笔订单中商品的销售详情'
```

```
PARTITIONED BY (pt_date STRING)
row format delimited fields terminated by '\t'
STORED AS PARQUET;
```

通过完成7 – 15步骤，您已经完成了在 EMR 集群中的 Hive 数据源中创建了一个 Hive 表。

向明细表中写入数据

下面我们将开始向明细表中写入数据：
请重复操作步骤7 – 15，其中注意点：

> ⚠ **注意：**
>
> - **以下步骤均为配图里标明的步骤序号**
> - 在同一个工作流中，新建 HiveQL 节点；
> - 步骤8：命名为：插入数据到明细表；
> - 步骤9、10：这两个步骤比较容易忽略；
> - 步骤11：HiveQL语句如下；
> - 步骤12：
>
> 调度周期改为：**周期**
> 执行时间设置为：01:00
> 说明：此任务需要每天运行1次
> - 步骤14：无论有多熟悉操作，请记得在提交前后试运行一次，保证任务运行准确。

插入数据到明细表 HiveQL 语句：

```
--插入数据到明细表HiveQL语句
SET hive.exec.dynamic.partition.mode=nonstrict;
INSERT INTO TABLE emall.dwd_trade_order_ordercreate_productsales
PARTITION (pt_date)
SELECT
    o.order_id,
    o.product_id,
    p.product_name,
    p.category_id,
    ca.category_name,
    o.quantity,
    o.unit_price,
    o.amount,
    o.order_time,
    o.shipping_city_id,
    ci.city_name,
```

```
    o.shipping_address,
    o.pt_date
FROM
    emall.ods_order_order o
JOIN
    emall.ods_product_product p ON o.product_id = p.product_id
JOIN
    emall.ods_product_category ca ON p.category_id = ca.category_id
JOIN
    emall.ods_order_city ci ON o.shipping_city_id = ci.city_id
WHERE o.pt_date = '${yyyy-MM-dd-1d}';
```

以上我们完成了明细表的开发任务，每日凌晨待数据从原始表同步到 Hive 集群中后，系统将自动关联四张表，将数据统总到明细表中。

> **说明：**
> 此时明细表中冗余了一些字段，仅为了加工汇总表时少一些关联，提高计算效率。

汇总表开发

创建商城汇总表

接下面我们将开始开发汇总表
首先请重复步骤5 – 6，新建一条工作流：商城汇总表开发。
接下来重复步骤7 – 15，创建汇总表，其中注意点：

> **注意：**
> - **以下步骤均为配图里标明的步骤序号**
> - 在同一个工作流中，新建HiveQL节点；
> - 步骤8：命名为：创建商城汇总表；
> - 步骤9、10：这两个步骤比较容易忽略；
> - 步骤11：建表SQL语句如下；
> - 步骤12：
>
> 调度周期改为：**一次性**
> 执行时间设置为：默认即可
> 说明：此任务只需要运行1次
> - 步骤14：无论有多熟悉操作，请记得在提交前后试运行一次，保证任务运行准确。

创建商城汇总表 HiveQL 语句：

```
--创建汇总表HiveQL语句
CREATE TABLE IF NOT EXISTS emall.dws_trade_order_productsales_1d (
    order_date DATE COMMENT '统计日期，主键',
    city_id INT COMMENT '城市ID',
    city_name STRING COMMENT '城市名称',
    category_id INT COMMENT '商品品类ID',
    category_name STRING COMMENT '商品品类名称',
    quantity INT COMMENT '商品总销量，正整数',
    amount DECIMAL(10, 2) COMMENT '商品总销售额，保留两位小数'
)
COMMENT '商品销售情况每日汇总表'
PARTITIONED BY (pt_date STRING)
row format delimited fields terminated by '\t'
STORED AS PARQUET;
```

向汇总表写入数据

接下来重复步骤7 – 15，向汇总表插入数据，其中注意点：

> ⚠ **注意：**
>
> - **以下步骤均为配图里标明的步骤序号**
> - 在同一个工作流中，新建HiveQL节点；
> - 步骤8：命名为：插入数据到汇总表；
> - 步骤9、10：这两个步骤比较容易忽略；
> - 步骤11：HiveQL语句如下；
> - 步骤12：
>
>   调度周期改为：**周期**
>
>   执行时间设置为：01:00
>
>   说明：此任务需要每天运行1次
> - 步骤14：无论有多熟悉操作，请记得在提交前后试运行一次，保证任务运行准确。

插入数据到汇总表表 HiveQL 语句：

```
--插入数据到汇总表HiveQL语句
SET hive.exec.dynamic.partition.mode=nonstrict;
INSERT INTO TABLE emall.dws_trade_order_productsales_1d PARTITION
(pt_date)
SELECT
    p.pt_date AS order_date,
    p.shipping_city_id,
```

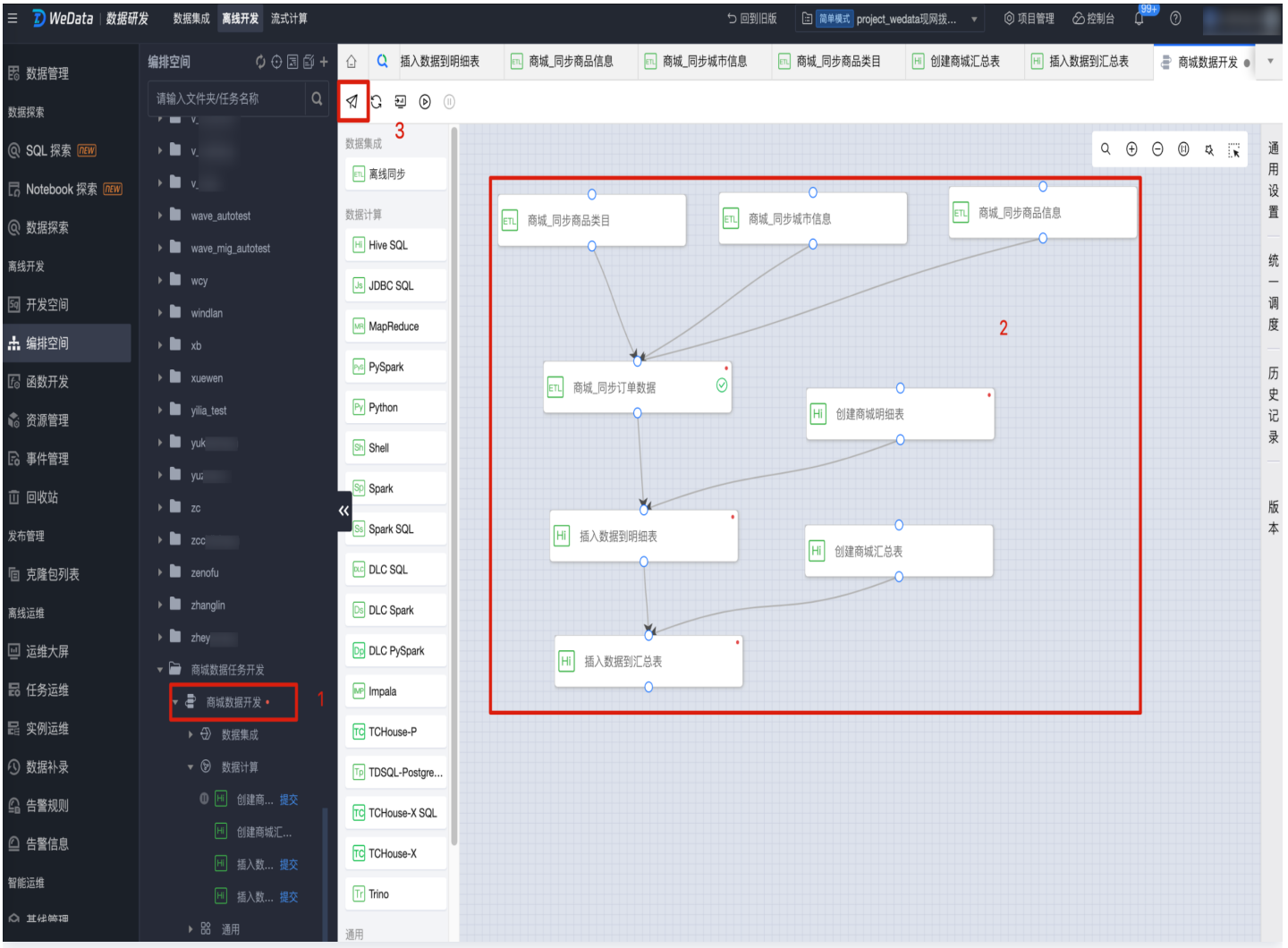
```
    p.shipping_city_name,
    p.category_id,
    p.category_name,
    SUM(p.quantity) AS quantity,
    SUM(p.amount) AS amount,
    p.pt_date
FROM
    emall.dwd_trade_order_ordercreate_productsales p
WHERE p.pt_date = '${yyyy-MM-dd-1d}'
GROUP BY
    p.pt_date,
    p.shipping_city_id,
    p.shipping_city_name,
    p.category_id,
    p.category_name;
```

以上我们已经完成了明细表和汇总表的离线开发任务，每日凌晨，WeData将自动进行明细表与汇总表的计算任务。

建立依赖并提交

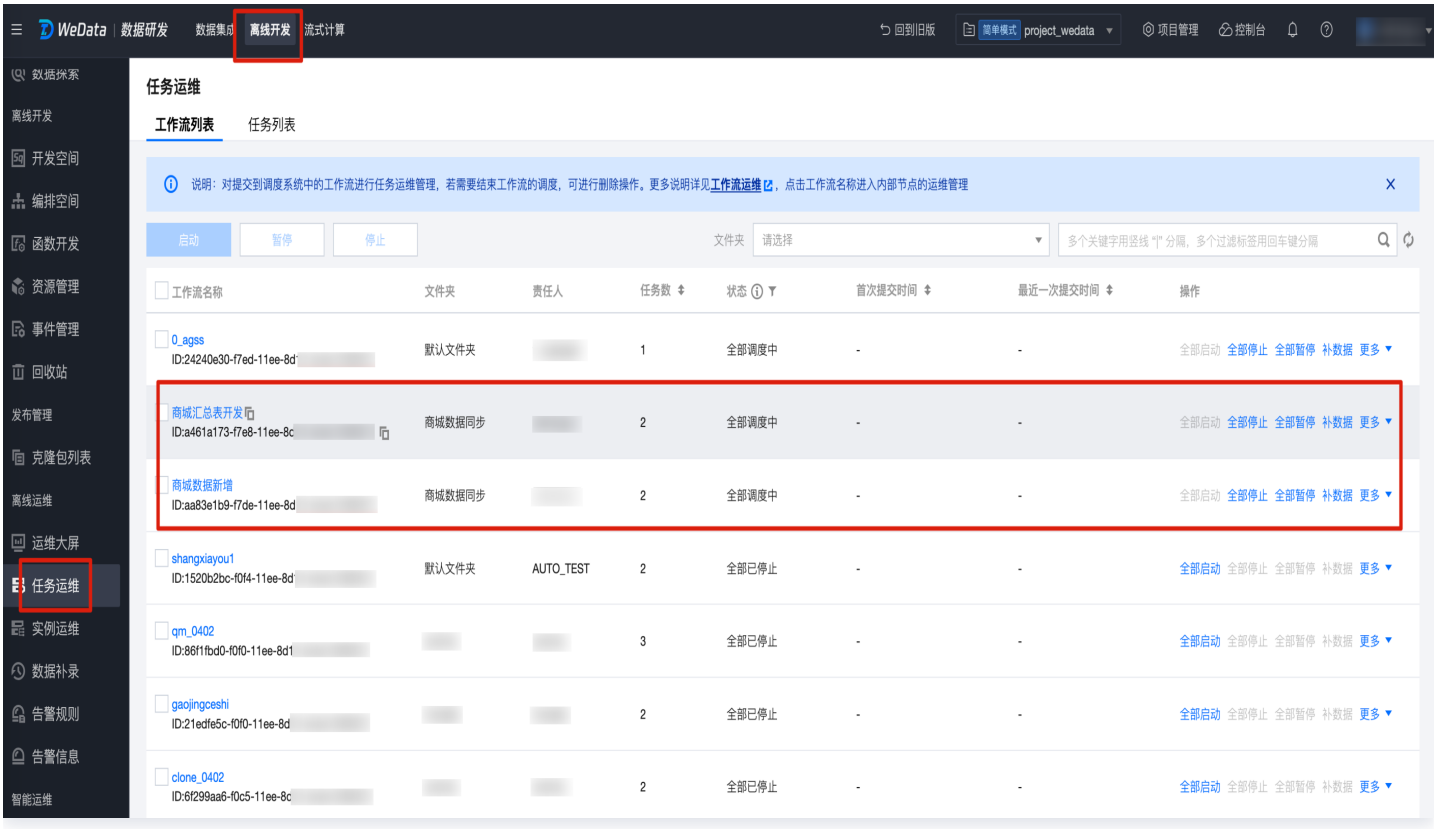
1. 双击商城数据开发工作流，会弹出工作流画布，显示该工作流下所有任务，依次建立工作流之间的依赖关系，完成后单击提交按钮。

- 依次建立依赖关系：
  - 商城_同步商品类目 → 商城_同步订单数据。
  - 商城_同步城市信息 → 商城_同步订单数据。
  - 商城_同步商品信息 → 商城_同步订单数据。
  - 商城_同步订单数据 → 插入数据到明细表。
  - 创建商城明细表 → 插入数据到明细表。
  - 插入数据到明细表 → 插入数据到汇总表。
  - 创建商城汇总表 → 插入数据到汇总表。
- 最后提交工作流，任务将按照设置的依赖关系有序运转。



离线开发任务运维

您可在任务运维中，查看工作流或离线任务的运行状态。


WeData | 数据研发
数据集成
离线开发
流式计算
回到旧版
简单模式
project_wedata
项目管理
控制台
数据探索
离线开发
开发空间
编排空间
函数开发
资源管理
事件管理
回收站
发布管理
克隆包列表
离线运维
运维大屏
任务运维
实例运维
数据补录
告警规则
告警信息
智能运维
任务运维
工作流列表
任务列表
说明：对提交到调度系统中的工作流进行任务运维管理，若需要结束工作流的调度，可进行删除操作。更多说明详见工作流运维，点击工作流名称进入内部节点的运维管理
启动
暂停
停止
文件夹
请选择
多个关键字用竖线 "|" 分隔，多个过滤标签用回车键分隔
工作流名称
文件夹
责任人
任务数
状态
首次提交时间
最近一次提交时间
操作
0_agss
ID:24240e30-f7ed-11ee-8d
默认文件夹
1
全部调度中
全部启动 全部停止 全部暂停 补数据 更多
商城汇总表开发
ID:a461a173-f7e8-11ee-8c
商城数据同步
2
全部调度中
全部启动 全部停止 全部暂停 补数据 更多
商城数据新增
ID:aa83e1b9-f7de-11ee-8d
商城数据同步
2
全部调度中
全部启动 全部停止 全部暂停 补数据 更多
shangxiayou1
ID:1520b2bc-f0f4-11ee-8d
默认文件夹
AUTO_TEST
2
全部已停止
全部启动 全部停止 全部暂停 补数据 更多
qm_0402
ID:86f1fbd0-f0f0-11ee-8d1
3
全部已停止
全部启动 全部停止 全部暂停 补数据 更多
gaojingceshi
ID:21edfe5c-f0f0-11ee-8d
2
全部已停止
全部启动 全部停止 全部暂停 补数据 更多
clone_0402
ID:6f299aa6-f0c5-11ee-8c
2
全部已停止
全部启动 全部停止 全部暂停 补数据 更多

数据质量

最近更新时间：2024-12-31 17:16:52

在此步骤中，我们将完成对数仓中的数据表进行质量监控，防止脏数据向下游传递。

质量监控任务设计

当明细表中，以下字段为空时，对汇总表将造成严重的影响：

- 监控表：dwd_trade_order_ordercreate_productsales。
- 监控字段：amount、order_date。
- 监控逻辑：依赖明细任务完成后，自动检测是否存在空值。

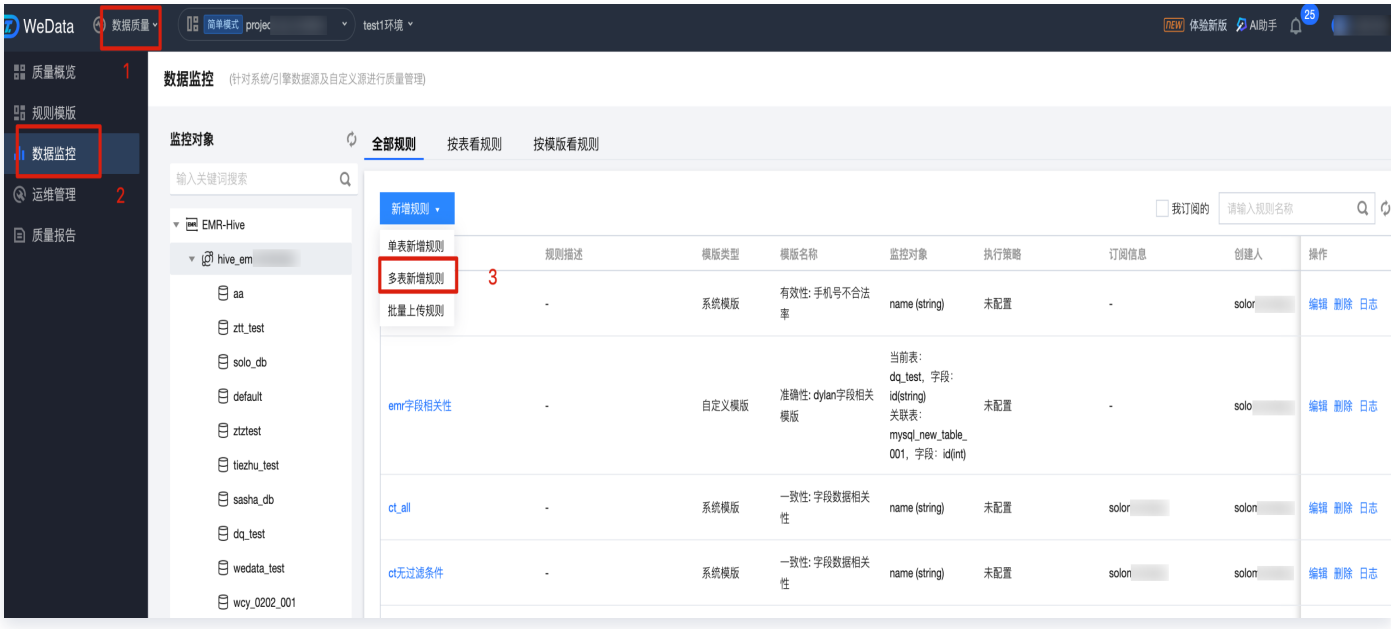
质量监控任务开发

质量任务开发主要包含以下7步：

空值检测任务

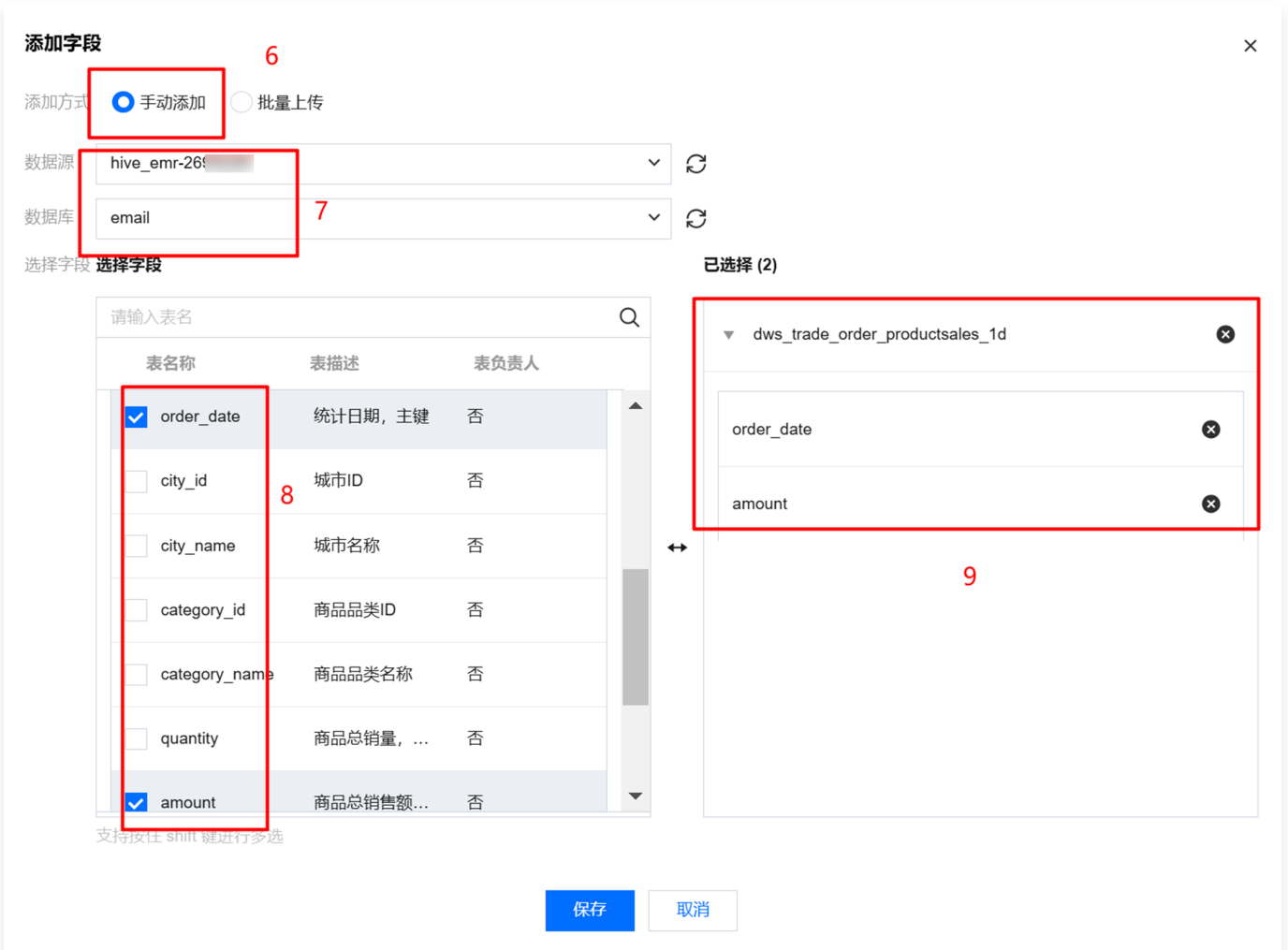
步骤1：选择监控字段

1. 单击**数据质量**模块，进入 数据监控 页面，再单击**多表新增规则**按钮。
   - 多表新增规则：支持一次性对多表或者多个字段设置监控规则。

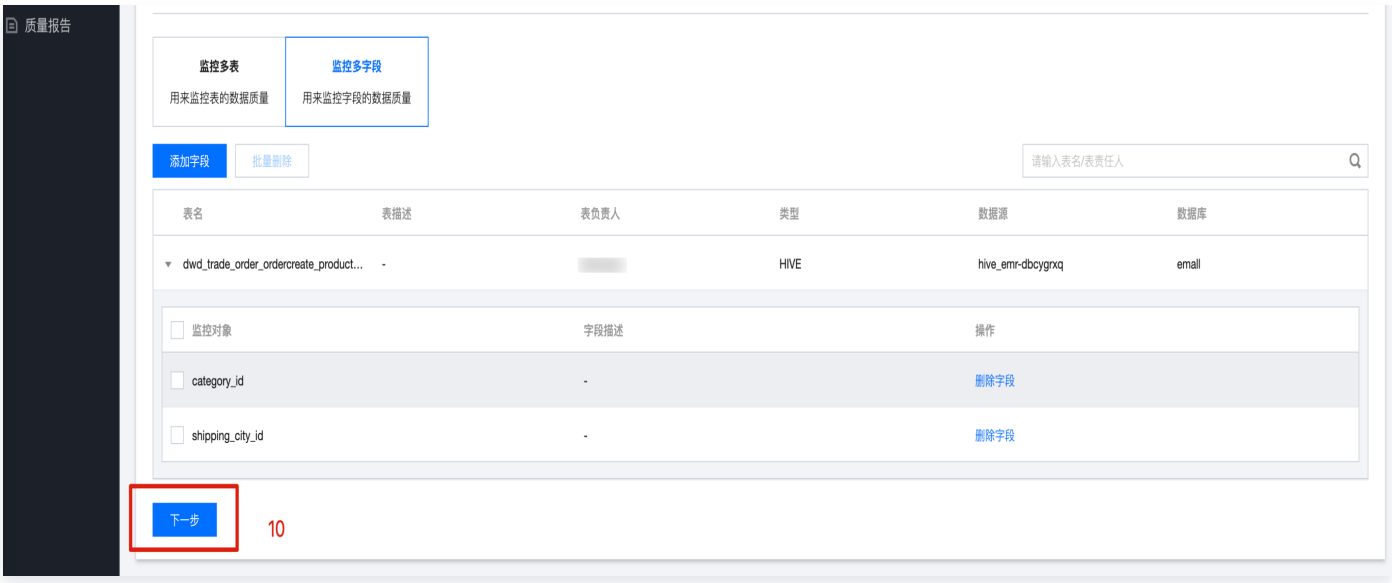


2. 单击**监控多字段**，再单击**添加字段**按钮，开始添加需要监控的字段。

3. 在添加字段页面中，选择添加方式为**手动添加**，选择**数据源**为 hive_emr-XXX，选择**数据库**为 hive 数据库 emall，选择完成后下方会刷新表与字段，选择表 dwd_trade_order_ordercreate_productsales 与对应的字段 amount、order_date，单击**保存**。

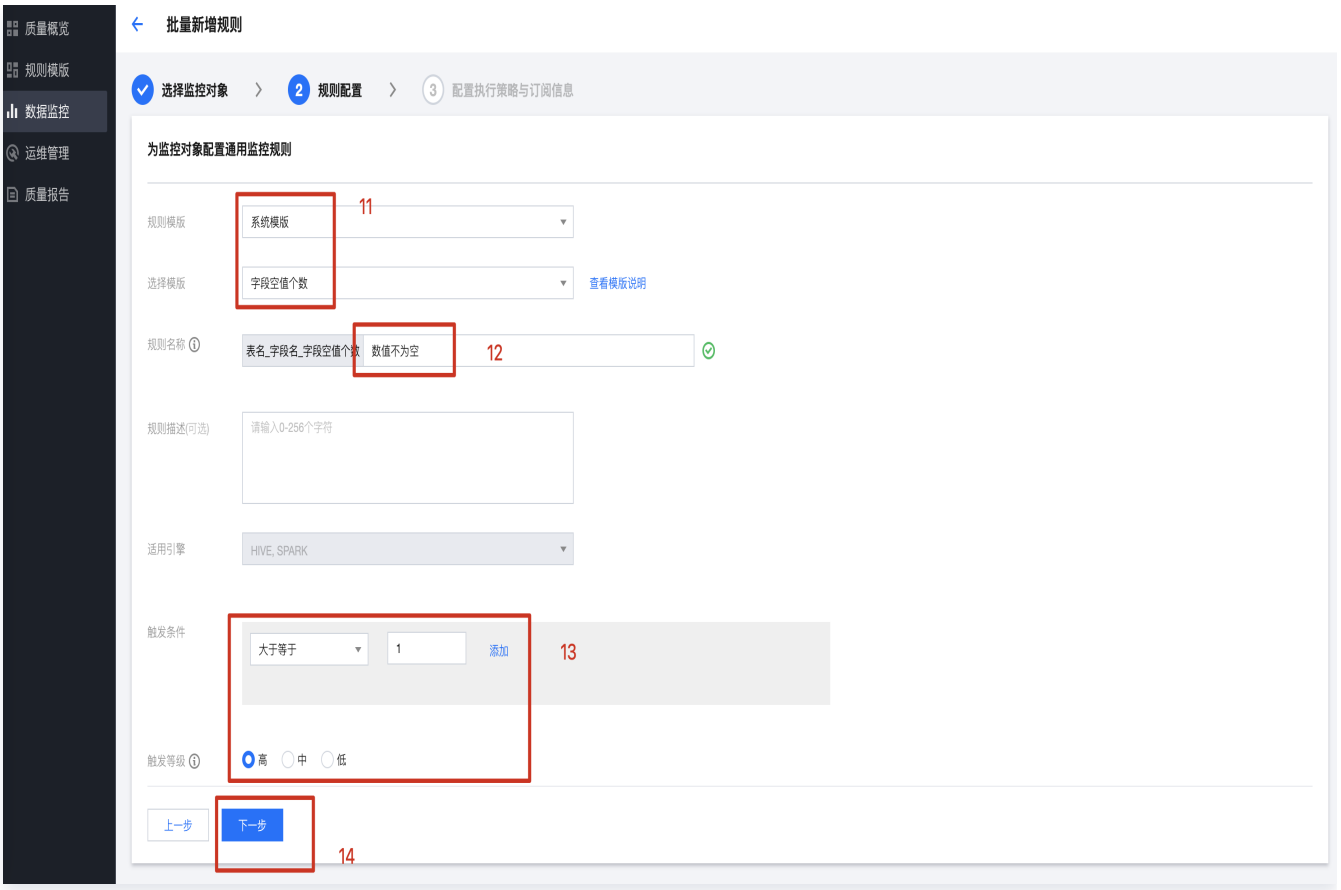


4. 保存成功后，页面会刷新显示选择的表与字段，单击**下一步**按钮，开始配置监控规则操作。



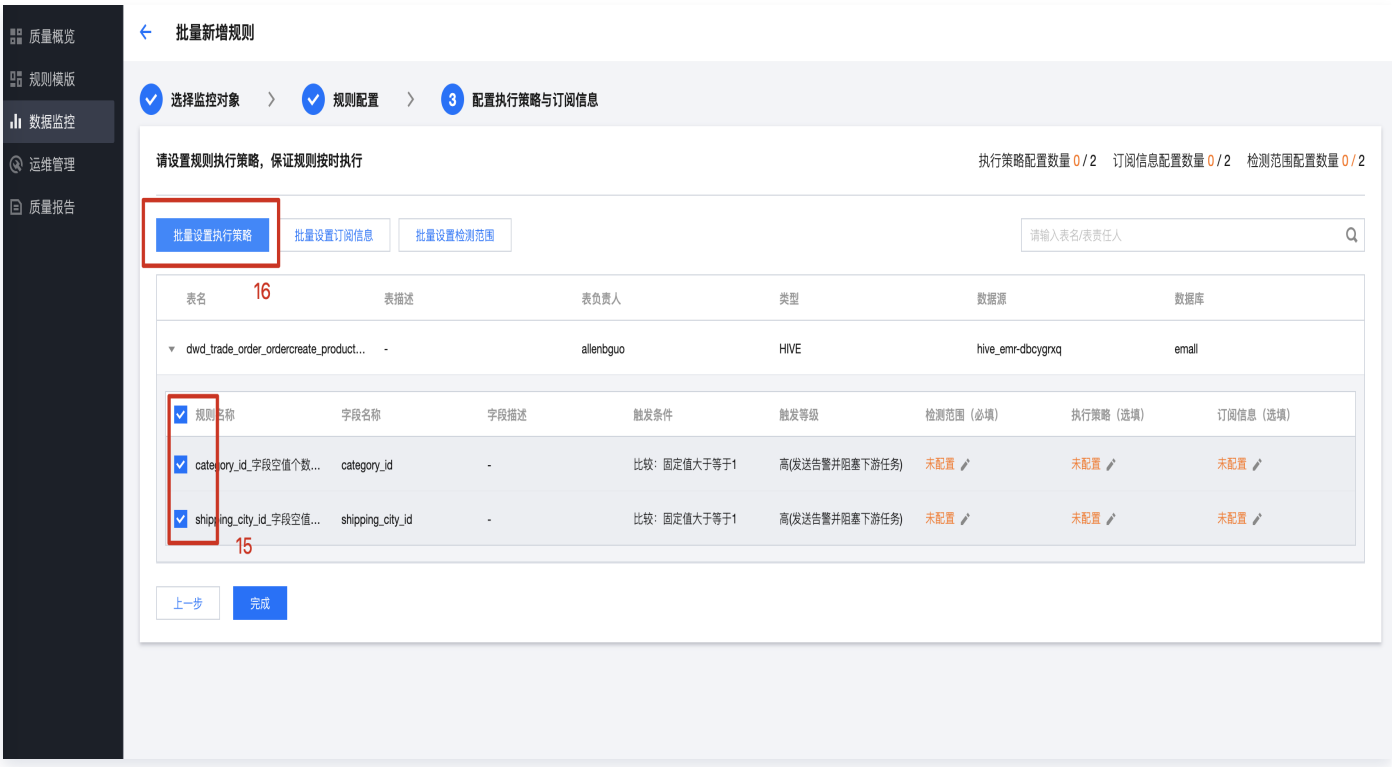
步骤2：配置监控规则

1. 为之前选择的表与字段配置监控规则，选择**规则模板**为系统模板，**选择模板**为字段空值个数，填写**规则名称**为数值不为空，设置**触发条件**为大于等于1，设置**触发等级**为高，设置完成后，单击**下一步**按钮，开始配置执行策略与订阅信息。
   - 规则模板：WeData 已经内置了50+系统模板，此处我们可以直接使用。
   - 选择模板：右侧可以**查看模板说明**。
   - 触发条件：表示当空值的个数大于等于1时，立即中断下游任务，并发送告警。



步骤3：设置执行策略

1. 单击**规则名称**，批量选择全部规则，再单击**批量设置执行策略**按钮，配置执行策略。

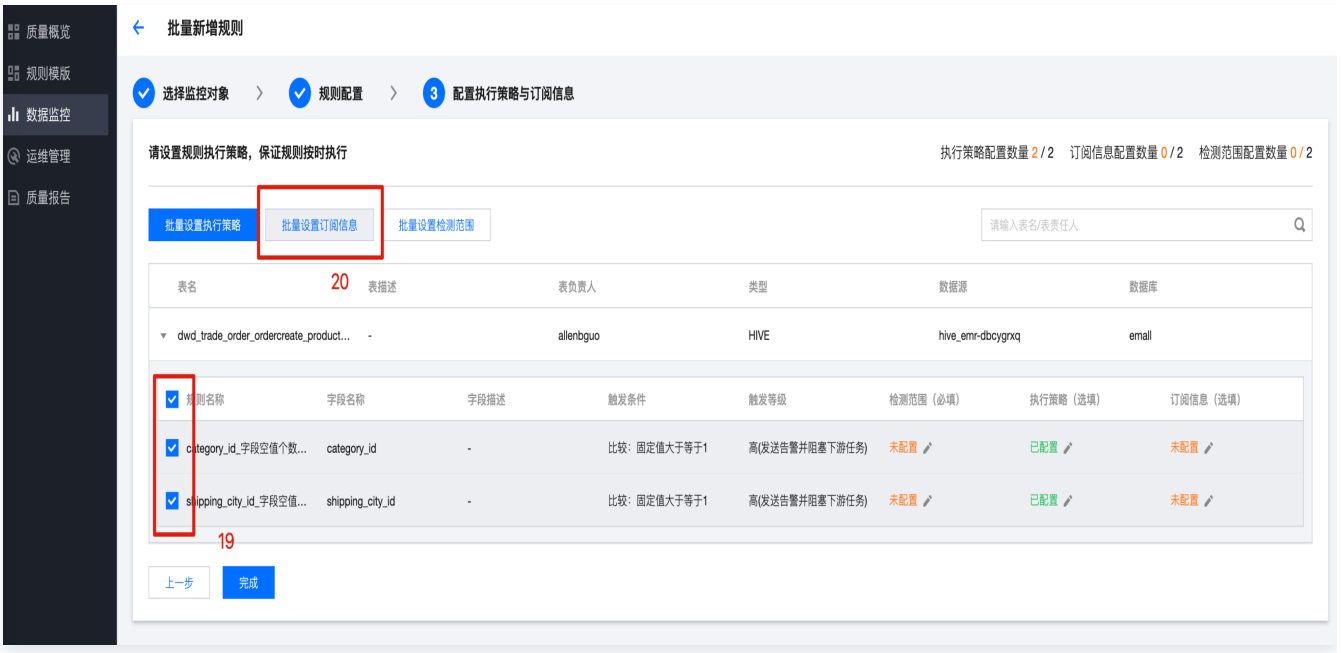


2. 选择**执行方式**为关联生产调度，**选择任务**为插入数据到明细表，设置完成后单击**保存**按钮。

- 关联生产调度：表示将质量任务与数据开发任务关联起来，只有被关联的任务执行完成后，才会执行这个质量监控任务。由于此处我们选择**插入数据到明细表**，即插入数据到明细表后，会立即检测数据的完整性。
- 执行引擎、计算资源、执行资源均与上文中的选择一致。
- 选择任务：即需要关联的数据开发任务。

步骤4：设置订阅通知

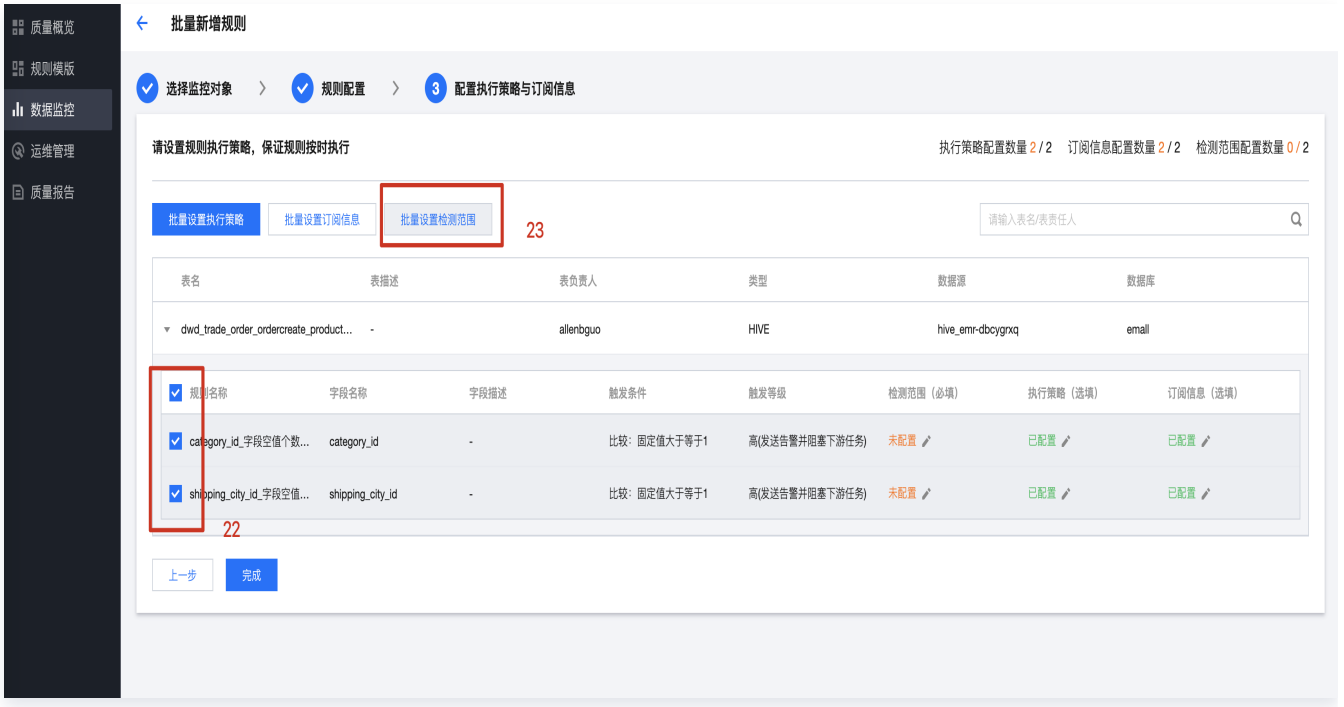
1. 单击**规则名称**，批量选择全部规则，再单击**批量设置订阅信息**按钮，配置订阅信息。
   - 设置订阅通知：设置当检测出现异常时，将使用何种方式发送消息提醒。



2. **订阅配置**选择**邮件**和**短信**，选择接收人为 XXX。

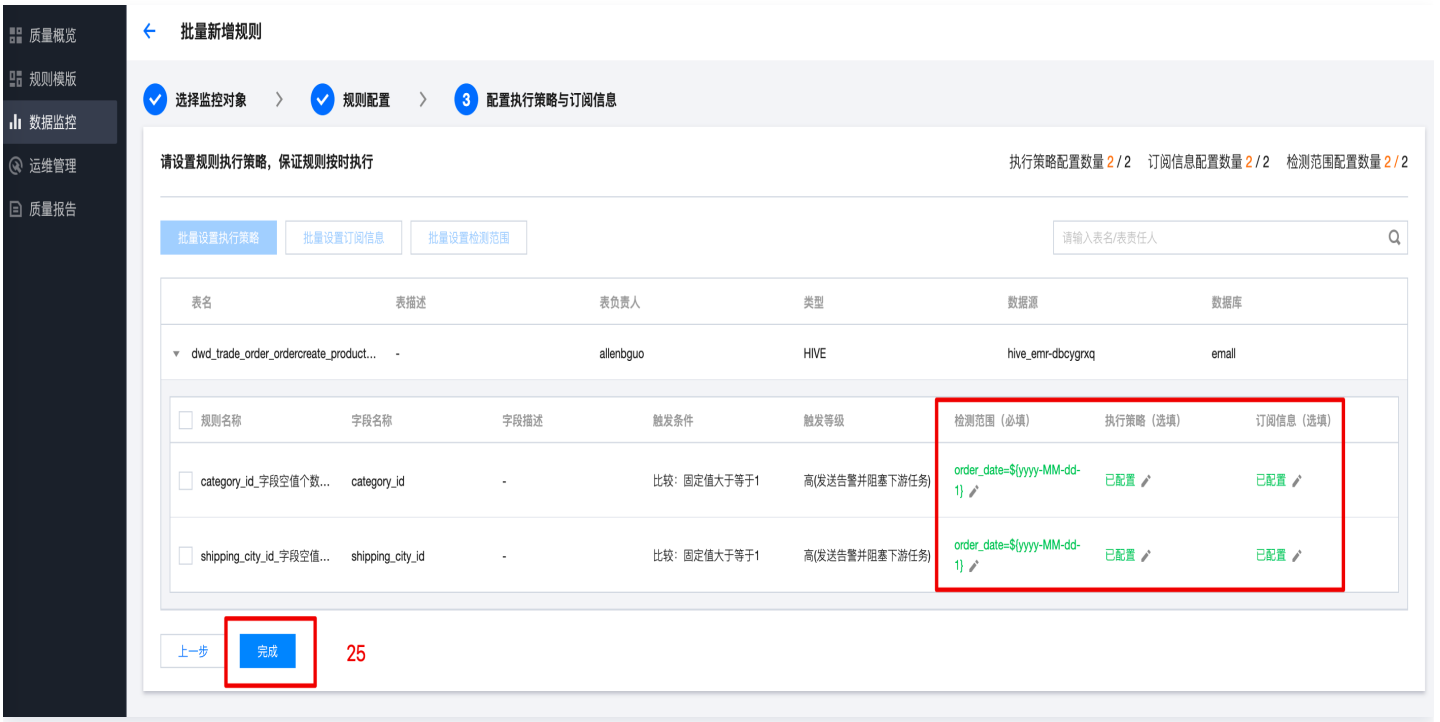
步骤5：设置检测范围

1. 单击**规则名称**，批量选择全部规则，再单击**批量设置检测范围**按钮，进行检测范围配置。
   - 设置检测范围：设置检测哪些范围的数据。



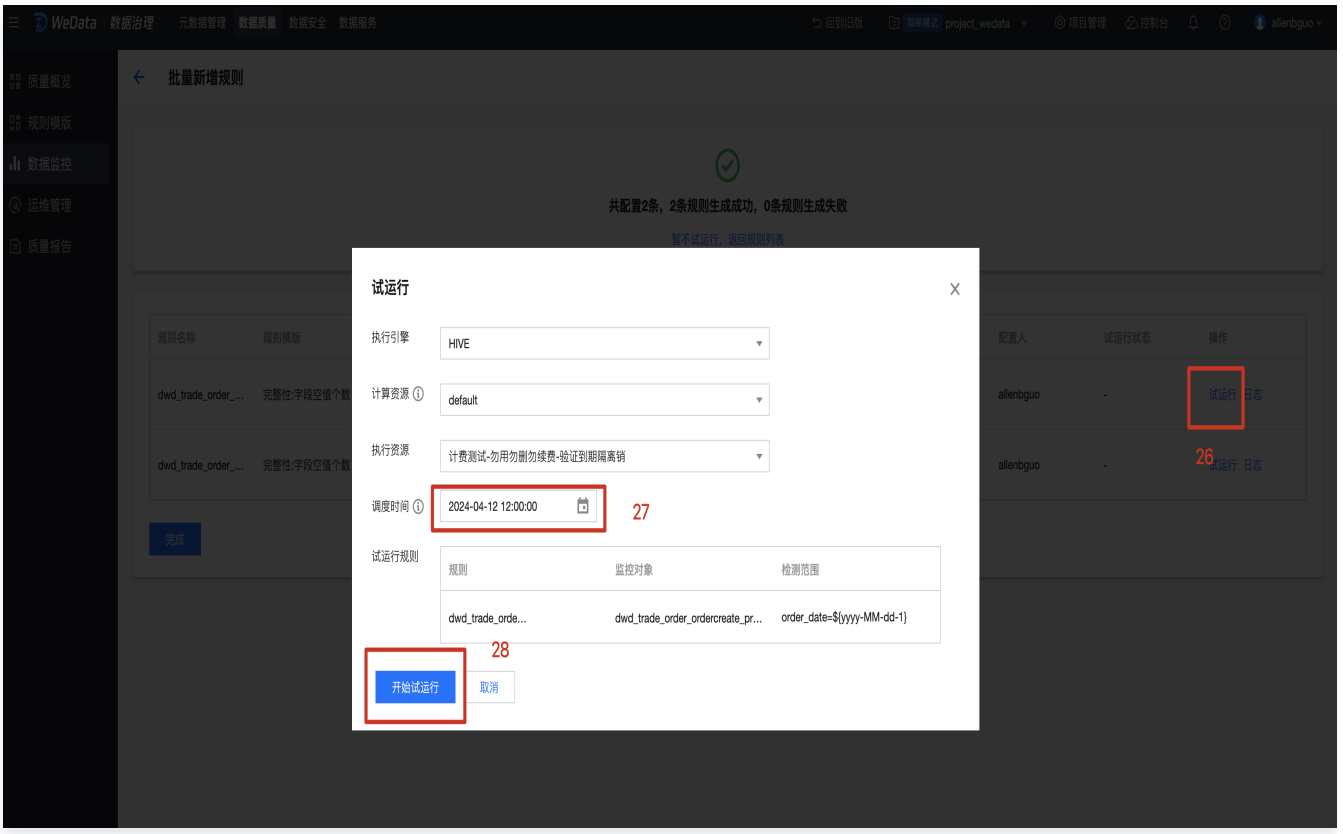
2. 设置**检测范围**为条件扫描，填入内容：order_date = ${yyyy-MM-dd-1}，单击**保存**按钮。
   - 条件扫描：根据所填入的条件，只检查每天新生产的数据，而不是每天都全量检查一遍。因为监控数据量越多对资源的消耗越大。
   - 右侧可以查看相关的参考信息。

3. 如下表示配置全部完成，单击**完成**按钮。



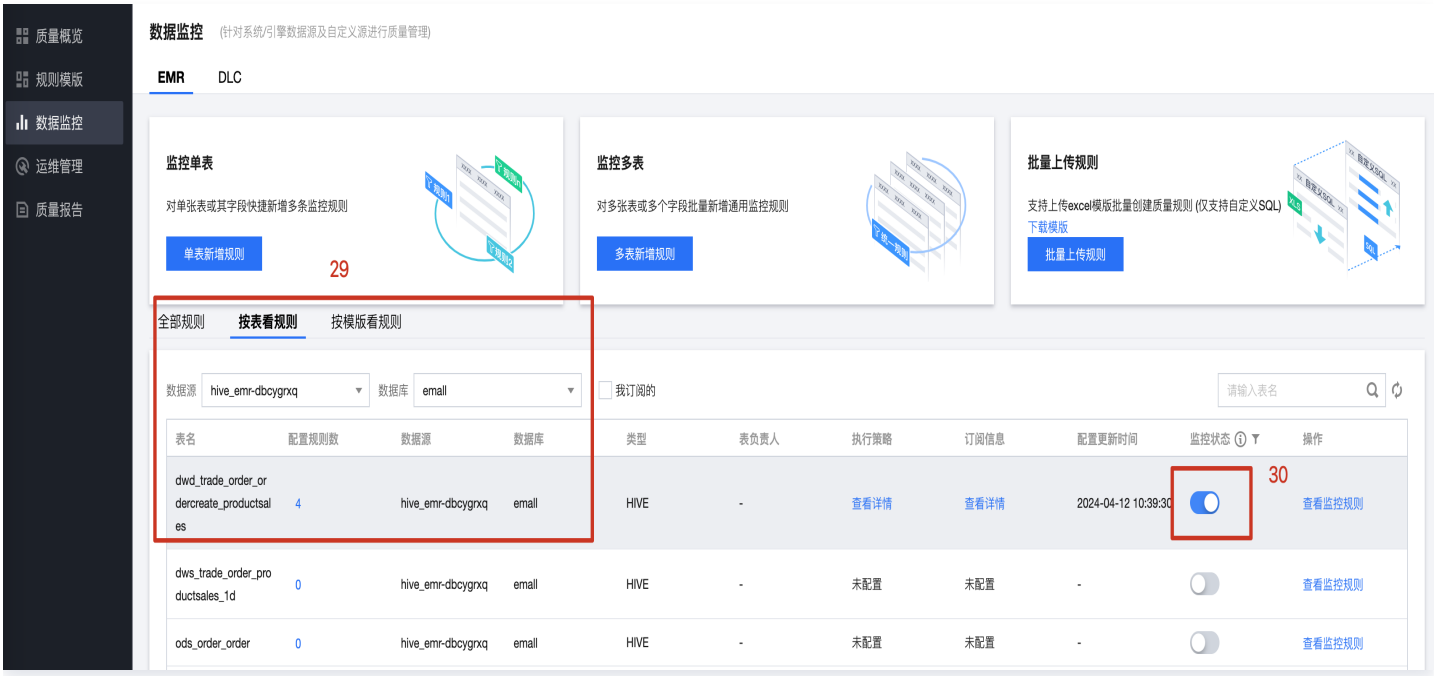
步骤6：任务试运行

1. 在页面右侧单击**试运行**按钮进行配置，选择调度时间为试运行时间，再单击**开始试运行**按钮。
   - 执行引擎、计算资源和执行资源与上文保持一致即可，**注意网络联通性**。
   - 试运行：在任务发布前，可以先试运行一次，检测任务是否存在错误。
   - 开始试运行：等待运行监测即可。
     - 试运行时需要保证明细表内是有数据的。如果在数据集成与数据开发过程中都操作了试运行，明细表应该是有数据的。



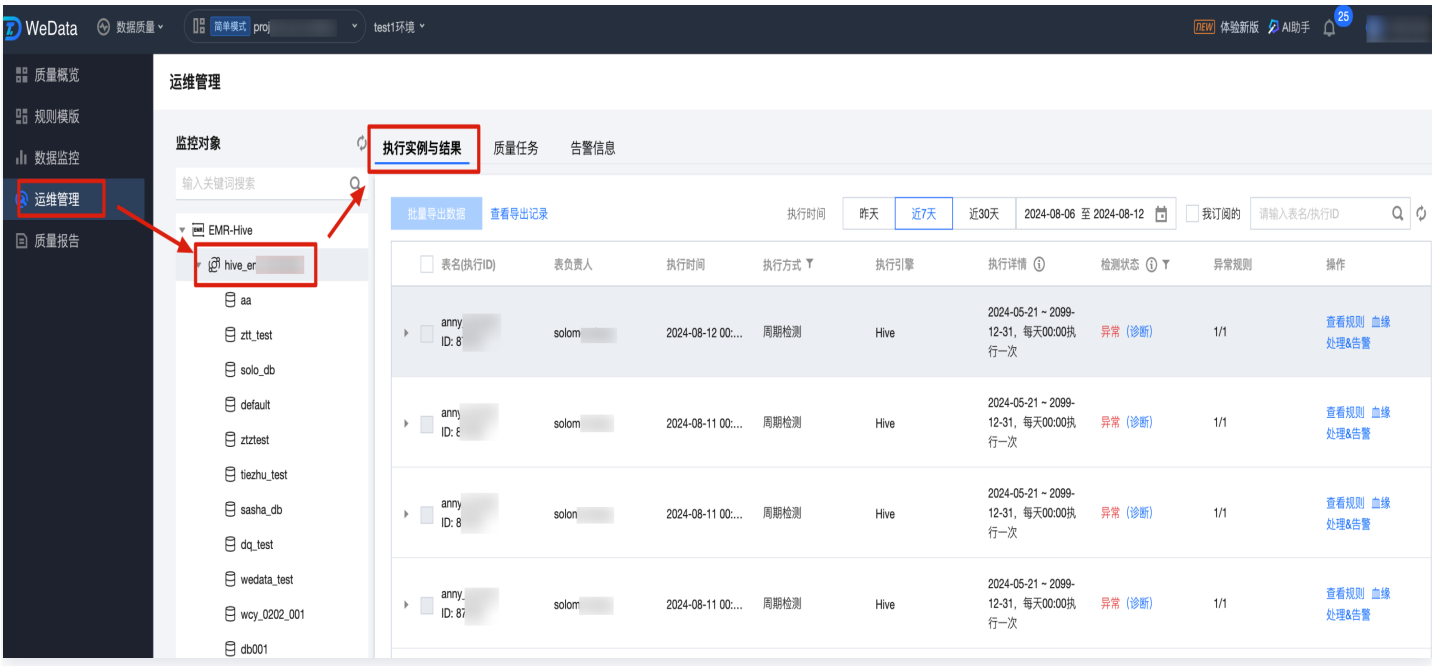
步骤7：任务发布

1. 回到 数据监控 页面，选择**按表看规则**页面，单击**监控状态**按钮开启监控。
   - 按表看规则：会根据数据源、数据库、数据表，筛选出规则。



质量监控任务运维

您可在运维管理中，查看质量监控任务的运行结果。


WeData
数据质量
简单模式 proj
test1环境
体验新版
AI助手
25
质量概览
规则模版
数据监控
运维管理
质量报告
运维管理
监控对象
执行实例与结果
质量任务
告警信息
输入关键词搜索
EMR-Hive
hive_er
aa
ztt_test
solo_db
default
ztztest
tiezhu_test
sasha_db
dq_test
wedata_test
wcy_0202_001
db001
批量导出数据
查看导出记录
执行时间
昨天
近7天
近30天
2024-08-06 至 2024-08-12
我订阅的
请输入表名/执行ID
表名(执行ID)
表负责人
执行时间
执行方式
执行引擎
执行详情
检测状态
异常规则
操作
solom
2024-08-12 00:...
周期检测
Hive
2024-05-21 ~ 2099-12-31，每天00:00执行一次
异常（诊断）
1/1
查看规则 血缘
处理&告警
2024-08-11 00:...